ユーザーガイド AWS CodeBuild



API バージョン 2016-10-06

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

AWS CodeBuild: ユーザーガイド

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

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客 に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできませ ん。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎり ません。

Table of Contents

とは AWS CodeBuild	1
 CodeBuild を実行する方法	1 1
CodeBuild 料金表	3
CodeBuild の開始方法	0 3
ecodeDuild V用如方法	J 3
Race CodeBuild の仕組み	0 3
ン の な テ に た の に に の に に の に に の に し の の に の の の の の の の の の の の の の	5
	5 6
コンソールを使用した開始方法	0 6
コンプ かどにパラルボガガ	0
ステップ 2: buildspec ファイルを作成する	10
ステップ 3: 2 つの S3 バケットを作成する	12
ステップ 4: ソースコードと buildspec ファイルをアップロードする	. 13
ステップ 5: ビルドプロジェクトを作成する	14
ステップ 6: ビルドを実行する	16
ステップ 7: ビルド情報の要約を表示する	17
ステップ 8: 詳細なビルド情報を表示する	18
ステップ 9: ビルド出力アーティファクトを取得する	19
ステップ 10: S3 入力バケットを削除する	. 20
まとめ	20
AWS CLIを使用した開始方法	21
ステップ 1: ソースコードを作成する	22
ステップ 2: buildspec ファイルを作成する	25
ステップ 3: 2 つの S3 バケットを作成する	27
ステップ 4: ソースコードと buildspec ファイルをアップロードする	. 28
ステップ 5: ビルドプロジェクトを作成する	29
ステップ 6: ビルドを実行する	33
ステップ 7: ビルド情報の要約を表示する	35
ステップ 8: 詳細なビルド情報を表示する	38
ステップ 9: ビルド出力アーティファクトを取得する	40
ステップ 10: S3 入力バケットを削除する	. 41
まとめ	42
ユースケースベースのサンプル	43

クロスサービス例	44
Amazon ECR のサンプル	45
Amazon EFS のサンプル	51
AWS CodePipeline サンプル	58
AWS Config サンプル	69
ビルド通知サンプル	71
ビルドバッジサンプル	86
ビルドバッジを使用してビルドプロジェクトを作成	87
AWS CodeBuild ビルドバッジにアクセスする	90
CodeBuild ビルドバッジの公開	91
CodeBuild バッジのステータス	91
テストレポートサンプル	91
テストレポートサンプルを実行	91
CodeBuild の Docker サンプル	98
カスタム Docker イメージのサンプル	99
Windows Docker ビルドのサンプル	102
'Docker イメージを Amazon ECR に公開' サンプル	104
AWS Secrets Manager サンプルを含むプライベートレジストリ	. 113
S3 バケットでのビルド出力のホスティング	117
複数の入出力のサンプル	. 120
複数の入力と出力を持つビルドプロジェクトを作成	. 121
ソースなしでプロジェクトを作成	. 124
buildspec ファイルサンプルのランタイムバージョン	. 125
buildspec ファイル内のランタイムバージョンを更新	. 125
2 つのランタイムの指定	. 130
ソースバージョンのサンプル	134
コミット ID で GitHub リポジトリバージョンを指定	135
参照とコミット ID を使用して GitHub リポジトリバージョンを指定	137
サードパーティーソースリポジトリのサンプル	138
Bitbucket サンプルを実行	139
GitHub Enterprise Server サンプルを実行	144
GitHub プルリクエストとウェブフックフィルタのサンプルを実行	151
チュートリアル: CodeBuild で証明書ストレージに S3 を使用して Fastlane で Apple コー	ド
署名する S3	156
チュートリアル: 証明書ストレージに GitHub を使用した CodeBuild での Fastlane を使用	し
た Apple コード署名	162

ビルド時にアーティファクト名を設定	168
Windows サンプルを実行	171
Windows サンプルを実行	171
ディレクトリ構造	172
F# と .NET Framework	173
Visual Basic と .NET Framework	173
ファイル	173
F# と .NET Framework	173
Visual Basic と .NET Framework	178
ビルドを計画する	192
ビルド仕様 (buildspec) に関するリファレンス	195
buildspec ファイル名とストレージの場所	195
buildspec の構文	196
version	199
run-as	199
env	200
proxy	205
phases	205
レポート	209
artifacts	211
cache	217
buildspec の例	219
buildspec のバージョン	222
バッチのビルド仕様 (buildspec) に関するリファレンス	223
バッチ	223
batch/build-graph	223
batch/build-list	226
<pre>batch/build-matrix</pre>	229
batch/build-fanout	231
ビルド環境に関するリファレンス	233
CodeBuild に用意されている Docker イメージ	234
現在の Docker イメージのリストを取得	234
EC2 コンピューティングイメージ	235
Lambda コンピューティングイメージ	237
非推奨の CodeBuild イメージ	241
使用可能なランタイム	243

ランタイムバージョン	261
ビルド環境のコンピューティングモードおよびタイプ	
コンピューティングについて	
リザーブドキャパシティ環境タイプについて	266
オンデマンド環境タイプについて	321
ビルド環境のシェルとコマンド	332
ビルド環境の環境変数	333
ビルド環境のバックグラウンドタスク	339
ビルドプロジェクト	
ビルドプロジェクトの作成	340
前提条件	341
ビルドプロジェクトの作成 (コンソール)	341
ビルドプロジェクトの作成 (AWS CLI)	363
ビルドプロジェクトを作成する (AWS SDKs)	
ビルドプロジェクトの作成 (AWS CloudFormation)	
通知ルールの作成	
ビルドプロジェクト設定を変更	387
ビルドプロジェクトの設定の変更 (コンソール)	
ビルドプロジェクトの設定の変更 (AWS CLI)	412
ビルドプロジェクトの設定の変更 (AWS SDK)	413
複数のアクセストークン	413
ステップ 1: Secrets Manager シークレットまたは CodeConnections 接続を作成	414
ステップ 2: CodeBuild プロジェクトの IAM ロールに Secrets Manager シークレッ	トへのア
クセスを許可	414
ステップ 3: Secrets Manager または CodeConnections トークンを設定	416
追加のセットアップオプション	420
ビルドプロジェクトを削除	423
ビルドプロジェクトの削除 (コンソール)	424
ビルドプロジェクトの削除 (AWS CLI)	424
ビルドプロジェクトの削除 (AWS SDKs)	425
パブリックビルドプロジェクトの URL を取得	425
ビルドプロジェクトを共有	426
プロジェクトを共有	427
関連サービス	429
共有プロジェクトにアクセス	430
共有プロジェクトを共有解除	430

共有プロジェクトを識別	431
共有プロジェクトへのアクセス許可	431
ビルドプロジェクトをタグ付け	431
プロジェクトにタグを追加する	432
プロジェクトのタグを表示する	434
プロジェクトのタグを編集する	435
プロジェクトからタグを削除する	436
ランナーを使用	437
GitHub Actions	437
GitLab ランナー	457
Buildkite ランナー	472
ウェブフックを使用	493
ウェブフック使用のベストプラクティス。	494
Bitbucket ウェブフックイベント	495
GitHub グローバルおよび組織のウェブフック	509
GitHub 手動ウェブフック	516
GitHub ウェブフックイベント	518
GitLab グループウェブフック	534
GitLab 手動ウェブフック	539
GitLab ウェブフックイベント	541
Buildkite 手動ウェブフック	555
ビルドプロジェクトの詳細を表示	557
ビルドプロジェクトの詳細を表示する (コンソール)	557
ビルドプロジェクトの詳細を表示する (AWS CLI)	557
ビルドプロジェクトの詳細を表示する (AWS SDK)	560
ビルドプロジェクト名を表示	560
ビルドプロジェクト名の一覧表示 (コンソール)	560
ビルドプロジェクト名の一覧表示 (AWS CLI)	560
ビルドプロジェクト名の一覧表示 (AWS SDK)	562
構築数	563
ビルドを手動で実行	564
ビルドをローカルで実行	564
ビルドの実行 (コンソール)	568
ビルドの実行 (AWS CLI)	569
バッチビルドの実行 (AWS CLI)	576
ビルドの実行の自動開始 (AWS CLI)	578

ビルドの実行の自動停止 (AWS CLI)	579
ビルドを実行する (AWS SDKs)	579
Lambda コンピューティングでビルドを実行	579
AWS Lambda上で実行される、選別されたランタイム環境の Docker イメージには、どの	
ツールとランタイムが含まれますか?	580
キュレートされたイメージに必要なツールが含まれていない場合はどうなりますか。	580
CodeBuild で AWS Lambda コンピューティングをサポートしているリージョンはどれです	
か?	581
AWS Lambda コンピューティングの制限	581
CodeBuild Lambda Java AWS SAM で を使用して Lambda 関数をデプロイする	582
CodeBuild Lambda Node.js を使用してシングルページの React アプリを作成	586
CodeBuild Lambda Python を使用して Lambda 関数の設定を更新	589
リザーブドキャパシティキャパシティフリートでビルドを実行	593
リザーブドキャパシティフリートを作成	594
ベストプラクティス	597
リザーブドキャパシティフリートを複数の CodeBuild プロジェクトで共有できますか?	597
属性ベースのコンピューティングの仕組み	597
フリートの Amazon EC2 インスタンスを手動で指定できますか?	598
リザーブドキャパシティフリートをサポートしているのはどのリージョンですか?	598
リザーブドキャパシティの macOS フリートを設定するにはどうすればよいですか。	599
リザーブドキャパシティフリートのカスタム Amazon マシンイメージ (AMI) を設定するに	
はどうすればよいですか?	600
リザーブドキャパシティフリートの制限	601
リザーブドキャパシティフリートのプロパティ	602
リザーブドキャパシティのサンプル	606
バッチビルドを実行	608
セキュリティロール	608
バッチビルドのタイプ	609
バッチレポートモード	613
詳細情報	613
並列テストを実行する	613
でのサポート AWS CodeBuild	615
バッチビルドで並列テストの実行を有効にする	617
codebuild-tests-run CLI コマンドを使用する	618
codebuild-glob-search CLI コマンドを使用する	621
テスト分割について	622

個々のビルドレポートを自動的にマージする	623
並列テスト実行サンプル	626
キャッシュビルド	636
Amazon S3 のキャッシュ	637
ローカルキャッシュ	643
ローカルキャッシュを指定	645
ビルドのデバッグ	647
CodeBuild サンドボックスを使用したビルドのデバッググ	647
Session Manager を使用したビルドのデバッグ	648
CodeBuild サンドボックスを使用したビルドのデバッググ	648
Session Manager を使用したビルドのデバッグ	678
ビルドの削除	683
ビルドの削除 (AWS CLI)	683
ビルドの削除 (AWS SDKs)	684
ビルドを手動で再試行	684
ビルドを手動で再試行 (コンソール)	684
ビルドを手動で再試行 (AWS CLI)	685
ビルドを手動で再試行する (AWS SDKs)	686
ビルドを自動的に再試行	686
ビルドを自動的に再試行 (コンソール)	686
ビルドを自動的に再試行 (AWS CLI)	687
ビルドを自動的に再試行する (AWS SDKs)	687
ビルドを停止	687
ビルドの停止 (コンソール)	688
ビルドの停止 (AWS CLI)	688
ビルドの停止 (AWS SDKs)	689
バッチビルドを停止	689
バッチビルドの停止 (コンソール)	689
バッチビルドを停止 (AWS CLI)	690
バッチビルドの停止 (AWS SDKs)	690
ビルドを自動的にトリガー	691
ビルドトリガーの作成	691
ビルドトリガーの編集	694
ビルドの詳細の表示	697
ビルドの詳細の表示 (コンソール)	697
ビルドの詳細の表示 (AWS CLI)	698

ビルドの詳細を表示する (AWS SDKs)	
ビルドフェーズの移行	699
ビルド ID を表示	699
ビルド ID の一覧表示 (コンソール)	699
ビルド ID の一覧表示 (AWS CLI)	
バッチビルド ID のリストを表示 (AWS CLI)	701
ビルド IDs (AWS SDKsのリストを表示する	703
ビルドプロジェクトのビルド ID を表示	
ビルドプロジェクトのビルド ID を一覧表示する (コンソール)	703
ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)	703
ビルドプロジェクトのバッチビルド ID のリストを表示 (AWS CLI)	705
ビルドプロジェクト (AWS SDKs) のビルド IDs のリストを表示する	
テストレポート	707
テストレポートの作成	
コードカバレッジレポートを作成	
コードカバレッジレポートの作成	
レポートを自動的に検出	711
コンソールを使用してレポートの自動検出を設定	712
プロジェクト環境変数を使用してレポートの自動検出を設定	713
レポートグループ	
Create a report group	714
Report group naming	720
レポートグループを共有	721
テストファイルの指定	
テストコマンドの指定	
レポートグループにタグを付ける	
レポートグループの更新	734
テストフレームワーク	737
Jasmine をセットアップ	
Jest をセットアップ	740
pytest のセットアップ	742
RSpec のセットアップ	743
テストレポートの表示	744
ビルドのテストレポートの表示	744
レポートグループのテストレポートの表示	745

AWS アカウントでのテストレポートの表示	
テストレポートのアクセス許可	
テストレポートの IAM ロール	
テストレポートオペレーションのアクセス許可	
テストレポートのアクセス許可の例	
テストレポートのステータス	748
VPC サポート	750
ユースケース	
VPC のベストプラクティス	751
VPC の制限事項	752
CodeBuild プロジェクトでの Amazon VPC アクセスを許可	
VPC 設定のトラブルシューティング	
VPC エンドポイントの使用	754
VPC エンドポイントを作成する前に	
CodeBuild の VPC エンドポイントを作成	
CodeBuild 用の VPC エンドポイントポリシーを作成する	
CodeBuild マネージドプロキシサーバーを使用する	756
リザーブドキャパシティフリートのマネージドプロキシ設定を構成	757
CodeBuild リザーブドキャパシティフリートを実行	758
プロキシサーバーの使用	758
プロキシサーバーで CodeBuild を実行するために必要なコンポーネントを設定	759
明示的なプロキシサーバーでの CodeBuild の実行	762
透過的なプロキシサーバーでの CodeBuild の実行	767
プロキシサーバーでのパッケージマネージャーなどのツールの実行	768
AWS CloudFormation VPC テンプレート	770
ログ記録とモニタリング	777
CodeBuild API コールをログに記録	777
CloudTrail AWS CodeBuild の情報について	777
AWS CodeBuild ログファイルエントリについて	778
ビルドをモニタリング	781
CloudWatch メトリクス	781
CloudWatch リソース使用率のメトリクス	
CloudWatch のディメンション	786
CloudWatch アラーム	
CodeBuild メトリクスを表示	787
CodeBuild リソース使用率メトリクスを表示	789

CloudWatch で CodeBuild アラームを作成	793
セキュリティ	795
データ保護	
データの暗号化	797
キー管理	798
トラフィックのプライバシー	798
Identity and Access Management	798
アクセス管理の概要	799
アイデンティティベースのポリシーの使用	803
AWS CodeBuild アクセス許可リファレンス	835
タグを使用した AWS CodeBuild リソースへのアクセスのコントロール	842
コンソールでのリソースの表示	846
コンプライアンス検証	847
耐障害性	848
インフラストラクチャセキュリティ	848
ソースプロバイダーのアクセス	849
Secrets Manager シークレットにトークンを作成して保存	849
GitHub および GitHub Enterprise Server アクセス	852
Bitbucket アクセス	864
GitLab アクセス	873
サービス間での不分別な代理処理の防止	879
高度なトピック	881
ユーザーに CodeBuild とのやり取りを許可	881
CodeBuild が他の AWS のサービスとやり取りすることを許可	888
ビルド出力を暗号化	896
を使用して CodeBuild を操作する AWS CLI	898
コマンドラインリファレンス	
AWS SDKsとツールのリファレンス	901
でサポートされている AWS SDKsとツール AWS CodeBuild	901
AWS SDKs の使用	
CodeBuild エンドポイントを指定	
AWS CodeBuild エンドポイントを指定する (AWS CLI)	904
AWS CodeBuild エンドポイントを指定する (AWS SDK)	904
、 CodePipeline で CodeBuild を使用	906
	907
パイプラインを作成する (コンソール)	

パイプラインを作成 (AWS CLI)	914
ビルドアクションを追加	919
テストアクションの追加	922
Codecov で CodeBuild を使用	926
Codecov とビルドプロジェクトの統合	926
Jenkins で CodeBuild を使用する	929
Jenkins の設定	930
プラグインをインストールする	930
プラグインを使用	930
サーバーレスアプリで CodeBuild を使用	932
関連リソース	933
サードパーティーの告知	933
1) 基本 Docker イメージ – windowsservercore	933
2) Windows ベースの Docker イメージ – Choco	935
3) Windows ベースの Docker イメージ – gitversion 2.16.2	935
4) Windows ベースの Docker イメージ – microsoft-build-toolsversion 15.0.26320.2	935
5) Windows ベースの Docker イメージ – nuget.commandlineversion 4.5.1	939
7) Windows ベースの Docker イメージ – netfx-4.6.2-devpack	940
8) Windows ベースの Docker イメージ – visualfsharptools, v 4.0	942
9) Windows ベースの Docker イメージ – netfx-pcl-reference-assemblies-4.6	942
10) Windows ベース Docker イメージ — visualcppbuildtools v 14.0.25420.1	946
11) Windows ベースの Docker イメージ – microsoft-windows-netfx3-ondemand-	
package.cab	950
12) Windows ベースの Docker イメージ – dotnet-sdk	951
CodeBuild 条件キーを IAM サービスロール変数として使用する	951
コードの例	953
基本	953
アクション	954
トラブルシューティング	971
Apache Maven が間違ったリポジトリのアーティファクトを参照している	972
ビルドコマンドがデフォルトでルートとして実行される	974
ファイル名に英語以外の文字が含まれているとビルドが失敗する場合があります	974
Amazon EC2 パラメータストアからパラメータを取得する際にビルドが失敗する場合がある	. 975
CodeBuild コンソールでブランチフィルタにアクセスできない	976
ビルドの成功または失敗を表示できない	976
ビルドステータスがソースプロバイダに報告されない	977

Windows Server Core 2019 プラットフォームの基本イメージが見つからず、選択できない 97	77
buildspec ファイルの以前のコマンドが、以降のコマンドで認識されない	77
 エラー: キャッシュのダウンロード時に「アクセスが拒否される」	78
エラー: 「BUILD CONTAINER UNABLE TO PULL IMAGE」カスタムビルドイメージを使用	
した場合	78
エラー: 「ビルドの完了前にビルドコンテナの停止が検出されました。ビルドコンテナがメ	
モリ不足のため停止したか、Docker イメージがサポートされていません」 エラーコード:	
500」	80
Error: "Cannot connect to the Docker daemon" when running a build (ビルドの実行時に	
「Docker デーモンに接続できません」)	80
エラー : ビルドプロジェクトを作成または更新する際、「CodeBuild は sts:AssumeRole の実	
行を許可されていません」	81
エラー: 「GetBucketAcl の呼び出しエラー: バケット所有者が変更されたか、サービスロール	
には、呼び出された s3:GetBucketAcl へのアクセス許可がありません」	82
エラー: ビルドの実行時に「アーティファクトのアップロードに失敗しました: 無効な	
ARN」	83
エラー: 「Git のクローンに失敗しました: ' your - repository - URL ' にアクセスできません:	
SSL 証明書の問題: 自己署名証明書」	83
エラー: ビルドの実行時に「アクセスしようとしているバケットは、指定されたエンドポイン	
トを使用してアドレス指定する必要があります」	84
エラー: "このビルドイメージではランタイムバージョンを少なくとも1つ選択する必要があり	
ます。"	84
ビルドキューのビルドが失敗するとエラー「QUEUED:INSUFFICIENT_SUBNET」が表示され	
る	85
エラー: 「キャッシュをダウンロードできません: RequestError: 次の理由により送信リクエス	
トが失敗しました: x509: システムのルートをロードできませんでした。ルートが指定されてい	
ません」	86
エラー:「S3 から証明書をダウンロードできません。AccessDenied"	86
エラー: 「認証情報を見つけることができません」	87
プロキシサーバーで CodeBuild を実行しているときの RequestError タイムアウトエラー 98	38
ビルドイメージ内に Bourne シェル (sh) が必要	89
警告 (ビルドの実行時): 「ランタイムのインストールをスキップします。このビルドイメージ	
ではランタイムバージョンの選択はサポートされていません」	90
エラー:「JobWorker ID を確認できません」	90
ビルドの開始の失敗	90
ローカルにキャッシュされたビルドの GitHub メタデータへのアクセス	91

AccessDenied: レポートグループのバケット所有者が S3 バケットの所有者と一致しません。	991
エラー: CodeConnections で CodeBuild プロジェクトを作成するときに、「認証情報に必要な	
権限スコープが 1 つ以上ありません」	992
エラー: Ubuntu install コマンドでビルドするときに「すみません、ターミナルがまったくリク	
エストされていません - 入力を取得できません」	993
クォータ	995
Service Quotas	995
その他の制限	001
ビルドプロジェクト	001
構築数	001
コンピューティングフリート	001
レポート	003
[タグ]	003
ドキュメント履歴	005
以前の更新	029
	nxlii

とは AWS CodeBuild

AWS CodeBuild は、 クラウドでのフルマネージドビルドサービスです。CodeBuild はソースコー ドをコンパイルし、単体テストを実行して、すぐにデプロイできるアーティファクトを生成しま す。CodeBuild では自分のビルドサーバーをプロビジョニング、管理、スケールする必要がありませ ん。Apache Maven、Gradle などの一般的なプログラミング言語とビルドツール用のパッケージ済み のビルド環境を提供します。ビルド環境をカスタマイズして、CodeBuild で独自のビルドツールを使 用することもできます。CodeBuild はピーク時のビルドリクエストに合わせて自動的にスケーリング します。

CodeBuild には、以下のような利点があります。

- 完全マネージド型 CodeBuild では、お客様独自のビルドサーバーをセットアップ、パッチ適用、 更新、管理する必要がありません。
- オンデマンド CodeBuild はビルドのニーズに合わせてオンデマンドでスケールされます。料金は、使用したビルド分数に対してのみ発生します。
- すぐに利用可能 CodeBuild は、最も一般的なプログラミング言語用に事前設定されたビルド環境 を提供します。最初のビルドを開始するには、ビルドスクリプトを指すだけです。

詳細については、「AWS CodeBuild」を参照してください。

CodeBuild を実行する方法

AWS CodeBuild または AWS CodePipeline コンソールを使用して CodeBuild を実行できます。また、 AWS Command Line Interface (AWS CLI) または AWS SDKs を使用して CodeBuild の実行を 自動化することもできます。



次の図に示すように、CodeBuild をビルドまたはテストアクションとして のパイプラインのビルド またはテストステージに追加できます AWS CodePipeline。 AWS CodePipeline は、コードをリリー スするために必要なステップをモデル化、視覚化、自動化するために使用できる継続的な配信サービ スです。これには、コードの構築が含まれます。パイプラインは、リリースプロセスを通したコード の変更を説明したワークフロー構造です。



CodePipeline を使用してパイプラインを作成し、CodeBuild ビルドまたはテストアクションを追加 するには、「<u>CodePipeline で CodeBuild を使用</u>」を参照してください。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、CodeBuild コンソールでは、リポジトリ、ビルドプロジェクト、デプロイアプリケーショ ン、パイプラインなどのリソースをすばやく検索することもできます。[Go to resource] を選択する か、/ キーを押して、リソースの名前を入力します。一致するものはすべてリストに表示されます。 検索では大文字と小文字が区別されません。リソースを表示する権限がある場合のみ表示されます。 詳細については、「コンソールでのリソースの表示」を参照してください。

CodeBuild 料金表

詳細については、「CodeBuild の料金」を参照してください。

CodeBuild の開始方法

次の手順を実行することをお勧めします。

- 1. CodeBuild の詳細については、「概念」の情報を参照してください。
- 2. 「<u>コンソールを使用した開始方法</u>」の手順に従って、サンプルのシナリオで CodeBuild を試し てみてください。
- 3. 独自のシナリオで CodeBuild を使用するには、「ビルドを計画する」の手順に従います。

AWS CodeBuild の概念

以下の概念は、CodeBuild の仕組みを理解するうえで重要です。

トピック

- CodeBuild の仕組み
- <u>次のステップ</u>

CodeBuild の仕組み

次の図は、CodeBuild でビルドを実行するとどうなるかを示しています。



- 入力として、CodeBuild にビルドプロジェクトを指定する必要があります。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表します。詳細については、以下を参照してください。
 - ビルドプロジェクトの作成
 - ビルド環境に関するリファレンス
- 2. CodeBuild は、ビルドプロジェクトを使用して、ビルド環境を作成します。
- CodeBuild は、ビルド環境にソースコードをダウンロードし、ビルドプロジェクトで定義されている、または、ソースコードに直接含まれているビルド仕様 (buildspec) を使用します。ビルド 環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プログラミ ング言語ランタイム、およびツールの組み合わせを表します。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。
- ビルド出力がある場合、ビルド環境はその出力をS3バケットにアップロードします。ビルド環境では、buildspecで指定したタスク (たとえば、ビルド通知を Amazon SNS トピックに送信す)

るなど) を実行することもできます。例については、「<u>ビルド通知サンプル</u>」を参照してください。

- 5. ビルドが実行されている間に、ビルド環境は CodeBuild および Amazon CloudWatch Logs に情報を送信します。
- ビルドの実行中に、AWS CodeBuild コンソールまたは AWS SDKs を使用して、CodeBuild か らビルド情報の概要を取得し AWS CLI、Amazon CloudWatch Logs からビルド情報の詳細を取 得できます。 AWS CodePipeline を使用してビルドを実行する場合、CodePipeline から制限さ れたビルド情報を取得できます。

次のステップ

詳細については AWS CodeBuild、次のステップをお勧めします。

- 1. 「<u>コンソールを使用した開始方法</u>」の手順に従って、サンプルのシナリオで CodeBuild を試し てみてください。
- 2. 独自のシナリオで CodeBuild を使用するには、「ビルドを計画する」の手順に従います。

CodeBuild の開始方法

以下のチュートリアルでは、 AWS CodeBuild を使用して、サンプルソースコード入力ファイルのコ レクションをソースコードのデプロイ可能なバージョンに構築します。

どちらのチュートリアルでも入力と結果は同じですが、一方は AWS CodeBuild コンソールを使用 し、もう一方は を使用します AWS CLI。

▲ Important

このチュートリアルを完了するために AWS ルートアカウントを使用することはお勧めしま せん。

トピック

- コンソール AWS CodeBuild を使用したの開始方法
- AWS CodeBuild を使用したの開始方法 AWS CLI

コンソール AWS CodeBuild を使用した の開始方法

このチュートリアルでは、AWS CodeBuild を使用して、サンプルソースコード入力ファイルの コレクション (ビルド入力アーティファクトまたはビルド入力) をソースコードのデプロイ可能な バージョン (ビルド出力アーティファクトまたはビルド出力) に構築します。具体的には、一般的な ビルドツールである Apache Maven を使用して Java クラスファイルのセットを Java アーカイブ (JAR) ファイルにビルドするように CodeBuild に指示します。このチュートリアルを完了するため に、Apache Maven または Java に精通している必要はありません。

CodeBuild は、CodeBuild コンソール、 AWS CodePipeline、 AWS CLIまたは AWS SDKs CodeBuild を使用して操作できます。このチュートリアルでは、CodeBuild コンソールを使用する方 法を示します。CodePipeline の使用については、「<u>CodePipeline で CodeBuild を使用</u>」を参照して ください。

A Important

このチュートリアルのステップでは、 AWS アカウントに課金される可能性のあるリソース (S3 バケットなど) を作成する必要があります。これには、Amazon S3、および CloudWatch Logs に関連する AWS リソースとアクションに対する CodeBuild AWS KMSと の料金が含ま れます。 Amazon S3 詳細については、<u>AWS CodeBuild 料金表</u>、<u>Amazon S3 料金表</u>、<u>AWS</u> <u>Key Management Service 料金表</u>、および <u>Amazon CloudWatch 料金表</u>を参照してくださ い。

トピック

- ステップ 1: ソースコードを作成する
- ステップ 2: buildspec ファイルを作成する
- ステップ 3:2 つの S3 バケットを作成する
- ステップ 4: ソースコードと buildspec ファイルをアップロードする
- ステップ 5: ビルドプロジェクトを作成する
- ステップ 6: ビルドを実行する
- ステップ 7: ビルド情報の要約を表示する
- ステップ 8: 詳細なビルド情報を表示する
- ステップ 9: ビルド出力アーティファクトを取得する
- ステップ 10: S3 入力バケットを削除する
- まとめ

ステップ 1: ソースコードを作成する

(一部: コンソール AWS CodeBuild を使用した の開始方法)

このステップでは、CodeBuild が出力バケットにビルドするソースコードを作成します。このソー スコードは 2 つの Java クラスファイルと Apache Maven プロジェクトオブジェクトモデル (POM) ファイルで構成されています。

 ローカルコンピュータまたはインスタンスの空のディレクトリに、このディレクトリ構造を作成 します。

```
(root directory name)
   `-- src
        |-- main
        | `-- java
        `-- test
```

`-- java

 任意のテキストエディタを使用して、このファイルを作成し、MessageUtil.java という名前 を付けて、src/main/java ディレクトリに保存します。

```
public class MessageUtil {
    private String message;

    public MessageUtil(String message) {
        this.message = message;
    }

    public String printMessage() {
        System.out.println(message);
        return message;
    }

    public String salutationMessage() {
        message = "Hi!" + message;
        System.out.println(message);
        return message;
    }
}
```

このクラスファイルは、渡された文字列を出力として作成します。MessageUtil コ ンストラクタは、文字列を設定します。printMessage メソッドは出力を作成しま す。salutationMessage メソッドが Hi! を出力した後に文字列が続きます。

 このファイルを作成し、TestMessageUtil.java という名前を付けて、/src/test/java ディレクトリに保存します。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;
public class TestMessageUtil {
   String message = "Robert";
   MessageUtil messageUtil = new MessageUtil(message);
   @Test
   public void testPrintMessage() {
      System.out.println("Inside testPrintMessage()");
```

```
assertEquals(message,messageUtil.printMessage());
}
@Test
public void testSalutationMessage() {
   System.out.println("Inside testSalutationMessage()");
   message = "Hi!" + "Robert";
   assertEquals(message,messageUtil.salutationMessage());
}
```

このクラスファイルは message クラスの MessageUtil 変数を Robert に設定します。そ の後、文字列 message および Robert が出力に表示されているかどうかを調べることによっ て、Hi!Robert 変数が正常に設定されたかどうかを調べます。

4. このファイルを作成し、pom.xml という名前を付けて、ルート (最上位) ディレクトリに保存します。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"</pre>
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
```

```
</plugins>
</build>
</project>
```

Apache Maven では、このファイルの指示に従って、MessageUtil.java および TestMessageUtil.java ファイルを messageUtil-1.0.jar という名前のファイルに変換 し、指定されたテストを実行します。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
  |-- pom.xml
  `-- src
        |-- main
        | `-- java
        | `-- MessageUtil.java
        `-- test
        `-- java
        `-- TestMessageUtil.java
```

ステップ 2: buildspec ファイルを作成する

(前のステップ: ステップ 1: ソースコードを作成する)

このステップでは、ビルド仕様ファイルを作成します。ビルド環境は、CodeBuild がビルドを実行す るために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み 合わせを表します。CodeBuild では、ビルド仕様がないと、ビルド入力をビルド出力に正常に変換で きません。また、ビルド環境でビルド出力アーティファクトを特定して出力バケットにアップロード することもできません。

このファイルを作成し、buildspec.yml という名前を付けて、ルート (最上位) ディレクトリに保 存します。

```
version: 0.2
phases:
    install:
        runtime-versions:
            java: correttol1
    pre_build:
            commands:
```

```
- echo Nothing to do in the pre_build phase...
build:
    commands:
        - echo Build started on `date`
        - mvn install
    post_build:
        commands:
        - echo Build completed on `date`
artifacts:
    files:
        - target/messageUtil-1.0.jar
```

▲ Important

ビルド仕様宣言は有効な YAML である必要があるため、ビルド仕様宣言のスペースは重要で す。ビルド仕様宣言のスペース数が YAML と一致しない場合、ビルドは即座に失敗する場合 があります。YAML validator を使用して、ビルド仕様宣言が有効な YAML かどうかをテスト できます。

Note

ソースコードにビルド仕様ファイルを含める代わりに、ビルドプロジェクトを作成するとき に個別にビルドコマンドを宣言することができます。これは、毎回ソースコードのリポジト リを更新せずに、異なるビルドコマンドでソースコードをビルドする場合に役立ちます。詳 細については、「buildspec の構文」を参照してください。

このビルド仕様宣言の詳細は次の通りです。

- versionは、使用されているビルド仕様スタンダードのバージョンを表します。このビルド仕様 宣言では、最新バージョン 0.2 が使用されます。
- phases は、CodeBuild にコマンドの実行を指示するビルドフェーズを表します。これらのビル ドフェーズは install、pre_build、build、post_build として、ここにリストされていま す。これらのビルドフェーズ名のスペルを変更することはできず、追加のビルドフェーズ名を作成 することもできません。

この例では、build フェーズ中に、 CodeBuild によって mvn install コマンドが実行されま す。このコマンドは、コンパイルされた Java クラスファイルをビルド出力アーティファクトにコ ンパイル、テスト、パッケージ化するように Apache Maven に指示します。完全にするために、 この例では、いくつかの echo コマンドが各ビルドフェーズに配置されています。このチュートリ アルの後半で詳細なビルド情報を見る際に、これらの echo コマンドの出力は、CodeBuild がコマ ンドを実行する方法とその順序を理解するのに役立ちます。(この例にはすべてのビルドフェーズ が含まれていますが、コマンドを実行しないビルドフェーズは含める必要がありません)。各ビル ドフェーズについて、CodeBuild は最初から最後まで、各コマンドを一度に1つずつ、指定された 順序で実行します。

「artifacts」は、CodeBuild が出力バケットにアップロードする一連のビルド出力アーティファクトを表します。「files」は、ビルド出力に含めるファイルを表します。CodeBuild は、ビルド環境の「messageUtil-1.0.jar」相対ディレクトリにある、単一の「target」ファイルをアップロードします。ファイル名 messageUtil-1.0.jar およびディレクトリ名 target は、この例でのみ Apache Maven がビルド出力アーティファクトを作成して格納する方法に基づいています。独自のビルドでは、これらのファイル名とディレクトリは異なります。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

この時点で、ディレクトリ構造は次のようになります。

ステップ 3:2 つの S3 バケットを作成する

(前のステップ: <u>ステップ 2: buildspec ファイルを作成する</u>)

このチュートリアル用として1つのバケットを使用することもできますが、2 つのバケットを使用す る方が、ビルド入力の送信元およびビルド出力の送信先の確認が簡単になります。

これらのバケットのいずれか(入力バケット)にビルド入力が保存されます。このチュートリアルでは、この入力バケットの名前はです。ここでcodebuild-<u>region-ID-account-ID</u>-input-

bucket、*region-ID* はバケットの AWS リージョン、*account-ID* は AWS アカウント ID で す。

もう1つのバケット(出力バケット)にはビルド出力が保存されます。このチュートリアルでは、この出力バケットの名前は codebuild-*region-ID-account-ID*-output-bucket です。

これらのバケットに別の名前を選択した場合は、このチュートリアル全体で、その名前を使用してく ださい。

これら 2 つのバケットは、ビルドと同じ AWS リージョンに存在する必要があります。たとえば、米 国東部 (オハイオ) リージョンでビルドを実行するように CodeBuild に指示している場合、バケット は米国東部 (オハイオ) リージョンにある必要があります。

詳細については、Amazon Simple Storage Service コンソールユーザーガイドの<u>「バケットの作</u> 成」を参照してください。

Note

CodeBuild では、CodeCommit、GitHub、および Bitbucket の各リポジトリに保存されている ビルド入力もサポートされますが、このチュートリアルではこれらの使用方法については説 明しません。詳細については、「<u>ビルドを計画する</u>」を参照してください。

ステップ 4: ソースコードと buildspec ファイルをアップロードする

(前のステップ: ステップ 3:2 つの S3 バケットを作成する)

このステップでは、入力バケットにソースコードとビルド仕様ファイルを追加します。

オペレーティングシステムの zip ユーティリティを使用し

て、MessageUtil.zip、MessageUtil.java、TestMessageUtil.java、および pom.xml を 含む buildspec.yml という名前のファイルを作成します。

MessageUtil.zip ファイルのディレクトリ構造は、次のようになっている必要があります。

```
| `-- MessageUtil.java
`-- test
    `-- java
    `-- TestMessageUtil.java
```

▲ Important

(root directory name) ディレクトリを含めないでください。(root directory name) ディレクトリ内のディレクトリとファイルのみを含めます。

MessageUtil.zip ファイルを codebuild-*region-ID-account-ID*-input-bucket という名 前の入力バケットにアップロードします。

A Important

CodeCommit、GitHub、および Bitbucket の各リポジトリでは、規約に従っ て、buildspec.yml というビルド仕様ファイルを各リポジトリのルート (最上位) に保存す るか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要があります。リポ ジトリのソースコードとビルド仕様ファイルを含む ZIP ファイルを作成しないでください。 S3 バケットに保存されたビルド入力に限り、ソースコードおよび規約に基づく buildspec.yml というビルド仕様ファイルを圧縮した ZIP ファイルをルート(最上位) に作成するか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要がありま す。 ビルド仕様ファイルに別の名前を使用するか、ルート以外の場所でビルド仕様を参照する場 合は、ビルドプロジェクト定義の一部としてビルド仕様の上書きを指定できます。詳細につ

いては、「buildspec ファイル名とストレージの場所」を参照してください。

ステップ 5: ビルドプロジェクトを作成する

(前のステップ: ステップ 4: ソースコードと buildspec ファイルをアップロードする)

このステップでは、 AWS CodeBuild を使用してビルドを実行するビルドプロジェクトを作成しま す。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソー スコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれま す。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プロ グラミング言語ランタイム、およびツールの組み合わせを表します。ビルド環境は Docker イメージ として表されます。詳細については、Docker Docs ウェブサイトの <u>Docker overview</u> を参照してくだ さい。

このビルド環境では、CodeBuild に、Java 開発キット (JDK) のバージョンおよび Apache Maven が 含まれる Docker イメージを使用するように指示します。

ビルドプロジェクトを作成するには

- 1. にサインイン AWS Management Console し、 AWS CodeBuild コンソールを <u>https://</u> <u>console.aws.amazon.com/codesuite/codebuild/home</u>://www.com で開きます。
- AWS リージョンセレクタを使用して、CodeBuild がサポートされている AWS リージョン を選択します。詳細については、「Amazon Web Services 全般のリファレンス」の「<u>AWS</u> CodeBuild エンドポイントとクォータ」を参照してください。
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [Create build project (ビルドプロジェクトの作成)] ページで、[Project configuration (プロジェ クト設定)] の [プロジェクト名] にこのビルドプロジェクトの名前を入力します (この例で は、codebuild-demo-project)。ビルドプロジェクト名は、各 AWS アカウントで一意であ る必要があります。別の名前を選択した場合は、このチュートリアル全体でその名前を使用して ください。

Note

[Create build project (ビルドプロジェクトの作成)] ページに「このオペレーションを実 行する権限がありません」というようなエラーメッセージが表示される場合がありま す。これは、ビルドプロジェクトを作成するアクセス許可を持たないユーザー AWS Management Console として にサインインしたことが原因である可能性が高くなりま す。これを修正するには、 からサインアウトし AWS Management Console、次のいず れかの IAM エンティティに属する認証情報でサインインし直します。

- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初 の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- AWS アカウントのユーザーでAWSCodeBuildAdminAccess、そのユーザーまたは ユーザーが属する IAM グループにアタッチされた AmazonS3ReadOnlyAccess、、 IAMFullAccess管理ポリシーを持つユーザー。これらのアクセス許可を持つ ユー ザーまたはグループが AWS アカウントになく、これらのアクセス許可をユーザーま

たはグループに追加できない場合は、AWS アカウント管理者にお問い合わせくださ い。詳細については、「<u>AWS の 管理 (事前定義) ポリシー AWS CodeBuild</u>」を参照し てください。

どちらのオプションにも、このチュートリアルを完了できるように、ビルドプロジェクトの作成を許可する管理者権限が含まれています。常に必要最小限のアクセス許可を使用してタスクを達成してください。詳細については、「<u>AWS CodeBuild アクセス許可リ</u>ファレンス」を参照してください。

- 5. [Source] (ソース) で、[Source provider] (ソースプロバイダー) として [Amazon S3] を選択しま す。
- 6. [Bucket] (バケット) として、[codebuild-*region-ID-account-ID*-input-bucket] を選択します。
- 7. [S3 オブジェクトキー] に「MessageUtil.zip」と入力します。
- 8. [環境] の [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択したままにしてお きます。
- 9. [オペレーティングシステム] で、[Amazon Linux] を選択します。
- 10. [ランタイム] で、[Standard (標準)] を選択します。
- 11. Image で、aws/codebuild/amazonlinux-x86_64-standard:corretto11を選択します。
- 12. [サービスロール] で、[New service role (新しいサービスロール)] は選択したままにして、[Role name (ロール名)] は変更しません。
- 13. [Buildspec] で、[Use a buildspec file (buildspec ファイルを使用)] を選択したままにしておきま す。
- 14. [Artifacts] (アーティファクト) で、[Type] (タイプ) として [Amazon S3] を選択します。
- 15. [Bucket name] (バケット名) として、[codebuild-*region-ID-account-ID*-output-bucket] を選 択します。
- 16. [名前] と [パス] を空白のままにします。
- 17. [Create build project (ビルドプロジェクトの作成)] を選択します。

ステップ 6: ビルドを実行する

(前のステップ: ステップ 5: ビルドプロジェクトを作成する)

このステップでは、ビルドプロジェクトの設定を使用してビルドを実行する AWS CodeBuild ように に指示します。

ステップ 6: ビルドを実行する

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- 2. ナビゲーションペインで、[Build projects] を選択します。
- 3. ビルドプロジェクトのリストで、[codebuild-demo-project]、[Start build] (ビルドの開始) の順に 選択します。ビルドがすぐに開始されます。

ステップ 7: ビルド情報の要約を表示する

(前のステップ: ステップ 6: ビルドを実行する)

このステップでは、ビルドのステータスに関する要約情報を表示します。

要約されたビルド情報を表示するには

- [codebuild-demo-project: < build-ID>] ページが表示されない場合は、ナビゲーションバーで [Build history] (ビルド履歴) を選択します。次にビルドプロジェクトのリストで、[Project] (プロ ジェクト) の [codebuild-demo-project] に対応する [Build run] (ビルドの実行) リンクを選択しま す。一致するリンクは 1 つだけです。(このチュートリアルを完了したことがある場合は、[完 了済み] 列で最新の値のリンクを選択します。)
- [Build status] (ビルドステータス) ページの [Phase details] (フェーズ詳細) に以下のビルドフェーズが表示され、[Status] (ステータス) 列に [Succeeded] (成功) と示されます。
 - SUBMITTED
 - QUEUED
 - PROVISIONING
 - DOWNLOAD_SOURCE
 - INSTALL
 - PRE_BUILD
 - BUILD
 - POST_BUILD
 - UPLOAD_ARTIFACTS
 - FINALIZING
 - COMPLETED

[ビルドステータス] で、[Succeeded (成功)] が表示されます。

代わりに [進行中] と表示される場合は、更新ボタンを選択します。

3. 各ビルドフェーズの横に表示される [所要時間] 値は、ビルドフェーズの所要時間を示します。 [終了時間] 値は、ビルドフェーズの完了日時を示します。

ステップ 8: 詳細なビルド情報を表示する

(前のステップ: ステップ 7: ビルド情報の要約を表示する)

このステップでは、CloudWatch Logs のビルドに関する詳細情報を表示します。

Note

機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。

- ・ AWS アクセスキー IDs。詳細については、AWS Identity and Access Management ユー ザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
- パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「<u>Systems Manager パラメータストア</u>」および 「<u>Systems Manager パラメータストアコンソールのチュートリアル</u>」を参照してください。
- を使用して指定された文字列 AWS Secrets Manager。詳細については、「<u>キー管理</u>」を参 照してください。

詳細なビルド情報を表示するには

- ビルドの詳細ページが前のステップから引き続き表示された状態で、ビルドログの最後の 10,000 行が [Build logs] に表示されます。CloudWatch Logs でビルドログ全体を表示するに は、[View entire log] (ログ全体の表示) リンクを選択します。
- CloudWatch Logs ログストリームでは、ログイベントを参照できます。デフォルトでは、ログ イベントの最後のセットだけが表示されます。以前のログイベントを表示するには、リストの先 頭にスクロールします。
- このチュートリアルでは、ほとんどのログイベントに、CodeBuild でビルド依存ファイルをビル ド環境にダウンロードおよびインストールする操作に関する詳細情報が含まれますが、これらの

情報は不要な場合があります。[Filter events] ボックスを使用すると、表示する情報量を減らす ことができます。例えば、[Filter events] (イベントのフィルター) で「"[INF0]"」と入力した 場合、「[INF0]」を含むイベントのみが表示されます。詳細については、Amazon CloudWatch ユーザーガイドの「フィルターとパターンの構文」を参照してください。

ステップ 9: ビルド出力アーティファクトを取得する

(前のステップ: ステップ 8: 詳細なビルド情報を表示する)

このステップでは、CodeBuild が構築して出力バケットにアップロードした 「messageUtil-1.0.jar」ファイルを取得します。

このステップを完了するには、CodeBuild コンソールまたは Amazon S3 コンソールを使用します。

ビルド出力アーティファクトを取得するには (AWS CodeBuild コンソール)

CodeBuild コンソールが開いていて、ビルドの詳細ページが前のステップから引き続き表示されている状態で、[Build details] を選択して [Artifacts] セクションまでスクロールします。

1 Note

ビルドの詳細ページが表示されていない場合は、ナビゲーションバーで [Build history] (ビルド履歴)、[Build run] (ビルドの実行) リンクの順に選択します。

 Amazon S3 フォルダへのリンクは、[Artifacts upload location] (アーティファクトのアップロー ド場所)の下にあります。Amazon S3 内のフォルダが開き、「messageUtil-1.0.jar」とい う名前のビルド出力アーティファクトファイルを見つけます。

ビルド出力アーティファクトを取得するには (Amazon S3 コンソール)

- 1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
- 2. codebuild-*region-ID-account-ID*-output-bucket を開きます。
- 3. codebuild-demo-project フォルダを開きます。
- target という名前のフォルダを開き、messageUtil-1.0.jar という名前のビルド出力アー ティファクトファイルを見つけます。

ステップ 10: S3 入力バケットを削除する

(前のステップ: ステップ 9: ビルド出力アーティファクトを取得する)

AWS アカウントへの継続的な課金を防ぐために、このチュートリアルで使用されている入出力バ ケットを削除できます。手順については、Amazon Simple Storage Service ユーザーガイドで<u>バケッ</u> トを削除、または空にする方法を参照してください。

IAM ユーザー を使用している場合、このバケットを削除するユーザーまたは管理者 IAM ユーザー には、さらに高いアクセス権限が必要です。マーカー間で次のステートメント (###BEGIN ADDING STATEMENT HERE### と ###END ADDING STATEMENTS HERE###)を既存のアクセスポリシーに 追加します。

このステートメントでは、簡潔にするために省略記号 (...) が使用されています。既存のアクセスポ リシーのステートメントは削除しないでください。これらの省略記号はポリシーに入力しないでくだ さい。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

まとめ

このチュートリアルでは、 AWS CodeBuild を使用して一連の Java クラスファイルを JAR ファイル に構築しました。次に、ビルドの結果を表示しました。 これで、独自のシナリオに CodeBuild を使用できます。「<u>ビルドを計画する</u>」の手順に従ってくだ さい。もう少し準備が必要な場合は、用意されているサンプルでビルドを試すことができます。詳細 については、「CodeBuild のユースケースベースのサンプル」を参照してください。

AWS CodeBuild を使用した の開始方法 AWS CLI

このチュートリアルでは、AWS CodeBuild を使用して、サンプルソースコード入力ファイル (ビル ド入力アーティファクトまたはビルド入力と呼ばれる) のコレクションを、ソースコードのデプロイ 可能なバージョン (ビルド出力アーティファクトまたはビルド出力と呼ばれる) に構築します。具体 的には、一般的なビルドツールである Apache Maven を使用して Java クラスファイルのセットを Java アーカイブ (JAR) ファイルにビルドするように CodeBuild に指示します。このチュートリアル を完了するために、Apache Maven または Java に精通している必要はありません。

CodeBuild は、CodeBuild コンソール、 AWS CodePipeline、 AWS CLIまたは AWS SDKs CodeBuild を使用して操作できます。このチュートリアルでは、 AWS CLIで CodeBuild を使用する 方法を示します。CodePipeline の使用については、「<u>CodePipeline で CodeBuild を使用</u>」を参照し てください。

A Important

このチュートリアルのステップでは、AWS アカウントに課金される可能性のあるリソース (S3 バケットなど) を作成する必要があります。これには、Amazon S3、および CloudWatch Logs に関連する AWS リソースとアクションに対する CodeBuild AWS KMSと の料金が含 まれます。 Amazon S3 詳細については、<u>CodeBuild 料金表、Amazon S3 料金表</u>、<u>AWS Key</u> <u>Management Service 料金表</u>、および <u>Amazon CloudWatch 料金表</u>を参照してください。

トピック

- <u>ステップ 1: ソースコードを作成する</u>
- ステップ 2: buildspec ファイルを作成する
- ステップ 3:2 つの S3 バケットを作成する
- ステップ 4: ソースコードと buildspec ファイルをアップロードする
- ステップ 5: ビルドプロジェクトを作成する
- ステップ 6: ビルドを実行する
- ステップ 7: ビルド情報の要約を表示する
- ステップ 8: 詳細なビルド情報を表示する
- ステップ 9: ビルド出力アーティファクトを取得する
- ステップ 10: S3 入力バケットを削除する
- まとめ

ステップ 1: ソースコードを作成する

(一部: AWS CodeBuild を使用した の開始方法 AWS CLI)

このステップでは、CodeBuild が出力バケットにビルドするソースコードを作成します。このソー スコードは 2 つの Java クラスファイルと Apache Maven プロジェクトオブジェクトモデル (POM) ファイルで構成されています。

 ローカルコンピュータまたはインスタンスの空のディレクトリに、このディレクトリ構造を作成 します。

 任意のテキストエディタを使用して、このファイルを作成し、MessageUtil.java という名前 を付けて、src/main/java ディレクトリに保存します。

```
public class MessageUtil {
  private String message;

  public MessageUtil(String message) {
    this.message = message;
  }

  public String printMessage() {
    System.out.println(message);
    return message;
  }

  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
}
```

}			
}			

このクラスファイルは、渡された文字列を出力として作成します。MessageUtil コ ンストラクタは、文字列を設定します。printMessage メソッドは出力を作成しま す。salutationMessage メソッドが Hi! を出力した後に文字列が続きます。

 このファイルを作成し、TestMessageUtil.java という名前を付けて、/src/test/java ディレクトリに保存します。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;
public class TestMessageUtil {
 String message = "Robert";
 MessageUtil messageUtil = new MessageUtil(message);
 @Test
 public void testPrintMessage() {
   System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
 }
 @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
   message = "Hi!" + "Robert";
   assertEquals(message,messageUtil.salutationMessage());
 }
}
```

このクラスファイルは message クラスの MessageUtil 変数を Robert に設定します。そ の後、文字列 message および Robert が出力に表示されているかどうかを調べることによっ て、Hi!Robert 変数が正常に設定されたかどうかを調べます。

4. このファイルを作成し、pom.xml という名前を付けて、ルート (最上位) ディレクトリに保存します。

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven では、このファイルの指示に従って、MessageUtil.java および TestMessageUtil.java ファイルを messageUtil-1.0.jar という名前のファイルに変換 し、指定されたテストを実行します。

この時点で、ディレクトリ構造は次のようになります。

ステップ 2: buildspec ファイルを作成する

(前のステップ: ステップ 1: ソースコードを作成する)

このステップでは、ビルド仕様ファイルを作成します。ビルド環境は、CodeBuild がビルドを実行す るために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み 合わせを表します。CodeBuild では、ビルド仕様がないと、ビルド入力をビルド出力に正常に変換で きません。また、ビルド環境でビルド出力アーティファクトを特定して出力バケットにアップロード することもできません。

このファイルを作成し、buildspec.yml という名前を付けて、ルート (最上位) ディレクトリに保 存します。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

A Important

ビルド仕様宣言は有効な YAML である必要があるため、ビルド仕様宣言のスペースは重要で す。ビルド仕様宣言のスペース数が YAML と一致しない場合、ビルドは即座に失敗する場合 があります。YAML validator を使用して、ビルド仕様宣言が有効な YAML かどうかをテスト できます。

Note

ソースコードにビルド仕様ファイルを含める代わりに、ビルドプロジェクトを作成するとき に個別にビルドコマンドを宣言することができます。これは、毎回ソースコードのリポジト リを更新せずに、異なるビルドコマンドでソースコードをビルドする場合に役立ちます。詳 細については、「buildspec の構文」を参照してください。

このビルド仕様宣言の詳細は次の通りです。

- version は、使用されているビルド仕様スタンダードのバージョンを表します。このビルド仕様 宣言では、最新バージョン 0.2 が使用されます。
- phases は、CodeBuild にコマンドの実行を指示するビルドフェーズを表します。これらのビルドフェーズは install、pre_build、build、post_build として、ここにリストされています。これらのビルドフェーズ名のスペルを変更することはできず、追加のビルドフェーズ名を作成することもできません。

この例では、build フェーズ中に、CodeBuild によって mvn install コマンドが実行されま す。このコマンドは、コンパイルされた Java クラスファイルをビルド出力アーティファクトにコ ンパイル、テスト、パッケージ化するように Apache Maven に指示します。完全にするために、 この例では、いくつかの echo コマンドが各ビルドフェーズに配置されています。このチュートリ アルの後半で詳細なビルド情報を見る際に、これらの echo コマンドの出力は、CodeBuild がコマ ンドを実行する方法とその順序を理解するのに役立ちます。(この例にはすべてのビルドフェーズ が含まれていますが、コマンドを実行しないビルドフェーズは含める必要がありません)。各ビル ドフェーズについて、CodeBuild は最初から最後まで、各コマンドを一度に1つずつ、指定された 順序で実行します。

「artifacts」は、CodeBuild が出力バケットにアップロードする一連のビルド出力アーティファクトを表します。「files」は、ビルド出力に含めるファイルを表します。CodeBuild は、ビルド環境の「messageUtil-1.0.jar」相対ディレクトリにある、単一の「target」ファイルをアップロードします。ファイル名 messageUtil-1.0.jar およびディレクトリ名 target は、この例でのみ Apache Maven がビルド出力アーティファクトを作成して格納する方法に基づいています。独自のビルドでは、これらのファイル名とディレクトリは異なります。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
  |-- pom.xml
  |-- buildspec.yml
  `-- src
        |-- main
        | `-- java
        | `-- MessageUtil.java
        `-- test
        `-- java
        `-- test
        `-- java
        `-- TestMessageUtil.java
```

ステップ 3:2 つの S3 バケットを作成する

(前のステップ: ステップ 2: buildspec ファイルを作成する)

このチュートリアル用として 1 つのバケットを使用することもできますが、2 つのバケットを使用す る方が、ビルド入力の送信元およびビルド出力の送信先の確認が簡単になります。

- これらのバケットのいずれか(入力バケット)にビルド入力が保存されます。このチュートリアルでは、この入力バケットの名前はです。codebuild-*region-ID-account-ID*-input-bucketregion-IDはバケットのAWSリージョン、account-IDはAWSアカウントIDです。
- もう1つのバケット(出力バケット)にはビルド出力が保存されます。このチュートリアルでは、この出力バケットの名前は codebuild-region-ID-account-ID-output-bucket です。

これらのバケットに別の名前を選択した場合は、このチュートリアル全体で、その名前を使用してく ださい。

これら 2 つのバケットは、ビルドと同じ AWS リージョンに存在する必要があります。たとえば、米 国東部 (オハイオ) リージョンでビルドを実行するように CodeBuild に指示している場合、バケット は米国東部 (オハイオ) リージョンにある必要があります。

詳細については、Amazon Simple Storage Service コンソールユーザーガイドの<u>「バケットの作</u> 成」を参照してください。 Note

CodeBuild では、CodeCommit、GitHub、および Bitbucket の各リポジトリに保存されている ビルド入力もサポートされますが、このチュートリアルではこれらの使用方法については説 明しません。詳細については、「<u>ビルドを計画する</u>」を参照してください。

ステップ 4: ソースコードと buildspec ファイルをアップロードする

(前のステップ: ステップ 3:2 つの S3 バケットを作成する)

このステップでは、入力バケットにソースコードとビルド仕様ファイルを追加します。

オペレーティングシステムの zip ユーティリティを使用し

て、MessageUtil.zip、MessageUtil.java、TestMessageUtil.java、および pom.xml を 含む buildspec.yml という名前のファイルを作成します。

MessageUtil.zip ファイルのディレクトリ構造は、次のようになっている必要があります。

```
MessageUtil.zip
   |-- pom.xml
   |-- buildspec.yml
   `-- src
        |-- main
        |    `-- java
        |    `-- fava
        `-- test
        `-- java
        `-- test
        `-- java
        `-- TestMessageUtil.java
```

A Important

<mark>(root directory name)</mark>ディレクトリを含めないでください。<mark>(root directory</mark> name)ディレクトリ内のディレクトリとファイルのみを含めます。

MessageUtil.zip ファイルを codebuild-*region-ID-account-ID*-input-bucket という名 前の入力バケットにアップロードします。

▲ Important

CodeCommit、GitHub、および Bitbucket の各リポジトリでは、規約に従っ て、buildspec.yml というビルド仕様ファイルを各リポジトリのルート (最上位) に保存す るか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要があります。リポ ジトリのソースコードとビルド仕様ファイルを含む ZIP ファイルを作成しないでください。 S3 バケットに保存されたビルド入力に限り、ソースコードおよび規約に基づく

buildspec.yml というビルド仕様ファイルを圧縮した ZIP ファイルをルート(最上位) に作成するか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要がありま す。

ビルド仕様ファイルに別の名前を使用するか、ルート以外の場所でビルド仕様を参照する場合は、ビルドプロジェクト定義の一部としてビルド仕様の上書きを指定できます。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。

ステップ 5: ビルドプロジェクトを作成する

(前のステップ: ステップ 4: ソースコードと buildspec ファイルをアップロードする)

このステップでは、AWS CodeBuild を使用してビルドを実行するビルドプロジェクトを作成しま す。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソー スコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれま す。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プロ グラミング言語ランタイム、およびツールの組み合わせを表します。ビルド環境は Docker イメージ として表されます。詳細については、Docker Docs ウェブサイトの <u>Docker overview</u> を参照してくだ さい。

このビルド環境では、CodeBuild に、Java 開発キット (JDK) のバージョンおよび Apache Maven が 含まれる Docker イメージを使用するように指示します。

ビルドプロジェクトを作成するには

1. を使用して create-project コマンド AWS CLI を実行します。

aws codebuild create-project --generate-cli-skeleton

JSON 形式のデータが出力に表示されます。 AWS CLI がインストールされているローカルコン ピュータまたはインスタンス上の場所create-project.jsonにある という名前のファイルに データをコピーします。別のファイル名を使用する場合は、このチュートリアル全体でそのファ イル名を使用してください。

コピーされたデータをこの形式に従って変更して、結果を保存します。

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

serviceIAMRole を、CodeBuild サービスロールの Amazon リソースネーム (ARN) (例: arn:aws:iam::*account-ID*:role/*role-name*) に置き換えます。サービスロールを作成す る場合は、「<u>CodeBuild が他の AWS のサービスとやり取りすることを許可</u>」を参照してくださ い。

このデータの各要素は以下のとおりです。

- name は、このビルドプロジェクト (この例では codebuild-demo-project) に必要な識別 子を表します。ビルドプロジェクト名は、アカウント内のすべてのビルドプロジェクト間で一 意である必要があります。
- source の場合、type はソースコードのリポジトリのタイプを表す必須の値です (この例では、Amazon S3 バケットの場合は S3)。
- source の場合、location は、ソースコードへのパスを表します (この例では、入力バケット名の後に ZIP ファイル名が続きます)。
- artifactsの場合、typeは、ビルド出力アーティファクトのリポジトリのタイプを表す必須の値です (この例では、Amazon S3 バケットの場合は S3)。

- artifacts の場合、location は、以前に作成または識別した出力バケットの名前を表します(この例では、codebuild-*region-ID-account-ID*-output-bucket)。
- environment を使用する場合、type はビルド環境のタイプを表す必須の値です (この例で は LINUX_CONTAINER)。
- environmentの場合、imageは、Dockerイメージリポジトリタイプで指定された、この ビルドプロジェクトが使用する Docker イメージ名とタグの組み合わせを表す必須の値です (この例では、aws/codebuild/standard:5.0 Docker イメージリポジトリ内の Docker イ メージの場合は aws/codebuild/standard)。CodeBuild は、Docker イメージの名前で す。5.0 は、Docker イメージのタグです。

シナリオで使用できる Docker イメージをさらに見つけるには、「<u>ビルド環境に関するリファ</u> レンス」を参照してください。

environmentの場合、computeTypeは、CodeBuildが使用するコンピューティングリソース (この例では BUILD_GENERAL1_SMALL)を表す必須の値です。

Note

description、buildspec、auth (type と resource を含 む)、path、namespaceType、name (artifacts)、packaging、environmentVariables (name と value を含 む)、timeoutInMinutes、encryptionKey、tags (key と value を含む) などの 元の JSON 形式のデータで使用可能なその他の値はオプションです。これらは、この チュートリアルで使用されていないため、ここには示されていません。詳細について は、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

 保存したばかりのファイルがあるディレクトリに移動してから、create-project コマンドをもう 一度実行します。

aws codebuild create-project --cli-input-json file://create-project.json

成功した場合、次のようなデータが出力に表示されます。

```
{
    "project": {
        "name": "codebuild-demo-project",
        "serviceRole": "serviceIAMRole",
        "tags": [],
```

```
"artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"
 }
}
```

- project は、このビルドプロジェクトに関する情報を表します。
 - tags は、宣言されたタグを表します。
 - packaging は、ビルド出力アーティファクトが出力バケットに保存される方法を表します。NONE は、出力バケット内にフォルダが作成されることを意味します。ビルド出力アー ティファクトはそのフォルダ内に格納されます。
 - lastModified は、ビルドプロジェクトに関する情報が最後に変更された時刻 (Unix の時間形式)を表します。
 - timeoutInMinutes は、ビルドが完了していない場合、CodeBuild がビルドを停止するまでの時間 (分) を表します。(デフォルトは 60 分です。)
 - created は、ビルドプロジェクトが作成された時刻 (Unix の時間形式) を表します。
 - environmentVariables は、CodeBuild がビルド中に使用するために宣言されて、使用 可能な環境変数を表します。
 - encryptionKey は、CodeBuild がビルド出力アーティファクトの暗号化に使用したカスタ マー管理キーの ARN を表します。

・ arn は、ビルドプロジェクトの ARN を表します。

Note

create-project コマンドの実行後に、次のようなメッセージが出力される場合 があります。「ユーザー: *user-ARN* は次のことを実行する権限がありません:

codebuild:CreateProject」 これは、CodeBuild を使用してビルドプロジェクトを作成する ための十分なアクセス許可を持たない ユーザーの認証情報 AWS CLI で を設定したことが原 因であると考えられます。これを修正するには、次の IAM エンティティのいずれかに属する 認証情報を使用して AWS CLI を設定します。

- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- AWS アカウントのユーザーでAWSCodeBuildAdminAccess、そのユーザーまたは ユーザーが属する IAM グループにアタッチされた AmazonS3ReadOnlyAccess、、 IAMFullAccess管理ポリシーを持つユーザー。これらのアクセス許可を持つ ユーザーま たはグループが AWS アカウントになく、これらのアクセス許可をユーザーまたはグルー プに追加できない場合は、AWS アカウント管理者にお問い合わせください。詳細につい ては、「AWS の 管理 (事前定義) ポリシー AWS CodeBuild」を参照してください。

ステップ 6: ビルドを実行する

(前のステップ: ステップ 5: ビルドプロジェクトを作成する)

このステップでは、ビルドプロジェクトの設定を使用してビルドを実行する AWS CodeBuild ように に指示します。

ビルドを実行するには

1. を使用して start-build コマンド AWS CLI を実行します。

aws codebuild start-build --project-name project-name

project-name を、前の手順のビルドプロジェクト名 (例: codebuild-demo-project) に置 き換えます。

2. 成功すると、次のようなデータが出力に表示されます。

```
{
  "build": {
    "buildComplete": false,
    "initiator": "user-name",
    "artifacts": {
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip"
    },
    "projectName": "codebuild-demo-project",
    "timeoutInMinutes": 60,
    "buildStatus": "IN_PROGRESS",
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "currentPhase": "SUBMITTED",
    "startTime": 1472848787.882,
    "id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",
    "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"
 }
}
```

• build は、このビルドに関する情報を表します。

- buildComplete は、ビルドの完了 () を表します。trueそうでない場合は、false です。
- initiator は、ビルドを開始したエンティティを表します。
- artifacts は、場所を含む、ビルド出力に関する情報を表します。
- projectName は、ビルドプロジェクトの名前を表します。
- buildStatus は、start-build コマンドが実行されたときの現在のビルドのステータスを表します。
- currentPhase は、start-build コマンドが実行されたときの現在のビルドフェーズを表します。
- ・ startTime は、ビルドプロセスが開始された時刻 (Unix の時間形式)を表します。

- id は、ビルドの ID を表します。
- arn は、ビルドの ARN を表します。

[id] の値を書き留めておきます。それは次の手順で必要となります。

ステップ 7: ビルド情報の要約を表示する

(前のステップ: ステップ 6: ビルドを実行する)

このステップでは、ビルドのステータスに関する要約情報を表示します。

要約されたビルド情報を表示するには

• AWS CLI を使用して batch-get-builds コマンドを実行します。

aws codebuild batch-get-builds --ids id

id を、前のステップの出力に表示された id 値に置き換えます。

成功した場合、次のようなデータが出力に表示されます。

```
{
  "buildsNotFound": [],
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1472848788.525,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
          "startTime": 1472848787.882
        },
        ... The full list of build phases has been omitted for brevity ...
        {
          "phaseType": "COMPLETED",
          "startTime": 1472848878.079
        }
      ],
```

```
"logs": {
        "groupName": "/aws/codebuild/codebuild-demo-project",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=region-
ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-
bef1-d52bfEXAMPLE",
        "streamName": "38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
      },
      "artifacts": {
        "md5sum": "MD5-hash",
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip",
        "sha256sum": "SHA-256-hash"
      },
      "projectName": "codebuild-demo-project",
      "timeoutInMinutes": 60,
      "initiator": "user-name",
      "buildStatus": "SUCCEEDED",
      "environment": {
        "computeType": "BUILD_GENERAL1_SMALL",
        "image": "aws/codebuild/standard:5.0",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
      },
      "currentPhase": "COMPLETED",
      "startTime": 1472848787.882,
      "endTime": 1472848878.079,
      "id": "codebuild-demo-project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
      "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
    }
  ]
}
```

- buildsNotFound は、情報が利用できないビルドのビルド ID を表します。この例では、空である必要があります。
- buildsは、利用可能な各ビルドに関する情報を表します。この例では、出力に1つのビルドのみに関する情報が表示されます。

 phases は、CodeBuild がビルドプロセス中に実行する一連のビルドフェーズを表しま す。各ビルドフェーズに関する情報が startTime、endTime、durationInSeconds (フェーズの開始時と終了時、Unix 時間形式で表示、持続時間(秒))、phaseType (SUBMITTED、PROVISIONING、DOWNLOAD_SOURCE、INSTALL、PRE_BUILD、BUILD、POST_E など)、phaseStatus
 (SUCCEEDED、FAILED、FAULT、TIMED_OUT、IN_PROGRESS、STOPPED など) として個 別にリストされます。batch-get-builds コマンドを初めて実行するときは、多くの(または まったく)フェーズが存在しない可能性があります。同じビルド ID を持つ batch-get-builds コマンドを続けて実行すると、より多くのビルドフェーズが出力に表示されます。

- logs は、Amazon CloudWatch Logs のビルドのログに関する情報を表します。
- md5sum および sha256sum は、ビルドの出力アーティファクトの MD5 と SHA-256 ハッシュを表します。これらは、関連するビルドプロジェクトの packaging 値が ZIP に設定されている場合にのみ出力に表示されます。(このチュートリアルでは設定していません。)これらのハッシュとチェックサムツールを使用して、ファイルの完全性と信頼性を確認することができます。

Note

Amazon S3 コンソールを使用して、これらのハッシュを表示することもできます。 ビルド出力アーティファクトの横にあるボックスを選択し、[アクション]、[プロパ ティ] の順に選択します。[Properties] ペインで [Metadata] を展開し、[x-amz-metacodebuild-content-md5] と [x-amz-meta-codebuild-content-sha256] の値を確認し ます。(Amazon S3 コンソールでは、ビルド出力アーティファクトの [ETag] 値を MD5 または SHA-256 ハッシュのいずれかに解釈することはできません。) AWS SDKs を使用してこれらのハッシュを取得する場合、値は codebuildcontent-md5と という名前になりますcodebuild-content-sha256。

・ endTime は、ビルドプロセスが終了した時刻 (Unix の時間形式) を表します。

Note

Amazon S3 メタデータには、x-amz-meta-codebuild-buildarn という名前の CodeBuild ヘッダーがあります。このヘッダーには、Amazon S3 にアーティファクトを 公開する CodeBuild ビルドの buildArn が含まれています。通知のソーストラッキン グを許可し、アーティファクトの生成元であるビルドを参照するために buildArn を追 加します。

ステップ 8: 詳細なビルド情報を表示する

(前のステップ: ステップ 7: ビルド情報の要約を表示する)

このステップでは、CloudWatch Logs のビルドに関する詳細情報を表示します。

- Note 機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。
 AWS アクセスキー IDs。詳細については、AWS Identity and Access Management ユー ザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
 - パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「<u>Systems Manager パラメータストア</u>」および 「<u>Systems Manager パラメータストアコンソールのチュートリアル</u>」を参照してください。
 - を使用して指定された文字列 AWS Secrets Manager。詳細については、「<u>キー管理</u>」を参照してください。

詳細なビルド情報を表示するには

- ウェブブラウザを使用して、前の手順の出力に表示された deepLink の場所に 移動します(例: https://console.aws.amazon.com/cloudwatch/home? region=*region-ID*#logEvent:group=/aws/codebuild/codebuild-demoproject;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE)。
- CloudWatch Logs ログストリームでは、ログイベントを参照できます。デフォルトでは、ログ イベントの最後のセットだけが表示されます。以前のログイベントを表示するには、リストの先 頭にスクロールします。
- このチュートリアルでは、ほとんどのログイベントに、CodeBuild でビルド依存ファイルをビル ド環境にダウンロードおよびインストールする操作に関する詳細情報が含まれますが、これらの 情報は不要な場合があります。[Filter events] ボックスを使用すると、表示する情報量を減らす ことができます。例えば、[Filter events] (イベントのフィルター) で「"[INF0]"」と入力した

場合、「[INF0]」を含むイベントのみが表示されます。詳細については、Amazon CloudWatch ユーザーガイドの「フィルターとパターンの構文」を参照してください。

CloudWatch Logs のログストリームのうち、このチュートリアルに関係する部分を以下に示します。

```
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INF0]
[Container] 2016/04/15 17:49:44 [INF0]
                                _____
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample App 1.0
[Container] 2016/04/15 17:49:44 [INF0]
_____
[Container] 2016/04/15 17:49:55
   -----
[Container] 2016/04/15 17:49:55 T E S T S
[Container] 2016/04/15 17:49:55
_____
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[Container] 2016/04/15 17:49:56 [INF0]
```

```
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INF0]
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INF0] Finished at: 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INF0]
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

この例では、CodeBuild はビルド前、ビルド、およびビルド後のビルドフェーズを正常に完了しました。ユニットテストを実行し、messageUtil-1.0.jar ファイルは正常に構築されました。

ステップ 9: ビルド出力アーティファクトを取得する

(前のステップ: ステップ 8: 詳細なビルド情報を表示する)

このステップでは、CodeBuild が構築して出力バケットにアップロードした 「messageUtil-1.0.jar」ファイルを取得します。

このステップを完了するには、CodeBuild コンソールまたは Amazon S3 コンソールを使用します。

ビルド出力アーティファクトを取得するには (AWS CodeBuild コンソール)

CodeBuild コンソールが開いていて、ビルドの詳細ページが前のステップから引き続き表示されている状態で、[Build details] を選択して [Artifacts] セクションまでスクロールします。

Note

ビルドの詳細ページが表示されていない場合は、ナビゲーションバーで [Build history] (ビルド履歴)、[Build run] (ビルドの実行) リンクの順に選択します。

ステップ 9: ビルド出力アーティファクトを取得する

 Amazon S3 フォルダへのリンクは、[Artifacts upload location] (アーティファクトのアップロー ド場所)の下にあります。Amazon S3 内のフォルダが開き、「messageUtil-1.0.jar」とい う名前のビルド出力アーティファクトファイルを見つけます。

ビルド出力アーティファクトを取得するには (Amazon S3 コンソール)

- 1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
- 2. codebuild-*region-ID-account-ID*-output-bucket を開きます。
- 3. codebuild-demo-project フォルダを開きます。
- target という名前のフォルダを開き、messageUtil-1.0.jar という名前のビルド出力アー ティファクトファイルを見つけます。

ステップ 10: S3 入力バケットを削除する

(前のステップ: <u>ステップ 9: ビルド出力アーティファクトを</u>取得する)

AWS アカウントへの継続的な課金を防ぐために、このチュートリアルで使用されている入出力バ ケットを削除できます。手順については、Amazon Simple Storage Service ユーザーガイドで<u>バケッ</u> トを削除、または空にする方法を参照してください。

IAM ユーザー を使用している場合、このバケットを削除するユーザーまたは管理者 IAM ユーザー には、さらに高いアクセス権限が必要です。マーカー間で次のステートメント (###BEGIN ADDING STATEMENT HERE### と ###END ADDING STATEMENTS HERE###)を既存のアクセスポリシーに 追加します。

このステートメントでは、簡潔にするために省略記号 (...) が使用されています。既存のアクセスポ リシーのステートメントは削除しないでください。これらの省略記号はポリシーに入力しないでくだ さい。

```
{
    "Version": "2012-10-17",
    "Id": "...",
    "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
        "Effect": "Allow",
        "Action": [
            "s3:DeleteBucket",
    }
}
```

```
"s3:DeleteObject"
],
"Resource": "*"
}
### END ADDING STATEMENT HERE ###
]
}
```

まとめ

このチュートリアルでは、 AWS CodeBuild を使用して一連の Java クラスファイルを JAR ファイル に構築しました。次に、ビルドの結果を表示しました。

これで、独自のシナリオに CodeBuild を使用できます。「<u>ビルドを計画する</u>」の手順に従ってくだ さい。もう少し準備が必要な場合は、用意されているサンプルでビルドを試すことができます。詳細 については、「CodeBuild のユースケースベースのサンプル」を参照してください。

CodeBuild のユースケースベースのサンプル

これらのユースケースベースのサンプルを使用して、以下を試すことができます AWS CodeBuild。

クロスサービス例

実験するクロスサービスサンプルのリスト AWS CodeBuild。

<u>ビルドバッジサンプル</u>

ビルドバッジを使用して CodeBuild を設定する方法を示します。

<u>テストレポートサンプル</u>

を使用して、テストレポート AWS CLI の作成、実行、結果の表示を行います。 CodeBuild の Docker サンプル

カスタム Docker イメージの使用、Amazon ECR のリポジトリへの Docker イメージの公開、プ ライベートレジストリでの Docker イメージの使用方法について説明します。

<u>S3 バケットでのビルド出力のホスティング</u>

暗号化されていないビルドアーティファクトを使用して S3 バケットに静的なウェブサイトを作 成する方法を示します。

複数の入出力のサンプル

ビルドプロジェクトで複数の入力ソースと複数の出力アーティファクトを使用する方法を示しま す。

並列テスト実行サンプル

codebuild-tests-run CLI コマンドを使用して、並列実行環境間でテストを分割して実行す る方法を示します。

buildspec ファイルサンプルのランタイムバージョン

buildspec ファイルでランタイムとバージョンを指定する方法を示します。

ソースバージョンのサンプル

ソースの特定のバージョンを CodeBuild ビルドプロジェクトで使用する方法を示します。 CodeBuild のサードパーティーソースリポジトリのサンプル

CodeBuild を使用して、ウェブフックで BitBucket、GitHub Enterprise Server、GitHub プルリク エストを作成する方法について説明します。

セマンティックバージョニングを使用してビルド時にアーティファクト名を設定

セマンティックバージョニングを使用して、ビルド時にアーティファクト名を作成する方法を示 します。

CodeBuild のクロスサービス例

これらのクロスサービスサンプルを使用して、以下を試すことができます AWS CodeBuild。

Amazon ECR のサンプル

Amazon ECR リポジトリの Docker イメージを使用して、Apache Maven を使用して単一の JAR ファイルを生成します。サンプル手順では、Docker イメージを作成して Amazon ECR にプッシュし、Go プロジェクトを作成し、プロジェクトをビルドし、プロジェクトを実行 し、CodeBuild が Amazon ECR に接続できるようにアクセス許可を設定する方法を示します。 Amazon EFS のサンプル

CodeBuild プロジェクトが Amazon EFS ファイルシステムをマウントしてビルドするよう に buildspec ファイルを設定する方法を示します。サンプル手順では、Amazon VPC を作成 し、Amazon VPC でファイルシステムを作成し、Amazon VPC を使用するプロジェクトを作成 してビルドし、生成されたプロジェクトファイルと変数を確認する方法について説明します。

AWS CodePipeline サンプル

AWS CodePipeline を使用して、バッチビルド、複数の入力ソース、複数の出力アーティファクトを含むビルドを作成する方法を示します。このセクションには、個別のアーティファクトと、 結合アーティファクトでバッチビルドを作成するパイプライン構造を示すサンプル JSON ファイルが含まれています。複数の入力ソースと複数の出力アーティファクトを含むパイプライン構造 を示す追加の JSON サンプルが提供されます。

AWS Config サンプル

のセットアップ方法を示します AWS Config。追跡される CodeBuild リソースを一覧表示 し、CodeBuild プロジェクトを検索する方法について説明します AWS Config。サンプル手順で は、と統合するための前提条件 AWS Config、セットアップする手順 AWS Config、CodeBuild プロジェクトとデータを検索する手順を示します AWS Config。

ビルド通知サンプル

Apache Maven を使用して単一の JAR ファイルを生成します。Amazon SNS トピックのサブ スクライバーにビルド通知を送信します。サンプル手順では、CodeBuild が Amazon SNS およ び CloudWatch と通信できるようにアクセス許可を設定する方法、Amazon SNS で CodeBuild トピックを作成および識別する方法、トピックに受信者をサブスクライブする方法、および CloudWatch でルールを設定する方法を示します。

CodeBuild の Amazon ECR サンプル

このサンプルでは Amazon Elastic Container Registry (Amazon ECR) イメージレポジトリの Docker イメージを使用して、サンプルの Go プロジェクトをビルドします。

A Important

このサンプルを実行すると、AWS アカウントに料金が発生する可能性があります。こ れには、Amazon S3、、AWS KMS CloudWatch Logs、Amazon ECR に関連する AWS リソースとアクション AWS CodeBuild に対する および の料金が含まれます。詳細につ いては、<u>CodeBuild 料金表、Amazon S3 料金表</u>、<u>AWS Key Management Service 料金</u> <u>表、Amazon CloudWatch 料金表、Amazon Elastic Container Registry 料金表</u>を参照してくだ さい。

トピック

Amazon ECR サンプルを実行

Amazon ECR サンプルを実行

CodeBuild の Amazon ECR サンプルを実行するには、以下の手順に従います。

このサンプルを実行するには

- Amazon ECR で Docker イメージを作成してイメージリポジトリにプッシュするには、 「<u>'Docker イメージを Amazon ECR に公開' サンプル</u>」の「<u>'Docker イメージを Amazon ECR に</u> 公開' サンプルを実行」セクションにある手順を完了します。
- 2. Go プロジェクトの作成:
 - a. このトピックの <u>Go プロジェクトの構造</u>および <u>Go プロジェクトのファイル</u>セク ションで説明されているようにファイルを作成し、S3 入力バケット、または AWS CodeCommit、GitHub、または Bitbucket リポジトリにアップロードします。

▲ Important

(root directory name)をアップロードしないでください。アップロードする のは、(root directory name)内のファイルのみです。 S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮して から入力バケットにアップロードしてください。(root directory name)を ZIP ファイルに追加しないでください。追加するのは、(root directory name) 内のファイルのみです。

b. ビルドプロジェクトを作成して、ビルドを実行し、関連するビルド情報を表示します。

を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-go-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

- c. ビルド出力アーティファクトを取得するには、S3 出力バケットを開きます。
- d. *GoOutputArtifact*.zip ファイルをローカルコンピュータまたはインスタンスへダウン ロードし、ファイルの内容を抽出します。展開したコンテンツから、hello ファイルを取 得します。

- 次のいずれかが true の場合、 が Docker イメージをビルド環境に AWS CodeBuild プルできるように、Amazon ECR のイメージリポジトリにアクセス許可を追加する必要があります。
 - プロジェクトで CodeBuild の認証情報を使用して Amazon ECR のイメージをプルしている場合。これは、CODEBUILD の imagePullCredentialsType 属性で ProjectEnvironment の値で示されます。
 - プロジェクトでクロスアカウントの Amazon ECR イメージを使用している場合。この場合は、プロジェクトでサービスロールを使用して Amazon ECR イメージをプルする必要があります。この動作を有効にするには、imagePullCredentialsType の ProjectEnvironment 属性を SERVICE_ROLE に設定します。
 - 1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
 - 2. リポジトリ名のリストで、作成または選択したリポジトリの名前を選択します。
 - 3. ナビゲーションペインで、[アクセス許可]、[編集]、[ステートメントを追加] の順に選択しま す。
 - 4. [ステートメント名] で、識別子 (CodeBuildAccess など) を入力します。
 - 5. [効果] で、[許可] を選択したままにしておきます。これにより、別の AWS アカウントへのア クセスを許可します。
 - 6. [プリンシパル] で、次のいずれかを実行します。
 - プロジェクトで CodeBuild の認証情報を使用して Amazon ECR のイメージをプルする場合
 は、[サービスプリンシパル] に「codebuild.amazonaws.com」と入力します。
 - プロジェクトでクロスアカウントの Amazon ECR イメージを使用する場合は、[AWS アカウント ID] に、アクセス権を付与する AWS アカウントの ID を入力します。
 - 7. [すべての IAM エンティティ] リストをスキップします。
 - 8. [アクション] で、プル専用アクションとして [ecr:GetDownloadUrlForLayer]、 [ecr:BatchGetImage]、および [ecr:BatchCheckLayerAvailability] を選択します。
 - 9. [条件] で、以下を追加します。

```
{
    "StringEquals":{
        "aws:SourceAccount":"<AWS-account-ID>",
        "aws:SourceArn":"arn:aws:codebuild:<region>:<AWS-account-
ID>:project/<project-name>"
    }
}
```

10[Save] を選択します。

このポリシーは [アクセス許可] に表示されます。プリンシパルは、この手順のステップ 3 で [プリンシパル] に入力した値です。

- プロジェクトで CodeBuild の認証情報を使用して Amazon ECR のイメージをプルする場合 は、[Service principals] (サービスプリンシパル) に「"codebuild.amazonaws.com"」と 入力します。
- プロジェクトでクロスアカウントの Amazon ECR イメージを使用している場合、アクセス を許可する AWS アカウントの ID がAWS アカウント IDsの下に表示されます。

次のサンプルポリシーでは、CodeBuild 認証情報とクロスアカウント Amazon ECR イメージの両方を使用します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccessPrincipal",
         "Effect":"Allow",
         "Principal":{
            "Service": "codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
         ],
         "Condition":{
             "StringEquals":{
                "aws:SourceArn":"arn:aws:codebuild:<region>:<aws-account-
id>:project/<project-name>",
                "aws:SourceAccount":"<aws-account-id>"
            }
         }
      },
      {
         "Sid": "CodeBuildAccessCrossAccount",
         "Effect":"Allow",
         "Principal":{
             "AWS":"arn:aws:iam::<<u>AWS-account-ID</u>>:root"
         },
```

```
"Action":[
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage",
    "ecr:BatchCheckLayerAvailability"
    ]
    }
]
```

 プロジェクトで CodeBuild 認証情報を使用し、CodeBuild プロジェクトに Amazon ECR リ ポジトリへのオープンアクセスを許可する場合は、Condition キーを省略し、次のサンプ ルポリシーを追加できます。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid": "CodeBuildAccessPrincipal",
      "Effect":"Allow",
      "Principal":{
        "Service": "codebuild.amazonaws.com"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid": "CodeBuildAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::<AWS-account-ID>:root"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
}
```

4. ビルドプロジェクトを作成して、ビルドを実行し、ビルド情報を表示します。

を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "amazon-ecr-sample-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "account-ID.dkr.ecr.region-ID.amazonaws.com/your-Amazon-ECR-repo-
name:tag",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

- 5. ビルド出力アーティファクトを取得するには、S3 出力バケットを開きます。
- GoOutputArtifact.zip ファイルをローカルコンピュータまたはインスタンスへダウン ロードし、GoOutputArtifact.zip ファイルの内容を抽出します。展開したコンテンツか ら、hello ファイルを取得します。

Go プロジェクトの構造

このサンプルのディレクトリ構造は次のとおりとします。

```
(root directory name)
### buildspec.yml
### hello.go
```

Go プロジェクトのファイル

このサンプルで使用するファイルは以下のとおりです。

buildspec.yml(内)(root directory name)

```
version: 0.2
phases:
  install:
   runtime-versions:
     golang: 1.13
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```

hello.go(内)(root directory name)

```
package main
import "fmt"
func main() {
  fmt.Println("hello world")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
}
```

の Amazon Elastic File System サンプル AWS CodeBuild

Amazon EC2 インスタンス用のスケーラブルな共有ファイルサービスである Amazon Amazon Elastic File System で AWS CodeBuild ビルドを作成することもできます。Amazon EFS のストレー ジ容量は伸縮自在なため、ファイルの追加および削除に合わせて拡大または縮小されます。また、 ファイルシステムを作成、設定するために使用できるシンプルなウェブサービスインターフェイス を提供します。さらに、ファイルストレージインフラストラクチャも自動的に管理されるため、ファ イルシステム設定のデプロイ、パッチ適用、保守について心配する必要がありません。詳細について は、Amazon Elastic File System ユーザーガイドの「<u>Amazon Elastic File System とは</u>」を参照して ください。

このサンプルでは、Java アプリケーションが Amazon EFS ファイルシステムにマウントされて構 築されるように CodeBuild プロジェクトを設定する方法を示します。開始する前に、S3 入力バケッ ト、または AWS CodeCommit、GitHub、GitHub Enterprise Server、Bitbucket リポジトリにアップ ロードされる Java アプリケーションを構築できる状態になっている必要があります。

ファイルシステムの転送中のデータは暗号化されます。別のイメージを使用して転送中のデータを暗 号化するには、「転送中のデータの暗号化」を参照してください。

トピック

- Amazon Elastic File System AWS CodeBuild で を使用する
- Amazon EFS 統合のトラブルシューティング

Amazon Elastic File System AWS CodeBuild で を使用する

このサンプルでは、Amazon EFS を で使用するために必要な 4 つの大まかなステップについて説明 します AWS CodeBuild。具体的には次の 2 つです。

- 1. AWS アカウントに Virtual Private Cloud (VPC) を作成します。
- 2. この VPC を使用するファイルシステムを作成します。
- VPC を使用する CodeBuild プロジェクトを作成および構築します。CodeBuild プロジェクトでは、以下を使用してファイルシステムが識別されます。
 - 一意のファイルシステム識別子。ビルドプロジェクトでファイルシステムを指定するときに識別子を選択します。
 - ファイルシステム ID。ID は、Amazon EFS コンソールでファイルシステムを開くと表示されます。
 - マウントポイント。ファイルシステムをマウントする Docker コンテナ内のディレクトリです。
 - マウントオプション。ファイルシステムのマウント方法に関する詳細が含まれます。

ビルドプロジェクトを確認して、正しいプロジェクトファイルと変数が生成されていることを確認します。

Note

Amazon EFS で作成されたファイルシステムは Linux プラットフォームでのみサポートされ ます。

トピック

- <u>ステップ 1</u>: を使用して VPC を作成する AWS CloudFormation
- ステップ 2: VPC を使用した Amazon Elastic File System ファイルシステムを作成
- ステップ 3: Amazon EFS で使用する CodeBuild プロジェクトを作成
- ステップ 4: ビルドプロジェクトを確認

ステップ 1: を使用して VPC を作成する AWS CloudFormation

AWS CloudFormation テンプレートを使用して VPC を作成します。

1. の手順に従って<u>AWS CloudFormation VPC テンプレート</u>、 AWS CloudFormation を使用して VPC を作成します。

この AWS CloudFormation テンプレートによって作成された VPC には、2 つのプライ ベートサブネットと 2 つのパブリックサブネットがあります。プライベートサブネット を使用するのは、Amazon EFS で作成したファイルシステムを、 AWS CodeBuild でマ ウントする場合のみです。いずれかのパブリックサブネットを使用する場合、ビルドに 失敗します。

- 2. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/vpc/</u>:// www.com」で Amazon VPC コンソールを開きます。
- 3. で作成した VPC を選択します AWS CloudFormation。
- [説明] タブに表示される VPC の名前と ID を書き留めます。どちらも、このサンプルの後半で AWS CodeBuild プロジェクトを作成するときに必要になります。

Note

ステップ 2: VPC を使用した Amazon Elastic File System ファイルシステムを作成

先ほど作成した VPC を使用して、このサンプルのシンプルな Amazon EFS ファイルシステムを作 成します。

- 1. にサインイン AWS Management Console し、Amazon EFS コンソールを<u>https://</u> console.aws.amazon.com/efs/://www..com で開きます。
- 2. [Create file system] を選択します。
- 3. [VPC] で、このサンプルの前のステップで書き留めた VPC 名を選択します。
- 4. サブネットに関連付けられているアベイラビリティーゾーンの選択したままにしておきます。
- 5. [Next Step] (次のステップ) をクリックします。
- 6. [Add tags] (タグの追加) のデフォルトの [Name] (名前) キーにある [Value] (値) に、Amazon EFS ファイルシステムの名前を入力します。
- デフォルトのパフォーマンスモードおよびスループットモードとして [General Purpose (汎用)] および [Bursting (バースト)] を選択したまま [Next Step (次のステップ)] を選択します。
- 8. [Configure client access (クライアントアクセスの設定)] で、[Next Step (次のステップ)] を選択 します。
- 9. [Create File System (ファイルシステムの作成)] を選択します。
- 10. (オプション) 転送時のデータ暗号化を適用するポリシーを、Amazon EFS ファイルシステムに 追加することをお勧めします。Amazon EFS コンソールで、[ファイルシステムポリシー]、[編 集]、[すべてのクライアントに転送中の暗号化を適用する] ボックス、[保存] の順に選択します。

ステップ 3: Amazon EFS で使用する CodeBuild プロジェクトを作成

このサンプルで前に作成した VPC を使用する AWS CodeBuild プロジェクトを作成します。ビルド を実行すると、先ほど作成した Amazon EFS ファイルシステムがマウントされます。次に、Java ア プリケーションによって作成された .jar ファイルがファイルシステムのマウントポイントディレクト リに保存されます。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. ナビゲーションペインで [ビルドプロジェクト] を選択し、次に [ビルドプロジェクトの作成] を 選択します。
- 3. [Project name (プロジェクト名)] にプロジェクトの名前を入力します。

- 4. [ソースプロバイダー] で、構築する Java アプリケーションが含まれているリポジトリを選択し ます。
- 5. CodeBuild がアプリケーションを見つけるために使用するリポジトリ URL などの情報を入力し ます。オプションはソースプロバイダーごとに異なります。詳細については、「<u>Choose source</u> <u>provider</u>」を参照してください。
- 6. [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
- 7. [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
- 8. [ランタイム] で、[Standard (標準)] を選択します。
- 9. Image から、aws/codebuild/amazonlinux-x86_64-standard:4.0 を選択します。
- 10. [環境タイプ] で、[Linux] を選択します。
- 11. [Service role (サービスロール)] で、[New service role (新しいサービスロール)] を選択します。 [Role name] (ロール名) に、CodeBuild により作成されたロールの名前を入力します。
- 12. [Additional configuration (追加設定)] を展開します。
- [Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Docker イメージを構築する場合、またはビルドで昇格された権限を取得する場合は、このフラ グを有効にする)] を選択します。

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビ ルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime</u> <u>Privilege and Linux Capabilities</u>」を参照して、特権モードを有効にします。ま た、Windows は特権モードをサポートしていません。

- 14. [VPC (VPC)] で、VPC ID を選択します。
- [サブネット] で、VPC に関連付けられているプライベートサブネットのうち 1 つ以上を選択し ます。Amazon EFS ファイルシステムをマウントするビルドでプライベートサブネットを使用 する必要があります。パブリックサブネットを使用している場合、ビルドに失敗します。
- 16. [Security groups (セキュリティグループ)] で、デフォルトのセキュリティグループを選択します。
- 17. [ファイルシステム] で、以下の情報を入力します。
 - (識別子)に、一意のファイルシステム識別子を入力します。識別子の長さは 129 文字未満である必要があります。英数字とアンダースコアのみを使用できます。一意のファイルシステム識別子。CodeBuild によって使用されて、伸縮自在なファイルシステムを識別する環境変数

が作成されます。環境変数の形式は大文字の CODEBUILD_*<file_system_identifier>* で す。たとえば、my_efs と入力すると、環境変数は CODEBUILD_MY_EFS になります。

- [ID] で、ファイルシステム ID を選択します。
- (オプション) ファイルシステムのディレクトリを入力します。CodeBuild はこのディレクトリ をマウントします。[ディレクトリパス] を空白のままにすると、CodeBuild はファイルシステ ム全体をマウントします。パスはファイルシステムのルートからの相対です。
- [マウントポイント]に、ファイルシステムをマウントするディレクトリの絶対パスを入力します。このディレクトリが存在しない場合は、CodeBuildによってビルド中に作成されます。
- (オプション) マウントオプションを入力します。[マウントオプション] を空白のままにする
 と、CodeBuild はデフォルトのマウントオプションを使用します。

```
nfsvers=4.1
rsize=1048576
wsize=1048576
hard
timeo=600
retrans=2
```

詳細については、Amazon Elastic File System ユーザーガイドの「<u>NFS の推奨されるマウント</u> オプション」を参照してください。

- 18. [ビルド仕様] で、[ビルドコマンドの挿入]、[Switch to editor (エディタに切り替え)] の順に選択し ます。
- エディタに次のビルド仕様コマンドを入力します。<file_system_identifier> をステップ
 17 で入力した識別子に置き換えます。大文字を使用します (CODEBUILD_MY_EFS など)。

```
version: 0.2
phases:
    install:
    runtime-versions:
        java: corretto11
    build:
        commands:
            - mvn compile -Dgpg.skip=true -Dmaven.repo.local=
$CODEBUILD_<file_system_identifier>
```

- 20. 他のすべての設定にはデフォルト値を使用し、[Create build project (ビルドプロジェクトの作成)] を選択します。ビルドが完了すると、プロジェクトのコンソールページが表示されます。
- 21. [Start build] を選択します。

ステップ 4: ビルドプロジェクトを確認

AWS CodeBuild プロジェクトの構築後:

- Java アプリケーションによって作成された .jar ファイルがあります。このファイルは Amazon EFS ファイルシステムのマウントポイントディレクトリにビルドされています。
- ファイルシステムを識別する環境変数は、プロジェクトの作成時に入力したファイルシステム識別
 子を使用して作成されます。

詳細については、Amazon Elastic File System ユーザーガイドの「<u>ファイルシステムのマウント</u>」を 参照してください。

Amazon EFS 統合のトラブルシューティング

CodeBuild で Amazon EFS を設定するときに発生する可能性のあるエラーは次のとおりです。

トピック

- <u>CLIENT_ERROR: mounting '127.0.0.1:/' failed. permission denied (クライアントエ</u> ラー:'127.0.0.1: /' のマウントに失敗しました。パーミッションが拒否されました)
- <u>CLIENT_ERROR</u>: mounting '127.0.0.1:/' failed. connection reset by peer (クライアントエ ラー:'127.0.0.1: /' のマウントに失敗しました。ピアによって接続がリセットされました)
- <u>VPC_CLIENT_ERROR</u>: Unexpected EC2 error: UnauthorizedOperation (VPC_CLIENT_ERROR: 予 期せぬEC2エラー UnauthorizedOperation)

CLIENT_ERROR: mounting '127.0.0.1:/' failed. permission denied (クライアントエラー:'127.0.0.1: /' のマウントに失敗しました。パーミッションが拒否されました)

IAM 認可は、CodeBuild を使用した Amazon EFS のマウントではサポートされていません。カスタ ム Amazon EFS ファイルシステムポリシーを使用している場合は、すべての IAM プリンシパルへの 読み取りおよび書き込みアクセスを許可する必要があります。例:

```
"Principal": {
    "AWS": "*"
}
```
CLIENT_ERROR: mounting '127.0.0.1:/' failed. connection reset by peer (クライアントエ ラー:'127.0.0.1: /' のマウントに失敗しました。ピアによって接続がリセットされました)

この問題の原因は2つ考えられます。

- CodeBuild VPC サブネットが、Amazon EFS マウントターゲットとは異なるアベイラビリティー ゾーンにあります。Amazon EFS マウントターゲットと同じアベイラビリティーゾーンに VPC サ ブネットを追加することで、この問題を解決できます。
- セキュリティグループには、Amazon EFS と通信する許可がありません。これを解決するには、VPC(VPC のプライマリ CIDR ブロックを追加する)またはセキュリティグループ自体からのすべてのトラフィックを許可するインバウンドルールを追加します。

VPC_CLIENT_ERROR: Unexpected EC2 error: UnauthorizedOperation (VPC_CLIENT_ERROR: 予期 せぬEC2エラー UnauthorizedOperation)

このエラーは、CodeBuild プロジェクトの VPC 設定内のすべてのサブネットがパブリックサブネッ トである場合に発生します。ネットワーク接続を確保するには、VPC 内に少なくとも1つのプライ ベートサブネットが必要です。

AWS CodePipeline CodeBuild のサンプル

このセクションでは、CodePipeline と CodeBuild 間のサンプル統合について説明します。

サンプル	説明
<u>CodePipeline/CodeBuild 統合とバッチビルドの</u> <u>サンプル</u>	これらのサンプルは、 AWS CodePipeline を使 用してバッチビルドを使用するビルドプロジェ クトを作成する方法を示しています。
複数の入力ソースおよび出力アーティファクト を持つ CodePipeline/CodeBuild の統合のサン プル	このサンプルでは、AWS CodePipeline を使用 して、複数の入力ソースを使用して複数の出力 アーティファクトを作成するビルドプロジェク トを作成する方法を示します。

CodePipeline/CodeBuild 統合とバッチビルドのサンプル

AWS CodeBuild はバッチビルドをサポートしています。次のサンプルは、 AWS CodePipeline を使用してバッチビルドを使用するビルドプロジェクトを作成する方法を示しています。

パイプラインの構造を定義する JSON 形式のファイルを使用し、それを で使用 AWS CLI してパ イプラインを作成できます。詳細については、『AWS CodePipeline ユーザーガイド』の「<u>AWS</u> CodePipeline パイプライン構造のリファレンス」を参照してください。

個々のアーティファクトを使用した Batch 構築

個別のアーティファクトを含むバッチビルドを作成するパイプライン構造の例として、次 の JSON ファイルを使用してください。CodePipeline でバッチビルドを有効にするには、

「BatchEnabled」パラメータのパラメータ「configuration」オブジェクトを「true」に設定 します。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "source1"
              }
            ],
            "configuration": {
              "S3Bucket": "<my-input-bucket-name>",
              "S3ObjectKey": "my-source-code-file-name.zip"
            },
            "runOrder": 1
          },
          {
            "inputArtifacts": [],
            "name": "Source2",
            "actionTypeId": {
              "category": "Source",
```

```
"owner": "AWS",
        "version": "1",
        "provider": "S3"
      },
      "outputArtifacts": [
        {
          "name": "source2"
        }
      ],
      "configuration": {
        "S3Bucket": "<my-other-input-bucket-name>",
        "S3ObjectKey": "my-other-source-code-file-name.zip"
      },
      "runOrder": 1
    }
  ]
},
{
  "name": "Build",
  "actions": [
    {
      "inputArtifacts": [
        {
          "name": "source1"
        },
        {
          "name": "source2"
        }
      ],
      "name": "Build",
      "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "version": "1",
        "provider": "CodeBuild"
      },
      "outputArtifacts": [
        {
          "name": "build1"
        },
        {
          "name": "build1_artifact1"
        },
        {
```

```
"name": "build1_artifact2"
              },
              {
                "name": "build2_artifact1"
              },
              {
                "name": "build2_artifact2"
              }
            ],
            "configuration": {
              "ProjectName": "my-build-project-name",
              "PrimarySource": "source1",
              "BatchEnabled": "true"
            },
            "runOrder": 1
          }
        ]
      }
    ],
    "artifactStore": {
      "type": "S3",
      "location": "<AWS-CodePipeline-internal-bucket-name>"
    },
    "name": "my-pipeline-name",
    "version": 1
  }
}
```

次の例は、このパイプライン設定で動作する CodeBuild buildspec ビルドファイルです。

```
version: 0.2
batch:
    build-list:
        - identifier: build1
        env:
            compute-type: BUILD_GENERAL1_SMALL
        - identifier: build2
        env:
            compute-type: BUILD_GENERAL1_MEDIUM
phases:
    build:
    commands:
```

```
- echo 'file' > output_file
artifacts:
    files:
        - output_file
secondary-artifacts:
        artifact1:
        files:
            - output_file
artifact2:
        files:
            - output_file
```

パイプラインの JSON ファイルで指定されている出力成果物の名前は、buildspec ファイルで定義 されているビルドおよびアーティファクトの識別子と一致していなければなりません。構文は、プ ライマリアーティファクトの場合は *buildIdentifier* で、セカンダリアーティファクトの場合は *buildIdentifier_artifactIdentifier* です。

たとえば、出力アーティファクト名 build1 の場合、CodeBuild は build1 の場所に「build1」を 出力します。出力名は「build1_artifact1」であり、CodeBuild はセカンダリアーティファクト を artifact1 の build1、build1_artifact1 の場所にアップロードします。出力場所が 1 つだ け指定されている場合、名前は *buildIdentifier* のみにします。

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを --cli-input-jsonパラメータに渡します。詳細につ いては、『AWS CodePipeline ユーザーガイド』の「<u>パイプラインの作成 (CLI)</u>」を参照してくださ い。

複合アーチファクトを使用したBatch ビルド

結合アーティファクトを含むバッチビルドを作成するパイプライン構造の例として、次の JSON ファイルを使用してください。CodePipeline でバッチビルドを有効にするには、

「BatchEnabled」パラメータのパラメータ「configuration」オブジェクトを「true」に設 定します。ビルド成果物を同じ場所に結合するには、「CombineArtifacts」オブジェクトの 「configuration」パラメータのパラメータを「true」に設置します。

```
{
    "pipeline": {
        "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
        "stages": [
        {
```

```
"name": "Source",
  "actions": [
    {
      "inputArtifacts": [],
      "name": "Source1",
      "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "version": "1",
        "provider": "S3"
      },
      "outputArtifacts": [
        {
          "name": "source1"
        }
      ],
      "configuration": {
        "S3Bucket": "<my-input-bucket-name>",
        "S3ObjectKey": "my-source-code-file-name.zip"
      },
      "runOrder": 1
    },
    {
      "inputArtifacts": [],
      "name": "Source2",
      "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "version": "1",
        "provider": "S3"
      },
      "outputArtifacts": [
        {
          "name": "source2"
        }
      ],
      "configuration": {
        "S3Bucket": "<my-other-input-bucket-name>",
        "S3ObjectKey": "my-other-source-code-file-name.zip"
      },
      "runOrder": 1
    }
  ]
},
```

```
{
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "source1"
            },
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "output1 "
            }
          ],
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1",
             "BatchEnabled": "true",
             "CombineArtifacts": "true"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "<AWS-CodePipeline-internal-bucket-name>"
  },
  "name": "my-pipeline-name",
  "version": 1
 }
}
```

次の例は、このパイプライン設定で動作する CodeBuild buildspec ビルドファイルです。

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM
phases:
  build:
    commands:
      - echo 'file' > output_file
artifacts:
  files:
    - output_file
```

結合アーチファクトがバッチ構築で有効になっている場合、出力は 1 つだけです。CodeBuild は、す べてのビルドの主要なアーティファクトを 1 つのZIPファイルに結合します。

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを --cli-input-jsonパラメータに渡します。詳細につ いては、『AWS CodePipeline ユーザーガイド』の「<u>パイプラインの作成 (CLI)</u>」を参照してくださ い。

複数の入力ソースおよび出力アーティファクトを持つ CodePipeline/CodeBuild の統合 のサンプル

AWS CodeBuild プロジェクトは複数の入力ソースを取ることができます。また、複数の出力アー ティファクトを作成することもできます。このサンプルでは、 AWS CodePipeline を使用して、複 数の入力ソースを使用して複数の出力アーティファクトを作成するビルドプロジェクトを作成する方 法を示します。詳細については、「<u>複数の入力ソースと出力アーティファクトのサンプル</u>」を参照し てください。

パイプラインの構造を定義する JSON 形式のファイルを使用し、それを で使用 AWS CLI してパイ プラインを作成できます。複数の入力ソースと複数の出力アーティファクトを含むビルドを作成す るパイプライン構造の例として、次の JSON ファイルを使用してください。このサンプルの後半で は、このファイルが複数の入力と出力をどのように指定しているかが分かります。詳細については、 『AWS CodePipeline ユーザーガイド』の「<u>CodePipeline パイプライン構造リファレンス</u>」を参照し てください。

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source1",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source1"
            }
          ],
          "configuration": {
            "S3Bucket": "my-input-bucket-name",
            "S3ObjectKey": "my-source-code-file-name.zip"
          },
          "runOrder": 1
        },
        {
          "inputArtifacts": [],
          "name": "Source2",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
```

```
}
      ],
      "configuration": {
        "S3Bucket": "my-other-input-bucket-name",
        "S3ObjectKey": "my-other-source-code-file-name.zip"
      },
      "runOrder": 1
    }
  ]
},
{
  "name": "Build",
  "actions": [
    {
      "inputArtifacts": [
        {
          "name": "source1"
        },
        {
          "name": "source2"
        }
      ],
      "name": "Build",
      "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "version": "1",
        "provider": "AWS CodeBuild"
      },
      "outputArtifacts": [
        {
          "name": "artifact1"
        },
        {
          "name": "artifact2"
        }
      ],
      "configuration": {
        "ProjectName": "my-build-project-name",
        "PrimarySource": "source1"
      },
      "runOrder": 1
    }
  ]
```

```
}
],
"artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
},
"name": "my-pipeline-name",
"version": 1
}
```

この JSON ファイルの制約事項:

- 入力ソースの1つを PrimarySourceに指定する必要があります。このソースは、CodeBuild が buildspec ファイルを探して実行するディレクトリです。キーワード PrimarySource は、JSON ファイルの CodeBuild ステージのconfiguration セクションにプライマリソースを指定するの に使用されます。
- 各入力ソースは、それぞれのディレクトリにインストールされます。このディレクトリは、組み込み環境変数 \$CODEBUILD_SRC_DIR (プライマリソースの場合) と \$CODEBUILD_SRC_DIR_yourInputArtifactName (他のすべてのソースの場合) に保存されます。このサンプルのパイプラインでは、2 つの入力ソースディレクトリは \$CODEBUILD_SRC_DIRと \$CODEBUILD_SRC_DIR_source2 です。詳細については、「ビルド環境の環境変数」を参照してください。
- パイプラインの JSON ファイルで指定されている出力成果物の名前は、buildspec ファイルで定義 されているセカンダリアーティファクトの名前と一致していなければなりません。このパイプライ ンは、次の buildspec ファイルを使用します。詳細については、「<u>buildspec の構文</u>」を参照して ください。

```
version: 0.2
phases:
    build:
        commands:
            - touch source1_file
            - cd $CODEBUILD_SRC_DIR_source2
            - touch source2_file
artifacts:
    files:
```

```
- '**/*'
secondary-artifacts:
artifact1:
    base-directory: $CODEBUILD_SRC_DIR
    files:
        - source1_file
artifact2:
    base-directory: $CODEBUILD_SRC_DIR_source2
    files:
        - source2_file
```

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを --cli-input-jsonパラメータに渡します。詳細につ いては、『AWS CodePipeline ユーザーガイド』の「<u>パイプラインの作成 (CLI)</u>」を参照してくださ い。

AWS Config CodeBuild を使用したサンプル

AWS Config は、 AWS リソースのインベントリと、これらのリソースの設定変更の履歴を提供します。 は AWS リソース AWS CodeBuild として をサポートする AWS Config ようになりました。つまり、サービスは CodeBuild プロジェクトを追跡できます。詳細については AWS Config、「 AWS Config デベロッパーガイド」の「What is AWS Config?」を参照してください。

CodeBuild リソースに関する以下の情報は、 AWS Config コンソールのリソースインベントリページ で確認できます。

- CodeBuild 設定変更のタイムライン。
- 各 CodeBuild プロジェクトの設定詳細。
- ・ 他の AWS リソースとの関係。
- CodeBuild プロジェクトの変更のリスト。

トピック

- で CodeBuild を使用する AWS Config
- ステップ 3: AWS Config コンソールで AWS CodeBuild データを表示する

で CodeBuild を使用する AWS Config

このトピックの手順では、CodeBuild プロジェクトをセットアップ AWS Config および検索する方法 を示します。

トピック

- 前提条件
- ステップ 1: をセットアップする AWS Config
- ステップ 2: AWS CodeBuild プロジェクトを検索する

前提条件

AWS CodeBuild プロジェクトを作成します。手順については、<u>ビルドプロジェクトの作成</u>を参照し てください。

ステップ 1: をセットアップする AWS Config

- ・ AWS Config のセットアップ (コンソール)
- AWS Config をセットアップする (AWS CLI)

Note

セットアップが完了すると、 AWS Config コンソールに AWS CodeBuild プロジェクトが表示されるまでに最大 10 分かかる場合があります。

ステップ 2: AWS CodeBuild プロジェクトを検索する

- 1. AWS マネジメントコンソールにサインインし、 AWS Config コンソールを <u>https://</u> <u>console.aws.amazon.com/config</u>://www.com で開きます。
- 2. [リソースインベントリ] ページで、[リソースタイプ] の [AWS CodeBuild プロジェクト] を選択 します。下方にスクロールして [CodeBuild プロジェクト] チェックボックスをオンにします。
- 3. [検索]を選択します。
- CodeBuild プロジェクトのリストが追加されたら、[Configのタイムライン] 列で CodeBuild プロ ジェクト名のリンクを選択します。

ステップ 3: AWS Config コンソールで AWS CodeBuild データを表示する

リソースインベントリページでリソースを検索するときに、 AWS Config タイムラインを選択して CodeBuild プロジェクトの詳細を表示できます。リソースの詳細ページは、リソースの設定、関係、 および変更回数の情報を提供します。

ページの上部にあるブロックは、まとめてタイムラインと呼ばれます。タイムラインは、記録を取った日付と時刻を示します。

詳細については、「AWS Config デベロッパーガイド」の<u>AWS Config 「コンソールでの設定の詳細</u> の表示」を参照してください。

CodeBuild のビルド通知サンプル

Amazon CloudWatch Events には、 のサポートが組み込まれています AWS

CodeBuild。CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのストリー ムです。CloudWatch Events では、宣言型のルールを書き込んで、目的のイベントを自動アクショ ンに関連付けます。このサンプルでは、Amazon CloudWatch Events と Amazon Simple Notification Service (Amazon SNS) を使用して、ビルドの成功、失敗、各ビルドフェーズへの移行、またはこれ らのイベントの組み合わせを行うたびに、ビルド通知をサブスクライバーに送信します。

A Important

このサンプルを実行すると、AWS アカウントに料金が発生する可能性があります。これ には、CodeBuild と Amazon CloudWatch および Amazon SNS に関連する AWS リソース とアクションの料金が含まれます。詳細については、「<u>CodeBuild 料金表</u>」、「<u>Amazon</u> CloudWatch 料金表」および「Amazon SNS 料金表」を参照してください。

トピック

- ビルド通知サンプルを実行
- ビルド通知の入力形式に関するリファレンス

ビルド通知サンプルを実行

ビルド通知サンプルを実行するには、次の手順に従います。

このサンプルを実行するには

 このサンプルで使用するトピックをすでに設定して Amazon SNS で購読している場合は、ス テップ4に進みます。それ以外の場合は、AWS ルートアカウントまたは管理者ユーザーの代 わりに IAM ユーザーを使用して Amazon SNS を操作する場合は、ユーザー (またはユーザーが 関連付けられている IAM グループ)に次のステートメント (### BEGIN ADDING STATEMENT HERE ### と ### END ADDING STATEMENT HERE ### の間)を追加します。AWS ルートア カウントの使用はお勧めしません。このステートメントにより、Amazon SNS のトピックへの 通知の表示、作成、サブスクライブ、および送信テストができます。省略記号(...)は、簡潔 にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを 削除しないでください、また、これらの省略記号を既存のポリシーに入力しないでください。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    . . .
  ],
  "Version": "2012-10-17"
}
```

Note

このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアク セス許可を持っている必要があります。 詳細については、「<u>カスタマー管理ポリシーの編集</u>」または、「IAM ユーザーガイド」 の「<u>インラインポリシーの使用 (コンソール)</u>」の「グループ、ユーザー、ロールのイン ラインポリシーを編集または削除するには」セクションを参照してください。

2. Amazon SNS でトピックを作成または識別します。 は CloudWatch Events AWS CodeBuild を 使用して、Amazon SNS を介してこのトピックにビルド通知を送信します。

トピックを作成するには:

- 1. Amazon SNS コンソール(https://console.aws.amazon.com/sns)を開きます。
- 2. [トピックの作成] を選択します。
- (新しいトピックの作成)で、[トピック名] にトピックの名前 (CodeBuildDemoTopic など) を入力します。(別の名前を選択する場合は、このサンプル全体でそれを置き換えてください。)
- 4. [トピックの作成]を選択します。
- 5. [トピックの詳細: CodeBuildDemoTopic] ページで、[トピック ARN] の値をコピーします。この値は次のステップで必要になります。

Topic details: CodeBuildDemoTopic	
Publish to topic	Other topic actions -
Topic ARN	arn:aws:sns:us-east-1: CodeBuildDemoTopic
Topic owner	1029G-0033-00291
Region	us-east-1
Display name	

詳細については、Amazon SNS デベロッパーガイドの「<u>トピックの作成</u>」を参照してくださ い。

3. 1 つかそれ以上の受信者にトピックをサブスクライブさせ、E メール通知を受け取ります。

受信者にトピックをサブスクライブさせるには:

- 前のステップで Amazon SNS コンソールを開いた状態のまま、ナビゲーションペインで、 [Subscriptions] (サブスクリプション) を選択してから、[Create subscription] (サブスクリプ ションの作成) を選択します。
- 2. [サブスクリプションの作成] の [トピック ARN] に、前のステップからコピーしたトピック ARN を貼り付けます。
- 3. [Protocol] で [Email] を選択します。
- 4. [エンドポイント] に、受信者の完全な E メールアドレスを入力します。

Create subscription	
Topic ARN	arn:aws:sns:us-east-1:
Protocol	Email -
Endpoint	mary@example.com
	Cancel Create subscription

- 5. [Create Subscription] を選択します。
- 6. Amazon SNS は受信者にサブスクリプション確認の E メールを送信します。E メール通知 の受信を開始するには、受信者は受信登録確認メールで [Confirm subscription] リンクを選 択する必要があります。受信者がリンクをクリックした後、正常にサブスクライブされた ら、Amazon SNS により受信者のウェブブラウザに確認メッセージが表示されます。

詳細については、Amazon SNS 開発者ガイドの「<u>トピックのサブスクライブ</u>」を参照してくだ さい。

4. AWS ルートアカウントまたは管理者ユーザーの代わりに ユーザーを使用して CloudWatch Events を操作する場合は、ユーザー (またはユーザーが関連付けられている IAM グループ) に次のステートメント (### BEGIN ADDING STATEMENT HERE ### と ### END ADDING STATEMENT HERE ### の間)を追加します。 AWS ルートアカウントの使用はお勧めしませ ん。このステートメントは、CloudWatch Events の使用をユーザーに許可するために使用しま す。省略記号 (...)は、簡潔にするために使用され、ステートメントを追加する場所の特定に 役立ちます。ステートメントを削除しないでください、また、これらの省略記号を既存のポリ シーに入力しないでください。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "events:*",
        "iam:PassRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    . . .
 ],
 "Version": "2012-10-17"
}
```

```
 Note
```

このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアク セス許可を持っている必要があります。 詳細については、「<u>カスタマー管理ポリシーの編集</u>」または、「IAM ユーザーガイド」 の「<u>インラインポリシーの使用 (コンソール)</u>」の「グループ、ユーザー、ロールのイン ラインポリシーを編集または削除するには」セクションを参照してください。

- 5. CloudWatch Events ルールを作成します。これを行うために、CloudWatch コンソール (<u>https://</u> <u>console.aws.amazon.com/cloudwatch/</u>) を開きます。
- 6. ナビゲーションペインの [Events] で、[Rules] を選択してから、[Create rule] を選択します。
- [ステップ 1: ルールの作成] ページで、[イベントパターン] と [サービス別のイベントに一致する イベントパターンの構築] が選択済みであることを確認します。
- 8. [サービス名] で、[CodeBuild] を選択します。[イベントタイプ] で、[すべてのイベント] が選択済 みであることを確認します。
- 9. [イベントパターンのプレビュー]には、次のコードが表示されます。



}

```
"aws.codebuild"
]
```

10. [編集] を選択し、[イベントパターンのプレビュー] のコードを、次の 2 つのルールパターンのい ずれかに置き換えます。

この最初のルールパターンは、 AWS CodeBuildで指定されたビルドプロジェクトのビルドが開 始または完了すると、イベントをトリガーします。

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build State Change"
  ],
  "detail": {
    "build-status": [
      "IN_PROGRESS",
      "SUCCEEDED",
      "FAILED",
      "STOPPED"
    ],
    "project-name": [
      "my-demo-project-1",
      "my-demo-project-2"
    1
  }
}
```

前述のルールで、必要に応じて次のコードを変更します。

- ビルドが開始または完了したときにイベントをトリガーするには、build-status 配列に表示されているすべての値をそのままにするか、build-status 配列を完全に削除します。
- ビルドが完了したときにのみイベントをトリガーするには、IN_PROGRESS 配列から buildstatus を削除します。
- ビルドの開始時にのみイベントをトリガーするには、IN_PROGRESS 配列から buildstatus を除くすべての値を削除します。
- すべてのビルドプロジェクトのイベントをトリガーするには、project-name 配列を完全に 削除します。

 個々のビルドプロジェクトのイベントのみをトリガーするには、project-name 配列に各ビ ルドプロジェクトの名前を指定します。

この 2 番目のルールパターンでは、 AWS CodeBuildで指定されたビルドプロジェクトのビルド フェーズが別のビルドフェーズに移動するたびに、イベントをトリガーします。

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build Phase Change"
  ],
  "detail": {
    "completed-phase": [
      "SUBMITTED",
      "PROVISIONING",
      "DOWNLOAD_SOURCE",
      "INSTALL",
      "PRE_BUILD",
      "BUILD",
      "POST_BUILD",
      "UPLOAD_ARTIFACTS",
      "FINALIZING"
    ],
    "completed-phase-status": [
      "TIMED_OUT",
      "STOPPED",
      "FAILED",
      "SUCCEEDED",
      "FAULT",
      "CLIENT_ERROR"
    ],
    "project-name": [
      "my-demo-project-1",
      "my-demo-project-2"
    ]
  }
}
```

前述のルールで、必要に応じて次のコードを変更します。

- ビルドフェーズの変更 (各ビルドで送信される通知は最大9個) ごとにイベントをトリガーするには、completed-phase 配列に表示されているすべての値をそのままにするか、completed-phase 配列を完全に削除します。
- 個々のビルドフェーズの変更に対してのみイベントをトリガーするには、イベントをトリガー しない completed-phase 配列の各ビルドフェーズの名前を削除します。
- 各ビルドフェーズステータスが変更するたびにイベントをトリガーするには、completedphase-status 配列に示すように、すべて値をそのままにするか、completed-phasestatus 配列を完全に削除します。
- 個々のビルドフェーズステータスの変更に対してのみイベントをトリガーするには、イベント をトリガーしない completed-phase-status 配列の各ビルドフェーズステータスの名前を 削除します。
- すべてのビルドプロジェクトのイベントをトリガーするには、project-name 配列を削除します。
- 個々のビルドプロジェクトのイベントをトリガーするには、project-name 配列に各ビルド プロジェクトの名前を指定します。

イベントパターンの詳細については、Amazon EventBridge ユーザーガイドの「<u>イベントパター</u> ン」を参照してください。

イベントパターンを用いたフィルタリングの詳細については、Amazon EventBridge ユーザーガ イドの「<u>イベントパターンを使用したコンテンツベースのフィルタリング</u>」を参照してくださ い。

Note

ビルド状態の変更とビルドフェーズの変更の両方に応じてイベントをトリガーする場合 は、ビルド状態の変更用とビルドフェーズの変更用に2つの別個のルールを作成する必 要があります。両方のルールを1つのルールに結合すると、結合したルールは予期しな い結果を引き起こすか、まったく動作しなくなる可能性があります。

コードの置換を完了したら、[Save]を選択します。

- 11. [Targets] で、[Add target] を選択します。
- 12. ターゲットのリストで、[SNS トピック] を選択します。

- 13. [Topic] で、以前に指定した、または作成したトピックを選択します。
- 14. [入力の設定]を展開して、[インプットトランスフォーマー]を閉じます。
- 15. [Input Path] ボックスに、次のいずれかの入力パスを入力します。

detail-type の値が CodeBuild Build State Change であるルールの場合は、次のよう に入力します。

{"build-id":"\$.detail.build-id","project-name":"\$.detail.project-name","buildstatus":"\$.detail.build-status"}

detail-type の値が CodeBuild Build Phase Change であるルールの場合は、次のよう に入力します。

{"build-id":"\$.detail.build-id","project-name":"\$.detail.project-name","completedphase":"\$.detail.completed-phase","completed-phase-status":"\$.detail.completedphase-status"}

他のタイプの情報を取得するには、「<u>ビルド通知の入力形式に関するリファレンス</u>」を参照して ください。

16. [入力テンプレート] ボックスに、次のいずれかの入力テンプレートを入力します。

detail-type の値が CodeBuild Build State Change であるルールの場合は、次のよう に入力します。

"Build '<build-id>' for build project '<project-name>' has reached the build status of '<build-status>'."

detail-type の値が CodeBuild Build Phase Change であるルールの場合は、次のよう に入力します。

"Build '<build-id>' for build project '<project-name>' has completed the build phase of '<completed-phase>' with a status of '<completed-phase-status>'."

