



ユーザーガイド

AWS CloudFormation



API バージョン 2010-05-15

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

AWS CloudFormation: ユーザーガイド

Copyright © 2026 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

CloudFormation とは？	1
インフラストラクチャ管理を簡略化	1
インフラストラクチャをすばやく複製	1
インフラストラクチャの制御や変更の追跡も簡単	2
CloudFormation の使用開始	2
関連情報	2
入門	3
CloudFormation の仕組み	3
主要なコンセプト	3
CloudFormation の仕組み	7
CloudFormation の使用を開始する方法	10
AWS アカウント へのサインアップ	10
AWS アカウント へのサインアップ	11
管理アクセスを持つユーザーを作成する	11
最初のスタックの作成	13
前提条件	14
コンソールを使用して CloudFormation スタックを作成する	14
スタックの作成をモニタリング	18
ウェブサーバーをテストする。	19
トラブルシューティング	19
クリーンアップ	20
次のステップ	21
ベストプラクティス	23
フィードバックループを短くしてデプロイ速度を向上させる	24
ライフサイクルと所有権によるスタックの整理	25
クロススタック参照を使用して、別のスタックがエクスポートした出力の値を返します。	26
マルチアカウントデプロイとマルチリージョンデプロイに CloudFormation StackSets を使用 する	26
すべてのリソースタイプのクォータを確認する	27
テンプレートを再利用して複数の環境にスタックを複製する	27
モジュールを使用してリソース構成を再利用する	27
Infrastructure as Code 慣行を導入する	28
テンプレートに認証情報を埋め込まない	28
AWS 固有のパラメータタイプを使用する	29

パラメータの制約の使用	29
疑似パラメータを使用して移植性を高める	29
AWS::CloudFormation::Init を使用して Amazon EC2 インスタンスにソフトウェアアプ リケーションをデプロイ	30
最新のヘルパースクリプトを使用する	30
テンプレートを使用する前に検証する	31
組織のポリシーに準拠するためにテンプレートを検証する	31
テンプレートオーサリングに YAML または JSON を使用する	31
包括的なタグ付け戦略を実装する	32
高度な変換にテンプレートマクロを活用する	32
CloudFormation ですべてのスタックリソースを管理する	33
スタックを更新する前に変更セットを作成する	33
スタックポリシーを使用してリソースを保護する	33
AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する	34
コードの確認とリビジョン管理を使用してテンプレートを管理する	34
Amazon EC2 インスタンスを定期的に更新	34
ドリフト検出を定期的に使用する	34
自動復旧のためのロールバックトリガーを設定する	35
効果的なスタックリファクタリング戦略を実装する	35
ライフサイクル管理に CloudFormation Hooks を使用する	36
IaC ジェネレーターを使用して既存のリソースからテンプレートを作成する	36
ビジュアルテンプレートの設計に AWS Infrastructure Composer を使用する	37
複雑なインフラストラクチャに AWS Cloud Development Kit (AWS CDK) を使用することを検 討する	37
IAM を使用したアクセス制御	38
最小特権の原則を適用する	38
機密性の高いパラメータをセキュア化する	38
AWS CloudFormation Guard を使用してポリシーをコードとして実装する	39
テンプレートの使用	40
テンプレートが保存される場所	41
テンプレートの検証	41
テンプレートの開始方法	2
CloudFormation テンプレートリファレンスを使用する予定	42
サンプルテンプレート	43
テンプレート形式	43
テンプレート構造	44

コメント	46
仕様	47
詳細はこちら	47
正規表現の使用	47
テンプレートセクション	48
Resources	50
Parameters	58
Outputs	72
Mappings	77
Metadata	85
Rules	91
Conditions	111
Transform	123
形式バージョン	124
Description	124
Infrastructure Composer	125
CloudFormation コンソールモードで Infrastructure Composer を使用する理由	126
Infrastructure Composer コンソールと異なる内容	126
AWS CloudFormation 言語サーバー	126
AWS CloudFormation 言語サーバーと統合している IDE	127
開始方法	127
IDE で CloudFormation プロジェクトを初期化する	133
オープンソース	135
サポートが必要ですか?	135
laC ジェネレーター	135
考慮事項	136
アクセス許可	137
よく使われるコマンド	137
テンプレートを AWS CDK に移行する	138
リソーススキャンを開始する	138
スキャンの概要を表示する	143
スキャンしたリソースからテンプレートを作成する	144
スキャンしたリソースからスタックを作成する	149
書き込み専用プロパティを解決する	150
他のサービスに格納されている値を取得する	156
一般的な考慮事項	156

Systems Manager のプレーンテキスト値を取得する	157
Systems Manager の SecureString を取得する	161
Secrets Manager の値を取得する	164
AWS 値の取得	169
構文	169
使用可能な擬似パラメータ	170
例	171
スタックの出力を取得する	176
スタック出力値のエクスポートとネストされたスタックの使用との比較	176
考慮事項	177
エクスポートされた出力値のリスト	177
エクスポートされた出力値をインポートするスタックのリスト	178
実行時に既存のリソースを指定する	179
概要:	180
例	181
考慮事項	184
サポートされている AWS 固有のパラメータタイプ	185
サポートされている Systems Manager パラメータタイプ	187
サポートされていない Systems Manager パラメータタイプ	188
チュートリアル	188
別の CloudFormation スタックのリソース出力を参照する	189
Amazon EC2 にアプリケーションをデプロイする	194
スタックを更新する	204
スケールおよび負荷分散されたアプリケーションを作成する	221
別のアカウントで VPC とピア接続	235
ECS ブルー/グリーンデプロイの実行	245
テンプレートスニペット	269
General	270
Auto scaling	280
AWS 請求コンソール	313
CloudFormation	316
CloudFront	323
CloudWatch	336
CloudWatch Logs	342
DynamoDB	376
Amazon EC2	380

Amazon ECS	418
Amazon EFS	441
Elastic Beanstalk	466
エラスティックロードバランシング	471
IAM	476
AWS Lambda	494
Amazon Redshift	500
Amazon RDS	510
Route 53	519
Amazon S3	527
Amazon SNS	536
Amazon SQS	537
Amazon Timestream	537
Windows ベースのスタック	566
Windows スタックのブートストラップ	568
CloudFormation 提供のリソースタイプを使用する	575
カスタムリソース	577
マクロのテンプレート	622
ネストされたスタック	639
待機条件	646
モジュールを使用して再利用可能なリソース設定の作成	652
モジュールを使用するときの考慮事項	653
モジュールのバージョニングの概要	654
テンプレートでモジュールを使用する	656
パラメータを使用してモジュール値を指定する	657
リファレンスモジュールリソース	660
スタックの作成と管理	662
スタックを管理するためのインターフェイス	663
Stack を作成する	664
スタックの作成	665
スタックオプションを設定する	668
スタックの設定をプレビューする	670
スタック情報を表示する	671
スタックテンプレートを更新する	674
スタックリソースの更新動作を理解する	676
変更セットを使用してスタックを更新する	677

変更セットを作成する	679
変更セットを表示する	685
ドリフト対応変更セット	694
変更セットを実行する	700
変更セットを削除する	702
変更セットのサンプル	704
ネストされたスタックの変更セット	716
スタックデプロイを検証する	723
デプロイ前検証の仕組み	723
考慮事項	724
前提条件	724
スタックデプロイを検証する (コンソール)	725
スタックデプロイを検証する (AWS CLI)	726
検証タイプ	727
リソースの制限	728
スタックを直接更新する	742
スタック更新をキャンセルする	746
スタックの更新をキャンセルするには (コンソール)	746
スタック更新をキャンセルするには (AWS CLI)	746
スタックを削除する	747
関連リソース	748
削除済みのスタックを表示する	749
スタックの進行状況を監視する	749
スタックイベントを表示する	750
オペレーション別にスタックイベントを表示する	758
スタックデプロイタイムライングラフを表示する	762
スタック作成イベントを理解する	765
スタック更新を監視する	767
更新のロールバックを続ける	769
スタックが失敗する原因を判断する	773
スタック障害オプション	775
アラーム違反時にスタックをロールバックする	785
スタックの作成中または更新中にロールバックトリガーを追加する	786
変更セットにロールバックトリガーを追加する	787
スタックのロールバックトリガーを表示する	787
変更セットのロールバックトリガーを表示する	787

スタックとリソースへのアンマネージド型設定変更を検出する	788
ドリフトとは	788
ドリフト検出ステータスコード	790
ドリフトを検出する際の考慮事項	792
CloudFormation スタック全体のドリフトを検出する	794
個々のスタックリソースのドリフトを検出	800
インポートオペレーションでドリフトを解決する	804
AWS リソースをインポートする	814
AWS リソースを手動でインポートする	815
AWS リソースを自動でインポートする	853
インポートオペレーションを元に戻す	855
スタックリファクタリング	860
スタックリファクタリングの仕組み	860
スタックリファクタリングに関する考慮事項	861
前提条件	862
スタックをリファクタリングする (コンソール)	862
スタックをリファクタリングする (AWS CLI)	864
リソースの制限	868
リソースタイプのサポート	873
クイック作成リンクを使用して CloudFormation スタックを作成する	1091
URL 形式	1092
例	1093
クイック作成リンクを使用してスタックを作成する	1094
AWS CLI および PowerShell のコマンド例	1095
CancelUpdateStack	1096
ContinueUpdateRollback	1097
CreateStack	1098
デプロイ	1102
DeleteStack	1103
DescribeStackEvents	1104
DescribeStackResource	1106
DescribeStackResources	1107
DescribeStacks	1111
GetTemplate	1115
ListStackResources	1116
ListStacks	1118

UpdateStack	1120
ValidateTemplate	1124
ローカルアーティファクトを S3 バケットにアップロードする	1127
StackSets を使用したスタックの管理	1130
StackSets の概念	1131
管理者アカウントおよびターゲットアカウント	1132
CloudFormation StackSets	1132
StackSets のアクセス許可モデル	1132
スタックインスタンス	1133
StackSet オペレーション	1134
StackSet オペレーションのオプション	1135
タグ	1137
StackSets ステータスコード	1138
スタックインスタンスのステータスコード	1139
前提条件	1140
AWS リージョン がデフォルトで無効になっていることを有効にする	1141
セルフマネージドのアクセス許可の付与	1142
信頼されたアクセスをアクティブ化する	1159
サンプルテンプレートを使用して開始する	1163
前提条件	1164
コンソールからサンプルテンプレートを使用して StackSet を作成する	1165
StackSet の作成をモニタリングする	1168
StackSet の結果を表示する	1168
StackSet を更新する	1168
スタックを StackSet に追加する	1169
クリーンアップ	1170
次のステップ	1172
StackSets を作成する (セルフマネージド許可)	1172
セルフマネージド型のアクセス許可を持つ StackSet の作成 (コンソール)	1173
セルフマネージド型のアクセス許可を持つ StackSet の作成 (AWS CLI)	1175
StackSets を作成する (サービスマネージド許可)	1176
考慮事項	1177
サービスマネージド型のアクセス許可を持つ StackSet の作成 (コンソール)	1178
サービスマネージド型のアクセス許可を持つ StackSet 作成 (AWS CLI)	1180
自動デプロイを有効または無効にする	1185
自動デプロイの仕組み	1185

考慮事項	1185
自動デプロイを有効または無効にする (コンソール)	1186
自動デプロイを有効または無効にする (AWS CLI)	1186
StackSets を更新する	1187
StackSet を更新する (コンソール)	1187
StackSet を更新する (AWS CLI)	1190
StackSets にスタックを追加する	1191
StackSet にスタックを追加する (コンソール)	1191
スタックを StackSet に追加する (AWS CLI)	1193
スタックのパラメータを上書きする	1194
スタックのパラメータを上書きする (コンソール)	1195
スタックのパラメータを上書きする (AWS CLI)	1197
StackSets からスタックを削除する	1198
StackSet からスタックを削除する (コンソール)	1198
StackSet からスタックを削除する (AWS CLI)	1200
StackSets を削除する	1202
StackSet を削除する (コンソール)	1202
StackSet を削除する (AWS CLI)	1202
Delete Service Roles (オプション)	1203
ターゲットアカウントゲート	1204
要件	1204
Lambda 関数を作成するための CloudFormation テンプレート	1205
同時実行モードの選択	1206
各同時実行モードの仕組み	1206
デプロイ速度に基づいて、厳密な耐障害性とソフト耐障害性のどちらかを選択します。 ..	1209
同時実行モードの選択 (コンソール)	1211
同時実行モードの選択 (AWS CLI)	1211
StackSets でのドリフトの検出	1212
CloudFormation が StackSet でドリフト検出を実行する方法	1213
StackSet のドリフトを検出する (コンソール)	1213
StackSet のドリフトを検出する (AWS CLI)	1215
StackSet でのドリフト検出の停止	1222
スタックを StackSets にインポートする	1222
セルフマネージド型のスタックインポート	1223
サービスマネージド型のスタックインポート	1227
スタックのインポートを元に戻す	1232

ベストプラクティス	1232
テンプレートの定義	1233
StackSet にスタックを作成または追加する	1233
StackSet でスタックを更新する	1234
サンプルテンプレート	1234
トラブルシューティング	1236
スタックオペレーション失敗の一般的な原因	1236
失敗したスタック作成または更新オペレーションを再試行する	1237
スタックインスタンスの削除に失敗する場合	1238
スタックのインポートオペレーションが失敗する	1238
StackSets オペレーションのスタックインスタンスの失敗回数	1238
スタックと Git ソースコードの同期	1245
Git 同期の仕組み	1246
Git 同期の仕組み	1246
プルリクエストに関するコメント	1247
スタックデプロイファイル	1248
CloudFormation テンプレートファイル	1249
テンプレート定義リポジトリ	1249
前提条件	1249
Git リポジトリ	1250
CloudFormation テンプレート	1250
Git 同期サービスロール	1250
コンソールユーザーの IAM アクセス権限	1253
リポジトリソースコードからスタックを作成する	1256
リポジトリソースコードからスタックを作成する	1257
Git リポジトリからスタックを更新する	1259
プルリクエストに関するコメントを有効にする	1260
ステータスダッシュボード	1260
Git 同期のステータス	1260
最新の同期イベント	1261
サポートされるスタック状態	1262
CloudFormation レジストリによる拡張機能の管理	1266
関連ドキュメント	1266
概念	1267
拡張タイプ	1267
パブリック拡張	1268

アクティブ化した拡張機能	1268
使用可能でアクティブ化された拡張を表示する	1269
パブリック拡張	1270
IAM ロール	1271
拡張の新しいバージョンを自動的に使用する	1273
拡張を参照するエイリアスを使用する	1274
パブリック拡張を操作するためによく使用される AWS CLI コマンド	1274
パブリック拡張を有効にする	1275
パブリック拡張を更新する	1279
パブリック拡張を無効にする	1280
プライベート拡張	1282
IAM アクセス許可	1282
プライベート拡張機能を操作するためによく使用される AWS CLI コマンド	1282
プライベート拡張を登録する	1283
プライベート拡張の登録解除	1285
拡張機能の構成データを編集する	1287
動的リファレンスを使用するために必要なアクセス許可	1288
拡張機能の構成データを編集する (コンソール)	1288
拡張機能の構成データを編集する (AWS CLI)	1289
AWS Config でリソースタイプを記録する	1290
構成項目に機密のプロパティが記録されないようにする	1291
継続的デリバリー	1292
チュートリアル: テストおよび本稼働スタック用のパイプラインを構築する	1292
前提条件	1293
チュートリアルの概要	1293
ステップ 1: アーティファクトを編集し、S3 バケットにアップロードする	1294
ステップ 2: パイプラインスタックを作成する	1296
ステップ 3: WordPress スタックの表示	1301
ステップ 4: リソースをクリーンアップする	1302
関連情報	1303
構成プロパティのリファレンス	1304
構成プロパティ (コンソール)	1304
構成プロパティ (JSON オブジェクト)	1307
関連情報	1312
CloudFormation アーティファクト	1312
スタックテンプレートファイル	1313

テンプレート構成ファイル	1313
関連情報	1314
CodePipeline パイプラインでのパラメーターオーバーライド関数の使用	1315
Fn::GetArtifactAtt	1315
Fn::GetParam	1317
関連情報	1320
テンプレートリファレンスガイド	1321
セキュリティ	1322
スタックの削除を防止する	1323
スタックの削除保護を変更するユーザーの制御	1325
スタックのリソースが更新されないようにする	1325
スタックポリシーの例	1326
スタックポリシーの定義	1327
スタックポリシーの設定	1331
保護されたリソースの更新	1332
スタックポリシーの変更	1336
スタックポリシーのその他の例	1337
データ保護	1340
保管中の暗号化	1341
転送中の暗号化	1342
インターネットトラフィックのプライバシー	1342
IAM によるアクセスを制御する	1342
CloudFormation の IAM アイデンティティベースのポリシーの定義	1342
CloudFormation テンプレートでの IAM リソースの承認	1351
Amazon EC2 インスタンスで実行されるアプリケーションの認証情報の管理	1353
一時的なアクセス権の付与 (フェデレーションアクセス)	1353
アイデンティティベースのポリシーの例	1355
AWS マネージドポリシー	1363
CloudFormation サービスロール	1366
サービス間の混乱した代理の防止	1367
FAS リクエストとアクセス許可の評価	1369
API コールのログ作成	1371
CloudTrail での CloudFormation 情報	1372
CloudFormation ログファイルエントリについて	1373
インフラストラクチャセキュリティ	1377
耐障害性	1377

コンプライアンス検証	1378
構成と脆弱性の分析	1378
セキュリティに関するベストプラクティス	1378
IAM を使用したアクセス制御	1379
テンプレートに認証情報を埋め込まない	1379
AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する	1380
AWS PrivateLink	1380
CloudFormation VPC エンドポイントに関する考慮事項	1380
CloudFormation のインターフェイス VPC エンドポイントの作成	1381
CloudFormation 用の VPC エンドポイントポリシーの作成	1382
EventBridge によるモニタリング	1384
CloudFormation および Git 同期イベントの概要	1385
アクセス許可	1387
カスタムイベントパターンの作成	1387
イベントの詳細リファレンス	1388
リソースステータスの変更	1389
スタックのステータスの変更	1392
ドリフト検出のステータスの変更	1394
StackSet ステータスの変更	1396
StackSet スタックインスタンスのステータス変更	1398
StackSet オペレーションステータスの変更	1401
リポジトリ同期ステータスの変更	1403
リソース同期ステータスの変更	1405
クォータ	1409
機能を利用できるリージョン	1417
StackSets とマクロ	1417
トラブルシューティング	1418
トラブルシューティングガイド	1418
エラーのトラブルシューティング	1419
スタックの削除の失敗	1420
依存関係のエラー	1421
AWS Config と AWS Systems Manager の競合	1421
リストを渡す際のパラメーター解析エラー	1422
IAM アクセス権限の不足	1422
値が無効またはリソースのプロパティがサポートされていない	1422
リソースクォータ	1422

ネストされたスタックのロールバック失敗	1423
実行する更新がない	1423
スタックの作成、更新、または削除オペレーションの際、リソースが安定しない	1424
セキュリティグループが VPC 内に存在しない	1424
更新のロールバックの失敗	1425
待機条件が Amazon EC2 インスタンスから所定の数のシグナルを受け取らなかった	1426
リソースはスタックから削除されましたが、削除されてはいません。	1427
Support へのお問い合わせ	1428
Amazon Q を使用したトラブルシューティング	1429
機能	1429
仕組み	1430
料金	1430
リージョナルな可用性	1430
制限事項	1430
ドキュメント履歴	1431
アーカイブされた更新	1458

CloudFormation とは？

CloudFormation は AWS リソースのモデル化およびセットアップに役立つサービスです。リソース管理に割く時間を減らし、AWS で実行するアプリケーションにさらに注力できるようになります。使用するすべての AWS リソース (Amazon EC2 インスタンスや Amazon RDS DB インスタンスなど) を記述するテンプレートを作成すれば、CloudFormation がお客様に代わってこれらのリソースのプロビジョニングや設定を受け持ちます。AWS リソースを個別に作成、設計して、それぞれの依存関係を考える必要はありません。CloudFormation がこれを処理します。次のシナリオは CloudFormation がどのように役立つかを示します。

インフラストラクチャ管理を簡略化

バックエンドデータベースを含むスケーラブルなウェブアプリケーションで、Auto Scaling グループ、Elastic Load Balancing ロードバランサー、Amazon Relational Database Service データベースインスタンスを使用している場合があります。個々のサービスを使用してこれらのリソースをプロビジョニングする必要があるでしょう。また、リソースを作成した後、一緒に動作するように設定する必要があるでしょう。このような作業は、まだアプリケーションを運用していないうちから、複雑な手間と時間がかかります。

代わりに、CloudFormation テンプレートを作成したり、既存のテンプレートを変更したりできます。テンプレートはすべてのリソースおよびプロパティについて記述します。テンプレートを使用して CloudFormation スタックを作成すると、CloudFormation が Auto Scaling グループ、ロードバランサー、およびデータベースをプロビジョニングします。スタックが正常に作成されると、AWS リソースの利用を開始できます。スタックの削除も同様に簡単です。スタックのすべてのリソースを削除できます。CloudFormation を使用すると、リソースのコレクションを単一のユニットとして簡単に管理できます。

インフラストラクチャをすばやく複製

アプリケーションに追加の可用性が必要な場合は、複数のリージョンに複製して、1 つのリージョンが使用できなくなった場合でも他のリージョンでアプリケーションを使用できるようにします。アプリケーションの複製における課題は、リソースも複製する必要があることです。アプリケーションに必要なすべてのリソースを記録する必要があるだけでなく、それぞれのリージョンでそれらのリソースをプロビジョニングし、設定する必要があります。

CloudFormation テンプレートを再利用して、一貫性のある繰り返し可能な方法でリソースを作成します。テンプレートを再利用するには、一度だけリソースを記述します。これにより、複数のリージョンで同じリソースを何度でもプロビジョニングできます。

インフラストラクチャの制御や変更の追跡も簡単

場合によっては、基盤となるリソースを段階的にアップグレードすることがあります。たとえば、Auto Scaling グループのインスタンスの最大数を減らすために、Auto Scaling 起動インスタンスをより効率のいいインスタンスタイプに変更する場合があります。更新が完了した後で問題が発生した場合、インフラストラクチャを最初の設定にロールバックする必要がある場合があります。これを手動で行うには、変更したリソースを覚えておく必要があるだけでなく、最初の設定を知っておく必要があります。

CloudFormation でインフラストラクチャをプロビジョニングすると、CloudFormation テンプレートにプロビジョニングされたリソースやその設定が正確に記述されます。これらのテンプレートはテキストファイルであるため、テンプレート間の違いを追跡するだけでインフラストラクチャに対する変更を追跡できます。これは開発者がソースコードの変更を管理する方法に似ています。たとえば、バージョン管理システムをテンプレートに使用して、変更内容、変更者、変更日時を正確に把握できます。インフラストラクチャの変更を元に戻す必要がある場合はいつでも、テンプレートの以前のバージョンを使用できます。

CloudFormation の使用開始

CloudFormation は、CloudFormation [コンソール](#)、[API](#)、[AWS CLI](#)、[AWS SDK](#) の、およびいくつかの統合を通じて使用できます。

CloudFormation の概要については、「[CloudFormation の仕組み](#)」を参照してください。

CloudFormation の使用を開始するには、「[最初のスタックの作成](#)」を参照してください。

関連情報

CloudFormation の詳細については、このユーザーガイドおよび以下のリソースを参照してください。

- 製品の詳細とよくある質問については、「[AWS CloudFormation 製品ページ](#)」を参照してください。
- 料金情報については、「[AWS CloudFormation の料金](#)」を参照してください。

CloudFormation の使用開始

サンプルテンプレートからスタックを作成することで、AWS マネジメントコンソール を通じて CloudFormation の使用を開始できます。これにより、スタック作成の基本を学ぶことができます。テンプレートは、スタック内のすべてのリソースを定義するテキストファイルです。スタックは CloudFormation テンプレートのデプロイです。1 つのテンプレートで複数のスタックを作成できます。各スタックには、1 のユニットとして管理できる AWS リソースのコレクションが含まれています。

CloudFormation は無料サービスです。ただし、スタックに含める AWS リソースに対しては、それぞれの現在の料金が課金されます AWS 料金に関する詳細は、<http://aws.amazon.com> で各製品の詳細ページを参照してください。

トピック

- [CloudFormation の仕組み](#)
- [AWS アカウント へのサインアップ](#)
- [最初のスタックの作成](#)

CloudFormation の仕組み

このトピックでは、CloudFormation の仕組みについて説明し、使用時に知っておくべき重要な概念を紹介します。

トピック

- [主要なコンセプト](#)
- [CloudFormation の仕組み](#)
- [CloudFormation の使用を開始する方法](#)

主要なコンセプト

CloudFormation を使用する際には、テンプレートとスタックの作業を行います。テンプレートは、AWS リソースとそのプロパティを記述するために作成します。スタックを作成するたびに、CloudFormation はテンプレートに記述されているリソースをプロビジョニングします。

トピック

- [テンプレート](#)
- [スタック](#)
- [変更セット](#)

テンプレート

CloudFormation テンプレートは YAML または JSON 形式のテキストファイルです。これらのファイルは、.yaml、.json、.template、.txt などの拡張子を使用して保存できます。CloudFormation は、AWS リソースを構築するブループリントとしてこれらのテンプレートを使用します。たとえば、テンプレートには、インスタンスタイプ、AMI ID、ブロックデバイスマッピング、Amazon EC2 キーペアの名前など、Amazon EC2 インスタンスの情報を記述できます。スタックを作成する際には、CloudFormation がテンプレートの記述内容を作成するために使用するテンプレートも指定します。

例えば、以下のテンプレートを指定してスタックを作成した場合、CloudFormation は ami-0ff8a91507f77f867 AMI ID、t2.micro インスタンスタイプ、testkey キーペアの名前、および Amazon EBS ボリュームを持つインスタンスをプロビジョニングします。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

JSON

```
{
```



```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "A sample template",
"Resources": {
  "MyEC2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "ami-0ff8a91507f77f867",
      "InstanceType": "t2.micro",
      "KeyName": "testkey",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sdm",
          "Ebs": {
            "VolumeType": "io1",
            "Iops": 200,
            "DeleteOnTermination": false,
            "VolumeSize": 20
          }
        }
      ]
    }
  }
}
```

また単独のテンプレートに複数のリソースを指定し、これらのリソースが連携するように設定できます。例えば、Elastic IP アドレス (EIP) を含めるように、前のテンプレートを修正して、次の例のように Amazon EC2 インスタンスと関連付けできます。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
```

```
        VolumeType: io1
        Iops: 200
        DeleteOnTermination: false
        VolumeSize: 20
MyEIP:
  Type: 'AWS::EC2::EIP'
  Properties:
    InstanceId: !Ref MyEC2Instance
```

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "A sample template",
  "Resources": {
    "MyEC2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": "ami-0ff8a91507f77f867",
        "InstanceType": "t2.micro",
        "KeyName": "testkey",
        "BlockDeviceMappings": [
          {
            "DeviceName": "/dev/sdm",
            "Ebs": {
              "VolumeType": "io1",
              "Iops": 200,
              "DeleteOnTermination": false,
              "VolumeSize": 20
            }
          }
        ]
      }
    },
    "MyEIP": {
      "Type": "AWS::EC2::EIP",
      "Properties": {
        "InstanceId": {
          "Ref": "MyEC2Instance"
        }
      }
    }
  }
}
```

```
}
```

上記のテンプレートは単一の Amazon EC2 インスタンスについて記述したものです。CloudFormation テンプレートの追加機能により、複雑なリソースセットを構築して、それらのテンプレートを複数のコンテキストで再利用することができます。例えば、CloudFormation スタックの作成時に値を設定できる入力パラメータを追加できます。つまり、インスタンスタイプなどの値を、テンプレート作成時ではなく、スタック作成時に指定することができ、さまざまな状況でテンプレートを再利用しやすくなります。

スタック

CloudFormation を使用する際、関連リソースはスタックと呼ばれる単一のユニットとして管理します。スタックを作成、更新、削除することで、リソースのコレクションを作成、更新、削除します。スタック内のすべてのリソースは、スタックの CloudFormation テンプレートで定義されます。Auto Scaling グループ、Elastic Load Balancing ロードバランサー、および Amazon Relational Database Service (Amazon RDS) データベースインスタンスを含むテンプレートを作成したとします。これらのリソースを作成するには、作成したテンプレートを送信することによってスタックを作成します。これにより、CloudFormation によってすべてのリソースがプロビジョニングされます。

変更セット

スタックで実行中のリソースに変更を加える必要がある場合は、スタックを更新します。リソースに変更を加える前に、変更案の概要である変更セットを生成できます。変更セットで、変更が実行中のリソース、特に重要なリソースに与える可能性のある影響を、実装前に確認できます。

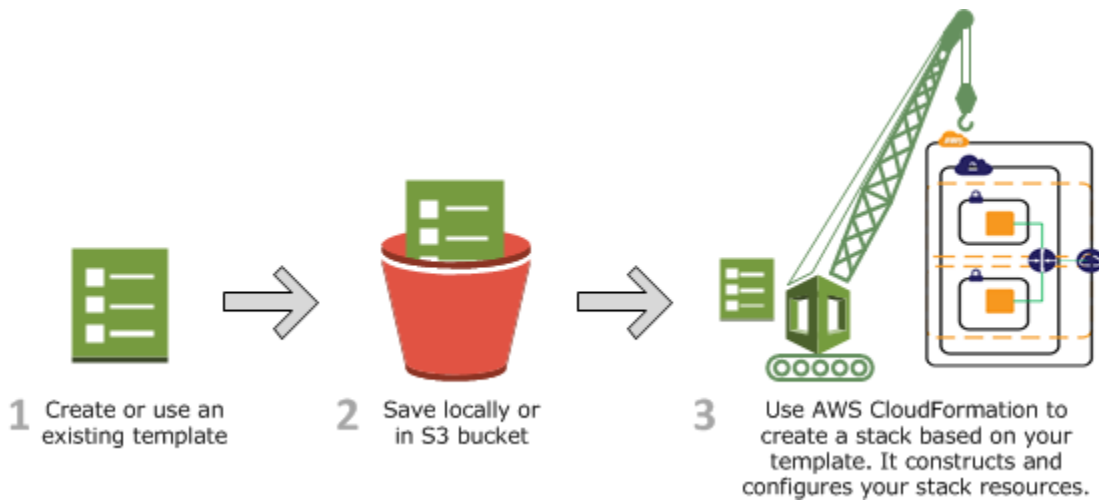
例えば、Amazon RDS データベースインスタンスの名前を変更すると、CloudFormation によって新しいデータベースが作成され、古いものは削除されます。古いデータベースのデータは、バックアップしていない限り、失われます。変更セットを生成すると、変更によってデータベースが置き換えられることがわかり、スタックを更新する前に対応策を立てることができます。

CloudFormation の仕組み

CloudFormation を使用してスタックを作成するとき、CloudFormation は AWS に基礎となるサービス呼び出しを行い、テンプレートで説明されているリソースをプロビジョニングして構成します。これらのリソースを作成するにはアクセス許可が必要です。例えば、CloudFormation を使用して EC2 インスタンスを作成するには、インスタンスを作成するためのアクセス権限が必要です。これらのアクセス許可は [AWS Identity and Access Management \(IAM\)](#) で管理します。

CloudFormation が行う呼び出しは、すべてテンプレートで宣言されます。たとえば、EC2 インスタンスのインスタンスタイプを `t2.micro` と記述するテンプレートがあるとします。このテン

プレートを使ってスタックを作成すると、CloudFormation は Amazon EC2 のインスタンス作成 API を呼び出し、インスタンスタイプを `t2.micro` と指定します。次の図は、スタックを作成する CloudFormation ワークフローをまとめたものです。



スタックを作成するには

1. テキストエディタを使用して YAML または JSON 形式で CloudFormation テンプレートを作成します。CloudFormation テンプレートは必要なリソースとその設定を記述します。[Infrastructure Composer](#) を使用してテンプレートを視覚化および検証します。テンプレートが適切に構造化され、構文エラーがないことを確認できます。詳細については、「[CloudFormation テンプレートの使用](#)」を参照してください。
2. テンプレートをローカルに保存、または Amazon S3 バケットに保存します。
3. テンプレートファイルの場所 (ローカルコンピュータのパスまたは Amazon S3 の URL など) を指定して CloudFormation スタックを作成します。テンプレートにパラメーターが含まれる場合は、スタック作成時に入力値を指定できます。パラメータは、テンプレートに値を渡すことを可能にして、スタックを作成するたびにリソースをカスタマイズできるようにします。

Note

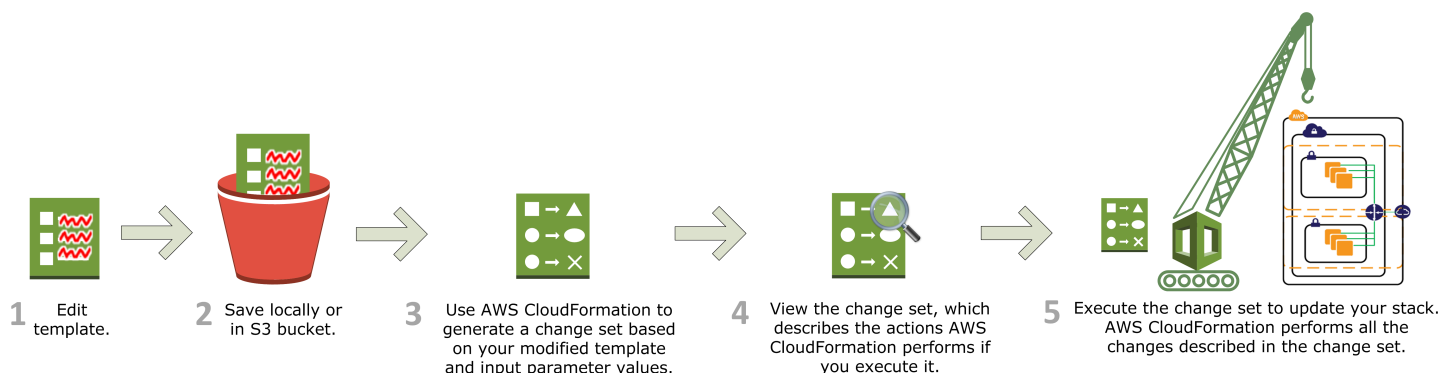
ローカルに保存されたテンプレートファイルを指定する場合、CloudFormation はそれを AWS アカウントの S3 バケットにアップロードします。CloudFormation は各リージョンにテンプレートファイルをアップロードするバケットを作成します。AWS アカウントで Amazon Simple Storage Service (Amazon S3) 許可があれば、誰でもバケットにアクセスできます。CloudFormation が作成したバケットが既に存在する場合は、テンプレートはそのバケットに追加されます。

手動でテンプレートを Amazon S3 にアップロードすることで、独自のバケットを使用してアクセス権限を管理できます。その場合は、スタックを作成または更新するたびに、テンプレートファイルの Amazon S3 URL を指定します。

すべてのリソースの作成が完了すると、CloudFormation はスタックが正常に作成された旨のレポートを送信します。これで、スタック内のリソースが使用できるようになります。スタックの作成が失敗した場合、CloudFormation は作成したリソースを削除して、変更をロールバックします。

変更セットを使用したスタックの更新

スタックのリソースを更新する必要がある場合、スタックのテンプレートを変更できます。新しいスタックを作成して古いものを削除する必要はありません。スタックを更新するには、元のスタックテンプレートの修正バージョン、異なる入力パラメータ値、またはその両方を送信して変更セットを作成します。CloudFormation は、修正されたテンプレートを元のテンプレートと比較して、変更セットを生成します。変更セットには、変更案がリストされます。変更を確認した後、変更セットを実行してスタックを更新、あるいは新しい変更セットを作成できます。次の図は、スタックを更新するワークフローの概要です。



変更セットを使用してスタックを更新する方法

1. [Infrastructure Composer](#) またはテキストエディタを使用して、CloudFormation スタックテンプレートを修正できます。詳細については、「[スタックテンプレートを更新する](#)」を参照してください。

テンプレートを更新する際、更新によって中断が発生する可能性があることに注意してください。更新しているリソースやプロパティによって、既存のリソースが中断されたり、置き換えられる場合もあります。詳細については、「[スタックリソースの更新動作を理解する](#)」を参照してください。

2. CloudFormation テンプレートをローカルまたは S3 バケットに保存します。

- 更新するスタックおよび修正済みテンプレートの場所 (ローカルコンピューターのパスや Amazon S3 の URL など) を指定し、変更セットを作成します。変更セットの作成方法については、[変更セットを使用して CloudFormation スタックを更新する](#) を参照してください。

Note

ローカルコンピューターに保存されたテンプレートを指定する場合、CloudFormation はテンプレートを自動的に AWS アカウント アカウントの S3 バケットにアップロードします。

- 変更セットを参照して、CloudFormation が想定どおりに変更を実行することを確認します。例えば、CloudFormation が重要なスタックリソースを置き換えるかどうかを確認します。必要な変更が含まれるまで、いくつでも変更セットを作成できます。

Important

変更セットでは、スタックの更新が正常に行われるかどうかはわかりません。例えば、変更セットはアカウント [クォータ](#) を超過したかどうかしませんが、更新をサポートしないリソースをする場合、あるいはリソースの修正に必要な [アクセス許可](#) が不十分な場合、スタックの更新が失敗する原因になる恐れがあります。

- スタックに適用する変更セットを開始します。CloudFormation は、修正したリソースのみを更新することでスタックを更新し、スタックが正常に更新された旨の信号を送信します。スタックの更新が失敗した場合、CloudFormation は変更箇所をロールバックして、スタックを正常な稼働が確認されている最新の状態に復元します。

CloudFormation の使用を開始する方法

コンソールで Hello world CloudFormation スタックを作成するには、「[最初のスタックの作成](#)」を参照してください。

ガイド付きで学習するには、テンプレート開発を実践的に学べる「[CloudFormation の使用開始](#)」ワークショップをお試しください。

AWS アカウント へのサインアップ

AWS にサインアップすると、あなたの AWS アカウント は、CloudFormation を含めた AWS 内のすべてのサービスに自動的にサインアップされます。AWS アカウント が既にある場合は、次のトピック

クに進んでください。AWS アカウントをお持ちでない場合は、次に説明する手順に従ってアカウントを作成してください。

AWS アカウント へのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

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、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

AWS アカウントにサインアップすると、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[ルートユーザーアクセスが必要なタスク](#)を実行してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。<https://aws.amazon.com/> 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

AWS アカウントにサインアップしたら、AWS アカウントのルートユーザーをセキュリティで保護し、AWS IAM Identity Center を有効にして、管理ユーザーを作成します。これにより、日常的なタスクにルートユーザーを使用しないようにします。

AWS アカウントのルートユーザーをセキュリティで保護する

1. [ルートユーザー] を選択し、AWS アカウントのメールアドレスを入力して、アカウント所有者として [AWS マネジメントコンソール](#) 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 ユーザーガイド」で [AWS アカウントのルートユーザーの仮想 MFA デバイスを有効にする方法 \(コンソール\)](#) を確認してください。

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「[AWS IAM Identity Center の有効化](#)」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。

IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「[Configure user access with the default 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 アイデンティティセンター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 User Guide」の「[Signing in to the AWS access portal](#)」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. 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「[権限設定を作成する](#)」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「[グループの追加](#)」を参照してください。

Note

アクセスを管理する方法の詳細については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。

最初のスタックの作成

このトピックでは、AWS マネジメントコンソール を使用して最初の CloudFormation スタックを作成する方法について説明します。このチュートリアルでは、基本的なAWSリソースのプロビジョニング、スタックイベントのモニタリング、出力の生成方法について説明します。

この例では、CloudFormation テンプレートは YAML で記述されています。YAML は、インフラストラクチャをコードとして定義するために広く使用されている人間が読める形式です。CloudFormation の詳細については、JSON 形式の他のテンプレートも表示されますが、このチュートリアルでは読みやすいように YAML が使われています。

Note

CloudFormation は無料で使用できますが、作成した Amazon EC2 および Amazon S3 リソースに対しては課金されます。ただし、AWS を初めて使用する場合は、[無料利用枠](#)を利用して、この学習プロセス中のコストを最小に抑えるか無料にできます。

トピック

- [前提条件](#)
- [コンソールを使用して CloudFormation スタックを作成する](#)
- [スタックの作成をモニタリング](#)
- [ウェブサーバーをテストする。](#)
- [トラブルシューティング](#)
- [クリーンアップ](#)
- [次のステップ](#)

前提条件

- Amazon EC2、Amazon S3、および CloudFormation を使用するアクセス許可を持つ IAM ユーザーまたはロール、または管理ユーザーアクセスで AWS アカウント にアクセス可能である必要があります。
- インターネットにアクセスできる Virtual Private Cloud (VPC) が必要です。このチュートリアルテンプレートには、新しい AWS アカウント に自動的に付属するデフォルトの VPC が必要です。デフォルトの VPC がない場合、または削除された場合は、このトピックのトラブルシューティングセクションで代替解決策を参照してください。

コンソールを使用して CloudFormation スタックを作成する

コンソールで Hello world CloudFormation スタックを作成するには

1. [クラウドフォーメーション コンソール](#)を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [スタックの作成] ページで、[Infrastructure Composer からビルド]、[Infrastructure Composer で作成] の順に選択します。これにより、CloudFormation コンソールモードで Infrastructure Composer に移動し、サンプルテンプレートをアップロードして検証できます。
4. サンプルテンプレートをアップロードして検証するには、次の手順を実行します。
 - a. [テンプレート] を選択します。次に、次の CloudFormation テンプレートをコピーしてテンプレートエディタに貼り付けます。

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2 Instance

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
```

AllowedValues:

- t3.micro
- t2.micro

ConstraintDescription: must be a valid EC2 instance type.

MyIP:

Description: Your IP address in CIDR format (e.g. 203.0.113.1/32).

Type: String

MinLength: '9'

MaxLength: '18'

Default: 0.0.0.0/0

AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\.\d{1,2}\$'

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Allow HTTP access via my IP address

SecurityGroupIngress:

- IpProtocol: tcp
- FromPort: 80
- ToPort: 80
- CidrIp: !Ref MyIP

WebServer:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref LatestAmiId

InstanceType: !Ref InstanceType

SecurityGroupIds:

- !Ref WebServerSecurityGroup

UserData: !Base64 |

#!/bin/bash

yum update -y

yum install -y httpd

systemctl start httpd

systemctl enable httpd

echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/index.html

Outputs:

WebsiteURL:

Value: !Join

```
- ''
- - http://
- - !GetAtt WebServer.PublicDnsName
Description: Website URL
```

次のステップに進む前に、テンプレートを確認し、CloudFormation の主要な概念を把握しましょう。

- この **Parameters** セクションでは、スタックを作成するときにテンプレートに渡すことができる値を宣言します。テンプレートで後で指定されるリソースは、これらの値を参照し、データを使用します。テンプレートにハードコーディングしたくない情報をパラメータに設定できます。また、デプロイする特定のアプリケーションまたは設定に固有の情報を指定する方法でもあります。
- テンプレートでは次のパラメータを定義します。
 - **LatestAmiId** – AWS Systems Manager Parameter Store から最新の Amazon Linux 2 AMI ID を取得します。
 - **InstanceType** – EC2 インスタンスタイプの選択を許可します (デフォルト: t2.micro、t3.micro、t2.micro を選択できます)。
 - **MyIP** – HTTP アクセスの IP アドレス範囲を指定します (デフォルト: 0.0.0.0/0、任意の IP からのアクセスを許可できます)。
- **Resources** セクションには、テンプレートで作成する AWS リソースの定義が含まれています。リソース宣言は、これらすべての構成を一度に指定するための効率的な方法です。テンプレートにリソース宣言を含めると、テンプレートを使用してスタックを作成することによって、宣言したすべてのリソースを作成および設定できます。同じテンプレートから新しいスタックを作成することで、同じ設定のリソースを複製することができます。
- このテンプレートによって以下のリソースが作成されます。
 - **WebServerSecurityGroup** – 指定された IP 範囲からのポート 80 でのインバウンド HTTP トラフィックを許可する EC2 セキュリティグループ。
 - **WebServer** – 次の設定の EC2 インスタンス。
 - 最新の Amazon Linux 2 AMI を使用します。
 - 選択したインスタンスタイプを適用します
 - SecurityGroupIds プロパティに WebServerSecurityGroup を追加します。
 - Apache HTTP Server をインストールするためのユーザーデータスクリプトが含まれています

- 論理名は、各リソースおよびパラメータ宣言の先頭に指定されます。例えば、`WebServerSecurityGroup` は EC2 セキュリティグループリソースに割り当てられた論理名です。Ref 関数は、テンプレート内の他の部分で、リソースやパラメータを論理名で参照するのに使用されます。あるリソースが別のリソースを参照すると、リソース間に依存関係が作成されます。
 - **Outputs** セクションは、スタックの作成後に返されるカスタム値を定義します。出力値を使用することで、スタック内のリソースに関する情報 (リソース識別子、URL など) を取得できます。
 - テンプレートは 1 つの出力を定義します。
 - **WebsiteURL** – EC2 インスタンスのパブリック DNS 名を使用して構築された、デプロイ済のウェブサーバーの URL。Join 関数は、固定の `http://` と `PublicDnsName` 変数を 1 つの文字列に結合し、ウェブサーバーの完全な URL を簡単に出力するのに役立ちます。
- b. テンプレートをアップロードする前に、[検証] を選択して YAML コードが有効であることを確認します。
 - c. 次に、[テンプレートを作成] を選択してテンプレートを作成し、S3 バケットに追加します。
 - d. 開いたダイアログボックスから、後で削除できるように S3 バケットの名前を書き留めておきます。次に、[確認] を選択し、CloudFormation に進みます。移動先の CloudFormation コンソールには、テンプレートへの S3 パスが指定されています。
5. [スタックの作成] ページで、[次へ] を選択します。
 6. [スタックの詳細を指定] ページの [スタック名] フィールドに名前を入力します。スタック名にスペースを含めることはできません。この例では **MyTestStack** を使用します。
 7. [パラメータ] で、次のようにパラメータ値を指定します。
 - `LatestAmild`: これはデフォルトで最新の Amazon Linux 2 AMI に設定されます。
 - `InstanceType`: EC2 インスタンスタイプとして `t2.micro` または `t3.micro` を選択します。

 Note

AWS を初めてご利用いただく場合、無料利用枠を使用して `t2.micro` インスタンスを起動し、12 か月間無料で使用できます (`t2.micro` が利用できないリージョンでは、無料利用枠で `t3.micro` インスタンスを使用できます)。

- MyIP: 実際のパブリック IP アドレスを /32 サフィックスで指定します。/32 サフィックスは、単一の IP アドレスを許可するように指定するために CIDR 表記で使用されます。これは、この特定の IP アドレスとの間のトラフィックを許可し、他の IP アドレスは許可しないことを意味します。
8. [次へ] を 2 回選択して、[名前、確認、および作成] ページに進みます。このチュートリアルでは、[スタックオプションの設定] ページのデフォルトをそのままにしておくことができます。
 9. スタックの情報を確認します。設定が適切であることを確認したら、[Submit (送信)] を選択します。

スタックの作成をモニタリング

[送信] を選択すると、CloudFormation はテンプレートで指定されたリソースの作成を開始します。新しいスタックである **MyTestStack** が、CloudFormation コンソールの上部のリストに表示されます。ステータスは `CREATE_IN_PROGRESS` である必要があります。スタックの詳細なステータスは、イベントを表示して確認できます。

スタックのイベントを表示するには

1. CloudFormation コンソールのリストで、スタック **MyTestStack** を選択します。
2. スタックの詳細ペインで、[Events] (イベント) タブを選択します。

コンソールは自動的に更新され、60 秒ごとに最新のイベントがイベントリストに表示されます。

[イベント] タブには、スタックの作成の主要な各ステップが、最新のイベントが一番上になるように時間順に表示されます。

最初のイベント (イベントリストの一番下のイベント) は、スタック作成プロセスの開始です。

```
2024-12-23 18:54 UTC-7 MyTestStack CREATE_IN_PROGRESS User initiated
```

続いて、各リソースの作成の開始と終了を示すイベントが表示されます。たとえば、EC2 インスタンスの作成は、次のようなエントリになります。

```
2024-12-23 18:59 UTC-7 WebServer CREATE_COMPLETE
```

```
2024-12-23 18:54 UTC-7 WebServer CREATE_IN_PROGRESS Resource creation initiated
```

CloudFormation がリソースの作成を開始したことがレポートされると、CREATE_IN_PROGRESS イベントが記録されます。リソースが正常に作成されると、CREATE_COMPLETE イベントが記録されます。

CloudFormation によって正常にスタックが作成されると、[Events] (イベント) タブの上部に次のイベントが表示されます。

```
2024-12-23 19:17 UTC-7 MyTestStack CREATE_COMPLETE
```

CloudFormation がリソースを作成できない場合は、CREATE_FAILED イベントがレポートされます。また、デフォルトではスタックがロールバックされて、作成されたリソースがある場合はすべて削除されます。[状況の理由] 列には、失敗の原因となった問題が表示されます。

スタックが作成された後、[リソース] タブで作成した EC2 インスタンスとセキュリティグループが表示されます。

ウェブサーバーをテストする。

スタックが正常に作成されたら、CloudFormation コンソールの [Outputs] タブに移動します。[WebsiteURL] フィールドを探します。これには、EC2 インスタンスのパブリック URL が含まれます。

ブラウザを開き、[WebsiteURL] にリストされている URL に移動します。ブラウザに「"Hello World!"」とメッセージが表示されます。

これにより、EC2 インスタンスが Apache HTTP Server を実行し、基本的なウェブページを提供していることがわかります。

トラブルシューティング

スタックの作成中にロールバックが発生した場合は、VPC の欠落が原因である可能性があります。この問題を解決する方法は次のとおりです。

使用可能なデフォルト VPC がない

このチュートリアルテンプレートには、デフォルトの VPC が必要です。VPC またはサブネットの可用性エラーが原因でスタックの作成が失敗した場合、アカウントにデフォルトの VPC がない可能性があります。次のオプションがあります。

- 新しいデフォルト VPC を作成する – Amazon VPC コンソールを使用して新しいデフォルト VPC を作成できます。手順については、「Amazon VPC ユーザーガイド」の「[デフォルトの VPC を作成する](#)」を参照してください。

- テンプレートを変更してサブネットを指定する – デフォルト以外の VPC がある場合は、テンプレートを変更して VPC とサブネット ID を明示的に指定できます。テンプレートに次のパラメータを追加します。

```
SubnetId:
  Description: The subnet ID to launch the instance into
  Type: AWS::EC2::Subnet::Id
```

次に、WebServer リソースを更新してサブネット ID を含めます。

```
WebServer:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref LatestAmiId
    InstanceType: !Ref InstanceType
    SecurityGroupIds:
      - !Ref WebServerSecurityGroup
    SubnetId: !Ref SubnetId
    UserData: !Base64 |
      #!/bin/bash
      yum update -y
      yum install -y httpd
      systemctl start httpd
      systemctl enable httpd
      echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
```

スタックを作成する際に、ウェブサーバーにアクセスできるようにするには、インターネットアクセスを持つサブネットを指定する必要があります。

クリーンアップ

不要なサービスに対して課金されていないことを確認するために、スタックとそのリソースを削除して、クリーンアップすることができます。スタックのテンプレートを保存する S3 バケットを削除することもできます。

スタックとそのリソースを削除するには

1. [クラウドフォーメーション コンソール](#)を開きます。

2. [スタック] ページで、作成したスタックの名前の横にあるオプション [MyTestStack] を選択し、続いて [削除] を選択します。
3. 確認を求めるメッセージが表示されたら、[削除] を選択してください。
4. [イベント] タブでスタック削除プロセスの進行状況をモニタリングします。MyTestStack のステータスが DELETE_IN_PROGRESS に変わります。CloudFormation がスタックの削除を完了すると、リストからスタックが削除されます。

サンプルテンプレートの使用が終了し、Amazon S3 バケットが不要になったら削除します。バケットを削除するには、バケットを空にする必要があります。バケットを空にすると、バケット内のすべてのオブジェクトが削除されます。

Amazon S3 バケットを空にして削除するには

1. [Amazon S3 コンソール](#)を開きます。
2. コンソールの左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、このチュートリアル用に作成したバケットの名前の横にあるオプションを選択し、[空] を選択します。
4. [バケットを空にする] ページで、テキストフィールドに **permanently delete** と入力して、[空にする] をクリックします。
5. [バケットを空にする: ステータス] ページで、バケットを空にするプロセスの進行状況をモニタリングします。
6. バケットリストに戻るには、[Exit (終了)] を選択します。
7. バケットの名前の横にあるオプションを選択し、[削除] を選択します。
8. 確認を求められたら、バケットの名前を入力し、[バケットを削除] を選択します。
9. [バケット] リストで、バケット削除プロセスの進捗状況を確認します。Amazon S3 がバケットの削除を完了すると、バケットはリストから消えます。

次のステップ

お疲れ様でした。スタックを正常に作成し、その作成をモニタリングし、その出力を使用しました。

学習を継続するには:

- 独自のテンプレートを作成できるように、テンプレートについて詳しく学習してください。詳細については、「[CloudFormation テンプレートの使用](#)」を参照してください。

- テンプレート作成の実践的な演習については、「[Getting Started with CloudFormation](#)」ワークショップをお試してください。
- 「[Getting Started with CloudFormation](#)」の短縮版については、「[Amazon EC2 にアプリケーションをデプロイする](#)」を参照してください。このトピックでは、CloudFormation ヘルパースクリプト `cfn-init` を使用して Amazon EC2 インスタンスをブートストラップする場合と同じシナリオについて説明します。

CloudFormation ベストプラクティス

ベストプラクティスは、ワークフローの全体を通じて CloudFormation をより効率的に使用し、安全な慣行を導入するために役立つ推奨事項です。スタックの計画方法と整理方法を学習し、リソースとそこで実行するソフトウェアアプリケーションを記述するテンプレートを作成して、スタックとそのリソースを管理します。次のベストプラクティスは、現在の CloudFormation のお客様の実際の経験に基づいています。

計画と編成

- [フィードバックループを短くしてデプロイ速度を向上させる](#)
- [ライフサイクルと所有権によるスタックの整理](#)
- [クロススタック参照を使用して、別のスタックがエクスポートした出力の値を返します。](#)
- [マルチアカウントデプロイとマルチリージョンデプロイに CloudFormation StackSets を使用する](#)
- [テンプレートを再利用して複数の環境にスタックを複製する](#)
- [すべてのリソースタイプのクォータを確認する](#)
- [モジュールを使用してリソース構成を再利用する](#)
- [Infrastructure as Code 慣行を導入する](#)

テンプレートの作成

- [テンプレートに認証情報を埋め込まない](#)
- [AWS 固有のパラメータタイプを使用する](#)
- [パラメータの制約の使用](#)
- [疑似パラメータを使用して移植性を高める](#)
- [AWS::CloudFormation::Init を使用して Amazon EC2 インスタンスにソフトウェアアプリケーションをデプロイ](#)
- [最新のヘルパースクリプトを使用する](#)
- [テンプレートを使用する前に検証する](#)
- [テンプレートオーサリングに YAML または JSON を使用する](#)
- [包括的なタグ付け戦略を実装する](#)
- [高度な変換にテンプレートマクロを活用する](#)

スタックの管理

- [CloudFormation ですべてのスタックリソースを管理する](#)

- [スタックを更新する前に変更セットを作成する](#)
- [スタックポリシーを使用してリソースを保護する](#)
- [AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する](#)
- [コードの確認とリビジョン管理を使用してテンプレートを管理する](#)
- [Amazon EC2 インスタンスを定期的に更新](#)
- [ドリフト検出を定期的に使用する](#)
- [自動復旧のためのロールバックトリガーを設定する](#)
- [効果的なスタックリファクタリング戦略を実装する](#)
- [ライフサイクル管理に CloudFormation Hooks を使用する](#)

ツールのオーサリング

- [IaC ジェネレーターを使用して既存のリソースからテンプレートを作成する](#)
- [ビジュアルテンプレートの設計に AWS Infrastructure Composer を使用する](#)
- [複雑なインフラストラクチャに AWS Cloud Development Kit \(AWS CDK\) を使用することを検討する](#)

セキュリティとコンプライアンス

- [IAM を使用したアクセス制御](#)
- [最小特権の原則を適用する](#)
- [機密性の高いパラメータをセキュア化する](#)
- [AWS CloudFormation Guard を使用してポリシーをコードとして実装する](#)

フィードバックループを短くしてデプロイ速度を向上させる

CloudFormation テンプレートで記述したインフラストラクチャのフィードバックループを短縮するのに役立つ、プラクティスやツールを採用します。これには、ワークステーションで早期にテンプレートのリンティングとテストを行うことが含まれます。そうすることで、コントリビューションをソースコードリポジトリに送信する前でも、潜在的な構文や構成上の問題を発見する機会が得られます。このような問題を早期に発見することで、開発、品質保証、生産などの正式なライフサイクル環境に問題を持ち込まないようにすることができます。この早期テストによるフェイルファーストのアプローチには、やり直しの待ち時間が減り、潜在的な影響範囲が減り、プロビジョニングオペレーションが成功する自信が高まるというメリットがあります。

フェイルファーストの実践に役立つツールの選択肢には、「[CloudFormation Linter](#)」(cfn-lint) や「[TaskCat](#)」 コマンドラインツールが含まれます。cfn-lint ツールは、CloudFormation テンブ

レートで「[CloudFormation リソース仕様](#)」に対して検証する機能をもたらします。これには、リソースプロパティの有効な値の確認やベストプラクティスが含まれます。cfn-lint のプラグインは[多くのコードエディターで利用できます](#)。エディター内の問題を視覚化し、リンターのフィードバックを直接得ることができます。ソースコードリポジトリの設定に cfn-lint を統合することも選択できるため、コントリビューションをコミットするときにテンプレートの検証を実行できます。詳細については、「[cfn-lint による CloudFormation テンプレートの Git コミット前検証](#)」を参照してください。最初のリンティングを実行し、cfn-lint で発生した可能性のある問題を修正したら、選択した AWS リージョン にプログラムでスタックを作成することにより、TaskCat を使用してテンプレートをテストできます。TaskCat は、選択した各リージョンに対して合格/不合格の評価を含むレポートも生成します。

両方のツールを使用してフィードバックループを短縮する方法について、ステップバイステップで実践的なウォークスルーを行うには、「[CloudFormation ワークショップ](#)」の「[リンティングとテストラボ](#)」に従ってください。

ライフサイクルと所有権によるスタックの整理

AWS リソースのライフサイクルと所有権を使用して、各スタックでどのリソースを使うを判断します。最初はすべてのリソースを 1 つのスタックに置いておかまいませんが、スタックの規模が大きくなり範囲が拡大するにつれて、単一のスタックの管理は面倒で時間かかる場合があります。共通のライフサイクルと所有権を持つリソースのグループ化により、所有者は独自のプロセスやスケジュールを使用して、他のリソースに影響を与えることなくリソースのセットを変更できます。

たとえば、ロードバランサーの背後にある Amazon EC2 Auto Scaling インスタンスにホストされているウェブサイト所有するデベロッパーおよびエンジニアのチームがあることを想定します。ウェブサイトは独自のライフサイクルを持ちウェブサイトチームによって保守されているため、このウェブサイトのスタックやそのリソースを作成できます。それでは、同様にバックエンドのデータベースを使用するウェブサイトがあり、そのデータベースは別のスタックでデータベース管理者が所有し保守しているとしましょう。ウェブサイトチームまたはデータベースチームがリソースを更新する必要があるときはいつでも、互いのスタックに影響を与えることなく実行できます。すべてのリソースが単一のスタックにある場合は、更新の連携や連絡は難しくなるでしょう。

スタックの整理についての詳細なガイダンスについては、2 つの一般的なフレームワークを使用できます。多層アーキテクチャーとサービス指向アーキテクチャー (SOA) です。

多層アーキテクチャーは、スタックを積み上げて構築する複数の水平の層に整理します。各層はその直下の層に依存します。各層には 1 つ以上のスタックを持つことができますが、各層のスタックは類似したライフサイクルと所有権を持つ AWS リソースを持つ必要があります。

サービス指向アーキテクチャを使用すると、業務上の大きな問題进行处理しやすい大きさに整理できます。これらのパートはそれぞれ、明確に定義された目的があり、機能の自己充足単位を表します。これらのサービスを、それぞれ独自のライフサイクルと所有者があるスタックにマッピングできます。これらのサービス (スタック) を 1 つに繋いで、相互に通信するようにできます。

クロススタック参照を使用して、別のスタックがエクスポートした出力の値を返します。

ライフサイクルと所有権に基づいて AWS リソースを整理するときに、別のスタックにあるリソースを使用するスタックを構築することもできます。値をハードコード化または入力パラメータを使用して、リソース名と ID を渡すことができます。ただし、これらのメソッドはテンプレートの再利用を困難にしたり、スタック実行のオーバーヘッドを増やす場合があります。その代わりにクロススタック参照を使用して別のスタックがエクスポートした出力の値を返し、他のスタックがそれらを使用できるようにします。スタックは `Fn::ImportValue` 関数を使用して、エクスポートされたリソースを呼び出して使用できます。

たとえば、VPC、セキュリティグループ、およびサブネットを含むネットワークスタックがあるかもしれません。すべてのパブリックウェブアプリケーションにこれらのリソースを使用する場合。リソースをエクスポートすることによって、パブリックウェブアプリケーションのすべてのスタックでリソースが使用できるようになります。詳細については、「[デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)」を参照してください。

マルチアカウントデプロイとマルチリージョンデプロイに CloudFormation StackSets を使用する

CloudFormation StackSets は、複数のアカウントとリージョン全体でのスタックの作成、更新、削除を単一のオペレーションで実行できるようにすることで、スタックの機能を拡張します。StackSet は、組織全体での一般的なインフラストラクチャコンポーネント、コンプライアンス制御、共有サービスのデプロイに使用します。

StackSet を使用するときは、許可の管理を簡素化するために、AWS Organizations を使用してサービスマネージド型の許可を実装します。このアプローチは、各アカウントで IAM ロールを手動設定することを必要とせずに、組織内のアカウントに StackSet をデプロイできるようにします。

StackSet の詳細については、「[StackSets の概念](#)」を参照してください。

すべてのリソースタイプのクォータを確認する

スタックを起動する前に、必要なすべてのリソースを AWS アカウントの制限に触れずに作成できることを確認します。制限に触れる場合は、クォータを増やすかまたは追加リソースを削除するまで、CloudFormation でスタックが正常に作成されません。各サービスには、スタックを起動する前に知っておくべきさまざまな制限があります。たとえば、デフォルトでは、AWS アカウント でリージョンごとに起動できる CloudFormation スタックは 2,000 件のみです。制限およびデフォルトの制限を引き上げる方法の詳細については、「AWS 全般のリファレンス」の「[AWS のサービスクォータ](#)」を参照してください。

テンプレートを再利用して複数の環境にスタックを複製する

スタックとリソースをセットアップした後、テンプレートを再利用してインフラストラクチャを複数の環境に複製できます。たとえば、開発、テスト、本番の環境を作成して、本番環境に実装する前に変更をテストすることができます。テンプレートを再利用可能にするには、パラメーター、マッピング、および条件セクションを使用してスタックの作成時にカスタマイズできるようにします。たとえば、開発環境では、本番環境と比べて低価格のインスタンスタイプを指定し、その他の構成や設定は同じにできます。パラメーター、マッピング、条件の詳細については、「[CloudFormation テンプレートセクション](#)」を参照してください。

モジュールを使用してリソース構成を再利用する

インフラストラクチャが大きくなるにつれ、各テンプレートで同じコンポーネントを宣言する共通パターンができてきます。モジュールは、スタックテンプレート全体に含めるリソース構成をパッケージ化する、透過的で管理しやすく、繰り返し可能な方法です。モジュールは、共通のサービス構成とベストプラクティスを、スタックテンプレートに含めるためのモジュール式のカスタマイズ可能なビルディングブロックとしてカプセル化できます。

これらのビルディングブロックは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを定義するためのベストプラクティスなど、単一のリソース用にすることも、アプリケーションアーキテクチャの一般的なパターンを定義するために、複数のリソース用にすることもできます。これらのビルディングブロックは他のモジュールにネストできるため、ベストプラクティスを上位レベルのビルディングブロックに積み重ねることができます。CloudFormation モジュールは [CloudFormation レジストリ](#) で利用できるため、ネイティブリソースのように使用できます。CloudFormation モジュールを使用すれば、モジュールテンプレートが使用テンプレートに展開され、[Ref](#) または [Fn::GetAtt](#) を使用してモジュール内のリソースにアクセスできるようになります。詳細については、

「[CloudFormation モジュールを使用してテンプレート全体に含めることができる再利用可能なリソース設定の作成](#)」を参照してください。

Infrastructure as Code 慣行を導入する

Infrastructure as Code (IaC) 慣行を実装することで、CloudFormation テンプレートをコードとして扱います。テンプレートをバージョン管理システムに保存し、コードレビューを実装して、自動化されたテストを使用して変更を検証します。このアプローチは、一貫性を確保し、コラボレーションを向上させ、インフラストラクチャ変更の監査証跡を提供します。

インフラストラクチャコードに CI/CD パイプラインを実装して、CloudFormation テンプレートのテストとデプロイを自動化することを検討してください。AWS CodePipeline、AWS CodeBuild、AWS CodeDeploy といったツールは、インフラストラクチャをデプロイするための自動化されたワークフローの作成に使用できます。

IaC ベストプラクティスの実装に関する詳細については、「[Using AWS CloudFormation as an IaC tool](#)」を参照してください。

CloudFormation での継続的デリバリーの使用に関する詳細については、「[CodePipeline を使用した継続的デリバリー](#)」を参照してください。

テンプレートに認証情報を埋め込まない

CloudFormation テンプレートに機密情報を埋め込むのではなく、スタックテンプレートで動的なリファレンスを使用することをお勧めします。

動的なリファレンスにより、AWS Systems Manager パラメータストアや AWS Secrets Manager などの他のサービスで保存、管理されている外部値をスタックテンプレートに指定するためのシンプルで強力な方法が提供されます。動的なリファレンスを使用すると、CloudFormation は、スタックオペレーションおよび変更セットオペレーション中に指定されたリファレンスの値を取得して、その値を適切なリソースに渡します。ただし、CloudFormation が実際の参照値を保存することはありません。詳細については、「[動的な参照を使用してテンプレート値を指定する](#)」を参照してください。

[AWS Secrets Manager](#) を使用して、データベースやその他のサービスの認証情報を安全に暗号化、保存、取得できます。[AWS Systems Manager パラメータストア](#) は、構成データ管理のための安全な階層型ストレージを提供します。

起動テンプレートのパラメータの詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

AWS 固有のパラメータタイプを使用する

テンプレートは、既存の Amazon Virtual Private Cloud ID または Amazon EC2 キーペア名など、既存の AWS 固有値の入力が必須の場合、AWS 固有のパラメータタイプを使用します。たとえば、パラメーターをタイプと指定して、AWS アカウント とスタックを作成するリージョンの既存のキーペア名を取得します。CloudFormation は、スタックを作成する前に、AWS 固有のパラメータタイプの値をすばやく検証できます。また、CloudFormation コンソールを使用する場合、CloudFormation が有効な値のドロップダウンリストを表示するため、正しい VPC ID またはキーペア名を調べたり暗記したりする必要はありません。詳細については、「[CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)」を参照してください。

パラメータの制約の使用

制約を使用すると、許可される入力値を記述することで CloudFormation がスタックを作成する前に無効な値を捕捉できます。最小文字数、最長文字数、許可パターンなどの制限を設定できます。例えば、データベースのユーザー名の値は最小 8 文字で英数字のみ、といった制限を設定できます。詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

疑似パラメータを使用して移植性を高める

テンプレート内の[疑似パラメータ](#)を、Ref や Fn::Sub などの[組み込み関数](#)の引数として使用できます。疑似パラメータは、CloudFormation によって事前定義されたパラメータです。テンプレートでは宣言しません。組み込み関数で疑似パラメータを使用すると、リージョンおよびアカウント間でスタックテンプレートのポータビリティが向上します。

例えば、特定のリソースプロパティについて、別の既存リソースの [Amazon リソースネーム](#) (ARN) を指定する必要があるテンプレートを作成したいとします。この場合、既存のリソースは ARN: `arn:aws:ssm:us-east-1:123456789012:parameter/MySampleParameter` を持つ [AWS Systems Manager パラメータストア](#) リソースです。「[ARN 形式](#)」を対象の AWS パーティション、リージョン、アカウント ID に合わせる必要があります。これらの値をハードコーディングする代わりに、AWS::Partition、AWS::Region、AWS::AccountId の疑似パラメータを使用してテンプレートの移植性を高めることができます。この場合に CloudFormation で ARN 内の要素を連結する方法の例としては、`!Sub 'arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/MySampleParameter'` があります。

別の例として、リソースや設定を複数のスタック間で共有したいとします。この例では、VPC の[サブネット](#)を作成し、その ID を同じ AWS アカウント とリージョンで他のスタックと使用するた

めにエクスポートしたことを想定します。別のスタックでは、Amazon EC2 インスタンスを記述するときに、サブネット ID のエクスポートされた値を参照します。Export 出力フィールドと Fn::ImportValue 組み込み関数の詳細な使用例については、[別の CloudFormation スタックのリソース出力を参照する](#) を参照してください。

スタックのエクスポートは、アカウントおよびリージョンごとに一意である必要があります。そのため、この場合は、AWS::StackName 擬似パラメータを使用してエクスポート用のプレフィックスを作成できます。スタック名もアカウントおよびリージョンごとに一意であることが必要なため、この疑似パラメータをプレフィックスとして使用すると、一意のエクスポート名を持つ可能性が高くなるだけでなく、値をエクスポートするスタック全体の再利用可能なアプローチも促進されます。また、任意のプレフィックスを使用することも可能です。

AWS::CloudFormation::Init を使用して Amazon EC2 インスタンスにソフトウェアアプリケーションをデプロイ

スタックを起動すると、cfn-init ヘルパースクリプトおよび AWS::CloudFormation::Init リソースを使用して、Amazon EC2 インスタンスにソフトウェアアプリケーションをインストールして設定できます。AWS::CloudFormation::Init を使用することで、スクリプトで手順を順番に実行するよりも自由に構成を記述できます。また、インスタンスを作り直すことなく構成を更新できます。また、構成に不具合があれば、CloudFormation が生成するログで問題を調査できます。

テンプレートで、AWS::CloudFormation::Init リソースにインストールと構成の状態を指定します。cfn-init および AWS::CloudFormation::Init の使用方法を記載したウォークスルーについては、「[Amazon EC2 にアプリケーションをデプロイする](#)」を参照してください。

最新のヘルパースクリプトを使用する

CloudFormation ヘルパースクリプトは定期的に更新されます。テンプレートの UserData プロパティに必ず次のコマンドを含めてからヘルパースクリプトを呼び出し、起動したインスタンスで最新のヘルパースクリプトを取得できるようにしてください。

```
yum install -y aws-cfn-bootstrap
```

最新のヘルパースクリプトの取得の詳細については、「CloudFormation テンプレートリファレンスガイド」の「[CloudFormation ヘルパースクリプトリファレンス](#)」を参照してください。

テンプレートを使用する前に検証する

スタックの作成または更新にテンプレートを使用する前に、CloudFormation を使用してテンプレートを検証できます。テンプレートを検証することで、CloudFormation がリソースを作成する前に、依存関係の循環などの構文エラーや意味的エラーを捕捉するのに役立ちます。CloudFormation コンソールを使用する場合は、入力パラメータを指定するとコンソールが自動的にテンプレートを検証します。AWS CLI または CloudFormation API の場合は、[validate-template](#) CLI コマンドまたは [ValidateTemplate](#) API 操作を実行してください。

検証中に、CloudFormation はテンプレートが有効な JSON であるかどうかをまず確認します。そうでない場合は、CloudFormation はテンプレートが有効な YAML であるかどうかを確認します。両方のチェックが失敗すると、CloudFormation はテンプレートの検証エラーを返します。

組織のポリシーに準拠するためにテンプレートを検証する

組織ポリシーガイドラインのコンプライアンス用のテンプレートを検証することもできます。AWS CloudFormation Guard (cfn-guard) は、オープンソースのコマンドラインインターフェース (CLI) ツールです。これは、コードとしてのポリシー言語を提供し、必要なリソースと禁止されたリソースの両方の構成をチェックできるルールを定義します。その後、これらのルールに対してテンプレートを検証できます。たとえば、管理者はルールを作成して、ユーザーが常に暗号化された Amazon S3 バケットを作成するようにできます。

ユーザーは、テンプレートの編集にローカルで、または CI/CD パイプラインの一部として自動的に cfn-guard を使用して、非準拠リソースのデプロイメントを停止できます。

さらに、cfn-guard には、既存の準拠 CloudFormation テンプレートからルールを抽出できる機能 `rulegen` が含まれています。

詳細については、GitHub の [cfn-guard](#) リポジトリを参照してください。

テンプレートオーサリングに YAML または JSON を使用する

CloudFormation は、テンプレート用に YAML 形式と JSON 形式の両方をサポートしています。それぞれの形式に利点があり、どちらを選択するかは特定のニーズに応じて異なります。

次の場合は YAML を使用します。

- 人間による可読性と保守性を優先する
- テンプレートを文書化するためのコメントを含めたい

- ネストされた構造を持つ複雑なテンプレートで作業している
- アンカーやエイリアスといった YAML 固有の機能を使用して反復作業を減らしたい

次の場合は JSON を使用します。

- JSON の使用が好ましいツールやシステムと統合する必要がある
- テンプレートのプログラマ的な生成や操作に取り組んでいる
- 厳密なデータ検証が必要になる

YAML には可読性があり、コメントをサポートすることから、手動でのテンプレートオーサリングに推奨されるのが一般的です。YAML は、インデントベースの構造がリソース階層の視覚化に役立つ、複雑なテンプレートに対して特に有用です。JSON は、自動化されたワークフローや、JSON 入力を期待する API を使用する場合に有利になり得ます。また、特定の構造を厳守する必要がある場合にも役に立ちます。どちらの形式を選択するかにかかわらず、適切に構造化され、文書化されて、保守が可能なテンプレートの作成に焦点を当ててください。YAML を使用する場合は、アンカーやエイリアスといった機能を活用して反復作業を減らし、保守性を向上させます。

包括的なタグ付け戦略を実装する

CloudFormation テンプレートによって作成されるすべてのリソースに対する一貫的なタグ付け戦略を実装します。タグは、リソースの編成、コスト配分、アクセスコントロール、自動化に役立ちます。環境、所有者、コストセンター、アプリケーション、目的に関するタグを含めることを検討してください。

AWS::CloudFormation::Stack リソースの Tags プロパティを使用して、スタック内でサポートされているすべてのリソースにタグを適用します。多数のリソースタイプで利用できる TagSpecifications プロパティを使用して、リソースの作成時にタグを適用することも可能です。

タグ付けの詳細については、「[リソースタグ](#)」を参照してください。

高度な変換にテンプレートマクロを活用する

CloudFormation マクロを使用すると、検索/置換オペレーションといったシンプルなアクションから、追加のリソースを生成する複雑な変換まで、さまざまなカスタム処理をテンプレートで実行できます。マクロを使用して CloudFormation テンプレートの機能を拡張し、組織全体に再利用可能なパターンを実装します。

AWS Serverless Application Model は、サーバーレスアプリケーションの開発を簡素化するマクロの例です。組織固有のパターンと要件に合わせたカスタムマクロの作成を検討してください。

テンプレートでのマクロの使用に関する詳細については、「[CloudFormation マクロの概要](#)」を参照してください。

CloudFormation ですべてのスタックリソースを管理する

スタックを起動した後、CloudFormation [コンソール](#)、[API](#)、[AWS CLI](#) を使用して、スタック内のリソースを更新します。スタックのリソースを CloudFormation 以外の方法で変更しないでください。変更するとスタックのテンプレートとスタックリソースの現在の状態の間で不一致が起これ、スタックの更新または削除でエラーが発生する場合があります。これをドリフトと呼びます。CloudFormation テンプレート外でリソースが変更され、その後にスタックを更新すると、リソースに直接行われた変更は破棄され、リソース設定はテンプレートの設定に戻ります。

ドリフトの詳細については、「[ドリフトとは](#)」を参照してください。

スタックの更新に関する詳細は、「[CloudFormation のスタックを更新する](#)」を参照してください。

スタックを更新する前に変更セットを作成する

変更セットは、スタックの変更案が実行中のリソースに与える影響を、実装前に確認できるようにします。CloudFormation は、変更セットが実行するまでスタックを変更しないため、変更案のまま続行するか、別の変更案を作成するかを決定できます。

変更セットを使用して、変更が実行中のリソース、特に重要なリソースに与える可能性のある影響を確認できます。例えば、Amazon RDS データベースインスタンスの名前を変更すると、CloudFormation によって新しいデータベースが作成され、古いものは削除されます。古いデータベースのデータをバックアップしていないと、データは失われます。変更設定を生成することで、変更によってデータベースが置き換えられることがわかります。このように、スタックを更新する前に計画を立てることができます。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

スタックポリシーを使用してリソースを保護する

スタックポリシーは、重要なスタックリソースを保護して、意図しない更新でリソースが中断されたり置き換えられるのを防ぎます。スタックポリシーは、指定したリソースに対して実行できる更新アクションを記述する JSON ドキュメントです。重要なリソースがあるスタックを作成するときは常に、スタックポリシーを指定してください。

スタックの更新中に、保護されたリソースを明示的に指定して更新する必要があります。指定しない場合は保護されたリソースは変更されません。詳細については、「[スタックのリソースが更新されないようにする](#)」を参照してください。

AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する

AWS CloudTrail は、AWS アカウントで CloudFormation API コールを実行するすべてのユーザーを追跡します。API 呼び出しは、CloudFormation API、CloudFormation コンソール、バックエンドコンソール、または CloudFormation AWS CLI コマンドの使用時にログに記録されます。ログ記録を有効にして Amazon S3 バケットを指定し、ログを保存します。こうすることで、必要ならばアカウントで誰がどのような CloudFormation 呼び出しを行ったかを監査できます。

詳細については、「[AWS CloudTrail による CloudFormation API コールのログ記録](#)」を参照してください。

コードの確認とリビジョン管理を使用してテンプレートを管理する

スタックのテンプレートには、プロパティ値などの AWS リソースの構成が記述されています。リソースの変更を確認し正確な履歴を保持するには、コードの確認とリビジョン管理を使用します。この方法は、テンプレートの異なるバージョン間の変更を追跡するのに役立ちます。また、スタックリソースの変更を追跡するのに役立ちます。また、履歴を保守することで、いつでもスタックをテンプレートの特定のバージョンに戻すことができます。

Amazon EC2 インスタンスを定期的に更新

CloudFormation で作成されたすべての Amazon EC2 Windows インスタンスと Amazon EC2 Linux インスタンスに対して、定期的に `yum update` コマンドを実行して RPM パッケージを更新します。こうすることで、最新の修正およびセキュリティの更新を確実に取得します。

ドリフト検出を定期的に使用する

CloudFormation のドリフト検出機能を定期的に使用して、CloudFormation の管理外で変更されたリソースを特定します。ドリフトの検出と解決は、Infrastructure as Code アプローチの整合性を維持するために役立ち、デプロイされたリソースの状態をテンプレートが正確に反映することを確実にします。

自動ドリフト検出を運用手順の一環として実装することを検討してください。Amazon EventBridge ルールによってトリガーされる AWS Lambda 関数を使用してドリフトを定期的にチェックし、不一致が検出されたときにチームに通知できます。

詳細については、「[ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#)」を参照してください。

自動復旧のためのロールバックトリガーを設定する

ロールバックトリガーを使用して、スタックの作成や更新オペレーション中に CloudFormation が監視する必要のある Amazon CloudWatch アラームを指定します。指定されたアラームのいずれかが ALARM 状態になると、CloudFormation がスタックオペレーション全体を自動的にロールバックして、インフラストラクチャが安定した状態を維持できるようにします。

アプリケーションのエラー率、システムリソースの使用率、またはアプリケーションやインフラストラクチャの正常性を示すカスタムビジネスメトリクスなどの重要なメトリクスに対するロールバックトリガーを設定します。

ロールバックトリガーの詳細については、「[アラーム違反時にスタックをロールバックする](#)」を参照してください。

効果的なスタックリファクタリング戦略を実装する

インフラストラクチャが進化するにつれて、メンテナンス性を向上させ、複雑性を軽減して、変化する要件に適応するために CloudFormation スタックをリファクタリングする必要がある場合があります。スタックリファクタリングには、外部動作と機能を維持しながらテンプレートとリソースを再構成することが含まれます。スタックリファクタリングは、以下のように CloudFormation と使用すると効果的です。

- モノリシックスタックの分割: 大規模で複雑なスタックを、ライフサイクルや所有権別に編成されたより小さく管理しやすいスタックに分割する
- 関連リソースの統合: 複数のスタックからの関連リソースをまとまりのある単一のスタックに統合して管理を簡素化する
- 再利用可能なコンポーネントの抽出: 一般的なパターンをモジュールまたはネストされたスタックに移動して、再利用と一貫性を促進する
- リソース組織の改善: 関係と依存関係をよりよく反映するために、スタック内のリソースを再構成する

CloudFormation スタックのリファクタリングに関する詳細については、「[スタックリファクタリング](#)」を参照してください。

ライフサイクル管理に CloudFormation Hooks を使用する

CloudFormation Hooks は、プロビジョニングする前に AWS リソースの設定をプロアクティブに検査するコードを提供し、複雑な検証チェックを実行します。Hooks は、リソース、スタック、変更セットが組織のセキュリティ、運用、コスト最適化のニーズに準拠しているかどうかをチェックします。設定方法に応じて、Hooks はリソースのプロビジョニング前、またはオペレーションを失敗させて全体を停止する前に警告を発します。違反と警告は、非準拠のデプロイに対する可視性を提供するために Amazon CloudWatch のログに記録されます。

Hooks に関するこれらのベストプラクティスの詳細については、「[AWS CloudFormation Hooks concepts](#)」を参照してください。

CloudFormation リソースに対する Hooks の活用方法の詳細については、「[What are AWS CloudFormation Hooks?](#)」を参照してください。

laC ジェネレーターを使用して既存のリソースからテンプレートを作成する

CloudFormation laC (Infrastructure as Code) ジェネレーターは、既存の AWS リソースからの CloudFormation テンプレートの作成に役立ちます。この機能は、既存のインフラストラクチャをレプリケートする、手動で作成されたリソースを文書化する、またはこれまで管理されていなかったリソースを CloudFormation で管理する必要がある場合に特に便利です。laC ジェネレーターは、以下のような CloudFormation テンプレートの作成に役立ちます

- テンプレートの高速作成: テンプレートを最初から作成する代わりに既存のリソースからテンプレートを生成する
- 一貫的なインフラストラクチャ: 生成されたテンプレートを出発点として使用することで、新しい環境が既存の環境と一致することを確実にする
- Infrastructure as Code への移行: 手動で作成されたリソースを CloudFormation の管理下に段階的に移行する
- 文書化: 既存のインフラストラクチャの記録をテンプレート形式で作成する

laC ジェネレーターの詳細については、「[laC ジェネレーターを使用して既存のリソースからテンプレートを生成する](#)」を参照してください。

ビジュアルテンプレートの設計に AWS Infrastructure Composer を使用する

AWS Infrastructure Composer は、ドラッグアンドドロップ形式のインターフェイスを使用して CloudFormation テンプレートを作成、視覚化、変更できるようにするビジュアル設計ツールです。これは、CloudFormation を次のように使用する場合に特に有益です。

- アーキテクチャの計画: インフラストラクチャアーキテクチャを設計して実装前に検証する
- テンプレートのモダナイズ: 既存のテンプレートを視覚化してテンプレートの構造を理解し、改善機会を特定する
- トレーニングとオンボーディング: 新しいチームメンバーが視覚的な学習を通して CloudFormation の概念や AWS サービスとの関係を理解できるようにする
- ステークホルダーとのコミュニケーション: 明確な視覚的表現を使用して、技術者以外のステークホルダーにインフラストラクチャ設計を提示する
- コンプライアンスレビュー: 視覚的な図表を使用して、インフラストラクチャ設計のセキュリティレビューとコンプライアンスレビューを円滑化する
- コンプライアンスレビュー: 視覚的な図表を使用して、インフラストラクチャ設計のセキュリティレビューとコンプライアンスレビューを円滑化する

Infrastructure Composer の詳細については、「[What is AWS Infrastructure Composer?](#)」を参照してください。

複雑なインフラストラクチャに AWS Cloud Development Kit (AWS CDK) を使用することを検討する

複雑なインフラストラクチャ要件については、TypeScript、Python、Java、.NET などの使い慣れたプログラミング言語を使用してクラウドリソースを定義するために CDK を使用してすることを検討してください。AWS CDK はコードから CloudFormation テンプレートを生成するため、希望する言語の抽象化とプログラミングコンストラクトを使用しながら CloudFormation のすべての機能を活用できます。

AWS CDK は、ベストプラクティスをカプセル化し、一般的なインフラストラクチャパターンの定義を簡素化する高レベルのコンストラクトを提供します。そのため、ベストプラクティスへの準拠を確実にしながら、インフラストラクチャの定義に必要なコードの量を大幅に削減することができます。

CDK の詳細については、「[AWS Cloud Development Kit \(AWS CDK\)](#)」を参照してください。

IAM を使用したアクセス制御

IAM は AWS のユーザーとそのアクセス許可を管理できる AWS サービスです。IAM と CloudFormation を使用して、スタックテンプレートの表示、スタックの作成、スタックの削除など、ユーザーが実行できる CloudFormation アクションを指定できます。さらに、CloudFormation スタックを管理するには、そのスタックのリソースに対するアクセス許可が必要です。例えば、ユーザーが CloudFormation を使用して Amazon EC2 インスタンスを、起動、更新、終了する場合、そのユーザーは関連する Amazon EC2 アクションを呼び出すアクセス許可を持っている必要があります。

ほとんどの場合、ユーザーはテンプレート内のすべてのリソースを管理するためにフルアクセスを必要とします。CloudFormation は、ユーザーに代わってこれらのリソースを作成、変更、削除するための呼び出しを行います。ユーザーと CloudFormation サービスの間でアクセス許可を分割するには、サービスロールを使用します。CloudFormation は、ユーザーのポリシーの代わりにサービスロールのポリシーを使用して呼び出しを行います。詳細については、「[CloudFormation サービスロール](#)」を参照してください。

最小特権の原則を適用する

CloudFormation のサービスロールや、テンプレートから作成されたリソースの IAM ロールを設定するときは、常に最小特権の原則を適用します。意図される機能に必要な許可のみを付与し、ワイルドカード許可の使用はできる限り避けるようにしてください。

IAM Access Analyzer を使用して、CloudFormation のサービスロールに付与された許可を確認し、削除できる未使用の許可を特定します。IAM ポリシーは定期的に見直して更新し、セキュリティ要件に適合していることを確認してください。

機密性の高いパラメータをセキュア化する

パスワード、API キー、その他のシークレットなどの機密情報については、テンプレートに直接埋め込む代わりに AWS Secrets Manager Parameter Store または AWS Systems Manager を使用します。スタックオペレーション中にこれらの値をセキュアに取得するには、テンプレートで動的参照を使用します。

テンプレートでパラメータを使用するときは、機密性の高いパラメータの NoEcho プロパティを `true` に設定して、パラメータの値がコンソール、API 応答、または CLI 出力に表示されないように

します。値をログに記録する可能性のあるその他サービスやリソースにこうした値が渡される場合は、NoEcho を使っても値がログに記録されてしまうことに注意してください。

CloudFormation での AWS Systems Manager Parameter Store の使用に関する詳細については、「[AWS Systems Manager Parameter Store からプレーンテキスト値を取得する](#)」を参照してください。

NoEcho プロパティの使用に関する詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

CloudFormation での AWS Secrets Manager の使用に関する詳細については、「[Create AWS Secrets Manager secrets in AWS CloudFormation](#)」を参照してください。

AWS CloudFormation Guard を使用してポリシーをコードとして実装する

AWS CloudFormation Guard (cfn-guard) は、CloudFormation テンプレートのルールを定義して適用できるようにするオープンソースの Policy-as-Code ツールです。cfn-guard を使用して、テンプレートが組織のポリシー、セキュリティベストプラクティス、ガバナンス要件に準拠することを確認します。

CI/CD パイプラインに cfn-guard を統合して、テンプレートをポリシールールに照らして自動検証してからデプロイします。これは、非準拠のリソースが環境にデプロイされるのを防ぎ、ポリシー違反に関する早期フィードバックを開発者に提供します。

Guard の詳細については、「[What is AWS CloudFormation Guard?](#)」を参照してください。

CloudFormation テンプレートの使用

AWS CloudFormation テンプレートは、スタックの一部として作成、更新、または削除する AWS リソースを定義します。これは複数のセクションで構成されていますが、必須のセクションは [Resources](#) セクションのみであり、このセクションは少なくとも 1 つのリソースを宣言する必要があります。

以下の方法で、ポリシーストアを作成できます。

- AWS Infrastructure Composer – テンプレートを設計するためのビジュアルインターフェイス。
- テキストエディタ – テンプレートを JSON または YAML 構文で記述します。
- IaC ジェネレーター – CloudFormation で現在管理されていないアカウントでプロビジョニングされたリソースからテンプレートを生成します。IaC ジェネレーターは、ご使用のリージョンで Cloud Control API によってサポートされている幅広いリソースタイプに対応しています。

このセクションでは、CloudFormation テンプレートのさまざまなセクションを使用する方法と、スタックテンプレートの作成を開始する方法に関する包括的なガイドを提供します。次のトピックについて説明します。

トピック

- [テンプレートが保存される場所](#)
- [テンプレートの検証](#)
- [テンプレートの開始方法](#)
- [サンプルテンプレート](#)
- [CloudFormation テンプレート形式](#)
- [CloudFormation テンプレートセクション](#)
- [Infrastructure Composer を使用してテンプレートを視覚的に作成](#)
- [AWS CloudFormation 言語サーバー](#)
- [IaC ジェネレーターを使用して既存のリソースからテンプレートを生成する](#)
- [動的参照を使用して他のサービスに格納されている値を取得する](#)
- [擬似パラメータを使用して AWS 値を取得する](#)
- [デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)
- [CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)

- [CloudFormation のチュートリアル](#)
- [CloudFormation テンプレートスニペット](#)
- [CloudFormation を使用して Windows ベースのスタックをデプロイする](#)
- [CloudFormation が提供するリソースタイプの使用によるテンプレートの機能の拡張](#)
- [CloudFormation モジュールを使用してテンプレート全体に含めることができる再利用可能なリソース設定の作成](#)

テンプレートが保存される場所

Amazon S3 バケット

CloudFormation テンプレートは Amazon S3 バケットに保存できます。スタックを作成または更新する際に、テンプレートを直接アップロードする代わりに、テンプレートの S3 URL を指定できます。

AWS マネジメントコンソール または AWS CLI を通じてテンプレートを直接アップロードすると、S3 バケットが自動的に作成されます。詳細については、「[CloudFormation コンソールからスタックを作成する](#)」を参照してください。

Git リポジトリ

[Git 同期](#)を使用すると、Git リポジトリにテンプレートを保存できます。スタックを作成または更新する際に、テンプレートを直接アップロードしたり、S3 URL を参照したりする代わりに、テンプレートを含む Git リポジトリの場所とブランチを指定できます。CloudFormation は、指定されたリポジトリとブランチでテンプレートの変更を自動的にモニタリングします。詳細については、「[Git 同期を使用してリポジトリソースコードからスタックを作成する](#)」を参照してください。

テンプレートの検証

構文の検証

テンプレートの JSON または YAML 構文を検証するには、[validate-template](#) CLI コマンドを使用するか、またはコンソールでテンプレートを指定します。コンソールは検証を自動的に実行します。詳細については、「[CloudFormation コンソールからスタックを作成する](#)」を参照してください。

ただし、これらのメソッドはテンプレートの構文のみを検証し、リソースのために指定したプロパティ値は検証しません。

その他の検証ツール

より複雑な検証とベストプラクティスチェックには、次のような追加のツールを使用できます。

- [CloudFormation Linter \(cfn-lint\)](#) – [CloudFormation リソースプロバイダスキーマ](#)に照らしてテンプレートを検証します。リソースプロパティの有効な値の確認やベストプラクティスが含まれます。
- [CloudFormation Rain \(rain fmt\)](#) – CloudFormation テンプレートを一貫性のある標準にフォーマットするか、または JSON から YAML (もしくは YAML から JSON) にテンプレートを再フォーマットします。YAML を使用する際のコメントを保持し、組み込み関数の使用を短い構文に切り替えます (可能な場合)。

テンプレートの開始方法

CloudFormation テンプレートの作成を開始するには、次のステップに従います。

1. リソースを選択する – EC2 インスタンス、VPC、セキュリティグループなど、スタックに含める AWS リソースを特定します。
2. テンプレートを記述する – テンプレートを JSON または YAML 形式で記述し、リソースとそのプロパティを定義します。
3. テンプレートの保存 – テンプレートは、.json、.yaml、.txt などのファイル拡張子でローカルに保存します。
4. テンプレートを検証する – [テンプレートの検証](#) セクションで説明されている方法を使用してテンプレートを検証します。
5. スタックを作成する – 検証済みテンプレートを使用してスタックを作成します。

CloudFormation テンプレートリファレンスを使用する予定

テンプレートを作成すると、「[AWS リソースおよびプロパティタイプのリファレンス](#)」でさまざまなリソースタイプの詳細な構文に関するドキュメントを確認できます。

多くの場合、スタックテンプレートには、ランタイムまで使用できないプロパティ値を割り当てる組み込み関数と、リソースの動作を制御するための特別な属性が必要です。テンプレートを作成するときは、以下のガイダンスのリソースを参照してください。

- [組み込み関数リファレンス](#) – 一般的に使用される組み込み関数には、次のようなものがあります。

- Ref – パラメータの値またはリソースの物理 ID を取得します。
- Sub – 文字列のプレースホルダーを実際の値に置き換えます。
- GetAtt – テンプレート内のリソースから属性の値を返します。
- Join – 値のセットを 1 つの文字列に結合します。
- [リソース属性リファレンス](#) – 一般的に使用される特殊な属性には、次のようなものがあります。
 - DependsOn – この属性を使用して、あるリソースに続けて別のリソースを作成する必要があることを指定できます。
 - DeletionPolicy – この属性を使用して、リソースの削除を CloudFormation でどのように処理するかを指定します。

サンプルテンプレート

CloudFormation は、使用を開始するために使用できるオープンソースのスタックテンプレートを提供します。詳細については、GitHub ウェブサイトの「[CloudFormation Sample Templates](#)」を参照してください。

これらのテンプレートは本番用ではないことに留意してください。どのように機能するかを知り、ニーズに適応させ、会社のコンプライアンス基準を満たしているようにするために時間を設けるべきです。

このリポジトリ内の各テンプレートは、[CloudFormation Linter](#) (cfn-lint) チェックに合格します。また、Center for Internet Security (CIS) の上位 20 個の AWS CloudFormation Guard ルールの基本的なセットにも合格します。ただし、サンプルの焦点を 1 つのユースケースに置くことが理にかなっている一部のルールは例外です。

CloudFormation テンプレート形式

CloudFormation テンプレートは JSON または YAML 形式で作成できます。いずれの形式も同じ目的に資するものですが、読みやすさと複雑さの点で明確に異なる利点があります。

- JSON – JSON は、マシンが解析して生成しやすい軽量のデータ交換形式です。ただし、特に複雑な設定では、人間による読み書きが煩雑になる可能性があります。JSON では、テンプレートはネストされた中括弧 {} と括弧 [] を使用して構築され、リソース、パラメータ、および他のコンポーネントを定義します。この構文では、すべての要素を明示的に宣言する必要があります。これにより、テンプレートは冗長になる可能性があります。構造化された形式への厳密な準拠を実現できます。

- YAML – YAML は、JSON と比較して、人間にとってより読みやすく、冗長性がより低くなるように設計されています。中括弧や括弧ではなくインデントを使用してネストを示すため、リソースとパラメータの階層をより簡単に視覚化できます。YAML は、特により複雑なテンプレートを扱う場合、その明確さと使いやすさを理由としてより好まれることがよくあります。ただし、YAML によるインデントへの依拠により、スペーシングに一貫性がない場合にエラーが発生する可能性があるため、精度を維持するには細心の注意が必要です。

テンプレート構造

CloudFormation テンプレートは、さまざまなセクションに分かれており、各セクションは特定のタイプの情報を保持するように設計されています。特定の順序で宣言する必要があるセクションと、順序は重要でないセクションがあります。ただし、あるセクションの値が前のセクションの値を参照している場合があるため、テンプレートを構築する際には、次の例で示されている論理的な順序を使用すると便利です。

テンプレートを作成するときは、Resources セクションなどの主要セクションを重複して使用しないでください。CloudFormation はテンプレートを受け入れる場合がありますが、テンプレートを処理するときに未定義の動作が発生し、リソースを誤ってプロビジョニングしたり、不可解なエラーを返したりする可能性があります。

JSON

次の例は、使用可能なすべてのセクションを含む JSON 形式のテンプレートの構造を示しています。

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Rules" : {
    set of rules
  }
```

```
},  
  
"Mappings" : {  
    set of mappings  
},  
  
"Conditions" : {  
    set of conditions  
},  
  
"Transform" : {  
    set of transforms  
},  
  
"Resources" : {  
    set of resources  
},  
  
"Outputs" : {  
    set of outputs  
}  
}
```

YAML

次の例は、使用可能なすべてのセクションを含む YAML 形式のテンプレートの構造を示しています。

```
---  
AWSTemplateFormatVersion: version date  
  
Description:  
    String  
  
Metadata:  
    template metadata  
  
Parameters:  
    set of parameters  
  
Rules:  
    set of rules
```

Mappings:
set of mappings

Conditions:
set of conditions

Transform:
set of transforms

Resources:
set of resources

Outputs:
set of outputs

コメント

JSON 形式のテンプレートでは、コメントはサポートされていません。JSON には、設計上、コメントの構文は含まれていません。つまり、JSON 構造内にコメントを直接追加することはできません。ただし、説明のためのメモやドキュメントを含める必要がある場合は、メタデータを追加することを検討できます。詳細については、「[Metadata 属性](#)」を参照してください。

YAML 形式のテンプレートでは、# 記号を使用してインラインコメントを含めることができます。

以下は、インラインコメントを使用した YAML テンプレートの例です。

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample CloudFormation template with YAML comments.
# Resources section
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      # Linux AMI
      ImageId: ami-1234567890abcdef0
      InstanceType: t2.micro
      KeyName: MyKey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
```

VolumeSize: 20

仕様

CloudFormation は、次の JSON および YAML の仕様をサポートしています。

JSON

CloudFormation は ECMA-404 JSON スタンドアードに従います。JSON フォーマットの詳細については、<http://www.json.org> をご覧ください。

YAML

CloudFormation は YAML バージョン 1.1 仕様をサポートしていますが、いくつかの例外があります。CloudFormation は次の機能をサポートしていません。

- binary、omap、pairs、set、およびtimestamp タグ
- エイリアス
- ハッシュ結合

YAML の詳細については、<https://yaml.org/> を参照してください。

詳細はこちら

テンプレート内で指定する各リソースについて、JSON または YAML のいずれかの構文規則に従って、プロパティとその値を定義します。各形式のテンプレート構文の詳細については、[CloudFormation テンプレートセクション](#) を参照してください。

CloudFormation テンプレートでの正規表現の使用

CloudFormation テンプレート内では、正規表現 (regex と呼ばれることもあります) をさまざまな場所で使用できます (例えば、テンプレート[パラメータ](#)の作成時の AllowedPattern プロパティ)。

CloudFormation の正規表現はすべて、Java の正規表現構文に準拠しています。Java の正規表現構文とそのコンストラクトの包括的な説明については、「[java.util.regex.Pattern](#)」を参照してください。

CloudFormation テンプレートを JSON 構文で記述する場合は、追加のバックスラッシュを追加して、正規表現のバックスラッシュ文字 (\) をエスケープする必要があります。これは、JSON がバックスラッシュをエスケープ文字として解釈するため、正規表現でリテラルバックスラッシュとして扱われるようにエスケープする必要があるためです。

たとえば、数字を示す `\d` を正規表現に含める場合、JSON テンプレートでは `\\d` と記述する必要があります。

次の例では、`AllowedPattern` プロパティは、4 つの連続した数字 (`\d{4}`) に一致する正規表現を指定します。ただし、正規表現は JSON テンプレートで定義されているため、バックスラッシュ文字は追加のバックスラッシュ (`\\d`) でエスケープする必要があります。

```
{
  "Parameters": {
    "MyParameter": {
      "Type": "String",
      "AllowedPattern": "\\d{4}"
    }
  }
}
```

CloudFormation テンプレートを YAML 構文で記述する場合は、正規表現を一重引用符 (") で囲む必要があります。追加のエスケープは必要ありません。

```
Parameters:
  MyParameter:
    Type: String
    AllowedPattern: '\d{4}'
```

Note

CloudFormation の正規表現は、`AllowedPattern` などの特定のコンテキストでの検証目的でのみサポートされます。これらは、パターンマッチングではなく正確な文字列比較のみを実行する `Fn::Equals` などの CloudFormation 組み込み関数のパターンマッチングオペレーションとしてはサポートされていません。

CloudFormation テンプレートセクション

あらゆる CloudFormation テンプレートは 1 つ以上のセクションで構成され、それぞれが特定の目的を果たします。

[リソース] セクションはあらゆる CloudFormation テンプレートで必要であり、テンプレートの中核となります。このセクションは、Amazon EC2 インスタンスや Amazon S3 バケットなど、スタック

リソースとそのプロパティを指定します。各リソースは、一意の論理 ID、タイプ、および特定の設定の詳細で定義されます。

[パラメータ] セクションはオプションですが、テンプレートをより柔軟にする上で重要な役割を果たします。これにより、ユーザーはスタックを作成または更新する際、実行時に値を渡すことができます。これらのパラメータは Resources および Outputs セクションで参照できるため、テンプレート自体の変更なしでのカスタマイズが可能です。例えば、パラメータを使用して、デプロイによって異なるインスタンスタイプまたは環境設定を指定できます。

[出力] セクションもオプションで、スタックのプロパティを表示する際に返される値を定義します。出力は、リソース識別子や URL などの有用な情報を提供します。これらの情報は、運用上の目的や他のスタックとの統合のために活用できます。このセクションでは、テンプレートによって作成されたリソースに関する重要な詳細を取得して使用する方法について説明します。

他のオプションのセクションには、条件値を管理するためのルックアップテーブルのように機能する [マッピング] が含まれます。マッピングでは、key-value ペアを定義し、Resources および Outputs セクションの `Fn::FindInMap` 組み込み関数で使います。これは、AWS リージョン や環境などの条件に基づいて設定を調整する必要があるシナリオで役立ちます。

[メタデータ] と [ルール] のセクションは、あまり一般的に使用されてはいませんが、追加機能を提供します。Metadata はテンプレートに関する追加情報を含むことができ、Rules はスタックの作成または更新中にパラメータ、またはパラメータの組み合わせを検証し、特定の基準を満たしていることを確認します。[条件] セクションは、環境タイプなどの条件に基づいて特定のリソースが作成されるか、またはプロパティに値が割り当てられるかを制御することで、柔軟性をさらに高めます。

最後に、[変換] セクションを使用して、テンプレートの処理中にマクロを適用します。サーバーレスアプリケーション (Lambda アプリケーションとも呼ばれる) の場合、使用する [AWS サーバーレスアプリケーションモデル \(AWSSAM\)](#) のバージョンを指定します。変換を指定する場合は、AWS SAM 構文を使用して、テンプレート内のリソースを宣言できます。このモデルでは、使用できる構文と、その処理方法を定義します。AWS::Include トランスフォームを使用すると、メインの CloudFormation テンプレートとは別に保存されているテンプレートスニペットを含めるもできます。

次のトピックでは、各セクションの使用に関する詳細と例を示します。

トピック

- [CloudFormation テンプレートの Resources 構文](#)
- [CloudFormation テンプレートの Parameters 構文](#)

- [CloudFormation テンプレートの Outputs 構文](#)
- [CloudFormation テンプレートの Mappings 構文](#)
- [CloudFormation テンプレートの Metadata 構文](#)
- [CloudFormation テンプレートの Rules 構文](#)
- [CloudFormation テンプレートの Conditions 構文](#)
- [CloudFormation テンプレートの Transform セクション](#)
- [CloudFormation テンプレート形式のバージョン構文](#)
- [CloudFormation テンプレートの Description 構文](#)

CloudFormation テンプレートの Resources 構文

Resources セクションは、CloudFormation テンプレートで必須の最上位セクションです。CloudFormation にスタックの一部としてプロビジョニングおよび設定させる AWS リソースを宣言します。

構文

Resources セクションでは、次の構文を使用します。

JSON

```
"Resources" : {  
  "LogicalResourceName1" : {  
    "Type" : "AWS::ServiceName::ResourceType",  
    "Properties" : {  
      "PropertyName1" : "PropertyValue1",  
      ...  
    }  
  },  
  
  "LogicalResourceName2" : {  
    "Type" : "AWS::ServiceName::ResourceType",  
    "Properties" : {  
      "PropertyName1" : "PropertyValue1",  
      ...  
    }  
  }  
}
```


YAML

```
Resources:
  LogicalResourceName1:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
      ...

  LogicalResourceName2:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
      ...
```

論理 ID (論理名とも呼ばれます)

CloudFormation テンプレート内では、リソースは論理リソース名によって識別されます。これら名前は英数字 (A~Z、a~z、0~9) でなければならず、テンプレート内で一意である必要があります。論理名は、テンプレートの他のセクションのリソースを参照するために使用されます。

リソースタイプ

リソースには、AWS リソースの種類を定義する Type 属性が必要です。Type 属性の形式は `AWS::ServiceName::ResourceType` です。例えば、Amazon S3 バケットの Type 属性は `AWS::S3::Bucket` です。

サポートされているリソースタイプの一覧については、「[AWS リソースおよびプロパティタイプのリファレンス](#)」を参照してください。

リソースプロパティ

リソースプロパティは、特定のリソースタイプの設定の詳細を定義するために指定できる追加オプションです。一部のプロパティは必須で、他のプロパティはオプションです。デフォルト値が設定されたプロパティもあり、そうしたプロパティを指定するのは任意です。

各リソースタイプのためにサポートされているプロパティの詳細については、「[AWS リソースおよびプロパティタイプのリファレンス](#)」のトピックを参照してください。

プロパティの値には、リテラル文字列、文字列のリスト、ブール値、動的参照、パラメータ参照、疑似参照、または関数によって返される値を使用できます。次の例は、異なるプロパティ値の型を宣言する方法を示しています。

JSON

```
"Properties" : {  
  "String" : "A string value",  
  "Number" : 123,  
  "LiteralList" : [ "first-value", "second-value" ],  
  "Boolean" : true  
}
```

YAML

```
Properties:  
  String: A string value  
  Number: 123  
  LiteralList:  
    - first-value  
    - second-value  
  Boolean: true
```

物理 ID

特定のリソースは、論理 ID の他に物理 ID も持っています。これは、そのリソースに実際に割り当てられている名前です (EC2 インスタンス ID、S3 バケット名など)。CloudFormation テンプレートに記載していないリソースを識別するには物理 ID を使用しますが、リソースの作成後にのみ使用できます。たとえば、EC2 インスタンスリソースに MyEC2Instance の論理 ID を指定します。CloudFormation は、インスタンスを作成すると、自動的に物理 ID (i-1234567890abcdef0) を生成してインスタンスに割り当てます。この物理 ID を使用して、Amazon EC2 コンソールでインスタンスを特定したり、そのプロパティ (DNS 名など) を表示したりすることができます。

Amazon S3 バケットをはじめとする各種リソースに明示的に物理名を指定していない場合は、CloudFormation が一意の物理名を自動的に生成します。CloudFormation スタックの名前、CloudFormation テンプレートに指定されているリソースの論理名、一意の ID を組み合わせた物理名です。例えば、MyStack というスタックに論理名が MyBucket の Amazon S3 バケットがある場合、そのバケットに物理名 MyStack-MyBucket-abcdefghijkl1 を含む名前を付けます。

カスタム名をサポートしているリソースの場合は、リソースを迅速に識別するために、独自の物理名を割り当てることができます。たとえば、ログを保存する S3 バケットに MyPerformanceLogs という名前を指定できます。詳細については、「[Name タイプ](#)」を参照してください。

リソースの参照

あるリソースのプロパティを設定する際に、別のリソースの名前やプロパティに基づいて設定しなければならないことがよくあります。例えば、EC2 セキュリティグループを使用する EC2 インスタンス、または S3 バケットを利用する CloudFront ディストリビューションを作成できます。これらのリソースはすべて、同じ CloudFormation テンプレートで作成できます。

CloudFormation には、他のリソースとそのプロパティを参照するための組み込み関数が用意されています。こうした組み込み関数を使用すると、リソース間に依存関係を確立し、リソース間で値を渡すことができます。

Ref 関数

同じ CloudFormation テンプレート内に定義されたリソースを識別できるプロパティを取得するには、Ref 関数を一般的に使用します。返される内容は、リソースのタイプによって異なります。ほとんどのリソースの場合、リソースの物理名が返されます。ただし、リソースタイプによっては、AWS::EC2::EIP リソースの IP アドレスや Amazon SNS トピックの Amazon リソースネーム (ARN) といった別の値が返されることもあります。

次の例では、プロパティで Ref 関数を使用する方法を示します。これらの各例では、Ref 関数はテンプレートの他の場所で宣言された LogicalResourceName リソースの実際の名前を返します。YAML 例の !Ref 構文例は、Ref 関数をより短く記述する方法です。

JSON

```
"Properties" : {  
  "PropertyName" : { "Ref" : "LogicalResourceName" }  
}
```

YAML

```
Properties:  
  PropertyName1:  
    Ref: LogicalResourceName  
  PropertyName2: !Ref LogicalResourceName
```

Ref 関数の詳細については、「[Ref](#)」を参照してください。

Fn::GetAtt 関数

Ref 関数は、リソースに対して返されるパラメータまたは値が目的の値そのものである場合に役立ちます。ただし、リソースの他の属性が必要になる場合もあります。たとえば、S3 オリジンを含む CloudFront ディストリビューションを作成するには、DNS 形式のアドレスを使用してバケットの場所を指定する必要があります。いくつかのリソースには、その値をテンプレートで利用できる追加の属性があります。にこれらの属性を取得するには、Fn::GetAtt 関数を使用します。

次の例では、プロパティで GetAtt 関数を使用する方法を示します。Fn::GetAtt 関数は、2 つのパラメーターを受け取ります。1 つはリソースの論理名、もう 1 つは取得する属性の名前です。YAML 例の !GetAtt 構文例は、GetAtt 関数をより短く記述する方法です。

JSON

```
"Properties" : {  
  "PropertyName" : {  
    "Fn::GetAtt" : [ "LogicalResourceName", "AttributeName" ]  
  }  
}
```

YAML

```
Properties:  
  PropertyName1:  
    Fn::GetAtt:  
      - LogicalResourceName  
      - AttributeName  
  PropertyName2: !GetAtt LogicalResourceName.AttributeName
```

GetAtt 関数の詳細については、「[Fn::GetAtt](#)」を参照してください。

例

次の例は、リソースを宣言する方法と、CloudFormation テンプレートが同じテンプレートと既存の AWS リソース内で定義された他のリソースを参照する方法を示しています。

トピック

- [カスタム名で単一のリソースを宣言する](#)
- [Ref 関数を使用した他のリソースの参照](#)

- [Fn::GetAtt 関数を使用したリソース属性の参照](#)

カスタム名で単一のリソースを宣言する

次の例では、論理名 `MyBucket` を持つ `AWS::S3::Bucket` タイプの 1 のリソースが宣言されます。BucketName プロパティは `amzn-s3-demo-bucket` に設定されます。これは、S3 バケットに必要な名前に置き換える必要があります。

このリソース宣言を使用してスタックを作成すると、CloudFormation がデフォルト設定の Amazon S3 バケットを作成します。Amazon EC2 インスタンス や Auto Scaling グループのようなその他のリソースの場合、CloudFormation はさらに多くの情報を必要とします。

JSON

```
{
  "Resources": {
    "MyBucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {
        "BucketName": "amzn-s3-demo-bucket"
      }
    }
  }
}
```

YAML

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
```

Ref 関数を使用した他のリソースの参照

次の例は、EC2 インスタンスとセキュリティグループを定義するリソース宣言を示しています。Ec2Instance リソースは、Ref 関数を使用して、SecurityGroupIds プロパティの一部として InstanceSecurityGroup リソースを参照します。また、テンプレートで宣言されていない既存のセキュリティグループ (sg-12a4c434) も含まれます。既存の AWS リソースを参照するには、リテラル文字列を使用します。

JSON

```
{
  "Resources": {
    "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "SecurityGroupIds": [
          {
            "Ref": "InstanceSecurityGroup"
          },
          "sg-12a4c434"
        ],
        "KeyName": "MyKey",
        "ImageId": "ami-1234567890abcdef0"
      }
    },
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Enable SSH access via port 22",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    }
  }
}
```

YAML

```
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
        - sg-12a4c434
```

```
KeyName: MyKey
ImageId: ami-1234567890abcdef0
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
```

Fn::GetAtt 関数を使用したリソース属性の参照

次の例は、CloudFront ディストリビューションリソースと S3 バケットを定義するリソース宣言を示しています。MyDistribution リソースは、バケットの DomainName 属性を取得するための Fn::GetAtt 関数を使用して、MyBucket リソースの DNS 名を指定します。Fn::GetAtt 関数では、2 つのパラメーターが配列として指定されています。複数のパラメーターを受け取る関数に対しては、配列を使用してパラメーターを指定します。

JSON

```
{
  "Resources": {
    "MyBucket": {
      "Type": "AWS::S3::Bucket"
    },
    "MyDistribution": {
      "Type": "AWS::CloudFront::Distribution",
      "Properties": {
        "DistributionConfig": {
          "Origins": [
            {
              "DomainName": {
                "Fn::GetAtt": [
                  "MyBucket",
                  "DomainName"
                ]
              },
            },
            {
              "Id": "MyS3Origin",
              "S3OriginConfig": {}
            }
          ],
        }
      }
    }
  }
}
```

```

        "Enabled": "true",
        "DefaultCacheBehavior": {
            "TargetOriginId": "MyS3Origin",
            "ForwardedValues": {
                "QueryString": "false"
            },
            "ViewerProtocolPolicy": "allow-all"
        }
    }
}
}
}
}
}
}
}

```

YAML

```

Resources:
  MyBucket:
    Type: AWS::S3::Bucket
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: !GetAtt
            - MyBucket
            - DomainName
          Id: MyS3Origin
          S3OriginConfig: {}
        Enabled: 'true'
      DefaultCacheBehavior:
        TargetOriginId: MyS3Origin
        ForwardedValues:
          QueryString: 'false'
        ViewerProtocolPolicy: allow-all

```

CloudFormation テンプレートの Parameters 構文

オプションの Parameters セクションを使用して、テンプレートをカスタマイズします。パラメータを使用すると、スタックを作成または更新するたびにテンプレートにカスタム値を入力できます。テンプレートでパラメータを使用することで、特定のシナリオに合わせて調整できる再利用可能で柔軟なテンプレートを構築できます。

適切なタイプのパラメータを定義することにより、コンソールを使用してスタックを作成するとき、既存リソースの識別子のリストから選択できるようになります。詳細については、「[CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)」を参照してください。

パラメータは、スタックリソースのプロパティ値を指定する一般的な方法です。ただし、リージョンに依存する設定もあれば、他の条件や依存関係のためにユーザーには理解しにくい複雑な設定もあります。このような場合、マッピングを使用するなどして、ユーザーがより単純な値を指定して (またはまったく値を指定しないで) 必要な結果を得ることができるよう、テンプレート自体に何らかのロジックを組み込むことをお勧めします。詳細については、「[CloudFormation テンプレートの Mappings 構文](#)」を参照してください。

構文

テンプレートの Parameters セクションでパラメータを宣言します。これは、次の一般的な構文を使用します。

JSON

```
"Parameters" : {  
  "ParameterLogicalID" : {  
    "Description": "Information about the parameter",  
    "Type" : "DataType",  
    "Default" : "value",  
    "AllowedValues" : ["value1", "value2"]  
  }  
}
```

YAML

```
Parameters:  
  ParameterLogicalID:  
    Description: Information about the parameter  
    Type: DataType  
    Default: value  
    AllowedValues:  
      - value1  
      - value2
```

パラメーターは、値と値に対する制約を定義する属性のリストを含みます。必須属性は Type だけで、String や Number のほか、CloudFormation に用意されているパラメータタイプを設定できま

す。どのような値を指定するかを説明する `Description` 属性を追加することもできます。[Create Stack] ウィザードでテンプレートを使用すると、パラメータの名前と説明が [Specify Parameters] ページに表示されます。

Note

デフォルトでは、CloudFormation コンソールでは入力パラメータが論理 ID によりアルファベット順に一覧表示されます。このデフォルトの順序付けを上書きし、関連するパラメータをグループ化するには、テンプレートで `AWS::CloudFormation::Interface` メタデータキーを使用します。詳細については、「[AWS::CloudFormation::Interface メタデータを使用した CloudFormation パラメータの整理](#)」を参照してください。

デフォルト値を持つパラメータについては、ユーザーが別の値を指定しない限り、CloudFormation でデフォルト値が使用されます。デフォルトの属性を省略する場合は、ユーザーにそのパラメータの値を指定するよう求める必要があります。ただし、ユーザーに値を指定するよう求めた場合、その値の有効性は保証されません。パラメータの値を検証するには、制限を宣言するか、AWS 固有のパラメータタイプを指定することができます。

デフォルト値のないパラメータの場合、ユーザーはスタックの作成時にキー名の値を指定する必要があります。指定しないと、CloudFormation はスタックを作成できないため、例外をスローします。

```
Parameters: [KeyName] must have values
```

プロパティ

AllowedPattern

String または CommaDelimitedList タイプに使用できるパターンを表す正規表現。タイプ String のパラメータに適用される場合、パターンは指定されたパラメータ値全体と一致する必要があります。タイプ CommaDelimitedList のパラメータに適用される場合、パターンはリスト内の各値と一致する必要があります。

必須: いいえ

AllowedValues

パラメーターに許容される一連の値を含む配列。タイプ String のパラメータに適用する場合、パラメータ値は許可された値のいずれかである必要があります。タイプ CommaDelimitedList

のパラメータに適用される場合、リスト内の各値は、指定および許可された値のいずれかである必要があります。

必須: いいえ

 Note

YAML を使用していて、AllowedValues に Yes と No 文字列を使用する場合、一重引用符を使用して YAML パーサーがこれらのブール値を考慮することを防ぎます。

ConstraintDescription

制約が違反された場合に、制約について説明する文字列。たとえば、制約の説明を指定しないとき、許容されているパターンが `[A-Za-z0-9]+` であるパラメーターの場合、ユーザーが無効な値を指定すると次のエラーメッセージが表示されます。

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

`must only contain letters (uppercase and lowercase) and numbers` などの制約の説明を追加することによって、次のようにカスタマイズされたエラーメッセージを表示することができます。

```
Malformed input-Parameter MyParameter must only contain uppercase and lowercase letters and numbers
```

必須: いいえ

Default

スタックの作成時に値を指定しなかった場合に、テンプレートで使用される適切な型の値。パラメーターの制約を定義する場合は、これらの制約に従う値を指定する必要があります。

必須: いいえ

Description

パラメーターについて説明する最大 4000 文字の文字列。

必須: いいえ

MaxLength

String 型に使用できる最大文字数を決定する整数値。

必須: いいえ

MaxValue

Number 型に使用できる数値の最大値を決定する数値。

必須: いいえ

MinLength

String 型に使用できる最小文字数を決定する整数値。

必須: いいえ

MinValue

Number 型に使用できる数値の最小値を決定する数値。

必須: いいえ

NoEcho

パラメータ値をマスクして、コンソール、コマンドラインツール、または API に表示されないようにするかどうか。NoEcho 属性を true に設定すると、CloudFormation は、スタックまたはスタックイベントを記述するすべての呼び出しに対して、アスタリスク (*****) としてマスクされたパラメータ値を返します。ただし、以下に指定された場所に保存されている情報は除きます。

必須: いいえ

Important

NoEcho 属性を使用しても、以下に保存されている情報はマスクされません。

- Metadata テンプレートセクション。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。詳細については、「[メタデータ](#)」を参照してください。
- Outputs テンプレートセクション。詳細については、「[出力](#)」を参照してください。
- リソース定義の Metadata 属性。詳細については、「[Metadata 属性](#)」を参照してください。

パスワードやシークレットなどの機密情報を含めるには、これらのメカニズムを使用しないことを強くお勧めします。

 Important

機密情報は、CloudFormation テンプレートに直接埋め込むのではなく、スタックテンプレートの動的パラメータを使用して CloudFormation の外部 (AWS Systems Manager パラメータストアや AWS Secrets Manager など) に保存して管理した上で 参照することをお勧めします。

詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティスを参照してください。

 Important

リソースのプライマリ識別子の一部であるリソースプロパティには、NoEcho パラメータや機密データを含めないことを強くお勧めします。

プライマリリソース識別子を形成するプロパティに NoEcho パラメータが含まれている場合、CloudFormation はプライマリリソース識別子に実際の平文値を使用する場合があります。このリソース ID は、派生した出力または送信先に表示されます。

リソースタイプのプライマリ識別子を構成するリソースプロパティを確認するには、「[AWS リソースおよびプロパティタイプのリファレンス](#)」でそのリソースのリソースリファレンスドキュメントを参照してください。[Return values] (戻り値) セクションの Ref 関数の戻り値は、リソースタイプのプライマリ識別子を構成するリソースプロパティを表します。

Type

パラメーターのデータ型 (DataType)。

必須: はい

CloudFormation では次のパラメータタイプをサポートしています。

String

リテラル文字

列。MinLength、MaxLength、Default、AllowedValues、AllowedPattern の各属性を使用して制約を宣言できます。

たとえば、次のように指定します。"MyUserName"

Number

整数または浮動小数点値。CloudFormation は、このパラメータを数値として検証しますが、テンプレート内の他の場所で使用した場合には (Ref 組み込み関数を使用した場合など) 文字列として扱います。

MinValue、MaxValue、Default、AllowedValues の各属性を使用して制約を宣言できます。

たとえば、次のように指定します。"8888"

List<Number>

カンマで区切られた整数または浮動小数点値の配列。CloudFormation は、このパラメータを数値として検証しますが、テンプレート内の他の場所で使用した場合には (Ref 組み込み関数を使用した場合など) 文字列のリストとして扱います。

たとえば、"80,20" と指定し、Ref を使用した場合には ["80","20"] となります。

CommaDelimitedList

カンマで区切られたリテラル文字列の配列。文字列の合計数は、カンマの合計数よりも 1 つ多いはずです。また、各メンバー文字列の前後の空白は削除されます。

たとえば、"test,dev,prod" と指定し、Ref を使用した場合には ["test","dev","prod"] となります。

AWS 固有のパラメータタイプ

Amazon EC2 キーペアの名前や VPC の ID などの AWS の値です。詳細については、「[実行時に既存のリソースを指定する](#)」を参照してください。

Systems Manager パラメータタイプ

Systems Manager パラメータストア内の既存のパラメーターに対応するパラメーター。Systems Manager パラメータキーは Systems Manager パラメータタイプの値として指定します。CloudFormation は Parameter Store から最新の値を取得し、スタックに使用します。詳細については、「[実行時に既存のリソースを指定する](#)」を参照してください。

パラメーターの一般要件

パラメーターを使用するときに以下の要件が適用されます。

- CloudFormation テンプレートに指定できるパラメータは最大 200 個です。

- 各パラメータには、英数字でテンプレート内のどの論理名とも重複しない論理名 (論理 ID と呼ばれます) を指定する必要があります。
- 各パラメータには、CloudFormation でサポートされているパラメータタイプを割り当てる必要があります。詳細については、「[Type](#)」を参照してください。
- CloudFormation がスタックを正常にプロビジョニングするには、各パラメータに実行時に値を割り当てる必要があります。別の値が指定されている場合を除き、必要に応じて、使用する CloudFormation のデフォルト値を指定できます。
- パラメーターは、同じテンプレート内から宣言および参照する必要があります。テンプレートの Resources および Outputs セクションのパラメーターを参照できます。

例

トピック

- [シンプルで文字列パラメータ](#)
- [パスワードパラメータ](#)
- [パラメータの参照](#)
- [カンマ区切りリストのパラメータ](#)
- [カンマ区切りリストのパラメータから値を返します](#)

シンプルで文字列パラメータ

以下の例では、String タイプの InstanceTypeParameter というパラメータを宣言します。このパラメータを使用すると、スタックの Amazon EC2 インスタンスタイプを指定できます。スタックの作成時または更新時に値が指定されていない場合、CloudFormation はデフォルト値の t2.micro を使用します。

JSON

```
"Parameters" : {
  "InstanceTypeParameter" : {
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro.",
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"]
  }
}
```

YAML

```
Parameters:
  InstanceTypeParameter:
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
```

パスワードパラメータ

次の例では、デフォルト値のない String タイプの DBPwd というパラメータを宣言します。NoEcho プロパティは true に設定され、スタックの説明にパラメータ値が表示されないようにします。指定できる最小値は 1、最大値は 41 です。パターンには、英文字の大文字と小文字、および数字を使用できます。この例では、AllowedPattern プロパティに正規表現を使用する方法も示しています。

JSON

```
"Parameters" : {
  "DBPwd" : {
    "NoEcho" : "true",
    "Description" : "The database admin account password",
    "Type" : "String",
    "MinLength" : "1",
    "MaxLength" : "41",
    "AllowedPattern" : "^[a-zA-Z0-9]*$"
  }
}
```

YAML

```
Parameters:
  DBPwd:
    NoEcho: true
    Description: The database admin account password
    Type: String
    MinLength: 1
    MaxLength: 41
```



```
AllowedPattern: ^[a-zA-Z0-9]*$
```

パラメータの参照

パラメータを参照するには、Ref 組み込み関数を使用します。CloudFormation は、パラメータの値を使用してスタックをプロビジョニングします。同じテンプレートの Resources および Outputs セクションのパラメーターを参照できます。

次の例では、EC2 インスタンスリソースの InstanceType プロパティが InstanceTypeParameter パラメーター値を参照します。

JSON

```
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },
    "ImageId" : "ami-0ff8a91507f77f867"
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType:
      Ref: InstanceTypeParameter
    ImageId: ami-0ff8a91507f77f867
```

カンマ区切りリストのパラメータ

CommaDelimitedList パラメータタイプは、1 つのプロパティに複数の値を指定する必要がある場合に役立ちます。次の例では、デフォルト値 (3 つの CIDR ブロックがカンマで区切られている) を使用した DbSubnetIpBlocks というパラメータを宣言します。

JSON

```
"Parameters" : {
  "DbSubnetIpBlocks": {
    "Description": "Comma-delimited list of three CIDR blocks",
    "Type": "CommaDelimitedList",
```

```
    "Default": "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
  }
}
```

YAML

```
Parameters:
  DbSubnetIpBlocks:
    Description: "Comma-delimited list of three CIDR blocks"
    Type: CommaDelimitedList
    Default: "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
```

カンマ区切りリストのパラメータから値を返します

パラメータのカンマ区切りリストから特定の値を参照するには、テンプレートの `Resources` セクションで、`Fn::Select` 組み込み関数を使用します。次の例のように、必要なオブジェクトのインデックス値とオブジェクトのリストを渡します。

JSON

```
{
  "Parameters": {
    "VPC": {
      "Type": "String",
      "Default": "vpc-123456"
    },
    "VpcAzs": {
      "Type": "CommaDelimitedList",
      "Default": "us-west-2a, us-west-2b, us-west-2c"
    },
    "DbSubnetIpBlocks": {
      "Type": "CommaDelimitedList",
      "Default": "172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26"
    }
  },
  "Resources": {
    "DbSubnet1": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "AvailabilityZone": {
          "Fn::Select": [
            0,
            {

```

```

        "Ref": "VpcAzs"
      }
    ]
  },
  "VpcId": {
    "Ref": "VPC"
  },
  "CidrBlock": {
    "Fn::Select": [
      0,
      { "Ref": "DbSubnetIpBlocks" }
    ]
  }
},
"DbSubnet2": {
  "Type": "AWS::EC2::Subnet",
  "Properties": {
    "AvailabilityZone": {
      "Fn::Sub": [
        "${AWS::Region}${AZ}",
        {
          "AZ": {
            "Fn::Select": [
              1,
              { "Ref": "VpcAzs" }
            ]
          }
        }
      ]
    },
    "VpcId": {
      "Ref": "VPC"
    },
    "CidrBlock": {
      "Fn::Select": [
        1,
        { "Ref": "DbSubnetIpBlocks" }
      ]
    }
  }
},
"DbSubnet3": {
  "Type": "AWS::EC2::Subnet",

```

```

    "Properties": {
      "AvailabilityZone": {
        "Fn::Sub": [
          "${AWS::Region}${AZ}",
          {
            "AZ": {
              "Fn::Select": [
                2,
                { "Ref": "VpcAzs" }
              ]
            }
          }
        ],
        "VpcId": {
          "Ref": "VPC"
        },
        "CidrBlock": {
          "Fn::Select": [
            2,
            { "Ref": "DbSubnetIpBlocks" }
          ]
        }
      }
    }
  }
}

```

YAML

```

Parameters:
  VPC:
    Type: String
    Default: vpc-123456
  VpcAzs:
    Type: CommaDelimitedList
    Default: us-west-2a, us-west-2b, us-west-2c
  DbSubnetIpBlocks:
    Type: CommaDelimitedList
    Default: 172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26
Resources:
  DbSubnet1:
    Type: AWS::EC2::Subnet

```

```
Properties:
  AvailabilityZone: !Select
    - 0
    - !Ref VpcAzs
  VpcId: !Ref VPC
  CidrBlock: !Select
    - 0
    - !Ref DbSubnetIpBlocks
DbSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    AvailabilityZone: !Sub
      - ${AWS::Region}${AZ}
      - AZ: !Select
        - 1
        - !Ref VpcAzs
    VpcId: !Ref VPC
    CidrBlock: !Select
      - 1
      - !Ref DbSubnetIpBlocks
DbSubnet3:
  Type: AWS::EC2::Subnet
  Properties:
    AvailabilityZone: !Sub
      - ${AWS::Region}${AZ}
      - AZ: !Select
        - 2
        - !Ref VpcAzs
    VpcId: !Ref VPC
    CidrBlock: !Select
      - 2
      - !Ref DbSubnetIpBlocks
```

関連リソース

CloudFormation では、動的参照を使用してプロパティ値を動的に指定することもできます。例えば、Systems Manager パラメータストアに保存されている Secure String を参照する必要がある場合があります。詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

Ref または Sub 関数内で擬似パラメータを使用して、値を動的に入力することもできます。詳細については、「[擬似パラメータを使用して AWS 値を取得する](#)」を参照してください。

CloudFormation テンプレートの Outputs 構文

オプションの Outputs セクションは、スタックの出力値を宣言します。これらの出力値は、さまざまな方法で使用できます。

- リソースに関する重要な詳細をキャプチャする – 出力は、リソースに関する重要な情報をキャプチャする便利な方法です。たとえば、見つけやすいバケットを作成するスタックの S3 バケット名を出力できます。出力値を表示するには、CloudFormation コンソールの [出力] タブまたは [describe-stacks](#) CLI コマンドを使用します。
- クロススタック参照 – 出力値を他のスタックにインポートして、[スタック間で参照を作成](#)できます。これは、複数のスタック間でリソースまたは設定を共有する必要がある場合に役立ちます。

Important

CloudFormation は、Outputs セクションに含める情報の編集または難読化を行いません。このセクションを使用して、パスワードやシークレットなどの機密情報を出力しないことを強くお勧めします。

出力値は、スタックオペレーションの完了後に使用できます。スタックのステータスが「IN_PROGRESS」[ステータス](#)のいずれかのとき、スタックの出力値は利用できません。出力値が常に利用できない場合があるため、サービスランタイムとスタック出力値の間の依存関係を確立することはお勧めしません。

構文

Outputs セクションは、キー名 Outputs で構成されます。1 つのテンプレートで宣言できる出力は最大 200 個です。

次の例は、Outputs セクションの構造を示します。

JSON

出力の宣言全体を中括弧で囲みます。複数の出力をカンマで区切ります。

```
"Outputs" : {  
  "OutputLogicalID" : {  
    "Description" : "Information about the value",  
    "Value" : "Value to return",
```

```
"Export" : {  
  "Name" : "Name of resource to export"  
}  
}  
}
```

YAML

```
Outputs:  
  OutputLogicalID:  
    Description: Information about the value  
    Value: Value to return  
    Export:  
      Name: Name of resource to export
```

出力フィールド

Outputs セクションでは、次のフィールドを含めることができます。

論理 ID (論理名とも呼ばれます)

現在の出力の識別子。論理 ID は英数字 (a-z、A-Z、0-9) とし、テンプレート内で一意である必要があります。

Description (オプション)

出力値について説明する String 型。説明の宣言の値は、長さ 0～1024 バイトのリテラル文字列である必要があります。説明の指定には、パラメータまたは機能を使用できません。

Value (必須)

[describe-stacks](#) コマンドから返されるプロパティの値。出力の値には、リテラル、パラメータ参照、疑似パラメータ、マッピング値、および組み込み関数を含めることができます。

Export (オプション)

クロススタック参照にエクスポートされるリソース出力の名前。

組み込み関数を使用して、エクスポートの Name の値をカスタマイズできます。

詳細については、「[デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)」を参照してください。

出力と条件を関連付けるには、テンプレートの [Conditions](#) セクションで条件を定義します。

例

以下の例では、スタック出力のしくみを示しています。

トピック

- [スタック出力](#)
- [Fn::Sub を使用してエクスポート名をカスタマイズする](#)
- [Fn::Join を使用してエクスポート名をカスタマイズする](#)
- [Fn::Join を使用して構築された URL を返す](#)

スタック出力

次の例では、BackupLoadBalancerDNSName という名前の出力によって、論理 ID が BackupLoadBalancer であるリソースの DNS 名が CreateProdResources 条件が true の場合に返されます。InstanceID という名前の出力は、論理 ID EC2Instance を持つ EC2 インスタンスの ID を返します。

JSON

```
"Outputs" : {
  "BackupLoadBalancerDNSName" : {
    "Description": "The DNSName of the backup load balancer",
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer", "DNSName" ] },
    "Condition" : "CreateProdResources"
  },
  "InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}
```

YAML

```
Outputs:
  BackupLoadBalancerDNSName:
    Description: The DNSName of the backup load balancer
    Value: !GetAtt BackupLoadBalancer.DNSName
```



```
Condition: CreateProdResources
InstanceID:
  Description: The Instance ID
  Value: !Ref EC2Instance
```

Fn::Sub を使用してエクスポート名をカスタマイズする

次の例では、StackVPC という名前の出力が VPC の ID を返し、スタック名の先頭に VPCID が追加された名前でクロススタックリファレンスの値をエクスポートします。

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : {"Fn::Sub": "${AWS::StackName}-VPCID" }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Sub "${AWS::StackName}-VPCID"
```

Fn::Sub 関数の詳細については、「[Fn::Sub](#)」を参照してください。

Fn::Join を使用してエクスポート名をカスタマイズする

また、**Fn::Join** 関数を使用して、パラメータ、リソース属性、および他の文字列に基づいて値を生成することもできます。

次の例では、**Fn::Sub** 関数の代わりに **Fn::Join** 関数を使用してエクスポート名をカスタマイズします。この **Fn::Join** 関数の例は、コロンを区切り文字として使用して、スタック名と名前 VPCID を連結します。

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : { "Fn::Join" : [ ":", [ { "Ref" : "AWS::StackName" }, "VPCID" ] ] }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Join [ ":", [ !Ref "AWS::StackName", VPCID ] ]
```

`Fn::Join` 関数の詳細については、「[Fn::Join](#)」を参照してください。

Fn::Join を使用して構築された URL を返す

WordPress サイトを作成するテンプレートの次の例では、`InstallURL` は、`http://`、リソース `ElasticLoadBalancer` の DNS 名、`/wp-admin/install.php` を連結する `Fn::Join` 関数呼び出しによって返される文字列です。出力値は次の例のようになります。

```
http://mywptests-elasticl-1gb51l6sl8y5v-206169572.aws-region.elb.amazonaws.com/wp-admin/install.php
```

JSON

```
{
  "Outputs": {
    "InstallURL": {
      "Value": {
        "Fn::Join": [
          "",
          [
```

```
        "http://",
        {
            "Fn::GetAtt": [
                "ElasticLoadBalancer",
                "DNSName"
            ]
        },
        "/wp-admin/install.php"
    ]
}
},
"Description": "Installation URL of the WordPress website"
}
}
```

YAML

```
Outputs:
  InstallURL:
    Value: !Join
      - ''
      - - 'http://'
        - !GetAtt
          - ElasticLoadBalancer
          - DNSName
        - /wp-admin/install.php
    Description: Installation URL of the WordPress website
```

`Fn::Join` 関数の詳細については、「[Fn::Join](#)」を参照してください。

CloudFormation テンプレートの Mappings 構文

オプションの Mappings セクションは、特定の条件または依存関係に基づいて値を指定するために使用できる key-value ペアを作成するのに役立ちます。

Mappings セクションの一般的なユースケースの 1 つは、スタックがデプロイされている AWS リージョンに基づいて値を設定することです。これは `AWS::Region` 擬似パラメータを使用することで実現できます。`AWS::Region` 擬似パラメータは、CloudFormation によってスタックが作成されるリージョンに解決される値です。擬似パラメータは、スタックの作成時に CloudFormation によって解決されます。

マップ内の値を取得するには、テンプレートの Resources セクション内で `Fn::FindInMap` 組み込み関数を使用できます。

構文

Mappings セクションでは、次の構文を使用します。

JSON

```
"Mappings" : {  
  "MappingLogicalName" : {  
    "Key1" : {  
      "Name" : "Value1"  
    },  
    "Key2" : {  
      "Name" : "Value2"  
    },  
    "Key3" : {  
      "Name" : "Value3"  
    }  
  }  
}
```

YAML

```
Mappings:  
  MappingLogicalName:  
    Key1:  
      Name: Value1  
    Key2:  
      Name: Value2  
    Key3:  
      Name: Value3
```

- MappingLogicalName はマッピングの論理名です。
- マッピング内では、各マッピングは、別のマッピングが後に続くキーです。
- キーは、名前と値のペアのマップであり、マッピング内で一意である必要があります。
- 名前と値のペアは、ラベルとマッピングする値から構成されます。値に名前を付けることで、1 つのキーに対して値のセットを複数、対応付けることができます。
- マッピングに含まれるキーは、リテラル文字列であることが必要です。

- 値はタイプ String または List です。

Note

Mappings セクションにパラメータ、擬似パラメータまたは組み込み関数を含めることはできません。

マッピングの値が現在スタックで使用されていない場合は、マップのみを更新することはできません。リソースを追加、変更、または削除する変更を含める必要があります。

例

トピック

- [基本的なマッピング](#)
- [複数の値のマッピング](#)
- [マッピングから値を返します](#)
- [入力パラメーターと Fn::FindInMap](#)

基本的なマッピング

以下に示したのは、Mappings というマップを持つ RegionToInstanceType セクションの例です。キーは 5 つ存在し、それぞれ単一の文字列値を含んだ名前と値のペアに対応しています。キーはリージョンの名前です。名前と値の各ペアは、キーで表されるリージョンで利用可能な T フォミリーのインスタンスタイプです。名前と値のペアには、名前 (この例では InstanceType) と値があります。

JSON

```
"Mappings" : {
  "RegionToInstanceType" : {
    "us-east-1"      : { "InstanceType" : "t2.micro" },
    "us-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-north-1"     : { "InstanceType" : "t3.micro" },
    "me-south-1"     : { "InstanceType" : "t3.micro" }
  }
}
```

YAML

```
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
```

複数の値のマッピング

次の例には、2 つの値のセットにマッピングされたリージョンキーがあります。1 つは MyAMI1 という名前で、もう 1 つは MyAMI2 という名前です。

Note

これらの例に示す AMI ID は、デモンストレーション用のプレースホルダーです。可能な場合は、Mappings セクションの代わりに AWS Systems Manager パラメータへの動的参照を使用することを検討してください。使用する AMI が変更されるたびに、すべてのテンプレートが新しい ID で更新されることを回避するには、AWS Systems Manager パラメータを使用してスタックの作成時または更新時に最新の AMI ID を取得します。一般的に使用される AMI の最新バージョンは、Systems Manager のパブリックパラメータとしても利用できます。詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

JSON

```
"Mappings" : {
  "RegionToAMI" : {
    "us-east-1"      : { "MyAMI1" : "ami-12345678901234567", "MyAMI2" :
"ami-23456789012345678" },
    "us-west-1"     : { "MyAMI1" : "ami-34567890123456789", "MyAMI2" :
"ami-45678901234567890" },
```

```

    "eu-west-1"      : { "MyAMI1" : "ami-56789012345678901", "MyAMI2" :
"ami-67890123456789012" },
    "ap-southeast-1" : { "MyAMI1" : "ami-78901234567890123", "MyAMI2" :
"ami-89012345678901234" },
    "ap-northeast-1" : { "MyAMI1" : "ami-90123456789012345", "MyAMI2" :
"ami-01234567890123456" }
  }
}

```

YAML

```

Mappings:
  RegionToAMI:
    us-east-1:
      MyAMI1: ami-12345678901234567
      MyAMI2: ami-23456789012345678
    us-west-1:
      MyAMI1: ami-34567890123456789
      MyAMI2: ami-45678901234567890
    eu-west-1:
      MyAMI1: ami-56789012345678901
      MyAMI2: ami-67890123456789012
    ap-southeast-1:
      MyAMI1: ami-78901234567890123
      MyAMI2: ami-89012345678901234
    ap-northeast-1:
      MyAMI1: ami-90123456789012345
      MyAMI2: ami-01234567890123456

```

マッピングから値を返します

`Fn::FindInMap` 関数を使用すると、指定したキーによって名前付きの値を取得できます。次に示したのは、Amazon EC2 というリソースを含んだテンプレートの例です。このリソースの `InstanceType` プロパティを、`FindInMap` 関数によって割り当てています。`FindInMap` 関数で、(`AWS::Region` 擬似パラメータを使用して) スタックを作成する AWS リージョンとしてキーを指定し、マッピング先の値の名前として `InstanceType` を指定しています。`ImageId` は `Systems Manager` パラメータを使用して、最新の Amazon Linux 2 AMI を動的に取得します。擬似 Parameters の詳細については、「[擬似パラメータを使用して AWS 値を取得する](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Mappings" : {
    "RegionToInstanceType" : {
      "us-east-1"      : { "InstanceType" : "t2.micro" },
      "us-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-north-1"     : { "InstanceType" : "t3.micro" },
      "me-south-1"     : { "InstanceType" : "t3.micro" }
    }
  },
  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
        "InstanceType" : { "Fn::FindInMap" : [ "RegionToInstanceType", { "Ref" :
"AWS::Region" }, "InstanceType" ] }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
Resources:
  myEC2Instance:
```



```

Type: AWS::EC2::Instance
Properties:
  ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
  InstanceType: !FindInMap [RegionToInstanceType, !Ref 'AWS::Region', InstanceType]

```

入力パラメーターと **Fn::FindInMap**

次のサンプルテンプレートは、複数のマッピングを使用して EC2 インスタンスを作成する方法を示しています。テンプレートはネストされたマッピングを使用して、ターゲットの AWS リージョンと環境タイプ (Dev または Prod) に基づいて適切なインスタンスタイプとセキュリティグループを自動的に選択します。また、Systems Manager パラメータを使用して、最新の Amazon Linux 2 AMI を動的に取得します。

JSON

```

{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Parameters" : {
    "EnvironmentType" : {
      "Description" : "The environment type (Dev or Prod)",
      "Type" : "String",
      "Default" : "Dev",
      "AllowedValues" : [ "Dev", "Prod" ]
    }
  },
  "Mappings" : {
    "RegionAndEnvironmentToInstanceType" : {
      "us-east-1"      : { "Dev" : "t3.micro", "Prod" : "c5.large" },
      "us-west-1"      : { "Dev" : "t2.micro", "Prod" : "m5.large" }
    },
    "RegionAndEnvironmentToSecurityGroup" : {
      "us-east-1"      : { "Dev" : "sg-12345678", "Prod" : "sg-abcdef01" },
      "us-west-1"      : { "Dev" : "sg-ghijkl23", "Prod" : "sg-45678abc" }
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",

```

```

      "InstanceType" : { "Fn::FindInMap": [ "RegionAndEnvironmentToInstanceType",
{ "Ref": "AWS::Region" }, { "Ref": "EnvironmentType" } ]},
      "SecurityGroupIds" : [{ "Fn::FindInMap" :
[ "RegionAndEnvironmentToSecurityGroup", { "Ref" : "AWS::Region" }, { "Ref" :
"EnvironmentType" } ]}]
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvironmentType:
    Description: The environment type (Dev or Prod)
    Type: String
    Default: Dev
    AllowedValues:
      - Dev
      - Prod
Mappings:
  RegionAndEnvironmentToInstanceType:
    us-east-1:
      Dev: t3.micro
      Prod: c5.large
    us-west-1:
      Dev: t2.micro
      Prod: m5.large
  RegionAndEnvironmentToSecurityGroup:
    us-east-1:
      Dev: sg-12345678
      Prod: sg-abcdef01
    us-west-1:
      Dev: sg-ghijkl23
      Prod: sg-45678abc
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'

```

```
InstanceType: !FindInMap [RegionAndEnvironmentToInstanceType, !Ref
'AWS::Region', !Ref EnvironmentType]
SecurityGroupIds:
  - !FindInMap [RegionAndEnvironmentToSecurityGroup, !Ref 'AWS::Region', !Ref
EnvironmentType]
```

関連リソース

これらの関連トピックは、Fn::FindInMap 関数を使用するテンプレートを作成する際に役立ちます。

- [Fn::FindInMap](#)
- [Fn::FindInMap の機能強化](#)
- [Fn::Sub](#)

CloudFormation テンプレートの Metadata 構文

Metadata は、JSON オブジェクトまたは YAML オブジェクトを使用して追加情報を保存します。テンプレートで利用できるテンプレートレベルのメタデータのタイプは次のとおりです。

カスタムメタデータ

ユーザー定義のキーと値のペアを保存します。たとえば、リソースの作成には影響しないが、インフラストラクチャ、チーム、またはデプロイの詳細に関する追加のコンテキストを提供する追加情報を提供できます。

AWS::CloudFormation::Interface

CloudFormation コンソールに表示される場合の入力パラメータのグループ化と順序を定義します。デフォルトでは、CloudFormation コンソールではパラメータが論理 ID によりアルファベット順にソートされます。

AWS::CloudFormation::Designer

CloudFormation Designer (Designer) は 2025 年 2 月 5 日にサポートが終了しました。

Important

スタックの更新時に、Metadata セクションを単独で更新することはできません。更新できるのは、リソースを追加、変更、または削除する変更を含める場合だけです。

CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。このため、パスワードやシークレットなどの機密情報を保存しないことを強くお勧めします。

構文

CloudFormation テンプレートでカスタムメタデータを宣言するには、次の構文を使用します。

JSON

```
"Metadata" : {  
  "Instances" : {"Description" : "Information about the instances"},  
  "Databases" : {"Description" : "Information about the databases"}  
}
```

YAML

```
Metadata:  
  Instances:  
    Description: "Information about the instances"  
  Databases:  
    Description: "Information about the databases"
```

AWS::CloudFormation::Interface の構文については、

「[AWS::CloudFormation::Interface メタデータを使用した CloudFormation パラメータの整理](#)」を参照してください。

AWS::CloudFormation::Interface メタデータを使用した CloudFormation パラメータの整理

AWS::CloudFormation::Interface は、CloudFormation コンソールでパラメータをグループ化およびソートする方法を定義するメタデータキーです。デフォルトでは、コンソールでスタックを作成または更新すると、コンソールに入力パラメータが論理 ID のアルファベット順にリストされます。ユーザーがパラメーター値を効率的に指定できるように、このキーを使用して、パラメーターのグループ化とソートを独自に定義することができます。たとえば、すべての EC2 関連パラメーターを 1 つのグループに含め、すべての VPC 関連パラメーターを別のグループに含めることができます。

メタデータキーで、作成するグループ、各グループに含めるパラメーター、コンソールで各パラメーターがそのグループ内に表示される順序を指定できます。

パラメータのラベルを定義することもできます。ラベルは、パラメーターの論理 ID の代わりにコンソールに表示されるフレンドリ名または説明です。ラベルは、各パラメーターに指定する値をユーザーが理解するのに役立ちます。たとえば、KeyPair パラメーターには Select an EC2 key pair というラベルを付けることができます。

メタデータキーで参照するすべてのパラメーターは、テンプレートの Parameters セクションで宣言する必要があります。

Note

CloudFormation コンソールのみが `AWS::CloudFormation::Interface` メタデータキーを使用します。AWS CLI および API 呼び出しはこのキーを使用しません。

構文

CloudFormation テンプレートでこのエンティティを宣言するには、次の構文を使用します。

JSON

```
"Metadata" : {
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups": [
      {
        "Label": {
          "default": "Group Label"
        },
        "Parameters": [
          "Parameter1",
          "Parameter2"
        ]
      }
    ],
    "ParameterLabels": {
      "Parameter1": {
        "default": "Friendly Name for Parameter1"
      }
    }
  }
}
```

```
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: Group Label
        Parameters:
          - Parameter1
          - Parameter2
    ParameterLabels:
      Parameter1:
        default: Friendly Name for Parameter1
```

プロパティ

ParameterGroups

パラメーターグループタイプ、グループ名を指定する場所、各グループ内のパラメーター、パラメーターが表示される順序のリスト。

必須: いいえ

Label

パラメーターグループの名前。

必須: いいえ

default

CloudFormation コンソールがパラメータグループの命名に使用するデフォルトのラベル。

必須: いいえ

タイプ: 文字列

Parameters

グループに含めるパラメーター論理 ID (大文字小文字が区別されます) のリスト。パラメーターは、テンプレートの Parameters セクションで既に定義されている必要があります。パラメーターは、1 つのパラメーターグループにのみ含めることができます。

コンソールには、Other parameters グループ内の、パラメーターグループに関連付けないパラメーターがアルファベット順でリストされます。

必須: いいえ

タイプ: 文字列値のリスト

ParameterLabels

スタックが作成または更新されたときに CloudFormation コンソールに表示されるパラメータとそのフレンドリ名のマッピング。

必須: いいえ

パラメータラベル

パラメーターのラベル。ラベルは、スタックが作成または更新されるときに CloudFormation コンソールが [Specify Parameters] (パラメータを指定) に表示するフレンドリ名または説明を定義します。パラメータラベルは、テンプレートの Parameters セクションで宣言された有効なパラメーターの論理 ID (大文字と小文字を区別) にする必要があります。

必須: いいえ

default

CloudFormation コンソールがパラメータの命名に使用するデフォルトのラベル。

必須: いいえ

タイプ: 文字列

例

次の例では、2 つのパラメーターグループ Network Configuration および Amazon EC2 Configuration を定義します。Network Configuration グループには、VPCID、SubnetId、SecurityGroupId の各パラメーターが含まれます。これらは、テンプレートの Parameters セクション (非表示) で定義されます。コンソールにこれらのパラメーターが表示される順序は、パラメーターがリストされる順序によって定義されます (先頭は VPCID パラメーター)。この例では、Amazon EC2 Configuration パラメーターも同様にグループ化およびソートされています。

この例では、VPCID パラメーターのラベルも定義しています。コンソールには、パラメータの論理 ID (VPCID) の代わりに [Which VPC should this be deployed to? (どの VPC をこれにデプロイする必要がありますか。)] が表示されます。

JSON

```
"Metadata" : {
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups" : [
      {
        "Label" : { "default" : "Network Configuration" },
        "Parameters" : [ "VPCID", "SubnetId", "SecurityGroupID" ]
      },
      {
        "Label" : { "default": "Amazon EC2 Configuration" },
        "Parameters" : [ "InstanceType", "KeyName" ]
      }
    ],
    "ParameterLabels" : {
      "VPCID" : { "default" : "Which VPC should this be deployed to?" }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Network Configuration"
        Parameters:
          - VPCID
          - SubnetId
          - SecurityGroupID
      - Label:
          default: "Amazon EC2 Configuration"
        Parameters:
          - InstanceType
          - KeyName
    ParameterLabels:
      VPCID:
        default: "Which VPC should this be deployed to?"
```


コンソールに表示されたパラメータグループ

次の図は、この例のメタデータキーを使用すると、スタックが作成または更新されたときにパラメータグループがコンソールにどのように表示されるかを示しています。コンソールのパラメータグループ

Parameters

Network Configuration

Which VPC should this be deployed to? Search by ID, or Name tag value
VpcId of your existing Virtual Private Cloud (VPC)

SubnetId Search by ID, or Name tag value
The list of SubnetIds in your Virtual Private Cloud (VPC)

SecurityGroupId Search by ID, name or Name tag value
VpcId of your existing Virtual Private Cloud (VPC)

Amazon EC2 Configuration

InstanceType m1.small Web

KeyName Search
Name of an existing EC2 KeyPair to enable SSH access to the instance

CloudFormation テンプレートの Rules 構文

Rules セクションは、カスタム検証ロジックを有効にする CloudFormation テンプレートのオプション部分です。このセクションが含まれている場合、CloudFormation がリソースを作成または更新する前にパラメータ値を検証するルール関数が含まれます。

ルールは、標準のパラメータ制約が不十分な場合に便利です。例えば、SSL が有効になっている場合は、証明書とドメイン名の両方を指定する必要があります。ルールを使用すると、これらの依存関係を確実に満たすことができます。

構文

Rules セクションでは、次の構文を使用します。

JSON

テンプレートの Rules セクションは、キーの名前 Rules とそれに続く単一のコロンで構成されます。すべてのルール宣言を囲むには、中括弧を使用する必要があります。複数のルールを宣言する場

合は、カンマで区切ります。ルールごとに、引用符で囲んだ論理名、単一のコロンの、およびルール条件とアサーションを囲む中括弧から成る形式で宣言します。

```
{
  "Rules": {
    "LogicalRuleName1": {
      "RuleCondition": {
        "rule-specific intrinsic function": "Value"
      },
      "Assertions": [
        {
          "Assert": {
            "rule-specific intrinsic function": "Value"
          },
          "AssertDescription": "Information about this assert"
        },
        {
          "Assert": {
            "rule-specific intrinsic function": "Value"
          },
          "AssertDescription": "Information about this assert"
        }
      ]
    },
    "LogicalRuleName2": {
      "Assertions": [
        {
          "Assert": {
            "rule-specific intrinsic function": "Value"
          },
          "AssertDescription": "Information about this assert"
        }
      ]
    }
  }
}
```

YAML

```
Rules:
  LogicalRuleName1:
    RuleCondition:
      rule-specific intrinsic function: Value
```

```
Assertions:
- Assert:
    rule-specific intrinsic function: Value
    AssertDescription: Information about this assert
- Assert:
    rule-specific intrinsic function: Value
    AssertDescription: Information about this assert
LogicalRuleName2:
Assertions:
- Assert:
    rule-specific intrinsic function: Value
    AssertDescription: Information about this assert
```

ルールフィールド

Rules セクションでは、次のフィールドを含めることができます。

論理 ID (論理名とも呼ばれます)

各ルール用の一意の識別子。

RuleCondition (オプション)

ルールがいつ有効になるかを決定するプロパティ。ルール条件を定義しない場合、ルールのアサーションが常に有効になります。それぞれのルールごとに、定義できるルール条件は 1 つだけです。

Assertions (必須)

特定のパラメータの許容値を指定する 1 つ以上のステートメント。

Assert

true に評価される必要がある条件。

AssertDescription

アサーションが失敗したときに表示されるメッセージ。

ルール固有の組み込み関数

ルールを定義するには、「ルール固有の関数」を使用します。これは、テンプレートの Rules セクションでのみ使用できる関数です。関数をネストすることができますが、ルール条件またはアサーションの最終結果は、true または false である必要があります。

以下のルール関数を使用できます。

- [Fn::And](#)
- [Fn::Contains](#)
- [Fn::EachMemberEquals](#)
- [Fn::EachMemberIn](#)
- [Fn::Equals](#)
- [Fn::If](#)
- [Fn::Not](#)
- [Fn::Or](#)
- [Fn::RefAll](#)
- [Fn::ValueOf](#)
- [Fn::ValueOfAll](#)

これらの関数は、ルールの条件またはアサーションで使用されます。条件プロパティによって、CloudFormation がアサーションを適用するかどうかが決まります。条件が `true` に評価された場合、プロビジョニング済み製品の作成または更新時に、CloudFormation はアサーションを評価してパラメータ値が有効かどうか確認します。パラメータ値が無効である場合、CloudFormation はスタックの作成や更新を行いません。条件が `false` に評価される場合、CloudFormation はパラメータ値を確認せず、スタックオペレーションに進みます。

例

トピック

- [パラメーター値の条件付きの確認](#)
- [パラメータ間の検証](#)

パラメーター値の条件付きの確認

次の例では、2 つのルールが `InstanceType` パラメータの値をチェックします。環境パラメータの値 (`test` または `prod`) に応じて、ユーザーは `InstanceType` パラメータに対して `t3.medium` または `t3.large` を指定する必要があります。 `InstanceType` および `Environment` パラメータは、同じテンプレートの `Parameters` セクションで宣言する必要があります。

JSON

```
{
  "Rules": {
    "testInstanceType": {
      "RuleCondition": {
        "Fn::Equals": [
          {"Ref": "Environment"},
          "test"
        ]
      },
      "Assertions": [
        {
          "Assert": {
            "Fn::Contains": [
              ["t3.medium"],
              {"Ref": "InstanceType"}
            ]
          },
          "AssertDescription": "For a test environment, the instance type must be t3.medium"
        }
      ],
      "prodInstanceType": {
        "RuleCondition": {
          "Fn::Equals": [
            {"Ref": "Environment"},
            "prod"
          ]
        },
        "Assertions": [
          {
            "Assert": {
              "Fn::Contains": [
                ["t3.large"],
                {"Ref": "InstanceType"}
              ]
            },
            "AssertDescription": "For a production environment, the instance type must be t3.large"
          }
        ]
      }
    }
  }
}
```

```
}  
}
```

YAML

```
Rules:  
  testInstanceType:  
    RuleCondition: !Equals  
      - !Ref Environment  
      - test  
    Assertions:  
      - Assert:  
          'Fn::Contains':  
            - - t3.medium  
              - !Ref InstanceType  
          AssertDescription: 'For a test environment, the instance type must be  
t3.medium'  
  prodInstanceType:  
    RuleCondition: !Equals  
      - !Ref Environment  
      - prod  
    Assertions:  
      - Assert:  
          'Fn::Contains':  
            - - t3.large  
              - !Ref InstanceType  
          AssertDescription: 'For a production environment, the instance type must be  
t3.large'
```

パラメータ間の検証

次のサンプルテンプレートは、パラメータ間の相互検証にルールを使用する方法を示しています。これらは、ロードバランサーの背後で Auto Scaling グループ上に稼働するサンプルウェブサイトを作成します。このウェブサイトは、入力パラメータに応じてポート 80 または 443 で利用できます。Auto Scaling グループ内のインスタンスは、任意のポートでリッスンするように設定できます (デフォルトは 8888)。

このテンプレート内のルールは、スタックの作成前に入力パラメータを検証します。これらのルールは、すべてのサブネットが指定された VPC に属していることを確認し、UseSSL パラメータが Yes に設定されている場合に、SSL 証明書の ARN とホストゾーン名の両方が指定されていることを検証します。

 Note

このテンプレートからスタックを作成した場合、AWS リソースに対する料金が発生します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "VpcId": {
      "Type": "AWS::EC2::VPC::Id",
      "Description": "VpcId of your existing Virtual Private Cloud (VPC)",
      "ConstraintDescription": "must be the VPC Id of an existing Virtual Private Cloud."
    },
    "Subnets": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "The list of SubnetIds in your Virtual Private Cloud (VPC)",
      "ConstraintDescription": "must be a list of at least two existing subnets associated with at least two different availability zones."
    },
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.micro",
      "AllowedValues": ["t2.micro", "t3.micro"],
      "ConstraintDescription": "must be a valid EC2 instance type."
    },
    "KeyName": {
      "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
    },
    "SSHLocation": {
      "Description": "The IP address range that can be used to SSH to the EC2 instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",

```

```

    "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})",
    "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
  },
  "UseSSL": {
    "AllowedValues": ["Yes", "No"],
    "Default": "No",
    "Description": "Select \"Yes\" to implement SSL, \"No\" to skip (default).",
    "Type": "String"
  },
  "ALBSSLCertificateARN": {
    "Default": "",
    "Description": "[Optional] The ARN of the SSL certificate to be used for the Application Load Balancer",
    "Type": "String"
  },
  "HostedZoneName": {
    "AllowedPattern": "^$|(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-\\-]*[a-zA-Z0-9])\\.)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-\\-]*[A-Za-z0-9])$",
    "Default": "",
    "Description": "[Optional] The domain name of a valid Hosted Zone on AWS.",
    "Type": "String"
  }
},
"Conditions": {
  "UseALBSSL": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]}
},
"Rules": {
  "SubnetsInVPC": {
    "Assertions": [
      {
        "Assert": {"Fn::EachMemberEquals": [{"Fn::ValueOf": ["Subnets", "VpcId"]}, {"Ref": "VpcId"}]},
        "AssertDescription": "All subnets must be in the VPC"
      }
    ]
  },
  "ValidateHostedZone": {
    "RuleCondition": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]},
    "Assertions": [
      {
        "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "ALBSSLCertificateARN"}, ""]}]},
        "AssertDescription": "ACM Certificate value cannot be empty if SSL is required"
      }
    ]
  }
}

```



```

    },
    {
      "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "HostedZoneName"}, ""]}]},
      "AssertDescription": "Route53 Hosted Zone Name is mandatory when SSL is
required"
    }
  ]
}
},
"Resources": {
  "WebServerGroup": {
    "Type": "AWS::AutoScaling::AutoScalingGroup",
    "Properties": {
      "VPCZoneIdentifier": {"Ref": "Subnets"},
      "LaunchTemplate": {
        "LaunchTemplateId": {"Ref": "LaunchTemplate"},
        "Version": {"Fn::GetAtt": ["LaunchTemplate", "LatestVersionNumber"]}
      },
      "MinSize": "2",
      "MaxSize": "2",
      "TargetGroupARNs": [{"Ref": "ALBTargetGroup"}]
    },
    "CreationPolicy": {
      "ResourceSignal": {"Timeout": "PT15M"}
    },
    "UpdatePolicy": {
      "AutoScalingRollingUpdate": {
        "MinInstancesInService": "1",
        "MaxBatchSize": "1",
        "PauseTime": "PT15M",
        "WaitOnResourceSignals": true
      }
    }
  },
  "LaunchTemplate": {
    "Type": "AWS::EC2::LaunchTemplate",
    "Metadata": {
      "Comment": "Install a simple application",
      "AWS::CloudFormation::Init": {
        "config": {
          "packages": {"yum": {"httpd": []}},
          "files": {
            "/var/www/html/index.html": {

```

```

        "content": {"Fn::Join": ["\n", ["<h1>Congratulations, you have
successfully launched the AWS CloudFormation sample.</h1>"]]},
        "mode": "000644",
        "owner": "root",
        "group": "root"
    },
    "/etc/cfn/cfn-hup.conf": {
        "content": {"Fn::Join": ["", [
            "[main]\n",
            "stack=", {"Ref": "AWS::StackId"}, "\n",
            "region=", {"Ref": "AWS::Region"}, "\n"
        ]]},
        "mode": "000400",
        "owner": "root",
        "group": "root"
    },
    "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
        "content": {"Fn::Join": ["", [
            "[cfn-auto-reloader-hook]\n",
            "triggers=post.update\n",
            "path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init\n",
            "action=/opt/aws/bin/cfn-init -v ",
            "    --stack ", {"Ref": "AWS::StackName"},
            "    --resource LaunchTemplate ",
            "    --region ", {"Ref": "AWS::Region"}, "\n",
            "runas=root\n"
        ]]},
        "mode": "000400",
        "owner": "root",
        "group": "root"
    }
},
"services": {
    "sysvinit": {
        "httpd": {
            "enabled": "true",
            "ensureRunning": "true"
        },
        "cfn-hup": {
            "enabled": "true",
            "ensureRunning": "true",
            "files": [
                "/etc/cfn/cfn-hup.conf",
                "/etc/cfn/hooks.d/cfn-auto-reloader.conf"
            ]
        }
    }
}

```

```

    ]
  }
}
}
}
},
"Properties": {
  "LaunchTemplateName": {"Fn::Sub": "${AWS::StackName}-launch-template"},
  "LaunchTemplateData": {
    "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
    "SecurityGroupIds": [{"Ref": "InstanceSecurityGroup"}],
    "InstanceType": {"Ref": "InstanceType"},
    "KeyName": {"Ref": "KeyName"},
    "UserData": {
      "Fn::Base64": {"Fn::Join": ["", [
        "#!/bin/bash\n",
        "yum install -y aws-cfn-bootstrap\n",
        "/opt/aws/bin/cfn-init -v ",
        "    --stack ", {"Ref": "AWS::StackName"},
        "    --resource LaunchTemplate ",
        "    --region ", {"Ref": "AWS::Region"}, "\n",
        "/opt/aws/bin/cfn-signal -e $? ",
        "    --stack ", {"Ref": "AWS::StackName"},
        "    --resource WebServerGroup ",
        "    --region ", {"Ref": "AWS::Region"}, "\n"
      ]]}
    ]
  }
},
"ELBSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "Allow access to the ELB",
    "VpcId": {"Ref": "VpcId"},
    "SecurityGroupIngress": [{
      "Fn::If": [
        "UseALBSSL",
        {
          "IpProtocol": "tcp",
          "FromPort": 443,
          "ToPort": 443,

```

```

        "CidrIp": "0.0.0.0/0"
      },
      {
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "CidrIp": "0.0.0.0/0"
      }
    ]
  ]]
}
},
"ApplicationLoadBalancer": {
  "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
  "Properties": {
    "Subnets": {"Ref": "Subnets"},
    "SecurityGroups": [{"Ref": "ELBSecurityGroup"}]
  }
},
"ALBListener": {
  "Type": "AWS::ElasticLoadBalancingV2::Listener",
  "Properties": {
    "DefaultActions": [{
      "Type": "forward",
      "TargetGroupArn": {"Ref": "ALBTargetGroup"}
    }],
    "LoadBalancerArn": {"Ref": "ApplicationLoadBalancer"},
    "Port": {"Fn::If": ["UseALBSSL", 443, 80]},
    "Protocol": {"Fn::If": ["UseALBSSL", "HTTPS", "HTTP"]},
    "Certificates": [{
      "Fn::If": [
        "UseALBSSL",
        {"CertificateArn": {"Ref": "ALBSSLCertificateARN"}},
        {"Ref": "AWS::NoValue"}
      ]
    }
  ]
}],
"ALBTargetGroup": {
  "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
  "Properties": {
    "HealthCheckIntervalSeconds": 30,
    "HealthCheckTimeoutSeconds": 5,
    "HealthyThresholdCount": 3,

```



```

        "Port": 80,
        "Protocol": "HTTP",
        "UnhealthyThresholdCount": 5,
        "VpcId": {"Ref": "VpcId"}
    }
},
"InstanceSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Enable SSH access and HTTP access on the inbound port",
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": 80,
                "ToPort": 80,
                "SourceSecurityGroupId": {"Fn::Select": [0, {"Fn::GetAtt":
["ApplicationLoadBalancer", "SecurityGroups"]}]}
            },
            {
                "IpProtocol": "tcp",
                "FromPort": 22,
                "ToPort": 22,
                "CidrIp": {"Ref": "SSHLocation"}
            }
        ],
        "VpcId": {"Ref": "VpcId"}
    }
},
"RecordSet": {
    "Type": "AWS::Route53::RecordSetGroup",
    "Condition": "UseALBSSL",
    "Properties": {
        "HostedZoneName": {"Fn::Join": [ "", [{"Ref": "HostedZoneName"}, "." ] ] },
        "RecordSets": [ {
            "Name": {"Fn::Join": [ "", [
                {"Fn::Select": [ "0", {"Fn::Split": [ ".", {"Fn::GetAtt":
["ApplicationLoadBalancer", "DNSName"]} ] ] ] },
                ".",
                {"Ref": "HostedZoneName"},
                "."
            ] ] },
            "Type": "A",
            "AliasTarget": {
                "DNSName": {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"] },

```

```

        "EvaluateTargetHealth": true,
        "HostedZoneId": {"Fn::GetAtt": ["ApplicationLoadBalancer",
"CanonicalHostedZoneID"]}
    }
  ]]
}
},
"Outputs": {
  "URL": {
    "Description": "URL of the website",
    "Value": {"Fn::Join": ["", [
      {"Fn::If": [
        "UseALBSSL",
        {"Fn::Join": ["", [
          "https://",
          {"Fn::Join": ["", [
            {"Fn::Select": ["0", {"Fn::Split": [".", {"Fn::GetAtt":
["ApplicationLoadBalancer", "DNSName"]}]}}],
            ".",
            {"Ref": "HostedZoneName"},
            "."
          ]}]},
        ]}],
        {"Fn::Join": ["", [
          "http://",
          {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"]}
        ]}]
      ]}
    ]}]
  }
}
}
}

```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Parameters:
```

```
  VpcId:
```

```
    Type: AWS::EC2::VPC::Id
```

```
    Description: VpcId of your existing Virtual Private Cloud (VPC)
```

```
    ConstraintDescription: must be the VPC Id of an existing Virtual Private Cloud.
```

```
  Subnets:
```

Type: List<AWS::EC2::Subnet::Id>

Description: The list of SubnetIds in your Virtual Private Cloud (VPC)

ConstraintDescription: >-

must be a list of at least two existing subnets associated with at least two different availability zones. They should be residing in the selected Virtual Private Cloud.

InstanceType:

Description: WebServer EC2 instance type

Type: String

Default: t2.micro

AllowedValues:

- t2.micro
- t3.micro

ConstraintDescription: must be a valid EC2 instance type.

KeyName:

Description: Name of an existing EC2 KeyPair to enable SSH access to the instances

Type: AWS::EC2::KeyPair::KeyName

ConstraintDescription: must be the name of an existing EC2 KeyPair.

SSHLocation:

Description: The IP address range that can be used to SSH to the EC2 instances

Type: String

MinLength: '9'

MaxLength: '18'

Default: 0.0.0.0/0

AllowedPattern: '(\d{1,3})\.\(\d{1,3})\.\(\d{1,3})\.\(\d{1,3})/(\d{1,2})'

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

UseSSL:

AllowedValues:

- 'Yes'
- 'No'

ConstraintDescription: Select Yes to create a HTTPS Listener

Default: 'No'

Description: 'Select "Yes" to implement SSL, "No" to skip (default).'

Type: String

ALBSSLCertificateARN:

Default: ''

Description: >-

[Optional] The ARN of the SSL certificate to be used for the Application Load Balancer

Type: String

HostedZoneName:

AllowedPattern: >-

^\$|((([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-\-]*[a-zA-Z0-9])\.)+)([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\-]*[A-Za-z0-9])\$

```
    Default: ''
    Description: '[Optional] The domain name of a valid Hosted Zone on AWS.'
    Type: String
Conditions:
  UseALBSSL: !Equals
    - !Ref UseSSL
    - 'Yes'
Rules:
  SubnetsInVPC:
    Assertions:
      - Assert:
          'Fn::EachMemberEquals':
            - 'Fn::ValueOf':
                - Subnets
                - VpcId
            - Ref: VpcId
          AssertDescription: All subnets must be in the VPC
  ValidateHostedZone:
    RuleCondition: !Equals
      - !Ref UseSSL
      - 'Yes'
    Assertions:
      - Assert: !Not
          - !Equals
            - !Ref ALBSSLCertificateARN
            - ''
          AssertDescription: ACM Certificate value cannot be empty if SSL is required
      - Assert: !Not
          - !Equals
            - !Ref HostedZoneName
            - ''
          AssertDescription: Route53 Hosted Zone Name is mandatory when SSL is required
Resources:
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier: !Ref Subnets
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MinSize: '2'
      MaxSize: '2'
      TargetGroupARNs:
        - !Ref ALBTargetGroup
```

```
CreationPolicy:
  ResourceSignal:
    Timeout: PT15M
UpdatePolicy:
  AutoScalingRollingUpdate:
    MinInstancesInService: '1'
    MaxBatchSize: '1'
    PauseTime: PT15M
    WaitOnResourceSignals: 'true'
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Metadata:
    Comment: Install a simple application
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
        files:
          /var/www/html/index.html:
            content: !Join
              - |+
              - - >-
                <h1>Congratulations, you have successfully launched the AWS
                CloudFormation sample.</h1>
            mode: '000644'
            owner: root
            group: root
          /etc/cfn/cfn-hup.conf:
            content: !Sub |
              [main]
              stack=${AWS::StackId}
              region=${AWS::Region}
            mode: '000400'
            owner: root
            group: root
          /etc/cfn/hooks.d/cfn-auto-reloader.conf:
            content: !Sub |-
              [cfn-auto-reloader-hook]
              triggers=post.update
              path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init
              action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
LaunchTemplate --region ${AWS::Region}
              runas=root
```

```

        mode: '000400'
        owner: root
        group: root
    services:
        sysvinit:
            httpd:
                enabled: 'true'
                ensureRunning: 'true'
            cfn-hup:
                enabled: 'true'
                ensureRunning: 'true'
            files:
                - /etc/cfn/cfn-hup.conf
                - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    Properties:
        LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
        LaunchTemplateData:
            ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
            SecurityGroupIds:
                - !Ref InstanceSecurityGroup
            InstanceType: !Ref InstanceType
            KeyName: !Ref KeyName
            UserData: !Base64
                Fn::Sub: |
                    #!/bin/bash
                    yum install -y aws-cfn-bootstrap
                    /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
LaunchTemplate --region ${AWS::Region}
                    /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
WebServerGroup --region ${AWS::Region}
            ELBSecurityGroup:
                Type: AWS::EC2::SecurityGroup
            Properties:
                GroupDescription: Allow access to the ELB
                VpcId: !Ref VpcId
                SecurityGroupIngress:
                    - !If
                        - UseALBSSL
                        - IpProtocol: tcp
                          FromPort: 443
                          ToPort: 443
                          CidrIp: 0.0.0.0/0
                    - IpProtocol: tcp

```

```
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Subnets: !Ref Subnets
    SecurityGroups:
      - !Ref ELBSecurityGroup
ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref ALBTargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: !If
      - UseALBSSL
      - 443
      - 80
    Protocol: !If
      - UseALBSSL
      - HTTPS
      - HTTP
    Certificates:
      - !If
        - UseALBSSL
        - CertificateArn: !Ref ALBSSLCertificateARN
        - !Ref 'AWS::NoValue'
ALBTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 30
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 3
    Port: 80
    Protocol: HTTP
    UnhealthyThresholdCount: 5
    VpcId: !Ref VpcId
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable SSH access and HTTP access on the inbound port
    SecurityGroupIngress:
```

```

    - IpProtocol: tcp
      FromPort: 80
      ToPort: 80
      SourceSecurityGroupId: !Select
        - 0
        - !GetAtt
          - ApplicationLoadBalancer
          - SecurityGroups
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref SSHLocation
      VpcId: !Ref VpcId
RecordSet:
  Type: AWS::Route53::RecordSetGroup
  Condition: UseALBSSL
  Properties:
    HostedZoneName: !Join
      - ''
      - - !Ref HostedZoneName
        - .
  RecordSets:
    - Name: !Join
      - ''
      - - !Select
        - '0'
        - !Split
          - .
          - !GetAtt
            - ApplicationLoadBalancer
            - DNSName
        - .
      - !Ref HostedZoneName
      - .
    Type: A
    AliasTarget:
      DNSName: !GetAtt
        - ApplicationLoadBalancer
        - DNSName
      EvaluateTargetHealth: true
      HostedZoneId: !GetAtt
        - ApplicationLoadBalancer
        - CanonicalHostedZoneID
  Outputs:

```



```

URL:
  Description: URL of the website
  Value: !Join
    - ''
    - - !If
      - UseALBSSL
      - !Join
        - ''
        - - 'https://'
          - !Join
            - ''
            - - !Select
              - '0'
              - !Split
                - .
                - !GetAtt
                  - ApplicationLoadBalancer
                  - DNSName
            - .
          - !Ref HostedZoneName
            - .
    - !Join
      - ''
      - - 'http://'
        - !GetAtt
          - ApplicationLoadBalancer
          - DNSName

```

CloudFormation テンプレートの Conditions 構文

オプションの Conditions セクションには、エンティティが作成または設定される状況を定義するステートメントが含まれています。例えば、条件を作成し、それをリソースまたは出力に関連付けることで、条件が true の場合にのみ CloudFormation がリソースまたは出力を作成できるようになります。同様に、条件をプロパティに関連付けて、条件が true の場合にのみ CloudFormation がプロパティを特定の値に設定するようにできます。条件が false の場合、CloudFormation はプロパティをユーザーが指定した代替値に設定します。

テンプレートを再利用して、テスト環境と本番環境などのさまざまなコンテキストでリソースを作成する場合は、条件を使用できます。例えば、テンプレートでは、prod または test のいずれかを入力として受け取る EnvironmentType 入力パラメータを追加できます。prod 環境では、特定の機能を備えた EC2 インスタンスを含めることができますが、test 環境ではコストを節約するために

機能を制限したインスタンスを使用する場合があります。この条件定義により、各環境タイプに応じて作成されるリソースとその構成方法を定義できます。

構文

Conditions セクションは、キー名 Conditions で構成されます。各条件宣言には、論理 ID と 1 つ以上の組み込み関数が含まれます。

JSON

```
"Conditions": {  
  "LogicalConditionName1": {  
    "Intrinsic function": ...  
  },  
  
  "LogicalConditionName2": {  
    "Intrinsic function": ...  
  }  
}
```

YAML

```
Conditions:  
  LogicalConditionName1:  
    Intrinsic function:  
      ...  
  
  LogicalConditionName2:  
    Intrinsic function:  
      ...
```

条件の仕組み

条件を使用するには、次の手順に従います。

1. パラメータ定義の追加 – テンプレートの Parameters セクションで条件が評価する入力を定義します。これらの入力パラメータの値に基づいて、条件は true または false と評価されます。疑似パラメータは自動的に利用可能であり、Parameters セクションで明示的な定義を必要としないことに注意してください。疑似 Parameters の詳細については、「[疑似パラメータを使用して AWS 値を取得する](#)」を参照してください。

2. 条件定義の追加 – Fn::If、Fn::Equals、などの組み込み関数を使用して Conditions セクションで条件を定義します。これらの条件は、CloudFormation が関連するリソースをいつ作成するかを指定します。条件は以下に基づいて定義できます。

- 入力または擬似パラメータ値
- その他の条件
- 値のマッピング

ただし、リソースの論理 ID やその属性を条件で参照することはできません。

3. リソースまたは出力に条件を関連付ける – Condition キーと条件の論理 ID を使用して、リソースまたは出力の条件を参照します。必要に応じて、テンプレートの他の部分 (プロパティ値など) の Fn::If を使用して、条件に基づいて値を設定します。詳細については、「[Condition キーの使用](#)」を参照してください。

CloudFormation は、スタックを作成または更新するときに条件を評価します。CloudFormation は、true に関連付けられたエンティティを作成し、false に関連付けられたエンティティを無視します。また、CloudFormation は、リソースを変更する前に、スタックの更新ごとにこれらの条件を再評価します。true の条件に関連付けられているエンティティは更新され、false の条件に関連付けられているエンティティは削除されます。

Important

スタックの更新時に、条件を単独で更新することはできません。条件を更新できるのは、リソースを追加、変更、または削除する変更を含める場合だけです。

条件の組み込み関数

次の組み込み関数を使用して条件を定義できます。

- [Fn::And](#)
- [Fn::Equals](#)
- [Fn::ForEach](#)
- [Fn::If](#)
- [Fn::Not](#)
- [Fn::Or](#)

Note

`Fn::If` は、テンプレートの `Resources` セクションと `Outputs` セクションのメタデータ属性、更新ポリシー属性、およびプロパティ値でのみ使用できます。

Condition キーの使用

条件を定義すると、Condition キーを使用して、テンプレート内の `Resources` や `Outputs` などの複数の場所に条件を適用できます。Condition キーは条件の論理名を参照し、指定された条件の評価結果を返します。

トピック

- [条件をリソースに関連付ける](#)
- [条件を出力に関連付ける](#)
- [他の条件での条件の参照](#)
- [Fn::If を使用して条件付きでプロパティ値を返す](#)

条件をリソースに関連付ける

条件付きでリソースを作成するには、Condition キーと条件の論理 ID をリソースの属性として追加します。CloudFormation は、条件が `true` に評価される場合にだけ リソースを作成します。

JSON

```
"NewVolume" : {
  "Type" : "AWS::EC2::Volume",
  "Condition" : "IsProduction",
  "Properties" : {
    "Size" : "100",
    "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ] }
  }
}
```

YAML

```
NewVolume:
  Type: AWS::EC2::Volume
  Condition: IsProduction
```

```
Properties:
  Size: 100
  AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

条件を出力に関連付ける

条件を出力に関連付けることもできます。CloudFormation は、関連付けられた条件が true に評価された場合にのみ出力を作成します。

JSON

```
"Outputs" : {
  "VolumeId" : {
    "Condition" : "IsProduction",
    "Value" : { "Ref" : "NewVolume" }
  }
}
```

YAML

```
Outputs:
  VolumeId:
    Condition: IsProduction
    Value: !Ref NewVolume
```

他の条件での条件の参照

Conditions セクションで条件を定義する際、Condition キーを使用して他の条件を参照できます。これにより、複数の条件を組み合わせ、より複雑な条件付きロジックを作成できます。

次の例では、IsProduction および IsFeatureEnabled 条件が true に評価された場合にのみ、IsProdAndFeatureEnabled 条件が true に評価されます。

JSON

```
"Conditions": {
  "IsProduction" : {"Fn::Equals" : [{"Ref" : "Environment"}, "prod"]},
  "IsFeatureEnabled" : { "Fn::Equals" : [{"Ref" : "FeatureFlag"}, "enabled"]},
  "IsProdAndFeatureEnabled" : {
    "Fn::And" : [
      {"Condition" : "IsProduction"},
      {"Condition" : "IsFeatureEnabled"}
    ]
  }
}
```

```
}  
}
```

YAML

```
Conditions:  
  IsProduction:  
    !Equals [!Ref Environment, "prod"]  
  IsFeatureEnabled:  
    !Equals [!Ref FeatureFlag, "enabled"]  
  IsProdAndFeatureEnabled: !And  
    - !Condition IsProduction  
    - !Condition IsFeatureEnabled
```

Fn::If を使用して条件付きでプロパティ値を返す

より詳細な制御を行う場合には、**Fn::If** 組み込み関数を使用すると、リソースまたは出力内の 2 つのプロパティ値のいずれかを条件付きで返すことができます。この関数は条件を評価し、条件が `true` の場合は 1 つの値を返し、条件が `false` の場合は別の値を返します。

条件付きプロパティの値

次の例は、環境条件に基づいて EC2 インスタンスタイプを設定する方法を示します。IsProduction 条件が `true` に評価された場合、インスタンスタイプは `c5.xlarge` に設定されます。それ以外の場合、`t3.small` に設定されます。

JSON

```
"Properties" : {  
  "InstanceType" : {  
    "Fn::If" : [  
      "IsProduction",  
      "c5.xlarge",  
      "t3.small"  
    ]  
  }  
}
```

YAML

```
Properties:  
  InstanceType: !If
```

- IsProduction
- c5.xlarge
- t3.small

条件付きプロパティの削除

AWS::NoValue 擬似パラメータを戻り値として使用し、条件が false の場合に対応するプロパティを削除することもできます。

JSON

```
"DBSnapshotIdentifier" : {
  "Fn::If" : [
    "UseDBSnapshot",
    {"Ref" : "DBSnapshotName"},
    {"Ref" : "AWS::NoValue"}
  ]
}
```

YAML

```
DBSnapshotIdentifier: !If
- UseDBSnapshot
- !Ref DBSnapshotName
- !Ref "AWS::NoValue"
```

例

トピック

- [環境ベースのリソースの作成](#)
- [複数条件のリソースプロビジョニング](#)

環境ベースのリソースの作成

次の例では、EC2 インスタンスをプロビジョニングし、環境タイプが prod の場合にのみ、条件付きで新しい EBS ボリュームを作成してアタッチします。環境が test の場合、追加のボリュームなしで EC2 インスタンスを作成するだけです。

JSON

```
{
```

```
"AWSTemplateFormatVersion": "2010-09-09",
"Parameters": {
  "EnvType": {
    "Description": "Environment type",
    "Default": "test",
    "Type": "String",
    "AllowedValues": [
      "prod",
      "test"
    ],
    "ConstraintDescription": "must specify prod or test"
  }
},
"Conditions": {
  "IsProduction": {
    "Fn::Equals": [
      {
        "Ref": "EnvType"
      },
      "prod"
    ]
  }
},
"Resources": {
  "EC2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "ami-1234567890abcdef0",
      "InstanceType": "c5.xlarge"
    }
  },
  "MountPoint": {
    "Type": "AWS::EC2::VolumeAttachment",
    "Condition": "IsProduction",
    "Properties": {
      "InstanceId": {
        "Ref": "EC2Instance"
      },
      "VolumeId": {
        "Ref": "NewVolume"
      },
      "Device": "/dev/sdh"
    }
  }
},
```



```

    "NewVolume": {
      "Type": "AWS::EC2::Volume",
      "Condition": "IsProduction",
      "Properties": {
        "Size": 100,
        "AvailabilityZone": {
          "Fn::GetAtt": [
            "EC2Instance",
            "AvailabilityZone"
          ]
        }
      }
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Description: Environment type
    Default: test
    Type: String
    AllowedValues:
      - prod
      - test
    ConstraintDescription: must specify prod or test
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: c5.xlarge
  MountPoint:
    Type: AWS::EC2::VolumeAttachment
    Condition: IsProduction
    Properties:
      InstanceId: !Ref EC2Instance

```

```
    VolumeId: !Ref NewVolume
    Device: /dev/sdh
  NewVolume:
    Type: AWS::EC2::Volume
    Condition: IsProduction
    Properties:
      Size: 100
      AvailabilityZone: !GetAtt
        - EC2Instance
        - AvailabilityZone
```

複数条件のリソースプロビジョニング

次の例では、バケット名が指定されている場合、条件付きで S3 バケットを作成し、環境が `prod` に設定されている場合にのみバケットポリシーをアタッチします。バケット名が指定されていない場合、または環境が `test` の場合、リソースは作成されません。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "EnvType": {
      "Type": "String",
      "AllowedValues": [
        "prod",
        "test"
      ]
    },
    "BucketName": {
      "Default": "",
      "Type": "String"
    }
  },
  "Conditions": {
    "IsProduction": {
      "Fn::Equals": [
        {
          "Ref": "EnvType"
        },
        "prod"
      ]
    }
  },
}
```

```
    "CreateBucket": {
      "Fn::Not": [
        {
          "Fn::Equals": [
            {
              "Ref": "BucketName"
            },
            ""
          ]
        }
      ]
    },
    "CreateBucketPolicy": {
      "Fn::And": [
        {
          "Condition": "IsProduction"
        },
        {
          "Condition": "CreateBucket"
        }
      ]
    }
  ],
  "Resources": {
    "Bucket": {
      "Type": "AWS::S3::Bucket",
      "Condition": "CreateBucket",
      "Properties": {
        "BucketName": {
          "Ref": "BucketName"
        }
      }
    },
    "Policy": {
      "Type": "AWS::S3::BucketPolicy",
      "Condition": "CreateBucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "Bucket"
        },
        "PolicyDocument": { ... }
      }
    }
  }
}
```

```
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Type: String
    AllowedValues:
      - prod
      - test
  BucketName:
    Default: ''
    Type: String
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
  CreateBucket: !Not
    - !Equals
      - !Ref BucketName
      - ''
  CreateBucketPolicy: !And
    - !Condition IsProduction
    - !Condition CreateBucket
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Condition: CreateBucket
    Properties:
      BucketName: !Ref BucketName
  Policy:
    Type: AWS::S3::BucketPolicy
    Condition: CreateBucketPolicy
    Properties:
      Bucket: !Ref Bucket
      PolicyDocument: ...
```

この例では、CreateBucketPolicy 条件が、Condition キーを使用して他の条件を参照する方法を示しています。ポリシーは、IsProduction および CreateBucket 条件の両方が true に評価された場合にのみ作成されます。

Note

条件を使用するより複雑な例については、「CloudFormation テンプレートリファレンスガイド」の「[Condition 属性](#)」トピックを参照してください。

CloudFormation テンプレートの Transform セクション

Transform セクション (オプション) では、CloudFormation が何らかの方法でテンプレートを処理するために使用するマクロを 1 つ以上指定します。

マクロを使うと、テキストの検索や置換といった簡単なタスクを実行できるほか、テンプレート全体に対して広範に変換を行うことができます。CloudFormation は、指定された順にマクロを実行します。変更セットを作成すると、CloudFormation は処理したテンプレートコンテンツを変更セットに含めます。その後、変更内容を確認して変更セットを実行できます。マクロの仕組みの詳細については、「[テンプレートマクロを使用して CloudFormation テンプレートでカスタム処理を実行する](#)」を参照してください。

CloudFormation は、トランスフォームもサポートしています。トランスフォームとは、CloudFormation がホストするマクロのことです。トランスフォームの実行順序とスコープは、自分で作成したマクロと同じように処理されます。詳細については、「[変換のリファレンス](#)」を参照してください。

複数のマクロを宣言するには、リスト形式を使用して 1 つ以上のマクロを指定します。

例えば、以下のテンプレートサンプルでは、CloudFormation はまず MyMacro を評価し、次に AWS::Serverless を評価します。どちらも Transform セクションに含まれているため、テンプレート全体のコンテンツを処理できます。

```
# Start of processable content for MyMacro and AWS::Serverless
Transform:
  - MyMacro
  - 'AWS::Serverless'
Resources:
  WaitCondition:
    Type: AWS::CloudFormation::WaitCondition
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
```

```
    Tags: [{"key": "value"}]
    CorsConfiguration: []
MyEc2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: ami-1234567890abcdef0
# End of processable content for MyMacro and AWS::Serverless
```

CloudFormation テンプレート形式のバージョン構文

AWSTemplateFormatVersion セクション (オプション) は、テンプレートが準拠するテンプレート形式バージョンを識別します。最新のテンプレートの形式バージョンは 2010-09-09 であり、現時点で唯一の有効な値です。

テンプレート形式のバージョンは API のバージョンと同じではありません。テンプレート形式のバージョンは API のバージョンとは関係なく変更できます。

テンプレートフォーマットのバージョン宣言の値は、リテラル文字列にする必要があります。テンプレートの形式バージョンの指定には、パラメータまたは機能を使用できません。値を指定しない場合、CloudFormation は最新のテンプレートの形式バージョンを使用します。次のスニペットは、有効なテンプレートの Format Version セクションの例です。

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

CloudFormation テンプレートの Description 構文

Description セクション (オプション) では、テンプレートを記述するテキスト文字列を含めることができます。このセクションは、必ずテンプレートの Format Version セクションの後に記述する必要があります。

説明の宣言の値は、長さ 0 ~ 1024 バイトのリテラル文字列である必要があります。説明の指定には、パラメーターまたは機能を使用できません。次のスニペットは、説明の宣言の例です。

⚠ Important

スタックの更新時に、Description セクションを単独で更新することはできません。更新できるのは、リソースを追加、変更、または削除する変更を含める場合だけです。

JSON

```
"Description" : "Here are some details about the template."
```

YAML

```
Description: > Here are some details about the template.
```

Infrastructure Composer を使用してテンプレートを視覚的に作成

AWS Infrastructure Composer (以前は [Application Composer] と呼ばれていた) は、AWS で最新のアプリケーションを視覚的に構成および設定できるようにします。コードを記述する代わりに、さまざまなリソースをドラッグとドロップの操作でアプリケーションを視覚的に構築できます。

CloudFormation コンソールモードの Infrastructure Composer は、CloudFormation テンプレートを視覚的に操作するために推奨されるツールです。CloudFormation コンソールからアクセスできるこのバージョンの Infrastructure Composer は、CloudFormation デザイナーと呼ばれる古いツールからの改良版です。

CloudFormation コンソールモードで Infrastructure Composer を使用すると、カードと呼ばれるさまざまなリソースをビジュアルキャンバスにドラッグ、ドロップ、設定、接続できます。この視覚的なアプローチにより、テンプレートを直接操作しなくても、アプリケーションアーキテクチャの設計と編集が簡単になります。「[CloudFormation コンソール](#)」からこのモードにアクセスするには、左側のナビゲーションメニューから [Infrastructure Composer] を選択します。

詳細については、「AWS Infrastructure Composer Developer Guide」の「[How to compose in AWS Infrastructure Composer](#)」を参照してください。

CloudFormation コンソールモードで Infrastructure Composer を使用する理由

Infrastructure Composer でテンプレートを視覚化することにより、CloudFormation テンプレートおよびアプリケーションアーキテクチャのギャップと改善対象領域を特定しやすくなります。Infrastructure Composer は、CloudFormation スタックの視覚的な構築および変更を簡単かつ効率的に行うことができるため、開発の操作性が向上します。最初のドラフトから開始してデプロイ可能なコードを作成し、Infrastructure Composer のビジュアルデザイナーでデベロッパーのワークフローを組み込むことができます。

Infrastructure Composer コンソールと異なる内容

Infrastructure Composer の CloudFormation コンソールバージョンには標準の Infrastructure Composer コンソールと同様の機能がありますが、いくつかの違いがあります。Lambda 関連のカード ([Lambda 関数] および [Lambda レイヤー]) には、CloudFormation コンソールモードの Infrastructure Composer では利用できないコードビルドおよびパッケージングソリューションが必要です。ローカル同期は、このモードでは利用できません。

ただし、これらの Lambda 関連のカードおよびローカル同期機能は、「[Infrastructure Composer コンソール](#)」または AWS Toolkit for Visual Studio Code で使用できます。詳細については、「AWS Toolkit for Visual Studio Code ユーザーガイド」の「[AWS Infrastructure Composer デベロッパーガイド](#)」および「[Infrastructure Composer](#)」を参照してください。

AWS CloudFormation 言語サーバー

AWS CloudFormation 言語サーバーは、Infrastructure as Code (IaC) の作成と AWS リソースのデプロイを安全かつ確実に高速化する機能を提供します。[言語サーバープロトコル](#) (LSP) に従って、ホバー、自動補完、静的検証による診断、定義およびコードアクションの参照に関するドキュメントを提供します。これらの従来の言語サーバー機能に加えて、サーバーは CloudFormation を介して AWS リソースを探索およびデプロイするためのオンライン機能を追加しています。これには、変更セットを使用してテンプレートを検証してデプロイする機能、スタック差分、イベント、リソース、および出力を表示する機能、タイプ別にスタックを一覧表示してリソースを参照する機能、ライブリソースの状態を CloudFormation テンプレートに直接挿入する機能などが含まれます。開発者ツールおよび統合開発環境 (IDE) は、言語サーバー機能を活用して、CloudFormation での IaC 開発者のエクスペリエンスを向上させることができます。

トピック

- [AWS CloudFormation 言語サーバーと統合している IDE](#)
- [開始方法](#)
- [IDE で CloudFormation プロジェクトを初期化する](#)
- [オープンソース](#)
- [サポートが必要ですか？](#)

AWS CloudFormation 言語サーバーと統合している IDE

[AWS Toolkit](#) 拡張機能は、CloudFormation 言語サーバーを統合して、オーサリングエクスペリエンスを向上させます。この統合は現在、以下を含む VS Code ベースの IDE でサポートされています。

- [Visual Studio Code](#)
- [Kiro](#)
- [カーソル](#)

開始方法

トピック

- [前提条件](#)
- [ステップ 1: AWS Toolkit をインストールまたはアップグレードする](#)
- [ステップ 2: AWS Toolkit パネルで CloudFormation にアクセスする](#)
- [ステップ 3: テンプレートを検証、テスト、および改善する](#)
- [ステップ 4: テンプレート内をナビゲートする](#)
- [ステップ 5: 検証してデプロイする](#)

前提条件

開始する前に、以下を確認してください。

- サポートされているオペレーティングシステム (macOS、Windows、または Linux) で VS Code ベースの IDE を使用している。
- IDE 用の [AWS Toolkit](#) の最新バージョンをインストールまたはアップグレードしている。

AWS CloudFormation 言語サーバーの一部の機能には、アクティブな AWS アカウント および設定済みの認証情報が必要です。有効な認証情報を使用して、AWS Toolkit を通じて AWS アカウント にサインインする必要があります。

ステップ 1: AWS Toolkit をインストールまたはアップグレードする

- IDE の拡張機能またはプラグインマネージャを開きます。
- 「AWS Toolkit」を検索します。
- IDE 用の Toolkit AWS のバージョン 3.85.0 以降をインストールまたは更新します。
- インストール後に IDE を再起動します。

注: インストール後、AWS Toolkit は自動的に CloudFormation IDE のサポートを有効にします。AWS CloudFormation 言語サーバー拡張機能を使用して AWS Toolkit を初めてインストールまたはアップグレードすると、匿名の使用状況データを収集するために AWS のアクセス許可を付与するよう求められます。このデータは、AWS が CloudFormation 言語サーバーを改善し、オーサリングエクスペリエンスを向上させるのに役立ちます。機密情報は収集されず、AWS はテンプレートコンテンツ、リソース設定、または識別可能な顧客データを記録したり保存したりしません。IDE 設定からテレメトリ設定は、いつでも変更できます。収集された使用状況データは、機能インタラクションとパフォーマンスメトリクスのみに関心を当てています。これらのインサイトは、AWS が検証の迅速化、オートコンプライトの強化、エラー診断の改良など、改善点を特定して優先順位を付けるのに役立ちます。

ステップ 2: AWS Toolkit パネルで CloudFormation にアクセスする

IDE で、アクティビティバーから AWS Toolkit パネルを開きます。AWS Toolkit で、CLOUDFORMATION を選択します。

パネルセクション:

- [リージョン]: 現在の AWS リージョン を表示します。リージョン名を選択するか、コマンドパレットから [AWS CloudFormation: リージョンを選択する] コマンドを使用して変更できます。
- [環境]: CFN init 中に選択した環境 (dev、test、prod など) を示します。この値は、環境選択ステップを完了した後にのみ表示されます。
- [スタック]: アカウントの CloudFormation スタックのページ分割されたリストを表示します。
 - 更新アイコンをクリックして、スタックのリストを更新します。
 - [+] アイコンを使用して、新しいテンプレートをスタックとしてデプロイします。
 - 各スタックエントリには以下が含まれます:

- [概要]: スタックの概要とステータスを表示します
 - スタック ID
 - 説明
 - [作成時刻] と [更新時刻]
 - [ステータス] と [ステータスの理由]
 - [イベント]、[出力]、および [リソース]
 - [変更セット]
- [リソース]: リソースタイプを選択すると、パネルにアカウントのそのタイプの AWS リソースが表示されます。テンプレートを表示、更新、コピー、またはテンプレートにインポートできます。

ステップ 3: テンプレートを検証、テスト、および改善する

CloudFormation テンプレートを作成すると、IDE はインテリジェントオーサリング支援を提供し、正確で準拠したインフラストラクチャをより迅速に作成できるようにします。CloudFormation 言語サーバーは、バックグラウンドで実行され、次のオーサリング機能を提供します。

- コード補完: CloudFormation スキーマに基づいてリソースタイプ、パラメータ、およびプロパティを提案します。
- 既存の AWS リソースの追加: AWS アカウント からテンプレートに既存のリソースをインポートできます。IDE は、[AWS クラウドコントロール API \(CCAPI\)](#) を使用してリソースのライブ設定とプロパティを取得し、テンプレート内の既存のインフラストラクチャのクローンを作成または再利用するのを支援します。

リソースをテンプレートに追加するには

- CloudFormation リソースパネルを展開する: AWS Toolkit サイドパネルで CloudFormation を展開してから、[リソース] で [Add +] アイコンをクリックします。
- リソースタイプを検索する: 特定の AWS リソースタイプを検索するには、検索バーで、追加する特定の AWS リソースタイプを入力します。例:
 - AWS::S3::Bucket
 - AWS::Lambda::Function

コマンドパレットで [AWS CloudFormation: リソースタイプを追加する] を使用して、リソースタイプを選択できます。

- リソースをテンプレートに追加する: [リソース] パネルにアカウントで検出された AWS リソースのページ分割されたリストが表示されます。多くのリソースがある場合は、最初のページのみが表示されます。パネルの下部にあるナビゲーションコントロールを使用して、追加のページ間を移動し、使用可能なすべてのリソースを表示します。
- テンプレートに含めるリソースを選択します。
- リソースをテンプレートに挿入するには、目標に応じて、2 つの方法があります。
 - 既存のリソースのクローンを作成する: 既存の AWS リソースのライブ設定とプロパティを使用して、テンプレートに新しいリソースを作成します。
 - 既存のリソースをインポートする: ライブの状態を使用して実際のリソースをテンプレートに追加することで、そのリソースをスタックに挿入します。

ヒント

- [リソース] パネルはいつでも更新して、アカウントまたはリージョンで利用可能なリソースの最新リストを表示できます。
- リソースをインポートしている場合は、同じアカウントの既存の CloudFormation スタックに既に属しているリソースを追加しないでください。
- リソースがすでに CloudFormation によって管理されているかどうかを確認するには、リソース名の横にある [i] (情報) アイコンをクリックします。
- または、コマンド [AWS CloudFormation: スタック管理情報を取得する] を使用して、リソースがスタックに属しているかどうかを確認できます。

関連リソースを追加する

コマンド [AWS CloudFormation: タイプ別に関連リソースを追加する] を使用して、関連リソースを選択したリソースに追加できます。テンプレートで既に定義されているリソースタイプから選択すると、IDE には、通常そのタイプに関連付けられているか、そのタイプに依存するリソースのリストが表示されます。例えば、AWS::EC2::Instance を選択した場合、IDE は AWS::EC2::SecurityGroup や AWS::EC2::Subnet などの関連リソースの追加を提案する場合があります。この機能は、互換性のあるリソースタイプを手作業で検索することなく、接続されたインフラストラクチャコンポーネントをすばやく構築するのに役立ちます。

静的検証

CloudFormation IDE は、[\[AWS CloudFormation Linter \(cfn-lint\)\]](#) と [\[AWS CloudFormation Guard\]](#) を使用した組み込みの静的検証を提供します。これらの検証は、テンプレートを作成したときにバック

グラウンドで実行され、デプロイ前に構文エラー、コンプライアンスギャップ、ベストプラクティスの問題を特定するのに役立ちます。

静的検証の概要

IDE には 2 種類のリアルタイム静的検証が示されます。

- CloudFormation Linter (cfn-lint): CloudFormation リソース仕様とスキーマルールに対してテンプレートを検証します。
- Guard (cfn-guard): コンプライアンスルールと組織ポリシーパックに対してテンプレートを検証します。

CloudFormation Linter (cfn-lint)

CloudFormation Linter は、IDE に統合されており、入力時にテンプレートの構文と構造を自動的にチェックします。

- スキーマの検証: 構文とスキーマのエラーを検出して、テンプレートが CloudFormation リソーススキーマに準拠していることを確認します。
- エラーの強調表示: デプロイのブロッカーまたは警告を表す赤または黄色の波形の線が問題の下に表示されます。
- マウスオーバーのヘルプ: エラーをマウスオーバーすると、その問題に関連する診断メッセージが IDE に表示されます。クイック修正が利用可能な場合は、ホバーパネルでも提供されます。

Guard 統合

Guard は、コンプライアンスポリシーとセキュリティポリシーを定義するルールセットに対してテンプレートを検証します。IDE は CloudFormation 言語サーバーを介してリアルタイムで Guard 検証を実行し、テンプレートの作成中にすぐにフィードバックを提供します。

- デフォルトルールパック: IDE には、リソースのセキュリティと設定の健全性に関する基本的なベストプラクティスに焦点を当てた、事前登録済みの一連の Guard ルールが含まれています。[ガードルールレジストリ](#)へのリンクの詳細について。
- ルールパックの追加: ルールセットを追加または変更するには、[設定]、[Guard] を開き、追加の Guard ルールパックを選択またはアップロードします。

ヒント: 波形の線のインジケータを理解する

- 青の波形: ベストプラクティスのヒントまたは最適化の推奨事項。
- 黄色の波形: ノンブロッキングの問題 (タグやパラメータの欠落など) に関する警告。
- 赤の波形: 無効なプロパティ名、必須フィールドの欠落、スキーマの不一致などのデプロイブロッカー。

ステップ 4: テンプレート内をナビゲートする

Explorer でテンプレートファイルをクリックすると、[概要] パネルに CloudFormation テンプレートの構造化された階層ビューが自動的に表示されます。テンプレートを Parameters、Resources、Outputs、Mappings などのセクションに整理し、各リソースタイプと論理 ID を表示します。これにより、大きなテンプレート内の特定のリソースやパラメータをすばやく見つけてナビゲートすることが容易になります。

GetAtt や Ref などの組み込み関数用に [定義に進む] を使用して、テンプレート内の参照されるリソースやパラメータに直接ジャンプすることができます。これは、依存関係を追跡し、リソースの関係を理解し、編集をより効率的に行うことに役立ちます。

ステップ 5: 検証してデプロイする

CloudFormation テンプレートをデプロイする準備ができたなら、コマンドパレットを開き、[AWS CloudFormation: 検証してデプロイする] を実行します。コマンドはテンプレートを検証し、ブロッキングエラーが見つからない場合は、デプロイフェーズに進みます。確認する前に、IDE に [\[リフト対応変更セット\]](#) の概要と差分ビューが表示されるので、提案されたすべての変更を確認できます。

検証の仕組み

IDE は [デプロイ前の検証チェック](#) を自動的に実行し、次のような一般的な障害の原因に対してテンプレートを検証します。

- 無効なプロパティ構文またはスキーマの不一致: これらの問題は通常、オーサリング中に cfn-lint によって見つかりますが、ユーザーがそれらに対処せずにデプロイを進めた場合、CloudFormation のデプロイ時の検証では、スタックが作成または更新される前に同じエラーが表示されます。
- リソース名が、アカウント内の既存のリソースと競合しています。
- S3 バケット名の競合や暗号化の欠落など、サービス固有の制約。

検証でエラーが検出された場合、IDE は赤または黄色の波形の線を使用してテンプレート内の問題を直接強調表示し、PROBLEMS パネルにエラーを一覧表示します。各問題には、推奨される修正とと

もに、障害の原因となった特定のプロパティまたはリソースが含まれます。ブロックエラーがない場合は、デプロイフェーズに進むことができます。

警告が見つかった場合 (ノンブロッキングの問題)、ダイアログが表示され、デプロイを続行するか、キャンセルして修正することができます。

IDE で [\[ドリフト対応変更セット\]](#) が開き、現在のテンプレートとデプロイされたスタック設定の違いが表示されます。これにより、実行前に変更セットを確認、確認、またはキャンセルできます。

注: ドリフト対応変更セットは、スタックドリフトを安全に処理できるようにすることで、CloudFormation のデプロイプロセスを強化します。スタックドリフトは、リソースの実際の状態が CloudFormation テンプレートで定義されているものと異なる場合に発生します。これは多くの場合、AWS マネジメントコンソール、CLI、または SDK を介して行われた手動による変更が原因です。CloudFormation [\[ドリフト対応変更セット\]](#) は、処理されたスタック設定をライブのリソースの状態と比較し、IDE はこれらの違いを明らかにするので、デプロイ前にリソースをコンプライアンス状態に戻すことができます。

スタックイベントを表示する

デプロイが開始されたら、パネルの [CloudFormation] タブにナビゲートして、進行状況をリアルタイムでモニタリングできます。[スタックイベント] には、デプロイ中に実行されたオペレーションのリストが表示されます。各イベントには、次のような詳細が含まれます:

- [タイムスタンプ] — イベントが発生した時刻
- [リソース]: 作成、更新、または削除される特定の AWS リソース
- [ステータス]: オペレーションの現在の状態
(CREATE_IN_PROGRESS、UPDATE_COMPLETE、ROLLBACK_IN_PROGRESS など)
- [理由]: 該当する場合、追加のコンテキストまたはエラーメッセージ

このパネルからスタックの [リソース] と [出力] を表示することもできます。AWS マネジメントコンソールでスタックを開く場合は、スタック名の横にある外部リンクアイコンを使用します。[スタックイベント] ビューは、デプロイの進行状況を追跡し、潜在的な問題を特定し、スタックが正常に完了したことを確認するのに役立ちます。

IDE で CloudFormation プロジェクトを初期化する

IDE で CloudFormation プロジェクトを初期化すると、正しいフォルダ、環境設定、および AWS 認証情報を使用して構造化ワークスペースをセットアップできるため、テンプレートを確実に検証して

デプロイできます。IDE から直接新しい CloudFormation プロジェクトを初期化して、この推奨セットアップを作成できます。

CloudFormation プロジェクトを初期化するには:

- コマンドパレットを開きます
 - IDE から、コマンドパレット (macOS の Ctrl+Shift+P または Cmd+Shift+P) を開きます。
 - [AWS CloudFormation: CFN Init: プロジェクトを初期化する] を選択します。
- プロジェクトディレクトリを選択する
 - デフォルト: IDE は現在の作業ディレクトリを使用します。
 - このパスを CloudFormation テンプレートを保存する任意のフォルダに変更することができます。
- AWS 認証情報プロファイルを選択する
 - AWS 認証情報プロファイルを選択するように求められます。選択したプロファイルは、環境の検出、検証、およびデプロイに使用されます。
- 環境をセットアップする
 - 環境を作成または選択するように求められます。
 - 環境は、テンプレートをデプロイまたは検証する場所と方法 (開発、ベータ、本番稼働など) を定義します。[AWS CloudFormation: CFN Init: 環境を追加する] を使用して、環境を選択または変更できます。
 - [AWS CloudFormation: CFN Init: 環境を削除する] を使用して、選択した環境を削除できます。
- (オプション) パラメータファイルをインポートする
 - 既存のパラメータファイルが既にある場合、IDE は初期化中にそれらをインポートすることを許可します。
 - IDE は互換性のあるファイルを自動的に検出し、テンプレートの検証とデプロイで使用するためにそれらをプロジェクトにリンクします。
- プロジェクトを命名して確定する
 - ベータ環境などのプロジェクト名を提供し、セットアップを完了します。
 - IDE は、最初のプロジェクト構造と設定ファイルを作成します。

IDE から直接、検証を実行したり、デプロイをプレビューしたり、環境を切り替えたりできます。

オープンソース

AWS CloudFormation 言語サーバーは、Apache-2.0 ライセンスでオープンソース化されているため、テンプレート診断、スキーマ検証、および静的分析の実行方法に対する完全な透明性を顧客に提供します。これにより、ツールを採用する前にソースレベルの可視性を必要とする顧客のセキュリティとコンプライアンスの摩擦が軽減されます。

コードベースは GitHub で一般公開されています: <https://github.com/aws-cloudformation/cloudformation-languageserver/>。

サポートが必要ですか？

AWS re:Post で [CloudFormation コミュニティ](#) をお試しください。

laC ジェネレーターを使用して既存のリソースからテンプレートを生成する

laC ジェネレーター (Infrastructure as Code ジェネレーター) では、まだ CloudFormation の管理対象になっていないアカウントにプロビジョニングされた AWS リソースを使用してテンプレートを生成することができます。

laC ジェネレーターの利点は次のとおりです。

- CloudFormation 管理でアプリケーション全体を使用するか、AWS CDK アプリに移行します。
- プロパティでリソースプロパティを記述することなくテンプレートを生成し、それを JSON または YAML 構文に変換します。
- 新しいアカウントかリージョンにリソースをレプリケートするときは、こちらのテンプレートを使用します。

laC 生成プロセスは、以下のステップで構成されています。

1. リソースのスキャン – 最初のステップは、リソースのスキャンを開始することです。このスキャンはリージョン全体で実行され、30 日後に失効します。この間、同じスキャンから複数のテンプレートを作成できます。
2. テンプレートの作成 – テンプレートを作成するには、次の 2 つのオプションがあります。
 - 新しいテンプレートを最初から作成し、スキャンしたリソースと関連リソースをそこに追加します。

- 既存の CloudFormation スタックを開始点として使用して、スキャンされたリソースと関連リソースをそのテンプレートに追加します。
3. リソースのインポート – テンプレートを使用して、リソースを CloudFormation スタックとしてインポートするか、AWS CDK アプリに移行します。

IaC ジェネレーターの機能は、すべての商用リージョンで利用可能で、多くの一般的な AWS リソースタイプをサポートしています。サポートされているリソースの一覧については、「[リソースタイプのサポート](#)」を参照してください。

トピック

- [考慮事項](#)
- [リソースのスキャンに必要な IAM アクセス許可](#)
- [テンプレートの生成、管理、削除によく使用されるコマンド](#)
- [テンプレートを AWS CDK に移行する](#)
- [CloudFormation IaC ジェネレーターを使用してリソーススキャンを開始する](#)
- [CloudFormation コンソールでスキャンの概要を表示する](#)
- [IaC ジェネレーターを使用してスキャンされたリソースから CloudFormation テンプレートを作成する](#)
- [スキャンしたリソースから CloudFormation スタックを作成する](#)
- [書き込み専用プロパティを解決する](#)

考慮事項

読み取り権限のある AWS リソースであれば、JSON テンプレートまたは YAML テンプレートを生成できます。IaC ジェネレーターの機能のテンプレートは、リソースプロパティをプロパティごとに記述することなく、クラウドリソースを確実にかつ迅速にモデル化します。

次の表は、IaC の生成機能に使用できるクォータの一覧です。

名前	フルスキャン	部分スキャン
1 回のスキャンで処理できるリソースの最大数	100,000	100,000
1 日あたりのスキャン数 (スキャンするリソースの数が 10,000 未満の場合)	10	10

名前	フルスキャン	部分スキャン
1 日あたりのスキャン数 (スキャンするリソースの数が 10,000 以上の場合)	1	1
アカウントあたりの同時に生成されるテンプレートの数	5	5
1 回のテンプレート生成で同時にモデル化されるリソースの数	5	5
1 つのテンプレートでモデル化できるリソースの合計数	500	500
生成されたテンプレートのアカウントあたりの最大数	1,000	1,000

Important

laC ジェネレーターは、お使いのリージョンの Cloud Control API でサポートされている AWS リソースのみをサポートします。詳細については、「[リソースタイプのサポート](#)」を参照してください。

リソースのスキャンに必要な IAM アクセス許可

laC ジェネレーターでリソースをスキャンするときは、お使いの IAM プリンシパル (ユーザー、ロール、またはグループ) に次のものがが必要です。

- CloudFormation のスキャンのアクセス許可
- ターゲットの AWS サービスの読み取りアクセス許可

スキャンの範囲は、ユーザーが読み取りアクセス許可を持っているリソースに限られます。アクセス許可がないためにスキャンに失敗することはありませんが、そのようなリソースは除外されます。

スキャンとテンプレート管理のアクセス許可を付与する IAM ポリシーの例については、「[すべての laC ジェネレーターオペレーションを許可する](#)」を参照してください。

テンプレートの生成、管理、削除によく使用されるコマンド

laC ジェネレーターの操作用によく使用されるコマンドには以下が含まれます。

- [start-resource-scan](#) - AWS リージョンでアカウント内のリソースのスキャンを開始します。
- [describe-resource-scan](#) - リソーススキャンの進行状況をモニタリングします。
- [list-resource-scans](#) - AWS リージョン内のリソーススキャンを一覧表示します。
- [list-resource-scan-resources](#) - リソーススキャン中に見つかったリソースを一覧表示します。
- [list-resource-scan-related-resources](#) - スキャンしたリソースに関連するリソースを一覧表示します。
- [create-generated-template](#) - スキャンされたリソースのセットから CloudFormation テンプレートを生成します。
- [update-generated-template](#) - 生成されたテンプレートを更新します。
- [describe-generated-template](#) - 生成されたテンプレートに関する情報を返します。
- [list-generated-templates](#) - アカウントと現在のリージョンで生成されたすべてのテンプレートを一覧表示します。
- [delete-generated-template](#) - 生成されたテンプレートを削除します。

テンプレートを AWS CDK に移行する

AWS Cloud Development Kit (AWS CDK) は、一般的なプログラミング言語を使って CloudFormation リソースを開発、管理、デプロイするときには使用できるオープンソースのソフトウェア開発フレームワークです。

AWS CDK CLI は、IaC ジェネレーターとの統合を提供します。CloudFormation テンプレートを変換し、ユーザーのリソースを含む新しい CDK アプリを作成するには、AWS CDK CLI `cdk migrate` コマンドを使用します。その後、AWS CDK を使用してリソースを管理し、CloudFormation にデプロイできます。

詳細については、「AWS Cloud Development Kit (AWS CDK) デベロッパーガイド」の「[Migrate to AWS CDK](#)」を参照してください。

CloudFormation IaC ジェネレーターを使用してリソーススキャンを開始する

既存のリソースからテンプレートを作成する前に、リソースをスキャンして現在のリソースとその関係性を特定する必要があります。

リソーススキャンは、次のいずれかの方法で開始できます。IaC ジェネレーターを使用するのが初めての場合は、1 つ目の方法をお勧めします。

- すべてのリソースをスキャン (フルスキャン) – 現在のアカウントおよびリージョンにある既存のリソースをすべてスキャンします。このスキャンでは、リソース 1,000 件に対して最大 10 分かかります。
- 特定のリソースをスキャン (部分スキャン) – 現在のアカウントおよびリージョンの中からスキャンするリソースタイプを手動で選択します。この方法は 1 つ目の方法よりも処理時間が短く、的を絞ったスキャンが行えるため、反復的なテンプレート開発に向いています。

スキャンが完了すると、テンプレートの生成時に含めるリソースとその関連リソースを選択できます。部分スキャンを使用する場合、関連リソースは、次のいずれかの場合にのみテンプレート生成中に使用できます。

- スキャンを開始する前に関連リソースを具体的に選択した場合、または
- 選択したリソースタイプを特定するときに必要となった場合。

例えば、AWS::EKS::Cluster を選択せずに AWS::EKS::Nodegroup を選択した場合、ノードグループを特定するには事前にクラスターを特定することが必要になるため、IaC ジェネレーターは自動的に AWS::EKS::Cluster リソースをスキャンに含めます。上記以外の場合、スキャンには具体的に選択したリソースのみが含まれます。

Note

先に進む前に、IaC ジェネレーターを操作するための必要な権限を持っていることを確認します。詳細については、「[リソースのスキャンに必要な IAM アクセス許可](#)」を参照してください。

トピック

- [リソーススキャンを開始する \(コンソール\)](#)
- [リソーススキャンを開始する \(AWS CLI\)](#)

リソーススキャンを開始する (コンソール)

すべてのリソースタイプのリソーススキャンを開始するには (フルスキャン)

1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。

2. 画面上部のナビゲーションバーで、スキャンするリソースを含む AWS リージョンを選択します。
3. [スキャン] パネルで [新しいスキャンを開始] を選択し、[すべてのリソースをスキャン] を選択します。

特定のリソースタイプのリソーススキャンを開始するには (部分スキャン)

1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。
2. 画面上部のナビゲーションバーで、スキャンするリソースを含む AWS リージョンを選択します。
3. [スキャン] パネルで [新しいスキャンを開始] を選択し、[特定のリソースをスキャン] を選択します。
4. [部分スキャンを開始] ダイアログボックスで、最大 100 個のリソースタイプを選択し、[スキャンを開始] を選択します。

リソーススキャンを開始する (AWS CLI)

すべてのリソースタイプのリソーススキャンを開始するには (フルスキャン)

以下の [start-resource-scan](#) コマンドを使用します。 *us-east-1* を、スキャンするリソースを含む AWS リージョンに置き換えます。

```
aws cloudformation start-resource-scan --region us-east-1
```

成功した場合、このコマンドはスキャンの ARN を返します。ResourceScanId プロパティの ARN を書き留めます。テンプレートを作成するには、これが必要です。

```
{
  "ResourceScanId":
    "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60"
}
```

特定のリソースタイプのリソーススキャンを開始するには (部分スキャン)

1. 以下の [cat](#) コマンドを使用して、スキャンするリソースタイプをホームディレクトリの config.json という名前の JSON ファイルに保存します。以下は、Amazon EC2 インスタンス

ス、セキュリティグループ、およびすべての Amazon S3 リソースをスキャンする場合のスキャン設定の例です。

```
$ cat > config.json
[
  {
    "Types": [
      "AWS::EC2::Instance",
      "AWS::EC2::SecurityGroup",
      "AWS::S3::*"
    ]
  }
]
```

2. [start-resource-scan](#) コマンドに `--scan-filters` オプションを付けて、作成した `config.json` ファイルと共に部分スキャンを開始します。`us-east-1` を、スキャンするリソースを含む AWS リージョンに置き換えます。

```
aws cloudformation start-resource-scan --scan-filters file://config.json --
region us-east-1
```

成功した場合、このコマンドはスキャンの ARN を返します。ResourceScanId プロパティの ARN を書き留めます。テンプレートを作成するには、これが必要です。

```
{
  "ResourceScanId":
    "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-
a3ef-3e6ecfa5da60"
}
```

リソーススキャンの進行状況を監視する

[describe-resource-scan](#) コマンドを使用します。--resource-scan-id オプションでは、サンプル ARN を実際の ARN に置き換えます。

```
aws cloudformation describe-resource-scan --region us-east-1 \
--resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60
```

成功すると、このコマンドは以下のような出力を返します。

```
{
  "ResourceScanId": "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
  "Status": "COMPLETE",
  "StartTime": "2023-08-21T03:10:38.485000+00:00",
  "EndTime": "2023-08-21T03:20:28.485000+00:00",
  "PercentageCompleted": 100.0,
  "ResourceTypes": [
    "AWS::CloudFront::CachePolicy",
    "AWS::CloudFront::OriginRequestPolicy",
    "AWS::EC2::DHCPOptions",
    "AWS::EC2::InternetGateway",
    "AWS::EC2::KeyPair",
    "AWS::EC2::NetworkAcl",
    "AWS::EC2::NetworkInsightsPath",
    "AWS::EC2::NetworkInterface",
    "AWS::EC2::PlacementGroup",
    "AWS::EC2::Route",
    "AWS::EC2::RouteTable",
    "AWS::EC2::SecurityGroup",
    "AWS::EC2::Subnet",
    "AWS::EC2::SubnetCidrBlock",
    "AWS::EC2::SubnetNetworkAclAssociation",
    "AWS::EC2::SubnetRouteTableAssociation",
    ...
  ],
  "ResourcesRead": 676
}
```

部分スキャンでは、出力は以下のようになります。

```
{
  "ResourceScanId": "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
  "Status": "COMPLETE",
  "StartTime": "2025-03-06T18:24:19.542000+00:00",
  "EndTime": "2025-03-06T18:25:23.142000+00:00",
  "PercentageCompleted": 100.0,
  "ResourceTypes": [
    "AWS::EC2::Instance",
    "AWS::EC2::SecurityGroup",
    "AWS::S3::Bucket",
    "AWS::S3::BucketPolicy"
  ]
}
```



```
    ],  
    "ResourcesRead": 65,  
    "ScanFilters": [  
        {  
            "Types": [  
                "AWS::EC2::Instance",  
                "AWS::EC2::SecurityGroup",  
                "AWS::S3::*"  
            ]  
        }  
    ]  
}
```

出力のフィールドの説明については、「AWS CloudFormation API リファレンス」の「[DescribeResourceScan](#)」を参照してください。

CloudFormation コンソールでスキャンの概要を表示する

スキャンが完了したら、スキャン中に見つかったリソースを視覚化した図を表示して、さまざまな製品タイプ全体でのリソースの集中を特定することができます。

スキャン中に見つかったリソースに関する情報を表示する

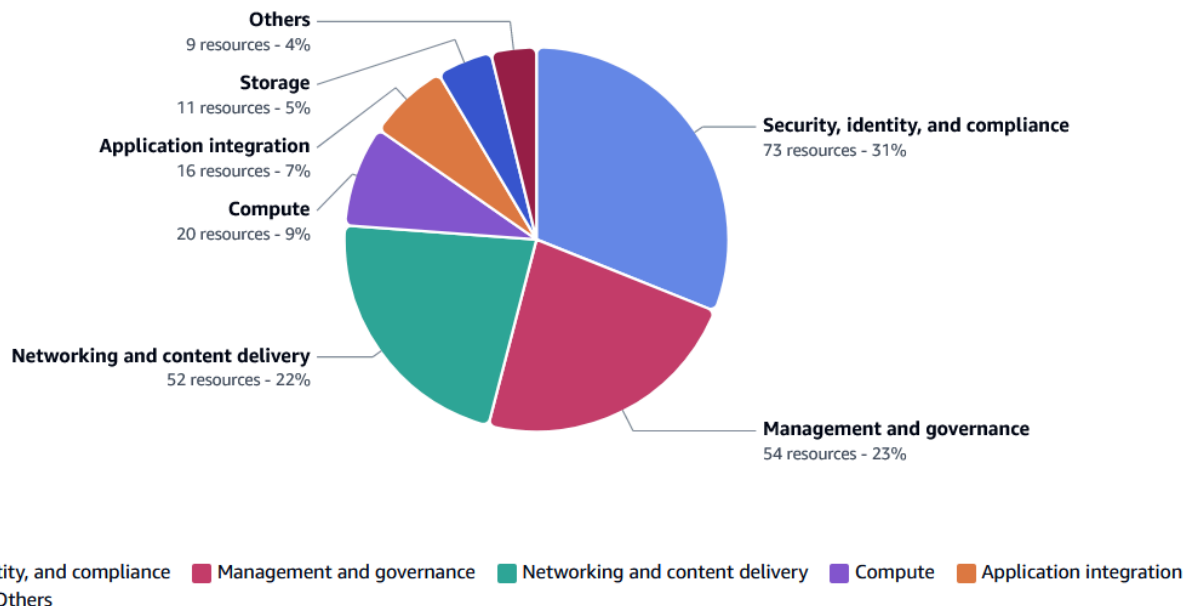
1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。
2. 画面上部のナビゲーションバーで、スキャンするリソースが含まれる AWS リージョンを選択します。
3. ナビゲーションペインで、[IaC ジェネレーター] をクリックします。
4. [スキャンされたリソースの内訳] に、スキャンされたリソースの視覚的な内訳が [コンピューティング] や [ストレージ] などの製品タイプ別に表示されます。
5. 表示される製品タイプの数のカスタマイズするには、[表示されたデータをフィルタリング] を選択します。フィルタリングは、視覚化された図をカスタマイズして、最も関心のある製品タイプに焦点を当てるようにするために役立ちます。
6. ページの右側には、[スキャン概要の詳細] パネルがあります。パネルを開くには、[パネルを開く] アイコンを選択します。

Scanned resources breakdown 235

View a visual breakdown of your scanned resources by product types. Select a product type for a further breakdown.

Filter displayed data

Filter by product type ▼



IaC ジェネレーターを使用してスキャンされたリソースから CloudFormation テンプレートを作成する

このトピックでは、IaC ジェネレーター機能を使用してスキャンされたリソースからテンプレートを作成する方法について説明します。

スキャンしたリソースからテンプレートを作成する (コンソール)

スキャンしたリソースからスタックテンプレートを作成するには

1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。
2. 画面上部のナビゲーションバーで、スキャンしたリソースが含まれる AWS リージョンを選択します。
3. [テンプレート] セクションから [テンプレートを作成] を選択します。
4. [新しいテンプレートから開始] を選択します。
 - a. [テンプレート名] に、テンプレートの名前を入力します。

- b. (オプション) [削除ポリシー] を [置換ポリシーを更新] に設定します。
- c. [次へ] をクリックし、スキャンしたリソースをテンプレートに追加します。
5. [スキャンしたリソースを追加] で、スキャンしたリソースのリストを参照し、テンプレートに追加するリソースを選択します。リソースは、リソース識別子、リソースタイプ、タグのいずれかでフィルターできます。フィルターは相互に包括的です。
6. 必要なリソースすべてをテンプレートに追加したときに、[次へ] を選択し、[スキャンしたリソースを追加] ページを閉じて [関連リソースを追加] ページに進みます。
7. 関連リソースの推奨リストを確認します。Amazon EC2 インスタンスやセキュリティグループなどの関連リソースは、相互に依存しており、通常は同じワークロードに属しています。生成されたテンプレートに含める関連リソースを選択します。

 Note

このテンプレートには、関連リソースをすべて追加することが推奨されます。

8. テンプレートの詳細、スキャンしたリソース、関連リソースを確認します。
9. [テンプレートを作成] をクリックし、[確認して作成] ページを閉じてテンプレートを作成します。

スキャンしたリソースからテンプレートを作成する (AWS CLI)

スキャンしたリソースからスタックテンプレートを作成するには

1. スキャン中に見つかったリソースを一覧表示するには、[list-resource-scan-resources](#) コマンドを使用し、オプションで出力を制限する `--resource-identifier` オプションを指定します。`--resource-scan-id` オプションでは、サンプル ARN を実際の ARN に置き換えます。

```
aws cloudformation list-resource-scan-resources \
  --resource-scan-id arn:aws:cloudformation:us-east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
  --resource-identifier MyApp
```

以下はレスポンスの例です。ManagedByStack は CloudFormation がリソースを既に管理しているかどうかを示します。出力をコピーします。これは次のステップで必要になります。

```
{
  "Resources": [
```

```
{
  "ResourceType": "AWS::EKS::Cluster",
  "ResourceIdentifier": {
    "ClusterName": "MyAppClusterName"
  },
  "ManagedByStack": false
},
{
  "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
  "ResourceIdentifier": {
    "AutoScalingGroupName": "MyAppASGName"
  },
  "ManagedByStack": false
}
]
```

出力の各フィールドについては、「AWS CloudFormation API リファレンス」の「[ScannedResource](#)」を参照してください。

2. cat コマンドを使用して、リソースタイプと識別子をホームディレクトリの `resources.json` という名前の JSON ファイルに保存します。前のステップの出力例に基づいた JSON の例を次に示します。

```
$ cat > resources.json
[
  {
    "ResourceType": "AWS::EKS::Cluster",
    "ResourceIdentifier": {
      "ClusterName": "MyAppClusterName"
    }
  },
  {
    "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
    "ResourceIdentifier": {
      "AutoScalingGroupName": "MyAppASGName"
    }
  }
]
```

3. 作成した `resources.json` ファイルとともに [list-resource-scan-related-resources](#) コマンドを使用して、スキャンしたリソースに関連するリソースを一覧表示します。

```
aws cloudformation list-resource-scan-related-resources \
  --resource-scan-id arn:aws:cloudformation:us-east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
  --resources file://resources.json
```

以下はレスポンスの例です。ManagedByStack は CloudFormation がリソースを既に管理しているかどうかを示します。これらのリソースを、前のステップで作成した JSON ファイルに追加します。テンプレートを作成するには、これが必要です。

```
{
  "RelatedResources": [
    {
      "ResourceType": "AWS::EKS::Nodegroup",
      "ResourceIdentifier": {
        "NodegroupName": "MyAppNodegroupName"
      },
      "ManagedByStack": false
    },
    {
      "ResourceType": "AWS::IAM::Role",
      "ResourceIdentifier": {
        "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"
      },
      "ManagedByStack": false
    }
  ]
}
```

出力の各フィールドについては、「AWS CloudFormation API リファレンス」の「[ScannedResource](#)」を参照してください。

Note

リソースの入力リストの長さは 100 を超えることはできません。100 を超えるリソースの関連リソースを一覧表示するには、100 のバッチで list-resource-scan-related-resources を実行した後、結果を統合します。

一覧に、重複したリソースが出力される場合があるため注意が必要です。

4. これらの変更とともに、次のように [create-generated-template](#) コマンドを使用して新しいスタックテンプレートを作成します。

- *us-east-1* を、スキャンされたリソースが含まれる AWS リージョンに置き換えます。
- *MyTemplate* を、作成するテンプレートの名前に置き換えます。

```
aws cloudformation create-generated-template --region us-east-1 \  
--generated-template-name MyTemplate \  
--resources file://resources.json
```

次は、resources.json ファイルの例です。

```
[  
  {  
    "ResourceType": "AWS::EKS::Cluster",  
    "LogicalResourceId": "MyCluster",  
    "ResourceIdentifier": {  
      "ClusterName": "MyAppClusterName"  
    }  
  },  
  {  
    "ResourceType": "AWS::AutoScaling::AutoScalingGroup",  
    "LogicalResourceId": "MyASG",  
    "ResourceIdentifier": {  
      "AutoScalingGroupName": "MyAppASGName"  
    }  
  },  
  {  
    "ResourceType": "AWS::EKS::Nodegroup",  
    "LogicalResourceId": "MyNodegroup",  
    "ResourceIdentifier": {  
      "NodegroupName": "MyAppNodegroupName"  
    }  
  },  
  {  
    "ResourceType": "AWS::IAM::Role",  
    "LogicalResourceId": "MyRole",  
    "ResourceIdentifier": {  
      "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"  
    }  
  }  
]
```

```
]
```

成功すると、このコマンドは以下を返します。

```
{
  "Arn":
    "arn:aws:cloudformation:region:account-id:generatedtemplate/7fc8512c-d8cb-4e02-b266-d39c48344e48",
  "Name": "MyTemplate"
}
```

スキャンしたリソースから CloudFormation スタックを作成する

テンプレートを作成したら、スタックを作成してスキャンされたリソースをインポートする前に、生成されたテンプレートを Infrastructure Composer でプレビューできます。これは、リソースとその関係を含む完全なアプリケーションアーキテクチャを視覚化するのに役立ちます。Infrastructure Composer の詳細については、「[Infrastructure Composer を使用してテンプレートを視覚的に作成](#)」を参照してください。

スタックを作成してスキャンしたリソースをインポートする

1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。
2. 画面上部にあるナビゲーションバーで、テンプレートの AWS リージョンを選択します。
3. [テンプレート] タブを選択してから、詳細を表示するテンプレートの名前を選択します。
4. [テンプレート] セクションの上部にある [テンプレートの定義] タブでは、希望に応じてテンプレートを YAML 構文から JSON 構文に切り替えることができます。
5. テンプレートの詳細を確認して、すべてが正しく設定されていることを確認します。テンプレートを簡単に確認して理解できるようにするには、Infrastructure Composer を使用して、テンプレートに記述されているインフラストラクチャのデフォルトのコードビューからグラフィカルビューに切り替えることができます。ビューを切り替えるには、[テンプレート] で、[テンプレート] の代わりに [キャンバス] を選択します。

キャンバスのアクション

- テンプレート内の特定のリソースの詳細に焦点を当てるには、カードをダブルクリックして [リソースプロパティ] パネルを表示します。

- キャンバス上のカードを視覚的に並び替えて整理するには、キャンバスの左上から [調整] を選択します。
 - キャンバスを拡大/縮小するには、キャンバスの右下にあるズームコントロールを使用します。
6. コンソールで特定のリソースを表示するには、[テンプレートリソース] タブを選択してから、確認したいリソースの物理 ID を選択します。選択すると、その特定のリソースのコンソールに移動します。[テンプレートリソース] タブからは、テンプレート定義でのリソースの追加、削除、再同期を実行することもできます。
 7. [テンプレートの定義] タブでは、書き込み専用プロパティが含まれるリソースに関する警告を IaC ジェネレーターが発行する場合があります。警告を確認したら、生成されたテンプレートをダウンロードし、必要な変更を加えることができます。詳細については、「[書き込み専用プロパティを解決する](#)」を参照してください。
 8. テンプレートの定義に問題がなければ、[テンプレートの定義] タブで [スタックにインポート] を選択してから、[次へ] を選択します。
 9. [スタックの詳細を指定] ページの [スタックを指定] パネルで、スタックの名前を入力し、[次へ] を選択します。
 10. スタックのパラメータを確認して入力します。[次へ] を選択します。
 11. [変更を確認] ページでオプションを確認し、[次へ] をクリックします。
 12. [確認してインポート] ページで詳細を確認し、[リソースをインポート] をクリックします。

書き込み専用プロパティを解決する

CloudFormation IaC ジェネレーターを使用すると、まだ CloudFormation で管理されていないアカウントにプロビジョニングされたリソースを使ってテンプレートを生成することができます。ただし、特定のリソースプロパティは書き込み専用として指定されます。つまり、データベースパスワードなどは、CloudFormation では書き込みはできますが、読み取りはできません。

既存のリソースから CloudFormation テンプレートを生成する場合、書き込み専用プロパティには課題があります。ほとんどの場合、CloudFormation はこれらのプロパティを生成されたテンプレートのパラメータに変換します。これにより、インポートオペレーション中にこのプロパティをパラメータ値として入力することが可能になります。ただし、この変換が不可能なシナリオがあり、CloudFormation はこれらのケースを別の方法で処理します。

相互に排他的なプロパティ

相互に排他的なプロパティのセットが複数あり、そのうちの一部分が書き込み専用である場合。これらの場合、IaC ジェネレーターは、作成時にリソースに適用されたのはどのプロパティのセットであるかを特定することはできません。例えば、以下のプロパティセットのいずれか 1 つを使うと、[AWS::Lambda::Function](#) のコードを指定できます。

- Code/S3Bucket、Code/S3Key、オプションで Code/S3ObjectVersion
- Code/ImageUri
- Code/ZipFile

これらのプロパティはすべて書き込み専用です。IaC ジェネレーターは、排他的なプロパティセットのうち 1 つを選択し、それを生成されたテンプレートに追加します。それぞれの書き込み専用プロパティにパラメータが追加されます。パラメータ名には `OneOf` が含まれ、パラメータの説明には、対応するプロパティは他の排他的なプロパティに置き換えることが可能であることが示されています。IaC ジェネレーターは、含まれているプロパティに `MUTUALLY_EXCLUSIVE_PROPERTIES` の警告タイプを設定します。

相互に排他的なタイプ

いくつかの場合、書き込み専用プロパティは、複数あるデータタイプに設定できます。例えば、[AWS::ApiGateway::RestApi](#) の `Body` プロパティは、`object` か `string` にすることができます。この場合、IaC ジェネレーターは `string` のタイプを使用して生成されたテンプレートにプロパティを含め、`MUTUALLY_EXCLUSIVE_TYPES` の警告タイプを設定します。

Array のプロパティ

書き込み専用プロパティのタイプが `array` の場合、パラメータはスカラー値にしかならないため、IaC ジェネレーターは生成されたテンプレートに含めることができません。この場合、プロパティはテンプレートから省略され、警告タイプ `UNSUPPORTED_PROPERTIES` が設定されます。

オプションプロパティ

オプションの書き込み専用プロパティの場合、IaC ジェネレーターは、リソースの設定時にプロパティが使用されたかどうかを検出することはできません。この場合、プロパティは生成されたテンプレートから省略され、警告タイプ `UNSUPPORTED_PROPERTIES` が設定されます。

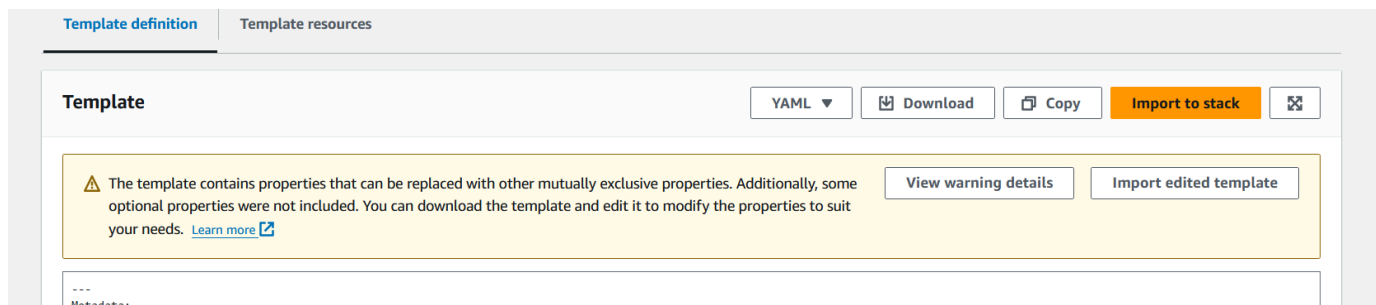
警告と次のステップ

どのプロパティが書き込み専用であるかを判断するには、IaC ジェネレーターコンソールが返した警告を確認する必要があります。[AWS リソースとプロパティタイプのリファレンス](#)は、プロパティが書き込み専用かどうか、または複数のタイプをサポートしているかどうかを示しません。

代替手段として、リソースプロバイダスキーマからどのプロパティが書き込み専用かを確認することもできます。リソースプロバイダスキーマをダウンロードするには、「[CloudFormation リソースプロバイダスキーマ](#)」を参照してください。

書き込み専用プロパティの問題を解決する

1. CloudFormation コンソールの [IaC ジェネレーターページ](#)を開きます。
2. 画面上部にあるナビゲーションバーで、テンプレートの AWS リージョンを選択します。
3. [テンプレート] タブを選択してから、作成したテンプレートの名前を選択します。
4. 生成されたテンプレートに書き込み専用プロパティを持つリソースが含まれる場合は、IaC ジェネレーターのコンソールの [テンプレートの定義] タブに、問題のタイプの概要を伴う警告が表示されます。例:



5. 警告の詳細については、[警告の詳細を表示] を選択します。書き込み専用プロパティを持つリソースは、生成されたテンプレートとリソースタイプで使用する論理 ID によって特定されます。

警告のリストを使用して、書き込み専用プロパティを持つリソースを特定し、各リソースを調べて、生成されたテンプレートに加えるべき変更 (もしある場合) を判断します。

Resource warnings (1)

▼ LambdaFunction001aClambda002fd6K (AWS::Lambda::Function) [Learn more](#)

Unsupported properties

1. SnapStart/ApplyOn

Applying SnapStart setting on function resource type. Possible values: [PublishedVersions, None] This property can be replaced with other types

2. Code/S3ObjectVersion

For versioned objects, the version of the deployment package object to use. This property can be replaced with other exclusive properties

Mutually exclusive properties

1. Code/S3Bucket

An Amazon S3 bucket in the same AWS Region as your function. The bucket can be in a different AWS account. This property can be replaced with other exclusive properties

2. Code/S3Key

The Amazon S3 key of the deployment package. This property can be replaced with other exclusive properties

6. 書き込み専用プロパティの問題を解決するためにテンプレートを更新する必要がある場合は、次の手順を実行します。
 - a. [ダウンロード] を選択して、テンプレートのコピーをダウンロードします。
 - b. テンプレートを編集します。
 - c. 変更し終わったら、[編集したテンプレートをインポート] ボタンを選択してインポートプロセスを続行できます。

AWS::ApiGateway::RestAPI リソースの書き込み専用プロパティの問題を解決する方法

このトピックでは、IaC ジェネレーターを使用するときの [AWS::ApiGateway::RestApi](#) リソースの書き込み専用プロパティの問題を解決する方法について説明します。

問題

生成されたテンプレートに `AWS::ApiGateway::RestApi` リソースが含まれている場合、`Body`、`BodyS3Location`、`CloneFrom` プロパティが `UNSUPPORTED_PROPERTIES` として認識されたことを示す警告が表示されます。これは、これらが任意の書き込み専用プロパティであるためです。IaC ジェネレーターは、これらのプロパティがこのリソースに適用されているかどうかを認識しません。したがって、生成されたテンプレートではこれらのプロパティは省略されます。

解決策

REST API の `Body` プロパティを設定するには、生成されたテンプレートを更新します。

1. Amazon API Gateway [GetExport](#) API アクションを使用して API をダウンロードします。例えば、[aws apigateway get-export](#) AWS CLI コマンドを使用します。詳細については、「API

Gateway デベロッパーガイド」の「[API Gateway から REST API をエクスポートする](#)」を参照してください。

2. GetExport API アクションのレスポンスから Body プロパティを取得します。これを Amazon S3 バケットにアップロードします。
3. 生成されたテンプレートをダウンロードします。
4. このテンプレートに BodyS3Location/Bucket と BodyS3Location/Key のプロパティを追加し、Body が保存されているバケット名とキーを指定します。
5. 生成されたテンプレートを IaC ジェネレーターのコンソールで開き、[編集したテンプレートをインポート] をクリックします。

AWS::Lambda::Function リソースの書き込み専用プロパティの問題を解決する方法

このトピックでは、IaC ジェネレーターを使用するときの [AWS::Lambda::Function](#) リソースの書き込み専用プロパティの問題を解決する方法について説明します。

問題

AWS::Lambda::Function リソースには、Lambda コードを指定するための相互に排他的なプロパティのセットが 3 つあります。

- Code/S3Bucket と Code/S3Key のプロパティ、およびオプションで Code/S3ObjectVersion のプロパティです。
- Code/ImageUri プロパティ
- Code/ZipFile プロパティ

特定の AWS::Lambda::Function リソースに使用できるのは、これらのセットの 1 つだけです。

IaC ジェネレーターは、リソースの作成または更新に排他的な書き込み専用プロパティのどのセットが使用されたのかを特定することはできません。そのため、これには、生成されたテンプレートの最初のプロパティのセットのみが含まれます。Code/ImageUri プロパティと Code/ZipFile プロパティは省略されます。

さらに、IaC ジェネレーターは次の警告を発行します。

- **MUTUALLY_EXCLUSIVE_PROPERTIES** – Code/S3Bucket および Code/S3Key が相互に排他的なプロパティとして識別されることを警告します。

- **UNSUPPORTED_PROPERTIES** – Code/S3ObjectVersion プロパティがサポートされていないことを警告します。

生成されたテンプレートに `AWS::Lambda::Function` リソースを含めるには、正しいコードプロパティでテンプレートをダウンロードして更新する必要があります。

解決策

Lambda コードを Amazon S3 バケットに保存し、**S3ObjectVersion** プロパティを使用しない場合、生成されたテンプレートを修正せずにそのままインポートすることができます。IaC ジェネレーターは、インポートオペレーション中に、テンプレートのパラメータとして Amazon S3 バケットとキーを要求します。

Lambda コードを Amazon ECR リポジトリとして保存する場合は、以下の手順に従ってテンプレートを更新します。

1. 生成されたテンプレートをダウンロードします。
2. 生成されたテンプレートから、Code/S3Bucket および Code/S3Key プロパティのプロパティと対応するパラメータを削除します。
3. 生成されたテンプレートで削除したプロパティを Code/ImageUri プロパティに置き換え、Amazon ECR リポジトリの URL を指定します。
4. 生成されたテンプレートを IaC ジェネレーターのコンソールで開き、[編集したテンプレートをインポート] をクリックします。

Lambda コードを zip ファイルとして保存する場合は、以下の手順に従ってテンプレートを更新します。

1. 生成されたテンプレートをダウンロードします。
2. 生成されたテンプレートから、Code/S3Bucket および Code/S3Key プロパティのプロパティと対応するパラメータを削除します。
3. 生成されたテンプレートの削除されたプロパティを Code/ZipFile プロパティに置き換えます。
4. 生成されたテンプレートを IaC ジェネレーターのコンソールで開き、[編集したテンプレートをインポート] をクリックします。

Lambda コードのコピーがない場合は、以下の手順に従ってテンプレートを更新します。

1. AWS Lambda [GetFunction](#) API アクションを使用します ([aws lambda get-function](#) AWS CLI コマンドを使用するなど)。
2. レスポンスの `RepositoryType` パラメータは、コードが Amazon S3 バケットにある場合は S3、コードが Amazon ECR リポジトリにある場合は ECR です。
3. レスポンスの `Location` パラメータには、デプロイパッケージを 10 分間ダウンロードするときを使用できる、署名付き URL が含まれています。コードをダウンロードします。
4. このコードを Amazon S3 バケットにファイルをアップロードします。
5. 生成されたテンプレートを使ってインポートオペレーションを実行し、パラメータ値としてバケット名とキーを指定します。

動的参照を使用して他のサービスに格納されている値を取得する

動的参照には、他のサービスに格納および管理されている外部値を指定し、Infrastructure as Code (IaC) テンプレートから機密情報を切り離すことができる便利な方法があります。CloudFormation は、スタックオペレーションおよび変更セットオペレーション中に、必要に応じて指定された参照の値を取得します。

動的参照を使用すると、次のことができます。

- **SecureString を使用する** – 機密データの場合、保管中のデータが暗号化されるように、常に AWS Systems Manager Parameter Store または AWS Secrets Manager のシークレットで SecureString パラメータを使用します。
- **アクセスを制限する** – Parameter Store パラメータまたは Secrets Manager シークレットへのアクセスを、認可されたプリンシパルとロールのみに制限します。
- **認証情報をローテーションする** – Parameter Store または Secrets Manager に保存されている機密データを定期的にローテーションして、高いレベルのセキュリティを維持します。
- **ローテーションを自動化する** – Secrets Manager の自動ローテーション機能を活用して、機密データを定期的に更新し、アプリケーションや環境で分散します。

一般的な考慮事項

CloudFormation テンプレートで動的参照を指定する前に考慮すべき一般的な考慮事項を次に示します。

- リソースのプライマリ識別子の一部であるリソースプロパティに、動的参照や機密データを含めないでください。CloudFormation は、プライマリリソース識別子で実際のプレーンテキスト値を使用する場合がありますが、これはセキュリティリスクとなる可能性があります。このリソース ID は、派生した出力または送信先に表示されます。

リソースタイプのプライマリ識別子を構成するリソースプロパティを確認するには、「[AWS リソースおよびプロパティタイプのリファレンス](#)」でそのリソースのリソースリファレンスドキュメントを参照してください。[Return values] (戻り値) セクションの Ref 関数の戻り値は、リソースタイプのプライマリ識別子を構成するリソースプロパティを表します。

- スタックテンプレートには最大 60 個の動的な参照を含めることができます。
- 変換 (AWS::Include や AWS::Serverless など) を使用している場合、CloudFormation は変換を適用する前に動的参照を解決しません。代わりに、動的参照のリテラル文字列を変換に渡し、テンプレートを使用して変更セットを実行する際に参照を解決します。
- カスタムリソース内の安全な値 (Parameter Store や Secrets Manager に保存されている値など) に動的参照を使用することはできません。
- また、動的参照は AWS::CloudFormation::Init メタデータと Amazon EC2 UserData プロパティでもサポートされていません。
- バックスラッシュ (\) で終わる動的参照を作成しないでください。CloudFormation はこれらの参照を解決できず、スタックオペレーションが失敗します。

次のトピックでは、動的参照の使用に関する情報と他の考慮事項について説明します。

トピック

- [Systems Manager Parameter Store からプレーンテキスト値を取得する](#)
- [Systems Manager Parameter Store から SecureString 値を取得する](#)
- [Secrets Manager からシークレットまたはシークレット値を取得する](#)

Systems Manager Parameter Store からプレーンテキスト値を取得する

CloudFormation テンプレートを作成する際には、Parameter Store に保存されているプレーンテキスト値を使用することをお勧めします。パラメータストアは AWS Systems Manager の一機能です。Parameter Store の概要については、「AWS Systems Manager ユーザーガイド」の「[AWS Systems Manager Parameter Store](#)」を参照してください。

テンプレート内で Parameter Store のプレーンテキスト値を使用するには、ssm 動的リファレンスを使用します。この参照では、Parameter Store のタイプ String または StringList のパラメータの値にアクセスできます。

スタック操作で使用する ssm 動的参照のバージョンを確認するには、スタック操作の変更セットを作成します。次に、[テンプレート] タブで処理済みテンプレートをレビューします。詳細については、「[CloudFormation スタックの変更セットを作成する](#)」を参照してください。

ssm 動的参照を使用する場合、留意すべき重要な点がいくつかあります。

- CloudFormation は、動的参照でのドリフト検出をサポートしていません。パラメータのバージョンを指定していない ssm 動的参照の場合、Systems Manager でパラメータのバージョンを更新する際は、最新バージョンのパラメータに変更するために、ssm 動的参照を含むすべてのスタックでスタックの更新オペレーションも実施することをお勧めします。
- CloudFormation テンプレートの Parameters セクションで ssm 動的参照を使用するには、バージョン番号を含める必要があります。CloudFormation では、このセクションのバージョン番号なしで Parameter Store 値を参照することはできません。あるいは、テンプレートで Systems Manager パラメータタイプとしてパラメータを定義することもできます。これを行う際には、Systems Manager パラメータキーを、パラメータのデフォルト値として指定できます。その後、バージョン番号を指定しなくても、CloudFormation は Parameter Store から最新バージョンのパラメータ値を取得します。これにより、テンプレートをよりシンプルかつ簡単にメンテナンスできます。詳細については、「[CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)」を参照してください。
- カスタムリソースの場合、CloudFormation はリクエストをカスタムリソースに送信する前に ssm の動的参照を解決します。
- CloudFormation は、別の AWS アカウント から共有されたパラメータを参照する際の動的参照の使用をサポートしません。
- CloudFormation は、動的参照での Systems Manager パラメータラベルの使用をサポートしません。

アクセス許可

Systems Manager パラメータストアに保存されているパラメータを指定するには、指定されたパラメータに対して [GetParameters](#) を呼び出すためのアクセス許可が必要です。特定の Systems Manager パラメータへのアクセスを提供する IAM ポリシーを作成する方法については、「AWS Systems Manager ユーザーガイド」の「[IAM ポリシーを使用して Systems Manager パラメータへのアクセスを制限する](#)」を参照してください。

リファレンスパターン

CloudFormation テンプレートで Systems Manager Parameter Store に保存されているプレーンテキスト値を参照するには、次の ssm 参照パターンを使用します。

```
{{resolve:ssm:parameter-name:version}}
```

参照は、parameter-name と version について、以下の正規表現パターンに従う必要があります。

```
{{resolve:ssm:[a-zA-Z0-9_.\-/]+(:\d+)?}}
```

parameter-name

パラメータストア内のパラメータの名前。パラメータ名では大文字と小文字が区別されます

必須。

version

使用するパラメータのバージョンを指定する整数。正確なバージョンを指定しない場合、CloudFormation は、スタックを作成または更新するたびに最新バージョンのパラメータを使用します。詳細については、「AWS Systems Manager ユーザーガイド」の「[パラメータバージョンの使用](#)」を参照してください。

オプション。

例

トピック

- [パブリック AMI ID パラメータ](#)
- [カスタム AMI ID パラメータ](#)

パブリック AMI ID パラメータ

次の例では、パブリック AMI パラメータを参照する EC2 インスタンスを作成します。動的リファレンスは、パブリックパラメータから最新の Amazon Linux 2023 AMI ID を取得します。パブリックパラメータの詳細については、「AWS Systems Manager ユーザーガイド」の「[Parameter Store でのパブリックパラメータの検出](#)」を参照してください。

JSON

```
{
  "Resources": {
    "MyInstance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}",
        "InstanceType": "t2.micro"
      }
    }
  }
}
```

YAML

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}'
      InstanceType: t2.micro
```

カスタム AMI ID パラメータ

次の例は、Parameter Store に保存されているカスタム AMI ID を参照する EC2 起動テンプレートを作成します。動的参照は、起動テンプレートからインスタンスが起動されると *golden-ami* パラメータの **2** から AMI ID を取得します。

JSON

```
{
  "Resources": {
    "MyLaunchTemplate": {
      "Type": "AWS::EC2::LaunchTemplate",
      "Properties": {
        "LaunchTemplateName": {
          "Fn::Sub": "${AWS::StackName}-launch-template"
        },

```

```

        "LaunchTemplateData": {
            "ImageId": "{{resolve:ssm:golden-ami:2}}",
            "InstanceType": "t2.micro"
        }
    }
}
}
}

```

YAML

```

Resources:
  MyLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: '{{resolve:ssm:golden-ami:2}}'
        InstanceType: t2.micro

```

Systems Manager Parameter Store から SecureString 値を取得する

CloudFormation では、機密データを「SecureString」として AWS Systems Manager Parameter Store に保存することで、パスワードやライセンスキーなどの機密データをテンプレートで直接公開することなく使用できます。Parameter Store の概要については、「AWS Systems Manager ユーザーガイド」の「[AWS Systems Manager Parameter Store](#)」を参照してください。

テンプレートで Parameter Store の Secure String を使用するには、`ssm-secure` 動的参照を使用します。CloudFormation が実際の SecureString 値を保存することはありません。代わりに、SecureString のプレーンテキストパラメータ名を含むリテラルな動的参照のみを保存します。

スタックの作成時または更新時に、CloudFormation は、実際の値を公開することなく、必要に応じて SecureString 値にアクセスします。SecureString は、`ssm-secure` の動的参照パターンをサポートするリソースプロパティのためにのみ使用できます。詳細については、「[Secure String のための動的なパラメータパターンをサポートするリソース](#)」を参照してください。

CloudFormation は、どの API コールでも SecureString の実際のパラメータ値を返しません。リテラル動的参照のみを返します。変更セットを使用して変更を比較する場合、CloudFormation は、リテラルの動的参照文字列のみを比較します。実際の SecureString の値を解決して比較することはありません。

ssm-secure 動的参照を使用する場合、留意すべき重要な点があります。

- CloudFormation は、他の AWS アカウント から Parameter Store 値にアクセスできません。
- CloudFormation は、動的参照での Systems Manager パラメータラベルまたはパブリックパラメータの使用をサポートしません。
- cn-north-1 および cn-northwest-1 リージョンでは、SecureString は Systems Manager によってサポートされていません。
- ssm-secure などの安全な値の動的参照は、現在カスタムリソースではサポートされていません。
- CloudFormation がスタックの更新をロールバックする必要があり、以前に指定したバージョンの SecureString パラメータが利用できなくなっている場合、ロールバックオペレーションは失敗します。このような場合、次の 2 つのオプションがあります。
 - リソースをスキップするには CONTINUE_UPDATE_ROLLBACK を使用してください。
 - Systems Manager パラメータストアで Secure String パラメータを再作成し、パラメータのバージョンがテンプレートで使用されているバージョンに達するまでそれを更新します。その後、リソースをスキップせずに CONTINUE_UPDATE_ROLLBACK を使用します。

Secure String のための動的なパラメータパターンをサポートするリソース

ssm-secure の動的参照パターンをサポートするリソースには、以下のものがあります。

リソース	プロパティタイプ	プロパティ
AWS::DirectoryService::MicrosoftAD		Password
AWS::DirectoryService::SimpleAD		Password
AWS::ElastiCache::ReplicationGroup		AuthToken
AWS::IAM::User	LoginProfile	Password
AWS::KinesisFirehose::DeliveryStream	RedshiftDestinationConfiguration	Password

リソース	プロパティタイプ	プロパティ
AWS::OpsWorks::App	ソース	Password
AWS::OpsWorks::Stack	CustomCookbooksSource	Password
AWS::OpsWorks::Stack	RdsDbInstances	DbPassword
AWS::RDS::DBCluster		MasterUserPassword
AWS::RDS::DBInstance		MasterUserPassword
AWS::Redshift::Cluster		MasterUserPassword

リファレンスパターン

CloudFormation テンプレートで Systems Manager Parameter Store の Secure String 値を参照するには、次の ssm-secure 参照パターンを使用します。

```
{{resolve:ssm-secure:parameter-name:version}}
```

参照は、parameter-name と version について、以下の正規表現パターンに従う必要があります。

```
{{resolve:ssm-secure:[a-zA-Z0-9_.\-/]+(:\d+)??}}
```

parameter-name

パラメータストア内のパラメータの名前。パラメータ名では大文字と小文字が区別されます
必須。

version

使用するパラメータのバージョンを指定する整数。正確なバージョンを指定しない場合、CloudFormation は、スタックを作成または更新するたびに最新バージョンのパラメータを使用します。詳細については、「AWS Systems Manager ユーザーガイド」の「[パラメータバージョンの使用](#)」を参照してください。
オプション。

例

次の例では、`ssm-secure` の動的参照を使用して、IAM ユーザーのパスワードを Parameter Store に格納されている `SecureString` に設定します。指定されているとおり、CloudFormation はスタック操作および変更セット操作に `IAMUserPassword` パラメータのバージョン `10` を使用します。

JSON

```
"MyIAMUser": {
  "Type": "AWS::IAM::User",
  "Properties": {
    "UserName": "MyUserName",
    "LoginProfile": {
      "Password": "{{resolve:ssm-secure:IAMUserPassword:10}}"
    }
  }
}
```

YAML

```
MyIAMUser:
  Type: AWS::IAM::User
  Properties:
    UserName: 'MyUserName'
    LoginProfile:
      Password: '{{resolve:ssm-secure:IAMUserPassword:10}}'
```

Secrets Manager からシークレットまたはシークレット値を取得する

Secrets Manager は、データベース認証情報、パスワード、サードパーティー API キーなどのシークレットを安全に保存および管理できるようにするサービスです。Secrets Manager を使用すると、これらのシークレットへのアクセスを一元的に保存および制御できるため、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API 呼び出しに置き換えて、プログラムでシークレットを取得できます。詳細については、「[AWS Secrets Manager ユーザーガイド](#)」の「[What is AWS Secrets Manager?](#)」を参照してください。

CloudFormation テンプレート内の Secrets Manager に保存されているシークレット全体またはシークレット値全体を使用するには、`secretsmanager` 動的参照を使用します。

ベストプラクティス

CloudFormation テンプレートで Secrets Manager の動的参照を使用する場合は、次のベストプラクティスに従ってください。

- CloudFormation テンプレートにバージョンレス参照を使用する – Secrets Manager に認証情報を保存し、適切なシークレットローテーションワークフローをサポートするために version-stage または version-id パラメータを指定せずに動的参照を使用します。
- 自動ローテーションの活用 – Secrets Manager の自動ローテーション機能を使用して、認証情報管理のためのバージョンレスの動的参照を行います。これにより、テンプレートを変更することなく、認証情報が定期的に更新されます。詳細については、「[AWS Secrets Manager シークレットのローテーション](#)」を参照してください。
- バージョン付き参照の使用は控えめにする – テストやロールバックなどの特定のシナリオでは、明示的な version-stage または version-id パラメータのみを指定します。

考慮事項

secretsmanager 動的参照を使用する場合は、次のような重要な考慮事項に留意する必要があります。

- CloudFormation は、以前のデプロイで使用されたシークレットのバージョンを追跡しません。動的な参照を実装する前に、シークレット管理戦略を慎重に計画してください。可能な場合はバージョンレス参照を使用して、自動シークレットローテーションを活用します。バージョンがない動的参照からバージョン付きの動的参照に移行する場合など、動的参照設定に変更を加えるときは、リソースの更新をモニタリングして検証します。
- Secrets Manager のシークレット値のみを更新しても、CloudFormation は自動的に新しい値を取得しません。CloudFormation は、リソースの作成時または動的参照を含むリソースを変更する更新時にのみシークレット値を取得します。

例えば、テンプレートに、MasterPassword プロパティが Secrets Manager 動的参照に設定されている [AWS::RDS::DBInstance](#) リソースが含まれているとします。このテンプレートからスタックを作成したら、Secrets Manager でシークレットの値を更新します。ただし、MasterPassword プロパティは古いパスワード値を保持します。

新しいシークレット値を適用するには、CloudFormation テンプレートで `AWS::RDS::DBInstance` リソースを変更し、スタックの更新を実行する必要があります。

今後この手動プロセスを回避するには、Secrets Manager を使用してシークレットを自動的にローテーションすることを検討してください。

- secretsmanager などの安全な値の動的参照は、現在カスタムリソースではサポートされていません。
- secretsmanager の動的な参照は、すべてのリソースプロパティで使用できます。secretsmanager の動的な参照を使用することは、Secrets Manager ログも CloudFormation ログも解決済みのシークレットの値を保持してはならないことを示します。ただし、シークレット値は、それが使用されているリソースを持つサービスに表示されることがあります。シークレットデータが漏れるのを防ぐために、使用方法を確認します。

アクセス許可

Secrets Manager に保存されているシークレットを指定するには、シークレットに対して [GetSecretValue](#) を呼び出すためのアクセス許可が必要です。

リファレンスパターン

CloudFormation テンプレートで Secrets Manager シークレットを参照するには、次の secretsmanager 参照パターンを使用します。

```
{{resolve:secretsmanager:secret-id:secret-string:json-key:version-stage:version-id}}
```

secret-id

シークレット名またはシークレット ARN。

AWS アカウントのシークレットにアクセスするには、シークレット名を指定するだけです。別の AWS アカウントのシークレットにアクセスするには、そのシークレットの完全な ARN を指定します。

必須。

secret-string

SecretString はサポートされる唯一の値です。デフォルトは SecretString です。

json-key

値を取得するペアのキー名を指定します。json-key を指定しない場合、CloudFormation はシークレットテキスト全体を取得します。

このセグメントにはコロン文字 (:) を含めることはできません。

version-stage

使用するシークレットのバージョンのステージングラベル。シークレットマネージャーは、ステージングラベルがローテーション処理中にさまざまなバージョンを追跡するために使用されます。version-stage を使用する場合は、version-id を指定することはできません。version-stage または version-id、を指定しない場合、デフォルトでは AWSCURRENT というラベルの付いたバージョンが取得されます。

このセグメントにはコロン文字 (:) を含めることはできません。

version-id

使用したいシークレットのバージョンの固有識別子を指定します。version-id を指定した場合は、version-stage を指定しないでください。version-stage または version-id を指定しない場合、次にデフォルトでは AWSCURRENT というバージョンが取得されます。

このセグメントにはコロン文字 (:) を含めることはできません。

例

トピック

- [シークレットからユーザー名とパスワードの値を取得する](#)
- [SecretString 全体を取得する](#)
- [シークレットの特定のバージョンから値を取得する](#)
- [別の AWS アカウントからシークレットを取得する](#)

シークレットからユーザー名とパスワードの値を取得する

次の [AWS::RDS::DBInstance](#) の例では、*MySecret* に保存されているユーザー名とパスワードの値を取得します。この例では、バージョンレスの動的参照に推奨されるパターンを示します。このパターンは、AWSCURRENT バージョンを自動的に使用し、テンプレートの変更を必要とせずに Secrets Manager のローテーションワークフローをサポートします。

JSON

```
{
  "MyRDSInstance": {
```

```

    "Type": "AWS::RDS::DBInstance",
    "Properties": {
        "DBName": "MyRDSInstance",
        "AllocatedStorage": "20",
        "DBInstanceClass": "db.t2.micro",
        "Engine": "mysql",
        "MasterUsername":
"{{resolve:secretsmanager:MySecret:SecretString:username}}",
        "MasterUserPassword":
"{{resolve:secretsmanager:MySecret:SecretString:password}}"
    }
}

```

YAML

```

MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: MyRDSInstance
    AllocatedStorage: '20'
    DBInstanceClass: db.t2.micro
    Engine: mysql
    MasterUsername: '{{resolve:secretsmanager:MySecret:SecretString:username}}'
    MasterUserPassword: '{{resolve:secretsmanager:MySecret:SecretString:password}}'

```

SecretString 全体を取得する

次の動的リファレンスは、*MySecret* の SecretString を取得します。

```
{{resolve:secretsmanager:MySecret}}
```

または:

```
{{resolve:secretsmanager:MySecret:::}}
```

シークレットの特定のバージョンから値を取得する

次の動的リファレンスは、*MySecret* の *AWSPREVIOUS* バージョンの *password* 値を取得します。

```
{{resolve:secretsmanager:MySecret:SecretString:password:AWSPREVIOUS}}
```

別の AWS アカウントからシークレットを取得する

次の動的リファレンスは、別の AWS アカウントにある *MySecret* の `SecretString` を取得します。別の AWS アカウントのシークレットにアクセスするには、完全なシークレット ARN を指定する必要があります。

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret}}
```

次の動的リファレンスは、別の AWS アカウントにある *MySecret* の *password* 値を取得します。別の AWS アカウントのシークレットにアクセスするには、完全なシークレット ARN を指定する必要があります。

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret:SecretString:password}}
```

擬似パラメータを使用して AWS 値を取得する

擬似パラメータは、アカウント ID、リージョン名、スタックの詳細など、デプロイや環境間で変更される可能性のある重要な AWS 環境情報へのアクセスを提供する組み込み変数です。

ハードコードされた値の代わりに擬似パラメータを使用すると、テンプレートの移植性が向上し、さまざまな AWS アカウント やリージョン間での再利用が容易になります。

構文

組み込み関数 `Ref` または `Fn::Sub` を使用して、擬似パラメータを参照できます。

参照番号

`Ref` 組み込み関数は、次の一般的な構文を使用します。詳細については、「[Ref](#)」を参照してください。

JSON

```
{ "Ref" : "AWS::PseudoParameter" }
```

YAML

```
!Ref AWS::PseudoParameter
```

Fn::Sub

Fn::Sub 組み込み関数は、擬似パラメータの周りに `${}` 構文を含む別の形式を使用します。詳細については、「[Fn::Sub](#)」を参照してください。

JSON

```
{ "Fn::Sub" : "${AWS::PseudoParameter}" }
```

YAML

```
!Sub '${AWS::PseudoParameter}'
```

使用可能な擬似パラメータ

AWS::AccountId

123456789012 など、スタックが作成されているアカウントの AWS アカウント ID を返します。

この擬似パラメータは、アカウント固有の ARN を含む IAM ロール、ポリシー、およびその他のリソースポリシーを定義するときに一般的に使用されます。

AWS::NotificationARNs

スタックイベント通知を受信する Amazon SNS トピックの Amazon リソースネーム (ARN) のリストを返します。これらの ARN は、スタックの作成または更新時に、AWS CLI の `--notification-arns` オプションまたはコンソールを使用して指定できます。

単一の値を返す他の擬似パラメータとは異なり、AWS::NotificationARNs は ARN のリストを返します。リスト内の特定の ARN にアクセスするには、Fn::Select 組み込み関数を使用します。詳細については、「[Fn::Select](#)」を参照してください。

AWS::NoValue

Fn::If 組み込み関数の戻り値として指定すると、対応するリソースプロパティを削除します。詳細については、「[Fn::If](#)」を参照してください。

この擬似パラメータは、特定の条件下でのみ含めるべき条件付きリソースプロパティを作成する場合に特に便利です。

AWS::Partition

リソースがあるパーティションを返します。標準 AWS リージョン の場合、パーティションは `aws` です。他のパーティションのリソースの場合、パーティションは `aws-partitionname` です。たとえば、中国 (北京および寧夏) リージョンにあるリソースのパーティションは `aws-cn` で、AWS GovCloud (米国西部) リージョンにあるリソースのパーティションは `aws-us-gov` です。

パーティションは、リソースの ARN の一部を形成します。AWS::Partition を使用することで、テンプレートがさまざまな AWS パーティション間で正しく動作します。

AWS::Region

`us-west-2` など、包括的なリソースが作成されているリージョンを表す文字列を返します。

これは、テンプレートが変更なしでさまざまな AWS リージョン に適応できるため、最も一般的に使用される擬似パラメータの 1 つです。

AWS::StackId

スタックの ID (ARN) を返します (例: `arn:aws:cloudformation:us-west-2:123456789012:stack/teststack/51af3dc0-da77-11e4-872e-1234567db123`)。

AWS::StackName

スタックの名前を返します (例: `teststack`)。

スタック名は、一般的に、特定のスタックに属するものとして簡単に識別できる一意のリソース名を作成するために使用されます。

AWS::URLSuffix

スタックがデプロイされている AWS リージョン の AWS ドメインのサフィックスを返します。サフィックスは通常 `amazonaws.com` ですが、中国 (北京) リージョンの場合、サフィックスは `amazonaws.com.cn` です。

このパラメータは、AWS サービスエンドポイントの URL を作成するときに特に便利です。

例

トピック

- [基本的な使用法](#)
- [AWS::NotificationARNsの使用](#)

- [AWS::NoValue を使用した条件付きプロパティ](#)

基本的な使用法

次の例では、Amazon SNS トピックと、そのトピックに通知を送信する CloudWatch アラームの 2 つのリソースを作成します。これらは `AWS::StackName`、`AWS::Region`、および `AWS::AccountId` を使用して、スタック名、現在の AWS リージョン、およびアカウント ID をリソース名、説明、および ARN に動的に挿入します。

JSON

```
{
  "Resources": {
    "MyNotificationTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "DisplayName": { "Fn::Sub": "Notifications for ${AWS::StackName}" }
      }
    },
    "CPUALarm": {
      "Type": "AWS::CloudWatch::Alarm",
      "Properties": {
        "AlarmDescription": { "Fn::Sub": "Alarm for high CPU in  
${AWS::Region}" },
        "AlarmName": { "Fn::Sub": "${AWS::StackName}-HighCPUALarm" },
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/EC2",
        "Statistic": "Average",
        "Period": 300,
        "EvaluationPeriods": 1,
        "Threshold": 80,
        "ComparisonOperator": "GreaterThanThreshold",
        "AlarmActions": [{ "Fn::Sub": "arn:aws:sns:${AWS::Region}:  
${AWS::AccountId}:${MyNotificationTopic}" }]
      }
    }
  }
}
```

YAML

```
Resources:
```

```

MyNotificationTopic:
  Type: AWS::SNS::Topic
  Properties:
    DisplayName: !Sub Notifications for ${AWS::StackName}
CPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: !Sub Alarm for high CPU in ${AWS::Region}
    AlarmName: !Sub ${AWS::StackName}-HighCPUAlarm
    MetricName: CPUUtilization
    Namespace: AWS/EC2
    Statistic: Average
    Period: 300
    EvaluationPeriods: 1
    Threshold: 80
    ComparisonOperator: GreaterThanThreshold
    AlarmActions:
      - !Sub arn:aws:sns:${AWS::Region}:${AWS::AccountId}:${MyNotificationTopic}

```

AWS::NotificationARNsの使用

次の例では、インスタンス起動イベントと起動エラーの通知を送信するように Auto Scaling グループを設定します。この設定では、AWS::NotificationARNs 擬似パラメータを使用します。これは、スタックの作成時に指定された Amazon SNS トピック ARN のリストを提供します。Fn::Select 関数はそのリストから最初の ARN を選択します。

JSON

```

"myASG": {
  "Type": "AWS::AutoScaling::AutoScalingGroup",
  "Properties": {
    "LaunchTemplate": {
      "LaunchTemplateId": { "Ref": "myLaunchTemplate" },
      "Version": { "Fn::GetAtt": [ "myLaunchTemplate", "LatestVersionNumber" ] }
    },
    "MaxSize": "1",
    "MinSize": "1",
    "VPCZoneIdentifier": [
      "subnetIdAz1",
      "subnetIdAz2",
      "subnetIdAz3"
    ],
    "NotificationConfigurations" : [{

```

```

    "TopicARN" : { "Fn::Select" : [ "0", { "Ref" : "AWS::NotificationARNs" } ] },
    "NotificationTypes" : [ "autoscaling:EC2_INSTANCE_LAUNCH",
"autoscaling:EC2_INSTANCE_LAUNCH_ERROR" ]
  }]
}
}

```

YAML

```

myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MinSize: '1'
    MaxSize: '1'
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    NotificationConfigurations:
      - TopicARN:
          Fn::Select:
            - '0'
            - Ref: AWS::NotificationARNs
        NotificationTypes:
          - autoscaling:EC2_INSTANCE_LAUNCH
          - autoscaling:EC2_INSTANCE_LAUNCH_ERROR

```

AWS::NoValue を使用した条件付きプロパティ

次の例では、スナップショット ID が指定されている場合にのみスナップショットを使用する Amazon RDS DB インスタンスを作成します。UseDBSnapshot 条件が true に評価された場合は、CloudFormation は DBSnapshotIdentifier プロパティに DBSnapshotName パラメータ値を使用します。条件が false に評価された場合は、CloudFormation は DBSnapshotIdentifier プロパティを削除します。

JSON

```

"MyDB" : {

```



```

    "Type" : "AWS::RDS::DBInstance",
    "Properties" : {
        "AllocatedStorage" : "5",
        "DBInstanceClass" : "db.t2.small",
        "Engine" : "MySQL",
        "EngineVersion" : "5.5",
        "MasterUsername" : { "Ref" : "DBUser" },
        "MasterUserPassword" : { "Ref" : "DBPassword" },
        "DBParameterGroupName" : { "Ref" : "MyRDSParamGroup" },
        "DBSnapshotIdentifier" : {
            "Fn::If" : [
                "UseDBSnapshot",
                {"Ref" : "DBSnapshotName"},
                {"Ref" : "AWS::NoValue"}
            ]
        }
    }
}

```

YAML

```

MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    EngineVersion: '5.5'
    MasterUsername:
      Ref: DBUser
    MasterUserPassword:
      Ref: DBPassword
    DBParameterGroupName:
      Ref: MyRDSParamGroup
    DBSnapshotIdentifier:
      Fn::If:
        - UseDBSnapshot
        - Ref: DBSnapshotName
        - Ref: AWS::NoValue

```

デプロイされた CloudFormation スタックからエクスポートされた出力を取得する

同じ AWS アカウントおよびリージョン内に複数のスタックがあるときは、スタック間での情報の共有が役に立つ場合があります。これは、あるスタックが別のスタックによって作成されたリソースを使用する必要があるときに便利です。

例えば、ウェブサーバーのためにサブネットやセキュリティグループなどのネットワークリソースを作成するスタックがあるとします。情報が共有されていると、実際のウェブサーバーを作成する他のスタックが、最初のスタックによって作成されたネットワークリソースを使用できます。リソースの ID をスタックのテンプレートにハードコード化する、または、ID を入力パラメータとして渡す必要はありません。

スタック間で情報を共有するには、1 つのスタックからの出力値をエクスポートし、それらを別のスタックにインポートします。処理の流れ

1. 最初のスタックのテンプレート (ネットワークスタックなど) で、Outputs セクションの Export フィールドを使用して、エクスポートする特定の値を定義します。詳細については、「[CloudFormation テンプレートの Outputs 構文](#)」を参照してください。
2. そのスタックを作成または更新すると、CloudFormation が出力値をエクスポートして、同じ AWS アカウントおよびリージョン内にある他のスタックがそれらを利用できるようにします。
3. 他のスタックのテンプレートで [Fn::ImportValue](#) 関数を使用して、最初のスタックからエクスポートされた値をインポートします。
4. 2 番目のスタック (ウェブサーバースタックなど) を作成または更新すると、CloudFormation が最初のスタックからエクスポートされた値を自動的に取得して、それらを使用します。

チュートリアル、およびサンプルテンプレートについては、[別の CloudFormation スタックのリソース出力を参照する](#) を参照してください。

スタック出力値のエクスポートとネストされたスタックの使用との比較

ネストされたスタックとは、別のスタック内で `AWS::CloudFormation::Stack` リソースを使用して作成するスタックです。ネストされたスタックにより、1 つのスタックからすべてのリソースをデプロイし、管理します。ネストされたスタックグループからの出力をグループ内の別のスタックへの入力として使用できます。これは値のエクスポートとは異なります。

情報共有をネストされたスタックグループ内に隔離する場合は、ネストされたスタックを使用することを推奨します。ネストされたスタックのグループ内に限らず、他のスタックと情報を共有するには、値をエクスポートします。たとえば、サブネットのあると 1 つのスタックを作成し、ID をエクスポートできます。他のスタックは、そのサブネットの ID をインポートすることで、そのサブネットを使用できます。各スタックは独自のサブネットを作成する必要はありません。スタックがサブネット ID をインポートする限り、それを変更または削除することはできません。

ネストされたスタックの詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。

考慮事項

クロススタック参照には以下の制限があります。

- AWS アカウント ごとに、Export 名がリージョン内で一意である必要があります。
- リージョン間でクロススタック参照は作成できません。組み込み関数 `Fn::ImportValue` を使用すると、同じリージョン内にエクスポートされた値のみをインポートできます。
- 出力の場合、Export の Name プロパティの値は、リソースに依存する Ref または GetAtt の関数を使用できません。

同様に、ImportValue 関数にリソースに依存する Ref または GetAtt 関数を含めることはできません。

- 別のスタックが出力値をインポート後、出力値をエクスポートしているスタックを削除したり、エクスポートされた出力値を変更することはできません。エクスポートしているスタックを削除したり出力値を変更する前に、インポートをすべて削除する必要があります。

エクスポートされた出力値のリスト

スタックからエクスポートされた出力値を表示する必要がある場合は、次のいずれかの方法を使用します。

エクスポートされた出力値 (コンソール) をリスト表示するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. 左側のナビゲーションペインで、[エクスポート] を選択します。

エクスポートされた出力値 (AWS CLI) をリスト表示するには

次の [list-exports](#) コマンドを使用します。 *us-east-1* を、ご利用の AWS リージョン に置き換えてください。

```
aws cloudformation list-exports --region us-east-1
```

以下は出力の例です。

```
{
  "Exports": [
    {
      "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-subnet-a",
      "Value": "subnet-07b410xmplddcfa03"
    },
    {
      "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-subnet-b",
      "Value": "subnet-075ed3xmplebd2fb1"
    },
    {
      "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-vpcid",
      "Value": "vpc-011d7xmpl100e9841"
    }
  ]
}
```

CloudFormation は、現在のリージョンのエクスポートされた出力の名前と値、およびエクスポート元のスタックを表示します。エクスポートされた出力値を別のスタックのテンプレートで使用するために、エクスポート名と `Fn::ImportValue` 関数を使用してその値を参照できます。

エクスポートされた出力値をインポートするスタックのリスト

エクスポートされた出力値を削除または変更するには、まずどのスタックがそれらをインポートしようとしているのかを確認する必要があります。

エクスポートされた出力値をインポートするスタックを表示するには、次のいずれかの方法を使用します。

エクスポートされた出力値をインポートするスタックをリストするには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 左側のナビゲーションペインで、[エクスポート] を選択します。
3. 特定のエクスポート値をインポートするスタックを確認するには、そのエクスポート値の [Export Name] (エクスポート名) を選択します。CloudFormation により、エクスポートの詳細ページが表示され、値をインポートしているすべてのスタックがリスト表示されます。

エクスポートされた出力値をインポートするスタックをリストするには (AWS CLI)

[list-imports](#) コマンドを使用します。*us-east-1* を AWS リージョン に、*private-vpc-vpcid* をエクスポートされた出力値の名前に置き換えます。

```
aws cloudformation list-imports --region us-east-1 \  
--export-name private-vpc-vpcid
```

CloudFormation から、値をインポートするスタックのリストが返されます。

```
{  
  "Imports": [  
    "my-app-stack"  
  ]  
}
```

エクスポートされた特定の値をインポートするスタックがわかったら、それらのスタックを変更して、出力値を参照する `Fn::ImportValue` 関数を削除する必要があります。エクスポートされた出力値を削除または編集する前に、エクスポートされた出力値を参照するインポートをすべて削除する必要があります。

CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する

テンプレートを作成するとき、CloudFormation が提供する特殊なパラメータタイプを使用して、既存の AWS リソースまたは Systems Manager パラメータの識別子の入力をユーザーに求めるパラメータを作成できます。

トピック

- [概要:](#)
- [例](#)
- [考慮事項](#)
- [サポートされている AWS 固有のパラメータタイプ](#)
- [サポートされている Systems Manager パラメータタイプ](#)
- [サポートされていない Systems Manager パラメータタイプ](#)

概要:

CloudFormation では、スタックの作成または更新中に入力値を指定して、パラメータを使用してスタックをカスタマイズできます。この機能は、テンプレートをさまざまなシナリオで再利用する柔軟性を実現します。

パラメータは、CloudFormation テンプレートの Parameters セクションで定義されます。各パラメータには名前とタイプがあり、デフォルト値や許可された値などの追加設定を持つことができます。詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

パラメータタイプは、パラメータが受け入れることができる入力値のタイプを決定します。例えば、Number は数値のみを受け入れ、String はテキスト入力を受け入れます。

CloudFormation には、テンプレートで既存の AWS リソースおよび Systems Manager パラメータを参照するために使用できる追加のパラメータタイプがいくつか用意されています。

これらのパラメータタイプは、次の 2 つのカテゴリに分類されます。

- **AWS 固有のパラメータタイプ** – CloudFormation は、スタックを作成または更新するときに無効な値をキャッチできるようにする一連のパラメータタイプを提供します。これらのパラメータタイプを使用すると、テンプレートを使用するすべてのユーザーは、スタックを作成する AWS アカウント およびリージョンから有効な値を指定する必要があります。

AWS マネジメントコンソールを使用する場合、CloudFormation はアカウントおよびリージョンから既存の値の事前入力されたリストを提供します。そのため、ユーザーは特定の名前や ID を記憶して正確に入力する必要がありません。代わりに、ドロップダウンリストから値の選択のみを行います。場合によっては、ID、名前、Name タグ値で値を検索することもできます。

- **Systems Manager パラメータタイプ** – CloudFormation は、Systems Manager Parameter Store の既存パラメータに対応するパラメータタイプも提供します。これらのパラメータタイプを使用する

と、テンプレートを使用するすべてのユーザーは、Systems Manager パラメータタイプの値として Parameter Store キーを指定する必要がある、CloudFormation はスタックで使用する最新の値を Parameter Store から取得します。新しい Amazon マシンイメージ (AMI) ID など、新しいプロパティ値を持つアプリケーションを頻繁に更新する必要があるときには便利です。パラメータストアについては、「[Systems Manager Parameter Store](#)」を参照してください。

Parameters セクションで定義したら、Ref 関数を使用して CloudFormation テンプレート全体のパラメータ値を参照することができます。

例

次の例では、次のパラメータタイプを使用するテンプレートが示されています。

- AWS::EC2::VPC::Id
- AWS::EC2::Subnet::Id
- AWS::EC2::KeyPair::KeyName
- AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

このテンプレートでスタックを作成するには、アカウントから既存の VPC ID、サブネット ID、キーペア名を指定する必要があります。目的の AMI ID を参照する既存の Parameter Store キーを指定するか、[/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2](#) のデフォルト値を維持することもできます。このパブリックパラメータは、最新の Amazon Linux 2 AMI のリージョン AMI ID 用エイリアスです。パブリックパラメータの詳細については、「AWS Systems Manager ユーザーガイド」の「[Parameter Store でパブリックパラメータの検索](#)」を参照してください。

JSON

```
{
  "Parameters": {
    "VpcId": {
      "Description": "ID of an existing Virtual Private Cloud (VPC).",
      "Type": "AWS::EC2::VPC::Id"
    },
    "PublicSubnetId": {
      "Description": "ID of an existing public subnet within the specified VPC.",
      "Type": "AWS::EC2::Subnet::Id"
    },
  },
}
```

```

    "KeyName": {
      "Description": "Name of an existing EC2 key pair to enable SSH access to the
instance.,
      "Type": "AWS::EC2::KeyPair::KeyName"
    },
    "AMIId": {
      "Description": "Name of a Parameter Store parameter that stores the ID of
the Amazon Machine Image (AMI).,
      "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>,
      "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
    }
  },
  "Resources": {
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Enable SSH access via port 22",
        "VpcId": { "Ref": "VpcId" },
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    },
    "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "KeyName": { "Ref": "KeyName" },
        "ImageId": { "Ref": "AMIId" },
        "NetworkInterfaces": [
          {
            "AssociatePublicIpAddress": "true",
            "DeviceIndex": "0",
            "SubnetId": { "Ref": "PublicSubnetId" },
            "GroupSet": [{ "Ref": "InstanceSecurityGroup" }]
          }
        ]
      }
    }
  }
},

```



```

    "Outputs": {
      "InstanceId": {
        "Value": { "Ref": "Ec2Instance" }
      }
    }
  }
}

```

YAML

```

Parameters:
  VpcId:
    Description: ID of an existing Virtual Private Cloud (VPC).
    Type: AWS::EC2::VPC::Id
  PublicSubnetId:
    Description: ID of an existing public subnet within the specified VPC.
    Type: AWS::EC2::Subnet::Id
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance.
    Type: AWS::EC2::KeyPair::KeyName
  AMIID:
    Description: Name of a Parameter Store parameter that stores the ID of the Amazon
    Machine Image (AMI).
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access via port 22
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      ImageId: !Ref AMIID
      NetworkInterfaces:
        - AssociatePublicIpAddress: "true"
          DeviceIndex: "0"

```

```
SubnetId: !Ref PublicSubnetId
GroupSet:
  - !Ref InstanceSecurityGroup

Outputs:
  InstanceId:
    Value: !Ref Ec2Instance
```

スタックを作成する AWS CLI コマンド

次の「[create-stack](#)」コマンドは、テンプレート例に基づいてスタックを作成します。

```
aws cloudformation create-stack --stack-name MyStack \
  --template-body file://sampletemplate.json \
  --parameters \
    ParameterKey="VpcId",ParameterValue="vpc-a123baa3" \
    ParameterKey="PublicSubnetId",ParameterValue="subnet-123a351e" \
    ParameterKey="KeyName",ParameterValue="MyKeyName" \
    ParameterKey="AMIId",ParameterValue="MyParameterKey"
```

List<AWS::EC2::Subnet::Id> などの文字列のリストを受け入れるパラメータタイプを使用するには、次の例で示されるように、ParameterValue 内のカンマを二重バックスラッシュでエスケープする必要があります。

```
--parameters ParameterKey="SubnetIDs",ParameterValue="subnet-5ea0c127\\, subnet-6194ea3b\\, subnet-c87f2be0"
```

考慮事項

動的参照を使用して、サードパーティーの認証情報などの機密性の高い設定定義へのアクセスを制限することを強くお勧めします。詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

テンプレートユーザーに異なる AWS アカウント から値を指定できるようにする場合、AWS 固有のパラメータタイプを使用しないでください。代わりに、String または CommaDelimitedList タイプのパラメータを定義します。

Systems Manager パラメータタイプには、いくつかの点に注意する必要があります。

- パラメータの解決された値はコンソールでスタックの [パラメータ] タブで確認するか、「[describe-stacks](#)」または「[describe-change-set](#)」を実行することによって確認できます。これ

らの値はスタックが作成または更新される時に設定されるため、Parameter Store の最新の値とは異なる場合があることに注意してください。

- スタックの更新には、[既存の値の使用] オプション (または UsePreviousValue を true に設定) を使用すると、同じ Parameter Store キーを使用し続けること (値ではなく) を意味します。CloudFormation は常に最新の値を取得します。
- 許可された値やその他の制約を指定する場合、CloudFormation は指定したパラメータキーに対して検証しますが、その値に対しては検証しません。値の検証は Parameter Store 自体で行う必要があります。
- スタックを作成または更新して変更セットを作成すると、CloudFormation はオペレーションの実行時点で Parameter Store に存在する値を使用します。指定されたパラメータが、呼び出し元の AWS アカウントの下のパラメータストアに存在しない場合、CloudFormation は検証エラーを返します。
- 変更セットを実行すると、CloudFormation は変更セットで指定されている値を使用します。これらの値は、変更セットの作成時と実行時の間にパラメータストアで変化する可能性があるため、変更セットを実行する前に確認してください。
- パラメータストアのパラメータが同じ AWS アカウントに保存されている場合は、そのパラメータ名を指定する必要があります。パラメータストアのパラメータが別の AWS アカウントによって共有されている場合は、完全パラメータ ARN を指定する必要があります。

サポートされている AWS 固有のパラメータタイプ

CloudFormation では次の AWS 固有のタイプをサポートしています。

`AWS::EC2::AvailabilityZone::Name`

アベイラビリティゾーン (us-west-2a など)。

`AWS::EC2::Image::Id`

Amazon EC2 イメージ ID (ami-0ff8a91507f77f867 など)。CloudFormation コンソールには、このパラメータタイプの値のドロップダウンリストが表示されないことに注意してください。

`AWS::EC2::Instance::Id`

Amazon EC2 インスタンス ID (i-1e731a32 など)。

`AWS::EC2::KeyPair::KeyName`

Amazon EC2 のキーペア名。

`AWS::EC2::SecurityGroup::GroupName`

デフォルトの VPC セキュリティグループ名 (my-sg-abc など)。

`AWS::EC2::SecurityGroup::Id`

セキュリティグループ ID (sg-a123fd85 など)。

`AWS::EC2::Subnet::Id`

サブネット ID (subnet-123a351e など)。

`AWS::EC2::Volume::Id`

Amazon EBS ボリューム ID (vol-3cdd3f56 など)。

`AWS::EC2::VPC::Id`

VPC ID (vpc-a123baa3 など)。

`AWS::Route53::HostedZone::Id`

Amazon Route 53 ホストゾーン ID (Z23YXV40VPL04A など)。

`List<AWS::EC2::AvailabilityZone::Name>`

リージョンのアベイラビリティゾーンの配列 (us-west-2a, us-west-2b など)。

`List<AWS::EC2::Image::Id>`

Amazon EC2 イメージ ID の配列 (ami-0ff8a91507f77f867, ami-0a584ac55a7631c0c など)。CloudFormation コンソールには、このパラメータタイプの値のドロップダウンリストが表示されないことに注意してください。

`List<AWS::EC2::Instance::Id>`

Amazon EC2 インスタンス ID の配列 (i-1e731a32, i-1e731a34 など)。

`List<AWS::EC2::SecurityGroup::GroupName>`

デフォルトの VPC セキュリティグループ名の配列 (my-sg-abc, my-sg-def など)。

`List<AWS::EC2::SecurityGroup::Id>`

セキュリティグループ ID の配列 (sg-a123fd85, sg-b456fd85 など)。

`List<AWS::EC2::Subnet::Id>`

サブネット ID の配列 (subnet-123a351e, subnet-456b351e など)。

List<AWS::EC2::Volume::Id>

Amazon EBS ボリューム ID の配列 (vol-3cdd3f56, vol-4cdd3f56 など)。

List<AWS::EC2::VPC::Id>

VPC ID の配列 (vpc-a123baa3, vpc-b456baa3 など)。

List<AWS::Route53::HostedZone::Id>

Amazon Route 53 ホストゾーン ID の配列 (Z23YXV40VPL04A, Z23YXV40VPL04B など)。

サポートされている Systems Manager パラメータタイプ

CloudFormation では、次の Systems Manager パラメータタイプをサポートしています。

AWS::SSM::Parameter::Name

Systems Manager パラメーターキーの名前。このパラメータタイプは、必要なパラメータが存在することを確認する場合にのみ使用します。CloudFormation は、パラメータに関連付けられている実際の値を取得しません。

AWS::SSM::Parameter::Value<String>

値が文字列の Systems Manager パラメーター。これは、パラメータストア内の String パラメータータイプに対応します。

AWS::SSM::Parameter::Value<List<String>>、または

AWS::SSM::Parameter::Value<CommaDelimitedList>

値が文字列のリストである Systems Manager パラメーター。これは、パラメータストア内の StringList パラメータータイプに対応します。

AWS::SSM::Parameter::Value<*AWS-specific parameter type*>

値が AWS 固有のパラメータタイプである Systems Manager パラメータ。

たとえば、次の例は AWS::EC2::KeyPair::KeyName タイプを指定します。

- AWS::SSM::Parameter::Value<AWS::EC2::KeyPair::KeyName>

AWS::SSM::Parameter::Value<List<*AWS-specific parameter type*>>

値が AWS 固有のパラメータタイプのリストである Systems Manager パラメータ。

たとえば、次の例は AWS::EC2::KeyPair::KeyName タイプのリストを指定します。

- AWS::SSM::Parameter::Value<List<AWS::EC2::KeyPair::KeyName>>

サポートされていない Systems Manager パラメータタイプ

CloudFormation では、次の Systems Manager パラメータタイプをサポートしていません。

- Systems Manager パラメータタイプのリストの例:

```
List<AWS::SSM::Parameter::Value<String>>
```

また、CloudFormation では、テンプレートパラメータを `SecureString` Systems Manager パラメータタイプとして定義できません。ただし、リソースによってはパラメータ値として `SecureString` を指定できます。詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

CloudFormation のチュートリアル

このドキュメントでは、スタックデプロイの実践的な練習を提供するように設計されたチュートリアルコレクションを提供します。

- [別の CloudFormation スタックのリソース出力を参照する](#) – ある CloudFormation スタックの出力を別のスタック内で参照する方法について説明します。1 つのスタックにすべてのリソースを含めるのではなく、必要な AWS リソースをいくつかのスタックに分けて作成して、モジュール性と再利用性を高めたテンプレートを作成できます。
- [Amazon EC2 にアプリケーションをデプロイする](#) – CloudFormation を使用して、Amazon EC2 インスタンスにアプリケーションを自動的にインストールし、設定して起動する方法について説明します。これにより、インスタンスに直接接続することなく、簡単にデプロイを複製し、既存のインストールを更新できます。
- [CloudFormation のスタックを更新する](#) – 実行中のスタックを CloudFormation を使って更新するシンプルな進行過程について説明します。
- [スケールおよび負荷分散されたアプリケーションを作成する](#) – CloudFormation を使用して、スケラブルで負荷分散対応のアプリケーションを作成する方法について説明します。具体的には、Auto Scaling グループ、ロードバランサー、その他の関連するリソースを作成して、アプリケーションがトラフィック負荷の変動に対応し、高可用性を維持できるようにする方法を取り上げます。
- [別の AWS アカウントで VPC とピア接続する](#) – 異なる AWS アカウントで 2 つの VPC 間に仮想プライベートクラウド (VPC) ピアリング接続を確立するプロセスについて説明します。VPC 間のトラフィックをルーティングして、同じネットワークの一部であるかのようにリソースにアクセスできるようにする場合には、VPC ピアリングが便利です。

- [CloudFormation を使用して CodeDeploy を通じて ECS ブルー/グリーンデプロイを実行する](#) – CloudFormation を使用して Amazon ECS で AWS CodeDeploy ブルー/グリーンデプロイを実行する方法について説明します。ブルー/グリーンデプロイは、最小限のダウンタイムでアプリケーションまたはサービスを更新する方法です。

別の CloudFormation スタックのリソース出力を参照する

このチュートリアルでは、ある CloudFormation スタックの出力を別のスタック内で参照して、モジュール性と再利用性を高めたテンプレートを作成する方法について説明します。

1 つのスタックにすべてのリソースを含めるのではなく、必要な AWS リソースをいくつかのスタックに分けて作成します。その後、他のスタックから必要なリソース出力を参照できます。クロススタックの参照を出力に制限することにより、他のスタックから参照されるスタックの部分を制御します。

たとえば、VPC のあるネットワークスタック、セキュリティグループ、パブリックウェブアプリケーションのサブネット、別個のパブリックウェブアプリケーションスタックがあるとします。ウェブアプリケーションで、ネットワークスタックのセキュリティグループとサブネットが使用されるようにするには、ウェブアプリケーションスタックがネットワークからのリソース出力を参照できるようにクロススタック参照を作成する必要があります。クロススタック参照を使用すると、ウェブアプリケーションスタックの所有者がネットワーキングのルールや資産を作成または維持する必要がなくなります。

クロススタックの参照を作成するには、Export 出力フィールドを使用してエクスポートのリソース出力の値を設定します。次に、値を `Fn::ImportValue` 組み込み関数を使用して値をインポートします。詳細については、「[デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)」を参照してください。

Note

CloudFormation は無料サービスです。ただし、スタックに追加する AWS リソースにはそれぞれ現在の料金が課金されます。AWS の料金の詳細については、[各製品の詳細ページ](#)を参照してください。

トピック

- [サンプルテンプレートを使用してネットワークスタックを作成する](#)
- [サンプルテンプレートを使用してウェブアプリケーションスタックを作成する](#)

- [スタックが設計どおりに機能することを確認する](#)
- [AMI マッピングエラーのトラブルシューティング](#)
- [リソースのクリーンアップ](#)

サンプルテンプレートを使用してネットワークスタックを作成する

このチュートリアルを開始する前に、IAM アクセス権限があり、Amazon VPC、Amazon EC2、CloudFormation のいずれのサービスも使用できることを確認してください。

ネットワークスタックは、ウェブアプリケーションスタックで使用する VPC、セキュリティグループ、サブネットで構成されます。これらのリソースに加えて、パブリックアクセスを可能にするインターネットゲートウェイとルーティングテーブルも、ネットワークスタックによって作成されます。

ウェブアプリケーションスタックの作成前に、このスタックを作成する必要があります。先にウェブアプリケーションスタックを作成した場合は、使用できるセキュリティグループやサブネットがない状態になります。

スタックテンプレートは、次の URL から入手できます: <https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template>。スタックによって作成されるリソースを表示するには、リンクを選択します。これにより、テンプレートが開きます。Outputs セクションでは、サンプルのテンプレートがエクスポートするネットワークングリソースを表示できます。他のスタックからネットワークングリソースをエクスポートした場合、エクスポートされたリソース名には、スタック名がプレフィックスとして付けられます。ユーザーがネットワークングリソースをインポートするときは、リソースをどのスタックからインポートするかを指定できます。

ネットワークスタックを作成するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。
3. [テンプレートの準備完了] を選択し、[テンプレートを指定] セクションで [Amazon S3 URL] を選択します。
4. Amazon S3 URL の場合は、次の URL を貼り付けます: **`https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template`**。

5. [次へ] を選択します。
6. [スタックの名前] に **SampleNetworkCrossStack** を入力し、[次へ] を選択します。

 Note

このスタックの名前をメモしておきます。ウェブアプリケーションスタックを起動するときにこのスタック名が必要になります。

7. [次へ] を選択します。このチュートリアルでは、タグの追加も詳細設定の指定も不要です。
8. スタック名とテンプレート URL が正しいことを確認し、[スタックの作成] を選択します。

CloudFormation によってスタックが作成されるまでに数分かかることもあります。ウェブアプリケーションスタックの作成に進む前に、すべてのリソースが正しく作成されるまで待ちます。

9. 進捗状況を監視するには、スタックイベントを確認します。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

サンプルテンプレートを使用してウェブアプリケーションスタックを作成する

ウェブアプリケーションスタックによって、ネットワークスタックのセキュリティグループとサブネットを使用する EC2 インスタンスが作成されます。

このスタックは、ネットワークスタックと同じ AWS リージョン に作成する必要があります。

スタックテンプレートは、次の URL から入手できます: <https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template>。スタックによって作成されるリソースを表示するには、リンクを選択します。これにより、テンプレートが開きます。Resources セクションで、EC2 インスタンスのプロパティを参照してください。Fn::ImportValue 関数を使用することで、別のスタックからどのようにネットワーキングリソースがインポートされたかを確認できます。

ウェブアプリケーションスタックを作成するには

1. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。
2. [テンプレートの準備完了] を選択し、[テンプレートを指定] セクションで [Amazon S3 URL] を選択します。

3. Amazon S3 URL の場合は、次の URL を貼り付けます: **`https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template`**。
4. [次へ] を選択します。
5. [スタックの名前] に **SampleWebAppCrossStack** を入力します。[パラメータ] セクションで、[NetworkStackName] パラメータのデフォルト値を使用し、[次へ] を選択します。

サンプルテンプレートでは、パラメータ値を使用してどのスタックから値をインポートするかを指定しています。

6. [次へ] を選択します。このチュートリアルでは、タグの追加も詳細設定の指定も不要です。
7. スタック名とテンプレート URL が正しいことを確認し、[スタックの作成] を選択します。

CloudFormation によってスタックが作成されるまでに数分かかることもあります。

スタックが設計どおりに機能することを確認する

スタックの作成後、そのリソースを表示し、インスタンス ID をメモしておきます。スタックリソースの表示の詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

インスタンスのセキュリティグループとサブネットを確認するには、[Amazon EC2 コンソール](#)でインスタンスのプロパティを確認します。インスタンスが SampleNetworkCrossStack スタックからセキュリティグループとサブネットを使用しているなら、クロススタックの参照が正常に作成されています。

コンソールを使用してスタック出力と例のウェブサイト URL を表示し、ウェブアプリケーションが実行中であることを確認します。詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

AMI マッピングエラーのトラブルシューティング

エラー Template error: Unable to get mapping for AWSRegionArch2AMI::[region]::HVM64 が表示された場合、テンプレートには AWS リージョンの AMI マッピングが含まれていません。マッピングを更新する代わりに、Systems Manager の公開パラメータを使用して最新の AMI を動的に参照することをお勧めします。

1. SampleWebAppCrossStack テンプレートを <https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template> からローカルマシンにダウンロードします。
2. AWSRegionArch2AMI マッピングセクション全体を削除します。
3. 次の Systems Manager パラメータを追加します。

```
"LatestAmiId": {
  "Description": "The latest Amazon Linux 2 AMI from the Parameter Store",
  "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
  "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}
```

4. 既存の ImageId 参照を置き換えます。

```
"ImageId": { "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" } ,
  "HVM64" ] },
```

次と置き換えます。

```
"ImageId": { "Ref": "LatestAmiId" },
```

このパラメータは、スタックをデプロイするリージョンの最新の Amazon Linux 2 AMI に自動的に解決されます。

他の Linux ディストリビューションの場合は、適切なパラメータパスを使用します。詳細については、「AWS Systems Manager ユーザーガイド」の「[Parameter Store でのパブリックパラメータの検出](#)」を参照してください。

5. 変更したテンプレートをアカウントの S3 バケットにアップロードします。

```
aws s3 cp SampleWebAppCrossStack.template s3://amzn-s3-demo-bucket/
```

6. スタックを作成するときは、サンプル URL の代わりに S3 テンプレート URL を指定します。

リソースのクリーンアップ

不要なサービスに対して課金されないように、スタックを削除します。

スタックを削除するには

1. CloudFormation コンソールで、SampleWebAppCrossStack スタックを選択します。
2. [アクション] を選択してから、[スタックの削除] を選択します。
3. 確認メッセージで、[削除] を選択します。
4. このスタックの削除後、SampleNetworkCrossStack スタックに対して同じ手順を繰り返します。

Note

CloudFormation によって SampleWebAppCrossStack スタックが完全に削除されるまで待ちます。EC2 インスタンスが VPC でまだ実行中であれば、CloudFormation は SampleNetworkCrossStack スタックの VPC を削除しません。

Amazon EC2 にアプリケーションをデプロイする

CloudFormation を使用して、Amazon EC2 インスタンスにアプリケーションを自動的にインストールし、設定および起動することができます。これにより、インスタンスへ直接接続することなく、簡単にデプロイを複製し、既存のインストールを更新できるので、多くの時間と労力を節約することができます。

CloudFormation には、cloud-init に基づく一連のヘルパースクリプト (cfn-init、cfn-signal、cfn-get-metadata、および cfn-hup) が含まれています。同じテンプレートに存在する Amazon EC2 インスタンスにアプリケーションをインストールし、設定および更新するには、CloudFormation テンプレートからこれらのヘルパースクリプトを呼び出します 詳細については、「CloudFormation テンプレートリファレンスガイド」の「[CloudFormation ヘルパースクリプトリファレンス](#)」を参照してください。

[入門チュートリアル](#)では、基本的な bash スクリプトで UserData を使用してシンプルなウェブサーバーを作成しました。これは簡単な「Hello World」ページでは機能しますが、実際のアプリケーションでは次のようなより高度な設定が必要になることが多くあります。

- 正しい順序でインストールされた複数のソフトウェアパッケージ。
- 特定のコンテンツで作成された複雑な設定ファイル。
- 自動的に開始し、実行するよう設定されたサービス。
- セットアッププロセスでのエラー処理と検証。

CloudFormation のヘルパースクリプトは、UserData の基本的な bash スクリプトより堅牢かつメンテナンスしやすい方法で EC2 インスタンスを設定できます。cfn-init ヘルパースクリプトは、テンプレートのメタデータから設定データを読み取り、インスタンスに体系的に適用します。

このチュートリアルでは、cfn-init ヘルパースクリプトを使用してブートストラッププロセスをモニタリングする方法を学習します。

Note

CloudFormation は無料で使用できますが、作成した Amazon EC2 リソースに対しては課金されます。ただし、AWS を初めて使用する場合は、[無料利用枠](#)を利用して、この学習プロセス中のコストを最小に抑えるか無料にできます。

トピック

- [前提条件](#)
- [ブートストラップの概念について](#)
- [シンプルなブートストラップの例から始める](#)
- [ファイルとコマンドの追加](#)
- [ネットワークセキュリティの強化](#)
- [完全なブートストラップテンプレート](#)
- [コンソールを使用してスタックを作成する](#)
- [ブートストラッププロセスをモニタリングする](#)
- [ブートストラップされたウェブサーバーをテストする](#)
- [ブートストラップ問題のトラブルシューティング](#)
- [リソースをクリーンアップする](#)
- [次のステップ](#)

前提条件

- [最初のスタックの作成](#) チュートリアルを完了しているか、CloudFormation の基本と同等の経験を有している必要があります。
- Amazon EC2 および CloudFormation を使用するアクセス許可を持つ IAM ユーザーまたはロール、または管理ユーザーアクセスで AWS アカウント にアクセスする必要があります。

- インターネットにアクセスできる Virtual Private Cloud (VPC) が必要です。このチュートリアルテンプレートには、新しい AWS アカウント に自動的に付属するデフォルトの VPC が必要です。デフォルトの VPC がない場合、またはデフォルトの VPC が削除された場合は、[最初のスタックの作成](#) チュートリアルのトラブルシューティングセクションで代替ソリューションを参照してください。

ブートストラップの概念について

テンプレートを作成する前に、ブートストラップを機能させる主要な概念について理解しておきましょう。

cfn-init ヘルパースクリプト

CloudFormation には、Amazon EC2 インスタンスでソフトウェアをインストールしたりサービスを開始したりするために使用できる Python ヘルパースクリプトが用意されています。cfn-init スクリプトはテンプレートからリソースメタデータを読み取り、設定をインスタンスに適用します。

このプロセスは次のように機能します。

1. 設定は、EC2 リソースの「Metadata」セクションで定義します。
2. cfn-init を UserData スクリプトから呼び出します。
3. cfn-init でメタデータを読み取り、設定を適用します。
4. インスタンスが仕様に従って設定されます。

メタデータ構造

設定は EC2 インスタンス内の特定の構造で定義されます。

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:
      # Metadata section for the resource
      AWS::CloudFormation::Init:  # Required key that cfn-init looks for
        config:                  # Configuration name (you can have multiple)
          packages:               # Install packages
          files:                  # Create files
          commands:               # Run commands
          services:               # Start/stop services
```

cfn-init スクリプトは、パッケージ、グループ、ユーザー、ソース、ファイル、コマンド、そしてサービスという特定の順序でこれらのセクションを処理します。

シンプルなブートストラップの例から始める

まず、Apache をインストールして起動する最小限のブートストラップの例から始めましょう。

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:          # Install Apache web server
            yum:
              httpd: []
          services:          # Start Apache and enable it to start on boot
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      UserData: !Base64      # Script that runs when instance starts
      Fn::Sub: |
        #!/bin/bash
        yum install -y aws-cfn-bootstrap
        /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource EC2Instance --
        region ${AWS::Region}
```

この簡単な例で、主要な概念について説明します。

- 「packages」セクションでは、yum を使用して httpd パッケージをインストールします。これは、Amazon Linux および yum を使用する他の Linux ディストリビューションで機能します。
- 「services」セクションでは、httpd が自動的に開始および実行されます。
- UserData で最新のブートストラップツールをインストールし、cfn-init を呼び出します。

ファイルとコマンドの追加

次に、EC2 インスタンスの `/var/log` ディレクトリにカスタムウェブページとログファイルを追加して、例を強化しましょう。

ファイルを作成する

「files」セクションでは、特定のコンテンツを含むファイルをインスタンスに作成できます。垂直パイプ (`|`) を使用すると、テキストのリテラルブロック (HTML コード) をファイルの内容 (`/var/www/html/index.html`) として渡すことができます。

```
files:
  /var/www/html/index.html:
    content: |
      <body>
        <h1>Congratulations, you have successfully launched the AWS CloudFormation
sample.</h1>
      </body>
```

コマンドの実行

「commands」セクションでは、ブートストラッププロセス中にシェルコマンドを実行できます。このコマンドによって、EC2 インスタンスの `/var/log/welcome.txt` にログファイルが作成されます。これを表示するには、SSH アクセスに使用する Amazon EC2 キーペアと、インスタンスへの SSH に使用できる IP アドレス範囲が必要です (ここでは説明しません)。

```
commands:
  createWelcomeLog:
    command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
```

ネットワークセキュリティの強化

ウェブサーバーを設定するため、ウェブトラフィック (HTTP) が EC2 インスタンスに到達できるようにする必要があります。これを行うには、IP アドレスからのポート 80 で受信トラフィックを許可するセキュリティグループを作成します。また、EC2 インスタンスでは、パッケージの更新をインストールするなどの用途でインターネットにトラフィックを送信する必要があります。デフォルトで、セキュリティグループはすべてのアウトバウンドトラフィックを許可します。次に、SecurityGroupIds プロパティを使用して、このセキュリティグループを EC2 インスタンスに関連付けます。

```
WebServerSecurityGroup:
```



```
Type: AWS::EC2::SecurityGroup
Properties:
  GroupDescription: Allow HTTP access from my IP address
  SecurityGroupIngress:
    - IpProtocol: tcp
      Description: HTTP
      FromPort: 80
      ToPort: 80
      CidrIp: !Ref MyIP
```

完全なブートストラップテンプレート

次に、すべての要素をまとめてみましょう。ここで説明したすべての概念を組み合わせた、完全なテンプレートを以下に示します。

```
AWSTemplateFormatVersion: 2010-09-09
Description: Bootstrap an EC2 instance with Apache web server using cfn-init

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\\d{1,3}\\.){3}\\d{1,3}\\/(\\d{1,2})$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
```

```
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP access from my IP address
    SecurityGroupIngress:
      - IpProtocol: tcp
        Description: HTTP
        FromPort: 80
        ToPort: 80
        CidrIp: !Ref MyIP

WebServer:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
        files:
          /var/www/html/index.html:
            content: |
              <body>
                <h1>Congratulations, you have successfully launched the AWS
CloudFormation sample.</h1>
              </body>
        commands:
          createWelcomeLog:
            command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
        services:
          systemd:
            httpd:
              enabled: true
              ensureRunning: true
  Properties:
    ImageId: !Ref LatestAmiId
    InstanceType: !Ref InstanceType
    SecurityGroupIds:
      - !Ref WebServerSecurityGroup
    UserData: !Base64
      Fn::Sub: |
        #!/bin/bash
        yum install -y aws-cfn-bootstrap
```

```
    /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServer --
region ${AWS::Region}
    Tags:
      - Key: Name
        Value: Bootstrap Tutorial Web Server

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServer.PublicDnsName}'
    Description: EC2 instance public DNS name
```

コンソールを使用してスタックを作成する

次の手順では、ファイルからサンプルスタックテンプレートをアップロードします。ローカルマシンでテキストエディタを開き、テンプレートを追加します。次に、samplelinux2stack.template という名前でファイルを保存します。

スタックテンプレートを起動するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタックの作成] を選択し、[新しいリソースの使用 (標準)] を選択します。
3. [テンプレートを指定] で、[テンプレートファイルをアップロード]、[ファイルを選択] を選択して samplelinux2stack.template ファイルをアップロードします。
4. [次へ] を選択します。
5. 「スタックの詳細を指定する」ページで、スタック名に「**BootstrapTutorialStack**」と入力します。
6. [パラメータ] で、次の操作を行います。
 - LatestAmild: デフォルト値のままにします。
 - InstanceType: EC2 インスタンスタイプとして t2.micro または t3.micro を選択します。
 - MyIP: パブリック IP アドレスを /32 サフィックスを使用して入力します。
7. [次へ] を 2 回選択し、次に [送信] を選択してスタックを作成します。

ブートストラッププロセスをモニタリングする

追加のソフトウェアがインストールおよび設定されているため、ブートストラッププロセスは単純な EC2 の起動よりも時間がかかります。

ブートストラップの進行状況をモニタリングするには

1. CloudFormation コンソールでスタックを選択し、[イベント] タブを開きます。
2. WebServer CREATE_IN_PROGRESS イベントをモニタリングします。ブートストラッププロセスは、インスタンスの起動後に開始されます。
3. このプロセスには、通常数分かかります。完了時に「WebServer CREATE_COMPLETE」と表示されます。

ブートストラッププロセス中の状況を確認したい場合は、インスタンスログをチェックできます。

ブートストラップログを表示するには (オプション)

1. [EC2 コンソール](#)を開いてインスタンスを確認します。
2. インスタンスを選択してから、[アクション]、[モニタリングとトラブルシューティング]、[システムログの取得] を選択し、ブートストラップの進捗状況を確認します。
3. ログがすぐに表示されない場合は、待機してからページを更新してください。

ブートストラップされたウェブサーバーをテストする

スタックに「CREATE_COMPLETE」と表示されたら、ウェブサーバーをテストします。

ウェブサーバーをテストするには

1. CloudFormation コンソールで、スタックの [出力] タブに移動します。
2. WebsiteURL 値をクリックし、新しいタブでウェブサーバーを開きます。
3. カスタムウェブページに「Congratulations, you have successfully launched the AWS CloudFormation sample」というメッセージが表示されます。

Note

ページがすぐにロードされない場合は、1 分待ってからもう一度試してください。スタックに「CREATE_COMPLETE」と表示された後に、ブートストラッププロセスが完了する場合があります。

ブートストラップ問題のトラブルシューティング

ブートストラッププロセスが失敗した場合、またはウェブサーバーが動作しない場合の一般的な問題と解決策を以下に示します。

一般的な問題

- スタックを作成できない – [イベント] タブで特定のエラーメッセージを確認します。
- ウェブサーバーにアクセスできない – MyIP パラメータで IP アドレスが正しいことを確認します。末尾に必ず /32 を含めてください。
- ブートストラッププロセスが失敗する – インスタンスは起動しても cfn-init で失敗する場合があります。「モニタリング」セクションの説明に従って、システムログを確認します。

リソースをクリーンアップする

継続的な課金を回避するには、スタックとそのリソースを削除してクリーンアップできます。

スタックとそのリソースを削除するには

1. [クラウドフォーメーション コンソール](#)を開きます。
2. [スタック] ページで、作成したスタックの名前の横にあるオプション **[BootstrapTutorialStack]** を選択し、続いて [削除] を選択します。
3. 確認を求めるメッセージが表示されたら、[削除] を選択してください。
4. [イベント] タブでスタック削除プロセスの進行状況をモニタリングします。**BootstrapTutorialStack** のステータスが DELETE_IN_PROGRESS に変わります。CloudFormation がスタックの削除を完了すると、リストからスタックが削除されます。

次のステップ

お疲れ様でした。CloudFormation で EC2 インスタンスをブートストラップする方法を学習しました。これで、以下のことが理解できました。

- cfn-init ヘルパースクリプトの使用方法
- ブートストラップのメタデータを構築する方法
- パッケージのインストール、ファイルの作成、コマンドの実行、サービスの管理方法
- ブートストラップの問題をモニタリングする方法

学習を継続するには:

- 実行されているスタックを更新し、cfn-hup ヘルパースクリプトを使用する方法について説明します。詳細については、「[CloudFormation のスタックを更新する](#)」を参照してください。
- Windows スタックをブートストラップする方法について学習します。詳細については、「[Windows ベースの CloudFormation スタックのブートストラップ](#)」を参照してください。
- 複数の設定セットを使用して、より複雑なブートストラップシナリオを調べます。詳細については、「CloudFormation テンプレートリファレンスガイド」の「[cfn-init](#)」および「[AWS::CloudFormation::Init](#)」を参照してください。
- ブートストラップ完了ステータスを報告するための、cfn-signal について学習します。詳細については、「CloudFormation テンプレートリファレンスガイド」の「[cfn-signal](#)」を参照してください。

CloudFormation のスタックを更新する

Note

このチュートリアルは、[Amazon EC2 にアプリケーションをデプロイする](#) チュートリアルでの概念に基づいています。そのチュートリアルを完了していない場合は、まず CloudFormation による EC2 ブートストラップを理解することをお勧めします。

このトピックでは、実行中のスタックの更新のシンプルな進行過程を示します。以下の手順について説明します。

1. 初期スタックを作成する — 基本の Amazon Linux 2 AMI を使用してスタックを作成し、CloudFormation ヘルパースクリプトを使用して Apache Web Server とシンプルな PHP アプリケーションをインストールします。
2. アプリケーションを更新する - アプリケーション内のファイルのいずれかを更新し、CloudFormation を使用してソフトウェアをデプロイします。
3. キーペアを追加する - インスタンスに Amazon EC2 キーペアを追加し、セキュリティグループを更新して、インスタンスへの SSH アクセスを可能にします。
4. インスタンスタイプを追加する - 基盤となる Amazon EC2 インスタンスのインスタンスタイプを変更します。
5. AMI を更新する - スタック内の Amazon EC2 インスタンスの Amazon マシンイメージ (AMI) を変更します。

 Note

CloudFormation は無料で使用できますが、作成した Amazon EC2 リソースに対しては課金されます。ただし、AWS を初めて使用する場合は、[無料利用枠](#)を利用して、この学習プロセス中のコストを最小に抑えるか無料にできます。

トピック

- [ステップ 1: 初期スタックを作成する](#)
- [ステップ 2: アプリケーションを更新する](#)
- [ステップ 3: キーペアを使用して SSH アクセスを追加する](#)
- [ステップ 4: インスタンスタイプを更新する](#)
- [ステップ 5: AMI を更新する](#)
- [アベイラビリティと影響に関する検討事項](#)
- [関連リソース](#)

ステップ 1: 初期スタックを作成する

まず、このトピック全体で使用するスタックを作成することから開始します。Apache Web Server でホストされ Amazon Linux 2 AMI 上で実行される単一のインスタンス PHP ウェブアプリケーションを起動する単純なテンプレートが用意されています。

Apache Web Server、PHP、および単純な PHP アプリケーションはすべて、Amazon Linux 2 AMI にデフォルトでインストールされている CloudFormation ヘルパースクリプトによってインストールされます。以下のテンプレートスニペットには、インストールするパッケージとファイルを記述するメタデータが表示されています。この場合は、Amazon Linux 2 AMI の Yum リポジトリからの Apache Web Server と PHP 実行環境です。またこのスニペットには、Apache Web Server の実行を確認する Services セクションが含まれています。

```
WebServerInstance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
```

```
    php: []
  files:
    /var/www/html/index.php:
      content: |
        <?php
        echo '<h1>Hello World!</h1>';
        ?>
      mode: '000644'
      owner: apache
      group: apache
  services:
    systemd:
      httpd:
        enabled: true
        ensureRunning: true
```

アプリケーション自体は「Hello World」サンプルであり、その全体がテンプレート内で定義されています。実際のアプリケーションでは、ファイルは Amazon S3、GitHub、その他のリポジトリに格納され、テンプレートから参照されます。CloudFormation はパッケージ (RPM や RubyGems など) をダウンロードし、個別のファイルを参照して .zip や .tar ファイルを解凍することで、Amazon EC2 インスタンス上にアプリケーションの中間生成物を作成することができます。

テンプレートは cfn-hup デーモンを有効化して構成し、Amazon EC2 インスタンスのメタデータ内で定義された構成への変更をリッスンします。cfn-hup デーモンを使用することで、Apache や PHP のバージョンなどのアプリケーションソフトウェアを更新したり、または CloudFormation から PHP アプリケーションファイル自体を更新したりできます。テンプレート内の同じ Amazon EC2 リソースからの次のスニペットは、2 分ごとに cfn-init を呼び出してメタデータに更新を通知および適用するように cfn-hup を設定するために必要な部分を示しています。それ以外の場合、cfn-init は起動時に 1 回だけ実行されます。

```
files:
  /etc/cfn/cfn-hup.conf:
    content: !Sub |
      [main]
      stack=${AWS::StackId}
      region=${AWS::Region}
      # The interval used to check for changes to the resource metadata in minutes.
      Default is 15
      interval=2
      mode: '000400'
      owner: root
```



```
group: root
/etc/cfn/hooks.d/cfn-auto-reloader.conf:
  content: !Sub |
    [cfn-auto-reloader-hook]
    triggers=post.update
    path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
    action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --region
    ${AWS::Region}
    runas=root
services:
  systemd:
    cfn-hup:
      enabled: true
      ensureRunning: true
      files:
        - /etc/cfn/cfn-hup.conf
        - /etc/cfn/hooks.d/cfn-auto-reloader.conf
```

スタックを完了するには、Amazon EC2 インスタンス定義の Properties セクションで、UserData プロパティに、パッケージとファイルをインストールするために cfn-init を呼び出す cloud-init スクリプトが含まれています。詳細については、「CloudFormation テンプレートリファレンスガイド」の「[CloudFormation ヘルパースクリプトリファレンス](#)」を参照してください。テンプレートは Amazon EC2 セキュリティグループも作成します。

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Parameters:
```

```
  LatestAmiId:
```

```
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
```

```
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
```

```
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
```

```
  InstanceType:
```

```
    Description: WebServer EC2 instance type
```

```
    Type: String
```

```
    Default: t3.micro
```

```
    AllowedValues:
```

- t3.nano
- t3.micro
- t3.small
- t3.medium
- t3a.nano
- t3a.micro

- t3a.small
- t3a.medium
- m5.large
- m5.xlarge
- m5.2xlarge
- m5a.large
- m5a.xlarge
- m5a.2xlarge
- c5.large
- c5.xlarge
- c5.2xlarge
- r5.large
- r5.xlarge
- r5.2xlarge
- r5a.large
- r5a.xlarge
- r5a.2xlarge

ConstraintDescription: must be a valid EC2 instance type.

Resources:

WebServerInstance:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref LatestAmiId

InstanceType: !Ref InstanceType

SecurityGroupIds:

- !Ref WebServerSecurityGroup

UserData:

Fn::Base64: !Sub |

#!/bin/bash -xe

Get the latest CloudFormation package

yum update -y aws-cfn-bootstrap

Run cfn-init

/opt/aws/bin/cfn-init -v --stack \${AWS::StackName} --resource

WebServerInstance --region \${AWS::Region} || error_exit 'Failed to run cfn-init'

Start up the cfn-hup daemon to listen for changes to the EC2 instance

metadata

/opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'

Signal success or failure

/opt/aws/bin/cfn-signal -e \$? --stack \${AWS::StackName} --resource

WebServerInstance --region \${AWS::Region}

Metadata:

AWS::CloudFormation::Init:

```
config:
  packages:
    yum:
      httpd: []
      php: []
  files:
    /var/www/html/index.php:
      content: |
        <?php
        echo "<h1>Hello World!</h1>";
        ?>
      mode: '000644'
      owner: apache
      group: apache
    /etc/cfn/cfn-hup.conf:
      content: !Sub |
        [main]
        stack=${AWS::StackId}
        region=${AWS::Region}
        # The interval used to check for changes to the resource metadata in
minutes. Default is 15
        interval=2
      mode: '000400'
      owner: root
      group: root
    /etc/cfn/hooks.d/cfn-auto-reloader.conf:
      content: !Sub |
        [cfn-auto-reloader-hook]
        triggers=post.update
        path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
        action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
      runas=root
  services:
    systemd:
      httpd:
        enabled: true
        ensureRunning: true
      cfn-hup:
        enabled: true
        ensureRunning: true
        files:
          - /etc/cfn/cfn-hup.conf
          - /etc/cfn/hooks.d/cfn-auto-reloader.conf
```

```
CreationPolicy:
  ResourceSignal:
    Timeout: PT5M

WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application
```

このテンプレートからスタックを起動するには

1. テンプレートをコピーし、システム上にローカルにテキストファイルとして保存します。このファイルは後続の手順で必要となるため、保存場所をメモしておいてください。
2. AWS マネジメントコンソールにサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
3. [スタックを作成する] を選択し、[新しいリソースの使用 (標準)] を選択します。
4. [既存のテンプレートを選択する] を選択します。
5. [テンプレートを指定する] で、[テンプレートファイルをアップロードする] を選択し、最初のステップで作成したファイルを参照し、[次へ] を選択します。
6. [スタックの詳細を指定する] ページで、[スタック名] として **UpdateTutorial** を入力します。
7. [パラメータ] で、すべてのパラメータを同じに保ち、[次へ] を 2 回選択します。
8. [確認して作成] 画面で、[送信] を選択します。

スタックのステータスが CREATE_COMPLETE に変更された後、[出力] タブにウェブサイトの URL が表示されます。WebsiteURL 出力の値を選択した場合、新しい PHP アプリケーションが実行されていることを確認できます。

ステップ 2: アプリケーションを更新する

スタックをデプロイしたので、今度はアプリケーションを更新してみましょう。アプリケーションによって出力されるテキストに簡単な変更を加えます。以下のテンプレートスニペットに示したように、index.php ファイルに echo コマンドを追加します。

```
files:
  /var/www/html/index.php:
    content: |
      <?php
      echo "<h1>Hello World!</h1>";
      echo "<p>This is an updated version of our application.</p>";
      ?>
    mode: '000644'
    owner: apache
    group: apache
```

テキストエディタを使用して、ローカルに保存したテンプレートファイルを手動で編集します。

次に、スタックを更新します。

更新されたテンプレートでスタックを更新するには

1. CloudFormation コンソールで、**UpdateTutorial** スタックを選択します。
2. [更新]、[直接更新を実行] を選択します。
3. [既存のテンプレートを置き換える] を選択します。
4. [テンプレートを指定する] で、[テンプレートファイルをアップロードする] を選択し、変更したテンプレートをアップロードしてから、[次へ] を選択します。
5. [スタックの詳細を指定する] ページで、すべてのパラメータを同じに保持し、[次へ] を選択します。
6. [確認] ページで変更内容を確認します。[変更]で、CloudFormation が WebServerInstance リソースを更新することがわかります。
7. [Submit] を選択してください。

スタックの状態が UPDATE_COMPLETE になったら、WebsiteURL の出力値を再度選択して、アプリケーションに対する変更が反映されていることを確認できます。cfn-hup デーモンが 2 分間隔で実行されます。そのため、スタックが更新されてからアプリケーションが変更されるまでに最大 2 分程度かかる場合があります。

更新された一連のリソースを確認するには、CloudFormation コンソールを使用します。[イベント] タブでスタックイベントに注目してください。この場合は、Amazon EC2 インスタンスのメタデータ `WebServerInstance` が更新されます。また CloudFormation によって他のリソース (`WebServerSecurityGroup`) も再評価され、ほかに変更がないことが確認されます。他のスタックリソースは変更されませんでした。CloudFormation によって更新されるスタック内のリソースは、そのスタックに対する変更の影響を受けるリソースだけです。そうした変更には、プロパティやメタデータの変更など直接的なものもあれば、`Ref`、`GetAtt` など、組み込みテンプレート関数を介したデータフローや依存関係に起因するものもあります。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

このシンプルな更新は、プロセスを示しています。ただし、Amazon EC2 インスタンスにデプロイするファイルやパッケージに対し、さらに複雑な変更を行うこともできます。たとえば、インスタンスに MySQL を追加しなければならなくなったとします。MySQL の PHP サポートも必要です。そのための作業は単に、追加サービスと併せて、必要なパッケージとファイルを構成に追加するだけです。そのうえでスタックを更新して変更をデプロイすることになります。

```
packages:
  yum:
    httpd: []
    php: []
    mysql: []
    php-mysql: []
    mysql-server: []
    mysql-libs: []

    ...

services:
  systemd:
    httpd:
      enabled: true
      ensureRunning: true
    cfn-hup:
      enabled: true
      ensureRunning: true
    files:
      - /etc/cfn/cfn-hup.conf
      - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    mysqld:
      enabled: true
      ensureRunning: true
```

CloudFormation のメタデータを更新して、アプリケーションで使用するパッケージを新しいバージョンに更新できます。これまでの例では、各パッケージのバージョンプロパティが空になっています。この場合、cfn-init によって最新バージョンのパッケージがインストールされます。

```
packages:
  yum:
    httpd: []
    php: []
```

必要であれば、パッケージのバージョン文字列を指定することもできます。以降のスタック更新要求でバージョン文字列を変更した場合、その新しいバージョンのパッケージがデプロイされます。以下に示したのは、RubyGems パッケージのバージョン番号を使用した例です。バージョンニングをサポートするパッケージはすべて、特定のバージョンを持つことができます。

```
packages:
  rubygems:
    mysql: []
    rubygems-update:
      - "1.6.2"
    rake:
      - "0.8.7"
    rails:
      - "2.3.11"
```

ステップ 3: キーペアを使用して SSH アクセスを追加する

テンプレートのリソースを更新して、テンプレート内で元々指定されていなかったプロパティを追加することもできます。これについて学ぶため、Amazon EC2 キーペアを既存の EC2 インスタンスに追加し、Amazon EC2 セキュリティグループでポート 22 を開きます。これにより、Secure Shell (SSH) を使用してインスタンスにアクセスできるようになります。

既存の Amazon EC2 インスタンスに SSH アクセスを追加するには

1. 既存の Amazon EC2 キーペアおよび SSH ロケーションの名前を渡すため、テンプレートに 2 つのパラメーターを追加します。

```
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the
    instance
```

```
Type: AWS::EC2::KeyPair::KeyName
ConstraintDescription: must be the name of an existing EC2 KeyPair.
```

SSHLocation:

Description: The IP address that can be used to SSH to the EC2 instances in CIDR format (e.g. 203.0.113.1/32)

Type: String

MinLength: 9

MaxLength: 18

Default: 0.0.0.0/0

AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\.\d{1,2}\$'

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

2. Amazon EC2 インスタンスに KeyName プロパティを追加します。

WebServerInstance:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref LatestAmiId

InstanceType: !Ref InstanceType

KeyName: !Ref KeyName

SecurityGroupIds:

- !Ref WebServerSecurityGroup

3. Amazon EC2 セキュリティグループの Ingress ルールに、ポート 22 と SSH ロケーションを追加します。

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable HTTP access via port 80 and SSH access via port 22

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: !Ref SSHLocation

4. 「[ステップ 2: アプリケーションを更新する](#)」で説明されているのと同じステップを使用してスタックを更新します。

ステップ 4: インスタンスタイプを更新する

次に、インスタンスタイプを変更して、基盤となるインフラストラクチャを更新する方法を示しましょう。

ここまでで構築したスタックは、t3.micro Amazon EC2 インスタンスを使用します。例として、新しく作成したウェブサイトには、t3.micro インスタンスで処理できる以上のトラフィックが発生しているとします。このため、m5.large Amazon EC2 インスタンスタイプに移行する必要があります。インスタンスタイプのアーキテクチャが変わると、異なる AMI でインスタンスが作成される必要があります。ただし、t3.micro と m5.large の両方は同じ CPU アーキテクチャを使用し、Amazon Linux 2 (x86_64) AMI を実行します。詳細については、Amazon EC2 ユーザーガイドの[インスタンスタイプ変更の互換性](#)を参照してください。

前のステップで編集したテンプレートを使用して、インスタンスタイプを変更しましょう。InstanceType はテンプレートへの入力パラメータであるため、テンプレートを変更する必要はありません。[スタックの詳細を指定する] ページでパラメータの値を変更できます。

スタックを新しいパラメータ値で更新するには

1. CloudFormation コンソールで、**UpdateTutorial** スタックを選択します。
2. [更新]、[直接更新を実行] を選択します。
3. [既存のテンプレート使用する] を選択し、[次へ] を選択します。
4. [スタックの詳細を指定する] ページで、InstanceType テキストボックスの値を t3.micro から m5.large に変更します。次に、[次へ] を 2 回選択します。
5. [確認] ページで変更内容を確認します。[変更]で、CloudFormation が WebServerInstance リソースを更新することがわかります。
6. [Submit] を選択してください。

インスタンスを起動および停止することで、EBS-backed Amazon EC2 インスタンスのインスタンスタイプを動的に変更できます。CloudFormation は、インスタンスタイプを更新してインスタンスを再起動することで変更を最適化しようとします。そのため、インスタンス ID は変更されません。ただしインスタンスが再開されると、インスタンスのパブリック IP アドレスは変更されます。変更後に Elastic IP アドレスが正しくバインドされるよう、CloudFormation は Elastic IP アドレスも更新します。CloudFormation コンソールの [イベント] タブで変更を見ることができます。

AWS マネジメントコンソール からインスタンスタイプを調べるには、Amazon EC2 コンソールを開き、インスタンスを探します。

ステップ 5: AMI を更新する

次に、Amazon Linux の次世代である Amazon Linux 2023 を使用するようにスタックを更新しましょう。

AMI の更新は、インスタンスの置き換えを必要とする大幅な変更です。単純にインスタンスの開始と停止によって AMI を変更することはできません。このような変更は、CloudFormation によって不変のリソースプロパティに対する変更とみなされます。不変のプロパティに変更を加える場合、CloudFormation は、代替リソース (このケースでは、新しい AMI を実行する新しい Amazon EC2 インスタンス) を起動する必要があります。

Amazon Linux 2023 を使用するために、スタックテンプレートを更新する方法を見てみましょう。主な変更には、AMI パラメータの更新と、yum から dnf パッケージマネージャーへの変更が含まれます。

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Parameters:
```

```
  LatestAmiId:
```

```
    Description: The latest Amazon Linux 2023 AMI from the Parameter Store
```

```
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
```

```
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64'
```

```
  InstanceType:
```

```
    Description: WebServer EC2 instance type
```

```
    Type: String
```

```
    Default: t3.micro
```

```
    AllowedValues:
```

- t3.nano
- t3.micro
- t3.small
- t3.medium
- t3a.nano
- t3a.micro
- t3a.small
- t3a.medium
- m5.large
- m5.xlarge
- m5.2xlarge
- m5a.large
- m5a.xlarge
- m5a.2xlarge

- c5.large
- c5.xlarge
- c5.2xlarge
- r5.large
- r5.xlarge
- r5.2xlarge
- r5a.large
- r5a.xlarge
- r5a.2xlarge

ConstraintDescription: must be a valid EC2 instance type.

KeyName:

Description: Name of an existing EC2 KeyPair to enable SSH access to the instance

Type: AWS::EC2::KeyPair::KeyName

ConstraintDescription: must be the name of an existing EC2 KeyPair.

SSHLocation:

Description: The IP address that can be used to SSH to the EC2 instances in CIDR format (e.g. 203.0.113.1/32)

Type: String

MinLength: 9

MaxLength: 18

Default: 0.0.0.0/0

AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\.\d{1,2}\$'

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:

WebServerInstance:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref LatestAmiId

InstanceType: !Ref InstanceType

KeyName: !Ref KeyName

SecurityGroupIds:

- !Ref WebServerSecurityGroup

UserData:

Fn::Base64: !Sub |

#!/bin/bash -xe

Get the latest CloudFormation package

dnf update -y aws-cfn-bootstrap

Run cfn-init

/opt/aws/bin/cfn-init -v --stack \${AWS::StackName} --resource

WebServerInstance --region \${AWS::Region} || error_exit 'Failed to run cfn-init'

```

    # Start up the cfn-hup daemon to listen for changes to the EC2 instance
metadata
    /opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'
    # Signal success or failure
    /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
WebServerInstance --region ${AWS::Region}
Metadata:
    AWS::CloudFormation::Init:
    config:
    packages:
    dnf:
    httpd: []
    php: []
    files:
    /var/www/html/index.php:
    content: |
    <?php
    echo "<h1>Hello World!</h1>";
    echo "<p>This is an updated version of our application.</p>";
    echo "<p>Running on Amazon Linux 2023!</p>";
    ?>
    mode: '000644'
    owner: apache
    group: apache
    /etc/cfn/cfn-hup.conf:
    content: !Sub |
    [main]
    stack=${AWS::StackId}
    region=${AWS::Region}
    # The interval used to check for changes to the resource metadata in
minutes. Default is 15
    interval=2
    mode: '000400'
    owner: root
    group: root
    /etc/cfn/hooks.d/cfn-auto-reloader.conf:
    content: !Sub |
    [cfn-auto-reloader-hook]
    triggers=post.update
    path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
    action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
    runas=root
    services:

```

```

systemd:
  httpd:
    enabled: true
    ensureRunning: true
  cfn-hup:
    enabled: true
    ensureRunning: true
  files:
    - /etc/cfn/cfn-hup.conf
    - /etc/cfn/hooks.d/cfn-auto-reloader.conf

```

```

CreationPolicy:
  ResourceSignal:
    Timeout: PT5M

```

```

WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation

```

```

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application

```

「[ステップ 2: アプリケーションを更新する](#)」で説明されているのと同じステップを使用してスタックを更新します。

新しいインスタンスが実行状態になった後、CloudFormation は、スタック内の他のリソースが新しいリソースを指し示すように更新します。新しいリソースがすべて作成されると、古いリソースは削除されます。このプロセスを UPDATE_CLEANUP といいます。更新後、スタック内のインスタンスの ID とアプリケーション URL が変わっています。[イベント] テーブル内のイベントには、「Requested update has a change to an immutable property and hence creating a new physical resource」という説明が表示され、リソースが置き換えられたことがわかります。

別の方法として: 更新対象の AMI に組み込むアプリケーションコードを記述した場合、同じスタック更新メカニズムを使用して AMI を更新し、新しいアプリケーションをロードすることができます。

カスタムアプリケーションコードを使用して AMI を更新するには

1. アプリケーションまたはオペレーティング システムの変更を含んだ新しい AMI を作成します。詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EBS-backed AMI の作成](#)」を参照してください。
2. テンプレートに新しい AMI ID を追加します。
3. 「[ステップ 2: アプリケーションを更新する](#)」で説明されているのと同じステップを使用してスタックを更新します。

スタックが更新されると、AMI ID が変更されたことを CloudFormation が検出し、先ほどと同じようにスタックの更新を開始します。

アベイラビリティと影響に関する検討事項

プロパティによって、スタック内のリソースに異なる影響が及ぼされます。CloudFormation を使用してあらゆるプロパティを更新できますが、変更を加える前に、以下の点について検討する必要があります。

1. アップデートによって、リソース自体にどのような影響が与えられるか。たとえば、アラームのしきい値を変更すると、アップデート中にアラームが非アクティブになります。前述の通り、インスタンスタイプを変更するには、インスタンスをいったん停止して再開する必要があります。CloudFormation は、基盤となるリソースの更新または変更アクションを使用して、リソースを変更します。アップデートの影響を理解するには、特定のリソースのドキュメントを参照してください。
2. 変更は可変か不変か。Amazon EC2 インスタンスでの AMI の変更など、リソースプロパティへの変更のタイプによっては、基盤となるサービスにサポートされていない場合があります。可変の変更の場合、CloudFormation は、基盤となるリソースの Update または Modify タイプの API を使用します。不変のプロパティの変更の場合、CloudFormation は、アップデートされたプロパティを使用して新しいリソースを作成し、古いリソースを削除する前にそれらをスタックにリンクします。CloudFormation はスタックリソースのダウンタイムを削減しようとはしますが、リソースは複数のステップが必要な処理であり、ある程度の時間がかかります。スタックの再構成中、アプリケーションは完全には機能しません。たとえば、要求への応答や、データベースへのアクセスができない可能性があります。

関連リソース

CloudFormation を使用してアプリケーションを開始する方法と、Puppet や Opscode Chef のようなその他の構成やデプロイメントサービスと統合する方法については、以下のホワイトペーパーをご参照ください。

- [CloudFormation によるアプリケーションのブートストラップ](#)
- [CloudFormation と Opscode Chef の統合](#)
- [CloudFormation と Puppet の統合](#)

スケールおよび負荷分散されたアプリケーションを作成する

このチュートリアルでは、スケールおよび負荷分散されたアプリケーションの設定に役立つスタックを作成します。チュートリアルでは、スタックの作成に使用するサンプルテンプレートを提供します。このサンプルテンプレートは、Auto Scaling グループ、Application Load Balancer、ロードバランサーと Auto Scaling グループへのトラフィックを制御するセキュリティグループ、およびスケーリングアクティビティに関する通知を発行する Amazon SNS 通知設定をプロビジョニングします。

このテンプレートは、1 つ以上の Amazon EC2 インスタンスと Application Load Balancer を作成します。このテンプレートからスタックを作成した場合、AWS リソースに対する料金が発生します。

フルスタックテンプレート

テンプレートから始めましょう。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Description: The EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.micro
      - t3.small
      - t3.medium
  KeyName:
    Description: Name of an existing EC2 key pair to allow SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
  LatestAmiId:
```

Description: The latest Amazon Linux 2 AMI from the Parameter Store

Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

OperatorEmail:

Description: The email address to notify when there are any scaling activities

Type: String

SSHLocation:

Description: The IP address range that can be used to SSH to the EC2 instances

Type: String

MinLength: 9

MaxLength: 18

Default: 0.0.0.0/0

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Subnets:

Type: 'List<AWS::EC2::Subnet::Id>'

Description: At least two public subnets in different Availability Zones in the selected VPC

VPC:

Type: AWS::EC2::VPC::Id

Description: A virtual private cloud (VPC) that enables resources in public subnets to connect to the internet

Resources:

ELBSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: ELB Security Group

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

EC2SecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: EC2 Security Group

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

SourceSecurityGroupId:

Fn::GetAtt:

- ELBSecurityGroup


```
- GroupId
- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp: !Ref SSHLocation
EC2TargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 30
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 15
    HealthyThresholdCount: 5
    Matcher:
      HttpCode: '200'
    Name: EC2TargetGroup
    Port: 80
    Protocol: HTTP
    TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: '20'
    UnhealthyThresholdCount: 3
    VpcId: !Ref VPC
ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref EC2TargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: 80
    Protocol: HTTP
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internet-facing
    Subnets: !Ref Subnets
    SecurityGroups:
      - !GetAtt ELBSecurityGroup.GroupId
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
    LaunchTemplateData:
      ImageId: !Ref LatestAmiId
```

```

    InstanceType: !Ref InstanceType
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref EC2SecurityGroup
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<h1>Hello World!</h1>" > /var/www/html/index.html
  NotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MaxSize: '3'
      MinSize: '1'
      NotificationConfigurations:
        - TopicARN: !Ref NotificationTopic
          NotificationTypes: ['autoscaling:EC2_INSTANCE_LAUNCH',
'autoscaling:EC2_INSTANCE_LAUNCH_ERROR', 'autoscaling:EC2_INSTANCE_TERMINATE',
'autoscaling:EC2_INSTANCE_TERMINATE_ERROR']
      TargetGroupARNs:
        - !Ref EC2TargetGroup
      VPCZoneIdentifier: !Ref Subnets

```

JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "InstanceType": {
      "Description": "The EC2 instance type",
      "Type": "String",

```

```

        "Default": "t3.micro",
        "AllowedValues": [
            "t3.micro",
            "t3.small",
            "t3.medium"
        ],
    },
    "KeyName": {
        "Description": "Name of an existing EC2 key pair to allow SSH access to the
instances",
        "Type": "AWS::EC2::KeyPair::KeyName"
    },
    "LatestAmiId": {
        "Description": "The latest Amazon Linux 2 AMI from the Parameter Store",
        "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
        "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
    },
    "OperatorEmail": {
        "Description": "The email address to notify when there are any scaling
activities",
        "Type": "String"
    },
    "SSHLocation": {
        "Description": "The IP address range that can be used to SSH to the EC2
instances",
        "Type": "String",
        "MinLength": 9,
        "MaxLength": 18,
        "Default": "0.0.0.0/0",
        "ConstraintDescription": "Must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "Subnets": {
        "Type": "List<AWS::EC2::Subnet::Id>",
        "Description": "At least two public subnets in different Availability Zones in the
selected VPC"
    },
    "VPC": {
        "Type": "AWS::EC2::VPC::Id",
        "Description": "A virtual private cloud (VPC) that enables resources in public
subnets to connect to the internet"
    },
    "Resources": {
        "ELBSecurityGroup": {

```

```
"Type": "AWS::EC2::SecurityGroup",
"Properties": {
  "GroupDescription": "ELB Security Group",
  "VpcId": {
    "Ref": "VPC"
  },
  "SecurityGroupIngress": [
    {
      "IpProtocol": "tcp",
      "FromPort": 80,
      "ToPort": 80,
      "CidrIp": "0.0.0.0/0"
    }
  ]
},
"EC2SecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "EC2 Security Group",
    "VpcId": {
      "Ref": "VPC"
    },
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "SourceSecurityGroupId": {
          "Fn::GetAtt": [
            "ELBSecurityGroup",
            "GroupId"
          ]
        }
      },
      {
        "IpProtocol": "tcp",
        "FromPort": 22,
        "ToPort": 22,
        "CidrIp": {
          "Ref": "SSHLocation"
        }
      }
    ]
  }
}
```

```
    }
  },
  "EC2TargetGroup":{
    "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties":{
      "HealthCheckIntervalSeconds":30,
      "HealthCheckProtocol":"HTTP",
      "HealthCheckTimeoutSeconds":15,
      "HealthyThresholdCount":5,
      "Matcher":{
        "HttpCode":"200"
      },
      "Name":"EC2TargetGroup",
      "Port":80,
      "Protocol":"HTTP",
      "TargetGroupAttributes":[
        {
          "Key":"deregistration_delay.timeout.seconds",
          "Value":"20"
        }
      ],
      "UnhealthyThresholdCount":3,
      "VpcId":{
        "Ref":"VPC"
      }
    }
  },
  "ALBListener":{
    "Type":"AWS::ElasticLoadBalancingV2::Listener",
    "Properties":{
      "DefaultActions":[
        {
          "Type":"forward",
          "TargetGroupArn":{
            "Ref":"EC2TargetGroup"
          }
        }
      ],
      "LoadBalancerArn":{
        "Ref":"ApplicationLoadBalancer"
      },
      "Port":80,
      "Protocol":"HTTP"
    }
  }
```

```
    },
    "ApplicationLoadBalancer":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{
        "Scheme":"internet-facing",
        "Subnets":{
          "Ref":"Subnets"
        },
      },
      "SecurityGroups":[
        {
          "Fn::GetAtt":[
            "ELBSecurityGroup",
            "GroupId"
          ]
        }
      ]
    },
  },
  "LaunchTemplate":{
    "Type":"AWS::EC2::LaunchTemplate",
    "Properties":{
      "LaunchTemplateName":{
        "Fn::Sub":"${AWS::StackName}-launch-template"
      },
      "LaunchTemplateData":{
        "ImageId":{
          "Ref":"LatestAmiId"
        },
        "InstanceType":{
          "Ref":"InstanceType"
        },
        "KeyName":{
          "Ref":"KeyName"
        },
        "SecurityGroupIds":[
          {
            "Ref":"EC2SecurityGroup"
          }
        ],
        "UserData":{
          "Fn::Base64":{
            "Fn::Join":[
              "",
              [

```

```

        "#!/bin/bash\n",
        "yum update -y\n",
        "yum install -y httpd\n",
        "systemctl start httpd\n",
        "systemctl enable httpd\n",
        "echo \"<h1>Hello World!</h1>\" > /var/www/html/index.html"
    ]
  ]
}
}
}
},
"NotificationTopic":{
  "Type":"AWS::SNS::Topic",
  "Properties":{
    "Subscription":[
      {
        "Endpoint":{
          "Ref":"OperatorEmail"
        },
        "Protocol":"email"
      }
    ]
  }
},
"WebServerGroup":{
  "Type":"AWS::AutoScaling::AutoScalingGroup",
  "Properties":{
    "LaunchTemplate":{
      "LaunchTemplateId":{
        "Ref":"LaunchTemplate"
      },
      "Version":{
        "Fn::GetAtt":[
          "LaunchTemplate",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize":"3",
    "MinSize":"1",
    "NotificationConfigurations":[
      {

```

```
        "TopicARN":{
            "Ref":"NotificationTopic"
        },
        "NotificationTypes":[
            "autoscaling:EC2_INSTANCE_LAUNCH",
            "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
            "autoscaling:EC2_INSTANCE_TERMINATE",
            "autoscaling:EC2_INSTANCE_TERMINATE_ERROR"
        ]
    }
],
"TargetGroupARNs":[
    {
        "Ref":"EC2TargetGroup"
    }
],
"VPCZoneIdentifier":{
    "Ref":"Subnets"
}
}
}
}
```

テンプレートのチュートリアル

このテンプレートの最初の部分では、Parameters を指定します。CloudFormation がスタックを正常にプロビジョニングするには、各パラメーターに実行時に値を割り当てる必要があります。テンプレートで後で指定されるリソースは、これらの値を参照し、データを使用します。

- **InstanceType:** Amazon EC2 Auto Scaling がプロビジョニングする EC2 インスタンスのタイプ。指定しない場合は、デフォルトの t3.micro が使用されます。
- **KeyName:** インスタンスへの SSH アクセスを許可する既存の EC2 キーペア。
- **LatestAmiId:** インスタンスの Amazon マシンイメージ (AMI)。指定しない場合、インスタンスは、AWS によって維持される AWS Systems Manager パブリックパラメータを使用して、Amazon Linux 2 AMI で起動されます。詳細については、「AWS Systems Manager ユーザーガイド」の「[パブリックパラメータの検索](#)」を参照してください。
- **OperatorEmail:** スケーリングアクティビティの通知を送信するメールアドレス。
- **SSHLocation:** インスタンスへの SSH 接続に使用できる IP アドレス範囲。

- Subnets: 少なくとも 2 つのパブリックサブネットが異なるアベイラビリティゾーンに存在している。
- VPC: パブリックサブネット内のリソースがインターネットに接続できるようにするアカウント内の仮想プライベートクラウド (VPC)。

 Note

デフォルトの VPC とデフォルトのサブネットを使用して、インスタンスがインターネットにアクセスできるようにすることができます。後者の場合は、VPC に、作業中のリージョンの各アベイラビリティゾーンにマッピングされたサブネットがあることを確認してください。ロードバランサーを作成するには、2 つ以上のパブリックサブネットが必要です。

このテンプレートの次の部分では、Resources を指定します。このセクションでは、スタックリソースとそのプロパティを指定します。

[AWS::EC2::SecurityGroup](#) リソース ELBSecurityGroup

- SecurityGroupIngress には、ポート 80 上のすべての IP アドレス ("CidrIp": "0.0.0.0/0") からのアクセスを許可する TCP イングレスルールが含まれています。

[AWS::EC2::SecurityGroup](#) リソース EC2SecurityGroup

- SecurityGroupIngress には、1) SSHLocation 入力パラメータ用に指定した IP アドレス範囲からの SSH アクセス (ポート 22) を許可する TCP イングレスルール、および 2) ロードバランサーのセキュリティグループを指定することによってロードバランサーからのアクセスを許可する TCP イングレスルールという 2 つのイングレスルールが含まれています。[GetAtt](#) 関数は、論理名 ELBSecurityGroup を持つセキュリティグループの ID を取得するために使用されます。

[AWS::ElasticLoadBalancingV2::TargetGroup](#) リソース EC2TargetGroup

- Port、Protocol、および HealthCheckProtocol は、ApplicationLoadBalancer がトラフィックをルーティングし、Elastic Load Balancing が EC2 インスタンスのヘルスをチェックするために使用する EC2 インスタンスのポート (80) とプロトコル (HTTP) を指定します。
- HealthCheckIntervalSeconds は、EC2 インスタンスのヘルスチェックの間隔が 30 秒であることを指定します。HealthCheckTimeoutSeconds は、ヘルスチェックターゲット

からのレスポンスを Elastic Load Balancing が待機する時間です (この例では 15 秒)。タイムアウトの時間が経過すると、Elastic Load Balancing は、EC2 インスタンスのヘルスチェックを異常としてマークします。EC2 インスタンスが 3 回連続してヘルスチェックに失敗すると (UnhealthyThresholdCount)、Elastic Load Balancing は、そのインスタンスが 5 回連続してヘルスチェックに合格するまで (HealthyThresholdCount)、その EC2 インスタンスへのトラフィックのルーティングを停止します。その時点で、Elastic Load Balancing は、そのインスタンスが正常であるとみなし、そのインスタンスへのトラフィックのルーティングを再開します。

- TargetGroupAttributes は、ターゲットグループの登録解除の遅延の値を 20 秒に更新します。デフォルトでは、Elastic Load Balancing は登録解除プロセスを完了するまで 300 秒間待機します。

[AWS::ElasticLoadBalancingV2::Listener](#) リソース ALBListener

- DefaultActions は、ロードバランサーがリッスンするポート、ロードバランサーがリクエストを転送するターゲットグループ、およびリクエストのルーティングに使用されるプロトコルを指定します。

[AWS::ElasticLoadBalancingV2::LoadBalancer](#) リソース ApplicationLoadBalancer

- Subnets は、Subnets 入力パラメータの値を、ロードバランサーノードが作成されるパブリックサブネットのリストとして受け取ります。
- SecurityGroup は、ロードバランサーノードが着信トラフィックを制御するための仮想ファイアウォールとして機能するセキュリティグループの ID を取得します。[GetAtt](#) 関数は、論理名 ELBSecurityGroup を持つセキュリティグループの ID を取得するために使用されます。

[AWS::EC2::LaunchTemplate](#) リソース LaunchTemplate

- ImageId は、使用する AMI として LatestAmiId 入力パラメータの値を受け取ります。
- KeyName は、使用する EC2 キーペアとして KeyName 入力パラメータの値を受け取ります。
- SecurityGroupIds は、EC2 インスタンスが着信トラフィックを制御するための仮想ファイアウォールとして機能する、論理名 EC2SecurityGroup を持つセキュリティグループの ID を取得します。
- UserData は、インスタンスが稼働を開始した後に実行される設定スクリプトです。この例では、スクリプトは Apache をインストールし、index.html ファイルを作成します。

[AWS::SNS::Topic](#) リソース NotificationTopic

- Subscription は、スケーリングアクティビティがある場合、OperatorEmail 入力パラメータの値を通知の受信者のメールアドレスとして受け取ります。

[AWS::AutoScaling::AutoScalingGroup](#) resource WebServerGroup

- MinSize および MaxSize には、Auto Scaling グループでの EC2 インスタンスの最小数と最大数を設定します。
- TargetGroupARNs は、論理名 EC2TargetGroup を持つターゲットグループの ARN を取得します。この Auto Scaling グループがスケールすると、このターゲットグループに対するインスタンスの登録および登録解除が自動的に行われます。
- VPCZoneIdentifier は、Subnets 入力パラメータの値を、EC2 インスタンスを作成できるパブリックサブネットのリストとして受け取ります。

ステップ 1: スタックを起動する

スタックを起動する前に、Amazon EC2、Amazon EC2 Auto Scaling、AWS Systems Manager、Elastic Load Balancing、Amazon SNS、および CloudFormation のすべてのサービスを使用するための AWS Identity and Access Management (IAM) 許可があることを確認してください。

次の手順では、ファイルからサンプルスタックテンプレートをアップロードします。ローカルマシンでテキストエディタを開き、いずれかのテンプレートを追加します。次に、sampleloadbalancedappstack.template という名前でファイルを保存します。

スタックテンプレートを起動するには

1. AWS マネジメントコンソールにサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタックの作成] を選択し、[新しいリソースの使用 (標準)] を選択します。
3. [テンプレートを指定] で、[テンプレートファイルをアップロード]、[ファイルを選択] を選択して sampleloadbalancedappstack.template ファイルをアップロードします。
4. [次へ] を選択します。
5. [スタックの詳細を指定] ページで、スタック名 (例: **SampleLoadBalancedAppStack**) を入力します。

6. [パラメータ] で、スタックのパラメータを確認し、[OperatorEmail]、[SSHLocation]、[KeyName]、[VPC]、[Subnets] など、デフォルト値がないすべてのパラメータの値を指定します。
7. [次へ] を 2 回選択します。
8. [レビュー] ページで、設定を確認します。
9. [Submit] を選択してください。

スタックのステータスは、CloudFormation コンソールの [ステータス] 列で確認できます。CloudFormation がスタックを正常に作成すると、[CREATE_COMPLETE] ステータスが表示されます。

 Note

スタックを作成した後、メールアドレスで通知の受信を開始するには、サブスクリプションを確認する必要があります。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「[Auto Scaling グループのスケール時に Amazon SNS 通知を受け取る](#)」を参照してください。

ステップ 2: サンプルリソースをクリーンアップする

未使用のサンプルリソースについて課金されないようにするには、スタックを削除します。

スタックを削除するには

1. CloudFormation コンソールで、[SampleLoadBalancedAppStack] スタックを選択します。
2. [削除] を選択します。
3. 確認メッセージで、[スタックを削除] を選択します。

[SampleLoadBalancedAppStack] のステータスが [DELETE_IN_PROGRESS] に変わります。CloudFormation がスタックの削除を完了すると、リストからスタックが削除されます。

このチュートリアルサンプルテンプレートを使用して、独自のスタックテンプレートを構築します。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「[チュートリアル: スケーリングとロードバランシングを使用するアプリケーションを設定する](#)」を参照してください。

別の AWS アカウントで VPC とピア接続する

[AWS::EC2::VPCPeeringConnection](#) を使用して、別の AWS アカウントの仮想プライベートクラウド (VPC) とピア接続できます。これにより、2 つの VPC 間でトラフィックをルーティングできるネットワーク接続が作成され、同じネットワーク内に存在しているかのように通信できます。VPC ピア接続により、データアクセスとデータ転送が容易になります。

VPC ピアリング接続を確立するには、単一の CloudFormation スタック内で別々に AWS アカウントを 2 つ認可する必要があります。

VPC ピアリングとその制限については、「[Amazon VPC ピアリングガイド](#)」を参照してください。

前提条件

1. ピアリング接続には、ピア VPC ID、ピア AWS アカウント ID、および[クロスアカウントアクセス](#)が必要です。

Note

このチュートリアルでは、2 つのアカウントについて説明します。1 つめは、クロスアカウントのピア接続を許可するアカウント (アクセプタアカウント) です。2 つめは、ピア接続をリクエストするアカウント (リクエスタアカウント) です。

2. VPC ピア接続を受け入れるには、クロスアカウントアクセスロールを自分で引き受ける必要があります。リソースは、同じアカウントで VPC ピア接続リソースと同じように動作します。IAM 管理者がクロスアカウントロールの継承を許可する方法については、「IAM ユーザーガイド」の「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

ステップ 1: VPC とクロスアカウントロールの作成

このステップでは、アクセプタアカウントに VPC とロールを作成します。

VPC とクロスアカウントアクセスロールを作成するには

1. AWS マネジメントコンソールにサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。

3. [前提条件 - テンプレートの準備] で、[既存のテンプレートを選択] を選択し、テンプレートファイルをアップロードし、ファイルを選択します。
4. ローカルマシンでテキストエディタを開き、次のいずれかのテンプレートを追加します。ファイルを保存し、コンソールに戻って、テンプレートファイルとして選択します。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and an assumable role for cross account VPC peering.",
  "Parameters": {
    "PeerRequesterAccountId": {
      "Type": "String"
    }
  },
  "Resources": {
    "vpc": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.1.0.0/16",
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
      }
    },
    "peerRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Principal": {
                "AWS": {
                  "Ref": "PeerRequesterAccountId"
                }
              },
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow"
            }
          ]
        }
      }
    }
  }
}
```

```

    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "root",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "ec2:AcceptVpcPeeringConnection",
              "Resource": "*"
            }
          ]
        }
      }
    ]
  }
},
"Outputs": {
  "VPCId": {
    "Value": {
      "Ref": "vpc"
    }
  },
  "RoleARN": {
    "Value": {
      "Fn::GetAtt": [
        "peerRole",
        "Arn"
      ]
    }
  }
}
}
}

```

Example YAML

AWSTemplateFormatVersion: 2010-09-09

Description: Create a VPC and an assumable role for cross account VPC peering.

Parameters:

PeerRequesterAccountId:

```
Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  peerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Principal:
              AWS: !Ref PeerRequesterAccountId
            Action:
              - 'sts:AssumeRole'
            Effect: Allow
      Path: /
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action: 'ec2:AcceptVpcPeeringConnection'
                Resource: '*'
Outputs:
  VPCId:
    Value: !Ref vpc
  RoleARN:
    Value: !GetAtt
      - peerRole
      - Arn
```

5. [次へ] を選択します。
6. スタックに名前 (**VPC-owner** など) を付け、PeerRequesterAccountId フィールドにリクエストアカウントの AWS アカウント ID を入力します。
7. デフォルトを受け入れて [次へ] を選択します。
8. [CloudFormation によって IAM リソースが作成される場合があることを承認します] を選択し、[スタックを作成] を選択します。

ステップ 2: **AWS::EC2::VPCPeeringConnection** を含むテンプレートを作成する

VPC とクロスアカウントロールが作成できたので、別の AWS アカウント (リクエスタアカウント) を使用して VPC とピア接続できます。

[AWS::EC2::VPCPeeringConnection](#) リソースを含むテンプレートを作成するには

1. CloudFormation コンソールのホームページに戻ります。
2. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。
3. [前提条件 - テンプレートの準備] で、[既存のテンプレートを選択] を選択し、テンプレートファイルをアップロードし、ファイルを選択します。
4. ローカルマシンでテキストエディタを開き、次のいずれかのテンプレートを追加します。ファイルを保存し、コンソールに戻って、テンプレートファイルとして選択します。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and a VPC Peering connection using the PeerRole to accept.",
  "Parameters": {
    "PeerVPCAccountId": {
      "Type": "String"
    },
    "PeerVPCId": {
      "Type": "String"
    },
    "PeerRoleArn": {
      "Type": "String"
    }
  },
  "Resources": {
    "vpc": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.2.0.0/16",
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
      }
    }
  }
}
```

```

    },
    "vpcPeeringConnection": {
      "Type": "AWS::EC2::VPCPeeringConnection",
      "Properties": {
        "VpcId": {
          "Ref": "vpc"
        },
        "PeerVpcId": {
          "Ref": "PeerVPCId"
        },
        "PeerOwnerId": {
          "Ref": "PeerVPCAccountId"
        },
        "PeerRoleArn": {
          "Ref": "PeerRoleArn"
        }
      }
    }
  },
  "Outputs": {
    "VPCId": {
      "Value": {
        "Ref": "vpc"
      }
    },
    "VPCPeeringConnectionId": {
      "Value": {
        "Ref": "vpcPeeringConnection"
      }
    }
  }
}

```

Example YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and a VPC Peering connection using the PeerRole to
  accept.
Parameters:
  PeerVPCAccountId:
    Type: String
  PeerVPCId:
    Type: String

```

```

PeerRoleArn:
  Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.2.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  vpcPeeringConnection:
    Type: AWS::EC2::VPCPeeringConnection
    Properties:
      VpcId: !Ref vpc
      PeerVpcId: !Ref PeerVPCId
      PeerOwnerId: !Ref PeerVPCAccountId
      PeerRoleArn: !Ref PeerRoleArn
Outputs:
  VPCId:
    Value: !Ref vpc
  VPCPeeringConnectionId:
    Value: !Ref vpcPeeringConnection

```

5. [次へ] を選択します。
6. スタックに名前 (**VPC-peering-connection** など) を付けます。
7. デフォルトを受け入れて [次へ] を選択します。
8. [CloudFormation によって IAM リソースが作成される場合があることを承認します] を選択し、[スタックを作成] を選択します。

制限がより厳格なポリシーでテンプレートを作成する

別の AWS アカウント を使用した VPC ピア接続で、制限の厳しいポリシーを作成できます。

次の例では、VPC ピアの所有者のテンプレート (上記のステップ 1 で作成したアクセプタアカウント) を変更し、より制限を厳しくする方法を示しています。

Example JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and an assumable role for cross account VPC peering.",
  "Parameters": {

```

```

    "PeerRequesterAccountId":{
      "Type":"String"
    }
  },
  "Resources":{
    "peerRole":{
      "Type":"AWS::IAM::Role",
      "Properties":{
        "AssumeRolePolicyDocument":{
          "Statement":[
            {
              "Action":[
                "sts:AssumeRole"
              ],
              "Effect":"Allow",
              "Principal":{
                "AWS":{
                  "Ref":"PeerRequesterAccountId"
                }
              }
            }
          ]
        }
      },
      "Path":"/",
      "Policies":[
        {
          "PolicyDocument":{
            "Statement":[
              {
                "Action":"ec2:acceptVpcPeeringConnection",
                "Effect":"Allow",
                "Resource":{
                  "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}"
                }
              },
              {
                "Action":"ec2:acceptVpcPeeringConnection",
                "Condition":{
                  "StringEquals":{
                    "ec2:AcceptorVpc":{
                      "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}"
                    }
                  }
                }
              }
            ]
          }
        }
      ]
    }
  }
}

```

```
        },
        "Effect": "Allow",
        "Resource": {
            "Fn::Sub": "arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-
peering-connection/*"
        }
    ],
    "Version": "2012-10-17"
},
"PolicyName": "root"
}
]
}
},
"vpc": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "CidrBlock": "10.1.0.0/16",
        "EnableDnsHostnames": false,
        "EnableDnsSupport": false,
        "InstanceTenancy": "default"
    }
}
},
"Outputs": {
    "RoleARN": {
        "Value": {
            "Fn::GetAtt": [
                "peerRole",
                "Arn"
            ]
        }
    },
    "VPCId": {
        "Value": {
            "Ref": "vpc"
        }
    }
}
}
```

Example YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and an assumable role for cross account VPC peering.
Parameters:
  PeerRequesterAccountId:
    Type: String
Resources:
  peerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              AWS:
                Ref: PeerRequesterAccountId
        Path: /
      Policies:
        - PolicyDocument:
            Statement:
              - Action: 'ec2:acceptVpcPeeringConnection'
                Effect: Allow
                Resource:
                  'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}'
              - Action: 'ec2:acceptVpcPeeringConnection'
                Condition:
                  StringEquals:
                    'ec2:AcceptorVpc':
                      'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}'
                Effect: Allow
                Resource:
                  'Fn::Sub': >-
                    arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-peering-
connection/*
            Version: 2012-10-17
            PolicyName: root
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16

```

```
    EnableDnsHostnames: false
    EnableDnsSupport: false
    InstanceTenancy: default
Outputs:
  RoleARN:
    Value:
      'Fn::GetAtt':
        - peerRole
        - Arn
  VPCId:
    Value:
      Ref: vpc
```

VPC にアクセスするには、上記のステップ 2 と同じリクエストテンプレートを使用できます。

詳細については、「Amazon VPC ピアリングガイド」の「[VPC ピアリングの Identity and Access Management](#)」を参照してください。

CloudFormation を使用して CodeDeploy を通じて ECS ブルー/グリーンデプロイを実行する

Amazon Elastic Container Service (Amazon ECS) で実行されているアプリケーションを更新するために、CodeDeploy ブルー/グリーンデプロイ戦略を使用できます。この戦略は、アプリケーションバージョンの変更による中断を最小限に抑えるのに役立ちます。

ブルー/グリーンデプロイでは、現在のライブ環境 (ブルー) とともに、新しいアプリケーション環境 (グリーン) を作成します。これにより、ライブトラフィックをブルー環境からグリーン環境にルーティングする前に、グリーン環境をモニタリングおよびテストできます。グリーン環境がライブトラフィックを処理するようになったら、ブルー環境を安全に終了できます。

CloudFormation を利用して ECS 上で CodeDeploy ブルー/グリーンデプロイを実行するには、スタックテンプレートに次の情報を含めます。

- `AWS::CodeDeploy::BlueGreen` フックを説明する Hooks セクション。
- `AWS::CodeDeployBlueGreen` 変換を指定する Transform セクション。

次のトピックでは、ECS でのブルー/グリーンデプロイのために CloudFormation テンプレートを設定する方法について説明します。

トピック

- [ブルー/グリーンデプロイについて](#)
- [CloudFormation を使用して ECS ブルー/グリーンデプロイメントを管理する際の考慮事項](#)
- [AWS::CodeDeploy::BlueGreen フック構文](#)
- [ブルー/グリーンデプロイテンプレートの例](#)

ブルー/グリーンデプロイについて

このトピックでは、CloudFormation でブルー/グリーンデプロイを実行する方法の概要を説明します。また、ブルー/グリーンデプロイ用の CloudFormation テンプレートを準備する方法についても説明します。

トピック

- [仕組み](#)
- [グリーンデプロイを開始するリソースの更新](#)
- [ECS ブルー/グリーンデプロイを実行するためのテンプレートの準備](#)
- [CloudFormation リソースを使用したブルー/グリーンデプロイのモデリング](#)
- [変更セット](#)
- [スタックイベントのモニタリング](#)
- [ブルー/グリーンデプロイ用の IAM 許可](#)

仕組み

CloudFormation で CodeDeploy を通じて ECS ブルー/グリーンデプロイを実行する場合は、まず、使用するトラフィックルーティングとスタビライズ設定を指定するなど、ブルーとグリーンの両方のアプリケーション環境のリソースを定義するスタックテンプレートを作成します。次に、そのテンプレートからスタックを作成します。これにより、ブルー (現在の) アプリケーションが生成されます。CloudFormation は、スタックの作成時にのみブルーリソースを作成します。グリーンデプロイのリソースは、必要になるまで作成されません。

その後、将来のスタックの更新で、ブルーのアプリケーションでタスク定義またはタスクセットリソースを更新した場合、CloudFormation は次のことを行います。

- 必要なグリーンアプリケーション環境リソースをすべて生成する
- 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する
- 青色のリソースを削除する

グリーンデプロイが成功してファイナライズされる前の任意の時点でエラーが発生した場合、CloudFormation は、グリーンデプロイ全体が開始される前の状態にスタックをロールバックします。

グリーンデプロイを開始するリソースの更新

特定の ECS リソースの特定のプロパティを更新するスタック更新を実行すると、CloudFormation はグリーンデプロイプロセスを開始します。このプロセスを開始するリソースは次のとおりです。

- [AWS::ECS::TaskDefinition](#)
- [AWS::ECS::TaskSet](#)

ただし、これらのリソースの更新に置換を必要とするプロパティの変更が含まれていない場合、グリーンデプロイは開始されません。詳細については、「[スタックリソースの更新動作を理解する](#)」を参照してください。

同じスタック更新オペレーションで、上記のリソースに対する更新と他のリソースに対する更新を組み合わせることはできないことに注意することが重要です。リストされているリソースと、同じスタック内の他のリソースの両方を更新する必要がある場合は、次の 2 つのオプションがあります。

- 2 つの別々のスタック更新オペレーションを実行します: 1 つは上記のリソースに対する更新のみを含み、もう 1 つは他のすべてのリソースに対する変更を含む別のスタック更新。
- テンプレートから Transform および Hooks セクションを削除し、スタックの更新を実行します。この場合、CloudFormation はグリーンデプロイを実行しません。

ECS ブルー/グリーンデプロイを実行するためのテンプレートの準備

スタックでブルー/グリーンデプロイメントを有効にするには、スタックの更新を実行する前に、以下のセクションをスタックテンプレートに含めます。

- `AWS::CodeDeployBlueGreen` 変換への参照をテンプレートに追加します。

```
"Transform": [  
  "AWS::CodeDeployBlueGreen"  
],
```

- `AWS::CodeDeploy::BlueGreen` フックを呼び出し、デプロイのプロパティを指定する Hooks セクションを追加します。詳細については、「[AWS::CodeDeploy::BlueGreen フック構文](#)」を参照してください。

- **Resources** セクションで、デプロイのブルーとグリーンのリソースを定義します。

これらのセクションは、最初にテンプレートを作成するときに (つまり、スタック自体を作成する前に) 追加することも、スタック更新を実行する前に既存のテンプレートに追加することもできます。新しいスタックにブルー/グリーンデプロイを指定した場合、CloudFormation はスタック作成時にブルーリソースのみを作成します。グリーンデプロイ用のリソースは、スタックの更新中に必要になるまで作成されません。

CloudFormation リソースを使用したブルー/グリーンデプロイのモデリング

ECS で CodeDeploy ブルー/グリーンデプロイを実行するには、CloudFormation テンプレートに、Amazon ECS サービスやロードバランサーなど、デプロイをモデル化するリソースを含める必要があります。これらのリソースが表す内容の詳細については、「AWS CodeDeploy ユーザーガイド」の「[Amazon ECS デプロイを開始する前に](#)」を参照してください。

要件	リソース	必須/オプション	置き換えられた場合、ブルー/グリーンデプロイを開始しますか？
Amazon ECS クラスター	AWS::ECS::Cluster	オプション。デフォルトのクラスターを使用できます。	いいえ
Amazon ECS サービス	AWS::ECS::Service	必須。	いいえ
アプリケーションまたは Network Load Balancer	AWS::ECS::Service LoadBalancer	必須。	いいえ
本稼働リスナー	AWS::ElasticLoadBalancingV2::Listener	必須。	いいえ
テストリスナー	AWS::ElasticLoadBalancingV2::Listener	オプション。	いいえ

要件	リソース	必須/オプション	置き換えられた場合、ブルー/グリーンデプロイを開始しますか？
2 つのターゲットグループ	AWS::ElasticLoadBalancingV2::TargetGroup	必須。	いいえ
Amazon ECS タスク定義	AWS::ECS::TaskDefinition	必須。	はい
Amazon ECS アプリケーション用のコンテナ	AWS::ECS::TaskDefinition Container Definition Name	必須。	いいえ
置き換えタスクセット用のポート	AWS::ECS::TaskDefinition PortMapping ContainerPort	必須。	いいえ

変更セット

グリーンデプロイを開始するスタックの更新を実行する前に、変更セットを作成することを強くお勧めします。これにより、スタックの更新を実行する前にスタックに加えられる実際の変更を確認できます。リソースの変更は、スタックの更新中に実行される順序で表示されない場合があることに注意してください。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

スタックイベントのモニタリング

ECS デプロイの各ステップで生成されるスタックイベントは、[Stack] (スタック) ページの [Events] (イベント) タブで、AWS CLI を使用して確認できます。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

ブルー/グリーンデプロイ用の IAM 許可

CloudFormation がブルー/グリーンデプロイを正常に実行するには、次の CodeDeploy アクセス権限が必要です。

- `codedeploy:Get*`
- `codedeploy:CreateCloudFormationDeployment`

詳細については、「サービス認可リファレンス」の「[CodeDeploy のアクション、リソース、および条件キー](#)」を参照してください。

CloudFormation を使用して ECS ブルー/グリーンデプロイメントを管理する際の考慮事項

CloudFormation を使用して CodeDeploy を通じて ECS ブルー/グリーンデプロイを実行するプロセスは、CodeDeploy のみを使用する標準の ECS デプロイとは異なります。これらの違いの詳細については、「AWS CodeDeploy ユーザーガイド」の「[CodeDeploy および CloudFormation を通じた Amazon ECS ブルー/グリーンデプロイの違い](#)」を参照してください。

CloudFormation を使用してブルー/グリーンデプロイを管理する場合、留意すべきいくつかの制限と考慮事項があります。

- 特定のリソースに対する更新のみがグリーンデプロイを開始します。詳細については、「[グリーンデプロイを開始するリソースの更新](#)」を参照してください。
- グリーンデプロイを開始するリソースへの更新と他のリソースへの更新を同じスタック更新に含めることはできません。詳細については、「[グリーンデプロイを開始するリソースの更新](#)」を参照してください。
- デプロイターゲットとして指定できる ECS サービスは、1 つだけです。
- CloudFormation によって難読化された値を持つパラメータは、グリーンデプロイ時に CodeDeploy によって更新できず、エラーやスタックの更新に失敗します。具体的には次のとおりです。
 - NoEcho 属性で定義されたパラメータ。
 - 動的な参照を使用して外部サービスから値を取得するパラメータ。動的参照の詳細については、[動的参照を使用して他のサービスに格納されている値を取得する](#) を参照してください。
- まだ進行中のグリーンデプロイをキャンセルするには、CodeDeploy または ECS ではなく、CloudFormation でスタックの更新をキャンセルします。詳細については、「[スタック更新をキャンセルする](#)」を参照してください。更新が完了した後にキャンセルすることはできません。ただし、以前の設定を使用してスタックを再度更新することはできます。
- 次の CloudFormation の機能は現在、ECS ブルー/グリーンデプロイを定義するテンプレートではサポートされていません。
 - 他のスタックから値をインポートするための[出力](#)の宣言または `Fn::ImportValue` の使用。

- リソースのインポート。リソースのインポートの詳細については、「[AWS リソースを CloudFormation スタックにインポートする](#)」を参照してください。
- ネストされたスタックリソースを含むテンプレートで `AWS::CodeDeploy::BlueGreen` フックを使用します。ネストされたスタックの詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。
- ネストされたスタックでの `AWS::CodeDeploy::BlueGreen` フックの使用。

AWS::CodeDeploy::BlueGreen フック構文

次の構文は、ECS ブルー/グリーンデプロイの `AWS::CodeDeploy::BlueGreen` フックの構造を示しています。

構文

```
"Hooks": {
  "Logical ID": {
    "Type": "AWS::CodeDeploy::BlueGreen",
    "Properties": {
      "TrafficRoutingConfig": {
        "Type": "Traffic routing type",
        "TimeBasedCanary": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        },
        "TimeBasedLinear": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        }
      },
      "AdditionalOptions": {"TerminationWaitTimeInMinutes": Integer},
      "LifecycleEventHooks": {
        "BeforeInstall": "FunctionName",
        "AfterInstall": "FunctionName",
        "AfterAllowTestTraffic": "FunctionName",
        "BeforeAllowTraffic": "FunctionName",
        "AfterAllowTraffic": "FunctionName"
      },
      "ServiceRole": "CodeDeployServiceRoleName",
      "Applications": [
        {
          "Target": {
```

```

    "Type": "AWS::ECS::Service",
    "LogicalID": "Logical ID of AWS::ECS::Service"
  },
  "ECSAttributes": {
    "TaskDefinitions": [
      "Logical ID of AWS::ECS::TaskDefinition (Blue)",
      "Logical ID of AWS::ECS::TaskDefinition (Green)"
    ],
    "TaskSets": [
      "Logical ID of AWS::ECS::TaskSet (Blue)",
      "Logical ID of AWS::ECS::TaskSet (Green)"
    ],
    "TrafficRouting": {
      "ProdTrafficRoute": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
(Production)"
      },
      "TestTrafficRoute": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
(Test)"
      },
      "TargetGroups": [
        "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Blue)",
        "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Green)"
      ]
    }
  }
}

```

プロパティ

論理 ID (論理名とも呼ばれます)

テンプレートの Hooks セクションで宣言されたフックの論理 ID。論理 ID は英数字 (A-Za-z0-9) とし、テンプレート内で一意である必要があります。

必須: はい

Type

フックのタイプ。AWS::CodeDeploy::BlueGreen

必須: はい

Properties

フックのプロパティ。

必須: はい

TrafficRoutingConfig

トラフィックルーティングの構成値。

必須: いいえ

デフォルトの構成は、時間ベースの canary トラフィックシフトで、15% のステップ率と 5 分間のベイク時間があります。

Type

デプロイ構成で使用するトラフィックシフトのタイプ。

有効値: AllAtOnce | TimeBasedCanary | TimeBasedLinear

必須: はい

TimeBasedCanary

デプロイの 1 つのバージョンから別のバージョンにトラフィックを 2 つずつシフトする構成を指定します。

必須: 条件付き: TimeBasedCanary をトラフィックルーティングタイプとして指定する場合、TimeBasedCanary パラメータを含める必要があります。

StepPercentage

TimeBasedCanary デプロイの最初の増分でシフトするトラフィックの割合。ステップの割合は 14% 以上である必要があります。

必須: いいえ

BakeTimeMins

TimeBasedCanary デプロイの 1 番目と 2 番目のトラフィックシフトの間の分数。

必須: いいえ

TimeBasedLinear

1 つのデプロイバージョンから別のデプロイバージョンにトラフィックを同じ増分で、各増分間隔を分単位でシフトする構成を指定します。

必須: 条件付き: TimeBasedLinear をトラフィックルーティングタイプとして指定する場合、TimeBasedLinear パラメータを含める必要があります。

StepPercentage

TimeBasedLinear デプロイの各増分開始時にシフトされるトラフィックの割合。ステップの割合は 14% 以上である必要があります。

必須: いいえ

BakeTimeMins

TimeBasedLinear デプロイの各増分トラフィックシフト間隔 (分)。

必須: いいえ

AdditionalOptions

ブルー/グリーンデプロイの追加オプション。

必須: いいえ

TerminationWaitTimeInMinutes

ブルーリソースを終了するまでの待機時間を分単位で指定します。

必須: いいえ

LifecycleEventHooks

ライフサイクルイベントフックを使用して、CodeDeploy がデプロイを検証するために呼び出すことができる Lambda 関数を指定します。デプロイライフサイクルイベントに対して、同じ関数または別の関数を使用することもできます。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。詳細については、「AWS CodeDeploy ユーザーガイド」の「[AppSpec の『フック』セクション](#)」を参照してください。

必須: いいえ

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する関数。

必須: いいえ

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する関数。

必須: いいえ

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する関数。

必須: いいえ

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、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する関数。

必須: いいえ

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する関数。

必須: いいえ

ServiceRole

ブルー/グリーンデプロイの実行に使用する CloudFormation の実行ロール。必要なアクセス許可のリストについては、「[ブルー/グリーンデプロイ用の IAM 許可](#)」を参照してください。

必須: いいえ

Applications

Amazon ECS アプリケーションのプロパティを指定します。

必須: はい

Target

必須: はい

Type

リソースのタイプ。

必須: はい

LogicalID

リソースの論理 ID。

必須: はい

ECSAttributes

Amazon ECS アプリケーションデプロイのさまざまな要件を表すリソース。

必須: はい

TaskDefinitions

Amazon ECS アプリケーションを含む Docker コンテナを実行するための [AWS::ECS::TaskDefinition](#) リソースの論理 ID。

必須: はい

TaskSets

アプリケーションのタスクセットとして使用する [AWS::ECS::TaskSet](#) リソースの論理 ID。

必須: はい

TrafficRouting

トラフィックルーティングに使用するリソースを指定します。

必須: はい

ProdTrafficRoute

ターゲットグループにトラフィックを指示するためのロードバランサーにより使用されるリスナー。

必須: はい

Type

リソースのタイプ。AWS::ElasticLoadBalancingV2::Listener

必須: はい

LogicalID

リソースの論理的な ID。

必須: はい

TestTrafficRoute

ターゲットグループにトラフィックを指示するためのロードバランサーにより使用されるリスナー。

必須: はい

Type

リソースのタイプ。AWS::ElasticLoadBalancingV2::Listener

必須: はい

LogicalID

リソースの論理的な ID。

必須: いいえ

TargetGroups

登録されたターゲットにトラフィックをルーティングするためのターゲットグループとして使用するリソースの論理 ID。

必須: はい

ブルー/グリーンデプロイテンプレートの例

次のサンプルテンプレートは、ECS 上に CodeDeploy ブルー/グリーンデプロイを設定し、トラフィックルーティングの進行状況をステップごとに 15 % に、各ステップ間の安定化期間を 5 分に設定します。

テンプレートを使用してスタックを作成すると、デプロイの初期構成がプロビジョニングされます。その後、リソースの置き換えを必要とする BlueTaskSet リソースのプロパティに変更を加えた場合、CloudFormation はスタック更新の一環としてグリーンデプロイを開始します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "Vpc": { "Type": "AWS::EC2::VPC::Id" },
    "Subnet1": { "Type": "AWS::EC2::Subnet::Id" },
    "Subnet2": { "Type": "AWS::EC2::Subnet::Id" }
  },
  "Transform": [ "AWS::CodeDeployBlueGreen" ],
  "Hooks": {
    "CodeDeployBlueGreenHook": {
      "Type": "AWS::CodeDeploy::BlueGreen",
      "Properties": {
        "TrafficRoutingConfig": {
          "Type": "TimeBasedCanary",
          "TimeBasedCanary": {
            "StepPercentage": 15,
            "BakeTimeMins": 5
          }
        }
      }
    },
    "Applications": [
      {
        "Target": {
          "Type": "AWS::ECS::Service",
          "LogicalID": "ECSDemoService"
        },
        "ECSAttributes": {
          "TaskDefinitions": [ "BlueTaskDefinition", "GreenTaskDefinition" ],
          "TaskSets": [ "BlueTaskSet", "GreenTaskSet" ],
          "TrafficRouting": {
            "ProdTrafficRoute": {
              "Type": "AWS::ElasticLoadBalancingV2::Listener",
              "LogicalID": "ALBListenerProdTraffic"
            },
            "TargetGroups": [ "ALBTargetGroupBlue", "ALBTargetGroupGreen" ]
          }
        }
      }
    ]
  }
}
```

```
    ]
  }
}
},
"Resources":{
  "ExampleSecurityGroup":{
    "Type":"AWS::EC2::SecurityGroup",
    "Properties":{
      "GroupDescription":"Security group for ec2 access",
      "VpcId":{"Ref":"Vpc" },
      "SecurityGroupIngress":[
        {
          "IpProtocol":"tcp",
          "FromPort":80,
          "ToPort":80,
          "CidrIp":"0.0.0.0/0"
        },
        {
          "IpProtocol":"tcp",
          "FromPort":8080,
          "ToPort":8080,
          "CidrIp":"0.0.0.0/0"
        },
        {
          "IpProtocol":"tcp",
          "FromPort":22,
          "ToPort":22,
          "CidrIp":"0.0.0.0/0"
        }
      ]
    }
  }
},
"ALBTargetGroupBlue":{
  "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
  "Properties":{
    "HealthCheckIntervalSeconds":5,
    "HealthCheckPath":"/",
    "HealthCheckPort":"80",
    "HealthCheckProtocol":"HTTP",
    "HealthCheckTimeoutSeconds":2,
    "HealthyThresholdCount":2,
    "Matcher":{"HttpCode":"200" },
    "Port":80,
    "Protocol":"HTTP",
```

```

    "Tags": [{ "Key": "Group", "Value": "Example" }],
    "TargetType": "ip",
    "UnhealthyThresholdCount": 4,
    "VpcId": { "Ref": "Vpc" }
  }
},
"ALBTargetGroupGreen": {
  "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
  "Properties": {
    "HealthCheckIntervalSeconds": 5,
    "HealthCheckPath": "/",
    "HealthCheckPort": "80",
    "HealthCheckProtocol": "HTTP",
    "HealthCheckTimeoutSeconds": 2,
    "HealthyThresholdCount": 2,
    "Matcher": { "HttpCode": "200" },
    "Port": 80,
    "Protocol": "HTTP",
    "Tags": [{ "Key": "Group", "Value": "Example" }],
    "TargetType": "ip",
    "UnhealthyThresholdCount": 4,
    "VpcId": { "Ref": "Vpc" }
  }
},
"ExampleALB": {
  "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
  "Properties": {
    "Scheme": "internet-facing",
    "SecurityGroups": [{ "Ref": "ExampleSecurityGroup" }],
    "Subnets": [{ "Ref": "Subnet1" }, { "Ref": "Subnet2" }],
    "Tags": [{ "Key": "Group", "Value": "Example" }],
    "Type": "application",
    "IpAddressType": "ipv4"
  }
},
"ALBListenerProdTraffic": {
  "Type": "AWS::ElasticLoadBalancingV2::Listener",
  "Properties": {
    "DefaultActions": [
      {
        "Type": "forward",
        "ForwardConfig": {
          "TargetGroups": [

```

```

        "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
        "Weight":1
    }
]
}
],
"LoadBalancerArn":{ "Ref":"ExampleALB" },
"Port":80,
"Protocol":"HTTP"
}
},
"ALBListenerProdRule":{
    "Type":"AWS::ElasticLoadBalancingV2::ListenerRule",
    "Properties":{
        "Actions":[
            {
                "Type":"forward",
                "ForwardConfig":{
                    "TargetGroups":[
                        {
                            "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
                            "Weight":1
                        }
                    ]
                }
            }
        ],
        "Conditions":[
            {
                "Field":"http-header",
                "HttpHeaderConfig":{
                    "HttpHeaderName":"User-Agent",
                    "Values":[ "Mozilla" ]
                }
            }
        ],
        "ListenerArn":{ "Ref":"ALBListenerProdTraffic" },
        "Priority":1
    }
},
"ECSTaskExecutionRole":{
    "Type":"AWS::IAM::Role",
    "Properties":{

```

```
"AssumeRolePolicyDocument":{
  "Version": "2012-10-17",
  "Statement":[
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
},
"ManagedPolicyArns": [ "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" ]
},
"BlueTaskDefinition":{
  "Type": "AWS::ECS::TaskDefinition",
  "Properties": {
    "ExecutionRoleArn": {
      "Fn::GetAtt": [ "ECSTaskExecutionRole", "Arn" ]
    },
    "ContainerDefinitions": [
      {
        "Name": "DemoApp",
        "Image": "nginxdemos/hello:latest",
        "Essential": true,
        "PortMappings": [
          {
            "HostPort": 80,
            "Protocol": "tcp",
            "ContainerPort": 80
          }
        ]
      }
    ]
  }
},
"RequiresCompatibilities": [ "FARGATE" ],
"NetworkMode": "awsvpc",
"Cpu": "256",
"Memory": "512",
"Family": "ecs-demo"
},
},
```



```
"ECSDemoCluster":{
  "Type":"AWS::ECS::Cluster",
  "Properties":{}
},
"ECSDemoService":{
  "Type":"AWS::ECS::Service",
  "Properties":{
    "Cluster":{"Ref":"ECSDemoCluster" },
    "DesiredCount":1,
    "DeploymentController":{"Type":"EXTERNAL" }
  }
},
"BlueTaskSet":{
  "Type":"AWS::ECS::TaskSet",
  "Properties":{
    "Cluster":{"Ref":"ECSDemoCluster" },
    "LaunchType":"FARGATE",
    "NetworkConfiguration":{
      "AwsVpcConfiguration":{
        "AssignPublicIp":"ENABLED",
        "SecurityGroups":[{ "Ref":"ExampleSecurityGroup" }],
        "Subnets":[{ "Ref":"Subnet1" },{ "Ref":"Subnet2" }]
      }
    },
    "PlatformVersion":"1.4.0",
    "Scale":{
      "Unit":"PERCENT",
      "Value":100
    },
    "Service":{"Ref":"ECSDemoService"},
    "TaskDefinition":{"Ref":"BlueTaskDefinition" },
    "LoadBalancers":[
      {
        "ContainerName":"DemoApp",
        "ContainerPort":80,
        "TargetGroupArn":{"Ref":"ALBTargetGroupBlue" }
      }
    ]
  }
},
"PrimaryTaskSet":{
  "Type":"AWS::ECS::PrimaryTaskSet",
  "Properties":{
    "Cluster":{"Ref":"ECSDemoCluster" },
```

```

        "Service":{ "Ref":"ECSDemoService" },
        "TaskSetId":{ "Fn::GetAtt":[ "BlueTaskSet","Id" ]
        }
    }
}
}
}
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Parameters:
  Vpc:
    Type: AWS::EC2::VPC::Id
  Subnet1:
    Type: AWS::EC2::Subnet::Id
  Subnet2:
    Type: AWS::EC2::Subnet::Id
Transform:
  - 'AWS::CodeDeployBlueGreen'
Hooks:
  CodeDeployBlueGreenHook:
    Type: AWS::CodeDeploy::BlueGreen
    Properties:
      TrafficRoutingConfig:
        Type: TimeBasedCanary
      TimeBasedCanary:
        StepPercentage: 15
        BakeTimeMins: 5
    Applications:
      - Target:
          Type: AWS::ECS::Service
          LogicalID: ECSDemoService
        ECSAttributes:
          TaskDefinitions:
            - BlueTaskDefinition
            - GreenTaskDefinition
          TaskSets:
            - BlueTaskSet
            - GreenTaskSet
          TrafficRouting:
            ProdTrafficRoute:
              Type: AWS::ElasticLoadBalancingV2::Listener

```

```
        LogicalID: ALBListenerProdTraffic
    TargetGroups:
      - ALBTargetGroupBlue
      - ALBTargetGroupGreen
Resources:
  ExampleSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ec2 access
      VpcId: !Ref Vpc
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 8080
          ToPort: 8080
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  ALBTargetGroupBlue:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 5
      HealthCheckPath: /
      HealthCheckPort: '80'
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 2
      HealthyThresholdCount: 2
      Matcher:
        HttpCode: '200'
      Port: 80
      Protocol: HTTP
      Tags:
        - Key: Group
          Value: Example
      TargetType: ip
      UnhealthyThresholdCount: 4
      VpcId: !Ref Vpc
  ALBTargetGroupGreen:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
```

Properties:

```
HealthCheckIntervalSeconds: 5
HealthCheckPath: /
HealthCheckPort: '80'
HealthCheckProtocol: HTTP
HealthCheckTimeoutSeconds: 2
HealthyThresholdCount: 2
Matcher:
  HttpCode: '200'
Port: 80
Protocol: HTTP
Tags:
  - Key: Group
    Value: Example
TargetType: ip
UnhealthyThresholdCount: 4
VpcId: !Ref Vpc
```

ExampleALB:

```
Type: AWS::ElasticLoadBalancingV2::LoadBalancer
```

Properties:

```
Scheme: internet-facing
SecurityGroups:
  - !Ref ExampleSecurityGroup
Subnets:
  - !Ref Subnet1
  - !Ref Subnet2
Tags:
  - Key: Group
    Value: Example
Type: application
IpAddressType: ipv4
```

ALBListenerProdTraffic:

```
Type: AWS::ElasticLoadBalancingV2::Listener
```

Properties:

```
DefaultActions:
  - Type: forward
    ForwardConfig:
      TargetGroups:
        - TargetGroupArn: !Ref ALBTargetGroupBlue
          Weight: 1
LoadBalancerArn: !Ref ExampleALB
Port: 80
Protocol: HTTP
```

ALBListenerProdRule:

```
Type: AWS::ElasticLoadBalancingV2::ListenerRule
Properties:
  Actions:
    - Type: forward
      ForwardConfig:
        TargetGroups:
          - TargetGroupArn: !Ref ALBTargetGroupBlue
            Weight: 1
  Conditions:
    - Field: http-header
      HttpHeaderConfig:
        HttpHeaderName: User-Agent
        Values:
          - Mozilla
      ListenerArn: !Ref ALBListenerProdTraffic
      Priority: 1
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'
BlueTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    ExecutionRoleArn: !GetAtt
      - ECSTaskExecutionRole
      - Arn
    ContainerDefinitions:
      - Name: DemoApp
        Image: 'nginxdemos/hello:latest'
        Essential: true
        PortMappings:
          - HostPort: 80
            Protocol: tcp
            ContainerPort: 80
    RequiresCompatibilities:
```

```
- FARGATE
NetworkMode: awsvpc
Cpu: '256'
Memory: '512'
Family: ecs-demo
ECSDemoCluster:
  Type: AWS::ECS::Cluster
  Properties: {}
ECSDemoService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref ECSDemoCluster
    DesiredCount: 1
    DeploymentController:
      Type: EXTERNAL
BlueTaskSet:
  Type: AWS::ECS::TaskSet
  Properties:
    Cluster: !Ref ECSDemoCluster
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsVpcConfiguration:
        AssignPublicIp: ENABLED
      SecurityGroups:
        - !Ref ExampleSecurityGroup
      Subnets:
        - !Ref Subnet1
        - !Ref Subnet2
    PlatformVersion: 1.4.0
    Scale:
      Unit: PERCENT
      Value: 100
    Service: !Ref ECSDemoService
    TaskDefinition: !Ref BlueTaskDefinition
    LoadBalancers:
      - ContainerName: DemoApp
        ContainerPort: 80
        TargetGroupArn: !Ref ALBTargetGroupBlue
PrimaryTaskSet:
  Type: AWS::ECS::PrimaryTaskSet
  Properties:
    Cluster: !Ref ECSDemoCluster
    Service: !Ref ECSDemoService
    TaskSetId: !GetAtt
```

- BlueTaskSet
- Id

CloudFormation テンプレートスニペット

このセクションでは、CloudFormation テンプレートの各種構成要素の宣言方法を示した多数のサンプルシナリオを紹介しています。カスタムテンプレートのセクションを独自に作成する際の参考にしてください。

Note

CloudFormation テンプレートは、JSON に準拠する関係上、行連結文字に対応していません。このドキュメントのスニペットでは、行の長さが 68 文字を超えた場合に適宜、改行しています。

トピック

- [一般的なテンプレートスニペット](#)
- [Auto Scaling CloudFormation テンプレートスニペット](#)
- [AWS 請求コンソールのテンプレートスニペット](#)
- [CloudFormation テンプレートスニペット](#)
- [Amazon CloudFront テンプレートスニペット](#)
- [Amazon CloudWatch テンプレートスニペット](#)
- [Amazon CloudWatch Logs テンプレートスニペット](#)
- [Amazon DynamoDB テンプレートスニペット](#)
- [Amazon EC2 CloudFormation テンプレートスニペット](#)
- [Amazon Elastic Container Service サンプルテンプレート](#)
- [Amazon Elastic File System サンプルテンプレート](#)
- [Elastic Beanstalk テンプレートスニペット](#)
- [Elastic Load Balancing テンプレートスニペット](#)
- [AWS Identity and Access Management テンプレートスニペット](#)
- [AWS Lambda テンプレート](#)
- [Amazon Redshift テンプレートスニペット](#)
- [Amazon RDS テンプレートスニペット](#)

- [Route 53 テンプレートスニペット](#)
- [Amazon S3 テンプレートスニペット](#)
- [Amazon SNS テンプレートスニペット](#)
- [Amazon SQS テンプレートスニペット](#)
- [Amazon Timestream テンプレートスニペット](#)

一般的なテンプレートスニペット

次の例では、AWS サービスに固有ではない異なる CloudFormation テンプレートの特徴を示しています。

トピック

- [Base64 エンコードされた UserData プロパティ](#)
- [AccessKey および SecretKey で Base64 エンコードされた UserData プロパティ](#)
- [Parameters セクション \(リテラル文字列パラメータが 1 つ存在\)](#)
- [Parameters セクション \(正規表現制約を設定した文字列パラメータが存在\)](#)
- [Parameters セクション \(MinValue および MaxValue 制約を設定した数値パラメータが存在\)](#)
- [Parameters セクション \(AllowedValues 制約を設定した数値パラメータが存在\)](#)
- [Parameters セクション \(1 つのリテラル CommaDelimitedList パラメータが存在\)](#)
- [Parameters セクション \(擬似パラメータに基づくパラメータ値が存在\)](#)
- [Mapping セクション \(3 つのマッピングが存在\)](#)
- [リテラル文字列を使用した Description](#)
- [Outputs セクション \(リテラル文字列の出力が 1 つ存在\)](#)
- [Outputs セクション \(リソースリファレンスが 1 つ、擬似リファレンス出力が 1 つ存在\)](#)
- [Outputs セクション \(関数、リテラル文字列、リファレンス、擬似パラメータによる出力\)](#)
- [テンプレート形式バージョン](#)
- [AWS Tags プロパティ](#)

Base64 エンコードされた UserData プロパティ

この例では、Fn::Base64 関数と Fn::Join 関数を使用して UserData プロパティを作成しています。MyValue と MyName のリファレンスはパラメータです。これらのパラメータは、テンプレート

の Parameters セクションで定義されている必要があります。それらとは別に、リテラル文字列の Hello World も、UserData の一部として渡しています。

JSON

```
"UserData" : {
  "Fn::Base64" : {
    "Fn::Join" : [ ",", [
      { "Ref" : "MyValue" },
      { "Ref" : "MyName" },
      "Hello World" ] ]
  }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
    Ref: MyValue
    Ref: MyName
    Hello World
```

AccessKey および SecretKey で Base64 エンコードされた UserData プロパティ

この例では、Fn::Base64 関数と Fn::Join 関数を使用して UserData プロパティを作成しています。このプロパティには、AccessKey と SecretKey の情報が含まれています。AccessKey と SecretKey のリファレンスはパラメーターです。これらのパラメーターは、テンプレートの Parameters セクションで定義されている必要があります。

JSON

```
"UserData" : {
  "Fn::Base64" : {
    "Fn::Join" : [ "", [
      "ACCESS_KEY=", { "Ref" : "AccessKey" },
      "SECRET_KEY=", { "Ref" : "SecretKey" } ]
    ]
  }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
    ACCESS_KEY=${AccessKey}
    SECRET_KEY=${SecretKey}
```

Parameters セクション (リテラル文字列パラメータが 1 つ存在)

次の例は、String 型パラメーターが 1 つだけ宣言された有効な Parameters セクションの宣言を表しています。

JSON

```
"Parameters" : {
  "UserName" : {
    "Type" : "String",
    "Default" : "nonadmin",
    "Description" : "Assume a vanilla user if no command-line spec provided"
  }
}
```

YAML

```
Parameters:
  UserName:
    Type: String
    Default: nonadmin
    Description: Assume a vanilla user if no command-line spec provided
```

Parameters セクション (正規表現制約を設定した文字列パラメータが存在)

次の例は、String 型パラメーターが 1 つだけ宣言された有効な Parameters セクションの宣言を表しています。AdminUserAccount パラメータのデフォルトは admin です。このパラメータの値は、長さが 1~16 で、かつアルファベット文字と数字で構成されている必要があります。また、先頭の文字はアルファベット文字である必要があります。

JSON

```
"Parameters" : {
```

```
"AdminUserAccount": {
  "Default": "admin",
  "NoEcho": "true",
  "Description" : "The admin account user name",
  "Type": "String",
  "MinLength": "1",
  "MaxLength": "16",
  "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
}
```

YAML

```
Parameters:
  AdminUserAccount:
    Default: admin
    NoEcho: true
    Description: The admin account user name
    Type: String
    MinLength: 1
    MaxLength: 16
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
```

Parameters セクション (MinValue および MaxValue 制約を設定した数値パラメータが存在)

次の例は、Number 型パラメーターが 1 つだけ宣言された有効な Parameters セクションの宣言を表しています。WebServerPort パラメータはデフォルト値が 80 で、最小値が 1、最大値が 65535 です。

JSON

```
"Parameters" : {
  "WebServerPort": {
    "Default": "80",
    "Description" : "TCP/IP port for the web server",
    "Type": "Number",
    "MinValue": "1",
    "MaxValue": "65535"
  }
}
```

YAML

```
Parameters:
  WebServerPort:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    MinValue: 1
    MaxValue: 65535
```

Parameters セクション (AllowedValues 制約を設定した数値パラメータが存在)

次の例は、Number 型パラメーターが 1 つだけ宣言された有効な Parameters セクションの宣言を表しています。WebServerPort パラメータは、デフォルト値が 80 で、許容される値は 80 と 8888 だけです。

JSON

```
"Parameters" : {
  "WebServerPortLimited": {
    "Default": "80",
    "Description" : "TCP/IP port for the web server",
    "Type": "Number",
    "AllowedValues" : ["80", "8888"]
  }
}
```

YAML

```
Parameters:
  WebServerPortLimited:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    AllowedValues:
      - 80
      - 8888
```

Parameters セクション (1 つのリテラル CommaDelimitedList パラメータが存在)

次の例は、CommaDelimitedList タイプパラメーターが 1 つだけ宣言された有効な Parameters セクションの宣言を表しています。NoEcho プロパティは TRUE に設定されます。この場合、その値は

describe-stacks 出力でアスタリスク (*****) によってマスクされます。ただし、以下に指定した場所に保存されている情報は除きます。

Important

NoEcho 属性を使用しても、以下に保存されている情報はマスクされません。

- Metadata テンプレートセクション。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。詳細については、「[メタデータ](#)」を参照してください。
- Outputs テンプレートセクション。詳細については、「[出力](#)」を参照してください。
- リソース定義の Metadata 属性。詳細については、「[Metadata 属性](#)」を参照してください。

パスワードやシークレットなどの機密情報を含めるには、これらのメカニズムを使用しないことを強くお勧めします。

Important

機密情報は、CloudFormation テンプレートに直接埋め込むのではなく、スタックテンプレートの動的パラメータを使用して CloudFormation の外部 (AWS Systems Manager パラメータストアや AWS Secrets Manager など) に保存して管理した上で参照することをお勧めします。

詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティスを参照してください。

JSON

```
"Parameters" : {  
  "UserRoles" : {  
    "Type" : "CommaDelimitedList",  
    "Default" : "guest,newhire",  
    "NoEcho" : "TRUE"  
  }  
}
```

YAML

```
Parameters:
  UserRoles:
    Type: CommaDelimitedList
    Default: "guest,newhire"
    NoEcho: true
```

Parameters セクション (擬似パラメータに基づくパラメータ値が存在)

次の例は、擬似パラメーター `AWS::StackName` および `AWS::Region` を使用する EC2 ユーザーデータのコマンドです。擬似 Parameters の詳細については、[「擬似パラメータを使用して AWS 値を取得する」](#) を参照してください。

JSON

```
"UserData"      : { "Fn::Base64" : { "Fn::Join" : ["", [
  "#!/bin/bash -xe\n",
  "yum install -y aws-cfn-bootstrap\n",

  "/opt/aws/bin/cfn-init -v ",
  "      --stack ", { "Ref" : "AWS::StackName" },
  "      --resource LaunchConfig ",
  "      --region ", { "Ref" : "AWS::Region" }, "\n",

  "/opt/aws/bin/cfn-signal -e $? ",
  "      --stack ", { "Ref" : "AWS::StackName" },
  "      --resource WebServerGroup ",
  "      --region ", { "Ref" : "AWS::Region" }, "\n"
]]}}
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash -xe
    yum update -y aws-cfn-bootstrap
    /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig --
region ${AWS::Region}
    /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServerGroup
--region ${AWS::Region}
```

Mapping セクション (3 つのマッピングが存在)

次の例は、3 つのマッピングを含んだ有効な Mapping セクションの宣言を表しています。このマップは、Stop、SlowDown、Go の各マッピングキーと比較され、対応する RGBColor 属性に割り当てられた RGB 値を返します。

JSON

```
"Mappings" : {
  "LightColor" : {
    "Stop" : {
      "Description" : "red",
      "RGBColor" : "RED 255 GREEN 0 BLUE 0"
    },
    "SlowDown" : {
      "Description" : "yellow",
      "RGBColor" : "RED 255 GREEN 255 BLUE 0"
    },
    "Go" : {
      "Description" : "green",
      "RGBColor" : "RED 0 GREEN 128 BLUE 0"
    }
  }
}
```

YAML

```
Mappings:
  LightColor:
    Stop:
      Description: red
      RGBColor: "RED 255 GREEN 0 BLUE 0"
    SlowDown:
      Description: yellow
      RGBColor: "RED 255 GREEN 255 BLUE 0"
    Go:
      Description: green
      RGBColor: "RED 0 GREEN 128 BLUE 0"
```

リテラル文字列を使用した Description

次の例は、リテラル文字列を値に使用した有効な Description セクションの宣言を表しています。このスニペットは、テンプレート、パラメーター、リソース、プロパティ、出力に使用できません。

JSON

```
"Description" : "Replace this value"
```

YAML

```
Description: "Replace this value"
```

Outputs セクション (リテラル文字列の出力が 1 つ存在)

この例では、リテラル文字列によって出力の割り当てを行っています。

JSON

```
"Outputs" : {  
  "MyPhone" : {  
    "Value" : "Please call 555-5555",  
    "Description" : "A random message for aws cloudformation describe-stacks"  
  }  
}
```

YAML

```
Outputs:  
  MyPhone:  
    Value: Please call 555-5555  
    Description: A random message for aws cloudformation describe-stacks
```

Outputs セクション (リソースリファレンスが 1 つ、擬似リファレンス出力が 1 つ存在)

この例は、出力の割り当てが 2 つ存在する Outputs セクションを示します。1 つはリソースに、もう 1 つは擬似リファレンスに基づいて出力を割り当てています。

JSON

```
"Outputs" : {  
  "SNSTopic" : { "Value" : { "Ref" : "MyNotificationTopic" } },  
  "StackName" : { "Value" : { "Ref" : "AWS::StackName" } }  
}
```

YAML

```
Outputs:  
  SNSTopic:  
    Value: !Ref MyNotificationTopic  
  StackName:  
    Value: !Ref AWS::StackName
```

Outputs セクション (関数、リテラル文字列、リファレンス、擬似パラメータによる出力)

この例は、出力の割り当てが 1 つ存在する Outputs セクションを示します。Join 関数を使用し、パーセント記号を区切り文字に使用して値を連結しています。

JSON

```
"Outputs" : {  
  "MyOutput" : {  
    "Value" : { "Fn::Join" :  
      [ "%", [ "A-string", { "Ref" : "AWS::StackName" } ] ]  
    }  
  }  
}
```

YAML

```
Outputs:  
  MyOutput:  
    Value: !Join [ %, [ 'A-string', !Ref 'AWS::StackName' ] ]
```

テンプレート形式バージョン

次のスニペットは、有効な AWSTemplateFormatVersion セクションの宣言を表しています。

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

AWS Tags プロパティ

この例は、AWS Tags プロパティを示します。このプロパティは、リソースの Properties セクションに指定します。作成されたリソースは、宣言したタグでタグ付けされます。

JSON

```
"Tags" : [  
  {  
    "Key" : "keyname1",  
    "Value" : "value1"  
  },  
  {  
    "Key" : "keyname2",  
    "Value" : "value2"  
  }  
]
```

YAML

```
Tags:  
-  
  Key: "keyname1"  
  Value: "value1"  
-  
  Key: "keyname2"  
  Value: "value2"
```

Auto Scaling CloudFormation テンプレートスニペット

Amazon EC2 Auto Scaling を使用すると、スケーリングポリシーまたはスケジュールされたスケーリングを使用して、Amazon EC2 インスタンスを自動的にスケールすることができます。Auto

Scaling グループは、スケーリングポリシー、スケジュールされたアクション、ヘルスチェック、ライフサイクルフック、ロードバランシングなど、スケーリングとフリートの自動管理機能が有効化された Amazon EC2 インスタンスの集合です。

アプリケーションの Auto Scaling では、スケーリングポリシーまたはスケジュールに基づくスケーリングを使用して、Amazon EC2 の対象外のさまざまなリソースを自動的にスケーリングできます。

CloudFormation テンプレートを使用すると、Auto Scaling グループ、スケーリングポリシー、スケジュールされたアクション、その他自動スケーリングのリソースをインフラストラクチャの一環として作成および設定することができます。テンプレートを使用すると、自動スケーリングリソースのデプロイを反復的にかつ一貫性のある方法で、簡単に管理し自動化することができます。

以下のテンプレートスニペットの例では、Amazon EC2 Auto Scaling と Application Auto Scaling の、CloudFormation リソースまたはコンポーネントについて説明しています。これらのスニペットはテンプレートに統合するように設計されており、単独で実行することを意図したものではありません。

スニペットカテゴリ

- [CloudFormation で Amazon EC2 Auto Scaling リソースを設定する](#)
- [CloudFormation で Application Auto Scaling リソースを設定する](#)

CloudFormation で Amazon EC2 Auto Scaling リソースを設定する

以下の例では、Amazon EC2 Auto Scaling で使用するテンプレートに含めるさまざまなスニペットを示しています。

スニペットカテゴリ

- [単一インスタンスの Auto Scaling グループを作成する](#)
- [ロードバランサーがアタッチされた Auto Scaling グループを作成する](#)
- [通知を使用する Auto Scaling グループを作成する](#)
- [CreationPolicy と UpdatePolicy を使用する Auto Scaling グループを作成する](#)
- [ステップスケーリングポリシーを作成する](#)
- [混合インスタンスグループの例](#)
- [起動構成の例](#)

単一インスタンスの Auto Scaling グループを作成する

この例に示すのは、作業を進める際に役立つ、単一インスタンスに関連付けられた [AWS::AutoScaling::AutoScalingGroup](#) リソースです。Auto Scaling グループの `VPCZoneIdentifier` プロパティは、3 つの異なるアベイラビリティゾーンの既存のサブネットのリストを指定します。スタックを作成する前に、アカウントから該当するサブネット ID を指定する必要があります。LaunchTemplate プロパティは、テンプレートの他の場所で定義されている論理名 `myLaunchTemplate` の [AWS::EC2::LaunchTemplate](#) リソースを参照します。

Note

起動テンプレートの例については、Amazon EC2 スニペットの「[CloudFormation を使用して起動テンプレートを作成する](#)」、および「AWS::EC2::LaunchTemplate リソース」の「[Examples](#)」のセクションを参照してください。

JSON

```
"myASG" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
    "LaunchTemplate" : {
      "LaunchTemplateId" : {
        "Ref" : "myLaunchTemplate"
      },
      "Version" : {
        "Fn::GetAtt" : [
          "myLaunchTemplate",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize" : "1",
    "MinSize" : "1"
  }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '1'
    MinSize: '1'
```

ロードバランサーがアタッチされた Auto Scaling グループを作成する

この例に示すのは、複数のサーバー間で負荷を分散する

[AWS::AutoScaling::AutoScalingGroup](#) リソースです。また、同じテンプレートの他の場所で宣言された AWS リソースの論理名を指定します。

1. VPCZoneIdentifier プロパティは、Auto Scaling グループの EC2 インスタンスが作成される 2 つの [AWS::EC2::Subnet](#) リソースの論理名を、次のように指定します:
myPublicSubnet1、myPublicSubnet2
2. LaunchTemplate プロパティは、論理名が myLaunchTemplate である [AWS::EC2::LaunchTemplate](#) リソースを指定します。
3. TargetGroupARNs プロパティは、トラフィックを Auto Scaling グループにルーティングするために使用される Application Load Balancer または Network Load Balancer のターゲットグループのリストを指定します。この例では、1 つのターゲットグループとして、論理名が myTargetGroup である [AWS::ElasticLoadBalancingV2::TargetGroup](#) リソースを指定しています。

JSON

```
"myServerGroup" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "VPCZoneIdentifier" : [ { "Ref" : "myPublicSubnet1" }, { "Ref" :
      "myPublicSubnet2" } ],
```

```

    "LaunchTemplate" : {
      "LaunchTemplateId" : {
        "Ref" : "myLaunchTemplate"
      },
      "Version" : {
        "Fn::GetAtt" : [
          "myLaunchTemplate",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize" : "5",
    "MinSize" : "1",
    "TargetGroupARNs" : [ { "Ref" : "myTargetGroup" } ]
  }
}

```

YAML

```

myServerGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref myPublicSubnet1
      - !Ref myPublicSubnet2
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    TargetGroupARNs:
      - !Ref myTargetGroup

```

関連情報

Application Load Balancer の ALBRequestCountPerTarget 事前定義メトリクスに基づくターゲット追跡スケーリングポリシーを関連付けた Auto Scaling グループを作成する詳細な例については、「AWS::AutoScaling::ScalingPolicy リソース」の「[例](#)」セクションを参照してください。

通知を使用する Auto Scaling グループを作成する

リソースタイプのプライマリ ID は テンプレートにある必要があります。NotificationConfigurations プロパティは、CloudFormation からの通知の送信

先となる SNS トピックと、CloudFormation が通知を送信するきっかけとなるイベントを指定します。NotificationTypes で指定したイベントが発生すると、CloudFormation から TopicARN で指定した SNS トピックに通知が送信されます。スタックを起動すると、CloudFormation は、同じテンプレート内で宣言された [AWS::SNS::Subscription](#) リソース (snsTopicForAutoScalingGroup) を作成します。

Auto Scaling グループの VPCZoneIdentifier プロパティは、3 つの異なるアベイラビリティーゾーンの既存のサブネットのリストを指定します。スタックを作成する前に、アカウントから該当するサブネット ID を指定する必要があります。LaunchTemplate プロパティは、同じテンプレート内の別の場所で宣言された [AWS::EC2::LaunchTemplate](#) リソースの論理名を参照します。

JSON

```
"myASG" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "DependsOn": [
    "snsTopicForAutoScalingGroup"
  ],
  "Properties" : {
    "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
    "LaunchTemplate" : {
      "LaunchTemplateId" : {
        "Ref" : "logicalName"
      },
      "Version" : {
        "Fn::GetAtt" : [
          "logicalName",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize" : "5",
    "MinSize" : "1",
    "NotificationConfigurations" : [
      {
        "TopicARN" : { "Ref" : "snsTopicForAutoScalingGroup" },
        "NotificationTypes" : [
          "autoscaling:EC2_INSTANCE_LAUNCH",
          "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
          "autoscaling:EC2_INSTANCE_TERMINATE",
          "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
          "autoscaling:TEST_NOTIFICATION"
        ]
      }
    ]
  }
}
```

```

    ]
  }
]
}
}

```

YAML

```

myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn:
    - snsTopicForAutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref logicalName
      Version: !GetAtt logicalName.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    NotificationConfigurations:
      - TopicARN: !Ref snsTopicForAutoScalingGroup
        NotificationTypes:
          - autoscaling:EC2_INSTANCE_LAUNCH
          - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
          - autoscaling:EC2_INSTANCE_TERMINATE
          - autoscaling:EC2_INSTANCE_TERMINATE_ERROR
          - autoscaling:TEST_NOTIFICATION

```

CreationPolicy と UpdatePolicy を使用する Auto Scaling グループを作成する

次の例では、[AWS::AutoScaling::AutoScalingGroup](#) リソースに CreationPolicy と UpdatePolicy の属性を追加する方法を示しています。

サンプルの作成ポリシーは、グループの準備ができたときに CloudFormation が Count の数の成功シグナルを受信するまで、Auto Scaling グループが CREATE_COMPLETE のステータスに到達するのを防ぎます。Auto Scaling グループの準備ができたことを通知するために、起動テンプレートのユーザーデータ (図に示されていません) に追加された cfn-signal ヘルパースクリプトが、インスタンス上で実行されます。インスタンスが指定された Timeout 範囲内でシグナルを送信しない

場合では、CloudFormation はインスタンスが作成されていないと想定し、リソースの作成は失敗し、CloudFormation はスタックをロールバックします。

サンプルの更新ポリシーでは、AutoScalingRollingUpdate プロパティを使用して、ローリング更新を実行するように CloudFormation に指示します。ローリング更新は、MaxBatchSize に基づいて小規模なバッチ (この例ではインスタンスごと) で Auto Scaling グループを変更し、PauseTime に基づいて更新のバッチ間の停止時間を変更します。MinInstancesInService 属性は、Auto Scaling が古いインスタンスを更新している間、CloudFormation グループ内で使用中となる必要があるインスタンスの最小数を指定します。

WaitOnResourceSignals 属性は true に設定しています。CloudFormation が更新を続行するには、指定された PauseTime 内に新しいインスタンスからシグナルを受け取る必要があります。スタックの更新の進行中、EC2 Auto Scaling プロセス HealthCheck、ReplaceUnhealthy、AZRebalance、AlarmNotification、ScheduledActions は一時停止されます。注: ローリング更新が適切に機能しなくなる可能性があるため、Launch、Terminate、または AddToLoadBalancer (Auto Scaling グループが Elastic Load Balancing で使用されている場合) タイプのプロセスを一時停止しないでください。

Auto Scaling グループの VPCZoneIdentifier プロパティは、3 つの異なるアベイラビリティーゾーンの既存のサブネットのリストを指定します。スタックを作成する前に、アカウントから該当するサブネット ID を指定する必要があります。LaunchTemplate プロパティは、同じテンプレート内の別の場所で宣言された [AWS::EC2::LaunchTemplate](#) リソースの論理名を参照します。

CreationPolicy 属性および UpdatePolicy 属性の詳細については、「[リソース属性リファレンス](#)」を参照してください。

JSON

```
{
  "Resources": {
    "myASG": {
      "CreationPolicy": {
        "ResourceSignal": {
          "Count": "3",
          "Timeout": "PT15M"
        }
      },
      "UpdatePolicy": {
        "AutoScalingRollingUpdate": {
          "MinInstancesInService": "3",
          "MaxBatchSize": "1",
```

```

        "PauseTime": "PT12M5S",
        "WaitOnResourceSignals": "true",
        "SuspendProcesses": [
            "HealthCheck",
            "ReplaceUnhealthy",
            "AZRebalance",
            "AlarmNotification",
            "ScheduledActions",
            "InstanceRefresh"
        ]
    },
    "Type": "AWS::AutoScaling::AutoScalingGroup",
    "Properties": {
        "VPCZoneIdentifier": [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
        "LaunchTemplate": {
            "LaunchTemplateId": {
                "Ref": "logicalName"
            },
            "Version": {
                "Fn::GetAtt": [
                    "logicalName",
                    "LatestVersionNumber"
                ]
            }
        },
        "MaxSize": "5",
        "MinSize": "3"
    }
}
}
}
}

```

YAML

```

---
Resources:
  myASG:
    CreationPolicy:
      ResourceSignal:
        Count: '3'
        Timeout: PT15M
    UpdatePolicy:

```

```
AutoScalingRollingUpdate:
  MinInstancesInService: '3'
  MaxBatchSize: '1'
  PauseTime: PT12M5S
  WaitOnResourceSignals: true
  SuspendProcesses:
    - HealthCheck
    - ReplaceUnhealthy
    - AZRebalance
    - AlarmNotification
    - ScheduledActions
    - InstanceRefresh
Type: AWS::AutoScaling::AutoScalingGroup
Properties:
  VPCZoneIdentifier:
    - subnetIdAz1
    - subnetIdAz2
    - subnetIdAz3
  LaunchTemplate:
    LaunchTemplateId: !Ref logicalName
    Version: !GetAtt logicalName.LatestVersionNumber
  MaxSize: '5'
  MinSize: '3'
```

ステップスケーリングポリシーを作成する

この例に示すのは、ステップスケーリングポリシーを使用して Auto Scaling グループをスケールアウトする [AWS::AutoScaling::ScalingPolicy](#) リソースです。AdjustmentType プロパティには、ChangeInCapacity が指定されています。この場合、ScalingAdjustment は、追加 (ScalingAdjustment が正数の場合) または削除 (負数の場合) するインスタンスの数を表します。この例では、ScalingAdjustment は 1 です。したがって、アラームのしきい値に対して違反があると、ポリシーが実行されて、グループ内の EC2 インスタンスの数が 1 つ増やされます。

[AWS::CloudWatch::Alarm](#) リソース CPUAlarmHigh は、アラームが ALARM 状態 (AlarmActions) になったときに実行するアクションとして、スケーリングポリシー ASGScalingPolicyHigh を指定します。Dimensions プロパティは、同じテンプレート内の別の場所で宣言された [AWS::AutoScaling::AutoScalingGroup](#) リソースの論理名を参照します。

JSON

```
{
```

```

"Resources":{
  "ASGScalingPolicyHigh":{
    "Type":"AWS::AutoScaling::ScalingPolicy",
    "Properties":{
      "AutoScalingGroupName":{"Ref":"logicalName" },
      "PolicyType":"StepScaling",
      "AdjustmentType":"ChangeInCapacity",
      "StepAdjustments":[
        {
          "MetricIntervalLowerBound":0,
          "ScalingAdjustment":1
        }
      ]
    }
  },
  "CPUALarmHigh":{
    "Type":"AWS::CloudWatch::Alarm",
    "Properties":{
      "EvaluationPeriods":"2",
      "Statistic":"Average",
      "Threshold":"90",
      "AlarmDescription":"Scale out if CPU > 90% for 2 minutes",
      "Period":"60",
      "AlarmActions":[ { "Ref":"ASGScalingPolicyHigh" } ],
      "Namespace":"AWS/EC2",
      "Dimensions":[
        {
          "Name":"AutoScalingGroupName",
          "Value":{"Ref":"logicalName" }
        }
      ],
      "ComparisonOperator":"GreaterThanOrEqualTo",
      "MetricName":"CPUUtilization"
    }
  }
}

```

YAML

```

---
Resources:
  ASGScalingPolicyHigh:

```

```
Type: AWS::AutoScaling::ScalingPolicy
Properties:
  AutoScalingGroupName: !Ref logicalName
  PolicyType: StepScaling
  AdjustmentType: ChangeInCapacity
  StepAdjustments:
    - MetricIntervalLowerBound: 0
      ScalingAdjustment: 1
CPUAlarmHigh:
  Type: AWS::CloudWatch::Alarm
  Properties:
    EvaluationPeriods: 2
    Statistic: Average
    Threshold: 90
    AlarmDescription: 'Scale out if CPU > 90% for 2 minutes'
    Period: 60
    AlarmActions:
      - !Ref ASGScalingPolicyHigh
  Namespace: AWS/EC2
  Dimensions:
    - Name: AutoScalingGroupName
      Value:
        !Ref logicalName
  ComparisonOperator: GreaterThanThreshold
  MetricName: CPUUtilization
```

関連情報

スケーリングポリシーのテンプレート例については、「AWS::AutoScaling::ScalingPolicy リソース」の「[例](#)」セクションを参照してください。

混合インスタンスグループの例

属性ベースのインスタンスタイプの選択を使用して Auto Scaling グループを作成する

この例に示すのは、属性ベースのインスタンスタイプを使って混合インスタンスグループを起動するための情報を含む、[AWS::AutoScaling::AutoScalingGroup](#) リソースです。VCpuCount プロパティの最小値と最大値を、および MemoryMiB プロパティの最小値を指定します。Auto Scaling グループで使用するインスタンスタイプはいずれも、必要なインスタンス属性と一致している必要があります。

Auto Scaling グループの VPCZoneIdentifier プロパティは、3 つの異なるアベイラビリティーゾーンの既存のサブネットのリストを指定します。スタックを作成する前に、アカウントから該当す

るサブネット ID を指定する必要があります。LaunchTemplate プロパティは、同じテンプレート内の別の場所で宣言された [AWS::EC2::LaunchTemplate](#) リソースの論理名を参照します。

JSON

```
{
  "Resources":{
    "myASG":{
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
        "VPCZoneIdentifier":[
          "subnetIdAz1",
          "subnetIdAz2",
          "subnetIdAz3"
        ],
        "MixedInstancesPolicy":{
          "LaunchTemplate":{
            "LaunchTemplateSpecification":{
              "LaunchTemplateId":{
                "Ref":"logicalName"
              },
              "Version":{
                "Fn::GetAtt":[
                  "logicalName",
                  "LatestVersionNumber"
                ]
              }
            }
          },
          "Overrides":[
            {
              "InstanceRequirements":{
                "VCpuCount":{
                  "Min":2,
                  "Max":4
                },
                "MemoryMiB":{
                  "Min":2048
                }
              }
            }
          ]
        }
      },
      "MaxSize":"5",
```

```
        "MinSize": "1"
    }
}
}
```

YAML

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - subnetIdAz1
        - subnetIdAz2
        - subnetIdAz3
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref logicalName
            Version: !GetAtt logicalName.LatestVersionNumber
          Overrides:
            - InstanceRequirements:
                VCpuCount:
                  Min: 2
                  Max: 4
                MemoryMiB:
                  Min: 2048
      MaxSize: '5'
      MinSize: '1'
```

起動構成の例

「起動設定を作成する」

この例に示すのは、ImageId、InstanceType、SecurityGroups プロパティの値を指定する、Auto Scaling グループの [AWS::AutoScaling::LaunchConfiguration](#) リソースです。SecurityGroups プロパティは、テンプレート内の他の場所で指定されている [AWS::EC2::SecurityGroup](#) リソースの論理名と、myExistingEC2SecurityGroup という既存の EC2 セキュリティグループの両方を指定します。

JSON

```
"mySimpleConfig" : {
  "Type" : "AWS::AutoScaling::LaunchConfiguration",
  "Properties" : {
    "ImageId" : "ami-02354e95b3example",
    "InstanceType" : "t3.micro",
    "SecurityGroups" : [ { "Ref" : "logicalName" }, "myExistingEC2SecurityGroup" ]
  }
}
```

YAML

```
mySimpleConfig:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    ImageId: ami-02354e95b3example
    InstanceType: t3.micro
    SecurityGroups:
      - !Ref logicalName
      - myExistingEC2SecurityGroup
```

起動設定を使用して Auto Scaling グループを作成する

この例に示すのは、単一インスタンスに関連付けられた

[AWS::AutoScaling::AutoScalingGroup](#) リソースです。Auto Scaling グループの `VPCZoneIdentifier` プロパティは、3 つの異なるアベイラビリティゾーンの既存のサブネットのリストを指定します。スタックを作成する前に、アカウントから該当するサブネット ID を指定する必要があります。LaunchConfigurationName プロパティは、論理名がテンプレート内で定義された mySimpleConfig である、[AWS::AutoScaling::LaunchConfiguration](#) リソースを参照します。

JSON

```
"myASG" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
    "LaunchConfigurationName" : { "Ref" : "mySimpleConfig" },
  }
}
```



```
    "MaxSize" : "1",  
    "MinSize" : "1"  
  }  
}
```

YAML

```
myASG:  
  Type: AWS::AutoScaling::AutoScalingGroup  
  Properties:  
    VPCZoneIdentifier:  
      - subnetIdAz1  
      - subnetIdAz2  
      - subnetIdAz3  
    LaunchConfigurationName: !Ref mySimpleConfig  
    MaxSize: '1'  
    MinSize: '1'
```

CloudFormation で Application Auto Scaling リソースを設定する

このセクションでは、さまざまな AWS リソースのアプリケーションの Auto Scaling スケーリングポリシーとスケジュールに基づくアクションに関する CloudFormation テンプレートの例を示します。

Important

アプリケーションの Auto Scaling スニペットがテンプレートに含まれている場合は、`DependsOn` 属性を使用して、テンプレートを通じて作成される特定のスケラブルなリソースへの依存関係を宣言する必要がある場合があります。これにより、デフォルトの並列処理が上書きされ、指定した順序で CloudFormation がリソースを操作ようになります。この宣言がないと、リソースの設定が完了する前にスケーリングの構成が適用される場合があります。

詳細については、「[DependsOn 属性](#)」を参照してください。

スニペットカテゴリ

- [AppStream フリートのスケーリングポリシーの作成](#)
- [Aurora DB クラスターのスケーリングポリシーを作成する](#)
- [DynamoDB テーブルのスケーリングポリシーを作成する](#)
- [Amazon ECS サービスのスケーリングポリシーを作成する \(メトリクス: 平均 CPU とメモリ\)](#)

- [Amazon ECS サービスのスケーリングポリシーを作成する \(メトリクス: ターゲットあたりの平均リクエスト数\)](#)
- [Lambda 関数の cron 式を使用した、計画されたアクションを作成する](#)
- [スポットフリートの at 式を使用して、計画されたアクションを作成する](#)

AppStream フリートのスケーリングポリシーの作成

このスニペットでは、ポリシーを作成し、[AWS::ApplicationAutoScaling::ScalingPolicy](#) リソースを使用して [AWS::AppStream::Fleet](#) リソースにそのポリシーを適用しています。[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースは、このポリシーが適用されるスケラブルターゲットを宣言します。Application Auto Scaling は、フリートインスタンスの数を最低 1 つ個のインスタンス、最大 20 個のインスタンスに拡張できます。このポリシーでは、フリートの平均容量使用率を 75% に保ち、スケールアウトとスケールインのクールダウン期間は 300 秒 (5 分) です。

Fn::Join と Rev の組み込み関数を使用して、同じテンプレートで指定された [AWS::AppStream::Fleet](#) リソースの、論理名を使用して ResourceId プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 20,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet" },
        "ServiceNamespace" : "appstream",
        "ScalableDimension" : "appstream:fleet:DesiredCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "fleet",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    }
  }
}
```

```

    }
  ]
]
}
},
"ScalingPolicyAppStreamFleet" : {
  "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
  "Properties" : {
    "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu75" },
    "PolicyType" : "TargetTrackingScaling",
    "ServiceNamespace" : "appstream",
    "ScalableDimension" : "appstream:fleet:DesiredCapacity",
    "ResourceId" : {
      "Fn::Join" : [
        "/",
        [
          "fleet",
          {
            "Ref" : "logicalName"
          }
        ]
      ]
    },
  },
  "TargetTrackingScalingPolicyConfiguration" : {
    "TargetValue" : 75,
    "PredefinedMetricSpecification" : {
      "PredefinedMetricType" : "AppStreamAverageCapacityUtilization"
    },
    "ScaleInCooldown" : 300,
    "ScaleOutCooldown" : 300
  }
}
}
}
}
}

```

YAML

```

---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget

```

```

Properties:
  MaxCapacity: 20
  MinCapacity: 1
  RoleARN:
    Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet'
  ServiceNamespace: appstream
  ScalableDimension: appstream:fleet:DesiredCapacity
  ResourceId: !Join
    - /
    - - fleet
      - !Ref logicalName
ScalingPolicyAppStreamFleet:
  Type: AWS::ApplicationAutoScaling::ScalingPolicy
  Properties:
    PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu75
    PolicyType: TargetTrackingScaling
    ServiceNamespace: appstream
    ScalableDimension: appstream:fleet:DesiredCapacity
    ResourceId: !Join
      - /
      - - fleet
        - !Ref logicalName
    TargetTrackingScalingPolicyConfiguration:
      TargetValue: 75
      PredefinedMetricSpecification:
        PredefinedMetricType: AppStreamAverageCapacityUtilization
      ScaleInCooldown: 300
      ScaleOutCooldown: 300

```

Aurora DB クラスターのスケールリングポリシーを作成する

このスニペットでは、[AWS::RDS::DBCluster](#) リソースを登録します。 [AWS::ApplicationAutoScaling::ScalableTarget](#) リソースは、DB クラスターが、1~8 個の Aurora レプリカを持つように動的にスケールされることを示します。また、ターゲット追跡スケールリングポリシーを、[AWS::ApplicationAutoScaling::ScalingPolicy](#) リソースを使用してクラスターに適用します。

この構成では、RDSReaderAverageCPUUtilization 事前定義メトリクスを使用して、Aurora DB クラスターのすべての Aurora レプリカでの平均 CPU 使用率 40% に基づいて Aurora DB クラスター

が調整されます。この構成では、10 分間のスケールインのクールダウン期間と 5 分間のスケールアウトのクールダウン期間が提供されます。

この例では、Fn::Sub 組み込み関数を使用して、同じテンプレートで指定されている AWS::RDS::DBCluster リソースの論理名で ResourceId プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 8,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/rds.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_RDSCluster" },
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" }
      }
    },
    "ScalingPolicyDBCluster" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu40" },
        "PolicyType" : "TargetTrackingScaling",
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 40,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "RDSReaderAverageCPUUtilization"
          },
          "ScaleInCooldown" : 600,
          "ScaleOutCooldown" : 300
        }
      }
    }
  }
}
```

```
}
```

YAML

```
---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 8
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster'
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
      ResourceId: !Sub cluster:${logicalName}
  ScalingPolicyDBCluster:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu40
      PolicyType: TargetTrackingScaling
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
      ResourceId: !Sub cluster:${logicalName}
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 40
        PredefinedMetricSpecification:
          PredefinedMetricType: RDSReaderAverageCPUUtilization
        ScaleInCooldown: 600
        ScaleOutCooldown: 300
```

DynamoDB テーブルのスケーリングポリシーを作成する

このスニペットでは、`TargetTrackingScaling` ポリシータイプを使ってポリシーを作成し、それを [AWS::ApplicationAutoScaling::ScalingPolicy](#) リソースを使用して [AWS::DynamoDB::Table](#) リソースに適用しています。 [AWS::ApplicationAutoScaling::ScalableTarget](#) リソースは、このポリシーが適用されるスケーラブルなターゲットとして、書き込みキャパシティーユニット (最大 5、最小 15) を宣言しています。スケーリングポリシーは、DynamoDBWriteCapacityUtilization 事前定義メトリクスに基づいて、ターゲット使用率を 50% に維持するように、テーブルの書き込み容量のスループットをスケーリングします。

`Fn::Join` と `Ref` の組み込み関数を使用して、同じテンプレートで指定された `AWS::DynamoDB::Table` リソースの、論理名を使用して `ResourceId` プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

Note

DynamoDB リソースの CloudFormation テンプレートを作成する方法の詳細については、AWS データベースブログの「[CloudFormation を使用して Amazon DynamoDB テーブルとインデックスの自動スケーリングを設定する方法](#)」を参照してください。

JSON

```
{
  "Resources" : {
    "WriteCapacityScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 15,
        "MinCapacity" : 5,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable" },
        "ServiceNamespace" : "dynamodb",
        "ScalableDimension" : "dynamodb:table:WriteCapacityUnits",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "table",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    },
    "WriteScalingPolicy" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : "WriteScalingPolicy",
```

```

    "PolicyType" : "TargetTrackingScaling",
    "ScalingTargetId" : { "Ref" : "WriteCapacityScalableTarget" },
    "TargetTrackingScalingPolicyConfiguration" : {
      "TargetValue" : 50.0,
      "ScaleInCooldown" : 60,
      "ScaleOutCooldown" : 60,
      "PredefinedMetricSpecification" : {
        "PredefinedMetricType" : "DynamoDBWriteCapacityUtilization"
      }
    }
  }
}
}
}
}
}

```

YAML

```

---
Resources:
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ServiceNamespace: dynamodb
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ResourceId: !Join
        - /
        - - table
          - !Ref logicalName
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0
        ScaleInCooldown: 60

```



```
ScaleOutCooldown: 60
PredefinedMetricSpecification:
  PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

Amazon ECS サービスのスケールリングポリシーを作成する (メトリクス: 平均 CPU とメモリ)

このスニペットでは、ポリシーを作成し、[AWS::ApplicationAutoScaling::ScalingPolicy](#) リソースを使用して [AWS::ECS::Service](#) リソースにそのポリシーを適用しています。[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースは、このポリシーが適用されるスケラブルターゲットを宣言します。Application Auto Scaling は、最小 1 タスク、最大 6 タスクのスケールリングが可能です。

また、TargetTrackingScaling ポリシータイプの 2 つのスケールリングポリシーを作成します。これらのポリシーは、サービスの平均 CPU およびメモリ使用量に基づいて ECS サービスをスケールするために使用されます。Fn::Join と Ref の組み込み関数を使用して、同じテンプレートで指定された [AWS::ECS::Cluster](#) (myContainerCluster) および AWS::ECS::Service (myService) リソースの論理名を使用して ResourceId プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

JSON

```
{
  "Resources" : {
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : "6",
        "MinCapacity" : "1",
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
                "Ref" : "myContainerCluster"
              }
            ]
          ]
        }
      }
    }
  }
```

```

        "Fn::GetAtt" : [
            "myService",
            "Name"
        ]
    }
]
]
}
},
"ServiceScalingPolicyCPU" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
    "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu70" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
            "TargetValue" : 70.0,
            "ScaleInCooldown" : 180,
            "ScaleOutCooldown" : 60,
            "PredefinedMetricSpecification" : {
                "PredefinedMetricType" : "ECSServiceAverageCPUUtilization"
            }
        }
    }
},
"ServiceScalingPolicyMem" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
    "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-mem90" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
            "TargetValue" : 90.0,
            "ScaleInCooldown" : 180,
            "ScaleOutCooldown" : 60,
            "PredefinedMetricSpecification" : {
                "PredefinedMetricType" : "ECSServiceAverageMemoryUtilization"
            }
        }
    }
}
}
}

```

```
}
```

YAML

```
---
Resources:
  ECSScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 6
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService'
      ServiceNamespace: ecs
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId: !Join
        - /
        - - service
          - !Ref myContainerCluster
          - !GetAtt myService.Name
  ServiceScalingPolicyCPU:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu70
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 70.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: ECSServiceAverageCPUUtilization
  ServiceScalingPolicyMem:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-mem90
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 90.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
```

```
PredefinedMetricSpecification:
```

```
  PredefinedMetricType: ECSServiceAverageMemoryUtilization
```

Amazon ECS サービスのスケーリングポリシーを作成する (メトリクス: ターゲットあたりの平均リクエスト数)

以下の例では、ALBRequestCountPerTarget 事前定義メトリクスを使用するターゲット追跡スケーリングポリシーを ECS サービスに適用します。このポリシーは、ターゲットあたり (1 分あたり) のリクエストカウントがターゲット値を超えた場合に ECS サービスに容量を追加するために使用されます。DisableScaleIn の値が true に設定されているため、ターゲット追跡ポリシーはスケラブルなターゲットから容量を削除しません。

Fn::Join と Fn::GetAtt の組み込み関数を使用して、同じテンプレートで指定された [AWS::ElasticLoadBalancingV2::LoadBalancer](#) (myLoadBalancer) および [AWS::ElasticLoadBalancingV2::TargetGroup](#) (myTargetGroup) リソースの論理名を使用して ResourceLabel プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

スケラブルなターゲットの MaxCapacity および MinCapacity プロパティ、スケーリングポリシーの TargetValue プロパティは、スタックの作成または更新時にテンプレートに渡すパラメータ値を参照します。

JSON

```
{
  "Resources" : {
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : { "Ref" : "MaxCount" },
        "MinCapacity" : { "Ref" : "MinCount" },
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
```

```

        "Ref" : "myContainerCluster"
    },
    {
        "Fn::GetAtt" : [
            "myService",
            "Name"
        ]
    }
]
]
}
},
"ServiceScalingPolicyALB" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
    "Properties" : {
        "PolicyName" : "alb-requests-per-target-per-minute",
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
            "TargetValue" : { "Ref" : "ALBPolicyTargetValue" },
            "ScaleInCooldown" : 180,
            "ScaleOutCooldown" : 30,
            "DisableScaleIn" : true,
            "PredefinedMetricSpecification" : {
                "PredefinedMetricType" : "ALBRequestCountPerTarget",
                "ResourceLabel" : {
                    "Fn::Join" : [
                        "/",
                        [
                            {
                                "Fn::GetAtt" : [
                                    "myLoadBalancer",
                                    "LoadBalancerFullName"
                                ]
                            },
                            {
                                "Fn::GetAtt" : [
                                    "myTargetGroup",
                                    "TargetGroupFullName"
                                ]
                            }
                        ]
                    ]
                }
            }
        }
    }
}
]
]

```



```
- !GetAtt myTargetGroup.TargetGroupFullName
```

Lambda 関数の cron 式を使用した、計画されたアクションを作成する

このスニペットでは、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、BLUE という名前の関数エイリアス ([AWS::Lambda::Alias](#)) のプロビジョニングされた同時実行を登録しています。また、Cron 式を使用して、定期的なスケジュールでスケジュールされたアクションを作成します。定期スケジュールのタイムゾーンは UTC です。

RoleARN プロパティで Fn::Join と Ref の組み込み関数を使用し、サービスリンクロールの ARN を指定します。Fn::Sub の組み込み関数を使用して、同じテンプレートで指定された [AWS::Lambda::Function](#) または [AWS::Serverless::Function](#) リソースの論理名を使用して ResourceId プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

Note

非公開バージョン (\$LATEST) を参照するエイリアスに、プロビジョニングされた同時実行を割り当てることもできません。

Lambda リソースの CloudFormation テンプレートを作成する方法の詳細については、AWS コンピューティングブログのブログ記事「[繰り返し発生するピーク使用量のために AWS Lambda プロビジョニングされた同時実行をスケジュールする](#)」を参照してください。

JSON

```
{
  "ScalableTarget" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties" : {
      "MaxCapacity" : 250,
      "MinCapacity" : 0,
      "RoleARN" : {
        "Fn::Join" : [
          ":",
          [
            "arn:aws:iam:",
            {
              "Ref" : "AWS::AccountId"
            }
          ]
        ]
      }
    }
  }
}
```

```

        "role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
        AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency"
    ]
  ],
  "ServiceNamespace" : "lambda",
  "ScalableDimension" : "lambda:function:ProvisionedConcurrency",
  "ResourceId" : { "Fn::Sub" : "function:${logicalName}:BLUE" },
  "ScheduledActions" : [
    {
      "ScalableTargetAction" : {
        "MinCapacity" : "250"
      },
      "ScheduledActionName" : "my-scale-out-scheduled-action",
      "Schedule" : "cron(0 18 * * ? *)",
      "EndTime" : "2022-12-31T12:00:00.000Z"
    }
  ]
}
}
}
}

```

YAML

```

ScalableTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: 250
    MinCapacity: 0
    RoleARN: !Join
      - ':'
      - - 'arn:aws:iam:'
        - !Ref 'AWS::AccountId'
        - role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
        AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency
    ServiceNamespace: lambda
    ScalableDimension: lambda:function:ProvisionedConcurrency
    ResourceId: !Sub function:${logicalName}:BLUE
    ScheduledActions:
      - ScalableTargetAction:
          MinCapacity: 250
          ScheduledActionName: my-scale-out-scheduled-action
          Schedule: 'cron(0 18 * * ? *)'

```



```
EndTime: '2022-12-31T12:00:00.000Z'
```

スポットフリートの **at** 式を使用して、計画されたアクションを作成する

このスニペットでは、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、[AWS::EC2::SpotFleet](#) リソースに対して 1 回だけ実行される 2 つの計画されたアクションを作成しています。1 回限りのスケジュールされた各アクションのタイムゾーンは UTC です。

Fn::Join と Ref の組み込み関数を使用して、同じテンプレートで指定された AWS::EC2::SpotFleet リソースの論理名を使用して ResourceId プロパティを作成します。詳細については、「[組み込み関数リファレンス](#)」を参照してください。

Note

スポットフリートリクエストには、タイプが maintain のリクエストが必要です。自動スケーリングは 1 回限りのリクエストまたはスポットブロックではサポートされません。

JSON

```
{
  "Resources" : {
    "SpotFleetScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 0,
        "MinCapacity" : 0,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ec2.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest" },
        "ServiceNamespace" : "ec2",
        "ScalableDimension" : "ec2:spot-fleet-request:TargetCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "spot-fleet-request",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    }
  }
}
```

```

    },
    "ScheduledActions" : [
      {
        "ScalableTargetAction" : {
          "MaxCapacity" : "10",
          "MinCapacity" : "10"
        },
        "ScheduledActionName" : "my-scale-out-scheduled-action",
        "Schedule" : "at(2022-05-20T13:00:00)"
      },
      {
        "ScalableTargetAction" : {
          "MaxCapacity" : "0",
          "MinCapacity" : "0"
        },
        "ScheduledActionName" : "my-scale-in-scheduled-action",
        "Schedule" : "at(2022-05-20T21:00:00)"
      }
    ]
  }
}
}
}

```

YAML

```

---
Resources:
  SpotFleetScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 0
      MinCapacity: 0
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ec2.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest'
      ServiceNamespace: ec2
      ScalableDimension: 'ec2:spot-fleet-request:TargetCapacity'
      ResourceId: !Join
        - /
        - - spot-fleet-request
          - !Ref logicalName
      ScheduledActions:

```

```
- ScalableTargetAction:
  MaxCapacity: 10
  MinCapacity: 10
  ScheduledActionName: my-scale-out-scheduled-action
  Schedule: 'at(2022-05-20T13:00:00)'
- ScalableTargetAction:
  MaxCapacity: 0
  MinCapacity: 0
  ScheduledActionName: my-scale-in-scheduled-action
  Schedule: 'at(2022-05-20T21:00:00)'
```

AWS 請求コンソールのテンプレートスニペット

この例では、10% のグローバルマークアップ価格設定ルールを持つ価格プランを 1 つ作成します。この料金プランは請求グループに添付されています。請求グループには、請求グループの合計費用に加えて 10 USD の請求と 10% の請求を適用する 2 つのカスタム明細項目もあります。

JSON

```
{
  "Parameters": {
    "LinkedAccountIds": {
      "Type": "ListNumber"
    },
    "PrimaryAccountId": {
      "Type": "Number"
    }
  },
  "Resources": {
    "TestPricingRule": {
      "Type": "AWS::BillingConductor::PricingRule",
      "Properties": {
        "Name": "TestPricingRule",
        "Description": "Test pricing rule created through Cloudformation. Mark everything by 10%.",
        "Type": "MARKUP",
        "Scope": "GLOBAL",
        "ModifierPercentage": 10
      }
    },
    "TestPricingPlan": {
      "Type": "AWS::BillingConductor::PricingPlan",
      "Properties": {
```

```

        "Name": "TestPricingPlan",
        "Description": "Test pricing plan created through Cloudformation.",
        "PricingRuleArns": [
            {"Fn::GetAtt": ["TestPricingRule", "Arn"]}
        ]
    },
    "TestBillingGroup": {
        "Type": "AWS::BillingConductor::BillingGroup",
        "Properties": {
            "Name": "TestBillingGroup",
            "Description": "Test billing group created through Cloudformation with 1
linked account. The linked account is also the primary account.",
            "PrimaryAccountId": {
                "Ref": "PrimaryAccountId"
            },
            "AccountGrouping": {
                "LinkedAccountIds": null
            },
            "ComputationPreference": {
                "PricingPlanArn": {
                    "Fn::GetAtt": ["TestPricingPlan", "Arn"]
                }
            }
        }
    },
    "TestFlatCustomLineItem": {
        "Type": "AWS::BillingConductor::CustomLineItem",
        "Properties": {
            "Name": "TestFlatCustomLineItem",
            "Description": "Test flat custom line item created through Cloudformation
for a $10 charge.",
            "BillingGroupArn": {
                "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
                "Flat": {
                    "ChargeValue": 10
                },
                "Type": "FEE"
            }
        }
    },
    "TestPercentageCustomLineItem": {

```

```

        "Type": "AWS::BillingConductor::CustomLineItem",
        "Properties": {
            "Name": "TestPercentageCustomLineItem",
            "Description": "Test percentage custom line item created through
Cloudformation for a %10 additional charge on the overall total bill of the billing
group.",
            "BillingGroupArn": {
                "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
                "Percentage": {
                    "PercentageValue": 10,
                    "ChildAssociatedResources": [
                        {"Fn::GetAtt": ["TestBillingGroup", "Arn"]}
                    ]
                },
                "Type": "FEE"
            }
        }
    }
}

```

YAML

```

Parameters:
  LinkedAccountIds:
    Type: ListNumber
  PrimaryAccountId:
    Type: Number
Resources:
  TestPricingRule:
    Type: AWS::BillingConductor::PricingRule
    Properties:
      Name: 'TestPricingRule'
      Description: 'Test pricing rule created through Cloudformation. Mark everything
by 10%.'
      Type: 'MARKUP'
      Scope: 'GLOBAL'
      ModifierPercentage: 10
  TestPricingPlan:
    Type: AWS::BillingConductor::PricingPlan
    Properties:

```

```

    Name: 'TestPricingPlan'
    Description: 'Test pricing plan created through Cloudformation.'
    PricingRuleArns:
      - !GetAtt TestPricingRule.Arn
  TestBillingGroup:
    Type: AWS::BillingConductor::BillingGroup
    Properties:
      Name: 'TestBillingGroup'
      Description: 'Test billing group created through Cloudformation with 1 linked
account. The linked account is also the primary account.'
      PrimaryAccountId: !Ref PrimaryAccountId
      AccountGrouping:
        LinkedAccountIds: !Ref LinkedAccountIds
      ComputationPreference:
        PricingPlanArn: !GetAtt TestPricingPlan.Arn
  TestFlatCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestFlatCustomLineItem'
      Description: 'Test flat custom line item created through Cloudformation for a $10
charge.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
        Flat:
          ChargeValue: 10
          Type: 'FEE'
  TestPercentageCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestPercentageCustomLineItem'
      Description: 'Test percentage custom line item created through Cloudformation for
a %10 additional charge on the overall total bill of the billing group.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
        Percentage:
          PercentageValue: 10
          ChildAssociatedResources:
            - !GetAtt TestBillingGroup.Arn
      Type: 'FEE'

```

CloudFormation テンプレートスニペット

トピック

- [ネストされたスタック](#)
- [待機条件](#)

ネストされたスタック

テンプレートにおけるスタックのネスト

このサンプルテンプレートには、myStack というスタックリソースがネストされています。CloudFormation でテンプレートからスタックを作成するときに、myStack が作成され、そのテンプレートは TemplateURL プロパティで指定されます。出力値 StackRef は myStack のスタック ID を、OutputFromNestedStack は myStack リソース内からの出力値 BucketName を返します。Outputs.*nestedstackoutputname* 形式は、ネストされたスタックからの出力値を指定する目的で予約されており、上位のテンプレート内の任意の場所で使用できます。

詳細については、「[AWS::CloudFormation::Stack](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "Properties" : {
        "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template",
        "TimeoutInMinutes" : "60"
      }
    }
  },
  "Outputs": {
    "StackRef": {"Value": { "Ref" : "myStack"}},
    "OutputFromNestedStack" : {
      "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
  myStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template
      TimeoutInMinutes: '60'
Outputs:
  StackRef:
    Value: !Ref myStack
  OutputFromNestedStack:
    Value: !GetAtt myStack.Outputs.BucketName
```

テンプレートでの入力パラメータによるスタックのネスト

このサンプルテンプレートには、入力パラメーターを指定するスタックリソースが存在します。CloudFormation は、このテンプレートからスタックを作成する際、Parameters プロパティ内で宣言されている値ペアを、myStackWithParams スタックの作成用テンプレートの入力パラメータとして使用します。この例では、InstanceType パラメーターと KeyName パラメーターを指定しています。

詳細については、「[AWS::CloudFormation::Stack](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myStackWithParams" : {
      "Type" : "AWS::CloudFormation::Stack",
      "Properties" : {
        "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1/EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t2.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}
```


YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myStackWithParams:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
        EC2ChooseAMI.template
      Parameters:
        InstanceType: t2.micro
        KeyName: mykey
```

待機条件

Amazon EC2 インスタンスでの待機条件の使用

Important

Amazon EC2 および Auto Scaling リソースについては、待機条件ではなく CreationPolicy 属性を使用することをお勧めします。CreationPolicy 属性をこれらのリソースに追加し、インスタンス作成プロセスが正常に完了したときにシグナルを送信するために cfn-signal ヘルパースクリプトを使用します。

作成ポリシーを使用できない場合は、待機条件を伴う Amazon EC2 インスタンスを宣言する次のテンプレート例を参照してください。待機条件 myWaitCondition では、myWaitConditionHandle を伝達手段に用いると共に、Amazon EC2 インスタンスリソースが作成された後で待機条件をトリガーする指定を DependsOn 属性で行っています。また、Timeout プロパティを使用して、待機条件の持続時間を 4500 秒に設定しています。加えて、待機条件へのシグナルの働きをする署名付き URL を、Ec2Instance リソースの UserData プロパティで Amazon EC2 インスタンスに渡しています。Amazon EC2 インスタンスで動作するアプリケーションまたはスクリプトはこの署名付き URL を取得し、成功または失敗を待機条件に伝達するための手段とします。cfn-signal を使用するか、待機条件にシグナルを送信するアプリケーションまたはスクリプトは自分で作成する必要があります。待機条件シグナルから戻されたデータは、出力値である ApplicationData に格納されます。

詳細については、「[CloudFormation テンプレートで待機条件を作成する](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {
        "AMI" : "ami-0123456789abcdef0"
      },
      "us-west-1" : {
        "AMI" : "ami-0987654321fedcba0"
      },
      "eu-west-1" : {
        "AMI" : "ami-0abcdef123456789a"
      },
      "ap-northeast-1" : {
        "AMI" : "ami-0fedcba987654321b"
      },
      "ap-southeast-1" : {
        "AMI" : "ami-0c1d2e3f4a5b6c7d8"
      }
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "UserData" : { "Fn::Base64" : {"Ref" : "myWaitHandle"} },
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "AMI" ] }
      }
    },
    "myWaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle",
      "Properties" : {
      }
    },
    "myWaitCondition" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "Ec2Instance",
      "Properties" : {
        "Handle" : { "Ref" : "myWaitHandle" },
        "Timeout" : "4500"
      }
    }
  }
}
```

```

    }
  },
  "Outputs" : {
    "ApplicationData" : {
      "Value" : { "Fn::GetAtt" : [ "myWaitCondition", "Data" ] },
      "Description" : "The data passed back as part of signalling the
WaitCondition."
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0123456789abcdef0
    us-west-1:
      AMI: ami-0987654321fedcba0
    eu-west-1:
      AMI: ami-0abcdef123456789a
    ap-northeast-1:
      AMI: ami-0fedcba987654321b
    ap-southeast-1:
      AMI: ami-0c1d2e3f4a5b6c7d8
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      UserData:
        Fn::Base64: !Ref myWaitHandle
      ImageId:
        Fn::FindInMap:
          - RegionMap
          - Ref: AWS::Region
          - AMI
  myWaitHandle:
    Type: AWS::CloudFormation::WaitConditionHandle
    Properties: {}
  myWaitCondition:
    Type: AWS::CloudFormation::WaitCondition
    DependsOn: Ec2Instance

```

Properties:

Handle: !Ref myWaitHandle

Timeout: '4500'

Outputs:**ApplicationData:**

Value: !GetAtt myWaitCondition.Data

Description: The data passed back as part of signalling the WaitCondition.

cfn-signal ヘルパー スクリプトを使用して待機条件を通知する

この例は、成功を待機条件に伝達する `cfn-signal` コマンドラインを示します。EC2 インスタンスの `UserData` プロパティでコマンドラインを定義する必要があります。

JSON

```
"UserData": {
  "Fn::Base64": {
    "Fn::Join": [
      "",
      [
        "#!/bin/bash -xe\n",
        "/opt/aws/bin/cfn-signal --exit-code 0 '",
        {
          "Ref": "myWaitHandle"
        },
        "'\n"
      ]
    ]
  }
}
```

YAML

```
UserData:
  'Fn::Base64':
    'Fn::Join':
      - ''
      - - |
          #!/bin/bash -xe
          - /opt/aws/bin/cfn-signal --exit-code 0 '
          - Ref: myWaitHandle
          - |
```

Curl を使用した待機条件への伝達

この例は、成功を待機条件に伝達する Curl コマンドラインを示します。

```
curl -T /tmp/a "https://cloudformation-waitcondition-test.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wy0oRejVoo%3D"
```

ファイル /tmp/a には、次の JSON 構造が格納されます。

```
{
  "Status" : "SUCCESS",
  "Reason" : "Configuration Complete",
  "UniqueId" : "ID1234",
  "Data" : "Application has completed configuration."
}
```

この例も、同じ成功シグナルを送信する Curl コマンドラインですが、コマンドラインのパラメーターとして JSON を送信しています。

```
curl -X PUT -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" : "Configuration Complete","UniqueId" : "ID1234","Data" : "Application has completed configuration."}' "https://cloudformation-waitcondition-test.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wy0oRejVoo%3D"
```

Amazon CloudFront テンプレートスニペット

CloudFormation の Amazon CloudFront に対して、テンプレートのサンプルスニペットを使用します。詳細については、「[Amazon CloudFront resource type reference](#)」を参照してください。

トピック

- [Amazon S3 オリジンを使用した Amazon CloudFront Distribution リソース](#)

- [カスタムオリジンを使用した Amazon CloudFront Distribution リソース](#)
- [複数のオリジンをサポートする Amazon CloudFront ディストリビューション](#)
- [Lambda 関数をオリジンとする Amazon CloudFront ディストリビューション](#)
- [関連情報](#)

Amazon S3 オリジンを使用した Amazon CloudFront Distribution リソース

次のサンプルテンプレートは、[S3Origin](#) とレガシーオリジンアクセスアイデンティティ (OAI) を使用する Amazon CloudFront [ディストリビューション](#) を示しています。代わりにオリジンアクセスコントロール (OAC) を使用する方法については、「Amazon CloudFront デベロッパーガイド」の「[Restricting access to an Amazon Simple Storage Service origin](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myDistribution" : {
      "Type" : "AWS::CloudFront::Distribution",
      "Properties" : {
        "DistributionConfig" : {
          "Origins" : [ {
            "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
            "Id" : "myS3Origin",
            "S3OriginConfig" : {
              "OriginAccessIdentity" : "origin-access-identity/cloudfront/E127EXAMPLE51Z"
            }
          } ],
          "Enabled" : "true",
          "Comment" : "Some comment",
          "DefaultRootObject" : "index.html",
          "Logging" : {
            "IncludeCookies" : "false",
            "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
            "Prefix" : "myprefix"
          },
          "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
          "DefaultCacheBehavior" : {
            "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT" ],
```



```
- mysite.example.com
- yoursite.example.com
DefaultCacheBehavior:
  AllowedMethods:
    - DELETE
    - GET
    - HEAD
    - OPTIONS
    - PATCH
    - POST
    - PUT
  TargetOriginId: myS3Origin
  ForwardedValues:
    QueryString: 'false'
    Cookies:
      Forward: none
  TrustedSigners:
    - 1234567890EX
    - 1234567891EX
  ViewerProtocolPolicy: allow-all
PriceClass: PriceClass_200
Restrictions:
  GeoRestriction:
    RestrictionType: whitelist
    Locations:
      - AQ
      - CV
ViewerCertificate:
  CloudFrontDefaultCertificate: 'true'
```

カスタムオリジンを使用した Amazon CloudFront Distribution リソース

次のテンプレートの例では、[CustomOrigin](#) を使用した Amazon CloudFront [\[Distribution\]](#) (ディストリビューション) を示します。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myDistribution" : {
      "Type" : "AWS::CloudFront::Distribution",
      "Properties" : {
```



```
"DistributionConfig" : {
  "Origins" : [ {
    "DomainName" : "www.example.com",
    "Id" : "myCustomOrigin",
    "CustomOriginConfig" : {
      "HTTPPort" : "80",
      "HTTPSPort" : "443",
      "OriginProtocolPolicy" : "http-only"
    }
  } ],
  "Enabled" : "true",
  "Comment" : "Somecomment",
  "DefaultRootObject" : "index.html",
  "Logging" : {
    "IncludeCookies" : "true",
    "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
    "Prefix" : "myprefix"
  },
  "Aliases" : [
    "mysite.example.com",
    "*.yoursite.example.com"
  ],
  "DefaultCacheBehavior" : {
    "TargetOriginId" : "myCustomOrigin",
    "SmoothStreaming" : "false",
    "ForwardedValues" : {
      "QueryString" : "false",
      "Cookies" : { "Forward" : "all" }
    }
  },
  "TrustedSigners" : [
    "1234567890EX",
    "1234567891EX"
  ],
  "ViewerProtocolPolicy" : "allow-all"
},
"CustomErrorResponses" : [ {
  "ErrorCode" : "404",
  "ResponsePagePath" : "/error-pages/404.html",
  "ResponseCode" : "200",
  "ErrorCachingMinTTL" : "30"
} ],
"PriceClass" : "PriceClass_200",
"Restrictions" : {
  "GeoRestriction" : {
```



```

    TrustedSigners:
      - 1234567890EX
      - 1234567891EX
    ViewerProtocolPolicy: allow-all
  CustomErrorResponses:
    - ErrorCode: '404'
      ResponsePagePath: "/error-pages/404.html"
      ResponseCode: '200'
      ErrorCachingMinTTL: '30'
  PriceClass: PriceClass_200
  Restrictions:
    GeoRestriction:
      RestrictionType: whitelist
      Locations:
        - AQ
        - CV
  ViewerCertificate:
    CloudFrontDefaultCertificate: 'true'

```

複数のオリジンをサポートする Amazon CloudFront デイストリビューション

次のテンプレート例では、複数のオリジンをサポートする CloudFront [デイストリビューション](#)を宣言する方法を示します。[DistributionConfig](#) では、オリジンのリストが指定され、[DefaultCacheBehavior](#) が設定されます。

JSON

```

{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myDistribution" : {
      "Type" : "AWS::CloudFront::Distribution",
      "Properties" : {
        "DistributionConfig" : {
          "Origins" : [ {
            "Id" : "myS3Origin",
            "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
            "S3OriginConfig" : {
              "OriginAccessIdentity" : "origin-access-identity/cloudfront/E127EXAMPLE51Z"
            }
          }
        ],
        "DefaultCacheBehavior" : {

```

```

        "Id" : "myCustomOrigin",
        "DomainName" : "www.example.com",
        "CustomOriginConfig" : {
            "HTTPPort" : "80",
            "HTTPSPort" : "443",
            "OriginProtocolPolicy" : "http-only"
        }
    },
    "Enabled" : "true",
    "Comment" : "Some comment",
    "DefaultRootObject" : "index.html",
    "Logging" : {
        "IncludeCookies" : "true",
        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
        "Prefix" : "myprefix"
    },
    "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
    "DefaultCacheBehavior" : {
        "TargetOriginId" : "myS3Origin",
        "ForwardedValues" : {
            "QueryString" : "false",
            "Cookies" : { "Forward" : "all" }
        },
        "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
        "ViewerProtocolPolicy" : "allow-all",
        "MinTTL" : "100",
        "SmoothStreaming" : "true"
    },
    "CacheBehaviors" : [ {
        "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
"PATCH", "POST", "PUT" ],
        "TargetOriginId" : "myS3Origin",
        "ForwardedValues" : {
            "QueryString" : "true",
            "Cookies" : { "Forward" : "none" }
        },
        "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
        "ViewerProtocolPolicy" : "allow-all",
        "MinTTL" : "50",
        "PathPattern" : "images1/*.jpg"
    },
    {

```



```
    HTTPPort: '80'
    HTTPSPort: '443'
    OriginProtocolPolicy: http-only
Enabled: 'true'
Comment: Some comment
DefaultRootObject: index.html
Logging:
  IncludeCookies: 'true'
  Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
  Prefix: myprefix
Aliases:
- mysite.example.com
- yoursite.example.com
DefaultCacheBehavior:
  TargetOriginId: myS3Origin
  ForwardedValues:
    QueryString: 'false'
    Cookies:
      Forward: all
  TrustedSigners:
  - 1234567890EX
  - 1234567891EX
  ViewerProtocolPolicy: allow-all
  MinTTL: '100'
  SmoothStreaming: 'true'
CacheBehaviors:
- AllowedMethods:
  - DELETE
  - GET
  - HEAD
  - OPTIONS
  - PATCH
  - POST
  - PUT
  TargetOriginId: myS3Origin
  ForwardedValues:
    QueryString: 'true'
    Cookies:
      Forward: none
  TrustedSigners:
  - 1234567890EX
  - 1234567891EX
  ViewerProtocolPolicy: allow-all
  MinTTL: '50'
```

```
    PathPattern: images1/*.jpg
  - AllowedMethods:
    - DELETE
    - GET
    - HEAD
    - OPTIONS
    - PATCH
    - POST
    - PUT
    TargetOriginId: myCustomOrigin
    ForwardedValues:
      QueryString: 'true'
      Cookies:
        Forward: none
    TrustedSigners:
    - 1234567890EX
    - 1234567891EX
    ViewerProtocolPolicy: allow-all
    MinTTL: '50'
    PathPattern: images2/*.jpg
  CustomErrorResponses:
  - ErrorCode: '404'
    ResponsePagePath: "/error-pages/404.html"
    ResponseCode: '200'
    ErrorCachingMinTTL: '30'
  PriceClass: PriceClass_All
  ViewerCertificate:
    CloudFrontDefaultCertificate: 'true'
```

Lambda 関数をオリジンとする Amazon CloudFront デистриビューション

次の例では、指定された Lambda 関数 URL (パラメータとして提供) をフロントとする CloudFront デистриビューションを作成し、HTTPS のみのアクセス、キャッシュ、圧縮、およびグローバル配信を有効にします。Lambda URL をカスタム HTTPS オリジンとして設定し、標準の AWS キャッシュポリシーを適用します。このデистриビューションは、HTTP/2 および IPv6 サポートによりパフォーマンスが最適化されており、CloudFront ドメイン名を出力します。これにより、ユーザーは安全な CDN ベースのエンドポイントを介して Lambda 関数にアクセスできます。詳細については、AWS ブログの「[Using Amazon CloudFront with AWS Lambda as origin to accelerate your web applications](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "LambdaEndpoint": {
      "Type": "String",
      "Description": "The Lambda function URL endpoint without the 'https://'"
    }
  },
  "Resources": {
    "MyDistribution": {
      "Type": "AWS::CloudFront::Distribution",
      "Properties": {
        "DistributionConfig": {
          "PriceClass": "PriceClass_All",
          "HttpVersion": "http2",
          "IPV6Enabled": true,
          "Origins": [
            {
              "DomainName": {
                "Ref": "LambdaEndpoint"
              },
              "Id": "LambdaOrigin",
              "CustomOriginConfig": {
                "HTTPSPort": 443,
                "OriginProtocolPolicy": "https-only"
              }
            }
          ],
          "Enabled": "true",
          "DefaultCacheBehavior": {
            "TargetOriginId": "LambdaOrigin",
            "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e58f6",
            "ViewerProtocolPolicy": "redirect-to-https",
            "SmoothStreaming": "false",
            "Compress": "true"
          }
        }
      }
    }
  },
  "Outputs": {
    "CloudFrontDomain": {
```



```

        "Description": "CloudFront default domain name configured",
        "Value": {
            "Fn::Sub": "https://${MyDistribution.DomainName}/"
        }
    }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Parameters:
  LambdaEndpoint:
    Type: String
    Description: The Lambda function URL endpoint without the 'https://'
Resources:
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        PriceClass: PriceClass_All
        HttpVersion: http2
        IPV6Enabled: true
        Origins:
          - DomainName: !Ref LambdaEndpoint
            Id: LambdaOrigin
            CustomOriginConfig:
              HTTPSPort: 443
              OriginProtocolPolicy: https-only
        Enabled: 'true'
      DefaultCacheBehavior:
        TargetOriginId: LambdaOrigin
        CachePolicyId: '658327ea-f89d-4fab-a63d-7e88639e58f6'
        ViewerProtocolPolicy: redirect-to-https
        SmoothStreaming: 'false'
        Compress: 'true'
Outputs:
  CloudFrontDomain:
    Description: CloudFront default domain name configured
    Value: !Sub https://${MyDistribution.DomainName}/

```

関連情報

Route 53 レコードにカスタムエイリアスを追加して CloudFront デイストリビューションにわかりやすい名前を付ける例については、「[CloudFront デイストリビューションのエイリアスリソースレコードセット](#)」を参照してください。

Amazon CloudWatch テンプレートスニペット

これらのサンプルテンプレートスニペットをは、CloudFormation で Amazon CloudWatch リソースの説明を記述するのに役立ちます。詳細については、[Amazon CloudWatch リソースタイプリファレンス](#)を参照してください。

トピック

- [請求アラーム](#)
- [CPU 使用率アラーム](#)
- [Amazon Elastic Compute Cloud インスタンスの復元](#)
- [基本ダッシュボードを作成する](#)
- [サイドバイサイドウィジェットでダッシュボードを作成する](#)

請求アラーム

次のサンプルでは、Amazon CloudWatch は AWS アカウントに対する課金アラームしきい値を超えた場合に E メール通知を送信します。使用量の通知を受け取るには、請求アラートを有効にします。詳細については、「Amazon CloudWatch ユーザーガイド」の「[AWS の推測料金をモニタリングする請求アラームの作成](#)」を参照してください。

JSON

```
"SpendingAlarm": {
  "Type": "AWS::CloudWatch::Alarm",
  "Properties": {
    "AlarmDescription": { "Fn::Join": ["", [
      "Alarm if AWS spending is over $",
      { "Ref": "AlarmThreshold" }
    ]}],
    "Namespace": "AWS/Billing",
    "MetricName": "EstimatedCharges",
    "Dimensions": [{
```

```

    "Name": "Currency",
    "Value" : "USD"
  ]],
  "Statistic": "Maximum",
  "Period": "21600",
  "EvaluationPeriods": "1",
  "Threshold": { "Ref": "AlarmThreshold" },
  "ComparisonOperator": "GreaterThanThreshold",
  "AlarmActions": [{
    "Ref": "BillingAlarmNotification"
  }],
  "InsufficientDataActions": [{
    "Ref": "BillingAlarmNotification"
  }]
}
}

```

YAML

```

SpendingAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription:
      'Fn::Join':
        - ''
        - - Alarm if AWS spending is over $
          - !Ref: AlarmThreshold
    Namespace: AWS/Billing
    MetricName: EstimatedCharges
    Dimensions:
      - Name: Currency
        Value: USD
    Statistic: Maximum
    Period: '21600'
    EvaluationPeriods: '1'
    Threshold:
      !Ref: "AlarmThreshold"
    ComparisonOperator: GreaterThanThreshold
    AlarmActions:
      - !Ref: "BillingAlarmNotification"
    InsufficientDataActions:
      - !Ref: "BillingAlarmNotification"

```

CPU 使用率アラーム

次のサンプルスニペットでは、Amazon EC2 インスタンスの平均 CPU 使用率が 3 つの評価期間にわたって 60 秒間以上 90% を超えたときに通知を送信するアラームを作成します。

JSON

```
"CPUAlarm" : {
  "Type" : "AWS::CloudWatch::Alarm",
  "Properties" : {
    "AlarmDescription" : "CPU alarm for my instance",
    "AlarmActions" : [ { "Ref" : "logical name of an AWS::SNS::Topic resource" } ],
    "MetricName" : "CPUUtilization",
    "Namespace" : "AWS/EC2",
    "Statistic" : "Average",
    "Period" : "60",
    "EvaluationPeriods" : "3",
    "Threshold" : "90",
    "ComparisonOperator" : "GreaterThanThreshold",
    "Dimensions" : [ {
      "Name" : "InstanceId",
      "Value" : { "Ref" : "logical name of an AWS::EC2::Instance resource" }
    } ]
  }
}
```

YAML

```
CPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: CPU alarm for my instance
    AlarmActions:
      - !Ref: "logical name of an AWS::SNS::Topic resource"
    MetricName: CPUUtilization
    Namespace: AWS/EC2
    Statistic: Average
    Period: '60'
    EvaluationPeriods: '3'
    Threshold: '90'
    ComparisonOperator: GreaterThanThreshold
    Dimensions:
      - Name: InstanceId
```

Value: !Ref: *"logical name of an AWS::EC2::Instance resource"*

Amazon Elastic Compute Cloud インスタンスの復元

次の CloudWatch アラームは、15 分間連続してステータスチェックに失敗した場合、EC2 インスタンスを復元します。アラームアクションの詳細については、「Amazon CloudWatch ユーザーガイド」の「[EC2 インスタンスを停止、終了、再起動、または復旧するアラームを作成する](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "RecoveryInstance" : {
      "Description" : "The EC2 instance ID to associate this alarm with.",
      "Type" : "AWS::EC2::Instance::Id"
    }
  },
  "Resources": {
    "RecoveryTestAlarm": {
      "Type": "AWS::CloudWatch::Alarm",
      "Properties": {
        "AlarmDescription": "Trigger a recovery when instance status check fails for 15 consecutive minutes.",
        "Namespace": "AWS/EC2" ,
        "MetricName": "StatusCheckFailed_System",
        "Statistic": "Minimum",
        "Period": "60",
        "EvaluationPeriods": "15",
        "ComparisonOperator": "GreaterThanThreshold",
        "Threshold": "0",
        "AlarmActions": [ {"Fn::Join" : [ "", [ "arn:aws:automate:", { "Ref" : "AWS::Region" }, ":ec2:recover" ] ] } ],
        "Dimensions": [{"Name": "InstanceId", "Value": {"Ref": "RecoveryInstance"}}]
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  RecoveryInstance:
    Description: The EC2 instance ID to associate this alarm with.
    Type: AWS::EC2::Instance::Id
Resources:
  RecoveryTestAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: Trigger a recovery when instance status check fails for 15
        consecutive minutes.
      Namespace: AWS/EC2
      MetricName: StatusCheckFailed_System
      Statistic: Minimum
      Period: '60'
      EvaluationPeriods: '15'
      ComparisonOperator: GreaterThanThreshold
      Threshold: '0'
      AlarmActions: [ !Sub "arn:aws:automate:${AWS::Region}:ec2:recover" ]
      Dimensions:
        - Name: InstanceId
          Value: !Ref: RecoveryInstance
```