- 17. [設定の詳細]を選択します。
- 18. [ステップ 2: ルールの詳細を設定する] ページで、名前と説明 (オプション) を入力します。[状態] は、[有効] のままとします。

19. [Create rule] を選択します。

ビルド通知サンプル

20. ビルドプロジェクトを作成して、ビルドを実行し、ビルド情報を表示します。

21. CodeBuild がビルド通知を現在正常に送信していることを確認します。たとえば、ビルド通知 E メールが受信トレイにあるかどうかを確認します。

ルールの動作を変更するには、CloudWatch コンソールで変更するルールを選択し、[アクション]、 [編集] の順に選択します。ルールを編集し、[設定の詳細]、[ルールの更新] の順に選択します。

ルールを使用したビルド通知の送信を停止するには、CloudWatch コンソールで、使用を停止する ルールを選択し、[アクション]、[無効化] の順に選択します。

ルールを完全に削除するには、CloudWatch コンソールで、削除するルールを選択し、[アクション]、[削除] の順に選択します。

ビルド通知の入力形式に関するリファレンス

CloudWatch では、JSON 形式で通知が送信されます。

ビルド状態変更通知は次の形式を使用します。

```
{
  "version": "0",
  "id": "c030038d-8c4d-6141-9545-00ff7b7153EX",
  "detail-type": "CodeBuild Build State Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:28Z",
  "region": "us-west-2",
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "build-status": "SUCCEEDED",
    "project-name": "my-sample-project",
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "additional-information": {
      "artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
```

```
"location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:13:05 PM",
          "duration-in-seconds": 36,
          "phase-type": "PROVISIONING",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
```

```
"start-time": "Sep 1, 2017 4:13:05 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 4,
  "phase-type": "DOWNLOAD_SOURCE",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 0,
  "phase-type": "INSTALL",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 0,
  "phase-type": "PRE_BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 70,
  "phase-type": "BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:14:21 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 0,
  "phase-type": "POST_BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:14:21 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 0,
  "phase-type": "UPLOAD_ARTIFACTS",
```

```
"phase-status": "SUCCEEDED"
        },
         {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "current-phase": "COMPLETED",
    "current-phase-context": "[]",
    "version": "1"
  }
}
```

ビルドフェーズ変更通知は次の形式を使用します。

```
{
  "version": "0",
  "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
  "detail-type": "CodeBuild Build Phase Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:21Z",
  "region": "us-west-2",
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "completed-phase": "COMPLETED",
    "project-name": "my-sample-project",
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "completed-phase-context": "[]",
    "additional-information": {
```

```
"artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:13:05 PM",
          "duration-in-seconds": 36,
          "phase-type": "PROVISIONING",
```

```
"phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:05 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 4,
  "phase-type": "DOWNLOAD_SOURCE",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 0,
  "phase-type": "INSTALL",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:13:10 PM",
  "duration-in-seconds": 0,
  "phase-type": "PRE_BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:13:10 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 70,
  "phase-type": "BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:14:21 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 0,
  "phase-type": "POST_BUILD",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
```

```
"start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "UPLOAD_ARTIFACTS",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "completed-phase-status": "SUCCEEDED",
    "completed-phase-duration-seconds": 4,
    "version": "1",
    "completed-phase-start": "Sep 1, 2017 4:14:21 PM",
    "completed-phase-end": "Sep 1, 2017 4:14:26 PM"
  }
}
```

CodeBuild でのビルドバッジサンプル

AWS CodeBuild では、ビルドバッジの使用がサポートされるようになりました。ビルドバッジは、 プロジェクトの最新のビルドのステータスを表示する埋め込み可能な動的に生成されたイメージ (バッジ)を提供します。このイメージにアクセスするには、CodeBuild プロジェクトに対して生成さ れるパブリックアクセス可能な URL を使用できます。そのため、誰でも CodeBuild プロジェクトの ステータスを確認できます。ビルドバッジにはセキュリティ情報が含まれないため、認証は不要で す。

トピック

- ・ ビルドバッジを使用してビルドプロジェクトを作成
- AWS CodeBuild ビルドバッジにアクセスする

- CodeBuild ビルドバッジの公開
- CodeBuild バッジのステータス

ビルドバッジを使用してビルドプロジェクトを作成

ビルドバッジを有効にしてビルドプロジェクトを作成するには、次のいずれかの手順を実行します。 AWS CLI または を使用できます AWS Management Console。

ビルドバッジを有効にしてビルドプロジェクトを作成するには (AWS CLI)

ビルドプロジェクトの作成の詳細については、「ビルドプロジェクトの作成 (AWS CLI)」」を参照してください。ビルドバッジを AWS CodeBuild プロジェクトに含めるには、badgeEnabledをtrueの値で指定する必要があります。

ビルドバッジを有効にしてビルドプロジェクトを作成するには (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
- 4. [ソース] の [ソースプロバイダ] で、ソースコードプロバイダタイプを選択し、次のいずれかの操作を行います。

Note

CodeBuild は、Amazon S3 ソースプロバイダーでのビルドバッジをサポートしてい ません。はアーティファクト転送に Amazon S3 AWS CodePipeline を使用するた め、CodePipeline で作成されたパイプラインの一部であるビルドプロジェクトではビル ドバッジはサポートされていません。

- [CodeCommit] を選択した場合は、[リポジトリ] で、リポジトリの名前を選択します。[Enable build badge (ビルドバッジを有効にする)] を選択すると、プロジェクトのビルドステータスが 表示可能および埋め込み可能になります。
- [GitHub] を選択した場合は、手順に従って GitHub に接続 (または再接続) します。GitHub の アプリケーションの承認ページで、組織アクセスで、アクセス AWS CodeBuild を許可する各 リポジトリの横にあるアクセスリクエストを選択します。[Authorize application (アプリケー ションの承認)] を選択した後で AWS CodeBuild コンソールに戻り、[リポジトリ] でソース コードが含まれているリポジトリの名前を選択します。[Enable build badge (ビルドバッジを 有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能 になります。
- [Bitbucket] を選択した場合は、手順に従って Bitbucket に接続 (または再接続) しま す。Bitbucket の [Confirm access to your account] ページで、[Organization access] の [Grant access] を選択します。アクセス許可を選択したら、 AWS CodeBuild コンソールに戻り、リ ポジトリ で、ソースコードを含むリポジトリの名前を選択します。[Enable build badge (ビル ドバッジを有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋 め込み可能になります。

A Important

プロジェクトソースを更新すると、プロジェクトのビルドバッジの正確性に影響する 場合があります。

5. [環境] で以下の操作を行います。

[Environment image (環境イメージ)] で、次のいずれかの操作を行います。

- ・によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム、ランタイム (複数可)、イメージ、イメージバージョンから選択します。利用可能な場合は、[環境タイプ] から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジスト リ URL)] に docker repository/docker image name の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選 択します。

- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択しま す。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジスト リ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。 認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「<u>AWS Secrets Managerとは</u>」を参照してください。
- 6. [Service role (サービスロール)] で、次のいずれかの操作を行います。
 - CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name] に、新しいロールの名前を入力します。
 - CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選 択します。[Role ARN] で、サービスロールを選択します。

Note

コンソールでは、ビルドプロジェクトの作成時や更新時に CodeBuild サービスロール も作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できま す。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、 この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロー ルは最大 10 個のビルドプロジェクトで使用できます。

- 7. [Buildspec] で、次のいずれかを行います。
 - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレ クトリの buildspec.yml を使用します。
 - [ビルドコマンドの挿入]を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

- 8. [アーティファクト]の[タイプ]で、次のいずれかの操作を行います。
 - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)]を選択します。
 - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を 行います。
 - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)]を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アー

ティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名 ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。

- [Bucket name (バケット名)] で、出力バケットの名前を選択します。
- この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。 複数の場所の場合は、各場所をコンマで区切ります (例: appspec.yml, target/my-app.jar)。詳細については、「files」で buildspec の構文の説明を参照してください。
- [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを選択します。
- 10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

AWS CodeBuild ビルドバッジにアクセスする

AWS CodeBuild コンソールまたは を使用して AWS CLI ビルドバッジにアクセスできます。

- CodeBuild コンソールでは、ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクトに 対応するリンクを選択します。[ビルドプロジェクト: *project-name*] ページで、[設定] の [Copy badge URL (バッジ URL のコピー)] を選択します。詳細については、「ビルドプロジェクトの詳 細を表示する (コンソール)」を参照してください。
- で AWS CLI、 batch-get-projects コマンドを実行します。ビルドバッジの URL は出力のプロジェクト環境の詳細セクションを含まれています。詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」を参照してください。

ビルドバッジのリクエスト URL は共通のデフォルトブランチのものですが、ビルドの実行に使用したソースリポジトリの任意のブランチを指定できます。次に例を示します。

https://codebuild.us-east-1.amazon.com/badges?uuid=...&branch=<branch>

また、バッジの URL の「branch」パラメーターで「tag」パラメーターを置き換えることにより、 ソースリポジトリからタグを指定することもできます。以下に例を示します。

https://codebuild.us-east-1.amazon.com/badges?uuid=...&tag=<tag>

CodeBuild ビルドバッジの公開

マークダウンのイメージのビルドバッジ URL を使用して、マークダウンファイルに最新ビルドのス テータスを表示できます。これは、ソースリポジトリ (GitHub や CodeCommit など) の readme.md ファイルに最新ビルドのステータスを表示する場合に便利です。例:

![](<build badge URL>)

CodeBuild バッジのステータス

CodeBuild ビルドバッジには、次のいずれかのステータスがあります。

- PASSING 該当するブランチで最新ビルドが成功しました。
- FAILING 該当するブランチで最新ビルドがタイムアウト、失敗、途中終了、または停止しました。
- IN_PROGRESS 該当するブランチで最新ビルドが進行中です。
- UNKNOWN 該当するブランチでプロジェクトがビルドをまだ実行していないか、まったく実行したことがありません。また、ビルドバッジ機能が無効になっている可能性もあります。

「 サンプルを使用したテストレポート AWS CLI」

buildspec ファイルで指定したテストは、ビルド中に実行されます。このサンプルでは、 を使用して CodeBuild のビルドにテスト AWS CLI を組み込む方法を示します。JUnit を使用して単体テストを 作成または、別のツールを使用して構成テストを作成することもできます。その後、テスト結果を評 価して、問題を修正したり、アプリケーションを最適化したりできます。

CodeBuild API または AWS CodeBuild コンソールを使用して、テスト結果にアクセスできます。こ のサンプルでは、テスト結果が S3 バケットにエクスポートされるようにレポートを設定する方法を 示します。

トピック

<u>テストレポートサンプルを実行</u>

テストレポートサンプルを実行

次の手順を使用して、テストレポートサンプルを実行します。

トピック

前提条件

- ステップ 1: レポートグループを作成
- <u>ステップ 2: レポートグループによるプロジェクトの</u>設定
- ステップ 3: レポートの実行と結果の表示

前提条件

● テストケースの作成 このサンプルは、サンプルテストレポートに含めるテストケースがあるとい う前提で書かれています。buildspec ファイルでテストファイルの場所を指定します。

以下のテストレポートファイル形式がサポートされています。

- Cucumber JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

Surefire JUnit plugin、TestNG、Cucumber などのいずれかの形式でレポートファイルを作成でき る任意のテストフレームワークを使用して、テストケースを作成します。

- S3 バケットを作成し、その名前を書き留めます。詳細については、Amazon S3 ユーザーガイドの 「S3 バケットを作成する方法」を参照してください。
- IAM ロールを作成し、その ARN を書き留めます。ビルドプロジェクトを作成する際は、ARN が 必要です。
- ロールに次の権限がない場合は、追加します。

```
{
       "Effect": "Allow",
       "Resource": [
           "*"
       ],
       "Action": [
           "codebuild:CreateReportGroup",
テストレポートサンプルを実行
```

```
"codebuild:CreateReport",
    "codebuild:UpdateReport",
    "codebuild:BatchPutTestCases"
]
}
```

詳細については、「テストレポートオペレーションのアクセス許可」を参照してください。

ステップ 1: レポートグループを作成

- 1. CreateReportGroupInput.jsonという名前のファイルを作成します。
- 2. S3 バケットに、テスト結果をエクスポートするフォルダを作成します。
- 以下を CreateReportGroupInput.json にコピーします。<bucket-name> で、S3 バケットの名前を使用します。<path-to-folder> で、S3 バケット内のフォルダへのパスを入力します。
 ます。

```
{
   "name": "<report-name>",
   "type": "TEST",
   "exportConfig": {
      "exportConfigType": "S3",
      "s3Destination": {
        "bucket": "<bucket-name>",
        "path": "<path-to-folder>",
        "packaging": "NONE"
      }
   }
}
```

CreateReportGroupInput.json が含まれているディレクトリで次のコマンドを実行します。

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

出力は次のようになります。reportGroup の ARN を書き留めます。これは、このレポートグ ループを使用するプロジェクトを作成するときに使用します。

{
```
"reportGroup": {
    "arn": "arn:aws:codebuild:us-west-2:123456789012:report-group/<report-name>",
    "name": "<report-name>",
    "type": "TEST",
    "exportConfig": {
      "exportConfigType": "S3",
      "s3Destination": {
        "bucket": "<s3-bucket-name>",
        "path": "<folder-path>",
        "packaging": "NONE",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3"
     }
    },
    "created": 1570837165.885,
    "lastModified": 1570837165.885
 }
}
```

ステップ 2: レポートグループによるプロジェクトの設定

レポートを実行するには、まずレポートグループで構成された CodeBuild ビルドプロジェクトを作 成します。レポートグループに指定されたテストケースは、ビルドの実行時に実行されます。

1. buildspec.yml という名前の buildspec ファイルを作成します。

 次のYAMLを buildspec.yml ファイルのテンプレートとして使用します。テストを実行する コマンドを必ず含めてください。reports セクションで、テストケースの結果を含むファイル を指定します。これらのファイルは、CodeBuild でアクセスできるテスト結果を保存します。作 成から 30 日後に有効期限が切れます。これらのファイルは、S3 バケットにエクスポートする 生のテストケース結果ファイルとは異なります。

```
version: 0.2
phases:
install:
    runtime-versions:
        java: openjdk8
build:
    commands:
        - echo Running tests
        - <enter commands to run your tests>
    reports:
```

```
<report-name-or-arn>: #test file information
files:
    - '<test-result-files>'
base-directory: '<optional-base-directory>'
discard-paths: false #do not remove file paths from test result files
```

Note

既存のレポートグループの ARN の代わりに、作成されていないレポートグループの 名前を指定することもできます。ARN の代わりに名前を指定すると、CodeBuild は ビルドの実行時にレポートグループを作成します。この名前には、プロジェクト名と buildspec ファイルで指定した名前が project-name-report-group-name の形 式で含まれます。詳細については、「<u>テストレポートの作成</u>」および「<u>Report group</u> naming」を参照してください。

- project.json という名前のファイルを作成します。このファイルには、create-project コマンドの入力が含まれます。
- 次の JSON を project.json にコピーします。source で、ソースファイルを含むリポジトリのタイプと場所を入力します。serviceRole で、使用しているロールの ARN を指定します。

```
{
  "name": "test-report-project",
  "description": "sample-test-report-project",
  "source": {
    "type": "CODECOMMIT|CODEPIPELINE|GITHUB|S3|BITBUCKET|GITHUB_ENTERPRISE|
NO_SOURCE",
    "location": "<your-source-url>"
 },
 "artifacts": {
    "type": "NO_ARTIFACTS"
 },
  "cache": {
    "type": "NO_CACHE"
 },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "small"
 },
  "serviceRole": "arn:aws:iam::<your-aws-account-id>:role/service-role/<your-role-
name>"
```

}

5. project.json が含まれているディレクトリで次のコマンドを実行します。これにより、test-project という名前のプロジェクトが作成されます。

aws codebuild create-project --cli-input-json file://project.json

ステップ 3: レポートの実行と結果の表示

このセクションでは、前に作成したプロジェクトのビルドを実行します。ビルドプロセス中 に、CodeBuild は、テストケースの結果を含むレポートを作成します。レポートは、指定したレポー トグループに含まれます。

 ビルドを開始するには、次のコマンドを実行します。「test-report-project」は、上記で 作成されたビルドプロジェクトの名前です。出力に表示されるビルド ID を書き留めます。

aws codebuild start-build --project-name test-report-project

 次のコマンドを実行して、レポートの ARN を含むビルドに関する情報を取得します。<buildid> で、ビルド ID を指定します。出力の「reportArns」プロパティのレポート ARN を書き 留めます。

aws codebuild batch-get-builds --ids <build-id>

 次のコマンドを実行して、レポートの詳細を取得します。
 report-arn> で、レポート ARN を指定します。

aws codebuild batch-get-reports --report-arns <report-arn>

出力は次のようになります。このサンプル出力は、成功、失敗、スキップされたテスト、エラー の結果、または不明なステータスを返したテストの数を示しています。

```
{
    "reports": [
    {
        "status": "FAILED",
        "reportGroupArn": "<report-group-arn>",
        "name": "<report-group-name>",
        "created": 1573324770.154,
    }
}
```

```
"exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
          "bucket": "<amzn-s3-demo-bucket>",
          "path": "<path-to-your-report-results>",
          "packaging": "NONE",
          "encryptionKey": "<encryption-key>"
        }
      },
      "expired": 1575916770.0,
      "truncated": false,
      "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/<name-of-
build-project>:2c254862-ddf6-4831-a53f-6839a73829c1",
      "type": "TEST",
      "arn": "<report-arn>",
      "testSummary": {
        "durationInNanoSeconds": 6657770,
        "total": 11,
        "statusCounts": {
          "FAILED": 3,
          "SKIPPED": 7,
          "ERROR": 0,
          "SUCCEEDED": 1,
          "UNKNOWN": 0
        }
      }
    }
  ],
  "reportsNotFound": []
}
```

 レポートのテストケースに関する情報を一覧表示するには、次のコマンドを実行しま す。<report-arn> で、レポートの ARN を指定します。オプションの --filter パラメータ では、(SUCCEEDED、FAILED、SKIPPED、 ERROR、または UNKNOWN) の 1 つのステータス結 果指定できます。

```
aws codebuild describe-test-cases \
    --report-arn <report-arn> \
    --filter status=SUCCEEDED|FAILED|SKIPPED|ERROR|UNKNOWN
```

出力は次のようになります。

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    },
    {
      "status": "SUCCEEDED",
      "name": "Test case 2",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    }
  ]
}
```

CodeBuild の Docker サンプル

このセクションでは、Docker との統合例について説明します AWS CodeBuild。

サンプル	説明
<u>CodeBuild のカスタム Docker イメージのサン</u> <u>プル</u>	このサンプルでは、CodeBuild とカスタ ム Docker ビルドイメージ (Docker Hub の docker : dind)を使用して Docker イメージ をビルドして実行します。
<u>CodeBuild 用の Windows Docker ビルドサンプ</u> <u>ル</u>	このサンプルは、CodeBuild を使用して Windows Docker イメージを構築して実行しま す。

サンプル	説明
<u>CodeBuild の 'Docker イメージを Amazon ECR</u> <u>イメージリポジトリに公開' サンプル</u>	このサンプルでは、Docker イメージをビル ド出力として生成し、Docker イメージを Amazon Elastic Container Registry (Amazon ECR) イメージリポジトリにプッシュします。
<u>CodeBuild AWS Secrets Manager のサンプル</u> <u>を含むプライベートレジストリ</u>	このサンプルでは、プライベートレジストリに 保存されている Docker イメージを CodeBuild ランタイム環境として使用する方法を示しま す。

CodeBuild のカスタム Docker イメージのサンプル

次のサンプルは、 とカスタム Docker ビルドイメージ (docker:dindDocker Hub の)を使用して Docker イメージを構築 AWS CodeBuild して実行します。

代わりに、Docker をサポートする CodeBuild に用意されているビルドイメージを使用して Docker イメージをビルドする方法については、「<u>'Docker イメージを Amazon ECR に公開' サンプル</u>」を参 照してください。

A Important

このサンプルを実行すると、AWS アカウントに料金が発生する可能性があります。これに は、Amazon S3、および CloudWatch Logs に関連する AWS リソースとアクションに対する CodeBuild AWS KMSと の料金が含まれます。詳細については、「[g465]CodeBuild 料金表[/ g465]」、「[g464]Amazon S3 料金表[/g464]」、「[g463]AWS Key Management Service 料 金表[/g463]」、および「[g462]Amazon CloudWatch 料金表[/g462]」を参照してください。

トピック

・ カスタムイメージサンプルで Docker を実行

カスタムイメージサンプルで Docker を実行

カスタムイメージサンプルで Docker を実行するには、次の手順に従います。このサンプルの詳細に ついては、「CodeBuild のカスタム Docker イメージのサンプル」を参照してください。

カスタムイメージサンプルで Docker を実行するには

 このトピックの ディレクトリ構造および ファイルセクションで説明されているようにファイル を作成し、S3 入力バケット、または AWS CodeCommit、GitHub、または Bitbucket リポジトリ にアップロードします。

A Important

(root directory name)をアップロードしないでください。アップロードするの は、(root directory name)内のファイルのみです。 S3入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから 入力バケットにアップロードしてください。(root directory name)を ZIP ファイ ルに追加しないでください。追加するのは、(root directory name)内のファイル のみです。

2. ビルドプロジェクトを作成して、ビルドを実行し、関連するビルド情報を表示します。

を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerCustomImageSample.zip"
 },
  "artifacts": {
    "type": "NO_ARTIFACTS"
 },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "docker:dind",
    "computeType": "BUILD_GENERAL1_SMALL",
    "privilegedMode": false
 },
 "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビ ルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime</u> <u>Privilege and Linux Capabilities</u>」を参照して、特権モードを有効にします。ま た、Windows は特権モードをサポートしていません。

ビルドの結果を表示するには、ビルドのログで文字列 Hello, World!を探します。詳細については、「ビルドの詳細の表示」を参照してください。

ディレクトリ構造

このサンプルのディレクトリ構造は次のとおりとします。

(root directory name)
buildspec.yml
Dockerfile

ファイル

このサンプルで使用されているオペレーティングシステムの基本イメージは Ubuntu です。このサン プルで使用するファイルは以下のとおりです。

buildspec.yml(内)(root directory name)

```
version: 0.2
phases:
    pre_build:
        commands:
        - docker build -t helloworld .
    build:
        commands:
        - docker images
        - docker run helloworld echo "Hello, World!"
```

Dockerfile(内)(root directory name)

FROM maven:3.3.9-jdk-8

RUN echo "Hello World"

CodeBuild 用の Windows Docker ビルドサンプル

次のサンプルは、CodeBuild を使用して Windows Docker イメージを構築して実行します。

トピック

• Windows Docker ビルドサンプルを実行する

Windows Docker ビルドサンプルを実行する

Windows Docker ビルドを実行するには、次の手順に従います。

Windows Docker ビルドサンプルを実行するには

 このトピックの ディレクトリ構造および ファイルセクションで説明されているようにファイル を作成し、S3 入力バケット、または AWS CodeCommit、GitHub、または Bitbucket リポジトリ にアップロードします。

M Important

(root directory name)をアップロードしないでください。アップロードするの は、(root directory name)内のファイルのみです。 S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから 入力バケットにアップロードしてください。(root directory name)を ZIP ファイ ルに追加しないでください。追加するのは、(root directory name)内のファイル のみです。

2. WINDOWS_EC2 フリートを作成します。

を使用してフリート AWS CLI を作成する場合、create-fleetコマンドへの JSON 形式の入 力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
    "name": "fleet-name",
    "baseCapacity": 1,
    "environmentType": "WINDOWS_EC2",
    "computeType": "BUILD_GENERAL1_MEDIUM"
}
```

3. ビルドプロジェクトを作成して、ビルドを実行し、関連するビルド情報を表示します。

を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "project-name",
  "source": {
    "type": "S3",
    "location": "bucket-name/DockerImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "WINDOWS_EC2",
    "image": "Windows",
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "fleet": {
       "fleetArn": "fleet-arn"
    }
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name"
}
```

ビルドの結果を表示するには、ビルドのログで文字列 Hello, World!を探します。詳細については、「ビルドの詳細の表示」を参照してください。

ディレクトリ構造

このサンプルのディレクトリ構造は次のとおりとします。

(root directory name) ### buildspec.yml ### Dockerfile

ファイル

このサンプルで使用されるオペレーティングシステムのベースイメージは で

すmcr.microsoft.com/windows/servercore:ltsc2022。このサンプルで使用するファイルは 以下のとおりです。

buildspec.yml(内)(root directory name)

```
version: 0.2
phases:
    pre_build:
        commands:
            - docker build -t helloworld .
        build:
            commands:
                - docker images
                - docker run helloworld powershell -Command "Write-Host 'Hello World!'"
```

Dockerfile(内)(root directory name)

FROM mcr.microsoft.com/windows/servercore:ltsc2022

RUN powershell -Command "Write-Host 'Hello World'"

CodeBuild の 'Docker イメージを Amazon ECR イメージリポジトリに公開' サンプル

このサンプルでは、Docker イメージをビルド出力として生成し、Docker イメージを Amazon Elastic Container Registry (Amazon ECR) イメージリポジトリにプッシュします。このサンプルを 適応させて、Docker イメージを Docker Hub にプッシュすることができます。詳細については、 「<u>Docker イメージを Amazon ECR に公開' サンプルを Docker Hub にプッシュするように変更</u>」を 参照してください。

カスタム Docker ビルドイメージ (Docker Hub の docker:dind) を使用して Docker イメージをビ ルドする方法については、「カスタム Docker イメージのサンプル」を参照してください。

このサンプルは、golang:1.12を参照してテストされています。

このサンプルでは、新しいマルチステージの Docker ビルド機能を使用しています。この機能により、Docker イメージがビルド出力として生成されます。次に、Docker イメージがAmazon ECR イメージリポジトリにプッシュされます。マルチステージの Docker イメージビルドは、最終的な Docker イメージのサイズを縮小するのに役立ちます。詳細については、「<u>Docker でのマルチステー</u> ジビルドの使用」を参照してください。

▲ Important

このサンプルを実行すると、AWS アカウントに料金が発生する可能性があります。こ れには、Amazon S3、、AWS KMS CloudWatch Logs、Amazon ECR に関連する AWS リソースとアクション AWS CodeBuild に対する および の料金が含まれます。詳細につ いては、<u>CodeBuild 料金表、Amazon S3 料金表、AWS Key Management Service 料金</u> <u>表、Amazon CloudWatch 料金表、Amazon Elastic Container Registry 料金表</u>を参照してくだ さい。

トピック

- ・ 'Docker イメージを Amazon ECR に公開' サンプルを実行
- ・ 'Docker イメージを Amazon ECR に公開' サンプルを Docker Hub にプッシュするように変更

'Docker イメージを Amazon ECR に公開' サンプルを実行

Docker イメージを Amazon ECR に公開するサンプルを実行するには、次の手順に従います。この サンプルの詳細については、「<u>CodeBuild の 'Docker イメージを Amazon ECR イメージリポジトリ</u> に公開' サンプル」を参照してください。

このサンプルを実行するには

 使用する Amazon ECR にイメージリポジトリがすでにある場合は、ステップ3に進みます。それ以外の場合は、AWS ルートアカウントまたは管理者ユーザーの代わりにユーザーを使用して Amazon ECR を操作する場合は、このステートメント (### BEGIN ADDING STATEMENT HERE ### と ### END ADDING STATEMENT HERE ### ##) をユーザー (またはユーザーが関連付けられている IAM グループ) に追加します。AWS ルートアカウントの使用はお勧めしません。このステートメントでは、Docker イメージを保存するための Amazon ECR リポジトリを作成できます。省略記号 (...)は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号をポリシーに入力しないでください。詳細については、ユーザーガイドの「AWS Management Consoleでのインラインポリシーの使用」を参照してください。

```
{
   "Statement": [
   ### BEGIN ADDING STATEMENT HERE ###
   {
        "Action": [
```

```
"ecr:CreateRepository"
],
    "Resource": "*",
    "Effect": "Allow"
},
    ### END ADDING STATEMENT HERE ###
    ...
],
    "Version": "2012-10-17"
}
```

Note

このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアク セス許可を持っている必要があります。

- Amazon ECR にイメージリポジトリを作成します。必ず、ビルド環境を作成してビルドを実行するのと同じ AWS リージョンにリポジトリを作成します。詳細については、Amazon ECR ユーザーガイドの「<u>リポジトリの作成</u>」を参照してください。このリポジトリの名前は、この手順で後ほど IMAGE_REP0_NAME 環境変数を使用して指定するリポジトリ名と一致させる必要があります。Amazon ECR リポジトリポリシーで、CodeBuild サービスの IAM ロールに、イメージをプッシュするアクセスが許可されていることを確認してください。
- このステートメント (### BEGIN ADDING STATEMENT HERE ### と ### END ADDING STATEMENT HERE ### ##) を AWS CodeBuild サービスロールにアタッチしたポリシーに追加 します。このステートメントでは、Amazon ECR リポジトリに Docker イメージをアップロー ドすることを CodeBuild に許可します。省略記号 (...) は、簡潔にするために使用され、ス テートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、ま た、これらの省略記号をポリシーに入力しないでください。

```
{
    "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
        "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:CompleteLayerUpload",
        "ecr:GetAuthorizationToken",
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
```

```
],
    "Resource": "*",
    "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
],
    "Version": "2012-10-17"
}
```

Note

このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアク セス許可を持っている必要があります。

 Cのトピックの ディレクトリ構造および ファイルセクションで説明されているようにファイル を作成し、S3 入力バケット、または AWS CodeCommit、GitHub、または Bitbucket リポジトリ に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「イメージ 定義ファイルのリファレンス」を参照してください。

A Important

(root directory name)をアップロードしないでください。アップロードするの は、(root directory name)内のファイルのみです。 S3入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから 入力バケットにアップロードしてください。(root directory name)を ZIP ファイ ルに追加しないでください。追加するのは、(root directory name)内のファイル のみです。

5. ビルドプロジェクトを作成して、ビルドを実行し、ビルド情報を表示します。

コンソールを使用してプロジェクトを作成する場合:

- a. [Operating system] で、[Ubuntu] を選択します。
- b. [ランタイム] で、[Standard (標準)] を選択します。
- c. [イメージ] で、[aws/codebuild/standard:5.0] を選択します。
- d. 次の環境変数を設定します。
 - AWS_DEFAULT_REGION (値は *region-ID*)

- AWS_ACCOUNT_ID (値は account-ID)
- ・ IMAGE_TAG (最新の値)
- IMAGE_REPO_NAME (値は Amazon-ECR-repo-name)

を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-docker-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"
 },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
   ],
 },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
```

}

6. CodeBuild が Docker イメージをリポジトリに正常にプッシュしたことを確認します。

- 1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
- 2. リポジトリ名を選択します。イメージは、[Image tag (イメージタグ)] 列に表示されています。

ディレクトリ構造

このサンプルのディレクトリ構造は次のとおりとします。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

ファイル

このサンプルで使用するファイルは以下のとおりです。

```
buildspec.yml(内)(root directory name)
```

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
```

```
- docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REP0_NAME:$IMAGE_TAG
```

Dockerfile(内)(root directory name)

```
FROM golang:1.12-alpine AS build
#Install git
RUN apk add --no-cache git
#Get the hello world package from a GitHub repository
RUN go get github.com/golang/example/hello
WORKDIR /go/src/github.com/golang/example/hello
# Build the project and send the output to /bin/HelloWorld
RUN go build -o /bin/HelloWorld
```

FROM golang:1.12-alpine
#Copy the build's output binary from the previous build container
COPY --from=build /bin/HelloWorld /bin/HelloWorld
ENTRYPOINT ["/bin/HelloWorld"]

Note

CodeBuild はカスタムDocker イメージの「ENTRYPOINT」をオーバーライドします。

'Docker イメージを Amazon ECR に公開' サンプルを Docker Hub にプッシュするよう に変更

'Docker イメージを Amazon ECR に公開' サンプルを調整して、Docker イメージが Amazon ECR の 代わりに Docker Hub にプッシュされるようにするには、サンプルのコードを編集します。サンプル の詳細については、「<u>CodeBuild の 'Docker イメージを Amazon ECR イメージリポジトリに公開' サ</u> ンプル」と「'Docker イメージを Amazon ECR に公開' サンプルを実行」を参照してください。

Note

使用している Docker のバージョンが 17.06 より前のものである場合は、--no-includeemail オプションを削除します。

1. buildspec.yml ファイルで、以下の Amazon ECR 固有のコード行を置き換えます。

•••
pre_build:
commands:
- echo Logging in to Amazon ECR
- aws ecr get-login-passwordregion \$AWS_DEFAULT_REGION
docker loginusername AWSpassword-stdin \$AWS_ACCOUNT_ID.dkr.ecr.
\$AWS_DEFAULT_REGION.amazonaws.com
build:
commands:
- echo Build started on `date`
- echo Building the Docker image
- docker build -t \$IMAGE_REPO_NAME:\$IMAGE_TAG .
docker tag \$IMAGE_REPO_NAME:\$IMAGE_TAG \$AWS_ACCOUNT_ID.dkr.ecr.
\$AWS_DEFAULT_REGION.amazonaws.com/\$IMAGE_REPO_NAME:\$IMAGE_TAG
post_build:
commands:
- echo Build completed on `date`
- echo Pushing the Docker image
docker push \$AWS_ACCOUNT_ID.dkr.ecr.\$AWS_DEFAULT_REGION.amazonaws.com/
<pre>\$IMAGE_REPO_NAME:\$IMAGE_TAG</pre>

代わりに、以下の Docker Hub 固有のコード行を使用します。

```
. . .
 pre_build:
   commands:
     - echo Logging in to Docker Hub...
     # Type the command to log in to your Docker Hub account here.
 build:
   commands:
     - echo Build started on `date`
     - echo Building the Docker image...
     - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
     - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
 post_build:
   commands:
     - echo Build completed on `date`
     - echo Pushing the Docker image...
     - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
. . .
```

2. 編集したコードを S3 入力バケット、または AWS CodeCommit、GitHub、または Bitbucket リ ポジトリにアップロードします。

A Important

(root directory name)をアップロードしないでください。アップロードするの は、(root directory name)内のファイルのみです。 S3入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから 入力バケットにアップロードしてください。(root directory name)を ZIP ファイ ルに追加しないでください。追加するのは、(root directory name)内のファイル のみです。

create-project コマンドに対する JSON 形式の入力で、以下のコード行が置き換えの対象です。

```
"environmentVariables": [
 {
    "name": "AWS_DEFAULT_REGION",
    "value": "region-ID"
 },
  {
    "name": "AWS_ACCOUNT_ID",
    "value": "account-ID"
 },
  ſ
    "name": "IMAGE_REPO_NAME",
    "value": "Amazon-ECR-repo-name"
 },
  {
    "name": "IMAGE_TAG",
    "value": "latest"
 }
]
```

以下のコード行に置き換えます。

```
...
"environmentVariables": [
        {
```

```
"name": "IMAGE_REPO_NAME",
    "value": "your-Docker-Hub-repo-name"
    },
    {
        "name": "IMAGE_TAG",
        "value": "latest"
    }
]
....
```

- 4. ビルド環境を作成して、ビルドを実行し、関連するビルド情報を表示します。
- が Docker イメージをリポジトリに AWS CodeBuild 正常にプッシュしたことを確認しま す。Docker Hub にサインインし、リポジトリに進み、[Tags] タブを選択します。1atest タグ には、ごく最近の [Last Updated] (最終更新) の値が含まれています。

CodeBuild AWS Secrets Manager のサンプルを含むプライベートレジスト リ

このサンプルでは、プライベートレジストリに保存されている Docker イメージを AWS CodeBuild ランタイム環境として使用する方法を示します。プライベートレジストリの認証情報は AWS Secrets Managerに保存されています。CodeBuild では任意のプライベートレジストリを使用できま す。このサンプルでは Docker Hub を使用します。

Note

シークレットはアクションに表示され、ファイルに書き込まれる際にマスクされません。

トピック

- プライベートレジストリのサンプルの要件
- プライベートレジストリを使用した CodeBuild プロジェクトの作成
- セルフホスト型ランナーのプライベートレジストリ認証情報を設定する

プライベートレジストリのサンプルの要件

でプライベートレジストリを使用するには AWS CodeBuild、以下が必要です。

 Docker Hub 認証情報を保存する Secrets Manager シークレット。この認証情報を使用してプライ ベートリポジトリにアクセスします。

Note

作成したシークレットに対して料金が発生します。

- プライベートリポジトリまたはアカウント。
- Secrets Manager シークレットへのアクセス許可を付与する CodeBuild サービスロールの IAM ポリシー。

これらのリソースを作成し、プライベートレジストリに保存されている Docker イメージを使用して CodeBuild ビルドプロジェクトを作成するには、以下の手順に従います。

プライベートレジストリを使用した CodeBuild プロジェクトの作成

 無料のプライベートリポジトリを作成する方法については、<u>Docker Hub のリポジトリ</u>に関する ページを参照してください。ターミナルで以下のコマンドを実行して、イメージのプル、ID の 取得、新しいリポジトリへのプッシュを行うこともできます。

```
docker pull amazonlinux
docker images amazonlinux --format {{.ID}}
docker tag image-id your-username/repository-name:tag
docker login
docker push your-username/repository-name
```

- 2. AWS Secrets Manager 「 ユーザーガイド」の「シー<u>AWS Secrets Manager クレットの作成</u>」 のステップに従います。
 - a. ステップ 3 の [シークレットのタイプを選択] で、[他の種類のシークレット] を選択しま す。
 - b. [キー/値のペア] で、Docker Hub ユーザー名として 1 つのキーと値のペアを作成し、Docker Hub パスワードとして 1 つのキーと値のペアを作成します
 - c. 「 AWS Secrets Manager シークレットを作成する」のステップを続行します。
 - d. ステップ 5 の [自動ローテーションの設定] ページで、自動ローテーションをオフにしま す。キーは Docker Hub の認証情報に対応しているためです。
 - e. 「AWS Secrets Manager シークレットを作成する」のステップに従って終了します。

詳細については、「What is AWS Secrets Manager?」を参照してください。

 コンソールで AWS CodeBuild プロジェクトを作成すると、CodeBuild は必要なアクセス許可を アタッチします。以外の AWS KMS キーを使用する場合はDefaultEncryptionKey、サービ スロールに追加する必要があります。詳細については、『[g852]IAM ユーザーガイド[/g852]』の 「[g851]ロールの修正 (コンソール)[/g851]」を参照してください。

Secrets Manager で使用するサービスロールには、少なくとも secretsmanager:GetSecretValue アクセス許可が必要です。

✓ KMS (1 action)			Clone	Remove
Service	KMS			
Actions	Write Decrypt			
Resources close	Specific All resources			
	key 🕜	arn:aws:kms: <region>:<account>:key/<your-key> EDIT Add ARN to restrict access</your-key></account></region>	0	🗌 Any
Request conditions	Specify request conditions (optional)			

 コンソールでプライベートレジストリに保存されている環境を使用してプロジェクトを作成する には、プロジェクトの作成時に以下の操作を行います。詳細については、ビルドプロジェクトの 作成 (コンソール)を参照してください。

Note

プライベートレジストリが VPC 内にある場合は、パブリックインターネットアクセス が必要です。CodeBuild は、VPC 内のプライベート IP アドレスからイメージを取得で きません。

- a. [環境イメージ] で、[カスタムイメージ] を選択します。
- b. [Environment type (環境タイプ)] で、[Linux] または [Windows] を選択します。
- c. [イメージレジストリ] で、[その他のレジストリ] を選択します。
- d. [外部レジストリの URL] で、画像の場所を入力し、[レジストリの認証情報 オプション] で
 Secrets Manager の認証情報の ARN または名前を入力します。

1 Note

認証情報が現在のリージョンに存在しない場合は、ARN を使用する必要がありま す。認証情報が別のリージョンに存在する場合、認証情報の名前は使用できませ ん。

セルフホスト型ランナーのプライベートレジストリ認証情報を設定する

セルフホスト型ランナーのレジストリ認証情報を設定するには、次の手順に従います。

Note

これらの認証情報は、イメージがプライベートレジストリのイメージで上書きされている場 合にのみ使用されることに注意してください。

AWS Management Console

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ビルドプロジェクトを作成するか、既存のプロジェクトを選択します。詳細については、 「ビルドプロジェクトの作成 (コンソール)」および「ビルドプロジェクトの設定の変更 (コ ンソール)」を参照してください。
- 3. [環境] で、[追加設定] を選択します。
- 追加設定で、レジストリ認証情報 AWS Secrets Manager の からシークレットの名前または ARN を入力します。オプションです。

Registry credential - optional

AWS CLI

1. 新しいプロジェクトを作成する場合は、create-project コマンドを実行します。

aws codebuild create-project \

```
--name project-name \
    --source type=source-type,location=source-location \
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn},imagePullCredentialsType=CODEBUILD|SERVICE_ROLE" \
    --artifacts type=artifacts-type \
    --service-role arn:aws:iam::account-ID:role/service-role/service-role-name
```

2. 既存のプロジェクトを更新する場合は、update-project コマンドを実行します。

```
aws codebuild update-project \
    --name project-name \
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn}"
```

ビルド出力を S3 バケットでホストする静的ウェブサイトの作成

ビルドのアーティファクトの暗号化を無効にすることができます。これにより、ウェブサイトをホス トするように設定した場所にアーティファクトを発行できます。(暗号化されたアーティファクトを 発行することはできません。) このサンプルは、ウェブフックを使用してビルドをトリガーし、ウェ ブサイトとして設定した S3 バケットにそのアーティファクトを発行する方法を示しています。

- 「<u>静的ウェブサイトのセットアップ</u>」の手順に従って、S3 バケットをウェブサイトとして動作 するように設定します。
- 2. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
- 5. [ソース] で、[ソースプロバイダ] として [GitHub] を選択します。手順に従って GitHub に接続 (または再接続) し、[Authorize (承認)] を選択します。

[ウェブフック] で、[Rebuild every time a code change is pushed to this repository (コードの変 更がこのレポジトリにプッシュされるたびに再構築する)] を選択します。このチェックボックス

は、[Use a repository in my account (自分のアカウントでリポジトリを使用する)] を選択した場合のみオンにできます。

Source	Add source			
Source 1 - Primary				
Source provider				
GitHub	▼			
Repository				
O Public repository	• Repository in my GitHub account			
GitHub repository C				
 Disconnect GitHub account Additional configuration Git clone depth 				
Git clone depth - optional				
1	▼			
Build Status - <i>optional</i> Report build statuses to source provider when your builds start and finish 				
Webhook - <i>optional</i> Rebuild every time a code change is pushed to this repository				
Branch filter - optional				
Enter a regular expression				

6. [環境]で以下の操作を行います。

[Environment image (環境イメージ)] で、次のいずれかの操作を行います。

- が管理する Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム、ランタイム (複数可)、イメージ、イメージバージョンから 選択します。利用可能な場合は、[環境タイプ]から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、[ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)]を選択した場合は、[External registry URL (外部のレジスト リ URL)] に docker repository/docker image name の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択する場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選 択します。
- プライベート Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。
- 7. [Service role (サービスロール)] で、次のいずれかの操作を行います。
 - CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name] に、新しいロールの名前を入力します。
 - CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選 択します。[Role ARN] で、サービスロールを選択します。

Note

コンソールでは、ビルドプロジェクトの作成時や更新時に CodeBuild サービスロール も作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できま す。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、 この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロー ルは最大 10 個のビルドプロジェクトで使用できます。

- 8. [Buildspec] で、次のいずれかを行います。
 - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレ クトリの buildspec.yml を使用します。

• [ビルドコマンドの挿入]を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

- 9. [アーティファクト] で、[タイプ] として Amazon S3 を選択し、S3 バケットにビルド出力を保存 します。
- 10. [バケット名] で、ステップ 1 でウェブサイトとして動作するよう設定した S3 バケットの名前を 選択します。
- [環境] で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、出力バケットに入れるビルドのファイルの場所を入力します。複数の場所がある場合は、カンマを使用して場所ごとに区切ります (例: appspec.yml, target/my-app.jar)。詳細については、「Artifacts reference-key in the buildspec file」を参照してください。
- 12. [Disable artifacts encryption (アーティファクトの暗号化を無効化)] を選択します。
- 13. [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを選択します。
- 14. [Create build project (ビルドプロジェクトの作成)] を選択します。ビルドプロジェクトページの [ビルド履歴] で、[ビルドの開始] を選択してビルドを実行します。
- 15. (オプション) Amazon S3 デベロッパーガイドの「<u>例: Amazon CloudFront でウェブサイトを高</u> 速化」の手順に従います。

複数の入力ソースと出力アーティファクトのサンプル

複数の入力ソースと複数の出力アーティファクトのセットを使用して AWS CodeBuild ビルドプロ ジェクトを作成できます。このサンプルは、ビルドプロジェクトをセットアップする方法を示してい ます。

- さまざまなタイプの複数のソースとリポジトリを使用します。
- ビルドアーティファクトを複数の S3 バケットに 1 つのビルドで発行します。

次のサンプルでは、ビルドプロジェクトを作成し、それを使用してビルドを実行します。このサン プルでは、ビルドプロジェクトの buildspec ファイルを使用して、複数のソースを組み込み、複数の アーティファクトセットを作成する方法を示します。 CodeBuild への複数のソース入力を使用して複数の出力アーティファクトを作成するパイプラインの作成方法については、「<u>複数の入力ソースおよび出力アーティファクトを持つ CodePipeline/</u> CodeBuild の統合のサンプル」を参照してください。

トピック

- 複数の入力と出力を持つビルドプロジェクトを作成
- ソースなしでビルドプロジェクトを作成

複数の入力と出力を持つビルドプロジェクトを作成

次の手順を使用して、複数の入力と出力を持つビルドプロジェクトを作成します。

複数の入力と出力を持つビルドプロジェクトを作成するには

- ソースを1つ以上のS3バケットにアップロードするか、1つ以上の CodeCommit、GitHub、GitHub Enterprise Server、またはBitbucket リポジトリにアップロード します。
- 2. プライマリソースを選択します。これは、CodeBuild が buildspec ファイルを探して実行する ソースです。
- ビルドプロジェクトを作成します。詳細については、「<u>でビルドプロジェクトを作成する AWS</u> CodeBuild」を参照してください。
- 4. ビルドプロジェクトを作成し、ビルドを実行して、ビルドに関する情報を取得します。
- 5. を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。

```
"type": "GITHUB",
      "location": "https://github.com/awslabs/aws-codebuild-jenkins-plugin",
      "sourceIdentifier": "source2"
    }
 ],
  "secondaryArtifacts": [ss
    {
      "type": "S3",
      "location": "<output-bucket>",
      "artifactIdentifier": "artifact1"
    },
    {
      "type": "S3",
      "location": "<other-output-bucket>",
      "artifactIdentifier": "artifact2"
    }
 ],
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
 },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

プライマリソースは、source 属性で定義されます。他のすべてのソースはセカンダリソー スと呼ばれ、secondarySources の下に表示されます。すべてのセカンダリソースは、 独自のディレクトリにインストールされます。このディレクトリは、組み込みの環境変数 CODEBUILD_SRC_DIR_*sourceIdentifer* に保存されます。詳細については、「<u>ビルド環境の環</u> 境変数」を参照してください。

secondaryArtifacts 属性には、アーティファクトの定義のリストが含まれます。これらのアー ティファクトは、secondary-artifacts ブロック内にネストされている buildspec ファイルの artifacts ブロックを使用します。

buildspec ファイル内のセカンダリアーティファクトは、アーティファクトと同じ構造を持ち、アー ティファクト識別子で区切られます。

Note

<u>CodeBuild API</u> で、セカンダリアーティファクトの artifactIdentifier は、CreateProject および UpdateProject で必須の属性です。セカンダリアーティファ クトを参照するために使用される必要があります。

前述の JSON 形式の入力を使用すると、プロジェクトの buildspec ファイルは次のようになります。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: openjdk11
  build:
    commands:
      - cd $CODEBUILD_SRC_DIR_source1
      - touch file1
      - cd $CODEBUILD_SRC_DIR_source2
      - touch file2
artifacts:
  files:
    _ '** *'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR_source1
     files:
        - file1
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - file2
```

sourceVersion の StartBuild 属性で API を使用して、プライマリソースのバージョンを 上書きすることができます。1 つまたは複数のセカンダリソースバージョンを上書きするに は、secondarySourceVersionOverride 属性を使用します。

の start-build コマンドへの JSON 形式の入力 AWS CLI は次のようになります。

{

```
"projectName": "sample-project",
   "secondarySourcesVersionOverride": [
        {
            "sourceIdentifier": "source1",
            "sourceVersion": "codecommit-branch"
        },
        {
            "sourceIdentifier": "source2",
            "sourceVersion": "github-branch"
        },
      ]
}
```

ソースなしでビルドプロジェクトを作成

ソースを設定するときに、「NO_SOURCE」ソースタイプを選択することによって CodeBuild プロ ジェクトを構成できます。ソースタイプが NO_SOURCE である場合、プロジェクトにはソースがない ため、buildspec ファイルを指定することはできません。代わりに、CLI の buildspec コマンドに 対する JSON 形式の入力の create-project 属性で、YAML 形式の buildspec 文字列を指定する必 要があります。次のように指定します。

```
{
    "name": "project-name",
    "source": {
        "type": "N0_SOURCE",
        "buildspec": "version: 0.2\n\nphases:\n build:\n commands:\n - command"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:5.0",
        "computeType": "BUILD_GENERAL1_SMALL",
    },
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",
        "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

詳細については、「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」を参照してください。

CodeBuild の buildspec ファイルサンプルのランタイムバージョン

Amazon Linux 2 (AL2) 標準イメージバージョン 1.0 以降、または Ubuntu 標準イメージバージョン 2.0 以降を使用している場合は、buildspec ファイルの runtime-versions セクションで 1 つ以上 のランタイムを指定できます。次のサンプルでは、プロジェクトランタイムを変更する方法、複数 のランタイムを指定する方法、および別のランタイムに依存するランタイムを指定する方法を示しま す。サポートされているランタイムについては、「<u>CodeBuild に用意されている Docker イメージ</u>」 を参照してください。

Note

ビルドコンテナで Docker を使用している場合、ビルドは特権モードで実行する必要があり ます。詳細については、<u>AWS CodeBuild ビルドを手動で実行する</u>および<u>でビルドプロジェク</u> <u>トを作成する AWS CodeBuild</u>を参照してください。

トピック

- buildspec ファイル内のランタイムバージョンを更新
- 2つのランタイムの指定

buildspec ファイル内のランタイムバージョンを更新

プロジェクトで使用されるランタイムを新しいバージョンに変更するには、buildpec ファイルの runtime-versions セクションを更新します。以下の例では、Java バージョン 8 および 11 を指 定する方法を示します。

Java バージョン 8 を指定する runtime-versions セクション:

```
phases:
    install:
        runtime-versions:
        java: corretto8
```

• Java バージョン 11 を指定する runtime-versions セクション:

```
phases:
    install:
        runtime-versions:
```

java: corretto11

次の例では、Ubuntu 標準イメージ 5.0 または Amazon Linux 2 標準イメージ 3.0 を使用し て、Python の異なるバージョンを指定する方法を示しています。

• Python バージョン 3.7 を指定する runtime-versions セクション:

```
phases:
    install:
        runtime-versions:
        python: 3.7
```

• Python バージョン 3.8 を指定する runtime-versions セクション:

```
phases:
    install:
        runtime-versions:
            python: 3.8
```

このサンプルでは、Java バージョン 8 ランタイムで始まり、その後で Java バージョン 10 ランタイ ムに更新されるプロジェクトを示します。

- 1. Maven をダウンロードし、インストールします。詳細については、Apache Maven ウェブサイトの「<u>Apache Maven のダウンロード</u>」および「<u>Apache Maven のインストール</u>」を参照してください。
- ローカルコンピュータまたはインスタンスの空のディレクトリに切り替えて、この Maven コマンドを実行します。

mvn archetype:generate "-DgroupId=com.mycompany.app" "-DartifactId=ROOT" "-DarchetypeArtifactId=maven-archetype-webapp" "-DinteractiveMode=false"

成功すると、このディレクトリ構造とファイルが作成されます。

```
.
### ROOT
### pom.xml
### src
### main
```

ユーザーガイド

resources
webapp
WEB-INF
web.xml
index.jsp

3. 次の内容で、buildspec.yml というファイルを作成します。ファイルを *(root directory name)*/my-web-app ディレクトリ内に保存します。

buildspec ファイル:

- この runtime-versions セクションでは、プロジェクトでバージョン 8 のJava ランタイム を使用することを指定します。
- この java -version コマンドは、ビルド時にプロジェクトで使用されている Java の バージョンを表示します。

ファイル構造は次のようになります。

```
(root directory name)
### my-web-app
    ### src
    # ### main
    # ### resources
    # ### webapp
    # ### WEB-INF
    # ### web.xml
```

```
# ### index.jsp
### buildspec.yml
### pom.xml
```

4. 「my-web-app」ディレクトリの内容を、S3 入力バケットにアップロードする か、CodeCommit、GitHub、または Bitbucket リポジトリにアップロードします。

A Important

(root directory name) または (root directory name)/my-web-app をアッ プロードしないでください。アップロードするのは、(root directory name)/myweb-app のディレクトリとファイルだけです。 S3 入力バケットを使用している場合は、ディレクトリ構造とファイルを必ず ZIP ファ イルに圧縮してから入力バケットにアップロードしてください。(root directory name) または (root directory name)/my-web-app を ZIP ファイルに追加しない でください。追加するのは、(root directory name)/my-web-app のディレクトリ とファイルだけです。

- 5. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。これらの設定を除いて、すべ ての設定をデフォルト値のままにします。
 - ・ [環境] の場合:
 - ・ [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
 - ・ [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
 - [ランタイム] で、[Standard (標準)] を選択します。
 - ・ Image で、aws/codebuild/amazonlinux-x86_64-standard:4.0 を選択します。
- 7. [Start build] を選択します。
- 8. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。
- ビルドが完了したら、[ビルドログ] タブでビルド出力を表示します。次のような出力が表示されます。

[Container] Date Time Phase is DOWNLOAD_SOURCE [Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src [Container] Date Time YAML location is /codebuild/output/src460614277/src/ buildspec.yml [Container] Date Time Processing environment variables [Container] Date Time Selecting 'java' runtime version 'corretto8' based on manual selections... [Container] Date Time Running command echo "Installing Java version 8 ..." Installing Java version 8 ... [Container] Date Time Running command export JAVA_HOME="\$JAVA_8_HOME" [Container] Date Time Running command export JRE_HOME="\$JRE_8_HOME" [Container] Date Time Running command export JDK_HOME="\$JDK_8_HOME" [Container] Date Time Running command for tool_path in "\$JAVA_8_HOME"/bin/* "\$JRE_8_HOME"/bin/*;

10. runtime-versions セクションを Java バージョン 11 で更新します。

```
install:
    runtime-versions:
        java: corretto11
```

11. 変更を保存したら、ビルドを再実行し、ビルド出力を表示します。現在インストールされている Java のバージョンが 11 であることが表示されます。次のような出力が表示されます:

[Container] Date Time Phase is DOWNLOAD_SOURCE [Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src [Container] Date Time YAML location is /codebuild/output/src460614277/src/ buildspec.yml [Container] Date Time Processing environment variables [Container] Date Time Selecting 'java' runtime version 'corretto11' based on manual selections... Installing Java version 11 ... [Container] Date Time Running command export JAVA_HOME="\$JAVA_11_HOME" [Container] Date Time Running command export JRE_HOME="\$JRE_11_HOME" [Container] Date Time Running command export JDK_HOME="\$JDK_11_HOME" [Container] Date Time Running command for tool_path in "\$JAVA_11_HOME"/bin/* "\$JRE_11_HOME"/bin/*;
2 つのランタイムの指定

同じ CodeBuild ビルドプロジェクトで、複数のランタイムを指定できます。このサンプルプロ ジェクトでは、2 つのソースファイルを使用します。1 つは Go ランタイムを使用し、もう 1 つは Node.js ランタイムを使用します。

- 1. my-source という名前のディレクトリを作成します。
- 2. my-source ディレクトリ内に golang-app という名前のディレクトリを作成します。
- 次の内容で、hello.go というファイルを作成します。ファイルを golang-app ディレクトリ 内に保存します。

```
package main
import "fmt"
func main() {
  fmt.Println("hello world from golang")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
  fmt.Println(!true)
  fmt.Println("good bye from golang")
}
```

- my-source ディレクトリ内に nodejs-app という名前のディレクトリを作成します。これは golang-app ディレクトリと同じレベルにある必要があります。
- 5. 次の内容で、index.js というファイルを作成します。ファイルを nodejs-app ディレクトリ 内に保存します。

```
console.log("hello world from nodejs");
console.log("1+1 =" + (1+1));
console.log("7.0/3.0 =" + 7.0/3.0);
console.log(true && false);
console.log(true || false);
console.log(!true);
console.log(!true);
```

 次の内容で、package.json というファイルを作成します。ファイルを nodejs-app ディレ クトリ内に保存します。

```
{
    "name": "mycompany-app",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "scripts": {
        "test": "echo \"run some tests here\""
    },
    "author": "",
    "license": "ISC"
}
```

 次の内容で、buildspec.yml というファイルを作成します。my-source および nodejsapp ディレクトリと同じレベルで、ファイルを golang-app ディレクトリに保存しま す。runtime-versions セクションでは、Node.js バージョン 12 および Go バージョン 1.13 ランタイムを指定します。

```
version: 0.2
phases:
 install:
    runtime-versions:
      golang: 1.13
      nodejs: 12
 build:
    commands:
      - echo Building the Go code...
      - cd $CODEBUILD_SRC_DIR/golang-app
      - go build hello.go
      - echo Building the Node code...
      - cd $CODEBUILD_SRC_DIR/nodejs-app
      - npm run test
artifacts:
 secondary-artifacts:
    golang_artifacts:
      base-directory: golang-app
      files:
        - hello
    nodejs_artifacts:
      base-directory: nodejs-app
      files:
        - index.js
```

```
- package.json
```

8. ファイル構造は次のようになります。

```
my-source
### golang-app
# ### hello.go
### nodejs.app
# ### index.js
# ### package.json
### buildspec.yml
```

9. 「my-source」ディレクトリの内容を、S3 入力バケットにアップロードする か、CodeCommit、GitHub、または Bitbucket リポジトリにアップロードします。

Important

S3 入力バケットを使用している場合は、ディレクトリ構造とファイルを必ず ZIP ファ イルに圧縮してから入力バケットにアップロードしてください。my-source を ZIP ファイルに追加しないでください。追加するのは、my-source のディレクトリとファ イルのみです。

- 10. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 11. ビルドプロジェクトを作成します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u> <u>ル)</u>」および「<u>ビルドの実行 (コンソール)</u>」を参照してください。これらの設定を除いて、すべ ての設定をデフォルト値のままにします。
 - ・ [環境] の場合:
 - ・ [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
 - ・ [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
 - ・ [ランタイム] で、[Standard (標準)] を選択します。
 - Image で、aws/codebuild/amazonlinux-x86_64-standard:4.0 を選択します。
- 12. [Create build project (ビルドプロジェクトの作成)] を選択します。
- 13. [Start build] を選択します。
- 14. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。

15. ビルドが完了したら、[ビルドログ] タブでビルド出力を表示します。次のような出力が表示されます。Go ランタイムおよび Node.js ランタイムからの出力が表示されます。また、Go アプリケーションおよび Node.js アプリケーションからの出力も表示されます。

[Container] Date Time Processing environment variables [Container] Date Time Selecting 'golang' runtime version '1.13' based on manual selections... [Container] Date Time Selecting 'nodejs' runtime version '12' based on manual selections... [Container] Date Time Running command echo "Installing Go version 1.13 ..." Installing Go version 1.13 ... [Container] Date Time Running command echo "Installing Node.js version 12 ..." Installing Node.js version 12 ... [Container] Date Time Running command n \$NODE_12_VERSION installed : v12.20.1 (with npm 6.14.10) [Container] Date Time Moving to directory /codebuild/output/src819694850/src [Container] Date Time Registering with agent [Container] Date Time Phases found in YAML: 2 [Container] Date Time INSTALL: 0 commands [Container] Date Time BUILD: 1 commands [Container] Date Time Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED [Container] Date Time Phase context status code: Message: [Container] Date Time Entering phase INSTALL [Container] Date Time Phase complete: INSTALL State: SUCCEEDED [Container] Date Time Phase context status code: Message: [Container] Date Time Entering phase PRE_BUILD [Container] Date Time Phase complete: PRE_BUILD State: SUCCEEDED [Container] Date Time Phase context status code: Message: [Container] Date Time Entering phase BUILD [Container] Date Time Running command echo Building the Go code... Building the Go code... [Container] Date Time Running command cd \$CODEBUILD_SRC_DIR/golang-app [Container] Date Time Running command go build hello.go [Container] Date Time Running command echo Building the Node code... Building the Node code... [Container] Date Time Running command cd \$CODEBUILD_SRC_DIR/nodejs-app

[Container] Date Time Running command npm run test > mycompany-app@1.0.0 test /codebuild/output/src924084119/src/nodejs-app > echo "run some tests here" run some tests here

を使用したソースバージョンサンプル AWS CodeBuild

このサンプルでは、コミット ID (コミット SHA とも呼ばれます) 以外の形式を使用してソースの バージョンを指定する方法を示します。ソースのバージョンは、以下の方法で指定できます。

- Amazon S3 ソースプロバイダーの場合は、ビルド入力 ZIP ファイルを表すオブジェクトのバー ジョン ID を使用します。
- CodeCommit、Bitbucket、GitHub、GitHub Enterprise Server の場合は、以下のいずれかを使用し ます。
 - ・プルリクエストの参照としてのプルリクエスト (例: refs/pull/1/head)。
 - ブランチ名としてのブランチ。
 - ・コミットID。
 - ・タグ。
 - ・参照とコミット ID。参照は、次のいずれかになります。
 - タグ(例:refs/tags/mytagv1.0^{full-commit-SHA})。
 - ブランチ(例:refs/heads/mydevbranch^{full-commit-SHA})。
 - プルリクエスト (例: refs/pull/1/head^{full-commit-SHA})。
- GitLab と GitLab セルフマネージドの場合は、次のいずれかを使用します。
 - ブランチ名としてのブランチ。
 - ・コミットID。
 - ・タグ。

Note

プルリクエストソースのバージョンを指定できるのは、リポジトリが GitHub または GitHub Enterprise Server である場合に限ります。

ソースバージョンのサンプル

参照とコミット ID を使用してバージョンを指定すると、バージョンのみを指定した場合よりもビル ドの DOWNLOAD_SOURCE フェーズが高速になります。これは、参照を追加すると、CodeBuild はコ ミットを見つけるためにリポジトリ全体をダウンロードする必要がないためです。

- ソースバージョンはコミット ID のみで指定できます (例: 12345678901234567890123467890123456789)。これを行う場合、CodeBuild はバージョン を見つけるためにリポジトリ全体をダウンロードする必要があります。
- ソースバージョンを指定するには、参照とコミット ID を次の形式で使用でき ます: refs/heads/branchname^{full-commit-SHA} (例: refs/heads/ main^{12345678901234567890123467890123456789})。この場合、CodeBuild は指定され たブランチだけをダウンロードしてバージョンを検索します。

Note

ビルドの DOWNLOAD_SOURCE フェーズを高速化するには、[Git clone depth] (Git のクローンの深さ) を低い数値に設定することもできます。CodeBuild がダウンロードするリポジトリの バージョン数が少なくなります。

トピック

- ・コミット ID で GitHub リポジトリバージョンを指定
- 参照とコミット ID を使用して GitHub リポジトリバージョンを指定

コミット ID で GitHub リポジトリバージョンを指定

ソースバージョンはコミット ID のみで指定できます (例:

12345678901234567890123467890123456789)。これを行う場合、CodeBuild はバージョンを 見つけるためにリポジトリ全体をダウンロードする必要があります。

コミット ID で GitHub リポジトリバージョンを指定するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。以下の設定を除いて、すべての設定をデフォルト値のままにします。

- [Source (ソース)] で、次のようにします。
 - [ソースプロバイダー] で [GitHub] を選択します。GitHub に接続されていない場合は、手順 に従って接続します。
 - ・ [レポジトリ] で、[パブリックレポジトリ] を選択します。
 - [リポジトリの URL] に、「https://github.com/aws/aws-sdk-ruby.git」と入力し ます。
- [環境] で以下の操作を行います。
 - ・ [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
 - ・ [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
 - [ランタイム] で、[Standard (標準)] を選択します。
 - Image で、aws/codebuild/amazonlinux-x86_64-standard:4.0 を選択します。
- [ビルド仕様] で、[ビルドコマンドの挿入] を選択して [Switch to editor (エディタに切り替え)] を 選択します。
- 4. [ビルドコマンド] で、プレースホルダーテキストを次のように置き換えます。

```
version: 0.2
phases:
    install:
    runtime-versions:
        ruby: 2.6
build:
        commands:
            - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
```

Ubuntu 標準イメージ 2.0 を使用する場合、runtime-versions セクションは必須です。こ こでは Ruby バージョン 2.6 ランタイムを指定していますが、任意のランタイムを使用できま す。echo コマンドは、CODEBUILD_RESOLVED_SOURCE_VERSION 環境変数に保存されている ソースコードのバージョンを表示します。

- 5. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。
- [ソースバージョン] に「046e8b67481d53bdc86c3f6affdd5d1afae6d369」と入力します。 これは、https://github.com/aws/aws-sdk-ruby.git リポジトリのコミットの SHA で す。
- 7. [Start build] を選択します。

- 8. ビルドが完了すると、以下が表示されます。
 - [ビルドログ] タブに、使用されたプロジェクトソースのバージョン。以下はその例です。

[Container] Date Time Running command echo \$CODEBUILD_RESOLVED_SOURCE_VERSION 046e8b67481d53bdc86c3f6affdd5d1afae6d369

[Container] Date Time Phase complete: BUILD State: SUCCEEDED

- [環境変数] タブに、ビルドを作成するために使用されたコミット ID と一致する [Resolved source version (解決されたソースバージョン)]。
- [フェーズ詳細] タブに、 DOWNLOAD_SOURCE フェーズの所要時間。

参照とコミット ID を使用して GitHub リポジトリバージョンを指定

ソースバージョンを指定するには、参照とコミット ID を次の形式で使用でき ます: *refs/heads/branchname*^{*full-commit-SHA*}(例: refs/heads/ main^{12345678901234567890123467890123456789})。この場合、CodeBuild は指定された ブランチだけをダウンロードしてバージョンを検索します。

参照とコミット ID を使用して GitHub リポジトリバージョンを指定するには

- 1. 「コミット ID で GitHub リポジトリバージョンを指定」のステップを完了します。
- 左のナビゲーションペインで、[ビルドプロジェクト]を選択し、先ほど作成したプロジェクトを 選択します。
- 3. [Start build] を選択します。
- [ソースバージョン]に「refs/heads/ main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}」と入力します。これは、次の 形式のブランチへのコミット ID および参照と同じです: refs/heads/branchname^{fullcommit-SHA}。
- 5. [Start build] を選択します。
- 6. ビルドが完了すると、以下が表示されます。
 - [ビルドログ] タブに、使用されたプロジェクトソースのバージョン。以下はその例です。

[Container] Date Time Running command echo \$CODEBUILD_RESOLVED_SOURCE_VERSION 046e8b67481d53bdc86c3f6affdd5d1afae6d369

[Container] Date Time Phase complete: BUILD State: SUCCEEDED

- [環境変数] タブに、ビルドを作成するために使用されたコミット ID と一致する [Resolved source version (解決されたソースバージョン)]。
- [フェーズ詳細] タブに、DOWNLOAD_SOURCE フェーズの所要時間。これは、コミット ID のみ を使用してソースのバージョンを指定する場合よりも短い時間であることが必要です。

CodeBuild のサードパーティーソースリポジトリのサンプル

このセクションでは、サードパーティーのソースリポジトリと CodeBuild 間のサンプル統合につい て説明します。

サンプル	説明
Bitbucket プルリクエストとウェブフックフィ ルタのサンプル - 「 <u>CodeBuild の 'Bitbucket プ</u> <u>ルリクエストとウェブフックフィルタ' のサン</u> <u>プルを実行</u> 」参照	このサンプルでは、Bitbucket リポジトリを使 用してプルリクエストを作成する方法について 説明します。また、Bitbucket ウェブフックを 使用して CodeBuild をトリガーし、プロジェク トのビルドを作成する方法についても説明しま す。
GitHub Enterprise Server サンプル - 「 <u>CodeBuild の GitHub Enterprise Server サン</u> <u>プルを実行</u> 」参照	このサンプルでは、GitHub Enterprise Server リポジトリに証明書がインストールされてい る場合に、CodeBuild を設定する方法を示し ます。また、Webhook を有効にして、GitHub Enterprise Server リポジトリにコード変更が プッシュされるたびに CodeBuild でソースコー ドを再ビルドする方法についても示します。
GitHub プルリクエストとウェブフックフィル タのサンプル - 「 <u>CodeBuild の GitHub プルリ</u> <u>クエストとウェブフックフィルタのサンプルを</u> <u>実行</u> 」参照	このサンプルでは、GitHub Enterprise Server リポジトリを使用してプルリクエストを作成す る方法について説明します。また、Webhook を有効にして、GitHub Enterprise Server リポ ジトリにコード変更がプッシュされるたびに CodeBuild でソースコードを再ビルドする方法 についても示します。

CodeBuild の 'Bitbucket プルリクエストとウェブフックフィルタ' のサンプ ルを実行

AWS CodeBuild は、ソースリポジトリが Bitbucket の場合にウェブフックをサポートします。つま り、ソースコードが Bitbucket リポジトリに保存されている CodeBuild ビルドプロジェクトでは、 ウェブフックを使用することで、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を再 構築できます。詳細については、「Bitbucket ウェブフックイベント」を参照してください。

このサンプルでは、Bitbucket リポジトリを使用してプルリクエストを作成する方法について説明し ます。また、Bitbucket ウェブフックを使用して CodeBuild をトリガーし、プロジェクトのビルドを 作成する方法についても説明します。

Note

Webhook を使用する場合、ユーザーが予期しないビルドをトリガーする可能性があります。 このリスクを軽減するには、「<u>ウェブフック使用のベストプラクティス。</u>」を参照してくだ さい。

トピック

- 前提条件
- ステップ 1: Bitbucket を使用してビルドプロジェクトを作成し、ウェブフックを有効化
- <u>ステップ 2: Bitbucket ウェブフックを使用してビルドをトリガー</u>

前提条件

このサンプルを実行するには、 AWS CodeBuild プロジェクトを Bitbucket アカウントに接続する必 要があります。

Note

CodeBuild によって、Bitbucket を使用したアクセス許可が更新されています。以前にプロ ジェクトを Bitbucket に接続し、Bitbucket 接続エラーになったことがある場合は、再接続の 上、CodeBuild アクセス許可を付与してウェブフックを管理する必要があります。 ステップ 1: Bitbucket を使用してビルドプロジェクトを作成し、ウェブフックを有効 化

次の手順では、Bitbucket をソースリポジトリとして使用して AWS CodeBuild プロジェクトを作成 し、ウェブフックを有効にする方法について説明します。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- 3. [Create build project (ビルドプロジェクトの作成)] を選択します。
- 4. [Project configuration (プロジェクトの設定)] で、次のようにします。

[Project name] (プロジェクト名)

このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウント ごとに一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理 解できるように、ビルドプロジェクトの説明を任意で指定することもできます。

5. [Source (ソース)] で、次のようにします。

ソースプロバイダー

[Bitbucket] を選択します。手順に従って Bitbucket に接続 (または再接続) し、[Authorize] (承認) を選択します。

リポジトリ

[Bitbucket アカウントのリポジトリ]を選択します。

まだ Bitbucket アカウントに接続していない場合は、Bitbucket のユーザーネームとパスワードを入力し、[Bitbucket 認証情報の保存] を選択します。

Bitbucket リポジトリ

Bitbucket リポジトリの URL を入力します。

6. [プライマリソース Webhook イベント] で、以下を選択します。

Note

[プライマリソース Webhook イベント] セクションは、前のステップで [Bitbucket アカ ウントのリポジトリ] を選択した場合のみに表示されます。

- プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
- 2. [イベントタイプ] から、1 つ以上のイベントを選択します。
- 3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルド を開始する] で、1 つ以上のオプションフィルタを追加します。
- イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを 開始しない] で、1 つ以上のオプションフィルタを追加します。
- 5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加] を選択しま す。

Bitbucket ウェブフックイベントタイプとフィルターの詳細については、「<u>Bitbucket ウェブフッ</u> クイベント」を参照してください。

7. [環境] で以下の操作を行います。

環境イメージ

次のいずれかを選択します。

によって管理される Docker イメージを使用するには AWS CodeBuild:

[Managed image (マネージドイメージ)] を選択し、次に [オペレーティングシステム]、[ランタイム]、[イメージ]、および [ランタイムバージョン] で適切な選択を行います。利用可能な場合は、[環境タイプ] から選択します。

別の Docker イメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、 [Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジス トリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository/docker image name* の形式に従って Docker Hub の Docker イメージの 名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。 プライベートDockerイメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、 [Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジスト リ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イ メージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要 があります。詳細については、AWS Secrets Manager 「 ユーザーガイド」の<u>「What is</u> AWS Secrets Manager?」を参照してください。

サービスロール

次のいずれかを選択します。

- CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)]
 を選択します。[Role ARN] で、サービスロールを選択します。

Note

コンソールでは、ビルドプロジェクトの作成時や更新時に CodeBuild サービスロー ルも作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用で きます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付け ると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サー ビスロールは最大 10 個のビルドプロジェクトで使用できます。

- 8. [Buildspec] で、次のいずれかを行います。
 - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレ クトリの buildspec.yml を使用します。
 - [ビルドコマンドの挿入]を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「<u>ビルド仕様 (buildspec) に関するリファレンス</u>」を参照してください。 9. [アーティファクト] で、次のようにします。 Type

次のいずれかを選択します。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)]
 を選択します。
- ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
 - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
 - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
 - この手順の前の方で[ビルドコマンドの挿入]を選択した場合は、[出力ファイル]に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります(例: appspec.yml, target/my-app.jar)。詳細については、「files」で <u>buildspec の構文</u>の説明を参照してください。

追加設定

[Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを設定します。

10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

ステップ 2: Bitbucket ウェブフックを使用してビルドをトリガー

Bitbucket ウェブフックを使用するプロジェクトの場合、Bitbucket リポジトリがソースコードの変更 を検出すると、 はビルド AWS CodeBuild を作成します。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ナビゲーションペインで [ビルドプロジェクト] を選択後、ウェブフックを使用して Bitbucket リ ポジトリと関連付けられているプロジェクトを選択します。Bitbucket Webhook プロジェクトの

作成の詳細については、「<u>the section called "ステップ 1: Bitbucket を使用してビルドプロジェ</u> クトを作成し、ウェブフックを有効化"」を参照してください。

- 3. プロジェクトの Bitbucket リポジトリのコードに一部変更を加えます。
- Bitbucket リポジトリにプルリクエストを作成します。詳細については、「<u>プルリクエストを行</u> う」を参照してください。
- 5. Bitbucket ウェブフックページで、[View request (リクエストの表示)] を選択して最新イベントの リストを表示します。
- 6. [View details] (詳細の表示) を選択して、CodeBuild より返されるレスポンスに関する詳細を表示 します。次のように表示されます。

"response":"Webhook received and build started: https://useast-1.console.aws.amazon.com/codebuild/home..."
"statusCode":200

7. Bitbucket プルリクエストページに移動して、ビルドのステータスを表示します。

CodeBuild の GitHub Enterprise Server サンプルを実行

AWS CodeBuild は、ソースリポジトリとして GitHub Enterprise Server をサポートしています。 このサンプルでは、GitHub Enterprise Server リポジトリが証明書をインストールしている場合 に、CodeBuild をセットアップする方法を示します。また、Webhook を有効にして、GitHub Enterprise Server リポジトリにコード変更がプッシュされるたびに CodeBuild でソースコードを再 ビルドする方法についても示します。

トピック

- 前提条件
- ステップ 1: GitHub Enterprise Server でビルドプロジェクトを作成し、ウェブフックを有効化

前提条件

CodeBuild プロジェクトの個人用アクセストークンを生成する。GitHub Enterprise ユーザーを作成して、このユーザーの個人用アクセストークンを生成することをお勧めします。CodeBuild プロジェクトを作成する際に使用できるように、クリップボードにこれをコピーします。詳細については、GitHub Help ウェブサイトの Creating a personal access token for the command line を参照してください。

個人用アクセストークンを作成するときには、定義にリポジトリスコープを含めてください。

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

🗹 геро	Full control of private repositories
🗹 repo:status	Access commit status
repo_deployment	Access deployment status
🗹 public_repo	Access public repositories

 GitHub Enterprise サーバーから証明書をダウンロードします。CodeBuild は、証明書を使用して 信頼された SSL 接続をリポジトリに作成します。

Linux/macOS クライアント:

のターミナルウィンドウから、以下のコマンドを実行します。

コマンドのプレースホルダを次の値で置き換えます。

HOST GitHub Enterprise Server リポジトリの IP アドレス。

PORTNUMBER。接続するときに使用するポート番号 (たとえば、443)。

folder 証明書をダウンロードしたフォルダ。

filename 証明書ファイルのファイル名。

A Important

証明書を.pem ファイルとして保存します。

Windows クライアント:

ブラウザを使用して GitHub Enterprise Server から証明書をダウンロードします。サイトの証明書 の詳細を表示するには、南京錠アイコンを選択します。証明書をエクスポートする方法について の詳細は、ブラウザのドキュメントを参照してください。

▲ Important

証明書を.pem ファイルとして保存します。

 証明書ファイルを S3 バケットにアップロードします。S3 バケットを作成する方法については、 「<u>S3 バケットを作成する方法</u>」を参照してください。S3 バケットにオブジェクトをアップロー ドする方法については、「バケットにファイルとフォルダをアップロードする方法」を参照して ください。

Note

このバケットは、ビルドと同じ AWS リージョンに存在する必要があります。たとえば、 米国東部 (オハイオ) リージョンでビルドを実行するように CodeBuild に指示している場 合、バケットは米国東部 (オハイオ) リージョンにある必要があります。

ステップ 1: GitHub Enterprise Server でビルドプロジェクトを作成し、ウェブフック を有効化

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
- 4. Source で、Source プロバイダーで GitHub Enterprise Server を選択します。
 - アカウント認証情報の管理を選択し、個人用アクセストークンを選択します。Service で Secrets Manager (推奨)を選択し、シークレットを設定します。次に、、GitHub Enterprise 個人用アクセストークンで、個人用アクセストークンを入力し、保存を選択します。
 - [Repository URL] に、リポジトリへのパス (例: リポジトリの名前) を入力します。
 - ・ [Additional configuration (追加設定)] を展開します。

- [Rebuild every time a code change is pushed to this repository (コード変更がこのリポジトリ にプッシュされるたび再構築)]を選択して、コード変更がこのリポジトリにプッシュされるたび びに再構築します。
- GitHub Enterprise Server プロジェクトリポジトリに接続するときの SSL 警告を無視するに は、[Enable insecure SSL (安全でない SSL を有効にする)] を選択します。

Note

[Enable insecure SSL (セキュアでない SSL を有効にする)] はテストのみに使用する ことが推奨されます。本番環境では使用しないでください。

Source	Add source
Source 1 - Primary	
Source provider	
GitHub Enterprise	
Repository URL	
https:// <host-name>/<user-name>/<repository-name></repository-name></user-name></host-name>	
Disconnect GitHub Enterprise account	
 Additional configuration Git clone depth, Insecure SSL Git clone depth - optional 	
1	
Webhook - optional	
Rebuild every time a code change is pushed to this repository	
Branch filter - optional	
Enter a regular expression	
Insecure SSL - optional Enable this flag to ignore SSL warnings while connecting to project source. Enable insecure SSL	

5. [環境] で以下の操作を行います。

[Environment image (環境イメージ)] で、次のいずれかの操作を行います。

- が管理する Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム、ランタイム (複数可)、イメージ、イメージバージョンから 選択します。利用可能な場合は、[環境タイプ]から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other

registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジスト リ URL)] に *docker repository/docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選 択します。

- プライベート Docker イメージを使用するには、[カスタムイメージ]を選択しま す。[Environment type (環境タイプ)]で、[ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジスト リ)]を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。 認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。
- 6. [Service role (サービスロール)] で、次のいずれかの操作を行います。
 - CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name] に、新しいロールの名前を入力します。
 - CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選 択します。[Role ARN] で、サービスロールを選択します。

1 Note

コンソールでは、ビルドプロジェクトの作成時や更新時に CodeBuild サービスロール も作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できま す。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、 この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロー ルは最大 10 個のビルドプロジェクトで使用できます。

7. [Additional configuration (追加設定)] を展開します。

CodeBuild を VPC と連携させたい場合:

- [VPC] で、CodeBuild が使用する VPC ID を選択します。
- ・ [VPC Subnets (サブネット)] で、CodeBuild が使用するリソースを含むサブネットを選択します。
- [VPC Security groups (VPC セキュリティグループ)] で、CodeBuild が VPC 内のリソースへの アクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「<u>Amazon Virtual Private Cloud AWS CodeBuild で を使用する</u>」を参照して ください。

- 8. [Buildspec] で、次のいずれかを行います。
 - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
 - [ビルドコマンドの挿入]を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

- 9. [アーティファクト]の[タイプ]で、次のいずれかの操作を行います。
 - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)]を選択します。
 - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を 行います。
 - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)]を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
 - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
 - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。 複数の場所の場合は、各場所をコンマで区切ります (例: appspec.yml, target/my-app.jar)。詳細については、「files」で buildspec の構文の説明を参照してください。
- 10. [キャッシュタイプ] で、以下のいずれかを選択します。
 - キャッシュを使用しない場合は、[No cache] を選択します。
 - Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
 - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
 - (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パスの プレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名に似 ています。これにより、バケット内の同じディレクトリにキャッシュを保存できます。

A Important

パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

ローカルキャッシュを使用する場合は、[ローカル]を選択し、ローカルキャッシュモードを1
 つ以上選択します。

Note

Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択す る場合、プロジェクトは権限モードで実行する必要があります。

キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使用 されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定に関 する詳細については、「<u>buildspec の構文</u>」を参照してください。キャッシングの詳細について は、「パフォーマンスを向上させるためのキャッシュビルド」を参照してください。

11. [Create build project (ビルドプロジェクトの作成)] を選択します。ビルドプロジェクトページ で、[Start build (ビルドの開始)] を選択します。

CodeBuild の GitHub プルリクエストとウェブフックフィルタのサンプルを 実行

AWS CodeBuild ソースリポジトリが GitHub の場合、 はウェブフックをサポートします。つまり、 ソースコードが GitHub リポジトリに保存されている CodeBuild ビルドプロジェクトでは、ウェブ フックを使用することで、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を再構築で きます。CodeBuild のサンプルについては、「AWS CodeBuild のサンプル」を参照してください。

Note

Webhook を使用する場合、ユーザーが予期しないビルドをトリガーする可能性があります。 このリスクを軽減するには、「<u>ウェブフック使用のベストプラクティス。</u>」を参照してくだ さい。

トピック

- ステップ 1: GitHub でビルドプロジェクトを作成し、ウェブフックを有効化
- ステップ 2: ウェブフックが有効になっていることを確認

ステップ 1: GitHub でビルドプロジェクトを作成し、ウェブフックを有効化

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- 3. [Create build project (ビルドプロジェクトの作成)] を選択します。
- 4. [Project configuration (プロジェクトの設定)] で、次のようにします。

[Project name] (プロジェクト名)

このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウント ごとに一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理 解できるように、ビルドプロジェクトの説明を任意で指定することもできます。

5. [Source (ソース)] で、次のようにします。

ソースプロバイダー

[GitHub] を選択します。手順に従って GitHub に接続 (または再接続) し、[Authorize (承認)] を選択します。

リポジトリ

[GitHub アカウントのリポジトリ] を選択します。

GitHub リポジトリ

GitHub リポジトリの URL を入力します。

6. [プライマリソース Webhook イベント] で、以下を選択します。

Note

[プライマリソースの Webhook イベント] セクションは、前のステップで [GitHub アカ ウントのリポジトリ] を選択した場合のみに表示されます。

- プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
- 2. [イベントタイプ] から、1 つ以上のイベントを選択します。
- イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルド を開始する] で、1 つ以上のオプションフィルタを追加します。
- イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを 開始しない] で、1 つ以上のオプションフィルタを追加します。
- 5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加] を選択しま す。

GitHub Webhook イベントタイプとフィルターの詳細については、「<u>GitHub ウェブフックイベ</u> <u>ント</u>」を参照してください。

7. [環境]で以下の操作を行います。

環境イメージ

次のいずれかを選択します。

によって管理される Docker イメージを使用するには AWS CodeBuild:

[Managed image (マネージドイメージ)] を選択し、次に [オペレーティングシステム]、[ランタイム]、[イメージ]、および [ランタイムバージョン] で適切な選択を行います。利用可能な場合は、[環境タイプ] から選択します。

別の Docker イメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、 [Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジス トリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository/docker image name* の形式に従って Docker Hub の Docker イメージの 名前とタグを入力します。[Amazon ECR] を選択した場合は、[Amazon ECR repository] (Amazon ECR レポジトリ) および [Amazon ECR image] (Amazon ECR イメージ) を使用 して AWS アカウントの Docker イメージを選択します。

プライベートDockerイメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、 [Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジス トリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必 要があります。詳細については、「 AWS Secrets Manager ユーザーガイド」の<u>「とは</u> AWS Secrets Manager」を参照してください。

サービスロール

次のいずれかを選択します。

- CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選択します。[Role ARN] で、サービスロールを選択します。

Note

コンソールでは、ビルドプロジェクトの作成時や更新時に CodeBuild サービスロー ルも作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用で きます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付け ると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サー ビスロールは最大 10 個のビルドプロジェクトで使用できます。

- 8. [Buildspec] で、次のいずれかを行います。
 - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレ クトリの buildspec.yml を使用します。
 - [ビルドコマンドの挿入]を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

9. [アーティファクト] で、次のようにします。

Туре

次のいずれかを選択します。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)]
 を選択します。
- ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。

- ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
- [Bucket name (バケット名)] で、出力バケットの名前を選択します。
- この手順の前の方で[ビルドコマンドの挿入]を選択した場合は、[出力ファイル]に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります(例: appspec.yml, target/my-app.jar)。詳細については、「files」で <u>buildspec の構文</u>の説明を参照してください。

追加設定

[Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを設定します。

10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

ステップ 2: ウェブフックが有効になっていることを確認

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- 2. ナビゲーションペインで、[Build projects] を選択します。
- 3. 次のいずれかを行ってください。
 - 確認する Webhook を持つビルドプロジェクトのリンクを選択し、[ビルドの詳細] を選択します。
 - 確認する Webhook を持つビルドプロジェクトの横にあるラジオボタンを選択して、[View details] (詳細を表示) を選択後、[ビルドの詳細] タブを選択します。
- 4. [プライマリソース Webhook イベント] で、[Webhook] の URL リンクを選択します。
- 5. GitHub リポジトリの [Settings] (設定) ページの [Webhooks] (ウェブフック) で、[Pull Requests] (プルリクエスト) と [Pushes] (プッシュ) が選択されていることを確認します。
- GitHub プロファイル設定の個人設定、アプリケーション、認可された OAuth アプリで、アプリ ケーションが選択した AWS リージョンへのアクセスを許可されていることを確認できます。

チュートリアル: CodeBuild で証明書ストレージに S3 を使用して Fastlane で Apple コード署名する S3

<u>fastlane</u> は、iOS および Android アプリのベータデプロイとリリースを自動化するための一般的な オープンソース自動化ツールです。スクリーンショットの生成、コード署名の処理、アプリケーショ ンのリリースなど、面倒なタスクをすべて処理します。

前提条件

このチュートリアルを完了するには、まず以下を設定する必要があります。

- ・ AWS アカウント
- Apple 開発者アカウント
- 証明書を保存するための S3 バケット
- ・ プロジェクトに fastlane がインストールされている fastlane をインストールするためのガイド

ステップ 1: ローカルマシンで S3 で Fastlane Match を設定する

<u>Fastlane Match</u>は <u>Fastlane ツール</u>の1つであり、ローカル開発環境と CodeBuild の両方でコード 署名のシームレスな設定を可能にします。Fastlane Match は、すべてのコード署名証明書とプロビ ジョニングプロファイルを Git リポジトリ/S3 バケット/Google クラウドストレージに保存し、必要 に応じて必要な証明書とプロファイルをダウンロードしてインストールします。

この例では、Amazon S3 バケットをストレージ用にセットアップして使用します。

1. プロジェクトで一致を初期化します。

fastlane match init

- 2. プロンプトが表示されたら、ストレージモードとして S3 を選択します。
- 3. 「Matchfile」を更新して S3 を使用します。

```
storage_mode("s3")
    s3_bucket("your-s3-bucket-name")
    s3_region("your-aws-region")
    type("appstore") # The default type, can be: appstore, adhoc, enterprise or
    development
```

ステップ 2: Fastfile を設定する

次のレーンで `Fastfile` を作成または更新します。

CodeBuild では、アプリを構築して署名するたびに Fastlane Match を実行する必要があります。こ れを行う最も簡単な方法は、アプリを構築するレーンに matchアクションを追加することです。

```
default_platform(:ios)

platform :ios do
   before_all do
     setup_ci
   end

   desc "Build and sign the app"
   lane :build do
     match(type: "appstore", readonly: true)
     gym(
        scheme: "YourScheme",
        export_method: "app-store"
     )
   end
end
```

Note

ー致アクションが正しく機能するには、 の setup_cibefore_all セクショ ンFastfileに を追加してください。これにより、適切なアクセス許可を持つ一時的な Fastlane キーチェーンが使用されます。これを使用しないと、ビルドの失敗や一貫性のない 結果が表示されることがあります。

ステップ 3: fastlane match コマンドを実行して、それぞれの証明書とプロファイ ルを生成する

特定のタイプ (開発、アプリストア、アドホック、エンタープライズなど) の fastlane マッチコマン ドは、リモートストアで使用できない場合に証明書とプロファイルを生成します。証明書とプロファ イルは fastlane によって S3 に保存されます。

bundle exec fastlane match appstore

コマンドの実行はインタラクティブになり、fastlane は証明書を復号するためのパスフレーズを設定 するよう に要求します。

ステップ 4: プロジェクトのアプリケーションファイルを作成する

プロジェクトに応じてアプリケーションファイルを作成または追加します。

- 1. プロジェクトのビルド要件に基づいて、<u>Gymfile</u>、<u>Appfile</u>、<u>Snapfile</u>、<u>Deliverfile</u> を作成または追 加します。
- 2. リモートリポジトリに変更をコミットする

ステップ 5: Secrets Manager で環境変数を作成する

Fastlane セッション Cookie と一致するパスフレーズを保存するための 2 つのシークレットを作成 します。Secrets Manager でのシークレットの作成の詳細については、「 シーク<u>レットの作成 AWS</u> Secrets Manager」を参照してください。

- 1. 次のように fastlane セッション Cookie にアクセスします。
 - a. シークレットキー FASTLANE_SESSION
 - b. シークレット値 ローカルマシンで次のコマンドを実行することで生成されたセッション Cookie。

③ Note この値は、ローカルファイルの認証後に使用できます: ~/.fastlane/ spaceship/my_appleid_username/cookie。

fastlane spaceauth -u <apple account>

- Fastlane Match パスフレーズ Fastlane Match が S3 バケットに保存されている証明書とプロ ファイルを復号できるようにするには、Match セットアップステップで設定した暗号化パスフ レーズを CodeBuild プロジェクトの環境変数に追加する必要があります。
 - a. シークレットキー MATCH_PASSWORD

Note

Secrets Manager で上記のシークレットを作成するときは、シークレット名に次のプレ フィックスを付けてください。 /CodeBuild/

ステップ 6: コンピューティングフリートを作成する

プロジェクトのコンピューティングフリートを作成します。

- 1. コンソールで、CodeBuild に移動し、新しいコンピューティングフリートを作成します。
- オペレーティングシステムとして「macOS」を選択し、適切なコンピューティングタイプとイメージを選択します。

ステップ 7: CodeBuild でプロジェクトを作成する

CodeBuild でプロジェクトを作成します。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- 2. ビルドプロジェクトを作成します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u>ル)」および「ビルドの実行 (コンソール)」を参照してください。
- ソースプロバイダー (GitHub、CodeCommit など) を設定します。これは iOS プロジェクトソー スリポジトリであり、証明書リポジトリではありません。
- 4. [環境] で以下の操作を行います。
 - リザーブドキャパシティを選択します。
 - フリート で、上記で作成したフリートを選択します。
 - CodeBuild が作成するサービスロールの名前を指定します。
 - 以下の環境変数を指定します。

- 名前: MATCH_PASSWORD、値: <secrets arn>、タイプ: Secrets Manager (ステップ 5 で MATCH_PASSWORD 用に作成されたシークレット ARN)
- 名前: FASTLANE_SESSION、値: *secrets arn*>、タイプ: Secrets Manager (FASTLANE_SESSION のステップ 5 で作成されたシークレット ARN)
- 5. Buildspec で、以下を追加します。

```
version: 0.2
phases:
 install:
    commands:
      - gem install bundler
      - bundle install
 build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
 post_build:
    commands:
      - echo "Build completed on date"
artifacts:
 files:
   - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

ステップ 8: IAM ロールを設定する

プロジェクトが作成されたら、CodeBuild プロジェクトのサービスロールに、証明書を含む S3 バ ケットへのアクセス許可があることを確認します。ロールに次のポリシーを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:GetBucketLocation",
               "s3:ListBucket"
        ],
    }
}
```

```
"Resource": "arn:aws:s3:::your-s3-bucket-name"
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
    ],
    "Resource": "arn:aws:s3:::your-s3-bucket-name/*"
}
]
```

ステップ 9: ビルドを実行する

ビルドを実行します。CodeBuild でビルドステータスとログを確認できます。

ジョブが完了すると、ジョブのログを表示できるようになります。

トラブルシューティング

- 証明書の取得で問題が発生した場合は、IAM アクセス許可が S3 アクセス用に正しく設定されていることを確認します。
- 証明書の復号で問題が発生した場合は、MATCH_PASSWORD 環境変数で正しいパスフレーズを 設定してください。
- コード署名の問題については、Apple Developer アカウントに必要な証明書とプロファイルがあり、Xcode プロジェクトのバンドル識別子がプロビジョニングプロファイルのものと一致することを確認します。

セキュリティに関する考慮事項

このチュートリアルのセキュリティ上の考慮事項を次に示します。

- S3 バケットに、保管時の暗号化など、適切なセキュリティ設定があることを確認します。特に、 バケットにパブリックアクセスがないことを確認し、アクセスが必要な CodeBuild とシステムの みにアクセスを制限します。
- MATCH_PASSWORD や FASTLANE_SESSION などの機密情報を保存 AWS Secrets Manager す るために を使用することを検討してください。

このサンプルでは、証明書ストレージに Amazon S3 を使用して CodeBuild の Fastlane で iOS コー ド署名をセットアップします。特定のプロジェクト要件と CodeBuild 環境に基づいて、いくつかの ステップを調整する必要がある場合があります。このアプローチでは、 AWS サービスを活用して AWS 、エコシステム内のセキュリティと統合を強化します。

チュートリアル: 証明書ストレージに GitHub を使用した CodeBuild での Fastlane による Apple コード署名

<u>fastlane</u> は、iOS および Android アプリのベータデプロイとリリースを自動化するための一般的な オープンソース自動化ツールです。スクリーンショットの生成、コード署名の処理、アプリケーショ ンのリリースなど、面倒なタスクをすべて処理します。

このサンプルでは、Mac フリートで実行されている CodeBuild プロジェクトで Fastlane を使用して Apple コード署名を設定する方法を示します。GitHub は証明書とプロビジョニングプロファイルの ストレージです。

前提条件

このチュートリアルを完了するには、まず以下を設定する必要があります。

- ・ AWS アカウント
- Apple 開発者アカウント
- 証明書を保存するためのプライベート GitHub リポジトリ
- ・ プロジェクトに fastlane がインストールされている fastlane をインストールするためのガイド

ステップ 1: ローカルマシンで GitHub で Fastlane Match を設定する

<u>Fastlane Match</u>は <u>Fastlane ツール</u>の1つであり、ローカル開発環境と CodeBuild の両方でコード 署名のシームレスな設定を可能にします。Fastlane Match は、すべてのコード署名証明書とプロビ ジョニングプロファイルを Git リポジトリ/S3 バケット/Google クラウドストレージに保存し、必要 に応じて必要な証明書とプロファイルをダウンロードしてインストールします。

この例では、ストレージに Git リポジトリをセットアップして使用します。

1. プロジェクトで一致を初期化します。

fastlane match init

- 2. プロンプトが表示されたら、ストレージモードとして GitHub を選択します。
- 3. GitHub を使用するように「Matchfile」を更新します。

```
git_url("https://github.com/your-username/your-certificate-repo.git")
storage_mode("git")
type("development") # The default type, can be: appstore, adhoc, enterprise or
    development
```

1 Note

fastlane が正常に認証およびクローンを作成するには、必ず Git リポジトリの HTTPS URL を入力してください。それ以外の場合は、一致を使用しようとすると認証エラーが表示され ることがあります。

ステップ 2: Fastfile をセットアップする

次のレーンで「Fastfile」を作成または更新します。

CodeBuild では、アプリを構築して署名するたびに Fastlane Match を実行する必要があります。こ れを行う最も簡単な方法は、アプリを構築するレーンに matchアクションを追加することです。

```
default_platform(:ios)

platform :ios do
   before_all do
      setup_ci
   end

   desc "Build and sign the app"
   lane :build do
      match(type: "appstore", readonly: true)
      gym(
        scheme: "YourScheme",
        export_method: "app-store"
      )
      end
end
```

(i) Note

ー致アクションが正しく機能するには、 の setup_cibefore_all セクショ ンFastfileに を追加してください。これにより、適切なアクセス許可を持つ一時的な Fastlane キーチェーンが使用されます。これを使用しないと、ビルドの失敗や一貫性のない 結果が表示されることがあります。

ステップ 3: fastlane match コマンドを実行して、それぞれの証明書とプロファイ ルを生成する

特定のタイプ (開発、アプリストア、アドホック、エンタープライズなど) の fastlane マッチコマン ドは、リモートストアで使用できない場合、証明書とプロファイルを生成します。証明書とプロファ イルは fastlane によって GitHub に保存されます。

bundle exec fastlane match appstore

コマンドの実行はインタラクティブになり、fastlane は証明書を復号するためのパスフレーズを設定 するよう に要求します。

ステップ 4: プロジェクトのアプリケーションファイルを作成する

プロジェクトに応じてアプリケーションファイルを作成または追加します。

- プロジェクトのビルド要件に基づいて、<u>Gymfile</u>、<u>Appfile</u>、<u>Snapfile</u>、<u>Deliverfile</u>を作成または追加します。
- 2. リモートリポジトリに変更をコミットします。

ステップ 5: Secrets Manager で環境変数を作成する

Fastlane セッション Cookie と一致するパスフレーズを保存するための 3 つのシークレットを作成 します。Secrets Manager でのシークレットの作成の詳細については、「 シーク<u>レットの作成 AWS</u> <u>Secrets Manager」</u>を参照してください。

- 1. 次のように fastlane セッション Cookie にアクセスします。
 - a. シークレットキー FASTLANE_SESSION

b. シークレット値 - ローカルマシンで次のコマンドを実行することで生成されたセッション Cookie。

Note この値は、ローカルファイルの認証後に使用できます: ~/.fastlane/ spaceship/my_appleid_username/cookie。

fastlane spaceauth -u <Apple_account>

- Fastlane Match パスフレーズ Fastlane Match が Git リポジトリに保存されている証明書とプロ ファイルを復号できるようにするには、Match setup ステップで設定した暗号化パスフレーズを CodeBuild プロジェクトの環境変数に追加する必要があります。
 - a. シークレットキー MATCH_PASSWORD
 - b. シークレット値 <match passphrase to decrypt certificates>。パスフレーズは、ステップ3で証明書を生成するときに設定されます。
- 3. Fastlane MATCH_GIT_BASIC_AUTHORIZATION 一致の基本的な認可を設定します。
 - a. シークレットキー :

MATCH_GIT_BASIC_AUTHORIZATION

 b. シークレット値 - 値は、ユーザー名と個人用アクセストークン (PAT) の base64 でエンコー ドされた文字列で、の形式である必要がありますusername:password。次のコマンドを 使用して生成できます。

echo -n your_github_username:your_personal_access_token | base64

PAT は、プロファイル > 設定 > 開発者設定 > 個人用アクセストークンの GitHub コ ンソールで生成できます。詳細については、次のガイドを参照してください: <u>https://</u> <u>docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-</u> <u>personal-access-tokens</u>:」。
Note

Secrets Manager で上記のシークレットを作成するときは、次のプレフィックスが付いた シークレット名を付けることを忘れないでください。 /CodeBuild/

ステップ 6: コンピューティングフリートを作成する

プロジェクトのコンピューティングフリートを作成します。

- 1. コンソールで、CodeBuild に移動し、新しいコンピューティングフリートを作成します。
- オペレーティングシステムmacOSとして を選択し、適切なコンピューティングタイプとイメージを選択します。

ステップ 7: CodeBuild でプロジェクトを作成する

CodeBuild でプロジェクトを作成します。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。
- 3. ソースプロバイダー (GitHub、CodeCommit など) を設定します。これは iOS プロジェクトソー スリポジトリであり、証明書リポジトリではありません。
- 4. [環境] で以下の操作を行います。
 - リザーブドキャパシティを選択します。
 - フリート で、上記で作成したフリートを選択します。
 - CodeBuild が作成するサービスロールの名前を指定します。
 - ・以下の環境変数を指定します。
 - 名前: MATCH_PASSWORD、値: <secrets arn>、タイプ: Secrets Manager (ステップ 5 で MATCH_PASSWORD 用に作成されたシークレット ARN)
 - 名前: FASTLANE_SESSION、値: <secrets arn>、タイプ: Secrets Manager (FASTLANE_SESSION のステップ 5 で作成されたシークレット ARN)

- 名前: MATCH_GIT_BASIC_AUTHORIZATION、値: <secrets ARN>、タイプ: Secrets Manager Secrets ARN(のステップ 5 で作成MATCH_GIT_BASIC_AUTHORIZATION)
- 5. Buildspec で、以下を追加します。

```
version: 0.2
phases:
 install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"
artifacts:
 files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

ステップ 8: ビルドを実行する

ビルドを実行します。CodeBuild でビルドステータスとログを確認できます。

ジョブが完了すると、ジョブのログを表示できるようになります。

トラブルシューティング

- GitHub リポジトリへのアクセスで問題が発生した場合は、個人用アクセストークンと MATCH_GIT_BASIC_AUTHORIZATION 環境変数を再確認してください。
- 証明書の復号化で問題が発生した場合は、MATCH_PASSWORD 環境変数で正しいパスフレーズ を設定してください。
- コード署名の問題については、Apple Developer アカウントに必要な証明書とプロファイルがあり、Xcode プロジェクトのバンドル識別子がプロビジョニングプロファイルのものと一致することを確認します。

セキュリティに関する考慮事項

このチュートリアルのセキュリティ上の考慮事項を次に示します。

- 証明書の GitHub リポジトリを非公開にし、定期的にアクセスを監査します。
- MATCH_PASSWORD や FASTLANE_SESSION などの機密情報を保存 AWS Secrets Manager す るために を使用することを検討してください。

このサンプルでは、証明書ストレージに GitHub を使用して CodeBuild の Fastlane で iOS コード 署名をセットアップします。特定のプロジェクト要件と CodeBuild 環境に基づいて、いくつかのス テップを調整する必要がある場合があります。このアプローチでは、 AWS サービスを活用して、 AWS エコシステム内のセキュリティと統合を強化します。

セマンティックバージョニングを使用してビルド時にアーティファ クト名を設定

このサンプルには、ビルド時に作成するアーティファクト名の指定方法を示す buildspec ファイルの サンプルが含まれています。buildspec ファイルで指定される名前には、シェルコマンドと環境変数 を組み込んで、一意の名前にすることができます。buildspec で指定した名前は、プロジェクトの作 成時にコンソールに入力した名前よりも優先されます。

複数回ビルドする場合、buildspec ファイルで指定されたアーティファクト名を使用すると、出力 アーティファクトファイル名が一意であることが保証されます。たとえば、ビルド時にアーティファ クト名に日付とタイムスタンプを挿入できます。

コンソールで入力したアーティファクト名を buildspec ファイルの名前で上書きする場合は、次のよ うにします。

- ビルドプロジェクトを設定して、アーティファクト名を buildspec ファイル内の名前で上書きします。
 - コンソールを使用してビルドプロジェクトを作成する場合は、[Enable semantic versioning (セマンティックバージョニングを有効にする)]を選択します。詳細については、「ビルドプ ロジェクトの作成 (コンソール)」を参照してください。
 - を使用する場合は AWS CLI、 overrideArtifactNameに渡された JSON 形式のファイル でを true に設定しますcreate-project。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

- AWS CodeBuild API を使用する場合は、プロジェクトの作成または更新時、またはビルドの 開始時に、 ProjectArtifacts オブジェクトに overrideArtifactNameフラグを設定し ます。
- buildspec ファイルに名前を指定します 次のサンプルの buildspec ファイルを参考として使用してください。

この Linux の例は、ビルドが作成された日付を含むアーティファクト名を指定する方法を示していま す。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
    files:
            - '**/*'
    name: myname-$(date +%Y-%m-%d)
```

この Linux の例は、CodeBuild 環境変数を使用するアーティファクト名を指定する方法を示していま す。詳細については、「ビルド環境の環境変数」を参照してください。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
    files:
            - '**/*'
    name: myname-$AWS_REGION
```

この Windows の例は、ビルドが作成された日時を含むアーティファクト名を指定する方法を示して います。

```
version: 0.2
env:
variables:
TEST_ENV_VARIABLE: myArtifactName
phases:
```

build:		
commands:		
 cd samples/helloworld 		
- dotnet restore		
- dotnet run		
artifacts:		
files:		
- '**/*'		
<pre>name: \$Env:TEST_ENV_VARIABLE-\$(Get-Date</pre>	-UFormat	"%Y%m%d-%H%M%S")

この Windows の例は、buildspec ファイルで宣言された変数と CodeBuild 環境変数を使用するアー ティファクト名を指定する方法を示しています。詳細については、「<u>ビルド環境の環境変数</u>」を参照 してください。

```
version: 0.2
env:
   variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
   build:
    commands:
        - cd samples/helloworld
        - dotnet restore
        - dotnet run
artifacts:
   files:
        - '**/*'
   name: $Env:TEST_ENV_VARIABLE-$Env:AWS_REGION
```

詳細については、「<u>CodeBuild のビルド仕様に関するリファレンス</u>」を参照してください。

CodeBuild の Microsoft Windows サンプルを実行

これらのサンプルでは、Microsoft Windows Server 2019、.NET Framework、および .NET Core SDK を実行する AWS CodeBuild ビルド環境を使用して、F# と Visual Basic で記述されたコードからラ ンタイムファイルをビルドします。

▲ Important

これらのサンプルを実行すると、AWS アカウントに料金が発生する可能性があります。こ れには、Amazon S3、および CloudWatch Logs に関連する AWS リソースとアクションに 対する CodeBuild AWS KMSと の料金が含まれます。詳細については、「[g465]CodeBuild 料金表[/g465]」、「[g464]Amazon S3 料金表[/g464]」、「[g463]AWS Key Management Service 料金表[/g463]」、および「[g462]Amazon CloudWatch 料金表[/g462]」を参照してく ださい。

Windows サンプルを実行

Windows サンプルを実行するには、次の手順に従います。

Windows サンプルを実行するには

 このトピックの「<u>ディレクトリ構造</u>」セクションと「<u>ファイル</u>」セクションで説明しているファ イルを作成し、これらのファイルをS3入力バケット、CodeCommit または GitHub のリポジト リにアップロードします。

A Important

(root directory name)をアップロードしないでください。アップロードするの は、(root directory name)内のファイルのみです。 S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから 入力バケットにアップロードしてください。(root directory name)を ZIP ファイ ルに追加しないでください。追加するのは、(root directory name)内のファイル のみです。

 ビルドプロジェクトを作成します。ビルドプロジェクトは、mcr.microsoft.com/dotnet/ framework/sdk:4.8 イメージを使用して、.NET Framework プロジェクトをビルドします。 を使用してビルドプロジェクト AWS CLI を作成する場合、create-projectコマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-windows-build-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/windows-build-input-
artifact.zip"
 },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "windows-build-output-artifact.zip"
 },
  "environment": {
    "type": "WINDOWS_SERVER_2019_CONTAINER",
    "image": "mcr.microsoft.com/dotnet/framework/sdk:4.8",
    "computeType": "BUILD_GENERAL1_MEDIUM"
 },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

- 3. ビルドを実行し、「ビルドを手動で実行」の手順を実行します。
- ビルド出力のアーティファクトを取得するには、S3 出力バケットで、windows-buildoutput-artifact.zip ファイルをローカルコンピュータまたはインスタンスにダウンロード します。コンテンツを抽出し、ランタイムおよび他のファイルにアクセスします。
 - .NET Framework を使用する F# サンプルのランタイム (FSharpHelloWorld.exe)
 は、FSharpHelloWorld\bin\Debug ディレクトリにあります。
 - .NET Framework を使用する Visual Basic サンプルランタイムファイル (VBHelloWorld.exe)は、VBHelloWorld\bin\Debug ディレクトリにあります。

ディレクトリ構造

これらのサンプルで想定しているディレクトリ構造は以下のとおりです。

F#と.NET Framework

roj

Visual Basic と .NET Framework

```
(root directory name)
### buildspec.yml
### VBHelloWorld.sln
### VBHelloWorld
### App.config
### HelloWorld.vb
### VBHelloWorld.vbproj
### My Project
### Application.Designer.vb
### Application.myapp
### AssemblyInfo.vb
### Resources.Designer.vb
### Settings.Designer.vb
### Settings.settings
```

ファイル

これらのサンプルでは、以下のファイルを使用します。

F#と.NET Framework

buildspec.yml((root directory name)内)

version: 0.2

env:

F#と.NET Framework

```
variables:
SOLUTION: .\FSharpHelloWorld.sln
PACKAGE_DIRECTORY: .\packages
DOTNET_FRAMEWORK: 4.8
phases:
build:
    commands:
        - '& nuget restore $env:SOLUTION -PackagesDirectory $env:PACKAGE_DIRECTORY'
        - '& msbuild -p:FrameworkPathOverride="C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
    files:
        - .\FSharpHelloWorld\bin\Debug\*
```

FSharpHelloWorld.sln(*(root directory name)*内)

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F2A71F9B-5D33-465A-A702-920D77279786}") = "FSharpHelloWorld",
 "FSharpHelloWorld\FSharpHelloWorld.fsproj", "{D60939B6-526D-43F4-9A89-577B2980DF62}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release Any CPU = Release Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release Any CPU.ActiveCfg = Release Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release Any CPU.Build.0 = Release Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config((root directory name)\FSharpHelloWorld内)

```
<?xml version="1.0" encoding="utf-8" ?>
```

F#と.NET Framework

```
<configuration>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
</startup>
</configuration>
```

AssemblyInfo.fs((*root directory name*)\FSharpHelloWorld内)

```
namespace FSharpHelloWorld.AssemblyInfo
open System.Reflection
open System.Runtime.CompilerServices
open System.Runtime.InteropServices
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[<assembly: AssemblyTitle("FSharpHelloWorld")>]
[<assembly: AssemblyDescription("")>]
[<assembly: AssemblyConfiguration("")>]
[<assembly: AssemblyCompany("")>]
[<assembly: AssemblyProduct("FSharpHelloWorld")>]
[<assembly: AssemblyCopyright("Copyright © 2017")>]
[<assembly: AssemblyTrademark("")>]
[<assembly: AssemblyCulture("")>]
// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[<assembly: ComVisible(false)>]
// The following GUID is for the ID of the typelib if this project is exposed to COM
[<assembly: Guid("d60939b6-526d-43f4-9a89-577b2980df62")>]
// Version information for an assembly consists of the following four values:
11
// Major Version
// Minor Version
// Build Number
// Revision
11
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
```

```
// [<assembly: AssemblyVersion("1.0.*")>]
[<assembly: AssemblyVersion("1.0.0.0")>]
[<assembly: AssemblyFileVersion("1.0.0.0")>]
do
   ()
```

FSharpHelloWorld.fsproj((*root directory name)*\FSharpHelloWorld内)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\</pre>
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>d60939b6-526d-43f4-9a89-577b2980df62</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>FSharpHelloWorld</RootNamespace>
    <AssemblyName>FSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    <TargetFSharpCoreVersion>4.4.0.0</TargetFSharpCoreVersion>
    <Name>FSharpHelloWorld</Name>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <Tailcalls>false</Tailcalls>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Debug\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
```

<Optimize>true</Optimize> <Tailcalls>true</Tailcalls> <OutputPath>bin\Release\</OutputPath> <DefineConstants>TRACE</DefineConstants> <WarningLevel>3</WarningLevel> <PlatformTarget>AnyCPU</PlatformTarget> <DocumentationFile>bin\Release\FSharpHelloWorld.XML</DocumentationFile> <Prefer32Bit>true</Prefer32Bit> </PropertyGroup> <ItemGroup> <Reference Include="mscorlib" /> <Reference Include="FSharp.Core, Version=\$(TargetFSharpCoreVersion), Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"> <Private>True</Private> </Reference> <Reference Include="System" /> <Reference Include="System.Core" /> <Reference Include="System.Numerics" /> </ItemGroup> <ItemGroup> <Compile Include="AssemblyInfo.fs" /> <Compile Include="Program.fs" /> <None Include="App.config" /> </ItemGroup> <PropertyGroup> <MinimumVisualStudioVersion Condition="'\$(MinimumVisualStudioVersion)' == ''">11</ MinimumVisualStudioVersion> </PropertyGroup> <Choose> <When Condition="'\$(VisualStudioVersion)' == '11.0'"> <propertyGroup Condition="Exists('\$(MSBuildExtensionsPath32)\...\Microsoft SDKs\F#</pre> \3.0\Framework\v4.0\Microsoft.FSharp.Targets')"> <FSharpTargetsPath>\$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F# \3.0\Framework\v4.0\Microsoft.FSharp.Targets</FSharpTargetsPath> </PropertyGroup> </When> <Otherwise> <propertyGroup Condition="Exists('\$(MSBuildExtensionsPath32)\Microsoft</pre> \VisualStudio\v\$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets')"> <FSharpTargetsPath>\$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v \$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets</FSharpTargetsPath> </PropertyGroup> </Otherwise> </Choose>

```
<Import Project="$(FSharpTargetsPath)" />
<!-- To modify your build process, add your task inside one of the targets below and
uncomment it.
        Other similar extension points exist, see Microsoft.Common.targets.
<Target Name="BeforeBuild">
        </Target>
        <target</target>
        <target>
        </target>
        <target>
        </target>
        <target>
        </target>
        </target>
        <target>
        </target>
        <target>
        </target>
        <target>
        <target>
        <target>
        </target>
        <target>
        </target>
        <target>
        <target>
        <target>
        <target>
        <target>
        <target>
        </target>
        </target>
        <target>
        <tar
```

Program.fs(内)(root directory name)\FSharpHelloWorld

```
// Learn more about F# at http://fsharp.org
// See the 'F# Tutorial' project for more help.
[<EntryPoint>]
let main argv =
   printfn "Hello World"
   0 // return an integer exit code
```

Visual Basic と .NET Framework

```
buildspec.yml((root directory name)内)
```

```
version: 0.2
env:
  variables:
    SOLUTION: .\VBHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8
phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -
PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -
p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft
\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
```

```
VBHelloWorld.sln((root directory name)内)
```

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "VBHelloWorld", "VBHelloWorld
\VBHelloWorld.vbproj", "{4DCEC446-7156-4FE6-8CCC-219E34DD409D}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release | Any CPU = Release | Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release Any CPU.ActiveCfg = Release Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config((root directory name)\VBHelloWorld内)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
</startup>
</configuration>
```

HelloWorld.vb((*root directory name*)\VBHelloWorld内)

```
Module HelloWorld
Sub Main()
MsgBox("Hello World")
```

End Sub

End Module

VBHelloWorld.vbproj((*root directory name*)\VBHelloWorld内)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\</pre>
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{4DCEC446-7156-4FE6-8CCC-219E34DD409D}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <StartupObject>VBHelloWorld.HelloWorld</StartupObject>
    <RootNamespace>VBHelloWorld</RootNamespace>
    <AssemblyName>VBHelloWorld</AssemblyName>
    <FileAlignment>512</FileAlignment>
    <MyType>Console</MyType>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <DefineDebug>true</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <OutputPath>bin\Debug\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <propertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <DefineDebug>false</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
```

```
<DocumentationFile>VBHelloWorld.xml</DocumentationFile>
  <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
</PropertyGroup>
<PropertyGroup>
  <OptionExplicit>On</OptionExplicit>
</PropertyGroup>
<PropertyGroup>
  <OptionCompare>Binary</OptionCompare>
</PropertyGroup>
<PropertyGroup>
  <OptionStrict>Off</OptionStrict>
</PropertyGroup>
<PropertyGroup>
  <OptionInfer>On</OptionInfer>
</PropertyGroup>
<ItemGroup>
  <Reference Include="System" />
  <Reference Include="System.Data" />
  <Reference Include="System.Deployment" />
  <Reference Include="System.Xml" />
  <Reference Include="System.Core" />
  <Reference Include="System.Xml.Ling" />
  <Reference Include="System.Data.DataSetExtensions" />
  <Reference Include="System.Net.Http" />
</ItemGroup>
<ItemGroup>
  <Import Include="Microsoft.VisualBasic" />
  <Import Include="System" />
  <Import Include="System.Collections" />
  <Import Include="System.Collections.Generic" />
  <Import Include="System.Data" />
  <Import Include="System.Diagnostics" />
  <Import Include="System.Ling" />
  <Import Include="System.Xml.Linq" />
  <Import Include="System.Threading.Tasks" />
</ItemGroup>
<ItemGroup>
  <Compile Include="HelloWorld.vb" />
  <Compile Include="My Project\AssemblyInfo.vb" />
  <Compile Include="My Project\Application.Designer.vb">
    <AutoGen>True</AutoGen>
    <DependentUpon>Application.myapp</DependentUpon>
  </Compile>
  <Compile Include="My Project\Resources.Designer.vb">
```

<autogen>True</autogen>
<designtime>True</designtime>
<dependentupon>Resources.resx</dependentupon>
<compile include="My Project\Settings.Designer.vb"></compile>
<autogen>True</autogen>
<dependentupon>Settings.settings</dependentupon>
<designtimesharedinput>True</designtimesharedinput>
<itemgroup></itemgroup>
<embeddedresource include="My Project\Resources.resx"></embeddedresource>
<generator>VbMyResourcesResXFileCodeGenerator</generator>
<lastgenoutput>Resources.Designer.vb</lastgenoutput>
<customtoolnamespace>My.Resources</customtoolnamespace>
<subtype>Designer</subtype>
<itemgroup></itemgroup>
<none include="My Project\Application.myapp"></none>
<generator>MyApplicationCodeGenerator</generator>
<lastgenoutput>Application.Designer.vb</lastgenoutput>
<none include="My Project\Settings.settings"></none>
<generator>SettingsSingleFileGenerator</generator>
<customtoolnamespace>My</customtoolnamespace>
<lastgenoutput>Settings.Designer.vb</lastgenoutput>
<none include="App.config"></none>
<import project="\$(MSBuildToolsPath)\Microsoft.VisualBasic.targets"></import>
To modify your build process, add your task inside one of the targets below and</td
uncomment it.
Other similar extension points exist, see Microsoft.Common.targets.
<target name="BeforeBuild"></target>
<target name="AfterBuild"></target>
>