基本ダッシュボードを作成する

次の例では、CPU 使用率を表示する 1 つのメトリクスウィジェットとメッセージを表示する 1 つのテキストウィジェットからなる単純な CloudWatch ダッシュボードを作成します。

JSON

```
{
  "BasicDashboard": {
    "Type": "AWS::CloudWatch::Dashboard",
    "Properties": {
      "DashboardName": "Dashboard1",
      "DashboardBody": "{\\\"widgets\\\": [{\\\"type\\\": \\\"metric\\\", \\\"x\\\": 0, \\\"y\\\": 0, \\\"width\\\": 12, \\\"height\\\": 6, \\\"properties\\\": {\\\"metrics\\\": [ [\\\"AWS/EC2\\\", \\\"CPUUtilization\\\", \\\"InstanceId\\\", \\\"i-012345\\\" ] ], \\\"period\\\": 300, \\\"stat\\\": \\\"Average\\\", \\\"region\\\": \\\"us-east-1\\\", \\\"title\\\": \\\"EC2 Instance CPU\\\" } }, {\\\"type\\\": \\\"text\\\", \\\"x\\\": 0, \\\"y\\\": 7, \\\"width\\\": 3, \\\"height\\\": 3, \\\"properties\\\": {\\\"markdown\\\": \\\"Hello world\\\" } } ] }"
    }
  }
}
```

```
    }
  }
}
```

YAML

```
BasicDashboard:
  Type: AWS::CloudWatch::Dashboard
  Properties:
    DashboardName: Dashboard1
    DashboardBody: '{"widgets":
[{"type":"metric","x":0,"y":0,"width":12,"height":6,"properties":{"metrics":[["AWS/
EC2","CPUUtilization","InstanceId","i-012345"]],"period":300,"stat":"Average","region":"us-
east-1","title":"EC2 Instance CPU"}},
{"type":"text","x":0,"y":7,"width":3,"height":3,"properties":{"markdown":"Hello
world"}}]}'
```

サイドバイサイドウィジェットでダッシュボードを作成する

次の例では、2つのメトリクスウィジェットを並べて表示するダッシュボードを作成します。

JSON

```
{
  "DashboardSideBySide": {
    "Type": "AWS::CloudWatch::Dashboard",
    "Properties": {
      "DashboardName": "Dashboard1",
      "DashboardBody": "{\"widgets\": [{\"type\": \"metric\", \"x\": 0, \"y\": 0,
\\\"width\\\": 12, \\\"height\\\": 6, \\\"properties\\\": {\\\"metrics\\\": [\\\"AWS/EC2\\\", \\\"CPUUtilization
\\\", \\\"InstanceId\\\", \\\"i-012345\\\"]], \\\"period\\\": 300, \\\"stat\\\": \\\"Average\\\", \\\"region\\\":
\\\"us-east-1\\\", \\\"title\\\": \\\"EC2 Instance CPU\\\"}}, {\\\"type\\\": \\\"metric\\\", \\\"x\\\": 12, \\\"y\\\": 0,
\\\"width\\\": 12, \\\"height\\\": 6, \\\"properties\\\": {\\\"metrics\\\": [\\\"AWS/S3\\\", \\\"BucketSizeBytes\\\",
\\\"BucketName\\\", \\\"amzn-s3-demo-bucket\\\"]], \\\"period\\\": 86400, \\\"stat\\\": \\\"Maximum\\\", \\\"region
\\\": \\\"us-east-1\\\", \\\"title\\\": \\\"amzn-s3-demo-bucket bytes\\\"}}] }"
    }
  }
}
```

YAML

```
DashboardSideBySide:
  Type: AWS::CloudWatch::Dashboard
```

Properties:

DashboardName: Dashboard1

DashboardBody: '{"widgets":

```
[{"type": "metric", "x": 0, "y": 0, "width": 12, "height": 6, "properties": {"metrics": [{"AWS/EC2", "CPUUtilization", "InstanceId", "i-012345"}], "period": 300, "stat": "Average", "region": "us-east-1", "title": "EC2 Instance CPU"}}, {"type": "metric", "x": 12, "y": 0, "width": 12, "height": 6, "properties": {"metrics": [{"AWS/S3", "BucketSizeBytes", "BucketName", "amzn-s3-demo-bucket"}], "period": 86400, "stat": "Maximum", "region": "us-east-1", "title": "amzn-s3-demo-bucket bytes"}}]}'
```

Amazon CloudWatch Logs テンプレートスニペット

Amazon CloudWatch Logs は、Amazon EC2 インスタンスまたはその他のリソースから、システム、アプリケーション、およびカスタムログファイルを監視できます。CloudFormation を使用してロググループおよびメトリクスフィルタをプロビジョニングし、管理できます。Amazon CloudWatch Logs の詳細については、「[Amazon CloudWatch Logs ユーザーガイド](#)」を参照してください。

トピック

- [Linux インスタンスから CloudWatch Logs へのログの送信](#)
- [Windows インスタンスから CloudWatch Logs へのログの送信](#)
- [関連情報](#)

Linux インスタンスから CloudWatch Logs へのログの送信

次のテンプレートは、CloudWatch Logs 統合を使用して Amazon Linux 2023 でウェブサーバーを設定する方法を示しています。テンプレートは、次のタスクを実行します。

- Apache と PHP をインストールします。
- Apache アクセスログを CloudWatch Logs に転送するように CloudWatch エージェントを設定します。
- CloudWatch エージェントが CloudWatch Logs にログデータを送信できるようにする IAM ロールを設定します。
- カスタムアラームと通知を作成して、404 エラーまたは高帯域幅の使用状況をモニタリングします。

ウェブサーバーからのログイベントは、CloudWatch アラームのメトリクスデータを提供します。2 つのメトリクスフィルタで、ログ情報を CloudWatch メトリクスに変換する方法を記述します。404 メトリクスは 404 の発生数をカウントします。サイズメトリクスはリクエストのサイズを追跡します。2 分間で 3 つ以上の 404 が発生した場合、または平均リクエストサイズが 3500 KB を超える状態が 10 分を超える場合は、2 つの CloudWatch アラームが通知を送信します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Sample template that sets up and configures CloudWatch Logs on Amazon Linux 2023 instance.",
  "Parameters": {
    "KeyName": {
      "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
    },
    "SSHLocation": {
      "Description": "The IP address range that can be used to SSH to the EC2 instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "OperatorEmail": {
      "Description": "Email address to notify when CloudWatch alarms are triggered (404 errors or high bandwidth usage)",
      "Type": "String"
    }
  },
  "Resources": {
    "LogRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
```

```

        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": [
                        "ec2.amazonaws.com"
                    ]
                },
                "Action": [
                    "sts:AssumeRole"
                ]
            }
        ],
        "Path": "/",
        "Policies": [
            {
                "PolicyName": "LogRolePolicy",
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": [
                                "logs:PutLogEvents",
                                "logs:DescribeLogStreams",
                                "logs:DescribeLogGroups",
                                "logs:CreateLogGroup",
                                "logs:CreateLogStream"
                            ],
                            "Resource": "*"
                        }
                    ]
                }
            }
        ]
    },
    "LogRoleInstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [{"Ref": "LogRole"}]
        }
    }
}

```

```

    },
    "WebServerSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Enable HTTP access via port 80 and SSH access via
port 22",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": {"Ref": "SSHLocation"}
          }
        ]
      }
    },
    "WebServerHost": {
      "Type": "AWS::EC2::Instance",
      "Metadata": {
        "Comment": "Install a simple PHP application on Amazon Linux 2023",
        "AWS::CloudFormation::Init": {
          "config": {
            "packages": {
              "dnf": {
                "httpd": [],
                "php": [],
                "php-fpm": []
              }
            },
            "files": {
              "/etc/amazon-cloudwatch-agent/amazon-cloudwatch-
agent.json": {
                "content": {
                  "logs": {
                    "logs_collected": {
                      "files": {
                        "collect_list": [{

```

```

        "file_path": "/var/log/httpd/
access_log",
        "log_group_name": {"Ref":
        "WebServerLogGroup"},
        "log_stream_name": "{instance_id}/
apache.log",
        "timestamp_format": "%d/%b/%Y:%H:
%M:%S %z"
    }
}
}
},
{
    "mode": "000644",
    "owner": "root",
    "group": "root"
},
"/var/www/html/index.php": {
    "content": "<?php\\necho '<h1>AWS CloudFormation sample
PHP application on Amazon Linux 2023</h1>';\\n?>\\n",
    "mode": "000644",
    "owner": "apache",
    "group": "apache"
},
"/etc/cfn/cfn-hup.conf": {
    "content": {
        "Fn::Join": [
            "",
            [
                "[main]\\n",
                "stack=",
                {"Ref": "AWS::StackId"},
                "\\n",
                "region=",
                {"Ref": "AWS::Region"},
                "\\n"
            ]
        ]
    },
    "mode": "000400",
    "owner": "root",
    "group": "root"
},
"/etc/cfn/hooks.d/cfn-auto-reloader.conf": {

```

```

        "content": {
            "Fn::Join": [
                "",
                [
                    "[cfn-auto-reloader-hook]\n",
                    "triggers=post.update\n",
                    "path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init\n",
                    "action=/opt/aws/bin/cfn-init -s ",
                    {"Ref": "AWS::StackId"},
                    " -r WebServerHost ",
                    " --region      ",
                    {"Ref": "AWS::Region"},
                    "\n",
                    "runas=root\n"
                ]
            ]
        },
        "services": {
            "systemd": {
                "httpd": {
                    "enabled": "true",
                    "ensureRunning": "true"
                },
                "php-fpm": {
                    "enabled": "true",
                    "ensureRunning": "true"
                }
            }
        },
        "CreationPolicy": {
            "ResourceSignal": {
                "Timeout": "PT5M"
            }
        },
        "Properties": {
            "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}",
            "KeyName": {"Ref": "KeyName"},

```

```

      "InstanceType": "t3.micro",
      "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
      "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
      "UserData": {"Fn::Base64": {"Fn::Join": [ "", [
        "#!/bin/bash\n",
        "dnf update -y aws-cfn-bootstrap\n",
        "dnf install -y amazon-cloudwatch-agent\n",
        "/opt/aws/bin/cfn-init -v --stack ", {"Ref": "AWS::StackName"}, "
--resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n",
        "\n",
        "# Verify Apache log directory exists and create if needed\n",
        "mkdir -p /var/log/httpd\n",
        "\n",
        "# Start CloudWatch agent\n",
        "/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -
a fetch-config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json
-s\n",
        "\n",
        "# Signal success\n",
        "/opt/aws/bin/cfn-signal -e $? --stack ", {"Ref":
"AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n"
]]}}
    }
  },
  "WebServerLogGroup": {
    "Type": "AWS::Logs::LogGroup",
    "DeletionPolicy": "Retain",
    "UpdateReplacePolicy": "Retain",
    "Properties": {
      "RetentionInDays": 7
    }
  },
  "404MetricFilter": {
    "Type": "AWS::Logs::MetricFilter",
    "Properties": {
      "LogGroupName": {"Ref": "WebServerLogGroup"},
      "FilterPattern": "[ip, identity, user_id, timestamp, request,
status_code = 404, size, ...]",
      "MetricTransformations": [
        {
          "MetricValue": "1",
          "MetricNamespace": "test/404s",
          "MetricName": "test404Count"
        }
      ]
    }
  }
}

```

```

        ]
    },
    "BytesTransferredMetricFilter": {
        "Type": "AWS::Logs::MetricFilter",
        "Properties": {
            "LogGroupName": {"Ref": "WebServerLogGroup"},
            "FilterPattern": "[ip, identity, user_id, timestamp, request,
status_code, size, ...]",
            "MetricTransformations": [
                {
                    "MetricValue": "$size",
                    "MetricNamespace": "test/BytesTransferred",
                    "MetricName": "testBytesTransferred"
                }
            ]
        }
    },
    "404Alarm": {
        "Type": "AWS::CloudWatch::Alarm",
        "Properties": {
            "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
            "MetricName": "test404Count",
            "Namespace": "test/404s",
            "Statistic": "Sum",
            "Period": "60",
            "EvaluationPeriods": "2",
            "Threshold": "2",
            "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
            "ComparisonOperator": "GreaterThanThreshold"
        }
    },
    "BandwidthAlarm": {
        "Type": "AWS::CloudWatch::Alarm",
        "Properties": {
            "AlarmDescription": "The average volume of traffic is greater 3500 KB
over 10 minutes",
            "MetricName": "testBytesTransferred",
            "Namespace": "test/BytesTransferred",
            "Statistic": "Average",
            "Period": "300",
            "EvaluationPeriods": "2",
            "Threshold": "3500",

```

```

        "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
        "ComparisonOperator": "GreaterThanThreshold"
    },
    "AlarmNotificationTopic": {
        "Type": "AWS::SNS::Topic",
        "Properties": {
            "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
"email"}]
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        },
        "PublicIP": {
            "Description": "Public IP address of the web server",
            "Value": {"Fn::GetAtt": ["WebServerHost", "PublicIp"]}
        },
        "CloudWatchLogGroupName": {
            "Description": "The name of the CloudWatch log group",
            "Value": {"Ref": "WebServerLogGroup"}
        }
    }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Sample template that sets up and configures CloudWatch Logs on Amazon
Linux 2023 instance.
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.

```


SSHLocation:

Description: The IP address range that can be used to SSH to the EC2 instances

Type: String

MinLength: '9'

MaxLength: '18'

Default: 0.0.0.0/0

AllowedPattern: '(\d{1,3})\.\d{1,3})\.\d{1,3})\.\d{1,3})/(\d{1,2})'

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

OperatorEmail:

Description: Email address to notify when CloudWatch alarms are triggered (404 errors or high bandwidth usage)

Type: String

Resources:**LogRole:**

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: 2012-10-17

Statement:

- Effect: Allow

Principal:

Service:

- ec2.amazonaws.com

Action:

- 'sts:AssumeRole'

Path: /

Policies:

- PolicyName: LogRolePolicy

PolicyDocument:

Version: 2012-10-17

Statement:

- Effect: Allow

Action:

- 'logs:PutLogEvents'
- 'logs:DescribeLogStreams'
- 'logs:DescribeLogGroups'
- 'logs:CreateLogGroup'
- 'logs:CreateLogStream'

Resource: '*'

LogRoleInstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles:

```

    - !Ref LogRole
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation
WebServerHost:
  Type: AWS::EC2::Instance
  Metadata:
    Comment: Install a simple PHP application on Amazon Linux 2023
    'AWS::CloudFormation::Init':
      config:
        packages:
          dnf:
            httpd: []
            php: []
            php-fpm: []
        files:
          /etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json:
            content: !Sub |
              {
                "logs": {
                  "logs_collected": {
                    "files": {
                      "collect_list": [
                        {
                          "file_path": "/var/log/httpd/access_log",
                          "log_group_name": "${WebServerLogGroup}",
                          "log_stream_name": "{instance_id}/apache.log",
                          "timestamp_format": "%d/%b/%Y:%H:%M:%S %z"
                        }
                      ]
                    }
                  }
                }
              }

```

```

        mode: '000644'
        owner: root
        group: root
/var/www/html/index.php:
        content: |
            <?php echo '<h1>AWS CloudFormation sample PHP application on Amazon
Linux 2023</h1>';
            ?>
        mode: '000644'
        owner: apache
        group: apache
/etc/cfn/cfn-hup.conf:
        content: !Sub |
            [main]
            stack=${AWS::StackId}
            region=${AWS::Region}
        mode: '000400'
        owner: root
        group: root
/etc/cfn/hooks.d/cfn-auto-reloader.conf:
        content: !Sub |
            [cfn-auto-reloader-hook]
            triggers=post.update
            path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init
            action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerHost --
region ${AWS::Region}
            runas=root
    services:
        systemd:
            httpd:
                enabled: 'true'
                ensureRunning: 'true'
            php-fpm:
                enabled: 'true'
                ensureRunning: 'true'
    CreationPolicy:
        ResourceSignal:
            Timeout: PT5M
    Properties:
        ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-
kernel-6.1-x86_64}}'
        KeyName: !Ref KeyName
        InstanceType: t3.micro
        SecurityGroupIds:

```

```

    - !Ref WebServerSecurityGroup
    IamInstanceProfile: !Ref LogRoleInstanceProfile
    UserData: !Base64
      Fn::Sub: |
        #!/bin/bash
        dnf update -y aws-cfn-bootstrap
        dnf install -y amazon-cloudwatch-agent
        /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServerHost
--region ${AWS::Region}

        # Verify Apache log directory exists and create if needed
        mkdir -p /var/log/httpd

        # Start CloudWatch agent
        /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json -s

        # Signal success
        /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
WebServerHost --region ${AWS::Region}
        echo "Done"
  WebServerLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      RetentionInDays: 7
  404MetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: >-
        [ip, identity, user_id, timestamp, request, status_code = 404, size, ...]
      MetricTransformations:
        - MetricValue: '1'
          MetricNamespace: test/404s
          MetricName: test404Count
  BytesTransferredMetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: '[ip, identity, user_id, timestamp, request, status_code,
size, ...]'
      MetricTransformations:

```

```
- MetricValue: $size
  MetricNamespace: test/BytesTransferred
  MetricName: testBytesTransferred
```

404Alarm:

```
Type: AWS::CloudWatch::Alarm
```

Properties:

```
  AlarmDescription: The number of 404s is greater than 2 over 2 minutes
```

```
  MetricName: test404Count
```

```
  Namespace: test/404s
```

```
  Statistic: Sum
```

```
  Period: '60'
```

```
  EvaluationPeriods: '2'
```

```
  Threshold: '2'
```

AlarmActions:

```
- !Ref AlarmNotificationTopic
```

```
  ComparisonOperator: GreaterThanThreshold
```

BandwidthAlarm:

```
Type: AWS::CloudWatch::Alarm
```

Properties:

```
  AlarmDescription: The average volume of traffic is greater 3500 KB over 10
minutes
```

```
  MetricName: testBytesTransferred
```

```
  Namespace: test/BytesTransferred
```

```
  Statistic: Average
```

```
  Period: '300'
```

```
  EvaluationPeriods: '2'
```

```
  Threshold: '3500'
```

AlarmActions:

```
- !Ref AlarmNotificationTopic
```

```
  ComparisonOperator: GreaterThanThreshold
```

AlarmNotificationTopic:

```
Type: AWS::SNS::Topic
```

Properties:**Subscription:**

```
- Endpoint: !Ref OperatorEmail
```

```
  Protocol: email
```

Outputs:**InstanceId:**

```
  Description: The instance ID of the web server
```

```
  Value: !Ref WebServerHost
```

WebsiteURL:

```
  Value: !Sub 'http://${WebServerHost.PublicDnsName}'
```

```
  Description: URL for the web server
```

PublicIP:

```
Description: Public IP address of the web server
Value: !GetAtt WebServerHost.PublicIp
CloudWatchLogGroupName:
Description: The name of the CloudWatch log group
Value: !Ref WebServerLogGroup
```

Windows インスタンスから CloudWatch Logs へのログの送信

次のテンプレートは、Windows 2012 R2 インスタンス用に CloudWatch Logs を設定します。

Windows の CloudWatch Logs エージェント (Windows 2012 R2 および Windows 2016 AMI の SSM エージェント) は、起動後にのみログを送信するため、起動前に生成されたログは送信されません。この問題を対処するため、テンプレートは次のようにして、ログが書き込まれる前にエージェントを確実に起動します。

- `cfn-init config` の最初の `configSets` 項目としてエージェントのセットアップを行います。
- `waitAfterCompletion` を使用して、コマンドがエージェントを起動した後で一時的に停止を挿入します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Sample template that sets up and configures CloudWatch Logs on
Windows 2012R2 instance.",
  "Parameters": {
    "KeyPair": {
      "Description": "Name of an existing EC2 KeyPair to enable RDP access to the
instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
    },
    "RDPLocation": {
      "Description": "The IP address range that can be used to RDP to the EC2
instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})",

```

```

        "ConstraintDescription": "must be a valid IP CIDR range of the form
x.x.x.x/x."
    },
    "OperatorEmail": {
        "Description": "Email address to notify when CloudWatch alarms are
triggered (404 errors)",
        "Type": "String"
    }
},
"Resources": {
    "WebServerSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Enable HTTP access via port 80 and RDP access via
port 3389",
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": "80",
                    "ToPort": "80",
                    "CidrIp": "0.0.0.0/0"
                },
                {
                    "IpProtocol": "tcp",
                    "FromPort": "3389",
                    "ToPort": "3389",
                    "CidrIp": {"Ref": "RDPLocation"}
                }
            ]
        }
    },
    "LogRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ec2.amazonaws.com"
                            ]
                        }
                    }
                ]
            }
        }
    },

```

```

        "Action": [
            "sts:AssumeRole"
        ]
    },
    ],
    "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
    ],
    "Path": "/",
    "Policies": [
        {
            "PolicyName": "LogRolePolicy",
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "logs:Create*",
                            "logs:PutLogEvents",
                            "s3:GetObject"
                        ],
                        "Resource": [
                            "arn:aws:logs:*:*:*",
                            "arn:aws:s3:::*"
                        ]
                    }
                ]
            }
        }
    ],
    "LogRoleInstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [{"Ref": "LogRole"}]
        }
    },
    "WebServerHost": {
        "Type": "AWS::EC2::Instance",
        "CreationPolicy": {

```



```

        "ResourceSignal": {
            "Timeout": "PT15M"
        }
    },
    "Metadata": {
        "AWS::CloudFormation::Init": {
            "configSets": {
                "config": [
                    "00-ConfigureCWLogs",
                    "01-InstallWebServer",
                    "02-ConfigureApplication",
                    "03-Finalize"
                ]
            },
            "00-ConfigureCWLogs": {
                "files": {
                    "C:\\Program Files\\Amazon\\SSM\\Plugins\\awsCloudWatch\\
\\AWS.EC2.Windows.CloudWatch.json": {
                        "content": {
                            "EngineConfiguration": {
                                "Components": [
                                    {
                                        "FullName":
"\"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch\"",
                                        "Id": "ApplicationEventLog",
                                        "Parameters": {
                                            "Levels": "7",
                                            "LogName": "Application"
                                        }
                                    },
                                    {
                                        "FullName":
"\"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch\"",
                                        "Id": "SystemEventLog",
                                        "Parameters": {
                                            "Levels": "7",
                                            "LogName": "System"
                                        }
                                    },
                                    {
                                        "FullName":
"\"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch\"",
                                        "Id": "SecurityEventLog",
                                        "Parameters": {

```

```

        "Levels": "7",
        "LogName": "Security"
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "EC2ConfigLog",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "ASCII",
            "Filter": "EC2ConfigLog.txt",
            "LogDirectoryPath": "C:\\Program
Files\\Amazon\\Ec2ConfigService\\Logs",
            "TimeZoneKind": "UTC",
            "TimestampFormat": "yyyy-MM-
ddTHH:mm:ss.fffZ:"
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "CfnInitLog",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "ASCII",
            "Filter": "cfn-init.log",
            "LogDirectoryPath": "C:\\cfn\\log",
            "TimeZoneKind": "Local",
            "TimestampFormat": "yyyy-MM-dd
HH:mm:ss,fff"
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "IISLogs",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "UTF-8",
            "Filter": "",
            "LineCount": "3",
            "LogDirectoryPath": "C:\\inetpub\\
logs\\LogFiles\\W3SVC1",

```

```

        "TimeZoneKind": "UTC",
        "TimestampFormat": "yyyy-MM-dd

HH:mm:ss"
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E
        "Id": "MemoryPerformanceCounter",
        "Parameters": {
            "CategoryName": "Memory",
            "CounterName": "Available MBytes",
            "DimensionName": "",
            "DimensionValue": "",
            "InstanceName": "",
            "MetricName": "Memory",
            "Unit": "Megabytes"
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchApplicationEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/

ApplicationEventLog",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchSystemEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/

SystemEventLog",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    }
}

```

```

    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchSecurityEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/
SecurityEventLog",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchEC2ConfigLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/
EC2ConfigLog",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchCfnInitLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/
CfnInitLog",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchIISLogs",

```

```

        "Parameters": {
            "AccessKey": "",
            "LogGroup": {"Ref": "LogGroup"},
            "LogStream": "{instance_id}/

IISLogs",

            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatch",
        "Parameters": {
            "AccessKey": "",
            "NameSpace": "Windows/Default",
            "Region": {"Ref": "AWS::Region"},
            "SecretKey": ""
        }
    }
],
"Flows": {
    "Flows": [

"ApplicationEventLog,CloudWatchApplicationEventLog",

"SystemEventLog,CloudWatchSystemEventLog",

"SecurityEventLog,CloudWatchSecurityEventLog",

"EC2ConfigLog,CloudWatchEC2ConfigLog",
"CfnInitLog,CloudWatchCfnInitLog",
"IISLogs,CloudWatchIISLogs",
"MemoryPerformanceCounter,CloudWatch"

    ]
},
    "PollInterval": "00:00:05"
},
    "IsEnabled": true
}
    },
    "commands": {
        "0-enableSSM": {

```

```

        "command": "powershell.exe -Command \"Set-Service -Name
AmazonSSMAgent -StartupType Automatic\" ",
        "waitAfterCompletion": "0"
    },
    "1-restartSSM": {
        "command": "powershell.exe -Command \"Restart-Service
AmazonSSMAgent \",
        "waitAfterCompletion": "30"
    }
},
"01-InstallWebServer": {
    "commands": {
        "01_install_webserver": {
            "command": "powershell.exe -Command \"Install-
WindowsFeature Web-Server -IncludeAllSubFeature\",
            "waitAfterCompletion": "0"
        }
    }
},
"02-ConfigureApplication": {
    "files": {
        "c:\\Inetpub\\wwwroot\\index.htm": {
            "content": "<html> <head> <title>Test Application
Page</title> </head> <body> <h1>Congratulations!! Your IIS server is configured.</h1>
</body> </html>"
        }
    }
},
"03-Finalize": {
    "commands": {
        "00_signal_success": {
            "command": {
                "Fn::Sub": "cfn-signal.exe -e 0 --resource
WebServerHost --stack ${AWS::StackName} --region ${AWS::Region}"
            },
            "waitAfterCompletion": "0"
        }
    }
}
},
"Properties": {
    "KeyName": {

```

```

        "Ref": "KeyPair"
    },
    "ImageId": "{{resolve:ssm:/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-English-64Bit-Base}}",
    "InstanceType": "t2.xlarge",
    "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
    "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
    "UserData": {
        "Fn::Base64": {
            "Fn::Join": [
                "",
                [
                    "<script>\n",
                    "wmic product where \"description='Amazon SSM Agent' \"
uninstall\n",
                    "wmic product where \"description='aws-cfn-bootstrap'
\" uninstall \n",
                    "start /wait c:\\\\Windows\\system32\\msiexec /passive /
qn /i https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi\n",
                    "powershell.exe -Command \"iwr https://
s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe -UseBasicParsing -OutFile C:\\\\AmazonSSMAgentSetup.exe\"\\n",
                    "start /wait C:\\\\AmazonSSMAgentSetup.exe /install /
quiet\n",
                    "cfn-init.exe -v -c config -s ", {"Ref":
"AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "
\n",
                    "</script>\n"
                ]
            ]
        }
    },
    "LogGroup": {
        "Type": "AWS::Logs::LogGroup",
        "Properties": {
            "RetentionInDays": 7
        }
    },
    "404MetricFilter": {
        "Type": "AWS::Logs::MetricFilter",
        "Properties": {

```

```

        "LogGroupName": {"Ref": "LogGroup"},
        "FilterPattern": "[timestamps, serverip, method, uri, query, port,
dash, clientip, useragent, status_code = 404, ...]",
        "MetricTransformations": [
            {
                "MetricValue": "1",
                "MetricNamespace": "test/404s",
                "MetricName": "test404Count"
            }
        ]
    },
    "404Alarm": {
        "Type": "AWS::CloudWatch::Alarm",
        "Properties": {
            "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
            "MetricName": "test404Count",
            "Namespace": "test/404s",
            "Statistic": "Sum",
            "Period": "60",
            "EvaluationPeriods": "2",
            "Threshold": "2",
            "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
            "ComparisonOperator": "GreaterThanThreshold"
        }
    },
    "AlarmNotificationTopic": {
        "Type": "AWS::SNS::Topic",
        "Properties": {
            "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
"email"}]
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        }
    },

```



```

    "PublicIP": {
      "Description": "Public IP address of the web server",
      "Value": {"Fn::GetAtt": ["WebServerHost", "PublicIp"]}
    },
    "CloudWatchLogGroupName": {
      "Description": "The name of the CloudWatch log group",
      "Value": {"Ref": "LogGroup"}
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: >-
  Sample template that sets up and configures CloudWatch Logs on Windows 2012R2
  instance.
Parameters:
  KeyPair:
    Description: Name of an existing EC2 KeyPair to enable RDP access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  RDPLocation:
    Description: The IP address range that can be used to RDP to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.\d{1,3})\.\d{1,3})\.\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  OperatorEmail:
    Description: Email address to notify when CloudWatch alarms are triggered (404
    errors)
    Type: String
Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and RDP access via port 3389
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'

```

```
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: '3389'
    ToPort: '3389'
    CidrIp: !Ref RDPLocation
LogRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
  Path: /
  Policies:
    - PolicyName: LogRolePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - 'logs:Create*'
              - 'logs:PutLogEvents'
              - 's3:GetObject'
            Resource:
              - 'arn:aws:logs:*:*:*'
              - 'arn:aws:s3::*:*'
LogRoleInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: /
    Roles:
      - !Ref LogRole
WebServerHost:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT15M
```

```

Metadata:
  'AWS::CloudFormation::Init':
    configSets:
      config:
        - 00-ConfigureCWLogs
        - 01-InstallWebServer
        - 02-ConfigureApplication
        - 03-Finalize
    00-ConfigureCWLogs:
      files:
        'C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch
\AWS.EC2.Windows.CloudWatch.json':
          content: !Sub |
            {
              "EngineConfiguration": {
                "Components": [
                  {
                    "FullName":
"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                    "Id": "ApplicationEventLog",
                    "Parameters": {
                      "Levels": "7",
                      "LogName": "Application"
                    }
                  },
                  {
                    "FullName":
"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                    "Id": "SystemEventLog",
                    "Parameters": {
                      "Levels": "7",
                      "LogName": "System"
                    }
                  },
                  {
                    "FullName":
"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                    "Id": "SecurityEventLog",
                    "Parameters": {
                      "Levels": "7",
                      "LogName": "Security"
                    }
                  }
                ]
              }
            }

```

```

        "FullName":
        "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "EC2ConfigLog",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "ASCII",
            "Filter": "EC2ConfigLog.txt",
            "LogDirectoryPath": "C:\\\\Program Files\\\\Amazon\\
\\Ec2ConfigService\\\\Logs",
            "TimeZoneKind": "UTC",
            "TimestampFormat": "yyyy-MM-ddTHH:mm:ss.fffZ:"
        }
    },
    {
        "FullName":
        "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "CfnInitLog",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "ASCII",
            "Filter": "cfn-init.log",
            "LogDirectoryPath": "C:\\\\cfn\\\\log",
            "TimeZoneKind": "Local",
            "TimestampFormat": "yyyy-MM-dd HH:mm:ss,fff"
        }
    },
    {
        "FullName":
        "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "IISLogs",
        "Parameters": {
            "CultureName": "en-US",
            "Encoding": "UTF-8",
            "Filter": "",
            "LineCount": "3",
            "LogDirectoryPath": "C:\\\\inetpub\\\\logs\\\\LogFiles\\
\\W3SVC1",
            "TimeZoneKind": "UTC",
            "TimestampFormat": "yyyy-MM-dd HH:mm:ss"
        }
    },
    {
        "FullName":
        "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E

```

```

        "Id": "MemoryPerformanceCounter",
        "Parameters": {
            "CategoryName": "Memory",
            "CounterName": "Available MBytes",
            "DimensionName": "",
            "DimensionValue": "",
            "InstanceName": "",
            "MetricName": "Memory",
            "Unit": "Megabytes"
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchApplicationEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/ApplicationEventLog",
            "Region": "${AWS::Region}",
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchSystemEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/SystemEventLog",
            "Region": "${AWS::Region}",
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchSecurityEventLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/SecurityEventLog",
            "Region": "${AWS::Region}",

```

```

        "SecretKey": ""
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchEC2ConfigLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/EC2ConfigLog",
            "Region": "${AWS::Region}",
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchCfnInitLog",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/CfnInitLog",
            "Region": "${AWS::Region}",
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatchIISLogs",
        "Parameters": {
            "AccessKey": "",
            "LogGroup": "${LogGroup}",
            "LogStream": "{instance_id}/IISLogs",
            "Region": "${AWS::Region}",
            "SecretKey": ""
        }
    },
    {
        "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
        "Id": "CloudWatch",
        "Parameters": {

```

```

        "AccessKey": "",
        "Namespace": "Windows/Default",
        "Region": "${AWS::Region}",
        "SecretKey": ""
    }
},
"Flows": {
    "Flows": [
        "ApplicationEventLog,CloudWatchApplicationEventLog",
        "SystemEventLog,CloudWatchSystemEventLog",
        "SecurityEventLog,CloudWatchSecurityEventLog",
        "EC2ConfigLog,CloudWatchEC2ConfigLog",
        "CfnInitLog,CloudWatchCfnInitLog",
        "IISLogs,CloudWatchIISLogs",
        "MemoryPerformanceCounter,CloudWatch"
    ]
},
"PollInterval": "00:00:05"
},
"IsEnabled": true
}
commands:
  0-enableSSM:
    command: >-
      powershell.exe -Command "Set-Service -Name AmazonSSMAgent
        -StartupType Automatic"
    waitAfterCompletion: '0'
  1-restartSSM:
    command: powershell.exe -Command "Restart-Service AmazonSSMAgent "
    waitAfterCompletion: '30'
01-InstallWebServer:
  commands:
    01_install_webserver:
      command: >-
        powershell.exe -Command "Install-WindowsFeature Web-Server
          -IncludeAllSubFeature"
      waitAfterCompletion: '0'
02-ConfigureApplication:
  files:
    'c:\Inetpub\wwwroot\index.htm':
      content: >-
        <html> <head> <title>Test Application Page</title> </head>
        <body> <h1>Congratulations !! Your IIS server is

```

```

        configured.</h1> </body> </html>
03-Finalize:
  commands:
    00_signal_success:
      command: !Sub >-
        cfn-signal.exe -e 0 --resource WebServerHost --stack
          ${AWS::StackName} --region ${AWS::Region}
      waitAfterCompletion: '0'
  Properties:
    KeyName: !Ref KeyPair
    ImageId: "{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2012-
R2_RTM-English-64Bit-Base}}"
    InstanceType: t2.xlarge
    SecurityGroupIds:
      - !Ref WebServerSecurityGroup
    IamInstanceProfile: !Ref LogRoleInstanceProfile
    UserData: !Base64
      'Fn::Sub': >
        <script>

          wmic product where "description='Amazon SSM Agent' " uninstall

          wmic product where "description='aws-cfn-bootstrap' " uninstall

          start /wait c:\\Windows\\system32\\msiexec /passive /qn /i
            https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi

          powershell.exe -Command "iwr
            https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe
            -UseBasicParsing -OutFile C:\\AmazonSSMAgentSetup.exe"

          start /wait C:\\AmazonSSMAgentSetup.exe /install /quiet

          cfn-init.exe -v -c config -s ${AWS::StackName} --resource
            WebServerHost --region ${AWS::Region}

        </script>
    LogGroup:
      Type: AWS::Logs::LogGroup
      Properties:
        RetentionInDays: 7
    404MetricFilter:

```



```
Type: AWS::Logs::MetricFilter
Properties:
  LogGroupName: !Ref LogGroup
  FilterPattern: >-
    [timestamps, serverip, method, uri, query, port, dash, clientip,
     useragent, status_code = 404, ...]
  MetricTransformations:
    - MetricValue: '1'
      MetricNamespace: test/404s
      MetricName: test404Count
404Alarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: The number of 404s is greater than 2 over 2 minutes
    MetricName: test404Count
    Namespace: test/404s
    Statistic: Sum
    Period: '60'
    EvaluationPeriods: '2'
    Threshold: '2'
    AlarmActions:
      - !Ref AlarmNotificationTopic
    ComparisonOperator: GreaterThanThreshold
AlarmNotificationTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
      - Endpoint: !Ref OperatorEmail
        Protocol: email
Outputs:
  InstanceId:
    Description: The instance ID of the web server
    Value: !Ref WebServerHost
  WebsiteURL:
    Value: !Sub 'http://${WebServerHost.PublicDnsName}'
    Description: URL for the web server
  PublicIP:
    Description: Public IP address of the web server
    Value: !GetAtt
      - WebServerHost
      - PublicIp
  CloudWatchLogGroupName:
    Description: The name of the CloudWatch log group
```

Value: !Ref LogGroup

関連情報

CloudWatch Logs リソースの詳細については、「[AWS::Logs::LogGroup](#)」または「[AWS::Logs::MetricFilter](#)」を参照してください。

Amazon DynamoDB テンプレートスニペット

トピック

- [Amazon DynamoDB テーブルを含む アプリケーションの Auto Scaling](#)
- [関連情報](#)

Amazon DynamoDB テーブルを含む アプリケーションの Auto Scaling

この例では、AWS::DynamoDB::Table リソースの Application Auto Scaling をセットアップします。このテンプレートでは、テーブル向けに TargetTrackingScaling スループットをスケールアップする WriteCapacityUnits スケーリングポリシーを定義します。

JSON

```
{
  "Resources": {
    "DDBTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "AttributeDefinitions": [
          {
            "AttributeName": "ArtistId",
            "AttributeType": "S"
          },
          {
            "AttributeName": "Concert",
            "AttributeType": "S"
          },
          {
            "AttributeName": "TicketSales",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
```

```

        {
            "AttributeName": "ArtistId",
            "KeyType": "HASH"
        },
        {
            "AttributeName": "Concert",
            "KeyType": "RANGE"
        }
    ],
    "GlobalSecondaryIndexes": [
        {
            "IndexName": "GSI",
            "KeySchema": [
                {
                    "AttributeName": "TicketSales",
                    "KeyType": "HASH"
                }
            ],
            "Projection": {
                "ProjectionType": "KEYS_ONLY"
            },
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 5,
                "WriteCapacityUnits": 5
            }
        }
    ],
    "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 5
    }
},
"WriteCapacityScalableTarget": {
    "Type": "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties": {
        "MaxCapacity": 15,
        "MinCapacity": 5,
        "ResourceId": {
            "Fn::Join": [
                "/",
                [
                    "table",
                    {

```



```

        AttributeType: "S"
    - AttributeName: "TicketSales"
      AttributeType: "S"
  KeySchema:
    - AttributeName: "ArtistId"
      KeyType: "HASH"
    - AttributeName: "Concert"
      KeyType: "RANGE"
  GlobalSecondaryIndexes:
    - IndexName: "GSI"
      KeySchema:
        - AttributeName: "TicketSales"
          KeyType: "HASH"
      Projection:
        ProjectionType: "KEYS_ONLY"
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
      ResourceId: !Join
        - /
        - - table
          - !Ref DDBTable
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ServiceNamespace: dynamodb
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteAutoScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0

```

```
ScaleInCooldown: 60
ScaleOutCooldown: 60
PredefinedMetricSpecification:
  PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

関連情報

詳細については、AWS データベースログのブログ投稿「[How to use CloudFormation to configure auto scaling for Amazon DynamoDB tables and indexes](#)」を参照してください。

DynamoDB リソースの詳細については、「[AWS::DynamoDB::Table](#)」を参照してください。

Amazon EC2 CloudFormation テンプレートスニペット

Amazon EC2 は、AWS クラウド内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 を使用すると、必要な数 (またはそれ以下) の仮想サーバーの起動、セキュリティおよびネットワーキングの構成、ストレージの管理ができます。インスタンスと呼ばれるこれらの仮想サーバーは、さまざまなオペレーティングシステムやアプリケーションを実行でき、特定の要件に合わせてカスタマイズできます。Amazon EC2 は、需要の変化や使用量の急増に対応して簡単にスケールアップやスケールダウンができます。

CloudFormation テンプレートを使用して、Amazon EC2 インスタンスをインフラストラクチャの一部として定義およびプロビジョニングできます。テンプレートを使用すると、Amazon EC2 リソースのデプロイを繰り返し可能で一貫性のある方法で簡単に管理および自動化できます。

以下のサンプルテンプレートスニペットでは、Amazon EC2 の CloudFormation リソースまたはコンポーネントについて説明しています。これらのスニペットはテンプレートに統合するように設計されており、単独で実行することを意図したものではありません。

スニペットカテゴリ

- [CloudFormation で Amazon EC2 インスタンスを設定する](#)
- [CloudFormation を使用して起動テンプレートを作成する](#)
- [CloudFormation でセキュリティグループを管理する](#)
- [Elastic IP アドレスを CloudFormation と割り当ておよび関連付けする](#)
- [CloudFormation を使用して Amazon VPC リソースを設定する](#)

CloudFormation で Amazon EC2 インスタンスを設定する

以下のスニペットは、CloudFormation を使用して Amazon EC2 インスタンスを設定する方法を示しています。

スニペットカテゴリ

- [Amazon EC2 の全般設定](#)
- [インスタンスのブロックデバイスマッピングの指定](#)

Amazon EC2 の全般設定

以下のスニペットは、CloudFormation を使用した Amazon EC2 インスタンスの全般設定を示しています。

例

- [指定されたアベイラビリティーゾーンに Amazon EC2 インスタンスを作成する](#)
- [EBS ボリュームとユーザーデータを使用した Amazon EC2 インスタンスの設定](#)
- [Amazon EC2 インスタンスを起動するためのユーザーデータに DynamoDB テーブル名を定義する](#)
- [DeletionPolicy を使用した Amazon EBS ボリュームの作成](#)

指定されたアベイラビリティーゾーンに Amazon EC2 インスタンスを作成する

次のスニペットは、[AWS::EC2::Instance](#) リソースを使用した Amazon EC2 インスタンスを作成しています。アベイラビリティーゾーンのコードは、リージョンコードとそれに続く文字識別子です。インスタンスは 1 つのアベイラビリティーゾーンで起動できます。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "AvailabilityZone": "aa-example-1a",
    "ImageId": "ami-1234567890abcdef0"
  }
}
```

YAML

```
Ec2Instance:
```

```
Type: AWS::EC2::Instance
Properties:
  AvailabilityZone: aa-example-1a
  ImageId: ami-1234567890abcdef0
```

EBS ボリュームとユーザーデータを使用した Amazon EC2 インスタンスの設定

次のスニペットは、タグ、EBS ボリューム、およびユーザーデータを使用して Amazon EC2 インスタンスを作成します。[AWS::EC2::Instance](#) リソースを使用します。同じテンプレートで [AWS::EC2::SecurityGroup](#) リソース、[AWS::SNS::Topic](#) リソース、[AWS::EC2::Volume](#) リソースを定義する必要があります。テンプレートの Parameters セクションに KeyName を定義する必要があります。

タグを使用すると、AWS リソースを目的、所有者、環境など好みに基づいて分類できます。ユーザーデータにより、起動時にカスタムスクリプトやデータをインスタンスにプロビジョニングできます。このデータにより、タスクの自動化、ソフトウェア設定、パッケージのインストール、および初期化中のインスタンスでのその他のアクションが容易になります。

リソースのタグ付けの詳細については、『Amazon EC2 ユーザーガイド』の「[Amazon EC2 リソースのタグ付け](#)」を参照してください。

ユーザーデータの詳細については、『Amazon EC2 ユーザーガイド』の「[インスタンスメタデータを使用して EC2 インスタンスを管理する](#)」を参照してください。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "KeyName": { "Ref": "KeyName" },
    "SecurityGroups": [ { "Ref": "Ec2SecurityGroup" } ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [ ":", [
          "PORT=80",
          "TOPIC=",
          { "Ref": "MySNSTopic" }
        ]
        ]
      }
    },
    "InstanceType": "aa.size",
```



```
"AvailabilityZone": "aa-example-1a",
"ImageId": "ami-1234567890abcdef0",
"Volumes": [
  {
    "VolumeId": { "Ref": "MyVolumeResource" },
    "Device": "/dev/sdk"
  }
],
"Tags": [ { "Key": "Name", "Value": "MyTag" } ]
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    KeyName: !Ref KeyName
    SecurityGroups:
      - !Ref Ec2SecurityGroup
    UserData:
      Fn::Base64:
        Fn::Join:
          - ":"
          - - "PORT=80"
            - "TOPIC="
            - !Ref MySNSTopic
    InstanceType: aa.size
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
    Volumes:
      - VolumeId: !Ref MyVolumeResource
        Device: "/dev/sdk"
    Tags:
      - Key: Name
        Value: MyTag
```

Amazon EC2 インスタンスを起動するためのユーザーデータに DynamoDB テーブル名を定義する

次のスニペットは Amazon EC2 インスタンスを作成し、起動時にインスタンスに渡すユーザーデータに DynamoDB テーブル名を定義します。[AWS::EC2::Instance](#) リソースを使用します。起動時に EC2 インスタンスを渡すパラメータまたは動的値をユーザーデータで定義できます。

ユーザーデータの詳細については、『Amazon EC2 ユーザーガイド』の「[インスタンスメタデータを使用して EC2 インスタンスを管理する](#)」を参照してください。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [
          "",
          [
            "TableName=",
            {
              "Ref": "DynamoDBTableName"
            }
          ]
        ]
      },
      "AvailabilityZone": "aa-example-1a",
      "ImageId": "ami-1234567890abcdef0"
    }
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    UserData:
      Fn::Base64:
        Fn::Join:
          - ''
          - - 'TableName='
            - Ref: DynamoDBTableName
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

DeletionPolicy を使用した Amazon EBS ボリュームの作成

以下のスニペットは、Amazon EC2 [AWS::EC2::Volume](#) リソースを使用して Amazon EBS ボリュームを作成します。Size または SnapshotID プロパティを使用してボリュームを定義できますが、両方を使用することはできません。スタックが削除されたときにボリュームのスナップショットを作成するように DeletionPolicy 属性が設定されます。

DeletionPolicy 属性の詳細については、「[DeletionPolicy 属性](#)」を参照してください。

Amazon EBS ボリュームの作成の詳細については、「[Amazon EBS ボリュームの作成](#)」を参照してください。

JSON

このスニペットは、指定されたサイズの Amazon EBS ボリュームを作成します。サイズは 10 に設定されていますが、必要に応じて調整できます。[AWS::EC2::Volume](#) リソースでは、サイズまたはスナップショット ID のいずれかを指定できますが、両方は指定できません。

```
"MyEBSVolume": {
  "Type": "AWS::EC2::Volume",
  "Properties": {
    "Size": "10",
    "AvailabilityZone": {
      "Ref": "AvailabilityZone"
    }
  },
  "DeletionPolicy": "Snapshot"
}
```

このスニペットは、指定されたスナップショット ID を使用した Amazon EBS ボリュームの作成です。[AWS::EC2::Volume](#) リソースでは、サイズまたはスナップショット ID のいずれかを指定できますが、両方は指定できません。

```
"MyEBSVolume": {
  "Type": "AWS::EC2::Volume",
  "Properties": {
    "SnapshotId" : "snap-1234567890abcdef0",
    "AvailabilityZone": {
      "Ref": "AvailabilityZone"
    }
  },
  "DeletionPolicy": "Snapshot"
}
```

```
}
```

YAML

このスニペットは、指定されたサイズの Amazon EBS ボリュームを作成します。サイズは 10 に設定されていますが、必要に応じて調整できます。[AWS::EC2::Volume](#) リソースでは、サイズまたはスナップショット ID のいずれかを指定できますが、両方は指定できません。

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 10
    AvailabilityZone:
      Ref: AvailabilityZone
    DeletionPolicy: Snapshot
```

このスニペットは、指定されたスナップショット ID を使用した Amazon EBS ボリュームの作成です。[AWS::EC2::Volume](#) リソースでは、サイズまたはスナップショット ID のいずれかを指定できますが、両方は指定できません。

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    SnapshotId: snap-1234567890abcdef0
    AvailabilityZone:
      Ref: AvailabilityZone
    DeletionPolicy: Snapshot
```

インスタンスのブロックデバイスマッピングの指定

ブロックデバイスマッピングでは、インスタンスにアタッチする、インスタンスストアボリュームと EBS ボリュームを含むブロックデバイスを定義します。ブロックデバイスマッピングは、AMI 作成時に、AMI から起動されるすべてのインスタンスによって使用されるように指定できます。また、インスタンスの起動時にブロックデバイスマッピングを指定することもできます。起動したインスタンスの AMI ですでに指定されているマッピングがこのマッピングによって上書きされるようにします。

以下のテンプレートスニペットを使用して、[AWS::EC2::Instance](#) リソースの `BlockDeviceMappings` プロパティを使用して EBS またはインスタンスストアボリュームのブロックデバイスマッピングを指定できます。

ブロックデバイスマッピングに関する詳細については、『Amazon EC2 ユーザーガイド』の「[Amazon EC2 インスタンスでのボリュームのブロックデバイスマッピング](#)」を参照してください。

シナリオ

- [2 つの EBS ボリュームのブロックデバイスマッピングを指定します。](#)
- [インスタンスストアボリュームのブロックデバイスマッピングの指定](#)

2 つの EBS ボリュームのブロックデバイスマッピングを指定します。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
    "KeyName": { "Ref": "KeyName" },
    "InstanceType": { "Ref": "InstanceType" },
    "SecurityGroups": [{ "Ref": "Ec2SecurityGroup" }],
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/sda1",
        "Ebs": { "VolumeSize": "50" }
      },
      {
        "DeviceName": "/dev/sdm",
        "Ebs": { "VolumeSize": "100" }
      }
    ]
  }
}
```

YAML

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}'
```

```

KeyName: !Ref KeyName
InstanceType: !Ref InstanceType
SecurityGroups:
  - !Ref Ec2SecurityGroup
BlockDeviceMappings:
  -
    DeviceName: /dev/sda1
    Ebs:
      VolumeSize: 50
  -
    DeviceName: /dev/sdm
    Ebs:
      VolumeSize: 100

```

インスタンスストアボリュームのブロックデバイスマッピングの指定

JSON

```

"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
    "KeyName" : { "Ref" : "KeyName" },
    "InstanceType": { "Ref": "InstanceType" },
    "SecurityGroups" : [{ "Ref" : "Ec2SecurityGroup" }],
    "BlockDeviceMappings" : [
      {
        "DeviceName" : "/dev/sdc",
        "VirtualName" : "ephemeral0"
      }
    ]
  }
}

```

YAML

```

EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}'
    KeyName: !Ref KeyName

```

```
InstanceType: !Ref InstanceType
SecurityGroups:
  - !Ref Ec2SecurityGroup
BlockDeviceMappings:
  - DeviceName: /dev/sdc
    VirtualName: ephemeral0
```

CloudFormation を使用して起動テンプレートを作成する

このセクションでは、CloudFormation を使用して Amazon EC2 起動テンプレートを作成する方法を例を挙げて説明します。起動テンプレートを使用すると、AWS 内で Amazon EC2 インスタンスを設定してプロビジョニングするためのテンプレートを作成することができます。起動テンプレートを使用すると起動パラメータを保存することができるため、インスタンスを起動するたびに起動パラメータ指定する必要がなくなります。その他の例については、AWS::EC2::LaunchTemplate リソースの「[Examples](#)」のセクションを参照してください。

起動テンプレートの詳細については、『Amazon EC2 ユーザーガイド』の「[Amazon EC2 起動テンプレートにインスタンス起動パラメータを保存する](#)」を参照してください。

Auto Scaling グループで使用する起動テンプレートの作成方法については、『Amazon EC2 Auto Scaling ユーザーガイド』の「[Auto Scaling 起動テンプレート](#)」を参照してください。

スニペットカテゴリ

- [セキュリティグループ、タグ、ユーザーデータ、IAM ロールを指定する起動テンプレートを作成する](#)

セキュリティグループ、タグ、ユーザーデータ、IAM ロールを指定する起動テンプレートを作成する

このスニペットには、インスタンスを起動する設定情報が含まれる [AWS::EC2::LaunchTemplate](#) リソースが記されています。ImageId、InstanceType、SecurityGroups、UserData、および TagSpecifications プロパティの値を指定します。SecurityGroups プロパティは、既存の EC2 セキュリティグループと新しいセキュリティグループを指定します。Ref 関数は、スタックテンプレートの他の場所で宣言されている [AWS::EC2::SecurityGroup](#) リソース myNewEC2SecurityGroup の ID を取得します。

起動テンプレートにはカスタムユーザーデータのセクションが含まれています。このセクションでは、インスタンスの起動時に実行される構成タスクとスクリプトを渡すことができます。この例で

は、ユーザーデータは AWS Systems Manager Agent をインストールし、エージェントを起動します。

起動テンプレートには、インスタンスで実行中のアプリケーションが、ユーザーの代わりにアクションを実行できるようにする IAM ロールも含まれています。この例では、起動テンプレートの [AWS::IAM::Role](#) リソースを示しており、`IamInstanceProfile` プロパティを使用して IAM ロールを指定します。Ref 関数は、[AWS::IAM::InstanceProfile](#) リソース `myInstanceProfile` の名前を取得します。IAM ロールのアクセス許可を設定するには、`ManagedPolicyArns` プロパティの値を指定します。

JSON

```
{
  "Resources":{
    "myLaunchTemplate":{
      "Type":"AWS::EC2::LaunchTemplate",
      "Properties":{
        "LaunchTemplateName":{" Fn::Sub": "${AWS::StackName}-launch-template" },
        "LaunchTemplateData":{
          "ImageId":"ami-02354e95b3example",
          "InstanceType":"t3.micro",
          "IamInstanceProfile":{
            "Name":{
              "Ref":"myInstanceProfile"
            }
          },
        },
        "SecurityGroupIds":[
          {
            "Ref":"myNewEC2SecurityGroup"
          },
          "sg-083cd3bfb8example"
        ],
        "UserData":{
          "Fn::Base64":{
            "Fn::Join": [
              "", [
                "#!/bin/bash\n",
                "cd /tmp\n",
                "yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm\n",
                "systemctl enable amazon-ssm-agent\n",
                "systemctl start amazon-ssm-agent\n"
              ]
            ]
          }
        }
      }
    }
  }
```



```
    ]
  },
  "TagSpecifications":[
    {
      "ResourceType":"instance",
      "Tags":[
        {
          "Key":"environment",
          "Value":"development"
        }
      ]
    },
    {
      "ResourceType":"volume",
      "Tags":[
        {
          "Key":"environment",
          "Value":"development"
        }
      ]
    }
  ]
},
"myInstanceRole":{
  "Type":"AWS::IAM::Role",
  "Properties":{
    "RoleName":"InstanceRole",
    "AssumeRolePolicyDocument":{
      "Version": "2012-10-17",
      "Statement":[
        {
          "Effect":"Allow",
          "Principal":{
            "Service":[
              "ec2.amazonaws.com"
            ]
          },
          "Action":[
            "sts:AssumeRole"
          ]
        }
      ]
    }
  }
}
```

```
]
},
"ManagedPolicyArns":[
    "arn:aws:iam::aws:policy/myCustomerManagedPolicy"
]
}
},
"myInstanceProfile":{
    "Type":"AWS::IAM::InstanceProfile",
    "Properties":{
        "Path":"/",
        "Roles":[
            {
                "Ref":"myInstanceRole"
            }
        ]
    }
}
}
}
```

YAML

```

---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        IamInstanceProfile:
          Name: !Ref myInstanceProfile
        SecurityGroupIds:
          - !Ref myNewEC2SecurityGroup
          - sg-083cd3bfb8example
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            cd /tmp
            yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/
latest/linux_amd64/amazon-ssm-agent.rpm

```

```

        systemctl enable amazon-ssm-agent
        systemctl start amazon-ssm-agent
    TagSpecifications:
    - ResourceType: instance
      Tags:
      - Key: environment
        Value: development
    - ResourceType: volume
      Tags:
      - Key: environment
        Value: development
  myInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: InstanceRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: 'Allow'
          Principal:
            Service:
            - 'ec2.amazonaws.com'
          Action:
          - 'sts:AssumeRole'
      ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/myCustomerManagedPolicy'
  myInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: '/'
      Roles:
      - !Ref myInstanceRole

```

CloudFormation でセキュリティグループを管理する

以下のスニペットは、CloudFormation を使用してセキュリティグループと Amazon EC2 インスタンスを管理し、AWS リソースへのアクセスを制御する方法を示しています。

スニペットカテゴリ

- [Amazon EC2 インスタンスをセキュリティグループと関連付けるには](#)
- [進入ルールを持つセキュリティグループの作成](#)
- [セキュリティグループの進入ルールを使用して Elastic Load Balancer を作成する](#)

Amazon EC2 インスタンスをセキュリティグループと関連付けるには

次のスニペットは、CloudFormation を使用して Amazon EC2 インスタンスをデフォルトの Amazon VPC セキュリティグループに関連付ける方法を示します。

例

- [Amazon EC2 インスタンスをデフォルトの VPC セキュリティグループと関連付けるには](#)
- [ボリュームとセキュリティグループをアタッチした Amazon EC2 インスタンスを作成する](#)

Amazon EC2 インスタンスをデフォルトの VPC セキュリティグループと関連付けるには

次のスニペットは、Amazon VPC、VPC 内のサブネット、Amazon EC2 インスタンスを作成します。VPC は [AWS::EC2::VPC](#) リソースを使用して作成されます。VPC の IP アドレス範囲は、大きい方のテンプレートで定義され、MyVPCCIDRRange パラメータによって参照されます。

サブネットは [AWS::EC2::Subnet](#) リソースを使用します。サブネットは、MyVPC として参照される VPC に関連付けられています。

EC2 インスタンスは、[AWS::EC2::Instance](#) リソースを使用して VPC とサブネット内で起動されます。このリソースは、インスタンスの起動に使用する Amazon マシンイメージ (AMI)、インスタンスが実行されるサブネット、インスタンスに関連付けるセキュリティグループを指定します。ImageId は Systems Manager パラメータを使用して、最新の Amazon Linux 2 AMI を動的に取得します。

セキュリティグループ ID は、MyVPC リソースからデフォルトのセキュリティグループを取得する Fn::GetAtt 関数を使用して取得されます。

インスタンスはスニペットで定義された MySubnet リソース内に配置されます。

CloudFormation を使用して VPC を作成すると、AWS はデフォルトのセキュリティグループを含むデフォルトリソースを VPC 内に自動的に作成します。ただし、CloudFormation テンプレート内で VPC を定義すると、テンプレートを作成するときにこれらのデフォルトリソースの ID にアクセスできない場合があります。テンプレートで指定されているデフォルトリソースにアクセスして使用するには、Fn::GetAtt などの組み込み関数を使用できます。この関数を使用すると、CloudFormation によって自動的に作成されたデフォルトリソースを使用できます。

JSON

```
"MyVPC": {
```

```

    "Type": "AWS::EC2::VPC",
    "Properties": {
      "CidrBlock": {
        "Ref": "MyVPCCIDRRange"
      },
      "EnableDnsSupport": false,
      "EnableDnsHostnames": false,
      "InstanceTenancy": "default"
    }
  },
  "MySubnet": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "CidrBlock": {
        "Ref": "MyVPCCIDRRange"
      },
      "VpcId": {
        "Ref": "MyVPC"
      }
    }
  },
  "MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
      "SecurityGroupIds": [
        {
          "Fn::GetAtt": [
            "MyVPC",
            "DefaultSecurityGroup"
          ]
        }
      ],
      "SubnetId": {
        "Ref": "MySubnet"
      }
    }
  }
}

```

YAML

MyVPC:

```
Type: AWS::EC2::VPC
Properties:
  CidrBlock:
    Ref: MyVPCCIDRRange
  EnableDnsSupport: false
  EnableDnsHostnames: false
  InstanceTenancy: default
MySubnet:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock:
      Ref: MyVPCCIDRRange
    VpcId:
      Ref: MyVPC
MyInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}'
    SecurityGroupIds:
      - Fn::GetAtt:
          - MyVPC
          - DefaultSecurityGroup
    SubnetId:
      Ref: MySubnet
```

ボリュームとセキュリティグループをアタッチした Amazon EC2 インスタンスを作成する

次のスニペットは、指定した AMI から起動する [AWS::EC2::Instance](#) リソースを使用する Amazon EC2 インスタンスを作成します。インスタンスは、[AWS::EC2::SecurityGroup](#) リソースを使用して、指定された IP アドレスからポート 22 で受信する SSH トラフィックを許可するセキュリティグループに関連付けられています。[AWS::EC2::Volume](#) リソースを使用して 100 GB の Amazon EBS ボリュームを作成します。ボリュームは、GetAtt 関数で指定されたインスタンスと同じアベイラビリティゾーンに作成され、/dev/sdh デバイスのインスタンスにマウントされます。

Amazon EBS ボリュームの作成の詳細については、「[Amazon EBS ボリュームの作成](#)」を参照してください。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
```

```
    "SecurityGroups": [
      {
        "Ref": "InstanceSecurityGroup"
      }
    ],
    "ImageId": "ami-1234567890abcdef0"
  }
},
"InstanceSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "Enable SSH access via port 22",
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": "22",
        "ToPort": "22",
        "CidrIp": "192.0.2.0/24"
      }
    ]
  }
},
"NewVolume": {
  "Type": "AWS::EC2::Volume",
  "Properties": {
    "Size": "100",
    "AvailabilityZone": {
      "Fn::GetAtt": [
        "Ec2Instance",
        "AvailabilityZone"
      ]
    }
  }
},
"MountPoint": {
  "Type": "AWS::EC2::VolumeAttachment",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    },
    "VolumeId": {
      "Ref": "NewVolume"
    },
    "Device": "/dev/sdh"
  }
}
```

```
}  
}
```

YAML

```
Ec2Instance:  
  Type: AWS::EC2::Instance  
  Properties:  
    SecurityGroups:  
      - !Ref InstanceSecurityGroup  
    ImageId: ami-1234567890abcdef0  
InstanceSecurityGroup:  
  Type: AWS::EC2::SecurityGroup  
  Properties:  
    GroupDescription: Enable SSH access via port 22  
    SecurityGroupIngress:  
      - IpProtocol: tcp  
        FromPort: 22  
        ToPort: 22  
        CidrIp: 192.0.2.0/24  
NewVolume:  
  Type: AWS::EC2::Volume  
  Properties:  
    Size: 100  
    AvailabilityZone: !GetAtt [Ec2Instance, AvailabilityZone]  
MountPoint:  
  Type: AWS::EC2::VolumeAttachment  
  Properties:  
    InstanceId: !Ref Ec2Instance  
    VolumeId: !Ref NewVolume  
    Device: /dev/sdh
```

進入ルールを持つセキュリティグループの作成

以下のサンプルスニペットは、CloudFormation を使用して特定の進入ルールでセキュリティグループを設定する方法を示しています。

スニペット

- [SSH と HTTP アクセスの進入ルールを使用してセキュリティグループを作成します。](#)
- [指定された CIDR 範囲からの HTTP と SSH アクセスの進入ルールを使用してセキュリティグループを作成します。](#)

- [進入ルールを使用して相互参照セキュリティグループを作成する](#)

SSH と HTTP アクセスの進入ルールを使用してセキュリティグループを作成します。

このスニペットは、2 つのセキュリティグループ進入ルールが記述された [AWS::EC2::SecurityGroup](#) リソースを示します。最初の進入ルールは、アカウント番号 1111-2222-3333 を持つ AWS アカウントが所有する、MyAdminSecurityGroup という名前の既存のセキュリティグループからの SSH (ポート 22) アクセスを許可します。2 番目の進入ルールは、同じテンプレートで作成された、MySecurityGroupCreatedInCFN という名前の別のセキュリティグループからの HTTP (ポート 80) アクセスを許可します。Ref 関数は、同じテンプレートで作成されたセキュリティグループの論理名を参照するために使用されます。

最初の進入ルールでは、SourceSecurityGroupName と SourceSecurityGroupOwnerId プロパティの両方に値を追加する必要があります。2 番目の進入ルールでは、MySecurityGroupCreatedInCFNTemplate は同じテンプレートで作成された別のセキュリティグループを参照します。論理名が、大きい方のテンプレートで指定するセキュリティグループリソースの実際の論理名 MySecurityGroupCreatedInCFNTemplate と一致することを確認してください。

セキュリティグループの詳細については、「[Amazon EC2 ユーザーガイド](#)」の「Amazon EC2 インスタンス用の Amazon EC2 セキュリティグループ」を参照してください。

JSON

```
"SecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "Allow connections from specified source security group",
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": "22",
        "ToPort": "22",
        "SourceSecurityGroupName": "MyAdminSecurityGroup",
        "SourceSecurityGroupOwnerId": "1111-2222-3333"
      },
      {
        "IpProtocol": "tcp",
        "FromPort": "80",
        "ToPort": "80",
        "SourceSecurityGroupName": {
```

```

        "Ref": "MySecurityGroupCreatedInCFNTemplate"
      }
    }
  ]
}

```

YAML

```

SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified source security group
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: '22'
        ToPort: '22'
        SourceSecurityGroupName: MyAdminSecurityGroup
        SourceSecurityGroupOwnerId: '1111-2222-3333'
      - IpProtocol: tcp
        FromPort: '80'
        ToPort: '80'
        SourceSecurityGroupName:
          Ref: MySecurityGroupCreatedInCFNTemplate

```

指定された CIDR 範囲からの HTTP と SSH アクセスの進入ルールを使用してセキュリティグループを作成します。

次のスニペットは、2 つのインバウンドルールを持つ Amazon EC2 インスタンスのセキュリティグループを作成します。インバウンドルールは、指定された CIDR 範囲からの指定ポートへの受信 TCP トラフィックを許可します。[AWS::EC2::SecurityGroup](#) リソースは、ルールの指定に使用されます。各ルールのプロトコルを指定する必要があります。TCP の場合は、ポートまたはポート範囲も指定する必要があります。ソースセキュリティグループまたは CIDR 範囲のいずれも指定しない場合、スタックは正常に起動しますが、ルールはセキュリティグループに適用されません。

セキュリティグループの詳細については、「[Amazon EC2 ユーザーガイド](#)」の「Amazon EC2 インスタンス用の Amazon EC2 セキュリティグループ」を参照してください。

JSON

```

"ServerSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",

```

```

"Properties": {
  "GroupDescription": "Allow connections from specified CIDR ranges",
  "SecurityGroupIngress": [
    {
      "IpProtocol": "tcp",
      "FromPort": "80",
      "ToPort": "80",
      "CidrIp": "192.0.2.0/24"
    },
    {
      "IpProtocol": "tcp",
      "FromPort": "22",
      "ToPort": "22",
      "CidrIp": "192.0.2.0/24"
    }
  ]
}
}

```

YAML

```

ServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified CIDR ranges
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 192.0.2.0/24
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 192.0.2.0/24

```

進入ルールを使用して相互参照セキュリティグループを作成する

次のスニペットは、[AWS::EC2::SecurityGroup](#) リソースを使用して、SGroup1 と SGroup2 の 2 つの Amazon EC2 セキュリティグループを作成します。2 つのセキュリティグループ間の通信を許可する進入ルールは、[AWS::EC2::SecurityGroupIngress](#) リソースを使用して作成されます。SGroup1Ingress は、送信元セキュリティグループ SGroup2 からポート 80 で受信 TCP トラフィックを許可する SGroup1 の進入ルールを確立します。SGroup2Ingress は、送信元セキュリ

ティグループ SGroup1 からのポート 80 への着信 TCP トラフィックを許可する SGroup2 の進入ルールを確立します。

JSON

```
"SGroup1": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "EC2 instance access"
  }
},
"SGroup2": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "EC2 instance access"
  }
},
"SGroup1Ingress": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties": {
    "GroupName": {
      "Ref": "SGroup1"
    },
    "IpProtocol": "tcp",
    "ToPort": "80",
    "FromPort": "80",
    "SourceSecurityGroupName": {
      "Ref": "SGroup2"
    }
  }
},
"SGroup2Ingress": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties": {
    "GroupName": {
      "Ref": "SGroup2"
    },
    "IpProtocol": "tcp",
    "ToPort": "80",
    "FromPort": "80",
    "SourceSecurityGroupName": {
      "Ref": "SGroup1"
    }
  }
}
```

```
}
```

YAML

```
SGroup1:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup2:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup1Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupName: !Ref SGroup1
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup2
SGroup2Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupName: !Ref SGroup2
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup1
```

セキュリティグループの進入ルールを使用して Elastic Load Balancer を作成する

次のテンプレートは、指定されたアベイラビリティゾーンに

[AWS::ElasticLoadBalancing::LoadBalancer](#) リソースを作成しま

す。[AWS::ElasticLoadBalancing::LoadBalancer](#) リソースは、ポート 80 で HTTP トラフィックを受信し、同じくポート 80 のインスタンスにリクエストを転送するように設定されています。Elastic Load Balancer は、インスタンス間で受信 HTTP トラフィックの負荷を分散します。

さらに、このテンプレートは、ロードバランサーに関連付けられた [AWS::EC2::SecurityGroup](#) リソースを生成します。このセキュリティグループは、ポート 80 での着信 TCP トラフィックを許可する、ELB ingress group という 1 つの進入ルールを使用して作成されます。この進入ルールのソースは、ロードバランサーリソースから属性を取得する Fn::GetAtt 関数を使用して定義され

まず、SourceSecurityGroupOwnerId は Fn::GetAtt を使用してロードバランサーのソースセキュリティグループの OwnerAlias を取得します。SourceSecurityGroupName は Fn::GetAtt を使用して ELB のソースセキュリティグループの GroupName を取得します。

この設定により、ELB とインスタンス間の安全な通信が保証されます。

ロードバランシングの詳細については、「[Elastic Load Balancing ユーザーガイド](#)」を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "MyELB": {
      "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
      "Properties": {
        "AvailabilityZones": [
          "aa-example-1a"
        ],
        "Listeners": [
          {
            "LoadBalancerPort": "80",
            "InstancePort": "80",
            "Protocol": "HTTP"
          }
        ]
      }
    },
    "MyELBIngressGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "ELB ingress group",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "SourceSecurityGroupOwnerId": {
              "Fn::GetAtt": [
                "MyELB",
                "SourceSecurityGroup.OwnerAlias"
              ]
            }
          }
        ]
      }
    }
  }
}
```


Elastic IP アドレスを CloudFormation と割り当ておよび関連付けする

以下のテンプレートスニペットは、Amazon EC2 の Elastic IP アドレス (EIP) に関連する例です。これらの例ではインスタンスの EIP の割り当て、関連付け、管理について説明します。

例

- [Elastic IP アドレスを割り当て Amazon EC2 インスタンスに関連付ける](#)
- [IP アドレスを指定して、Elastic IP アドレスを Amazon EC2 インスタンスに関連付ける](#)
- [IP アドレスの割り当て ID を指定して、Elastic IP アドレスを Amazon EC2 インスタンスに関連付ける](#)

Elastic IP アドレスを割り当て Amazon EC2 インスタンスに関連付ける

次のスニペットでは、[AWS::EC2::EIP](#) リソースを使用して Amazon EC2 Elastic IP (EIP) アドレスを Amazon EC2 インスタンスに関連付けます。EIP アドレスは、AWS が所有するアドレスプールから、または [Bring Your Own IP \(BYOIP\)](#) により AWS リソースで使用するために AWS に持ち込んだパブリック IPv4 アドレス範囲から作成されたアドレスプールから、割り当てることができます。この例では、EIP は AWS が所有するアドレスプールから割り当てられます。

Elastic IP アドレスの詳細については、Amazon EC2 ユーザーガイドの「[Elastic IP アドレス](#)」を参照してください。

JSON

```
"ElasticIP": {
  "Type": "AWS::EC2::EIP",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    }
  }
}
```

YAML

```
ElasticIP:
  Type: AWS::EC2::EIP
  Properties:
    InstanceId: !Ref EC2Instance
```


IP アドレスを指定して、Elastic IP アドレスを Amazon EC2 インスタンスに関連付ける

次のスニペットでは、[AWS::EC2::EIPAssociation](#) リソースを使用して既存の Amazon EC2 Elastic IP アドレスを EC2 インスタンスに関連付けます。アカウントで使用するためには、まず Elastic IP アドレスを割り当てます。Elastic IP アドレスは、1 つのインスタンスに関連付けられます。

JSON

```
"IPAssoc": {
  "Type": "AWS::EC2::EIPAssociation",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    },
    "EIP": "192.0.2.0"
  }
}
```

YAML

```
IPAssoc:
  Type: AWS::EC2::EIPAssociation
  Properties:
    InstanceId: !Ref EC2Instance
    EIP: 192.0.2.0
```

IP アドレスの割り当て ID を指定して、Elastic IP アドレスを Amazon EC2 インスタンスに関連付ける

次のスニペットでは、[AWS::EC2::EIPAssociation](#) リソースを使用して割り当て ID を使用し、既存の Elastic IP アドレスを Amazon EC2 インスタンスに関連付けます。Elastic IP アドレスが割り当てられると、割り当て ID が Elastic IP アドレスに付与されます。

JSON

```
"IPAssoc": {
  "Type": "AWS::EC2::EIPAssociation",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    },
    "AllocationId": "eipalloc-1234567890abcdef0"
  }
}
```

```
}  
}
```

YAML

```
IPAssoc:  
  Type: AWS::EC2::EIPAssociation  
  Properties:  
    InstanceId: !Ref EC2Instance  
    AllocationId: eipalloc-1234567890abcdef0
```

CloudFormation を使用して Amazon VPC リソースを設定する

このセクションでは、CloudFormation を使用して Amazon VPC リソースを設定するための例について説明します。VPC を使用すると、AWS 内に仮想ネットワークを作成できます。以下のスニペットでは、ネットワーク要件に合わせて VPC のさまざまな要素を設定する方法を示しています。

例

- [VPC で IPv6 Egress-Only インターネットアクセスを有効にする](#)
- [Elastic Network Interface \(ENI\) のテンプレートスニペット](#)

VPC で IPv6 Egress-Only インターネットアクセスを有効にする

Egress-Only インターネットゲートウェイを使用すると、VPC 内のインスタンスがインターネットにアクセスできるようになり、インターネット上のリソースがインスタンスと通信できなくなります。次のスニペットは、VPC 内からの IPv6 出力専用インターネットアクセスを有効にします。[AWS::EC2::VPC](#) リソースを使用する IPv4 アドレス範囲 10.0.0/16 で VPC を作成します。ルートテーブルは [AWS::EC2::RouteTable](#) リソースを使用してこの VPC リソースに関連付けられます。ルートテーブルは VPC 内のインスタンスのルート进行管理します。[AWS::EC2::EgressOnlyInternetGateway](#) は、Egress-Only インターネットゲートウェイを作成するために使用されます。これにより、VPC 内のインスタンスからのアウトバウンドトラフィックの IPv6 通信が可能になり、インバウンドトラフィックは防止されます。[AWS::EC2::Route](#) リソースを指定して、すべてのアウトバウンド IPv6 トラフィック (:::/0) を Egress-Only インターネットゲートウェイに転送する IPv6 ルートをルートテーブルに作成します。

Egress-Only インターネットゲートウェイの詳細については、『Amazon VPC ユーザーガイド』の「[エグレス専用インターネットゲートウェイを使用してアウトバウンド IPv6 トラフィックを有効にする](#)」を参照してください。

JSON

```
"DefaultIpv6Route": {
  "Type": "AWS::EC2::Route",
  "Properties": {
    "DestinationIpv6CidrBlock": "::/0",
    "EgressOnlyInternetGatewayId": {
      "Ref": "EgressOnlyInternetGateway"
    },
    "RouteTableId": {
      "Ref": "RouteTable"
    }
  }
},
"EgressOnlyInternetGateway": {
  "Type": "AWS::EC2::EgressOnlyInternetGateway",
  "Properties": {
    "VpcId": {
      "Ref": "VPC"
    }
  }
},
"RouteTable": {
  "Type": "AWS::EC2::RouteTable",
  "Properties": {
    "VpcId": {
      "Ref": "VPC"
    }
  }
},
"VPC": {
  "Type": "AWS::EC2::VPC",
  "Properties": {
    "CidrBlock": "10.0.0.0/16"
  }
}
```

YAML

```
DefaultIpv6Route:
  Type: AWS::EC2::Route
  Properties:
    DestinationIpv6CidrBlock: "::/0"
```

```
EgressOnlyInternetGatewayId:
  Ref: "EgressOnlyInternetGateway"
RouteTableId:
  Ref: "RouteTable"
EgressOnlyInternetGateway:
  Type: AWS::EC2::EgressOnlyInternetGateway
  Properties:
    VpcId:
      Ref: "VPC"
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: "VPC"
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: "10.0.0.0/16"
```

Elastic Network Interface (ENI) のテンプレートスニペット

Elastic Network Interface (ENI) がアタッチされた Amazon EC2 インスタンスを作成する

次のスニペット例では、[AWS::EC2::Instance](#) リソースを使用して Amazon EC2 インスタンスを指定された Amazon VPC とサブネットに作成します。2 つの Elastic Network Interface (ENI) をインスタンスにアタッチし、アタッチされた ENI を通じて Elastic IP アドレスをインスタンスに関連付け、SSH と HTTP アクセス用のセキュリティグループを設定します。ユーザーデータは、インスタンスの作成時に起動設定の一部としてインスタンスに提供されます。ユーザーデータには、確実にインスタンスに渡されるように、base64 フォーマットにエンコードされたスクリプトが含まれています。インスタンスが起動すると、ブートストラッププロセスの一環としてスクリプトが自動的に実行されます。ec2-net-utils をインストールし、ネットワークインターフェイスを設定し、HTTP サービスを開始します。

選択したリージョンに基づいて適切な Amazon マシンイメージ (AMI) を決定するために、スニペットは RegionMap マッピング内の値を検索する Fn::FindInMap 関数を使用します。このマッピングは大きい方のテンプレートで定義する必要があります。2 つのネットワークインターフェイスは [AWS::EC2::NetworkInterface](#) リソースを使用して作成されます。Elastic IP アドレスは、vpc ドメインに割り当てられた [AWS::EC2::EIP](#) リソースを使用して指定されます。これらの Elastic IP アドレスは、[AWS::EC2::EIPAssociation](#) リソースを使用してネットワークインターフェイスに関連付けられます。

Outputs セクションでは、スタックが作成された後にアクセスする値またはリソースを定義します。次のスニペットで定義されている出力は InstancePublicIp で、スタックによって作成された EC2 インスタンスのパブリック IP アドレスを表すものです。この出力は、CloudFormation コンソールの [出力] タブまたは [describe-stacks](#) コマンドを使用して取得できます。

Elastic Network Interface の詳細については、「[Elastic Network Interface](#)」を参照してください。

JSON

```
"Resources": {
  "ControlPortAddress": {
    "Type": "AWS::EC2::EIP",
    "Properties": {
      "Domain": "vpc"
    }
  },
  "AssociateControlPort": {
    "Type": "AWS::EC2::EIPAssociation",
    "Properties": {
      "AllocationId": {
        "Fn::GetAtt": [
          "ControlPortAddress",
          "AllocationId"
        ]
      },
      "NetworkInterfaceId": {
        "Ref": "controlXface"
      }
    }
  },
  "WebPortAddress": {
    "Type": "AWS::EC2::EIP",
    "Properties": {
      "Domain": "vpc"
    }
  },
  "AssociateWebPort": {
    "Type": "AWS::EC2::EIPAssociation",
    "Properties": {
      "AllocationId": {
        "Fn::GetAtt": [
          "WebPortAddress",
          "AllocationId"
        ]
      }
    }
  }
}
```

```
        ],
        },
        "NetworkInterfaceId": {
            "Ref": "webXface"
        }
    },
    "SSHSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
            "GroupDescription": "Enable SSH access via port 22",
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 22,
                    "IpProtocol": "tcp",
                    "ToPort": 22
                }
            ]
        }
    },
    "WebSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
            "GroupDescription": "Enable HTTP access via user-defined port",
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 80,
                    "IpProtocol": "tcp",
                    "ToPort": 80
                }
            ]
        }
    },
    "controlXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
```

```
        "SubnetId": {
            "Ref": "SubnetId"
        },
        "Description": "Interface for controlling traffic such as SSH",
        "GroupSet": [
            {
                "Fn::GetAtt": [
                    "SSHSecurityGroup",
                    "GroupId"
                ]
            }
        ],
        "SourceDestCheck": true,
        "Tags": [
            {
                "Key": "Network",
                "Value": "Control"
            }
        ]
    },
    "webXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
            "SubnetId": {
                "Ref": "SubnetId"
            },
            "Description": "Interface for web traffic",
            "GroupSet": [
                {
                    "Fn::GetAtt": [
                        "WebSecurityGroup",
                        "GroupId"
                    ]
                }
            ],
            "SourceDestCheck": true,
            "Tags": [
                {
                    "Key": "Network",
                    "Value": "Web"
                }
            ]
        }
    }
}
```

```

    },
    "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": {
          "Fn::FindInMap": [
            "RegionMap",
            {
              "Ref": "AWS::Region"
            }
          ],
          "AMI"
        },
        "KeyName": {
          "Ref": "KeyName"
        },
        "NetworkInterfaces": [
          {
            "NetworkInterfaceId": {
              "Ref": "controlXface"
            },
            "DeviceIndex": "0"
          },
          {
            "NetworkInterfaceId": {
              "Ref": "webXface"
            },
            "DeviceIndex": "1"
          }
        ],
        "Tags": [
          {
            "Key": "Role",
            "Value": "Test Instance"
          }
        ],
        "UserData": {
          "Fn::Base64": {
            "Fn::Sub": "#!/bin/bash -xe\nyum install ec2-net-utils -y\nec2ifup  
eth1\nservice httpd start\n"
          }
        }
      }
    }
  }
}

```



```
},
"Outputs": {
  "InstancePublicIp": {
    "Description": "Public IP Address of the EC2 Instance",
    "Value": {
      "Fn::GetAtt": [
        "Ec2Instance",
        "PublicIp"
      ]
    }
  }
}
```

YAML

```
Resources:
  ControlPortAddress:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateControlPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - ControlPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: controlXface
  WebPortAddress:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateWebPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - WebPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: webXface
  SSHSecurityGroup:
```

```
Type: AWS::EC2::SecurityGroup
Properties:
  VpcId:
    Ref: VpcId
  GroupDescription: Enable SSH access via port 22
  SecurityGroupIngress:
    - CidrIp: 0.0.0.0/0
      FromPort: 22
      IpProtocol: tcp
      ToPort: 22
WebSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    VpcId:
      Ref: VpcId
    GroupDescription: Enable HTTP access via user-defined port
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        FromPort: 80
        IpProtocol: tcp
        ToPort: 80
controlXface:
  Type: AWS::EC2::NetworkInterface
  Properties:
    SubnetId:
      Ref: SubnetId
    Description: Interface for controlling traffic such as SSH
    GroupSet:
      - Fn::GetAtt:
          - SSHSecurityGroup
          - GroupId
    SourceDestCheck: true
    Tags:
      - Key: Network
        Value: Control
webXface:
  Type: AWS::EC2::NetworkInterface
  Properties:
    SubnetId:
      Ref: SubnetId
    Description: Interface for web traffic
    GroupSet:
      - Fn::GetAtt:
          - WebSecurityGroup
```

```
    - GroupId
SourceDestCheck: true
Tags:
  - Key: Network
    Value: Web
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId:
      Fn::FindInMap:
        - RegionMap
        - Ref: AWS::Region
        - AMI
    KeyName:
      Ref: KeyName
    NetworkInterfaces:
      - NetworkInterfaceId:
          Ref: controlXface
          DeviceIndex: "0"
      - NetworkInterfaceId:
          Ref: webXface
          DeviceIndex: "1"
    Tags:
      - Key: Role
        Value: Test Instance
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash -xe
        yum install ec2-net-utils -y
        ec2ifup eth1
        service httpd start
Outputs:
  InstancePublicIp:
    Description: Public IP Address of the EC2 Instance
    Value:
      Fn::GetAtt:
        - Ec2Instance
        - PublicIp
```

Amazon Elastic Container Service サンプルテンプレート

Amazon Elastic Container Service (Amazon ECS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターで Docker コンテナを簡単に実行、停止、管理できるようにするコンテナ管理サービスです。

AL2023 Amazon ECS-Optimized-AMI を使用してクラスターを作成する

キャパシティープロバイダーを使用して Amazon EC2 で AL2023 インスタンスを起動するクラスターを定義します。

Important

最新の AMI ID については、「Amazon Elastic Container Service デベロッパーガイド」の「[Amazon ECS-optimized AMI](#)」(Amazon ECS に最適化された AMI) を参照してください。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances yet. An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.",
  "Parameters": {
    "InstanceType": {
      "Type": "String",
      "Description": "EC2 instance type",
      "Default": "t2.medium",
      "AllowedValues": [
        "t1.micro",
        "t2.2xlarge",
        "t2.large",
        "t2.medium",
        "t2.micro",
        "t2.nano",
        "t2.small",
        "t2.xlarge",
        "t3.2xlarge",
        "t3.large",
        "t3.medium",
        "t3.micro",

```

```

        "t3.nano",
        "t3.small",
        "t3.xlarge"
    ]
},
"DesiredCapacity": {
    "Type": "Number",
    "Default": "0",
    "Description": "Number of EC2 instances to launch in your ECS cluster."
},
"MaxSize": {
    "Type": "Number",
    "Default": "100",
    "Description": "Maximum number of EC2 instances that can be launched in your
ECS cluster."
},
"ECSAMI": {
    "Description": "The Amazon Machine Image ID used for the cluster",
    "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
    "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/
image_id"
},
"VpcId": {
    "Type": "AWS::EC2::VPC::Id",
    "Description": "VPC ID where the ECS cluster is launched",
    "Default": "vpc-1234567890abcdef0"
},
"SubnetIds": {
    "Type": "List<AWS::EC2::Subnet::Id>",
    "Description": "List of subnet IDs where the EC2 instances will be launched",
    "Default": "subnet-021345abcdef67890"
}
},
"Resources": {
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterSettings": [
                {
                    "Name": "containerInsights",
                    "Value": "enabled"
                }
            ]
        }
    }
}

```

```
    },
    "ECSAutoScalingGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "DependsOn": [
        "ECSCluster",
        "EC2Role"
      ],
      "Properties": {
        "VPCZoneIdentifier": {
          "Ref": "SubnetIds"
        },
        "LaunchTemplate": {
          "LaunchTemplateId": {
            "Ref": "ContainerInstances"
          },
          "Version": {
            "Fn::GetAtt": [
              "ContainerInstances",
              "LatestVersionNumber"
            ]
          }
        },
        "MinSize": 0,
        "MaxSize": {
          "Ref": "MaxSize"
        },
        "DesiredCapacity": {
          "Ref": "DesiredCapacity"
        },
        "NewInstancesProtectedFromScaleIn": true
      },
      "UpdatePolicy": {
        "AutoScalingReplacingUpdate": {
          "WillReplace": "true"
        }
      }
    },
    "ContainerInstances": {
      "Type": "AWS::EC2::LaunchTemplate",
      "Properties": {
        "LaunchTemplateName": "asg-launch-template",
        "LaunchTemplateData": {
          "ImageId": {
            "Ref": "ECSAMI"
          }
        }
      }
    }
  }
}
```

```

    },
    "InstanceType": {
      "Ref": "InstanceType"
    },
    "IamInstanceProfile": {
      "Name": {
        "Ref": "EC2InstanceProfile"
      }
    },
    "SecurityGroupIds": [
      {
        "Ref": "ContainerHostSecurityGroup"
      }
    ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Sub": "
#!/bin/bash -xe\n
echo ECS_CLUSTER=${ECSCluster}

>> /etc/ecs/ecs.config\n
yum install -y aws-cfn-bootstrap\n
/opt/aws/bin/cfn-init -
v --stack ${AWS::StackId} --resource ContainerInstances --configsets full_install --
region ${AWS::Region} &\n"
      }
    },
    "MetadataOptions": {
      "HttpEndpoint": "enabled",
      "HttpTokens": "required"
    }
  }
},
"EC2InstanceProfile": {
  "Type": "AWS::IAM::InstanceProfile",
  "Properties": {
    "Path": "/",
    "Roles": [
      {
        "Ref": "EC2Role"
      }
    ]
  }
},
"CapacityProvider": {
  "Type": "AWS::ECS::CapacityProvider",
  "Properties": {
    "AutoScalingGroupProvider": {

```

```
        "AutoScalingGroupArn": {
            "Ref": "ECSAutoScalingGroup"
        },
        "ManagedScaling": {
            "InstanceWarmupPeriod": 60,
            "MinimumScalingStepSize": 1,
            "MaximumScalingStepSize": 100,
            "Status": "ENABLED",
            "TargetCapacity": 100
        },
        "ManagedTerminationProtection": "ENABLED"
    }
},
"CapacityProviderAssociation": {
    "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
    "Properties": {
        "CapacityProviders": [
            {
                "Ref": "CapacityProvider"
            }
        ],
        "Cluster": {
            "Ref": "ECSCluster"
        },
        "DefaultCapacityProviderStrategy": [
            {
                "Base": 0,
                "CapacityProvider": {
                    "Ref": "CapacityProvider"
                },
                "Weight": 1
            }
        ]
    }
},
"ContainerHostSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Access to the EC2 hosts that run containers",
        "VpcId": {
            "Ref": "VpcId"
        }
    }
}
```



```

    },
    "EC2Role": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "ec2.amazonaws.com"
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role",
          "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
        ]
      }
    },
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "ecs-tasks.amazonaws.com"
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ],
              "Condition": {
                "ArnLike": {

```

```

        "aws:SourceArn": {
            "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
        }
    },
    "StringEquals": {
        "aws:SourceAccount": {
            "Fn::Sub": "${AWS::AccountId}"
        }
    }
}

],
},
"Path": "/",
"ManagedPolicyArns": [
    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
]
}
}
},
"Outputs": {
    "ClusterName": {
        "Description": "The ECS cluster into which to launch resources",
        "Value": "ECSCluster"
    },
    "ECSTaskExecutionRole": {
        "Description": "The role used to start up a task",
        "Value": "ECSTaskExecutionRole"
    },
    "CapacityProvider": {
        "Description": "The cluster capacity provider that the service should use to
request capacity when it wants to start up a task",
        "Value": "CapacityProvider"
    }
}
}

```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet.

An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.

Parameters:

InstanceType:

Type: String

Description: EC2 instance type

Default: "t2.medium"

AllowedValues:

- t1.micro
- t2.2xlarge
- t2.large
- t2.medium
- t2.micro
- t2.nano
- t2.small
- t2.xlarge
- t3.2xlarge
- t3.large
- t3.medium
- t3.micro
- t3.nano
- t3.small
- t3.xlarge

DesiredCapacity:

Type: Number

Default: "0"

Description: Number of EC2 instances to launch in your ECS cluster.

MaxSize:

Type: Number

Default: "100"

Description: Maximum number of EC2 instances that can be launched in your ECS cluster.

ECSAMI:

Description: The Amazon Machine Image ID used for the cluster

Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id

VpcId:

Type: AWS::EC2::VPC::Id

Description: VPC ID where the ECS cluster is launched

Default: vpc-1234567890abcdef0

SubnetIds:

Type: List<AWS::EC2::Subnet::Id>

Description: List of subnet IDs where the EC2 instances will be launched

```
    Default: "subnet-021345abcdef67890"
Resources:
# This is authorizes ECS to manage resources on your
# account on your behalf. This role is likely already created on your account
# ECSRole:
#   Type: AWS::IAM::ServiceLinkedRole
#   Properties:
#     AWSServiceName: 'ecs.amazonaws.com'

# ECS Resources
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterSettings:
      - Name: containerInsights
        Value: enabled

# Autoscaling group. This launches the actual EC2 instances that will register
# themselves as members of the cluster, and run the docker containers.
ECSAutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn:
    # This is to ensure that the ASG gets deleted first before these
    # resources, when it comes to stack teardown.
    - ECSCluster
    - EC2Role
  Properties:
    VPCZoneIdentifier:
      Ref: SubnetIds
    LaunchTemplate:
      LaunchTemplateId: !Ref ContainerInstances
      Version: !GetAtt ContainerInstances.LatestVersionNumber
    MinSize: 0
    MaxSize:
      Ref: MaxSize
    DesiredCapacity:
      Ref: DesiredCapacity
    NewInstancesProtectedFromScaleIn: true
  UpdatePolicy:
    AutoScalingReplacingUpdate:
      WillReplace: "true"
# The config for each instance that is added to the cluster
ContainerInstances:
  Type: AWS::EC2::LaunchTemplate
```

```

Properties:
  LaunchTemplateName: "asg-launch-template"
  LaunchTemplateData:
    ImageId:
      Ref: ECSAMI
    InstanceType:
      Ref: InstanceType
    IamInstanceProfile:
      Name: !Ref EC2InstanceProfile
    SecurityGroupIds:
      - !Ref ContainerHostSecurityGroup
    # This injected configuration file is how the EC2 instance
    # knows which ECS cluster on your AWS account it should be joining
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash -xe
        echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
        yum install -y aws-cfn-bootstrap
        /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
        # Disable IMDSv1, and require IMDSv2
    MetadataOptions:
      HttpEndpoint: enabled
      HttpTokens: required
  EC2InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
        - !Ref EC2Role
    # Create an ECS capacity provider to attach the ASG to the ECS cluster
    # so that it autoscales as we launch more containers
  CapacityProvider:
    Type: AWS::ECS::CapacityProvider
    Properties:
      AutoScalingGroupProvider:
        AutoScalingGroupArn: !Ref ECSAutoScalingGroup
      ManagedScaling:
        InstanceWarmupPeriod: 60
        MinimumScalingStepSize: 1
        MaximumScalingStepSize: 100
        Status: ENABLED
        # Percentage of cluster reservation to try to maintain
        TargetCapacity: 100

```

```

    ManagedTerminationProtection: ENABLED
  # Create a cluster capacity provider association so that the cluster
  # will use the capacity provider
  CapacityProviderAssociation:
    Type: AWS::ECS::ClusterCapacityProviderAssociations
    Properties:
      CapacityProviders:
        - !Ref CapacityProvider
      Cluster: !Ref ECSCluster
      DefaultCapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
  # A security group for the EC2 hosts that will run the containers.
  # This can be used to limit incoming traffic to or outgoing traffic
  # from the container's host EC2 instance.
  ContainerHostSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the EC2 hosts that run containers
      VpcId:
        Ref: VpcId
  # Role for the EC2 hosts. This allows the ECS agent on the EC2 hosts
  # to communicate with the ECS control plane, as well as download the docker
  # images from ECR to run on your host.
  EC2Role:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
        # See reference: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/
        # security-iam-awsmanpol.html#security-iam-awsmanpol-AmazonEC2ContainerServiceforEC2Role
        - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
        # This managed policy allows us to connect to the instance using SSM
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
  # This is a role which is used within Fargate to allow the Fargate agent

```

```

# to download images, and upload logs.
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ecs-tasks.amazonaws.com
          Action:
            - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount: !Sub ${AWS::AccountId}
      Path: /
    # This role enables all features of ECS. See reference:
    # https://docs.aws.amazon.com/AmazonECS/latest/developerguide/security-iam-awsmanpol.html#security-iam-awsmanpol-AmazonECSTaskExecutionRolePolicy
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
      request capacity when it wants to start up a task
    Value: CapacityProvider

```

サービスをデプロイします。

次のテンプレートは、キャパシティープロバイダーを使用して AL2023 キャパシティーの実行をリクエストするサービスを定義しています。コンテナは、オンラインになると AL2023 インスタンスで起動します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "An example service that deploys in AWS VPC networking mode on EC2 capacity. Service uses a capacity provider to request EC2 instances to run on. Service runs with networking in private subnets, but still accessible to the internet via a load balancer hosted in public subnets.",
  "Parameters": {
    "VpcId": {
      "Type": "String",
      "Description": "The VPC that the service is running inside of"
    },
    "PublicSubnetIds": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "List of public subnet ID's to put the load balancer in"
    },
    "PrivateSubnetIds": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "List of private subnet ID's that the AWS VPC tasks are in"
    },
    "ClusterName": {
      "Type": "String",
      "Description": "The name of the ECS cluster into which to launch capacity."
    },
    "ECSTaskExecutionRole": {
      "Type": "String",
      "Description": "The role used to start up an ECS task"
    },
    "CapacityProvider": {
      "Type": "String",
      "Description": "The cluster capacity provider that the service should use to request capacity when it wants to start up a task"
    },
    "ServiceName": {
      "Type": "String",
      "Default": "web",
      "Description": "A name for the service"
    },
    "ImageUrl": {
      "Type": "String",
      "Default": "public.ecr.aws/docker/library/nginx:latest",
      "Description": "The url of a docker image that contains the application process that will handle the traffic for this service"
    }
  }
}
```



```
    },
    "ContainerCpu": {
      "Type": "Number",
      "Default": 256,
      "Description": "How much CPU to give the container. 1024 is 1 CPU"
    },
    "ContainerMemory": {
      "Type": "Number",
      "Default": 512,
      "Description": "How much memory in megabytes to give the container"
    },
    "ContainerPort": {
      "Type": "Number",
      "Default": 80,
      "Description": "What port that the application expects traffic on"
    },
    "DesiredCount": {
      "Type": "Number",
      "Default": 2,
      "Description": "How many copies of the service task to run"
    }
  },
  "Resources": {
    "TaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "Family": {
          "Ref": "ServiceName"
        },
        "Cpu": {
          "Ref": "ContainerCpu"
        },
        "Memory": {
          "Ref": "ContainerMemory"
        },
        "NetworkMode": "awsvpc",
        "RequiresCompatibilities": [
          "EC2"
        ],
        "ExecutionRoleArn": {
          "Ref": "ECSTaskExecutionRole"
        },
        "ContainerDefinitions": [
          {
```

```

        "Name": {
            "Ref": "ServiceName"
        },
        "Cpu": {
            "Ref": "ContainerCpu"
        },
        "Memory": {
            "Ref": "ContainerMemory"
        },
        "Image": {
            "Ref": "ImageUrl"
        },
        "PortMappings": [
            {
                "ContainerPort": {
                    "Ref": "ContainerPort"
                },
                "HostPort": {
                    "Ref": "ContainerPort"
                }
            }
        ],
        "LogConfiguration": {
            "LogDriver": "awslogs",
            "Options": {
                "mode": "non-blocking",
                "max-buffer-size": "25m",
                "awslogs-group": {
                    "Ref": "LogGroup"
                },
                "awslogs-region": {
                    "Ref": "AWS::Region"
                },
                "awslogs-stream-prefix": {
                    "Ref": "ServiceName"
                }
            }
        }
    }
}

},
"Service": {
    "Type": "AWS::ECS::Service",

```

```
"DependsOn": "PublicLoadBalancerListener",
"Properties": {
  "ServiceName": {
    "Ref": "ServiceName"
  },
  "Cluster": {
    "Ref": "ClusterName"
  },
  "PlacementStrategies": [
    {
      "Field": "attribute:ecs.availability-zone",
      "Type": "spread"
    },
    {
      "Field": "cpu",
      "Type": "binpack"
    }
  ],
  "CapacityProviderStrategy": [
    {
      "Base": 0,
      "CapacityProvider": {
        "Ref": "CapacityProvider"
      },
      "Weight": 1
    }
  ],
  "NetworkConfiguration": {
    "AwsvpcConfiguration": {
      "SecurityGroups": [
        {
          "Ref": "ServiceSecurityGroup"
        }
      ],
      "Subnets": {
        "Ref": "PrivateSubnetIds"
      }
    }
  },
  "DeploymentConfiguration": {
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 75
  },
  "DesiredCount": {
```

```

        "Ref": "DesiredCount"
    },
    "TaskDefinition": {
        "Ref": "TaskDefinition"
    },
    "LoadBalancers": [
        {
            "ContainerName": {
                "Ref": "ServiceName"
            },
            "ContainerPort": {
                "Ref": "ContainerPort"
            },
            "TargetGroupArn": {
                "Ref": "ServiceTargetGroup"
            }
        }
    ]
},
"ServiceSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Security group for service",
        "VpcId": {
            "Ref": "VpcId"
        }
    }
},
"ServiceTargetGroup": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "HealthCheckIntervalSeconds": 6,
        "HealthCheckPath": "/",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 2,
        "TargetType": "ip",
        "Port": {
            "Ref": "ContainerPort"
        },
        "Protocol": "HTTP",
        "UnhealthyThresholdCount": 10,
        "VpcId": {

```

```
        "Ref": "VpcId"
      },
      "TargetGroupAttributes": [
        {
          "Key": "deregistration_delay.timeout_seconds",
          "Value": 0
        }
      ]
    }
  },
  "PublicLoadBalancerSG": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Access to the public facing load balancer",
      "VpcId": {
        "Ref": "VpcId"
      },
      "SecurityGroupIngress": [
        {
          "CidrIp": "0.0.0.0/0",
          "IpProtocol": -1
        }
      ]
    }
  },
  "PublicLoadBalancer": {
    "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
    "Properties": {
      "Scheme": "internet-facing",
      "LoadBalancerAttributes": [
        {
          "Key": "idle_timeout.timeout_seconds",
          "Value": "30"
        }
      ],
      "Subnets": {
        "Ref": "PublicSubnetIds"
      },
      "SecurityGroups": [
        {
          "Ref": "PublicLoadBalancerSG"
        }
      ]
    }
  }
}
```

```

    },
    "PublicLoadBalancerListener": {
      "Type": "AWS::ElasticLoadBalancingV2::Listener",
      "Properties": {
        "DefaultActions": [
          {
            "Type": "forward",
            "ForwardConfig": {
              "TargetGroups": [
                {
                  "TargetGroupArn": {
                    "Ref": "ServiceTargetGroup"
                  },
                  "Weight": 100
                }
              ]
            }
          }
        ],
        "LoadBalancerArn": {
          "Ref": "PublicLoadBalancer"
        },
        "Port": 80,
        "Protocol": "HTTP"
      }
    },
    "ServiceIngressfromLoadBalancer": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "Description": "Ingress from the public ALB",
        "GroupId": {
          "Ref": "ServiceSecurityGroup"
        },
        "IpProtocol": -1,
        "SourceSecurityGroupId": {
          "Ref": "PublicLoadBalancerSG"
        }
      }
    },
    "LogGroup": {
      "Type": "AWS::Logs::LogGroup"
    }
  }
}

```

```
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service
  runs with networking in private subnets, but still accessible to the internet
  via a load balancer hosted in public subnets.
Parameters:
  VpcId:
    Type: String
    Description: The VPC that the service is running inside of
  PublicSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of public subnet ID's to put the load balancer in
  PrivateSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of private subnet ID's that the AWS VPC tasks are in
  ClusterName:
    Type: String
    Description: The name of the ECS cluster into which to launch capacity.
  ECSTaskExecutionRole:
    Type: String
    Description: The role used to start up an ECS task
  CapacityProvider:
    Type: String
    Description: >-
      The cluster capacity provider that the service should use to request
      capacity when it wants to start up a task
  ServiceName:
    Type: String
    Default: web
    Description: A name for the service
  ImageUrl:
    Type: String
    Default: 'public.ecr.aws/docker/library/nginx:latest'
    Description: >-
      The url of a docker image that contains the application process that will
      handle the traffic for this service
  ContainerCpu:
    Type: Number
```

```
    Default: 256
    Description: How much CPU to give the container. 1024 is 1 CPU
ContainerMemory:
    Type: Number
    Default: 512
    Description: How much memory in megabytes to give the container
ContainerPort:
    Type: Number
    Default: 80
    Description: What port that the application expects traffic on
DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
Resources:
    TaskDefinition:
        Type: AWS::ECS::TaskDefinition
        Properties:
            Family: !Ref ServiceName
            Cpu: !Ref ContainerCpu
            Memory: !Ref ContainerMemory
            NetworkMode: awsvpc
            RequiresCompatibilities:
                - EC2
            ExecutionRoleArn: !Ref ECSTaskExecutionRole
            ContainerDefinitions:
                - Name: !Ref ServiceName
                  Cpu: !Ref ContainerCpu
                  Memory: !Ref ContainerMemory
                  Image: !Ref ImageUrl
                  PortMappings:
                      - ContainerPort: !Ref ContainerPort
                        HostPort: !Ref ContainerPort
                  LogConfiguration:
                      LogDriver: awslogs
                      Options:
                          mode: non-blocking
                          max-buffer-size: 25m
                          awslogs-group: !Ref LogGroup
                          awslogs-region: !Ref AWS::Region
                          awslogs-stream-prefix: !Ref ServiceName
    Service:
        Type: AWS::ECS::Service
        DependsOn: PublicLoadBalancerListener
```


Properties:

ServiceName: !Ref ServiceName

Cluster: !Ref ClusterName

PlacementStrategies:

- Field: 'attribute:ecs.availability-zone'
Type: spread
- Field: cpu
Type: binpack

CapacityProviderStrategy:

- Base: 0
CapacityProvider: !Ref CapacityProvider
Weight: 1

NetworkConfiguration:**AwsvpcConfiguration:****SecurityGroups:**

- !Ref ServiceSecurityGroup

Subnets: !Ref PrivateSubnetIds

DeploymentConfiguration:

MaximumPercent: 200

MinimumHealthyPercent: 75

DesiredCount: !Ref DesiredCount

TaskDefinition: !Ref TaskDefinition

LoadBalancers:

- ContainerName: !Ref ServiceName
ContainerPort: !Ref ContainerPort
TargetGroupArn: !Ref ServiceTargetGroup

ServiceSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Security group for service

VpcId: !Ref VpcId

ServiceTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

HealthCheckIntervalSeconds: 6

HealthCheckPath: /

HealthCheckProtocol: HTTP

HealthCheckTimeoutSeconds: 5

HealthyThresholdCount: 2

TargetType: ip

Port: !Ref ContainerPort

Protocol: HTTP

UnhealthyThresholdCount: 10

VpcId: !Ref VpcId

```
TargetGroupAttributes:
  - Key: deregistration_delay.timeout_seconds
    Value: 0
PublicLoadBalancerSG:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Access to the public facing load balancer
    VpcId: !Ref VpcId
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        IpProtocol: -1
PublicLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internet-facing
    LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
    Subnets: !Ref PublicSubnetIds
    SecurityGroups:
      - !Ref PublicLoadBalancerSG
PublicLoadBalancerListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        ForwardConfig:
          TargetGroups:
            - TargetGroupArn: !Ref ServiceTargetGroup
              Weight: 100
    LoadBalancerArn: !Ref PublicLoadBalancer
    Port: 80
    Protocol: HTTP
ServiceIngressfromLoadBalancer:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    Description: Ingress from the public ALB
    GroupId: !Ref ServiceSecurityGroup
    IpProtocol: -1
    SourceSecurityGroupId: !Ref PublicLoadBalancerSG
LogGroup:
  Type: AWS::Logs::LogGroup
```

Amazon Elastic File System サンプルテンプレート

Amazon Elastic File System (Amazon EFS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのファイルストレージサービスです。Amazon EFS を使用すると、ファイルの追加および削除に合わせてストレージ容量が自動的に拡大/縮小するため、アプリケーションには必要なときにストレージが提供されます。

以下のサンプルテンプレートは、Amazon EFS ファイルシステムに関連付けられる EC2 インスタンス (Auto Scaling グループ内) をデプロイします。ファイルシステムにインスタンスに関連付けるために、インスタンスは cfn-init ヘルパースクリプトを実行します。これにより、nfs-utils yum パッケージがダウンロードおよびインストールされて、新しいディレクトリが作成され、ファイルシステムの DNS 名を使用してそのディレクトリにファイルシステムがマウントされます。ファイルシステムの DNS 名は、Amazon EC2 インスタンスのアベイラビリティゾーンでマウントターゲットの IP アドレスに解決されます。DNS 名構造については、「Amazon Elastic File System ユーザーガイド」の「[Mounting File Systems](#)」(ファイルシステムのマウント) を参照してください。

Network File System (NFS) のアクティビティを測定するために、テンプレートにはカスタムの Amazon CloudWatch メトリクスが含まれています。このテンプレートでは、VPC、サブネット、およびセキュリティグループも作成します。インスタンスとファイルシステムとの通信を許可するには、VPC で DNS が有効になっていて、マウントターゲットと EC2 インスタンスはサブネットに指定された同じアベイラビリティゾーン (AZ) にある必要があります。

マウントターゲットのセキュリティグループは TCP ポート 2049 へのネットワーク接続を有効にします。これは、NFSv4 クライアントがファイルシステムをマウントするために必要です。EC2 インスタンスとマウントターゲット向けのセキュリティグループの詳細については、「[Amazon Elastic File System User Guide](#)」(Amazon Elastic File System ユーザーガイド) の「[Security](#)」(セキュリティ) を参照してください。

Note

置き換えの原因となる更新をマウントターゲットに対して行った場合は、関連するファイルシステムを使用するインスタンスまたはアプリケーションが中断される可能性があります。このため、コミットされていない書き込みが失われることがあります。中断を回避するには、希望する容量をゼロに設定することで、マウントターゲットの更新時にインスタンスを停止します。これにより、マウントターゲットを削除する前に、インスタンスがファイルシステムをアンマウントできるようになります。マウントの更新が完了したら、希望する容量を設定して後続の更新のインスタンスを開始します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This template creates an Amazon EFS file system and mount target and associates it with Amazon EC2 instances in an Auto Scaling group. **WARNING** This template creates Amazon EC2 instances and related resources. You will be billed for the AWS resources used if you create a stack from this template.",
  "Parameters": {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [
        "t1.micro",
        "t2.nano",
        "t2.micro",
        "t2.small",
        "t2.medium",
        "t2.large",
        "m1.small",
        "m1.medium",
        "m1.large",
        "m1.xlarge",
        "m2.xlarge",
        "m2.2xlarge",
        "m2.4xlarge",
        "m3.medium",
        "m3.large",
        "m3.xlarge",
        "m3.2xlarge",
        "m4.large",
        "m4.xlarge",
        "m4.2xlarge",
        "m4.4xlarge",
        "m4.10xlarge",
        "c1.medium",
        "c1.xlarge",
        "c3.large",
        "c3.xlarge",
        "c3.2xlarge",
        "c3.4xlarge",
        "c3.8xlarge",
```

```

        "c4.large",
        "c4.xlarge",
        "c4.2xlarge",
        "c4.4xlarge",
        "c4.8xlarge",
        "g2.2xlarge",
        "g2.8xlarge",
        "r3.large",
        "r3.xlarge",
        "r3.2xlarge",
        "r3.4xlarge",
        "r3.8xlarge",
        "i2.xlarge",
        "i2.2xlarge",
        "i2.4xlarge",
        "i2.8xlarge",
        "d2.xlarge",
        "d2.2xlarge",
        "d2.4xlarge",
        "d2.8xlarge",
        "hi1.4xlarge",
        "hs1.8xlarge",
        "cr1.8xlarge",
        "cc2.8xlarge",
        "cg1.4xlarge"
    ],
    "ConstraintDescription" : "must be a valid EC2 instance type."
},
"KeyName": {
    "Type": "AWS::EC2::KeyPair::KeyName",
    "Description": "Name of an existing EC2 key pair to enable SSH access to the EC2
instances"
},
"AsgMaxSize": {
    "Type": "Number",
    "Description": "Maximum size and initial desired capacity of Auto Scaling Group",
    "Default": "2"
},
"SSHLocation" : {
    "Description" : "The IP address range that can be used to connect to the EC2
instances by using SSH",
    "Type": "String",
    "MinLength": "9",
    "MaxLength": "18",

```

```

    "Default": "0.0.0.0/0",
    "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})",
    "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
  },
  "VolumeName" : {
    "Description" : "The name to be used for the EFS volume",
    "Type": "String",
    "MinLength": "1",
    "Default": "myEFSvolume"
  },
  "MountPoint" : {
    "Description" : "The Linux mount point for the EFS volume",
    "Type": "String",
    "MinLength": "1",
    "Default": "myEFSvolume"
  }
},
"Mappings" : {
  "AWSInstanceType2Arch" : {
    "t1.micro"      : { "Arch" : "HVM64" },
    "t2.nano"      : { "Arch" : "HVM64" },
    "t2.micro"      : { "Arch" : "HVM64" },
    "t2.small"     : { "Arch" : "HVM64" },
    "t2.medium"    : { "Arch" : "HVM64" },
    "t2.large"     : { "Arch" : "HVM64" },
    "m1.small"     : { "Arch" : "HVM64" },
    "m1.medium"    : { "Arch" : "HVM64" },
    "m1.large"     : { "Arch" : "HVM64" },
    "m1.xlarge"    : { "Arch" : "HVM64" },
    "m2.xlarge"    : { "Arch" : "HVM64" },
    "m2.2xlarge"   : { "Arch" : "HVM64" },
    "m2.4xlarge"   : { "Arch" : "HVM64" },
    "m3.medium"    : { "Arch" : "HVM64" },
    "m3.large"     : { "Arch" : "HVM64" },
    "m3.xlarge"    : { "Arch" : "HVM64" },
    "m3.2xlarge"   : { "Arch" : "HVM64" },
    "m4.large"     : { "Arch" : "HVM64" },
    "m4.xlarge"    : { "Arch" : "HVM64" },
    "m4.2xlarge"   : { "Arch" : "HVM64" },
    "m4.4xlarge"   : { "Arch" : "HVM64" },
    "m4.10xlarge"  : { "Arch" : "HVM64" },
    "c1.medium"    : { "Arch" : "HVM64" },
    "c1.xlarge"    : { "Arch" : "HVM64" },
    "c3.large"     : { "Arch" : "HVM64" },

```

```

    "c3.xlarge" : { "Arch" : "HVM64" },
    "c3.2xlarge" : { "Arch" : "HVM64" },
    "c3.4xlarge" : { "Arch" : "HVM64" },
    "c3.8xlarge" : { "Arch" : "HVM64" },
    "c4.large" : { "Arch" : "HVM64" },
    "c4.xlarge" : { "Arch" : "HVM64" },
    "c4.2xlarge" : { "Arch" : "HVM64" },
    "c4.4xlarge" : { "Arch" : "HVM64" },
    "c4.8xlarge" : { "Arch" : "HVM64" },
    "g2.2xlarge" : { "Arch" : "HVMG2" },
    "g2.8xlarge" : { "Arch" : "HVMG2" },
    "r3.large" : { "Arch" : "HVM64" },
    "r3.xlarge" : { "Arch" : "HVM64" },
    "r3.2xlarge" : { "Arch" : "HVM64" },
    "r3.4xlarge" : { "Arch" : "HVM64" },
    "r3.8xlarge" : { "Arch" : "HVM64" },
    "i2.xlarge" : { "Arch" : "HVM64" },
    "i2.2xlarge" : { "Arch" : "HVM64" },
    "i2.4xlarge" : { "Arch" : "HVM64" },
    "i2.8xlarge" : { "Arch" : "HVM64" },
    "d2.xlarge" : { "Arch" : "HVM64" },
    "d2.2xlarge" : { "Arch" : "HVM64" },
    "d2.4xlarge" : { "Arch" : "HVM64" },
    "d2.8xlarge" : { "Arch" : "HVM64" },
    "hi1.4xlarge" : { "Arch" : "HVM64" },
    "hs1.8xlarge" : { "Arch" : "HVM64" },
    "cr1.8xlarge" : { "Arch" : "HVM64" },
    "cc2.8xlarge" : { "Arch" : "HVM64" }
  },
  "AWSRegionArch2AMI" : {
    "us-east-1" : { "HVM64" : "ami-0ff8a91507f77f867", "HVMG2" :
"ami-0a584ac55a7631c0c"},
    "us-west-2" : { "HVM64" : "ami-a0cfeed8", "HVMG2" :
"ami-0e09505bc235aa82d"},
    "us-west-1" : { "HVM64" : "ami-0bdb828fd58c52235", "HVMG2" :
"ami-066ee5fd4a9ef77f1"},
    "eu-west-1" : { "HVM64" : "ami-047bb4163c506cd98", "HVMG2" :
"ami-0a7c483d527806435"},
    "eu-west-2" : { "HVM64" : "ami-f976839e", "HVMG2" : "NOT_SUPPORTED"},
    "eu-west-3" : { "HVM64" : "ami-0ebc281c20e89ba4b", "HVMG2" :
"NOT_SUPPORTED"},
    "eu-central-1" : { "HVM64" : "ami-0233214e13e500f77", "HVMG2" :
"ami-06223d46a6d0661c7"},

```

```

    "ap-northeast-1" : {"HVM64" : "ami-06cd52961ce9f0d85", "HVMG2" :
"ami-053cdd503598e4a9d"},
    "ap-northeast-2" : {"HVM64" : "ami-0a10b2721688ce9d2", "HVMG2" :
"NOT_SUPPORTED"},
    "ap-northeast-3" : {"HVM64" : "ami-0d98120a9fb693f07", "HVMG2" :
"NOT_SUPPORTED"},
    "ap-southeast-1" : {"HVM64" : "ami-08569b978cc4dfa10", "HVMG2" :
"ami-0be9df32ae9f92309"},
    "ap-southeast-2" : {"HVM64" : "ami-09b42976632b27e9b", "HVMG2" :
"ami-0a9ce9fecc3d1daf8"},
    "ap-south-1" : {"HVM64" : "ami-0912f71e06545ad88", "HVMG2" :
"ami-097b15e89dbdcfcf4"},
    "us-east-2" : {"HVM64" : "ami-0b59bfac6be064b78", "HVMG2" :
"NOT_SUPPORTED"},
    "ca-central-1" : {"HVM64" : "ami-0b18956f", "HVMG2" : "NOT_SUPPORTED"},
    "sa-east-1" : {"HVM64" : "ami-07b14488da8ea02a0", "HVMG2" :
"NOT_SUPPORTED"},
    "cn-north-1" : {"HVM64" : "ami-0a4eaf6c4454eda75", "HVMG2" :
"NOT_SUPPORTED"},
    "cn-northwest-1" : {"HVM64" : "ami-6b6a7d09", "HVMG2" : "NOT_SUPPORTED"}
  },
  "Resources": {
    "CloudWatchPutMetricsRole" : {
      "Type" : "AWS::IAM::Role",
      "Properties" : {
        "AssumeRolePolicyDocument" : {
          "Statement" : [ {
            "Effect" : "Allow",
            "Principal" : {
              "Service" : [ "ec2.amazonaws.com" ]
            },
            "Action" : [ "sts:AssumeRole" ]
          } ]
        },
        "Path" : "/"
      }
    },
    "CloudWatchPutMetricsRolePolicy" : {
      "Type" : "AWS::IAM::Policy",
      "Properties" : {
        "PolicyName" : "CloudWatch_PutMetricData",
        "PolicyDocument" : {
          "Version": "2012-10-17",

```



```

        "Statement": [
            {
                "Sid": "CloudWatchPutMetricData",
                "Effect": "Allow",
                "Action": ["cloudwatch:PutMetricData"],
                "Resource": ["*"]
            }
        ],
        "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
    }
},
"CloudWatchPutMetricsInstanceProfile" : {
    "Type" : "AWS::IAM::InstanceProfile",
    "Properties" : {
        "Path" : "/",
        "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
    }
},
"VPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "EnableDnsSupport" : "true",
        "EnableDnsHostnames" : "true",
        "CidrBlock": "10.0.0.0/16",
        "Tags": [ { "Key": "Application", "Value": { "Ref": "AWS::StackId" } } ]
    }
},
"InternetGateway" : {
    "Type" : "AWS::EC2::InternetGateway",
    "Properties" : {
        "Tags" : [
            { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },
            { "Key" : "Network", "Value" : "Public" }
        ]
    }
},
"GatewayToInternet" : {
    "Type" : "AWS::EC2::VPCGatewayAttachment",
    "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "InternetGatewayId" : { "Ref" : "InternetGateway" }
    }
},

```

```

"RouteTable":{
  "Type":"AWS::EC2::RouteTable",
  "Properties":{
    "VpcId": {"Ref":"VPC"}
  }
},
"SubnetRouteTableAssoc": {
  "Type" : "AWS::EC2::SubnetRouteTableAssociation",
  "Properties" : {
    "RouteTableId" : {"Ref":"RouteTable"},
    "SubnetId" : {"Ref":"Subnet"}
  }
},
"InternetGatewayRoute": {
  "Type":"AWS::EC2::Route",
  "Properties":{
    "DestinationCidrBlock":"0.0.0.0/0",
    "RouteTableId":{"Ref":"RouteTable"},
    "GatewayId":{"Ref":"InternetGateway"}
  }
},
"Subnet": {
  "Type": "AWS::EC2::Subnet",
  "Properties": {
    "VpcId": { "Ref": "VPC" },
    "CidrBlock": "10.0.0.0/24",
    "Tags": [ { "Key": "Application", "Value": { "Ref": "AWS::StackId" } } ]
  }
},
"InstanceSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "VpcId": { "Ref": "VPC" },
    "GroupDescription": "Enable SSH access via port 22",
    "SecurityGroupIngress": [
      { "IpProtocol": "tcp", "FromPort": 22, "ToPort": 22, "CidrIp": { "Ref":
"SSHLocation" } },
      { "IpProtocol": "tcp", "FromPort": 80, "ToPort": 80, "CidrIp": "0.0.0.0/0" }
    ]
  }
},
"MountTargetSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {

```

```
"VpcId": { "Ref": "VPC" },
"GroupDescription": "Security group for mount target",
"SecurityGroupIngress": [
  {
    "IpProtocol": "tcp",
    "FromPort": 2049,
    "ToPort": 2049,
    "CidrIp": "0.0.0.0/0"
  }
]
},
"FileSystem": {
  "Type": "AWS::EFS::FileSystem",
  "Properties": {
    "PerformanceMode": "generalPurpose",
    "FileSystemTags": [
      {
        "Key": "Name",
        "Value": { "Ref" : "VolumeName" }
      }
    ]
  }
},
"MountTarget": {
  "Type": "AWS::EFS::MountTarget",
  "Properties": {
    "FileSystemId": { "Ref": "FileSystem" },
    "SubnetId": { "Ref": "Subnet" },
    "SecurityGroups": [ { "Ref": "MountTargetSecurityGroup" } ]
  }
},
"LaunchConfiguration": {
  "Type": "AWS::AutoScaling::LaunchConfiguration",
  "Metadata" : {
    "AWS::CloudFormation::Init" : {
      "configSets" : {
        "MountConfig" : [ "setup", "mount" ]
      },
      "setup" : {
        "packages" : {
          "yum" : {
            "nfs-utils" : []
          }
        }
      }
    }
  }
}
```

```

    },
    "files" : {
      "/home/ec2-user/post_nfsstat" : {
        "content" : { "Fn::Join" : [ "", [
          "#!/bin/bash\n",
          "\n",
          "INPUT=\"$(cat)\"\n",
          "CW_JSON_OPEN='{ \"Namespace\": \"EFS\", \"MetricData\": [ '\n",
          "CW_JSON_CLOSE=' ] }'\n",
          "CW_JSON_METRIC=''\n",
          "METRIC_COUNTER=0\n",
          "\n",
          "for COL in 1 2 3 4 5 6; do\n",
          "\n",
          "  COUNTER=0\n",
          "  METRIC_FIELD=$COL\n",
          "  DATA_FIELD=$((COL+(COL-1)))\n",
          "\n",
          "  while read line; do\n",
          "    if [[ COUNTER -gt 0 ]]; then\n",
          "\n",
          "      LINE=`echo $line | tr -s ' ' '\n",
          "      AWS_COMMAND=\"aws cloudwatch put-metric-data --region ",
          { "Ref": "AWS::Region" }, "\"\n",
          "      MOD=$(( COUNTER % 2))\n",
          "\n",
          "      if [ $MOD -eq 1 ]; then\n",
          "        METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`\n",
          "\n",
          "        else\n",
          "          METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`\n",
          "          fi\n",
          "\n",
          "          if [[ -n \"$METRIC_NAME\" && -n \"$METRIC_VALUE\" ]]; then\n",
          "\n",
          "            INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-
          data/instance-id)\n",
          "            CW_JSON_METRIC=\"$CW_JSON_METRIC { \\\"MetricName\\\": \\\"
          \"$METRIC_NAME\\\", \\\"Dimensions\\\": [{\\\"Name\\\": \\\"InstanceId\\\", \\\"Value\\\"
          \": \\\"$INSTANCE_ID\\\"} ], \\\"Value\\\": $METRIC_VALUE },\n",
          "\n",
          "            unset METRIC_NAME\n",
          "            unset METRIC_VALUE\n",
          "\n",
          "            METRIC_COUNTER=$((METRIC_COUNTER+1))\n",

```

```

        "        if [ $METRIC_COUNTER -eq 20 ]; then\n",
        "            # 20 is max metric collection size, so we have to
submit here\n",
        "            aws cloudwatch put-metric-data --region ", { "Ref":
"AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
$CW_JSON_CLOSE`\n",
        "\n",
        "            # reset\n",
        "            METRIC_COUNTER=0\n",
        "            CW_JSON_METRIC=' '\n",
        "        fi\n",
        "    fi\n",
        "\n",
        "\n",
        "\n",
        "        COUNTER=$((COUNTER+1))\n",
        "    fi\n",
        "\n",
        "    if [[ \"$line\" == \"Client nfs v4:\" ]]; then\n",
        "        # the next line is the good stuff\n",
        "        COUNTER=$((COUNTER+1))\n",
        "    fi\n",
        "done <<< \"$INPUT\"\n",
        "done\n",
        "\n",
        "# submit whatever is left\n",
        "aws cloudwatch put-metric-data --region ", { "Ref":
"AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
$CW_JSON_CLOSE`\n"
    ] ] },
    "mode": "000755",
    "owner": "ec2-user",
    "group": "ec2-user"
},
"/home/ec2-user/crontab" : {
    "content" : { "Fn::Join" : [ "", [
        "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
    ] ] },
    "owner": "ec2-user",
    "group": "ec2-user"
}
},
"commands" : {
    "01_createdir" : {

```

```

        "command" : { "Fn::Join" : [ "", [ "mkdir /", { "Ref" :
"MountPoint" }]]] }
    }
  },
  "mount" : {
    "commands" : {
      "01_mount" : {
        "command" : { "Fn::Sub": "sudo mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsiz=1048576,hard,timeo=600,retrans=2 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}" }
      },
      "02_permissions" : {
        "command" : { "Fn::Join" : [ "", [ "chown ec2-user:ec2-user /",
{ "Ref" : "MountPoint" }]]] }
      }
    }
  }
},
"Properties": {
  "AssociatePublicIpAddress" : true,
  "ImageId": {
    "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" }, {
      "Fn::FindInMap": [ "AWSInstanceType2Arch", { "Ref": "InstanceType" },
"Arch" ]
    } ]
  },
  "InstanceType": { "Ref": "InstanceType" },
  "KeyName": { "Ref": "KeyName" },
  "SecurityGroups": [ { "Ref": "InstanceSecurityGroup" } ],
  "IamInstanceProfile" : { "Ref" : "CloudWatchPutMetricsInstanceProfile" },
  "UserData" : { "Fn::Base64" : { "Fn::Join" : [ "", [
    "#!/bin/bash -xe\n",
    "yum install -y aws-cfn-bootstrap\n",

    "/opt/aws/bin/cfn-init -v ",
    "      --stack ", { "Ref" : "AWS::StackName" },
    "      --resource LaunchConfiguration ",
    "      --configsets MountConfig ",
    "      --region ", { "Ref" : "AWS::Region" }, "\n",

    "crontab /home/ec2-user/crontab\n",

```

```

        "/opt/aws/bin/cfn-signal -e $? ",
        "--stack ", { "Ref" : "AWS::StackName" },
        "--resource AutoScalingGroup ",
        "--region ", { "Ref" : "AWS::Region" }, "\n"
    ]]]}
}
},
"AutoScalingGroup": {
    "Type": "AWS::AutoScaling::AutoScalingGroup",
    "DependsOn": ["MountTarget", "GatewayToInternet"],
    "CreationPolicy" : {
        "ResourceSignal" : {
            "Timeout" : "PT15M",
            "Count" : { "Ref": "AsgMaxSize" }
        }
    },
    "Properties": {
        "VPCZoneIdentifier": [ { "Ref": "Subnet" } ],
        "LaunchConfigurationName": { "Ref": "LaunchConfiguration" },
        "MinSize": "1",
        "MaxSize": { "Ref": "AsgMaxSize" },
        "DesiredCapacity": { "Ref": "AsgMaxSize" },
        "Tags": [ {
            "Key": "Name",
            "Value": "EFS FileSystem Mounted Instance",
            "PropagateAtLaunch": "true"
        } ]
    }
}
},
"Outputs" : {
    "MountTargetID" : {
        "Description" : "Mount target ID",
        "Value" : { "Ref" : "MountTarget" }
    },
    "FileSystemID" : {
        "Description" : "File system ID",
        "Value" : { "Ref" : "FileSystem" }
    }
}
}
}

```

YAML

AWSTemplateFormatVersion: '2010-09-09'

Description: This template creates an Amazon EFS file system and mount target and associates it with Amazon EC2 instances in an Auto Scaling group. ****WARNING**** This template creates Amazon EC2 instances and related resources. You will be billed for the AWS resources used if you create a stack from this template.

Parameters:

InstanceType:

Description: WebServer EC2 instance type

Type: String

Default: t2.small

AllowedValues:

- t1.micro
- t2.nano
- t2.micro
- t2.small
- t2.medium
- t2.large
- m1.small
- m1.medium
- m1.large
- m1.xlarge
- m2.xlarge
- m2.2xlarge
- m2.4xlarge
- m3.medium
- m3.large
- m3.xlarge
- m3.2xlarge
- m4.large
- m4.xlarge
- m4.2xlarge
- m4.4xlarge
- m4.10xlarge
- c1.medium
- c1.xlarge
- c3.large
- c3.xlarge
- c3.2xlarge
- c3.4xlarge
- c3.8xlarge
- c4.large

- c4.xlarge
- c4.2xlarge
- c4.4xlarge
- c4.8xlarge
- g2.2xlarge
- g2.8xlarge
- r3.large
- r3.xlarge
- r3.2xlarge
- r3.4xlarge
- r3.8xlarge
- i2.xlarge
- i2.2xlarge
- i2.4xlarge
- i2.8xlarge
- d2.xlarge
- d2.2xlarge
- d2.4xlarge
- d2.8xlarge
- hi1.4xlarge
- hs1.8xlarge
- cr1.8xlarge
- cc2.8xlarge
- cg1.4xlarge

ConstraintDescription: must be a valid EC2 instance type.

KeyName:

Type: `AWS::EC2::KeyPair::KeyName`

Description: Name of an existing EC2 key pair to enable SSH access to the ECS instances

AsqMaxSize:

Type: Number

Description: Maximum size and initial desired capacity of Auto Scaling Group

Default: '2'

SSHLocation:

Description: The IP address range that can be used to connect to the EC2 instances by using SSH

Type: String

MinLength: '9'

MaxLength: '18'

Default: 0.0.0.0/0

AllowedPattern: "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})"

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

VolumeName:

Description: The name to be used for the EFS volume

```
Type: String
MinLength: '1'
Default: myEFSvolume
MountPoint:
  Description: The Linux mount point for the EFS volume
  Type: String
  MinLength: '1'
  Default: myEFSvolume
Mappings:
  AWSInstanceType2Arch:
    t1.micro:
      Arch: HVM64
    t2.nano:
      Arch: HVM64
    t2.micro:
      Arch: HVM64
    t2.small:
      Arch: HVM64
    t2.medium:
      Arch: HVM64
    t2.large:
      Arch: HVM64
    m1.small:
      Arch: HVM64
    m1.medium:
      Arch: HVM64
    m1.large:
      Arch: HVM64
    m1.xlarge:
      Arch: HVM64
    m2.xlarge:
      Arch: HVM64
    m2.2xlarge:
      Arch: HVM64
    m2.4xlarge:
      Arch: HVM64
    m3.medium:
      Arch: HVM64
    m3.large:
      Arch: HVM64
    m3.xlarge:
      Arch: HVM64
    m3.2xlarge:
      Arch: HVM64
```

```
m4.large:
  Arch: HVM64
m4.xlarge:
  Arch: HVM64
m4.2xlarge:
  Arch: HVM64
m4.4xlarge:
  Arch: HVM64
m4.10xlarge:
  Arch: HVM64
c1.medium:
  Arch: HVM64
c1.xlarge:
  Arch: HVM64
c3.large:
  Arch: HVM64
c3.xlarge:
  Arch: HVM64
c3.2xlarge:
  Arch: HVM64
c3.4xlarge:
  Arch: HVM64
c3.8xlarge:
  Arch: HVM64
c4.large:
  Arch: HVM64
c4.xlarge:
  Arch: HVM64
c4.2xlarge:
  Arch: HVM64
c4.4xlarge:
  Arch: HVM64
c4.8xlarge:
  Arch: HVM64
g2.2xlarge:
  Arch: HVMG2
g2.8xlarge:
  Arch: HVMG2
r3.large:
  Arch: HVM64
r3.xlarge:
  Arch: HVM64
r3.2xlarge:
  Arch: HVM64
```

```
r3.4xlarge:
  Arch: HVM64
r3.8xlarge:
  Arch: HVM64
i2.xlarge:
  Arch: HVM64
i2.2xlarge:
  Arch: HVM64
i2.4xlarge:
  Arch: HVM64
i2.8xlarge:
  Arch: HVM64
d2.xlarge:
  Arch: HVM64
d2.2xlarge:
  Arch: HVM64
d2.4xlarge:
  Arch: HVM64
d2.8xlarge:
  Arch: HVM64
hi1.4xlarge:
  Arch: HVM64
hs1.8xlarge:
  Arch: HVM64
cr1.8xlarge:
  Arch: HVM64
cc2.8xlarge:
  Arch: HVM64
AWSRegionArch2AMI:
us-east-1:
  HVM64: ami-0ff8a91507f77f867
  HVMG2: ami-0a584ac55a7631c0c
us-west-2:
  HVM64: ami-a0cfeed8
  HVMG2: ami-0e09505bc235aa82d
us-west-1:
  HVM64: ami-0bdb828fd58c52235
  HVMG2: ami-066ee5fd4a9ef77f1
eu-west-1:
  HVM64: ami-047bb4163c506cd98
  HVMG2: ami-0a7c483d527806435
eu-west-2:
  HVM64: ami-f976839e
  HVMG2: NOT_SUPPORTED
```

```
eu-west-3:
  HVM64: ami-0ebc281c20e89ba4b
  HVMG2: NOT_SUPPORTED
eu-central-1:
  HVM64: ami-0233214e13e500f77
  HVMG2: ami-06223d46a6d0661c7
ap-northeast-1:
  HVM64: ami-06cd52961ce9f0d85
  HVMG2: ami-053cdd503598e4a9d
ap-northeast-2:
  HVM64: ami-0a10b2721688ce9d2
  HVMG2: NOT_SUPPORTED
ap-northeast-3:
  HVM64: ami-0d98120a9fb693f07
  HVMG2: NOT_SUPPORTED
ap-southeast-1:
  HVM64: ami-08569b978cc4dfa10
  HVMG2: ami-0be9df32ae9f92309
ap-southeast-2:
  HVM64: ami-09b42976632b27e9b
  HVMG2: ami-0a9ce9fecc3d1daf8
ap-south-1:
  HVM64: ami-0912f71e06545ad88
  HVMG2: ami-097b15e89dbdcfcf4
us-east-2:
  HVM64: ami-0b59bfac6be064b78
  HVMG2: NOT_SUPPORTED
ca-central-1:
  HVM64: ami-0b18956f
  HVMG2: NOT_SUPPORTED
sa-east-1:
  HVM64: ami-07b14488da8ea02a0
  HVMG2: NOT_SUPPORTED
cn-north-1:
  HVM64: ami-0a4eaf6c4454eda75
  HVMG2: NOT_SUPPORTED
cn-northwest-1:
  HVM64: ami-6b6a7d09
  HVMG2: NOT_SUPPORTED
```

Resources:

CloudWatchPutMetricsRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

```
Statement:
- Effect: Allow
Principal:
Service:
- ec2.amazonaws.com
Action:
- sts:AssumeRole
Path: "/"
CloudWatchPutMetricsRolePolicy:
Type: AWS::IAM::Policy
Properties:
PolicyName: CloudWatch_PutMetricData
PolicyDocument:
Version: '2012-10-17'
Statement:
- Sid: CloudWatchPutMetricData
Effect: Allow
Action:
- cloudwatch:PutMetricData
Resource:
- "*"
Roles:
- Ref: CloudWatchPutMetricsRole
CloudWatchPutMetricsInstanceProfile:
Type: AWS::IAM::InstanceProfile
Properties:
Path: "/"
Roles:
- Ref: CloudWatchPutMetricsRole
VPC:
Type: AWS::EC2::VPC
Properties:
EnableDnsSupport: 'true'
EnableDnsHostnames: 'true'
CidrBlock: 10.0.0.0/16
Tags:
- Key: Application
Value:
Ref: AWS::StackId
InternetGateway:
Type: AWS::EC2::InternetGateway
Properties:
Tags:
- Key: Application
```

```
    Value:
      Ref: AWS::StackName
  - Key: Network
    Value: Public
GatewayToInternet:
  Type: AWS::EC2::VPCEGatewayAttachment
  Properties:
    VpcId:
      Ref: VPC
    InternetGatewayId:
      Ref: InternetGateway
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: VPC
SubnetRouteTableAssoc:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId:
      Ref: RouteTable
    SubnetId:
      Ref: Subnet
InternetGatewayRoute:
  Type: AWS::EC2::Route
  Properties:
    DestinationCidrBlock: 0.0.0.0/0
    RouteTableId:
      Ref: RouteTable
    GatewayId:
      Ref: InternetGateway
Subnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId:
      Ref: VPC
    CidrBlock: 10.0.0.0/24
    Tags:
      - Key: Application
        Value:
          Ref: AWS::StackId
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
```

```
VpcId:
  Ref: VPC
GroupDescription: Enable SSH access via port 22
SecurityGroupIngress:
- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp:
    Ref: SSHLocation
- IpProtocol: tcp
  FromPort: 80
  ToPort: 80
  CidrIp: 0.0.0.0/0
MountTargetSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    VpcId:
      Ref: VPC
    GroupDescription: Security group for mount target
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 2049
      ToPort: 2049
      CidrIp: 0.0.0.0/0
FileSystem:
  Type: AWS::EFS::FileSystem
  Properties:
    PerformanceMode: generalPurpose
    FileSystemTags:
    - Key: Name
      Value:
        Ref: VolumeName
MountTarget:
  Type: AWS::EFS::MountTarget
  Properties:
    FileSystemId:
      Ref: FileSystem
    SubnetId:
      Ref: Subnet
    SecurityGroups:
    - Ref: MountTargetSecurityGroup
LaunchConfiguration:
  Type: AWS::AutoScaling::LaunchConfiguration
Metadata:
```



```

AWS::CloudFormation::Init:
  configSets:
    MountConfig:
      - setup
      - mount
  setup:
    packages:
      yum:
        nfs-utils: []
    files:
      "/home/ec2-user/post_nfsstat":
        content: !Sub |
          #!/bin/bash

          INPUT="$(cat)"
          CW_JSON_OPEN='{ "Namespace": "EFS", "MetricData": [ '
          CW_JSON_CLOSE=' ] }'
          CW_JSON_METRIC=''
          METRIC_COUNTER=0

          for COL in 1 2 3 4 5 6; do

            COUNTER=0
            METRIC_FIELD=$COL
            DATA_FIELD=$(( $COL + ( $COL - 1 ) ))

            while read line; do
              if [[ COUNTER -gt 0 ]]; then

                LINE=`echo $line | tr -s ' ' `
                AWS_COMMAND="aws cloudwatch put-metric-data --region
${AWS::Region}"

                MOD=$(( $COUNTER % 2 ))

                if [ $MOD -eq 1 ]; then
                  METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`
                else
                  METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`
                fi

                if [[ -n "$METRIC_NAME" && -n "$METRIC_VALUE" ]]; then
                  INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/
instance-id)

```

```

        CW_JSON_METRIC="$CW_JSON_METRIC { \"MetricName\": \"$METRIC_NAME
\", \"Dimensions\": [{\"Name\": \"InstanceId\", \"Value\": \"$INSTANCE_ID\"} ], \"Value
\": $METRIC_VALUE },\"

        unset METRIC_NAME
        unset METRIC_VALUE

        METRIC_COUNTER=$((METRIC_COUNTER+1))
        if [ $METRIC_COUNTER -eq 20 ]; then
            # 20 is max metric collection size, so we have to submit here
            aws cloudwatch put-metric-data --region ${AWS::Region} --cli-
input-json "`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"

            # reset
            METRIC_COUNTER=0
            CW_JSON_METRIC=''
        fi
    fi

    COUNTER=$((COUNTER+1))
fi

if [[ "$line" == "Client nfs v4:" ]]; then
    # the next line is the good stuff
    COUNTER=$((COUNTER+1))
fi
done <<< "$INPUT"
done

# submit whatever is left
aws cloudwatch put-metric-data --region ${AWS::Region} --cli-input-json
"`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"
mode: '000755'
owner: ec2-user
group: ec2-user
"/home/ec2-user/crontab":
    content: "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
    owner: ec2-user
    group: ec2-user
commands:
    01_createdir:
        command: !Sub "mkdir /${MountPoint}"
mount:

```

```

    commands:
      01_mount:
        command: !Sub >
          mount -t nfs4 -o nfsvers=4.1 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}
      02_permissions:
        command: !Sub "chown ec2-user:ec2-user /${MountPoint}"
  Properties:
    AssociatePublicIpAddress: true
    ImageId:
      Fn::FindInMap:
        - AWSRegionArch2AMI
        - Ref: AWS::Region
        - Fn::FindInMap:
            - AWSInstanceType2Arch
            - Ref: InstanceType
            - Arch
    InstanceType:
      Ref: InstanceType
    KeyName:
      Ref: KeyName
    SecurityGroups:
      - Ref: InstanceSecurityGroup
    IamInstanceProfile:
      Ref: CloudWatchPutMetricsInstanceProfile
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash -xe
        yum install -y aws-cfn-bootstrap
        /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
LaunchConfiguration --configsets MountConfig --region ${AWS::Region}
        crontab /home/ec2-user/crontab
        /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
AutoScalingGroup --region ${AWS::Region}
    AutoScalingGroup:
      Type: AWS::AutoScaling::AutoScalingGroup
      DependsOn:
        - MountTarget
        - GatewayToInternet
      CreationPolicy:
        ResourceSignal:
          Timeout: PT15M
        Count:
          Ref: AsgMaxSize

```

```
Properties:
  VPCZoneIdentifier:
    - Ref: Subnet
  LaunchConfigurationName:
    Ref: LaunchConfiguration
  MinSize: '1'
  MaxSize:
    Ref: AsgMaxSize
  DesiredCapacity:
    Ref: AsgMaxSize
  Tags:
    - Key: Name
      Value: EFS FileSystem Mounted Instance
    PropagateAtLaunch: 'true'
Outputs:
  MountTargetID:
    Description: Mount target ID
    Value:
      Ref: MountTarget
  FileSystemID:
    Description: File system ID
    Value:
      Ref: FileSystem
```

Elastic Beanstalk テンプレートスニペット

Elastic Beanstalk を使用すると、アプリケーションを実行しているインフラストラクチャについて心配することなく、AWS でアプリケーションのデプロイと管理を簡単に行うことができます。次のサンプルテンプレートは、CloudFormation テンプレート内で Elastic Beanstalk リソースを記述するのに役立ちます。

Elastic Beanstalk のサンプル PHP

次のサンプルテンプレートは、Amazon S3 バケットに保存されているサンプル PHP ウェブアプリケーションをデプロイします。この環境は、最低 2 つ、最大 6 つの Amazon EC2 インスタンスを持つ負荷分散型自動スケーリング環境でもあります。これはレガシー起動設定を使用する Elastic Beanstalk 環境を示します。代わりに起動テンプレートを使用する方法については、「AWS Elastic Beanstalk デベロッパーガイド」の「[起動テンプレート](#)」を参照してください。

solution-stack をソリューションスタック名 (プラットフォームバージョン) に置き換えます。使用可能なソリューションスタックのリストについては、AWS CLI コマンド `aws elasticbeanstalk list-available-solution-stacks` を使用します。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "sampleApplication": {
      "Type": "AWS::ElasticBeanstalk::Application",
      "Properties": {
        "Description": "AWS Elastic Beanstalk Sample Application"
      }
    },
    "sampleApplicationVersion": {
      "Type": "AWS::ElasticBeanstalk::ApplicationVersion",
      "Properties": {
        "ApplicationName": {
          "Ref": "sampleApplication"
        },
        "Description": "AWS ElasticBeanstalk Sample Application Version",
        "SourceBundle": {
          "S3Bucket": {
            "Fn::Sub": "elasticbeanstalk-samples-${AWS::Region}"
          },
          "S3Key": "php-newsample-app.zip"
        }
      }
    },
    "sampleConfigurationTemplate": {
      "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
      "Properties": {
        "ApplicationName": {
          "Ref": "sampleApplication"
        },
        "Description": "AWS ElasticBeanstalk Sample Configuration Template",
        "OptionSettings": [
          {
            "Namespace": "aws:autoscaling:asg",
            "OptionName": "MinSize",
            "Value": "2"
          },
          {
            "Namespace": "aws:autoscaling:asg",
            "OptionName": "MaxSize",
            "Value": "6"
          }
        ]
      }
    }
  }
}
```

```

        {
            "Namespace": "aws:elasticbeanstalk:environment",
            "OptionName": "EnvironmentType",
            "Value": "LoadBalanced"
        },
        {
            "Namespace": "aws:autoscaling:launchconfiguration",
            "OptionName": "IamInstanceProfile",
            "Value": {
                "Ref": "MyInstanceProfile"
            }
        }
    ],
    "SolutionStackName": "solution-stack"
}
},
"sampleEnvironment": {
    "Type": "AWS::ElasticBeanstalk::Environment",
    "Properties": {
        "ApplicationName": {
            "Ref": "sampleApplication"
        },
        "Description": "AWS ElasticBeanstalk Sample Environment",
        "TemplateName": {
            "Ref": "sampleConfigurationTemplate"
        },
        "VersionLabel": {
            "Ref": "sampleApplicationVersion"
        }
    }
},
"MyInstanceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "ec2.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
},

```



```
S3Key: php-newsample-app.zip
sampleConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName:
      Ref: sampleApplication
    Description: AWS ElasticBeanstalk Sample Configuration Template
    OptionSettings:
      - Namespace: aws:autoscaling:asg
        OptionName: MinSize
        Value: '2'
      - Namespace: aws:autoscaling:asg
        OptionName: MaxSize
        Value: '6'
      - Namespace: aws:elasticbeanstalk:environment
        OptionName: EnvironmentType
        Value: LoadBalanced
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref MyInstanceProfile
    SolutionStackName: solution-stack
sampleEnvironment:
  Type: AWS::ElasticBeanstalk::Environment
  Properties:
    ApplicationName:
      Ref: sampleApplication
    Description: AWS ElasticBeanstalk Sample Environment
    TemplateName:
      Ref: sampleConfigurationTemplate
    VersionLabel:
      Ref: sampleApplicationVersion
MyInstanceRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - sts:AssumeRole
    Description: Beanstalk EC2 role
```



```
ManagedPolicyArns:
  - arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier
  - arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker
  - arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier
MyInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Roles:
      - !Ref MyInstanceRole
```

Elastic Load Balancing テンプレートスニペット

Application Load Balancer、Network Load Balancer、または Gateway Load Balancer を作成するには、AWS::ElasticLoadBalancingV2 で始まる V2 リソースタイプを使用します。Classic Load Balancer を作成するには、AWS::ElasticLoadBalancing で始まるリソースタイプを使用します。

内容

- [ELBv2 リソース](#)
- [Classic Load Balancer リソース](#)

ELBv2 リソース

この例では、HTTP リスナーと、トラフィックをターゲットグループに転送するデフォルトのアクションを使用して Application Load Balancer を定義します。ロードバランサーはデフォルトのヘルスチェック設定を使用します。ターゲットグループには 2 つの登録済み EC2 インスタンスがあります。

YAML

```
Resources:
  myLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
```

```
SecurityGroups:
  - !Ref mySecurityGroup

myHTTPListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    LoadBalancerArn: !Ref myLoadBalancer
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - Type: "forward"
        TargetGroupArn: !Ref myTargetGroup

myTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: "my-target-group"
    Protocol: HTTP
    Port: 80
    TargetType: instance
    VpcId: !Ref myVPC
    Targets:
      - Id: !GetAtt Instance1.InstanceId
        Port: 80
      - Id: !GetAtt Instance2.InstanceId
        Port: 80
```

JSON

```
{
  "Resources": {
    "myLoadBalancer": {
      "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties": {
        "Name": "my-alb",
        "Type": "application",
        "Scheme": "internal",
        "Subnets": [
          {
            "Ref": "subnet-AZ1"
          },
          {
            "Ref": "subnet-AZ2"
          }
        ]
      }
    }
  }
}
```

```
        }
      ],
      "SecurityGroups": [
        {
          "Ref": "mySecurityGroup"
        }
      ]
    }
  },
  "myHTTPListener": {
    "Type": "AWS::ElasticLoadBalancingV2::Listener",
    "Properties": {
      "LoadBalancerArn": {
        "Ref": "myLoadBalancer"
      },
      "Protocol": "HTTP",
      "Port": 80,
      "DefaultActions": [
        {
          "Type": "forward",
          "TargetGroupArn": {
            "Ref": "myTargetGroup"
          }
        }
      ]
    }
  },
  "myTargetGroup": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
      "Name": "my-target-group",
      "Protocol": "HTTP",
      "Port": 80,
      "TargetType": "instance",
      "VpcId": {
        "Ref": "myVPC"
      },
      "Targets": [
        {
          "Id": {
            "Fn::GetAtt": [
              "Instance1",
              "InstanceId"
            ]
          }
        }
      ]
    }
  }
}
```

```

        },
        "Port": 80
      },
      {
        "Id": {
          "Fn::GetAtt": [
            "Instance2",
            "InstanceId"
          ]
        },
        "Port": 80
      }
    ]
  }
}

```

Classic Load Balancer リソース

この例では、HTTP リスナーを持ち、登録済み EC2 インスタンスを持たない Classic Load Balancer を定義します。ロードバランサーはデフォルトのヘルスチェック設定を使用します。

YAML

```

myLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
      - "us-east-1a"
    Listeners:
      - LoadBalancerPort: '80'
        InstancePort: '80'
        Protocol: HTTP

```

JSON

```

"myLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "AvailabilityZones" : [ "us-east-1a" ],
    "Listeners" : [ {

```

```
        "LoadBalancerPort" : "80",
        "InstancePort" : "80",
        "Protocol" : "HTTP"
    } ]
}
}
```

この例では、HTTP リスナー、2 つの登録済み EC2 インスタンス、カスタムヘルスチェック設定を使用して Classic Load Balancer を定義します。

YAML

```
myClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
      - "us-east-1a"
    Instances:
      - Ref: Instance1
      - Ref: Instance2
    Listeners:
      - LoadBalancerPort: '80'
        InstancePort: '80'
        Protocol: HTTP
    HealthCheck:
      Target: HTTP:80/
      HealthyThreshold: '3'
      UnhealthyThreshold: '5'
      Interval: '30'
      Timeout: '5'
```

JSON

```
"myClassicLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "AvailabilityZones" : [ "us-east-1a" ],
    "Instances" : [
      { "Ref" : "Instance1" },
      { "Ref" : "Instance2" }
    ],
    "Listeners" : [ {
```

```
        "LoadBalancerPort" : "80",
        "InstancePort" : "80",
        "Protocol" : "HTTP"
    } ],

    "HealthCheck" : {
        "Target" : "HTTP:80/",
        "HealthyThreshold" : "3",
        "UnhealthyThreshold" : "5",
        "Interval" : "30",
        "Timeout" : "5"
    }
}
}
```

AWS Identity and Access Management テンプレートスニペット

このセクションでは、AWS Identity and Access Management のテンプレートスニペットを紹介しています。

トピック

- [IAM ユーザーリソースの宣言](#)
- [IAM アクセスキーリソースの宣言](#)
- [IAM グループ リソースの宣言](#)
- [グループへのユーザーの追加](#)
- [IAM ポリシーの宣言](#)
- [Amazon S3 バケットポリシーの宣言](#)
- [Amazon SNS トピックポリシーの宣言](#)
- [Amazon SQS ポリシーの宣言](#)
- [IAM ロールのテンプレートの例](#)

Important

IAM リソースを含んだテンプレートを使ってスタックを作成または更新する際は、IAM の機能の使用に同意する必要があります。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。

IAM ユーザーリソースの宣言

このスニペットは、IAM ユーザーを作成するための [AWS::IAM::User](#) リソースの宣言を示しています。ユーザーの宣言には、パス ("/") と、パスワード (myP@ssW0rd) のあるログインプロファイルとを使用しています。

giveaccesstoqueueonly という名前のポリシードキュメントは、Amazon SQS キューリソース myqueue に対するすべての Amazon SQS アクションの実行権限をユーザーに付与するとともに、他のすべての Amazon SQS キューリソースに対するアクセスを拒否します。Fn::GetAtt 関数は、[AWS::SQS::Queue](#) リソース myqueue の Arn 属性を取得します。

giveaccesstotopiconly という名前のポリシードキュメントをユーザーに追加し、Amazon SNS トピックリソース mytopic に対するすべての Amazon SNS アクションの実行権限をユーザーに付与するとともに、他のすべての Amazon SNS リソースへのアクセスを拒否します。Ref 関数は [AWS::SNS::Topic](#) リソース mytopic の ARN を取得します。

JSON

```
"myuser" : {
  "Type" : "AWS::IAM::User",
  "Properties" : {
    "Path" : "/",
    "LoginProfile" : {
      "Password" : "myP@ssW0rd"
    },
    "Policies" : [ {
      "PolicyName" : "giveaccesstoqueueonly",
      "PolicyDocument" : {
        "Version": "2012-10-17",
        "Statement" : [ {
          "Effect" : "Allow",
          "Action" : [ "sqs:*" ],
          "Resource" : [ {
            "Fn::GetAtt" : [ "myqueue", "Arn" ]
          } ]
        }, {
          "Effect" : "Deny",
          "Action" : [ "sqs:*" ],
          "NotResource" : [ {
            "Fn::GetAtt" : [ "myqueue", "Arn" ]
          } ]
        } ]
      }
    } ]
  }
}
```

```

    ] }
  }, {
    "PolicyName" : "giveaccesstotopiconly",
    "PolicyDocument" : {
      "Version": "2012-10-17",
      "Statement" : [ {
        "Effect" : "Allow",
        "Action" : [ "sns:*" ],
        "Resource" : [ { "Ref" : "mytopic" } ]
      }, {
        "Effect" : "Deny",
        "Action" : [ "sns:*" ],
        "NotResource" : [ { "Ref" : "mytopic" } ]
      } ]
    }
  } ]
}
}
}

```

YAML

```

myuser:
  Type: AWS::IAM::User
  Properties:
    Path: "/"
    LoginProfile:
      Password: myP@ssW0rd
    Policies:
      - PolicyName: giveaccesstoqueueonly
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - sqs:*
              Resource:
                - !GetAtt myqueue.Arn
            - Effect: Deny
              Action:
                - sqs:*
              NotResource:
                - !GetAtt myqueue.Arn
      - PolicyName: giveaccesstotopiconly

```



```
PolicyDocument:
  Version: '2012-10-17'
  Statement:
    - Effect: Allow
      Action:
        - sns:*
      Resource:
        - !Ref mytopic
    - Effect: Deny
      Action:
        - sns:*
      NotResource:
        - !Ref mytopic
```

IAM アクセスキーリソースの宣言

このスニペットは、[AWS::IAM::AccessKey](#) リソースを示します。myaccesskey リソースは、アクセスキーを作成し、テンプレートに [AWS::IAM::User](#) リソースとして宣言されている IAM ユーザーにそれを割り当てます。

JSON

```
"myaccesskey" : {
  "Type" : "AWS::IAM::AccessKey",
  "Properties" : {
    "UserName" : { "Ref" : "myuser" }
  }
}
```

YAML

```
myaccesskey:
  Type: AWS::IAM::AccessKey
  Properties:
    UserName:
      !Ref myuser
```

`AWS::IAM::AccessKey` リソースのシークレットキーは、`Fn::GetAtt` 関数を使用して取得できます。シークレットキーを取得する 1 つの方法は、それを Output 値に追加することです。Ref 関数

を使用してアクセスキーを取得することができます。次の Output 値の宣言は、myaccesskey のアクセスキーとシークレットキーを取得します。

JSON

```
"AccessKeyformyaccesskey" : {
  "Value" : { "Ref" : "myaccesskey" }
},
"SecretKeyformyaccesskey" : {
  "Value" : {
    "Fn::GetAtt" : [ "myaccesskey", "SecretAccessKey" ]
  }
}
```

YAML

```
AccessKeyformyaccesskey:
  Value:
    !Ref myaccesskey
SecretKeyformyaccesskey:
  Value: !GetAtt myaccesskey.SecretAccessKey
```

AWS のアクセスキーとシークレットキーを、テンプレートに定義されている Amazon EC2 インスタンスまたは Auto Scaling グループに渡すこともできます。次の [AWS::EC2::Instance](#) 宣言では、UserData プロパティを使用して、myaccesskey リソースのアクセスキーとシークレットキーを渡しています。

JSON

```
"myinstance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "AvailabilityZone" : "us-east-1a",
    "ImageId" : "ami-0ff8a91507f77f867",
    "UserData" : {
      "Fn::Base64" : {
        "Fn::Join" : [
          "", [
            "ACCESS_KEY=", {
              "Ref" : "myaccesskey"
            }
          ]
        ]
      }
    }
  }
}
```



```

    "Type" : "AWS::IAM::Group",
    "Properties" : {
      "Path" : "/myapplication/",
      "Policies" : [ {
        "PolicyName" : "myapppolicy",
        "PolicyDocument" : {
          "Version": "2012-10-17",
          "Statement" : [ {
            "Effect" : "Allow",
            "Action" : [ "sqs:*" ],
            "Resource" : [ {
              "Fn::GetAtt" : [ "myqueue", "Arn" ]
            } ]
          },
          {
            "Effect" : "Deny",
            "Action" : [ "sqs:*" ],
            "NotResource" : [ { "Fn::GetAtt" : [ "myqueue", "Arn" ] } ]
          }
        ]
      } ]
    }
  }
}

```

YAML

```

mygroup:
  Type: AWS::IAM::Group
  Properties:
    Path: "/myapplication/"
    Policies:
      - PolicyName: myapppolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - sqs:*
              Resource: !GetAtt myqueue.Arn
            - Effect: Deny
              Action:
                - sqs:*
              NotResource: !GetAtt myqueue.Arn

```

グループへのユーザーの追加

グループには、[AWS::IAM::UserToGroupAddition](#) リソースでユーザーを追加します。次のスニペットで、addUserToGroup リソースは、myexistinggroup2 という名前の既存のグループに対し、既存のユーザー existinguser1 と、テンプレートに [AWS::IAM::User](#) リソースとして宣言されているユーザー myuser を追加しています。

JSON

```
"addUserToGroup" : {
  "Type" : "AWS::IAM::UserToGroupAddition",
  "Properties" : {
    "GroupName" : "myexistinggroup2",
    "Users" : [ "existinguser1", { "Ref" : "myuser" } ]
  }
}
```

YAML

```
addUserToGroup:
  Type: AWS::IAM::UserToGroupAddition
  Properties:
    GroupName: myexistinggroup2
    Users:
      - existinguser1
      - !Ref myuser
```

IAM ポリシーの宣言

このスニペットでは、ポリシーを作成し、mypolicy という名前の [AWS::IAM::Policy](#) リソースを使用して、複数のグループにそのポリシーを適用しています。mypolicy リソースには、ARN PolicyDocument で表される S3 バケット内のオブジェクトに対し、GetObject、PutObject、PutObjectAcl の各アクションを許可する arn:aws:s3:::myAWSBucket プロパティが存在します。mypolicy リソースのポリシーの適用先は、myexistinggroup1 という名前の既存のグループと、テンプレート内で [AWS::IAM::Group](#) リソースとして宣言されているグループ mygroup です。この例では、Groups プロパティを使用してグループにポリシーを適用する方法を示していますが、Users プロパティを使用して、ポリシードキュメントを一連のユーザーに追加することもできます。

JSON

```
"mypolicy" : {
  "Type" : "AWS::IAM::Policy",
  "Properties" : {
    "PolicyName" : "mygrouppolicy",
    "PolicyDocument" : {
      "Version": "2012-10-17",
      "Statement" : [ {
        "Effect" : "Allow",
        "Action" : [
          "s3:GetObject" , "s3:PutObject" , "s3:PutObjectAcl" ],
        "Resource" : "arn:aws:s3:::myAWSBucket/*"
      } ]
    },
    "Groups" : [ "myexistinggroup1", { "Ref" : "mygroup" } ]
  }
}
```

YAML

```
mypolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: mygrouppolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:PutObject
            - s3:PutObjectAcl
          Resource: arn:aws:s3:::myAWSBucket/*
    Groups:
      - myexistinggroup1
      - !Ref mygroup
```

Amazon S3 バケットポリシーの宣言

このスニペットでは、ポリシーを作成し、[AWS::S3::BucketPolicy](#) リソースを使用して Amazon S3 バケットにそのポリシーを適用しています。このポリシーの適用先となる S3 バケットの

すべてのオブジェクトに対する GetObject アクションを user1 IAM ユーザーに許可するポリシードキュメントが mybucketpolicy リソースで宣言されています。このスニペットでは、Fn::GetAtt リソースの ARN を user1 関数で取得しています。mybucketpolicy リソースは、AWS::S3::BucketPolicy リソース mybucket にポリシーを適用します。リソースのバケット名は、Refmybucket 関数で取得しています。

JSON

```
"mybucketpolicy" : {
  "Type" : "AWS::S3::BucketPolicy",
  "Properties" : {
    "PolicyDocument" : {
      "Id" : "MyPolicy",
      "Version": "2012-10-17",
      "Statement" : [ {
        "Sid" : "ReadAccess",
        "Action" : [ "s3:GetObject" ],
        "Effect" : "Allow",
        "Resource" : { "Fn::Join" : [
          "", [ "arn:aws:s3:::", { "Ref" : "mybucket" } , "/*" ]
        ] },
        "Principal" : {
          "AWS" : { "Fn::GetAtt" : [ "user1", "Arn" ] }
        }
      } ]
    },
    "Bucket" : { "Ref" : "mybucket" }
  }
}
```

YAML

```
mybucketpolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    PolicyDocument:
      Id: MyPolicy
      Version: '2012-10-17'
      Statement:
        - Sid: ReadAccess
          Action:
            - s3:GetObject
```

```
Effect: Allow
Resource: !Sub "arn:aws:s3:::${mybucket}/*"
Principal:
  AWS: !GetAtt user1.Arn
Bucket: !Ref mybucket
```

Amazon SNS トピックポリシーの宣言

このスニペットでは、ポリシーを作成し、[AWS::SNS::TopicPolicy](#) リソースを使用して Amazon SNS トピックにそのポリシーを適用しています。mysnspolicy リソースに含まれている PolicyDocument プロパティでは、[AWS::IAM::User](#) リソース myuser に対して Publish アクションを [AWS::SNS::Topic](#) リソース mytopic で実行することを許可します。スニペットでは、Fn::GetAtt 関数で myuser リソースの ARN を取得し、Ref 関数で mytopic リソースの ARN を取得します。

JSON

```
"mysnspolicy" : {
  "Type" : "AWS::SNS::TopicPolicy",
  "Properties" : {
    "PolicyDocument" : {
      "Id" : "MyTopicPolicy",
      "Version": "2012-10-17",
      "Statement" : [ {
        "Sid" : "My-statement-id",
        "Effect" : "Allow",
        "Principal" : {
          "AWS" : { "Fn::GetAtt" : [ "myuser", "Arn" ] }
        },
        "Action" : "sns:Publish",
        "Resource" : "*"
      } ]
    },
    "Topics" : [ { "Ref" : "mytopic" } ]
  }
}
```

YAML

```
mysnspolicy:
  Type: AWS::SNS::TopicPolicy
  Properties:
```



```
PolicyDocument:
  Id: MyTopicPolicy
  Version: '2012-10-17'
  Statement:
    - Sid: My-statement-id
      Effect: Allow
      Principal:
        AWS: !GetAtt myuser.Arn
      Action: sns:Publish
      Resource: "*"
Topics:
  - !Ref mytopic
```

Amazon SQS ポリシーの宣言

このスニペットでは、ポリシーを作成し、[AWS::SQS::QueuePolicy](#) リソースを使用して Amazon SQS キューにそのポリシーを適用しています。PolicyDocument プロパティは、既存のキュー (URL で指定) および [AWS::SQS::Queue](#) リソース myqueue に対する SendMessage アクションを既存のユーザー myapp (ARN で指定) に許可します。リソースの URL は [Ref](#) 関数で取得しています。myqueue

JSON

```
"mysqspolicy" : {
  "Type" : "AWS::SQS::QueuePolicy",
  "Properties" : {
    "PolicyDocument" : {
      "Id" : "MyQueuePolicy",
      "Version": "2012-10-17",
      "Statement" : [ {
        "Sid" : "Allow-User-SendMessage",
        "Effect" : "Allow",
        "Principal" : {
          "AWS" : "arn:aws:iam::123456789012:user/myapp"
        },
        "Action" : [ "sqs:SendMessage" ],
        "Resource" : "*"
      } ]
    },
    "Queues" : [
      "https://sqs.us-east-2aws-region.amazonaws.com/123456789012/myexistingqueue",
      { "Ref" : "myqueue" }
    ]
  }
}
```

```
}  
}
```

YAML

```
mysqspolicy:  
  Type: AWS::SQS::QueuePolicy  
  Properties:  
    PolicyDocument:  
      Id: MyQueuePolicy  
      Version: '2012-10-17'  
      Statement:  
        - Sid: Allow-User-SendMessage  
          Effect: Allow  
          Principal:  
            AWS: arn:aws:iam::123456789012:user/myapp  
          Action:  
            - sqs:SendMessage  
          Resource: "*"br/>    Queues:  
      - https://sqs.aws-region.amazonaws.com/123456789012/myexistingqueue  
      - !Ref myqueue
```

IAM ロールのテンプレートの例

このセクションでは、CloudFormation テンプレートで EC2 インスタンスの IAM ロールを使用する例を紹介しています。

詳細については、[Amazon EC2 ユーザーガイド](#) の「Amazon EC2 の IAM ロール」を参照してください。

IAM ロールと EC2

この例では、インスタンスプロファイルは EC2 インスタンスの `IamInstanceProfile` プロパティによって参照されます。インスタンスポリシーとロールポリシーのどちらも、[AWS::IAM::Role](#) を参照します。

JSON

```
{  
  "AWSTemplateFormatVersion": "2010-09-09",  
  "Resources": {  
    "myEC2Instance": {
```

```
"Type": "AWS::EC2::Instance",
"Version": "2009-05-15",
"Properties": {
  "ImageId": "ami-0ff8a91507f77f867",
  "InstanceType": "m1.small",
  "Monitoring": "true",
  "DisableApiTermination": "false",
  "IamInstanceProfile": {
    "Ref": "RootInstanceProfile"
  }
},
"RootRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [ {
        "Effect": "Allow",
        "Principal": {
          "Service": [ "ec2.amazonaws.com" ]
        },
        "Action": [ "sts:AssumeRole" ]
      } ]
    },
    "Path": "/"
  }
},
"RolePolicies": {
  "Type": "AWS::IAM::Policy",
  "Properties": {
    "PolicyName": "root",
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [ {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
      } ]
    },
    "Roles": [ { "Ref": "RootRole" } ]
  }
},
"RootInstanceProfile": {
```

```

        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
        }
    }
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myEC2Instance:
    Type: AWS::EC2::Instance
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      Monitoring: 'true'
      DisableApiTermination: 'false'
      IamInstanceProfile:
        !Ref RootInstanceProfile
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:

```

```
- Effect: Allow
  Action: "*"
  Resource: "*"
Roles:
- !Ref RootRole
RootInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: "/"
    Roles:
    - !Ref RootRole
```

IAM ロールと Auto Scaling グループ

この例では、インスタンスプロファイルは、Amazon EC2 Auto Scaling の起動構成の `IamInstanceProfile` プロパティによって参照されます。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "myLCOne": {
      "Type": "AWS::AutoScaling::LaunchConfiguration",
      "Version": "2009-05-15",
      "Properties": {
        "ImageId": "ami-0ff8a91507f77f867",
        "InstanceType": "m1.small",
        "InstanceMonitoring": "true",
        "IamInstanceProfile": { "Ref": "RootInstanceProfile" }
      }
    },
    "myASGrpOne": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "Version": "2009-05-15",
      "Properties": {
        "AvailabilityZones": [ "us-east-1a" ],
        "LaunchConfigurationName": { "Ref": "myLCOne" },
        "MinSize": "0",
        "MaxSize": "0",
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": "120"
      }
    }
  }
}
```

```
    },
    "RootRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [ {
            "Effect": "Allow",
            "Principal": {
              "Service": [ "ec2.amazonaws.com" ]
            },
            "Action": [ "sts:AssumeRole" ]
          } ]
        },
        "Path": "/"
      }
    },
    "RolePolicies": {
      "Type": "AWS::IAM::Policy",
      "Properties": {
        "PolicyName": "root",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [ {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
          } ]
        },
        "Roles": [ { "Ref": "RootRole" } ]
      }
    },
    "RootInstanceProfile": {
      "Type": "AWS::IAM::InstanceProfile",
      "Properties": {
        "Path": "/",
        "Roles": [ { "Ref": "RootRole" } ]
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myLCOne:
    Type: AWS::AutoScaling::LaunchConfiguration
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      InstanceMonitoring: 'true'
      IamInstanceProfile:
        !Ref RootInstanceProfile
  myASGrpOne:
    Type: AWS::AutoScaling::AutoScalingGroup
    Version: '2009-05-15'
    Properties:
      AvailabilityZones:
        - "us-east-1a"
      LaunchConfigurationName:
        !Ref myLCOne
      MinSize: '0'
      MaxSize: '0'
      HealthCheckType: EC2
      HealthCheckGracePeriod: '120'
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
```

```
Statement:
  - Effect: Allow
    Action: "*"
    Resource: "*"
Roles:
  - !Ref RootRole
RootInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: "/"
    Roles:
      - !Ref RootRole
```

AWS Lambda テンプレート

以下のテンプレートは、AWS Lambda (Lambda) 関数とカスタムのリソースを使用して、既存のセキュリティグループのリストに新しいセキュリティグループを追加します。この関数は、新規および既存のセキュリティグループが両方含まれるリストを作成できるように、セキュリティグループのリストを動的に構築する場合に便利です。たとえば、パラメーター値として既存のセキュリティグループのリストを渡し、新しい値をリストに追加して、すべての値を EC2 インスタンスに関連付けることができます。Lambda 関数のリソース タイプの詳細については、「[AWS::Lambda::Function](#)」を参照してください。

次の例に示した CloudFormation は、AllSecurityGroups カスタムリソースを作成するときに、まず AppendItemToListFunction Lambda 関数を呼び出しています。次にその関数に、既存のセキュリティグループと新しいセキュリティグループ (NewSecurityGroup) のリストを渡します。関数は、新しいセキュリティグループをリストに追加し、最新のリストを返します。CloudFormation は、最新のリストを使用して、すべてのセキュリティグループを MyEC2Instance リソースに関連付けます。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "ExistingSecurityGroups": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>"
    },
    "ExistingVPC": {
      "Type": "AWS::EC2::VPC::Id",
```



```
    "Description": "The VPC ID that includes the security groups in the
ExistingSecurityGroups parameter."
  },
  "InstanceType": {
    "Type": "String",
    "Default": "t2.micro",
    "AllowedValues": [
      "t2.micro",
      "t3.micro"
    ]
  }
},

"Resources": {
  "SecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Allow HTTP traffic to the host",
      "VpcId": {
        "Ref": "ExistingVPC"
      },
    },
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "CidrIp": "0.0.0.0/0"
      }
    ],
    "SecurityGroupEgress": [
      {
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "CidrIp": "0.0.0.0/0"
      }
    ]
  },
  "AllSecurityGroups": {
    "Type": "Custom::Split",
    "Properties": {
      "ServiceToken": {
        "Fn::GetAtt": [
```

```

        "AppendItemToListFunction",
        "Arn"
    ]
},
"List": {
    "Ref": "ExistingSecurityGroups"
},
"AppendedItem": {
    "Ref": "SecurityGroup"
}
}
},
"AppendItemToListFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
        "Handler": "index.handler",
        "Role": {
            "Fn::GetAtt": [
                "LambdaExecutionRole",
                "Arn"
            ]
        },
        "Code": {
            "ZipFile": {
                "Fn::Join": [
                    "",
                    [
                        "var response = require('cfn-response');",
                        "exports.handler = function(event, context) {",
                        "    var responseData = {Value:",
event.ResourceProperties.List};",
                        "    responseData.Value.push(event.ResourceProperties.AppendedItem);",
                        "    response.send(event, context, response.SUCCESS,",
responseData);",
                        "};"
                    ]
                ]
            }
        },
        "Runtime": "nodejs20.x"
    }
},
"MyEC2Instance": {

```

```

        "Type": "AWS::EC2::Instance",
        "Properties": {
            "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}",
            "SecurityGroupIds": {
                "Fn::GetAtt": [
                    "AllSecurityGroups",
                    "Value"
                ]
            },
            "InstanceType": {
                "Ref": "InstanceType"
            }
        }
    },
    "LambdaExecutionRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "lambda.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "root",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [

```

YAML

API バージョン 2010-05-15 498

```

    - t2.micro
    - t3.micro
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP traffic to the host
      VpcId: !Ref ExistingVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
  AllSecurityGroups:
    Type: Custom::Split
    Properties:
      ServiceToken: !GetAtt AppendItemToListFunction.Arn
      List: !Ref ExistingSecurityGroups
      AppendedItem: !Ref SecurityGroup
  AppendItemToListFunction:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: !Join
          - ''
          - - var response = require('cfn-response');
            - exports.handler = function(event, context) {
              - '   var responseData = {Value: event.ResourceProperties.List};'
              - '   responseData.Value.push(event.ResourceProperties.AppendedItem);'
              - '   response.send(event, context, response.SUCCESS, responseData);'
              - '};'
            - ''
      Runtime: nodejs20.x
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'

```

```
SecurityGroupIds: !GetAtt AllSecurityGroups.Value
InstanceType: !Ref InstanceType
LambdaExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - lambda.amazonaws.com
          Action:
            - sts:AssumeRole
    Path: /
  Policies:
    - PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - logs:*
            Resource: arn:aws:logs:*:*:*
Outputs:
  AllSecurityGroups:
    Description: Security Groups that are associated with the EC2 instance
    Value: !Join
      - ', '
      - !GetAtt AllSecurityGroups.Value
```

Amazon Redshift テンプレートスニペット

Amazon Redshift は、クラウド内でのフルマネージド型、ペタバイトスケールのデータウェアハウスサービスです。CloudFormation を使用して、Amazon Redshift クラスターをプロビジョニングおよび管理できます。

Amazon Redshift クラスター

以下のサンプルテンプレートは、スタックの作成時に指定されたパラメーター値に従って Amazon Redshift クラスターを作成するものです。Amazon Redshift クラスターに関連付けられたクラスターパラメーターグループによって、ユーザーアクティビティログを記録することができます。また、こ

のテンプレートは、テンプレートに定義されている Amazon VPC 内で Amazon Redshift クラスターを起動します。インターネットから Amazon Redshift クラスターにアクセスできるように、VPC にはインターネットゲートウェイが追加されています。ただし、クラスターとインターネットゲートウェイ間の通信も有効にする必要があります。これは、ルートテーブルエントリで行っています。

 Note

テンプレートには、IsMultiNodeCluster 条件が追加されており、NumberOfNodes パラメーターの値が ClusterType に設定されたときにだけ multi-node パラメーターが宣言されます。

この例では、MysqlRootPassword プロパティを NoEcho に設定して true パラメータを定義しています。NoEcho 属性を true に設定すると、CloudFormation は、スタックまたはスタックイベントを記述するすべての呼び出しに対して、アスタリスク (****) としてマスクされたパラメータ値を返します。ただし、以下に指定された場所に保存されている情報は除きます。

 Important

NoEcho 属性を使用しても、以下に保存されている情報はマスクされません。

- Metadata テンプレートセクション。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。詳細については、「[メタデータ](#)」を参照してください。
- Outputs テンプレートセクション。詳細については、「[出力](#)」を参照してください。
- リソース定義の Metadata 属性。詳細については、「[Metadata 属性](#)」を参照してください。

パスワードやシークレットなどの機密情報を含めるには、これらのメカニズムを使用しないことを強くお勧めします。

 Important

機密情報は、CloudFormation テンプレートに直接埋め込むのではなく、スタックテンプレートの動的パラメータを使用して CloudFormation の外部 (AWS Systems Manager パラメータ

ストアや AWS Secrets Manager など) に保存して管理した上で 参照することをお勧めします。

詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティスを参照してください。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "DatabaseName" : {
      "Description" : "The name of the first database to be created when the cluster is
created",
      "Type" : "String",
      "Default" : "dev",
      "AllowedPattern" : "([a-z]|[0-9])+"
    },
    "ClusterType" : {
      "Description" : "The type of cluster",
      "Type" : "String",
      "Default" : "single-node",
      "AllowedValues" : [ "single-node", "multi-node" ]
    },
    "NumberOfNodes" : {
      "Description" : "The number of compute nodes in the cluster. For multi-node
clusters, the NumberOfNodes parameter must be greater than 1",
      "Type" : "Number",
      "Default" : "1"
    },
    "NodeType" : {
      "Description" : "The type of node to be provisioned",
      "Type" : "String",
      "Default" : "ds2.xlarge",
      "AllowedValues" : [ "ds2.xlarge", "ds2.8xlarge", "dc1.large", "dc1.8xlarge" ]
    },
    "MasterUsername" : {
      "Description" : "The user name that is associated with the master user account
for the cluster that is being created",
      "Type" : "String",
      "Default" : "defaultuser",
      "AllowedPattern" : "([a-z])([a-z]|[0-9])*"
    }
  }
}
```



```

    },
    "MasterUserPassword" : {
      "Description" : "The password that is associated with the master user account for
the cluster that is being created.",
      "Type" : "String",
      "NoEcho" : "true"
    },
    "InboundTraffic" : {
      "Description" : "Allow inbound traffic to the cluster from this CIDR range.",
      "Type" : "String",
      "MinLength" : "9",
      "MaxLength" : "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" : "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3}) / (\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },
    "PortNumber" : {
      "Description" : "The port number on which the cluster accepts incoming
connections.",
      "Type" : "Number",
      "Default" : "5439"
    }
  },
  "Conditions" : {
    "IsMultiNodeCluster" : {
      "Fn::Equals" : [ { "Ref" : "ClusterType" }, "multi-node" ]
    }
  },
  "Resources" : {
    "RedshiftCluster" : {
      "Type" : "AWS::Redshift::Cluster",
      "DependsOn" : "AttachGateway",
      "Properties" : {
        "ClusterType" : { "Ref" : "ClusterType" },
        "NumberOfNodes" : { "Fn::If" : [ "IsMultiNodeCluster", { "Ref" :
"NumberOfNodes" }, { "Ref" : "AWS::NoValue" } ] },
        "NodeType" : { "Ref" : "NodeType" },
        "DBName" : { "Ref" : "DatabaseName" },
        "MasterUsername" : { "Ref" : "MasterUsername" },
        "MasterUserPassword" : { "Ref" : "MasterUserPassword" },
        "ClusterParameterGroupName" : { "Ref" : "RedshiftClusterParameterGroup" },
        "VpcSecurityGroupIds" : [ { "Ref" : "SecurityGroup" } ],
        "ClusterSubnetGroupName" : { "Ref" : "RedshiftClusterSubnetGroup" },

```

```

        "PubliclyAccessible" : "true",
        "Port" : { "Ref" : "PortNumber" }
    }
},
"RedshiftClusterParameterGroup" : {
    "Type" : "AWS::Redshift::ClusterParameterGroup",
    "Properties" : {
        "Description" : "Cluster parameter group",
        "ParameterGroupFamily" : "redshift-1.0",
        "Parameters" : [{
            "ParameterName" : "enable_user_activity_logging",
            "ParameterValue" : "true"
        }]
    }
},
"RedshiftClusterSubnetGroup" : {
    "Type" : "AWS::Redshift::ClusterSubnetGroup",
    "Properties" : {
        "Description" : "Cluster subnet group",
        "SubnetIds" : [ { "Ref" : "PublicSubnet" } ]
    }
},
"VPC" : {
    "Type" : "AWS::EC2::VPC",
    "Properties" : {
        "CidrBlock" : "10.0.0.0/16"
    }
},
"PublicSubnet" : {
    "Type" : "AWS::EC2::Subnet",
    "Properties" : {
        "CidrBlock" : "10.0.0.0/24",
        "VpcId" : { "Ref" : "VPC" }
    }
},
"SecurityGroup" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
        "GroupDescription" : "Security group",
        "SecurityGroupIngress" : [ {
            "CidrIp" : { "Ref" : "InboundTraffic" },
            "FromPort" : { "Ref" : "PortNumber" },
            "ToPort" : { "Ref" : "PortNumber" },
            "IpProtocol" : "tcp"
        }
    ]
}

```

```
    } ],
    "VpcId" : { "Ref" : "VPC" }
  },
  "myInternetGateway" : {
    "Type" : "AWS::EC2::InternetGateway"
  },
  "AttachGateway" : {
    "Type" : "AWS::EC2::VPCGatewayAttachment",
    "Properties" : {
      "VpcId" : { "Ref" : "VPC" },
      "InternetGatewayId" : { "Ref" : "myInternetGateway" }
    }
  },
  "PublicRouteTable" : {
    "Type" : "AWS::EC2::RouteTable",
    "Properties" : {
      "VpcId" : {
        "Ref" : "VPC"
      }
    }
  },
  "PublicRoute" : {
    "Type" : "AWS::EC2::Route",
    "DependsOn" : "AttachGateway",
    "Properties" : {
      "RouteTableId" : {
        "Ref" : "PublicRouteTable"
      },
      "DestinationCidrBlock" : "0.0.0.0/0",
      "GatewayId" : {
        "Ref" : "myInternetGateway"
      }
    }
  },
  "PublicSubnetRouteTableAssociation" : {
    "Type" : "AWS::EC2::SubnetRouteTableAssociation",
    "Properties" : {
      "SubnetId" : {
        "Ref" : "PublicSubnet"
      },
      "RouteTableId" : {
        "Ref" : "PublicRouteTable"
      }
    }
  }
```

```

    }
  }
},
"Outputs" : {
  "ClusterEndpoint" : {
    "Description" : "Cluster endpoint",
    "Value" : { "Fn::Join" : [ ":", [ { "Fn::GetAtt" : [ "RedshiftCluster",
"Endpoint.Address" ] }, { "Fn::GetAtt" : [ "RedshiftCluster",
"Endpoint.Port" ] } ] ] }
  },
  "ClusterName" : {
    "Description" : "Name of cluster",
    "Value" : { "Ref" : "RedshiftCluster" }
  },
  "ParameterGroupName" : {
    "Description" : "Name of parameter group",
    "Value" : { "Ref" : "RedshiftClusterParameterGroup" }
  },
  "RedshiftClusterSubnetGroupName" : {
    "Description" : "Name of cluster subnet group",
    "Value" : { "Ref" : "RedshiftClusterSubnetGroup" }
  },
  "RedshiftClusterSecurityGroupName" : {
    "Description" : "Name of cluster security group",
    "Value" : { "Ref" : "SecurityGroup" }
  }
}
}
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  DatabaseName:
    Description: The name of the first database to be created when the cluster is
      created
    Type: String
    Default: dev
    AllowedPattern: "([a-z]|[0-9])+"
  ClusterType:
    Description: The type of cluster
    Type: String
    Default: single-node

```

AllowedValues:

- single-node
- multi-node

NumberOfNodes:

Description: The number of compute nodes in the cluster. For multi-node clusters, the NumberOfNodes parameter must be greater than 1

Type: Number

Default: '1'

Node Type:

Description: The type of node to be provisioned

Type: String

Default: ds2.xlarge

AllowedValues:

- ds2.xlarge
- ds2.8xlarge
- dc1.large
- dc1.8xlarge

MasterUsername:

Description: The user name that is associated with the master user account for the cluster that is being created

Type: String

Default: defaultuser

AllowedPattern: "([a-z])([a-z]|[0-9])"

MasterUserPassword:

Description: The password that is associated with the master user account for the cluster that is being created.

Type: String

```
NoEcho: 'true'
```

InboundTraffic:

Description: Allow inbound traffic to the cluster from this CIDR range.

Type: String

MinLength: '9'

MaxLength: '18'

Default: 0.0.0.0/0

AllowedPattern: "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})/(\\d{1,2})"

ConstraintDescription: must be a valid CIDR range of the form x.x.x.x/x.

PortNumber:

Description: The port number on which the cluster accepts incoming connections.

Type: Number

Default: '5439'

Conditions:

IsMultiNodeCluster:

Fn::Equals:

- Ref: ClusterType

- multi-node

Resources:**RedshiftCluster:**

Type: AWS::Redshift::Cluster

DependsOn: AttachGateway

Properties:**ClusterType:**

Ref: ClusterType

NumberOfNodes:**Fn::If:**

- IsMultiNodeCluster
- Ref: NumberOfNodes
- Ref: AWS::NoValue

NodeType:

Ref: NodeType

DBName:

Ref: DatabaseName

MasterUsername:

Ref: MasterUsername

MasterUserPassword:

Ref: MasterUserPassword

ClusterParameterGroupName:

Ref: RedshiftClusterParameterGroup

VpcSecurityGroupIds:

- Ref: SecurityGroup

ClusterSubnetGroupName:

Ref: RedshiftClusterSubnetGroup

PubliclyAccessible: 'true'

Port:

Ref: PortNumber

RedshiftClusterParameterGroup:

Type: AWS::Redshift::ClusterParameterGroup

Properties:

Description: Cluster parameter group

ParameterGroupFamily: redshift-1.0

Parameters:

- ParameterName: enable_user_activity_logging
ParameterValue: 'true'

RedshiftClusterSubnetGroup:

Type: AWS::Redshift::ClusterSubnetGroup

Properties:

Description: Cluster subnet group

SubnetIds:

- Ref: PublicSubnet

```
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: 10.0.0.0/16
PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock: 10.0.0.0/24
    VpcId:
      Ref: VPC
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security group
    SecurityGroupIngress:
      - CidrIp:
          Ref: InboundTraffic
        FromPort:
          Ref: PortNumber
        ToPort:
          Ref: PortNumber
        IpProtocol: tcp
    VpcId:
      Ref: VPC
myInternetGateway:
  Type: AWS::EC2::InternetGateway
AttachGateway:
  Type: AWS::EC2::VPCElasticNetworkAttachment
  Properties:
    VpcId:
      Ref: VPC
    InternetGatewayId:
      Ref: myInternetGateway
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: VPC
PublicRoute:
  Type: AWS::EC2::Route
  DependsOn: AttachGateway
  Properties:
    RouteTableId:
      Ref: PublicRouteTable
```

```
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId:
      Ref: myInternetGateway
  PublicSubnetRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId:
        Ref: PublicSubnet
      RouteTableId:
        Ref: PublicRouteTable
Outputs:
  ClusterEndpoint:
    Description: Cluster endpoint
    Value: !Sub "${RedshiftCluster.Endpoint.Address}:${RedshiftCluster.Endpoint.Port}"
  ClusterName:
    Description: Name of cluster
    Value:
      Ref: RedshiftCluster
  ParameterGroupName:
    Description: Name of parameter group
    Value:
      Ref: RedshiftClusterParameterGroup
  RedshiftClusterSubnetGroupName:
    Description: Name of cluster subnet group
    Value:
      Ref: RedshiftClusterSubnetGroup
  RedshiftClusterSecurityGroupName:
    Description: Name of cluster security group
    Value:
      Ref: SecurityGroup
```

関連情報

[AWS::Redshift::Cluster](#)

Amazon RDS テンプレートスニペット

トピック

- [Amazon RDS DB Instance リソース](#)
- [Amazon RDS Oracle Database DB Instance リソース](#)
- [CIDR 範囲の Amazon RDS DBSecurityGroup リソース](#)

- [Amazon EC2 セキュリティグループを使用した Amazon RDS DBSecurityGroup](#)
- [複数の VPC セキュリティグループ](#)
- [VPC セキュリティグループの Amazon RDS データベースインスタンス](#)

Amazon RDS DB Instance リソース

この例では、マネージドマスターユーザーパスワードを持つ Amazon RDS DB Instance のリソースを示しています。詳細については、Amazon RDS ユーザー ガイドの「[AWS Secrets Manager を使用したパスワード管理](#)」と Aurora ユーザー ガイドの「[AWS Secrets Manager を使用したパスワード管理](#)」を参照してください。EngineVersion プロパティ (オプション) が指定されていないため、この DB インスタンスにはデフォルトのエンジンバージョンが使用されます。デフォルトのエンジンバージョンおよびその他のデフォルト設定の詳細については、「[CreateDBInstance](#)」を参照してください。DBSecurityGroups プロパティでは、MyDbSecurityByEC2SecurityGroup と MyDbSecurityByCIDRIPGroup という AWS::RDS::DBSecurityGroup リソースを指定して、これらのリソースへのネットワーク進入を許可しています。詳細については、「[AWS::RDS::DBInstance](#)」を参照してください。また、この DB インスタンスリソースでは、DeletionPolicy 属性が Snapshot に設定されています。DeletionPolicy が Snapshot に設定されている場合、スタックが削除されると、CloudFormation は DB インスタンスのスナップショットを作成します。

JSON

```
"MyDB" : {
  "Type" : "AWS::RDS::DBInstance",
  "Properties" : {
    "DBSecurityGroups" : [
      {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
"MyDbSecurityByCIDRIPGroup"} ],
    "AllocatedStorage" : "5",
    "DBInstanceClass" : "db.t2.small",
    "Engine" : "MySQL",
    "MasterUsername" : "MyName",
    "ManageMasterUserPassword" : true,
    "MasterUserSecret" : {
      "KmsKeyId" : {"Ref" : "KMSKey"}
    }
  },
  "DeletionPolicy" : "Snapshot"
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
      - Ref: MyDbSecurityByEC2SecurityGroup
      - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    MasterUsername: MyName
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
    DeletionPolicy: Snapshot
```

Amazon RDS Oracle Database DB Instance リソース

この例では、マネージドマスターユーザーパスワードを使用して Oracle データベース (DB) インスタンスリソースを作成します。詳細については、Amazon RDS ユーザー ガイドの「[AWS Secrets Manager を使用したパスワード管理](#)」を参照してください。この例では、Engine を bring-your-own-license のライセンスモデルを持つ oracle-ee として指定しています。Oracle Database DB Instance の設定の詳細については、「[CreateDBInstance](#)」を参照してください。MyDbSecurityByEC2SecurityGroup および MyDbSecurityByCIDRIPGroup という名前の AWS::RDS::DBSecurityGroup リソースに対し、DBSecurityGroups プロパティでネットワーク進入許可を与えています。詳細については、「[AWS::RDS::DBInstance](#)」を参照してください。また、この DB インスタンスリソースでは、DeletionPolicy 属性が Snapshot に設定されています。DeletionPolicy が Snapshot に設定されている場合、スタックが削除されると、CloudFormation は DB インスタンスのスナップショットを作成します。

JSON

```
"MyDB" : {
  "Type" : "AWS::RDS::DBInstance",
  "Properties" : {
    "DBSecurityGroups" : [
      {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
"MyDbSecurityByCIDRIPGroup"} ],
    "AllocatedStorage" : "5",
    "DBInstanceClass" : "db.t2.small",
```

```

    "Engine" : "oracle-ee",
    "LicenseModel" : "bring-your-own-license",
    "MasterUsername" : "master",
    "ManageMasterUserPassword" : true,
    "MasterUserSecret" : {
        "KmsKeyId" : {"Ref" : "KMSKey"}
    }
},
"DeletionPolicy" : "Snapshot"
}

```

YAML

```

MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
      - Ref: MyDbSecurityByEC2SecurityGroup
      - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: oracle-ee
    LicenseModel: bring-your-own-license
    MasterUsername: master
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
    DeletionPolicy: Snapshot

```

CIDR 範囲の Amazon RDS DBSecurityGroup リソース

この例では、ddd.ddd.ddd.ddd/dd 形式で指定された CIDR 範囲に対して Amazon RDS DBSecurityGroup リソースへの進入を許可しています。詳細については、「[AWS::RDS::DBSecurityGroup](#)」と「[Ingress](#)」を参照してください。

JSON

```

"MyDbSecurityByCIDRIPGroup" : {
  "Type" : "AWS::RDS::DBSecurityGroup",
  "Properties" : {
    "GroupDescription" : "Ingress for CIDRIP",
    "DBSecurityGroupIngress" : {
      "CIDRIP" : "192.168.0.0/32"
    }
  }
}

```

```

    }
  }
}
```

YAML

```

MyDbSecurityByCIDRIPGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    GroupDescription: Ingress for CIDRIP
    DBSecurityGroupIngress:
      CIDRIP: "192.168.0.0/32"
```

Amazon EC2 セキュリティグループを使用した Amazon RDS DBSecurityGroup

この例では、MyEc2SecurityGroup で参照される Amazon EC2 セキュリティグループに対して [AWS::RDS::DBSecurityGroup](#) リソースへの進入を許可しています。

そのために、EC2 セキュリティグループを定義し、次に Ref 組み込み関数を使用して DBSecurityGroup 内の EC2 セキュリティグループを参照しています。

JSON

```

"DBInstance" : {
  "Type": "AWS::RDS::DBInstance",
  "Properties": {
    "DBName"          : { "Ref" : "DBName" },
    "Engine"          : "MySQL",
    "MasterUsername"  : { "Ref" : "DBUsername" },
    "DBInstanceClass" : { "Ref" : "DBClass" },
    "DBSecurityGroups" : [ { "Ref" : "DBSecurityGroup" } ],
    "AllocatedStorage" : { "Ref" : "DBAllocatedStorage" },
    "MasterUserPassword": { "Ref" : "DBPassword" }
  }
},

"DBSecurityGroup": {
  "Type": "AWS::RDS::DBSecurityGroup",
  "Properties": {
    "DBSecurityGroupIngress": {
      "EC2SecurityGroupName": {
        "Fn::GetAtt": ["WebServerSecurityGroup", "GroupName"]
      }
    }
  }
}
```

```

    }
  },
  "GroupDescription" : "Frontend Access"
}
},
"WebServerSecurityGroup" : {
  "Type" : "AWS::EC2::SecurityGroup",
  "Properties" : {
    "GroupDescription" : "Enable HTTP access via port 80 and SSH access",
    "SecurityGroupIngress" : [
      {"IpProtocol" : "tcp", "FromPort" : 80, "ToPort" : 80, "CidrIp" :
"0.0.0.0/0"},
      {"IpProtocol" : "tcp", "FromPort" : 22, "ToPort" : 22, "CidrIp" : "0.0.0.0/0"}
    ]
  }
}
}

```

YAML

この例は、以下の完全なサンプルから抽出したものです。
[Drupal_Single_Instance_With_RDS.template](#)

```

DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MasterUsername:
      Ref: DBUsername
    DBInstanceClass:
      Ref: DBClass
    DBSecurityGroups:
      - Ref: DBSecurityGroup
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
      Ref: DBPassword
DBSecurityGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    DBSecurityGroupIngress:

```

```

    EC2SecurityGroupName:
      Ref: WebServerSecurityGroup
    GroupDescription: Frontend Access
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0

```

複数の VPC セキュリティグループ

この例では、「[AWS::RDS::DBSecurityGroupIngress](#)」で複数の Amazon EC2 VPC セキュリティグループに対する進入認可を持つ「[AWS::RDS::DBSecurityGroup](#)」リソースが示されています。

JSON

```

{
  "Resources" : {
    "DBInstance" : {
      "Type" : "AWS::RDS::DBInstance",
      "Properties" : {
        "AllocatedStorage" : "5",
        "DBInstanceClass" : "db.t2.small",
        "DBName" : { "Ref" : "MyDBName" },
        "DBSecurityGroups" : [ { "Ref" : "DbSecurityByEC2SecurityGroup" } ],
        "DBSubnetGroupName" : { "Ref" : "MyDBSubnetGroup" },
        "Engine" : "MySQL",
        "MasterUserPassword": { "Ref" : "MyDBPassword" },
        "MasterUsername" : { "Ref" : "MyDBUsername" }
      },
      "DeletionPolicy" : "Snapshot"
    },
    "DbSecurityByEC2SecurityGroup" : {
      "Type" : "AWS::RDS::DBSecurityGroup",
      "Properties" : {

```

```

        "GroupDescription" : "Ingress for Amazon EC2 security group",
        "EC2VpcId" : { "Ref" : "MyVPC" },
        "DBSecurityGroupIngress" : [ {
            "EC2SecurityGroupId" : "sg-b0ff1111",
            "EC2SecurityGroupOwnerId" : "111122223333"
        }, {
            "EC2SecurityGroupId" : "sg-ffd72222",
            "EC2SecurityGroupOwnerId" : "111122223333"
        } ]
    }
}
}
}
}
}

```

YAML

```

Resources:
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: '5'
      DBInstanceClass: db.t2.small
      DBName:
        Ref: MyDBName
      DBSecurityGroups:
        - Ref: DbSecurityByEC2SecurityGroup
      DBSubnetGroupName:
        Ref: MyDBSubnetGroup
      Engine: MySQL
      MasterUserPassword:
        Ref: MyDBPassword
      MasterUsername:
        Ref: MyDBUsername
      DeletionPolicy: Snapshot
  DbSecurityByEC2SecurityGroup:
    Type: AWS::RDS::DBSecurityGroup
    Properties:
      GroupDescription: Ingress for Amazon EC2 security group
      EC2VpcId:
        Ref: MyVPC
      DBSecurityGroupIngress:
        - EC2SecurityGroupId: sg-b0ff1111
          EC2SecurityGroupOwnerId: '111122223333'

```

```
- EC2SecurityGroupId: sg-ffd72222
  EC2SecurityGroupOwnerId: '111122223333'
```

VPC セキュリティグループの Amazon RDS データベースインスタンス

この例は、Amazon EC2 VPC セキュリティグループに関連付けられた Amazon RDS データベースインスタンスを示しています。

JSON

```
{
  "DBEC2SecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Open database for access",
      "SecurityGroupIngress": [{
        "IpProtocol": "tcp",
        "FromPort": 3306,
        "ToPort": 3306,
        "SourceSecurityGroupName": { "Ref": "WebServerSecurityGroup" }
      }]
    }
  },
  "DBInstance": {
    "Type": "AWS::RDS::DBInstance",
    "Properties": {
      "DBName": { "Ref": "DBName" },
      "Engine": "MySQL",
      "MultiAZ": { "Ref": "MultiAZDatabase" },
      "MasterUsername": { "Ref": "DBUser" },
      "DBInstanceClass": { "Ref": "DBClass" },
      "AllocatedStorage": { "Ref": "DBAllocatedStorage" },
      "MasterUserPassword": { "Ref": "DBPassword" },
      "VPCSecurityGroups": [ { "Fn::GetAtt": [ "DBEC2SecurityGroup", "GroupId" ] } ]
    }
  }
}
```

YAML

```
DBEC2SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
```



```
GroupDescription: Open database for access
SecurityGroupIngress:
- IpProtocol: tcp
  FromPort: 3306
  ToPort: 3306
  SourceSecurityGroupName:
    Ref: WebServerSecurityGroup
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MultiAZ:
      Ref: MultiAZDatabase
    MasterUsername:
      Ref: DBUser
    DBInstanceClass:
      Ref: DBClass
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
      Ref: DBPassword
    VPCSecurityGroups:
      - !GetAtt DBEC2SecurityGroup.GroupId
```

Route 53 テンプレートスニペット

トピック

- [ホストゾーンの名前または ID を使用した Amazon Route 53 リソースレコードセット](#)
- [RecordSetGroup を使用した加重リソースレコードセットの設定](#)
- [RecordSetGroup を使用したエイリアスリソースレコードセットの設定](#)
- [CloudFront デイストリビューションのエイリアスリソースレコードセット](#)

ホストゾーンの名前または ID を使用した Amazon Route 53 リソースレコードセット

Amazon Route 53 リソースレコードセットを作成する際は、その追加先となるホストゾーンを指定する必要があります。CloudFormation では、2 つの方法でホストゾーンを指定できます。

- HostedZoneId プロパティを使用して、ホストゾーンを明示的に指定できます。

- HostedZoneName プロパティを使用して、CloudFormation にホストゾーンを検索させることができます。HostedZoneName プロパティを使用して同じ名前のホストゾーンが複数見つかった場合、CloudFormation はスタックを作成しません。

HostedZoneId を使用したリソースレコードセットの追加

この例では、ドメイン名が SPF の `mysite.example.com` レコードを保存し、HostedZoneId プロパティを使用してホストゾーンを指定する、Amazon Route 53 リソースレコードセットを追加しています。

JSON

```
"myDNSRecord" : {
  "Type" : "AWS::Route53::RecordSet",
  "Properties" :
  {
    "HostedZoneId" : "Z3DG6IL3SJCGPX",
    "Name" : "mysite.example.com.",
    "Type" : "SPF",
    "TTL" : "900",
    "ResourceRecords" : [ "\"v=spf1 ip4:192.168.0.1/16 -all\"" ]
  }
}
```

YAML

```
myDNSRecord:
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: Z3DG6IL3SJCGPX
    Name: mysite.example.com.
    Type: SPF
    TTL: '900'
    ResourceRecords:
      - '"v=spf1 ip4:192.168.0.1/16 -all"'
```

HostedZoneName を使用したリソースレコードセットの追加

この例では、HostedZoneName プロパティを使用してホストゾーンを指定するドメイン名「`mysite.example.com`」の Amazon Route 53 リソースレコードセットを追加します。

JSON

```
"myDNSRecord2" : {
    "Type" : "AWS::Route53::RecordSet",
    "Properties" : {
        "HostedZoneName" : "example.com.",
        "Name" : "mysite.example.com.",
        "Type" : "A",
        "TTL" : "900",
        "ResourceRecords" : [
            "192.168.0.1",
            "192.168.0.2"
        ]
    }
}
```

YAML

```
myDNSRecord2:
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneName: example.com.
    Name: mysite.example.com.
    Type: A
    TTL: '900'
    ResourceRecords:
      - 192.168.0.1
      - 192.168.0.2
```

RecordSetGroup を使用した加重リソースレコードセットの設定

この例では、[AWS::Route53::RecordSetGroup](#) を使用して、「example.com」の 2 つの CNAME レコードを設定します。ホストゾーン RecordSets プロパティは、「mysite.example.com」という DNS 名の CNAME レコードセットを保持します。各レコードセットには、識別子 (SetIdentifier) と重み (Weight) が含まれます。リソースにルーティングされるインターネットトラフィックの割合は、次の計算に基づきます。

- Frontend One: $140 / (140 + 60) = 140 / 200 = 70\%$
- Frontend Two: $60 / (140 + 60) = 60 / 200 = 30\%$

加重リソースレコードセットの詳細については、「Amazon Route 53 デベロッパーガイド」の「[Weighted routing](#)」(加重ルーティング)を参照してください。

JSON

```
"myDNSOne" : {
  "Type" : "AWS::Route53::RecordSetGroup",
  "Properties" : {
    "HostedZoneName" : "example.com.",
    "Comment" : "Weighted RR for my frontends.",
    "RecordSets" : [
      {
        "Name" : "mysite.example.com.",
        "Type" : "CNAME",
        "TTL" : "900",
        "SetIdentifier" : "Frontend One",
        "Weight" : "140",
        "ResourceRecords" : ["example-ec2.amazonaws.com"]
      },
      {
        "Name" : "mysite.example.com.",
        "Type" : "CNAME",
        "TTL" : "900",
        "SetIdentifier" : "Frontend Two",
        "Weight" : "60",
        "ResourceRecords" : ["example-ec2-larger.amazonaws.com"]
      }
    ]
  }
}
```

YAML

```
myDNSOne:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Weighted RR for my frontends.
    RecordSets:
      - Name: mysite.example.com.
        Type: CNAME
        TTL: '900'
        SetIdentifier: Frontend One
```

```
Weight: '140'
ResourceRecords:
- example-ec2.amazonaws.com
- Name: mysite.example.com.
  Type: CNAME
  TTL: '900'
  SetIdentifier: Frontend Two
  Weight: '60'
  ResourceRecords:
  - example-ec2-larger.amazonaws.com
```

RecordSetGroup を使用したエイリアスリソースレコードセットの設定

次の例では、[AWS::Route53::RecordSetGroup](#) を使用して、ELB バージョン 1 (Classic) ロードバランサーとバージョン 2 (Application または Network) Iロードバランサーにトラフィックをルーティングする example.com という名前のエイリアスリソースレコードセットを設定します。[AliasTarget](#) プロパティは、GetAtt 組み込み関数を使用して、myELB LoadBalancer のホストゾーン ID と DNS 名を指定します。GetAtt は、トラフィックをバージョン 1 かバージョン 2 のどちらのロードバランサーにルーティングするかに応じて、myELB リソースのさまざまなプロパティを取得します。

- バージョン 1 ロードバランサー: CanonicalHostedZoneNameID および DNSName
- バージョン 2 ロードバランサー: CanonicalHostedZoneID および DNSName

エイリアスリソースレコードセットの詳細については、「Route 53 デベロッパーガイド」の「[Choosing between alias and non-alias records](#)」(エイリアスレコードと非エイリアスレコードの選択) を参照してください。

バージョン 1 ロードバランサーの JSON

```
"myELB" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "AvailabilityZones" : [ "us-east-1a" ],
    "Listeners" : [ {
      "LoadBalancerPort" : "80",
      "InstancePort" : "80",
      "Protocol" : "HTTP"
    } ]
  }
}
```

```

    },
    "myDNS" : {
      "Type" : "AWS::Route53::RecordSetGroup",
      "Properties" : {
        "HostedZoneName" : "example.com.",
        "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
        "RecordSets" : [
          {
            "Name" : "example.com.",
            "Type" : "A",
            "AliasTarget" : {
              "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
"CanonicalHostedZoneNameID"] },
              "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
            }
          }
        ]
      }
    }
  }
}

```

バージョン 1 ロードバランサーの YAML

```

myELB:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
      - "us-east-1a"
    Listeners:
      - LoadBalancerPort: '80'
        InstancePort: '80'
        Protocol: HTTP
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
    RecordSets:
      - Name: example.com.
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneNameID'
          DNSName: !GetAtt 'myELB.DNSName'

```

バージョン 2 ロードバランサーの JSON

```
"myELB" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "Subnets" : [
      {"Ref": "SubnetAZ1"},
      {"Ref" : "SubnetAZ2"}
    ]
  }
},
"myDNS" : {
  "Type" : "AWS::Route53::RecordSetGroup",
  "Properties" : {
    "HostedZoneName" : "example.com.",
    "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
    "RecordSets" : [
      {
        "Name" : "example.com.",
        "Type" : "A",
        "AliasTarget" : {
          "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
"CanonicalHostedZoneID"] },
          "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
        }
      }
    ]
  }
}
```

バージョン 2 ロードバランサーの YAML

```
myELB:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Subnets:
      - Ref: SubnetAZ1
      - Ref: SubnetAZ2
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
```

```
RecordSets:
- Name: example.com.
  Type: A
  AliasTarget:
    HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneID'
    DNSName: !GetAtt 'myELB.DNSName'
```

CloudFront デистриビューションのエイリアスリソースレコードセット

次の例では、エイリアス A レコードを作成して、カスタムドメイン名を既存の CloudFront デистриビューションにポイントします。myHostedZoneID は、同じテンプレート内の実際の AWS::Route53::HostedZone リソースへの参照またはパラメータのいずれかであると想定されます。myCloudFrontDistribution は、同じテンプレート内の AWS::CloudFront::Distribution リソースを参照します。エイリアスレコードは、標準の CloudFront ホストゾーン ID (Z2FDTNDATAQYW2) を使用し、Fn::GetAtt を使用してデистриビューションのドメイン名を自動的に解決します。この設定により、IP アドレスを必要とせずに、ウェブトラフィックをカスタムドメインから CloudFront デистриビューションにルーティングできます。

Note

エイリアスリソースレコードセットを作成する場合は、HostedZoneId プロパティに Z2FDTNDATAQYW2 を指定する必要があります。CloudFront 用のエイリアスリソースレコードセットは、プライベートホストゾーンでは作成できません。

JSON

```
{
  "myDNS": {
    "Type": "AWS::Route53::RecordSetGroup",
    "Properties": {
      "HostedZoneId": {
        "Ref": "myHostedZoneID"
      },
      "RecordSets": [
        {
          "Name": {
            "Ref": "myRecordSetDomainName"
          },
          "Type": "A",
```



```

        "AliasTarget": {
            "HostedZoneId": "Z2FDTNDATAQYW2",
            "DNSName": {
                "Fn::GetAtt": [
                    "myCloudFrontDistribution",
                    "DomainName"
                ]
            },
            "EvaluateTargetHealth": false
        }
    ]
}
}
}
}

```

YAML

```

myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneId: !Ref myHostedZoneID
    RecordSets:
      - Name: !Ref myRecordSetDomainName
        Type: A
        AliasTarget:
          HostedZoneId: Z2FDTNDATAQYW2
          DNSName: !GetAtt
            - myCloudFrontDistribution
            - DomainName
        EvaluateTargetHealth: false

```

Amazon S3 テンプレートスニペット

これらの Amazon S3 サンプルテンプレートを使用すると、CloudFormation で Amazon S3 バケットを記述するのに役立ちます。その他の例については、AWS::S3::Bucket リソースの「[Examples](#)」のセクションを参照してください。

トピック

- [デフォルトでの Amazon S3 バケットの作成](#)
- [ウェブサイトホスティングの Amazon S3 バケット \(DeletionPolicyを持つもの\) を作成](#)

- [カスタムドメインを使用した静的ウェブサイトの作成](#)

デフォルトでの Amazon S3 バケットの作成

次の例では、[AWS::S3::Bucket](#) を使用してデフォルト設定でバケットを作成します。

JSON

```
"myS3Bucket" : {
  "Type" : "AWS::S3::Bucket"
}
```

YAML

```
MyS3Bucket:
  Type: AWS::S3::Bucket
```

ウェブサイトホスティングの Amazon S3 バケット (**DeletionPolicy**を持つもの) を作成

この例では、バケットをウェブサイトとして作成し、ブロックパブリックアクセスを無効にします (ウェブサイトのホスティングのために設定されたバケットにはパブリック読み取り許可が必要です)。その後、パブリックバケットポリシーがバケットに追加されます。このバケットリソースは、`DeletionPolicy` 属性が `Retain` に設定されているため、CloudFormation がスタックを削除する際、このバケットは削除されません。Output セクションは `Fn::GetAtt` を使用して `S3Bucket` リソースの `WebsiteURL` 属性と `DomainName` 属性を取得します。

Note

次の例では、アカウントレベルで `BlockPublicPolicy` および `RestrictPublicBuckets` ブロックパブリックアクセス設定が無効になっていることを前提としています。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
```

```

"S3Bucket": {
  "Type": "AWS::S3::Bucket",
  "Properties": {
    "PublicAccessBlockConfiguration": {
      "BlockPublicAcls": false,
      "BlockPublicPolicy": false,
      "IgnorePublicAcls": false,
      "RestrictPublicBuckets": false
    },
    "WebsiteConfiguration": {
      "IndexDocument": "index.html",
      "ErrorDocument": "error.html"
    }
  },
  "DeletionPolicy": "Retain",
  "UpdateReplacePolicy": "Retain"
},
"BucketPolicy": {
  "Type": "AWS::S3::BucketPolicy",
  "Properties": {
    "PolicyDocument": {
      "Id": "MyPolicy",
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "PublicReadForGetBucketObjects",
          "Effect": "Allow",
          "Principal": "*",
          "Action": "s3:GetObject",
          "Resource": {
            "Fn::Join": [
              "",
              [
                "arn:aws:s3:::",
                {
                  "Ref": "S3Bucket"
                },
                "/*"
              ]
            ]
          }
        }
      ]
    }
  }
},

```

```

        "Bucket": {
            "Ref": "S3Bucket"
        }
    },
    "Outputs": {
        "WebsiteURL": {
            "Value": {
                "Fn::GetAtt": [
                    "S3Bucket",
                    "WebsiteURL"
                ]
            },
            "Description": "URL for website hosted on S3"
        },
        "S3BucketSecureURL": {
            "Value": {
                "Fn::Join": [
                    "",
                    [
                        "https://",
                        {
                            "Fn::GetAtt": [
                                "S3Bucket",
                                "DomainName"
                            ]
                        }
                    ]
                ]
            },
            "Description": "Name of S3 bucket to hold website content"
        }
    }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:

```

```
PublicAccessBlockConfiguration:
  BlockPublicAcls: false
  BlockPublicPolicy: false
  IgnorePublicAcls: false
  RestrictPublicBuckets: false
WebsiteConfiguration:
  IndexDocument: index.html
  ErrorDocument: error.html
DeletionPolicy: Retain
UpdateReplacePolicy: Retain
BucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    PolicyDocument:
      Id: MyPolicy
      Version: 2012-10-17
      Statement:
        - Sid: PublicReadForGetBucketObjects
          Effect: Allow
          Principal: '*'
          Action: 's3:GetObject'
          Resource: !Join
            - ''
            - - 'arn:aws:s3:::'
              - !Ref S3Bucket
            - /*
        Bucket: !Ref S3Bucket
Outputs:
  WebsiteURL:
    Value: !GetAtt
      - S3Bucket
      - WebsiteURL
    Description: URL for website hosted on S3
  S3BucketSecureURL:
    Value: !Join
      - ''
      - - 'https://'
        - !GetAtt
          - S3Bucket
          - DomainName
    Description: Name of S3 bucket to hold website content
```

カスタムドメインを使用した静的ウェブサイトの作成

登録済みのドメインで Route 53 を使用できます。次のサンプルでは、ドメイン用に Route 53 に既にホストゾーンを作成していると仮定しています。例では、ウェブサイトホスティング用の 2 個のバケットを作成します。ルートバケットはコンテンツをホストし、もう 1 つのバケットが `www.domainname.com` リクエストをルートバケットにリダイレクトします。レコードセットはドメイン名を Amazon S3 エンドポイントにマッピングします。

上記の例で示しているように、バケットポリシーを追加する必要があります。

カスタムドメインの使用に関する詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定](#)」を参照してください。

Note

次の例では、アカウントレベルで BlockPublicPolicy および RestrictPublicBuckets ブロックパブリックアクセス設定が無効になっていることを前提としています。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : { "S3hostedzoneID" : "Z3AQBSTGFYJSTF", "websiteendpoint" :
"s3-website-us-east-1.amazonaws.com" },
      "us-west-1" : { "S3hostedzoneID" : "Z2F56UZL2M1ACD", "websiteendpoint" :
"s3-website-us-west-1.amazonaws.com" },
      "us-west-2" : { "S3hostedzoneID" : "Z3BJ6K6RIION7M", "websiteendpoint" :
"s3-website-us-west-2.amazonaws.com" },
      "eu-west-1" : { "S3hostedzoneID" : "Z1BKCTXD74EZPE", "websiteendpoint" :
"s3-website-eu-west-1.amazonaws.com" },
      "ap-southeast-1" : { "S3hostedzoneID" : "Z300J2DXBE1FTB",
"websiteendpoint" : "s3-website-ap-southeast-1.amazonaws.com" },
      "ap-southeast-2" : { "S3hostedzoneID" : "Z1WCIGYICN2BYD",
"websiteendpoint" : "s3-website-ap-southeast-2.amazonaws.com" },
      "ap-northeast-1" : { "S3hostedzoneID" : "Z2M4EHUR26P7ZW",
"websiteendpoint" : "s3-website-ap-northeast-1.amazonaws.com" },
    }
  }
```

```

        "sa-east-1" : { "S3hostedzoneID" : "Z31GFT0UA1I2HV", "websiteendpoint" :
"s3-website-sa-east-1.amazonaws.com" }
    },
    "Parameters": {
        "RootDomainName": {
            "Description": "Domain name for your website (example.com)",
            "Type": "String"
        }
    },
    "Resources": {
        "RootBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName" : {"Ref":"RootDomainName"},
                "PublicAccessBlockConfiguration": {
                    "BlockPublicAcls": false,
                    "BlockPublicPolicy": false,
                    "IgnorePublicAcls": false,
                    "RestrictPublicBuckets": false
                },
                "WebsiteConfiguration": {
                    "IndexDocument":"index.html",
                    "ErrorDocument":"404.html"
                }
            }
        },
        "WWWBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": {
                    "Fn::Join": [ "", [ "www.", {"Ref":"RootDomainName"} ] ]
                },
                "AccessControl": "BucketOwnerFullControl",
                "WebsiteConfiguration": {
                    "RedirectAllRequestsTo": {
                        "HostName": {"Ref": "RootBucket"}
                    }
                }
            }
        },
        "myDNS": {
            "Type": "AWS::Route53::RecordSetGroup",
            "Properties": {

```

```

    "HostedZoneName": {
      "Fn::Join": [ "", [ {"Ref": "RootDomainName"}, "." ] ]
    },
    "Comment": "Zone apex alias.",
    "RecordSets": [
      {
        "Name": { "Ref": "RootDomainName" },
        "Type": "A",
        "AliasTarget": {
          "HostedZoneId": { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "S3hostedzoneID" ] },
          "DNSName": { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "websiteendpoint" ] }
        },
        {
          "Name": {
            "Fn::Join": [ "", [ "www.", { "Ref": "RootDomainName" } ] ]
          },
          "Type": "CNAME",
          "TTL" : "900",
          "ResourceRecords" : [
            { "Fn::GetAtt": [ "WWWBucket", "DomainName" ] }
          ]
        }
      ]
    }
  },
  "Outputs": {
    "WebsiteURL": {
      "Value": { "Fn::GetAtt": [ "RootBucket", "WebsiteURL" ] },
      "Description": "URL for website hosted on S3"
    }
  }
}

```

YAML

```

Parameters:
  RootDomainName:
    Description: Domain name for your website (example.com)
    Type: String

```


Mappings:**RegionMap:****us-east-1:**

S3hostedzoneID: Z3AQBSTGIFYJSTF

websiteendpoint: s3-website-us-east-1.amazonaws.com

us-west-1:

S3hostedzoneID: Z2F56UZL2M1ACD

websiteendpoint: s3-website-us-west-1.amazonaws.com

us-west-2:

S3hostedzoneID: Z3BJ6K6RIION7M

websiteendpoint: s3-website-us-west-2.amazonaws.com

eu-west-1:

S3hostedzoneID: Z1BKCTXD74EZPE

websiteendpoint: s3-website-eu-west-1.amazonaws.com

ap-southeast-1:

S3hostedzoneID: Z300J2DXBE1FTB

websiteendpoint: s3-website-ap-southeast-1.amazonaws.com

ap-southeast-2:

S3hostedzoneID: Z1WCIGYICN2BYD

websiteendpoint: s3-website-ap-southeast-2.amazonaws.com

ap-northeast-1:

S3hostedzoneID: Z2M4EHUR26P7ZW

websiteendpoint: s3-website-ap-northeast-1.amazonaws.com

sa-east-1:

S3hostedzoneID: Z31GFT0UA1I2HV

websiteendpoint: s3-website-sa-east-1.amazonaws.com

Resources:**RootBucket:**

Type: AWS::S3::Bucket

Properties:

BucketName: !Ref RootDomainName

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

WebsiteConfiguration:

IndexDocument: index.html

ErrorDocument: 404.html

WWWBucket:

Type: AWS::S3::Bucket

Properties:

BucketName: !Sub

- www.\${Domain}

```

    - Domain: !Ref RootDomainName
    AccessControl: BucketOwnerFullControl
    WebsiteConfiguration:
      RedirectAllRequestsTo:
        HostName: !Ref RootBucket
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: !Sub
    - ${Domain}.
    - Domain: !Ref RootDomainName
    Comment: Zone apex alias.
    RecordSets:
      - Name: !Ref RootDomainName
        Type: A
        AliasTarget:
          HostedZoneId: !FindInMap [ RegionMap, !Ref 'AWS::Region', S3hostedzoneID]
          DNSName: !FindInMap [ RegionMap, !Ref 'AWS::Region', websiteendpoint]
      - Name: !Sub
        - www.${Domain}
        - Domain: !Ref RootDomainName
        Type: CNAME
        TTL: 900
        ResourceRecords:
          - !GetAtt WWWBucket.DomainName
  Outputs:
    WebsiteURL:
      Value: !GetAtt RootBucket.WebsiteURL
      Description: URL for website hosted on S3

```

Amazon SNS テンプレートスニペット

この例は、Amazon SNS トピックリソースを示します。有効なメールアドレスが必要です。

JSON

```

"MySNSTopic" : {
  "Type" : "AWS::SNS::Topic",
  "Properties" : {
    "Subscription" : [ {
      "Endpoint" : "add valid email address",
      "Protocol" : "email"
    } ]
  }
}

```

```
}  
}
```

YAML

```
MySNSTopic:  
  Type: AWS::SNS::Topic  
  Properties:  
    Subscription:  
      - Endpoint: "add valid email address"  
        Protocol: email
```

Amazon SQS テンプレートスニペット

この例は、Amazon SQS キューを示します。

JSON

```
"MyQueue" : {  
  "Type" : "AWS::SQS::Queue",  
  "Properties" : {  
    "VisibilityTimeout" : "value"  
  }  
}
```

YAML

```
MyQueue:  
  Type: AWS::SQS::Queue  
  Properties:  
    VisibilityTimeout: value
```

Amazon Timestream テンプレートスニペット

Amazon Timestream for InfluxDB を使用すると、アプリケーションデベロッパーや DevOps チームは、オープンソース API を使用して、フルマネージド InfluxDB データベースを AWS で簡単に実行し、リアルタイムの時系列アプリケーションを実行できます。要求の厳しい時系列ワークロードを処理する InfluxDB データベースをすばやく作成できます。いくつかの簡単な API コールを使用すると自動化されたソフトウェア パッチ適用、バックアップ、リカバリを使用して、AWS の InfluxDB データベースをセットアップ、移行、運用、スケーリングできます。これらのサンプルは、GitHub

の awslabs/amazon-timestream-tools/tree/mainline/integrations/cloudformation/timestream-influxdb でも入手できます。

トピック

- [デフォルト値を使用した最小限のサンプル](#)
- [パラメータを使用したより完全な例](#)

これらの CloudFormation テンプレートは、Amazon Timestream for InfluxDB インスタンスを正常に作成、接続、モニタリングするために必要な以下のリソースを作成します。

Amazon VPC

- VPC
- 1 つ以上のSubnet
- InternetGateway
- RouteTable
- SecurityGroup

Amazon S3

- Bucket

Amazon Timestream

- InfluxDBInstance

デフォルト値を使用した最小限のサンプル

この例では、可能であればデフォルト値を使用して、マルチ AZ およびパブリックにアクセス可能なインスタンスをデプロイします。

JSON

```
{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
      "ParameterGroups": [
```

```

    {
      "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
      "Parameters": [
        "DbInstanceName",
        "InfluxDBPassword"
      ]
    }
  ],
  "ParameterLabels": {
    "VPCCIDR": {"default": "VPC CIDR"}
  }
},
"Parameters": {
  "DbInstanceName": {
    "Description": "The name that uniquely identifies the DB instance when
interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
will also be a prefix included in the endpoint. DB instance names must be unique per
customer and per Region.",
    "Type": "String",
    "Default": "mydbinstance",
    "MinLength": 3,
    "MaxLength": 40,
    "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
  },
  "InfluxDBPassword": {
    "Description": "The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in AWS Secrets Manager in your account.",
    "Type": "String",
    "NoEcho": true,
    "MinLength": 8,
    "MaxLength": 64,
    "AllowedPattern": "^[a-zA-Z0-9]+$"
  }
},
"Resources": {
  "VPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {"CidrBlock": "10.0.0.0/16"}
  },
  "InternetGateway": {"Type": "AWS::EC2::InternetGateway"},
  "InternetGatewayAttachment": {

```

```

    "Type": "AWS::EC2::VPCGatewayAttachment",
    "Properties": {
      "InternetGatewayId": {"Ref": "InternetGateway"},
      "VpcId": {"Ref": "VPC"}
    }
  },
  "Subnet1": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": {"Ref": "VPC"},
      "AvailabilityZone": {
        "Fn::Select": [
          0,
          {"Fn::GetAZs": ""}
        ]
      },
      "CidrBlock": {
        "Fn::Select": [
          0,
          {
            "Fn::Cidr": [
              {
                "Fn::GetAtt": [
                  "VPC",
                  "CidrBlock"
                ]
              },
              2,
              12
            ]
          }
        ]
      },
      "MapPublicIpOnLaunch": true
    }
  },
  "Subnet2": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": {"Ref": "VPC"},
      "AvailabilityZone": {
        "Fn::Select": [
          1,
          {"Fn::GetAZs": ""}
        ]
      }
    }
  }
}

```

```

    ]
  },
  "CidrBlock": {
    "Fn::Select": [
      1,
      {
        "Fn::Cidr": [
          {
            "Fn::GetAtt": [
              "VPC",
              "CidrBlock"
            ]
          },
          2,
          12
        ]
      }
    ]
  },
  "MapPublicIpOnLaunch": true
},
"RouteTable": {
  "Type": "AWS::EC2::RouteTable",
  "Properties": {
    "VpcId": {"Ref": "VPC"}
  }
},
"DefaultRoute": {
  "Type": "AWS::EC2::Route",
  "DependsOn": "InternetGatewayAttachment",
  "Properties": {
    "RouteTableId": {"Ref": "RouteTable"},
    "DestinationCidrBlock": "0.0.0.0/0",
    "GatewayId": {"Ref": "InternetGateway"}
  }
},
"Subnet1RouteTableAssociation": {
  "Type": "AWS::EC2::SubnetRouteTableAssociation",
  "Properties": {
    "RouteTableId": {"Ref": "RouteTable"},
    "SubnetId": {"Ref": "Subnet1"}
  }
},

```

```

    "Subnet2RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet2"}
      }
    },
    "InfluxDBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupName": "influxdb-sg",
        "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
        "VpcId": {"Ref": "VPC"}
      }
    },
    "InfluxDBSecurityGroupIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": {"Ref": "InfluxDBSecurityGroup"},
        "IpProtocol": "tcp",
        "CidrIp": "0.0.0.0/0",
        "FromPort": 8086,
        "ToPort": 8086
      }
    },
    "InfluxDBLogsS3Bucket": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain"
    },
    "InfluxDBLogsS3BucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "s3:PutObject",
              "Effect": "Allow",
              "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*"},
              "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
            }
          ]
        }
      }
    }
  }

```



```

        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
            {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
        ],
        "Principal": "*",
        "Condition": {
            "Bool": {"aws:SecureTransport": false}
        }
    }
]
}
},
"DbInstance": {
    "Type": "AWS::Timestream::InfluxDBInstance",
    "DependsOn": "InfluxDBLogsS3BucketPolicy",
    "Properties": {
        "AllocatedStorage": 20,
        "DbInstanceType": "db.influx.medium",
        "Name": {"Ref": "DbInstanceName"},
        "Password": {"Ref": "InfluxDBPassword"},
        "PubliclyAccessible": true,
        "DeploymentType": "WITH_MULTIAZ_STANDBY",
        "VpcSecurityGroupIds": [
            {"Ref": "InfluxDBSecurityGroup"}
        ],
        "VpcSubnetIds": [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
        ],
        "LogDeliveryConfiguration": {
            "S3Configuration": {
                "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
                "Enabled": true
            }
        }
    }
},
"Outputs": {
    "VPC": {
        "Description": "A reference to the VPC used to create network resources",

```

```

    "Value": {"Ref": "VPC"}
  },
  "Subnets": {
    "Description": "A list of the subnets created",
    "Value": {
      "Fn::Join": [
        ",",
        [
          {"Ref": "Subnet1"},
          {"Ref": "Subnet2"}
        ]
      ]
    }
  },
  "Subnet1": {
    "Description": "A reference to the subnet in the 1st Availability Zone",
    "Value": {"Ref": "Subnet1"}
  },
  "Subnet2": {
    "Description": "A reference to the subnet in the 2nd Availability Zone",
    "Value": {"Ref": "Subnet2"}
  },
  "InfluxDBSecurityGroup": {
    "Description": "Security group with port 8086 ingress rule",
    "Value": {"Ref": "InfluxDBSecurityGroup"}
  },
  "InfluxDBLogsS3Bucket": {
    "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
    "Value": {"Ref": "InfluxDBLogsS3Bucket"}
  },
  "DbInstance": {
    "Description": "A reference to the Timestream for InfluxDB DB instance",
    "Value": {"Ref": "DbInstance"}
  },
  "InfluxAuthParametersSecretArn": {
    "Description": "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret containing the initial InfluxDB authorization parameters. The secret value is a JSON formatted key-value pair holding InfluxDB authorization values: organization, bucket, username, and password.",
    "Value": {
      "Fn::GetAtt": [
        "DbInstance",
        "InfluxAuthParametersSecretArn"
      ]
    }
  }
}

```

```

    }
  },
  "Endpoint": {
    "Description": "The endpoint URL to connect to InfluxDB",
    "Value": {
      "Fn::Join": [
        "",
        [
          "https://",
          {
            "Fn::GetAtt": [
              "DbInstance",
              "Endpoint"
            ]
          }
        ]
      ],
      ":8086"
    ]
  }
}

```

YAML

```

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBPassword
    ParameterLabels:
      VPCCIDR:
        default: VPC CIDR

Parameters:
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a

```

prefix included in the endpoint. DB instance names must be unique per customer and per Region.

Type: String

Default: mydbinstance

MinLength: 3

MaxLength: 40

AllowedPattern: `^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$`

InfluxDBPassword:

Description: The password of the initial admin user created in InfluxDB. This password will allow you to access the InfluxDB UI to perform various administrative tasks and also use the InfluxDB CLI to create an operator token. These attributes will be stored in a Secret created in AWS Secrets Manager in your account.

Type: String

NoEcho: true

MinLength: 8

MaxLength: 64

AllowedPattern: `^[a-zA-Z0-9]+$`

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/16

InternetGateway:

Type: AWS::EC2::InternetGateway

InternetGatewayAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

Subnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [0, !GetAZs '']

CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12]]

MapPublicIpOnLaunch: true

Subnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [1, !GetAZs '']

CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12]]

MapPublicIpOnLaunch: true

```
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
DefaultRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref RouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
Subnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref RouteTable
    SubnetId: !Ref Subnet1
Subnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref RouteTable
    SubnetId: !Ref Subnet2
InfluxDBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "influxdb-sg"
    GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
    VpcId: !Ref VPC
InfluxDBSecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !Ref InfluxDBSecurityGroup
    IpProtocol: tcp
    CidrIp: 0.0.0.0/0
    FromPort: 8086
    ToPort: 8086
InfluxDBLogsS3Bucket:
  Type: AWS::S3::Bucket
  DeletionPolicy: Retain
InfluxDBLogsS3BucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    Bucket: !Ref InfluxDBLogsS3Bucket
    PolicyDocument:
      Version: '2012-10-17'
```

Statement:

- Action: "s3:PutObject"
Effect: Allow
Resource: !Sub arn:aws:s3:::\${InfluxDBLogsS3Bucket}/InfluxLogs/*
Principal:
Service: timestream-influxdb.amazonaws.com
- Action: "s3:*"
Effect: Deny
Resource:
 - !Sub arn:aws:s3:::\${InfluxDBLogsS3Bucket}/*
 - !Sub arn:aws:s3:::\${InfluxDBLogsS3Bucket}
- Principal: "*"
 - Condition:
 - Bool:
 - aws:SecureTransport: false

DbInstance:

Type: AWS::Timestream::InfluxDBInstance
 DependsOn: InfluxDBLogsS3BucketPolicy
 Properties:
 AllocatedStorage: 20
 DbInstanceType: db.influx.medium
 Name: !Ref DbInstanceName
 Password: !Ref InfluxDBPassword
 PubliclyAccessible: true
 DeploymentType: WITH_MULTIAZ_STANDBY
 VpcSecurityGroupIds:

- !Ref InfluxDBSecurityGroup

 VpcSubnetIds:

- !Ref Subnet1
- !Ref Subnet2

 LogDeliveryConfiguration:
 S3Configuration:
 BucketName: !Ref InfluxDBLogsS3Bucket
 Enabled: true

Outputs:

Network Resources

VPC:

Description: A reference to the VPC used to create network resources
 Value: !Ref VPC

Subnets:

Description: A list of the subnets created
 Value: !Join [",", [!Ref Subnet1, !Ref Subnet2]]

Subnet1:

```

    Description: A reference to the subnet in the 1st Availability Zone
    Value: !Ref Subnet1
Subnet2:
    Description: A reference to the subnet in the 2nd Availability Zone
    Value: !Ref Subnet2
InfluxDBSecurityGroup:
    Description: Security group with port 8086 ingress rule
    Value: !Ref InfluxDBSecurityGroup

# Timestream for InfluxDB Resources
InfluxDBLogsS3Bucket:
    Description: S3 Bucket containing InfluxDB logs from the DB instance
    Value: !Ref InfluxDBLogsS3Bucket
DbInstance:
    Description: A reference to the Timestream for InfluxDB DB instance
    Value: !Ref DbInstance
InfluxAuthParametersSecretArn:
    Description: "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
    containing the initial InfluxDB authorization parameters. The secret value is a JSON
    formatted key-value pair holding InfluxDB authorization values: organization, bucket,
    username, and password."
    Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn
Endpoint:
    Description: The endpoint URL to connect to InfluxDB
    Value: !Join ["", ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]

```

パラメータを使用したより完全な例

このサンプルテンプレートは、指定されたパラメータに基づいてネットワークリソースを動的に変更します。パラメータには、PubliclyAccessible と DeploymentType が含まれます。

JSON

```

{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
      "ParameterGroups": [
        {
          "Label": {"default": "Network Configuration"},
          "Parameters": ["VPCCIDR"]
        },
        {

```

```

    "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
    "Parameters": [
        "DbInstanceName",
        "InfluxDBUsername",
        "InfluxDBPassword",
        "InfluxDBOrganization",
        "InfluxDBBucket",
        "DbInstanceType",
        "DbStorageType",
        "AllocatedStorage",
        "PubliclyAccessible",
        "DeploymentType"
    ]
  },
  "ParameterLabels": {
    "VPCCIDR": {"default": "VPC CIDR"}
  }
},
"Parameters": {
  "VPCCIDR": {
    "Description": "Please enter the IP range (CIDR notation) for the new VPC",
    "Type": "String",
    "Default": "10.0.0.0/16"
  },
  "DbInstanceName": {
    "Description": "The name that uniquely identifies the DB instance when
interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
will also be a prefix included in the endpoint. DB instance names must be unique per
customer and per Region.",
    "Type": "String",
    "Default": "mydbinstance",
    "MinLength": 3,
    "MaxLength": 40,
    "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
  },
  "InfluxDBUsername": {
    "Description": "The username of the initial admin user created in InfluxDB. Must
start with a letter and can't end with a hyphen or contain two consecutive hyphens.
For example, my-user1. This username will allow you to access the InfluxDB UI to
perform various administrative tasks and also use the InfluxDB CLI to create an
operator token. These attributes will be stored in a Secret created in AWS Secrets
Manager in your account.",

```



```

    "Type": "String",
    "Default": "admin",
    "MinLength": 1,
    "MaxLength": 64
  },
  "InfluxDBPassword": {
    "Description": "The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in AWS Secrets Manager in your account.",
    "Type": "String",
    "NoEcho": true,
    "MinLength": 8,
    "MaxLength": 64,
    "AllowedPattern": "^[a-zA-Z0-9]+$"
  },
  "InfluxDBOrganization": {
    "Description": "The name of the initial organization for the initial admin user
in InfluxDB. An InfluxDB organization is a workspace for a group of users.",
    "Type": "String",
    "Default": "org",
    "MinLength": 1,
    "MaxLength": 64
  },
  "InfluxDBBucket": {
    "Description": "The name of the initial InfluxDB bucket. All InfluxDB data
is stored in a bucket. A bucket combines the concept of a database and a retention
period (the duration of time that each data point persists). A bucket belongs to an
organization.",
    "Type": "String",
    "Default": "bucket",
    "MinLength": 2,
    "MaxLength": 64,
    "AllowedPattern": "^[^_\\\\""][^\\\\""]*$"
  },
  "DeploymentType": {
    "Description": "Specifies whether the Timestream for InfluxDB is deployed as
Single-AZ or with a MultiAZ Standby for High availability",
    "Type": "String",
    "Default": "WITH_MULTIAZ_STANDBY",
    "AllowedValues": [
      "SINGLE_AZ",
      "WITH_MULTIAZ_STANDBY"
    ]
  }
]

```

```

    },
    "AllocatedStorage": {
      "Description": "The amount of storage to allocate for your DB storage type in GiB
(gibibytes).",
      "Type": "Number",
      "Default": 400,
      "MinValue": 20,
      "MaxValue": 16384
    },
    "DbInstanceType": {
      "Description": "The Timestream for InfluxDB DB instance type to run InfluxDB
on.",
      "Type": "String",
      "Default": "db.influx.medium",
      "AllowedValues": [
        "db.influx.medium",
        "db.influx.large",
        "db.influx.xlarge",
        "db.influx.2xlarge",
        "db.influx.4xlarge",
        "db.influx.8xlarge",
        "db.influx.12xlarge",
        "db.influx.16xlarge"
      ]
    },
    "DbStorageType": {
      "Description": "The Timestream for InfluxDB DB storage type to read and write
InfluxDB data.",
      "Type": "String",
      "Default": "InfluxIOIncludedT1",
      "AllowedValues": [
        "InfluxIOIncludedT1",
        "InfluxIOIncludedT2",
        "InfluxIOIncludedT3"
      ]
    },
    "PubliclyAccessible": {
      "Description": "Configures the DB instance with a public IP to facilitate
access.",
      "Type": "String",
      "Default": true,
      "AllowedValues": [
        true,
        false
      ]
    }
  }
}

```

```

    ]
  }
},
"Conditions": {
  "IsMultiAZ": {
    "Fn::Equals": [
      {"Ref": "DeploymentType"},
      "WITH_MULTIAZ_STANDBY"
    ]
  },
  "IsPublic": {
    "Fn::Equals": [
      {"Ref": "PubliclyAccessible"},
      true
    ]
  }
},
"Resources": {
  "VPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
      "CidrBlock": {"Ref": "VPCCIDR"}
    }
  },
  "InternetGateway": {
    "Type": "AWS::EC2::InternetGateway",
    "Condition": "IsPublic"
  },
  "InternetGatewayAttachment": {
    "Type": "AWS::EC2::VPCGatewayAttachment",
    "Condition": "IsPublic",
    "Properties": {
      "InternetGatewayId": {"Ref": "InternetGateway"},
      "VpcId": {"Ref": "VPC"}
    }
  },
  "Subnet1": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": {"Ref": "VPC"},
      "AvailabilityZone": {
        "Fn::Select": [
          0,
          {"Fn::GetAZs": ""}
        ]
      }
    }
  }
}

```

```

    ]
  },
  "CidrBlock": {
    "Fn::Select": [
      0,
      {
        "Fn::Cidr": [
          {
            "Fn::GetAtt": [
              "VPC",
              "CidrBlock"
            ]
          },
          2,
          12
        ]
      }
    ]
  },
  "MapPublicIpOnLaunch": {
    "Fn::If": [
      "IsPublic",
      true,
      false
    ]
  }
},
"Subnet2": {
  "Type": "AWS::EC2::Subnet",
  "Condition": "IsMultiAZ",
  "Properties": {
    "VpcId": {"Ref": "VPC"},
    "AvailabilityZone": {
      "Fn::Select": [
        1,
        {"Fn::GetAZs": ""}
      ]
    },
    "CidrBlock": {
      "Fn::Select": [
        1,
        {
          "Fn::Cidr": [

```

```

        {
            "Fn::GetAtt": [
                "VPC",
                "CidrBlock"
            ]
        },
        2,
        12
    ]
}
]
},
"MapPublicIpOnLaunch": {
    "Fn::If": [
        "IsPublic",
        true,
        false
    ]
}
},
"RouteTable": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
        "VpcId": {"Ref": "VPC"}
    }
},
"DefaultRoute": {
    "Type": "AWS::EC2::Route",
    "Condition": "IsPublic",
    "DependsOn": "InternetGatewayAttachment",
    "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {"Ref": "InternetGateway"}
    }
},
"Subnet1RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet1"}
    }
},

```

```

"Subnet2RouteTableAssociation": {
  "Type": "AWS::EC2::SubnetRouteTableAssociation",
  "Condition": "IsMultiAZ",
  "Properties": {
    "RouteTableId": {"Ref": "RouteTable"},
    "SubnetId": {"Ref": "Subnet2"}
  }
},
"InfluxDBSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupName": "influxdb-sg",
    "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
    "VpcId": {"Ref": "VPC"}
  }
},
"InfluxDBSecurityGroupIngress": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties": {
    "GroupId": {"Ref": "InfluxDBSecurityGroup"},
    "IpProtocol": "tcp",
    "CidrIp": "0.0.0.0/0",
    "FromPort": 8086,
    "ToPort": 8086
  }
},
"InfluxDBLogsS3Bucket": {
  "Type": "AWS::S3::Bucket",
  "DeletionPolicy": "Retain"
},
"InfluxDBLogsS3BucketPolicy": {
  "Type": "AWS::S3::BucketPolicy",
  "Properties": {
    "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "s3:PutObject",
          "Effect": "Allow",
          "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*"},
          "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
        }
      ]
    }
  }
},

```

```

        {
            "Action": "s3:*",
            "Effect": "Deny",
            "Resource": [
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
            ],
            "Principal": "*",
            "Condition": {
                "Bool": {"aws:SecureTransport": false}
            }
        }
    ]
}
},
"DbInstance": {
    "Type": "AWS::Timestream::InfluxDBInstance",
    "DependsOn": "InfluxDBLogsS3BucketPolicy",
    "Properties": {
        "DbStorageType": {"Ref": "DbStorageType"},
        "AllocatedStorage": {"Ref": "AllocatedStorage"},
        "DbInstanceType": {"Ref": "DbInstanceType"},
        "Name": {"Ref": "DbInstanceName"},
        "Username": {"Ref": "InfluxDBUsername"},
        "Password": {"Ref": "InfluxDBPassword"},
        "Organization": {"Ref": "InfluxDBOrganization"},
        "Bucket": {"Ref": "InfluxDBBucket"},
        "PubliclyAccessible": {
            "Fn::If": [
                "IsPublic",
                true,
                false
            ]
        },
        "DeploymentType": {"Ref": "DeploymentType"},
        "VpcSecurityGroupIds": [
            {"Ref": "InfluxDBSecurityGroup"}
        ],
        "VpcSubnetIds": {
            "Fn::If": [
                "IsMultiAZ",
                [
                    {"Ref": "Subnet1"},

```

```

        {"Ref": "Subnet2"}
      ],
      [
        {"Ref": "Subnet1"}
      ]
    ]
  },
  "LogDeliveryConfiguration": {
    "S3Configuration": {
      "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
      "Enabled": true
    }
  }
}
},
"Outputs": {
  "VPC": {
    "Description": "A reference to the VPC used to create network resources",
    "Value": {"Ref": "VPC"}
  },
  "Subnets": {
    "Description": "A list of the subnets created",
    "Value": {
      "Fn::If": [
        "IsMultiAZ",
        {
          "Fn::Join": [
            ",",
            [
              {"Ref": "Subnet1"},
              {"Ref": "Subnet2"}
            ]
          ]
        },
        {"Ref": "Subnet1"}
      ]
    }
  },
  "Subnet1": {
    "Description": "A reference to the subnet in the 1st Availability Zone",
    "Value": {"Ref": "Subnet1"}
  },
  "Subnet2": {

```



```

    "Condition": "IsMultiAZ",
    "Description": "A reference to the subnet in the 2nd Availability Zone",
    "Value": {"Ref": "Subnet2"}
  },
  "InfluxDBSecurityGroup": {
    "Description": "Security group with port 8086 ingress rule",
    "Value": {"Ref": "InfluxDBSecurityGroup"}
  },
  "InfluxDBLogsS3Bucket": {
    "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
    "Value": {"Ref": "InfluxDBLogsS3Bucket"}
  },
  "DbInstance": {
    "Description": "A reference to the Timestream for InfluxDB DB instance",
    "Value": {"Ref": "DbInstance"}
  },
  "InfluxAuthParametersSecretArn": {
    "Description": "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
containing the initial InfluxDB authorization parameters. The secret value is a JSON
formatted key-value pair holding InfluxDB authorization values: organization, bucket,
username, and password.",
    "Value": {
      "Fn::GetAtt": [
        "DbInstance",
        "InfluxAuthParametersSecretArn"
      ]
    }
  },
  "Endpoint": {
    "Description": "The endpoint URL to connect to InfluxDB",
    "Value": {
      "Fn::Join": [
        "",
        [
          "https://",
          {
            "Fn::GetAtt": [
              "DbInstance",
              "Endpoint"
            ]
          },
          ":8086"
        ]
      ]
    }
  }
}

```

```

    }
  }
}
}

```

YAML

```

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Network Configuration"
        Parameters:
          - VPCCIDR
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBUsername
          - InfluxDBPassword
          - InfluxDBOrganization
          - InfluxDBBucket
          - DbInstanceType
          - DbStorageType
          - AllocatedStorage
          - PubliclyAccessible
          - DeploymentType
    ParameterLabels:
      VPCCIDR:
        default: VPC CIDR

Parameters:
  # Network Configuration
  VPCCIDR:
    Description: Please enter the IP range (CIDR notation) for the new VPC
    Type: String
    Default: 10.0.0.0/16
  # Timestream for InfluxDB Configuration
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
    with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a

```

prefix included in the endpoint. DB instance names must be unique per customer and per Region.

Type: String

Default: mydbinstance

MinLength: 3

MaxLength: 40

AllowedPattern: `^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$`

InfluxDB initial user configurations

InfluxDBUsername:

Description: The username of the initial admin user created in InfluxDB. Must start with a letter and can't end with a hyphen or contain two consecutive hyphens. For example, my-user1. This username will allow you to access the InfluxDB UI to perform various administrative tasks and also use the InfluxDB CLI to create an operator token. These attributes will be stored in a Secret created in AWS Secrets Manager in your account.

Type: String

Default: admin

MinLength: 1

MaxLength: 64

InfluxDBPassword:

Description: The password of the initial admin user created in InfluxDB. This password will allow you to access the InfluxDB UI to perform various administrative tasks and also use the InfluxDB CLI to create an operator token. These attributes will be stored in a Secret created in AWS in your account.

Type: String

NoEcho: true

MinLength: 8

MaxLength: 64

AllowedPattern: `^[a-zA-Z0-9]+$`

InfluxDBOrganization:

Description: The name of the initial organization for the initial admin user in InfluxDB. An InfluxDB organization is a workspace for a group of users.

Type: String

Default: org

MinLength: 1

MaxLength: 64

InfluxDBBucket:

Description: The name of the initial InfluxDB bucket. All InfluxDB data is stored in a bucket. A bucket combines the concept of a database and a retention period (the duration of time that each data point persists). A bucket belongs to an organization.

Type: String

Default: bucket

MinLength: 2

MaxLength: 64

```
AllowedPattern: ^[^\"]*[^\"]*$
DeploymentType:
  Description: Specifies whether the Timestream for InfluxDB is deployed as Single-AZ
or with a MultiAZ Standby for High availability
  Type: String
  Default: WITH_MULTIAZ_STANDBY
  AllowedValues:
    - SINGLE_AZ
    - WITH_MULTIAZ_STANDBY
AllocatedStorage:
  Description: The amount of storage to allocate for your DB storage type in GiB
(gibibytes).
  Type: Number
  Default: 400
  MinValue: 20
  MaxValue: 16384
DbInstanceType:
  Description: The Timestream for InfluxDB DB instance type to run InfluxDB on.
  Type: String
  Default: db.influx.medium
  AllowedValues:
    - db.influx.medium
    - db.influx.large
    - db.influx.xlarge
    - db.influx.2xlarge
    - db.influx.4xlarge
    - db.influx.8xlarge
    - db.influx.12xlarge
    - db.influx.16xlarge
DbStorageType:
  Description: The Timestream for InfluxDB DB storage type to read and write InfluxDB
data.
  Type: String
  Default: InfluxIOIncludedT1
  AllowedValues:
    - InfluxIOIncludedT1
    - InfluxIOIncludedT2
    - InfluxIOIncludedT3
PubliclyAccessible:
  Description: Configures the DB instance with a public IP to facilitate access.
  Type: String
  Default: true
  AllowedValues:
    - true
```

- false

Conditions:

IsMultiAZ: !Equals [!Ref DeploymentType, WITH_MULTIAZ_STANDBY]

IsPublic: !Equals [!Ref PubliclyAccessible, true]

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VPCCIDR

InternetGateway:

Type: AWS::EC2::InternetGateway

Condition: IsPublic

InternetGatewayAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Condition: IsPublic

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

Subnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [0, !GetAZs '']

CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12]]

MapPublicIpOnLaunch: !If [IsPublic, true, false]

Subnet2:

Type: AWS::EC2::Subnet

Condition: IsMultiAZ

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [1, !GetAZs '']

CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12]]

MapPublicIpOnLaunch: !If [IsPublic, true, false]

RouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

DefaultRoute:

Type: AWS::EC2::Route

Condition: IsPublic

DependsOn: InternetGatewayAttachment

Properties:

```
    RouteTableId: !Ref RouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
Subnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref RouteTable
    SubnetId: !Ref Subnet1
Subnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Condition: IsMultiAZ
  Properties:
    RouteTableId: !Ref RouteTable
    SubnetId: !Ref Subnet2
InfluxDBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "influxdb-sg"
    GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
    VpcId: !Ref VPC
InfluxDBSecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !Ref InfluxDBSecurityGroup
    IpProtocol: tcp
    CidrIp: 0.0.0.0/0
    FromPort: 8086
    ToPort: 8086
InfluxDBLogsS3Bucket:
  Type: AWS::S3::Bucket
  DeletionPolicy: Retain
InfluxDBLogsS3BucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    Bucket: !Ref InfluxDBLogsS3Bucket
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Action: "s3:PutObject"
          Effect: Allow
          Resource: !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*
          Principal:
            Service: timestream-influxdb.amazonaws.com
        - Action: "s3:*"
```

```

    Effect: Deny
    Resource:
      - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/*
      - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}
    Principal: "*"
    Condition:
      Bool:
        aws:SecureTransport: false

```

DbInstance:

```

Type: AWS::Timestream::InfluxDBInstance
DependsOn: InfluxDBLogsS3BucketPolicy
Properties:
  DbStorageType: !Ref DbStorageType
  AllocatedStorage: !Ref AllocatedStorage
  DbInstanceType: !Ref DbInstanceType
  Name: !Ref DbInstanceName
  Username: !Ref InfluxDBUsername
  Password: !Ref InfluxDBPassword
  Organization: !Ref InfluxDBOrganization
  Bucket: !Ref InfluxDBBucket
  PubliclyAccessible: !If [IsPublic, true, false]
  DeploymentType: !Ref DeploymentType
  VpcSecurityGroupIds:
    - !Ref InfluxDBSecurityGroup
  VpcSubnetIds: !If
    - IsMultiAZ
    -
      - !Ref Subnet1
      - !Ref Subnet2
    -
      - !Ref Subnet1
  LogDeliveryConfiguration:
    S3Configuration:
      BucketName: !Ref InfluxDBLogsS3Bucket
      Enabled: true

```

Outputs:

Network Resources

VPC:

```

Description: A reference to the VPC used to create network resources
Value: !Ref VPC

```

Subnets:

```

Description: A list of the subnets created
Value: !If

```

- IsMultiAZ
- !Join [",", [!Ref Subnet1, !Ref Subnet2]]
- !Ref Subnet1

Subnet1:

Description: A reference to the subnet in the 1st Availability Zone

Value: !Ref Subnet1

Subnet2:

Condition: IsMultiAZ

Description: A reference to the subnet in the 2nd Availability Zone

Value: !Ref Subnet2

InfluxDBSecurityGroup:

Description: Security group with port 8086 ingress rule

Value: !Ref InfluxDBSecurityGroup

Timestream for InfluxDB Resources

InfluxDBLogsS3Bucket:

Description: S3 Bucket containing InfluxDB logs from the DB instance

Value: !Ref InfluxDBLogsS3Bucket

DbInstance:

Description: A reference to the Timestream for InfluxDB DB instance

Value: !Ref DbInstance

InfluxAuthParametersSecretArn:

Description: "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret containing the initial InfluxDB authorization parameters. The secret value is a JSON formatted key-value pair holding InfluxDB authorization values: organization, bucket, username, and password."

Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn

Endpoint:

Description: The endpoint URL to connect to InfluxDB

Value: !Join [""], ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]

CloudFormation を使用して Windows ベースのスタックをデプロイする

このページでは、Windows ベースのデプロイで一般的に使用される CloudFormation リソースの技術リファレンスドキュメントへのリンクを提供します。

CloudFormation は、Infrastructure as Code (IaC) を介して Microsoft Windows スタックをデプロイおよび管理するためのサポートを提供します。CloudFormation を使用すると、Windows ベースの EC2 インスタンス、Amazon RDS 上の SQL Server、および Directory Service を通じた Microsoft Active Directory を自動プロビジョニングできます。

AWS は、Windows プラットフォーム向けに設計された事前構成済みの Amazon マシンイメージ (AMI) を提供しており、Amazon EC2 上でアプリケーションを迅速にデプロイできます。これらの AMI には、Microsoft のデフォルト設定と AWS 固有のカスタマイズが含まれます。CloudFormation では、他の Windows Server と同様に、適切な AMI を選択し、インスタンスを起動してリモート デスクトップ接続を使用してアクセスできます。AMI には、EC2Launch (バージョンは Windows Server のエディションによって異なります)、AWS Systems Manager、CloudFormation、AWS Tools for PowerShell に加えて、ネットワーク、ストレージ、グラフィックス用の各種ドライバーなど、AWS サービスとの互換性と最適なパフォーマンスを確保するために必要なソフトウェアコンポーネントが含まれています。詳細については、「[AWS Windows AMI リファレンス](#)」を参照してください。

CloudFormation は、EC2 インスタンスの初回起動時に PowerShell またはバッチコマンドを実行できる UserData スクリプトなどのソフトウェア設定ツールもサポートしています。また、ヘルパー スクリプト (cfn-init、cfn-signal、cfn-get-metadata、cfn-hup) を提供し、Windows インスタンスのパッケージ、ファイル、サービスを管理するための AWS::CloudFormation::Init メタデータをサポートします。

エンタープライズ環境の場合、CloudFormation はドメイン結合、EC2 ライセンスモデルによる Windows ライセンス管理、AWS Secrets Manager による安全な認証情報処理を可能にします。CloudFormation は、バージョン管理されたテンプレートと反復可能なデプロイと組み合わせることで、組織が複数の AWS リージョン およびアカウントにわたって一貫性があり、安全で、スケーラブルな Windows 環境を維持するのに役立ちます。

Windows ベースのデプロイで一般的に使用される CloudFormation リソースの詳細については、以下の技術リファレンスピックを参照してください。

リソースタイプ	説明
AWS::EC2::Instance	Windows EC2 インスタンスを起動するためのリソースタイプ。
AWS::EC2::SecurityGroup	Windows ワークロードのファイアウォールルールを定義するためのリソースタイプ。
AWS::AutoScaling::AutoScalingGroup	Windows EC2 インスタンスをスケーリングするためのリソースタイプ。
AWS::EC2::LaunchTemplate	

リソースタイプ	説明
AWS::DirectoryService::MicrosoftAD	Microsoft Active Directory をデプロイするためのリソースタイプ。
AWS::FSx::FileSystem	FSx for Windows File Server をデプロイするためのリソースタイプ。
AWS::RDS::DBInstance	Amazon RDS の SQL Server をプロビジョニングするためのリソースタイプ。
AWS::CloudFormation::Init	インスタンスを設定するために EC2 メタデータ内で使用されます。 詳細については、「Windows ベースの CloudFormation スタックのブートストラップ」を参照してください。
AWS::SecretsManager::Secret	認証情報と Windows のパスワードを安全に管理するためのリソースタイプ。
AWS::SSM::Parameter	設定値を安全に保存するためのリソースタイプ。
AWS::IAM::InstanceProfile AWS::IAM::Role	EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与するためのリソースタイプ。

Windows ベースの CloudFormation スタックのブートストラップ

このトピックでは、Windows スタックをブートストラップする方法と、スタック作成時の問題をトラブルシューティングする方法について説明します。

トピック

- [EC2 インスタンスのユーザーデータ](#)
- [CloudFormation ヘルパースクリプト](#)
- [Windows スタックのブートストラップの例](#)
- [Windows ファイルパスのバックスラッシュをエスケープする](#)
- [Windows のサービスを管理する](#)
- [スタック作成の問題のトラブルシューティング](#)

EC2 インスタンスのユーザーデータ

ユーザーデータは、起動時にスクリプトまたは設定情報を EC2 インスタンスに渡すことができる Amazon EC2 機能です。

Windows EC2 インスタンスの場合:

- バッチスクリプト (<script> タグを使用) または PowerShell スクリプト (<powershell> タグを使用) を使用できます。
- スクリプトの実行は EC2Launch によって処理されます。

Important

CloudFormation で使用する独自の Windows AMI を作成する場合は、EC2Launch v2 が正しく設定されていることを確認してください。スタック作成時に Windows インスタンスを適切に初期化および設定するには、CloudFormation のブートストラップツールに EC2Launch v2 が必要です。詳細については、『Amazon EC2 ユーザーガイド』の「[EC2 Windows インスタンスの起動時に EC2Launch v2 エージェントを使用してタスクを実行する](#)」を参照してください。

AWS Windows AMI の詳細については、「[AWSWindows AMI リファレンス](#)」を参照してください。

CloudFormation ヘルパースクリプト

ヘルパースクリプトは、ブートストラッププロセス中にインスタンスを設定するためのユーティリティです。Amazon EC2 ユーザーデータで使用され、強力な設定オプションを提供します。

CloudFormation には、スタックの一部として作成する Amazon EC2 インスタンスでソフトウェアをインストールしたりサービスを開始したりするために使用できる、次の Python ヘルパースクリプトが用意されています。

- `cfn-init` – リソースメタデータの取得と解釈、パッケージのインストール、ファイルの作成、およびサービスの開始で使します。
- `cfn-signal` – `CreationPolicy` でシグナルを送信するために使用し、前提となるリソースやアプリケーションの準備ができたときに、スタックの他のリソースを同期できるようにします。
- `cfn-get-metadata` – 特定のキーへのリソースまたはパスのメタデータを取得するために使します。

- `cfn-hup` – メタデータへの更新を確認し、変更が検出されたときにカスタムフックを実行するために使用します。

テンプレートからスクリプトを直接呼び出します。スクリプトは、同じテンプレートで定義されているリソースメタデータと組み合わせられて機能します。スクリプトは Amazon EC2 インスタンスでスタック作成プロセス中に実行されます。

詳細については、「CloudFormation テンプレートリファレンスガイド」の「[CloudFormation ヘルパースクリプトリファレンス](#)」を参照してください。

Windows スタックのブートストラップの例

次のアクションを実行する Windows Server テンプレートのサンプルスニペットを見てみましょう。

- `TestInstance` という名前の EC2 インスタンスを Windows Server 2022 AMI から起動します。
- `cfn-init` が動作していることを確認するためのシンプルなテストファイルを作成します。
- `cfn-hup` を継続的な設定管理のために設定します。
- `CreationPolicy` を使用して、インスタンスが正常に完了したというシグナルを送信するようにします。

`cfn-init` のヘルパースクリプトを使用して、テンプレートの `AWS::CloudFormation::Init` リソースの情報に基づいて、これらの各アクションを実行します。

`AWS::CloudFormation::Init` セクションは `TestInstance` と呼ばれ、次の宣言で始まります。

```
TestInstance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      configSets:
        default:
          - create_files
          - start_services
```

この後、`AWS::CloudFormation::Init` の `files` セクションが宣言されます。

```
create_files:
  files:
```

```
c:\cfn\test.txt:
content: !Sub |
    Hello from ${AWS::StackName}
c:\cfn\cfn-hup.conf:
content: !Sub |
    [main]
    stack=${AWS::StackName}
    region=${AWS::Region}
    interval=2
c:\cfn\hooks.d\cfn-auto-reloader.conf:
content: !Sub |
    [cfn-auto-reloader-hook]
    triggers=post.update
    path=Resources.TestInstance.Metadata.AWS::CloudFormation::Init
    action=cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --
region ${AWS::Region}
```

3つのファイルがここで作成され、サーバーインスタンスのC:\cfnディレクトリに配置されます。

- test.txt は、cfn-init が正しく動作していることを検証するシンプルなテストファイルであり、動的コンテンツを含むファイルを作成できます。
- cfn-hup.conf は、cfn-hup 用の設定ファイルで、チェック間隔は 2 分間です。
- cfn-auto-reloader.conf、AWS::CloudFormation::Init のメタデータが変更されるときに cfn-hup が更新 (cfn-init の呼び出し) を開始するために使用するフックの設定ファイル。

次に、Windows サービスを設定する start_services セクションです。

```
start_services:
  services:
    windows:
      cfn-hup:
        enabled: true
        ensureRunning: true
        files:
          - c:\cfn\cfn-hup.conf
          - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

このセクションでは、cfn-hup サービスが開始され、設定ファイルが変更されると自動的に再起動します。サービスは CloudFormation メタデータの変更をモニタリングし、更新が検出されると cfn-init を再実行します。

次は Properties セクションです。

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Metadata:
    AWS::CloudFormation::Init:
      # ... metadata configuration ...
  Properties:
    InstanceType: t2.large
    ImageId: '{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-Base}}'
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    KeyName: !Ref KeyPairName
    UserData:
      Fn::Base64: !Sub |
        <powershell>
          cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --region
${AWS::Region}
          cfn-signal.exe -e $lastexitcode --stack ${AWS::StackName} --resource
TestInstance --region ${AWS::Region}
        </powershell>
```

このセクションでは、UserData プロパティに、PowerShell が実行する EC2Launch が含まれています。この部分は <powershell> タグで囲まれています。スクリプトは default configSet を使用して cfn-init を実行し、cfn-signal を使用して終了コードを CloudFormation に報告します。CreationPolicy は、スタックの作成が完了と見なされる前にインスタンスが適切に設定されていることを確認するために使用されます。

ImageId プロパティは、Systems Manager パラメータストアのパブリックパラメータを使用して、最新の Windows Server 2022 AMI ID を自動的に取得します。このアプローチにより、リージョン固有の AMI マッピングが不要になり、常に最新の AMI を取得できます。Systems Manager パラメータを AMI ID に使用することは、最新の AMI リファレンスを維持するためのベストプラクティスです。インスタンスに接続する場合は、SecurityGroupIds プロパティが RDP アクセスを許可するセキュリティグループを参照していることを確認してください。

CreationPolicy はリソースプロパティの一部として宣言され、タイムアウト期間を指定します。インスタンス設定が完了すると、ユーザーデータ内の cfn-signal コマンドがシグナルを送信します。

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Properties:
    # ... other properties ...
```

ブートストラッピングプロセスは最小限で、ファイルの作成とサービスの開始のみを行うため、CreationPolicy はタイムアウトするまで 20 分 (PT20M) 待機します。タイムアウトは ISO 8601 期間形式を使用して指定します。Windows インスタンスの起動には通常 Linux インスタンスよりも時間がかかるため、徹底的にテストしてニーズに最適なタイムアウト値を判断してください。

すべて問題がなければ、CreationPolicy は正常に完了し、パブリック IP アドレスを使用して Windows Server インスタンスにアクセスできます。スタックの作成が完了すると、インスタンス ID と IP アドレスが CloudFormation コンソールの [出力] タブに表示されます。

```
Outputs:
  InstanceId:
    Value: !Ref TestInstance
    Description: Instance ID of the Windows Server
  PublicIP:
    Value: !GetAtt TestInstance.PublicIp
    Description: Public IP address of the Windows Server
```

RDP 経由でインスタンスに接続し、ファイル C:\cfn\test.txt が存在し、予想されるコンテンツが含まれていることを確認することで、ブートストラッピングが正しく機能したことを手動で確認することもできます。Windows インスタンスへの接続の詳細については、「Amazon EC2 ユーザーガイド」の「[Connect to your Windows instance using RDP](#)」を参照してください。

Windows ファイルパスのバックスラッシュをエスケープする

CloudFormation テンプレートで Windows パスを参照するときは、使用しているテンプレート形式に従ってバックスラッシュ (\) を適切にエスケープするようにしてください。

- JSON テンプレートでは、JSON はバックスラッシュをエスケープ文字として扱うため、Windows ファイルパスにダブルバックスラッシュを使用する必要があります。最初のバックスラッシュは 2

番目のバックスラッシュをエスケープするため、単一のリテラルバックスラッシュとして解釈されます。

```
"commands" : {
  "1-extract" : {
    "command" : "C:\\\\SharePoint\\\\SharePointFoundation2010.exe /extract:C:\\\\SharePoint
\\\\SPF2010 /quiet /log:C:\\\\SharePoint\\\\SharePointFoundation2010-extract.log"
  }
}
```

- YAML テンプレートの場合、通常、1 つのバックスラッシュで十分です。

```
commands:
  1-extract:
    command: C:\SharePoint\SharePointFoundation2010.exe /extract:C:\SharePoint
\SPF2010 /quiet /log:C:\SharePoint\SharePointFoundation2010-extract.log
```

Windows のサービスを管理する

Linux サービスと同じように Windows サービスを管理しますが、sysvinit の代わりに、windows キーを使用します。次の例では、cfn-hup サービスを開始し、それを Automatic に設定します。c:\cfn\cfn-hup.conf または c:\cfn\hooks.d\cfn-auto-reloader.conf 設定ファイルを cfn-init が変更する場合は、サービスを再起動します。

```
services:
  windows:
    cfn-hup:
      enabled: true
      ensureRunning: true
      files:
        - c:\cfn\cfn-hup.conf
        - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

サービスを参照する名前 (表示名ではない) を使用して、同じ方法で他の Windows サービスを管理できます。

スタック作成の問題のトラブルシューティング

スタックの作成に失敗する場合、デフォルトでは失敗時にロールバックが実行されます。不必要な課金を避けられるため、通常はこのデフォルト設定が適切ですが、スタック作成が失敗する原因をデバッグすることは難しくなります。

CloudFormation コンソールでスタックを作成または更新するときにこの動作をオフにするには、[スタックの失敗オプション] で [正常にプロビジョニングされたリソースの保持] オプションを選択します。詳細については、「[リソースのプロビジョニング時における失敗への対応方法を選択する](#)」を参照してください。これにより、インスタンスにログインし、ログファイルを確認できるようになり、スタートアップスクリプト実行時に発生した問題を特定できます。

調査対象の重要なログは次のとおりです。

- %ProgramData%\Amazon\EC2Launch\log\agent.log にある EC2 構成ログ
- C:\cfn\log\cfn-init.log の cfn-init ログ (特定の障害ポイントの終了コードとエラーメッセージを確認する)

その他のログについては、「Amazon EC2 ユーザーガイド」の次のトピックを参照してください。

- [EC2Launch ディレクトリ構造](#)
- [EC2Launch v2 ディレクトリ構造](#)

ブートストラッピングの問題のトラブルシューティングの詳細については、「[Windows インスタンスがある CloudFormation スタックでブートストラップしないヘルパースクリプトをトラブルシューティングするにはどうすればよいですか?](#)」を参照してください。

CloudFormation が提供するリソースタイプの使用によるテンプレートの機能の拡張

CloudFormation はスタックテンプレートで利用できるいくつかのリソースタイプを提供し、シンプルなスタックテンプレートを超えて機能を拡張できます。

これらのリソースタイプには以下が含まれます。

リソースタイプ	説明	ドキュメント
カスタムリソース	この <code>AWS::CloudFormation::CustomResource</code> リソースタイプは、特定のプロビジョニングタスクを実行するカスタムリソースを作成したり、CloudFormation リソースタイプとして利用することができないリソースを含めたりすることを可能にします。	カスタムリソース
マクロ	この <code>AWS::CloudFormation::Macro</code> リソースタイプは、CloudFormation テンプレートでカスタム処理を実行できる再利用可能なコードを定義します。マクロは、スタックの作成中または更新中にテンプレートを変更する、追加のリソースを生成する、またはその他のカスタムオペレーションを実行することができます。	マクロのテンプレート
ネストされたスタック	この <code>AWS::CloudFormation::Stack</code> リソースタイプは、よりモジュール化された再利用可能なスタックアーキテクチャのために、CloudFormation テンプレート内でネストされたスタックを作成できるようにします。	ネストされたスタック
StackSet	この <code>AWS::CloudFormation::StackSet</code> リソースタイプは、CloudFormation StackSet を作成または更新します。StackSet は、複数の AWS アカウント およびリージョン全体にデプロイできる、スタック用のコンテナです。	StackSets を使用したスタックの管理
待機条件	この <code>AWS::CloudFormation::WaitCondition</code> リソースタイプは、特定の条件 (長時間実行されるプロセスの正常な完了、または外部リソースの可用性など) が満たされるまで、スタックの作成または更新を一時停止します。	待機条件

リソースタイプ	説明	ドキュメント
待機条件ハンドル	この <code>AWS::CloudFormation::WaitConditionHandle</code> リソースタイプは、 <code>AWS::CloudFormation::WaitCondition</code> リソースタイプとともに動作します。特定の条件が満たされたことを示すシグナルを送信するために使用される署名付き URL を提供します。これらのシグナルにより、スタックの作成または更新プロセスを続行できます。	待機条件

カスタムリソースを使用してカスタムプロビジョニングロジックを作成する

カスタムリソースを使用すると、CloudFormation テンプレートにカスタムプロビジョニングロジックを記述し、スタックを作成、更新 (カスタムリソースを変更した場合)、または削除するたびに CloudFormation にそのロジックを実行させることができます。これは、プロビジョニング要件に、CloudFormation の組み込みリソースタイプでは表現できない複雑なロジックやワークフローが含まれる場合に役立ちます。

例えば、CloudFormation のリソースタイプとして使用できないリソースを含める必要があるとします。それらのリソースは、カスタムリソースを使用して含めることができます。この方法により、すべての関連リソースを 1 つのスタックで管理できます。

CloudFormation テンプレートでカスタムリソースを定義するには、`AWS::CloudFormation::CustomResource` または `Custom::MyCustomResourceTypeName` リソースタイプを使用します。カスタムリソースにはサービストークンを示す 1 つのプロパティが必要です。このプロパティは、CloudFormation がリクエストを送信する宛先 (Amazon SNS トピックまたは Lambda 関数など) を指定します。

次のトピックでは、カスタムリソースの使用方法について説明します。

トピック

- [カスタムリソースのしくみ](#)
- [応答タイムアウト](#)

- [CloudFormation カスタムリソースのリクエストおよびレスポンスのリファレンス](#)
- [Amazon SNS を利用するカスタムリソース](#)
- [Lambda を使用するカスタムリソース](#)

Note

CloudFormation レジストリとカスタムリソースには、それぞれ独自の利点があります。カスタムリソースには次の利点があります。

- リソースを登録する必要はありません。
- リソース全体を登録せずにテンプレートの一部として含めることができます。
- Create、Update、および Delete 操作をサポート

レジストリベースのリソースが提供する利点は次のとおりです。

- サードパーティー製アプリケーションリソースのモデリング、プロビジョニング、および管理をサポート
- Create、Read、Update、Delete、および List (CRUDL) 操作をサポート
- プライベートおよびサードパーティーのリソースタイプでドリフト検出をサポート

カスタムリソースとは異なり、レジストリベースのリソースは、CRUDL 操作を実行するために Amazon SNS トピックや Lambda 関数を関連付ける必要がありません。詳細については、「[CloudFormation レジストリによる拡張機能の管理](#)」を参照してください。

カスタムリソースのしくみ

新しいカスタムリソースを設定する一般的なプロセスには、次の手順が含まれます。これらの手順には、カスタムリソースを所有するカスタムリソースプロバイダーと、カスタムリソースタイプを含むテンプレートを作成するテンプレート開発者の 2 つの役割が必要です。これは同じユーザーでもかまいませんが、そうでない場合は、カスタムリソースプロバイダーがテンプレート開発者と連携する必要があります。

1. カスタムリソースプロバイダーは、CloudFormation からのリクエストを処理し、カスタムリソースに対してアクションを実行する方法を決定するロジックを書き込みます。

2. カスタムリソースプロバイダーは、CloudFormation がリクエストを送信できる Amazon SNS トピックまたは Lambda 関数を作成します。Amazon SNS トピックまたは Lambda 関数は、スタックを作成するリージョンと同じリージョンに存在する必要があります。
3. カスタムリソースプロバイダーは、Amazon SNS トピック ARN または Lambda 関数 ARN をテンプレート開発者に提供します。
4. テンプレート開発者は、CloudFormation テンプレートでカスタムリソースを定義します。これには、サービストークンと入力データパラメータが含まれます。サービストークンや入力データの構造は custom resource provider によって定義されます。サービストークンは Amazon SNS トピック ARN または Lambda 関数 ARN を指定し、常に必須ですが、入力データはカスタムリソースに応じてオプションです。

テンプレートを使用してカスタムリソースを作成、更新、削除するたびに、CloudFormation は指定されたサービストークンにリクエストを送信し、スタック操作を続行する前に、応答を待機します。

テンプレートからスタックを作成するためのフローを以下に示します。

1. CloudFormation は、指定されたサービストークンにリクエストを送信します。このリクエストには、リクエストタイプおよびカスタムリソースが応答を送信する宛先の Amazon S3 バケットの署名付き URL などの情報が含まれています。リクエストに含まれている情報の詳細については、「[CloudFormation カスタムリソースのリクエストおよびレスポンスのリファレンス](#)」を参照してください。

以下のサンプルデータは、CloudFormation が Create リクエストに含める内容を示しています。

この例では、ResourceProperties は CloudFormation が Lambda 関数へ送信するカスタムペイロードの作成を許可します。

```
{
  "RequestType" : "Create",
  "RequestId" : "unique id for this create request",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "ResponseURL" : "http://pre-signed-S3-url-for-response",
  "ResourceType" : "Custom::TestResource",
  "LogicalResourceId" : "MyTestResource",
  "ResourceProperties" : {
    "Name" : "Value",
    "List" : [ "1", "2", "3" ]
  }
}
```

```
}
```

2. custom resource provider は CloudFormation のリクエストを処理し、SUCCESS または FAILED の応答を署名付き URL に返します。custom resource provider は、応答を JSON 形式のファイルで提供し、それを署名付き S3 URL にアップロードします。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[署名付き URL を使用したオブジェクトのアップロード](#)」を参照してください。

その応答について、custom resource provider には template developer がアクセスできる名前と値のペアも含まれている場合があります。たとえば、応答には要求が成功した場合は出力データ、要求が失敗した場合はエラーメッセージが含まれることがあります。応答の詳細については、「[CloudFormation カスタムリソースのリクエストおよびレスポンスのリファレンス](#)」を参照してください。

Important

名前と値のペアに機密情報が含まれている場合は、NoEcho フィールドを使用して、カスタムリソースの出力をマスクします。それ以外の場合は、プロパティ値 (DescribeStackEvents など) を表示する API を通じて値が表示されます。

NoEcho を使用して機密情報をマスクする方法、および動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」ベストプラクティスを参照してください。

custom resource provider には要求をリッスンして応答する責任があります。例えば、Amazon SNS 通知の場合、custom resource provider は特定のトピック ARN に送信される通知をリッスンして応答する必要があります。CloudFormation は、事前署名済みの URL の場所で応答を待機してリッスンします。

次のサンプルデータは、カスタムリソースの応答に含まれる可能性がある情報を示します。

```
{
  "Status" : "SUCCESS",
  "RequestId" : "unique id for this create request",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "LogicalResourceId" : "MyTestResource",
  "PhysicalResourceId" : "TestResource1",
  "Data" : {
```

```
"OutputName1" : "Value1",
"OutputName2" : "Value2",
}
}
```

3. SUCCESS 応答を取得した後、CloudFormation はスタックオペレーションを続けます。FAILED 応答が返されるか、応答が返されない場合、操作は失敗します。カスタムリソースからの出力データは署名付き URL の場所に保存されます。テンプレートデベロッパーは [Fn::GetAtt](#) 関数を使用してそのデータを取得できます。

Note

AWS PrivateLink を使用する場合は、VPC のカスタムリソースには CloudFormation 固有の S3 バケットへのアクセスが必要です。カスタムリソースは、署名付き Amazon S3 URL に応答を送信する必要があります。Amazon S3 に応答を送信できない場合、CloudFormation は応答を受信せず、スタックオペレーションは失敗となります。詳細については、「[インターフェイスエンドポイントを使用した CloudFormation へのアクセス \(AWS PrivateLink\)](#)」を参照してください。

応答タイムアウト

カスタムリソースのデフォルトのタイムアウトは 3600 秒 (1 時間) です。この間に応答が受信されない場合、スタック操作は失敗します。

タイムアウト値は、カスタムリソースからの応答にかかると予想される時間に基づいて調整できます。例えば、5 分以内に応答することが予想される Lambda 関数を呼び出すカスタムリソースをプロビジョニングする場合、ServiceTimeout プロパティを指定してスタックテンプレートで 5 分のタイムアウトを設定できます。詳細については、「[CloudFormation カスタムリソースのリクエストおよびレスポンスのリファレンス](#)」を参照してください。これにより、Lambda 関数にエラーが発生してスタックが停止した場合、CloudFormation は 1 時間待機するのではなく 5 分後にスタック操作を失敗させます。

ただし、タイムアウト値を低く設定しすぎないように注意してください。予期しないタイムアウトを回避するには、カスタムリソースに必要なアクションを実行して応答を返すのに十分な時間があることを確認してください。

CloudFormation カスタムリソースのリクエストおよびレスポンスのリファレンス

CloudFormation は、カスタムリソースプロバイダーと通信するリクエスト/レスポンスプロトコルを使用してカスタムリソースを管理します。各リクエストにはリクエストタイプ (Create、Update、または Delete) が含まれ、次の高レベルのワークフローに従います。

1. テンプレート開発者は、テンプレート内の `ServiceToken` と `ServiceTimeout` を使用してカスタムリソースを定義し、スタックオペレーションを開始します。
2. CloudFormation は、JSON リクエストを SNS または Lambda を介してカスタムリソースプロバイダーに送信します。
3. カスタムリソースプロバイダーはリクエストを処理し、タイムアウト期間が終了する前に JSON レスポンスを署名付き Amazon S3 バケット URL に返します。
4. CloudFormation はレスポンスを読み取り、スタックオペレーションを続行します。タイムアウト期間が終了する前に応答を受信しない場合、リクエストは不成功と見なされ、スタック操作は失敗します。

詳細については、「[カスタムリソースのしくみ](#)」を参照してください。

このセクションでは、各リクエストタイプの構造、パラメータ、および予想されるレスポンスについて説明します。

Note

レスポンス本文の合計サイズは 4,096 バイトを超えることはできません。

テンプレートのセットアップ

テンプレートでカスタムリソースを定義する場合、テンプレート開発者は次のプロパティで [AWS::CloudFormation::CustomResource](#) を使用します。

ServiceToken

スタックと同じリージョンからの Amazon SNS トピック ARN または Lambda 関数 ARN。

必須: はい

タイプ: 文字列

ServiceTimeout

カスタムリソースオペレーションがタイムアウトするまでの最大時間 (秒単位)。1 ~ 3600 の範囲の値にする必要があります。デフォルト: 3600 秒 (1 時間)。

必須: いいえ

タイプ: 文字列

追加のリソースプロパティがサポートされています。リソースプロパティは、リクエストに `ResourceProperties` として含まれます。カスタムリソースプロバイダーは、有効なプロパティとそれらの許容値を決定する必要があります。

オブジェクトをリクエストする

Create

テンプレート開発者がカスタムリソースを含むスタックを作成すると、CloudFormation は `RequestType` を `Create` に設定してリクエストを送信します。

作成リクエストには、以下のフィールドが含まれています。

RequestType

Create.

必須: はい

タイプ: 文字列

RequestId

リクエストの一意の ID。

`StackId` と `RequestId` を組み合わせると、特定のカスタムリソースに対するリクエストを一意に識別するために使用できる値が形成されます。

必須: はい

タイプ: 文字列

StackId

カスタムリソースを含むスタックを識別する Amazon リソースネーム (ARN)。

StackId と RequestId を組み合わせると、特定のカスタムリソースに対するリクエストを一意に識別するために使用できる値が形成されます。

必須: はい

タイプ: 文字列

ResponseURL

応答 URL は、カスタムリソースプロバイダーから CloudFormation への応答を受信する署名済み S3 バケットを識別します。

必須: はい

タイプ: 文字列

ResourceType

CloudFormation テンプレート内のカスタムリソースに対し、テンプレートのデベロッパーが選択したリソースタイプ。カスタムリソースタイプの名前は、60 文字までの長さで指定できます。また、英数字や記号 _@- を含めることができます。

必須: はい

タイプ: 文字列

LogicalResourceId

CloudFormation テンプレートで開発者が選択したカスタムリソースの名前 (論理 ID)。

必須: はい

タイプ: 文字列

ResourceProperties

このフィールドにはテンプレート開発者によって送信された Properties オブジェクトの内容が含まれます。その内容は、custom resource provider によって定義されます。

必須: いいえ

タイプ: JSON オブジェクト

例

```
{
```

```
"RequestType" : "Create",
"RequestId" : "unique-request-id",
"StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
"ResponseURL" : "pre-signed-url-for-create-response",
"ResourceType" : "Custom::MyCustomResourceType",
"LogicalResourceId" : "resource-logical-id",
"ResourceProperties" : {
  "key1" : "string",
  "key2" : [ "list" ],
  "key3" : { "key4" : "map" }
}
}
```

Update

テンプレート開発者がテンプレート内のカスタムリソースのプロパティを変更し、スタックを更新すると、CloudFormation は RequestType を Update に設定して custom resource provider にリクエストを送信します。カスタムリソースコードは、リクエスト タイプが Update のときにプロパティが変更されたことを認識しているため、リソースの変更を検出する必要がないということ、これは意味します。

更新リクエストには、以下のフィールドが含まれています。

RequestType

Update.

必須: はい

タイプ: 文字列

RequestId

リクエストの一意の ID。

StackId と RequestId を組み合わせると、特定のカスタムリソースに対するリクエストを一意に識別するために使用できる値が形成されます。

必須: はい

タイプ: 文字列

StackId

カスタムリソースを含むスタックを識別する Amazon リソースネーム (ARN)。

StackId と RequestId を組み合わせると、特定のカスタムリソースに対するリクエストを一意に識別するために使用できる値が形成されます。

必須: はい

タイプ: 文字列

ResponseURL

応答 URL は、カスタムリソースプロバイダーから CloudFormation への応答を受信する署名済み S3 バケットを識別します。

必須: はい

タイプ: 文字列

ResourceType

CloudFormation テンプレート内のカスタムリソースに対し、テンプレートのデベロッパーが選択したリソースタイプ。カスタムリソースタイプの名前は、60 文字までの長さで指定できます。また、英数字や記号 _@- を含めることができます。更新時にタイプを変更することはできません。

必須: はい

タイプ: 文字列

LogicalResourceId

CloudFormation テンプレートで開発者が選択したカスタムリソースの名前 (論理 ID)。

必須: はい

タイプ: 文字列

PhysicalResourceId

カスタムリソースプロバイダーによって定義された、プロバイダーで一意となる物理 ID。

必須: はい

タイプ: 文字列

ResourceProperties

このフィールドにはテンプレート開発者によって送信された Properties オブジェクトの内容が含まれます。その内容は、custom resource provider によって定義されます。

必須: いいえ

タイプ: JSON オブジェクト

OldResourceProperties

Update リクエストにのみ使用されます。CloudFormation テンプレート内のテンプレートデベロッパーによって以前に宣言されたリソースプロパティ値。

必須: はい

タイプ: JSON オブジェクト

例

```
{
  "RequestType" : "Update",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
  "ResponseURL" : "pre-signed-url-for-update-response",
  "ResourceType" : "Custom::MyCustomResourceType",
  "LogicalResourceId" : "resource-logical-id",
  "PhysicalResourceId" : "provider-defined-physical-id",
  "ResourceProperties" : {
    "key1" : "new-string",
    "key2" : [ "new-list" ],
    "key3" : { "key4" : "new-map" }
  },
  "OldResourceProperties" : {
    "key1" : "string",
    "key2" : [ "list" ],
    "key3" : { "key4" : "map" }
  }
}
```

Delete

テンプレート開発者がスタックを削除するか、テンプレートからカスタムリソースを削除してからスタックを更新すると、CloudFormation は RequestType を Delete に設定してリクエストを送信します。

削除リクエストには、以下のフィールドが含まれています。

RequestType

Delete.

必須: はい

タイプ: 文字列

RequestId

リクエストの一意的 ID。

必須: はい

タイプ: 文字列

StackId

カスタムリソースを含むスタックを識別する Amazon リソースネーム (ARN)。

必須: はい

タイプ: 文字列

ResponseURL

応答 URL は、カスタムリソースプロバイダーから CloudFormation への応答を受信する署名済み S3 バケットを識別します。

必須: はい

タイプ: 文字列

ResourceType

CloudFormation テンプレート内のカスタムリソースに対し、テンプレートのデベロッパーが選択したリソースタイプ。カスタムリソースタイプの名前は、60 文字までの長さで指定できます。また、英数字や記号 `_@-` を含めることができます。

必須: はい

タイプ: 文字列

LogicalResourceId

CloudFormation テンプレートで開発者が選択したカスタムリソースの名前 (論理 ID)。

必須: はい

タイプ: 文字列

PhysicalResourceId

カスタムリソースプロバイダーによって定義された、プロバイダーで一意となる物理 ID。

必須: はい

タイプ: 文字列

ResourceProperties

このフィールドにはテンプレート開発者によって送信された Properties オブジェクトの内容が含まれます。その内容は、custom resource provider によって定義されます。

必須: いいえ

タイプ: JSON オブジェクト

例

```
{
  "RequestType" : "Delete",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
  "ResponseURL" : "pre-signed-url-for-delete-response",
  "ResourceType" : "Custom::MyCustomResourceType",
  "LogicalResourceId" : "resource-logical-id",
  "PhysicalResourceId" : "provider-defined-physical-id",
  "ResourceProperties" : {
    "key1" : "string",
    "key2" : [ "list" ],
    "key3" : { "key4" : "map" }
  }
}
```

レスポンスオブジェクト

カスタムリソースプロバイダーは、レスポンスをすべてのリクエストタイプの署名付き URL に送信します。カスタムリソースプロバイダーがレスポンスを送信しない場合、CloudFormation はオペレーションがタイムアウトするまで待機します。

レスポンスは、次のフィールドを持つ JSON オブジェクトである必要があります。

Status

SUCCESS または FAILED である必要があります。

必須: はい

タイプ: 文字列

RequestId

リクエストの一意の ID。リクエストに表示されるように、この値を正確にコピーします。

必須: はい

タイプ: 文字列

StackId

カスタムリソースを含むスタックを識別する Amazon リソースネーム (ARN)。リクエストに表示されるように、この値を正確にコピーします。

必須: はい

タイプ: 文字列

LogicalResourceId

CloudFormation テンプレートで開発者が選択したカスタムリソースの名前 (論理 ID)。リクエストに表示されるように、この値を正確にコピーします。

必須: はい

タイプ: 文字列

PhysicalResourceId

この値は、カスタムリソースベンダーに固有の識別子である必要があります、サイズは最大 1 KB までです。値は空でない文字列でなければならず、同じリソースに対するすべての応答で同一である必要があります。

カスタムリソースを更新する場合、PhysicalResourceId に返される値は更新動作を決定します。値が同じままの場合、CloudFormation はこれを通常の更新と見なします。値が変わっている場合には、CloudFormation は新しい方が更新用のものであると解釈し、古いリソースに削除リクエストを送信します。詳細については、「[AWS::CloudFormation::CustomResource](#)」を参照してください。

必須: はい

タイプ: 文字列

Reason

失敗応答の理由を説明します。

Status が FAILED の場合は必須です。それ以外の場合はオプションです。

必須: 条件に応じて異なります

タイプ: 文字列

NoEcho

Fn::GetAtt 関数を使用して取得したときに、カスタムリソースの出力をマスクするかどうかを示します。true に設定すると、テンプレートの *Metadata* セクションに保存されているものを除き、すべての戻り値はアスタリスク (*****) でマスクされます。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。デフォルト値は false です。

NoEcho を使用して機密情報をマスクする方法、および動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」ベストプラクティスを参照してください。

Create および Update レスポンスでのみ使用できます。Delete レスポンスではサポートされていません。

必須: いいえ

型: ブール

Data

レスポンスで送信するカスタムリソースプロバイダー定義の名前と値のペア。ここで提供されている値には、Fn::GetAtt を使用してテンプレートの名前でアクセスできます。

Create および Update レスポンスでのみ使用できます。Delete レスポンスではサポートされていません。

Important

名前と値のペアに機密情報が含まれている場合は、NoEcho フィールドを使用して、カスタムリソースの出力をマスクします。それ以外の場合は、プロパティ値 (DescribeStackEvents など) を表示する API を通じて値が表示されます。

必須: いいえ

タイプ: JSON オブジェクト

成功したレスポンスの例

Create と Update レスポンス

```
{
  "Status": "SUCCESS",
  "RequestId": "unique-request-id",
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
  "LogicalResourceId": "resource-logical-id",
  "PhysicalResourceId": "provider-defined-physical-id",
  "NoEcho": true,
  "Data": {
    "key1": "value1",
    "key2": "value2"
  }
}
```

Delete 応答

```
{
  "Status": "SUCCESS",
  "RequestId": "unique-request-id",
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
  "LogicalResourceId": "resource-logical-id",
  "PhysicalResourceId": "provider-defined-physical-id"
}
```

失敗したレスポンスの例

```
{
  "Status": "FAILED",
  "RequestId": "unique-request-id",
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
  "LogicalResourceId": "resource-logical-id",
  "PhysicalResourceId": "provider-defined-physical-id",
  "Reason": "Required failure reason string"
}
```

Amazon SNS を利用するカスタムリソース

次のトピックでは、CloudFormation がリクエストを送信する Amazon SNS トピックを指定するサービストークンを使用してカスタムリソースを設定する方法を示します。また、カスタムリソーススタックの作成、更新、削除の結果として送受信されるイベントとメッセージのシーケンスについても学習します。

カスタムリソースと Amazon SNS を使用すると、新しいリソースのスタックへの追加、スタックへの動的データの挿入などのシナリオが可能になります。例えば、スタックを作成する際に、CloudFormation は、Amazon EC2 インスタンスで実行しているアプリケーションによってモニタリングされるトピックに Create リクエストを送信できます。Amazon SNS 通知は、許可リストされた Elastic IP アドレスのプールの取得など、追加のプロビジョニングタスクを実行するようにアプリケーションをトリガーします。完了すると、アプリケーションは、CloudFormation にスタックオペレーションの続行を通知する応答 (および該当する場合は出力データ) を送信します。

カスタムリソースのターゲットとして Amazon SNS トピックを指定すると、CloudFormation はカスタムリソースに関連するスタックオペレーション中に指定された SNS トピックにメッセージを送信します。これらのメッセージを処理して必要なアクションを実行するには、サポートされているエンドポイントが SNS トピックをサブスクライブするようになる必要があります。

カスタムリソースとその仕組みの概要については、「[カスタムリソースのしくみ](#)」を参照してください。Amazon SNS とその仕組みの詳細については、「[Amazon Simple Notification Service デベロッパーガイド](#)」を参照してください。

Amazon SNS を利用してカスタムリソースを作成する

トピック

- [ステップ 1: スタックの作成](#)
- [ステップ 2: スタックの更新](#)
- [ステップ 3: スタックの削除](#)

ステップ 1: スタックの作成

1. テンプレートデベロッパーは、カスタムリソースを含む CloudFormation スタックを作成します。

以下のテンプレート例では、論理 ID *MySeleniumTest* を持つカスタムリソースにカスタムリソースタイプ名 *Custom::SeleniumTester* を使用します。カスタムリソースタイプの名前は、英数字で 60 文字までの長さにする必要があります。

カスタムリソースのタイプとともに、カスタムリソースプロバイダーによって定義された情報として、サービストークン、オプションのプロバイダー固有のプロパティ、オプションの `Fn::GetAtt` 属性が宣言されています。これらのプロパティと属性を使用して、template developer から custom resource provider へ、または custom resource provider から template developer へ情報を受け渡しすることができます。サービストークンは、リソースプロバイダーが設定した Amazon SNS トピックを指定します。

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "MySeleniumTest" : {
      "Type": "Custom::SeleniumTester",
      "Version" : "1.0",
      "Properties" : {
        "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
        "seleniumTester" : "SeleniumTest()",
        "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
"http://search.mysite.com" ],
        "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
      }
    }
  },
  "Outputs" : {
    "topItem" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
    },
    "numRespondents" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
    }
  }
}
```

Note

`Fn::GetAtt` を使用してアクセスしたデータの名前と値は、プロバイダーから CloudFormation への応答中にカスタムリソースプロバイダーから返されます。custom resource provider がサードパーティーである場合、template developer は、それらの戻り値の名前を custom resource provider から入手する必要があります。

- CloudFormation が、スタックに関する情報、スタックテンプレートからのカスタムリソースプロパティ、および応答のための S3 URL が含まれた "RequestType" : "Create" を使用して、リソースプロバイダーに Amazon SNS 通知を送信します。

通知を送信する際に使用する SNS トピックは、テンプレートの ServiceToken プロパティに埋め込まれます。値をハードコーディングすることを避けるために、テンプレート開発者はテンプレートパラメータを使用します。スタックの起動時に値を入力することができます。

次の例は、カスタムリソースの Create リクエストを示します。Custom::SeleniumTester を LogicalResourceId として作成されたカスタムリソースタイプの名前 (MySeleniumTester) が含まれています。

```
{
  "RequestType" : "Create",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "ResponseURL" : "http://pre-signed-S3-url-for-response",
  "ResourceType" : "Custom::SeleniumTester",
  "LogicalResourceId" : "MySeleniumTester",
  "ResourceProperties" : {
    "seleniumTester" : "SeleniumTest()",
    "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://search.mysite.com" ],
    "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
  }
}
```

Create リクエストのリクエストオブジェクトの詳細については、[リクエストおよびレスポンスのリファレンス](#) トピックを参照してください。

- custom resource provider は、template developer から送信されたデータを処理し、Create リクエストが成功したかどうかを判断します。リソースプロバイダーは、CloudFormation から送信された S3 URL を使用して、SUCCESS または FAILED の応答を送信します。

CloudFormation に返される応答フィールドは、応答のタイプによって異なります。特定のリクエストタイプの応答フィールドの詳細については、[リクエストおよびレスポンスのリファレンス](#) セクションのそのリクエストタイプのドキュメントを参照してください。

custom resource provider は、作成リクエストまたは更新リクエストへの応答として、レスポンスの Data フィールドでデータ要素を返すことができます。これらは名前/値ペアであり、名

前は、スタックテンプレートのカスタムリソースで使用されている Fn::GetAtt 属性に対応します。値は、テンプレート開発者が、リソースから属性名を使って Fn::GetAtt を呼び出したときに返されるデータです。

次に、カスタムリソースのレスポンスの例を示します。

```
{
  "Status" : "SUCCESS",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "LogicalResourceId" : "MySeleniumTester",
  "PhysicalResourceId" : "Tester1",
  "Data" : {
    "resultsPage" : "http://www.myexampledomain/test-results/guid",
    "lastUpdate" : "2012-11-14T03:30Z"
  }
}
```

Create リクエストの応答オブジェクトの詳細については、[リクエストおよびレスポンスのリアレンジ](#) トピックを参照してください。

StackId、RequestId、および LogicalResourceId フィールドはリクエストからそのままコピーする必要があります。

- CloudFormation は、スタックのステータスを CREATE_COMPLETE または CREATE_FAILED として宣言します。スタックが正常に作成された場合、テンプレート開発者は、作成したカスタムリソースの出力値に [Fn::GetAtt](#) でアクセスすることで使用できます。

たとえば、先ほど例示したカスタムリソーステンプレートでは、Fn::GetAtt を使用してリソースの出力をスタック出力にコピーしています。

```
"Outputs" : {
  "topItem" : {
    "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
  },
  "numRespondents" : {
    "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
  }
}
```

ステップ 2: スタックの更新

既存のスタックを更新するには、以下の例に示すように、スタック内のリソースのプロパティの更新を指定するテンプレートを送信する必要があります。CloudFormation は、テンプレートで指定された変更があるリソースのみを更新します。詳細については、「[スタックリソースの更新動作を理解する](#)」を参照してください。

カスタムリソースの更新には、基になる物理リソースの置き換えが伴うことがあります。CloudFormation テンプレートでカスタムリソースを更新すると、CloudFormation が、そのカスタムリソースに更新リクエストを送信します。カスタムリソースを置き換えることが必要になった場合には、新しいカスタムリソースから新しい物理 ID を含むレスポンスを送信する必要があります。CloudFormation は、そのレスポンスを受け取ると、古いカスタムリソースと新しいカスタムリソースの `PhysicalResourceId` を比較します。この 2 つが異なる場合には、CloudFormation は置き換えが必要な更新と認識し、古いリソースに削除リクエストを送信します (「[ステップ 3: スタックの削除](#)」を参照)。

Note

カスタムリソースに変更を行わない場合、CloudFormation はスタック更新中にリクエストを送信しません。

1. template developerが、カスタムリソースを含んだスタックの更新を開始します。更新時、template developerは、新しいプロパティをスタックテンプレートで指定できます。

カスタムリソースタイプを使用したスタックテンプレートに対する Update の例を以下に示します。

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "MySeleniumTest" : {
      "Type": "Custom::SeleniumTester",
      "Version" : "1.0",
      "Properties" : {
        "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
        "seleniumTester" : "SeleniumTest()",
        "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
          "http://search.mysite.com",
          "http://mynewsite.com" ],

```

```

        "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
    }
},
"Outputs" : {
    "topItem" : {
        "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
    },
    "numRespondents" : {
        "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
    }
}
}

```

2. CloudFormation が、"RequestType" : "Update" を使用して Amazon SNS 通知をリソースプロバイダーに送信します。これには Create コールに類似する情報が含まれていますが、OldResourceProperties フィールドに古いリソースプロパティが含まれており、ResourceProperties に更新されたリソースプロパティ (存在する場合) が含まれている点が異なります。

次は、Update リクエストの例です。

```

{
    "RequestType" : "Update",
    "RequestId" : "unique-request-id",
    "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
    "ResponseURL" : "http://pre-signed-S3-url-for-response",
    "ResourceType" : "Custom::SeleniumTester",
    "LogicalResourceId" : "MySeleniumTester",
    "PhysicalResourceId" : "Tester1",
    "ResourceProperties" : {
        "seleniumTester" : "SeleniumTest()",
        "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://search.mysite.com", "http://mynewsite.com" ],
        "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
    },
    "OldResourceProperties" : {
        "seleniumTester" : "SeleniumTest()",
        "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://search.mysite.com" ],
        "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
    }
}

```



```
}  
}
```

Update リクエストのリクエストオブジェクトの詳細については、[リクエストおよびレスポンスのリファレンス](#) トピックを参照してください。

- CloudFormation から送信されたデータをカスタムリソースプロバイダーが処理します。カスタムリソースは更新を実行し、SUCCESS または FAILED のいずれかの応答を S3 URL に送信します。その後、CloudFormation は古いカスタムリソースと新しいカスタムリソースの PhysicalResourceIDs を比較します。この 2 つが異なる場合には、CloudFormation は置き換えが必要な更新と認識し、古いリソースに削除リクエストを送信します。以下に示したのは、Update リクエストに対する custom resource provider のレスポンスの例です。

```
{  
  "Status" : "SUCCESS",  
  "RequestId" : "unique-request-id",  
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",  
  "LogicalResourceId" : "MySeleniumTester",  
  "PhysicalResourceId" : "Tester2"  
}
```

Update リクエストのレスポンスオブジェクトの詳細については、[リクエストおよびレスポンスのリファレンス](#) 「」 トピックを参照してください。

StackId、RequestId、および LogicalResourceId フィールドはリクエストからそのままコピーする必要があります。

- CloudFormation は、スタックのステータスを UPDATE_COMPLETE または UPDATE_FAILED として宣言します。更新に失敗すると、スタックはロールバックされます。スタックが正常に更新された場合、template developer は、作成したカスタムリソースの新しい出力値に Fn::GetAtt でアクセスすることができます。

ステップ 3: スタックの削除

- テンプレートデベロッパーは、カスタムリソースを含むスタックを削除します。CloudFormation は、スタックテンプレートで指定されている現在のプロパティを SNS トピックとともに取得し、カスタムリソースプロバイダーに対してリクエストを実行する準備をします。

- CloudFormation が、スタックに関する現在の情報、スタックテンプレートからのカスタムリソースプロパティ、および応答のための S3 URL が含まれた "RequestType" : "Delete" を使用して、リソースプロバイダーに Amazon SNS 通知を送信します。

スタックを削除したり、カスタムリソースの削除や置き換えを伴う更新を行ったりした場合は必ず、CloudFormation によって、新旧のカスタムリソース間で PhysicalResourceId が比較されます。この 2 つが異なる場合には、CloudFormation は置き換えが必要な更新と認識し、古いリソース (OldPhysicalResource) の削除リクエストを送信します (以下の Delete リクエストの例を参照)。

```
{
  "RequestType" : "Delete",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "ResponseURL" : "http://pre-signed-S3-url-for-response",
  "ResourceType" : "Custom::SeleniumTester",
  "LogicalResourceId" : "MySeleniumTester",
  "PhysicalResourceId" : "Tester1",
  "ResourceProperties" : {
    "seleniumTester" : "SeleniumTest()",
    "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://search.mysite.com",
    "http://mynewsite.com" ],
    "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
  }
}
```

Delete リクエストのリクエストオブジェクトの詳細については、[リクエストおよびレスポンスのリファレンス](#) トピックを参照してください。

ユーザー定義の名前が指定されている場

合、DescribeStackResource、DescribeStackResources、ListStackResources で表示できます。

- カスタムリソースプロバイダーは、CloudFormation から送信されたデータを処理し、Delete リクエストが成功したかどうかを判断します。リソースプロバイダーは、CloudFormation から送信された S3 URL を使用して、SUCCESS または FAILED の応答を送信します。カスタムリソースを持つスタックを正常に削除するには、削除リクエストに対して custom resource provider が適切に応答する必要があります。

以下に示したのは、Delete リクエストに対する custom resource provider のレスポンスの例です。

```
{
  "Status" : "SUCCESS",
  "RequestId" : "unique-request-id",
  "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10",
  "LogicalResourceId" : "MySeleniumTester",
  "PhysicalResourceId" : "Tester1"
}
```

Delete リクエストのレスポンスオブジェクトの詳細については、[リクエストおよびレスポンスのリファレンス](#)「」トピックを参照してください。

StackId、RequestId、および LogicalResourceId フィールドはリクエストからそのままコピーする必要があります。

4. CloudFormation は、スタックのステータスを DELETE_COMPLETE または DELETE_FAILED として宣言します。

Lambda を使用するカスタムリソース

Lambda 関数とカスタム リソースを関連付けた場合、この関数はカスタムリソースが作成、更新、または削除されるたびに呼び出されます。CloudFormation は Lambda API を呼び出して関数を呼び出し、すべてのリクエストデータ (リクエストタイプやリソースプロパティなど) を関数に渡します。Lambda 関数と CloudFormation との組み合わせによる機能性およびカスタマイズ性を活用して、スタック作成中の AMI ID の動的な検索やユーティリティ関数 (文字列反転関数など) の使用などの広範なシナリオを実現できます。

カスタムリソースとその仕組みの概要については、「[カスタムリソースのしくみ](#)」を参照してください。

トピック

- [チュートリアル: Lambda-backed カスタムリソースを使用した遅延メカニズムの作成](#)
- [cfn-response モジュール](#)

チュートリアル: Lambda-backed カスタムリソースを使用した遅延メカニズムの作成

このチュートリアルでは、サンプルの CloudFormation テンプレートを使用して Lambda-backed カスタムリソースを設定および起動する方法を示します。このテンプレートは、指定した時間だけスタックのデプロイを一時停止する遅延メカニズムを作成します。これは、依存リソースが作成される前にリソースが安定するのを待つ場合など、リソースのプロビジョニング中に意図的な遅延を発生させる必要がある場合に便利です。

Note

以前は Lambda-backed カスタムリソースによる AMI ID の取得が推奨されていましたが、現在は AWS Systems Manager パラメータの使用を推奨しています。この方法により、テンプレートの再利用性が高まり簡単にメンテナンスできます。詳細については、「[Systems Manager Parameter Store からプレーンテキスト値を取得する](#)」を参照してください。

トピック

- [概要:](#)
- [サンプルテンプレート](#)
- [サンプルテンプレートのチュートリアル](#)
- [前提条件](#)
- [スタックを起動する](#)
- [リソースのクリーンアップ](#)
- [関連情報](#)

概要:

このチュートリアルで使用するスタックテンプレートのサンプルは、Lambda-backed カスタムリソースを作成します。このカスタムリソースは、スタックの作成中に設定可能な遅延 (デフォルトでは 60 秒) を発生させます。遅延は、カスタムリソースのプロパティが変更された場合にのみ、スタックの更新中に発生します。

このテンプレートでは以下のリソースが提供されます。

- カスタムリソース
- Lambda 関数
- Lambda が CloudWatch にログを書き込めるようにする IAM ロール。

また、次の 2 つの出力も定義します。

- 関数が実際に待機した時間。
- Lambda 関数の実行ごとに生成される一意の識別子。

Note

CloudFormation は無料のサービスですが、Lambda は関数のリクエスト数とコードの実行時間に基づいて課金します。Lambda の料金の詳細については、「[AWS Lambda の料金](#)」を参照してください。

サンプルテンプレート

遅延メカニズムを含む Lambda-backed カスタムリソースのサンプルテンプレートは、以下のとおりです。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "LambdaExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": ["lambda.amazonaws.com"] },
            "Action": ["sts:AssumeRole"]
          }]
        },
        "Path": "/",
        "Policies": [{
          "PolicyName": "AllowLogs",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": ["logs:*"],
              "Resource": "*"
            }]
          }
        }]
      }
    }
  }
}
```

```

        }
    ]]
}
},
"CFNWaiter": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
        "Handler": "index.handler",
        "Runtime": "python3.9",
        "Timeout": 900,
        "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
        "Code": {
            "ZipFile": { "Fn::Join": ["\n", [
                "from time import sleep",
                "import json",
                "import cfnresponse",
                "import uuid",
                "",
                "def handler(event, context):",
                "    wait_seconds = 0",
                "    id = str(uuid.uuid1())",
                "    if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
                "        wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout", 0))",
                "        sleep(wait_seconds)",
                "        response = {",
                "            \"TimeWaited\": wait_seconds,",
                "            \"Id\": id ",
                "        }",
                "        cfnresponse.send(event, context, cfnresponse.SUCCESS, response,",
                "\"Waiter-\"+id)"
            ]]}
        ]}
    }
},
"CFNWaiterCustomResource": {
    "Type": "AWS::CloudFormation::CustomResource",
    "Properties": {
        "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
        "ServiceTimeout": 60
    }
}
},
"Outputs": {

```

```
"TimeWaited": {
  "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
  "Export": { "Name": "TimeWaited" }
},
"WaiterId": {
  "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
  "Export": { "Name": "WaiterId" }
}
}
```

YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: "AllowLogs"
          PolicyDocument:
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:*"
                Resource: "*"
  CFNWaiter:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Runtime: python3.9
      Timeout: 900
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
```

```
ZipFile:
  !Sub |
  from time import sleep
  import json
  import cfnresponse
  import uuid

  def handler(event, context):
    wait_seconds = 0
    id = str(uuid.uuid1())
    if event["RequestType"] in ["Create", "Update"]:
      wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
      sleep(wait_seconds)
    response = {
      "TimeWaited": wait_seconds,
      "Id": id
    }
    cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
"Waiter-"+id)
CFNWaiterCustomResource:
  Type: AWS::CloudFormation::CustomResource
  Properties:
    ServiceToken: !GetAtt CFNWaiter.Arn
    ServiceTimeout: 60
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
  WaiterId:
    Value: !GetAtt CFNWaiterCustomResource.Id
    Export:
      Name: WaiterId
```

サンプルテンプレートのチュートリアル

次のスニペットでは、Lambda 関数がどのようにカスタムリソースに関連付けられているのか、およびその出力を理解できるように、サンプルテンプレートの要素を抜粋して解説します。

[AWS::Lambda::Function](#) リソース CFNWaiter

`AWS::Lambda::Function` リソースは、関数のソースコード、ハンドラー名、ランタイム環境、実行ロールの Amazon リソースネーム (ARN) を指定します。

Handler プロパティは Python ソースコードを使用するため、`index.handler` に設定されます。インライン関数のソースコードを使用する場合の受け入れ済みのハンドラー識別子の詳細については、「[AWS::Lambda::Function Code](#)」を参照してください。

ソースファイルは Python コードであるため、Runtime は `python3.9` として指定されます。

Timeout は 900 秒に設定されます。

Role プロパティは `Fn::GetAtt` 関数を使用して、テンプレートの `AWS::IAM::Role` リソースで宣言された `LambdaExecutionRole` 実行ロールの ARN を取得します。

Code プロパティは、関数コードを、Python 関数を使用してインラインで定義します。サンプルテンプレートの Python 関数は、以下を実行します。

- UUID を使用して一意の ID を作成する
- リクエストが作成または更新されたかどうかをチェックする
- Create または Update のリクエスト中に、`ServiceTimeout` に指定された時間だけスリープ状態にする
- 待機時間と一意の ID を返す

JSON

```
...
"CFNWaiter": {
  "Type": "AWS::Lambda::Function",
  "Properties": {
    "Handler": "index.handler",
    "Runtime": "python3.9",
    "Timeout": 900,
    "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
    "Code": {
      "ZipFile": { "Fn::Join": ["\n", [
        "from time import sleep",
        "import json",
        "import cfnresponse",
        "import uuid",
        "",
        "def handler(event, context):",
        "    wait_seconds = 0",
        "    id = str(uuid.uuid1())",
        "    if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
```

```

        "    wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout
\", 0))",
        "    sleep(wait_seconds)",
        "    response = {",
        "        \"TimeWaited\": wait_seconds,",
        "        \"Id\": id ",
        "    }",
        "    cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
\"Waiter-\"+id)"
    ]}]
}
}
},
...

```

YAML

```

...
CFNWaiter:
  Type: AWS::Lambda::Function
  Properties:
    Handler: index.handler
    Runtime: python3.9
    Timeout: 900
    Role: !GetAtt LambdaExecutionRole.Arn
    Code:
      ZipFile:
        !Sub |
        from time import sleep
        import json
        import cfnresponse
        import uuid

        def handler(event, context):
            wait_seconds = 0
            id = str(uuid.uuid1())
            if event["RequestType"] in ["Create", "Update"]:
                wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
                sleep(wait_seconds)
            response = {
                "TimeWaited": wait_seconds,
                "Id": id
            }

```

```
        cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
        "Waiter-"+id)
    ...
```

[AWS::IAM::Role](#) リソース LambdaExecutionRole

AWS::IAM::Role リソースは Lambda 関数の実行ロールを作成します。これには、Lambda にその使用を許可するロール割り当てポリシーが含まれます。また、CloudWatch Logs へのアクセスを許可するポリシーも含まれます。

JSON

```
...
  "LambdaExecutionRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [{
          "Effect": "Allow",
          "Principal": { "Service": ["lambda.amazonaws.com"] },
          "Action": ["sts:AssumeRole"]
        }]
      },
      "Path": "/",
      "Policies": [{
        "PolicyName": "AllowLogs",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Action": ["logs:*"],
            "Resource": "*"
          }]
        }
      ]
    }
  },
  ...
```

YAML

```
...
  LambdaExecutionRole:
```

```

Type: AWS::IAM::Role
Properties:
  AssumeRolePolicyDocument:
    Statement:
      - Effect: "Allow"
        Principal:
          Service:
            - "lambda.amazonaws.com"
        Action:
          - "sts:AssumeRole"
  Path: "/"
  Policies:
    - PolicyName: "AllowLogs"
      PolicyDocument:
        Statement:
          - Effect: "Allow"
            Action:
              - "logs:*"
            Resource: "*"

```

...

[AWS::CloudFormation::CustomResource](#) リソース CFNWaiterCustomResource

カスタムリソースは、!GetAtt CFNWaiter.Arn を使用して、ARN を持つ Lambda 関数にリンクします。ServiceTimeout で設定されているように、作成および更新オペレーションに 60 秒の待機時間を実装します。リソースは、プロパティが変更された場合にのみ、更新オペレーションのために呼び出されます。

JSON

```

...
  "CFNWaiterCustomResource": {
    "Type": "AWS::CloudFormation::CustomResource",
    "Properties": {
      "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
      "ServiceTimeout": 60
    }
  }
},
...

```

YAML

```
...
CFNWaiterCustomResource:
  Type: AWS::CloudFormation::CustomResource
  Properties:
    ServiceToken: !GetAtt CFNWaiter.Arn
    ServiceTimeout: 60
...
```

Outputs

このテンプレートの Outputs は TimeWaited と WaiterId です。TimeWaited 値は、Fn::GetAtt 関数を使用して、ウェーターリソースが実際に待機した時間を指定します。WaiterId は Fn::GetAtt 関数を使用して、生成され実行に関連付けられた一意の ID を提示します。

JSON

```
...
"Outputs": {
  "TimeWaited": {
    "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
    "Export": { "Name": "TimeWaited" }
  },
  "WaiterId": {
    "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
    "Export": { "Name": "WaiterId" }
  }
}
}
```

YAML

```
...
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
```

```
WaiterId:
  Value: !GetAtt CFNWaiterCustomResource.Id
Export:
  Name: WaiterId
...
```

前提条件

Lambda や CloudFormation など、対応するサービスをすべて使用するには、IAM アクセス許可が必要です。

スタックを起動する

スタックを作成するには

1. [サンプルテンプレート](#) セクションから希望するテンプレート (YAML または JSON) を見つけ、`samplelambdabackedcustomresource.template` という名前を付けてお使いのマシンに保存します。
2. クラウドフォーメーション コンソール の <https://console.aws.amazon.com/cloudformation/> 開きます :
3. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。
4. [前提条件 - テンプレートの準備] で、[既存のテンプレートを選択] を選択します。
5. [テンプレートの指定] で [テンプレートファイルのアップロード] を選択し、[ファイルの選択] を選択します。
6. 前に保存した `samplelambdabackedcustomresource.template` テンプレートファイルを選択します。
7. [次へ] を選択します。
8. [スタック名] に **SampleCustomResourceStack** と入力し、[次へ] を選択します。
9. このチュートリアルでは、タグの追加も詳細設定の指定も不要です。[次へ] を選択します。
10. スタックの名前が正しいことを確認し、[作成] を選択します。

CloudFormation によってスタックが作成されるまでに数分かかることもあります。進捗状況を監視するには、スタックイベントを確認します。詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

スタックを正常に作成できた場合は、スタック内のすべてのリソース (Lambda 関数、カスタムリソースなど) も作成されています。Lambda 関数とカスタムリソースを正常に使用できました。

Lambda 関数からエラーが返される場合は、CloudWatch Logs [コンソール](#)で関数のログを確認します。ログストリームの名前は、カスタムリソースの物理 ID です。これは、スタックのリソースを表示して確認できます。詳細については、「[Amazon CloudWatch ユーザーガイド](#)」の「Logs に送信された CloudWatch ログデータを表示する」を参照してください。

リソースのクリーンアップ

不要なリソースに対して課金されないよう、作成したすべてのスタックリソースをクリーンアップするために、スタックを削除します。

スタックを削除するには

1. CloudFormation コンソールから、SampleCustomResourceStack スタックを選択します。
2. [アクション] を選択し、[スタックの削除] を選択します。
3. 確認メッセージで、[はい、削除する] を選択します。

作成したすべてのリソースが削除されます。

Lambda-backed カスタムリソースの作成方法と使用方法について理解できたので、今度はこのチュートリアルで紹介したサンプルテンプレートとコードを使用して他のスタックや関数を作成し、実際に使ってみましょう。

関連情報

- [CloudFormation カスタムリソースリファレンス](#)
- [AWS::CloudFormation::CustomResource](#)

cfn-response モジュール

CloudFormation テンプレートでは、Lambda 関数をカスタムリソースのターゲットとして指定できます。ZipFile プロパティを使用して[関数](#)のソースコードを指定すると、cfn-response モジュールをロードして、Lambda 関数からの応答をカスタムリソースに送信できます。cfn-response モジュールは、Lambda 関数を呼び出したカスタムリソースへの応答の送信を簡素化するライブラリです。このモジュールには send メソッドが含まれています。このメソッドは、Amazon S3 の署名付き URL (ResponseURL) を経由して、カスタムリソースに[応答オブジェクト](#)を送信します。

`cfn-response` モジュールは、ZipFile プロパティを使用してソースコードを作成した場合にのみ使用できます。Amazon S3 バケットに保存されたソースコードには使用できません。バケットのコードでは、独自の関数を作成してレスポンスを送信する必要があります。

Note

`send` メソッドを実行した後、Lambda 関数は終了するため、メソッドの後の記述は無視されます。

`cfn-response` モジュールの読み込み

Node.js 関数の場合は、`require()` 関数を使用して `cfn-response` モジュールをロードします。たとえば、次のコードでは、`cfn-response` という名前で `response` オブジェクトを作成しています。

```
var response = require('cfn-response');
```

Python の場合は、次の例に示すように、`import` ステートメントを使用して `cfnresponse` モジュールをロードします。

Note

この完全インポートステートメントを使用します。インポートステートメントの他の形式では、CloudFormation では応答モジュールが含まれません。

```
import cfnresponse
```

`send` メソッドのパラメータ

`send` メソッドで次のパラメータを使用できます。

`event`

[カスタムリソースのリクエスト](#)に含まれるフィールド。

`context`

関数および任意のコールバックが実行完了したとき、または Lambda 実行環境内からの情報にアクセスするときに指定できる Lambda 関数固有のオブジェクト。詳細については、『AWS

Lambda デベロッパーガイド』の「[Node.js による Lambda 関数のビルド](#)」を参照してください。

responseStatus

関数が正常に完了したかどうか。このステータスを指定するには、cfnresponse モジュール定数を使用します。成功に実行した場合は SUCCESS、失敗した場合は FAILED を指定します。

responseData

カスタムリソースの[応答オブジェクト](#)の Data フィールド。データの内容は、名前と値のペアのリストです。

physicalResourceId

オプション。関数を呼び出したカスタムリソースの一意の識別子。モジュールのデフォルトでは、Lambda 関数に関連付けられている Amazon CloudWatch Logs ログストリームの名前が使用されます。

PhysicalResourceId に返された値は、カスタムリソース更新オペレーションを変更できます。返される値が同じであれば、通常の更新と見なされます。返された値が異なる場合には、CloudFormation は新しい方が更新用のものであると認識し、古いリソースに削除リクエストを送信します。詳細については、「[AWS::CloudFormation::CustomResource](#)」を参照してください。

noEcho

オプション。Fn::GetAtt 関数を使用してカスタムリソースの出力を取得したときに、それをマスクするかどうかを示します。true に設定すると、返される値はすべてアスタリスク (****) でマスクされます。ただし、以下に指定した場所に保存されている情報は除きます。デフォルトでは、この値は false です。

Important

NoEcho 属性を使用しても、以下に保存されている情報はマスクされません。

- Metadata テンプレートセクション。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。詳細については、「[メタデータ](#)」を参照してください。
- Outputs テンプレートセクション。詳細については、「[出力](#)」を参照してください。
- リソース定義の Metadata 属性。詳細については、「[Metadata 属性](#)」を参照してください。

パスワードやシークレットなどの機密情報を含めるには、これらのメカニズムを使用しないことを強くお勧めします。

機密情報をマスキングするために NoEcho を使用方法の詳細については、[テンプレートに認証情報を埋め込まない](#) に関するベストプラクティスを参照してください。

例

Node.js

次の Node.js の例では、インラインの Lambda 関数で入力値を受け取り、その値に 5 を乗算しています。インライン関数は、パッケージを作成して Amazon S3 バケットにアップロードするのではなく、ソースコードをテンプレート内で直接指定できるため、小さな関数の場合は特に便利です。この関数では、cfn-response の send メソッドを使用して、呼び出たカスタムリソースに結果を返しています。

JSON

```
"ZipFile": { "Fn::Join": ["", [
  "var response = require('cfn-response');",
  "exports.handler = function(event, context) {",
  "  var input = parseInt(event.ResourceProperties.Input);",
  "  var responseData = {Value: input * 5};",
  "  response.send(event, context, response.SUCCESS, responseData);",
  "};",
]]}
```

YAML

```
ZipFile: >
  var response = require('cfn-response');
  exports.handler = function(event, context) {
    var input = parseInt(event.ResourceProperties.Input);
    var responseData = {Value: input * 5};
    response.send(event, context, response.SUCCESS, responseData);
  };
```

Python

次の Python の例では、インラインの Lambda 関数で整数値を受け取り、その値に 5 を乗算しています。

JSON

```
"ZipFile" : { "Fn::Join" : ["\n", [  
    "import json",  
    "import cfnresponse",  
    "def handler(event, context):",  
    "     responseValue = int(event['ResourceProperties']['Input']) * 5",  
    "     responseData = {}",  
    "     responseData['Data'] = responseValue",  
    "     cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,  
    \"CustomResourcePhysicalID\")",  
  ]}] }
```

YAML

```
ZipFile: |  
    import json  
    import cfnresponse  
    def handler(event, context):  
        responseValue = int(event['ResourceProperties']['Input']) * 5  
        responseData = {}  
        responseData['Data'] = responseValue  
        cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,  
        "CustomResourcePhysicalID")
```

モジュールのソースコード

トピック

- [非同期 Node.js ソースコード](#)
- [Node.js ソースコード](#)
- [Python ソースコード](#)

非同期 Node.js ソースコード

ハンドラーが非同期である場合の Node.js 関数のレスポンスモジュールのソースコードは次のとおりです。これを確認してモジュールの動作を理解し、独自の応答関数の実装に役立ててください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
    physicalResourceId, noEcho) {

    return new Promise((resolve, reject) => {
        var responseBody = JSON.stringify({
            Status: responseStatus,
            Reason: "See the details in CloudWatch Log Stream: " +
context.logStreamName,
            PhysicalResourceId: physicalResourceId || context.logStreamName,
            StackId: event.StackId,
            RequestId: event.RequestId,
            LogicalResourceId: event.LogicalResourceId,
            NoEcho: noEcho || false,
            Data: responseData
        });

        console.log("Response body:\n", responseBody);

        var https = require("https");
        var url = require("url");

        var parsedUrl = url.parse(event.ResponseURL);
        var options = {
            hostname: parsedUrl.hostname,
            port: 443,
            path: parsedUrl.path,
            method: "PUT",
            headers: {
                "content-type": "",
                "content-length": responseBody.length
            }
        };

        var request = https.request(options, function(response) {
            console.log("Status code: " + parseInt(response.statusCode));
            resolve(context.done());
        });
    });
};
```

```

        request.on("error", function(error) {
            console.log("send(..) failed executing https.request(..): " +
maskCredentialsAndSignature(error));
            reject(context.done(error));
        });

        request.write(responseBody);
        request.end();
    })
}

function maskCredentialsAndSignature(message) {
    return message.replace(/X-Amz-Credential=[^\s]+/i, 'X-Amz-Credential=*****')
        .replace(/X-Amz-Signature=[^\s]+/i, 'X-Amz-Signature=*****');
}

```

Node.js ソースコード

ハンドラーが非同期でない場合の Node.js 関数のレスポンスモジュールのソースコードは次のとおりです。これを確認してモジュールの動作を理解し、独自の応答関数の実装に役立ててください。

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
    physicalResourceId, noEcho) {

    var responseBody = JSON.stringify({
        Status: responseStatus,
        Reason: "See the details in CloudWatch Log Stream: " + context.logStreamName,
        PhysicalResourceId: physicalResourceId || context.logStreamName,
        StackId: event.StackId,
        RequestId: event.RequestId,
        LogicalResourceId: event.LogicalResourceId,
        NoEcho: noEcho || false,
        Data: responseData
    });

    console.log("Response body:\n", responseBody);

```

```
var https = require("https");
var url = require("url");

var parsedUrl = url.parse(event.ResponseURL);
var options = {
    hostname: parsedUrl.hostname,
    port: 443,
    path: parsedUrl.path,
    method: "PUT",
    headers: {
        "content-type": "",
        "content-length": responseBody.length
    }
};

var request = https.request(options, function(response) {
    console.log("Status code: " + parseInt(response.statusCode));
    context.done();
});

request.on("error", function(error) {
    console.log("send(..) failed executing https.request(..): " +
maskCredentialsAndSignature(error));
    context.done();
});

request.write(responseBody);
request.end();
}
```

Python ソースコード

Python 関数のレスポンスモジュールのソースコードは次のとおりです。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0

from __future__ import print_function
import urllib3
import json
import re

SUCCESS = "SUCCESS"
```

```
FAILED = "FAILED"

http = urllib3.PoolManager()

def send(event, context, responseStatus, responseData, physicalResourceId=None,
noEcho=False, reason=None):
    responseUrl = event['ResponseURL']

    responseBody = {
        'Status' : responseStatus,
        'Reason' : reason or "See the details in CloudWatch Log Stream:
{}".format(context.log_stream_name),
        'PhysicalResourceId' : physicalResourceId or context.log_stream_name,
        'StackId' : event['StackId'],
        'RequestId' : event['RequestId'],
        'LogicalResourceId' : event['LogicalResourceId'],
        'NoEcho' : noEcho,
        'Data' : responseData
    }

    json_responseBody = json.dumps(responseBody)

    print("Response body:")
    print(json_responseBody)

    headers = {
        'content-type' : '',
        'content-length' : str(len(json_responseBody))
    }

    try:
        response = http.request('PUT', responseUrl, headers=headers,
body=json_responseBody)
        print("Status code:", response.status)

    except Exception as e:

        print("send(..) failed executing http.request(..):",
mask_credentials_and_signature(e))

def mask_credentials_and_signature(message):
```

```
message = re.sub(r'X-Amz-Credential=[^\s]+', 'X-Amz-Credential=*****', message,
flags=re.IGNORECASE)
return re.sub(r'X-Amz-Signature=[^\s]+', 'X-Amz-Signature=*****', message,
flags=re.IGNORECASE)
```

テンプレートマクロを使用して CloudFormation テンプレートでカスタム処理を実行する

マクロを使用すると、検索して置換操作のような単純なアクションからテンプレート全体の広範な変換まで、テンプレートに対してカスタム処理を実行できるようになります。

幅広い可能性を把握するために、CloudFormation がホストするマクロである `AWS::Include` および `AWS::Serverless` トランスフォームを検討してください。

- [AWS::Include 変換](#) を使用すると、定型のテンプレートスニペットをテンプレートに挿入できます。
- [AWS::Serverless 変換](#) は、AWS サーバーレスアプリケーションモデル (AWS SAM) 構文で記述されたテンプレート全体を受け取って、互換性のある CloudFormation テンプレートに変換および拡張します。サーバーレスアプリケーションおよび AWS SAM の詳細については、「[AWS Serverless Application Model デベロッパーガイド](#)」を参照してください。

トピック

- [料金](#)
- [マクロの例](#)
- [関連リソース](#)
- [CloudFormation マクロの概要](#)
- [CloudFormation マクロ定義を作成する](#)
- [シンプルな文字列置換マクロの例](#)
- [処理されたテンプレートをトラブルシューティングする](#)

料金

マクロが実行されると、その関数の実行に関連する料金が Lambda 関数の所有者に請求されます。

`AWS::Include` および `AWS::Serverless` トランスフォームは、CloudFormation がホストするマクロです。これらの使用料は発生しません。

マクロの例

このセクションの例に加えて、ソースコードやテンプレートを含むサンプルマクロが [GitHub リポジトリ](#)にあります。これらの例は、説明を目的として「現状のまま」提供されています。

関連リソース

- [AWS::CloudFormation::Macro](#)
- [CloudFormation テンプレートの Transform セクション](#)
- [Fn::Transform](#)
- [AWS::Serverless 変換](#)
- [AWS::Include 変換](#)

CloudFormation マクロの概要

マクロを使用してテンプレート进行处理するには、2つの主要なステップがあります。マクロ自体を作成すること、そして次にマクロを使用してテンプレートに対して処理を実行することです。

マクロ定義を作成するには、以下を作成する必要があります：

- テンプレート処理を実行するための Lambda 関数。この Lambda 関数は、スニペットまたはテンプレート全体、およびユーザーが定義した追加のパラメーターを受け入れます。処理されたテンプレートスニペットまたはテンプレート全体をレスポンスとして返します。
- [AWS::CloudFormation::Macro](#) タイプのリソース。これにより、ユーザーは CloudFormation テンプレート内から Lambda 関数を呼び出すことができます。このリソースは、このマクロを呼び出して Lambda 関数の ARN、およびデバッグを支援するための追加のオプションプロパティを指定します。このリソースをアカウント内に作成するには、AWS::CloudFormation::Macro リソースを記述したテンプレートを作成し、次にそのテンプレートからセルフマネージド型のアクセス許可を持つスタックまたはスタックセットを作成します。CloudFormationStackSets では現在、マクロを参照するテンプレートから、サービスマネージド型アクセス許可を使用してスタックセットを作成または更新することができません。

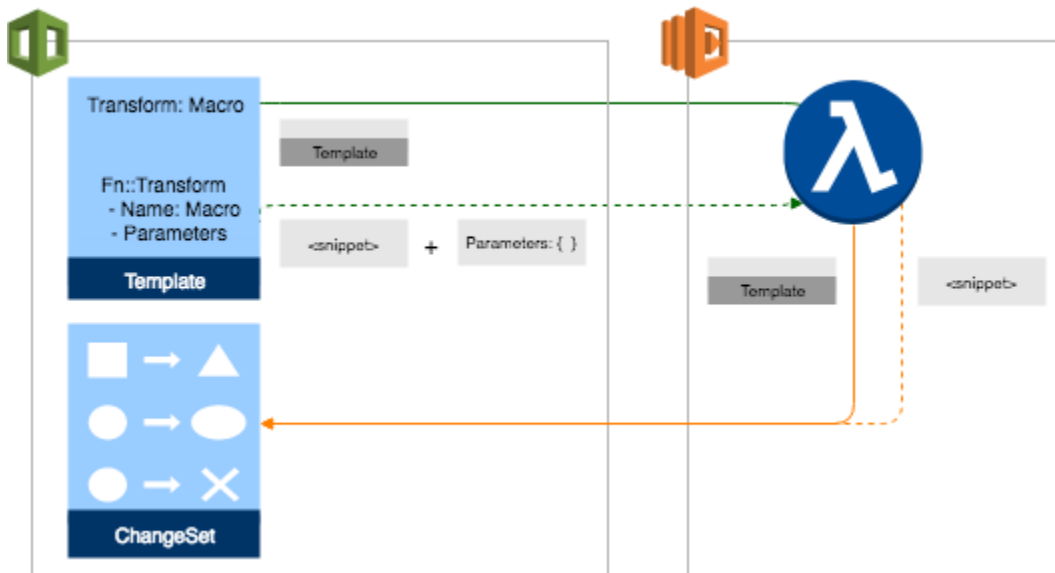
マクロを使用するには、テンプレート内の次のマクロを参照してください。

- テンプレートのセクション、つまりパート进行处理するには、変換したいテンプレートのコンテンツを基準にして配置されている Fn::Transform 関数内のマクロを参照します。Fn::Transform を使うときは、必要とされる特定のパラメータを渡すこともできます。

- テンプレート全体を処理するには、[Transform](#) セクションでマクロを参照してください。

次に、通常は変更セットを作成して実行します。(マクロを処理すると、気付かいうちに複数のリソースが追加されていることがあります。マクロに伴うすべての変更を確実に認識できるように、変更セットを使用することを強くお勧めします。) CloudFormation は、マクロリソースに指定された Lambda 関数に、指定されたテンプレートコンテンツとその他の指定されたパラメータを渡します。Lambda 関数は、スニペットまたはテンプレート全体で処理されたテンプレートコンテンツを返します。

テンプレート内のすべてのマクロが呼び出されたら、CloudFormation はテンプレートで処理したコンテンツを含めた変更セットを生成します。変更セットを確認したら、これを実行して変更を適用します。



スタックを直接作成する方法

マクロを参照するテンプレートを使用してスタックを作成または更新するには、通常、変更セットを作成して実行します。変更セットは、処理されたテンプレートに基づいて CloudFormation が実行するアクションを記述します。マクロを処理すると、知らないうちに複数のリソースが追加される可能性があります。マクロに伴うすべての変更を確実に認識できるように、変更セットを使用することを強くお勧めします。変更セットを確認したら、これを実行して実際に変更を適用します。

マクロによってテンプレートに IAM リソースが追加されることがあります。これらのリソースの場合、[それぞれどのような機能を備えているかを確認](#)する必要があります。CloudFormation は、テンプレートを処理しなければ、どのリソースが追加されるのかを認識することができません。そのため、参照されたマクロに IAM リソースが含まれている場合は、IAM の機能を把握し

たとえば変更セットを作成する必要があります。こうすることで、変更セットを実行するとき、CloudFormation が IAM リソースを作成するために必要な機能を準備できます。

変更セット内の提案された変更を最初に確認せずに、処理済みのテンプレートから直接スタックを作成または更新するには、CreateStack または UpdateStack リクエスト中に CAPABILITY_AUTO_EXPAND 機能を指定します。マクロが含まれているスタックテンプレートから直接スタックを作成するのは、どのような処理がマクロで実行されるかを把握している場合に限りです。スタックセットマクロで変更セットを使用することはできません。スタックセットを直接更新してください。

詳細については、「AWS CloudFormation API リファレンス」の「[CreateStack](#)」または「[UpdateStack](#)」を参照してください。

Important

スタックセットテンプレートに 1 つ以上のマクロが参照されており、変更セットの結果を確認せずに、処理されたテンプレートから直接スタックセットを作成する必要があります。マクロを処理すると、知らないうちに複数のリソースが追加される可能性があります。マクロを参照するテンプレートから直接スタックを作成または更新する前に、どのような処理がマクロで実行されるかを確認してください。

マクロを参照するテンプレートからスタックを起動する際のステップ数を減らすには、package および deploy AWS CLI コマンドを使用できます。詳細については、「[AWS CLI を使用してローカルアーティファクトを S3 バケットにアップロードする](#)」および「[変換が含まれるスタックを作成する](#)」を参照してください。

考慮事項

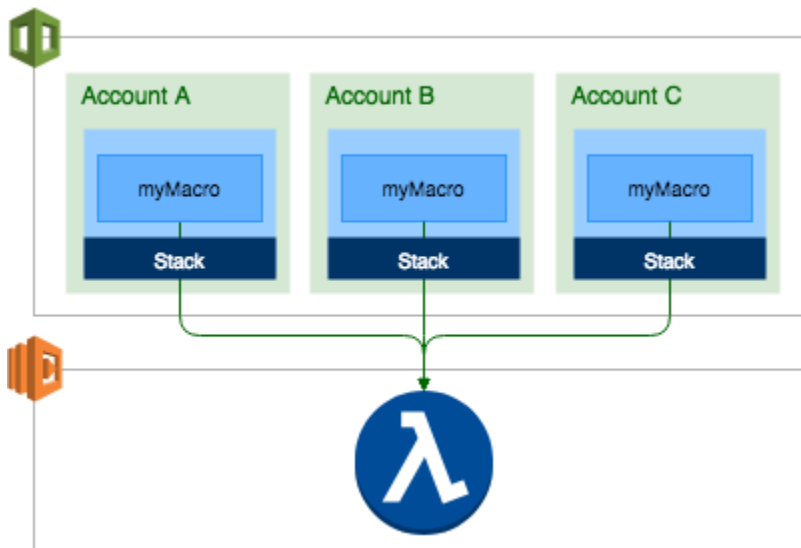
マクロを使用する場合は、次の注意事項と制限事項に留意してください。

- マクロは Lambda を使用できる AWS リージョン でのみサポートされます。Lambda を使用できるリージョンのリストについては、「[AWS Lambda endpoints and quotas](#)」(エンドポイントとクォータ)を参照してください。
- 処理されたテンプレートスニペットはすべて有効な JSON である必要があります。
- 処理されたテンプレートのスニペットは、スタックの作成、スタックの更新、スタックセットの作成、またはスタックセットの更新オペレーションの検証チェックに合格する必要があります。
- CloudFormation は、まずマクロを解決し、次にテンプレートを処理します。生成されるテンプレートは有効な JSON である必要があり、テンプレートサイズ制限を超えることはできません。

- CloudFormation がテンプレート内のエレメントを処理する順序のため、マクロは CloudFormation に返される処理済みテンプレートコンテンツにモジュールを含めることはできません。詳細については、「[マクロ評価順](#)」を参照してください。
- 更新ロールバック機能を使用する場合、CloudFormation は元のテンプレートのコピーを使用します。含まれるスニペットが変更されていても、元のテンプレートにロールバックされます。
- マクロは再帰的に処理されないため、マクロをマクロ内に含めることはできません。
- Fn::ImportValue 組み込み関数は、現在マクロではサポートされていません。
- テンプレートに含まれている組み込み関数は、マクロの後で評価されます。したがって、マクロが返す処理済みテンプレートコンテンツに組み込み関数の呼び出しを含めることができ、それらは通常どおりに評価されます。
- StackSets では現在、マクロを参照するテンプレートから、サービスマネージド型アクセス許可を使用してスタックセットを作成または更新することができません。

マクロアカウントの範囲と許可

マクロは、リソースとして作成されたアカウントでのみ使用できます。マクロ名は指定のアカウント内で一意である必要があります。ただし、基盤となる Lambda 関数でマクロアカウントアクセスを有効にし、その後複数のアカウントでその機能を参照するマクロ定義を作成することで、同じ機能を複数のアカウントでできるようにすることができます。以下の例では、3 つのアカウントにそれぞれ同じ Lambda 関数を指すマクロ定義が含まれています。



マクロ定義を作成するには、ユーザーは指定されたアカウント内にスタックまたはスタックセットを作成するための許可を持っていない限りなりません。

CloudFormation がテンプレートに含まれているマクロを正常に実行するには、ユーザーは基盤となる Lambda 関数に対する Invoke アクセス権を持っている必要があります。アクセス権がエスケーションしないよう、マクロの実行中に CloudFormation はユーザーとして振る舞います。

詳細については、「AWS Lambda デベロッパーガイド」の「[Managing permissions in AWS Lambda](#)」および「サービス認可リファレンス」の「[Actions, resources, and condition keys for AWS Lambda](#)」を参照してください。

CloudFormation マクロ定義を作成する

マクロ定義を作成すると、指定されたアカウントで基盤となる Lambda 関数が使用可能になり、CloudFormation はその関数を呼び出してテンプレートを処理します。

イベントマッピング

CloudFormation がマクロの Lambda 関数を呼び出すと、次の構造を使用して JSON 形式でリクエストを送信します:

```
{
  "region" : "us-east-1",
  "accountId" : "$ACCOUNT_ID",
  "fragment" : { ... },
  "transformId" : "$TRANSFORM_ID",
  "params" : { ... },
  "requestId" : "$REQUEST_ID",
  "templateParameterValues" : { ... }
}
```

- region

マクロが存在するリージョン。

- accountId

マクロが Lambda 関数を呼び出しているアカウントのアカウント ID。

- fragment

カスタム処理に使用可能なテンプレートコンテンツ (JSON 形式)。

- Transform テンプレートセクションに含まれるマクロの場合は、Transform セクションを除くテンプレート全体になります。

- `Fn::Transform` 組み込み関数呼び出しに含まれるマクロでは、`Fn::Transform` 関数を除く、テンプレート内の組み込み関数の場所に基づくすべての兄弟ノード (およびその子ノード) が含まれます。詳細については、「[マクロテンプレートスコープ](#)」を参照してください。
- `transformId`

この関数を呼び出すマクロの名前

- `params`

`Fn::Transform` 関数呼び出しの場合、関数に指定されたパラメータ。CloudFormation は、事前に評価することなく、こうしたパラメータを関数に渡します。

`Transform` テンプレートセクションに含まれるマクロの場合、このセクションは空です。

- `requestId`

この関数を呼び出すリクエストの ID です。

- `templateParameterValues`

テンプレートの [Parameters](#) セクションに指定されたパラメータ。CloudFormation は、事前に評価してから、こうしたパラメータを関数に渡します。

レスポンスの形式

CloudFormation は、Lambda 関数が次の JSON 形式でレスポンスを返すと想定しています。

```
{
  "requestId" : "$REQUEST_ID",
  "status" : "$STATUS",
  "fragment" : { ... },
  "errorMessage": "optional error message for failures"
}
```

- `requestId`

この関数を呼び出すリクエストの ID です。関数を呼び出すときに CloudFormation から提供されたリクエスト ID と一致する必要があります。

- `status`

リクエストのステータスです (大文字小文字を区別しません)。success のように設定する必要があります。CloudFormation は、他のすべてのレスポンスを失敗として扱います。

- fragment

処理済みのテンプレートに含められる、CloudFormation が処理したテンプレートコンテンツ (兄弟など)。CloudFormation は、Lambda 関数に渡されるテンプレートコンテンツを Lambda レスポンスで受け取るテンプレートフラグメントに置き換えます。

処理されたテンプレートのコンテンツは有効な JSON であり、処理されたテンプレートに含まれると有効なテンプレートになる必要があります。

関数が実際に CloudFormation から渡されるテンプレートコンテンツを変更しないものの、処理済みのテンプレートにそのコンテンツを含める必要がある場合、関数はレスポンスとしてそのテンプレートコンテンツを CloudFormation に返す必要があります。

- errorMessage

変換が失敗した理由を説明するエラーメッセージ。CloudFormation は、スタックの [Stack details] (スタックの詳細) ページの [Events] (イベント) ペインに、このエラーメッセージを表示します。

例えば、次のようになります。

```
Error creating change set: Transform
      AWS ##### account
      number::macro name failed with:
      error message string.
```

マクロ定義を作成する

CloudFormation マクロ定義を作成するには

1. テンプレートコンテンツの処理を処理する [Lambda 関数を構築](#) します。テンプレート全体までの、テンプレートの任意の部分の処理を行うことができます。
2. `AWS::CloudFormation::Macro` リソースタイプを含む CloudFormation テンプレートを作成し、Name および FunctionName プロパティを指定します。FunctionName プロパティは、CloudFormation がマクロを実行するときに呼び出す Lambda 関数の ARN を含む必要があります。
3. (オプション) デバッグを支援するために、マクロの `AWS::CloudFormation::Macro` リソースタイプを作成する際に LogGroupName プロパティと LogRoleArn プロパティを指定することもできます。これらのプロパティを使用すると、マクロの基盤となる Lambda 関数を呼び出すときに CloudFormation がエラーログ情報を送信する CloudWatch Logs ロググループを指定

でき、またそれらのログにログエントリを送信するときに CloudFormation が引き受けるロールを指定できます。

4. 使用するアカウントのマクロでテンプレートを使用して[スタックを作成](#)します。あるいは、管理者アカウントのマクロでテンプレートを使用して[セルフマネージド許可を持つスタックセットを作成](#)し、ターゲットアカウントにスタックインスタンスを作成します。
5. CloudFormation がマクロ定義を含むスタックを正常に作成したら、そのアカウント内でマクロを使用できるようになります。処理するテンプレートの内容に関連する適切な場所で、テンプレート内でマクロを参照して使用します。

マクロテンプレートスコープ

テンプレートの Transform セクションで参照されているマクロは、そのテンプレートの内容全体を処理できます。

Fn::Transform 関数で参照されているマクロは、テンプレート内のその Fn::Transform 関数の兄弟要素 (子を含む) の内容を処理できます。

たとえば、以下のテンプレートサンプルでは、AWS::Include は、それ自身を含む Fn::Transform 関数の場所に基づいて、MyBucket プロパティを処理できます。MyMacro は、Transform セクションに含まれているため、テンプレート全体の内容を処理できます。

```
# Start of processable content for MyMacro
AWSTemplateFormatVersion: 2010-09-09
Transform: [MyMacro]
Resources:
  WaitCondition:
    Type: AWS::CloudFormation::WaitCondition
  MyBucket:
    Type: AWS::S3::Bucket
    # Start of processable content for AWS::Include
    Properties:
      BucketName: amzn-s3-demo-bucket1
      Tags: [{"key": "value"}]
      'Fn::Transform':
        - Name: 'AWS::Include'
          Parameters:
            Location: s3://amzn-s3-demo-bucket2/MyFileName.yaml
      CorsConfiguration: []
    # End of processable content for AWS::Include
  MyEc2Instance:
```



```
Type: AWS::EC2::Instance
Properties:
  ImageID: ami-1234567890abcdef0
# End of processable content for MyMacro
```

マクロ評価順

テンプレートによっては、CloudFormation がホストするトランスフォーム ([AWS::Include](#) や [AWS::Serverless](#) など) をはじめ複数のマクロを参照できます。

マクロは、テンプレート内の位置に基づいて、最も深く外側にネストされているものから最も一般的なものまで順番に評価されます。テンプレート内の同じ場所にあるマクロは、リストされている順序に基づいて順番に評価されます。

AWS::Include や AWS::Transform などの変換は、アクションの順序と範囲の点で他のマクロと同じように扱われます。

例えば、以下のテンプレートサンプルでは、PolicyAdder マクロが最初に評価されます。テンプレート内で最も深くネストされているからです。次に、AWS::Serverless よりも先に MyMacro が評価されます。Transform セクションで AWS::Serverless より前に記述されているからです。

```
AWSTemplateFormatVersion: 2010-09-09
Transform: [MyMacro, AWS::Serverless]
Resources:
  WaitCondition:
    Type: AWS::CloudFormation::WaitCondition
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      Tags: [{"key": "value"}]
      'Fn::Transform':
        - Name: PolicyAdder
      CorsConfiguration: []
  MyEc2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageID: ami-1234567890abcdef0
```

シンプルな文字列置換マクロの例

次の例では、マクロを使用するプロセスとして、テンプレートでのマクロの定義、マクロ用の Lambda 関数の作成、テンプレートでのマクロの使用について説明します。

この例では、処理されたテンプレートの指定されたターゲットコンテンツの代わりに指定された文字列を挿入する簡単なマクロを作成します。次に、それを使用して、処理されたテンプレート内の指定された場所に空白の `WaitHandleCondition` を挿入します。

マクロの作成

マクロを使用する前に、次の 2 つのことを完了する必要があります。目的のテンプレート処理を実行する Lambda 関数を作成してからマクロ定義を作成し、その Lambda 関数を CloudFormation で使用できるようにすることです。

次のサンプルテンプレートには、サンプルマクロの定義が含まれています。特定の AWS アカウントでマクロを利用できるようにするために、テンプレートからスタックを作成します。マクロ定義は、マクロ名、簡単な説明を指定し、このマクロがテンプレートで使用されたときに CloudFormation が呼び出す Lambda 関数の ARN を参照します。(エラーログ用の `LogGroupName` または `LogRoleARN` プロパティは含めていません。)

この例では、このテンプレートから作成されたスタックの名前が `JavaMacroFunc` であるとします。マクロの `Name` プロパティはスタック名に設定されているので、結果として得られるマクロも `JavaMacroFunc` という名前になります。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Macro:
    Type: AWS::CloudFormation::Macro
    Properties:
      Name: !Sub '${AWS::StackName}'
      Description: Adds a blank WaitConditionHandle named WaitHandle
      FunctionName: 'arn:aws:lambda:us-east-1:012345678910:function:JavaMacroFunc'
```

マクロの使用

マクロを使用するには、`Fn::Transform` 組み込み関数を使用してそれをテンプレートに含めます。

以下のテンプレートを使用してスタックを作成すると、CloudFormation はサンプルマクロを呼び出します。基盤となる Lambda 関数は、指定された文字列を別の指定された文字列で置き換えます。

この場合、結果は空白になり `AWS::CloudFormation::WaitConditionHandle` が処理済みテンプレートに挿入されます。

```
Parameters:
  ExampleParameter:
    Type: String
    Default: 'SampleMacro'

Resources:
  2a:
    Fn::Transform:
      Name: "JavaMacroFunc"
      Parameters:
        replacement: 'AWS::CloudFormation::WaitConditionHandle'
        target: '$$REPLACEMENT$$'
      Type: '$$REPLACEMENT$$'
```

- 呼び出すマクロは、前のマクロ定義の例の `JavaMacroFunc` として指定されています。
- このマクロには、`target` と `replacement` の 2 つのパラメータが渡されます。これらは、ターゲット文字列とその目的の置換値を表します。
- `Type` はマクロを参照する `Fn::Transform` 関数の兄弟であるため、マクロは `Type` ノードの内容を操作できます。
- 結果の `AWS::CloudFormation::WaitConditionHandle` は `2a` という名前になります。
- テンプレートには、テンプレートパラメータ `ExampleParameter` も含まれています。このパラメータには、マクロからもアクセスできます (ただし、この場合は使用しません)。

Lambda 入力データ

CloudFormation がスタックの作成中にサンプルのテンプレートを処理するとき、`JavaMacroFunc` マクロ定義で参照されている Lambda 関数に次のイベントマッピングを渡します。

- `region: us-east-1`
- `accountId: 012345678910`
- `fragment:`

```
{
  "Type": "$$REPLACEMENT$$"
```

```
}
```

- `transformId`: 012345678910::JavaMacroFunc
- `params`:

```
{  
    "replacement": "AWS::CloudFormation::WaitConditionHandle",  
    "target": "$$REPLACEMENT$$"  
}
```

- `requestId`: 5dba79b5-f117-4de0-9ce4-d40363bfb6ab
- `templateParameterValues`:

```
{  
    "ExampleParameter": "SampleMacro"  
}
```

fragment には、マクロが処理できるテンプレートフラグメントを表す JSON が含まれています。このフラグメントは `Fn::Transform` 関数呼び出しの兄弟で構成されていますが、関数呼び出しそのものではありません。また、`params` にはマクロパラメータを表す JSON が含まれています。この場合は、置き換えとターゲットです。同様に、`templateParameterValues` には、テンプレート全体として指定されたパラメータを表す JSON が含まれています。

Lambda 関数コード

以下は、JavaMacroFunc マクロの基礎となる Lambda 関数の実際のコードです。指定されたターゲット文字列を探しながら、応答に含まれるテンプレートフラグメント (文字列、リスト、マップ形式など) を繰り返し処理します。指定されたターゲット文字列が見つかった場合、Lambda 関数はターゲット文字列を指定された置換文字列で置き換えます。そうでない場合は、関数はテンプレートフラグメントを変更しないままにします。それから、関数は CloudFormation に期待されるプロパティのマップを返します。詳細は後述します。

```
package com.macroexample.lambda.demo;  
  
import java.util.List;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
  
import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class LambdaFunctionHandler implements RequestHandler<Map<String, Object>,
    Map<String, Object>> {

    private static final String REPLACEMENT = "replacement";
    private static final String TARGET = "target";
    private static final String PARAMS = "params";
    private static final String FRAGMENT = "fragment";
    private static final String REQUESTID = "requestId";
    private static final String STATUS = "status";
    private static final String SUCCESS = "SUCCESS";
    private static final String FAILURE = "FAILURE";

    @Override
    public Map<String, Object> handleRequest(Map<String, Object> event, Context
context) {
        // TODO: implement your handler
        final Map<String, Object> responseMap = new HashMap<String, Object>();
        responseMap.put(REQUESTID, event.get(REQUESTID));
        responseMap.put(STATUS, FAILURE);
        try {
            if (!event.containsKey(PARAMS)) {
                throw new RuntimeException("Params are required");
            }

            final Map<String, Object> params = (Map<String, Object>) event.get(PARAMS);
            if (!params.containsKey(REPLACEMENT) || !params.containsKey(TARGET)) {
                throw new RuntimeException("replacement or target under Params are
required");
            }

            final String replacement = (String) params.get(REPLACEMENT);
            final String target = (String) params.get(TARGET);
            final Object fragment = event.getOrDefault(FRAGMENT, new HashMap<String,
Object>());
            final Object retFragment;
            if (fragment instanceof String) {
                retFragment = iterateAndReplace(replacement, target, (String) fragment);
            } else if (fragment instanceof List) {
                retFragment = iterateAndReplace(replacement, target, (List<Object>) fragment);
            } else if (fragment instanceof Map) {
                retFragment = iterateAndReplace(replacement, target, (Map<String, Object>)
fragment);
            } else {
```

```

        retFragment = fragment;
    }
    responseMap.put(STATUS, SUCCESS);
    responseMap.put(FRAGMENT, retFragment);
    return responseMap;
} catch (Exception e) {
    e.printStackTrace();
    context.getLogger().log(e.getMessage());
    return responseMap;
}
}

```

```

private Map<String, Object> iterateAndReplace(final String replacement, final
String target, final Map<String, Object> fragment) {
    final Map<String, Object> retFragment = new HashMap<String, Object>();
    final List<String> replacementKeys = new ArrayList<>();
    fragment.forEach((k, v) -> {
        if (v instanceof String) {
            retFragment.put(k, iterateAndReplace(replacement, target, (String)v));
        } else if (v instanceof List) {
            retFragment.put(k, iterateAndReplace(replacement, target, (List<Object>)v));
        } else if (v instanceof Map ) {
            retFragment.put(k, iterateAndReplace(replacement, target, (Map<String, Object>
v)));
        } else {
            retFragment.put(k, v);
        }
    });
    return retFragment;
}

```

```

private List<Object> iterateAndReplace(final String replacement, final String
target, final List<Object> fragment) {
    final List<Object> retFragment = new ArrayList<>();
    fragment.forEach(o -> {
        if (o instanceof String) {
            retFragment.add(iterateAndReplace(replacement, target, (String) o));
        } else if (o instanceof List) {
            retFragment.add(iterateAndReplace(replacement, target, (List<Object>) o));
        } else if (o instanceof Map) {
            retFragment.add(iterateAndReplace(replacement, target, (Map<String, Object>)
o));
        } else {
            retFragment.add(o);
        }
    });
}

```

```
    }
  });
  return retFragment;
}

private String iterateAndReplace(final String replacement, final String target,
final String fragment) {
  System.out.println(replacement + " == " + target + " == " + fragment );
  if (fragment != null AND_AND fragment.equals(target))
    return replacement;
  return fragment;
}
}
```

Lambda 関数レスポンス

以下は、Lambda 関数が処理のために CloudFormation に返すマッピングです。

- `requestId`: 5dba79b5-f117-4de0-9ce4-d40363bfb6ab
- `status`: SUCCESS
- `fragment`:

```
{
  "Type": "AWS::CloudFormation::WaitConditionHandle"
}
```

`requestId` は CloudFormation から送信されたものと一致し、`status` 値 SUCCESS は Lambda 関数がリクエストに含まれるテンプレートフラグメントを正常に処理したことを示します。このレスポンスでは、`fragment` は元のテンプレートスニペットの代わりに処理されたテンプレートに挿入するコンテンツを表す JSON を含んでいます。

結果として得られる処理済みテンプレート

CloudFormation は、Lambda 関数から正常な応答を受け取ると、返されたテンプレートフラグメントを処理済みテンプレートに挿入します。

以下は、この例で処理されたテンプレートです。JavaMacroFunc マクロを参照した `Fn::Transform` 組み込み関数呼び出しは含まれなくなりました。Lambda 関数によって返されたテンプレートフラグメントは適切な場所に含まれ、その結果、`"Type": "$$REPLACEMENT$$"` は `"Type": "AWS::CloudFormation::WaitConditionHandle"` に置き換えられています。

```
{
  "Parameters": {
    "ExampleParameter": {
      "Default": "SampleMacro",
      "Type": "String"
    }
  },
  "Resources": {
    "2a": {
      "Type": "AWS::CloudFormation::WaitConditionHandle"
    }
  }
}
```

処理されたテンプレートをトラブルシューティングする

マクロを使用する場合、処理されたテンプレートは CloudFormation コンソールにあります。

テンプレートのステージは、その処理ステータスを示します:

- **Original:** ユーザーがスタックまたはスタックセットを作成または更新するために最初に送信したテンプレートです。
- **Processed:** 参照されたマクロを処理した後にスタックまたはスタックセットを作成または更新するために CloudFormation が使用したテンプレートです。元のテンプレートが YAML としてフォーマットされていても、処理されたテンプレートは JSON としてフォーマットされます。

トラブルシューティングには、処理済みのテンプレートを使用します。テンプレートがマクロを参照していない場合は、オリジナルと処理済みのテンプレートは同一です。

詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

AWS CLI を使用して処理されたテンプレートを取得するには、[get-template](#) コマンドを使用します。

サイズ制限

処理されたスタックテンプレートの最大サイズは、CreateStack、UpdateStack、または ValidateTemplate リクエスト内に直接渡す場合は 51,200 バイトです。Amazon S3 テンプレート URL を使用して S3 オブジェクトとして渡す場合は 1 MB です。ただし、CloudFormation はテンプレート内のマクロを連続的に処理するため、処理中にテンプレートの一時的な状態が更新されます。

このため、処理中のテンプレートのサイズは、完全に処理されたテンプレートの許容サイズを一時的に超える場合があります。CloudFormation は、これらのインプロセステンプレートにいくらかのバッファを許可します。ただし、テンプレートやマクロを設計する際は、処理済みのスタックテンプレートの最大許容サイズに留意してください。

テンプレートの処理中に CloudFormation から Transformation data limit exceeded エラーが返された場合は、CloudFormation で処理中に許容される最大テンプレートサイズを超えています。

この問題を解決するには、以下の対処を検討してください。

- テンプレートを複数のテンプレートに再構成し、処理中のテンプレートが最大サイズを超えないようにします。例:
 - ネストされたスタックテンプレートを使用して、テンプレートの各部分をカプセル化します。詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。
 - 複数のスタックを作成し、クロススタック参照を使用してスタック間で情報を交換します。詳細については、「[別の CloudFormation スタックのリソース出力を参照する](#)」を参照してください。
- 特定のマクロから返されるテンプレートフラグメントのサイズを小さくします。CloudFormation はマクロから返されるフラグメントの内容には干渉しません。

ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する

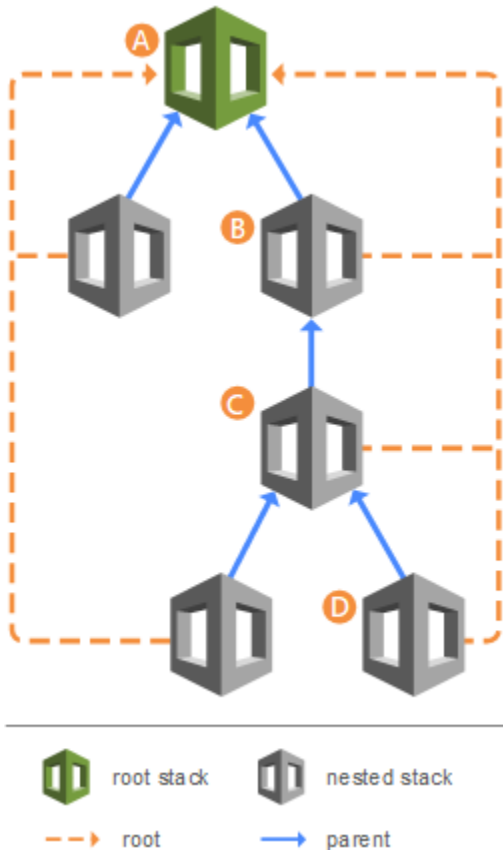
インフラストラクチャが拡張するにつれて、複数のテンプレートで同一のリソース構成を繰り返し作成する場合があります。このような重複を避けるために、共通の構成を専用のテンプレートに分離できます。次に、他のテンプレートで [AWS::CloudFormation::Stack](#) リソースを使用してこれらの専用テンプレートを参照し、ネストされたスタックを作成できます。

たとえば、ほとんどのスタックに使用しているロードバランサー構成があると仮定します。テンプレートに同じ構成をコピーアンドペーストする代わりに、ロードバランサーを対象にした専用テンプレートを作成できます。次に、同じロードバランサー構成を必要とする他のテンプレート内からこのテンプレートを参照できます。

ネストされたスタックには他のネストされたスタックを含めることができるため、次の図に示すようにスタックの階層になっています。ルートスタックは、ネストされたすべてのスタックが最終的に属

する最上位スタックです。ネストされたスタックにはそれぞれ、直接の親スタックが存在します。第 1 レベルのネストされたスタックの場合は、ルートスタックにも親スタックが存在します。

- スタック A は、階層内のネストされたその他すべてのスタックのルートスタックを表します。
- スタック B から見ると、スタック A は親スタックであると同時にルートスタックでもあります。
- スタック D から見るとスタック C が親スタック、スタック C から見るとスタック B が親スタックです。



トピック

- [テンプレート分割の前後の例](#)
- [ネストされたスタックのアーキテクチャの例](#)
- [ネストされたスタックでのスタックオペレーションの実行](#)
- [関連情報](#)

テンプレート分割の前後の例

この例では、単一の大規模な CloudFormation テンプレートを、ネストされたテンプレートを使用するより構造的で再利用可能な設計に再編成する方法を説明します。元の「ネスト前のスタック」テンプレートでは、1つのファイルにすべてのリソースが定義されているのがわかります。このテンプレートは、リソースの数が増えるにつれて煩雑になり、管理しにくくなる可能性があります。「ネスト後のスタック」テンプレートは、リソースをより小規模な個別のテンプレートに分割しています。ネストされたスタックは、それぞれが関連するリソースの特定のセットを処理するため、構造全体がよりよく整理され、維持しやすくなります。

スタックをネストする前

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Type: String
    Default: t2.micro
    Description: The EC2 instance type

  Environment:
    Type: String
    Default: Production
    Description: The deployment environment

Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: !Ref InstanceType

  MyS3Bucket:
    Type: AWS::S3::Bucket
```

スタックをネストした後

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyFirstNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/ amzn-s3-demo-bucket /first-nested-stack.yaml
      Parameters:
        # Pass parameters to the nested stack if needed
        InstanceType: t3.micro

  MySecondNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/ amzn-s3-demo-bucket /second-nested-stack.yaml
      Parameters:
        # Pass parameters to the nested stack if needed
        Environment: Testing
      DependsOn: MyFirstNestedStack
```

ネストされたスタックのアーキテクチャの例

このセクションでは、ネストされたスタックを参照する最上位スタックで構成される、ネストされたスタックのアーキテクチャを示します。ネストされたスタックは、Node.js Lambda 関数をデプロイし、最上位スタックからパラメータ値を受け取り、出力値を返します。この出力値は最上位スタックを通じて公開されます。

トピック

- [ステップ 1: ローカルシステムにネストされたスタック用のテンプレートを作成する](#)
- [ステップ 2: ローカルシステムで最上位スタックのテンプレートを作成する](#)
- [ステップ 3: テンプレートをパッケージ化してデプロイする](#)

ステップ 1: ローカルシステムにネストされたスタック用のテンプレートを作成する

次の例は、ネストされたスタックテンプレートの形式を示しています。

YAML

```
AWS::TemplateFormatVersion: 2010-09-09
Description: Nested stack template for Lambda function deployment
Parameters:
  MemorySize:
    Type: Number
    Default: 128
    MinValue: 128
    MaxValue: 10240
    Description: Lambda function memory allocation (128-10240 MB)
Resources:
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: !Sub "${AWS::StackName}-Function"
      Runtime: nodejs18.x
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify('Hello from Lambda!')
            }
          }
```

```

    };
  };
  MemorySize: !Ref MemorySize
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  Outputs:
    LambdaArn:
      Description: ARN of the created Lambda function
      Value: !GetAtt LambdaFunction.Arn

```

ステップ 2: ローカルシステムで最上位スタックのテンプレートを作成する

次の例は、最上位スタックのテンプレートの形式と、前のステップで作成したスタックを参照する [AWS::CloudFormation::Stack](#) リソースを示しています。

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Top-level stack template that deploys a nested stack
Resources:
  NestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: /path_to_template/nested-template.yaml
      Parameters:
        MemorySize: 256
  Outputs:
    NestedStackLambdaArn:
      Description: ARN of the Lambda function from nested stack
      Value: !GetAtt NestedStack.Outputs.LambdaArn

```

ステップ 3: テンプレートをパッケージ化してデプロイする

Note

ローカルでテンプレート进行操作する際は、AWS CLI `package` コマンドを使用すると、デプロイの準備を行うのに役立ちます。このコマンドは、ローカルのアーティファクト (TemplateURL を含む) を自動的に Amazon S3 にアップロードし、これらの S3 の場所への更新された参照を含む新しいテンプレートファイルを生成します。詳細については、「[AWS CLI を使用してローカルアーティファクトを S3 バケットにアップロードする](#)」を参照してください。

次に、[package](#) コマンドを使用して、ネストされたテンプレートを Amazon S3 バケットにアップロードできます。

```
aws cloudformation package \  
  --s3-bucket amzn-s3-demo-bucket \  
  --template /path_to_template/top-level-template.yaml \  
  --output-template-file packaged-template.yaml \  
  --output json
```

このコマンドは、`--output-template-file` で指定されたパスで新しいテンプレートを生成します。次に示すように、TemplateURL 参照を Amazon S3 の場所に置き換えます。

作成されるテンプレート

```
AWSTemplateFormatVersion: 2010-09-09  
Description: Top-level stack template that deploys a nested stack  
Resources:  
  NestedStack:  
    Type: AWS::CloudFormation::Stack  
    Properties:  
      TemplateURL: https://s3.us-west-2.amazonaws.com/amzn-s3-demo-bucket/8b3bb7aa7abfc6e37e2d06b869484bed.template  
      Parameters:  
        MemorySize: 256  
Outputs:  
  NestedStackLambdaArn:  
    Description: ARN of the Lambda function from nested stack  
    Value:
```

```
Fn::GetAtt:  
- NestedStack  
- Outputs.LambdaArn
```

package コマンドを実行したら、[deploy](#) コマンドを使用して処理されたテンプレートをデプロイできます。IAM リソースが含まれるネストされたスタックについては、`--capabilities` オプションを含めることによって IAM の機能に同意する必要があります。

```
aws cloudformation deploy \  
  --template-file packaged-template.yaml \  
  --stack-name stack-name \  
  --capabilities CAPABILITY_NAMED_IAM
```

ネストされたスタックでのスタックオペレーションの実行

ネストされたスタックを扱う場合は、操作時に注意が必要です。特定のスタックオペレーション (例: スタックの更新) では、ネストされたスタックから直接実行するのではなく、ルートスタックから起動する必要があります。ルートスタックを更新すると、テンプレートが変更されたネストされたスタックのみが更新されます。

さらに、ネストされたスタックがあると、ルートスタックでのオペレーションに影響する場合があります。たとえば、ネストされたスタックの 1 つが `UPDATE_ROLLBACK_IN_PROGRESS` 状態でスタックした場合、ルートスタックはそのネストされたスタックがロールバックを完了するまで待機してから続行します。更新オペレーションに進む前に、スタックの更新がロールバックされた場合にキャンセルする IAM アクセス権限があることを確認してください。詳細については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。

ルートスタックとネストされたスタックを検索するには、次の手順を実行します。

ネストされたスタックのルートスタックを表示するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページで、ルートスタックを表示するネストされたスタックの名前を選択します。

ネストされたスタックについては、スタック名の上に [NESTED] と表示されます。

3. [スタック情報] タブの [概要] セクションで、[ルートスタック] としてリストされているスタック名を選択します。

ルートスタックに属するネストされたスタックを表示するには

1. ネストされたスタックを表示するルートスタックから、[リソース] タブを選択します。
2. [タイプ] 列で、AWS::CloudFormation::Stack タイプのリソースを探します。

関連情報

- [既存のスタックのネスト化](#)
- [スタックリソースの更新動作を理解する](#)
- [失敗した、ネストされたスタックの更新からのロールバックを続ける](#)
- [ネストされたスタックのロールバック失敗](#)

CloudFormation テンプレートで待機条件を作成する

このトピックでは、テンプレートに待機条件を作成して、スタックリソースの作成を調整したり設定プロセスの進行状況を追跡する方法を説明します。たとえば、アプリケーション構成の一部が完了した後に別のリソースの作成を開始したり、インストールおよび構成プロセス中にシグナルを送信して進行状況を追跡できます。

CloudFormation が待機条件を含むスタックを作成する場合:

- 他のリソースと同様に待機条件を作成し、待機条件のステータスを CREATE_IN_PROGRESS に設定します。
- CloudFormation は、必要な数の成功シグナルを受信するか、待機条件のタイムアウト期間が経過するまで待機します。
- タイムアウト期間が経過する前に必要な数の成功シグナルを受信した場合:
 - 待機条件のステータスが CREATE_COMPLETE に変わります
 - スタックの作成が続行されます
- タイムアウト期間が経過するか、障害シグナルを受信した場合:
 - 待機条件のステータスが CREATE_FAILED に変わります
 - スタックのロールバック

⚠ Important

Amazon EC2 および Auto Scaling リソースについては、待機条件ではなく CreationPolicy 属性を使用することをお勧めします。CreationPolicy 属性をこれらのリソースに追加し、インスタンス作成プロセスが正常に完了したときにシグナルを送信するために cfn-signal ヘルパースクリプトを使用します。

詳細については、「[CreationPolicy 属性](#)」を参照してください。

ℹ Note

AWS PrivateLink を使用する場合、待機条件に応答する VPC のリソースには、CloudFormation 固有の Amazon Simple Storage Service (Amazon S3) バケットへのアクセス権が必要です。リソースは、署名付き Amazon S3 URL に応答する待機条件を送信する必要があります。Amazon S3 に応答を送信できない場合、CloudFormation は応答を受信せず、スタックオペレーションは失敗となります。詳細については、[インターフェイスエンドポイントを使用した CloudFormation へのアクセス \(AWS PrivateLink\)](#) および「[バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール](#)」を参照してください。

トピック

- [テンプレートでの待機条件の作成](#)
- [待機条件シグナル構文](#)
- [シグナルデータへのアクセス](#)

テンプレートでの待機条件の作成

1. 待機条件ハンドル

まず、スタックのテンプレートで [AWS::CloudFormation::WaitConditionHandle](#) リソースを定義します。このリソースは、シグナルの送信に必要な署名付き URL を生成します。これにより、AWS 認証情報を提供しなくてもシグナルを送信できるようになります。例えば、次のようになります。

Resources:

MyWaitHandle:

Type: AWS::CloudFormation::WaitConditionHandle

2. 待機条件

次に、スタックのテンプレートで [AWS::CloudFormation::WaitCondition](#) リソースを定義します。AWS::CloudFormation::WaitCondition の基本的な構造は次のようになります:

```
MyWaitCondition:
  Type: AWS::CloudFormation::WaitCondition
  Properties:
    Handle: String
    Timeout: String
    Count: Integer
```

AWS::CloudFormation::WaitCondition リソースには、2 つの必須プロパティと 1 つのオプションプロパティがあります。

- Handle (必須) - テンプレートで宣言された WaitConditionHandle の参照。
- Timeout (必須) - CloudFormation が、必要な数のシグナルを受信するのを待機する秒数。Timeout は最小バインドプロパティです。これは、指定された時刻よりも早くタイムアウトが発生することはありませんが、その後すぐに発生する可能性があることを意味します。指定できる最大時間は 43,200 秒 (12 時間) です。
- Count (オプション) — 待機条件のステータスを CREATE_COMPLETE に設定し、スタックの作成を再開する前に CloudFormation が受信する必要がある成功シグナルの数。指定がなければ、デフォルト値は 1 です。

通常、特定のリソースの作成直後に待機条件を開始する必要があります。そのためには、待機条件に DependsOn 属性を追加します。待機条件に DependsOn 属性を追加すると、CloudFormation は最初に DependsOn 属性にリソースを作成し、その後に待機条件を作成します。詳細については、「[DependsOn 属性](#)」を参照してください。

次の例は、以下を満たす待機条件を示しています:

- MyEC2Instance リソースが正常に作成された後に開始される
- MyWaitHandle リソースを WaitConditionHandle として使用する
- タイムアウトは 4,500 秒
- デフォルト Count が 1 である (Count プロパティが指定されていないため)

```
MyWaitCondition:
```

```
Type: AWS::CloudFormation::WaitCondition
DependsOn: MyEC2Instance
Properties:
  Handle: !Ref MyWaitHandle
  Timeout: '4500'
```

3. シグナルの送信

CloudFormation に成功または失敗を通知するには、通常、いくつかのコードまたはスクリプトを実行します。例えば、EC2 インスタンスで実行されているアプリケーションは、いくつかの追加の設定タスクを実行してから、完了を示すシグナルを CloudFormation に送信する場合があります。

シグナルは、待機条件ハンドルによって生成された署名付き URL に送信される必要があります。その署名済み URL を使用して、成功または失敗のシグナルを送信します。

シグナルを送信するには

1. テンプレート内の署名付き URL を取得するには、Ref 組み込み関数を、待機条件ハンドルの論理名とともに使用します。

次の例に示すように、テンプレートは Amazon EC2 インスタンスを宣言し、Amazon EC2 UserData プロパティを使用して署名付き URL を Amazon EC2 インスタンスに渡すことができます。これにより、これらのインスタンスで実行されているスクリプトまたはアプリケーションは、成功または失敗のシグナルを CloudFormation に送信できます。

```
MyEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro # Example instance type
    ImageId: ami-055e3d4f0bbeb5878 # Change this as needed (Amazon Linux 2023 in
us-west-2)
    UserData:
      Fn::Base64:
        Fn::Join:
          - ""
          - - "SignalURL="
            - { "Ref": "MyWaitHandle" }
```

この結果、次のような UserData 出力が得られます:

```
SignalURL=https://amzn-s3-demo-bucket.s3.amazonaws.com/....
```

注: AWS マネジメントコンソール およびコマンドラインツールでは、署名済み URL は待機条件 ハンドルリソースの物理 ID として表示されます。

2. (オプション) スタックが待機条件に入るタイミングを検出するために、次のいずれかの方法を使用できます:
 - 通知を有効にしてスタックを作成した場合、CloudFormation はすべてのスタックイベントの通知を指定のトピックに発行します。ユーザーまたはアプリケーションがそのトピックにサブスクライブしている場合は、待機条件ハンドル作成イベントの通知を監視し、通知メッセージから署名済み URL を取得できます。
 - AWS マネジメントコンソール、AWS CLI、または SDK を使用してスタックのイベントをモニタリングすることもできます。
3. シグナルを送信するには、署名済み URL を使用して HTTP リクエストメッセージを送信します。リクエスト方法は PUT である必要があります。Content-Type ヘッダーは空の文字列であるか省略される必要があります。リクエストメッセージは、「[待機条件シグナル構文](#)」で指定されている形式の JSON 構造である必要があります。

CloudFormation がスタックの作成を続行するようにするには、Count プロパティで指定された数の成功シグナルを送信する必要があります。Count が 1 より大きい場合、各シグナルの UniqueId 値は、特定の待機条件に送信されるすべてのシグナルにわたって一意である必要があります。UniqueId は任意の英数字の文字列です。

curl コマンドは、シグナルを送信する方法の 1 つです。次の例は、待機条件に成功のシグナルを送信する curl コマンドラインを示しています。

```
$ curl -T /tmp/a \  
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-  
west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-  
WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?  
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wy0oRejVo  
%3D"
```

ファイル `/tmp/a` には、次の JSON 構造が格納されます。

```
{  
  "Status" : "SUCCESS",  
  "Reason" : "Configuration Complete",  
  "UniqueId" : "ID1234",  
  "Data" : "Application has completed configuration."
```

```
}
```

この例は、同じ成功のシグナルを送信する curl コマンドラインを示していますが、コマンドラインのパラメータとして JSON 構造を送信している点が異なります。

```
$ curl -X PUT \
  -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS", "Reason" : "Configuration Complete", "UniqueId" : "ID1234", "Data" : "Application has completed configuration."}' \
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wy0oRejVo%3D"
```

待機条件シグナル構文

待機条件ハンドルによって生成された URL にシグナルを送信する場合は、次の JSON 形式を使用する必要があります。

```
{
  "Status" : "StatusValue",
  "UniqueId" : "Some UniqueId",
  "Data" : "Some Data",
  "Reason" : "Some Reason"
}
```

プロパティ

Status フィールドの値は次のいずれかである必要があります。

- SUCCESS
- FAILURE

UniqueId フィールドは CloudFormation に対するシグナルを識別します。待機条件の Count プロパティが 1 より大きい場合、UniqueId の値は特定の待機条件に送信されるすべてのシグナルにわたって一意でなければなりません。そうでない場合、CloudFormation は同じ UniqueId のシグナルを以前に送信されたものの再送信であるとみなして無視します。

Data フィールドには、シグナルとともに送り返す任意の情報を含めることができます。テンプレート内で [Fn::GetAtt](#) 関数を使用して Data の値にアクセスできます。

Reason フィールドは、内容に関して JSON コンプライアンス以外の制限がない文字列です。

シグナルデータへのアクセス

有効なシグナルによって送信されたデータにアクセスするために、CloudFormation テンプレートで待機条件の出力値を作成できます。例えば、次のようになります。

Outputs:

WaitConditionData:

Description: *The data passed back as part of signalling the WaitCondition*

Value: !GetAtt *MyWaitCondition*.Data

その後、[describe-stacks](#) コマンドまたは CloudFormation コンソールの [出力] タブを使用してこのデータを表示できます。

Fn::GetAtt 関数は、JSON 構造内の名前と値のペアとして UniqueId と Data を返します。例：

```
{"Signal1": "Application has completed configuration."}
```

CloudFormation モジュールを使用してテンプレート全体に含めることができる再利用可能なリソース設定の作成

モジュールは、スタックテンプレート全体に含めるリソース構成をパッケージ化する、透過的で管理しやすく、繰り返し可能な方法です。モジュールは、共通のサービス構成とベストプラクティスを、スタックテンプレートに含めるためのモジュール式のカスタマイズ可能なビルディングブロックとしてカプセル化できます。モジュールを使用すると、リソース実装の複雑さを深く理解することなく、ベストプラクティス、専門分野の知識、承認されたガイドライン (セキュリティ、コンプライアンス、ガバナンス、業界規制などの領域について) をテンプレートに含めることができます。

たとえば、ネットワークのドメイン専門家は、組み込みのセキュリティグループと、セキュリティガイドラインに準拠した入力/出力ルールを含むモジュールを作成できます。そうして、そのモジュールをテンプレートに含めることで、VPC、サブネット、セキュリティグループ、ゲートウェイがどのように機能するかを考えるのに時間を費やすことなく、セキュアなネットワーキングインフラストラクチャをスタックにプロビジョニングできます。また、モジュールはバージョン管理されているた

め、セキュリティガイドラインが時間の経過とともに変更された場合、モジュールの作成者はそれらの変更を組み込んだ新しいバージョンのモジュールを作成できます。

テンプレートでモジュールを使用することの特徴は次のとおりです。

- 予測可能性 – モジュールは CloudFormation レジストリに登録されているスキーマに準拠する必要があります。そのため、テンプレートにモジュールを含めると解決できるリソースがわかります。
- 再利用性 – 複数のテンプレートとアカウントで同じモジュールを使用できます。
- トレーサビリティ – CloudFormation では、スタック内のどのリソースがモジュールからプロビジョニングされたかを把握できるため、リソース変更のソースを簡単に把握できます。
- 管理機能 – モジュールを登録したら、バージョンング、アカウント、リージョンの可用性など、CloudFormation レジストリを使用してモジュールを管理できます。

モジュールには次のものを含めることができます。

- モジュールからプロビジョニングされる 1 つ以上のリソース、および出力や条件などの関連データ。
- 任意のモジュールパラメータ。モジュールが使用されるたびにカスタム値を指定できます。

モジュールの開発の詳細については、「CloudFormation CLI ユーザーガイド」の「[モジュールの開発](#)」を参照してください。

トピック

- [モジュールを使用するときの考慮事項](#)
- [モジュールのバージョンングの概要](#)
- [CloudFormation プライベートレジストリのモジュールの使用](#)
- [パラメータを使用してモジュール値を指定する](#)
- [CloudFormation テンプレートのリファレンスモジュールリソース](#)

モジュールを使用するときの考慮事項

- モジュールは追加料金なしで使用できます。スタック内でそれらのモジュールが解決したリソースに対してのみお支払いいただきます。
- CloudFormation クォータ (スタックで許可されるリソースの最大数、テンプレート本文の最大サイズなど) は、テンプレートに含まれるリソースがモジュールから取得されているかどうかにかかわ

らず、処理されたテンプレートに適用されます。詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

- スタックレベルで指定するタグは、モジュールから派生した個々のリソースに割り当てられます。
- モジュールレベルで指定されたヘルパースクリプトは、CloudFormation がテンプレートを処理するときに、モジュールに含まれる個々のリソースには伝達されません。
- モジュールで指定された出力は、テンプレートレベルで出力に伝播されます。

各出力には、モジュールで定義されたモジュールの論理名と出力名を連結した論理 ID が割り当てられます。詳細については、「[デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)」を参照してください。

- モジュールで指定されたパラメータは、テンプレートレベルのパラメータには伝達されません。

ただし、モジュールレベルのパラメータを参照するテンプレートレベルのパラメータを作成できます。詳細については、「[パラメータを使用してモジュール値を指定する](#)」を参照してください。

モジュールのバージョニングの概要

CloudFormation レジストリは、AWS アカウント およびリージョン内で使用するモジュールの登録および管理ができるリポジトリとして機能します。アカウントおよびリージョン内で AWS、サードパーティーのパブリッシャー、独自のカスタム拡張機能など、さまざまなソースからモジュールを登録できます。詳細については、「[CloudFormation レジストリによる拡張機能の管理](#)」を参照してください。

モジュールには異なるバージョンが存在する場合もあるため、使用するモジュールのバージョンを指定できます。このバージョニング機能は、モジュールに依存する既存のスタックを壊さずにモジュールを更新または変更する必要があるときには特に便利です。

複数のバージョンのモジュールを使用するとき、次の点に注意してください。

- スタックオペレーション中、CloudFormation は、スタックオペレーションが実行されている AWS アカウント およびリージョンで、デフォルトバージョンとして現在登録されているモジュールのバージョンを使用します。これには、他のモジュールにネストされているモジュールも含まれます。

したがって、異なるアカウントまたはリージョンで、同じモジュールの異なるバージョンをデフォルトバージョンとして登録している場合、同じテンプレートを使用しても異なる結果になる可能性があることに注意してください。

- スタックオペレーション中、CloudFormation は、スタックオペレーションが実行されている AWS アカウント およびリージョンで、デフォルトバージョンとして現在登録されているリソースのバージョンを使用します。これには、モジュールを含めることによって生成されたリソースが含まれます。
- モジュールのデフォルトバージョンを変更しても、スタックの更新オペレーションは開始されません。ただし、次にそのモジュールを含むテンプレート (スタックの更新など) でスタックオペレーションを実行すると、CloudFormation は新しいデフォルトバージョンをオペレーションで使用します。

これに関する例外の 1 つは、以下で説明するように、[use previous template] オプションを指定してスタックの更新を実行することです。

- スタックの更新オペレーションで、[use previous template] (以前のテンプレートを使用) オプションを指定すると、CloudFormation はスタック更新のために以前に処理されたテンプレートを使用し、モジュールに加えた変更については再処理しません。
- 一貫した結果を保証するには、スタックセットで使用するスタックテンプレートにモジュールを含める場合、スタックインスタンスのデプロイを考えているすべてのアカウントおよびリージョンで、同じバージョンのモジュールがデフォルトバージョンとして設定されていることを確認してください。これは、他のモジュールにネストされているモジュールの場合を含みます。詳細については、「[StackSets を使用したアカウントとリージョン全体でのスタックの管理](#)」を参照してください。

サードパーティーのパブリックモジュールのアクティブ化における要件

アカウントおよびリージョンでサードパーティーのパブリックモジュールを正常にアクティブ化するには、モジュールに含まれる各サードパーティーのパブリック拡張機能 (リソースまたはモジュール) に対して次の事項が当てはまる必要があります。

- 拡張機能のアクティブ化 – 拡張機能は、使用するアカウントおよびリージョンでアクティブ化する必要があります。詳細については、「[CloudFormation レジストリからサードパーティーのパブリック拡張を使用する](#)」を参照してください。
- エイリアス登録 – モジュール内のエクステンションがタイプ名のエイリアスを使用する場合、エクステンションは同じタイプ名のエイリアスを使用してアカウントおよびリージョンに登録する必要があります。詳細については、「[拡張を参照するエイリアスを使用する](#)」を参照してください。
- バージョン互換性 – 現在アクティブ化されているエクステンションのバージョンは、モジュールで指定されているエクステンションのサポートされているメジャーバージョンのいずれかである必要があります。

適切なサードパーティのパブリック拡張機能および拡張機能のバージョンがアクティブ化されていない場合、CloudFormation はオペレーションに失敗し、モジュールを正常にアクティブ化する前にアクティブ化する必要がある拡張機能およびバージョンを一覧表示したエラーが表示されます。

CloudFormation プライベートレジストリのモジュールの使用

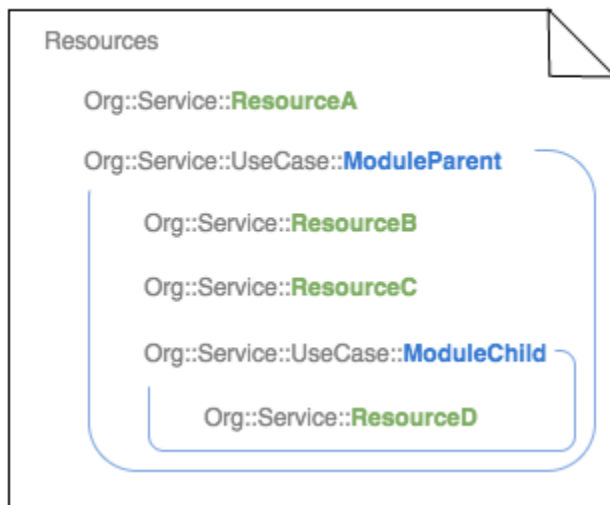
このトピックでは、CloudFormation テンプレートでモジュールを使用する方法について説明します。モジュールは、テンプレートに追加できるリソースの事前作成されたバンドルと考えてください。

モジュールを使用するには、次の手順を実行してください。

- モジュールの登録 – CloudFormation レジストリでモジュールをプライベート拡張機能として登録します。作業している AWS アカウント およびリージョンで登録されていることを確認します。詳細については、「[CloudFormation レジストリ](#)の概念」を参照してください。
- テンプレートに含める – 他のリソースと同様に、モジュールを CloudFormation テンプレートの [Resources](#) セクションに追加します。モジュールに必要なプロパティを指定する必要もあります。
- スタックの作成または更新 – スタックオペレーションを開始すると、含まれているモジュールを適切なリソースに解決する処理済みテンプレートが CloudFormation によって生成します。
- 変更のプレビュー – 変更を行う前に、変更セットを使用して追加または変更されるリソースを確認できます。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

次の例を考えてみましょう。リソースとモジュールの両方を含むテンプレートがあります。テンプレートには、1 つの個別のリソース ResourceA とモジュール ModuleParent が含まれます。このモジュールには、ResourceB および ResourceC の 2 つのリソースと、ネストされたモジュール ModuleChild が含まれます。ModuleChild には、1 つのリソース ResourceD が含まれます。このテンプレートからスタックを作成すると、CloudFormation はテンプレートを処理し、モジュールを適切なリソースに解決します。結果のスタックには、ResourceA、ResourceB、ResourceC、ResourceD の 4 つのリソースがあります。

Template



Processed Template



CloudFormation は、スタック内のどのリソースがモジュールから作成されたかを追跡します。この情報は、特定のスタックの [Events] (イベント)、[Resources] (リソース)、および [Drifts] (ドリフト) タブに表示できます。また、変更セットのプレビューにも表示されます。

モジュールはテンプレート内のリソースと区別できます。これは、リソースで 사용되는一般的な 3 つのパート規則とは対照的に、次の 4 つの部分からなる命名規則に従っているためです。

```
organization::service::use-case::MODULE
```

パラメータを使用してモジュール値を指定する

CloudFormation では、スタックの作成時または更新中に入力値を指定することにより、テンプレートパラメータを使用してスタックをカスタマイズできます。これらのパラメータにより、必要に応じてスタックの特定の特性を変更できます。テンプレートのパラメータの定義に関する詳細は、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

同様に、モジュールにパラメータを含めることもできます。これらのモジュールパラメータにより、使用しているテンプレート (または別のモジュール) からモジュールにカスタム値を入力できます。モジュールはこれらのカスタム値を使用し、プロパティ値が含まれるリソースにそのプロパティ値を設定できます。

スタック操作時にモジュールに渡される値を入力できるように、モジュールプロパティを設定するテンプレートパラメータを定義することもできます。

独自のモジュールパラメータを持つネストされたモジュールがモジュールに含まれている場合、次のいずれかの操作ができます。

- ネストされたモジュールのパラメータの値を親モジュールで直接指定します。
- 親モジュール内の対応するモジュールパラメータを定義して、ネストされたモジュールのパラメータを、親モジュールが含まれているテンプレート (またはモジュール) で設定できるようにします。

テンプレートパラメータを使用したモジュールパラメータ値の指定

次の例は、モジュールに値を渡すテンプレートパラメータを定義する方法を示しています。

My::

```
# Template containing My::
```

親モジュールから子モジュール内のリソースのプロパティを指定する

次の例は、別のモジュール内にネストされているモジュールでパラメータ値を指定する方法を示しています。

この最初のモジュール My::

```
# My::
```

```
    Description: Name for the bucket
    Type: String
  AccessControl:
    Description: AccessControl for the bucket
    Type: String
  Resources:
    S3Bucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: !Ref BucketName
        AccessControl: !Ref AccessControl
        DeletionPolicy: Retain
        VersioningConfiguration:
          Status: Enabled
```

次に、前のモジュールは親モジュール `My::S3::SampleBucket::MODULE` 内にネストされます。親モジュール `My::S3::SampleBucket::MODULE` は、次の方法で子モジュールのパラメータを設定します。

- `My::S3::SampleBucketPrivate::MODULE` の `AccessControl` パラメータを `Private` に設定します。
- `BucketName` では、モジュールパラメータを定義します。これにより、`My::S3::SampleBucket::MODULE` を含むテンプレート (またはモジュール) でバケット名を指定できます。

```
# My::S3::SampleBucket::MODULE
AWSTemplateFormatVersion: 2010-09-09
Description: A sample S3 Bucket. With Private AccessControl.
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: 'My::S3::SampleBucketPrivate::MODULE'
    Properties:
      BucketName: !Ref BucketName
      AccessControl: Private
```

モジュールパラメータの制約の指定

モジュールパラメータは、制約の適用をサポートしていません。モジュールパラメータに対して制約チェックを実行するには、必要な制約を持つテンプレートパラメータを作成します。その後、モジュールパラメータでそのテンプレートパラメータを参照します。テンプレートのパラメータの定義に関する詳細は、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

CloudFormation テンプレートのリファレンスモジュールリソース

CloudFormation テンプレートでは、別のリソースの名前またはプロパティに基づいて、あるリソースのプロパティを設定しなければならないことがよくあります。詳細については、「[リソースの参照](#)」を参照してください。

CloudFormation テンプレートのモジュール内に含まれるリソースを参照するには、次の 2 つの論理名を組み合わせる必要があります。

- テンプレートにモジュールを含めたときにそのモジュール自体に付けた論理名。
- そのモジュール内にある特定リソースの論理名。

これらの 2 つの論理名は、それらの間にピリオド (.) を使用するか、使用せずに組み合わせることができます。たとえば、モジュールの論理名が「MyModule」でリソースの論理名が「MyBucket」の場合、「MyModule.MyBucket」または「MyModuleMyBucket」とすることで、そのリソースを参照できます。

モジュール内のリソースの論理名を確認するには、CloudFormation レジストリまたは「[DescribeType](#)」操作を使用することにより、モジュールのスキーマを参照できます。スキーマリストには、モジュールの一部であるすべてのリソースおよびその論理名が一覧表示されます。

完全な論理名を取得したら、GetAtt や Ref などの CloudFormation 関数を使用してモジュールリソースのプロパティ値にアクセスできます。

たとえば、「S3Bucket」という論理名の `AWS::S3::Bucket` リソースを含む `My::S3::SampleBucket::MODULE` モジュールがあるとします。Ref 関数を使用してこのバケットの名前を参照するには、テンプレート (MyBucket) のモジュール名をモジュール (S3Bucket) のリソースの論理名と組み合わせます。完全な論理名は「MyBucket.S3Bucket」または「MyBucketS3Bucket」のいずれかです。

サンプルテンプレート

次のテンプレート例では、`My::S3::SampleBucket::MODULE` モジュールを使用して S3 バケットを作成します。Amazon SQS キューも作成し、名前をモジュールのバケット名と同じに設定します。さらに、テンプレートは作成された S3 バケットの Amazon リソースネーム (ARN) を出力します。

```
# Template that uses My::S3::SampleBucket::MODULE
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: My::S3::SampleBucket::MODULE
    Properties:
      BucketName: !Ref BucketName
  exampleQueue:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: !Ref MyBucket.S3Bucket
Outputs:
  BucketArn:
    Value: !GetAtt MyBucket.S3Bucket.Arn
```

CloudFormation スタックを使用した単一ユニットとしての AWS リソースの管理

スタックは、単一のユニットとして管理できる AWS リソースのコレクションです。つまり、スタックを作成、更新、削除することで、リソースのコレクションを作成、更新、削除できます。

スタックの作成には、リソースとそれらの設定を指定する CloudFormation テンプレートのデプロイが含まれます。その後、これらのリソースは、CloudFormation によってプロビジョニングおよび設定されます。

スタックの更新には、テンプレートまたはパラメータの変更が含まれます。CloudFormation は、送信された変更をスタックの現在の状態と比較して、変更されたリソースのみを更新します。CloudFormation は、どのプロパティを更新するかに応じて、リソースを中断したり、更新されたリソースを置き換えたりする場合があります。リソースの更新動作の詳細については、[スタックリソースの更新動作を理解する](#)を参照してください。

CloudFormation は、スタックを更新するための方法を 2 つ提供しています。

- **変更セット** – 変更セットを使用すると、CloudFormation がスタックに行う変更をプレビューしてから、それらの変更を適用するかどうかを決定できます。変更セットは、CloudFormation がスタックに行う変更を要約する JSON 形式のドキュメントです。変更セットは、CloudFormation が意図しない変更を行わないようにしたい場合や、複数のオプションを検討したい場合に使用します。例えば、変更セットを使用して、更新中に CloudFormation がスタックのデータベースインスタンスを置き換えないことを確認することができます。
- **直接的な更新** – スタックを直接更新するときは、変更を送信すると、CloudFormation が変更を直ちにデプロイします。更新を迅速にデプロイするには、直接更新を使用します。

スタックの削除は、スタックに関連付けられたリソースを削除します。たとえば、ウェブサーバー、データベース、ネットワークルールなど、ウェブアプリケーションの実行に必要なすべてのリソースをスタックに格納できます。ウェブアプリケーションが不要になった場合は、スタックを削除するだけで関連リソースをすべて削除できます。

Note

スタックをすぐに削除した場合でも、スタックリソースが運用されていた時間に対して課金されます。

CloudFormation は、すべてのスタックリソースが適切な作成または削除を確実に実行します。CloudFormation はスタックリソースを単一のユニットとして扱うため、スタックを作成または削除するにはすべてのリソースが正常に作成または削除される必要があります。リソースが作成できない場合、CloudFormation はスタックをロールバックし、作成されたリソースを自動的に削除します。リソースが削除できない場合、残されたリソースはスタックが正常に削除されるまで保持されます。

トピック

- [スタックを管理するためのインターフェイス](#)
- [CloudFormation コンソールからスタックを作成する](#)
- [CloudFormation コンソールからスタック情報を表示する](#)
- [スタックテンプレートを更新する](#)
- [スタックリソースの更新動作を理解する](#)
- [変更セットを使用して CloudFormation スタックを更新する](#)
- [スタックデプロイを検証する](#)
- [スタックを直接更新する](#)
- [スタック更新をキャンセルする](#)
- [CloudFormation コンソールからスタックを削除する](#)
- [スタックの進行状況を監視する](#)
- [ロールバックトリガーを使用して、アラーム違反時に CloudFormation スタックをロールバックする](#)
- [ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#)
- [AWS リソースを CloudFormation スタックにインポートする](#)
- [スタックリファクタリング](#)
- [リソースタイプのサポート](#)
- [クイック作成リンクを使用して CloudFormation スタックを作成する](#)
- [AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例](#)

スタックを管理するためのインターフェイス

CloudFormation スタックは、次のインターフェイスを使用して管理できます。

- CloudFormation コンソール – スタックにアクセスするために使用できるウェブインターフェイスを提供します。CloudFormation コンソールにアクセスするには、AWS マネジメントコンソールにサインインし、ナビゲーションバーの検索ボックスを使用して [CloudFormation] を検索し、検索結果から [CloudFormation] を選択します。
- AWS Command Line Interface – CloudFormation を含むさまざまな AWS のサービス用のコマンドを備えており、Windows、Mac、および Linux でサポートされています。CloudFormation コマンドについては、「AWS CLI コマンドリファレンス」の「[cloudformation](#)」を参照してください。
- AWS Tools for PowerShell – SDK for .NETから公開されている機能に基づいて構築された PowerShell モジュールのセットです。Tools for PowerShell では、PowerShell のコマンドラインから AWS リソースのオペレーションのスクリプトを作成できます。「[AWS Tools for PowerShell コマンドレットリファレンス](#)」で、CloudFormation のコマンドレットを確認できます。
- クエリ API – HTTPS リクエストを使用して呼び出す低レベル API アクションを提供します。アプリケーションで API コールを行う場合は、リクエストに署名するハッシュの生成など、低レベルの詳細を処理するコードを記述する必要があります。CloudFormation の API アクションの詳細については、「AWS CloudFormation API リファレンス」の「[アクション](#)」を参照してください。
- AWS SDK – 言語固有の API を提供し、署名の計算、リクエストの再試行処理、エラー処理など、接続のさまざまな詳細を処理します。詳細については、「[AWS での構築ツール](#)」を参照してください。
- AWS Cloud Development Kit (AWS CDK) – AWS CDK はオープンソースのソフトウェア開発フレームワークで、TypeScript、Python、Java、.NET などの使い慣れたプログラミング言語を使用して AWS インフラストラクチャを定義できます。CDK を使用すると、アプリケーションリソースをモデル化し、統合開発環境 (IDE) から直接 CloudFormation を使用してプロビジョニングできます。詳細については、[を参照してください。AWS Cloud Development Kit \(AWS CDK\)](#)

CloudFormation コンソールからスタックを作成する

スタックテンプレートを作成した後、CloudFormation コンソールまたはコマンドラインツールを使用してスタックを作成できます。コンソールは、事前定義されたオプションを備えたウィザード方式のインターフェイスを提供して、スタック作成プロセスを効率化します。

トピック

- [スタックの作成](#)
- [スタックオプションを設定する](#)
- [スタックの設定をプレビューする](#)

スタックの作成

このセクションのステップに従って、テンプレートをデプロイし、スタックを作成します。

前提条件

- スタックテンプレートを作成しておく必要があります。詳細については、「[CloudFormation テンプレートの使用](#)」を参照してください。

スタックを作成するには (コンソール)

- <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
- 画面の上部のナビゲーションバーで、スタックを作成する AWS リージョンを選択します。
- [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。

または、[既存のリソースを使用 (リソースをインポート)] オプションを選択して、テンプレートに記述されている既存の AWS リソースをインポートすることもできます。このオプションの詳細については、「[AWS リソースを CloudFormation スタックにインポートする](#)」を参照してください。

- [スタックの作成] ページで、以下の手順のいずれかを実行します。
 - 既存のテンプレートを使用するには、[前提条件 - テンプレートの準備] で [既存のテンプレートを選択] を選択します。次に、[テンプレートの指定] で、テンプレートの場所に基づいて [Amazon S3 URL] または [テンプレートファイルのアップロード] を選択します。
 - [Amazon S3 URL] を選択する場合は、S3 バケット内のテンプレートファイルへの URL を指定します。

ネストされたスタック (例えば、サブディレクトリにある他のテンプレートドキュメントに記述されているスタック) がテンプレートに含まれている場合は、S3 バケットに必要なファイルとディレクトリが含まれていることを確認してください。

バージョニング対応バケットからのテンプレートがある場合は、URL に ?

versionId=**version-id** を付加することでテンプレートの具体的なバージョンを指定することができます。バージョニング対応バケットの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バージョニングが有効なバケットでのオブジェクトの操作](#)」を参照してください。

URL は、許可を読み込んだ S3 バケットに保存されたテンプレート (最大サイズは 1 MB) を指す必要があります。URL は最大 1,024 文字を含むことができます。リソースによっては、バケットがスタックと同じリージョンにある必要があります。

- [テンプレートファイルのアップロード] を選択する場合は、[ファイルの選択] を選択して、ローカルコンピュータからテンプレートファイルを選択します。テンプレートファイルのサイズは 1 MB 以下にする必要があります。

テンプレートを選択すると、CloudFormation によってファイルがアップロードされ、S3 URL が表示されます。CloudFormation は、AWS アカウント内の Amazon S3 バケットにファイルをアップロードします。AWS アカウントに、CloudFormation によって作成された S3 バケットが既にある場合、CloudFormation はそのバケットにテンプレートを追加します。CloudFormation が作成した S3 バケットがない場合は、テンプレートファイルをアップロードするリージョンごとに CloudFormation が一意のバケットを作成します。

以下は、CloudFormation が作成した S3 バケットの使用時における考慮事項です。

- バケットには、AWS アカウントで Amazon S3 許可を持つすべてのユーザーがアクセスできます。
- CloudFormation は、デフォルトで有効になっているサーバー側の暗号化を使用してバケットを作成し、バケットに保存されているすべてのオブジェクトを暗号化します。

CloudFormation が作成したバケットの暗号化オプションは、直接管理することができます。これには、<https://console.aws.amazon.com/s3/> にある Amazon S3 コンソール、または AWS CLI などを使用できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」で「[Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定](#)」を参照してください。

- 手動でテンプレートを Amazon S3 にアップロードすることで、独自のバケットを使用してアクセス権限を管理できます。スタックを作成または更新する場合は、テンプレートファイルの Amazon S3 の URL を指定します。
5. [次へ] を選択して続行し、テンプレートを検証します。

続行する前に、CloudFormation がテンプレートを検証し、循環依存などの構文エラーやセマンティックエラーを検出します。検証中に、CloudFormation はテンプレートが有効な JSON であるかどうかをまず確認します。そうでない場合は、CloudFormation はテンプレートが有効な

YAML であるかどうかを確認します。両方のチェックが失敗すると、CloudFormation はテンプレートの検証エラーを返します。

6. [Specify stack details (スタック詳細の指定)] ページで、[スタックの名前] ボックスにスタック名を入力します。

スタック名は、スタックのリストから特定のスタックを見つけるために役立つ識別子です。スタック名には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字はアルファベット文字である必要があります。また、128 文字より長くすることはできません。

7. [パラメータ] セクションで、テンプレートで定義されているパラメータの値を指定します。
8. [次へ] を選択して、スタックの作成を続行します。
9. (オプション) [スタックオプションの設定] ページで、デフォルトのスタックオプションを変更します。詳細については、「[スタックオプションを設定する](#)」を参照してください。
10. テンプレートに IAM リソースが含まれる場合は、[機能] で [I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
11. [次へ] を選択して続行します。
12. [確認して作成] ページで、スタックの詳細を確認します。

スタックを起動する前に値を変更するには、変更する設定があるセクションで [編集] を選択します。

13. (オプション) 変更セットを作成して、スタックを作成する前にその設定をプレビューすることができます。[確認して作成] ページで [変更セットの作成] を選択し、指示に従います。詳細については、「[スタックの設定をプレビューする](#)」を参照してください。
14. [送信] を選択してスタックを起動します。

その後、CloudFormation がテンプレートで定義されているすべてのリソースの作成を開始します。

スタック作成の進捗状況とステータスは、新しいスタックの [イベント] タブで監視できます。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

コマンドラインを使用してスタックを作成するには

以下のコマンドのいずれかを使用できます。

- [create-stack](#) (AWS CLI)
- [New-CFNStack](#) (AWS Tools for Windows PowerShell)

コマンドラインを使用してスタックを作成する例については、「[AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例](#)」を参照してください。

スタックオプションを設定する

[スタックオプションの設定] ページでは、タグ、スタックイベントの通知、またはスタックポリシーといった CloudFormation スタックのオプションを設定できます。

次のスタックオプションを設定できます。

タグ

スタックや、CloudFormation がタグ付けをサポートしている任意のリソースに、最大 50 個のタグキーペアを追加できます。タグは、お客様定義のキーと値で、コスト追跡などの目的で AWS リソースに割り当てることができます。

キーは、任意の英数字またはスペースで構成されます。タグキーの値には最大 127 文字を使用できます。

値は、任意の英数字またはスペースで構成されます。タグの値には最大 255 文字を使用できます。

スタックを作成した後、スタックレベルのタグを追加、更新、または削除すると、スタックで更新が開始されます。スタックレベルのタグ伝播をサポートするすべてのリソースも併せて更新されます。

アクセス許可

CloudFormation が引き受けられる既存の IAM サービスロール。アカウントの認証情報を使用する代わりに、CloudFormation はロールの認証情報を使用してスタックを作成します。詳細については、「[CloudFormation サービスロール](#)」を参照してください。

スタック障害オプション

すべてのスタックデプロイと変更セットオペレーションのプロビジョニング障害オプションを指定します。詳細については、「[リソースのプロビジョニング時における失敗への対応方法を選択する](#)」を参照してください。

[Roll back all stack resources] (すべてのスタックリソースをロールバック) オプションは、スタックステータスが CREATE_FAILED または UPDATE_FAILED のとき、テンプレートで指定されたすべてのリソースをロールバックします。

作成オペレーションの場合、[Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースの保持) オプションは、正常に処理されたリソースの状態を保持しますが、失敗したリソースは次の更新オペレーションが実行されるまで失敗した状態が維持されます。

更新と変更セットのオペレーションの場合、[Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースの保持) オプションは、正常に処理されたリソースの状態を保持しますが、失敗したリソースを最後の既知の安定状態にロールバックします。失敗したリソースは UPDATE_FAILED 状態になります。最後の安定状態が確認されていないリソースは、次のスタックオペレーション時に削除されます。

スタック作成の以下の詳細オプションを設定することもできます。

スタックのポリシー

スタックの更新中、意図しない更新から保護するリソースを定義します。デフォルトでは、スタックの更新時にすべてのリソースを更新できます。

スタックポリシーを JSON として直接入力するか、スタックポリシーを含む JSON ファイルをアップロードすることができます。詳細については、「[スタックのリソースが更新されないようにする](#)」を参照してください。

ロールバック構成

スタックの作成および更新中にスタックの状態を CloudFormation で監視し、指定されたいずれかのアラームのしきい値をスタックが超過した場合に、そのオペレーションをロールバックすることができます。CloudFormation がモニタリングする Cloudwatch アラームを指定します。スタックオペレーションまたはモニタリング期間中にいずれかのアラームが ALARM 状態になった場合、CloudFormation はスタックオペレーション全体をロールバックします。詳細については、「[ロールバックトリガーを使用して、アラーム違反時に CloudFormation スタックをロールバックする](#)」を参照してください。

[Notification options] (通知オプション)

スタックイベントに関する通知が送信される新規または既存の Amazon Simple Notification Service トピックを指定できます。

Amazon SNS トピックを作成する場合、スタックイベントの通知の送信先となる名前と E メールアドレスを指定する必要があります。

スタック作成オプション

スタック作成には以下のオプションが含まれていますが、スタック更新には含まれていません。
タイムアウト

CloudFormation がスタック作成オペレーションのタイムアウトまでに割り当てる時間を分で指定します。CloudFormation が割り当てられた時間内にスタック全体を作成できない場合、スタックの作成がタイムアウトで失敗し、スタックはロールバックされます。

デフォルトでは、スタックの作成にタイムアウトはありません。ただし、個々のリソースには、実装するサービスの性質に基づいて、独自のタイムアウトがある可能性があります。例えば、スタック内の個々のリソースがタイムアウトになった場合、スタックの作成に指定されたタイムアウトに到達していない場合でも、スタックの作成もタイムアウトになります。

削除保護

スタックが誤って削除されるのを防ぎます。削除保護を有効にした状態でスタックを削除しようとする、削除は失敗し、ステータスを含め、スタックが変更されることはありません。詳細については、「[CloudFormation スタックの削除を防止する](#)」を参照してください。

削除保護はデフォルトで [無効] になっています。

スタックの設定をプレビューする

CloudFormation スタックを作成する前にスタックがどのように設定されるかをプレビューするには、変更セットを作成します。この機能で、変更セットを実行する前に、さまざまな構成を検証し、スタックに修正や変更を行うことができます。変更セットの詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

新しいスタックのために変更セットを作成する

新しいスタックの変更セットを作成するには、新しいスタックを作成する場合と同じようにスタックテンプレートを選択し、スタックの構成を行ってから、新しいスタックではなく新しい変更セットの作成を選択します。

新しいスタックのために変更セットを作成するには

1. [確認して作成] ページで、[変更セットを作成] をクリックします。

2. [変更セットを作成] ダイアログボックスで、変更セットの名前と、必要に応じて説明を入力します。[変更セットの作成] を選択します。

新しいスタックの変更セットを作成すると、CloudFormation によって以下の処理が実行されます。

- REVIEW_IN_PROGRESS のステータスを持つ新しいスタックを起動します。
- 前の手順で指定したスタック構成を反映する新しいスタックの変更セットを作成します。

CloudFormation により、提案されたスタックの [Change sets] (変更セット) ページが表示されます。CloudFormation は変更セットを作成しますが、そのステータスは CREATE_IN_PROGRESS で、実行ステータスは UNAVAILABLE です。CloudFormation が変更セットの作成を正常に完了するとき、変更セットのステータスが CREATE_COMPLETE に設定され、実行ステータスは AVAILABLE になります。スタックのステータスは REVIEW_IN_PROGRESS に更新されます。この時点で、変更セットを実行して、新しいスタックの作成を完了することができます。

CloudFormation によって、[Changes] (変更) ペインにスタックの構成案がリスト表示されます。

CloudFormation による変更セットの作成が失敗すると、変更セットのステータスが CREATE_FAILED に設定されます。[状況の理由] フィールドに表示されるエラーを修正してから、新しい変更セットを作成します。この段階で、次の変更セットを実行する前に、さまざまな構成を試みスタックへ修正と変更を加えることができます。

3. 変更セットに基づいて新しいスタックの作成を完了するには、[Execute] (実行) を選択し、ロールバック構成を指定して、[Execute change set] (変更セットの実行) を選択します。

CloudFormation コンソールからスタック情報を表示する

CloudFormation スタックの作成後は、AWS マネジメントコンソールを使用してスタックのデータやリソースを表示することができます。

CloudFormation スタックに関する情報を表示するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックが所在する AWS リージョン を選択します。

3. [スタック] ページで、スタック名を選択します。CloudFormation は選択したスタックのスタック詳細を表示します。

Note

スタックが削除されている場合、[フィルタステータス] オプションを使用して見つけることができます。詳細については、「[CloudFormation コンソールから削除されたスタックを表示する](#)」を参照してください。

表示できるスタック情報は次のとおりです。

Stack info (スタック情報)

以下のような、スタックとその構成に関する一般情報が表示されます。

概要:

スタックステータス、ドリフトステータス、削除保護などのステータス情報とともに、スタック名、スタック ID、ルートスタック、[IAM ロール](#)が表示されます。

タグ

スタックに関連付けられたタグを表示します。

スタックのポリシー

スタックの更新に対して保護されているスタックリソースについて説明します。これらのリソースを更新するには、スタックの更新中に明示的に許可する必要があります。詳細については、「[スタックのリソースが更新されないようにする](#)」を参照してください。

ロールバック構成

スタックオペレーション中または指定されたモニタリング期間中に CloudFormation にモニタリングするように指定した CloudWatch アラームが表示されます。スタックオペレーションまたはモニタリング期間中にいずれかのアラームが ALARM 状態になった場合、CloudFormation はスタックオペレーション全体をロールバックします。詳細については、「[ロールバックトリガーを使用して、アラーム違反時に CloudFormation スタックをロールバックする](#)」を参照してください。

[Notification options] (通知オプション)

スタックイベントに関する通知が送信される Amazon Simple Notification Service トピックが表示されます (指定されている場合)。

Events

スタックを作成、更新、削除した場合に追跡された操作を表示します。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

指定されたスタックオペレーションでトリガーされるイベントはすべて、同一クライアントのリクエストトークンに割り当てられ、オペレーションの追跡に使用できます。コンソールから開始されたスタック操作は、Console-StackOperation-ID のトークン形式を使用し、スタック操作の識別が容易になります。たとえば、コンソールを使用してスタックを作成した場合、作成されたスタックイベントは Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002 の形式の同じトークンを割り当てられます。

リソース

スタックの一部であるリソースを表示します。

アウトプット

スタックのテンプレートで宣言した出力を表示します。詳細については、「[デプロイされた CloudFormation スタックからエクスポートされた出力を取得する](#)」を参照してください。

パラメータ

スタックのパラメータとその値を表示します。

Systems Manager パラメータが含まれるスタックの場合、[解決済みの値] 列に、Systems Manager パラメータのスタック定義に使用された値が表示されます。詳細については、「[CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)」を参照してください。

テンプレート

スタックのテンプレートを表示します。

マクロを含むスタックの場合、[View original template] (元のテンプレートの確認) を選択して、ユーザーが送信したテンプレートを確認するか、[View processed template] (処理されたテンプレートの確認) を選択して、CloudFormation が参照されたマクロを処理した後にテンプレートを確認します。CloudFormation は、処理されたテンプレートを使用してスタックを作成または更新します。

変更セット

スタックの変更セットを表示します。

詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。

Git 同期

スタックの Git 同期ダッシュボードを表示します。

詳細については、「[Git 同期ステータスダッシュボード](#)」を参照してください。

スタックテンプレートを更新する

CloudFormation スタック内のリソースまたはプロパティを変更するには、スタックのテンプレートを更新する必要があります。そのスタックの既存のテンプレートから始めて、変更を行っていきます。テンプレートをソース管理システムに保存している場合は、テンプレートのコピーを開始点として使用します。そうでない場合は、CloudFormation からテンプレートのコピーを取得できます。

スタックのパラメータや設定 (スタックの Amazon SNS トピックなど) のみを変更する場合は、コピーを取得することなく既存のテンプレートを再利用できます。

[Infrastructure Composer](#) またはテキストエディタを使用して CloudFormation スタックテンプレートを更新できます。

Infrastructure Composer を使用して既存のスタックテンプレートを更新するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページで、更新するスタックの名前を選択します。
3. [テンプレート] タブを選択したら、[Infrastructure Composer で表示] を選択します。

CloudFormation が Infrastructure Composer でテンプレートを開きます。

4. 次のいずれかの方法を使用してテンプレートを更新します。
 - [Canvas] インターフェイス: ここでは、[リソース] パレットからドラッグアンドドロップできます。カードをダブルクリックしてリソースを設定し、[リソースプロパティ] パネルを開きます。必要に応じてリソースを接続します。[Canvas] インターフェイスの使用に関する詳細な手順については、「[AWS Infrastructure Composer で構成する方法](#)」を参照してください。
 - [テンプレート] インターフェイス: [Canvas] から [テンプレート] インターフェイスに切り替えます。テンプレートコードにインライン更新を行います。必要に応じて JSON 形式と YAML 形式に切り替えます。
5. [検証] を選択し、テンプレートに構文エラーがないか確認します。

6. CloudFormation に変更をエクスポートする準備ができたなら、[テンプレートの更新] を選択します。

AWS CLI を使用して既存のスタックテンプレートを更新する

1. 更新するスタックのテンプレートを取得するには、[get-template](#) CLI コマンドを使用します。
2. テンプレートをコピーしてテキストファイルに貼り付け、変更して保存します。テンプレートのみをコピーします。コマンドではテンプレートを引用符で囲みますが、テンプレートを囲んでいる引用符はコピーしません。テンプレート自体は、左中括弧で開始され、右中括弧で終了します。このファイルで、スタックのリソースに対する変更を指定します。

テンプレートを変更するときは、以下の点に留意してください。

- 更新をサポートしないリソースが使用しているパラメータを追加、変更、または削除することはできません。
- ほとんどのリソースの場合、リソースの論理名を変更することは、そのリソースを削除して新しいリソースと置き換えることと同等です。名前を変更したリソースに依存する他のリソースも更新する必要があります。場合によっては置き換えられます。その他のリソースは、更新を開始するために、(論理名だけでなく) プロパティの更新が必要になります。
- リソースによっては、特定のプロパティに設定できる値に関する制限がある場合があります。例えば、RDS データベースインスタンスの `AllocatedStorage` プロパティへの変更は、現在の値よりも大きくする必要があります。更新がこれらのルールに違反すると、その部分が失敗します。
- 1 つのリソースを更新すると、それを参照する他のリソースにも影響がおよぶ可能性があります。[Ref 関数](#) または [Fn::GetAtt 関数](#) といった関数を使用して別のリソースに基づくプロパティを設定する場合、CloudFormation は、参照先リソースの変更時に参照元リソースも更新します。
- 特定のリソースプロパティを更新した場合の影響については、「[AWS リソースおよびプロパティタイプのリファレンス](#)」を参照してください。各プロパティについての更新の効果は、以下のいずれかになります。
 - アップデートに関する要件: [中断はありません](#)
 - 更新に伴う要件: [一時的な中断](#)
 - 更新に伴う要件: [置換](#)
- テンプレートの JSON または YAML 構文を検証するには、[validate-template](#) CLI コマンドを使用するか、またはコンソールでテンプレートを指定します。コンソールは検証を自動的に実行します。ただし、これらのメソッドはテンプレートの構文のみを検証し、リソースのために指定したプ

プロパティ値がそのリソースについて有効であることを検証しません。より複雑な検証やベストプラクティスの確認には、[CloudFormation Linter \(cfn-lint\)](#) および [CloudFormation Rain \(rain fmt\)](#) などの追加のツールを使用することもできます。

Note

CloudFormation は、ユーザーが実行しようとする特定の変更を許可せず、変更が許可されないことをユーザーに通知することがあります。ただし、デフォルトで CloudFormation によるリソースの作成と更新の順序は不確定であるため、このメッセージは非同期的に発生する可能性があります。

スタックリソースの更新動作を理解する

更新を送信すると、CloudFormation はスタックの現在のテンプレートと送信された内容の違いに基づいてリソースを更新します。変更されていないリソースは、更新プロセス中も中断されることなく実行されます。更新されたリソースの場合、CloudFormation では以下のいずれかの更新動作を使用します。

中断を伴わない更新

CloudFormation は、リソースの使用を中断することなく、またリソースの物理 ID を変更することなく、リソースを更新します。たとえば、[AWS::CloudTrail::Trail](#) リソースの特定のプロパティを更新すると、CloudFormation は中断せずに証跡を更新します。

一時的な中断を伴う更新

CloudFormation は、一部の実行を中断してリソースの更新を行います。例えば、[AWS::EC2::Instance](#) リソースの特定のプロパティを更新すると、CloudFormation や Amazon EC2 でインスタンスを再設定する際にインスタンスが一時的に中断される場合があります。

置換

CloudFormation は更新の際にリソースを再作成し、新しい物理 ID も生成されます。CloudFormation は、通常まず置換リソースを作成し、他の従属するリソースからの参照が置換リソースを指すように変更してから、古いリソースを削除します。たとえば、[AWS::EC2::Instance](#) リソースタイプの AvailabilityZone プロパティを更新すると、CloudFormation によって新しいリソースが作成され、現在の EC2 インスタンスリソースと置き換えられます。

置換が必要なプロパティを追加または削除すると、更新もトリガーされます。更新は、プロパティの実際の値が変更されなくても、発生します。

CloudFormation が使用するメソッドは、特定のリソースタイプに対してどのプロパティを更新するかによって異なります。各プロパティの更新動作は、「[AWS リソースおよびプロパティタイプのリファレンス](#)」に記載されています。

更新動作によって、リソースを変更するタイミングを決定し、アプリケーションに対する変更の影響を軽減できます。特に、更新中にリソースが置き換えられる必要があるタイミングを計画することができます。たとえば、[AWS::RDS::DBInstance](#) リソースタイプの Port プロパティを更新すると、CloudFormation によって新しい DB インスタンスが作成され、ポート設定が更新されて古い DB インスタンスが削除されることで、DB インスタンスが置き換えられます。更新の前に、以下を実行してデータベースの置き換えに対して準備することをお勧めします。

- 現在のデータベースのスナップショットを作成します。
- DB インスタンスが置き換えられる場合、その DB インスタンスを使用するアプリケーションが中断をどのように処理するかについての戦略を準備します。
- その DB インスタンスを使用するアプリケーションが、更新されたポート設定や、その他に行われた更新を考慮することを確認します。
- DB スナップショットを使用して、新しい DB インスタンスにデータベースを復元します。

この例は完全なものではありませんが、更新中にリソースが置き換えられる場合に考慮する事項について紹介するためのものです。

Note

テンプレートに[ネストされたスタック](#)が 1 つ以上含まれる場合、CloudFormation もすべてのネストされたスタックに対して更新を開始します。これは、ネストされたスタックが変更されているかどうかを判定するために必要です。CloudFormation はネストされたスタック内の、対応するテンプレートで指定した変更があったリソースのみを更新します。

変更セットを使用して CloudFormation スタックを更新する

スタックを更新する必要がある場合は、変更の実装前に実行中のリソースに与える影響を理解することで、安心してスタックを更新できます。変更セットは、スタックに対して提案された変更が

実行中のリソースにおよぼす影響を事前に確認できるようにします。これには、リソースのプロパティや属性に対する影響が含まれます。変更が重要なリソースを削除するか置き換えるかにかかわらず、CloudFormation が変更を行うのはユーザーが変更セットの実行を決定した場合だけです。そのため、提案された変更をそのまま続行するか、別の変更セットを作成して他の変更を検討するかをユーザーが決定できます。変更セットは、CloudFormation コンソール、AWS CLI、または CloudFormation API を使用して作成および管理できます。

トピック

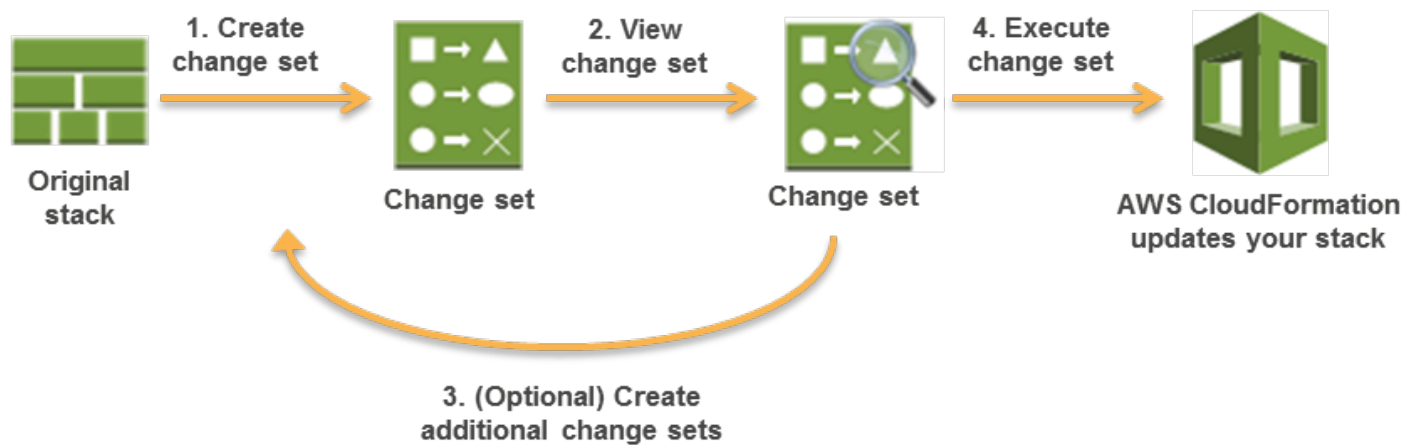
- [CloudFormation スタックの変更セットを作成する](#)
- [CloudFormation スタックの変更セットを表示する](#)
- [ドリフト対応変更セットの使用](#)
- [CloudFormation スタックの変更セットを実行する](#)
- [CloudFormation スタックの変更セットを削除する](#)
- [CloudFormation スタックの変更セットの例](#)
- [ネストされたスタックの変更セット](#)

Important

変更セットでは、CloudFormation によるスタックの更新が正常に行われるかどうかはわかりません。例えば、アカウントクォータを超過する、更新をサポートしていないリソースを更新しようとしている、リソースの変更に必要な許可が足りていないなど、スタックの更新が失敗する原因になる可能性があるものを、変更セットでは確認しません。更新が失敗した場合、CloudFormation では元の状態にリソースをロールバックするように試みます。

変更セットの概要

次の図は、変更セットを使用したスタックの更新方法をまとめたものです。



1. 更新するスタックの変更内容を送信して変更セットを作成します。変更したスタックテンプレートまたは変更した入力パラメータ値を送信できます。CloudFormation で送信された変更とスタックが比較され、変更セットが生成されます。この時点ではスタックは変更されません。
2. 変更セットを参照して、変更されるスタック設定やリソースを確認します。例えば、CloudFormation によって追加、変更、または削除されるリソースを確認できます。さらに、CloudFormation が変更するタグなど、リソースのプロパティと属性の変更前と変更後の比較を確認することもできます。
3. オプション: 実行する変更を決定する前に他の変更を検討する場合は、追加で変更セットを作成します。複数の変更セットを作成することで、異なる変更がリソースとプロパティにどのように影響するかを理解し、評価することができます。変更セットは必要な数だけ作成できます。
4. スタックに適用する変更を含む変更セットを実行します。CloudFormation はこれらの変更でスタックを更新します。

Note

変更を実行すると、スタックに関連付けられた変更セットは CloudFormation によってすべて削除されます。更新されたスタックには適用できないためです。

適用しない変更セットを実行してしまわないように、変更セットを削除することもできます。

CloudFormation スタックの変更セットを作成する

実行中のスタックの変更セットを作成するには、変更するテンプレート、新しい入力パラメータ値、またはその両方を提供して、変更内容を送信します。CloudFormation は、スタックを提出した変更と比較することにより、変更セットを生成します。

変更セットを作成する前、または変更セットの作成中にテンプレートを変更できます。

Create a change set (console)

変更セットを作成するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. [Stacks] ページで、変更セットを作成する実行中のスタックを選択します。
4. スタックの詳細ペインで、[スタックアクション]、[変更セットの作成] を順に選択します。
5. [スタック **stack-name** の変更セットの作成] ページで、次のいずれかを実行して入力パラメータ値を変更し、更新されたテンプレートの場所を指定するか、テンプレートを変更します。

タスク	アクション
入力パラメータ値を変更するには	[既存のテンプレートを使用する] を選択してから [次へ] を選択して、入力パラメータ値の入力または変更に進みます。
更新されたテンプレートの場所を指定するには	テンプレートを変更した場合は、[既存のテンプレートを置き換える] を選択してから、次のいずれかを実行します。 • Amazon S3 バケットに保存されているテンプレートの場合は、[Amazon S3 URL] を選択します。テンプレートの URL を入力するか貼り付けて、[次へ] を選択します。 バージョニング対応バケットにテンプレートがある場合は、URL に ?versionId= version-id を付加することでテンプレートの具体的なバージョンを指定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バージョニングが有効なバケット内のオブジェクトの使用」を参照してください。 • ローカルコンピュータに保存されているテンプレートの場合は、[テンプレートファイルをアップロード] を選択します。[ファイルの選択] を選択してファイルの場所に移動し、ファイルを選択して、[次へ] を選択します。

タスク	アクション
テンプレートを変更するには	テンプレートを変更していない場合は、[Infrastructure Composer でテンプレートを編集する] を選択してから、[Infrastructure Composer で編集] を選択します。AWS Infrastructure Composer にリダイレクトされます。テンプレートを編集したら、[変更セットを作成] を選択し、次に [確認して CloudFormation に移動する] を選択して [stack-name] の変更セットを作成] ページに戻り、[次へ] を選択します。

6. [スタックの詳細を指定] ページで変更セットの名前を指定してから、[概要] ページでその目的を特定するための変更セットの説明をオプションで入力します。テンプレートにパラメータが含まれている場合、[Specify stack details (スタックの詳細を指定)] ページで、適用可能な入力パラメータ値を入力または変更し、[次へ] を選択します。

スタックのテンプレートを再利用する場合は、CloudFormation によって各パラメータにスタックの現在の値が入力されています。ただし、NoEcho 属性で宣言されているパラメータは除きます。これらのパラメータに既存の値を使用するには、[既存の値の使用] を選択します。

NoEcho を使用して機密情報をマスクする方法、および動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」ベストプラクティスを参照してください。

7. [スタックオプションの設定] ページで、スタックのタグ、IAM サービスロール、スタックポリシー、ロールバック構成、Amazon SNS 通知トピック (該当する場合)、または変更セットを変更します。

 Note

ネストされたスタックの変更セットは、デフォルトで有効になっています。これにより、テンプレートで指定されているすべての入れ子スタックの変更セットが作成されます。現在のスタックにのみ変更セットを作成するには、[Disabled] を選択します。ネストされたスタックの変更セットに関する詳細については、「[ネストされたスタックの変更セット](#)」を参照してください。

8. テンプレートに IAM リソースが含まれる場合、[機能] で [I acknowledge that CloudFormation might create IAM resources (CloudFormation が IAM リソースを作成する可能性があることを確認します)] を選択します。IAM リソースは、AWS アカウントのアクセス許可を変更でき

ます。これらのリソースを確認し、意図したアクションのみを許可していることを確認してください。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。

9. [次へ] を選択して続行します。
10. [Review **stack-name** (stack-name の確認)] ページで、この変更セットの変更を確認します。
11. [Submit] を選択してください。

変更セットの詳細ページの [変更] タブにリダイレクトされます。CloudFormation が変更セットを生成する間、変更セットのステータスは CREATE_PENDING になります。変更セットが作成されると、CloudFormation によってステータスが CREATE_COMPLETE に設定されます。[Changes] (変更) セクションに、CloudFormation によってスタックに対するすべての変更のリストが表示されます。詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。

[プロパティレベルの変更] 列で [詳細を表示] を選択して、プロパティレベルで行われた変更を表示します。

CloudFormation で変更セットの作成に失敗した場合 (FAILED ステータスが報告された場合)、[Status] (状況) フィールドに表示されたエラーを修正して、変更セットを再作成します。

12. 変更内容が正しいことを確認したら、[変更セットを実行] を選択します。

Create a change set for nested stacks (console)

ネストされたスタックの変更セットを作成するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョンを選択します。
3. [スタック] ページで、変更セットを作成する実行中のスタックを選択します。
4. スタックの詳細ペインで、[スタックアクション]、[変更セットの作成] を順に選択します。
5. [スタック **stack-name** の変更セットの作成] ページで、次のいずれかを実行して入力パラメータ値を変更し、更新されたテンプレートの場所を指定するか、テンプレートを変更します。

タスク	アクション
入力パラメータ値を変更するには	[既存のテンプレートを使用する] を選択してから [次へ] を選択して、入力パラメータ値の入力または変更に進みます。
更新されたテンプレートの場所を指定するには	テンプレートを変更した場合は、[既存のテンプレートを置き換える] を選択してから、次のいずれかを実行します。 - Amazon S3 バケットに保存されているテンプレートの場合は、[Amazon S3 URL] を選択します。テンプレートの URL を入力するか貼り付けて、[次へ] を選択します。 バージョニング対応バケットにテンプレートがある場合は、URL に <code>?versionId= <i>version-id</i></code> を付加することでテンプレートの具体的なバージョンを指定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バージョニングが有効なバケット内のオブジェクトの使用」を参照してください。 - ローカルコンピュータに保存されているテンプレートの場合は、[テンプレートファイルをアップロード] を選択します。[ファイルの選択] を選択してファイルの場所に移動し、ファイルを選択して、[次へ] を選択します。
テンプレートを変更するには	テンプレートを変更していない場合は、[Infrastructure Composer でテンプレートを編集する] を選択してから、[Infrastructure Composer で編集] を選択します。AWS Infrastructure Composer にリダイレクトされます。テンプレートを編集したら、[変更セットを作成] を選択し、次に [確認して CloudFormation に移動する] を選択して [<i>stack-name</i>] の変更セットを作成] ページに戻り、[次へ] を選択します。
6. [スタックの詳細を指定] ページで変更セットの名前を指定してから、[概要] ページでその目的を特定するための変更セットの説明をオプションで入力します。テンプレートにパラメータが含まれている場合、[Specify stack details (スタックの詳細を指定)] ページで、適用可能な入力パラメータ値を入力または変更し、[次へ] を選択します。	

スタックのテンプレートを再利用する場合は、CloudFormation によって各パラメータにスタックの現在の値が入力されています。ただし、NoEcho 属性で宣言されているパラメータは除きます。これらのパラメータに既存の値を使用するには、[既存の値の使用] を選択します。

NoEcho を使用して機密情報をマスクする方法、また動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」ベストプラクティスを参照してください。

7. [スタックオプションの設定] ページで、スタックのタグ、IAM サービスロール、スタックポリシー、ロールバック構成、Amazon SNS 通知トピック (該当する場合)、または変更セットを変更します。詳細については、「[スタックオプションを設定する](#)」を参照してください。

 Note

ネストされたスタックの変更セットは、デフォルトで有効になっています。これにより、テンプレートで指定されているすべての入れ子スタックの変更セットが作成されます。ネストされたスタックの変更セットに関する詳細については、「[ネストされたスタックの変更セット](#)」を参照してください。

8. テンプレートに IAM リソースが含まれる場合、[機能] で [I acknowledge that CloudFormation might create IAM resources (CloudFormation が IAM リソースを作成する可能性があることを確認します)] を選択します。IAM リソースは、AWS アカウントのアクセス許可を変更できます。これらのリソースを確認し、意図したアクションのみを許可していることを確認してください。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
9. [次へ] を選択して続行します。
10. [Review **stack-name** (stack-name の確認)] ページで、この変更セットの変更を確認します。
11. [Submit] を選択してください。

 Note

CloudFormation プロパティレベルの変更セットは、ネストされたスタックの変更セットを作成するときに、クロススタックリファレンスを解決しません。変更セット

は、親スタックの出力を参照し、親スタックが変更された場合、条件付き置換のために子スタックのリソースをマークできます

変更セットの詳細ページの [変更] タブにリダイレクトされます。CloudFormation が変更セットを生成する間、変更セットのステータスは CREATE_PENDING になります。変更セットが作成されると、CloudFormation によってステータスが CREATE_COMPLETE に設定されます。[Changes] (変更) セクションに、CloudFormation によってスタックに対するすべての変更のリストが表示されます。詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。

CloudFormation で変更セットの作成に失敗した場合 (FAILED ステータスが報告された場合)、[Status] (状況) フィールドに表示されたエラーを修正して、変更セットを再作成します。

12. 変更内容が正しいことを確認したら、[変更セットを実行] を選択します。

変更セットを作成するには (AWS CLI)

- [create-change-set](#) コマンドを使用します。

変更はコマンドオプションとして送信します。新しいパラメータ値、修正済みのテンプレート、または両方を指定できます。たとえば、次のコマンドで SampleChangeSet スタックの MyStack という名前の変更セットが作成されます。変更セットは現在のスタックのテンプレートを使用しますが、Purpose パラメータの値が異なります。

```
aws cloudformation create-change-set --stack-name MyStack \  
  --change-set-name SampleChangeSet --use-previous-template \  
  --parameters \  
    ParameterKey="InstanceType",UsePreviousValue=true  
    ParameterKey="KeyPairName",UsePreviousValue=true  
    ParameterKey="Purpose",ParameterValue="production"
```

CloudFormation スタックの変更セットを表示する

変更セットの作成後、実行前に目的の変更を確認できます。変更セットの表示には CloudFormation コンソール、AWS CLI、または CloudFormation API を使用できます。CloudFormation コンソール

は、変更の概要、および JSON 形式の変更の詳細なリストを提供します。AWS CLI および AWS CloudFormation API は、JSON 形式の変更の詳細なリストを返します。

View a change set (console)

変更セットを表示するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョンを選択します。
3. [Stacks] ページで、表示する変更セットを含むスタックの名前を選択します。
4. ナビゲーションペインで、[変更セット] を選択して、スタックの変更セットのリストを表示します。
5. 表示する変更セットの名前を選択します。

CloudFormation コンソールに変更セットの詳細ページが表示されます。ここで変更セットの作成時刻、ステータス、変更セットの生成に使用された入力、および変更の要約を確認できます。

[Changes] (変更) セクションの各行は、CloudFormation が追加、変更、削除するリソースを表します。

- Add (追加) – スタックのテンプレートにリソースを追加すると、CloudFormation によってリソースが作成されます。
- Modify (変更) – スタックのテンプレート内のリソースのプロパティを変更すると、CloudFormation はリソースを変更します。
- Remove (削除) – スタックのテンプレートから既存のリソースを削除すると、CloudFormation によってリソースが削除されます。

Note

変更によってリソースが中断されたり置き換え (再作成) されたりすることがあります。リソースの更新動作の詳細については、[スタックリソースの更新動作を理解する](#)を参照してください。

特定の変更に注目するには、フィルタビューを使用します。たとえば、AWS::EC2::Instance などの特定のリソースタイプをフィルタリングします。特定のリソースをフィルタリングするには、myWebServer や i-123abcd4 のような論理 ID または物理 ID を指定します。

6. [変更] セクションで、[プロパティレベルの変更] 列にある [詳細を表示] を選択して、リソースに対して行われたプロパティ値の変更を表示します。
7. CloudFormation コンソールがリソースのプロパティレベルの変更ページを表示します。このページでは、変更セットを実行する前のリソースのテンプレート設定と、変更セットを実行した後のテンプレート設定を確認できます。

[プロパティレベルの変更] セクションの表には、影響を受けるプロパティの [パス]、[変更タイプ]、[変更前の値]、および [変更後の値] が表示されます。この表で、テンプレートの [変更前] ビューと [変更後] ビューで強調表示したい各変更のチェックボックスを選択して、プロパティレベルでどのような変更が行われるのかを確認します。

- 追加 – 追加されたプロパティは緑色で強調表示されます。
- 変更 – 変更されたプロパティは青色で強調表示されます。
- 削除 – 削除されたプロパティは赤で強調表示されます。

View a change set for nested stack (console)

ネストされたスタックの変更セットを表示するには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョンを選択します。
3. [Stacks] ページで、表示する変更セットを含むスタックの名前を選択します。
4. ナビゲーションペインで [変更セット] を選択して、スタックの変更セットのリストを表示します。
5. 表示する変更セットの名前を選択します。

CloudFormation コンソールに変更セットの詳細ページが表示されます。ここで変更セットの作成時刻、ステータス、変更セットの生成に使用された入力、および変更の要約を確認できます。

[Changes] (変更) セクションの各行は、CloudFormation が動的な状態を追加、変更、削除、表示するリソースを表します。

- Add (追加) – スタックのテンプレートにリソースを追加すると、CloudFormation によってリソースが作成されます。
- Modify (変更) – スタックのテンプレート内のリソースのプロパティを変更すると、CloudFormation はリソースを変更します。
- Remove (削除) – スタックのテンプレートから既存のリソースを削除すると、CloudFormation によってリソースが削除されます。
- Dynamic – CloudFormation はネストされたスタックのテンプレートから正確なリソース変更アクションを特定できません。

 Note

変更によってリソースが中断されたり置き換え (再作成) されたりすることがあります。リソースの更新動作の詳細については、[スタックリソースの更新動作を理解する](#)を参照してください。

特定の変更に注目するには、フィルタビューを使用します。たとえば、**AWS::CloudFormation::Stack** などの特定のリソースタイプをフィルタリングします。特定のリソースをフィルタリングするには、**DeadLetterQueue** や **NestedStack** のような論理 ID または物理 ID を指定します。

6. [Changes] セクションで、表示するネストされた変更セットの [View nested change set] を選択します。

CloudFormation コンソールによって、ネストされた変更セットの詳細ページへ移動します。[Go to root change set] を選択してルート変更セットを表示するか、[View parent change set] を選択して親変更セットを表示できます。詳細については、「[ネストされたスタックの変更セット](#)」を参照してください。

 Note

CloudFormation プロパティレベルの変更セットは、ネストされたスタックの変更セットを作成するときに、クロススタックリファレンスを解決しません。変更セット

は、親スタックの出力を参照し、親スタックが変更された場合、条件付き置換のために子スタックのリソースをマークできます

変更セットを表示するには (AWS CLI)

1. 変更セットの ID を取得するには、[change-sets](#) コマンドを実行します。

次の例のように、表示する変更セットがあるスタックの名前を指定します。

```
aws cloudformation list-change-sets --stack-name MyStack
```

CloudFormation から、次のような変更セットのリストが返されます。

```
{
  "Summaries": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
      "Status": "CREATE_COMPLETE",
      "ChangeSetName": "SampleChangeSet",
      "CreationTime": "2020-11-18T20:44:05.889Z",
      "StackName": "MyStack",
      "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
    },
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
      "Status": "CREATE_COMPLETE",
      "ChangeSetName": "SampleChangeSet-conditional",
      "CreationTime": "2020-11-18T21:15:56.398Z",
      "StackName": "MyStack",
      "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet-conditional/1a2345b6-0000-00a0-a123-00abc0abc000"
    },
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
      "Status": "CREATE_COMPLETE",
      "ChangeSetName": "SampleChangeSet-replacement",

```

```

        "CreationTime": "2020-11-18T21:03:37.706Z",
        "StackName": "MyStack",
        "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet-replacement/1a2345b6-0000-00a0-a123-00abc0abc000"
    }
]
}

```

2. 表示する変更セットの ID を指定して [describe-change-set](#) コマンドを実行します。例えば、次のようになります。

```

aws cloudformation describe-change-set \
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000

```

CloudFormation から、指定された変更セットに関する情報が返されます。

```

{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
  "Status": "CREATE_COMPLETE",
  "ChangeSetName": "SampleChangeSet-direct",
  "Parameters": [
    {
      "ParameterValue": "testing",
      "ParameterKey": "Purpose"
    },
    {
      "ParameterValue": "ellioty-useast1",
      "ParameterKey": "KeyPairName"
    },
    {
      "ParameterValue": "t2.micro",
      "ParameterKey": "InstanceType"
    }
  ],
  "Changes": [
    {
      "ResourceChange": {
        "ResourceType": "AWS::EC2::Instance",
        "PhysicalResourceId": "i-1abc23d4",
        "Details": [

```

```
        {
            "ChangeSource": "DirectModification",
            "Evaluation": "Static",
            "Target": {
                "Attribute": "Tags",
                "RequiresRecreation": "Never"
            }
        },
        "Action": "Modify",
        "Scope": [
            "Tags"
        ],
        "LogicalResourceId": "MyEC2Instance",
        "Replacement": "False"
    },
    "Type": "Resource"
}
],
"CreationTime": "2020-11-18T23:35:25.813Z",
"Capabilities": [],
"StackName": "MyStack",
"NotificationARNs": [],
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-direct/9edde307-960d-4e6e-ad66-b09ea2f20255"
}
```

--include-property-values を describe-change-set とともに使用して、プロパティレベルの変更を一覧表示します。

Changes キーリストはリソースを変更します。この変更セットを実行した場合、CloudFormation によって i-1abc23d4 EC2 インスタンスのタグが更新されます。各フィールドの説明については、「AWS CloudFormation API リファレンス」の「[Change](#) データ型」を参照してください。

変更セットのその他の例については、[CloudFormation スタックの変更セットの例](#)を参照してください。

変更セット内のプロパティレベルの変更を表示する (AWS CLI)

- 以下のコマンドは、Ipv4Prefixes プロパティを削除し、リソースの Description を変更して、Tag を追加する、AWS::EC2::NetworkInterface リソースの変更セットに関連するプロパティレベルの変更を一覧表示します。

```
aws cloudformation describe-change-set --include-property-values \  
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/  
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841
```

以下は出力の例です。

```
"ChangeSetName": "ExampleChangeSet",  
  "ChangeSetId": "arn:aws:cloudformation:us-east-1:803642222207:changeSet/  
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",  
  "StackId": "arn:aws:cloudformation:us-east-1:803642222207:stack/ExampleStack/  
ab664180-f686-11ee-9e29-12cd92393671",  
  "StackName": "ExampleStack",  
  "Description": null,  
  "Parameters": null,  
  "CreationTime": "2024-04-09T18:04:59.935000+00:00",  
  "ExecutionStatus": "AVAILABLE",  
  "Status": "CREATE_COMPLETE",  
  "StatusReason": null,  
  "NotificationARNs": [],  
  "RollbackConfiguration": {  
    "RollbackTriggers": []  
  },  
  "Capabilities": [],  
  "Tags": null,  
  "ParentChangeSetId": null,  
  "IncludeNestedStacks": true,  
  "RootChangeSetId": null,  
  "OnStackFailure": null,  
  {  
    "Changes": [  
      {  
        "Type": "Resource",  
        "ResourceChange": {  
          "Action": "Modify",  
          "LogicalResourceId":  
"EC2NetworkInterface00eni067fd35b649a05b7100Tpyls",
```

```
"PhysicalResourceId": "eni-067fd35b649a05b71",
"ResourceType": "AWS::EC2::NetworkInterface",
"Replacement": "False",
"Scope": [
    "Properties",
    "Tags"
],
"Details": [
    {
        "Target": {
            "Attribute": "Properties",
            "Name": "Ipv4Prefixes",
            "RequiresRecreation": "Never",
            "Path": "/Properties/Ipv4Prefixes",
            "BeforeValue": "[]",
            "AttributeChangeType": "Remove"
        },
        "Evaluation": "Static",
        "ChangeSource": "DirectModification"
    },
    {
        "Target": {
            "Attribute": "Properties",
            "Name": "Description",
            "RequiresRecreation": "Never",
            "Path": "/Properties/Description",
            "BeforeValue": "",
            "AfterValue": "Description",
            "AttributeChangeType": "Modify"
        },
        "Evaluation": "Static",
        "ChangeSource": "DirectModification"
    },
    {
        "Target": {
            "Attribute": "Tags",
            "RequiresRecreation": "Never",
            "Path": "/Properties/Tags/0",
            "AfterValue": "{\"Key\":\"Test\",\"Value\":\"Test\"}",
            "AttributeChangeType": "Add"
        },
        "Evaluation": "Static",
        "ChangeSource": "DirectModification"
    }
]
```

```

    ],
    "BeforeContext": "{ \"Properties\": { \"Description\": \"\",
    \"PrivateIpAddress\": \"172.31.76.2\", \"PrivateIpAddresses\": [ { \"PrivateIpAddress
    \": \"172.31.76.2\", \"Primary\": \"true\" } ], \"SecondaryPrivateIpAddressCount\": \"0\",
    \"Ipv6PrefixCount\": \"0\", \"Ipv4Prefixes\": [], \"Ipv4PrefixCount\": \"0\", \"GroupSet
    \": [ \"sg-05a45689b1059e82d\" ], \"Ipv6Prefixes\": [], \"SubnetId\": \"subnet-455e8969\",
    \"SourceDestCheck\": \"true\", \"InterfaceType\": \"interface\", \"Tags\": [] },
    \"UpdateReplacePolicy\": \"Retain\", \"DeletionPolicy\": \"Retain\" }",
    "AfterContext": "{ \"Properties\": { \"Description\": \"Description\",
    \"PrivateIpAddress\": \"172.31.76.2\", \"PrivateIpAddresses\": [ { \"PrivateIpAddress
    \": \"172.31.76.2\", \"Primary\": \"true\" } ], \"SecondaryPrivateIpAddressCount
    \": \"0\", \"Ipv6PrefixCount\": \"0\", \"Ipv4PrefixCount\": \"0\", \"GroupSet\":
    [ \"sg-05a45689b1059e82d\" ], \"Ipv6Prefixes\": [], \"SubnetId\": \"subnet-455e8969\",
    \"SourceDestCheck\": \"true\", \"InterfaceType\": \"interface\", \"Tags\": [ { \"Value\":
    \"Test\", \"Key\": \"Test\" } ] }, \"UpdateReplacePolicy\": \"Retain\", \"DeletionPolicy\":
    \"Retain\" }"
  }
}
],
"ChangeSetName": "ExampleChangeSet",
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",
"StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/ExampleStack/
ab664180-f686-11ee-9e29-12cd92393671",

```

ドリフト対応変更セットの使用

ドリフト対応変更セットは、スタックドリフトを安全な方法で特定および管理できる強化された CloudFormation 変更セットです。基盤となるサービスの AWS マネジメントコンソール、AWS CLI、または SDK を使用して、スタックリソースが CloudFormation の外部で変更されると、スタックは予想されるテンプレート設定からドリフトします。ドリフト対応変更セットは、テンプレートをスタックリソースの実際の状態と比較し、ドリフトされたリソースをテンプレート定義に合わせます。リソースのテンプレート定義が実際の状態と一致するように更新されると、ドリフト対応変更セットは、リソースを変更せずにリソースのドリフトステータスをリセットします。

ドリフト対応変更セットを使用するメリットは次のとおりです。

- ドリフトの上書きのプレビュー – インシデント対応中に行われた重要な帯域外の変更をデプロイが元に戻すかどうかを検出します。

- 体系的なドリフト調整 – ドリフトされたプロパティをテンプレート値で上書きするか、実際のリソースの状態に合わせてテンプレートを更新するかのいずれかで、ドリフトされたリソースを準拠したテンプレート定義と安全に調整します。
- 拡張ロールバック機能 – デプロイが失敗した場合、リソースを以前のテンプレート状態ではなく、実際のデプロイ前の状態に復元します。
- [完全な可視性] – 変更を行う前に、デプロイが実際のリソース設定にどのように影響するかを正確に確認します。

従来の変更セットは、新しいテンプレートとスタックの以前のテンプレートとの比較を提供しますが、スタックドリフトについては考慮しません。ドリフト対応変更セットは、次の間の 3 方向比較を提供することで、この問題を解決します。

- [実際の状態] – リソースのライブ設定。CloudFormation は、基盤となるサービス API 経由で、変更セットの作成時にアカウントのリソースの設定を読み取ります。
- [以前のデプロイ状態] – 前回の CloudFormation デプロイからテンプレートで定義された設定。
- 必要な状態 – 新しいテンプレートで定義された設定。

ドリフト対応変更セットは、テンプレートでリソースが明示的に変更されていない場合でも、すべてのスタックリソースの実際の状態を必要な状態に合わせて更新します。

考慮事項

- リソースタイプのサポート – ドリフト対応変更セットは、何百ものリソースタイプの必要な状態と実際の状態との比較をサポートします。サポートされていないリソースタイプの場合、ドリフト対応変更セットは、以前のデプロイ状態と必要な状態との比較にフォールバックします。詳細については、「サポートされているリソースタイプ」セクションを参照してください。
- 書き込み専用プロパティ – 機密データ (パスワード、シークレット) を含むプロパティの場合、ドリフト対応変更セットは、実際の値ではなく以前のデプロイ値と比較されます。
- AWS マネージドプロパティ – ドリフト対応変更セットは、Auto Scaling グループの必要な容量など、AWS のサービスによってマネージドプロパティに対して行われた帯域外の変更を保持します。詳細については、「AWS マネージドプロパティ」セクションを参照してください。
- 外部タグキー – ドリフト対応変更セットは、テンプレートで指定されていないタグキーを削除または変更しないため、属性ベースのアクセス制御 (ABAC) システムとの競合を防ぎます。
- ドリフトしたリソースの置き換え – ドリフト対応変更セットは、イミュータブルプロパティのドリフト調整をサポートしていません。

- クロススタックアタッチメント – `AWS::IAM::Policy` などの一部のリソースタイプは、`AWS::IAM::Role` などの他のリソースタイプの実際の状態に影響を与える可能性があります。ドリフト対応変更セットは、スタック内のリソースのアタッチを処理します。別のスタックからのセカンダリリソースのアタッチを介してリソースが変更された場合、ドリフト対応変更セットは変更をドリフトとして検出し、アタッチを元に戻すことができます。一般的なアタッチ可能なリソースには `AWS::IAM::Policy`、`AWS::IAM::ManagedPolicy`、`AWS::EC2::SecurityGroupIngress`、`AWS::` などがあります。

AWS マネージドプロパティ

アクティブな AWS 管理用に、特定のリソースプロパティを設定できます。例えば、Amazon Relational Database Service (Amazon RDS) が Amazon RDS テーブルのマイナーエンジンバージョンを自動的に更新することを許可できます。これらの変更は、CloudFormation でスタックドリフトとして表示される可能性があります。ドリフト対応変更セットは、AWS マネージドプロパティでドリフトが予想されることを認識し、テンプレートでプロパティを変更していない場合は実際の値をそのままにします。AWS マネージドプロパティの上位の例は次のとおりです。

- Amazon RDS テーブルの `AutoMinorVersionUpgrade` プロパティを有効にして、エンジンバージョンの自動更新を許可する。
- `AWS::ApplicationAutoScaling::ScalableTarget` リソースを使用して、Amazon DynamoDB テーブルの読み取り/書き込みキャパシティーユニットや Amazon Elastic Container Service クラスターの必要数などのプロパティの自動スケーリングを有効にする。
- Auto Scaling グループの `AWS::AutoScaling::ScalingPolicy` を使用する。

ドリフト対応変更セットは、AWS マネージドとして特定されたプロパティを明確にします。詳細については、「AWS CLI」セクションを参照してください。

ドリフト対応変更セットの使用 (コンソール)

デプロイモードの追加オプションとともに、従来の変更セットと同じワークフローを使用して、CloudFormation コンソールを通じてドリフト対応変更セットを作成および管理できます。

ドリフト対応変更セットの作成

ドリフト対応変更セットを作成するには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページで、更新する実行中のスタックを選択します。
3. [スタックアクション] を選択し、[現在のスタックの変更セットを作成する] を選択します。
4. [変更セットを作成する] ページの [変更セットタイプ] で、[ドリフト対応変更セット] を選択します。
5. [変更セットの作成] を選択します。
6. 変更セットの詳細ページで、影響を受ける各リソースの実際のデプロイ、以前のデプロイ、および必要な状態を示す 3 方向比較を確認します。
7. 変更満足したら、[変更セットを実行する] を選択します。

ドリフト対応変更セットの確認

コンソールでドリフト対応変更セットを表示すると、従来の変更セットと比較して強化された情報が表示されます。

- [スタックドリフトステータス] – スタックが前回のデプロイからドリフトしたかどうかを示します。
- プロパティ比較 – 影響を受けるリソースの実際の状態と必要な状態の間の JSON 差分を示します。
- [ドリフトインジケータ] – ドリフトした JSON 差分内のプロパティを明確にマークします。[ドリフトを表示する] をクリックして、プロパティの以前のデプロイ値を表示します。
- 値ソースインジケータ – プロパティの before 値が実際の状態からのものか以前のデプロイ状態からのものかを示します。

ドリフト対応変更セットの使用 (AWS CLI)

create-change-set コマンドに `--deployment-mode REVERT_DRIFT` パラメータを追加することで、AWS CLI を使用してドリフト対応変更セットを作成および管理できます。

ドリフト対応変更セットの作成

ドリフト対応変更セットを作成するには

[create-change-set](#) コマンドを `--deployment-mode REVERT_DRIFT` パラメータとともに使用して、ドリフト対応変更セットを作成します。

```
aws cloudformation create-change-set \  
  --stack-name my-stack \  
  --change-set-name my-drift-aware-changeset \  
  --template-body file://updated-template.yaml \  
  --deployment-mode REVERT_DRIFT \  
  --capabilities CAPABILITY_IAM
```

ドリフト対応変更セットの確認

ドリフト対応変更セットの詳細を確認するには

[describe-change-set](#) コマンドを使用して、ドリフト対応変更セットの詳細を確認します。

```
aws cloudformation describe-change-set \  
  --change-set-name my-drift-aware-changeset \  
  --stack-name my-stack
```

レスポンスには、ドリフト対応変更セットの拡張情報が含まれます。

- `StackDriftStatus` – スタックがドリフトしたかどうかを表示します (DRIFTED、IN_SYNC、NOT_CHECKED、または UNKNOWN)。
- `ResourceDriftStatus` – 各リソース (DELETED、MODIFIED、IN_SYNC、または NOT_CHECKED) のドリフトステータスを表示します。
- `BeforeValueFrom` – プロパティの before 値が ACTUAL_STATE からのものか PREVIOUS_DEPLOYMENT_STATE からのものかを示します。
- `Drift` – PreviousValue、ActualValue、DriftDetectionTimestamp などのプロパティのドリフトの詳細が含まれます。
- `[ResourceDriftIgnoredProperties]` – 変更セットがドリフトを元に戻さないリソースのプロパティと、ドリフトを無視する理由が含まれます。

詳細については、「AWS CloudFormation API リファレンス」の「[DescribeChangeSet](#)」を参照してください。

ドリフト対応変更セットの実行

ドリフト対応変更セットを実行するには

変更セットを確認したら、[execute-change-set](#) コマンドを使用して変更を適用します。

```
aws cloudformation execute-change-set \  
  --change-set-name my-drift-aware-changeset \  
  --stack-name my-stack
```

デプロイが失敗した場合、CloudFormation はリソースを実際のデプロイ前状態にロールバックし、前回のデプロイ以降に行われた帯域外の変更を保持します。

サポートされているリソースタイプ

ドリフト対応変更セットは、ドリフト検出をサポートするすべての[リソースタイプ](#)について、実際の状態、以前のデプロイ状態、および必要な状態の 3 方向比較をサポートしますが、次のリソースタイプは除きます。

- AWS::SageMaker::MlflowTrackingServer
- AWS::Route53Resolver::FirewallRuleGroup
- AWS::MediaLive::Multiplexprogram
- AWS::VpcLattice::ResourceConfiguration
- AWS::S3::StorageLensGroup
- AWS::Bedrock::AgentAlias
- AWS::MSK::Cluster
- AWS::RDS::DBProxy
- AWS::Redshift::ClusterParameterGroup
- AWS::QBusiness::Index
- AWS::NetworkManager::CoreNetwork
- AWS::IAM::OIDCProvider
- AWS::Organizations::ResourcePolicy
- AWS::SNS::TopicInlinePolicy
- AWS::Route53::KeySigningKey
- AWS::DataZone::PolicyGrant
- AWS::Transfer::Certificate
- AWS::SageMaker::ImageVersion
- AWS::Neptune::DBParameterGroup

- AWS::ODB::CloudVmCluster
- AWS::RolesAnywhere::TrustAnchor
- AWS::Detective::Graph
- AWS::Maester::DocumentType
- AWS::SageMaker::ModelPackageGroup
- AWS::S3Express::BucketPolicy
- AWS::Panorama::PackageVersion
- AWS::S3Tables::TableBucketPolicy

ドリフト対応変更セットは、3 方向比較をサポートしていないリソースの以前のデプロイ状態と必要な状態の比較にフォールバックします。

CloudFormation スタックの変更セットを実行する

変更セットに記述された変更をスタックに加えるには、変更セットを実行します。

Important

変更セットを実行すると、スタックに関連付けられた変更セットは CloudFormation によってすべて削除されます。更新されたスタックに対して有効でなくなるためです。更新が失敗した場合は、新しい変更セットを作成する必要があります。

スタックポリシーおよび変更セットの実行

スタックポリシーが関連付けられているスタックで変更セットを実行する場合、更新の際に CloudFormation によってそのポリシーがスタックに適用されます。変更セットの実行時に、既存のポリシーを上書きする一時的なスタックポリシーを指定することはできません。保護されたリソースを更新するには、スタックポリシーを更新するか、直接更新方式を使用する必要があります。詳細については、「[スタックを直接更新する](#)」を参照してください。

Execute a change set (console)

変更セットを実行するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。

2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. [Stacks] ページで、更新するスタックの名前を選択します。
4. ナビゲーションペインで [変更セット] を選択して、スタックの変更セットのリストを表示します。
5. 実行する変更セットの名前を選択します。
6. 変更セットの詳細ページで、[変更セットを実行] を選択します。

CloudFormation ですぐにスタックの更新が開始されます。CloudFormation コンソールによって [Events] (イベント) タブが表示されます。このタブで、スタックの更新の進捗状況をモニタリングできます。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

Execute a change set for nested stacks (console)

ネストされたスタックの変更セットを実行するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. [Stacks] ページで、更新するスタックの名前を選択します。ルート変更セットに関連付けられたスタック名を選択する必要があります。
4. ナビゲーションペインで [変更セット] を選択して、スタックの変更セットのリストを表示します。
5. 実行するルート変更セットの名前を選択します。
6. 変更セットの詳細ページで、[変更セットを実行] を選択します。

Note

CloudFormation は、[CloudFormation スタックの変更セットを作成する](#) プロセス中にネストされたスタックの変更セットに対して [Enabled] (有効) が選択されている場合、ルート変更セットおよびネストされた変更セットに記述された変更を実行します。

CloudFormation ですぐにスタックの更新が開始されます。CloudFormation コンソールによって [Events] (イベント) タブが表示されます。このタブで、スタックの更新の進捗状況を

モニタリングできます。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

変更セットを実行するには (AWS CLI)

- [execute-change-set](#) コマンドを実行します。

次の例に示すように、実行する変更セットの変更セット ID を指定します。

```
aws cloudformation execute-change-set \  
  --change-set-name \  
    arn:aws:cloudformation:us-east-1:123456789012:changeSet/  
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

例のコマンドは、ID `arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000` の変更セットを実行します。

コマンドを実行すると、CloudFormation でスタックの更新が開始されます。スタックの進行状況を表示するには、[describe-stacks](#) コマンドを使用します。

CloudFormation スタックの変更セットを削除する

変更セットを削除すると、スタックの変更セットのリストから削除されます。変更セットを削除することで、自身や別のユーザーが適用してはいけない変更セットを意図せず実行してしまうことを防止できます。削除しない限り、スタックを更新するまで、CloudFormation によってすべての変更セットが保持されます。

Delete a change set

変更セットを削除するには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョンを選択します。
3. [Stacks] ページで、削除する変更セットを含むスタックの名前を選択します。
4. ナビゲーションペインで [変更セット] を選択して、スタックの変更セットのリストを表示します。

5. 削除する変更セットの名前を選択します。
6. 変更セットの詳細ページで、[変更セットの削除] を選択します。

CloudFormation によってすぐに、スタックの変更セットのリストから変更セットを削除する処理が開始され、[Stacks] (スタック) ページにリダイレクトされます。

Delete a change set for nested stacks (console)

ネストされたスタックの変更セットを削除するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. [Stacks] ページで、ルート変更セットに関連付けられたスタック名を選択します。
4. ナビゲーションペインで [変更セット] を選択して、スタックの変更セットのリストを表示します。
5. 削除する変更セットの名前を選択します。
6. 変更セットの詳細ページで、[削除] を選択します。[変更セットの削除] を選択すると、ネストされた変更セットの階層全体が削除されます。

Note

ネストされたスタックの変更セットに対する削除操作は非同期で、DELETE_PENDING ステータスに続いて DELETE_IN_PROGRESS ステータスが表示されます。変更セットの削除操作が完了すると、変更セットはリストから削除されます。REVIEW_IN_PROGRESS ステータス内のネストされたスタックは、変更セットの作成中に作成された場合でも削除されます。

CloudFormation は、スタックの変更セットリストから変更セットの削除を直ちに開始します。

Note

ネストされたスタックが進行中のオペレーションで停止した場合には、「[ネストされたスタックのロールバック失敗](#)」でエラーのトラブルシューティングを参照してください。

変更セットを削除するには (AWS CLI)

- 次の例に示すように、削除する変更セットの ID を指定して [delete-change-set](#) コマンドを実行します。

```
aws cloudformation delete-change-set \
  --change-set-name \
    arn:aws:cloudformation:us-east-1:123456789012:changeSet/
    SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

CloudFormation スタックの変更セットの例

このセクションでは、一般的なスタックの変更で CloudFormation によって作成される変更セットの例を示します。例では、テンプレートを直接編集する方法、単独の入力パラメータを変更する方法、バックアップされていないデータの損失またはスタックで実行中のアプリケーションの中断を防止するリソース再作成の計画、およびリソースの追加方法と削除方法が示されています。変更セットの機能を理解するために、送信された変更を説明して、その結果である変更セットについて解説します。各例は、お客様が前出の例を理解していることを前提に構築されているため、順を追って読むことをお勧めします。変更セットの各フィールドの説明については、「AWS CloudFormation API リファレンス」の「[変更データ型](#)」を参照してください。

変更セットの詳細は、[コンソール](#)、AWS CLI、または CloudFormation [DescribeChangeSet](#) API 操作を使用して表示できます。

以下の各変更セットは、以下の[サンプルテンプレート](#)を使用したスタックから生成されています。

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample EC2 instance template for testing change sets.",
  "Parameters" : {
    "Purpose" : {
```

```

    "Type" : "String",
    "Default" : "testing",
    "AllowedValues" : ["testing", "production"],
    "Description" : "The purpose of this instance."
  },
  "KeyPairName" : {
    "Type": "AWS::EC2::KeyPair::KeyName",
    "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
instance"
  },
  "InstanceType" : {
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "t2.small", "t2.medium"],
    "Description" : "The EC2 instance type."
  }
},
"Resources" : {
  "MyEC2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "KeyName" : { "Ref" : "KeyPairName" },
      "InstanceType" : { "Ref" : "InstanceType" },
      "ImageId" : "ami-8fcee4e5",
      "Tags" : [
        {
          "Key" : "Purpose",
          "Value" : { "Ref" : "Purpose" }
        }
      ]
    }
  }
}
}
}
}

```

テンプレートの直接編集

スタックのテンプレートでリソースを直接変更して変更セットを生成すると、この変更は CloudFormation で直接変更として分類されます。パラメータ値の更新に伴って開始された変更とはみなされません。以下の変更セットは、i-1abc23d4 インスタンスに新しいタグを追加する直接変更の例です。パラメータ値や機能など、その他の入力値はすべて変更しないため、Changes 構造ついて説明します。

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
  "Status": "CREATE_COMPLETE",
  "ChangeSetName": "SampleChangeSet-direct",
  "Parameters": [
    {
      "ParameterValue": "testing",
      "ParameterKey": "Purpose"
    },
    {
      "ParameterValue": "MyKeyName",
      "ParameterKey": "KeyPairName"
    },
    {
      "ParameterValue": "t2.micro",
      "ParameterKey": "InstanceType"
    }
  ],
  "Changes": [
    {
      "ResourceChange": {
        "ResourceType": "AWS::EC2::Instance",
        "PhysicalResourceId": "i-1abc23d4",
        "Details": [
          {
            "ChangeSource": "DirectModification",
            "Evaluation": "Static",
            "Target": {
              "Attribute": "Tags",
              "RequiresRecreation": "Never"
            }
          }
        ],
        "Action": "Modify",
        "Scope": [
          "Tags"
        ],
        "LogicalResourceId": "MyEC2Instance",
        "Replacement": "False"
      },
      "Type": "Resource"
    }
  ]
}
```

```
    ],  
    "CreationTime": "2020-11-18T23:35:25.813Z",  
    "Capabilities": [],  
    "StackName": "MyStack",  
    "NotificationARNs": [],  
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/  
SampleChangeSet-direct/1a2345b6-0000-00a0-a123-00abc0abc000"  
}
```

Changes 構造には、ResourceChange 構造 1 つのみが存在します。この構造は、CloudFormation が変更するリソースのタイプ、CloudFormation が実行するアクション、リソースの ID、変更対象、および変更で置き換え (CloudFormation が新しいリソースを作成して古いものを削除する) が必要かどうかなどの情報を記述します。例では、CloudFormation によって i-1abc23d4 EC2 インスタンスの Tags 属性が変更され、インスタンスの置き換えは必要ないことが変更セットによって示されています。

Details 構造で、この変更がインスタンスの再作成 (置き換え) が不要な直接変更であると CloudFormation で分類されます。CloudFormation がインスタンスを置き換えないことを確認し、安心してこの変更を実行できます。

CloudFormation は、この変更を Static 評価として表示します。静的評価とは、変更セットを実行する前に CloudFormation によってタグの値を決定できることを意味します。場合によっては、変更セットを実行した後にしか CloudFormation が値を決定できないこともあります。CloudFormation は、これらの変更を Dynamic 評価としてラベル付けします。例えば、更新されたリソースの参照でそのリソースが条件付きで置き換えられる場合、CloudFormation では更新されたリソースへの参照が変更されるかどうかは決定できません。

入力パラメータ値の修正

入力パラメータ値を変更すると、CloudFormation では更新されたパラメータ値を使用する各リソースで 2 つの変更が生成されます。この例では、これらの変更がどのようなもので、どの情報に注目すべきかを取り上げます。次の例は、Purpose 入力パラメータの値の変更のみによって生成されています。

Purpose パラメータは、EC2 インスタンスのタグキーの値を指定します。例では、パラメータ値が testing から production に変更されています。新しい値は Parameters 構造に表示されます。

```
{  
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",  
    "Status": "CREATE_COMPLETE",
```

```
"ChangeSetName": "SampleChangeSet",
"Parameters": [
  {
    "ParameterValue": "production",
    "ParameterKey": "Purpose"
  },
  {
    "ParameterValue": "MyKeyName",
    "ParameterKey": "KeyPairName"
  },
  {
    "ParameterValue": "t2.micro",
    "ParameterKey": "InstanceType"
  }
],
"Changes": [
  {
    "ResourceChange": {
      "ResourceType": "AWS::EC2::Instance",
      "PhysicalResourceId": "i-1abc23d4",
      "Details": [
        {
          "ChangeSource": "DirectModification",
          "Evaluation": "Dynamic",
          "Target": {
            "Attribute": "Tags",
            "RequiresRecreation": "Never"
          }
        },
        {
          "CausingEntity": "Purpose",
          "ChangeSource": "ParameterReference",
          "Evaluation": "Static",
          "Target": {
            "Attribute": "Tags",
            "RequiresRecreation": "Never"
          }
        }
      ],
      "Action": "Modify",
      "Scope": [
        "Tags"
      ],
      "LogicalResourceId": "MyEC2Instance",
```

```
        "Replacement": "False"
      },
      "Type": "Resource"
    }
  ],
  "CreationTime": "2020-11-18T23:59:18.447Z",
  "Capabilities": [],
  "StackName": "MyStack",
  "NotificationARNs": [],
  "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

Changes 構造は、「[テンプレートの直接編集](#)」の例に似た方法で機能します。ResourceChange 構造 1 つのみがあり、Tags EC2 インスタンスの i-1abc23d4 属性への変更を記述します。

ただし、Details 構造では、変更されたパラメータ値が単一でも、変更セットには Tags 属性に対する 2 つの変更が表示されます。変更されたパラメータ値を参照するリソース (Ref 組み込み関数を使用) は、常に 2 つの変更になります。ひとつは Dynamic 評価、もうひとつは Static 評価です。以下のフィールドを参照して、このようなタイプの変更を確認できます。

- Static 評価の変更の場合、ChangeSource フィールドを参照します。この例では、ChangeSource フィールドは ParameterReference と等しく、この変更が更新されたパラメータの参照値の結果であることを意味します。変更セットには、似たような Dynamic 評価の変更があるはずです。
- 同じ情報を含む 2 つの変更の Dynamic 構造を比較することで、一致する Target 評価の変更を検索できます。この例では、両方の変更の Target 構造で、Attribute および RequireRecreation フィールドに同じ値が含まれます。

このようなタイプの変更では、静的評価に注目します。ここには、変更に関するもっとも詳細な情報が提供されています。この例では、変更がパラメータ参照値 (ParameterReference) の変更の結果であることが静的評価に示されています。変更された正確なパラメータは CauseEntity フィールド (Purpose パラメータ) で示されます。

Replacement フィールドの値の決定

ResourceChange 構造の Replacement フィールドには、CloudFormation でリソースが再作成されるかどうかを示されています。リソースの再作成 (置き換え) に備えておくと、バックアップされていないデータの損失やスタックで実行中のアプリケーションの中断を防ぐことができます。

Replacement フィールドの値は、変更で置き換えが必要かどうかによって異なります。これは変更の RequiresRecreation 構造の Target フィールドに示されています。たとえば、RequiresRecreation フィールドが Never であれば、Replacement フィールドは False です。ただし、1 つのリソースに複数の変更があり、各変更の RequiresRecreation フィールドの値が異なる場合、CloudFormation では、もっとも大きい動作を使用してリソースを更新します。つまり、多くの変更の中の 1 つのみで置き換えが必要な場合、CloudFormation はリソースを置き換えるため、Replacement フィールドは True に設定されます。

次の変更セットは各パラメータ (Purpose、InstanceType、KeyPairName) の値を変更することで生成されています。これらすべてのパラメータは EC2 インスタンスで使用されます。これらの変更に伴い、Replacement フィールドが True に等しくなるため、CloudFormation によるインスタンスの置き換えが必要になります。

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
  "Status": "CREATE_COMPLETE",
  "ChangeSetName": "SampleChangeSet-multiple",
  "Parameters": [
    {
      "ParameterValue": "production",
      "ParameterKey": "Purpose"
    },
    {
      "ParameterValue": "MyNewKeyName",
      "ParameterKey": "KeyPairName"
    },
    {
      "ParameterValue": "t2.small",
      "ParameterKey": "InstanceType"
    }
  ],
  "Changes": [
    {
      "ResourceChange": {
        "ResourceType": "AWS::EC2::Instance",
        "PhysicalResourceId": "i-7bef86f8",
        "Details": [
          {
            "ChangeSource": "DirectModification",
            "Evaluation": "Dynamic",
            "Target": {
```



```
        "Attribute": "Properties",
        "Name": "KeyName",
        "RequiresRecreation": "Always"
    },
    {
        "ChangeSource": "DirectModification",
        "Evaluation": "Dynamic",
        "Target": {
            "Attribute": "Properties",
            "Name": "InstanceType",
            "RequiresRecreation": "Conditionally"
        }
    },
    {
        "ChangeSource": "DirectModification",
        "Evaluation": "Dynamic",
        "Target": {
            "Attribute": "Tags",
            "RequiresRecreation": "Never"
        }
    },
    {
        "CausingEntity": "KeyPairName",
        "ChangeSource": "ParameterReference",
        "Evaluation": "Static",
        "Target": {
            "Attribute": "Properties",
            "Name": "KeyName",
            "RequiresRecreation": "Always"
        }
    },
    {
        "CausingEntity": "InstanceType",
        "ChangeSource": "ParameterReference",
        "Evaluation": "Static",
        "Target": {
            "Attribute": "Properties",
            "Name": "InstanceType",
            "RequiresRecreation": "Conditionally"
        }
    },
    {
        "CausingEntity": "Purpose",
```

```

        "ChangeSource": "ParameterReference",
        "Evaluation": "Static",
        "Target": {
            "Attribute": "Tags",
            "RequiresRecreation": "Never"
        }
    },
    "Action": "Modify",
    "Scope": [
        "Tags",
        "Properties"
    ],
    "LogicalResourceId": "MyEC2Instance",
    "Replacement": "True"
},
{
    "Type": "Resource"
},
],
"CreationTime": "2020-11-18T00:39:35.974Z",
"Capabilities": [],
"StackName": "MyStack",
"NotificationARNs": [],
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet-multiple/1a2345b6-0000-00a0-a123-00abc0abc000"
}

```

リソースの置換を要する変更を特定するには、各変更 (Details 構造の静的評価) を表示します。この例では、各変更の `RequireRecreation` フィールドの値は異なりますが、`KeyName` プロパティに対する変更がもっとも大きな更新動作になるため、常に再作成が必要になります。キー名が変更されたため、CloudFormation はインスタンスを置き換えます。

キー名が変わらない場合は、`InstanceType` プロパティに対する変更がもっとも大きな更新動作 (Conditionally) になります。したがって、`Replacement` フィールドは `Conditionally` です。CloudFormation がインスタンスを置き換える条件を把握するには、[AWS::EC2::Instance](#) リソースタイプの `InstanceType` プロパティの更新動作を確認します。

リソースの追加および削除

次の例は、EC2 インスタンスを削除し、Auto Scaling グループと起動構成を追加するように変更したテンプレートを送信することで生成されています。

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
  "Status": "CREATE_COMPLETE",
  "ChangeSetName": "SampleChangeSet-addremove",
  "Parameters": [
    {
      "ParameterValue": "testing",
      "ParameterKey": "Purpose"
    },
    {
      "ParameterValue": "MyKeyName",
      "ParameterKey": "KeyPairName"
    },
    {
      "ParameterValue": "t2.micro",
      "ParameterKey": "InstanceType"
    }
  ],
  "Changes": [
    {
      "ResourceChange": {
        "Action": "Add",
        "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
        "Scope": [],
        "Details": [],
        "LogicalResourceId": "AutoScalingGroup"
      },
      "Type": "Resource"
    },
    {
      "ResourceChange": {
        "Action": "Add",
        "ResourceType": "AWS::AutoScaling::LaunchConfiguration",
        "Scope": [],
        "Details": [],
        "LogicalResourceId": "LaunchConfig"
      },
      "Type": "Resource"
    },
    {
      "ResourceChange": {
        "ResourceType": "AWS::EC2::Instance",
```

```

        "PhysicalResourceId": "i-1abc23d4",
        "Details": [],
        "Action": "Remove",
        "Scope": [],
        "LogicalResourceId": "MyEC2Instance"
    },
    "Type": "Resource"
}
],
"CreationTime": "2020-11-18T01:44:08.444Z",
"Capabilities": [],
"StackName": "MyStack",
"NotificationARNs": [],
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-addremove/1a2345b6-0000-00a0-a123-00abc0abc000"
}

```

Changes 構造には 3 つの ResourceChange 構造があり、各リソースに 1 つずつ対応しています。各リソースで、Action フィールドには CloudFormation によってリソースが追加または削除されるかどうかを示されています。Scope および Details フィールドはリソースの変更にのみ適用されるため、空になります。

新しいリソースの場合、CloudFormation では変更セットが実行されるまで一部のフィールドの値を決定できません。例えば、Auto Scaling グループの物理 ID と起動構成はまだ存在していないため、CloudFormation では入力できません。CloudFormation は、変更セットを実行すると新しいリソースを作成します。

プロパティレベルの変更の表示

以下の例は、Amazon EC2 インスタンスの Tag プロパティに対するプロパティレベルの変更を示しています。Value と Key タグが Test に変更されます。

```

"ChangeSetName": "SampleChangeSet",
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/38d91d27-798d-4736-9bf1-fb7c46207807",
"StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
SampleEc2Template/68edcdc0-f6b6-11ee-966c-126d572cdd11",
"StackName": "SampleEc2Template",
"Description": "A sample EC2 instance template for testing change sets.",
"Parameters": [
    {
        "ParameterKey": "KeyPairName",

```

```

        "ParameterValue": "BatchTest"
    },
    {
        "ParameterKey": "Purpose",
        "ParameterValue": "testing"
    },
    {
        "ParameterKey": "InstanceType",
        "ParameterValue": "t2.micro"
    }
],
"CreationTime": "2024-04-09T21:29:10.759000+00:00",
"ExecutionStatus": "AVAILABLE",
"Status": "CREATE_COMPLETE",
"StatusReason": null,
"NotificationARNs": [],
"RollbackConfiguration": {
...skipping...
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "MyEC2Instance",
                "PhysicalResourceId": "i-0cc7856a36315e62b",
                "ResourceType": "AWS::EC2::Instance",
                "Replacement": "False",
                "Scope": [
                    "Tags"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0/Value",
                            "BeforeValue": "testing",
                            "AfterValue": "Test",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ]
            }
        }
    ]
}

```

```

        {
            "Target": {
                "Attribute": "Tags",
                "RequiresRecreation": "Never",
                "Path": "/Properties/Tags/0/Key",
                "BeforeValue": "Purpose",
                "AfterValue": "Test",
                "AttributeChangeType": "Modify"
            },
            "Evaluation": "Static",
            "ChangeSource": "DirectModification"
        }
    ],
    "BeforeContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"testing\",\"Key\":\"Purpose\"}]}}",
    "AfterContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"Test\",\"Key\":\"Test\"}]}}"
}
    ]
}

```

Details 構造は、変更セットが実行される前の Key と Value の値と、変更セットが実行された後の値を示しています。

ネストされたスタックの変更セット

ネストされたスタックの変更セットを使用すると、ネストされたスタック階層全体でアプリケーションとインフラストラクチャリソースに対する変更をプレビューし、すべての変更が意図したとおりに行われたことを確認してから、更新を続行できます。

ネストされたスタックの変更セットの詳細については、次のセクションを参照してください:

トピック

- [変更セットとネストされたスタックの概要](#)
- [ネストされたスタックの変更セットの操作 \(コンソール\)](#)
- [ネストされたスタックの変更セットの操作 \(AWS CLI\)](#)

変更セットとネストされたスタックの概要

ネストされたスタックの変更セットは、次の機能を組み合わせて、スタック階層全体に対する変更をプレビューする範囲を拡大します。

- 変更セットは、スタックへの提案された変更が既存または新しく作成されたリソースにどのように影響するかをプレビューする CloudFormation の機能です。変更セットを作成すると、提出したリソースに対する変更とスタックを比較して、CloudFormation は提案された変更のリストを提供します。変更セットに関する詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。
- ネストされたスタックは、別のスタックの一部として作成されます。たとえば、あるネストされたスタックにネットワークおよびセキュリティ関連のリソースがあり、別のスタックにアプリケーションリソースがあるとします。この方法でアプリケーションモデルをパーティション化すると、コードの保守性と再利用に役立ちます。ネストされたスタックの詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。

ネストされたスタックの変更セットの操作 (コンソール)

- 変更セットを作成する – スタック階層の任意のレベルから変更を送信して、変更セットを作成します。変更したスタックテンプレートまたは変更した入力パラメータ値を送信すると、CloudFormation はネストされたスタックを変更セットを生成するために送信した変更と比較します。ネストされたスタックの変更セットは、デフォルトで CloudFormation コンソールで有効になります。詳細については、「[CloudFormation スタックの変更セットを作成する](#)」を参照してください。

▼ Change sets

Change sets for nested stacks

Creates change sets for all nested stacks specified in your template. [Learn more](#)

☒ Enabled

Creates a change set for all nested stacks specified in the template.