Application.Designer.vb((*root directory name*)\VBHelloWorld\My Project内)

```
' ------
' <auto-generated>
' This code was generated by a tool.
' Runtime Version:4.0.30319.42000
'
' Changes to this file may cause incorrect behavior and will be lost if
' the code is regenerated.
' </auto-generated>
'
Option Strict On
Option Explicit On
```

Application.myapp((root directory name)\VBHelloWorld\My Project内)

```
<?xml version="1.0" encoding="utf-8"?>
<MyApplicationData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<MySubMain>false</MySubMain>
<SingleInstance>false</SingleInstance>
<ShutdownMode>0</ShutdownMode>
<EnableVisualStyles>true</EnableVisualStyles>
<AuthenticationMode>0</AuthenticationMode>
<ApplicationType>2</ApplicationType>
<SaveMySettingsOnExit>true</SaveMySettingsOnExit>
</MyApplicationData>
```

AssemblyInfo.vb((*root directory name*)\VBHelloWorld\My Project内)

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices
' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.
' Review the values of the assembly attributes
<Assembly: AssemblyTitle("VBHelloWorld")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
```

```
<Assembly: AssemblyProduct("VBHelloWorld")>
<Assembly: AssemblyCopyright("Copyright © 2017")>
<Assembly: AssemblyTrademark("")>
<Assembly: ComVisible(False)>
'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("137c362b-36ef-4c3e-84ab-f95082487a5a")>
' Version information for an assembly consists of the following four values:
' Major Version
' Minor Version
' Build Number
' Revision
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>
<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

Resources.Designer.vb((*root directory name*)\VBHelloWorld\My Project内)

```
'with the /str option, or rebuild your VS project.
 '''<summary>
 ''' A strongly-typed resource class, for looking up localized strings, etc.
 '''</summary>
<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedRe
"4.0.0.0"), _
 Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
 Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
 Global.Microsoft.VisualBasic.HideModuleNameAttribute()> _
 Friend Module Resources
   Private resourceMan As Global.System.Resources.ResourceManager
   Private resourceCulture As Global.System.Globalization.CultureInfo
   '''<summary>
   ''' Returns the cached ResourceManager instance used by this class.
   '''</summary>
<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
   Friend ReadOnly Property ResourceManager() As
Global.System.Resources.ResourceManager
     Get
       If Object.ReferenceEquals(resourceMan, Nothing) Then
         Dim temp As Global.System.Resources.ResourceManager = New
Global.System.Resources.ResourceManager("VBHelloWorld.Resources",
GetType(Resources).Assembly)
         resourceMan = temp
       End If
       Return resourceMan
     End Get
   End Property
   '''<summary>
   ''' Overrides the current thread's CurrentUICulture property for all
   ''' resource lookups using this strongly typed resource class.
   '''</summary>
<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
   Friend Property Culture() As Global.System.Globalization.CultureInfo
     Get
```

```
Return resourceCulture
End Get
Set(ByVal value As Global.System.Globalization.CultureInfo)
resourceCulture = value
End Set
End Property
End Module
End Namespace
```

```
Resources.resx((root directory name)\VBHelloWorld\My Project内)
```

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <!--
   Microsoft ResX Schema
   Version 2.0
   The primary goals of this format is to allow a simple XML format
   that is mostly human readable. The generation and parsing of the
   various data types are done through the TypeConverter classes
    associated with the data types.
    Example:
    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader,
System.Windows.Forms, ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter,
System.Windows.Forms, ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
      <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing"</pre>
mimetype="application/x-microsoft.net.object.bytearray.base64">
      <value>[base64 mime encoded string representing a byte array form of the .NET
Framework object]</value>
      <comment>This is a comment</comment>
```

</data> There are any number of "resheader" rows that contain simple name/value pairs. Each data row contains a name, and value. The row also contains a type or mimetype. Type corresponds to a .NET class that support text/value conversion through the TypeConverter architecture. Classes that don't support this are serialized and stored with the mimetype set. The mimetype is used for serialized objects, and tells the ResXResourceReader how to depersist the object. This is currently not extensible. For a given mimetype the value must be set accordingly: Note - application/x-microsoft.net.object.binary.base64 is the format that the ResXResourceWriter will generate, however the reader can read any of the formats listed below. mimetype: application/x-microsoft.net.object.binary.base64 value : The object must be serialized with : System.Serialization.Formatters.Binary.BinaryFormatter : and then encoded with base64 encoding. mimetype: application/x-microsoft.net.object.soap.base64 value : The object must be serialized with : System.Runtime.Serialization.Formatters.Soap.SoapFormatter : and then encoded with base64 encoding. mimetype: application/x-microsoft.net.object.bytearray.base64 : The object must be serialized into a byte array value : using a System.ComponentModel.TypeConverter : and then encoded with base64 encoding. --> <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"</pre> xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"> <xsd:element name="root" msdata:IsDataSet="true"> <re><xsd:complexType> <rsd:choice maxOccurs="unbounded"> <xsd:element name="metadata"> <re><xsd:complexType> <xsd:sequence> <xsd:element name="value" type="xsd:string" minOccurs="0" />

```
</xsd:sequence>
```

```
<xsd:attribute name="name" type="xsd:string" />
             <xsd:attribute name="type" type="xsd:string" />
             <xsd:attribute name="mimetype" type="xsd:string" />
           </xsd:complexType>
         </xsd:element>
         <xsd:element name="assembly">
           <re><xsd:complexType>
             <xsd:attribute name="alias" type="xsd:string" />
             <xsd:attribute name="name" type="xsd:string" />
           </xsd:complexType>
         </xsd:element>
         <rpre><xsd:element name="data">
           <re><xsd:complexType>
             <xsd:sequence>
               <xsd:element name="value" type="xsd:string" minOccurs="0"</pre>
msdata:Ordinal="1" />
               <xsd:element name="comment" type="xsd:string" minOccurs="0"</pre>
msdata:Ordinal="2" />
             </xsd:sequence>
             <xsd:attribute name="name" type="xsd:string" msdata:Ordinal="1" />
             <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
             <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
           </xsd:complexType>
         </xsd:element>
         <xsd:element name="resheader">
           <re><xsd:complexType>
             <xsd:sequence>
               <xsd:element name="value" type="xsd:string" minOccurs="0"</pre>
msdata:Ordinal="1" />
             </xsd:sequence>
             <xsd:attribute name="name" type="xsd:string" use="required" />
           </xsd:complexType>
         </xsd:element>
       </xsd:choice>
     </xsd:complexType>
   </xsd:element>
 </xsd:schema>
 <resheader name="resmimetype">
   <value>text/microsoft-resx</value>
 </resheader>
 <resheader name="version">
   <value>2.0</value>
 </resheader>
 <resheader name="reader">
```

```
<value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
<resheader name="writer">
<value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
</root>
```

Settings.Designer.vb((*root directory name*)\VBHelloWorld\My Project内)

```
•_____
' <auto-generated>
     This code was generated by a tool.
     Runtime Version: 4.0.30319.42000
     Changes to this file may cause incorrect behavior and will be lost if
     the code is regenerated.
' </auto-generated>
Option Strict On
Option Explicit On
Namespace My
 <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _</pre>
Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Settings
 "11.0.0.0"), _
Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
 Partial Friend NotInheritable Class MySettings
   Inherits Global.System.Configuration.ApplicationSettingsBase
   Private Shared defaultInstance As MySettings =
CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New
MySettings), MySettings)
   #Region "My.Settings Auto-Save Functionality"
     #If _MyType = "WindowsForms" Then
       Private Shared addedHandler As Boolean
```

```
Private Shared addedHandlerLockObject As New Object
        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),
 Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
        Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal
 e As Global.System.EventArgs)
          If My.Application.SaveMySettingsOnExit Then
            My.Settings.Save()
          End If
        End Sub
      #End If
    #End Region
    Public Shared ReadOnly Property [Default]() As MySettings
      Get
        #If _MyType = "WindowsForms" Then
          If Not addedHandler Then
            SyncLock addedHandlerLockObject
              If Not addedHandler Then
                AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                addedHandler = True
              End If
            End SyncLock
          End If
        #End If
        Return defaultInstance
      End Get
    End Property
  End Class
End Namespace
Namespace My
  <Global.Microsoft.VisualBasic.HideModuleNameAttribute(), _
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
  Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute()> _
  Friend Module MySettingsProperty
    <Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")> _
    Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
      Get
```

```
Return Global.VBHelloWorld.My.MySettings.Default
End Get
End Property
End Module
End Namespace
```

Settings.settings((*root directory name*)\VBHelloWorld\My Project内)

```
<?xml version='1.0' encoding='utf-8'?>
<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings"
CurrentProfile="(Default)" UseMySettingsClassName="true">
<Profiles>
<Profile Name="(Default)" />
</Profiles>
<Settings />
</SettingsFile>
```

でビルドを計画する AWS CodeBuild

を使用する前に AWS CodeBuild、以下の質問に答える必要があります。

 ソースコードはどこにありますか。CodeBuild は現在、次のソースコードのリポジトリプロバイ ダからのビルドをサポートしています。ソースコードには、ビルド仕様 (buildspec) ファイルが 含まれている必要があります。ビルド環境は、CodeBuild がビルドを実行するために使用するオ ペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表しま す。buildspec は、ビルドプロジェクト定義で宣言できます。

リポジトリプロ バイダ	必須	ドキュメント
CodeCommit	リポジトリ名。 (オプション) ソースコードに 関連付けられて いるコミット ID。	AWS CodeCommit ユーザーガイドで以下のトピックを 参照してください。 「 <u>CodeCommit リポジトリを作成</u> 」 <u>Create a commit in CodeCommit</u>
Amazon S3	バケット名を入 力します。 ソースコード を含むビルド入 力 ZIP ファイル に対応するオブ ジェクト名。 (オプション) ビ ルド入力 ZIP ファイルに関連 付けられている バージョン ID。	「Amazon S3 入門ガイド」の以下のトピックを参照し てください。 <u>バケットの作成</u> <u>バケットにオブジェクトを追加する</u>

リポジトリプロ バイダ	必須	ドキュメント
GitHub	リポジトリ名。 (オプション) ソースコードに 関連付けられて いるコミット ID。	GitHub のヘルプウェブサイトでこのトピックを参照し てください。 <u>リポジトリを作成する</u>
Bitbucket	リポジトリ名。 (オプション) ソースコードに 関連付けられて いるコミット ID。	Bitbucket Cloud のドキュメントウェブサイトでこのト ピックを参照してください。 <u>リポジトリの作成</u>

- どのビルドコマンドを、どのような順番で実行する必要がありますか? デフォルトでは、CodeBuild は指定したプロバイダからビルド入力をダウンロードし、指定したバケットにビルド出力をアップロードします。ビルド仕様を使用して、ダウンロードされたビルド入力を想定されるビルド出力に変換する方法を指示します。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。
- ビルドを実行するためにどのランタイムとツールが必要ですか? たとえ ば、Java、Ruby、Python、Node.js を構築していますか? ビルドでは、Maven、Ant また は、Java、Ruby、Python のコンパイラが必要ですか? ビルドには Git、 AWS CLI、またはその他 のツールが必要ですか?

CodeBuild は、Docker イメージを使用するビルド環境でビルドを実行します。これらの Docker イメージを CodeBuild でサポートされているリポジトリタイプに保存する必要があります。これ らには、CodeBuild Docker イメージリポジトリ、Docker ハブ、および Amazon Elastic Container Registry (Amazon ECR) が含まれます。CodeBuild Docker イメージリポジトリの詳細について は、「CodeBuild に用意されている Docker イメージ」を参照してください。

 CodeBuild によって自動的に提供されない AWS リソースが必要ですか? そのようなリソース には、どのセキュリティポリシーが必要ですか? たとえば、CodeBuild サービスロールを変更し て、CodeBuild がそれらのリソースを操作できるようにする必要が生じることがあります。 5. CodeBuild を VPC と連携させますか。その場合は、VPC ID、サブネット ID、および VPC 設定 のセキュリティグループ ID が必要です。詳細については、「<u>Amazon Virtual Private Cloud AWS</u> CodeBuild で を使用する」を参照してください。

これらの質問に答えると、ビルドを正常に実行するために必要な設定とリソースがあるはずです。ビ ルドを実行するには、次の操作を実行できます。

- AWS CodeBuild コンソール、AWS CLI、または AWS SDKs。詳細については、「ビルドを手動 で実行」を参照してください。
- でパイプラインを作成または識別し AWS CodePipeline、CodeBuild にコードの自動テスト、ビルドの実行、またはその両方を指示するビルドまたはテストアクションを追加します。詳細については、「CodePipeline で CodeBuild を使用」を参照してください。

CodeBuild のビルド仕様に関するリファレンス

このトピックでは、ビルド仕様 (buildspec) ファイルに関する重要なリファレンス情報を提供しま す。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プロ グラミング言語ランタイム、およびツールの組み合わせを表します。buildspec をソースコードの一 部として含めることも、ビルドプロジェクトの作成時に buildspec を定義することもできます。ビル ド仕様の仕組みについては、「CodeBuild の仕組み」を参照してください。

トピック

- buildspec ファイル名とストレージの場所
- buildspec の構文
- buildspec の例
- buildspec のバージョン
- バッチビルドのビルド仕様 (buildspec) のリファレンス

buildspec ファイル名とストレージの場所

buildspec をソースコードの一部として含める場合、デフォルトの buildspec ファイルの名前は buildspec.yml で、ソースディレクトリのルートに配置する必要があります。

デフォルトの buildspec ファイルの名前と場所を変更することができます。たとえば、以下のことが 可能です。

- 同じリポジトリ内の異なるビルドに、buildspec_debug.ymlや buildspec_release.yml などの異なる buildspec ファイルを使用する。
- config/buildspec.yml など、ソースディレクトリのルート以外の場所や、S3 バケットに buildspec ファイルを保存する。S3 バケットは、ビルドプロジェクトと同じ AWS リージョンにあ る必要があります。ARN を使用して buildspec ファイルを指定します(例: arn:aws:s3:::<mycodebuild-sample2>/buildspec.yml)。

buildspec ファイルの名前に関係なく、ビルドプロジェクトには 1 つの buildspec しか指定できません。

デフォルトの buildspec ファイルの名前、場所、またはその両方をオーバーライドするには、次のい ずれかを実行します。

- または update-project コマンドを実行し AWS CLI create-project、buildspec値を組み 込み環境変数 の値に対する代替 buildspec ファイルのパスに設定しますCODEBUILD_SRC_DIR。 また、AWS SDKs の create projectオペレーションでも同等の操作を実行できます。詳細に ついては、「ビルドプロジェクトの作成」または「ビルドプロジェクト設定を変更」を参照してく ださい。
- コマンドを実行し AWS CLI start-build、buildspecOverride値を組み込み環境変数の値に 対する代替 buildspec ファイルへのパスに設定しますCODEBUILD_SRC_DIR。また、 AWS SDKs の start buildオペレーションでも同等の操作を実行できます。詳細については、「ビルドを手 動で実行」を参照してください。
- AWS CloudFormation テンプレートで、タイプのリソースSourceの BuildSpecプロパ ティAWS::CodeBuild::Projectを、組み込み環境変数の値に対する代替 buildspec ファイルの パスに設定しますCODEBUILD_SRC_DIR。詳細については、AWS CloudFormation ユーザーガイ ドのAWS CodeBuild プロジェクトソースの BuildSpec プロパティを参照してください。

buildspec の構文

buildspec ファイルは YAML 形式で表現する必要があります。

YAML でサポートされていない文字または文字列がコマンドに含まれている場合は、そのコマンド を引用符 ("") で囲む必要があります。次のコマンドが引用符で囲まれているのは、YAML ではコロン (:) に続けてスペースを使用できないためです。コマンド内の引用符はエスケープ (\") されます。

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }'
| sed 's/[\",]//g')"
```

buildspec の構文は次のとおりです。

```
version: 0.2

run-as: Linux-user-name

env:
    shell: shell-tag
    variables:
        key: "value"
        key: "value"
        key: "value"
        key: "value"
        key: "value"
```

```
exported-variables:
    - variable
    - variable
  secrets-manager:
    key: secret-id:json-key:version-stage:version-id
  git-credential-helper: no | yes
proxy:
  upload-artifacts: no | yes
  logs: no | yes
batch:
  fast-fail: false | true
  # build-list:
  # build-matrix:
  # build-graph:
  # build-fanout:
phases:
  install:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    runtime-versions:
      runtime: version
      runtime: version
    commands:
      - command
      - command
    finally:
      - command
      - command
  pre_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command
```

```
build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command
  post_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command
reports:
  report-group-name-or-arn:
    files:
      - location
      - location
    base-directory: location
    discard-paths: no | yes
    file-format: report-format
artifacts:
  files:
    - location
    - location
  name: artifact-name
  discard-paths: no | yes
  base-directory: location
  exclude-paths: excluded paths
  enable-symlinks: no | yes
  s3-prefix: prefix
  secondary-artifacts:
    artifactIdentifier:
      files:
        - location
```

- location name: secondary-artifact-name discard-paths: no | yes base-directory: location artifactIdentifier: files: - location - location discard-paths: no | yes base-directory: location cache: key: key fallback-keys: - fallback-key - fallback-key action: restore | save paths: - path - path

buildspec には、次のものが含まれています。

version

必要なマッピング。buildspec のバージョンを表します。0.2 を使用することをお勧めします。

Note バージョン 0.1 も引き続きサポートされていますが、可能な場合はバージョン 0.2 を使用 することをお勧めします。詳細については、「<u>buildspec のバージョン</u>」を参照してくださ い。

run-as

オプションのシーケンス。Linux ユーザーのみが使用できます。この buildspec ファイルでコマンド を実行する Linux ユーザーを指定します。run-as は、指定したユーザーに読み取りおよび実行の 許可を付与します。buildspec ファイルの上で run-as を指定すると、すべてのコマンドにグロー バルに適用されます。すべての buildspec ファイルコマンドのユーザーを指定しない場合、phases ブロックのいずれかで run-as を使用することによりフェーズでいずれかのコマンドを指定できま す。run-as を指定しない場合、すべてのコマンドがルートユーザーとして実行されます。

env

オプションのシーケンス。1つ以上のカスタム環境変数の情報を表します。

Note 機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。 AWS アクセスキー IDs。詳細については、AWS Identity and Access Management フ

- AWS アクセスキー IDs。詳細については、AWS Identity and Access Management ユー ザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
- パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「<u>Systems Manager パラメータストア</u>」および 「<u>Systems Manager パラメータストアコンソールのチュートリアル</u>」を参照してください。
- を使用して指定された文字列 AWS Secrets Manager。詳細については、「<u>キー管理</u>」を参 照してください。

env/shell

オプションのシーケンス。Linux または Windows オペレーティングシステムでサポートされる シェルを指定します。

Linux オペレーティングシステムで、サポートされているシェルタグは次のとおりです。

- bash
- /bin/sh

Windows オペレーティングシステムで、サポートされているシェルタグは次のとおりです。

- powershell.exe
- cmd.exe

env/variables

env を指定し、プレーンテキストでカスタム環境変数を定義する場合は必須です。##と#のスカ ラーのマッピングを含み、各マッピングはプレーンテキストで1つのカスタム環境変数を表しま す。##は、カスタム環境変数の名前で、#はその変数の値です。

▲ Important

機密の値は環境変数に保存しないことを強くお勧めします。環境変数は、CodeBuild コン ソールや AWS CLIなどのツールを使用してプレーンテキストで表示できます。機密情報 については、このセクションの後半で説明するように、parameter-store マッピング または secrets-manager マッピングを代わりに使用することをお勧めします。 既存の環境変数は、設定した環境変数によって置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれ ていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場 合、my_value が other_value に置き換えられます。同様に、Docker イメージに / usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に 含まれていて、\$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数 を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 \$PATH:/usr/ share/ant/bin に置き換えられます。

CODEBUILD_で始まる名前の環境変数は設定しないでください。このプレフィックスは 内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定され ます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。ビルドの作成時に環境変数を追加または上書きできます。詳細については、「<u>AWS CodeBuild ビル</u>ドを手動で実行する」を参照してください。
- ビルドプロジェクト定義の値が次に優先されます。プロジェクトを作成または編集する ときに、プロジェクトレベルで環境変数を追加できます。詳細については、「<u>でビルド</u> <u>プロジェクトを作成する AWS CodeBuild</u>」および「<u>でビルドプロジェクト設定を変更</u> <u>する AWS CodeBuild</u>」を参照してください。
- ビルド仕様宣言の値の優先順位が最も低くなります。

env/parameter-store

env が指定されていて、Amazon EC2 Systems Manager パラメータストアに保存されている カスタム環境変数を取得する場合は必須です。##と#のマッピングを含み、各マッピングは単 ーのカスタム環境変数を表し、Amazon EC2 Systems Manager パラメータストアに保存され ます。##は、後でビルドコマンドで使用するこのカスタム環境変数を参照する名前で、#は Amazon EC2 Systems Manager パラメータストアに保存されているカスタム環境変数の名前 です。重要な値を保存するには、Amazon EC2 Systems Manager ユーザーガイドの「Systems
<u>Manager パラメータストア</u>」および「<u>チュートリアル: String パラメータの作成とテスト (コン</u> ソール)」を参照してください。

A Important

Amazon EC2 Systems Manager パラメータストアに保存されているカスタム環境 変数を取得することを CodeBuild に許可するには、CodeBuild サービスロールに ssm:GetParameters アクションを追加する必要があります。詳細については、 「<u>CodeBuild が他の AWS のサービスとやり取りすることを許可</u>」を参照してください。 Amazon EC2 Systems Manager パラメータストアから取得する環境変数は、既存の 環境変数を置き換えます。たとえば、Docker イメージに MY_VAR という名前で値が my_value の環境変数がすでに含まれていて、MY_VAR という名前で値が other_value の環境変数を取得した場合、my_value は other_value に置き換えられます。同様 に、Docker イメージに /usr/local/sbin:/usr/local/bin という名前で値が PATH の環境変数を取得した場合、/usr/local/sbin:/usr/local/bin という名前で値が PATH の環境変数を取得した場合、/usr/local/sbin:/usr/local/bin はリテラル値 \$PATH:/usr/share/ant/bin に置き換えられます。

CODEBUILD_で始まる名前の環境変数は保存しないでください。このプレフィックスは 内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定され ます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。ビルドの作成時に環境変数を追加または上書きできます。詳細については、「<u>AWS CodeBuild ビル</u>ドを手動で実行する」を参照してください。
- ビルドプロジェクト定義の値が次に優先されます。プロジェクトを作成または編集する ときに、プロジェクトレベルで環境変数を追加できます。詳細については、「<u>でビルド</u> <u>プロジェクトを作成する AWS CodeBuild</u>」および「<u>でビルドプロジェクト設定を変更</u> する AWS CodeBuild」を参照してください。
- ビルド仕様宣言の値の優先順位が最も低くなります。

env/secrets-manager

に保存されているカスタム環境変数を取得する場合に必要です AWS Secrets Manager。次のパ ターンを使用して、Secrets Manager reference-key を指定します。

<key>: <secret-id>:<json-key>:<version-stage>:<version-id>

<kev>

(必須) ローカル環境変数の名前。この名前を使用して、ビルド中に変数にアクセスします。 <<u>secret-id</u>>

(必須) シークレットの一意の識別子として機能する名前または Amazon リソースネーム (ARN) です。 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、シークレット ARN を指定 します。

<json-key>

(オプション) 値を取得する Secrets Manager のキーと値のペアのキー名を指定しま

す。json-keyを指定しない場合、CodeBuild はシークレットテキスト全体を取得します。

<version-stage>

(オプション) バージョンに添付されているステージングラベルによって取得するシークレット のバージョンを指定します。ステージングラベルは、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 を指定しないでください。バージョンステージもバージョン ID も指定しない場合、デフォル トでは AWSCURRENT のバージョンステージ値でバージョンが取得されます。

<version-id>

(オプション) 使用したいシークレットのバージョンの固有 ID 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。バージョンステージもバー ジョン ID も指定しない場合、デフォルトでは AWSCURRENT のバージョンステージ値でバー ジョンが取得されます。

次の例で、TestSecret は Secrets Manager に保存されているキーと値のペアの名前で す。TestSecret のキーは MY_SECRET_VAR です。ビルド中に変数にアクセスするには、名前 「LOCAL_SECRET_VAR」を使用します。

env:
 secrets-manager:
 LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"

詳細については、<u>AWS Secrets Managerユーザーガイド</u>の「AWS Secrets Manager とは?」を参 照してください。

env/exported-variables

オプションのマッピング。エクスポートする環境変数をリストするために使用します。エクス ポートする各変数の名前を、exported-variablesの別の行で指定します。エクスポートする 変数は、ビルド中にコンテナで使用できる必要があります。エクスポートする変数は、環境変数 にすることができます。

エクスポートされた環境変数は、と組み合わせて使用 AWS CodePipeline して、現在のビルド ステージからパイプラインの後続のステージに環境変数をエクスポートします。詳細について は、AWS CodePipeline ユーザーガイドの変数の操作を参照してください。

ビルド中、変数の値は、install フェーズから開始して使用できます。これは、install フェーズの開始と post_build フェーズの終了の間に更新することができます。post_build フェーズが終了すると、エクスポートされた変数の値は変更できません。

Note

以下はエクスポートできません:

- ビルドプロジェクトで指定された Amazon EC2 Systems Manager Parameter Store シークレット
- ・ ビルドプロジェクトで指定された Secrets Manager のシークレット
- AWS_ で始まる環境変数。

env/git-credential-helper

オプションのマッピング。CodeBuild が Git 認証情報ヘルパーを使用して Git 認証情報を提供す るかどうかを示します。使用する場合は yes です。それ以外の場合は、no または指定なしで す。詳細については、Git ウェブサイトの「gitcredentials」を参照してください。

Note

git-credential-helper は、パブリック Git リポジトリの Webhook によってトリ ガーされるビルドではサポートされません。

proxy

オプションのシーケンス。明示的なプロキシサーバーでビルドを実行する場合、設定を表すために使 用されます。詳細については、「<u>明示的なプロキシサーバーでの CodeBuild の実行</u>」を参照してく ださい。

proxy/upload-artifacts

オプションのマッピング。明示的なプロキシサーバーのビルドでアーティファクトをアップロー ドする場合は、yes に設定します。デフォルトは no です。

proxy/logs

オプションのマッピング。明示的なプロキシサーバーのビルドで、CloudWatch ログを作成する には、yes に設定します。デフォルトは no です。

phases

必要なシーケンス。ビルドの各段階で CodeBuild が実行するコマンドを表します。

Note

buildspec バージョン 0.1 では、CodeBuild はビルド環境のデフォルトシェルの各インスタン スで各コマンドを実行します。つまり、各コマンドは他のすべてのコマンドとは独立して実 行されます。したがって、デフォルトでは、以前のコマンド (ディレクトリの変更や環境変 数の設定など)の状態に依存する単一のコマンドを実行することはできません。この制限を 回避するには、バージョン 0.2 を使用することをお勧めします。これにより、問題が解決さ れます。buildspec バージョン 0.1 を使用する必要がある場合は、「ビルド環境のシェルとコ マンド」のアプローチをお勧めします。

phases/*/run-as

オプションのシーケンス。ビルドフェーズで使用し、そのコマンドを実行する Linux ユーザー を指定します。buildspec ファイルの上ですべてのコマンドに対して run-as もグローバルに指 定されている場合、フェーズレベルのユーザーが優先されます。例えば、run-as がグローバ ルに User-1 を指定し、run-as ステートメントが install フェーズでのみ User-2 を指定し た場合、buildspec ファイル内のすべてのコマンドは User-1 として実行されますが、install フェーズのコマンドは除きます (これらのコマンドは User-2 として実行されます)。

phases/*/on-failure

オプションのシーケンス。フェーズ中に障害が発生した場合に実行するアクションを指定しま す。これには、次のいずれかの値を指定できます。

- ABORT ビルドを中止します。
- CONTINUE 次のフェーズに進みます。
- RETRY 正規表現 に一致するエラーメッセージでビルドを最大3回再試行します.*。
- RETRY-*count* 正規表現 に一致するエラーメッセージで####で表される、指定された回数だ けビルドを再試行します.*。####は 0~100 の間でなければならないことに注意してください。たとえば、有効な値には RETRY-4と が含まれますRETRY-8。
- RETRY-*regex* ビルドを最大3回再試行し、####を使用して指定されたエラーメッセージ に一致する正規表現を含めます。たとえば、有効な値にはRetry-.*Error: Unable to connect to database.*とが含まれますRETRY-invalid+。
- RETRY-count-regex ####で表される指定された回数だけビルドを再試行します。# ###は 0~100 の間でなければならないことに注意してください。####を使用して、エ ラーメッセージに一致する正規表現を含めることもできます。たとえば、有効な値には Retry-3-.*connection timed out.*と が含まれますRETRY-8-invalid+。

このプロパティを指定しない場合は、<u>ビルドフェーズの移行</u> に示すように、失敗処理が遷移 フェーズに続きます。

phases/*/finally

オプションのブロック。finally ブロックに指定したコマンドは、commands ブロックのコマ ンドの実行後に実行されます。finally ブロックに指定したコマンドは、commands ブロック のコマンドが失敗した場合でも実行されます。例えば、commands ブロック内に 3 つのコマン ドがあり、最初のコマンドが失敗した場合、CodeBuild は残りの 2 つのコマンドをスキップして finally ブロック内のコマンドを実行します。commands ブロックと finally ブロックのすべ てのコマンドが正常に実行されると、フェーズは成功します。フェーズのいずれかのコマンドが 失敗すると、フェーズは失敗します。

許可されるビルドフェーズ名は次のとおりです。

phases/install

オプションのシーケンス。インストール時に CodeBuild が実行するコマンドがある場合は、その コマンドを表します。install フェーズは、ビルド環境でのパッケージのインストールにのみ使 用することをお勧めします。たとえば、このフェーズを使用して、Mocha や RSpec などのコー ドテストフレームワークをインストールすることができます。

phases/install/runtime-versions

オプションのシーケンス。ランタイムバージョンは、Ubuntu 標準イメージ 5.0 以降および Amazon Linux 2 標準イメージ 4.0 以降でサポートされています。指定した場合、少なくと も 1 つのランタイムをこのセクションに含める必要があります。ランタイムを指定するに は、特定のバージョンを使用するか、メジャーバージョンに .x を続けて CodeBuild がこのメ ジャーバージョンと最新マイナーバージョンを使用することを指定するか、1atest を指定し て最新のメジャーバージョンとマイナーバージョン (ruby: 3.2、nodejs: 18.x、java: 1atest など)を使用します。数値または環境変数を使用してランタイムを指定できます。例 えば、Amazon Linux 2 標準イメージ 4.0 を使用している場合、次の例は、Java のバージョン 17、Python バージョン 3 の最新マイナーバージョン、および Ruby の環境変数内のバージョ ンをインストールすることを指定します。詳細については、「<u>CodeBuild に用意されている</u> Docker イメージ」を参照してください。

phases: install: runtime-versions: java: corretto8 python: 3.x ruby: "\$MY_RUBY_VAR"

buildspec ファイルの runtime-versions セクションで 1 つ以上のランタイムを指定で きます。ランタイムが別のランタイムに依存している場合は、依存しているランタイムを buildspec ファイルで指定することもできます。buildspec ファイルでランタイムを指定し ない場合は、CodeBuild により、使用するイメージのデフォルトのランタイムが選択されま す。1 つ以上のランタイムを指定すると、CodeBuild により、それらのランタイムのみが使用 されます。依存関係のあるランタイムが指定されていない場合、CodeBuild により、依存関係 のあるランタイムの選択が試みられます。

2 つの指定されたランタイムが競合する場合、ビルドは失敗します。たとえば、android: 29 と java: openjdk11 が矛盾するので、両方が指定されている場合は、ビルドは失敗し ます。

使用できるランタイムの詳細については、「使用可能なランタイム」を参照してください。

Note

runtime-versions セクションを指定して、Ubuntu 標準イメージ 2.0 以降や Amazon Linux 2 (AL2) 標準イメージ 1.0 以降以外のイメージを使用した場合は、ビル ドで「Skipping install of runtimes. Runtime version selection is not supported by this build image」の警告が表示されます。

phases/install/commands

オプションのシーケンス。一連のスカラーが含まれ、各スカラーは、インストール中に CodeBuild が実行する単一のコマンドを表します。CodeBuild は、最初から最後まで、各コマ ンドを一度に1つずつ、指定された順序で実行します。

phases/pre_build

オプションのシーケンス。ビルドの前に CodeBuild が実行するコマンドがあれば、それを表しま す。たとえば、このフェーズを使用して Amazon ECR にサインインするか、npm の依存関係を インストールすることができます。

phases/pre_build/commands

pre_build が指定されている場合は必須のシーケンスです。一連のスカラーが含まれ、各ス カラーは、ビルドの前に CodeBuild が実行する単一のコマンドを表します。CodeBuild は、 最初から最後まで、各コマンドを一度に 1 つずつ、指定された順序で実行します。

phases/build

オプションのシーケンス。ビルド中に CodeBuild が実行するコマンドがあれば、それを表しま す。たとえば、このフェーズを使用して、Mocha、RSpec、または sbt を実行できます。 phases/build/commands

build が指定されている場合は必須です。一連のスカラーが含まれ、各スカラーは、ビルド 中に CodeBuild が実行する単一のコマンドを表します。CodeBuild は、最初から最後まで、 各コマンドを一度に 1 つずつ、指定された順序で実行します。

phases/post_build

オプションのシーケンス。ビルドの後に CodeBuild が実行するコマンドがあれば、それを表 します。たとえば、Maven を使用してビルドアーティファクトを JAR または WAR ファイル にパッケージ化するか、Docker イメージを Amazon ECR にプッシュすることができます。次 に、Amazon SNS を介してビルド通知を送信できます。

phases/post_build/commands

post_build が指定されている場合は必須です。一連のスカラーが含まれ、各スカラーは、 ビルドの後に CodeBuild が実行する単一のコマンドを表します。CodeBuild は、最初から最 後まで、各コマンドを一度に1つずつ、指定された順序で実行します。

レポート

report-group-name-or-arn

オプションのシーケンス。レポートの送信先のレポートグループを指定します。プロジェクトには、最大5つのレポートグループを含めることができます。既存のレポートグループのARN、または新しいレポートグループの名前を指定します。名前を指定した場合、CodeBuildは、プロジェクト名と <project-name>-<report-group-name> 形式で指定した名前を使用してレポートグループを作成します。レポートグループ名は、\$REPORT_GROUP_NAME などのbuildspecの環境変数を使用して設定することもできます。詳細については、「<u>Report group</u>naming」を参照してください。

reports/<report-group>/files

必要なシーケンス。レポートによって生成されたテスト結果の生データを含む場所を表しま す。スカラーのシーケンスが含まれます。各スカラーは、元のビルドの場所または basedirectory (設定されている場合)を基準にして、CodeBuild がテストファイルを検索できる個 別の場所を表します。場所には次のものが含まれます。

- 1つのファイル (例: my-test-report-file.json)。
- サブディレクトリ内の単一のファイル (my-subdirectory/my-test-report-file.json や my-parent-subdirectory/my-subdirectory/my-test-report-file.json な ど)。
- ・ '**/*' はすべてのファイルを再帰的に表します。
- my-subdirectory/*は、my-subdirectoryという名前のサブディレクトリ内のすべての ファイルを表します。
- my-subdirectory/**/* は、my-subdirectory という名前のサブディレクトリから再帰 的にすべてのファイルを表します。

reports/<report-group>/file-format

オプションのマッピング。レポートファイル形式を表します。指定しない場合は、JUNITXML を 使用します。この値は大文字と小文字が区別されません。想定される値は次のとおりです。 テストレポート

CUCUMBERJSON

Cucumber JSON

JUNITXML

JUnit XML

NUNITXML

NUnit XML

NUNIT3XML

NUnit 3 XML

TESTNGXML

TestNG XML

VISUALSTUDIOTRX

Visual Studio TRX

コードカバレッジレポート

CLOVERXML

クローバー XML

COBERTURAXML

Cobertura XML

JACOCOXML

JaCoCo XML

SIMPLECOV

SimpleCov JSON

Note

CodeBuild は、<u>simplecov-json</u> ではなく、<u>simplecov</u> によって生成された JSON コードカバレッジレポートを受け入れます。

reports/<report-group>/base-directory

オプションのマッピング。CodeBuild が生のテストファイルを見つける場所を決定するために使 用する元のビルド場所に対する相対的な 1 つ以上のトップレベルディレクトリを表します。

reports/<report-group>/discard-paths

オプション。レポートファイルのディレクトリを出力でフラット化するかどうかを指定します。 これが指定されていない場合、または no を含む場合、レポートファイルはディレクトリ構造の まま出力されます。このディレクトリに yes が含まれている場合、すべてのテストファイルが同 じ出力ディレクトリに配置されます。たとえば、テスト結果へのパスが com/myapp/mytests/ TestResult.xml である場合、yes を指定すると、このファイルが /TestResult.xml に配置 されます。

artifacts

オプションのシーケンス。CodeBuild がビルド出力を見つけることができる場所に関する情 報、CodeBuild が S3 出力バケットへのアップロード用にその出力を準備する方法に関する情報を表 します。たとえば、Docker イメージを作成して Amazon ECR にプッシュしている場合、または、 ソースコードでユニットテストを実行していてもビルドしていない場合、このシーケンスは必要あり ません。

Note

Amazon S3 メタデータには、x-amz-meta-codebuild-buildarn という名前の CodeBuild ヘッダーがあります。このヘッダーには、Amazon S3 にアーティファクトを公開 する CodeBuild ビルドの buildArn が含まれています。通知のソーストラッキングを許可 し、アーティファクトの生成元であるビルドを参照するには、buildArn を追加します。

artifacts/files

必要なシーケンス。ビルド環境でのビルド出力アーティファクトを含む場所を表します。スカ ラーのシーケンスが含まれ、各スカラーは、CodeBuild が元のビルドの場所を基準に、あるいは 設定されている場合はベースディレクトリを基準にして、ビルド出力アーティファクトを見つけ ることができる個別の場所を表します。場所には次のものが含まれます。

• 1つのファイル (例: my-file.jar)。

- サブディレクトリ内の単一のファイル (my-subdirectory/my-file.jar や my-parentsubdirectory/my-subdirectory/my-file.jar など)。
- ・ '**/*' はすべてのファイルを再帰的に表します。
- my-subdirectory/*は、my-subdirectoryという名前のサブディレクトリ内のすべての ファイルを表します。
- my-subdirectory/**/* は、my-subdirectory という名前のサブディレクトリから再帰 的にすべてのファイルを表します。

ビルド出力アーティファクトの場所を指定すると、CodeBuild はビルド環境で元のビルドの場所 を特定できます。ビルドアーティファクトの出力先の場所に、元のビルドの場所へのパスを追 加する、または ./ などで場所を指定する必要はありません。この場所へのパスを知りたい場合 は、ビルド中に echo \$CODEBUILD_SRC_DIR などのコマンドを実行できます。各ビルド環境の 場所は多少異なる場合があります。

artifacts/name

オプション名。ビルドアーティファクトの名前を指定します。この名前は、次のいずれかに該当 する場合に使用されます。

- CodeBuild API を使用してビルドを作成し、プロジェクトの更新、プロジェクトの作成、また はビルドの開始時に ProjectArtifacts オブジェクトに overrideArtifactName フラグ を設定した場合。
- CodeBuild コンソールを使用してビルドを作成し、buildspec ファイルで名前を指定して、プロジェクトの作成または更新時に [Enable semantic versioning] (セマンティックバージョニングの有効化)を選択した場合。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

ビルド時に計算される buildspec ファイルの名前を指定できます。buildspec ファイルで指定され た名前は、Shell コマンド言語を使用します。たとえば、アーティファクト名に日付と時刻を追加 して常に一意にできます。アーティファクト名を一意にすると、アーティファクトが上書きされ るのを防ぐことができます。詳細については、「<u>Shell コマンド言語</u>」を参照してください。

• これは、アーティファクトが作成された日付が付加されたアーティファクト名の例です。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
```

```
files:
    - '**/*'
name: myname-$(date +%Y-%m-%d)
```

 この例は、CodeBuild 環境変数を使用するアーティファクト名の例です。詳細については、 「ビルド環境の環境変数」を参照してください。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
    files:
            - '**/*'
    name: myname-$AWS_REGION
```

 これは、アーティファクトの作成日が付加された CodeBuild 環境変数を使用するアーティファ クト名の例です。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
    files:
            - '**/*'
    name: $AWS_REGION-$(date +%Y-%m-%d)
```

名前にパス情報を追加して、名前のアーチファクトが名前のパスに基づいてディレクトリに配置 されるようにできます。この例では、ビルドアーティファクトは、builds/<build number>/ my-artifactsの下に配置されます。

```
version: 0.2
phases:
    build:
        commands:
            - rspec HelloWorld_spec.rb
artifacts:
    files:
            - '**/*'
```

name: builds/\$CODEBUILD_BUILD_NUMBER/my-artifacts

artifacts/discard-paths

オプション。ビルドアーティファクトのディレクトリが出力でフラット化されるかどうかを指 定します。これが指定されていない場合、または no を含む場合は、ディレクトリ構造はそのま まで、ビルドアーティファクトが出力されます。このディレクトリに yes が含まれる場合、す べてのビルドアーティファクトが同じ出力ディレクトリに配置されます。たとえば、ビルド出力 アーティファクト内のファイルへのパスが com/mycompany/app/HelloWorld.java である場 合、yes を指定すると、このファイルが /HelloWorld.java に配置されます。

artifacts/base-directory

オプションのマッピング。CodeBuild がビルド出力アーティファクトに含めるファイルとサブ ディレクトリを決定するために使用する、元のビルドの場所を基準とした、1 つ以上の最上位 ディレクトリを表します。有効な値を次に示します。

- 単一の最上位ディレクトリ (例:my-directory)。
- 'my-directory*' my-directory で始まる名前を持つすべての最上位ディレクトリを表します。

一致する最上位ディレクトリはビルド出力アーティファクトに含まれず、ファイルとサブディレ クトリにのみ含まれます。

files および discard-paths を使用して、どのファイルとサブディレクトリを含めるかをさ らに制限できます。たとえば、以下のようなディレクトリ構造があります。

```
.
### my-build-1
# ### my-file-1.txt
### my-build-2
### my-file-2.txt
### my-subdirectory
### my-file-3.txt
```

また、次のような artifacts シーケンスがあります。

```
artifacts:
    files:
        - '*/my-file-3.txt'
```

```
base-directory: my-build-2
```

次のサブディレクトリとファイルが、ビルド出力アーティファクトに含まれます。

```
### my-subdirectory
    ### my-file-3.txt
```

さらに、次のような artifacts シーケンスがあります。

```
artifacts:
    files:
        - '**/*'
    base-directory: 'my-build*'
    discard-paths: yes
```

次のファイルが、ビルド出力アーティファクトに含まれます。

```
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/exclude-paths

オプションのマッピング。base-directory に相対的な 1 つまたは複数のパスを表しま す。CodeBuild はビルドのアーティファクトから、このパスを除外します。アスタリスク (*) 記 号は、フォルダの境界を超えない、0 文字以上の名前の要素と一致します。二重アスタリスク (**) は、すべてのディレクトリをまたいで、0 文字以上の名前の要素と一致します。

除外パスの例は以下のとおりです。

- すべてのディレクトリからファイルを除外する場合: "**/file-name/**/*"
- すべてのドットフォルダを除外する場合: "**/.*/**/*"
- すべてのドットファイルを除外する場合: "**/.*"

artifacts/enable-symlinks

オプション。出力タイプが ZIP の場合、内部シンボリックリンクを ZIP ファイルに保持するかど うかを指定します。これに yes が含まれる場合、ソース内のすべての内部シンボリックリンクが アーティファクト ZIP ファイルに保持されます。

artifacts/s3-prefix

オプション。アーティファクトを Amazon S3 バケットに出力し、名前空間タイプが BUILD_ID の場合に使用するプレフィックスを指定します。使用した場合、バケット内の出力パスは「<s3prefix>/<build-id>/<name>.zip」となります。

artifacts/secondary-artifacts

オプションのシーケンス。アーティファクト識別子とアーティファクト定義との間のマッピング としての1つ以上のアーティファクト定義を表します。このブロック内の各アーティファクト識 別子は、プロジェクトの secondaryArtifacts 属性で定義されたアーティファクトと一致する 必要があります。各個別の定義は、上記の artifacts ブロックと同じ構文を持っています。

Note

「<u>artifacts/files</u>」シーケンスは、セカンダリアーティファクトしか定義されていな い場合でも、常に必要です。

たとえば、プロジェクトに次の構造があるとします。

```
{
  "name": "sample-project",
  "secondaryArtifacts": [
    {
      "type": "S3",
      "location": "<output-bucket1>",
      "artifactIdentifier": "artifact1",
      "name": "secondary-artifact-name-1"
    },
    {
      "type": "S3",
      "location": "<output-bucket2>",
      "artifactIdentifier": "artifact2",
      "name": "secondary-artifact-name-2"
    }
  ]
}
```

次に、buildspec は次のようになります。

version: 0.2

```
phases:
build:
  commands:
    - echo Building...
artifacts:
  files:
    - '**/*'
 secondary-artifacts:
    artifact1:
      files:
        - directory/file1
      name: secondary-artifact-name-1
    artifact2:
      files:
        - directory/file2
      name: secondary-artifact-name-2
```

cache

オプションのシーケンス。CodeBuild が S3 キャッシュバケットへのキャッシュのアップロード 用にファイルを準備できる場所に関する情報を表します。プロジェクトのキャッシュタイプが No Cache の場合、このシーケンスは不要です。

キャッシュ/キー

オプションのシーケンス。キャッシュを検索または復元するときに使用するプライマリキーを表 します。CodeBuild はプライマリキーと完全に一致します。

キーの例を次に示します。

key: npm-key-\$(codebuild-hash-files package-lock.json) }

キャッシュ/フォールバックキー

オプションのシーケンス。プライマリキーを使用してキャッシュが見つからない場合に順次使用 されるフォールバックキーのリストを表します。最大5つのフォールバックキーがサポートさ れ、それぞれがプレフィックス検索を使用して照合されます。キーが指定されていない場合、こ のシーケンスは無視されます。

フォールバックキーの例を次に示します。

```
fallback-keys:
    - npm-key-$(codebuild-hash-files package-lock.json) }
    - npm-key-
    - npm-
```

キャッシュ/アクション

オプションのシーケンス。キャッシュで実行するアクションを指定します。有効な値を次に示し ます。

- restore 更新を保存せずにキャッシュのみを復元します。
- save 以前のバージョンを復元せずにキャッシュのみを保存します。

値が指定されていない場合、CodeBuild はデフォルトで復元と保存の両方を実行します。 cache/paths

必要なシーケンス。キャッシュの場所を表します。スカラーのシーケンスが含まれ、各スカラー は、CodeBuild が元のビルドの場所を基準に、あるいは設定されている場合はベースディレクト リを基準にして、ビルド出力アーティファクトを見つけることができる個別の場所を表します。 場所には次のものが含まれます。

- 1つのファイル (例: my-file.jar)。
- サブディレクトリ内の単一のファイル (my-subdirectory/my-file.jar や my-parentsubdirectory/my-subdirectory/my-file.jar など)。
- ・ '**/*' はすべてのファイルを再帰的に表します。
- my-subdirectory/*は、my-subdirectoryという名前のサブディレクトリ内のすべての ファイルを表します。
- my-subdirectory/**/* は、my-subdirectory という名前のサブディレクトリから再帰 的にすべてのファイルを表します。

A Important

buildspec 宣言は有効な YAML である必要があるため、buildspec 宣言のスペースは重要で す。buildspec の宣言にあるスペースの数が無効な場合、すぐにビルドが失敗する可能性が あります。YAML validator を使用して、buildspec の宣言が有効な YAML かどうかをテスト できます。

AWS CLIビルドプロジェクトを作成または更新するときに、 または AWS SDKs を使用して buildspec を宣言する場合、buildspec は YAML 形式で表される 1 つの文字列で、必要な空 白文字と改行エスケープ文字が含まれている必要があります。次のセクションに例がありま す。

buildspec.yml ファイルの代わりに CodeBuild または AWS CodePipeline コンソールを使用 する場合は、 buildフェーズのコマンドのみを挿入できます。上記の構文を使用する代わり に、ビルドフェーズで実行するすべてのコマンドを 1 行に記載します。複数のコマンドにつ いては、&& で各コマンドを区切ります (例: mvn test && mvn package)。

buildspec.yml ファイルの代わりに CodeBuild または CodePipeline コンソールを使用して、 ビルド環境でビルド出力アーティファクトの場所を指定することができます。上記の構文を 使用する代わりに、すべての場所を1行に記載します。複数の場所の場合は、各場所をコン マで区切ります (例: buildspec.yml, target/my-app.jar)。

buildspec の例

buildspec.yml ファイルの例を次に示します。

```
version: 0.2
env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword
phases:
  install:
    commands:
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
    finally:
      - echo This always runs even if the update or install command fails
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - docker login -u User -p $LOGIN_PASSWORD
    finally:
      - echo This always runs even if the login command fails
  build:
    commands:
      - echo Entered the build phase...
```

```
- echo Build started on `date`
      - mvn install
    finally:
      - echo This always runs even if the install command fails
  post_build:
    commands:
      - echo Entered the post_build phase...
      - echo Build completed on `date`
reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
    files:
      - "**/*"
    base-directory: 'target/tests/reports'
    discard-paths: no
  reportGroupCucumberJson:
    files:
      - 'cucumber/target/cucumber-tests.xml'
    discard-paths: yes
    file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
    artifact1:
      files:
        - target/artifact-1.0.jar
      discard-paths: yes
    artifact2:
      files:
        - target/artifact-2.0.jar
      discard-paths: yes
cache:
  paths:
    - '/root/.m2/**/*'
```

以下は、 AWS CLIまたは AWS SDKs。

```
"version: 0.2\n\nenv:\n variables:\n JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-
amd64\\"\n parameter-store:\n LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n
phases:\n\n install:\n commands:\n - echo Entered the install phase...\n
- apt-get update -y\n - apt-get install -y maven\n finally:\n - echo This
```

always runs even if the update or install command fails \n pre_build:\n commands: - echo Entered the pre_build phase...\n - docker login -u User -p n/\$LOGIN PASSWORD\n - echo This always runs even if the login command finally:\n fails \n build:\n commands:\n - echo Entered the build phase...\n - echo - echo This always Build started on `date`\n - mvn install\n finally:\n runs even if the install command fails\n post_build:\n commands:\n - echo Entered the post_build phase...\n - echo Build completed on `date`\n\n reports: - \"**/*\"\n \n reportGroupJunitXml:\n files:\n base-directory: 'target/ tests/reports'\n discard-paths: false\n reportGroupCucumberJson:\n files:\n - 'cucumber/target/cucumber-tests.xml'\n file-format: CUCUMBERJSON\n\nartifacts:\n - target/messageUtil-1.0.jar\n discard-paths: yes\n secondary-artifacts: files:\n discardn/artifact1:\n files:\n - target/messageUtil-1.0.jar\n artifact2:\n files:\n - target/messageUtil-1.0.jar\n paths: yes\n - '/root/.m2/**/*'" discard-paths: yes\n cache:\n paths:\n

CodeBuild または CodePipeline コンソールで使用する build フェーズのコマンドの例を次に示します。

echo Build started on `date` && mvn install

これらの例では:

- JAVA_HOME のキーと /usr/lib/jvm/java-8-openjdk-amd64 の値を持つプレーンテキスト のカスタム環境変数が設定されます。
- Amazon EC2 Systems Manager パラメータストアに保存した dockerLoginPassword というカ スタム環境変数は、後で LOGIN_PASSWORD キーを使用してビルドコマンドで参照します。
- これらのビルドフェーズ名は変更できません。この例で実行されるコマンドは、apt-get update -y と apt-get install -y maven (Apache Maven をインストールする)、mvn install (ソースコードをコンパイル、テストして、ビルド出力アーティファクトにパッ ケージ化し、ビルド出力アーティファクトを内部リポジトリにインストールする)、docker login (Amazon EC2 Systems Manager パラメータストアで設定したカスタム環境変数 dockerLoginPassword の値に対応するパスワードを使用して Docker ヘサインインする)、およ びいくつかの echo コマンドです。これらの echo コマンドは、CodeBuild がコマンドを実行する 方法と、コマンドの実行順序を示すためにここに含めています。
- files はビルド出力場所にアップロードするファイルを表します。この例では、CodeBuild が 1つのファイル「messageUtil-1.0.jar」をアップロードします。messageUtil-1.0.jar ファイルは、ビルド環境の target という名の相対ディレクトリ内にあります。discardpaths: yes が指定されたため、messageUtil-1.0.jar は直接アップロードされます (中間の target ディレクトリにはアップロードされません)。ファイル名 messageUtil-1.0.jar およ

び相対ディレクトリ名 target は、Apache Maven がこの例のビルド出力アーティファクトを作 成して保存する方法にのみ基づいています。独自のシナリオでは、これらのファイル名とディレク トリは異なります。

- reportsは、ビルド中にレポートを生成する2つのレポートグループを表します。
 - arn:aws:codebuild:your-region:your-aws-account-id:report-group/reportgroup-name-1は、レポートグループの ARN を指定します。テストフレームワークによって 生成されたテスト結果は、target/tests/reports ディレクトリにあります。ファイル形式 は JunitXml であり、パスはテスト結果を含むファイルから削除されません。
 - reportGroupCucumberJsonは、新しいレポートグループを指定します。プロジェクトの名前がmy-projectの場合、ビルドの実行時にmy-project-reportGroupCucumberJsonという名前のレポートグループが作成されます。テストフレームワークによって生成されたテスト結果は、cucumber/target/cucumber-tests.xmlにあります。テストファイルの形式は、CucumberJsonで、テスト結果を含むファイルからパスが削除されます。

buildspec のバージョン

次の表に、buildspec のバージョンとバージョン間の変更を示します。

バージョン	変更
0.2	 environment_variables が env に名称変更されました。 plaintext が variables に名称変更されました。 artifacts の type プロパティは廃止されました。 バージョン 0.1 では、はビルド環境のデフォルトシェルの個別のインスタンスで各ビルドコマンド AWS CodeBuild を実行します。バージョン 0.2 では、CodeBuild はビルド環境のデフォルトシェルの同じインスタンスですべてのビルドコマンドを実行します。
0.1	これは、ビルド仕様形式の最初の定義です。

バッチビルドのビルド仕様 (buildspec) のリファレンス

このトピックでは、バッチビルドプロパティの ビルド仕様 (buildspec) リファレンスを示します。

バッチ

オプションのマッピング。プロジェクトのバッチビルド設定。

batch/fast-fail

オプション。1つ以上のビルドタスクが失敗した場合のバッチビルドの動作を指定します。 false

デフォルト値。実行中のすべてのビルドが完了します。

true

ビルドタスクの1つが失敗すると、実行中のすべてのビルドが停止します。

デフォルトでは、すべてのバッチビルドタスクは、buildspec ファイルで指定された、env や phases などのビルド設定で実行されます。デフォルトのビルド設定をオーバーライドするには、 「env」値または別の buildspec ファイルを「batch/*<batch-type*>/buildspec」パラメータで 指定します。

batch プロパティの内容は、指定されたバッチビルドのタイプによって異なります。可能なバッチ ビルドタイプは次のとおりです。

- batch/build-graph
- <u>batch/build-list</u>
- batch/build-matrix
- batch/build-fanout

batch/build-graph

ビルドグラフを定義します。ビルドグラフは、バッチ内の他のタスクに依存する一連のタスクを定義 します。詳細については、「ビルドグラフ」を参照してください。

この要素には、ビルドタスクの配列が含まれます。各ビルドタスクには、以下のプロパティが含まれ ます。

バッチのビルド仕様 (buildspec) に関するリファレンス

identifier

必須。タスクの識別子。

buildspec

オプション。このタスクに使用する buildspec ファイルのパスとファイル名。このパラメータを 指定しないと、現在の buildspec ファイルが使用されます。

debug-session

オプション。セッションデバッグがこのバッチビルドで有効かどうかを示す、ブール値。セッ ションデバッグの詳細については、「<u>Session Manager を使用したビルドのデバッグ</u>」を参照し てください。

false

セッションデバッグは無効です。

true

セッションデバッグは有効です。

depend-on

オプション。このタスクが依存するタスク識別子の配列。このタスクは、これらのタスクが完了 するまで実行されません。

オプション。タスクのビルド環境がオーバーライドされます。次のプロパティが含まれています。

compute-type

タスクに使用するコンピューティングタイプの識別子。指定できる値については、「<u>the</u> <u>section called "ビルド環境のコンピューティングモードおよびタイプ"</u>」の「computeType」 を参照してください。

フリート

タスクに使用するフリートの識別子。詳細については「<u>the section called "リザーブドキャパ</u> シティキャパシティフリートでビルドを実行"」を参照してください。

ユーザーガイド

env

イメージ

タスクに使用するイメージの識別子。指定できる値については、「<u>the section called</u> <u>"CodeBuild に用意されている Docker イメージ"</u>」の「イメージ識別子」を参照してくださ い。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロジェ クトを使用して Docker イメージをビルドする場合にのみ、true に設定します。そうしな いと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設定 はfalse です。

type

タスクに使用する環境タイプの識別子です。指定できる値については、「<u>the section called</u> <u>"ビルド環境のコンピューティングモードおよびタイプ"</u>」の「環境タイプ」を参照してくださ い。

variables

ビルド環境に存在する環境変数。詳細については「env/variables」を参照してください。

Note

コンピューティングタイプとフリートは、1 つのビルドの同じ ID で提供できないことに 注意してください。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

false

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

true

このビルドタスクが失敗した場合でも、バッチビルドが成功します。

ビルドグラフの buildspec エントリの例を次に示します。

batch:

batch/build-graph

```
fast-fail: false
build-graph:
  - identifier: build1
    env:
      variables:
        BUILD_ID: build1
    ignore-failure: false
  - identifier: build2
    buildspec: build2.yml
    env:
      variables:
        BUILD_ID: build2
    depend-on:
      - build1
  - identifier: build3
    env:
      variables:
        BUILD_ID: build3
    depend-on:
      - build2
  - identifier: build4
    env:
      compute-type: ARM_LAMBDA_1GB
  - identifier: build5
    env:
      fleet: fleet_name
```

batch/build-list

ビルドリストを定義します。ビルドリストは、並行して実行されるタスクの数を定義するために使用 されます。詳細については、「ビルドリスト」を参照してください。

この要素には、ビルドタスクの配列が含まれます。各ビルドタスクには、以下のプロパティが含まれ ます。

identifier

必須。タスクの識別子。

buildspec

オプション。このタスクに使用する buildspec ファイルのパスとファイル名。このパラメータを 指定しないと、現在の buildspec ファイルが使用されます。

debug-session

オプション。セッションデバッグがこのバッチビルドで有効かどうかを示す、ブール値。セッ ションデバッグの詳細については、「<u>Session Manager を使用したビルドのデバッグ</u>」を参照し てください。

false

セッションデバッグは無効です。

true

セッションデバッグは有効です。

env

オプション。タスクのビルド環境がオーバーライドされます。次のプロパティが含まれていま す。

compute-type

タスクに使用するコンピューティングタイプの識別子。指定できる値については、「<u>the</u> <u>section called "ビルド環境のコンピューティングモードおよびタイプ"</u>」の「computeType」 を参照してください。

フリート

タスクに使用するフリートの識別子。詳細については「<u>the section called "リザーブドキャパ</u> <u>シティキャパシティフリートでビルドを実行"</u>」を参照してください。

イメージ

タスクに使用するイメージの識別子。指定できる値については、「<u>the section called</u> <u>"CodeBuild に用意されている Docker イメージ"</u>」の「イメージ識別子」を参照してくださ い。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロジェ クトを使用して Docker イメージをビルドする場合にのみ、true に設定します。そうしな いと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設定 はfalse です。 type

タスクに使用する環境タイプの識別子です。指定できる値については、「<u>the section called</u> <u>"ビルド環境のコンピューティングモードおよびタイプ"</u>」の「環境タイプ」を参照してくださ い。

variables

ビルド環境に存在する環境変数。詳細については「env/variables」を参照してください。

Note

コンピューティングタイプとフリートは、1 つのビルドの同じ ID で提供できないことに 注意してください。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

false

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

true

このビルドタスクが失敗しても、バッチビルドは成功します。

buildspec エントリの例を次に示します。

```
batch:
fast-fail: false
build-list:
        - identifier: build1
        env:
            variables:
            BUILD_ID: build1
        ignore-failure: false
        - identifier: build2
        buildspec: build2.yml
        env:
        variables:
```

```
BUILD_ID: build2
ignore-failure: true
- identifier: build3
env:
    compute-type: ARM_LAMBDA_1GB
- identifier: build4
env:
    fleet: fleet_name
- identifier: build5
env:
    compute-type: GENERAL_LINUX_XLAGRE
```

batch/build-matrix

ビルドマトリックスを定義します。ビルドマトリックスは、並行して実行される異なる構成のタスク を定義します。CodeBuild は、設定可能な組み合わせごとに個別のビルドを作成します。詳細につい ては、「<u>ビルドマトリックス</u>」を参照してください。

static

静的プロパティは、すべてのビルドタスクに適用されます。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

false

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

true

このビルドタスクが失敗しても、バッチビルドは成功します。

env

オプション。ビルド環境はすべてのタスクに対して上書きされます。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロ ジェクトを使用して Docker イメージをビルドする場合にのみ、true に設定します。そう しないと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設 定はfalse です。 type

タスクに使用する環境タイプの識別子です。指定できる値については、「<u>the section</u> <u>called "ビルド環境のコンピューティングモードおよびタイプ"</u>」の「環境タイプ」を参照 してください。

dynamic

動的プロパティはビルドマトリックスを定義します。

buildspec

オプション。これらのタスクに使用する buildspec ファイルのパスとファイル名を含む配列。 このパラメータを指定しないと、現在の buildspec ファイルが使用されます。

env

オプション。これらのタスクでは、ビルド環境が上書きされます。

compute-type

これらのタスクに使用するコンピューティングタイプの識別子を含む配列。指定できる 値については、「<u>the section called "ビルド環境のコンピューティングモードおよびタイ</u> <u>プ</u>"」の「computeType」を参照してください。

イメージ

これらのタスクに使用するイメージの識別子を含む配列。指定できる値については、「<u>the</u> <u>section called "CodeBuild に用意されている Docker イメージ"</u>」の「イメージ識別子」を 参照してください。

variables

これらのタスクのビルド環境に存在する環境変数を含む配列。詳細については、 「env/variables」を参照してください。

ビルドマトリックス「buildspec」のエントリの例を次に示します。

batch: build-matrix: static: ignore-failure: false dynamic: buildspec:

詳細については、「ビルドマトリックス」を参照してください。

batch/build-fanout

ビルドファンアウトを定義します。ビルドファンアウトは、並行して実行される複数のビルドに分割 されるタスクを定義するために使用されます。詳細については、「<u>バッチビルドで並列テストを実行</u> する」を参照してください。

この要素には、複数のビルドに分割できるビルドタスクが含まれています。build-fanout セク ションには、次のプロパティが含まれています。

並列処理

必須。テストを並行して実行するビルドの数。

ignore-failure

オプション。ファンアウトビルドタスクの失敗を無視できるかどうかを示すブール値。この ignore-failure の値は、すべてのファンアウトビルドに適用されます。

false

デフォルト値。ファンアウトビルドタスクが失敗すると、バッチビルドは失敗します。 true

ファンアウトビルドタスクが失敗しても、バッチビルドは成功する可能性があります。

ビルドファンアウト buildspec エントリの例を次に示します。

version: 0.2

batch:

fast-fail: false

```
build-fanout:
     parallelism: 5
     ignore-failure: false
phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run ∖
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
```

詳細については、<u>ファンアウトの構築</u>および<u>codebuild-tests-run CLI コマンドを使用する</u>を参 照してください。

のビルド環境リファレンス AWS CodeBuild

を呼び出し AWS CodeBuild てビルドを実行するときは、ビルド環境に関する情報を入力する必要が あります。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステ ム、プログラミング言語ランタイム、およびツールの組み合わせを表します。ビルド環境の仕組みに ついては、「CodeBuild の仕組み」を参照してください。

ビルド環境には Docker イメージが含まれています。詳細については、Docker Docs ウェブサイトの Docker 用語集を参照してください。

ビルド環境について CodeBuild に情報を提供する場合は、サポートされているリポジトリタイプの Docker イメージの識別子を指定します。これには、CodeBuild Docker イメージリポジトリ、Docker Hub で公開されているイメージ、および AWS アカウントがアクセス許可を持つ Amazon Elastic Container Registry (Amazon ECR) リポジトリが含まれます。

- CodeBuild Docker イメージリポジトリに格納されている Docker イメージは、サービスで使用す るために最適化されているため、使用することをお勧めします。詳細については、「<u>CodeBuild に</u> 用意されている Docker イメージ」を参照してください。
- Docker Hub に保存されて公開されている Docker イメージの識別子を取得するには、Docker Docs ウェブサイトの Searching for Repositories を参照してください。
- AWS アカウントの Amazon ECR リポジトリに保存されている Docker イメージの操作方法については、Amazon ECR のサンプル を参照してください。

Docker イメージ識別子に加えて、ビルド環境で使用する一連のコンピューティングリソースも指定 します。詳細については、「<u>ビルド環境のコンピューティングモードおよびタイプ</u>」を参照してくだ さい。

トピック

- CodeBuild に用意されている Docker イメージ
- ビルド環境のコンピューティングモードおよびタイプ
- ビルド環境のシェルとコマンド
- ビルド環境の環境変数
- ビルド環境のバックグラウンドタスク

CodeBuild に用意されている Docker イメージ

サポートされているイメージは、CodeBuild で利用できるイメージの最新のメジャーバージョン であり、マイナーバージョンとパッチバージョンのアップデートにより更新されます。CodeBuild は、サポートされているイメージをマシンの Amazon マシンイメージ (AMI) にキャッシュする ことで、ビルドのプロビジョニング時間を最適化します。キャッシュを有効に活用し、ビルドの プロビジョニング時間を最小限に抑えたい場合は、aws/codebuild/amazonlinux-x86_64standard:4.0-1.0.0 のようなより詳細なバージョンではなく、CodeBuild コンソールの [イメー ジバージョン] セクションで、[常にこのランタイムバージョンの最新イメージを使用する] を選択し ます。

トピック

- 現在の Docker イメージのリストを取得
- EC2 コンピューティングイメージ
- Lambda コンピューティングイメージ
- 非推奨の CodeBuild イメージ
- 使用可能なランタイム
- ランタイムバージョン

現在の Docker イメージのリストを取得

CodeBuild は Docker イメージのリストを頻繁に更新して最新のイメージを追加し、古いイメージを 廃止します。最新のリストを取得するには、次のいずれかを実行します。

- CodeBuild コンソールの [Create build project] (ビルドプロジェクトの作成) ウィザードまたは [Edit Build Project] (ビルドプロジェクトの編集) ページで、環境イメージとして [Managed image] (マ ネージド型イメージ) を選択します。[オペレーティングシステム]、[ランタイム]、[ランタイムバー ジョン] の各ドロップダウンリストで適切な選択を行います。詳細については、「ビルドプロジェ クトの作成 (コンソール)」または「ビルドプロジェクトの設定の変更 (コンソール)」を参照してく ださい。
- で AWS CLI、 list-curated-environment-images コマンドを実行します。

aws codebuild list-curated-environment-images

 AWS SDKs、ターゲットプログラミング言語の ListCuratedEnvironmentImagesオペレー ションを呼び出します。詳細については、「<u>AWS SDKsとツールのリファレンス</u>」を参照してく ださい。

EC2 コンピューティングイメージ

AWS CodeBuild は、CodeBuild の EC2 コンピューティングで使用できる以下の Docker イメージを サポートしています。

Note

Windows Server Core 2019 プラットフォームの基本イメージは、以下のリージョンでのみ利 用可能です。

- 米国東部(バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- ・ 欧州 (アイルランド)

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-sta ndard:4.0	al/standard/4.0
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-sta ndard:5.0	al/standard/5.0
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-sta ndard:corretto8	al/standard/corretto8
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-sta ndard:corretto11	al/standard/corretto11

AWS CodeBuild

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-st andard:2.0	al/aarch64/standard/2.0
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-st andard:3.0	al/aarch64/standard/3.0
Ubuntu 20.04	aws/codebuild/stan dard:5.0	ubuntu/standard/5.0
Ubuntu 22.04	aws/codebuild/stan dard:6.0	ubuntu/standard/6.0
Ubuntu 22.04	aws/codebuild/stan dard:7.0	ubuntu/standard/7.0
Windows Server Core 2019	aws/codebuild/wind ows-base:2019-1.0	該当なし
Windows Server Core 2019	aws/codebuild/wind ows-base:2019-2.0	該当なし
Windows Server Core 2019	aws/codebuild/wind ows-base:2019-3.0	該当なし
Windows Server Core 2022	aws/codebuild/wind ows-base:2022-1.0	該当なし
macOS	aws/codebuild/macos- arm-base:14	該当なし

Note

2024 年 11 月 22 日、Linux ベースの標準ランタイムイメージのエイリアスが から amazonlinux2に更新されましたamazonlinux。以前のエイリアスは引き続き有効である ため、手動更新は必要ありません。

Lambda コンピューティングイメージ

AWS CodeBuild は、CodeBuild で AWS Lambda コンピューティングに使用できる以下の Docker イ メージをサポートしています。

aarch64 アーキテクチャ

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:dotn et6	al-lambda/aarch64/dotnet6
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:dotn et8	<u>al-lambda/aarch64/dotnet8</u>
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:go1. 21	al-lambda/aarch64/go1.21
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:go1. 24	al-lambda/aarch64/go1.24
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la	al-lambda/aarch64/corretto11
プラットフォーム	イメージ識別子	定義
-------------------	--	------------------------------
	<pre>mbda-standard:corr etto11</pre>	
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:corr etto17	al-lambda/aarch64/corretto17
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:corr etto21	al-lambda/aarch64/corretto21
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:node js18	al-lambda/aarch64/nodejs18
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:node js20	al-lambda/aarch64/nodejs20
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:node js22	al-lambda/aarch64/nodejs22
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:pyth on3.11	al-lambda/aarch64/python3.11

プラットフォーム	イメージ識別子	定義
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:pyth on3.12	al-lambda/aarch64/python3.12
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:pyth on3.13	al-lambda/aarch64/python3.13
Amazon Linux 2	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:ruby 3.2	al-lambda/aarch64/ruby3.2
Amazon Linux 2023	aws/codebuild/amaz onlinux-aarch64-la mbda-standard:ruby 3.4	al-lambda/aarch64/ruby3.4

x86_64 アーキテクチャ

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:dotne t6	<u>al-lambda/x86_64/dotnet6</u>
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:dotne t8	al-lambda/x86_64/dotnet8

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:go1.21	al-lambda/x86_64/go1.21
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:go1.24	al-lambda/x86_64/go1.24
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:corre tto11	al-lambda/x86_64/corretto11
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:corre tto17	al-lambda/x86_64/corretto17
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:corre tto21	<u>al-lambda/x86_64/corretto21</u>
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:nodej s18	al-lambda/x86_64/nodejs18
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:nodej s20	al-lambda/x86_64/nodejs20

プラットフォーム	イメージ識別子	定義
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:nodej s22	al-lambda/x86_64/nodejs22
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:pytho n3.11	al-lambda/x86_64/python3.11
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:pytho n3.12	al-lambda/x86_64/python3.12
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:pytho n3.13	al-lambda/x86_64/python3.13
Amazon Linux 2	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:ruby3 .2	al-lambda/x86_64/ruby3.2
Amazon Linux 2023	aws/codebuild/amaz onlinux-x86_64-lam bda-standard:ruby3 .4	al-lambda/x86_64/ruby3.4

非推奨の CodeBuild イメージ

廃止イメージとは、CodeBuild によってキャッシュまたは更新されなくなったイメージです。廃止 イメージは、マイナーバージョンやパッチバージョンの更新の対象外となって、更新されなくなる ため、使用すると安全ではない場合があります。CodeBuild プロジェクトが古いイメージバージョン を使用するように設定されている場合、プロビジョニングプロセスはこの Docker イメージをダウン ロードし、それを使用してコンテナ化されたランタイム環境を作成します。これにより、プロビジョ ニング時間と全体的なビルド時間が長くなる可能性があります。

CodeBuild は、以下の Docker イメージを廃止しました。これらのイメージは引き続き使用できますが、ビルドホストにキャッシュされないため、プロビジョニング時間が長くなります。

プラットフォーム	イメージ識別子	定義	廃止日
Amazon Linux 2	aws/codebuild/ amazonlinux2- x86_64-st andard:3.0	al2/standard/3.0	2023 年 5 月 9 日
Ubuntu 18.04	aws/codebuild/ standard:4.0	ubuntu/standard/4.0	2023 年 3 月 31 日
Amazon Linux 2	aws/codebuild/ amazonlinux2- aarch64-s tandard:1.0	al2/aarch64/standa rd/1.0	2023 年 3 月 31 日
Ubuntu 18.04	aws/codebuild/ standard:3.0	ubuntu/standard/3.0	2022 年 6 月 30 日
Amazon Linux 2	aws/codebuild/ amazonlinux2- x86_64-st andard:2.0	al2/standard/2.0	2022 年 6 月 30 日

トピック

- 使用可能なランタイム
- ランタイムバージョン

使用可能なランタイム

buildspec ファイルの runtime-versions セクションで 1 つ以上のランタイムを指定できます。ラ ンタイムが別のランタイムに依存している場合は、依存しているランタイムを buildspec ファイル で指定することもできます。buildspec ファイルでランタイムを指定しない場合は、CodeBuild によ り、使用するイメージのデフォルトのランタイムが選択されます。1 つ以上のランタイムを指定する と、CodeBuild により、それらのランタイムのみが使用されます。依存関係のあるランタイムが指定 されていない場合、CodeBuild により、依存関係のあるランタイムの選択が試みられます。詳細につ いては、「Specify runtime versions in the buildspec file」を参照してください。

トピック

- Linux イメージのランタイム
- macOS イメージランタイム
- Windows イメージのランタイム

Linux イメージのランタイム

次の表に、使用可能なランタイムと、それらをサポートする標準 Linux イメージを示します。

Ubuntu および Amazon Linux プラットフォームランタイム

ランタイム名	バージョン	イメージ
dotnet	3.1	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0
	5.0	Ubuntu standard:5.0
	6.0	Amazon Linux 2 x86_64 Lambda stanc dotnet6
		Amazon Linux 2 AArch64 Lambda star dotnet6
		Amazon Linux 2 x86_64 standard: 4.0

AWS CodeBuild

ランタイム名	バージョン	イメージ
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard 3.0
		Ubuntu standard:6.0
		Ubuntu standard:7.0
	8.0	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard 3.0
		Ubuntu standard:7.0
golang	1.12	Amazon Linux 2 AArch64 standard: 2.0
	1.13	Amazon Linux 2 AArch64 standard: 2.0
	1.14	Amazon Linux 2 AArch64 standard: 2.0
	1.15	Ubuntu standard:5.0
	1.16	Ubuntu standard:5.0
	1.18	Amazon Linux 2 x86_64 standard: 4.0
		Ubuntu standard:6.0

ランタイム名	バージョン	イメージ
	1.20	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	1.21	Amazon Linux 2 x86_64 Lambda standa go1.21
		Amazon Linux 2 AArch64 Lambda stan go1.21
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	1.22	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	1.23	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0

バージョン	イメージ
1.24	Amazon Linux 2023 x86_64 Lambda 標準:go1.24
	Amazon Linux 2023AArch 64Lambda 標準:go1.24
corretto8	Amazon Linux 2 x86_64 standard: corretto8
	Amazon Linux 2023 x86_64 standard: 5.0
	Amazon Linux 2 AArch64 standard: 2.0
	Amazon Linux 2023 AArch64 standard 3.0
	Ubuntu standard:5.0
	Ubuntu standard:7.0
	バージョン 1.24 corretto8

ランタイム名 バージョン イメージ Corretto11 Amazon Linux 2 x86_64 stand corretto11 Amazon Linux 2 x86_64 Lamb corretto11 Amazon Linux 2 x86_64 Lamb corretto11 Amazon Linux 2 x86_64 Lamb corretto11	
corretto11 Amazon Linux 2 x86_64 stand corretto11 Amazon Linux 2 x86_64 Lamb corretto11 Amazon Linux 2 x86_64 Lamb corretto11 Amazon Linux 2023 x86_64 stand corretto11	
Amazon Linux 2 x86_64 Lamb corretto11 Amazon Linux 2023 x86_64 st	ard:
Amazon Linux 2023 x86_64 si	da stand
5.0	andard:
Amazon Linux 2 AArch64 Lan corretto11	bda stan
Amazon Linux 2 AArch64 standard: 2.0	
Amazon Linux 2023 AArch64 3.0	standard:
Ubuntu standard:5.0	
Ubuntu standard:7.0	

ランタイム名	バージョン	イメージ
	corretto17	Amazon Linux 2 x86_64 Lambda standa corretto17
		Amazon Linux 2 AArch64 Lambda stan corretto17
		Amazon Linux 2 x86_64 standard: 4.0
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:6.0
		Ubuntu standard:7.0
	corretto21	Amazon Linux 2 x86_64 Lambda stand corretto21
		Amazon Linux 2 AArch64 Lambda stan corretto21
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
NodeJS	10	Amazon Linux 2 AArch64 standard: 2.0

ランタイム名	バージョン	イメージ
	12	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0
	14	Ubuntu standard:5.0
	16	Amazon Linux 2 x86_64 standard: 4.0
		Ubuntu standard:6.0
	18	Amazon Linux 2 x86_64 Lambda stand nodejs18
		Amazon Linux 2 AArch64 Lambda stan nodejs18
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	20	Amazon Linux 2 x86_64 Lambda stand nodejs20
		Amazon Linux 2 AArch64 Lambda stan nodejs20
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0

AWS CodeBuild

ランタイム名	バージョン	イメージ	
	22	Amazon Linux 2023 x86_64 Lambda 標準:nodejs22	
		Amazon Linux 2023AArch 64Lambda 標準:nodejs22	
		Amazon Linux 2023 x86_64 standard: 5.0	
		Amazon Linux 2023 AArch64 standard: 3.0	
		Ubuntu standard:7.0	
php	73	Amazon Linux 2 AArch64 standard: 2.0	
		Ubuntu standard:5.0	
	7.4	Amazon Linux 2 AArch64 standard: 2.0	
		Ubuntu standard:5.0	
	8.0	Ubuntu standard:5.0	
	8.1	Amazon Linux 2 x86_64 standard: 4.0	
		Amazon Linux 2023 AArch64 standard: 3.0	
		Ubuntu standard:6.0	

AWS CodeBuild

ランタイム名	バージョン	イメージ
	8.2	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	8.3	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
python	3.7	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0
	3.8	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0

AWS CodeBuild

ランタイム名	バージョン	イメージ
	3.9	Amazon Linux 2 x86_64 standard: 4.0
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2 AArch64 standard: 2.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:5.0
		Ubuntu standard:7.0
	3.10	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:6.0
		Ubuntu standard:7.0
	3.11	Amazon Linux 2 x86_64 Lambda stand python3.11
		Amazon Linux 2 AArch64 Lambda stan python3.11
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	3.12	Amazon Linux 2 x86_64 Lambda standa python3.12
		Amazon Linux 2 AArch64 Lambda stan python3.12
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	3.13	Amazon Linux 2023 x86_64 Lambda 標準:python3.13
		Amazon Linux 2023AArch 64Lambda 標準:python3.13
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
ruby	2.6	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0
	2.7	Amazon Linux 2 AArch64 standard: 2.0
		Ubuntu standard:5.0

ランタイム名	バージョン	イメージ
	3.1	Amazon Linux 2 x86_64 standard: 4.0
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:6.0
		Ubuntu standard:7.0
	3.2	Amazon Linux 2 x86_64 Lambda stand ruby3.2
		Amazon Linux 2 AArch64 Lambda stan ruby3.2
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0
	3.3	Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard: 3.0
		Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	3.4	Amazon Linux 2023 x86_64 Lambda 標準:ruby3.4
		Amazon Linux 2023AArch 64Lambda 標準:ruby3.4
		Amazon Linux 2023 x86_64 standard: 5.0
		Amazon Linux 2023 AArch64 standard 3.0
		Ubuntu standard:7.0

macOS イメージランタイム

▲ Important

Mac ビルド用の CodeBuild キュレーションイメージには、macOS と Xcode がプリインス トールされています。Xcode ソフトウェアを使用することにより、「<u>XcodeとApple SDK</u> <u>の利用規約</u>」を承認、理解し、同意したものとみなされます。契約条件に同意しない場合 は、Xcode ソフトウェアを使用しないでください。代わりに、独自の Amazon マシンイメー ジ (AMI) を指定します。詳細については、<u>リザーブドキャパシティの macOS フリートを設</u> 定するにはどうすればよいですか。を参照してください。

次の表は、macOS でサポートされている利用可能なランタイムを示しています。

ランタイム名	バージョン	イメージ	追加のメモ
bash	3.2.57	macos-arm-base:14	
		macos-arm-base:15	
clang	15.0.0	macos-arm-base:14	

macOS プラットフォームランタイム

ランタイム名	バージョン	イメージ	追加のメモ
	16.0.0	macos-arm-base:15	
dotnet sdk	8.0.406	macos-arm-base:14	
		macos-arm-base:15	
gcc	11.5.0	macos-arm-base:14	gcc-11 エイリアスを
		macos-arm-base:15	使用して利用可能
	12.4.0	macos-arm-base:14	gcc-12 エイリアスを
		macos-arm-base:15	使用して利用り能
	13.3.0	macos-arm-base:14	gcc-13 エイリアスを
		macos-arm-base:15	使用して利用り能
	14.2.0	macos-arm-base:14	gcc-14 エイリアスを
		macos-arm-base:15	使用して利用り能
gnu	11.5.0	macos-arm-base:14	gfortran-11 エイ
		macos-arm-base:15	リアスを使用して利 用可能
	12.4.0	macos-arm-base:14	gfortran-12 エイ
		macos-arm-base:15	リアスを使用して利 用可能
	13.3.0	macos-arm-base:14	gfortran-13 エイ
		macos-arm-base:15	リアスを使用して利 用可能
	14.2.0	macos-arm-base:14	gfortran-14 エイ
		macos-arm-base:15	リアスを使用して利 用可能

AWS CodeBuild

ランタイム名	バージョン	イメージ	追加のメモ
golang	1.22.12	macos-arm-base:14	
		macos-arm-base:15	
	1.23.6	macos-arm-base:14	
		macos-arm-base:15	
	1.24.0	macos-arm-base:14	
		macos-arm-base:15	
java	Corretto8	macos-arm-base:14	
		macos-arm-base:15	
	Corretto11	macos-arm-base:14	
		macos-arm-base:15	
	Corretto17	macos-arm-base:14	
		macos-arm-base:15	
	Corretto21	macos-arm-base:14	
		macos-arm-base:15	
kotlin	2.1.10	macos-arm-base:14	
		macos-arm-base:15	
mono	6.12.0	macos-arm-base:14	
		macos-arm-base:15	
nodejs	18.20.7	macos-arm-base:14	

AWS CodeBuild

ランタイム名	バージョン	イメージ	追加のメモ
	20.18.3	macos-arm-base:14	
		macos-arm-base:15	
	22.14.0	macos-arm-base:14	
		macos-arm-base:15	
perl	5.34.1	macos-arm-base:14	
		macos-arm-base:15	
php	8.1.31	macos-arm-base:14	
	8.2.27	macos-arm-base:14	
		macos-arm-base:15	
	8.3.17	macos-arm-base:14	
		macos-arm-base:15	
	8.4.4	macos-arm-base:14	
		macos-arm-base:15	
python	3.9.21	macos-arm-base:14	
	3.10.16	macos-arm-base:14	
		macos-arm-base:15	
	3.11.11	macos-arm-base:14	
		macos-arm-base:15	
	3.12.9	macos-arm-base:14	
		macos-arm-base:15	

ランタイム名	バージョン	イメージ	追加のメモ
	3.13.2	macos-arm-base:14	
		macos-arm-base:15	
ruby	3.1.6	macos-arm-base:14	
	3.2.7	macos-arm-base:14	
		macos-arm-base:15	
	3.3.7	macos-arm-base:14	
		macos-arm-base:15	
	3.4.2	macos-arm-base:14	
		macos-arm-base:15	
rust	1.85.0	macos-arm-base:14	
		macos-arm-base:15	
swift	5.10.0.13	macos-arm-base:14	
	6.0.3.1.10	macos-arm-base:14	
Xcode	15.4	macos-arm-base:14	
	16.2	macos-arm-base:15	

Windows イメージのランタイム

Windows Server Core 2019 のベースイメージには、以下のランタイムが含まれています。

Windows プラットフォームのランタイム

ランタイム名	Windows Server Core 20 1.0 バージョン	Windows Server Core 20 2.0 バージョン	Windows Server Core 20 3.0 バージョン	19 stai
dotnet	3.1	3.1	8.0	

AVVS CodeBuild

ランタイム名	Windows Server Core 20 1.0 バージョン	Windows Server Core 20 2.0 バージョン	Windows Server Core 2019 sta 3.0 バージョン
	5.0	6.0	
		7.0	
dotnet sdk	3.1	3.1	8.0
	5.0	6.0	
		7.0	
golang	1.14	1.18	1.21
			1.22
			1.23
gradle	6.7	7.6	8.12
java	Corretto11	Corretto11	Corretto8
		Corretto17	Corretto11
			Corretto17
			Corretto21
Maven (メイヴン)	3.6	3.8	3.9
nodejs	14.15	16.19	20.18
			22.13
php	7.4	8.1	8.3
			8.4
powershell	7.1	7.2	7.4

r

ランタイム名	Windows Server Core 20 1.0 バージョン	Windows Server Core 20 2.0 バージョン	Windows Server Core 20′ 3.0 バージョン	19 sta
python	3.8	3.10	3.10	
			3.11	
			3.12	
			3.13	
ruby	2.7	3.1	3.2	
			3.3	
			3.4	

ランタイムバージョン

buildspec ファイルの <u>runtime-versions</u> セクションでランタイムを指定するときは、特定のバー ジョン、特定のメジャーバージョンと最新のマイナーバージョン、または最新バージョンを指定でき ます。次の表に、使用可能なランタイムとその指定方法を示します。すべてのランタイムバージョン が、すべてのイメージで使用できるわけではありません。ランタイムバージョンの選択は、カスタム イメージでもサポートされていません。詳細については、「<u>使用可能なランタイム</u>」を参照してくだ さい。プリインストールされたランタイムバージョンの代わりにカスタムランタイムバージョンをイ ンストールして使用する場合は、「<u>カスタムランタイムバージョン</u>」を参照してください。

Ubuntu および Amazon Linux 2 プラットフォームランタイムバージョン

ランタイム名	バージョン	特定の バージョン	特定のメジャー バージョン と最新のマイ ナーバージョン	最新バージョン	
android	28	android: 28	android: 28.x	android: lates	
	29	android: 29	android: 29.x		
dotnet	3.1	dotnet: 3.1	dotnet: 3.x	dotnet: latest	

ランタイム名	バージョン	特定の バージョン	特定のメジャー バージョン と最新のマイ ナーバージョン	最新バージョン
	5.0	dotnet: 5.0	dotnet: 5.x	
	6.0	dotnet: 6.0	dotnet: 6.x	
	8.0	dotnet: 8.0	dotnet: 8.x	
golang	1.12	golang: 1.12	golang: 1.x	golang: latest
	1.13	golang: 1.13		
	1.14	golang: 1.14		
	1.15	golang: 1.15		
	1.16	golang: 1.16		
	1.18	golang: 1.18		
	1.20	golang: 1.20		
	1.21	golang: 1.21		
	1.22	golang: 1.22		
	1.23	golang: 1.23		
	1.24	golang: 1.24		
java	corretto8	java: corretto	java: corretto .x	java: latest
	corretto11	java: corretto 1	java: corretto 1.x	
	corretto17	java: corretto 7	java: corretto 7.x	

ランタイム名	バージョン	特定の バージョン	特定のメジャー バージョン と最新のマイ ナーバージョン	最新バージョン
	corretto21	java: corretto 1	java: corretto 1.x	
NodeJS	10	nodejs: 10	nodejs: 10.x	nodejs: latest
	12	nodejs: 12	nodejs: 12.x	
	14	nodejs: 14	nodejs: 14.x	
	16	nodejs: 16	nodejs: 16.x	
	18	nodejs: 18	nodejs: 18.x	
	20	nodejs: 20	nodejs: 20.x	
	22	nodejs: 22	nodejs: 22.x	
php	73	php: 7.3	php: 7.x	php: latest
	7.4	php: 7.4		
	8.0	php: 8.0	php: 8.x	
	8.1	php: 8.1		
	8.2	php: 8.2		
	8.3	php: 8.3		
python	3.7	python: 3.7	python: 3.x	python: latest
	3.8	python: 3.8		
	3.9	python: 3.9		
	3.10	python: 3.10		

ランタイム名	バージョン	特定の バージョン	特定のメジャー バージョン と最新のマイ ナーバージョン	最新バージョン
	3.11	python: 3.11		
	3.12	python: 3.12		
	3.13	python: 3.13		
ruby	2.6	ruby: 2.6	ruby: 2.x	ruby: latest
	2.7	ruby: 2.7		
	3.1	ruby: 3.1	ruby: 3.x	
	3.2	ruby: 3.2		
	3.3	ruby: 3.3		
	3.4	ruby: 3.4		

ビルド仕様を使用して、installビルドフェーズ中に他のコンポーネント (Apache Maven AWS CLI、Apache Ant、Mocha、RSpec など) をインストールできます。詳細については、「<u>buildspec</u> の例」を参照してください。

カスタムランタイムバージョン

CodeBuild マネージドイメージにプリインストールされたランタイムバージョンを使用する代わり に、選択したカスタムバージョンをインストールして使用できます。次の表に、利用可能なランタイ ムとその指定方法を示します。

Note

カスタムランタイムバージョンの選択は、Ubuntu イメージと Amazon Linux イメージでのみ サポートされています。

カスタムランタイムバージョン

ランタイム名	Syntax	例
dotnet	<major>.<minor>.<patch></patch></minor></major>	5.0.408
golang	<major>.<minor></minor></major>	1.19
	<major>.<minor>.<patch></patch></minor></major>	1.19.1
java	corretto< <i>major></i>	corretto15
nodejs	<major></major>	14
	<major>.<minor></minor></major>	14.21
	<major>.<minor>.<patch></patch></minor></major>	14.21.3
php	<major>.<minor>.<patch></patch></minor></major>	8.0.30
python	<major></major>	3
	<major>.<minor></minor></major>	3.7
	<major>.<minor>.<patch></patch></minor></major>	3.7.16
ruby	<major>.<minor>.<patch></patch></minor></major>	3.0.6

カスタムランタイム buildspec の例

以下は、カスタムランタイムバージョンを指定する buildspec の例です。

version: 0.2
phases:
 install:
 runtime-versions:
 java: corretto15
 php: 8.0.30
 ruby: 3.0.6
 golang: 1.19
 python: 3.7
 nodejs: 14

dotnet: 5.0.408

ビルド環境のコンピューティングモードおよびタイプ

CodeBuild では、CodeBuild がビルドの実行に使用するコンピューティングとランタイム環境イメー ジを指定できます。コンピューティングとは、CodeBuild によって管理および保守されるコンピュー ティングエンジン (CPU、メモリ、およびオペレーティングシステム) を指します。ランタイム環境 イメージは、選択したコンピュートプラットフォーム上で実行されるコンテナイメージで、ビルドで 必要になる可能性があるその他のツール (AWS CLIなど) が含まれています。

トピック

- <u>コンピューティングについて</u>
- リザーブドキャパシティ環境タイプについて
- オンデマンド環境タイプについて

コンピューティングについて

CodeBuild には EC2 モードと AWS Lambda コンピューティングモードが用意されています。EC2 は、ビルド中の柔軟性を最適化し AWS Lambda 、起動速度を最適化します。 は、起動レイテンシー が低いため、より高速なビルド AWS Lambda をサポートします。 AWS Lambda また、 は自動的に スケールするため、ビルドはキュー内で実行されるのを待つことはありません。詳細については、 「AWS Lambda コンピューティングでビルドを実行する」を参照してください。

EC2 コンピューティングモードでは、オンデマンドまたはリザーブドキャパシティフリートを使用 してビルドを実行できます。オンデマンドフリートでは、 や などのBUILD_GENERAL1_SMALL事前 定義されたコンピューティングタイプを選択できますBUILD_GENERAL1_LARGE。詳細については、 「<u>オンデマンド環境タイプについて</u>」を参照してください。リザーブドキャパシティフリートの場 合、vCPU、メモリ、ディスク容量などのコンピューティング設定を選択できます。設定を指定した 後、CodeBuild は要件に合ったサポートされているコンピューティングタイプを選択します。詳細に ついては、「リザーブドキャパシティ環境タイプについて」を参照してください。

リザーブドキャパシティ環境タイプについて

AWS CodeBuild は、リザーブドキャパシティフリート用の Linux x86、Arm、GPU、Windows、macOS 環境タイプを提供します。次の表は、使用可能なマシンタイ プ、メモリ、vCPUsディスク容量をリージョン別にソートしたものです。

US East (N. Virginia)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

4

Linux EC2

リナックス	96	192 GiB	
リナックス	48	96 GiB	
リナックス	72	144 GiB	

			量	プ	ティングイ ンスタンス タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib

8 GiB

128 GB

GENERAL

「メモリ」

ディスク容

マシンタイ

環境タイプ

vCPUs

reserved.

x86-64.4c

pu.8gib

API バージョン 2016-10-06 269

コンピュー

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved. gpu.32cpu .128gib.n vme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	reserved. gpu.64cpu .256gib.n vme

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	96	384 GiB	3785 GB (SSD)	NVME	reserved. gpu.96cpu .384gib.n vme
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved. arm.m2.12 cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。

US East (Ohio)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
---------	-------	--------	------------	------------	-------------------------------------
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved. gpu.32cpu .128gib.n vme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved. arm.m2.12 cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。

US West (Oregon)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved. gpu.32cpu .128gib.n vme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	reserved. gpu.64cpu .256gib.n vme

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved. arm.m2.12 cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。

Asia Pacific (Tokyo)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。

Asia Pacific (Mumbai)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

AWS	Code	Bu	ild		
	_			•	

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。

Asia Pacific (Singapore)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

Linux EC2	2	4 GiB
Linux EC2	4	8 GiB

					タイプ
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

ディスク容

量

マシンタイ

プ

「メモリ」

環境タイプ

vCPUs

API バージョン 2016-10-06 296

コンピュー

ティングイ ンスタンス

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。 Asia Pacific (Sydney)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

AWS CodeBuild								
	環境タイプ	vCPUs						

					タイプ
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e

「メモリ」

ディスク容

量

マシンタイ

プ

コンピュー

ティングイ ンスタンス

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved. arm.m2.12 cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。 Europe (Frankfurt)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

2

Linux EC2

S	CodeBuild				
	環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ
	リナックス	16	32 GiB	256 GB	GENERAL
	リナックス	36	72 GiB	256 GB	GENERAL
	リナックス	48	96 GiB	512 GB	GENERAL
	リナックス	72	144 GiB	824 GB	GENERAL
	リナックス	96	192 GiB	824 GB	GENERAL
	リナックス	72	144 GiB	824 GB (SSD)	NVME

4 GiB

64 GB

コンピュー ティングイ ンスタンス

タイプ

reserved. x86-64.16 cpu.32gib

reserved. x86-64.36 cpu.72gib

reserved. x86-64.48 cpu.96gib

reserved. x86-64.72 cpu.144gi

reserved. x86-64.96 cpu.192gi

reserved. x86-64.72

cpu.144gi

reserved. x86-64.2c pu.4gib

b.nvme

b

b

GENERAL

API	バー	ジョ	ン	2016	-10-	06	306

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved. gpu.32cpu .128gib.n vme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。 Europe (Ireland)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
AWS CodeBuild

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

AWS	AWS CodeBuild								
	環境タイプ	vCPUs	「メモリ」						
	リナックス	72	144 GiB						
	リナックス	96	192 GiB						

					cpu.192gi b
リナックス	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

ディスク容

量

824 GB

824 GB

マシンタイ

GENERAL

GENERAL

プ

コンピュー

ティングイ

ンスタンス タイプ

reserved. x86-64.72 cpu.144gi

reserved. x86-64.96

b

AWS CodeBuild

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved. gpu.4cpu. 16gib.nvm e
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved. gpu.8cpu. 32gib.nvm e
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved. gpu.16cpu .64gib.nv me
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved. gpu.32cpu .128gib.n vme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved. gpu.48cpu .192gib.n vme
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、<u>https://aws.amazon.com/codebuild/pricing/</u>を参照してください。 South America (São Paulo)

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
リナックス	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
リナックス	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
リナックス	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
リナックス	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
リナックス	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
リナックス	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
リナックス	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
リナックス	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gi b.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b

環境タイプ	vCPUs	「メモリ」	ディスク容 量	マシンタイ プ	コンピュー ティングイ ンスタンス タイプ
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

料金識別子の詳細については、https://aws.amazon.com/codebuild/pricing/を参照してください。

コンピューティングタイプを選択するには:

- CodeBuild コンソールのコンピューティングフリート設定ページで、vCPUsメモリ、ディスクのいずれかのオプションを選択します。詳細については、「<u>リザーブドキャパシティフリートを作成</u>」 を参照してください。
- で AWS CLI、 create-fleetまたは update-fleet コマンドを実行し、 computeType か ら の値を指定しますATTRIBUTE_BASED_COMPUTE。詳細については、「create-fleet」また は「update-fleet」を参照してください。
- AWS SDKs、ターゲットプログラミング言語の CreateFleetまたは UpdateFleetオペレーションに相当する を呼び出し、の値を computeTypeに指定しますATTRIBUTE_BASED_COMPUTE。
 詳細については、「AWS SDKsとツールのリファレンス」を参照してください。
 - Note

AWS CLI および AWS SDKs では、 などのcomputeType入力を使用し てBUILD_GENERAL1_SMALL、 の代わりにコンピューティングタイプを選択できま すATTRIBUTE_BASED_COMPUTE。詳細については、「<u>オンデマンド環境タイプについて</u>」 を参照してください。

サポートされるインスタンスファミリー

AWS CodeBuild は、リザーブドキャパシティフリートに対して次のインスタンスをサポートしま す。

- 汎用: M5 | M5a | M5ad | M5d | M5dn | M5n | M5zn | M6a | M6g | M6gd | M6i | M6id | M6idn | M6in | M7a | M7g | M7gd | M7i | M7i-flex | M8g | T3 | T3a | T4g
- コンピューティング最適化: C5 | C5a | C5ad | C5d | C5n | C6a | C6g | C6gd | C6gn | C6i | C6id | C6in | C7a | C7g | C7gd | C7gn | C7i | C7i-flex | C8g
- メモリ最適化: R5 | R5a | R5ad | R5b | R5d | R5dn | R5n | R6a | R6g | R6gd | R6i | R6idn | R6in | R6id | R7a | R7g | R7gd | R7i | R7iz | R8g | U-3tb1 | U-6tb1 | U-9tb1 | U-12tb1 | U-18tb1 | U-24tb1 | U7i-6tb | U7i-8tb | U7i-12tb | UU7in-16tbin-24tb | U7in-32tb | X1 X1e | X1X2gd U7in-24tb X2idn X2iedn X2iezn X8g
- ・ストレージ最適化: D3 | D3en | I3 | I3en | I4g | I4i | I7ie | I8g | Im4gn | Is4gen
- 高速コンピューティング: DL1 | DL2q | F1 | F2 | G4ad | G4dn | G5 | G5g | G6 | G6e | Gr6 | Inf1 | Inf2 | P3 | P3dn | P4d | P5 | P5e | P5en | Trn1 | Trn1n | Trn2 | VT1
- ハイパフォーマンスコンピューティング: Hpc6a | Hpc6id | Hpc7a | Hpc7g
- 前の世代: A1

特定のインスタンスタイプでリザーブドキャパシティフリートを作成するには:

- CodeBuild コンソールのコンピューティングフリート設定ページで、キャパシティ設定セクション に移動します。コンピューティング選択モードで手動入力を選択し、コンピューティングインスタ ンスタイプでドロップダウンメニューからインスタンスタイプのいずれかを選択します。詳細につ いては、「リザーブドキャパシティフリートを作成」を参照してください。
- で AWS CLI、 create-fleetまたは update-fleet コマンドを実行し、 の値を computeType に、 CUSTOM_INSTANCE_TYPE の値を指定されたインスタンスタイ プComputeConfigurationinstanceTypeに指定します。詳細については、「create-fleet」ま たは「update-fleet」を参照してください。
- AWS SDKs、ターゲットプログラミング言語の CreateFleetまたは UpdateFleetオペ レーションに相当する を呼び出し、の値を computeType に、 ComputeConfiguration CUSTOM_INSTANCE_TYPEを指定されたインスタンスタイプinstanceTypeに指定します。詳細 については、「<u>AWS SDKsとツールのリファレンス</u>」を参照してください。

オンデマンド環境タイプについて

AWS CodeBuild は、EC2 コンピューティングモード用に次の使用可能なメモリ、vCPUs、ディスク 容量を備えたビルド環境を提供します。

コンピュー ティングタイ プ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
ARM Small ¹	BUILD_GEN ERAL1_SMA LL	ARM_CONTA INER ARM_EC2	4 GiB	2	64 GB
ARM Medium	BUILD_GEN ERAL1_MED IUM	ARM_CONTA INER ARM_EC2	8 GiB	4	128 GB
ARM Large ¹	BUILD_GEN ERAL1_LAR GE	ARM_CONTA INER ARM_EC2	16 GiB	8	128 GB
ARM XLarge	BUILD_GEN ERAL1_XLA RGE	ARM_CONTA INER	64 GiB	32	256 GB
ARM 2XLarge ¹	BUILD_GEN ERAL1_2XL ARGE	ARM_CONTA INER	96 GiB	48	824 GB
Linux Small ¹	BUILD_GEN ERAL1_SMA LL	LINUX_CON TAINER LINUX_EC2	4 GiB	2	64 GB
Linux Medium	BUILD_GEN ERAL1_MED IUM	LINUX_CON TAINER	8 GiB	4	128 GB

コンピュー ティングタイ プ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
		LINUX_EC2			
Linux Large ¹	BUILD_GEN ERAL1_LAR GE	LINUX_CON TAINER LINUX_EC2	16 GiB	8	128 GB
Linux XLarge	BUILD_GEN ERAL1_XLA RGE	LINUX_CON TAINER	72 GiB	36	256 GB
Linux 2xlarge	BUILD_GEN ERAL1_2XL ARGE	LINUX_CON TAINER	144 GiB	72	824 GB (SSD)
Linux GPU Sma	BUILD_GEN ERAL1_SMA LL	LINUX_GPU _CONTAINE R	16 GiB	4	235 GB (SSD)
Linux GPU large	BUILD_GEN ERAL1_LAR GE	LINUX_GPU _CONTAINE R	255 GiB	32	50 GB
Windows Medium ¹	BUILD_GEN ERAL1_MED IUM	WINDOWS_S ERVER_201 9_CONTAIN ER	8 GiB	4	128 GB
		WINDOWS_S ERVER_202 2_CONTAIN ER			
		WINDOWS_E C2			

コンピュー ティングタイ プ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
Windows Large ¹	BUILD_GEN ERAL1_LAR GE	WINDOWS_S ERVER_201 9_CONTAIN ER	16 GiB	8	128 GB
		WINDOWS_S ERVER_202 2_CONTAIN ER WINDOWS_E C2			
WindowsXL arge 1	BUILD_GEN ERAL1_XLA RGE	WINDOWS_S ERVER_202 2_CONTAIN ER	72 GiB	36	256 GB
Windows2X Large 1	BUILD_GEN ERAL1_2XL ARGE	WINDOWS_S ERVER_202 2_CONTAIN ER	144 GiB	72	824 GB

¹各イメージの最新バージョンがキャッシュされます。具体的なバージョンを指定すると、キャッ シュされたバージョンではなく、そのバージョンのプロビジョニングが CodeBuild によって行われ ます。これにより、ビルド時間が長くなることがあります。たとえば、キャッシュのメリットを得る には、aws/codebuild/amazonlinux-x86_64-standard:5.0のような詳細バージョンではな く aws/codebuild/amazonlinux-x86_64-standard:5.0-1.0.0を指定します。

AWS CodeBuild は、 AWS Lambda コンピューティングモード用に以下の使用可能なメモリとディ スク容量を備えたビルド環境を提供します。

コンピューティ ングタイプ	環境 computeTy pe 値	環境タイプ値	「メモリ」	ディスク容量
ARM Lambda 1GB	BUILD_LAM BDA_1GB	ARM_LAMBD A_CONTAIN ER	1 GiB	10 GB
ARM Lambda 2GB	BUILD_LAM BDA_2GB	ARM_LAMBD A_CONTAIN ER	2 GiB	10 GB
ARM Lambda 4GB	BUILD_LAM BDA_4GB	ARM_LAMBD A_CONTAIN ER	4 GiB	10 GB
ARM Lambda 8GB	BUILD_LAM BDA_8GB	ARM_LAMBD A_CONTAIN ER	8 GiB	10 GB
ARM Lambda 10G	BUILD_LAM BDA_10GB	ARM_LAMBD A_CONTAIN ER	10 GiB	10 GB
Linux Lambda 1GE	BUILD_LAM BDA_1GB	LINUX_LAM BDA_CONTA INER	1 GiB	10 GB
Linux Lambda 2GE	BUILD_LAM BDA_2GB	LINUX_LAM BDA_CONTA INER	2 GiB	10 GB
Linux Lambda 4GE	BUILD_LAM BDA_4GB	LINUX_LAM BDA_CONTA INER	4 GiB	10 GB
Linux Lambda 8GE	BUILD_LAM BDA_8GB	LINUX_LAM BDA_CONTA INER	8 GiB	10 GB

コンピューティ ングタイプ	環境 computeTy pe 値	環境タイプ値	「メモリ」	ディスク容量
Linux Lambda 10G	BUILD_LAM BDA_10GB	LINUX_LAM BDA_CONTA INER	10 GiB	10 GB

他の環境タイプを使用する場合は、キャッシュされたイメージを使用してビルド時間を短縮すること をお勧めします。

各ビルド環境にリストされているディスク容量は、CODEBUILD_SRC_DIR 環境変数で指定された ディレクトリでのみ使用できます。

コンピューティングタイプを選択するには:

- CodeBuild コンソールで、[Create build project] (ビルドプロジェクトの作成) ウィザードまたは [Edit Build Project] (ビルドプロジェクトの編集) ページの [Environment] (環境変数) で、[Additional configuration] (追加設定) を展開し、[Compute type] (コンピューティングタイプ) からいずれかの オプションを選択します。詳細については、「ビルドプロジェクトの作成 (コンソール)」または 「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。
- で AWS CLI、 environment オブジェクトcomputeTypeの値を指定して、 create-projectま たは update-project コマンドを実行します。詳細については、ビルドプロジェクトの作成 (AWS CLI)またはビルドプロジェクトの設定の変更 (AWS CLI)を参照してください。
- AWS SDKs、ターゲットプログラミング言語の CreateProjectまたは UpdateProjectオペ レーションに相当する を呼び出し、environmentオブジェクトcomputeTypeの値に相当する を 指定します。詳細については、「<u>AWS SDKsとツールのリファレンス</u>」を参照してください。

一部の環境タイプとリージョン可用性には制限があります。

- コンピューティングタイプ Linux GPU Small (LINUX_GPU_CONTAINER) は、次のリージョンのみで利用可能です。
 - 米国東部 (バージニア北部)
 - 米国西部 (オレゴン)
 - アジアパシフィック(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)
- コンピューティングタイプ Linux GPU Large (LINUX_GPU_CONTAINER) は、次のリージョンのみで利用可能です。
 - 米国東部(オハイオ)
 - 米国東部 (バージニア北部)
 - 米国西部 (オレゴン)
 - アジアパシフィック (ソウル)
 - 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)
 - 中国 (北京)
 - 中国 (寧夏)
 - 欧州 (フランクフルト)
 - ・ 欧州 (アイルランド)
 - 欧州 (ロンドン)
- コンピューティングタイプ「BUILD_GENERAL1_2XLARGE」は、次のリージョンのみで利用可能です。
 - 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 (ハイデラバード)
 - ・アジアパシフィック(香港)
 - アジアパシフィック (ジャカルタ)
 - アジアパシフィック (メルボルン)
 - 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)

- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- 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 (スペイン)
- 欧州 (ストックホルム)
- 欧州 (チューリッヒ)
- ・ イスラエル (テルアビブ)
- ・ 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- ・ 南米 (サンパウロ)
- ・環境タイプ「ARM_CONTAINER」は、次のリージョンのみで利用可能です。
 - 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(香港)
 - アジアパシフィック (ジャカルタ)
 - アジアパシフィック (ハイデラバード)
 - アジアパシフィック (ムンバイ)
 - ・アジアパシフィック(大阪)
 - アジアパシフィック (ソウル)
 - アジアパシフィック (シンガポール)
 - アジアパシフィック (シドニー)
 - アジアパシフィック(東京)

<u>・ カナダ (中部)</u>

- 中国 (寧夏)
- ・ 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- ・ イスラエル (テルアビブ)
- ・ 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- 南米(サンパウロ)
- 環境タイプ「WINDOWS_SERVER_2022_CONTAINER」は、次のリージョンのみで利用可能です。
 - 米国東部 (オハイオ)
 - 米国東部 (バージニア北部)
 - 米国西部 (オレゴン)
 - アジアパシフィック (シドニー)
 - ・アジアパシフィック(東京)
 - ・ 欧州 (フランクフルト)
 - 欧州 (アイルランド)
 - ・ 南米 (サンパウロ)

• 環境タイプ LINUX_EC2

(BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE)は、次の リージョンでのみ使用できます。

- 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)
- ・ アフリカ (ケープタウン)
- ・ アジアパシフィック (香港)

オシデマンジャーク(ジャカルタ)

- アジアパシフィック (メルボルン)
- 欧州 (チューリッヒ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ムンバイ)
- ・アジアパシフィック(大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック(東京)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- ・ 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- ・ イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- ・ 南米 (サンパウロ)
- AWS GovCloud (米国西部)
- AWS GovCloud (米国東部)
- ・ 環境タイプ ARM_EC2

(BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE)は、次の リージョンでのみ使用できます。

- 米国東部(オハイオ)
- オンデ米国東部 (パージニア北部)
 - 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(香港)
- アジアパシフィック (ジャカルタ)
- 欧州 (チューリッヒ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ムンバイ)
- ・アジアパシフィック(大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- ・アジアパシフィック(東京)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- ・ 欧州 (フランクフルト)
- ・ 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- ・ 欧州 (ストックホルム)
- ・ イスラエル (テルアビブ)
- ・中東 (バーレーン)
- ・ 南米 (サンパウロ)
- AWS GovCloud (米国西部)
- AWS GovCloud (米国東部)
- 環境タイプ WINDOWS_EC2 (BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE) は、次の リージョンでのみ使用できます。
 - 米国東部(オハイオ)
- オンデマンド環境タイプについて • 米国東部 (バージニア北部)

- 米国西部 (オレゴン)
- アジアパシフィック (シドニー)
- アジアパシフィック(東京)
- 欧州 (フランクフルト)
- ・ 欧州 (アイルランド)
- 南米(サンパウロ)
- ・コンピューティングモード AWS Lambda (ARM_LAMBDA_CONTAINER および LINUX_LAMBDA_CONTAINER) は、次のリージョンでのみ使用できます。
 - 米国東部 (バージニア北部)
 - 米国東部 (オハイオ)
 - 米国西部 (オレゴン)
 - アジアパシフィック (ムンバイ)
 - アジアパシフィック (シンガポール)
 - アジアパシフィック (シドニー)
 - アジアパシフィック(東京)
 - 欧州 (フランクフルト)
 - ・ 欧州 (アイルランド)
 - 南米 (サンパウロ)
- コンピューティングモード MAC_ARM は、次のリージョンのみで利用可能です。
 - 米国東部 (バージニア北部)
 - 米国東部 (オハイオ)
 - 米国西部 (オレゴン)
 - アジアパシフィック (シドニー)
 - ・ 欧州 (フランクフルト)

コンピューティングタイプ BUILD_GENERAL1_2XLARGE では、最大 100 GB までの圧縮されていな い Docker イメージがサポートされています。

Note

<u>カスタムビルド環境イメージとして、CodeBuild は、コンピューティングタイプを問</u> ^{オンデマンド環境タイプについて API バージョン 2016-10-06 331 わず、Linux および Windows で最大 50 GB の未圧縮の Docker イメージをサポートし} ます。ビルドイメージのサイズを確認するには、Docker を使用して docker images *REPOSITORY*: *TAG* コマンドを実行します。

Amazon EFS を使用してビルドコンテナのより多くの領域にアクセスできます。詳細については、 「<u>の Amazon Elastic File System サンプル AWS CodeBuild</u>」を参照してください。コンテナのディ スク領域をビルド中に操作する場合は、ビルドを特権モードで実行している必要があります。

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime Privilege and</u> <u>Linux Capabilities</u>」を参照して、特権モードを有効にします。また、Windows は特権モード をサポートしていません。

ビルド環境のシェルとコマンド

ビルドのライフサイクル中にビルド環境で を実行する AWS CodeBuild ための一連のコマンドを提供 します (ビルドの依存関係のインストール、ソースコードのテストとコンパイルなど)。これらのコ マンドを指定する方法はいくつかあります。

- ビルド仕様ファイルを作成し、それをソースコードに組み込みます。このファイルでは、ビルドラ イフサイクルの各段階で実行するコマンドを指定します。詳細については、「<u>CodeBuild のビルド</u> 仕様に関するリファレンス」を参照してください。
- CodeBuild コンソールを使用してビルドプロジェクトを作成します。[ビルドコマンドの挿入]の [ビルドコマンド] に、[build] フェーズで実行するコマンドを入力します。詳細については、「ビ ルドプロジェクトの作成 (コンソール)」を参照してください。
- CodeBuild コンソールを使用してビルドプロジェクトの設定を変更します。[ビルドコマンドの挿入]の[ビルドコマンド]に、[build] フェーズで実行するコマンドを入力します。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。
- AWS CLI AWS SDKs を使用してビルドプロジェクトを作成するか、ビルドプロジェクトの 設定を変更します。コマンドを使用して buildspec ファイルを含むソースコードを参照する か、buildspec ファイルと同等の内容を含む単一の文字列を指定します。詳細については、ビルド プロジェクトの作成 または ビルドプロジェクト設定を変更 を参照してください。

 AWS CLI AWS SDKs を使用してビルドを開始し、buildspec ファイルまたは同等の buildspec ファイルの内容を含む単一の文字列を指定します。詳細については、ビルドを手動で実行 にある buildspec0verride 値の説明を参照してください。

任意の Shell コマンド言語 (sh) のコマンドを指定できます。ビルド仕様バージョン 0.1 で は、CodeBuild は各ビルド環境の各インスタンスで各シェルコマンドを実行します。つまり、各コマ ンドは他のすべてのコマンドとは独立して実行されます。したがって、デフォルトでは、以前のコマ ンド (ディレクトリの変更や環境変数の設定など) の状態に依存する単一のコマンドを実行すること はできません。この制限を回避するには、バージョン 0.2 を使用することをお勧めします。これによ り、問題が解決されます。バージョン 0.1 を使用する場合は、以下のアプローチをお勧めします。

- デフォルトシェルの単一のインスタンスで実行するコマンドを含むシェルスクリプトをソース コードに含めます。たとえば、my-script.shという名前のファイルを、cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;などのコマンドを含むソースコードに含めます。次 に、buildspec ファイルで./my-script.sh コマンドを指定します。
- buildspec ファイル (または フェーズに限ってはコンソールの [Build commandbuild] 設定) で、 デフォルトシェルの単一のインスタンスで実行するすべてのコマンドが含まれている単一のコマン ドを指定します (例: cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd)。

CodeBuild でエラーが発生した場合は、デフォルトシェルの独自のインスタンスで単一のコマンドを 実行するのに比べて、トラブルシューティングが難しくなる場合があります。

Windows Server Core イメージで実行されるコマンドには、Powershell シェルが使用されます。

ビルド環境の環境変数

AWS CodeBuild には、ビルドコマンドで使用できる環境変数がいくつか用意されています。

AWS_DEFAULT_REGION

ビルドが実行されている AWS リージョン (例: us-east-1) 。この環境変数は、 AWS CLIで主 に使用されます。

AWS_REGION

ビルドが実行されている AWS リージョン (例: us-east-1) 。この環境変数は、主に AWS SDKs によって使用されます。

CODEBUILD_BATCH_BUILD_IDENTIFIER

バッチビルドでのビルドの識別子。これは、バッチの buildspec で指定されています。詳細については、「<u>the section called "バッチのビルド仕様 (buildspec)</u>に関するリファレンス"」を参照してください。

CODEBUILD_BUILD_ARN

ビルドの Amazon リソースネーム (ARN) (例:arn:aws:codebuild:*region-ID:account-ID*:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE)。 CODEBUILD BUILD ID

ビルドの CodeBuild ID (例: codebuild-demo-project:b1e6661ee4f2-4156-9ab9-82a19EXAMPLE)。

CODEBUILD_BUILD_IMAGE

CodeBuild のビルドイメージ識別子 (例: aws/codebuild/standard:2.0)。

CODEBUILD_BUILD_NUMBER

プロジェクトの現在のビルド番号。

CODEBUILD_BUILD_SUCCEEDING

現在のビルドが成功かどうか。ビルドが失敗の場合は 0 に設定され、成功の場合は 1 に設定され ます。

CODEBUILD_INITIATOR

ビルドを開始したエンティティ。CodePipeline でビルドが開始された場合は、パイプラインの 名前を表します (例: codepipeline/my-demo-pipeline)。ユーザーがビルドを開始した場合 は、ユーザーの名前を表します (例: MyUserName)。CodeBuild の Jenkins プラグインがビルドを 開始した場合、これは文字列「CodeBuild-Jenkins-Plugin」です。

CODEBUILD_KMS_KEY_ID

CodeBuild がビルド出力アーティファクトの暗号化に使用する AWS KMS キーの識別子 (例: arn:aws:kms:*region-ID*:*account-ID*:key/*key-ID*または alias/*key-alias*)。

CODEBUILD_PROJECT_ARN

プロジェクトの Amazon リソースネーム (ARN) (例: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name*)。

CODEBUILD_PUBLIC_BUILD_URL

パブリックビルドのウェブサイトにある、このビルドのビルド結果の URL。この変数は、ビル ドプロジェクトでパブリックビルドが有効になっている場合にのみ設定されます。詳細について は、「パブリックビルドプロジェクトの URL を取得」を参照してください。

CODEBUILD_RESOLVED_SOURCE_VERSION

ビルドのソースコードのバージョンの識別子。内容は、以下のようなソースコードリポジトリに よって異なります。

CodeCommit、GitHub、GitHub Enterprise Server、Bitbucket

この変数には、コミット ID が含まれます。

CodePipeline

この変数には、CodePipeline によって提供されるソースのリビジョンが含まれます。

ソースがバージョニングが有効になっていない Amazon S3 バケットである場合な

ど、CodePipeline がソースリビジョンを解決できない場合、この環境変数は設定されません。

Amazon S3

この変数は設定されていません。

該当する場合、CODEBUILD_RESOLVED_SOURCE_VERSION 変数は、フェーズ DOWNLOAD_SOURCE の後でのみ利用可能です。

CODEBUILD_SOURCE_REPO_URL

入力アーティファクトまたはソースコードリポジトリの URL。Amazon S3 では、これは s3:// の後にバケット名と入力アーティファクトへのパスが続きます。CodeCommit および GitHub の 場合、これはリポジトリのクローン URL です。CodePipeline から生成されたビルドの場合、こ の環境変数は空の場合があります。

セカンダリソースの場合、セカンダリソースリポジトリの URL の環境変 数は「CODEBUILD_SOURCE_REPO_URL_<*sourceIdentifier>*」です。 「<*sourceIdentifier>*」は、作成するソース識別子です。

CODEBUILD_SOURCE_VERSION

値の形式は、ソースコードリポジトリによって異なります。

- Amazon S3 では、入力アーティファクトに関連付けられたバージョン ID です。
- CodeCommit では、ビルドするソースコードのバージョンに関連付けられたコミット ID また はブランチ名です。
- GitHub、GitHub Enterprise Server、Bitbucket の場合、ビルドするソースコードのバージョン に関連付けられたコミット ID、ブランチ名、またはタグ名です。

Note

Webhook プルリクエストイベントによりトリガーされた GitHub または GitHub Enterprise Server ビルドの場合、pr/*pull-request-number* です。

セカンダリソースの場合、セカンダリソースバージョンの環境変数は

「CODEBUILD_SOURCE_VERSION_<*sourceIdentifier*>」です。

「*<sourceIdentifier>*」は、作成するソース識別子です。詳細については、「<u>複数の入力</u> ソースと出力アーティファクトのサンプル」を参照してください。

CODEBUILD_SRC_DIR

CodeBuild がビルドに使用するディレクトリパス (例: /tmp/src123456789/src)。

セカンダリソースの場合、ディレクトリパスの環境変数は

「CODEBUILD_SRC_DIR_*<sourceIdentifier>*」です。「*<sourceIdentifier>*」は作成 するソース識別子です。詳細については、「<u>複数の入力ソースと出力アーティファクトのサンプ</u> ル」を参照してください。

CODEBUILD_START_TIME

Unix タイムスタンプとして指定されたビルドの開始時間 (ミリ秒単位)。

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

Webhook イベントをトリガーしたユーザーのアカウント ID。

CODEBUILD_WEBHOOK_BASE_REF

現在のビルドをトリガーする Webhook イベントの基本参照名。プルリクエストでは、ブランチ 参照を表します。

CODEBUILD_WEBHOOK_EVENT

現在のビルドをトリガーした Webhook イベント。

CODEBUILD_WEBHOOK_MERGE_COMMIT

ビルドに使用されるマージコミットの識別子。この変数は、Bitbucket プルリクエストがスカッシュ戦略とマージされ、プルリクエストブランチが閉じられたときに設定されます。この場合、 元のプルリクエストコミットは存在しなくなるため、この環境変数には圧縮されたマージコミットの識別子が含まれます。

CODEBUILD_WEBHOOK_PREV_COMMIT

現在のビルドをトリガーする Webhook プッシュイベントの前の最新のコミットの ID。

CODEBUILD_WEBHOOK_HEAD_REF

現在のビルドをトリガーする Webhook イベントのヘッド参照名。ブランチ参照またはタグ参照 を表します。

CODEBUILD_WEBHOOK_TRIGGER

ビルドをトリガーした Webhook イベントを表示します。この変数は、Webhook によってト リガーされるビルドにのみ使用できます。値は、GitHub、GitHub Enterprise Server、または Bitbucket から CodeBuild に送信されたペイロードから解析されます。値の形式は、ビルドをト リガーしたイベントのタイプによって異なります。

- ・プルリクエストによってトリガーされたビルドの場合、pr/pull-request-numberです。
- 新しいブランチを作成するか、ブランチにコミットをプッシュすることでトリガーされたビルドの場合、branch/branch-nameです。

タグをリポジトリにプッシュすることでトリガーされたビルドの場合、tag/tag-nameです。
 HOME

この環境変数は常に「/root」に設定されます。

AWS CodeBuild は、セルフホスト型ランナービルドの一連の環境変数もサポートしていま す。CodeBuild セルフホスト型ランナーの詳細については、「<u>チュートリアル: CodeBuild がホスト</u> する GitHub Actions ランナーを設定」を参照してください。

CODEBUILD_RUNNER_OWNER

セルフホスト型ランナーのビルドをトリガーするリポジトリの所有者です。 CODEBUILD_RUNNER_REPO

セルフホスト型ランナーのビルドをトリガーするリポジトリの名前です。

CODEBUILD_RUNNER_REPO_DOMAIN

セルフホスト型ランナーのビルドをトリガーするリポジトリのドメインです。指定された GitHub Enterprise ビルドのみです。

CODEBUILD_WEBHOOK_LABEL

ビルド中のビルドの上書きとセルフホスト型ランナーの設定に使用されるラベルです。

CODEBUILD_WEBHOOK_RUN_ID

ビルドに関連付けられたワークフローの実行 ID です。

CODEBUILD_WEBHOOK_JOB_ID

ビルドに関連付けられたジョブのジョブ ID です。

CODEBUILD_WEBHOOK_WORKFLOW_NAME

ウェブフックのリクエストペイロードに存在する場合、ビルドに関連付けられたワークフローの 名前です。

CODEBUILD_RUNNER_WITH_BUILDSPEC

buildspec の上書きがセルフホスト型ランナーリクエストラベルで設定されている場合、これは true に設定されます。

独自の環境変数を持つビルド環境を提供することもできます。詳細については、以下のトピックを参 照してください。

- CodePipeline で CodeBuild を使用
- ビルドプロジェクトの作成
- ・ ビルドプロジェクト設定を変更
- ビルドを手動で実行
- ・ビルド仕様 (buildspec) に関するリファレンス

ビルド環境で使用できる環境変数を一覧表示するには、構築時に printenv コマンド (Linux ベース のビルド環境) または "Get-ChildItem Env:" (Windows ベースのビルド環境) を実行できます。 前述のものを除いて、「CODEBUILD_」で始まる環境変数は、CodeBuild の内部使用のためのもので す。それらはビルドコマンドで使用できません。

▲ Important

環境変数を使用して、特に AWS アクセスキー IDs。環境変数は、CodeBuild コンソールや AWS CLIなどのツールを使用してプレーンテキストで表示できます。 機密値は Amazon EC2 Systems Manager パラメータストアに保存後、ビルド仕様から取 得することをお勧めします。重要な値を保存するには、Amazon EC2 Systems Manager ユーザーガイドの「<u>Systems Manager パラメータストア</u>」および「<u>チュートリアル: String</u> <u>パラメータの作成とテスト (コンソール)</u>」を参照してください。これらを取得するには、 「parameter-store」の buildspec の構文 マッピングを参照してください。

ビルド環境のバックグラウンドタスク

ビルド環境でバックグラウンドタスクを実行できます。これを行うには、ビルドプロセスでシェルが 終了される場合でも、buildspec で nohup コマンドを使用してバックグラウンドのタスクとしてコ マンドを実行します。実行中のバックグラウンドタスクを強制終了するには、disown コマンドを使 用します。

例:

バックグラウンドプロセスを開始し、その後、完了するまで待機します。

```
nohup sleep 30 & echo $! > pidfile
...
wait $(cat pidfile)
```

• バックグラウンドプロセスを開始し、その後、完了するまで待機しません。

nohup sleep 30 & disown \$!