☐ Disabled

Create a change set only on the target stack.

Note

ルート変更セットは、変更セットの階層全体が作成されるスタックに関連付けられた変更セットです。ネストされたスタックの変更セットを、ルート変更セットから実行または削

除する必要があります。詳細については、「[ネストされたスタックでのスタックオペレーションの実行](#)」を参照してください。

- 変更セットを表示– ネストされたスタック内のリソースに対する変更を、実行する前に可視化します。現在のスタックとそのネストされた変更セットを移動すると、変更セットの変更セクションに、提案された変更を表示できます。詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。
- 変更セットの実行– 現在のスタックとその子孫に関する変更セットに記載されている変更を実行します。実行オペレーションはルート変更セットから実行する必要があります。詳細については、「[CloudFormation スタックの変更セットを実行する](#)」を参照してください。
- 変更セットを削除– 現在のスタックから変更セットを削除します。変更セットを削除すると、自分または他のユーザーが誤って適用すべきでない変更セットを開始するのを防ぐことができます。削除オペレーションはルート変更セットから実行する必要があります。詳細については、「[CloudFormation スタックの変更セットを削除する](#)」を参照してください。

ネストされたスタックの変更セットの操作 (AWS CLI)

create-change-set

- [create-change-set](#) - ネストされたスタックの変更セットは、デフォルトで AWS CLI で有効になっていません。スタック階層全体の変更セットを作成するには、`--include-nested-stacks` オプションを指定します。詳細については、「[CloudFormation スタックの変更セットを作成する](#)」を参照してください。

次の AWS CLI の例では、指定されたルートスタックの変更セットを作成します。

```
aws cloudformation create-change-set \  
  --stack-name my-root-stack \  
  --change-set-name my-root-stack-change-set \  
  --template-body file://template.yaml \  
  --capabilities CAPABILITY_IAM \  
  --include-nested-stacks
```

以下は出力の例です。

```
{  
  "Id": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
```



```
"StackId": "arn:aws:cloudformation:us-west-2:123456789012:Stack/my-root-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204"
```

describe-change-set

- [describe-change-set](#) – 変更セットを実行した場合に CloudFormation が行う変更のリストを返します。指定された変更セットに、ネストされたスタックに属する子変更セットが含まれている場合、ChangeSetId はその変更セットに関する情報を返します。詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。

次の AWS CLI の例は、指定されたルートスタックの変更セットについて説明します。

```
aws cloudformation describe-change-set \
  --change-set-name my-root-stack-change-set \
  --stack-name my-root-stack
```

以下は出力の例です。

```
{
  "Changes": [
    {
      "Type": "Resource",
      "ResourceChange": {
        "Action": "Modify",
        "LogicalResourceId": "ChildStack",
        "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-nested-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
        "ResourceType": "AWS::CloudFormation::Stack",
        "Replacement": "False",
        "ChangeSetId": "arn:aws:cloudformation:us-
west-2:123456789012:changeSet/my-nested-stack-change-set/4eca1a01-e285-
xmpl-8026-9a1967bfb4b0",
        "Scope": [
          "Properties"
        ],
        "Details": [
          {
            "Target": {
              "Attribute": "Properties",
              "RequiresRecreation": "Never"
            }
          }
        ]
      }
    }
  ]
}
```

```

        "Evaluation": "Dynamic",
        "ChangeSource": "Automatic"
      }
    ]
  }
},
"ChangeSetName": "my-root-stack-change-set",
"ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
"StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-root-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
"StackName": "my-root-stack",
"IncludeNestedStacks": true,
"ParentChangeSetId": null,
"RootChangeSetId": null,
"Description": null,
"Parameters": null,
"CreationTime": "2020-11-18T05:20:56.651Z",
"ExecutionStatus": "AVAILABLE",
"Status": "CREATE_COMPLETE",
"StatusReason": null,
"NotificationARNs": [

],
"RollbackConfiguration": {

},
"Capabilities": [
  "CAPABILITY_IAM"
],
"Tags": null
}

```

次の AWS CLI の例は、指定された、ネストされたスタックの変更セットについて説明します。

```

aws cloudformation describe-change-set \
  --change-set-name my-nested-stack-change-set \
  --stack-name my-nested-stack

```

以下は出力の例です。

```
{
```

```

    "Changes": [
      {
        "Type": "Resource",
        "ResourceChange": {
          "Action": "Modify",
          "LogicalResourceId": "function",
          "PhysicalResourceId": "my-function",
          "ResourceType": "AWS::Lambda::Function",
          "Replacement": "False",
          "ChangeSetId": null,
          "Scope": [
            "Properties"
          ],
          "Details": [
            {
              "Target": {
                "Attribute": "Properties",
                "Name": "Timeout",
                "RequiresRecreation": "Never"
              },
              "Evaluation": "Static",
              "ChangeSource": "DirectModification"
            }
          ]
        }
      }
    ],
    "ChangeSetName": "my-nested-stack-change-set",
    "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-nested-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-nested-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
    "ParentChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "RootChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "IncludeNestedStacks": true,
    "StackName": "my-nested-stack",
    "Description": null,
    "Parameters": null,
    "CreationTime": "2020-11-18T05:20:56.651Z",
    "ExecutionStatus": "UNAVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": "Executable from root change set",

```

```
"NotificationARNs": [  
  
],  
"RollbackConfiguration": {  
  
},  
"Capabilities": [  
    "CAPABILITY_IAM"  
],  
"Tags": null  
}
```

execute-change-set

- [execute-change-set](#) – 指定された変更セットの作成時に提供された入力情報を使い、スタックを作成または更新します。スタック階層全体に対して変更セットを作成するには、create-change-set オペレーション中に `--include-nested-stacks` オプションを指定する必要があります。詳細については、「[CloudFormation スタックの変更セットを実行する](#)」を参照してください。

Note

execute-change-set はルート変更セットから実行する必要があり、スタックの階層全体に変更セットを適用します。

次の AWS CLI の例では、指定されたルートスタックの変更セットを実行します。

```
aws cloudformation execute-change-set \  
    --stack-name my-root-stack \  
    --change-set-name my-root-stack-change-set
```

delete-change-set

- [delete-change-set](#) – 指定された変更セットを削除します。変更セットを削除することで、誤った変更セットの実行を防止できます。`--include-nested-stacks` オプションを使用して作成された変更セットでは、変更セットの削除は非同期になります。詳細については、「[CloudFormation スタックの変更セットを削除する](#)」を参照してください。

 Note

delete-change-set はルート変更セットから実行する必要があり、変更セットの階層全体を削除します。REVIEW_IN_PROGRESS ステータスのネストされたスタックは、create-change-set オペレーション中に作成された場合にも削除されます。

次の AWS CLI の例では、指定されたルートスタックの変更セットを削除します。

```
aws cloudformation delete-change-set \  
  --stack-name my-root-stack \  
  --change-set-name my-root-stack-change-set
```

スタックデプロイを検証する

デプロイ前検証を使用すると、CloudFormation 変更セットを実行する前に、潜在的なデプロイの問題を特定して解決することができます。この機能は、一般的な障害シナリオに対してテンプレートを検証するので、開発サイクルの早い段階で問題を検出するのに役立ちます。

トピック

- [デプロイ前検証の仕組み](#)
- [考慮事項](#)
- [前提条件](#)
- [スタックデプロイを検証する \(コンソール\)](#)
- [スタックデプロイを検証する \(AWS CLI\)](#)
- [検証タイプ](#)
- [リソースの制限](#)

デプロイ前検証の仕組み

デプロイ前検証には次のフェーズが含まれます。

1. 変更セットを作成する – CloudFormation スタックの更新で通常どおり変更セットを生成します。デプロイ前検証は、変更セットを作成したときにデフォルトで有効になっています。

2. [検証の実行] – CloudFormation は、テンプレートとターゲット環境に対して複数の検証チェックを実行します。現在、3 種類の検証がサポートされています: リソーススキーマに対するプロパティ構文の検証、既存のリソースとのリソース名の競合検出、および削除オペレーションの S3 バケット空検証。
3. 検証結果を確認する – CloudFormation は、テンプレート内の問題の場所を正確に示すなど、手動テンプレートデバッグを排除して、見つかった問題に関する詳細なフィードバックを提供します。
4. [問題を解決する] – テンプレートを更新するか、デプロイに進む前に競合を解決することで、特定された問題に対処します。
5. [自信を持って実行する] – 一般的な障害シナリオが事前に検証されていることを認識して、変更セットをデプロイします。

考慮事項

デプロイ前検証を使用する際は、次の点に注意してください。

- デプロイ前検証は、3 つの一般的なデプロイ障害シナリオに焦点を当てています。それによってデプロイが成功することは保証されませんが、一般的な障害の可能性は低下します。
- 検証モードの動作は異なります:
 - FAIL モードは、検証がエラーを検出したときに変更セットの実行を防ぎ、問題のあるテンプレートがデプロイに進むことができないようにします。これは、プロパティ構文エラーとリソース命名競合に適用されます。
 - WARN モードでは、検証障害にもかかわらず変更セットの作成が成功でき、開発者が実行前に確認して対処できる警告が提供されます。これは、手動介入によって解決できる場合がある S3 バケットの空などの制約違反に適用されます。
- 検証結果は、特定の変更セットに関連付けられます。テンプレートを変更する場合は、新しい変更セットを作成して、更新された検証結果を取得する必要があります。
- S3 バケット検証は、削除を防ぐ可能性のあるバケットポリシーやその他の制約ではなく、オブジェクトの存在のみをチェックします。

前提条件

デプロイ前検証を使用するには、以下が必要です:

- アカウントで変更セットを作成し、リソースを読み取るために必要な IAM アクセス許可。S3 バケットの空チェックでは、s3:ListBucket アクセス許可が必要です。
- スタックがデプロイされている AWS リージョン へのアクセス。
- デプロイ前に検証する CloudFormation テンプレート。

スタックデプロイを検証する (コンソール)

コンソールを使用してスタックデプロイを検証するには、次の手順に従います。

デプロイ前にテンプレートを検証するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックが所在する AWS リージョン を選択します。
3. [Stacks] ページで、変更セットを作成する実行中のスタックを選択します。
4. スタックの詳細ペインで、[スタックを更新する]、[変更セットを作成する] を順に選択します。
5. [**stack-name** の変更セットを作成する] ページで、更新されたテンプレートをアップロードするか、テンプレートソースを指定します。
6. [次へ] を選択して、残りの変更セット設定ステップに進みます。
7. テンプレートに IAM リソースが含まれる場合、[機能] で [I acknowledge that CloudFormation might create IAM resources (CloudFormation が IAM リソースを作成する可能性があることを確認します)] を選択します。IAM リソースは、AWS アカウントのアクセス許可を変更できます。これらのリソースを確認し、意図したアクションのみを許可していることを確認してください。詳細については、「[CloudFormation テンプレートの IAM リソースの承認](#)」を参照してください。
8. [確認] ページで、[変更セットを作成する] をクリックします。
9. CloudFormation は、変更セットを作成し、検証チェックを実行します。[デプロイ検証] タブで検証結果を確認します。
10. 検証に合格した場合、または警告に満足した場合は、[変更セットを実行する] を選択して変更をデプロイします。
11. 検証に失敗した場合は、問題を修正し、新しい変更セットを作成して、デプロイを再検証します。

スタックデプロイを検証する (AWS CLI)

デプロイ前検証の AWS CLI コマンドには、以下が含まれます。

- 変更セットの作成中に自動的に検証を行う [create-change-set](#)。
- 変更セットのステータスを検証するための [describe-change-set](#)
- 検証結果を確認するための [describe-events](#)。

AWS CLI を使用してスタックデプロイを検証するには、次の手順に従います。

デプロイ前にテンプレートを検証するには

1. [create-change-set](#) コマンドを使用します。

```
aws cloudformation create-change-set \  
  --stack-name MyStack \  
  --change-set-name MyChangeSet \  
  --change-set-type "CREATE" \  
  --template-body file://updated-template.yaml
```

コマンドは変更セット ARN とスタック ARN の両方を返します。

2. [describe-events](#) コマンドを変更セット ARN または変更セット名のいずれかとともに使用して、検証のステータスと結果を確認します。

```
aws cloudformation describe-events \  
  --change-set-name "arn:aws:cloudformation:us-east-1:123456789012:changeSet/MyChangeSet/94498df5-1afb-43b1-9869-9f82b2d877ac"
```

検証エラーの出力例:

```
{  
  "OperationEvents": [  
    {  
      "EventId": "9b5c9a29-4704-4ad0-8082-afb49418d55b",  
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/c3908380-b357-11f0-a97f-0ad08f35df65",  
      "OperationId": "f558b823-e1e3-4de3-a222-e6b930ddcad4",  
      "OperationType": "CREATE_CHANGESET",  
      "OperationStatus": "FAILED",
```



```

    "EventType": "STACK_EVENT",
    "Timestamp": "2025-10-27T17:10:02.923Z",
    "StartTime": "2025-10-27T17:09:57.537Z",
    "EndTime": "2025-10-27T17:10:02.923Z"
  },
  {
    "EventId": "2d8c3262-3468-4283-82fb-6e780e9e4f1d",
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/c3908380-b357-11f0-a97f-0ad08f35df65",
    "OperationId": "f558b823-e1e3-4de3-a222-e6b930ddcad4",
    "OperationType": "CREATE_CHANGESET",
    "EventType": "VALIDATION_ERROR",
    "LogicalResourceId": "NotificationBucket",
    "PhysicalResourceId": "",
    "ResourceType": "AWS::S3::Bucket",
    "Timestamp": "2025-10-27T17:10:02.461Z",
    "ValidationFailureMode": "FAIL",
    "ValidationName": "PROPERTY_VALIDATION",
    "ValidationStatus": "FAILED",
    "ValidationStatusReason": "#/NotificationConfiguration/QueueConfigurations/0: required key [Event] not found",
    "ValidationPath": "/Resources/NotificationBucket/Properties/NotificationConfiguration/QueueConfigurations/0"
  }
]
}

```

3. テンプレートを更新して検証エラーに対処してから、新しい変更セットを作成します。
4. 検証に合格したならば、変更セットを実行します。

```

aws cloudformation execute-change-set \
  --change-set-name MyChangeSet \
  --stack-name MyStack

```

検証タイプ

デプロイ前検証には、次のタイプのチェックが含まれます。

- プロパティ構文の検証 – AWS リソーススキーマに対してリソースプロパティを検証します。必要なプロパティと有効なプロパティ値をチェックし、非推奨またはサポートされていないプロパティの組み合わせを特定します。

- リソース名の競合検出 – 既存の AWS リソースとの命名競合をチェックします。リソース名が AWS 命名要件を満たしていることを検証し、デプロイの試行前に潜在的な競合を特定します。
- S3 バケットの空の検証 – オブジェクトを含む S3 バケットを削除しようとするすると警告します。削除の影響を評価し、一般的な S3 削除障害を防ぐのに役立つオブジェクト数を提供します。

各検証タイプは、特定のエラーメッセージおよび問題を迅速に解決するのに役立つテンプレート内のエラーの場所を提供します。

リソースの制限

次のリソースタイプは、デプロイ前検証ではサポートされていません。

- `AWS::ApiGatewayV2::ApiGatewayManagedOverrides`
- `AWS::ApiGatewayV2::Stage`
- `AWS::AppMesh::GatewayRoute`
- `AWS::AppMesh::Mesh`
- `AWS::AppMesh::Route`
- `AWS::AppMesh::VirtualGateway`
- `AWS::AppMesh::VirtualNode`
- `AWS::AppMesh::VirtualRouter`
- `AWS::AppMesh::VirtualService`
- `AWS::AppStream::Fleet`
- `AWS::AppStream::Stack`
- `AWS::AppStream::StackFleetAssociation`
- `AWS::AppStream::StackUserAssociation`
- `AWS::AppStream::User`
- `AWS::AppSync::ApiCache`
- `AWS::AppSync::ApiKey`
- `AWS::AppSync::GraphQLSchema`
- `AWS::AutoScalingPlans::ScalingPlan`
- `AWS::Budgets::Budget`
- `AWS::CertificateManager::Certificate`

- `AWS::Cloud9::EnvironmentEC2`
- `AWS::CloudFormation::CustomResource`
- `AWS::CloudFormation::Macro`
- `AWS::CloudFormation::WaitCondition`
- `AWS::CloudFormation::WaitConditionHandle`
- `AWS::CloudFront::StreamingDistribution`
- `AWS::CloudWatch::AnomalyDetector`
- `AWS::CloudWatch::InsightRule`
- `AWS::CodeBuild::Project`
- `AWS::CodeBuild::ReportGroup`
- `AWS::CodeBuild::SourceCredential`
- `AWS::CodeCommit::Repository`
- `AWS::CodeDeploy::DeploymentGroup`
- `AWS::CodeStar::GitHubRepository`
- `AWS::Config::ConfigurationRecorder`
- `AWS::Config::DeliveryChannel`
- `AWS::Config::OrganizationConfigRule`
- `AWS::Config::RemediationConfiguration`
- `AWS::DAX::Cluster`
- `AWS::DAX::ParameterGroup`
- `AWS::DAX::SubnetGroup`
- `AWS::DirectoryService::MicrosoftAD`
- `AWS::DLM::LifecyclePolicy`
- `AWS::DMS::Certificate`
- `AWS::DMS::Endpoint`
- `AWS::DMS::EventSubscription`
- `AWS::DMS::ReplicationInstance`
- `AWS::DMS::ReplicationSubnetGroup`
- `AWS::DMS::ReplicationTask`

- `AWS::DocDB::DBCluster`
- `AWS::DocDB::DBClusterParameterGroup`
- `AWS::DocDB::DBInstance`
- `AWS::DocDB::DBSubnetGroup`
- `AWS::DocDB::EventSubscription`
- `AWS::EC2::ClientVpnAuthorizationRule`
- `AWS::EC2::ClientVpnEndpoint`
- `AWS::EC2::ClientVpnRoute`
- `AWS::EC2::ClientVpnTargetNetworkAssociation`
- `AWS::EC2::NetworkInterfacePermission`
- `AWS::ElastiCache::CacheCluster`
- `AWS::ElastiCache::ReplicationGroup`
- `AWS::ElastiCache::SecurityGroup`
- `AWS::ElastiCache::SecurityGroupIngress`
- `AWS::ElasticLoadBalancing::LoadBalancer`
- `AWS::ElasticLoadBalancingV2::ListenerCertificate`
- `AWS::Elasticsearch::Domain`
- `AWS::EMR::Cluster`
- `AWS::EMR::InstanceFleetConfig`
- `AWS::EMR::InstanceGroupConfig`
- `AWS::FSx::FileSystem`
- `AWS::FSx::Snapshot`
- `AWS::FSx::StorageVirtualMachine`
- `AWS::FSx::Volume`
- `AWS::Glue::Classifier`
- `AWS::Glue::Connection`
- `AWS::Glue::CustomEntityType`
- `AWS::Glue::DataCatalogEncryptionSettings`
- `AWS::Glue::DataQualityRuleset`

- `AWS::Glue::DevEndpoint`
- `AWS::Glue::MLTransform`
- `AWS::Glue::Partition`
- `AWS::Glue::SecurityConfiguration`
- `AWS::Glue::Table`
- `AWS::Glue::TableOptimizer`
- `AWS::Glue::Workflow`
- `AWS::Greengrass::ConnectorDefinition`
- `AWS::Greengrass::ConnectorDefinitionVersion`
- `AWS::Greengrass::CoreDefinition`
- `AWS::Greengrass::CoreDefinitionVersion`
- `AWS::Greengrass::DeviceDefinition`
- `AWS::Greengrass::DeviceDefinitionVersion`
- `AWS::Greengrass::FunctionDefinition`
- `AWS::Greengrass::FunctionDefinitionVersion`
- `AWS::Greengrass::Group`
- `AWS::Greengrass::GroupVersion`
- `AWS::Greengrass::LoggerDefinition`
- `AWS::Greengrass::LoggerDefinitionVersion`
- `AWS::Greengrass::ResourceDefinition`
- `AWS::Greengrass::ResourceDefinitionVersion`
- `AWS::Greengrass::SubscriptionDefinition`
- `AWS::Greengrass::SubscriptionDefinitionVersion`
- `AWS::IAM::AccessKey`
- `AWS::IAM::UserToGroupAddition`
- `AWS::IoT::PolicyPrincipalAttachment`
- `AWS::IoT::ThingPrincipalAttachment`
- `AWS::IoTThingsGraph::FlowTemplate`
- `AWS::KinesisAnalytics::Application`

- `AWS::KinesisAnalytics::ApplicationOutput`
- `AWS::KinesisAnalytics::ApplicationReferenceDataSource`
- `AWS::KinesisAnalyticsV2::ApplicationCloudWatchLoggingOption`
- `AWS::KinesisAnalyticsV2::ApplicationOutput`
- `AWS::KinesisAnalyticsV2::ApplicationReferenceDataSource`
- `AWS::LakeFormation::DataLakeSettings`
- `AWS::LakeFormation::Permissions`
- `AWS::LakeFormation::Resource`
- `AWS::ManagedBlockchain::Member`
- `AWS::ManagedBlockchain::Node`
- `AWS::MediaConvert::JobTemplate`
- `AWS::MediaConvert::Preset`
- `AWS::MediaConvert::Queue`
- `AWS::MediaLive::Channel`
- `AWS::MediaLive::Input`
- `AWS::MediaLive::InputSecurityGroup`
- `AWS::MediaStore::Container`
- `AWS::OpsWorks::App`
- `AWS::OpsWorks::ElasticLoadBalancerAttachment`
- `AWS::OpsWorks::Instance`
- `AWS::OpsWorks::Layer`
- `AWS::OpsWorks::Stack`
- `AWS::OpsWorks::UserProfile`
- `AWS::OpsWorks::Volume`
- `AWS::Pinpoint::ADMChannel`
- `AWS::Pinpoint::APNSChannel`
- `AWS::Pinpoint::APNSSandboxChannel`
- `AWS::Pinpoint::APNSVoipChannel`
- `AWS::Pinpoint::APNSVoipSandboxChannel`

- `AWS::Pinpoint::App`
- `AWS::Pinpoint::ApplicationSettings`
- `AWS::Pinpoint::BaiduChannel`
- `AWS::Pinpoint::Campaign`
- `AWS::Pinpoint::EmailChannel`
- `AWS::Pinpoint::EmailTemplate`
- `AWS::Pinpoint::EventStream`
- `AWS::Pinpoint::GCMChannel`
- `AWS::Pinpoint::PushTemplate`
- `AWS::Pinpoint::Segment`
- `AWS::Pinpoint::SMSChannel`
- `AWS::Pinpoint::SmsTemplate`
- `AWS::Pinpoint::VoiceChannel`
- `AWS::PinpointEmail::ConfigurationSet`
- `AWS::PinpointEmail::ConfigurationSetEventDestination`
- `AWS::PinpointEmail::DedicatedIpPool`
- `AWS::PinpointEmail::Identity`
- `AWS::QLDB::Ledger`
- `AWS::RDS::DBSecurityGroup`
- `AWS::RDS::DBSecurityGroupIngress`
- `AWS::Redshift::ClusterSecurityGroup`
- `AWS::Redshift::ClusterSecurityGroupIngress`
- `AWS::Route53::RecordSet`
- `AWS::Route53::RecordSetGroup`
- `AWS::SageMaker::CodeRepository`
- `AWS::SageMaker::EndpointConfig`
- `AWS::SageMaker::Model`
- `AWS::SageMaker::NotebookInstance`
- `AWS::SageMaker::NotebookInstanceLifecycleConfig`

- `AWS::SageMaker::Workteam`
- `AWS::SDB::Domain`
- `AWS::ServiceCatalog::AcceptedPortfolioShare`
- `AWS::ServiceCatalog::LaunchRoleConstraint`
- `AWS::ServiceCatalog::Portfolio`
- `AWS::ServiceCatalog::StackSetConstraint`
- `AWS::ServiceDiscovery::HttpNamespace`
- `AWS::ServiceDiscovery::Instance`
- `AWS::ServiceDiscovery::PrivateDnsNamespace`
- `AWS::ServiceDiscovery::PublicDnsNamespace`
- `AWS::ServiceDiscovery::Service`
- `AWS::SES::ReceiptFilter`
- `AWS::SES::ReceiptRule`
- `AWS::SES::ReceiptRuleSet`
- `AWS::SSM::MaintenanceWindow`
- `AWS::SSM::MaintenanceWindowTarget`
- `AWS::SSM::MaintenanceWindowTask`
- `AWS::WAF::ByteMatchSet`
- `AWS::WAF::IPSet`
- `AWS::WAF::Rule`
- `AWS::WAF::SizeConstraintSet`
- `AWS::WAF::SqlInjectionMatchSet`
- `AWS::WAF::WebACL`
- `AWS::WAF::XssMatchSet`
- `AWS::WAFRegional::ByteMatchSet`
- `AWS::WAFRegional::GeoMatchSet`
- `AWS::WAFRegional::IPSet`
- `AWS::WAFRegional::RateBasedRule`
- `AWS::WAFRegional::RegexPatternSet`

- `AWS::WAFRegional::Rule`
- `AWS::WAFRegional::SizeConstraintSet`
- `AWS::WAFRegional::SqlInjectionMatchSet`
- `AWS::WAFRegional::WebACL`
- `AWS::WAFRegional::WebACLAssociation`
- `AWS::WAFRegional::XssMatchSet`
- `AWS::WorkSpaces::Workspace`
- `AWS::AmazonMQ::ConfigurationAssociation`
- `AWS::ApiGateway::DomainNameAccessAssociation`
- `AWS::AppConfig::ExtensionAssociation`
- `AWS::AppStream::ApplicationEntitlementAssociation`
- `AWS::AppStream::ApplicationFleetAssociation`
- `AWS::AppSync::DomainNameApiAssociation`
- `AWS::AppSync::SourceApiAssociation`
- `AWS::CleanRooms::ConfiguredTableAssociation`
- `AWS::CleanRooms::IdNamespaceAssociation`
- `AWS::CodeGuruReviewer::RepositoryAssociation`
- `AWS::Cognito::IdentityPoolRoleAttachment`
- `AWS::Cognito::UserPoolRiskConfigurationAttachment`
- `AWS::Cognito::UserPoolUICustomizationAttachment`
- `AWS::Cognito::UserPoolUserToGroupAttachment`
- `AWS::Connect::IntegrationAssociation`
- `AWS::Deadline::QueueFleetAssociation`
- `AWS::Deadline::QueueLimitAssociation`
- `AWS::EC2::EIPAssociation`
- `AWS::EC2::EnclaveCertificateIamRoleAssociation`
- `AWS::EC2::GatewayRouteTableAssociation`
- `AWS::EC2::IPAMResourceDiscoveryAssociation`
- `AWS::EC2::IpPoolRouteTableAssociation`

- `AWS::EC2::LocalGatewayRouteTableVPCAssociation`
- `AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation`
- `AWS::EC2::NetworkInterfaceAttachment`
- `AWS::EC2::RouteServerAssociation`
- `AWS::EC2::SecurityGroupVpcAssociation`
- `AWS::EC2::SubnetNetworkAclAssociation`
- `AWS::EC2::SubnetRouteTableAssociation`
- `AWS::EC2::TransitGatewayAttachment`
- `AWS::EC2::TransitGatewayMulticastDomainAssociation`
- `AWS::EC2::TransitGatewayPeeringAttachment`
- `AWS::EC2::TransitGatewayRouteTableAssociation`
- `AWS::EC2::TransitGatewayVpcAttachment`
- `AWS::EC2::VPCDHCPOptionsAssociation`
- `AWS::EC2::VPCGatewayAttachment`
- `AWS::EC2::VolumeAttachment`
- `AWS::ECS::ClusterCapacityProviderAssociations`
- `AWS::EKS::PodIdentityAssociation`
- `AWS::FSx::DataRepositoryAssociation`
- `AWS::FSx::S3AccessPointAttachment`
- `AWS::GlobalAccelerator::CrossAccountAttachment`
- `AWS::LakeFormation::TagAssociation`
- `AWS::NetworkFirewall::VpcEndpointAssociation`
- `AWS::NetworkManager::ConnectAttachment`
- `AWS::NetworkManager::CustomerGatewayAssociation`
- `AWS::NetworkManager::DirectConnectGatewayAttachment`
- `AWS::NetworkManager::LinkAssociation`
- `AWS::NetworkManager::SiteToSiteVpnAttachment`
- `AWS::NetworkManager::TransitGatewayRouteTableAttachment`
- `AWS::NetworkManager::VpcAttachment`

- `AWS::Notifications::ChannelAssociation`
- `AWS::Notifications::ManagedNotificationAccountContactAssociation`
- `AWS::Notifications::ManagedNotificationAdditionalChannelAssociation`
- `AWS::Notifications::OrganizationalUnitAssociation`
- `AWS::ResourceExplorer2::DefaultViewAssociation`
- `AWS::Route53Profiles::ProfileAssociation`
- `AWS::Route53Profiles::ProfileResourceAssociation`
- `AWS::Route53Resolver::FirewallRuleGroupAssociation`
- `AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation`
- `AWS::Route53Resolver::ResolverRuleAssociation`
- `AWS::SSM::Association`
- `AWS::SecretsManager::SecretTargetAttachment`
- `AWS::SecurityHub::PolicyAssociation`
- `AWS::ServiceCatalog::PortfolioPrincipalAssociation`
- `AWS::ServiceCatalog::PortfolioProductAssociation`
- `AWS::ServiceCatalog::ServiceActionAssociation`
- `AWS::ServiceCatalog::TagOptionAssociation`
- `AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation`
- `AWS::ServiceCatalogAppRegistry::ResourceAssociation`
- `AWS::VpcLattice::ServiceNetworkResourceAssociation`
- `AWS::VpcLattice::ServiceNetworkServiceAssociation`
- `AWS::VpcLattice::ServiceNetworkVpcAssociation`
- `AWS::WAFv2::WebACLAssociation`
- `AWS::Wisdom::AssistantAssociation`
- `AWS::WorkspacesInstances::VolumeAssociation`
- `AWS::IAM::Policy`
- `AWS::SNS::TopicPolicy`
- `AWS::SQS::QueuePolicy`
- `AWS::EC2::NetworkAclEntry`

- `AWS::EC2::VPNGatewayRoutePropagation`
- `AWS::CloudFormation::Stack`
- `AWS::CloudWatch::MetricStream`
- `AWS::WorkSpaces::ConnectionAlias`
- `AWS::IoT::ProvisioningTemplate`
- `AWS::MediaPackage::Channel`
- `AWS::CloudFront::OriginRequestPolicy`
- `AWS::Route53Resolver::ResolverQueryLoggingConfig`
- `AWS::NetworkManager::TransitGatewayRegistration`
- `AWS::ImageBuilder::Image`
- `AWS::Config::ConformancePack`
- `AWS::S3::AccessPoint`
- `AWS::CodeStarConnections::Connection`
- `AWS::CloudFront::CachePolicy`
- `AWS::FMS::NotificationChannel`
- `AWS::ImageBuilder::InfrastructureConfiguration`
- `AWS::Detective::Graph`
- `AWS::EC2::CarrierGateway`
- `AWS::CloudWatch::CompositeAlarm`
- `AWS::CodeArtifact::Repository`
- `AWS::GroundStation::DataflowEndpointGroup`
- `AWS::ElasticLoadBalancingV2::Listener`
- `AWS::ImageBuilder::ImageRecipe`
- `AWS::NetworkManager::Device`
- `AWS::Kendra::DataSource`
- `AWS::Timestream::Database`
- `AWS::CodeGuruProfiler::ProfilingGroup`
- `AWS::Lambda::EventSourceMapping`
- `AWS::ECR::Repository`

- `AWS::WAFv2::IPSet`
- `AWS::GameLift::Alias`
- `AWS::IoTSiteWise::Asset`
- `AWS::OpsWorksCM::Server`
- `AWS::IoT::Authorizer`
- `AWS::WAFv2::RuleGroup`
- `AWS::NetworkManager::Site`
- `AWS::ResourceGroups::Group`
- `AWS::MediaPackage::PackagingConfiguration`
- `AWS::ImageBuilder::ImagePipeline`
- `AWS::ECS::TaskDefinition`
- `AWS::Macie::CustomDataIdentifier`
- `AWS::MediaPackage::OriginEndpoint`
- `AWS::Logs::LogGroup`
- `AWS::CodeArtifact::Domain`
- `AWS::Kendra::Faq`
- `AWS::ECS::TaskSet`
- `AWS::WAFv2::RegexPatternSet`
- `AWS::ECS::Cluster`
- `AWS::SSO::Assignment`
- `AWS::GlobalAccelerator::Listener`
- `AWS::ServiceCatalog::CloudFormationProvisionedProduct`
- `AWS::RDS::DBProxy`
- `AWS::EC2::FlowLog`
- `AWS::ImageBuilder::Component`
- `AWS::CloudFront::RealtimeLogConfig`
- `AWS::NetworkManager::GlobalNetwork`
- `AWS::RDS::DBProxyTargetGroup`
- `AWS::WAFv2::WebACL`

- `AWS::IVS::StreamKey`
- `AWS::IVS::PlaybackKeyPair`
- `AWS::Macie::Session`
- `AWS::Route53::HealthCheck`
- `AWS::Synthetics::Canary`
- `AWS::Lambda::CodeSigningConfig`
- `AWS::EFS::AccessPoint`
- `AWS::Timestream::Table`
- `AWS::MediaPackage::PackagingGroup`
- `AWS::ECS::PrimaryTaskSet`
- `AWS::Config::ConfigurationAggregator`
- `AWS::GroundStation::Config`
- `AWS::IoTSiteWise::AssetModel`
- `AWS::SES::ConfigurationSet`
- `AWS::ImageBuilder::DistributionConfiguration`
- `AWS::Config::OrganizationConformancePack`
- `AWS::EC2::LocalGatewayRoute`
- `AWS::KMS::Key`
- `AWS::Detective::MemberInvitation`
- `AWS::EKS::FargateProfile`
- `AWS::MediaPackage::Asset`
- `AWS::GlobalAccelerator::EndpointGroup`
- `AWS::Macie::FindingsFilter`
- `AWS::IoT::Certificate`
- `AWS::SageMaker::MonitoringSchedule`
- `AWS::IVS::Channel`
- `AWS::Kendra::Index`
- `AWS::EventSchemas::RegistryPolicy`
- `AWS::KinesisFirehose::DeliveryStream`

- `AWS::GlobalAccelerator::Accelerator`
- `AWS::EC2::PrefixList`
- `AWS::GameLift::GameServerGroup`
- `AWS::NetworkManager::Link`
- `AWS::EFS::FileSystem`
- `AWS::Route53::HostedZone`
- `AWS::GroundStation::MissionProfile`
- `AWS::KMS::Alias`
- `AWS::FMS::Policy`
- `AWS::SSO::PermissionSet`
- `AWS::StepFunctions::StateMachine`
- `AWS::QLDB::Stream`
- `AWS::IoTSiteWise::Gateway`
- `AWS::ECS::Service`
- `AWS::ECS::CapacityProvider`
- `AWS::EC2::SecurityGroup`
- `AWS::EC2::SecurityGroupIngress`
- `AWS::EC2::SecurityGroupEgress`
- `AWS::EC2::EC2Fleet`
- `AWS::IAM::Group`
- `AWS::IAM::Role`
- `AWS::IAM::User`
- `AWS::ApiGateway::GatewayResponse`
- `AWS::S3::BucketPolicy`
- `AWS::SNS::Topic`
- `AWS::SNS::Subscription`
- `AWS::RDS::DBInstance`
- `AWS::RDS::DBParameterGroup`
- `AWS::RDS::DBCluster`

- `AWS::RDS::DBClusterParameterGroup`
- `AWS::RDS::DBSubnetGroup`
- `AWS::RDS::EventSubscription`
- `AWS::RDS::GlobalCluster`
- `AWS::RDS::OptionGroup`
- `AWS::Neptune::DBInstance`
- `AWS::Neptune::DBParameterGroup`
- `AWS::Neptune::DBCluster`
- `AWS::Neptune::DBClusterParameterGroup`
- `AWS::Neptune::DBSubnetGroup`
- `AWS::Redshift::Cluster`
- `AWS::Redshift::ClusterParameterGroup`
- `AWS::Redshift::ClusterSubnetGroup`
- `AWS::Redshift::EndpointAccess`
- `AWS::Redshift::EndpointAuthorization`
- `AWS::Redshift::EventSubscription`
- `AWS::Redshift::ScheduledAction`
- `AWS::ElastiCache::SubnetGroup`

スタックを直接更新する

更新をスタックにすばやくデプロイするには、直接更新を実行します。直接更新では、テンプレートを送信するか、スタック内のリソースに対して更新を指定する入力パラメータを送信すると、CloudFormation によりすぐにデプロイされます。テンプレートを使用して更新する場合、現在のテンプレートを変更し、ローカルまたは Amazon S3 バケットに保存できます。

更新がサポートされていないリソースプロパティの場合、現在の値を保持する必要があります。更新前に CloudFormation がスタックに加える変更を確認するには、変更セットを使用します。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

スタックを更新する際に、更新するプロパティによっては、CloudFormation によってリソースが中断されたり、更新されたリソースに置き換えられる場合があります。リソースの更新動作の詳細については、[スタックリソースの更新動作を理解する](#)を参照してください。

スタックを更新するには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. [スタック] ページで、更新する実行中のスタックを選択します。
4. スタックの詳細ペインで、[更新] を選択します。
5. スタックテンプレートを変更していない場合は、[現在のテンプレートの使用] を選択し、[次へ] を選択します。

テンプレートを変更した場合は、[既存のテンプレートを置換] を選択し、[テンプレートの指定] セクションで更新されたテンプレートの場所を指定します。

- ローカルコンピュータに保存されているテンプレートの場合は、[テンプレートファイルをアップロード] を選択します。[Choose file] (ファイルの選択) を選択してファイルの場所に移動し、ファイルを選択して、[Next] (次へ) を選択します。

Note

ローカルテンプレートファイルをアップロードする場合、CloudFormation は AWS アカウント の Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。CloudFormation によって作成された S3 バケットがない場合は、テンプレートファイルをアップロードしたリージョンごとに一意のバケットが作成されます。AWS アカウント に、CloudFormation によって作成された S3 バケットが既にある場合、CloudFormation はそのバケットにテンプレートを追加します。

CloudFormation によって作成された S3 バケットについて覚えておくべき考慮事項

- バケットには、AWS アカウントで Amazon S3 許可を持つすべてのユーザーがアクセスできます。
- CloudFormation は、デフォルトで有効になっているサーバー側の暗号化を使用してバケットを作成し、バケットに保存されているすべてのオブジェクトを暗号化します。

CloudFormation が作成したバケットの暗号化オプションを直接管理できます。例えば、<https://console.aws.amazon.com/s3/> の Amazon S3 コンソールまたは AWS CLI を使用できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」で「[Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定](#)」を参照してください。

- 手動でテンプレートを Amazon S3 にアップロードすることで、独自のバケットを使用してアクセス権限を管理できます。スタックを作成または更新する場合は、テンプレートファイルの Amazon S3 の URL を指定します。

- Amazon S3 バケットに保存されているテンプレートの場合は、[Amazon S3 URL] を選択します。テンプレートの URL を入力するか貼り付けて、[次へ] を選択します。

バージョニング対応バケットにテンプレートがある場合は、URL に ?versionId=**version-id** を付加することでテンプレートの具体的なバージョンを指定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バージョニングが有効なバケット内のオブジェクトの使用](#)」を参照してください。

構文の問題が検出された場合、コンソールにはテンプレートの修正に役立つエラーメッセージが表示されます。

6. テンプレートにパラメータが含まれている場合、[Specify stack details] (スタックの詳細を指定) ページで、パラメータ値を入力または変更し、[Next] (次へ) を選択します。

CloudFormation により、各パラメータには、NoEcho 属性で宣言されたパラメータを除いて、スタックに現在設定されている値が入力されますが、[既存の値の使用] をオンにすることで、現在の値をそのまま使用することもできます。

NoEcho を使用して機密情報をマスクする方法、および動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティス」を参照してください。

7. [Configure stack options] (スタックオプションの構成) ページで、スタックに適用されるタグとアクセス権限を更新したり、スタックポリシー、ロールバック構成などの高度なオプションを変更したり、Amazon SNS 通知トピックを更新したりすることができます。これらのパラメータの詳細については「[スタックオプションを設定する](#)」を参照してください。
8. テンプレートに IAM リソースが含まれる場合は、[機能] で [I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
9. [次へ] を選択して続行します。
10. 送信したスタック情報と変更を確認します。

パラメータ値やテンプレートの URL が適切かなど、正しい情報を送信したことを確認します。

[変更セットのプレビュー] セクションで、必要な変更がすべて CloudFormation により加えられることを確認します。例えば、追加、削除、または変更する予定のリソースを、CloudFormation が追加、削除、および変更することを確認できます。CloudFormation は、スタックの変更セットを作成することによって、このプレビューを生成します。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

11. 変更内容が適切であることを確認できたら、[Update stack] (スタックの更新) を選択します。

 Note

この時点で、変更セットを表示するオプションを使用すると、提案された更新をより徹底的に確認することもできます。これを行うには、[Update stack] (スタックを更新) の代わりに [View change set] (変更セットを表示) を選択してください。CloudFormation により、更新に基づいて生成された変更セットが表示されます。スタックの更新を実行する準備ができたら、[Execute] (実行) を選択します。

CloudFormation によりスタックの詳細ページが表示され、[イベント] ペインが選択された状態になっています。これで、スタックのステータスは UPDATE_IN_PROGRESS になりました。CloudFormation によるスタックの更新が正常に完了したら、スタックの状況が UPDATE_COMPLETE に設定されます。

スタックの更新が失敗した場合、CloudFormation によって変更が自動的にロールバックされ、スタックの状況が UPDATE_ROLLBACK_COMPLETE に設定されます。

 Note

更新が UPDATE_IN_PROGRESS 状態のときはキャンセルできます。詳細については、「[スタック更新をキャンセルする](#)」を参照してください。

コマンドラインを使用してスタックを更新するには

以下のコマンドのいずれかを使用できます。

- [update-stack](#) (AWS CLI)
- [Update-CFNStack](#) (AWS Tools for Windows PowerShell)

コマンドを使用してスタックを更新する例については、「[AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例](#)」を参照してください。

スタック更新をキャンセルする

スタックの更新が開始された後、スタックが UPDATE_IN_PROGRESS 状態である間はスタックの更新をキャンセルできます。更新が完了した後にキャンセルすることはできません。ただし、以前の設定を使用してスタックを再度更新することはできます。

スタックの更新をキャンセルした場合、更新を開始する前に存在していたスタック構成まで、スタックがロールバックされます。

トピック

- [スタックの更新をキャンセルするには \(コンソール\)](#)
- [スタック更新をキャンセルするには \(AWS CLI\)](#)

スタックの更新をキャンセルするには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックが所在する AWS リージョン を選択します。
3. [Stacks] ページで、現在更新されているスタックを選択します。ステータスは UPDATE_IN_PROGRESS である必要があります。
4. [Stack actions (スタックアクション)] を選択し、[スタックの更新をキャンセル] を選択します。
5. 更新のキャンセルを続行するには、[Cancel update] (更新のキャンセル) を選択します。それ以外の場合は、[Cancel] (キャンセル) を選択して更新を再開します。

スタックは UPDATE_ROLLBACK_IN_PROGRESS 状態になります。更新のキャンセルが完了すると、スタックは UPDATE_ROLLBACK_COMPLETE に設定されます。

スタック更新をキャンセルするには (AWS CLI)

コマンド [cancel-update-stack](#) を使用して、更新をキャンセルします。詳細については、「[スタックの更新をキャンセルする](#)」を参照してください。

CloudFormation コンソールからスタックを削除する

スタック内のリソースが不要になった場合は、スタック全体を削除できます。

スタックを削除すると、特定のリソースを保持する削除ポリシーを使用しない限り、CloudFormation はそのスタック内のすべてのリソースを削除します。詳細については、「[DeletionPolicy 属性](#)」を参照してください。

スタックセットを削除するには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックが所在する AWS リージョン を選択します。
3. [スタック] ページで、削除するスタックを選択します。スタックは現在実行中である必要があります。
4. [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、[削除] を選択します。

Note

いったんスタックの削除が開始されると、スタックの削除オペレーションは停止できません。スタックは DELETE_IN_PROGRESS 状態になります。

スタックの削除操作が完了すると、スタックの状態が DELETE_COMPLETE になります。デフォルトでは、DELETE_COMPLETE 状態のスタックは CloudFormation コンソールに表示されません。削除したスタックを表示するには、「[CloudFormation コンソールから削除されたスタックを表示する](#)」に説明するようにスタックの表示フィルターを変更する必要があります。

スタックを強制削除するには (コンソール)

スタック内のリソースが削除に失敗したため、スタックの削除が失敗することがあります。例えば、CloudFormation は、別のスタックも依存するリソースの削除に失敗します。削除されなかったリソースは、スタックが正常に削除されるまで保持されます。削除が失敗して DELETE_FAILED 状態が返された場合は、2 つの方法のいずれかを使用して再試行できます。

1. CloudFormation コンソールの [スタック] ページで、強制削除するスタックを選択します。
2. スタックの詳細ペインで、[削除を再試行] を選択します。

3. 以下のオプションから選択します。

- このスタックは削除しますが、リソースは保持します。このオプションでは、元々削除に失敗したが、強制スタック削除中に保持したい特定のリソースを選択できます。
- スタック全体を強制的に削除: このオプションは、削除に失敗したすべてのリソースを保持し、それらのリソースの依存関係を保持します。

4. [削除] を選択すると、選択内容に基づいて強制削除プロセスが開始されます。

保持されているリソースを確認するには (コンソール)

スタックを削除すると、コンソールに保持されていたリソースを表示できます。

1. スタックリストで、[フィルターステータス]、[削除済み] の順に選択します。
2. 削除されたスタックを選択します。
3. [リソース] タブを選択してください。
4. 保持されているすべてのリソースには、[DELETE_SKIPPEDステータス] が表示されます。
5. 確認する保持されたリソースを選択します。

コマンドラインを使用してスタックを削除するには

以下のコマンドのいずれかを使用できます。

- [delete-stack](#) (AWS CLI)
- [Remove-CFNStack](#) (AWS Tools for Windows PowerShell)

コマンドを使用してスタックを削除する例については、「[AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例](#)」を参照してください。

関連リソース

スタック削除エラーのトラブルシューティングに関するヘルプについては、[スタックの削除の失敗](#)に関するトラブルシューティングのトピックを参照してください。

スタックが誤って削除されないように保護する方法については、「[CloudFormation スタックの削除を防止する](#)」を参照してください。

CloudFormation コンソールから削除されたスタックを表示する

デフォルトでは、CloudFormation コンソールでは、ステータスが DELETE_COMPLETE に設定されているスタックは表示されません。削除済みスタックの情報を表示するには、スタックの表示設定を変更する必要があります。

削除済みのスタックを表示するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、削除されたスタックがある AWS リージョン を選択します。
3. [スタック] ページで、[ステータスをフィルタリング] ドロップダウンから [削除済み] を選択します。

CloudFormation は、すべてのスタックを「DELETE_COMPLETE」のステータスで一覧表示します。

関連情報

- [CloudFormation コンソールからスタックを削除する](#)
- [CloudFormation コンソールからスタック情報を表示する](#)

スタックの進行状況を監視する

このセクションでは、現在進行中のスタックデプロイを監視する方法について説明します。CloudFormation は、デプロイイベントの詳細な時系列リストを提供します。これには、デプロイの進捗状況と、デプロイ中に発生した問題が表示されます。

トピック

- [CloudFormation スタックイベントを表示する](#)
- [オペレーション別にスタックイベントを表示する](#)
- [CloudFormation スタックデプロイのタイムラインを表示する](#)
- [CloudFormation スタック作成イベントを理解する](#)
- [スタック更新の進行状況を監視する](#)
- [更新のロールバックを続ける](#)

- [スタックが失敗する原因を判断する](#)
- [リソースのプロビジョニング時における失敗への対応方法を選択する](#)

CloudFormation スタックイベントを表示する

スタックイベントを表示して、スタックとそのリソースの進行状況とステータスをモニタリングできます。スタックイベントは、リソースがいつ作成、更新、削除されるか、およびスタックデプロイが想定どおりに進行しているかどうかを理解するのに役立ちます。

トピック

- [スタックイベントを表示する \(コンソール\)](#)
- [スタックイベントを表示する \(AWS CLI\)](#)
- [スタックステータスコード](#)

スタックイベントを表示する (コンソール)

スタックイベントを表示する

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面の上部のナビゲーションバーで、スタックを作成した AWS リージョンを選択します。
3. CloudFormation コンソールの [Stacks] (スタック) ページで、スタック名を選択します。CloudFormation は選択したスタックのスタック詳細を表示します。
4. [イベント] タブを選択して、スタックのために CloudFormation が生成したスタックイベントを表示します。

CloudFormation は、スタックイベントを 1 分ごとに自動的に更新します。これに加えて、CloudFormation は新しいスタックイベントの発生時に [新しいイベントを利用できます] バッジを表示します。更新アイコンを選択して、これらのイベントをリストにロードします。スタックの作成イベントを表示することで、スタックの作成 (または失敗、スタックをデバッグしている場合) につながる一連のイベントを理解することができます。

スタックの作成中、スタックは [スタック] ページでステータスが CREATE_IN_PROGRESS と表示されます。スタックが正常に作成されると、ステータスが CREATE_COMPLETE に変わります。

詳細については、「[CloudFormation スタック作成イベントを理解する](#)」および「[スタック更新の進行状況を監視する](#)」を参照してください。

スタックイベントを表示する (AWS CLI)

代替手段として、スタックが作成されている間に [describe-stack-events](#) コマンドを使用し、報告されるイベントを表示することができます。

以下の describe-stack-events コマンドは、*my-stack* スタックイベントを記述します。

```
aws cloudformation describe-stack-events --stack-name my-stack
```

以下に、応答の例を示します。

```
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
      "EventId": "7b755820-7edf-11f0-ab15-0673b09f3847",
      "StackName": "my-stack",
      "LogicalResourceId": "my-stack",
      "PhysicalResourceId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2025-08-21T22:37:56.243000+00:00",
      "ResourceStatus": "CREATE_COMPLETE",
      "ClientRequestToken": "token"
    },
    {
      "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
      "EventId": "WebServer-CREATE_COMPLETE-2025-08-21T22:37:54.356Z",
      "StackName": "my-stack",
      "LogicalResourceId": "WebServer",
      "PhysicalResourceId": "i-099df76cb31b866a9",
      "ResourceType": "AWS::EC2::Instance",
      "Timestamp": "2025-08-21T22:37:54.356000+00:00",
      "ResourceStatus": "CREATE_COMPLETE",
      "ResourceProperties": "{ \"UserData\": \"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXZzLWVmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I\", \"ImageId\": \"ami-0bbc328167dee8f3c\", \"InstanceType\": \"t2.micro\", \"SecurityGroupIds\": [\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"], \"Tags\": [{ \"Value\": \"Bootstrap Tutorial Web Server\", \"Key\": \"Name\" } ] }",
      "ClientRequestToken": "token"
    }
  ],
}
```

```

{
  "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
  "EventId": "WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:31.226Z",
  "StackName": "my-stack",
  "LogicalResourceId": "WebServer",
  "PhysicalResourceId": "i-099df76cb31b866a9",
  "ResourceType": "AWS::EC2::Instance",
  "Timestamp": "2025-08-21T22:37:31.226000+00:00",
  "ResourceStatus": "CREATE_IN_PROGRESS",
  "ResourceStatusReason": "Resource creation Initiated",
  "ResourceProperties": "{\"UserData\":
\\\"IyEvYm1uL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYm1uL2Nmbi1pbml0IC12I
\\\",\\\"ImageId\\\":\\\"ami-0bbc328167dee8f3c\\\",\\\"InstanceType\\\":\\\"t2.micro\\\",
\\\"SecurityGroupIds\\\":[\\\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\\\"],\\\"Tags\\\":
[{\\\"Value\\\":\\\"Bootstrap Tutorial Web Server\\\",\\\"Key\\\":\\\"Name\\\"}]}\",
  \"ClientRequestToken\": \"token\"
},
{
  \"StackId\": \"arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325\",
  \"EventId\": \"WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:29.210Z\",
  \"StackName\": \"my-stack\",
  \"LogicalResourceId\": \"WebServer\",
  \"PhysicalResourceId\": \"\",
  \"ResourceType\": \"AWS::EC2::Instance\",
  \"Timestamp\": \"2025-08-21T22:37:29.210000+00:00\",
  \"ResourceStatus\": \"CREATE_IN_PROGRESS\",
  \"ResourceProperties\": \"{\\\"UserData\\\":
\\\"IyEvYm1uL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYm1uL2Nmbi1pbml0IC12I
\\\",\\\"ImageId\\\":\\\"ami-0bbc328167dee8f3c\\\",\\\"InstanceType\\\":\\\"t2.micro\\\",
\\\"SecurityGroupIds\\\":[\\\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\\\"],\\\"Tags\\\":
[{\\\"Value\\\":\\\"Bootstrap Tutorial Web Server\\\",\\\"Key\\\":\\\"Name\\\"}]}\",
  \"ClientRequestToken\": \"token\"
},
{
  \"StackId\": \"arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325\",
  \"EventId\": \"WebServerSecurityGroup-
CREATE_COMPLETE-2025-08-21T22:37:28.803Z\",
  \"StackName\": \"my-stack\",
  \"LogicalResourceId\": \"WebServerSecurityGroup\",
  \"PhysicalResourceId\": \"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\",
  \"ResourceType\": \"AWS::EC2::SecurityGroup\",

```

```

        "Timestamp": "2025-08-21T22:37:28.803000+00:00",
        "ResourceStatus": "CREATE_COMPLETE",
        "ResourceProperties": "{\"GroupDescription\": \"Allow HTTP access from my IP address\", \"SecurityGroupIngress\": [{\"CidrIp\": \"0.0.0.0/0\", \"Description\": \"HTTP\", \"FromPort\": \"80\", \"ToPort\": \"80\", \"IpProtocol\": \"tcp\"}]}",
        "ClientRequestToken": "token"
    },
    {
        "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
        "EventId": "WebServerSecurityGroup-CREATE_IN_PROGRESS-2025-08-21T22:37:22.626Z",
        "StackName": "my-stack",
        "LogicalResourceId": "WebServerSecurityGroup",
        "PhysicalResourceId": "my-stack-WebServerSecurityGroup-n8A43bQT1ty2",
        "ResourceType": "AWS::EC2::SecurityGroup",
        "Timestamp": "2025-08-21T22:37:22.626000+00:00",
        "ResourceStatus": "CREATE_IN_PROGRESS",
        "ResourceStatusReason": "Resource creation Initiated",
        "ResourceProperties": "{\"GroupDescription\": \"Allow HTTP access from my IP address\", \"SecurityGroupIngress\": [{\"CidrIp\": \"0.0.0.0/0\", \"Description\": \"HTTP\", \"FromPort\": \"80\", \"ToPort\": \"80\", \"IpProtocol\": \"tcp\"}]}",
        "ClientRequestToken": "token"
    },
    {
        "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
        "EventId": "WebServerSecurityGroup-CREATE_IN_PROGRESS-2025-08-21T22:37:20.186Z",
        "StackName": "my-stack",
        "LogicalResourceId": "WebServerSecurityGroup",
        "PhysicalResourceId": "",
        "ResourceType": "AWS::EC2::SecurityGroup",
        "Timestamp": "2025-08-21T22:37:20.186000+00:00",
        "ResourceStatus": "CREATE_IN_PROGRESS",
        "ResourceProperties": "{\"GroupDescription\": \"Allow HTTP access from my IP address\", \"SecurityGroupIngress\": [{\"CidrIp\": \"0.0.0.0/0\", \"Description\": \"HTTP\", \"FromPort\": \"80\", \"ToPort\": \"80\", \"IpProtocol\": \"tcp\"}]}",
        "ClientRequestToken": "token"
    },
    {
        "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
        "EventId": "64740fe0-7edf-11f0-8a36-06453a64f325",

```

```
    "StackName": "my-stack",
    "LogicalResourceId": "my-stack",
    "PhysicalResourceId": "arn:aws:cloudformation:aws-
region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
    "ResourceType": "AWS::CloudFormation::Stack",
    "Timestamp": "2025-08-21T22:37:17.819000+00:00",
    "ResourceStatus": "CREATE_IN_PROGRESS",
    "ResourceStatusReason": "User Initiated",
    "ClientRequestToken": "token"
  }
]
}
```

最新のイベントが先頭に表示されます。次の表は、describe-stack-events コマンドによって返されるフィールドとその説明です。

フィールド	説明
EventId	イベントの識別子
StackName	イベントに対応するスタックの名前
StackId	イベントに対応するスタックの識別子
LogicalResourceId	リソースの論理識別子
PhysicalResourceId	リソースの物理識別子
ResourceProperties	リソースのプロパティ
ResourceType	リソースのタイプ。
Timestamp	イベントが発生した時刻
ResourceStatus	リソースのステータスは、以下のステータスコードのいずれかになります。CREATE_COMPLETE CREATE_FAILED CREATE_IN_PROGRESS DELETE_COMPLETE DELETE_FAILED DELETE_IN_PROGRESS DELETE_SKIPPED IMPORT_COMPLETE IMPORT_IN_PROGRESS IMPORT_ROLLBACK_COMPLETE IMPORT_ROLLBACK_FAILED IMPORT_ROLLBACK_IN_PROGRESS REVIEW_IN

フィールド	説明
	<code>_PROGRESS</code> <code>ROLLBACK_COMPLETE</code> <code>ROLLBACK_FAILED</code> <code>ROLLBACK_IN_PROGRESS</code> <code>UPDATE_COMPLETE</code> <code>UPDATE_COMPLETE_CLEANUP_IN_PROGRESS</code> <code>UPDATE_FAILED</code> <code>UPDATE_IN_PROGRESS</code> <code>UPDATE_ROLLBACK_COMPLETE</code> <code>UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS</code> <code>UPDATE_ROLLBACK_FAILED</code> <code>UPDATE_ROLLBACK_IN_PROGRESS</code> <code>DELETE_SKIPPED</code> ステータスは、削除ポリシー属性が保持になっているリソースに適用されます。
DetailedStatus	スタックの詳細ステータス。 <code>CONFIGURATION_COMPLETE</code> が存在する場合、スタックリソースの設定フェーズは完了済みで、リソースの安定化が進行中です。
ResourceStatusReason	ステータスの詳細

スタックステータスコード

次の表は、スタックステータスコードの説明です。

スタックステータスと、オプションの詳細ステータス	説明
CREATE_COMPLETE	1 つ以上のスタックを正常に作成。
CREATE_IN_PROGRESS	1 つ以上のスタックを作成中。
CREATE_FAILED	1 つ以上のスタックの作成に失敗。関連するエラーメッセージを確認するには、スタックのイベントを調べます。作成の失敗の理由としては、スタックのすべてのリソースを操作するためのアクセス許可の不足、AWS サービスによるパラメーター値の拒否、リソース作成中のタイムアウトなどが考えられます。

スタックステータスと、オプションの詳細ステータス	説明
DELETE_COMPLETE	1 つ以上のスタックを正常に削除。削除されたスタックは、90 日間は保持され、表示できます。
DELETE_FAILED	1 つ以上のスタックの削除に失敗。削除が失敗したので、まだ実行されているリソースがいくつかある可能性があります。ただし、スタックを操作したり、更新したりすることはできません。スタックを再度削除するか、スタックのイベントを調べて関連するエラーメッセージを確認します。
DELETE_IN_PROGRESS	1 つ以上のスタックを削除中。
REVIEW_IN_PROGRESS	進行中の、予測される StackId を持つがテンプレートまたはリソースのない 1 つ以上のスタックの作成。
 Important このステータスコードのスタックは、スタックの可能な限りの最大数に対してカウントされます。	
ROLLBACK_COMPLETE	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックを正常に削除。スタックは以前の動作状態に戻ります。スタック作成オペレーションの実行中に作成されたすべてのリソースが削除されます。 このステータスが存在するのは、スタックの作成が失敗した後のみです。部分的に作成されたスタックからすべてのオペレーションが適切にクリーンアップされたことを示します。この状態では、削除オペレーションのみを実行できます。
ROLLBACK_FAILED	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックの削除に失敗。スタックを削除するか、スタックのイベントを調べて関連するエラーメッセージを確認します。

スタックステータスと、オプションの詳細ステータス	説明
ROLLBACK_IN_PROGRESS	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックを削除中。
UPDATE_COMPLETE	1 つ以上のスタックを正常に更新。
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	スタックの正常な更新後に、1 つ以上のスタックの古いリソースを削除中。リソースの置き換えが必要なスタック更新の場合、CloudFormation はまず新しいリソースを作成し、次に古いリソースを削除して、スタックの中断を減らすことに役立ちます。この状態では、スタックは更新済みで使用可能ですが、CloudFormation はまだ古いリソースを削除中です。
UPDATE_FAILED	1 つ以上のスタックの失敗した更新。関連するエラーメッセージを確認するには、スタックのイベントを調べます。
UPDATE_IN_PROGRESS	1 つ以上のスタックを更新中。
UPDATE_ROLLBACK_COMPLETE	スタックの更新の失敗後、1 つ以上のスタックを前の動作状態に正常に戻しました。
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	スタックの更新の失敗後、1 つ以上のスタックの新しいリソースを削除中。この状態では、スタックが前の動作状態にロールバックされて使用可能ですが、CloudFormation がスタックの更新中に作成した新しいリソースをまだ削除中です。
UPDATE_ROLLBACK_FAILED	スタックの更新の失敗後、1 つ以上のスタックを前の動作ステートに戻すことに失敗。この状態の場合は、スタックを削除するか、 ロールバックを続行する ことができます。スタックが動作状態に戻るには、エラーを修正する必要がある場合があります。または、サポート に連絡してスタックを使用可能な状態に復元することができます。
UPDATE_ROLLBACK_IN_PROGRESS	スタックの更新の失敗後、1 つ以上のスタックを前の動作状態に戻しています。

スタックステータスと、オプションの詳細ステータス	説明
IMPORT_IN_PROGRESS	インポートオペレーションは現在進行中です。
IMPORT_COMPLETE	resource import をサポートするスタック内のすべてのリソースについて、インポート操作が正常に完了しました。
IMPORT_ROLLBACK_IN_PROGRESS	インポートは、以前のテンプレート構成にロールバックされます。
IMPORT_ROLLBACK_FAILED	スタック内の少なくとも 1 つのリソースに対するインポートのロールバックオペレーションが失敗しました。正常にインポートされたリソース CloudFormation の結果が利用可能になります。
IMPORT_ROLLBACK_COMPLETE	インポートは、以前のテンプレート構成に正常にロールバックされました。

オペレーション別にスタックイベントを表示する

オペレーション別にグループ化されたスタックイベントを表示して、スタックに加えられた変更のシーケンスと範囲をより良く理解できます。オペレーションベースのグループ化は、関連するイベントをまとめて追跡するのに役立つため、進行状況のモニタリングや問題発生時のトラブルシューティングが容易になります。

各スタックオペレーション (作成、更新、削除、ロールバック) には、すべての関連するイベントをグループ化する一意のオペレーション ID が割り当てられます。これにより、特定のオペレーションに焦点を当て、障害の根本原因をすばやく特定できます。

トピック

- [前提条件](#)
- [オペレーション別にスタックイベントを表示する \(コンソール\)](#)
- [オペレーション別にスタックイベントを表示する \(AWS CLI\)](#)
- [スタックステータスコード](#)

前提条件

DescribeEvents API を使用するには、DescribeEvents に必要な IAM アクセス許可が必要です。

オペレーション別にスタックイベントを表示する (コンソール)

オペレーション別にグループ化されたスタックイベントを表示するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックを作成した AWS リージョンを選択します。
3. [スタック] ページで、スタック名を選択します。CloudFormation は選択したスタックのスタック詳細を表示します。
4. [イベント] タブを選択して、スタックのために CloudFormation が生成したスタックイベントを表示します。
5. イベントはオペレーション ID によって自動的にグループ化されます。各オペレーションは、オペレーションタイプ、ステータス、およびタイムスタンプを示す展開可能なセクションとして表示されます。
6. オペレーション ID をクリックして、その特定のオペレーションに関連するイベントのみを示す詳細なビューを開きます。
7. オペレーションの詳細ビューで、[障害のみを表示] チェックボックスを選択して、根本原因分析のために失敗したイベントのみを表示します。

CloudFormation は、スタックイベントを 1 分ごとに自動的に更新します。新しいスタックイベントが発生すると、使用可能な新しいイベントバッジが表示されます。更新アイコンを選択して、これらのイベントをリストにロードします。

オペレーションごとにグループ化されたスタックイベントを表示することで、各オペレーションのイベントのシーケンスを理解し、どの特定のオペレーションが問題を引き起こしたかをすばやく特定できます (スタックをデバッグしている場合)。

スタックオペレーションの実行中に、CREATE_IN_PROGRESS、UPDATE_IN_PROGRESS、または DELETE_IN_PROGRESS のステータスで一覧表示されます。オペレーションが正常に完了すると、そのステータスは CREATE_COMPLETE、UPDATE_COMPLETE、または DELETE_COMPLETE に変わります。

詳細については、「[CloudFormation スタック作成イベントを理解する](#)」および「[スタック更新の進行状況を監視する](#)」を参照してください。

オペレーション別にスタックイベントを表示する (AWS CLI)

コマンド `describe-events` をオペレーション ID フィルタリングとともに使用して、特定のオペレーションのイベントを表示できます。

最後のオペレーション ID を取得する

`describe-stacks` API を介して利用可能なスタックの説明に、最近のオペレーション ID とそのタイプを示す `LastOperations` 情報が含まれるようになりました。これにより、イベントログを通じて解析することなく、発生したオペレーションとその現在のステータスをすばやく特定できます。

```
aws cloudformation describe-stacks --stack-name MyStack
```

以下は、失敗した更新オペレーション後の最後のオペレーションがロールバックであったことを示すレスポンスの例です。

```
{
  "Stacks": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/07580010-bb79-11f0-8f6c-0289bb5c804f",
      "StackName": "MyStack",
      "Description": "A simple CloudFormation template to create an S3 bucket.",
      "CreationTime": "2025-11-07T01:28:13.778000+00:00",
      "LastUpdatedTime": "2025-11-07T01:43:39.838000+00:00",
      "RollbackConfiguration": {},
      "StackStatus": "UPDATE_ROLLBACK_COMPLETE",
      "DisableRollback": false,
      "NotificationARNs": [],
      "Tags": [],
      "EnableTerminationProtection": false,
      "DriftInformation": {
        "StackDriftStatus": "NOT_CHECKED"
      },
      "LastOperations": [
        {
          "OperationType": "ROLLBACK",
          "OperationId": "d0f12313-7bdb-414d-a879-828a99b36f29"
```

```
    },  
    {  
      "OperationType": "UPDATE_STACK",  
      "OperationId": "1c211b5a-4538-4dc9-bfed-e07734371e57"  
    }  
  ]  
}  
]  
}
```

オペレーション ID でイベントをフィルタリングする

次の `describe-events` コマンドは、特定のオペレーション ID のイベントについて説明します。

```
aws cloudformation describe-events \  
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57
```

トラブルシューティングで失敗したイベントのみを表示するには、`--filter FailedEvents=true` パラメータを使用します。

```
aws cloudformation describe-events \  
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57 \  
  --filter FailedEvents=true
```

新しいオペレーション ID フィルタリング機能を使用すると、特定のオペレーションとその関連するイベントに焦点を当てることができます。これは特に以下の場合に便利です。

- 特定の障害のトラブルシューティング: 失敗したオペレーションからイベントを分離して、何が問題だったのかを理解します。
- 実行時間が長いオペレーションのモニタリング: 複雑な更新や大規模なスタックデプロイの進行状況を追跡します。
- 変更の監査: 特定の更新オペレーションに関連付けられているすべてのイベントを確認します。
- 根本原因の分析: 障害フィルターを使用して、デプロイの問題の原因をすばやく特定します。

スタックステータスコード

次の表は、オペレーション ID フィルタリングを使用しているときに `describe-events` コマンドによって返されるフィールドを説明しています。

フィールド	説明
EventId	イベントの識別子
OperationId	このイベントを生成したオペレーションの一意の ID。
StackName	イベントに対応するスタックの名前
StackId	イベントに対応するスタックの識別子
LogicalResourceId	リソースの論理識別子
PhysicalResourceId	リソースの物理識別子
ResourceProperties	リソースのプロパティ
ResourceType	リソースのタイプ。
Timestamp	イベントが発生した時刻
ResourceStatus	リソースのステータス (CREATE_COMPLETE 、 UPDATE_FAILED など)。
DetailedStatus	スタックの詳細ステータス。CONFIGURATION_COMPLETE が存在する場合、スタックリソースの設定フェーズは完了済みで、リソースの安定化が進行中です。
ResourceStatusReason	ステータスの詳細

CloudFormation スタックデプロイのタイムラインを表示する

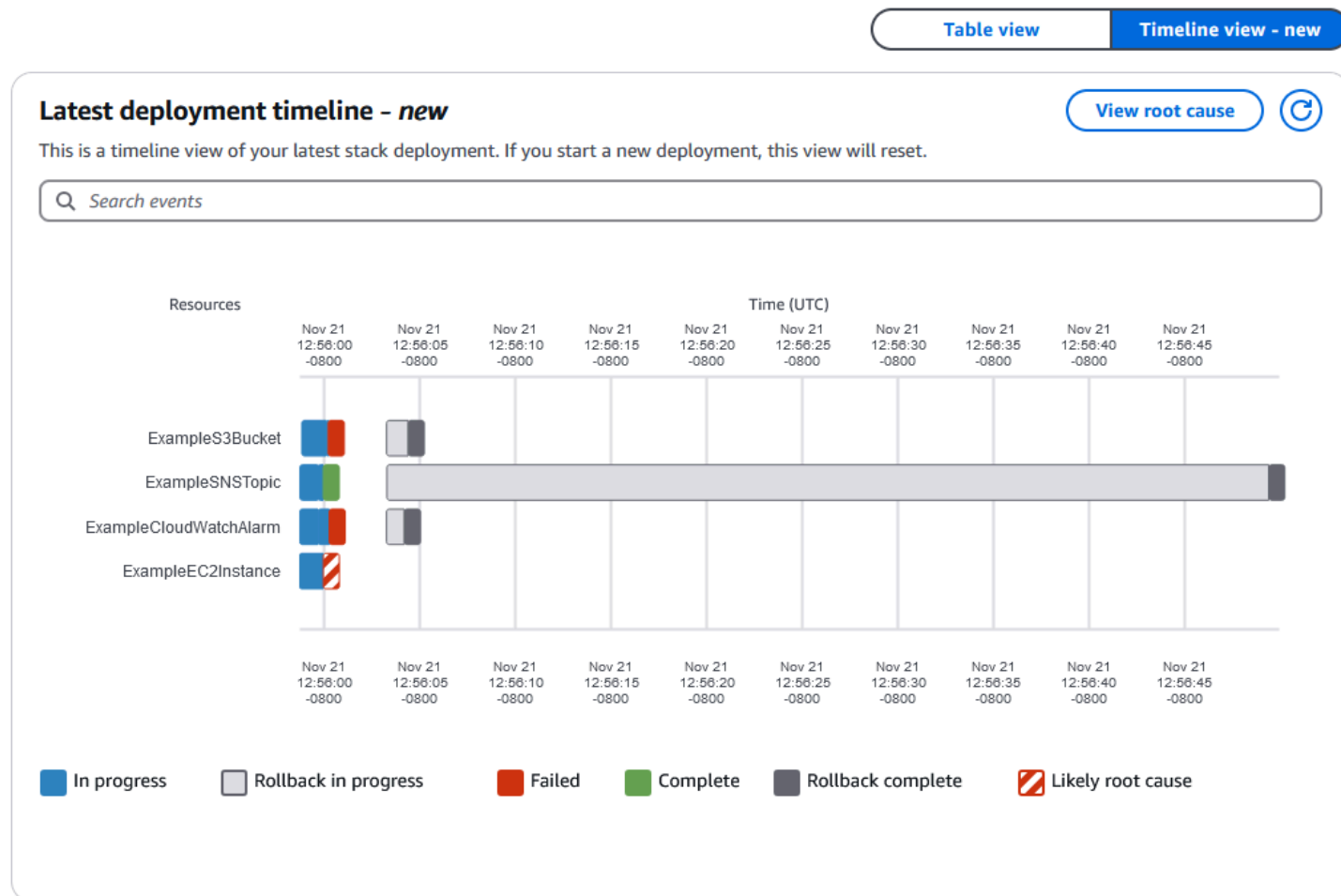
スタックデプロイタイムライングラフは、スタックデプロイタイムラインの視覚的な表現を提供します。このビューでは、スタックとその各リソースのデプロイステータス、および各ステータスが変更された時刻が示されます。スタックのデプロイステータスは、対応する色で表されます。

トピック

- [スタックデプロイタイムライングラフについて](#)
- [スタックデプロイタイムライングラフの表示 \(コンソール\)](#)

スタックデプロイタイムライングラフについて

次の画像は、起動に失敗した Amazon EC2 インスタンスリソースを理由として失敗したスタックデプロイのタイムライングラフを示しています。

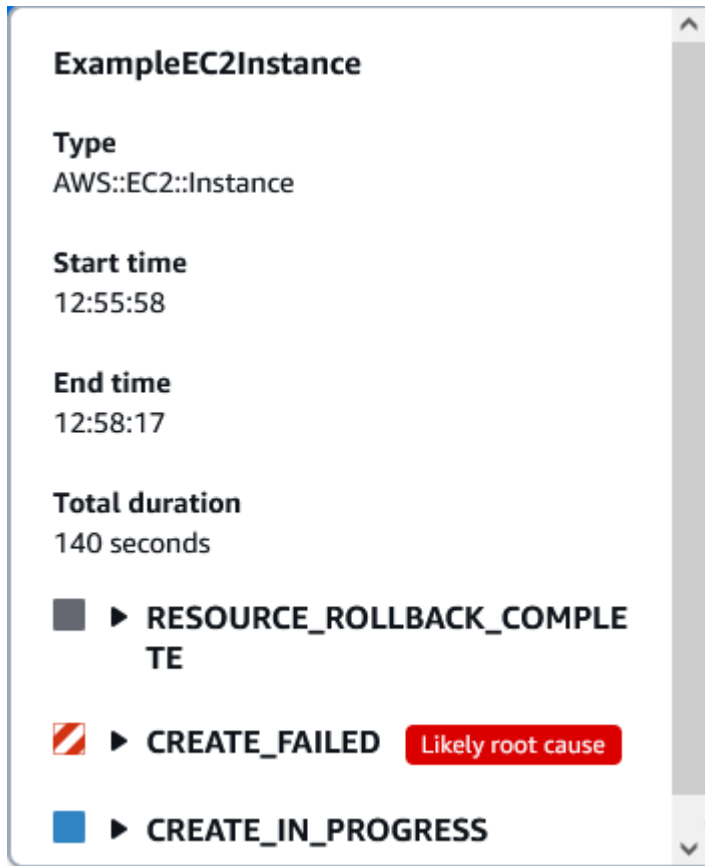


スタックリソースの名前はグラフの左側にあり、デプロイ時刻に関連する日時はグラフの上部にあります。

各リソースは、[進行中] ステータスで始まります。デプロイが成功するたびに、ステータスバーは [完了] に変わります。リソースのデプロイに失敗すると、ステータスバーは [失敗] に変わります。リソースのデプロイに失敗し、スタックのデプロイにも失敗すると、スタックのデプロイの失敗を引き起こしたリソースは、[ルート失敗の可能性] ステータスを受け取ります。

スタックのデプロイオペレーションが失敗すると、正常にデプロイされたリソースはロールバックを開始し、[ロールバックが進行中] ステータスに変わります。リソースのロールバックが完了すると、ステータスは [ロールバックが完了] に変わります。

各リソースを選択すると、デプロイタイムラインのさらにきめ細かい詳細が表示されます:



リソースを選択すると、[タイプ]、デプロイの [開始時刻]、デプロイの [終了時刻]、デプロイの [合計時間] が表示されます。また、以下のドロップダウンメニューには、各デプロイステータスの [開始時刻]、[終了時刻]、[期間] も表示されます。リソースのデプロイに失敗した場合、[失敗の理由] が表示されます。

スタックステータスの詳細については、「[スタックステータスコード](#)」を参照してください。

スタックデプロイタイムライングラフの表示 (コンソール)

スタックデプロイタイムライングラフを表示するには:

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 画面の上部のナビゲーションバーで、スタックを作成した AWS リージョンを選択します。

3. CloudFormation コンソールの [Stacks] (スタック) ページで、スタック名を選択します。CloudFormation は選択したスタックのスタック詳細を表示します。
4. [イベント] タブを選択して、スタックのために CloudFormation が生成したスタックイベントを表示します。
5. [タイムライングラフ] ボタンを選択すると、スタックのタイムライングラフが表示されます。

CloudFormation スタック作成イベントを理解する

スタックのデプロイ中は、スタックテンプレートに定義されているリソースを作成、設定、および検証するためのイベントがいくつか発生します。これらのイベントを理解することは、スタック作成プロセスを最適化し、デプロイを合理化するために役立ちます。

- リソース作成イベント – 各リソースが作成プロセスを開始すると、CREATE_IN_PROGRESS イベントの [ステータス] が設定されます。このイベントは、リソースがプロビジョニングされていることを示します。
- 結果整合性チェック – スタック作成時間の大部分は、スタックによって作成されたリソースに対する結果整合性チェックの実行に費やされます。このフェーズでは、サービスが内部整合性チェックを実行して、リソースが完全に機能し、各 AWS のサービスが定義するサービス安定化基準を満たしていることを確認します。
- 設定完了イベント – 各リソースがプロビジョニングの結果整合性チェックフェーズを完了すると、CONFIGURATION_COMPLETE イベントの [詳細ステータス] が設定されます。
- リソース作成完了イベント – リソースが指定されたとおりに作成および設定され、設定がテンプレートで指定されたものと一致すると、CREATE_COMPLETE イベントの [ステータス] が設定されます。

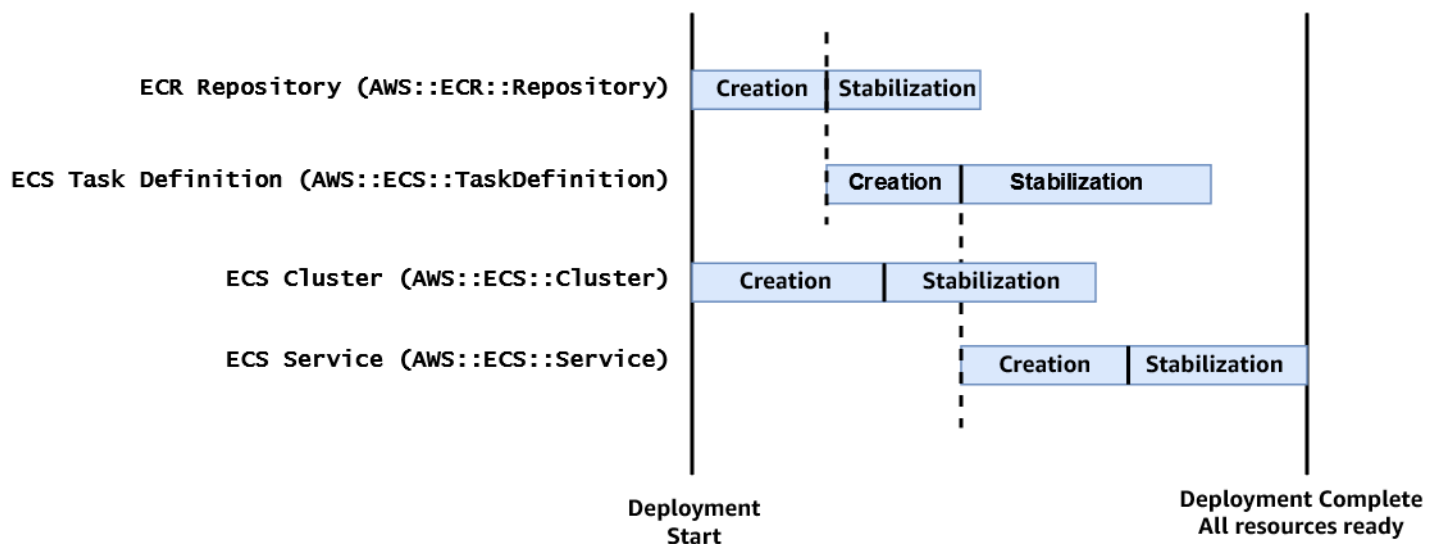
本番稼働前のスタック設定やクロススタックプロビジョニングの検証など、リソースの結果整合性チェックが不要なシナリオでは、CONFIGURATION_COMPLETE イベントを活用してスタック作成プロセスを合理化できます。このイベントは複数の方法で使用できます。例えば、視覚的なシグナルとして使用することで、リソースまたはスタックの整合性チェックの終了を待たずに済みます。または、継続的インテグレーションと継続的デリバリー (CI/CD) を使用して自動メカニズムを作成し、追加のアクションをトリガーすることもできます。

Important

CONFIGURATION_COMPLETE イベントを利用するとスタックの作成時間が短縮されますが、そのトレードオフに注意する必要があります。まず、サポートされているのは、ドリフト

検出をサポートする一部のリソースタイプのみです。ドリフト検出をサポートするリソースタイプのリストについては、「[リソースタイプのサポート](#)」を参照してください。このアプローチは、クラウド環境全体 (本番環境など) の運用準備を完全に確保するために、リソースが徹底的な結果整合性チェックを必要とするシナリオには適さない場合があります。デプロイ要件と各リソースの整合性チェックの重要性を慎重に評価することをお勧めします。CONFIGURATION_COMPLETE イベントを使用して、インフラストラクチャの整合性と信頼性を損なうことなく、デプロイ速度を最適化します。

CONFIGURATION_COMPLETE イベントの設定は保証されないため、CONFIGURATION_COMPLETE イベントが設定済みでない場合は、それを使用するシナリオで CREATE_COMPLETE イベントを処理する準備を整える必要があります。



スタックのデプロイが開始されると、リソース `AWS::ECR::Repository` および `AWS::ECS::Cluster` が作成プロセスを開始します (`ResourceStatus = CREATE_IN_PROGRESS`)。AWS::ECR::Repository リソースタイプによって結果整合性チェックが開始されると (`DetailedStatus = CONFIGURATION_COMPLETE`)、AWS::ECS::TaskDefinition リソースは作成プロセスを開始できるようになります。同様に、AWS::ECS::TaskDefinition リソースが結果整合性チェックを開始すると、AWS::ECS::Service リソースは作成プロセスを開始します。

CREATE_IN_PROGRESS イベントと CREATE_COMPLETE イベント

- [スタック]: CREATE_IN_PROGRESS
- [リソース]: ECR リポジトリ CREATE_IN_PROGRESS

- [リソース]: ECS クラスター CREATE_IN_PROGRESS
- [リソース]: ECR リポジトリ CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [リソース]: ECS タスク定義 CREATE_IN_PROGRESS
- [リソース]: ECS クラスター CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [リソース]: ECS タスク定義 CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [リソース]: ECS サービス CREATE_IN_PROGRESS
- [リソース]: ECR リポジトリ CREATE_COMPLETE
- [リソース]: ECS クラスター CREATE_COMPLETE
- [リソース]: ECS サービス CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [スタック]: CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [リソース]: ECS タスク定義 CREATE_COMPLETE
- [リソース]: ECS サービス CREATE_COMPLETE
- [スタック]: CREATE_COMPLETE

スタック更新の進行状況を監視する

スタックのイベントを表示して、スタック更新の進行状況を監視できます。スタックの [イベント] タブには、スタックの作成と更新における各主要ステップが、各イベントの時間順 (最新イベントが最上位) に表示されます。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

トピック

- [スタックの正常な更新中に生成されたイベント](#)
- [リソース更新が失敗したときに生成されるイベント](#)

スタックの正常な更新中に生成されたイベント

スタックの更新プロセスの開始は、スタックの UPDATE_IN_PROGRESS イベントでマークされます。

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_IN_PROGRESS
```

次のイベントは、更新されたテンプレートで変更された各リソースの更新の開始と完了をマークするイベントです。例えば、MyDB という名前の [AWS::RDS::DBInstance](#) リソースの更新は、次のようなエントリになります。

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

CloudFormation がリソースの更新を開始したことがレポートされると、UPDATE_IN_PROGRESS イベントが記録されます。リソースが正常に作成されると、UPDATE_COMPLETE イベントが記録されます。

CloudFormation がスタックを正常に更新すると、以下のイベントが表示されます。

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_COMPLETE
```

Important

スタックの更新オペレーション中に、CloudFormation が既存のリソースを置き換える必要がある場合、最初に新しいリソースを作成し、次に古いリソースを削除します。ただし、CloudFormation が古いリソースを削除できない場合があります (例えば、ユーザーが特定のタイプのリソースを削除するためのアクセス権限を持っていない場合)。

CloudFormation は、古いリソースの削除を 3 回試みます。CloudFormation が古いリソースを削除できない場合、古いリソースをスタックから削除し、スタックの更新を続行します。スタックの更新が完了すると、CloudFormation は UPDATE_COMPLETE スタックイベントを発行しますが、1 つ以上のリソースを削除できなかったことを示す StatusReason が含まれます。CloudFormation は、特定のリソースに対して DELETE_FAILED イベントを発行し、対応する StatusReason は CloudFormation がリソースの削除に失敗した理由の詳細を提供します。

古いリソースはまだ存在し、引き続き課金されますが、CloudFormation からはアクセスできなくなります。古いリソースを削除するには、基盤となるサービスのコンソールまたは API を使用して、古いリソースに直接アクセスします。

これは、スタックテンプレートから削除したリソースにも当てはまります。そのため、スタックの更新中にスタックから削除されます。

リソース更新が失敗したときに生成されるイベント

リソースの更新が失敗した場合、CloudFormation は失敗の理由を含めた UPDATE_FAILED イベントを報告します。例えば、[AWS::RDS::DBInstance](#) リソースの AllocatedStorage サイズの減少など、リソースでサポートされていないプロパティ変更を更新テンプレートで指定した場合、次のようなイベントが表示されます。

```
2011-09-30 09:36 PDT AWS::RDS::DBInstance MyDB UPDATE_FAILED Size cannot be less than
current size; requested: 5; current: 10
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

リソース更新が失敗した場合、CloudFormation はアップグレード中に更新したすべてのリソースを、更新前の設定にロールバックします。更新のロールバック中に表示されるイベントの例を次に示します。

```
2011-09-30 09:38 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_COMPLETE
2011-09-30 09:38 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:37 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
2011-09-30 09:37 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_IN_PROGRESS The
following resource(s) failed to update: [MyDB]
```

更新のロールバックを続ける

時折、CloudFormation がスタック更新のロールバックを試みるときに、更新プロセス中に行ったすべての変更をロールバックできない場合があります。これは、UPDATE_ROLLBACK_FAILED 状態と呼ばれます。例えば、CloudFormation の外部で削除された古いデータベースのインスタンスにロールバックを開始するスタックがある場合などです。CloudFormation にはデータベースが削除されたことがわからないため、データベースインスタンスがまだ存在する前提でそれにロールバックしようとし、更新のロールバックが失敗します。

UPDATE_ROLLBACK_FAILED 状態のスタックは更新できませんが、動作状態 (UPDATE_ROLLBACK_COMPLETE) にロールバックできます。スタックを元の設定に戻すと、もう一度更新を試行することができます。

ほとんどの場合、スタックのロールバックを続けるには、更新のロールバックが失敗する原因となるエラーを修正する必要があります。それ以外の場合(たとえば、スタック操作のタイムアウト)では、変更を加えずに更新のロールバックを続けることができます。

 Note

ネストされたスタックを使用する場合には、親スタックをロールバックすると、すべての子スタックも同じような転送を試みます。

更新のロールバックを続けるには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックが所在する AWS リージョン を選択します。
3. [Stacks] ページで、更新するスタックを選択し、[スタックアクション]、[更新のロールバックの続行] の順に選択します。

[エラーのトラブルシューティング](#) の解決策でも解決できなかった場合は、高度なオプションを使用して、CloudFormation が正常にロールバックできないリソースをスキップできます。スキップするリソースの論理 ID を[検索](#)して入力する必要があります。更新の転送中ではなく、UpdateRollback 中に UPDATE_FAILED 状態になったリソースのみを指定します。

 Warning

CloudFormation は指定されたリソースのステータスを UPDATE_COMPLETE に設定し、引き続きスタックをロールバックします。ロールバックが完了したら、スキップされたリソースのステータスはスタックテンプレートのリソースのステータスと一致しません。別のスタックの更新を実行する前に、スタックまたはリソースを更新して、整合性を取る必要があります。そうしないと、以降のスタックの更新が失敗して、スタックが回復不可能になる可能性があります。

スタックを正常にロールバックするために必要なリソースの最小数を指定します。たとえば、あるリソース更新が失敗したために、それに依存するリソースが失敗することがあります。この場合、依存リソースをスキップする必要はない可能性があります。

ネストされたスタックの一部であるリソースをスキップするには、次の形式を使用します: *NestedStackName.ResourceLogicalID*。ResourcesToSkip リストでスタックリソース (Type: AWS::CloudFormation::Stack) の論理 ID を指定す

る場合、対応する組み込みスタックは以下のいずれかの状態である必要があります:
DELETE_IN_PROGRESS、DELETE_COMPLETE、または DELETE_FAILED。

更新のロールバックを続けるには (AWS CLI)

- ロールバックを続けるスタックの ID を指定するために、[continue-update-rollback](#) コマンドで `--stack-name` オプションを使用します。

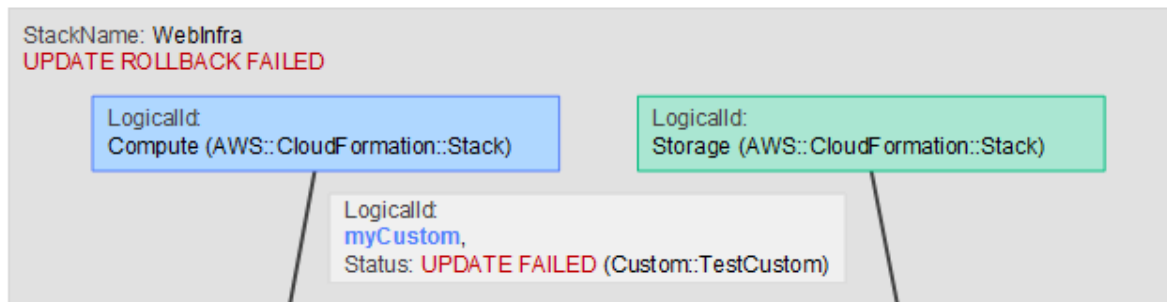
失敗した、ネストされたスタックの更新からのロールバックを続ける

複数のスタックが互いにネストされている場合、複数のネストレベル全体でリソースをスキップして、スタック階層全体を動作状態に戻す必要がある場合があります。

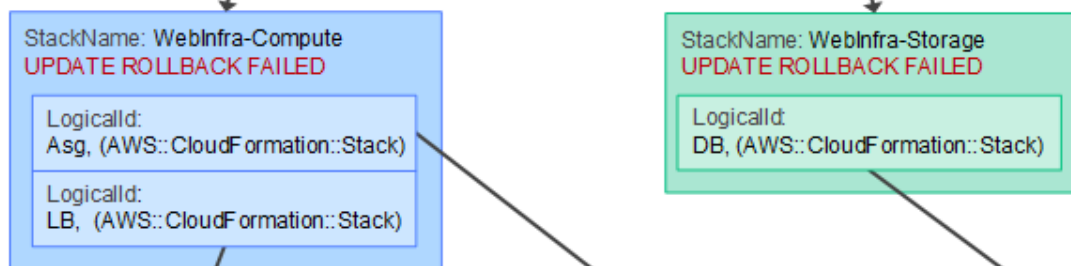
例えば、WebInfra というルートスタックがあり、その中に WebInfra-Compute と WebInfra-Storage の 2 つの小さなスタックが含まれているとします。これらの 2 つのスタック内にも、独自のネストされたスタックがあります。

更新中に問題が発生し、更新プロセスが失敗した場合、次の図に示すように、スタック階層全体が UPDATE_ROLLBACK_FAILED 状態になる可能性があります。

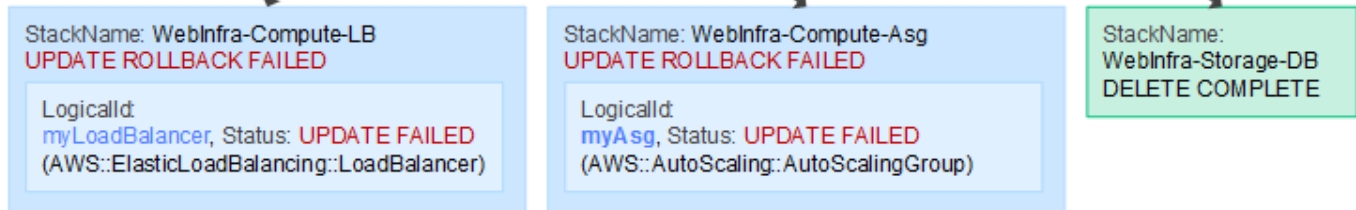
Root



Level 1



Level 2



Note

この例でスタック名は単純にするために切り捨てられています。子スタック名は通常 CloudFormation によって生成され、一意のランダムな文字列を含んでいるため、実際の名前はユーザーフレンドリーではない可能性があります。

continue-update-rollback コマンドを使用してルートスタックを操作可能な状態にするには、--resources-to-skip オプションを使用して、ロールバックに失敗したリソースをスキップする必要があります。

以下の continue-update-rollback の例は、前回失敗したスタック更新からのロールバック操作を再開します。この例では、--resources-to-skip オプションには次の項目が含まれます。

- *myCustom*
- *WebInfra-Compute-Asg.myAsg*
- *WebInfra-Compute-LB.myLoadBalancer*
- *WebInfra-Storage.DB*

ルートスタックのリソースの場合、必要なのは論理 ID を指定することだけです (例: *myCustom*)。ただし、ネストされたスタックに含まれるリソースについては、ネストされたスタック名とその論理 ID の両方をピリオドで区切って指定する必要があります。例えば、*WebInfra-Compute-Asg.myAsg*。

```
aws cloudformation continue-update-rollback --stack-name WebInfra \  
  --resources-to-skip myCustom WebInfra-Compute-Asg.myAsg WebInfra-Compute-  
  LB.myLoadBalancer WebInfra-Storage.DB
```

ネストされたスタックのスタック名を検索するには

子スタックのスタック ID または Amazon リソースネーム (ARN) 内で見つけることができます。

次の ARN の例は、WebInfra-Storage-Z2VKC706XKXT という名前のスタックを参照しています。

```
arn:aws:cloudformation:us-east-1:123456789012:stack/WebInfra-Storage-Z2VKC706XKXT/  
ea9e7f90-54f7-11e6-a032-028f3d2330bd
```

ネストされたスタックの論理 ID を検索するには

子スタックの論理 ID は、親のテンプレート定義で検索できます。この図では、WebInfra-Storage-DB 子スタックの LogicalId は、親 WebInfra-Storage の DB です。

CloudFormation コンソールでは、[Resources] (リソース) タブ、または [Events] (イベント) タブのスタックリソースの [Logical ID] (論理 ID) 列で論理 ID を見つけることもできます。詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

スタックが失敗する原因を判断する

スタックの作成が失敗する場合は、CloudFormation がスタック失敗の根本原因である可能性が高いイベントの判断に役立ちます。シナリオと許可によっては、[イベント] で提供された [ステータスの理由] が明確でない場合でも、AWS CloudTrail イベントで根本原因に関する詳細情報を確認できる場合があります。

スタックが失敗する根本原因を判断するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. [スタック] ページで、失敗したスタックを選択します。
3. [イベント] タブを選択します。
4. [根本原因を検出] を選択します。CloudFormation は障害を分析し、特定のイベントの [ステータス] に [考えられる根本原因] ラベルを追加することで、障害の原因である可能性のあるイベントを示します。CloudFormation コンソールのステータスの詳細な説明については、[ステータスの理由] を表示します。
5. 失敗の原因の詳細については、[考えられる根本原因] ラベルが付いた [失敗] ステータスを選択します。シナリオと許可に応じて、詳細な CloudTrail イベントを確認できる場合があります。[ステータス] を選択すると、次のような結果が得られる可能性があります。
 - この問題に関連する CloudTrail イベントが使用可能であり、解決に役立つ可能性があります。CloudTrail イベントを表示します。
 - この問題に関連して解決に役立つ可能性のある CloudTrail イベントは見つかりませんでした。
 - 現在の許可は、CloudTrail イベントを表示するためのアクセスを許可していません。詳細はこちら。
 - 使用可能な CloudTrail イベントを確認しています。数分後にもう一度確認してください。
 - CloudTrail イベントの取得中にエラーが発生しました。手動で検査する場合は、CloudTrail コンソールにアクセスしてください。
6. [ステータスの理由] に示された理由が明確でなく、根本原因で CloudTrail コンソールへのリンクが表示されている場合は、リンクを開いてイベントを表示し、詳細な根本原因を見つけます。

CloudTrail イベントの詳細については、「[Understanding CloudTrail events](#)」および「[CloudTrail record contents](#)」を参照してください。

CloudTrail イベント履歴の詳細については、「[Working with CloudTrail Event history](#)」を参照してください。

Note

ネストされたスタックは [根本原因を検出] をサポートしません。

リソースのプロビジョニング時における失敗への対応方法を選択する

スタックオペレーションが失敗した場合でも、既に正常にプロビジョニングされたリソースを毎回ロールバックし、最初からやり直す必要はありません。その代わりに、CREATE_FAILED または UPDATE_FAILED ステータスのリソースをトラブルシューティングして、問題が発生した箇所からプロビジョニングを再開することができます。

これを行うには、[正常にプロビジョニングされたリソースの保持] オプションを有効にする必要があります。このオプションは、すべてのスタックデプロイと変更セットオペレーションで利用できます。

- スタックの作成で [正常にプロビジョニングされたリソースの保持] オプションを選択すると、CloudFormation は正常に作成されたリソースの状態を保持しますが、失敗したリソースは次の更新オペレーションが実行されるまで失敗状態のままにしておきます。
- 更新オペレーションや変更セットオペレーション中に [正常にプロビジョニングされたリソースの保持] を選択すると、成功状態のリソースが保持されると同時に、失敗したリソースが最後に認識された安定した状態にロールバックされます。失敗したリソースは UPDATE_FAILED 状態になります。最後の安定状態が確認されていないリソースは、次のスタックオペレーション時に削除されます。

トピック

- [スタック障害オプションの概要](#)
- [スタックのロールバックを一時停止するために必要な条件](#)
- [正常にプロビジョニングされたリソースを保持する \(コンソール\)](#)
- [正常にプロビジョニングされたリソースを保持する \(AWS CLI\)](#)

スタック障害オプションの概要

CloudFormation コンソール、API、または AWS CLI からオペレーションを実行する前に、プロビジョニングされたリソースの失敗時の動作を指定します。次に、他の変更を加えずに、リソースのデプロイプロセスを続行します。運用上の障害が発生した場合、CloudFormation は各独立したプロビジョニングパスの最初の障害時に停止します。CloudFormation は、リソース間の依存関係を特定して、独立したプロビジョニングアクションを並列化します。その後、障害が発生するまで、各独立したプロビジョニングパスでリソースのプロビジョニングに進みます。1 つのパスで障害が発生しても、他のプロビジョニングパスには影響しません。CloudFormation は、完了するか、別の障害で停止するまで、リソースのプロビジョニングを続けます。

問題を修正してデプロイプロセスを続行します。CloudFormation は、以前に正常にプロビジョニングできなかったリソースに対するプロビジョニングアクションを再試行する前に、必要な更新を実行します。[Retry] (再試行)、[Update] (更新)、または [Roll back] (ロールバック) オペレーションを送信して、問題を修正します。例えば、Amazon EC2 インスタンスのプロビジョニング中に、EC2 インスタンスの作成オペレーション中に失敗した場合、失敗したリソースをすぐにロールバックするのではなく、そのエラーを調査する必要がある場合があります。システムステータスチェックとインスタンスステータスチェックを確認し、問題が解決したら [Retry] (再試行) オペレーションを選択できます。

スタックオペレーションが失敗し、[Stack failure options] (スタック障害のオプション) メニューから [Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースの保存) を指定した場合、次のオプションを選択できます。

- [Retry] (再試行) - 失敗したリソースでプロビジョニングオペレーションを再試行し、スタックオペレーションが正常に完了するか、次の失敗が発生するまでテンプレートのプロビジョニングを続行します。AWS Identity and Access Management (IAM) アクセス許可など、テンプレートの変更を必要としない問題が原因でリソースのプロビジョニングに失敗した場合に、このオプションを選択します。
- [Update] (更新) - プロビジョニングされたリソースは、テンプレートの更新時に更新されます。作成または更新に失敗したリソースは再試行されます。テンプレートエラーが原因でリソースのプロビジョニングに失敗し、テンプレートを変更した場合は、このオプションを選択します。FAILED 状態にあるスタックを更新する場合、スタックの更新を続行するには、[Stack failure options] (スタックの障害オプション) で [Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースを保持する) を選択する必要があります。
- [Roll back] (ロールバック) - CloudFormation はスタックを最後の既知の安定状態にロールバックします。

スタックのロールバックを一時停止するために必要な条件

CloudFormation が正常に作成されたリソースを自動的にロールバックして削除しないようにするには、次の条件を満たす必要があります。

1. スタックを作成または更新するときに、[正常にプロビジョニングされたリソースの保持] オプションを選択すること。そうすることで、スタックオペレーション全体が失敗した場合でも、正常に作成されたリソースは削除しないよう CloudFormation に指示します。
2. スタックオペレーションが失敗していること。つまり、スタックのステータスが CREATE_FAILED または UPDATE_FAILED である必要があります。

 Note

不変の更新の種類はサポートされていません。

正常にプロビジョニングされたリソースを保持する (コンソール)

Create stack

スタックの作成オペレーション中に正常にプロビジョニングされたリソースを保持するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページでは、右上の [スタックの作成] を選択してから、[新しいリソースを使用 (標準)] を選択します。
3. [前提条件 - テンプレートの準備] で、[既存のテンプレートを選択] を選択します。
4. [Specify template (テンプレートの指定)] で、スタックテンプレートを含む S3 バケットの URL を指定するか、スタックテンプレートファイルをアップロードするかを選択します。その後、[Next] を選択します。
5. [Specify stack details] (スタック詳細の指定) ページで、[Stack name] (スタックの名前) ボックスにスタック名を入力します。
6. [Parameters] (パラメータ) セクションで、スタックテンプレートで定義されているパラメータを指定します。

デフォルト値を持つパラメータを使用または変更できます。

7. パラメータ値が適切であることを確認したら、[Next] (次へ) を選択します。
8. [Configure stack options] (スタックオプションを設定する) ページでは、スタックのオプションを追加設定できます。
9. [Stack failure options] (スタック障害オプション) の場合、[Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースを保持する) を選択します。
10. スタックオプションが適切であることを確認したら、[Next] (次へ) を選択します。
11. [Review] (確認) ページでスタックを確認し、[Create stack] (スタックの作成) を選択します。

結果: 作成に失敗したリソースは、スタックオペレーションで障害が発生したときにスタックがロールバックされないように、スタックステータスを CREATE_FAILED に移行します。正常にプ

ロビジョニングされたリソースは、CREATE_COMPLETE 状態にあります。[Stack events] (スタックイベント) タブでスタックを監視できます。

Update stack

スタックの更新オペレーション中に正常にプロビジョニングされたリソースを保持するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 更新するスタックを選択し、次に [Update] (更新) を選択します。
3. [Update stack] (スタックの更新) ページで、以下のいずれかのオプションを使用してスタックのテンプレートを選択します。
 - 既存のテンプレートを使用する
 - [Replace current template] (現在のテンプレートを置換する)
 - Infrastructure Composer でテンプレートを編集する

設定を受け入れて、[Next] (次へ) を選択してください。

4. [Specify stack details] (スタックの詳細を指定する) ページで、スタックテンプレートで定義されているパラメータを指定します。

デフォルト値を持つパラメータを使用または変更できます。

5. パラメータ値が適切であることを確認したら、[Next] (次へ) を選択します。
6. [Configure stack options] (スタックオプションを設定する) ページでは、スタックのオプションを追加設定できます。
7. [Behavior on provisioning failure] (プロビジョニングの失敗時の動作) で、[Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースを保持する) を選択します。
8. スタックオプションが適切であることを確認したら、[Next] (次へ) を選択します。
9. [Review] (確認) ページでスタックを確認し、[Update stack] (スタックの更新) を選択します。

結果: 更新に失敗したリソースは、スタックステータスを UPDATE_FAILED に移行させ、最後の既知の安定状態にロールバックします。最後の安定状態が確認されていないリソースは、次のスタックオペレーション時に CloudFormation によって削除されます。正常にプロビジョニン

グされたリソースは、CREATE_COMPLETE または UPDATE_COMPLETE 状態にあります。[Stack events] (スタックイベント) タブでスタックを監視できます。

Change set

Note

ステータスが CREATE_FAILED または UPDATE_FAILED のスタックに対して変更セットを開始できますが、ステータスが UPDATE_ROLLBACK_FAILED のスタックに対しては開始できません。

変更セットオペレーション中に正常にプロビジョニングされたリソースを保持するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 開始する変更セットを含むスタックを選択し、[Change sets] (変更セット) タブを選択します。
3. 変更セットを選択し、[Execute] (実行) を選択します。
4. [Execute change set] (変更セットの実行) の場合、[Preserve successfully provisioned resources] (正常にプロビジョニングされたリソースを保持する) オプションを選択します。
5. [Execute change set] (変更セットの実行) を選択します。

結果: 更新に失敗したリソースは、スタックステータスを UPDATE_FAILED に移行させ、最後の既知の安定状態にロールバックします。最後の安定状態が確認されていないリソースは、次のスタックオペレーション時に CloudFormation によって削除されます。正常にプロビジョニングされたリソースは、CREATE_COMPLETE または UPDATE_COMPLETE 状態にあります。[Stack events] (スタックイベント) タブでスタックを監視できます。

正常にプロビジョニングされたリソースを保持する (AWS CLI)

Create stack

スタックの作成オペレーション中に正常にプロビジョニングされたリソースを保持するには

[create-stack](#) オペレーション中に `--disable-rollback` オプションまたは `on-failure DO_NOTHING` 列挙を指定します。

1. `create-stack` コマンドに `--disable-rollback` オプションでスタック名とテンプレートを指定します。

```
aws cloudformation create-stack --stack-name myteststack \  
  --template-body file://template.yaml \  
  --disable-rollback
```

このコマンドは、以下の出力を返します。

```
{  
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"  
}
```

2. `describe-stacks` コマンドを使用してスタックの状態を記述します。

```
aws cloudformation describe-stacks --stack-name myteststack
```

このコマンドは、以下の出力を返します。

```
{  
  "Stacks": [  
    {  
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",  
      "Description": "AWS CloudFormation Sample Template",  
      "Tags": [],  
      "Outputs": [],  
      "StackStatusReason": "The following resource(s) failed to create:  
[MyBucket]",  
      "CreationTime": "2013-08-23T01:02:15.422Z",  
      "Capabilities": [],  
      "StackName": "myteststack",  
      "StackStatus": "CREATE_FAILED",  
      "DisableRollback": true  
    }  
  ]  
}
```

Update stack

スタックの更新オペレーション中に正常にプロビジョニングされたリソースを保持するには

1. `update-stack` コマンドに `--disable-rollback` オプションで既存のスタック名とテンプレートを指定します。

```
aws cloudformation update-stack --stack-name myteststack \  
  --template-url https://s3.amazonaws.com/amzn-s3-demo-bucket/updated.template  
  --disable-rollback
```

このコマンドは、以下の出力を返します。

```
{  
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"  
}
```

2. `describe-stacks` または `describe-stack-events` コマンドを使用してスタックの状態を記述します。

```
aws cloudformation describe-stacks --stack-name myteststack
```

このコマンドは、以下の出力を返します。

```
{  
  "Stacks": [  
    {  
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",  
      "Description": "AWS CloudFormation Sample Template",  
      "Tags": [],  
      "Outputs": [],  
      "CreationTime": "2013-08-23T01:02:15.422Z",  
      "Capabilities": [],  
      "StackName": "myteststack",  
      "StackStatus": "UPDATE_COMPLETE",  
      "DisableRollback": true  
    }  
  ]  
}
```

Change set

Note

ステータスが `CREATE_FAILED` または `UPDATE_FAILED` のスタックに対して変更セットを開始できますが、ステータスが `UPDATE_ROLLBACK_FAILED` のスタックに対しては開始できません。

変更セットオペレーション中に正常にプロビジョニングされたリソースを保持するには

[execute-change-set](#) オペレーション中に `--disable-rollback` オプションを指定します。

1. `execute-change-set` コマンドに `--disable-rollback` オプションでスタック名とテンプレートを指定します。

```
aws cloudformation execute-change-set --stack-name myteststack \  
--change-set-name my-change-set --template-body file://template.yaml
```

このコマンドは、以下の出力を返します。

```
{  
  "Id": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/my-change-set/  
bc9555ba-a949-xmpl-bfb8-f41d04ec5784",  
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"  
}
```

2. `--disable-rollback` オプションで変更セットを開始します。

```
aws cloudformation execute-change-set --stack-name myteststack \  
--change-set-name my-change-set --disable-rollback
```

3. `describe-stacks` または `describe-stack-events` コマンドを使用してスタックのステータスを確認します。

```
aws cloudformation describe-stack-events --stack-name myteststack
```

このコマンドは、以下の出力を返します。


```
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
      "EventId": "49c966a0-7b74-11ea-8071-024244bb0672",
      "StackName": "myteststack",
      "LogicalResourceId": " MyBucket",
      "PhysicalResourceId": "myteststack-MyBucket-abcdefghijkl",
      "ResourceType": "AWS::S3::Bucket",
      "Timestamp": "2020-04-10T21:43:17.015Z",
      "ResourceStatus": "UPDATE_FAILED"
      "ResourceStatusReason": "User XYZ is not allowed to perform S3::UpdateBucket on MyBucket"
    }
  ]
}
```

4. 権限エラーを修正してオペレーションを再試行してください。

```
aws cloudformation update-stack --stack-name myteststack \
  --use-previous-template --disable-rollback
```

このコマンドは、以下の出力を返します。

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

5. `describe-stacks` または `describe-stack-events` コマンドを使用してスタックの状態を記述します。

```
aws cloudformation describe-stacks --stack-name myteststack
```

このコマンドは、以下の出力を返します。

```
{
  "Stacks": [
    {
```

```
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "Description": "AWS CloudFormation Sample Template",
    "Tags": [],
    "Outputs": [],
    "CreationTime": "2013-08-23T01:02:15.422Z",
    "Capabilities": [],
    "StackName": "myteststack",
    "StackStatus": "UPDATE_COMPLETE",
    "DisableRollback": true
  }
]
```

スタックをロールバックする

[rollback-stack](#) コマンドを使用して、スタックステータスが `CREATE_FAILED` または `UPDATE_FAILED` のスタックを最後の安定した状態にロールバックできます。

以下の `rollback-stack` コマンドは、指定されたスタックをロールバックします。

```
aws cloudformation rollback-stack --stack-name myteststack
```

このコマンドは、以下の出力を返します。

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

Note

`rollback-stack` オペレーションは、スタックに最新の既知の安定状態が含まれていない場合、スタックを削除します。

ロールバックトリガーを使用して、アラーム違反時に CloudFormation スタックをロールバックする

ロールバックトリガーを使用することで、スタックの作成中および更新中にアプリケーションの状態を CloudFormation で監視し、指定されたアラームのしきい値をアプリケーションが超過した場合は、そのオペレーションをロールバックすることができます。作成したロールバックトリガーごとに、CloudFormation がモニタリングする CloudWatch アラームを指定します。CloudFormation は、スタックの作成または更新オペレーション時、およびすべてのリソースがデプロイされてから指定された時間、指定されたアラームをモニタリングします。スタックオペレーションまたはモニタリング期間中にいずれかのアラームが ALARM 状態になった場合、CloudFormation はスタックオペレーション全体をロールバックします。

モニタリング時間は、デフォルト値の 0 から最大 180 分まで設定できます。この時間中、CloudFormation はスタックの作成、または更新オペレーションによってすべての必要なリソースがデプロイされた後、すべてのロールバックトリガーをモニタリングします。スタックオペレーションまたはこのモニタリング期間中にいずれかのアラームが ALARM 状態になった場合、CloudFormation はスタックオペレーション全体をロールバックします。更新オペレーションの場合、その後アラームが ALARM 状態にならずにモニタリング期間が切れると、CloudFormation は通常どおり古いリソースの破棄に進みます。モニタリング時間を設定したがどのロールバックトリガーも指定しない場合でも、CloudFormation は更新オペレーション用に古いリソースがクリーンアップされるまで指定された時間待機します。このモニタリング期間を使用して必要な手動スタック検証を実行し、必要に応じてスタックの作成または更新を手動でキャンセルできます。モニタリング時間を 0 分に設定した場合でも、CloudFormation はスタックの作成および更新オペレーション時にロールバックトリガーをモニタリングし、アラームが ALARM 状態になった場合にオペレーションをロールバックします。アラームを超過しない更新オペレーションの場合、オペレーションが完了するとすぐに古いリソースの破棄が開始されます。

デフォルトでは、CloudFormation は、アラームが INSUFFICIENT_DATA 状態ではなく ALARM 状態になった場合にのみ、スタックオペレーションをロールバックします。アラームが INSUFFICIENT_DATA 状態になった場合も CloudFormation にスタックオペレーションをロールバックさせるには、欠落データを breaching として処理するように CloudWatch アラームを編集します。詳細については、Amazon CloudWatch ユーザーガイドの「[CloudWatch アラームが欠落データを処理する方法の設定](#)」を参照してください。

更新オペレーション中にスタックをロールバックしている時は、CloudFormation はロールバックトリガーをモニタリングしません。

最大 5 個のロールバックトリガーを追加できます。ロールバックトリガーを追加するには、CloudWatch アラームの Amazon リソースネーム (ARN) を指定します。現在、AWS::CloudWatch::Alarm タイプと AWS::CloudWatch::CompositeAlarm タイプをロールバックトリガーとして使用できます。CloudWatch アラームの詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon CloudWatch でのアラームの使用](#)」を参照してください。

指定された CloudWatch アラームが欠落している場合、スタックオペレーション全体が失敗してロールバックします。

CloudWatch へのアクセスには認証情報が必要であることに注意してください。これらの認証情報は、リソースに関する CloudWatch メトリクスデータの取得などの AWS リソースへの許可が必要です。詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon CloudWatch に対する認証とアクセスコントロール](#)」を参照してください。

スタックの作成中または更新中にロールバックトリガーを追加する

スタックの作成時または更新時にロールバックトリガーを追加するには (コンソール)

1. スタックを作成または更新するときは、[スタックオプションの設定] ページの [詳細オプション] で、[ロールバック設定] セクションを展開します。
2. 0〜180 分の監視時間を入力します。デフォルト値は 0 です。
3. ロールバックトリガーとして使用する CloudWatch アラームまたは複合アラームの ARN を指定し、[Add CloudWatch alarm ARN] (CloudWatch アラームの ARN を追加) を選択します。

例えば、以下は CloudWatch アラームまたは複合アラーム `arn:aws:cloudwatch:us-east-1:123456789012:alarm:MyAlarmName` の ARN です。

4. [次へ] を選択して、スタックの詳細を確認します。
5. 準備ができたなら、[送信]を選択してスタックを作成または更新します。

スタックの作成または更新時にロールバックトリガーを追加するには (AWS CLI)

[create-stack](#) コマンドを使用するか、または `--rollback-configuration` オプションを設定して [update-stack](#) コマンドを使用します。

例えば、次の `update-stack` コマンドでは、`MyCompositeAlarm` を 5 分間のモニタリング時間のロールバックトリガーとして設定しています。

```
aws cloudformation update-stack --stack-name MyStack \
```

```
--use-previous-template \  
--rollback-configuration \  
"RollbackTriggers=[{Arn=arn:aws:cloudwatch:us-  
east-1:123456789012:alarm:MyCompositeAlarm,Type=AWS::CloudWatch::CompositeAlarm}],MonitoringTim
```

変更セットにロールバックトリガーを追加する

変更セットにロールバックトリガーを追加するには (コンソール)

1. 変更セットを作成するときは、[スタックオプションの設定] ページの [詳細オプション] にある [ロールバック設定] セクションを展開します。
2. 0〜180 分の監視時間を入力します。デフォルト値は 5 です。
3. ロールバックトリガーとして使用する CloudWatch アラームまたは複合アラームの ARN を指定し、[Add CloudWatch alarm ARN] (CloudWatch アラームの ARN を追加) を選択します。

例えば、以下は CloudWatch アラームまたは複合アラーム `arn:aws:cloudwatch:us-east-1:123456789012:alarm:MyAlarmName` の ARN です。

4. [次へ] を選択し、変更セットの詳細を確認します。
5. 準備ができたなら、[変更セットの作成] を選択して変更セットを作成します。

変更セットにロールバックトリガーを追加するには (AWS CLI)

--rollback-configuration オプションで [create-change-set](#) コマンドを使用します。

スタックのロールバックトリガーを表示する

スタックのロールバックトリガーを表示するには、「ロールバック構成」セクションを参照してください。

1. [スタック] ページで、表示するスタックを左側のリストから選択します。
2. [スタック情報] タブで、[ロールバック設定] セクションを展開します。

変更セットのロールバックトリガーを表示する

変更セットのロールバックトリガーを表示するには、「Rollback configuration」(ロールバック構成) セクションを参照してください。

1. [スタック] ページで、表示するスタックを左側のリストから選択します。

2. [変更セット] タブを選択し、表示する変更セットを選択します。
3. [入力] タブを選択し、[ロールバック設定] セクションを表示します。

ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する

CloudFormation を通じてリソースを管理している場合でも、ユーザーは CloudFormation の外部でそれらのリソースを変更できます。ユーザーは、リソースを作成した基盤となるサービスを使用してリソースを直接編集できます。例えば、Amazon EC2 コンソールを使用して、CloudFormation スタックの一部として作成されたサーバーインスタンスを更新できます。一部の変更は偶発的になされますが、時間的制約のある操作上のイベントに対応するために意図的になされる場合もあります。しかし、CloudFormation の外部で行われた変更はスタックの更新または削除オペレーションを複雑にする可能性があります。ドリフト検出を使用して、CloudFormation 管理外で構成変更がされたスタックリソースを識別できます。その後、スタックリソースがスタックテンプレート内の定義と再度同期するように是正措置を行うことができます。たとえば、ドリフトされたリソースがテンプレート定義と一致するように直接更新します。ドリフトを解決することにより、構成の一貫性と正常なスタックオペレーションを確実にします。

トピック

- [ドリフトとは](#)
- [ドリフト検出ステータスコード](#)
- [ドリフトを検出する際の考慮事項](#)
- [CloudFormation スタック全体のドリフトを検出する](#)
- [個々のスタックリソースのドリフトを検出](#)
- [インポートオペレーションでドリフトを解決する](#)

ドリフトとは

ドリフトの検出により、スタックの実際の構成が、意図した構成と異なっていたり、ドリフトしていないか確認できます。CloudFormation を使用して、スタック全体、またはスタック内の個々のリソースのドリフトを検出します。リソースの実際のプロパティ値が予期されるプロパティ値と異なる場合、そのリソースはドリフトしたと見なされます。これには、プロパティまたはリソースが削除されたかどうかも含まれます。スタックの 1 つ以上のリソースがドリフトした場合、スタックはドリフトしたと見なされます。

リソースがドリフトしたかどうかを判断するために、CloudFormation はスタックテンプレートとテンプレートパラメータとして指定された値で定義されているように、期待されるリソースプロパティ値を決定します。CloudFormation は、それらの期待値を現在スタックに存在しているリソースプロパティの実際の値と比較します。1 つ以上のリソースのプロパティが削除された場合、またはリソースの値が変更された場合、リソースはドリフトしたと見なされます。

CloudFormation は、ドリフトしたスタック内の各リソースに関する詳細情報を生成します。

CloudFormation は、これらのドリフト検出をサポートしている AWS リソースのドリフトを検出します。ドリフト検出をサポートしないリソースには、ドリフトステータス `NOT_CHECKED` が割り当てられます。ドリフト検出をサポートする AWS リソースのリストについては、「[リソースタイプのサポート](#)」を参照してください。

さらに、CloudFormation は、プロビジョニング可能なプライベートリソースタイプ、つまりプロビジョニングタイプが `FULLY_MUTABLE` または `IMMUTABLE` のいずれかであるドリフト検出をサポートします。プライベートリソースタイプのリソースでドリフト検出を実行するには、アカウントに登録されたリソースタイプのデフォルトバージョンがプロビジョニング可能である必要があります。リソースのプロビジョニングタイプの詳細については、「AWS CloudFormation API リファレンス」の [DescribeType](#) アクションおよび「AWS CLI コマンドリファレンス」の [DescribeType](#) コマンドの `ProvisioningType` パラメータをご覧ください。プライベートリソースの詳細については、「[CloudFormation レジストリによる拡張機能の管理](#)」をご参照ください。

次のステータスでスタックのドリフト検出を実行できま

す。`CREATE_COMPLETE`、`UPDATE_COMPLETE`、`UPDATE_ROLLBACK_COMPLETE`、および `UPDATE_ROLLBACK_FAILED`。

スタックのドリフトを検出すると、CloudFormation はそのスタックに属するネストされたスタックのドリフトを検出しません。詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。代わりに、ネストしたスタックに対して直接ドリフト検出オペレーションを開始できます。

Note

CloudFormation は、スタックテンプレートを通じて、またはテンプレートパラメータを指定することによって、明示的に設定されたプロパティについてのみドリフトを判断します。これには、リソースプロパティのデフォルト値は含まれません。ドリフトを決定する目的で CloudFormation にリソースプロパティを追跡させるには、デフォルト値に設定している場合でも、プロパティ値を明示的に設定します。

ドリフト検出ステータスコード

このセクションの表では、ドリフト検出で使用するさまざまなステータスタイプについて説明しています。

- [Drift detection operation status (ドリフト検出オペレーションステータス)] は、ドリフト動作の現在の状態を示します。
- ドリフトステータス

スタックセットの場合、スタックセットに属するスタックインスタンスのドリフトステータスに基づいて、スタックセット全体のドリフトステータスについて説明します。

スタックインスタンスの場合、関連付けられたスタックのドリフトステータスに基づいて、スタックインスタンスのドリフトステータスについて説明します。

スタックの場合、これは、そのリソースのドリフトステータスに基づいて、スタック全体のドリフトステータスを表します。

- [リソースのドリフトステータス] は、個々のリソースのドリフトステータスを表します。

次の表は、CloudFormation がスタックドリフト検出オペレーションに割り当てるステータスコードの一覧です。

ドリフト検知オペレーションステータス	説明
DETECTION_COMPLETE	スタックドリフト検出オペレーションは、ドリフト検出をサポートするスタック内のすべてのリソースに対して正常に完了しました。
DETECTION_FAILED	スタックドリフト検出オペレーションは、スタック内の少なくとも 1 つのリソースに対して失敗しました。CloudFormation がドリフト検出を正常に完了したリソースで、結果が利用可能になります。
DETECTION_IN_PROGRESS	スタックドリフト検出オペレーションは現在進行中です。

次の表は、CloudFormation がスタックに割り当てるドリフトステータスコードの一覧です。

ドリフトステータス	説明
DRIFTED	スタックの場合: スタックは、意図したテンプレート構成とは異なるか、ドリフトしています。スタックの 1 つ以上のリソースがドリフトした場合、スタックはドリフトしたと見なされます。 スタックインスタンスの場合: スタックインスタンスに関連付けられたスタックがドリフトした場合、スタックインスタンスはドリフトしたと見なされます。 スタックセットの場合: 1 つ以上のスタックインスタンスがドリフトした場合、スタックセットはドリフトしたと見なされます。
NOT_CHECKED	CloudFormation では、スタック、スタックセット、またはスタックインスタンスが、予想されるテンプレート構成と異なるかどうかをチェックしません。
IN_SYNC	サポートされている各リソースの現在の構成は、意図したテンプレート構成と一致します。ドリフト検出をサポートするリソースを持たないスタック、スタックセット、またはスタックインスタンスも IN_SYNC ステータスになります。

次の表は、CloudFormation がスタックリソースに割り当てるドリフトステータスコードの一覧です。

リソースのドリフトステータス	説明
DELETED	リソースが削除されたため、リソースは意図したテンプレート構成と異なります。
MODIFIED	リソースは意図したテンプレート構成とは異なります。
NOT_CHECKED	CloudFormation は、リソースが意図したテンプレート構成と異なるかどうかを確認していません。

リソースのドリフトステータス	説明
IN_SYNC	リソースの現在の構成は、意図したテンプレート構成と一致します。

次の表は、CloudFormation が意図したテンプレート構成とは異なるリソースプロパティに割り当てる差分タイプのステータスコードを示しています。

差分タイプのプロパティ	説明
ADD	配列またはリストのデータ型であるリソースプロパティに値が追加されました。
REMOVE	プロパティは現在のリソース構成から削除されました。
NOT_EQUAL	現在のプロパティ値は、スタックテンプレートで定義されている予測値とは異なります。

ドリフトを検出する際の考慮事項

スタックでドリフト検出を正常に実行するには、ユーザーは次のアクセス許可を持っている必要があります。

- スタックに含まれるドリフト検出をサポートする各リソースに対する読み取り権限です。たとえば、スタックに `AWS::EC2::Instance` リソースが含まれている場合、スタックでドリフト検出を実行するには `ec2:DescribeInstances` アクセス許可が必要です。
- `cloudformation:DetectStackDrift`
- `cloudformation:DetectStackResourceDrift`
- `cloudformation:BatchDescribeTypeConfigurations`

CloudFormation でアクセス許可を設定する方法の詳細については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」をご参照ください。

特定のエッジケースでは、CloudFormation が正確なドリフト結果を返すことができない場合があります。ドリフト検出結果を正しく解釈するために、これらのエッジケースに注意する必要があります。

- 場合によっては、プロパティ配列に含まれるオブジェクトが、実際にはリソースを担当する基盤となるサービスからプロパティに提供されたデフォルト値であった場合に、ドリフトとして報告されます。
- 一部のリソースは、関連するリソースとアタッチメントの関係をもちます。そのため、リソースは、同一のまたは別のテンプレートで定義された別のリソースのプロパティ値を実際アタッチまたは削除できます。たとえば、AWS::EC2::SecurityGroupIngress および AWS::EC2::SecurityGroupEgress リソースを使用して、AWS::EC2::SecurityGroup リソースから値をアタッチしたり削除したりできます。このような場合、CloudFormation はドリフト比較を実行する前にアタッチメントのスタックテンプレートを分析します。ただし、CloudFormation はスタック間ではこの分析を実行できないため、接続されているリソースが異なるスタックに存在する場合、正確なドリフト結果を返さない可能性があります。

ドリフト検出をサポートし、他のリソースからの添付ファイルを許可または要求するリソースには、次のものがあります。

リソースタイプ	添付リソースの種類
AWS::SNS::Topic	AWS::SNS::Subscription
AWS::IAM::User	AWS::IAM::UserToGroupAddition
AWS::IAM::Group	AWS::IAM::ManagedPolicy
AWS::IAM::Role	AWS::IAM::Policy
AWS::IAM::User	AWS::IAM::RolePolicy
AWS::ElasticLoadBalancingV2::Listener	AWS::ElasticLoadBalancingV2::ListenerCertificate
AWS::EC2::SecurityGroup	AWS::EC2::SecurityGroupEgress AWS::EC2::SecurityGroupIngress

- CloudFormation は、リソースの KMSKeyId プロパティに対してドリフト検出を実行しません。AWS KMS キーは複数のエイリアスによって参照される可能性があるため、CloudFormation はこのプロパティに対して常に正確なドリフト結果を保証することはできません。
- スタックテンプレートで指定できる特定のリソースプロパティがあります。その性質上、CloudFormation は結果として得られるスタックリソースのプロパティと比較できません。し

たがって、これらのプロパティをドリフト検出結果に含めることはできません。このようなプロパティは、大きく 2 つのカテゴリに分けられます。

- CloudFormation がスタックテンプレートの初期リソースプロパティ値にマッピングできないプロパティ値です。

例えば、CloudFormation は Lambda 関数のソースコードを [AWS::Lambda::Function](#) リソースの [Code](#) プロパティタイプにマッピングできません。そのため CloudFormation はそれをドリフト検出結果に含めることができません。

- リソースを担当するサービスが返さないプロパティ値です。

設計上、リソースが属するサービスでは返されることがない特定のプロパティ値があります。これらには、パスワードやその他の機密データなどの、公開されない機密情報が含まれる傾向があります。例えば、IAM サービスは [AWS::IAM::User LoginProfile](#) プロパティタイプの Password プロパティの値を返すことはないため、CloudFormation はそれをドリフト検出結果に含めることができません。

- 配列内のオブジェクトは実際にはサービスの既定値である場合があり、手動で追加されたドリフトではありません。
- 誤検知が発生した場合は、[CloudFormation コンソール] のフィードバックリンクを使用してご意見をご送信いただくか、[AWS re:Post](#) からご連絡ください。
- プロパティによっては、等しいが同一ではない入力値を持つことがあります。誤検出を防ぐには、想定した設定が実際の設定と一致していることを確認する必要があります。
- たとえば、リソースプロパティの予想される設定は 1,024 MB で、同じリソースプロパティの実際の設定は 1 GB になる可能性があります。1,024 MB と 1 GB は同等ですが同一ではありません。

このリソースプロパティでドリフト検出を実行すると、ドリフトの結果が通知されます。

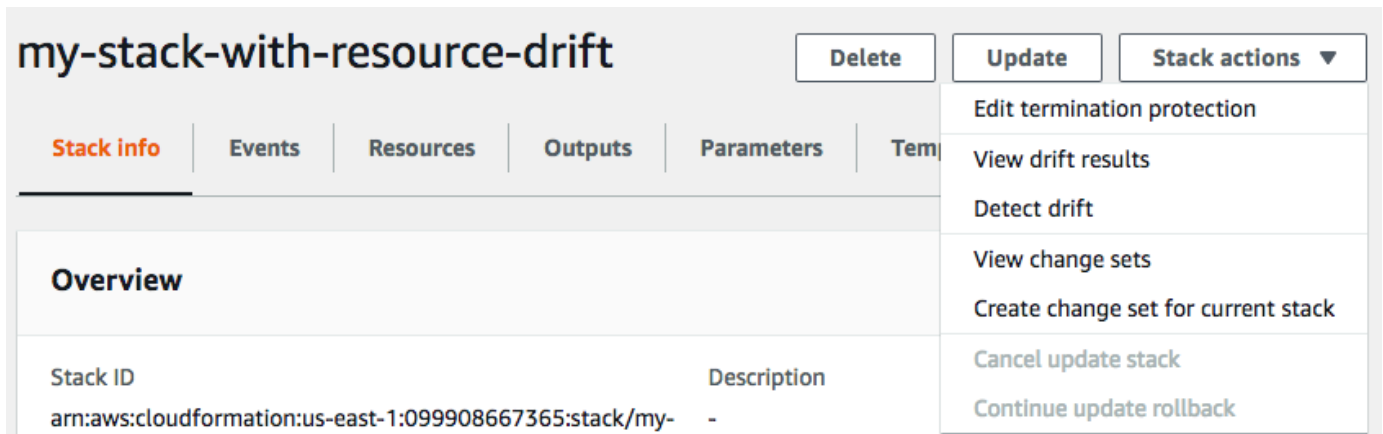
この誤検出を回避するには、リソースプロパティの想定される設定を 1,024 MB に変更してから、ドリフト検出を実行します。

CloudFormation スタック全体のドリフトを検出する

スタックでドリフト検出オペレーションを実行すると、スタックが予想されるテンプレート構成からドリフトしたかどうか判断され、ドリフト検出をサポートするスタック内の各リソースのドリフトステータスに関する詳細情報が返されます。

AWS マネジメントコンソール を使用してスタック全体のドリフトを検出するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. スタックのリストから、ドリフト検出を実行するスタックを選択します。スタックの詳細ペインで、[Stack actions (スタックアクション)] を選択し、[Detect drift (ドリフトの検出)] を選択します。



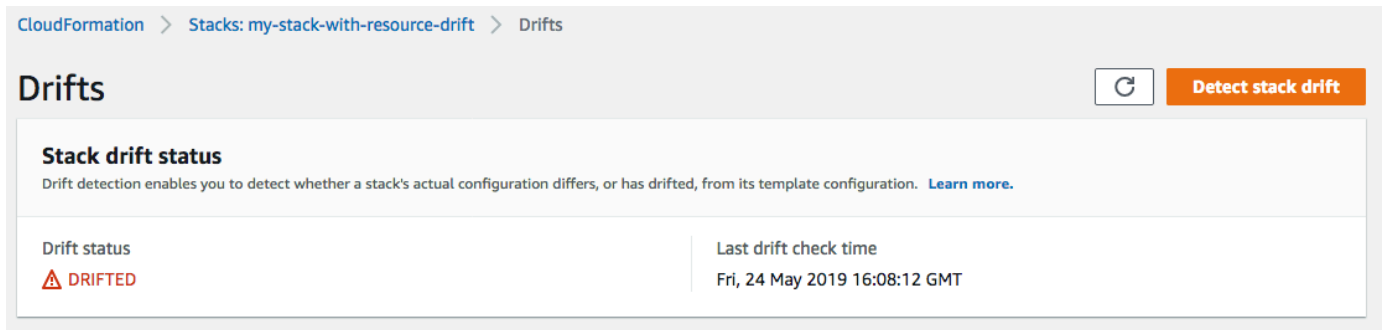
CloudFormation により、選択されたスタックに対してドリフト検出が開始されたことを示す情報バーが表示されます。

3. CloudFormation がドリフト検出オペレーションを完了するまで待ちます。ドリフト検出オペレーションが完了すると、CloudFormation によって、スタックの [Drift status] (ドリフトステータス) と [Last drift check time] (前回のドリフトチェック時間) が更新されます。これらのフィールドは、スタックの詳細ページの [Stack info] (スタック情報) ペインの [Overview] (概要) セクションにリスト表示されます。

スタックに含まれるリソースの数によっては、ドリフト検出オペレーションに数分かかることがあります。特定のスタックに対して同時に実行できるドリフト検出オペレーションは 1 つだけです。情報バーを閉じた後でも、CloudFormation によってドリフト検出オペレーションが継続されます。

4. スタックとそのリソースのドリフト検出結果を確認します。スタックを選択した状態で、[Stack actions (スタックアクション)] メニューから [View drift results (ドリフト結果の表示)] を選択します。

CloudFormation によって、スタックの全体的なドリフトステータス、およびスタックまたはその個々のリソースのいずれかで最後にドリフト検出が開始された時刻がリスト表示されます。スタックの 1 つ以上のリソースがドリフトした場合、スタックはドリフトしたと見なされます。



[Resource drift status] (リソースのドリフトステータス) のセクションに、CloudFormation で各スタックリソース、そのドリフトステータス、およびリソースで最後にドリフト検出が開始された時刻がリスト表示されます。各リソースの論理 ID と物理 ID は、それらを識別しやすくするために表示されます。さらに、ステータスが [MODIFIED] のリソースの場合、CloudFormation はリソースドリフトの詳細を表示します。

ドリフトステータスに基づいてリソースをソートするには、[ドリフトステータス] 列を使用します。

- 変更されたリソースの詳細を表示します。
- 変更されたリソースを選択した状態で、[View drift details (ドリフトの詳細を表示)] を選択します。

CloudFormation により、そのリソースのドリフト詳細ページが表示されます。このページに、リソースの意図した値と現在のプロパティ値、およびその 2 つの違いがリスト表示されます。

違いを強調表示するには、[違い] セクションでプロパティ名を選択します。

- 追加されたプロパティは [Details] (詳細) セクションの [Current] (最新の) 列で緑色で強調表示されます。
- 削除されたプロパティは [Details] (詳細) セクションの [Expected] (期待される) 列で赤色で強調表示されます。
- 値が変更されたプロパティは、[Expected] 列と [最新の] 列の両方で黄色で強調表示されます。

Differences (4)
< 1 >

☒	Property	Change	Expected value	Current value
☒	AttributeDefinitions.1	ADD	-	{"AttributeName": "stuff", "AttributeType": "S"}
☒	ProvisionedThroughput.ReadCapacityUnits	NOT_EQUAL	5	6
☒	ProvisionedThroughput.WriteCapacityUnits	NOT_EQUAL	5	4
☒	Tags	REMOVE	[{"Key": "test", "Value": "test"}]	-

Details

Expected

Actual

```

{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    },
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  },
  "Tags": [
    {
      "Key": "test",
      "Value": "test"
    }
  ]
}

```

```

{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    },
    {
      "AttributeName": "stuff",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 6,
    "WriteCapacityUnits": 4
  }
}

```

AWS CLI を使用してスタック全体のドリフトを検出するには

Important

古いデータが使用されることのないようにするため、スタックの [前回のドリフトチェック時刻] をチェックして、リソースのドリフト結果に表示されるタイムスタンプより前の時刻になっていることを確認します。

AWS CLI を使用してスタック全体のドリフトを検出するには、次の AWS CLI コマンドを使用します。

- `detect-stack-drift` はスタックでドリフト検出オペレーションを開始します。
 - `describe-stack-drift-detection-status` はスタックドリフト検出動作の状態を監視します。
 - `describe-stack-resource-drifts` はスタックドリフト検出オペレーションの詳細を確認します。
1. `detect-stack-drift` を使用してスタック全体のドリフトを検出します。スタックの名前または ARN を指定してください。このドリフト検出オペレーションのフィルターとして使用したい特定のリソースの論理 ID を指定することもできます。

```
aws cloudformation detect-stack-drift --stack-name my-stack-with-resource-drift
```

出力:

```
{
  "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample"
}
```

2. スタックドリフト検出オペレーションは長時間実行される可能性があるため、`describe-stack-drift-detection-status` を使用してドリフト操作のステータスを監視します。このコマンドは、`detect-stack-drift` コマンドによって返されたスタックドリフト検出 ID を受け取ります。

以下の例では、上記の `detect-stack-drift` の例から返されたスタックドリフト検出 ID を取得し、`describe-stack-drift-detection-status` へのパラメータとして渡します。このパラメータは、ドリフト検出オペレーションが完了したこと、単一のスタックリソースがドリフトしたこと、および結果としてスタック全体がドリフトしたと見なされることを示すオペレーションの詳細を返します。

```
aws cloudformation describe-stack-drift-detection-status --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

出力:

```
{
  "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
  "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample",
}
```



```

    "StackDriftStatus": "DRIFTED",
    "Timestamp": "2018-03-26T17:23:22.279Z",
    "DetectionStatus": "DETECTION_COMPLETE",
    "DriftedStackResourceCount": 1
  }

```

3. スタックドリフト検出オペレーションが完了したら、`describe-stack-resource-drifts` コマンドを使用して、ドリフトしたリソースの実際のプロパティ値と意図したプロパティ値を含む結果を確認します。

以下の例では `--stack-resource-drift-status-filters` オプションを使用し、変更または削除されたリソースのスタックドリフト情報を要求します。リクエストは、値が変更された2つのプロパティに関する詳細を含む、変更された1つのリソースに関する情報を返します。リソースは削除されていません。

```

aws cloudformation describe-stack-resource-drifts --stack-name my-stack-with-resource-drift --stack-resource-drift-status-filters MODIFIED DELETED

```

出力:

```

{
  "StackResourceDrifts": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
      "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0, \"DelaySeconds\":120, \"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\"}, \"maxReceiveCount\":12}, \"MessageRetentionPeriod\":345600, \"MaximumMessageSize\":262144, \"VisibilityTimeout\":60, \"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHC076H4\"}",
      "ResourceType": "AWS::SQS::Queue",
      "Timestamp": "2018-03-26T17:23:34.489Z",
      "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-stack-with-resource-drift-Queue-494PBHC076H4",
      "StackResourceDriftStatus": "MODIFIED",
      "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0, \"DelaySeconds\":20, \"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\"}, \"maxReceiveCount\":10}, \"MessageRetentionPeriod\":345600, \"MaximumMessageSize

```

```

\" :262144,\"VisibilityTimeout\":60,\"QueueName\": \"my-stack-with-resource-drift-
Queue-494PBHC076H4\"}],
  \"PropertyDifferences\": [
    {
      \"PropertyPath\": \"/DelaySeconds\",
      \"ActualValue\": \"120\",
      \"ExpectedValue\": \"20\",
      \"DifferenceType\": \"NOT_EQUAL\"
    },
    {
      \"PropertyPath\": \"/RedrivePolicy/maxReceiveCount\",
      \"ActualValue\": \"12\",
      \"ExpectedValue\": \"10\",
      \"DifferenceType\": \"NOT_EQUAL\"
    }
  ],
  \"LogicalResourceId\": \"Queue\"
}
]
}

```

個々のスタックリソースのドリフトを検出

スタック全体ではなく、スタック内の特定のリソースの変動を検出できます。これは、特定のリソースが意図されたテンプレート構成に一致しているかどうかを再度判断するだけでよい場合に特に役立ちます。

リソースでドリフト検出を実行すると、CloudFormation は、必要に応じて、スタックドリフトステータス全体および [Last drift check time] (前回のドリフトチェック時刻) も更新します。たとえば、スタックに IN_SYNC のドリフトステータスがあることを想定します。CloudFormation はそのスタックに含まれる 1 つ以上のリソースに対してドリフト検出を実行し、CloudFormation はそれらのリソースの 1 つ以上がドリフトしたことを検出します。CloudFormation はスタックドリフトのステータスを DRIFTED に更新します。逆に、ドリフトリソースが 1 つあることでドリフトステータスが DRIFTED になっているスタックがあるとしめます。そのリソースを意図したプロパティ値に戻し、リソースで再度ドリフトを検出した場合、CloudFormation はリソースのドリフトステータスとスタックのドリフトステータスの両方を IN_SYNC に更新するため、スタック全体でドリフトを再度検出する必要はありません。

AWS マネジメントコンソール を使用して個々のリソースの変動を検出するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. スタックのリストから、リソースを含むスタックを選択します。CloudFormation により、そのスタックのスタックの詳細が表示されます。
3. 左のナビゲーションペインの [スタック] で、[スタックアクション] を選択し、[ドリフトの検出] を選択します。
4. [リソースのドリフトステータス] でリソースを選択し、[リソースのドリフトを検出] を選択します。

CloudFormation は選択されたリソースに対してドリフト検出を実行します。成功すると、CloudFormation は必要に応じてリソースのドリフトステータスと全体的なスタックドリフトステータスを更新します。CloudFormation は、ドリフト検出がリソースおよびスタック全体で最後に実行された時刻のタイムスタンプも更新します。リソースが変更されている場合、CloudFormation はリソースの意図した値と現在のプロパティ値に関する詳細なドリフト情報を表示します。

5. リソースのドリフト検出結果を確認します。
 - 変更されたリソースの詳細を表示します。
 - 変更されたリソースを選択した状態で、[View drift details (ドリフトの詳細を表示)] を選択します。

CloudFormation により、リソースのドリフトの詳細が表示され、リソースの意図した値と現在のプロパティ値、およびその 2 つの差異が示されます。

違いを強調表示するには、[違い] セクションでプロパティ名を選択します。

- 追加されたプロパティは [Details] (詳細) セクションの [Current] (最新の) 列で緑色で強調表示されます。
- 削除されたプロパティは [Details] (詳細) セクションの [Expected] (期待される) 列で赤色で強調表示されます。
- 値が変更されたプロパティは、[Expected] 列と [最新の] 列の両方で黄色で強調表示されます。

Differences (4)
< 1 >

✓	Property	Change	Expected value	Current value
✓	AttributeDefinitions.1	ADD	-	{"AttributeName":"stuff","AttributeType":"S"}
✓	ProvisionedThroughput.ReadCapacityUnits	NOT_EQUAL	5	6
✓	ProvisionedThroughput.WriteCapacityUnits	NOT_EQUAL	5	4
✓	Tags	REMOVE	[{"Key":"test","Value":"test"}]	-

Details

Expected

Actual

```

{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    },
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  },
  "Tags": [
    {
      "Key": "test",
      "Value": "test"
    }
  ]
}

```

```

{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    },
    {
      "AttributeName": "stuff",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 6,
    "WriteCapacityUnits": 4
  }
}

```

AWS CLI を使用して個々のリソースの変動を検出するには

•

⚠ Important

古いデータが使用されることのないようにするため、スタックリソースの [前回のドリフトチェック時刻] をチェックして、リソースのドリフト結果に表示されるタイムスタンプより前の時刻になっていることを確認します。

detect-stack-resource-drift を使用して個々のリソースの変動を検出するには AWS CLI コマンドを使用します。リソースの論理 ID とそれが含まれているスタックを指定します。

次の例では、特定のスタックリソースの my-drifted-resource に対してドリフト検出オペレーションが実行されます。レスポンスは、値が変更された 2 つのプロパティに関する詳細を含む、リソースが変更されたことを確認する情報を返します。

```
aws cloudformation detect-stack-resource-drift \
  --stack-name my-stack-with-resource-drift \
  --logical-resource-id my-drifted-resource
```

出力:

```
{
  "StackResourceDrift": {
    "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
    "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds\":120,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\"},\"maxReceiveCount\":12,\"MessageRetentionPeriod\":345600,\"MaximumMessageSize\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHC076H4\"}\",
    "ResourceType": "AWS::SQS::Queue",
    "Timestamp": "2018-03-26T18:54:28.462Z",
    "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-stack-with-resource-drift-Queue-494PBHC076H4",
    "StackResourceDriftStatus": "MODIFIED",
    "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds\":20,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\"},\"maxReceiveCount\":10,\"MessageRetentionPeriod\":345600,\"MaximumMessageSize\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHC076H4\"}\",
    "PropertyDifferences": [
      {
        "PropertyPath": "/DelaySeconds",
        "ActualValue": "120",
        "ExpectedValue": "20",
        "DifferenceType": "NOT_EQUAL"
      }
    ]
  }
}
```

```
    },
    {
      "PropertyPath": "/RedrivePolicy/maxReceiveCount",
      "ActualValue": "12",
      "ExpectedValue": "10",
      "DifferenceType": "NOT_EQUAL"
    }
  ],
  "LogicalResourceId": "my-drifted-resource"
}
```

インポートオペレーションでドリフトを解決する

リソースの構成が意図した構成から逸脱したために、新しい構成を意図した構成として受け入れる必要が生じることがあります。ほとんどの場合は、スタックテンプレート内のリソースの定義を新しい構成で更新することでドリフト結果を解決してから、スタックの更新を実行します。ただし、置換が必要なリソースプロパティが新しい構成によって更新された場合、リソースはスタックの更新中に再作成されます。既存のリソースを保持する場合は、リソースのインポート機能を使用してリソースを更新し、リソースを置き換えることなくドリフト結果を解決できます。

インポートオペレーションによるリソースのドリフトの解決は、次の基本的なステップで行います。

- [Retain に設定されている DeletionPolicy 属性をリソースに追加します](#)。これにより、既存のリソースがスタックから取り除かれたときに、削除されるのではなく、保持されます。
- [テンプレートからリソースを取り除き、スタック更新オペレーションを実行します](#)。これにより、スタックからリソースが取り除かれますが、削除はされません。
- [スタックテンプレートでリソースの実際の状態を記述し、既存のリソースをスタックにインポートし直します](#)。これにより、リソースがスタックに戻され、ドリフト結果を発生させていたプロパティの差異が解決されます。

リソースのインポートについて詳しくは、「[AWS リソースを CloudFormation スタックに手動でインポートする](#)」を参照してください。インポートをサポートしているリソースのリストについては、「[リソースタイプのサポート](#)」を参照してください。

この例では、templateToImport.json という名前の次のテンプレートを使用します。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "BillingMode": "PROVISIONED",
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    },
    "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Games",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ]
      }
    }
  }
}
```


WriteCapacityUnits: 1

この例では、ユーザーが CloudFormation の 外部でリソースを変更したと仮定します。ドリフト検出の実行により、GamesTable の BillingMode が PAY_PER_REQUEST に変更されていることが検出されました。ドリフト検出の詳細については、「[ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#)」を参照してください。

Differences (3)				
Property	Change	Expected value	Current value	
BillingMode	NOT_EQUAL	PROVISIONED	PAY_PER_REQUEST	
ProvisionedThroughput.ReadCapacityUnits	NOT_EQUAL	5	0	
ProvisionedThroughput.WriteCapacityUnits	NOT_EQUAL	1	0	

Expected	Actual
<pre>{ "AttributeDefinitions": [{ "AttributeName": "key", "AttributeType": "S" }], "BillingMode": "PROVISIONED", "KeySchema": [{ "AttributeName": "key", "KeyType": "HASH" }], "ProvisionedThroughput": { "ReadCapacityUnits": 5, "WriteCapacityUnits": 1 }, "TableName": "Games" }</pre>	<pre>{ "AttributeDefinitions": [{ "AttributeName": "key", "AttributeType": "S" }], "BillingMode": "PAY_PER_REQUEST", "KeySchema": [{ "AttributeName": "key", "KeyType": "HASH" }], "ProvisionedThroughput": { "ReadCapacityUnits": 0, "WriteCapacityUnits": 0 }, "TableName": "Games" }</pre>

スタックは古くなっており、リソースはライブですが、意図したリソース構成を保持する必要があります。サービスを中断することなく、インポートオペレーションでドリフトを解決することでこれを達成できます。

CloudFormation コンソールを使用して、インポートオペレーションでドリフトを解決する

ステップ 1. Retain 削除ポリシーでスタックを更新する

DeletionPolicy 属性の **Retain** オプションを使用してスタックを更新するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [Stacks (スタック)] ページで、ドリフトしているスタックを選択します。
3. [Update (更新する)] を選択し、スタックの詳細ペインから [Replace current template (既存テンプレートを置き換える)] を選択します。
4. [Specify template (テンプレートの指定)] ページで、次のいずれかの方法を使用して、Retain オプションが指定されている DeletionPolicy 属性を含む更新済みテンプレートを指定します。

- [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
- [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。

その後、[Next] を選択します。

5. [Specify stack details (スタックの詳細を指定)] ページを確認し、[Next (次へ)] を選択します。
6. [Configure stack options (スタックオプションの設定)] ページを確認し、[Next (次へ)] を選択します。
7. [Review **stack-name** (スタック名の確認)] ページで、[Update stack (スタックの更新)] を選択します。

結果: スタックの [Events (イベント)] ページのステータスは UPDATE_COMPLETE です。

サービスを中断せずにインポートオペレーションでドリフトを解決するには、スタックから取り除くリソースに対して Retain [DeletionPolicy](#) を指定します。次の例では、Retain に設定されている [DeletionPolicy](#) 属性を GamesTable リソースに追加しました。

Example JSON

```
"GamesTable": {
  "Type": "AWS::DynamoDB::Table",
  "DeletionPolicy": "Retain",
  "Properties": {
    "TableName": "Games",
```

Example YAML

```
GamesTable:
  Type: AWS::DynamoDB::Table
  DeletionPolicy: Retain
  Properties:
    TableName: Games
```

ステップ 2. ドリフトしたリソース、関連パラメータ、出力を取り除く

ドリフトしたリソース、関連パラメータ、出力を取り除くには

1. [Update (更新する)] を選択し、スタックの詳細ペインから [Replace current template (既存テンプレートを置き換える)] を選択します。
2. [Specify template (テンプレートの指定)] ページで、次のいずれかの方法を使用して、更新されたテンプレートにリソース、関連パラメータ、およびスタックテンプレートから削除された出力を指定します。
 - [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。

その後、[Next] を選択します。

3. [Specify stack details (スタックの詳細を指定)] ページを確認し、[Next (次へ)] を選択します。
4. [Configure stack options (スタックオプションの設定)] ページを確認し、[Next (次へ)] を選択します。
5. [Review **stack-name** (スタック名の確認)] ページで、[Update stack (スタックの更新)] を選択します。

結果: スタックの [Events] (イベント) ページの [Logical ID] (論理 ID) GamesTable のステータスは DELETE_SKIPPED です。

CloudFormation がスタックの更新オペレーションを完了するまで待ちます。スタックの更新オペレーションが完了したら、リソース、関連パラメータ、および出力をスタックテンプレートから取り除きます。次に、更新されたテンプレートをインポートします。これらのアクションを完了すると、テンプレートの例は次のようになります。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
```

```

        "TableName": "Service",
        "AttributeDefinitions": [
            {
                "AttributeName": "key",
                "AttributeType": "S"
            }
        ],
        "KeySchema": [
            {
                "AttributeName": "key",
                "KeyType": "HASH"
            }
        ],
        "BillingMode": "PROVISIONED",
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 1
        }
    }
}

```

Example YAML

```

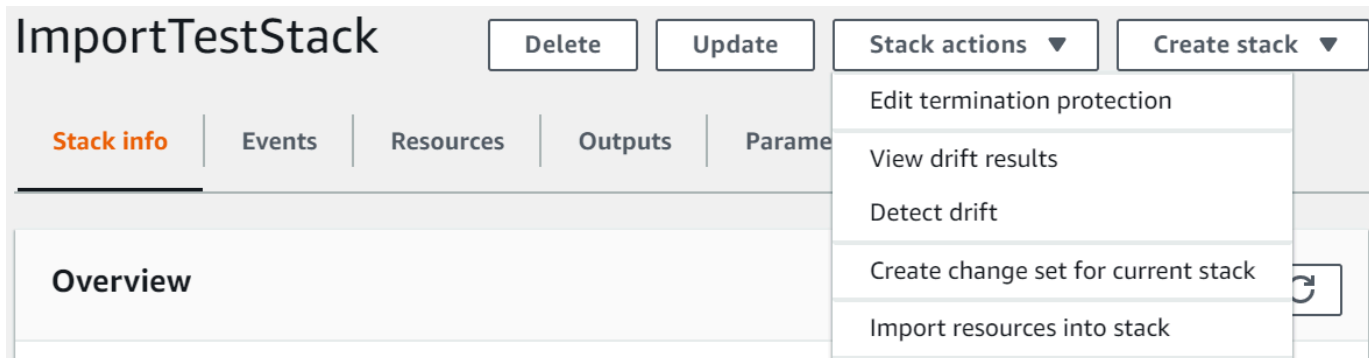
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1

```

ステップ 3. リソースのライブ状態に合わせてテンプレートを更新する

リソースのライブ状態に合わせてテンプレートを更新するには

1. 更新されたテンプレートをインポートするには、[Stack actions (スタックアクション)] を選択し、[Import resources into stack (スタックへのリソースのインポート)] を選択します。



2. このオペレーション中に指定する必要がある項目の一覧に関して [Import overview (概要をインポート)] ページを確認し、[Next (次へ)] を選択します。
3. [Specify template (テンプレートの指定)] ページで、次のいずれかの方法を使用して、更新したテンプレートを指定します。

- [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
- [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。

その後、[Next] を選択します。

4. [Identify resources (リソースの識別)] ページで、各ターゲットリソースを識別します。詳細については、「[リソース識別子](#)」を参照してください。
 - a. [Identifier property (識別子のプロパティ)] で、リソース識別子のタイプを選択します。たとえば、TableName プロパティは `AWS::DynamoDB::Table` リソースを識別します。
 - b. [Identifier value (識別子の値)] に、実際のプロパティ値を入力します。テンプレートの例では、GamesTable リソースの TableName は Games です。
 - c. [次へ] を選択します。
5. [Specify stack details (スタックの詳細を指定)] ページを確認し、[Next (次へ)] を選択します。
6. [Import overview (概要をインポート)] ページで、インポートするリソースを確認し、[Import resources (リソースをインポート)] を選択します。これにより、`AWS::DynamoDB::Table` リソースタイプがスタックにインポートされます。

結果: この例では、サービスを中断することなく、インポートオペレーションによってリソースのドリフトを解決しました。インポートアクションの進行状況は、CloudFormation コンソールの [Events] (イベント) タブで確認できます。インポートされたリソースでは、IMPORT_COMPLETE ステータスに続いて CREATE_COMPLETE ステータスが続き、ステータス理由はリソースのインポート完了になります。

CloudFormation がスタックの更新オペレーションを完了するまで待ちます。スタックの更新オペレーションが完了したら、リソースの実際のドリフト状態と一致するようにテンプレートを更新します。たとえば、BillingMode は PAY_PER_REQUEST に設定され、ReadCapacityUnits と WriteCapacityUnits は 0 に設定されます。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "BillingMode": "PROVISIONED",
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    },
    "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
```

```

        "DeletionPolicy": "Retain",
        "Properties": {
            "TableName": "Games",
            "AttributeDefinitions": [
                {
                    "AttributeName": "key",
                    "AttributeType": "S"
                }
            ],
            "KeySchema": [
                {
                    "AttributeName": "key",
                    "KeyType": "HASH"
                }
            ],
            "BillingMode": "PAY_PER_REQUEST",
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 0,
                "WriteCapacityUnits": 0
            }
        }
    }
}

```

Example YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5

```

```
WriteCapacityUnits: 1
GamesTable:
  Type: AWS::DynamoDB::Table
  DeletionPolicy: Retain
  Properties:
    TableName: Games
    AttributeDefinitions:
      - AttributeName: key
        AttributeType: S
    KeySchema:
      - AttributeName: key
        KeyType: HASH
    BillingMode: PAY_PER_REQUEST
    ProvisionedThroughput:
      ReadCapacityUnits: 0
      WriteCapacityUnits: 0
```

AWS リソースを CloudFormation スタックにインポートする

既存のリソースを CloudFormation スタックにインポートすることができます。この機能は、CloudFormation を使って CloudFormation 以外の場所で作成されたリソースを管理するときに有用です。この場合、リソースを削除したり再度作成したりする必要はありません。

CloudFormation には、既存のリソースをスタックにインポートする方法が 2 つ用意されています。

- [IaC ジェネレーター](#)は、既存のリソースを自動的にスキャンし、リソースの現在の状態に基づいて CloudFormation テンプレートを生成するツールです。作成されたテンプレートは、スタックにこれらのリソースをインポートするために使用できます。
- [リソースインポート](#)は、CloudFormation テンプレートに既存のリソースを記述してから、それらをスタックにインポートする手動プロセスです。このアプローチでは、テンプレートにリソースのプロパティと設定を手動で指定する必要があります。
- [自動インポート](#)は、CloudFormation テンプレート内の既存のリソースを記述する自動プロセスで、CloudFormation は、一致するカスタム名を持つリソースをスタックにインポートします。
- [スタックリファクタリング](#)は、既存のリソースプロパティおよびデータを保持しながら、CloudFormation スタックのリソースの再編成を簡素化する機能です。スタックリファクタリングを使用すると、スタック間でリソースを移動したり、モノリシックスタックを小さなコンポーネントに分割したり、複数のスタックを 1 つに統合したりできます。

リソースインポート機能は、既存のリソースを CloudFormation 管理の対象にすることに加えて、以下のシナリオでも役に立ちます。

- スタック間でのリソースの移動 – 1 つのスタックから別のスタックにリソースをインポートできるため、必要に応じてインフラストラクチャを再編成することが可能になります。
- 既存のスタックのネスト化 – 別のスタック内にネストされたスタックとして既存のスタックをインポートすることで、モジュール形式の再利用可能なインフラストラクチャ設計を実現します。

CloudFormation は幅広いリソースのインポートをサポートしています。詳細については、「[リソースタイプのサポート](#)」を参照してください。

トピック

- [AWS リソースを CloudFormation スタックに手動でインポートする](#)
- [AWS リソースを CloudFormation スタックに自動でインポートする](#)
- [インポートオペレーションを元に戻す](#)

AWS リソースを CloudFormation スタックに手動でインポートする

リソースインポート機能を使用すると、既存の AWS リソースを新規または既存の CloudFormation スタックにインポートすることができます。インポートオペレーション中に、既存のリソースをスタックにインポートするか、既存のリソースから新しいスタックを作成する変更セットを作成します。インポート時に以下を指定します。

- 元のスタックリソースとインポートするリソースの両方を含む、スタック全体を記述するテンプレート。インポートする各リソースには、[DeletionPolicy 属性](#)が必要です。
- CloudFormation がテンプレート内の論理 ID を既存のリソースにマッピングするために使用できる、インポートするリソースの識別子。

Note

CloudFormation は、リソースインポートを使用した 1 レベルのネスト化のみをサポートしています。つまり、子スタックにスタックをインポートしたり、子を持つスタックをインポートしたりすることはできません。

トピック

- [リソース識別子](#)
- [リソースのインポートの検証](#)
- [リソースのインポートのステータスコード](#)
- [インポートオペレーション中の考慮事項](#)
- [その他のリソース](#)
- [既存のリソースからのスタックの作成](#)
- [スタックへの既存リソースのインポート](#)
- [スタック間でのリソースの移動](#)
- [既存のスタックのネスト化](#)

リソース識別子

インポートする各リソースを識別するために 2 つの値を指定します。

- 識別子のプロパティ。これは、各リソースタイプを識別するために使用できるリソースプロパティです。たとえば、AWS::S3::Bucket リソースは、その BucketName を使用して識別できます。

インポートするリソースを識別するために使用するリソースプロパティは、リソースタイプによって異なります。リソースプロパティは CloudFormation コンソールで確認できます。インポートするリソースを含むテンプレートを作成したら、インポートプロセスを開始できます。このプロセスでは、インポートするリソースの識別子プロパティを見つけることができます。一部のリソースタイプでは、識別方法が複数ある場合があり、ドロップダウンリストで使用するプロパティを選択できます。

あるいは、[get-template-summary](#) CLI コマンドを呼び出して、--template-url オプションの値としてスタックテンプレートの S3 URL を指定することにより、インポートするリソースの識別子プロパティを取得することもできます。

- 識別子の値。これは、リソースの実際のプロパティ値です。たとえば、BucketName プロパティの実際の値は MyS3Bucket です。

リソースのサービスコンソールから識別子プロパティの値を取得できます。

リソースのインポートの検証

インポートオペレーション中に、CloudFormation は次の検証を実行します。

- インポートするリソースが存在します。
- インポートする各リソースのプロパティと構成値は、リソースタイプスキーマに準拠します。リソースタイプスキーマでは、受け入れられるプロパティ、必須プロパティ、およびサポートされるプロパティ値が定義されます。
- 必要なプロパティはテンプレートで指定されています。各リソースタイプに必要なプロパティについては、「[AWS リソースおよびプロパティタイプのリファレンス](#)」を参照してください。
- インポートするリソースは、同じリージョン内の別のスタックに属していません。

CloudFormation は、テンプレート構成がリソースプロパティの実際の構成と一致しているかどうかをチェックしません。

Important

予期しない変更を避けるために、テンプレートで定義されているリソースとそのプロパティが、意図したリソースインポートの構成と一致していることを確認します。

リソースのインポートのステータスコード

この表は、リソースインポート機能で使用するさまざまなステータスタイプを説明しています。

インポートオペレーションステータス	説明
IMPORT_IN_PROGRESS	インポートオペレーションは進行中です。
IMPORT_COMPLETE	スタック内のすべてのリソースについて、インポートオペレーションが完了しました。
IMPORT_ROLLBACK_IN_PROGRESS	ロールバックインポートオペレーションは、以前のテンプレート構成をロールバックしています。
IMPORT_ROLLBACK_FAILED	インポートのロールバックオペレーションが失敗しました。
IMPORT_ROLLBACK_COMPLETE	インポートは、以前のテンプレート構成にロールバックされました。

インポートオペレーション中の考慮事項

- インポートが完了した後、後続のスタックオペレーションを実行する前に、インポートされたリソースでドリフト検出を実行することをお勧めします。ドリフト検出により、テンプレート構成が実際の構成と一致することが保証されます。詳細については、「[CloudFormation スタック全体のドリフトを検出する](#)」を参照してください。
- インポートオペレーションでは、新しいリソース作成、リソースの削除、プロパティ構成の変更は許可されません。
- インポートオペレーションを成功させるには、インポートする各リソースに DeletionPolicy 属性が必要です。DeletionPolicy は任意の使用できる値に設定できます。インポートするリソースのみに DeletionPolicy が必要です。スタックにすでに含まれているリソースは、DeletionPolicy を必要としません。
- 同じリソースを複数のスタックにインポートすることはできません。
- cloudformation:ImportResourceTypes IAM ポリシー条件を使用して、インポートオペレーション中にユーザーが操作できるリソースタイプを制御できます。詳細については、「[CloudFormation のポリシー条件キー](#)」を参照してください。
- CloudFormation スタックの制限は、リソースのインポート時に適用されます。制限の詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

その他のリソース

リソースインポートでのスタックドリフトを解決するには、「[インポートオペレーションでドリフトを解決する](#)」を参照してください。

既存のリソースからのスタックの作成

このトピックでは、テンプレートに既存の AWS リソースを記述することで、それらからスタックを作成する方法を説明します。その代わりに、既存のリソースをスキャンして、既存のリソースを CloudFormation にインポートしたり、新しいアカウントでリソースを複製したりするために使用できるテンプレートを自動的に生成するには、「[IaC ジェネレーターを使用して既存のリソースからテンプレートを生成する](#)」を参照してください。

前提条件

始めるには以下のものがが必要です。

- 新しいスタックに含めるすべてのリソースを記述するテンプレート。テンプレートをローカルに保存、または Amazon S3 バケットに保存します。

- インポートするリソースごとに、以下を含めてください。
 - リソースの現在の設定を定義するプロパティとプロパティ値。
 - リソースの固有識別子 (リソース名など)。詳細については、「[リソース識別子](#)」を参照してください。
 - [DeletionPolicy](#) 属性。

トピック

- [サンプルテンプレート](#)
- [AWS マネジメントコンソール を使用した既存のリソースからのスタックの作成](#)
- [AWS CLI を使用した既存のリソースからのスタックの作成](#)

サンプルテンプレート

このチュートリアルでは、CloudFormation 外で作成された 2 つの DynamoDB テーブルを指定する、TemplateToImport.json という名前の以下のサンプルテンプレートを使用していることを前提としています。ServiceTable と GamesTable がインポートの対象です。

Note

これは、テンプレートの一例にすぎません。独自のテスト目的のために使用するには、サンプルリソースをお使いのアカウントのリソースに置き換えてください。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ]
      }
    }
  }
}
```

```

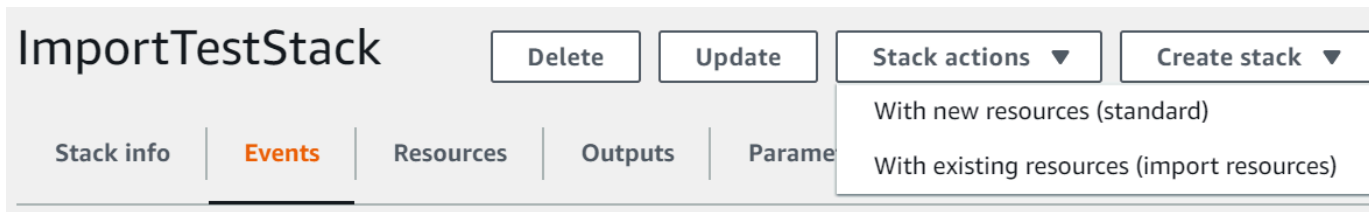
    ],
    "KeySchema": [
      {
        "AttributeName": "key",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 1
    }
  },
  "GamesTable": {
    "Type": "AWS::DynamoDB::Table",
    "DeletionPolicy": "Retain",
    "Properties": {
      "TableName": "Games",
      "AttributeDefinitions": [
        {
          "AttributeName": "key",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "key",
          "KeyType": "HASH"
        }
      ],
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 1
      }
    }
  }
}

```

AWS マネジメントコンソール を使用した既存のリソースからのスタックの作成

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。

2. [スタック] ページで、[スタックを作成] を選択し、[With existing resources (import resources) (既存のリソースを使用 (リソースのインポート))] を選択します。



3. [Import overview (インポートの概要)] ページを参照して、このオペレーション中に指定する必要がある項目の一覧を確認してください。その後、[Next] を選択します。
4. [Specify template (テンプレートを指定)] ページで、以下のいずれかの方法を使用してテンプレートを指定し、[次へ] を選択します。
 - [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。
5. [Identify resources (リソースの識別)] ページで、各ターゲットリソースを識別します。詳細については、「[リソース識別子](#)」を参照してください。
 - a. [Identifier property (識別子のプロパティ)] で、リソース識別子のタイプを選択します。たとえば、AWS::DynamoDB::Table リソースは TableName プロパティを使用して識別できます。
 - b. [Identifier value (識別子の値)] に、実際のプロパティ値を入力します。たとえば、サンプルテンプレートの GamesTable リソースの TableName は **Games** です。
 - c. [次へ] を選択します。
6. [Specify stack details (スタック詳細の指定)] ページで、任意のパラメータを変更し、[次へ] を選択します。これにより、変更セットが自動的に作成されます。

⚠ Important

作成、更新、または削除オペレーションを開始する既存のパラメータを変更すると、インポートオペレーションは失敗します。

7. [Review **stack_name** (stack_name の確認)] ページで、インポートしようとしているリソースが正しいことを確認し、[Import resources (リソースのインポート)] を選択します。これにより、最後のステップで作成した変更セットが自動的に実行されます。

新しいスタックの [Stack details] (スタックの詳細) ページの [Events] (イベント) ペインが表示されます。

ImportTestStack

Delete

Update

Stack actions

Create stack

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Events

Q Search events

Timestamp	Logical ID	Status	Status reason
2019-10-24 15:52:49 UTC-0700	ImportTestStack	IMPORT_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	UPDATE_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2019-10-24 15:52:46 UTC-0700	SampleIamUser	IMPORT_COMPLETE	Resource import completed.
2019-10-24 15:52:45 UTC-0700	SampleIamUser	IMPORT_IN_PROGRESS	Resource import started.
2019-10-24 15:52:43 UTC-0700	ImportTestStack	IMPORT_IN_PROGRESS	User Initiated
2019-10-24 15:52:01 UTC-0700	ImportTestStack	REVIEW_IN_PROGRESS	User Initiated

8. (オプション) スタックでドリフト検出を実行し、インポートされたリソースのテンプレートと実際の構成が一致していることを確認します。ドリフト検出の詳細については、「[CloudFormation スタック全体のドリフトを検出する](#)」を参照してください。
9. (オプション) インポートしたリソースが予想されるテンプレート構成と一致しない場合は、テンプレート構成を修正するか、リソースを直接更新します。このチュートリアルでは、テンプレートの構成を実際の構成に合わせて修正します。
 - a. 影響を受けるリソースの[インポートオペレーションを元に戻します](#)。
 - b. インポートターゲットをテンプレートに再度追加して、テンプレートの構成が実際の構成と一致していることを確認します。
 - c. 変更したテンプレートを使用してステップ 2~8 を繰り返し、リソースを再度インポートします。

AWS CLI を使用した既存のリソースからのスタックの作成

1. テンプレート内の各リソースタイプを識別するプロパティを知るには、テンプレートの S3 URL を指定して `get-template-summary` コマンドを実行します。たとえば、`AWS::DynamoDB::Table` リソースは `TableName` プロパティを使用して識別できます。サンプルテンプレートの `GamesTable` リソースの場合、`TableName` の値は `Games` です。この情報は、次のステップで必要になります。

```
aws cloudformation get-template-summary \
```



```
--template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/  
TemplateToImport.json
```

詳細については、「[リソース識別子](#)」を参照してください。

2. テンプレートから実際のリソースとそれらの一意の識別子のリストを次の JSON 文字列形式で作成します。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifier":{"TableName":"Games"}},  
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifier":{"TableName":"Service"}}]
```

または、設定ファイルで JSON 形式のパラメータを指定することもできます。

例えば、ServiceTable と GamesTable をインポートするには、次の設定を含む *ResourcesToImport.txt* ファイルを作成することが考えられます。

```
[  
  {  
    "ResourceType":"AWS::DynamoDB::Table",  
    "LogicalResourceId":"GamesTable",  
    "ResourceIdentifier":{"  
      "TableName":"Games"  
    }  
  },  
  {  
    "ResourceType":"AWS::DynamoDB::Table",  
    "LogicalResourceId":"ServiceTable",  
    "ResourceIdentifier":{"  
      "TableName":"Service"  
    }  
  }  
]
```

3. 変更セットを作成するには、次の create-change-set コマンドを使用してプレースホルダーテキストを置き換えます。--change-set-type オプションの場合、**IMPORT** の値を指定します。--resources-to-import オプションでは、サンプルの JSON 文字列を、作成した実際の JSON 文字列に置き換えます。

```
aws cloudformation create-change-set \  
  --stack-name TargetStack --change-set-name ImportChangeSet \  
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/  
  --template-name TemplateToImport.json
```

```
--change-set-type IMPORT \
--template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
--resources-to-
import '["ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifi
{"TableName":"Games"}},
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifi
{"TableName":"Service"}]]'
```

Note

--resources-to-import はインライン YAML をサポートしていません。JSON 文字列で引用符をエスケープするための要件は、ターミナルに応じて異なります。詳細については、「AWS Command Line Interface ユーザーガイド」の「[Using quotation marks inside strings](#)」を参照してください。

または、次の例に示すように、ファイル URL を --resources-to-import オプションの入力として使用できます。

```
--resources-to-import file:///ResourcesToImport.txt
```

4. 変更セットを確認し、正しいリソースがインポートされることを確認します。

```
aws cloudformation describe-change-set \
--change-set-name ImportChangeSet --stack-name TargetStack
```

5. 変更セットを開始してリソースをインポートするには、次の execute-change-set コマンドを使用してプレースホルダーテキストを置き換えます。オペレーション (IMPORT_COMPLETE) が正常に完了すると、リソースは正常にインポートされています。

```
aws cloudformation execute-change-set \
--change-set-name ImportChangeSet --stack-name TargetStack
```

6. (オプション) IMPORT_COMPLETE スタックでドリフト検出を実行し、インポートされたリソースのテンプレートと実際の構成が一致していることを確認します。ドリフト検出に関する詳細は、「[個々のスタックリソースのドリフトを検出](#)」を参照してください。
 - a. 指定されたスタックでドリフト検出を実行します。

```
aws cloudformation detect-stack-drift --stack-name TargetStack
```

正常に完了した場合、このコマンドは次のサンプル出力を返します。

```
{ "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
```

- b. 指定されたスタックドリフト検出 ID のドリフト検出オペレーションの進行状況を表示します。

```
aws cloudformation describe-stack-drift-detection-status \  
  --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

- c. 指定されたスタック内でドリフトが確認されたリソースのドリフト情報を表示します。

```
aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
```

7. (オプション) インポートしたリソースが予想されるテンプレート構成と一致しない場合は、テンプレート構成を修正するか、リソースを直接更新します。このチュートリアルでは、テンプレートの構成を実際の構成に合わせて修正します。
- a. 影響を受けるリソースの [インポートオペレーションを元に戻します](#)。
- b. インポートターゲットをテンプレートに再度追加して、テンプレートの構成が実際の構成と一致していることを確認します。
- c. 変更したテンプレートを使用してステップ 3~6 を繰り返し、リソースを再度インポートします。

スタックへの既存リソースのインポート

このトピックでは、テンプレートに既存の AWS リソースを記述することで、それらを既存のスタックにインポートする方法を説明します。その代わりに、既存のリソースをスキャンして、既存のリソースを CloudFormation にインポートしたり、新しいアカウントでリソースを複製したりするために使用できるテンプレートを自動的に生成するには、[「IaC ジェネレーターを使用して既存のリソースからテンプレートを生成する」](#)を参照してください。

前提条件

始めるには以下のものがが必要です。

- スタックにすでに含まれているリソースとインポートするリソースの両方を含む、スタック全体を記述するテンプレート。テンプレートをローカルに保存、または Amazon S3 バケットに保存します。

実行中スタックのテンプレートのコピーを取得する

1. クラウドフォーメーション コンソール の <https://console.aws.amazon.com/cloudformation/> 開きます：
 2. スタックのリストから、テンプレートの取得元となるスタックを選択します。
 3. [スタックの詳細] ペインで [テンプレート] タブを選択してから、[クリップボードにコピー] を選択します。
 4. コードをテキストエディタに貼り付けて、テンプレートへのその他リソースの追加を開始します。
- インポートするリソースごとに、以下を含めてください。
 - リソースの現在の設定を定義するプロパティとプロパティ値。
 - リソースの固有識別子 (リソース名など)。詳細については、「[リソース識別子](#)」を参照してください。
 - [DeletionPolicy](#) 属性。

トピック

- [サンプルテンプレート](#)
- [AWS マネジメントコンソール を使用した既存のリソースのスタックへのインポート](#)
- [AWS CLI を使用した既存のリソースのスタックへのインポート](#)

サンプルテンプレート

このチュートリアルでは、2 つの DynamoDB テーブルを指定する、TemplateToImport.json という名前の以下のサンプルテンプレートを使用していることを前提としています。ServiceTable は現在スタックに含まれているテーブルで、GamesTable はインポートするテーブルです。

Note

これは、テンプレートの一例にすぎません。独自のテスト目的のために使用するには、サンプルリソースをお使いのアカウントのリソースに置き換えてください。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    },
    "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
        "TableName": "Games",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ]
      }
    }
  }
}
```

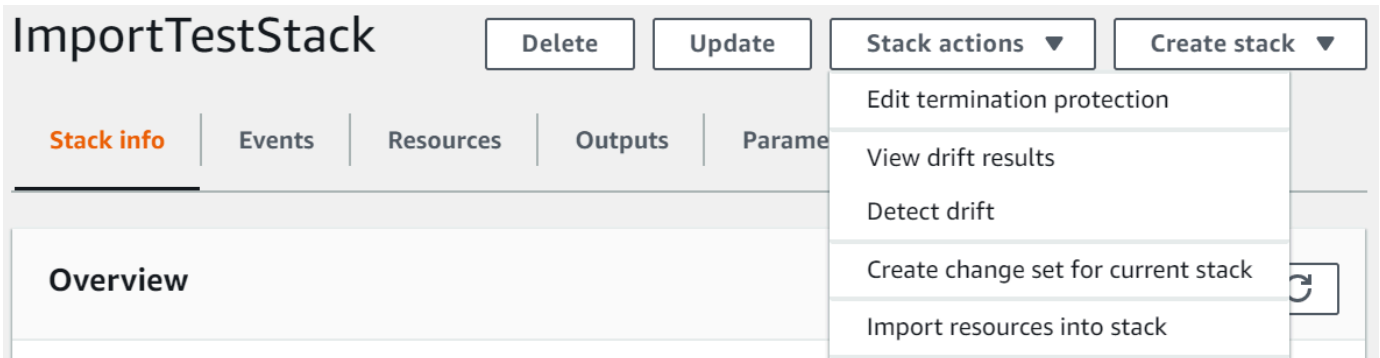
```
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 1
        }
    }
}
```

AWS マネジメントコンソール を使用した既存のリソースのスタックへのインポート

Note

CloudFormation コンソールは、リソースをインポートする場合に、組み込み関数である `Fn::Transform` の使用をサポートしません。AWS CLI を使用して関数 `Fn::Transform` を使用するリソースをインポートできます。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. [スタック] ページで、リソースのインポート先のスタックを選択します。
3. [Stack actions (スタックアクション)] を選択し、[Import resources into stack (リソースをスタックにインポートする)] を選択します。



4. [インポートの概要] ページを確認し、[次へ] を選択します。
5. [Specify template (テンプレートを指定)] ページで、以下のいずれかの方法を使用して更新したテンプレートを指定し、[次へ] を選択します。
 - [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。

6. [Identify resources (リソースの識別)] ページで、各ターゲットリソースを識別します。詳細については、「[リソース識別子](#)」を参照してください。
 - a. [Identifier property (識別子のプロパティ)] で、リソース識別子のタイプを選択します。たとえば、AWS::DynamoDB::Table リソースは TableName プロパティを使用して識別できます。
 - b. [Identifier value (識別子の値)] に、実際のプロパティ値を入力します。たとえば、サンプルテンプレートの GamesTable リソースの TableName は **Games** です。
 - c. [次へ] を選択します。
7. [Specify stack details (スタック詳細の指定)] ページで、任意のパラメータを更新し、[次へ] を選択します。これにより、変更セットが自動的に作成されます。

Note

作成、更新、または削除オペレーションを開始する既存のパラメータを変更すると、インポートオペレーションは失敗します。

8. [Review **stack-name** (スタック名の確認)] ページで、インポートするリソースを確認し、[Import resources (リソースのインポート)] を選択します。これにより、最後のステップで作成した変更セットが自動的に実行されます。この時点で、インポートされたリソースにはスタックレベルのタグが適用されます。詳細については、「[スタックオプションを設定する](#)」を参照してください。

スタックの [イベント] ページが表示されます。

ImportTestStack

Delete

Update

Stack actions

Create stack

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Events

Search events

Timestamp	Logical ID	Status	Status reason
2019-10-24 15:52:49 UTC-0700	ImportTestStack	IMPORT_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SamplelamUser	UPDATE_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SamplelamUser	UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2019-10-24 15:52:46 UTC-0700	SamplelamUser	IMPORT_COMPLETE	Resource import completed.
2019-10-24 15:52:45 UTC-0700	SamplelamUser	IMPORT_IN_PROGRESS	Resource import started.
2019-10-24 15:52:43 UTC-0700	ImportTestStack	IMPORT_IN_PROGRESS	User Initiated
2019-10-24 15:52:01 UTC-0700	ImportTestStack	REVIEW_IN_PROGRESS	User Initiated

9. (オプション) スタックでドリフト検出を実行し、インポートされたリソースのテンプレートと実際の構成が一致していることを確認します。ドリフト検出の詳細については、「[CloudFormation スタック全体のドリフトを検出する](#)」を参照してください。
10. (オプション) インポートしたリソースが予想されるテンプレート構成と一致しない場合は、テンプレート構成を修正するか、リソースを直接更新します。ドリフトしたリソースのインポートに関する詳細は、「[インポートオペレーションでドリフトを解決する](#)」を参照してください。

AWS CLI を使用した既存のリソースのスタックへのインポート

1. テンプレート内の各リソースタイプを識別するプロパティを知るには、テンプレートの S3 URL を指定して `get-template-summary` コマンドを実行します。たとえば、`AWS::DynamoDB::Table` リソースは `TableName` プロパティを使用して識別できます。サンプルテンプレートの `GamesTable` リソースの場合、`TableName` の値は `Games` です。この情報は、次のステップで必要になります。

```
aws cloudformation get-template-summary \  
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/  
TemplateToImport.json
```

詳細については、「[リソース識別子](#)」を参照してください。

2. インポートする実際のリソースとその一意の識別子のリストを次の JSON 文字列形式で作成します。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifier":{"TableName":"Games"}}]
```

または、設定ファイルで JSON 形式のパラメータを指定することもできます。

例えば、`GamesTable` をインポートするには、次の設定を含む *ResourcesToImport.txt* ファイルを作成します。

```
[  
  {  
    "ResourceType":"AWS::DynamoDB::Table",  
    "LogicalResourceId":"GamesTable",  
    "ResourceIdentifier": {  
      "TableName":"Games"  
    }  
  }  
]
```



```
}
]
```

3. 変更セットを作成するには、次の `create-change-set` コマンドを使用してプレースホルダーテキストを置き換えます。`--change-set-type` オプションの場合、**IMPORT** の値を指定します。`--resources-to-import` オプションでは、サンプルの JSON 文字列を、作成した実際の JSON 文字列に置き換えます。

```
aws cloudformation create-change-set \
  --stack-name TargetStack --change-set-name ImportChangeSet \
  --change-set-type IMPORT \
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
  --resources-to-
import '[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceProperties":{
  {"TableName":"Games"}}]'
```

Note

`--resources-to-import` はインライン YAML をサポートしていません。JSON 文字列で引用符をエスケープするための要件は、ターミナルに応じて異なります。詳細については、「AWS Command Line Interface ユーザーガイド」の「[Using quotation marks inside strings](#)」を参照してください。

または、次の例に示すように、ファイル URL を `--resources-to-import` オプションの入力として使用できます。

```
--resources-to-import file:///ResourcesToImport.txt
```

4. 変更セットを確認し、正しいリソースがインポートされることを確認します。

```
aws cloudformation describe-change-set \
  --change-set-name ImportChangeSet --stack-name TargetStack
```

5. 変更セットを開始してリソースをインポートするには、次の `execute-change-set` コマンドを使用してプレースホルダーテキストを置き換えます。この時点で、インポートされたリソースにはスタックレベルのタグが適用されます。詳細については、「[スタックオプションを設定する](#)」を

参照してください。オペレーション (IMPORT_COMPLETE) が正常に完了すると、リソースは正常にインポートされています。

```
aws cloudformation execute-change-set \  
  --change-set-name ImportChangeSet --stack-name TargetStack
```

6. (オプション) IMPORT_COMPLETE スタックでドリフト検出を実行し、インポートされたリソースのテンプレートと実際の構成が一致していることを確認します。ドリフト検出の詳細については、「[CloudFormation スタック全体のドリフトを検出する](#)」を参照してください。

- a. 指定されたスタックでドリフト検出を実行します。

```
aws cloudformation detect-stack-drift --stack-name TargetStack
```

正常に完了した場合、このコマンドは次のサンプル出力を返します。

```
{ "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
```

- b. 指定されたスタックドリフト検出 ID のドリフト検出オペレーションの進行状況を表示します。

```
aws cloudformation describe-stack-drift-detection-status \  
  --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

- c. 指定されたスタック内でドリフトが確認されたリソースのドリフト情報を表示します。

```
aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
```

7. (オプション) インポートしたリソースが予想されるテンプレート構成と一致しない場合は、テンプレート構成を修正するか、リソースを直接更新します。ドリフトしたリソースのインポートに関する詳細は、「[インポートオペレーションでドリフトを解決する](#)」を参照してください。

スタック間でのリソースの移動

resource import 機能を使用すると、スタック間でリソースを移動、またはリファクタリングできます。まず、移動するリソースに Retain 削除ポリシーを追加して、ソーススタックからリソースを削除してターゲットスタックにインポートするときにリソースが保持されるようにする必要があります。

インポートを初めて行う場合は、まず [AWS リソースを CloudFormation スタックにインポートする](#) トピックの概要情報を確認することをお勧めします。

⚠ Important

すべてのリソースがインポートオペレーションをサポートしているわけではありません。スタックからリソースを削除する前に、「[インポートオペレーションをサポートするリソース](#)」を参照してください。インポートオペレーションをサポートしていないリソースをスタックから削除する場合、そのリソースを別のスタックにインポートしたり、ソーススタックに戻したりすることはできません。

AWS マネジメントコンソール を使用したスタックのリファクタリング

1. 移動元テンプレートで、移動するリソースに Retain [DeletionPolicy](#) を指定します。

次のサンプルソーステンプレートでは、Games がこのリファクタリングの対象です。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
```

```
        "WriteCapacityUnits":1
      }
    },
    "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
        "TableName": "Games",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    }
  }
}
```

2. CloudFormation コンソールを開き、スタック更新を実行して削除ポリシーを適用します。
 - a. [スタック] ページで、スタックを選択した状態で [更新] を選択します。
 - b. [テンプレートを準備] で、[現在のテンプレートを置換] を選択します。
 - c. [Specify template (テンプレートを指定)] の下にある GamesTable で、DeletionPolicy 属性により更新されたソーステンプレートを指定し、[次へ] を選択します。
 - [Amazon S3 URL] を選択し、テキストボックスで更新されたソーステンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、更新されたソーステンプレートファイルを参照します。

- d. [Specify stack details (スタック詳細の指定)] ページでは、変更の必要はありません。[次へ] を選択します。
 - e. [スタックオプションの設定] ページでは、変更の必要はありません。[次へ] を選択します。
 - f. [**SourceStackName**を確認] ページで、変更内容を確認します。テンプレートに IAM リソースが含まれる場合は、[I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。テンプレート内の IAM リソースの使用の詳細については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。次に、変更セットを作成してソーススタックを更新するか、ソーススタックを直接更新します。
3. ソーステンプレートからリソース、関連パラメータ、および出力を削除し、それらをターゲットテンプレートに追加します。

ソーステンプレートは次のようになります。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    }
  }
}
```

```
    }  
  }  
}  
}
```

次のターゲットテンプレートの例には、現在 PlayersTable リソースがあり、さらに GamesTable も含まれています。

Example JSON

```
{  
  "AWSTemplateFormatVersion": "2010-09-09",  
  "Description": "Import test",  
  "Resources": {  
    "PlayersTable": {  
      "Type": "AWS::DynamoDB::Table",  
      "Properties": {  
        "TableName": "Players",  
        "AttributeDefinitions": [  
          {  
            "AttributeName": "key",  
            "AttributeType": "S"  
          }  
        ],  
        "KeySchema": [  
          {  
            "AttributeName": "key",  
            "KeyType": "HASH"  
          }  
        ],  
        "ProvisionedThroughput": {  
          "ReadCapacityUnits": 5,  
          "WriteCapacityUnits": 1  
        }  
      }  
    },  
    "GamesTable": {  
      "Type": "AWS::DynamoDB::Table",  
      "DeletionPolicy": "Retain",  
      "Properties": {  
        "TableName": "Games",  
        "AttributeDefinitions": [  

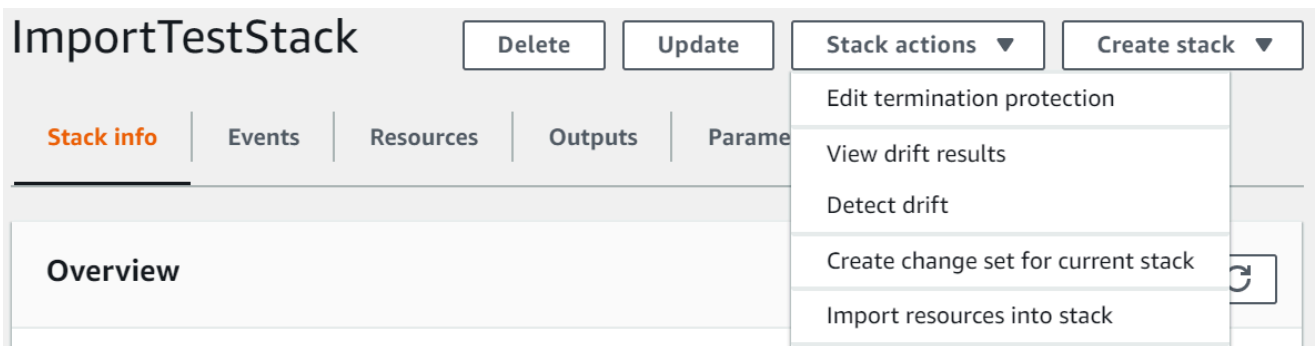
```

```

        {
            "AttributeName": "key",
            "AttributeType": "S"
        }
    ],
    "KeySchema": [
        {
            "AttributeName": "key",
            "KeyType": "HASH"
        }
    ],
    "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 1
    }
}
}
}
}

```

4. ステップ 2~3 を繰り返してソーススタックを再度更新します。今回は、スタックからターゲットリソースを削除します。
5. インポートオペレーションを実行して、GamesTable をターゲットスタックに追加します。
 - a. [スタック] ページで、親スタックを選択した状態で [Stack actions (スタックアクション)] を選択し、[Import resources into stack (リソースをスタックにインポートする)] を選択します。



- b. [Import overview (インポートの概要)] ページを参照して、このオペレーション中に指定する必要がある項目の一覧を確認してください。[次へ] を選択します。
 - c. [Specify template] (テンプレートを指定) ページで、以下のいずれかの操作を実行し、[Next] (次へ) を選択します。

- [Amazon S3 URL] を選択し、テキストボックスで URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、アップロードするファイルを参照します。
- d. [Identify resources (リソースの識別)] ページで、移動するリソース (この例では GamesTable) を特定します。詳細については、「[リソース識別子](#)」を参照してください。
1. [Identifier property (識別子のプロパティ)] で、リソース識別子のタイプを選択します。例えば、AWS::DynamoDB::Table リソースは TableName プロパティを使用して識別できます。
 2. [Identifier value (識別子の値)] に、実際のプロパティ値を入力します。例えば、*GamesTables*。
 3. [次へ] を選択します。
- e. [Specify stack details (スタック詳細の指定)] ページで、任意のパラメータを変更し、[次へ] を選択します。これにより、変更セットが自動的に作成されます。

 Important

作成、更新、または削除オペレーションを開始する既存のパラメータを変更すると、インポートオペレーションは失敗します。

- f. [*TargetStackName* を確認] ページで、インポートしようとしているリソースが正しいことを確認し、[リソースをインポート] を選択します。これにより、最後のステップで作成した変更セットが自動的に開始されます。この時点で、インポートされたリソースには[スタックレベルのタグ](#)が適用されます。
- g. 親スタックの [Stack details] (スタックの詳細) ページの [Events] (イベント) ペインが表示されます。

ImportTestStack

Delete

Update

Stack actions

Create stack

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Events

Search events

Timestamp	Logical ID	Status	Status reason
2019-10-24 15:52:49 UTC-0700	ImportTestStack	IMPORT_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	UPDATE_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2019-10-24 15:52:46 UTC-0700	SampleIamUser	IMPORT_COMPLETE	Resource import completed.
2019-10-24 15:52:45 UTC-0700	SampleIamUser	IMPORT_IN_PROGRESS	Resource import started.
2019-10-24 15:52:43 UTC-0700	ImportTestStack	IMPORT_IN_PROGRESS	User Initiated
2019-10-24 15:52:01 UTC-0700	ImportTestStack	REVIEW_IN_PROGRESS	User Initiated

Note

AWS::CloudFormation::Stack リソースは既に CloudFormation によって管理されているため、このインポートオペレーションの後、親スタックでドリフト検出を実行する必要はありません。

AWS CLI を使用したスタックのリファクタリング

1. 移動元テンプレートで、移動するリソースに Retain [DeletionPolicy](#) を指定します。

次のサンプルソーステンプレートでは、GameTable がこのリファクタリングの対象です。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ]
      }
    }
  }
}
```

```

    ],
    "KeySchema": [
      {
        "AttributeName": "key",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 1
    }
  },
  "GamesTable": {
    "Type": "AWS::DynamoDB::Table",
    "DeletionPolicy": "Retain",
    "Properties": {
      "TableName": "Games",
      "AttributeDefinitions": [
        {
          "AttributeName": "key",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "key",
          "KeyType": "HASH"
        }
      ],
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 1
      }
    }
  }
}

```

2. ソーススタックを更新して、削除ポリシーをリソースに適用します。

```
aws cloudformation update-stack --stack-name SourceStackName
```

3. ソーステンプレートからリソース、関連パラメータ、および出力を削除し、それらをターゲットテンプレートに追加します。

ソーステンプレートは次のようになります。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    }
  }
}
```

次のターゲットテンプレートの例には、現在 `PlayersTable` リソースがあり、さらに `GamesTable` も含まれています。

Example JSON

```
{
```

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "Import test",
"Resources": {
  "PlayersTable": {
    "Type": "AWS::DynamoDB::Table",
    "Properties": {
      "TableName": "Players",
      "AttributeDefinitions": [
        {
          "AttributeName": "key",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "key",
          "KeyType": "HASH"
        }
      ],
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 1
      }
    }
  },
  "GamesTable": {
    "Type": "AWS::DynamoDB::Table",
    "DeletionPolicy": "Retain",
    "Properties": {
      "TableName": "Games",
      "AttributeDefinitions": [
        {
          "AttributeName": "key",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "key",
          "KeyType": "HASH"
        }
      ],
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
```

```

        "WriteCapacityUnits": 1
      }
    }
  }
}

```

4. ソーススタックを更新して、GamesTable リソースとそれに関連するパラメータと出力をスタックから削除します。

```
aws cloudformation update-stack --stack-name SourceStackName
```

5. インポートする実際のリソースとその一意の識別子のリストを次の JSON 文字列形式で作成します。詳細については、「[リソース識別子](#)」を参照してください。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifier":{"TableName":"Games"}}]
```

または、設定ファイルで JSON 形式のパラメータを指定することもできます。

例えば、GamesTable をインポートするには、次の設定を含む *ResourcesToImport.txt* ファイルを作成します。

```
[
  {
    "ResourceType":"AWS::DynamoDB::Table",
    "LogicalResourceId":"GamesTable",
    "ResourceIdentifier": {
      "TableName":"Games"
    }
  }
]
```

6. 変更セットを作成するには、次の create-change-set コマンドを使用してプレースホルダーテキストを置き換えます。--change-set-type オプションの場合、**IMPORT** の値を指定します。--resources-to-import オプションでは、サンプルの JSON 文字列を、作成した実際の JSON 文字列に置き換えます。

```
aws cloudformation create-change-set \
  --stack-name TargetStackName --change-set-name ImportChangeSet \
  --change-set-type IMPORT \
```

```
--template-body file://TemplateToImport.json \  
--resources-to-import  
" '[{"ResourceType": "AWS::DynamoDB::Table", "LogicalResourceId": "GamesTable", "ResourceIdentifiers": [{"TableName": "Games"}]}] '"
```

Note

--resources-to-import はインライン YAML をサポートしていません。JSON 文字列で引用符をエスケープするための要件は、ターミナルに応じて異なります。詳細については、「AWS Command Line Interface ユーザーガイド」の「[Using quotation marks inside strings](#)」を参照してください。

あるいは、次の例に示すように、--resources-to-import オプションの入力としてファイル URL を使用することもできます。

```
--resources-to-import file://ResourcesToImport.txt
```

7. 変更セットを確認し、正しいリソースがターゲットスタックにインポートされることを確認します。

```
aws cloudformation describe-change-set \  
--change-set-name ImportChangeSet
```

8. 変更セットを開始してリソースをインポートするには、次の execute-change-set コマンドを使用してプレースホルダーテキストを置き換えます。この時点で、インポートされたリソースにはスタックレベルのタグが適用されます。オペレーション (IMPORT_COMPLETE) が正常に完了すると、リソースは正常にインポートされています。

```
aws cloudformation execute-change-set \  
--change-set-name ImportChangeSet --stack-name TargetStackName
```

Note

リソースは既に CloudFormation によって管理されているため、このインポートオペレーションの後、ターゲットスタックでドリフト検出を実行する必要はありません。

既存のスタックのネスト化

resource import 機能を使用して、既存のスタックを別の既存のスタック内にネスト化します。ネスト化されたスタックは、他のテンプレート内から宣言および参照する一般的なコンポーネントです。これにより、同じ構成をテンプレートにコピーして貼り付けるのを避け、スタックの更新を簡素化できます。共通コンポーネントのテンプレートがある場合は、AWS::CloudFormation::Stack リソースを使用して、別のテンプレート内からこのテンプレートを参照できます。ネストされたスタックの詳細については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。

CloudFormation は、resource import を使用して 1 つのレベルのネスト化のみをサポートします。つまり、子スタックにスタックをインポートしたり、子を持つスタックをインポートしたりすることはできません。

インポートを初めて行う場合は、まず [AWS リソースを CloudFormation スタックに手動でインポートする](#) トピックの概要情報を確認することをお勧めします。

ネストされたスタックのインポートの検証

ネストされたスタックのインポートオペレーション中に、CloudFormation は以下の検証を実行します。

- 親スタックテンプレートのネスト化された AWS::CloudFormation::Stack 定義は、実際のネスト化されたスタックのテンプレートと一致します。
- 親スタックテンプレートのネスト化された AWS::CloudFormation::Stack 定義のタグは、実際のネスト化されたスタックリソースのタグと一致します。

AWS マネジメントコンソール を使用した既存のスタックのネスト化

1. Retain [DeletionPolicy](#) を使用して、AWS::CloudFormation::Stack リソースを親スタックテンプレートに追加します。次の親スタックテンプレート例では、MyNestedStack がインポートのターゲットです。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
      "Type":"AWS::DynamoDB::Table",
```

```

    "Properties":{
      "TableName":"Service",
      "AttributeDefinitions":[
        {
          "AttributeName":"key",
          "AttributeType":"S"
        }
      ],
      "KeySchema":[
        {
          "AttributeName":"key",
          "KeyType":"HASH"
        }
      ],
      "ProvisionedThroughput":{
        "ReadCapacityUnits":5,
        "WriteCapacityUnits":1
      }
    },
    "MyNestedStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "DeletionPolicy": "Retain",
      "Properties" : {
        "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t1.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service

```

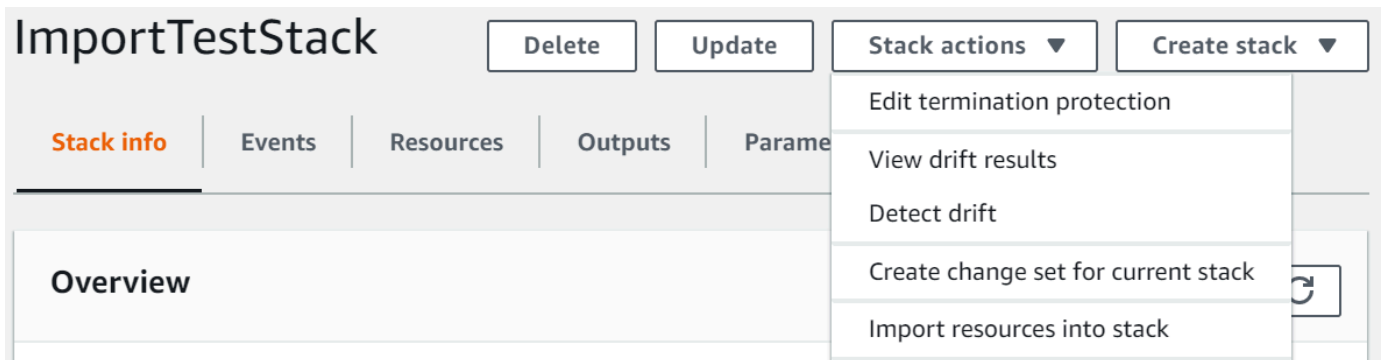


```

AttributeDefinitions:
  - AttributeName: key
    AttributeType: S
KeySchema:
  - AttributeName: key
    KeyType: HASH
ProvisionedThroughput:
  ReadCapacityUnits: 5
  WriteCapacityUnits: 1
MyNestedStack:
  Type: AWS::CloudFormation::Stack
  DeletionPolicy: Retain
  Properties:
    TemplateURL: >-
      https://s3.amazonaws.com/cloudformation-templates-us-east-2/
      EC2ChooseAMI.template
    Parameters:
      InstanceType: t1.micro
      KeyName: mykey

```

2. CloudFormation コンソールを開きます。
3. [スタック] ページで、親スタックを選択した状態で [Stack actions (スタックアクション)] を選択し、[Import resources into stack (リソースをスタックにインポートする)] を選択します。



4. [Import overview (インポートの概要)] ページを参照して、このオペレーション中に指定する必要がある項目の一覧を確認してください。その後、[Next] を選択します。
5. [Specify template (テンプレートを指定)] ページで、以下のいずれかの方法を使用して更新した親テンプレートを指定し、[次へ] を選択します。
 - [Amazon S3 URL] を選択し、テキストボックスでテンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、テンプレートを参照します。

6. [Identify resources (リソースの識別)] ページで、AWS::CloudFormation::Stack リソースを識別します。
 - a. [Identifier property (識別子のプロパティ)] で、リソース識別子のタイプを選択します。例えば、AWS::CloudFormation::Stack リソースは StackId プロパティを使用して識別できます。
 - b. [識別子の値] で、インポートするスタックの ARN を入力します。例えば、`arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10`。

Resources to import (1)

The following logical IDs are new to your template. Provide the identifier values for the resource you are importing.

NestedStack
AWS::CloudFormation::Stack

Identifier property	Identifier value
StackId ▼	Enter StackId

- c. [次へ] を選択します。
 7. [Specify stack details (スタック詳細の指定)] ページで、任意のパラメータを変更し、[次へ] を選択します。これにより、変更セットが自動的に作成されます。
- ⚠ Important**

作成、更新、または削除オペレーションを開始する既存のパラメータを変更すると、インポートオペレーションは失敗します。
8. [**MyParentStack**を確認] ページで、インポートしようとしているリソースが正しいことを確認し、[リソースをインポート] を選択します。これにより、最後のステップで作成した変更セットが自動的に実行されます。この時点で、インポートされたリソースにはスタックレベルのタグが適用されます。
 9. 親スタックの [Stack details] (スタックの詳細) ページの [Events] (イベント) ペインが表示されます。

ImportTestStack

Delete

Update

Stack actions

Create stack

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Events

🔍 Search events

🔄

⚙️

Timestamp	Logical ID	Status	Status reason
2019-10-24 15:52:49 UTC-0700	ImportTestStack	🟢 IMPORT_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	🟢 UPDATE_COMPLETE	-
2019-10-24 15:52:47 UTC-0700	SampleIamUser	🔄 UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2019-10-24 15:52:46 UTC-0700	SampleIamUser	🟢 IMPORT_COMPLETE	Resource import completed.
2019-10-24 15:52:45 UTC-0700	SampleIamUser	🔄 IMPORT_IN_PROGRESS	Resource import started.
2019-10-24 15:52:43 UTC-0700	ImportTestStack	🔄 IMPORT_IN_PROGRESS	User Initiated
2019-10-24 15:52:01 UTC-0700	ImportTestStack	🔄 REVIEW_IN_PROGRESS	User Initiated

Note

AWS::CloudFormation::Stack リソースはすでに CloudFormation によって管理されていたため、このインポートオペレーションの後、親スタックでドリフト検出を実行する必要はありません。

AWS CLI を使用した既存のスタックのネスト化

1. Retain [DeletionPolicy](#) を使用して、AWS::CloudFormation::Stack リソースを親スタックテンプレートに追加します。次の親テンプレート例では、MyNestedStack がインポートのターゲットです。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
      "Type":"AWS::DynamoDB::Table",
      "Properties":{
        "TableName":"Service",
        "AttributeDefinitions":[
          {
            "AttributeName":"key",
            "AttributeType":"S"
          }
        ]
      }
    }
  }
}
```

```

    ],
    "KeySchema": [
      {
        "AttributeName": "key",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 1
    }
  },
  "MyNestedStack" : {
    "Type" : "AWS::CloudFormation::Stack",
    "DeletionPolicy": "Retain",
    "Properties" : {
      "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template",
      "Parameters" : {
        "InstanceType" : "t1.micro",
        "KeyName" : "mykey"
      }
    }
  }
}
}
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      ProvisionedThroughput:

```

```

    ReadCapacityUnits: 5
    WriteCapacityUnits: 1
  MyNestedStack:
    Type: AWS::CloudFormation::Stack
    DeletionPolicy: Retain
    Properties:
      TemplateURL: >-
        https://s3.amazonaws.com/cloudformation-templates-us-east-2/
        EC2ChooseAMI.template
      Parameters:
        InstanceType: t1.micro
        KeyName: mykey

```

2. 次の例に示すように、次の変更を加えて JSON 文字列を作成します。

- *MyNestedStack* を、テンプレートで指定されているターゲットリソースの論理 ID に置き換えます。
- *arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10* を、インポートするスタックの ARN に置き換えます。

```

[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack","Resource
{"StackId":"arn:aws:cloudformation:us-east-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}]}]

```

または、設定ファイルでパラメータを指定することもできます。

例えば、MyNestedStack をインポートするには、次の設定を含む *ResourcesToImport.txt* ファイルを作成することが考えられます。

JSON

```

[
  {
    "ResourceType":"AWS::CloudFormation::Stack",
    "LogicalResourceId":"MyNestedStack",
    "ResourceIdentifier": {
      "StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/
mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10"
    }
  }
]

```

]

YAML

```

ResourceType: AWS::CloudFormation::Stack
  LogicalResourceId: MyNestedStack
  ResourceIdentifier:
    StackId: >-
      arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10

```

3. 変更セットを作成するには、次の `create-change-set` コマンドを使用してプレースホルダーテキストを置き換えます。--change-set-type オプションの場合、**IMPORT** の値を指定します。--resources-to-import オプションでは、サンプルの JSON 文字列を、作成した実際の JSON 文字列に置き換えます。

```

aws cloudformation create-change-set \
  --stack-name MyParentStack --change-set-name ImportChangeSet \
  --change-set-type IMPORT \
  --template-body file://TemplateToImport.json \
  --resources-to-
import '[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack",
{"StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}}]'

```

 Note

--resources-to-import はインライン YAML をサポートしていません。JSON 文字列で引用符をエスケープするための要件は、ターミナルに応じて異なります。詳細については、「AWS Command Line Interface ユーザーガイド」の「[Using quotation marks inside strings](#)」を参照してください。

または、次の例に示すように、ファイル URL を --resources-to-import オプションの入力として使用できます。

```
--resources-to-import file://ResourcesToImport.txt
```

正常に完了した場合、このコマンドは次のサンプル出力を返します。

```
{
  "Id": "arn:aws:cloudformation:us-west-2:12345678910:changeSet/
  ImportChangeSet/8ad75b3f-665f-46f6-a200-0b4727a9442e",
  "StackId": "arn:aws:cloudformation:us-west-2:12345678910:stack/
  MyParentStack/4e345b70-1281-11ef-b027-027366d8e82b"
}
```

4. 変更セットを確認し、正しいスタックがインポートされることを確認します。

```
aws cloudformation describe-change-set --change-set-name ImportChangeSet
```

5. 変更セットを開始し、スタックをソースの親スタックにインポートするには、次の `execute-change-set` コマンドを使用してプレースホルダーテキストを置き換えます。この時点で、インポートされたリソースには[スタックレベルのタグ](#)が適用されます。インポートオペレーション (IMPORT_COMPLETE) が正常に完了すると、スタックは正常にネストされています。

```
aws cloudformation execute-change-set --change-set-name ImportChangeSet
```

Note

AWS::CloudFormation::Stack リソースはすでに CloudFormation によって管理されているため、このインポートオペレーションの後、親スタックでドリフト検出を実行する必要はありません。

AWS リソースを CloudFormation スタックに自動でインポートする

CloudFormation スタックを作成または更新するときに、名前付きリソースを自動的にインポートできるようになりました。名前付きリソースとはカスタム名を持つリソースのことです。詳細については、「CloudFormation テンプレートリファレンス」の「[Name タイプ](#)」を参照してください。

自動インポートを開始すると、CloudFormation がテンプレートに一致する既存のリソースをチェックし、それらをデプロイ中にインポートします。ネストされたスタックの場合は、ルートスタックから変更セットを作成します。

インポートが完了した後、後続のスタックオペレーションを実行する前に、インポートされたリソースでドリフト検出を実行することをお勧めします。ドリフト検出により、テンプレート構成が実際の

構成と一致することが保証されます。詳細については、「[CloudFormation スタック全体のドリフトを検出する](#)」を参照してください。

リソースをインポートするときは、次の要件を満たす必要があります。

- リソースに、テンプレートで定義した静的カスタム名が付いていること。動的な名前 (!Ref または他の関数を使用) は現在サポートされていません。
- リソースに Retain または RetainExceptOnCreate の DeletionPolicy があること。
- リソースが別の CloudFormation スタックに属していないこと。
- リソースタイプが CloudFormation のインポートオペレーションをサポートしていること。詳細については、「[リソースタイプのサポート](#)」を参照してください。
- リソースタイプのプライマリ ID はテンプレートにある必要があります。読み取り専用プロパティを持つプライマリ ID はサポートされていません。タイプに対するプライマリ ID を確認するには、リソーススキーマで primaryIdentifier プロパティを探します。プロパティの詳細については、「[primaryIdentifier](#)」を参照してください。

Example 自動インポートの例

次の例では、変更セット CreateChangeSet を使用して、テンプレートファイル template.yaml に基づいて my-stack という名前のスタックを作成し、一致するリソースを自動的にインポートします。

```
aws cloudformation create-change-set \  
  --stack-name my-stack \  
  --change-set-name CreateChangeSet \  
  --change-set-type CREATE \  
  --template-body file://template.yaml \  
  --import-existing-resources
```

トラブルシューティング

自動インポートが失敗した場合は、以下を実行してトラブルシューティングを行います。

- テンプレートのリソース名がリソースの名前と正確に一致していることを確認する。
- リソースがまだ別のスタックで管理されていないことを確認する。
- リソースタイプがインポートオペレーションをサポートしていることを確認する。
- リソースタイプに必要なすべてのプロパティがテンプレートに含まれていることを確認する。

インポートオペレーションを元に戻す

インポートオペレーションを元に戻すには、テンプレートから削除するリソースの Retain 削除ポリシーを指定して、スタックから削除するときにそのリソースが保持されるようにします。

AWS マネジメントコンソール を使用してインポートオペレーションを元に戻す

1. スタックから削除するリソースの Retain [DeletionPolicy](#) を指定します。次のテンプレート例では、GameTable はこの元に戻すオペレーションのターゲットです。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    },
    "GameTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
        "TableName": "Games",
```

```

        "AttributeDefinitions": [
            {
                "AttributeName": "key",
                "AttributeType": "S"
            }
        ],
        "KeySchema": [
            {
                "AttributeName": "key",
                "KeyType": "HASH"
            }
        ],
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 1
        }
    }
}

```

2. CloudFormation コンソールを開き、スタック更新を実行して削除ポリシーを適用します。
 - a. [スタック] ページで、スタックを選択した状態で [更新] を選択し、[Update stack (standard) (スタックを更新 (標準))] を選択します。
 - b. [テンプレートを準備] で、[現在のテンプレートを置換] を選択します。
 - c. [Specify template (テンプレートを指定)] の下にある GamesTable で、DeletionPolicy 属性により更新されたソーステンプレートを指定し、[次へ] を選択します。
 - [Amazon S3 URL] を選択し、テキストボックスで更新されたソーステンプレートの URL を指定します。
 - [Upload a template file (テンプレートファイルのアップロード)] を選択し、更新されたソーステンプレートファイルを参照します。
 - d. [Specify stack details (スタック詳細の指定)] ページでは、変更の必要はありません。[次へ] を選択します。
 - e. [スタックオプションの設定] ページでは、変更の必要はありません。[次へ] を選択します。
 - f. **[MyStack を確認]** ページで、変更内容を確認します。テンプレートに IAM リソースが含まれる場合は、[I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレート](#)

[レートでの IAM リソースの承認](#)」を参照してください。次に、変更セットを作成してソーススタックを更新するか、ソーススタックを直接更新します。

3. スタックテンプレートからリソース、関連するパラメータ、および出力を削除します。この例では、テンプレートは次のようになります。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    }
  }
}
```

4. ステップ 2 を繰り返して、リソース (GamesTable) とそれに関連するパラメータと出力をスタックから削除します。

AWS CLI を使用してインポートオペレーションを元に戻す

1. スタックから削除するリソースの Retain [DeletionPolicy](#) を指定します。次のテンプレート例では、GameTable はこの元に戻すオペレーションのターゲットです。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 5,
          "WriteCapacityUnits": 1
        }
      }
    },
    "GameTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
        "TableName": "Games",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ]
      }
    }
  }
}
```

```
    ],
    "KeySchema": [
      {
        "AttributeName": "key",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 1
    }
  }
}
```

2. スタックを更新して、削除ポリシーをリソースに適用します。

```
aws cloudformation update-stack --stack-name MyStack
```

3. スタックテンプレートからリソース、関連するパラメータ、および出力を削除します。この例では、テンプレートは次のようになります。

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Import test",
  "Resources": {
    "ServiceTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "TableName": "Service",
        "AttributeDefinitions": [
          {
            "AttributeName": "key",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "key",
            "KeyType": "HASH"
          }
        ]
      }
    }
  }
}
```

```
        }
      ],
      "ProvisionedThroughput":{
        "ReadCapacityUnits":5,
        "WriteCapacityUnits":1
      }
    }
  }
}
```

4. スタックを更新して、リソース (GamesTable) とそれに関連するパラメータと出力をスタックから削除します。

```
aws cloudformation update-stack --stack-name MyStack
```

スタックリファクタリング

スタックリファクタリングを使用すると、既存のリソースプロパティとデータを維持しながら CloudFormation スタック内のリソースを再編成できます。スタック間でのリソースの移動、大きなスタックの小さなスタックへの分割、または複数スタックの1つのスタックへの統合が可能です。

トピック

- [スタックリファクタリングの仕組み](#)
- [スタックリファクタリングに関する考慮事項](#)
- [前提条件](#)
- [スタックをリファクタリングする \(コンソール\)](#)
- [スタックをリファクタリングする \(AWS CLI\)](#)
- [リソースの制限](#)

スタックリファクタリングの仕組み

スタックリファクタリングには以下のフェーズがあります。

1. 現在のインフラストラクチャを評価する – 既存の CloudFormation スタックとリソースを見直して、スタックリファクタリングの機会を特定します。

2. リファクタリングを計画する – リソースの編成方法を定義します。依存関係、命名規則、運用上の制限を検討してください。後に CloudFormation の検証に影響する可能性があります。
3. リファクタリング先スタックを決定する – リソースをリファクタリングするスタックを決定します。リソースは、最小で 2 つのスタック (コンソールを使用する)、最大で 5 つのスタック (AWS CLI を使用する) 間で移動できます。リソースはネストされたスタック間で移動できます。
4. テンプレートを更新する – CloudFormation テンプレートを変更して、テンプレート間でのリソース定義の移動などの計画済みの変更を反映します。論理 ID の名前はこのプロセス中に変更できます。
5. スタックリファクタリングを作成する – リファクタリングするスタック名とテンプレートのリストを指定します。
6. リファクタリングによる影響を確認し、競合があれば解決する – CloudFormation が指定されたテンプレートを検証し、スタック間の依存関係、タグの更新問題があるリソースタイプ、リソース論理 ID の競合をチェックします。

検証が成功した場合は、リファクタリングの実行中に行われるリファクタリングアクションのプレビューを CloudFormation が生成します。

検証に失敗した場合は、特定された問題を解決して再試行します。競合が存在する場合は、競合するリソースのソースとリファクタリング先を示すリソース論理 ID マッピングを指定します。

7. リファクタリングを実行する – 変更がリファクタリングの目標と一致することを確認したら、スタックのリファクタリングを完了します。
8. 監視する – 実行ステータスを追跡して、オペレーションが正常に完了することを確認します。

スタックリファクタリングに関する考慮事項

スタックをリファクタリングするときは、次の点に留意してください。

- スタックリファクタリングは、既存のリソースを再編成すること限定されます。リファクタリング中に、リソースの作成や削除、リソース設定の変更、または新しいパラメータ、条件、マッピングの変更や追加などを行うことはできません。これらの変更を行うには、まずスタックを更新してから、スタックのリファクタリングを実施します。
- 同じリソースを複数のスタックにリファクタリングすることはできません。
- `AWS::StackName` など、ソーススタックとリファクタリング先スタックで値が異なる擬似パラメータを参照するリソースをリファクタリングすることはできません。

- CloudFormation は空のスタックをサポートしません。リファクタリングによってスタックがリソースのない状態になる場合は、まずそのスタックに少なくとも 1 つのリソースを追加してから [create-stack-refactor](#) を実行する必要があります。これは、AWS::SNS::Topic や AWS::CloudFormation::WaitCondition といったシンプルなリソースにすることができます。例えば、次のようになります。

```
Resources:
  MySimpleSNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: MySimpleTopic
```

- スタックリファクタリングは、スタックポリシーが何を許可し、拒否するかにかかわらず、スタックポリシーがアタッチされているスタックをサポートしません。

前提条件

スタックをリファクタリングするには、改訂されたテンプレートを既に作成している必要があります。

[get-template](#) コマンドを使用して、リファクタリングするスタックの CloudFormation テンプレートを取得します。

```
aws cloudformation get-template --stack-name Stack1
```

テンプレートを取得したら、希望の統合開発環境 (IDE) を使用してテンプレートを更新し、目的の構造とリソース組織を使用するようにします。

スタックをリファクタリングする (コンソール)

次の手順を使用してスタックをリファクタリングするには、コンソールを使用します。

スタックをリファクタリングするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックがある場所の AWS リージョン を選択します。
3. 左側のナビゲーションペインで、[スタックのリファクタリング] を選択します。

4. [スタックのリファクタリング] ページで、[スタックのリファクタリングを開始する] を選択します。
5. [説明] では、スタックのリファクタリングを識別するのに役立つ説明を入力します。その後、[Next] を選択します。
6. [スタック 1] で、以下を行います。

- a. [既存のスタックのテンプレートを更新する] または [新しいスタックを作成する] を選択します。

[既存のスタックのテンプレートを更新する] を選択した場合は、リストから既存のスタックを選択します。または、[スタック ARN を入力する] を選択して、既存のスタックの ARN を入力します。

[新しいスタックを作成する] を選択した場合は、[スタック名] で新しいスタックの名前を指定します。

- b. [既存のテンプレートをリファクタリングされたテンプレートに置き換える] で、[Amazon S3 URL] または [テンプレートファイルをアップロードする] を選択して、スタック 1 で必要なテンプレートをアップロードします。
- c. [次へ] を選択します。

7. [スタック 2] で、以下を行います。

- a. [既存のスタックのテンプレートを更新する] または [新しいスタックを作成する] を選択します。

[既存のスタックのテンプレートを更新する] を選択した場合は、リストから既存のスタックを選択します。または、[スタック ARN を入力する] を選択して、既存のスタックの ARN を入力します。

[新しいスタックを作成する] を選択した場合は、[スタック名] で新しいスタックの名前を指定します。

- b. 既存のテンプレートをリファクタリングされたテンプレートに置き換える] で、[Amazon S3 URL] または [テンプレートファイルをアップロードする] を選択して、スタック 2 で必要なテンプレートをアップロードします。
- c. [次へ] を選択します。

8. [論理リソース ID の名前変更を指定する] ページで、表示されたリソースを正しい論理 ID にマッピングすることで、CloudFormation がスタックのリファクタリング方法を知っていることを確認します。スタックリファクタリングの一部として、リソースの論理 ID が変更された場合は、

ソーススタック名、元の論理 ID、送信先スタック名、および名前が変更された論理 ID を指定して、名前の変更方法を指定する必要があります。場合によっては、CloudFormation コンソールによってリソースマッピングが自動的に検出され、先に進む前に、事前に入力されたリソースマッピングが正しいことを確認できます。

9. [次へ] を選択します。
10. [確認して実行する] ページで、前のステップで選択したすべての内容を確認し、すべてが正しくセットアップされていることを確認します。
11. スタックをリファクタリングする準備ができたなら、[スタックリファクタリングを実行] を選択します。

スタックをリファクタリングする (AWS CLI)

スタックリファクタリングの AWS CLI コマンドには以下が含まれます。

- 計画された変更を検証してプレビューを生成するための [create-stack-refactor](#)。
- スタックリファクタリングオペレーションのステータスと詳細を取得するための [describe-stack-refactor](#)。
- 検証済みのスタックリファクタリングオペレーションを完了するための [execute-stack-refactor](#)。
- アカウント内のすべてのスタックリファクタリングオペレーションを、現在のステータスと基本情報とともに一覧表示するための [list-stack-refactors](#)。
- リファクタリングの実行中に CloudFormation が各スタックとリソースに対して実行する特定のアクションのプレビューを表示するための [list-stack-refactor-actions](#)。

AWS CLI を使用してスタックをリファクタリングするには、以下の手順を実行します。

スタックをリファクタリングするには

1. [create-stack-refactor](#) コマンドを使用して、リファクタリングするスタックのスタック名と更新されたテンプレートを指定します。新しいスタックがまだ存在しない場合は、CloudFormation が新しいスタックを作成できるようにする `--enable-stack-creation` オプションを含めます。

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file:///template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file:///template2-updated.yaml \
```

```
--enable-stack-creation
```

このコマンドは、今後のステップで使用する StackRefactorId を返します。

```
{
  "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841"
}
```

テンプレートの検証中に競合が検出された場合は (次のステップで確認できます)、[create-stack-refactor](#) コマンドを --resource-mappings オプションとともに使用します。

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file://template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file://template2-updated.yaml \
  --enable-stack-creation \
  --resource-mappings file://resource-mapping.json
```

次は、resource-mapping.json ファイルの例です。

```
[
  {
    "Source": {
      "StackName": "Stack1",
      "LogicalResourceId": "MySNSTopic"
    },
    "Destination": {
      "StackName": "Stack2",
      "LogicalResourceId": "MyLambdaSNSTopic"
    }
  }
]
```

2. [describe-stack-refactor](#) コマンドを使用して、Status が CREATE_COMPLETE であることを確認します。このコマンドは、検証が完了していることを実証します。

```
aws cloudformation describe-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

出力例:

```
{
  "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",
  "StackIds": [
    "arn:aws:cloudformation:us-east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
    "arn:aws:cloudformation:us-east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
  ],
  "ExecutionStatus": "AVAILABLE",
  "Status": "CREATE_COMPLETE"
}
```

3. [list-stack-refactor-actions](#) コマンドを使用して、実行される特定のアクションをプレビューします。

```
aws cloudformation list-stack-refactor-actions \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

出力例:

```
{
  "StackRefactorActions": [
    {
      "Action": "MOVE",
      "Entity": "RESOURCE",
      "PhysicalResourceId": "MyTestLambdaRole",
      "Description": "No configuration changes detected.",
      "Detection": "AUTO",
      "TagResources": [],
      "UntagResources": [],
      "ResourceMapping": {
        "Source": {
          "StackName": "arn:aws:cloudformation:us-east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
          "LogicalResourceId": "MyLambdaRole"
        },
        "Destination": {
          "StackName": "arn:aws:cloudformation:us-east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
          "LogicalResourceId": "MyLambdaRole"
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "Action": "MOVE",
    "Entity": "RESOURCE",
    "PhysicalResourceId": "MyTestFunction",
    "Description": "Resource configuration changes will be validated during
refactor execution.",
    "Detection": "AUTO",
    "TagResources": [
      {
        "Key": "aws:cloudformation:stack-name",
        "Value": "Stack2"
      },
      {
        "Key": "aws:cloudformation:logical-id",
        "Value": "MyFunction"
      },
      {
        "Key": "aws:cloudformation:stack-id",
        "Value": "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
      }
    ],
    "UntagResources": [
      "aws:cloudformation:stack-name",
      "aws:cloudformation:logical-id",
      "aws:cloudformation:stack-id"
    ],
    "ResourceMapping": {
      "Source": {
        "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
        "LogicalResourceId": "MyFunction"
      },
      "Destination": {
        "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
        "LogicalResourceId": "MyFunction"
      }
    }
  }
]

```

```
}
```

4. 変更を確認して確認したら、[execute-stack-refactor](#) コマンドを使用してスタックリファクタリングオペレーションを完了します。

```
aws cloudformation execute-stack-refactor \  
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

5. [describe-stack-refactor](#) コマンドを使用して、実行ステータスを監視します。

```
aws cloudformation describe-stack-refactor \  
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

出力例:

```
{  
  "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",  
  "StackIds": [  
    "arn:aws:cloudformation:us-east-1:123456789012:stack/  
Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",  
    "arn:aws:cloudformation:us-east-1:123456789012:stack/  
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"  
  ],  
  "ExecutionStatus": "SUCCEEDED",  
  "Status": "COMPLETE"  
}
```

リソースの制限

- スタックリファクタリングがサポートするのは、`provisioningType` が `FULLY_MUTABLE` のリソースタイプのみです。これは [describe-type](#) コマンドを使用してチェックできます。
- CloudFormation はリファクタリングの作成時にリソースの適格性を検証し、サポートされていないリソースがあれば [describe-stack-refactor](#) コマンドの出力で報告します。
- 以下のリソースはスタックリファクタリングでサポートされていません。
 - `AWS::ACMPCA::Certificate`
 - `AWS::ACMPCA::CertificateAuthority`
 - `AWS::ACMPCA::CertificateAuthorityActivation`
 - `AWS::ApiGateway::BasePathMapping`

- `AWS::ApiGateway::Method`
- `AWS::AppConfig::ConfigurationProfile`
- `AWS::AppConfig::Deployment`
- `AWS::AppConfig::Environment`
- `AWS::AppConfig::Extension`
- `AWS::AppConfig::ExtensionAssociation`
- `AWS::AppStream::DirectoryConfig`
- `AWS::AppStream::StackFleetAssociation`
- `AWS::AppStream::StackUserAssociation`
- `AWS::AppStream::User`
- `AWS::BackupGateway::Hypervisor`
- `AWS::CertificateManager::Certificate`
- `AWS::CloudFormation::CustomResource`
- `AWS::CloudFormation::Macro`
- `AWS::CloudFormation::WaitCondition`
- `AWS::CloudFormation::WaitConditionHandle`
- `AWS::CodeDeploy::DeploymentGroup`
- `AWS::CodePipeline::CustomActionType`
- `AWS::Cognito::UserPoolRiskConfigurationAttachment`
- `AWS::Cognito::UserPoolUICustomizationAttachment`
- `AWS::Cognito::UserPoolUserToGroupAttachment`
- `AWS::Config::ConfigRule`
- `AWS::Config::ConfigurationRecorder`
- `AWS::Config::DeliveryChannel`
- `AWS::DataBrew::Dataset`
- `AWS::DataBrew::Job`
- `AWS::DataBrew::Project`
- `AWS::DataBrew::Recipe`
- `AWS::DataBrew::Ruleset`
- `AWS::DataBrew::Schedule`

- AWS::DataZone::DataSource
- AWS::DataZone::Environment
- AWS::DataZone::EnvironmentBlueprintConfiguration
- AWS::DataZone::EnvironmentProfile
- AWS::DataZone::Project
- AWS::DataZone::SubscriptionTarget
- AWS::DirectoryService::MicrosoftAD
- AWS::DynamoDB::GlobalTable
- AWS::EC2::CustomerGateway
- AWS::EC2::EIP
- AWS::EC2::LaunchTemplate
- AWS::EC2::NetworkInterfacePermission
- AWS::EC2::PlacementGroup
- AWS::EC2::SpotFleet
- AWS::EC2::VPCDHCPOptionsAssociation
- AWS::EC2::VolumeAttachment
- AWS::EMR::Cluster
- AWS::EMR::InstanceFleetConfig
- AWS::EMR::InstanceGroupConfig
- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::ReplicationGroup
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache::SecurityGroupIngress
- AWS::ElasticBeanstalk::ConfigurationTemplate
- AWS::ElasticLoadBalancing::LoadBalancer
- AWS::ElasticLoadBalancingV2::ListenerCertificate
- AWS::Elasticsearch::Domain
- AWS::FIS::ExperimentTemplate
- AWS::Glue::Schema
- AWS::GuardDuty::IPSet

- AWS::GuardDuty::PublishingDestination
- AWS::GuardDuty::ThreatIntelSet
- AWS::IAM::AccessKey
- AWS::IAM::UserToGroupAddition
- AWS::ImageBuilder::Component
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoTFleetWise::Campaign
- AWS::IoTWireless::WirelessDeviceImportTask
- AWS::Lambda::EventInvokeConfig
- AWS::Lex::BotVersion
- AWS::M2::Application
- AWS::MSK::Configuration
- AWS::MSK::ServerlessCluster
- AWS::Maester::DocumentType
- AWS::MediaTailor::Channel
- AWS::NeptuneGraph::PrivateGraphEndpoint
- AWS::Omics::AnnotationStore
- AWS::Omics::ReferenceStore
- AWS::Omics::SequenceStore
- AWS::OpenSearchServerless::Collection
- AWS::OpsWorks::App
- AWS::OpsWorks::ElasticLoadBalancerAttachment
- AWS::OpsWorks::Instance
- AWS::OpsWorks::Layer
- AWS::OpsWorks::Stack
- AWS::OpsWorks::UserProfile
- AWS::OpsWorks::Volume
- AWS::PCAConnectorAD::Connector
- AWS::PCAConnectorAD::DirectoryRegistration

- `AWS::PCAConectorAD::Template`
- `AWS::PCAConectorAD::TemplateGroupAccessControlEntry`
- `AWS::Panorama::PackageVersion`
- `AWS::QuickSight::Theme`
- `AWS::RDS::DBSecurityGroup`
- `AWS::RDS::DBSecurityGroupIngress`
- `AWS::Redshift::ClusterSecurityGroup`
- `AWS::Redshift::ClusterSecurityGroupIngress`
- `AWS::RefactorSpaces::Environment`
- `AWS::RefactorSpaces::Route`
- `AWS::RefactorSpaces::Service`
- `AWS::RoboMaker::RobotApplication`
- `AWS::RoboMaker::SimulationApplication`
- `AWS::Route53::RecordSet`
- `AWS::Route53::RecordSetGroup`
- `AWS::SDB::Domain`
- `AWS::SageMaker::InferenceComponen`
- `AWS::ServiceCatalog::PortfolioPrincipalAssociation`
- `AWS::ServiceCatalog::PortfolioProductAssociation`
- `AWS::ServiceCatalog::PortfolioShare`
- `AWS::ServiceCatalog::TagOptionAssociation`
- `AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation`
- `AWS::ServiceCatalogAppRegistry::ResourceAssociation`
- `AWS::StepFunctions::StateMachineVersion`
- `AWS::Synthetics::Canary`
- `AWS::VoiceID::Domain`
- `AWS::WAF::ByteMatchSet`
- `AWS::WAF::IPSet`
- `AWS::WAF::Rule`
- `AWS::WAF::SizeConstraintSet`

- `AWS::WAF::SqlInjectionMatchSet`
- `AWS::WAF::WebACL`
- `AWS::WAF::XssMatchSet`
- `AWS::WAFv2::IPSet`
- `AWS::WAFv2::RegexPatternSet`
- `AWS::WAFv2::RuleGroup`
- `AWS::WAFv2::WebACL`
- `AWS::WorkSpaces::Workspace`

リソースタイプのサポート

以下のテーブルは、インポート、ドリフト検出、Infrastructure as Code (IaC) ジェネレーターの各オペレーションを現在サポートしている AWS リソースタイプを一覧表示したものです。各リソースタイプ名は、「[CloudFormation テンプレートリファレンスガイド](#)」の対応するリファレンストピックにリンクしています。

リソースのインポートをサポートするリソースは、自動インポートもサポート可能です。詳細については、「[AWS リソースを CloudFormation スタックにインポートする](#)」を参照してください。