バックグラウンドプロセスを開始し、その後、強制終了します。

```
I
nohup sleep 30 & echo $! > pidfile
...
kill $(cat pidfile)
```

ビルドプロジェクト

ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコー ドの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。

ビルドプロジェクトを操作して以下のタスクを実行できます。

トピック

- でビルドプロジェクトを作成する AWS CodeBuild
- 通知ルールの作成
- でビルドプロジェクト設定を変更する AWS CodeBuild
- CodeBuild の複数のアクセストークン
- でビルドプロジェクトを削除する AWS CodeBuild
- パブリックビルドプロジェクトの URL を取得
- ビルドプロジェクトを共有
- ビルドプロジェクトをタグ付け
- でランナーを使用する AWS CodeBuild
- でウェブフックを使用する AWS CodeBuild
- でビルドプロジェクトの詳細を表示する AWS CodeBuild
- ・ <u>でビルドプロジェクト名を表示する AWS CodeBuild</u>

でビルドプロジェクトを作成する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、または AWS SDKsを使用してビルドプロジェクトを作成できます。

トピック

- 前提条件
- ビルドプロジェクトの作成 (コンソール)
- ・ <u>ビルドプロジェクトの作成 (AWS CLI)</u>
- ・ <u>ビルドプロジェクトを作成する (AWS SDKs)</u>
- ・ ビルドプロジェクトの作成 (AWS CloudFormation)

前提条件

ビルドプロジェクトを作成する前に、ビルドを計画するの質問に回答します。

ビルドプロジェクトの作成 (コンソール)

AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>://https:// https://https://https://https

CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ以外 の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。

[Create build project (ビルドプロジェクトの作成)] を選択します。

次のセクションに入力します。完了したら、ページの下部にある [Create build project] (ビルドプロ ジェクトを作成する) を選択します。

セクション:

- プロジェクトの設定
- ・ソース
- 環境
- Buildspec
- Batch 構成
- アーティファクト
- ログ

プロジェクトの設定

[Project name] (プロジェクト名)

このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごと に一意である必要があります。

説明

また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの 説明を任意で指定することもできます。

ビルドバッジ

(オプション)[Enable build badge] (ビルドバッジを有効にする) を選択すると、プロジェクトのビ ルドステータスが表示可能および埋め込み可能になります。詳細については、「<u>ビルドバッジサ</u> ンプル」を参照してください。

Note

ソースプロバイダーが Amazon S3 の場合、ビルドバッジは適用されません。

同時ビルド制限を有効にする

(オプション) このプロジェクトで同時ビルド数を制限するには、次の手順を実行します。

- [Restrict number of concurrent builds this project can start] (このジョブで許可される同時実行の最大数を設定)を選択します。
- [Concurrent build limit] (同時ビルド制限) で、このジョブで許可される同時実行の最大数を設定します。この制限は、アカウントに設定された同時ビルド制限より大きくすることはできません。アカウント制限を超える数値を入力しようとすると、エラーメッセージが表示されます。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウ ントがこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

追加情報

(オプション)タグには、サポート AWS サービスで使用するタグの名前と値を入力します。[Add row] を使用して、タグを追加します。最大 50 個のタグを追加できます。

ソース

ソースプロバイダー

ソースコードプロバイダーのタイプを選択します。次のリストを使用して、ソースプロバイダー に関する適切な選択を行います。

Note

CodeBuild は Bitbucket サーバーをサポートしていません。

Amazon S3

バケット

ソースコードが格納されている入力バケットの名前を選択します。

S3 オブジェクトキーまたは S3 フォルダ

ZIP ファイルの名前、またはソースコードを含むフォルダへのパスを入力します。S3 バケットの中身をすべてダウンロードするには、スラッシュ記号 (/) を入力します。

ソースバージョン

入力ファイルのビルドを表すオブジェクトのバージョン IDを入力。詳細については、「<u>を使</u> 用したソースバージョンサンプル AWS CodeBuild」を参照してください。

CodeCommit

リポジトリ

使用するリポジトリを選択します。

参照タイプ

[Branch] (ブランチ) または [Git tag] (Git タグ) を選択するか、[Commit ID] (コミット ID) を入 カして、ソースコードのバージョンを指定します。詳細については、「<u>を使用したソースバー</u> ジョンサンプル AWS CodeBuild」を参照してください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。 Git クローンの深度

選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全ク ローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

Bitbucket

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections]、[OAuth]、[アプリパスワード]、または [個人用アクセストークン] を選択 して CodeBuild に接続します。

Connection

Bitbucket 接続または Secrets Manager シークレットを選択して、指定した接続タイプ経由で 接続します。

リポジトリ

[Bitbucket アカウントのリポジトリ] または [パブリックリポジトリ] を選択し、リポジトリ URL を入力します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細につい ては、「を使用したソースバージョンサンプル AWS CodeBuild」を参照してください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。 Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、Bitbucket コミットステータスの name パラ メータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「<u>ビル</u> ド」を参照してください。

[Target URL] (ターゲットURL) に、Bitbucket コミットステータスの url パラメータに使用す る値を記入します。詳細については、Bitbucket API ドキュメントの「<u>ビルド</u>」を参照してく ださい。

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポート されます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告さ れた API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。

[Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築します。Webhook およびフィルターグループの詳細については、「<u>Bitbucket ウェブフックイベン</u>ト」を参照してください。
GitHub

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[GitHub アプリ]、[OAuth]、または [個人用アクセストークン] を選択して CodeBuild に接続します。

Connection

GitHub 接続または Secrets Manager シークレットを選択して、指定した接続タイプ経由で接 続します。

リポジトリ

[GitHub アカウントのリポジトリ]、[パブリックリポジトリ]、または [GitHub スコープ付き ウェブフック] を選択し、リポジトリ URL を入力します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細につい ては、「を使用したソースバージョンサンプル AWS CodeBuild」を参照してください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「<u>ソースプロバイダーのアクセス</u>」を参照してください。

[Status context] (ステータスコンテキスト) に、GitHub コミットステータスの contextパラ メータに使用する値を記入します。 q 詳細については、GitHub デベロッパーガイドの「<u>コ</u> ミットステータスの作成」を参照してください。

[Target URL] (ターゲット URL) に、 GitHub コミットステータスの target_url パラメー タに使用する値を記入します。詳細については、GitHub デベロッパーガイドの「<u>コミットス</u> テータスの作成」を参照してください。

webhook によってトリガーされたビルドのステータスは、常にソースプロバイダーにレポー トされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告 された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。

[Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび 再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築 します。Webhook およびフィルターグループの詳細については、「<u>GitHub ウェブフックイベン</u> ト」を参照してください。

GitHub Enterprise Server

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] または [個人用アクセストークン] を選択して CodeBuild に接続します。

Connection

GitHub Enterprise 接続または Secrets Manager シークレットを選択して、指定した接続タイ プ経由で接続します。

リポジトリ

[自分の GitHub Enterprise アカウントのレポジトリ] または [GitHub Enterprise スコープ付き ウェブフック] を選択し、リポジトリ URL を入力します。

ソースバージョン

プルリクエスト、ブランチ、コミット ID、コミット ID、参照、およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。 [Status context] (ステータスコンテキスト) に、GitHub コミットステータスの contextパラ メータに使用する値を記入します。 q 詳細については、GitHub デベロッパーガイドの「<u>コ</u> ミットステータスの作成」を参照してください。

[Target URL] (ターゲット URL) に、 GitHub コミットステータスの target_url パラメー タに使用する値を記入します。詳細については、GitHub デベロッパーガイドの「<u>コミットス</u> テータスの作成」を参照してください。

webhook によってトリガーされたビルドのステータスは、常にソースプロバイダーにレポー トされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告 された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。

安全でない SSL

[Enable insecure SSL (セキュアでない SSL を有効にする)] を選択して、GitHub Enterprise プロジェクトリポジトリに接続するときの SSL 警告を無視します。

[Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび 再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築 します。Webhook およびフィルターグループの詳細については、「<u>GitHub ウェブフックイベン</u> <u>ト</u>」を参照してください。

GitLab

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] は、GitLab を CodeBuild に接続するために使用されます。

Connection

CodeConnections 経由で接続する GitLab 接続を選択します。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、または参照およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

GitLab Self Managed

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] は、GitLab セルフマネージドを CodeBuild に接続するために使用されます。

Connection

CodeConnections 経由で接続する GitLab セルフマネージド接続を選択します。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、または参照およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「<u>ソースプロバイダーのアクセス</u>」を参照してください。

環境

[プロビジョニングモデル]

次のいずれかを行います:

- が管理するオンデマンドフリートを使用するには AWS CodeBuild、オンデマンドを選択します。オンデマンドフリートでは、CodeBuild がビルドのコンピューティングを行います。マシンはビルドが終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応できる自動スケーリング機能を備えています。
- ・が管理するリザーブドキャパシティフリートを使用するには AWS CodeBuild、リザーブドキャ パシティを選択し、フリート名を選択します。リザーブドキャパシティフリートでは、ビルド 環境に合わせて専有インスタンスのセットを設定します。これらのマシンはアイドル状態のま まで、ビルドやテストをすぐに処理できる状態になり、ビルド時間を短縮します。リザーブド キャパシティフリートでは、マシンは常に稼働しており、プロビジョニングされている間はコ ストが発生し続けます。

詳細については、<u>リザーブドキャパシティキャパシティフリートでビルドを実行</u>を参照してくだ さい。

環境イメージ

次のいずれかを行います:

- が管理する Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム、ランタイム (複数可)、イメージ、イメージバージョンから 選択します。利用可能な場合は、[環境タイプ]から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)]を選択した場合は、[External registry URL (外部のレジスト リ URL)] に docker repository/docker image name の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択 します。
- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択しま す。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を 選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を 選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報 は、Secrets Manager で作成する必要があります。詳細については、 AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager とは」を参照してください。

Note

CodeBuild はカスタムDocker イメージの「ENTRYPOINT」をオーバーライドします。

コンピューティング

次のいずれかを行います:

- EC2 コンピューティングを使用するには、[EC2] を選択します。EC2 コンピューティングは、 アクションの実行中に最適化された柔軟性を提供します。
- Lambda コンピューティングを使用するには、[Lambda] を選択します。Lambda コンピュー ティングは、ビルドの起動速度を最適化します。Lambda は、起動レイテンシーが低いため、 より高速なビルドをサポートします。また、Lambda は自動的にスケールされるため、ビルド はキュー内で実行を待機することはありません。詳細については、<u>AWS Lambda コンピュー</u> ティングでビルドを実行する を参照してください。

サービスロール

次のいずれかを行ってください。

- CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name]
 に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選 択します。[Role ARN] で、サービスロールを選択します。
 - Note

コンソールでは、ビルドプロジェクトの作成時に CodeBuild サービスロールも作成でき ます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソー ルでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルド プロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個の ビルドプロジェクトで使用できます。

追加設定

[自動再試行の制限]

ビルドが失敗した後の追加の自動再試行回数を指定します。例えば、自動再試行の制限が2 に設定されている場合、CodeBuild は RetryBuild API を呼び出して、さらに最大2回まで ビルドを自動的に再試行します。

タイムアウト

5 分~36 時間の間の値を指定します。この時間が経過してもビルドが完了していない場 合、CodeBuild はビルドを停止します。[hours] と [minutes] を空白のままにすると、デフォル ト値の 60 分が使用されます。

特権付与

(オプション) このビルドプロジェクトを使って Dockerイメージをビルドする場合にの み、[Docker イメージをビルドする場合、またはビルドで昇格された権限を取得する場合 は、このフラグを有効にする] を選択します。それ以外の場合、関連付けられているビルド で Docker デーモンと通信しようとすると、すべて失敗します。ビルドが Docker デーモンと 連係動作できるように、Docker デーモンも起動する必要があります。これを行う 1 つの方法 は、次のビルドコマンドを実行してビルドスペックの install フェーズで Docker デーモン を初期化することです。Docker をサポートする CodeBuild によって提供されるビルド環境イ メージを選択した場合は、これらのコマンドを実行しないでください。

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビ ルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime</u> <u>Privilege and Linux Capabilities</u>」を参照して、特権モードを有効にします。ま た、Windows は特権モードをサポートしていません。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

CodeBuild を VPC と連携させたい場合

- [VPC] で、CodeBuild が使用する VPC ID を選択します。
- ・ [VPC Subnets (サブネット)] で、CodeBuild が使用するリソースを含むサブネットを選択します。
- [VPC Security groups (VPC セキュリティグループ)] で、CodeBuild が VPC 内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「<u>Amazon Virtual Private Cloud AWS CodeBuild で を使用する</u>」を参照し てください。

コンピューティング

使用可能なオプションの1つを選択します。

レジストリ認証情報

プロジェクトが非プライベートレジストリイメージで設定されている場合は、レジストリ認証 情報を指定します。

Note

この認証情報は、イメージがプライベートレジストリのイメージで上書きされている 場合にのみ使用されます。

環境変数

[環境変数] で、名前と値を入力してから、ビルドによって使用される各環境変数の種類を選択 します。

Note

CodeBuild は、 AWS リージョンの環境変数を自動的に設定します。以下の環境変数を buildspec.yml に追加していない場合は、それらの変数を設定する必要があります。

- AWS_ACCOUNT_ID
- IMAGE_REPO_NAME
- IMAGE_TAG

コンソールと AWS CLI ユーザーは環境変数を表示できます。環境変数の表示に懸念がない場合は、[Name] および [Value] フィールドを設定し、[Type] を [Plaintext] に設定します。

アクセスキー ID、 AWS シークレット AWS アクセスキー、パスワードなどの機密性の高い値 を持つ環境変数をパラメータとして Amazon EC2 Systems Manager パラメータストアまたは に保存することをお勧めします AWS Secrets Manager。

Amazon EC2 Systems Manager パラメータストアを使用する場合は、[Type (タイプ)] で、 [Parameter (パラメータ)] を選択します。[Name] (名前) に、参照する CodeBuild の識別子を 入力します。[Value] (値) に、Amazon EC2 Systems Manager パラメータストアに保存され ているパラメータの名前を入力します。たとえば、/CodeBuild/dockerLoginPassword という名前のパラメータを使用して、[タイプ] で [Parameter (パラメータ)] を選択しま す。[Name (名前)] に LOGIN_PASSWORD と入力します。[Value (値)] に「/CodeBuild/ dockerLoginPassword」と入力します。

▲ Important

Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは / CodeBuild/ で始まるパラメータ名 (例: /CodeBuild/dockerLoginPassword) で保存することをお勧めします。CodeBuild コンソールを使用して、Amazon EC2 Systems Manager にパラメータを作成することができます。[パラメータの作成] を 選択し、ダイアログボックスの手順に従います。 (このダイアログボックスの KMS キーでは、アカウントで AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager はこのキーを使用して、ストレージ中にパラメータの値を暗号化 し、取得中に復号します。) CodeBuild コンソールを使用してパラメータを作成した 場合、コンソールは保存されている /CodeBuild/ パラメータ名を開始します。詳細 については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュー トリアル」を参照してください。

ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存 されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に [New service role] (新しいサービスロール)を選択した場合は、CodeBuild のビルドプロジェクト のデフォルトのサービスロールにこのアクションが含まれています。ただし [既存の サービスロール] を選択した場合は、このアクションをサービスロールに個別に含め る必要があります。

ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、[新 しいサービスロール] を選択した場合、/CodeBuild/ で始まらないパラメータ名に アクセスできるようにサービスロールを更新する必要があります。これは、サービス ロールで、/CodeBuild/ で始まるパラメータ名にのみアクセスが許可されるためで す。

[新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータ を復号するアクセス権限が含まれます。

既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれ ていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場 合、my_value が other_value に置き換えられます。同様に、Docker イメージ に /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数 が既に含まれていて、\$PATH:/usr/share/ant/bin の値を持つ PATH という名前 の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 \$PATH:/usr/share/ant/bin に置き換えられます。

CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックス は内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定 されます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ビルドプロジェクト定義の値が次に優先されます。
- ・ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合は、[Type] (タイプ) で、[Secrets Manager] を選択しま す。[Name] (名前) に、参照する CodeBuild の識別子を入力します。[Value (値)] に、パターン reference-key を使用して *secret-id:json-key:version-stage:version-id* を入 力します。詳細については、<u>Secrets Manager reference-key in the buildspec file</u> を参照して ください。

▲ Important

Secrets Manager を使用する場合は、「/CodeBuild/」で始まる名前で シークレットを保存することをお勧めします(たとえば、/CodeBuild/ dockerLoginPassword)。詳細については、AWS Secrets Managerユーザーガイ ドの「<u>AWS Secrets Manager とは</u>」を参照してください。 ビルドプロジェクトが Secrets Manager パラメータストアに保存されて いるパラメータを参照する場合、ビルドプロジェクトのサービスロールで secretsmanager:GetSecretValue アクションを許可する必要があります。以前 に [New service role] (新しいサービスロール)を選択した場合は、CodeBuild のビルド プロジェクトのデフォルトのサービスロールにこのアクションが含まれています。た だし [既存のサービスロール]を選択した場合は、このアクションをサービスロールに 個別に含める必要があります。

Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した 場合、/CodeBuild/ で始まらないシークレット名にアクセスできるようにサービス ロールを更新する必要があります。これは、サービスロールで、/CodeBuild/ で始 まるシークレット名にのみアクセスが許可されるためです。 [新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets Manager の /CodeBuild/ 名前空間ですべてのシークレットを復号するアクセス許可 が含まれます。

Buildspec

ビルド仕様

次のいずれかを行ってください。

- ソースコードにビルド仕様ファイルが含まれている場合は、[Use a buildspec file (buildspec ファイルを使用)] を選択します。デフォルトでは、CodeBuild はソースコードのルートディレクトリで buildspec.yml という名前のファイルを探します。buildspec ファイルに別の名前または場所が使用されている場合は、Buildspec 名 にソースルートからのパスを入力します(例えば、buildspec-two.yml または configuration/buildspec.yml)。buildspec ファイルが S3 バケットにある場合は、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します(例: arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml)。
- ソースコードにビルド仕様ファイルが含まれていない場合、または、ソースコードのルート ディレクトリで build ファイルの buildspec.yml フェーズに指定されているものと異なる ビルドコマンドを実行する場合は、[ビルドコマンドの挿入] を選択します。[ビルドコマンド] に、build フェーズで実行するコマンドを入力します。複数のコマンドについては、&& で各 コマンドを区切ります (例: mvn test && mvn package)。他のフェーズでコマンドを実行 する場合、または build フェーズのコマンドの長いリストがある場合は、ソースコマンドの ルートディレクトリに buildspec.yml ファイルを追加し、ファイルにコマンドを追加してか ら、[Use the buildspec.yml in the source code root directory] (ソースコードのルートディレクト リの「buildspec.yml」を使用)を選択します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

Batch 構成

ビルドのグループを1つの操作として実行できます。詳細については、「<u>ビルドをバッチで実行</u>」 を参照してください。 バッチ構成の定義

このプロジェクトでバッチビルドを許可する場合に選択します。 Batch サービスロール

バッチビルドのサービスロールを提供します。

次のいずれかを選択します。

- バッチサービスロールがない場合は、[New service role] (新しいサービスロール) を選択します。[Service role] (サービスロール) に、新しいロールの名前を入力します。
- バッチサービスロールがある場合は、[Existing service role] (既存のサービスロール) を選択します。[Service role] (サービスロール) で、サービスロールを選択します。

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロール では、CodeBuild が StartBuild、StopBuild および RetryBuild アクションを使用して、 バッチの一部としてビルドを実行する上で必要です。次の2つの理由により、お客様はビルドで 使用するものと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用 すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドと計算タイプの数を制限 する制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を 回避する可能性があります。

バッチで許可されるコンピューティングタイプ

バッチに使用できる計算タイプを選択します。該当するものをすべて選択します。 バッチに許可されるフリート

バッチに許可されているフリートを選択します。該当するものをすべて選択します。 バッチで許可される最大ビルド

バッチで許可されるビルドの最大数を入力します。バッチがこの制限を超えると、バッチは失敗 します。

バッチのタイムアウト

バッチビルドが完了する最大時間を入力します。

アーティファクトの結合

[Combine all artifacts from batch into a single location] (バッチのすべてのアーチファクト) を1つ の場所に結合するを選択して、バッチのすべてのアーチファクトを単一の場所に結合します。

バッチレポートモード

バッチビルドに対して望ましいビルドステータスレポートモードを選択します。

Note

このフィールドが利用可能になるのは、プロジェクトソースが Bitbucket、GitHub、ま たは GitHub Enterprise であり、[Source] (ソース) で [Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダー にビルドステータスをレポートする) が選択されている場合のみです。

集約されたビルド

これを選択して、バッチ内にあるすべてのビルドのステータスを単一のステータスレポートに まとめます。

個々のビルド

これを選択して、バッチ内にあるすべてのビルドのビルドステータスが個別に報告されるよう にします。

アーティファクト

Type

次のいずれかを行ってください。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts] を選択します。ビルドテストのみを実行している場合や、Docker イメージを Amazon ECR リポジトリにプッシュする場合には、これを行うことができます。
- ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を 行います。
 - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)]
 を空白のままにします。それ以外の場合は、名前を入力します。(ZIP ファイルを出力して
 ZIP ファイルにファイル拡張子を付ける場合は、必ず ZIP ファイル名の後に含めます)。
 - buildspec ファイルで指定した名前で、コンソールで指定した名前を上書きする場合は、 [Enable semantic versioning (セマンティックバージョニングを有効にする)] を選択しま す。buildspec ファイル内の名前は、ビルド時に計算され、Shell コマンド言語を使用しま す。たとえば、アーティファクト名に日付と時刻を追加して常に一意にできます。アーティ

ファクト名を一意にすると、アーティファクトが上書きされるのを防ぐことができます。詳 細については、「buildspec の構文」を参照してください。

- [Bucket name (バケット名)] で、出力バケットの名前を選択します。
- この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド 出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の 場所の場合は、各場所をコンマで区切ります (例: appspec.yml, target/my-app.jar)。 詳細については、「files」で buildspec の構文 の説明を参照してください。
- ビルドアーティファクトを暗号化しない場合は、[アーティファクト暗号化の削除]を選択します。

アーティファクトのセカンダリセットごとに:

- 1. [Artifact 識別子] には、英数字とアンダースコアのみを使用して 128 文字未満の値を入力しま す。
- 2. [アーティファクトの追加]を選択します。
- 3. セカンダリアーティファクトを設定するには、前のステップに従います。
- 4. [アーティファクトの保存]を選択します。

追加設定

暗号化キー

次のいずれかを実行します。

- アカウントの Amazon S3 の AWS マネージドキー を使用してビルド出力アーティファクト を暗号化するには、[暗号化キー] を空白のままにします。これがデフォルトです。
- カスタマー管理のキーを使用してビルド出力アーティファクトを暗号化するには、[暗号化キー] に KMS キーの ARN を入力します。arn:aws:kms:*region-ID*:account-ID:key/key-ID の形式を使用します。

キャッシュタイプ

[キャッシュタイプ] で、以下のいずれかを選択します。

- キャッシュを使用しない場合は、[No cache]を選択します。
- Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
 - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
 - ・ (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パス のプレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名

に似ています。これにより、バケット内の同じディレクトリにキャッシュを保存できま す。

▲ Important パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

 ローカルキャッシュを使用する場合は、[ローカル]を選択し、ローカルキャッシュモードを 1つ以上選択します。

Note

Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択 する場合、プロジェクトは権限モードで実行する必要があります。

キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使 用されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定 に関する詳細については、「<u>buildspec の構文</u>」を参照してください。キャッシングの詳細に ついては、「パフォーマンスを向上させるためのキャッシュビルド」を参照してください。

ログ

作成するログを選択します。Amazon CloudWatch Logs、Amazon S3 ログ、または両方のログを作 成できます。

CloudWatch

Amazon CloudWatch Logs が必要な場合:

CloudWatch Logs

[CloudWatch logs] を選択します。

Amazon CloudWatch Logs のログのグループ名を入力します。

ストリーム名

Amazon CloudWatch Logs ログストリーム名を入力します。

S3

Amazon S3 ログが必要な場合は、以下のようになります。

S3 ログ

[S3 logs (S3 ログ)] を選択します。

バケット

ログを保存する S3 バケットの名前を選択します。 パスプレフィックス

ログのプレフィックスを入力します。

S3 ログの暗号化を無効にする

S3 ログを暗号化しない場合は、選択します。

ビルドプロジェクトの作成 (AWS CLI)

CodeBuild AWS CLI で を使用する方法の詳細については、「」を参照してください<u>コマンドライン</u> リファレンス。

を使用して CodeBuild ビルドプロジェクトを作成するには AWS CLI、JSON 形式の<u>プロジェクト</u>構 造を作成し、構造を入力し、 create-project コマンドを呼び出してプロジェクトを作成します。

JSON ファイルの作成

--generate-cli-skeleton オプションを使用して、<u>create-project</u> コマンドでスケルトン JSON ファイルを作成します。

aws codebuild create-project --generate-cli-skeleton > <json-file>

これにより、<json-file> で指定されるパスとファイル名で JSON ファイルが作成されます。

JSON ファイルを入力します。

JSON データを次のように変更して、結果を保存します。

```
{
    "name": "<project-name>",
    "description": "<description>",
    "source": {
}
```

```
"type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" | "GITLAB" |
"GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
   "location": "<source-location>",
   "gitCloneDepth": "<git-clone-depth>",
   "buildspec": "<buildspec>",
   "InsecureSsl": "<insecure-ssl>",
   "reportBuildStatus": "<report-build-status>",
   "buildStatusConfig": {
     "context": "<context>",
     "targetUrl": "<target-url>"
   },
   "gitSubmodulesConfig": {
     "fetchSubmodules": "<fetch-submodules>"
   },
   "auth": {
     "type": "<auth-type>",
     "resource": "<auth-resource>"
  },
   "sourceIdentifier": "<source-identifier>"
},
 "secondarySources": [
   {
       "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" |
"GITLAB" | "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
       "location": "<source-location>",
       "gitCloneDepth": "<git-clone-depth>",
       "buildspec": "<buildspec>",
       "InsecureSsl": "<insecure-ssl>",
       "reportBuildStatus": "<report-build-status>",
       "auth": {
         "type": "<auth-type>",
         "resource": "<auth-resource>"
       },
       "sourceIdentifier": "<source-identifier>"
  }
],
 "secondarySourceVersions": [
  {
     "sourceIdentifier": "<secondary-source-identifier>",
     "sourceVersion": "<secondary-source-version>"
  }
],
 "sourceVersion": "<source-version>",
 "artifacts": {
```

```
"type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
   "location": "<artifacts-location>",
   "path": "<artifacts-path>",
   "namespaceType": "<artifacts-namespacetype>",
   "name": "<artifacts-name>",
   "overrideArtifactName": "<override-artifact-name>",
   "packaging": "<artifacts-packaging>"
},
 "secondaryArtifacts": [
  {
     "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
     "location": "<secondary-artifact-location>",
     "path": "<secondary-artifact-path>",
     "namespaceType": "<secondary-artifact-namespaceType>",
     "name": "<secondary-artifact-name>",
     "packaging": "<secondary-artifact-packaging>",
     "artifactIdentifier": "<secondary-artifact-identifier>"
  }
],
 "cache": {
  "type": "<cache-type>",
   "location": "<cache-location>",
   "mode": [
     "<cache-mode>"
  1
},
 "environment": {
   "type": "LINUX_CONTAINER" | "LINUX_GPU_CONTAINER" | "ARM_CONTAINER" |
"WINDOWS_SERVER_2019_CONTAINER" | "WINDOWS_SERVER_2022_CONTAINER",
   "image": "<image>",
   "computeType": "BUILD_GENERAL1_SMALL" | "BUILD_GENERAL1_MEDIUM" |
"BUILD_GENERAL1_LARGE" | "BUILD_GENERAL1_2XLARGE",
   "certificate": "<certificate>",
   "environmentVariables": [
     {
       "name": "<environmentVariable-name>",
       "value": "<environmentVariable-value>",
       "type": "<environmentVariable-type>"
     }
   ],
   "registryCredential": [
     {
       "credential": "<credential-arn-or-name>",
       "credentialProvider": "<credential-provider>"
```

```
}
  ],
  "imagePullCredentialsType": "CODEBUILD" | "SERVICE_ROLE",
  "privilegedMode": "<privileged-mode>"
},
"serviceRole": "<service-role>",
"autoRetryLimit": <auto-retry-limit>,
"timeoutInMinutes": <timeout>,
"queuedTimeoutInMinutes": <queued-timeout>,
"encryptionKey": "<encryption-key>",
"tags": [
  {
    "key": "<tag-key>",
    "value": "<tag-value>"
  }
],
"vpcConfig": {
 "securityGroupIds": [
       "<security-group-id>"
  ],
  "subnets": [
      "<subnet-id>"
  ],
  "vpcId": "<vpc-id>"
},
"badgeEnabled": "<badge-enabled>",
"logsConfig": {
  "cloudWatchLogs": {
    "status": "<cloudwatch-logs-status>",
    "groupName": "<group-name>",
    "streamName": "<stream-name>"
  },
  "s3Logs": {
    "status": "<s3-logs-status>",
    "location": "<s3-logs-location>",
    "encryptionDisabled": "<s3-logs-encryption-disabled>"
  }
},
"fileSystemLocations": [
  {
    "type": "EFS",
    "location": "<EFS-DNS-name-1>:/<directory-path>",
    "mountPoint": "<mount-point>",
    "identifier": "<efs-identifier>",
```

```
"mountOptions": "<efs-mount-options>"
    }
  ],
  "buildBatchConfig": {
    "serviceRole": "<batch-service-role>",
    "combineArtifacts": <combine-artifacts>,
    "restrictions": {
      "maximumBuildsAllowed": <max-builds>,
      "computeTypesAllowed": [
        "<compute-type>"
      ],
      "fleetsAllowed": [
        "<fleet-name>"
      ]
    },
    "timeoutInMins": <batch-timeout>,
    "batchReportMode": "REPORT_AGGREGATED_BATCH" | "REPORT_INDIVIDUAL_BUILDS"
  },
  "concurrentBuildLimit": <concurrent-build-limit>
}
```

以下に置き換えます。

[名前]

必須。このビルドプロジェクトの名前。この名前は、 AWS アカウント内のすべてのビルドプロジェ クトで一意である必要があります。

[Description](説明)

オプション。このビルドの説明。

source

必須。このビルドプロジェクトのソースコード設定に関する情報が含まれている、<u>ProjectSource</u> オ ブジェクト。source オブジェクトを追加したら、 を使用して最大 12 個のソースを追加できます。 これらの設定には以下が含まれます。

source/type

必須。ビルドするソースコードを含むリポジトリのタイプ。有効な値を次に示します。

• CODECOMMIT

- CODEPIPELINE
- GITHUB
- GITHUB_ENTERPRISE
- GITLAB
- GITLAB_SELF_MANAGED
- BITBUCKET
- S3
- NO_SOURCE

NO_SOURCE を使用すると、プロジェクトにはソースがないため、buildspec をファイルとして使 用できません。代わりに、buildspec 属性を使用して buildspec に YAML 形式の文字列を指定 する必要があります。詳細については、「<u>ソースなしでビルドプロジェクトを作成</u>」を参照して ください。

source/location

<<u>source-type</u>> を CODEPIPELINE に設定しない場合は必須です。指定されたリポジトリタイ プのソースコードの場所。

- CodeCommit の場合は、ソースコードと buildspec ファイルが格納されているリポジトリの HTTPS クローン URL(例: https://git-codecommit.<*region-id*>.amazonaws.com/ v1/repos/<*repo-name*>)。
- Amazon S3 では、ビルド入力バケット名の後に、ソースコードと buildspec を含む ZIP ファイ ルのパスと名前が続きます。次に例を示します。
 - 入力バケットのルートにある ZIP ファイルの場合: <bucket-name>/<objectname>.zip。
 - 入力バケットのサブフォルダーにある ZIP ファイルの場合: <bucket-name>/<subfolerpath>/<object-name>.zip。
- GitHubの場合は、ソースコードと buildspec ファイルが格納されているリポジトリへの HTTPS クローン URL。URL には github.com が含まれている必要があります。 AWS アカウン トを GitHub アカウントに接続する必要があります。これを行うには、CodeBuild コンソール を使用してビルドプロジェクトを作成します。
 - [Authorize application] を選択します。(GitHub アカウントに接続した後、ビルドプロジェクトの作成を完了する必要はありません。CodeBuild コンソールを閉じることができます。)
- ・ GitHub Enterprise Server の場合は、ソースコードと buildspec ファイルを含むリポジトリへの HTTP または HTTPS クローン URL。また、 AWS アカウントを GitHub Enterprise Server ア

カウントに接続する必要があります。これを行うには、CodeBuild コンソールを使用してビル ドプロジェクトを作成します。

- 1. GitHub Enterprise Server で個人用アクセストークンを作成します。
- 2. このトークンをクリップボードにコピーし、CodeBuild プロジェクトの作成時に使用しま す。詳細については、GitHub Help ウェブサイトの <u>Creating a personal access token for the</u> command line を参照してください。
- 3. コンソールを使用して CodeBuild プロジェクトを作成する場合、[ソース] の [ソースプロバ イダー] で [GitHub Enterprise] を選択します。
- 4. [個人用アクセストークン] には、クリップボードにコピーしたトークンを貼り付けます。 [トークンの保存] を選択します。これで、CodeBuild アカウントが GitHub Enterprise Server アカウントに接続されました。
- GitLab および GitLab セルフマネージドの場合は、ソースコードと buildspec ファイルが格納されているリポジトリへの HTTPS クローン URL です。GitLab を使用する場合は、URL に gitlab.com を含める必要があります。GitLab セルフマネージドを使用する場合、URL に gitlab.com を含める必要はありません。 AWS アカウントを GitLab または GitLab セルフマ ネージドアカウントに接続する必要があります。これを行うには、CodeBuild コンソールを使 用してビルドプロジェクトを作成します。
 - デベロッパーツールのナビゲーションペインで [設定]、[接続]、[接続を作成] の順に選択します。このページで、GitLab または GitLab セルフマネージド接続を作成し、[GitLab に接続] を選択します。
- Bitbucket の場合は、ソースコードと buildspec ファイルが格納されているリポジトリへの HTTPS クローン URL。URL には bitbucket.org が含まれている必要があります。また、 AWS アカウントを Bitbucket アカウントに接続する必要があります。これを行うには、CodeBuild コンソールを使用してビルドプロジェクトを作成します。
 - 1. コンソールを使用して Bitbucket に接続 (または再接続) する場合は、Bitbucket の [Confirm access to your account] ページで、[Grant access] を選択します (Bitbucket アカウントに接続した後、ビルドプロジェクトの作成を完了する必要はありません。CodeBuild コンソールを閉じることができます。)
- の場合は AWS CodePipeline、 locationの値を指定しないでくださいsource。CodePipeline ではパイプラインを作成するときに、パイプラインのソースステージでソースコードの場所を 指定するため、この値は CodePipeline では無視されます。

source/gitCloneDepth

オプション。ダウンロードする履歴の深さ。最小値は 0 です。この値が 0、あるいは 25 より大 きいか指定されていない場合、完全な履歴が各ビルドプロジェクトと共にダウンロードされま す。ソースタイプが Amazon S3 である場合、この値はサポートされません。

source/buildspec

オプション。使用するビルド仕様定義またはファイル。この値が指定されていない場合や、 空の文字列に設定されている場合、ソースコードのルートディレクトリに buildspec.yml ファイルが含まれている必要があります。この値が設定されている場合は、インラインのビ ルド仕様定義か、プライマリソースのルートディレクトリからの相対的な代替 buildspec ファ イルへのパス、S3 バケットへのパスになります。バケットは、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します (例: arn:aws:s3:::

source/auth

ビルドするソースコードにアクセスするための CodeBuild の承認設定に関する情報が含まれています。

source/auth/type

必須。使用する権限付与タイプ。次の値を指定できます:

- OAUTH
- CODECONNECTIONS
- SECRETS_MANAGER

source/auth/resource

オプション。指定した権限付与タイプに適用されるリソース値。これは Secrets Manager ARN または CodeConnections ARN になります。

source/reportBuildStatus

ビルドの開始と完了のステータスをソースプロバイダーに送信するかどうかを指定します。これ を GitHub、GitHub Enterprise Server、または Bitbucket 以外のソースプロバイダーに対して設定 すると、invalidInputException がスローされます。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられ たユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き 込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、 「ソースプロバイダーのアクセス」を参照してください。

source/buildStatusConfig

CodeBuild ビルドプロジェクトがソースプロバイダにビルドステータスを報告す る方法を定義する情報が含まれています。このオプションは、ソースタイプが GITHUB、GITHUB_ENTERPRISE、または BITBUCKET の場合にのみ使用されます。

source/buildStatusConfig/context

Bitbucket リソースでは、このパラメータは、Bitbucket コミットステータスの name パラメー タに使用されます。GitHub ソースでは、このパラメータは、GitHub コミットステータスの context パラメータに使用されます。

例えば、context には、CodeBuild 環境変数を使用したビルド番号と webhook トリガーが含まれます。

AWS CodeBuild sample-project Build #\$CODEBUILD_BUILD_NUMBER - \$CODEBUILD_WEBHOOK_TRIGGER

これにより、webhook プルリクエストイベントによってトリガーされた build #24 では、コン テキストは次のようになります。

AWS CodeBuild sample-project Build #24 - pr/8

source/buildStatusConfig/targetUrl

Bitbucket リソースでは、このパラメータは、Bitbucket コミットステータスの url パラメー タに使用されます。GitHub ソースでは、このパラメータは、GitHub コミットステータスの target_url パラメータに使用されます。

たとえば、「targetUrl」と「https://aws.amazon.com/codebuild/<*path to build*>」とコミットステータスをこのURLにリンクします。

また、URL に情報を追加するには、CodeBuild 環境変数を targetUrl に含めることもでき ます。例えば、ビルド領域を URL に追加するには、targetUrl を以下に設定します:

"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=
\$AWS_REGION"

ビルド領域が us-east-2 の場合、これは次のように展開されます。

https://aws.amazon.com/codebuild/<path to build>?region=us-east-2

source/gitSubmodulesConfig

オプション。Git サブモジュール設定に関する情報。CodeCommit、GitHub、GitHub Enterprise Server、Bitbucket でのみ使用されます。

source/gitSubmodulesConfig/fetchSubmodules

リポジトリに Git サブモジュールを含める場合は、fetchSubmodules を true に設定しま す。含まれている Git サブモジュールは HTTPS として設定する必要があります。

source/InsecureSsl

オプション。GitHub Enterprise Server でのみ使用されます。GitHub Enterprise Server プロジェ クトリポジトリに接続するときの TLS 警告を無視するには、この値を true に設定します。デ フォルト値は false です。InsecureSsl は、テスト目的でのみ使用してください。本番環境で は使用しないでください。

source/sourceldentifier

プロジェクトソースのユーザー定義識別子。プライマリソースの場合、省略可能です。セカンダ リソースでは必須です。

secondarySources

オプション。ビルドプロジェクトのセカンダリソースに関する情報が含まれている、<u>ProjectSource</u> オブジェクトの配列。最大 12 個のセカンダリソースを追加できます。secondarySources オブ ジェクトは、 オブジェクトで使用されるのと同じプロパティを使用します。セカンダリソースオブ ジェクトでは、sourceIdentifier は必須です。

secondarySourceVersions

オプション。<u>ProjectSourceVersion</u> オブジェクトの配列。secondarySourceVersions をビルド レベルで指定すると、これよりも優先されます。

sourceVersion

オプション。このプロジェクト用に構築するビルド入力のバージョン。指定しない場合、最新のバー ジョンが使用されます。指定した場合、次のいずれかであることが必要です。

- CodeCommit の場合: 使用するコミット ID、ブランチ、または Git タグ。
- GitHub の場合、ビルドするソースコードのバージョンに対応するコミット ID、プルリクエスト ID、ブランチ名、またはタグ名。プルリクエスト ID を指定する場合、pr/pull-request-ID (例: pr/25) 形式を使用する必要があります。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。指定しない場合は、デフォルトブランチの HEAD コミット ID が使用され ます。
- GitLab の場合、コミット ID、プルリクエスト ID、ブランチ名、タグ名、または参照およびコミット ID です。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照してください。
- Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ名、またはタグ名。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。指定しない場合は、デフォルトブランチの HEAD コミット ID が使用されます。
- Amazon S3 の場合、使用するビルド入力 ZIP ファイルを表すオブジェクトのバージョン ID。

sourceVersion をビルドレベルで指定した場合、そのバージョンはこの (プロジェクトレベルの) sourceVersion より優先されます。詳細については、「<u>を使用したソースバージョンサンプル</u> AWS CodeBuild」を参照してください。

artifacts

必須。このビルドプロジェクトの出力アーティファクト設定に関する情報が含まれてい る、<u>ProjectArtifacts</u> オブジェクト。artifacts オブジェクトを追加したら、 を使用して最大 12 個 のアーティファクトを追加できます。これらの設定には以下が含まれます。

artifacts/type

必須。ビルド出力アーティファクトのタイプ。有効な値は次のとおりです。

- CODEPIPELINE
- NO_ARTIFACTS
- S3

artifacts/location

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

前提条件で作成または識別した出力バケットの名前。

artifacts/path

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

ZIP ファイルまたはフォルダを配置する出力バケットのパス。path の値を指定しない場 合、CodeBuild では namespaceType (指定されている場合) と name を使用して、ビルド 出力 ZIP ファイルまたはフォルダのパスと名前を決定します。たとえば、MyPath を path に、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/MyArtifact.zip」にな ります。

artifacts/namespaceType

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

ビルド出力 ZIP ファイルまたはフォルダの名前空間。有効な値は、BUILD_ID および NONE です。BUILD_ID を使用してビルド出力 ZIP ファイルまたはフォルダのパスにビルド ID を挿 入します。それ以外の場合は、NONE を使用します。namespaceType の値を指定しない場 合、CodeBuild では path (指定されている場合) と name を使用して、ビルド出力 ZIP ファイ ルまたはフォルダのパスと名前を決定します。たとえば、MyPath を path に、BUILD_ID を namespaceType、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/build-ID/MyArtifact.zip」になります。

artifacts/name

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

location 内のビルド出力 ZIP ファイルまたはフォルダの名前。たとえば、MyPath を path に、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/MyArtifact.zip」にな ります。

artifacts/overrideArtifactName

S3 アーティファクトタイプ でのみ使用されます。他のアーティファクトタイプには使用されま せん。

オプション。true に設定すると、buildspec ファイルの artifacts ブロックで指定された名 前が、name を上書きします。詳細については、「<u>CodeBuild のビルド仕様に関するリファレン</u> ス」を参照してください。

artifacts/packaging

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

オプション。アーティファクトをパッケージ化する方法を指定します。許可された値は次のとお りです:

なし

ビルドアーティファクトを含むフォルダを作成します。これは、デフォルト値です。 ZIP

ビルドアーティファクトを含む ZIP ファイルを作成します。

secondaryArtifacts

オプション。ビルドプロジェクトのセカンダリアーティファクト設定に関する情報が含まれてい る、<u>ProjectArtifacts</u> オブジェクトの配列。最大 12 個のセカンダリアーティファクトを追加できま す。secondaryArtifacts は、 オブジェクトで使用されているのと同じ設定の多くを使用しま す。

cache

必須。このビルドプロジェクトのキャッシュ設定に関する情報が含まれている、<u>ProjectCache</u>オブ ジェクト。詳細については、「キャッシュビルド」を参照してください。

環境

必須。このプロジェクトのビルド環境設定に関する情報が含まれている、<u>ProjectEnvironment</u> オブ ジェクト。設定は次のとおりです。

environment/type

必須。構築環境のタイプ。詳細については、CodeBuild API リファレンスの「<u>型</u>」を参照してく ださい。

environment/image

必須。このビルド環境で使用される Docker イメージ識別子。通常、この識別子は *image-name:tag* として表されます。例えば、CodeBuild で Docker イメージの管理に使用する Docker リポジトリの場合、これは aws/codebuild/standard:5.0 です。Docker Hub で は、maven:3.3.9-jdk-8 です。Amazon ECR では、*account-id*.dkr.ecr.*region-* *id*.amazonaws.com/*your-Amazon-ECR-repo-name*:*tag* です。詳細については、 「CodeBuild に用意されている Docker イメージ」を参照してください。

environment/computeType

必須。このビルド環境で使用されるコンピュートリソースを指定します。詳細について は、CodeBuild API リファレンスの「computeType」を参照してください。

environment/certificate

オプション。Amazon S3 バケットの ARN、パスのプレフィックス、および PEM エンコー ドされた証明書を含むオブジェクトキー。オブジェクトキーとして、PEM エンコードされ た証明書が含まれている .pem ファイルまたは .zip ファイルのいずれかを使用できます。た とえば、Amazon S3 バケット名が *<my-bucket*>、パスのプレフィックスが *<cert*>、オ ブジェクトキー名が *<certificate.pem*> である場合、certificate に使用できる形式 は *<my-bucket/cert/certificate.pem*> または arn:aws:s3:::*<my-bucket/cert/ certificate.pem*> です。

environment/environmentVariables

オプション。このビルド環境に指定する環境変数が含まれている、<u>EnvironmentVariable</u>オブ ジェクトの配列。各環境変数は、オブジェクトとして表されます。name、value、および type の name、value、 および type。

コンソールと AWS CLI ユーザーは、すべての環境変数を表示できます。環境変数の表示に懸念がない場合は、「name」を「value」および「type」を「PLAINTEXT」に設定します。

Amazon EC2 Systems Manager パラメータストアまたは のパラメータとして、 AWS アクセス キー ID、 AWS シークレットアクセスキー、パスワードなどの機密性の高い値を持つ環境変数を 保存することをお勧めします AWS Secrets Manager。name の場合、保存されているパラメータ については、CodeBuild の識別子を参照するように設定します。

Amazon EC2 Systems Manager パラメータストアを使用する場合、value には、パラメータス トアに保存されているとおりにパラメータの名前を設定します。type を PARAMETER_STORE に設定します。/CodeBuild/dockerLoginPassword という名前のパラメータを使用す るには、たとえば、「name」を「LOGIN_PASSWORD」に設定。value を /CodeBuild/ dockerLoginPassword に設定します。type を PARAMETER_STORE に設定します。

🛕 Important

Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは / CodeBuild/ で始まるパラメータ名(例: /CodeBuild/dockerLoginPassword)で保 存することをお勧めします。CodeBuild コンソールを使用して、Amazon EC2 Systems Manager にパラメータを作成することができます。[パラメータの作成] を選択し、ダイ アログボックスの手順に従います。(このダイアログボックスの KMS キーでは、アカウ ントで AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager はこの キーを使用して、ストレージ中にパラメータの値を暗号化し、取得中に復号します。) CodeBuild コンソールを使用してパラメータを作成した場合、コンソールは保存されてい る /CodeBuild/ パラメータ名を開始します。詳細については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。 ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存 されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に [New service role] (新しいサービスロール)を選択した場合は、CodeBuild のビルドプロジェクトのデフォル トのサービスロールにこのアクションが含まれています。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。 ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、「新しいサー ビスロール]を選択した場合、/CodeBuild/ で始まらないパラメータ名にアクセスで きるようにサービスロールを更新する必要があります。これは、サービスロールで、/ CodeBuild/で始まるパラメータ名にのみアクセスが許可されるためです。 [新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータを 復号するアクセス権限が含まれます。

既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれ ていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場 合、my_value が other_value に置き換えられます。同様に、Docker イメージに / usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に 含まれていて、\$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数 を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 \$PATH:/usr/ share/ant/bin に置き換えられます。

CODEBUILD_で始まる名前の環境変数は設定しないでください。このプレフィックスは 内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定され ます。

ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。

- ビルドプロジェクト定義の値が次に優先されます。
- ・ ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合、value には、Secrets Manager に保存されているパラ メータの名前を設定します。type を SECRETS_MANAGER に設定します。/CodeBuild/ dockerLoginPassword という名前のシークレットを使用するには、たとえば、「name」を 「LOGIN_PASSWORD」に設定。value を /CodeBuild/dockerLoginPassword に設定しま す。type を SECRETS_MANAGER に設定します。

A Important

Secrets Manager を使用する場合は、「/CodeBuild/」で始まる名前でシークレットを 保存することをお勧めします(たとえば、/CodeBuild/dockerLoginPassword)。詳細 については、 AWS Secrets Managerユーザーガイドの「<u>AWS Secrets Manager とは</u>」を 参照してください。

ビルドプロジェクトが Secrets Manager パラメータストアに保存されて

いるパラメータを参照する場合、ビルドプロジェクトのサービスロールで

secretsmanager:GetSecretValue アクションを許可する必要があります。以前に [New service role] (新しいサービスロール) を選択した場合は、CodeBuild のビルドプロ ジェクトのデフォルトのサービスロールにこのアクションが含まれています。ただし [既 存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含め る必要があります。

ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Secrets Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した場 合、/CodeBuild/ で始まらないシークレット名にアクセスできるようにサービスロール を更新する必要があります。これは、サービスロールで、/CodeBuild/ で始まるシーク レット名にのみアクセスが許可されるためです。

[新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets Manager の /CodeBuild/ 名前空間ですべてのシークレットを復号するアクセス許可が 含まれます。

environment/registryCredential

オプション。プライベート Docker レジストリへのアクセスを提供する認証情報を指定する <u>RegistryCredential</u> オブジェクト。 environment/registryCredential/credential

AWS Managed Servicesを使用して作成された認証情報の ARN または名前を指定します。認証情報の名前を使用できるのは、認証情報が現在のリージョン内に存在する場合のみです。

environment/registryCredential/credentialProvider

唯一の有効な値は SECRETS_MANAGER です。

これを設定した場合:

• imagePullCredentials を SERVICE_ROLE に設定する必要があります。

• 選別されたイメージや Amazon ECR イメージは使用できません。

environment/imagePullCredentialsType

オプション。ビルドのイメージをプルするために CodeBuild で使用する認証情報のタイプ。2 つ の有効な値があります。

CODEBUILD

CODEBUILD は、CodeBuild で独自の認証情報を使用することを指定します。CodeBuild サー ビスプリンシパルを信頼するには、Amazon ECR リポジトリポリシーを編集する必要があり ます。

SERVICE_ROLE

CodeBuild でビルドプロジェクトのサービスロールを使用することを指定します。

クロスアカウントまたはプライベートレジストリイメージを使用する場合は、SERVICE_ROLE の認証情報を使用する必要があります。CodeBuild の選別されたイメージを使用する場合 は、CODEBUILD の認証情報を使用する必要があります。

environment/privilegedMode

このビルドプロジェクトを使用して Docker イメージをビルドする計画の場合のみ、true に設 定します。それ以外の場合、関連付けられているビルドで Docker デーモンと通信しようとする と、すべて失敗します。ビルドが Docker デーモンと連係動作できるように、Docker デーモンも 起動する必要があります。これを行う 1 つの方法は、次のビルドコマンドを実行して buildspec ファイルの install フェーズで Docker デーモンを初期化することです。Docker をサポートす る CodeBuild によって提供されるビルド環境イメージを指定した場合は、これらのコマンドを実 行しないでください。 Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビルド に Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime Privilege</u> <u>and Linux Capabilities</u>」を参照して、特権モードを有効にします。また、Windows は特 権モードをサポートしていません。

- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock -host=tcp://127.0.0.1:2375 --storage-driver=overlay2 & - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"

serviceRole

必須。CodeBuild がユーザーに代わってサービスとやり取りするために使用するサービスロールの ARN (例: arn:aws:iam::*account-id*:role/*role-name*)。

autoRetryLimit

オプション。ビルドが失敗した後、さらに自動で再試行する回数です。例えば、自動再試行の制限が 2 に設定されている場合、CodeBuild は RetryBuild API を呼び出して、さらに最大 2 回までビル ドを自動的に再試行します。

timeoutInMinutes

オプション。5〜2,160 分 (36 時間) の分単位の時間。この時間が経過してもビルドが完了してい ない場合、CodeBuild はビルドを停止します。指定しない場合は、デフォルトの 60 が使用されま す。CodeBuild がタイムアウトによりビルドを停止したかどうか、およびそのタイミングを確認す るには、batch-get-builds コマンドを実行します。ビルドが停止しているかどうかを確認するに は、出力で FAILED の buildStatus 値を調べます。ビルドがタイムアウトした時間を確認するに は、出力で TIMED_OUT のphaseStatus 値に関連付けられている endTime 値を調べます。

queuedTimeoutInMinutes

オプション。5〜480 分 (8 時間) の分単位の時間。この時間が経過すると、ビルドがキューされてい る場合に CodeBuild によってビルドが停止されます。指定しない場合は、デフォルトの 60 が使用さ れます。

encryptionKey

オプション。CodeBuild AWS KMS key がビルド出力を暗号化するために使用する のエイリアスまた は ARN。エイリアスを指定する場合に、arn:aws:kms:*region-ID*:*account-ID*:key/*key-ID* 形式を使用し、エイリアスが存在する場合には、alias/*key-alias* 形式を使用します。指定しな い場合、Amazon S3 の AWSマネージド KMS キーが使用されます。 Amazon S3

tags

オプション。このビルドプロジェクトに関連付けるタグを提供する <u>Tag</u> オブジェクトの配列。最大 50 個のタグを指定できます。これらのタグは、CodeBuild ビルドプロジェクトタグをサポートする 任意の AWS サービスで使用できます。各タグは、「key」と「value」オブジェクトとして表現さ れます。

vpcConfig

オプション。<u>VpcConfig</u> オブジェクト。プロジェクトの VPC 設定に関する情報を含む。詳細につい ては、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

これらのプロパティには、次のものがあります。

vpcld

必須。CodeBuild で使用される VPC ID。リージョン内の VPC ID を一覧表示するには、次のコマ ンドを実行します。

aws ec2 describe-vpcs --region <region-ID>

サブネット

必須。CodeBuild で使用されるリソースを含むサブネット ID の配列。これらの ID を取得するに は、次のコマンドを実行します。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-
ID>
```

securityGroupIds

必須。VPC 内のリソースへのアクセスを許可するために CodeBuild で使用されるセキュリティグ ループ ID の配列。これらの ID を取得するには、次のコマンドを実行します。
```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-
ID>
```

badgeEnabled

オプション。CodeBuild プロジェクトにビルドバッジを含めるかどうかを指定します。true に設 定してビルドバッジを有効にするか、そうでない場合は false に設定します。詳細については、 「CodeBuild でのビルドバッジサンプル」を参照してください。

logsConfig

このビルドのログが配置されている場所に関する情報が含まれている、LogsConfig オブジェクト。

logsConfig/cloudWatchLogs

CloudWatch Logs へのログのプッシュに関する情報が含まれている、<u>CloudWatchLogsConfig</u> オ ブジェクト。

logsConfig/s3Logs

Amazon S3 へのログのプッシュに関する情報が含まれている、S3LogsConfig オブジェクト。

fileSystemLocations

オプション。Amazon EFS 設定に関する情報が含まれている、<u>ProjectFileSystemsLocation</u> オブジェ クトの配列。

buildBatchConfig

オプション。buildBatchConfig オブジェクトは <u>ProjectBuildBatchConfig</u> 構造体であり、プロ ジェクトのバッチビルド設定情報を含みます。

buildBatchConfig/serviceRole

バッチビルドプロジェクトのサービスロール ARN を指定します。

buildBatchConfig/combineArtifacts

バッチビルドのビルドアーティファクトを 1 つのアーティファクトの場所に結合するかどうかを 指定するブール値。 buildBatchConfig/restrictions/maximumBuildsAllowed

許可されるビルドの最大数。

buildBatchConfig/restrictions/computeTypesAllowed

バッチビルドで許可されるコンピューティングタイプを指定する文字列の配列。これらの値につ いては、「ビルド環境のコンピューティングタイプ」を参照してください。

buildBatchConfig/restrictions/fleetsAllowed

バッチビルドに許可されるフリートを指定する文字列の配列。詳細については、<u>「リザーブド</u> キャパシティフリートでビルドを実行する」を参照してください。

buildBatchConfig/timeoutInMinutes

バッチビルドを完了するまでの最大時間 (分単位)。

buildBatchConfig/batchReportMode

バッチビルドのソースプロバイダーにビルドステータスレポートを送信する方法を指定します。 有効な値を次に示します。

REPORT_AGGREGATED_BATCH

(デフォルト)すべてのビルドステータスを1つのステータスレポートに集約します。 REPORT INDIVIDUAL BUILDS

個々のビルドごとに個別のステータスレポートを送信します。

concurrentBuildLimit

このジョブで許可される同時実行の最大数を設定します。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウント がこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

プロジェクトの作成

プロジェクトを作成するには、<u>create-project</u> コマンドを再度実行し、JSON ファイルを渡しま す。

aws codebuild create-project --cli-input-json file://<json-file>

成功した場合、JSON 表現の <u>Project</u> オブジェクトが、コンソール出力に表示されます。このデータ の例については、「CreateProject レスポンスの構文」を参照してください。

ビルドプロジェクトの名前を除いて、後でビルドプロジェクトの設定を変更することができます。詳 細については、「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。

ビルドの実行を開始するには、「ビルドの実行 (AWS CLI)」を参照してください。

ソースコードが GitHub リポジトリに保存されていて、コード変更がリポジトリにプッシュされるた びに CodeBuild でソースコードを再構築する場合は、「<u>ビルドの実行の自動開始 (AWS CLI)</u>」を参 照してください。

ビルドプロジェクトを作成する (AWS SDKs)

SDK AWS CodeBuild で を使用する方法については、「」を参照してください<u>AWS SDKsとツール</u> のリファレンス。 AWS SDKs

ビルドプロジェクトの作成 (AWS CloudFormation)

AWS CodeBuild で を使用する方法については AWS CloudFormation、AWS CloudFormation 「 ユー ザーガイド」のCodeBuild の AWS CloudFormation テンプレート」を参照してください。

通知ルールの作成

通知ルールを使用すると、ビルドの成功や失敗などの重要な変更が発生したときにユーザーに通知で きます。通知ルールは、イベントと、通知の送信に使用される Amazon SNS トピックの両方を指定 します。詳細については、「通知とは」を参照してください。

コンソールまたは を使用して AWS CLI 、通知ルールを作成できます AWS CodeBuild。

通知ルールを作成するには (コンソール)

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codebuild/</u>:// www.com」で CodeBuild コンソールを開きます。
- 2. [Build (ビルド)]、[Build projects (ビルドプロジェクト)] の順に選択し、通知を追加するビルドプロジェクトを選択します。
- ビルドプロジェクトページで、[Notify (通知)]、[Create notification rule (通知ルールの作成)] の順 に選択します。ビルドプロジェクトの [Settings (設定)] ページに移動し、[Create notification rule (通知ルールの作成)] を選択することもできます。

- 4. [通知名] に、ルールの名前を入力します。
- 5. Amazon EventBridge に提供された情報のみを通知に含める場合は、[Detail type (詳細タイプ)] で [Basic (基本)] を選択します。Amazon EventBridge に提供される情報に加えて、CodeBuild または通知マネージャから提供される場合がある情報も含める場合は、[完全] を選択します。

詳細については、「通知の内容とセキュリティについて」を参照してください。

- [Events that trigger notifications (通知をトリガーするイベント)] で、通知を送信するイベントを 選択します。詳細については、「ビルドプロジェクトでの通知ルールのイベント」を参照してく ださい。
- 7. [Targets (ターゲット)] で、次のいずれかの操作を行います。
 - 通知で使用するリソースをすでに設定している場合は、ターゲットタイプを選択す るで、チャットアプリケーション (Slack) で Amazon Q Developer または SNS トピックを選 択します。ターゲットの選択で、クライアントの名前 (チャットアプリケーションで Amazon Q Developer で設定された Slack クライアントの場合) または Amazon SNS トピックの Amazon リソースネーム (ARN) (通知に必要なポリシーで既に設定された Amazon SNS ト ピックの場合) を選択します。
 - 通知で使用するリソースを設定していない場合は、[Create target]、[SNS topic] の順に選択します。codestar-notifications-の後にトピックの名前を指定し、[Create] を選択します。

Note

- 通知ルールの作成の一環として Amazon SNS トピックを作成すると、トピックへの イベント発行を通知機能に許可するポリシーが適用されます。通知ルール用に作成し たトピックを使用すると、このリソースに関する通知を受信するユーザーのみをサブ スクライブできます。
- 通知ルールの作成の一環として、チャットアプリケーションクライアントで Amazon Q Developer を作成することはできません。チャットアプリケーション (Slack) で Amazon Q Developer を選択すると、チャットアプリケーションで Amazon Q Developer でクライアントを設定するように指示するボタンが表示されます。 このオプションを選択すると、チャットアプリケーションコンソールで Amazon Q Developer が開きます。詳細については、「Configure Integrations Between Notifications and Amazon Q Developer in chat applications」を参照してください。
- 既存の Amazon SNS トピックをターゲットとして使用する場合は、このトピック用の他のすべてのポリシーに加えて、AWS CodeStar Notifications に必要なポリシーを

追加する必要があります。詳細については、「<u>通知用の Amazon SNS トピックを設</u> 定する」および「通知の内容とセキュリティについて」を参照してください。

- 8. ルールの作成を終了するには、[Submit (送信)]を選択します。
- 通知を受け取るには、そのルールの Amazon SNS トピックにユーザーをサブスクライブする必要があります。詳細については、「<u>ターゲットである Amazon SNS トピックへのユーザーのサブスクライブ</u>」を参照してください。チャットアプリケーションで通知と Amazon Q Developer の統合を設定して、Amazon Chime チャットルームに通知を送信することもできます。詳細については、「Configure Integration Between Notifications and Amazon Q Developer in chat applications」を参照してください。

通知ルールを作成するには (AWS CLI)

1. ターミナルまたはコマンドプロンプトで、create-notification rule コマンドを実行して、JSON スケルトンを生成します。

ファイルには任意の名前を付けることができます。この例では、ファイルの名前を *rule.json* とします。

プレーンテキストエディタで JSON ファイルを開き、これを編集してルールに必要なリソース、イベントタイプ、ターゲットを含めます。次の例は、ID が 123456789012 AWS アカウントで MyBuildProject という名前のMyNotificationRuleビルドプロジェクトの という名前の通知ルールを示しています。ビルドが成功したとき、通知は、完全な詳細タイプで Amazon SNS トピック codestar-notifications-MyNotificationTopic に送信されます:

```
}
],
"Status": "ENABLED",
"DetailType": "FULL"
}
```

ファイルを保存します。

3. 先ほど編集したファイルを使用して、ターミナルまたはコマンドラインで、create-notificationrule コマンドを再度実行し、通知ルールを作成します。

```
aws codestarnotifications create-notification-rule --cli-input-json
file://rule.json
```

4. 成功すると、コマンドは次のような通知ルールの ARN を返します。

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

でビルドプロジェクト設定を変更する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、または AWS SDKs を使用して、ビルドプロジェクトの設定 を変更できます。

ビルドプロジェクトにテストレポートを追加する場合は、<u>テストレポートのアクセス許可</u>で記載されている権限が IAM ロールに付与されていることを確認してください。

トピック

- ビルドプロジェクトの設定の変更 (コンソール)
- ・ ビルドプロジェクトの設定の変更 (AWS CLI)
- ・ <u>ビルドプロジェクトの設定の変更 (AWS</u> SDK)

ビルドプロジェクトの設定の変更 (コンソール)

ビルドプロジェクトの設定を変更するには、次の手順を実行します。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Build projects] を選択します。
- 3. 次のいずれかを行ってください。
 - 変更するビルドプロジェクトのリンクを選択し、[ビルドの詳細]を選択します。
 - 変更するビルドプロジェクトの横にあるラジオボタンを選択して、[View details (詳細を表示)]
 を選択後 [ビルドの詳細] を選択します。

次のセクションを変更できます。

セクション

- プロジェクトの設定
- ソース
- 環境
- Buildspec
- Batch 構成
- アーティファクト
- ログ

プロジェクトの設定

[プロジェクトの設定] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択 して新しい設定を保存します。

次のプロパティを変更できます。

説明

また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの 説明を任意で指定することもできます。

ビルドバッジ

[Enable build badge (ビルドバッジを有効にする)] を選択すると、プロジェクトのビルドステータ スが表示可能および埋め込み可能になります。詳細については、「<u>ビルドバッジサンプル</u>」を参 照してください。 Note

ソースプロバイダーが Amazon S3 の場合、ビルドバッジは適用されません。

同時ビルド制限を有効にする

このプロジェクトで同時ビルド数を制限するには、次の手順を実行します。

- [Restrict number of concurrent builds this project can start] (このジョブで許可される同時実行の最大数を設定)を選択します。
- [Concurrent build limit] (同時ビルド制限) で、このジョブで許可される同時実行の最大数を設定します。この制限は、アカウントに設定された同時ビルド制限より大きくすることはできません。アカウント制限を超える数値を入力しようとすると、エラーメッセージが表示されます。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウ ントがこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

パブリックビルドアクセスを有効にする

AWS アカウントにアクセスできないユーザーを含め、プロジェクトのビルド結果を一般公開す るには、パブリックビルドアクセスを有効にするを選択し、ビルド結果を公開することを確認し ます。パブリックビルドプロジェクトでは、次のプロパティが使用されます。

パブリックビルドのサービスロール

CodeBuild で新しいサービスロールを作成する場合は [New service role] (新しいサービスロール) を、既存のサービスロールを使用する場合は [Existing service role] (既存のサービスロール) を選択します。

パブリックビルドのサービスロールを使用することにより、CodeBuild で CloudWatch Logs を読み取り、プロジェクトのビルド用の Amazon S3 アーティファクトをダウンロードできま す。これは、プロジェクトのビルドログとアーティファクトを一般に公開するために必要で す。

サービスロール

新しいサービスロールまたは既存のサービスロールの名前を入力します。

プロジェクトのビルド結果をプライベートにするには、[Enable public build access] (パブリック ビルドアクセスを有効にする) のチェックを外します。 詳細については、「パブリックビルドプロジェクトの URL を取得」を参照してください。

▲ Warning プロジェクトのビルド結果を一般に公開する際には、以下に留意してください。

- プロジェクトがプライベートだったときに実行されたビルドも含めて、プロジェクトの
 ビルド結果、ログ、アーティファクトはすべて、一般に公開されます。
- すべてのビルドログとアーティファクトが一般に公開されます。環境変数、ソースコード、およびその他の機密情報がビルドログとアーティファクトに出力されている可能性があります。ビルドログに出力される情報には注意が必要です。以下にベストプラクティスを示します。
 - 機密性の高い値、特に AWS アクセスキー IDsとシークレットアクセスキーを環境変数に保存しないでください。Amazon EC2 Systems Manager パラメータストアまたは AWS Secrets Manager を使用して、機密性の高い値を保存することをお勧めします。
 - ウェブフック使用のベストプラクティス。
 に従って、ビルドをトリガーできるエン ティティを制限し、buildspec をプロジェクト自体に保存しないことで、Webhook を可能な限り安全に保つことができます。
- 悪意のあるユーザーがパブリックビルドを利用して、悪意のあるアーティファクトを配信する可能性があります。プロジェクト管理者は、すべてのプルリクエストを確認し、 プルリクエストが正当な変更であるか検証することをお勧めします。また、チェックサムを使ってすべてのアーティファクトを検証し、正しいアーティファクトがダウンロードされているか確認することを推奨します。

追加情報

タグには、サポート AWS サービスで使用するタグの名前と値を入力します。[Add row] を使用し て、タグを追加します。最大 50 個のタグを追加できます。

ソース

[ソース] セクションで [編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定 を保存します。

次のプロパティを変更できます。

ソースプロバイダー

ソースコードプロバイダーのタイプを選択します。次のリストを使用して、ソースプロバイダー に関する適切な選択を行います。

Note

CodeBuild は Bitbucket サーバーをサポートしていません。

Amazon S3

バケット

ソースコードが格納されている入力バケットの名前を選択します。

S3 オブジェクトキーまたは S3 フォルダ

ZIP ファイルの名前、またはソースコードを含むフォルダへのパスを入力します。S3 バケットの中身をすべてダウンロードするには、スラッシュ記号 (/) を入力します。

ソースバージョン

入力ファイルのビルドを表すオブジェクトのバージョン IDを入力。詳細については、「<u>を使</u> 用したソースバージョンサンプル AWS CodeBuild」を参照してください。

CodeCommit

リポジトリ

使用するリポジトリを選択します。

参照タイプ

[Branch] (ブランチ) または [Git tag] (Git タグ) を選択するか、[Commit ID] (コミット ID) を入 力して、ソースコードのバージョンを指定します。詳細については、「<u>を使用したソースバー</u> ジョンサンプル AWS CodeBuild」を参照してください。 Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全ク ローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

Bitbucket

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections]、[OAuth]、[アプリパスワード]、または [個人用アクセストークン] を選択 して CodeBuild に接続します。

Connection

Bitbucket 接続または Secrets Manager シークレットを選択して、指定した接続タイプ経由で 接続します。

リポジトリ

[Bitbucket アカウントのリポジトリ] または [パブリックリポジトリ] を選択し、リポジトリ URL を入力します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細につい ては、「を使用したソースバージョンサンプル AWS CodeBuild」を参照してください。 Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、Bitbucket コミットステータスの name パラ メータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「<u>ビル</u> <u>ド</u>」を参照してください。

[Target URL] (ターゲットURL) に、Bitbucket コミットステータスの url パラメータに使用す る値を記入します。詳細については、Bitbucket API ドキュメントの「<u>ビルド</u>」を参照してく ださい。

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポート されます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告さ れた API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。 [Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築します。Webhook およびフィルターグループの詳細については、「<u>Bitbucket ウェブフックイベン</u>ト」を参照してください。

GitHub

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[GitHub アプリ]、[OAuth]、または [個人用アクセストークン] を選択して CodeBuild に接続し ます。

Connection

GitHub 接続または Secrets Manager シークレットを選択して、指定した接続タイプ経由で接続します。

リポジトリ

[GitHub アカウントのリポジトリ]、[パブリックリポジトリ]、または [GitHub スコープ付き ウェブフック] を選択し、リポジトリ URL を入力します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細につい ては、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照してください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。 Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、GitHub コミットステータスの contextパラ メータに使用する値を記入します。 q 詳細については、GitHub デベロッパーガイドの「<u>コ</u> ミットステータスの作成」を参照してください。

[Target URL] (ターゲット URL) に、 GitHub コミットステータスの target_url パラメー タに使用する値を記入します。詳細については、GitHub デベロッパーガイドの「<u>コミットス</u> テータスの作成」を参照してください。

webhook によってトリガーされたビルドのステータスは、常にソースプロバイダーにレポー トされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告 された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。

[Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび 再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築 します。Webhook およびフィルターグループの詳細については、「<u>GitHub ウェブフックイベン</u>ト」を参照してください。

GitHub Enterprise Server

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] または [個人用アクセストークン] を選択して CodeBuild に接続します。 Connection

GitHub Enterprise 接続または Secrets Manager シークレットを選択して、指定した接続タイプ経由で接続します。

リポジトリ

[自分の GitHub Enterprise アカウントのレポジトリ] または [GitHub Enterprise スコープ付き ウェブフック] を選択し、リポジトリ URL を入力します。

ソースバージョン

プルリクエスト、ブランチ、コミット ID、コミット ID、参照、およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択し ます。 ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、GitHub コミットステータスの contextパラ メータに使用する値を記入します。 q 詳細については、GitHub デベロッパーガイドの「<u>コ</u> ミットステータスの作成」を参照してください。

[Target URL] (ターゲット URL) に、 GitHub コミットステータスの target_url パラメー タに使用する値を記入します。詳細については、GitHub デベロッパーガイドの「<u>コミットス</u> テータスの作成」を参照してください。

webhook によってトリガーされたビルドのステータスは、常にソースプロバイダーにレポー トされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告 された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効に するには、新しいコミットをリポジトリにプッシュする必要があります。

安全でない SSL

[Enable insecure SSL (セキュアでない SSL を有効にする)] を選択して、GitHub Enterprise プ ロジェクトリポジトリに接続するときの SSL 警告を無視します。

[Primary source webhook events] (プライマリソース Webhook イベント) で [Rebuild every time a code change is pushed to this repository] (コード変更がこのリポジトリにプッシュされるたび 再構築) を選択して、コード変更がこのリポジトリにプッシュされるたびに CodeBuild で再構築 します。Webhook およびフィルターグループの詳細については、「<u>GitHub ウェブフックイベン</u>ト」を参照してください。

GitLab

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] は、GitLab を CodeBuild に接続するために使用されます。

Connection

CodeConnections 経由で接続する GitLab 接続を選択します。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、または参照およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

GitLab Self Managed

[認証情報]

[デフォルトソース認証情報] または [カスタムソース認証情報] を選択し、手順に従ってデフォ ルトソース認証情報を管理するか、ソース認証情報をカスタマイズします。

[接続タイプ]

[CodeConnections] は、GitLab セルフマネージドを CodeBuild に接続するために使用されます。

Connection

CodeConnections 経由で接続する GitLab セルフマネージド接続を選択します。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、または参照およびコミット ID を入力し ます。詳細については、「<u>を使用したソースバージョンサンプル AWS CodeBuild</u>」を参照し てください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コ ミット ID と似ていない Git ブランチ名を選択することをお勧めします。これによ り、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いク ローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソース プロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けら れたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが 書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細につい ては、「ソースプロバイダーのアクセス」を参照してください。

環境

[環境] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定 を保存します。

次のプロパティを変更できます。

[プロビジョニングモデル]

プロビジョニングモデルを変更するには、[プロビジョニングモデルを変更] を選択し、次のいず れかを実行します。

- が管理するオンデマンドフリートを使用するには AWS CodeBuild、オンデマンドを選択します。オンデマンドフリートでは、CodeBuild がビルドのコンピューティングを行います。マシンはビルドが終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応できる自動スケーリング機能を備えています。
- が管理するリザーブドキャパシティフリートを使用するには AWS CodeBuild、リザーブドキャ パシティを選択し、フリート名を選択します。リザーブドキャパシティフリートでは、ビルド 環境に合わせて専有インスタンスのセットを設定します。これらのマシンはアイドル状態のま まで、ビルドやテストをすぐに処理できる状態になり、ビルド時間を短縮します。リザーブド キャパシティフリートでは、マシンは常に稼働しており、プロビジョニングされている間はコ ストが発生し続けます。

詳細については、<u>リザーブドキャパシティキャパシティフリートでビルドを実行</u> を参照してくだ さい。

環境イメージ

ビルドイメージを変更するには、[イメージの上書き]を選択し、次のいずれかを実行します。

- が管理する Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム、ランタイム (複数可)、イメージ、イメージバージョンから 選択します。利用可能な場合は、[環境タイプ]から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)]を選択した場合は、[External registry URL (外部のレジスト リ URL)] に docker repository/docker image name の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択 します。
- プライベート Docker イメージを使用するには、[カスタムイメージ]を選択します。[Environment type (環境タイプ)]で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を 選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を 選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報 は、Secrets Manager で作成する必要があります。詳細については、 AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager とは」を参照してください。
 - Note

CodeBuild はカスタムDocker イメージの「ENTRYPOINT」をオーバーライドします。

サービスロール

次のいずれかを行ってください。

- CodeBuild サービスロールがない場合は、[新しいサービスロール] を選択します。[Role name]
 に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、[Existing service role (既存のサービスロール)] を選 択します。[Role ARN] で、サービスロールを選択します。

Note

コンソールでは、ビルドプロジェクトの作成時に CodeBuild サービスロールも作成でき ます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソー ルでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルド プロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個の ビルドプロジェクトで使用できます。

追加設定

タイムアウト

5 分~36 時間の間の値を指定します。この時間が経過してもビルドが完了していない場 合、CodeBuild はビルドを停止します。[hours] と [minutes] を空白のままにすると、デフォル ト値の 60 分が使用されます。

特権付与

[Docker イメージをビルドする場合、またはビルドで昇格された権限を取得する場合は、この フラグを有効にする] を選択します。それ以外の場合、関連付けられているビルドで Docker デーモンと通信しようとすると、すべて失敗します。ビルドが Docker デーモンと連係動作で きるように、Docker デーモンも起動する必要があります。これを行う 1 つの方法は、次のビ ルドコマンドを実行してビルドスペックの install フェーズで Docker デーモンを初期化す ることです。Docker をサポートする CodeBuild によって提供されるビルド環境イメージを選 択した場合は、これらのコマンドを実行しないでください。

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビ ルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime</u> <u>Privilege and Linux Capabilities</u>」を参照して、特権モードを有効にします。ま た、Windows は特権モードをサポートしていません。

- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock -host=tcp://127.0.0.1:2375 --storage-driver=overlay2 & - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"

VPC

CodeBuild を VPC と連携させたい場合

- [VPC] で、CodeBuild が使用する VPC ID を選択します。
- ・ [VPC Subnets (サブネット)] で、CodeBuild が使用するリソースを含むサブネットを選択します。
- ・ [VPC Security groups (VPC セキュリティグループ)] で、CodeBuild が VPC 内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「<u>Amazon Virtual Private Cloud AWS CodeBuild で を使用する</u>」を参照し てください。

コンピューティング

使用可能なオプションの1つを選択します。

レジストリ認証情報

プロジェクトが非プライベートレジストリイメージで設定されている場合は、レジストリ認証 情報を指定します。

Note

この認証情報は、イメージがプライベートレジストリのイメージで上書きされている 場合にのみ使用されます。

環境変数

[環境変数] で、名前と値を入力してから、ビルドによって使用される各環境変数の種類を選択 します。

Note

CodeBuild は、 AWS リージョンの環境変数を自動的に設定します。以下の環境変数を buildspec.yml に追加していない場合は、それらの変数を設定する必要があります。

- AWS_ACCOUNT_ID
- IMAGE_REPO_NAME
- IMAGE_TAG

コンソールと AWS CLI ユーザーは環境変数を表示できます。環境変数の表示に懸念がない場合は、[Name] および [Value] フィールドを設定し、[Type] を [Plaintext] に設定します。

アクセスキー ID、 AWS シークレット AWS アクセスキー、パスワードなどの機密性の高い値 を持つ環境変数をパラメータとして Amazon EC2 Systems Manager パラメータストアまたは に保存することをお勧めします AWS Secrets Manager。 Amazon EC2 Systems Manager パラメータストアを使用する場合は、[Type (タイプ)] で、 [Parameter (パラメータ)] を選択します。[Name] (名前) に、参照する CodeBuild の識別子を 入力します。[Value] (値) に、Amazon EC2 Systems Manager パラメータストアに保存され ているパラメータの名前を入力します。たとえば、/CodeBuild/dockerLoginPassword という名前のパラメータを使用して、[タイプ] で [Parameter (パラメータ)] を選択しま す。[Name (名前)] に LOGIN_PASSWORD と入力します。[Value (値)] に「/CodeBuild/ dockerLoginPassword」と入力します。

▲ Important

Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは / CodeBuild/ で始まるパラメータ名 (例: /CodeBuild/dockerLoginPassword) で保存することをお勧めします。CodeBuild コンソールを使用して、Amazon EC2 Systems Manager にパラメータを作成することができます。[パラメータの作成] を 選択し、ダイアログボックスの手順に従います。(このダイアログボックスの KMS キーでは、アカウントで AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager はこのキーを使用して、ストレージ中にパラメータの値を暗号化 し、取得中に復号します。)CodeBuild コンソールを使用してパラメータを作成した 場合、コンソールは保存されている /CodeBuild/ パラメータ名を開始します。詳細 については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュー トリアル」を参照してください。

ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存 されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に [New service role] (新しいサービスロール)を選択した場合は、CodeBuild のビルドプロジェクト のデフォルトのサービスロールにこのアクションが含まれています。ただし [既存の サービスロール] を選択した場合は、このアクションをサービスロールに個別に含め る必要があります。

ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、[新 しいサービスロール] を選択した場合、/CodeBuild/ で始まらないパラメータ名に アクセスできるようにサービスロールを更新する必要があります。これは、サービス ロールで、/CodeBuild/ で始まるパラメータ名にのみアクセスが許可されるためで す。 [新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータ を復号するアクセス権限が含まれます。

既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれ ていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場 合、my_value が other_value に置き換えられます。同様に、Docker イメージ に /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数 が既に含まれていて、\$PATH:/usr/share/ant/bin の値を持つ PATH という名前 の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 \$PATH:/usr/share/ant/bin に置き換えられます。

CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックス は内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定 されます。

- ・ ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ・ ビルドプロジェクト定義の値が次に優先されます。
- ・ ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合は、[Type] (タイプ) で、[Secrets Manager] を選択しま す。[Name] (名前) に、参照する CodeBuild の識別子を入力します。[Value (値)] に、パターン reference-key を使用して *secret-id:json-key:version-stage:version-id* を入 力します。詳細については、<u>Secrets Manager reference-key in the buildspec file</u> を参照して ください。

▲ Important

Secrets Manager を使用する場合は、「/CodeBuild/」で始まる名前で シークレットを保存することをお勧めします(たとえば、/CodeBuild/ dockerLoginPassword)。詳細については、 AWS Secrets Managerユーザーガイ ドの「<u>AWS Secrets Manager とは</u>」を参照してください。 ビルドプロジェクトが Secrets Manager パラメータストアに保存されて いるパラメータを参照する場合、ビルドプロジェクトのサービスロールで secretsmanager:GetSecretValue アクションを許可する必要があります。以前 に [New service role] (新しいサービスロール)を選択した場合は、CodeBuild のビルド プロジェクトのデフォルトのサービスロールにこのアクションが含まれています。た だし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに 個別に含める必要があります。 ビルドプロジェクトが、/CodeBuild/で始まらないパラメータ名を持つ、Secrets Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した 場合、/CodeBuild/で始まらないシークレット名にアクセスできるようにサービス ロールを更新する必要があります。これは、サービスロールで、/CodeBuild/で始 まるシークレット名にのみアクセスが許可されるためです。 [新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets

[Minor y CodeBuild/名前空間ですべてのシークレットを復号するアクセス許可 が含まれます。

Buildspec

[Buildspec] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい 設定を保存します。

次のプロパティを変更できます。

ビルド仕様

次のいずれかを行ってください。

- ソースコードにビルド仕様ファイルが含まれている場合は、[Use a buildspec file (buildspec ファイルを使用)] を選択します。デフォルトでは、CodeBuild はソースコードのルートディレクトリで buildspec.yml という名前のファイルを探します。buildspec ファイルに別の名前または場所が使用されている場合は、Buildspec 名 にソースルートからのパスを入力します(例えば、buildspec-two.yml または configuration/buildspec.yml)。buildspec ファイルが S3 バケットにある場合は、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します(例: arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml)。
- ソースコードにビルド仕様ファイルが含まれていない場合、または、ソースコードのルート ディレクトリで build ファイルの buildspec.yml フェーズに指定されているものと異なる ビルドコマンドを実行する場合は、[ビルドコマンドの挿入]を選択します。[ビルドコマンド] に、build フェーズで実行するコマンドを入力します。複数のコマンドについては、&& で各 コマンドを区切ります (例: mvn test && mvn package)。他のフェーズでコマンドを実行 する場合、または build フェーズのコマンドの長いリストがある場合は、ソースコマンドの

ルートディレクトリに buildspec.yml ファイルを追加し、ファイルにコマンドを追加してか ら、[Use the buildspec.yml in the source code root directory] (ソースコードのルートディレクト リの「buildspec.yml」を使用) を選択します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

Batch 構成

[バッチ構成] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新し い設定を保存します。詳細については、「ビルドをバッチで実行」を参照してください。

次のプロパティを変更できます。

Batch サービスロール

バッチビルドのサービスロールを提供します。

次のいずれかを選択します。

- バッチサービスロールがない場合は、[New service role] (新しいサービスロール) を選択します。[Service role] (サービスロール) に、新しいロールの名前を入力します。
- バッチサービスロールがある場合は、[Existing service role] (既存のサービスロール) を選択します。[Service role] (サービスロール) で、サービスロールを選択します。

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロール では、CodeBuild が StartBuild、StopBuild および RetryBuild アクションを使用して、 バッチの一部としてビルドを実行する上で必要です。次の2つの理由により、お客様はビルドで 使用するものと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用 すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドと計算タイプの数を制限 する制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を 回避する可能性があります。

バッチで許可されるコンピューティングタイプ

バッチに使用できる計算タイプを選択します。該当するものをすべて選択します。 バッチに許可されるフリート

バッチに許可されているフリートを選択します。該当するものをすべて選択します。

バッチで許可される最大ビルド

バッチで許可されるビルドの最大数を入力します。バッチがこの制限を超えると、バッチは失敗 します。

バッチのタイムアウト

バッチビルドが完了する最大時間を入力します。

アーティファクトの結合

[Combine all artifacts from batch into a single location] (バッチのすべてのアーチファクト) を1つ の場所に結合するを選択して、バッチのすべてのアーチファクトを単一の場所に結合します。 バッチレポートモード

バッチビルドに対して望ましいビルドステータスレポートモードを選択します。

Note

このフィールドが利用可能になるのは、プロジェクトソースが Bitbucket、GitHub、ま たは GitHub Enterprise であり、[Source] (ソース) で [Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダー にビルドステータスをレポートする) が選択されている場合のみです。

集約されたビルド

これを選択して、バッチ内にあるすべてのビルドのステータスを単一のステータスレポートに まとめます。

個々のビルド

これを選択して、バッチ内にあるすべてのビルドのビルドステータスが個別に報告されるよう にします。

アーティファクト

[アーティファクト] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択し て新しい設定を保存します。

次のプロパティを変更できます。

Type

次のいずれかを行ってください。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts] を選択します。ビルドテストのみを実行している場合や、Docker イメージを Amazon ECR リポジトリにプッシュする場合には、これを行うことができます。
- ビルド出力をS3バケットに保存する場合は、[Amazon S3]を選択して次のいずれかの操作を 行います。
 - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)]
 を空白のままにします。それ以外の場合は、名前を入力します。(ZIP ファイルを出力して
 ZIP ファイルにファイル拡張子を付ける場合は、必ず ZIP ファイル名の後に含めます)。
 - buildspec ファイルで指定した名前で、コンソールで指定した名前を上書きする場合は、 [Enable semantic versioning (セマンティックバージョニングを有効にする)]を選択しま す。buildspec ファイル内の名前は、ビルド時に計算され、Shell コマンド言語を使用しま す。たとえば、アーティファクト名に日付と時刻を追加して常に一意にできます。アーティ ファクト名を一意にすると、アーティファクトが上書きされるのを防ぐことができます。詳 細については、「buildspec の構文」を参照してください。
 - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
 - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド 出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の 場所の場合は、各場所をコンマで区切ります (例: appspec.yml, target/my-app.jar)。 詳細については、「files」で <u>buildspec の構文</u>の説明を参照してください。
 - ビルドアーティファクトを暗号化しない場合は、[アーティファクト暗号化の削除]を選択します。

アーティファクトのセカンダリセットごとに:

- 1. [Artifact 識別子] には、英数字とアンダースコアのみを使用して 128 文字未満の値を入力しま す。
- 2. [アーティファクトの追加]を選択します。
- 3. セカンダリアーティファクトを設定するには、前のステップに従います。
- 4. [アーティファクトの保存]を選択します。

追加設定

暗号化キー

次のいずれかを行います:

- アカウントで AWS マネージドキー Amazon S3 を使用してビルド出力アーティファクトを 暗号化するには、暗号化キーを空白のままにします。これがデフォルトです。
- カスタマー管理のキーを使用してビルド出力アーティファクトを暗号化するには、[暗号化キー]にカスタマー管理のキーのARNを入力します。arn:aws:kms:*region-ID*:account-ID:key/key-IDの形式を使用します。

キャッシュタイプ

[キャッシュタイプ] で、以下のいずれかを選択します。

- ・ キャッシュを使用しない場合は、[No cache] を選択します。
- Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
 - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
 - (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パスのプレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名に似ています。これにより、バケット内の同じディレクトリにキャッシュを保存できます。

▲ Important

パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

 ローカルキャッシュを使用する場合は、[ローカル]を選択し、ローカルキャッシュモードを 1つ以上選択します。

Note

Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択 する場合、プロジェクトは権限モードで実行する必要があります。

キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使 用されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定 に関する詳細については、「<u>buildspec の構文</u>」を参照してください。キャッシングの詳細に ついては、「パフォーマンスを向上させるためのキャッシュビルド」を参照してください。

ログ

[ログ] セクションで [編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を 保存します。

次のプロパティを変更できます。

作成するログを選択します。Amazon CloudWatch Logs、Amazon S3 ログ、または両方のログを作 成できます。

CloudWatch

Amazon CloudWatch Logs が必要な場合:

CloudWatch Logs

[CloudWatch logs] を選択します。

グループ名

Amazon CloudWatch Logs のログのグループ名を入力します。

ストリーム名

Amazon CloudWatch Logs ログストリーム名を入力します。

S3

Amazon S3 ログが必要な場合は、以下のようになります。

S3 ログ

[S3 logs (S3 ログ)] を選択します。

バケット

ログを保存する S3 バケットの名前を選択します。 パスプレフィックス

ログのプレフィックスを入力します。

S3 ログの暗号化を無効にする

S3 ログを暗号化しない場合は、選択します。

ビルドプロジェクトの設定の変更 (AWS CLI)

AWS CLI で を使用する方法については AWS CodeBuild、「」を参照してください<u>コマンドライン</u> リファレンス。

で CodeBuild プロジェクトを更新するには AWS CLI、更新されたプロパティを含む JSON ファイル を作成し、そのファイルを <u>update-project</u> コマンドに渡します。更新ファイルに含まれていない プロパティは変更されません。

更新 JSON ファイルでは、name プロパティおよび変更されたプロパティのみが必要です。name プロパティにより、変更するプロジェクトを識別します。変更された構造については、それらの構造に必要なパラメータも含める必要があります。たとえば、プロジェクトの環境を変更するには、「environment/type」および「environment/computeType」プロパティが必要です。環境イメージを更新する例を次に示します。

```
{
   "name": "<project-name>",
   "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:4.0"
   }
}
```

プロジェクトの現在のプロパティ値を取得する必要がある場合は、<u>batch-get-projects</u> コマンドを使 用して、変更するプロジェクトの現在のプロパティを取得し、出力をファイルに書き込みます。

aws codebuild batch-get-projects --names "<project-name>" > project-info.json

project-info.json ファイルには、プロジェクトの配列が含まれているため、プロジェクト を更新するために直接使用することはできません。ただし、変更したいプロパティを projectinfo.json ファイルからコピーして更新ファイル内に貼り付け、変更するプロパティのベースライ ンとすることができます。詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」 を参照してください。

「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」の説明に従って、更新 JSON ファイルを変更し、結果を 保存します。更新 JSON ファイルの変更が完了したら、<u>update-project</u> コマンドを実行し、更新 JSON ファイルを渡します。 aws codebuild update-project --cli-input-json file://<update-project-file>

成功した場合、更新されたプロジェクトの JSON が出力に表示されます。必要なパラメータが欠落 している場合は、欠落しているパラメータを識別するエラーメッセージが出力に表示されます。たと えば、このエラーメッセージは、「environment/type」パラメータが無いエラーです。

aws codebuild update-project --cli-input-json file://update-project.json

Parameter validation failed: Missing required parameter in environment: "type"

ビルドプロジェクトの設定の変更 (AWS SDK)

SDK AWS CodeBuild で を使用する方法については、「」を参照してください<u>AWS SDKsとツール</u> のリファレンス。 AWS SDKs

CodeBuild の複数のアクセストークン

CodeBuild は、 内 AWS Secrets Manager または AWS CodeConnections 接続を介してシークレット からサードパーティープロバイダーにアクセストークンを調達することをサポートしています。シー クレットまたは接続は、GitHub、GitHub Enterprise、Bitbucket などの指定されたサードパーティー プロバイダとのやり取りのデフォルトの認証情報として設定できます。

ソース認証情報は、次の3つの異なるレベルで設定できます。

- すべてのプロジェクトのアカウントレベルの認証情報: これらは、AWS アカウント内のすべての プロジェクトのデフォルトの認証情報です。プロジェクトまたはソースレベルの認証情報が指定 されていない場合、プロジェクトで使用されます。
- 特定のリポジトリのソースレベルの認証情報: これは、プロジェクトソースで Secrets Manager シークレットまたは CodeConnections 接続が定義されている場合です。これらの認証情報は、指 定されたソースリポジトリでのオペレーションにのみ使用されます。これにより、同じプロジェ クトで異なるアクセス許可スコープを持つ複数のアクセストークンを設定でき、デフォルトのア カウントレベルの認証情報を使用しないようにすることができます。
- プロジェクトレベルのフォールバック認証情報: プライマリソースタイプとして NO_SOURCE を使用してプロジェクトレベルのフォールバック認証情報を設定し、それにシークレットまたは接続を定義できます。これは、プロジェクトに複数のソースがあるものの、同じ認証情報をそれらのソースに使用する場合、またはプロジェクトのデフォルトのアカウントレベルの認証情報を使用しない場合に使用できます。

トピック

- ステップ 1: Secrets Manager シークレットまたは CodeConnections 接続を作成
- <u>ステップ 2: CodeBuild プロジェクトの IAM ロールに Secrets Manager シークレットへのアクセス</u> を許可
- <u>ステップ 3: Secrets Manager または CodeConnections トークンを設定</u>
- 追加のセットアップオプション

ステップ 1: Secrets Manager シークレットまたは CodeConnections 接続 を作成

Secrets Manager シークレットまたは CodeConnections 接続を作成するには、次の手順に従いま す。

- Secrets Manager シークレットにトークンを作成して保存.
- <u>GitHub への接続を作成</u>
- ・ GitHub Enterprise Server への接続を作成
- Bitbucket への接続を作成

ステップ 2: CodeBuild プロジェクトの IAM ロールに Secrets Manager シークレットへのアクセスを許可

Note

続行する前に、Secrets Manager または CodeConnections で作成されたトークンにアクセス できるようにしておく必要があります。

CodeBuild プロジェクトの IAM ロールに Secrets Manager または CodeConnections へのアクセスを 許可するには、次の IAM ポリシーを追加する必要があります。

CodeBuild プロジェクトの IAM ロールにアクセスを許可するには

- CodeBuild プロジェクトの CodeBuild が他の AWS のサービスとやり取りすることを許可 の手順に従って、CodeBuild プロジェクトの IAM ロールを作成します。
- 2. 次のいずれかを行います:

CodeBuild プロジェクトロールに次の IAM ポリシーを追加して、シークレットへのアクセスを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "secretsmanager:GetSecretValue"
        ],
        "Resource": [
               "<secret-arn>"
        ]
        }
    ]
}
```

(オプション) AWS KMS カスタマーマネージドキーを使用して Secrets Manager シーク レットを暗号化する場合は、次のポリシーステートメントを追加してアクセスを許可できま す。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "kms:Decrypt"
            ],
            "Resource": "<kms-key-arn>",
            "Condition": {
                 "StringEquals": {
                     "kms:EncryptionContext:SecretARN": "<secret-arn>"
                }
            }
        }
    ]
}
```

 CodeBuild プロジェクトロールに次の IAM ポリシーを追加して、接続へのアクセスを許可 します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
             "codeconnections:GetConnectionToken",
             "codeconnections:GetConnection"
        ],
        "Resource": [
             <connection-arn>
        ]
      }
]
```

ステップ 3: Secrets Manager または CodeConnections トークンを設定

Secrets Manager トークンまたは CodeConnections トークンを使用して、ソース認証情報を3つの 異なるレベルで設定できます。

Secrets Manager または CodeConnections トークンをアカウントレベルの認証情報として設定

Secrets Manager シークレットまたは CodeConnections 接続をアカウントレベルの認証情報として 設定し、プロジェクトで使用できます。

AWS Management Console

でアカウントレベルの認証情報として接続を設定するには AWS Management Console

- 1. [ソースプロバイダ] には、[Bitbucket]、[GitHub]、または [GitHub Enterprise] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - [デフォルトソース認証情報]を選択し、アカウントのデフォルトソース認証情報を使用 して、すべてのプロジェクトに適用します。
 - a. ソースプロバイダに接続していない場合は、[デフォルトソース認証情報を管理] を 選択します。
 - b. [認証情報タイプ]では、認証情報タイプを選択します。

c. [CodeConnections] を選択した場合は、既存の接続を使用するか、新しい接続を作 成することを選択します。

別の認証情報タイプを選択した場合は、[サービス] でトークンの保存に使用する サービスを選択し、以下を実行します。

- [Secrets Manager] を使用することを選択した場合は、既存のシークレット接続 を使用するか、新しいシークレットを作成して [保存] を選択できます。新しい シークレットの作成方法の詳細については、「<u>Secrets Manager シークレット</u> にトークンを作成して保存」を参照してください。
- [CodeBuild] を使用することを選択した場合は、トークンまたはユーザー名とア プリパスワードを入力し、[保存] を選択します。
- [カスタムソース認証情報]を選択し、カスタムソース認証情報を使用してアカウントの デフォルト設定を上書きします。
 - a. [認証情報タイプ]では、認証情報タイプを選択します。
 - b. [接続] で、既存の接続を使用するか、新規の接続を作成することを選択します。

AWS CLI

でアカウントレベルの認証情報として接続を設定するには AWS CLI

 ターミナル (Linux/macOS/Unix) またはコマンドプロンプト (Windows) を開きます。 AWS CLI を使用して import-source-credentials コマンドを実行します。

次のコマンドを使用して Secrets Manager のシークレットを設定します。

```
aws codebuild import-source-credentials \
    --token "<secret-arn>" \
    --server-type <source-provider> \
    --auth-type SECRETS_MANAGER \
    --region <aws-region>
```

次のコマンドを使用して CodeConnections 接続を設定します。

```
aws codebuild import-source-credentials \
    --token "<connection-arn>" \
    --server-type <source-provider> \
    --auth-type CODECONNECTIONS \
```
--region <aws-region>

このコマンドを使用すると、アカウントレベルのデフォルトソース認証情報としてトークン をインポートできます。<u>ImportSourceCredentials</u> API を使用して認証情報をインポートする 場合、CodeBuild は、より具体的な認証情報のセットがプロジェクトで設定されていない限 り、ウェブフック、ビルドステータスレポート、git clone オペレーションなど、ソースプロ バイダとのすべてのやり取りにそのトークンを使用します。

これで、ビルドプロジェクトでトークンを使用して実行できるようになりました。詳細について は、<u>でビルドプロジェクトを作成する AWS CodeBuild</u>および<u>AWS CodeBuild ビルドを手動で実行す</u> るを参照してください。

複数のトークンをソースレベルの認証情報として設定

Secrets Manager シークレットまたは CodeConnections 接続をソースレベルの認証情報として使用 するには、CodeBuild プロジェクトのトークンを直接参照し、ビルドを開始します。

AWS Management Console

でソースレベルの認証情報として複数のトークンを設定するには AWS Management Console

- 1. [ソースプロバイダー] で [GitHub] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - [デフォルトソース認証情報]を選択し、アカウントのデフォルトソース認証情報を使用 して、すべてのプロジェクトに適用します。
 - a. GitHub に接続していない場合は、[デフォルトソース認証情報を管理] を選択します。
 - b. [認証情報タイプ] では、[GitHub アプリ] を選択します。
 - c. [接続] で、既存の接続を使用するか、新規の接続を作成することを選択します。
 - [カスタムソース認証情報]を選択し、カスタムソース認証情報を使用してアカウントの デフォルト設定を上書きします。
 - a. [認証情報タイプ] では、[GitHub アプリ] を選択します。
 - b. [接続]で、既存の接続を使用するか、新規の接続を作成することを選択します。
- 3. [ソースを追加]を選択し、ソースプロバイダと認証情報を選択するプロセスを繰り返しま す。

AWS CLI

でソースレベルの認証情報として複数のトークンを設定するには AWS CLI

 ターミナル (Linux/macOS/Unix) またはコマンドプロンプト (Windows) を開きます。 AWS CLI を使用して create-project コマンドを実行します。

以下のコマンドを使用します。

```
aws codebuild create-project --region <aws-region> \
    --name <project-name> \
    --artifacts type=NO_ARTIFACTS \
    --environment "type=LINUX_CONTAINER,
                   computeType=BUILD_GENERAL1_SMALL,
                   image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
    --service-role <service-role-name> \
    --source "type=GITHUB,
              location=<github-repository-1>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-1>}"
 /
    --secondary-sources "type=GITHUB,
              location=<github-repository-2>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-2>},
              sourceIdentifier=secondary"
aws codebuild start-build --region <aws-region> --project-name <project-name>
```

プロジェクトレベルのソース認証情報のフォールバックを設定

プロジェクトレベルのソース認証情報のフォールバックを設定するには、プロジェクトのプライマリ ソースに NO_SOURCE を使用し、トークンを参照します。

aws codebuild start-build --region <aws-region> --project-name <project_name>

NO_SOURCE を使用する場合、通常、buildspec は外部ソースを使用して <u>buildspec</u> を取得するように直接設定されていないため、ソースモデル内で提供されます。通常、NO_SOURCE ソースは buildspec 内からすべての関連するリポジトリのクローンを作成します。設定済みの認証情報をこれ らのオペレーションで使用できるようにするには、buildspec で git-credential-helper オプ ションを有効にします。

env:

git-credential-helper: yes

ビルド中、CodeBuild は設定されたトークンから AuthServer フィールドを読み取り、その特定の サードパーティソースプロバイダへのすべての git リクエストにトークン認証情報を使用します。

追加のセットアップオプション

AWS CloudFormation テンプレートを使用して Secrets Manager のアカウントレベルの認証情報を 設定できます。次の AWS CloudFormation テンプレートを使用して、アカウントレベルの認証情報 を設定できます。

```
Parameters:
  GitHubToken:
    Type: String
    NoEcho: true
    Default: placeholder
Resources:
  CodeBuildAuthTokenSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: CodeBuild auth token
      Name: codebuild-auth-token
      SecretString:
        !Join
          _ !!
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubToken
```

- '"}'
Tags:
- Key: codebuild:source:provider
Value: github
- Key: codebuild:source:type
Value: personal_access_token
CodeBuildSecretsManagerAccountCredential:
Type: AWS::CodeBuild::SourceCredential
Properties:
ServerType: GITHUB
AuthType: SECRETS_MANAGER
Token: !Ref CodeBuildAuthTokenSecret

Note

同じスタックにプロジェクトを作成する場合は、 AWS CloudFormation 属性 <u>DependsOn</u> を使用して、プロジェクトの前に AccountCredential が作成されていることを確認しま す。

AWS CloudFormation テンプレートを使用して Secrets Manager の複数のソースレベルの認証情報 を設定することもできます。次の AWS CloudFormation テンプレートを使用して、複数のトークン を使用して複数のソースをプルできます。

Parameters: GitHubTokenOne: Type: String NoEcho: true Default: placeholder GitHubTokenTwo: Type: String NoEcho: true Default: placeholder Resources: CodeBuildSecretsManagerProject: Type: AWS::CodeBuild::Project **Properties:** Name: codebuild-multitoken-example ServiceRole: <service-role> Environment: Type: LINUX_CONTAINER

```
ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/amazonlinux-x86_64-standard:5.0
    Source:
      Type: GITHUB
      Location: <github-repository-one>
      Auth:
        Type: SECRETS_MANAGER
        Resource: !Ref CodeBuildAuthTokenSecretOne
    SecondarySources:
      - Type: GITHUB
        Location: <github-repository-two>
        Auth:
          Type: SECRETS_MANAGER
          Resource: !Ref CodeBuildAuthTokenSecretTwo
        SourceIdentifier: secondary
    Artifacts:
      Type: NO_ARTIFACTS
    LogsConfig:
      CloudWatchLogs:
        Status: ENABLED
CodeBuildProjectIAMRoleSecretAccess:
  Type: AWS::IAM::RolePolicy
  Properties:
    RoleName: <role-name>
    PolicyName: CodeBuildProjectIAMRoleSecretAccessPolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:

    secretsmanager:GetSecretValue

          Resource:
            - !Ref CodeBuildAuthTokenSecretOne
            - !Ref CodeBuildAuthTokenSecretTwo
CodeBuildAuthTokenSecretOne:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: CodeBuild auth token one
    Name: codebuild-auth-token-one
    SecretString:
      !Join
        _ ''
        - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
          - !Ref GitHubTokenOne
```

- '"}'	
Tags:	
- Key: codebuild:source:provider	
Value: github	
- Key: codebuild:source:type	
Value: personal_access_token	
CodeBuildAuthTokenSecretTwo:	
Type: AWS::SecretsManager::Secret	
Properties:	
Description: CodeBuild auth token two	
Name: codebuild-auth-token-two	
SecretString:	
!Join	
_ ''	
'{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'	
- !Ref GitHubTokenTwo	
- '"}'	
Tags:	
- Key: codebuild:source:provider	
Value: github	
- Key: codebuild:source:type	
Value: personal_access_token	

でビルドプロジェクトを削除する AWS CodeBuild

CodeBuild コンソール AWS CLI、または AWS SDKs を使用してCodeBuild でビルドプロジェクトを 削除できます。プロジェクトを削除しても、そのビルドは削除されません。

▲ Warning

ビルドとリソースポリシーを持つプロジェクトは削除できません。リソースポリシーとビル ドを持つプロジェクトを削除するには、最初にリソースポリシーを削除し、そのビルドを削 除する必要があります。

トピック

- ビルドプロジェクトの削除 (コンソール)
- ・ビルドプロジェクトの削除 (AWS CLI)
- ・ビルドプロジェクトの削除 (AWS SDKs)

ビルドプロジェクトの削除 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Build projects] を選択します。
- 3. 次のいずれかを行ってください。
 - ・ 削除するビルドプロジェクトの横にあるラジオボタンを選択して、[削除]を選択します。
 - ・ 削除するビルドプロジェクトのリンクを選択し、[Delete] を選択します。

Note

デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多 くのビルドプロジェクトを表示するには、[Projects per page] で別の値を選択するか、 [Viewing projects] で前後の矢印を選択します。

ビルドプロジェクトの削除 (AWS CLI)

1. delete-project コマンドを実行します。

aws codebuild delete-project --name name

次のプレースホルダを置き換えます。

- name: 必須の文字列。削除するビルドプロジェクトの名前。使用可能なビルドプロジェクトの リストを取得するには、list-projects コマンドを実行します。詳細については、「ビル ドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。
- 2. 成功した場合、データは出力されず、エラーも出力に表示されません。

AWS CLI で を使用する方法の詳細については AWS CodeBuild、「」を参照してください<u>コマンド</u> ラインリファレンス。

ビルドプロジェクトの削除 (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> <u>ツールのリファレンス</u>。 AWS SDKs

パブリックビルドプロジェクトの URL を取得

AWS CodeBuild を使用すると、ビルドプロジェクトのビルド結果、ログ、アーティファクトを一般 公開できます。これにより、ソースリポジトリのコントリビューターは、 AWS アカウントへのアク セスを必要とせずに、ビルドの結果を表示したり、アーティファクトをダウンロードしたりできま す。

プロジェクトのビルドを一般に公開すると、プロジェクトがプライベートだったときに実行された ビルドも含めて、プロジェクトのビルド結果、ログ、アーティファクトはすべて、一般に公開されま す。同様に、パブリックビルドプロジェクトをプライベートにすると、そのプロジェクトのビルド結 果は一般に公開されなくなります。

プロジェクトのビルド結果の公開範囲を変更する方法については、「<u>パブリックビルドアクセスを有</u> 効にする」を参照してください。

CodeBuild では、お客様のプロジェクトに固有のパブリックビルド用 URL を提供しています。

ビルドプロジェクトのパブリック URL を取得するには、以下の手順を実行します。

パブリックビルドプロジェクトの URL を取得するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. ナビゲーションペインで、[ビルドプロジェクト]を選択します。
- 3. パブリック URL を取得したいビルドプロジェクトのリンクを選択します。
- 4. パブリック URL は、[Configuration] (構成) セクションの [Public project URL] (パブリックプロ ジェクト URL) フィールドに表示されます。リンクを選択して URL を開くか、コピーボタンを 使用して URL をコピーすることができます。

▲ Warning

プロジェクトのビルド結果を一般に公開する際には、以下に留意してください。

- プロジェクトがプライベートだったときに実行されたビルドも含めて、プロジェクトのビルド結果、ログ、アーティファクトはすべて、一般に公開されます。
- すべてのビルドログとアーティファクトが一般に公開されます。環境変数、ソースコード、およびその他の機密情報がビルドログとアーティファクトに出力されている可能性があります。ビルドログに出力される情報には注意が必要です。以下にベストプラクティスを示します。
 - 機密値、特に AWS アクセスキー IDsとシークレットアクセスキーを環境変数に保存 しないでください。Amazon EC2 Systems Manager パラメータストアまたは AWS Secrets Manager を使用して、機密性の高い値を保存することをお勧めします。
 - ウェブフック使用のベストプラクティス。
 に従って、ビルドをトリガーできるエンティ ティを制限し、buildspec をプロジェクト自体に保存しないことで、Webhook を可能な 限り安全に保つことができます。
- 悪意のあるユーザーがパブリックビルドを利用して、悪意のあるアーティファクトを配信 する可能性があります。プロジェクト管理者は、すべてのプルリクエストを確認し、プ ルリクエストが正当な変更であるか検証することをお勧めします。また、チェックサムを 使ってすべてのアーティファクトを検証し、正しいアーティファクトがダウンロードされ ているか確認することを推奨します。

ビルドプロジェクトを共有

プロジェクト共有を使用すると、プロジェクト所有者は自分の AWS CodeBuild プロジェクトを他の AWS アカウントやユーザーと共有できます。このモデルでは、プロジェクトを所有するアカウント (所有者) は、他のアカウント (コンシューマー) とプロジェクトを共有します。コンシューマーは、 プロジェクトを編集または実行できません。

トピック

- プロジェクトを共有
- 関連サービス
- 共有されている CodeBuild プロジェクトにアクセス
- 共有プロジェクトを共有解除
- 共有プロジェクトを識別
- 共有プロジェクトへのアクセス許可

プロジェクトを共有

コンシューマーは、 コンソール AWS CLI と AWS CodeBuild コンソールの両方を使用して、共有し たプロジェクトとビルドを表示できます。コンシューマーは、プロジェクトを編集または実行できま せん。

既存のリソース共有にプロジェクトを追加すること、そして、<u>AWS RAM コンソール</u>でプロジェク トを作成することもできます。

Note

リソース共有に追加されたビルドを含むプロジェクトは削除できません。

組織単位または組織全体でプロジェクトを共有するには、 AWS Organizationsとの共有を有効にする 必要があります。詳細については、AWS RAM ユーザーガイドの「<u>AWS Organizationsで共有を有効</u> 化する」を参照してください。

AWS CodeBuild コンソール、 AWS RAM コンソール、または AWS CLI を使用して、所有するプロ ジェクトを共有できます。

プロジェクトを共有するための前提条件

プロジェクトの共有を開始する前に、 AWS アカウントがプロジェクトを所有していることを確認し てください。自身が共有を受けているプロジェクトは共有できません。

所有するプロジェクトを共有するには (CodeBuild コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- 2. ナビゲーションペインで、[Build projects] を選択します。

Note

デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多 くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

3. 共有するプロジェクトを選択し、[Share (共有)] を選択します。詳細については、AWS RAM ユーザーガイドの「リソースの共有の作成」を参照してください。 所有しているプロジェクトを共有するには (AWS RAM コンソール)

「AWS RAM ユーザーガイド」の「リソース共有の作成」を参照してください。

所有しているプロジェクトを共有するには (AWS RAM コマンド)

create-resource-share コマンドを使用します。

所有するプロジェクトを共有するには (CodeBuild コマンド)

put-resource-policy コマンドを使用します:

1. policy.json という名前のファイルを作成し、その中に次をコピーします。

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
     "Principal":{
        "AWS":"<consumer-aws-account-id-or-user>"
     },
     "Action":[
        "codebuild:BatchGetProjects",
        "codebuild:BatchGetBuilds",
        "codebuild:ListBuildsForProject"],
     "Resource":"<arn-of-project-to-share>"
     }]
}
```

 共有するプロジェクト ARN と識別子で policy.json を更新します。次の の例で は、123456789012「」で識別される AWS アカウントのルートユーザーに読み取り専用アクセ スを許可します。

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Principal":{
            "AWS": [
               "123456789012"
        ]
        },
        "Action":[
```

```
"codebuild:BatchGetProjects",
    "codebuild:BatchGetBuilds",
    "codebuild:ListBuildsForProject"],
    "Resource":"arn:aws:codebuild:us-west-2:123456789012:project/my-project"
}]
}
```

3. put-resource-policy コマンドを使用します。

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. AWS RAM リソース共有 ARN を取得します。

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

これにより、次のような応答が得られます。

```
{
    "resources": [
    {
        "arn": "<project-arn>",
        "type": "<type>",
        "resourceShareArn": "<resource-share-arn>",
        "creationTime": "<creation-time>",
        "lastUpdatedTime": "<last-update-time>"
    }
]
}
```

応答から、<resource-share-arn>値は、次のステップで使用します。

5. AWS RAM promote-resource-share-created-from-policy コマンドを実行します。

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-
share-arn>
```

関連サービス

プロジェクト共有は AWS Resource Access Manager (AWS RAM) と統合されます。これは、 AWS アカウントまたは を通じて AWS リソースを共有できるようにするサービスです AWS Organizations。 AWS RAMでは、リソースを共有するリソースとコンシューマを指定するリソース 共有を作成して、リソースを共有します。コンシューマーは、個々の AWS アカウント、 の組織単 位 AWS Organizations、または の組織全体にすることができます AWS Organizations。

詳細については、AWS RAM ユーザーガイドをご参照ください。

共有されている CodeBuild プロジェクトにアクセス

共有プロジェクトにアクセスするには、コンシューマーの IAM ロールに BatchGetProjects アク セス許可が必要です。次のポリシーを IAM ロールに付けることができます。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetProjects"
    ]
}
```

詳細については、「<u>でのアイデンティティベースのポリシーの使用 AWS CodeBuild</u>」を参照してく ださい。

共有プロジェクトを共有解除

ビルドを含む共有されていないプロジェクトには、その所有者のみがアクセスできます。プロジェクトの共有を解除すると、以前に共有した AWS アカウントまたはユーザーは、プロジェクトまたはそのビルドにアクセスできなくなります。

自己所有の共有プロジェクトを共有解除するには、それをリソース共有から削除する必要がありま す。これを行う AWS CLI には、 AWS CodeBuild コンソール、 AWS RAM コンソール、または を使 用できます。

所有している共有プロジェクトの共有を解除するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の更新」を参照してください。

所有する共有プロジェクトの共有を解除するには (AWS CLI)

disassociate-resource-share コマンドを使用します。

所有するプロジェクトの共有を解除する (CodeBuild コマンド)

delete-resource-policy コマンドを実行し、共有を解除するプロジェクトの ARN を指定します。

aws codebuild delete-resource-policy --resource-arn project-arn

共有プロジェクトを識別

所有者とコンシューマーは、 AWS CLI を使用して共有プロジェクトを識別できます。

AWS アカウントまたはユーザーと共有されているプロジェクトを特定するには (AWS CLI)

list-shared-projects コマンドを使用して、自分に共有されているプロジェクトを返します。

共有プロジェクトへのアクセス許可

所有者のアクセス許可

プロジェクトの所有者は、プロジェクトを編集し、それを使用してビルドを実行できます。

コンシューマーのアクセス許可

プロジェクトコンシューマーは、プロジェクトとそのビルドを表示できますが、プロジェクトを編集 したり、プロジェクトを使用してビルドを実行することはできません。

ビルドプロジェクトをタグ付け

タグは、 AWS リソース AWS に割り当てるカスタム属性ラベルです。各 AWS タグには 2 つの部分 があります。

- タグキー (CostCenter、Environment、Project、Secret など)。タグキーでは、大文字と小 文字が区別されます。
- タグ値と呼ばれるオプションのフィールド (111122223333、Production、チーム名など)。タ グ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値では 大文字と小文字が区別されます。

これらは共にキーと値のペアと呼ばれます。プロジェクトに付けることができるタグの最大数、およ びタグのキーと値の制限については、「[タグ]」を参照してください。 タグは、AWS リソースの識別と整理に役立ちます。多くの AWS サービスはタグ付けをサポートし ているため、異なる サービスのリソースに同じタグを割り当てることで、リソースが関連している ことを示すことができます。たとえば、S3 バケットに割り当てたものと同じタグを CodeBuild プロ ジェクトに割り当てることができます。タグの使用の詳細については、「<u>タグ付けのベストプラク</u> ティス」を参照してください。

CodeBuild では、主なリソースはプロジェクトとレポートグループです。CodeBuild コンソール、 AWS CLI、CodeBuild APIs、または AWS SDKs を使用して、プロジェクトのタグを追加、管理、削 除できます。タグを使用して、プロジェクトを識別、整理、追跡するだけでなく、IAM ポリシーで タグを使用して、プロジェクトを表示および操作できるユーザーを管理することもできます。タグ ベースのアクセスポリシーの例については、「<u>タグを使用した AWS CodeBuild リソースへのアクセ</u> スのコントロール」を参照してください。

A Important

リザーブドキャパシティ機能を使用すると、ソースファイル、Docker レイヤー、buildspec で指定されキャッシュされたディレクトリなどを含む、フリートインスタンスにキャッシュ されたデータに、同じアカウント内の他のプロジェクトからアクセスできます。これは設計 によるもので、同じアカウント内のプロジェクトがフリートインスタンスを共有できるよう にしています。

トピック

- プロジェクトにタグを追加する
- プロジェクトのタグを表示する
- プロジェクトのタグを編集する
- プロジェクトからタグを削除する

プロジェクトにタグを追加する

プロジェクトにタグを追加すると、 AWS リソースを識別して整理し、リソースへのアクセスを管理 するのに役立ちます。まず、プロジェクトに 1 つ以上のタグ (キーと値のペア) を追加します。プロ ジェクトに付けることができるタグの数には制限があります。キーフィールドおよび値フィールドに 使用できる文字には制限があります。詳細については、「[タグ]」を参照してください。タグを追加 した後、IAM ポリシーを作成して、それらのタグに基づいてプロジェクトへのアクセスを管理でき ます。CodeBuild コンソールまたは AWS CLI を使用して、プロジェクトにタグを追加できます。

▲ Important

リザーブドキャパシティ機能を使用すると、ソースファイル、Docker レイヤー、buildspec で指定されキャッシュされたディレクトリなどを含む、フリートインスタンスにキャッシュ されたデータに、同じアカウント内の他のプロジェクトからアクセスできます。これは設計 によるもので、同じアカウント内のプロジェクトがフリートインスタンスを共有できるよう にしています。

プロジェクトの作成時にタグを追加する方法の詳細については、「<u>プロジェクトにタグを追加する</u> (コンソール)」を参照してください。

A Important

プロジェクトにタグを追加する前に、タグを使用してビルドプロジェクトなどのリソースへ のアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグ ベースのアクセスポリシーの例については、「<u>タグを使用した AWS CodeBuild リソースへ</u> <u>のアクセスのコントロール</u>」を参照してください。

トピック

- プロジェクトにタグを追加する (コンソール)
- ・プロジェクトにタグを追加する (AWS CLI)

プロジェクトにタグを追加する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに1つ以上のタグを追加できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Build projects (ビルドプロジェクト)] で、タグを追加するプロジェクトの名前を選択します。
- 3. ナビゲーションペインで [設定] を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
- プロジェクトにいずれのタグも追加されていない場合は、[Add tag (タグの追加)] を選択しま す。それ以外の場合は、[Edit]、[Add tag] の順に選択します。
- 5. [Key] に、タグの名前を入力します。[値] では、任意でタグに値を追加できます。
- 6. (オプション)別のタグを追加するには、[Add tag]を再度選択します。

7. タグの追加を完了したら、[Submit] を選択します。

プロジェクトにタグを追加する (AWS CLI)

プロジェクトの作成時にタグを追加するには、「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」を参照し てください。create-project.json で、タグを追加します。

以下のステップでは、 AWS CLI の最新版を既にインストールしているか、最新版に更新しているも のと想定します。詳細については、「<u>AWS Command Line Interfaceのインストール</u>」を参照してく ださい。

成功すると、このコマンドは何も返しません。

プロジェクトのタグを表示する

タグは、 AWS リソースを識別して整理し、リソースへのアクセスを管理するのに役立ちます。タグ の使用の詳細については、「<u>タグ付けのベストプラクティス</u>」ホワイトペーパーを参照してくださ い。タグベースのアクセスポリシーの例については、「<u>タグを使用した AWS CodeBuild リソースへ</u> <u>のアクセスのコントロール</u>」を参照してください。

プロジェクトのタグを表示する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに関連付けられたタグを表示できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Build projects (ビルドプロジェクトの)] で、タグを表示するプロジェクトの名前を選択します。
- 3. ナビゲーションペインで [設定] を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。

プロジェクトのタグを表示する (AWS CLI)

ビルドプロジェクトのタグを表示するには、以下のコマンドを実行します。--names パラメータに はプロジェクトの名前を使用します。

aws codebuild batch-get-projects --names your-project-name

成功すると、このコマンドは、ビルドプロジェクトに関する以下のような JSON 形式の情報を返し ます。

プロジェクトのタグを表示する

```
{
    "tags": {
        "Status": "Secret",
        "Team": "JanesProject"
    }
}
```

プロジェクトにいずれのタグも追加されていない場合、tags セクションは空になります。

"tags": []

プロジェクトのタグを編集する

プロジェクトに関連付けられたタグの値を変更できます。キーの名前を変更することもできます。こ れは、現在のタグを削除して、新しい名前と他のタグと同じ値を持つ、別のタグを追加することにな ります。キーフィールドと値フィールドに使用できる文字には制限があることにご注意ください。詳 細については、「[タグ]」を参照してください。

▲ Important

プロジェクトのタグを編集すると、そのプロジェクトへのアクセスに影響を与える可能性が あります。プロジェクトのタグの名前 (キー) または値を編集する前に、タグのキーや値を 使用してビルドプロジェクトなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、 「<u>タグを使用した AWS CodeBuild リソースへのアクセスのコントロール</u>」を参照してくだ さい。

プロジェクトのタグを編集する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに関連付けられたタグを表示できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Build projects (ビルドプロジェクト)] で、タグを編集するプロジェクトの名前を選択します。
- 3. ナビゲーションペインで [設定] を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
- 4. [編集]を選択します。
- 5. 次のいずれかを行ってください。

- タグを変更するには、[Key] に新しい名前を入力します。タグの名前を変更することは、タグ を削除して、新しいキー名を持つタグを追加することになります。
- タグの値を変更するには、新しい値を入力します。値を空にする場合は、現在の値を削除して フィールドを空のままにします。
- 6. タグの編集を完了したら、[Submit] を選択します。

プロジェクトのタグを編集する (AWS CLI)

ビルドプロジェクトのタグを追加、変更、または削除するには、「<u>ビルドプロジェクトの設定の変更</u> (AWS CLI)」を参照してください。プロジェクトの更新に使用する JSON 形式のデータの tags セク ションを更新します。

プロジェクトからタグを削除する

プロジェクトに関連付けられた1つ以上のタグを削除できます。タグを削除しても、そのタグに関 連付けられている他の AWS リソースからタグは削除されません。

▲ Important

プロジェクトからタグを削除すると、そのプロジェクトへのアクセスに影響を与える可能性 があります。プロジェクトからタグを削除する前に、タグのキーや値を使用してビルドプロ ジェクトなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必 ず確認してください。タグベースのアクセスポリシーの例については、「<u>タグを使用した</u> AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

プロジェクトからタグを削除する (コンソール)

CodeBuild コンソールを使用して、タグと CodeBuild プロジェクトとの関連付けを解除できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Build projects (ビルドプロジェクト)] で、タグを削除するプロジェクトの名前を選択します。
- 3. ナビゲーションペインで [設定] を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
- 4. [Edit]を選択します。
- 5. 削除するタグを見つけ、[Remove tag] を選択します。

6. タグの削除を完了したら、[Submit] を選択します。

プロジェクトからタグを削除する (AWS CLI)

ビルドプロジェクトから 1 つ以上のタグを削除するには、「<u>ビルドプロジェクトの設定の変更 (AWS</u> <u>CLI)</u>」を参照してください。JSON 形式のデータの tags セクションを、削除するタグが含まれてい ない最新のタグのリストで更新します。すべてのタグを削除する場合は、tags セクションを以下の ように更新します。

"tags: []"

Note

CodeBuild ビルドプロジェクトを削除すると、削除されたビルドプロジェクトからすべての タグの関連付けが解除されます。ビルドプロジェクトを削除する前にタグを削除する必要は ありません。

でランナーを使用する AWS CodeBuild

AWS CodeBuild は、GitHub Actions ランナー、セルフマネージド GitLab ランナー、および Buildkite ランナーとの統合をサポートしています。

トピック

- <u>のセルフホスト型 GitHub Actions ランナー AWS CodeBuild</u>
- のセルフマネージド GitLab ランナー AWS CodeBuild
- のセルフマネージド Buildkite ランナー AWS CodeBuild

のセルフホスト型 GitHub Actions ランナー AWS CodeBuild

CodeBuild コンテナにセルフホスト型 GitHub Actions ランナーを設定して、GitHub Actions ワーク フロージョブを処理するようにプロジェクトを構成できます。これは、CodeBuild プロジェクトを使 用してウェブフックを設定し、CodeBuild マシンでホストされているセルフホスト型ランナーを使用 するように GitHub Actions ワークフロー YAML を更新することによって実行できます。

GitHub Actions ジョブを実行するように CodeBuild プロジェクトを設定する大まかな手順は次のとおりです。

- まだ行っていない場合は、個人用アクセストークンを作成するか、OAuth アプリに接続してプロジェクトを GitHub に接続します。
- CodeBuild コンソールに移動し、ウェブフックを使用して CodeBuild プロジェクトを作成し、 ウェブフックフィルタを設定します。
- 3. GitHub の GitHub Actions ワークフロー YAML を更新して、ビルド環境を設定します。

より詳細な手順については、「<u>チュートリアル: CodeBuild がホストする GitHub Actions ランナーを</u> <u>設定</u>」を参照してください。

この機能を使用すると、GitHub Actions ワークフロージョブを とネイティブに統合できます。これ により AWS、IAM、統合、 AWS Secrets Manager Amazon VPC などの機能を通じてセキュリティ AWS CloudTrailと利便性が提供されます。ARM ベースのインスタンスなど、最新のインスタンスタ イプにアクセスできます。

トピック

- CodeBuild がホストする GitHub Actions ランナーについて
- チュートリアル: CodeBuild がホストする GitHub Actions ランナーを設定
- ウェブフックのトラブルシューティング
- CodeBuild がホストする GitHub Actions ランナーでサポートされているラベルの上書き
- <u>CodeBuild がホストする GitHub Actions ランナーでサポートされているコンピューティングイメージ</u>

CodeBuild がホストする GitHub Actions ランナーについて

以下は、CodeBuild がホストする GitHub Actions ランナーに関する、よくある質問です。

ラベルにイメージとインスタンスの上書きを含める必要があるのはいつですか。

イメージとインスタンスの上書きをラベルに含めることで、GitHub Actions ワークフロージョブごと に異なるビルド環境を指定できます。これは、複数の CodeBuild プロジェクトやウェブフックを作 成しなくても実行できます。例えば、<u>ワークフロージョブにマトリックス</u>を使用する必要がある場合 に便利です。

name: Hello World
on: [push]
jobs:
 Hello-World-Job:

```
runs-on:
        - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        - image:${{ matrix.os }}
        - instance-size:${{ matrix.size }}
strategy:
        matrix:
        include:
        - os: arm-3.0
        size: small
        - os: linux-5.0
        size: large
steps:
        - run: echo "Hello World!"
```

Note

runs-on に GitHub Actions コンテキストを含む複数のラベルがある場合、引用符が必要に なる場合があります。

この機能 AWS CloudFormation に を使用できますか?

はい。プロジェクトのウェブフックで GitHub Actions ワークフロージョブイベントフィルターを指 定するフィルターグループを AWS CloudFormation テンプレートに含めることができます。

Triggers: Webhook: true FilterGroups: - - Type: EVENT Pattern: WORKFLOW_JOB_QUEUED

詳細については、「<u>GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation)</u>」を参照 してください。

AWS CloudFormation テンプレートでプロジェクト認証情報の設定に関するヘルプが必要な場合は、 「AWS CloudFormation ユーザーガイド」の<u>AWS::CodeBuild::SourceCredential</u>」を参照してくださ い。 この機能を使用する際にシークレットをマスクするにはどうすればよいですか。

デフォルトでは、ログに出力されるシークレットはマスクされません。シークレットをマスクする場 合は、次の構文を使用できます。::add-mask::*value*。次に、YAML でこの構文を使用する方法 の例を示します。

```
name: Secret Job
on: [push]
jobs:
   Secret-Job:
   runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
   env:
      SECRET_NAME: "secret-name"
   steps:
      - run: echo "::add-mask::$SECRET_NAME"
```

詳細については、GitHub の「Masking a value in a log」を参照してください。

単一プロジェクト内の複数のリポジトリから GitHub Actions ウェブフックイベントを受信すること はできますか。

CodeBuild は、指定された組織またはエンタープライズからイベントを受信する、組織レベルおよび グローバルレベルのウェブフックをサポートします。詳細については、「<u>GitHub グローバルおよび</u> 組織のウェブフック」を参照してください。

CodeBuild がホストする GitHub Actions ランナーの使用をサポートしているリージョンはどれですか。

CodeBuild がホストする GitHub Actions ランナーは、すべての CodeBuild リージョンでサポートさ れています。CodeBuild が利用可能な AWS リージョン 場所の詳細については、<u>AWS 「リージョン</u> 別のサービス」を参照してください。

CodeBuild がホストする GitHub Actions ランナーの使用をサポートしているプラットフォームはど れですか。

CodeBuild がホストする GitHub Actions ランナーは、Amazon EC2 と <u>AWS Lambda</u> コンピューティ ングの両方でサポートされています。Amazon Linux 2、Amazon Linux 2023、Ubuntu、Windows Server Core 2019 のプラットフォームを使用できます。詳細については、「<u>EC2 コンピューティン</u> <u>グイメージ</u>」および「<u>Lambda コンピューティングイメージ</u>」を参照してください。

チュートリアル: CodeBuild がホストする GitHub Actions ランナーを設定

このチュートリアルでは、CodeBuild プロジェクトを設定して GitHub Actions ジョブを実行する方 法について説明します。CodeBuild で GitHub Actions を使用する方法の詳細については、「<u>チュー</u> トリアル: CodeBuild がホストする GitHub Actions ランナーを設定」を参照してください。

このチュートリアルを完了するには、まず以下を行う必要があります。

- 個人用アクセストークン、Secrets Manager シークレット、OAuth アプリ、または GitHub アプリ に接続します。OAuth アプリを使用して接続する場合は、CodeBuild コンソールを使用して接続 する必要があります。個人用アクセストークンを作成する場合は、CodeBuild コンソールを使用 するか、<u>ImportSourceCredentials API</u>を使用できます。詳細な手順については、「<u>CodeBuild の</u> GitHub および GitHub Enterprise Server アクセス」を参照してください。
- CodeBuild を GitHub アカウントに接続します。これを行うには、次のいずれかで実行できます。
 - コンソールで GitHub をソースプロバイダーとして追加できます。個人用アクセストークン、Secrets Manager シークレット、OAuth アプリ、または GitHub アプリのいずれかを使用して接続できます。手順については、「CodeBuild の GitHub および GitHub Enterprise Server アクセス」を参照してください。
 - GitHub 認証情報は、<u>ImportSourceCredentials API</u> 経由でインポートできます。これは、個人用 アクセストークンでのみ実行できます。OAuth アプリを使用して接続する場合は、代わりにコ ンソールを使用して接続する必要があります。手順については、「<u>GitHub をアクセストークン</u> で接続する(CLI)」を参照してください。

Note

これを行う必要があるのは、アカウントで GitHub に接続していない場合のみです。

ステップ 1: ウェブフックを使用して CodeBuild プロジェクトを作成

このステップでは、ウェブフックを使用して CodeBuild プロジェクトを作成し、GitHub コンソー ルで確認します。ソースプロバイダとして GitHub Enterprise を選択することもできます。GitHub Enterprise 内でウェブフックを作成する方法の詳細については、「<u>GitHub 手動ウェブフック</u>」を参 照してください。 ウェブフックを使用して CodeBuild プロジェクトを作成するには

- 1. <u>https://console.aws.amazon.com/codesuite/codebuild/home</u> で AWS CodeBuild コンソールを開 きます。
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
- 3. プロジェクトタイプで、Runner プロジェクトを選択します。

Runner の場合:

- a. Runner プロバイダーで、GitHub を選択します。
- b. Runner の場所で、リポジトリを選択します。
- c. リポジトリの下にあるリポジトリ URL で、https://github.com/user-name/repository-name を選択します。

Note

デフォルトでは、プロジェクトは単一リポジトリの WORKFLOW_JOB_QUEUED イベント のみを受信します。組織またはエンタープライズ内のすべてのリポジトリのイベントを 受信する場合は、「<u>GitHub グローバルおよび組織のウェブフック</u>」を参照してくださ い。

- 4. [環境] で以下の操作を行います。
 - サポートされている [環境イメージ] と [コンピューティング] を選択します。GitHub Actions ワークフロー YAML のラベルを使用して、イメージとインスタンスの設定を上書きする オプションがあることに注意してください。詳細については、ステップ 2: GitHub Actions ワークフロー YAML を更新を参照してください。
 - [Buildspec (Buildspec)] で、次のようにします。
 - buildspec-override:true がラベルとして追加されない限り、buildspec は無視される ことに注意してください。代わりに、CodeBuild は、セルフホスト型ランナーを設定するコ マンドを使用するように上書きします。
- 5. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。
- https://github.com/user-name/repository-name/settings/hooks で GitHub コン ソールを開き、ウェブフックが作成され、[ワークフロージョブ] イベントの配信が有効になって いることを確認します。

ステップ 2: GitHub Actions ワークフロー YAML を更新

このステップでは、<u>GitHub</u> で GitHub Actions ワークフロー YAML ファイルを更新してビルド環境 を設定し、CodeBuild で GitHub Actions セルフホスト型ランナーを使用します。詳細については、 「<u>Using labels with self-hosted runners</u>」および「<u>CodeBuild がホストする GitHub Actions ランナー</u> <u>でサポートされているラベルの上書き</u>」を参照してください。

GitHub Actions ワークフロー YAML を更新

<u>GitHub</u> に移動し、GitHub Actions ワークフロー YAML の <u>runs-on</u> の設定を更新して、ビルド環境 を設定します。これを行うには、次のいずれかで実行できます。

 プロジェクト名と実行 ID を指定できます。その場合、ビルドはコンピューティング、イメージ、イメージバージョン、インスタンスサイズに既存のプロジェクト設定を使用します。GitHub Actions ジョブの AWS関連の設定を特定の CodeBuild プロジェクトにリンクするには、プロジェクト名が必要です。YAML にプロジェクト名を含めることで、CodeBuild は正しいプロジェクト設定でジョブを呼び出すことができます。実行 ID を指定することで、CodeBuild はビルドを特定のワークフロー実行にマッピングし、ワークフロー実行がキャンセルされたときにビルドを停止します。詳細については、「github context」を参照してください。

runs-on: codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}

Note

<project-name> が、前のステップで作成したプロジェクトの名前と一致していること を確認してください。一致しない場合、CodeBuild はウェブフックを処理せず、GitHub Actions ワークフローがハングする可能性があります。

次に、GitHub Actions ワークフロー YAML の例を示します。

```
name: Hello World
on: [push]
jobs:
    Hello-World-Job:
    runs-on:
        - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    steps:
        - run: echo "Hello World!"
```

 ラベル内のイメージとコンピューティングタイプを上書きすることもできます。厳選されたイメージのリストCodeBuild がホストする GitHub Actions ランナーでサポートされているコンピュー <u>ティングイメージ</u>については、「」を参照してください。カスタムイメージの使用については、「」を参照してくださいCodeBuild がホストする GitHub Actions ランナーでサポートされている <u>ラベルの上書き</u>。ラベル内のコンピューティングタイプとイメージは、プロジェクトの環境設定を 上書きします。CodeBuild EC2 または Lambda コンピューティングビルドの環境設定を上書きす るには、次の構文を使用します。

runs-on:

- codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}
- image:<environment-type>-<image-identifier>
- instance-size:<instance-size>

次に、GitHub Actions ワークフロー YAML の例を示します。

```
name: Hello World
on: [push]
jobs:
Hello-World-Job:
runs-on:
    codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    image:arm-3.0
    instance-size:small
    steps:
    run: echo "Hello World!"
```

 ラベル内のビルドに使用するフリートを上書きできます。これにより、プロジェクトで設定された フリート設定が上書きされ、指定されたフリートが使用されます。詳細については、「<u>リザーブ</u> <u>ドキャパシティキャパシティフリートでビルドを実行</u>」を参照してください。Amazon EC2 コン ピューティングビルドのフリート設定を上書きするには、次の構文を使用します。

runs-on:

- codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}
- fleet:<fleet-name>

ビルドに使用されるフリートとイメージの両方を上書きするには、次の構文を使用します。

runs-on:

- codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}
- fleet:<fleet-name>

次に、GitHub Actions ワークフロー YAML の例を示します。

```
name: Hello World
on: [push]
jobs:
    Hello-World-Job:
    runs-on:
        - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        - fleet:myFleet
        - image:arm-3.0
        steps:
        - run: echo "Hello World!"
```

- カスタムイメージで GitHub Actions ジョブを実行するには、CodeBuild プロジェクトでカスタム イメージを設定し、イメージ上書きラベルを指定しないようにします。CodeBuild は、イメージ上 書きラベルが指定されていない場合、プロジェクトで設定されたイメージを使用します。
- 必要に応じて、CodeBuild がサポートするラベル以外のラベルを提供できます。これらのラベル は、ビルドの属性を上書きする目的で無視されますが、ウェブフックリクエストは失敗しません。
 例えば、testLabel をラベルとして追加しても、ビルドの実行は妨げられません。
 - Note

GitHub がホストするランナーが提供する依存関係が CodeBuild 環境で利用できない場合 は、ワークフロー実行時に GitHub アクションを使用して依存関係をインストールできま す。例えば、<u>setup-python</u> アクションを使用して、ビルド環境に Python をインストール できます。

INSTALL、PRE_BUILD、POST_BUILD フェーズで buildspec コマンドを実行

デフォルトでは、CodeBuild はセルフホスト型 GitHub Actions ビルドを実行するときに buildspec コ マンドを無視します。ビルド中に buildspec コマンドを実行するには、ラベルにサフィックスとして buildspec-override:true を追加できます。

runs-on:

- codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}
- buildspec-override:true

このコマンドを使用すると、CodeBuild はコンテナのプライマリソースフォルダに actionsrunner というフォルダを作成します。GitHub Actions ランナーが BUILD フェーズ中に起動する と、ランナーはその actions-runner ディレクトリで実行されます。

セルフホスト型 GitHub Actions ビルドで buildspec の上書きを使用する場合、いくつかの制限があり ます。

- CodeBuild は、セルフホスト型ランナーが BUILD フェーズで実行されるため、BUILD フェーズ中 は buildspec コマンドを実行しません。
- CodeBuild は、DOWNLOAD_SOURCE フェーズ中はプライマリソースもセカンダリソースもダウン ロードしません。buildspec ファイルが設定されている場合、プロジェクトのプライマリソースか らそのファイルのみがダウンロードされます。
- ビルドコマンドが PRE_BUILD または INSTALL フェーズで失敗した場合、CodeBuild はセルフホ スト型ランナーを起動せず、GitHub Actions ワークフロージョブは手動でキャンセルする必要があ ります。
- CodeBuild は、DOWNLOAD_SOURCE フェーズ中にランナートークンを取得します。有効期限は1時間です。PRE_BUILD または INSTALL フェーズが1時間を超えると、GitHub セルフホスト型ランナーが起動する前にランナートークンの有効期限が切れる可能性があります。

ステップ 3: 結果を確認

GitHub Actions ワークフローが実行されるたびに、CodeBuild はウェブフックを介してワークフロー ジョブイベントを受信します。ワークフロー内のジョブごとに、CodeBuild はビルドを開始して一時 的な GitHub Actions ランナーを実行します。ランナーには、単一のワークフロージョブを実行する 役割があります。ジョブが完了すると、ランナーおよび関連付けられたビルドプロセスは即座に終了 します。

ワークフロージョブログを表示するには、GitHub のリポジトリに移動し、[アクション] を選択し て、目的のワークフローを選択し、ログを確認する特定の [ジョブ] を選択します。

ジョブが CodeBuild のセルフホスト型ランナーによって取得されるのを待っている間に、リクエス トされたラベルをログで確認できます。



ジョブが完了すると、ジョブのログを表示できるようになります。

Hello-World-Job succeeded now in 4s	Beta) Give feedback Q Search logs	ŝ
> 🥥 Set up job		
✓ ✓ Run echo "Hello World!"		0s
1 ▶ Run echo "Hello World!" 4 Hello World!		
> 🤡 Complete job		

GitHub Actions ランナー設定オプション

プロジェクト設定で次の環境変数を指定して、セルフホスト型ランナーのセットアップ設定を変更で きます。

CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME

CodeBuild は、この環境変数の値として指定された組織名にセルフホスト型ランナーを登録しま す。ランナーを組織レベルで登録する方法と必要なアクセス許可の詳細については、「組織の just-in-timeランナーの設定を作成する」を参照してください。

CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME

CodeBuild は、この環境変数の値として指定されたエンタープライズ名にセルフホストランナー を登録します。ランナーをエンタープライズレベルで登録する方法と必要なアクセス許可の詳細 については、<u>「Create configuration for a just-in-time runner for an Enterprise</u>」を参照してくださ い。

Note

Enterprise Runner は、デフォルトでは組織リポジトリでは使用できません。セルフホス ト型ランナーがワークフロージョブを取得するには、ランナーグループのアクセス設定を 構成する必要がある場合があります。詳細については、「エンタープライズランナーをリ <u>ポジトリで使用できるようにする</u>」を参照してください。

CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID

CodeBuild は、この環境変数の値として保存されている整数ランナーグループ ID にセルフホスト ランナーを登録します。デフォルトでは、この値は 1 です。セルフホスト型ランナーグループの 詳細については、<u>「グループを使用したセルフホスト型ランナーへのアクセスの管理</u>」を参照し てください。

GitHub Actions ウェブフックイベントをフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートの次の YAML 形式の部分は、true と評価されたときにビル ドをトリガーするフィルタグループを作成します。次のフィルタグループは、正規表現 \[CI-CodeBuild\] に一致するワークフロー名を持つ GitHub Actions ワークフロージョブリクエストを 指定します。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
      FilterGroups:
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

GitHub Actions ウェブフックイベントをフィルタリング (AWS CDK)

次の AWS CDK テンプレートは、ビルドが true と評価されたときにビルドをトリガーするフィル ターグループを作成します。次のフィルタグループは、GitHub Actions ワークフロージョブリクエス トを指定します。

```
import { aws_codebuild as codebuild } from 'aws-cdk-lib';
import {EventAction, FilterGroup} from "aws-cdk-lib/aws-codebuild";
const source = codebuild.Source.gitHub({
    owner: 'owner',
    repo: 'repo',
    webhook: true,
    webhookFilters: [FilterGroup.inEventOf(EventAction.WORKFLOW_JOB_QUEUED)],
  })
```

GitHub Actions ウェブフックイベントをフィルタリング (Terraform)

次の Terraform テンプレートは、true と評価されたときにビルドをトリガーするフィルタグループを 作成します。次のフィルタグループは、GitHub Actions ワークフロージョブリクエストを指定しま す。

```
resource "aws_codebuild_webhook" "example" {
    project_name = aws_codebuild_project.example.name
    build_type = "BUILD"
    filter_group {
        filter {
            type = "EVENT"
            pattern = "WORKFLOW_JOB_QUEUED"
        }
    }
}
```

ウェブフックのトラブルシューティング

問題: <u>チュートリアル: CodeBuild がホストする GitHub Actions ランナーを設定</u> で設定したウェブ フックが機能していないか、ワークフロージョブが GitHub でハングしています。

考えられる原因:

- ウェブフックワークフロージョブイベントがビルドのトリガーに失敗している可能性があります。[レスポンス] ログを確認して、レスポンスまたはエラーメッセージを表示します。
- ラベル設定のため、ジョブが誤ったランナーエージェントに割り当てられています。この問題は、1つのワークフロー実行内のいずれかのジョブのラベルが別のジョブよりも少ない場合に発生する可能性があります。たとえば、同じワークフロー実行に次のラベルを持つ2つのジョブがある場合です。

- ジョブ 1: codebuild-myProject-\${{ github.run_id }}-\${{ github.run_attempt }}
- ジョブ 2: codebuild-myProject-\${{ github.run_id }}-\${{ github.run_attempt }}、instance-size:medium

セルフホスト型 GitHub Actions ジョブをルーティングする場合、GitHub はジョブのすべての指定 されたラベルを持つ任意のランナーにジョブをルーティングします。この動作により、ジョブ 1 はジョブ 1 またはジョブ 2 用に作成されたランナーによって取得できますが、ジョブ 2 には追加 のラベルがあるため、ジョブ 2 用に作成されたランナーによってのみ取得できます。ジョブ 1 が ジョブ 2 用に作成されたランナーによって取得された場合、ジョブ 1 ランナーに ラベルがないた め、ジョブ 2 はスタックします。 instance-size:medium

推奨される解決策:

同じワークフロー実行内で複数のジョブを作成する場合は、各ジョブに同じ数のラベルオーバーライ ドを使用するか、または job1 や などのカスタムラベルを各ジョブに割り当てますjob2。

エラーが解決しない場合は、次の手順を使用して問題をデバッグします。

- https://github.com/user-name/repository-name/settings/hooks で GitHub コン ソールを開き、リポジトリのウェブフック設定を表示します。このページには、リポジトリ用に 作成されたウェブフックが表示されます。
- 2. [編集] を選択し、ウェブフックの [ワークフロージョブ] イベントの配信が有効になっていること を確認します。



- [最近の配信] タブに移動し、対応する workflow_job.queued イベントを見つけて、イベント を展開します。
- 4. [ペイロード]の[ラベル]フィールドを確認し、期待どおりに動作していることを確認します。
- 5. 最後に、[レスポンス] タブを確認します。このタブには、CodeBuild から返されたレスポンスまたはエラーメッセージが含まれています。

Settings	Recent Deliveries					
	13478-efat-11ee-8787	M248/13-584	workflow_job.queued			1014-03-01 W 10-01
Request	Response 400			Redeliver	Ŀ	Completed in seconds.
Headers						

6. または、GitHub の API を使用してウェブフックの障害をデバッグすることもできます。「<u>List</u> <u>deliveries for a repository webhook</u>」API を使用して、ウェブフックの最近の配信を表示できま す。

gh api ∖ -H "Accept: application/vnd.github+json" \ -H "X-GitHub-Api-Version: 2022-11-28" \ /repos/owner/repo/hooks/hook-id/deliveries

デバッグするウェブフック配信を見つけて配信 ID をメモしたら、<u>Get a delivery for a repository</u> webhook API を使用できます。ウェブフックの配信ペイロードに対する CodeBuild のレスポン スは、response セクションにあります。

gh api \
 -H "Accept: application/vnd.github+json" \
 -H "X-GitHub-Api-Version: 2022-11-28" \
 /repos/owner/repo/hooks/hook-id/deliveries/delivery-id

CodeBuild がホストする GitHub Actions ランナーでサポートされているラベルの上書 き

GitHub Actions ワークフロー YAML では、セルフホスト型ランナーのビルドを変更するさまざまな ラベルの上書きを指定できます。CodeBuild で認識されないビルドは無視されますが、ウェブフック リクエストは失敗しません。たとえば、次のワークフロー YAML には、イメージ、インスタンスサ イズ、フリート、および buildspec のオーバーライドが含まれます。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:${{ matrix.os }}
      - instance-size:${{ matrix.size }}
      - fleet:myFleet
      - buildspec-override:true
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

Note

ワークフロージョブが GitHub でハングアップしている場合は、<u>ウェブフックのトラブル</u> <u>シューティング</u>「」および<u>「カスタムラベルを使用してジョブをルーティング</u>する」を参照 してください。

codebuild-<project-name>-\${{github.run_id}}-\${{github.run_attempt}}(必須)

- 例: codebuild-fake-project-\${{ github.run_id }}-\${{ github.run_attempt }}
- すべての GitHub Actions ワークフロー YAML に必須です。<project name> は、セルフホスト 型ランナーウェブフックが設定されているプロジェクトの名前と同じである必要があります。

image:<environment-type>-<image-identifier>

• 例:image:arm-3.0

- キュレートされたイメージでセルフホストランナービルドを開始するときに使用されるイメージと環境タイプを上書きします。サポートされている値については、「CodeBuild がホストする GitHub Actions ランナーでサポートされているコンピューティングイメージ」を参照してください。
 - カスタムイメージで使用されるイメージと環境タイプを上書きするには、を使用します。 image:custom-<environment-type>-<custom-image-identifier>
 - 例:image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64standard:3.0

Note

カスタムイメージがプライベートレジストリにある場合は、「」を参照してください<u>セ</u> ルフホスト型ランナーのプライベートレジストリ認証情報を設定する。

instance-size:<instance-size>

- 例:instance-size:medium
- セルフホスト型ランナーのビルドの開始時に使用するインスタンスタイプを上書きします。サポートされている値については、「CodeBuild がホストする GitHub Actions ランナーでサポートされているコンピューティングイメージ」を参照してください。

fleet:<fleet-name>

- 例:fleet:myFleet
- 指定されたフリートを使用するために、プロジェクトに設定されたフリート設定を上書きします。
 詳細については、「<u>リザーブドキャパシティキャパシティフリートでビルドを実行</u>」を参照してく ださい。

buildspec-override:<boolean>

- 例: buildspec-override:true
- ・ true に設定されている場合、ビルドが INSTALL、PRE_BUILD、および POST_BUILD フェーズ で buildspec コマンドを実行できるようにします。
単一ラベルの上書き (レガシー)

CodeBuild では、以下を使用して、単一のラベルに複数の上書きを指定できます。

• Amazon EC2/Lambda コンピューティングビルドの環境設定を上書きするには、次の構文を使用 します。

runs-on: codebuild-<project-name>-\${{ github.run_id }}\${{ github.run_attempt }}-<environment-type>-<image-identifier>-<instance-size>

Amazon EC2 コンピューティングビルドのフリート設定を上書きするには、次の構文を使用します。

runs-on: codebuild-<project-name>-\${{ github.run_id }}-\${{ github.run_attempt }}fleet-<fleet-name>

• ビルドに使用されるフリートとイメージの両方を上書きするには、次の構文を使用します。

runs-on: codebuild-<project-name>-\${{ github.run_id }}\${{ github.run_attempt }}-image-<image-version>-fleet-<fleet-name>

 ビルド中に buildspec コマンドを実行するには、ラベルにサフィックスとして -withbuildspec を追加できます。

runs-on: codebuild-<project-name>-\${{ github.run_id }}\${{ github.run_attempt }}-<image>-<image-version>-<instance-size>-with-buildspec

 オプションで、イメージを上書きせずにインスタンスサイズの上書きを指定できます。Amazon EC2 ビルドでは、環境タイプとイメージ識別子の両方を除外できます。Lambda ビルドでは、イ メージ識別子を除外できます。

CodeBuild がホストする GitHub Actions ランナーでサポートされているコンピュー ティングイメージ

「<u>チュートリアル: CodeBuild がホストする GitHub Actions ランナーを設定</u>」で設定したラベルで は、最初の 3 つの列の値を使用して Amazon EC2 環境設定を上書きできます。CodeBuild では、次 の Amazon EC2 コンピューティングイメージが用意されています。詳細については、以下を参照し てください。

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	解像イメージ	定義
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:4.0	<u>al/standa</u> <u>rd/4.0</u>
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:5.0	<u>al/standa</u> rd/5.0
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-a arch64-st andard:2. Ø	<u>al/aarch64/</u> <u>standard/2.0</u>
arm	3.0	2xlarge	Amazon Linux 2023	aws/codeb uild/amaz onlinux-a arch64-st andard:3. Ø	<u>al/aarch64/</u> <u>standard/3.0</u>
ubuntu	5.0	small medium large xlarge	Ubuntu 20.04	aws/codeb uild/stan dard:5.0	ubuntu/st andard/5.0

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	解像イメージ	定義
ubuntu	6.0	2xlarge gpu_small	Ubuntu 22.04	aws/codeb uild/stan dard:6.0	<u>ubuntu/st</u> andard/6.0
ubuntu	7.0	gpu_large	Ubuntu 22.04	aws/codeb uild/stan dard:7.0	ubuntu/st andard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-1.0	該当なし
			Windows Server Core 2022	aws/codeb uild/wind ows-base: 2022-1.0	該当なし
windows	2.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-2.0	該当なし
windows	3.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-3.0	該当なし

さらに、次の値を使用して Lambda 環境設定を上書きできます。CodeBuild Lambda コンピューティ ングの詳細については、「<u>AWS Lambda コンピューティングでビルドを実行する</u>」を参照してくだ さい。CodeBuild は、次の Lambda コンピューティングイメージをサポートしています。

環境タイプ	イメージ識別 子	インスタンス サイズ
linux-lam bda	dotnet6	1GB
arm-lambd	go1.21	2GB
a	corretto1	4GB
	1	8GB
	corretto1 7	10GB
	corretto2 1	
	nodejs18	
	nodejs20	
	python3.1 1	
	python3.1 2	
	ruby3.2	

詳細については、「<u>ビルド環境のコンピューティングモードおよびタイプ</u>」および「<u>CodeBuild に用</u> 意されている Docker イメージ」を参照してください。

のセルフマネージド GitLab ランナー AWS CodeBuild

GitLab には、CI/CD パイプラインで GitLab ジョブを実行するための 2 つの実行モードがありま す。1 つのモードは GitLab ホストランナーで、GitLab によって管理され、GitLab と完全に統合され ています。もう 1 つのモードはセルフマネージド型ランナーです。これにより、独自のカスタマイ ズされた環境を導入して、GitLab CI/CD パイプラインでジョブを実行できます。 GitLab CI/CD パイプラインジョブを実行するように CodeBuild プロジェクトを設定する大まかな手 順は次のとおりです。

- 1. まだ行っていない場合は、OAuth アプリを使用してプロジェクトを GitLab に接続します。
- CodeBuild コンソールに移動し、ウェブフックを使用して CodeBuild プロジェクトを作成し、 ウェブフックフィルタを設定します。
- 3. GitLab で GitLab CI/CD パイプライン YAML を更新して、ビルド環境を設定します。

より詳細な手順については、「<u>チュートリアル: CodeBuild がホストする GitLab ランナーを設定</u>」を 参照してください。

この機能を使用すると、GitLab CI/CD パイプラインジョブを AWSとネイティブ統合できます。これ により、IAM、 AWS CloudTrail、Amazon VPC などの機能を通じてセキュリティと利便性が提供さ れます。ARM ベースのインスタンスなど、最新のインスタンスタイプにアクセスできます。

トピック

- CodeBuild がホストする GitLab ランナーについて
- ・ チュートリアル: CodeBuild がホストする GitLab ランナーを設定
- CodeBuild がホストする GitLab ランナーでサポートされているラベルの上書き
- CodeBuild がホストする GitLab ランナーでサポートされているコンピューティングイメージ

CodeBuild がホストする GitLab ランナーについて

以下は、CodeBuild がホストする GitLab ランナーに関する、よくある質問です。

CodeBuild がホストする GitLab ランナーでは、どのようなソースタイプがサポートされていますか?

CodeBuild がホストする GitLab ランナーは、 GITLABおよび GITLAB_SELF_MANAGEDソースタイプ でサポートされています。

ラベルにイメージとインスタンスの上書きを含める必要があるのはいつですか。

イメージとインスタンスの上書きをラベルに含めることで、GitLab CI/CD パイプラインジョブごと に異なるビルド環境を指定できます。これは、複数の CodeBuild プロジェクトやウェブフックを作 成しなくても実行できます。

この機能 AWS CloudFormation に を使用できますか?

はい。プロジェクトウェブフックで GitLab ワークフロージョブイベントフィルターを指定するフィ ルターグループを AWS CloudFormation テンプレートに含めることができます。

Triggers: Webhook: true FilterGroups: - - Type: EVENT Pattern: WORKFLOW_JOB_QUEUED

詳細については、「<u>GitLab ウェブフックイベントのフィルタリング (AWS CloudFormation)</u>」を参照 してください。

AWS CloudFormation テンプレートでのプロジェクト認証情報の設定に関するヘルプが必要な場合 は、AWS CloudFormation 「ユーザーガイド」の<u>AWS::CodeBuild::SourceCredential</u>」を参照してく ださい。

この機能を使用する際にシークレットをマスクするにはどうすればよいですか。

デフォルトでは、ログに出力されるシークレットはマスクされません。シークレットをマスクする場合は、CI/CD 環境変数設定を更新してマスクできます。

GitLab でシークレットをマスクするには

1. [GitLab 設定] で [CI/CD] を選択します。

2. [変数] で、マスクするシークレットの [編集] を選択します。

3. [可視性] で、[マスク変数] を選択し、[変数を更新] を選択して変更を保存します。

単一グループ内の複数のプロジェクトから GitLab ウェブフックイベントを受信することはできます か。

CodeBuild は、指定された GitLab グループからイベントを受信するグループウェブフックをサポートしています。詳細については、「GitLab グループウェブフック」を参照してください。

セルフマネージド型ランナーの Docker Executor でジョブを実行することはできますか。例えば、特 定のイメージでパイプラインジョブを実行して、分離された別のコンテナに同じビルド環境を維持し ます。

CodeBuild で GitLab セルフマネージド型ランナーを特定のイメージで実行するには、<u>カスタムイ</u> <u>メージを使用してプロジェクトを作成する</u>か、.gitlab-ci.yml ファイル内の<u>イメージを上書き</u>し ます。

CodeBuild のセルフマネージド型ランナーはどのエグゼキュターで実行されますか。

CodeBuild のセルフマネージド型ランナーはシェルエグゼキュターで実行され、ビルドは Docker コ ンテナ内で実行されている GitLab ランナーとともにローカルで実行されます。

セルフマネージド型ランナーと一緒に buildspec コマンドを提供できますか。

はい。セルフマネージド型ランナーと一緒に buildspec コマンドを追加できます。GitLab リポジトリ に buildspec.yml ファイルを指定し、ジョブの buildspec-override:true タグセクションで [タ グ] を使用できます。詳細については、「<u>buildspec ファイル名とストレージの場所</u>」を参照してく ださい。

CodeBuild がホストする GitLab ランナーの使用をサポートしているリージョンはどれですか。

CodeBuild がホストする GitLab ランナーは、すべての CodeBuild リージョンでサポートされていま す。CodeBuild が利用可能な AWS リージョン 場所の詳細については、<u>AWS 「リージョン別のサー</u> ビス」を参照してください。

CodeBuild がホストする GitLab ランナーの使用をサポートしているプラットフォームはどれですか。

CodeBuild がホストする GitLab ランナーは、Amazon EC2 と <u>AWS Lambda</u> コンピューティングの 両方でサポートされています。Amazon Linux 2、Amazon Linux 2023、Ubuntu、Windows Server Core 2019 のプラットフォームを使用できます。詳細については、「<u>EC2 コンピューティングイ</u> メージ」および「Lambda コンピューティングイメージ」を参照してください。

チュートリアル: CodeBuild がホストする GitLab ランナーを設定

このチュートリアルでは、GitLab CI/CD パイプラインジョブを実行するように CodeBuild プロジェ クトを設定する方法について説明します。CodeBuild で GitLab または GitLab セルフマネージドを使 用する方法の詳細については、「<u>のセルフマネージド GitLab ランナー AWS CodeBuild</u>」を参照して ください。 このチュートリアルを完了するには、まず以下を行う必要があります。

- CodeConnections を使用して OAuth アプリに接続します。OAuth アプリに接続する場合 は、CodeBuild コンソールを使用して接続する必要があることに注意してください。詳細な手順に ついては、「CodeBuild での GitLab アクセス」を参照してください。
- CodeBuild を GitLab アカウントに接続します。これを行うには、コンソールで GitLab をソースプ ロバイダとして追加できます。手順については、「<u>CodeBuild での GitLab アクセス</u>」を参照して ください。

Note

これを行う必要があるのは、アカウントで GitLab に接続していない場合のみです。 この機能では、CodeBuild に GitLab OAuth アプリからの create_runner や manage_runner などの追加のアクセス許可が必要です。特定の GitLab アカウントに既 存の CodeConnections がある場合、アクセス許可の更新は自動的にリクエストされませ ん。これを行うには、CodeConnections コンソールに移動し、同じ GitLab アカウントへ のダミー接続を作成して、追加のアクセス許可を取得するための再認証をトリガーしま す。これにより、すべての既存の接続でランナー機能を使用できます。完了したら、ダ ミー接続を削除できます。

ステップ 1: ウェブフックを使用して CodeBuild プロジェクトを作成

このステップでは、ウェブフックを使用して CodeBuild プロジェクトを作成し、GitLab コンソール で確認します。

ウェブフックを使用して CodeBuild プロジェクトを作成するには

- 1. <u>https://console.aws.amazon.com/codesuite/codebuild/home</u> で AWS CodeBuild コンソールを開 きます。
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。

プロジェクトタイプで、Runner プロジェクトを選択します。

- Runner の場合:
 - Runner プロバイダーで、GitLab を選択します。
 - ・ [認証情報] で、次のいずれかを選択します。

- [デフォルトソース認証情報] を選択します。デフォルト接続は、すべてのプロジェクトに デフォルトの GitLab 接続を適用します。
- [カスタムソース認証情報] を選択します。カスタム接続は、アカウントのデフォルト設定 を上書きするカスタム GitLab 接続を適用します。

Note プロバイダへの接続をまだ作成していない場合は、新しい GitLab 接続を作成する必 要があります。手順については、「<u>CodeBuild を GitLab に接続</u>」を参照してください。

- Runner の場所で、リポジトリを選択します。
- [リポジトリ] で、プロジェクトのパスと名前空間を指定して、GitLab 内のプロジェクトの 名前を選択します。
- [環境] で以下の操作を行います。
 - サポートされている [環境イメージ] と [コンピューティング] を選択します。GitLab CI/CD パイプライン YAML のラベルを使用して、イメージとインスタンスの設定を上書きする オプションがあることに注意してください。詳細については、「<u>ステップ 2: リポジトリ</u> に .gitlab-ci.yml ファイルを作成」を参照してください。
- [Buildspec (Buildspec)] で、次のようにします。
 - buildspec-override:true がラベルとして追加されない限り、buildspec は無視される ことに注意してください。代わりに、CodeBuild は、セルフマネージド型ランナーを設定す るコマンドを使用するように上書きします。
- 3. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。
- https://gitlab.com/user-name/repository-name/-/hooks で GitLab コンソールを開き、ウェブフックが作成され、[ワークフロージョブ] イベントの配信が有効になっていることを確認します。

ステップ 2: リポジトリに .gitlab-ci.yml ファイルを作成

このステップでは、<u>GitLab</u> で .gitlab-ci.yml ファイルを作成してビルド環境を設定 し、CodeBuild で GitLab セルフマネージド型ランナーを使用します。詳細については、「<u>Use self-</u> managed runners」を参照してください。 GitLab CI/CD パイプライン YAML を更新

「https://gitlab.com/*user-name/project-name*/-/tree/*branch-name*」に移動して、リ ポジトリで .gitlab-ci.yml ファイルを作成します。ビルド環境を設定するには、次のいずれかを 実行します。

 CodeBuild プロジェクト名を指定できます。その場合、ビルドはコンピューティング、イメージ、 イメージバージョン、インスタンスサイズに既存のプロジェクト設定を使用します。GitLab ジョ ブの AWS関連の設定を特定の CodeBuild プロジェクトにリンクするには、プロジェクト名が必要 です。YAML にプロジェクト名を含めることで、CodeBuild は正しいプロジェクト設定でジョブを 呼び出すことができます。

tags:

- codebuild-<codebuild-project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME

\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME は、ビルドを特定のパイプラインジョ ブの実行にマッピングし、パイプラインの実行がキャンセルされたときにビルドを停止するために 必要です。

Note

<project-name> が、CodeBuild で作成したプロジェクトの名前と一致していることを 確認してください。一致しない場合、CodeBuild はウェブフックを処理せず、GitLab Cl/ CD パイプラインがハングする可能性があります。

GitLab CI/CD パイプライン YAML の例を次に示します。

タグ内のイメージとコンピューティングタイプを上書きすることもできます。厳選されたイメージのリストCodeBuild がホストする GitLab ランナーでサポートされているコンピューティングイメージについては、「」を参照してください。カスタムイメージの使用については、「」を参照してくださいCodeBuild がホストする GitLab ランナーでサポートされているラベルの上書き。タグのコンピューティングタイプとイメージは、プロジェクトの環境設定を上書きします。Amazon EC2 コンピューティングビルドの環境設定を上書きするには、次の構文を使用します。

tags:

- codebuild-<codebuild-project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- image:<environment-type>-<image-identifier>
- instance-size:<instance-size>

GitLab CI/CD パイプライン YAML の例を次に示します。

```
stages:
    - build
build-job:
    stage: build
    script:
    - echo "Hello World!"
    tags:
        - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
        - image:arm-3.0
        - instance-size:small
```

 タグ内のビルドに使用するフリートを上書きできます。これにより、プロジェクトで設定された フリート設定が上書きされ、指定されたフリートが使用されます。詳細については、「<u>リザーブ</u> <u>ドキャパシティキャパシティフリートでビルドを実行</u>」を参照してください。Amazon EC2 コン ピューティングビルドのフリート設定を上書きするには、次の構文を使用します。

tags:

- codebuild-<codebuild-project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- fleet:<fleet-name>

ビルドに使用されるフリートとイメージの両方を上書きするには、次の構文を使用します。

tags:

- codebuild-<codebuild-project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- fleet:<fleet-name>

- image:<environment-type>-<image-identifier>

GitLab CI/CD パイプライン YAML の例を次に示します。

```
stages:
  - build
build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - fleet:myFleet
    - image:arm-3.0
```

 カスタムイメージで GitLab CI/CD パイプラインジョブを実行するには、CodeBuild プロジェクト でカスタムイメージを設定し、イメージ上書きラベルを指定しないようにします。CodeBuild は、 イメージ上書きラベルが指定されていない場合、プロジェクトで設定されたイメージを使用しま す。

.gitlab-ci.yml に変更をコミットすると、GitLab パイプラインがトリガーされ、build-job か らウェブフック通知が送信され、CodeBuild でのビルドが開始されます。

INSTALL、PRE_BUILD、POST_BUILD フェーズで buildspec コマンドを実行

デフォルトでは、CodeBuild はセルフマネージド型 GitLab ビルドを実行するときに buildspec コマンドを無視します。ビルド中に buildspec コマンドを実行するには、サフィックスとして buildspec-override:true を tags に追加できます。

tags:

- codebuild-<codebuild-project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- buildspec-override:true

このコマンドを使用すると、CodeBuild はコンテナのプライマリソースフォルダに gitlab-runner というフォルダを作成します。GitLab ランナーが BUILD フェーズ中に起動すると、ランナーはその gitlab-runner ディレクトリで実行されます。

セルフマネージド型 GitLab ビルドで buildspec の上書きを使用する場合、いくつかの制限がありま す。

- CodeBuild は、セルフマネージド型ランナーが BUILD フェーズで実行されるため、BUILD フェー ズ中は buildspec コマンドを実行しません。
- CodeBuild は、DOWNLOAD_SOURCE フェーズ中はプライマリソースもセカンダリソースもダウン ロードしません。buildspec ファイルが設定されている場合、プロジェクトのプライマリソースか らそのファイルのみがダウンロードされます。
- ビルドコマンドが PRE_BUILD または INSTALL フェーズで失敗した場合、CodeBuild はセルフマ ネージド型ランナーを起動せず、GitLab CI/CD パイプラインジョブは手動でキャンセルする必要 があります。
- CodeBuild は、DOWNLOAD_SOURCE フェーズ中にランナートークンを取得します。有効期限は 1 時間です。PRE_BUILD または INSTALL フェーズが 1 時間を超えると、GitLab セルフマネージド 型ランナーが起動する前にランナートークンの有効期限が切れる可能性があります。

ステップ 3: 結果を確認

GitLab CI/CD パイプラインの実行が発生するたびに、CodeBuild はウェブフックを介して CI/CD パ イプラインジョブイベントを受信します。CI/CD パイプライン内のジョブごとに、CodeBuild はビ ルドを開始して一時的な GitLab ランナーを実行します。ランナーには、単一の CI/CD パイプライン ジョブを実行する役割があります。ジョブが完了すると、ランナーおよび関連付けられたビルドプロ セスは即座に終了します。

CI/CD パイプラインジョブのログを表示するには、GitLab のリポジトリに移動し、[ビルド]、[ジョ ブ] の順に選択し、ログを確認する特定の [ジョブ] を選択します。

ジョブが CodeBuild のセルフマネージド型ランナーによって取得されるのを待っている間に、リク エストされたラベルをログで確認できます。

GitLab ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートの次の YAML 形式の部分は、true と評価されたときにビル ドをトリガーするフィルタグループを作成します。次のフィルタグループは、正規表現 \[CI-CodeBuild\] に一致する CI/CD パイプライン名を持つ GitLab CI/CD パイプラインジョブリクエス トを指定します。

```
CodeBuildProject:
Type: AWS::CodeBuild::Project
Properties:
Name: MyProject
ServiceRole: service-role
```

CodeBuild がホストする GitLab ランナーでサポートされているラベルの上書き

GitLab CI/CD パイプライン YAML では、セルフマネージド型ランナーのビルドを変更するさまざま なラベルの上書きを指定できます。CodeBuild で認識されないビルドは無視されますが、ウェブフッ クリクエストは失敗しません。例えば、次の YAML には、イメージ、インスタンスサイズ、フリー ト、および buildspec の上書きが含まれます。

```
workflow:
    name: HelloWorld
stages:
    - build
build-job:
    stage: build
    script:
        - echo "Hello World!"
    tags:
        - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
        - image:arm-3.0
        - instance-size:small
        - fleet:myFleet
        - buildspec-override:true
```

codebuild-<project-name>-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME(必須)

- 例: codebuild-myProject-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- すべての GitLab CI/CD パイプライン YAML に必須です。<project name> は、セルフマネージ ド型ランナーウェブフックが設定されているプロジェクトの名前と同じである必要があります。

image:<environment-type>-<image-identifier>

- 例:image:arm-3.0
- セルフマネージド型ランナーのビルドの開始時に使用するイメージと環境タイプを上書きします。 サポートされている値については、「<u>CodeBuild がホストする GitLab ランナーでサポートされて</u>いるコンピューティングイメージ」を参照してください。
 - カスタムイメージで使用されるイメージと環境タイプを上書きするには、を使用します。 image:custom-<environment-type>-<custom-image-identifier>
 - 例:image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64standard:3.0

Note

カスタムイメージがプライベートレジストリにある場合は、「」を参照してください<u>セ</u> ルフホスト型ランナーのプライベートレジストリ認証情報を設定<u>する</u>。

instance-size:<instance-size>

- 例: instance-size: small
- セルフマネージド型ランナーのビルドの開始時に使用するインスタンスタイプを上書きします。サポートされている値については、「CodeBuild がホストする GitLab ランナーでサポートされているコンピューティングイメージ」を参照してください。

fleet:<fleet-name>

- 例:fleet:myFleet
- 指定されたフリートを使用するために、プロジェクトに設定されたフリート設定を上書きします。
 詳細については、「<u>リザーブドキャパシティキャパシティフリートでビルドを実行</u>」を参照してく ださい。

GitLab ランナー

buildspec-override:<boolean>

- 例: buildspec-override:true
- true に設定されている場合、ビルドが INSTALL、PRE_BUILD、および POST_BUILD フェーズ で buildspec コマンドを実行できるようにします。

CodeBuild がホストする GitLab ランナーでサポートされているコンピューティングイ メージ

「<u>チュートリアル: CodeBuild がホストする GitLab ランナーを設定</u>」で設定したラベルでは、最初の 3 つの列の値を使用して Amazon EC2 環境設定を上書きできます。CodeBuild では、次の Amazon EC2 コンピューティングイメージが用意されています。詳細については、以下を参照してくださ い。

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	イメージ	定義
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:4.0	<u>al/standa</u> <u>rd/4.0</u>
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:5.0	<u>al/standa</u> rd/5.0
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-a arch64-st andard:2. Ø	<u>al/aarch64/</u> standard/2.0

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	イメージ	定義
arm	3.0	2xlarge	Amazon Linux 2023	<pre>aws/codeb uild/amaz onlinux-a arch64-st andard:3. 0</pre>	al/aarch64/ standard/3.0
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codeb uild/stan dard:5.0	ubuntu/st andard/5.0
ubuntu	6.0	large xlarge	Ubuntu 22.04	aws/codeb uild/stan dard:6.0	<u>ubuntu/st</u> andard/6.0
ubuntu	7.0	gpu_small gpu_large	Ubuntu 22.04	aws/codeb uild/stan dard:7.0	<u>ubuntu/st</u> andard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-1.0	該当なし
			Windows Server Core 2022	aws/codeb uild/wind ows-base: 2022-1.0	該当なし
windows	2.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-2.0	該当なし

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	イメージ	定義
windows	3.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-3.0	該当なし

さらに、次の値を使用して Lambda 環境設定を上書きできます。CodeBuild Lambda コンピューティ ングの詳細については、「<u>AWS Lambda コンピューティングでビルドを実行する</u>」を参照してくだ さい。CodeBuild は、次の Lambda コンピューティングイメージをサポートしています。

環境タイプ	ランタイム バージョン	インスタンス サイズ	
linux-lam bda	dotnet6	1GB	
	go1.21	2GB	
arm-lambd a	corretto1	4GB	
	1	8GB	
	corretto1 7	10GB	
	corretto2		
	T		
	nodejs18		
	nodejs20		
	python3.1 1		
	python3.1 2		

環境タイプ	ランタイム バージョン	インスタンス サイズ		
	ruby3.2			

詳細については、「<u>ビルド環境のコンピューティングモードおよびタイプ</u>」および「<u>CodeBuild に用</u> 意されている Docker イメージ」を参照してください。

のセルフマネージド Buildkite ランナー AWS CodeBuild

プロジェクトを設定して、CodeBuild コンテナにセルフホスト Buildkite ランナーを設定して Buildkite ジョブを処理できます。これは、CodeBuild プロジェクトを使用してウェブフックを設定 し、CodeBuild マシンでホストされているセルフホストランナーを使用するように Buildkite パイプ ライン YAML ステップを更新することで実行できます。

Buildkite ジョブを実行するように CodeBuild プロジェクトを設定する大まかな手順は次のとおりです。

- CodeBuild コンソールに移動し、Buildkite ランナープロジェクトランナータイプ設定を使用して CodeBuild プロジェクトを作成します。
- Buildkite 組織にjob.scheduledウェブフックを追加します。
- Buildkite パイプラインの YAML ステップを更新して、ビルド環境を設定します。

より詳細な手順については、「<u>チュートリアル: CodeBuild がホストする Buildkite ランナーを設</u> <u>定する</u>」を参照してください。この機能を使用すると、Buildkite ジョブは とネイティブに統合さ れ、IAM AWS、、AWS Secrets Manager、Amazon VPC などの機能を通じてセキュリティ AWS CloudTrailと利便性を提供します。ARM ベースのインスタンスなど、最新のインスタンスタイプに アクセスできます。

CodeBuild がホストする Buildkite ランナーについて

以下は、CodeBuild がホストする Buildkite ランナーに関する一般的な質問です。

ラベルにイメージとインスタンスの上書きを含める必要があるのはいつですか。

Buildkite ジョブごとに異なるビルド環境を指定するために、イメージとインスタンスのオーバーラ イドをラベルに含めることができます。これは、複数の CodeBuild プロジェクトやウェブフックを 作成しなくても実行できます。例えば、<u>Buildkite ジョブにマトリックス</u>を使用する必要がある場合 に便利です。

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

CodeBuild は Buildkite 内にウェブフックを自動的に作成できますか?

現在、Buildkite では、すべてのウェブフックをコンソールを使用して手動で作成する必要がありま す。のチュートリアルに従って<u>チュートリアル: CodeBuild がホストする Buildkite ランナーを設定す</u> る、Buildkite コンソールで Buildkite ウェブフックを手動で作成できます。

AWS CloudFormation を使用して Buildkite ウェブフックを作成できますか?

AWS CloudFormation Buildkite ではコンソールを使用してウェブフックを手動で作成する必要がある ため、 は現在 Buildkite ランナーウェブフックではサポートされていません。

CodeBuild がホストする Buildkite ランナーの使用をサポートしているのはどのリージョンですか?

CodeBuild がホストする Buildkite ランナーは、すべての CodeBuild リージョンでサポートされてい ます。CodeBuild が利用可能な AWS リージョンの詳細については、<u>AWS「リージョン別のサービ</u> ス」を参照してください。

チュートリアル: CodeBuild がホストする Buildkite ランナーを設定する

このチュートリアルでは、Buildkite ジョブを実行するように CodeBuild プロジェクトを設定する 方法を示します。CodeBuild で Buildkite を使用する方法の詳細については、「」を参照してくださ いのセルフマネージド Buildkite ランナー AWS CodeBuild。 このチュートリアルを完了するには、まず以下を行う必要があります。

- Buildkite 組織にアクセスできます。Buildkite アカウントと組織の設定の詳細については、この入 門チュートリアルを参照してください。
- セルフホスト型ランナーを使用するように設定された Buildkite パイプライン、クラスター、 キューを作成します。これらのリソースの設定の詳細については、<u>「Buildkite Pipeline Setup</u> Tutorial」を参照してください。

M	AWS CodeBuild 🗸		Pipelines	Agents	Test Suites	Settings				My Builds	Help
		ŵ	myPipe	line				Q	Settings	New Build	
			\odot			_					
			\odot			-		_	-		
						Run your first build Your pipeline is ready! Run a build by selecting New After running a build, the history displays here New Build	v Build e.	I.			
						Arun a build via the API					
						泰Configure GitHub webhooks	→				

ステップ 1: Buildkite エージェントトークンを生成する

このステップでは、CodeBuild セルフホストランナーの認証に使用されるエージェントトークンを Buildkite 内で生成します。このリソースの詳細については、<u>「Buildkite エージェントトークン</u>」を 参照してください。

Buildkite エージェントトークンを生成するには

- 1. Buildkite クラスターで、エージェントトークンを選択し、新しいトークンを選択します。
- 2. トークンに説明を追加し、トークンの作成をクリックします。
- エージェントトークン値は、後で CodeBuild プロジェクトのセットアップ中に使用されるため、保存します。

Agent Tokens > New
Description — Required
myToken
Describe the set of agents this token is for
Allowed IP Addresses
0.0.0/0 ::/0
Restrict which network addresses are allowed to use this agent token. Use space-separated, IPv4 CIDR notation, for example: 192.0.2.0/24 198.51.100.12 25c7:c056:9943:1e01::/64.
Create Token

ステップ 2: ウェブフックを使用して CodeBuild プロジェクトを作成する

ウェブフックを使用して CodeBuild プロジェクトを作成するには

- 1. <u>https://console.aws.amazon.com/codesuite/codebuild/home</u> で AWS CodeBuild コンソールを開 きます。
- 2. セルフホストビルドプロジェクトを作成します。詳細については、「<u>ビルドプロジェクトの作成</u> (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
 - ・プロジェクト設定で、Runner プロジェクトを選択します。Runner の場合:
 - Runner プロバイダーで、Buildkite を選択します。
 - Buildkite エージェントトークンで、シークレットの作成ページを使用して新しいエージェントトークンを作成するを選択します。で、上記で生成した Buildkite エージェントトークンと等しいシークレット値 AWS Secrets Manager を持つ新しいシークレットを作成するように求められます。
 - (オプション)ジョブに CodeBuild マネージド認証情報を使用する場合は、Buildkite ソース 認証情報オプションでジョブのソースリポジトリプロバイダーを選択し、アカウントに認証 情報が設定されていることを確認します。さらに、Buildkite パイプラインが HTTPS を使用 したチェックアウトを使用していることを確認します。

Note

Buildkite では、ジョブのソースをプルするために、ビルド環境内でソース認証情 報が必要です。使用可能なソース認証情報オプション<u>プライベートリポジトリへの</u> Buildkite の認証については、「」を参照してください。

- ・ (オプション)環境の場合:
 - ・ サポートされている [環境イメージ] と [コンピューティング] を選択します。

Buildkite YAML ステップでラベルを使用してイメージとインスタンスの設定を上書きする オプションがあることに注意してください。詳細については、「<u>ステップ 4: Buildkite パイ</u> プラインステップを更新する」を参照してください。

- (オプション) Buildspec で:
 - がラベルとして追加されない限り、buildspec buildspec-override: "true" はデフォ ルトで無視されます。代わりに、CodeBuild はセルフホストランナーをセットアップするコ マンドを使用するように上書きします。

Note

CodeBuild は、Buildkite セルフホストランナービルドの buildspec ファイルをサ ポートしていません。インライン buildspecs の場合、CodeBuild マネージドソース 認証情報を設定している場合は、buildspec で<u>git-credential-helper</u> を有効にする必 要があります。

- 3. デフォルト値のまま続行し、[ビルドプロジェクトを作成する]を選択します。
- Create Webhook ポップアップからペイロード URL とシークレット値を保存します。ポップ アップの指示に従って新しい Buildkite 組織のウェブフックを作成するか、次のセクションに進 みます。

ステップ 3: Buildkite 内で CodeBuild ウェブフックを作成する

このステップでは、CodeBuild ウェブフックのペイロード URL とシークレット値を使用し て、Buildkite 内に新しいウェブフックを作成します。このウェブフックは、有効な Buildkite ジョブ の開始時に CodeBuild 内でビルドをトリガーするために使用されます。 Buildkite で新しいウェブフックを作成するには

- 1. Buildkite 組織の設定ページに移動します。
- 2. 統合で、通知サービスを選択します。
- 3. Webhook ボックスの横にある追加 を選択します。ウェブフック通知の追加ページで、次の設定 を使用します。
 - a. Webhook URL で、保存されたペイロード URL 値を追加します。
 - b. トークン で、X-Buildkite-Token としてトークンを送信する が選択されていることを確認し ます。ウェブフックシークレット値をトークンフィールドに追加します。
 - c. で、X-Buildkite-Token としてトークンを送信するが選択されていることを確認します。 ウェブフックのシークレット値をトークンフィールドに追加します。
 - d. イベント で、job.scheduledウェブフックイベントを選択します。
 - e. (オプション) Pipelines では、オプションで特定のパイプラインのビルドのみをトリガーす るように選択できます。
- 4. Webhook 通知の追加を選択します。

ステップ 4: Buildkite パイプラインステップを更新する

このステップでは、Buildkite パイプラインのステップを更新して、必要なラベルとオプションの オーバーライドを追加します。サポートされているラベルオーバーライドの完全なリストについて は、「」を参照してください<u>CodeBuild がホストする Buildkite ランナーでサポートされているラベ</u> ルオーバーライド。

パイプラインステップを更新する

Buildkite パイプラインを選択し、設定を選択し、ステップを選択して、Buildkite パイプラインステップページに移動します。

まだの場合は、YAML ステップに変換を選択します。

ŷ	myPipeline	Steps	
තු	General	1 steps:	
\bigcirc	Steps	2 - command: ""	
ð	Builds		
0	GitHub		
╚	Schedules 0		
\oslash	Build Badges		
Ŋ	Personal Email Settings		
		Save Steps Save and Build	Show Guide

 少なくとも、CodeBuild パイプラインの名前を参照する <u>Buildkite エージェントタグ</u>を指定する 必要があります。Buildkite ジョブの AWS関連設定を特定の CodeBuild プロジェクトにリンクす るには、プロジェクト名が必要です。YAML にプロジェクト名を含めることで、CodeBuild は正 しいプロジェクト設定でジョブを呼び出すことができます。

agents: project: "codebuild-<project name>"

以下は、プロジェクトラベルタグのみを含む Buildkite パイプラインステップの例です。

```
agents:
    project: "codebuild-myProject"
    steps:
    - command: "echo \"Hello World\""
```

ラベル内のイメージとコンピューティングタイプを上書きすることもできます。使用可能なイ メージのリストについては、「<u>CodeBuild がホストする Buildkite ランナーでサポートされてい</u> <u>るイメージをコンピューティングする</u>」を参照してください。ラベル内のコンピューティングタ イプとイメージは、プロジェクトの環境設定を上書きします。CodeBuild EC2 または Lambda コンピューティングビルドの環境設定を上書きするには、次の構文を使用します。

```
agents:
    project: "codebuild-<project name>"
    image: "<environment-type>-<image-identifier>"
    instance-size: "<instance-size>"
```

イメージサイズとインスタンスサイズの上書きを含む Buildkite パイプラインステップの例を次 に示します。

```
agents:
    project: "codebuild-myProject"
    image: "arm-3.0"
    instance-size: "small"
steps:
    - command: "echo \"Hello World\""
```

ラベル内のビルドに使用するフリートを上書きできます。これにより、プロジェクトで設定され たフリート設定が上書きされ、指定されたフリートが使用されます。詳細については、<u>「リザー</u> ブドキャパシティフリートでビルドを実行する」を参照してください。

Amazon EC2 コンピューティングビルドのフリート設定を上書きするには、次の構文を使用し ます。

```
agents:
    project: "codebuild-<project name>"
    fleet: "<fleet-name>"
```

ビルドに使用されるフリートとイメージの両方を上書きするには、次の構文を使用します。

```
agents:
    project: "codebuild-<project name>"
    fleet: "<fleet-name>"
    image: "<environment-type>-<image-identifier>"
```

フリートとイメージの上書きを含む Buildkite パイプラインステップの例を次に示します。

```
agents:
    project: "codebuild-myProject"
    fleet: "myFleet"
    image: "arm-3.0"
```

steps:

- command: "echo \"Hello World\""

 セルフホスト Buildkite ランナービルド中にインライン buildspec コマンドを実行することを選 択できます (詳細については<u>INSTALL、PRE_BUILD、POST_BUILDの各フェーズで buildspec</u> <u>コマンドを実行する</u>、「」を参照してください)。Buildkite セルフホスト型ランナービルド中 に CodeBuild ビルドが buildspec コマンドを実行するように指定するには、次の構文を使用しま す。

```
agents:
    project: "codebuild-<project name>"
    buildspec-override: "true"
```

buildspec オーバーライドを使用した Buildkite パイプラインの例を次に示します。

```
agents:
   project: "codebuild-myProject"
   buildspec-override: "true"
   steps:
   - command: "echo \"Hello World\""
```

 必要に応じて、CodeBuild がサポートするラベル以外のラベルを提供できます。これらのラベル は、ビルドの属性を上書きする目的で無視されますが、ウェブフックリクエストは失敗しませ ん。例えば、myLabel: "testLabel" をラベルとして追加しても、ビルドの実行は妨げられ ません。

ステップ 5: 結果を確認する

Buildkite ジョブがパイプラインで開始されるたびに、CodeBuild は Buildkite job.scheduledウェ ブフックを介してウェブフックイベントを受け取ります。Buildkite ビルド内のジョブごと に、CodeBuild はエフェメラル Buildkite ランナーを実行するビルドを開始します。ランナーは、単

に、CodeBuild はエフェメフル Buildkite フクリーを実行するヒルドを開始します。フクリーは、単 ーの Buildkite ジョブを実行する責任があります。ジョブが完了すると、ランナーおよび関連付けら れたビルドプロセスは即座に終了します。

ワークフロージョブログを表示するには、Buildkite パイプラインに移動し、最新のビルドを選択 します (新しいビルドを選択して新しいビルドをトリガーできます)。各ジョブに関連付けられた CodeBuild ビルドが開始されてジョブが取得されると、Buildkite コンソールにジョブのログが表示さ れます。

Jobs Canvas New Waterfall Upgrade	
All Failures	
echo "Hello World" echo "Hello World"	Waited 10s · Ran in 2s 🛛 🕸 Agent
Log Artifacts o Timeline Environment	
+ Expand groups 🗕 Collapse groups 🗎 Show timestamps	Ø Theme t Delete L Download C Open ↓ Jump to end
 Preparing working directory 46 ~ Running commands 47 \$ echo "Hello World" 48 Hello World 	15 (05)
Exited with status 0	♠ Back to Job

プライベートリポジトリへの Buildkite の認証

Buildkite パイプライン内にプライベートリポジトリが設定されている場合、Buildkite はプライベー トリポジトリからプルするためにセルフホスト型ランナーに認証情報を供給しないため、Buildkite にはリポジトリをプルするための<u>ビルド環境内の追加のアクセス許可</u>が必要です。Buildkite セルフ ホスト型ランナーエージェントを外部プライベートソースリポジトリに対して認証するには、次のい ずれかのオプションを使用できます。

CodeBuild で認証するには

CodeBuild は、サポートされているソースタイプのマネージド認証情報処理を提供しま す。CodeBuild ソース認証情報を使用してジョブのソースリポジトリをプルするには、次の手順を使 用します。

- CodeBuild コンソールで、プロジェクトの編集に移動するか、「」の手順を使用して新しい CodeBuild プロジェクトを作成します<u>ステップ 2: ウェブフックを使用して CodeBuild プロジェ</u> クトを作成する。
- Buildkite ソース認証情報オプションで、ジョブのソースリポジトリプロバイダーを選択します。
 - 1. アカウントレベルの CodeBuild 認証情報を使用する場合は、それらが正しく設定されてい ることを確認します。さらに、プロジェクトにインライン buildspec が設定されている場合 は、git-credential-helper が有効になっていることを確認します。
 - 2. プロジェクトレベルの CodeBuild 認証情報を使用する場合は、このプロジェクトの上書き認 証情報のみを使用するを選択し、プロジェクトの認証情報を設定します。
- Buildkite パイプライン設定で、リポジトリ設定に移動します。ソースリポジトリのチェックアウト設定を HTTPS を使用してチェックアウトに設定する

Repository Settings		
Repository — Required		
No description	JavaScript	×
Checkout using: O SSH HTTPS The repository your agents will use t	https://github.com/ o checkout your code. Need to choose another repository or l	JRL?
Save Repository		

Buildkite シークレットで認証するには

Buildkite は、<u>ssh キーを使用して外部ソースリポジトリに対してセルフホスト型ランナーを認証する</u> ために使用できる ssh-checkout プラグイン を維持します。キー値は <u>Buildkite シークレット</u>として保 存され、プライベートリポジトリをプルしようとすると Buildkite セルフホスト型ランナーエージェ ントによって自動的に取得されます。Buildkite パイプラインの ssh-checkout プラグインを設定する には、次の手順を使用します。

- 1. Eメールアドレスを使用してプライベートおよびパブリック SSH キーを生成します。例: sshkeygen -t rsa -b 4096 -C "myEmail@address.com"
- パブリックキーをプライベートソースリポジトリに追加します。たとえば、<u>このガイド</u>に従って GitHub アカウントにキーを追加できます。
- Buildkite クラスターに<u>新しい SSH キーシークレット</u>を追加します。Buildkite クラスター内 で、シークレット → 新しいシークレットを選択します。Key フィールドにシークレットの名前 を追加し、Value フィールドにプライベート SSH キーを追加します。

New Secret		
Key — Required		
SOURCE_SSH_KEY		
Keys are case insensitive, can only contain alphanumeric and underscore characters, and can't start with BUILDKITE or BK		
Value — Required		
BEGIN <u>OPENSSH</u> PRIVATE KEY		

 Buildkite パイプライン内で、リポジトリ設定に移動し、SSH を使用するようにチェックアウト を設定します。

Repository Settings		
Repository — Required		
No description		×
Checkout using: SSH O HTTPS	git@github.com o checkout your code. Need to choose another repository or URL?	
Save Repository		

5. パイプラインの YAML ステップを更新して、 git-ssh-checkoutプラグインを使用します。 たとえば、次のパイプライン YAML ファイルでは、上記の Buildkite シークレットキーを使用し てチェックアウトアクションを使用します。

```
agents:

project: "codebuild-myProject"

steps:

- command: "npm run build"

plugins:

- git-ssh-checkout#v0.4.1:
```

 CodeBuild 内で Buildkite セルフホストランナージョブを実行すると、プライベートリポジトリ をプルするときに Buildkite が設定されたシークレット値を自動的に使用するようになりました。

Runner 設定オプション

プロジェクト設定で次の環境変数を指定して、セルフホスト型ランナーのセットアップ設定を変更で きます。

- CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN: CodeBuild は、Buildkite セルフホスト型ラン ナーエージェントを登録するために、この環境変数の値として設定されたシークレット値をから AWS Secrets Manager 取得します。この環境変数はタイプ でSECRETS_MANAGER、値は Secrets Manager のシークレットの名前である必要があります。Buildkite エージェントトークン環境変数 は、すべての Buildkite ランナープロジェクトに必要です。
- CODEBUILD_CONFIG_BUILDKITE_CREDENTIAL_DISABLE: デフォルトでは、CodeBuild はアカ ウントまたはプロジェクトレベルのソース認証情報をビルド環境にロードします。これらの認証情 報は Buildkite エージェントによってジョブのソースリポジトリをプルするために使用されます。 この動作を無効にするには、 値を に設定してこの環境変数をプロジェクトに追加します。これに よりtrue、ソース認証情報がビルド環境にロードされなくなります。

INSTALL、PRE_BUILD、POST_BUILD の各フェーズで buildspec コマンドを実行する

デフォルトでは、CodeBuild はセルフホスト Buildkite ランナービルドを実行するときに buildspec コ マンドを無視します。ビルド中に buildspec コマンドを実行するには

buildspec-override: "true"

は、ラベルのサフィックスとして追加できます。

```
agents:
    project: "codebuild-<project name>"
    buildspec-override: "true"
```

このコマンドを使用すると、CodeBuild はコンテナのプライマリソースフォルダに buildkiterunner というフォルダを作成します。Buildkite ランナーが BUILDフェーズ中に起動すると、ラン ナーは buildkite-runner ディレクトリで実行されます。

セルフホスト Buildkite ビルドで buildspec オーバーライドを使用する場合、いくつかの制限があり ます。

 Buildkite エージェントでは、ジョブのソースリポジトリをプルするために、ソース認証情報が ビルド環境内に存在する必要があります。認証に CodeBuild ソース認証情報を使用する場合 は、buildspec git-credential-helperで を有効にする必要があります。たとえば、次の buildspec を使用して Buildkite ビルドgit-credential-helperで を有効にできます。

```
version: 0.2
env:
  git-credential-helper: yes
phases:
  pre_build:
     commands:
        - echo "Hello World"
```

- CodeBuild は、セルフホスト型ランナーが BUILD フェーズで実行されるため、BUILD フェーズ中 は buildspec コマンドを実行しません。
- CodeBuild は、Buildkite ランナービルドの buildspec ファイルをサポートしていません。Buildlkite セルフホスト型ランナーではインライン buildspec のみがサポートされています
- PRE_BUILD または INSTALLフェーズでビルドコマンドが失敗した場合、CodeBuild はセルフホ スト型ランナーを起動せず、Buildkite ジョブを手動でキャンセルする必要があります。

Buildkite ランナーをプログラムでセットアップする

Buildkite ランナープロジェクトをプログラムで設定するには、次のリソースを設定する必要があり ます。

プログラムで Buildkite ランナーを作成するには

- Buildkite エージェントトークンを作成し、トークンを内にプレーンテキストで保存します AWS Secrets Manager。
- 2. 任意の設定で CodeBuild プロジェクトを設定します。次の追加属性を設定する必要があります。

- 1. という名前の環境値CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN、タイプ SECRETS_MANAGER、および Buildkite クラスターに関連付けられた Buildkite エージェント トークンと等しい値。
- 2. と等しいソースタイプ NO_SOURCE
- プロジェクトのサービスロールのステップ1で作成したシークレットにアクセスするための アクセス許可

たとえば、次のコマンドを使用して、 CLI を使用して有効な Buildkite ランナープロジェクトを 作成できます。

aws codebuild create-project \ --name buildkite-runner-project \ --source "{\"type\": \"NO_SOURCE\", \"buildspec\":\"\"}" \ --environment "{\"image\":\"aws/codebuild/amazonlinux-x86_64-standard:5.0\", \"type\":\"LINUX_CONTAINER\",\"computeType\":\"BUILD_GENERAL1_MEDIUM\", \"environmentVariables\":[{\"name\":\"CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN\", \"type\":\"SECRETS_MANAGER\",\"value\":\"<buildkite-secret-name>\"}]}" \ --artifacts "{\"type\": \"NO_ARTIFACTS\"}" \ --service-role <service-role>

- ステップ2で作成したプロジェクトに Buildkite ランナーウェブフックを作成します。ウェブ フックを作成するときは、次の設定オプションを使用する必要があります。
 - 1. build-type は と等しくなければなりません RUNNER_BUILDKITE_BUILD
 - 2. タイプEVENTが でパターンが に等しいフィルター WORKFLOW_JOB_QUEUED

たとえば、次のコマンドを使用して、 CLI を使用して有効な Buildkite ランナーウェブフックを 作成できます。

aws codebuild create-webhook \
--project-name buildkite-runner-project \
--filter-groups "[[{\"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]" \
--build-type RUNNER_BUILDKITE_BUILD

create-webhook 呼び出しによって返されたペイロード URL とシークレット値を保存し、認証情報を使用して Buildkite コンソール内にウェブフックを作成します。このリソースの設定方法については、チュートリアル: CodeBuild がホストする Buildkite ランナーを設定する「」の「ステップ 3: Buildkite 内で CodeBuild ウェブフックを作成する」を参照してください。

失敗したビルドまたはハングアップジョブのウェブフックのトラブルシューティング

問題:

で設定したウェブフック<u>チュートリアル: CodeBuild がホストする Buildkite ランナーを設定する</u>が機 能していないか、ワークフロージョブが Buildkite でハングしています。

考えられる原因:

- webhook job.scheduled イベントがビルドのトリガーに失敗している可能性があります。[レスポンス] ログを確認して、レスポンスまたはエラーメッセージを表示します。
- ジョブを処理するために Buildkite セルフホスト型ランナーエージェントを開始する前に、CodeBuild ビルドが失敗します。

推奨される解決策:

失敗した Buildkite ウェブフックイベントをデバッグするには:

- Buildkite 組織設定で、通知サービスに移動し、CodeBuild ウェブフックを選択し、リクエストロ グを見つけます。
- スタックした Buildkite ジョブに関連付けられたjob.scheduledウェブフックイベントを見つ けます。ウェブフックペイロード内のジョブ ID フィールドを使用して、ウェブフックイベント を Buildkite ジョブに関連付けることができます。
- レスポンスタブを選択し、レスポンス本文を確認します。レスポンスステータスコードが 200であり、レスポンス本文に予期しないメッセージが含まれていないことを確認します。

Request Log			
400		job.scheduled 0.29s	2024-12-11 23:03:33 UTC 🚿
Request	Response		
Headers			
Date: Wed, 1	11 Dec 2024 23:03:33 GMT		
Connection:	close		
Content-Type	e: application/json		
Content-Len	gth: 92		
X-Amzn-Erro	rtype: InvalidInputExcept	tion:http://	
X-Amzn-Requ	estid: 7a931bed-0bae-4c10	a-9f5b-2178900cd180	
Body			
{ "message" }	: "Project name in label	nonMatchingProjectName did not match	actual project name"

ウェブフックのアクセス許可に関する問題のトラブルシューティング

問題:

アクセス許可の問題により、Buildkite ジョブはジョブのソースリポジトリのチェックアウトに失敗 します。

考えられる原因:

- CodeBuildには、ジョブのソースリポジトリをチェックアウトするための十分なアクセス許可がありません。
- パイプラインのリポジトリ設定は、CodeBuild マネージド認証情報の SSH を使用してチェックア ウトするように設定されています。

推奨される解決策:

- CodeBuild に、ジョブのソースリポジトリをチェックアウトするための十分なアクセス許可が設定 されていることを確認します。さらに、CodeBuild プロジェクトのサービスロールに、設定された ソースアクセス許可オプションにアクセスするための十分なアクセス許可があることを確認しま す。
- CodeBuild マネージドソースリポジトリ認証情報を使用している場合は、Buildkite パイプラインが HTTPS を使用したチェックアウトを使用するように設定されていることを確認します。

CodeBuild がホストする Buildkite ランナーでサポートされているラベルオーバーライ ド

Buildkite パイプラインステップのエージェントタグラベルでは、セルフホスト型ランナービルドを 変更するさまざまなラベルオーバーライドを指定できます。CodeBuild で認識されないビルドは無視 されますが、ウェブフックリクエストは失敗しません。たとえば、次のワークフロー YAML には、 イメージ、インスタンスサイズ、フリート、および buildspec のオーバーライドが含まれます。

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
      buildspec-override: "true"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

project:codebuild-<project-name>(必須)

- 例: project: "codebuild-myProject"
- すべての Buildkite パイプラインステップ設定に必要です。<project name>は、セルフホスト 型ランナーウェブフックが設定されているプロジェクトの名前と同じである必要があります。

```
queue: "<queue-name>"
```

- 例: queue: "<queue-name>"
- Buildkite ジョブを特定のキューにルーティングするために使用されます。詳細については、「Buildkite エージェントキュータグ」を参照してください。

image: "<environment-type>-<image-identifier>"
- 例:image: "arm-3.0"
- キュレートされたイメージでセルフホストランナービルドを開始するときに使用されるイメージと環境タイプを上書きします。サポートされている値については、「CodeBuild がホストする Buildkite ランナーでサポートされているイメージをコンピューティングする」を参照してください。
 - カスタムイメージで使用されるイメージと環境タイプを上書きするには、を使用します。 image: "custom-<environment-type>-<custom-image-identifier>"
 - 2. 例:

image:
 "custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0"

Note

カスタムイメージがプライベートレジストリに存在する場合は、CodeBuild プロジェクト で適切なレジストリ認証情報を設定する必要があります。

instance-size: "<instance-size>"

- 例:instance-size: "medium"
- セルフホスト型ランナーのビルドの開始時に使用するインスタンスタイプを上書きします。サポートされている値については、「CodeBuild がホストする Buildkite ランナーでサポートされている イメージをコンピューティングする」を参照してください。

fleet: "<fleet-name>"

- 例:fleet: "myFleet"
- 指定されたフリートを使用するために、プロジェクトに設定されたフリート設定を上書きします。
 詳細については、「リザーブドキャパシティフリートでビルドを実行する」を参照してください。

buildspec-override: "<boolean>"

- 例: buildspec-override: "true"
- ・ true に設定されている場合、ビルドが INSTALL、PRE_BUILD、および POST_BUILD フェーズ で buildspec コマンドを実行できるようにします。

CodeBuild がホストする Buildkite ランナーでサポートされているイメージをコン ピューティングする

「<u>のセルフマネージド Buildkite ランナー AWS CodeBuild</u>」で設定したラベルでは、最初の 3 つの列 の値を使用して Amazon EC2 環境設定を上書きできます。CodeBuild では、次の Amazon EC2 コン ピューティングイメージが用意されています。詳細については、以下を参照してください。

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	解像イメージ	定義
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:4.0	<u>al/standa</u> <u>rd/4.0</u>
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codeb uild/amaz onlinux-x 86_64-sta ndard:5.0	<u>al/standa</u> <u>rd/5.0</u>
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codeb uild/amaz onlinux-a arch64-st andard:2. Ø	<u>al/aarch64/</u> standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codeb uild/amaz onlinux-a arch64-st andard:3. Ø	al/aarch64/ standard/3.0

環境タイプ	イメージ識別 子	インスタンス サイズ	プラット フォーム	解像イメージ	定義
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codeb uild/stan dard:5.0	<u>ubuntu/st</u> andard/5.0
ubuntu	6.0	large xlarge	Ubuntu 22.04	aws/codeb uild/stan dard:6.0	<u>ubuntu/st</u> andard/6.0
ubuntu	7.0	gpu_small	Ubuntu 22.04	aws/codeb uild/stan dard:7.0	<u>ubuntu/st</u> andard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-1.0	該当なし
			Windows Server Core 2022	aws/codeb uild/wind ows-base: 2022-1.0	該当なし
windows	2.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-2.0	該当なし
windows	3.0		Windows Server Core 2019	aws/codeb uild/wind ows-base: 2019-3.0	該当なし

さらに、次の値を使用して Lambda 環境設定を上書きできます。CodeBuild Lambda コンピューティ ングの詳細については、「<u>AWS Lambda コンピューティングでビルドを実行する</u>」を参照してくだ さい。CodeBuild は、次の Lambda コンピューティングイメージをサポートしています。

環境タイプ	イメージ識別 子	インスタンス サイズ
linux-lam bda	dotnet6	1GB
arm lambd	go1.21	2GB
arm-rampo a	corretto1	4GB
	T	8GB
	corretto1 7	10GB
	corretto2 1	
	nodejs18	
	nodejs20	
	python3.1 1	
	python3.1 2	
	ruby3.2	

詳細については、「<u>ビルド環境のコンピューティングモードおよびタイプ</u>」および「<u>CodeBuild に用</u> 意されている Docker イメージ」を参照してください。

でウェブフックを使用する AWS CodeBuild

AWS CodeBuild は、GitHub、GitHub Enterprise Server、GitLab、GitLab Self Managed、Bitbucket とのウェブフック統合をサポートしています。

トピック

- AWS CodeBuildでのウェブフック使用のベストプラクティス
- Bitbucket ウェブフックイベント
- GitHub グローバルおよび組織のウェブフック
- GitHub 手動ウェブフック
- GitHub ウェブフックイベント
- GitLab グループウェブフック
- GitLab 手動ウェブフック
- GitLab ウェブフックイベント
- Buildkite 手動ウェブフック

AWS CodeBuildでのウェブフック使用のベストプラクティス

パブリックリポジトリを使用してウェブフックをセットアップするプロジェクトでは、以下のオプ ションを使用することをお勧めします。

ACTOR_ACCOUNT_ID フィルタを設定

プロジェクトのウェブフックフィルタグループに ACTOR_ACCOUNT_ID フィルタを追加して、 ビルドをトリガーできるユーザーを指定します。CodeBuild に配信されるすべてのウェブフック イベントには、アクターの識別子を指定する送信者情報が含まれています。CodeBuild は、フィ ルタで提供される正規表現パターンに基づいてウェブフックをフィルタリングします。このフィ ルタを使用して、ビルドのトリガーを許可する特定のユーザーを指定できます。詳細について は、<u>GitHub ウェブフックイベント</u>および<u>Bitbucket ウェブフックイベント</u>を参照してください。

FILE_PATH フィルタを設定

プロジェクトのウェブフックフィルタグループに FILE_PATH フィルタを追加して、変更時にビ ルドをトリガーできるファイルを含めるか除外します。例えば、^buildspec.yml\$ などの正規 表現パターンを excludeMatchedPattern プロパティと使用して、buildspec.yml ファイル への変更に対するビルドリクエストを拒否できます。詳細については、<u>GitHub ウェブフックイベ</u> ントおよびBitbucket ウェブフックイベントを参照してください。

ビルドの IAM ロールのアクセス権限を絞り込む

Webhook によってトリガーされたビルドは、プロジェクトで指定された IAM サービスロールを 使用します。サービスロールのアクセス許可は、ビルドの実行に必要な最小限のアクセス許可 セットに設定することをお勧めします。たとえば、テストおよびデプロイのシナリオでは、テスト用にプロジェクトを1つ作成し、デプロイ用に別のプロジェクトを作成します。テストプロジェクトは、リポジトリからの webhook ビルドを受け付けますが、リソースへの書き込み権限は提供しません。デプロイメントプロジェクトはリソースへの書き込み権限を提供し、webhookフィルターは信頼済みのユーザーにのみビルドをトリガーできるように設定されています。

インラインまたは Amazon S3 に保管した buildspec を使用する

プロジェクト自体内で buildspec をインラインで定義する場合、または buildspec ファイルを Amazon S3 バケットに格納する場合、buildspec ファイルはプロジェクト所有者のみに表示 されます。これにより、プル要求が buildspec ファイルにコードを変更したり、不要なビル ドをトリガーしたりするのを防ぎます。詳細については、「CodeBuild API リファレンス」の 「ProjectSource.buildspec」を参照してください。

Bitbucket ウェブフックイベント

Webhook フィルタグループを使用して、ビルドをトリガーする Bitbucket ウェブフックイベントを 指定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定で きます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされま す。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグ ループを作成する際、以下を指定します。

イベント

Bitbucket では、次のイベントのうち、1 つ以上を選択できます:

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

ウェブフックのイベントタイプは、X-Event-Key フィールドのヘッダーに含まれています。次 の表に、X-Event-Key ヘッダー値がイベントタイプにマッピングされる方法を示します。

Note

PULL_REQUEST_MERGED イベントタイプを使用するウェブフックフィルタグループを作 成する場合は、Bitbucket ウェブフック設定で merged イベントを有効にする必要があり ます。PULL_REQUEST_CLOSED イベントタイプを使用するウェブフックフィルタグルー プを作成する場合は、Bitbucket ウェブフック設定で declined イベントも有効にする必 要があります。

X-Event-Key ヘッダー値	イベントタイプ
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED
pullrequest:rejected	PULL_REQUEST_CLOSED

PULL_REQUEST_MERGED の場合、プルリクエストがスカッシュ戦略とマージされ、プルリク エストブランチが閉じられると、元のプルリクエストコミットは存在しなくなります。この場 合、CODEBUILD_WEBHOOK_MERGE_COMMIT 環境変数には、圧縮されたマージコミットの識別子 が含まれます。

1つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

Bitbucket アカウント ID が正規表現パターンと一致すると、ビルドがウェブフックイベント でトリガーされます。この値は、ウェブフックフィルタペイロードの actor オブジェクトの account_id プロパティに表示されます。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると (refs/heads/branch-name と refs/tags/tag-name など)、ウェブフックイベントによってビルドがトリガーされま す。HEAD_REF フィルタは、ブランチまたはタグについて Git 参照名を評価します。ブランチ 名またはタグ名は、ウェブフックペイロードの push オブジェクトにある、new オブジェク トの name フィールドに表示されます。プルリクエストイベントの場合、ブランチ名はウェブ フックペイロードの source オブジェクトにある、branch オブジェクトの name フィールド に表示されます。

BASE_REF

基本参照が正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされま す。BASE_REF フィルタは、プルリクエストイベントでのみ使用できます (例: refs/heads/ branch-name)。BASE_REF フィルタは、ブランチの Git 参照名を評価します。ブランチ名 は、ウェブフックペイロードの destination オブジェクトにある、branch オブジェクト の name フィールドに表示されます。

FILE_PATH

変更されたファイルのパスが正規表現パターンに一致すると、ビルドが Webhook イベントで トリガーされます。

COMMIT_MESSAGE

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをトリ ガーします。

WORKFLOW_NAME

ワークフロー名が正規表現パターンに一致する場合に、ウェブフックはビルドをトリガーしま す。

Note

ウェブフックペイロードは、Bitbucket リポジトリのウェブフック設定で見つかります。

トピック

- Bitbucket ウェブフックイベントのフィルタリング (コンソール)
- Bitbucket ウェブフックイベントのフィルタリング (SDK)

• Bitbucket ウェブフックイベントのフィルタリング (AWS CloudFormation)

Bitbucket ウェブフックイベントのフィルタリング (コンソール)

を使用してウェブフックイベント AWS Management Console をフィルタリングするには:

- 1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を 選択します。
- 2. [イベントタイプ] から、1 つ以上のイベントを選択します。
- イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する]で、1つ以上のオプションフィルタを追加します。
- イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始 しない] で、1 つ以上のオプションフィルタを追加します。
- 5. 別のフィルタグループを追加するには、[フィルタグループの追加]を選択します。

詳細については、「AWS CodeBuild API リファレンス」の「<u>ビルドプロジェクトの作成 (コンソー</u> ル)」および「WebhookFilter」を参照してください。

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーしま す。

Filter group 1

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.



- Start a build under these conditions optional
- Don't start a build under these conditions optional

2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリ ガーされます。

```
Bitbucket ウェブフックイベント
```

Remove filter group

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ に一致する Git 参照と ^refs/ heads/branch1! に一致するヘッド参照を含むブランチで作成または更新されたプルリクエスト を指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/branch1\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

Webhook event filter grou	p 1				
Event type Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.					
		•]		
PULL_REQUEST_CREATED	PULL_REQUEST_CREATED X PULL_REQUEST_UPDATED X				
Start a build under the	se conditions				
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional		
	^refs/heads/branch1\$	^refs/heads/main\$			
COMMIT_MESSAGE - optional					
Don't start a build und	er these conditions				
Webhook event filter grou	p 2		Remove filter group		
Webhook event filter grou Event type Add one or more a webhook eve then a new build is triggered ev	p 2 ent filter groups to specify which ever ery time a code change is pushed to y	its trigger a new build. If you do no our repository.	Remove filter group		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event	p 2 ent filter groups to specify which ever ery time a code change is pushed to y	nts trigger a new build. If you do no our repository.	Remove filter group		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event PUSH X	p 2 ent filter groups to specify which ever ery time a code change is pushed to y	nts trigger a new build. If you do no our repository.	Remove filter group		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event PUSH X Start a build under the ACTOR ID - ontional	ent filter groups to specify which ever ery time a code change is pushed to y ese conditions HEAD_REE - optional	nts trigger a new build. If you do no our repository.	Remove filter group		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event PUSH X Start a build under the ACTOR_ID - optional	ent filter groups to specify which ever ery time a code change is pushed to y ese conditions HEAD_REF - optional ^refs/heads/branch1\$	nts trigger a new build. If you do no our repository.	Remove filter group ot add a webhook event filter group, FILE_PATH - optional		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event PUSH ★ ▼ Start a build under the ACTOR_ID - optional COMMIT_MESSAGE - optional	ent filter groups to specify which ever ery time a code change is pushed to y ese conditions HEAD_REF - optional ^refs/heads/branch1\$	nts trigger a new build. If you do no our repository. Image: start string start start start start start start string start sta	Remove filter group ot add a webhook event filter group, FILE_PATH - optional		
Webhook event filter group Event type Add one or more a webhook event then a new build is triggered event PUSH ★ ▼ Start a build under the ACTOR_ID - optional COMMIT_MESSAGE - optional	ent filter groups to specify which ever ery time a code change is pushed to y ese conditions HEAD_REF - optional ^refs/heads/branch1\$	BASE_REF - optional	Remove filter group ot add a webhook event filter group, FILE_PATH - optional		

この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビ ルドをトリガーします。

Bitbucket ウェブフックイベント

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	d a webhook event filter group,
•	
PUSH \times PULL_REQUEST_CREATED \times PULL_REQUEST_UPDATED \times	
PULL_REQUEST_MERGED \times PULL_REQUEST_CLOSED \times	
Start a build under these conditions - optional	
Don't start a build under these conditions - optional	Add filter
Filter 1	
Туре	
HEAD_REF	
Pattern	
^refs/tags/.*	

この例では、ウェブフックフィルタグループは、正規表現 ^buildspec.* に一致する名前のファイ ルが変更された場合にのみビルドをトリガーします。

Webhook event filter group 1

Event type

PUSH X		•	
 Start a build under th 	ese conditions		
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional ^buildspec.*
COMMIT_MESSAGE -			

この例で、Webhook フィルターグループは、ファイルが src または test フォルダーで変更された 場合にのみ、ビルドをトリガーします。

Webhook event filter group 1

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

Ŧ

push imes

Start a build under these conditions

ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional
			^src/.+ ^test/.+
COMMIT_MESSAGE - optional			
Don't start a build und	er these conditions		

この例では、正規表現 actor-account-id と一致するアカウント ID を持たない Bitbucket ユー ザーが変更を行った場合にのみ、ウェブフックフィルタグループがビルドをトリガーします。

Note

Bitbucket アカウント ID の検索方法については、「https://api.bitbucket.org/2.0/users/*user-name*」を参照してください。ここで、*user-name* は、Bitbucket のユーザー名を表します。

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

▼				
PUSH \times PULL_REQUEST_CREATED \times PULL_REQUEST_UPDATED \times				
PULL_REQUEST_MERGED \times PULL_REQUEST_CLOSED \times				
Start a build under these conditions - optional	Add filter			
Filter 2				
Туре				
ACTOR_ACCOUNT_ID				
Pattern				
actor-account-id				

この例では、HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。

Webhook event filter group 1

Event type

push \times			
 Start a build under the 	hese conditions		
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional
COMMIT_MESSAGE -			
\[CodeBuild\]			

Don't start a build under these conditions

Bitbucket ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、 CreateWebhookまたは UpdateWebhook API メソッドのリクエスト構文で filterGroupsフィー ルドを使用します。詳細については、CodeBuild API リファレンスの「<u>WebhookFilter</u>」を参照して ください。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリ クエスト構文に挿入します。

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するに は、pattern パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定しま す。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されると トリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ に一致する Git 参照と ^refs/ heads/myBranch\$ に一致するヘッド参照を含むブランチで作成または更新されたプルリクエス トを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/myBranch\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

```
"filterGroups": [
  Γ
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  Γ
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
]
```

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定する ことができます。この例では、ビルドは、タグイベントを除くすべてのリクエストに対してトリガー されます。

アカウント ID actor-account-id を持つ Bitbucket ユーザーによって変更が行われた場合にのみ ビルドをトリガーするフィルタを作成できます。

Note

Bitbucket アカウント ID の検索方法については、「https://api.bitbucket.org/2.0/users/*user-name*」を参照してください。ここで、*user-name* は、Bitbucket のユーザー名を表します。

引数 pattern の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーする フィルタを作成することができます。この例のフィルタグループでは、正規表現 ^buildspec.* に 一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
 [
 {
    "type": "EVENT",
    "pattern": "PUSH"
 },
 {
    "type": "FILE_PATH",
    "pattern": "^buildspec.*"
 }
]
]
```

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にの み、ビルドをトリガーするように指定しています。

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーする フィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミット メッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定しま す。

```
"filterGroups": [
   [
    {
        "type": "EVENT",
        "pattern": "PUSH"
   },
    {
        "type": "COMMIT_MESSAGE",
```

```
ユーザーガイド
```

```
"pattern": "\[CodeBuild\]"
}
]
]
```

Bitbucket ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、 AWS CodeBuild プロジェクトの FilterGroupsプロパティを使用します。以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが作成されます。また、 一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルタグループでは、アカウント ID ^refs/heads/main\$を持たない Bitbucket ユー ザーが、正規表現 12345 と一致する Git 参照名を持つブランチに対してプルリクエストを作成ま たは更新することを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/.*と一致する Git 参照名を持つブラン
 チに対するプッシュリクエストを作成することを指定します。
- 3番目のフィルタグループでは、正規表現 \[CodeBuild\] に一致する HEAD コミットメッセー ジを使用してプッシュリクエストを指定します。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED
          - Type: BASE_REF
```

Pattern: ^refs/heads/main\$	
ExcludeMatchedPattern: false	
- Type: ACTOR_ACCOUNT_ID	
Pattern: 12345	
ExcludeMatchedPattern: true	
Type: EVENT	
Pattern: PUSH	
- Type: HEAD_REF	
Pattern: ^refs/heads/.*	
- Type: FILE_PATH	
Pattern: READ_ME	
ExcludeMatchedPattern: true	
Type: EVENT	
Pattern: PUSH	
- Type: COMMIT_MESSAGE	
<pre>Pattern: \[CodeBuild\]</pre>	
- Type: FILE_PATH	
<pre>Pattern: ^src/.+ ^test/.+</pre>	

GitHub グローバルおよび組織のウェブフック

CodeBuild GitHub グローバルまたは組織のウェブフックを使用して、GitHub 組織またはエンター プライズ内の任意のリポジトリからのウェブフックイベントでビルドを開始できます。グローバ ルおよび組織のウェブフックは、既存の GitHub ウェブフックイベントタイプのいずれでも動作 し、CodeBuild ウェブフックの作成時にスコープ設定を追加することで設定できます。グローバル および組織のウェブフックを使用して CodeBuild 内でセルフホスト型 GitHub Action Runner を設 定し、単一のプロジェクト内の複数のリポジトリから WORKFLOW_JOB_QUEUED イベントを受信する こともできます。

トピック

- グローバルまたは組織の GitHub ウェブフックを設定
- GitHub グローバルまたは組織のウェブフックイベントをフィルタリング (コンソール)
- ・ GitHub 組織のウェブフックイベントをフィルタリング (AWS CloudFormation)

グローバルまたは組織の GitHub ウェブフックを設定

グローバルまたは組織の GitHub ウェブフックを設定するための大まかなステップは次のとおりで す。グローバルおよび組織の GitHub ウェブフックの詳細については、「<u>GitHub グローバルおよび</u> 組織のウェブフック」を参照してください。

- 1. プロジェクトのソースの場所を CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION に設定します。
- 2. ウェブフックのスコープ設定で、組織か<u>グローバルウェブフック</u>かに応じて、スコープを GITHUB_ORGANIZATION または GITHUB_GLOBAL に設定します。詳細については、「<u>Types of</u> webhooks」を参照してください。
- ウェブフックのスコープ設定の一部として名前を指定します。組織ウェブフックの場合、これは 組織名であり、グローバルウェブフックの場合、これはエンタープライズ名です。

Note

プロジェクトのソースタイプが GITHUB_ENTERPRISE の場合、ウェブフックスコープ設 定の一部としてドメインも指定する必要があります。

- (オプション) 組織またはエンタープライズ内の特定のリポジトリのウェブフックイベントのみを 受信する場合は、ウェブフックの作成時に REPOSITORY_NAME をフィルタとして指定できます。
- 5. 組織ウェブフックを作成する場合は、CodeBuild に GitHub 内で組織レベルのウェブフックを作 成するアクセス許可があることを確認してください。組織のウェブフックアクセス許可を持つ GitHub 個人用アクセストークンを作成するか、CodeBuild OAuth を使用できます。詳細について は、「GitHub および GitHub Enterprise Server アクセストークン」を参照してください。

組織のウェブフックは、既存の GitHub ウェブフックイベントタイプのいずれでも動作することに 注意してください。

グローバルウェブフックを作成する場合は、ウェブフックを手動で作成する必要があります。GitHub 内でウェブフックを手動で作成する方法の詳細については、「GitHub 手動ウェブフック」を参照してください。

グローバルウェブフックは WORKFLOW_JOB_QUEUED イベントタイプのみをサポートすることに 注意してください。詳細については、「<u>チュートリアル: CodeBuild がホストする GitHub Actions</u> ランナーを設定」を参照してください。

GitHub グローバルまたは組織のウェブフックイベントをフィルタリング (コンソール)

コンソールから GitHub プロジェクトを作成するときは、次のオプションを選択して、プロジェクト 内に GitHub グローバルまたは組織のウェブフックを作成します。グローバルおよび組織の GitHub ウェブフックの詳細については、「<u>GitHub グローバルおよび組織のウェブフック</u>」を参照してくだ さい。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- 2. ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
 - [Source (ソース)] で、次のようにします。
 - ・ [ソースプロバイダ] には、[GitHub]、または [GitHub Enterprise] を選択します。
 - [リポジトリ] で、[GitHub スコープ付きウェブフック] を選択します。

GitHub リポジトリは自動的に CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION に設 定されます。これは、グローバルおよび組織のウェブフックに必要なソースの場所です。

1 Note

組織ウェブフックを使用している場合は、CodeBuild に GitHub 内で組織レベル のウェブフックを作成するアクセス許可があることを確認してください。既存の OAuth 接続を使用している場合は、CodeBuild にこのアクセス許可を付与するため に、接続を再生成する必要がある場合があります。または、<u>CodeBuild の手動ウェ</u> <u>ブフック機能</u>を使用して、ウェブフックを手動で作成することもできます。既存の GitHub OAuth トークンがあり、追加の組織アクセス許可を追加する場合は、<u>OAuth</u> トークンのアクセス許可を取り消し、CodeBuild コンソールからトークンを再接続 できます。

Source		Add source
Source 1 - Primary		
Source provider		
GitHub		
Repository		
 Repository in my GitHub account 	O Public repository	• GitHub scoped webhook
GitHub repository		
CODEBUILD_DEFAULT_WEBHOOK_	SOURCE_LOCATION	
Connection status		
You are connected to GitHub using a	personal access token.	