このリストはすべての AWS リソースタイプを網羅していません。以下に表示されていないリソースタイプがある場合は、AWS クラウドコントロール API からアクセスできない可能性があります。詳細については、「Cloud Control API ユーザーガイド」の「[Resource types that support Cloud Control API](#)」を参照してください。Cloud Control API からどのリソースタイプにアクセスできるかは、AWS サービスごとに個別に決められています。

また、CloudFormation は、プロビジョニング可能なプライベートリソースタイプ (プロビジョニングタイプが `FULLY_MUTABLE` または `IMMUTABLE`) のインポートオペレーションおよびドリフト検出オペレーションもサポートしています。プライベートリソースタイプのリソースでインポートまたはドリフト検出を実行するには、まず、お使いのアカウントのリソースタイプのデフォルトバージョンを登録し、プロビジョニング可能にする必要があります。詳細については、「[共有されているサードパーティーのプライベート拡張機能を使用する](#)」を参照してください。

IaC ジェネレーターがサポートしているのは、ご使用のリージョンの Cloud Control API でサポートされている AWS リソースのみであることに注意してください。

プロビジョニング可能なパブリックおよびプライベートのリソースタイプに関する情報にプログラムでアクセスするときは、AWS クラウドコントロール API を使用します。詳細については、「Cloud

Control API ユーザーガイド」の「[リソースタイプが Cloud Control API をサポートしているかどうかの判断](#)」を参照してください。

インポート、ドリフト検出、または IaC ジェネレーターの使用を開始する際は、こちらの役立つピックをご覧ください。

- [AWS リソースを CloudFormation スタックにインポートする](#)
- [ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#)
- [IaC ジェネレーターを使用して既存のリソースからテンプレートを生成する](#)

リソース	[Import] (インポート)	ドリフト検出	IaC ジェネレーター
AWS::ACMPCA::Certificate	 はい	 いいえ	 いいえ
AWS::ACMPCA::CertificateAuthority	 はい	 はい	 はい
AWS::ACMPCA::CertificateAuthorityActivation	 はい	 いいえ	 いいえ
AWS::ACMPCA::Permission	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AIOps::InvestigationGroup</u>	 はい	 はい	 いいえ
<u>AWS::APS::AnomalyDetector</u>	 はい	 はい	 いいえ
<u>AWS::APS::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::APS::RuleGroupsNamespace</u>	 はい	 はい	 はい
<u>AWS::APS::Scraper</u>	 はい	 はい	 いいえ
<u>AWS::APS::Workspace</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ARCRegionSwitch::Plan</u>	 はい	 はい	 いいえ
<u>AWS::ARCZonalShift::AutoshiftObserverNotificationStatus</u>	 はい	 はい	 いいえ
<u>AWS::ARCZonalShift::ZonalAutoshiftConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::AccessAnalyzer::Analyzer</u>	 はい	 はい	 はい
<u>AWS::AmazonMQ::Broker</u>	 はい	 はい	 いいえ
<u>AWS::AmazonMQ::Configuration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Amplify::App</u>	 はい	 はい	 はい
<u>AWS::Amplify::Branch</u>	 はい	 はい	 はい
<u>AWS::Amplify::Domain</u>	 はい	 はい	 はい
<u>AWS::AmplifyUIBuilder::Component</u>	 はい	 はい	 はい
<u>AWS::AmplifyUIBuilder::Form</u>	 はい	 はい	 はい
<u>AWS::AmplifyUIBuilder::Theme</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGateway::Account</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::ApiKey</u>	 はい	 はい	 はい
<u>AWS::ApiGateway::Authorizer</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::BasePathMapping</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::BasePathMappingV2</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::ClientCertificate</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGateway::Deployment</u>	 はい	 はい	 はい
<u>AWS::ApiGateway::DocumentationPart</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::DocumentationVersion</u>	 はい	 はい	 はい
<u>AWS::ApiGateway::DomainName</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::DomainNameAccessAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::DomainNameV2</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGateway::GatewayResponse</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::Method</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::Model</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::RequestValidator</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::Resource</u>	 はい	 はい	 いいえ
<u>AWS::ApiGateway::RestApi</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGateway::Stage</u>	 はい	 いいえ	 はい
<u>AWS::ApiGateway::UsagePlan</u>	 はい	 はい	 はい
<u>AWS::ApiGateway::UsagePlanKey</u>	 はい	 いいえ	 はい
<u>AWS::ApiGateway::VpcLink</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::Api</u>	 はい	 はい	 はい
<u>AWS::ApiGatewayV2::ApiMapping</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGatewayV2::Authorizer</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::Deployment</u>	 はい	 はい	 はい
<u>AWS::ApiGatewayV2::DomainName</u>	 はい	 はい	 はい
<u>AWS::ApiGatewayV2::Integration</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::IntegrationResponse</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::Model</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApiGatewayV2::Route</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::RouteResponse</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::RoutingRule</u>	 はい	 はい	 いいえ
<u>AWS::ApiGatewayV2::VpcLink</u>	 はい	 はい	 はい
<u>AWS::AppConfig::Application</u>	 はい	 はい	 はい
<u>AWS::AppConfig::ConfigurationProfile</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	IaC ジェネレーター
<u>AWS::AppConfig::Deployment</u>	 はい	 はい	 いいえ
<u>AWS::AppConfig::DeploymentStrategy</u>	 はい	 はい	 いいえ
<u>AWS::AppConfig::Environment</u>	 はい	 はい	 いいえ
<u>AWS::AppConfig::Extension</u>	 はい	 はい	 いいえ
<u>AWS::AppConfig::ExtensionAssociation</u>	 はい	 はい	 いいえ
<u>AWS::AppConfig::HostedConfigurationVersion</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::AppFlow::Connector	 はい	 はい	 はい
AWS::AppFlow::ConnectorProfile	 はい	 はい	 はい
AWS::AppFlow::Flow	 はい	 はい	 はい
AWS::AppIntegrations::Application	 はい	 はい	 いいえ
AWS::AppIntegrations::DataIntegration	 はい	 はい	 はい
AWS::AppIntegrations::EventIntegration	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AppRunner::AutoScalingConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::AppRunner::ObservabilityConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::AppRunner::Service</u>	 はい	 はい	 いいえ
<u>AWS::AppRunner::VpcConnector</u>	 はい	 はい	 いいえ
<u>AWS::AppRunner::VpcIngressConnection</u>	 はい	 はい	 いいえ
<u>AWS::AppStream::AppBlock</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AppStream::AppBlockBuilder</u>	 はい	 はい	 いいえ
<u>AWS::AppStream::Application</u>	 はい	 はい	 いいえ
<u>AWS::AppStream::ApplicationEntitlementAssociation</u>	 はい	 はい	 いいえ
<u>AWS::AppStream::ApplicationFleetAssociation</u>	 はい	 はい	 いいえ
<u>AWS::AppStream::DirectoryConfig</u>	 はい	 はい	 はい
<u>AWS::AppStream::Entitlement</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AppStream::ImageBuilder</u>	 はい	 はい	 はい
<u>AWS::AppSync::Api</u>	 はい	 はい	 いいえ
<u>AWS::AppSync::ChannelNamespace</u>	 はい	 はい	 いいえ
<u>AWS::AppSync::DataSource</u>	 はい	 はい	 いいえ
<u>AWS::AppSync::DomainName</u>	 はい	 はい	 はい
<u>AWS::AppSync::DomainNameApiAssociation</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AppSync::FunctionConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::AppSync::GraphQLApi</u>	 いいえ	 はい	 いいえ
<u>AWS::AppSync::Resolver</u>	 はい	 はい	 いいえ
<u>AWS::AppSync::SourceApiAssociation</u>	 はい	 はい	 はい
<u>AWS::AppTest::TestCase</u>	 はい	 はい	 いいえ
<u>AWS::ApplicationAutoScaling::ScalableTarget</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ApplicationAutoScaling::ScalingPolicy</u>	 はい	 いいえ	 いいえ
<u>AWS::ApplicationInsights::Application</u>	 はい	 はい	 いいえ
<u>AWS::ApplicationSignals::Discovery</u>	 はい	 はい	 いいえ
<u>AWS::ApplicationSignals::GroupingConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::ApplicationSignals::ServiceLevelObjective</u>	 はい	 はい	 いいえ
<u>AWS::Athena::CapacityReservation</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Athena::DataCatalog</u>	 はい	 はい	 はい
<u>AWS::Athena::NamedQuery</u>	 はい	 はい	 はい
<u>AWS::Athena::PreparedStatement</u>	 はい	 はい	 はい
<u>AWS::Athena::WorkGroup</u>	 はい	 はい	 はい
<u>AWS::AuditManager::Assessment</u>	 はい	 いいえ	 いいえ
<u>AWS::AutoScaling::AutoScalingGroup</u>	 はい	 いいえ	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::AutoScaling::LaunchConfiguration</u>	 はい	 はい	 はい
<u>AWS::AutoScaling::LifecycleHook</u>	 はい	 はい	 はい
<u>AWS::AutoScaling::ScalingPolicy</u>	 はい	 はい	 はい
<u>AWS::AutoScaling::ScheduledAction</u>	 はい	 はい	 はい
<u>AWS::AutoScaling::WarmPool</u>	 はい	 はい	 いいえ
<u>AWS::B2BI::Capability</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::B2BI::Partnership</u>	 はい	 はい	 いいえ
<u>AWS::B2BI::Profile</u>	 はい	 はい	 いいえ
<u>AWS::B2BI::Transformer</u>	 はい	 はい	 いいえ
<u>AWS::BCMDDataExports::Export</u>	 はい	 はい	 いいえ
<u>AWS::Backup::BackupPlan</u>	 はい	 はい	 いいえ
<u>AWS::Backup::BackupSelection</u>	 はい	 いいえ	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Backup::BackupVault</u>	 はい	 はい	 いいえ
<u>AWS::Backup::Framework</u>	 はい	 はい	 はい
<u>AWS::Backup::LogicallyAirGappedBackupVault</u>	 はい	 はい	 いいえ
<u>AWS::Backup::ReportPlan</u>	 はい	 はい	 はい
<u>AWS::Backup::RestoreTestingPlan</u>	 はい	 はい	 いいえ
<u>AWS::Backup::RestoreTestingSelection</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::BackupGateway::Hypervisor</u>	 はい	 はい	 いいえ
<u>AWS::Batch::ComputeEnvironment</u>	 はい	 はい	 はい
<u>AWS::Batch::ConsumableResource</u>	 はい	 はい	 いいえ
<u>AWS::Batch::JobDefinition</u>	 はい	 はい	 いいえ
<u>AWS::Batch::JobQueue</u>	 はい	 はい	 はい
<u>AWS::Batch::SchedulingPolicy</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::Batch::ServiceEnvironment	 はい	 はい	 いいえ
AWS::Bedrock::Agent	 はい	 はい	 はい
AWS::Bedrock::AgentAlias	 はい	 はい	 はい
AWS::Bedrock::ApplicationInferenceProfile	 はい	 はい	 はい
AWS::Bedrock::AutomatedReasoningPolicy	 はい	 はい	 いいえ
AWS::Bedrock::AutomatedReasoningPolicyVersion	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Bedrock::Blueprint</u>	 はい	 はい	 はい
<u>AWS::Bedrock::DataAutomationProject</u>	 はい	 はい	 はい
<u>AWS::Bedrock::DataSource</u>	 はい	 はい	 はい
<u>AWS::Bedrock::Flow</u>	 はい	 はい	 はい
<u>AWS::Bedrock::FlowAlias</u>	 はい	 はい	 はい
<u>AWS::Bedrock::FlowVersion</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Bedrock::Guardrail</u>	 はい	 はい	 はい
<u>AWS::Bedrock::GuardrailVersion</u>	 はい	 はい	 いいえ
<u>AWS::Bedrock::IntelligentPromptRoute</u>	 はい	 はい	 はい
<u>AWS::Bedrock::KnowledgeBase</u>	 はい	 いいえ	 はい
<u>AWS::Bedrock::Prompt</u>	 はい	 はい	 はい
<u>AWS::Bedrock::PromptVersion</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::BedrockAgentCore::BrowserCustom	 はい	 はい	 いいえ
AWS::BedrockAgentCore::CodeInterpreterCustom	 はい	 はい	 いいえ
AWS::BedrockAgentCore::Gateway	 はい	 はい	 いいえ
AWS::BedrockAgentCore::GatewayTarget	 はい	 はい	 いいえ
AWS::BedrockAgentCore::Memory	 はい	 はい	 いいえ
AWS::BedrockAgentCore::Runtime	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::BedrockAgentCore::RuntimeEndpoint</u>	 はい	 はい	 いいえ
<u>AWS::BedrockAgentCore::WorkloadIdentity</u>	 はい	 はい	 いいえ
<u>AWS::Billing::BillingView</u>	 はい	 はい	 いいえ
<u>AWS::BillingConductor::BillingGroup</u>	 はい	 はい	 はい
<u>AWS::BillingConductor::CustomLineItem</u>	 はい	 はい	 はい
<u>AWS::BillingConductor::PricingPlan</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::BillingConductor::PricingRule</u>	 はい	 はい	 はい
<u>AWS::Budgets::BudgetsAction</u>	 はい	 はい	 はい
<u>AWS::CE::AnomalyMonitor</u>	 はい	 はい	 いいえ
<u>AWS::CE::AnomalySubscription</u>	 はい	 はい	 いいえ
<u>AWS::CE::CostCategory</u>	 はい	 はい	 はい
<u>AWS::CUR::ReportDefinition</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Cases::CaseRule</u>	 はい	 はい	 いいえ
<u>AWS::Cases::Domain</u>	 はい	 はい	 いいえ
<u>AWS::Cases::Field</u>	 はい	 はい	 いいえ
<u>AWS::Cases::Layout</u>	 はい	 はい	 いいえ
<u>AWS::Cases::Template</u>	 はい	 はい	 いいえ
<u>AWS::Cassandra::Keyspace</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Cassandra::Table</u>	 はい	 はい	 はい
<u>AWS::Cassandra::Type</u>	 はい	 はい	 いいえ
<u>AWS::CertificateManager::Account</u>	 はい	 はい	 いいえ
<u>AWS::Chatbot::CustomAction</u>	 はい	 はい	 いいえ
<u>AWS::Chatbot::MicrosoftTeamsChannelConfiguration</u>	 はい	 はい	 はい
<u>AWS::Chatbot::SlackChannelConfiguration</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CleanRooms::AnalysisTemplate</u>	 はい	 はい	 はい
<u>AWS::CleanRooms::Collaboration</u>	 はい	 はい	 はい
<u>AWS::CleanRooms::ConfiguredTable</u>	 はい	 はい	 はい
<u>AWS::CleanRooms::ConfiguredTableAssociation</u>	 はい	 はい	 はい
<u>AWS::CleanRooms::IdMappingTable</u>	 はい	 はい	 いいえ
<u>AWS::CleanRooms::IdNamespaceAssociation</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CleanRooms::Membership</u>	 はい	 はい	 はい
<u>AWS::CleanRooms::PrivacyBudgetTemplate</u>	 はい	 はい	 いいえ
<u>AWS::CleanRoomsML::TrainingDataset</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::GuardHook</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::HookDefaultVersion</u>	 はい	 はい	 はい
<u>AWS::CloudFormation::HookTypeConfig</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudFormation::HookVersion</u>	 はい	 はい	 はい
<u>AWS::CloudFormation::LambdaHook</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::ModuleDefaultVersion</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::ModuleVersion</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::PublicTypeVersion</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::Publisher</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudFormation::ResourceDefaultVersion</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::ResourceVersion</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::Stack</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::StackSet</u>	 はい	 はい	 いいえ
<u>AWS::CloudFormation::TypeActivation</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::AnycastIpList</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudFront::CachePolicy</u>	 はい	 はい	 はい
<u>AWS::CloudFront::CloudFrontOriginAccessIdentity</u>	 はい	 はい	 はい
<u>AWS::CloudFront::ConnectionFunction</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::ConnectionGroup</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::ContinuousDeploymentPolicy</u>	 はい	 はい	 はい
<u>AWS::CloudFront::Distribution</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudFront::DistributionTenant</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::Function</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::KeyGroup</u>	 はい	 はい	 はい
<u>AWS::CloudFront::KeyValueStore</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::MonitoringSubscription</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::OriginAccessControl</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudFront::OriginRequestPolicy</u>	 はい	 はい	 はい
<u>AWS::CloudFront::PublicKey</u>	 はい	 はい	 はい
<u>AWS::CloudFront::RealtimeLogConfig</u>	 はい	 はい	 はい
<u>AWS::CloudFront::ResponseHeadersPolicy</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::TrustStore</u>	 はい	 はい	 いいえ
<u>AWS::CloudFront::VpcOrigin</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudTrail::Channel</u>	 はい	 はい	 いいえ
<u>AWS::CloudTrail::Dashboard</u>	 はい	 はい	 いいえ
<u>AWS::CloudTrail::EventDataStore</u>	 はい	 はい	 はい
<u>AWS::CloudTrail::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::CloudTrail::Trail</u>	 はい	 はい	 はい
<u>AWS::CloudWatch::Alarm</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CloudWatch::CompositeAlarm</u>	 はい	 はい	 はい
<u>AWS::CloudWatch::Dashboard</u>	 はい	 はい	 いいえ
<u>AWS::CloudWatch::MetricStream</u>	 はい	 はい	 いいえ
<u>AWS::CodeArtifact::Domain</u>	 はい	 はい	 はい
<u>AWS::CodeArtifact::PackageGroup</u>	 はい	 はい	 いいえ
<u>AWS::CodeArtifact::Repository</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CodeBuild::Fleet</u>	 はい	 はい	 いいえ
<u>AWS::CodeConnections::Connection</u>	 はい	 はい	 いいえ
<u>AWS::CodeDeploy::Application</u>	 はい	 はい	 はい
<u>AWS::CodeDeploy::DeploymentConfig</u>	 はい	 はい	 いいえ
<u>AWS::CodeGuruProfiler::ProfilingGroup</u>	 はい	 はい	 はい
<u>AWS::CodeGuruReviewer::RepositoryAssociation</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CodePipeline::CustomActionType</u>	 はい	 はい	 いいえ
<u>AWS::CodePipeline::Pipeline</u>	 はい	 はい	 いいえ
<u>AWS::CodePipeline::Webhook</u>	 はい	 はい	 いいえ
<u>AWS::CodeStarConnections::Connection</u>	 はい	 はい	 はい
<u>AWS::CodeStarConnections::RepositoryLink</u>	 はい	 はい	 いいえ
<u>AWS::CodeStarConnections::SyncConfiguration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CodeStarNotifications::NotificationRule</u>	 はい	 はい	 はい
<u>AWS::Cognito::IdentityPool</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::IdentityPoolPrincipalTag</u>	 はい	 はい	 はい
<u>AWS::Cognito::IdentityPoolRoleAttachment</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::LogDeliveryConfiguration</u>	 はい	 いいえ	 いいえ
<u>AWS::Cognito::ManagedLoginBranding</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	IaC ジェネレーター
<u>AWS::Cognito::Terms</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPool</u>	 はい	 いいえ	 いいえ
<u>AWS::Cognito::UserPoolClient</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPoolDomain</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPoolGroup</u>	 はい	 はい	 はい
<u>AWS::Cognito::UserPoolIdentityProvider</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Cognito::UserPoolResourceServer</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPoolRiskConfigurationAttachment</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPoolUICustomizationAttachment</u>	 はい	 はい	 いいえ
<u>AWS::Cognito::UserPoolUser</u>	 はい	 いいえ	 いいえ
<u>AWS::Cognito::UserPoolUserToGroupAttachment</u>	 はい	 はい	 いいえ
<u>AWS::Comprehend::DocumentClassifier</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::Comprehend::Flywheel	 はい	 はい	 はい
AWS::Config::AggregationAuthorization	 はい	 はい	 はい
AWS::Config::ConfigRule	 はい	 はい	 はい
AWS::Config::ConfigurationAggregator	 はい	 はい	 はい
AWS::Config::ConformancePack	 はい	 はい	 はい
AWS::Config::OrganizationConformancePack	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Config::StoredQuery</u>	 はい	 はい	 はい
<u>AWS::Connect::AgentStatus</u>	 はい	 はい	 はい
<u>AWS::Connect::ApprovedOrigin</u>	 はい	 はい	 はい
<u>AWS::Connect::ContactFlow</u>	 はい	 はい	 はい
<u>AWS::Connect::ContactFlowModule</u>	 はい	 はい	 はい
<u>AWS::Connect::ContactFlowVersion</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Connect::DataTable</u>	 はい	 はい	 いいえ
<u>AWS::Connect::DataTableAttribute</u>	 はい	 はい	 いいえ
<u>AWS::Connect::DataTableRecord</u>	 はい	 はい	 いいえ
<u>AWS::Connect::EmailAddress</u>	 はい	 はい	 はい
<u>AWS::Connect::EvaluationForm</u>	 はい	 はい	 はい
<u>AWS::Connect::HoursOfOperation</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Connect::Instance</u>	 はい	 はい	 はい
<u>AWS::Connect::InstanceStorageConfig</u>	 はい	 はい	 はい
<u>AWS::Connect::IntegrationAssociation</u>	 はい	 はい	 はい
<u>AWS::Connect::PhoneNumber</u>	 はい	 はい	 はい
<u>AWS::Connect::PredefinedAttribute</u>	 はい	 はい	 はい
<u>AWS::Connect::Prompt</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Connect::Queue</u>	 はい	 はい	 はい
<u>AWS::Connect::QuickConnect</u>	 はい	 はい	 はい
<u>AWS::Connect::RoutingProfile</u>	 はい	 はい	 はい
<u>AWS::Connect::Rule</u>	 はい	 はい	 いいえ
<u>AWS::Connect::SecurityKey</u>	 はい	 はい	 はい
<u>AWS::Connect::SecurityProfile</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Connect::TaskTemplate</u>	 はい	 はい	 はい
<u>AWS::Connect::TrafficDistributionGroup</u>	 はい	 はい	 はい
<u>AWS::Connect::User</u>	 はい	 はい	 はい
<u>AWS::Connect::UserHierarchyGroup</u>	 はい	 はい	 はい
<u>AWS::Connect::UserHierarchyStructure</u>	 はい	 はい	 いいえ
<u>AWS::Connect::View</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Connect::ViewVersion</u>	 はい	 はい	 はい
<u>AWS::Connect::Workspace</u>	 はい	 はい	 いいえ
<u>AWS::ConnectCampaigns::Campaign</u>	 はい	 はい	 はい
<u>AWS::ConnectCampaignsV2::Campaign</u>	 はい	 はい	 はい
<u>AWS::ControlTower::EnabledBaseline</u>	 はい	 はい	 いいえ
<u>AWS::ControlTower::EnabledControl</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ControlTower::LandingZone</u>	 はい	 はい	 いいえ
<u>AWS::CustomerProfiles::CalculatedAttributeDefinition</u>	 はい	 はい	 はい
<u>AWS::CustomerProfiles::Domain</u>	 はい	 はい	 はい
<u>AWS::CustomerProfiles::EventStream</u>	 はい	 はい	 はい
<u>AWS::CustomerProfiles::EventTrigger</u>	 はい	 はい	 いいえ
<u>AWS::CustomerProfiles::Integration</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::CustomerProfiles::ObjectType</u>	 はい	 はい	 はい
<u>AWS::CustomerProfiles::SegmentDefinition</u>	 はい	 はい	 いいえ
<u>AWS::DMS::DataMigration</u>	 はい	 はい	 いいえ
<u>AWS::DMS::DataProvider</u>	 はい	 はい	 いいえ
<u>AWS::DMS::InstanceProfile</u>	 はい	 はい	 いいえ
<u>AWS::DMS::MigrationProject</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::DMS::ReplicationConfig	 はい	 はい	 はい
AWS::DSQL::Cluster	 はい	 はい	 いいえ
AWS::DataBrew::Dataset	 はい	 はい	 はい
AWS::DataBrew::Job	 はい	 はい	 はい
AWS::DataBrew::Project	 はい	 はい	 はい
AWS::DataBrew::Recipe	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataBrew::Ruleset</u>	 はい	 はい	 はい
<u>AWS::DataBrew::Schedule</u>	 はい	 はい	 はい
<u>AWS::DataPipeline::Pipeline</u>	 はい	 はい	 はい
<u>AWS::DataSync::Agent</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationAzureBlob</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationEFS</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataSync::LocationFSxLustre</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationFSxONTAP</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationFSxOpenZFS</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationFSxWindows</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationHDFS</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationNFS</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataSync::LocationObjectStorage</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationS3</u>	 はい	 はい	 はい
<u>AWS::DataSync::LocationSMB</u>	 はい	 はい	 はい
<u>AWS::DataSync::Task</u>	 はい	 はい	 はい
<u>AWS::DataZone::Connection</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::DataSource</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataZone::Domain</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::DomainUnit</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::Environment</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::EnvironmentActions</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::EnvironmentBlueprintConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::EnvironmentProfile</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataZone::FormType</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::GroupProfile</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::Owner</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::PolicyGrant</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::Project</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::ProjectMembership</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DataZone::ProjectProfile</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::SubscriptionTarget</u>	 はい	 はい	 いいえ
<u>AWS::DataZone::UserProfile</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::Farm</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::Fleet</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::LicenseEndpoint</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Deadline::Limit</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::MeteredProduct</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::Monitor</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::Queue</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::QueueEnvironment</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::QueueFleetAssociation</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Deadline::QueueLimitAssociation</u>	 はい	 はい	 いいえ
<u>AWS::Deadline::StorageProfile</u>	 はい	 はい	 いいえ
<u>AWS::Detective::Graph</u>	 はい	 はい	 はい
<u>AWS::Detective::MemberInvitation</u>	 はい	 はい	 はい
<u>AWS::Detective::OrganizationAdmin</u>	 はい	 はい	 はい
<u>AWS::DevOpsAgent::AgentSpace</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DevOpsAgent::Association</u>	 はい	 はい	 いいえ
<u>AWS::DevOpsGuru::LogAnomalyDetectionIntegration</u>	 はい	 はい	 はい
<u>AWS::DevOpsGuru::NotificationChannel</u>	 はい	 はい	 はい
<u>AWS::DevOpsGuru::ResourceCollection</u>	 はい	 はい	 はい
<u>AWS::DeviceFarm::DevicePool</u>	 はい	 はい	 いいえ
<u>AWS::DeviceFarm::InstanceProfile</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DeviceFarm::NetworkProfile</u>	 はい	 はい	 いいえ
<u>AWS::DeviceFarm::Project</u>	 はい	 はい	 いいえ
<u>AWS::DeviceFarm::TestGridProject</u>	 はい	 はい	 いいえ
<u>AWS::DeviceFarm::VPCEConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::DirectoryService::SimpleAD</u>	 はい	 はい	 はい
<u>AWS::DocDBElastic::Cluster</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::DynamoDB::GlobalTable</u>	 はい	 はい	 いいえ
<u>AWS::DynamoDB::Table</u>	 はい	 はい	 はい
<u>AWS::EC2::CapacityManagerDataExport</u>	 はい	 はい	 いいえ
<u>AWS::EC2::CapacityReservation</u>	 はい	 はい	 はい
<u>AWS::EC2::CapacityReservationFleet</u>	 はい	 はい	 はい
<u>AWS::EC2::CarrierGateway</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::CustomerGateway</u>	 はい	 はい	 はい
<u>AWS::EC2::DHCPOptions</u>	 はい	 はい	 はい
<u>AWS::EC2::EC2Fleet</u>	 はい	 はい	 いいえ
<u>AWS::EC2::EIP</u>	 はい	 はい	 はい
<u>AWS::EC2::EIPAssociation</u>	 はい	 いいえ	 はい
<u>AWS::EC2::EgressOnlyInternetGateway</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::EC2::EnclaveCertificateIamRoleAssociation	 はい	 はい	 はい
AWS::EC2::FlowLog	 はい	 はい	 はい
AWS::EC2::GatewayRouteTableAssociation	 はい	 はい	 いいえ
AWS::EC2::Host	 はい	 はい	 はい
AWS::EC2::IPAM	 はい	 はい	 はい
AWS::EC2::IPAMAllocation	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::EC2::IPAMPool	 はい	 はい	 はい
AWS::EC2::IPAMPoolCidr	 はい	 はい	 はい
AWS::EC2::IPAMResourceDiscovery	 はい	 はい	 はい
AWS::EC2::IPAMResourceDiscoveryAssociation	 はい	 はい	 はい
AWS::EC2::IPAMScope	 はい	 はい	 はい
AWS::EC2::Instance	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::InstanceConnectEndpoint</u>	 はい	 はい	 はい
<u>AWS::EC2::InternetGateway</u>	 はい	 はい	 はい
<u>AWS::EC2::IpPoolRouteTableAssociation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::KeyPair</u>	 はい	 はい	 はい
<u>AWS::EC2::LaunchTemplate</u>	 はい	 はい	 いいえ
<u>AWS::EC2::LocalGatewayRoute</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::LocalGatewayRouteTable</u>	 はい	 はい	 はい
<u>AWS::EC2::LocalGatewayRouteTableVPCAssociation</u>	 はい	 はい	 はい
<u>AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation</u>	 はい	 はい	 はい
<u>AWS::EC2::LocalGatewayVirtualInterface</u>	 はい	 はい	 いいえ
<u>AWS::EC2::LocalGatewayVirtualInterfaceGroup</u>	 はい	 はい	 いいえ
<u>AWS::EC2::NatGateway</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::EC2::NetworkAcl	 はい	 はい	 はい
AWS::EC2::NetworkInsightsAccessScope	 はい	 はい	 いいえ
AWS::EC2::NetworkInsightsAccessScopeAnalysis	 はい	 はい	 はい
AWS::EC2::NetworkInsightsAnalysis	 はい	 はい	 いいえ
AWS::EC2::NetworkInsightsPath	 はい	 はい	 はい
AWS::EC2::NetworkInterface	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::EC2::NetworkInterfaceAttachment	 はい	 はい	 はい
AWS::EC2::NetworkPerformanceMetricSubscription	 はい	 はい	 はい
AWS::EC2::PlacementGroup	 はい	 はい	 はい
AWS::EC2::PrefixList	 はい	 はい	 いいえ
AWS::EC2::Route	 はい	 はい	 はい
AWS::EC2::RouteServer	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::RouteServerAssociation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::RouteServerEndpoint</u>	 はい	 はい	 いいえ
<u>AWS::EC2::RouteServerPeer</u>	 はい	 はい	 いいえ
<u>AWS::EC2::RouteServerPropagation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::RouteTable</u>	 はい	 はい	 はい
<u>AWS::EC2::SecurityGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::SecurityGroupEgress</u>	 はい	 はい	 いいえ
<u>AWS::EC2::SecurityGroupIngress</u>	 はい	 はい	 いいえ
<u>AWS::EC2::SecurityGroupVpcAssociation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::SnapshotBlockPublicAccess</u>	 はい	 はい	 いいえ
<u>AWS::EC2::SpotFleet</u>	 はい	 はい	 いいえ
<u>AWS::EC2::Subnet</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::SubnetCidrBlock</u>	 はい	 はい	 はい
<u>AWS::EC2::SubnetNetworkAclAssociation</u>	 はい	 はい	 はい
<u>AWS::EC2::SubnetRouteTableAssociation</u>	 はい	 いいえ	 はい
<u>AWS::EC2::TrafficMirrorFilter</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TrafficMirrorFilterRule</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TrafficMirrorSession</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::TrafficMirrorTarget</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGateway</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayAttachment</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayConnect</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayConnectPeer</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGatewayMeteringPolicy</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::TransitGatewayMeteringPolicyEntry</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGatewayMulticastDomain</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayMulticastDomainAssociation</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayMulticastGroupMember</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayMulticastGroupSource</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayPeeringAttachment</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::TransitGatewayRoute</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGatewayRouteTable</u>	 はい	 はい	 はい
<u>AWS::EC2::TransitGatewayRouteTableAssociation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGatewayRouteTablePropagation</u>	 はい	 はい	 いいえ
<u>AWS::EC2::TransitGatewayVpcAttachment</u>	 はい	 はい	 はい
<u>AWS::EC2::VPC</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::VPCLockPublicAccessExclusion</u>	 はい	 はい	 いいえ
<u>AWS::EC2::VPCLockPublicAccessOptions</u>	 はい	 はい	 いいえ
<u>AWS::EC2::VPCCIDRBlock</u>	 はい	 はい	 いいえ
<u>AWS::EC2::VPCDHCPOptionsAssociation</u>	 はい	 はい	 はい
<u>AWS::EC2::VPCEncryptionControl</u>	 はい	 はい	 いいえ
<u>AWS::EC2::VPCEndpoint</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::EC2::VPCEndpointConnectionNotification	 はい	 はい	 はい
AWS::EC2::VPCEndpointService	 はい	 はい	 はい
AWS::EC2::VPCEndpointServicePermissions	 はい	 はい	 いいえ
AWS::EC2::VPCGatewayAttachment	 はい	 はい	 いいえ
AWS::EC2::VPCPeeringConnection	 はい	 はい	 はい
AWS::EC2::VPNConcentrator	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::VPNConnection</u>	 はい	 はい	 はい
<u>AWS::EC2::VPNConnectionRoute</u>	 はい	 はい	 はい
<u>AWS::EC2::VPNGateway</u>	 はい	 はい	 はい
<u>AWS::EC2::VerifiedAccessEndpoint</u>	 はい	 はい	 はい
<u>AWS::EC2::VerifiedAccessGroup</u>	 はい	 はい	 はい
<u>AWS::EC2::VerifiedAccessInstance</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EC2::VerifiedAccessTrustProvider</u>	 はい	 はい	 はい
<u>AWS::EC2::Volume</u>	 はい	 はい	 はい
<u>AWS::EC2::VolumeAttachment</u>	 はい	 はい	 はい
<u>AWS::ECR::PublicRepository</u>	 はい	 はい	 はい
<u>AWS::ECR::PullThroughCacheRule</u>	 はい	 はい	 はい
<u>AWS::ECR::PullTimeUpdateExclusion</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ECR::RegistryPolicy</u>	 はい	 はい	 はい
<u>AWS::ECR::RegistryScanningConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::ECR::ReplicationConfiguration</u>	 はい	 はい	 はい
<u>AWS::ECR::Repository</u>	 はい	 はい	 はい
<u>AWS::ECR::RepositoryCreationTemplate</u>	 はい	 はい	 いいえ
<u>AWS::ECR::SigningConfiguration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ECS::CapacityProvider</u>	 はい	 はい	 いいえ
<u>AWS::ECS::Cluster</u>	 はい	 いいえ	 はい
<u>AWS::ECS::ClusterCapacityProviderAssociations</u>	 はい	 いいえ	 はい
<u>AWS::ECS::ExpressGatewayService</u>	 はい	 はい	 いいえ
<u>AWS::ECS::PrimaryTaskSet</u>	 はい	 はい	 いいえ
<u>AWS::ECS::Service</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ECS::TaskDefinition</u>	 はい	 はい	 はい
<u>AWS::ECS::TaskSet</u>	 はい	 はい	 いいえ
<u>AWS::EFS::AccessPoint</u>	 はい	 はい	 はい
<u>AWS::EFS::FileSystem</u>	 はい	 はい	 はい
<u>AWS::EFS::MountTarget</u>	 はい	 はい	 はい
<u>AWS::EKS::AccessEntry</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EKS::Addon</u>	 はい	 はい	 はい
<u>AWS::EKS::Capability</u>	 はい	 はい	 いいえ
<u>AWS::EKS::Cluster</u>	 はい	 いいえ	 はい
<u>AWS::EKS::FargateProfile</u>	 はい	 はい	 はい
<u>AWS::EKS::IdentityProviderConfig</u>	 はい	 はい	 はい
<u>AWS::EKS::Nodegroup</u>	 はい	 いいえ	 はい

リソース	[Import] (インポート)	ドリフト検出	IaC ジェネレーター
<u>AWS::EKS::PodIdentityAssociation</u>	 はい	 はい	 いいえ
<u>AWS::EMR::SecurityConfiguration</u>	 はい	 はい	 はい
<u>AWS::EMR::Step</u>	 はい	 いいえ	 いいえ
<u>AWS::EMR::Studio</u>	 はい	 はい	 はい
<u>AWS::EMR::StudioSessionMapping</u>	 はい	 はい	 はい
<u>AWS::EMR::WALWorkspace</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EMRContainers::VirtualCluster</u>	 はい	 はい	 いいえ
<u>AWS::EMRServerless::Application</u>	 はい	 はい	 はい
<u>AWS::EVS::Environment</u>	 はい	 はい	 いいえ
<u>AWS::ElastiCache::GlobalReplicationGroup</u>	 はい	 はい	 いいえ
<u>AWS::ElastiCache::ParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::ElastiCache::ServerlessCache</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ElastiCache::SubnetGroup</u>	 はい	 はい	 はい
<u>AWS::ElastiCache::User</u>	 はい	 はい	 はい
<u>AWS::ElastiCache::UserGroup</u>	 はい	 はい	 はい
<u>AWS::ElasticBeanstalk::Application</u>	 はい	 はい	 はい
<u>AWS::ElasticBeanstalk::ApplicationVersion</u>	 はい	 はい	 はい
<u>AWS::ElasticBeanstalk::ConfigurationTemplate</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ElasticBeanstalk::Environment</u>	 はい	 いいえ	 いいえ
<u>AWS::ElasticLoadBalancingV2::Listener</u>	 はい	 はい	 はい
<u>AWS::ElasticLoadBalancingV2::ListenerRule</u>	 はい	 いいえ	 はい
<u>AWS::ElasticLoadBalancingV2::LoadBalancer</u>	 はい	 はい	 はい
<u>AWS::ElasticLoadBalancingV2::TargetGroup</u>	 はい	 はい	 はい
<u>AWS::ElasticLoadBalancingV2::TrustStore</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ElasticLoadBalancingV2::TrustStoreRevocation</u>	 はい	 はい	 いいえ
<u>AWS::EntityResolution::IdMappingWorkflow</u>	 はい	 はい	 はい
<u>AWS::EntityResolution::IdNamespace</u>	 はい	 はい	 いいえ
<u>AWS::EntityResolution::MatchingWorkflow</u>	 はい	 はい	 はい
<u>AWS::EntityResolution::PolicyStatement</u>	 はい	 はい	 いいえ
<u>AWS::EntityResolution::SchemaMapping</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::EventSchemas::Discoverer</u>	 はい	 はい	 いいえ
<u>AWS::EventSchemas::Registry</u>	 はい	 はい	 いいえ
<u>AWS::EventSchemas::RegistryPolicy</u>	 はい	 はい	 いいえ
<u>AWS::EventSchemas::Schema</u>	 はい	 はい	 いいえ
<u>AWS::Events::ApiDestination</u>	 はい	 はい	 はい
<u>AWS::Events::Archive</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Events::Connection</u>	 はい	 はい	 いいえ
<u>AWS::Events::Endpoint</u>	 はい	 はい	 はい
<u>AWS::Events::EventBus</u>	 はい	 はい	 はい
<u>AWS::Events::EventBusPolicy</u>	 はい	 はい	 いいえ
<u>AWS::Events::Rule</u>	 はい	 はい	 はい
<u>AWS::Evidently::Experiment</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Evidently::Feature</u>	 はい	 はい	 いいえ
<u>AWS::Evidently::Launch</u>	 はい	 はい	 いいえ
<u>AWS::Evidently::Project</u>	 はい	 はい	 いいえ
<u>AWS::Evidently::Segment</u>	 はい	 はい	 いいえ
<u>AWS::FIS::ExperimentTemplate</u>	 はい	 はい	 はい
<u>AWS::FIS::TargetAccountConfiguration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::FMS::NotificationChannel</u>	 はい	 はい	 いいえ
<u>AWS::FMS::Policy</u>	 はい	 いいえ	 いいえ
<u>AWS::FMS::ResourceSet</u>	 はい	 はい	 いいえ
<u>AWS::FSx::DataRepositoryAssociation</u>	 はい	 はい	 はい
<u>AWS::FSx::S3AccessPointAttachment</u>	 はい	 はい	 いいえ
<u>AWS::FinSpace::Environment</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Forecast::Dataset</u>	 はい	 はい	 いいえ
<u>AWS::Forecast::DatasetGroup</u>	 はい	 はい	 いいえ
<u>AWS::FraudDetector::Detector</u>	 はい	 はい	 はい
<u>AWS::FraudDetector::EntityType</u>	 はい	 はい	 はい
<u>AWS::FraudDetector::EventType</u>	 はい	 はい	 はい
<u>AWS::FraudDetector::Label</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::FraudDetector::List</u>	 はい	 はい	 はい
<u>AWS::FraudDetector::Outcome</u>	 はい	 はい	 はい
<u>AWS::FraudDetector::Variable</u>	 はい	 はい	 はい
<u>AWS::GameLift::Alias</u>	 はい	 はい	 はい
<u>AWS::GameLift::Build</u>	 はい	 はい	 はい
<u>AWS::GameLift::ContainerFleet</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GameLift::ContainerGroupDefinition</u>	 はい	 はい	 いいえ
<u>AWS::GameLift::Fleet</u>	 はい	 はい	 はい
<u>AWS::GameLift::GameServerGroup</u>	 はい	 はい	 いいえ
<u>AWS::GameLift::GameSessionQueue</u>	 はい	 はい	 いいえ
<u>AWS::GameLift::Location</u>	 はい	 はい	 いいえ
<u>AWS::GameLift::MatchmakingConfiguration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GameLift::MatchmakingRuleSet</u>	 はい	 はい	 いいえ
<u>AWS::GameLift::Script</u>	 はい	 はい	 いいえ
<u>AWS::GameLiftStreams::Application</u>	 はい	 はい	 いいえ
<u>AWS::GameLiftStreams::StreamGroup</u>	 はい	 はい	 いいえ
<u>AWS::GlobalAccelerator::Accelerator</u>	 はい	 はい	 はい
<u>AWS::GlobalAccelerator::CrossAccount Attachment</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GlobalAccelerator::EndpointGroup</u>	 はい	 はい	 はい
<u>AWS::GlobalAccelerator::Listener</u>	 はい	 はい	 はい
<u>AWS::Glue::Crawler</u>	 はい	 はい	 いいえ
<u>AWS::Glue::Database</u>	 はい	 はい	 いいえ
<u>AWS::Glue::IdentityCenterConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::Glue::Integration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Glue::IntegrationResourceProperty</u>	 はい	 はい	 いいえ
<u>AWS::Glue::Job</u>	 はい	 はい	 いいえ
<u>AWS::Glue::Registry</u>	 はい	 はい	 はい
<u>AWS::Glue::Schema</u>	 はい	 はい	 はい
<u>AWS::Glue::SchemaVersion</u>	 はい	 はい	 はい
<u>AWS::Glue::SchemaVersionMetadata</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Glue::Trigger</u>	 はい	 はい	 いいえ
<u>AWS::Glue::UsageProfile</u>	 はい	 はい	 いいえ
<u>AWS::Grafana::Workspace</u>	 はい	 はい	 はい
<u>AWS::GreengrassV2::ComponentVersion</u>	 はい	 はい	 いいえ
<u>AWS::GreengrassV2::Deployment</u>	 はい	 はい	 はい
<u>AWS::GroundStation::Config</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GroundStation::DataflowEndpointGroup</u>	 はい	 はい	 はい
<u>AWS::GroundStation::DataflowEndpointGroupV2</u>	 はい	 はい	 いいえ
<u>AWS::GroundStation::MissionProfile</u>	 はい	 はい	 はい
<u>AWS::GuardDuty::Detector</u>	 はい	 はい	 はい
<u>AWS::GuardDuty::Filter</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::IPSet</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GuardDuty::MalwareProtectionPlan</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::Master</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::Member</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::PublishingDestination</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::ThreatEntitySet</u>	 はい	 はい	 いいえ
<u>AWS::GuardDuty::ThreatIntelSet</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::GuardDuty::TrustedEntitySet</u>	 はい	 はい	 いいえ
<u>AWS::HealthImaging::Datastore</u>	 はい	 はい	 はい
<u>AWS::HealthLake::FHIRDatastore</u>	 はい	 はい	 はい
<u>AWS::IAM::Group</u>	 はい	 はい	 はい
<u>AWS::IAM::GroupPolicy</u>	 はい	 はい	 いいえ
<u>AWS::IAM::InstanceProfile</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::IAM::ManagedPolicy	 はい	 はい	 はい
AWS::IAM::OIDCProvider	 はい	 はい	 はい
AWS::IAM::Role	 はい	 はい	 はい
AWS::IAM::RolePolicy	 はい	 はい	 いいえ
AWS::IAM::SAMLProvider	 はい	 はい	 はい
AWS::IAM::ServerCertificate	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IAM::ServiceLinkedRole</u>	 はい	 はい	 いいえ
<u>AWS::IAM::User</u>	 はい	 はい	 はい
<u>AWS::IAM::UserPolicy</u>	 はい	 はい	 いいえ
<u>AWS::IAM::VirtualMFADevice</u>	 はい	 はい	 いいえ
<u>AWS::IVS::Channel</u>	 はい	 はい	 はい
<u>AWS::IVS::EncoderConfiguration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IVS::IngestConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::IVS::PlaybackKeyPair</u>	 はい	 はい	 はい
<u>AWS::IVS::PlaybackRestrictionPolicy</u>	 はい	 はい	 いいえ
<u>AWS::IVS::PublicKey</u>	 はい	 はい	 いいえ
<u>AWS::IVS::RecordingConfiguration</u>	 はい	 はい	 はい
<u>AWS::IVS::Stage</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IVS::StorageConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::IVS::StreamKey</u>	 はい	 はい	 いいえ
<u>AWS::IVSChat::LoggingConfiguration</u>	 はい	 はい	 はい
<u>AWS::IVSChat::Room</u>	 はい	 はい	 はい
<u>AWS::IdentityStore::Group</u>	 はい	 はい	 はい
<u>AWS::IdentityStore::GroupMembership</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ImageBuilder::Component</u>	 はい	 はい	 はい
<u>AWS::ImageBuilder::ContainerRecipe</u>	 はい	 はい	 いいえ
<u>AWS::ImageBuilder::DistributionConfiguration</u>	 はい	 はい	 はい
<u>AWS::ImageBuilder::Image</u>	 はい	 はい	 はい
<u>AWS::ImageBuilder::ImagePipeline</u>	 はい	 はい	 はい
<u>AWS::ImageBuilder::ImageRecipe</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ImageBuilder::InfrastructureConfiguration</u>	 はい	 はい	 はい
<u>AWS::ImageBuilder::LifecyclePolicy</u>	 はい	 はい	 いいえ
<u>AWS::ImageBuilder::Workflow</u>	 はい	 はい	 いいえ
<u>AWS::Inspector::AssessmentTarget</u>	 はい	 はい	 はい
<u>AWS::Inspector::AssessmentTemplate</u>	 はい	 はい	 はい
<u>AWS::Inspector::ResourceGroup</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::InspectorV2::CisScanConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::InspectorV2::CodeSecurityIntegration</u>	 はい	 はい	 いいえ
<u>AWS::InspectorV2::CodeSecurityScanConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::InspectorV2::Filter</u>	 はい	 はい	 いいえ
<u>AWS::InternetMonitor::Monitor</u>	 はい	 はい	 はい
<u>AWS::Invoicing::InvoiceUnit</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT::AccountAuditConfiguration</u>	 はい	 はい	 はい
<u>AWS::IoT::Authorizer</u>	 はい	 はい	 はい
<u>AWS::IoT::BillingGroup</u>	 はい	 はい	 はい
<u>AWS::IoT::CACertificate</u>	 はい	 はい	 はい
<u>AWS::IoT::Certificate</u>	 はい	 はい	 はい
<u>AWS::IoT::CertificateProvider</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT::Command</u>	 はい	 はい	 いいえ
<u>AWS::IoT::CustomMetric</u>	 はい	 はい	 はい
<u>AWS::IoT::Dimension</u>	 はい	 はい	 はい
<u>AWS::IoT::DomainConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::IoT::EncryptionConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::IoT::FleetMetric</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::IoT::JobTemplate	 はい	 はい	 いいえ
AWS::IoT::Logging	 はい	 はい	 はい
AWS::IoT::MitigationAction	 はい	 はい	 はい
AWS::IoT::Policy	 はい	 はい	 はい
AWS::IoT::ProvisioningTemplate	 はい	 はい	 はい
AWS::IoT::ResourceSpecificLogging	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT::RoleAlias</u>	 はい	 はい	 はい
<u>AWS::IoT::ScheduledAudit</u>	 はい	 はい	 はい
<u>AWS::IoT::SecurityProfile</u>	 はい	 はい	 はい
<u>AWS::IoT::SoftwarePackage</u>	 はい	 はい	 はい
<u>AWS::IoT::SoftwarePackageVersion</u>	 はい	 はい	 いいえ
<u>AWS::IoT::Thing</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT::ThingGroup</u>	 はい	 はい	 はい
<u>AWS::IoT::ThingType</u>	 はい	 はい	 いいえ
<u>AWS::IoT::TopicRule</u>	 はい	 はい	 はい
<u>AWS::IoT::TopicRuleDestination</u>	 はい	 はい	 はい
<u>AWS::IoTAnalytics::Channel</u>	 はい	 はい	 はい
<u>AWS::IoTAnalytics::Dataset</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoTAnalytics::Datastore</u>	 はい	 はい	 はい
<u>AWS::IoTAnalytics::Pipeline</u>	 はい	 はい	 はい
<u>AWS::IoTCoreDeviceAdvisor::SuiteDefinition</u>	 はい	 はい	 はい
<u>AWS::IoTEvents::AlarmModel</u>	 はい	 はい	 はい
<u>AWS::IoTEvents::DetectorModel</u>	 はい	 はい	 はい
<u>AWS::IoTEvents::Input</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT FleetWise::Campaign</u>	 はい	 はい	 はい
<u>AWS::IoT FleetWise::DecoderManifest</u>	 はい	 はい	 はい
<u>AWS::IoT FleetWise::Fleet</u>	 はい	 はい	 はい
<u>AWS::IoT FleetWise::ModelManifest</u>	 はい	 はい	 はい
<u>AWS::IoT FleetWise::SignalCatalog</u>	 はい	 はい	 はい
<u>AWS::IoT FleetWise::StateTemplate</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT FleetWise::Vehicle</u>	 はい	 はい	 はい
<u>AWS::IoT SiteWise::AccessPolicy</u>	 はい	 はい	 いいえ
<u>AWS::IoT SiteWise::Asset</u>	 はい	 はい	 はい
<u>AWS::IoT SiteWise::AssetModel</u>	 はい	 はい	 はい
<u>AWS::IoT SiteWise::ComputationModel</u>	 はい	 はい	 いいえ
<u>AWS::IoT SiteWise::Dashboard</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoTSiteWise::Dataset</u>	 はい	 はい	 いいえ
<u>AWS::IoTSiteWise::Gateway</u>	 はい	 はい	 はい
<u>AWS::IoTSiteWise::Portal</u>	 はい	 はい	 いいえ
<u>AWS::IoTSiteWise::Project</u>	 はい	 はい	 いいえ
<u>AWS::IoTtwinMaker::ComponentType</u>	 はい	 はい	 はい
<u>AWS::IoTtwinMaker::Entity</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoT TwinMaker::Scene</u>	 はい	 はい	 はい
<u>AWS::IoT TwinMaker::SyncJob</u>	 はい	 はい	 はい
<u>AWS::IoT TwinMaker::Workspace</u>	 はい	 はい	 はい
<u>AWS::IoT Wireless::Destination</u>	 はい	 はい	 はい
<u>AWS::IoT Wireless::DeviceProfile</u>	 はい	 はい	 はい
<u>AWS::IoT Wireless::FirmwareTask</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoTWireless::MulticastGroup</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::NetworkAnalyzerConfiguration</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::PartnerAccount</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::ServiceProfile</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::TaskDefinition</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::WirelessDevice</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::IoTWireless::WirelessDeviceImportTask</u>	 はい	 はい	 はい
<u>AWS::IoTWireless::WirelessGateway</u>	 はい	 はい	 はい
<u>AWS::KMS::Alias</u>	 はい	 いいえ	 はい
<u>AWS::KMS::Key</u>	 はい	 いいえ	 はい
<u>AWS::KMS::ReplicaKey</u>	 はい	 いいえ	 はい
<u>AWS::KafkaConnect::Connector</u>	 はい	 いいえ	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::KafkaConnect::CustomPlugin	 はい	 はい	 いいえ
AWS::KafkaConnect::WorkerConfiguration	 はい	 はい	 いいえ
AWS::Kendra::DataSource	 はい	 はい	 はい
AWS::Kendra::FAQ	 はい	 はい	 はい
AWS::Kendra::Index	 はい	 はい	 はい
AWS::KendraRanking::ExecutionPlan	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Kinesis::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::Kinesis::Stream</u>	 はい	 はい	 はい
<u>AWS::Kinesis::StreamConsumer</u>	 はい	 はい	 いいえ
<u>AWS::KinesisAnalyticsV2::Application</u>	 はい	 はい	 いいえ
<u>AWS::KinesisFirehose::DeliveryStream</u>	 はい	 いいえ	 いいえ
<u>AWS::KinesisVideo::SignalingChannel</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::KinesisVideo::Stream	 はい	 はい	 いいえ
AWS::LakeFormation::DataCellsFilter	 はい	 はい	 いいえ
AWS::LakeFormation::PrincipalPermissions	 はい	 はい	 いいえ
AWS::LakeFormation::Tag	 はい	 はい	 いいえ
AWS::LakeFormation::TagAssociation	 はい	 はい	 いいえ
AWS::Lambda::Alias	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lambda::CapacityProvider</u>	 はい	 はい	 いいえ
<u>AWS::Lambda::CodeSigningConfig</u>	 はい	 はい	 はい
<u>AWS::Lambda::EventInvokeConfig</u>	 はい	 はい	 いいえ
<u>AWS::Lambda::EventSourceMapping</u>	 はい	 はい	 はい
<u>AWS::Lambda::Function</u>	 はい	 はい	 はい
<u>AWS::Lambda::LayerVersion</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lambda::LayerVersionPermission</u>	 はい	 はい	 はい
<u>AWS::Lambda::Permission</u>	 はい	 はい	 はい
<u>AWS::Lambda::Url</u>	 はい	 はい	 はい
<u>AWS::Lambda::Version</u>	 はい	 はい	 はい
<u>AWS::LaunchWizard::Deployment</u>	 はい	 はい	 いいえ
<u>AWS::Lex::Bot</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lex::BotAlias</u>	 はい	 はい	 いいえ
<u>AWS::Lex::BotVersion</u>	 はい	 はい	 いいえ
<u>AWS::Lex::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::LicenseManager::Grant</u>	 はい	 はい	 いいえ
<u>AWS::LicenseManager::License</u>	 はい	 はい	 いいえ
<u>AWS::Lightsail::Alarm</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lightsail::Bucket</u>	 はい	 はい	 はい
<u>AWS::Lightsail::Certificate</u>	 はい	 はい	 はい
<u>AWS::Lightsail::Container</u>	 はい	 はい	 はい
<u>AWS::Lightsail::Database</u>	 はい	 はい	 いいえ
<u>AWS::Lightsail::Disk</u>	 はい	 はい	 はい
<u>AWS::Lightsail::DiskSnapshot</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lightsail::Distribution</u>	 はい	 はい	 はい
<u>AWS::Lightsail::Domain</u>	 はい	 はい	 いいえ
<u>AWS::Lightsail::Instance</u>	 はい	 はい	 はい
<u>AWS::Lightsail::InstanceSnapshot</u>	 はい	 はい	 いいえ
<u>AWS::Lightsail::LoadBalancer</u>	 はい	 はい	 はい
<u>AWS::Lightsail::LoadBalancerTlsCertificate</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Lightsail::StaticIp</u>	 はい	 はい	 はい
<u>AWS::Location::APIKey</u>	 はい	 はい	 いいえ
<u>AWS::Location::GeofenceCollection</u>	 はい	 はい	 はい
<u>AWS::Location::Map</u>	 はい	 はい	 はい
<u>AWS::Location::PlaceIndex</u>	 はい	 はい	 はい
<u>AWS::Location::RouteCalculator</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Location::Tracker</u>	 はい	 はい	 はい
<u>AWS::Location::TrackerConsumer</u>	 はい	 はい	 はい
<u>AWS::Logs::AccountPolicy</u>	 はい	 はい	 はい
<u>AWS::Logs::Delivery</u>	 はい	 はい	 いいえ
<u>AWS::Logs::DeliveryDestination</u>	 はい	 はい	 いいえ
<u>AWS::Logs::DeliverySource</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Logs::Destination</u>	 はい	 はい	 はい
<u>AWS::Logs::Integration</u>	 はい	 はい	 いいえ
<u>AWS::Logs::LogAnomalyDetector</u>	 はい	 はい	 いいえ
<u>AWS::Logs::LogGroup</u>	 はい	 はい	 はい
<u>AWS::Logs::LogStream</u>	 はい	 はい	 はい
<u>AWS::Logs::MetricFilter</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Logs::QueryDefinition</u>	 はい	 はい	 いいえ
<u>AWS::Logs::ResourcePolicy</u>	 はい	 はい	 はい
<u>AWS::Logs::SubscriptionFilter</u>	 はい	 はい	 はい
<u>AWS::Logs::Transformer</u>	 はい	 はい	 いいえ
<u>AWS::LookoutEquipment::InferenceScheduler</u>	 はい	 はい	 いいえ
<u>AWS::LookoutVision::Project</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	IaC ジェネレーター
AWS::M2::Application	 はい	 はい	 いいえ
AWS::M2::Deployment	 はい	 はい	 いいえ
AWS::M2::Environment	 はい	 はい	 はい
AWS::MPA::ApprovalTeam	 はい	 はい	 いいえ
AWS::MPA::IdentitySource	 はい	 はい	 いいえ
AWS::MSK::BatchScramSecret	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MSK::Cluster</u>	 はい	 はい	 はい
<u>AWS::MSK::ClusterPolicy</u>	 はい	 はい	 はい
<u>AWS::MSK::Configuration</u>	 はい	 はい	 はい
<u>AWS::MSK::Replicator</u>	 はい	 はい	 はい
<u>AWS::MSK::ServerlessCluster</u>	 はい	 はい	 はい
<u>AWS::MSK::VpcConnection</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MWAA::Environment</u>	 はい	 はい	 いいえ
<u>AWS::Macie::AllowList</u>	 はい	 はい	 いいえ
<u>AWS::Macie::CustomDataIdentifier</u>	 はい	 はい	 いいえ
<u>AWS::Macie::FindingsFilter</u>	 はい	 はい	 いいえ
<u>AWS::Macie::Session</u>	 はい	 はい	 はい
<u>AWS::ManagedBlockchain::Accessor</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaConnect::Bridge</u>	 はい	 はい	 はい
<u>AWS::MediaConnect::BridgeOutput</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::BridgeSource</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::Flow</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::FlowEntitlement</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::FlowOutput</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaConnect::FlowSource</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::FlowVpcInterface</u>	 はい	 はい	 はい
<u>AWS::MediaConnect::Gateway</u>	 はい	 はい	 はい
<u>AWS::MediaConnect::RouterInput</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::RouterNetworkInterface</u>	 はい	 はい	 いいえ
<u>AWS::MediaConnect::RouterOutput</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaLive::ChannelPlacementGroup</u>	 はい	 はい	 いいえ
<u>AWS::MediaLive::CloudWatchAlarmTemplate</u>	 はい	 はい	 いいえ
<u>AWS::MediaLive::CloudWatchAlarmTemplateGroup</u>	 はい	 はい	 いいえ
<u>AWS::MediaLive::Cluster</u>	 はい	 はい	 いいえ
<u>AWS::MediaLive::EventBridgeRuleTemplate</u>	 はい	 はい	 いいえ
<u>AWS::MediaLive::EventBridgeRuleTemplateGroup</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::MediaLive::Multiplex	 はい	 はい	 いいえ
AWS::MediaLive::Multiplexprogram	 はい	 はい	 いいえ
AWS::MediaLive::Network	 はい	 はい	 いいえ
AWS::MediaLive::SdiSource	 はい	 はい	 いいえ
AWS::MediaLive::SignalMap	 はい	 はい	 いいえ
AWS::MediaPackage::Asset	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaPackage::Channel</u>	 はい	 はい	 はい
<u>AWS::MediaPackage::OriginEndpoint</u>	 はい	 はい	 はい
<u>AWS::MediaPackage::PackagingConfiguration</u>	 はい	 はい	 はい
<u>AWS::MediaPackage::PackagingGroup</u>	 はい	 はい	 はい
<u>AWS::MediaPackageV2::Channel</u>	 はい	 はい	 はい
<u>AWS::MediaPackageV2::ChannelGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaPackageV2::ChannelPolicy</u>	 はい	 はい	 いいえ
<u>AWS::MediaPackageV2::OriginEndpoint</u>	 はい	 はい	 はい
<u>AWS::MediaPackageV2::OriginEndpointPolicy</u>	 はい	 はい	 いいえ
<u>AWS::MediaTailor::Channel</u>	 はい	 はい	 いいえ
<u>AWS::MediaTailor::ChannelPolicy</u>	 はい	 はい	 いいえ
<u>AWS::MediaTailor::LiveSource</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MediaTailor::PlaybackConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::MediaTailor::SourceLocation</u>	 はい	 はい	 はい
<u>AWS::MediaTailor::VodSource</u>	 はい	 はい	 はい
<u>AWS::MemoryDB::ACL</u>	 はい	 はい	 はい
<u>AWS::MemoryDB::Cluster</u>	 はい	 はい	 はい
<u>AWS::MemoryDB::MultiRegionCluster</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::MemoryDB::ParameterGroup</u>	 はい	 はい	 はい
<u>AWS::MemoryDB::SubnetGroup</u>	 はい	 はい	 はい
<u>AWS::MemoryDB::User</u>	 はい	 はい	 はい
<u>AWS::Neptune::DBCluster</u>	 はい	 はい	 はい
<u>AWS::Neptune::DBClusterParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::Neptune::DBInstance</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Neptune::DBParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::Neptune::DBSubnetGroup</u>	 はい	 はい	 いいえ
<u>AWS::Neptune::EventSubscription</u>	 はい	 はい	 いいえ
<u>AWS::NeptuneGraph::Graph</u>	 はい	 はい	 いいえ
<u>AWS::NeptuneGraph::PrivateGraphEndpoint</u>	 はい	 はい	 いいえ
<u>AWS::NetworkFirewall::Firewall</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::NetworkFirewall::FirewallPolicy</u>	 はい	 はい	 はい
<u>AWS::NetworkFirewall::LoggingConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::NetworkFirewall::RuleGroup</u>	 はい	 はい	 はい
<u>AWS::NetworkFirewall::TLSInspectionConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::NetworkFirewall::VpcEndpointAssociation</u>	 はい	 はい	 いいえ
<u>AWS::NetworkManager::ConnectAttachment</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::NetworkManager::ConnectPeer</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::CoreNetwork</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::CoreNetworkPrefixListAssociation</u>	 はい	 はい	 いいえ
<u>AWS::NetworkManager::CustomerGatewayAssociation</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::Device</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::DirectConnectGatewayAttachment</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::NetworkManager::GlobalNetwork</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::Link</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::LinkAssociation</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::Site</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::SiteToSiteVpnAttachment</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::TransitGatewayPeering</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::NetworkManager::TransitGatewayRegistration</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::TransitGatewayRouteTableAttachment</u>	 はい	 はい	 はい
<u>AWS::NetworkManager::VpcAttachment</u>	 はい	 はい	 はい
<u>AWS::Notifications::ChannelAssociation</u>	 はい	 はい	 いいえ
<u>AWS::Notifications::EventRule</u>	 はい	 はい	 いいえ
<u>AWS::Notifications::ManagedNotificationAccountContactAssociation</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Notifications::ManagedNotificationAdditionalChannelAssociation</u>	 はい	 はい	 いいえ
<u>AWS::Notifications::NotificationConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::Notifications::NotificationHub</u>	 はい	 はい	 いいえ
<u>AWS::Notifications::OrganizationalUnitAssociation</u>	 はい	 はい	 いいえ
<u>AWS::NotificationsContacts::EmailContact</u>	 はい	 はい	 いいえ
<u>AWS::ODB::CloudAutonomousVmCluster</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ODB::CloudExadataInfrastructure</u>	 はい	 はい	 いいえ
<u>AWS::ODB::CloudVmCluster</u>	 はい	 はい	 いいえ
<u>AWS::ODB::OdbNetwork</u>	 はい	 はい	 いいえ
<u>AWS::ODB::OdbPeeringConnection</u>	 はい	 はい	 いいえ
<u>AWS::OSIS::Pipeline</u>	 はい	 はい	 はい
<u>AWS::Oam::Link</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Oam::Sink</u>	 はい	 はい	 はい
<u>AWS::ObservabilityAdmin::OrganizationCentralizationRule</u>	 はい	 はい	 いいえ
<u>AWS::ObservabilityAdmin::OrganizationTelemetryRule</u>	 はい	 はい	 いいえ
<u>AWS::ObservabilityAdmin::S3TableIntegration</u>	 はい	 はい	 いいえ
<u>AWS::ObservabilityAdmin::TelemetryPipelines</u>	 はい	 はい	 いいえ
<u>AWS::ObservabilityAdmin::TelemetryRule</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Omics::AnnotationStore</u>	 はい	 はい	 はい
<u>AWS::Omics::ReferenceStore</u>	 はい	 はい	 はい
<u>AWS::Omics::RunGroup</u>	 はい	 はい	 はい
<u>AWS::Omics::SequenceStore</u>	 はい	 はい	 はい
<u>AWS::Omics::VariantStore</u>	 はい	 はい	 はい
<u>AWS::Omics::Workflow</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Omicss::WorkflowVersion</u>	 はい	 はい	 いいえ
<u>AWS::OpenSearchServerless::AccessPolicy</u>	 はい	 はい	 はい
<u>AWS::OpenSearchServerless::Collection</u>	 はい	 はい	 いいえ
<u>AWS::OpenSearchServerless::CollectionGroup</u>	 はい	 はい	 いいえ
<u>AWS::OpenSearchServerless::Index</u>	 はい	 はい	 いいえ
<u>AWS::OpenSearchServerless::LifecyclePolicy</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::OpenSearchServerless::SecurityConfig</u>	 はい	 はい	 はい
<u>AWS::OpenSearchServerless::SecurityPolicy</u>	 はい	 はい	 はい
<u>AWS::OpenSearchServerless::VpcEndpoint</u>	 はい	 はい	 はい
<u>AWS::OpenSearchService::Application</u>	 はい	 はい	 いいえ
<u>AWS::OpenSearchService::Domain</u>	 はい	 はい	 いいえ
<u>AWS::Organizations::Account</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Organizations::Organization</u>	 はい	 はい	 はい
<u>AWS::Organizations::OrganizationalUnit</u>	 はい	 はい	 はい
<u>AWS::Organizations::Policy</u>	 はい	 はい	 はい
<u>AWS::Organizations::ResourcePolicy</u>	 はい	 はい	 はい
<u>AWS::PCAConnectorAD::Connector</u>	 はい	 はい	 はい
<u>AWS::PCAConnectorAD::DirectoryRegistration</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::PCAConnectorAD::ServicePrincipalName</u>	 はい	 はい	 はい
<u>AWS::PCAConnectorAD::Template</u>	 はい	 はい	 はい
<u>AWS::PCAConnectorAD::TemplateGroupAccessControlEntry</u>	 はい	 はい	 はい
<u>AWS::PCAConnectorSCEP::Challenge</u>	 はい	 はい	 いいえ
<u>AWS::PCAConnectorSCEP::Connector</u>	 はい	 はい	 いいえ
<u>AWS::PCS::Cluster</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::PCS::ComputeNodeGroup</u>	 はい	 はい	 いいえ
<u>AWS::PCS::Queue</u>	 はい	 はい	 いいえ
<u>AWS::Panorama::ApplicationInstance</u>	 はい	 はい	 はい
<u>AWS::Panorama::Package</u>	 はい	 はい	 はい
<u>AWS::Panorama::PackageVersion</u>	 はい	 はい	 いいえ
<u>AWS::PaymentCryptography::Alias</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::PaymentCryptography::Key</u>	 はい	 はい	 いいえ
<u>AWS::Personalize::Dataset</u>	 はい	 はい	 はい
<u>AWS::Personalize::DatasetGroup</u>	 はい	 はい	 はい
<u>AWS::Personalize::Schema</u>	 はい	 はい	 はい
<u>AWS::Personalize::Solution</u>	 はい	 はい	 いいえ
<u>AWS::Pinpoint::InAppTemplate</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::Pipes::Pipe	 はい	 はい	 はい
AWS::Proton::EnvironmentAccountConnection	 はい	 はい	 はい
AWS::Proton::EnvironmentTemplate	 はい	 はい	 はい
AWS::Proton::ServiceTemplate	 はい	 はい	 はい
AWS::QBusiness::Application	 はい	 はい	 いいえ
AWS::QBusiness::DataAccessor	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::QBusiness::DataSource</u>	 はい	 はい	 いいえ
<u>AWS::QBusiness::Index</u>	 はい	 はい	 いいえ
<u>AWS::QBusiness::Permission</u>	 はい	 はい	 いいえ
<u>AWS::QBusiness::Plugin</u>	 はい	 はい	 いいえ
<u>AWS::QBusiness::Retriever</u>	 はい	 はい	 いいえ
<u>AWS::QBusiness::WebExperience</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::QLDB::Stream	 はい	 はい	 はい
AWS::QuickSight::Analysis	 はい	 はい	 いいえ
AWS::QuickSight::CustomPermissions	 はい	 はい	 いいえ
AWS::QuickSight::Dashboard	 はい	 はい	 いいえ
AWS::QuickSight::DataSet	 はい	 はい	 はい
AWS::QuickSight::DataSource	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::QuickSight::Folder</u>	 はい	 はい	 いいえ
<u>AWS::QuickSight::RefreshSchedule</u>	 はい	 はい	 はい
<u>AWS::QuickSight::Template</u>	 はい	 はい	 いいえ
<u>AWS::QuickSight::Theme</u>	 はい	 はい	 いいえ
<u>AWS::QuickSight::Topic</u>	 はい	 はい	 はい
<u>AWS::QuickSight::VPCConnection</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RAM::Permission</u>	 はい	 はい	 いいえ
<u>AWS::RAM::ResourceShare</u>	 はい	 はい	 いいえ
<u>AWS::RDS::CustomDBEngineVersion</u>	 はい	 はい	 いいえ
<u>AWS::RDS::DBCluster</u>	 はい	 はい	 はい
<u>AWS::RDS::DBClusterParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::RDS::DBInstance</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RDS::DBParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::RDS::DBProxy</u>	 はい	 はい	 はい
<u>AWS::RDS::DBProxyEndpoint</u>	 はい	 はい	 はい
<u>AWS::RDS::DBProxyTargetGroup</u>	 はい	 はい	 はい
<u>AWS::RDS::DBShardGroup</u>	 はい	 はい	 いいえ
<u>AWS::RDS::DBSubnetGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RDS::EventSubscription</u>	 はい	 はい	 はい
<u>AWS::RDS::GlobalCluster</u>	 はい	 はい	 はい
<u>AWS::RDS::Integration</u>	 はい	 はい	 いいえ
<u>AWS::RDS::OptionGroup</u>	 はい	 はい	 はい
<u>AWS::RTBFabric::InboundExternalLink</u>	 はい	 はい	 いいえ
<u>AWS::RTBFabric::Link</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RTBFabric::OutboundExternalLink</u>	 はい	 はい	 いいえ
<u>AWS::RTBFabric::RequesterGateway</u>	 はい	 はい	 いいえ
<u>AWS::RTBFabric::ResponderGateway</u>	 はい	 はい	 いいえ
<u>AWS::RUM::AppMonitor</u>	 はい	 はい	 はい
<u>AWS::Rbin::Rule</u>	 はい	 はい	 いいえ
<u>AWS::Redshift::Cluster</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Redshift::ClusterParameterGroup</u>	 はい	 はい	 いいえ
<u>AWS::Redshift::ClusterSubnetGroup</u>	 はい	 はい	 いいえ
<u>AWS::Redshift::EndpointAccess</u>	 はい	 はい	 はい
<u>AWS::Redshift::EndpointAuthorization</u>	 はい	 はい	 はい
<u>AWS::Redshift::EventSubscription</u>	 はい	 はい	 いいえ
<u>AWS::Redshift::Integration</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Redshift::ScheduledAction</u>	 はい	 はい	 はい
<u>AWS::RedshiftServerless::Namespace</u>	 はい	 はい	 いいえ
<u>AWS::RedshiftServerless::Snapshot</u>	 はい	 はい	 いいえ
<u>AWS::RedshiftServerless::Workgroup</u>	 はい	 はい	 いいえ
<u>AWS::RefactorSpaces::Application</u>	 はい	 はい	 はい
<u>AWS::RefactorSpaces::Environment</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RefactorSpaces::Route</u>	 はい	 はい	 はい
<u>AWS::RefactorSpaces::Service</u>	 はい	 はい	 はい
<u>AWS::Rekognition::Collection</u>	 はい	 はい	 はい
<u>AWS::Rekognition::Project</u>	 はい	 はい	 はい
<u>AWS::Rekognition::StreamProcessor</u>	 はい	 はい	 はい
<u>AWS::ResilienceHub::App</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ResilienceHub::ResiliencyPolicy</u>	 はい	 はい	 はい
<u>AWS::ResourceExplorer2::DefaultViewAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ResourceExplorer2::Index</u>	 はい	 はい	 はい
<u>AWS::ResourceExplorer2::View</u>	 はい	 はい	 はい
<u>AWS::ResourceGroups::Group</u>	 はい	 はい	 はい
<u>AWS::ResourceGroups::TagSyncTask</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::RolesAnywhere::CRL</u>	 はい	 はい	 はい
<u>AWS::RolesAnywhere::Profile</u>	 はい	 はい	 はい
<u>AWS::RolesAnywhere::TrustAnchor</u>	 はい	 はい	 いいえ
<u>AWS::Route53::CidrCollection</u>	 はい	 はい	 はい
<u>AWS::Route53::DNSSEC</u>	 はい	 はい	 はい
<u>AWS::Route53::HealthCheck</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Route53::HostedZone</u>	 はい	 はい	 はい
<u>AWS::Route53::KeySigningKey</u>	 はい	 はい	 はい
<u>AWS::Route53Profiles::Profile</u>	 はい	 はい	 いいえ
<u>AWS::Route53Profiles::ProfileAssociation</u>	 はい	 はい	 いいえ
<u>AWS::Route53Profiles::ProfileResourceAssociation</u>	 はい	 はい	 いいえ
<u>AWS::Route53RecoveryControl::Cluster</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Route53RecoveryControl::ControlPanel</u>	 はい	 はい	 はい
<u>AWS::Route53RecoveryControl::RoutingControl</u>	 はい	 はい	 はい
<u>AWS::Route53RecoveryControl::SafetyRule</u>	 はい	 はい	 はい
<u>AWS::Route53RecoveryReadiness::Cell</u>	 はい	 はい	 はい
<u>AWS::Route53RecoveryReadiness::ReadinessCheck</u>	 はい	 はい	 はい
<u>AWS::Route53RecoveryReadiness::RecoveryGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::Route53RecoveryReadiness::ResourceSet	 はい	 はい	 はい
AWS::Route53Resolver::FirewallDomainList	 はい	 はい	 いいえ
AWS::Route53Resolver::FirewallRuleGroup	 はい	 はい	 はい
AWS::Route53Resolver::FirewallRuleGroupAssociation	 はい	 はい	 はい
AWS::Route53Resolver::OutpostResolver	 はい	 はい	 はい
AWS::Route53Resolver::ResolverConfig	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Route53Resolver::ResolverDNSSECConfig</u>	 はい	 はい	 はい
<u>AWS::Route53Resolver::ResolverEndpoint</u>	 はい	 はい	 いいえ
<u>AWS::Route53Resolver::ResolverQueryLoggingConfig</u>	 はい	 はい	 はい
<u>AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation</u>	 はい	 はい	 はい
<u>AWS::Route53Resolver::ResolverRule</u>	 はい	 はい	 いいえ
<u>AWS::Route53Resolver::ResolverRuleAssociation</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::S3::AccessGrant</u>	 はい	 はい	 いいえ
<u>AWS::S3::AccessGrantsInstance</u>	 はい	 はい	 いいえ
<u>AWS::S3::AccessGrantsLocation</u>	 はい	 はい	 いいえ
<u>AWS::S3::AccessPoint</u>	 はい	 はい	 はい
<u>AWS::S3::Bucket</u>	 はい	 はい	 はい
<u>AWS::S3::BucketPolicy</u>	 はい	 いいえ	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::S3::MultiRegionAccessPoint	 はい	 はい	 はい
AWS::S3::MultiRegionAccessPointPolicy	 はい	 いいえ	 いいえ
AWS::S3::StorageLens	 はい	 はい	 はい
AWS::S3::StorageLensGroup	 はい	 はい	 いいえ
AWS::S3Express::AccessPoint	 はい	 はい	 いいえ
AWS::S3Express::BucketPolicy	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::S3Express::DirectoryBucket</u>	 はい	 はい	 いいえ
<u>AWS::S3ObjectLambda::AccessPoint</u>	 はい	 はい	 はい
<u>AWS::S3ObjectLambda::AccessPointPolicy</u>	 はい	 はい	 いいえ
<u>AWS::S3Outposts::AccessPoint</u>	 はい	 はい	 はい
<u>AWS::S3Outposts::Bucket</u>	 はい	 はい	 はい
<u>AWS::S3Outposts::BucketPolicy</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::S3Outposts::Endpoint</u>	 はい	 はい	 はい
<u>AWS::S3Tables::Namespace</u>	 はい	 はい	 いいえ
<u>AWS::S3Tables::Table</u>	 はい	 はい	 いいえ
<u>AWS::S3Tables::TableBucket</u>	 はい	 はい	 いいえ
<u>AWS::S3Tables::TableBucketPolicy</u>	 はい	 はい	 いいえ
<u>AWS::S3Tables::TablePolicy</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::S3Vectors::Index</u>	 はい	 はい	 いいえ
<u>AWS::S3Vectors::VectorBucket</u>	 はい	 はい	 いいえ
<u>AWS::S3Vectors::VectorBucketPolicy</u>	 はい	 はい	 いいえ
<u>AWS::SES::ConfigurationSet</u>	 はい	 はい	 いいえ
<u>AWS::SES::ConfigurationSetEventDestination</u>	 いいえ	 はい	 いいえ
<u>AWS::SES::ContactList</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SES::DedicatedIpPool</u>	 はい	 はい	 はい
<u>AWS::SES::EmailIdentity</u>	 はい	 はい	 はい
<u>AWS::SES::MailManagerAddonInstance</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerAddonSubscription</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerAddressList</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerArchive</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SES::MailManagerIngressPoint</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerRelay</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerRuleSet</u>	 はい	 はい	 いいえ
<u>AWS::SES::MailManagerTrafficPolicy</u>	 はい	 はい	 いいえ
<u>AWS::SES::MultiRegionEndpoint</u>	 はい	 はい	 いいえ
<u>AWS::SES::Template</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SES::Tenant</u>	 はい	 はい	 いいえ
<u>AWS::SES::VdmAttributes</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::ConfigurationSet</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::OptOutList</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::PhoneNumber</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::Pool</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SMSVOICE::ProtectConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::SMSVOICE::SenderId</u>	 はい	 はい	 いいえ
<u>AWS::SNS::Subscription</u>	 はい	 はい	 いいえ
<u>AWS::SNS::Topic</u>	 はい	 はい	 はい
<u>AWS::SNS::TopicInlinePolicy</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::SQS::Queue	 はい	 はい	 はい
AWS::SQS::QueueInlinePolicy	 はい	 はい	 いいえ
AWS::SSM::Association	 はい	 はい	 はい
AWS::SSM::Document	 はい	 いいえ	 いいえ
AWS::SSM::MaintenanceWindowTarget	 はい	 はい	 いいえ
AWS::SSM::MaintenanceWindowTask	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SSM::Parameter</u>	 はい	 はい	 いいえ
<u>AWS::SSM::PatchBaseline</u>	 はい	 はい	 いいえ
<u>AWS::SSM::ResourceDataSync</u>	 はい	 はい	 いいえ
<u>AWS::SSM::ResourcePolicy</u>	 はい	 はい	 はい
<u>AWS::SSMContacts::Contact</u>	 はい	 はい	 はい
<u>AWS::SSMContacts::ContactChannel</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SSMContacts::Plan</u>	 はい	 はい	 いいえ
<u>AWS::SSMContacts::Rotation</u>	 はい	 はい	 いいえ
<u>AWS::SSMGuiConnect::Preferences</u>	 はい	 はい	 いいえ
<u>AWS::SSMIncidents::ReplicationSet</u>	 はい	 はい	 はい
<u>AWS::SSMIncidents::ResponsePlan</u>	 はい	 はい	 はい
<u>AWS::SSMQuickSetup::ConfigurationManager</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SSMQuickSetup::LifecycleAutomation</u>	 はい	 はい	 いいえ
<u>AWS::SSO::Application</u>	 はい	 はい	 いいえ
<u>AWS::SSO::ApplicationAssignment</u>	 はい	 はい	 いいえ
<u>AWS::SSO::Assignment</u>	 はい	 はい	 いいえ
<u>AWS::SSO::Instance</u>	 はい	 はい	 いいえ
<u>AWS::SSO::InstanceAccessControlAttributeConfiguration</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SageMaker::App</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::AppImageConfig</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::Cluster</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::DataQualityJobDefinition</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::Device</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::DeviceFleet</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SageMaker::Domain</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::Endpoint</u>	 いいえ	 はい	 いいえ
<u>AWS::SageMaker::FeatureGroup</u>	 はい	 はい	 はい
<u>AWS::SageMaker::Image</u>	 はい	 はい	 はい
<u>AWS::SageMaker::ImageVersion</u>	 はい	 はい	 はい
<u>AWS::SageMaker::InferenceComponent</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SageMaker::InferenceExperiment</u>	 はい	 はい	 はい
<u>AWS::SageMaker::MlflowTrackingServer</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::ModelBiasJobDefinition</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::ModelCard</u>	 はい	 はい	 はい
<u>AWS::SageMaker::ModelExplainabilityJobDefinition</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::ModelPackage</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SageMaker::ModelPackageGroup</u>	 はい	 はい	 はい
<u>AWS::SageMaker::ModelQualityJobDefinition</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::MonitoringSchedule</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::PartnerApp</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::Pipeline</u>	 はい	 はい	 はい
<u>AWS::SageMaker::ProcessingJob</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SageMaker::Project</u>	 はい	 はい	 はい
<u>AWS::SageMaker::Space</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::StudioLifecycleConfig</u>	 はい	 はい	 いいえ
<u>AWS::SageMaker::UserProfile</u>	 はい	 はい	 いいえ
<u>AWS::Scheduler::Schedule</u>	 はい	 はい	 はい
<u>AWS::Scheduler::ScheduleGroup</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SecretsManager::ResourcePolicy</u>	 はい	 いいえ	 いいえ
<u>AWS::SecretsManager::RotationSchedule</u>	 はい	 はい	 いいえ
<u>AWS::SecretsManager::Secret</u>	 はい	 はい	 はい
<u>AWS::SecretsManager::SecretTargetAttachment</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::AggregatorV2</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::AutomationRule</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::SecurityHub::AutomationRuleV2	 はい	 はい	 いいえ
AWS::SecurityHub::ConfigurationPolicy	 はい	 はい	 いいえ
AWS::SecurityHub::ConnectorV2	 はい	 はい	 いいえ
AWS::SecurityHub::DelegatedAdmin	 はい	 はい	 いいえ
AWS::SecurityHub::FindingAggregator	 はい	 はい	 いいえ
AWS::SecurityHub::Hub	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SecurityHub::HubV2</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::Insight</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::OrganizationConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::PolicyAssociation</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::ProductSubscription</u>	 はい	 はい	 いいえ
<u>AWS::SecurityHub::SecurityControl</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SecurityHub::Standard</u>	 はい	 はい	 いいえ
<u>AWS::SecurityLake::AwsLogSource</u>	 はい	 はい	 いいえ
<u>AWS::SecurityLake::DataLake</u>	 はい	 はい	 いいえ
<u>AWS::SecurityLake::Subscriber</u>	 はい	 はい	 いいえ
<u>AWS::SecurityLake::SubscriberNotification</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::CloudFormationProvisionedProduct</u>	 はい	 いいえ	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ServiceCatalog::LaunchNotificationConstraint</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::LaunchTemplateConstraint</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::PortfolioPrincipalAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::PortfolioProductAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::PortfolioShare</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::ResourceUpdateConstraint</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ServiceCatalog::ServiceAction</u>	 はい	 はい	 はい
<u>AWS::ServiceCatalog::ServiceActionAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::TagOption</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalog::TagOptionAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalogAppRegistry::Application</u>	 はい	 はい	 はい
<u>AWS::ServiceCatalogAppRegistry::AttributeGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation</u>	 はい	 はい	 いいえ
<u>AWS::ServiceCatalogAppRegistry::ResourceAssociation</u>	 はい	 いいえ	 いいえ
<u>AWS::Shield::DRTAccess</u>	 はい	 はい	 いいえ
<u>AWS::Shield::ProactiveEngagement</u>	 はい	 はい	 いいえ
<u>AWS::Shield::Protection</u>	 はい	 はい	 いいえ
<u>AWS::Shield::ProtectionGroup</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Signer::ProfilePermission</u>	 はい	 はい	 いいえ
<u>AWS::Signer::SigningProfile</u>	 はい	 はい	 はい
<u>AWS::SimSpaceWeaver::Simulation</u>	 はい	 はい	 はい
<u>AWS::StepFunctions::Activity</u>	 はい	 はい	 いいえ
<u>AWS::StepFunctions::StateMachine</u>	 はい	 はい	 はい
<u>AWS::StepFunctions::StateMachineAlias</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::StepFunctions::StateMachineVersion</u>	 はい	 はい	 はい
<u>AWS::SupportApp::AccountAlias</u>	 はい	 はい	 はい
<u>AWS::SupportApp::SlackChannelConfiguration</u>	 はい	 はい	 はい
<u>AWS::SupportApp::SlackWorkspaceConfiguration</u>	 はい	 はい	 はい
<u>AWS::Synthetics::Canary</u>	 はい	 はい	 はい
<u>AWS::Synthetics::Group</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::SystemsManagerSAP::Application</u>	 はい	 はい	 いいえ
<u>AWS::Timestream::Database</u>	 はい	 はい	 はい
<u>AWS::Timestream::InfluxDBInstance</u>	 はい	 はい	 いいえ
<u>AWS::Timestream::ScheduledQuery</u>	 はい	 はい	 いいえ
<u>AWS::Timestream::Table</u>	 はい	 はい	 はい
<u>AWS::Transfer::Agreement</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Transfer::Certificate</u>	 はい	 はい	 はい
<u>AWS::Transfer::Connector</u>	 はい	 はい	 はい
<u>AWS::Transfer::Profile</u>	 はい	 はい	 はい
<u>AWS::Transfer::Server</u>	 はい	 はい	 いいえ
<u>AWS::Transfer::User</u>	 はい	 はい	 いいえ
<u>AWS::Transfer::WebApp</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::Transfer::Workflow	 はい	 はい	 はい
AWS::VerifiedPermissions::IdentitySource	 はい	 はい	 はい
AWS::VerifiedPermissions::Policy	 はい	 はい	 はい
AWS::VerifiedPermissions::PolicyStore	 はい	 はい	 はい
AWS::VerifiedPermissions::PolicyTemplate	 はい	 はい	 はい
AWS::VoiceID::Domain	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::VpcLattice::AccessLogSubscription</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::AuthPolicy</u>	 はい	 はい	 いいえ
<u>AWS::VpcLattice::DomainVerification</u>	 はい	 はい	 いいえ
<u>AWS::VpcLattice::Listener</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::ResourceConfiguration</u>	 はい	 はい	 いいえ
<u>AWS::VpcLattice::ResourceGateway</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::VpcLattice::ResourcePolicy</u>	 はい	 はい	 いいえ
<u>AWS::VpcLattice::Rule</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::Service</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::ServiceNetwork</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::ServiceNetworkResourceAssociation</u>	 はい	 はい	 いいえ
<u>AWS::VpcLattice::ServiceNetworkServiceAssociation</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::VpcLattice::ServiceNetworkVpcAssociation</u>	 はい	 はい	 はい
<u>AWS::VpcLattice::TargetGroup</u>	 はい	 はい	 はい
<u>AWS::WAFv2::IPSet</u>	 はい	 はい	 はい
<u>AWS::WAFv2::LoggingConfiguration</u>	 はい	 はい	 はい
<u>AWS::WAFv2::RegexPatternSet</u>	 はい	 はい	 はい
<u>AWS::WAFv2::RuleGroup</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
AWS::WAFv2::WebACL	 はい	 はい	 はい
AWS::WAFv2::WebACLAssociation	 はい	 はい	 いいえ
AWS::Wisdom::AIAgent	 はい	 はい	 いいえ
AWS::Wisdom::AIAgentVersion	 はい	 はい	 いいえ
AWS::Wisdom::AIGuardrail	 はい	 はい	 いいえ
AWS::Wisdom::AIGuardrailVersion	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Wisdom::AIPrompt</u>	 はい	 はい	 いいえ
<u>AWS::Wisdom::AIPromptVersion</u>	 はい	 はい	 いいえ
<u>AWS::Wisdom::Assistant</u>	 はい	 はい	 はい
<u>AWS::Wisdom::AssistantAssociation</u>	 はい	 はい	 はい
<u>AWS::Wisdom::KnowledgeBase</u>	 はい	 はい	 はい
<u>AWS::Wisdom::MessageTemplate</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::Wisdom::MessageTemplateVersion</u>	 はい	 はい	 いいえ
<u>AWS::Wisdom::QuickResponse</u>	 はい	 はい	 いいえ
<u>AWS::WorkSpaces::ConnectionAlias</u>	 はい	 はい	 いいえ
<u>AWS::WorkSpaces::WorkspacesPool</u>	 はい	 はい	 いいえ
<u>AWS::WorkSpacesThinClient::Environment</u>	 はい	 はい	 いいえ
<u>AWS::WorkSpacesWeb::BrowserSettings</u>	 はい	 はい	 はい

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::WorkSpacesWeb::DataProtectionSettings</u>	 はい	 はい	 いいえ
<u>AWS::WorkSpacesWeb::IdentityProvider</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::IpAccessSettings</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::NetworkSettings</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::Portal</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::SessionLogger</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::WorkSpacesWeb::TrustStore</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::UserAccessLoggingSettings</u>	 はい	 はい	 はい
<u>AWS::WorkSpacesWeb::UserSettings</u>	 はい	 はい	 はい
<u>AWS::WorkspacesInstances::Volume</u>	 はい	 はい	 いいえ
<u>AWS::WorkspacesInstances::VolumeAssociation</u>	 はい	 はい	 いいえ
<u>AWS::WorkspacesInstances::WorkspaceInstance</u>	 はい	 はい	 いいえ

リソース	[Import] (インポート)	ドリフト検出	laC ジェネレーター
<u>AWS::XRay::Group</u>	 はい	 はい	 はい
<u>AWS::XRay::ResourcePolicy</u>	 はい	 はい	 はい
<u>AWS::XRay::SamplingRule</u>	 はい	 はい	 いいえ
<u>AWS::XRay::TransactionSearchConfig</u>	 はい	 はい	 いいえ

クイック作成リンクを使用して CloudFormation スタックを作成する

クイック作成リンクは、CloudFormation コンソールの URL から直接 CloudFormation スタックを起動する効率的な方法を提供します。テンプレートの URL、スタック名、テンプレートパラメータを URL クエリパラメータとして指定することで、[スタックを作成] ページにあらかじめ値を入力し、スタックの作成を迅速に進めることができます。これにより、ウィザードのページ数や、ユーザーの入力量が減り、容易にスタックを作成できるようになります。また、同一テンプレートに対して異なる値を指定した URL を複数作成することができるため、効果的にテンプレートを再利用することもできます。

URL 形式

クイック作成リンクは次の URL 形式に従います。

```
https://region-code.console.aws.amazon.com/cloudformation/home?region=region-code#/  
stacks/create/review  
?templateURL=TemplateURL  
&stackName=StackName  
&param_parameterName=parameterValue
```

CloudFormation では、次の URL のクエリパラメータをサポートしています。

テンプレートの URL

必須。templateURL パラメータは、Amazon S3 バケットにあるスタックテンプレートの URL を指定します。署名付き S3 URL によるアクセスエラーを回避するには、URL を URL エンコードしておく必要があります。

サポートされている S3 URL 形式:

- `https://s3.region-code.amazonaws.com/bucket-name/template-name`
- `https://bucket-name.s3.region-code.amazonaws.com/template-name`
- `https://s3-region-code.amazonaws.com/bucket-name/template-name` (従来の形式)

スタック名

オプション。stackName パラメータを使用して、作成する CloudFormation スタックの名前を指定します。スタック名には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字はアルファベット文字である必要があります。また、128 文字より長くすることはできません。

テンプレートパラメータ

オプション。スタックテンプレート内のパラメータが NoEcho 型でない場合は、URL クエリ文字列で param_*parameterName* 型の形式を使用してください。URL パラメータには、プレフィックス (param_) を含む必要があり、パラメータ名のセグメントは、テンプレートのパラメータ名と完全に一致する必要があります。例: param_DBName。

CloudFormation では、テンプレートに存在しないパラメータ、および NoEcho プロパティを true タイプ (通常はユーザー名とパスワード) に設定して定義したすべてのパラメータが無視さ

れます。URL パラメータは、テンプレートで指定されているデフォルト値を上書きします。必要に応じて、任意の数のパラメータを含めることができます。

⚠ Important

機密情報は、CloudFormation テンプレートに直接埋め込むのではなく、スタックテンプレートの動的パラメータを使用して CloudFormation の外部 (AWS Systems Manager パラメータストアや AWS Secrets Manager など) に保存して管理した上で 参照することをお勧めします。

詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティスを参照してください。

すべてのクエリパラメータ名は、大文字と小文字が区別されます。ユーザーは、スタックを作成する前に、コンソールでこれらの値を上書きできます。

例

次の例は、「[基本的な WordPress \(単一インスタンス\)](#)」のサンプルテンプレートに基づいています。クエリ文字列には、必須の `templateURL` パラメータに加え、`stackName`、`DBName`、`InstanceType`、および `KeyName` パラメータなどがあります。

次の URL には、読みやすくするために改行が追加されています。

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/
create/review
?templateURL=https://s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template
&stackName=MyWPBlog
&param_DBName=mywpblog
&param_InstanceType=t2.medium
```

次の URL には、前の例と同じパラメータが含まれていますが、改行は削除されています。これは実際の URL 形式です。

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?
region=us-east-2#/stacks/create/review?templateURL=https://
s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template&stackName=MyWPBlog&param_DBName=mywpblog&param_InstanceType=
```

クイック作成リンクを使用してスタックを作成する

クイック作成リンクを開くと、CloudFormation コンソールに移動します。[スタックのクイック作成] ページが直接開き、指定された値がパラメータに自動的に使用されます。

クイック作成リンクを使用してスタックを作成するには (コンソール)

1. [スタックのクイック作成] ページの [テンプレート]、[テンプレート URL] で、テンプレート URL が正しいことを確認します。
2. [テンプレートを表示] セクションを展開して、テンプレートを確認します。
3. [スタック名] で、事前入力されたスタック名を確認します。
4. [パラメータ] セクションを確認します。事前入力されたパラメータ値が正しいことを確認します。URL で指定されなかった必須パラメータを入力します。必要に応じて、事前入力された値を変更します。
5. 以下の設定を行うことができます。
 - [タグ] — キーと値のペアを使用してリソースを整理します。
 - [アクセス許可] — スタックオペレーションの IAM サービスロールを選択します。
 - [スタックの失敗オプション] — リソースをロールバック (デフォルト) または保持することを選択します。
 - [スタックポリシー] — リソースの更新アクセス許可を制御します。
 - [ロールバック設定] — CloudWatch アラームベースのロールバックを設定します。
 - [通知オプション] — スタックイベントの Amazon SNS 通知を設定します。
 - [スタックの作成オプション] — 最大スタック作成時間を定義し、誤って削除されないように終了保護を有効にします。

詳細については、「[スタックオプションを設定する](#)」を参照してください。

6. [機能] では、必要な確認を実行します。たとえば、テンプレートに IAM リソースが含まれる場合は、[このテンプレートで IAM リソースが作成される可能性があることを承認します] を選択して、テンプレート内で IAM リソースの使用を明示します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
7. (オプション) 変更セットを作成して、スタックを作成する前にその設定をプレビューすることができます。[変更セットの作成] を選択し、指示に従います。詳細については、「[スタックの設定をプレビューする](#)」を参照してください。

8. 準備ができたら、[スタックの作成] を選択してスタックを起動し、[イベント] タブでスタックの作成の進行状況をモニタリングします。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例

次のコマンドライン例は、AWS CLI と PowerShell を使用して個々の CloudFormation アクションを実行する方法を示しています。これらの例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「AWS CLI コマンドリファレンス」の「[cloudformation](#)」を参照してください。

このガイドの例では、バックスラッシュ (\) の規則を使用して、長いコマンドラインが次の行に続くことを示します。

トピック

- [スタックの更新をキャンセルする](#)
- [更新のロールバックを続ける](#)
- [Stack を作成する](#)
- [変換が含まれるスタックを作成する](#)
- [スタックを削除する](#)
- [スタックイベントを記述する](#)
- [スタックリソースを記述する](#)
- [スタックリソースを記述する](#)
- [スタックを記述する](#)
- [テンプレートを取得する](#)
- [スタックリソースを一覧表示する](#)
- [スタックを一覧表示する](#)
- [スタックを更新する](#)
- [テンプレートを検証する](#)
- [AWS CLI を使用してローカルアーティファクトを S3 バケットにアップロードする](#)

スタックの更新をキャンセルする

[cancel-update-stack](#) コマンドを使用して、スタックの更新をキャンセルします。詳細については、「[スタック更新をキャンセルする](#)」を参照してください。

CLI

AWS CLI

進行中のスタック更新をキャンセルする方法

以下の `cancel-update-stack` コマンドは、`myteststack` スタックで行われているスタック更新をキャンセルします。

```
aws cloudformation cancel-update-stack --stack-name myteststack
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CancelUpdateStack](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックで行われている更新をキャンセルします。

```
Stop-CFNUUpdateStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[CancelUpdateStack](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックで行われている更新をキャンセルします。

```
Stop-CFNUUpdateStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[CancelUpdateStack](#)」を参照してください。

更新のロールバックを続ける

[continue-update-rollback](#) コマンドを使用して更新のロールバックを続行します。詳細については、「[更新のロールバックを続ける](#)」を参照してください。

CLI

AWS CLI

更新のロールバックを再試行する

以下の `continue-update-rollback` の例は、前回失敗したスタック更新からのロールバック操作を再開します。

```
aws cloudformation continue-update-rollback \  
  --stack-name my-stack
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ContinueUpdateRollback](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックのロールバックを続行します。このスタックの状態は「UPDATE_ROLLBACK_FAILED」になっている必要があります。続行されたロールバックが成功すると、スタックの状態が「UPDATE_ROLLBACK_COMPLETE」になります。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[ContinueUpdateRollback](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックのロールバックを続行します。このスタックの状態は「UPDATE_ROLLBACK_FAILED」になっている必要があります。続行されたロールバックが成功すると、スタックの状態が「UPDATE_ROLLBACK_COMPLETE」になります。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[ContinueUpdateRollback](#)」を参照してください。

Stack を作成する

[create-stack](#) コマンドを使用してスタックを作成します。スタック名、有効なテンプレートの場所、入力パラメーターが必要です。パラメータキーの名前では大文字と小文字が区別されます。パラメータキーの名前を誤って入力した場合、CloudFormation はスタックを作成せず、テンプレートにそのパラメータが含まれていないことを報告します。

以下の例は、指定された名前、テンプレート、および入力パラメータを使用して新しいスタックを作成する方法を示しています。

CLI

AWS CLI

AWS CloudFormation スタックを作成する方法

次の `create-stack` コマンドは、`sampletemplate.json` テンプレートを使用して「myteststack」という名称のスタックを作成します。

```
aws cloudformation create-stack --stack-name myteststack --template-body file://sampletemplate.json --parameters ParameterKey=KeyPairName,ParameterValue=TestKey  
ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

出力:

```
{  
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"  
}
```

詳細については、「AWS CloudFormation ユーザーガイド」の「スタック」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateStack](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗しても、スタックはロールバックされません。

```
New-CFNStack -StackName "myStack" `
              -TemplateBody "{TEMPLATE CONTENT HERE}" `
              -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                             @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
              -DisableRollback $true
```

例 2: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
              -TemplateBody "{TEMPLATE CONTENT HERE}" `
              -Parameter @( $p1, $p2 ) `
              -OnFailure "ROLLBACK"
```

例 3: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、Amazon S3 URL から取得されます (「PK1」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」はパラメータの値を表しま

す)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます (-DisableRollback \$false を指定する場合と同じです)。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 4: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、Amazon S3 URL から取得されます (「PK1」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」はパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます (-DisableRollback \$false を指定する場合と同じです)。指定された通知 AEN は、公開されたスタック関連のイベントを受け取ります。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
              -NotificationARN @( "arn1", "arn2" )
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[CreateStack](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗しても、スタックはロールバックされません。

```
New-CFNStack -StackName "myStack" `
              -TemplateBody "{TEMPLATE CONTENT HERE}" `
              -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                             @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
              -DisableRollback $true
```

例 2: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( $p1, $p2 ) `
    -OnFailure "ROLLBACK"
```

例 3: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、Amazon S3 URL から取得されます (「PK1」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」はパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます (-DisableRollback \$false を指定する場合と同じです)。

```
New-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 4: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、Amazon S3 URL から取得されます (「PK1」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」はパラメータの値を表します)。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。スタックの作成に失敗すると、スタックがロールバックされます (-DisableRollback \$false を指定する場合と同じです)。指定された通知 AEN は、公開されたスタック関連のイベントを受け取ります。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
              -NotificationARN @( "arn1", "arn2" )
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[CreateStack](#)」を参照してください。

Note

AWS CLI `--template-url` オプションを使用して、Amazon S3 または AWS Systems Manager のテンプレートファイルの場所を指定できます。
Amazon S3 の場合、URL は `https://` で始まる必要があります。S3 静的ウェブサイトの URL はサポートされていません。

```
--template-url https://s3.region-code.amazonaws.com/bucket-name/template-name
```

AWS Systems Manager には次の形式を使用します。

```
--template-url "ssm-doc://arn:aws:ssm:region-code:account-id:document/document-name"
```

変換が含まれるスタックを作成する

[deploy](#) コマンドを使用して、変換が含まれるスタックを作成します。変換が含まれるテンプレートからスタックを作成するときには、変更セットを使用する必要があります。deploy コマンドは、2つのステップ (変更セットを作成して実行する) を 1 つのコマンドにまとめます。

AWS CLI

以下の deploy コマンドは、指定された名前、テンプレート、および入力パラメータを使用してスタックを作成します。

```
aws cloudformation deploy --stack-name myteststack \
  --template /path_to_template/my-template.json \
  --parameter-overrides Key1=Value1 Key2=Value2
```


スタックを削除する

[delete-stack](#) コマンドを使用してスタックを削除します。詳細については、「[スタックを削除する](#)」を参照してください。

CLI

AWS CLI

スタックを削除する方法

以下の delete-stack 例は、指定されたスタックを削除します。

```
aws cloudformation delete-stack \  
  --stack-name my-stack
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteStack](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックを削除します。

```
Remove-CFNStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[DeleteStack](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックを削除します。

```
Remove-CFNStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[DeleteStack](#)」を参照してください。

削除が失敗して `DELETE_FAILED` 状態が返された場合は、`--deletion-mode` オプションを強制的に使用することでスタックの削除を選択できます。deletion-mode で使用できる値は次のとおりです:

- `STANDARD`: スタックを通常どおり削除します。これはデフォルトの削除モードです。
- `FORCE_DELETE_STACK`: スタックを削除し、削除に失敗したすべてのリソースをスキップします。

AWS CLI

次の `delete-stack` コマンドは、`deletion-mode` パラメータの `FORCE_DELETE_STACK` 値を使用して *myteststack* スタックを強制的に削除します。

```
aws cloudformation delete-stack --stack-name myteststack \
  --deletion-mode FORCE_DELETE_STACK
```

このコマンドでは何も出力されません。

`FORCE_DELETE_STACK` を使用した後は、`list-stack-resources` コマンドを使用して、スタックの削除プロセス中にスキップされたリソースを一覧表示することができます。保持されたリソースには `DELETE_SKIPPED` ステータスが表示されます。詳細については、「[スタックリソースを一覧表示する](#)」を参照してください。

スタックイベントを記述する

[describe-stack-events](#) コマンドを使用して、スタックイベントを記述します。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

CLI

AWS CLI

スタックイベントを記述する方法

次の `describe-stack-events` の例は、指定されたスタックの最新イベントを 2 件表示します。

```
aws cloudformation describe-stack-events \
  --stack-name my-stack \
  --max-items 2
```

```
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "EventId": "4e1516d0-e4d6-xmpl-b94f-0a51958a168c",
      "StackName": "my-stack",
      "LogicalResourceId": "my-stack",
      "PhysicalResourceId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2019-10-02T05:34:29.556Z",
      "ResourceStatus": "UPDATE_COMPLETE"
    },
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "EventId": "4dd3c810-e4d6-xmpl-bade-0aaf8b31ab7a",
      "StackName": "my-stack",
      "LogicalResourceId": "my-stack",
      "PhysicalResourceId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2019-10-02T05:34:29.127Z",
      "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
    }
  ],
  "NextToken": "eyJ0ZXh0VG9XMPLiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeStackEvents](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックに関するすべてのスタック関連イベントを返します。

```
Get-CFNStackEvent -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[DescribeStackEvents](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックに関するすべてのスタック関連イベントを返します。

```
Get-CFNStackEvent -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[DescribeStackEvents](#)」を参照してください。

スタックリソースを記述する

CLI

AWS CLI

リソースに関する情報を取得する方法

次の `describe-stack-resource` の例は、指定されたスタック内にある「MyFunction」という名称のリソースに関する詳細情報を表示します。

```
aws cloudformation describe-stack-resource \  
  --stack-name MyStack \  
  --logical-resource-id MyFunction
```

出力:

```
{  
  "StackResourceDetail": {  
    "StackName": "MyStack",  
    "StackId": "arn:aws:cloudformation:us-east-2:123456789012:stack/MyStack/  
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",  
    "LogicalResourceId": "MyFunction",  
    "PhysicalResourceId": "my-function-SEZV4XMPL4S5",  
    "ResourceType": "AWS::Lambda::Function",  
    "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",  
    "ResourceStatus": "UPDATE_COMPLETE",  
    "Metadata": "{}",  
    "DriftInformation": {
```

```
        "StackResourceDriftStatus": "IN_SYNC"
    }
}
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeStackResource](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 論理 ID 「MyDBInstance」を使用して、指定されたスタックに関連付けられているテンプレートで識別されたリソースの説明を返します。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[DescribeStackResource](#)」を参照してください。

Tools for PowerShell V5

例 1: 論理 ID 「MyDBInstance」を使用して、指定されたスタックに関連付けられているテンプレートで識別されたリソースの説明を返します。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[DescribeStackResource](#)」を参照してください。

スタックリソースを記述する

CLI

AWS CLI

リソースに関する情報を取得する方法

次の `describe-stack-resources` 例は、指定されたスタック内にあるリソースの詳細情報を表示します。

```
aws cloudformation describe-stack-resources \
  --stack-name my-stack
```

出力:

```
{
  "StackResources": [
    {
      "StackName": "my-stack",
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "LogicalResourceId": "bucket",
      "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
      "ResourceType": "AWS::S3::Bucket",
      "Timestamp": "2019-10-02T04:34:11.345Z",
      "ResourceStatus": "CREATE_COMPLETE",
      "DriftInformation": {
        "StackResourceDriftStatus": "IN_SYNC"
      }
    },
    {
      "StackName": "my-stack",
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "LogicalResourceId": "function",
      "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
      "ResourceType": "AWS::Lambda::Function",
      "Timestamp": "2019-10-02T05:34:27.989Z",
      "ResourceStatus": "UPDATE_COMPLETE",
      "DriftInformation": {
        "StackResourceDriftStatus": "IN_SYNC"
      }
    },
    {
      "StackName": "my-stack",
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "LogicalResourceId": "functionRole",
      "PhysicalResourceId": "my-functionRole-HIZXMPLE0M9E",
      "ResourceType": "AWS::IAM::Role",
      "Timestamp": "2019-10-02T04:34:06.350Z",
      "ResourceStatus": "CREATE_COMPLETE",
      "DriftInformation": {
```

```
        "StackResourceDriftStatus": "IN_SYNC"
    }
}
]
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeStackResources](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックに関連付けられている最大 100 個のリソースの AWS リソース説明を返します。スタックに関連付けられているすべてのリソースの詳細情報を取得するには、Get-CFNStackResourceSummary を使用します。これは、結果の手動ページングもサポートしています。

```
Get-CFNStackResourceList -StackName "myStack"
```

例 2: 論理 ID 「Ec2Instance」を使用して、指定されたスタックに関連付けられているテンプレートで識別された Amazon EC2 インスタンスの説明を返します。

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

例 3: インスタンス ID 「i-123456」で識別される Amazon EC2 インスタンスが含まれるスタックに関連付けられた、最大 100 個のリソースの説明を返します。スタックに関連付けられているすべてのリソースの詳細情報を取得するには、Get-CFNStackResourceSummary を使用します。これは、結果の手動ページングもサポートしています。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

例 4: スタックのテンプレートで論理 ID 「Ec2Instance」によって識別される Amazon EC2 インスタンスの説明を返します。スタックは、スタックに含まれるリソースの物理リソース ID を使用して識別され、この場合はインスタンス ID が 「i-123456」 の Amazon EC2 インスタンスも含まれます。テンプレートのコンテンツによっては、スタックの識別に異なる物理リソース (Amazon S3 バケットなど) を使用することもできます。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId  
"Ec2Instance"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[DescribeStackResources](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックに関連付けられている最大 100 個のリソースの AWS リソース説明を返します。スタックに関連付けられているすべてのリソースの詳細情報を取得するには、Get-CFNStackResourceSummary を使用します。これは、結果の手動ページングもサポートしています。

```
Get-CFNStackResourceList -StackName "myStack"
```

例 2: 論理 ID 「Ec2Instance」を使用して、指定されたスタックに関連付けられているテンプレートで識別された Amazon EC2 インスタンスの説明を返します。

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

例 3: インスタンス ID 「i-123456」で識別される Amazon EC2 インスタンスが含まれるスタックに関連付けられた、最大 100 個のリソースの説明を返します。スタックに関連付けられているすべてのリソースの詳細情報を取得するには、Get-CFNStackResourceSummary を使用します。これは、結果の手動ページングもサポートしています。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

例 4: スタックのテンプレートで論理 ID 「Ec2Instance」によって識別される Amazon EC2 インスタンスの説明を返します。スタックは、スタックに含まれるリソースの物理リソース ID を使用して識別され、この場合はインスタンス ID が 「i-123456」 の Amazon EC2 インスタンスも含まれます。テンプレートのコンテンツによっては、スタックの識別に異なる物理リソース (Amazon S3 バケットなど) を使用することもできます。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId  
"Ec2Instance"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[DescribeStackResources](#)」を参照してください。

スタックを記述する

CLI

AWS CLI

AWS CloudFormation スタックを記述する方法

以下の `describe-stacks` 例は、`myteststack` スタックの概要情報を表示します。

```
aws cloudformation describe-stacks --stack-name myteststack
```

出力:

```
{
  "Stacks": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
      "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
template showing how to create a publicly accessible S3 bucket. **WARNING** This
template creates an S3 bucket. You will be billed for the AWS resources used if
you create a stack from this template.",
      "Tags": [],
      "Outputs": [
        {
          "Description": "Name of S3 bucket to hold website content",
          "OutputKey": "BucketName",
          "OutputValue": "myteststack-s3bucket-jssofilzie2w"
        }
      ],
      "StackStatusReason": null,
      "CreationTime": "2013-08-23T01:02:15.422Z",
      "Capabilities": [],
      "StackName": "myteststack",
      "StackStatus": "CREATE_COMPLETE",
      "DisableRollback": false
    }
  ]
}
```

詳細については、「AWS CloudFormation ユーザーガイド」の「スタック」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeStacks](#)」を参照してください。

Go

SDK for Go V2

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#)で完全な例を見つけて、設定と実行の方法を確認してください。

```
import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
    string) StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &cloudformation.DescribeStacksInput{
            StackName: aws.String(stackName),
        })
    if err != nil || len(output.Stacks) == 0 {
```

```
log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
stackName, err)
}
stackOutputs := StackOutputs{}
for _, out := range output.Stacks[0].Outputs {
    stackOutputs[*out.OutputKey] = *out.OutputValue
}
return stackOutputs
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[DescribeStacks](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: ユーザーのスタックのすべてを記述するスタックインスタンスのコレクションを返します。

```
Get-CFNStack
```

例 2: 指定されたスタックを記述するスタックインスタンスを返します。

```
Get-CFNStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[DescribeStacks](#)」を参照してください。

Tools for PowerShell V5

例 1: ユーザーのスタックのすべてを記述するスタックインスタンスのコレクションを返します。

```
Get-CFNStack
```

例 2: 指定されたスタックを記述するスタックインスタンスを返します。

```
Get-CFNStack -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[DescribeStacks](#)」を参照してください。

デフォルトでは、describe-stacks コマンドはパラメータの値が返します。パスワードなど、機密を要するパラメータ値が返されないようにするには、CloudFormation テンプレートで、NoEcho プロパティを TRUE に設定します。

Important

NoEcho 属性を使用しても、以下に保存されている情報はマスクされません。

- Metadata テンプレートセクション。CloudFormation は、Metadata セクションに含める情報の変換、変更、または編集を行いません。詳細については、「[メタデータ](#)」を参照してください。
- Outputs テンプレートセクション。詳細については、「[出力](#)」を参照してください。
- リソース定義の Metadata 属性。詳細については、「[Metadata 属性](#)」を参照してください。

パスワードやシークレットなどの機密情報を含めるには、これらのメカニズムを使用しないことを強くお勧めします。

Important

機密情報は、CloudFormation テンプレートに直接埋め込むのではなく、スタックテンプレートの動的パラメータを使用して CloudFormation の外部 (AWS Systems Manager パラメータストアや AWS Secrets Manager など) に保存して管理した上で参照することをお勧めします。

詳細については、「[テンプレートに認証情報を埋め込まない](#)」のベストプラクティスを参照してください。

テンプレートを取得する

CLI

AWS CLI

AWS CloudFormation スタックのテンプレート本文を表示する方法

以下の `get-template` の例は、`myteststack` スタックのテンプレートを示しています。

```
aws cloudformation get-template --stack-name myteststack
```

出力:

```
{
  "TemplateBody": {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Outputs": {
      "BucketName": {
        "Description": "Name of S3 bucket to hold website content",
        "Value": {
          "Ref": "S3Bucket"
        }
      }
    },
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
template showing how to create a publicly accessible S3 bucket. **WARNING** This
template creates an S3 bucket. You will be billed for the AWS resources used if
you create a stack from this template.",
    "Resources": {
      "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
          "AccessControl": "PublicRead"
        }
      }
    }
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetTemplate](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックに関連付けられているテンプレートを返します。

```
Get-CFNTemplate -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[GetTemplate](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックに関連付けられているテンプレートを返します。

```
Get-CFNTemplate -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[GetTemplate](#)」を参照してください。

スタックリソースを一覧表示する

CLI

AWS CLI

スタック内のリソースを一覧表示する方法

以下のコマンドは、指定されたスタック内にあるリソースを一覧表示します。

```
aws cloudformation list-stack-resources \  
  --stack-name my-stack
```

出力:

```
{  
  "StackResourceSummaries": [  
    {  
      "LogicalResourceId": "bucket",  
      "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",  
      "ResourceType": "AWS::S3::Bucket",
```

```
        "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
        "ResourceStatus": "CREATE_COMPLETE",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    },
    {
        "LogicalResourceId": "function",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",
        "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
        "ResourceStatus": "UPDATE_COMPLETE",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    },
    {
        "LogicalResourceId": "functionRole",
        "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
        "ResourceType": "AWS::IAM::Role",
        "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
        "ResourceStatus": "CREATE_COMPLETE",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    }
]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListStackResources](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたスタックに関連付けられているすべてのリソースの説明を返します。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[ListStackResources](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたスタックに関連付けられているすべてのリソースの説明を返します。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[ListStackResources](#)」を参照してください。

スタックを一覧表示する

[list-stacks](#) コマンドを使用して、スタックを一覧表示します。指定されたステータスコードを持つスタックだけを一覧表示するには、`--stack-status-filter` オプションを含めてください。`--stack-status-filter` オプションには、1 つ、または複数のスタックステータスコードを指定できます。詳細については、「[スタックステータスコード](#)」を参照してください。

CLI

AWS CLI

AWS CloudFormation スタックを一覧表示する方法

以下の `list-stacks` コマンドは、ステータスが `CREATE_COMPLETE` になっているすべてのスタックの概要を表示します。

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE
```

出力:

```
[
  {
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket:
Sample template showing how to create a publicly accessible S3 bucket.
**WARNING** This template creates an S3 bucket. You will be billed for the AWS
resources used if you create a stack from this template.",
    "StackStatusReason": null,
    "CreationTime": "2013-08-26T03:27:10.190Z",
    "StackName": "myteststack",
    "StackStatus": "CREATE_COMPLETE"
```



```
}  
]
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListStacks](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: すべてのスタックの概要情報を返します。

```
Get-CFNStackSummary
```

例 2: 現在作成中のすべてのスタックの概要情報を返します。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

例 3: 現在作成中または更新中のすべてのスタックの概要情報を返します。

```
Get-CFNStackSummary -StackStatusFilter @( "CREATE_IN_PROGRESS",  
"UPDATE_IN_PROGRESS" )
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[ListStacks](#)」を参照してください。

Tools for PowerShell V5

例 1: すべてのスタックの概要情報を返します。

```
Get-CFNStackSummary
```

例 2: 現在作成中のすべてのスタックの概要情報を返します。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

例 3: 現在作成中または更新中のすべてのスタックの概要情報を返します。

```
Get-CFNStackSummary -StackStatusFilter @( "CREATE_IN_PROGRESS",  
"UPDATE_IN_PROGRESS" )
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[ListStacks](#)」を参照してください。

スタックを更新する

[update-stack](#) コマンドを使用して、スタックを直接更新します。更新するスタック、パラメータ値、機能を指定し、更新されたテンプレートを使用する場合はテンプレートの名前も指定します。詳細については、「[スタックを直接更新する](#)」を参照してください。

CLI

AWS CLI

AWS CloudFormation スタックを更新する方法

以下の `update-stack` コマンドは、`mystack` スタックのテンプレートと入力パラメータを更新します。

```
aws cloudformation update-stack --stack-name mystack --  
template-url https://s3.amazonaws.com/sample/updated.template --  
parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair  
ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

以下の `update-stack` コマンドは、`mystack` スタックの `SubnetIDs` パラメータ値のみを更新します。パラメータ値を指定しない場合は、テンプレートで指定されているデフォルト値が使用されます。

```
aws cloudformation update-stack --stack-name mystack --  
template-url https://s3.amazonaws.com/sample/updated.template  
--parameters ParameterKey=KeyPairName,UsePreviousValue=true  
ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

以下の `update-stack` コマンドは、`mystack` スタックに 2 つのスタック通知トピックを追加します。

```
aws cloudformation update-stack --stack-name mystack --use-previous-template --  
notification-arns "arn:aws:sns:use-east-1:123456789012:mytopic1" "arn:aws:sns:us-  
east-1:123456789012:mytopic2"
```

詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS CloudFormation スタックの更新](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateStack](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV1」はその値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
    -TemplateBody "{Template Content Here}" `
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 2: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
    -TemplateBody "{Template Content Here}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 3: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV2」はその値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 4: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言さ

れているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                              @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 5: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」(この例では IAM リソースが含まれていることを想定) を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。IAM リソースが含まれるスタックでは、Capabilities「CAPABILITY_IAM」パラメータを指定する必要があります。指定されていない場合は、更新が「InsufficientCapabilities」エラーで失敗します。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                              @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[UpdateStack](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV1」はその値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 2: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
    -TemplateBody "{Template Content Here}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 3: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV2」はその値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 4: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。

```
Update-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 5: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」(この例では IAM リソースが含まれていることを想定) を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイゼーションパラメータは、「ParameterKey」と「ParameterValue」の代わりに、「Key」と「Value」を使用して指定することもできます。IAM リソースが含まれるスタックでは、Capabilities「CAPABILITY_IAM」パラメータを指定する必要があります。指定されていない場合は、更新が「InsufficientCapabilities」エラーで失敗します。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[UpdateStack](#)」を参照してください。

Note

すべての通知を削除するには、`--notification-arns` オプションに `[]` を指定します。

テンプレートを検証する

[validate-template](#) コマンドを使用して、テンプレートファイルの構文エラーをチェックします。

検証中に、CloudFormation はテンプレートが有効な JSON であるかどうかをまず確認します。そうでない場合は、CloudFormation はテンプレートが有効な YAML であるかどうかを確認します。両方のチェックが失敗すると、CloudFormation はテンプレートの検証エラーを返します。

CLI

AWS CLI

AWS CloudFormation テンプレートを検証する方法

以下の `validate-template` コマンドは、`sampletemplate.json` テンプレートを検証します。

```
aws cloudformation validate-template --template-body file://sampletemplate.json
```

出力:

```
{
  "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
showing how to create a publicly accessible S3 bucket. **WARNING** This template"
```

```
creates an S3 bucket. You will be billed for the AWS resources used if you
create a stack from this template.",
  "Parameters": [],
  "Capabilities": []
}
```

詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation テンプレートの使用」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ValidateTemplate](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定されたテンプレートコンテンツを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

例 2: Amazon S3 URL 経由でアクセスされる、指定されたテンプレートを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V4)」の「[ValidateTemplate](#)」を参照してください。

Tools for PowerShell V5

例 1: 指定されたテンプレートコンテンツを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

例 2: Amazon S3 URL 経由でアクセスされる、指定されたテンプレートを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference (V5)」の「[ValidateTemplate](#)」を参照してください。

以下は、検証エラーを生成する応答の例です。

```
{
  "ResponseMetadata": {
    "RequestId": "4ae33ec0-1988-11e3-818b-e15a6df955cd"
  },
  "Errors": [
    {
      "Message": "Template format error: JSON not well-formed. (line 11, column
8)",
      "Code": "ValidationError",
      "Type": "Sender"
    }
  ],
  "Capabilities": [],
  "Parameters": []
}
A client error (ValidationError) occurred: Template format error: JSON not well-formed.
(line 11, column 8)
```

Note

validate-template コマンドは、テンプレートの構文のみをチェックするように設計されています。リソースに対して指定したプロパティ値が、そのリソースに対して有効であるかどうかは保証されません。また、スタックの作成時に存在するリソースの数も決定しません。

動作の妥当性を確認するには、スタックの作成を試みる必要があります。CloudFormation スタックにはサンドボックスやテスト領域がないため、テスト中に作成したリソースに対して課金されます。

AWS CLI を使用してローカルアーティファクトを S3 バケットにアップロードする

AWS CLI を使用して、CloudFormation テンプレートで参照されるローカルアーティファクトを Amazon S3 バケットにアップロードできます。ローカルアーティファクトは、テンプレートで参照するファイルです。ファイルを手動で S3 バケットにアップロードしてからテンプレートにその場所を追加する代わりに、ローカルアーティファクトをテンプレート内で指定し、[package](#) コマンドを使用してすばやくアップロードできます。

ローカルアーティファクトは、package コマンドが Amazon S3 にアップロードしたファイルまたはフォルダへのパスです。例えば、アーティファクトは AWS Lambda 関数のソースコードへのローカルパスであったり、Amazon API Gateway REST API の OpenAPI ファイルであったりします。

package コマンドを使用する場合:

- ファイルを指定する場合、コマンドはそれを S3 バケットに直接アップロードします。
- フォルダを指定すると、コマンドはフォルダの .zip ファイルを作成し、その .zip ファイルをアップロードします。
- パスを指定しない場合、コマンドは作業ディレクトリの .zip ファイルを作成し、アップロードします。

絶対パスまたは相対パスを指定できます。相対パスはテンプレートの場所からの相対です。

アーティファクトをアップロードしたら、コマンドはテンプレートのコピーを返します。その中で、ローカルアーティファクトへの参照は、コマンドがアーティファクトをアップロードした S3 の場所に置き換えられます。返されたテンプレートを使用してスタックを作成または更新できます。

Note

ローカルアーティファクトは、package コマンドがサポートするリソースプロパティにのみ使用できます。このコマンドとサポートされているリソースプロパティのリストの詳細については、「[AWS CLI コマンド リファレンス](#)」の「[package](#) ドキュメント」を参照してください。

前提条件

開始する前に、既存の Amazon S3 バケットが必要です。

ローカルアーティファクトを使用してテンプレートをパッケージ化してデプロイする

以下のテンプレートは Lambda 関数のソースコードのローカルアーティファクトを指定します。ソースコードは `/home/user/code/lambdafunction` フォルダに保存されています。

元のテンプレート

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Resources": {
    "MyFunction": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "nodejs18.x",
        "CodeUri": "/home/user/code/lambdafunction"
      }
    }
  }
}
```

次の [package](#) コマンドは、関数のソースコードフォルダの `.zip` ファイルを作成し、指定されたバケットのルートにアップロードします。

```
aws cloudformation package \
  --s3-bucket amzn-s3-demo-bucket \
  --template /path_to_template/template.json \
  --output-template-file packaged-template.json \
  --output json
```

このコマンドは、`--output-template-file` で指定されたパスで新しいテンプレートを生成します。次に示すように、アーティファクトの参照を Amazon S3 の場所に置き換えます。`.zip` ファイルは、フォルダ名自体を使用するのではなく、フォルダの内容の MD5 チェックサムを使用して名前が付けられます。

作成されるテンプレート

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
```

```
"Resources": {
  "MyFunction": {
    "Type": "AWS::Serverless::Function",
    "Properties": {
      "Handler": "index.handler",
      "Runtime": "nodejs18.x",
      "CodeUri": "s3://amzn-s3-demo-bucket/md5 checksum"
    }
  }
}
```

テンプレートのアーティファクトをパッケージ化したら、[deploy](#) コマンドを使用して処理されたテンプレートをデプロイします。

```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name
```

51,200 バイトを超えるテンプレートをデプロイする場合は、次の例のように、`--s3-bucket` オプションを指定した状態で `deploy` コマンドを使用し、テンプレートを S3 にアップロードします。

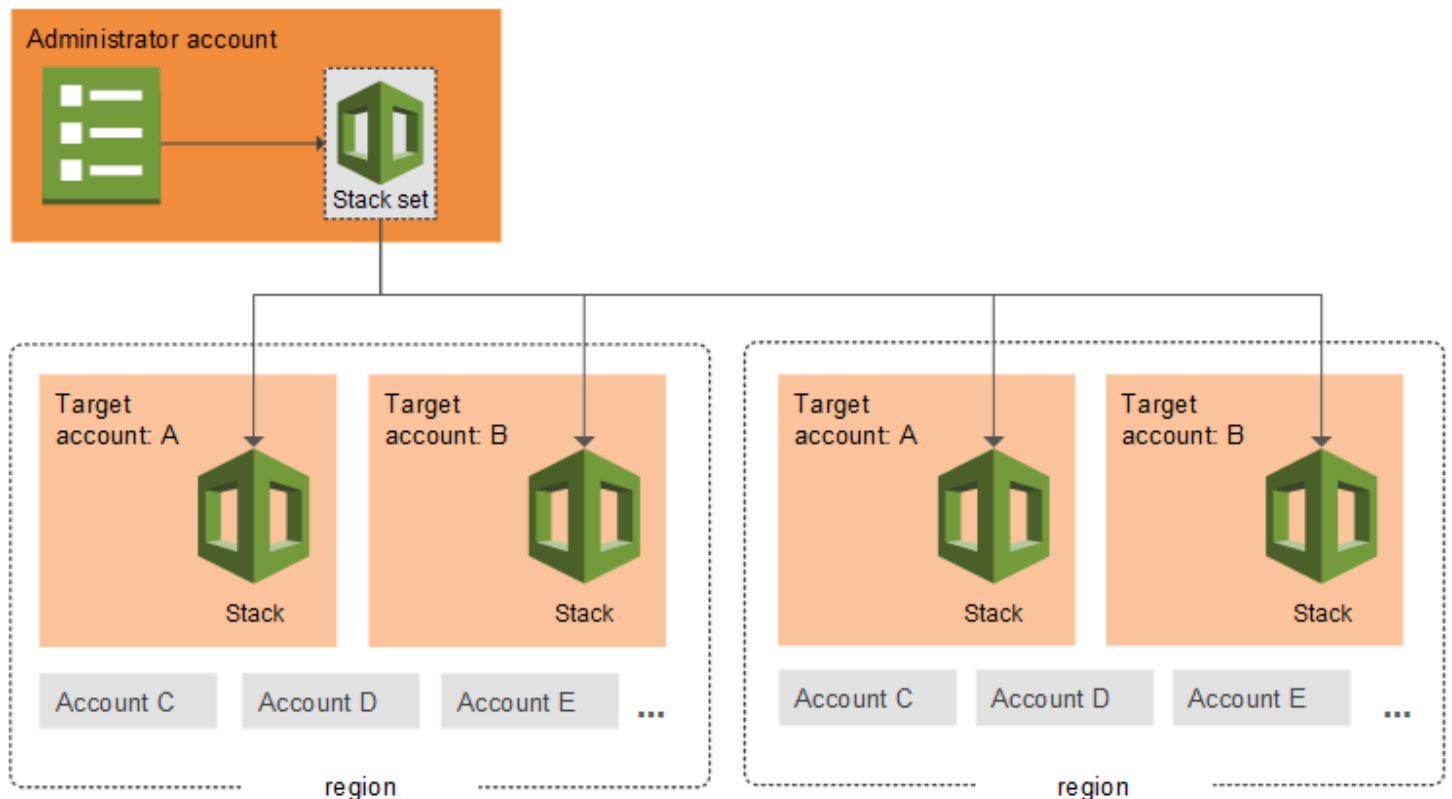
```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name \
  --s3-bucket amzn-s3-demo-bucket
```

Note

`package` コマンドを使用してローカルアーティファクトをアップロードする別の例については、「[ネストされたスタックを使用してテンプレートを再利用可能な部分に分割する](#)」を参照してください。

StackSets を使用したアカウントとリージョン全体でのスタックの管理

CloudFormation StackSets は、複数のアカウントおよび AWS リージョン全体でのスタックの作成、更新、削除を単一のオペレーションで実行できるようにすることで、スタックの機能を拡張します。管理者アカウントを使用して、CloudFormation テンプレートを定義および管理し、指定の AWS リージョン 間で選択したターゲット アカウントにスタックをプロビジョニングするためのベースとして使用します。



このセクションは、StackSets の使用開始に役立つほか、StackSet の作成、更新、削除に伴う操作やトラブルシューティングの手順に関する一般的な質問への回答を確認できます。

トピック

- [StackSets の概念](#)
- [CloudFormation StackSets を使用するための前提条件](#)
- [サンプルテンプレートを使用して StackSets の使用を開始する](#)
- [セルフマネージド型のアクセス許可を持つ CloudFormation StackSets の作成](#)
- [サービスマネージド型のアクセス許可を持つ CloudFormation StackSets の作成](#)

- [AWS Organizations で StackSets の自動デプロイを有効または無効にする](#)
- [CloudFormation StackSets を更新する](#)
- [CloudFormation StackSets にスタックを追加する](#)
- [CloudFormation StackSet 内のスタックのパラメータ値を上書きする](#)
- [CloudFormation StackSets からスタックを削除する](#)
- [CloudFormation StackSets を削除する](#)
- [ターゲットアカウントゲートを使用して失敗した StackSets デプロイを防止する](#)
- [CloudFormation StackSets の同時実行モードの選択](#)
- [CloudFormation StackSets でのドリフト検出の実行](#)
- [スタックを CloudFormation StackSets にインポートする](#)
- [CloudFormation StackSets を使用するためのベストプラクティス](#)
- [CloudFormation StackSets サンプルテンプレート](#)
- [CloudFormation StackSets のトラブルシューティング](#)

StackSets の概念

StackSets を理解して使用するために重要な用語と概念を以下に示します。

トピック

- [管理者アカウントおよびターゲットアカウント](#)
- [CloudFormation StackSets](#)
- [StackSets のアクセス許可モデル](#)
- [スタックインスタンス](#)
- [StackSet オペレーション](#)
- [StackSet オペレーションのオプション](#)
- [タグ](#)
- [StackSets ステータスコード](#)
- [スタックインスタンスのステータスコード](#)

管理者アカウントおよびターゲットアカウント

管理者アカウントは、StackSets を作成する AWS アカウント です。サービスマネージド型のアクセス許可を持つ StackSets の場合、管理者アカウントは組織の管理アカウントまたは委任管理者アカウントです。StackSet を作成した AWS 管理者アカウントにサインインすることで、StackSet を管理できます。

ターゲットアカウントは、StackSet の 1 つ以上のスタックを作成、更新、削除するアカウントです。StackSet を使用してターゲットアカウントにスタックを作成する前に、管理者アカウントとターゲットアカウントの間に信頼関係をセットアップします。

CloudFormation StackSets

StackSet は、指定された AWS アカウント およびリージョンにデプロイされる複数のスタックのコンテナとして機能します。各スタックは同じ CloudFormation テンプレートに基づいていますが、パラメータを使用して個々のスタックをカスタマイズすることができます。

StackSet を定義したら、指定したターゲットアカウントや AWS リージョン でスタックを作成、更新、削除できるようになります。スタックを作成、更新、または削除する際には、オペレーションの環境設定も指定できます。例えば、オペレーションを実行するリージョンの順序、スタックオペレーションが停止する前の耐障害性のしきい値、およびスタックオペレーションを同時に実行するアカウントの数を含めます。

StackSet はリージョンのリソースです。StackSet を 1 つの AWS リージョン で作成する場合は、そのリージョンを表示しているときにのみ表示または変更できます。

StackSets のアクセス許可モデル

StackSets は、セルフマネージド型のアクセス許可、またはサービスマネージド型のアクセス許可のいずれかを使用して作成できます。

セルフマネージド型アクセス許可を使用する場合、アカウントとリージョン間でデプロイするために StackSets で必要な IAM ロールを作成します。これらのロールは、StackSet を管理するアカウントとスタックインスタンスをデプロイするアカウントとの間に信頼できる関係を確立するために必要です。このアクセス許可モデルを使用すると、StackSets は、IAM ロールを作成するアクセス許可を持つ任意の AWS アカウント にデプロイできます。

サービスマネージド型のアクセス許可を使用する場合、AWS Organizations が管理するアカウントにスタックインスタンスをデプロイできます。このアクセス許可モデルを使用すると、必要な IAM

ロールを作成する必要はありません。ユーザーに代わって StackSets が IAM ロールを作成します。このモデルでは、将来組織に追加するアカウントへの自動デプロイをオンにすることもできます。

AWS Organizations は、CloudFormation と統合され、AWS リソースの拡張や成長に伴い、環境の一元管理およびガバナンスを支援します。

- 管理アカウント - 組織を作成するために使用するアカウント。詳細については、「[AWS Organizations の用語と概念](#)」を参照してください。
- 委任管理者 - 互換性のある AWS サービスでは、組織内の AWS メンバーアカウントをそのサービスにおける組織のアカウントの管理者として登録できます。詳細については、「[AWS Organizations で使用できる AWS のサービス](#)」を参照してください。

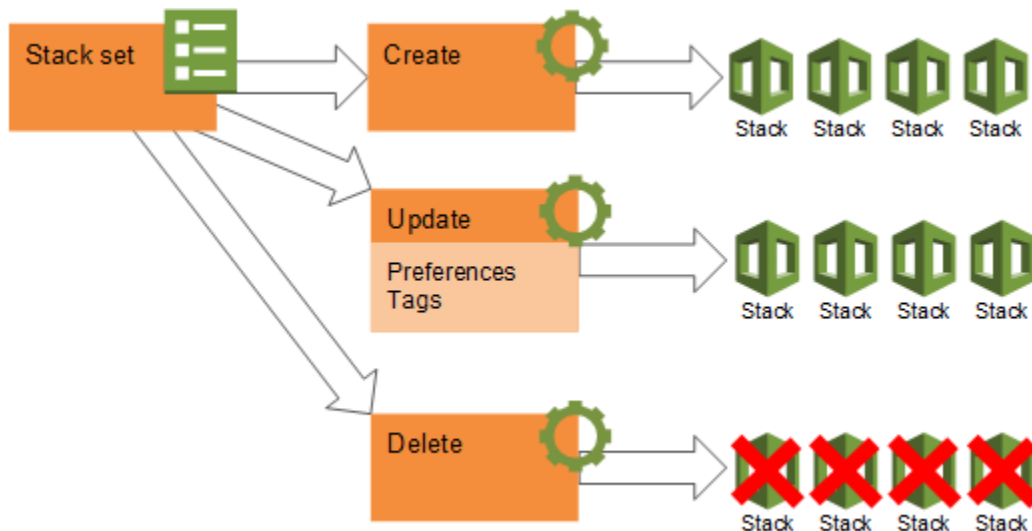
サービスマネージド型のアクセス許可を使用して StackSets を作成および管理する方法の詳細については、次のトピックを参照してください。

- [AWS Organizations を使用して StackSets のための信頼されたアクセスをアクティブ化する](#)
- [委任された管理者メンバーアカウントを登録する](#)
- [サービスマネージド型のアクセス許可を持つ CloudFormation StackSets の作成](#)

スタックインスタンス

スタックインスタンスは、リージョン内のターゲットアカウントのスタックへのリファレンスです。スタックインスタンスはスタックなしで存在できます。例えば、何らかの理由によりスタックが作成されていない場合は、スタック作成の失敗理由がスタックインスタンスに表示されます。スタックインスタンスは、1 つの StackSet にのみ関連付けられます。

次の図は、StackSets、スタックオペレーション、スタック間の論理関係を表しています。StackSet を更新すると、関連付けられているすべてのスタックインスタンスは、すべてのアカウントおよびリージョンで更新されます。



StackSet オペレーション

StackSets では、以下のオペレーションを実行できます。

StackSet の作成

新しい StackSet を作成するには、スタックの作成に使用する CloudFormation テンプレートの指定、スタックを作成するターゲットアカウントの指定、ターゲットアカウントのスタックをデプロイする AWS リージョン の特定などを行います。StackSet では、選択したリージョン内の指定したすべてのターゲットアカウントに対して同一の設定を使用して、同一のスタックリソースを一貫してデプロイできます。

StackSet を更新する

StackSet を更新したら、StackSet のスタックに変更内容をプッシュします。StackSet は、以下のいずれかの方法で更新できます。テンプレートをアップデートすると必ずすべてのスタックに反映されます。StackSet の一部スタックのテンプレートを選択してアップデートすることはできません。その他のスタックもアップデートされます。

- テンプレートの既存の設定を変更するか (例: 特定のサービスのパラメータ設定の更新)、新しいリソースを追加します (例: 新しい Amazon EC2 インスタンスの追加)。
- テンプレートを別のテンプレートに置き換えます。
- 既存のリージョンまたは追加リージョンの既存または追加のターゲットアカウントにスタックを追加します。

スタックの削除

スタックを削除する場合は、指定したリージョン内の指定したターゲットアカウントから、スタックおよび関連付けられているリソースを削除します。スタックは次の方法で削除できます。

- 一部のターゲットアカウントからスタックを削除し、実行中の他のターゲットアカウントの他のスタックはそのままにしておきます。
- 一部のリージョンからスタックを削除し、実行中の他のリージョンのスタックはそのままにしておきます。
- StackSet からスタックを削除しても保存されるため、[スタックを保持] オプションを選択して、引き続き、個別にスタックを実行できます。その後、CloudFormation で設定された StackSet の外部で保持されたスタックを管理できます。
- StackSet 全体を削除する準備段階として、StackSet 内のすべてのスタックを削除します。

StackSet を削除する

StackSet は、StackSet 内にスタックインスタンスが存在しない場合のみ、削除できます。

StackSet オペレーションのオプション

このセクションで説明されているオプションを使用すると、StackSet オペレーションを成功させるために許容されている障害時間や障害数を制御しやすくなるだけでなく、スタックリソースの損失を抑えるのにも役立ちます。

同時アカウントの最大数

この設定は、ワークフローの作成、更新、削除に適用され、一度にオペレーションを実行するターゲットアカウントの最大数または割合 (%) を指定することができます。数または割合 (%) が少ない場合は、オペレーションが、一度に少数のターゲットアカウントで実行していることを表します。オペレーションは、一度に 1 つのリージョンで、[Deployment order] (デプロイ順序) ボックスで指定した順序で実行します。例えば、スタックを 2 つのリージョンの 10 のターゲットアカウントにデプロイした場合、[Maximum concurrent accounts] (同時アカウントの最大数) を 50 と [By percentage] (割合) に設定すると、スタックは最初のリージョンの 5 つのアカウント、次に最初のリージョンの次の 5 つのアカウントにデプロイされた後、次のリージョンに移動し、最初の 5 つのターゲットアカウントにデプロイされます。

[By percentage] (割合) を選択して、指定された割合 (%) が指定したアカウントの整数にならない場合は、CloudFormation によって丸められます。例えば、スタックを 10 のターゲットアカ

ウントにデプロイし、[Maximum concurrent accounts] (同時アカウントの最大数) を [25] と [By percentage] (割合) に設定した場合は、同時にデプロイされるスタック数は、CloudFormation によって 2.5 (設定できません) から 2 に丸められます。

この設定では、オペレーションの最大値を指定できます。大規模なデプロイの場合、特定の状況下では、同時に処理される実際のアカウント数はサービスのスロットリングのために低くなる可能性があります。

同時アカウントの最大数は、耐障害性の値によって異なることがあります (耐障害性の値は、ご使用の同時実行モードに応じて異なります)。同時実行モードが厳格な耐障害性に設定されている場合、同時アカウントの最大数を耐障害性設定よりも 1 つだけ多くすることができます。

同時実行モード

この設定は、作成、更新、および削除のワークフローで利用可能であり、StackSet オペレーション時の同時実行レベルの動作を選択できます。詳細については、「[CloudFormation StackSets の同時実行モードの選択](#)」を参照してください。

耐障害性

この設定は、ワークフローの作成、更新、削除で利用でき、各リージョンで発生する場合があるスタックオペレーションの失敗の最大数や割合を指定できます。この値を超えると、オペレーションは CloudFormation によって自動的に停止されます。数値または割合 (%) が少ない場合は、オペレーションで実行しているスタックが少数であることを意味しますが、失敗したオペレーションのトラブルシューティングをすばやく開始することができます。たとえば、10 のスタックを 3 つのリージョン内の 10 のターゲットアカウントにデプロイした場合、[Failure tolerance (耐障害性)] を [20] と [By percentage (割合)] に設定すると、オペレーションを続行するために、最大 2 つのスタックのアップデートがリージョン内で失敗することがあります。同リージョンの 3 番目のスタックで失敗した場合、CloudFormation はオペレーションを停止します。最初のリージョンでスタックを更新できない場合、更新オペレーションはそのリージョンで続行され、その後、次のリージョンに移行します。2 番目のスタックを 2 番目のリージョンで更新できない場合、耐障害性は 20% に達し、同リージョンの 3 番目のスタックが失敗した場合、CloudFormation は更新オペレーションを停止し、次のリージョンに移行することもあります。

[By percentage] (割合) を選択して、指定された割合 (%) が、各リージョン内のスタックの整数にならない場合、CloudFormation は切り捨てられます。例えば、スタックを 3 つのリージョンの 10 のターゲットアカウントにデプロイし、[Failure tolerance] (耐障害性) を [25] と [By percentage] (割合) に設定した場合、リージョンあたりの耐障害性 2.5 (設定できません) は、CloudFormation によって 2 に切り捨てられます。

スタックの保持

スタック削除ワークフローで利用できるこの設定により、StackSet からスタックを削除した後でも、スタックとそのリソースを実行し続けることができます。スタックを保持すると、CloudFormation は、個別のアカウントとリージョンインタクトにスタックを残します。スタックは StackSet から関連付けを解除しますが、スタックとそのリソースを保存します。スタックの削除オペレーションが完了したら、スタックを作成したターゲットアカウント (管理者アカウントではありません) を使用して、CloudFormation で保持されているスタックを管理します。

リージョンの同時実行

この設定は、作成、更新、および削除のワークフローで利用可能であり、StackSets をリージョンにデプロイする方法を選択できます。

順次 - リージョンのデプロイ失敗が指定された耐障害性を超えない限り、リージョンの [デプロイ順序] ボックスで指定されているように、一度に 1 つのリージョンに StackSets オペレーションをデプロイします。デフォルトでは、順次デプロイが選択されています。

並列 - リージョンのデプロイ失敗が指定された耐障害性を超えない限り、指定されたすべてのリージョンに StackSets オペレーションを同時にデプロイします。

タグ

StackSet の作成オペレーションおよび更新オペレーション時にタグを追加するには、キーと値のペアを指定します。請求額とコストの配分を行うために StackSet リソースのソートおよびフィルタリングを行う場合は、タグが便利です。AWS でのタグの使用方法に関する詳細については、「AWS Billing and Cost Management ユーザーガイド」の「[Organizing and tracking costs using AWS cost allocation tags](#)」を参照してください。キーと値のペアを指定したあと、[+] を選択してタグを保存します。タグの右側にある赤い X を選択すると、使用しなくなったタグを削除できます。

StackSets に適用するタグは、すべてのスタックと、スタックにより作成されるリソースに適用されます。CloudFormation のスタックのみのレベルでタグを追加することもできますが、そのタグは StackSets には表示されない場合があります。

StackSets でシステム定義のタグが追加されることはありませんが、文字列 `aws:` を含むタグのキー名から開始しないようにしてください。

StackSets ステータスコード

CloudFormation StackSets は、StackSet オペレーションのステータスコードを生成します。

StackSet オペレーションのステータスコードを以下のテーブルに示します。

RUNNING

オペレーションは現在進行中です。

SUCCEEDED

オペレーションは、オペレーションの耐障害性を越えることなく終了しました。

FAILED

オペレーションが完了できなかったスタックの数は、ユーザー定義の耐障害性を超えました。オペレーション向けに設定した耐障害性の値は、スタック作成およびスタック更新のオペレーション時に各リージョンに適用されます。リージョン内で失敗したスタックの数が耐障害性を超えた場合、リージョン内のオペレーションのステータスは、FAILED に変更されます。オペレーション全体のステータスも FAILED に設定され、その他のリージョンのオペレーションは CloudFormation によってキャンセルされます。

QUEUED

Service-managed permissions 一連のオペレーションを必要とする自動デプロイの場合、オペレーションはキューに入り、実行されます。例:

- アカウントをある組織単位 (OU) OU1 から別の組織単位 OU2 に移動すると、自動デプロイがトリガーされます。StackSets は、削除オペレーションを実行してターゲットリージョンのターゲット OU1 アカウントからスタックインスタンスを削除し、ターゲットリージョンのターゲット OU2 アカウントにスタックインスタンスを追加するための作成オペレーションをキューに入れます。
- OU にアカウント AccountA を追加すると、自動デプロイがトリガーされます。StackSets は、ターゲットリージョンの AccountA にスタックインスタンスを追加するための作成オペレーションを実行します。この作成オペレーションの実行中に別のアカウント AccountB を OU に追加すると、StackSets は 2 番目の作成オペレーションをキューに入れます。最初の作成オペレーションが完了すると、StackSets は 2 番目の作成オペレーションを実行して、ターゲットリージョンの AccountB にスタックインスタンスを追加します。

STOPPING

ユーザーのリクエストにより、オペレーションは停止中です。

STOPPED

ユーザーのリクエストにより、オペレーションは停止しました。

スタックインスタンスのステータスコード

CloudFormation StackSets は、スタックインスタンスのステータスコードを生成します。

次の表は、StackSets 内のスタックインスタンスのステータスコードについて説明したものです。

CURRENT

スタックは、StackSet と同様、最新の状態です。

OUTDATED

スタックは、以下のいずれかの理由により、StackSet で最新状態ではありません。

- 関連付けられたスタックの [CreateStackSet](#) オペレーションまたは [UpdateStackSet](#) オペレーションが失敗しました。
- スタックは作成または更新される前に、失敗または停止した [CreateStackSet](#) オペレーションまたは [UpdateStackSet](#) オペレーションの一部でした。

INOPERABLE

[DeleteStackInstances](#) オペレーションが失敗し、スタックは不安定な状態のままになっています。この状態のスタックは、今後の [UpdateStackSet](#) オペレーションから除外されます。必要に応じて、RetainStacks を true に設定して [DeleteStackInstances](#) オペレーションを実行し、スタックインスタンスを削除します。その後、手動でスタックを削除します。

CANCELLED

指定されたアカウントとリージョンでのオペレーションがキャンセルされました。これは、ユーザーが StackSet オペレーションを停止したか、StackSet オペレーションの耐障害性を越えたためです。

FAILED

指定されたアカウントとリージョンでのオペレーションに失敗しました。リージョン内の十分な数のアカウントで StackSet オペレーションが失敗した場合、StackSet オペレーション全体の耐障害性を越える可能性があります。

FAILED_IMPORT

指定されたアカウントとリージョンのスタックインスタンスのインポートが失敗し、スタックが不安定な状態になりました。障害の原因となった問題が修正されたら、インポート操作を再試行できます。リージョン内の十分な数のアカウントで十分な数の StackSet オペレーションが失敗した場合、StackSet オペレーション全体の障害許容度を超える可能性があります。

PENDING

指定されたアカウントとリージョンでのオペレーションはまだ開始されていません。

RUNNING

指定されたアカウントとリージョンでのオペレーションは現在進行中です。

SKIPPED_SUSPENDED_ACCOUNT

操作の時点でアカウントが停止されたため、指定されたアカウントとリージョンでの操作がスキップされました。

SUCCEEDED

指定されたアカウントとリージョンでのオペレーションが正常に完了しました。

CloudFormation StackSets を使用するための前提条件

StackSets はスタックの機能を拡張するため、1 回のオペレーションで複数のアカウントとリージョンにまたがるスタックを作成、更新、削除できます。

StackSets では複数のアカウントでスタックオペレーションが実行されるため、最初の StackSet の作成を開始する前に、AWS アカウント で定義される必要なアクセス許可が必要です。

セルフマネージドまたはサービスマネージドの許可を使用して StackSets を管理できます。

- セルフマネージド StackSets の場合、各ターゲットアカウントと AWS リージョン で IAM ロールを作成および管理する必要があります。詳細については、「[セルフマネージドのアクセス許可の付与](#)」を参照してください。
- サービスマネージド StackSets の場合、各アカウントで IAM ロールを手動で作成および管理する必要はありません。AWS がロールの作成と許可を処理します。詳細については、「[信頼されたアクセスをアクティブ化する](#)」を参照してください。

トピック

- [デフォルトでは無効になっている AWS リージョン で StackSet オペレーションを実行する準備をする](#)
- [セルフマネージド型のアクセス許可を付与する](#)
- [AWS Organizations を使用して StackSets のための信頼されたアクセスをアクティブ化する](#)

デフォルトでは無効になっている AWS リージョン で StackSet オペレーションを実行する準備をする

2019 年 3 月 20 日以降に導入されたアジアパシフィック (香港) などの AWS リージョン は、デフォルトで無効になっています。これらのリージョンは使用前にアカウントで有効にする必要があります。詳細については、「AWS アカウント管理 リファレンスガイド」の「[アカウントでの AWS リージョン の有効化または無効化](#)」を参照してください。

デフォルトで無効になっているリージョンで、StackSet の管理者アカウント (セルフマネージド型のアクセス許可を使用している場合) または組織の管理アカウント (サービスマネージド型のアクセス許可を使用している場合) から StackSet を作成するには、最初に管理者アカウントまたは管理アカウントでそのリージョンを有効にする必要があります。

CloudFormation で、スタックインスタンスを正常に作成または更新するには、以下が必要です。

- ターゲットアカウントは、そのターゲットアカウントで現在有効になっているリージョンに存在する必要があります。
- StackSet の管理者アカウントまたは組織の管理アカウントで、ターゲットアカウントと同じリージョンを有効にする必要があります。

Important

StackSet のオペレーション中、管理者アカウントとターゲットアカウントは、アカウント自体だけでなく、さらに関連する StackSet およびそれらのインスタンスに関するメタデータをやり取りします。

また、StackSet のインスタンスがあるアカウントを含むリージョンを無効にした場合、必要に応じて、そのようなインスタンスまたはリソースをすべて削除する必要があります。さらに、無効にしたリージョンにあるターゲットアカウントに関するメタデータは、管理者アカウントに保持されます。

セルフマネージド型のアクセス許可を付与する

このトピックでは、セルフマネージド許可を使用して、複数のアカウントおよび AWS リージョンにデプロイするために、StackSets によって必要とされる IAM サービスロールを作成する方法について説明します。これらのロールは、StackSet を管理するアカウントとスタックインスタンスをデプロイするアカウントとの間に信頼できる関係を確認するために必要です。このアクセス許可モデルを使用すると、StackSets は、IAM ロールを作成するアクセス許可を持つ任意の AWS アカウントにデプロイできます。

サービス管理アクセス許可を使用するには、[信頼されたアクセスをアクティブ化する](#)代わりに「」を参照してください。

トピック

- [セルフマネージド許可の概要](#)
- [管理者アカウントのすべてのユーザーに、すべてのターゲットアカウントのスタックを管理するための許可を付与します。](#)
- [StackSet オペレーションのアドバンスドアクセス許可オプションの設定](#)
- [混乱した代理問題を軽減するためにグローバルキーを設定する](#)

セルフマネージド許可の概要

セルフマネージド型のアクセス許可を持つ StackSet を作成する前に、各アカウントで IAM サービスロールを作成しておく必要があります。

基本的なステップは次のとおりです。

1. どの AWS アカウント が管理者アカウントであることを判断します。

StackSets はこの管理者アカウントで作成されます。ターゲットアカウントは、StackSet に属する個別のスタックを作成するアカウントです。

2. StackSet のアクセス許可を構成する方法を決定します。

最も単純な (そして最も許容性の高い) アクセス許可設定は、管理者アカウントのすべてのユーザーとグループに、そのアカウントで管理されているすべての StackSet を作成および更新する権限を付与することです。きめ細かな制御が必要な場合は、以下を指定するアクセス許可を設定できます。

- どのユーザーとグループがどのターゲットアカウントで StackSet オペレーションを実行できるか。

- ユーザーとグループが StackSets に含めることができるリソース。
 - 特定のユーザーおよびグループが実行できる StackSet オペレーション。
3. 管理者とターゲットアカウントに必要な IAM サービスロールを作成して、必要なアクセス許可を定義します。

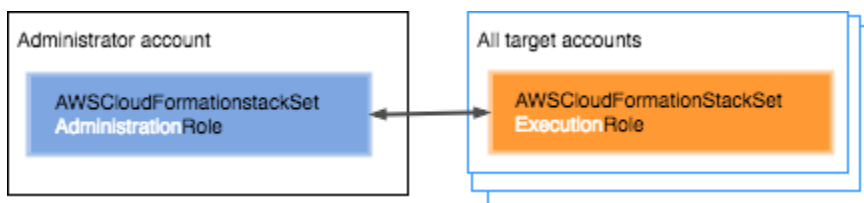
具体的には、必要な 2 つのロールは次のとおりです：

- `AWSCloudFormationStackSetAdministrationRole` – このロールは管理者アカウントにデプロイされます。
- `AWSCloudFormationStackSetExecutionRole` – このロールは、スタックインスタンスを作成するすべてのアカウントにデプロイされます。

管理者アカウントのすべてのユーザーに、すべてのターゲットアカウントのスタックを管理するための許可を付与します。

このセクションでは、管理者アカウントのすべてのユーザーとグループが、すべてのターゲットアカウントで StackSet オペレーションを実行することを許可するためのアクセス許可を設定する方法を説明します。管理者アカウントとターゲットアカウントで必要な IAM サービスロールを作成する手順を説明します。管理者アカウントのユーザーは誰でも、任意のターゲットアカウント全体でスタックを作成、更新、または削除できるようになります。

このようにアクセス許可を構成することで、ユーザーは StackSet を作成または更新するときに管理ロールを渡す必要がなくなります。



Administrator account

管理者アカウントで、`AWSCloudFormationStackSetAdministrationRole` という名前の IAM ロールを作成します。

そのためには、CloudFormation テンプレートからスタックを作成します。このテンプレートは <https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetAdministrationRole.yml> で入手できます。

Example アクセス許可ポリシーの例。

前述のテンプレートによって作成された管理ロールには、次の許可ポリシーが含まれています。

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

Example 信頼ポリシーの例 1

また、前述のテンプレートには、管理ロールと、ロールにアタッチされた許可を使用するためのサービス許可を付与する次の信頼ポリシーも含まれています。

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

Example信頼ポリシーの例 2

デフォルトで無効になっているリージョンにあるターゲットアカウントにスタックインスタンスをデプロイするには、そのリージョンのリージョンサービスプリンシパルを含める必要があります。デフォルトで無効になっているリージョンごとに、独自のリージョンサービスプリンシパルがあります。

次の信頼ポリシーのサンプルは、デフォルトでは無効になっているアジアパシフィック (香港) リージョン (ap-east-1) で管理ロールを使用するための許可をサービスに付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "cloudformation.amazonaws.com",
          "cloudformation.ap-east-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

詳細については、「[デフォルトでは無効になっている AWS リージョンで StackSet オペレーションを実行する準備をする](#)」を参照してください。リージョンコードの一覧については、「AWS 全般のリファレンス ガイド」の「[Regional endpoints](#)」を参照してください。

Target accounts

各ターゲットアカウントで、管理者アカウントを信頼する AWSCloudFormationStackSetExecutionRole という名前のサービスロールを作成します。ロールにはこの正確な名前が必要です。そのためには、以下の CloudFormation テンプレートからスタックを作成します。このテンプレートは <https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetExecutionRole.yml> でオンラインで入手できます。このテンプレートを使用すると、ターゲットアカウントと信頼関係を持っている必要がある管理者アカウントのアカウント ID を指定するよう求められます。

⚠ Important

このテンプレートは管理者アクセス権を付与することに注意してください。テンプレートを使用してターゲットアカウント実行ロールを作成した後、ポリシーステートメントのアクセス許可を、StackSets を使用して作成するリソースの種類にスコープする必要があります。

ターゲットアカウントサービスロールは、CloudFormation テンプレートで指定されたすべてのオペレーションを実行する許可を必要とします。たとえば、テンプレートが S3 バケットを作成している場合、S3 の新しいオブジェクトを作成するためのアクセス許可が必要です。ターゲットアカウントは常に完全な CloudFormation 許可を必要とします。これには、スタックを作成、更新、削除、および記述するための許可が含まれます。

Example アクセス許可ポリシー 1 の例。

このテンプレートで作成されたロールは、ターゲットアカウントで次のポリシーを有効にします。

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

Example アクセス許可ポリシー 2 の例。

以下の例は、StackSets が機能するための最低限のアクセス許可を持つポリシーステートメントを示します。CloudFormation 以外のサービスのリソースを使用するターゲットアカウントでスタックを作成するには、それらのサービスアクションおよびリソースを、各ターゲットアカウントの `AWSCloudFormationStackSetExecutionRole` ポリシーステートメントに追加する必要があります。

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

Example信頼ポリシーの例

次の信頼関係は、テンプレートによって作成されています。管理者アカウントの ID は、*admin_account_id* として表示されます。

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

既存のターゲットアカウントの実行ロールの信頼関係を設定して、管理者アカウントで特定のロールを信頼できます。管理者アカウントでロールを削除し、代わりとなる新しいロールを作成する場合、新しい管理者アカウントロールでターゲットアカウントの信頼関係を設定する必要があります。これは、前の例の *admin_account_id* によって表されます。

StackSet オペレーションのアドバンスドアクセス許可オプションの設定

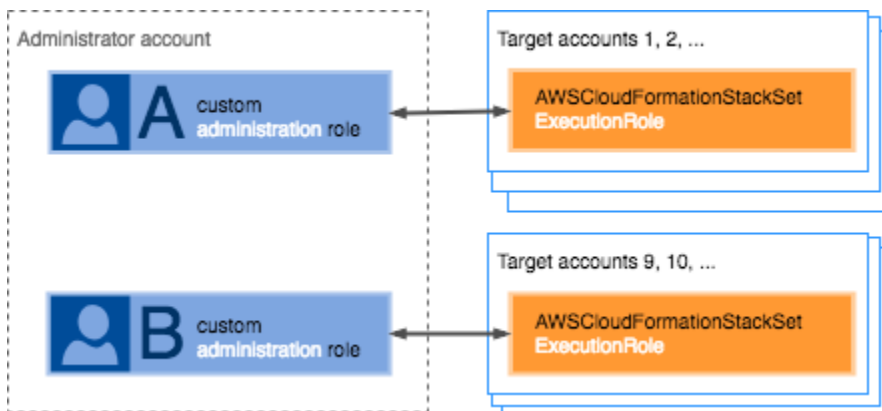
ユーザーとグループが 1 つの管理者アカウントを使用して作成している StackSets に対してきめ細かな制御が必要な場合は、IAM ロールを使用して次の項目を指定できます。

- どのユーザーとグループがどのターゲットアカウントで StackSet オペレーションを実行できるか。
- ユーザーとグループが StackSets に含めることができるリソース。
- 特定のユーザーおよびグループが実行できる StackSet オペレーション。

特定のターゲットアカウントで StackSet オペレーションを実行できるユーザーを制御する

カスタマイズされた管理ロールを使用して、どのユーザーおよびグループがどのターゲットアカウントで StackSet オペレーションを実行できるかを制御します。StackSet オペレーションを実行する管理者アカウントのユーザーとターゲットアカウントを制御します。これを行うには、管理者アカウント自体に `AWSCloudFormationStackSetAdministrationRole` サービスロールを作成するのではなく、各ターゲットアカウントと特定のカスタマイズされた管理ロールの間に信頼関係を作成します。これで、特定のターゲットアカウントで StackSet オペレーションを実行する際、特定のユーザーとグループをアクティブ化して、カスタマイズされた管理ロールを使用します。

たとえば、ロール A とロール B を管理者アカウント内に作成できます。ターゲットアカウント 1 ~ 8 にアクセスする許可を ロール A に付与することができます。ターゲットアカウント 9 ~ 16 にアクセスする許可を ロール B に付与することができます。



必要な許可を設定するには、カスタマイズされた管理ロールの定義、ターゲットアカウントのサービスロールの作成、ユーザーへの許可の付与を行い、StackSet オペレーションを実行するときに、カスタマイズされた管理ロールを渡す必要があります。

必要な許可を設定したときの一般的な動作は次のとおりです。StackSet を作成する際、ユーザーは、カスタマイズされた管理ロールを指定する必要があります。ユーザーには、CloudFormation にロールを渡すための許可が必要です。さらに、カスタマイズされた管理ロールには、StackSet に指定されたターゲットアカウントとの信頼関係が必要です。CloudFormation では StackSet を作成し、カスタマイズされた管理ロールをこの StackSet に関連付けます。StackSet を更新する場合、ユーザーはカスタマイズされた管理ロールを明示的に指定する必要があります (この StackSet で以前に使用したのと同じカスタマイズされた管理ロールである場合でも必要です)。CloudFormation は、上記の要件に従い、このロールを使用してスタックを更新します。

Administrator account

Example アクセス許可ポリシーの例。

StackSet ごとに、ターゲットアカウント実行ロールを引き受けるためのアクセス許可を持つカスタマイズされた管理ロールを作成します。

ターゲットアカウント実行ロール名は、すべてのターゲットアカウントで同じである必要があります。ロール名が `AWSCloudFormationStackSetExecutionRole` である場合、StackSets は StackSet を作成する際にその名前を自動的に使用します。カスタムロール名を指定する場合、ユーザーは StackSet を作成する際に実行ロール名を指定する必要があります。

カスタム名と次の許可ポリシーを使用して [IAM サービスロール](#) を作成します。以下の例では、`custom_execution_role` はターゲットアカウントの実行ロールを指します。

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

単一のステートメントで複数のアカウントを指定するには、アカウントをカンマで区切ります。

```
"Resource": [  
  "arn:aws:iam::target_account_id_1:role/custom_execution_role",  
  "arn:aws:iam::target_account_id_2:role/custom_execution_role"  
]
```

アカウント ID の代わりにワイルドカード (*) を使用して、すべてのターゲットアカウントを指定できます。

```
"Resource": [  
  "arn:aws:iam::*:role/custom_execution_role"  
]
```

Example信頼ポリシーの例 1

どの IAM プリンシパルがロールを引き受けることができるかを定義するには、サービスロールの信頼ポリシーを指定する必要があります。

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

Example信頼ポリシーの例 2

デフォルトで無効になっているリージョンにあるターゲットアカウントにスタックインスタンスをデプロイするには、そのリージョンのリージョンサービスプリンシパルを含める必要もあります。デフォルトで無効になっているリージョンごとに、独自のリージョンサービスプリンシパルがあります。

次の信頼ポリシーのサンプルは、デフォルトでは無効になっているアジアパシフィック (香港) リージョン (ap-east-1) で管理ロールを使用するための許可をサービスに付与します。


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "cloudformation.amazonaws.com",
          "cloudformation.ap-east-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

詳細については、「[デフォルトでは無効になっている AWS リージョン で StackSet オペレーションを実行する準備をする](#)」を参照してください。リージョンコードのリストについては、「AWS 全般のリファレンスガイド」の「[Regional endpoints](#)」を参照してください。

Example パスロールポリシーの例

また、StackSet オペレーションを実行する際にユーザーがカスタマイズされた管理ロールを渡すことを許可する IAM ユーザーの IAM 許可ポリシーも必要です。詳細については、[Granting a user permissions to pass a role to an AWS service](#) を参照してください。

以下の例では、*customized_admin_role* は、ユーザーが渡す必要のある管理ロールを意味します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/customized_admin_role"
    }
  ]
}
```

```
]
}
```

Target accounts

ターゲットアカウントごとに、このアカウントで使用するカスタマイズされた管理者ロールを信頼するサービスロールを作成します。

ターゲットアカウントロールは、CloudFormation テンプレートで指定されたすべてのオペレーションを実行する許可を必要とします。たとえば、テンプレートが S3 バケットを作成している場合、S3 の新しいオブジェクトを作成するためのアクセス許可が必要です。ターゲットアカウントは常に完全な CloudFormation 許可を必要とします。これには、スタックを作成、更新、削除、および記述するための許可が含まれます。

ターゲットアカウントのロール名は、すべてのターゲットアカウントで同じである必要があります。ロール名が `AWSCloudFormationStackSetExecutionRole` である場合、StackSets は StackSet を作成する際にその名前を自動的に使用します。カスタムロール名を指定する場合、ユーザーは StackSet を作成する際に実行ロール名を指定する必要があります。

Example アクセス許可ポリシーの例。

以下の例は、StackSets が機能するための最低限のアクセス許可を持つポリシーステートメントを示します。CloudFormation 以外のサービスのリソースを使用するターゲットアカウントでスタックを作成するには、それらのサービスアクションおよびリソースを、許可ポリシーに追加する必要があります。

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

Example信頼ポリシーの例

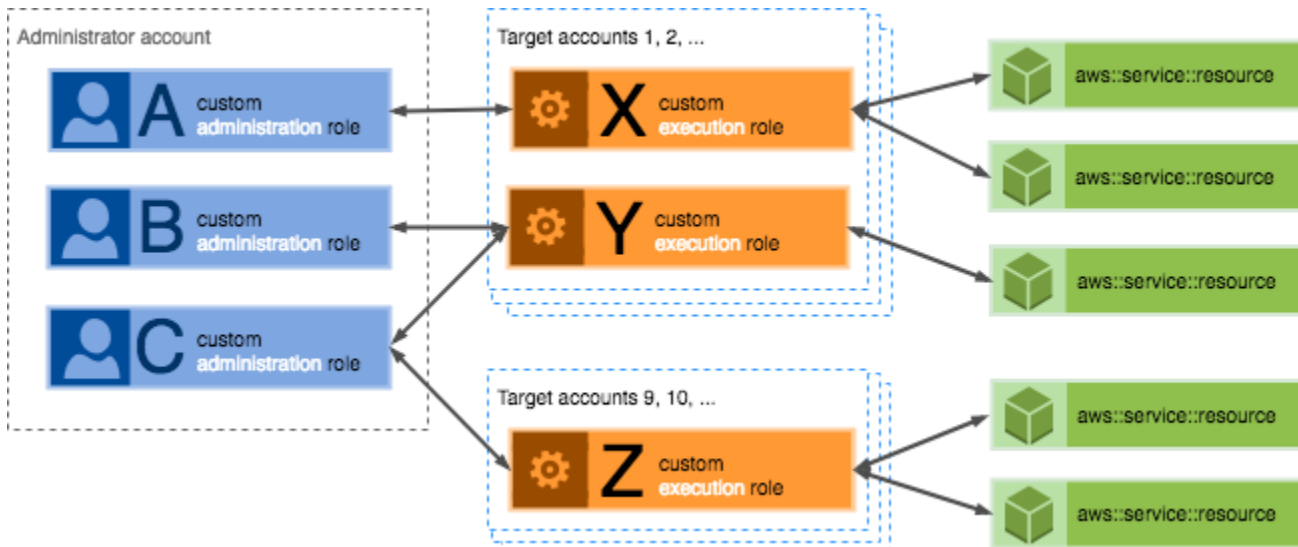
信頼関係を定義するロールを作成する際、次の信頼ポリシーを指定する必要があります。

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

ユーザーが特定の StackSets に含めることができるリソースを制御する

カスタマイズされた実行ロールを使用して、ユーザーおよびグループが StackSets に含めることができるスタックリソースを制御します。例えば、作成した StackSets に Amazon S3 関連のリソースのみを含めることができるグループを設定し、別のチームには DynamoDB リソースのみを含めることができます。これを行うには、各グループのカスタマイズされた管理ロールと各リソースセットのカスタマイズされた実行ロールとの間に信頼関係を作成します。カスタマイズされた実行ロールは、どのスタックリソースを StackSets に含めることができるかを定義します。カスタマイズされた管理ロールは管理者アカウントにあり、カスタマイズされた実行ロールは定義されたリソースを使用して StackSets を作成する各ターゲットアカウントにあります。次に、StackSets オペレーションを実行するときにカスタマイズされた管理ロールを使用するように特定のユーザーとグループをアクティブ化します。

例えば、管理アカウントでカスタマイズされた管理者ロール A、B、C を作成できます。ロール A を使用するアクセス許可を持つユーザーおよびグループは、カスタマイズされた実行ロール X に明確にリストされているスタックリソースを含む StackSets を作成できますが、ロール Y または Z のリソースや実行ロールに含まれていないリソースは含まれません。



StackSet を更新する場合、ユーザーはカスタマイズされた管理ロールを明示的に指定する必要があります (この StackSet で以前に使用したのと同じカスタマイズされた管理ロールである場合でも必要です)。ユーザーが StackSet に対してオペレーションを実行するアクセス許可を持っている場合、CloudFormation は指定のカスタマイズされた管理ロールを使用して更新を実行します。

同様に、ユーザーはカスタマイズされた実行ロールを指定することもできます。カスタマイズされた実行ロールを指定した場合、CloudFormation は、上記の要件に従って、スタックを更新するロールを使用します。ユーザーがカスタマイズされた実行ロールを指定しない場合、CloudFormation は、StackSet でオペレーションを実行するアクセス許可がユーザーに付与されている限り、StackSet に以前関連付けられていた、カスタマイズされた実行ロールを使用して更新を行います。

Administrator account

[「特定のターゲットアカウントで StackSet オペレーションを実行できるユーザーを制御する」](#)

で説明されているように、カスタマイズされた管理ロールを管理者アカウントで作成します。カスタマイズされた管理ロールと、使用するカスタマイズされた実行ロールとの間に信頼関係を含めます。

Example アクセス許可ポリシーの例。

次の例は、ターゲットアカウントのために定義された `AWSCloudFormationStackSetExecutionRole` とカスタマイズされた実行ロールの両方のための許可ポリシーです。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "Stmt1487980684000",
    "Effect": "Allow",
    "Action": [
      "sts:AssumeRole"
    ],
    "Resource": [
      "arn:aws:iam::*:role/AWSCloudFormationStackSetExecutionRole",
      "arn:aws:iam::*:role/custom_execution_role"
    ]
  }
]
```

Target accounts

StackSets を作成するターゲットアカウントで、ユーザーとグループが StackSets に含めることができるようにするサービスとリソースにアクセス許可を付与するカスタマイズされた実行ロールを作成します。

Example アクセス許可ポリシーの例。

次の例では、StackSets の最小限のアクセス許可に加えて、Amazon DynamoDB テーブルを作成するアクセス許可を付与しています。

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

```
    ],  
    "Resource": "*"    
  }  
]  
}
```

Example信頼ポリシーの例

信頼関係を定義するロールを作成する際、次の信頼ポリシーを指定する必要があります。

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

特定の StackSet オペレーションのアクセス許可を設定する

さらに、StackSets またはスタックインスタンスの作成、更新、削除など、特定の StackSet オペレーションを実行できるユーザーとグループのアクセス許可を設定できます。詳細については、「サービス認可リファレンス」の「[CloudFormation のアクション、リソース、および条件キー](#)」を参照してください。

混乱した代理問題を軽減するためにグローバルキーを設定する

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。AWS では、サービス間でのなりすましによって、混乱した代理問題が発生する場合があります。サービス間でのなりすましは、あるサービス (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が操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐため、AWS では、アカウント内のリソースへのアクセス許可が付与され

たサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで [aws:SourceArn](#) および [aws:SourceAccount](#) のグローバル条件コンテキストキーを使用して、CloudFormation StackSets が別のサービスに付与するアクセス許可をそのリソースに制限することをお勧めします。両方のグローバル条件コンテキストキーを同じポリシーステートメントで使用する場合は、aws:SourceAccount 値と、aws:SourceArn 値に含まれるアカウントが、同じアカウント ID を示している必要があります。

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 グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合は、aws:SourceArn グローバルコンテキスト条件キーを使用して、ARN の未知部分をワイルドカード (*) で表します。例えば、arn:aws:*cloudformation*::*123456789012*:* です。より具体的であるため、可能な限り aws:SourceArn を使用してください。aws:SourceAccount は、正しい ARN または ARN パターンを判別できない場合にのみ使用してください。

StackSets が [administrator] (管理者) アカウントで [Administration] (管理者) ロールを引き受けるとき、StackSets は [administrator] (管理者) アカウント ID と StackSets Amazon リソースネーム (ARN) を入力します。したがって、信頼関係で [グローバルキー](#) aws:SourceAccount および aws:SourceArn の条件を定義して、[混乱した代理問題](#)を防ぐことができます。次の例では、StackSets で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。

Administrator account

Example **aws:SourceAccount** および **aws:SourceArn** のグローバルキー

StackSets を使用する場合は、混乱した代理の問題を防ぐために、AWSCloudFormationStackSetAdministrationRole 信頼ポリシーでグローバルキー aws:SourceAccount および aws:SourceArn を定義します。

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

```

    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:cloudformation:*:111122223333:stackset/*"
      }
    }
  }
]
}

```

Example StackSets ARN

より細かく制御するために、関連する StackSets ARN を指定します。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333",
          "aws:SourceArn": [
            "arn:aws:cloudformation:STACKSETS-REGION:111122223333:stackset/STACK-SET-ID-1",
            "arn:aws:cloudformation:STACKSETS-REGION:111122223333:stackset/STACK-SET-ID-2"
          ]
        }
      }
    }
  ]
}

```



```
}
```

AWS Organizations を使用して StackSets のための信頼されたアクセスをアクティブ化する

このトピックには、サービスマネージド許可を使用して、アカウントおよび AWS リージョン 間でデプロイするために StackSets によって必要とされる信頼されたアクセスを AWS Organizations でアクティブ化する方法の手順が記載されています。セルフマネージド許可を使用するには、代わりに「[セルフマネージドのアクセス許可の付与](#)」を参照してください。

サービスマネージド型のアクセス許可を持つ StackSet を作成する前に、まず次のタスクを実行する必要があります。

- AWS Organizations で[すべての機能を有効にします](#)。一括請求 (コンソリデेटィッドビルディング) 機能のみが有効になっている場合、サービスマネージド型のアクセス許可を持つ StackSet を作成することはできません。
- AWS Organizations の信頼されたアクセスをアクティブ化します。このアクションにより、CloudFormation は管理アカウントにサービスリンクロールを作成できます。信頼されたアクセスをアクティブ化した後に、サービスマネージド型のアクセス許可を持つ StackSet を作成すると、CloudFormation は必要なサービスリンクロールと、stacksets-exec-* という名前のサービスロールの両方をターゲット (メンバー) アカウントに作成します。

信頼されたアクセスをアクティブ化すると、管理アカウントと委任管理者アカウントは、組織のサービスマネージド型の StackSets を作成して管理できます。

信頼されたアクセスをアクティブ化するには、管理アカウントの管理者ユーザーである必要があります。管理者ユーザーは、AWS アカウント に対するフルの許可を持つユーザーです。詳細については、「AWS アカウント管理 リファレンスガイド」の「[Create an administrator user](#)」を参照してください。管理アカウントのセキュリティを保護するための推奨事項については、「AWS Organizations ユーザーガイド」の「[管理アカウントのベストプラクティス](#)」を参照してください。

信頼されたアクセスをアクティブ化するには

1. 管理アカウントの管理者として AWS にサインインし、<https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. ナビゲーションペインから [StackSets] を選択します。信頼されたアクセスを非アクティブ化した場合、信頼されたアクセスを有効にするように求めるバナーが表示されます。

CloudFormation > StackSets

 Activate trusted access with AWS Organizations to use service-managed permissions. [Learn more](#)[Activate trusted access](#)

3. [信頼されるアクセスのアクティブ化] を選択します。

次のバナーが表示されると、信頼されたアクセスが正常にアクティブ化になります。

 Successfully activated trusted access with AWS Organizations. To change this setting, please work through the AWS Organizations service console.

✕

Note

[組織アクセスのアクティブ化] は [組織アクセスの有効化] と同じであり、[組織アクセスの非アクティブ化] は [組織アクセスの無効化] と同じです。これらの条件は、マーケティングガイドラインに基づいて更新されました。

信頼されたアクセスを非アクティブ化する方法

「AWS Organizations ユーザーガイド」の「[CloudFormation StackSets と AWS Organizations](#)」を参照してください。

AWS Organizations を使用する信頼されたアクセスを非アクティブ化にする前、委任されたすべての管理者の登録を解除する必要があります。詳細については、「[委任された管理者の登録](#)」を参照してください。

Note

コンソールの代わりに API オペレーションを使用して信頼されたアクセスをアクティブ化または非アクティブ化する方法については、次を参照してください:

- [ActivateOrganizationsAccess](#)
- [DeactivateOrganizationsAccess](#)
- [DescribeOrganizationsAccess](#)

サービスリンクロール

管理アカウントは、`AWSServiceRoleForCloudFormationStackSetsOrgAdmin` サービスにリンクされたロールを使用します。AWS Organizations を使用する信頼されたアクセスが非アクティブ化になっている場合のみ、このロールを変更または削除できます。

各ターゲットアカウントは、`AWSServiceRoleForCloudFormationStackSetsOrgMember` サービスにリンクされたロールを使用します。2 つの条件下 (AWS Organizations を使用する信頼されたアクセスが非アクティブ化されているか、アカウントがターゲット組織または組織単位 (OU) から削除されている場合) でのみ、このロールを変更または削除できます。

詳細については、「AWS Organizations ユーザーガイド」の「[CloudFormation StackSets と AWS Organizations](#)」を参照してください。

委任された管理者メンバーアカウントを登録する

組織の管理アカウントの他に、委任管理者アクセス許可を持つメンバーアカウントは、組織に対するサービスマネージド型のアクセス許可を使用して StackSets を作成および管理できます。サービスマネージド型のアクセス許可を持つ StackSets は、委任管理者によって作成された StackSets を含め、管理アカウントに作成されます。組織の委任された管理者として登録するには、組織内にメンバーアカウントが必要です。組織に参加する方法については、「[Inviting an AWS アカウント to join your organization](#)」を参照してください。

組織では、登録済みの委任された管理者を一度に最大 5 人設定できます。委任された管理者は、組織内のすべてのアカウントにデプロイするか、特定の OU にデプロイするかを選択できます。委任された管理者が組織によって管理されるアカウントにデプロイする前に、AWS Organizations を使用する信頼されたアクセスをアクティブ化する必要があります。詳細については、「[AWS Organizations を使用して StackSets のための信頼されたアクセスをアクティブ化する](#)」を参照してください。

Important

以下の点に注意してください。

- 委任された管理者は、組織内のアカウントにデプロイするための完全なアクセス許可を持っています。管理アカウントでは、特定の OU にデプロイしたり、特定の StackSet のオペレーションを実行したりする、委任管理者のアクセス許可を制限することはできません。

- 潜在的なエラーを回避するため、委任された管理者に `organizations:ListDelegatedAdministrators` アクセス許可が付与されていることを確認します。

組織の委任された管理者は、米国東部 (オハイオ)、米国東部 (バージニア北部)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)、アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京)、カナダ (中部)、欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (ロンドン)、欧州 (パリ)、欧州 (ストックホルム)、イスラエル (テルアビブ)、南米 (サンパウロ)、AWS GovCloud (米国東部)、AWS GovCloud (US-West) の各リージョンに登録できます。

[CloudFormation コンソール](#)、[AWS CLI](#)、または [AWS SDK](#) を使用して、委任された管理者の登録および登録解除を行うことができます。

委任された管理者を登録するには (コンソール)

1. 管理アカウントの管理者として AWS にサインインし、<https://console.aws.amazon.com/cloudformation/> で CloudFormation コンソールを開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. [委任された管理者] で、[委任された管理者の登録] を選択します。
4. [委任された管理者の登録] ダイアログボックスで、[委任された管理者の登録] を選択します。

成功のメッセージは、メンバーアカウントが委任された管理者として正常に登録されたことを示します。

委任された管理者の登録を解除するには (コンソール)

1. 管理アカウントの管理者として AWS にサインインし、<https://console.aws.amazon.com/> で CloudFormation コンソールを開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. [委任された管理者] で、登録解除するアカウントを選択し、[登録の解除] を選択します。

成功のメッセージは、メンバーアカウントが委任された管理者として正常に登録解除されたことを示します。

このアカウントはいつでも再登録できます。

委任された管理者を登録する (AWS CLI)

1. AWS CLI を開きます。
2. `register-delegated-administrator` コマンドを実行します。

```
$ aws organizations register-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. 指定されたメンバーアカウントが委任された管理者として正常に登録されていることを確認するには、`list-delegated-administrators` コマンドを実行します。

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

委任された管理者の登録を解除する (AWS CLI)

1. AWS CLI を開きます。
2. `deregister-delegated-administrator` コマンドを実行します。

```
$ aws organizations deregister-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. 指定されたメンバーアカウントが委任された管理者として正常に登録解除されていることを確認するには、`list-delegated-administrators` コマンドを実行します。

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

このアカウントはいつでも再登録できます。

サンプルテンプレートを使用して StackSets の使用を開始する

このチュートリアルは、AWS マネジメントコンソール を使用して StackSets の使用を開始するのに役立ちます。サンプルテンプレートを使用して StackSet を作成する手順を説明します。複数のリージョンにスタックをデプロイし、StackSet オペレーションをモニタリングして、結果を表示する方法について説明します。

このチュートリアルでは、米国西部 (オレゴン) リージョン (us-west-2) および米国東部 (バージニア北部) リージョン (us-east-1) 内の AWS アカウント で AWS Config を有効にする StackSet を作成します。StackSets を使用すると、1 回のオペレーションで複数のアカウントとリージョンにまたがるスタックを作成、更新、または削除できるため、大規模なインフラストラクチャを管理するのに最適なソリューションです。このチュートリアルでは、簡素化のために 1 つのアカウントを使用していますが、StackSets のマルチリージョン機能を効果的に示す内容となっています。

サンプルテンプレートは、次の S3 バケットにあります: <https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfig.yml>。

Note

StackSets は無料ですが、このチュートリアルで作成する AWS Config などの AWS リソースに対して課金されます。詳細については、「[AWS Config 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [コンソールからサンプルテンプレートを使用して StackSet を作成する](#)
- [StackSet の作成をモニタリングする](#)
- [StackSet の結果を表示する](#)
- [StackSet を更新する](#)
- [スタックを StackSet に追加する](#)
- [クリーンアップ](#)
- [次のステップ](#)

前提条件

このチュートリアル開始する前に、以下の前提条件を満たしていることを確認してください。

- セルフマネージド型のアクセス許可に必要な IAM ロールを設定しておく必要があります。StackSet を作成して 1 つのアカウント内にスタックをデプロイするには、アカウントで次のロールが必要です。
 - `AWSCloudFormationStackSetAdministrationRole`
 - `AWSCloudFormationStackSetExecutionRole`

これらのロールの設定手順の詳細については、「[セルフマネージドのアクセス許可の付与](#)」を参照してください。

コンソールからサンプルテンプレートを使用して StackSet を作成する

AWS Config を有効にする StackSet を作成するには

1. [クラウドフォーメーション コンソール](#)を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を管理する AWS リージョン を選択します。

StackSets がサポートされているリージョンはいずれも選択できます。選択したリージョンは、StackSet でデプロイできるリージョンには影響しません。
3. ナビゲーションペインから [StackSets] を選択します。
4. [StackSets] ページの上部から [StackSets の作成] を選択します。
5. [アクセス許可] で、[セルフサービスのアクセス許可] を選択し、前提条件で作成した IAM ロールを選択します。
 - IAM 管理者ロールで、[AWSCloudFormationStackSetAdministrationRole] を選択します。
 - IAM 実行ロール名で、[AWSCloudFormationStackSetExecutionRole] を選択します。
6. [Prerequisite - Prepare template (前提条件 - テンプレートの準備)] で [Use a sample template (サンプルテンプレートの使用)] を選択します。
7. [サンプルテンプレートの選択] で、ドロップダウンメニューから [AWS Config の有効化] テンプレートを選択します。その後、[Next] を選択します。

このテンプレートは、設定レコーダーや配信チャネルなど、アカウントで AWS Config を有効にするために必要なリソースを作成します。
8. [StackSet の詳細を指定] ページで、[StackSet 名] に **my-awsconfig-stackset** と入力します。
9. [StackSet の説明] には、**A StackSet that enables Config across multiple Regions** と入力します。
10. [パラメータ] で、次のように AWS Config 設定を行います。
 - a. [すべてのリソースタイプをサポート] で、デフォルト値の true のままにして、サポートされているすべてのリソースタイプを記録します。

- b. [グローバルリソースタイプを含める] で、デフォルト値の `false` のままにして、IAM ロールなどのグローバルリソースを除外します。
- c. [すべてがサポートされていない場合のリソースタイプのリスト] を `<All>` のままにしておきます
- d. [Config サービスにリンクされたロールのリソースを含むリージョン] で、`<DeployToAnyRegion>` を `us-west-2` に置き換えます。

つまり、`AWSServiceRoleForConfig` という名前のサービスにリンクされたロールは、スタックが米国西部 (オレゴン) リージョンにデプロイされている場合にのみ作成されます。デプロイリージョンは、この手順の後半で選択します。

- e. [設定レコーダーの記録頻度] で、[DAILY] 記録を選択します。
11. [次へ] を選択して続行します。
 12. [StackSet オプションの設定] ページで、[新しいタグの追加] を選択し、キーと値のペアを指定してタグを追加します。
 - a. [キー] に「**Stage**」と入力します。
 - b. [Value (値)] に「**Test**」と入力します。

StackSets に適用するタグは、スタックによって作成されるリソースに適用されます。

13. [実行設定] で、[アクティブ] を選択して CloudFormation の最適化されたオペレーション処理を有効にします。
 - 競合しないオペレーションは、デプロイ時間を短縮するために同時に実行されます。
 - 競合するオペレーションは自動的にキューに入れられ、リクエストされた順序で処理されます。

オペレーションが実行中またはキューに入れられている間、CloudFormation ではオペレーションが競合しない場合でもすべての受信オペレーションがキューに入ります。この間は実行設定を変更できません。

14. [次へ] を選択します。
15. [デプロイオプションの設定] ページの [StackSet にスタックを追加] で、[新しいスタックのデプロイ] オプションを選択します。
16. [アカウント] で [Deploy stacks in accounts (スタックをアカウントにデプロイ)] を選択します。
17. テキストボックスに AWS アカウント ID を入力します。

18. [リージョンの指定] では、次のリージョンをこの順序で選択します。

1. 米国西部 (オレゴン) リージョン (us-west-2)
2. 米国東部 (バージニア北部) リージョン (us-east-1)

必要に応じて、米国西部 (オレゴン) リージョンの横にある上矢印を使用して、リストの最初の項目になるように移動します。リージョンの順序に従って、デプロイの順序が決まります。

19. [デプロイオプション] では、次の設定を行います。

- a. [Maximum concurrent accounts (同時アカウントの最大数)] については、デフォルト値の [数値] と 1 をそのまま使用します。

つまり、マルチアカウントデプロイの場合、この設定では、CloudFormation は一度に 1 つのアカウントのみでスタックをデプロイします。

- b. [Failure tolerance (耐障害性)] については、デフォルトの [数値] と [0] をそのまま使用します。

つまり、CloudFormation が現在のリージョンでデプロイを停止し、残りのリージョンでのデプロイをキャンセルする前に、最大で 0 個のスタックデプロイが、指定されたリージョンのうちいずれかで失敗できるということを意味します。

- c. [リージョンの同時実行] では、[順次] (デフォルト) を選択します。

この設定により、CloudFormation は 1 つのリージョンでデプロイを完了してから次のリージョンに移動します。

- d. [同時実行モード] では、[厳格な耐障害性] のデフォルト設定を維持します。

マルチアカウントデプロイの場合、これにより障害発生時のアカウントの同時実行レベルが下がり、[耐障害性] +1 の範囲内に維持されます。

20. [次へ] を選択します。

21. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。

22. StackSet を作成する準備ができたなら、[送信] を選択します。

StackSet の作成をモニタリングする

[送信] を選択すると、CloudFormation は StackSet の作成と、アカウント内の指定されたリージョンへのスタックのデプロイを開始します。StackSet の詳細ページが自動的に開き、オペレーションの進行状況をモニタリングできます。

StackSet の作成をモニタリングするには

1. StackSet の詳細ページで、[オペレーション] タブがデフォルトで表示され、進行中の現在のオペレーションが表示されます。
2. オペレーションのステータスは、最初は RUNNING である必要があります。CloudFormation は、設定したデプロイオプションに従って、指定したリージョンにスタックを作成します。
3. オペレーションの詳細については、リストでオペレーション ID を選択します。
4. オペレーションの詳細ページで、各リージョンで作成されているスタックインスタンスのステータスを表示できます。
5. オペレーションステータスが SUCCEEDED に変わるまで待ちます。これは、StackSet とそのすべてのスタックインスタンスが正常に作成されたことを示します。

StackSet の結果を表示する

StackSet の作成が完了したら、デプロイされたスタックインスタンスを表示し、指定したリージョンのアカウントで AWS Config が有効になっていることを確認できます。

StackSet の結果を表示するには

1. StackSet の詳細ページで、[スタックインスタンス] タブを選択します。
2. 指定したリージョンのアカウントで作成されたスタックインスタンスのリストが表示されます。各スタックインスタンスのステータスは SUCCEEDED である必要があり、これは正常にデプロイされたことを示します。
3. アカウントで AWS Config が有効になっていることを確認するには、デプロイされた各リージョンで AWS Config コンソールを確認します。

StackSet を更新する

StackSet を作成したら、それを更新してパラメータ値を変更したり、リージョンを追加したりできます。このセクションでは、AWS Config 記録頻度パラメータを更新する方法について説明します。

StackSet を更新するには

1. [StackSets] ページで、**my-awsconfig-stackset** を選択します。
2. StackSet を選択した状態で、[アクション] メニューから [StackSet の詳細を編集] を選択します。
3. [テンプレートの選択] ページの [前提条件 - テンプレートの準備] で、[現在のテンプレートの使用] を選択します。
4. [次へ] を選択します。
5. [StackSet の詳細を指定] ページの [パラメータ] で、[設定レコーダーの記録頻度] を見つけて、[DAILY] から [CONTINUOUS] に変更します。
6. [次へ] を選択します。
7. [StackSet オプションの設定] ページで、設定をそのままにして [次へ] を選択します。
8. [デプロイオプションの設定] ページで、アカウント ID と StackSet の作成時に使用したのと同じリージョンを指定します。
9. [デプロイオプション] では、以前と同じ設定のままにします。
10. [次へ] を選択します。
11. [レビュー] ページで変更内容を確認し、[送信] を選択します。
12. CloudFormation が StackSet の更新を開始します。StackSet の詳細ページの [オペレーション] タブで進行状況をモニタリングできます。

スタックを StackSet に追加する

追加のリージョンにデプロイすることで、StackSet にさらにスタックを追加できます。このセクションでは、新しいリージョンにスタックを追加する方法を示します。

StackSet にスタックを追加するには

1. [StackSets] ページで、**my-awsconfig-stackset** を選択します。
2. StackSet を選択した状態で、[アクション] メニューから [StackSet にスタックを追加] を選択します。
3. [デプロイオプションの設定] ページの [StackSet にスタックを追加] で、[新しいスタックのデプロイ] オプションを選択します。
4. [アカウント] で、[スタックをアカウントにデプロイ] を選択し、アカウント ID を入力します。

5. [リージョンの指定] には、[欧州 (アイルランド)] (eu-west-1) などの新しいリージョンを選択します。
6. [デプロイオプション] では、以前と同じ設定のままにします。
7. [次へ] を選択します。
8. [上書きの指定] ページでは、プロパティ値を指定されたままにして、[次へ] を選択します。
9. [確認] ページで選択内容を確認し、[送信] を選択します。
10. CloudFormation は、指定されたリージョンで新しいスタックの作成を開始します。StackSet の詳細ページの [オペレーション] タブで進行状況をモニタリングできます。

クリーンアップ

不要な AWS Config リソースに対して料金が発生しないようにするには、StackSet からスタックを削除し、StackSet 自体を削除し、このチュートリアル用に作成した IAM ロールを削除してクリーンアップする必要があります。すべてのリソースはアカウント内にデプロイされるため、クリーンアップは簡単です。

StackSet からスタックを削除するには

1. [StackSets] ページで、**my-awsconfig-stackset** を選択します。
2. StackSet を選択した状態で、[アクション] メニューから [StackSet からスタックを削除] を選択します。
3. [デプロイオプションの設定] ページの [アカウント] で、[スタックをアカウントにデプロイ] を選択し、アカウント ID を入力します。
4. [リージョンの指定] で、スタックをデプロイしたすべてのリージョンを選択します。
5. [デプロイオプション] では、デフォルト設定のままにします。
6. スタックとそのリソースが削除されるように、[スタックを保持] が有効になっていないことを確認します。
7. [次へ] を選択します。
8. [確認] ページで選択内容を確認し、[送信] を選択します。
9. CloudFormation は StackSet からのスタックの削除を開始します。StackSet の詳細ページの [オペレーション] タブで進行状況をモニタリングできます。

StackSet を削除するには

1. すべてのスタックが削除されたら、[StackSets] ページで **my-awsconfig-stackset** を選択します。
2. StackSet を選択した状態で、[アクション] メニューから [StackSet の削除] を選択します。
3. 確認を求められたら、[削除] を選択します。

IAM サービスロールを削除するには

一つのアカウトにのみデプロイしたため、このアカウトから IAM ロールを削除するだけで済み、マルチアカウトデプロイに比べてクリーンアップが非常に簡単です。

1. [IAM コンソール](#) を開きます。
2. ナビゲーションペインで、[Roles] を選択します。
3. 検索ボックスに **AWSCloudFormationStackSet** と入力して、このチュートリアル用に作成したロールを見つけます。
4. [AWSCloudFormationStackSetAdministrationRole] の横にあるチェックボックスをオンにします。
5. ページの上部から、[削除] を選択します。
6. 確認ダイアログボックスで、**delete** と入力し、[削除] を選択します。
7. 同じプロセスを繰り返して、[AWSCloudFormationStackSetExecutionRole] を削除します。

StackSet を削除すると、AWS::S3::Bucket リソースの DeletionPolicy 属性により、Amazon S3 バケットが各 AWS リージョンに残ります。これにより、AWS Config 履歴データが保持されます。このデータが不要になった場合は、バケットを手動で削除しても問題ありません。バケットを削除するには、バケットを空にする必要があります。バケットを空にすると、バケット内のすべてのオブジェクトが削除されます。

Amazon S3 バケットを空にして削除するには

1. [Amazon S3 コンソール](#) を開きます。
2. コンソールの左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストには、デプロイした各リージョンでこの StackSet 用に作成されたバケットが表示されます。この StackSet 用に作成されたバケットの名前の横にあるオプションを選択し、[空] を選択します。

4. [バケットを空にする] ページで、テキストフィールドに **permanently delete** と入力して、[空にする] をクリックします。
5. [バケットを空にする: ステータス] ページで、バケットを空にするプロセスの進行状況をモニタリングします。
6. バケットリストに戻るには、[Exit (終了)] を選択します。
7. バケットの名前の横にあるオプションを選択し、[削除] を選択します。
8. 確認を求められたら、バケットの名前を入力し、[バケットを削除] を選択します。
9. [バケット] リストで、バケット削除プロセスの進捗状況を確認します。Amazon S3 がバケットの削除を完了すると、バケットはリストから消えます。
10. 異なるリージョンで StackSet によって作成された各バケットに対して、このプロセスを繰り返します。

次のステップ

お疲れ様でした。サンプルテンプレートを使用した StackSet の作成、アカウント内の複数のリージョンへのスタックのデプロイ、StackSet の更新、スタックの追加、リソースのクリーンアップが正常に完了しました。シングルアカウントデプロイに焦点を当てることで、StackSets のマルチリージョン機能の核心を学習しながら、クリーンアッププロセスを簡素化しました。

StackSets の詳細については、次のトピックをご覧ください。

- [CloudFormation StackSet 内のスタックのパラメータ値を上書きする](#) – 特定のアカウントとリージョンのパラメータ値を上書きする方法を説明します。
- [サービスマネージド型のアクセス許可を持つ CloudFormation StackSets の作成](#) – AWS Organizations を使用したマルチアカウントデプロイ用の StackSets の作成について説明します。

セルフマネージド型のアクセス許可を持つ CloudFormation StackSets の作成

このトピックでは、AWS アカウント およびリージョンにスタックをデプロイするためのセルフマネージド型のアクセス許可を持つ StackSets を作成する方法について説明します。

Note

続行する前に、StackSets に必要な IAM サービスロールを作成して、StackSet の管理元アカウントとスタックのデプロイ先のアカウントとの間に信頼関係を確立します。詳細については、「[セルフマネージド型のアクセス許可を付与する](#)」を参照してください。

トピック

- [セルフマネージド型のアクセス許可を持つ StackSet の作成 \(コンソール\)](#)
- [セルフマネージド型のアクセス許可を持つ StackSet の作成 \(AWS CLI\)](#)

セルフマネージド型のアクセス許可を持つ StackSet の作成 (コンソール)

StackSet を作成するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を管理する AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。
4. [StackSets] ページの上部から [StackSets の作成] を選択します。
5. [アクセス許可] で、[セルフサービスのアクセス許可] を選択し、作成した IAM ロールを選択します。
6. [前提条件 - テンプレートの準備] で、[テンプレートの準備完了] を選択します。
7. [Specify template (テンプレートの指定)] で、スタックテンプレートを含む S3 バケットの URL を指定するか、スタックテンプレートファイルをアップロードするかを選択します。その後、[Next] を選択します。
8. [StackSet の詳細を指定] ページで、StackSet の名前を指定し、パラメータを指定して、[次へ] を選択します。
9. [次へ] を選択して続行します。
10. [StackSet オプションの設定] ページの [タグ] で、スタックのリソースに適用するタグを指定します。AWS でのタグの使用方法に関する詳細については、AWS Billing and Cost Management ユーザーガイドの「[AWS コスト配分タグを使用してコストを整理および追跡する](#)」を参照してください。

11. [実行設定] で、[アクティブ] を選択して CloudFormation の最適化されたオペレーション処理を有効にします。

- 競合しないオペレーションは、デプロイ時間を短縮するために同時に実行されます。
- 競合するオペレーションは自動的にキューに入れられ、リクエストされた順序で処理されます。

オペレーションが実行中またはキューに入れられている間、CloudFormation ではオペレーションが競合しない場合でもすべての受信オペレーションがキューに入ります。この間は実行設定を変更できません。

12. テンプレートに IAM リソースが含まれる場合は、[機能] で[I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、[「CloudFormation テンプレートでの IAM リソースの承認」](#)を参照してください。

13. [Next] (次へ) を選択します。

14. [デプロイオプションの設定] ページの [StackSet にスタックを追加] で、[新しいスタックのデプロイ] オプションを選択します。

15. [アカウント] で [Deploy stacks in accounts (スタックをアカウントにデプロイ)] を選択します。ターゲット AWS アカウント 番号をテキストボックスに貼り付け、複数の数字をカンマで区切ります。

 Note

管理者アカウントにもスタックをデプロイする場合は、管理者アカウント ID を含めることができます。

16. [リージョンの指定] で、スタックをデプロイするリージョンを選択します。

17. [デプロイオプション] で、次の操作を行います。

- [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。
- [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。
- [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。

- [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
- [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
- [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。

18. [次へ] を選択します。

19. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。

20. StackSet を作成する準備ができたなら、[送信] を選択します。

CloudFormation で StackSet の作成が開始されます。StackSet 内のスタック作成の進捗と状況は、[送信] を選択した際に開く StackSet 詳細ページで確認できます。

セルフマネージド型のアクセス許可を持つ StackSet の作成 (AWS CLI)

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- StackSet コンテナを作成します。
- スタックインスタンスをデプロイします。

StackSet を作成するには

1. [create-stack-set](#) コマンドを使用して、*my-stackset* という名前の新しい StackSet を作成します。次の例では、S3 バケットに保存されているテンプレートを使用し、パラメーターとして *KeyPairName* に *TestKey* という値を設定しています。

```
aws cloudformation create-stack-set \  
  --stack-set-name my-stackset \  
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/MyApp.template \  
  --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. `create-stack-set` コマンドが完了した後、[list-stack-sets](#) コマンドを実行して StackSet が作成されたことを確認します。新しい StackSet が表示結果で確認できます。

```
aws cloudformation list-stack-sets
```

3. [create-stack-instances](#) コマンドを使用して、StackSet 内にスタックをデプロイします。次の例では、2 つのリージョン (*us-west-2* と *us-east-1*) にわたって 2 つの AWS アカウント (*account_ID_1* と *account_ID_2*) にスタックをデプロイします。

--operation-preferences オプションを使用して、同時アカウント処理およびその他のデプロイ設定を設定します。この例では、カウントベースの設定を使用します。MaxConcurrentCount は FailureToleranceCount + 1 を超えることはできません。パーセンテージベースの設定の場合は、代わりに FailureTolerancePercentage または MaxConcurrentPercentage を使用します。

```
aws cloudformation create-stack-instances \  
  --stack-set-name my-stackset \  
  --accounts account_ID_1 account_ID_2 \  
  --regions us-west-2 us-east-1 \  
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

詳細については、「AWS CloudFormation API リファレンス」の「[CreateStackInstances](#)」を参照してください。

4. [describe-stack-set-operation](#) コマンドを使用して、スタックが正常に作成されたことを確認します。--operation-id オプションで、create-stack-instances 出力の一部として返されたオペレーション ID を指定します。

```
aws cloudformation describe-stack-set-operation \  
  --stack-set-name my-stackset \  
  --operation-id operation_ID
```

サービスマネージド型のアクセス許可を持つ CloudFormation StackSets の作成

サービスマネージド型のアクセス許可を使用すると、AWS Organizations が管理する特定のリージョン内のアカウントにスタックをデプロイできます。このモデルを使用すると、ターゲットアカウントと AWS リージョンそれぞれに IAM ロールを作成する必要がなくなります。IAM ロールは CloudFormation がユーザーに代わって作成します。詳細については、「[信頼されたアクセスをアクティブ化する](#)」を参照してください。

トピック

- [考慮事項](#)
- [サービスマネージド型のアクセス許可を持つ StackSet の作成 \(コンソール\)](#)
- [サービスマネージド型のアクセス許可を持つ StackSet 作成 \(AWS CLI\)](#)

考慮事項

サービスマネージド型の許可を持つ StackSet を作成する前に、次の点を考慮してください。

- サービスマネージド型の許可を持つ StackSet は、組織の管理アカウントまたは委任管理者アカウントのいずれかが開始できますが、すべてのオペレーションは管理アカウントが実行します。
- CloudFormation は、管理アカウントが組織の一部であったり、組織単位 (OU) に属していたりする場合でも、スタックを管理アカウントにデプロイしません。
- StackSet は、組織全体 (すべてのアカウントを含む) または指定した OU をターゲットにすることができます。StackSet が親 OU をターゲットにするときは、子 OU も自動的にターゲットになります。デフォルトで、StackSet が特定の OU をターゲットにする場合は、それらの OU 内にあるすべてのアカウントもターゲットになります。ただし、アカウントフィルターオプションを使用して特定のアカウントをターゲットにすることはできます。
- 複数の StackSets で同じ組織または OU をターゲットにすることができます。
- 組織外のアカウントをターゲットにすることはできません。
- StackSet にデプロイする認可は、管理アカウントへのサインインに使用される、IAM プリンシパル (ユーザー、ロール、グループ) に割り当てられた許可に応じて異なります。組織にデプロイする許可を付与する IAM ポリシーの例については、「[リージョンとリソースタイプに基づいてスタックセットのオペレーションを制限する](#)」を参照してください。
- 委任管理者は、組織内の任意のアカウントにデプロイするための完全な許可を持っています。管理アカウントは、特定の OU や StackSet オペレーションにデプロイする委任管理者の許可を制限できません。
- 自動デプロイ設定は StackSet レベルで適用されます。OU、アカウント、リージョンを選択して自動デプロイを調整することはできません。
- サービスマネージド型の許可を使用する StackSet は、マクロまたは変換が含まれるネストされたスタックやテンプレートをサポートしません。

サービスマネージド型のアクセス許可を持つ StackSet の作成 (コンソール)

StackSet を作成するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を管理する AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。
4. [StackSets] ページの上部から [StackSets の作成] を選択します。
5. [アクセス権限] で、[Service-managed permissions (サービスマネージド型のアクセス許可)] を選択します。

Note

AWS Organizations との信頼されたアクセスが無効になっている場合は、バナーが表示されます。サービスマネージド型のアクセス許可を持つ StackSet を作成または更新するには、信頼されたアクセスが必要です。組織の管理カウントの管理者のみが、[AWS Organizations](#) を使用して [StackSets のための信頼されたアクセスをアクティブ化する](#) のアクセス許可を持ちます。

6. [前提条件 – テンプレートの準備] で、[テンプレートの準備完了] を選択します。
7. [Specify template (テンプレートの指定)] で、スタックテンプレートを含む S3 バケットの URL を指定するか、スタックテンプレートファイルをアップロードするかを選択します。その後、[Next] を選択します。
8. [StackSet の詳細を指定] ページで、StackSet の名前を指定し、パラメータを指定して、[次へ] を選択します。
9. [StackSet オプションの設定] ページの [タグ] で、スタックのリソースに適用するタグを指定します。AWS でのタグの使用方法に関する詳細については、AWS Billing and Cost Management ユーザーガイドの「[AWS コスト配分タグを使用してコストを整理および追跡する](#)」を参照してください。
10. [実行設定] で、[アクティブ] を選択して CloudFormation の最適化されたオペレーション処理を有効にします。
 - 競合しないオペレーションは、デプロイ時間を短縮するために同時に実行されます。
 - 競合するオペレーションは自動的にキューに入れられ、リクエストされた順序で処理されます。

オペレーションが実行中またはキューに入れられている間、CloudFormation ではオペレーションが競合しない場合でもすべての受信オペレーションがキューに入ります。この間は実行設定を変更できません。

11. テンプレートに IAM リソースが含まれる場合は、[機能] で[I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
12. [次へ] を選択して続行し、信頼されたアクセスがまだアクティブ化になっていない場合はアクティブ化します。
13. [デプロイオプションの設定] ページの[デプロイターゲット] で、次のいずれかを実行します。
 - 組織内のすべてのアカウントにデプロイするには、[Deploy to organization (組織にデプロイ)] を選択します。
 - 特定の OU のすべてのアカウントにデプロイするには、[Deploy to organizational units (OUs) (組織単位 (OU) にデプロイ)] を選択します。[Add an OU (OU の追加)] を選択し、テキストボックスにターゲット OU ID を貼り付けます。新しいターゲット OU ごとに繰り返します。

[組織単位 (OU) へのデプロイ] を選択した場合、[アカウントフィルタータイプ] で、次のいずれかのオプションを選択し、アカウント番号を指定することで、デプロイターゲットを特定の個々のアカウントに設定できます。

- [なし] (デフォルト) – 指定した OU 内のすべてのアカウントにスタックをデプロイします。
 - [共通集合] – 選択した OU 内の特定の個々のアカウントにスタックをデプロイします。
 - [差異] – 特定のアカウントを除く、選択した OU 内のすべてのアカウントにスタックをデプロイします。
 - [和集合] – 指定した OU と追加の個々のアカウントにスタックをデプロイします。
14. [自動デプロイ] で、今後ターゲットの組織または OU に追加されるアカウントに自動デプロイを行うかどうかを選択します。詳細については、「[AWS Organizations で StackSets の自動デプロイを有効または無効にする](#)」を参照してください。
 15. 自動デプロイを有効にした場合、[Account removal behavior (アカウントの削除の動作)] で、ターゲットの組織または OU からアカウントを削除するときに、スタックリソースを保持するか削除するかを選択します。

 Note

[スタックを保持] を選択すると、スタックは StackSet から削除されますが、スタックとそれに関連付けられたリソースは保持されます。リソースは現在の状態のままですが、StackSet には含まれなくなります。

16. [リージョンの指定] で、スタックをデプロイするリージョンを選択します。
17. [デプロイオプション] で、次の操作を行います。
 - [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。
 - [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。
 - [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。
 - [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
 - [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
 - [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。
 - StackSet の依存関係では、依存する StackSet ARN を追加して、最大 10 個以内で依存関係を維持します。詳細については、「[AWS Organizations で StackSets の自動デプロイを有効または無効にする](#)」を参照してください。
18. [次へ] を選択して続行します。
19. [レビュー] ページで、StackSet が正しいリージョンに正しいアカウントをデプロイすることを確認し、[StackSet の作成] を選択します。

[StackSet の詳細] ページが開きます。StackSet 内のスタックの作成の進行状況とステータスを表示できます。

サービスマネージド型のアクセス許可を持つ StackSet 作成 (AWS CLI)

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- StackSet コンテナを作成します。

- スタックインスタンスをデプロイします。

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

Deploy to your organization

StackSet を作成するには

1. [create-stack-set](#) コマンドを使用して、*my-stackset* という名前の新しい StackSet を作成します。次の例では、S3 バケットに保存されているテンプレートを使用し、自動デプロイを有効にして、アカウントが削除されたときにスタックを保持します。詳細については、「[AWS Organizations で StackSets の自動デプロイを有効または無効にする](#)」を参照してください。

```
aws cloudformation create-stack-set \  
  --stack-set-name my-stackset \  
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/MyApp.template \  
  --permission-model SERVICE_MANAGED \  
  --auto-deployment  
Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
```

2. [list-stack-sets](#) コマンドを使用して、StackSet が作成されたことを確認します。新しい StackSet が結果に表示されます。

```
aws cloudformation list-stack-sets
```

- メンバーアカウントにサインインしているときに `--call-as` オプションを `DELEGATED_ADMIN` に設定した場合、`list-stack-sets` は組織の管理アカウント内のサービスマネージド型のアクセス許可を持つすべての StackSets を返します。
- AWS アカウント にサインインしているときに `--call-as` オプションを `SELF` に設定した場合、`list-stack-sets` は AWS アカウント 内のすべてのセルフマネージド型の StackSets を返します。

- 組織の管理アカウントにサインインしているときに `--call-as` オプションを `SELF` に設定した場合、`list-stack-sets` は組織の管理アカウント内のすべての `StackSets` を返します。

3. [create-stack-instances](#) コマンドを使用して、`StackSet` にスタックを追加します。 `--deployment-targets` オプションには、組織内のすべてのアカウントにデプロイする組織のルート ID を指定します。

`--operation-preferences` オプションを使用して、同時アカウント処理およびその他のデプロイ設定を設定します。この例では、カウントベースの設定を使用します。`MaxConcurrentCount` は `FailureToleranceCount + 1` を超えることはできません。パーセンテージベースの設定の場合は、代わりに `FailureTolerancePercentage` または `MaxConcurrentPercentage` を使用します。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=r-a1b2c3d4e5 \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

詳細については、「AWS CloudFormation API リファレンス」の「[CreateStackInstances](#)」を参照してください。

4. `create-stack-instances` 出力の一部として返された `operation-id` を使用して、次の [describe-stack-set-operation](#) コマンドを使用し、スタックが正常に作成されたことを確認します。

```
aws cloudformation describe-stack-set-operation \
  --stack-set-name my-stackset \
  --operation-id operation_ID
```

Deploy to organizational units (OUs)

StackSet を作成するには

1. [create-stack-set](#) コマンドを使用して、`my-stackset` という名前の新しい `StackSet` を作成します。次の例では、S3 バケットに保存されているテンプレートを使用し、`KeyPairName` に値 `TestKey` を設定するパラメータが含まれています

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
```



```
--template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/MyApp.template \  
--permission-model SERVICE_MANAGED \  
--parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. [list-stack-sets](#) コマンドを使用して、StackSet が作成されたことを確認します。新しい StackSet が結果に表示されます。

```
aws cloudformation list-stack-sets
```

- メンバーアカウントにサインインしているときに `--call-as` オプションを `DELEGATED_ADMIN` に設定した場合、`list-stack-sets` は組織の管理アカウント内のサービスマネージド型のアクセス許可を持つすべての StackSets を返します。
 - AWS アカウント にサインインしているときに `--call-as` オプションを `SELF` に設定した場合、`list-stack-sets` は AWS アカウント 内のすべてのセルフマネージド型の StackSets を返します。
 - 組織の管理アカウントにサインインしているときに `--call-as` オプションを `SELF` に設定した場合、`list-stack-sets` は組織の管理アカウント内のすべての StackSets を返します。
3. [create-stack-instances](#) コマンドを使用して、StackSet にスタックを追加します。 `--deployment-targets` オプションで、デプロイ先の OU ID を指定します。

`--operation-preferences` オプションを使用して、同時アカウント処理およびその他のデプロイ設定を設定します。この例では、カウントベースの設定を使用します。MaxConcurrentCount は FailureToleranceCount + 1 を超えることはできません。パーセンテージベースの設定の場合は、代わりに FailureTolerancePercentage または MaxConcurrentPercentage を使用します。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \  
--deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \  
--regions us-west-2 us-east-1 \  
--operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

詳細については、「AWS CloudFormation API リファレンス」の「[CreateStackInstances](#)」を参照してください。

4. `create-stack-instances` 出力の一部として返された `operation-id` を使用して、次の [describe-stack-set-operation](#) コマンドを使用し、スタックが正常に作成されたことを確認します。

```
aws cloudformation describe-stack-set-operation \  
  --stack-set-name my-stackset \  
  --operation-id operation_ID
```

Deploy to specific accounts in OUs

特定の組織単位 (OU) をターゲットにし、アカウントフィルタリングを使用して、スタックデプロイを受信するアカウントを正確に制御できます。デフォルトでは、アカウントフィルタリングが指定されていない場合、スタックは指定された OU 内のすべてのアカウントにデプロイされます。

AWS CLI で、`--deployment-targets` オプションを使用してアカウントフィルタリングを指定します。詳細については、「[DeploymentTargets](#)」を参照してください。

`create-stack-set` コマンドを使用して StackSet コンテナを作成したら、次のいずれかの例を使用してスタックを特定のアカウントにデプロイします。

OU 内の特定のアカウントをターゲットにする

次の例では、OU1 内のアカウント A1 と A2 にのみスタックをデプロイします。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \  
  --deployment-targets  
  OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=INTERSECTION \  
  --regions us-west-2 us-east-1
```

OU からアカウントを除外する

次の例では、アカウント A1 と A2 を除く OU1 内のすべてのアカウントにスタックをデプロイします。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \  
  --deployment-targets  
  OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=DIFFERENCE \  
  --regions us-west-2 us-east-1
```

AWS Organizations で StackSets の自動デプロイを有効または無効にする

CloudFormation は、新しい AWS Organizations アカウントがターゲットの組織または組織単位 (OU) に追加されると、そのアカウントに追加のスタックを自動的にデプロイします。自動デプロイを有効にして、アカウントがターゲット OU から削除されたときにスタックおよび関連するリソースを削除するか保持するかを選択することができます。これらの設定は、サービスマネージド型のアクセス許可を使用する StackSets ではいつでも変更できます。

自動デプロイの仕組み

自動デプロイを有効にすると、アカウントがターゲット組織または OU に追加されたとき、ターゲット組織または OU から削除されたとき、ターゲット OU 間で移動したときに、トリガーされます。

例えば、us-east-1 リージョンの OU1 をターゲットにする StackSet1 と、us-east-1 リージョンの OU2 をターゲットにする StackSet2 がある場合、OU1 には AccountA が含まれます。

自動デプロイを有効にして AccountA を OU1 から OU2 に移動すると、CloudFormation は自動的に削除オペレーションを実行して StackSet1 スタックを AccountA から削除し、StackSet2 スタックを AccountA に追加する作成オペレーションをキューに入れます。

考慮事項

自動デプロイを使用する際の考慮事項を次に示します。

- 自動デプロイ機能は StackSet レベルで有効になります。OU、アカウント、リージョンを選択して自動デプロイを調整することはできません。
- 上書きされたパラメータ値は、ターゲット OU とその子 OU に現在存在するアカウントにのみ適用されます。今後ターゲット OU とその子 OU に追加されるアカウントでは、StackSet のデフォルト値が使用され、オーバーライドされた値は使用されません。
- 自動デプロイでは、アカウントレベルのターゲットリングフィルターは考慮しません。特定のアカウントをターゲットにして自動デプロイを有効にすると、StackSet はデプロイを継続して、デプロイされた組織内の新しく追加されたアカウントにデプロイします。新しく追加されたアカウントへのデプロイを防ぐには、自動デプロイを無効にします。
- 依存関係管理: StackSet ごとに最大 10 個の依存関係を定義し、アカウントごとに最大 100 個の依存関係にします。例えば、それぞれ 5 つの依存関係を持つ 5 つの StackSets がある場合、100 個

の依存関係制限に向けてカウントしている 25 の依存関係があります。[Service Quotas コンソール](#)を使用して制限の引き上げをリクエストできます。依存関係は、StackSets が削除されるか、組織が非アクティブ化されると削除されます。

- 自動デプロイを使用する場合は、マネージド実行を有効にすることをお勧めします。マネージド実行では、複数のターゲットアカウントでの自動デプロイオペレーションを StackSet 内で同時に実行できるため、特に大規模な組織では処理速度が向上します。

自動デプロイを有効または無効にする (コンソール)

自動デプロイを有効または無効にするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。
4. [StackSets] ページで、更新する StackSet の名前の横にあるオプションを選択します。
5. 右上の [アクション] メニューで [自動デプロイを編集] を選択します。
6. 開いたダイアログボックスで、次の操作を行います。
 - a. [自動デプロイ] で、[アクティブ化済み] または [非アクティブ化済み] を選択します。
 - b. [アカウント削除の動作] で、[スタックを削除] または [スタックを保持] を選択します。保持されたリソースは現在の状態のままですが、StackSet には含まれなくなります。
 - c. StackSet の依存関係では、依存する StackSet ARN を追加して、最大 10 個以内で依存関係を維持します。
7. [保存] を選択します。

自動デプロイを有効または無効にする (AWS CLI)

自動デプロイを有効または無効にするには

1. `--auto-deployment` オプションで [update-stack-set](#) コマンドを使用します。

次のコマンドを実行すると、自動デプロイが有効になります。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \
```

```
--use-previous-template --auto-deployment  
Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
```

自動デプロイを無効にしたい場合は、次の例のように、Enabled=false を --auto-deployment オプションの値として指定します。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \  
--use-previous-template --auto-deployment Enabled=false
```

2. update-stack-set 出力の一部として返されたオペレーション ID を使用して、[describe-stack-set-operation](#) を実行し、StackSet が正常に更新されたことを確認します。

```
aws cloudformation describe-stack-set-operation --operation-id operation_ID
```

CloudFormation StackSets を更新する

CloudFormation コンソールまたは AWS CLI を使用して StackSet を更新できます。

StackSet からアカウントとリージョンを追加および削除するには、「[StackSets にスタックを追加する](#)」および「[StackSets からスタックを削除する](#)」を参照してください。スタックのパラメータ値を上書きするには、「[スタックのパラメータを上書きする](#)」を参照してください。

トピック

- [StackSet を更新する \(コンソール\)](#)
- [StackSet を更新する \(AWS CLI\)](#)

StackSet を更新する (コンソール)

StackSet を更新するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。
4. [StackSets] ページで、更新する StackSet を選択します。

5. StackSet を選択した状態で、[アクション] メニューから [StackSet の詳細を編集] を選択します。
6. [テンプレートの選択] ページで、必要に応じて[アクセス許可] セクションを更新するか、次のステップに進みます。
7. 前提条件 - テンプレートを準備するには、[現在のテンプレートを使用する] を選択して現在のテンプレートを使用するか、[現在のテンプレートを置き換える] を選択して別のテンプレートに S3 URL を指定するか、新しいテンプレートをアップロードします。
8. [次へ] を選択します。
9. [StackSet の詳細を指定] ページの [StackSet の説明] で、必要に応じて StackSet の説明を更新します。
10. パラメータ の場合、必要に応じてパラメータ値を更新します。
11. [次へ] を選択します。
12. [StackSet オプションの設定] ページの [タグ] で、タグを必要に応じて変更します。タグを追加、更新、または削除できます。AWS でのタグの使用方法に関する詳細については、AWS Billing and Cost Management ユーザーガイドの「[AWS コスト配分タグを使用してコストを整理および追跡する](#)」を参照してください。
13. [実行設定] では、必要に応じて実行設定を更新できます。

 Note

オペレーションが実行中またはキューに入っている場合、実行設定を変更することはできません。

14. テンプレートに IAM リソースが含まれる場合は、[機能] で [I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。
15. [Next] (次へ) を選択します。
16. [デプロイオプションの設定] ページで、更新用のアカウントとリージョンを指定します。

CloudFormation は、リージョンのデプロイ失敗が指定された耐障害性を超えない限り、最初のリージョン内の指定されたアカウントにスタック更新をデプロイし、次のリージョンに移行し、それが繰り返されます。

- a. [セルフマネージド型のアクセス許可] [アカウント] の [Deployment locations (デプロイ先)] で [Deploy stacks in accounts (スタックをアカウントにデプロイ)] を選択します。StackSet の

作成に使用したターゲットアカウント ID をテキストボックスに貼り付け、複数の数字をカンマで区切ります。

[サービスマネージド型のアクセス許可] 次のいずれかを実行します。

- [Deploy to organizational units (OUs) (組織単位 (OU) にデプロイ)] を選択します。StackSet の作成に使用したターゲット OU を入力します。
 - [Deploy to accounts (アカウントにデプロイ)] を選択します。StackSet の作成に使用したターゲット OU ID またはアカウント ID を貼り付けます。
- b. [Specify regions (リージョンの指定)] では、CloudFormation で更新をデプロイする順序を指定します。
- c. [デプロイオプション] で、次の操作を行います。
- [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。
 - [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。
 - [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。
 - [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
 - [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
 - [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。
 - [サービスマネージドアクセス許可] StackSet の依存関係では、依存する StackSet ARN を追加して、最大 10 個以内で依存関係を維持します。詳細については、「[AWS Organizations で StackSets の自動デプロイを有効または無効にする](#)」を参照してください。
- d. [次へ] を選択して続行します。
17. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。
18. 続行する準備ができたなら、[送信] を選択します。

CloudFormation によって StackSet への更新の適用が開始され、StackSet 詳細ページの [オペレーション] タブが表示されます。[オペレーション] タブで更新オペレーションの進捗と状況を確認できます。

StackSet を更新する (AWS CLI)

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

1. StackSet を更新するには

[update-stack-set](#) コマンドを実行して、StackSet を変更します。

次の例では、`--parameters` オプションを使用して StackSet を更新します。特に、送信チャネル設定のデフォルトスナップショット送信頻度を、`[TwentyFour_Hours]` から `[Twelve_Hours]` に変更します。現在のテンプレートをまだ使用しているため、`--use-previous-template` オプションを追加します。

`--operation-preferences` オプションを使用して、同時アカウント処理およびその他のデプロイ設定を設定します。これらの例では、カウントベースの設定を使用します。MaxConcurrentCount は FailureToleranceCount + 1 を超えることはできません。パーセンテージベースの設定の場合は、代わりに FailureTolerancePercentage または MaxConcurrentPercentage を使用します。

[セルフマネージド型のアクセス許可] `--accounts` オプションでは、更新のターゲットにするアカウント ID を指定します。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \  
  --use-previous-template \  
  --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \  
  --accounts account_ID_1 account_ID_2 \  
  --regions us-west-2 us-east-1 \  
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```


[サービスマネージド型のアクセス許可] `--deployment-targets` オプションでは、更新のターゲットにする組織ルート ID または 組織単位 (OU) ID を指定します。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \  
  --use-previous-template \  
  --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \  
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \  
  --regions us-west-2 us-east-1 \  
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

詳細については、AWS CloudFormation API リファレンスの「[UpdateStackSet](#)」を参照してください。

2. `describe-stack-set-operation` コマンドを実行して更新オペレーションのステータスおよび結果を表示し、StackSet が正常に更新されたことを確認します。`--operation-id` には、`update-stack-set` コマンドに返されたオペレーション ID を使用します。

```
aws cloudformation describe-stack-set-operation \  
  --operation-id operation_ID
```

CloudFormation StackSets にスタックを追加する

StackSet の作成時に、StackSet のスタックを作成できます。CloudFormation では、StackSet の作成後はいつでも、追加のアカウントやリージョン用に、さらにスタックを追加することもできます。CloudFormation コンソールまたは AWS CLI を使用してスタックを追加できます。

トピック

- [StackSet にスタックを追加する \(コンソール\)](#)
- [スタックを StackSet に追加する \(AWS CLI\)](#)

StackSet にスタックを追加する (コンソール)

スタックを StackSet に追加するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。

3. ナビゲーションペインから [StackSets] を選択します。[StackSets] ページで、作成した StackSet を選択します。
4. StackSet を選択した状態で、[アクション] メニューから [StackSet にスタックを追加] を選択します。
5. [デプロイオプションの設定] ページで、以下を実行します。
 - a. [StackSet にスタックを追加] で、[新しいスタックのデプロイ] を選択します。
 - b. 次に、StackSet のアクセス許可設定に応じて、以下を実行します。
 - [セルフマネージド型のアクセス許可] [アカウント] の [Deployment locations (デプロイ先)] で [Deploy stacks in accounts (スタックをアカウントにデプロイ)] を選択します。ターゲットアカウント番号をテキストボックスに貼り付け、複数の数字をカンマで区切ります。
 - [サービスマネージド型のアクセス許可] [デプロイターゲット] で次のいずれかを実行します。
 - 組織内のすべてのアカウントにデプロイするには、[Deploy to organization (組織にデプロイ)] を選択します。
 - 特定の OU のすべてのアカウントにデプロイするには、[Deploy to organizational units (OUs) (組織単位 (OU) にデプロイ)] を選択します。[Add another OU (別の OU の追加)] を選択し、テキストボックスにターゲット OU ID を貼り付けます。新しいターゲット OU ごとに繰り返します。CloudFormation は選択したターゲットの子 OU もターゲットにします。
 - c. [リージョンを指定] では、前のステップで指定したターゲットアカウントにデプロイする AWS リージョン を指定します。デフォルトでは、CloudFormation は、リージョンのデプロイ失敗が指定された耐障害性を超えない限り、最初のリージョン内の指定されたアカウントにスタックをデプロイし、次のリージョンに移行し、それが繰り返されます。
 - d. [デプロイオプション] で、次の操作を行います。
 - [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。

 Note

StackSet が既にターゲットにしている OU を追加すると、CloudFormation は、StackSet でスタックを持たない OU 内のアカウント (例えば、StackSet が作成され、自動デプロイが無効になった後に OU に追加されたアカウントなど) に新しいスタックを作成します。

- [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。
 - [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。
 - [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
 - [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
 - [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。
- e. [次へ] を選択します。
6. [Set Overrides (上書きを設定)] ページでは、プロパティ値は指定されたとおりにしておきます。作成しているスタックではプロパティ値を上書きすることはありません。[次へ] を選択します。
 7. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。
 8. 続行する準備ができたなら、[送信] を選択します。

CloudFormation でスタックの作成が開始されます。StackSet 内のスタック作成の進捗と状況は、[送信] を選択した際に関開 StackSet 詳細ページで確認できます。完了すると、新しいスタックが [スタックインスタンス] タブに一覧表示されます。

スタックを StackSet に追加する (AWS CLI)

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

セルフマネージド型のアクセス許可を持つ StackSet にスタックを追加するには

`create-stack-instances` CLI コマンドを使用します。--accounts オプションでは、スタックを作成するアカウント ID を指定します。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
```

```
--accounts account_id --regions eu-west-1 us-west-2
```

サービスマネージド型のアクセス許可を持つ StackSet にスタックを追加するには

create-stack-instances CLI コマンドを使用します。--deployment-targets オプションでは、スタックを作成する組織 (ルート) ID または OU ID を指定します。特定のアカウントをターゲットとするコマンドの例については、「[サービスマネージド型のアクセス許可を持つ StackSet 作成 \(AWS CLI\)](#)」を参照してください。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \  
--deployment-targets OrganizationalUnitIds=ou-rcuk-r1qi0wl7 --regions eu-west-1 us-west-2
```

Note

StackSet が既にターゲットにしている OU を追加すると、CloudFormation は、StackSet でスタックを持たない OU 内のアカウント (例えば、StackSet が作成され、自動デプロイが無効になった後に OU に追加されたアカウントなど) に新しいスタックを作成します。

CloudFormation StackSet 内のスタックのパラメータ値を上書きする

場合によっては、特定のリージョンまたはアカウントのスタックのプロパティ値を、StackSet そのものに指定されたものとは異なる値にすることがあります。たとえば、アカウントが開発に使用されているか本稼働に使用されているかに基づいて、指定されたパラメータに異なる値を指定する場合を考えます。このような状況の場合、CloudFormation では、アカウントおよびリージョン別にスタックのパラメータ値を上書きできます。スタックを初めて作成する際に、テンプレートパラメータの値を上書きできます。また、既存のスタックのパラメータ値を上書きすることもできます。スタックで以前に上書きしたパラメータは、StackSet に指定されている値にのみ戻すことができます。

パラメータ値の上書きは、選択したアカウントとリージョンのスタックに適用されます。StackSet の更新中、スタックで上書きされたすべてのパラメータ値は更新されず、上書きされた値を保持します。

StackSet で指定されたパラメータ値のみを上書きできます。パラメータそのものを追加または削除するには、StackSet のテンプレートを更新する必要があります。StackSet テンプレートにパラメータを追加する場合、スタックのパラメータ値を上書きするには、まずすべてのスタックを StackSet

に指定された新しいパラメータと値で更新する必要があります。すべてのスタックが新しいパラメータに更新されたら、個々のスタックのパラメータ値を必要に応じて上書きできます。

スタックの作成時に StackSet のパラメータ値を上書きする方法については、「[StackSets にスタックを追加する](#)」を参照してください。

トピック

- [スタックのパラメータを上書きする \(コンソール\)](#)
- [スタックのパラメータを上書きする \(AWS CLI\)](#)

スタックのパラメータを上書きする (コンソール)

特定のスタックのパラメータを上書きするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。[StackSets] ページで、StackSet を選択します。
4. StackSet を選択した状態で、[アクション] メニューから [StackSet パラメータを上書き] を選択します。
5. [デプロイオプションの設定] ページで、パラメータを上書きするスタックのアカウントとリージョンを指定します。

デフォルトでは、CloudFormation は、リージョンのデプロイ失敗が指定された耐障害性を超えないという条件で、最初のリージョン内の指定されたアカウントにスタックをデプロイし、次のリージョンに移行し、それが繰り返されます。

- a. [セルフマネージド型のアクセス許可] [Deployment locations (デプロイ先)] で [Deploy stacks in accounts (スタックをアカウントにデプロイ)] を選択します。StackSet の作成に使用した一部またはすべてのターゲットアカウント ID を貼り付けます。

[サービスマネージド型のアクセス許可] 次のいずれかを実行します。

- [Deploy to organizational units (OUs) (組織単位 (OU) にデプロイ)] を選択します。StackSet の作成に使用したターゲット OU を 1 つ以上入力します。上書きされたパラメータ値は、ターゲット OU とその子 OU に現在存在するアカウントにのみ適用さ

れます。今後ターゲット OU とその子 OU に追加されるアカウントでは、StackSet のデフォルト値が使用され、オーバーライドされた値は使用されません。

- [Deploy to accounts (アカウントにデプロイ)] を選択します。StackSet の作成に使用した一部またはすべてのターゲット OU ID またはアカウント ID を貼り付けます。

- b. [リージョンの指定] で、この StackSet のスタックをデプロイしたリージョンを 1 つまたは複数追加します。

複数のリージョンを追加すると、[Specify regions (リージョンの指定)] のリージョンの順序に従って、デプロイの順序が決まります。

- c. [デプロイオプション] で、次の操作を行います。

- [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。
- [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。
- [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。
- [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
 - [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
 - [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。

- d. [次へ] を選択します。

6. [Specify Overrides (上書きの指定)] ページで、上書きするパラメータのチェックボックスをオンにし、[Edit override value (上書き値の編集)] メニューから [Override StackSet value (StackSet 値の上書き)] を選択します。
7. [Override StackSet parameter values (StackSet パラメータ値の上書き)] ページで、変更を加え、[変更の保存] を選択します。

Note

上書きされたパラメータを、StackSet で指定された値を再度使用するように設定するには、すべてのパラメータを確認し、[上書き値の編集] メニューから [StackSet 値に設定] を選択します。これにより、スタックを更新すると、すべての上書きされた値が削除されます。

8. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。
9. 続行する準備ができたなら、[送信] を選択します。

CloudFormation がスタックの更新を開始します。スタックの進捗状況とステータスは、[送信] を選択した際に開く StackSet 詳細ページで確認できます。

スタックのパラメータを上書きする (AWS CLI)

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

特定のスタックのパラメータを上書きするには

1. [update-stack-instances](#) AWS CLI コマンドを使用し、`--parameter-overrides` オプションを指定します。

[セルフマネージド型のアクセス許可] `--accounts` オプションでは、スタックのパラメータ値を上書きするアカウント ID を指定します。

```
aws cloudformation update-stack-instances --stack-set-name my-stackset \  
  --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351 \  
  \, subnet-27b86940 \  
  --accounts account_id --regions us-east-1
```

[サービスマネージド型のアクセス許可] `--deployment-targets` オプションでは、スタックのパラメータ値を上書きする組織のルート ID、OU ID、または AWS Organizations アカウント ID を指定します。この例では、OU 内のすべてのアカウントのスタックのパラメータ値を *ou-rcuk-1x5j1lwo* ID で上書きします。

上書きされたパラメータ値は、ターゲット OU とその子 OU に現在存在するアカウントにのみ適用されます。今後ターゲット OU とその子 OU に追加されるアカウントでは、StackSet のデフォルト値が使用され、上書きされた値は使用されません。

```
aws cloudformation update-stack-instances --stack-set-name my-stackset \  
  --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351 \  
  \, subnet-27b86940 \  
  --deployment-targets ou-rcuk-1x5j1lwo --regions us-east-1
```



```
--parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351\  
\,subnet-27b86940 \  
--deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo \  
--regions us-east-1
```

2. describe-stack-set-operation コマンドを実行して更新オペレーションのステータスおよび結果を表示し、スタックのパラメータ値が正常に上書きされたことを確認します。--operation-id には、update-stack-instances コマンドに返されたオペレーション ID を使用します。