Disconnect from GitHub

- ・ [プライマリソースのウェブフックイベント] の場合:
 - [スコープタイプ] では、組織ウェブフックを作成する場合は [組織レベル]、グローバルウェ ブフックを作成する場合は [エンタープライズレベル] を選択します。
 - [名前]には、ウェブフックがグローバルウェブフックか組織ウェブフックかに応じて、エン タープライズまたは組織名を入力します。

プロジェクトのソースタイプが GITHUB_ENTERPRISE の場合、ウェブフック組織設定 の一部としてドメインも指定する必要があります。例えば、組織の URL が https:// domain.com/orgs/org-name の場合、ドメインは https://domain.com です。

Note

この名前をウェブフックの作成後に変更することはできません。名前を変更する には、ウェブフックを削除して再作成します。ウェブフックを完全に削除する場合 は、プロジェクトソースの場所を GitHub リポジトリに更新することもできます。

Primary source webhook events Info	Add filter group	
Webhook - <i>optional</i> Info 🔀 Rebuild every time a code change is pushed to	o this repository	
Scope type		
• Organization level	○ Enterprise level	
Organization name Your GitHub organization name.		
organization-name		
Build type		
• Single build Triggers single build	O Batch build Triggers multiple builds as single	execution
Additional configuration		

 (オプション) [ウェブフックイベントフィルタグループ] では、<u>新しいビルドをトリガーする</u> <u>イベント</u>を指定できます。また、REPOSITORY_NAME をフィルタとして指定して、特定の リポジトリからのウェブフックイベントでのみ、ビルドをトリガーすることもできます。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1	Remove filter group
Event type - <i>optional</i> Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	a webhook event filter group,
•	
WORKFLOW_JOB_QUEUED X	
Start a build under these conditions - optional	Add filter
Filter 1	
Туре	
REPOSITORY_NAME 🔻	
Pattern	
repository-name	
Remove	

イベントタイプを WORKFLOW_JOB_QUEUED に設定して、セルフホスト型 GitHub Actions ランナーを設定することもできます。詳細については、「<u>チュートリアル: CodeBuild がホ</u> ストする GitHub Actions ランナーを設定」を参照してください。

3. デフォルト値のまま続行し、[ビルドプロジェクトを作成する]を選択します。

GitHub 組織のウェブフックイベントをフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用して組織のウェブフックイベントを AWS CodeBuild フィ ルタリングするには、プロジェクトの ScopeConfigurationプロパティを使用します。グローバ ルおよび組織の GitHub ウェブフックの詳細については、「<u>GitHub グローバルおよび組織のウェブ</u> フック」を参照してください。

Note

グローバルウェブフックと GitHub Enterprise ウェブフックは ではサポートされていません AWS CloudFormation。 CodeBuildProject:

テンプレートの次の YAML 形式の部分は、4 つのフィルターグループ AWS CloudFormation を作 成します。1 つまたはすべてが true と評価されると、これらが一緒になってビルドをトリガーしま す。

- 最初のフィルタグループでは、アカウント ID ^refs/heads/main\$を持たない GitHub ユーザー が、正規表現 12345 と一致する Git 参照名を持つブランチに対してプルリクエストを作成または 更新することを指定します。
- 2番目のフィルターグループでは、正規表現 READ_ME に一致する Git 参照名を持つブランチで正 規表現 ^refs/heads/.*に一致する名前のファイルに対してプッシュリクエストが作成されるこ とを指定します。
- 3番目のフィルタグループでは、正規表現 \[CodeBuild\] に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。
- 4 番目のフィルタグループは、正規表現 \[CI-CodeBuild\] に一致するワークフロー名を持つ GitHub Actions ワークフロージョブリクエストを指定します。

```
Type: AWS::CodeBuild::Project
Properties:
  Name: MyProject
  ServiceRole: service-role
  Artifacts:
    Type: NO_ARTIFACTS
  Environment:
    Type: LINUX_CONTAINER
    ComputeType: BUILD_GENERAL1_SMALL
    Image: aws/codebuild/standard:5.0
  Source:
    Type: GITHUB
    Location: source-location
  Triggers:
    Webhook: true
    ScopeConfiguration:
      Name: organization-name
      Scope: GITHUB_ORGANIZATION
    FilterGroups:
      - - Type: EVENT
          Pattern: PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED
        - Type: BASE_REF
          Pattern: ^refs/heads/main$
          ExcludeMatchedPattern: false
```

	-	Type: ACTOR_ACCOUNT_ID
		Pattern: 12345
		ExcludeMatchedPattern: true
-	-	Type: EVENT
		Pattern: PUSH
	-	Type: HEAD_REF
		Pattern: ^refs/heads/.*
	-	Type: FILE_PATH
		Pattern: READ_ME
		ExcludeMatchedPattern: true
-	-	Type: EVENT
		Pattern: PUSH
	-	Type: COMMIT_MESSAGE
		Pattern: \[CodeBuild\]
	-	Type: FILE_PATH
		Pattern: ^src/.+ ^test/.+
-	-	Type: EVENT
		Pattern: WORKFLOW_JOB_QUEUED
	-	Type: WORKFLOW_NAME
		Pattern: \[CI-CodeBuild\]

GitHub 手動ウェブフック

手動 GitHub ウェブフックを設定して、CodeBuild が GitHub 内で自動的にウェブフックを作成す るのを防ぐことができます。CodeBuild は、ウェブフックを作成するための呼び出しの一部とし てペイロード URL を返します。これを使用して、GitHub 内でウェブフックを手動で作成できま す。GitHub アカウントでのウェブフックの作成を許可するリストに CodeBuild が登録されていない 場合でも、ビルドプロジェクト用にウェブフックを手動で作成できます。

GitHub 手動ウェブフックを作成するには、次の手順に従います。

GitHub 手動ウェブフックを作成するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。
 - ・ [Source (ソース)] で、次のようにします。
 - ・ [ソースプロバイダー] で [GitHub] を選択します。
 - [リポジトリ] では、[GitHub アカウントのリポジトリ] を選択します。

- [リポジトリの URL] に、「https://github.com/user-name/repository-name」と 入力します。
- [プライマリソースのウェブフックイベント] の場合:
 - [ウェブフック オプション] で、[コードの変更がこのレポジトリにプッシュされるたびに再 ビルド] を選択します。
 - [追加設定] を選択し、[手動作成 オプション] で、[GitHub コンソールでこのリポジトリの ウェブフックを手動で作成] を選択します。
- 3. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。[ペイロード URL] と [シークレット] 値は後で使用するため、メモしておきます。

Create webhook		×
You must create a webhook for your GitHub repository.		
Payload URL	Copy payload URL	
Secret	Copy secret 🗇	
		Close

- 4. https://github.com/user-name/repository-name/settings/hooks で GitHub コン ソールを開き、[ウェブフックを追加]を選択します。
 - [ペイロード URL] には、先ほどメモしたペイロード URL 値を入力します。
 - [コンテンツタイプ] には、[application/json] を選択します。
 - [シークレット]には、先ほどメモしたシークレット値を入力します。
 - CodeBuild にウェブフックペイロードを送信する個々のイベントを設定します。[このウェブ フックをトリガーするイベント] として、[個々のイベントを選択] を選択し、[プッシュ]、[プ ルリクエスト]、および [リリース] のイベントから選択します。WORKFLOW_JOB_QUEUED イ ベントのビルドを開始する場合は、[ワークフロージョブ] を選択します。GitHub Actions ラン ナーの詳細については、「チュートリアル: CodeBuild がホストする GitHub Actions ランナー を設定」を参照してください。CodeBuild でサポートされているイベントタイプの詳細につい ては、「GitHub ウェブフックイベント」を参照してください。
- 5. [ウェブフックを追加]を選択します。

GitHub ウェブフックイベント

Webhook フィルタグループを使用して、ビルドをトリガーする GitHub ウェブフックイベントを指 定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定でき ます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされま す。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグ ループを作成する際、以下を指定します。

イベント

GitHub では、次のイベントのうち、1 つ以上を選択できます:

PUSH、PULL_REQUEST_CREATED、PULL_REQUEST_UPDATED、PULL_REQUEST_REOPENED、PULL_R ウェブフックのイベントタイプは、ウェブフックペイロードの X-GitHub-Event ヘッダーに含 まれています。X-GitHub-Event ヘッダーで、pull_request または push が表示される場合 があります。プルリクエストイベントの場合、このタイプはウェブフックイベントペイロードの action フィールドに含まれています。以下の表に示すのは、X-GitHub-Event ヘッダー値と ウェブフックのプルリクエストペイロードの action フィールドが、利用可能なイベントタイプ にマッピングされる方法を示しています。

X-GitHub-Event ヘッ ダー値	ウェブフックイベントペイ ロードの action 値	イベントタイプ
pull_request	opened	PULL_REQUEST_CREATED
pull_request	reopened	PULL_REQUEST_REOPE NED
pull_request	synchronize	PULL_REQUEST_UPDATED
pull_request	closed、および merged フィールドは true	PULL_REQUEST_MERGED
pull_request	closed、および merged フィールドは false	PULL_REQUEST_CLOSED
push	該当なし	PUSH

X-GitHub-Event ヘッ ダー値	ウェブフックイベントペイ ロードの action 値	イベントタイプ
release	released	RELEASED
release	prereleased	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

Note

PULL_REQUEST_REOPENED イベントタイプは GitHub および GitHub Enterprise Server でのみ使用できます。RELEASED および PRERELEASEDイベントタイプは GitHub で のみ使用できます。WORKFLOW_JOB_QUEUED の詳細については、「<u>チュートリアル:</u> CodeBuild がホストする GitHub Actions ランナーを設定」をご参照ください。

1 つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

GitHub、GitHub Enterprise サーバーのアカウント ID が正規表現 パターンと一致する と、Webhook イベントによってビルドがトリガーされます。この値は、ウェブフックペイ ロードの sender オブジェクトの id プロパティで見つかります。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると、ウェブフックイベントによりビルドが トリガーされます (例: refs/heads/branch-name または refs/tags/tag-name)。プッ シュイベントの場合、参照名はウェブフックペイロードの ref プロパティで見つかります。 プルリクエストイベントの場合、ブランチ名はウェブフックペイロードの head オブジェクト の ref プロパティで見つかります。

BASE_REF

基本参照が正規表現パターンと一致するとウェブフックイベントによってビルドがトリガーさ れます。(例 refs/heads/branch-name) BASE_REF フィルタは、プルリクエストイベント でのみ使用できます。ブランチ名は、ウェブフックペイロードで base オブジェクトの ref プロパティで見つかります。

FILE_PATH

変更されたファイルのパスが正規表現パターンと一致すると、ビルドがウェブフックイベン トでトリガーされます。FILE_PATH フィルタは、GitHub のプッシュおよびプルリクエスト イベントと GitHub Enterprise Server のプッシュイベントで使用できます。GitHub Enterprise Server のプルリクエストイベントでは使用できません。

COMMIT_MESSAGE

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをト リガーします。COMMIT_MESSAGE フィルタは、GitHub のプッシュおよびプルリクエストイ ベントと GitHub Enterprise Server のプッシュイベントで使用できます。GitHub Enterprise Server のプルリクエストイベントでは使用できません。

TAG_NAME

リリースのタグ名が正規表現パターンに一致すると、ウェブフックはビルドをトリガーしま す。TAG_NAME フィルタは、GitHub リリースおよびプレリリースされたリクエストイベント で使用できます。

RELEASE_NAME

リリース名が正規表現パターンに一致すると、ウェブフックはビルドをトリガーしま す。RELEASE_NAME フィルタは、GitHub リリースおよびプレリリースされたリクエストイベ ントで使用できます。

REPOSITORY_NAME

リポジトリ名が正規表現パターンに一致すると、ウェブフックはビルドをトリガーしま す。REPOSITORY_NAME フィルタは、GitHub グローバルまたは組織のウェブフックでのみ使 用できます。

ORGANIZATION_NAME

ウェブフックは、組織名が正規表現パターンと一致するときにビルドをトリガーしま す。ORGANIZATION_NAME フィルターは GitHub グローバルウェブフックでのみ使用できま す。

WORKFLOW_NAME

ワークフロー名が正規表現パターンに一致する場合に、ウェブフックはビルドをトリガーしま す。WORKFLOW_NAME フィルタは、GitHub Actions ワークフロージョブのキューに入れられ たリクエストイベントで使用できます。

Note

ウェブフックペイロードは、GitHub リポジトリのウェブフック設定で見つかります。

トピック

- GitHub ウェブフックイベントのフィルタリング (コンソール)
- GitHub ウェブフックイベントのフィルタリング (SDK)
- GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation)

GitHub ウェブフックイベントのフィルタリング (コンソール)

AWS Management Consoleを使用して GitHub ウェブフックイベントをフィルタリングするには、次 の手順を使用します。GitHub ウェブフックイベントの詳細については、「<u>GitHub ウェブフックイベ</u> ント」を参照してください。

[プライマリソース Webhook イベント] で、以下を選択します。このセクションは、ソースリポジト リで [GitHub アカウントのリポジトリ] を選択した場合のみに表示されます。

- プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を 選択します。
- 2. [イベントタイプ] から、1 つ以上のイベントを選択します。
- イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する]で、1つ以上のオプションフィルタを追加します。
- イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
- 5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加]を選択します。

詳細については、「AWS CodeBuild API リファレンス」の「<u>ビルドプロジェクトの作成 (コンソー</u> ル)」および「WebhookFilter」を参照してください。

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーしま す。

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	l a webhook event filter group,
•	
PULL_REQUEST_CREATED \times PULL_REQUEST_UPDATED \times	
PULL_REQUEST_REOPENED \times PULL_REQUEST_MERGED \times	
PULL_REQUEST_CLOSED \times	
Start a build under these conditions - optional	
Don't start a build under these conditions - optional	

2 つのウェブフックフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価 されるとトリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ と一致する Git 参照名および ^refs/heads/branch1\$ と一致するヘッド参照を持つブランチに対してプルリクエストを作 成、更新、または再開することを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/branch1\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

Webhook event filter group Event type Add one or more a webhook event then a new build is triggered every PULL_REQUEST_CREATED PULL_REQUEST_REOPENED	1 t filter groups to specify which even t time a code change is pushed to yo PULL_REQUEST_UPDAT O Constraints	ts trigger a new build. If you do not our repository.	add a webhook event filter group,
 Start a build under these ACTOR_ID - optional COMMIT_MESSAGE - optional Don't start a build under 	HEAD_REF - optional Arefs/heads/branch1\$	BASE_REF - optional ^refs/heads/main\$	FILE_PATH - optional
Webhook event filter group 2 Event type Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository. ■ PUSH ×			
ACTOR_ID - optional COMMIT_MESSAGE - optional Don't start a build under	HEAD_REF - optional <pre>^refs/heads/branch1\$</pre>	BASE_REF - optional	FILE_PATH - optional

この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビ ルドをトリガーします。

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	a webhook event filter group,
•	
PUSH \times PULL_REQUEST_CREATED \times PULL_REQUEST_UPDATED \times	
PULL_REQUEST_REOPENED \times PULL_REQUEST_MERGED \times	
PULL_REQUEST_CLOSED \times	
Start a build under these conditions - optional	
Don't start a build under these conditions - optional	Add filter
Filter 1	
Туре	
HEAD_REF 🔹	
Pattern	
^refs/tags/.*	

この例では、ウェブフックフィルタグループは、正規表現 ^buildspec.* に一致する名前のファイ ルが変更された場合にのみビルドをトリガーします。

Webhook event filter group 1

Event type

PUSH X		•	
 Start a build under th 	lese conditions		
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional ^buildspec.*
OMMIT_MESSAGE -			

この例で、Webhook フィルターグループは、ファイルが src または test フォルダーで変更された 場合にのみ、ビルドをトリガーします。

Webhook event filter group 1

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

		•	
push \times			
▼ Start a build under th	nese conditions		
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional
			^src/.+ ^test/.+
COMMIT_MESSAGE - optional]		
Don't start a build un	der these conditions		
この例では、指定した GitHub ユーザーや GitHub Enterprise Server ユーザーが、正規表現 actoraccount-id と一致するアカウント ID を使用して変更を行った場合にのみ、Webhook フィルタグ ループがビルドをトリガーします。

(i) Note	
GitHub アカウント ID の検索方法については、	「https://api.github.com/users/ <i>user-name</i> 」
を参照してください。ここで、 <i>user-name</i> は	、GitHub のユーザー名を表します。

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	d a webhook event filter group,
▼	
PUSH \times PULL_REQUEST_CREATED \times PULL_REQUEST_UPDATED \times	
PULL_REQUEST_REOPENED \times PULL_REQUEST_MERGED \times	
PULL_REQUEST_CLOSED ×	
Start a build under these conditions - optional	Add filter
Filter 2	
Туре	
ACTOR_ACCOUNT_ID	
Pattern	
actor-account-id	
Remove	
Don't start a build under these conditions - optional	

この例では、HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。

Event type

push ×		•	
Start a build under the start and start a build under the start a build und	hese conditions		
ACTOR_ID - optional	HEAD_REF - optional	BASE_REF - optional	FILE_PATH - optional
COMMIT_MESSAGE - optional			
V.C. 1. D. 11.0.1			

この例では、ウェブフックフィルタグループは GitHub Actions ワークフロージョブイベントのみの ビルドをトリガーします。

Note

CodeBuild は、ウェブフックに [WORKFLOW_JOB_QUEUED] イベントフィルタを含むフィ ルタグループがある場合にのみ、GitHub Actions ワークフロージョブを処理します。

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do no hen a new build is triggered every time a code change is pushed to your repository.	ot add a webhook event filter group,
•	,
WORKFLOW_JOB_QUEUED ×	
Start a build under these conditions - optional	

Don't start a build under these conditions - optional

この例では、ウェブフックフィルタグループが、正規表現 CI-CodeBuild に一致するワークフロー 名のビルドをトリガーします。

Filter group 1	Remove	e filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	i a webhook e	event filter group,
▼		
WORKFLOW_JOB_QUEUED X		
Start a build under these conditions - optional		Add filter
Filter 1		
Туре		
WORKFLOW_NAME		
Pattern		
CI-CodeBuild		
Remove		

Don't start a build under these conditions - optional

GitHub ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、 CreateWebhook または UpdateWebhook API メソッドのリクエスト構文で filterGroupsフィー ルドを使用します。詳細については、CodeBuild API リファレンスの「<u>WebhookFilter</u>」を参照して ください。

GitHub ウェブフックイベントの詳細については、「<u>GitHub ウェブフックイベント</u>」を参照してくだ さい。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリ クエスト構文に挿入します。

```
"filterGroups": [
[
{
"type": "EVENT",
```

```
"pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
}
]
```

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するに は、pattern パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定しま す。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されると トリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ と一致する Git 参照名および ^refs/heads/myBranch\$ と一致するヘッド参照を持つブランチに対してプルリクエストを作 成、更新、または再開することを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/myBranch\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

```
"filterGroups": [
    Ε
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_REOPENED"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
        },
        {
            "type": "BASE_REF",
            "pattern": "^refs/heads/main$"
        }
    ],
    Ε
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
```

	}				
]					
]					

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定する ことができます。たとえば、この例で、ビルドは、タグイベントを除くすべてのリクエストに対して トリガーされます。

引数 pattern の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーする フィルタを作成することができます。この例のフィルタグループでは、正規表現 ^buildspec.* に 一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にの み、ビルドをトリガーするように指定しています。

指定した GitHub ユーザーまたは GitHub Enterprise Server ユーザーがアカウント ID actoraccount-id を使用して変更を行った場合にのみ、ビルドをトリガーするフィルタを作成できま す。

Note

GitHub アカウント ID の検索方法については、「https://api.github.com/users/*user-name*」 を参照してください。ここで、*user-name* は、GitHub のユーザー名を表します。

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーする フィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミット メッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定しま す。

GitHub Actions ワークフロージョブのビルドのみをトリガーするウェブフックフィルタを作成するに は、リクエスト構文に以下を挿入します。

GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、 AWS CodeBuild プロジェクトの FilterGroupsプロパティを使用します。

GitHub ウェブフックイベントの詳細については、「<u>GitHub ウェブフックイベント</u>」を参照してくだ さい。

以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが 作成されます。また、一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルタグループでは、アカウント ID ^refs/heads/main\$を持たない GitHub ユーザー が、正規表現 12345 と一致する Git 参照名を持つブランチに対してプルリクエストを作成または 更新することを指定します。
- 2番目のフィルターグループでは、正規表現 READ_ME に一致する Git 参照名を持つブランチで正 規表現 ^refs/heads/.*に一致する名前のファイルに対してプッシュリクエストが作成されるこ とを指定します。
- ・ 3 番目のフィルタグループでは、正規表現 ∖[CodeBuild\] に一致する HEAD コミットメッセー ジを使用してプッシュリクエストを指定します。
- 4番目のフィルタグループは、正規表現 \[CI-CodeBuild\] に一致するワークフロー名を持つ GitHub Actions ワークフロージョブリクエストを指定します。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD GENERAL1 SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
```

	-	Type: FILE_PATH
		Pattern: READ_ME
		ExcludeMatchedPattern: true
-	-	Type: EVENT
		Pattern: PUSH
	-	Type: COMMIT_MESSAGE
		Pattern: \[CodeBuild\]
	-	Type: FILE_PATH
		Pattern: ^src/.+ ^test/.+
-	-	Type: EVENT
		Pattern: WORKFLOW_JOB_QUEUED
	-	Type: WORKFLOW_NAME
		Pattern: \[CI-CodeBuild\]

GitLab グループウェブフック

CodeBuild GitLab グループウェブフックを使用して、GitLab グループ内の任意のリポジトリから ウェブフックイベントでビルドを開始できます。グループウェブフックは、既存の GitLab ウェブ フックイベントタイプのいずれでも動作し、CodeBuild ウェブフックの作成時にスコープ設定を追加 することで設定できます。グループウェブフックを使用して <u>CodeBuild 内でセルフホスト型 GitLab</u> <u>ランナーを設定</u>し、単一のプロジェクト内の複数のリポジトリから WORKFLOW_JOB_QUEUED イベン トを受信することもできます。

トピック

- ・ グループ GitLab ウェブフックを設定
- GitLab グループウェブフックイベントのフィルタリング (コンソール)
- GitLab グループウェブフックイベントのフィルタリング (AWS CloudFormation)

グループ GitLab ウェブフックを設定

グループ GitLab ウェブフックを設定する大まかなステップは次のとおりです。グループ GitLab ウェ ブフックの詳細については、「GitLab グループウェブフック」を参照してください。

- 1. プロジェクトのソースの場所を CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION に設定します。
- 2. ウェブフックのスコープ設定で、スコープを GITLAB_GROUP に設定します。
- ウェブフックのスコープ設定の一部として名前を指定します。グループウェブフックの場合、これはグループ名です。

(i) Note

プロジェクトのソースタイプが GITLAB_SELF_MANAGED の場合、ウェブフックスコープ 設定の一部としてドメインも指定する必要があります。

- (オプション) 組織またはエンタープライズ内の特定のリポジトリのウェブフックイベントのみを 受信する場合は、ウェブフックの作成時に REPOSITORY NAME をフィルタとして指定できます。
- 5. グループウェブフックを作成するときは、CodeBuild に GitLab 内でグループレベルのウェブフッ クを作成するアクセス許可があることを確認してください。CodeConnections から CodeBuild OAuth を使用して確認できます。詳細については、「<u>CodeBuild での GitLab アクセス</u>」を参照し てください。

グループウェブフックは、既存の GitLab ウェブフックイベントタイプのいずれでも動作すること に注意してください。

GitLab グループウェブフックイベントのフィルタリング (コンソール)

コンソールから GitLab プロジェクトを作成するときは、次のオプションを選択して、プロジェクト 内に GitLab グループウェブフックを作成します。グループ GitLab ウェブフックの詳細については、 「<u>GitLab グループウェブフック</u>」を参照してください。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- 2. ビルドプロジェクトを作成します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u>ル)」および「ビルドの実行 (コンソール)」を参照してください。
 - [Source (ソース)] で、次のようにします。
 - ・ [ソースプロバイダ] では、[GitLab] または [GitLab セルフマネージド] を選択します。
 - [リポジトリ] で、[GitLab スコープ付きウェブフック] を選択します。

GitLab リポジトリは自動的に CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION に設 定されます。これは、グループウェブフックに必要なソースの場所です。

Note

グループウェブフックを使用するときは、CodeBuild に GitLab 内でグループレベ ルのウェブフックを作成するアクセス許可があることを確認してください。既存の



- [プライマリソースのウェブフックイベント]の場合:
 - [グループ名] に、グループ名を入力します。

プロジェクトのソースタイプが GITLAB_SELF_MANAGED の場合、ウェブフックグ ループ設定の一部としてドメインも指定する必要があります。例えば、グループの URL が https://domain.com/group/group-name の場合、ドメインは https:// domain.com です。

Note

この名前をウェブフックの作成後に変更することはできません。名前を変更する には、ウェブフックを削除して再作成します。ウェブフックを完全に削除する場合 は、プロジェクトソースの場所を GitLab リポジトリに更新することもできます。 ٠

Primary source webhook events Info	Add filter group
Nebhook - optional Info 🔀	
Rebuild every time a code change is pushed to this re	epository
Group name 'our GitLab group name.	
group-name	
Build type	
Single build Triggers single build	O Batch build Triggers multiple builds as single execution
 Additional configuration 	
ベント を指定できます。また、REPOSI	TORY_NAME をフィルタとして指定して、特定
ベント を指定できます。また、REPOSI ポジトリからのウェブフックイベントて Webhook event filter groups A build is triggered if any filter group evaluates to true, which o	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます
<u>ベント</u> を指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true.
<u>ベント</u> を指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which o then a new build is triggered every time a code change is pushe	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベント を指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which o then a new build is triggered every time a code change is pushe	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which o then a new build is triggered every time a code change is pushe WORKFLOW_JOB_QUEUED × Start a build under these conditions - optional	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which o then a new build is triggered every time a code change is pushe WORKFLOW_JOB_QUEUED × Start a build under these conditions - optional Filter 1	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which o then a new build is triggered every time a code change is pushe WORKFLOW_JOB_QUEUED × ▼ Start a build under these conditions - optional Filter 1 Type	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which the then a new build is triggered every time a code change is pushed WORKFLOW_JOB_QUEUED × ▼ Start a build under these conditions - optional Filter 1 Type REPOSITORY_NAME	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which of then a new build is triggered every time a code change is pushed WORKFLOW_JOB_QUEUED × ▼ Start a build under these conditions - optional Filter 1 Type REPOSITORY_NAME Pattern	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.
ベントを指定できます。また、REPOSI ポジトリからのウェブフックイベントで Webhook event filter groups A build is triggered if any filter group evaluates to true, which o Filter group 1 Event type - optional Add one or more webhook event filter groups to specify which of then a new build is triggered every time a code change is pushed WORKFLOW_JOB_QUEUED × ▼ Start a build under these conditions - optional Filter 1 Type REPOSITORY_NAME Pattern repository-name	TORY_NAME をフィルタとして指定して、特定 でのみ、ビルドをトリガーすることもできます occurs when all the filters in the group evaluate to true. Remove filter group events trigger a new build. If you do not add a webhook event filter group ed to your repository.

イベントタイプを WORKFLOW_JOB_QUEUED に設定して、セルフホスト型 GitLab ランナー を設定することもできます。詳細については、「<u>のセルフマネージド GitLab ランナー</u> AWS CodeBuild」を参照してください。

3. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。

GitLab グループウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してグループウェブフックイベントを AWS CodeBuild フィルタリングするには、プロジェクトの ScopeConfigurationプロパティを使用します。グ ループ GitLab ウェブフックの詳細については、「<u>GitLab グループウェブフック</u>」を参照してくださ い。

AWS CloudFormation テンプレートの次の YAML 形式の部分は、4 つのフィルターグループを作成 します。1 つまたはすべてが true と評価されると、これらが一緒になってビルドをトリガーしま す。

- 最初のフィルタグループでは、アカウント ID 12345 を持たない GitLab ユーザーが、正規表現 ^refs/heads/main\$ と一致する Git 参照名を持つブランチに対してプルリクエストを作成また は更新することを指定します。
- 2番目のフィルターグループでは、正規表現 READ_ME に一致する Git 参照名を持つブランチで正 規表現 ^refs/heads/.*に一致する名前のファイルに対してプッシュリクエストが作成されるこ とを指定します。
- 3番目のフィルタグループでは、正規表現 \[CodeBuild\] に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。
- 4番目のフィルタグループは、正規表現 \[CI-CodeBuild\] に一致する CI/CD パイプライン名 を持つ GitLab CI/CD パイプラインジョブリクエストを指定します。

```
CodeBuildProject:

Type: AWS::CodeBuild::Project

Properties:

Name: MyProject

ServiceRole: service-role

Artifacts:

Type: NO_ARTIFACTS

Environment:

Type: LINUX_CONTAINER
```

```
ComputeType: BUILD_GENERAL1_SMALL
  Image: aws/codebuild/standard:5.0
Source:
  Type: GITLAB
  Location: source-location
Triggers:
 Webhook: true
  ScopeConfiguration:
    Name: group-name
    Scope: GITLAB_GROUP
  FilterGroups:
    - - Type: EVENT
        Pattern: PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED
      - Type: BASE_REF
        Pattern: ^refs/heads/main$
        ExcludeMatchedPattern: false
      - Type: ACTOR_ACCOUNT_ID
        Pattern: 12345
        ExcludeMatchedPattern: true
    - - Type: EVENT
        Pattern: PUSH
      - Type: HEAD_REF
        Pattern: ^refs/heads/.*
      - Type: FILE_PATH
        Pattern: READ ME
        ExcludeMatchedPattern: true
    - - Type: EVENT
        Pattern: PUSH
      - Type: COMMIT_MESSAGE
        Pattern: \[CodeBuild\]
      - Type: FILE_PATH
        Pattern: ^src/.+|^test/.+
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
      - Type: WORKFLOW_NAME
        Pattern: \[CI-CodeBuild\]
```

GitLab 手動ウェブフック

手動 GitLab ウェブフックを設定して、CodeBuild が GitLab 内でウェブフックを自動的に作成し ようとしないようにできます。CodeBuild は、ウェブフックを作成するための呼び出しの一部と して にペイロード URL を返し、GitLab 内でウェブフックを手動で作成するために使用できま す。CodeBuild が GitLab アカウントにウェブフックを作成することを許可リストに登録されていな い場合でも、ビルドプロジェクトのウェブフックを手動で作成できます。

GitLab 手動ウェブフックを作成するには、次の手順に従います。

GitLab 手動ウェブフックを作成するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
 - [Source (ソース)] で、次のようにします。
 - [ソースプロバイダ] で [GitLab] を選択します。
 - Repository で、GitLab アカウントの Repository を選択します。
 - [リポジトリの URL] に、「https://gitlab.com/user-name/repository-name」と 入力します。
 - [プライマリソースのウェブフックイベント] の場合:
 - [ウェブフック オプション] で、[コードの変更がこのレポジトリにプッシュされるたびに再 ビルド] を選択します。
 - 追加設定を選択し、手動作成 オプションで、GitLab コンソールでこのリポジトリのウェ ブフックを手動で作成を選択します。
- デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。[ペイロード URL]
 と [シークレット] 値は後で使用するため、メモしておきます。
- で GitLab コンソールを開きhttps://gitlab.com/user-name/repository-name/-/ hooks、新しいウェブフックの追加を選択します。
 - URLには、前にメモしたペイロード URL 値を入力します。
 - シークレットトークンには、前にメモしたシークレット値を入力します。
 - CodeBuild にウェブフックペイロードを送信する個々のイベントを設定します。トリガー では、プッシュイベント、マージリクエストイベント、リリースイベント、ジョブイベントのいずれかを選択します。 CodeBuild でサポートされているイベントタイプの詳細については、「GitLab ウェブフックイベント」を参照してください。
- 5. [ウェブフックを追加]を選択します。

GitLab ウェブフックイベント

ウェブフックフィルタグループを使用して、ビルドをトリガーする GitLab ウェブフックイベントを 指定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定で きます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされま す。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグ ループを作成する際、以下を指定します。

イベント

GitLab では、次のイベントのうち、1 つ以上を選択できます: PUSH、PULL_REQUEST_CREATED、PULL_REQUEST_UPDATED、PULL_REQUEST_MERGED、PULL_REQ

ウェブフックのイベントタイプは、X-GitLab-Event フィールドのヘッダーに含まれて います。次の表に、X-GitLab-Event ヘッダー値がイベントタイプにマッピングされる 方法を示します。Merge Request Hook ウェブフックイベントの場合、ペイロードの object_atttributes.action にはマージリクエストタイプに関する追加情報が含まれます。

X-GitLab-Event ヘッ ダー値	object_atttributes .action	イベントタイプ
Push Hook	該当なし	PUSH
Merge Request Hook	open	PULL_REQUEST_CREATED
Merge Request Hook	更新	PULL_REQUEST_UPDATED
Merge Request Hook	merge	PULL_REQUEST_MERGED
Merge Request Hook	再オープンする	PULL_REQUEST_REOPE NED
Merge Request Hook	close	PULL_REQUEST_CLOSED
Release Hook	create、update	RELEASED
Job Hook	該当なし	WORKFLOW_JOB_QUEUED

PULL_REQUEST_MERGED の場合、プルリクエストがスカッシュ戦略とマージされ、プルリク エストブランチが閉じられると、元のプルリクエストコミットは存在しなくなります。この場 合、CODEBUILD_WEBHOOK_MERGE_COMMIT 環境変数には、圧縮されたマージコミットの識別子 が含まれます。

1つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

GitLab アカウント ID が正規表現パターンと一致すると、ビルドがウェブフックイベントで トリガーされます。この値は、ウェブフックフィルタペイロードの actor オブジェクトの account_id プロパティに表示されます。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると (refs/heads/branch-name と refs/tags/tag-name など)、ウェブフックイベントによってビルドがトリガーされま す。HEAD_REF フィルタは、ブランチまたはタグについて Git 参照名を評価します。ブランチ 名またはタグ名は、ウェブフックペイロードの push オブジェクトにある、new オブジェク トの name フィールドに表示されます。プルリクエストイベントの場合、ブランチ名はウェブ フックペイロードの source オブジェクトにある、branch オブジェクトの name フィールド に表示されます。

BASE_REF

基本参照が正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされま す。BASE_REF フィルタは、プルリクエストイベントでのみ使用できます (例: refs/heads/ branch-name)。BASE_REF フィルタは、ブランチの Git 参照名を評価します。ブランチ名 は、ウェブフックペイロードの destination オブジェクトにある、branch オブジェクト の name フィールドに表示されます。

FILE_PATH

変更されたファイルのパスが正規表現パターンに一致すると、ビルドが Webhook イベントで トリガーされます。

COMMIT_MESSAGE

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをトリ ガーします。

WORKFLOW_NAME

ワークフロー名が正規表現パターンに一致する場合に、ウェブフックはビルドをトリガーしま す。

1 Note

ウェブフックペイロードは、GitLab リポジトリのウェブフック設定で見つかります。

トピック

- GitLab ウェブフックイベントのフィルタリング (コンソール)
- GitLab ウェブフックイベントのフィルタリング (SDK)
- GitLab ウェブフ<u>ックイベントのフィルタリング</u> (AWS CloudFormation)

GitLab ウェブフックイベントのフィルタリング (コンソール)

以下の手順に従って、 を使用してウェブフックイベント AWS Management Console をフィルタリ ングします。GitLab ウェブフックイベントの詳細については、「<u>GitLab ウェブフックイベント</u>」を 参照してください。

- 1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を 選択します。
- 2. [イベントタイプ] から、1 つ以上のイベントを選択します。
- イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する]で、1つ以上のオプションフィルタを追加します。
- 4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始 しない] で、1 つ以上のオプションフィルタを追加します。
- 5. 別のフィルタグループを追加するには、[フィルタグループの追加]を選択します。

詳細については、「AWS CodeBuild API リファレンス」の「<u>ビルドプロジェクトの作成 (コンソー</u> ル)」および「WebhookFilter」を参照してください。

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーしま す。

GitLab ウェブフックイベント

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.



- Don't start a build under these conditions optional

2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリ ガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ に一致する Git 参照と ^refs/ heads/branch1! に一致するヘッド参照を含むブランチで作成または更新されたプルリクエスト を指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/branch1\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

vent type dd one or more webhook event filter groups to specify which events trigger a new build. If you do not a nen a new build is triggered every time a code change is pushed to your repository. PULL_REQUEST_CREATED X PULL_REQUEST_UPDATED X Start a build under these conditions - optional ilter 1 ype HEAD_REF attern	dd a webhook event filter group. Add filter
PULL_REQUEST_CREATED × PULL_REQUEST_UPDATED × Start a build under these conditions - optional ilter 1 ype HEAD_REF attern	Add filter
▼ PULL_REQUEST_CREATED × PULL_REQUEST_UPDATED × Start a build under these conditions - optional ilter 1 ype HEAD_REF attern	Add filter
PULL_REQUEST_CREATED × PULL_REQUEST_UPDATED × Start a build under these conditions - optional ilter 1 ype HEAD_REF attern	Add filter
Start a build under these conditions - optional ilter 1 ype HEAD_REF attern	Add filter
 Start a build under these conditions - optional ilter 1 ype HEAD_REF attern 	Add filter
ilter 1 ype HEAD_REF ▼ attern	
ype HEAD_REF ▼ attern	
HEAD_REF attern	
attern	
^refs/heads/branch1\$	
Remove	
ilter 2	
уре	
BASE_REF 🔹	
attern	
^refs/heads/main	
Remove	
Don't start a build under these conditions - optional	
	[
ilter group 2	Remove filter group
vent type	dd a wabback avant filter aroun
the on more webhook event fitter groups to specify which events trigger a new build. If you do not a new build is triggered every time a code change is pushed to your repository.	uu a webnook event niter group
•	
PUSH X	

この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビ ルドをトリガーします。

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	a webhook event filter group,
•	
PUSH X PULL_REQUEST_CREATED X PULL_REQUEST_UPDATED X	
PULL_REQUEST_MERGED ×	
Start a build under these conditions - optional	
Don't start a build under these conditions - optional	Add filter
Filter 1	
Туре	
HEAD_REF	
Pattern	
^refs/tags/.*	

この例では、ウェブフックフィルタグループは、正規表現 ^buildspec.* に一致する名前のファイ ルが変更された場合にのみビルドをトリガーします。

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	a webhook event filter group,
PUSH X	
Start a build under these conditions - optional	Add filter
Filter 1	
Туре	
FILE_PATH	
Pattern	
^buildspec.*	
Remove	

Don't start a build under these conditions - optional

この例で、Webhook フィルターグループは、ファイルが src または test フォルダーで変更された 場合にのみ、ビルドをトリガーします。

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	a webhook event filter group,
PUSH X	
Start a build under these conditions - optional	Add filter
Filter 1	
Туре	
FILE_PATH	
Pattern	
^src/.+ ^test/.+	
Remove	

Don't start a build under these conditions - optional

この例では、正規表現 actor-account-id と一致するアカウント ID を持たない GitLab ユーザー が変更を行った場合にのみ、ウェブフックフィルタグループがビルドをトリガーします。

Note

GitLab アカウント ID の検索方法については、「https://api.github.com/users/*user-name*」 を参照してください。ここで、*user-name* は、GitLab のユーザー名を表します。

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not ad then a new build is triggered every time a code change is pushed to your repository.	d a webhook event filter group,
PUSH ×	
Start a build under these conditions - optional	Add filter
Filter 1	
Туре	
ACTOR_ACCOUNT_ID	
Pattern	
actor-account-id	
Remove	

Don't start a build under these conditions - *optional*

この例では、HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1	Remove filter group
Event type Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add then a new build is triggered every time a code change is pushed to your repository.	d a webhook event filter group,
•	
PUSH X	
Start a build under these conditions - optional	Add filter
Filter 1	
Туре	
COMMIT_MESSAGE	
Pattern	
\[CodeBuild]\	
Remove	

Don't start a build under these conditions - *optional*

GitLab ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、

CreateWebhookまたは UpdateWebhook API メソッドのリクエスト構文で filterGroupsフィー ルドを使用します。詳細については、CodeBuild API リファレンスの「<u>WebhookFilter</u>」を参照して ください。

GitLab ウェブフックイベントの詳細については、「<u>GitLab ウェブフックイベント</u>」を参照してくだ さい。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリ クエスト構文に挿入します。

"filterGroups": [

```
[
    {
        "type": "EVENT",
        "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    }
]
```

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するに は、pattern パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定しま す。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されると トリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main\$ に一致する Git 参照と ^refs/ heads/myBranch\$ に一致するヘッド参照を含むブランチで作成または更新されたプルリクエス トを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/myBranch\$ に一致する Git 参照を含む ブランチでプッシュリクエストを指定します。

```
"filterGroups": [
 Ε
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
 ],
 Ε
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
```

]

```
"pattern": "^refs/heads/myBranch$"
}
```

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定する ことができます。この例では、ビルドは、タグイベントを除くすべてのリクエストに対してトリガー されます。

アカウント ID actor-account-id を持つ GitLab ユーザーによって変更が行われた場合にのみビル ドをトリガーするフィルタを作成できます。

1 Note

GitLab アカウント ID の検索方法については、「https://api.github.com/users/*user-name*」 を参照してください。ここで、*user-name* は、GitLab のユーザー名を表します。

```
ユーザーガイド
```

```
"type": "ACTOR_ACCOUNT_ID",
    "pattern": "actor-account-id"
    }
]
```

引数 pattern の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーする フィルタを作成することができます。この例のフィルタグループでは、正規表現 ^buildspec . * に 一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
 [
 {
    "type": "EVENT",
    "pattern": "PUSH"
 },
 {
    "type": "FILE_PATH",
    "pattern": "^buildspec.*"
 }
]
]
```

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にの み、ビルドをトリガーするように指定しています。

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーする フィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミット

メッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
  [
     [
          "type": "EVENT",
          "pattern": "PUSH"
     },
     {
          "type": "COMMIT_MESSAGE",
          "pattern": "\[CodeBuild\]"
     }
  ]
]
```

GitLab ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、 AWS CodeBuild プロジェクトの FilterGroupsプロパティを使用します。GitLab ウェブフックイ ベントの詳細については、「GitLab ウェブフックイベント」を参照してください。

以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが 作成されます。また、一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルタグループでは、アカウント ID 12345 を持たない GitLab ユーザーが、正規表現 ^refs/heads/main\$ と一致する Git 参照名を持つブランチに対してプルリクエストを作成また は更新することを指定します。
- 2番目のフィルタグループでは、正規表現 ^refs/heads/.* と一致する Git 参照名を持つブラン
 チに対するプッシュリクエストを作成することを指定します。
- 3番目のフィルタグループでは、正規表現 \[CodeBuild\] に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。
- ・ 4 番目のフィルタグループは、正規表現 ∖[CI-CodeBuild∖] に一致するワークフロー名を持つ GitHub Actions ワークフロージョブリクエストを指定します。

CodeBuildProject: Type: AWS::CodeBuild::Project Properties:

```
Name: MyProject
ServiceRole: service-role
Artifacts:
  Type: NO_ARTIFACTS
Environment:
  Type: LINUX_CONTAINER
  ComputeType: BUILD_GENERAL1_SMALL
  Image: aws/codebuild/standard:5.0
Source:
  Type: GITLAB
  Location: source-location
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED
      - Type: BASE_REF
        Pattern: ^refs/heads/main$
        ExcludeMatchedPattern: false
      - Type: ACTOR_ACCOUNT_ID
        Pattern: 12345
        ExcludeMatchedPattern: true
    - - Type: EVENT
        Pattern: PUSH
      - Type: HEAD REF
        Pattern: ^refs/heads/.*
    - - Type: EVENT
        Pattern: PUSH
      - Type: COMMIT_MESSAGE
        Pattern: \[CodeBuild\]
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
      - Type: WORKFLOW_NAME
        Pattern: \[CI-CodeBuild\]
```

Buildkite 手動ウェブフック

現在、CodeBuild では、すべての Buildkite ウェブフックを手動で作成する必要がありま す。CodeBuild は、ウェブフックを作成するための呼び出しの一部としてペイロード URL を返しま す。ウェブフックは、Buildkite 内でウェブフックを手動で作成するために使用できます。

Buildkite 手動ウェブフックを作成するには、次の手順に従います。

ウェブフックを使用して CodeBuild プロジェクトを作成するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ビルドプロジェクトを作成します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u>ル)」および「ビルドの実行 (コンソール)」を参照してください。
- 3. プロジェクト設定で、Runner プロジェクトを選択します。

Runner の場合:

- Runner プロバイダーで、Buildkite を選択します。
- Buildkite エージェントトークンで、シークレットの作成ページを使用して新しいエージェントトークンを作成するを選択します。上記で生成した Buildkite エージェントトークンと等しいシークレット値を使用して、AWS Secrets Manager で新しいシークレットを作成するように求められます。
- (オプション)ジョブに CodeBuild マネージド認証情報を使用する場合は、Buildkite ソース認 証情報オプションでジョブのソースリポジトリプロバイダーを選択し、アカウントに認証情 報が設定されていることを確認します。さらに、Buildkite パイプラインが HTTPS を使用した チェックアウトを使用していることを確認します。
- 4. [環境] で以下の操作を行います。
 - サポートされている [環境イメージ] と [コンピューティング] を選択します。GitHub Actions ワークフロー YAML のラベルを使用して、イメージとインスタンスの設定を上書きするオ プションがあることに注意してください。詳細については、「ステップ 2: GitHub Actions ワークフロー YAML を更新」を参照してください
 - [Buildspec (Buildspec)] で、次のようにします。
 - buildspec-override:true がラベルとして追加されない限り、buildspec は無視される ことに注意してください。代わりに、CodeBuild は、セルフホスト型ランナーを設定するコ マンドを使用するように上書きします。
- 5. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。
- ペイロード URL とシークレット値をウェブフックの作成ポップアップから保存します。ポップ アップの指示に従って、新しい Buildkite 組織のウェブフックを作成します。

でビルドプロジェクトの詳細を表示する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、または AWS SDKs を使用して、CodeBuild でビルドプロ ジェクトの詳細を表示できます。

トピック

- ・ ビルドプロジェクトの詳細を表示する (コンソール)
- ・ <u>ビルドプロジェクトの詳細を表示する (AWS CLI)</u>
- ・ ビルドプロジェクトの詳細を表示する (AWS SDK)

ビルドプロジェクトの詳細を表示する (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Build projects] を選択します。

Note

デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多 くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

- 3. ビルドプロジェクトのリストの[名前]列で、ビルドプロジェクトのリンクを選択します。
- 4. [ビルドプロジェクト: project-name] ページで、[ビルドの詳細] を選択します。

ビルドプロジェクトの詳細を表示する (AWS CLI)

batch-get-projects コマンドを実行します。

aws codebuild batch-get-projects --names names

上記のコマンドで、次のプレースホルダを置き換えます。

• *names*: 詳細を表示する 1 つ以上のビルドプロジェクト名を示すのに必要な文字列。複数のビルド プロジェクトを指定するには、各ビルドプロジェクトの名前をスペースで区切ります。最大 100 のビルドプロジェクト名を指定できます。ビルドプロジェクトのリストを表示するには、「ビルド プロジェクト名の一覧表示 (AWS CLI)」を参照してください。

たとえば、次のコマンドを実行するとします。

aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2
my-other-demo-project

次のような結果が出力に表示されます。省略記号 (...) は簡潔にするために省略されたデータを表 すのに使用されます。

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
       . . .
      "name": codebuild-demo-project,
      . . .
    },
    {
       . . .
      "name": codebuild-demo-project2",
    }
  ]
}
```

上記の出力では、指定されたビルドプロジェクト名はすべて projectsNotFound 配列にリストさ れていますが、情報は見つかりませんでした。projects 配列は、情報が見つかった各ビルドプロ ジェクトの詳細を示しています。ビルドプロジェクトの詳細は、簡潔にするために前の出力から省略 されています。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」の出力を参照してくだ さい。

batch-get-projects コマンドは、特定のプロパティ値のフィルタリングをサポートしていませんが、 プロジェクトのプロパティを列挙するスクリプトを記述できます。たとえば、次の Linux シェルスク リプトは、現在のアカウントの現在のリージョンのプロジェクトを列挙し、各プロジェクトで使用さ れるイメージを出力します。

```
#!/usr/bin/sh
# This script enumerates all of the projects for the current account
# in the current region and prints out the image that each project is using.
imageName=""
function getImageName(){
 local environmentValues=(${1//$'\t'/ })
 imageName=${environmentValues[1]}
}
function processProjectInfo() {
 local projectInfo=$1
 while IFS=$'\t' read -r section value; do
   if [[ "$section" == *"ENVIRONMENT"* ]]; then
     getImageName "$value"
   fi
 done <<< "$projectInfo"</pre>
}
# Get the list of projects.
projectList=$(aws codebuild list-projects --output=text)
for projectName in $projectList
do
 if [[ "$projectName" != *"PROJECTS"* ]]; then
   # Get the detailed information for the project.
   projectInfo=$(aws codebuild batch-get-projects --output=text --names
 "$projectName")
   processProjectInfo "$projectInfo"
   printf 'Project "%s" has image "%s"\n' "$projectName" "$imageName"
 fi
done
```

AWS CLI で を使用する方法の詳細については AWS CodeBuild、「」を参照してください<u>コマンド</u> <u>ラインリファレンス</u>。

ビルドプロジェクトの詳細を表示する (AWS SDK)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> ツールのリファレンス。 AWS SDKs

でビルドプロジェクト名を表示する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、または AWS SDKs を使用して、CodeBuild のビルドプロ ジェクトのリストを表示できます。

トピック

- ビルドプロジェクト名の一覧表示 (コンソール)
- ・ビルドプロジェクト名の一覧表示 (AWS CLI)
- ・ビルドプロジェクト名の一覧表示 (AWS SDK)

ビルドプロジェクト名の一覧表示 (コンソール)

コンソールで、 AWS リージョンのビルドプロジェクトのリストを表示できます。この情報には、名 前、ソースプロバイダー、リポジトリ、最新のビルドステータス、説明 (ある場合) が含まれます。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. ナビゲーションペインで、[Build projects] を選択します。

Note

デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多 くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

ビルドプロジェクト名の一覧表示 (AWS CLI)

list-projects コマンドを実行します。

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-token next-token
```

ビルドプロジェクトの詳細を表示する (AWS SDK)

上記のコマンドで、次のプレースホルダを置き換えます。

- sort-by: ビルドプロジェクト名を一覧表示するために使用する条件を示すためのオプションの文字列。有効な値を次に示します。
 - CREATED_TIME: 各ビルドプロジェクトがいつ作成されたかに基づいて、ビルドプロジェクト名 を一覧表示します。
 - LAST_MODIFIED_TIME: 各ビルドプロジェクトに関する情報が最後に変更されたときに基づいてビルドプロジェクト名を一覧表示します。
 - NAME 各ビルドプロジェクト名に基づいて、ビルドプロジェクト名を一覧表示します。
- sort-order: ビルドプロジェクトのリストを表示するためのオプションの文字列。sort-by に 基づく。有効な値は、ASCENDING および DESCENDING です。
- next-token: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

aws codebuild list-projects --sort-by NAME --sort-order ASCENDING

次のような結果が出力に表示されることがあります。



このコマンドをもう一度実行します。

aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

ビルドプロジェクト名の一覧表示 (AWS CLI)
次のような結果が出力に表示されることがあります。

```
{
   "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
]
}
```

ビルドプロジェクト名の一覧表示 (AWS SDK)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> ツールのリファレンス。 AWS SDKs

でのビルド AWS CodeBuild

ビルドは、一連の入力アーティファクト (Java クラスファイルのコレクションなど) に基づいて出力 アーティファクト (JAR ファイルなど) を作成 AWS CodeBuild するために によって実行される一連 のアクションを表します。

複数のビルドを実行するときは、以下のルールが適用されます。

- 可能であれば、ビルドが同時に実行されます。同時実行ビルドの最大数は変化する可能性があります。詳細については、「」を参照してくださいのクォータ AWS CodeBuild
- ビルドプロジェクトに同時ビルド制限が設定されている場合、実行中のビルド数がプロジェクトの 同時ビルド制限に達すると、ビルドがエラーを返します。詳細については、「<u>同時ビルド制限を有</u> 効にする」を参照してください。
- ビルドプロジェクトに同時ビルド制限が設定されていない場合、実行中のビルド数がプラット フォームとコンピューティングタイプの同時ビルド制限に達すると、ビルドがキューに入れられま す。キュー内のビルドの最大数は、同時ビルド制限の5倍です。詳細については、「」を参照し てくださいのクォータ AWS CodeBuild

タイムアウト値で指定された時間 (分) が経過しても開始されないキュー内のビルドは、キューか ら削除されます。デフォルトのタイムアウト値は 8 時間です。ビルドを実行するとき、5 分~ 8 時間の値でビルドのキュータイムアウトをオーバーライドできます。詳細については、「<u>AWS</u> CodeBuild ビルドを手動で実行する」を参照してください。

キューに入れられたビルドが開始される順序を予測することはできません。

Note

ビルドの履歴には、1年間アクセスできます。

ビルドを操作するときに、次のタスクを実行できます。

トピック

- AWS CodeBuild ビルドを手動で実行する
- AWS Lambda コンピューティングでビルドを実行する
- リザーブドキャパシティキャパシティフリートでビルドを実行

- ビルドをバッチで実行
- バッチビルドで並列テストを実行する
- パフォーマンスを向上させるためのキャッシュビルド
- でのビルドのデバッグ AWS CodeBuild
- でビルドを削除する AWS CodeBuild
- でビルドを手動で再試行する AWS CodeBuild
- ・ でビルドを自動的に再試行する AWS CodeBuild
- でビルドを停止する AWS CodeBuild
- でバッチビルドを停止する AWS CodeBuild
- AWS CodeBuild ビルドを自動的にトリガーする
- でビルドの詳細を表示する AWS CodeBuild
- でビルド IDsのリストを表示する AWS CodeBuild
- AWS CodeBuildでビルドプロジェクトのビルド ID を一覧表示する

AWS CodeBuild ビルドを手動で実行する

CodeBuild でビルドを実行するには、 AWS CodeBuild コンソール AWS CLI、、または AWS SDKs を使用できます。

トピック

- AWS CodeBuild エージェントを使用してビルドをローカルで実行する
- ビルドの実行 (コンソール)
- ビルドの実行 (AWS CLI)
- バッチビルドの実行 (AWS CLI)
- ・ ビルドの実行の自動開始 (AWS CLI)
- ビルドの実行の自動停止 (AWS CLI)
- ビルドを実行する (AWS SDKs)

AWS CodeBuild エージェントを使用してビルドをローカルで実行する

AWS CodeBuild エージェントを使用して、ローカルマシンで CodeBuild ビルドを実行できます。x86_64 および ARM プラットフォームで使用できるエージェントがあります。

通知にサブスクライブして、 エージェントの新しいバージョンがリリースされたときに通知を受信 できます。

前提条件

開始する前に、以下を実行する必要があります。

- ・ ローカルマシンで Git をインストールします。
- ・ ローカルマシンで、<u>Docker</u> をインストールしてセットアップします。

ビルドイメージの設定方法

ビルドイメージを設定する必要があるのは、エージェントを初めて実行するとき、またはイメージが 変更されたときだけです。

ビルドイメージの設定方法

厳選された Amazon Linux 2 イメージを使用する場合は、次のコマンドを使用して、https://<u>https://gallery.ecr.aws/codebuild/amazonlinux-x86_64-standard</u>.https://www.comのCodeBuild パブリック Amazon ECR リポジトリからイメージをプルできます。

\$ docker pull public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0

その代わりに別の Linux イメージを使用する場合は、以下のステップを実行してください。

a. CodeBuild イメージレポジトリをクローンします。

\$ git clone https://github.com/aws/aws-codebuild-docker-images.git

 b. イメージディレクトリを変更します。この例では、aws/codebuild/standard:5.0 イ メージを使用します。

\$ cd aws-codebuild-docker-images/ubuntu/standard/5.0

c. イメージを構築します。これには数分間かかります。

\$ docker build -t aws/codebuild/standard:5.0 .

2. CodeBuild エージェントをダウンロードします。

エージェントの x86_64 バージョンをダウンロードするには、次のコマンドを実行します。

\$ docker pull public.ecr.aws/codebuild/local-builds:latest

次のコマンドを使用して、ARM バージョンのエージェントをダウンロードしてインストールし ます。

\$ docker pull public.ecr.aws/codebuild/local-builds:aarch64

3. CodeBuild エージェントは、https://gallery.ecr.aws/codebuild/local-builds から入手できます。

エージェントの x86_64 バージョンのセキュアハッシュアルゴリズム (SHA) 署名は次のとおり です。

sha256:ccb19bdd7af94e4dc761e4c58c267e9455c28ec68d938086b4dc1cf8fe6b0940

エージェントの ARM バージョンの SHA 署名は次のとおりです。

sha256:7d7b5d35d2ac4e062ae7ba8c662ffed15229a52d09bd0d664a7816c439679192

SHA を使用してエージェントのバージョンを識別できます。エージェントの SHA 署名を表示す るには、次のコマンドを実行して、RepoDigests の下で SHA を探します。

\$ docker inspect public.ecr.aws/codebuild/local-builds:latest

CodeBuild エージェントを実行する

CodeBuild エージェントを実行するには

- 1. ビルドプロジェクトソースを含むディレクトリに移動します。
- 2. codebuild.sh スクリプトをダウンロードします。

\$ curl -0 https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/
master/local_builds/codebuild_build.sh
\$ chmod +x codebuild_build.sh

codebuild_build.sh スクリプトを実行し、コンテナイメージおよび出力ディレクトリを指定します。

x86_64 ビルドを実行するには、次のコマンドを実行します。

\$./codebuild_build.sh -i <container-image> -a <output directory>

ARM ビルドを開始するには、次のコマンドを実行します。

\$./codebuild_build.sh -i <container-image> -a <output directory> -l
public.ecr.aws/codebuild/local-builds:aarch64

<container-image>は、コンテナイメージの名前 (aws/codebuild/standard:5.0 または public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0 など) に置き換えて ください。

スクリプトはビルドイメージを起動し、現在のディレクトリにあるプロジェクトを使用して ビルドを実行します。ビルドプロジェクトの場所を指定するには、-s *<build project directory>* オプションをスクリプトコマンドに追加します。

CodeBuild エージェントの新しいバージョンに関する通知の受信

Amazon SNS 通知をサブスクライブして、 AWS CodeBuild エージェントの新しいバージョンがリ リースされたときに通知を受け取ることができます。

CodeBuild エージェントの通知にサブスクライブするには

- 1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
- ナビゲーションバーで、まだ選択されていない場合は、 AWS リージョンを米国東部 (バージニ ア北部) に変更します。サブスクライブする Amazon SNS 通知がこの AWS リージョンで作成 されるため、このリージョンを選択する必要があります。
- 3. ナビゲーションペインで [Subscriptions] を選択してください。
- 4. [Create subscription] を選択します。
- 5. [Create subscription] (サブスクリプションの作成) で、次の操作を行います。
 - a. [Topic ARN] (トピック ARN) で、以下の Amazon リソースネーム (ARN) を使用します。

arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates

- b. [プロトコル] で、[E メール] または [SMS] を選択します。
- c. [エンドポイント] で、通知を受信する場所 (E メールまたは SMS) を選択します。E メー ル、住所、または電話番号 (市外局番を含む) を入力します。
- d. [Create subscription] (サブスクリプションの作成) を選択します。
- e. [Email] (E メール) を選択した場合は、サブスクリプションの確認を求める E メールが届き ます。E メールの指示に従ってサブスクリプションを完了します。

通知が不要になった場合は、次の手順で受信登録を解除します。

CodeBuild エージェントの通知のサブスクリプションを解除するには

- 1. Amazon SNS コンソール(https://console.aws.amazon.com/sns/v3/home)を開きます。
- 2. ナビゲーションペインで [Subscriptions] (サブスクリプション) を選択します。
- サブスクリプションを選択し、[Actions] (アクション) から [Delete subscriptions] (サブスクリプ ションの削除) を選択します。確認を求められたら [Delete] (削除) を選択します。

ビルドの実行 (コンソール)

AWS CodePipeline を使用して CodeBuild でビルドを実行するには、以下の手順をスキップし、「」の手順に従いますCodePipeline で CodeBuild を使用。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Build projects] を選択します。
- 3. ビルドプロジェクトのリストで、ビルドプロジェクトを選択します。
- デフォルトのビルドプロジェクト設定でビルドを実行することも、このビルドのみのビルド設定 を上書きすることもできます。
 - a. デフォルトのビルドプロジェクト設定を使用してビルドを実行するには、[ビルドの開始] を 選択します。ビルドがすぐに開始されます。
 - b. デフォルトのビルドプロジェクト設定を上書きする場合は、[上書きでビルドを開始] を選択 します。[ビルドを開始] ページで、以下を上書きできます。

- [ビルド設定]
- ・ソース
- [環境変数の上書き]

より高度な上書きを選択する必要がある場合は、[高度なビルドの上書き]を選択します。こ のページでは、以下の操作を上書きできます。

- [ビルド設定]
- ・ソース
- 環境
- Buildspec
- アーティファクト
- ・ログ

上書きを選択したら、[ビルドを開始]を選択します。

このビルドの詳細については、「ビルドの詳細の表示 (コンソール)」を参照してください。

ビルドの実行 (AWS CLI)

Note

CodePipeline を使用してビルドを実行するには AWS CodeBuild、以下のステップをスキッ プし、「」の手順に従います<u>CodeBuild を使用するパイプラインの作成 (AWS CLI)</u>。 CodeBuild AWS CLI で を使用する方法の詳細については、「」を参照してください<u>コマン</u> <u>ドラインリファレンス</u>。

1. 次のいずれかの方法で start-build コマンドを実行します。

aws codebuild start-build --project-name <project-name>

ビルド入力アーティファクトの最新バージョンとビルドプロジェクトの既存の設定を使用するビ ルドを実行する場合は、これを使用します。 aws codebuild start-build --generate-cli-skeleton

以前のバージョンのビルド入力アーティファクトを使用してビルドを実行する場合、またはビル ド出力アーティファクト、環境変数、ビルド仕様、またはデフォルトのビルドタイムアウト期間 の設定をオーバーライドする場合は、これを使用します。

- --project-name オプションを指定して start-build コマンドを実行する場合は、<projectname> をビルドプロジェクトの名前に置き換えて、この手順のステップ6に進みます。ビルド プロジェクトのリストを表示するには、「ビルドプロジェクト名を表示」を参照してください。
- --idempotency-token オプションを指定して start-build コマンドを実行すると、大文字と小 文字を区別する一意の識別子 (トークン) が start-build リクエストに含まれます。このトー クンは、 リクエスト後 5 分間有効です。同じトークンで start-build リクエストを繰り返し 行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。
- start-build オプションを指定して --generate-cli-skeleton コマンドを実行すると、出力に JSON 形式のデータが表示されます。 AWS CLI がインストールされているローカルコン ピュータまたはインスタンス上の場所にあるファイル (などstart-build.json)にデータを コピーします。コピーしたデータを次の形式に変更して、結果を保存します。

```
{
  "projectName": "projectName",
  "sourceVersion": "sourceVersion",
  "artifactsOverride": {
    "type": "type",
    "location": "location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifactsOverride-name",
    "packaging": "packaging"
 },
  "buildspecOverride": "buildspecOverride",
  "cacheOverride": {
    "location": "cacheOverride-location",
    "type": "cacheOverride-type"
 },
  "certificateOverride": "certificateOverride",
  "computeTypeOverride": "computeTypeOverride",
  "environmentTypeOverride": "environmentTypeOverride",
  "environmentVariablesOverride": {
    "name": "environmentVariablesOverride-name",
```



次のプレースホルダーを置き換えます。

- projectName: 必須の文字列。このビルドに使用するビルドプロジェクトの名前。
- sourceVersion: オプションの文字列。作成するソースコードのバージョンで、次のように なります。
 - Amazon S3 の場合、ビルドする入力 ZIP ファイルのバージョンに対応するバージョン
 ID。sourceVersion が指定されなければ、最新のバージョンが使用されます。
 - CodeCommit の場合、ビルドするソースコードのバージョンに対応するコミット
 ID。sourceVersion が指定されなければ、デフォルトブランチの HEAD コミット ID が使用されます。(sourceVersion にタグ名は指定できません。しかし、タグのコミット ID は指定できます。)
 - GitHubの場合、ビルドするソースコードのバージョンに対応するコミット ID、プルリク エスト ID、ブランチ名、またはタグ名。プルリクエスト ID を指定する場合、pr/pullrequest-ID (例: pr/25) 形式を使用する必要があります。ブランチ名を指定すると、ブラ ンチの HEAD コミット ID が使用されます。sourceVersion が指定されなければ、デフォ ルトブランチの HEAD コミット ID が使用されます。
 - Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ 名、またはタグ名。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されま す。sourceVersion が指定されなければ、デフォルトブランチの HEAD コミット ID が使 用されます。
- 次に示すプレースホルダーは、artifacts0verride が対象です。

- *type*: オプション。このビルドでオーバーライドするビルド出力アーティファクトタイプは、ビルドプロジェクトで定義されたものです。
- location: オプション。このビルドでオーバーライドするビルド出力アーティファクトの 場所は、ビルドプロジェクトで定義されたものです。
- path:オプション。このビルドでオーバーライドするビルド出力アーティファクトパスは、 ビルドプロジェクトで定義されたものです。
- namespaceType: オプション。このビルドでオーバーライドするビルド出力アーティファ クトパスのタイプは、ビルドプロジェクトで定義されたものです。
- name: オプション。このビルドでオーバーライドするビルド出力アーティファクト名は、ビルドプロジェクトで定義されたものです。
- packaging: オプション。このビルドでオーバーライドするビルド出力アーティファクト パッケージタイプは、ビルドプロジェクトで定義されたものです。
- buildspecOverride: オプション。ビルドプロジェクトに定義されている buildspec 宣言を 上書きする、このビルドの buildspec 宣言。この値が設定されている場合は、インラインのビ ルド仕様定義か、組み込みの環境変数 CODEBUILD_SRC_DIR の値に相対的な代替 buildspec ファイルへのパスか、S3 バケットへのパスになります。S3 バケットは、ビルドプロジェク トと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを 指定します(例: arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml)。この値 が指定されていない場合や、空の文字列に設定されている場合、ソースコードのルートディ レクトリに buildspec.yml ファイルが含まれている必要があります。詳細については、 「buildspec ファイル名とストレージの場所」を参照してください。
- 次に示すプレースホルダーは、cacheOverride が対象です。
 - cacheOverride-location: オプション。ビルドプロジェクトで指定された
 ProjectCache オブジェクトを上書きする、このビルドの ProjectCache オブジェクトの場所。cacheOverride はオプションで、ProjectCache オブジェクトを受け取ります。location は ProjectCache オブジェクトで必要です。
 - cacheOverride-type: オプション。ビルドプロジェクトで指定された ProjectCache オブジェクトを上書きする、このビルドの ProjectCache オブジェクトのタイ プ。cacheOverride はオプションで、ProjectCache オブジェクトを受け取りま す。type は ProjectCache オブジェクトで必要です。
- certificateOverride:オプション。ビルドプロジェクトで指定された証明書を上書きする、このビルドの証明書の名前。

- environmentTypeOverride: オプション。ビルドプロジェクトで指定されたコンテナタイプを上書きする、このビルドのコンテナタイプ。現在の有効な文字列は LINUX_CONTAINER です。
- 次に示すプレースホルダーは、environmentVariablesOverrideが対象です。
 - environmentVariablesOverride-name: オプション。このビルドで値を上書きするビ ルドプロジェクトの環境変数の名前。
 - environmentVariablesOverride-type: オプション。このビルドで値を上書きするビ ルドプロジェクトの環境変数のタイプ。
 - environmentVariablesValue:オプション。このビルドで値を上書きするビルドプロ ジェクトで定義された環境変数の値。
- gitCloneDepthOverride: オプション。このビルドで上書きする、ビルドプロジェクトの [Git のクローンの深さ] の値。ソースタイプが Amazon S3 である場合、この値はサポートさ れません。
- imageOverride: オプション。ビルドプロジェクトで指定されたイメージを上書きする、このイメージの名前。
- idempotencyToken: オプション。ビルドリクエストがべき等であることを指定する、トークンとして機能する文字列。64 文字以下の任意の文字列を選択できます。このトークンは、ビルド開始リクエスト後5分間有効です。同じトークンでビルド開始リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。
- insecureSs10verride: ビルドプロジェクトに指定されている安全でない TLS 設定を上書 きするかどうかを指定するブール値 (オプション)。安全でない TLS 設定により、プロジェク トのソースコードに接続するときに TLS 警告を無視するかどうかが決まります。この上書き が適用されるのは、ビルドのソースが GitHub Enterprise Server である場合のみです。
- privilegedModeOverride: オプションのブール値。true に設定すると、ビルドは、ビルド プロジェクトで権限モードを上書きします。
- queuedTimeoutInMinutesOverride: ビルドをキューに入れてからタイムアウトするまでの時間(分)を指定するオプションの整数。その最小値は5分、最大値は480分(8時間)です。
- reportBuildStatusOverride: ビルドの開始と完了のステータスをソースプロバイダ に送信するかどうかを指定するオプションのブール値。これを GitHub、GitHub Enterprise Server、Bitbucket 以外のソースプロバイダーに対して設定すると、invalidInputException が スローされます。

- sourceAuthOverride:オプションの文字列。ビルドプロジェクトで定義された認可タイプ を上書きする、このビルドの認可タイプ。この上書きが適用されるのは、ビルドプロジェクト のソースが Bitbucket または GitHub である場合のみです。
- sourceLocationOverride:オプションの文字列。このビルドで、ビルドプロジェクトで定義されたソースの場所を上書きする場所。
- serviceRoleOverride:オプションの文字列。ビルドプロジェクトで指定されたサービス ロールを上書きする、このビルドのサービスロールの名前。
- sourceTypeOverride:オプションの文字列。このビルドで、ビルドプロジェ クトで定義されたソース入力を上書きするソース入力タイプ。有効な文字列 は、NO_SOURCE、CODECOMMIT、CODEPIPELINE、GITHUB、S3、BITBUCKET、および GITHUB_ENTERPRISEです。
- timeoutInMinutesOverride:オプション番号。このビルドで上書きするビルドタイムアウトの分数は、ビルドプロジェクトで定義されたものです。

アクセスキー ID、 AWS シークレット AWS アクセスキー、パスワードなどの機密性の高い値 を持つ環境変数をパラメータとして Amazon EC2 Systems Manager パラメータストアに保存 することをお勧めします。CodeBuild では、Amazon EC2 Systems Manager パラメータストア に保存されているパラメータは、そのパラメータの名前が /CodeBuild/ (例: /CodeBuild/ dockerLoginPassword) で始まる場合にのみ使用できます。CodeBuild コンソールを使用 して、Amazon EC2 Systems Manager にパラメータを作成することができます。[Create a parameter (パラメータの作成)] を選択し、手順に従います。(ダイアログボックスでは、[KMS キー] の場合、オプションでアカウントの AWS KMS キーの ARN を指定できます。Amazon EC2 Systems Manager では、このキーを使用して、保存中にパラメータの値を暗号化し、取 得中に復号化します。) CodeBuild コンソールを使用してパラメータを作成した場合、コンソー ルは保存されている /CodeBuild/ パラメータを開始します。ただし、Amazon EC2 Systems Manager パラメータストアコンソールを使用してパラメータを作成する場合、パラメータの名 前を /CodeBuild/ で開始する必要があり、[タイプ] を [Secure String (安全な文字列)] に設定 する必要があります。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の 「AWS Systems Manager パラメータストア」および「チュートリアル: String パラメータの作 成とテスト (コンソール)」を参照してください。

ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存されているパ ラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アク ションを許可する必要があります。以前に [アカウントに新しいサービスロールを作成する] を 選択している場合、CodeBuild は、このアクションをビルドプロジェクトのデフォルトのサービ スロールに自動的に含めます。ただし [Choose an existing service role from your account] を選 択した場合は、このアクションをサービスロールに個別に含める必要があります。

既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれていて、other_value の 値を持つ MY_VAR という名前の環境変数を設定した場合、my_value が other_value に置き 換えられます。同様に、Docker イメージに /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に含まれていて、\$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラ ル値 \$PATH:/usr/share/ant/bin に置き換えられます。

CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使 用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合、環境変数の値は次のように決定されま す。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ビルドプロジェクト定義の値が次に優先されます。
- buildspec ファイル宣言の値の優先順位が最も低くなります。

これらのプレースホルダの有効な値の詳細については、「<u>ビルドプロジェクトの作成 (AWS</u> <u>CLI)</u>」を参照してください。ビルドプロジェクトの最新の設定の一覧については、「<u>ビルドプロ</u> ジェクトの詳細を表示」を参照してください。

保存したばかりのファイルがあるディレクトリに移動し、start-build コマンドをもう一度実行します。

aws codebuild start-build --cli-input-json file://start-build.json

6. 成功した場合は、「<u>ビルドを実行するには</u>」の手順で説明されているのと同様のデータが出力に 表示されます。

このビルドの詳細情報を使用するには、出力の id の値を書き留めてから、「<u>ビルドの詳細の表示</u> (AWS CLI)」を参照してください。

バッチビルドの実行 (AWS CLI)

1. 次のいずれかの方法で start-build-batch コマンドを実行します。

aws codebuild start-build-batch --project-name <project-name>

ビルド入力アーティファクトの最新バージョンとビルドプロジェクトの既存の設定を使用するビ ルドを実行する場合は、これを使用します。

aws codebuild start-build-batch --generate-cli-skeleton > <json-file>

以前のバージョンのビルド入力アーティファクトを使用してビルドを実行する場合、またはビル ド出力アーティファクト、環境変数、ビルド仕様、またはデフォルトのビルドタイムアウト期間 の設定をオーバーライドする場合は、これを使用します。

- --project-name オプションを指定して start-build-batch コマンドを実行する場合 は、<project-name> をビルドプロジェクトの名前に置き換えて、この手順のステップ 6 に進 みます。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名を表示」を参照 してください。
- --idempotency-token オプションを指定して start-build-batch コマンドを実行すると、大文 字と小文字を区別する一意の識別子 (トークン) が start-build-batch リクエストに含まれ ます。このトークンは、 リクエスト後 5 分間有効です。同じトークンで start-build-batch リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラー を返します。
- --generate-cli-skeleton オプションを指定して start-build-batch コマンドを実行する と、JSON 形式のデータが <json-file> ファイルに出力されます。このファイルは、startbuild コマンド実行により生成されるスケルトンに似ていますが、次のオブジェクトが追加され ています。共通オブジェクトの詳細については、「ビルドの実行 (AWS CLI)」を参照してくだ さい。

このファイルを変更してビルドオーバーライドを追加し、結果を保存します。

```
"buildBatchConfigOverride": {
    "combineArtifacts": combineArtifacts,
    "restrictions": {
        "computeTypesAllowed": [
            allowedComputeTypes
     ],
```

}

```
"maximumBuildsAllowed": maximumBuildsAllowed
},
"serviceRole": "batchServiceRole",
"timeoutInMins": batchTimeout
```

buildBatchConfigOverride オブジェクトは、<u>ProjectBuildBatchConfig</u> 構造体で、このビル ドのバッチビルド設定の上書きを含んでいます。

combineArtifacts

バッチビルドのビルドアーティファクトを 1 つのアーティファクトの場所に結合するかどう かを指定するブール値。

allowedComputeTypes

バッチビルドで許可されるコンピューティングタイプを指定する文字列の配列。これらの値 については、「ビルド環境のコンピューティングタイプ」を参照してください。

maximumBuildsAllowed

許可されるビルドの最大数を指定します。

batchServiceRole

バッチビルドプロジェクトのサービスロール ARN を指定します。

batchTimeout

バッチビルドを完了するまでの最大時間 (分単位)を指定します。

5. 保存したばかりのファイルがあるディレクトリに移動し、start-build-batch コマンドをも う一度実行します。

aws codebuild start-build-batch --cli-input-json file://start-build.json

成功した場合、<u>BuildBatch</u>の JSON 表現オブジェクトが、コンソール出力に表示されます。このデータの例については、「StartBuildBatch レスポンスの構文」を参照してください。

ビルドの実行の自動開始 (AWS CLI)

ソースコードが GitHub または GitHub Enterprise Server リポジトリに保存されている場合は、コー ド変更がリポジトリにプッシュされるたびにGitHub ウェブフックを使用してソースコードを AWS CodeBuild 再構築できます。

次のように create-webhook コマンドを実行します。

aws codebuild create-webhook --project-name <project-name>

<project-name>は、再ビルドするソースコードを含むビルドプロジェクトの名前です。

GitHub では、次のような情報が出力に表示されます。

```
{
    "webhook": {
        "url": "<url>"
    }
}
```

<url>は GitHub ウェブフックへの URL です。

GitHub Enterprise Server の場合、以下のような情報が出力に表示されます。



- 1. 出力からシークレットキーとペイロード URL をコピーします。これらは、GitHub Enterprise Server に Webhook を追加するために必要となります。
- 2. GitHub Enterprise Server で、CodeBuild プロジェクトが保存されているリポジトリを選択しま す。[設定]、[Hooks & services]、[Add webhook] の順に選択します。
- 3. ペイロード URL とシークレットキーを入力し、その他のフィールドにはデフォルト値を選択して、[Add webhook] を選択します。

ビルドの実行の自動停止 (AWS CLI)

ソースコードが GitHub または GitHub Enterprise Server リポジトリに保存されている場合は、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を AWS CodeBuild 再構築するように GitHub ウェブフックを設定できます。詳細については、「<u>ビルドの実行の自動開始 (AWS CLI)</u>」を 参照してください。

この動作を有効にしている場合、次の delete-webhook コマンドを実行して無効化できます。

aws codebuild delete-webhook --project-name <project-name>

• <project-name>は、再構築するソースコードを含むビルドプロジェクトの名前です。

このコマンドが成功すると、情報やエラーはなにも出力に表示されません。

Note

これは、CodeBuild プロジェクトからのみ webhook を削除します。GitHub または GitHub Enterprise Server でも Webhook を削除する必要があります。

ビルドを実行する (AWS SDKs)

CodePipeline を使用してビルドを実行するには AWS CodeBuild、これらのステップをスキップ し、<u>AWS CodeBuild で AWS CodePipeline を使用してコードをテストし、ビルドを実行する</u>代わり に「」の手順に従います。

SDK で CodeBuild を使用する方法については、「」を参照してください<u>AWS SDKsとツールのリ</u> <u>ファレンス</u>。 AWS SDKs

AWS Lambda コンピューティングでビルドを実行する

AWS Lambda コンピューティングは、ビルドの起動速度を最適化します。 は、起動レイテンシー が低いため、より高速なビルド AWS Lambda をサポートします。 AWS Lambda も自動的にスケー リングするため、ビルドはキュー内で実行を待つことはありません。ただし、 AWS Lambda がサ ポートしていないユースケースがいくつかあり、それらがユーザーに影響を与える場合は EC2 コン ピューティングを使用します。詳細については、「<u>AWS Lambda コンピューティングの制限</u>」を参 照してください。 トピック

- AWS Lambda上で実行される、選別されたランタイム環境の Docker イメージには、どのツールと ランタイムが含まれますか?
- キュレートされたイメージに必要なツールが含まれていない場合はどうなりますか。
- CodeBuild で AWS Lambda コンピューティングをサポートしているリージョンはどれですか?
- AWS Lambda コンピューティングの制限
- CodeBuild Lambda Java AWS SAM で を使用して Lambda 関数をデプロイする
- CodeBuild Lambda Node.js を使用してシングルページの React アプリを作成
- CodeBuild Lambda Python を使用して Lambda 関数の設定を更新

AWS Lambda上で実行される、選別されたランタイム環境の Docker イ メージには、どのツールとランタイムが含まれますか?

AWS Lambda は、次のツールをサポートしています。 AWS CLI v2、 AWS SAM CLI、git、go、Java、Node.js、Python、pip、Ruby、.NET。

キュレートされたイメージに必要なツールが含まれていない場合はどうな りますか。

キュレートされたイメージに必要なツールが含まれていない場合は、必要なツールを含むカスタム環 境の Docker イメージを提供できます。

Note

{

Lambda は、マルチアーキテクチャのコンテナイメージを使用する関数をサポートしません。詳細については、<u>「デベロッパーガイド」の「コンテナイメージを使用して Lambda 関数を作成する</u>」を参照してください。 AWS Lambda

Lambda コンピューティングにカスタムイメージを使用するには、次の Amazon ECR アクセス許可 が必要です。

```
"Version": "2012-10-17",
"Statement": [
```

```
{
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "arn:aws:ecr:image-region:image-account-id:repository/image-repo"
    }
  ]
}
```

また、カスタムイメージを使用するには、curl または wget をインストールする必要があります。

CodeBuild で AWS Lambda コンピューティングをサポートしているリー ジョンはどれですか?

CodeBuild では、 AWS Lambda 米国 AWS リージョン東部 (バージニア北部)、米国東部 (オハイ オ)、米国西部 (オレゴン)、アジアパシフィック (ムンバイ)、アジアパシフィック (シンガポー ル)、アジアパシフィック (シドニー)、アジアパシフィック (東京)、欧州 (フランクフルト)、欧 州 (アイルランド)、南米 (サンパウロ) でコンピューティングがサポートされています。CodeBuild が使用可能な AWS リージョン の詳細については、「<u>AWS サービス (リージョン別)</u>」を参照して ください。

AWS Lambda コンピューティングの制限

AWS Lambda がサポートしていないユースケースがいくつかあり、それらが影響する場合は EC2 コ ンピューティングを使用します。

- AWS Lambda は、ルートアクセス許可を必要とするツールをサポートしていません。yum や rpm などのツールには、EC2 コンピューティングタイプや root 権限を必要としないその他のツールを 使用してください。
- AWS Lambda は Docker のビルドまたは実行をサポートしていません。

- AWS Lambda では、の外部ファイルへの書き込みはサポートされていません/tmp。付属のパッケージマネージャーは、パッケージのダウンロードと参照にデフォルトで /tmp ディレクトリを使用するように設定されています。
- ・ AWS Lambda は 環境タイプをサポートしておらずLINUX_GPU_CONTAINER、Windows Server Core 2019 ではサポートされていません。
- AWS Lambda は、キャッシュ、カスタムビルドタイムアウト、キュータイムアウト、ビルドバッジ、特権モード、カスタムランタイム環境、または 15 分を超えるランタイムをサポートしていません。
- AWS Lambda は、VPC 接続、固定範囲の CodeBuild ソース IP アドレス、EFS、証明書のインス トール、または Session Manager を使用した SSH アクセスをサポートしていません。

CodeBuild Lambda Java AWS SAM で を使用して Lambda 関数をデプロイ する

AWS Serverless Application Model (AWS SAM)は、サーバーレスアプリケーションを構築するた めのオープンソースフレームワークです。詳細については、GitHubの「<u>AWS Serverless Application</u> <u>Model repository</u>」を参照してください。次の Java サンプルでは、Gradle を使用して AWS Lambda 関数を構築およびテストします。その後、 CLI AWS SAM を使用して AWS CloudFormation テンプ レートとデプロイバンドルをデプロイします。CodeBuild Lambda を使用すると、ビルド、テスト、 デプロイのステップがすべて自動的に処理されるため、1 つのビルドで、手動介入なしにインフラス トラクチャをすばやく更新できます。

AWS SAM リポジトリをセットアップする

CLI AWS SAM を使用して プロジェクトを作成します AWS SAM Hello World。

AWS SAM プロジェクトを作成するには

- 1. ローカルマシンに <u>AWS SAM CLI をインストールするには、</u>AWS Serverless Application Model 「 デベロッパーガイド」の手順に従います。
- 2. sam init を実行し、次のプロジェクト設定を選択します。

Which template source would you like to use?: 1 - AWS Quick Start Templates Choose an AWS Quick Start application template: 1 - Hello World Example Use the most popular runtime and package type? (Python and zip) [y/N]: N Which runtime would you like to use?: 8 - java21 What package type would you like to use?: 1 - Zip Which dependency manager would you like to use?: 1 - gradle Would you like to enable X-Ray tracing on the function(s) in your application? [y/ N]: N Would you like to enable monitoring using CloudWatch Application Insights? [y/N]: N Would you like to set Structured Logging in JSON format on your Lambda functions? [y/N]: N Project name [sam-appl: <insert project name>

- Project name [sam-app]: <insert project name>
- AWS SAM プロジェクトフォルダをサポートされているソースリポジトリにアップロードしま す。サポートされているソースタイプのリストについては、「<u>ProjectSource</u>」を参照してくだ さい。

CodeBuild Lambda Java プロジェクトを作成

AWS CodeBuild Lambda Java プロジェクトを作成し、ビルドに必要な IAM アクセス許可を設定し ます。

CodeBuild Lambda Java プロジェクトを作成するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
- 4. Source で、 AWS SAM プロジェクトが配置されているソースリポジトリを選択します。
- 5. [環境] で以下の操作を行います。
 - [コンピューティング] で、[Lambda] を選択します。
 - ・ [ランタイム] で [Java] を選択します。
 - [イメージ] で、[aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21] を選択します。
 - [サービスロール] では、[新しいサービスロール] を選択したままにします。[ロール名] を書き 留めます。これは、このサンプルの後半でプロジェクトの IAM アクセス許可を更新するとき に必要です。
- 6. Create build project (ビルドプロジェクトの作成)を選択します。

- 7. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
- 5. ナビゲーションペインで、[ロール] を選択し、プロジェクトに関連付けられたサービスロール を選択します。CodeBuild でプロジェクトロールを見つけるには、ビルドプロジェクトを選択 し、[編集]、[環境]、[サービスロール] を選択します。
- 9. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。
- IAM ロールに以下のインラインポリシーを追加します。これは、後で AWS SAM インフラスト ラクチャをデプロイするために使用されます。詳細については、「 IAM ユーザーガイド」の 「IAM ID アクセス許可の追加および削除」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                 "cloudformation:*",
                 "lambda:*",
                 "iam:*",
                 "apigateway:*",
                 "s3:*"
            ],
            "Resource": "*"
        }
    ]
}
```

プロジェクトの buildspec の設定

Lambda 関数をビルド、テスト、デプロイするために、CodeBuild は buildspec からビルドコマンド を読み取り、実行します。

プロジェクトの buildspec を設定するには

- 1. CodeBuild コンソールで、ビルドプロジェクトを選択し、[編集] と [Buildspec] を選択します。
- 2. [Buildspec] で、[ビルドコマンドを挿入]、[エディタに切り替え] の順に選択します。
- 3. 事前入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

version: 0.2

```
env:
  variables:
    GRADLE_DIR: "HelloWorldFunction"
phases:
    build:
    commands:
        echo "Running unit tests..."
        echo "Running unit tests..."
        cd $GRADLE_DIR; gradle test; cd ..
        echo "Running build..."
        sam build --template-file template.yaml
        echo "Running deploy..."
        sam package --output-template-file packaged.yaml --resolve-s3 --template-
file template.yaml
        _ yes | sam deploy
```

4. [Update buildspec (buildspec の更新)] を選択します。

AWS SAM Lambda インフラストラクチャをデプロイする

CodeBuild Lambda を使用して Lambda インフラストラクチャを自動的にデプロイ

Lambda インフラストラクチャをデプロイするには

- 1. [Start build] を選択します。これにより、 AWS Lambda を使用して AWS SAM アプリケーショ ンが自動的に構築、テスト、デプロイされます AWS CloudFormation。
- ビルドが完了したら、 AWS Lambda コンソールに移動し、 AWS SAM プロジェクト名で新しい Lambda 関数を検索します。
- [関数] の概要で [API Gateway] を選択し、[API エンドポイント] URL をクリックして、Lambda
 関数をテストします。メッセージ "message": "hello world" を含むページが開きます。

インフラストラクチャをクリーンアップ

このチュートリアルで使用したリソースの追加料金を回避するには、 AWS SAM テンプレートと CodeBuild によって作成されたリソースを削除します。

インフラストラクチャをクリーンアップするには

- AWS CloudFormation コンソールに移動し、を選択しますaws-sam-cli-manageddefault。
- 2. [リソース]で、デプロイバケット SamCliSourceBucket を空にします。

- 3. aws-sam-cli-managed-default スタックを削除します。
- AWS SAM プロジェクトに関連付けられている AWS CloudFormation スタックを削除します。
 このスタックの名前は AWS SAM プロジェクトと同じである必要があります。
- 5. CloudWatch コンソールに移動し、CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
- 6. CodeBuild コンソールに移動し、[ビルドプロジェクトを削除] を選択して CodeBuild プロジェクトを削除します。

CodeBuild Lambda Node.js を使用してシングルページの React アプリを作 成

「<u>Create React App</u>」は、シングルページの React アプリケーションを作成する方法です。次の Node.js サンプルは、Node.js を使用して「Create React App」からソースアーティファクトをビル ドし、ビルドアーティファクトを返します。

ソースリポジトリとアーティファクトバケットを設定

yarn と「Create React App」を使用して、プロジェクトのソースリポジトリを作成します。

ソースリポジトリとアーティファクトバケットを設定するには

- ローカルマシンで yarn create react-app <app-name> を実行して、シンプルな React ア プリを作成します。
- サポートされているソースリポジトリに、React アプリプロジェクトフォルダをアップロードします。サポートされているソースタイプのリストについては、「<u>ProjectSource</u>」を参照してください。

CodeBuild Lambda Node.js プロジェクトを作成

AWS CodeBuild Lambda Node.js プロジェクトを作成します。

CodeBuild Lambda Node.js プロジェクトを作成するには

1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https

- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
- 4. Source で、 AWS SAM プロジェクトが配置されているソースリポジトリを選択します。
- 5. [環境] で以下の操作を行います。
 - [コンピューティング] で、[Lambda] を選択します。
 - ・ [ランタイム] で、[Node.js] を選択します。
 - [イメージ] で、[aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20] を選択します。
- 6. [アーティファクト] で、次のようにします。
 - [タイプ] で、[Amazon S3] を選択します。
 - [バケット名] で、先ほど作成したプロジェクトアーティファクトバケットを選択します。
 - ・ [アーティファクトのパッケージ化] では、[Zip] を選択します。
- 7. Create build project (ビルドプロジェクトの作成)を選択します。

プロジェクトの buildspec の設定

React アプリをビルドするために、CodeBuild は buildspec ファイルからビルドコマンドを読み取 り、実行します。

プロジェクトの buildspec を設定するには

- 1. CodeBuild コンソールで、ビルドプロジェクトを選択し、[編集] と [Buildspec] を選択します。
- 2. [Buildspec] で、[ビルドコマンドを挿入]、[エディタに切り替え] の順に選択します。
- 3. 事前入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

version: 0.2
phases:
 build:
 commands:
 - yarn

```
- yarn add --dev jest-junit @babel/plugin-proposal-private-property-in-object
      - yarn run build
      - yarn run test -- --coverage --watchAll=false --testResultsProcessor="jest-
junit" --detectOpenHandles
artifacts:
  name: "build-output"
 files:
    - "**/*"
reports:
  test-report:
    files:
      - 'junit.xml'
    file-format: 'JUNITXML'
  coverage-report:
    files:
      - 'coverage/clover.xml'
    file-format: 'CLOVERXML'
```

4. [Update buildspec (buildspec の更新)] を選択します。

React アプリをビルドして実行

CodeBuild Lambda で React アプリをビルドし、ビルドアーティファクトをダウンロードして、React アプリをローカルで実行します。

React アプリをビルドして実行するには

- 1. [Start build] を選択します。
- ビルドが完了したら、Amazon S3 プロジェクトアーティファクトバケットに移動し、React ア プリアーティファクトをダウンロードします。
- React ビルドアーティファクトと run npm install -g serve && serve -s build をプ ロジェクトフォルダに解凍します。
- serve コマンドは、ローカルポートで静的サイトを提供し、出力をターミナルに出力します。
 ターミナル出力の Local: にある localhost URL にアクセスして、React アプリを表示できます。

React ベースのサーバーのデプロイを処理する方法の詳細については、「<u>Create React App</u> Deployment」を参照してください。

インフラストラクチャをクリーンアップ

このチュートリアルで使用したリソースに対して追加料金が発生しないようにするには、CodeBuild プロジェクト用に作成されたリソースを削除します。

インフラストラクチャをクリーンアップするには

- 1. プロジェクトアーティファクト Amazon S3 バケットを削除
- CloudWatch コンソールに移動し、CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
- 3. CodeBuild コンソールに移動し、[ビルドプロジェクトを削除] を選択して CodeBuild プロジェクトを削除します。

CodeBuild Lambda Python を使用して Lambda 関数の設定を更新

次の Python サンプルは、<u>Boto3</u> と CodeBuild Lambda Python を使用して Lambda 関数の設定を更 新します。このサンプルを拡張して、他の AWS リソースをプログラムで管理できます。詳細につい ては、「<u>Boto3 ドキュメント</u>」を参照してください。

前提条件

アカウントで Lambda 関数を作成または検索します。

このサンプルは、アカウントで Lambda 関数を既に作成しており、CodeBuild を使用して Lambda 関数の環境変数を更新することを前提としています。CodeBuild を使用して Lambda 関数を設定す る方法の詳細については、「<u>CodeBuild Lambda Java AWS SAM で を使用して Lambda 関数をデプ</u> ロイする」サンプルを参照するか、「AWS Lambda」を参照してください。

ソースリポジトリを設定

Boto3 Python スクリプトを保存するソースリポジトリを作成します。

ソースコードリポジトリを設定するには

次の Python スクリプトを update_lambda_environment_variables.py という名前の新しいファイルにコピーします。

import boto3
from os import environ

```
def update_lambda_env_variable(lambda_client):
    lambda_function_name = environ['LAMBDA_FUNC_NAME']
   lambda_env_variable = environ['LAMBDA_ENV_VARIABLE']
   lambda_env_variable_value = environ['LAMBDA_ENV_VARIABLE_VALUE']
    print("Updating lambda function " + lambda_function_name + " environment
variable "
          + lambda_env_variable + " to " + lambda_env_variable_value)
   lambda_client.update_function_configuration(
        FunctionName=lambda_function_name,
        Environment={
            'Variables': {
                lambda_env_variable: lambda_env_variable_value
            }
       },
    )
if ___name___ == "___main___":
    region = environ['AWS_REGION']
    client = boto3.client('lambda', region)
    update_lambda_env_variable(client)
```

2. サポートされているソースリポジトリに Python ファイルをアップロードします。サポートされ ているソースタイプのリストについては、「<u>ProjectSource</u>」を参照してください。

CodeBuild Lambda Python プロジェクトを作成

CodeBuild Lambda Python プロジェクトを作成します。

CodeBuild Lambda Java プロジェクトを作成するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- CodeBuild の情報ページが表示された場合、ビルドプロジェクトを作成するを選択します。それ 以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
- [プロジェクト名]に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、 AWS アカウントごとに一意である必要があります。また、他のユーザーがこのプロジェクトの 使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。

- 4. Source で、 AWS SAM プロジェクトが配置されているソースリポジトリを選択します。
- 5. [環境] で以下の操作を行います。
 - [コンピューティング] で、[Lambda] を選択します。
 - [ランタイム] で [Python] を選択します。
 - ・ [イメージ] で、[aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12] を選択しま す。
 - [サービスロール] では、[新しいサービスロール] を選択したままにします。[ロール名] を書き 留めます。これは、このサンプルの後半でプロジェクトの IAM アクセス許可を更新するとき に必要です。
- 6. Create build project (ビルドプロジェクトの作成)を選択します。
- 7. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
- ナビゲーションペインで、[ロール] を選択し、プロジェクトに関連付けられたサービスロール を選択します。CodeBuild でプロジェクトロールを見つけるには、ビルドプロジェクトを選択 し、[編集]、[環境]、[サービスロール] を選択します。
- 9. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。
- IAM ロールに以下のインラインポリシーを追加します。これは、後で AWS SAM インフラスト ラクチャをデプロイするために使用されます。詳細については、「 IAM ユーザーガイド」の 「IAM ID アクセス許可の追加および削除」を参照してください。

プロジェクトの buildspec の設定

Lambda 関数を更新するために、スクリプトは buildspec から環境変数を読み取り、Lambda 関数の 名前、環境変数名、および環境変数値を検索します。

プロジェクトの buildspec を設定するには

- 1. CodeBuild コンソールで、ビルドプロジェクトを選択し、[編集] と [Buildspec] を選択します。
- 2. [Buildspec] で、[ビルドコマンドを挿入]、[エディタに切り替え] の順に選択します。
- 3. 事前入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

```
version: 0.2
env:
  variables:
  LAMBDA_FUNC_NAME: "<lambda-function-name>"
  LAMBDA_ENV_VARIABLE: "FEATURE_ENABLED"
  LAMBDA_ENV_VARIABLE_VALUE: "true"
phases:
  install:
    commands:
        - pip3 install boto3
build:
    commands:
        - python3 update_lambda_environment_variables.py
```

4. [Update buildspec (buildspec の更新)] を選択します。

Lambda 設定を更新

CodeBuild Lambda Python を使用して、Lambda 関数の設定を自動的に更新します。

Lambda 関数の設定を更新するには

- 1. [Start build] を選択します。
- 2. ビルドが完了したら、Lambda 関数に移動します。
- [設定] を選択してから、[環境] 変数を選択します。キー FEATURE_ENABLED と値 true を持つ 新しい環境変数が表示されます。

インフラストラクチャをクリーンアップ

このチュートリアルで使用したリソースに対して追加料金が発生しないようにするには、CodeBuild プロジェクト用に作成されたリソースを削除します。

インフラストラクチャをクリーンアップするには

- 1. CloudWatch コンソールに移動し、CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
- 2. CodeBuild コンソールに移動し、[ビルドプロジェクトを削除] を選択して CodeBuild プロジェクトを削除します。
- このサンプル用に Lambda 関数を作成した場合は、[アクション] および [関数を削除] を選択して Lambda 関数をクリーンアップします。

拡張子

AWS CodeBuild Lambda Python を使用して他の AWS リソースを管理するためにこのサンプルを拡張する場合:

- Boto3 を使用して新しいリソースを変更するように Python スクリプトを更新します。
- CodeBuild プロジェクトに関連付けられた IAM ロールを更新して、新しいリソースに対するアクセス許可を付与します。
- 新しいリソースに関連付けられた新しい環境変数を buildspec に追加します。

リザーブドキャパシティキャパシティフリートでビルドを実行

CodeBuild には以下のコンピューティングフリートがあります。

- ・ オンデマンドフリート
- リザーブドキャパシティフリート

オンデマンドフリートでは、CodeBuild がビルドのコンピューティングを行います。マシンはビルド が終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応でき る自動スケーリング機能を備えています。 (i) Note

オンデマンドフリートは macOS をサポートしていません。

CodeBuild では、CodeBuild が管理する Amazon EC2 ベースのインスタンスを含むリザーブドキャ パシティフリートも提供しています。リザーブドキャパシティフリートでは、ビルド環境に合わせて 専有インスタンスのセットを設定します。これらのマシンはアイドル状態のままで、ビルドやテスト をすぐに処理できる状態になり、ビルド時間を短縮します。リザーブドキャパシティフリートでは、 マシンは常に稼働しており、プロビジョニングされている間はコストが発生し続けます。

▲ Important

インスタンスの実行時間に関係なく、リザーブドキャパシティフリートにはインスタンスご とに初期料金が発生し、その後は追加の関連コストが発生する場合があります。詳細につい ては、「https://aws.amazon.com/codebuild/pricing/」を参照してください。

トピック

- リザーブドキャパシティフリートを作成
- ベストプラクティス
- リザーブドキャパシティフリートを複数の CodeBuild プロジェクトで共有できますか?
- 属性ベースのコンピューティングの仕組み
- フリートの Amazon EC2 インスタンスを手動で指定できますか?
- リザーブドキャパシティフリートをサポートしているのはどのリージョンですか?
- リザーブドキャパシティの macOS フリートを設定するにはどうすればよいですか。
- <u>リザーブドキャパシティフリートのカスタム Amazon マシンイメージ (AMI) を設定するにはどう</u> すればよいですか?
- リザーブドキャパシティフリートの制限
- <u>リザーブドキャパシティフリートのプロパティ</u>
- AWS CodeBuildを使用したリザーブドキャパシティのサンプル

リザーブドキャパシティフリートを作成

以下の手順に従って、リザーブドキャパシティフリートを作成します。

リザーブドキャパシティフリートを作成するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codesuite/</u> codebuild/home で AWS CodeBuild コンソールを開きます。
- 2. ナビゲーションペインで、[コンピューティングフリート]、[フリートを作成] の順に選択しま す。
- 3. [コンピューティングフリート名] テキストフィールドに、フリートの名前を入力します。
- [オペレーティングシステム] ドロップダウンメニューから、オペレーティングシステムを選択し ます。
- 5. [アーキテクチャ] ドロップダウンメニューから、アーキテクチャを選択します。
- (オプション)インスタンス実行モードを使用する オプションで、Docker コンテナの代わりに Amazon EC2 インスタンスで直接実行します。次に、メジャーバージョンとマイナーバージョ ンを選択します。
- 7. (オプション)[追加設定]で、以下を実行します。
 - VPC の設定 フリートを VPC に接続して使用中にプライベートリソースにアクセスする場合 はオプションを選択します。
 - ・ VPC ドロップダウンメニューから、CodeBuild フリートがアクセスする VPC を選択します。
 - ・ [サブネット] ドロップダウンメニューから、CodeBuild が VPC 設定のセットアップに使用 するサブネットを選択します。
 - [セキュリティグループ]のドロップダウンメニューから、CodeBuild が VPC の操作に使用 するセキュリティグループを選択します。
 - [フリートサービスロール] フィールドで、既存のサービスロールを選択します。

フリートロールに必要なアクセス許可が付与されていることを確認してください。 詳細については、「<u>フリートサービスロールのアクセス許可ポリシーを追加するこ</u> とをユーザーに許可」を参照してください。

Amazon Linux オペレーティングシステムを選択した場合は、[プロキシ設定の定義 - オプション]を選択して、リザーブドキャパシティインスタンスにネットワークアクセスコントロールを適用します。

Note

- [デフォルトの動作] では、デフォルトですべての送信先への送信トラフィックを許可または 拒否することを選択します。
- [プロキシルール]では、[プロキシルールを追加]を選択して、ネットワークアクセスコントロールを許可または拒否する送信先ドメインまたは IP を指定します。
- カスタム AMI の設定 カスタム Amazon マシンイメージ (AMI) を使用するにはオプション を 選択します。
 - ・ AMI ドロップダウンメニューから、フリートの Amazon マシンイメージ (AMI) を選択します。
 - [フリートサービスロール] フィールドで、既存のサービスロールを選択します。

Note

フリートロールに必要なアクセス許可が付与されていることを確認してください。 詳細については、「<u>フリートサービスロールのアクセス許可ポリシーを追加するこ</u> <u>とをユーザーに許可</u>」を参照してください。

- 8. キャパシティ設定で、コンピューティング選択モードから次のいずれかを選択します。
 - ガイド付き選択を選択した場合は、次の操作を行います。
 - Compute で、このフリートに含まれるインスタンスのタイプを選択します。
 - [容量] テキストフィールドに、フリート内の最小インスタンス数を入力します。
 - ・ (オプション) [追加設定] で、以下を実行します。
 - スケーリングの設定 オプションを選択して、この設定に基づいてフリートを自動的にス ケーリングします。スケーリングモード - オプションのドロップダウンメニューから、需 要がフリート容量を超えたときの動作を選択します。
 - カスタムインスタンスを選択した場合は、次の操作を行います。
 - Compute インスタンスタイプのドロップダウンメニューから、このフリートに含まれるインスタンスのタイプを選択します。
 - 追加の EBS ボリュームサイズ オプションのテキストフィールドに、提供された 64GB の ディスク容量に加えてボリュームを入力します。
 - [容量] テキストフィールドに、フリート内の最小インスタンス数を入力します。
 - ・ (オプション) [追加設定] で、以下を実行します。

- スケーリングの設定 オプションを選択して、この設定に基づいてフリートを自動的にス ケーリングします。スケーリングモード - オプションのドロップダウンメニューから、需 要がフリート容量を超えたときの動作を選択します。
- 9. [コンピューティングフリートの作成]を選択します。
- 10. コンピューティングフリートを作成したら、新しい CodeBuild プロジェクトを作成するか、既 存の CodeBuild プロジェクトを編集します。[環境] から [プロビジョニングモデル] の [リザーブ ドキャパシティ] を選択し、[フリート名] で指定したフリートを選択します。

ベストプラクティス

リザーブドキャパシティフリートを使用する場合は、以下のベストプラクティスに従うことをお勧め します。

- ソースをキャッシュしてビルドパフォーマンスを向上させるには、ソースキャッシュモードを使用 することをお勧めします。
- Docker レイヤーキャッシュを使用し、既存の Docker レイヤーをキャッシュしてビルドパフォーマンスを向上させることをお勧めします。

リザーブドキャパシティフリートを複数の CodeBuild プロジェクトで共有 できますか?

はい。複数のプロジェクトで使用することで、フリートの容量を最大限に活用できます。

A Important

リザーブドキャパシティ機能を使用すると、ソースファイル、Docker レイヤー、buildspec で指定されキャッシュされたディレクトリなどを含む、フリートインスタンスにキャッシュ されたデータに、同じアカウント内の他のプロジェクトからアクセスできます。これは設計 によるもので、同じアカウント内のプロジェクトがフリートインスタンスを共有できるよう にしています。

属性ベースのコンピューティングの仕組み

フリートの ATTRIBUTE_BASED_COMPUTEとして を選択した場合はcomputeType、 という新 しいフィールドに属性を指定できますcomputeConfiguration。これらの属性には、vCPUs
メモリ、ディスク容量、および が含まれますmachineType。これは GENERALまたは のいずれ かmachineTypeですNVME。CodeBuild は、使用可能な属性の 1 つまたは一部を指定した後、サポー トされている使用可能なインスタンスタイプからコンピューティングタイプを確定済み として選択 しますcomputeConfiguration。

Note

CodeBuild は、すべての入力要件に一致する最も安価なインスタンスを選択します。選択し たインスタンスのメモリ、vCPUs、ディスク容量はすべて、入力要件以上になります。作成 または更新されたフリートcomputeConfigurationで解決された を確認できます。

CodeBuild で満たすcomputeConfigurationことができない を入力すると、検証例外が表示さ れます。また、 がオンデマンドで利用できない場合、オンデマンドフリートのオーバーフロー動 作computeConfigurationはキュー動作に上書きされることに注意してください。

フリートの Amazon EC2 インスタンスを手動で指定できますか?

はい。カスタムインスタンスを選択するか、API パラメータ を設定することで、コンソール で目的の Amazon EC2 インスタンスを直接入力できますInstanceType。 このフィールド は、CreateFleet、UpdateFleet、CreateProject、UpdateProject、StartBuild の APIs で使用されま す。詳細については、「Compute instance type」を参照してください。

リザーブドキャパシティフリートをサポートしているのはどのリージョン ですか?

リザーブドキャパシティの Amazon Linux および Windows フリートは、 AWS リージョン米国東 部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)、アジアパシフィック (ムンバ イ)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィッ ク (東京)、欧州 (フランクフルト)、欧州 (アイルランド)、南米 (サンパウロ) でサポートされて います。CodeBuild が使用可能な AWS リージョン の詳細については、「<u>AWS サービス (リージョ</u> ン別)」を参照してください。

リザーブドキャパシティの macOS Medium フリートは、 AWS リージョン米国東部 (バージニア北 部)、米国東部 (オハイオ)、米国西部 (オレゴン)、アジアパシフィック (シドニー)、欧州 (フラ ンクフルト) でサポートされています。リザーブドキャパシティの macOS Large フリートは、 AWS リージョン米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)、アジアパシ フィック (シドニー) でサポートされています。

リザーブドキャパシティの macOS フリートを設定するにはどうすればよ いですか。

リザーブドキャパシティの macOS フリートを設定するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codesuite/</u> codebuild/home で AWS CodeBuild コンソールを開きます。
- ナビゲーションペインで、[コンピューティングフリート]、[フリートを作成] の順に選択します。
- 3. [コンピューティングフリート名] テキストフィールドに、フリートの名前を入力します。
- 4. [オペレーティングシステム] のドロップダウンメニューから、[macOS] を選択します。
- 5. [コンピューティング] フィールドで、[Apple M2、24 GB メモリ、8 vCPU] または [Apple M2、32 GB メモリ、12 vCPU] のいずれかのコンピューティングマシンタイプを選択します。
- 6. [容量] テキストフィールドに、フリート内の最小インスタンス数を入力します。
- (オプション) フリートにカスタムイメージを使用するには、<u>リザーブドキャパシティフリート</u> <u>のカスタム Amazon マシンイメージ (AMI) を設定するにはどうすればよいですか?</u>「」を参照 して、Amazon マシンイメージ (AMI) に必要な前提条件があることを確認します。
- 8. (オプション) フリートで VPC を設定するには、[追加設定] で以下を実行します。
 - ・ [VPC オプション] のドロップダウンメニューから、CodeBuild フリートがアクセスする VPC を選択します。
 - ・ [サブネット] ドロップダウンメニューから、CodeBuild が VPC 設定のセットアップに使用す るサブネットを選択します。
 - ・ [セキュリティグループ] のドロップダウンメニューから、CodeBuild が VPC の操作に使用す るセキュリティグループを選択します。
 - [フリートサービスロール] フィールドで、既存のサービスロールを選択します。

Note

フリートロールに必要なアクセス許可が付与されていることを確認してください。詳 細については、「フリートサービスロールのアクセス許可ポリシーを追加することを <u>ユーザーに許可</u>」を参照してください。

9. [コンピューティングフリートを作成]を選択し、フリートインスタンスの起動を待ちます。起動 すると、キャパシティは n/n になります。n は指定されたキャパシティです。 10. コンピューティングフリートが起動したら、新しい CodeBuild プロジェクトを作成するか、既 存の CodeBuild プロジェクトを編集します。[環境] から [プロビジョニングモデル] の [リザーブ ドキャパシティ] を選択し、[フリート名] で指定したフリートを選択します。

リザーブドキャパシティフリートのカスタム Amazon マシンイメージ (AMI) を設定するにはどうすればよいですか?

リザーブドキャパシティフリートのカスタム Amazon マシンイメージ (AMI) を設定するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codesuite/</u> codebuild/home で AWS CodeBuild コンソールを開きます。
- 2. ナビゲーションペインで、[コンピューティングフリート]、[フリートを作成] の順に選択します。
- 3. [コンピューティングフリート名] テキストフィールドに、フリートの名前を入力します。
- 4. フリートのカスタムイメージを選択し、Amazon マシンイメージ (AMI) に次の前提条件があるこ とを確認します。
 - ・環境タイプが の場合はMAC_ARM、AMI アーキテクチャが 64 ビット であることを確認しま すMac-Arm。
 - ・環境タイプが の場合はLINUX_EC2、AMI アーキテクチャが 64 ビット であることを確認しま すx86。
 - ・環境タイプが の場合はARM_EC2、AMI アーキテクチャが 64 ビット であることを確認しますArm。
 - ・環境タイプが の場合はWINDOWS_EC2、AMI アーキテクチャが 64 ビット であることを確認し ますx86。
 - AMI は CodeBuild サービスに [組織 ARN] を許可します。組織 ARN のリストについては、 「Amazon Machine Images (AMI)」を参照してください。
 - AMI が AWS KMS キーで暗号化されている場合、 AWS KMS キーは CodeBuild サービス組織 ID も許可する必要があります。組織 ID のリストについては、「<u>Amazon Machine Images</u> (AMI)」を参照してください。 AWS KMS キーの詳細については、Amazon EC2 <u>ユーザーガ</u> イド」のOUs に KMS キーの使用を許可する」を参照してください。CodeBuild 組織に KMS キーを使用するアクセス許可を付与するには、キーポリシーに次のステートメントを追加しま す。

[{]

```
"Sid": "Allow access for organization root",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
        "kms:Describe*",
        "kms:List*",
        "kms:Get*",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:PrincipalOrgID": "o-123example"
        }
    }
}
```

・ [フリートサービスロール] フィールドで、次の Amazon EC2 アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "ec2:DescribeImages",
               "ec2:DescribeSnapshots"
              ],
             "Resource": "*"
        }
    ]
}
```

リザーブドキャパシティフリートの制限

リザーブドキャパシティフリートではサポートされていないユースケースがいくつかあります。それ により影響が出る場合は、代わりにオンデマンドフリートを使用してください。 リザーブドキャパシティフリートは、ビルド使用率メトリクスをサポートしていません。

• リザーブドキャパシティの macOS フリートはデバッグセッションをサポートしていません。

クォータと制限の詳細については、「コンピューティングフリート」を参照してください。

リザーブドキャパシティフリートのプロパティ

リザーブドキャパシティフリートには以下のプロパティが含まれます。リザーブドキャパシティフ リートの詳細については、「<u>リザーブドキャパシティキャパシティフリートでビルドを実行</u>」を参照 してください。

オペレーティングシステム

オペレーティングシステム。使用できるオペレーションシステムは次のとおりです。

- ・ Amazon リナックス
- macOS
- [Windows Server 2019]
- Windows Server 2022
- アーキテクチャ

プロセッサアーキテクチャ。以下のアーキテクチャが利用可能です。

- x86_64
- Arm64
- 環境タイプ

Amazon Linux が選択されているときに使用できる環境タイプ。次の環境タイプを使用できます。

- Linux EC2
- Linux GPU

コンピューティングインスタンスタイプ

フリートインスタンスのコンピューティング設定。

ガイド付き選択

vCPU、メモリ、ディスク容量の設定を選択して、さまざまなコンピューティングタイプを指 定します。リージョン別のコンピューティングタイプの可用性については、「」を参照してく ださい<u>リザーブドキャパシティ環境タイプについて</u>。 カスタムインスタンス

目的のインスタンスタイプを手動で指定します。

容量

フリートに割り当てられるマシンの初期数。これにより、並列で実行できるビルドの数が定義されます。

オーバーフロー動作

ビルド数がフリート容量を超えたときの動作を定義します。

[オンデマンド]

オーバーフロービルドは CodeBuild でオンデマンドで実行されます。

Note

VPC 接続フリートの作成中にオーバーフロー動作をオンデマンドに設定する場合は、 必要な VPC アクセス許可をプロジェクトサービスロールに追加してください。詳細 については、「<u>Example policy statement to allow CodeBuild access to AWS services</u> required to create a VPC network interface」を参照してください。

A Important

オーバーフロー動作をオンデマンドに設定する場合は、オンデマンドの Amazon EC2 と同様に、オーバーフロービルドには別途請求されることに注意してください。詳細 については、「https://aws.amazon.com/codebuild/pricing/」を参照してください。

キュー

ビルドの実行は、マシンが使用可能になるまでキューに入れられます。これにより、さらにマ シンが割り当てられないため、追加のコストが抑えられます。

Amazon マシンイメージ (AMI)

フリートの Amazon マシンイメージ (AMI) プロパティです。CodeBuild では以下のプロパティが サポートされています。

AWS リージョン	組織 ARN	組織 ID
us-east-1	arn:aws:organizati ons::851725618577: organization/o-c6w cu152r1	o-c6wcu152r1
us-east-2	arn:aws:organizati ons::992382780434: organization/o-seu fr2suvq	o-seufr2suvq
us-west-2	arn:aws:organizati ons::381491982620: organization/o-041 2099a4r	o-0412o99a4r
ap-northeast-1	arn:aws:organizati ons::891376993293: organization/o-b6k 3sjqavm	o-b6k3sjqavm
ap-south-1	arn:aws:organizati ons::891376924779: organization/o-krt ah1lkeg	o-krtah1lkeg
ap-southeast-1	arn:aws:organizati ons::654654522137: organization/o-mcn 8uvc3tp	o-mcn8uvc3tp
ap-southeast-2	arn:aws:organizati ons::767398067170: organization/o-6cr t0f6bu4	o-6crt0f6bu4

AWS リージョン	組織 ARN	組織 ID
eu-central-1	arn:aws:organizati ons::590183817084: organization/o-lb2 lne3te6	o-lb2lne3te6
eu-west-1	arn:aws:organizati ons::891376938588: organization/o-ull rrg5qf0	o-ullrrg5qf0
sa-east-1	arn:aws:organizati ons::533267309133: organization/o-db6 3c45ozw	o-db63c45ozw

追加設定

[VPC - オプション]

CodeBuild フリートがアクセスする VPC です。詳細については、「<u>Amazon Virtual Private</u> Cloud AWS CodeBuild で を使用する」を参照してください。

サブネット

CodeBuild が VPC 設定のセットアップに使用する VPC サブネット。リザーブドキャパシ ティフリートは、単一のアベイラビリティゾーンで 1 つのサブネットのみをサポートするこ とに注意してください。また、サブネットに NAT ゲートウェイが含まれていることを確認し てください。

セキュリティグループ

CodeBuild が VPC で使用する VPC セキュリティグループです。セキュリティグループがア ウトバウンド接続を許可していることを確認します。

[フリートサービスロール]

アカウント内の既存のサービスロールからフリートのサービスロールを定義します。

[プロキシ設定の定義 - オプション]

リザーブドキャパシティインスタンスにネットワークアクセスコントロールを適用するプロキ シ設定。詳細については、「<u>マネージドプロキシサーバー AWS CodeBuild で を使用する</u>」 を参照してください。

Note

プロキシ設定は、VPC、Windows、または MacOS をサポートしていません。

デフォルトの動作

送信トラフィックの動作を定義します。

許可

デフォルトでは、すべての送信先への送信トラフィックを許可します。

拒否

デフォルトでは、すべての送信先への送信トラフィックを拒否します。

プロキシルール

ネットワークアクセスコントロールを許可または拒否する送信先ドメインまたは IP を指定します。

AWS CodeBuildを使用したリザーブドキャパシティのサンプル

これらのサンプルを使用して、CodeBuild のリザーブドキャパシティフリートを試すことができま す。

トピック

リザーブドキャパシティのサンプルを使用したキャッシュ

リザーブドキャパシティのサンプルを使用したキャッシュ

キャッシュでは、ビルド環境の再利用可能な部分が保存され、複数のビルドでそれらを使用すること ができます。このサンプルでは、リザーブドキャパシティを使用してビルドプロジェクト内のキャッ シュを有効にする方法を示しました。詳細については、「<u>パフォーマンスを向上させるためのキャッ</u> シュビルド」を参照してください。

プロジェクト設定で1つ以上のキャッシュモードを指定することから開始できます。

Cache:

Type: LOCAL

Modes:

- LOCAL_CUSTOM_CACHE
- LOCAL_DOCKER_LAYER_CACHE
- LOCAL_SOURCE_CACHE

Note

Docker レイヤーキャッシュを使用するには、必ず特権モードを有効にしてください。

プロジェクトの buildspec 設定は以下のようになります。

```
version: 0.2
      phases:
        build:
          commands:
            - echo testing local source cache
            - touch /codebuild/cache/workspace/foobar.txt
            - git checkout -b cached_branch
            - echo testing local docker layer cache
            - docker run alpine:3.14 2>&1 | grep 'Pulling from' || exit 1
            - echo testing local custom cache
            - touch foo
            - mkdir bar && ln -s foo bar/foo2
            - mkdir bar/bar && touch bar/bar/foo3 && touch bar/bar/foo4
            - "[ -f foo ] || exit 1"
            - "[ -L bar/foo2 ] || exit 1"
            - "[ -f bar/bar/foo3 ] || exit 1"
            - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
           - './foo'
           - './bar/**/*'
           - './bar/bar/foo3'
```

新しいプロジェクトでビルドを実行してキャッシュをシードすることから開始できます。それが完了 したら、次のように buildspec を上書きして別のビルドを開始する必要があります。

version: 0.2
phases:
build:
commands:
- echo testing local source cache
- git branch if grep 'cached_branch'; then (exit 0); else (exit 1); fi
- ls /codebuild/cache/workspace if grep 'foobar.txt'; then (exit 0); else
(exit 1); fi
- echo testing local docker layer cache
- docker run alpine:3.14 2>&1 if grep 'Pulling from'; then (exit 1); else
(exit 0); fi
- echo testing local custom cache
- "[-f foo] exit 1"
- "[-L bar/foo2] exit 1"
- "[-f bar/bar/foo3] exit 1"
- "[-f bar/bar/foo4] exit 1"
cache:
paths:
- './foo'
- './bar/**/*'
- './bar/bar/foo3'

ビルドをバッチで実行

を使用して AWS CodeBuild 、バッチビルドでプロジェクトの同時ビルドと調整ビルドを実行できま す。

```
トピック
```

- セキュリティロール
- バッチビルドのタイプ
- バッチレポートモード
- 詳細情報

セキュリティロール

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロールで は、CodeBuild が StartBuild、StopBuild および RetryBuild アクションを使用して、バッチ の一部としてビルドを実行する上で必要です。次の2つの理由により、お客様はビルドで使用するも のと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドと計算タイプの数を制限する 制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を回避する 可能性があります。

バッチビルドのタイプ

CodeBuild は、次のバッチビルドタイプをサポートしています。

バッチビルドのタイプ

- ビルドグラフ
- ビルドリスト
- ビルドマトリックス
- ファンアウトの構築

ビルドグラフ

ビルドグラフは、バッチ内の他のタスクに依存する一連のタスクを定義します。

次の例では、依存関係チェーンを作成するビルドグラフを定義します。

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
```

```
variables:
	BUILD_ID: build3
depend-on:
	- build2
- identifier: build4
env:
	compute-type: ARM_LAMBDA_1GB
- identifier: build5
env:
	fleet: fleet_name
```

この例では、以下のことを行います。

- ・ build1 は、依存関係を持たないため、最初に実行されます。
- build2 は build1 への依存関係があるため、build2 は build1 の完了後に実行されます。
- build3 は build2 への依存関係があるため、build3 は build2 の完了後に実行されます。

ビルドグラフの buildspec 構文の詳細については、「batch/build-graph」を参照してください。

ビルドリスト

ビルドリストは、並行して実行されるタスクの数を定義します。

次の例では、ビルドリストを定義します。build1 ビルドと build2 ビルドは並行して実行されま す。

```
batch:
fast-fail: false
build-list:
    - identifier: build1
    env:
      variables:
      BUILD_ID: build1
    ignore-failure: false
    - identifier: build2
    buildspec: build2.yml
    env:
      variables:
      BUILD_ID: build2
    ignore-failure: true
    - identifier: build3
```

```
env:
    compute-type: ARM_LAMBDA_1GB
- identifier: build4
    env:
      fleet: fleet_name
- identifier: build5
    env:
      compute-type: GENERAL_LINUX_XLAGRE
```

ビルドリストの buildspec 構文の詳細については、「batch/build-list」を参照してください。

ビルドマトリックス

ビルドマトリックスは、並行して実行される異なる構成のタスクを定義します。CodeBuild は、設定 可能な組み合わせごとに個別のビルドを作成します。

次の例は、2 つの buildspec ファイルと環境変数の 3 つの値を含むビルド行列を示しています。

```
batch:
    build-matrix:
    static:
        ignore-failure: false
    dynamic:
        buildspec:
            - matrix1.yml
            - matrix2.yml
    env:
        variables:
        MY_VAR:
            - VALUE1
            - VALUE1
            - VALUE2
            - VALUE3
```

この例では、CodeBuild は 6 つのビルドを作成します。

- matrix1.yml(を含む)\$MY_VAR=VALUE1
- matrix1.yml(を含む)\$MY_VAR=VALUE2
- matrix1.yml(を含む)\$MY_VAR=VALUE3
- matrix2.yml(を含む)\$MY_VAR=VALUE1
- matrix2.yml(を含む)\$MY_VAR=VALUE2

• matrix2.yml(を含む)\$MY_VAR=VALUE3

各ビルドには次の設定があります。

- ignore-failure が false に設定
- env/type が LINUX_CONTAINER に設定
- env/image が aws/codebuild/amazonlinux-x86_64-standard:4.0 に設定
- env/privileged-mode が true に設定

これらのビルドは並行して実行されます。

ビルドマトリックスの buildspec 構文の詳細については、「<u>batch/build-matrix</u>」を参照してく ださい。

ファンアウトの構築

ビルドファンアウトは、バッチ内の複数のビルドに分割されるタスクを定義します。これは、テスト を並行して実行するために使用できます。CodeBuild は、 parallelismフィールドで設定された値 に基づいて、テストケースのシャードごとに個別のビルドを作成します。

次の例では、並行して実行される5つのビルドを作成するビルドファンアウトを定義します。

```
version: 0.2
batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false
phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run ∖
```

--test-command 'npx jest --runInBand --coverage' \
--files-search "codebuild-glob-search '**/test/**/*.test.js'" \
--sharding-strategy 'equal-distribution'

この例では、実行する必要があるテストが 100 件あると仮定して、CodeBuild は 5 つのビルドを作 成し、それぞれ 20 件のテストを並行して実行します。

ビルドグラフの buildspec 構文の詳細については、「<u>batch/build-fanout</u>」を参照してくださ い。

バッチレポートモード

プロジェクトのソースプロバイダーが Bitbucket、GitHub、または GitHub Enterprise であり、ソース プロバイダーにビルドステータスを報告するようにプロジェクトが設定されている場合は、ソースプ ロバイダーにバッチビルドステータスを送信する方法を選択できます。バッチに関する単一の集約ス テータスレポートとしてステータスを送信する、またはバッチ内の各ビルドのステータスを個別に報 告することを選択できます。

詳細については、以下の各トピックを参照してください。

- Batch 設定 (作成)
- Batch 設定 (更新)

詳細情報

詳細については、以下の各トピックを参照してください。

- バッチビルドのビルド仕様 (buildspec) のリファレンス
- Batch 構成
- バッチビルドの実行 (AWS CLI)
- でバッチビルドを停止する AWS CodeBuild

バッチビルドで並列テストを実行する

を使用して AWS CodeBuild 、バッチビルドで並列テストを実行できます。並列テスト実行は、複数 のテストケースを順番に実行するのではなく、異なる環境、マシン、またはブラウザ間で同時に実行 するテストアプローチです。このアプローチにより、全体的なテスト実行時間を大幅に短縮し、テス ト効率を向上させることができます。CodeBuild では、テストを複数の環境に分割し、同時に実行で きます。

並列テスト実行の主な利点は次のとおりです。

- 1. 実行時間の短縮 数時間かかるテストは、数分で完了できます。
- 2. リソース使用率の向上 利用可能なコンピューティングリソースを効率的に使用します。
- 以前のフィードバック テストの完了が早いほど、開発者へのフィードバックが迅速になります。
- 4. コスト効率 長期的に時間とコンピューティングコストの両方を節約します。

並列テストの実行を実装する場合、一般的に2つの主なアプローチとして、個別の環境とマルチス レッドが考慮されます。どちらの方法も同時テスト実行の実現を目指していますが、その実装と有効 性は大きく異なります。個別の環境では、各テストスイートが独立して実行される独立したインスタ ンスが作成されますが、マルチスレッドでは異なるスレッドを使用して同じプロセススペース内で複 数のテストが同時に実行されます。

マルチスレッドに比べて個別の環境の主な利点は次のとおりです。

- 1. 分離 各テストは完全に分離された環境で実行され、テスト間の干渉を防ぎます。
- 2. リソースの競合 マルチスレッドで頻繁に発生する共有リソースとの競合はありません。
- 3. 安定性 競合状態や同期の問題が発生しにくい。
- デバッグが容易 テストが失敗すると、各環境が独立しているため、原因を特定する方が簡単です。
- 5. 状態管理 マルチスレッドテストを悩ます共有状態の問題を簡単に管理できます。
- 6. スケーラビリティの向上 複雑さを伴わずに簡単に環境を追加できます。

トピック

- でのサポート AWS CodeBuild
- バッチビルドで並列テストの実行を有効にする
- codebuild-tests-run CLI コマンドを使用する
- codebuild-glob-search CLI コマンドを使用する
- テスト分割について
- 個々のビルドレポートを自動的にマージする。

さまざまなテストフレームワークのサンプルの並列テスト実行

でのサポート AWS CodeBuild

AWS CodeBuild は、個別の環境実行を活用するように特別に設計されたバッチビルド機能を通じ て、並列テスト実行の堅牢なサポートを提供します。この実装は、分離されたテスト環境の利点と完 全に一致します。

テストディストリビューションを使用したバッチビルド

CodeBuild のバッチビルド機能を使用すると、同時に実行される複数のビルド環境を作成できま す。各環境は、独自のコンピューティングリソース、ランタイム環境、依存関係を持つ完全に分 離されたユニットとして動作します。バッチビルド設定を使用して、必要な並列環境の数と、そ れらの間でテストを分散する方法を指定できます。

シャーディング CLI をテストする

CodeBuild には、CLI ツールである を通じてテスト分散メカニズムが組み込まれておりcodebuild-tests-run、テストを異なる環境に自動的に分割します。

レポートの集約

CodeBuild の実装の主な強みの1つは、テスト結果の集約をシームレスに処理できることです。 テストが別々の環境で実行される間、CodeBuild は各環境からテストレポートを自動的に収集 し、バッチビルドレベルで統合されたテストレポートに結合します。この統合により、並列実行 の効率上の利点を維持しながら、テスト結果を包括的に把握できます。

次の図は、 での並列テスト実行の完全な概念を説明しています AWS CodeBuild。



バッチビルドで並列テストの実行を有効にする

テストを並行して実行するには、次に示すように、バッチビルド buildspec ファイルを更新して build-fanout フィールドと、テストスイートを parallelismフィールドで分割する並列ビルドの数 を含めます。parallelism フィールドは、テストスイートを実行するためにセットアップされる独 立したエグゼキュターの数を指定します。

複数の並列実行環境でテストを実行するには、 parallelismフィールドを 0 より大きい値に設定し ます。以下の例では、 parallelismは 5 に設定されています。つまり、CodeBuild はテストスイー トの一部を並行して実行する 5 つの同一のビルドを開始します。

<u>codebuild-tests-run</u> CLI コマンドを使用して、テストを分割して実行できます。テストファイルは分 割され、テストの一部は各ビルドで実行されます。これにより、完全なテストスイートの実行にかか る全体的な時間が短縮されます。次の例では、テストは 5 つに分割され、分割ポイントはテストの 名前に基づいて計算されます。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
      - npm install jest-junit --save-dev
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - 1
        codebuild-tests-run ∖
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/_tests_/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - codebuild-glob-search '**/*.xml'
```

- echo "Runnin	g post-build	steps.
- echo "Build	completed on	`date`
reports:		
test-reports:		
files:		
- '**/junit.xm	11'	
<pre>base-directory:</pre>		
discard-paths: y	'es	
file-format: JUN	IITXML	

ビルドファンアウトビルド用にレポートが設定されている場合、テストレポートはビルドごとに個別 に生成され、 AWS CodeBuild コンソールの対応するビルドのレポートタブに表示されます。

••"

並列テストをバッチで実行する方法の詳細については、「」を参照してください<u>さまざまなテストフ</u> レームワークのサンプルの並列テスト実行。

codebuild-tests-run CLI コマンドを使用する

AWS CodeBuild は、テストコマンドとテストファイルの場所を入力として受け取る CLI を提供しま す。これらの入力を持つ CLI は、テストファイル名に基づいて、 parallelismフィールドで指定さ れたシャードの数にテストを分割します。テストファイルのシャードへの割り当ては、シャーディン グ戦略によって決まります。

```
codebuild-tests-run \
    --files-search "codebuild-glob-search '**/__tests__/*.js'" \
    --test-command 'npx jest --runInBand --coverage' \
    --sharding-strategy 'equal-distribution'
```

次の表に、 codebuild-tests-run CLI コマンドのフィールドを示します。

フィールド名	タイプ	必須またはオプショ ン	定義
test-command	String	必須	このコマンドは、テ ストの実行に使用さ れます。
files-search	String	必須	このコマンドは、テ ストファイルのリス

API バージョン 2016-10-06 618

フィールド名	タイプ	必須またはオプショ ン	定義
			トを提供します。 AWS CodeBuild 提供 されている <u>codebuild</u> <u>-glob-search</u> CLI コマ ンドまたは任意の他 のファイル検索ツー ルを使用できます。
			Note files-sea rch コマン ドがファイル 名を出力し、 それぞれが新 しい行で区切 られているこ とを確認しま す。

AWS CodeBuild

フィールド名	タイプ	必須またはオプショ ン	定義
sharding- strategy	列举型	オプションです。	有効な値: equal- distribution (デフォル ト)、stability ・ equal-dis tribution :テ ストファイル名に 基づいてテスト ファイルを均等に シャードします。 ・ stability : ファ イル名の一貫し たハッシュを使用 してテストファイ ルをシャードしま す。
			詳細については、「 <u>テスト分割について</u> 」を参照してくださ い。

CLIは、最初に codebuild-tests-run files-searchパラメータで指定された コマンドを使用 してテストファイルのリストを識別します。次に、指定されたシャーディング戦略を使用して、現 在のシャード (環境)に指定されたテストファイルのサブセットを決定します。最後に、このテスト ファイルのサブセットはスペース区切りリストにフォーマットされ、実行前に test-commandパラ メータで指定されたコマンドの末尾に追加されます。

スペース区切りリストを受け入れないテストフレームワークの場合、 CLI codebuild-testsrun は CODEBUILD_CURRENT_SHARD_FILES環境変数を通じて柔軟な代替手段を提供します。こ の変数には、現在のビルドシャードに指定されたテストファイルパスの改行区切りリストが含ま れています。この環境変数を活用することで、さまざまなテストフレームワーク要件に簡単に適 応し、スペース区切りリストとは異なる入力形式を想定する要件に対応できます。さらに、テス トフレームワークの必要性に応じてテストファイル名をフォーマットすることもできます。以下 は、Django フレームワークでの Linux CODEBUILD_CURRENT_SHARD_FILESでの の使用例です。 以下はCODEBUILD_CURRENT_SHARD_FILES、Django でサポートされているドット表記ファイルパ スを取得するために使用されます。

```
codebuild-tests-run \
    -files-search "codebuild-glob-search '/tests/test_.py'" \
    -test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
    -E "s/\/__/g; s/\.py$//; s/__/./g")' \
    -sharding-strategy 'equal-distribution'
```

```
    Note
```

CODEBUILD_CURRENT_SHARD_FILES 環境変数は CLI codebuild-tests-run の範囲内でのみ使用できることに注意してください。

また、test-command CODEBUILD_CURRENT_SHARD_FILES内で を使用している場合は、上 記の例に示すように二重引用符CODEBUILD_CURRENT_SHARD_FILESで囲んでください。

codebuild-glob-search CLI コマンドを使用する

AWS CodeBuild には、1 つ以上の glob パターンに基づいて作業ディレクトリ内のファイルを検 索codebuild-glob-searchできる という組み込み CLI ツールが用意されています。このツール は、プロジェクト内の特定のファイルまたはディレクトリでテストを実行する場合に特に便利です。

使用方法

CLI codebuild-glob-search には次の使用構文があります。

codebuild-glob-search <glob_pattern1> [<glob_pattern2> ...]

- <glob_pattern1>、 <glob_pattern2>など: 作業ディレクトリ内のファイルと一致する 1 つ以 上の glob パターン。
- ・ *: 任意の文字シーケンス (パス区切り文字を除く) に一致します。
- ・ **: 任意の文字シーケンス (パス区切り文字を含む) に一致します。

Note

glob 文字列に引用符があることを確認します。pattern-matching の結果を確認するには、 echo コマンドを使用します。