```
aws cloudformation describe-stack-set-operation --operation-id operation_ID
```

CloudFormation StackSets からスタックを削除する

CloudFormation コンソールまたは AWS CLI を使用して、StackSets からスタックを削除できます。

トピック

- [StackSet からスタックを削除する \(コンソール\)](#)
- [StackSet からスタックを削除する \(AWS CLI\)](#)

Note

最上位の組織単位 (OU) からスタックを削除すると、その OU は StackSet のターゲットとして削除されます。

StackSet からスタックを削除する (コンソール)

スタックを削除するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [StackSets] を選択します。[StackSets] ページで、StackSet を選択します。
4. StackSet を選択した状態で、[アクション] メニューから [StackSet からスタックを削除] を選択します。

5. [デプロイオプションの設定] ページで、まずスタックを削除するアカウントとリージョンを選択します。

- a. [セルフマネージド型のアクセス許可] [アカウント] で、[スタックをアカウントにデプロイ] または [スタックを組織単位にデプロイ] を選択します。

[スタックをアカウントにデプロイ] を選択した場合、ターゲットアカウント番号を (複数の数字はカンマで区切って) [アカウント番号] テキストボックスに貼り付けます。

[スタックを組織単位にデプロイ] を選択した場合、[組織番号] テキストボックスにターゲット OU ID を貼り付けると、指定された組織の一部であるすべてのアカウントをターゲットにできます。

- b. [サービスマネージド型のアクセス許可] [組織単位 (OU)] では、ターゲット OU ID を指定します。

 Important

CloudFormation は、指定されたターゲット OU とその子 OU の両方からスタックを削除します。

[アカウントフィルタータイプ] では、次のいずれかのオプションを選択し、アカウント番号を指定することで、スタックを削除するアカウントを絞り込むことができます。

- [なし] (デフォルト) – 指定した OU 内のすべてのアカウントからスタックを削除します。
- [共通集合] – 選択した OU 内の特定の個々のアカウントからのみスタックを削除します。
- [差異] – 特定のアカウントを除く、選択した OU 内のすべてのアカウントからスタックを削除します。
- [和集合] – 指定した OU と追加の個々のアカウントからスタックを削除します。

- c. [リージョンの指定] で、ターゲットアカウント内のスタックを削除するリージョンを選択します。

6. [デプロイオプション] で、次の操作を行います。

- [同時アカウントの最大数] で、同時に処理されるアカウントの数を指定します。
- [耐障害性] には、リージョンごとに許可されるアカウント障害の最大数を指定します。この制限に達すると、オペレーションは停止し、他のリージョンには進みません。

- [スタックを保持] オプションを有効にすると、StackSet からスタックを削除するときに、スタックとそれに関連付けられたリソースが保存されます。リソースは現在の状態のままですが、StackSet には含まれなくなります。
 - [リージョンの同時実行] で、リージョンの処理方法を選択します。[順次] (一度に 1 つのリージョン) または [並列] (複数のリージョンを同時に) を選択できます。
 - [同時実行モード] で、オペレーションの実行中に同時実行がどのように動作するかを選択します。
 - [厳格な耐障害性] – 障害発生時の同時実行レベルを下げ、[耐障害性] の値を +1 の範囲内にとどめます。
 - [ソフトな障害耐性] – 障害に関係なく、指定された同時実行レベル ([同時アカウントの最大数] の値) を維持します。
7. [次へ] を選択します。
 8. [確認] ページで選択内容を確認します。変更するには、関連セクションで [編集] をクリックします。
 9. StackSet からスタックを削除する準備ができたなら、[送信] を選択します。

スタックの削除が完了したら、StackSet の詳細ページの [スタックインスタンス] タブで、StackSet からスタックが削除されたことを確認できます。

StackSet からスタックを削除する (AWS CLI)

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

StackSet 名を指定した `delete-stack-instances` コマンドを使用します。

これらの例では、スタックを保持していないため、`--no-retain-stacks` オプションを使用します。スタックとそのリソースを保持する場合は、`--no-retain-stacks` ではなく `--retain-stacks` を使用します。

`--regions` で、スタックを削除する AWS リージョン (`us-west-2`、`us-east-1` など) を指定します。

--operation-preferences オプションを使用して、同時アカウント処理およびその他の設定を設定します。これらの例では、カウントベースの設定を使用します。MaxConcurrentCount は FailureToleranceCount + 1 を超えることはできません。パーセンテージベースの設定の場合は、代わりに FailureTolerancePercentage または MaxConcurrentPercentage を使用します。

スタックを削除するには (セルフマネージド型のアクセス許可)

--accounts オプションでは、スタックを削除するアカウントの ID を指定します。

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \  
  --accounts account_ID_1 account_ID_2 \  
  --regions us-west-2 us-east-1 \  
  --no-retain-stacks \  
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

スタックを削除するには (サービスマネージド型のアクセス許可)

--deployment-targets では、スタックを削除する組織のルート ID または組織単位 (OU) ID を指定します。

Important

CloudFormation は、指定されたターゲット OU とその子 OU の両方からスタックを削除します。

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \  
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5jlwo,ou-rcuk-slr5lh0a \  
  --regions us-west-2 us-east-1 \  
  --no-retain-stacks \  
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

詳細については、「AWS CloudFormation API リファレンス」の「[DeleteStackInstances](#)」を参照してください。

必要に応じて、スタックの削除が完了した後に、describe-stack-set-operation コマンドを実行してスタック削除オペレーションのステータスおよび結果を表示し、StackSet からスタックが削除されたことを確認します。--operation-id には、delete-stack-instances コマンドに返されたオペレーション ID を使用します。

```
aws cloudformation describe-stack-set-operation --stack-set-name my-stackset \  
--operation-id ddf16f54-ad62-4d9b-b0ab-3ed8e9example
```

CloudFormation StackSets を削除する

StackSet を削除するには、まず StackSet のスタックをすべて削除する必要があります。すべてのスタックを削除する方法の詳細については、「[StackSets からスタックを削除する](#)」を参照してください。

トピック

- [StackSet を削除する \(コンソール\)](#)
- [StackSet を削除する \(AWS CLI\)](#)
- [Delete Service Roles \(オプション\)](#)

StackSet を削除する (コンソール)

StackSet を削除するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、StackSet を作成した AWS リージョン を選択します。
3. [StackSets] ページで、StackSet を選択します。
4. StackSet を選択した状態で、[アクション] メニューから [StackSet の削除] を選択します。
5. StackSet の削除を確認するプロンプトが表示されたら、[StackSet の削除] を選択します。

StackSet を削除する (AWS CLI)

Note

委任管理者として操作する場合は、コマンドに `--call-as DELEGATED_ADMIN` を含める必要があります。

StackSet を削除するには

1. 次の delete-stack-set コマンドを使用します。確認を求められたら、**y** と入力してから [Enter] キーを押して続行します。

```
aws cloudformation delete-stack-set --stack-set-name my-stackset
```

2. list-stack-sets コマンドを実行して、StackSet が削除されたことを確認します。list-stack-sets コマンドの結果は、DELETED のステータスを持つスタックを表示します。

```
aws cloudformation list-stack-sets
```

Delete Service Roles (オプション)

CloudFormation が StackSet オペレーションを実行するために必要な IAM サービスロールが不要になった場合は、ロールを削除することをお勧めします。

セルフマネージド型の StackSets の場合、作成したロールになります。これらのユーザーロールの詳細については、「[セルフマネージド型のアクセス許可を付与する](#)」を参照してください。

サービスマネージド型の StackSets の場合、StackSets 用に自動的に作成されたロールには、組織管理アカウントに CloudFormationStackSetsOrgAdmin、各ターゲットアカウントに CloudFormationStackSetsOrgMember のサフィックスが付いています。詳細については、「[サービスリンクロール](#)」を参照してください。

サービスロールを削除するには (コンソール)

1. AWS マネジメントコンソール にサインインして、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から、[ロール] を選択し、削除するロールの隣にあるチェックボックスをオンにします。
3. ページ上部の [Role actions (ロールのアクション)] メニューで、[Delete role (ロールの削除)] を選択します。
4. 確認ダイアログボックスで、[Yes, Delete] を選択します。確実に削除する場合は、サービスの最終アクセス時間データをまだロード中であっても、削除を実行できます。

サービスロールを削除するには (AWS CLI)

- 次の delete-role コマンドを使用します。確認を求められたら、y と入力してから [Enter] を押します。

```
aws iam delete-role --role-name role name
```

ロールの削除に関する詳細については、「IAM ユーザーガイド」の「[Deleting roles and instance profiles \(ロールおよびインスタンスプロファイルの削除\)](#)」を参照してください。

ターゲットアカウントゲートを使用して失敗した StackSets デプロイを防止する

アカウントゲートは、CloudFormation がターゲットアカウントで StackSets オペレーションを開始する前に、そのアカウントが特定の要件を満たしていることを検証できるオプション機能です。この検証は、前提条件チェックとして機能する AWS Lambda 関数を通じて実行されます。

アカウントゲートの一般的な例としては、ターゲットアカウントにアクティブまたは未解決の CloudWatch アラームがないことを検証することが挙げられます。CloudFormation は、ターゲットアカウントのスタックオペレーションを開始する度に Lambda 関数を呼び出し、SUCCEEDED コードが返る場合のみ次に進みます。Lambda 関数より FAILED ステータスが返される場合、CloudFormation は要求されたオペレーションを続行しません。Lambda 関数でアカウントゲートを設定していない場合、CloudFormation はこの確認をスキップし、オペレーションを継続します。

ターゲットアカウントでアカウントのゲートチェックを行うことができない場合、失敗したオペレーションは、スタックの指定した耐障害性の数または割合 (%) に含められます。耐障害性の詳細については、「[StackSet オペレーションのオプション](#)」を参照してください。

アカウントゲート機能は、StackSets オペレーションでのみ使用できます。この機能は、StackSets 以外の他の CloudFormation オペレーションでは使用できません。

要件

アカウントゲートでは、次の要件を満たす必要があります。

- この機能を使用するには、Lambda 関数に AWSCloudFormationStackSetAccountGate という名前を付ける必要があります。

- `AWSCloudFormationStackSetExecutionRole` では、Lambda 関数を呼び出すためのアクセス許可が必要です。これらのアクセス許可がない場合、CloudFormation はアカウントゲート機能の確認をスキップし、スタックオペレーションを続行します。
- アカウントのゲート作成を行うには、Lambda `InvokeFunction` のアクセス許可をターゲットアカウントに追加する必要があります。ターゲットアカウントの信頼ポリシーには、管理者アカウントとの信頼関係が必要です。Lambda のアクセス許可 (`InvokeFunction`) を付与するポリシーステートメントの例を次に示します。

JSON

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

Lambda 関数を作成するための CloudFormation テンプレート

次のサンプルテンプレートを使用して Lambda `AWSCloudFormationStackSetAccountGate` 関数を作成します。これらのいずれかのテンプレートを使用して新しいスタックを作成するには、「[CloudFormation コンソールからスタックを作成する](#)」を参照してください。

テンプレートの場所	説明
https://s3.amazonaws.com/cloudformation-templates-us-east-1/AccountGateSucceeded.yml	スタックを作成して、SUCCEEDED ステータスを返す Lambda アカウントゲート関数を実装します。
https://s3.amazonaws.com/cloudformation-templates-us-east-1/AccountGateFailed.yml	スタックを作成して、FAILED ステータスを返す Lambda アカウントゲート関数を実装します。

CloudFormation StackSets の同時実行モードの選択

[同時実行モード] は、StackSet オペレーション中の同時実行レベルの動作を選択できる [StackSetOperationPreferences](#) のパラメータです。以下のいずれかのモードを選択できます。

- 厳密な耐障害性: このオプションでは、失敗したアカウントの数が耐障害性 +1 の値を超えないように、同時実行レベルを動的に下げます。実際の同時実行数の初期値は、同時アカウントの最大数の値または耐障害性 +1 の値のいずれか低い方に設定されます。その場合、実際の同時実行数は障害の数に比例して減少します。これがデフォルトの動作です。
- ソフト耐障害性: このオプションは耐障害性と実際の同時実行性を切り離します。これにより、障害の数に関係なく、StackSet オペレーションを [同時アカウントの最大数] の値で設定された同時実行レベルで実行できます。

[厳格な耐障害性] では、StackSet オペレーションの障害が発生すると、障害ごとに同時実行性が低下するため、デプロイ速度が低下します。[ソフト耐障害性] は、CloudFormation 安全機能を活用しながらデプロイ速度を優先させます。これにより、既存のリソース、サービスクォータ、アクセス許可に関連する問題など、一般的な問題による StackSet オペレーションの障害を確認して対処できます。

StackSets スタック操作の失敗の詳細については、「[スタックオペレーション失敗の一般的な原因](#)」を参照してください。

同時アカウントの最大数と耐障害性の詳細については、「[StackSet オペレーションのオプション](#)」を参照してください。

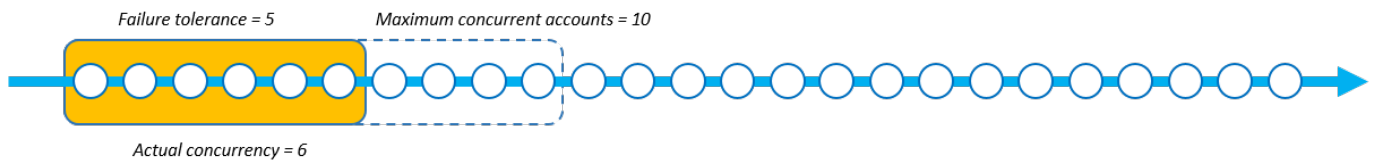
各同時実行モードの仕組み

以下の画像は、StackSet オペレーション中の各 [同時実行モード] の動作を視覚的に表したものです。ノードの文字列は 1 つの AWS リージョンへのデプロイを表し、各ノードがターゲット AWS アカウントです。

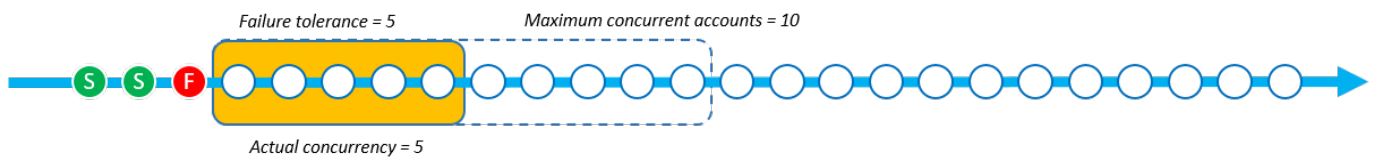
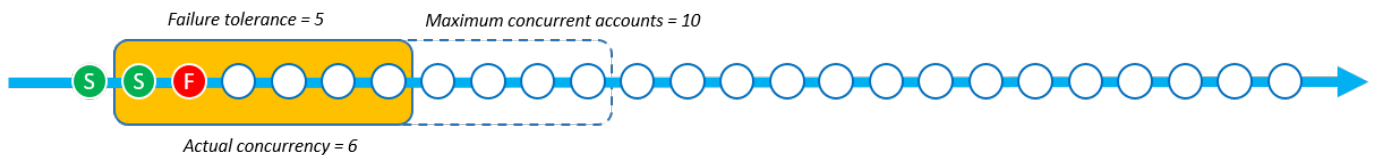
厳密な耐障害性

[厳格な耐障害性] を使用する StackSet オペレーションで、[耐障害性] の値が 5 に設定され、[同時アカウントの最大数] の値が 10 に設定されている場合、実際の同時実行数は 6 になります。これは、耐障害性の値 5 +1 が同時アカウントの最大数の値よりも低いためです。

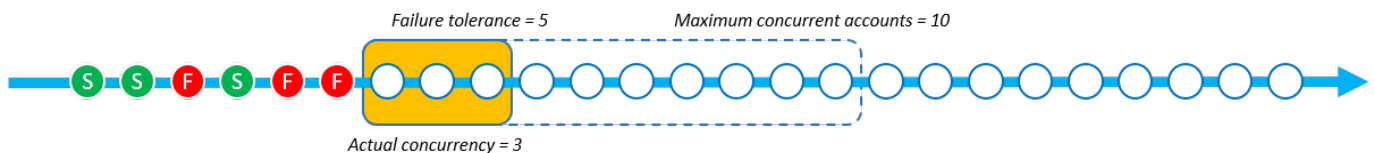
以下の図は、[耐障害性] の値が [同時アカウントの最大数] の値に与える影響と、両方が StackSet オペレーションの実際の同時実行性に与える影響を示しています。



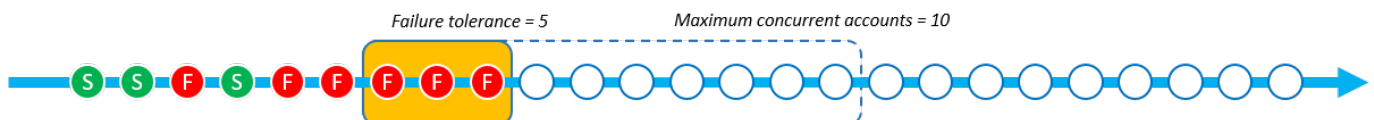
デプロイが開始され、スタックインスタンスに障害が発生すると、安全なデプロイを実現するために実際の同時実行数が減少します。StackSets が 1 つのスタックインスタンスのデプロイに失敗すると、実際の同時実行数は 6 から 5 に減少します。



厳密な耐障害性モードでは、障害が発生したスタックインスタンスの数に比例して実際の同時実行数が減少します。次の例では、StackSets がさらに 2 つのスタックインスタンスのデプロイに失敗し、実際の同時実行数が 5 から 3 に減少して、失敗したスタックインスタンスの合計が 3 になっています。



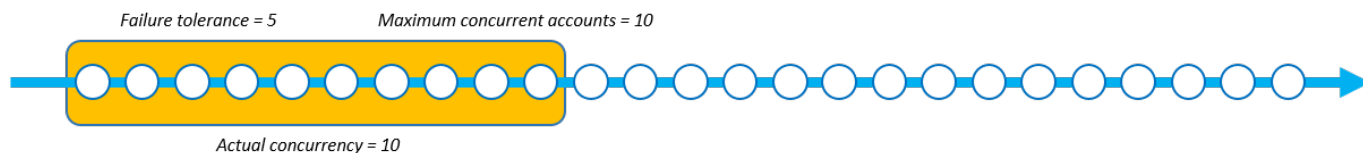
StackSets では、障害が発生したスタックインスタンスの数が [耐障害性] +1 の定義値に等しい場合、StackSet オペレーションに失敗します。次の例では、障害が発生したスタックインスタンスが 6 つあり、耐障害性の値が 5 の場合、StackSets は操作に失敗します。



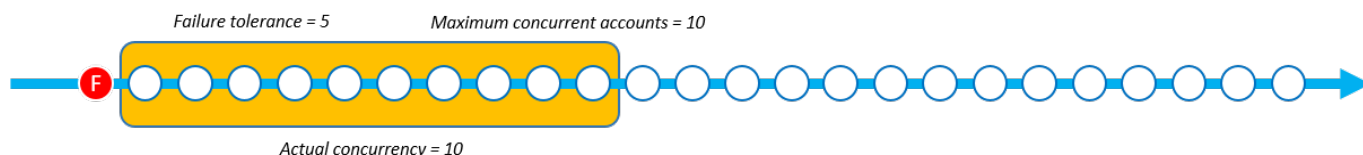
この例では、CloudFormation は StackSet オペレーションを停止する前に 9 つのスタックインスタンス (3 つは成功、6 つは失敗) をデプロイしました。

ソフト耐障害性

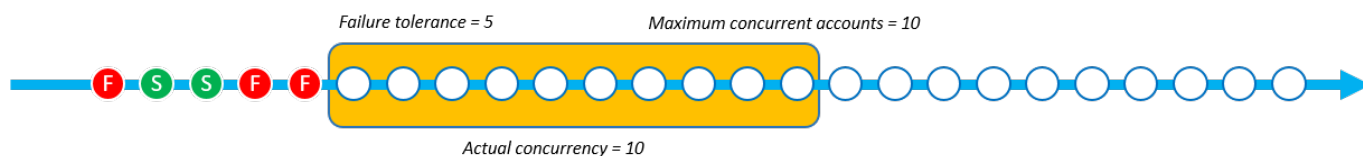
[ソフト耐障害性] を使用する StackSet オペレーションで、[耐障害性] の値が 5 に設定され、[同時アカウントの最大数] の値が 10 に設定されている場合、実際の同時実行数は 10 になります。



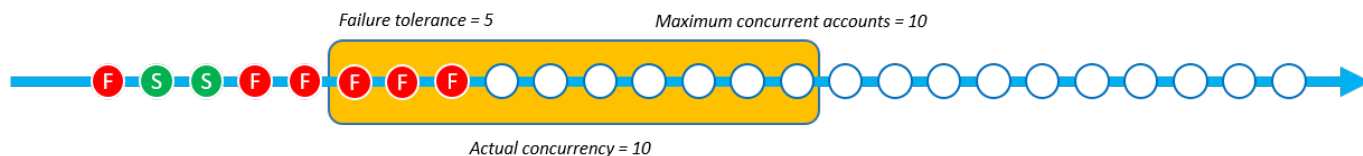
デプロイが開始され、スタックインスタンスに障害が発生しても、実際の同時実行数は変わりません。次の例では、1 つのスタック操作が失敗しましたが、実際の同時実行数は 10 のままです。



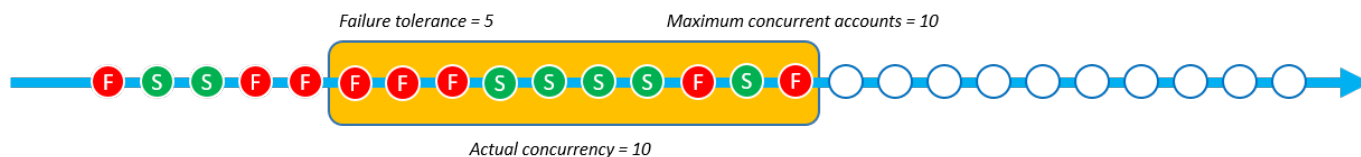
スタックインスタンスにさらに 2 回障害が発生しても、実際の同時実行数は 10 のままです。



StackSets では、障害が発生したスタックインスタンスの数が [耐障害性] を超えた場合、StackSet オペレーションに失敗します。次の例では、障害が発生したスタックインスタンスが 6 つあり、耐障害性のカウン트가 5 の場合、StackSets は操作に失敗します。ただし、同時実行キューの残りの操作が終了するまで操作は終了しません。



StackSets は、同時実行キューにすでに存在するスタックインスタンスを引き続きデプロイします。つまり、障害が発生したスタックインスタンスの数は耐障害性よりも多くなる可能性があるということです。次の例では、StackSet オペレーションが [耐障害性] の 5 に達したにもかかわらず、同時実行キューに実行すべきオペレーションが 7 つ残っているため、8 つのスタックインスタンスが失敗しています。



この例では、StackSets はスタックセットの操作を停止する前に 15 つのスタックインスタンス (7 つは成功、8 つは失敗) をデプロイしました。

デプロイ速度に基づいて、厳密な耐障害性とソフト耐障害性のどちらかを選択します。

[厳格な耐障害性] モードと [ソフト耐障害性] モードのどちらを選択するかは、StackSet のデプロイの優先速度と許容されるデプロイ失敗数に左右されます。

次の表は、合計 1,000 個のスタックインスタンスをデプロイしようとして失敗した StackSet オペレーションを、各同時実行モードがどのように処理するかを示しています。各シナリオでは、耐障害性の値は 100 スタックインスタンスに設定され、同時アカウントの最大数の値は 250 スタックインスタンスに設定されています。

StackSets は実際にはスライディングウィンドウとしてアカウントをキューに入れていますが ([「各同時実行モードの仕組み」](#)を参照)、この例では操作をバッチで示して各モードの速度を示しています。

厳密な耐障害性

この例では、厳密な耐障害性モードを使用しているため、先行する各バッチで発生した障害の数と比較して、実際の同時実行性が低下します。各バッチには 20 個の失敗したインスタンスがあり、StackSet オペレーションが [耐障害性] の値の 100 に達するまで、次のバッチの実際の同時実行数は 20 ずつ低下します。

次の表では、最初のバッチの実際の同時実行数は 101 スタックインスタンスです。実際の同時実行数は 101 です。これは、同時アカウントの最大数 (250) と耐障害性 (100) +1 のどちらか低い値であるためです。各バッチには 20 の失敗したスタックインスタンスのデプロイが含まれ、それ以降の各バッチの実際の同時実行数は 20 スタックインスタンスずつ減少します。

厳密な耐障害性	バッチ 1	バッチ 2	バッチ 3	バッチ 4	バッチ 5	バッチ 6
実際の同時実行数	101	81	61	41	21	-
失敗したインスタンス数	20	20	20	20	20	-
成功したスタックインスタンス数	81	61	41	21	1	-

[厳格な耐障害性] を使用したオペレーションでは、StackSet オペレーションが 100 個のスタックインスタンスの [耐障害性] の値に達するまでに、5 つのバッチで 305 個のスタックインスタンスのデプロイが完了しました。StackSet オペレーションは、失敗する前に 205 個のスタックインスタンスを正常にデプロイします。

ソフト耐障害性

ソフト耐障害性モードを使用したこの例では、障害が発生したインスタンスの数に関係なく、250 個のスタックインスタンスの同時アカウントの最大数の値で定義されている実際の同時実行数と同じ値が維持されます。StackSet オペレーションは、[耐障害性] の値である 100 個のインスタンスに達するまで、実際の同時実行数を維持します。

次の表では、最初のバッチの実際の同時実行数は 250 スタックインスタンスです。実際の同時実行数は 250 です。これは、同時アカウントの最大数の値が 250 に設定されており、ソフト耐障害性モードにより、StackSets は障害の数に関係なく、この値を実際の同時実行数として使用できるためです。この例では、各バッチに 50 件の障害が発生していますが、実際の同時実行性には影響しません。

ソフト耐障害性	バッチ 1	バッチ 2	バッチ 3	バッチ 4	バッチ 5	バッチ 6
実際の同時実行数	250	250	-	-	-	-

ソフト耐障害性	バッチ 1	バッチ 2	バッチ 3	バッチ 4	バッチ 5	バッチ 6
失敗したインスタンス数	50	50	-	-	-	-
成功したスタックインスタンス数	200	200	-	-	-	-

同じ同時アカウントの最大数の値と耐障害性の値を使用したソフト耐障害性モードでの操作では、2つのバッチで 500 件のスタックインスタンスのデプロイが完了しました。StackSet オペレーションは、失敗する前に 400 個のスタックインスタンスを正常にデプロイします。

同時実行モードの選択 (コンソール)

StackSet を作成または更新する際には、[デプロイオプションの設定] ページの [同時実行モード] で、[厳格な耐障害性] または [ソフト耐障害性] を選択します。

同時実行モードの選択 (AWS CLI)

ConcurrencyMode パラメータと次の StackSets コマンドを使用できます。

- [create-stack-instances](#)
- [delete-stack-instances](#)
- [detect-stack-set-drift](#)
- [import-stacks-to-stack-set](#)
- [update-stack-instances](#)
- [update-stack-set](#)

これらのコマンドには、ConcurrencyMode 設定を使用できる `--operation-preferences` という既存のパラメータがあります。ConcurrencyMode には次のいずれかの値を設定できます。

- STRICT_FAILURE_TOLERANCE
- SOFT_FAILURE_TOLERANCE

次の例では、`STRICT_FAILURE_TOLERANCE` `ConcurrencyMode` を使用し、`FailureToleranceCount` を 10、`MaxConcurrentCount` を 5 に設定したスタックインスタンスを作成します。

```
aws cloudformation create-stack-instances \  
  --stack-set-name example-stackset \  
  --accounts 123456789012 \  
  --regions eu-west-1 \  
  --operation-preferences  
    ConcurrencyMode=STRICT_FAILURE_TOLERANCE,FailureToleranceCount=10,MaxConcurrentCount=5
```

Note

StackSet の作成と更新の詳細な手順については、以下のトピックを参照してください。

- [StackSets を作成する \(セルフマネージド許可\)](#)
- [StackSets を作成する \(サービスマネージド許可\)](#)
- [StackSets を更新する](#)

CloudFormation StackSets でのドリフト検出の実行

スタックとスタックに含まれるリソースを CloudFormation で管理している場合でも、ユーザーは CloudFormation の外部でそれらのリソースを変更できます。ユーザーは、リソースを作成した基盤となるサービスを使用してリソースを直接編集できます。StackSet でドリフト検出を実行すると、その StackSet に属するスタックインスタンスのいずれかが予想される構成と異なっているか、またはドリフトしているかを判断できます。

トピック

- [CloudFormation が StackSet でドリフト検出を実行する方法](#)
- [StackSet のドリフトを検出する \(コンソール\)](#)
- [StackSet のドリフトを検出する \(AWS CLI\)](#)
- [StackSet でのドリフト検出の停止](#)

CloudFormation が StackSet でドリフト検出を実行する方法

CloudFormation が StackSet でドリフト検出を実行する場合、StackSet 内の各スタックインスタンスに関連付けられたスタックでドリフト検出が実行されます。これを行うために、CloudFormation は、スタック内の各リソースの現在の状態を、スタックのテンプレート、および指定された入力パラメータで定義されているリソースの予想される状態と比較します。リソースの現在の状態が予想される状態と異なる場合、そのリソースはドリフトしたものとみなされます。スタック内の 1 つ以上のリソースがドリフトしている場合、スタック自体がドリフトしたとみなされ、スタックが関連付けられているスタックインスタンスもドリフトしたとみなされます。StackSet 内の 1 つ以上のスタックインスタンスがドリフトした場合、StackSet 自体がドリフトしたとみなされます。

ドリフト検出では、管理されていない変更、つまり CloudFormation の外部にあるスタックに加えられた変更を識別します。StackSet レベルではなく、CloudFormation を介してスタックに直接行った変更は、ドリフトとはみなされません。例えば、StackSet のスタックインスタンスに関連付けられたスタックがあるとしします。CloudFormation を使用してそのスタックを更新して別のテンプレートを使用した場合、そのスタックのテンプレートが StackSet に属する他のスタックのテンプレートとは異なるようになった場合でも、ドリフトとはみなされません。これは、スタックが CloudFormation で予想されるテンプレートとパラメータ構成を一致させるためです。

CloudFormation がスタックでドリフト検出を実行する方法の詳細については、[ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#) を参照してください。

CloudFormation では各スタックで個別にドリフト検出を実行するため、スタックがドリフトしているかどうかを判断する際に、オーバーライドされたパラメータ値が考慮されます。スタックインスタンスでのテンプレートパラメータのオーバーライドの詳細については、「[CloudFormation StackSet 内のスタックのパラメータ値を上書きする](#)」を参照してください。

スタックインスタンスに関連付けられた[スタックで直接](#)ドリフト検出を実行する場合、これらのドリフト結果は [StackSets] コンソールページからは利用できません。

StackSet のドリフトを検出する (コンソール)

StackSet のドリフトを検出するには

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. [StackSets] ページで、ドリフト検出を実行する StackSet を選択します。
3. [アクション] メニューから、[Detect drifts (ドリフトを検出)] を選択します。

CloudFormation により、選択された StackSet に対してドリフト検出が開始されたことを示す情報バーが表示されます。

4. オプション: ドリフト検出オペレーションの進行状況を監視するには:
 - a. StackSet 名を選択して、[Stackset の詳細] ページを表示します。
 - b. [オペレーション] タブを選択し、ドリフト検出オペレーションを選択してから、[View drift details (ドリフトの詳細の表示)] を選択します。

CloudFormation に [Operation details (オペレーションの詳細)] ダイアログボックスが表示されます。

5. CloudFormation がドリフト検出オペレーションを完了するまで待ちます。ドリフト検出オペレーションが完了すると、CloudFormation によって、StackSet の [ドリフトステータス] と [前回のドリフトチェック時刻] が更新されます。これらのフィールドは、選択した StackSet の [StackSet の詳細] ページの [概要] タブにリストされます。

StackSet に含まれるスタックインスタンスの数や、StackSet に含まれるリソースの数によっては、ドリフト検出オペレーションに時間がかかる場合があります。一度に実行できるドリフト検出オペレーションは、特定の StackSet で 1 つだけです。情報バーを閉じた後でも、CloudFormation によってドリフト検出オペレーションが継続されます。

6. StackSet 内のスタックインスタンスのドリフト検出結果を確認するには、[スタックインスタンス] タブを選択します。

[スタックの名前] 列には、各スタックインスタンスに関連付けられたスタックの名前がリストされ、[ドリフトステータス] 列にはスタックのドリフトステータスがリストされます。スタックの 1 つ以上のリソースがドリフトした場合、スタックはドリフトしたと見なされます。

7. 特定のスタックインスタンスに関連付けられたスタックのドリフト検出結果を確認するには、次の手順を実行します。
 - a. [操作] タブを選択します。
 - b. 確認するドリフト検出結果のドリフト操作を選択します。分割パネルには、スタックインスタンスのステータスと選択した操作の理由が表示されます。ドリフト操作の場合、ステータス理由の列にはスタックインスタンスのドリフトステータスが表示されます。
 - c. 確認するドリフト詳細のスタックインスタンスを選択し、[リソースドリフトを表示] を選択します。「リソースドリフト」ページの [リソースドリフトステータス] テーブルでは、各スタックリソースはそれぞれのドリフトのステータスと、リソースにドリフト検出が実行さ

れた最後の時間が一覧表示されます。各リソースの論理 ID と物理 ID は、それらを識別しやすくするために表示されます。

8. ドリフトステータスに基づいてリソースをソートするには、[ドリフトステータス] 列を使用します。

変更されたリソースの詳細を表示する方法

- リソースを選択した状態で、[ドリフト詳細の表示] を選択します。

CloudFormation には、その特定リソースのドリフト詳細ページが表示されます。このページには、リソースの違いが一覧表示されます。リソースの予想プロパティ値と現在のプロパティ値も一覧表示されます。

Note

スタックが現在サインインしているリージョンとアカウントと異なるリージョンやアカウントに属している場合、[ドリフトを検出] ボタンは無効になり、詳細を表示できなくなります。

StackSet のドリフトを検出する (AWS CLI)

AWS CLI を使用してスタック全体のドリフトを検出するには、次の手順を使用します:

StackSet のドリフトを検出するには

1. StackSet 全体と関連するスタックインスタンスのドリフトを検出するには、[detect-stack-set-drift](#) コマンドを使用します。

次の例では、StackSet `stack-set-drift-example` でドリフト検出を開始します。

```
aws cloudformation detect-stack-set-drift \  
  --stack-set-name stack-set-drift-example
```

出力:

```
{  
  "OperationId": "c36e44aa-3a83-411a-b503-cb611example"  
}
```

2. StackSet ドリフト検出オペレーションは長時間実行されるオペレーションとなる可能性があるため、[describe-stack-set-operation](#) コマンドを使用してドリフトオペレーションのステータスをモニタリングします。このコマンドは、detect-stack-set-drift コマンドによって返された StackSet オペレーション ID を受け取ります。

次の例では、前の例のオペレーション ID を使用して、StackSet ドリフト検出オペレーションに関する情報を返します。この例では、オペレーションはまだ実行中です。この StackSet に関連付けられている 7 つのスタックインスタンスのうち、1 つのスタックインスタンスが既にドリフトしていて、2 つのインスタンスが同期しており、残り 4 つのスタックインスタンスのドリフト検出はまだ進行中であることがわかりました。1 つのインスタンスがドリフトしているため、StackSet 自体のドリフトステータスは DRIFTED になります。

```
aws cloudformation describe-stack-set-operation \  
  --stack-set-name stack-set-drift-example \  
  --operation-id c36e44aa-3a83-411a-b503-cb611example
```

出力:

```
{  
  "StackSetOperation": {  
    "Status": "RUNNING",  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "OperationPreferences": {  
      "RegionOrder": []  
    },  
    "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",  
    "StackSetDriftDetectionDetails": {  
      "DriftedStackInstancesCount": 1,  
      "TotalStackInstancesCount": 7,  
      "LastDriftCheckTimestamp": "2019-12-04T20:34:28.543Z",  
      "InSyncStackInstancesCount": 2,  
      "InProgressStackInstancesCount": 4,  
      "DriftStatus": "DRIFTED",  
      "FailedStackInstancesCount": 0  
    },  
    "Action": "DETECT_DRIFT",  
    "CreationTimestamp": "2019-12-04T20:33:13.673Z",  
    "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-  
a73f-8c22example",  
    "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
```

```
}  
}
```

後で同じコマンドを実行した、この例ではドリフト検出操作が完了した後に返される情報を示しています。この StackSet に関連付けられている合計 7 つのスタックインスタンスのうち 2 つがドリフトしているため、StackSet 自体のドリフトステータスは DRIFTED になっています。

```
aws cloudformation describe-stack-set-operation \  
  --stack-set-name stack-set-drift-example \  
  --operation-id c36e44aa-3a83-411a-b503-cb611example
```

出力:

```
{  
  "StackSetOperation": {  
    "Status": "SUCCEEDED",  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "OperationPreferences": {  
      "RegionOrder": []  
    }  
  },  
  "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",  
  "EndTimestamp": "2019-12-04T20:37:32.829Z",  
  "StackSetDriftDetectionDetails": {  
    "DriftedStackInstancesCount": 2,  
    "TotalStackInstancesCount": 7,  
    "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",  
    "InSyncStackInstancesCount": 5,  
    "InProgressStackInstancesCount": 0,  
    "DriftStatus": "DRIFTED",  
    "FailedStackInstancesCount": 0  
  },  
  "Action": "DETECT_DRIFT",  
  "CreationTimestamp": "2019-12-04T20:33:13.673Z",  
  "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-  
a73f-8c22example",  
  "OperationId": "c36e44aa-3a83-411a-b503-cb611example"  
}
```

3. StackSet ドリフト検出オペレーションが完了したら、describe-stack-set、list-stack-instances、describe-stack-instance、list-stack-instance-resource-drifts コマンドを使用して結果を確認します。

[describe-stack-set](#) コマンドには、describe-stack-set-operation コマンドで返されるのと同じ詳細なドリフト情報が含まれます。

```
aws cloudformation describe-stack-set \  
  --stack-set-name stack-set-drift-example
```

出力:

```
{  
  "StackSet": {  
    "Status": "ACTIVE",  
    "Description": "Demonstration of drift detection on stack sets.",  
    "Parameters": [],  
    "Tags": [  
      {  
        "Value": "Drift detection",  
        "Key": "Feature"  
      }  
    ],  
    "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",  
    "Capabilities": [],  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "StackSetDriftDetectionDetails": {  
      "DriftedStackInstancesCount": 2,  
      "TotalStackInstancesCount": 7,  
      "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",  
      "InProgressStackInstancesCount": 0,  
      "DriftStatus": "DRIFTED",  
      "DriftDetectionStatus": "COMPLETED",  
      "InSyncStackInstancesCount": 5,  
      "FailedStackInstancesCount": 0  
    },  
    "StackSetARN": "arn:aws:cloudformation:us-east-1:123456789012:stackset/  
stack-set-drift-example:bd1f4017-d4f9-432e-a73f-8c22example",  
    "TemplateBody": [details omitted],  
    "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-  
a73f-8c22ebexample",  
  }  
}
```

```
    "StackSetName": "stack-set-drift-example"
  }
}
```

[list-stack-instances](#) コマンドを使用して、各スタックインスタンスのドリフトステータスなど、StackSet に関連付けられたスタックインスタンスに関する概要情報を返すことができます。

この例では、ドリフトステータスフィルターを DRIFTED に設定した状態で例の StackSet に list-stack-instances を実行すると、2 つのスタックインスタンスのうち、どちらが DRIFTED のドリフトステータスになっているかを特定できるようになります。

```
aws cloudformation list-stack-instances \
  --stack-set-name stack-set-drift-example \
  --filters Name=DRIFT_STATUS,Values=DRIFTED
```

出力:

```
{
  "Summaries": [
    {
      "StackId": "arn:aws:cloudformation:eu-west-1:123456789012:stack/StackSet-stack-set-drift-example-b0fb6083-60c0-4e39-af15-2f071e0db90c/0e4f0940-16d4-11ea-93d8-0641cexample",
      "Status": "CURRENT",
      "Account": "012345678910",
      "Region": "eu-west-1",
      "LastDriftCheckTimestamp": "2019-12-04T20:37:32.687Z",
      "DriftStatus": "DRIFTED",
      "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-a73f-8c22eexample",
      "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
    },
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
      "Status": "CURRENT",
      "Account": "123456789012",
      "Region": "us-east-1",
      "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
      "DriftStatus": "DRIFTED",

```

```

        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
        "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
    },
    [additional stack instances omitted]
]
}

```

[describe-stack-instance](#) コマンドでもこの情報は返されますが、以下の例のように 1 つのスタックインスタンスに対する情報です。

```

aws cloudformation describe-stack-instance \
  --stack-set-name stack-set-drift-example \
  --stack-instance-account 012345678910 --stack-instance-region us-east-1

```

出力:

```

{
  "StackInstance": {
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-
ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
    "Status": "CURRENT",
    "Account": "123456789012",
    "Region": "us-east-1",
    "ParameterOverrides": [],
    "DriftStatus": "DRIFTED",
    "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
    "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
    "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
  }
}

```

4. どのスタックインスタンスがドリフトしたかを特定したら、`list-stack-instances` コマンドまたは `describe-stack-instance` コマンドによって返されるスタックインスタンスに関する情報を使用して、[list-stack-instance-resource-drifts](#) コマンドを実行できます。このコマンドは、スタックでのリソースが、特定のドリフト操作でドリフトしたかに関する詳細情報を返します。

次の例では、`--stack-instance-resource-drift-statuses` パラメーターを使用して、以前のドリフト操作の例で変更または削除されたリソースのスタックドリフト情報を要求しています。リクエストは、2つのプロパティに関する詳細とそれぞれの変更された値を含め、変更された1つのリソースに関する情報を返します。リソースは削除されていません。

```
aws cloudformation list-stack-instance-resource-drifts \
  --stack-set-name my-stack-set-with-resource-drift \
  --stack-instance-account 123456789012 \
  --stack-instance-region us-east-1 \
  --operation-id c36e44aa-3a83-411a-b503-cb611example \
  --stack-instance-resource-drift-statuses MODIFIED DELETED
```

出力:

```
{
  "Summaries": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/my-stack-set-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
      "ResourceType": "AWS::SQS::Queue",
      "Timestamp": "2018-03-26T17:23:34.489Z",
      "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/123456789012/my-stack-with-resource-drift-Queue-494PBHC076H4",
      "StackResourceDriftStatus": "MODIFIED",
      "PropertyDifferences": [
        {
          "PropertyPath": "/DelaySeconds",
          "ActualValue": "120",
          "ExpectedValue": "20",
          "DifferenceType": "NOT_EQUAL"
        },
        {
          "PropertyPath": "/RedrivePolicy/maxReceiveCount",
          "ActualValue": "12",
          "ExpectedValue": "10",
          "DifferenceType": "NOT_EQUAL"
        }
      ],
      "LogicalResourceId": "Queue"
    }
  ]
}
```

```
}
```

StackSet でのドリフト検出の停止

StackSet のドリフト検出は長時間実行されるオペレーションになる可能性があるため、StackSet で現在実行中のドリフト検出オペレーションを停止するのが望ましい場合があります。

StackSet のドリフト検出を停止するには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. [StackSets] ページで、StackSet の名前を選択します。

CloudFormation に選択した StackSet の [StackSets の詳細] ページが表示されます。

3. [StackSets details (セットのスタックセットの詳細)] ページで、[オペレーション] タブを選択し、ドリフト検出オペレーションを選択します。
4. [Stop operation (オペレーションを停止)] を選択します。

StackSet のドリフト検出を停止するには (AWS CLI)

- [stop-stack-set-operation](#) コマンドを使用します。ドリフト検出 StackSet オペレーションの StackSet 名とオペレーション ID の両方を指定する必要があります。

```
aws cloudformation stop-stack-set-operation \  
  --stack-set-name stack-set-drift-example \  
  --operation-id 624af370-311a-11e8-b6b7-500cexample
```

スタックを CloudFormation StackSets にインポートする

スタックのインポートオペレーションでは、既存のスタックを新規または既存の StackSets にインポートできるため、既存のスタックを 1 回のオペレーションで StackSet に移行できます。

セルフマネージド型スタックセットの場合、インポートオペレーションでは、管理者アカウントまたは異なるターゲットアカウントと AWS リージョン にあるスタックをインポートできます。サービス管理された StackSets の場合、インポートオペレーションでは、管理アカウントと同じ AWS Organizations の任意のスタックをインポートできます。

スタックを StackSets にインポートする際の考慮事項と制限事項は次のとおりです:

- インポートオペレーションでは、インラインスタック ID を使用して最大 10 スタック、または Amazon S3 オブジェクトを使用して最大 200 スタック、インポートできます。
- NoEcho プロパティはサポートされていません。NoEcho を含むスタックは、StackSet のインポートを介して新しい StackSets にインポートされません。
- スタックは、1 つの StackSet にのみ属することができます。
- スタックのインポートオペレーションでパラメータとしてタグを明示的に指定することで、StackSet にスタックタグを実装できます。
- スタックのカスタムパラメータオーバーライドは、インポートオペレーション中には影響を受けません。
- インポートされたスタックインスタンスのパラメータを [StackSet の詳細を編集] オプションを使用して更新することはできません。これには [StackSet パラメータを上書き] を使用する必要があります。パラメータの上書きに関する詳細については、「[スタックのパラメータを上書きする](#)」を参照してください。
- StackSets クォータとスタックインスタンスは、スタックをインポートするときに適用されます。クォータの詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

トピック

- [CloudFormation StackSets のためのセルフマネージドスタックインポート](#)
- [CloudFormation StackSets のためのサービスマネージドスタックインポート](#)
- [CloudFormation StackSets へのスタックのインポートを元に戻す](#)

CloudFormation StackSets のためのセルフマネージドスタックインポート

CloudFormation スタックのインポートオペレーションでは、既存のスタックを新規または既存の StackSets にインポートできるため、既存のスタックを 1 回のオペレーションで StackSet に移行できます。スタックインポートを使用することで、それらのリソースを削除して再作成することなく、ダウンタイムや停止を回避できます。スタックが StackSet にインポートされると、元のスタックは指定されたスタックセットのスタックインスタンスになります。

セルフマネージドスタックのインポートに関する考慮事項

- スタックのインポートオペレーションには、StackSet を作成する管理者アカウントと、スタックを含むターゲットアカウントが必要です。
- ターゲットアカウントには、スタック ID または ARN の入力で GetTemplate オペレーションを使用する権限が必要です。そのため、管理者ア

メントには `AWSCloudFormationStackSetAdministrationRole` または `AWSCloudFormationStackSetsExecutionRole` 許可を付与する必要があります。

トピック

- [既存のスタックを新しい StackSet にインポートする \(コンソール\)](#)
- [既存のスタックを既存の StackSet にインポートする \(コンソール\)](#)
- [スタックを StackSet にインポートする \(AWS CLI\)](#)

既存のスタックを新しい StackSet にインポートする (コンソール)

開始する前に、インポートするスタックを特定します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. [StackSets] ページの上部で [Create StackSet (StackSet の作成)] を選択します。
4. [テンプレートを選択する] ページで、次のいずれかのオプションでテンプレートを指定し、[次へ] を選択します。
 - Amazon S3 URL を選択し、テキストボックスでテンプレートの URL を指定します。
 - [Upload a template file] (テンプレートファイルのアップロード) を選択し、テンプレートを参照します。
 - [From stack ID] (スタック ID から) を選択し、スタック ID を入力します。
5. [StackSet の詳細を指定] ページで、作成する StackSet の名前を入力し、[次へ] を選択します。

(オプション) StackSet の説明を入力します。

6. [スタックオプションの設定] ページで、選択内容を確認し、[次へ] を選択します。
7. [デプロイオプションの設定] ページで、[スタックをスタックセットにインポート] を選択します。
8. インポートするスタックのスタック ID を [Stacks to import] (インポートするスタック) フィールドに入力します。例えば、`arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786`。

(オプション) [別のスタック ID の追加] を追加するを選択し、インポートする別のスタックのスタック ID を入力します。スタックのインポートオペレーションごとに最大 10 個のスタックを追加できます。

9. デプロイのオプションを確認し、[次へ] を選択します。

10.[レビュー] ページで、設定と StackSet のプロパティを確認します。スタックを StackSet にインポートする準備ができたなら、[送信] を選択します。

結果: インポートされたスタックは、指定された StackSet のスタックインスタンスになりました。スタックインポートステータスの詳細については、「[StackSets ステータスコード](#)」を参照してください。

既存のスタックを既存の StackSet にインポートする (コンソール)

開始する前に、インポートするスタックを特定します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. [StackSets] ページで、スタックをインポートする StackSet を選択します。
4. StackSet を選択した状態で、[アクション] メニューから [StackSet にスタックを追加] を選択します。
5. [デプロイオプションの設定] ページで、[スタックをスタックセットにインポート] を選択し、インポートするスタックのスタック ID を [インポートするスタック] に入力します。例えば、`arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786`。

(オプション) [別のスタック ID の追加] を追加するを選択し、インポートする別のスタックのスタックID を入力します。スタックのインポートオペレーションごとに最大 10 個のスタックを追加できます。

6. [次へ] を選択します。
7. [Specify overrides] (オーバーライドを指定) ページで、選択内容を確認し、[次へ] を選択します。
8. [レビュー] ページで、設定と StackSet のプロパティを確認します。StackSet を作成する準備ができたなら、[送信] を選択します。

結果: インポートされたスタックは、指定された StackSet のスタックインスタンスになりました。スタックインポートステータスの詳細については、「[StackSets ステータスコード](#)」を参照してください。

スタックを StackSet にインポートする (AWS CLI)

既存のスタックを新しい StackSet にインポートするには

次の `create-stack-set` コマンドは StackSet を作成し、指定されたスタックをインポートします。インポートするスタックは ARN によって識別されます。placeholder text を独自の情報に置き換えます。

```
aws cloudformation create-stack-set \  
  --stack-set-name MyStackSet \  
  --stack-id arn:aws:cloudformation:us-east-1:123456789012:stack/  
StackToImport/466df9e0-0dff-08e3-8e2f-5088487c4896 \  
  --administration-role-arn arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole \  
  --execution-role-name AWSCloudFormationStackSetExecutionRole
```

既存のスタックを既存の StackSet にインポートするには

次の `import-stacks-to-stack-sets` コマンドは、指定されたスタックを *MyStackSet* StackSet にインポートします。インポートするスタックは ARN によって識別されます。placeholder text を独自の情報に置き換えます。

```
aws cloudformation import-stacks-to-stack-set \  
  --stack-set MyStackSet \  
  --stack-ids arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/  
f449b250-b969-11e0-a185-5081d0136786
```

複数のスタックを指定するには、`--stack-ids` オプションの値に次の形式を使用します。

```
--stack-ids "arn_1" "arn_2"
```

インポートしたスタックのクローンを他のリージョンやアカウントに作成するには

次の `create-stack-instances` コマンドは、StackSet にスタックインスタンスを追加します。placeholder text を独自の情報に置き換えます。

```
aws cloudformation create-stack-instances \  
  --stack-set-name MyStackSet \  
  --accounts '["account_ID_1","account_ID_2"]' \  
  --regions '["region_1","region_2"]'
```

CloudFormation StackSets のためのサービスマネージドスタックインポート

CloudFormation スタックのインポートオペレーションでは、既存のスタックを新規または既存の StackSets にインポートできるため、既存のスタックを 1 回のオペレーションで StackSet に移行できます。StackSets はスタックの機能を拡張するため、1 回のオペレーションで複数のアカウントとリージョンにまたがるスタックを作成、更新、削除できます。

サービスマネージドスタックのインポートに関する考慮事項

- スタックのインポートオペレーションには、StackSets で信頼アクセスを有効にするなど、関連付けられた AWS Organizations を管理できる管理アカウントまたは委任された管理アカウントが必要です。
- ターゲットアカウントは、管理アカウントまたは委任された管理者アカウントによって管理される AWS Organizations のメンバーである必要があります。
- ターゲットスタックは、ターゲット OU の 1 つに存在します。
- ターゲットアカウントは AWS Organizations のメンバーである必要があります。
- AWS Organizations アクセスは、組織の ACTIVATED 状態である必要があります。
- インポートされるスタックは、管理アカウントではなく、メンバーアカウントのいずれかに存在する必要があります。

トピック

- [サービスマネージド型のスタックを新しい StackSet にインポートする \(コンソール\)](#)
- [サービスマネージド型のスタックを作成して、既存の StackSet にインポートする \(コンソール\)](#)
- [サービスマネージド型のスタックを既存の StackSet にインポートする \(コンソール\)](#)
- [サービスマネージド型のスタックを StackSet にインポートする \(AWS CLI\)](#)

サービスマネージド型のスタックを新しい StackSet にインポートする (コンソール)

AWS マネジメントコンソール を使用して、スタックを新しい StackSet にインポートする

新しいスタックを StackSet にインポートするには、インポートするリソースを含むスタックを特定します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。

2. ナビゲーションペインから [StackSets] を選択します。
3. [StackSets] ページの上部で [Create StackSet (StackSet の作成)] を選択します。
4. [テンプレートを選択] ページで、次のようにします。
 - a. [StackSet アクセス許可モデル] で、[サービス管理のアクセス許可] (を選択します。
 - b. 前提条件 - テンプレートの準備 で、[テンプレートの準備完了] を選択し、次のいずれかのオプションを使用してテンプレートを選択します。
 - [Amazon S3 URL] で、[Amazon S3 URL] フィールドに Amazon S3 URL を入力します。
 - [テンプレートファイルのアップロード] で、ローカルコンピュータの CloudFormation テンプレートを選択します。

設定を受け入れて、[Next] (次へ) を選択してください。

5. [StackSet の詳細を指定] ページで、以下を実行します。
 - a. [StackSet 名] ボックスに StackSet 名を入力します。
 - b. (オプション) [StackSet description] (StackSet の説明) セクションに説明を入力します。

[スタックオプションの設定] ページで、選択内容を確認し、[次へ] を選択します。

6. [デプロイオプションの設定] ページで、以下を実行します。
 - a. [StackSet にスタックを追加] で、[スタックをスタックセットにインポート] を選択します。
 - b. [Stacks to import] (インポートするスタック) で、スタックのインポート方法を選択します。
 - i. [Stack ID] (スタック ID) には、スタック ID を入力します。
 - ii. [スタック URL] で、Amazon S3 URL を入力します。
7. [組織単位に関連付ける] で、次を実行します:
 - a. ルート OU を使用するには、[組織に関連付ける] を選択します。
 - b. [組織単位 (OU) に関連付ける] を選択して、インポートするスタックの親 OU ID を入力します。例えば、Stack 1 と Stack 2 が OU1 未満で、Stack 3 が OU2 未満の場合は、OU1 と OU2 を入力します。

設定を受け入れて、[Next] (次へ) を選択してください。

8. [Review] (確認) ページで、設定を確認し、[Submit] (送信) を選択します。

サービスマネージド型のスタックを作成して、既存の StackSet にインポートする (コンソール)

既存のスタックを新しい StackSet にインポートするには、インポートするリソースを含むスタックを特定します。

StackSet を作成してスタックをインポートするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. [StackSets] ページの上部で [Create StackSet (StackSet の作成)] を選択します。
4. [テンプレートを選択] ページで、次のようにします。
 - a. [StackSet アクセス許可モデル] で、[サービス管理のアクセス許可] (を選択します。
 - b. 前提条件 - テンプレートの準備 で、[テンプレートの準備完了] を選択し、次のいずれかのオプションを使用してテンプレートを選択します。
 - [Amazon S3 URL] で、[Amazon S3 URL] フィールドに Amazon S3 URL を入力します。
 - [テンプレートファイルのアップロード] で、ローカルコンピュータの CloudFormation テンプレートを選択します。

設定を受け入れて、[Next] (次へ) を選択してください。

5. [StackSet の詳細を指定] ページで、以下を実行します。
 - a. [StackSet 名] ボックスに StackSet 名を入力します。
 - b. (オプション) [StackSet description] (StackSet の説明) セクションに説明を入力します。

[スタックオプションの設定] ページで、選択内容を確認し、[次へ] を選択します。

6. [デプロイオプションの設定] ページで、以下を実行します。
 - [Add stacks to stack set] (スタックセットにスタックを追加) で、[Deploy new stacks] (新しいスタックのデプロイ) を選択します。
7. 次の情報を [組織単位を関連付ける] セクションに追加します。
 - a. ルート OU を使用するには、[組織に関連付ける] を選択します。

- b. [組織単位 (OU) に関連付ける] を選択して、インポートするスタックの親 OU ID を入力します。例えば、Stack 1 と Stack 2 が OU1 未満で、Stack 3 が OU2 未満の場合は、OU1 と OU2 を入力します。
8. [Specify regions] (リージョンの指定) と [Deployment options] (デプロイオプション) で、選択内容を確認します。

設定を受け入れて、[Next] (次へ) を選択してください。

9. [Review] (確認) ページで、設定を確認し、[Submit] (送信) を選択します。

サービスマネージド型のスタックを既存の StackSet にインポートする (コンソール)

StackSet を選択し、インポートするスタックを特定します。

スタックを既存の StackSet にインポートするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. ナビゲーションペインから [StackSets] を選択します。
3. スタックをインポートする StackSet を選択し、[アクション] ドロップダウンから [StackSet にスタックを追加] を選択します。
4. [デプロイオプションの設定] ページで、以下を実行します。
 - a. [StackSet にスタックを追加] で、[スタックをスタックセットにインポート] を選択します。
 - b. [インポートするスタック] で、次の操作を行います。
 - i. [Stack ID] (スタック ID) には、スタック ID を入力します。
 - ii. [スタック URL] で、Amazon S3 URL を入力します。
 - c. [組織単位に関連付ける] で、次を実行します:
 - i. ルート OU を使用するには、[組織に関連付ける] を選択します。
 - ii. [組織単位 (OU) に関連付ける] を選択して、インポートするスタックの親 OU ID を入力します。例えば、Stack 1 と Stack 2 が OU1 未満で、Stack 3 が OU2 未満の場合は、OU1 と OU2 を入力します。

設定を受け入れて、[Next] (次へ) を選択してください。

5. [Specify overrides] (上書きの指定) ページを確認し、[Next] (次へ) を選択します。

6. [Review] (確認) ページを確認して見直し、[Submit] (送信) を選択します。

サービスマネージド型のスタックを StackSet にインポートする (AWS CLI)

StackSet が作成されたら、インポートするスタックのスタック ID を渡すことでスタックをインポートできます。マッピング先の OU ID リストを渡すこともできます。

CloudFormation は、これらの OU 内のユーザー提供のスタックをインポートし、それらの OU を StackSet のデプロイターゲットとして使用します。入力に示されたスタック ID は、内部的に OU ID リスト入力の最も近い OU にマッピングされます。スタックが入力リスト内の既存の OU ID に属していない場合、AWS CLI は StackNotFoundException エラーを返します。

import-stacks-to-stack-set オペレーションは、OU ID 入力のスタックのスタックインスタンスを作成します。次の AWS CLI の例では、import-stacks-to-stack-set オペレーションを使用して、スタックを StackSet にインポートします。

- import-stacks-to-stack-sets オペレーションを使用するには、スタックセットにインポートする stack-ids または stack-ids-url を指定します。

```
aws cloudformation import-stacks-to-stack-set \  
  --stack-set-name ServiceManagedStackSet \  
  --stack-ids "arn:123456789012:us-east-1:Stack1" \  
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

```
aws cloudformation import-stacks-to-stack-set \  
  --stack-set-name ServiceManagedStackSet \  
  --stack-ids-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-name.json \  
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

Note

import-stacks-to-stack-sets オペレーションでは、インポートされるスタックをその特定の OU に関連付けることができるように、少なくとも 1 つの組織単位 ID (OU ID) を指定する必要があります。このオペレーションでは、関連する OU 内の他のメンバーアカウントのスタックインスタンスは作成されません。関連する OU のメンバーアカウントを更新するには、create-stack-instances または update-stack-instances を使用します。

create-stack-set は、直接アップロードまたは Amazon S3 のいずれかから、ユーザー提供のテンプレートを使用して、OU の下のすべてのアカウントのスタック インスタンスを作成します。次の AWS CLI の例では、create-stack-set オペレーションを使用して、スタックを新しい StackSet にインポートします。

- create-stack-set オペレーションを使用するには、StackSet 名を指定し、新しく作成した StackSet にスタックをインポートします。

```
aws cloudformation create-stack-set \  
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-name.json \  
  --permission-model SERVICE_MANAGED \  
  --auto-deployment Enabled=true
```

CloudFormation StackSets へのスタックのインポートを元に戻す

スタックインスタンスに不要な変更がある場合は、スタックインスタンスのインポートを元に戻すことができます。

スタックインスタンスのインポートを元に戻すと、CloudFormation は StackSet からスタックインスタンスを削除しますが、スタックのリソースは保持します。

スタックのインポートオペレーションを元に戻すには、次の手順を完了します。

1. スタックインスタンスが削除された後に保持するリソースごとに Retain の DeletionPolicy 属性を指定します。詳細については、「[インポートオペレーションを元に戻す](#)」を参照してください。
2. StackSet からスタックインスタンスを削除します。詳細については、「[CloudFormation StackSets からスタックを削除する](#)」を参照してください。
3. StackSet を削除します。詳細については、「[CloudFormation StackSets を削除する](#)」を参照してください。

CloudFormation StackSets を使用するためのベストプラクティス

このセクションでは、StackSet テンプレートの定義、StackSet へのスタックの作成または追加、または StackSet の更新を実行するためのベストプラクティスを説明します。

CloudFormation を初めて使用する場合は、[CloudFormation ベストプラクティス](#) トピックで CloudFormation をより効果的かつ安全に使用するのに役立つ他のレコメンデーションを確認してください。

トピック

- [テンプレートの定義](#)
- [StackSet にスタックを作成または追加する](#)
- [StackSet でスタックを更新する](#)

テンプレートの定義

- 複数のリージョン内の、複数のアカウント内で標準化したいテンプレートを定義します。
- テンプレートを作成するときに、グローバルリソース (IAM ロールや Amazon S3 バケットなど) を同一アカウントの 1 つ以上のリージョンで作成する場合は、名前の競合が起きないように注意してください。
- StackSet は単一のテンプレートとパラメータのセットを持ちます。StackSet に関連付けられたすべてのアカウントで同じスタックが作成されます。テンプレートを作成するときは、制御と標準化のバランスが取れるよう、十分に細かくしてください。
- テンプレートは Amazon S3 バケットに保存することをお勧めします。

StackSet にスタックを作成または追加する

- 多数のスタックインスタンスを StackSet に追加する前に、スタックインスタンスを初期の StackSet に追加することが機能することを確認してください。
- ユースケースで機能するデプロイ (ロールアウト) オプションを選択します。
 - デプロイをより控えめにするには、[Maximum Concurrent Accounts (同時アカウントの最大数)] を 1 に、[Failure Tolerance (耐障害性)] を 0 に設定してください。影響が最も小さいリージョンが [Region Order (リージョンオーダー)] リストの先頭になるように設定します。1 つのリージョンから開始します。
 - デプロイを高速にする場合、最大アカウントの同時数と耐障害性の値を増やします。
- StackSets のオペレーションは、関係するスタックインスタンスの数に依存し、莫大な時間がかかることがあります。

StackSet でスタックを更新する

- デフォルトでは、StackSet を更新すると、すべてのスタックインスタンスが更新されます。2 つのリージョンにそれぞれ 20 個のアカウントがある場合、40 個のスタックインスタンスを持ち、StackSet を更新するとすべて更新されます。

多数のスタックインスタンスを含む StackSets の場合、テンプレートの更新バージョンをテストするには、すべてのスタックインスタンスを更新する前に、いくつかのテストアカウントでスタックインスタンスを選択的に更新することをお勧めします。

- StackSet 内の個々のスタックの更新をよりきめ細かく制御するには、複数の StackSets を作成することを計画してください。
- 多数のスタックを持つ StackSet を更新するには、長い時間がかかることがあります。このリリースでは、一度に 1 つのオペレーションのみが StackSet に許可されます。StackSet で他のオペレーションを実行できなくなることがないように更新を計画してください。

CloudFormation StackSets サンプルテンプレート

このセクションには、エンタープライズで CloudFormation StackSets を使用する手助けとなる、いくつかの CloudFormation テンプレートへのリンクが含まれています。このセクションに示すテンプレートは [AWS CloudTrail](#) または [AWS Config](#) と共に、その中のルールも有効にします。

Important

AWS Config から Amazon S3 バケットへのアクセスを許可する場合のセキュリティのベストプラクティスとして、バケットポリシーで `AWS:SourceAccount` 条件を使用してアクセスを制限することを強くお勧めします。新しいテンプレートは、`AWS:SourceAccount` を含むよう更新されました。既存のバケットポリシーがこのセキュリティのベストプラクティスに従わない場合は、この保護を含めるようにバケットポリシーを編集することを強くお勧めします。これにより、AWS Config は想定されるユーザーにのみアクセス権が付与されます。

説明	S3 リンク
AWS CloudTrail の有効化	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml

説明	S3 リンク
AWS Config の有効化	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnabledAWSConfig.yml
中央ログで AWS Config を有効にする	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnabledAWSConfigForOrganizations.yml
AWS 組織、または特定の AWS アカウント全体で Amazon Data Lifecycle Manager デフォルトポリシーを有効にする	• EBS スナップショットのデフォルトポリシー – https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerEBSSnapshotDefaultPolicy.yml • EBS-backed AMI のデフォルトポリシー – https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerAMIDefaultPolicy.yml https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml
CloudTrail が有効になっているかどうかを判断するために AWS Config ルールを設定する	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleCloudtrailEnabled.yml
ルート MFA が有効になっているか判断するために AWS Config を設定する	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleRootAccountMFAEnabled.yml
EIP が添付されているか判断するために AWS Config を設定する	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEipAttached.yml

説明	S3 リンク
EBS ボリュームが暗号化されているか判断するために AWS Config を設定する	https://s3.amazonaws.com/cloudformation-stacks/sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml

CloudFormation StackSets のトラブルシューティング

このトピックでは、StackSets の一般的な問題と推奨される解決方法について説明します。

トピック

- [スタックオペレーション失敗の一般的な原因](#)
- [失敗したスタック作成または更新オペレーションを再試行する](#)
- [スタックインスタンスの削除に失敗する場合](#)
- [スタックのインポートオペレーションが失敗する](#)
- [StackSets オペレーションのスタックインスタンスの失敗回数](#)

スタックオペレーション失敗の一般的な原因

問題: スタックオペレーション失敗。スタックインスタンスのステータスは OUTDATED です。

原因: スタックオペレーションが失敗する一般的な原因は複数あります。

- テンプレートで指定されているリソースを作成するターゲットアカウントのアクセス権限が不十分。
- CloudFormation テンプレートでエラーが発生する場合があります。CloudFormation のテンプレートを検証し、StackSet を作成する前にエラーを修正します。
- テンプレートで、グローバルリソースを作成しようとしています。S3 バケットのように一意である必要がありますが、このリソースは一意ではありません。
- 指定されたターゲットアカウント番号は存在しません。ウィザードの [Set deployment options (デプロイオプションの設定)] ページで指定したターゲットアカウント番号を確認します。
- この管理者アカウントには、ターゲットアカウントとの信頼関係はありません。
- テンプレートで指定されているリソースの最大数が既にターゲットアカウントに存在しています。たとえば、ターゲットアカウントで許可されている IAM ロールの制限に達していますが、テンプレートを使用して、さらに IAM ロールを作成できます。

- StackSet で許可されているスタックの最大数に達しました。StackSet あたりのスタックの最大数については、「[CloudFormation クォータを理解する](#)」を参照してください。

ソリューション: StackSet を作成する前にターゲットアカウントおよび管理者アカウントで必要とされるアクセス許可の詳細については、「[管理者アカウントのすべてのユーザーに、すべてのターゲットアカウントのスタックを管理するための許可を付与します。](#)」を参照してください。

失敗したスタック作成または更新オペレーションを再試行する

問題: スタック作成またはスタック更新の失敗。スタックインスタンスのステータスは OUTDATED です。スタックの作成または更新に失敗した場合にトラブルシューティングするには、CloudFormation コンソールを開いてスタックのイベントを表示します。ステータスは、DELETED (作成オペレーションに失敗した場合)、または FAILED (更新オペレーションに失敗した場合) と表示されます。スタックイベントを参照し、[状況の理由] 列を検索します。[状況の理由] の値は、スタックオペレーションが失敗した理由を表します。

スタック作成失敗の根本的な原因を修正したら、スタック作成を再試行するために次の手順を行います。

ソリューション: スタックオペレーションを再試行するために、次のステップを行います。

1. コンソールで、オペレーションが失敗したスタックを含む StackSet を選択します。
2. [アクション] メニューで [Edit StackSet details (StackSet の詳細を編集)] を選択して、スタックの作成または更新を再試行します。
3. [テンプレートの指定] ページで、同一の CloudFormation テンプレートを使用するために、デフォルトオプション [現在のテンプレートの使用] をそのまま使用します。テンプレートを変更せざるを得なかったことでスタックオペレーションに失敗したため、修正後のテンプレートをアップロードする場合は、[テンプレートを Amazon S3 にアップロード]、[参照] の順に選択して、アップデート後のテンプレートを選択します。修正後のテンプレートのアップロードが完了したら、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、テンプレート固有のパラメータを変更していない場合は、[次へ] を選択します。
5. [Set deployment options (デプロイオプションの設定)] ページで、必要に応じて [Maximum concurrent accounts (同時アカウントの最大数)] および [Failure tolerance (耐障害性)] のデフォルトを変更します。これらの設定の詳細については、「[StackSet オペレーションのオプション](#)」をご参照ください。

6. [確認] ページで選択内容を確認し、チェックボックスをオンにして必要な IAM 機能を確認します。[Submit] を選択してください。
7. スタックが正常に更新されたら、スタックの作成を妨げている根本的な問題が解決されるまで、この手順を繰り返します。

スタックインスタンスの削除に失敗する場合

問題: スタックの削除に失敗した。

原因: 削除保護が有効になっているスタックでは、スタックを削除できない。

解決策: スタックの削除保護が有効になっているかどうかを確認する。有効になっている場合は、削除保護を無効にし、スタックインスタンスの実行を再度行います。

スタックのインポートオペレーションが失敗する

問題: スタックのインポートオペレーションで、既存のスタックを新規または既存の StackSets にインポートできません。スタックインスタンスは INOPERABLE ステータスです。

解決策: 次のタスクを完了して、スタックのインポートオペレーションを元に戻します。

1. [StackSets からスタックを削除] オプションを使用し、設定中に [RetainStacks] を有効にしてから、続行して StackSet からスタックインスタンスを削除します。詳細については、「[CloudFormation StackSets からスタックを削除する](#)」を参照してください。
2. INOPERABLE スタックインスタンスを削除するために Stackset のスタックインスタンスが更新されていることがわかります。
3. インポート失敗エラーに従ってスタックインスタンスを修正し、スタックインポートオペレーションを再試行します。

StackSets オペレーションのスタックインスタンスの失敗回数

スタックインスタンスの失敗回数は、スタックインスタンスのプロビジョニングまたは更新に失敗した場合に警告を発します。これらのスタックインスタンスは、次のいずれかの理由でデプロイされませんでした。

- 類似した設定を持つ既存のリソース
- AWS Identity and Access Management (IAM) ロールなどの依存関係の欠落
- その他の相反要因

最大同時実行数でデプロイする場合、最大同時実行数は最大で失敗許容回数より 1 つ多くなります。例えば、失敗許容回数が 9 である場合、最大同時実行数は 10 を超えることはできません。これにより、一部のスタックインスタンスが更新に失敗した場合でも、オペレーションは SUCCEEDED を返します。新しいスタックインスタンスの失敗回数により、失敗許容回数がすべての失敗を許容するように設定されているため、オペレーションが条件付きでのみ成功したかどうかを判断できます。

AWS マネジメントコンソール、AWS SDK、または AWS CLI を使用して失敗回数を取得し、スタックインスタンスをフィルタリングして、再デプロイが必要なインスタンスを判断できます。

コンソールを使用する

失敗したスタックインスタンスの数を表示するには:

1. [CloudFormation コンソール](#)を開き、[StackSets] を選択します。
2. StackSet を選択し、[Operations] (オペレーション) タブを選択します。
3. [Status] (ステータス) 列でステータスを選択すると、ステータスの詳細が表示されます。特定のオペレーションで失敗したスタックインスタンスの数は、ステータスの詳細で確認できます。

オペレーションのスタックインスタンスのステータス、リージョン、アカウントを表示するには:

1. ステータスの詳細で、失敗したスタックインスタンスの回数を選択します。例: [Stack instances: **<number of failed stack instances>**] (スタックインスタンス:)。
2. パネルヘッダーを選択してサイドパネルを拡張します。サイドパネルの結果は、選択したオペレーションが完了した後のスタックインスタンスのステータスです。

オペレーションの現在のスタックインスタンスの詳細を表示するには:

1. [Stack Instances] (スタックインスタンス) タブを選択します。
2. [Last operation ID] (最終オペレーション ID) でフィルタリングします。結果には、インスタンスを変更した最後のオペレーションからの現在のステータスとステータス理由が表示されます。このフィルターを [AWS アカウント]、[AWS リージョン]、[詳細ステータス]、[ドリフトステータス] と組み合わせて使用すると、検索結果をさらに絞り込むことができます。

AWS CLI を使用する場合

失敗したスタックインスタンスの数を取得するには、describe-stack-set-operation または list-stack-set-operations を呼び出して、StatusDetails を参照してください。

```
aws cloudformation describe-stack-set-operation --stack-set-name ss1 \  
--operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60
```

```
{  
  "StackSetOperation": {  
    "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",  
    "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",  
    "Action": "CREATE",  
    "Status": "SUCCEEDED",  
    "OperationPreferences": {  
      "RegionOrder": [],  
      "FailureToleranceCount": 10,  
      "MaxConcurrentCount": 10  
    },  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",  
    "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",  
    "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",  
    "StatusDetails": {  
      "FailedStackInstancesCount": 3  
    }  
  }  
}
```

```
aws cloudformation list-stack-set-operations --stack-set-name ss1
```

```
{  
  "Summaries": [  
    {  
      "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",  
      "Action": "CREATE",  
      "Status": "SUCCEEDED",  
      "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",  
      "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",  
      "StatusDetails": {  
        "FailedStackInstancesCount": 3  
      },  
      "OperationPreferences": {  
        "RegionOrder": [],  
        "FailureToleranceCount": 10,  

```

```

        "MaxConcurrentCount": 10
    }
}
]
}

```

特定のオペレーションの履歴概要を把握するには、`list-stack-set-operation-results` を使用して、オペレーションが完了した後の各スタックインスタンスのステータスとステータスの理由を表示します。Status と StatusReason をを見つけるには、次の例を参照してください。

```

aws cloudformation list-stack-set-operation-results --stack-set-name ss1 \
  --operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60 --
  filters Name=OPERATION_RESULT_STATUS,Values=FAILED

```

```

{
  "Summaries": [
    {
      "Account": "123456789012",
      "Region": "us-west-2",
      "Status": "FAILED",
      "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'.",
      "AccountGateResult": {
        "Status": "SKIPPED",
        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'."
      },
      "OrganizationalUnitId": ""
    },
    {
      "Account": "123456789012",
      "Region": "us-west-1",
      "Status": "FAILED",
      "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'.",
      "AccountGateResult": {
        "Status": "SKIPPED",

```

```

        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'."
    },
    "OrganizationalUnitId": ""
},
{
    "Account": "123456789012",
    "Region": "us-east-1",
    "Status": "FAILED",
    "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'."
    "AccountGateResult": {
        "Status": "SKIPPED",
        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'."
    },
    "OrganizationalUnitId": ""
}
]
}

```

`list-stack-instances` を `DETAILED_STATUS` フィルターおよび `LAST_OPERATION_ID` フィルターと一緒に使用すると、スタックインスタンスをデプロイしようとした最後のオペレーションで失敗したスタックインスタンスのリストを取得できます。`DETAILED_STATUS` と `LAST_OPERATION_ID` を含む例の `--filters` フラグを参照してください。

```

aws cloudformation list-stack-instances --stack-set-name ss1 \
  --filters Name=DETAILED_STATUS,Values=FAILED Name=LAST_OPERATION_ID,Values=5550e62f-
c822-4331-88fa-21c1d7bafc60

```

```

{
  "Summaries": [
    {
      "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
      "Region": "us-east-1",
      "Account": "123456789012",
      "Status": "OUTDATED",

```

```

        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'.",
        "StackInstanceStatus": {
            "DetailedStatus": "FAILED"
        },
        "OrganizationalUnitId": "",
        "DriftStatus": "NOT_CHECKED",
        "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
    },
    {
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Region": "us-west-1",
        "Account": "123456789012",
        "Status": "OUTDATED",
        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'.",
        "StackInstanceStatus": {
            "DetailedStatus": "FAILED"
        },
        "OrganizationalUnitId": "",
        "DriftStatus": "NOT_CHECKED",
        "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
    },
    {
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Region": "us-west-2",
        "Account": "123456789012",
        "Status": "OUTDATED",
        "StatusReason": "Account 123456789012 should have
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
'AWSCloudFormationStackSetAdministrationRole'.",
        "StackInstanceStatus": {
            "DetailedStatus": "FAILED"
        },
        "OrganizationalUnitId": "",
        "DriftStatus": "NOT_CHECKED",
        "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
    }
}
]
}

```

スタックインスタンスを変更する最後のオペレーション ID をを見つけるには、`list-stack-instances` または `describe-stack-instance` を使用して `LastOperationId` を取得します。

```
aws cloudformation describe-stack-instance --stack-set-name ss1 \  
--stack-instance-account 123456789012 --stack-instance-region us-east-2
```

```
{  
  "StackInstance": {  
    "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",  
    "Region": "us-west-2",  
    "Account": "123456789012",  
    "ParameterOverrides": [],  
    "Status": "OUTDATED",  
    "StackInstanceStatus": {  
      "DetailedStatus": "FAILED"  
    },  
    "StatusReason": "Account 123456789012 should have  
'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role  
'AWSCloudFormationStackSetAdministrationRole'.",  
    "OrganizationalUnitId": "",  
    "DriftStatus": "NOT_CHECKED",  
    "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"  
  }  
}
```

Git 同期で Git リポジトリに保存されているソースコードとスタックを同期する

Git 同期を使用すると、ソースコントロールを使用して CloudFormation スタックを管理できます。これを行うには、Git リポジトリをモニタリングするように CloudFormation を設定します。リポジトリの次の 2 つのファイルへの変更に関して監視が行われます。

- スタックを定義する CloudFormation テンプレートファイル
- スタックを設定するパラメータを含むスタックデプロイファイル

Git 同期を使用すると、プルリクエストとバージョントラッキングを使用し、一元化された場所から CloudFormation スタックの設定、デプロイ、更新を行うことができます。テンプレートまたはデプロイファイルへの変更をコミットすると、CloudFormation はスタックを自動的に更新します。プルリクエストを使用する場合、CloudFormation は実際にスタックを更新する前に加えられる変更を説明するコメントをプルリクエストに残すことができます。ただし、最初にこの機能を有効にする必要があります。

Git 同期には、リポジトリへのリンク、スタックデプロイファイルの生成、CloudFormation テンプレートの更新、リポジトリへのプルリクエストの送信に使用できるコンソールインターフェイスが用意されています。Git 同期には、アクティブな Git 同期スタックのデプロイを監視、編集、トラブルシューティングできるステータスダッシュボードもあります。[スタックを作成](#)するとき、Git 同期には [CloudFormation コンソール](#) からアクセスします。Git 同期には CodeConnections を使用してアクセスすることもできます。詳細については、「Developer Tools Console ユーザーガイド」の「[リンクされたリポジトリの同期設定を操作する](#)」を参照してください。

Git 同期は [GitHub](#)、[GitHub Enterprise](#)、[GitLab](#)、[Bitbucket](#)、[GitLab セルフマネージド](#) のリポジトリをサポートしています。

Note

Git 同期を利用できるリージョンは、米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)、カナダ (中部)、アジアパシフィック (ムンバイ)、アジアパシフィック (東京)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、欧州 (アイルランド)、欧州 (ロンドン)、欧州 (パリ)、欧州 (ストックホルム)、欧州 (フランクフルト)、欧州 (ミラノ)、南米 (サンパウロ) です。

マルチアカウント戦略での Git 同期の使用については、ブログ記事「[Use CloudFormation Git sync to configure resources in customer accounts](#)」を参照してください。

トピック

- [Git 同期と CloudFormation の連携](#)
- [Git 同期を使用してスタックを Git リポジトリに同期するための前提条件](#)
- [Git 同期を使用してリポジトリソースコードからスタックを作成する](#)
- [CloudFormation がプルリクエストでスタック変更の概要を追加できるようにする](#)
- [Git 同期ステータスダッシュボード](#)

Git 同期と CloudFormation の連携

このトピックでは、Git 同期の仕組みについて説明し、それを使用するために必要な主要な概念を紹介します。

トピック

- [Git 同期の仕組み](#)
- [プルリクエストに関するコメント](#)
- [スタックデプロイファイル](#)
- [CloudFormation テンプレートファイル](#)
- [テンプレート定義リポジトリ](#)

Git 同期の仕組み

Git 同期を使用するには、まず [CodeConnections](#) サービスを使用して Git プロバイダーを CloudFormation に接続する必要があります。このガイドの手順では、接続は CodeConnections コンソールを通じて作成されます。または、AWS CLI を使用して接続を作成することもできます。次のいずれかの Git プロバイダーを使用できます。

- [GitHub](#)
- [GitHub エンタープライズ](#)
- [GitLab](#)
- [Bitbucket](#)

• [GitLab セルフマネージド](#)

次に、スタックを定義する CloudFormation テンプレートを作成し、リポジトリに追加します。このテンプレートファイルは監視されます。CloudFormation は、変更がコミットされると自動的にスタックを更新します。

CloudFormation コンソールで新しいスタックを作成し、[Git から同期] を選択して、Git 同期を使用するように CloudFormation に指示します。CloudFormation に監視させるリポジトリとブランチを指定し、スタックを定義するリポジトリ内の CloudFormation テンプレートを指定します。

設定時に、リポジトリから独自のスタックデプロイファイル指定するか、Git 同期に生成させることができます。スタックデプロイファイルには、スタック内のリソースを設定するパラメータと値が含まれています。このスタックデプロイファイルは監視されます。CloudFormation は、変更がコミットされると自動的にスタックを更新します。

Git 同期は、スタックを CloudFormation テンプレートファイルおよびスタックデプロイファイルと同期するためのプルリクエストをリポジトリに作成します。Git 同期でスタックデプロイファイルを生じた場合、そのファイルは Git 同期によってリポジトリに送信されます。

その後、プルリクエストをリポジトリにマージすることで、CloudFormation がスタックをプロビジョニングし、デプロイパラメータで設定し、リポジトリの変更のモニタリングを開始するようにします。

それ以降、テンプレートファイルまたはスタックデプロイファイルを変更してリポジトリにコミットするたびに、CloudFormation は自動的に変更を検出します。チームがプルリクエストを使用する場合、デプロイ前にチームメンバーは変更を確認して承認できます。プルリクエストが受け入れられると、CloudFormation は変更をデプロイします。

CloudFormation コンソールで、スタックの Git 同期設定のステータスをモニタリングし、スタックに適用されたコミットの履歴を確認できます。コンソールには、Git 同期の再設定や問題のトラブルシューティングを行うためのツールも用意されています。

プルリクエストに関するコメント

コンソールで [プルリクエストに対するコメントの有効化] オプションをオンにすることにより、CloudFormation が CodeConnections サービスを通じてプルリクエストのコード変更の概要を作成するように選択できます。プルリクエストの変更の概要を提供することは、プルリクエストをマージする前に、提案された変更の影響をチームメンバーが簡単に確認して理解できることを意味します。詳細については、「[CloudFormation がプルリクエストでスタック変更の概要を追加できるようにする](#)」を参照してください。

スタックデプロイファイル

スタックデプロイファイルは、JavaScript Object Notation (JSON) または YAML の標準フォーマットファイルであり、CloudFormation スタックを管理するパラメータと値を含みます。このファイルの変更は監視されます。ファイルへの変更がリポジトリにコミットされると、関連するスタックが自動的に更新されます。

スタックデプロイファイルには、キーと値のペアと 2 つのディクショナリが含まれます。

- `template-file-path`

これは CloudFormation テンプレートファイルの完全なリポジトリパスです。テンプレートファイルは、このデプロイファイルに関連付けられた CloudFormation スタックのリソースを宣言します。

- `parameters`

パラメータディクショナリには、スタック内のリソースを構成するキーと値のペアが含まれます。スタックデプロイファイルには、最大 50 個のパラメータを含めることができます。

- `tags`

タグディクショナリには、スタックにあるリソースの特定、および分類に使用できる任意のキーと値のペアが含まれます。スタックデプロイファイルには、最大 50 個のタグを含めることができます。

独自のスタックデプロイファイルを指定することも、Git 同期で作成し、プルリクエストを自動的にリポジトリに送信することもできます。スタックのデプロイファイルを編集し、変更をリポジトリにコミットすることで、パラメータとタグを管理できます。

Git 同期スタックデプロイファイルの例を次に示します。

```
template-file-path: fargate-srvs/my-stack-template.yaml

parameters:
  image: public.ecr.aws/lts/nginx:latest
  task_size: x-small
  max_capacity: 5
  port: 8080
  env: production
tags:
  cost-center: '123456'
```

```
org: 'AWS'
```

CloudFormation テンプレートファイル

テンプレートファイルには、CloudFormation スタックを構成する AWS リソースの宣言が含まれています。Git 同期では、テンプレートファイルは Git リポジトリに保存され、スタックデプロイファイルによって参照されます。テンプレートを編集し、変更をリポジトリにコミットすると、スタックを管理できます。

詳細については、「[CloudFormation テンプレートの使用](#)」を参照してください。

テンプレート定義リポジトリ

テンプレート定義リポジトリは、Git 同期を通じて CloudFormation にリンクされている Git リポジトリです。リポジトリは、CloudFormation テンプレートとスタックデプロイファイルへの変更について監視されます。ファイルへの変更がコミットされると、関連するスタックが自動的に更新されます。

Important

Git 同期コンソールでテンプレート定義リポジトリを設定する際、Git 接続から正しいリポジトリとブランチを選択します。Git 同期は、設定したリポジトリとブランチのみを監視して、CloudFormation テンプレートとスタックデプロイファイルへの変更を確認します。

Git 同期を使用してスタックを Git リポジトリに同期するための前提条件

CloudFormation スタックを Git リポジトリに同期する前に、次の要件が満たされていることを確認します:

トピック

- [Git リポジトリ](#)
- [CloudFormation テンプレート](#)
- [Git 同期サービスロール](#)
- [コンソールユーザーの IAM アクセス権限](#)

Git リポジトリ

Git リポジトリは、次のいずれかのプラットフォームでホストされている必要があります。

- [GitHub](#)
- [GitHub エンタープライズ](#)
- [GitLab](#)
- [Bitbucket](#)
- [GitLab セルフマネージド](#)

リポジトリは、パブリックでもプライベートでもかまいません。[接続コンソール](#)を使用して、Git リポジトリを CloudFormation に接続する必要があります。

CloudFormation テンプレート

Git リポジトリには、Git 同期で接続する対象のブランチにチェックインされた [CloudFormation テンプレートファイル](#)が含まれている必要があります。このテンプレートは[スタックデプロイファイル](#)によって参照されます。

Git 同期サービスロール

Git 同期には IAM ロールが必要です。Git 同期の設定時にスタック用の IAM ロールを作成するか、既存のロールを使用するかを選択できます。

Note

自動的に生成された IAM ロールは、そのロールが生成されたスタックにのみ権限を適用します。自動生成された IAM ロールを再利用するには、新しいスタックのロールを編集する必要があります。

Git 同期サービスロールに必要な権限

Git 同期に指定する IAM ロールには、以下の権限が必要です。

- `cloudformation:CreateChangeSet`
- `cloudformation>DeleteChangeSet`

- `cloudformation:DescribeChangeSet`
- `cloudformation:DescribeStackEvents`
- `cloudformation:DescribeStacks`
- `cloudformation:ExecuteChangeSet`
- `cloudformation:ListChangeSets`
- `cloudformation:ValidateTemplate`
- `events:PutRule`
- `events:PutTargets`

 Note

上記の必要な権限は、Git 同期が生成する IAM ロールに自動的に追加されます。

次の IAM ロールの例には、Git 同期の前提条件となる権限が含まれています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SyncToCloudFormation",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateChangeSet",
        "cloudformation>DeleteChangeSet",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:GetTemplate",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStacks",
        "cloudformation:ValidateTemplate"
      ],
      "Resource": "*"
    },
  ],
}
```

```
{
  "Sid": "PolicyForManagedRules",
  "Effect": "Allow",
  "Action": [
    "events:PutRule",
    "events:PutTargets"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "events:ManagedBy":
["cloudformation.sync.codeconnections.amazonaws.com"]
    }
  },
  {
    "Sid": "PolicyForDescribingRule",
    "Effect": "Allow",
    "Action": "events:DescribeRule",
    "Resource": "*"
  }
]
```

信頼ポリシー

信頼関係を定義するロールを作成する際、次の信頼ポリシーを指定する必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
}
```

Confused Deputy Problem (混乱した使節の問題) から自分を守るために、aws:SourceArn および aws:SourceAccount の条件キーを使用することをお勧めします。ソースアカウントはアカウント ID で、ソース ARN は、CloudFormation が Git リポジトリに接続できるようにする [CodeConnections](#) サービス内の接続の ARN です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:codeconnections:us-east-1:123456789012:connection/EXAMPLE64-8aad-4d5d-8878-dfcab0bc441f"
        }
      }
    }
  ]
}
```

混乱した代理に関する問題の詳細については、「[サービス間の混乱した代理の防止](#)」を参照してください。

コンソールユーザーの IAM アクセス権限

CloudFormation コンソールを使用して Git 同期を正常にセットアップするには、エンドユーザーにも IAM を通じてアクセス許可を付与する必要があります。

Git リポジトリへの接続を作成および管理するには、次の codeconnections アクセス許可が必要です。

- codeconnections:CreateRepositoryLink
- codeconnections:CreateSyncConfiguration
- codeconnections>DeleteRepositoryLink
- codeconnections>DeleteSyncConfiguration
- codeconnections:GetRepositoryLink
- codeconnections:GetSyncConfiguration
- codeconnections:ListRepositoryLinks
- codeconnections:ListSyncConfigurations
- codeconnections:ListTagsForResource
- codeconnections:TagResource
- codeconnections:UntagResource
- codeconnections:UpdateRepositoryLink
- codeconnections:UpdateSyncBlocker
- codeconnections:UpdateSyncConfiguration
- codeconnections:UseConnection

コンソールユーザーには、Git 同期セットアッププロセス中にスタックを表示および管理するための次の cloudformation アクセス許可も必要です。

- cloudformation:CreateChangeSet
- cloudformation>DeleteChangeSet
- cloudformation:DescribeChangeSet
- cloudformation:DescribeStackEvents
- cloudformation:DescribeStacks
- cloudformation:ExecuteChangeSet
- cloudformation:GetTemplate
- cloudformation:ListChangeSets
- cloudformation:ListStacks
- cloudformation:ValidateTemplate

 Note

コンソールでのみ使用では、変更セットのアクセス許可 (cloudformation:CreateChangeSet、cloudformation>DeleteChangeSet、cloudformation:DeleteStack、cloudformation:DescribeStacks、cloudformation:ListStackResources、cloudformation:ListStacks、cloudformation:RollbackStack、cloudformation:SetStackPolicy、cloudformation:UpdateStack) は厳密に必要な場合がありますが、フルスタック検査および管理機能を有効にするために推奨されます。

次の IAM ポリシーの例には、コンソールを使用して Git 同期を設定するために必要なユーザーアクセス許可が含まれています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeConnectionsPermissions",
      "Effect": "Allow",
      "Action": [
        "codeconnections:CreateRepositoryLink",
        "codeconnections:CreateSyncConfiguration",
        "codeconnections>DeleteRepositoryLink",
        "codeconnections>DeleteSyncConfiguration",
        "codeconnections:GetRepositoryLink",
        "codeconnections:GetSyncConfiguration",
        "codeconnections:ListRepositoryLinks",
        "codeconnections:ListSyncConfigurations",
        "codeconnections:ListTagsForResource",
        "codeconnections:TagResource",
        "codeconnections:UntagResource",
        "codeconnections:UpdateRepositoryLink",
        "codeconnections:UpdateSyncBlocker",
        "codeconnections:UpdateSyncConfiguration",
        "codeconnections:UseConnection",
        "codeconnections:CreateForcedTargetSync",
        "codeconnections:CreatePullRequestForResource"
      ],
      "Resource": "*"
    }
  ],
  {
```

```
        "Sid": "CloudFormationConsolePermissions",
        "Effect": "Allow",
        "Action": [
            "cloudformation:CreateChangeSet",
            "cloudformation:DeleteChangeSet",
            "cloudformation:DescribeChangeSet",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStacks",
            "cloudformation:ExecuteChangeSet",
            "cloudformation:GetTemplate",
            "cloudformation:ListChangeSets",
            "cloudformation:ListStacks",
            "cloudformation:ValidateTemplate"
        ],
        "Resource": "*"
    }
}
```

Note

アクセス許可 `codeconnections:CreateForcedTargetSync` と `codeconnections:CreatePullRequestForResource` を含む IAM ポリシーを作成すると、これらのアクションが存在しないことを示す警告が IAM コンソールに表示されることがあります。この警告を無視しても、ポリシーは正常に作成されます。これらのアクセス許可は、IAM コンソールでは認識されないものの、特定の Git 同期オペレーションには必要です。

Git 同期を使用してリポジトリソースコードからスタックを作成する

このトピックでは、Git 同期を使用して Git リポジトリに同期する CloudFormation スタックを作成する方法について説明します。

Important

先に進む前に、前のセクションのすべての[前提条件](#)を満たしてください。

リポジトリソースコードからスタックを作成する

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックを作成する AWS リージョンを選択します。
3. [スタック] ページで、[スタックの作成]、[新しいリソースを使用 (標準)] の順に選択します。
4. [Create stack] (スタックの作成) ページで、次の手順を実行します。
 - a. [前提条件 - テンプレートを準備] で、[既存のテンプレートを選択] を選択したままにします。
 - b. [テンプレートを指定] で、[Git から同期] を選択し、[次へ] を選択します。
5. [スタックの詳細を指定] ページの [スタック名] で、スタックの名前を入力します。スタック名には、文字 (A~Z および a~z)、数字 (0~9)、およびダッシュ (-) を使用できます。
6. [スタックデプロイファイル] の [デプロイファイルの作成] で、次のように操作します。
 - スタックデプロイファイルを作成してリポジトリに追加していない場合は、[次のパラメータを使用してファイルを作成し、リポジトリに配置する] を選択します。
 - リポジトリにスタックデプロイファイルがある場合は、[リポジトリに独自のファイルを提供しています] を選択します。
7. [テンプレート定義リポジトリ] で、既に CloudFormation にリンクされている Git リポジトリを選択する場合は、[リンクされた Git リポジトリを選択する] を、新しいリポジトリをリンクする場合は、[Git リポジトリをリンクする] を選択します。[Git リポジトリをリンクする] を選択した場合は、次の操作を行います。
 - a. [レポジトリプロバイダーを選択] で、次のいずれかを選択します。
 - GitHub
 - GitHub Enterprise Server
 - GitLab
 - Bitbucket
 - GitLab セルフマネージド
 - b. [接続] でリストから接続を選択します。[接続] リストにオプションが表示されない場合は、[新しい接続を追加] を選択して、[接続コンソール](#)に移動し、リポジトリへの接続を作成します。

8. [リポジトリ] では、スタックテンプレートファイルが含まれている Git リポジトリを選択します。
9. [ブランチ] リストで、Git 同期に監視させるブランチを選択します。

 Note

Git 同期は、CloudFormation テンプレートとスタックデプロイファイルへの変更について、選択したブランチのみを監視します。スタックに適用する変更はすべてこのブランチにコミットする必要があります。

10. デプロイファイルパスには、リポジトリブランチのルートにあるスタックデプロイファイル名を含むフルパスを指定します。

CloudFormation がファイルを生成している場合、ここでファイルがリポジトリにコミットされます。ご自身でファイルを設置する場合、これがリポジトリ内のファイルの場所です。

11. IAM ロールを追加します。IAM ロールには、CloudFormation が Git リポジトリからスタックを同期するために必要な権限が含まれています。新しいロールを生成するには [新しい IAM ロール] を選択し、AWS アカウント から既存のロールを選択するには [既存の IAM ロール] を選択できます。新しいロールの生成を選択すると、必要な権限がロールに含まれます。

12. プルリクエストに対するコメントを有効または無効にします。

- CloudFormation がスタック更新のプルリクエストで変更セット情報を追加するようにするには、[プルリクエストに対するコメントを有効にする] トグルをオンにしたままにします。
- このトグルをオフにすると、CloudFormation は、リポジトリファイルの更新時に現在のスタック設定とプルリクエストで提案された変更の違いを記述しません。

13. テンプレートファイルパスには、リポジトリのルートからのスタックテンプレートファイルのフルパスを指定します。

14. (オプション) スタックパラメータを指定するには、[パラメータを追加] を選択し、各パラメータにキーと値を指定してから、[次へ] を選択します。詳細については、「[スタックデプロイファイル](#)」を参照してください。

例えば、スタックデプロイファイルで **port=8080** パラメータを指定するには、次を実行します。

- a. [Add parameter] (パラメータの追加) を選択します。
- b. [キー] に「**port**」と入力します。
- c. [Value (値)] に「**8080**」と入力します。

15. (オプション) スタックタグを指定するには、[新しいタグを追加] を選択し、各タグのタグキーと値を指定して、[次へ] を選択します。詳細については、「[スタックデプロイファイル](#)」を参照してください。
16. [次へ] を選択して [スタックオプションの設定] に進みます。スタックオプションの設定については、「[スタックオプションを設定する](#)」を参照してください。

スタックの設定が完了したら、[次へ] を選択してレビューとデプロイを続行します。

17. スタック設定を確認し、以下を確認します。
 - スタックテンプレートは正しく設定され、[Git から同期] に設定されています。
 - デプロイファイルは正しく設定されています。
 - テンプレート定義リポジトリは正しく設定されており、具体的には、正しいリポジトリとブランチ名が選択されています。
 - デプロイファイルのプレビューは正しく、期待通りのパラメータと値が含まれています。
18. [送信] を選択してスタックを作成します。

[送信] を選択すると、Git リポジトリにプルリクエストが自動的に作成されます。スタックを作成するには、このプルリクエストを Git リポジトリにマージする必要があります。スタックが作成されると、CloudFormation は Git リポジトリの変更をモニタリングします。

Git リポジトリからスタックを更新する

スタックを更新するには、Git リポジトリのテンプレートファイルまたはスタックデプロイファイルに直接変更を加えます。モニタリング対象のブランチに変更をコミットすると、CloudFormation は自動的にスタックを更新します。プルリクエストを使用する場合、スタックが更新される前に Git リポジトリにプルリクエストが自動的に作成されます。スタックを更新するには、このプルリクエストを Git リポジトリにマージする必要があります。

CloudFormation コンソールで、スタックを選択し、[Git 同期] タブを選択すると、スタックと同期イベントのステータスに関する情報が表示されます。詳細については、「[Git 同期ステータスダッシュボード](#)」を参照してください。

CloudFormation がプルリクエストでスタック変更の概要を追加できるようにする

このトピックでは、CloudFormation が Git リポジトリのプルリクエストでスタックの変更の概要を追加できるようにする方法について説明します。

プルリクエストに対するコメント機能を有効にすると、CloudFormation は、現在のスタック設定と、リポジトリファイルの更新時に提案された変更の違いを説明するコメントを追加できるようになります。このコメントは、追加、変更、または削除されるリソースの概要を提供します。これを利用することで、プルリクエストをマージする前に徹底したコードレビューを実行できます。

新しいスタックのプルリクエストに対するコメントを有効にするには (コンソール)

スタックを作成する際に、[スタックの詳細を指定] ページの [テンプレート定義リポジトリ] で、[プルリクエストに対するコメントを有効にする] トグルがオンになっているようにします。これは、新しいスタックのデフォルトの設定です。

既存のスタックのプルリクエストに対するコメントを有効にするには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面の上部のナビゲーションバーで、スタックを作成した AWS リージョン を選択します。
3. [スタック] ページで、更新する実行中のスタックを選択します。
4. [Git 同期] タブを選択してから、[編集] をクリックします。
5. [Git 同期設定を編集] ページの [テンプレート定義リポジトリ] で、[プルリクエストに対するコメントを有効にする] トグルをオンに切り替えます。
6. [Update configuration (設定の更新)] を選択します。

Git 同期ステータスダッシュボード

AWS CloudFormation Git 同期デプロイのステータスを表示するには、CloudFormation コンソールでスタックを選択してから、[Git 同期] タブを選択します。

Git 同期タブは、[Git 同期ステータス] と [最新の同期イベント] の 2 つのパネルに分かれています。

Git 同期のステータス

上部パネルには、スタックの Git 同期設定に関する以下の情報が表示されます。

リポジトリ

Git 同期に接続されているリポジトリへのリンク

リポジトリプロバイダー

リポジトリプロバイダーの名前

ブランチ

Git 同期が監視しているブランチの名前

デプロイファイルパス

このスタックのスタックデプロイファイルへのフルパス

リポジトリの同期ステータス

直近の同期オペレーションのステータス

リポジトリ同期ステータスメッセージ

直近の同期オペレーションのメッセージ

Git 同期のステータス

このスタックの Git 同期のステータス

プロビジョニングのステータス

プロビジョニング操作のステータス

パネルの右上にある次のボタンを使用して Git 同期を変更または更新します。

- 編集-Git 同期設定を編集します。
- 最新のコミットを再試行-リポジトリへの最新のコミットに従ってスタックを更新します。
- 切断-Git 同期をスタックから切断します。
- 更新-Git 同期ステータスパネルを更新します。

最新の同期イベント

[最新の同期イベント] パネルには、スタックに適用されたコミットの表が表示されます。

各列のヘッダーにある矢印を使用してテーブルをソートできます。テーブルは、以下によって、昇順または降順にソートできます。

- 日付
- コミット ID
- イベント
- 日付
- イベントタイプ

サポートされるスタック状態

Git 同期をスタックに設定できるのは、スタックが以下のサポートされている状態のいずれかにある場合のみです。

- CREATE_COMPLETE
- UPDATE_COMPLETE
- UPDATE_ROLLBACK_COMPLETE
- IMPORT_COMPLETE
- IMPORT_ROLLBACK_COMPLETE

次の表で、スタックステータスコードの完全なリストと説明を示します。

スタックステータスと、オプションの詳細ステータス	説明
CREATE_COMPLETE	1 つ以上のスタックを正常に作成。
CREATE_IN_PROGRESS	1 つ以上のスタックを作成中。
CREATE_FAILED	1 つ以上のスタックの作成に失敗。関連するエラーメッセージを確認するには、スタックのイベントを調べます。作成の失敗の理由としては、スタックのすべてのリソースを操作するためのアクセス許可の不足、AWS サービスによるパラメーター値の拒否、リソース作成中のタイムアウトなどが考えられます。
DELETE_COMPLETE	1 つ以上のスタックを正常に削除。削除されたスタックは、90 日間は保持され、表示できます。

スタックステータスと、オプションの詳細ステータス	説明
DELETE_FAILED	1 つ以上のスタックの削除に失敗。削除が失敗したので、まだ実行されているリソースがいくつかある可能性があります。ただし、スタックを操作したり、更新したりすることはできません。スタックを再度削除するか、スタックのイベントを調べて関連するエラーメッセージを確認します。
DELETE_IN_PROGRESS	1 つ以上のスタックを削除中。
REVIEW_IN_PROGRESS	進行中の、予測される StackId を持つがテンプレートまたはリソースのない 1 つ以上のスタックの作成。  Important このステータスコードのスタックは、スタックの可能な限りの最大数に対してカウントされます。
ROLLBACK_COMPLETE	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックを正常に削除。スタックは以前の動作状態に戻ります。スタック作成オペレーションの実行中に作成されたすべてのリソースが削除されます。 このステータスが存在するのは、スタックの作成が失敗した後のみです。部分的に作成されたスタックからすべてのオペレーションが適切にクリーンアップされたことを示します。この状態では、削除オペレーションのみを実行できます。
ROLLBACK_FAILED	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックの削除に失敗。スタックを削除するか、スタックのイベントを調べて関連するエラーメッセージを確認します。
ROLLBACK_IN_PROGRESS	スタックの作成の失敗後、またはスタックの作成の明示的なキャンセル後に、1 つ以上のスタックを削除中。
UPDATE_COMPLETE	1 つ以上のスタックを正常に更新。

スタックステータスと、オプションの詳細ステータス	説明
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	スタックの正常な更新後に、1 つ以上のスタックの古いリソースを削除中。リソースの置き換えが必要なスタック更新の場合、CloudFormation はまず新しいリソースを作成し、次に古いリソースを削除して、スタックの中断を減らすことに役立ちます。この状態では、スタックは更新済みで使用可能ですが、CloudFormation はまだ古いリソースを削除中です。
UPDATE_FAILED	1 つ以上のスタックの失敗した更新。関連するエラーメッセージを確認するには、スタックのイベントを調べます。
UPDATE_IN_PROGRESS	1 つ以上のスタックを更新中。
UPDATE_ROLLBACK_COMPLETE	スタックの更新の失敗後、1 つ以上のスタックを前の動作状態に正常に戻しました。
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	スタックの更新の失敗後、1 つ以上のスタックの新しいリソースを削除中。この状態では、スタックが前の動作状態にロールバックされて使用可能ですが、CloudFormation がスタックの更新中に作成した新しいリソースをまだ削除中です。
UPDATE_ROLLBACK_FAILED	スタックの更新の失敗後、1 つ以上のスタックを前の動作ステートに戻すことに失敗。この状態の場合は、スタックを削除するか、 ロールバックを続行する ことができます。スタックが動作状態に戻るには、エラーを修正する必要がある場合があります。または、サポート に連絡してスタックを使用可能な状態に復元することができます。
UPDATE_ROLLBACK_IN_PROGRESS	スタックの更新の失敗後、1 つ以上のスタックを前の動作状態に戻しています。
IMPORT_IN_PROGRESS	インポートオペレーションは現在進行中です。
IMPORT_COMPLETE	resource import をサポートするスタック内のすべてのリソースについて、インポート操作が正常に完了しました。

スタックステータスと、オプションの詳細ステータス	説明
IMPORT_ROLLBACK_IN_PROGRESS	インポートは、以前のテンプレート構成にロールバックされます。
IMPORT_ROLLBACK_FAILED	スタック内の少なくとも 1 つのリソースに対するインポートのロールバックオペレーションが失敗しました。正常にインポートされたリソース CloudFormation の結果が利用可能になります。
IMPORT_ROLLBACK_COMPLETE	インポートは、以前のテンプレート構成に正常にロールバックされました。

CloudFormation レジストリによる拡張機能の管理

CloudFormation レジストリは、AWS アカウントの CloudFormation テンプレートに統合できる拡張機能を管理するための一元的なハブとして機能します。拡張機能には、AWS およびサードパーティーのパブリッシャーによるリソースタイプ、モジュール、フックの他に、独自のカスタム拡張機能が含まれます。レジストリを使用すると、AWS が提供するリソースを使用するのと同じように、CloudFormation テンプレートの拡張を簡単に検出し、プロビジョニングできます。

このセクションでは、CloudFormation レジストリを使用し、次の内容を含むアカウントのサードパーティ拡張機能を有効化する方法について説明します。

- パブリック拡張の有効化
- プライベート拡張機能の登録とアクティブ化

トピック

- [関連ドキュメント](#)
- [CloudFormation レジストリの概念](#)
- [CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)
- [CloudFormation レジストリからサードパーティーのパブリック拡張を使用する](#)
- [共有されているサードパーティーのプライベート拡張機能を使用する](#)
- [アカウントの拡張機能の構成データを編集する](#)
- [AWS Config でリソースタイプを記録する](#)

関連ドキュメント

独自の拡張機能を作成することにご関心がある開発者の方は、次のドキュメントを参照してください。

- 「CloudFormation コマンドラインインターフェイスのユーザーガイド」の「[CloudFormation CLI を使用してモジュールの開発](#)」
- 「CloudFormation コマンドラインインターフェイスのユーザーガイド」の「[CloudFormation CLI を使用してリソースタイプの作成](#)」
- 「CloudFormation フックユーザーガイド」の「[Developing custom Hooks using the CloudFormation CLI](#)」

さらに、CloudFormation レジストリで利用できる、プロビジョニング可能なすべての AWS リソースタイプは、標準の JSON スキーマで定義された属性およびプロパティとともに AWS クラウドコントロール API で使用することができます。詳細については、「[Cloud Control API ユーザーガイド](#)」を参照してください。Cloud Control API を使用して AWS リソースに対して CRUDL (作成、読み取り、更新、削除、一覧表示) オペレーションを実行するときは、自分の AWS アカウント 内の AWS リソースに対してのみ実行できます。

CloudFormation レジストリの概念

このトピックでは、CloudFormation レジストリの理解および使用に役立つ主な概念について説明します。

拡張タイプ

CloudFormation レジストリには次の拡張タイプが用意されています。

フック

フックは、CloudFormation によってスタックまたは特定のリソースが作成、更新、または削除される前に、それらを検査する検証チェックです。さらに、フックは変更セットの作成オペレーション中に呼び出すことができます。特定の要件に対してリソース設定を検証することにより、組織の基準およびベストプラクティスを実施するメカニズムを提供します。フックがフックロジックに準拠していない設定を検出した場合、警告を配信するか、プロビジョニングプロセスに失敗させて非準拠のリソースのデプロイを防止することができます。詳細については、「[CloudFormation Hooks ユーザーガイド](#)」を参照してください。

CloudFormation コンソールを使用して Hooks を設定する方法に関するドキュメントについては、「CloudFormation Hooks User Guide」にある次のセクションを参照してください。

- [AWS Control Tower proactive controls as Hooks](#)
- [Guard フック](#)
- [Lambda フック](#)

モジュール

モジュールは、複数の CloudFormation スタックテンプレート間に含めることができる再利用可能なリソース設定です。複雑な、または頻繁に使用されるリソース設定を再利用可能なコンポーネントにカプセル化することにより、CloudFormation テンプレートの作成およびメンテナンスを簡素化するように設計されています。組織のインフラストラクチャのデプロイ全体で一貫性および標準化が促進されます。

リソースタイプ:

リソースタイプは、CloudFormation でネイティブにサポートされていないサードパーティーリソースまたはカスタムリソースをモデル化および自動化できるようにします。リソースタイプを開発することによって CloudFormation の機能を拡張し、さまざまなサードパーティーのサービスからリソースをプロビジョニングおよび管理できます。

パブリック拡張

パブリック拡張機能は、すべての AWS アカウント が使用できるようにレジストリに公開されている CloudFormation の拡張機能です。具体的には次のとおりです。

- **AWS 拡張機能** – AWS によって公開された拡張機能は誰でも利用でき、デフォルトでアクティブ化されるため、アカウントで使用する前に何らかの操作を行う必要はありません。これらの拡張機能のバージョンは AWS によって制御されるため、利用可能な最新のバージョンを常に使用できます。
- **サードパーティ拡張機能** – AWS 以外のパブリッシャーによって一般使用向けに提供されている拡張機能です。サードパーティのパブリック拡張機能を使用するには、まずご自身のアカウントおよびリージョンでアクティブ化する必要があります。

アクティブ化した拡張機能

特定の AWS アカウント の CloudFormation レジストリには、3 種類のアクティブ化された拡張機能が含まれています。

- **AWS 拡張機能** – すべての AWS パブリック拡張機能が自動的にアクティブ化されます。
- **アクティブ化されたサードパーティ** – アカウントおよびリージョンに対してアクティブ化されたサードパーティ拡張機能のローカルコピーです。サードパーティのパブリック拡張機能をアクティブ化すると、CloudFormation はローカルコピーをアカウントのレジストリに作成します。詳細については、「[CloudFormation レジストリからサードパーティーのパブリック拡張を使用する](#)」を参照してください。
- **非公開登録** – パブリック CloudFormation レジストリにリストされていない非公開の拡張機能などです。自分で作成した拡張機能、または組織やその他サードパーティから共有された拡張機能などがこれに該当します。アカウントでこのようなプライベート拡張を使用するには、必ず先に登録しておく必要があります。拡張を登録すると、そのコピーがアカウントの CloudFormation レジストリにアップロードされ、アクティブ化されます。詳細については、「[共有されているサードパーティーのプライベート拡張機能を使用する](#)」を参照してください。

アカウントのサードパーティパブリッシャーから非公開で登録された拡張機能と、アクティブ化されたパブリック拡張機能を使用することは、サンドボックス環境でそれらを使用することに似ています。拡張機能はバージョン管理で管理されるため、プロビジョニングの動作は特定のバージョンに関連付けられます。その結果、これらの拡張機能はパブリック拡張機能と同じように機能し、バージョン固有の同じルールに従います。

Note

非公開で登録された拡張機能、およびアクティブ化されたサードパーティのパブリック拡張機能は、作成、読み取り、更新、一覧表示、削除のオペレーション中に実行される、イベントハンドラーを実装している場合があります。CloudFormation スタックでこれらの拡張機能を使用すると、作成されたリソースに対する料金のほかに、アカウントに対して料金が発生する可能性があります。詳細については、「[CloudFormation 料金表](#)」を参照してください。

CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する

CloudFormation レジストリで使用可能でアクティブ化された拡張を表示するには、AWS マネジメントコンソール または AWS CLI を使用します。

利用可能でアクティブ化された拡張を表示するには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. ナビゲーションペインの [Registry (レジストリ)] で、表示する拡張カテゴリを選択します。
 - [Public extensions] (パブリック拡張) は、アカウントで利用可能なパブリック拡張を表示します。
 1. フィルター、拡張タイプで、拡張タイプ (リソースタイプ、モジュール、またはフック) を選択します。
 2. フィルター、パブリッシャーでは、パブリッシャー (AWS またはサードパーティー) を選択します。
 - [Activated extensions] (アクティブ化された拡張) は、アカウントでアクティブ化されたパブリック拡張とプライベート拡張を表示します。

1. 拡張のタイプを選択してください: [Resource types] (リソースタイプ)、[Modules] (モジュール)、または [Hooks] (フック)。
2. [フィルター] ドロップダウンメニューを使用して、表示する拡張をさらに選択します。
 - AWS - AWS によって公開された拡張を一覧表示します。AWS によって発行されている拡張がデフォルトでアクティブ化されています。
 - [Third-party] (サードパーティー) - このアカウントでアクティブ化した AWS 以外の発行元からの公開拡張を一覧表示します。
 - [Registered] (登録済み) - このアカウントで有効にしたプライベート拡張を一覧表示します。
 - [Publisher] (パブリッシャー) - このアカウントを使用して公開したパブリック拡張を表示します。
4. 拡張名を検索または選択して、拡張の詳細を表示します。

利用可能でアクティブ化された拡張を表示するには (AWS CLI)

[list-types](#) コマンドを使用します。

CloudFormation レジストリからサードパーティーのパブリック拡張を使用する

テンプレートでサードパーティーのパブリック拡張を使用するには、まず、使用するアカウントとリージョンに対して拡張を有効にする必要があります。拡張を有効にすると、それが有効になっているアカウントとリージョンのスタックオペレーションで使用できるようになります。

サードパーティーのパブリック拡張を有効にすると、CloudFormation はアカウントの拡張レジストリに、有効化された拡張機能のエントリをプライベート拡張として作成します。これにより、拡張機能に含まれる設定プロパティを設定できます。設定プロパティは、拡張を特定の AWS アカウントとリージョンに対して設定する方法を定義します。

設定プロパティの設定に加えて、以下の方法で拡張をカスタマイズすることもできます。

- 拡張のログ設定に加えて、CloudFormation が拡張を有効にするために使用する実行ロールを指定します。
- 新しいマイナーバージョンまたはパッチバージョンが利用可能になったときに、拡張を自動的に更新するかどうかを指定します。

- サードパーティーのパブリック拡張名ではなく、使用するエイリアスを指定します。これは、サードパーティーの拡張間の名前の衝突を避けるのに役立ちます。

トピック

- [IAM アクセス許可とパブリック拡張アクセスの信頼ポリシーを使用して実行ロールを設定する](#)
- [拡張の新しいバージョンを自動的に使用する](#)
- [拡張を参照するエイリアスを使用する](#)
- [パブリック拡張を操作するためによく使用される AWS CLI コマンド](#)
- [アカウントでサードパーティーのパブリック拡張をアクティブ化する](#)
- [アカウントでパブリックサードパーティー拡張を更新する](#)
- [アカウントでサードパーティーのパブリック拡張を無効にする](#)

IAM アクセス許可とパブリック拡張アクセスの信頼ポリシーを使用して実行ロールを設定する

CloudFormation レジストリからパブリック拡張をアクティブ化すると、AWS アカウント およびリージョンでその拡張を呼び出すために必要なアクセス許可を CloudFormation に付与する実行ロールを提供できます。

実行ロールに必要なアクセス許可は、拡張スキーマのハンドラーセクションで定義されています。拡張に必要な特定のアクセス許可を付与し、実行ロールにアタッチする IAM ポリシーを作成する必要があります。

アクセス許可ポリシーに加えて、実行ロールには、CloudFormation がロールを引き受けることを許可する信頼ポリシーも必要です。「IAM ユーザーガイド」の「[カスタム信頼ポリシーを使用してロールを作成する](#)」のガイダンスに従って、カスタム信頼ポリシーを使用してロールを作成します。

信頼関係

使用する信頼ポリシーの例を下記に示します。

1 つ以上のグローバル条件コンテキストキーと Condition フィールドを使用して、必要に応じてサービス間での混乱した代理の防止のためのアクセス許可の範囲を制限します。詳細については、「[サービス間の混乱した代理の防止](#)」を参照してください。

- `aws:SourceAccount` の値をアカウント ID に設定します。

- `aws:SourceArn` の値を拡張機能の ARN に設定します。

信頼ポリシーの例 1

リソースタイプ拡張の IAM ロールの信頼ポリシーの例を次に示します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "resources.cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/Organization-Service-Resource"
        }
      }
    }
  ]
}
```

信頼ポリシーの例 2

フック拡張に関する IAM ロールの信頼ポリシーの例を次に示します。

JSON

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

```
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "resources.cloudformation.amazonaws.com",
        "hooks.cloudformation.amazonaws.com"
      ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/Organization-Service-Hook"
      }
    }
  }
}
```

拡張の新しいバージョンを自動的に使用する

拡張を有効にするときに、最新のマイナーバージョンを使用する拡張の種類を指定することもできます。パブリッシャーが有効化された拡張で新しいバージョンをリリースするたびに、拡張の種類によってマイナーバージョンが更新されます。

例えば、次にその拡張を含むテンプレートを使用してスタックの作成、更新などのスタックオペレーションを実行すると、CloudFormation は新しいマイナーバージョンを使用します。

自動または手動で新しい拡張バージョンに更新しても、スタックですでにプロビジョニングされている拡張インスタンスには影響しません。

CloudFormation は、拡張のメジャーバージョン更新を重大な変更を含む可能性があるものとして扱うため、拡張の新しいメジャーバージョンに手動で更新する必要があります。

AWS で公開された拡張は、利用可能なすべてのアカウントとリージョンでデフォルトで有効化され、常に各 AWS リージョン で利用可能な最新バージョンを使用します。

⚠ Important

拡張を最新バージョンに更新するかどうか、いつ更新するかはアカウントで制御できるため、異なるアカウントやリージョンに同じ拡張の異なるバージョンがデプロイされる可能性があります。

これにより、その拡張を含む同じテンプレートをそれらのアカウントとリージョンをまたいで使用すると、予期しない結果につながる可能性があります。

拡張を参照するエイリアスを使用する

特定の AWS アカウント とリージョンにおいて、特定の名前で複数の拡張を有効にすることはできません。異なるパブリッシャーが同じ拡張名でパブリック拡張を提供している場合があるため、CloudFormation では、有効にするサードパーティーのパブリック拡張のエイリアスを指定できません。

拡張にエイリアスを指定すると、CloudFormation はそのエイリアスをこのアカウントとリージョン内の拡張タイプ名として扱います。テンプレート、API 呼び出し、および CloudFormation コンソールで拡張を参照するには、エイリアスを使用する必要があります。

拡張のエイリアスは、特定のアカウントとリージョン内で一意である必要があります。異なるタイプ名のエイリアスを使用して、同じアカウントとリージョン内で同じパブリックリソースを複数回有効にできます。

⚠ Important

拡張のエイリアスが一意である必要があるのは、特定のアカウントとリージョン内のみのことではありますが、アカウントとリージョンをまたいで異なるサードパーティーのパブリック拡張に同じエイリアスを割り当ててはしないことを強くお勧めします。複数のアカウントまたはリージョンにまたがる拡張エイリアスを含むテンプレートを使用すると、予期しない結果が生じる可能性があります。

パブリック拡張を操作するためによく使用される AWS CLI コマンド

パブリック拡張を操作するために一般的に使用されるコマンドは次のとおりです。

- [activate-type](#) は、アカウント内のパブリックサードパーティのモジュールまたはリソースタイプを有効にします。

- アカウント内の拡張機能の設定データを指定し、フックを無効または有効にする [set-type-configuration](#)。
- [list-types](#) は、アカウントの拡張機能を一覧表示します。
- [describe-type](#) は、現在の設定データなど、特定の拡張機能または特定の拡張機能バージョンに関する詳細情報を返します。
- [set-type-default-version](#) は、どのバージョンの拡張機能がデフォルトバージョンであるかを指定します。
- [deactivate-type](#) は、アカウントで以前に有効化されたパブリックサードパーティーモジュールまたはリソースタイプを無効にします。

アカウントでサードパーティーのパブリック拡張をアクティブ化する

次のトピックでは、アカウントでサードパーティーのパブリック拡張機能をアクティブ化する方法を示します。これにより、それがアクティブ化されたアカウントとリージョンで使えるようになります。

Note

続行する前に、この拡張機能で使用する [IAM ロール](#) が作成されていることを確認してください。

トピック

- [パブリック拡張をアクティブ化する \(コンソール\)](#)
- [パブリック拡張を有効にする \(AWS CLI\)](#)

パブリック拡張をアクティブ化する (コンソール)

このセクションの手順に従い、コンソールを使用して次の操作を実行します。

- サードパーティーのパブリック拡張を有効化する
- アカウントの追加の拡張機能構成データを指定する

アカウントで使用するパブリック拡張を有効にするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. ナビゲーションペインの [レジストリ] の下で、[パブリック拡張機能] を選択します。
4. [フィルタ] を使用して拡張タイプを選択し、[サードパーティー] を選択します。AWS によって発行されている拡張がデフォルトで有効化されています。
5. 拡張を選択してから、[有効化] を選択します。

拡張の複数のバージョンが使用可能な場合は、[バージョン] メニューを使用し、有効にする拡張のバージョンを選択します。デフォルトは最新バージョンです。

6. [拡張機能の名前] は、[デフォルトを使用] のままにするか、[デフォルトを上書き] を選択し、この拡張機能に使用する拡張機能タイプのエイリアスを入力します。エイリアスは、拡張タイプの推奨形式に従う必要があります。詳細については、「[拡張を参照するエイリアスを使用する](#)」を参照してください。
7. アクティブ化している拡張機能がフックのタイプまたはリソースタイプである場合、[実行ロールの ARN] で、この拡張機能呼び出すときに CloudFormation が引き受ける IAM ロールを指定します。詳細については、「[IAM アクセス許可とパブリック拡張アクセスの信頼ポリシーを使用して実行ロールを設定する](#)」を参照してください。
8. [ログ記録設定] で、拡張機能のログ記録設定の情報を指定します。例えば、次のようになります。

```
{
  "logRoleArn": "arn:aws:iam::account:role/rolename",
  "logGroupName": "log-group-name"
}
```

構成情報のロギングは必須ではありませんが、デバッグの目的で推奨されます。フックでログ記録設定を使用するには、指定された実行ロールと同じ信頼ポリシーを追加して、ログロールがロググループにログを書き込めるようにします。

logRoleArn および logGroupName キー名では大文字と小文字は区別されません。

9. [バージョニング] の [自動更新] で、更新の受信方法を選択します。
 - [On] (オン) - 自動的に最新のマイナーバージョンに更新します。メジャーバージョンは手動で更新されます。

- [Off] (オフ) - 自動的に最新バージョンに更新しません。すべてのバージョンは手動で更新されます。

詳細については、「[拡張の新しいバージョンを自動的に使用する](#)」を参照してください。

拡張機能に追加の構成が必要な場合は、構成データを今すぐ指定するか、拡張が有効化された後に指定するかを選択できます。

Important

有効化する拡張機能がフックの場合、このステップが必要です。HookInvocationStatus プロパティには ENABLED を指定する必要があります。このオペレーションにより、フックのスキーマ properties セクションで定義されているフックのプロパティが有効になります。詳細については、「CloudFormation フックのユーザーガイド」の「[フック設定スキーマ構文リファレンス](#)」を参照してください。

構成データを指定するには

1. [設定] で、[今すぐ設定] を選択し、[拡張の有効化] を選択します。

CloudFormation は、[Configure extension] (拡張の設定) ページを表示します。拡張の現在の構成スキーマを表示するには、[View configuration schema] (構成スキーマの表示) が有効になっていることを確認してください。

2. [Configuration JSON (構成 JSON)] テキストボックスに、この拡張に指定する構成データを表す JSON 文字列を入力します。指定する JSON は、拡張の構成スキーマに対して検証する必要があります。
3. [Configure extension (拡張の設定)] を選択します。

アクティベーション後に拡張機能を設定する場合は、このステップをスキップし、後で構成データを提供できます。

1. [設定] で、[後で設定] を選択し、[拡張の有効化] を選択します。
2. 拡張機能が有効化されたら、有効化された拡張機能ページから拡張機能に移動し、構成データを提供することで、拡張機能を設定できます。

パブリック拡張を有効にする (AWS CLI)

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- サードパーティーのパブリック拡張を有効化する
- アカウントの追加の拡張機能構成データを指定する

パブリックフックを有効化する

アカウントでフックを有効にすると、AWS アカウント から定義されたアクセス許可を使用するフックを承認します。CloudFormation は、アクセス許可をフックに渡す前に、不要なアクセス許可を削除します。CloudFormation では、アカウントでフックを有効にする前に、お客様またはフックユーザーがフックのアクセス許可を確認し、フックが許可されているアクセス許可を考慮することをお勧めします。

アカウントで使用するパブリックフックを有効にするには (AWS CLI)

1. フックの ARN を取得し、保存します。AWS マネジメントコンソール または AWS CLI を使用してフックの ARN を取得できます。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。

```
export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/  
Organization-Service-Hook/"
```

2. [set-type-configuration](#) コマンドを使用して、構成データを指定します。--configuration に渡す JSON は、フックの構成スキーマに対して検証する必要があります。すべてのスタックオペレーションでフックを有効にするには、HookConfiguration セクションで HookInvocationStatus プロパティを ENABLED に設定する必要があります。

```
aws cloudformation set-type-configuration \  
  --configuration '{"CloudFormationConfiguration":{"HookConfiguration":  
  {"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}' \  
  --type-arn $HOOK_TYPE_ARN --region us-west-2
```

HookConfiguration の設定オプションの詳細については、「CloudFormation フックのユーザーガイド」の「[フック設定スキーマ構文リファレンス](#)」を参照してください。

パブリックモジュールとリソースタイプを有効にする

アカウントで使用するパブリック拡張を有効にするには (AWS CLI)

- [activate-type](#) コマンドを使用して拡張を有効にし、拡張の新しいマイナーバージョンが公開されるたびに拡張を自動更新するかどうかを指定します。

以下の例では、このアカウントに対して有効にするパブリック拡張のパブリック Amazon リソースネーム (ARN) を指定します。さらに、新しいマイナーバージョンが公開されるたびに CloudFormation が拡張を更新するように指定しています。

```
aws cloudformation activate-type \  
  --public-type-arn public_extension_ARN \  
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \  
  --auto-update true --region us-west-2
```

このコマンドは、有効化された拡張の ARN を返します。

```
{  
  "Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-  
Resource-Example"  
}
```

パブリック拡張のバージョンを更新する (AWS CLI)

[activate-type](#) を使用して拡張を再度有効にします。

--version-bump オプションを使用して、拡張を最新の MAJOR バージョン、または最新の MINOR バージョンに更新するかどうかを指定します。

```
aws cloudformation activate-type --type RESOURCE \  
  --type-name Example::Test::1234567890abcdef0 \  
  --type-name-alias Example::Test::Alias \  
  --version-bump MAJOR --region us-west-2
```

アカウントでパブリックサードパーティー拡張を更新する

サードパーティーのパブリック拡張を有効にすると、アカウントからほとんどの拡張機能の詳細を更新できます。

アカウントでパブリック拡張を更新するには (コンソール)

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. ナビゲーションペインの [レジストリ] の下で、[アクティブ化済みの拡張機能] を選択します。
4. 更新する拡張機能を見つけ、選択します。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。
5. [アクション] メニューから [編集] を選択し、次に適切な編集オプションを選択します。
 - 構成スキーマを更新するには、「[アカウントの拡張機能の構成データを編集する](#)」を参照してください。
 - 自動更新を有効または無効にするには:
 1. [自動更新の編集] を選択します。
 2. [オン] または [オフ] を選択した上で、[保存] を選択します。詳細については、「[拡張の新しいバージョンを自動的に使用する](#)」を参照してください。
 - 実行ロールを更新するには、次の手順を実行します。
 1. [実行ロールの編集] を選択します。
 2. この拡張を呼び出すときに CloudFormation が使用する IAM ロールの ARN を指定し、次に [保存] を選択します。詳細については、「[IAM アクセス許可とパブリック拡張アクセスの信頼ポリシーを使用して実行ロールを設定する](#)」を参照してください。
 - ログ記録設定を更新するには、次の手順を実行します。
 1. [ログ記録設定の編集] を選択します。
 2. ログ記録設定の JSON を編集し、次に [保存] を選択します。

アカウントでサードパーティーのパブリック拡張を無効にする

有効にしたサードパーティーのパブリック拡張が不要になった場合は、次の手順を使用してアカウントで無効にします。

トピック

- [アカウントでパブリック拡張を無効にする \(コンソール\)](#)
- [アカウントでパブリック拡張を無効にする \(AWS CLI\)](#)
- [アカウントのフックを無効にする \(AWS CLI\)](#)

アカウントでパブリック拡張を無効にする (コンソール)

アカウントでパブリック拡張を無効にするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. ナビゲーションペインの [レジストリ] の下で、[アクティブ化済みの拡張機能] を選択します。
4. 無効にする拡張機能を見つけ、選択します。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。
5. [アクション] メニューから、[無効化] を選択します。
6. [無効化] を選択します。

アカウントでパブリック拡張を無効にする (AWS CLI)

次の [deactivate-type](#) コマンドを使用します。

```
aws cloudformation deactivate-type --type MODULE \  
  --type-name Example::Test::Type::MODULE \  
  --region us-west-2
```

アカウントのフックを無効にする (AWS CLI)

フックを無効にすると、フックを削除することなく、AWS アカウント でのフックの実行ができなくなります。

[set-type-configuration](#) コマンドを使用し、HookInvocationStatus を DISABLED として指定して、フックを無効にします。

次の例では、無効化されるフックの AWS リージョン と Amazonリソースネーム (ARN) を指定します。

```
aws cloudformation set-type-configuration \  
  --configuration '{"CloudFormationConfiguration":{"HookConfiguration":  
{"HookInvocationStatus": "DISABLED", "FailureMode": "FAIL", "Properties":{}}}}' \  
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" --  
  region us-west-2
```

詳細については、「CloudFormation フックユーザーガイド」の「[Disable and enable CloudFormation Hooks](#)」を参照してください。

共有されているサードパーティーのプライベート拡張機能を使用する

共有されているサードパーティーのプライベート拡張機能を使用するには、まずそれらを使用するアカウントとリージョンで、CloudFormation に登録する必要があります。拡張を登録すると、そのコピーがアカウントの CloudFormation レジストリにアップロードされ、アクティブ化されます。プライベート拡張を登録すると、その AWS アカウント とリージョンの CloudFormation レジストリに表示され、スタックテンプレートで使用できます。

トピック

- [サードパーティーのプライベート拡張の登録のための IAM アクセス許可](#)
- [プライベート拡張機能进行操作するためによく使用される AWS CLI コマンド](#)
- [アカウントにサードパーティーのプライベート拡張機能を登録する](#)
- [アカウントからのサードパーティープライベート拡張機能の削除](#)

サードパーティーのプライベート拡張の登録のための IAM アクセス許可

プライベート拡張の登録の一環として、拡張の登録の一環として、拡張のプロジェクトパッケージを含む Amazon S3 バケットを指定します。このパッケージには、登録する拡張に必要なソースファイルが含まれています。拡張を登録するユーザーは、Amazon S3 バケット内のプロジェクトパッケージにアクセスする必要があります。これを行うには、ユーザーは、拡張パッケージに対する [GetObject](#) アクセス許可を持っている必要があります。

これは、AWS CLI の [register-type](#) コマンドを使用している場合でも、CloudFormation CLI の [submit](#) コマンドを使用している場合でも当てはまります。

詳細については、「サービス認可リファレンス」の「[Actions, resources, and condition keys for Amazon S3](#)」を参照してください。

プライベート拡張機能进行操作するためによく使用される AWS CLI コマンド

プライベート拡張機能进行操作のために一般的に使用されるコマンドは次のとおりです。

- [register-type](#) は、プライベート拡張機能をアカウントに登録します。
- [describe-type-registration](#) は、登録リクエストの現在のステータスを返します。
- [list-types](#) は、アカウントの拡張機能を一覧表示します。
- [describe-type](#) は、現在の設定データなど、特定の拡張機能または特定の拡張機能バージョンに関する詳細情報を返します。
- アカウント内の拡張機能の設定データを指定し、フックを無効または有効にする [set-type-configuration](#)。
- [set-type-default-version](#) は、どのバージョンの拡張機能がデフォルトバージョンであるかを指定します。
- [deregister-type](#) は、アカウントからプライベート拡張機能または拡張機能バージョンを削除します。

アカウントにサードパーティーのプライベート拡張機能を登録する

このトピックでは、アカウントでできるように、共有されているサードパーティーのプライベート拡張機能を登録するステップについて説明します。

Note

続行する前に、プライベート拡張を登録するために必要な [IAM アクセス許可](#) があることを確認してください。

共有されているプライベート拡張機能を登録するには (AWS CLI)

1. アカウントに登録するプライベート拡張のプロジェクトパッケージを含む Amazon S3 バケットを見つけます。
2. [register-type](#) コマンドを使用して、アカウントにプライベート拡張機能を登録します。

例えば、次のコマンドは、指定した AWS アカウント に `My::Resource::Example` リソースタイプを登録します。

```
aws cloudformation register-type --type RESOURCE \  
  --type-name My::Resource::Example \  
  --schema-handler-package [s3 object path] --region us-west-2
```

RegisterType は非同期オペレーションであり、登録リクエストの進行状況を追跡するために使用できる登録トークンを返します。

```
{
  "RegistrationToken": "f5525280-104e-4d35-bef5-8f1fexample"
}
```

拡張がその機能の一部として AWS API を呼び出す場合は、これらの AWS API を呼び出すために必要なアクセス許可を含む IAM 実行ロールを作成し、アカウントにその実行ロールをプロビジョニングする必要があります。その後、`--execution-role-arn` オプションを使用してこの実行ロールを指定できます。CloudFormation は、その実行ロールを引き受け、リソースタイプに適切な認証情報を提供します。

```
--execution-role-arn arn:aws:iam::123456789012:role/MyIAMRole
```

3. (オプション) 登録トークンを [describe-type-registration](#) コマンドとともに使用して、登録リクエストの進行状況を追跡します。

CloudFormation が登録リクエストを完了すると、リクエストの進捗ステータスが COMPLETE に設定されます。

次の例では、上記の `describe-type-registration` コマンドによって返された登録トークンを使用して、登録ステータス情報を返します。

```
aws cloudformation describe-type-registration \
  --registration-token f5525280-104e-4d35-bef5-8f1fexample \
  --region us-west-2
```

このコマンドは、以下の出力を返します。

```
{
  "ProgressStatus": "COMPLETE",
  "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-Resource-Example",
  "Description": "Deployment is currently in DEPLOY_STAGE of status COMPLETED; ",
  "TypeVersionArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-Resource-Example/00000001"
}
```

⚠ Important

登録する拡張機能がフックの場合、この次のステップが必要です。HookInvocationStatus プロパティには ENABLED を指定する必要があります。このオペレーションにより、フックのスキーマ properties セクションで定義されているフックのプロパティが有効になります。詳細については、「CloudFormation フックのユーザーガイド」の「[フック設定スキーマ構文リファレンス](#)」を参照してください。

フックの設定データを指定するには (AWS CLI)

1. フックの ARN を取得し、保存します。AWS マネジメントコンソール または AWS CLI を使用してフックの ARN を取得できます。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。

```
export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/  
Organization-Service-Hook/"
```

2. [set-type-configuration](#) コマンドを使用して、構成データを指定します。--configuration に渡す JSON は、フックの構成スキーマに対して検証する必要があります。フックをアクティブ化するには、HookConfiguration セクションで HookInvocationStatus プロパティを ENABLED に設定する必要があります。

```
aws cloudformation set-type-configuration \  
  --configuration '{"CloudFormationConfiguration":{"HookConfiguration":  
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}' \  
  --type-arn $HOOK_TYPE_ARN --region us-west-2
```

詳細については、「CloudFormation フックのユーザーガイド」の「[フック設定スキーマ構文リファレンス](#)」を参照してください。

アカウントからのサードパーティープライベート拡張機能の削除

サードパーティのプライベート拡張または拡張機能バージョンを削除するには、[deregister-type](#) コマンドを使用します。

特定の拡張機能バージョン、または拡張機能全体を登録解除できます。拡張機能の登録を解除するには、その拡張機能のすべての登録済みバージョンを個別に登録解除する必要があります。拡張機能に

登録されたバージョンが 1 つだけの場合、そのバージョンの登録を解除すると、拡張機能自体の登録が解除されます。拡張機能のデフォルトバージョンを登録解除することはできません。ただし、その拡張機能の唯一の登録済みバージョンの場合は別で、その場合、その拡張機能自体も登録解除されます。

Warning

プライベート拡張機能の登録は解除できません。このアクションにより、以下のようになります。

- 拡張機能をすべての CloudFormation オペレーションで使用できないようにする。
- この拡張機能を使用する今後のスタック更新で失敗する可能性がある (モジュールとリソースタイプの場合)。拡張機能は後でプライベートで再登録できますが、CloudFormation が以前のバージョンに依存している場合、これが失敗の原因となる可能性があります。

続行する前に、[\[list-stacks\]](#) コマンドと [\[get-template\]](#) コマンドを使用して、アクティブなスタックがこの拡張機能を使用していないことを確認します。

拡張機能の登録解除コマンドの例

このセクションでは、プライベート拡張機能の登録を解除するさまざまな方法を示す例を紹介します。

タイプ名による登録解除

[deregister-type](#) コマンドを `--type` および `--type-name` オプションとともに使用して、拡張機能の登録を解除します。

```
aws cloudformation deregister-type \  
  --type MODULE \  
  --type-name My::S3::SampleBucket::MODULE
```

タイプ名とバージョンによる登録解除

特定のバージョンの拡張機能の登録を解除するには、コマンドで `--version-id` オプションを指定します。

```
aws cloudformation deregister-type \  
  --version-id VERSION_ID
```



```
--type MODULE \  
--type-name My::S3::SampleBucket::MODULE \  
--version-id 00000001
```

Tip

最初に別のバージョンの拡張機能をデフォルトとして設定するには、[set-type-default-version](#) コマンドを使用します。

ARN による登録解除

--arn オプションを使用して拡張機能の ARN を指定し、登録を解除します。

```
aws cloudformation deregister-type \  
  --arn arn:aws:cloudformation:us-west-2:123456789012:type/resource/Organization-Service-Resource
```

アカウントの拡張機能の構成データを編集する

このトピックでは、特定のリージョン内のアカウントで拡張機能の構成データを編集するためのガイドランスを提供します。拡張機能には、特定のアカウントとリージョンの拡張のすべてのインスタンスに適用される設定プロパティを含めることができます。これらは、拡張機能の作成者が拡張機能の設定定義で定義します。拡張の構成定義に必要なプロパティがある場合、アカウントとリージョンで拡張を使用する前に、それらのプロパティを指定する必要があります。

拡張機能の開発時に設定定義を定義する方法の詳細については、以下のドキュメントを参照してください。

- [フック設定スキーマ構文リファレンス](#)
- [拡張のアカウントレベルの構成を定義する](#)

トピック

- [動的リファレンスを使用するために必要なアクセス許可](#)
- [拡張機能の構成データを編集する \(コンソール\)](#)
- [拡張機能の構成データを編集する \(AWS CLI\)](#)

動的リファレンスを使用するために必要なアクセス許可

構成データに、AWS Systems Manager または AWS Secrets Manager に格納されている値への動的参照が含まれている場合では、タイプをプロビジョニングするために使用されるロール (スタックの作成や更新など) には、その値を取得するための適切なアクセス許可が必要です。具体的には次のとおりです。

- 構成データに AWS Systems Manager Parameter Store に格納されたパラメータが含まれている場合、タイプをプロビジョニングするために使用されるユーザーまたはロールには、[GetParameter](#) を呼び出すためのアクセス許可が必要です。
- 構成データに AWS Secrets Manager に格納されたシークレットが含まれている場合、タイプをプロビジョニングするために使用されるユーザーまたはロールには、[GetSecretValue](#) を呼び出すための許可が必要です。

詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

拡張機能の構成データを編集する (コンソール)

このセクションの手順に従い、コンソールを使用して次の操作を実行します。

- 拡張機能の現在の構成データを表示する
- アカウントの拡張機能構成データを更新する

拡張機能の現在の構成データを表示するには

- AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
- 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
- ナビゲーションペインの [レジストリ] の下で、[アクティブ化済みの拡張機能] を選択します。
- 表示する拡張機能を見つけます。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。
- 拡張を選択して、拡張の詳細を表示します。
- [extension details] (拡張の詳細) ページで、[Configuration] (構成) タブを選択します。
- [Configuration schema] (構成スキーマ) タブを展開して、拡張に定義されている構成スキーマを確認します。

8. [Configuration] (構成) タブを展開して、この拡張に設定した現在の構成を確認します。

拡張機能の構成データを更新するには

1. 拡張の詳細ページで、[Configuration (構成)] タブから [Edit configuration (構成を編集する)] を選択します。

または、[アクション] から、[編集] を選択し、次に [Edit configuration (構成を編集する)] を選択します。

CloudFormation は、[Configure extension] (拡張の設定) ページを表示します。拡張の現在の構成定義スキーマを確認するには、[View configuration schema] (構成のスキーマの表示) がオンになっていることを確認してください。

2. [Configuration JSON] (構成 JSON) テキストボックスに、この拡張に設定する構成スキーマを表す JSON 文字列を入力します。[Configuration schema] (構成スキーマ) で定義されたスキーマに対して検証する必要があります
3. [Configure extension (拡張の設定)] を選択します。

拡張機能の構成データを編集する (AWS CLI)

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- 拡張機能の現在の構成データを表示する
- アカウントの拡張機能構成データを更新する

拡張機能の現在の構成データを表示するには

- [describe-type](#) コマンドを使用して、拡張機能の詳細情報を返します。出力の ConfigurationSchema 要素には、指定されたリージョンの拡張の現在の構成定義が含まれます。

または、[batch-describe-type-configurations](#) コマンドを使用して、複数の拡張に関する構成データを返します。

拡張機能の構成データを更新するには

- [set-type-configuration](#) コマンドを使用して、構成データを指定します。--configuration に渡す JSON は、拡張の構成定スキーマに対して検証する必要があります。

次の例では、set-type-configuration コマンドは --configuration オプションに対し構成データ `{"CredentialKey": "testUserCredential"}` を指定します。

```
aws cloudformation set-type-configuration --type RESOURCE \  
  --type-name My::Resource::Example \  
  --configuration-alias default \  
  --configuration {"CredentialKey": "testUserCredential"} \  
  --region us-west-2
```

AWS Config でリソースタイプを記録する

AWS Config がプライベートリソースタイプを自動的に追跡し、それらのリソースに対する変更を構成項目として記録するように指定できます。これにより、構成のベストプラクティスを検証するための AWS Config ルールを作成するだけでなく、これらのプライベートリソースタイプの構成履歴を表示することができます。AWS Config はフック拡張に必要です。

AWS Config でプライベートリソースタイプを自動的に追跡するには:

- CloudFormation を使用してリソースを管理します。これには、CloudFormation によるすべてのリソースの作成、更新、削除オペレーションの実行が含まれます。

Note

IAM ロールを使用してスタックオペレーションを実行する場合、その IAM ロールには次の AWS Config アクションを呼び出すためのアクセス権限が必要です。

- [PutResourceConfig](#)
- [DeleteResourceConfig](#)

- すべてのリソースタイプを記録するように AWS Config を設定します。詳細については、「AWS Config デベロッパーガイド」の「[AWS CLI を使用したサードパーティーのリソースの記録構成](#)」を参照してください。

Note

AWS Config は、必須と書き込み専用の両方として定義されたプロパティを含むプライベートリソースの記録をサポートしていません。

設計上、書き込み専用として定義されたリソースプロパティは、AWS Config 構成項目の作成に使用されるスキーマでは返されません。このため、書き込み専用と必須の両方として定義されたプロパティを含めると、必須プロパティが存在しないため、構成項目の作成が失敗します。構成項目の作成に使用されるスキーマを表示するには、[DescribeType](#) アクションの `schema` プロパティを確認します。

構成項目の詳細については、「AWS Config デベロッパーガイド」の「[Configuration items](#)」(構成項目) を参照してください。

構成項目に機密のプロパティが記録されないようにする

リソースタイプには、パスワード、シークレット、その他の機密データなど、構成項目の一部として記録したくない、機密情報とみなされるプロパティが含まれている場合があります。プロパティが構成項目に記録されないようにするには、そのプロパティをリソースタイプスキーマの `writeOnlyproperties` 一覧に含めます。ユーザーは `writeOnlyproperties` としてリストされているリソースプロパティを指定できますが、`read` または `list` リクエストでは返されなくなります。

詳細については、「CloudFormation CLI ユーザーガイド」の「[writeOnlyProperties](#)」を参照してください。

CodePipeline を使用した継続的デリバリー

継続的デリバリーは、コードの変更が毎回自動的に構築、テストされ、本稼働環境へのリリースのために準備されるリリースのプラクティスです。CloudFormation および CodePipeline では、継続的な配信を使用して、CloudFormation テンプレートへの変更を自動的に構築してテストしてから、本稼働スタックに昇格させることができます。このリリースプロセスにより、すばやく確実に AWS インフラストラクチャに変更を加えることができます。

たとえば、コードリポジトリに更新されたテンプレートを送信すると、自動的にテストスタックを構築するワークフローを作成できます。CloudFormation がテストスタックを作成した後、それをテストして、変更を本稼働スタックにプッシュするかどうかを決定できます。継続的デリバリーの利点の詳細については、「[継続的な配信とは](#)」を参照してください。

CodePipeline を使用して、CloudFormation スタックのパイプラインを構築することにより、継続的な配信ワークフローを構築します。CodePipeline には CloudFormation との統合機能が組み込まれているため、パイプライン内でスタックの作成、更新、削除など CloudFormation 固有のアクションを指定できます。CodePipeline の詳細については、[AWS CodePipeline ユーザーガイド](#)を参照してください。

トピック

- [チュートリアル: テストおよび本稼働スタック用のパイプラインを構築する](#)
- [CloudFormation 構成プロパティのリファレンス](#)
- [CloudFormation アーティファクト](#)
- [CodePipeline パイプラインでのパラメーターオーバーライド関数の使用](#)

チュートリアル: テストおよび本稼働スタック用のパイプラインを構築する

CloudFormation テンプレートを送信すると、CloudFormation がそれを使用して自動的にテストスタックを構築するリリースプロセスを考えてみてください。テストスタックを確認してから、変更によって本稼働スタックがどのように変更されるかをプレビューして、実装するかどうかを選択できます。このワークフローは、CloudFormation を使用してテストスタックの構築、テストスタックの削除、変更設定の作成、そして変更設定の実行を行うことで実現できます。ただし、各アクションで、手動で CloudFormation を操作する必要があります。このウォークスルーでは、このようなアクショ

ンの多くを自動化する CodePipeline パイプラインを構築し、CloudFormation スタックを使用した継続的な配信ワークフローを実現します。

前提条件

このウォークスルーは、お客様は CodePipeline と CloudFormation を使用した経験があり、パイプライン、AWS CloudFormation テンプレート、スタックの動作について理解していることを前提としています。CodePipeline の詳細については、[AWS CodePipeline ユーザーガイド](#)を参照してください。また、パイプラインを作成するのと同じ AWS リージョン に Amazon S3 バケットがある必要があります。

Important

サンプルの WordPress テンプレートは、インターネットへの接続を必要とする EC2 インスタンスを作成します。インターネットへのトラフィックを許可しているデフォルト VPC とサブネットがあることを確認してください。

チュートリアル概要

このウォークスルーでは、スタックにある WordPress サンプルサイト用のパイプラインを構築します。パイプラインは、3 つのステージに分かれています。各ステージに少なくとも 1 つのアクションが含まれている必要があります。このアクションは、お客様のアーティファクト (入力) に対してパイプラインが実行するタスクです。ステージはパイプラインのアクションを整理します。ステージが新しいアーティファクトを処理する前に (例えば、パイプラインを再実行する新しい入力を送信した場合)、CodePipeline は 1 つのステージですべてのアクションを完了する必要があります。

このウォークスルーを完了すると、次のワークフローを実行するパイプラインができあがります。

1. パイプラインの最初のステージで、リポジトリからソースアーティファクト (CloudFormation テンプレートおよび設定ファイル) を取得します。

WordPress のサンプルテンプレートを含むアーティファクトを準備して、S3 バケットにアップロードします。

2. 2 番目のステージで、パイプラインはテストスタックを作成し、承認を待ちます。

テストスタックを確認した後、元のパイプラインを使用して続行するか、別のアーティファクトを作成して送信することで変更を加えるかを選択できます。承認すると、このステージはテストスタックを削除し、パイプラインは次のステージに進みます。

3. 3 番目のステージで、パイプラインは本稼働スタックに対する変更セットを作成し、承認を待ちます。

初めて実行する場合は、本稼働スタックはありません。変更設定には、CloudFormation によって作成されるリソースがすべて表示されます。承認すると、このステージは変更セットを実行し、本番用スタックを構築します。

Note

CloudFormation は無料サービスです。ただし、スタックに含める EC2 インスタンスなどの AWS リソースには、それぞれの現在の料金が課金されます。AWS 料金に関する詳細については、<http://aws.amazon.com> で各製品の詳細ページを参照してください。

ステップ 1: アーティファクトを編集し、S3 バケットにアップロードする

パイプラインを構築する前に、ソースリポジトリとファイルをセットアップする必要があります。CodePipeline はこれらのソースファイルをパイプラインの [アーティファクトストア](#) にコピーし、CloudFormation スタックの作成など、パイプライン内でアクションの実行に使用されます。

ソースリポジトリとして Amazon Simple Storage Service (Amazon S3) を使用する場合、CodePipeline ではソースファイルを zip にしてから S3 バケットにアップロードする必要があります。zip ファイルは、CloudFormation テンプレート、テンプレート設定ファイル、またはその両方を含む CodePipeline アーティファクトです。WordPress サンプルテンプレートと 2 つのテンプレート構成ファイルを含むアーティファクトが用意されています。2 つの構成ファイルは、WordPress テンプレートのパラメータ値を指定します。CodePipeline は WordPress スタックを作成時に使用します。ファイルの 1 つにはテストスタック用のパラメーター値、もう 1 つには本稼働スタック用のパラメーター値が含まれます。たとえばお客様が所有する既存の EC2 キーペア名を指定するために、構成ファイルを編集する必要があります。アーティファクトの詳細については、「[CloudFormation アーティファクト](#)」を参照してください。

アーティファクトを構築したら、S3 バケットにアップロードします。

アーティファクトを編集してアップロードするには

1. サンプルアーティファクトをダウンロードして開きます (<https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/wordpress-single-instance.zip>)。

アーティファクトには 3 つのファイルが含まれています。

- WordPress サンプルテンプレート: `wordpress-single-instance.yaml`
 - テストスタック用テンプレート構成ファイル: `test-stack-configuration.json`
 - 本稼働スタック用テンプレート構成ファイル: `prod-stack-configuration.json`
2. ファイルをすべて展開し、任意のテキストエディタを使用してテンプレート構成ファイルを変更します。

構成ファイルを開いて、WordPress テンプレートのパラメーターにマッピングするキーと値のペアが含まれていることを確認します。構成ファイルは、パイプラインがテストスタックおよび本稼働スタックを作成するときに使用するパラメーター値を指定します。

テストスタック用のパラメーター値を指定するには `test-stack-configuration.json` ファイルを編集します。本稼働スタック用には `prod-stack-configuration.json` ファイルです。

- DBPassword キーおよび DBRootPassword キーの値の値を、WordPress データベースへのログインに使用するパスワードに変更します。WordPress テンプレートで定義されたとおり、このパラメーターの値は英文字のみを使用する必要があります。
 - KeyName キーの値を、パイプラインを作成するリージョンにある既存の EC2 キーペア名に変更します。
3. 変更した構成ファイルを元のアーティファクト (.zip) ファイルに追加して、重複したファイルを置き換えます。

これで、S3 バケットにアップロードできるカスタマイズされたアーティファクトができました。

4. [お客様が所有する S3 バケットにアーティファクトをアップロードします。](#)

ファイルの場所をメモしておきます。パイプラインを構築するときに、このファイルの場所を指定します。

アーティファクトおよび S3 バケットに関する注意事項:

- パイプラインを作成する AWS リージョンと同じリージョンに存在するバケットを使用します。
- CodePipeline では、バケットは [バージョニングが有効](#) になっている必要があります。
- ソースリポジトリには、GitHub や CodeCommit など、ファイルをアップロードする前に zip 化する必要がないサービスも使用できます。

- アーティファクトには、パスワードなどの機密情報が含まれることがあります。アクセスを制限して、許可されたユーザーのみがファイルを表示できるようにしてください。これを行う際、CodePipeline がファイルにアクセスできるようにしてください。たとえば、アーティファクトを S3 バケットにアップロードする場合、[S3 バケットポリシー](#)または[ユーザーポリシー](#)を使用してアクセスを制限します。

これで、CodePipeline がパイプラインに引き込むことができるアーティファクトができました。次のステップでは、アーティファクトの位置を指定して、WordPress のパイプラインを構築します。

ステップ 2: パイプラインスタックを作成する

WordPress パイプラインを作成するには、サンプルの CloudFormation テンプレートを使用します。パイプラインの構築に加えて、テンプレートは CodePipeline と CloudFormation 用に AWS Identity and Access Management (IAM) サービスロール、CodePipeline アーティファクトストア用の S3 バケット、Amazon Simple Notification Service (Amazon SNS) トピック (レビューの通知など、パイプラインが通知を送付する送信先) をセットアップします。サンプルテンプレートを使用すると、単一の CloudFormation スタック内のこのようなリソースを簡単にプロビジョニングし設定できます。

パイプラインの構成の詳細については、「[パイプラインの仕組み](#)」を参照してください。

Important

サンプルの WordPress テンプレートは、インターネットへの接続を必要とする EC2 インスタンスを作成します。ご使用のデフォルト VPC とサブネットがインターネットへのトラフィックを許可していることを確認してください。

パイプラインスタックを作成するには

1. <https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/basic-pipeline.yml> でサンプルテンプレートをダウンロードします。ご使用のコンピューターに保存します。
2. クラウドフォーメーション コンソール の <https://console.aws.amazon.com/cloudformation/> 開きます：
3. CodePipeline および CloudFormation をサポートする AWS リージョンを選択します。

詳細については、AWS 全般のリファレンスの「[CodePipeline endpoints and quotas](#)」と「[CloudFormation endpoints and quotas](#)」を参照してください。

4. [スタックの作成] を選択してください。
5. [Specify template (テンプレートの指定)] で [Upload a template file (テンプレートファイルのアップロード)] を選択し、先ほどダウンロードしたテンプレート `basic-pipeline.yml` を選択します。
6. [次へ] を選択します。
7. [スタックの名前] に `sample-WordPress-pipeline` を入力します。
8. [パラメータ] セクションで、次のパラメータ値を指定し、[次へ] を選択します。WordPress テンプレートとその構成ファイルの名前を同じにしている場合は、スタックのパラメーターを構成する際にデフォルト値を使用できます。そうでない場合は、使用するファイル名を指定します。

PipelineName

パイプラインの名前 (たとえば `WordPress-test-pipeline`)。

S3 バケット

アーティファクト (.zip ファイル) を保存した S3 バケットの名前。

SourceS3Key

アーティファクトのファイル名。アーティファクトをフォルダに保存した場合、`folder/subfolder/wordpress-single-instance.zip` のように、ファイル名の一部に含めます。

Email

CodePipeline からのパイプラインの通知を受信するメールアドレス (myemail@example.com など)。

9. このチュートリアルでは、タグの追加も詳細設定の指定も不要です。[次へ] を選択します。
10. スタック名とテンプレート URL が正しいことを確認し、[スタックの作成] を選択します。
11. CloudFormation が IAM リソースを作成する可能性があることを認識していることを確認するには、チェックボックスをオンにします。

AWS CloudFormation によってスタックが作成されるまでに数分かかることもあります。進捗状況を監視するには、スタックイベントを確認します。詳細については、「[スタックの進行状況を監視する](#)」を参照してください。

スタックが作成されると、CodePipeline は新しいパイプラインを開始します。ステータスを表示するには、「[CodePipeline コンソール](#)」を参照してください。パイプラインのリストから、[WordPress-test-pipeline] を選択します。

パイプラインの仕組み

このセクションでは、WordPress パイプラインテンプレートの例のスニペットを使用して、パイプラインの 3 つのステージについて説明します。

ステージ 1: Source

パイプラインの最初のステージはソースステージです。ここでソースコードの位置を指定します。この場所にリビジョンをプッシュするたびに、CodePipeline はパイプラインを再実行します。

ソースコードは S3 バケットにあり、ファイル名で識別されます。パイプラインのスタックを作成するときに、これらの値を入力パラメーター値として指定しました。その後のステージでソースアーティファクトを使用できるように、スニペットでは OutputArtifacts プロパティを TemplateSource という名前で指定しています。この後のステージでこのアーティファクトを使用するには、入力アーティファクトに TemplateSource を指定します。

```
- Name: S3Source
  Actions:
    - Name: TemplateSource
      ActionTypeId:
        Category: Source
        Owner: AWS
        Provider: S3
        Version: '1'
      Configuration:
        S3Bucket: !Ref 'S3Bucket'
        S3ObjectKey: !Ref 'SourceS3Key'
      OutputArtifacts:
        - Name: TemplateSource
```

ステージ 2: TestStage

TestStage ステージでは、パイプラインはテストスタックを作成し、承認を待ってから、テストスタックを削除します。

CreateStack アクションでは、パイプラインはテスト構成ファイルおよび WordPress テンプレートを使用してテストスタックを作成します。ファイルはどちらもソースステージで持ち込まれた TemplateSource 入力アーティファクトに含まれています。このスニペットで

は、REPLACE_ON_FAILURE アクションモードを使用します。スタックの作成が失敗すると、パイプラインはこれを置き換えます。パイプラインを再実行する前にスタックを消去したりトラブルシューティングする必要はありません。アクションモードは、テストスタックですばやく反復処理する場合に便利です。RoleArn プロパティの値は、テンプレートの他の場所で指定されている AWS CloudFormation サービスロールです。

ApproveTestStack アクションはパイプラインを一時停止し、パイプラインスタックの作成時に指定された E メールアドレスに通知を送信します。パイプラインが一時停止している間に、WordPress テストスタックとそのリソースを確認できます。CodePipeline を使用してこのアクションを承認または却下します。CustomData プロパティには、承認するアクションの説明が含まれています。これはパイプラインによって通知 E メールに追加されています。

このアクションを承認すると、CodePipeline は DeleteTestStack アクションに移行し、テスト WordPress スタックおよびそのリソースを削除します。

```
- Name: TestStage
  Actions:
    - Name: CreateStack
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      InputArtifacts:
        - Name: TemplateSource
      Configuration:
        ActionMode: REPLACE_ON_FAILURE
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref TestStackName
        TemplateConfiguration: !Sub "TemplateSource::${TestStackConfig}"
        TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
      RunOrder: '1'
    - Name: ApproveTestStack
      ActionTypeId:
        Category: Approval
        Owner: AWS
        Provider: Manual
        Version: '1'
      Configuration:
        NotificationArn: !Ref CodePipelineSNSTopic
        CustomData: !Sub 'Do you want to create a change set against the production
stack and delete the ${TestStackName} stack?'
```

```
RunOrder: '2'
- Name: DeleteTestStack
  ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: CloudFormation
    Version: '1'
  Configuration:
    ActionMode: DELETE_ONLY
    RoleArn: !GetAtt [CFNRole, Arn]
    StackName: !Ref TestStackName
  RunOrder: '3'
```

ステージ 3: ProdStage

パイプラインの ProdStage ステージでは、既存の本稼働スタックに対する変更セットを作成し、承認を待ってから、変更セットを実行します。

AWS CloudFormation が本稼働スタックに対して行うすべての変更を、実装前に変更セットで確認できます。初めてパイプラインを実行する場合は、実行中の本稼働スタックはありません。変更セットには、テストスタック作成時に AWS CloudFormation が実行したアクションが表示されます。CreateChangeSet アクションは、変更セットの作成に、TemplateSource 入力アーティファクトからの WordPress サンプルテンプレートおよび本稼働テンプレート構成を使用します。

前のステージと同様に、ApproveChangeSet アクションでパイプラインが一時停止し、E メール通知が送信されます。パイプラインが一時停止している間に、変更セットを確認して、本稼働 WordPress スタックに対する変更案をすべて確認できます。CodePipeline を使用して、このアクションを[承認または却下](#)し、それに合わせてパイプラインを続行または停止します。

アクションを承認した場合、ExecuteChangeSet アクションによって変更セットが実行されるため、変更セットに記述されたアクションがすべて AWS CloudFormation によって実行されます。初めて実行する場合は、AWS CloudFormation によって WordPress 本稼働スタックが作成されます。それ以降の実行では、AWS CloudFormation はそのスタックを更新します。

```
- Name: ProdStage
  Actions:
    - Name: CreateChangeSet
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
```

```

    Version: '1'
  InputArtifacts:
    - Name: TemplateSource
  Configuration:
    ActionMode: CHANGE_SET_REPLACE
    RoleArn: !GetAtt [CFNRole, Arn]
    StackName: !Ref ProdStackName
    ChangeSetName: !Ref ChangeSetName
    TemplateConfiguration: !Sub "TemplateSource::${ProdStackConfig}"
    TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
  RunOrder: '1'
- Name: ApproveChangeSet
  ActionTypeId:
    Category: Approval
    Owner: AWS
    Provider: Manual
    Version: '1'
  Configuration:
    NotificationArn: !Ref CodePipelineSNSTopic
    CustomData: !Sub 'A new change set was created for the ${ProdStackName} stack.
Do you want to implement the changes?'
  RunOrder: '2'
- Name: ExecuteChangeSet
  ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: CloudFormation
    Version: '1'
  Configuration:
    ActionMode: CHANGE_SET_EXECUTE
    ChangeSetName: !Ref ChangeSetName
    RoleArn: !GetAtt [CFNRole, Arn]
    StackName: !Ref ProdStackName
  RunOrder: '3'

```

ステップ 3: WordPress スタックの表示

CodePipeline は、パイプラインを実行する際に、CloudFormation を使用してテストスタックおよび本稼働スタックを作成します。これらのスタックのステータスとその出力を表示するには、CloudFormation コンソールを使用します。

スタックを表示するには

1. クラウドフォーメーション コンソール の <https://console.aws.amazon.com/cloudformation/> 開きます :
2. パイプラインがテストステージにあるか本稼働ステージにあるかによって、Test-MyWordPressSite または Prod-MyWordPressSite スタックを選択します。
3. スタックのステータスを確認するには、スタック [イベント](#) を表示します。

スタックが失敗状態の場合は、ステータスの理由を参照してスタックのエラーを見つけください。エラーを修正してから、パイプラインを再実行します。スタックが CREATE_COMPLETE 状態にある場合は、その出力を表示して WordPress サイトの URL を取得します。

これで、CodePipeline を使用してサンプルの WordPress サイト用の継続的な配信フローを正常に構築できました。S3 バケットに変更を送信すると、CodePipeline は自動的に新しいバージョンを検出し、パイプラインを再実行します。このワークフローによって、本稼働サイトに変更を加える前に、簡単に変更を送信してテストできます。

ステップ 4: リソースをクリーンアップする

不要なサービスに対して課金されないように、リソースを削除します。

Important

パイプラインのスタックを削除する前に、テストおよび本稼働の WordPress スタックを削除します。パイプラインのスタックには、WordPress スタックを削除するために必要なサービスロールが含まれています。先にパイプラインのスタックを削除する場合は、WordPress スタックに別のサービスロールの Amazon リソースネーム (ARN) を関連付けてから、それらを削除できます。

アーティファクトストアのオブジェクトを削除するには

1. Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. パイプラインのアーティファクトストアとして CodePipeline が使用した S3 バケットを選択します。

バケット名の形式は次の通りです。*stackname-artifactstorebucket-id*このウォークスルーに従った場合は、バケット名は次の例に似たものになります。sample-WordPress-pipeline-artifactstorebucket-12345abcd12345

3. アーティファクトストア S3 バケットのすべてのオブジェクトを削除します。

次のステップでパイプラインのスタックを削除する際、このバケットは空である必要があります。そうしないと、CloudFormation がバケットを削除できません。

スタックを削除するには

1. CloudFormation コンソールから、削除するスタックを選択します。

パイプラインによって作成された WordPress スタックがまだ実行中であれば、それらを最初に選択します。デフォルトでは、スタック名は Test-MyWordPressSite および Prod-MyWordPressSite です。

既に WordPress スタックを削除してある場合は、sample-WordPress-pipeline スタックを選択します。

2. [アクション] を選択してから、[スタックの削除] を選択します。
3. 確認メッセージで、[はい、削除する] を選択します。

CloudFormation がスタックおよび EC2 インスタンス、通知トピック、サービスロール、パイプラインなどのスタックのリソースをすべて削除します。

これで、CodePipeline を使用した基本的な CloudFormation ワークフローの構築方法を理解できたため、サンプルのテンプレートとアーティファクトを開始点として使用して、独自に構築できます。

関連情報

これらのパラメータを利用する際に役立つ関連リソースは以下の通りです。

- CodePipeline の CloudFormation アクションパラメータの詳細については、「AWS CodePipeline ユーザーガイド」の「[アクション設定リファレンスの CloudFormation デプロイ](#)」を参照してください。
- アクションプロバイダーごとのテンプレート値における例の場合 (owner フィールドまたは configuration フィールドなど)、「AWS CodePipeline ユーザーガイド」の「[アクション構造リファレンス](#)」を参照してください。

- サンプルのパイプラインスタックテンプレートを YAML または JSON 形式でダウンロードするには、「AWS CodePipeline ユーザーガイド」の「[Tutorial: Create a pipeline with CloudFormation](#)」を参照してください。

CloudFormation 構成プロパティのリファレンス

CodePipeline パイプラインを構築する際、CloudFormation をプロバイダーとして Deploy アクションをパイプラインに追加します。その後、パイプラインが起動する CloudFormation アクションと、そのアクションの設定を指定する必要があります。このトピックでは、CloudFormation 構成プロパティについて説明します。プロパティを指定するには、CodePipeline コンソールを使用、あるいは JSON オブジェクトを作成して、AWS CLI、CodePipeline API、CloudFormation テンプレートを使用できます。

トピック

- [構成プロパティ \(コンソール\)](#)
- [構成プロパティ \(JSON オブジェクト\)](#)
- [関連情報](#)

構成プロパティ (コンソール)

CodePipeline [コンソール](#) に構成プロパティが表示され、選択したアクションモードに基づいて必須プロパティが示されます。

Note

新しいパイプラインを作成するときは、[Create or update a stack (スタックの作成または更新)] または [Create or replace a change set (変更セットの作成または置換)] のアクションモードのみを指定できます。また、[Advanced] セクションのプロパティは、既存のパイプラインを編集するときのみ使用できます。

アクションモード

CodePipeline が関連ステージを処理する際に呼び出す CloudFormation アクション。以下のいずれかのアクションモードを選択します。

- [Create or replace a change set (変更セットの作成または置換)] は、変更セットが存在しない場合は送信したスタック名とテンプレートに基づいて作成します。変更セットがある場合は、CloudFormation それを削除して新しいものを作成します。
- [Create or update a stack (スタックの作成または更新)] は、指定されたスタックが存在しない場合、スタックを作成します。スタックが存在する場合、CloudFormation はそのスタックを更新します。既存のスタックを更新するには、このアクションを使用します。CodePipeline はスタックを置き換えません。
- [Delete a stack (スタックの削除)] は、スタックを削除します。存在しないスタックを指定した場合は、アクションはスタックを削除せずに正常に終了します。
- [Execute a change set (変更セットの実行)] は、変更セットを実行します。
- [Replace a failed stack (失敗したスタックの置換)] は、指定されたスタックが存在しない場合、スタックを作成します。スタックが存在しており、失敗状態の場合 (ROLLBACK_COMPLETE、ROLLBACK_FAILED、CREATE_FAILED、DELETE_FAILED または UPDATE_ROLLBACK_FAILED として報告されている場合)、CloudFormation はそのスタックを削除して新しいスタックを作成します。スタックが失敗状態ではない場合は、CloudFormation はそれを更新します。失敗したスタックをリカバリーまたはトラブルシューティングせずに置き換えるには、このアクションを使用します。通常、このモードはテスト用に選択されます。

スタック名

既存のスタックまたは作成するスタックに関連付けられている名前のことを指します。この名前は、作成しているスタックの AWS リージョン内で一意であることが必要です。

Note

スタック名には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字はアルファベット文字である必要があります。また、128 文字より長くすることはできません。

変更セット名

既存の変更セットの名前、または指定されたスタック用に作成する新しい変更セットの名前です。

テンプレート

CloudFormation テンプレートファイルの場所です。形式は ***ArtifactName::TemplateFileName*** です。

テンプレート構成

テンプレート構成ファイルの場所です。形式は

ArtifactName::TemplateConfigurationFileName です。テンプレート構成ファイルには、テンプレートのパラメータ値、スタックポリシー、およびタグを含めることができます。機密情報 (パスワードなど) を含める場合は、このファイルへのアクセスを制限します。詳細については、「[CloudFormation アーティファクト](#)」を参照してください。

機能

特定のリソースを含むスタックの場合、CloudFormation がそれらのリソースを作成または更新する場合があることの明示的に承認。例えば、スタックテンプレートに AWS Identity and Access Management (IAM) リソースが含まれている場合、CAPABILITY_IAM を指定する必要があります。詳しくは、「[CreateStack](#) API 操作リクエストパラメータ」を参照してください。

スタックテンプレートに IAM リソースがある場合は、このプロパティを指定する必要があります。

複数のキャパシティーを指定できます。

ロール名

CloudFormation が指定されたスタックのリソースで操作する際に引き受ける IAM サービスロールの名前。

出力ファイル名

[Advanced] セクションで、CodePipeline が指定されたアクションを実行した後に[出力アーティファクト](#)に追加する出力ファイル名 (CreateStackOutput.json など) を指定できます。出力アーティファクトには、CloudFormation テンプレートの Outputs セクションの内容を含む JSON ファイルが含まれています。

名前を指定しない場合、CodePipeline は出力アーティファクトを生成しません。

パラメータの上書き

パラメータはテンプレートで定義されます。パラメータを使用すると、スタックの作成時または更新時にカスタム値を入力できます。テンプレート構成ファイルのテンプレートパラメータ値を上書きする JSON オブジェクトを指定できます。すべてのパラメータ名がスタックテンプレートに存在する必要があります。詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

Note

ParameterOverrides プロパティに保存することができる JSON オブジェクトのサイズは、最大 1 キロバイトに制限されています。

パラメータ値のほとんどは、テンプレート構成ファイルを使用して指定することをお勧めします。パラメータの上書き機能を使用して、動的パラメータ値のみを指定します。動的パラメータは、パイプラインを実行するまでは不明です。

次の例では、パラメータオーバーライド関数を使用して ParameterName パラメータの値を定義します。この関数は、CodePipeline 入力アーティファクトから値を取得します。パラメータオーバーライド関数の詳細については、「[CodePipeline パイプラインでのパラメーターオーバーライド関数の使用](#)」を参照してください。

```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
    "ParamName"]}
}
```

構成プロパティ (JSON オブジェクト)

ステージのアクションで CloudFormation をプロバイダーとして指定する場合は、Configuration プロパティ内で次のプロパティを定義します。AWS CLI、CodePipeline API、CloudFormation テンプレートに JSON オブジェクトを使用します。例については、「[チュートリアル: テストおよび本稼働スタック用のパイプラインを構築する](#)」および「[CloudFormation 構成プロパティのリファレンス](#)」を参照してください。

ActionMode

CodePipeline が関連ステージを処理する際に呼び出す CloudFormation アクション。次のアクションモードのいずれか 1 つのみを指定します。

- CHANGE_SET_EXECUTE は変更セットを実行します。
- CHANGE_SET_REPLACE は、変更セットが存在しない場合、指定されたスタック名とテンプレートに基づいて変更セットを作成します。変更セットがある場合は、CloudFormation それを削除して新しいものを作成します。

- `CREATE_UPDATE` は、指定されたスタックが存在しない場合、そのスタックを作成します。スタックが存在する場合、CloudFormation はそのスタックを更新します。既存のスタックを更新するには、このアクションを使用します。CodePipeline はスタックを置き換えません。
- `DELETE_ONLY` は、スタックを削除します。存在しないスタックを指定した場合は、アクションはスタックを削除せずに正常に終了します。
- `REPLACE_ON_FAILURE` は、指定されたスタックが存在しない場合、スタックを作成します。スタックが存在しており、失敗状態の場合 (`ROLLBACK_COMPLETE`、`ROLLBACK_FAILED`、`CREATE_FAILED`、`DELETE_FAILED` または `UPDATE_ROLLBACK_FAILED` として報告されている場合)、CloudFormation はそのスタックを削除して新しいスタックを作成します。スタックが失敗状態ではない場合は、CloudFormation はそれを更新します。失敗したスタックをリカバリーまたはトラブルシューティングせずに自動的に置き換えるには、このアクションを使用します。通常、このモードはテスト用に選択されます。

このプロパティは必須です。

Capabilities

特定のリソースを含むスタックの場合、CloudFormation がそれらのリソースを作成または更新する場合があることの明示的に承認。例えば、スタックテンプレートに AWS Identity and Access Management (IAM) リソースが含まれている場合、`CAPABILITY_IAM` を指定する必要があります。詳しくは、「[CreateStack](#) API 操作リクエストパラメータ」を参照してください。

このプロパティは条件付きです。スタックテンプレートに IAM リソースがある場合は、このプロパティを指定する必要があります。

複数の機能を指定できます。次の例では、`CAPABILITY_IAM` および `CAPABILITY_AUTO_EXPAND` プロパティをテンプレートに追加します。

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
  ChangeSetName: pipeline-changeset
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
"configuration": {
  "ActionMode": "CHANGE_SET_REPLACE",
  "Capabilities": "CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND",
  "ChangeSetName": "pipeline-changeset",
  "RoleArn": "CloudFormation_Role_ARN",
  "StackName": "my-pipeline-stack",
  "TemplateConfiguration": "my-pipeline-stack::template-configuration.json",
  "TemplatePath": "my-pipeline-stack::template-export.yml"
}
```

ChangeSetName

既存の変更セットの名前、または指定されたスタック用に作成する新しい変更セットの名前です。

このプロパティは `CHANGE_SET_REPLACE` および `CHANGE_SET_EXECUTE` アクションモードでは必須です。他のすべてのアクションモードでは、このプロパティは無視されます。

OutputFileName

`CreateStackOutput.json` など、出力ファイルの名前。CodePipeline は指定されたアクションを実行した後、[出力アーティファクト](#) にファイルを追加します。出力アーティファクトには、CloudFormation テンプレートの `Outputs` セクションの内容を含む JSON ファイルが含まれています。

このプロパティはオプションです。名前を指定しない場合、CodePipeline は出力アーティファクトを生成しません。

ParameterOverrides

パラメータはテンプレートで定義されます。パラメータを使用すると、スタックの作成時または更新時にカスタム値を入力できます。テンプレート構成ファイルのテンプレートパラメータ値を上書きする JSON オブジェクトを指定できます。すべてのパラメータ名がスタックテンプレートに存在する必要があります。詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

次の例では、`InstanceType` および `KeyName` パラメータの上書きをテンプレートに追加します。

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_NAMED_IAM
  ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"InstanceType": "t2.small","KeyName": "my-keypair"}'
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
"configuration": {
  "ActionMode": "CHANGE_SET_REPLACE",
  "Capabilities": "CAPABILITY_NAMED_IAM",
  "ChangeSetName": "pipeline-changeset",
  "ParameterOverrides": "{\"InstanceType\": \"t2.small\", \"KeyName\": \"my-keypair\"}",
  "RoleArn": "CloudFormation_Role_ARN",
  "StackName": "my-pipeline-stack",
  "TemplateConfiguration": "my-pipeline-stack::template-configuration.json",
  "TemplatePath": "my-pipeline-stack::template-export.yml"
}
```

Note

ParameterOverrides プロパティに保存できる JSON オブジェクトの最大サイズは 1 KB です。

パラメータ値のほとんどは、テンプレート構成ファイルを使用して指定することをお勧めします。パラメータの上書き機能を使用して、動的パラメータ値のみを指定します。動的パラメータ値は、パイプラインを実行するまでは不明です。

次の例では、パラメータオーバーライド関数を使用して ParameterName パラメータの値を定義します。この関数は、CodePipeline 入力アーティファクトから値を取得します。パラメータオーバーライド関数の詳細については、「[CodePipeline パイプラインでのパラメーターオーバーライド関数の使用](#)」を参照してください。


```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
    "ParamName"]}
}
```

このプロパティはオプションです。

RoleArn

CloudFormation がスタックのリソースを操作する際に引き受ける IAM サービスロールの Amazon リソースネーム (ARN) です。

このプロパティは、CREATE_UPDATE、REPLACE_ON_FAILURE、DELETE_ONLY、および CHANGE_SET_REPLACE アクションモードに必要です。RoleArn は、変更セットを実行するときには適用されません。変更セットの作成に CodePipeline を使用しない場合は、変更セットまたはスタックにロールが関連付けられていることを確認します。

StackName

既存のスタックの名前、または作成するスタックの名前です。

このプロパティはすべてのアクションモードで必須です。

TemplateConfiguration

TemplateConfiguration はテンプレート構成ファイルです。このアクションへの入力アーティファクトにファイルを含めます。テンプレート構成ファイル名は以下の形式に従います。

Artifactname::TemplateConfigurationFileName

Artifactname は、CodePipeline に表示される入力アーティファクト名です。たとえば、アーティファクト名 SourceArtifact と test-configuration.json ファイル名を持つソースステージでは、次の例に示すような TemplateConfiguration の名前が作成されます。

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

テンプレート構成ファイルには、テンプレートのパラメーター値およびスタックポリシーを含めることができます。機密情報 (パスワードなど) を含める場合は、このファイルへのアクセスを制限します。設定ファイルのテンプレートの例については、「[CloudFormation アーティファクト](#)」を参照してください。

このプロパティはオプションです。

TemplatePath

TemplatePath は CloudFormation テンプレートファイルを表します。このアクションへの入力アーティファクトにファイルを含めます。ファイル名の形式は次のとおりです:

Artifactname::TemplateFileName

Artifactname は、CodePipeline に表示される入力アーティファクト名です。たとえば、アーティファクト名 SourceArtifact と template.yaml ファイル名を持つソースステージでは、次の例に示すような TemplatePath の名前が作成されます。

```
"TemplatePath": "SourceArtifact::template.yaml"
```

このプロパティは CREATE_UPDATE、REPLACE_ON_FAILURE および CHANGE_SET_REPLACE アクションモードでは必須です。他のすべてのアクションモードでは、このプロパティは無視されます。

関連情報

これらのパラメータを利用する際に役立つ関連リソースは以下の通りです。

- CodePipeline の CloudFormation アクションパラメータの詳細については、「AWS CodePipeline ユーザーガイド」の「[アクション設定リファレンスの CloudFormation デプロイ](#)」を参照してください。
- アクションプロバイダーごとのテンプレート値における例の場合 (owner フィールドまたは configuration フィールドなど)、「AWS CodePipeline ユーザーガイド」の「[アクション構造リファレンス](#)」を参照してください。
- サンプルのパイプラインスタックテンプレートを YAML または JSON 形式でダウンロードするには、「AWS CodePipeline ユーザーガイド」の「[Tutorial: Create a pipeline with CloudFormation](#)」を参照してください。

CloudFormation アーティファクト

CodePipeline では、CodePipeline パイプラインを実行しながらアーティファクトのタスクを実行します。CloudFormation の場合、アーティファクトはスタックテンプレートファイル、テンプレート構成ファイル、または両方を含むことができます。CodePipeline はこれらのアーティファクトを使用して CloudFormation スタックとチェンジセットと連動します。

ソースリポジトリとして Amazon Simple Storage Service (Amazon S3) を使用する場合は、テンプレートおよびテンプレート構成ファイルを 1 つのファイルに zip 化してから S3 バケットにアップロードする必要があります。GitHub や AWS CodeCommit などその他のリポジトリの場合は、アーティファクトを圧縮しないでアップロードします。詳細については、「AWS CodePipeline ユーザーガイド」の「[Create a pipeline, stages, and actions](#)」を参照してください。

リポジトリには、必要なだけファイルを追加できます。たとえば、同じテンプレートに、テスト構成と本稼働構成の 2 つの異なる構成を含めることもできます。

このトピックでは、各アーティファクトタイプを説明します。

トピック

- [スタックテンプレートファイル](#)
- [テンプレート構成ファイル](#)
- [関連情報](#)

スタックテンプレートファイル

スタックテンプレートファイルは、CloudFormation がプロビジョニングおよび設定するリソースを定義します。これらのファイルは、CloudFormation を使用してスタックを作成または更新する際に使用するテンプレートファイルと同じです。YAML または JSON 形式のテンプレートを使用できます。テンプレートの詳細については、「」を参照してください[CloudFormation テンプレートセクション](#)

テンプレート構成ファイル

テンプレート構成ファイルは、テンプレートのパラメータ値、[スタックポリシー](#)、タグを指定できる JSON 形式のテキストファイルです。これらの構成ファイルを使用して、スタックにパラメーター値またはスタックポリシーを指定します。指定するパラメーター値はすべて、関連するテンプレートで宣言する必要があります。

このファイルに機密情報 (パスワードなど) を含める場合は、このファイルへのアクセスを制限します。たとえば、アーティファクトを S3 バケットにアップロードする場合、[S3 バケットポリシー](#)または[ユーザーポリシー](#)を使用してアクセスを制限します。

構成ファイルを作成するには、次の形式を使用します。

```
{
```

```
"Parameters" : {
  "NameOfTemplateParameter" : "ValueOfParameter",
  ...
},
"Tags" : {
  "TagKey" : "TagValue",
  ...
},
"StackPolicy" : {
  "Statement" : [
    StackPolicyStatement
  ]
}
}
```

以下の例では、TestEC2Key パラメータに KeyName を指定し、Department タグを追加して、その値を Marketing に設定します。続いて、リソースを削除する更新を除くすべての更新アクションを許可するスタックポリシーを追加します。

```
{
  "Parameters" : {
    "KeyName" : "TestEC2Key"
  },
  "Tags" : {
    "Department" : "Marketing"
  },
  "StackPolicy" : {
    "Statement" : [
      {
        "Effect" : "Allow",
        "NotAction" : "Update:Delete",
        "Principal": "*",
        "Resource" : "*"
      }
    ]
  }
}
```

関連情報

これらのパラメータを利用する際に役立つ関連リソースは以下の通りです。

- CodePipeline の CloudFormation アクションパラメータの詳細については、「AWS CodePipeline ユーザーガイド」の「[アクション設定リファレンスの CloudFormation デプロイ](#)」を参照してください。
- アクションプロバイダーごとのテンプレート値における例の場合 (Owner フィールドまたは configuration フィールドなど)、「AWS CodePipeline ユーザーガイド」の「[アクション構造リファレンス](#)」を参照してください。
- サンプルのパイプラインスタックテンプレートを YAML または JSON 形式でダウンロードするには、「AWS CodePipeline ユーザーガイド」の「[Tutorial: Create a pipeline with CloudFormation](#)」を参照してください。

CodePipeline パイプラインでのパラメーターオーバーライド関数の使用

CodePipeline ステージでは、CloudFormation アクションの[パラメーターオーバーライド](#)を指定できます。パラメーターオーバーライドを使用すると、テンプレート構成ファイルの値をオーバーライドするテンプレートパラメータ値を指定できます。CloudFormation では、動的な値 (パイプラインが実行されるまで不明な値) を指定できる関数を提供しています。

トピック

- [Fn::GetArtifactAtt](#)
- [Fn::GetParam](#)
- [関連情報](#)

Fn::GetArtifactAtt

Fn::GetArtifactAtt 関数は、入力アーティファクトから属性の値を取得します。たとえば、アーティファクトが保存されている S3 バケット名などです。この関数を使用して、アーティファクトのファイル名や Amazon S3 バケット名など、アーティファクトの属性を指定します。

パイプラインを実行すると、CodePipeline はファイルをコピーしてパイプラインのアーティファクトストア (S3 バケット) に書き込みます。CodePipeline は、アーティファクトストアにファイル名を生成します。これらのファイル名は、パイプラインが実行されるまでは不明です。

例えば、パイプラインで、CodePipeline が AWS Lambda 関数ソースコードをアーティファクトストアにコピーするソースステージがあるとします。次のステージでは、関数を作成する

CloudFormation テンプレートがありますが、CloudFormation が関数を作成するにはそのファイル名が必要です。Fn::GetArtifactAtt 関数を使用して、正確な S3 バケット名およびファイル名を渡す必要があります。

構文

以下の構文を使用してアーティファクトの属性値を取得します。

```
{ "Fn::GetArtifactAtt" : [ "artifactName", "attributeName" ] }
```

artifactName

入力アーティファクトの名前。このアーティファクトを、関連付けられたアクションの入力として宣言する必要があります。

attributeName

値の取得元となるアーティファクト属性の名前。各アーティファクト属性の詳細については、後述の属性セクションを参照してください。

例

以下のパラメータオーバーライドは、LambdaFunctionSource アーティファクトの S3 バケット名およびファイル名を取得して、BucketName および ObjectKey パラメータを指定します。この例では、たとえばソースステージの一部として、CodePipeline が Lambda 関数のソースコードをコピーし、アーティファクトとして保存したとします。

```
{
  "BucketName" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "BucketName"] },
  "ObjectKey" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "ObjectKey"] }
}
```

属性

アーティファクトの次の属性を取得できます。

BucketName

アーティファクトが保存されている S3 バケットの名前。

ObjectKey

CodePipeline によって生成されるアーティファクトを含む .zip ファイルの名前 (1ABCyZZ.zip など)。

URL

アーティファクトの Amazon Simple Storage Service (Amazon S3) URL (https://s3.us-west-2.amazonaws.com/artifactstorebucket-yivczw8jma0c/test/TemplateSo/1ABCyZZ.zip など)。

Fn::GetParam

Fn::GetParam 関数はキーと値のペアからの値を JSON 形式ファイルで返します。この JSON ファイルをアーティファクトに含める必要があります。

この関数を使用して CloudFormation スタックから出力値を取得し、別のアクションの入力として使用します。例えば、CloudFormation アクションの出力ファイル名を指定すると、CodePipeline は出力を JSON ファイルで保存して、それを出力アーティファクトの .zip ファイルに追加します。Fn::GetParam 関数を使用して出力値を取得し、それを別のアクションの入力として使用します。

構文

以下の構文を使用してキーと値のペアから値を取得します。

```
{ "Fn::GetParam" : [ "artifactName", "JSONFileName", "keyName" ] }
```

artifactName

アーティファクトの名前です。関連アクションの入力アーティファクトとして含める必要があります。

JSONFileName

アーティファクトに含まれる JSON ファイルの名前。

keyName

値の取得元となるキーの名前。

例

次の例では、パラメーターオーバーライドで `Fn::GetParam` 関数を使用する方法を示します。

構文

以下のパラメーターオーバーライドは、`WebSiteURL` アーティファクトに存在する URL ファイルから `stack-output.json` キーの値を取得することで、`WebStackOutput` パラメーターを指定します。

```
{
  "WebSiteURL" : { "Fn::GetParam" : ["WebStackOutput", "stack-output.json", "URL"]}
}
```

CloudFormation テンプレートスニペット

以下の CodePipeline パイプラインからの CloudFormation テンプレートスニペットは、スタックの出力を渡す方法を示します。これらのスニペットは、パイプライン定義の 2 つのステージを示します。最初のステージは、スタックを作成し、出力を `StackAOutput` アーティファクトの `TestOutput.json` ファイルに保存します。これらの値は、`OutputFileName` および `OutputArtifacts` プロパティによって指定されます。

ステージのソース入力アーティファクトの名前は `TemplateSource` です。スタックテンプレートのファイル名は `teststackA.yaml` で、構成ファイルの名前は `test-configuration.json` です。どちらのステージにおいても、これらの値は `TemplateConfiguration` と `TemplatePath` プロパティに対して次のように指定されます。

```
TemplateConfiguration: TemplateSource::test-configuration.json
TemplatePath: TemplateSource::teststackA.yaml
```

Exampleスタック A ステージの作成

```
- Name: CreateTestStackA
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
```



```

Configuration:
  ActionMode: CREATE_UPDATE
  Capabilities: CAPABILITY_IAM
  OutputFileName: TestOutput.json
  RoleArn: !GetAtt [CFNRole, Arn]
  StackName: StackA
  TemplateConfiguration: TemplateSource::test-configuration.json
  TemplatePath: TemplateSource::teststackA.yaml
InputArtifacts:
  - Name: TemplateSource
OutputArtifacts:
  - Name: StackAOutput
RunOrder: '1'

```

後続のステージで、スタック B はスタック A からの出力を使用します。例では、ParameterOverrides プロパティに Fn::GetParam 関数を使用して StackBInputParam パラメーターを指定します。結果の値は、StackAOutputName キーに関連付けられる値です。

Exampleスタック B ステージの作成

```

- Name: CreateTestStackB
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CREATE_UPDATE
        Capabilities: CAPABILITY_IAM
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: StackB
        TemplateConfiguration: TemplateSource::test-configuration.json
        TemplatePath: TemplateSource::teststackB.yaml
        ParameterOverrides: |
          {
            "StackBInputParam" : { "Fn::GetParam" : ["StackAOutput", "TestOutput.json",
"StackAOutputName"]}
          }
      InputArtifacts:
        - Name: TemplateSource
        - Name: StackAOutput

```

```
RunOrder: '1'
```

関連情報

これらのパラメータを利用する際に役立つ関連リソースは以下の通りです。

- CodePipeline の CloudFormation アクションパラメータの詳細については、「AWS CodePipeline ユーザーガイド」の「[アクション設定リファレンスの CloudFormation デプロイ](#)」を参照してください。
- アクションプロバイダーごとのテンプレート値における例の場合 (owner フィールドまたは configuration フィールドなど)、「AWS CodePipeline ユーザーガイド」の「[アクション構造リファレンス](#)」を参照してください。
- サンプルのパイプラインスタックテンプレートを YAML または JSON 形式でダウンロードするには、「AWS CodePipeline ユーザーガイド」の「[Tutorial: Create a pipeline with CloudFormation](#)」を参照してください。

CloudFormation テンプレートリファレンスガイド

CloudFormation テンプレートの作成に使用できるリソースとコンポーネントの詳細については、「[CloudFormation テンプレートリファレンスガイド](#)」を参照してください。このガイドはテンプレートに関する技術リファレンスであり、CloudFormation ユーザーガイドの開始方法に関するコンテンツを補完します。

CloudFormation テンプレートリファレンスガイドには、以下に関する詳細が記載されています。

- リソースタイプとそのプロパティ
- DeletionPolicy およびその他のリソースレベルの属性
- Ref や Sub などの組み込み関数
- 変換 (CloudFormation マクロ)
- EC2 インスタンス設定のヘルパースクリプト
- 機械読み取り可能なリソースタイプ定義

このガイドは、以下を反映するために定期的に更新されます。

- 新しいリソースタイプ
- プロパティの変更
- 新しい戻り値
- ドキュメントの更新

「CloudFormation」のセキュリティ

AWS でのクラウドセキュリティは最優先事項です。AWS のユーザーは、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利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ - AWS はAWS Cloud で AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は使用するサービスを安全に提供します。[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。CloudFormation に適用するコンプライアンスプログラムの詳細については、「[コンプライアンスプログラムによる対象範囲内の AWS 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 - ユーザーの責任は、使用する AWS のサービスに応じて異なります。また、お客様は、データの機密性、お客様の会社の要件、および適用される法律および規制など、その他の要因についても責任を負います。

このドキュメントは、CloudFormation を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために CloudFormation を設定する方法を示します。また、CloudFormation リソースのモニタリングや保護に役立つ、その他 AWS サービスの使用方法についても説明します。

トピック

- [CloudFormation スタックの削除を防止する](#)
- [スタックのリソースが更新されないようにする](#)
- [CloudFormation でのデータ保護](#)
- [AWS Identity and Access Management で CloudFormation アクセスを制御する](#)
- [AWS CloudTrail による CloudFormation API コールのログ記録](#)
- [CloudFormation 内のインフラストラクチャセキュリティ](#)
- [CloudFormation での耐障害性](#)
- [のコンプライアンス検証CloudFormation](#)
- [CloudFormation での構成と脆弱性の分析](#)

- [CloudFormation のセキュリティのベストプラクティス](#)
- [インターフェイスエンドポイントを使用した CloudFormation へのアクセス \(AWS PrivateLink\)](#)

CloudFormation スタックの削除を防止する

スタックの削除保護を有効にして、スタックが誤って削除されるのを防ぐことができます。削除保護を有効にした状態でスタックを削除しようとする、削除は失敗し、ステータスを含め、スタックが変更されることはありません。終了保護は、スタック作成時に有効にすることができます。デフォルトでは、スタックの削除保護は無効です。スタックの削除保護は、任意のステータスで設定できます ([DELETE_IN_PROGRESS] または [DELETE_COMPLETE] を除く)。

スタックで削除保護を有効または無効にすると、そのスタックに属するネストされたスタックにも同じ選択が適用されます。ネストされたスタックの削除保護を直接有効または無効にすることはできません。削除保護が有効になっているスタックに属するネストされたスタックを直接削除しようとする、そのオペレーションは失敗し、ネストされたスタックは変更されません。

ただし、ネストされたスタックが削除されるスタック更新を行うと、CloudFormation は状況に応じて、ネストされたスタックを削除します。

削除保護は、ロールバックの無効化とは異なります。削除保護がスタックの削除の試行に適用されるのに対し、ロールバックの無効化は、スタックの作成に失敗した際の自動ロールバックに適用されます。

スタック作成時に削除保護を有効にするには

[Create stack] (スタックの作成) ウィザードの [Specify stack options] (スタックオプションの指定) ページの [Advanced options] (詳細オプション) で、[Termination Protection] (削除保護) セクションを展開し、[Enable] (有効化) を選択します。詳細については、「[スタックオプションを設定する](#)」を参照してください。

既存のスタックの削除保護を有効または無効にするには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. 変更するスタックを選択します。

 Note

スタック名の横に [ネスト] と表示されている場合、そのスタックは、ネストされたスタックです。ネストされたスタックが属するルートスタックの削除保護のみ変更することができます。

4. スタックの詳細ペインで、[Stack actions (スタックアクション)]、[Edit termination protection (削除保護の編集)] の順に選択します。

CloudFormation によって、[Edit termination protection] (削除保護の編集) ダイアログボックスが表示されます。

5. [有効化] または [無効化] を選択し、[保存] を選択します。

ネストされたスタックの削除保護を有効または無効にするには

スタック名の横に [ネスト] と表示されている場合、そのスタックは、ネストされたスタックです。ネストされたスタックが属するルートスタックの削除保護のみ変更することができます。ルートスタックの削除保護を変更するには

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (<https://console.aws.amazon.com/cloudformation>) を開きます。
2. 画面上部にあるナビゲーションバーで、AWS リージョン を選択します。
3. 変更するネストされたスタックを選択します。
4. [Stack info (スタック情報)] ペインの [概要] セクションで、[ルートスタック] としてリストされているスタック名を選択します。

CloudFormation により、ルートスタックのスタックの詳細が表示されます。

5. [Stack actions (スタックアクション)]を選択し、[Edit Termination Protection (削除保護の編集)] を選択します。

CloudFormation によって、[Edit termination protection] (削除保護の編集) ダイアログボックスが表示されます。

6. [有効化] または [無効化] を選択し、[保存] を選択します。

コマンドラインを使用して終了保護を有効または無効にするには

[update-termination-protection](#) コマンドを使用します。

スタックの削除保護を変更するユーザーの制御

スタックの削除保護を有効または無効にするに

は、cloudformation:UpdateTerminationProtection アクションの権限が必要です。たとえば、以下のポリシーでは、ユーザーは、スタックの削除保護を有効または無効にすることができます。

CloudFormation でアクセス許可を指定する方法については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。

Exampleスタックの削除保護を変更するアクセス許可を付与するサンプルポリシー。

JSON

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

スタックのリソースが更新されないようにする

スタックを作成するとき、すべてのリソースですべての更新アクションが許可されます。デフォルトでは、スタック更新アクセス権限を持つユーザーであれば誰でもスタック内のすべてのリソースを更新できます。更新処理中、リソースの中断や完全な置き換えが必要になった場合は、新しい物理 ID やまったく新しいストレージが使用される結果になります。スタックポリシーを使用すると、スタックの更新中にスタックのリソースが意図せずに更新または削除されるのを防止できます。スタックポリシーは、指定したリソースに対して実行できる更新アクションを定義する JSON ドキュメントです。

スタックポリシーを設定すると、デフォルトでスタック内のすべてのリソースが保護されます。特定のリソースでの更新を許可するには、スタックポリシーでこれらのリソースに対して Allow ス

ステートメントを明示的に指定します。1 つのスタックに定義できるスタックポリシーは 1 つのみですが、1 つのポリシー内で複数のリソースを保護することができます。スタックポリシーは、スタックの更新を試みるすべての CloudFormation ユーザーに適用されます。異なるスタックポリシーを異なるユーザーに関連付けることはできません。

スタックポリシーは、スタックの更新時のみ適用されます。AWS Identity and Access Management (IAM) ポリシーのようなアクセスコントロールは提供しません。特定のスタックリソースを誤って更新しないように、スタックポリシーはフェイルセーフメカニズムとしてのみ使用してください。AWS リソースまたはアクションへのアクセスを制御するには、IAM を使用します。

トピック

- [スタックポリシーの例](#)
- [スタックポリシーの定義](#)
- [スタックポリシーの設定](#)
- [保護されたリソースの更新](#)
- [スタックポリシーの変更](#)
- [スタックポリシーのその他の例](#)

スタックポリシーの例

次のスタックポリシーの例は、ProductionDatabase リソースの更新が禁止されます。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    },
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```


スタックポリシーを設定すると、デフォルトですべてのリソースが保護されます。すべてのリソースの更新を許可するには、すべてのリソースですべてのアクションを許可する Allow ステートメントを追加します。Allow ステートメントではすべてのリソースが指定されますが、ProductionDatabase 論理 ID を持つリソースに関しては明示的な Deny ステートメントが優先されます。この Deny ステートメントにより、置き換えや削除など、すべての更新アクションが ProductionDatabase リソースで禁止されます。

Principal 要素が必要ですが、ワイルドカード (*) のみサポートされています。つまり、ステートメントはすべての [プリンシパル](#) に適用されます。

Note

スタックの更新中、CloudFormation は、更新された他のリソースに依存するリソースを自動的に更新します。例えば、CloudFormation は、更新されたリソースを参照するリソースを更新します。CloudFormation は自動的に更新されたリソースに物理的な変更 (リソースの ID など) は行いませんが、これらのリソースにスタックポリシーが関連付けられている場合は、リソースを更新する許可が必要になります。

スタックポリシーの定義

スタックを作成するとき、スタックポリシーは設定されないため、すべてのリソースですべての更新アクションが許可されます。スタックリソースを更新アクションから保護するには、スタックポリシーを定義し、そのポリシーをスタックに設定します。スタックポリシーは JSON ドキュメントで、CloudFormation ユーザーが実行できる CloudFormation スタック更新アクションと、そのアクションが適用されるリソースを定義します。スタックポリシーを含むテキストファイルを指定するか、入力することにより、スタックの作成時にスタックポリシーを設定します。スタックにスタックポリシーを設定した場合、明示的に許可されていないすべての更新がデフォルトで拒否されます。

スタックポリシーは、Effect、Action、Principal、Resource、Condition の 5 つの要素によって定義します。スタックポリシーの構文を次の擬似コードに示します。

```
{
  "Statement" : [
    {
      "Effect" : "Deny_or-Allow",
      "Action" : "update_actions",
      "Principal" : "*",
      "Resource" : "LogicalResourceId/resource_logical_ID",
```

```
"Condition" : {  
  "StringEquals_or_StringLike" : {  
    "ResourceType" : [resource_type, ...]  
  }  
}  
}  
]  
}
```

Effect

指定されたリソースでの指定されたアクションを拒否するかまたは許可するかを決定します。Deny または Allow のみ指定できます (など)。

```
"Effect" : "Deny"
```

Important

スタックポリシーに重複ステートメントが含まれる場合 (リソースでの更新を許可するステートメントと拒否するステートメントの両方)、常に Deny ステートメントが Allow ステートメントより優先されます。リソースが必ず保護されるようにするには、そのリソースに対して Deny ステートメントを使用します。

Action

拒否または許可する更新アクションを指定します。

Update:Modify

変更の適用中にリソースで中断が発生しない更新アクションまたはなんらかの中断が発生する更新アクションを指定します。すべてのリソースで物理 ID が保持されます。

Update:Replace

リソースが再作成される更新アクションを指定します。CloudFormation は、指定された更新を使用して新しいリソースを作成してから、古いリソースを削除します。リソースが再作成されるため、新しいリソースの物理 ID が変更される場合があります。

Update>Delete

リソースが削除される更新アクションを指定します。スタックテンプレートから完全にリソースを削除する更新の場合は、このアクションを使用する必要があります。

更新:*

すべての更新アクションを指定します。アスタリスクは、すべての更新アクションを表すワイルドカードです。

次の例では、replace アクションと delete アクションのみを指定しています。

```
"Action" : ["Update:Replace", "Update:Delete"]
```

1 つのアクションを除くすべての更新アクションを許可するには、NotAction を使用します。たとえば、Update:Delete を除くすべての更新アクションを許可するには、次の例に示すように NotAction を使用します。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "NotAction" : "Update:Delete",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Principal

Principal 要素は、ポリシーを適用するエンティティを指定します。この要素は必須ですが、ワイルドカード (*) のみサポートされています。つまり、ポリシーはすべての[プリンシパル](#)に適用されます。

リソース

ポリシーを適用するリソースの論理 ID を指定します。リソースのタイプを指定するには、Condition 要素を使用します。

1 つのリソースを指定するには、論理 ID を使用します。例:

```
"Resource" : ["LogicalResourceId/myEC2instance"]
```

論理 ID には、ワイルドカードを使用できます。たとえば、すべての関連リソースに共通の論理 ID プレフィックスを使用している場合、ワイルドカードを使用してすべてのリソースを指定できます。

```
"Resource" : ["LogicalResourceId/CriticalResource*"]
```

これらのリソースに対して Not 要素を使用することもできます。たとえば、1 つを除くすべてのリソースに対する更新を許可するには、NotResource 要素を使用してそのリソースを保護します。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "NotResource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

スタックポリシーを設定した場合、明示的に許可されていないすべての更新が拒否されます。ProductionDatabase リソース以外のリソースに対する更新をすべて許可することで、ProductionDatabase リソースに対する更新を拒否します。

条件

ポリシーを適用するリソースタイプを指定します。論理 ID を指定するには、Resource 要素を使用します。

次の例に示すように、EC2 インスタンスや RDS DB インスタンスのようなリソースタイプを指定できます。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Principal" : "*",
      "Action" : "Update:*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "ResourceType" : ["AWS::EC2::Instance", "AWS::RDS::DBInstance"]
        }
      }
    }
  ],
}
```

```
{
  "Effect" : "Allow",
  "Principal" : "*",
  "Action" : "Update:*",
  "Resource" : "*"
}
]
```

Allow ステートメントはすべてのリソースに更新アクセス権限を付与し、Deny ステートメントは EC2 および RDS DB インスタンスに対する更新を拒否します。Deny ステートメントは、すべてのアクションを常にオーバーライドします。

リソースタイプでワイルドカードを使用できます。たとえば、次の例に示すように、ワイルドカードを使用して、すべての Amazon EC2 リソース (— インスタンス、セキュリティグループ、サブネット— など) に対する更新アクセス許可を拒否できます。

```
"Condition" : {
  "StringLike" : {
    "ResourceType" : ["AWS::EC2::*"]
  }
}
```

ワイルドカードを使用する場合は、StringLike 条件を使用する必要があります。

スタックポリシーの設定

スタックを作成する際にスタックのポリシーを適用するには、コンソールまたは AWS CLI を使用します。AWS CLI を使用して、既存のスタックにスタックポリシーを適用することもできます。スタックのポリシーを適用すると、スタックからそれを削除することはできません。ただし、AWS CLI を使用して変更することはできます。

スタックポリシーは、スタックの更新を試みるすべての CloudFormation ユーザーに適用されます。異なるスタックポリシーを異なるユーザーに関連付けることはできません。

ポリシーの記述方法については、「[スタックポリシーの定義](#)」を参照してください。

スタックの作成時にスタックポリシーを設定するには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。

2. 画面の上部のナビゲーションバーで、スタックを作成する AWS リージョンを選択します。
3. [CloudFormation Stacks] ページで、[スタックの作成] を選択します。
4. [スタックの作成] ウィザードの [Configure stack options (スタックオプションの設定)] ページで、[詳細設定] セクションを展開し、[スタックのポリシー] を選択します。
5. 次のようにスタックポリシーを指定します。
 - コンソールにポリシーを直接作成するには、[Enter stack policy (スタックポリシーの入力)] を選択し、テキストフィールドにスタックポリシーを直接入力します。
 - 別のファイルで定義されたポリシーを使用するには、[ファイルのアップロード] を選択し、[ファイルの選択] を選択して、スタックポリシーを含むファイルを選択します。

スタックの作成時にスタックポリシーを設定するには (AWS CLI)

- [create-stack](#) コマンドを使用する際、--stack-policy-body オプションを使用して変更されたポリシーを入力するか、--stack-policy-url オプションを使用してポリシーを含むファイルを指定します。

既存のスタックでスタックポリシーを設定するには (AWS CLI のみ)

- [set-stack-policy](#) コマンドを使用する際、--stack-policy-body オプションを使用して変更されたポリシーを入力するか、--stack-policy-url オプションを使用してポリシーを含むファイルを指定します。

Note

既存のスタックにポリシーを追加するには、CloudFormation の [SetStackPolicy](#) アクションに対する許可が必要です。

保護されたリソースの更新

保護されたリソースを更新するには、スタックポリシーをオーバーライドする一時ポリシーを作成し、それらのリソースでの更新を許可します。スタックを更新するときにオーバーライドポリシーを指定します。オーバーライドポリシーによって、スタックポリシーが永続的に変更されるわけではありません。

保護されたリソースを更新するには、CloudFormation の [SetStackPolicy](#) アクションを使用する許可が必要です。CloudFormation 許可の設定については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。

 Note

スタックの更新中、CloudFormation は、更新された他のリソースに依存するリソースを自動的に更新します。例えば、CloudFormation は、更新されたリソースを参照するリソースを更新します。CloudFormation は自動的に更新されたリソースに物理的な変更 (リソースの ID など) は行いませんが、これらのリソースにスタックポリシーが関連付けられている場合は、リソースを更新する許可が必要になります。

保護されたリソースを更新するには (コンソール)

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソール を開きます。
2. 更新するスタックを選択し、[Stack actions (スタックアクション)]、[スタックの更新] の順に選択します。
3. スタックテンプレートを変更していない場合は、[現在のテンプレートの使用] を選択し、[次へ] をクリックします。テンプレートを変更した場合は、[Replace current template (現在のテンプレートを置換)] を選択し、[Specify template (テンプレートの指定)] セクションで更新されたテンプレートの場所を指定します。
 - ローカルコンピュータに保存されているテンプレートの場合は、[テンプレートファイルをアップロード] を選択します。[ファイルの選択] を選択してファイルの場所に移動し、ファイルを選択して、[次へ] をクリックします。
 - Amazon S3 バケットに保存されているテンプレートの場合は、[Amazon S3 URL] を選択します。テンプレートの URL を入力するか貼り付けて、[次へ] をクリックします。

バージョニング対応バケットにテンプレートがある場合、URL に `?versionId=version-id` を付加することによってテンプレートの具体的なバージョンを指定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バージョニングが有効なバケット内のオブジェクトの使用](#)」を参照してください。
4. テンプレートにパラメータが含まれている場合、[Specify stack details (スタックの詳細を指定)] ページで、パラメータ値を入力または変更し、[次へ] を選択します。

CloudFormation は、スタックで現在設定されている値を各パラメータに入力します (NoEcho 属性で宣言されたパラメータを除く)。[既存の値の使用] を選択することで、それらのパラメータに現在の値を使用できます。

NoEcho を使用して機密情報をマスクする方法、また動的なパラメータを使用してシークレットを管理する方法の詳細については、「[テンプレートに認証情報を埋め込まない](#)」ベストプラクティスを参照してください。

5. オーバーライドスタックポリシーを指定します。
 - a. [Configure stack options (スタックオプションの設定)] ページの [詳細オプション] セクションで、[スタックのポリシー] を選択します。
 - b. [ファイルのアップロード] を選択します。
 - c. [ファイルの選択] をクリックし、オーバーライドする側のスタックポリシーを含むファイルに移動するか、ポリシーを入力します。
 - d. [次へ] を選択します。

オーバーライドポリシーでは、更新する保護されたリソースを対象として Allow ステートメントを指定する必要があります。たとえば、すべての保護されたリソースを更新するには、すべての更新を許可する一時オーバーライドポリシーを指定します。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Note

CloudFormation がオーバーライドポリシーを適用するのは、この更新の実行中のみです。オーバーライドポリシーによって、スタックポリシーが永続的に変更されるわけで

はありません。スタックのポリシーを変更するには、「[スタックポリシーの変更](#)」を参照してください。

6. 送信したスタック情報と変更を確認します。

パラメータ値やテンプレートの URL が適切かなど、正しい情報を送信したことを確認します。テンプレートに IAM リソースが含まれる場合は、[I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択して、テンプレート内の IAM リソースを使用することを指定します。詳細については、「[CloudFormation テンプレートでの IAM リソースの承認](#)」を参照してください。

[変更のプレビュー] セクションで、期待される変更のすべてを CloudFormation が実行することを確認します。例えば、追加、削除、または変更する予定のリソースを、CloudFormation が追加、削除、および変更することを確認します。CloudFormation は、スタックの変更セットを作成することによって、このプレビューを生成します。詳細については、「[変更セットを使用して CloudFormation スタックを更新する](#)」を参照してください。

7. 変更内容が適切であることを確認できたら、[更新] をクリックします。

 Note

この時点で、変更セットを表示するオプションを使用すると、提案された更新をより徹底的に確認することもできます。これを行うには、[Update] (更新) の代わりに [View change set] (変更セットを表示) をクリックしてください。CloudFormation により、更新に基づいて生成された変更セットが表示されます。スタックの更新を実行する準備ができたら、[実行] をクリックします。

CloudFormation により、スタックの [Stack details (スタックの詳細)] ページが表示されます。これで、スタックのステータスは UPDATE_IN_PROGRESS になりました。CloudFormation によるスタックの更新が正常に完了したら、スタックの状況が UPDATE_COMPLETE に設定されます。

スタックの更新が失敗した場合、CloudFormation によって変更が自動的にロールバックされ、スタックの状況が UPDATE_ROLLBACK_COMPLETE に設定されます。

保護されたリソースを更新するには (AWS CLI)

- [update-stack](#) コマンドを使用する際、`--stack-policy-during-update-body` オプションを使用して変更されたポリシーを入力するか、`--stack-policy-during-update-url` オプションを使用してポリシーを含むファイルを指定します。

 Note

CloudFormation がオーバーライドポリシーを適用するのは、この更新の実行中のみです。オーバーライドポリシーによって、スタックポリシーが永続的に変更されるわけではありません。スタックのポリシーを変更するには、「[スタックポリシーの変更](#)」を参照してください。

スタックポリシーの変更

追加のリソースを保護するか、リソースから保護を削除するには、スタックポリシーを変更します。たとえば、保護するデータベースをスタックに追加するときは、そのデータベースの Deny ステートメントをスタックポリシーに追加します。ポリシーを変更するには、[SetStackPolicy](#) アクションを使用するアクセス許可が必要です。

AWS CLI を使用して、スタックポリシーを変更します。

スタックポリシーを変更するには (AWS CLI)

- [set-stack-policy](#) コマンドを使用する際、`--stack-policy-body` オプションを使用して変更されたポリシーを入力するか、`--stack-policy-url` オプションを使用してポリシーを含むファイルを指定します。

スタックポリシーは削除できません。すべてのリソースからすべて保護を削除するには、すべてのリソースに対するすべてのアクションを明示的に許可するようにポリシーを変更します。次のポリシーは、すべてのリソースですべての更新を許可します。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*"
    }
  ]
}
```

```
    "Resource" : "*"
  }
]
}
```

スタックポリシーのその他の例

次のポリシー例では、すべてのスタックリソースの更新を禁止する方法、特定のリソースの更新を禁止する方法、特定のタイプの更新を禁止する方法を示します。

すべてのスタックリソースの更新を禁止する

すべてのスタックリソースの更新を禁止するため、次のポリシーでは、すべてのリソースのすべての更新アクションに対して Deny ステートメントを指定しています。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

1 つのリソースの更新を禁止する

次のポリシーでは、MyDatabase という論理 ID を持つデータベースについて、すべての更新アクションを拒否します。また、Allow ステートメントにより、他のすべてのスタックリソースにおいてすべての更新アクションを許可します。Deny ステートメントは常に許可アクションをオーバーライドするため、Allow ステートメントは MyDatabase リソースには適用されません。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyDatabase"
    },
  ],
```

```
{
  "Effect" : "Allow",
  "Action" : "Update:*",
  "Principal" : "*",
  "Resource" : "*"
}
```

デフォルト拒否を使用することで、前の例と同じ結果が得られます。スタックポリシーを設定すると、CloudFormation は明示的に許可されていない更新をすべて拒否します。次のポリシーでは、デフォルトで拒否されている ProductionDatabase リソースを除くすべてのリソースの更新が許可されます。

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal" : "*",
      "NotResource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

Important

デフォルト拒否の使用にはリスクがあります。ポリシーの他の場所で Allow ステートメントを使用している場合 (ワイルドカードを使用する Allow ステートメントなど)、知らないうちに意図しないリソースに更新アクセス権限を付与する可能性があります。ただし、明示的な拒否はすべての許可アクションをオーバーライドするため、Deny ステートメントを使用することでリソースが保護されることを保証できます。

特定のリソースタイプについてすべてのインスタンスの更新を禁止する

次のポリシーは、RDS DB インスタンスリソースタイプについて、すべての更新アクションを拒否します。また、Allow ステートメントにより、他のすべてのスタックリソースにおいてすべての更新アクションを許可します。Allow ステートメントは常に許可アクションをオーバーライドするため、Deny ステートメントは RDS DB インスタンスリソースには適用されません。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "ResourceType" : ["AWS::RDS::DBInstance"]
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

特定のインスタンスについて置き換え更新を防止する

次のポリシーは、MyInstance という論理 ID を持つインスタンスの置き換えが発生する更新を拒否します。また、Allow ステートメントにより、他のすべてのスタックリソースにおいてすべての更新アクションを許可します。Deny ステートメントは常に許可アクションをオーバーライドするため、Allow ステートメントは MyInstance リソースには適用されません。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:Replace",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyInstance"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

```
    }  
  ]  
}
```

ネストされたスタックの更新を禁止する

以下のポリシーは、CloudFormation スタックリソースタイプ (ネストされたスタック) でのすべての更新アクションを拒否します。また、Allow ステートメントにより、他のすべてのスタックリソースにおいてすべての更新アクションを許可します。Deny ステートメントは常に許可アクションを上書きするため、CloudFormation リソースに Allow ステートメントは適用されません。

```
{  
  "Statement" : [  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*",  
      "Condition" : {  
        "StringEquals" : {  
          "ResourceType" : ["AWS::CloudFormation::Stack"]  
        }  
      }  
    },  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"   
    }  
  ]  
}
```

CloudFormation でのデータ保護

[AWS 責任共有モデル](#) は、AWS CloudFormation 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、AWS クラウド のすべて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があります。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、[データプライバシーに関](#)

[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。

データを保護するため、「AWS アカウント」認証情報を保護し、「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セットアップ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。
- AWS CloudTrail で API とユーザーアクティビティロギングを設定します。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 のサービス 内のすべてのデフォルトセキュリティコントロールに加え、AWS 暗号化ソリューションを使用します。
- 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 を使用して「AWS」にアクセスする際に FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-3](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK で CloudFormation または他の AWS のサービスを使用する場合も同様です。タグ、または名前に使用される自由形式のテキストフィールドに入力されるデータは、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。

保管中の暗号化

AWS の責任共有モデルに従って、CloudFormation は保存時に暗号化されたデータを保存します。アカウントに保存されているデータの暗号化ポリシーとストレージポリシーを設定するのは顧客の責任です。たとえば、S3 バケットまたは SNS トピックに保存されているテンプレートやその他のデータについては、保存時の暗号化を有効にすることをお勧めします。同様に、顧客は CloudFormation によってプロビジョニングされるデータストレージシステムの暗号化設定を定義します。

送信中の暗号化

CloudFormation は、責任共有モデルに基づくサービス通信に暗号化されたチャネルを使用します。

インターネットトラフィックのプライバシー

CloudFormation サービス通信は、デフォルトでリージョンまたはアベイラビリティゾーン間で安全に暗号化されます。

AWS Identity and Access Management で CloudFormation アクセスを制御する

AWS Identity and Access Management (IAM) を使えば、IAM ユーザーを作成して、AWS アカウントの特定のリソースへの、各ユーザーのアクセスを制御することができます。IAM を使用すると、スタックテンプレートの表示、スタックの作成、スタックの削除など、ユーザーが CloudFormation で実行できるアクションを制御できます。

CloudFormation の特定のアクションに加えて、各ユーザーが利用できる AWS サービスおよびリソースを管理できます。この方法で、ユーザーが CloudFormation の使用時にアクセスできるリソースを管理できます。たとえば、Amazon EC2 インスタンスを作成できるユーザー、データベースインスタンスを終了できるユーザ、または VPC を更新できるユーザを指定できます。CloudFormation を使用してそれらのアクションを実行するときは、いつでも同じアクセス許可が適用されます。

CloudFormation にアクセスできるユーザーを制御するときは、次のセクションの情報を使用します。また、テンプレートでの IAM リソースの作成を承認する方法、EC2 インスタンスで実行中のアプリケーションに必要なアクセス許可を付与する方法、AWS 環境のセキュリティ強化のために一時的なセキュリティ認証情報を使用する方法についても説明します。

CloudFormation の IAM アイデンティティベースのポリシーの定義

CloudFormation へのアクセスを許可するには、必要な API アクションを呼び出すためのアクセス許可を、IAM アイデンティティ (ユーザーやロールなど) に付与する IAM ポリシーを作成し、これを割り当てる必要があります。

IAM 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、またアクションを許可または拒否する条件を指定できます。CloudFormation は、特定のアクション、リソース、条件キーをサポートしています。

IAM を初めて使用する場合は、まず IAM JSON ポリシーの要素を理解します。詳細については、「IAM ユーザーガイド」の「[IAM JSON ポリシー要素のリファレンス](#)」を参照してください。IAM ポリシーの作成方法については、IAM ドキュメントの「[はじめてのカスタマー管理ポリシーの作成とアタッチ](#)」のチュートリアルを参照してください。

トピック

- [CloudFormation のポリシーアクション](#)
- [CloudFormation のコンソール固有のアクション](#)
- [CloudFormation のポリシーリソース](#)
- [CloudFormation のポリシー条件キー](#)

CloudFormation のポリシーアクション

IAM ポリシーステートメントの Action 要素では、CloudFormation で提供されている API アクションを指定できます。アクション名には、接頭辞として小文字の文字列 `cloudformation:` を含める必要があります。例えば、`cloudformation:CreateStack`、`cloudformation:CreateChangeSet`、および `cloudformation:UpdateStack` のようになります。

単一のステートメントで複数のアクションを指定するには、次のようにカンマで区切ります。

```
"Action": [ "cloudformation:action1", "cloudformation:action2" ]
```

ワイルドカードを使用して複数のアクションを指定することもできます。例えば、`Get` という単語で始まるアクション名すべてを次のように指定できます。

```
"Action": "cloudformation:Get*"
```

`cloudformation` サービスプレフィックスに関連付けられたアクションの一覧については、「Service Authorization Reference」の「[Actions, resources, and condition keys for CloudFormation](#)」および「[Actions, resources, and condition keys for AWS クラウドコントロール API](#)」を参照してください。

例

以下は、CloudFormation スタックを表示するためのアクセス許可を付与するアクセス許可ポリシーの例です。

Example 1: スタックを表示するためのアクセス許可を付与するポリシーの例

JSON

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

スタックを作成または削除するユーザーには、スタックテンプレートに基づく追加のアクセス許可が必要です。例えば、テンプレートに Amazon SQS キューが記述されている場合、次のポリシー例に示すように、ユーザーは CloudFormation アクションと Amazon SQS アクションの両方に対するアクセス許可を持っている必要があります。

Example 2: スタックアクションの作成と表示、およびすべての Amazon SQS アクションを付与するサンプルポリシー

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "sqs:*",
      "cloudformation:CreateStack",
      "cloudformation:DescribeStacks",
      "cloudformation:DescribeStackEvents",
      "cloudformation:DescribeStackResources",
      "cloudformation:GetTemplate",
      "cloudformation:ValidateTemplate"
    ]
  }]
}
```

```
    ],  
    "Resource": "*"    
  }  
}
```

CloudFormation のコンソール固有のアクション

CloudFormation のコンソールを使用するユーザーには、AWS Command Line Interface または CloudFormation API に必要なアクセス許可に加えて、追加の許可が必要です。これらの追加のアクセス許可は、Amazon S3 バケットへのテンプレートのアップロードや、AWS 固有のパラメータタイプのドロップダウンリストなど、コンソールに固有の機能をサポートしています。

次のすべてのアクションで、すべてのリソースに対するアクセス許可を付与します。その際、特定のスタックまたはバケットに制限しないようにしてください。

次のアクションは、CloudFormation コンソールでのみ使用されるため、API リファレンスには記載されていません。このアクションにより、ユーザーは Amazon S3 バケットにテンプレートをアップロードできるようになります。

- `cloudformation:CreateUploadBucket`

ユーザーがテンプレートをアップロードするときは、次の Amazon S3 のアクセス許可も必要になります。

- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetObject`
- `s3:CreateBucket`

AWS 固有のパラメータタイプを持つテンプレートで、パラメータ ドロップダウン リストの値を確認するには、ユーザーは、対応する記述 API コールを行うためのアクセス許可が必要です。例えば、これらのパラメータタイプをテンプレートで使用する場合は次のアクセス許可が必要です。

- `ec2:DescribeKeyPairs` – `AWS::EC2::KeyPair::KeyName` パラメータタイプに必要。
- `ec2:DescribeSecurityGroups` – `AWS::EC2::SecurityGroup::Id` パラメータタイプに必要。
- `ec2:DescribeSubnets` – `AWS::EC2::Subnet::Id` パラメータタイプに必要。

- `ec2:DescribeVpcs – AWS::EC2::VPC::Id` パラメータタイプに必要。

AWS 固有のパラメータタイプに関する詳細は、「[CloudFormation が提供するパラメータタイプを使用して、実行時に既存のリソースを指定する](#)」を参照してください。

CloudFormation のポリシーリソース

IAM ポリシーステートメントで、Resource 要素は、ステートメントがカバーするオブジェクトを指定します。CloudFormation では、各 IAM ポリシーステートメントは、ユーザーが Amazon リソースネーム (ARN) を使用して指定したリソースに適用されます。具体的 ARN 形式はリソースによって異なります。

CloudFormation リソースのタイプとその ARN のリストの一覧は、「Service Authorization Reference」の「[Resource types defined by CloudFormation](#)」を参照してください。各リソースの ARN を使って指定できるアクションについては、「[Actions defined by CloudFormation](#)」を参照してください。

次のポリシー例に示すように、特定のスタックのアクションを指定できます。ARN を指定するには、*placeholder text* をリソース固有の情報に置き換えます。

Example 1: 指定したスタックに対するスタックアクションの削除と更新を拒否するサンプルポリシー

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:us-east-1:111122223333:stack/MyProductionStack/*"
    }
  ]
}
```

上記のポリシーでは、スタック名の最後にワイルドカードを使用することで、フルスタック ID (arn:aws:cloudformation:*region*:*account-id*:stack/*MyProductionStack*/*abc9dbf0-43c2-11e3-a6e8-50fa526be49c* など) とスタック名 (*MyProductionStack* など) の両方に基づいて、スタックの削除と更新を拒否します。

変更セットの作成を AWS::Serverless トランスフォームに許可するには、次のポリシーに示すように、arn:aws:cloudformation:*region*:aws:transform/Serverless-2016-10-31 リソースレベルのアクセス許可を含めます。

Example 2: 指定した変換に対して変更セットの作成アクションを許可するポリシーの例

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateChangeSet"
      ],
      "Resource": "arn:aws:cloudformation:us-east-1:aws:transform/Serverless-2016-10-31"
    }
  ]
}
```

CloudFormation のポリシー条件キー

IAM ポリシーのステートメントでは、必要に応じて、ポリシーを有効にするタイミングを制御する条件を指定できます。たとえば、特定のテンプレート URL を指定したときにのみ ユーザーにスタックの作成を許可するように、ポリシーを定義できます。CloudFormation 固有の条件や AWS 全体に対する、ポリシーが有効でなくなるタイミングを指定する DateLessThan などの条件を定義できます。AWS 全体の条件のリストについては、「IAM ユーザーガイド」の「[IAM ポリシー要素のリファレンス](#)」を参照してください。

 Note

aws:SourceIp AWS 全体の条件を使用しないでください。CloudFormation はリクエストの送信元 IP アドレスではなく、独自の IP アドレスを使用してリソースをプロビジョニングします。例えば、スタックを作成する際に、CloudFormation は、Amazon EC2 インスタンスを起動したり Amazon S3 バケットを作成したりするために、CreateStack コールや create-stack コマンドを実行した IP アドレスではなく、その IP アドレスからリクエストを行います。

次のリストは、CloudFormation 固有の条件要件について説明しています。これらの条件は、ユーザーがスタックを作成または更新する場合にのみ適用されます。

cloudformation:ChangeSetName

ポリシーに関連付ける CloudFormation 変更セットの名前。この条件を使用して、ユーザーがどの変更セットを実行、または削除するかを制御します。

cloudformation:ImportResourceTypes

ポリシーに関連付ける、AWS::EC2::Instance などのテンプレートリソースタイプ。この条件を使用して、ユーザーがスタックにリソースをインポートするときに、操作できるリソースタイプを指定します。この条件は、ユーザーが ResourcesToImport パラメータで宣言したリソースタイプに対して確認されます。このパラメータは、現在のところ AWS CLI および API リクエストでのみサポートされています。このパラメータを使用する場合は、インポートオペレーション中にユーザーが制御するリソースタイプをすべて指定する必要があります。ResourcesToImport パラメータの詳細については、「AWS CloudFormation API リファレンス」の「[CreateChangeSet](#) アクション」を参照してください。

可能な ResourcesToImport の一覧については、「[リソースタイプのサポート](#)」を参照してください。

3 つの部分からなるリソース命名規則を使用して、組織全体のすべてのリソースから個々のリソースタイプまで、ユーザーが操作できるリソースタイプを指定します。

organization::*

所定の組織のすべてのリソースタイプを指定します。

organization::service_name::*

所定の組織内で指定されたサービスのすべてのリソースタイプを指定します。

organization::service_name::resource_type

特定のリソースタイプを指定します。

例:

`AWS::*`

サポートされているすべての AWS リソースを指定します。

`AWS::service_name::*`

固有の AWS サービスでサポートされているすべてのリソースを指定します。

`AWS::service_name::resource_type`

`AWS::EC2::Instance` (すべての EC2 インスタンス) などの特定の AWS リソースタイプを指定します。

cloudformation:ResourceTypes

ポリシーに関連付ける `AWS::EC2::Instance` などのテンプレートリソース。この条件を使用して、ユーザーがスタックを作成または更新したときに、操作できるリソースタイプを指定します。この条件は、ユーザーが `ResourceTypes` パラメータで宣言したリソースタイプに対して確認されます。このパラメータは、現在のところ AWS CLI および API リクエストでのみサポートされています。このパラメータを使用する場合、ユーザーはそのテンプレートにあるすべてのリソースタイプを指定する必要があります。`ResourceTypes` パラメータの詳細については、「AWS CloudFormation API リファレンス」の「[CreateStack](#) アクション」を参照してください。

リソースタイプのリストについては、「[CloudFormation テンプレートリファレンスガイド](#)」を参照してください。

3 つの部分からなるリソース命名規則を使用して、組織全体のすべてのリソースから個々のリソースタイプまで、ユーザーが操作できるリソースタイプを指定します。

*organization::**

所定の組織のすべてのリソースタイプを指定します。

*organization::service_name::**

所定の組織内で指定されたサービスのすべてのリソースタイプを指定します。

organization::service_name::resource_type

特定のリソースタイプを指定します。

例:

`AWS::*`

サポートされているすべての AWS リソースを指定します。

`AWS::service_name::*`

固有の AWS サービスでサポートされているすべてのリソースを指定します。

`AWS::service_name::resource_type`

`AWS::EC2::Instance` (すべての EC2 インスタンス) などの特定の AWS リソースタイプを指定します。

`Alexa::ASK::*`

Alexa スキルキットですべてのリソースタイプを指定します。

`Alexa::ASK::Skill`

個々の [Alexa::ASK::Skill](#) リソースタイプを指定します。

`Custom::*`

すべてのカスタムリソースを指定します。

詳細については、「[カスタムリソースを使用してカスタムプロビジョニングロジックを作成する](#)」を参照してください。

`Custom::resource_type`

特定のカスタムリソースタイプを指定します。

詳細については、「[カスタムリソースを使用してカスタムプロビジョニングロジックを作成する](#)」を参照してください。

`cloudformation:RoleARN`

ポリシーに関連付ける IAM サービスロールの Amazon リソースネーム (ARN)。この条件を使用して、ユーザーがスタックまたは変更セットを操作するとき、どのサービスロールを使用できるかを制御します。

`cloudformation:StackPolicyUrl`

ポリシーに関連付ける Amazon S3 スタックポリシーの URL。ユーザーがスタックの作成または更新アクション時にスタックに関連付けることができるスタックポリシーを制御するには、この条件を使用します。スタックポリシーの詳細については、「[スタックのリソースが更新されないようにする](#)」を参照してください。

Note

アップロードしたスタックポリシーを使用した場合にのみユーザーがスタックを作成または更新できるようにするには、それらのユーザーのために S3 バケットを読み取り専用を設定します。

cloudformation:TemplateUrl

ポリシーに関連付ける Amazon S3 テンプレートの URL。この条件を使用して、スタックを作成または更新したときにユーザーが使用できるテンプレートを指定します。

Note

アップロードしたテンプレートを使用した場合にのみユーザーがスタックを作成または更新できるようにするには、それらのユーザーのために S3 バケットを読み取り専用を設定します。

Note

同じ名前の API パラメータには、次の CloudFormation 固有の条件が適用されます。

- `cloudformation:ChangeSetName`
- `cloudformation:RoleARN`
- `cloudformation:StackPolicyUrl`
- `cloudformation:TemplateUrl`

例えば、`cloudformation:TemplateUrl` は、`CreateStack`、`UpdateStack`、および `CreateChangeSet` API の `TemplateUrl` パラメータにのみ適用されます。

アクセス制御に条件キーを使用する IAM ポリシーの例については、「[CloudFormation の IAM アイデンティティベースのポリシーの例](#)」を参照してください。

CloudFormation テンプレートでの IAM リソースの承認

スタックを作成する前に、CloudFormation によってテンプレートが検証されます。検証中、CloudFormation は作成する可能性のある IAM リソースについてテンプレートをチェックしま

す。IAM リソース (フルアクセス権を持つユーザーなど) は、AWS アカウント の任意のリソースにアクセスして変更できます。したがって、拡大した権限で意図していないリソースを作成しないように、進める前に各 IAM に関連付けられた権限を確認することをお勧めします。実行されたことを確認するには、スタックを作成する前に、CloudFormation に指定された機能を付与して、テンプレートにそれらのリソースを含めることを承認する必要があります。

CloudFormation コンソール、AWS Command Line Interface (AWS CLI)、または API を使用して、CloudFormation テンプレートの機能を承認できます。

- CloudFormation コンソールで、[スタックを作成] または [スタックを更新] ウィザードの [スタック オプションの設定] ページで [I acknowledge that this template may create IAM resources (このテンプレートが IAM リソースを作成する可能性を認識しています)] を選択します。
- AWS CLI で、[create-stack](#) および [update-stack](#) コマンドを使用する場合は、`--capabilities` オプションに `CAPABILITY_IAM` または `CAPABILITY_NAMED_IAM` 値を指定します。テンプレートに IAM リソースがある場合、いずれかの機能を指定できます。テンプレートの IAM リソースにカスタム名がある場合、`CAPABILITY_NAMED_IAM` を指定する必要があります。
- API で [CreateStack](#) と [UpdateStack](#) アクションを使用する際には、`Capabilities.member.1=CAPABILITY_IAM` または `Capabilities.member.1=CAPABILITY_NAMED_IAM` を指定します。テンプレートに IAM リソースがある場合、いずれかの機能を指定できます。テンプレートの IAM リソースにカスタム名がある場合、`CAPABILITY_NAMED_IAM` を指定する必要があります。

Important

テンプレートにカスタム名の IAM リソースが含まれている場合、同じテンプレートを再利用する複数のスタックを作成しないでください。IAM リソースはアカウント内でグローバルに一意でなければなりません。同じテンプレートを使用して異なるリージョンで複数のスタックを作成する場合、それぞれのスタックは一意なリソースを持つのではなく、同じ IAM リソースを共有する可能性があります。スタック間でリソースを共有すると、回復不可能な意図しない結果をまねく可能性があります。たとえば、1 つのスタックの共有された IAM リソースを削除または更新すると、意図せずに他のスタックのリソースの変更することになります。

Amazon EC2 インスタンスで実行されるアプリケーションの認証情報の管理

Amazon EC2 インスタンスで実行されるアプリケーションで、Amazon S3 バケットや DynamoDB テーブルなどの AWS リソースにリクエストを送信する必要がある場合、そのアプリケーションでは AWS セキュリティ認証情報が必要になります。ただし、起動するすべてのインスタンスにおける長期的なセキュリティ認証情報を配信して埋め込むことは非常に難しく、潜在的なセキュリティリスクを伴います。IAM ユーザー認証情報のような長期認証情報を使用する代わりに、インスタンスが起動される際に Amazon EC2 インスタンスに関連付けられる IAM ロールを作成することをお勧めします。アプリケーションは Amazon EC2 インスタンスから一時的な認証情報を得ることができます。インスタンスに長期認証情報を埋め込む必要はありません。また、認証情報を管理しやすくするために、複数の Amazon EC2 インスタンスに対して 1 つのロールのみを指定できます。インスタンスごとに個別の認証情報を作成する必要はありません。

ロールでのインスタンスの起動方法を示すテンプレートスニペットについては、「[IAM ロールのテンプレートの例](#)」を参照してください。

Note

一時的なセキュリティ認証情報を使用するインスタンスにあるアプリケーションでは、CloudFormation アクションを呼び出すことができます。ただし、CloudFormation は他の多数の AWS サービスとやり取りするため、使用するすべてのサービスが一時的なセキュリティ認証情報をサポートすることを確認する必要があります。一時的なセキュリティ認証情報を受け入れるサービスのリストについては、IAM ユーザーガイドの「[IAM と連携する AWS サービス](#)」を参照してください。

一時的なアクセス権の付与 (フェデレーションアクセス)

場合によっては、AWS アカウント に対して AWS 認証情報のない一時的なアクセス権をユーザーに付与する必要があります。一時的なアクセス権を付与する場合は、長期認証情報を作成して削除するのではなく、AWS Security Token Service (AWS STS) を使用します。たとえば、IAM ロールを使用できます。1 つの IAM ロールから、多くの一時的なセキュリティ認証情報 (アクセスキー、シークレットアクセスキー、およびセキュリティトークンを含む) をプログラムで作成し、配信することができます。これらの認証情報の有効期限は限られているため、期限が切れた後は AWS アカウントへのアクセスに使用することはできません。また、個々のユーザーに異なるレベルのアクセス許可を

付与するために、複数の IAM ロールを作成することもできます。IAM ロールは、連合 ID やシングルサインオンのようなシナリオに対して有効です。

フェデレーテッドアイデンティティは、複数のシステムで使用できる固有のアイデンティティです。確立されたオンプレミスアイデンティティシステム (LDAP または Active Directory など) を使用している企業ユーザーの場合、そのオンプレミスアイデンティティシステムを使用してすべての認証を処理できます。ユーザーが認証された後で、適切な IAM ユーザーまたは IAM ロールから一時的なセキュリティ認証情報を提供します。例えば、管理者が AWS アカウントへの完全アクセス権を持ち、デベロッパーが CloudFormation スタックでの作業のみのアクセス許可を持つ、管理者ロールおよび開発者ロールを作成できます。管理者は、認証された後で、管理者ロールから一時的なセキュリティ認証情報を取得することが認可されます。一方で、開発者は開発者ロールからのみ一時的なセキュリティ認証情報を取得できます。

また、AWS マネジメントコンソールにフェデレーションユーザーアクセスを付与できます。ユーザーがオンプレミスアイデンティティシステムで認証された後で、AWS マネジメントコンソールへの直接アクセス権を付与する一時的な URL をプログラムによって作成できます。ユーザーは既に認証されているため (シングルサインオン)、その一時的な URL を使用するときには AWS にサインインする必要がありません。また、この URL はユーザーの一時的なセキュリティ認証情報から作成されるため、この認証情報で利用できるアクセス許可によって、AWS マネジメントコンソールでユーザーに付与されるアクセス許可が決まります。

一時的なセキュリティ認証情報を生成するために複数のさまざまな AWS STS API を使用できます。使用する API の詳細については、「IAM ユーザーガイド」の「[AWS STS 認証情報を比較する](#)」を参照してください。

Important

GetFederationToken API から生成された一時的なセキュリティ認証情報を使用する場合は、IAM で作業することはできません。IAM での作業が必要な場合、代わりにロールからの一時的なセキュリティ認証情報を使用します。

CloudFormation は他の多数の AWS サービスとインタラクションします。CloudFormation で一時的なセキュリティ認証情報を使用する場合は、使用するすべてのサービスが一時的なセキュリティ認証情報をサポートすることを確認してください。一時的なセキュリティ認証情報を受け入れるサービスのリストについては、IAM ユーザーガイドの「[IAM と連携する AWS サービス](#)」を参照してください。

詳細については、「IAM ユーザーガイド」の次の関連リソースを参照してください。

- [一時的な認証情報の一般的なシナリオ](#)
- [AWS コンソールへのカスタム ID ブローカーアクセスを有効にする](#)

CloudFormation の IAM アイデンティティベースのポリシーの例

デフォルトでは、ユーザーおよびロールには、CloudFormation リソースを作成または変更するアクセス許可がありません。また、AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。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、ユーザーはロール引き継ぐことができます。詳細については、「[CloudFormation の IAM アイデンティティベースのポリシーの定義](#)」を参照してください。