```
version: 0.2
phases:
    build:
        commands:
            echo $(codebuild-glob-search '**/_tests_/*.js')
                 codebuild-glob-search '**/_tests_/*.js' | xargs -n 1 echo
```

Output

CLI は、指定された glob パターンに一致するファイルパスの改行区切りリストを出力します。返さ れるファイルパスは、作業ディレクトリを基準としています。

指定されたパターンに一致するファイルが見つからない場合、CLI はファイルが見つからないことを 示すメッセージを出力します。

特定のパターンが原因で見つかったディレクトリは、検索結果から除外されることに注意してくださ い。

例

.js 拡張子が付けられたテストディレクトリとそのサブディレクトリ内のファイルのみを検索する 場合は、 CLI codebuild-glob-search で次のコマンドを使用できます。

codebuild-glob-search '**/__tests__/*.js'

このコマンドは、 パターンで示されるように、 __tests__ディレクトリとそのサブディレクトリ内 に.js拡張子を持つすべてのファイルを検索します。

テスト分割について

AWS CodeBuildのテスト分割機能を使用すると、複数のコンピューティングインスタンス間で テストスイートの実行を並列化できるため、全体的なテスト実行時間を短縮できます。この機 能は、CodeBuild プロジェクト設定のバッチ設定と buildspec ファイルの codebuild-testsrunユーティリティを通じて有効になります。

テストは、指定されたシャーディング戦略に基づいて分割されます。CodeBuild には、以下に示すよ うに 2 つのシャーディング戦略が用意されています。

等分散

equal-distribution シャーディング戦略は、テストファイル名のアルファベット順に基づい て、テストを並列ビルドに分割します。このアプローチでは、まずテストファイルをソートし、 次にチャンクベースのメソッドを使用してファイルを配布します。これにより、同様のファイル がテスト用にグループ化されます。比較的小さなテストファイルのセットを扱う場合は、をお勧 めします。この方法は、各シャードにほぼ等しい数のファイルを割り当てることを目指していま すが、最大差は1ですが、安定性を保証するものではありません。以降のビルドでテストファイ ルを追加または削除すると、既存のファイルの配布が変更され、シャード間で再割り当てされる 可能性があります。

安定性

stability シャーディング戦略では、一貫したハッシュアルゴリズムを使用してテストを シャードに分割し、ファイル配布が安定していることを確認します。新しいファイルを追加また は削除する場合、このアプローチにより、既存のfile-to-shard割り当てはほとんど変更されませ ん。大規模なテストスイートでは、安定性オプションを使用してテストをシャード間で均等に分 散することをお勧めします。このメカニズムは、ほぼ等しい分散を提供し、各シャードが最小の 分散で同じ数のファイルを受け取るようにすることを目的としています。安定性戦略は理想的な 等分散を保証するものではありませんが、ファイルが追加または削除された場合でも、ビルド間 でファイル割り当ての一貫性を維持するほぼ等しい分散を提供します。

テスト分割を有効にするには、CodeBuild プロジェクト設定でバッチセクションを設定し、必要 なparallelismレベルとその他の関連パラメータを指定する必要があります。さらに、適切なテス トコマンドと分割方法とともに、buildspec ファイルに codebuild-tests-runユーティリティを 含める必要があります。

個々のビルドレポートを自動的にマージする

ファンアウトバッチビルドでは、 は個々のビルドレポートを一括バッチレベルレポートに自動的に マージすること AWS CodeBuild をサポートします。この機能は、バッチ内のすべてのビルドのテス ト結果とコードカバレッジを包括的に表示します。

仕組み

fanout バッチビルドを実行すると、個々のビルドごとに<u>テストレポート</u>が生成されます。その 後、CodeBuild は異なるビルドの同じレポートを、バッチビルドにアタッチされた統合レポートに自 動的に統合します。これらの統合レポートは、<u>BatchGetBuildBatches</u> API の reportArnsフィール ドから簡単にアクセスでき、コンソールのレポートタブでも表示できます。このマージ機能は、自動 検出されたレポートにも拡張されます。

統合レポートは、buildspec で指定されているかCodeBuild によって自動検出された<u>レポートグルー</u> <u>プ</u>の下に作成されます。マージされたレポートの傾向をこれらのレポートグループの直下に分析し、 同じビルドバッチプロジェクトの過去のビルド全体のビルドパフォーマンスと品質メトリクスに関す る貴重なインサイトを提供できます。

バッチ内の個々のビルドごとに、CodeBuild は個別のレポートグループを自動的に作成します。 これらは特定の命名規則に従い、バッチビルドレポートグループ名と のサフィックスを組み合わ せます。 はBuildFanoutShard<shard_number>、レポートグループが作成されるシャードの 数shard_numberを表します。この組織では、統合ビルドレベルと個々のビルドレベルの両方で傾 向を追跡および分析できるため、ビルドプロセスを柔軟にモニタリングおよび評価できます。

バッチビルドレポートは、<u>個々のビルドレポート</u>と同じ構造に従います。レポートタブの次のキー フィールドは、バッチビルドレポートに固有です。

バッチビルドレポートのステータス

バッチビルドレポートのステータスは、レポートタイプに応じて特定のルールに従います。

テストレポート:

- 成功: すべての個々のビルドレポートが成功すると、ステータスは成功に設定されます。
- 失敗: 個々のビルドレポートが失敗した場合、ステータスは失敗に設定されます。
- 未完了:個々のビルドレポートがないか、ステータスが不完全である場合、ステータスは未 完了としてマークされます。
- コードカバレッジレポート:
 - 完了:ステータスは、個々のビルドレポートがすべて完了すると完了するように設定されます。
 - 失敗: 個々のビルドレポートが失敗した場合、ステータスは失敗に設定されます。
 - 未完了:個々のビルドレポートがないか、ステータスが不完全である場合、ステータスは未 完了としてマークされます。

テストの概要

マージされたテストレポートは、すべての個々のビルドレポートから次のフィールドを統合しま す。

- duration-in-nano-seconds: すべての個々のビルドレポートにおけるナノ秒単位の最大テスト期間。
- total: すべてのテストケースの合計数。各ビルドのテストの合計数を合計します。
- status-counts: 合格、不合格、スキップなどのテストステータスの統合ビューを提供し、すべてのビルドで各ステータスタイプのカウントを集計して計算します。

コードカバレッジの概要

マージされたコードカバレッジレポートは、次の計算を使用して、すべての個々のビルドの フィールドを組み合わせます。

- branches-covered: 個々のレポートのすべての対象ブランチの合計。
- branches-missed: 個々のレポートから欠落したすべてのブランチの合計。
- branch-coverage-percentage: (Total covered branches / Total branches) * 100
- lines-covered: 個々のレポートのすべての対象行の合計。
- lines-missed: 個々のレポートから欠落したすべての行の合計。
- lines-coverage-percentage: (Total covered lines / Total lines) * 100

実行 ID

バッチビルド ARN。

テストケース

マージされたレポートには、個々のビルドのすべてのテストケースの統合リストが含まれており、<u>DescribeTestCases</u> API と コンソールのバッチビルドレポートの両方からアクセスできます。

コードカバレッジ

マージされたコードカバレッジレポートは、すべての個々のビルドにわたる各ファイルの統合さ れたラインとブランチカバレッジ情報を提供し、<u>DescribeCodeCoverages</u> API と コンソールの バッチビルドレポートの両方からアクセスできます。注:異なるシャードに分散された複数のテ ストファイルの対象となるファイルの場合、マージされたレポートは次の選択基準を使用しま す。

1. プライマリ選択は、シャード間の最大のラインカバレッジに基づいています。

2. 複数のシャードでラインカバレッジが等しい場合、ブランチカバレッジが最も高いシャードが 選択されます。

さまざまなテストフレームワークのサンプルの並列テスト実行

codebuild-tests-run CLI コマンドを使用して、並列実行環境間でテストを分割して実行できま す。次のセクションでは、 codebuild-tests-run コマンドの使用法を示すさまざまなフレーム ワークbuildspec.ymlのサンプルを示します。

- ・以下の各例には5つのparallelismレベルが含まれています。つまり、テストを分割するために 5つの同一の実行環境が作成されます。build-fanout セクションparallelismの値を変更する ことで、プロジェクトに適したparallelismレベルを選択できます。
- ・以下の各例は、デフォルトでテストファイル名で分割するようにテストを設定する方法を示しています。これにより、テストが並列実行環境に均等に分散されます。

開始する前に、「」で詳細バッチビルドで並列テストを実行するを確認してください。

CLI コマンドを使用する際のオプションの完全なリストについては、codebuild-tests-run「」 を参照してくださいcodebuild-tests-run CLI コマンドを使用する。

トピック

- Django で並列テストを設定する
- Elixir で並列テストを設定する
- Go で並列テストを設定する
- Java (Maven) で並列テストを設定する
- Javascript (Jest) を使用した並列テストの設定
- Kotlin で並列テストを設定する
- PHPUnit を使用した並列テストの設定
- Pytest で並列テストを設定する
- Ruby (キューンバー) で並列テストを設定する
- <u>Ruby (RSpec) で並列テストを設定する</u>

Django で並列テストを設定する

Ubuntu プラットフォームでの Django による並列テストの実行buildspec.ymlを示す の例を次に 示します。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - sudo yum install -y python3 python3-pip
      - python3 -m ensurepip --upgrade
      - python3 -m pip install django
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Django Tests'
      - |
        codebuild-tests-run ∖
         --test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES"
 | sed -E "s/\//__/g; s/\.py$//; s/__/./g")' \
         --files-search "codebuild-glob-search '**/tests/*test_*.py'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo 'Test execution completed'
```

上記の例は、環境変数 の使用を示していますCODEBUILD_CURRENT_SHARD_FILES。ここで はCODEBUILD_CURRENT_SHARD_FILES、Django でサポートされているドット表記ファイルパスを 取得します。上記のように、二重引用符CODEBUILD_CURRENT_SHARD_FILES内で を使用します。

Elixir で並列テストを設定する

以下は、Ubuntu プラットフォームでの Elixir を使用した並列テストの実行buildspec.ymlを示す のサンプルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
phases:
  install:
    commands:
      - echo 'Installing Elixir dependencies'
      - sudo apt update
      - sudo DEBIAN_FRONTEND=noninteractive apt install -y elixir
      - elixir --version
      - mix --version
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Elixir Tests'
      - |
        codebuild-tests-run ∖
         --test-command 'mix test' \
         --files-search "codebuild-glob-search '**/test/**/*_test.exs'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

Go で並列テストを設定する

以下は、Linux プラットフォームでの Go を使用した並列テストの実行buildspec.ymlを示す のサ ンプルです。

version: 0.2

batch:

```
fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
      - echo 'Fetching Go version'
      - go version
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running go Tests'
      - go mod init calculator
      - cd calc
      - |
        codebuild-tests-run ∖
         --test-command "go test -v calculator.go" \
         --files-search "codebuild-glob-search '**/*test.go'"
  post_build:
    commands:
      - echo "Test execution completed"
```

上記の例では、 calculator.go関数にはテストする単純な数学関数が含まれており、すべてのテ ストファイルとcalculator.goファイルは calcフォルダ内にあります。

Java (Maven) で並列テストを設定する

以下は、Linux プラットフォームでの Java による並列テストの実行buildspec.ymlを示す のサン プルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
```

phases:

```
pre_build:
    commands:
        - echo 'prebuild'
build:
    commands:
        - echo "Running mvn test"
        - |
            codebuild-tests-run \
            --test-command 'mvn test -Dtest=$(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
"s|src/test/java/||g; s/\.java//g; s|/|.|g; s/ /,/g" | tr "\n" "," | sed "s/,$//")' \
            --files-search "codebuild-glob-search '**/test/**/*.java'"
post_build:
    commands:
        - echo "Running post-build steps..."
        - echo "Test execution completed"
```

特定の例では、 環境変数CODEBUILD_CURRENT_SHARD_FILESには、現在のシャード内のテスト ファイルが改行で区切られています。これらのファイルは、Maven の -Dtestパラメータで受け入 れられる形式のクラス名のカンマ区切りリストに変換されます。

Javascript (Jest) を使用した並列テストの設定

以下は、Ubuntu プラットフォームでの Javascript による並列テストの実行buildspec.ymlを示す のサンプルです。

```
version: 0.2
batch:
  fast-fail: true
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
        - echo 'Installing Node.js dependencies'
        - apt-get update
        - apt-get update
        - apt-get install -y nodejs
        - npm install
        - npm install
        - npm install --save-dev jest-junit
    pre_build:
```

```
commands:
    - echo 'prebuild'
build:
    commands:
    - echo 'Running JavaScript Tests'
    - |
      codebuild-tests-run \
      --test-command "npx jest" \
      --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
      --sharding-strategy 'stability'
post_build:
    commands:
    - echo 'Test execution completed'
```

Kotlin で並列テストを設定する

以下は、Linux プラットフォームでの Kotlin を使用した並列テストの実行buildspec.ymlを示す の サンプルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 2
    ignore-failure: false
phases:
  install:
    runtime-versions:
      java: corretto11
    commands:
      - echo 'Installing dependencies'
      - KOTLIN_VERSION="1.8.20" # Replace with your desired version
      - curl -o kotlin-compiler.zip -L "https://github.com/JetBrains/kotlin/releases/
download/v${KOTLIN_VERSION}/kotlin-compiler-${KOTLIN_VERSION}.zip"
      - unzip kotlin-compiler.zip -d /usr/local
      - export PATH=$PATH:/usr/local/kotlinc/bin
      - kotlin -version
      - curl -0 https://repo1.maven.org/maven2/org/junit/platform/junit-platform-
console-standalone/1.8.2/junit-platform-console-standalone-1.8.2.jar
  pre_build:
    commands:
```

```
- echo 'prebuild'
  build:
    commands:
      - echo 'Running Kotlin Tests'
      - |
        codebuild-tests-run ∖
          --test-command 'kotlinc src/main/kotlin/*.kt $(echo
 "$CODEBUILD_CURRENT_SHARD_FILES" | tr "\n" " ") -d classes -cp junit-platform-console-
standalone-1.8.2.jar' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
      - |
        codebuild-tests-run ∖
          --test-command '
            java -jar junit-platform-console-standalone-1.8.2.jar --class-path classes
 /
              $(for file in $CODEBUILD_CURRENT_SHARD_FILES; do
                 class_name=$(basename "$file" .kt)
                 echo "--select-class $class_name"
               done)
          ' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
  post_build:
    commands:
      - echo "Test execution completed"
```

上記の例では、 CLI codebuild-tests-run は 2 回使用されます。最初の実行時に、kotlinc は ファイルをコンパイルします。CODEBUILD_CURRENT_SHARD_FILES 変数は、現在のシャード に割り当てられたテストファイルを取得し、スペース区切りリストに変換します。2 回目の実行 では、JUnit がテストを実行します。ここでも、 CODEBUILD_CURRENT_SHARD_FILES は現在の シャードに割り当てられたテストファイルを取得しますが、今回はクラス名に変換されます。

PHPUnit を使用した並列テストの設定

以下は、Linux プラットフォームでの PHPUnit を使用した並列テストの実行buildspec.ymlを示す のサンプルです。

```
version: 0.2
batch:
   fast-fail: false
   build-fanout:
      parallelism: 5
```

```
ignore-failure: false
phases:
   install:
     commands:
       - echo 'Install dependencies'
       - composer require --dev phpunit/phpunit
   pre_build:
     commands:
       - echo 'prebuild'
   build:
     commands:
       - echo 'Running phpunit Tests'
       - composer dump-autoload
       - |
         codebuild-tests-run ∖
          --test-command "./vendor/bin/phpunit --debug" \
          --files-search "codebuild-glob-search '**/tests/*Test.php'"
   post_build:
       commands:
         - echo 'Test execution completed'
```

Pytest で並列テストを設定する

以下は、Ubuntu プラットフォームでの Pytest を使用した並列テストの実行buildspec.ymlを示す のサンプルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
        - echo 'Installing Python dependencies'
        - apt-get update
        - apt-get update
        - apt-get install -y python3 python3-pip
        - pip3 install --upgrade pip
        - pip3 install pytest
```
```
build:
    commands:
        - echo 'Running Python Tests'
        - |
        codebuild-tests-run \
            --test-command 'python -m pytest' \
            --files-search 'codebuild-glob-search 'tests/test_*.py'" \
            --sharding-strategy 'equal-distribution'
post_build:
    commands:
        - echo "Test execution completed"
```

以下は、Windows プラットフォームでの Pytest を使用した並列テストの実行buildspec.ymlを示 す のサンプルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - pip install pytest
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running pytest'
      - |
        & codebuild-tests-run `
         --test-command 'pytest @("$env:CODEBUILD_CURRENT_SHARD_FILES" -split \"`r?`n
\")'
         --files-search "codebuild-glob-search '**/test_*.py' '**/*_test.py'" `
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

上記の例では、CODEBUILD_CURRENT_SHARD_FILES環境変数を使用して、現在のシャードに割り 当てられ、配列として pytest コマンドに渡されるテストファイルを取得します。

Ruby (キューンバー) で並列テストを設定する

以下は、Linux プラットフォームでの Cucumber を使用した並列テストの実行buildspec.ymlを示すのサンプルです。

```
version: 0.2
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - gem install bundler
      - bundle install
  pre build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running Cucumber Tests'
      - cucumber --init
      - |
        codebuild-tests-run ∖
         --test-command "cucumber" \setminus
         --files-search "codebuild-glob-search '**/*.feature'"
  post_build:
    commands:
      - echo "Test execution completed"
```

Ruby (RSpec) で並列テストを設定する

以下は、Ubuntu プラットフォームでの RSpec を使用した並列テストの実行buildspec.ymlを示す のサンプルです。

version: 0.2

並列テスト実行サンプル

```
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - apt-get update
      - apt-get install -y ruby ruby-dev build-essential
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo 'Running Ruby Tests'
      - |
         codebuild-tests-run ∖
          --test-command 'bundle exec rspec' \
          --files-search "codebuild-glob-search 'spec/**/*_spec.rb'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

パフォーマンスを向上させるためのキャッシュビルド

キャッシュを使用すると、プロジェクトを構築する時間を短縮できます。キャッシュでは、ビルド 環境の再利用可能な部分が保存され、複数のビルドでそれらを使用することができます。ビルドプ ロジェクトでは、Amazon S3 とローカルの 2 種類のキャッシュのうち、いずれかを使用できます。 ローカルキャッシュを使用する場合は、3 つのキャッシュモード (ソースキャッシュ、Docker レイ ヤーキャッシュ、カスタムキャッシュ)のうち 1 つ以上を選択する必要があります。

Note

Docker レイヤーキャッシュモードは Linux 環境でのみ利用可能です。このモードを選択する 場合は、権限モードでビルドを実行する必要があります。CodeBuild のプロジェクトでは、 権限モードは、ビルドプロジェクトの Docker コンテナにすべてのデバイスへのアクセスを 許可します。詳細については、Docker Docs ウェブサイトの「<u>ランタイム特権と Linux 機</u> 能」を参照してください。

トピック

- Amazon S3 のキャッシュ
- ローカルキャッシュ
- ローカルキャッシュを指定

Amazon S3 のキャッシュ

Amazon S3 キャッシュでは、複数のビルドホスト間で利用できるキャッシュを Amazon S3 バケッ トに保存します。これは、ダウンロードするよりも構築にコストがかかる小規模から中間ビルドアー ティファクトに適したオプションです。

ビルドで Amazon S3 を使用するには、 にキャッシュするファイルのパスを指定できま すbuildspec.yml。CodeBuild はキャッシュを自動的に保存し、プロジェクトで設定された Amazon S3 の場所に更新します。ファイルパスを指定しない場合、CodeBuild はビルドの高速化に 役立つ一般的な言語の依存関係をベストエフォートキャッシュします。キャッシュの詳細は、ビルド ログで表示できます。

さらに、複数のバージョンのキャッシュが必要な場合は、 でキャッシュキーを定義できま すbuildspec.yml。CodeBuild はこのキャッシュキーのコンテキストにキャッシュを保存し、作成 後に更新されない一意のキャッシュコピーを作成します。キャッシュキーはプロジェクト間で共有 することもできます。動的キー、キャッシュバージョニング、ビルド間のキャッシュ共有などの機能 は、キーが指定されている場合にのみ使用できます。

buildspec ファイルのキャッシュ構文の詳細については、buildspec リファレンス<u>cache</u>の「」を参照 してください。

トピック

- 動的キーの生成
- codebuild-hash-files
- キャッシュバージョン
- プロジェクト間のキャッシュ共有
- Buildspec の例

動的キーの生成

キャッシュキーには、シェルコマンドと環境変数を含めて一意にすることができ、キーが変更された ときの自動キャッシュ更新を可能にします。たとえば、 package-lock.jsonファイルのハッシュ を使用してキーを定義できます。そのファイルの依存関係が変更されると、ハッシュ、つまりキャッ シュキーが変更され、新しいキャッシュの自動作成がトリガーされます。

cache:

key: npm-key-\$(codebuild-hash-files package-lock.json)

CodeBuild は式を評価し\$(codebuild-hash-files package-lock.json)て最終キーを取得し ます。

npm-key-abc123

などの環境変数を使用してキャッシュキーを定義することもできま すCODEBUILD_RESOLVED_SOURCE_VERSION。これにより、ソースが変更されるたびに新しいキー が生成され、新しいキャッシュが自動的に保存されます。

cache:

key: npm-key-\$CODEBUILD_RESOLVED_SOURCE_VERSION

CodeBuild は式を評価し、最終キーを取得します。

npm-key-046e8b67481d53bdc86c3f6affdd5d1afae6d369

codebuild-hash-files

codebuild-hash-files は、glob パターンを使用して CodeBuild ソースディレクトリ内の一連の ファイルの SHA-256 ハッシュを計算する CLI ツールです。

codebuild-hash-files <glob-pattern-1> <glob-pattern-2> ...

を使用した例をいくつか次に示しますcodebuild-hash-files。

```
codebuild-hash-files package-lock.json
codebuild-hash-files '**/*.md'
```

キャッシュバージョン

キャッシュバージョンは、キャッシュされるディレクトリのパスから生成されるハッシュです。2つ のキャッシュのバージョンが異なる場合、それらはマッチングプロセス中に個別のキャッシュとして 扱われます。たとえば、次の2つのキャッシュは異なるパスを参照するため、異なると見なされま す。

```
version: 0.2
phases:
    build:
        commands:
            - pip install pandas==2.2.3 --target pip-dependencies
cache:
    key: pip-dependencies
    paths:
        - "pip-dependencies/**/*"
```

プロジェクト間のキャッシュ共有

cache セクションの cacheNamespace API フィールドを使用して、複数のプロジェクト間で キャッシュを共有できます。このフィールドはキャッシュの範囲を定義します。キャッシュを共有す るには、 は以下を実行する必要があります。

- 同じを使用しますcacheNamespace。
- 同じキャッシュを指定しますkey。
- 同一のキャッシュパスを定義します。
- 同じ Amazon S3 バケットを使用し、pathPrefix設定されている場合は を使用します。

これにより、一貫性が確保され、プロジェクト間でキャッシュ共有が可能になります。

キャッシュ名前空間を指定する (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- 2. [プロジェクトを作成]を選択します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
- 3. Artifacts で、追加設定を選択します。
- 4. キャッシュタイプで、Amazon S3 を選択します。
- 5. キャッシュ名前空間 オプションで、名前空間値を入力します。
 - Additional configuration

Cache, encryption key

Encryption key - optional

Provide the AWS KMS customer master key used to encrypt this build's output artifacts. The default is your AWS-managed customer master key for S3.

arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>

Cache type

Amazon S3

Cache bucket

Q

Cache path prefix - optional

Cache lifecycle (days) - optional

You can apply a lifecycle expiration action to all or a subset of objects in the cache bucket based on the path prefix.

	Ŷ	+	Add expiration	
Cache namespace - optional				
test-cache-namespace				
Provide a cache namespace if you wa	it to share cache	es acro	ss projects.	

6. デフォルト値のまま続行し、[ビルドプロジェクトを作成する]を選択します。

キャッシュ名前空間を指定する (AWS CLI)

の --cacheパラメータを使用して AWS CLI 、キャッシュ名前空間を指定できます。

--cache '{"type": "S3", "location": "your-s3-bucket", "cacheNamespace": "test-cachenamespace"}'

Buildspec の例

一般的な言語の buildspec の例をいくつか示します。

トピック

- Node.js 依存関係をキャッシュする
- Python 依存関係をキャッシュする
- キャッシュ Ruby の依存関係
- キャッシュ Go の依存関係

Node.js 依存関係をキャッシュする

プロジェクトに package-lock json ファイルが含まれており、 npmを使用して Node.js の依存関 係を管理する場合、次の例はキャッシュを設定する方法を示しています。デフォルトでは、 は依存 関係を node_modules ディレクトリにnpmインストールします。

```
version: 0.2
phases:
    build:
        commands:
            - npm install
cache:
    key: npm-$(codebuild-hash-files package-lock.json)
    paths:
        - "node_modules/**/*"
```

Python 依存関係をキャッシュする

プロジェクトに requirements.txt ファイルが含まれ、pip を使用して Python の依存関係を管理 する場合、次の例はキャッシュを設定する方法を示しています。デフォルトでは、pip はシステムの site-packages ディレクトリにパッケージをインストールします。

さらに、依存関係を特定のディレクトリにインストールし、そのディレクトリのキャッシュを設定で きます。

```
version: 0.2
phases:
    build:
        commands:
            - pip install -r requirements.txt --target python-dependencies
cache:
    key: python-$(codebuild-hash-files requirements.txt)
    paths:
        - "python-dependencies/**/*"
```

キャッシュ Ruby の依存関係

プロジェクトに Gemfile.lock ファイルが含まれ、 Bundlerを使用して gem 依存関係を管理する 場合、次の例はキャッシュを効果的に設定する方法を示しています。

```
version: 0.2
phases:
    build:
        commands:
            - bundle install --path vendor/bundle
cache:
    key: ruby-$(codebuild-hash-files Gemfile.lock)
    paths:
            - "vendor/bundle/**/*"
```

キャッシュ Go の依存関係

プロジェクトに go.sum ファイルが含まれ、Go モジュールを使用して依存関係を管理する場合、次 の例はキャッシュを設定する方法を示しています。デフォルトでは、Go モジュールは \${GOPATH}/ pkg/mod ディレクトリにダウンロードされて保存されます。

```
version: 0.2
phases:
    build:
        commands:
            - go mod download
cache:
    key: go-$(codebuild-hash-files go.sum)
    paths:
            - "/go/pkg/mod/**/*"
```

ローカルキャッシュ

ローカルキャッシュは、そのビルドホストのみが利用できるキャッシュをそのビルドホストにローカ ルに保存します。キャッシュはビルドホストですぐに利用できるため、この方法は大規模から中間ビ ルドアーティファクトに適しています。ビルドの頻度が低い場合、これは最適なオプションではあり ません。つまり、ビルドパフォーマンスはネットワーク転送時間の影響を受けません。

ローカルキャッシングを選択した場合は、次のキャッシュモードを 1 つ以上選択する必要がありま す。

- ソースキャッシュモードは、プライマリソースとセカンダリソースの Git メタデータをキャッシュします。キャッシュ作成後のビルドでは、コミット間の変更のみプルされます。このモードは、クリーンな作業ディレクトリと、大きな Git リポジトリであるソースを持つプロジェクトに適しています。このオプションを選択しても、プロジェクトで Git リポジトリ (AWS CodeCommit、GitHub、GitHub Enterprise Server、または Bitbucket) を使用しない場合、このオプションは無視されます。
- Docker レイヤーキャッシュモードは、既存の Docker レイヤーをキャッシュします。このモードは、大きな Docker イメージを構築または取得するプロジェクトに適しています。そのため、大きな Docker イメージをネットワークからプルすることによって生じるパフォーマンス上の問題を回避できます。

Note

- Docker レイヤーキャッシュは Linux 環境でのみ使用できます。
- プロジェクトに必要な Docker アクセス許可が付与されるように、privileged フラグ を設定する必要があります。

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビ ルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime</u> <u>Privilege and Linux Capabilities</u>」を参照して、特権モードを有効にします。ま た、Windows は特権モードをサポートしていません。

Docker レイヤーキャッシュを使用する前に、セキュリティへの影響を考慮してください。

カスタムキャッシュモードは buildspec ファイルで指定したディレクトリをキャッシュします。このシナリオは、ビルドシナリオが他の2つのローカルキャッシュモードのいずれにも適していない場合に適しています。カスタムキャッシュを使用する場合:

- キャッシュに指定できるのはディレクトリのみです。個々のファイルを指定することはできません。
- キャッシュされたディレクトリを参照するには、シンボリックリンクを使用します。
- キャッシュされたディレクトリは、プロジェクトソースをダウンロードする前にビルドにリン クされます。キャッシュされたアイテムにより、同じ名前のソースアイテムが上書きされます。 ディレクトリは buildspec ファイルのキャッシュパスを使って指定されます。詳細については、 「buildspec の構文」を参照してください。
- ソースとキャッシュで同じディレクトリ名は使用しないでください。ローカルにキャッシュされたディレクトリにより、ソースリポジトリ内の同じ名前のディレクトリの内容が上書きまたは削除される場合があります。

Note

ローカルキャッシュは、環境タイプ LINUX_GPU_CONTAINER とコンピューティングタイプ BUILD_GENERAL1_2XLARGE ではサポートされていません。詳細については、「<u>ビルド環境</u> のコンピューティングモードおよびタイプ」を参照してください。 (i) Note

VPC で動作するように CodeBuild を設定する場合、ローカルキャッシュはサポートされま せん。CodeBuild で VPC を使用する方法については、「<u>Amazon Virtual Private Cloud AWS</u> CodeBuild で を使用する」を参照してください。

ローカルキャッシュを指定

AWS CLI、コンソール、SDK、または AWS CloudFormation を使用して、ローカルキャッシュを指 定できます。ローカルキャッシュの詳細については、「ローカルキャッシュ」を参照してください。

トピック

- ローカルキャッシュの指定 (CLI)
- ローカルキャッシュの指定 (コンソール)
- ・ ローカルキャッシュの指定 (AWS CloudFormation)

ローカルキャッシュの指定 (CLI)

の --cacheパラメータを使用して AWS CLI、3 つのローカルキャッシュタイプをそれぞれ指定でき ます。

ソースキャッシュを指定するには:

--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]

• Docker レイヤーキャッシュを指定するには:

--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]

カスタムキャッシュを指定するには:

--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]

詳細については、「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」を参照してください。

ローカルキャッシュを指定

ローカルキャッシュの指定 (コンソール)

キャッシュは、コンソールの [アーティファクト] セクションで指定します。[Cache type] (キャッ シュタイプ) で、[Amazon S3] または [Local] (ローカル) を選択します。[ローカル] を選択した場合 は、3 つのローカルキャッシュオプションのうち、1 つ以上を選択します。

Cache type
Local
Select one or more local cache options.
Docker layer cache Caches existing Docker layers so they can be reused. Requires privileged mode.
Caches .git metadata so subsequent builds only pull the change in commits.
Custom cache Caches directories specified in the buildspec file.

詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

ローカルキャッシュの指定 (AWS CloudFormation)

AWS CloudFormation を使用してローカルキャッシュを指定する場合は、 Cacheプロパティの に Typeを指定しますLOCAL。次の YAML 形式の AWS CloudFormation コード例では、3 つのローカ ルキャッシュタイプをすべて指定します。任意のタイプの組み合わせを指定できます。Docker レ イヤーキャッシュを使用する場合は、Environment で、PrivilegedMode を true、Type を LINUX_CONTAINER に設定する必要があります。

```
CodeBuildProject:

Type: AWS::CodeBuild::Project

Properties:

Name: MyProject

ServiceRole: <service-role>

Artifacts:

Type: S3

Location: <bucket-name>

Name: myArtifact

EncryptionDisabled: true

OverrideArtifactName: true

Environment:

Type: LINUX_CONTAINER
```

ComputeType: BUILD_GENERAL1_SMALL Image: aws/codebuild/standard:5.0 Certificate: <bucket/cert.zip> # PrivilegedMode must be true if you specify LOCAL_DOCKER_LAYER_CACHE PrivilegedMode: true Source: Type: GITHUB Location: <github-location> InsecureSsl: true GitCloneDepth: 1 ReportBuildStatus: false TimeoutInMinutes: 10 Cache: Type: LOCAL Modes: # You can specify one or more cache mode, - LOCAL_CUSTOM_CACHE - LOCAL_DOCKER_LAYER_CACHE - LOCAL_SOURCE_CACHE

Note

デフォルトでは、Docker デーモンは非 VPC ビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「<u>Runtime Privilege and</u> <u>Linux Capabilities</u>」を参照して、特権モードを有効にします。また、Windows は特権モード をサポートしていません。

詳細については、「ビルドプロジェクトの作成 (AWS CloudFormation)」を参照してください。

でのビルドのデバッグ AWS CodeBuild

AWS CodeBuild には、開発とトラブルシューティング中にビルドをデバッグするための 2 つの方法 があります。CodeBuild サンドボックス環境を使用して問題を調査し、修正をリアルタイムで検証で きます。または、 AWS Systems Manager Session Manager を使用してビルドコンテナに接続し、 コンテナの状態を表示できます。

CodeBuild サンドボックスを使用したビルドのデバッグ

CodeBuild サンドボックス環境は、安全で隔離された環境でインタラクティブなデバッグセッション を提供します。環境と直接やり取りするには AWS CLI、 AWS Management Console または を使用 してコマンドを実行し、ビルドプロセスをステップバイステップで検証します。コスト効果の高い 1 秒あたりの請求モデルを使用し、ビルド環境と同じソースプロバイダーや AWS サービスとのネイ ティブ統合をサポートします。SSH クライアントを使用してサンドボックス環境に接続するか、統 合開発環境 (IDEs) からサンドボックス環境に接続することもできます。

CodeBuild サンドボックス料金の詳細については、<u>CodeBuild 料金ドキュメント</u>を参照してくださ い。詳細な手順については、 <u>CodeBuild サンドボックスを使用したビルドのデバッグ</u>ドキュメント を参照してください。

Session Manager を使用したビルドのデバッグ

AWS Systems Manager Session Manager を使用すると、実際の実行環境で実行中のビルドに直接ア クセスできます。このアプローチにより、アクティブなビルドコンテナに接続し、ビルドプロセスを リアルタイムで検査できます。ファイルシステムを調べ、実行中のプロセスを監視し、問題が発生し たときにトラブルシューティングできます。

詳細な手順については、 <u>Session Manager を使用したビルドのデバッグ</u>ドキュメントを参照してく ださい。

CodeBuild サンドボックスを使用したビルドのデバッグ

では AWS CodeBuild、CodeBuild サンドボックスを使用してカスタムコマンドを実行し、ビルドを トラブルシューティングすることで、ビルドをデバッグできます。

トピック

- 前提条件
- CodeBuild サンドボックスを使用したビルドのデバッグ (コンソール)
- CodeBuild サンドボックスを使用したビルドのデバッグ (AWS CLI)
- チュートリアル: SSH を使用したサンドボックスへの接続
- AWS CodeBuild サンドボックス SSH 接続の問題のトラブルシューティング

前提条件

CodeBuild サンドボックスを使用する前に、CodeBuild サービスロールに次の SSM ポリシーがある ことを確認してください。

{

Session Manager を使用したビルドのデバッグ

```
"Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
          "ssmmessages:CreateControlChannel",
          "ssmmessages:CreateDataChannel",
          "ssmmessages:OpenControlChannel",
          "ssmmessages:OpenDataChannel"
       ],
       "Resource": "*"
     },
     {
       "Effect": "Allow",
       "Action": [
          "ssm:StartSession"
       ],
       "Resource": [
          "arn:aws:codebuild:<region>:<account-id>:build/*",
          "arn:aws:ssm:<region>::document/AWS-StartSSHSession"
       ]
     }
   ]
}
```

CodeBuild サンドボックスを使用したビルドのデバッグ (コンソール)

次の手順を使用してコマンドを実行し、コンソールで CodeBuild サンドボックスで SSH クライアントを接続します。

CodeBuild サンドボックスでコマンドを実行する (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>.com で開きます。
- 2. ナビゲーションペインで、[ビルドプロジェクト] を選択します。ビルドプロジェクトを選択し、デバッグビルドを選択します。

sandbox-project	Actions v Create trigger Edit	Clone Clear cache Debug build	Start build with overrides Start build
Configuration			
Source provider No source	Primary repository -	Artifacts upload location -	Service role am:aws:lam: role/codebuild-sandbox-project-service-role
Public builds Disabled			
Build history Batch history	Project details Build triggers Metri	cs Debug sessions	
Project configuration			Edit
Name sandbox-project		Description -	
Project ARN	:project/sandbox-project	Build badge Disabled	

3. Run command タブで、カスタムコマンドを入力し、Run command を選択します。

Debug build	
Run Command SSH Client Session Manager	
 Run custom commands with sandbox Launches a sandbox environment mirroring your project configuration. Automatically downloads source code, while skipping project buildspec execution. Ideal for reproducing failure, experimenting fixes and investigtion. 	Learn more [2]
Command 1 pwd	
⊗0 ∆0	<u>1:4</u> SH
Run command	

4. CodeBuild サンドボックスが初期化され、カスタムコマンドの実行が開始されます。出力が完了 すると、出力タブに表示されます。

Run Command SSH Client Session Manager	
Sandbox is running Your sandbox sandbox-project:ef8f3204-a9e8-4707-afcf-b4bb49b6bc18 is ready and available for use.	Stop sandbox
Command	
1 pwd	
⊗o ∆o	<u>1:1</u> SH
Run command	
Command output Sandbox phases Sandbox logs Sandbox configurations Command history	
View entire log in <u>CloudWatch console</u> 1 /codebuild/output/src3141870147/src	

5. トラブルシューティングが完了したら、サンドボックスを停止を選択してサンドボックスを停止できます。次に、Stopを選択してサンドボックスが停止することを確認します。

Stop sandbox	×
Stopping this sandbox will terminate all active sessions and running comm sure you want to stop the sandbox?	ands. Are you
 sandbox-project:ef8f3204-a9e8-4707-afcf-b4bb49b6bc18 	
Cancel	Stop

Debug build
Run Command SSH Client Session Manager
Sandbox is stopped Your sandbox sandbox-project:ef8f3204-a9e8-4707-afcf-b4bb49b6bc18 is currently inactive.
Command output Sandbox phases Sandbox logs Sandbox configurations Command history
View entire log in <u>CloudWatch console</u>
1 /codebuild/output/src3141870147/src 2

CodeBuild サンドボックスを使用して SSH クライアントに接続する (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- 2. ナビゲーションペインで、[ビルドプロジェクト] を選択します。ビルドプロジェクトを選択し、ビルドのデバッグを選択します。

sandbox-project	Actions Create trigger Ec	dit Clone Clear cache Debug build	d Start build with overrides Start build
Configuration			
Source provider No source	Primary repository -	Artifacts upload location -	Service role am:aws:lam:::::role/service- role/codebuild-sandbox-project-service-role
Public builds Disabled			
Build history Batch history	Project details Build triggers Me	etrics Debug sessions	
Project configuration			Edit
Name sandbox-project		Description -	
Project ARN	:project/sandbox-project	Build badge Disabled	

3. SSH クライアントタブで、開始サンドボックスを選択します。

Developer Tools > CodeBuild > Build projects > sandbox-project > Debug build	
Debug build	
Run Command SSH Client Session Manager	
Connect to your SSH client with sandbox	Learn more [2]
Launches a sandbox environment with SSH connectivity.	
Connect anectly using 55H calents of your preferred lbE.	
	Start sandbox

4. CodeBuild サンドボックスの実行が開始されたら、コンソールの指示に従って SSH クライアン トをサンドボックスに接続します。

Debug build	
Run Command SSH Client Session Manager	
Sandbox is running Your sandbox sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8 is ready and available for use.	Stop sandbox
Terminal Visual Studio Code IntelliJ IDEA	
Linux macOS Windows	
If you haven't done so already, paste and execute the following command in macOS Terminal. For more information about using SSH, see documentation page [.	
<pre>curl -0 https://codefactory-us-east-1-prod-default-build-agent-executor.s3.us-east-1.amazonaws.com/mac-sandbox-ssh.sh chmod +x mac-sandbox-ssh.sh ./mac-sandbox-ssh.sh rm mac-sandbox-ssh.sh</pre>	Ē
Make sure your CLI user has the codebuild: StartSandboxConnection permission. For more information, see AWS CLI authentication [documentation.	
Connect to your sandbox environment with following command:	
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1: :sandbox/sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8	G

5. トラブルシューティングが完了したら、サンドボックスを停止を選択してサンドボックスを停止できます。次に、Stopを選択してサンドボックスが停止することを確認します。



CodeBuild サンドボックスを使用したビルドのデバッグ (AWS CLI)

次の手順を使用してコマンドを実行し、SSH クライアントを CodeBuild サンドボックスに接続しま す。

CodeBuild サンドボックスを起動する (AWS CLI)

CLI command

aws codebuild start-sandbox --project-name \$PROJECT_NAME

• --project-name : CodeBuild プロジェクト名

Sample request

```
aws codebuild start-sandbox --project-name "project-name"
```

```
{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "QUEUED",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
        "cloudWatchLogs": {
```

```
"status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "currentPhase": "QUEUED",
        "status": "QUEUED",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }
```

}

サンドボックスのステータスに関する情報を取得する (AWS CLI)

CLI command

```
aws codebuild batch-get-sandboxes --ids $SANDBOX_IDs
```

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

• --ids:sandboxIdsまたはのカンマ区切りリストsandboxArns。

サンドボックス ID またはサンドボックス ARN を指定できます。

サンドボックス ID: <codebuild-project-name>:<UUID>

例えば、project-name:d25be134-05cb-404a-85da-ac5f85d2d72c。

 サンドボックス ARN: arn:aws:codebuild:<region> : <account-id>:sandbox/<codebuildproject-name> : <UUID>

例えば、arn:aws:codebuild:us-west-2:962803963624:sandbox/projectname:d25be134-05cb-404a-85da-ac5f85d2d72c。

```
{
    "sandboxes": [{
        "id": "project-name",
        "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
        "projectName": "project-name",
        "requestTime": "2025-02-06T11:24:15.560000-08:00",
        "endTime": "2025-02-06T11:39:21.587000-08:00",
        "status": "STOPPED",
        "source": {
            "type": "S3",
        "
}
```

```
"location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
            "insecureSsl": false
        },
        "environment": {
            "type": "LINUX CONTAINER",
            "image": "aws/codebuild/standard:6.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "environmentVariables": [{
                    "name": "foo",
                    "value": "bar",
                    "type": "PLAINTEXT"
                },
                {
                    "name": "bar",
                    "value": "baz",
                    "type": "PLAINTEXT"
                }
            ],
            "privilegedMode": false,
            "imagePullCredentialsType": "CODEBUILD"
        },
        "timeoutInMinutes": 10,
        "queuedTimeoutInMinutes": 480,
        "logConfig": {
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        },
        "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/
SampleEncryptionKey",
        "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
        "currentSession": {
            "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "currentPhase": "COMPLETED",
            "status": "STOPPED",
```

```
"startTime": "2025-02-06T11:24:15.626000-08:00",
"endTime": "2025-02-06T11:39:21.600000-08:00",
"phases": [{
        "phaseType": "SUBMITTED",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2025-02-06T11:24:15.577000-08:00",
        "endTime": "2025-02-06T11:24:15.606000-08:00",
        "durationInSeconds": 0
   },
    {
        "phaseType": "QUEUED",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2025-02-06T11:24:15.606000-08:00",
        "endTime": "2025-02-06T11:24:16.067000-08:00",
        "durationInSeconds": 0
   },
    {
        "phaseType": "PROVISIONING",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2025-02-06T11:24:16.067000-08:00",
        "endTime": "2025-02-06T11:24:20.519000-08:00",
        "durationInSeconds": 4,
        "contexts": [{
            "statusCode": "",
            "message": ""
        }]
   },
   {
        "phaseType": "DOWNLOAD_SOURCE",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2025-02-06T11:24:20.519000-08:00",
        "endTime": "2025-02-06T11:24:22.238000-08:00",
        "durationInSeconds": 1,
        "contexts": [{
            "statusCode": "",
            "message": ""
        }]
   },
    {
        "phaseType": "RUNNING_SANDBOX",
        "phaseStatus": "TIMED_OUT",
        "startTime": "2025-02-06T11:24:22.238000-08:00",
        "endTime": "2025-02-06T11:39:21.560000-08:00",
        "durationInSeconds": 899,
```

```
"contexts": [{
                        "statusCode": "BUILD_TIMED_OUT",
                        "message": "Build has timed out. "
                    }]
                },
                {
                    "phaseType": "COMPLETED",
                    "startTime": "2025-02-06T11:39:21.560000-08:00"
                }
            ],
            "logs": {
                "groupName": "group",
                "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/
home?region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    }],
    "sandboxesNotFound": []
}
```

サンドボックスを停止する (AWS CLI)

CLI command

aws codebuild stop-sandbox --id \$SANDBOX-ID

・ --id:AsandboxIdまたはsandboxArn。

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

```
{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "STOPPING",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
```

```
"privilegedMode": false,
    "imagePullCredentialsType": "CODEBUILD"
},
"timeoutInMinutes": 10,
"queuedTimeoutInMinutes": 480,
"logConfig": {
    "cloudWatchLogs": {
        "status": "ENABLED",
        "groupName": "group",
        "streamName": "stream"
    },
    "s3Logs": {
        "status": "ENABLED",
        "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
        "encryptionDisabled": false
    }
},
"encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
"serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
"currentSession": {
    "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
    "currentPhase": "RUN_SANDBOX",
    "status": "STOPPING",
    "startTime": "2025-02-06T11:24:15.626000-08:00",
    "phases": [{
            "phaseType": "SUBMITTED",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-08T14:33:26.144000-08:00",
            "endTime": "2025-02-08T14:33:26.173000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "QUEUED",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-08T14:33:26.173000-08:00",
            "endTime": "2025-02-08T14:33:26.702000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "PROVISIONING",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-08T14:33:26.702000-08:00",
            "endTime": "2025-02-08T14:33:30.530000-08:00",
            "durationInSeconds": 3,
```

```
"contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
                "phaseType": "DOWNLOAD_SOURCE",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:30.530000-08:00",
                "endTime": "2025-02-08T14:33:33.478000-08:00",
                "durationInSeconds": 2,
                "contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
                "phaseType": "RUN_SANDBOX",
                "startTime": "2025-02-08T14:33:33.478000-08:00"
            }
        ],
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
```

```
}
}
```

コマンド実行を開始する (AWS CLI)

}

CLI command

```
aws codebuild start-command-execution --command $COMMAND --type $TYPE --sandbox-id
$SANDBOX-ID
```

- --command: 実行する必要があるコマンド。
- ・ --sandbox-id: A sandboxIdまたは sandboxArn。
- --type:コマンドタイプ SHELL。

Sample request

```
aws codebuild start-command-execution --command "echo "Hello World"" --type SHELL --
sandbox-id "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name
```

```
{
    "id": "e1c658c2-02bb-42a8-9abb-94835241fcd6",
    "sandboxId": "f7126a4a-b0d5-452f-814c-fea73718f805",
    "submitTime": "2025-02-06T20:12:02.683000-08:00",
    "status": "SUBMITTED",
    "command": "echo \"Hello World\"",
    "type": "SHELL",
    "logs": {
        "groupName": "group",
        "streamName": "stream",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
        "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-
pool-1-us-west-2-beta-default-build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz?
region=us-west-2",
        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
```

```
"s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-default-
build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz",
    "cloudWatchLogs": {
        "status": "ENABLED",
        "groupName": "group",
        "streamName": "stream"
     },
     "s3Logs": {
        "status": "ENABLED",
        "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
        "encryptionDisabled": false
     }
  }
}
```

コマンド実行に関する情報を取得する (AWS CLI)

CLI command

```
aws codebuild batch-get-command-executions --command-execution-ids $COMMAND-IDs --
sandbox-id $SANDBOX-IDs
```

- --command-execution-ids:のカンマ区切りリストcommandExecutionIds。
- ・ --sandbox-id: A sandboxIdまたは sandboxArn。

Sample request

```
aws codebuild batch-get-command-executions --command-execution-
ids"c3c085ed-5a8f-4531-8e95-87d547f27ffd" --sandbox-id "arn:aws:codebuild:us-
west-2:962803963624:sandbox/project-name"
```

```
{
    "commandExecutions": [{
        "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
        "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
        "submitTime": "2025-02-10T20:18:17.118000-08:00",
        "startTime": "2025-02-10T20:18:17.939000-08:00",
        "endTime": "2025-02-10T20:18:17.976000-08:00",
```

```
"status": "SUCCEEDED",
        "command": "echo \"Hello World\"",
        "type": "SHELL",
        "exitCode": "0",
        "standardOutputContent": "Hello World\n",
        "logs": {
            "groupName": "group",
            "streamName": "stream",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }],
    "commandExecutionsNotFound": []
}
```

サンドボックスのコマンド実行を一覧表示する (AWS CLI)

CLI command

aws codebuild list-command-executions-for-sandbox --sandbox-id \$SANDBOX-ID --nexttoken \$NEXT_TOKEN --max-results \$MAX_RESULTS --sort-order \$SORT_ORDER

--next-token:ページ分割された結果を取得するための次のトークンがある場合。この値は、リストサンドボックスの以前の実行から取得されます。

- ・ --max-results: (オプション) 取得するサンドボックスレコードの最大数。
- --sort-order:サンドボックスレコードを取得する順序。

Sample request

```
aws codebuild list-command-executions-for-sandbox --sandbox-id
"arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

```
{
    "commandExecutions": [{
            "id": "aad6687e-07bc-45ab-a1fd-f5440229b528",
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:35.304000-08:00",
            "startTime": "2025-02-10T20:18:35.615000-08:00",
            "endTime": "2025-02-10T20:18:35.651000-08:00",
            "status": "FAILED",
            "command": "fail command",
            "type": "SHELL",
            "exitCode": "127",
            "standardErrContent": "/codebuild/output/tmp/script.sh: 4: fail: not
 found\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
```

```
"location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        },
        {
            "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:17.118000-08:00",
            "startTime": "2025-02-10T20:18:17.939000-08:00",
            "endTime": "2025-02-10T20:18:17.976000-08:00",
            "status": "SUCCEEDED",
            "command": "echo \"Hello World\"",
            "type": "SHELL",
            "exitCode": "0",
            "standardOutputContent": "Hello World\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    1
```

サンドボックスを一覧表示する (AWS CLI)

CLI command

```
aws codebuild list-sandboxes --next-token $NEXT_TOKEN --max-results $MAX_RESULTS --
sort-order $SORT_ORDER
```

Sample request

aws codebuild list-sandboxes

```
{
    "ids": [
        "s3-log-project-integ-test-temp173925062814985d64e0f-7880-41df-9a3c-
fb6597a266d2:827a5243-0841-4b69-a720-4438796f6967",
        "s3-log-project-integ-test-temp1739249999716bbd438dd-8bb8-47bd-
ba6b-0133ac65b3d3:e2fa4eab-73af-42e3-8903-92fddaf9f378",
        "s3-log-project-integ-test-
temp17392474779450fbdacc2-2d6e-4190-9ad5-28f891bb7415:cd71e456-2a4c-4db4-ada5-
da892b0bba05",
        "s3-log-project-integ-test-temp17392246284164301421c-5030-4fa1-b4d3-
ca15e44771c5:9e26ab3f-65e4-4896-a19c-56b1a95e630a",
        "s3-log-project-integ-test-temp173921367319497056d8d-6d8e-4f5a-a37c-
a62f5686731f:22d91b06-df1e-4e9c-a664-c0abb8d5920b",
        "s3-log-project-integ-test-temp1739213439503f6283f19-390c-4dc8-95a9-
c8480113384a:82cc413e-fc46-47ab-898f-ae23c83a613f",
        "s3-log-project-integ-test-temp1739054385570b1f1ddc2-0a23-4062-
bd0c-24e9e4a99b99:c02562f3-2396-42ec-98da-38e3fe5da13a",
        "s3-log-project-integ-test-temp173905400540237dab1ac-1fde-4dfb-a8f5-
c0114333dc89:d2f30493-f65e-4fa0-a7b6-08a5e77497b9",
        "s3-log-project-integ-test-
temp17390534055719c534090-7bc4-48f1-92c5-34acaec5bf1e:df5f1c8a-f017-43b7-91ba-
ad2619e2c059",
        "s3-log-project-integ-test-temp1739052719086a61813cc-
ebb9-4db4-9391-7f43cc984ee4:d61917ec-8037-4647-8d52-060349272c4a",
        "s3-log-project-integ-test-temp173898670094078b67edb-
c42f-42ed-9db2-4b5c1a5fc66a:ce33dfbc-beeb-4466-8c99-a3734a0392c7",
```
```
"s3-log-project-integ-test-
temp17389863425584d21b7cd-32e2-4f11-9175-72c89ecaffef:046dadf0-1f3a-4d51-a2c0-
e88361924acf",
        "s3-log-project-integ-test-
temp1738985884273977ccd23-394b-46cc-90d3-7ab94cf764dc:0370dc41-9339-4b0a-91ed-51929761b244",
        "s3-log-project-integ-test-temp1738985365972241b614f-8e41-4387-
bd25-2b8351fbc9e0:076c392a-9630-47d8-85a9-116aa34edfff",
        "s3-log-project-integ-test-
temp1738985043988a51a9e2b-09d6-4d24-9c3c-1e6e21ac9fa8:6ea3949c-435b-4177-
aa4d-614d5956244c",
        "s3-log-project-integ-test-temp1738984123354c68b31ad-49d1-4f4b-981d-
b66c00565ff6:6c3fff6c-815b-48b5-ada3-737400a6dee8",
        "s3-log-project-integ-test-
temp1738977263715d4d5bf6c-370a-48bf-8ea6-905358a6cf92:968a0f54-724a-42d1-9207-6ed854b2fae8",
        "s3-log-project-integ-test-
temp173897358796816ce8d7d-2a5e-41ef-855b-4a94a8d2795d:80f9a7ce-930a-402e-934e-
d8b511d68b04",
        "s3-log-project-integ-test-temp17389730633301af5e452-0966-467c-
b684-4e36d47f568c:cabbe989-2e8a-473c-af25-32edc8c28646",
        "s3-log-project-integ-test-temp1738901503813173fd468-
b723-4d7b-9f9f-82e88d17f264:f7126a4a-b0d5-452f-814c-fea73718f805",
        "s3-log-project-integ-test-temp1738890502472c13616fb-
bd0f-4253-86cc-28b74c97a0ba:c6f197e5-3a53-45b6-863e-0e6353375437",
        "s3-log-project-integ-test-
temp17388903044683610daf3-8da7-43c6-8580-9978432432ce:d20aa317-8838-4966-
bbfc-85b908213df1",
        "s3-log-project-integ-test-temp173888857196780b5ab8b-e54b-44fd-a222-
c5a374fffe96:ab4b9970-ffae-47a0-b3a8-7b6790008cad",
        "s3-log-project-integ-test-temp1738888336931c11d378d-e74d-49a4-
a723-3b92e6f7daac:4922f0e8-9b7d-4119-9c9f-115cd85e703e",
        "s3-log-project-integ-test-temp17388881717651612a397-c23f-4d88-
ba87-2773cd3fc0c9:be91c3fc-418e-4feb-8a3a-ba58ff8f4e8a",
        "s3-log-project-integ-test-
temp17388879727174c3c62ed-6195-4afb-8a03-59674d0e1187:a48826a8-3c0d-43c5-
a1b5-1c98a0f978e9",
        "s3-log-project-integ-test-temp1738885948597cef305e4-b8b4-46b0-a65b-
e2d0a7b83294:c050e77d-e3f8-4829-9a60-46149628fe96",
        "s3-log-project-integ-test-temp173888561463001a7d2a8-
e4e4-4434-94db-09d3da9a9e17:8c3ac3f5-7111-4297-aec9-2470d3ead873",
        "s3-log-project-integ-test-
temp1738869855076eb19cafd-04fe-41bd-8aa0-40826d0c0d27:d25be134-05cb-404a-85da-
ac5f85d2d72c",
        "s3-project-integ-test-temp1738868157467148eacfc-d39b-49fc-a137-
e55381cd2978:4909557b-c221-4814-b4b6-7d9e93d37c35",
```



チュートリアル: SSH を使用したサンドボックスへの接続

このチュートリアルでは、SSH クライアントを使用して CodeBuild サンドボックスに接続する方法 を示します。

このチュートリアルを完了するには、まず以下を行う必要があります。

- 既存の AWS CodeBuild プロジェクトがあることを確認します。
- CodeBuild プロジェクトロールに設定された適切な IAM アクセス許可を設定します。
- ・ ローカルマシン AWS CLI に をインストールして設定します。

ステップ 1: サンドボックスを開始する

コンソールで CodeBuild サンドボックスを起動するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.https://www.com で開きます。
- 2. ナビゲーションペインで、[ビルドプロジェクト] を選択します。ビルドプロジェクトを選択し、デバッグビルドを選択します。

Developer Tools > CodeBuild > Build	projects > sandbox-project							
sandbox-project		Actions v	Create trigger	Edit	Clone	Debug build	Start build with overrides	Start build
Configuration								
Source provider No source	Primary repository -		Artifacts -	ipload locat	ion	S	ervice role rn:aws:lam::012345678910:role/se odebuild-sandbox-project-service-	ervice-role/ role
Public builds Disabled								
Build history Batch history	Project details Build triggers	Metrics	Debug sessions					
Project configuration								Edit
Name sandbox-project			Descriptic	n				
Project ARN	5678910:project/sandbox-project		Build bad Disabled	ge				

3. SSH クライアントタブで、開始サンドボックスを選択します。

Developer Tools > CodeBuild > Build projects > sandbox-project > Debug build	
Debug build	
Run Command SSH Client Session Manager	
Connect to your SSH client with sandbox Learn m • Launches a sandbox environment with SSH connectivity. • Connect directly using SSH clients or your preferred IDE.	more [2]
Sta	rt sandbox

 サンドボックスの初期化プロセスには時間がかかる場合があります。サンドボックスのステータ スがに変わったら、サンドボックスに接続できますRUN_SANDDBOX。

eloper Tools > CodeBuild >	Build projects > sandbox	-project > Debug build				
bug build						
tun Command SSH Clier	nt Session Manager					
Sandbox is running Your sandbox sandbox	x-project:253616fd-9624-43	34e-bb9a-bbe52620d256 is	ready and available	for use.		Stop sandbox
Terminal Visual Studi	o Code IntelliJ IDEA					
Linux macOS	Windows	eules command la macOS T	united for more in	formation about using SSU and downwardship	[7]	
curl -0 https://code chmod +x mac-sandbox- ./mac-sandbox-ssh.sh rm mac-sandbox-ssh.sh	factory-us-east-1-prod -ssh.sh n	-default-build-agent-e	xecutor.s3.us-	east-1.amazonaws.com/mac-sandbox-ssh	. sh	Ē
Make sure your CLI user has Connect to your sandbox er	the codebuild:StartSand	lboxConnection permissior	. For more informa	tion, see AWS CLI authentication [documental	tion.	
ssh codebuild-sandbox	x-ssh=arn:aws:codebuil	d:us-east-1:0123456789	10:sandbox/san	dbox-project:253616fd-9624-434e-bb9a	-bbe52620d256	
Sandbox phases Sand	lbox logs Sandbox con	figurations				
Name	Status	Context	Duration	Start time	End time	
SUBMITTED	⊘ Succeeded	-	<1 sec	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 P	M (UTC-7:00)
	Succeeded	-	<1 sec	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 P	M (UTC-7:00)
QUEUED						
QUEUED PROVISIONING	Succeeded		4 secs	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 P	M (UTC-7:00)
QUEUED PROVISIONING DOWNLOAD_SOURCE	SucceededSucceeded		4 secs 6 secs	Apr 1, 2025 4:33 PM (UTC-7:00) Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 P Apr 1, 2025 4:33 P	M (UTC-7:00) M (UTC-7:00)

ステップ 2: ローカル SSH 設定を変更する

サンドボックスに初めて接続する場合は、次の手順を使用して 1 回限りのセットアッププロセスを 実行する必要があります。

コンソールでローカル SSH 設定を変更するには

- 1. オペレーティングシステムのセットアップコマンドを見つけます。
- ローカルターミナルを開き、提供されたコマンドをコピーして実行し、スクリプトをダウンロードして実行し、ローカル SSH 設定をセットアップします。たとえば、オペレーティングシステムが macOS の場合は、次のコマンドを使用します。

Linux macOS Windows	
If you haven't done so already, paste and execute the following command in macOS Terminal. For more information about using SSH, see documentation page [].	
<pre>curl -0 https://codefactory-us-east-1-prod-default-build-agent-executor.s3.us-east-1.amazonaws.com/mac-sandbox-ssh.sh chmod +x mac-sandbox-ssh.sh ./mac-sandbox-ssh.sh rm mac-sandbox-ssh.sh</pre>	

- 設定スクリプトは、サンドボックスに接続するために必要な設定を追加します。これらの変更を 受け入れるように求められます。
- 4. 設定が成功すると、CodeBuild サンドボックスの新しい SSH 設定エントリが作成されます。

Host codebuild-sandbox-ssh*	
StrictHostKeyChecking no	
LogLevel INFO	
ForwardAgent yes	Star V
ControlMaster auto	
ControlPersist 10m	
ProxyCommand sh -c "/Users/	'.aws/codebuild-dev-env/codebuild-sandbox-connect.sh %n"

ステップ 3: サンドボックスに接続する

コンソールでローカル SSH 設定を変更するには

- AWS CLI 認証を設定し、AWS CLI ユーザーに アクセ スcodebuild:StartSandboxConnection許可があることを確認します。詳細については、 「バージョン <u>1 コマンドラインインターフェイスユーザーガイド」の「の IAM ユーザー認証</u> 情報を使用した認証 AWS CLI」を参照してください。AWS
- 2. 次のコマンドを使用してサンドボックスに接続します。

ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<accountid>:sandbox/<sandbox-id>

Note

接続障害のトラブルシューティングを行うには、 -vフラグを使用して詳細な出力を有効 にします。例えば、ssh -v codebuild-sandbox-ssh=arn:aws:codebuild:useast-1:<account-id>:sandbox/<sandbox-id>。 その他のトラブルシューティングガイダンスについては、「」を参照してくださいAWS CodeBuild サンドボックス SSH 接続の問題のトラブルシューティング。

ステップ 4: 結果を確認する

接続すると、ビルド障害のデバッグ、ビルドコマンドのテスト、設定変更の実験、サンドボックスで の環境変数と依存関係の検証を行うことができます。 AWS CodeBuild サンドボックス SSH 接続の問題のトラブルシューティング

このトピックの情報を使用して、CodeBuild サンドボックス SSH 接続の問題を特定、診断、および 対処できます。

トピック

- <u>StartSandboxConnectionInvalidInputException CodeBuild サンドボックス環境への SSH 時の工</u> ラー
- エラー: CodeBuild サンドボックス環境への SSH 時に「認証情報が見つかりません」
- <u>StartSandboxConnectionAccessDeniedException CodeBuild サンドボックス環境への SSH 時のエ</u>ラー
- エラー: CodeBuild サンドボックス環境への SSH 時に「ssh: ホスト名を解決できませんでした」

StartSandboxConnectionInvalidInputException CodeBuild サンドボックス環境への SSH 時のエラー

問題: コマンド を使用して CodeBuild サンドボックス環境に接続しようとするとssh codebuild-sandbox-ssh=<*sandbox-arn*>、次のようなInvalidInputExceptionエラーが発 生することがあります。

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection
operation: Failed to start SSM session for {sandbox-arn}
User: arn:aws:sts::<account-ID>:assumed-role/<service-role-name>/AWSCodeBuild-<UUID>
is not authorized to perform: ssm:StartSession on resource.
```

An error occurred (InvalidInputException) when calling the StartSandboxConnection operation: Failed to start SSM session for sandbox <sandbox-arn>: codebuild:<UUID> is not connected.

考えられる原因:

- Amazon EC2 Systems Manager Agent が見つからない: ビルドイメージに SSM エージェントが正しくインストールまたは設定されていません。
- アクセス許可が不十分: CodeBuild プロジェクトサービスロールに必要な SSM アクセス許可がありません。

推奨される解決策:ビルドにカスタムイメージを使用している場合は、次の操作を行います。

- 1. SSM Agent をインストールします。詳細については、<u>『』の「Linux 用 Amazon EC2 インスタ</u> <u>ンスに SSM エージェントを手動でインストールおよびアンインストール</u>する」を参照してくだ さい。SSM エージェントのバージョンは 3.0.1295.0 以降である必要があります。
- ファイル <u>https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/7.0/</u> <u>amazon-ssm-agent.json</u>://www.com」をイメージ内の /etc/amazon/ssm/ ディレクトリにコ ピーします。これにより、SSM エージェントでコンテナモードが有効になります。
- CodeBuild プロジェクトのサービスロールに次のアクセス許可があることを確認し、サンドボックス環境を再起動します。

```
{
   "Effect": "Allow",
      "Action": [
         "ssmmessages:CreateControlChannel",
         "ssmmessages:CreateDataChannel",
         "ssmmessages:OpenControlChannel",
         "ssmmessages:OpenDataChannel"
      ],
      "Resource": "*"
},
 {
    "Effect": "Allow",
    "Action": [
       "ssm:StartSession"
     ],
     "Resource": [
        "arn:aws:codebuild:region:account-id:build/*",
        "arn:aws:ssm:region::document/AWS-StartSSHSession"
     ]
 }
```

エラー: CodeBuild サンドボックス環境への SSH 時に「認証情報が見つかりません」

問題: コマンド を使用して CodeBuild サンドボックス環境に接続しようとするとssh codebuild-sandbox-ssh=<*sandbox-arn*>、次の認証情報エラーが発生することがあります。

```
Unable to locate credentials. You can configure credentials by running "aws configure".
```

考えられる原因: AWS 認証情報がローカル環境で正しく設定されていません。

推奨される解決策: 公式ドキュメント「 バージョン 2 の コマンドラインインターフェイスユーザー ガイド」の<u>「 の設定 AWS CLI</u>」に従って AWS CLI 認証情報を設定します。 AWS

StartSandboxConnectionAccessDeniedException CodeBuild サンドボックス環境への SSH 時のエラー

問題: コマンド を使用して CodeBuild サンドボックス環境に接続しようとするとssh codebuild-sandbox-ssh=<*sandbox-arn*>、次のアクセス許可エラーが発生することがありま す。

An error occurred (AccessDeniedException) when calling the StartSandboxConnection operation: User: arn:aws:sts::account-id:assumed-role/role-name is not authorized to perform: codebuild:StartSandboxConnection on resource: sandbox-arn because no identity-based policy allows the codebuild:StartSandboxConnection action

考えられる原因: AWS 認証情報に、このオペレーションを実行するために必要な CodeBuild アクセ ス許可がありません。

推奨される解決策: AWS CLI 認証情報に関連付けられた IAM ユーザーまたはロールに次のアクセス 許可があることを確認します。

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:StartSandboxConnection"
    ],
    "Resource": [
        "arn:aws:codebuild:region:account-id:sandbox/*"
    ]
}
```

エラー: CodeBuild サンドボックス環境への SSH 時に「ssh: ホスト名を解決できませんでした」

問題: コマンド を使用して CodeBuild サンドボックス環境に接続しようとするとssh codebuild-sandbox-ssh=<*sandbox-arn*>、次のホスト名解決エラーが発生します。

ssh: Could not resolve hostname

考えられる原因: このエラーは通常、必要な CodeBuild サンドボックス接続スクリプトがローカル 環境で正しく実行されていない場合に発生します。

推奨される解決策:

- 1. CodeBuild サンドボックス接続スクリプトをダウンロードします。
- 2. ターミナルでスクリプトを実行して、必要な SSH 設定を確立します。
- 3. サンドボックス環境への SSH 接続を再試行します。

Session Manager を使用したビルドのデバッグ

では AWS CodeBuild、実行中のビルドを一時停止し、 AWS Systems Manager Session Manager を 使用してビルドコンテナに接続し、コンテナの状態を表示できます。

Note

この機能は、Windows 環境では使用できません。

トピック

- <u>前提条件</u>
- ビルドの一時停止
- ビルドを開始します
- ビルドコンテナに接続する
- ビルドを再開する

前提条件

ビルドセッションでセッションマネージャーを使用できるようにするには、ビルドのセッション接続 を有効にする必要があります。次の 2 つの前提条件があります。

CodeBuild Linux 標準キュレーションイメージには、すでに SSM エージェントがインストールされており、SSM エージェント コンテナモードが有効になっています。

ビルドにカスタムイメージを使用している場合は、次の操作を行います。

- SSM Agent をインストールします。詳細については、 AWS Systems Manager ユーザーガイ ドの「<u>Linux 用 EC2 インスタンスに SSM Agent を手動でインストールする</u>」を参照してくだ さい。SSM Agent は、バージョン 3.0.1295.0 以降である必要があります。
- イメージ内の /etc/amazon/ssm/ ディレクトリにファイル <u>https://github.com/aws/aws-</u> codebuild-docker-images/blob/master/ubuntu/standard/5.0/amazon-ssm-agent.json://www.jp をコピーします。これにより、SSM エージェントでコンテナモードがイネーブルになりま す。

Note

この機能が正常に動作するには、カスタムイメージに最新の SSM エージェントが必要で す。

• CodeBuild サービスロールには、次の SSM ポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
          "ssmmessages:CreateControlChannel",
          "ssmmessages:CreateDataChannel",
          "ssmmessages:OpenControlChannel",
          "ssmmessages:OpenDataChannel"
        ],
        "Resource": "*"
     }
  ]
}
```

CodeBuild コンソールは、ビルドの開始時にこのポリシーをサービスロールに自動的にアタッチす るように設定できます。または、このポリシーを手動でサービスロールにアタッチすることもでき ます。

 セッションアクティビティのログ記録と監査を Systems Manager 設定で有効にしている場合 は、CodeBuild サービスロールにも追加のアクセス許可が必要です。アクセス許可は、ログが格納 されている場所によって異なります。

[CloudWatch Logs]

CloudWatch Logs を使用してログを保存する場合は、CodeBuild サービスロールに次のアクセ ス権限を追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:DescribeLogGroups",
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      1,
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:<log-group-
name>:*"
    }
  ]
}
```

Amazon S3

Amazon S3 を使用してログを保存する場合は、CodeBuild サービスロールに次のアクセス権限 を追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:GetEncryptionConfiguration",
               "s3:PutObject"
        ],
            "Resource": [
               "arn:aws:s3:::<bucket-name>",
               "arn:aws:s3:::<bucket-name>/*"
```

] }

詳細については、AWS Systems Manager ユーザーガイドの「<u>セッションアクティビティのログ記</u> 録と監査」を参照してください。

ビルドの一時停止

ビルドを一時停止するには、buildspec ファイルのビルドフェーズのいずれかで codebuildbreakpoint コマンドを実行します。この時点でビルドは一時停止されます。これにより、ビルドコン テナに接続し、コンテナを現在の状態で表示できます。

たとえば、buildspec ファイルのビルドフェーズに、以下を追加します。

phases: pre_build: commands: - echo Entered the pre_build phase... - echo "Hello World" > /tmp/hello-world codebuild-breakpoint

このコードは、/tmp/hello-worldファイルを作成し、この時点でビルドを一時停止します。

ビルドを開始します

ビルドセッションでセッションマネージャーを使用できるようにするには、ビルドのセッション接続 を有効にする必要があります。これを行うには、ビルドを開始するときに、以下の手順を実行しま す。

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- ナビゲーションペインで、[ビルドプロジェクト] を選択します。ビルドプロジェクトを選択した 後、[Start build with overrides] を選択します。
- 3. [Advanced build overrides (高度なビルドの上書き)] を選択します。
- [Environment] セクションで、[Enable session connection] オプションを選択します。このオプ ションが選択されていない場合、codebuild-breakpoint および codebuild-resume コマンドは無 視されます。

- 5. その他の必要な変更を行い、[Start build] を選択します。
- 6. コンソールでビルドステータスを監視します。セッションが利用可能になると、AWS セッショ ンマネージャーリンクが [Build status] セクションに表示されます。

ビルドコンテナに接続する

ビルドコンテナには、次の2つのいずれかに接続できます。

CodeBuild コンソール

ウェブブラウザで、AWS セッションマネージャーリンクをクリックして、ビルドコンテナに接 続します。ターミナルセッションが開き、ビルドコンテナを表示して制御できます。

AWS CLI

Note

この手順を実行するには、ローカルマシンにセッションマネージャプラグインがインス トールされている必要があります。詳細については、「 AWS Systems Manager ユー ザーガイド<u>」の「 CLI AWS 用の Session Manager プラグインのインストール</u>」を参照し てください。

batch-get-builds APIを呼び出し、ビルドIDに置き換えて、セッションターゲット識別子を含むビルドに関する情報を取得します。セッションターゲット識別子のプロパティ名は、awsコマンドの出力タイプによって異なります。これが、コマンドに --output json が追加される理由です。

aws codebuild batch-get-builds --ids
 solution --region
 region --output json

- プロパティの値 sessionTarget をコピーします。sessionTarget プロパティ名は、aws コマンドの出力タイプによって異なる場合があります。これが、前のステップでコマンドに --output json が追加される理由です。
- 3. ビルドコンテナに接続するには、次のコマンドを使用します。

aws ssm start-session --target <sessionTarget> --region <region>

この例では、/tmp/hello-worldファイルが存在し、Hello World テキストを含む検証です。

ビルドを再開する

ビルドコンテナーを調べ終わったら、codebuild-resume コマンドをコンテナーシェルから実行します。

\$ codebuild-resume

でビルドを削除する AWS CodeBuild

AWS CLI または AWS SDKsを使用してビルドを削除できます AWS CodeBuild。

トピック

- ビルドの削除 (AWS CLI)
- ビルドの削除 (AWS SDKs)

ビルドの削除 (AWS CLI)

batch-delete-builds コマンドを実行します。

aws codebuild batch-delete-builds --ids ids

上記のコマンドで、次のプレースホルダを置き換えます。

- ids: 必須の文字列。削除するビルドの ID。複数のビルドを指定するには、各ビルド ID をスペースで区切ります。ビルド ID のリストを取得するには、次のトピックを参照してください。
 - ビルド ID の一覧表示 (AWS CLI)
 - ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

成功すると、buildsDeleted 配列が出力に表示されます。この配列には、正常に削除された各ビ ルドの Amazon リソースネーム (ARN) が含まれています。正常に削除されなかったビルドに関する 情報は、出力の buildsNotDeleted 配列内に表示されます。

たとえば、次のコマンドを実行するとします。

aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX 次のような情報が出力に表示されます。

```
{
    "buildsNotDeleted": [
    {
        "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-build-
project:f8b888d2-5e1e-4032-8645-b115195648EX",
        "statusCode": "BUILD_IN_PROGRESS"
    }
    ],
    "buildsDeleted": [
        "arn:aws:codebuild:us-west-2:123456789012:build/my-other-demo-build-
project:a18bc6ee-e499-4887-b36a-8c90349c7eEX"
    ]
}
```

ビルドの削除 (AWS SDKs)

SDK AWS CodeBuild で を使用する方法については、「」を参照してください<u>AWS SDKsとツール</u> <u>のリファレンス</u>。 AWS SDKs

でビルドを手動で再試行する AWS CodeBuild

AWS CodeBuild コンソール、 AWS CLI、または AWS SDKs を使用して、1 つのビルドまたはバッ チビルドを手動で再試行できます AWS CodeBuild。

トピック

- ・ ビルドを手動で再試行 (コンソール)
- ビルドを手動で再試行 (AWS CLI)
- ・ビルドを手動で再試行する (AWS SDKs)

ビルドを手動で再試行 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. 次のいずれかを行います:
 - [build-project-name:build-ID] ページが表示された場合は、[ビルドの再試行] を選択します。

- ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボック スを選択後、[ビルドの再試行] を選択します。
- ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの再試行] を選択します。
- i Note

デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さ らに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択す るか、前後の矢印を使用します。

ビルドを手動で再試行 (AWS CLI)

• retry-build コマンドを実行します。

aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>

上記のコマンドで、次のプレースホルダを置き換えます。

- <build-id>: 必須の文字列。再試行するビルドまたはバッチビルドの ID。ビルド ID のリストを取得するには、次のトピックを参照してください。
 - ビルド ID の一覧表示 (AWS CLI)
 - バッチビルド ID のリストを表示 (AWS CLI)
 - ・ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)
 - ビルドプロジェクトのバッチビルド ID のリストを表示 (AWS CLI)
- --idempotency-token: オプション。オプションを指定して retry-build コマンドを実行する と、大文字と小文字を区別する一意の識別子 (トークン) が retry-build リクエストに含ま れます。このトークンは、 リクエスト後 5 分間有効です。同じトークンで retry-build リ クエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラー を返します。

ビルドを手動で再試行する (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> <u>ツールのリファレンス</u>。 AWS SDKs

でビルドを自動的に再試行する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、、または AWS SDKs を使用して、ビルドを自動的に再試行 できます AWS CodeBuild。自動再試行を有効にすると、CodeBuild は、ビルドが失敗した後、指定 された制限回数までプロジェクトのサービスロールを使用して自動的に RetryBuild を呼び出しま す。例えば、自動再試行の制限が2に設定されている場合、CodeBuild は RetryBuild API を呼び 出して、さらに最大2回までビルドを自動的に再試行します。

1 Note

CodeBuild は CodePipeline の自動再試行をサポートしていません。

トピック

- ビルドを自動的に再試行 (コンソール)
- ・ ビルドを自動的に再試行 (AWS CLI)
- ビルドを自動的に再試行する (AWS SDKs)

ビルドを自動的に再試行 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- [プロジェクトを作成]を選択します。詳細については、「ビルドプロジェクトの作成 (コンソー ル)」および「ビルドの実行 (コンソール)」を参照してください。
 - [環境] で以下の操作を行います。
 - [自動再試行の制限] には、ビルドが失敗した後に希望する自動再試行の最大回数を入力しま す。
- 3. [環境] で、[追加設定] を選択します。
- 4. デフォルト値のまま続行し、[ビルドプロジェクトを作成する] を選択します。

ビルドを自動的に再試行 (AWS CLI)

• create-project コマンドを実行します。

```
aws codebuild create-project \
    --name "<project-name>" \
    --auto-retry-limit <auto-retry-limit> \
    --source "<source>" \
    --artifacts {<artifacts>} \
    --environment "{\"type\": \"environment-type>\",\"image\": \"image-type>\",
    \"computeType\": \"compute-type>\"}" \
    --service-role "service-role>"
```

上記のコマンドで、次のプレースホルダを置き換えます。

- <auto-retry-limit>: 自動再試行の制限を、ビルドが失敗した後に希望する自動再試行の 最大回数に設定します。
- <project-name>、<source>、<artifacts>、<environment-type>、<imagetype>、<compute-type>、<service-role>:希望するプロジェクト設定を構成します。

ビルドを自動的に再試行する (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> ツールのリファレンス。 AWS SDKs

でビルドを停止する AWS CodeBuild

AWS CodeBuild コンソール、 AWS CLI、または AWS SDKs を使用して、 でビルドを停止できます AWS CodeBuild。

トピック

- ビルドの停止 (コンソール)
- ビルドの停止 (AWS CLI)
- <u>ビルドの停止 (AWS SDKs)</u>

ビルドの停止 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. 次のいずれかを行います:
 - [build-project-name:build-ID] ページが表示された場合は、[ビルドの停止] を選択します。
 - ・ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボック スを選択後、[ビルドの停止] を選択します。
 - ・ ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。

Note

デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さ らに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択す るか、前後の矢印を使用します。

がビルドを正常に停止 AWS CodeBuild できない場合 (ビルドプロセスがすでに完了している 場合など)、停止ボタンが無効になっているか、表示されないことがあります。

ビルドの停止 (AWS CLI)

stop-build コマンドを実行します。

aws codebuild stop-build --id id

上記のコマンドで、次のプレースホルダを置き換えます。

- *id*: 必須の文字列。停止するビルドの ID。ビルド ID のリストを取得するには、次のトピック を参照してください。
 - ビルド ID の一覧表示 (AWS CLI)
 - ・ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

AWS CodeBuild ビルドが正常に停止した場合、出力の build オブジェクトのbuildStatus値 は ですST0PPED。

CodeBuild がビルドを正常に停止できない場合 (たとえば、ビルドがすでに完了している場合)、build オブジェクトの出力の オブジェクトの buildStatus 値が最終的なビルドステー タス (例: SUCCEEDED) になります。

ビルドの停止 (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> <u>ツールのリファレンス</u>。 AWS SDKs

でバッチビルドを停止する AWS CodeBuild

AWS CodeBuild コンソール、 AWS CLI、または AWS SDKs を使用して、 でバッチビルドを停止で きます AWS CodeBuild。

Note

バッチビルドで Lambda コンピューティングを使用する場合、進行中の Lambda ビルドを停止することはできません。

トピック

- バッチビルドの停止 (コンソール)
- バッチビルドを停止 (AWS CLI)
- バッチビルドの停止 (AWS SDKs)

バッチビルドの停止 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. 次のいずれかを行います:

- [build-project-name:build-ID] ページが表示された場合は、[ビルドの停止] を選択します。
- ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボック スを選択後、[ビルドの停止] を選択します。
- ・ ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。

(i) Note

デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さ らに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択す るか、前後の矢印を使用します。

バッチビルドを停止 (AWS CLI)

• stop-build-batch コマンドを実行します。

aws codebuild stop-build-batch --id <batch-build-id>

上記のコマンドで、次のプレースホルダを置き換えます。

- <batch-build-id>: 必須の文字列。停止するバッチビルドの ID。バッチビルド ID のリスト を取得するには、次のトピックを参照してください。
 - バッチビルド ID のリストを表示 (AWS CLI)
 - ・ ビルドプロジェクトのバッチビルド ID のリストを表示 (AWS CLI)

バッチビルドの停止 (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> <u>ツールのリファレンス</u>。 AWS SDKs

AWS CodeBuild ビルドを自動的にトリガーする

プロジェクトでトリガーを作成し、1 時間、1 日、または 1 週間に 1 回ビルドをスケジュールできま す。Amazon CloudWatch cron 式でカスタムルールを使用してトリガーを編集することもできます。 たとえば、cron 式を使用して、毎週特定の時間にビルドをスケジュールできます。トリガーの作成 および編集に関する詳細は、「<u>AWS CodeBuild トリガーの作成</u>」および「<u>AWS CodeBuild トリガー</u> の編集」を参照してください。

トピック

- AWS CodeBuild トリガーの作成
- AWS CodeBuild トリガーの編集

AWS CodeBuild トリガーの作成

プロジェクトでトリガーを作成し、1 時間、1 日、または 1 週間に 1 回ビルドをスケジュールできま す。Amazon CloudWatch cron 式でカスタムルールを使用してトリガーを作成することもできます。 たとえば、cron 式を使用して、毎週特定の時間にビルドをスケジュールできます。

Note

ビルドトリガー、Amazon EventBridge イベント、または AWS Step Functions タスクから バッチビルドを開始することはできません。

トピック

- AWS CodeBuild トリガーの作成 (コンソール)
- プログラムで AWS CodeBuild トリガーを作成する

AWS CodeBuild トリガーの作成 (コンソール)

次の手順で、 AWS Management Consoleを使用してトリガーを作成します。

トリガーを作成するには

1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https

- 2. ナビゲーションペインで、[ビルドプロジェクト] を選択します。
- トリガーを追加するビルドプロジェクトのリンクを選択し、[トリガーのビルド] タブを選択します。

Note

デフォルトでは、最新の 100 個のビルドプロジェクトが表示されます。さらに多くのビ ルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ 毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

- 4. [Create trigger (トリガーの作成)] を選択します。
- 5. [トリガー名]に名前を入力します。
- 6. [Frequency] (頻度) ドロップダウンリストから、トリガーの頻度を選択します。CRON 式を使用 して頻度を作成する場合は、[Custom] (カスタム) を選択します。
- トリガーの頻度のパラメータを指定します。選択肢の最初の数文字をテキストボックスに入力すると、ドロップダウンメニュー項目がフィルタリングされます。

Note

開始時間と分はゼロベースです。開始分は 0 から 59 までの数値です。開始時は 0 か ら 23 までの数値です。たとえば、毎日午後 12:15 に開始する日次トリガーは、開始時 が 12、開始分が 15 になります。毎日深夜に開始される日次トリガーは、開始時がゼロ で、開始分がゼロです。毎日午後 11:59 に開始する毎日のトリガーは、開始時が 23、開 始分が 59 です。

Frequency	必須パラメータ	詳細
時間単価	開始時間 (分)	[開始時間 (分)] ドロップダウ ンメニューを使用します。
1日1回	開始時間 (分) 開始時間 (時)	[開始時間 (分)] ドロップダウ ンメニューを使用します。 [開始時間 (時)] ドロップダウ ンメニューを使用します。

Frequency	必須パラメータ	詳細
毎週	開始時間 (分) 開始時間 (時) 開始日	[開始時間 (分)] ドロップダウ ンメニューを使用します。 [開始時間 (時)] ドロップダウ ンメニューを使用します。 [開始時間 (日)] ドロップダウ ンメニューを使用します。
Custom	Cron 式	[Cron 式] に Cron 式を入力 します。Cron 式には、空 白で区切られた6つの必須 フィールドがあります。こ れらのフィールドでは、分、 時、日付、月、曜日、および 年の開始値を指定します。 範囲や追加の値などを指定 するには、ワイルドカード を使用します。例えば、cron 式09?* MON-FRI *は 毎週平日午前9時にビルド をスケジュールします。詳 細については、「Amazon CloudWatch Events ユー ザーガイド」の「 <u>cron 式</u> 」 を参照してください。

- 8. [Enable this trigger (このトリガーの有効化)] を選択します。
- 9. (オプション) [アドバンスト] セクションを展開します。[ソースバージョン] に、ソースのバー ジョンを入力します。
 - Amazon S3 の場合、ビルドする入力アーティファクトのバージョンに対応するバージョン ID を入力します。[ソースバージョン] が空白のままの場合は、最新バージョンが使用されます。
 - には AWS CodeCommit、コミット ID を入力します。[ソースバージョン] が空白のままの場合は、デフォルトブランチの HEAD コミット ID が使用されます。

- GitHub または GitHub Enterprise の場合は、ビルドするソースコードのバージョンに対応する コミット ID、プルリクエスト ID、ブランチ名、またはタグ名を入力します。プルリクエスト ID を指定する場合、pr/pull-request-ID (例: pr/25) 形式を使用する必要があります。ブ ランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。[Source version] が 空白の場合は、デフォルトのブランチの HEAD コミット ID が使用されます。
- Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ 名、またはタグ名を入力します。ブランチ名を指定すると、ブランチの HEAD コミット ID が 使用されます。[Source version] が空白の場合は、デフォルトのブランチの HEAD コミット ID が使用されます。
- 10. (オプション) 5 分~2,160 分 (36 時間) の間のタイムアウトを指定します。この値は、ビルドが 停止するまでの AWS CodeBuild 試行時間を指定します。[時間] と [分] が空白のままの場合、プ ロジェクトで指定されたデフォルトのタイムアウト値が使用されます。
- 11. [Create trigger (トリガーの作成)] を選択します。

プログラムで AWS CodeBuild トリガーを作成する

CodeBuild は、ビルドトリガーに Amazon EventBridge ルールを使用します EventBridge API を使用 して、CodeBuild プロジェクトのビルドトリガーをプログラムで作成できます。詳細については、 「<u>Amazon EventBridge API リファレンス</u>」を参照してください。

AWS CodeBuild トリガーの編集

プロジェクトでトリガーを編集し、1 時間、1 日、または 1 週間に 1 回ビルドをスケジュールできま す。Amazon CloudWatch cron 式でカスタムルールを使用してトリガーを編集することもできます。 たとえば、cron 式を使用して、毎週特定の時間にビルドをスケジュールできます。トリガーの作成 方法については、「AWS CodeBuild トリガーの作成」を参照してください。

トピック

- AWS CodeBuild トリガーを編集する (コンソール)
- AWS CodeBuild トリガーをプログラムで編集する

AWS CodeBuild トリガーを編集する (コンソール)

次の手順で、 AWS Management Consoleを使用してトリガーを編集します。

トリガーを編集するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[ビルドプロジェクト]を選択します。
- 3. 変更するビルドプロジェクトのリンクを選択し、[ビルドのトリガー] タブを選択します。

Note

デフォルトでは、最新の 100 個のビルドプロジェクトが表示されます。さらに多くのビ ルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ 毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

- 4. 変更するトリガーの横にあるラジオボタンを選択して、[Edit (編集)]を選択します。
- 5. [Frequency] (頻度) ドロップダウンリストから、トリガーの頻度を選択します。CRON 式を使用 して頻度を作成する場合は、[Custom] (カスタム) を選択します。
- トリガーの頻度のパラメータを指定します。選択肢の最初の数文字をテキストボックスに入力すると、ドロップダウンメニュー項目がフィルタリングされます。

Note

開始時間と分はゼロベースです。開始分は 0 から 59 までの数値です。開始時は 0 か ら 23 までの数値です。たとえば、毎日午後 12:15 に開始する日次トリガーは、開始時 が 12、開始分が 15 になります。毎日深夜に開始される日次トリガーは、開始時がゼロ で、開始分がゼロです。毎日午後 11:59 に開始する毎日のトリガーは、開始時が 23、開 始分が 59 です。

Frequency	必須パラメータ	詳細
時間単価	開始時間 (分)	[開始時間 (分)] ドロップダウ ンメニューを使用します。
1日1回	開始時間 (分) 開始時間 (時)	[開始時間 (分)] ドロップダウ ンメニューを使用します。

Frequency	必須パラメータ	詳細 [開始時間 (時)] ドロップダウ ンメニューを使用します。
毎週	開始時間 (分) 開始時間 (時) 開始日	[開始時間 (分)] ドロップダウ ンメニューを使用します。 [開始時間 (時)] ドロップダウ ンメニューを使用します。 [開始時間 (日)] ドロップダウ ンメニューを使用します。
Custom	Cron 式	[Cron 式] に Cron 式を入力 します。Cron 式には、空 白で区切られた 6 つの必須 フィールドがあります。こ れらのフィールドでは、分、 時、日付、月、曜日、および 年の開始値を指定します。 範囲や追加の値などを指定 するには、ワイルドカード を使用します。例えば、cron 式 0 9 ? * MON-FRI * は 毎週平日午前 9 時にビルド をスケジュールします。詳 細については、「Amazon CloudWatch Events ユー ザーガイド」の「 <u>cron 式</u> 」 を参照してください。

7. [Enable this trigger (このトリガーの有効化)] を選択します。

Note

ソースバージョン、タイムアウト、および AWS CodeBuildで使用できないその他のオプ ションを編集するには、Amazon CloudWatch コンソール (<u>https://console.aws.amazon.com/</u> cloudwatch/) を使用できます。

AWS CodeBuild トリガーをプログラムで編集する

CodeBuild は、ビルドトリガーに Amazon EventBridge ルールを使用します EventBridge API を使用 して、CodeBuild プロジェクトのビルドトリガーをプログラムで編集できます。詳細については、 「Amazon EventBridge API リファレンス」を参照してください。

でビルドの詳細を表示する AWS CodeBuild

AWS CodeBuild コンソール、または AWS SDKs を使用して AWS CLI、CodeBuild によって管理さ れるビルドの詳細を表示できます。

トピック

- ・ ビルドの詳細の表示 (コンソール)
- <u>ビルドの</u>詳細の表示 (AWS CLI)
- ・ビルドの詳細を表示する (AWS SDKs)
- ビルドフェーズの移行

ビルドの詳細の表示 (コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. 次のいずれかを行います:
 - ナビゲーションペインで、[Build history] を選択します。ビルドのリストの [Build run (ビルドの実行)] 列で、ビルドのリンクを選択します。
 - ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。次に、ビルドのリストの [Build run (ビルドの実行)] 列で、ビルドのリンクを選択します。

Note

デフォルトでは、最新の 10 個のビルドまたはビルドプロジェクトのみ表示されます。 さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択し てから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の 値を選択するか、前後の矢印を使用します。

ビルドの詳細の表示 (AWS CLI)

AWS CLI で を使用する方法の詳細については AWS CodeBuild、「」を参照してください<u>コマンド</u> ラインリファレンス。

batch-get-builds コマンドを実行します。

aws codebuild batch-get-builds --ids ids

次のプレースホルダを置き換えます。

- ids: 必須の文字列。詳細を表示する1つ以上のビルドID。複数のビルドIDを指定するには、各ビルドIDをスペースで区切ります。最大100のビルドIDを指定できます。ビルドIDのリストを取得するには、次のトピックを参照してください。
 - ビルド ID の一覧表示 (AWS CLI)
 - ・ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

たとえば、次のコマンドを実行するとします。

aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-otherproject:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE

コマンドが正常に実行されると、「<u>要約されたビルド情報を表示するには</u>」に示されているものと 同様のデータが出力に表示されます。

ビルドの詳細を表示する (AWS SDKs)

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> ツールのリファレンス。 AWS SDKs

ビルドフェーズの移行

ビルドは段階的に AWS CodeBuild 続行されます。



▲ Important

UPLOAD_ARTIFACTS フェーズは、BUILD フェーズが失敗した場合でも必ず試行されます。

でビルド IDsのリストを表示する AWS CodeBuild

AWS CodeBuild コンソール AWS CLI、または AWS SDKs を使用して、CodeBuild によって管理されるビルドIDs のリストを表示できます。

トピック

- ビルド ID の一覧表示 (コンソール)
- ビルド ID の一覧表示 (AWS CLI)
- バッチビルド ID のリストを表示 (AWS CLI)
- ビルド IDs (AWS SDKsのリストを表示する)

ビルド ID の一覧表示 (コンソール)

1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https

2. ナビゲーションペインで、[Build history] を選択します。

Note

デフォルトでは、最新の 10 個のビルドのみ表示されます。さらに多くのビルドを表示 するには、歯車アイコンを選択し、[Builds per page (ページ毎ビルド数)] で別の値を選 択するか、前後の矢印を使用します。

ビルド ID の一覧表示 (AWS CLI)

CodeBuild AWS CLI で を使用する方法の詳細については、「」を参照してください<u>コマンドライン</u> <u>リファレンス</u>。

• list-builds コマンドを実行します。

aws codebuild list-builds --sort-order sort-order --next-token next-token

上記のコマンドで、次のプレースホルダを置き換えます。

- sort-order: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- next-token: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

aws codebuild list-builds --sort-order ASCENDING

次のような結果が出力に表示されることがあります。

```
{
    "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
    "ids": [
    "sedebuild dome president:015e755f bade (e7e 00f0 efe515YAMD)5"
```

"codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"

```
"codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
... The full list of build IDs has been omitted for brevity ...
"codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
token has been omitted for brevity...MzY20A==
```

次のような結果が出力に表示されることがあります。

```
{
   "ids": [
   "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
   "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
   ... The full list of build IDs has been omitted for brevity ...
   "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
]
}
```

バッチビルド ID のリストを表示 (AWS CLI)

CodeBuild AWS CLI で を使用する方法の詳細については、「」を参照してください<u>コマンドライン</u> リファレンス。

• list-build-batches コマンドを実行します。

aws codebuild list-build-batches --sort-order sort-order --next-token next-token

上記のコマンドで、次のプレースホルダを置き換えます。

- sort-order: バッチビルド ID の一覧表示方法を示すオプションの文字列です。有効な値は、ASCENDING および DESCENDING です。
- next-token: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出

しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

```
aws codebuild list-build-batches --sort-order ASCENDING
```

次のような結果が出力に表示されることがあります。



このコマンドをもう一度実行します。

aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The
full token has been omitted for brevity...MzY20A==

次のような結果が出力に表示されることがあります。

```
{
   "ids": [
   "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
   "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
   ... The full list of build IDs has been omitted for brevity ...
   "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

ビルド IDs (AWS SDKsのリストを表示する

SDK で CodeBuild を使用する方法の詳細については、「」を参照してください<u>AWS SDKsとツール</u> <u>のリファレンス</u>。 AWS SDKs

AWS CodeBuildでビルドプロジェクトのビルド ID を一覧表示する

AWS CodeBuild コンソール AWS CLI、または AWS SDKs を使用して、CodeBuild のビルドプロ ジェクトのビルド IDs のリストを表示できます。

トピック

- ビルドプロジェクトのビルド ID を一覧表示する (コンソール)
- ・ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)
- ・ビルドプロジェクトのバッチビルド ID のリストを表示 (AWS CLI)
- ・ ビルドプロジェクト (AWS SDKs) のビルド IDs のリストを表示する

ビルドプロジェクトのビルド ID を一覧表示する (コンソール)

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [プロジェクト] 列で、ビルドプロジェクトを選択します。

デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さ らに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択す るか、前後の矢印を使用します。

ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

AWS CLI で を使用する方法の詳細については AWS CodeBuild、「」を参照してください<u>コマンド</u> <u>ラインリファレンス</u>。

次のように list-builds-for-project コマンドを実行します。

Note

aws codebuild list-builds-for-project --project-name project-name --sort-order sortorder --next-token next-token

上記のコマンドで、次のプレースホルダを置き換えます。

- project-name: ビルド ID を一覧表示するビルドプロジェクトの名前を示すのに必要な文字列。
 ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示 (AWS CLI)」
 を参照してください。
- sort-order: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- next-token: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、次に続くトークンが返されるごとにこのコマンドを実行し続けます。

たとえば、このコマンドを次のように実行するとします。

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-
order ASCENDING
```

次のような結果が出力に表示されます。

```
{
    "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
    "ids": [
        "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
        "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
        ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
    ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --
sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
brevity...MzY20A==
```

次のような結果が出力に表示されます。

{
"ids": [
"codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
"codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
The full list of build IDs has been omitted for brevity
"codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
]
}

ビルドプロジェクトのバッチビルド ID のリストを表示 (AWS CLI)

AWS CLI で を使用する方法の詳細については AWS CodeBuild、「」を参照してください<u>コマンド</u> ラインリファレンス。

次のように list-build-batches-for-project コマンドを実行します。

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- project-name: ビルド ID を一覧表示するビルドプロジェクトの名前を示すのに必要な文字列。
 ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示 (AWS CLI)」
 を参照してください。
- sort-order: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- next-token: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、次に続くトークンが返されるごとにこのコマンドを実行し続けます。

たとえば、このコマンドを次のように実行するとします。

aws codebuild list-build-batches-for-project --project-name codebuild-demo-project -sort-order ASCENDING
次のような結果が出力に表示されます。

```
{
   "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
   "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
   "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project
    --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
    brevity...MzY20A==
```

次のような結果が出力に表示されます。

```
{
   "ids": [
   "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
   "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
   ... The full list of build IDs has been omitted for brevity ...
   "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

ビルドプロジェクト (AWS SDKs) のビルド IDs のリストを表示する

SDK AWS CodeBuild で を使用する方法の詳細については、「」を参照してください<u>AWS SDKsと</u> ツールのリファレンス。 AWS SDKs

でのテストレポート AWS CodeBuild

ビルド時に実行したテストの詳細を含むレポートを CodeBuild で作成できます。単体テスト、設定 テスト、機能テストなどのテストを作成できます。

以下のテストレポートファイル形式がサポートされています。

- Cucumber JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

Note

cucumber-jsのサポートされている最新バージョンは7.3.2です。

Surefire JUnit plugin、TestNG、Cucumber などのいずれかの形式でレポートファイルを作成できる 任意のテストフレームワークを使用して、テストケースを作成します。

テストレポートを作成するには、ビルドプロジェクトの buildspec ファイルにテストケースに関する 情報を含むレポートグループ名を追加します。ビルドプロジェクトを実行すると、テストケースが実 行され、テストレポートが作成されます。テストケースを実行するたびに、レポートグループに新し いテストレポートが作成されます。テストを実行する前にレポートグループを作成する必要はありま せん。レポートグループ名を指定すると、CodeBuildはレポートの実行時にレポートグループを作成 します。既に存在するレポートグループを使用する場合は、buildspec ファイルでその ARN を指定 します。

テストレポートを使用すると、ビルドの実行中に問題をトラブルシューティングできます。ビルドプ ロジェクトの複数のビルドから多数のテストレポートがある場合、テストレポートを使用してトレン ドやテストと失敗率を表示し、ビルドを最適化できます。

レポートは、作成から 30 日後に有効期限が切れます。期限切れのテストレポートは表示できません。30 日以上テストレポートを保持する場合は、テスト結果の生データファイルをAmazon S3 バ

ケットにエクスポートできます。エクスポートされたテストファイルは期限切れになりません。S3 バケットに関する情報は、レポートグループを作成するときに指定します。

Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするア クセス許可に使用されます。

トピック

- テストレポートの作成
- コードカバレッジレポートを作成
- CodeBuild でレポートを自動的に検出
- レポートグループ
- テストフレームワーク
- テストレポートの表示
- テストレポートのアクセス許可
- テストレポートのステータス

テストレポートの作成

テストレポートを作成するには、buildspec ファイルに 1 つから 5 つのレポートグループで設定され ているビルドプロジェクトを実行します。テストレポートは、実行中に作成されます。レポートグ ループに対して指定されたテストケースの結果が含まれます。同じ buildspec ファイルを使用する後 続のビルドごとに、新しいテストレポートが生成されます。

テストレポートを作成するには

- ビルドプロジェクトを作成します。詳細については、<u>でビルドプロジェクトを作成する AWS</u> CodeBuild を参照してください。
- 2. テストレポート情報を使用してプロジェクトの buildspec ファイルを設定します。
 - a. reports:セクションを追加し、既存のレポートグループの ARN、またはレポートグルー プの名前を指定します。

ARN を指定すると、CodeBuild はそのレポートグループを使用します。

名前を指定した場合、CodeBuild は、プロジェクト名と *<project-name>-<reportgroup-name>*の形式で指定した名前を使用してレポートグループを作成します。名前付き レポートグループが既に存在する場合、CodeBuild はそのレポートグループを使用します。

- b. レポートグループで、テスト結果が含まれるファイルの場所を指定します。複数のレポート グループを使用する場合は、各レポートグループに対してテスト結果ファイルの場所を指定 します。ビルドプロジェクトを実行するたびに、新しいテストレポートが作成されます。詳 細については、「テストファイルの指定」を参照してください。
- c. build または post_build シーケンスの commands セクションで、レポートグループに 対して指定したテストケースを実行するコマンドを指定します。詳細については、「<u>テス</u> トコマンドの指定」を参照してください。

buildspec reports セクションの例を以下に示します。

```
reports:
    php-reports:
        files:
            - "reports/php/*.xml"
        file-format: "JUNITXML"
    nunit-reports:
        files:
            - "reports/nunit/*.xml"
        file-format: "NUNITXML"
```

- ビルドプロジェクトのビルドを実行します。詳細については、「<u>AWS CodeBuild ビルドを手動</u> で実行する」を参照してください。
- ビルドが完了したら、プロジェクトページの [Build history (ビルド 履歴)] から新しいビルド実行 を選択します。[Reports (レポート)] を選択して、テストレポートを表示します。詳細について は、「ビルドのテストレポートの表示」を参照してください。

コードカバレッジレポートを作成

CodeBuild を使用して、テストのコードカバレッジレポートを生成できます。次のコードカバレッジ レポートが用意されています。 ラインカバレッジ

ラインカバレッジは、テストがカバーするステートメントの数を測定します。ステートメント は、コメントや条件を含まない、単一の命令です。

line coverage = (total lines covered)/(total number of lines) ブランチカバレッジ

ブランチカバレッジは、コントロール構造のすべてのブランチ内のテスト可能なブランチの数を 測定します (「if」または「case」ステートメントなど)。

branch coverage = (total branches covered)/(total number of branches)

以下のコードカバレッジレポートファイル形式がサポートされています。

- JaCoCo XML
- SimpleCov JSON¹
- クローバー XML
- Cobertura XML
- LCOV INFO

¹ CodeBuild は、simplecov-json ではなく、simplecov によって生成された JSON コードカバレッジレポートを受け入れます。

コードカバレッジレポートの作成

コードカバレッジレポートを作成するには、buildspec ファイルの最低 1 つのコードカバレッジグ ループで設定されているビルドプロジェクトを実行します。CodeBuild は、コードカバレッジの結果 を解釈し、実行のコードカバレッジレポートを提供します。同じ buildspec ファイルを使用する後続 のビルドごとに、新しいテストレポートが生成されます。

テストレポートを作成するには

- ビルドプロジェクトを作成します。詳細については、<u>でビルドプロジェクトを作成する AWS</u> CodeBuild を参照してください。
- 2. テストレポート情報を使用してプロジェクトの buildspec ファイルを設定します。
 - a. reports: セクションを追加し、レポートグループの名前を指定します。名前を指定した場合、CodeBuild は、プロジェクト名と指定した project-name report-group-

name-in-buildspec 形式でレポートグループを作成します。使用するレポートグルー プがすでにある場合は、その ARN を指定します。ARN の代わりに名前を使用する場 合、CodeBuild は新しいレポートグループを作成します。詳細については、「<u>Reports</u> syntax in the buildspec file」を参照してください。

b. レポートグループで、コードカバレッジの結果を保存するファイルの場所を指定します。複数のレポートグループを使用する場合は、各レポートグループに対して結果ファイルの場所を指定します。ビルドプロジェクトを実行するたびに、新しいコードカバレッジが作成されます。詳細については、「テストファイルの指定」を参照してください。

これは「test-results/jacoco-coverage-report.xml」にある JaCoCo XML 結果ファ イルのコードカバレッジレポートを生成する例です。

```
reports:
  jacoco-report:
    files:
        - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

- c. 「build」または「post_build」シーケンスの「commands」セクションで、コードカバレッジ分析を実行するコマンドを指定します。詳細については、「<u>テストコマンドの指定</u>」を参照してください。
- ビルドプロジェクトのビルドを実行します。詳細については、「<u>AWS CodeBuild ビルドを手動</u> で実行する」を参照してください。
- ビルドが完了したら、プロジェクトページの [Build history (ビルド 履歴)] から新しいビルド実行 を選択します。レポートを選択して、コードカバレッジレポートを表示します。詳細について は、「ビルドのテストレポートの表示」を参照してください。

CodeBuild でレポートを自動的に検出

自動検出を使用すると、CodeBuild はビルドフェーズが完了した後にすべてのビルドファイルを検索 し、サポートされているレポートファイルタイプを検索して、新しいテストおよびコードカバレッジ レポートグループとレポートを自動的に作成します。CodeBuild は、検出されたレポートタイプに対 して、新しいレポートグループを次のパターンで作成します。

<project-name>-<report-file-format>-AutoDiscovered

(i) Note

検出されたレポートファイルが同じ形式タイプである場合、それらは同じレポートグループ またはレポートに配置されます。

レポートの自動検出は、プロジェクト環境変数によって設定されます。

CODEBUILD_CONFIG_AUTO_DISCOVER

この変数は、ビルド中にレポートの自動検出を無効にするかどうかを決定します。デフォルト では、レポートの自動検出はすべてのビルドで有効になっています。この機能を無効にするに は、CODEBUILD_CONFIG_AUTO_DISCOVER を false に設定します。

CODEBUILD_CONFIG_AUTO_DISCOVER_DIR

(オプション) この変数は、CodeBuild が潜在的なレポートファイルを検索する場所を決定しま す。CodeBuild はデフォルトで **/* を検索することに注意してください。

これらの環境変数は、ビルドフェーズ中に変更できます。例えば、main git ブランチのビルドのレ ポート自動検出のみを有効にする場合は、ビルドプロセス中に git ブランチをチェックし、ビルドが main ブランチにない場合は CODEBUILD_CONFIG_AUTO_DISCOVER を false に設定できます。レ ポートの自動検出は、コンソールまたはプロジェクト環境変数を使用して無効にできます。

トピック

- コンソールを使用してレポートの自動検出を設定
- プロジェクト環境変数を使用してレポートの自動検出を設定

コンソールを使用してレポートの自動検出を設定

コンソールを使用してレポートの自動検出を設定するには、次の手順に従います。

コンソールを使用してレポートの自動検出を設定するには

- ビルドプロジェクトを作成するか、編集するビルドプロジェクトを選択します。詳細について は、「<u>でビルドプロジェクトを作成する AWS CodeBuild</u>」または「<u>でビルドプロジェクト設定</u> を変更する AWS CodeBuild」を参照してください。
- 2. [環境] で、[追加設定] を選択します。

- 3. レポート自動検出を無効にするには、[レポート自動検出] で [レポート自動検出を無効化] を選択 します。
- (オプション) [自動検出ディレクトリ オプション] で、CodeBuild のディレクトリパターンを入 力して、サポートされているレポート形式のファイルを検索します。CodeBuild はデフォルトで **/* を検索することに注意してください。

プロジェクト環境変数を使用してレポートの自動検出を設定

プロジェクト環境変数を使用してレポートの自動検出を設定するには、次の手順に従います。

プロジェクト環境変数を使用してレポートの自動検出を設定するには

- ビルドプロジェクトを作成するか、編集するビルドプロジェクトを選択します。詳細について は、「<u>でビルドプロジェクトを作成する AWS CodeBuild</u>」または「<u>でビルドプロジェクト設定</u> を変更する AWS CodeBuild」を参照してください。
- 2. [環境変数] で、以下の操作を実行します。
 - a. レポートの自動検出を無効にするには、[名前] に CODEBUILD_CONFIG_AUTO_DISCOVER
 を入力し、[値] に false を入力します。これにより、レポートの自動検出が無効になります。
 - b. (オプション) [名前] に CODEBUILD_CONFIG_AUTO_DISCOVER_DIR を入力し、[値] には
 CodeBuild がサポートされているレポート形式のファイルを検索するディレクトリを入力し
 ます。例えば、output/*xml は output ディレクトリ内の.xml ファイルを検索します。

レポートグループ

レポートグループにはテストレポートが含まれており、共有設定を指定します。buildspec ファイル を使用して、実行するテストケースと、ビルド時に実行するコマンドを指定します。ビルドプロジェ クトで設定された各レポートグループに対して、ビルドプロジェクトの実行によってテストレポート が作成されます。レポートグループで設定されたビルドプロジェクトを複数実行すると、そのレポー トグループに複数のテストレポートが作成され、そのレポートグループに指定された同じテストケー スの結果がそれぞれ作成されます。

テストケースは、ビルドプロジェクトの buildspec ファイル内のレポートグループに対して指定され ています。1 つのビルドプロジェクトで最大 5 つのレポートグループを指定できます。ビルドを実行 すると、すべてのテストケースが実行されます。新しいテストレポートは、レポートグループに指定 された各テストケースの結果で作成されます。新しいビルドを実行するたびに、テストケースが実行 され、新しいテスト結果を使用して新しいテストレポートが作成されます。

レポートグループは、複数のビルドプロジェクトで使用できます。1つのレポートグループで作成さ れたすべてのテストレポートは、異なるビルドプロジェクトを使用してテストレポートを作成した場 合でも、エクスポートオプションやアクセス権限など、同じ設定を共有します。複数のビルドプロ ジェクトで1つのレポートグループを使用して作成されたテストレポートには、異なるテストケー スセット (ビルドプロジェクトごとに1セットのテストケース)の実行結果を含めることができま す。これは、各プロジェクトの buildspec ファイルで、レポートグループに異なるテストケースファ イルを指定できるためです。また、buildspec ファイルを編集して、ビルドプロジェクトのレポート グループのテストケースファイルを変更することもできます。その後のビルド実行では、更新された buildspec のテストケースファイルの結果を含む新しいテストレポートが作成されます。

トピック

- Create a report group
- Report group naming
- レポートグループを共有
- テストファイルの指定
- テストコマンドの指定
- でレポートグループにタグを付ける AWS CodeBuild
- レポートグループの更新

Create a report group

CodeBuild コンソール、 AWS CLI、または buildspec ファイルを使用して、レポートグループを作 成できます。IAM ロールには、レポートグループを作成するために必要なアクセス権限が必要で す。詳細については、「テストレポートのアクセス許可」を参照してください。

トピック

- ・ レポートグループの作成 (buildspec)
- <u>Create a report group (console)</u>
- ・ <u>レポートグループの作成 (CLI)</u>
- レポートグループの作成 (AWS CloudFormation)

レポートグループの作成 (buildspec)

buildspec を使用して作成されたレポートグループは、生のテスト結果ファイルをエクスポートしま せん。レポートグループを表示し、エクスポート設定を指定できます。詳細については、「<u>レポート</u> グループの更新」を参照してください。

buildspec ファイルを使用してレポートグループを作成するには

- AWS アカウントのレポートグループに関連付けられていないレポートグループ名を選択します。
- buildspec ファイルの reports セクションをこの名前で設定します。この例では、レポートグ ループ名は new-report-group で、ユーステストケースは JUnit フレームワークを使用して作 成されます。

```
reports:
    new-report-group: #surefire junit reports
    files:
        - '**/*'
    base-directory: 'surefire/target/surefire-reports'
```

レポートグループ名は、buildspec で環境変数を使用して指定することもできます。

```
version: 0.2
env:
variables:
    REPORT_GROUP_NAME: "new-report-group"
phases:
    build:
    commands:
        - ...
...
reports:
    $REPORT_GROUP_NAME:
    files:
        - '**/*'
    base-directory: 'surefire/target/surefire-reports'
```

詳細については、<u>テストファイルの指定</u>および<u>Reports syntax in the buildspec file</u>を参照してく ださい。

Create a report group

- commands セクションで、テストを実行するコマンドを指定します。詳細については、「<u>テス</u> トコマンドの指定」を参照してください。
- ビルドを実行します。ビルドが完了すると、形式 project-name-report-group-name を 使用する名前で新しいレポートグループが作成されます。詳細については、「<u>Report group</u> naming」を参照してください。

Create a report group (console)

次の手順で、 AWS Management Consoleを使用してレポートグループを作成します。

レポートグループの作成

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https://https
- 2. ナビゲーションペインで、[Report groups (レポートグループ)]を選択します。
- 3. [Create report group (レポートグループを作成)] を選択します。
- 4. [Report group name (レポートグループ名)] に、レポートグループの名前を入力します。
- 5. (オプション) タグには、サポート AWS サービスで使用するタグの名前と値を入力します。 [Add row] を使用して、タグを追加します。最大 50 個のタグを追加できます。
- 6. テストレポート結果の raw データを Amazon S3 バケットにアップロードする場合は、次のよう にします。
 - a. [Amazon S3 にエクスポート] を選択します。
 - b. [S3 bucket name (S3 バケット名)] に、S3 バケットの名前を入力します。
 - c. (オプション)S3 バケット所有者で S3 バケットを所有するアカウントの AWS アカウント識別子を入力します。これにより、レポートデータを、ビルドを実行しているアカウント以外のアカウントが所有する Amazon S3 バケットにエクスポートできます。
 - d. [Path prefix (パスプレフィックス)] に、テスト結果をアップロードする S3 バケットのパス を入力します。
 - e. 生のテスト結果データファイルを圧縮するには、[Compress test result data in a zip file (テ スト結果データを圧縮する)] を選択します。
 - f. [Additional configuration (追加の設定)] を展開して、暗号化オプションを表示します。次の いずれかを選択します。

- Amazon S3 AWS マネージドキーのを使用するデフォルトのAWS マネージドキー。詳細については、AWS Key Management Service ユーザーガイドの「<u>カスタマーマネージ</u>ドCMKs」を参照してください。これはデフォルトの暗号化オプションです。
- カスタムキーを選択して、ユーザーが作成して設定するカスタマー管理のキーを使用 します。AWS KMS 暗号化キーの場合は、暗号化キーの ARN を入力します。形式は arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id> です。詳細に ついては、AWS Key Management Service ユーザーガイドの「KMS キーの作成」を参照 してください。
- ・暗号化を無効にするには、アーティファクト暗号化を無効にします。テスト結果を共有したり、静的ウェブサイトに公開したりする場合は、このオプションを選択します。(動的ウェブサイトでは、テスト結果を復号化するコードを実行できます)。

保管時の暗号化の詳細については、「データの暗号化」を参照してください。

Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードす るアクセス許可に使用されます。

7. [Create report group (レポートグループを作成)] を選択します。

レポートグループの作成 (CLI)

次の手順で、 AWS CLIを使用してレポートグループを作成します。

レポートグループの作成

- 1. CreateReportGroup.json という名前のファイルを作成します。
- 2. 要件に応じて、以下の JSON コードスニペットのいずれかを CreateReportGroup.json にコ ピーします。
 - 次の JSON を使用して、テストレポートグループが生のテスト結果ファイルを Amazon S3 バケットにエクスポートするように指定します。

{
 "name": "<report-name>",

```
"type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "bucketOwner": "<bucket-owner>",
      "path": "<path>",
      "packaging": "NONE | ZIP",
      "encryptionDisabled": "false",
      "encryptionKey": "<your-key>"
    },
    "tags": [
      {
        "key": "tag-key",
        "value": "tag-value"
      }
    ]
 }
}
```

- <bucket-name> を Amazon S3 バケット名に、<path> をファイルをエクスポートするバ ケット内のパスに置き換えます。
- エクスポートされたファイルを packaging に圧縮する場合は、ZIP を指定します。それ 以外の場合は、NONE を指定します。
- bucket0wner はオプションで、Amazon S3 バケットがビルドを実行しているアカウント 以外のアカウントによって所有されている場合にのみ必要です。
- エクスポートされたファイルを暗号化するかどうかを指定するために encryptionDisabledを使用します。エクスポートしたファイルを暗号化する場合は、 カスタマー管理のキーを入力します。詳細については、「レポートグループの更新」を参照 してください。
- 次の JSON を使用して、テストレポートで生のテストファイルをエクスポートしないように 指定します。

```
{
    "name": "<report-name>",
    "type": "TEST",
    "exportConfig": {
        "exportConfigType": "NO_EXPORT"
    }
```

}

Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードす るアクセス許可に使用されます。

3. 次のコマンドを実行します。

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

レポートグループの作成 (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してレポートグループを作成するには、次の手順に従います。

AWS CloudFormation テンプレートを使用してレポートグループを作成するには

AWS CloudFormation テンプレートファイルを使用して、レポートグループを作成およびプロビジョ ニングできます。詳細については、「<u>AWS CloudFormation ユーザーガイド</u>」を参照してください。

次の AWS CloudFormation YAML テンプレートは、未加工のテスト結果ファイルをエクスポートしないレポートグループを作成します。

```
Resources:
CodeBuildReportGroup:
Type: AWS::CodeBuild::ReportGroup
Properties:
Name: my-report-group-name
Type: TEST
ExportConfig:
ExportConfig:
ExportConfigType: NO_EXPORT
```

次の YAML テンプレートは、raw AWS CloudFormation テスト結果ファイルを Amazon S3 バケット にエクスポートするレポートグループを作成します。

Resources: CodeBuildReportGroup:

Create a report group

```
Type: AWS::CodeBuild::ReportGroup

Properties:

Name: my-report-group-name

Type: TEST

ExportConfig:

ExportConfigType: S3

S3Destination:

Bucket: amzn-s3-demo-bucket

Path: path-to-folder-for-exported-files

Packaging: ZIP

EncryptionKey: my-KMS-encryption-key

EncryptionDisabled: false
```

Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするア クセス許可に使用されます。

Report group naming

AWS CLI または AWS CodeBuild コンソールを使用してレポートグループを作成する場合は、レ ポートグループの名前を指定します。buildspec を使用して新しいレポートグループを作成する場 合、project-name-report-group-name-specified-in-buildspec 形式を使用して名前が 付けられます。そのビルドプロジェクトのビルドを実行することによって作成されたすべてのレポー トは、新しい名前を持つ新しいレポートグループに属します。

CodeBuild が新しいレポートグループを作成しない場合は、ビルドプロジェクトの buildspec ファイ ルでレポートグループの ARN を指定します。レポートグループの ARN は、複数のビルドプロジェ クトで指定できます。各ビルドプロジェクトが実行されると、レポートグループには各ビルドプロ ジェクトによって作成されたテストレポートが含まれます。

たとえば、my-report-group という名前のレポートグループを1つ作成し、その名前をmyproject-1とmy-project-2という名前の2つの異なるビルドプロジェクトで使用し、両方のプ ロジェクトのビルドを作成した場合、2つの新しいレポートグループが作成されます。結果は、次の 名前を持つ3つのレポートグループになります。

- my-report-group: テストレポートはありません。
- my-project-1-my-report-group: という名前のビルドプロジェクトによって実行されたテストの結果を含むレポートが含まれます。my-project-1

my-project-2-my-report-group: という名前のビルドプロジェクトによって実行されたテストの結果を含むレポートが含まれます。my-project-2

両方のプロジェクトで my-report-group という名前のレポートグループの ARN を使用し、各プ ロジェクトのビルドを実行しても、1 つのレポートグループ (my-report-group) は残ります。そ のレポートグループには、両方のビルドプロジェクトによって実行されるテストの結果を含むテスト レポートが含まれます。

AWS アカウントのレポートグループに属していないレポートグループ名を選択し、buildspec ファ イル内のレポートグループにその名前を使用し、ビルドプロジェクトのビルドを実行すると、新しい レポートグループが作成されます。新しいレポートグループの名前の形式は project-name-newgroup-name です。例えば、AWS アカウントに という名前のレポートグループがなくnewreport-group、 という名前のビルドプロジェクトで指定した場合test-project、ビルド実 行によって という名前の新しいレポートグループが作成されますtest-project-new-reportgroup。

レポートグループを共有

レポートグループ共有を使用すると、複数の AWS アカウントまたはユーザーがレポートグループ、 期限切れでないレポート、およびレポートのテスト結果を表示できます。このモデルでは、レポート グループを所有するアカウント (所有者) は、レポートグループを他のアカウント (コンシューマー) と共有します。コンシューマは、レポートグループを編集できません。レポートは、作成から 30 日 後に期限切れになります。