以下は、1 つ以上の CloudFormation アクションを使用するためのアクセス許可を、許可または拒否する際に使用できるポリシーステートメントの例です。

トピック

- [特定のテンプレート URL を要求する](#)
- [すべての CloudFormation インポートオペレーションを拒否する](#)
- [特定のリソースタイプのインポートオペレーションを許可する](#)
- [スタックテンプレートで IAM リソースを拒否する](#)
- [特定のリソースタイプでスタックの作成を許可する](#)
- [リソース変更 API アクションに基づいてアクセスを制御する](#)
- [リージョンとリソースタイプに基づいてスタックセットのオペレーションを制限する](#)
- [すべての IaC ジェネレーターオペレーションを許可する](#)

特定のテンプレート URL を要求する

次のポリシーは、スタックを作成または更新するときに、<https://s3.amazonaws.com/amzn-s3-demo-bucket/test.template> テンプレート URL の使用のみを許可します。

JSON

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

```
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:UpdateStack"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "cloudformation:TemplateUrl": [
              "https://s3.amazonaws.com/amzn-s3-demo-bucket/
test.template"
            ]
          }
        }
      }
    ]
  }
}
```

すべての CloudFormation インポートオペレーションを拒否する

次のポリシーは、インポートオペレーションを除く、すべての CloudFormation オペレーションの実行を許可します。

JSON

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

```
    "Condition": {
      "ForAnyValue:StringLike": {
        "cloudformation:ImportResourceTypes": [
          "*"
        ]
      }
    }
  ]
}
```

特定のリソースタイプのインポートオペレーションを許可する

次のポリシーは、指定したリソース (以下の例では `AWS::S3::Bucket`) に加えて、すべてのスタックオペレーションをインポートオペレーションを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImport",
      "Effect": "Allow",
      "Action": "cloudformation:*",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEqualsIgnoreCase": {
          "cloudformation:ImportResourceTypes": [
            "AWS::S3::Bucket"
          ]
        }
      }
    }
  ]
}
```

スタックテンプレートで IAM リソースを拒否する

次のポリシーは、スタックの作成を許可しますが、スタックのテンプレートに IAM サービスのリソースが含まれている場合はリクエストを拒否します。このポリシーでは、ユーザーが `ResourceTypes` パラメータ (AWS CLI および API リクエストでのみ使用可能) を指定することも要求されます。このポリシーは、明示的な拒否ステートメントを使用することにより、他のポリシーが追加のアクセス権限を付与している場合、このポリシーが常に有効なままになるようにします (明示的な拒否ステートメントは常に明示的な許可ステートメントより優先されます)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "cloudformation:CreateStack" ],
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [ "cloudformation:CreateStack" ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [ "cloudformation:CreateStack" ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "cloudformation:ResourceTypes": "true"
        }
      }
    }
  ]
}
```


特定のリソースタイプでスタックの作成を許可する

次のポリシーは、1 つ前の例に似ています。このポリシーは、スタックのテンプレートに IAM サービスのリソースが含まれていない場合に、スタックの作成を許可します。さらに、ユーザーが `ResourceTypes` パラメータ (AWS CLI および API リクエストでのみ使用可能) を指定することも要求します。このポリシーはよりシンプルですが、明示的な拒否ステートメントを使用しません。追加のアクセス権限を付与する他のポリシーにより、このポリシーがオーバーライドされる可能性があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "cloudformation:CreateStack" ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringNotLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        },
        "Null": {
          "cloudformation:ResourceTypes": "false"
        }
      }
    }
  ]
}
```

リソース変更 API アクションに基づいてアクセスを制御する

次のポリシーは、リソース変異 API アクションの名前でアクセスをフィルターすることを許可します。これは、IAM ユーザーがスタックまたはスタックセットのタグを追加または削除するために使用できる API を制御するために使用されます。タグの追加または削除に使用されるオペレーションは、条件キーの値として追加する必要があります。次のポリシーは、変更オペレーション `CreateStack` に対する `TagResource` および `UntagResource` アクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "CreateActionConditionPolicyForTagUntagResources",
    "Effect": "Allow",
    "Action": [
      "cloudformation:TagResource",
      "cloudformation:UntagResource"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "cloudformation:CreateAction": [
          "CreateStack"
        ]
      }
    }
  }]
}
```

リージョンとリソースタイプに基づいてスタックセットのオペレーションを制限する

次のポリシーは、サービスマネージド型のスタックセットのアクセス許可を付与します。このポリシーを持つユーザーは、Amazon S3 リソースタイプ (AWS::S3::*) または AWS::SES::ConfigurationSet リソースタイプを含むテンプレートを含むスタックセットに対してのみオペレーションを実行できます。ID が 123456789012 である組織管理アカウントにサインインした場合、ユーザーはまた、ID が **ou-1fsfsrsdsfrewr** である OU をターゲットとするスタックセットに対してのみオペレーションを実行でき、さらに、ID が **987654321012** である AWS アカウントをターゲットとする、ID が stack-set-id であるスタックセットに対してのみオペレーションを実行できます。

スタックセットテンプレートに、ポリシーで指定されているリソース以外のリソースタイプが含まれている場合、またはデプロイターゲットが、対応する管理アカウントとスタックセットのポリシーで指定されているもの以外の OU またはアカウント ID である場合、スタックセットオペレーションは失敗します。

これらのポリシー制限は、スタックセットオペレーションが us-east-1、us-west-2、または eu-west-2 AWS リージョン をターゲットとする場合にのみ適用されます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stackset/*",
        "arn:aws:cloudformation:*:*:type/resource/AWS-S3-*",
        "arn:aws:cloudformation:us-west-2:111122223333:type/resource/AWS-SES-ConfigurationSet",
        "arn:aws:cloudformation:*:111122223333:stackset-target/*/ou-1fsfsrdsfrewr",
        "arn:aws:cloudformation:*:111122223333:stackset-target/stack-set-id/444455556666"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIgnoreCase": {
          "cloudformation:TargetRegion": [
            "us-east-1",
            "us-west-2",
            "eu-west-1"
          ]
        }
      }
    }
  ]
}
```

すべての IaC ジェネレーターオペレーションを許可する

次のポリシーは、IaC ジェネレーターリソースのスキャンとテンプレート管理に関連する CloudFormation アクションへのアクセスを許可します。最初のステートメントは、リソース スキャンの記述、一覧表示、開始を許可します。また、IaC ジェネレーターがリソースと使

用可能なリソースタイプに関する情報を取得できるようにする、必要な追加のアクセス許可 (cloudformation:GetResource、cloudformation:ListResources、cloudformation:ListTypes) へのアクセスも許可します。2 番目のステートメントは、生成されたテンプレートを作成、削除、記述、一覧表示、更新するための完全なアクセス許可を付与します。

また、IaC ジェネレーターでリソースをスキャンするすべてのユーザーに、ターゲットの AWS サービスの読み取りアクセス許可も付与する必要があります。詳細については、「[リソースのスキャンに必要な IAM アクセス許可](#)」を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ResourceScanningOperations",
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeResourceScan",
        "cloudformation:GetResource",
        "cloudformation:ListResources",
        "cloudformation:ListResourceScanRelatedResources",
        "cloudformation:ListResourceScanResources",
        "cloudformation:ListResourceScans",
        "cloudformation:ListTypes",
        "cloudformation:StartResourceScan"
      ],
      "Resource": "*"
    },
    {
      "Sid": "TemplateGeneration",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateGeneratedTemplate",
        "cloudformation>DeleteGeneratedTemplate",
        "cloudformation:DescribeGeneratedTemplate",
        "cloudformation:GetResource",
        "cloudformation:GetGeneratedTemplate",
        "cloudformation:ListGeneratedTemplates",
        "cloudformation:UpdateGeneratedTemplate"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}  
]  
}
```

AWS CloudFormation の AWS マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースに対してアクセス許可を提供するように設計されているため、ユーザー、グループ、ロールへのアクセス権の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることにご注意ください。これは、すべての AWS ユーザーが使用できるようになるのを避けるためです。ユースケースに固有の[カスタマー管理ポリシー](#)を定義して、アクセス許可を絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されているアクセス許可を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、役割) に影響します。新しい AWS サービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: AWSCloudFormationFullAccess

ユーザー、グループおよびロールに AWSCloudFormationFullAccess をアタッチできます。

このポリシーは、CloudFormation アクションとリソースへのフルアクセスを許可するアクセス許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- cloudformation - プリンシパルが、すべてのリソースに対してすべての CloudFormation アクションを実行できるようにします。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンスガイド」の「[AWSCloudFormationFullAccess](#)」を参照してください。

AWS マネージドポリシー: AWSCloudFormationReadOnlyAccess

ユーザー、グループおよびロールに `AWSCloudFormationReadOnlyAccess` をアタッチできます。

このポリシーは、CloudFormation リソースとアクションへの読み取り専用アクセスを許可するアクセス許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `cloudformation` – プリンシパルがスタックの説明、リソースの一覧表示、テンプレートの表示などの読み取り専用の CloudFormation アクションを実行できるようにしますが、スタックの作成、更新、削除は許可しません。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンスガイド」の「[AWSCloudFormationReadOnlyAccess](#)」を参照してください。

CloudFormation の AWS マネージドポリシーへの更新

CloudFormation の AWS マネージドポリシーへの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページの変更に関する自動通知については、CloudFormation ドキュメントの履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AWSCloudFormationReadOnlyAccess – 既存のポリシーへの更新	CloudFormation は、バッチ記述オペレーションの <code>cloudformation:BatchDescribe*</code> アクションを許可する新しいアクセス許可を追加しました。	2026 年 1 月 30 日
AWSCloudFormationReadOnlyAccess – 既存のポリシーへの更新	CloudFormation は、スタックドリフト検出機能の <code>cloudformation:Detect*</code> アクションを許可する	2019 年 11 月 13 日

変更	説明	日付
	新しいアクセス許可を追加しました。	
AWSCloudFormationReadOnlyAccess – 既存のポリシーへの更新	CloudFormation は、cloudformation:EstimateTemplateCost 、cloudformation:Get* 、および cloudformation:ValidateTemplate アクションを許可する新しいアクセス許可を追加しました。	2017 年 11 月 2 日
AWSCloudFormationFullAccess – 新規ポリシー	CloudFormation は、CloudFormation のアクションとリソースへのフルアクセスを提供する新しい AWS マネージドポリシーを追加しました。	2019 年 7 月 26 日
AWSCloudFormationReadOnlyAccess – 既存のポリシーへの更新	CloudFormation は、スタックドリフト検出の cloudformation:DetectStackDrift および cloudformation:DetectStackResourceDrift アクションを許可する新しいアクセス許可を追加しました。	2019 年 2 月 6 日
AWSCloudFormationReadOnlyAccess – 新規ポリシー	CloudFormation は、CloudFormation アクションとリソースへの読み取り専用アクセスを提供する新しい AWS マネージドポリシーを追加しました。	2015 年 2 月 6 日

変更	説明	日付
CloudFormation が変更の追跡を開始しました	CloudFormation が AWS マネージドポリシーの変更の追跡を開始しました。	2015 年 2 月 6 日

CloudFormation サービスロール

サービスロールは、ユーザーに代わってスタック内のリソースを呼び出すことを CloudFormation に許可する AWS Identity and Access Management (IAM) ロールです。スタックリソースを作成、更新、または削除することを CloudFormation に許可する IAM ロールを指定できます。CloudFormation はデフォルトで、スタックオペレーションのためのユーザー認証情報から CloudFormation が生成する一時的なセッションを使用します。サービスロールを指定すると、CloudFormation はロールの認証情報を使用します。

サービスロールを使用して、CloudFormation が実行できるアクションを明示的に指定します。これらは、必ずしもユーザー自身または他のユーザーが実行できるアクションと同じであるとは限りません。例えば、管理者権限がある場合でも、CloudFormation アクセスを制限できるのは Amazon EC2 アクションへのアクセスのみです。

IAM サービスで、サービスロールと権限ポリシーを作成します。サービスロールの作成について詳しくは、「IAM ユーザーガイド」の「[AWS のサービスに権限を委任するロールの作成](#)」を参照してください。ロールを引き受けることのできるサービスとして CloudFormation (cloudformation.amazonaws.com) を指定します。

サービスロールをスタックと関連付けるには、スタックを作成するときにロールを指定します。詳細については、「[スタックオプションを設定する](#)」を参照してください。サービスロールは、コンソールでスタックを更新するとき、または API を使用してスタックを [DeleteStack](#) するときにも変更できます。サービスロールを指定する前に、サービスロールを割り当てるアクセス権限 (iam:PassRole) があるかどうかを確認します。iam:PassRole アクセス権限は、どのロールを使用できるかを指定します。詳細については、「IAM ユーザーガイド」の「[AWS サービスにロールを渡すアクセス許可をユーザーに付与する](#)」を参照してください。

Important

サービスロールを指定すると、CloudFormation はスタックで実行されるすべてのオペレーションに対して、そのロールを常に使用します。スタックの作成後に、スタックにアタッチされたサービスロールを削除することはできません。このスタックで操作を実行する権限を

持つ他のユーザーは、iam:PassRole 権限の有無にかかわらず、このロールを使用できません。ユーザーが持つべきではないアクセス権限がロールに含まれる場合、ユーザーのアクセス権限を非意図的にエスカレーションできてしまいます。ロールが最小特権を付与することを確認します。詳細については、「IAM ユーザーガイド」の「[最小特権アクセス許可を適用する](#)」を参照してください。

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。AWS では、サービス間でのなりすましによって、混乱した代理問題が発生する場合があります。サービス間でのなりすましは、あるサービス (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が操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐため、AWS では、アカウント内のリソースへのアクセス許可が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで「[aws:SourceArn](#)」および「[aws:SourceAccount](#)」のグローバル条件コンテキストキーを使用し、CloudFormation が CloudFormation 拡張機能などの特定リソースに別のサービスを付与する許可を制限することをお勧めします。クロスサービスアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、aws:SourceAccount を使用します。

aws:SourceArn の値は CloudFormation が保存するリソースの ARN であることを確認します。

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 グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合は、aws:SourceArn グローバルコンテキスト条件キーを使用して、ARN の未知部分をワイルドカード (*) で表します。例えば、arn:aws:*cloudformation*:*:123456789012:* です。

aws:SourceArn の値にアカウント ID が含まれていない場合、両方のグローバル条件コンテキストキーを使用して許可を制限する必要があります。

次の例では、CloudFormation で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。

aws:SourceArn および aws:SourceAccount 条件キーを使用する信頼ポリシーの例

レジストリサービスの場合、CloudFormation は AWS Security Token Service (AWS STS) を呼び出してアカウントでのサービスロールを引き受けます。このロールは、[RegisterType](#) オペレーションで ExecutionRoleArn に対して構成され、[LoggingConfig](#) オペレーションで設定された LogRoleArn に対して構成されます。詳細については、「[IAM アクセス許可とパブリック拡張アクセスの信頼ポリシーを使用して実行ロールを設定する](#)」を参照してください。

このロールの信頼ポリシーの例では、条件文を使用して、サービスロールの AssumeRole 機能を、指定されたアカウントの指定された CloudFormation 拡張機能に対するアクションのみに制限します。aws:SourceArn および aws:SourceAccount の条件は個別に評価されます。サービスロールを使用するリクエストでは、両方の条件が満たされている必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "resources.cloudformation.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:us-east-1:123456789012:type/resource/Organization-Service-Resource"
        }
      }
    }
  ]
}
```

追加情報

StackSets で使用されるサービスロールの `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用するポリシーの例については、「[混乱した代理問題を軽減するためにグローバルキーを設定する](#)」を参照してください。

詳細については、「IAM ユーザーガイド」の「[ロールの信頼ポリシーの変更](#)」を参照してください。

転送アクセスセッション (FAS) リクエストとアクセス許可の評価

CloudFormation スタックを作成、更新、削除する場合、ユーザーはオプションで IAM ロール ARN を指定できます。ロールが指定されていない場合、CloudFormation はデフォルトのサービスメカニズムを使用して他の AWS サービスとやり取りします。このシナリオでは、呼び出し元は管理対象のリソースに必要なアクセス許可を持っている必要があります。または、ユーザーが独自の IAM ロールを提供すると、CloudFormation はそのロールを引き受けて、ユーザーに代わってサービスのやり取りを実行します。

ユーザーが IAM ロールを提供するかどうかにかかわらず、CloudFormation はリソースオペレーションごとに新しいスコープが限定された FAS トークンを生成します。その結果、`aws:ViaAWSService` を含む [FAS 関連の条件キー](#) は、両方のシナリオで入力されます。

FAS の使用は、CloudFormation オペレーション中の IAM ポリシーの評価方法に影響します。FAS 関連の条件キーの影響を受けるリソースを含むテンプレートを使用してスタックを作成すると、アクセス許可が拒否される場合があります。

IAM ポリシーの例

次の IAM ポリシーを検討してください。Statement2 は CloudFormation での `AWS::KMS::Key` リソースの作成を一貫して禁止します。スタックオペレーション中に IAM ロールが指定されているかどうかにかかわらず、制限は一貫して適用されます。これは、FAS の使用により、`aws:ViaAWSService` 条件キーが常に `true` に設定されているためです。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
```

```

        "Effect": "Allow",
        "Action": [
            "kms:CreateKey"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Sid": "Statement2",
        "Effect": "Deny",
        "Action": [
            "kms:CreateKey"
        ],
        "Resource": [
            "*"
        ],
        "Condition": {
            "Bool": {
                "aws:ViaAWSService": "true"
            }
        }
    }
}
]
}

```

スタックテンプレートの例

例えば、ユーザーが次のサンプルテンプレートを使用してスタックを作成すると、aws:ViaAWSService は true に設定され、ロールのアクセス許可は FAS ポリシーによって上書きされます。スタックの作成は、CreateKey アクションを拒否する IAM ポリシーの Statement2 の影響を受けます。その結果、アクセス許可拒否エラーが発生します。

```

Resources:
  myPrimaryKey:
    Type: AWS::KMS::Key
    Properties:
      Description: An example multi-Region primary key
      KeyPolicy:
        Version: '2012-10-17'
        Id: key-default-1
        Statement:

```

```
- Sid: Enable IAM User Permissions
  Effect: Allow
  Principal:
    AWS: !Join
      - ''
      - - 'arn:aws:iam::'
        - !Ref AWS::AccountId
        - ':root'
  Action: kms:*
  Resource: '*'
```

FAS の詳細については、「IAM ユーザーガイド」の「[転送アクセスセッション](#)」を参照してください。

Note

ほとんどのリソースはこの動作に従います。ただし、リソースの作成、更新、削除時に予期しない成功または失敗が発生し、IAM ポリシーに FAS 関連の条件キーが含まれている場合、問題のリソースは、この標準的な動作に従わないリソースの小さなサブセットに属している可能性があります。

AWS CloudTrail による CloudFormation API コールのログ記録

CloudFormation は AWS CloudTrail (ユーザー、ロール、CloudFormation の AWS サービスが行ったアクションの記録を提供するサービス) と統合されます。CloudTrail は、CloudFormation のすべての API コールをイベントとしてキャプチャし、これには CloudFormation コンソールからの呼び出しと CloudFormation API へのコード呼び出しが含まれます。証跡を作成する場合は、CloudFormation のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、CloudFormation に対するリクエスト、リクエスト元の IP アドレス、リクエストしたユーザー、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

トピック

- [CloudTrail での CloudFormation 情報](#)
- [CloudFormation ログファイルエントリについて](#)

CloudTrail での CloudFormation 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。CloudFormation でアクティビティが発生するとき、そのアクティビティは、[Event history] (イベント履歴) の他の AWS サービスイベントとともに、CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、「[CloudTrail Event 履歴でのイベントの表示](#)」を参照してください。

CloudFormation のイベントを含め、AWS アカウントのイベントの継続的な記録については、トレイルを作成します。[Trail] (追跡) により、CloudTrail はログファイルを Simple Storage Service (Amazon S3) バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべてのリージョンに証跡が適用されます。証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- [証跡を作成するための概要](#)
- [CloudTrail がサポートされているサービスと統合](#)
- 「[CloudTrail の Amazon SNS 通知の設定](#)」
- [CloudTrail ログファイルを複数のリージョンから受け取る、複数のアカウントから CloudTrail ログファイルを受け取る](#)

すべての CloudFormation アクションは、CloudTrail によってログ記録され、[AWS CloudFormation API リファレンス](#)にドキュメント化されます。たとえば、[CreateStack]、[DeleteStack]、[ListStacks] セクションの呼び出しは、CloudTrail ログファイルにエントリを生成します。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「[CloudTrail userIdentity エlement](#)」を参照してください。

CloudFormation ログファイルエントリについて

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一または複数のログエントリがあります。各イベントは任意の送信元からの単一のリクエストを表し、リクエストされたオペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

CreateStack オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。オペレーションは Alice という名前の IAM ユーザーによって行われました。

Note

入力パラメータのキー名のみがログ記録されます。パラメータ値は記録されません。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHJKLMNOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:02:43Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "CreateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
    "tags": [
      {
        "key": "test",
        "value": "tag"
      }
    ]
  },
}
```

```
{
  "stackName": "my-test-stack",
  "disableRollback": true,
  "parameters": [
    {
      "parameterKey": "password"
    },
    {
      "parameterKey": "securitygroup"
    }
  ]
},
"responseElements": {
  "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/
a38e6a60-b397-11e3-b0fc-08002755629e"
},
"requestID": "9f960720-b397-11e3-bb75-a5b75389b02d",
"eventID": "9bf6cfb8-83e1-4589-9a70-b971e727099b"
}
```

次の例では、Alice が my-test-stack スタックの UpdateStack オペレーションを呼び出したことを示しています。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHijklmnopq",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:04:29Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "UpdateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
    "parameters": [
      {
        "parameterKey": "password"
      }
    ]
  }
}
```



```

    },
    {
      "parameterKey": "securitygroup"
    }
  ],
  "stackName": "my-test-stack"
},
"responseElements": {
  "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/a38e6a60-b397-11e3-b0fc-08002755629e"
},
"requestID": "def0bf5a-b397-11e3-bb75-a5b75389b02d",
"eventID": "637707ce-e4a3-4af1-8edc-16e37e851b17"
}

```

次の例では、Alice が ListStacks オペレーションを呼び出したことを示しています。

```

{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHJKLMNOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:03:16Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "ListStacks",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "b7d351d7-b397-11e3-bb75-a5b75389b02d",
  "eventID": "918206d0-7281-4629-b778-b91eb0d83ce5"
}

```

次の例では、Alice が my-test-stack スタックの DescribeStacks オペレーションを呼び出したことを示しています。

```

{

```

```
"eventVersion": "1.01",
"userIdentity": {
  "type": "IAMUser",
  "principalId": "AIDAABCDEFGHijklmOPQ",
  "arn": "arn:aws:iam::012345678910:user/Alice",
  "accountId": "012345678910",
  "accessKeyId": "AKIDEXAMPLE",
  "userName": "Alice"
},
"eventTime": "2014-03-24T21:06:15Z",
"eventSource": "cloudformation.amazonaws.com",
"eventName": "DescribeStacks",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
"requestParameters": {
  "stackName": "my-test-stack"
},
"responseElements": null,
"requestID": "224f2586-b398-11e3-bb75-a5b75389b02d",
"eventID": "9e5b2fc9-1ba8-409b-9c13-587c2ea940e2"
}
```

次の例では、Alice が my-test-stack スタックの DeleteStack オペレーションを呼び出したことを示しています。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHijklmOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:07:15Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "DeleteStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
```

```
"stackName": "my-test-stack"
},
"responseElements": null,
"requestID": "42dae739-b398-11e3-bb75-a5b75389b02d",
"eventID": "4965eb38-5705-4942-bb7f-20ebe79aa9aa"
}
```

CloudFormation 内のインフラストラクチャセキュリティ

マネージドサービスである AWS CloudFormation は AWS グローバルネットワークセキュリティで保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「[AWSクラウドセキュリティ](#)」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、セキュリティの柱 - AWS Well-Architected Frameworkの[インフラストラクチャ保護](#)を参照してください。

AWS の発行済み API コールを使用して、ネットワーク経由で CloudFormation にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができますが、CloudFormation ではリソースベースのアクセスポリシーがサポートされているため、ソース IP アドレスに基づく制限を含めることができます。また、CloudFormation ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントあるいは特定の VPC からのアクセスを制御することもできます。これにより、実質的に CloudFormation ネットワーク内の特定の VPC からの特定の AWS リソースへのネットワークアクセスが分離されます。

CloudFormation での耐障害性

AWS の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ゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティゾーンがあります。アベイラビリティゾーンでは、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。アベイラビリティゾーン

は、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティ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。

のコンプライアンス検証CloudFormation

AWS のサービス が特定のコンプライアンスプログラムの対象であるかどうかを確認するには、「[コンプライアンスプログラムによる対象範囲内の AWS のサービス](#)」をご覧ください、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、[AWSコンプライアンスプログラム](#)を参照してください。

AWS Artifact 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。詳細については、[AWS Artifact でレポートをダウンロードする](#)を参照してください。

AWS のサービスを使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。AWS のサービスを使用する際のコンプライアンス責任の詳細については、「[AWS セキュリティドキュメント](#)」を参照してください。

CloudFormation での構成と脆弱性の分析

構成および IT 管理は、AWS とお客様の間で共有される責任です。詳細については、AWS [責任共有モデル](#)を参照してください。

CloudFormation のセキュリティのベストプラクティス

CloudFormation には、独自のセキュリティポリシーを開発および実装する際に考慮する必要のあるいくつかのセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションに相当するものではありません。これらのベストプラクティスはお客様の環境に必ずしも適切または十分でない可能性があるため、処方箋ではなく、あくまで有用な検討事項とお考えください。

トピック

- [IAM を使用したアクセス制御](#)
- [テンプレートに認証情報を埋め込まない](#)

- [AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する](#)

IAM を使用したアクセス制御

IAM は AWS のユーザーとそのアクセス許可を管理できる AWS サービスです。IAM と CloudFormation を使用して、スタックテンプレートの表示、スタックの作成、スタックの削除など、ユーザーが実行できる CloudFormation アクションを指定できます。さらに、CloudFormation スタックを管理するには、そのスタックのリソースに対するアクセス許可が必要です。例えば、ユーザーが CloudFormation を使用して Amazon EC2 インスタンスを、起動、更新、終了する場合、そのユーザーは関連する Amazon EC2 アクションを呼び出すアクセス許可を持っている必要があります。

ほとんどの場合、ユーザーはテンプレート内のすべてのリソースを管理するためにフルアクセスを必要とします。CloudFormation は、ユーザーに代わってこれらのリソースを作成、変更、削除するための呼び出しを行います。ユーザーと CloudFormation サービスの間でアクセス許可を分割するには、サービスロールを使用します。CloudFormation は、ユーザーのポリシーの代わりにサービスロールのポリシーを使用して呼び出しを行います。詳細については、「[CloudFormation サービスロール](#)」を参照してください。

テンプレートに認証情報を埋め込まない

CloudFormation テンプレートに機密情報を埋め込むのではなく、スタックテンプレートで動的なリファレンスを使用することをお勧めします。

動的なリファレンスにより、AWS Systems Manager パラメータストアや AWS Secrets Manager などの他のサービスで保存、管理されている外部値をスタックテンプレートに指定するためのシンプルで強力な方法が提供されます。動的なリファレンスを使用すると、CloudFormation は、スタックオペレーションおよび変更セットオペレーション中に指定されたリファレンスの値を取得して、その値を適切なリソースに渡します。ただし、CloudFormation が実際の参照値を保存することはありません。詳細については、「[動的参照を使用して他のサービスに格納されている値を取得する](#)」を参照してください。

[AWS Secrets Manager](#) を使用して、データベースやその他のサービスの認証情報を安全に暗号化、保存、取得できます。[AWS Systems Manager パラメータストア](#) は、構成データ管理のための安全な階層型ストレージを提供します。

起動テンプレートのパラメータの詳細については、「[CloudFormation テンプレートの Parameters 構文](#)」を参照してください。

AWS CloudTrail を使用して CloudFormation 呼び出しをログに記録する

AWS CloudTrail は、AWS アカウントで CloudFormation API コールを実行するすべてのユーザーを追跡します。API 呼び出しは、CloudFormation API、CloudFormation コンソール、バックエンドコンソール、または CloudFormation AWS CLI コマンドの使用時にログに記録されます。ログ記録を有効にして Amazon S3 バケットを指定し、ログを保存します。こうすることで、必要ならばアカウントで誰がどのような CloudFormation 呼び出しを行ったかを監査できます。詳細については、「[AWS CloudTrail による CloudFormation API コールのログ記録](#)」を参照してください。

インターフェイスエンドポイントを使用した CloudFormation へのアクセス (AWS PrivateLink)

AWS PrivateLink を使用して、VPC と CloudFormation の間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、Direct Connect 接続のいずれかを使用せずに、VPC 内にあるかのように CloudFormation にアクセスできます。VPC 内のインスタンスは CloudFormation にアクセスするためにパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、AWS PrivateLink を利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、CloudFormation 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リクエスト管理型ネットワークインターフェイスです。

CloudFormation は、インターフェイスエンドポイントを介してすべての API アクションの呼び出しをサポートしています。

CloudFormation VPC エンドポイントに関する考慮事項

インターフェイスエンドポイントを設定する前に、まず、「AWS PrivateLinkガイド」の「[インターフェイス VPC エンドポイントを使用した AWS サービスへのアクセス](#)」で説明されている前提条件を満たしていることを確認してください。

CloudFormation のインターフェイスエンドポイントを設定する場合、以下の追加の前提条件と考慮事項が適用されます。

- VPC 内にカスタムリソースリクエストまたは待機条件に応答する必要があるリソースがある場合は、それらが必要な CloudFormation 固有の Amazon S3 バケットにアクセスできることを確認してください。CloudFormation では、[カスタムリソース](#)リクエストまたは[待機条件](#)への応答をモニ

タリングするための S3 バケットが各リージョンにあります。テンプレートに VPC 内のカスタムリソースや待機条件が含まれている場合、その VPC のエンドポイントポリシーにより、次のバケットへの応答の送信がユーザーに許可されている必要があります。

- カスタムリソースの場合、cloudformation-custom-resource-response-**region** バケットへのトラフィックを許可します。カスタムリソースを使用する場合、AWS リージョン名にダッシュは含まれません。例えば、uswest2。
- 待機条件の場合、cloudformation-waitcondition-**region** バケットへのトラフィックを許可します。待機条件を使用する場合、AWS リージョン名にはダッシュが含まれます。例えば、us-west-2。

エンドポイントポリシーでこれらのバケットへのトラフィックがブロックされている場合、CloudFormation は応答を受け取らず、スタックオペレーションは失敗します。たとえば、待機条件に応答する必要がある us-west-2 リージョンの VPC にリソースがある場合、リソースは cloudformation-waitcondition-us-west-2 バケットに応答を送信できる必要があります。

CloudFormation が現在利用可能な AWS リージョンの一覧については、「Amazon Web Services 全般のリファレンス」の [CloudFormation エンドポイントとクォータ](#) のページを参照してください。

- 現在、VPC エンドポイントはクロスリージョンの要求をサポートしていません。必ず CloudFormation への API 呼び出しを発行する予定のリージョンでエンドポイントを作成してください。
- VPC エンドポイントは、Route 53 を介して Amazon が提供する DNS のみをサポートします。独自の DNS を使用したい場合は、条件付き DNS 転送を使用できます。詳細については、「Amazon VPC ユーザーガイド」の「[Amazon VPC の DHCP オプションセット](#)」を参照してください。
- VPC エンドポイントにアタッチされたセキュリティグループでは、VPC のプライベートサブネットから、ポート 443 で着信接続を許可する必要があります。

CloudFormation のインターフェイス VPC エンドポイントの作成

CloudFormation 用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して作成できます。詳細については、「AWS PrivateLink ガイド」の「[Create a VPC endpoint](#)」を参照してください。

以下のサービス名を使用して、CloudFormation 用のインターフェイス・エンドポイントを作成します。：

- `com.amazonaws.region.cloudformation` – CloudFormation API オペレーションのエンドポイントを作成します。

インターフェイスエンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名を使用して、CloudFormation への API リクエストを実行できます。例えば、`cloudformation.us-east-1.amazonaws.com`。

FIPS 固有のエンドポイントがサポートされている AWS リージョン では、次のサービス名を使用して CloudFormation のインターフェイスエンドポイントを作成することも可能です。

- `com.amazonaws.region.cloudformation-fips` – [Federal Information Processing Standard \(FIPS\) 140-2](#) に準拠する CloudFormation API のエンドポイントを作成します。

CloudFormation エンドポイントの完全なリストについては、「Amazon Web Services 全般のリファレンス」の「[CloudFormation endpoints and quotas](#)」を参照してください。

CloudFormation 用の VPC エンドポイントポリシーの作成

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイントを介した CloudFormation へのフルアクセスが許可されています。VPC から CloudFormation に許可されるアクセスを制御するには、カスタム・エンドポイント・ポリシーをインターフェースのエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、『AWS PrivateLink ガイド』の「[Control access to VPC endpoints using endpoint policies \(エンドポイントポリシーを使用して VPC エンドポイントへのアクセスをコントロールする\)](#)」を参照してください。

例: CloudFormation アクション用の VPC エンドポイントポリシー

以下は、CloudFormation 用のエンドポイントポリシーの例です。エンドポイントにアタッチされると、このポリシーは、すべてのリソースですべてのプリンシパルに、リストされている

CloudFormation アクションへのアクセス権を付与します。次の例では、VPC エンドポイントを経由してスタックを作成するアクセス許可をすべてのユーザーに拒否し、CloudFormation サービスの他のすべてのアクションへのフルアクセスを許可します。

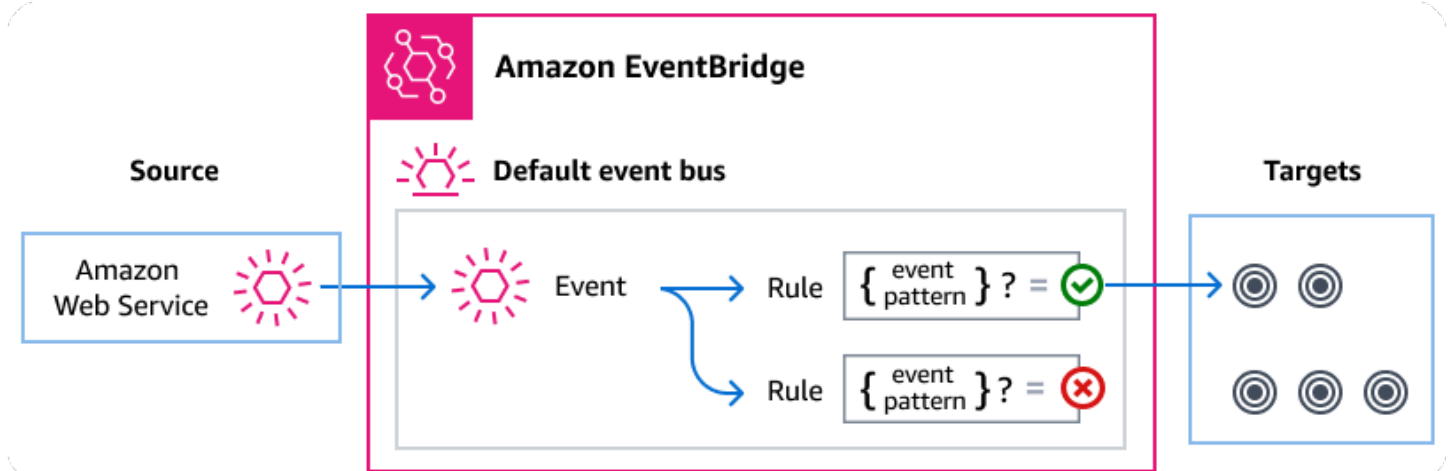
```
{
  "Statement": [
    {
      "Action": "cloudformation:*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    },
    {
      "Action": "cloudformation:CreateStack",
      "Effect": "Deny",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

EventBridge による CloudFormation および Git 同期イベントのモニタリング

Amazon EventBridge は、イベントを使用してアプリケーションコンポーネント同士を接続するサーバーレスサービスです。これにより、スケーラブルなイベント駆動型アプリケーションを簡単に構築できます。イベント駆動型アーキテクチャとは、イベントの発信と応答によって連携する、疎結合のソフトウェアシステムを構築するスタイルです。イベントとは、リソースまたは環境で発生した変更を指します。

多くの AWS のサービスと同様に、CloudFormation は、すべての AWS アカウントに自動的にプロビジョニングされている EventBridge のデフォルトのイベントバスにイベントを生成して送信します。イベントバスは、イベントを受信して、ゼロ個以上の送信先 (ターゲット) に配信するルーターです。イベントを受信されると、ユーザーがイベントバスに対して指定したルールによって評価されます。各ルールは、イベントがルールのイベントパターンに一致するかどうかをチェックします。一致する場合、イベントバスはそのイベントを指定されたターゲットに送信します。

詳細については、「Amazon EventBridge ユーザーガイド」の「[Getting started with Amazon EventBridge](#)」を参照してください。



トピック

- [CloudFormation および Git 同期イベントの概要](#)
- [Amazon EventBridge アクセス許可](#)
- [EventBridge ルールのカスタムイベントパターンの作成](#)
- [CloudFormation イベントの詳細リファレンス](#)

CloudFormation および Git 同期イベントの概要

CloudFormation は、スタックに対する作成、更新、削除、またはドリフト検出操作が実行されるたびに、EventBridge にイベントを送信します。また、CloudFormation は、スタックセットとスタックセットインスタンスのステータス変更について EventBridge にイベントを送信します。EventBridge ルールを使用すると、イベントを定義したターゲットにルートできます。これらのイベントは配信されることが保証されているため、順不同で配信される場合があります。

CloudFormation イベントはスタックまたはスタックセットとそのリソースの変更を表すため、それらを使用してそれぞれのイベントに関連するワークフローを開始できます。例えば、次のようになります。

- CloudFormation を使用してプロビジョニングされたすべてのリソースに、スタックまたはスタックセット固有のタグを作成します。
- CloudFormation スタックまたはスタックセットと Amazon WorkSpaces Application Manager (Amazon WAM) との関連付けを確立します。
- 作成されたスタックまたはスタックセットの AppRegistry との関連付けを指定します。

以下のイベントは CloudFormation によって生成され、EventBridge のデフォルトのイベントバスに送信されます。詳細については、「[???](#)」を参照してください。

イベントタイプ	説明	イベントソース
リソースのステータスの変更	スタック上で実行され、基になるリソースプロパティが変更されるすべての更新。 サポートされている AWS リソースタイプの一覧については、「 AWS リソースおよびプロパティタイプのリファレンス 」を参照してください。	AWS CloudFormation
スタックのステータスの変更	特定のスタックのステータス変更を表します。 コードの詳細については、「 スタックステータスコード 」を参照してください。	AWS CloudFormation
ドリフト検出のステータスの変更	特定のスタックでユーザーが開始したドリフト検出更新を表します。	AWS CloudFormation

イベントタイプ	説明	イベントソース
	ドリフト検出をサポートする完全なミュータブル型イミュータブル型の全数リストについては、「 リソースタイプのサポート 」を参照してください。	
StackSet のステータスの変更	特定のスタックセットのステータス変更を表します。	AWS CloudFormation
StackSet スタックインスタンスのステータスの変更	特定の StackSet スタックインスタンスへのステータス変更を表します。 コードの詳細については、「 スタックインスタンスのステータスコード 」を参照してください。	AWS CloudFormation
StackSet オペレーションステータス	特定の StackSet オペレーションのステータス変更を表します。 コードの詳細については、「 StackSets ステータスコード 」を参照してください。	AWS CloudFormation

さらに、AWS CloudFormation の Git 同期は、リポジトリ同期とリソース同期のステータス変更に関するイベントを EventBridge に送信します。

以下の Git 同期イベントは CodeConnections によって生成され、EventBridge のデフォルトイベントバスに送信されます。詳細については、「[???](#)」を参照してください。

イベントタイプ	説明	イベントソース
リポジトリ同期ステータスの変更	Git リポジトリの同期ステータスの変更を表します。	AWS CodeConnections
リソース同期ステータスの変更	Git リソース同期のステータス変更を表します。	AWS CodeConnections

Amazon EventBridge アクセス許可

CloudFormation には、EventBridge にイベントを配信するための追加のアクセス許可は必要ありません。イベントには、CloudFormation の API オペレーションを通じて既に入手可能な情報が含まれています。

指定するターゲットに、特定のアクセス許可または設定が必要になることがあります。ターゲットに特定のサービスを使用する方法の詳細については、Amazon EventBridge ユーザーガイドの「[Amazon EventBridge ターゲット](#)」を参照してください。

EventBridge ルールのカスタムイベントパターンの作成

EventBridge には CloudFormation および Git 同期イベントの事前定義されたパターンがいくつかあります。これは、イベントパターンが作成される方法を簡素化します。独自のイベントパターンを記述する代わりに、フォーム上でフィールド値を選択すると、EventBridge がパターンを生成します。これらの事前定義されたイベントパターンのいずれかを使用して新しいルールを作成するか、独自のカスタムイベントパターンを作成することができます。

CloudFormation のようなサービスがイベントをデフォルトのイベントバスに配信するとき、EventBridge は、ルールで定義されたイベントパターンを使用して、そのイベントをルールの各ターゲットに配信すべきかどうかを判断します。イベントパターンは、目的の CloudFormation イベントのデータに一致します。

各イベントパターンは JSON 形式のオブジェクトで、以下が含まれています。

- イベントを送信されるサービスを識別する `source` 属性。例えば、`aws.cloudformation`、`aws.codeconnections` です。
- (オプション): 照合するイベントタイプの配列を含む `detail-type` 属性。
- (オプション): 照合対象となるその他のイベントデータを含む `detail` 属性。

例えば、スタック ID、関連するリソース、さまざまなリソースのステータス、特定の種類のイベントに関連するその他のデータなどです。

例えば、以下のイベントパターンはすべてのリソースステータスの変更イベントと一致します。

```
{
  "source": ["aws.cloudformation"],
```

```
"detail-type": ["CloudFormation Resource Status Change"]
}
```

次のイベントパターンでは、イベント詳細データを使用して、CloudFormation が新しい AWS::S3::Bucket または AWS::SNS::Topic リソースを作成するリソースステータスの変更イベントのみを照合します。

```
{
  "source": ["aws.cloudformation"],
  "detail-type": ["CloudFormation Resource Status Change"],
  "detail": {
    "status-details": {
      "status": ["CREATE_COMPLETE"]
    },
    "resource-type": ["AWS::S3::Bucket", "AWS::SNS::Topic"]
  }
}
```

イベントパターンの記述の詳細については、「Amazon EventBridge ユーザーガイド」の「[イベントパターン](#)」を参照してください。

CloudFormation イベントの詳細リファレンス

AWS のサービスのすべてのイベントには、イベントのソースである AWS のサービス、イベントが生成された時刻、イベントが発生したアカウントとリージョンなど、イベントに関するメタデータを含む共通のフィールドセットがあります。これらの一般的なフィールドの定義については、『AWS イベントリファレンス』の「[AWS のサービスイベントメタデータ](#)」を参照してください。

さらに、各イベントには、その特定のイベントに固有のデータを含む detail フィールドがあります。以下のリファレンスでは、さまざまな CloudFormation イベントの詳細フィールドを定義しています。

EventBridge を使用して CloudFormation イベントの選択と管理を行う場合、以下の点に留意するのが有用です。

- source フィールドはイベントソースを指定します。

例えば、aws.cloudformation、aws.codeconnections です。

- detail-type フィールドはイベントタイプを指定します。

例えば、CloudFormation Resource Status Change、CloudFormation Drift Detection Status Change です。

- detail フィールドには、その特定のイベントに固有のデータが含まれます。

例えば、スタック ID、関連するリソース、さまざまなリソースのステータス、特定の種類のイベントに関連するその他のデータなどです。

CloudFormation イベントに一致するようにルールを有効化するイベントパターンの作成方法については、「Amazon EventBridge ユーザーガイド」の「[イベントパターン](#)」を参照してください。

イベントおよび EventBridge がイベントを処理する方法の詳細については、「Amazon EventBridge ユーザーガイド」の「[EventBridge イベント](#)」を参照してください。

トピック

- [リソースステータスの変更イベントの詳細](#)
- [スタックステータスの変更イベントの詳細](#)
- [ドリフト検出のステータスの変更イベントの詳細](#)
- [StackSet ステータスの変更イベントの詳細](#)
- [StackSet スタックインスタンスのステータス変更イベントの詳細](#)
- [StackSet オペレーションステータスの変更イベントの詳細](#)
- [リポジトリ同期ステータスの変更イベントの詳細](#)
- [リソース同期ステータスの変更イベントの詳細](#)

リソースステータスの変更イベントの詳細

以下は、リソースステータスの変更イベントの、詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "CloudFormation Resource Status Change",
  "source": "aws.cloudformation",
  . . . ,
}
```

```
"detail": {
  "stack-id" : "string",
  "logical-resource-id" : "string",
  "physical-resource-id": "string",
  "status-details": {
    "status": "string",
    "status-reason": "string"
  },
  "resource-type": "string",
  "client-request-token": "string"
}
```

detail-type

イベントのタイプを示します。

リソースステータスイベントの場合、この値は CloudFormation Resource Status Change です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は aws.cloudformation です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

リソースステータスイベントの場合、このデータには以下が含まれます。

stack-id

スタックと関連付けられている一意のスタック ID。

logical-resource-id

テンプレートに定義されたリソースの論理名。

physical-resource-id

CloudFormation がサポートするリソースの物理インスタンス ID に対応する名前または一意の識別子。

status-details

status

リソースのステータス。

status-reason

リソースのステータス理由。

resource-type

リソースのタイプ。例えば、AWS::S3::Bucket。

client-request-token

API を呼び出すために使用されるアクセストークン。指定されたスタックオペレーションで開始されるイベントはすべて、同一クライアントのリクエストトークンに割り当てられ、オペレーションの追跡に使用できます。コンソールから開始されたスタック操作は、Console-StackOperation-ID のトークン形式を使用し、スタック操作の識別が容易になります。たとえば、コンソールを使用してスタックを作成した場合、作成されたスタックイベントは Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002 の形式の同じトークンを割り当てられます。

Example例: リソースステータスの変更イベント

以下は、リソースステータスイベントの例です。このイベントは、CloudFormation が指定されたスタックにリクエストされたリソースである Amazon S3 バケットを正常に作成したことを詳述します。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "CloudFormation Resource Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
  ],
  "detail": {
    "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
    "logical-resource-id": "my-s3-bucket",
```

```
    "physical-resource-id":"arn:aws:s3::my-s3-bucket-us-east-1",
    "status-details":{
        "status":"CREATE_COMPLETE",
        "status-reason":""
    },
    "resource-type":"AWS::S3::Bucket",
    "client-request-token":""
}
```

スタックステータスの変更イベントの詳細

以下は、スタックステータスの変更イベントの詳細フィールドです。

`source` と `detail-type` のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
    . . . ,
    "detail-type":"CloudFormation Stack Status Change",
    "source":"aws.cloudformation",
    . . . ,
    "detail":{
        "stack-id":"string",
        "status-details":{
            "status":"string",
            "status-reason":"string"
        },
        "client-request-token":"string"
    }
}
```

detail-type

イベントのタイプを示します。

スタックステータスイベントの場合、この値は `CloudFormation Stack Status Change` です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は `aws.cloudformation` です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

スタックステータスイベントの場合、このデータには以下が含まれます。

stack-id

スタックと関連付けられている一意のスタック ID。

status-details

status

スタックのステータス。

スタックステータスコードの完全なリストについては、「[スタックステータスコード](#)」を参照してください。

status-reason

リソースのステータス理由。

client-request-token

API を呼び出すために使用されるアクセストークン。指定されたスタックオペレーションで開始されるイベントはすべて、同一クライアントのリクエストトークンに割り当てられ、オペレーションの追跡に使用できます。コンソールから開始されたスタック操作は、Console-StackOperation-ID のトークン形式を使用し、スタック操作の識別が容易になります。たとえば、コンソールを使用してスタックを作成した場合、作成されたスタックイベントは Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002 の形式の同じトークンを割り当てられます。

Example例: スタックステータスイベント

以下は、CloudFormation がリクエストされたスタック teststack を正常に作成したスタックステータスイベントの例です。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "CloudFormation Stack Status Change",
```

```

    "source": "aws.cloudformation",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "us-west-1",
    "resources": [
        "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
    ],
    "detail": {
        "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "status-details": {
            "status": "CREATE_COMPLETE",
            "status-reason": ""
        },
        "client-request-token": ""
    }
}

```

ドリフト検出のステータスの変更イベントの詳細

以下は、スタックドリフト検出イベントの詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```

{
    . . . ,
    "detail-type": "CloudFormation Drift Detection Status Change",
    "source": "aws.cloudformation",
    . . . ,
    "detail": {
        "stack-id": "string",
        "stack-drift-detection-id": "string",
        "status-details": {
            "stack-drift-status": "string",
            "detection-status": "string"
        },
        "drift-detection-details": {
            "drifted-stack-resource-count": integer
        },
        "client-request-token": "string"
    }
}

```

detail-type

イベントのタイプを示します。

スタックドリフト検出イベントの場合、この値は CloudFormation Drift Detection Status Change です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は `aws.cloudformation` です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

スタックドリフト検出イベントの場合、このデータには以下が含まれます。

stack-id

スタックと関連付けられている一意のスタック ID。

stack-drift-detection-id

スタックドリフト検知 ID。

status-details

stack-drift-status

スタックのドリフトステータス。

detection-status

ドリフト検知オペレーションのステータス

drift-detection-details

drifted-stack-resource-count

ドリフトしたリソースの数。値が -1 のとき、ドリフト検出が進行中です。その他すべての非負の整数は、ドリフトされたリソースの実際の数を表します。

client-request-token

API を呼び出すために使用されるアクセストークン。指定されたスタックオペレーションで開始されるイベントはすべて、同一クライアントのリクエストトークンに割り当てられ、オ

ペレーションの追跡に使用できます。コンソールから開始されたスタック操作は、Console-StackOperation-ID のトークン形式を使用し、スタック操作の識別が容易になります。たとえば、コンソールを使用してスタックを作成した場合、作成されたスタックイベントは Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002 の形式の同じトークンを割り当てられます。

Example例: スタックドリフト検出イベント

以下は、スタックドリフト検出イベントの例です。このイベントは、CloudFormation が指定されたスタックのドリフト検出を完了したこと、および 1 つのドリフトされたリソースによるスタックの現在のドリフトステータスが DRIFTED であることを詳述します。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "CloudFormation Drift Detection Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": ["string"],
  "detail": {
    "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
    "stack-drift-detection-id": "624af370-311a-11e8-b6b7-500cexample",
    "status-details": {
      "stack-drift-status": "DRIFTED",
      "detection-status": "DETECTION_COMPLETE"
    },
    "drift-detection-details": {
      "drifted-stack-resource-count": 1
    },
    "client-request-token": ""
  }
}
```

StackSet ステータスの変更イベントの詳細

以下は、StackSet ステータスの変更イベントの詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  . . . ,
  "detail": {
    "stack-set-arn" : "string",
    "status-details": {
      "status": "string"
    }
  }
}
```

detail-type

イベントのタイプを示します。

StackSet ステータスイベントの場合、この値は CloudFormation StackSet Status Change です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は aws.cloudformation です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

StackSet ステータスイベントの場合、このデータには以下が含まれます。

stack-set-arn

スタックセットに関連付けられている Amazon リソースネーム (ARN)。

status-details

status

StackSet ステータス。

有効な値: ACTIVE | DELETED

Example例: StackSet ステータス変更イベント

以下は、StackSet ステータス変更イベントの例です。このイベントは、CloudFormation が指定されたスタックセットを削除したことを詳述します。

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-23T17:06:18Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test12345:3f3a3fbe-c937-4eb3-a87d-e36a0af3f663"
  ],
  "detail": {
    "stack-set-arn" : "arn:aws:cloudformation:us-east-1:111122223333:stackset/test12345:3f3a3fbe-c937-4eb3-a87d-e36a0af3f663",
    "status-details": {
      "status":"DELETED"
    }
  }
}
```

StackSet スタックインスタンスのステータス変更イベントの詳細

以下は、StackSet スタックインスタンスのステータスイベントの詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  . . . ,
  "detail": {
    "stack-set-arn" : "string",
    "stack-id" : "string",
    "action" : "string",
    "status-details": {
```



```
    "status": "string",  
    "status-reason": "string",  
    "detailed-status": "string"  
  }  
}  
}
```

detail-type

イベントのタイプを示します。

StackSet スタックインスタンスのステータスイベントの場合、この値は CloudFormation StackSet StackInstance Status Change です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は aws.cloudformation です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

StackSet スタックインスタンスのステータスイベントの場合、このデータには以下が含まれます。

stack-set-arn

StackSet に関連付けられている Amazon リソースネーム (ARN)。

stack-id

スタックインスタンスと関連付けられている一意のスタック ID。

action

スタックセットオペレーションのタイプ。

有効な値: CREATE | UPDATE | DELETE | DETECT_DRIFT

status-details

status

StackSet インスタンスのステータス。

詳細については、[スタックインスタンスのステータスコード](#)を参照してください。

有効な値: CURRENT | OUTDATED | INOPERABLE

status-reason

StackSet インスタンスのステータス理由。

detailed-status

詳細な StackSet インスタンスの詳細ステータス。

有効な値: CANCELLED | FAILED | FAILED_IMPORT | INOPERABLE | PENDING | RUNNING | SKIPPED_SUSPENDED_ACCOUNT | SUCCEEDED

Example例: StackSet スタックインスタンスのステータス変更イベント

以下は、StackSet スタックインスタンスステータス変更イベントの例です。

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T19:19:23Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
    "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
    "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
    "status-details": {
      "status": "OUTDATED",
      "status-reason": "User Initiated",
      "detailed-status": "PENDING"
    }
  }
}
```

StackSet オペレーションステータスの変更イベントの詳細

以下は、StackSet オペレーションステータスの変更イベントの、詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  . . . ,
  "detail": {
    "stack-set-arn" : "string",
    "stack-set-operation-id" : "string",
    "status-details": {
      "status": "string"
    }
  }
}
```

detail-type

イベントのタイプを示します。

StackSet 操作ステータスイベントの場合、この値は CloudFormation StackSet Operation Status Change です。

source

イベントを発生させたサービスを識別します。CloudFormation イベントの場合、この値は aws.cloudformation です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

StackSet の操作ステータスイベントの場合、このデータには以下が含まれます。

stack-set-arn

StackSet に関連付けられている Amazon リソースネーム (ARN)。

stack-set-operation-id

StackSet オペレーションと関連付けられている一意のスタック ID。

status-details

status

StackSet オペレーションステータス。

詳細については、[StackSets ステータスコード](#)を参照してください。

有効な値: RUNNING | SUCCEEDED | FAILED | STOPPING | STOPPED | QUEUED

Example例: StackSet オペレーションステータス変更イベント

以下は、StackSet オペレーションステータスの変更イベントの例です。このイベントは、CloudFormation が指定されたスタックセットで要求された操作を正常に完了したことを通知します。

```
{
  "version": "0",
  "id": "4de89905-fd92-6a6b-9509-23c04bcb6a21",
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T05:46:24Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
    "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
    "stack-set-operation-id": "ce69adce-2221-4483-8c4b-c51f284f25e8",
    "status-details": {
      "status": "SUCCEEDED"
    }
  }
}
```

リポジトリ同期ステータスの変更イベントの詳細

以下は、リポジトリ同期ステータスの変更イベントの詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "Git Sync Repository Sync Status Change",
  "source": "aws.codeconnections",
  . . . ,
  "detail": {
    "connectionArn" : "string",
    "providerType" : "string",
    "repositoryName": "string",
    "providerType": "string",
    "repositoryName": "string",
    "repositoryArn": "string",
    "repositoryLinkId": "string",
    "ownerId": "string",
    "commit": "string",
    "branch": "string",
    "syncType": "string",
    "status": "string",
    "previousSync": "string"
  }
}
```

detail-type

イベントのタイプを示します。

リポジトリ同期ステータスイベントの場合、この値は Git Sync Repository Sync Status Change です。

source

イベントを発生させたサービスを識別します。Git 同期イベントの場合、この値は aws.codeconnections です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

リポジトリ同期ステータスイベントの場合、このデータには以下が含まれます。

`connectionArn`

CodeConnections に関連付けられている Amazon リソースネーム (ARN)。

`providerType`

CloudFormation に接続されている Git プロバイダー。

有効な値: GitHub | GitHub Enterprise | GitLab | BitBucket

`repositoryName`

Git リポジトリの名前。

`repositoryArn`

Git リポジトリに関連付けられた ARN。

`repositoryLinkId`

リポジトリリンクに関連付けられた一意の ID。

`ownerId`

リポジトリ所有者に関連付けられた一意の ID。

`commit`

リポジトリコミットに関連付けられた一意の ID。

`branch`

リポジトリのブランチに関連付けられた一意の ID。

`syncType`

実行中の同期のタイプ。

`status`

現在のリポジトリ同期ステータス。

有効な値: FAILED | INITIATED | IN_PROGRESS | SUCCEEDED

previousSync

現在のステータスの前の同期ステータス。

有効な値: FAILED | INITIATED | IN_PROGRESS | SUCCEEDED

Example例: リポジトリ同期ステータスの変更イベント

以下は、リポジトリ同期ステータスの変更イベントの例です。イベントは、CodeConnections がリポジトリを正常に同期したことを詳述します。

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "GitSync Repository Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
  "resources": ["arn:aws:aws:codestar-connections:us-east-1:111122223333:repository-link/550e8400-e29b-41d4-a716-446655440000"],
  "detail": {
    "connectionArn": "arn:aws:codestar-connections:us-east-1:111122223333:connection/sample-connection-id",
    "providerType": "GitHub",
    "repositoryName": "sample-repository-name",
    "repositoryArn": "arn:aws:aws:codestar-connections:us-east-1:111122223333:repository-link/550e8400-e29b-41d4-a716-446655440000",
    "repositoryLinkId": "550e8400-e29b-41d4-a716-446655440000",
    "ownerId": "sample-owner-id",
    "commit": "sample-commit-id",
    "branch": "main",
    "syncType": "CFN_STACK_SYNC",
    "status": "SUCCEEDED",
    "previousStatus": "IN_PROGRESS",
  }
}
```

リソース同期ステータスの変更イベントの詳細

以下は、リソース同期ステータスの変更イベントの詳細フィールドです。

source と detail-type のフィールドがあるのは、イベントに、これらに固有の値が含まれているためです。

```
{
  . . . ,
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
  . . . ,
  "detail": {
    "providerType" : "string",
    "commit" : "string",
    "repositoryName": "string",
    "branch": "string",
    "syncType": "string",
    "syncTarget": "string",
    "status": "string",
    "previousSync": "string"
  }
}
```

detail-type

イベントのタイプを示します。

リポジトリ同期ステータスイベントの場合、この値は Git Sync Repository Sync Status Change です。

source

イベントを発生させたサービスを識別します。Git 同期イベントの場合、この値は aws.codeconnections です。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

リソース同期ステータスイベントの場合、このデータには以下が含まれます。

providerType

CloudFormation に接続されている Git プロバイダー。

有効な値: GitHub | GitHub Enterprise | GitLab | BitBucket

`commit`

リポジトリコミットに関連付けられた一意の ID。

`repositoryName`

Git リポジトリの名前。

`branch`

リポジトリのブランチに関連付けられた一意の ID。

`syncType`

実行中の同期のタイプ。

`syncTarget`

リソース同期のターゲットスタック。

`status`

現在のリポジトリ同期ステータス。

有効な値: FAILED | INITIATED | IN_PROGRESS | SUCCEEDED

`previousSync`

現在のステータスの前の同期ステータス。

有効な値: FAILED | INITIATED | IN_PROGRESS | SUCCEEDED

Example例: リソース同期ステータスの変更イベント。

以下は、リソース同期ステータスの変更イベントの例です。イベントは、CodeConnections がリソースを正常に同期したことを詳述します。

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
  "resources": ["arn:aws:aws:cloudformation:us-east-1:111122223333:stack/targetStack1"],
}
```

```
"detail": {
  "providerType": "GitHub",
  "commit": "sample-commit-id",
  "repositoryName": "sample-repository-name",
  "branch": "main",
  "syncType": "CFN_STACK_SYNC",
  "syncTarget": "arn:aws:aws:cloudformation:us-east-1:111122223333:stack/
targetStack1",
  "status": "SUCCEEDED",
  "previousStatus": "IN_PROGRESS"
}
}
```

CloudFormation クォータを理解する

AWS アカウント には、テンプレートとスタックを作成する際に知っておく必要がある CloudFormation のクォータがあります。これらのクォータを理解することで、テンプレートまたはスタックの再設計が必要になる、クォータによるエラーを回避できます。

次の表は、CloudFormation のクォータを示しています。

クォータ	説明	値	調整方法
cfn-signal の待機条件データ	cfn-signal が渡すことができるデータの最大量。	4,096 バイト	より多くの量を渡すには、データを Amazon S3 バケットに送信します。次に、cfn-signal を使用して Amazon S3 URL をそのバケットに渡します。
カスタムリソース応答	カスタムリソースプロバイダーが渡すことができるデータの最大量。	4,096 バイト	
テンプレートあたりの動的参照数	単一の CloudFormation スタックテンプレートで許可される動的参照の最大数。	1 つのスタックテンプレートに最大 60 個の動的参照	
アカウントあたりのフック	リージョンごとにアカウントあたりの許可されるフックの最大数。	100 フック	
リソースあたりのフック	リソースあたりの設定可能なフックの最大数。	100 フック	

クォータ	説明	値	調整方法
フックの構成サイズ	フックが保存できる設定データの最大量。	204.8 KB	
マッピング	CloudFormation テンプレートで宣言できるマッピングの最大数。	200 個のマッピング	さらに多くのマッピングを指定するには、 ネストされたスタック を使用するなどして、1 つのテンプレートを複数のテンプレートに分離します。
マッピング属性	CloudFormation テンプレートで宣言できる各マッピングのマッピング属性の最大数。	200 個の属性	さらに多くのマッピング属性を指定するには、1 つの属性を複数のマッピングに分離します。
マッピング名およびマッピング属性名	各マッピング名の最大サイズ。	255 文字	
モジュール	アカウントおよびリージョンごとに、CloudFormation レジストリに登録できるモジュールの最大数。	100 モジュール	
モジュールのバージョン	特定モジュールの CloudFormation レジストリに登録できるバージョンの最大数。	100 バージョン	新しいバージョンを登録するには、最初に DeregisterType を使用して、使用していないバージョンの登録を解除します。

クォータ	説明	値	調整方法
ネストされたスタック	ネストされたスタックがオペレーションごとに作成、更新、削除できる CloudFormation リソースの最大数。 例えば、合計 2,500 を超えるリソースを持つネストされたスタック階層を持つことはできますが、1 回のデプロイで 2,500 を超えるリソースを作成、更新、または削除することはできません。	2,500 のリソース	スタック階層を別のスタックに分割します。
出力	CloudFormation テンプレートで宣言できる出力の最大数。	200 個の出力	
出力名	出力名の最大サイズ。	255 文字	
パラメータ	CloudFormation テンプレートで宣言できるパラメータの最大数。	200 個のパラメータ	さらに多くのパラメータを指定するには、複数の値を 1 つのパラメータに割り当てるために、マッピングまたはリストを使用します。
パラメータ名	パラメータ名の最大サイズ。	255 文字	

クォータ	説明	値	調整方法
パラメータ値	パラメータ値の最大サイズ。	4,096 バイト	より大きなパラメータ化された値を使用するには、複数のパラメータを作成してから、Fn::Join 機能を使用して単一の値に複数の値を連結します。
プライベートリソース	リージョンごとにアカウントあたり、CloudFormation レジストリに登録できるプライベートリソースの最大数。	50 プライベートリソース	
プライベートリソースのバージョン	特定のプライベートリソースの CloudFormation レジストリに登録できるバージョンの最大数。	50 プライベートリソース	新しいバージョンを登録するには、最初に DeregisterType を使用して、使用していないバージョンの登録を解除します。
リソース	CloudFormation テンプレートで宣言できるリソースの最大数。	500 個のリソース	さらに多くのリソースを指定するには、 ネストされたスタック を使用するなどして、1 つのテンプレートを複数のテンプレートに分離します。

クォータ	説明	値	調整方法
同時スタックオペレーションのリソース	特定の時間にリージョンでスタックオペレーション (作成、更新、削除オペレーション) に関与させることができるリソースの最大数。	特定のリージョンにおけるアカウントの現在の制限を確認するには、 DescribeAccountLimits API を使用します。	
リソース名	リソース名の最大サイズ。	255 文字	
スタック	作成できる CloudFormation スタックの最大数。	2,000 スタック	さらに多くのスタックを作成するには、不要なスタックを削除するか、AWS アカウントの最大スタック数の増加をリクエストしてください。詳細については、「AWS 全般のリファレンス」の「 AWS の Service Quotas 」を参照してください。
スタック名	スタック名の最大サイズ。	128 文字	

クォータ	説明	値	調整方法
スタックセット	管理者アカウントで作成できる CloudFormation スタックセットの最大数。	1000 個のスタックセット	さらに多くのスタックセットを作成するには、不要なスタックセットを削除するか、AWS アカウントの最大スタックセット数の引き上げをリクエストしてください。詳細については、「AWS 全般のリファレンス」の「 AWS の Service Quotas 」を参照してください。
スタックインスタンス	スタックセットごとに作成できるスタックインスタンスの最大数。	スタックセットごとの 100,000 個スタックインスタンス	さらに多くのスタックインスタンスを作成するには、不要なスタックインスタンスを削除するか、AWS アカウントの最大スタックインスタンス数の引き上げをリクエストしてください。詳細については、「AWS 全般のリファレンス」の「 AWS の Service Quotas 」を参照してください。

クォータ	説明	値	調整方法
StackSets インスタンスオペレーション	同時に各リージョンで、管理者アカウントごとにオペレーションを実行できるすべてのスタックセット間のスタックインスタンスの最大数。	10,000 オペレーション	この制限は、リージョンに関係するすべてのスタックセットに適用されます。これには、スタックセットの作成および更新オペレーションの影響を受けるスタックインスタンス、およびスタックインスタンスの直接作成、更新、または削除が含まれます。
StackSets がキューに入れられたオペレーション	特定の時間にスタックセットに対してキューを入れたオペレーションの最大数。	10,000 オペレーション	
S3 オブジェクトを使用してインポートされるスタックのスタックセットオペレーションあたりの数	スタックセットオペレーションごとに S3 オブジェクトを使用してインポートできるスタックの最大数。	200 個のスタック	
インラインスタック ID を使用してインポートされるスタックのスタックセットオペレーションあたりの数	スタックセットオペレーションごとにインラインスタック ID を使用してインポートできるスタックの最大数。	10 スタック	

クォータ	説明	値	調整方法
リクエストのテンプレート本文のサイズ	CreateStack、UpdateStack、ValidateTemplate リクエストに渡すことができるテンプレート本文の最大サイズ。	51,200 バイト	さらに大きなテンプレート本文を使用するには、 ネストされたスタック を使用するなどして、1 つのテンプレートを複数のテンプレートに分離します。または、テンプレートを Amazon S3 バケットにアップロードします。
Amazon S3 オブジェクト内のテンプレート本文のサイズ	Amazon S3 テンプレート URL を含む CreateStack、UpdateStack、ValidateTemplate リクエストに対し、Amazon S3 オブジェクトを渡すことができるテンプレート本文の最大サイズ。	1 MB	さらに大きなテンプレート本文を使用するには、 ネストされたスタック を使用するなどして、1 つのテンプレートを複数のテンプレートに分離します。または、縮小を使用して CloudFormation テンプレートのサイズを小さくします。
テンプレートの説明	テンプレート説明の最大サイズ。	1,024 バイト	
フックごとのバージョン	フックごとに作成できるバージョンの最大数	100 バージョン	

機能を利用できるリージョン

CloudFormation のすべての機能が全リージョンで使用できるわけではありません。AWS リージョンの詳細については、「[グローバルインフラストラクチャ](#)」を参照してください。

- [マクロ](#)は現在、以下のリージョンでは利用できません。
 - アジアパシフィック (ジャカルタ)
- [CloudFormation を使用して CodeDeploy による ECS ブルー/グリーンデプロイを実行すること](#)は、現在、次のリージョンでは利用できません。
 - アフリカ (ケープタウン)
 - アジアパシフィック (大阪)
 - ヨーロッパ (ミラノ)

StackSets とマクロ

StackSets は現在、マクロを含むテンプレートによるサービスマネージド許可で、スタックセットの作成または更新をサポートしていません。これには、CloudFormation によってホストされるマクロである変換が含まれます。マクロの詳細については、[マクロのテンプレート](#) を参照してください。

CloudFormation のトラブルシューティング

CloudFormation を使用するときは、CloudFormation スタックの作成、更新、または削除時に問題が発生する場合があります。以降のセクションは、発生する可能性のある一般的な問題のトラブルシューティングに役立ちます。

CloudFormation に関する一般的な質問については、「[CloudFormation のよくある質問](#)」を参照してください。また、AWS re:Post の [CloudFormation コミュニティ](#) で答えを探したり、質問を投稿したりすることもできます。

トピック

- [トラブルシューティングガイド](#)
- [エラーのトラブルシューティング](#)
- [Support へのお問い合わせ](#)

トラブルシューティングガイド

CloudFormation がスタックの作成、更新、または削除に失敗する場合は、問題に関する詳細情報の入手に役立つエラーメッセージやログを表示できます。ここでは、CloudFormation の問題をトラブルシューティングするための一般的な方法について説明します。特定のエラーと解決方法については、「[エラーのトラブルシューティング](#)」セクションを参照してください。

- ターゲットを絞ったトラブルシューティングには、オペレーション ID と障害フィルタリングを使用します。CloudFormation はデプロイイベントに一意のオペレーション ID を割り当てるため、関連するイベントの追跡と関連付けが容易になります。オペレーション ID でスタックイベントをフィルタリングして特定のデプロイフェーズに焦点を当て、障害フィルターを使用してデプロイ障害の根本原因を迅速に特定できます。「[オペレーション別にスタックイベントを表示する](#)」を参照してください。
- CloudFormation コンソールを使用して、スタックのステータスを表示します。コンソールには、スタックを作成、更新、または削除する際のスタックイベントのリストを表示できます。スタックイベントには、関連するイベントを関連させ、個々のオペレーションを追跡するのに役立つオペレーション ID が含まれます。オペレーション ID をクリックして、焦点を当てたトラブルシューティングを行うことができます。オペレーションイベントリストから障害イベントを探して、そのイベントのステータスの理由を確認します。ステータスの理由には、CloudFormation、または問題のトラブルシューティングに役立つ特定のサービスからのエラーメッセージが含まれている場

合があります。オペレーション ID があるスタックイベントの表示の詳細については、「[オペレーション別にスタックイベントを表示する](#)」を参照してください。

- Amazon EC2 の問題については、cloud-init および cfn のログを参照してください。これらのログは、/var/log/ ディレクトリの Amazon EC2 インスタンスに発行されます。これらのログは、CloudFormation がインスタンスをセットアップしているときのプロセスとコマンド出力をキャプチャします。Windows の場合、%ProgramFiles%\Amazon\EC2ConfigService で EC2Configure サービス、%ProgramData%\Amazon\EC2-Windows\Launch\Logs で EC2 Launch、%ProgramData%\Amazon\EC2Launch\log で EC2 Launch v2、C:\cfn\log で cfn ログを表示します。

また、CloudFormation テンプレートを設定して、ログが Amazon CloudWatch に発行されるようにすることも可能です。Amazon CloudWatch はAWS マネジメントコンソールにログを表示するため、Amazon EC2 インスタンスに接続する必要がなくなります。詳細については、アプリケーション管理ブログの [View CloudFormation logs in the console](#) を参照してください。

エラーのトラブルシューティング

CloudFormation スタックで以下のエラーが発生した場合は、問題の原因を特定して修正するために役立つ以下の解決策を使用できます。

トピック

- [スタックの削除の失敗](#)
- [依存関係のエラー](#)
- [AWS Config と AWS Systems Manager の競合](#)
- [リストを渡す際のパラメーター解析エラー](#)
- [IAM アクセス権限の不足](#)
- [値が無効またはリソースのプロパティがサポートされていない](#)
- [リソースクォータ](#)
- [ネストされたスタックは UPDATE_COMPLETE_CLEANUP_IN_PROGRESS、UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS または UPDATE_ROLLBACK_IN_PROGRESS 状態にあります。](#)
- [実行する更新がない](#)
- [スタックの作成、更新、または削除オペレーションの際、リソースが安定しない](#)
- [セキュリティグループが VPC 内に存在しない](#)

- [更新のロールバックの失敗](#)
- [待機条件が Amazon EC2 インスタンスから所定の数のシグナルを受け取らなかった](#)
- [リソースはスタックから削除されましたが、削除されてはいません。](#)

スタックの削除の失敗

この状況を解決するには、以下の手順を実行します。

- 一部のリソースは、削除する前に空にしなければなりません。たとえば、バケットまたはセキュリティグループを削除するには、Amazon S3 バケットのすべてのオブジェクト、または Amazon EC2 セキュリティグループのすべてのインスタンスを削除する必要があります。
- スタックのリソースを削除するのに必要な IAM アクセス権限があることを確認します。CloudFormation 許可に加えて、Amazon S3 や Amazon EC2 などの基盤となるサービスを使用するための許可も必要です。
- CloudFormation がリソースを削除できなかったことからスタックが DELETE_FAILED 状態になった場合は、RetainResources パラメータを使用し、CloudFormation が削除できないリソースを指定することで、[削除](#)を再度実行します。CloudFormation はスタックを削除しますが、保持されたリソースは削除しません。残したいオブジェクトを含む S3 バケットなど、リソースは削除できないが、スタックは削除したい場合に便利です。

スタックを削除した後で、リソースに関連した AWS サービスを使用して、残したリソースを手動で削除できます。

あるいは、DeletionMode パラメータを含む FORCE_DELETE_STACK オプションを使用することを検討することもできます。スタックを強制削除する方法の詳細については、「[CloudFormation コンソールからスタックを削除する](#)」を参照してください。

- 削除保護を有効にしているスタックを削除することはできません。削除保護を有効にした状態でスタックを削除しようとする、削除は失敗し、ステータスを含め、スタックが変更されることはありません。スタックの削除保護を無効にし、削除オペレーションを再度行います。

これには、ルートスタックの削除保護が有効になっている「[ネストされたスタック](#)」が含まれます。ルートスタックの削除保護を無効にし、削除オペレーションを再度行います。ネストされたスタックは直接削除しないことが推奨されていますが、ルートスタックとそのリソースを削除する場合のみ削除します。

詳細については、「[CloudFormation スタックの削除を防止する](#)」を参照してください。

- AWS サポート がある場合は、その他すべての問題について、サポート ケースを作成できます。
「[Support へのお問い合わせ](#)」を参照してください。

依存関係のエラー

依存関係のエラーを解決するには、テンプレート内の他のリソースに依存するリソースに DependsOn 属性を追加します。場合によっては、CloudFormation が正しい順序でリソースを作成または削除できるように、依存関係を明示的に宣言する必要があります。たとえば、同じスタック内に Elastic IP およびインターネットゲートウェイを持つ VPC を作成する場合、Elastic IP はインターネットゲートウェイのアタッチメントに依存させる必要があります。詳細については、「[DependsOn 属性](#)」を参照してください。

AWS Config と AWS Systems Manager の競合

AWS Config と AWS Systems Manager はインフラストラクチャ管理タスクを自動化できますが、これが CloudFormation スタックのデプロイとの競合の原因になる場合があります。潜在的な競合を避けるため、以下を実行してください。

- 関連付けられた AWS アカウントと AWS リージョンで、AWS Config と Systems Manager の設定を確認します。
- CloudFormation のデプロイ中にトリガーされる可能性のあるアクティブなルールやオートメーションドキュメントをチェックします。これらは、競合、またはデプロイと競合するリソースの依存関係の原因になる可能性があります。
- CloudFormation テンプレートで、AWS Config と Systems Manager が管理するリソースをチェックします。潜在的な重複や相互依存関係をチェックして、競合を避けるためにテンプレートまたはオートメーション設定を調整することを検討してください。
- CloudFormation のデプロイ中は、関連する AWS Config ルールや Systems Manager のオートメーションを一時的に無効化または停止します。望ましいレベルのオートメーションとコンプライアンスを維持するため、デプロイが正常に行われた後は元の設定に戻すことを忘れないでください。
- CloudFormation のログとエラーメッセージに AWS Config および Systems Manager に関連する問題への参照がないかどうかを確認し、競合の原因の特定に役立てます。

AWS Config ルールの詳細については、「[Evaluating Resources with AWS Config ルール](#)」を参照してください。

Systems Manager のオートメーションの詳細については、「[AWS Systems Manager Automation](#)」を参照してください。

リストを渡す際のパラメーター解析エラー

AWS CLI または CloudFormation を使用してリストを渡すときは、各カンマの前にエスケープ文字 (\) を追加します。次の例は、AWS CLI を使用する場合には入力パラメータを指定する方法を示しています。

```
ParameterKey=CIDR,ParameterValue='10.10.0.0/16\,10.10.0.0/24\,10.10.1.0/24'
```

IAM アクセス権限の不足

CloudFormation スタックを使用するときは、CloudFormation を使用する許可だけでなく、テンプレートに記述されている基盤となるサービスを使用する許可も必要になります。たとえば、Amazon S3 バケットを作成する場合や、Amazon EC2 インスタンスを開始する場合には、Amazon S3 または Amazon EC2 に対するアクセス権限が必要となります。CloudFormation スタックを使用する前に、IAM ポリシーを見直して、必要な許可があることを確認します。詳細については、「[AWS Identity and Access Management で CloudFormation アクセスを制御する](#)」を参照してください。

値が無効またはリソースのプロパティがサポートされていない

CloudFormation スタックを作成または更新するときは、無効な入力パラメータ、サポートされないリソースプロパティ名、またはサポートされないリソースプロパティ値が原因で、スタックが失敗する場合があります。入力パラメーターについては、そのリソースが存在することを確認します。たとえば、Amazon EC2 キーペアまたは VPC ID を指定する場合、リソースはアカウント内およびスタックを作成/更新しているリージョンに存在している必要があります。確実に有効な値を使用するには、[AWS 固有のパラメータ型](#)を使用できます。

リソースのプロパティ名と値については、テンプレートを更新して有効な名前と値を使用するようにしてください。詳細については、「[AWS リソースおよびプロパティタイプのリファレンス](#)」を参照してください。

リソースクォータ

リソースクォータに達していないことを確認します。例えば、起動できる Amazon EC2 オンデマンドインスタンスのデフォルトの最大数は 5 です。アカウントクォータよりも多くの Amazon EC2 オンデマンドインスタンスを作成しようとする、インスタンスの作成が失敗し、エラー Status=start_failed が返されます。サービスごとのデフォルト AWS クォータを表示するには、「AWS 全般のリファレンス」の「[AWS サービスクォータ](#)」を参照してください。

CloudFormation のクォータと調整戦略については、「[CloudFormation クォータを理解する](#)」を参照してください。

また、更新中にリソースが置き換えられた場合は、CloudFormation が古いリソースを削除する前に新しいリソースを作成します。このリソースの置換によってリソースクォータに達してしまい、更新が失敗する場合があります。過剰なリソースを削除するか、[\[quota increase\]](#) (クォータの増加) をリクエストすることができます。

ネストされたスタックは

UPDATE_COMPLETE_CLEANUP_IN_PROGRESS、**UPDATE_ROLLBACK_COMPLETE**

または **UPDATE_ROLLBACK_IN_PROGRESS** 状態にあります。

ネストされたスタックがロールバックに失敗しました。ネストされたスタック間における潜在的なリソースの依存関係が原因で、CloudFormation は、ネストされたスタックのすべてが更新される、またはロールバックされるまで、ネストされたスタックのクリーンアップを開始しません。ネストされたスタックのロールバックが失敗した場合、CloudFormation は他のスタックの状態にかかわらず、すべてのオペレーションをキャンセルします。ネストされたスタックで、更新またはロールバックを完了したが、別のネストされたスタックがロールバックに失敗したために CloudFormation からクリーンアップを開始するシグナルをから受け取らなかったというスタックは、**UPDATE_COMPLETE_CLEANUP_IN_PROGRESS** または **UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS** 状態になります。アップデートに失敗してもロールバックを開始する信号が届かないネストスタックは、**UPDATE_ROLLBACK_IN_PROGRESS** の状態になります。

ネストされたスタックは、スタックテンプレートがスタックの状態を正確に反映しないときに、CloudFormation 外で行われた変更が原因でロールバックに失敗する場合があります。また、Auto Scaling グループが作成またはアップデートされた際に、このグループのネストスタックに限定時間内のリソースが不足していると、ネストされたスタックが失敗することもあります。

スタックを修正するには、[AWS サポート](#) に連絡してください。

実行する更新がない

CloudFormation スタックを更新するには、テンプレートまたはパラメータの値の変更を CloudFormation に送信する必要があります。ただし、CloudFormation は一部のテンプレート変更を更新として認識しません。これには、削除ポリシー、更新ポリシー、条件の宣言、または出力の宣言に対する変更などがあります。他の変更を行わずにこのような変更を加える必要がある場合は、リソースのメタデータ属性を追加または変更します。CloudFormation はメタデータ属性のコンテン

ツを解釈しないため、この属性は任意の値にすることができます。詳細については、「[Metadata 属性](#)」を参照してください。

スタックの作成、更新、または削除オペレーションの際、リソースが安定しない

オペレーションが CloudFormation のタイムアウト時間を超過したか、AWS サービスが中断されたため、リソースが応答しませんでした。サービス中断の場合は、関連する AWS サービスが実行中であることを[確認](#)し、次にスタックオペレーションを再試行します。

AWS サービスが正常に実行中であれば、スタックに以下のリソースの 1 つが含まれているかどうかを確認してください。

- AWS::AutoScaling::AutoScalingGroup 作成、更新、削除オペレーション
- AWS::CertificateManager::Certificate 作成オペレーション
- AWS::CloudFormation::Stack 作成、更新、削除オペレーション
- AWS::ElasticSearch::Domain 更新オペレーション
- AWS::RDS::DBCluster 作成および更新オペレーション
- AWS::RDS::DBInstance 作成、更新、削除オペレーション
- AWS::Redshift::Cluster 更新オペレーション

これらのリソースのオペレーションは、デフォルトのタイムアウト期間以上の時間がかかる場合があります。タイムアウト期間は、リソースと使用する認証情報によって異なります。タイムアウト期間を延長するには、スタックオペレーションを実行する際に[サービスロール](#)を指定します。すでにサービスロールを使用しているか、またはスタックにリストにないリソースが含まれている場合、[AWS サポート](#) にお問い合わせください。

スタックが UPDATE_ROLLBACK_FAILED という状態の場合、「[更新のロールバックの失敗](#)」を参照してください。

セキュリティグループが VPC 内に存在しない

セキュリティグループが指定した VPC に存在していることを確認します。セキュリティグループが存在する場合、セキュリティグループの名前ではなく、セキュリティグループの ID を指定していることを確認します。たとえば、AWS::EC2::SecurityGroupIngress リソースには、SourceSecurityGroupName プロパティと SourceSecurityGroupId プロパティがありま

す。VPC セキュリティグループについては、SourceSecurityGroupId を使用し、プロパティとセキュリティグループ ID を指定する必要があります。

更新のロールバックの失敗

ロールバックが失敗するため (UPDATE_ROLLBACK_FAILED 状態)、依存リソースを元の状態に戻すことはできません。例えば、CloudFormation 外で削除された古いデータベースインスタンスにロールバックするスタックがある場合などです。CloudFormation はデータベースが削除されたことを認識しておらず、データベースインスタンスがまだ存在していると想定し、それにロールバックしようとすることから、更新のロールバックが失敗します。

失敗の原因によっては、エラーを手動で修正してロールバックを続けることができます。ロールバックを続けることで、スタックを動作状態 (UPDATE_ROLLBACK_COMPLETE 状態) に戻し、スタックの更新を再度試みることができます。次のリストでは、更新のロールバック失敗の原因となる一般的なエラーに対する解決策について説明します。

- 必要なシグナルの数の受け取りに失敗した

[signal-resource](#) コマンドを使用し、必要な数の成功シグナルを待機中のリソースに手動で送信した後、更新のロールバックを続けます。たとえば、更新のロールバック中、Auto Scaling グループ内のインスタンスが指定されたタイムアウト期間内に成功のシグナルの送信に失敗する可能性があります。Auto Scaling グループに成功シグナルを手動で送信します。更新のロールバックを続行すると、CloudFormation がシグナルを認識し、ロールバックを続行します。

- リソースに対する変更が CloudFormation 外で行われた

元のスタックテンプレートと一致するようにリソースを手動で同期した後、更新のロールバックを続けます。例えば、CloudFormation がロールバックしようとしているリソースを手動で削除した場合は、元のスタックで設定されていたものと同じ名前とプロパティを使用して、そのリソースを手動で作成する必要があります。

- アクセス権限の不足

リソースを変更する IAM アクセス権限が十分にあることを確認し、更新のロールバックを続けます。たとえば、IAM ポリシーで S3 バケットの作成が許可されるが、バケットの変更は許可されないことがあります。ポリシーに変更アクションを追加します。

- 無効なセキュリティトークン

CloudFormation には、新しい認証情報セットが必要です。変更は必要ありません。更新のロールバックを続けると、認証情報が更新されます。

- 制限エラー

不要なリソースを削除するか、[\[quota increase\]](#) (クォータの増加) をリクエストし、更新のロールバックを続けます。例えば、EC2 オンデマンドインスタンス数のアカウントクォータが 5 で、更新のロールバックがそのクォータを超えた場合、失敗します。

- リソースが安定しませんでした

オペレーションが CloudFormation のタイムアウト時間を超過したか、AWS サービスが中断された可能性があるため、リソースが応答しませんでした。変更は必要ありません。リソース操作が完了するか、AWS サービスが実行状態に戻ったら、更新のロールバックを続けてください。

更新のロールバックを続行するには、CloudFormation コンソール、または AWS コマンドラインインターフェース (AWS CLI) を使用できます。詳細については、「[更新のロールバックを続ける](#)」を参照してください。

これらのどの解決策を使用しても問題が解決されない場合は、CloudFormation が正常にロールバックできないリソースをスキップすることができます。詳細については、「AWS CloudFormation API リファレンス」の「[ContinueUpdateRollback](#) API オペレーション用の ResourcesToSkip パラメータ」を参照してください。CloudFormation は指定されたリソースのステータスを UPDATE_COMPLETE に設定し、引き続きスタックをロールバックします。ロールバックが完了したら、スキップされたリソースのステータスはスタックテンプレートのリソースのステータスと一致しません。別のスタックの更新を実行する前に、リソースを変更するか、またはスタックを更新して、整合性を取る必要があります。そうしないと、以降のスタックの更新は失敗し、スタックは回復不可能になる可能性があります。

待機条件が Amazon EC2 インスタンスから所定の数のシグナルを受け取らなかった

この状況を解決するには、以下の手順を実行します。

- 使用している AMI に CloudFormation ヘルパースクリプトがインストールされていることを確認します。AMI にヘルパースクリプトが含まれていない場合は、インスタンスにダウンロードすることもできます。詳細については、「CloudFormation テンプレートリファレンスガイド」の「[CloudFormation ヘルパースクリプトリファレンス](#)」を参照してください。
- インスタンスで `cfn-signal` コマンドが正常に実行されたことを確認します。`/var/log/cloud-init.log` や `/var/log/cfn-init.log` などのログを確認して、インスタンスの起動

に関するデバッグに役立てることができます。ログはインスタンスにログインすることで取得できますが、これには[失敗時のロールバックを無効化](#)しておく必要があります。無効化されていないと、スタックの作成失敗後に CloudFormation がインスタンスを削除します。また、Amazon CloudWatch に対して[ログを発行する](#)こともできます。Windows の場合、%ProgramFiles%\Amazon\EC2ConfigService で cfn ログを、C:\cfn\log で EC2Config サービスログを確認できます。

- インスタンスがインターネットに接続されていることを確認します。インスタンスが VPC 内に存在しており、プライベートサブネットにある場合は NAT デバイスを介して、パブリックサブネットにある場合はインターネットゲートウェイを介してインターネットに接続されます。インスタンスのインターネット接続をテストするには、`http://aws.amazon.com` などの公開ウェブページへアクセスしてください。たとえば、インスタンスで次のコマンドを実行すると、HTTP 200 ステータスコードが返されるはずです。

```
curl -I https://aws.amazon.com
```

NAT デバイスの設定の詳細については、「Amazon VPC ユーザーガイド」の「[NAT](#)」を参照してください。

リソースはスタックから削除されましたが、削除されてはいません。

スタックの更新中に、CloudFormation はスタックからリソースを削除しましたが、リソースは削除されませんでした。リソースはまだ存在しますが、CloudFormation からはアクセスできなくなります。これは、スタックの更新中に発生する可能性があります。

- CloudFormation は既存のリソースを置き換える必要があるため、最初に新しいリソースを作成し、次に古いリソースを削除しようとします。
- スタックテンプレートからリソースを削除したため、CloudFormation はスタックからリソースを削除しようとします。

ただし、CloudFormation がリソースを削除できない場合があります。例えば、特定のタイプのリソースを削除するためのアクセス権限がユーザーにない場合などです。

CloudFormation は、古いリソースを 3 回削除しようとします。CloudFormation が古いリソースを削除できない場合、古いリソースをスタックから削除し、スタックの更新を続行します。スタックの更新が完了すると、CloudFormation は UPDATE_COMPLETE スタックイベントを発行しますが、1 つ以上のリソースを削除できなかったことを示す StatusReason が含まれます。CloudFormation

は、特定のリソースに対して DELETE_FAILED イベントを発行し、対応する StatusReason は CloudFormation がリソースの削除に失敗した理由の詳細を提供します。

この状況を解決するには、基盤となるサービスのコンソールまたは API を使用してリソースを直接削除します。

Support へのお問い合わせ

AWS サポート がある場合は、<https://console.aws.amazon.com/support/home#/> で技術サポートケースを作成できます。サポートへ連絡する前に、以下の情報を収集します。

- スタックの ID。スタック ID は、[CloudFormation コンソール](#)の [概要] タブにあります。詳細については、「[CloudFormation コンソールからスタック情報を表示する](#)」を参照してください。

Important

CloudFormation の外部でスタックのリソースを変更しないでください。CloudFormation の外部でスタックのリソースを変更すると、スタックが回復不可能な状態になる可能性があります。

- スタックのエラーメッセージ。スタックのエラーメッセージの表示については、「[トラブルシューティングガイド](#)」セクションを参照してください。
- Amazon EC2 の問題については、cloud-init および cfn のログを収集してください。これらのログは、/var/log/ ディレクトリの Amazon EC2 インスタンスに発行されます。これらのログには、インスタンスのセットアップ中のプロセスおよびコマンドの出力がキャプチャされます。Windows の場合、%ProgramFiles%\Amazon\EC2ConfigService および C:\cfn\log に保存される EC2Configure サービスおよび cfn ログを収集してください。

また、AWS re:Post の [CloudFormation コミュニティ](#)で答えを探したり、質問を投稿したりすることもできます。

Amazon Q Developer を使用して、失敗した CloudFormation スタックデプロイをトラブルシューティングする

Amazon Q Developer は、生成人工知能 (AI) を活用した会話型アシスタントであり、AWS アプリケーションの理解、構築、拡張、運用を支援します。Amazon Q Developer の詳細については、「[Amazon Q Developer ユーザーガイド](#)」を参照してください。

このドキュメントでは、失敗した CloudFormation デプロイをトラブルシューティングするのに役立つよう Amazon Q Developer を使用方法について説明します。この機能は、スタックのデプロイ中に発生した問題を迅速に特定して解決するのに役立つよう設計されています。

Amazon Bedrock を利用: Amazon Q は Amazon Bedrock に構築され、Amazon Bedrock に実装された[不正使用の自動検出](#)機能を搭載して、安全、セキュリティ、AI の責任ある使用を強化しています。

トピック

- [機能](#)
- [仕組み](#)
- [料金](#)
- [リージョナルな可用性](#)
- [制限事項](#)

機能

Amazon Q Developer ソリューションを使用したスタックデプロイのトラブルシューティングは、次の機能を提供します:

- エラー分析 — CloudFormation スタックのデプロイエラーのわかりやすい分析を提供します。
- 解決策の提案 — 特定された問題を解決するためのステップバイステップの手順を提供します。

仕組み

CloudFormation スタックの作成または更新が失敗すると、エラーメッセージが AWS マネジメントコンソール に表示されます。

エラーメッセージの横にある [Q で診断] ボタンを選択できます。

Amazon Q Developer はエラーを分析し、問題の説明を提供します。

その他のサポートについては、[解決のヘルプ] を選択すると、詳細な解決ステップを受け取ることができます。

料金

料金の詳細については、「[Amazon Q Developer の料金](#)」を参照してください。

リージョナルな可用性

Amazon Q Developer は、ほとんどの AWS リージョン で使用できます。Amazon Q Developer が現在利用可能なすべてのリージョンのリストについては、「Amazon Q Developer ユーザーガイド」の「[Supported Regions for Amazon Q Developer](#)」を参照してください。

制限事項

- この機能は、AWS マネジメントコンソール を通じてのみアクセスできます。
- この機能は、上位 20 の最も一般的なエラーをカバーしています。考え得るすべてのエラーをカバーしているわけではありません。
- 複雑なトラブルシューティングケースでは、サポート の介入が必要になる場合があります。

AWS CloudFormation ユーザーガイドのドキュメント履歴

次の表は、2018 年 5 月以降の AWS CloudFormation ユーザーガイドに対する重要な変更点の一覧です。ドキュメントの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

Important

2018 年 5 月以降のテンプレートリファレンスの内容の更新を示す表の行は、新しい AWS CloudFormation テンプレートリファレンスに移動しました。これらの更新と今後の変更については、『AWS CloudFormation テンプレートリファレンス』の「[AWS CloudFormation テンプレートリファレンスのドキュメント履歴](#)」を参照してください。

変更	説明	日付
PDF ガイドを提供開始	AWS CloudFormation ユーザーガイドを PDF としてダウンロードできるようになりました。	2025 年 5 月 30 日
テンプレートリファレンスの内容を新しいガイドに移動しました	CloudFormation は「CloudFormation テンプレートリファレンスガイド」を公開しました。詳細については、『 CloudFormation テンプレートリファレンスガイド 』を参照してください。	2025 年 5 月 30 日
IaC ジェネレーターが部分スキャンをサポート	スキャンする特定のリソースタイプを選択できるようになりました。これにより、既存のリソースから Infrastructure-as-Code (IaC) テンプレートを簡単に生成できるようになります。詳細については、「 IaC ジェネレーターを使用してリ	2025 年 3 月 27 日

[ソーススキャンを開始する」](#)

を参照してください。

[スタックリファクタリング](#)

スタックリファクタリングにより、既存のリソースプロパティおよびデータを維持しながら、CloudFormation スタックのリソースの再編成が簡素化されます。詳細については、「[スタックリファクタリング](#)」を参照してください。

2025 年 2 月 6 日

[Amazon Q Developer を使用してスタックデプロイをトラブルシューティングする](#)

Amazon Q Developer を使用して、CloudFormation スタックをデプロイする際の一般的なエラーをトラブルシューティングできるようになりました。詳細については、「[Amazon Q Developer を使用して、失敗した CloudFormation スタックデプロイをトラブルシューティングする](#)」を参照してください。

2024 年 11 月 22 日

[スタックデプロイタイムライングラフ](#)

スタックデプロイを視覚的に表示できるようになりました。スタックデプロイタイムライングラフには、スタックのデプロイステータス、個々のリソースのデプロイステータス、デプロイステータスが変更された時刻が表示されます。詳細については、「[CloudFormation スタックデプロイのタイムラインを表示する](#)」を参照してください。

2024 年 11 月 11 日

[スキャンしたリソースと生成されたテンプレートを視覚化する](#)

インフラストラクチャスタックをデプロイする前に、スキャンの概要の詳細を視覚化し、生成されたテンプレートをプレビューすることで、Infrastructure as Code (IaC) ジェネレーターのワークフローを合理化できるようになりました。詳細については、「[CloudFormation コンソールでスキャンの概要を表示する](#)」および「[スキャンされたリソースから CloudFormation スタックを作成する](#)」を参照してください。

2024 年 8 月 22 日

[Amazon EventBridge と AWS CloudFormation Git 同期の統合](#)

AWS CloudFormation Git 同期は、同期ステータスの変更をイベントとして Amazon EventBridge に発行するようになりました。詳細については、「[リポジトリ同期ステータスの変更イベントの詳細](#)」および「[リソース同期ステータスの変更イベントの詳細](#)」を参照してください。

2024 年 7 月 29 日

[スタックを強制的に削除する](#)

スタックの削除操作が停止してしまった場合に備えて、スタックを強制的に削除するための新しいオプションが 2 つ追加されました。スタックを強制削除してリソースを保持するか、スタック全体を強制削除するかを選択できるようになりました。詳細については、「[CloudFormation コンソールからスタックを削除する](#)」を参照してください。

2024 年 5 月 22 日

[AWS CloudTrail イベントスタックオペレーションの根本原因](#)

CloudFormation は、新しい AWS CloudTrail デイブリンク統合により、スタックオペレーションのトラブルシューティングエクスペリエンスを改善します。この機能は、CloudFormation コンソールのスタックオペレーションイベントを、関連する CloudTrail イベントに直接リンクします。詳細については、「[スタックが失敗する原因を判断する](#)」を参照してください。

2024 年 5 月 15 日

[プロパティレベルの変更セット](#)

プロパティレベルの変更セットを使用すると、CloudFormation デプロイがリソースのプロパティ値に加える変更をプレビューできます。詳細については、「[CloudFormation スタックの変更セットを表示する](#)」を参照してください。

2024 年 4 月 12 日

[CloudFormation が CONFIGURATION_COMPLETE イベントを導入](#)

CONFIGURATION_COMPLETE イベントを使用して、リソースの作成を含むワークフローを高速化しました。詳細については、「[CloudFormation スタック作成イベントを理解する](#)」を参照してください。

2024 年 3 月 11 日

[既存の AWS リソースから AWS CloudFormation テンプレートと AWS CDK アプリケーションを生成する](#)

まだ CloudFormation で管理されていないアカウントにプロビジョニングされたリソースを使ってテンプレートを生成できるようになりました。詳細については、「[IaC ジェネレーターを使用して既存のリソースからテンプレートを生成する](#)」を参照してください。

2024 年 2 月 2 日

[StackSets の同時実行モード](#)

同時実行モードは、スタックセット操作中の同時実行レベルの動作を選択できる `StackSetOperationPreferences` のパラメータです。詳細については、「[CloudFormation StackSets の同時実行モードの選択](#)」を参照してください。

2023 年 11 月 9 日

[StackSet のドリフト情報詳細](#)

次の API で、どのスタックインスタンスが StackSet テンプレートからドリフトし、どのリソースがドリフトしたかを確認できます。

2023 年 7 月 24 日

[ListStackInstanceResourceDrifts](#)

スタックインスタンスのリソースのドリフト情報を返します。

[StackInstanceResourceDriftsSummary](#)

スタックインスタンスのリソースのドリフトに関する概要情報を含む構造体。

[AWS Organizations の信頼アクセスを制御する CloudFormation StackSets API](#)

CloudFormation StackSets は、AWS Organizations の信頼アクセスを管理するために次の API を提供します。

2023 年 6 月 5 日

[ActivateOrganizationsAccess](#)

AWS Organizations の信頼されたアクセスをアクティブ化します。StackSets と組織の間で信頼されたアクセスがアクティブ化されていると、管理アカウントのアクセス許可で組織の StackSets を作成して管理することができます。

[DeactivateOrganizationsAccess](#)

AWS Organizations の信頼されたアクセスを非アクティブ化します。信頼されたアクセスが非アクティブ化されている場合、管理アカウントには組織のサービスマネージド StackSets を作成して管理する許可がありません。

[DescribeOrganizationsAccess](#)

アカウントの OrganizationAccess ステータスに関する情報を取得します。管理アカウントか、委任された管理者が CallAs パラメーターを使用すると、この API を呼び出す

ことができます。この API は、管理アカウントによって CallAs パラメーターを使用せずに呼び出すこともできます。

[DescribeStackSet API](#)

DescribeStackSet API には、スタックセットがデプロイされているリージョンのリストに新しいパラメータがあります。詳細については、「[DescribeStackSet](#)」を参照してください。

2023 年 2 月 1 日

[CloudFormation と Amazon EventBridge を使用して StackSets イベントを管理する](#)

CloudFormation StackSets は Amazon EventBridge 経由でイベント通知を起動します。CloudFormation スタックセットを作成、更新、削除した後、イベント駆動型アクションをトリガーできます。詳細については、「[EventBridge による CloudFormation および Git 同期イベントのモニタリング](#)」を参照してください。

2022 年 11 月 16 日

[スタックセット操作のスタックインスタンスに関する分析情報の改善](#)

CloudFormation StackSets は、スタックセット操作のスタックインスタンスに関するより詳細な情報を提供します。

2022 年 11 月 4 日

[DescribeStackSetOperation](#)

DescribeStackSetOperation を使用して、デプロイ中のスタックセット操作で失敗したスタックインスタンスの数を求められるようになりました。

[ListStackInstances](#)

フィルタリングオプション LastOperationID を使用してスタックセット操作のスタックインスタンスを一覧表示できるようになりました。

[CloudFormation と Amazon EventBridge を使用してイベントを管理する](#)

特定の CloudFormation イベントが発生したときに通知を受信します。詳細については、「[EventBridge による CloudFormation および Git 同期イベントのモニタリング](#)」を参照してください。

2022 年 7 月 20 日

[アカウントレベル](#)

CloudFormation により、「アカウントフィルタータイプ」の一般発売が発表されます。これは、お客様がデプロイターゲットを個々のアカウントに制限したり、提供された OU にアカウントを追加したりできる機能です。詳細については、[「サービスマネージド型スタックセットのアカウントレベルのターゲット」](#)を参照してください。

2022 年 7 月 7 日

[CloudFormation レジストリ](#)

CloudFormation により、「フック」の一般発売が発表されます。これは、スタックの作成、更新、または削除のオペレーションの前に、カスタムロジックを呼び出してアクションを自動化したり、リソース設定を検査したりできる機能です。詳細については、[「CloudFormation Hooks ユーザーガイド」](#)を参照してください。

2022 年 2 月 10 日

スタック障害オプション

プロビジョニングに失敗した場合、正常にプロビジョニングされたリソースをロールバックせずに失敗時点から開始することで、アプリケーションを繰り返し開発できます。スタック障害オプションを指定することにより、CREATE_FAILED または UPDATE_FAILED ステータスのリソースのトラブルシューティングを行うことができます。すべてのスタックデプロイと変更セットオペレーションの障害オプションをプロビジョニングできます。詳細については、「[リソースのプロビジョニング時における失敗への対応方法を選択する](#)」を参照してください。

2021 年 8 月 30 日

スタックをスタックセットに インポートする

既存のスタックを新規または既存のスタックセットにインポートできるようになりました。詳細については、「[CloudFormation StackSets へのスタックのインポート](#)」を参照してください。

2021 年 7 月 28 日

引き上げられたクォータ

AWS アカウント でデフォルトの最大 2000 のスタックを宣言できるようになりました。詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

2021 年 7 月 15 日

[パブリックサードパーティの拡張を公開する](#)

AWS からの拡張と同じように、サードパーティのパブリッシャーが提供するパブリック拡張を使用できるようになりました。詳細については、「[CloudFormation レジストリからサードパーティのパブリック拡張を使用する](#)」を参照してください。

2021 年 6 月 21 日

[スタックセットテンプレートでマクロを参照する](#)

StackSets では、マクロを参照するテンプレートから、セルフマネージド型アクセス許可を使用してスタックセットを作成または更新できるようになりました。マクロの詳細については、「[テンプレートマクロを使用して CloudFormation テンプレートでカスタム処理を実行する](#)」を参照してください。

2021 年 4 月 14 日

[動的リファレンスで Systems Manager パラメータの最新値を使用する](#)

スタックを作成または更新するたびに、CloudFormation で Systems Manager パラメータの最新バージョンを使用できるようになりました。特定のバージョンを指定する必要がなくなりました。詳細については、「[Systems Manager Parameter Store からプレーンテキスト値を取得する](#)」を参照してください。

2021 年 4 月 13 日

[モジュールはリソース名での ピリオド区切り文字の使用を サポート](#)

モジュールに含まれるリソースの完全修飾論理名を指定する際に、区切り文字としてピリオドを使用できるようになりました。詳細については、「[CloudFormation テンプレートのリファレンスモジュールリソース](#)」を参照してください。

2021 年 4 月 8 日

[CloudFormation StackSets が 並列リージョンのデプロイを サポート開始](#)

StackSet をリージョンに順次デプロイするか、並列にデプロイするかを選択できるようになりました。詳細については、[スタックセットの操作オプション](#)を参照してください。

2021 年 4 月 6 日

[CloudFormation StackSets が、AWS Organizations で委 任された管理者のサポートを 開始](#)

組織の 管理アカウント の他に、委任された管理者アカウントは、組織のサービスマネージド型のアクセス許可を持つスタックセットを作成および管理できます。詳細については、「[委任された管理者メンバーアカウントを登録する](#)」および「[サービスマネージド許可を持つ CloudFormation StackSets を作成する](#)」を参照してください。

2021 年 2 月 18 日

[CloudFormation StackSets リージョンの可用性](#)

CloudFormation StackSets が
アジアパシフィック (大阪)
リージョンでご利用いただける
ようになりました。詳細に
ついては、「[StackSets を使
用したアカウントとリージョ
ン全体でのスタックの管理](#)」
を参照してください。

2021 年 2 月 10 日

[モジュール](#)

モジュールは、スタックテン
プレート全体に含めるリソー
ス構成をパッケージ化する、
透過的で管理しやすく、繰
り返し可能な方法です。モ
ジュールは、共通のサービ
ス構成とベストプラクティス
を、スタックテンプレートに
含めるためのモジュール式の
カスタマイズ可能なビルディ
ングブロックとしてカプセル
化できます。詳細については
、「[CloudFormation モジュー
ルを使用してテンプレート全
体に含めることができる再
利用可能なリソース設定の作
成](#)」を参照してください。

2020 年 11 月 24 日

ネストされたスタックの変更 セット

ネストされたスタックの変更セットを使用すると、ネストされたスタック階層全体でアプリケーションとインフラストラクチャリソースに対する変更をプレビューし、すべての変更が意図したとおりに行われたことを確認してから、更新を続行できます。詳細については、「[ネストされたスタックの変更セット](#)」を参照してください。

2020 年 11 月 18 日

引き上げられたクォータ

次の AWS CloudFormation クォータが更新されました。

2020 年 10 月 22 日

- AWS CloudFormation テンプレートで最大 200 個のマッピングを宣言できるようになりました。
- AWS CloudFormation テンプレート内のマッピングごとに最大 200 個のマッピング属性を宣言できるようになりました。
- AWS CloudFormation テンプレートで最大 200 個の出力を宣言できるようになりました。
- AWS CloudFormation テンプレートで最大 200 個のパラメータを宣言できるようになりました。
- AWS CloudFormation テンプレートで最大 500 個のリソースを宣言できるようになりました。
- Amazon S3 オブジェクトで最大サイズが 1 MB のテンプレート本文を渡すことができるようになりました。

[プライベートリソースのドリフト検出](#)

CloudFormation は、さまざまな AWS リソースに加え、CloudFormation レジストリでプロビジョニング可能と定義されたプライベートリソースに対するドリフト検出オペレーションをサポートするようになりました。詳細については、「[リソースタイプのサポート](#)」を参照してください。

2020 年 10 月 1 日

[リソースプロバイダーの登録に必要なアクセス許可を更新しました](#)

リソースプロバイダーをアカウントに登録するには、そのリソースプロバイダーの S3 バケットにアップロードされたスキーマハンドラーパッケージにアクセスするためのアクセス許可が必要になりました。詳細については、「[サードパーティのプライベート拡張の登録のための IAM アクセス許可](#)」を参照してください。

2020 年 8 月 7 日

[リソースのインポートは、プロビジョニング可能なプライベートリソースタイプをサポートします](#)

インポートオペレーションは、プロビジョニング可能なプライベートリソースタイプ (プロビジョニングタイプが FULLY_MUTABLE または IMMUTABLE) がサポートされるようになりました。詳細については、「[インポートオペレーションをサポートするリソース](#)」を参照してください。

2020 年 6 月 3 日

[CodeDeploy による ECS ブルー/グリーンデプロイ](#)

CloudFormation を使用して、CodeDeploy による ECS ブルー/グリーンデプロイを実行できるようになりました。ブルー/グリーンデプロイは、AWS CodeDeploy が提供する安全なデプロイ戦略で、アプリケーションバージョンの変更による中断を最小限に抑えます。詳細については、「[CloudFormation を使用した CodeDeploy による ECS ブルー/グリーンデプロイの実行](#)」を参照してください。

2020 年 5 月 19 日

[CloudFormation StackSets リージョンの可用性](#)

CloudFormation StackSets が AWS GovCloud (米国西部) リージョンでご利用可能になりました。

2020 年 5 月 18 日

[AWS CloudFormation スタックセットが AWS Organizations と統合](#)

StackSets を使用して、AWS Organizations 組織内のすべてのアカウントまたは特定の組織単位 (OU) へのデプロイを一元的に管理できるようになりました。また、組織または OU に追加された新しいアカウントへの自動デプロイを有効にすることもできます。アカウント間でデプロイするために必要なアクセス許可は、StackSets によって自動的に処理されます。詳細については、「[StackSets を使用したアカウントとリージョン全体でのスタックの管理](#)」を参照してください。

2020 年 2 月 11 日

[スタックセットのドリフト検出](#)

スタックセットとそれに含まれるすべてのスタックインスタンスでドリフト検出を実行できるようになりました。詳細については、「[CloudFormation StackSets でのドリフト検出の実行](#)」を参照してください。

2019 年 11 月 19 日

CloudFormation レジストリが 利用可能になりました

CloudFormation コンソールを使用して、アカウントで使用できるプライベートリソースとパブリックリソースを表示できるようになりました。詳細については、「[CloudFormation レジストリで使用可能でアクティブ化された拡張を表示する](#)」を参照してください。

2019 年 11 月 18 日

[CloudFormation レジストリ API アクション](#)

CloudFormation レジストリで
タイプを管理するための以下
の API アクションが使用でき
るようになりました。

2019 年 11 月 18 日

[DeregisterType](#)

CloudFormation レジスト
リでのアクティブな使用か
らタイプまたはタイプバー
ジョンを削除します。

[DescribeType](#)

登録された型に関する詳細
情報を返します。

[DescribeTypeRegistration](#)

現在のステータス、タイ
プ、バージョン識別子な
ど、タイプの登録に関する
情報を返します。

[ListTypeRegistrations](#)

指定されたタイプに登録リ
クエスト識別子のリストを
返します。

[ListTypes](#)

CloudFormation に登録さ
れたタイプに関する概要情
報を返します。

[ListTypeVersions](#)

タイプのバージョンに関す
る概要情報を返します。

[RegisterType](#)

CloudFormation レジストリにタイプを登録します。タイプを登録すると、AWS アカウントの CloudFormation テンプレートで使えるようになります。

[SetTypeDefaultVersion](#)

タイプのデフォルトバージョンを指定します。タイプのデフォルトバージョンは、CloudFormation オペレーションで使用されます。

CloudFormation レジストリの詳細については、「[CloudFormation レジストリによる拡張機能の管理](#)」を参照してください。

[リソースのインポートが追加されました](#)

CloudFormation 管理外の AWS リソースを作成した場合は、`resource import` を使用してこの既存のリソースを CloudFormation 管理に取り込むことができます。詳細については、「[リソースインポートを使用して AWS リソースを CloudFormation スタックにインポートする](#)」を参照してください。

2019 年 11 月 11 日

スタックセット制限の引き上げ

管理者アカウントで最大 100 個のスタックセット、スタックセットごとに最大 2000 個のスタックインスタンスを作成し、各リージョンで同時に管理者アカウントごとに最大 3500 個のスタックインスタンスオペレーションを実行できるようになりました。詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

2019 年 8 月 2 日

同時スタックオペレーションのリソース数の制限

CloudFormation では、同時スタックオペレーションのリソース数に対してアカウント制限を適用するようになりました。この制限はリージョン別に決定されます。詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

2019 年 4 月 30 日

スタックインスタンスオペレーションの制限

StackSet では、管理者アカウントごとに、特定のリージョンで同時に最大 1500 のスタックインスタンスオペレーションを実行できるようになりました。詳細については、「[CloudFormation クォータを理解する](#)」を参照してください。

2018 年 12 月 13 日

CAPABILITY_AUTO_EX

PAND 機能が利用可能になりました。

CAPABILITY_AUTO_EX

PAND 機能を使用して、マクロを含むスタックテンプレートから直接スタックを作成または更新できるようになりました。最初に変更セット内の変更を確認する必要はありません。詳細については、「AWS CloudFormation API リファレンス」の「[CreateStack](#)」または「[UpdateStack](#)」を参照してください。

2018 年 12 月 7 日

スタックドリフト検出の追加

スタックの実際の設定が、CloudFormation 内で定義されている、意図されたテンプレート設定からドリフトしていないか確認できるようになりました。スタック全体またはスタックのリソース別のドリフトを検出できます。詳細については、「[ドリフト検出を使用してスタックとリソースへのアンマネージド型設定変更を検出する](#)」を参照してください。

2018 年 11 月 13 日

secretsmanager の動的な参照機能の提供開始

secretsmanager の動的な参照を使用し、AWS Secrets Manager に保存されている全体のシークレットまたはシークレット値を取得できるようになりました。シークレットとは、データベース認証情報、パスワード、サードパーティーの API キーなどの任意のテキストです。secretsmanager の動的な参照を使用すると、Secrets Manager や CloudFormation で解決済みのシークレットの値が記録または保持されなくなります。詳細については、「[Secrets Manager からシークレットまたはシークレット値を取得する](#)」を参照してください。

2018 年 11 月 9 日

マクロが利用可能になりました

マクロを使用して、検索と置換操作などの単純なアクションからテンプレート全体の広範な変換まで、テンプレートに対してカスタム処理を実行できるようになりました。詳細については、「[テンプレートマクロを使用して CloudFormation テンプレートでカスタム処理を実行する](#)」を参照してください。

2018 年 9 月 6 日

[CloudFormation が PrivateLink によって強化された VPC エンドポイントをサポート開始](#)

VPC エンドポイントを使用すると、インターネット、NAT インスタンス、VPN 接続、または Direct Connect を経由せずに、VPC と CloudFormation とをプライベートに接続できます。詳細については、「[インターフェイスエンドポイント \(AWS PrivateLink\) を使用して CloudFormation にアクセスする](#)」を参照してください。

2018 年 8 月 22 日

[動的な参照で Secure String をサポート](#)

新しい動的な参照を使用して、スタックテンプレート内の Systems Manager パラメータストアの SecureString 型パラメータなど、他のサービスで格納および管理されている値を指定できるようになりました。詳細については、「[Systems Manager Parameter Store から SecureString 値を取得する](#)」を参照してください。

2018 年 8 月 16 日

[スタックセットでは、カスタマイズされた実行ロールがサポートされるようになりました。](#)

ターゲットアカウントでカスタマイズされた実行ロールを使用して、ユーザーまたはグループがスタックセットに含めることができるスタックリソースを制御できるようになりました。詳細については、「[スタックセットオペレーションのアドバンスドアクセス許可オプションの設定](#)」を参照してください。

2018 年 5 月 30 日

[スタックインスタンスの選択的な更新](#)

オプションの Accounts および Regions パラメータを使用して、スタックセットの更新操作中にスタックインスタンスを更新するアカウントとリージョンを指定できるようになりました。詳細については、「CloudFormation API リファレンス」の「[UpdateStackSet](#)」を参照してください。

2018 年 5 月 30 日

[CloudFormation が暗号化を有効にした S3 バケットの作成機能の提供開始](#)

アップロードされたスタックテンプレートを保存するために CloudFormation が作成する Amazon S3 バケットの場合、サーバー側の暗号化がデフォルトで有効になり、バケットに保存されているすべてのオブジェクトが暗号化されるようになりました。詳細については、「[CloudFormation コンソールからスタックを作成する](#)」を参照してください。

2018 年 5 月 24 日

[追加された FIPS エンドポイント](#)

CloudFormation は現在、US-East-1、US-East-2、US-West-1、および US-West-2 の米国の FIPS 140-2 検証済み暗号化モジュールを使用する新しいエンドポイントを提供しています。新しい FIPS 準拠エンドポイントの URL については、『Amazon Web Services 全般のリファレンス』の「[CloudFormation エンドポイントとクォータ](#)」を参照してください。

2018 年 5 月 17 日

CloudFormation フックユーザーガイドの更新については、『CloudFormation フックユーザーガイド』の「[CloudFormation フックユーザーガイドのドキュメント履歴](#)」を参照してください。

アーカイブされた更新

以下の表に、2018 年 5 月より前の「AWS CloudFormation 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	リリース日	説明	API バージョン
更新されたリソース	2019 年 7 月 22 日	encryptionOptions プロパティを使用して、AWS 所有のキー または Amazon MQ ブローカー用のカスタマー管理キーを指定します。	2010-05-15
スタックセットの命名規則	2018 年 4 月 10 日	スタックセットを使用して作成された CloudFormation スタックは、新しい命名規則に従うようになりました。この命名規則では、スタック名にスタックセット名が含まれます。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
新しいリソース	2018 年 4 月 10 日	AWS::AppSync::ApiKey AWS AppSync で GraphQL オペレーションを実行しているクライアントに配信できる一意のキーを作成するには、AWS::AppSync::ApiKey リソースを使用します。 AWS::AppSync::DataSource AWS::AppSync::DataSource リソースを使用して、AWS AppSync でリゾルバのデータソースを作成します。 AWS::AppSync::GraphQLApi AWS AppSync で新しい GraphQL API を作成するには、AWS::AppSync::GraphQLApi リソースを使用します。 AWS::AppSync::GraphQLSchema AWS AppSync の GraphQL API 向けにデータモデルを作成するには、AWS::AppSync::GraphQLSchema リソースを使用します。 AWS::AppSync::Resolver スキーマのフィールドにアタッチする論理的な GraphQL リゾルバを定義するには、AWS::AppSync::Resolver リソースを使用します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2018 年 4 月 10 日	AWS::Config::ConfigurationAggregator AWS Config 構成アグリゲータに集約する AWS Config データのリージョンと、AWS Organizations の詳細を取得するために使用する IAM ロールを指定するには、OrganizationAggregationSource プロパティタイプを使用します。	2010-05-15
新しいリソース	2018 年 4 月 4 日	AWS::Config::AggregationAuthorization AWS Config データを収集するアグリゲータアカウントにアクセス権限を付与するには、AWS::Config::AggregationAuthorization リソースを使用します。 AWS::Config::ConfigurationAggregator AWS Config の構成アグリゲータを作成するには、AWS::Config::ConfigurationAggregator リソースを使用します。	2010-05-15
スタックセットでは、カスタマイズされた管理者ロールがサポートされるようになりました。	2018 年 3 月 29 日	同じ管理者アカウント内で特定のスタックセットを管理するユーザーやグループを制御するには、カスタマイズされた管理者ロールを使用します。詳細については、「 スタックセットオペレーションのアドバンストアクセス許可オプションの設定 」を参照してください。	2010-05-15
新しいリソース	2018 年 3 月 29 日	AWS::EC2::LaunchTemplate Amazon EC2 インスタンスの起動テンプレートを作成するには、AWS::EC2::LaunchTemplate を使用します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2018 年 3 月 29 日	AWS::AutoScaling::AutoScalingGroup インスタンスを起動するために使用する起動テンプレートを指定するには、LaunchTemplate プロパティを使用します。 AWS::EC2::SpotFleet SpotFleetRequestConfigData プロパティタイプの LaunchTemplateConfigs プロパティを使用して、起動テンプレートと上書きを記述します。	2010-05-15
新しい Fn::Cidr 組み込み関数	2018 年 3 月 6 日	指定された Cidr アドレスブロックを返します。詳細については、「 Fn::Cidr 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
新しいリソース	2018 年 3 月 6 日	AWS::ApiGateway::VpcLink AWS::ApiGateway::VpcLink リソースを使用して、Amazon Virtual Private Cloud (VPC) のリソースにアクセスするための AWS::ApiGateway::RestApi の API ゲートウェイの VPC リンクを指定します。 AWS::GuardDuty::Master AWS::GuardDuty::Master リソースを使用して、GuardDuty プライマリアカウントを作成します。 AWS::GuardDuty::Member AWS::GuardDuty::Member リソースを使用して、GuardDuty メンバーアカウントを作成します。 AWS::SES::ConfigurationSet AWS::SES::ConfigurationSet リソースを使用して、送信する E メールに適用できるルールのグループを作成します。 AWS::SES::ConfigurationSetEventDestination AWS::SES::ConfigurationSetEventDestination リソースを使用して、構成セットイベント送信先を指定します。 AWS::SES::ReceiptFilter AWS::SES::ReceiptFilter リソースを使用して、IP アドレスまたは IP アドレス範囲から送信されるメールを受け入れるか拒否するかを指定します。	2010-05-15

変更	リリース日	説明	API バー ジョン
		AWS::SES::ReceiptRule AWS::SES::ReceiptRule リソースを使用して、所有する 1 つ以上の、E メールアドレスやドメインに代わってメールを受信したときに Amazon SES が実行するアクションを指定します。 AWS::SES::ReceiptRuleSet AWS::SES::ReceiptRuleSet リソースを使用して、Amazon SES の空のルールセットを指定します。 AWS::SES::Template AWS::SES::Template リソースを使用して、件名、HTML 部分、およびテキストのみの部分で構成される E メールの内容を指定します。	

変更	リリース日	説明	API バージョン
更新されたリソース	2018 年 3 月 6 日	AWS::AutoScaling::AutoScalingGroup AutoScalingGroupName プロパティを使用して、Auto Scaling グループの名前を指定します。 AWS::ApiGateway::RestApi ApiKeySourceType プロパティを使用して、使用量プランに沿ってリクエストを計測するための API キーのソースを指定します。 MinimumCompressionSize プロパティを使用して、API での圧縮を有効または無効にするために使用される、NULL が使用可能な整数を指定します。 AWS::ApplicationAutoScaling::ScalingPolicy TargetTrackingScalingPolicyConfiguration プロパティタイプの DisableScaleIn プロパティを使用して、ターゲット追跡ポリシーによるスケールインを無効にするかどうかを指定します。 AWS::EC2::SpotFleet LaunchSpecifications プロパティタイプの TagSpecifications プロパティを使用して、SpotFleet の作成時に適用するタグを指定します。 AWS::Elasticsearch::Domain Arn 属性を使用して、Fn::GetAtt でドメインの Amazon リソースネーム (ARN) を返します。 DomainArn の Fn::GetAtt 属性は廃止されました。	2010-05-15

変更	リリース日	説明	APIバージョン
		AWS::RDS::DBCluster DBClusterIdentifier プロパティを使用して DB クラスター識別子を指定します。 AWS::RDS::DBCluster DBClusterIdentifier プロパティを使用して DB クラスター識別子を指定します。 AWS::Redshift::Cluster ClusterIdentifier プロパティを使用してクラスターの一意の識別子を指定します。 AWS::Route53::HealthCheck HealthCheckConfig プロパティタイプの Regions プロパティを使用して、指定したエンドポイントを Route 53 ヘルスチェッカーで確認するリージョンを指定します。 AWS::SSM::Document Tags プロパティを使用して、ドキュメントに適用する CloudFormation リソースタグを指定します。	
更新されたリソース	2018 年 2 月 19 日	AWS::CodeBuild::Project Triggers プロパティを使用してプロジェクトのウェブフックの開始を設定し、コード変更がリポジトリにプッシュされるたびにソースコードが自動的に再構築されるようにします。このオプションは、CloudFormation の GitHub プロジェクトにのみ使用できます。GitHub Enterprise プロジェクトでは使用できません。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
更新されたリ ソース	2018 年 2 月 8 日	AWS::DynamoDB::Table SSESpecification プロパティを使用して、サー バー側の暗号化を有効にする設定を指定します。	2010-05-1 5
更新されたリ ソース	2018 年 2 月 5 日	AWS::CodeBuild::Project Source CodeBuild Project Source プロパティ タイプでは、次のプロパティを使用できます。 • GitCloneDepth プロパティを使用して、ダウ ンロードする履歴の深さを指定します。 • InsecureSsl プロパティを使用して、GitHub Enterprise プロジェクトリポジトリに接続する ときに SSL 警告を無視するかどうかを指定 します。	2010-05-1 5

変更	リリース日	説明	API バージョン
更新されたリソース	2018 年 1 月 23 日	AWS::AutoScaling::LifecycleHook LifecycleHookName プロパティを使用して、ライフサイクルフックの名前を指定します。 AWS::DynamoDB::Table AttributeDefinitions プロパティでは、更新されたときに置換が必要になりました。 AWS::EC2::Instance CreditSpecification プロパティを使用して、T2 インスタンスの CPU 使用率のクレジットオプションを指定します。 ElasticGpuSpecifications プロパティを使用して、アプリケーションのグラフィックスパフォーマンスを向上させるためにインスタンスにアタッチできる Elastic GPU、GPU リソースを指定します。 AWS::EC2::VPC InstanceTenancy プロパティでは、"dedicated" から "default" に更新するときに中断は不要になりました。 AWS::ECS::Service HealthCheckGracePeriodSeconds プロパティを使用して、Amazon ECS サービススケジューラが、タスクが最初に開始された後で異常な Elastic Load Balancing ターゲットのヘルスチェックを無視する期間 (秒単位) を指定します。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
		AWS::IoT::TopicRule DynamoDBAction プロパティタイプの RangeKeyField プロパティと RangeKeyValue プロパティは必須でなくなりました。 AWS::KinesisAnalytics::ApplicationOutput ApplicationOutput プロパティタイプの LambdaOutput プロパティを使用して、アプリケーション出力を設定する際に Lambda 関数を送信先として識別します。 AWS::Kinesis::Stream StreamEncryption プロパティを使用して、指定されたストリームに AWS KMS key を使用してサーバー側の暗号化を有効化または更新します。 AWS::Lambda::Function ReservedConcurrentExecutions プロパティを使用して、その関数用に予約する最大同時実行数を設定します。 AWS::RDS::DBSubnetGroup DBSubnetGroupName プロパティを使用して、DBサブネットグループの名前を指定します。 AWS::S3::Bucket BucketEncryption プロパティを使用して、Amazon S3 で管理されたキー、または SSE-S3 や AWS KMS keys (SSE-KMS) バケットとサーバー側暗号化によりデフォルトの暗号化を指定します。	

変更	リリース日	説明	API バー ジョン
		ReplicationRule プロパティタイプの <code>SourceSelectionCriteria</code> プロパティを使用して、レプリケートするソースオブジェクトを識別する際の追加のフィルタを指定します。 ReplicationDestination プロパティタイプでは、次のプロパティを使用できます。 • <code>AccessControlTranslation</code> プロパティを使用して、送信先バケットを所有している AWS アカウントのレプリカの所有権を指定します。 • <code>Account</code> プロパティを使用して、送信先バケット所有者のアカウント ID を指定します。 • <code>EncryptionConfiguration</code> プロパティを使用して、レプリケートされたオブジェクトの送信先であるバケットの暗号化関連情報を指定します。 AWS::SSM::Association <code>AssociationName</code> プロパティを使用して、SSM ドキュメントと、ドキュメントを処理する構成エージェントを含む EC2 インスタンス間の関連付けの名前を指定します。	
ロールバックトリガーが CloudFormation コンソールに追加されました。	2018 年 1 月 15 日	ロールバックトリガーを使用すると、スタックの作成および更新中にアプリケーションの状態を CloudFormation でモニタリングし、指定したいずれかのアラームのしきい値をアプリケーションが超過した場合に、そのオペレーションをロールバックできます。詳細については、「スタックオペレーションをモニタリングおよびロールバックする」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
更新されたリ ソース	2018 年 1 月 12 日	AWS::SSM::Parameter AllowedPattern プロパティを使用して、パラ メータ値を検証するために使用する正規表現を指定 します。	2010-05-1 5

変更	リリース日	説明	API バー ジョン
新しいリソース	2017 年 5 月 12 日	AWS::Inspector::AssessmentTarget AWS::Inspector::AssessmentTarget リソースを使用して、Amazon Inspector 評価ターゲットを作成します。 AWS::Inspector::AssessmentTemplate AWS::Inspector::AssessmentTemplate リソースを使用して、Amazon Inspector 評価テンプレートを作成します。 AWS::Inspector::ResourceGroup AWS::Inspector::ResourceGroup リソースを使用して、Amazon Inspector リソースグループを作成します。これは、Amazon Inspector 評価ターゲットを構成する AWS リソースを識別するタグを定義します。 AWS::ServiceDiscovery::Instance AWS::ServiceDiscovery::Instance リソースを使用して、Amazon Route 53 が作成するインスタンスに関する情報を指定します。 AWS::ServiceDiscovery::PrivateDnsNamespace AWS::ServiceDiscovery::PrivateDnsNamespace リソースを使用して、Amazon Route 53 のプライベート名前空間に関する情報を指定します。 AWS::ServiceDiscovery::PublicDnsNamespace AWS::ServiceDiscovery::PublicDnsNamespace リソースを使用して、Amazon Route 53 のパブリック名前空間に関する情報を指定します。	2010-05-15

変更	リリース日	説明	API バー ジョン
		AWS::ServiceDiscovery::Service AWS::ServiceDiscovery::Service リソースを使用して、インスタンスの登録時に Amazon Route 53 によって作成される最大 5 つのレコードとオプションのヘルスチェックのテンプレートを定義します。	
更新されたリソース	2017 年 5 月 12 日	AWS::KinesisAnalytics::Application Input プロパティタイプの InputProcessingConfiguration プロパティを使用してレコードをストリームから受信する際に変換します。	2010-05-15
更新されたリソース	2017 年 12 月 1 日	AWS::CodeBuild::Project BadgeEnabled プロパティを使用して、プロジェクトのビルドバッジのパブリックアクセス可能な URL を生成します。 Cache プロパティを使用して、ビルド依存関係のキャッシュ設定を構成します。 VpcConfig プロパティを使用して、CodeBuild が Amazon VPC のリソースにアクセスできるようにします。 EnvironmentVariable プロパティタイプの Type プロパティを使用して環境変数のタイプを指定します。	2010-05-15
新しいリソース	2017 年 11 月 30 日	AWS::Cloud9::EnvironmentEC2 AWS::Cloud9::EnvironmentEC2 リソースを使用して、AWS Cloud9 で Amazon EC2 開発環境を作成します。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
更新されたリ ソース	2017 年 11 月 29 日	AWS::ECS::TaskDefinition Cpu プロパティを使用して、タスクに必要な CPU ユニットの数を指定します。 ExecutionRoleArn プロパティを使用して、実行ロールの ARN を指定します。 Memory プロパティを使用して、タスクに必要なメモリ容量 (MiB) を指定します。 RequiresCompatibilities プロパティを使用して、タスクに必要な起動タイプを指定します。 AWS::ECS::Service LaunchType プロパティを使用して、サービスを実行する起動タイプを指定します。 NetworkConfiguration プロパティを使用して、サービスのネットワーク構成を指定します。 PlatformVersion プロパティを使用して、サービスを実行するプラットフォームのバージョンを指定します。	2010-05-1 5

変更	リリース日	説明	API バー ジ ョ ン
新しいリソース	2017 年 11 月 28 日	AWS::GuardDuty::Detector AWS::GuardDuty::Detector リソースを使用して、単一の Amazon GuardDuty デテクターを作成します。 AWS::GuardDuty::IPSet AWS::GuardDuty::IPSet リソースを使用して、Amazon GuardDutyIP セットを作成します。 AWS::GuardDuty::ThreatIntelSet AWS::GuardDuty::ThreatIntelSet リソースを使用して、ThreatIntelSet を作成します。	2010-05-15

変更	リリース日	説明	API バージョン
更新されたリソース	2017 年 11 月 28 日	AWS::CodeDeploy::Application ComputePlatform プロパティを使用して、アプリケーションをデプロイする CodeDeploy の AWS Lambda コンピューティングプラットフォームを指定します。 AWS::CodeDeploy::DeploymentGroup DeploymentStyle プロパティタイプの DeploymentType プロパティを使用して、Lambda コンピューティングプラットフォームでの Blue/Green デプロイを指定します。 AWS::EC2::SpotFleet SpotFleetRequestConfigData プロパティタイプの SpotPrice プロパティは省略可能になりました。 AWS::Lambda::Alias RoutingConfig プロパティを使用して、AWS Lambda 関数の 2 つの別のバージョンを指定し、各バージョンを呼び出すトラフィックの割合を指定できるようにします。	2010-05-15
新しい CodeDeployLambdaAliasUpdate 更新ポリシー	2017 年 11 月 28 日	CodeDeployLambdaAliasUpdate 更新ポリシーを使用して、AWS::Lambda::Alias リソースでのバージョンの変更時に CodeDeploy のデプロイを実行します。詳細については、「UpdatePolicy 属性」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
新しい SSM パラメータタイプ	2017 年 11 月 21 日	SSM パラメータタイプを使用して、Systems Manager パラメータストア内の既存のパラメータを使用します。注: CloudFormation は現在 SecureString タイプをサポートしていません。詳細については、「 SSM パラメータタイプ 」を参照してください。	2010-05-15
ResolvedValue データ型の新しい Parameter フィールド	2017 年 11 月 21 日	ResolvedValue フィールドは SSM パラメータのスタック定義に使用される値を返します。詳細については、「CloudFormation API リファレンス」の「 Parameter データタイプ」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 11 月 20 日	AWS::ApiGateway::ApiKey CustomerId プロパティを使用して、AWS Marketplace のお客様の識別子を指定します。 GenerateDistinctId プロパティを使用して、キー識別子が作成された API キーの値と異なるかどうかを指定します。 AWS::ApiGateway::Authorizer AuthType プロパティを使用して、機能に影響を与えずに Swagger インポートおよびエクスポートで使用するオプションの顧客定義のフィールドを指定します。 AWS::ApiGateway::DomainName EndpointConfiguration プロパティを使用して、API Gateway ドメイン名のエンドポイントタイプを指定します。 RegionalCertificateArn プロパティを使用して、ドメイン名のリージョン別エンドポイントが使用する証明書を参照します。 AWS::ApiGateway::Method Integration プロパティタイプおよび IntegrationResponse プロパティタイプで ContentHandling プロパティを使用して、リクエストペイロードのコンテンツタイプの変換の処理方法を指定します。	2010-05-15

変更	リリース日	説明	API バー ジョン
		AWS::ApiGateway::RestApi EndpointConfiguration プロパティを使用して API Gateway REST API のエンドポイントタイプを指定します。 AWS::ApplicationAutoScaling::ScalableTarget ScheduledActions プロパティを使用して、アプリケーションの Auto Scaling のスケーラブルなターゲットのスケジュールされたアクションを指定します。 AWS::ECR::Repository LifecyclePolicy プロパティを使用して、Amazon ECR リポジトリのライフサイクルポリシーを指定します。 AWS::ECS::TaskDefinition ContainerDefinition プロパティタイプの LinuxParameters プロパティを使用して Amazon ECS コンテナの Linux 固有オプションを指定します。 AWS::ElastiCache::ReplicationGroup AtRestEncryptionEnabled プロパティを使用して、保管時暗号化を有効にします。 AuthToken プロパティを使用して、パスワード保護されたサーバーへのアクセスに使用されるパスワードを指定します。 TransitEncryptionEnabled プロパティを使用して、転送中の暗号化を有効にします。	

変更	リリース日	説明	API バー ジヨ ン
		AWS::ElasticLoadBalancingV2::TargetGroup Fn::GetAtt 関数で TargetGroupName 属性を使用して、Elastic Load Balancing ターゲットグループの名前を取得します。 AWS::Elasticsearch::Domain VPCOptions プロパティを使用して、OpenSearch Service ドメインの VPC 構成を指定します。 AWS::EMR::Cluster EbsRootVolumeSize プロパティを使用して、Amazon EMR クラスターの EBS ルートボリュームのサイズを指定します。 AWS::RDS::DBInstance SourceRegion および KmsKeyId プロパティを使用して、クロスリージョンソース DB インスタンスから暗号化されたリードレプリカを作成します。 AWS::Route53::HostedZone QueryLoggingConfig プロパティを使用して、DNS クエリログ記録の構成を指定します。	
カスタムリソース NoEcho オブジェクトの新しい Response フィールド	2017 年 11 月 20 日	オプションの NoEcho フィールドを使用して、カスタムリソースの出力をマスクできるようになりました。詳細については、「カスタムリソースの応答オブジェクト」を参照してください。 対応する noEcho パラメータは、send メソッドでサポートされています。詳細については、「cfn 応答モジュール」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
スタックセットでスタックインスタンスの上書きが追加されました。	2017 年 11 月 17 日	CloudFormation スタックセットでは、アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書きできます。スタックインスタンスの作成時、または既存のスタックインスタンスの更新時にパラメータ値を上書きできます。詳細については、 スタックインスタンスのパラメータを上書きする 参照してください。	2010-05-15
更新されたリソース	2017 年 11 月 15 日	AWS::StepFunctions::StateMachine AWS::StepFunctions::StateMachine を使用して、ステートマシンの作成時に StateMachineName を指定します。DefinitionString および RoleArn の両方が、ステートマシンを置き換えることなく更新されます。	2010-05-15
StackSet はスタックセットごとに最大 500 のスタックインスタンスをサポートするようになりました。	2017 年 11 月 6 日	スタックセットごとに最大 500 のスタックインスタンスを作成できるようになりました。CloudFormation の制限に関する詳細については、「 CloudFormation クォータを理解する 」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
新しいリソース	2017 年 11 月 2 日	AWS::CloudFront::CloudFrontOriginAccessIdentity AWS::CloudFront::CloudFrontOriginAccessIdentity リソースを使用して、CloudFront デイストリビューションのオリジンと関連づける、Amazon CloudFront オリジンアクセスアイデンティティを指定します。 AWS::CloudFront::StreamingDistribution CloudFront の Adobe Real-Time Messaging Protocol (RTMP) ストリーミング配信を指定するには、AWS::CloudFront::StreamingDistribution リソースを使用します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 11 月 2 日	AWS::ApiGateway::Deployment StageDescription プロパティタイプの StageName プロパティは廃止されました。 AWS::ApiGateway::Method OperationName プロパティを使用して、API Gateway メソッドに分かりやすい名前を割り当てます。 RequestValidatorId プロパティを使用してメソッドにリクエスト検証元を関連付けます。 AWS::AutoScaling::AutoScalingGroup LifecycleHookSpecificationList プロパティを使用して、Auto Scaling がインスタンスを起動または終了するときに実行するアクションを指定します。 AWS::CloudFront::Distribution Tags プロパティを使用して、CloudFront ディストリビューションに関連付ける任意のタグのセット (キーと値のペア) を指定します。 CacheBehavior プロパティタイプおよび DefaultCacheBehavior プロパティタイプで LambdaFunctionAssociations プロパティを使用して、CloudFront ディストリビューションに関連付ける Lambda 関数を指定します。 CustomOriginConfig プロパティタイプの OriginKeepaliveTimeout プロパティを使用してカスタムキープアライブタイムアウトを指定	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
		し、OriginReadTimeout プロパティを使用してカスタムオリジンの読み取りタイムアウトを指定します。 DistributionConfig プロパティタイプの IPV6Enabled プロパティを使用して、ディストリビューションの IPv6 アドレスを持つ IPv6 DNS リクエストに CloudFront が応答するかどうかを指定します。 AWS::CodeDeploy::DeploymentGroup LoadBalancerInfo プロパティタイプの TargetGroupInfoList プロパティを使用して、デプロイで使用する Elastic Load Balancing のターゲットグループに関する情報を指定します。 AWS::EC2::SecurityGroup、AWS::EC2::SecurityGroupEgress、および AWS::EC2::SecurityGroupIngress Description プロパティを使用して、セキュリティグループの説明を指定します。 AWS::EC2::Subnet Ipv6CidrBlock プロパティで No interruption 更新がサポートされるようになりました。 AWS::EC2::VPNGateway AmazonSideAsn プロパティを使用して、BGP セッションの Amazon 側のプライベート自律システム番号 (ASN) を指定します。 AWS::EC2::VPNConnection VpnTunnelOptionsSpecifications プロパティを使用して VPN 接続のトンネルオプションを設定します。	

変更	リリース日	説明	API バー ジョ ン
		AWS::ElasticBeanstalk::ConfigurationTemplate および AWS::ElasticBeanstalk::Environment ConfigurationOptionSetting プロパティタイプおよび OptionSetting プロパティタイプで ResourceName プロパティを使用して、時間ベースのスケールリング構成オプションのリソース名を指定します。 AWS::EMR::Cluster CustomAmiId プロパティを使用してクラスターのカスタム Amazon Linux AMI を指定します。 AWS::KinesisFirehose::DeliveryStream Arn 関数の Fn::GetAtt 属性を使用して、配信ストリームの Amazon リソースネーム (ARN) を取得します。 AWS::KMS::Key Tags プロパティを使用して、カスタマーマネージドキーに関連付ける任意のタグのセット (キーと値のペア) を指定します。 AWS::OpsWorks::Layer および AWS::OpsWorks::Stack Tags プロパティを使用して、OpsWorks レイヤーまたはスタックに関連付ける任意のタグのセット (キーと値のペア) を指定します。 AWS::RDS::OptionGroup OptionConfiguration プロパティタイプの OptionVersion プロパティを使用してオプションのバージョンを指定します。	

変更	リリース日	説明	API バー ジヨ ン
		AWS::S3::Bucket AnalyticsConfigurations プロパティを使用して、Amazon S3 バケットの分析フィルターを設定します。	

変更	リリース日	説明	APIバージョン
新しいリソース	2017 年 10 月 24 日	AWS::Glue::Classifier AWS::Glue::Classifier リソースを使用して、AWS Glue 分類子を作成します。 AWS::Glue::Connection AWS::Glue::Connection リソースを使用してデータソースへの AWS Glue 接続を指定します。 AWS::Glue::Crawler AWS Glue クローラを指定するには、AWS::Glue::Crawler リソースを使用します。 AWS::Glue::Database AWS::Glue::Database リソースを使用して AWS Glue データベースを作成します。 AWS::Glue::DevEndpoint リモートデバッグ ETL スクリプトの開発エンドポイントを指定するには、AWS::Glue::DevEndpoint リソースを使用します。 AWS::Glue::Job AWS::Glue::Job リソースを使用してデータカタログに AWS Glue ジョブを指定します。 AWS::Glue::Partition AWS::Glue::Partition リソースを使用して、テーブルデータのスライスを表す AWS Glue パーティションを作成します。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
		AWS::Glue::Table AWS::Glue::Table リソースを使用して AWS Glue テーブルを作成します。 AWS::Glue::Trigger AWS::Glue::Trigger リソースを使用して AWS Glue ジョブを実行するトリガーを指定します。	
新しいリソース	2017 年 10 月 11 日	AWS::SSM::MaintenanceWindow AWS::SSM::MaintenanceWindow リソースを使用して、AWS Systems Manager メンテナンスウィンドウを作成します。 AWS::SSM::MaintenanceWindowTarget AWS::SSM::MaintenanceWindowTarget リソースを使用して、ターゲットをメンテナンスウィンドウに登録します。 AWS::SSM::MaintenanceWindowTask AWS::SSM::MaintenanceWindowTask リソースを使用して、メンテナンスウィンドウのタスクを定義します。 AWS::SSM::PatchBaseline AWS::SSM::PatchBaseline リソースを使用して、Systems Manager パッチベースラインを定義します。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
新しいリソース	2017 年 10 月 10 日	AWS::ElasticLoadBalancingV2::ListenerCertificate AWS::ElasticLoadBalancingV2::ListenerCertificate リソースを使用して、Elastic Load Balancing リスナーの証明書を指定します。	2010-05-15
新しいリソース	2017 年 9 月 27 日	AWS::Athena::NamedQuery AWS::Athena::NamedQuery リソースを使用して Amazon Athena クエリを作成します。	2010-05-15

変更	リリース日	説明	API バージョン
更新されたリソース	2017 年 9 月 27 日	AWS::EC2::NatGateway Tags プロパティを使用して、NAT ゲートウェイのリソースタグを指定します。 AWS::ElasticBeanstalk::Application ResourceLifecycleConfig プロパティを使用して、アプリケーションに属するリソースのライフサイクル設定と、そのライフサイクル設定を適用するために Elastic Beanstalk が引き受けるサービスロールを定義します。 AWS::ElasticBeanstalk::ConfigurationTemplate および AWS::ElasticBeanstalk::Environment PlatformArn プロパティを使用して、Elastic Beanstalk のカスタムプラットフォームを指定します。 AWS::ElasticLoadBalancingV2::TargetGroup TargetDescription プロパティタイプの AvailabilityZone プロパティを使用して、IP アドレスを登録するアベイラビリティゾーンを指定します。 AWS::Events::Rule Target プロパティタイプの以下のプロパティを使用して、イベントの入力変換や、Amazon ECS タスクおよび Kinesis ストリームのターゲットの設定を行います。 - EcsParameters - InputTransformer - KinesisParameters	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
		RunCommandParameters AWS::KinesisFirehose::DeliveryStream ストリームタイプを指定するには DeliveryStreamType プロパティ、配信ストリームのソースとして使用される Kinesis ストリームのストリームとロールの ARN を指定するには KinesisStreamSourceConfiguration プロパティを使用します。 AWS::RDS::DBInstance Engine プロパティで、oracle-se または oracle-se1 を指定した場合は、oracle-se2 に更新することができます。データベースインスタンスを置き換える必要はありません。 AWS::S3::Bucket AccelerateConfiguration プロパティを使用して、Amazon S3 バケットの Transfer Acceleration 状態を設定します。	
スタックの削除保護を追加しました。	2017 年 9 月 26 日	スタックの削除保護を有効にすると、スタックが誤って削除されるのを防ぐことができます。削除保護が有効になっているスタックを削除することはできません。詳細については、「 スタックの削除保護 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
1.4-22 以降のバージョンから、デフォルトの umask 値を変更しました。	2017 年 9 月 14 日	cfn-hup.conf 構成ファイルの umask パラメータのデフォルト値が 022 に設定されました。詳細については、「 cfn-hup 」を参照してください。	
更新されたリソース	2017 年 9 月 7 日	AWS::ElasticLoadBalancingV2::LoadBalancer SubnetMappings プロパティを使用して、ロードバランサーにアタッチするサブネットの ID を指定します。 Type プロパティを使用して、作成するロードバランサーのタイプを指定します。 AWS::ElasticLoadBalancingV2::TargetGroup TargetType プロパティを使用して、このターゲットグループのターゲットの登録タイプを指定します。	2010-05-15
CloudFormationAPI に追加されたロールバックトリガー	2017 年 8 月 31 日	ロールバックトリガーを使用すると、スタックの作成および更新中にアプリケーションの状態を CloudFormation でモニタリングし、指定したいずれかのアラームのしきい値をアプリケーションが超過した場合に、そのオペレーションをロールバックできます。詳細については、「CloudFormation API リファレンス」の「 RollbackConfiguration 」を参照してください。	2010-05-15
cfn-hup.conf ファイルの新しい umask パラメータ	2017 年 8 月 31 日	cfn-hup.conf 構成ファイルで umask パラメータを使用して、cfn-hup デーモン (バージョン 1.4-21) によって使用されるファイルのアクセス許可を管理します。詳細については、「 cfn-hup 」を参照してください。	

変更	リリース日	説明	API バー ジヨ ン
VPC サイズ設 定のサポート について更新 されたリソー ス	2017 年 8 月 29 日	AWS::EC2::VPC::CidrBlock CidrBlock プロパティを使用して、IPv4 CIDR ブ ロックを VPC に関連付けます。 AWS::EC2::VPC CidrBlockAssociations 関数で Fn::GetAt t 属性を使用して、VPC の IPv4 CIDR ブロック関 連付け ID のリストを取得します。	2010-05-1 5

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 8 月 23 日	AWS::S3::Bucket Rule プロパティタイプの <code>TagFilters</code> プロパティを使用して、Amazon S3 バケット内のオブジェクトのサブセットを識別するために使用するタグを指定します。 <code>MetricsConfiguration</code> プロパティを使用して、Amazon S3 バケットからの CloudWatch リクエストメトリクスのメトリクス構成を指定します。 AWS::IoT::TopicRule Action プロパティタイプの <code>DynamoDBv2Action</code> プロパティを使用して、DynamoDB テーブルにデータを書き込む AWS IoT アクションを定義します。 Action プロパティタイプで、<code>DynamoDBAction</code> プロパティが <code>HashKeyType</code> プロパティおよび <code>RangeKeyType</code> プロパティをサポートするようになりました。 AWS::Lambda::Permission <code>EventSourceToken</code> プロパティを使用して、関数を呼び出すプリンシパルによって指定される必要のある一意のトークンを指定します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しい擬似パラメータ	2017 年 8 月 23 日	<code>AWS::Partition</code> 擬似パラメータを使用して、リソースがあるパーティションを返します。 <code>AWS::URLSuffix</code> 擬似パラメータを使用して、ドメインのサフィックスを返します。 詳細については、「擬似パラメータのリファレンス」を参照してください。	2010-05-15
DAX サポートの新しいリソース	2017 年 8 月 22 日	AWS::DAX::Cluster <code>AWS::DAX::Cluster</code> リソースを使用して、Amazon DynamoDB で使用する DAX クラスターを作成します。 AWS::DAX::ParameterGroup <code>AWS::DAX::ParameterGroup</code> リソースを使用して、Amazon DynamoDB で使用するパラメータグループを作成します。 AWS::DAX::SubnetGroup <code>AWS::DAX::SubnetGroup</code> リソースを使用して、DAX (DynamoDB Accelerator) で使用するサブネットグループを作成します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2017 年 8 月 18 日	AWS::ApiGateway::DocumentationPart および AWS::ApiGateway::DocumentationPart AWS::ApiGateway::DocumentationPart リソースを使用して、API Gateway API のドキュメントを作成します。 AWS::ApiGateway::GatewayResponse AWS::ApiGateway::GatewayResponse リソースを使用して、API Gateway API のカスタムレスポンスを作成します。 AWS::ApiGateway::RequestValidator AWS::ApiGateway::RequestValidator リソースを使用して、API Gateway API への受信リクエストの検証ルールを設定します。 AWS::EC2::NetworkInterfacePermission AWS::EC2::NetworkInterfacePermission リソースを使用して、AWS アカウントにネットワークインターフェースに対するアクセス権限を付与します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 8 月 18 日	AWS::ApiGateway::Stage DocumentationVersion プロパティを使用して、API ドキュメントのバージョニングが有効なスナップショットを指定します。 AWS::AutoScaling::ScalingPolicy TargetTrackingConfiguration プロパティを使用して、Auto Scaling のターゲット追跡スケールリングポリシー構成を指定します。 AWS::CloudTrail::Trail EventSelectors プロパティを使用して、Amazon S3 データイベントをサポートします。 AWS::CodeDeploy::DeploymentGroup LoadBalancerInfo および DeploymentStyle プロパティを使用して、インプレースデプロイ用の Elastic Load Balancing ロードバランサーを指定します。 AutoRollbackConfiguration プロパティを使用して、デプロイの自動ロールバックを設定します。 AWS::EC2::SpotFleet SpotFleetRequestConfigData プロパティタイプの ReplaceUnhealthyInstances プロパティを使用してスポットフリートで異常なインスタンスを置き換えるかどうかを指定し、Type プロパティを使用してリクエストのタイプを指定します。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
		AWS::EC2::Subnet AssignIpv6AddressOnCreation および Ipv6CidrBlock プロパティを使用して、IPv6 CIDR ブロックを関連付けたサブネットを作成します。 AWS::KinesisFirehose::DeliveryStream ExtendedS3DestinationConfiguration プロパティを使用して、Amazon S3 のターゲットを設定します。 各送信先構成内で ProcessingConfiguration サブプロパティを使用して、受信ソースデータを変換してからターゲットに配信する Lambda 関数を呼び出します。 AWS::RDS::DBCluster および AWS::RDS::DBInstance デフォルトの DeletionPolicy は、AWS::RDS::DBCluster リソースの場合も、DBCluster Identifier プロパティを指定しない AWS::RDS::DBInstance リソースの場合も、Snapshot になりました。詳細については、「DeletionPolicy 属性」を参照してください。 AWS::S3::Bucket Rule プロパティタイプの AbortIncompleteMultipartUpload プロパティを使用して、Amazon S3 バケットへの不完全なマルチパートアップロードを中止するライフサイクルルールを指定します。	

変更	リリース日	説明	API バー ジョン
		AWS::SQS::Queue KmsMasterKeyId および KmsDataKeyReusePeriodSeconds プロパティを使用して、Amazon SQS のサーバー側暗号化を設定します。 以下のリソースの Arn 組み込み関数に Fn::GetAtt 属性を追加しました。 • AWS::CloudTrail::Trail。SnsTopicArn も追加されました。 • AWS::CloudWatch::Alarm • AWS::DynamoDB::Table • AWS::ECS::Cluster • AWS::IoT::Policy • AWS::IoT::TopicRule • AWS::Logs::Destination	
CodePipeline アーティファクトでのスタックタグのサポート	2017 年 8 月 18 日	テンプレート構成ファイルでスタックのタグを指定して、CodePipeline パイプラインのアーティファクトとして使用できるようになりました。指定したタグは、テンプレート構成ファイルを使用して作成したスタックに適用されます。詳細については、「CloudFormation アーティファクト」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
暗号化されたファイルシステムの作成	2017 年 8 月 14 日	AWS::EFS::FileSystem Encrypted プロパティを使用して、Amazon EFS ファイルシステムを作成中に暗号化します。 KmsKeyId プロパティを使用して、暗号化されたファイルシステムの保護に使用するカスタムのカスタマーマネージドキーをオプションで指定します。	2010-05-15
AWS Batch 対応の新しいリソース	2017 年 8 月 8 日	AWS::Batch::ComputeEnvironment AWS::Batch::ComputeEnvironment リソースを使用して、AWS Batch コンピューティング環境を定義します。 AWS::Batch::JobDefinition AWS::Batch::JobDefinition リソースを使用して、AWS Batch ジョブ定義のパラメータを指定します。 AWS::Batch::JobQueue AWS::Batch::JobQueue リソースを使用して、AWS Batch ジョブキューを定義します。	2010-05-15

変更	リリース日	説明	API バージョン
Amazon Managed Service for Apache Flink サポートの新しいリソース	2017 年 7 月 28 日	AWS::KinesisAnalytics::Application AWS::KinesisAnalytics::Application リソースを使用して、Amazon Managed Service for Apache Flink アプリケーションを作成します。 AWS::KinesisAnalytics::ApplicationOutput AWS::KinesisAnalytics::ApplicationOutput リソースを使用して Amazon Managed Service for Apache Flink アプリケーションに外部接続先を追加します。 AWS::KinesisAnalytics::ApplicationReferenceDataSource AWS::KinesisAnalytics::ApplicationReferenceDataSource リソースを使用して既存の Amazon Managed Service for Apache Flink アプリケーションにリファレンスデータソースを追加します。	2010-05-15
StackSets を使用して、アカウントとリージョンのスタックを一元管理します。	2017 年 7 月 25 日	StackSets を使用すると、1 回のオペレーションで、複数のアカウントとリージョンでスタックを作成、更新、または削除することができます。管理者アカウントを使用して、CloudFormation テンプレートを定義および管理し、指定のリージョン間で選択したターゲットアカウントにスタックをプロビジョニングするためのベースとして使用します。詳細については、「StackSets を使用したアカウントとリージョン全体でのスタックの管理」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
クライアントのリクエストトークンでスタックイベントを表示する	2017 年 14 月 7 日	コンソールで、スタックオペレーションは、クライアントのリクエストトークンの [イベント] タブに表示されます。指定されたスタックオペレーションでトリガーされるイベントはすべて、同一クライアントのリクエストトークンに割り当てられ、オペレーションの追跡に使用できます。詳細については、『AWS CloudFormation API リファレンス』の「AWS マネジメントコンソールで CloudFormation スタックデータとリソースを表示する」と「 StackEvent 」を参照してください。	2010-05-15
スタックのクイック作成リンクを使用する	2017 年 14 月 7 日	クイック作成リンクを使用すると、スタックを作成して、すぐに作業を開始できます。テンプレート URL、スタック名、テンプレートパラメータを使用して、事前に [Create Stack Wizard (スタック作成ウィザード)] ページを 1 つ設定します。詳細については、「 スタックのクイック作成リンクを作成する 」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
AWS Database Migration Service 対応の新しいリソース	2017 年 7 月 12 日	AWS::DMS::Certificate AWS::DMS::Certificate リソースを使用して、AWS DMS エンドポイントとレプリケーションインスタンス間の接続を暗号化する SSL 証明書を作成します。 AWS::DMS::Endpoint AWS::DMS::Endpoint リソースを使用して、AWS DMS エンドポイントを作成します。 AWS::DMS::EventSubscription AWS::DMS::EventSubscription リソースを使用して、Amazon Simple Notification Service を通じて AWS DMS イベントの通知を受け取ります。 AWS::DMS::ReplicationInstance AWS::DMS::ReplicationInstance リソースを使用して、AWS DMS レプリケーションインスタンスを作成します。 AWS::DMS::ReplicationSubnetGroup AWS::DMS::ReplicationSubnetGroup リソースを使用して、AWS DMS レプリケーションのサブネットグループを作成します。 AWS::DMS::ReplicationTask AWS::DMS::ReplicationTask リソースを使用して、AWS DMS レプリケーションタスクを作成します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2017 年 7 月 5 日	AWS::CloudWatch::Dashboard AWS::CloudWatch::Dashboard リソースを使用して、CloudWatch コンソールのカスタム CloudWatch ダッシュボードを指定します。 AWS::ApiGateway::DomainName AWS::ApiGateway::DomainName リソースを使用して、Amazon API Gateway にデプロイされた API のわかりやすいカスタム URL を指定します。 AWS::EC2::EgressOnlyInternetGateway AWS::EC2::EgressOnlyInternetGateway リソースを使用して、VPC の Egress-Only インターネットゲートウェイを作成します。 InstanceFleetConfig InstanceFleetConfig リソースを使用して、Amazon EMR クラスターのスポットインスタンスフリートを設定します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 7 月 5 日	AWS::ApiGateway::RestApi BinaryMediaTypes プロパティを使用して、サポートされているバイナリメディアタイプを指定します。 AWS::ApplicationAutoScaling::ScalingPolicy TargetTrackingScalingPolicyConfiguration プロパティを使用して、ターゲット追跡スケールリングポリシー構成を指定します。 AWS::CloudTrail::Trail TrailName プロパティを使用して、AWS CloudTrail リソースのカスタム名を指定します。 Tags プロパティを使用して、リソースタグを指定します。 AWS::CodeDeploy::DeploymentGroup AlarmConfiguration プロパティを使用して、デプロイグループのアラームを設定します。 TriggerConfigurations プロパティを使用して、デプロイグループの通知トリガーを設定します。 AWS::EMR::Cluster JobFlowInstancesConfig プロパティタイプの CoreInstanceFleet プロパティと MasterInstanceFleet プロパティを使用して、Amazon EMR クラスターのスポットインスタンスフリートを設定します。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
		AWS::DynamoDB::Table TimeToLiveSpecification プロパティを使用して、Amazon DynamoDB テーブルの有効期限 (TTL) 設定を指定します。 Tags プロパティを使用して、DynamoDB テーブルのリソースタグを指定します。 AWS::EC2::Instance IamInstanceProfile プロパティで No interruption 更新がサポートされるようになりました。 AWS::EC2::Route EgressOnlyInternetGatewayId プロパティを使用して、EC2 ルートの Egress-Only インターネットゲートウェイを指定します。 AWS::Kinesis::Stream RetentionPeriodHours プロパティを使用して、シャードに保存されたデータレコードがアクセス可能な状態で維持される時間数を指定します。 AWS::RDS::DBCluster ReplicationSourceIdentifier プロパティを使用して、別の DB クラスターまたは Amazon RDS MySQL DB インスタンスのリードレプリカとして DB クラスターを作成します。	

変更	リリース日	説明	APIバージョン
		AWS::Redshift::Cluster LoggingProperties プロパティを使用して、監査ログファイルを作成し、Amazon S3 に保存します。	
新しいリソース	2017 年 6 月 6 日	AWS::EMR::SecurityConfiguration AWS::EMR::SecurityConfiguration リソースを使用して、サービスに保存されてクラスターの作成時に指定できるセキュリティ構成を作成します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 6 月 6 日	AWS::AutoScaling::LifecycleHook NotificationTargetARN および RoleARN プロパティは省略可能になりました。 AWS::CloudWatch::Alarm これで EvaluateLowSampleCountPercentile リソースの作成時に ExtendedStatistic、TreatMissingData、および AWS::CloudWatch::Alarm プロパティを使用できるようになりました。 AWS::EC2::SpotFleet CloudFormation では、スポットフリートのプロパティに対する可変の変更がサポートされています。 SpotFleetRequestConfigData プロパティの以下のプロパティで Replacement 更新がサポートされています。 • AllocationStrategy • IamFleetRole • LaunchSpecifications • SpotPrice • TerminateInstancesWithExpiration • ValidFrom • ValidUntil SpotFleetRequestConfigData プロパティの以下のプロパティで No interruption 更新がサポートされています。 • ExcessCapacityTerminationPolicy	2010-05-15

変更	リリース日	説明	API バー ジョン
		• TargetCapacity AWS::EMR::InstanceGroupConfig CloudFormation で Amazon EMR タスクインスタンスグループの Auto Scaling がサポートされるようになりました。 AWS::Events::Rule RoleArn プロパティは Rule リソースでは廃止されました。 Target プロパティタイプの RoleArn プロパティを使用して、ターゲットに使用する IAM ロールを指定します。 AWS::Kinesis::Stream ShardCount プロパティで No interruption 更新がサポートされるようになりました。 AWS::Lambda::Function TracingConfig プロパティを使用して、Lambda 関数の追跡設定を設定します。 AWS::Redshift::Cluster、AWS::Redshift::ClusterParameterGroup、AWS::Redshift::ClusterSecurityGroup、および AWS::Redshift::ClusterSubnetGroup Tags プロパティを使用して、リソースタグを指定します。 AWS::RDS::DBCluster ReadEndpoint.Address 組み込み関数に Fn::GetAtt 属性を追加しました。	

変更	リリース日	説明	API バー ジヨ ン
		AWS::S3::Bucket Arn 組み込み関数に Fn::GetAtt 属性を追加しました。	

変更	リリース日	説明	API バージョン
新しいリソース	2017 年 5 月 11 日	以下の新しいリソースで Elastic Load Balancing (ELB) Application Load Balancer での AWS WAF の使用がサポートされています。 AWS::WAFRegional::ByteMatchSet AWS::WAFRegional::ByteMatchSet リソースを使用して、調査するウェブリクエストの一部を識別します。 AWS::WAFRegional::IPSet AWS::WAFRegional::IPSet リソースを使用して、リクエストの送信元 IP アドレスに基づいて許可またはブロックするウェブリクエストを指定します。 AWS::WAFRegional::Rule AWS::WAFRegional::Rule リソースを使用して、許可、ブロック、またはカウントするウェブリクエストを識別する IPSet、ByteMatchSet、SqlInjectionMatchSet オブジェクトの組み合わせを指定します。 AWS::WAFRegional::SizeConstraintSet AWS::WAFRegional::SizeConstraintSet リソースを使用して、ウェブリクエストのサイズをチェックするのに使用するサイズ制約と、チェックするリクエストの部分を指定します。 AWS::WAFRegional::SqlInjectionMatchSet AWS::WAFRegional::SqlInjectionMatchSet リソースを使用して、ウェブリクエストの特定	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
		の部分で悪意のある SQL コードを含むリクエストを許可、ブロック、またはカウントします。 AWS::WAFRegional::WebACL AWS::WAFRegional::WebACL リソースを使用して、許可、ブロック、またはカウントするウェブリクエストを識別します。 AWS::WAFRegional::WebACLAssociation AWS::WAFRegional::WebACLAssociation リソースを使用して、ウェブアクセスコントロールグループ (ACL) をリソースと関連付けます。 AWS::WAFRegional::XssMatchSet AWS::WAFRegional::XssMatchSet リソースを使用して、AWS WAF でクロスサイトスクリプト攻撃を調べるウェブリクエストの一部と、調査のヘッダーの名前を指定します。	

変更	リリース日	説明	APIバージョン
新しいリソース	2017 年 4 月 28 日	AWS::Cognito::IdentityPool AWS::Cognito::IdentityPool リソースを使用して、Amazon Cognito ID プールを作成します。 AWS::Cognito::IdentityPoolRoleAttachment AWS::Cognito::IdentityPoolRoleAttachment リソースを使用して、Amazon Cognito ID プールのロール構成を管理します。 AWS::Cognito::UserPool AWS::Cognito::UserPool リソースを使用して、Amazon Cognito ユーザープールを作成します。 AWS::Cognito::UserPoolClient AWS::Cognito::UserPoolClient リソースを使用して、ユーザープールクライアントを作成します。 AWS::Cognito::UserPoolGroup AWS::Cognito::UserPoolGroup リソースを使用して、Amazon Cognito ユーザープールでユーザーグループを作成します。 AWS::Cognito::UserPoolUser AWS::Cognito::UserPoolUser リソースを使用して、Amazon Cognito ユーザープールユーザーを作成します。 AWS::Cognito::UserPoolUserToGroupAttachment AWS::Cognito::UserPoolUserToGroupAttachment リソースを使用して、Amazon Cognito	2010-05-15

変更	リリース日	説明	APIバージョン
		ユーザープールグループにユーザーをアタッチします。	

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 4 月 28 日	SourceDetails MaximumExecutionFrequency リソースの AWS::Config::ConfigRule サブプロパティを使用して、定期的トリガーを使用するカスタムルールの評価を実行します。 AWS::EC2::Volume 現在 CloudFormation で Amazon Elastic Block Store (Amazon EBS) の Elastic Volumes をサポートしています。現在 VolumeType、Size、および Iops の 3 つのプロパティで No interruption 更新をサポートしています。 AWS::EC2::SecurityGroup GroupName プロパティを使用して、Amazon EC2 セキュリティグループの名前を指定します。 AWS::ECS::Service AWS::ECS::Service には、Placement Constraints、PlacementStrategies、および ServiceName という 3 つの新しいプロパティがあります。 AWS::ECS::TaskDefinition PlacementConstraints プロパティを使用して、サービスにおけるタスクの配置制約事項を定義します。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
		AWS::ElastiCache::ReplicationGroup ConfigurationEndPoint.Address 属性および ConfigurationEndPoint.Port 属性が Fn::GetAtt 組み込み関数に追加されました。 AWS::ElasticLoadBalancingV2::LoadBalancer IpAddressType プロパティを使用して、ロードバランサーのサブネットによって使用される IP アドレスのタイプを指定します。 AWS::EMR::Cluster CloudFormation では Amazon EMR クラスターの Auto Scaling がサポートされるようになりました。 AWS::IAM::ManagedPolicy ManagedPolicyName プロパティを使用して、IAM 管理ポリシーのカスタム名を指定します。 AWS::Lambda::Function Tags プロパティを使用して、Lambda 関数にタグを追加します。 AWS::OpsWorks::Instance Fn::GetAtt 組み込み関数に次の属性が追加されました。AvailabilityZone 、 PrivateDnsName 、 PrivateIp 、 および PublicDnsName 。 AWS::OpsWorks::UserProfile SshUsername プロパティを使用して、ユーザーの SSH 名を指定します。	

変更	リリース日	説明	API バー ジョン
		SshUsername 組み込み関数に Fn::GetAtt 属性を追加しました。 AWS::Redshift::Cluster IamRoles プロパティを使用して、Amazon Redshift クラスターが他の AWS サービスにアクセスするために使用可能な 1 つ以上の AWS Identity and Access Management ロールのリストを提供します。	
CloudFormation デザイナーを使用して YAML および JSON でテンプレートを編集する	2017 年 4 月 6 日	デザイナーを使用して CloudFormation テンプレートを作成すると、統合エディタで YAML と JSON の両方のテンプレートを編集できるようになりました。テンプレート作成言語に応じて、JSON テンプレートを YAML に変換することもでき、その逆も可能です。詳細については、「CloudFormation デザイナー とは」を参照してください。	2010-05-15
新しいリソース	2017 年 4 月 6 日	AWS::SSM::Parameter AWS::SSM::Parameter リソースを使用して、パラメータストアに SSM パラメータを作成します。	2010-05-15
AWS::Include 変換	2017 年 3 月 28 日	AWS::Include 変換を使用して、Amazon S3 バケットに保存された再利用可能なスニペットを参照します。詳細については、「AWS::Include 変換」を参照してください。	2010-05-15
Amazon VPC を別のアカウントにピア接続する	2017 年 3 月 28 日	CloudFormation を使用して Amazon VPC を別の AWS アカウントの VPC とピア接続できるようになりました。詳細については、「別の AWS アカウントの Amazon VPC とピア接続する」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
新しいリソース	2017 年 3 月 28 日	AWS::ApiGateway::UsagePlanKey AWS::ApiGateway::UsagePlanKey リソース を使用して、使用量プランキーを関連付けてその使 用量プランが適用されるユーザーを決定します。	2010-05-1 5

変更	リリース日	説明	API バージョン
更新されたリソース	2017 年 3 月 28 日	AWS::EC2::VPCPeeringConnection PeerOwnerId プロパティおよび PeerRoleArn プロパティを使用して、別の AWS アカウントの VPC にピア接続します。 詳細については、「別の AWS アカウントの Amazon VPC とピア接続する」を参照してください。 AWS::IAM::InstanceProfile InstanceProfileName を使用してインスタンスプロファイルを設定します。 AWS::Lambda::Function DeadLetterConfig プロパティを使用して、AWS Lambda が処理できないイベントの対処方法を設定します。 Node.js v0.10 は Runtime プロパティでサポートされなくなりました。 AWS::Route53::HealthCheck HealthCheckConfig HealthCheckConfig プロパティには次の 7 つの新しいリソースサブプロパティがあります: AlarmIdentifier 、 ChildHealthChecks 、 EnableSNI 、 HealthThreshold 、 InsufficientDataHealthStatuses 、 Inverted、 および MeasureLatency 。 AWS::SQS::Queue ContentBasedDeduplication および FifoQueue プロパティを使用して、先入れ先出し	2010-05-15

変更	リリース日	説明	APIバージョン
		(FIFO) Amazon Simple Queue Service キューを作成します。 AWS::S3::Bucket Amazon S3 バケットの IPv6 ドメイン名を指定できるようになりました。	
新しいリソース	2017 年 2 月 10 日	AWS::StepFunctions::Activity AWS::StepFunctions::Activity リソースを使用して AWS Step Functions アクティビティを作成します。 AWS::StepFunctions::StateMachine AWS::StepFunctions::StateMachine リソースを使用して、Step Functions ステートマシンを作成します。	2010-05-15
新しい組み込み関数	2017 年 1 月 17 日	Fn::Split 関数を使用して、文字列を文字列値のリストに分割します。詳細については、「 Fn::Split 」を参照してください。	2010-05-15
インポートをリストするためのコンソールのサポート	2017 年 1 月 17 日	CloudFormation コンソールを使用して、エクスポートされた出力値をインポートしているすべてのスタックを表示します。詳細については、「 エクスポートされた出力値をインポートするスタックのリスト 」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2017 年 1 月 17 日	AWS::AutoScaling::AutoScalingGroup LoadBalancerNames プロパティは、Auto Scaling グループを置き換えることなく更新できます。 AWS::ECS::TaskDefinition NetworkMode プロパティおよび MemoryReservation プロパティを追加しました。 AWS::RDS::DBCluster CloudFormation は Tags プロパティの更新をサポートします。 AWS::RDS::DBInstance Timezone プロパティを追加しました。 FirehoseAction Separator プロパティを追加しました。 AWS::OpsWorks::Instance PublicIp 組み込み関数の Fn::GetAtt 属性を追加しました。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2016 年 12 月 01 日	AWS::CodeBuild::Project AWS::CodeBuild::Project リソースを使用し、CodeBuild によるソースコードのビルド方法を定義する AWS CodeBuild プロジェクトを作成します。 AWS::SSM::Association AWS::SSM::Association リソースを使用して Amazon EC2 Systems Manager ドキュメントを EC2 インスタンスと関連付けます。 AWS::EC2::SubnetCidrBlock AWS::EC2::SubnetCidrBlock リソースを使用し、単独の IPv6 CIDR ブロックを Amazon VPC サブネットと関連付けます。 AWS::EC2::VPCCidrBlock AWS::EC2::VPCCidrBlock リソースを使用して、Amazon が提供する単独の IPv6 CIDR ブロックを Amazon VPC と関連付けます。	2010-05-15

変更	リリース日	説明	API バージョン
IPv6 サポートの更新されたリソース	2016 年 12 月 01 日	AWS::EC2::Instance Ipv6AddressCount プロパティおよび Ipv6Addresses プロパティを追加しました。 AWS::EC2::NetworkAclEntry Ipv6CidrBlock プロパティを追加しました。 AWS::EC2::NetworkInterface Ipv6AddressCount プロパティおよび Ipv6Addresses プロパティを追加しました。 AWS::EC2::Route DestinationIpv6CidrBlock プロパティを追加しました。 AWS::EC2::SecurityGroupEgress CidrIpv6 プロパティを追加しました。 AWS::EC2::SecurityGroupIngress CidrIpv6 プロパティを追加しました。 AWS::EC2::SpotFleet 特定のネットワークインターフェースを起動するための Ipv6AddressCount プロパティと Ipv6Addresses プロパティが追加されました。 AWS::EC2::Subnet Ipv6CidrBlocks 関数の Fn::GetAtt 属性を追加しました。	2010-05-15

変更	リリース日	説明	API バージョン
		AWS::EC2::VPC Ipv6CidrBlocks 関数の Fn::GetAtt 属性を追加しました。 AWS::SSM::Document DocumentType プロパティを追加しました。	
リソース仕様	2016 年 11 月 22 日	CloudFormation リソース仕様を使用して、CloudFormation テンプレートの作成に役立つツールを構築します。仕様は、機械による読み取りが可能な JSON 形式のテキストファイルです。詳細については、「 CloudFormation リソース仕様 」を参照してください。	2010-05-15
新しいリソース	2016 年 11 月 22 日	AWS::OpsWorks::UserProfile AWS::OpsWorks::UserProfile リソースを使用して、OpsWorks スタックのインスタンスにアクセスする必要があるユーザー用の SSH アクセスを設定します。 AWS::OpsWorks::Volume AWS::OpsWorks::Volume リソースを使用して、OpsWorks スタックに Amazon Elastic Block Store ボリュームを登録します。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2016 年 11 月 22 日	AWS::OpsWorks::App DataSourcees プロパティを追加しました。 AWS::OpsWorks::Instance BlockDeviceMappings 、 AgentVersion 、 ElasticIps 、 Hostname、 Tenancy、 および Volumes プロパティを追加しました。 AWS::OpsWorks::Layer CustomJson プロパティおよび VolumeConfigurations プロパティを追加しました。 AWS::OpsWorks::Stack ElasticIps 、 EcsClusterArn 、 RdsDbInstances 、 CloneAppIds 、 ClonePermissions 、 および SourceStackId プロパティを追加しました。 AWS::RDS::DBInstance CopyTagsToSnapshot プロパティを追加しました。	2010-05-15
リストのインポート	2016 年 11 月 22 日	エクスポートされた出力値のインポートをリストして、どの CloudFormation スタックが値をインポートしているかを追跡します。詳細については、「エクスポートされた出力値をインポートするスタックのリスト」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
変換	2016 年 11 月 17 日	CloudFormation がサーバーレスアプリケーションの AWS SAM 構文を処理するために使用する AWS サーバーレスアプリケーションモデル (AWS SAM) を指定します。詳細については、 「変換」 を参照してください。	2010-05-15
新しいリソース	2016 年 11 月 17 日	AWS::SNS::Subscription AWS::SNS::Subscription リソースを使用して、エンドポイントを Amazon Simple Notification Service トピックにサブスクライブします。	2010-05-15
更新されたリソース	2016 年 11 月 17 日	AWS::Lambda::Function Environment プロパティを使用して、AWS Lambda 関数がアクセスできるキーと値のペア (環境変数) を指定します。 KmsKeyArn プロパティを使用して、AWS Lambda が環境変数を暗号化および復号するために使用する KMS キーを指定します。	2010-05-15
新しい CLI コマンド	2016 年 11 月 17 日	S3 バケットへのローカルアーティファクトのアップロード package コマンドを使用して、CloudFormation テンプレート内で S3 バケットを参照するローカルアーティファクトをアップロードします。 変換を伴うテンプレートの迅速なデプロイ deploy コマンドを使用して、変更セットの作成および実行アクションを単一のコマンドに結合します。このコマンドは、変換を含むスタックをすばやく作成または更新する際に便利です。	2010-05-15

変更	リリース日	説明	API バージョン
更新されたリソース	2016 年 11 月 03 日	AWS::CloudFront::Distribution DistributionConfig プロパティの場合は、HttpVersion プロパティを使用して Amazon CloudFront との通信に閲覧者が使用できる最新の HTTP バージョンを指定します。 ForwardedValues プロパティの場合は、QueryStringCacheKeys プロパティを使用してキャッシュするコンテンツを決定するために CloudFront で使用するクエリ文字列パラメータを指定します。	2010-05-15
スタックのエクスポートのリスト	2016 年 11 月 03 日	CloudFormation コンソール、API、または AWS CLI を使用して、リージョンでのすべてのエクスポートされた出力値のリストを表示します。詳細については、「 スタックの出力値のエクスポート 」を参照してください。	2010-05-15
スタックを使用した継続的な配信	2016 年 11 月 03 日	AWS CodePipeline を使用して、CloudFormation スタックを使用した継続的な配信ワークフローを構築します。詳細については、「 CodePipeline を使用した継続的デリバリー 」を参照してください。	2010-05-15
ロールバック中のリソースのスキップ	2016 年 11 月 03 日	UPDATE_ROLLBACK_FAILED 状態のスタックがある場合、ContinueUpdateRollback アクションで ResourcesToSkip パラメータを指定して、CloudFormation がロールバックできないリソースをスキップします。詳細については、「トラブルシューティング」セクションの「 更新のロールバックの失敗 」を参照してください。	2010-05-15
変更セットの機能強化	2016 年 11 月 03 日	変更セットを使用して新しいスタックを作成 できます。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2016 年 10 月 12 日	AWS::ElastiCache::CacheCluster クラスターを置き換えずに CacheNodeType プロパティを更新します。 AWS::ElastiCache::ReplicationGroup 複数のノードグループ (シャード) を含めることができ、それぞれにプライマリクラスターとリードレプリカのある Redis (有効なクラスターモード) レプリケーショングループを作成できます。 AWS::ElastiCache::SubnetGroup CacheSubnetGroupName プロパティを使用して、Amazon ElastiCache サブネットグループの名前を指定します。	2010-05-15
新しいリソース	2016 年 10 月 06 日	AWS::ApiGateway::UsagePlan AWS::ApiGateway::UsagePlan リソースを使用して、デプロイした Amazon API Gateway API の使用プランを指定します。 AWS::CodeCommit::Repository AWS::CodeCommit::Repository リソースを使用して、Amazon Web Service がホストする CodeCommit リポジトリを作成します。	2010-05-15

変更	リリース日	説明	API バージョン
更新されたリソース	2016 年 10 月 06 日	AWS::ApiGateway::Authorizer ProviderARNs プロパティを使用して、Amazon Cognito ユーザープールを Amazon API Gateway API オーソライザーとして使用します。 AWS::ApiGateway::Deployment StageName プロパティは必要なくなりました。 AWS::ElasticLoadBalancingV2::TargetGroup GetAtt 関数で、LoadBalancerArns 属性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するロードバランサーの Amazon リソースネーム (ARN) を取得します。 AWS::RDS::DBInstance Domain、および DomainIAMRoleName プロパティを使用して、ユーザーが RDS DB インスタンスに接続するときに Windows 認証を使用します。 AWS::EC2::SecurityGroupEgress DestinationPrefixListId プロパティを使用して、Amazon VPC エンドポイントの AWS サービスプレフィックスを指定します。	2010-05-15
クロススタック参照の拡張	2016 年 10 月 06 日	組み込み関数を使用してエクスポートの Name 値をカスタマイズするか、ImportValue 関数の値を参照します。	2010-05-15

変更	リリース日	説明	APIバージョン
CloudFormation サービスロール	2016 年 9 月 26 日	CloudFormation スタックオペレーションには AWS Identity and Access Management (IAM) サービスロールを使用します。CloudFormation は、ユーザーに代わってスタックリソースを呼び出すために、このロールの認証情報を使用します。詳細については、「 CloudFormation サービスロール 」を参照してください。	2010-05-15
機能を追加	2016 年 9 月 19 日	Export 出力フィールドと Fn::ImportValue 組み込み関数を使用して、別のスタックのリソースの出力を参照するスタックを作成できます。詳細については、 出力 、 Fn::ImportValue 、および「 チュートリアル: 別の CloudFormation スタックのリソース出力の参照 」を参照してください。	2010-05-15
YAML サポート	2016 年 9 月 19 日	CloudFormation テンプレートの記述に YAML 形式を使用できます。YAML では、たとえばテンプレートにコメントを追加したり、組み込み関数の短縮形を使用することもできます。詳細については、「 CloudFormation テンプレート形式 」を参照してください。	2010-05-15
新しい組み込み関数	2016 年 9 月 19 日	Fn::Sub 関数を使用して、入力文字列の変数を指定した値に置き換えます。詳細については、「 Fn::Sub 」を参照してください。	2010-05-15
新しいリソース	2016 年 9 月 19 日	AWS::KMS::Alias AWS::KMS::Alias リソースを使用して、AWS KMS key のエイリアスを作成します。	

変更	リリース日	説明	APIバージョン
更新されたリソース	2016 年 9 月 19 日	AWS::EC2::SpotFleet LaunchSpecifications プロパティでは、特定のインスタンスタイプの入札価格を指定するために SpotPrice プロパティを使用します。 AWS::ECS::Cluster ClusterName プロパティを使用して、Amazon Elastic Container Service クラスターの名前を指定します。 AWS::ECS::TaskDefinition TaskRoleArn プロパティを使用して、Amazon Elastic Container Service コンテナがユーザーに代わって AWS 呼び出しに使用する AWS Identity and Access Management ロールを指定します。 Family プロパティを使用して、特定のファミリーのタスク定義を登録します。 AWS::Elasticsearch::Domain 使用する Elasticsearch のバージョンを指定するために、ElasticsearchVersion プロパティを使用します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2016 年 8 月 11 日	複数のアベイラビリティーゾーン内で受信アプリケーショントラフィックを EC2 インスタンスなどの複数のターゲットに配信するためには、次の Elastic Load Balancing Application Load Balancer リソースを使用します: • AWS::ElasticLoadBalancingV2::Listener • AWS::ElasticLoadBalancingV2::ListenerRule • AWS::ElasticLoadBalancingV2::LoadBalancer • AWS::ElasticLoadBalancingV2::TargetGroup	2010-05-15
更新されたリソース	2016 年 8 月 11 日	AWS::AutoScaling::AutoScalingGroup 1 つまたは複数の Application Load Balancer ターゲットグループに Auto Scaling グループを関連付けるためには、TargetGroupARNs プロパティを使用します。 AWS::ECS::Service LoadBalancers プロパティをロードするためには、Amazon Elastic Container Service サービスを Application Load Balancer ターゲットグループに関連付けるために TargetGroupArn プロパティを使用します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2016 年 8 月 09 日	CloudFormation では、次のリソースが追加されました。 AWS::ApplicationAutoScaling::ScalableTarget および AWS::ApplicationAutoScaling::ScalingPolicy ターゲット リソースをいつどのようにスケールするかを定義するには、アプリケーションの Auto Scaling スケーリングポリシーを使用します。 AWS::CertificateManager::Certificate 安全な接続を有効にするには、他の AWS サービスで利用できる AWS Certificate Manager 証明書を設定します。	2010-05-15
更新されたリソース	2016 年 8 月 09 日	CloudFormation では、次のリソースが更新されました。 AWS::CloudFront::Distribution ディストリビューション構成 ViewerCertificate プロパティについては、AWS Certificate Manager 証明書を指定できます。ディストリビューション構成 Origin プロパティについては、カスタムオリジンのカスタムヘッダーと SSL プロトコルを指定できます。 AWS::EFS::FileSystem Amazon Elastic File Systemファイルシステムにパフォーマンス モードを指定できます。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
新しいリソース	2016 年 7 月 20 日	AWS IoT AWS IoT を使用して、AWS IoT ポリシー、X.509 証明書、ポリシーとプリンシパル (X.509 証明書または他の認証情報) の関連付け、AWS IoT モノ、プリンシパルとモノの間の関連付け、または AWS IoT ルールを宣言します。 • AWS::IoT::Certificate • AWS::IoT::Policy • AWS::IoT::PolicyPrincipalAttachment • AWS::IoT::Thing • AWS::IoT::ThingPrincipalAttachment • AWS::IoT::TopicRule	2010-05-15

変更	リリース日	説明	API バージョン
更新されたリソース	2016 年 7 月 20 日	CloudFormation では、次のリソースが更新されました。 AWS::IAM::Group, AWS::IAM::Role, AWS::IAM::User AWS Identity and Access Management (IAM) リソースのカスタム名を指定するには名前のプロパティを使用します。 AWS::ApiGateway::Method Integration プロパティでは、PassthroughBehavior プロパティを使用していつ Amazon API Gateway がリクエストを対象バックエンドに渡すかを指定できます。 AWS::ApiGateway::Model および AWS::ApiGateway::RestApi Schema と Body プロパティの JSON オブジェクトを指定できます。	2010-05-15
Auto Scaling グループの UpdatePolicy	2016 年 6 月 9 日	UpdatePolicy 属性では、AutoScalingReplacingUpdate プロパティを使用して、Auto Scaling グループを更新する場合に Auto Scaling グループおよびそこに含まれるインスタンスを置き換えるかどうかを指定します。置き換え中は、更新が失敗した場合に CloudFormation が元の Auto Scaling グループにロールバックできるように、新しい Auto Scaling グループが正常に作成されるまで古いグループは CloudFormation によって保持されます。詳細については、「UpdatePolicy 属性」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2016 年 6 月 9 日	CloudFormation では、次のリソースが追加されました。 AWS::EC2::FlowLog 指定されたネットワークインターフェース、サブネット、または VPC の IP トラフィックをキャプチャする Amazon Elastic Compute Cloud フローログを作成します。 AWS::KinesisFirehose::DeliveryStream Amazon Simple Storage Service、Amazon Redshift、または Amazon OpenSearch Service などの送信先にリアルタイムでストリーミングデータを送信する配信ストリームを作成します。	2010-05-15
更新されたリソース	2016 年 6 月 9 日	CloudFormation では、次のリソースが更新されました。 AWS::Kinesis::Stream Name プロパティを使用して、Amazon Kinesis ストリームの名前を指定します。 AWS::Lambda::Function Code プロパティでは、ZipFile ランタイム環境で nodejs4.3 プロパティおよび cfn 応答モジュールを使用できます。 AWS::SNS::Topic CloudFormation で Amazon Simple Notification Service トピックリソースの更新が有効になりました。	2010-05-15

変更	リリース日	説明	API バージョン
新しいリソース	2016 年 4 月 25 日	AWS::EC2::Host リソースを使用して、EC2 インスタンスの起動用に完全に専用の物理サーバーを割り当てます。	2010-05-15
更新されたリソース	2016 年 4 月 25 日	AWS::EC2::Instance Affinity および HostId の各プロパティを使用して、Amazon Elastic Compute Cloud の専用ホストにインスタンスを起動します。 AWS::ECS::Service DeploymentConfiguration プロパティを使用して、デプロイメント中に実行できるタスクの数を設定します。 AWS::ECS::TaskDefinition CloudFormation に、追加の Amazon Elastic Container Service のコンテナ定義プロパティのサポートが追加されました。 AWS::GameLift::Fleet MaxSize および MinSize の各プロパティを使用して、Amazon GameLift Servers のフリートで許可される EC2 インスタンスの最大数と最小数を指定します。 AWS::Lambda::Function FunctionName プロパティを使用して、AWS Lambda 関数の名前を指定します。Python 2.7 を使用してインライン関数を指定することもできます。	2010-05-15

変更	リリース日	説明	API バージョン
新しいリソース	2016 年 4 月 18 日	Amazon API Gateway Amazon API Gateway リソースを使用して、規模にかかわらず API を発行、保守、監視します。EC2 インスタンスを実行するアプリケーションや AWS Lambda で実行されるコードなどのバックエンドサービスにアクセスするためにクライアントが呼び出す API を作成できます。 • AWS::ApiGateway::Account • AWS::ApiGateway::ApiKey • AWS::ApiGateway::Authorizer • AWS::ApiGateway::BasePathMapping • AWS::ApiGateway::ClientCertificate • AWS::ApiGateway::Deployment • AWS::ApiGateway::Method • AWS::ApiGateway::Model • AWS::ApiGateway::Resource • AWS::ApiGateway::RestApi • AWS::ApiGateway::Stage AWS::Events::Rule アカウントの AWS リソースの変化 (イベント) を監視する Amazon CloudWatch Events ルールを作成できます。受信イベントがルールで示された条件に一致する場合、Amazon CloudWatch Events は AWS Lambda 関数や Amazon Simple Notification Service トピックなどの指定されたターゲットにメッセージを送信してアクティベートします。	2010-05-15

変更	リリース日	説明	API バージョン
		AWS::WAF::SizeConstraintSet および AWS::WAF::XssMatchSet 2 つの AWS WAF ルールを使用して、ウェブリクエストのサイズを確認したり、クロスサイトスクリプト攻撃を防いだりできます。	
新しいリソース	2016 年 3 月 31 日	AWS::Lambda::Alias リソースを使用して AWS Lambda 関数のエイリアスと AWS::Lambda::Version リソースを作成し、関数のバージョンを作成します。	2010-05-15
更新されたリソース	2016 年 3 月 31 日	CloudFormation では、次のリソースが更新されました。 AWS::EMR::Cluster および AWS::EMR::InstanceGroupConfig EbsConfiguration プロパティを使用して、Amazon EMR クラスターの Amazon Elastic Block Store ストレージボリュームまたはインスタンスグループを設定します。 AWS::Lambda::Function VpcConfig プロパティを使用して、AWS Lambda 関数を有効化し VPC のリソースにアクセスします。 AWS::S3::Bucket Amazon Simple Storage Service ライフサイクルルールで、オブジェクトを指定のストレージクラスに移行する際に、複数の移行ルールを指定できます。	2010-05-15
変更セット	2016 年 3 月 29 日	スタックを更新する前に、変更セットを使用して、変更が実行中のリソースに与える可能性のある影響を確認できます。詳細については、「 変更セットを使用したスタックの更新 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
新しいリソース	2016 年 3 月 15 日	AWS::GameLift::Alias 、 AWS::GameLift::Build 、 AWS::GameLift::Fleet リソースを使用して AWS にマルチプレイヤーゲームサーバーをデプロイします。	2010-05-15
新しいリソース	2016 年 2 月 26 日	CloudFormation では、次のリソースが追加されました。 AWS::ECR::Repository ユーザーが Docker イメージをプル/プッシュできる Amazon Elastic Container Registry リポジトリを作成します。 AWS::EC2::NatGateway ネットワークアドレス変換 (NAT) ゲートウェイを使用して、プライベートサブネットの EC2 インスタンスからインターネットに接続できるようにします。 AWS::Elasticsearch::Domain レガシー Elasticsearch OSS クラスターを実行する Amazon OpenSearch Service のドメインを作成します。 AWS::EMR::Cluster、AWS::EMR::InstanceGroupConfig、AWS::EMR::Step Amazon EMR リソースを使用して大量のデータの分析と処理を支援します。クラスターを作成し、そのクラスターでジョブを実行できます。	2010-05-15

変更	リリース日	説明	APIバージョン
更新されたリソース	2016 年 2 月 26 日	CloudFormation では、次のリソースが更新されました。 AWS::CloudTrail::Trail IsMultiRegionTrail プロパティを使用して、スタックを作成する特定のリージョンまたはすべてのリージョンで AWS CloudTrail 証跡を作成するかどうかを指定します。 AWS::Config::ConfigurationRecorder 記録グループの場合、IncludeGlobalResourceTypes プロパティを使用してすべてのグローバルリソースタイプを記録します。 AWS::RDS::DBCluster KmsKeyId プロパティと StorageEncrypted プロパティを使用してクラスターのデータベースインスタンスを暗号化します。	2010-05-15
リソースの保持	2016 年 2 月 26 日	DELETE_FAILED 状態のスタックについては、RetainResources パラメータを使用して CloudFormation で削除できないリソースを保持します。詳細については、「スタックの削除の失敗」を参照してください。	2010-05-15
スタックタグの更新	2016 年 2 月 26 日	スタックを更新する際にスタックタグを追加、変更、または削除できます。詳細については、「CloudFormation スタックの更新」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
失敗した更新ロールバックのロールバックを続ける	2016 年 1 月 25 日	UPDATE_ROLLBACK_FAILED 状態のスタックの場合、更新のロールバックを続けて動作状態のスタックを取得することができます。そのようにして、スタックを元の設定に戻し、もう一度更新を試行することができます。詳細については、「 更新のロールバックを続ける 」を参照してください。	2010-05-15
アジアパシフィック (ソウル) リージョンで使用可能な新しいサンプルテンプレート。	2016 年 1 月 7 日	次の CloudFormation サンプルテンプレートのコレクションは、ap-northeast-2 リージョン用です。 - サンプルソリューション - アプリケーションフレームワーク - サービス 詳細については、「 Working with CloudFormation templates 」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
新しいリソース	2015 年 12 月 28 日	CloudFormation では、次のリソースが追加されました。 AWS::DirectoryService::MicrosoftAD Microsoft Active Directory リソースを使用して、AWS に Microsoft Active Directory デイレクトリを作成します。 AWS::Logs::Destination および AWS::Logs::LogStream Amazon CloudWatch Logs リソースを使用して、それぞれログデータをリアルタイムで処理するための宛先とログストリームを作成するための宛先を作成します。 AWS::WAF::ByteMatchSet, AWS::WAF::IPSet, AWS::WAF::Rule, AWS::WAF::SqlInjectionMatchSet, および AWS::WAF::WebACL AWS WAF リソースを使用して、コンテンツへのウェブリクエストを制御およびモニタリングします。	2010-05-15

変更	リリース日	説明	APIバージョン
リソースの更新	2015 年 12 月 28 日	CloudFormation では、次のリソースが更新されました。 AWS::CloudFront::Distribution ディストリビューション構成の場合、WebACLId プロパティを使用して AWS WAF ウェブアクセスコントロールリスト (ACL) と Amazon CloudFront ディストリビューションを関連付けます。キャッシュ動作とデフォルトのキャッシュ動作の場合、デフォルト値と最大有効期限 (TTL) 値を指定できます。 AWS::DynamoDB::Table Amazon DynamoDB テーブルを置き換えなくても、グローバルセカンダリインデックスを作成、更新、または削除できます。 AWS::S3::Bucket ReplicationConfiguration プロパティを使用して、レプリケーションするオブジェクトとそれらの保存場所を指定します。 NotificationConfiguration プロパティ内のプロパティを使用してフィルタを指定し、指定したオブジェクトに Amazon Simple Storage Service が通知を送信するようにします。	2010-05-15
パラメータのグループ化とソート	2015 年 12 月 3 日	ユーザーがテンプレートを使用してスタックを作成または更新したときに、AWS::CloudFormation::Interface メタデータキーを使用して、CloudFormation コンソールでパラメータのグループ化とソートを行います。	2010-05-15

変更	リリース日	説明	APIバージョン
更新ポリシー属性	2015 年 12 月 3 日	Auto Scaling 更新ポリシー属性 の場合、MinSuccessfulInstancesPercent プロパティを使用して、更新が成功した場合に成功のシグナルを送信する必要があるインスタンスの割合を指定します。	2010-05-15
新しいリソース	2015 年 12 月 3 日	CloudFormation では、次のリソースが追加されました。 AWS::CodePipeline::Pipeline および AWS::CodePipeline::CustomActionType CodePipeline リソースを使用して、リリースプロセスでソフトウェアの変更がどのように行われるかを記述するパイプラインを作成します。 AWS::Config::ConfigurationRecorder、AWS::Config::DeliveryChannel、および AWS::Config::ConfigRule AWS Config リソースを使用して、特定の AWS リソースの構成変更をモニタリングします。 AWS::KMS::Key AWS Key Management Service (AWS KMS) リソースを使用して、ユーザーが少量のデータの暗号化に使用できるカスターマネージドキーを AWS KMS で作成します。 AWS::SSM::Document Amazon EC2 Systems Manager を使用して、インスタンス上構成を指定するドキュメントを作成します。	2010-05-15

変更	リリース日	説明	APIバージョン
リソースの更新	2015 年 12 月 3 日	CloudFormation では、次のリソースが更新されました。 AWS::AutoScaling::LaunchConfiguration EBS ボリュームが暗号化されているかどうかを指定します。 AWS::AutoScaling::ScalingPolicy 2 つのポリシータイプ (シンプルおよびステップスケーリング) を使用して、Amazon CloudWatch (CloudWatch) アラームを超過したときの Auto Scaling グループのスケーリング方法を指定できます。 AWS::CloudTrail::Trail CloudWatch プロパティを使用して、CloudWatch ロググループにログを送信します。証跡にタグを追加し、ログの暗号化に使用する AWS KMS key を指定できます。 AWS::CodeDeploy::Application、AWS::CodeDeploy::DeploymentConfig、および AWS::CodeDeploy::DeploymentGroup ApplicationName 、DeploymentConfigName 、DeploymentGroupName の各プロパティを使用して、CodeDeploy リソースのカスタム名を指定します。 AWS::DynamoDB::Table StreamSpecification プロパティを使用して、Amazon DynamoDB (DynamoDB) テーブルに保存された項目の変更をキャプチャする設定を指定します。	2010-05-15

変更	リリース日	説明	APIバージョン
		AWS::EC2::Instance SsmAssociations プロパティを使用して、Amazon EC2 Systems Manager ドキュメントとインスタンスを関連付けます。 AWS::EC2::SpotFleet AllocationStrategy プロパティを使用して、スポットプール間でターゲット容量を割り当てる方法を指定します。ExcessCapacityTerminationPolicy プロパティを使用して、ターゲット容量がスポットフリートのサイズを下回った場合にインスタンスを削除する方法を指定します。 AWS::Redshift::Cluster KmsKeyId プロパティを使用して、Amazon Redshift クラスター内のデータを暗号化するための AWS KMS key を指定します。 AWS::WorkSpaces::Workspace 暗号化プロパティを使用して、ボリュームに保存されたデータを暗号化します。	
リソースの更新	2015 年 11 月 4 日	AWS::EC2::Volume リソースの場合、ボリュームのデータに整合性がなくなった場合に AutoEnableIO プロパティを使用して I/O オペレーションを自動的に再開します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2015 年 10 月 1 日	CloudFormation では、次のリソースが追加されました。 AWS::CodeDeploy::Application、AWS::CodeDeploy::DeploymentGroup、および AWS::CodeDeploy::DeploymentConfig CodeDeploy を使用してデプロイ環境を作成し、EC2 またはオンプレミスのインスタンスに適用します。 AWS::DirectoryService::SimpleAD Simple Active Directory を使用して Directory Service Simple AD を作成します。これは、Microsoft Active Directory 互換のディレクトリです。 AWS::EC2::PlacementGroup プレイスメントグループを使用して、レイテンシーの小さいネットワークでインスタンスのクラスターを作成します。 AWS::EC2::SpotFleet スポットフリートを使用して、中断可能なタスクを実行するスポットインスタンス群を起動します。 AWS::Lambda::EventSourceMapping イベントソースマッピングリソースを使用して、AWS Lambda (Lambda) 関数のイベントのソースとしてストリームを指定します。 AWS::Lambda::Permission Lambda アクセス許可を使用して、Lambda 関数のポリシーにステートメントを追加します。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
		AWS::Logs::SubscriptionFilter サブスクリプションフィルタを使用して、Kinesis ストリームに配信されるログイベントを定義します。 AWS::RDS::DBCluster および AWS::RDS::DBClusterParameterGroup クラスターとクラスターパラメータグループのリソースを使用して、Amazon Aurora DB クラスターを作成します。 AWS::WorkSpaces::Workspace WorkSpaces を使用してクラウドベースのデスクトップ体験を作成します。	

変更	リリース日	説明	API バージョン
リソースの更新	2015 年 10 月 1 日	CloudFormation では、次のリソースが更新されました。 AWS::ElastiCache::ReplicationGroup Fn::GetAtt 組み込み関数を使用して、読み取り専用レプリカのアドレスとポートの一覧を取得します。 AWS::OpsWorks::Stack AgentVersion プロパティを使用して、特定の OpsWorks エージェントを指定します。 AWS::OpsWorks::App Environment プロパティを使用して、OpsWorks アプリケーション用の環境変数を指定します。 AWS::S3::Bucket NotificationConfiguration プロパティについては、Lambda 関数と Amazon Simple Queue Service (Amazon SQS) キューの通知構成を設定できます。	2010-05-15
IAM 条件キー	2015 年 10 月 1 日	AWS Identity and Access Management (IAM) ポリシーでは、CloudFormation 固有の条件キーを使用して、IAM ポリシーをいつ有効にするかを指定します。詳細については、AWS Identity and Access Management によるアクセスの制御をご参照ください。	2010-05-15
CloudFormation デザイナー	2015 年 10 月 1 日	CloudFormation デザイナーを使用して、ドラッグアンドドロップインターフェースでテンプレートの作成と変更を行います。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2015 年 8 月 24 日	AWS::EC2::VPCEndpoint リソースを使用して VPC と別の AWS サービスとの間でプライベート接続を確立します。	2010-05-15
リソースの更新	2015 年 8 月 24 日	CloudFormation では、次のリソースが更新されました。 AWS::ElasticBeanstalk::Environment Tags プロパティを使用して、AWS Elastic Beanstalk (Elastic Beanstalk) 環境のタグ (キーと値のペア) を指定します。 AWS::Lambda::Function Code プロパティの場合、ZipFile プロパティを使用して Lambda 関数のソースコードをテンプレートで直接記述します。現在、ZipFile プロパティを使用できるのは nodejs ランタイム環境の場合のみです。ただし、java8、nodejs など、すべてのランタイム環境で S3 バケット内のファイルを指すことができます。 AWS::OpsWorks::Instance EbsOptimized プロパティを使用して、インスタンスが Amazon Elastic Block Store (Amazon EBS) I/O 用に最適化されているかどうかを示します。 AWS::RDS::DBInstance SourceDBInstanceIdentifier プロパティでは、別のリージョンにあるデータベースインスタンスを指定して、クロスリージョンリードレプリカを作成できます。	2010-05-15

変更	リリース日	説明	APIバージョン
Amazon S3 テンプレートの URL	2015 年 8 月 24 日	バージョニングが有効なバケットの場合は、スタックの作成時または更新時に、Amazon S3 テンプレートの URL にバージョン ID を指定できます (例: https://s3.amazonaws.com/templates/myTemplate.template?versionId=123ab1cdeKd0W5IH4GAcYbEngcpTJTdW)。	2010-05-15
新しいリソース	2015 年 8 月 3 日	AWS::EFS::FileSystem リソースを使用して Amazon Elastic File System (Amazon EFS) ファイル システムを作成し、 AWS::EFS::MountTarget リソースを使用してファイル システムのマウントポイントを作成します。	2010-05-15
アクセス権限の条件の変更	2015 年 6 月 11 日	AWS::RDS::DBInstance リソースを作成またはアップデートする場合には、 <code>ec2:DescribeAccountAttributes</code> アクションを実行できる権限が必要になりました。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいリソース	2015 年 6 月 11 日	CloudFormation では、次のリソースが追加されました。 AWS::DataPipeline::Pipeline データパイプラインを使用して、データの移動と変換を自動化します。 Amazon Elastic Container Service リソース AWS::ECS::Service、AWS::ECS::Cluster、AWS::ECS::TaskDefinition の各リソースを使用して、EC2 インスタンスのクラスターで Docker コンテナを作成します。 AWS::ElastiCache::ReplicationGroup レプリケーショングループを使用して、1 個の読み書き可能なプライマリクラスターと、最大 5 個の読み取り専用セカンダリクラスターで構成されるノードの集合を作成します。 AWS::IAM::ManagedPolicy 管理ポリシーを使用し、IAM ユーザー、グループ、ロールに権限を適用するために使用できる AWS アカウントのポリシーを作成します。 AWS::Lambda::Function イベントに応答してコードを実行する Lambda 機能を使用します。 AWS::RDS::OptionGroup オプショングループは、Amazon Relational Database Service (Amazon RDS) データベースの作成と管理に役立ちます。	2010-05-15

変更	リリース日	説明	API バージョン
リソースの更新	2015 年 6 月 11 日	CloudFormation では、次のリソースが更新されました。 AWS::EC2::Subnet MapPublicIpOnLaunch プロパティを使用して、サブネット内のインスタンスにパブリック IP アドレスを自動的に割り当てます。 AWS::ElastiCache::CacheCluster SnapshotName プロパティを使用して、新しい Redis キャッシュクラスターにスナップショットデータを復元します。 AWS::IAM::User LoginProfile プロパティには、PasswordResetRequired プロパティを使用して AWS マネジメントコンソール へのログイン時にユーザーに新しいパスワードの設定を求めます。 AWS::OpsWorks::Layer LifecycleEventConfiguration プロパティを使用して、OpsWorks Layer のライフサイクルイベントを設定します。 AWS::S3::Bucket LifecycleConfiguration プロパティには、NoncurrentVersionExpirationInDays プロパティと NoncurrentVersionTransition プロパティを使用して以前のオブジェクトバージョンのライフサイクルルールを指定します。	2010-05-15

変更	リリース日	説明	APIバージョン
新しいパラメータタイプ	2015 年 5 月 19 日	CloudFormation コンソールを使用してスタックを作成または更新するときにはいつでも、ID、名前、または Name タグ値によって AWS 固有のパラメータタイプ値を検索できます。 CloudFormation には、次の AWS 固有のパラメータタイプのサポートも追加されました。詳細については、「パラメータ」を参照してください。 • AWS::EC2::AvailabilityZone::Name • List<AWS::EC2::AvailabilityZone ::Name> • AWS::EC2::Instance::Id • List<AWS::EC2::Instance ::Id> • AWS::EC2::Image::Id • List<AWS::EC2::Image::Id > • AWS::EC2::SecurityGroup::GroupName • List<AWS::EC2::SecurityGroup::Gr oupName > • AWS::EC2::Volume::Id • List<AWS::EC2::Volume::Id > • AWS::Route53::HostedZone::Id • List<AWS::Route53::HostedZone ::Id>	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
新しいリソース	2015 年 4 月 16 日	CloudFormation では、次のリソースが追加されました。 AWS::AutoScaling::LifecycleHook Auto Scaling のライフサイクルフックを使用して、インスタンスを起動または終了した後のインスタンスの状態を管理します。 AWS::RDS::EventSubscription イベントサブスクリプションを使用して、Amazon RDS イベントに関する通知を受け取ります。	2010-05-15

変更	リリース日	説明	APIバージョン
リソースの更新	2015 年 4 月 16 日	CloudFormation では、次のリソースが更新されました。 AWS::AutoScaling::AutoScalingGroup NotificationConfigurations プロパティを使用して、複数の通知を指定します。 AWS::AutoScaling::LaunchConfiguration PlacementTenancy プロパティを使用して、インスタンスのテナンシーを指定します。 ClassicLinkVPCId プロパティと ClassicLinkVPCSecurityGroups プロパティを使用して、EC2-Classic インスタンスを ClassicLink が有効な VPC にリンクします。 AWS::AutoScaling::ScalingPolicy MinAdjustmentStep プロパティを使用して、スケールイベント中に追加または削除されるインスタンスの最小数を指定します。 AWS::CloudFront::Distribution ビューア証明書の場合、MinimumProtocolVersion プロパティを使用して最小バージョンのプロトコルを指定します。キャッシュ動作の場合、CachedMethods プロパティを使用して Amazon CloudFront (CloudFront) キャッシュが応答するメソッドを指定します。オリジンの場合、OriginPath を使用して CloudFront がコンテンツの要求に使用するパスを指定します。	2010-05-15

変更	リリース日	説明	APIバージョン
		AWS::ElastiCache::CacheCluster Memcached キャッシュクラスターの場合、AZMode プロパティと PreferredAvailabilityZones プロパティを使用して複数のアベイラビリティーゾーン (AZ) のノードを指定します。 AWS::EC2::Volume KmsKeyId プロパティを使用して、暗号化されたボリュームのカスタマーマネージドキーを指定します。 AWS::OpsWorks::Instance TimeBasedAutoScaling プロパティを使用して、指定したスケジュールに基づいて自動的にインスタンスをスケールします。 AWS::OpsWorks::Layer LoadBasedAutoScaling プロパティを使用して、負荷ベースのスケールリングポリシーを指定します。ボリューム構成の場合、VolumeType プロパティと Iops プロパティを使用してそれぞれボリュームの種類と 1 秒あたりの数値の I/O オペレーションを指定します。 AWS::RDS::DBInstance CharacterSetName プロパティを使用して、サポートされるデータベースエンジンの文字セットを指定します。 StorageEncrypted プロパティを使用してデータベースインスタンスが暗号化されるかどうかを示し、KmsKeyId を使用して暗号化されたデータベース	

変更	リリース日	説明	API バー ジョン
		インスタンスのカスタマーマネージドキーを指定します。 AWS::Route53::HealthCheck HealthCheckTags プロパティを使用して、タグをヘルスチェックと関連付けます。 AWS::Route53::HostedZone VPCs プロパティを使用して、プライベートホストゾーンを作成します。 HostedZoneTags プロパティを使用して、タグをホストゾーンと関連付けます。	
新しいテンプレートセクション	2015 年 4 月 16 日	テンプレートに Metadata セクションを追加して、設計や実装の詳細など、テンプレートを記述する任意の JSON オブジェクトを含めます。	2010-05-15
リソースの更新	2015 年 4 月 8 日	AWS::CloudFormation::CustomResource リソースには、ServiceToken プロパティに Lambda 関数の Amazon リソース ネーム (ARN) を指定できます。	2010-05-15
Amazon RDS の更新	2014 年 12 月 24 日	CloudFormation は RDS DB インスタンスに 2 つの新しいプロパティを追加します。オプショングループを DB インスタンスと関連付け、DB インスタンスのストレージタイプを指定できます。詳細については、「 AWS::RDS::DBInstance 」を参照してください。	2010-05-15
Elastic Load Balancing の更新	2014 年 12 月 24 日	ConnectionSettings プロパティを使用して、接続のアイドル状態を保持する期間を指定できます。詳細については、「 AWS::ElasticLoadBalancing::LoadBalancer 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
Route 53 の更新	2014 年 11 月 6 日	Route 53 [hosted zones] (ホストゾーン)、 [health checks] (ヘルスチェック)、 [failover record sets] (フェイルオーバーレコードセット)、および [geolocation record sets] (位置情報レコードセット) をプロビジョニングして管理できるようになりました。	2010-05-15
Auto Scaling ロールリング更新の拡張	2014 年 11 月 6 日	更新中に、WaitOnResourceSignals フラグを使用して CloudFormation がインスタンスを待機して成功シグナルを送信をするよう指示できます。こうすると、CloudFormation は現在のバッチの準備が整うまでインスタンスの次のバッチを更新しません。詳細については、「 UpdatePolicy 属性 」を参照してください。	2010-05-15
新しい VPC Fn::GetAtt 属性	2014 年 11 月 6 日	VPC ID がある場合、その VPC のデフォルトのセキュリティグループとネットワーク ACL を取得できます。詳細については、「 Fn::GetAtt 」を参照してください。	2010-05-15
AWS 固有のパラメータタイプ	2014 年 11 月 6 日	CloudFormation テンプレートで AWS 固有パラメータタイプを指定できます。CloudFormation コンソールでは、これらのパラメータタイプに対する有効値のドロップダウンリストが表示されます。API または AWS CLI を使用すると、CloudFormation はスタックを作成または更新する前にすばやくこれらのパラメータタイプの値を検証できます。詳細については、「 パラメータ 」を参照してください。	2010-05-15
CreationPolicy 属性	2014 年 11 月 6 日	CreationPolicy 属性を使用して、スタックの作成に進む前に、アプリケーションが EC2 インスタンスで準備ができるまで待つように CloudFormation に指示できます。待機条件および待機条件ハンドルの代わりに、作成ポリシーを使用できます。詳細については、 CreationPolicy 属性 を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
Amazon CloudFront で転送された値	2014 年 9 月 29 日	キャッシュ動作については、オリジンにヘッダーを転送できます。See ForwardedValues 。	2010-05-15
AWS OpsWorks の更新	2014 年 9 月 29 日	Chef 11.10 では、ChefConfiguration プロパティを使用して Berkshelf を有効にできます。または AWS OpsWorks スタックで AWS OpsWorks 組み込みのセキュリティグループを使用できます。詳細については、「 AWS::OpsWorks::Stack 」を参照してください。	2010-05-15
Elastic Load Balancing のタグ指定のサポート	2014 年 9 月 29 日	AWS CloudFormation は Elastic Load Balancing ロードバランサーにスタックレベルのタグをつけます。また、ロードバランサーに独自のタグを追加できます。「 AWS::ElasticLoadBalancing::LoadBalancer 」を参照してください。	2010-05-15
Amazon Simple Notification Service トピックのポリシー更新	2014 年 9 月 29 日	これで、Amazon SNS トピックポリシーを更新できます。詳細については、「 AWS::SNS::TopicPolicy 」を参照してください。	2010-05-15
RDS DB インスタンスの更新	2014 年 9 月 5 日	AWS::RDS::DBInstance リソースの PubliclyAccessible プロパティを使用して、DB インスタンスをインターネットに接続するかどうかを指定できます。	2010-05-15
UpdatePolicy 属性の更新	2014 年 9 月 5 日	スケジュールされたアクションが関連付けられている Auto Scaling グループに更新ポリシーを指定できます。詳細については、「 UpdatePolicy 属性 」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
Amazon CloudWatch サポート	2014 年 7 月 10 日	CloudFormation を使用して Amazon CloudWatch Logs (CloudWatch Logs) ロググループおよびメトリック スフィルタをプロビジョニングし、管理できます。詳細については、「 AWS::Logs::LogGroup 」または「 AWS::Logs::MetricFilter 」を参照してください。	2010-05-15
Amazon CloudFront ディストリビューション 構成の更新	2014 年 6 月 17 日	追加の CloudFront ディストリビューション構成プロパティを指定できます: - カスタムエラーレスポンスは、4xx および 5xx HTTP ステータスコードのカスタムエラーメッセージを定義します。 - 価格クラスは、CloudFront サービスに支払う上限価格を定義します。 - 制限により、コンテンツを参照できるユーザーを定義します。 - ビューア証明書は、ビューアが HTTPS を使用するとき使用する証明書を指定します。 - キャッシュ動作については、許可される HTTP メソッドを指定し、Cookie を転送するかどうかを示すことができます。 詳細については、「AWS::CloudFront::Distribution」を参照してください。	2010-05-15
EC2 インスタンスの更新	2014 年 6 月 17 日	インスタンスのオペレーティングシステムのシャットダウンコマンドを呼び出すときにインスタンスが停止するか終了するかを指定できます。詳細については、「 AWS::EC2::Instance 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
EBS ボリュームの更新	2014 年 6 月 17 日	サポートされているインスタンスタイプの暗号化されている EBS ボリュームを使用できます。詳細については、「 AWS::EC2::Volume 」を参照してください。	2010-05-15
新しい Amazon VPC ピア接続	2014 年 6 月 17 日	CloudFormation を使用して Amazon Virtual Private Cloud (Amazon VPC) ピア接続を作成できます。この接続は、2 つの VPC 間のネットワーク接続を確立します。詳細については、「 AWS::EC2::VPCPeeringConnection 」を参照してください。	2010-05-15
Amazon EC2 Auto Scaling グループの更新	2014 年 6 月 17 日	Amazon EC2 Auto Scaling グループのインスタンスを起動する、既存のクラスタープレイスメントグループを指定できます。詳細については、「 AWS::AutoScaling::AutoScalingGroup 」を参照してください。	2010-05-15
AWS CloudTrail のサポート	2014 年 6 月 17 日	CloudFormation は AWS CloudTrail をサポートします。AWS アカウントから行われた API 呼び出しをキャプチャし、指定された場所にログを発行できます。詳細については、「 AWS::CloudTrail::Trail 」を参照してください。	2010-05-15
スタック更新の強化	2014 年 5 月 12 日	CloudFormation は、スタックを更新するための追加機能をサポートします。 - スタックのテンプレートを再送信しないで、CloudFormation スタックパラメータを更新できます。 - CloudFormation スタックの Amazon SNS 通知トピックを追加または削除できます。 詳細については、「CloudFormation スタックの更新」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
Amazon Kinesis サポート	2014 年 5 月 6 日	CloudFormation を使用して、データソースのデータレコードをキャプチャおよびトランスポートする Amazon Kinesis ストリームを作成できます。詳細については、「 AWS::Kinesis::Stream 」を参照してください。	2010-05-15
新しい S3 バケットのプロパティ	2014 年 5 月 5 日	CloudFormation は追加の S3 バケットプロパティをサポートします。 • Cross-Origin Resource Sharing (CORS) は、バケット内のオブジェクトの Cross-Origin Resource Sharing を定義します。 • ライフサイクルは、Amazon S3 がオブジェクトをその存続期間にわたってどのように管理するかを定義します。 • アクセスログ作成ポリシーは、バケットに対するリクエストについての情報をキャプチャします。 • 通知は、どのイベントをレポートし、どの Amazon SNS トピックにメッセージを送信するかを定義します。 • バージョニングは、バケット内のすべてのオブジェクトの複数のバリエーションを有効にします。 • リダイレクトとルーティングルールは、バケットのウェブサイトエンドポイントに対して作成されたリクエストのリダイレクト動作を管理します。 詳細については、「AWS::S3::Bucket」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジョン
Amazon EC2 Auto Scaling サポート	2014 年 5 月 5 日	CloudFormation は、Auto Scaling グループに対するメトリクス収集をサポートします。詳細については、「 AWS::AutoScaling::AutoScalingGroup 」を参照してください。	2010-05-15
Fn::If の更新	2014 年 5 月 5 日	テンプレートの出力セクションで Fn::If 組み込み関数を使用できます。詳細については、「 条件関数 」を参照してください。	2010-05-15
AWS CloudTrail による API ログ作成	2014 年 4 月 2 日	AWS CloudTrail (CloudTrail) を使用して、CloudFormation リクエストのログを作成できます。CloudTrail で、自分のアカウントに対する CloudFormation API 呼び出しの履歴を取得できます。詳細については、「 AWS CloudTrail による CloudFormation API コールのログ記録 」を参照してください。	2010-05-15
Elastic Load Balancing の更新	2014 年 3 月 20 日	ロードバランサーへのリクエストについての情報をキャプチャするためのアクセスログ作成ポリシーを指定できます。インスタンスが登録解除されているか正常でなくなっているときに実行中のリクエストの処理方法を指定する Connection Draining ポリシーも指定できます。詳細については、「 AWS::ElasticLoadBalancing::LoadBalancer 」を参照してください。	2010-05-15
OpsWorks のサポート	2014 年 3 月 3 日	CloudFormation を使用して、OpsWorks スタックをプロビジョニングおよび管理できます。詳細については、 AWS::OpsWorks::Stack または AWS OpsWorks のテンプレートスニペット を参照してください。	2010-05-15
Amazon S3 テンプレート サイズ制限の引き上げ	2014 年 2 月 18 日	Amazon S3では、テンプレートのサイズを最大 460,800 バイトまで指定できます。	2010-05-15

変更	リリース日	説明	APIバージョン
Amazon Redshift サポート	2014 年 2 月 10 日	CloudFormation を使用して、Amazon Redshift クラスターをプロビジョニングおよび管理できます。詳細については、 Amazon Redshift テンプレート スニペット または AWS::Redshift::Cluster を参照してください。	2010-05-15
S3 バケットとバケットポリシーの更新	2014 年 2 月 10 日	S3 バケットとバケットポリシーリソースのいくつかのプロパティを更新できます。詳細については「 AWS::S3::Bucket 」または「 AWS::S3::BucketPolicy 」を参照してください。	2010-05-15
Elastic Beanstalk 環境およびアプリケーションバージョンの更新	2014 年 2 月 10 日	Elastic Beanstalk 環境構成およびアプリケーションバージョンを更新できます。詳細については、「 AWS::ElasticBeanstalk::Environment 」、「 AWS::ElasticBeanstalk::ConfigurationTemplate 」、または「 AWS::ElasticBeanstalk::ApplicationVersion 」を参照してください。	2010-05-15
Amazon SQS の更新	2014 年 1 月 29 日	Amazon SQS キューに対してデッドレターキューを指定できます。詳細については、「 AWS::SQS::Queue 」を参照してください。	2010-05-15
Auto Scaling のスケジュールされたアクション	2014 年 1 月 27 日	スケジュールに基づいて、Auto Scaling グループ内の EC2 インスタンスの数を拡張できます。スケジュールを使用することで、予測可能な負荷の変化に応じてアプリケーションを拡張できます。詳細については、「 AWS::AutoScaling::ScheduledAction 」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
DynamoDB セカンダリインデックス	2014 年 1 月 27 日	DynamoDB データベースのローカルおよびグローバルのセカンダリインデックスを作成できます。セカンダリインデックスを使用することで、プライマリキー以外の属性を持つデータに効率的にアクセスできます。詳細については、「 AWS::DynamoDB::Table 」を参照してください。	2010-05-15
Auto Scaling の更新	2014 年 1 月 2 日	Auto Scaling グループのインスタンス ID を指定したり、構成を起動したりすることができます。追加の Auto Scaling ブロックデバイスプロパティを指定することもできます。詳細については「 AWS::AutoScaling::AutoScalingGroup 」または「 AWS::AutoScaling::LaunchConfiguration 」を参照してください。	2010-05-15
Amazon SQS の更新	2014 年 1 月 2 日	SQS キューを更新し、追加のプロパティを指定できます。詳細については、「 AWS::SQS::Queue 」を参照してください。	2010-05-15
制限の緩和	2014 年 1 月 2 日	CloudFormation テンプレートでは、最大で 60 個のパラメータおよび 60 個の出力を指定できます	2010-05-15
新しいコンソール	2013 年 12 月 19 日	新しい CloudFormation コンソールでは、スタックイベントの自動更新やスタックパラメータのアルファベット順の並べ替えなどの機能が追加されています。	2010-05-15
クロスゾーンロードバランサー	2013 年 12 月 19 日	クロスゾーン負荷分散を使用すると、すべてのアベイラビリティゾーン (AZ) 間でバックエンドインスタンスへのトラフィックをルーティングできます。詳細については、「 AWS::ElasticLoadBalancing::LoadBalancer 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
AWS Elastic Beanstalk 環境枠	2013 年 12 月 19 日	AWS Elastic Beanstalk プロビジョニングリソースがウェブサーバーをサポートするか、バックグラウンドプロセスタスクを処理するかを指定できます。詳細については、「 AWS::ElasticBeanstalk::Environment 」を参照してください。	2010-05-15
リソース名	2013 年 12 月 19 日	次のリソースに名前 (物理 ID) を割り当てることができます: • ElastiCache クラスター • Elastic Load Balancing ロードバランサー • RDS DB インスタンス 詳細については、「Name タイプ」を参照してください。	2010-05-15
VPN サポート	2013 年 11 月 22 日	仮想プライベートゲートウェイ (VGW) による VPC のルーティングテーブルへのルートの伝達を有効にすることができます。詳細については、「 AWS::EC2::VPNGatewayRoutePropagation 」を参照してください。	2010-05-15
条件付きのリソースの作成とプロパティの割り当て	2013 年 11 月 8 日	入力パラメータを使用して、CloudFormation テンプレート内で条件を定義することによって、指定スタックリソースの作成と設定を制御できます。たとえば、条件を使用して、本稼働環境用のスタックリソースを作成することができます。同じテンプレートを使用して、より少ない容量のスタックリソースをテスト環境用に作成できます。詳細については、「 条件関数 」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
誤ってスタックリソースを更新することの回避	2013 年 11 月 8 日	スタックリソースへの意図しない変更によって発生する可能性があるスタックの更新を阻止できます。たとえば、ほとんど更新されないデータベース層を持つスタックがある場合は、ほとんどのユーザーがそのデータベース層を更新できないようにするスタックポリシーを設定できます。詳細については、「 スタックのリソースが更新されないようにする 」を参照してください。	2010-05-15
リソースの名前付け	2013 年 11 月 8 日	CloudFormation によって生成された物理 ID を使用する代わりに、特定のリソースに名前を割り当てることができます。次の CloudFormation リソースが名前付けをサポートしています • Amazon CloudWatch アラーム • DynamoDB テーブル • AWS Elastic Beanstalk アプリケーションおよび環境 • Amazon S3 バケット • Amazon SNS トピック • Amazon SQS キュー 詳細については、「Name タイプ」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
カスタムリソースタイプの割り当て	2013 年 11 月 8 日	テンプレートで、CloudFormation カスタムリソース (AWS::CloudFormation::CustomResource) に独自のリソースタイプを指定できます。独自のカスタムリソースタイプ名を使用すると、スタック内にあるカスタムリソースのタイプをすばやく識別できます。例えば、"Type": "Custom:: <i>MyCustomResource</i> " と指定することができます。詳細については、「 AWS::CloudFormation::CustomResource 」を参照してください。	2010-05-15
擬似パラメータの追加	2013 年 11 月 8 日	AWS::AccountID 擬似パラメータを参照することによって、CloudFormation テンプレート内の AWS AccountID を参照できるようになりました。詳細については、「 擬似パラメータのリファレンス 」を参照してください。	2010-05-15
IAM ポリシーでのスタックの指定	2013 年 11 月 8 日	IAM ユーザー、グループ、またはロールに、特定の CloudFormation スタックの操作を許可または拒否できます。たとえば、特定のスタック ID に対するスタック削除アクションを拒否できます。詳細については、 AWS Identity and Access Management によるアクセスの制御 をご参照ください。	2010-05-15

変更	リリース日	説明	APIバージョン
フェデレーションのサポート	2013 年 10 月 14 日	CloudFormation では、IAM ロールの一時的なセキュリティ認証情報がサポートされています。これにより、AWS マネジメントコンソール へのフェデレーションやシングルサインオンなどのシナリオが可能になります。また、IAM ロールを使用することで、長期的なセキュリティ認証情報を埋め込まなくても、EC2 インスタンスから CloudFormation への呼び出しを行うことができます。CloudFormation および IAM の詳細については、「 AWS Identity and Access Management によるアクセスの制御 」を参照してください。	2010-05-15
Amazon RDS リードレプリカのサポート	2013 年 9 月 24 日	ソース DB インスタンスから Amazon RDS リードレプリカを作成できるようになりました。詳細については、 AWS::RDS::DBInstance リソースの SourceDBInstanceIdentifier プロパティを参照してください。	2010-05-15
パブリック IP アドレスを Auto Scaling グループ内のインスタンスに関連付けます。	2013 年 9 月 19 日	パブリック IP アドレスを Auto Scaling グループ内のインスタンスに関連付けられるようになりました。詳細については、「 AWS::AutoScaling::LaunchConfiguration 」を参照してください。	2010-05-15

変更	リリース日	説明	API バージョン
追加の VPC のサポート	2013 年 9 月 17 日	VPC および VPN 機能をサポートするために、Cloud Formation にいくつかの機能強化が追加されます。 - パブリック IP アドレスおよび複数のプライベート IP アドレスを Amazon EC2 ネットワークインターフェースに関連付けることができます。詳細については、「AWS::EC2::NetworkInterface」を参照してください。プライマリプライベート IP アドレスを Elastic IP アドレス (EIP) に関連付けることもできます。 - DNS サポートを有効にし、DNS ホスト名を指定できます。詳細については、「AWS::EC2::VPC」を参照してください。 - 仮想プライベートゲートウェイと VPN ゲートウェイとの間の静的ルートを指定できます。詳細については、「AWS::EC2::VPNConnectionRoute」を参照してください。	2010-05-15
Redis および VPC セキュリティグループが Amazon ElastiCache をサポートします。	2013 年 9 月 3 日	Amazon ElastiCache (ElastiCache) クラスターのキャッシュエンジンとして Redis を指定できるようになりました。また、VPC セキュリティグループを ElastiCache クラスターに割り当てることもできるようになりました。詳細については、「AWS::ElastiCache::CacheCluster」を参照してください。	2010-05-15
並列的なスタックの作成、更新、および削除と、ネストされたスタックの更新	2013 年 8 月 12 日	AWS CloudFormation は、リソースの作成、更新、および削除を並行して行うことができるようになり、操作のパフォーマンスが向上しました。最上位のテンプレートを更新すると、変更されたネストされているスタックも AWS CloudFormation によって自動的に更新されます。詳細については、「CloudFormation スタックの更新」を参照してください。	2010-05-15

変更	リリース日	説明	APIバージョン
VPC セキュリティグループが、RDS DB インスタンスで設定できるようになりました	2013 年 2 月 28 日	CloudFormation で VPC セキュリティグループを RDS DB インスタンスに割り当てることができるようになりました。詳細については、 AWS::RDS::DBInstance の VPCSecurityGroups を参照してください。	2010-05-15
Amazon EC2 Auto Scaling グループのローリングデプロイ	2013 年 2 月 20 日	CloudFormation は、Amazon EC2 Auto Scaling グループに対する更新ポリシーをサポートするようになりました。このポリシーは、Amazon EC2 Auto Scaling グループがインスタンスを追加または削除するときに、Amazon EC2 Auto Scaling グループ内のインスタンスがどのように置き換えられたり変更されたりするかを指定します。スタック作成時またはスタック更新中に、これらの設定を変更できます。 詳細と例については、「 UpdatePolicy 属性 」を参照してください。	2010-05-15
スタック更新のキャンセルおよびロールバックアクション	2013 年 2 月 20 日	CloudFormation は、スタック更新をキャンセルする機能をサポートしています。更新リクエストが行われるときに、スタックは UPDATE_IN_PROGRESS 状態である必要があります。詳細については、以下の各トピックを参照してください。 • スタック更新のキャンセル • cancel-update-stack • CloudFormation API リファレンスの CancelUpdateStack	2010-05-15

変更	リリース日	説明	API バー ジョン
Amazon EC2 Auto Scaling グループの EBS 最適化インスタンス	2013 年 2 月 20 日	自動スケーリングされているインスタンスの Amazon Elastic Block Store (Amazon EBS) への専用のスループット用に、Amazon EC2 Auto Scaling グループ内の EBS 最適化インスタンスをプロビジョニングできるようになりました。実装は、以前にリリースされた、最適化された Amazon EBS EC2 インスタンスのサポートでの実装に似ています。 詳細については、AWS::AutoScaling::LaunchConfiguration の新しい EbsOptimized プロパティを参照してください。	2010-05-15
新規ドキュメント	2012 年 12 月 21 日	AWS::EC2::Instance では、EC2 インスタンスのブロックデバイスマッピングを設定できるように、BlockDeviceMappings プロパティが提供されるようになりました。 この変更により、2 つの新しいタイプが追加されました。 • BlockDeviceMapping • Block Device	2010-05-15
新規ドキュメント	2012 年 12 月 21 日	最近設計し直された AWS マネジメントコンソール を使用してスタックを作成および表示する手順を記述するために、新しいセクションが追加されました。それらは次のとおりです。 • CloudFormation コンソールからスタックを作成する • CloudFormation コンソールからスタック情報を表示する	2010-05-15

変更	リリース日	説明	APIバージョン
新規ドキュメント	2012 年 11 月 15 日	カスタムリソースについての情報は、以下のトピックに記載されています。 • カスタムリソース • AWS::CloudFormation::CustomResource • カスタムリソースリファレンス	2010-05-15
更新版	2012 年 11 月 15 日	CloudFormation は、RDS DB インスタンスのためのプロビジョニングされる 1 秒あたりの I/O 操作 (IOPS) の指定をサポートするようになりました。AWS::RDS::DBInstance の新しい Iops プロパティを使用してこの値を 1,000 IOPS 単位で、1,000 ~ 10,000 の範囲で設定できます。 RDS DB インスタンスに対する IOPS の指定の詳細については、「Amazon Relational Database Service ユーザーガイド」の「Provisioned IOPS」(プロビジョンド IOPS) を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジ ョ ン
新規のドキュメントおよび更新されたドキュメント	2012 年 8 月 27 日	AWS マネジメントコンソール の使用と CloudFormation コマンドラインインターフェイス (CLI) の使用に関する具体的な情報をより明確に提供するために、トピックが再編成されました。 CloudFormation スタックへのタグ付けについての情報が追加されました (新しいガイドや、更新されたリファレンストピックなど)。 - 新しいトピック: スタックオプションを設定する。 - CloudFormation API リファレンスのタグに関する新しい情報: CreateStack、Stack、Tag。 Windows スタックの操作に関する新しい情報: Microsoft Windows Amazon マシンイメージ (AMI) と CloudFormation テンプレート CloudFormation Windows スタックのブートストラップ 新しいトピック: CloudFormation テンプレートでの正規表現の使用。	2010-05-15

変更	リリース日	説明	APIバージョン
機能を追加	2012 年 4 月 25 日	CloudFormation は、Amazon EC2 で仮想プライベートクラウド (VPC) セキュリティを完全にサポートするようになりました。1 つの CloudFormation テンプレートを使用して VPC 全体を作成し、すべてのタイプの VPC リソース (サブネット、ゲートウェイ、ネットワーク ACL、ルートテーブルなど) を設定できるようになりました。 以下のリソースタイプのドキュメントが更新されました。 • AWS::EC2::SecurityGroup • AWS::EC2::SecurityGroupIngress • AWS::EC2::SecurityGroupEgress • AWS::EC2::Instance • AWS::AutoScaling::AutoScalingGroup • AWS::EC2::EIP • AWS::EC2::EIPAssociation • AWS::ElasticLoadBalancing::LoadBalancer 新しいリソースタイプがドキュメントに追加されました。 • AWS::EC2::VPC • AWS::EC2::InternetGateway • AWS::EC2::DHCPOptions • AWS::EC2::DHCPOptions • AWS::EC2::RouteTable • AWS::EC2::NetworkAcl	2010-05-15

変更	リリース日	説明	APIバージョン
		• AWS::EC2::NetworkAclEntry • AWS::EC2::Subnet • AWS::EC2::VPNGateway • AWS::EC2::CustomerGateway	
機能を追加	2012 年 4 月 13 日	CloudFormation で、スタックの更新時にスタックの要素を追加または削除できるようになりました。 CloudFormation スタックの更新 が更新され、チュートリアルに新しいセクションとして スタックのリソースの変更 が追加されました。このセクションでは、スタック更新時のリソースの追加および削除の方法が説明されています。	2010-05-15

変更	リリース日	説明	APIバージョン
機能を追加	2012 年 2 月 2 日	CloudFormation で、既存の Amazon Virtual Private Cloud (Amazon VPC) 内のリソースがサポートされるようになりました。このリリースでは、次の操作を行うことができます: • 既存の Amazon VPC で EC2 専用インスタンスを起動します。詳細については、「AWS::EC2::Instance」を参照してください。 • 既存の Amazon VPC 内に存在する EC2 インスタンスの SourceDestCheck 属性を設定します。詳細については、「AWS::EC2::Instance」を参照してください。 • 既存の Amazon VPC 内に Elastic IP アドレスを作成します。詳細については、「AWS::EC2::EIP」を参照してください。 • CloudFormation を使用して、既存の VPC 内に Amazon VPC セキュリティグループおよび Ingress/Egress ルールを作成します。詳細については、「AWS::EC2::SecurityGroup」を参照してください。 • <code>AWS::AutoScaling::AutoScalingGroup</code> リソースの <code>VPCZoneIdentifier</code> プロパティを設定して、Auto Scaling グループを既存の Amazon VPC と関連付けます。詳細については、「AWS::AutoScaling::AutoScalingGroup」を参照してください。 • Elastic Load Balancing ロードバランサーを Amazon VPC サブネットにアタッチし、ロードバランサーのセキュリティグループを作成します。詳細については、「AWS::ElasticLoadBalancing::LoadBalancer」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジョ ン
		既存の Amazon VPC で RDS DB インスタンスを作成します。詳細については、「AWS::RDS::DBInstance」を参照してください。	
機能を追加	2012 年 2 月 2 日	既存のスタック内の以下のリソースのプロパティを更新できるようになりました: AWS::EC2::SecurityGroupIngress AWS::EC2::SecurityGroupEgress AWS::EC2::EIPAssociation AWS::RDS::DBSubnetGroup AWS::RDS::DBSecurityGroup AWS::RDS::DBSecurityGroupIngress AWS::Route53::RecordSetGroup 更新可能なリソースの完全なリストと、スタックを更新するときの考慮事項の詳細については、「CloudFormation スタックの更新」を参照してください。	2010-05-15
ガイドの構成変更	2012 年 2 月 2 日	既存のセクションが新しいセクションに再編成されました。「CloudFormation テンプレートの使用」と「スタックの管理」です。「テンプレートリファレンス」を目次の先頭に移動しました。「CloudFormation スタックのコスト見積もり」を「はじめに」セクションに移動しました。	2010-05-15

変更	リリース日	説明	API バージョン
新しいコンテンツ	2012 年 2 月 2 日	新しいセクションが追加されました: 「ウォークスルー: スタックの更新」は、LAMP スタックの更新手順を説明するチュートリアルです。 「AWS CloudFormation による Amazon EC2 へのアプリケーションのデプロイ」では、CloudFormation ヘルパースクリプトを使用してテンプレートに保存されているメタデータを使用し、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。 「CloudFormation ヘルパースクリプトリファレンス」は、CloudFormation ヘルパースクリプト (cfn-init、cfn-get-metadata、cfn-signal、および cfn-hup) の参考資料を提供します。	2010-05-15
機能を追加	2011 年 5 月 26 日	CloudFormation は、list-stacks コマンドを提供するようになりました。これによって、スタックのステータスによってフィルターされたスタックを一覧表示できます。削除されたスタックは、削除されてから最大 90 日間、リストに表示されます。詳細については、「 Describing and Listing Your Stacks 」を参照してください。	2010-05-15
新機能	2011 年 5 月 26 日	describe-stack-resources および get-template コマンドを使用すると、削除されたスタックから、削除後 90 日間は情報を取得できるようになりました。詳細については、「 Listing Resources 」と「 Retrieving a Template 」を参照してください。	2010-05-15
新しいリンク	2011 年 3 月 1 日	CloudFormation エンドポイント情報が、AWS 全般のリファレンスに配置されるようになりました。詳細については、「 AWS の全般的なリファレンス 」の「リージョンとエンドポイント」を参照してください。	2010-05-15

変更	リリース日	説明	API バー ジヨ ン
初回リリース	2011 年 2 月 25 日	これは、CloudFormation の最初の一般リリースです。	2010-05-1 5