トピック

- レポートグループを共有
- 関連サービス
- 共有されているレポートグループにアクセス
- 共有レポートグループを共有解除
- 共有レポートグループを識別
- 共有レポートグループのアクセス許可

レポートグループを共有

レポートグループを共有すると、コンシューマーには、レポートグループとそのレポートに対する 読み取り専用アクセス権が付与されます。コンシューマーは を使用して AWS CLI 、レポートグルー プ、そのレポート、および各レポートのテストケース結果を表示できます。コンシューマは次を行う ことはできません。

- CodeBuild コンソールでの共有レポートグループまたはそのレポートの表示。
- 共有レポートグループの編集。
- プロジェクト内の共有レポートグループの ARN を使用してレポートを実行。共有レポートグループを指定するプロジェクトのビルドが失敗します。

CodeBuild コンソールを使用して、既存のリソース共有にレポートグループを追加できます。新しい リソース共有にレポートグループを追加する場合は、まず<u>AWS RAM コンソール</u> でレポートグルー プを作成する必要があります。

レポートグループを組織単位または組織全体と共有するには、 AWS Organizationsとの共有を有効に する必要があります。詳細については、AWS RAM ユーザーガイドの「<u>AWS Organizationsで共有を</u> 有効化する」を参照してください。

CodeBuild コンソール、 AWS RAM コンソール、または AWS CLI を使用して、所有しているレポートグループを共有できます。

前提条件

レポートグループを共有するには、 AWS アカウントがそのグループを所有している必要がありま す。自分と共有されているレポートグループは共有できません。

所有するレポートグループを共有するには (CodeBuild コンソール)

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Report groups (レポートグループ)] を選択します。
- 3. 共有するプロジェクトを選択し、[Share (共有)] を選択します。詳細については、AWS RAM ユーザーガイドの「リソースの共有の作成」を参照してください。

所有しているレポートグループを共有するには (AWS RAM コンソール)

「AWS RAM ユーザーガイド」の「リソース共有の作成」を参照してください。

所有しているレポートグループを共有するには (AWS RAM コマンド)

create-resource-share コマンドを使用します。

所有するレポートグループを共有するには (CodeBuild コマンド)

put-resource-policy コマンドを使用します:

1. policy.json という名前のファイルを作成し、その中に次をコピーします。

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Principal":{
        "AWS":"consumer-aws-account-id-or-user"
      },
      "Action":[
        "codebuild:BatchGetReportGroups",
        "codebuild:BatchGetReports",
        "codebuild:ListReportsForReportGroup",
        "codebuild:DescribeTestCases"],
        "Resource":"arn-of-report-group-to-share"
    }]
}
```

レポートグループ ARN とそれを共有する識別子で policy.json を更新します。次の例では、ARN を持つレポートグループへの読み取り専用アクセスarn:aws:codebuild:us-west-2:123456789012:report-group/my-report-groupを Alice と 123456789012「」で識別される AWS アカウントのルートユーザーに付与します。

```
"Resource":"arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-
group"
    }]
}
```

3. 以下のコマンドを実行してください。

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://
policy.json
```

関連サービス

レポートグループ共有は AWS Resource Access Manager 、 (AWS RAM) と統合されます。これ は、 AWS リソースを任意の AWS アカウントまたは を通じて共有できるようにするサービスです AWS Organizations。では AWS RAM、リソースと共有するコンシューマーを指定するリソース共 有を作成して、所有しているリソースを共有します。コンシューマーは、個々の AWS アカウント、 の組織単位 AWS Organizations、または の組織全体にすることができます AWS Organizations。

詳細については、AWS RAM ユーザーガイドをご参照ください。

共有されているレポートグループにアクセス

共有レポートグループにアクセスするには、コンシューマーの IAM ロールに BatchGetReportGroups アクセス許可が必要です。次のポリシーを IAM ロールにアタッチするこ とができます。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetReportGroups"
    ]
}
```

詳細については、「<u>でのアイデンティティベースのポリシーの使用 AWS CodeBuild</u>」を参照してく ださい。

レポートグループを共有

共有レポートグループを共有解除

レポートとテストケースの結果が含まれる共有が解除されたレポートグループは、その所有者だけが アクセスできます。レポートグループの共有を解除すると、以前に共有した AWS アカウントまたは ユーザーは、レポートグループ、そのレポート、またはレポート内のテストケースの結果にアクセス できなくなります。

所有するレポートグループを共有または共有を解除するには、リソース共有から削除する必要があり ます。これを行う AWS CLI には、 AWS RAM コンソールまたは を使用できます。

所有している共有レポートグループの共有を解除するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の更新」を参照してください。

所有している共有レポートグループの共有を解除するには (AWS RAM コマンド)

disassociate-resource-share コマンドを使用します。

CodeBuild コマンドを所有しているレポートグループの共有を解除するには)

delete-resource-policy コマンドを実行し、共有を解除したいレポートグループの ARN を指定する:

aws codebuild delete-resource-policy --resource-arn report-group-arn

共有レポートグループを識別

所有者とコンシューマーは、 AWS CLI を使用して共有レポートグループを識別できます。

共有レポートグループとそのレポートを識別して情報を取得するには、次のコマンドを使用します。

 自分と共有されているレポートグループの ARN を表示するには、<u>list-shared-report-</u> groupsを実行します。

aws codebuild list-shared-report-groups

 レポートグループ内のレポートの ARN を表示するには、レポートグループ ARN を使い <u>list-</u> <u>reports-for-report-group</u> を実行します。

aws codebuild list-reports-for-report-group --report-group-arn report-group-arn

レポートグループを共有

レポート内のテストケースに関する情報を表示するには、レポート ARN を使い、<u>describe-</u> test-cases を実行します。

aws codebuild describe-test-cases --report-arn report-arn

出力は次のようになります。

```
{
    "testCases": [
        {
            "status": "FAILED",
            "name": "Test case 1",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        },
        {
            "status": "SUCCEEDED",
            "name": "Test case 2",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        }
    ]
}
```

共有レポートグループのアクセス許可

所有者のアクセス許可

レポートグループの所有者は、レポートグループを編集し、プロジェクトで指定してレポートを実行 できます。 コンシューマーのアクセス許可

レポートグループのコンシューマーは、レポートグループ、そのレポート、およびレポートのテス トケース結果を表示できます。コンシューマーは、レポートグループまたはそのレポートを編集した り、レポートを作成したりすることはできません。

テストファイルの指定

ビルドプロジェクトの buildspec ファイルの reports セクションで、各レポートグループのテスト 結果ファイルとその場所を指定します。詳細については、「<u>Reports syntax in the buildspec file</u>」を 参照してください。

以下は、ビルドプロジェクトの2つのレポートグループを指定するサンプル reports セクション です。1 つは ARN で指定され、もう 1 つは名前で指定されます。files セクションでは、テスト ケースの結果を含むファイルを指定します。オプション base-directory セクションでは、テス トケースファイルがあるディレクトリを指定します。オプションの discard-paths セクションで は、Amazon S3 バケットにアップロードされたテスト結果ファイルへのパスを破棄するかどうかを 指定します。

```
reports:
```

arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
#surefire junit reports

```
files:
    - '**/*'
base-directory: 'surefire/target/surefire-reports'
discard-paths: false
```

sampleReportGroup: #Cucumber reports from json plugin
files:
 - 'cucumber-json/target/cucumber-json-report.json'
file-format: CUCUMBERJSON #Type of the report, defaults to JUNITXML

テストコマンドの指定

テストケースを実行するコマンドは、buildspec ファイルの commands セクションで指定します。 これらのコマンドは、buildspec ファイルの reports セクションでレポートグループに指定され たテストケースを実行します。次に、テストファイルでテストを実行するコマンドを含むサンプル commands セクションを示します。

commands:

- echo Running tests for surefire junit
- mvn test -f surefire/pom.xml -fn
- echo
- echo Running tests for cucumber with json plugin

- mvn test -Dcucumber.options="--plugin json:target/cucumber-json-report.json" -f
cucumber-json/pom.xml -fn

詳細については、「buildspec の構文」を参照してください。

でレポートグループにタグを付ける AWS CodeBuild

タグは、 AWS リソース AWS に割り当てるカスタム属性ラベルです。各 AWS タグには 2 つの部分 があります。

- タグキー (CostCenter、Environment、Project、Secret など)。タグキーでは、大文字と小 文字が区別されます。
- タグ値と呼ばれるオプションのフィールド (111122223333、Production、チーム名など)。タ グ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値では 大文字と小文字が区別されます。

これらは共にキーと値のペアと呼ばれます。レポートグループに付けることができるタグの最大数、 およびタグのキーと値の制限については、「[タグ]」を参照してください。

タグは、AWS リソースの識別と整理に役立ちます。多くの AWS サービスはタグ付けをサポート しているため、異なる サービスのリソースに同じタグを割り当てることで、リソースが関連して いることを示すことができます。たとえば、Amazon S3 バケットに割り当てたものと同じタグを CodeBuild レポートグループに割り当てることができます。タグの使用の詳細については、「<u>タグ付</u> けのベストプラクティス」ホワイトペーパーを参照してください。

CodeBuild では、主なリソースはレポートグループとプロジェクトです。CodeBuild コンソール、 AWS CLI、CodeBuild APIs、または AWS SDKs を使用して、レポートグループのタグを追加、 管理、削除できます。タグを使用して、レポートグループを識別、組織付け、追跡するだけでな く、IAM ポリシーでタグを使用して、レポートグループを表示および操作できるユーザーをコン トロールすることもできます。タグベースのアクセスポリシーの例については、「<u>タグを使用した</u> AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

トピック

- レポートグループにタグを追加
- レポートグループのタグを表示する

- レポートグループのタグを編集する
- レポートグループからタグを削除

レポートグループにタグを追加

レポートグループにタグを追加すると、AWS リソースを識別して整理し、リソースへのアクセスを 管理するのに役立ちます。まず、レポートグループに1つ以上のタグ (キーと値のペア) を追加しま す。レポートグループに付けることができるタグの数には制限があります。キーフィールドおよび 値フィールドに使用できる文字には制限があります。詳細については、「[タグ]」を参照してくださ い。タグを追加した後、IAM ポリシーを作成して、それらのタグに基づいてレポートグループへの アクセスを管理できます。CodeBuild コンソールまたは を使用して AWS CLI、レポートグループに タグを追加できます。

▲ Important

レポートグループにタグを追加すると、そのレポートグループへのアクセスに影響を与え る可能性があります。レポートグループにタグを追加する前に、タグを使用してレポートグ ループなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確 認してください。タグベースのアクセスポリシーの例については、「<u>タグを使用した AWS</u> <u>CodeBuild リソースへのアクセスのコントロール</u>」を参照してください。

レポートグループの作成時にタグを追加する方法の詳細については、「<u>Create a report group</u> (console)」を参照してください。

トピック

- レポートグループにタグを追加する (コンソール)
- ・レポートグループにタグを追加する (AWS CLI)

レポートグループにタグを追加する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに1つ以上のタグを追加できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Report groups (レポートグループ)] で、タグを追加するレポートグループの名前を選択します。
- 3. ナビゲーションペインで[設定]を選択します。

- 4. レポートグループにいずれのタグも追加されていない場合は、[Add tag (タグの追加)] を選択し ます。[Edit (編集)] を選択してから、[Add tag (タグの追加)] を選択することもできます。
- 5. [Key] に、タグの名前を入力します。[値] では、任意でタグに値を追加できます。
- 6. (オプション)別のタグを追加するには、[Add tag]を再度選択します。
- 7. タグの追加を完了したら、[Submit] を選択します。

レポートグループにタグを追加する (AWS CLI)

作成時にレポートグループにタグを追加するには、「<u>レポートグループの作成 (CLI)</u>」を参照してく ださい。CreateReportGroup.json で、タグを追加します。

既存のレポートグループにタグを追加するには、「<u>レポートグループの更新 (CLI)</u>」を参照 し、UpdateReportGroupInput.json でタグを追加します。

以下の手順では、 AWS CLI の最新版をすでにインストールしているか、最新版に更新しているも のとします。詳細については、「<u>AWS Command Line Interfaceのインストール</u>」を参照してくださ い。

レポートグループのタグを表示する

タグは、AWS リソースを識別して整理し、リソースへのアクセスを管理するのに役立ちます。タ グの使用の詳細については、「<u>タグ付けのベストプラクティス</u>」ホワイトペーパーを参照してくだ さい。タグベースのアクセスポリシーの例については、「<u>Deny or allow actions on report groups</u> based on resource tags」を参照してください。

レポートグループのタグを表示する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに関連付けられたタグを表示できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Report groups (レポートグループ)] で、タグを表示するレポートグループの名前を選択します。
- 3. ナビゲーションペインで [設定] を選択します。

レポートグループのタグを表示する (AWS CLI)

を使用してレポートグループの AWS タグ AWS CLI を表示するには、次の手順に従います。いずれ のタグも追加されていない場合、返されるタグは空になります。 1. コンソールまたは AWS CLI を使用して、レポートグループの ARN を見つけます。その ARN を メモしておきます。

AWS CLI

次のコマンドを実行します。

```
aws list-report-groups
```

このコマンドは、以下のような JSON 形式の情報を返します。



レポートグループの ARN はそのグループの名前で終わります。この名前はレポートグルー プの ARN を識別するために使用できます。

Console

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Report groups (レポートグループ)] で、表示するタグが付いたレポートグループの名前 を選択します。
- 3. [Configuration (設定)] で、レポートグループの ARN を見つけます。
- 次のコマンドを実行します。--report-group-arns パラメータにはメモしておいた ARN を 使用します。

aws codebuild batch-get-report-groups --report-group-arns
arn:aws:codebuild:region:123456789012:report-group/report-group-name

成功すると、このコマンドは、以下のような tags セクションを含む JSON 形式の情報を返し ます。

{

レポートグループにタグを付ける

```
"tags": {
    "Status": "Secret",
    "Project": "TestBuild"
  }
   ...
}
```

レポートグループのタグを編集する

レポートグループに関連付けられたタグの値を変更できます。キーの名前を変更することもできま す。これは、現在のタグを削除して、新しい名前と他のタグと同じ値を持つ、別のタグを追加するこ とになります。キーフィールドと値フィールドに使用できる文字には制限があります。詳細について は、「[タグ]」を参照してください。

▲ Important

レポートグループのタグを編集すると、そのレポートグループへのアクセスに影響を与える 可能性があります。レポートグループのタグの名前 (キー) または値を編集する前に、タグの キーや値を使用してレポートグループなどのリソースへのアクセスをコントロールする可能 性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例につい ては、「<u>Deny or allow actions on report groups based on resource tags</u>」を参照してくださ い。

レポートグループのタグを編集する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに関連付けられたタグを編集できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Report groups (レポートグループ)] で、タグを編集するレポートグループの名前を選択します。
- 3. ナビゲーションペインで[設定]を選択します。
- 4. [編集]を選択します。
- 5. 次のいずれかを行ってください。
 - タグを変更するには、[Key] に新しい名前を入力します。タグの名前を変更することは、タグ を削除して、新しいキー名を持つタグを追加することになります。

- タグの値を変更するには、新しい値を入力します。値を空にする場合は、現在の値を削除して フィールドを空のままにします。
- 6. タグの編集を完了したら、[Submit] を選択します。

レポートグループのタグを編集する (AWS CLI)

レポートグループのタグを追加、変更、または削除するには、「<u>レポートグループの更新 (CLI)</u>」を 参照してください。UpdateReportGroupInput.json のタグを更新します。

レポートグループからタグを削除

レポートグループに関連付けられた1つ以上のタグを削除できます。タグを削除しても、そのタグ に関連付けられている他の AWS リソースからタグは削除されません。

▲ Important

レポートグループからタグを削除すると、そのレポートグループへのアクセスに影響を与え る可能性があります。レポートグループからタグを削除する前に、タグのキーや値を使用し てレポートグループなどのリソースへのアクセスをコントロールする可能性のある IAM ポリ シーを必ず確認してください。タグベースのアクセスポリシーの例については、「<u>タグを使</u> <u>用した AWS CodeBuild リソースへのアクセスのコントロール</u>」を参照してください。

レポートグループからタグを削除する (コンソール)

CodeBuild コンソールを使用して、タグと レポートグループとの関連付けを解除できます。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- 2. [Report groups (レポートグループ)] で、タグを削除するレポートグループの名前を選択します。
- 3. ナビゲーションペインで[設定]を選択します。
- 4. [Edit]を選択します。
- 5. 削除するタグを見つけ、[Remove tag] を選択します。
- 6. タグの削除を完了したら、[Submit] を選択します。

レポートグループからタグを削除する (AWS CLI)

を使用して CodeBuild レポートグループからタグ AWS CLI を削除するには、次の手順に従います。 タグを削除してもそのタグがなくなるわけではありません。タグとレポートグループとの関連付けが 解除されるだけです。

Note

CodeBuild レポートグループを削除すると、削除されたレポートグループからすべてのタグの関連付けが解除されます。レポートグループを削除する前にタグを削除する必要はありません。

レポートグループから 1 つ以上のタグを削除するには、「<u>レポートグループのタグを編集する (AWS</u> <u>CLI)</u>」を参照してください。JSON 形式のデータの tags セクションを、削除するタグが含まれてい ない最新のタグのリストで更新します。すべてのタグを削除する場合は、tags セクションを以下の ように更新します。

"tags: []"

レポートグループの更新

レポートグループを更新するときは、生のテスト結果データを Amazon S3 バケット内のファイルに エクスポートするかどうかに関する情報を指定できます。S3 バケットへのエクスポートを選択した 場合は、レポートグループについて以下を指定します。

- 生のテスト結果ファイルが ZIP ファイルに圧縮されているかどうか。
- 生のテスト結果ファイルが暗号化されているかどうか。次のいずれかの方法で暗号化を指定できます。
 - Amazon S3 AWS マネージドキー 用の 。
 - ユーザーが作成して設定するカスタマー管理のキー。

詳細については、「データの暗号化」を参照してください。

を使用してレポートグループ AWS CLI を更新する場合は、タグを更新または追加することもでき ます。詳細については、「<u>でレポートグループにタグを付ける AWS CodeBuild</u>」を参照してくださ い。 Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするア クセス許可に使用されます。

トピック

- レポートグループの更新 (コンソール)
- レポートグループの更新 (CLI)

レポートグループの更新 (コンソール)

次の手順で、 AWS Management Consoleを使用してレポートグループを更新します。

レポートグループを更新するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// www.com で開きます。
- 2. ナビゲーションペインで、[Report groups (レポートグループ)]を選択します。
- 3. 更新するレポートグループを選択します。
- 4. [編集]を選択します。
- 5. [Backup to Amazon S3] (Amazon S3 にバックアップ) を選択または選択解除します。このオプ ションを選択した場合は、エクスポート設定を指定します。
 - a. [S3 bucket name (S3 バケット名)] に、S3 バケットの名前を入力します。
 - b. [Path prefix (パスプレフィックス)] に、テスト結果をアップロードする S3 バケットのパス を入力します。
 - c. 生のテスト結果データファイルを圧縮するには、[Compress test result data in a zip file (テ スト結果データを圧縮する)] を選択します。
 - d. [Additional configuration (追加の設定)] を展開して、暗号化オプションを表示します。次の いずれかを選択します。
 - Amazon S3 AWS マネージドキーのを使用するデフォルトのAWS マネージドキー。詳細については、AWS Key Management Service ユーザーガイドの「<u>カスタマーマネージ</u>ド CMKs」を参照してください。これはデフォルトの暗号化オプションです。

- カスタムキーを選択して、ユーザーが作成して設定するカスタマー管理のキーを使用 します。AWS KMS 暗号化キーの場合は、暗号化キーの ARN を入力します。形式は arn:aws:kms:<*region-id*>: <*aws-account-id*>:key/<*key-id*> です。詳細に ついては、AWS Key Management Service ユーザーガイドの「<u>KMS キーの作成</u>」を参照 してください。
- ・暗号化を無効にするには、アーティファクト暗号化を無効にします。テスト結果を共有したり、静的ウェブサイトに公開したりする場合は、このオプションを選択します。(動的ウェブサイトでは、テスト結果を復号化するコードを実行できます)。

レポートグループの更新 (CLI)

次の手順で、 AWS CLIを使用してレポートグループを更新します。

レポートグループを更新するには

- 1. UpdateReportGroupInput.jsonという名前のファイルを作成します。
- 2. 以下を UpdateReportGroupInput.json にコピーします。

```
{
    "arn": "",
    "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
            "bucket": "bucket-name",
            "path": "path",
            "packaging": "NONE | ZIP",
            "encryptionDisabled": "false",
            "encryptionKey": "your-key"
         }
     },
     "tags": [
        {
            "key": "tag-key",
            "value": "tag-value"
        }
     ]
}
```

- 3. レポートグループの ARN を arn 行に入力します ("arn":"arn:aws:codebuild:*region:123456789012*:report-group/*report-group-1*") など)。
- 4. レポートグループに適用する更新内容で UpdateReportGroupInput.json を更新します。
 - レポートグループを更新して生のテスト結果ファイルをS3バケットにエクスポートする場合は、exportConfig セクションを更新します。bucket-name をS3バケット名に、pathをファイルをエクスポートするS3バケット内のパスに置き換えます。エクスポートされたファイルをpackagingに圧縮する場合は、ZIPを指定します。それ以外の場合は、NONEを指定します。エクスポートされたファイルを暗号化するかどうかを指定するためにencryptionDisabledを使用します。エクスポートしたファイルを暗号化する場合は、カスタマー管理のキーを入力します。
 - 生のテスト結果ファイルをS3バケットにエクスポートしないようにレポートグループを更新 する場合は、exportConfig セクションを以下の JSON で更新します。

{
 "exportConfig": {
 "exportConfigType": "NO_EXPORT"
 }
}

レポートグループのタグを更新する場合は、tags セクションを更新します。タグは変更、追加、または削除できます。すべてのタグを削除する場合は、以下の JSON で更新します。

"tags": []

5. 次のコマンドを実行してください。

aws codebuild update-report-group \
--cli-input-json file://UpdateReportGroupInput.json

テストフレームワーク

このセクションのトピックでは、さまざまなテストフレームワーク AWS CodeBuild の でテストレ ポートを設定する方法を示します。

トピック

• Jasmine によるテストレポートのセットアップ

- Jest によるテストレポートのセットアップ
- pytest によるテストレポートのセットアップ
- RSpec を使用したテストレポートのセットアップ

Jasmine によるテストレポートのセットアップ

次の手順は、JasmineBDD テストフレームワーク AWS CodeBuild を使用して でテストレポートを 設定する方法を示しています。 JasmineBDD

この手順には、次の前提条件が必要です。

- ・既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、Jasmine テストフレームワークを使用するようにセットアップされた Node.js プロジェクトである。

jasmine-reporters パッケージを devDependencies セクションの package.json ファイル に追加します。このパッケージには、Jasmine で使用できる JavaScript レポータークラスのコレク ションがあります。

```
npm install --save-dev jasmine-reporters
```

まだ存在しない場合は、test スクリプトをプロジェクトの package . json ファイルに追加しま す。test スクリプトは、npm test が実行されたときに Jasmine が確実に呼び出されるようにしま す。

```
{
   "scripts": {
    "test": "npx jasmine"
   }
}
```

CodeBuild は、以下の Jasmine テストレポーターをサポートしています。

JUnitXmlReporter

JunitXml 形式でレポートを生成するために使用されます。

NUnitXmlReporter

NunitXml 形式でレポートを生成するために使用されます。

Jasmine を使用する Node.js プロジェクトには、デフォルトで Jasmine 設定とテストスクリプトを 含む spec サブディレクトリが作成されます。

JunitXML 形式でレポートを生成するように Jasmine を設定するには、テストに次のコードを追加 して、JUnitXmlReporter レポーターをインスタンス化します。

```
var reporters = require('jasmine-reporters');
var junitReporter = new reporters.JUnitXmlReporter({
    savePath: <test report directory>,
    filePrefix: <report filename>,
    consolidateAll: true
});
jasmine.getEnv().addReporter(junitReporter);
```

NunitXML 形式でレポートを生成するように Jasmine を設定するには、テストに次のコードを追加 して、NUnitXmlReporter レポーターをインスタンス化します。

```
var reporters = require('jasmine-reporters');
var nunitReporter = new reporters.NUnitXmlReporter({
   savePath: <test report directory>,
   filePrefix: <report filename>,
   consolidateAll: true
});
```

jasmine.getEnv().addReporter(nunitReporter)

テストレポートは、*<test report directory>/<report filename>* で指定されたファイルに エクスポートされます。

buildspec.yml ファイルで、次のセクションを追加/更新します。

version: 0.2

phases:

Jasmine をセットアップ

```
pre_build:
    commands:
        - npm install
build:
    commands:
        - npm build
        - npm test
reports:
    jasmine_reports:
    files:
        - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

NunitXml レポート形式を使用している場合は、file-format 値を次のように変更します。

file-format: NUNITXML

Jest によるテストレポートのセットアップ

次の手順は、Jest テストフレームワーク AWS CodeBuild を使用して でテストレポートを設定する 方法を示しています。 https://jestjs.io/

この手順には、次の前提条件が必要です。

- ・既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、Jest テストフレームワークを使用するようにセットアップされた Node.js プロジェクトである。

<u>jest-junit</u> パッケージを devDependencies セクションの package.json ファイルに追加しま す。CodeBuild では、このパッケージを使用して、レポートを JunitXml の形式で生成します。

npm install --save-dev jest-junit

まだ存在しない場合は、test スクリプトをプロジェクトの package.json ファイルに追加しま す。test スクリプトは、npm test が実行されたときに Jest が確実に呼び出されるようにします。

{

```
ユーザーガイド
```

```
"scripts": {
    "test": "jest"
}
```

Jest の設定ファイルに以下を追加して、JunitXml レポーターを使用するよう Jest を設定しま す。プロジェクトに Jest 設定ファイルがない場合は、プロジェクトのルートに jest.config.js という名前のファイルを作成し、以下を追加します。テストレポートは、<test report directory>/<report filename> で指定されたファイルにエクスポートされます。

```
module.exports = {
  reporters: [
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

buildspec.yml ファイルで、次のセクションを追加/更新します。

```
version: 0.2
phases:
    pre_build:
        commands:
            - npm install
    build:
        commands:
            - npm build
            - npm test

reports:
    jest_reports:
    files:
            - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```
pytest によるテストレポートのセットアップ

次の手順は、<u>pytest テストフレームワーク</u> AWS CodeBuild を使用して でテストレポートを設定する 方法を示しています。

この手順には、次の前提条件が必要です。

- ・ 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、pytest テストフレームワークを使用するようにセットアップされた Python プロジェクトである。

build ファイルの post_build または buildspec.yml フェーズに、次のエントリを追加 します。このコードは、自動的に現在のディレクトリ内でテストを検出し、<test report directory>/<report filename> で指定されたファイルにテストレポートをエクスポートしま す。レポートでは、JunitXml 形式が使用されます。

- python -m pytest --junitxml=<test report directory>/<report filename>

buildspec.yml ファイルで、次のセクションを追加/更新します。

```
version: 0.2
phases:
  install:
    runtime-versions:
      python: 3.7
    commands:
      - pip3 install pytest
  build:
    commands:
      - python -m pytest --junitxml=<test report directory>/<report filename>
reports:
  pytest_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

RSpec を使用したテストレポートのセットアップ

次の手順は、RSpec テストフレームワーク AWS CodeBuild を使用して でテストレポートを設定す る方法を示しています。 RSpec

この手順には、次の前提条件が必要です。

- ・ 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、RSpec テストフレームワークを使用するようにセットアップされた Ruby プロジェクトである。

buildspec.yml ファイルに以下を追加/更新します。このコードは、<test source directory> ディレクトリでテストを実行し、<test report directory>/<report filename> で指定されたファイルにテストレポートをエクスポートします。レポートで は、JunitXml 形式が使用されます。

```
version: 0.2
phases:
  install:
    runtime-versions:
      ruby: 2.6
  pre_build:
    commands:
      - gem install rspec
      - gem install rspec_junit_formatter
  build:
    commands:
      - rspec <test source directory>/* --format RspecJunitFormatter --out <test report
 directory>/<report filename>
reports:
    rspec_reports:
        files:
            - <report filename>
        base-directory: <test report directory>
        file-format: JUNITXML
```

テストレポートの表示

テストケースに関する情報、合格番号と不合格番号、実行にかかった時間など、テストレポートに関 する詳細を表示できます。ビルド実行、レポートグループ、または AWS アカウント別にグループ化 されたテストレポートを表示できます。コンソールでテストレポートを選択すると、テストケースの 詳細と結果が表示されます。

期限切れでないテストレポートを表示できます。テストレポートは、作成から 30 日後に有効期限が 切れます。CodeBuild で期限切れのレポートを表示することはできません。

トピック

- ビルドのテストレポートの表示
- レポートグループのテストレポートの表示
- AWS アカウントでのテストレポートの表示

ビルドのテストレポートの表示

ビルドのテストレポートを表示するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 表示するビルドを見つけます。テストレポートを作成したビルドを実行したプロジェクトがわかっている場合:
 - 1. ナビゲーションペインで、[Build projects (ビルドプロジェクト)] を選択し、表示するテストレポートを実行したビルドを含むプロジェクトを選択します。
 - 2. [Build history (ビルド履歴)] を選択し、表示するレポートを作成したビルドを選択します。

AWS アカウントのビルド履歴でビルドを見つけることもできます。

- 1. ナビゲーションペインで [Build history (ビルド履歴)] を選択し、表示するテストレポートを作 成したビルドを選択します。
- 3. ビルドページで [Reports (レポート)] を選択し、テストレポートを選択して詳細を確認します。

レポートグループのテストレポートの表示

レポートグループ内のテストレポートを表示するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで、[Report groups (レポートグループ)]を選択します。
- 3. 表示するテストレポートを含むレポートグループを選択します。
- 4. テストレポートを選択すると、その詳細が表示されます。

AWS アカウントでのテストレポートの表示

AWS アカウントのテストレポートを表示するには

- 1. AWS CodeBuild コンソールを <u>https://console.aws.amazon.com/codesuite/codebuild/home</u>:// https://https://https://https://https
- 2. ナビゲーションペインで [Report history (レポート履歴)] を選択します。
- 3. テストレポートを選択すると、その詳細が表示されます。

テストレポートのアクセス許可

このトピックでは、テストレポートに関連するアクセス権限に関する重要な情報について説明しま す。

トピック

- <u>テストレポートの IAM ロール</u>
- テストレポートオペレーションのアクセス許可
- テストレポートのアクセス許可の例

テストレポートの IAM ロール

テストレポートを実行し、テストレポートを含めるようにプロジェクトを更新するには、IAM ロー ルに以下のアクセス権限が必要です。これらのアクセス許可は、事前定義された AWS マネージドポ リシーに含まれています。既存のビルドプロジェクトにテストレポートを追加する場合は、これらの アクセス権限を自分で追加する必要があります。

- CreateReportGroup
- CreateReport
- UpdateReport
- BatchPutTestCases

コードカバレッジレポートを実行するには、IAM ロールに BatchPutCodeCoverages アクセス許 可が付与されている必要もあります。

Note

BatchPutTestCases、CreateReport、UpdateReport、および BatchPutCodeCoverages はパブリック権限ではありません。これらのアクセス許可に対 応する AWS CLI コマンドまたは SDK メソッドを呼び出すことはできません。

これらのアクセス許可があることを確認するには、次のポリシーを IAM ロールにアタッチします。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

このポリシーは、使用する必要があるレポートグループだけに制限することをお勧めします。以下の 例では、ポリシー内の 2 つの ARN を持つレポートグループのみにアクセス権限を制限します。

```
"Effect": "Allow",
```

テストレポートの IAM ロール

{



以下の例では、my-project という名前のプロジェクトのビルドを実行することによって作成され たレポートグループのみにアクセス権限を制限しています。



Note

プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするア クセス許可に使用されます。

テストレポートオペレーションのアクセス許可

次のテストレポート CodeBuild API オペレーションのアクセス権限を指定できます。

- BatchGetReportGroups
- BatchGetReports
- CreateReportGroup
- DeleteReportGroup
- DeleteReport
- DescribeTestCases
- ListReportGroups
- ListReports
- ListReportsForReportGroup
- UpdateReportGroup

詳細については、「AWS CodeBuild アクセス許可リファレンス」を参照してください。

テストレポートのアクセス許可の例

テストレポートに関連するサンプルポリシーの詳細については、以下を参照してください。

- ・レポートグループの変更をユーザーに許可する
- レポートグループの作成をユーザーに許可する
- ・レポートの削除をユーザーに許可する
- レポートグループの削除をユーザーに許可する
- レポートグループに関する情報の取得をユーザーに許可する
- レポートに関する情報の取得をユーザーに許可する
- レポートグループの一覧表示をユーザーに許可する
- ・レポートの一覧表示をユーザーに許可する
- レポートグループのレポートの一覧表示をユーザーに許可する
- レポートのテストケースの一覧表示をユーザーに許可する。

テストレポートのステータス

テストレポートのステータスは、次のいずれかになります。

• GENERATING: テストケースの実行はまだ進行中です。

- DELETING: テストレポートは削除されています。テストレポートが削除されると、そのテスト ケースも削除されます。S3 バケットにエクスポートされた生のテスト結果データファイルは削除 されません。
- INCOMPLETE: テストレポートは完了していません。このステータスは、次のいずれかの理由で返 されることがあります。
 - レポートのテストケースを指定するレポートグループの設定に問題があります。たとえば、buildspec ファイルのレポートグループのテストケースへのパスが正しくない可能性があります。
 - ビルドを実行した IAM ユーザーには、テストを実行するアクセス権限がありません。詳細については、「テストレポートのアクセス許可」を参照してください。
 - テストに関連していないエラーのため、ビルドは完了しませんでした。
- SUCCEEDED: すべてのテストケースが成功しました。
- FAILED: いくつかのテストケースは成功しませんでした。

各テストケースは、ステータスを返します。テストケースのステータスは、次のいずれかになりま す。

- SUCCEEDED: テストケースが成功しました。
- FAILED: テストケースが失敗しました。
- ERROR: テストケースで予期しないエラーが発生しました。
- SKIPPED: テストケースは実行されませんでした。
- UNKNOWN: テストケースが、SUCCEEDED、FAILED、ERROR、SKIPPED 以外のステータスを返し ました。

テストレポートには、最大 500 件のテストケース結果を設定できます。500 を超えるテストケース が実行されている場合、CodeBuild はステータス FAILED でテストの優先順位を付け、テストケー スの結果を切り捨てます。

Amazon Virtual Private Cloud AWS CodeBuild で を使用する

通常、 AWS CodeBuild は VPC 内のリソースにアクセスできません。アクセスできるようにするに は、CodeBuild プロジェクト設定で追加の VPC 固有の設定情報を指定する必要があります。これに は、VPC ID、VPC サブネット ID、および VPC セキュリティグループ ID が含まれます。これによ り、VPC 対応のビルドは VPC 内のリソースにアクセスできます。Amazon VPC で VPC を設定する 方法の詳細については、Amazon VPC ユーザーガイドを参照してください。

トピック

- ユースケース
- VPC のベストプラクティス
- VPC の制限事項
- CodeBuild プロジェクトでの Amazon VPC アクセスを許可
- VPC 設定のトラブルシューティング
- VPC エンドポイントの使用
- マネージドプロキシサーバー AWS CodeBuild で を使用する
- ・ プロキシサーバー AWS CodeBuild で を使用する
- ・ AWS CloudFormation VPC テンプレート

ユースケース

AWS CodeBuild ビルドからの VPC 接続により、次のことが可能になります。

- プライベートサブネット上に分離された Amazon RDS データベース内のデータに対して、ビルド から統合テストを実行する。
- Amazon ElastiCache クラスターのデータをテストから直接クエリする。
- Amazon EC2、Amazon ECS、または内部 Elastic Load Balancing を使用するサービスでホストさ れる内部ウェブサービスを操作する。
- Python 用 PyPI、Java 用 Maven、Node.js 用 npm など、セルフホスト型の内部アーティファクト リポジトリから依存関係を取得する。
- Amazon VPC エンドポイント経由でのみアクセスできるように設定された S3 バケット内のオブ ジェクトにアクセスする。

 固定 IP アドレスを必要とする外部ウェブサービスを、サブネットに関連付けられた NAT ゲート ウェイまたは NAT インスタンスの Elastic IP アドレスを使用してクエリする。

お客様のビルドは、VPC でホストされている任意のリソースにアクセスできます。

VPC のベストプラクティス

CodeBuild を使用するように VPC を設定する場合は、このチェックリストを使用します。

- パブリックおよびプライベートサブネットと NAT ゲートウェイを使用して VPC を設定します。NAT ゲートウェイはパブリックサブネットにある必要があります。詳細については、Amazon VPC ユーザーガイドの「パブリックサブネットとプライベートサブネットを持つ VPC (NAT)」を参照してください。
 - A Important

VPC で CodeBuild を使用するには、NAT ゲートウェイまたは NAT インスタンスが必要 です。これにより、CodeBuild はパブリックエンドポイントにアクセスできるようにな ります (ビルドの実行時に CLI コマンドを実行する場合など)。CodeBuild は、作成した ネットワークインターフェイスへの Elastic IP アドレスの割り当てをサポートしていない ため、NAT ゲートウェイや NAT インスタンスの代わりにインターネットゲートウェイを 使用することはできません。また、Amazon EC2 は、 Amazon EC2 インスタンスの起動 以外で作成されたネットワークインターフェイスに対しては、パブリック IP アドレスの自 動割り当てをサポートしていません。

- VPCに複数のアベイラビリティーゾーンを含めます。
- セキュリティグループに、ビルドに許可されたインバウンド (進入) トラフィックがないこと を確認します。CodeBuild にはアウトバウンドトラフィックに関する特定の要件はありません が、GitHub や Amazon S3 など、ビルドに必要なインターネットリソースへのアクセスを許可す る必要があります。

詳細については、「Amazon VPC ユーザーガイド」の「<u>セキュリティグループルール</u>」を参照し てください。

- ・ビルド用に別個のサブネットを設定します。
- VPC にアクセスするように CodeBuild プロジェクトを設定する場合、プライベートサブネットの みを選択します。

Amazon VPC で VPC を設定する方法の詳細については、<u>Amazon VPC ユーザーガイド</u>を参照して ください。

AWS CloudFormation を使用して CodeBuild VPC 機能を使用するように VPC を設定する方法の詳 細については、「」を参照してくださいAWS CloudFormation VPC テンプレート。

VPC の制限事項

• CodeBuild からの VPC 接続は、共有 VPC ではサポートされていません。

CodeBuild プロジェクトでの Amazon VPC アクセスを許可

以下の設定を VPC 設定に含めます。

- [VPC ID] で、CodeBuild が使用する VPC ID を選択します。
- ・ [サブネット] で、CodeBuild が使用するリソースへのルートを含む NAT 変換を持つプライベート サブネットを選択します。
- [セキュリティグループ] で、CodeBuild が VPC 内のリソースへのアクセスを許可するために使用 するセキュリティグループを選択します。

コンソールを使用してビルドプロジェクトを作成する方法については、「<u>ビルドプロジェクトの作</u> <u>成 (コンソール)</u>」を参照してください。CodeBuild プロジェクトを作成または変更する場合、[VPC] で、VPC ID、サブネット、セキュリティグループを選択します。

を使用してビルドプロジェクトを作成するには、 AWS CLI 「」を参照してください<u>ビルドプロジェ</u> <u>クトの作成 (AWS CLI)</u>。CodeBuild AWS CLI で を使用している場合、CodeBuild が IAM ユーザーに 代わって サービスとやり取りするために使用するサービスロールには、ポリシーがアタッチされて いる必要があります。詳細については、「<u>VPC ネットワークインターフェイスの作成に必要な AWS</u> <u>サービスへの CodeBuild アクセスを許可する</u>」を参照してください。

vpcConfig オブジェクトには、*vpcId、securityGroupIds、*および *subnets* が含まれている 必要があります。

vpcId: 必須。CodeBuild で使用される VPC ID。リージョン内の Amazon VPC ID を一覧表示する
 には、次のコマンドを実行します。

aws ec2 describe-vpcs

subnets: 必須。CodeBuild で使用されるリソースを含むサブネット ID。この ID を取得するには、次のコマンドを実行します。

aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1

Note

us-east-1は、実際のリージョンに置き換えます。

 securityGroupIds: 必須。VPC内のリソースへのアクセスを許可するために CodeBuild で使用 されるセキュリティグループ ID。この ID を取得するには、次のコマンドを実行します。

aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region useast-1

Note

us-east-1は、実際のリージョンに置き換えます。

VPC 設定のトラブルシューティング

エラーメッセージに表示される情報を、問題の特定、診断、対処に役立てます。

一般的な CodeBuild VPC エラー「Build does not have internet connectivity. Please check subnet network configuration」のトラブルシューティングに役立つガイドラインを以下に示します。

- 1. インターネットゲートウェイが VPC にアタッチされていることを確認します。
- パブリックサブネットのルートテーブルがインターネットゲートウェイを参照していることを確認します。
- 3. ネットワーク ACL がトラフィックのフローを許可していることを確認します。
- 4. セキュリティグループがトラフィックのフローを許可していることを確認します。
- 5. NAT ゲートウェイのトラブルシューティングを行います。

- 6. <u>プライベートサブネットのルートテーブルが NAT ゲートウェイを参照していることを確認しま</u> す。
- IAM ユーザーに代わってサービスを操作するために CodeBuild が使用するサービスロール に、<u>このポリシー</u>のアクセス許可が付与されていることを確認します。詳細については、 「CodeBuild が他の AWS のサービスとやり取りすることを許可」を参照してください。

CodeBuild にアクセス許可がない場合は、「Unexpected EC2 error: UnauthorizedOperation」というエラーが表示されることがあります。このエラーは、VPC を使用するために必要な Amazon EC2 へのアクセス許可を CodeBuild が持っていない場合に発 生することがあります。

VPC エンドポイントの使用

インターフェイス VPC エンドポイントを使用する AWS CodeBuild ように を設定することで、ビル ドのセキュリティを向上させることができます。インターフェイスエンドポイントは、プライベート IP アドレスを通じて Amazon EC2 および CodeBuild にプライベートにアクセスできるテクノロジー である PrivateLink を使用しています。PrivateLink は、マネージドインスタンス、CodeBuild および Amazon EC2 間のすべてのネットワークトラフィックを Amazon ネットワークに限定します。(マ ネージドインスタンスはインターネットにアクセスできません)。また、インターネットゲートウェ イ、NAT デバイスあるいは仮想プライベートゲートウェイの必要はありません。PrivateLink の設定 は要件ではありませんが、推奨されます。PrivateLink エンドポイントと VPC エンドポイントの詳細 については、「とは AWS PrivateLink」を参照してください。

VPC エンドポイントを作成する前に

の VPC エンドポイントを設定する前に AWS CodeBuild、次の制限と制約に注意してください。

Note

Amazon VPC PrivateLink 接続をサポートしていない AWS サービスで CodeBuild を使用する 場合は、NAT ゲートウェイを使用します。

VPC エンドポイントは、Amazon Route 53 を介してのみ Amazon 提供の DNS をサポートします。独自の DNS を使用する場合には、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「DHCP オプションセット」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リクエストをサポートしていません。ビルドの入 出力を保存する S3 バケットと同じ AWS リージョンにエンドポイントを作成してください。バ ケットの場所は、Amazon S3 コンソールまたは get-bucket-locationコマンドを使用して確認でき ます。リージョン固有の Amazon S3 エンドポイントを使用してバケットにアクセスします (例: <bucket-name>.s3-us-west-2.amazonaws.com)。Amazon S3 のリージョン固有のエンド ポイントの詳細については、「Amazon Web Services 全般のリファレンス」の「Amazon Simple Storage Service」を参照してください。を使用して Amazon S3 にリクエスト AWS CLI を行う場 合は、デフォルトのリージョンをバケットが作成されたリージョンと同じリージョンに設定する か、リクエストで --regionパラメータを使用します。

CodeBuild の VPC エンドポイントを作成

「インターフェイスエンドポイントの作成」の手順に従って、エンドポイント

com.amazonaws.*region*.codebuild を作成します。これは の VPC エンドポイントです AWS CodeBuild。

Service Name	com.amazonaws.us-west-2.codebuild			
	Q Filter by attributes			
		Service Name	Owner	Туре
	0	com.amazonaws.us-west-2.cloudformation	amazon	Interface
	۲	com.amazonaws.us-west-2.codebuild	amazon	Interface
	\bigcirc	com.amazonaws.us-west-2.codebuild-fips	amazon	Interface
	\bigcirc	com.amazonaws.us-west-2.dynamodb	amazon	Gateway
	0	com.amazonaws.us-west-2.ec2	amazon	Interface

region は、米国東部 (オハイオ) AWS リージョンなど、CodeBuild でサポートされているリージョ ンus-east-2のリージョン識別子を表します。サポートされている AWS リージョンのリストにつ いては、「 AWS 全般のリファレンス」のCodeBuild」を参照してください。エンドポイントには、 サインイン時に指定したリージョンが事前に入力されています AWS。リージョンを変更すると、そ れに応じて VPC エンドポイントが更新されます。

CodeBuild 用の VPC エンドポイントポリシーを作成する

Amazon VPC エンドポイントのポリシーを作成して、以下 AWS CodeBuild を指定できます。

- アクションを実行できるプリンシパル。
- ・ 実行可能なアクション。
- ・ 自身に対してアクションを実行できたリソース。

次のポリシー例では、すべてのプリンシパルが project-name プロジェクトのビルドの開始と表示のみ行えることを示します。

```
{
    "Statement": [
        {
          "Action": [
             "codebuild:ListBuildsForProject",
             "codebuild:StartBuild",
             "codebuild:BatchGetBuilds"
             ],
             "Effect": "Allow",
             "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
             "Principal": "*"
        }
    ]
}
```

詳細については、Amazon VPC ユーザーガイドの「<u>VPC エンドポイントによるサービスのアクセス</u> <u>コントロール</u>」を参照してください。

マネージドプロキシサーバー AWS CodeBuild で を使用する

マネージドプロキシサーバーで AWS CodeBuild リザーブドキャパシティフリートを実行するには、 プロキシルールを使用して外部サイトとの間のトラフィックを許可または拒否するようにプロキシ サーバーを設定する必要があります。なお、マネージドプロキシサーバーでのリザーブドキャパシ ティフリートの実行は、VPC、Windows、または MacOS ではサポートされていません。

▲ Important

- フリートにプロキシ設定が存在する期間に応じて、追加料金が発生します。詳細について
- は、「https://aws.amazon.com/codebuild/pricing/://www.https://https://www.https://www.https

トピック

- リザーブドキャパシティフリートのマネージドプロキシ設定を構成
- CodeBuild リザーブドキャパシティフリートを実行

リザーブドキャパシティフリートのマネージドプロキシ設定を構成

リザーブドキャパシティフリート用にマネージドプロキシサーバーを設定するには、コンソールでフ リートを作成するとき、または AWS CLIを使用してこの機能を有効にする必要があります。定義す る必要があるプロパティがいくつかあります。

[プロキシ設定を定義 - オプション]

リザーブドキャパシティインスタンスにネットワークアクセスコントロールを適用するプロキシ 設定。

デフォルトの動作

送信トラフィックの動作を定義します。

許可

デフォルトでは、すべての送信先への送信トラフィックを許可します。

拒否

デフォルトでは、すべての送信先への送信トラフィックを拒否します。

[プロキシルール]

ネットワークアクセスコントロールを制限する送信先ドメインを指定します。

コンソールでプロキシ設定を定義する手順については、「<u>リザーブドキャパシティフリートを作成</u>」 を参照してください。を使用してプロキシ設定を定義するには AWS CLI、次の JSON 構文を変更 し、結果を保存します。

"proxyConfiguration": {

]

}

JSON ファイルは次のようになります。

```
"proxyConfiguration": {
    "defaultBehavior": "DENY_ALL",
    "orderedProxyRules": [
        {
            "type": "DOMAIN",
            "effect": "ALLOW",
            "entities": [
              "github.com"
        ]
        }
    ]
}
```

CodeBuild リザーブドキャパシティフリートを実行

マネージドプロキシサーバーで AWS CodeBuild リザーブドキャパシティフリートを実行する と、CodeBuild はマネージドプロキシアドレスを使用して HTTP_PROXYおよび HTTPS_PROXY環境 変数を自動的に設定します。依存関係のあるソフトウェアに独自の設定があり、環境変数に準拠し ていない場合は、ビルドコマンドでこれらの値を参照し、ソフトウェア設定を更新して、マネージ ドプロキシ経由でビルドトラフィックを適切にルーティングできます。詳細については、「<u>でビル</u> ドプロジェクトを作成する AWS CodeBuild」および「<u>でビルドプロジェクト設定を変更する AWS</u> CodeBuild」を参照してください。

プロキシサーバー AWS CodeBuild で を使用する

プロキシサーバー AWS CodeBuild で を使用して、インターネットとの間で送受信される HTTP および HTTPS トラフィックを規制できます。プロキシサーバーで CodeBuild を実行するには、 パブリックサブネットにプロキシサーバーをインストールし、VPC のプライベートサブネットに CodeBuild をインストールします。

プロキシサーバーで CodeBuild を実行するためのプライマリユースケースが 2 つあります。

- これにより、VPC で NAT ゲートウェイや NAT インスタンスを使用する必要がなくなります。
- プロキシサーバーのインスタンスがアクセスを許可する URL と、プロキシサーバーがアクセスを 拒否する URL を指定できます。

CodeBuild は、2 種類のプロキシサーバーで使用できます。どちらの場合も、プロキシサーバーはパ ブリックサブネットで動作し、CodeBuild はプライベートサブネットで動作します。

- 明示的なプロキシ: 明示的なプロキシサーバーを使用する場合は、N0_PROXY、HTTP_PROXY、および HTTPS_PROXY の環境変数を CodeBuild のプロジェクトレベルで設定する必要があります。 詳細については、「<u>でビルドプロジェクト設定を変更する AWS CodeBuild</u>」および「<u>でビルドプ</u> ロジェクトを作成する AWS CodeBuild」を参照してください。
- Transparent Proxy: 透過的なプロキシサーバーを使用する場合は、特別な設定は必要ありません。

トピック

- プロキシサーバーで CodeBuild を実行するために必要なコンポーネントを設定
- 明示的なプロキシサーバーでの CodeBuild の実行
- 透過的なプロキシサーバーでの CodeBuild の実行
- プロキシサーバーでのパッケージマネージャーなどのツールの実行

プロキシサーバーで CodeBuild を実行するために必要なコンポーネントを 設定

これらのコンポーネントを透過的または明示的なプロキシサーバー AWS CodeBuild で実行する必要 があります。

- VPC。
- プロキシサーバー用に VPC 内の1つのパブリックサブネット。
- CodeBuild 用に VPC 内の 1 つのプライベートサブネット。
- VPC とインターネットの間の通信を可能にするインターネットゲートウェイ。



VPC、サブネット、ネットワークゲートウェイのセットアップ

透過的または明示的なプロキシサーバー AWS CodeBuild で を実行するには、次のステップが必要で す。

- 1. VPC を作成します。VPC の作成の詳細については、「<u>Amazon VPC ユーザーガイド</u>」の「VPC を作成する」をご参照ください。
- 2. VPC 内に 2 つのサブネットを作成します。1 つは、プロキシサーバーを実行する Public Subnet という名前のパブリックサブネットです。もう 1 つは、CodeBuild を実行する Private Subnet という名前のプライベートサブネットです。

詳細については、「VPC でのサブネットの作成」を参照してください。

- 3. インターネットゲートウェイを作成して VPC にアタッチします。詳細については、「<u>インター</u> ネットゲートウェイの作成とアタッチ」を参照してください。
- 4. VPC (0.0.0.0/0) からインターネットゲートウェイに送信トラフィックをルーティングするルール をデフォルトルートテーブルに追加します。詳細については、「<u>ルートテーブルでのルートの追</u> 加および削除」を参照してください。
- 5. VPC (0.0.0.0/0) からの着信 SSH トラフィック (TCP 22) を許可するルールを VPC のデフォルト セキュリティグループに追加します。

- 「Amazon EC2 ユーザーガイド」の「コンソールのインスタンス起動ウィザードを使用して EC2 インスタンスを起動する」の指示に従って Amazon Linux インスタンスを起動します。ウィザー ドを実行する場合は次のオプションを選択してください。
 - [インスタンスタイプの選択] で、Amazon Linux の Amazon マシンイメージ (AMI) を選択します。
 - [サブネット] で、このトピックで先に作成したパブリックサブネットを選択します。推奨された名前を使用した場合は、[Public Subnet] です。
 - [Auto-assign Public IP] で、[Enable] を選択します。
 - [セキュリティグループの設定] ページの [セキュリティグループの割り当て] で、[Select an existing security group (既存のセキュリティグループの選択)] を選択します。次に、デフォルトのセキュリティグループを選択します。
 - [起動]を選択したら、既存のキーペアを選択するか、新しいキーペアを作成します。

それ以外のオプションについては、デフォルト設定を選択します。

- 7. EC2 インスタンスの実行後は、送信元/送信先チェックを無効にします。詳細については、 「<u>Amazon VPC ユーザーガイド</u>」の「Disabling Source/Destination checks」を参照してくださ い。
- VPC にルートテーブルを作成します。インターネット用のトラフィックをプロキシサーバーに ルーティングするためのルールをルートテーブルに追加します。このルートテーブルをプライ ベートサブネットに関連付けます。これは、CodeBuild が実行されているプライベートサブネッ ト内のインスタンスからのアウトバウンドリクエストを、常にプロキシサーバーを介してルー ティングするために必要です。

プロキシサーバーのインストールと設定

選択できるプロキシサーバーは多数あります。ここでは、オープンソースのプロキシサーバーである Squid を使用して、プロキシサーバーで AWS CodeBuild がどのように実行されるかを示します。同 じ概念を他のプロキシサーバーにも適用できます。

Squid をインストールするには、次のコマンドを実行して yum repo を使用します。

sudo yum update -y
sudo yum install -y squid

Squid をインストールしたら、このトピックで後述する手順に従って、その squid.conf ファイル を編集します。

HTTPS トラフィック用の Squid の設定

HTTPS では、HTTP トラフィックは Transport Layer Security (TLS) 接続でカプセル化されま す。Squid では、<u>SslPeekAndSplice</u> と呼ばれる機能を使用して、リクエストされたインターネッ トホストを含む TLS 初期化から Server Name Indication (SNI) を取得します。これは必須のた め、Squid で HTTPS トラフィックを復元する必要はありません。SslPeekAndSplice を有効にする には、Squid に証明書が必要です。OpenSSL を使用してこの証明書を作成する:

sudo mkdir /etc/squid/ssl cd /etc/squid/ssl sudo openssl genrsa -out squid.key 2048 sudo openssl req -new -key squid.key -out squid.csr -subj "/C=XX/ST=XX/L=squid/0=squid/ CN=squid" sudo openssl x509 -req -days 3650 -in squid.csr -signkey squid.key -out squid.crt sudo cat squid.key squid.crt | sudo tee squid.pem

Note

HTTP では、Squid の設定は必要ありません。すべての HTTP/1.1 リクエストメッセージか ら、ホストヘッダーフィールドを取得することができます。これにより、リクエストされて いるインターネットホストが指定されます。

明示的なプロキシサーバーでの CodeBuild の実行

明示的なプロキシサーバー AWS CodeBuild で を実行するには、外部サイトとの間のトラフィックを 許可または拒否するようにプロキシサーバーを設定し、 HTTP_PR0XYおよび HTTPS_PR0XY環境変 数を設定する必要があります。

トピック

- ・ 明示的なプロキシサーバーとしての Squid の設定
- CodeBuild プロジェクトを作成する
- ・ 明示的なプロキシサーバーのサンプル squid.conf ファイル

明示的なプロキシサーバーとしての Squid の設定

Squid プロキシサーバーが明示的になるように設定するには、/etc/squid/squid.conf ファイル に次の変更を加える必要があります。 ・ 以下のデフォルトのアクセスコントロールリスト (ACL) ルールを削除します。

acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10

削除したデフォルトの ACL ルールの代わりに次のコードを追加します。最初の行では VPC から のリクエストを許可します。次の 2 つの行では、 AWS CodeBuildによって使用されている可能性 のある送信先 URL へのアクセス権をプロキシサーバーに付与します。最後の行の正規表現を編集 して、 AWS リージョンの S3 バケットまたは CodeCommit リポジトリを指定します。例:

- 送信元が Amazon S3 の場合は、acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws
 \.com コマンドを使用して、us-west-1 リージョンの S3 バケットへのアクセスを許可します。
- ソースがの場合はAWS CodeCommit、git-codecommit.<*yourregion*>.amazonaws.comを使用してリージョンAWSを許可リストに追加します。

acl localnet src 10.1.0.0/16 #Only allow requests from within the VPC acl allowed_sites dstdomain .github.com #Allows to download source from GitHub acl allowed_sites dstdomain .bitbucket.com #Allows to download source from Bitbucket acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from Amazon S3 or CodeCommit

http_access allow localnet を次のように置き換えます。

http_access allow localnet allowed_sites
http_access allow localnet download_src

- ビルドでログとアーティファクトをアップロードする場合は、次のいずれかを実行します。
 - http_access deny all ステートメントの前に、次のステートメントを挿入します。これにより、CodeBuild が CloudWatch と Amazon S3 にアクセスできるようになります。CodeBuild が CloudWatch Logs を作成できるようにするには、CloudWatch へのアクセスが必要です。Amazon S3 へのアクセスは、アーティファクトのアップロードと Amazon S3 のキャッシングを行う上で必要です。

https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept acl SSL_port port 443 http_access allow SSL_port

```
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
```

squid.confを保存した後、次のコマンドを実行します。

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. proxy を buildspec ファイルに追加します。詳細については、「<u>buildspec の構文</u>」を参照して ください。

```
version: 0.2
proxy:
    upload-artifacts: yes
    logs: yes
phases:
    build:
        commands:
        - command
```

```
    Note
```

RequestError タイムアウトエラーが表示される場合は、「<u>プロキシサーバーで CodeBuild</u> <u>を実行しているときの RequestError タイムアウトエラー</u>」を参照してください。

詳細については、このトピックで後述する「<u>明示的なプロキシサーバーのサンプル squid.conf</u> ファイル」を参照してください。

CodeBuild プロジェクトを作成する

明示的なプロキシサーバー AWS CodeBuild で を実行するには、プロキシサーバー用に作成した EC2 インスタンスのプライベート IP アドレスとポート 3128 を使用して、その HTTP_PROXYお よび HTTPS_PROXY環境変数をプロジェクトレベルで設定します。プライベート IP アドレス は、http://*your-ec2-private-ip-address*:3128 のようになります。詳細については、「<u>で</u> <u>ビルドプロジェクトを作成する AWS CodeBuild</u>」および「<u>でビルドプロジェクト設定を変更する</u> AWS CodeBuild」を参照してください。

Squid プロキシのアクセスログを表示するには、次のコマンドを使用します。

sudo tail -f /var/log/squid/access.log

明示的なプロキシサーバーのサンプル squid.conf ファイル

明示的なプロキシサーバー用に設定した squid.conf ファイルの例を次に示します。

```
acl localnet src 10.0.0.0/16 #Only allow requests from within the VPC
 # add all URLS to be whitelisted for download source and commands to be run in build
environment
 acl allowed_sites dstdomain .github.com
                                           #Allows to download source from github
 acl allowed_sites dstdomain .bitbucket.com #Allows to download source from bitbucket
 acl allowed_sites dstdomain ppa.launchpad.net #Allows to run apt-get in build
environment
 acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from S3
or CodeCommit
 acl SSL_ports port 443
 acl Safe_ports port 80 # http
 acl Safe_ports port 21 # ftp
 acl Safe_ports port 443 # https
 acl Safe_ports port 70 # gopher
 acl Safe_ports port 210 # wais
 acl Safe_ports port 1025-65535 # unregistered ports
 acl Safe_ports port 280 # http-mgmt
 acl Safe_ports port 488 # gss-http
 acl Safe_ports port 591 # filemaker
 acl Safe_ports port 777 # multiling http
 acl CONNECT method CONNECT
 #
 # Recommended minimum Access Permission configuration:
 #
 # Deny requests to certain unsafe ports
 http_access deny !Safe_ports
 # Deny CONNECT to other than secure SSL ports
 http_access deny CONNECT !SSL_ports
 # Only allow cachemgr access from localhost
 http_access allow localhost manager
 http_access deny manager
```

```
# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet allowed_sites
http_access allow localnet download_src
http_access allow localhost
# Add this for CodeBuild to access CWL end point, caching and upload artifacts S3
bucket end point
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
# And finally deny all other access to this proxy
http_access deny all
# Squid normally listens to port 3128
http_port 3128
# Uncomment and adjust the following to add a disk cache directory.
#cache_dir ufs /var/spool/squid 100 16 256
# Leave coredumps in the first cache dir
coredump_dir /var/spool/squid
#
# Add any of your own refresh_pattern entries above these.
#
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern . 0 20% 4320
```

透過的なプロキシサーバーでの CodeBuild の実行

透過的なプロキシサーバー AWS CodeBuild で を実行するには、プロキシサーバーとやり取りする ウェブサイトとドメインへのアクセスを設定する必要があります。

トピック

- 透過的なプロキシサーバーとしての Squid の設定
- CodeBuild プロジェクトを作成する

透過的なプロキシサーバーとしての Squid の設定

プロキシサーバーが透過的になるように設定するには、アクセスするドメインやウェブサイトへのアクセス権を付与する必要があります。透過的なプロキシサーバー AWS CodeBuild で を実行 するには、 へのアクセスを許可する必要がありますamazonaws.com。また、CodeBuild で使用 する他のウェブサイトへのアクセス権も付与します。これらのアクセス権は、CodeBuild プロジェ クトの作成方法によって異なります。ウェブサイトの例は、GitHub、Bitbucket、Yum、Maven な どのリポジトリ用です。特定のドメインやウェブサイトへのアクセスを Squid に許可するには、 次のようなコマンドを使用して squid.conf ファイルを更新します。このサンプルコマンドは amazonaws.com、github.com、および bitbucket.com へのアクセスを許可します。このサン プルは、他のウェブサイトへのアクセス権を付与するように編集できます。

```
cat | sudo tee /etc/squid/squid.conf #EOF
visible_hostname squid
#Handling HTTP requests
http_port 3129 intercept
acl allowed_http_sites dstdomain .amazonaws.com
#acl allowed_http_sites dstdomain domain_name [uncomment this line to add another
domain]
http_access allow allowed_http_sites
#Handling HTTPS requests
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl allowed_https_sites ssl::server_name .github.com
acl allowed_https_sites ssl::server_name .bitbucket.com
#acl allowed_https_sites ssl::server_name [uncomment this line to add another website]
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
```

```
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
http_access deny all
EOF
```

プライベートサブネット内のインスタンスからの着信リクエストで、Squid ポートにリダイレクトす る必要があります。Squid は HTTP トラフィック (80 の代理) をポート 3129、HTTPS トラフィック (443 の代理) をポート 3130 でリッスンします。トラフィックをルーティングするには、iptables コ マンドを使用します。

sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service iptables save
sudo service squid start

CodeBuild プロジェクトを作成する

プロキシサーバーを設定したら、それ以上設定することなく、プライベートサブネット AWS CodeBuild で で使用できます。HTTP および HTTPS リクエストはすべて、パブリックプロキシサー バーを経由します。Squid プロキシのアクセスログを表示するには、次のコマンドを使用します。

sudo tail -f /var/log/squid/access.log

プロキシサーバーでのパッケージマネージャーなどのツールの実行

次の手順で、パッケージマネージャーやその他のツールをプロキシサーバーで実行します。

パッケージマネージャーなどのツールをプロキシサーバーで実行する方法

- squid.conf ファイルにステートメントを追加し、プロキシサーバーの許可リストにツールを 追加します。
- 2. プロキシサーバーのプライベートエンドポイントを指す行を buildspec ファイルに追加します。

次の例では、apt-get、curl、および maven でこの作業を行う方法を示しています。別のツール を使用する場合は、同じ原則が適用されます。squid.conf ファイルの許可リストに追加し、コマ ンドを buildspec ファイルに追加して、プロキシサーバーのエンドポイントを CodeBuild に認識させ ます。

プロキシサーバーで apt-get を実行するには

 次のステートメントを squid.conf ファイルに追加し、プロキシサーバーの許可リストに apt-get を追加します。最初の3行は、apt-get がビルド環境で実行できるようにします。

acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to run in the build environment acl apt_get dstdom_regex .*\.launchpad.net # Required for CodeBuild to run apt-get in the build environment acl apt_get dstdom_regex .*\.ubuntu.com # Required for CodeBuild to run apt-get in the build environment http_access allow localnet allowed_sites http_access allow localnet apt_get

 apt-get コマンドで /etc/apt/apt.conf.d/00proxy のプロキシ設定を検索できるよう に、次のステートメントを buildspec ファイルを追加します。

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";
Acquire::https::Proxy "http://<private-ip-of-proxy-server>:3128";
Acquire::ftp::Proxy "http://<private-ip-of-proxy-server>:3128";' > /etc/apt/
apt.conf.d/00proxy
```

プロキシサーバーで curl を実行するには

1. 次の内容を squid.conf ファイルに追加し、ビルド環境の許可リストに curl を追加します。

acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the build environment acl allowed_sites dstdomain google.com # Required for access to a webiste. This example uses www.google.com. http_access allow localnet allowed_sites http_access allow localnet apt_get

 curl でプライベートプロキシサーバーを使用して squid.conf に追加したウェブサイトにア クセスできるように、次のステートメントを buildspec ファイルに追加します。この例では、 ウェブサイトは google.com です。

curl -x <private-ip-of-proxy-server>:3128 https://www.google.com

プロキシサーバーで maven を実行するには

次の内容を squid.conf ファイルに追加し、ビルド環境の許可リストに maven を追加します。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the
build environment
acl maven dstdom_regex .*\.maven.org # Allows access to the maven repository in the
build environment
http_access allow localnet allowed_sites
http_access allow localnet maven
```

2. buildspec ファイルに次のステートメントを追加します。

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

AWS CloudFormation VPC テンプレート

AWS CloudFormation では、テンプレートファイルを使用してリソースのコレクションを1つの ユニット (スタック) としてまとめて作成および削除することで、 AWS インフラストラクチャの デプロイを予測どおりに繰り返し作成およびプロビジョニングできます。詳細については、<u>AWS</u> CloudFormation ユーザーガイドをご参照ください。

使用する VPC AWS CloudFormation を設定するための YAML テンプレートを次に示します AWS CodeBuild。このファイルは「samples.zip」からも入手可能です。

```
Description: This template deploys a VPC, with a pair of public and private subnets
spread
across two Availability Zones. It deploys an internet gateway, with a default
route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
and default routes for them in the private subnets.
Parameters:
EnvironmentName:
Description: An environment name that is prefixed to resource names
Type: String
VpcCIDR:
Description: Please enter the IP range (CIDR notation) for this VPC
Type: String
```

Default: 10.192.0.0/16 PublicSubnet1CIDR: Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone Type: String Default: 10.192.10.0/24 PublicSubnet2CIDR: Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone Type: String Default: 10.192.11.0/24 PrivateSubnet1CIDR: Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone Type: String Default: 10.192.20.0/24 PrivateSubnet2CIDR: Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone Type: String Default: 10.192.21.0/24 Resources: VPC: Type: AWS::EC2::VPC **Properties:** CidrBlock: !Ref VpcCIDR EnableDnsSupport: true EnableDnsHostnames: true Tags: - Key: Name Value: !Ref EnvironmentName InternetGateway: Type: AWS::EC2::InternetGateway Properties: Tags: - Key: Name Value: !Ref EnvironmentName

```
InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
   Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)
PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ2)
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
   AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
```

```
CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1
NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes
DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
```

```
GatewayId: !Ref InternetGateway
PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1
PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2
PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
   Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)
DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1
PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1
PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)
```

```
DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2
  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2
  NoIngressSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "no-ingress-sg"
      GroupDescription: "Security group with no ingress rule"
      VpcId: !Ref VPC
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]
  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]
  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1
  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2
  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1
```

PrivateSubnet2: Description: A reference to the private subnet in the 2nd Availability Zone Value: !Ref PrivateSubnet2

NoIngressSecurityGroup:

Description: Security group with no ingress rule Value: !Ref NoIngressSecurityGroup

でのログ記録とモニタリング AWS CodeBuild

ログ記録とモニタリングは、 および AWS CodeBuild AWS ソリューションの信頼性、可用性、パ フォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をよ り簡単にデバッグできるように、 AWS ソリューションのすべての部分からモニタリングデータを収 集する必要があります。 には、CodeBuild リソースとビルドをモニタリングし、潜在的なインシデ ントに対応するための以下のツール AWS が用意されています。

トピック

- を使用した AWS CodeBuild API コールのログ記録 AWS CloudTrail
- CloudWatch で CodeBuild ビルドをモニタリング

を使用した AWS CodeBuild API コールのログ記録 AWS CloudTrail

AWS CodeBuild は AWS CloudTrail、CodeBuild のユーザー、ロール、または のサービスによって 実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、 ス タックコンソールからの呼び出しや、 スタック API へのコード呼び出しを含む、 スタックのすべ ての API コールをイベントとしてキャプチャします。証跡を作成する場合は、CodeBuild のイベ ントなど、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 で収集された情報を使用して、CodeBuild に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

トピック

- CloudTrail AWS CodeBuild の情報について
- AWS CodeBuild ログファイルエントリについて

CloudTrail AWS CodeBuild の情報について

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。CodeBuild でアクティビ ティが発生すると、そのアクティビティは [Event history] (イベント履歴) の他の AWS のサービス イベントと共に CloudTrail イベントに記録されます。 AWS アカウントで最近のイベントを表示、 検索、ダウンロードできます。詳細については、「AWS CloudTrail ユーザーガイド」の「Viewing
<u>events with CloudTrail event history」</u>(CloudTrail イベント履歴でのイベントの表示) を参照してくだ さい。

CodeBuild のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。証跡より、CloudTrail はログファイルをS3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定したS3 バケットにログファイルを配信します。CloudTrail ログで収集されたイベントデータをさらに分析して処理するように、他のAWS サービスを設定できます。詳細については、以下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」
- ・「<u>複数のリージョンから CloudTrail ログファイルを受け取る</u>」および「<u>複数のアカウントから</u> CloudTrail ログファイルを受け取る」

すべての CodeBuild アクションは CloudTrail が記録します。これらのアクションは <u>CodeBuild API</u> <u>リファレンス</u>で説明されています。たとえば、 CreateProject(では create-project) AWS CLI、 StartBuild(では start-project) AWS CLI、UpdateProjectおよび(では updateproject) アクションを呼び出すと AWS CLI、CloudTrail ログファイルにエントリが生成されま す。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。ID 情 報は次の判断に役立ちます。

- リクエストが、ルートと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、ロールと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のどちらを使用して送信されたか。
- ・ リクエストが別の AWS サービスによって行われたかどうか。

詳細については、AWS CloudTrail ユーザーガイドの <u>CloudTrail userIdentity エレメント</u>を参照してく ださい。

AWS CodeBuild ログファイルエントリについて

証跡は、指定した S3 バケットにイベントをログファイルとして配信するように設定できま す。CloudTrail ログファイルには、1 つ以上のログエントリがあります。イベントはあらゆるソース からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパ ラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けら れたスタックトレースではないため、特定の順序では表示されません。

Note

機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。

- AWS アクセスキー IDs。詳細については、AWS Identity and Access Management ユー ザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
- パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「<u>Systems Manager パラメータストア</u>」および 「<u>Systems Manager パラメータストアコンソールのチュートリアル</u>」を参照してください。
- を使用して指定された文字列 AWS Secrets Manager。詳細については、「<u>キー管理</u>」を参 照してください。

次の例は、CodeBuild でビルドプロジェクトを作成する方法を示す CloudTrail ログエントリを示して います。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "FederatedUser",
    "principalId": "account-ID:user-name",
    "arn": "arn:aws:sts::account-ID:federated-user/user-name",
    "accountId": "account-ID",
    "accessKeyId": "access-key-ID",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-09-06T17:59:10Z"
      },
      "sessionIssuer": {
        "type": "IAMUser",
        "principalId": "access-key-ID",
        "arn": "arn:aws:iam::account-ID:user/user/name",
        "accountId": "account-ID",
        "userName": "user-name"
```

```
}
    }
  },
  "eventTime": "2016-09-06T17:59:11Z",
  "eventSource": "codebuild.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "user-agent",
  "requestParameters": {
    "awsActId": "account-ID"
  },
  "responseElements": {
    "project": {
      "environment": {
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "name": "codebuild-demo-project",
      "description": "This is my demo project",
      "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project:project-ID",
      "encryptionKey": "arn:aws:kms:region-ID:key-ID",
      "timeoutInMinutes": 10,
      "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket",
        "type": "S3",
        "packaging": "ZIP",
        "outputName": "MyOutputArtifact.zip"
      },
      "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
      "lastModified": "Sep 6, 2016 10:59:11 AM",
      "source": {
        "type": "GITHUB",
        "location": "https://github.com/my-repo.git"
      },
      "created": "Sep 6, 2016 10:59:11 AM"
    }
  },
  "requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
  "eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
  "eventType": "AwsApiCall",
```

AWS CodeBuild ログファイルエントリについて

}

"recipientAccountId": "account-ID"

CloudWatch で CodeBuild ビルドをモニタリング

Amazon CloudWatch を使用してビルドをモニタリングし、異常が発生した報告して、必要に応じて 対応策を取ることができます。ビルドは、次の 2 つのレベルでモニタリングできます。

プロジェクトレベル

これらのメトリクスは、指定したプロジェクトのすべてのビルドが対象となります。プロジェク トのメトリクスを表示するには、CloudWatch でディメンションとして ProjectName を指定し ます。

AWS アカウントレベル

これらのメトリクスは、1 つのアカウントのすべてのビルドが対象となります。 AWS アカウン トレベルでメトリクスを表示するには、CloudWatch でディメンションを入力しないでくださ い。ビルドリソース使用率メトリクスは、 AWS アカウントレベルでは使用できません。

CloudWatch メトリクスには、一定期間におけるビルドの動作が示されます。たとえば、以下のことをモニタリングできます。

- ・ ビルドプロジェクトまたは AWS アカウントで試行されたビルドの数。
- ・ビルドプロジェクトまたは AWS アカウントで成功したビルドの数。
- ・ビルドプロジェクトまたは AWS アカウントで失敗したビルドの数。
- CodeBuild がビルドプロジェクトまたは AWS アカウントでビルドの実行に費やした時間。
- ・ビルドまたはビルドプロジェクト全体のリソース使用率を構築します。CPU、メモリ、ストレージ使用率などのリソース使用率メトリクスを構築します。

詳細については、「CodeBuild メトリクスを表示」を参照してください。

CodeBuild CloudWatch メトリクス

以下のメトリクスは、 AWS アカウントまたはビルドプロジェクトごとに追跡できます。CodeBuild で CloudWatch を使用する方法の詳細については、「<u>CloudWatch で CodeBuild ビルドをモニタリン</u> グ」を参照してください。

ビルドをモニタリング

BuildDuration

ビルドの BUILD フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

ビルド数

トリガーされたビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

DownloadSourceDuration

ビルドの DOWNLOAD_SOURCE フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

Duration

一定期間におけるすべてのビルドの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

FailedBuilds

クライアントエラーまたはタイムアウトのために失敗したビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

FinalizingDuration

ビルドの FINALIZING フェーズの所要時間を測定します。

単位: 秒

CloudWatch メトリクス

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum InstallDuration

ビルドの INSTALL フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum PostBuildDuration

ビルドの POST_BUILD フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

PreBuildDuration

ビルドの PRE_BUILD フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum ProvisioningDuration

ビルドの PROVISIONING フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

QueuedDuration

ビルドの QUEUED フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum SubmittedDuration

ビルドの SUBMITTED フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum SucceededBuilds

成功したビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

UploadArtifactsDuration

ビルドの UPLOAD_ARTIFACTS フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CodeBuild CloudWatch リソース使用率メトリックス

1 Note

CodeBuild リソース使用率メトリクスは、以下のリージョンでのみ利用可能です。

- Asia Pacific (Tokyo) Region
- Asia Pacific (Seoul) Region
- Asia Pacific (Mumbai) Region
- アジアパシフィック (シンガポール) リージョン
- Asia Pacific (Sydney) Region
- Canada (Central) Region
- Europe (Frankfurt) Region
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- ・ 欧州 (パリ) リージョン
- South America (São Paulo) Region
- ・米国東部 (バージニア北部) リージョン
- US East (Ohio) Region

- 米国西部 (北カリフォルニア) リージョン
- ・ 米国西部 (オレゴン) リージョン

次のリソース使用率メトリックを記録できます。CodeBuild で CloudWatch を使用する方法の詳細に ついては、「CloudWatch で CodeBuild ビルドをモニタリング」を参照してください。

CPUUtilized

ビルドコンテナで使用されている、割り当てられた処理の CPU ユニットの数。

単位: CPU 単位

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CPUUtilizedPercent

ビルドコンテナによって使用される割り当てられた処理の割合。

単位: パーセント

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum MemoryUtilized

ビルドコンテナで使用されるメモリのメガバイト数。

単位: メガバイト

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum MemoryUtilizedPercent

ビルドコンテナで使用されている、割り当てられたメモリの割合。

単位: パーセント

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum StorageReadBytes

ビルドコンテナーによって使用されるストレージの読み取り速度。

単位: バイト/秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum StorageWriteBytes

ビルドコンテナによって使用されるストレージ書き込み速度。

単位: バイト/秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CodeBuild CloudWatch のディメンション

CodeBuild には、以下の CloudWatch メトリクスディメンションが用意されています。これらのいず れも指定されていない場合、メトリクスは現在の AWS アカウントのものです。

BuildId、BuildNumber、ProjectName

メトリクスは、ビルド識別子、ビルド番号、およびプロジェクト名に対して提供されます。 ProjectName

プロジェクト名には、メトリクスが提供されます。

CodeBuild CloudWatch アラーム

CloudWatch コンソールを使用してCodeBuild メトリクスに基づいてアラームを作成できるため、 ビルドで問題が発生した場合に対応できます。アラームで最も役立つ 2 つのメトリクスについて は、次の箇条書きで説明します。CodeBuild で CloudWatch を使用する方法の詳細については、 「<u>CloudWatch で CodeBuild ビルドをモニタリング</u>」を参照してください。

- FailedBuild。事前に設定した秒数内に失敗したビルドが一定数検出されたときにトリガーされるアラームを作成できます。CloudWatchで、秒数と、アラームをトリガーするための失敗したビルド数を指定します。
- Duration。ビルドに予想より時間がかかったときにトリガーするアラームを作成できます。ビルドの開始からビルドの完了までの経過所要時間を指定します。この時間を超えるとアラームがトリガーされます。

CodeBuild メトリクスのアラームを作成する方法については、「<u>CloudWatch アラームを使用して</u> <u>CodeBuild ビルドをモニタリング</u>」を参照してください。アラームの詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch アラームの作成」を参照してください。

CloudWatch のディメンション

CodeBuild メトリクスを表示

AWS CodeBuild は、ユーザーに代わって関数をモニタリングし、Amazon CloudWatch を介してメ トリクスをレポートします。これらのメトリクスには、ビルドの合計数、失敗したビルドの数、成功 したビルドの数、ビルドの所要時間が含まれます。

CodeBuild コンソールまたは CloudWatch コンソールを使用することで、CodeBuild のメトリクスを モニタリングできます。次の手順は、メトリクスを表示する方法を示しています。

トピック

- ・ ビルドメトリクスにアクセス (CodeBuild コンソール)
- ・ビルドメトリクスを表示 (Amazon CloudWatch コンソール)

ビルドメトリクスにアクセス (CodeBuild コンソール)

Note

CodeBuild コンソールで、表示に使用されるメトリクスまたはグラフをカスタマイズすることはできません。表示をカスタマイズする場合は、Amazon CloudWatch コンソールを使用して設定されたビルドメトリクスを表示します。

アカウントレベルのメトリクス

AWS アカウントレベルのメトリクスを表示するには

- 1. にサインイン AWS Management Console し、 AWS CodeBuild コンソールを <u>https://</u> console.aws.amazon.com/codesuite/codebuild/home://www.com で開きます。
- 2. ナビゲーションペインで [Account metrics (アカウントメトリクス)] を選択します。

プロジェクトレベルのメトリクス

プロジェクトレベルのメトリクスを表示するには

- 1. にサインイン AWS Management Console し、 AWS CodeBuild コンソールを <u>https://</u> console.aws.amazon.com/codesuite/codebuild/home://www.com で開きます。
- 2. ナビゲーションペインで、[Build projects] を選択します。

- ビルドプロジェクトのリストの [名前] 列で、メトリクスを表示するプロジェクトを選択します。
- 4. [Metrics] タブを選択します。

ビルドメトリクスを表示 (Amazon CloudWatch コンソール)

CloudWatch コンソールでの表示に使用されるメトリクスまたはグラフをカスタマイズすることはで きます。

アカウントレベルのメトリクス

アカウントレベルのメトリクスを表示するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/cloudwatch/</u>:// www.com」で CloudWatch コンソールを開きます。
- 2. ナビゲーションペインで Metrics (メトリクス) を選択します。
- 3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。

Metrics						
Favorites	All metrics	Graphed metrics	Graph option	s Source		
Add a dashboard						
	Q Search for any metric, dimension or resource id					
	155 Metrics	3				
	CodeBuil	d		Events		Lambda
	44 Metrics			12 Metrics		14 Metrics

- 4. [アカウントメトリクス]を選択します。
- プロジェクトとメトリクスを1つ以上選択します。プロジェクトごと に、[SucceededBuilds]、[FailedBuilds]、[Builds]、[Duration]の各メトリクスを選択できます。 選択されたすべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されま す。

プロジェクトレベルのメトリクス

プロジェクトレベルのメトリクスを表示するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/cloudwatch/</u>:// www.com」で CloudWatch コンソールを開きます。
- 2. ナビゲーションペインで Metrics (メトリクス) を選択します。

3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。

Metrics					***			
Favorites	All metrics	Graphed metrics	Graph options	Source				
O Add a dashboard	Q Search for any metric, dimension or resource id							
	155 Metrics							
	CodeBuil	d	E	Events			Lambda	
	44 Metrics		1	2 Metrics			14 Metrics	

- 4. [By Project (プロジェクト別)] を選択します。
- プロジェクトとメトリクスの組み合わせを1つ以上選択します。プロジェクトごとに、[SucceededBuilds]、[FailedBuilds]、[Builds]、[Duration]の各メトリクスを選択できます。 選択されたすべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されます。
- (オプション) メトリクスとグラフをカスタマイズすることができます。例えば、[統計] 列のドロップダウンリストから、表示する別の統計を選択できます。または、[期間] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「<u>メトリクスのグラフ化</u>」および 「利用可能なメトリクスを表示する」を参照してください。

CodeBuild リソース使用率メトリクスを表示

AWS CodeBuild は、ユーザーに代わってビルドリソース使用率を監視し、Amazon CloudWatch を 通じてメトリクスをレポートします。これには、CPU、メモリ、ストレージ使用率などのメトリク スが含まれます。

Note

CodeBuild リソース使用率メトリクスは、1 分以上実行されるビルドに対してのみ記録されます。

CodeBuild コンソールまたは CloudWatch コンソールを使用して、CodeBuild のリソース使用率メト リクスをモニタリングできます。

Note

CodeBuild リソース使用率メトリクスは、以下のリージョンでのみ利用可能です。

- Asia Pacific (Tokyo) Region
- Asia Pacific (Seoul) Region
- Asia Pacific (Mumbai) Region
- アジアパシフィック (シンガポール) リージョン
- Asia Pacific (Sydney) Region
- Canada (Central) Region
- Europe (Frankfurt) Region
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- 欧州 (パリ) リージョン
- South America (São Paulo) Region
- ・米国東部 (バージニア北部) リージョン
- US East (Ohio) Region
- 米国西部 (北カリフォルニア) リージョン
- 米国西部 (オレゴン) リージョン

次の手順は、リソース使用率メトリクスにアクセスする方法を示しています。

トピック

- リソース使用率メトリクスへのアクセス (CodeBuild コンソール)
- リソース使用率メトリクスへのアクセス(Amazon CloudWatch コンソール)

リソース使用率メトリクスへのアクセス (CodeBuild コンソール)

Note

CodeBuild コンソールで、表示に使用されるメトリクスまたはグラフをカスタマイズすることはできません。表示をカスタマイズする場合は、Amazon CloudWatch コンソールを使用して設定されたビルドメトリクスを表示します。

プロジェクトレベルのリソース使用率メトリクス

プロジェクトレベルのリソース使用率メトリクスにアクセスするには

- 1. にサインイン AWS Management Console し、 AWS CodeBuild コンソールを <u>https://</u> console.aws.amazon.com/codesuite/codebuild/home://www.com で開きます。
- 2. ナビゲーションペインで、[Build projects] を選択します。
- ビルドプロジェクトのリストの [名前] 列で、使用率メトリクスを表示するプロジェクトを選択します。
- [Metrics] タブを選択します。リソース使用率のメトリクスは、[リソース使用率メトリクス] セクションに表示されます。
- 5. CloudWatch コンソールでプロジェクトレベルのリソース使用率のメトリクスを表示するに は、[リソース使用率メトリクス] セクションで [CloudWatch で表示] を選択します。

ビルドレベルのリソース使用率メトリクス

ビルドレベルのリソース使用率メトリクスにアクセスするには

- 1. にサインイン AWS Management Console し、 AWS CodeBuild コンソールを <u>https://</u> console.aws.amazon.com/codesuite/codebuild/home://www.com で開きます。
- 2. ナビゲーションペインで、[Build history]を選択します。
- 3. ビルドのリストでは、[ビルドの実行] 列で、使用率メトリクスを表示するビルドを選択します。
- 4. [リソース使用率] タブを選択します。
- 5. CloudWatch コンソールにビルドレベルのリソース使用率のメトリクスを表示するには、[リソー ス使用率メトリクス] セクションで [CloudWatch で表示] を選択します。

リソース使用率メトリクスへのアクセス(Amazon CloudWatch コンソール)

Amazon CloudWatch コンソールを使用して、CodeBuild リソース使用率メトリクスにアクセスでき ます。

プロジェクトレベルのリソース使用率メトリクス

プロジェクトレベルのリソース使用率メトリクスにアクセスするには

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

2. ナビゲーションペインで Metrics (メトリクス) を選択します。

3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。

Metrics		***				
Favorites	All metrics Graphed metrics Graph op	tions Source				
C Add a dashboard	Q Search for any metric, dimension or resource id					
	CodeBuild 44 Metrics	Events 12 Metrics	Lambda 14 Metrics			

- 4. [By Project (プロジェクト別)] を選択します。
- 5. グラフに追加するプロジェクトとメトリクスの組み合わせを1つ以上選択します。選択された すべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されます。
- (オプション) [グラフ化したメトリクス] タブからメトリクスとグラフをカスタマイズできます。
 例えば、[統計] 列のドロップダウンリストから、表示する別の統計を選択できます。または、
 [期間] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「<u>メトリクスのグラフ化</u>」および 「利用可能なメトリクスを表示する」を参照してください。

ビルドレベルのリソース使用率メトリクス

ビルドレベルのリソース使用率メトリクスにアクセスするには

- 1. にサインイン AWS Management Console し、「https://<u>https://console.aws.amazon.com/</u> cloudwatch/.com」で CloudWatch コンソールを開きます。
- 2. ナビゲーションペインで Metrics (メトリクス) を選択します。
- 3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。

Metrics						
Favorites	All metrics	Graphed metrics	Graph options	s Source		
O Add a dashboard						
	Q Search for	r any metric, dimension	or resource id			
	155 Metrics	6				
	CodeBuil	d		Events	Lambo	da
	44 Metrics			12 Metrics	14 Metr	rics

4. [BuildId、BuildNumber、ProjectName] を選択します。

- 5. グラフに追加するビルドとメトリクスの組み合わせを1つ以上選択します。選択されたすべて のビルドとメトリクスの組み合わせが、ページのグラフに表示されます。
- (オプション) [グラフ化したメトリクス] タブからメトリクスとグラフをカスタマイズできます。
 例えば、[統計] 列のドロップダウンリストから、表示する別の統計を選択できます。または、
 [期間] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「<u>メトリクスのグラフ化</u>」および 「利用可能なメトリクスを表示する」を参照してください。

CloudWatch アラームを使用して CodeBuild ビルドをモニタリング

ビルドの CloudWatch アラームを作成できます。アラームは、指定した期間にわたって1つのメト リクスをモニタリングし、複数期間にわたる指定しきい値との比較結果に基づいて1つ以上のアク ションを実行します。ネイティブ CloudWatch アラーム機能を使用して、しきい値を超えたときに CloudWatch にサポートされる任意のアクションを指定できます。たとえば、15 分以内にアカウン トで3つ以上のビルドが失敗したときに Amazon SNS 通知が送信されるように指定できます。

CodeBuild メトリクスの CloudWatch アラームを作成するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/cloudwatch/</u>:// www.com」で CloudWatch コンソールを開きます。
- 2. ナビゲーションペインで、[Alarms] (アラーム) を選択します。
- 3. [Create Alarm (アラーム作成)] を選択します。
- [CloudWatch Metrics by Category (カテゴリ別の CloudWatch メトリクス)] で、[CodeBuild Metrics (CodeBuild メトリクス)] を選択します。プロジェクトレベルのメトリクスのみでよいこ とがわかっている場合は、[By Project (プロジェクト別)] を選択します。アカウントレベルのメ トリクスのみでよいことがわかっている場合は、[Account Metrics (アカウントメトリクス)] を選 択します。
- 5. [Create Alarm (アラームの作成)] で、まだ選択されていない場合は [Select Metric (メトリクスの 選択)] を選択します。
- 6. アラームを作成する対象のメトリクスを選択します。オプションは [By Project (プロジェクト別)] または [Account Metrics (アカウントメトリクス)] です。
- [Next (次へ)] または [Define Alarm (アラームの定義)] を選択し、アラームを作成します。詳細 については、Amazon CloudWatchユーザーガイドの「<u>Amazon CloudWatchアラームの作成</u>]
 を参照してください。アラームがトリガーされたときのAmazon SNS 通知の設定の詳細につい

ては、「Amazon SNS デベロッパーガイド」の「<u>Amazon SNS 通知の設定</u>」を参照してくださ い。

8. アラームの作成(アラームの作成)を選択します。

のセキュリティ AWS CodeBuild

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する 組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できま す。

セキュリティとコンプライアンスは、 AWS とお客様の間の責任共有です。この共有モデルは、ホス トオペレーティングシステムと仮想化レイヤーからサービス施設の物理的なセキュリティまで、コン ポーネントを AWS 運用、管理、制御し、運用上の負担を軽減するのに役立ちます。お客様は、ゲス トオペレーティングシステム (更新やセキュリティパッチなど) とその他の関連アプリケーションソ フトウェアの管理責任を負います。また、 AWS 提供されたセキュリティグループファイアウォール の設定についても責任を負います。お客様の責任は、利用するサービス、お客様の IT 環境へのこれ らのサービスの統合、適用される法律および規制によって異なります。したがって、貴社で使用する サービスについて注意深く検討してください。詳細については、「<u>責任共有モデル</u>」を参照してくだ さい。

CodeBuild のリソースをセキュリティで保護する方法については、以下のトピックを参照してください。

トピック

- でのデータ保護 AWS CodeBuild
- での ID とアクセスの管理 AWS CodeBuild
- AWS CodeBuild のコンプライアンス検証
- の耐障害性 AWS CodeBuild
- インフラストラクチャセキュリティ in AWS CodeBuild
- CodeBuild でソースプロバイダにアクセスする
- サービス間での不分別な代理処理の防止

でのデータ保護 AWS CodeBuild

責任 AWS <u>共有モデル</u>、 でのデータ保護に適用されます AWS CodeBuild。このモデルで説明されて いるように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があり ます AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管 理を維持する責任があります。また、使用する「 AWS のサービス」のセキュリティ設定と管理タ スクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関 <u>するよくある質問</u>を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティ ブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。 この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。 また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用 します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検 証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様のEメールアドレスなどの極秘または機密情報を、タグ、または[名前]フィールドなどの自 由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、また は SDK を使用して CodeBuild AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請 求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサー バーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。

- CodeBuild プロジェクト環境変数のパラメータストアまたは buildspec の env/parameterstore セクションを使用して指定した文字列。詳細については、Amazon EC2 Systems Manager ユーザーガイドの「<u>Systems Manager パラメータストア</u>」および「<u>Systems Manager パラメータ</u> <u>ストアコンソールのチュートリアル</u>」を参照してください。
- CodeBuild プロジェクト環境変数または buildspec env/secrets-managerセクション AWS Secrets Manager で を使用して指定された文字列。詳細については、「<u>キー管理</u>」を参照してく ださい。

データ保護の詳細については、AWS セキュリティブログのブログ投稿「<u>AWS の責任共有モデルと</u> GDPR」を参照してください。

トピック

- データの暗号化
- キー管理
- トラフィックのプライバシー

データの暗号化

暗号化は CodeBuild セキュリティの重要な部分です。一部の暗号化 (伝送中のデータの暗号化など) はデフォルトで提供されるため、特に操作は不要です。その他の暗号化 (保管時のデータの暗号化な ど) については、プロジェクトまたはビルドの作成時に設定できます。

- 保管時のデータの暗号化 キャッシュ、ログ、エクスポートされた生のテストレポートデータファ イル、ビルド結果などのビルドアーティファクトは、デフォルトでを使用して暗号化されます AWS マネージドキー。これらの KMS キーを使用しない場合は、カスタマー管理キーを作成して 設定する必要があります。詳細については、AWS Key Management Service ユーザーガイドの 「KMS キーの作成」および「AWS Key Management Service の概念」を参照してください。
 - CodeBuild がビルド出力アーティファクトを暗号化するために使用する AWS KMS キーの識別 子をCODEBUILD_KMS_KEY_ID環境変数に保存できます。詳細については、ビルド環境の環境変 数を参照してください。
 - ビルドプロジェクトの作成時にカスタマー管理キーを指定できます。詳細については、「<u>Set</u> the Encryption Key Using the Console」および「<u>CLI を使用して暗号化キーを設定する</u>」を参照 してください。

ビルドフリートの Amazon Elastic Block Store ボリュームは、デフォルトで を使用して暗号化さ れます AWS マネージドキー。

- 転送時のデータの暗号化 カスタマーと CodeBuild とのすべての通信、および CodeBuild とその ダウンストリーム依存関係とのすべての通信は、署名バージョン 4 の署名プロセスで署名された TLS 接続を使用して保護されます。すべての CodeBuild エンドポイントは、 によって管理され る SHA-256 証明書を使用します AWS Private Certificate Authority。詳細については、「<u>署名バー</u> <u>ジョン 4 の署名プロセス</u>」および「<u>ACM PCA とは</u>」を参照してください。
- ビルドアーティファクトの暗号化 プロジェクトに関連付けられた CodeBuild サービスロールには、そのビルド出力アーティファクトを暗号化するために、KMS キーへのアクセス権が必要です。デフォルトでは、CodeBuild は AWS アカウントで Amazon S3 AWS マネージドキー の を使

用します。この AWS マネージドキーを使用しない場合は、カスタマー管理キーを作成して設定す る必要があります。詳細については、「<u>ビルド出力を暗号化</u>」および AWS KMS デベロッパーガ イドの「Creating Keys」を参照してください。

キー管理

暗号化によりコンテンツを不正使用から保護できます。暗号化キーを に保存し AWS Secrets Manager、ビルドプロジェクトに関連付けられた CodeBuild サービスロールに、Secrets Manager アカウントから暗号化キーを取得するアクセス許可を付与します。詳細については、「<u>カスタ</u> マーマネージドキーを使用してビルド出力を暗号化」、「<u>でビルドプロジェクトを作成する AWS</u> <u>CodeBuild」、AWS CodeBuild ビルドを手動で実行する</u>」、「<u>チュートリアル: シークレットの保存</u> と取得」を参照してください。

ビルドコマンドで CODEBUILD_KMS_KEY_ID環境変数を使用して、 AWS KMS キー識別子を取得し ます。詳細については、「ビルド環境の環境変数」を参照してください。

ランタイム環境用の Docker イメージを保存するプライベートレジストリへの認証情報を保護するに は、Secrets Manager を使用できます。詳細については、「<u>CodeBuild AWS Secrets Manager のサ</u> ンプルを含むプライベートレジストリ」を参照してください。

トラフィックのプライバシー

インターフェイス VPC エンドポイントを使用するように CodeBuild を設定することで、ビルドの セキュリティを強化できます。これを行う場合、インターネットゲートウェイ、NAT デバイス、 または仮想プライベートゲートウェイは必要ありません。また、PrivateLink の設定も必須ではあり ません (ただし、お勧めします)。詳細については、「<u>VPC エンドポイントの使用</u>」を参照してく ださい。PrivateLink および VPC エンドポイントの詳細については、「<u>AWS PrivateLink</u>」および 「PrivateLink を介した AWS のサービスへのアクセス」を参照してください。

での ID とアクセスの管理 AWS CodeBuild

へのアクセスには認証情報 AWS CodeBuild が必要です。これらの認証情報には、S3 バケットへ のビルドアーティファクトの保存と取得、ビルド用の Amazon CloudWatch Logs の表示など、 AWS リソースへのアクセス許可が必要です。以下のセクションでは、<u>AWS Identity and Access</u> <u>Management</u> (IAM) と CodeBuild を使用してリソースに安全にアクセスする方法について説明しま す。

AWS CodeBuild リソースへのアクセス許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースを作成またはアクセスする ためのアクセス許可はアクセス許可ポリシーによって管理されます。アカウント管理者は、IAM ア イデンティティ (ユーザー、グループ、ロール) にアクセス許可ポリシーをアタッチできます。

Note

アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、IAM ユーザーガイドの[IAM のベストプラクティス]を参照してください。

アクセス許可を付与するときは、アクセス許可を取得するユーザー、アクセスできるリソース、およ びそれらのリソースに対して実行できるアクションを決定します。

トピック

- AWS CodeBuild リソースとオペレーション
- ・ リソース所有権についての理解
- リソースへのアクセスの管理
- ポリシー要素 (アクション、効果、プリンシパル)の指定

AWS CodeBuild リソースとオペレーション

では AWS CodeBuild、プライマリリソースはビルドプロジェクトです。ポリシーで Amazon リ ソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。ビルドもリソース で、ARN が関連付けられています。詳細については、<u>『』の「Amazon リソースネーム (ARN) と</u> AWS サービス名前空間」を参照してくださいAmazon Web Services 全般のリファレンス。

リソースタイプ	ARN 形式		
ビルドプロジェクト	arn:aws:codebuild: <project-name< pre=""></project-name<>	<pre>region-ID :account-ID :project/</pre>	
Build	arn:aws:codebuild: <i>D</i> :build/ <i>build-ID</i>	region-ID :account-I	

リソースタイプ	ARN 形式
レポートグループ	<pre>arn:aws:codebuild: region-ID :account-ID :report-g roup/ report-group-name</pre>
レポート	arn:aws:codebuild: <i>region-ID</i> : <i>account-I</i> <i>D</i> :report/ <i>report-ID</i>
フリート	arn:aws:codebuild: <i>region-ID</i> :account-I D :fleet/fleet-ID
すべての CodeBuild リ ソース	arn:aws:codebuild:*
指定された AWS リー ジョン内の指定された アカウントが所有する すべての CodeBuild リ ソース	arn:aws:codebuild: <i>region-ID</i> :account-ID :*

▲ Important

リザーブドキャパシティ機能を使用すると、ソースファイル、Docker レイヤー、buildspec で指定されキャッシュされたディレクトリなどを含む、フリートインスタンスにキャッシュ されたデータに、同じアカウント内の他のプロジェクトからアクセスできます。これは設計 によるもので、同じアカウント内のプロジェクトがフリートインスタンスを共有できるよう にしています。

(i) Note

ほとんどの AWS サービスは、ARN でARNs。ただし、CodeBuild では、リソースパターン とルールで完全一致が使用されます。イベントパターンの作成時に正しい文字を使用して、 リソース内の ARN 構文とそれらの文字が一致する必要があります。 たとえば、以下のように ARN を使用して、ステートメント内で特定のビルドプロジェクト (*myBuildProject*) を指定できます。

"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"

すべてのリソースを指定する場合、または API アクションが ARN をサポートしていない場合は、以 下の要領で、Resource エレメント内でワイルドカード文字 (*) を使用します。

"Resource": "*"

ー部の CodeBuild API アクションは複数のリソースを受け入れます (例: BatchGetProjects)。単 ーのステートメントに複数のリソースを指定するには、以下のようにコンマで ARN を区切ります。

```
"Resource": [
    "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",
    "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"
]
```

CodeBuild には、CodeBuild リソースを操作するための一連のオペレーションが用意されています。 リストについては、「AWS CodeBuild アクセス許可リファレンス」を参照してください。

リソース所有権についての理解

AWS アカウントは、リソースを作成したユーザーに関係なく、アカウントで作成されたリソースを 所有します。具体的には、リソース所有者は、リソース作成リクエストを認証する<u>プリンシパルエン</u> <u>ティティ</u> (ルートアカウント、 ユーザー、または IAM ロール) の AWS アカウントです。次の例は、 この仕組みを示しています。

- AWS アカウントのルートアカウントの認証情報を使用してルールを作成する場合、 AWS アカウ ントは CodeBuild リソースの所有者です。
- AWS アカウントに ユーザーを作成し、そのユーザーに CodeBuild リソースを作成するアクセス許可を付与すると、そのユーザーは CodeBuild リソースを作成できます。ただし、ユーザーが属する AWS アカウントは CodeBuild リソースを所有します。
- CodeBuild リソースを作成するアクセス許可を持つ IAM ロールを AWS アカウントに作成する場合、ロールを引き受けることのできるすべてのユーザーが CodeBuild リソースを作成できます。
 ロールが属する AWS アカウントが CodeBuild リソースを所有しています。

リソースへのアクセスの管理

許可ポリシーでは、誰がどのリソースにアクセスできるかを記述します。

Note

このセクションでは、AWS CodeBuildでの IAM の使用について説明します。ここで は、IAM サービスに関する詳細情報を提供しません。完全な IAM ドキュメンテーションにつ いては、「IAM ユーザーガイド」の「<u>IAM とは</u>」を参照してください。IAM ポリシー構文の 詳細と説明については、IAM ユーザーガイドの <u>AWS IAM ポリシーの参照</u>を参照してくださ い。

IAM アイデンティティにアタッチされているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースに添付されたポリシーは、リソースベースのポリシーと呼ばれま す。CodeBuild は、アイデンティティベースのポリシーと、アカウント間のリソース共有を目的とし た、特定の読み取り専用 API のリソースベースのポリシーをサポートしています。

S3 バケットへの安全なアクセス

CodeBuild プロジェクトに関連付けられている S3 バケットが本人または本人が信頼するユーザーに よって所有されていることを確認するために、次のアクセス許可を IAM ロールに含めることを強く お勧めします。これらのアクセス許可は、 AWS マネージドポリシーとロールには含まれません。自 分で追加する必要があります。

s3:GetBucketAcl

s3:GetBucketLocation

プロジェクトで使用している S3 バケットの所有者が変更された場合は、自分を本来のバケット所有 者にして IAM ロールのアクセス許可を更新する必要があります (まだ更新していない場合)。詳細に ついては、「<u>ユーザーに CodeBuild とのやり取りを許可</u>」および「<u>CodeBuild が他の AWS のサービ</u> スとやり取りすることを許可」を参照してください。

ポリシー要素 (アクション、効果、プリンシパル)の指定

サービスは、 AWS CodeBuild リソースごとに一連の API オペレーションを定義します。これらの API オペレーションを実行するためのアクセス許可を付与するために、CodeBuild ではポリシーに一 連のアクションを定義できます。一部の API オペレーションは、API オペレーションを実行するた めに複数のアクションに対するアクセス許可を要求できます。詳細については、「<u>AWS CodeBuild</u> <u>リソースとオペレーション</u>」および「<u>AWS CodeBuild アクセス許可リファレンス</u>」を参照してくだ さい。

以下は、基本的なポリシーの要素です。

- ・ リソース Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。
- アクション アクションのキーワードを使用して、許可または拒否するリソースオペレーション を識別します。たとえば、codebuild:CreateProject 許可は、CreateProject オペレー ションを実行する許可をユーザーに与えます。
- 効果 ユーザーがアクションをリクエストする際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。これは、別のポリシーがアクセスを許可している場合でも、ユーザーがリソースにアクセスできないようにするために行うことができます。
- プリンシパル アイデンティティベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

IAM ポリシーの構文と記述の詳細については、「IAM ユーザーガイド」の「<u>AWS IAM ポリシーリ</u> ファレンス」を参照してください。

すべての CodeBuild API アクションとそれらが適用されるリソースの表については、「<u>AWS</u> CodeBuild アクセス許可リファレンス」を参照してください。

でのアイデンティティベースのポリシーの使用 AWS CodeBuild

このトピックでは、アカウント管理者が IAM ID (ユーザー、グループ、ロール) にアクセス権限ポリ シーをアタッチし、それによって AWS CodeBuild リソースでオペレーションを実行するアクセス権 限を付与する方法を示すアイデンティティベースのポリシーの例を示します。

A Important

初めに、CodeBuild リソースへのアクセスを管理するための基本概念と、使用可能なオプ ションについて説明する概要トピックをお読みになることをお勧めします。詳細について は、「AWS CodeBuild リソースへのアクセス許可の管理の概要」を参照してください。 トピック

- AWS CodeBuild コンソールの使用に必要な許可
- が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアクセス許可
- AWS CodeBuild コンソールがソースプロバイダーに接続するために必要なアクセス許可
- ・ AWS の 管理 (事前定義) ポリシー AWS CodeBuild
- CodeBuild の管理ポリシーと通知
- AWS マネージドポリシーに対する CodeBuild の更新
- カスタマー管理ポリシーの例

us-east-2 リージョンにあり、123456789012 のアカウントで、名前が my で始まるビルドプロ ジェクトについての情報のみをユーザーが取得するのを許可するアクセス許可ポリシーの例を次に示 します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:BatchGetProjects",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

AWS CodeBuild コンソールの使用に必要な許可

AWS CodeBuild コンソールを使用するユーザーには、 AWS アカウントの他の AWS リソースを記 述できる最小限のアクセス許可セットが必要です。次のサービスからのアクセス許可を持っている必 要があります。

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit (ソースコードを AWS CodeCommit リポジトリに保存している場合)
- Amazon Elastic Container Registry (Amazon ECR) (Amazon ECR リポジトリの Docker イメージ に依存するビルド環境を使用している場合)

Note

2022 年 7 月 26 日に、デフォルトの IAM ポリシーが更新されました。詳細については、 「<u>が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアクセ</u> <u>ス許可</u>」を参照してください。

- Amazon Elastic Container Service (Amazon ECS) (Amazon ECR リポジトリの Docker イメージに 依存するビルド環境を使用している場合)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

これらの最小限必要なアクセス許可よりも制限された IAM ポリシーを作成している場合、コンソー ルは意図したとおりには機能しません。

が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアク セス許可

2022 年 7 月 26 日現在、 AWS CodeBuild は Amazon ECR アクセス許可のデフォルトの IAM ポリ シーを更新しました。次のアクセス許可がデフォルトポリシーから削除されました。

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

2022 年 7 月 26 日より前に作成された CodeBuild プロジェクトについては、次の Amazon ECR ポ リシーでポリシーを更新することをお勧めします。

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

ポリシーの更新の詳細については、「<u>ユーザーに CodeBuild とのやり取りを許可</u>」を参照してくだ さい。 AWS CodeBuild コンソールがソースプロバイダーに接続するために必要なアクセス許可

AWS CodeBuild コンソールでは、次の API アクションを使用してソースプロバイダー (GitHub リポ ジトリなど) に接続します。

- codebuild:ListConnectedOAuthAccounts
- codebuild:ListRepositories
- codebuild:PersistOAuthToken
- codebuild:ImportSourceCredentials

AWS CodeBuild コンソールを使用して、ソースプロバイダー (GitHub リポジトリなど) をビルドプ ロジェクトに関連付けることができます。これを行うには、まず AWS CodeBuild 、コンソールへの アクセスに使用するユーザーに関連付けられた IAM アクセスポリシーに前述の API アクションを追 加する必要があります。

ListConnected0AuthAccounts、ListRepositories、および Persist0AuthToken の API アクションは、コードで呼び出すことを想定していません。したがって、これらの API アクション は AWS CLI および AWS SDKs に含まれません。

AWS の 管理 (事前定義) ポリシー AWS CodeBuild

AWS は、によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多 くの一般的なユースケースに対処します AWS。これらの AWS 管理ポリシーは、一般的なユー スケースに必要なアクセス許可を付与するため、必要なアクセス許可を調査する必要がなくな ります。CodeBuild の マネージドポリシーは、問題のポリシーが付与されたユーザーの責任 に応じて、IAM、 AWS CodeCommit、Amazon EC2、Amazon ECR、Amazon SNS、Amazon CloudWatch Events などの他のサービスでオペレーションを実行するアクセス許可も提供します。 たとえば、AWSCodeBuildAdminAccess ポリシーは管理レベルのユーザーポリシーであり、こ のポリシーが適用されるユーザーは、プロジェクトのビルドに関する CloudWatch Events ルー ルと、プロジェクト関連イベントに関する通知の Amazon SNS トピック (名前にプレフィックス arn:aws:codebuild: が付いているトピック) を作成および管理でき、また、CodeBuild でプロ ジェクトとレポートグループを管理できます。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。

アカウントのユーザーにアタッチできる次の AWS 管理ポリシーは、 に固有です AWS CodeBuild。

AWSCodeBuildAdminAccess

CodeBuild ビルドプロジェクトを管理するためのアクセス許可を含む CodeBuild へのフルアクセ スを提供します。

AWSCodeBuildDeveloperAccess

CodeBuild へのアクセスを提供しますが、ビルドプロジェクトの管理は許可しません。

AWSCodeBuildReadOnlyAccess

CodeBuild への読み取り専用アクセスを許可します。

CodeBuild が作成するビルド出力アーティファクトにアクセスするには、

「AmazonS3ReadOn1yAccess」という名前の AWS 管理ポリシーもアタッチする必要がありま す。

CodeBuild サービスロールを作成および管理するには、 という名前 AWS の管理ポリシーもアタッチ する必要がありますIAMFullAccess。

独自のカスタム IAM ポリシーを作成して、CodeBuild アクションとリソースのための権限を許可す ることもできます。こうしたカスタムポリシーは、該当するアクセス許可が必要なユーザーまたはグ ループにアタッチできます。

トピック

{

- AWSCodeBuildAdminAccess
- AWSCodeBuildDeveloperAccess
- AWSCodeBuildReadOnlyAccess

AWSCodeBuildAdminAccess

「AWSCodeBuildAdminAccess」ポリシーは、CodeBuild へのフルアクセスを許可します。たとえ ば、ビルドプロジェクトを管理するアクセス許可を付与します。このポリシーは、管理者レベルの ユーザーにのみ適用し、プロジェクトやレポートグループを削除する機能など、 AWS アカウント内 の CodeBuild プロジェクト、レポートグループ、および関連リソースを完全に制御できるようにし ます。

AWSCodeBuildAdminAccess ポリシーには、次のポリシーステートメントが含まれています。

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

```
"Statement": [
 {
    "Sid": "AWSServicesAccess",
    "Action": [
      "codebuild:*",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:GetRepository",
      "codecommit:ListBranches",
      "codecommit:ListRepositories",
      "cloudwatch:GetMetricStatistics",
      "ec2:DescribeVpcs",
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSubnets",
      "ecr:DescribeRepositories",
      "ecr:ListImages",
      "elasticfilesystem:DescribeFileSystems",
      "events:DeleteRule",
      "events:DescribeRule",
      "events:DisableRule",
      "events:EnableRule",
      "events:ListTargetsByRule",
      "events:ListRuleNamesByTarget",
      "events:PutRule",
      "events:PutTargets",
      "events:RemoveTargets",
      "logs:GetLogEvents",
      "s3:GetBucketLocation",
      "s3:ListAllMyBuckets"
    ],
    "Effect": "Allow",
    "Resource": "*"
 },
  {
    "Sid": "CWLDeleteLogGroupAccess",
    "Action": [
      "logs:DeleteLogGroup"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:logs:*:*:log-group:/aws/codebuild/*:log-stream:*"
 },
  {
    "Sid": "SSMParameterWriteAccess",
    "Effect": "Allow",
```

```
"Action": [
    "ssm:PutParameter"
  ],
  "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
},
{
  "Sid": "SSMStartSessionAccess",
  "Effect": "Allow",
  "Action": [
    "ssm:StartSession"
  ],
  "Resource": "arn:aws:ecs:*:*:task/*/*"
},
{
  "Sid": "CodeStarConnectionsReadWriteAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-connections:CreateConnection",
    "codestar-connections:DeleteConnection",
    "codestar-connections:UpdateConnectionInstallation",
    "codestar-connections:TagResource",
    "codestar-connections:UntagResource",
    "codestar-connections:ListConnections",
    "codestar-connections:ListInstallationTargets",
    "codestar-connections:ListTagsForResource",
    "codestar-connections:GetConnection",
    "codestar-connections:GetIndividualAccessToken",
    "codestar-connections:GetInstallationUrl",
    "codestar-connections:PassConnection",
    "codestar-connections:StartOAuthHandshake",
    "codestar-connections:UseConnection"
  ],
  "Resource": [
    "arn:aws:codestar-connections:*:*:connection/*",
    "arn:aws:codeconnections:*:*:connection/*"
  1
},
{
  "Sid": "CodeStarNotificationsReadWriteAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:CreateNotificationRule",
    "codestar-notifications:DescribeNotificationRule",
    "codestar-notifications:UpdateNotificationRule",
```

```
"codestar-notifications:DeleteNotificationRule",
       "codestar-notifications:Subscribe",
       "codestar-notifications:Unsubscribe"
     ],
     "Resource": "*",
     "Condition": {
       "ArnLike": {
         "codestar-notifications:NotificationsForResource":
"arn:aws:codebuild:*:*:project/*"
       }
     }
  },
   {
     "Sid": "CodeStarNotificationsListAccess",
     "Effect": "Allow",
     "Action": [
       "codestar-notifications:ListNotificationRules",
       "codestar-notifications:ListEventTypes",
       "codestar-notifications:ListTargets",
       "codestar-notifications:ListTagsforResource"
     ],
     "Resource": "*"
  },
   {
     "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
     "Effect": "Allow",
     "Action": [
       "sns:CreateTopic",
      "sns:SetTopicAttributes"
     ],
    "Resource": "arn:aws:sns:*:*:codestar-notifications*"
  },
   {
     "Sid": "SNSTopicListAccess",
     "Effect": "Allow",
     "Action": [
       "sns:ListTopics",
      "sns:GetTopicAttributes"
     ],
     "Resource": "*"
  },
   {
     "Sid": "CodeStarNotificationsChatbotAccess",
     "Effect": "Allow",
```

```
"Action": [
    "chatbot:DescribeSlackChannelConfigurations",
    "chatbot:ListMicrosoftTeamsChannelConfigurations"
    ],
    "Resource": "*"
    }
]
```

AWSCodeBuildDeveloperAccess

AWSCodeBuildDeveloperAccess ポリシーの CodeBuild のすべての機能へのアクセス、プロジェ クトおよびレポートグループの関連リソースへのアクセスを許可します。このポリシーでは、ユー ザーが CodeBuild プロジェクトやレポートグループ、または CloudWatch Events などの他の AWS サービスの関連リソースを削除することはできません。ほとんどのユーザーにこのポリシーを適用す ることをお勧めします。

AWSCodeBuildDeveloperAccess ポリシーには、次のポリシーステートメントが含まれています。

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:StartBuildBatch",
        "codebuild:StopBuildBatch",
        "codebuild:RetryBuild",
        "codebuild:RetryBuildBatch",
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
        "codebuild:List*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
```

```
"events:ListRuleNamesByTarget",
    "logs:GetLogEvents",
    "s3:GetBucketLocation",
    "s3:ListAllMyBuckets"
  ],
  "Effect": "Allow",
  "Resource": "*"
},
{
  "Sid": "SSMParameterWriteAccess",
  "Effect": "Allow",
  "Action": [
    "ssm:PutParameter"
  ],
  "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
},
{
  "Sid": "SSMStartSessionAccess",
  "Effect": "Allow",
  "Action": [
    "ssm:StartSession"
  ],
  "Resource": "arn:aws:ecs:*:*:task/*/*"
},
{
  "Sid": "CodeStarConnectionsUserAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-connections:ListConnections",
    "codestar-connections:GetConnection"
  ],
  "Resource": [
    "arn:aws:codestar-connections:*:*:connection/*",
    "arn:aws:codeconnections:*:*:connection/*"
  ]
},
{
  "Sid": "CodeStarNotificationsReadWriteAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:CreateNotificationRule",
    "codestar-notifications:DescribeNotificationRule",
    "codestar-notifications:UpdateNotificationRule",
    "codestar-notifications:Subscribe",
```

```
"codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
 "arn:aws:codebuild:*:*:project/*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SNSTopicListAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsChatbotAccess",
      "Effect": "Allow",
      "Action": [
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```
AWSCodeBuildReadOnlyAccess

このAWSCodeBuildReadOnlyAccessポリシーは、CodeBuild および他の AWS サービスの関連リ ソースへの読み取り専用アクセスを許可します。ビルドの表示と実行、プロジェクトの表示、レポー トグループの表示はできるが、それらの変更はできないユーザーにこのポリシーを適用します。

AWSCodeBuildReadOnlyAccess ポリシーには、次のポリシーステートメントが含まれています。

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:List*",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarConnectionsUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:ListConnections",
        "codestar-connections:GetConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
```

```
"Sid": "CodeStarNotificationsPowerUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
 "arn:aws:codebuild:*:*:project/*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

CodeBuild の管理ポリシーと通知

CodeBuild は、ビルドプロジェクトへの重要な変更をユーザーに知らせることができる通知機能をサ ポートしています。CodeBuild の管理ポリシーには、通知機能のポリシーステートメントが含まれま す。詳細については、通知とは を参照してください。

読み取り専用マネージドポリシーの通知に関連するアクセス許可

AWSCodeBuildReadOnlyAccess 管理ポリシーには、通知への読み取り専用アクセスを許可する以下のステートメントが含まれています。この管理ポリシーが適用されたユーザーは、リソースの通知 を表示することはできますが、リソースの作成や管理、リソースへのサブスクライブを行うことはで きません。

"Sid": "CodeStarNotificationsPowerUserAccess", "Effect": "Allow",

{

```
"Action": [
           "codestar-notifications:DescribeNotificationRule"
       ٦,
       "Resource": "*",
       "Condition" : {
           "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*:*:project/*"}
       }
   },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListEventTypes",
           "codestar-notifications:ListTargets"
       ],
       "Resource": "*"
   }
```

その他の管理ポリシーの通知に関連するアクセス許可

AWSCodeBuildDeveloperAccess 管理ポリシーには、ユーザーが通知を作成、編集、サブスクラ イブできるようにする次のステートメントが含まれています。ユーザーは通知ルールを削除したり、 リソースのタグを管理したりすることはできません。

```
{
       "Sid": "CodeStarNotificationsReadWriteAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:CreateNotificationRule",
           "codestar-notifications:DescribeNotificationRule",
           "codestar-notifications:UpdateNotificationRule",
           "codestar-notifications:Subscribe",
           "codestar-notifications:Unsubscribe"
       ],
       "Resource": "*",
       "Condition" : {
           "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*:*:project/*"}
       }
  },
   {
```

```
"Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource",
        "codestar-notifications:ListEventTypes"
    ],
    "Resource": "*"
},
{
    "Sid": "SNSTopicListAccess",
    "Effect": "Allow",
    "Action": [
        "sns:ListTopics"
    ],
    "Resource": "*"
},
{
    "Sid": "CodeStarNotificationsChatbotAccess",
    "Effect": "Allow",
    "Action": [
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      1,
   "Resource": "*"
}
```

IAM と通知の詳細については、「<u>AWS CodeStar Notifications の Identity and Access Management</u>」 を参照してください。

AWS マネージドポリシーに対する CodeBuild の更新

このサービスがこれらの変更の追跡を開始してからの CodeBuild の AWS マネージドポリシーの更新 に関する詳細を表示します。このページの変更に関する自動通知については、<u>AWS CodeBuild ユー</u> <u>ザーガイドのドキュメント履歴</u>の RSS フィードを購読してください。

変更	説明	日付
AWSCodeBuildAdminA ccess 、AWSCodeBu ildDeveloperAccess および AWSCodeBu ildReadOnlyAccess - 既 存のポリシーに対する更新	CodeBuild はリソースをこれ らのポリシーに更新しまし た。 AWSCodeBuildAdminA ccess、AWSCodeBu ildDeveloperAccess 、および AWSCodeBu ildReadOnlyAccess ポ リシーが変更され、既存の リソースが更新されました。 元のリソースarn:aws:c odebuild:* がに更 新されましたarn:aws:c odebuild:*:*:proje ct/* 。	2024年11月15日
AWSCodeBuildAdminA ccess 、AWSCodeBu ildDeveloperAccess および AWSCodeBu ildReadOnlyAccess - 既 存のポリシーに対する更新	CodeBuild は、ブランド AWS CodeConnections 変更をサ ポートするために、これらの ポリシーにリソースを追加し ました。 AWSCodeBuildAdminA ccess 、AWSCodeBu ildDeveloperAccess 、および AWSCodeBu ildReadOnlyAccess ポ リシーが変更され、リソー ス arn:aws:codeconnec tions:*:*:connecti on/* が追加されました。	2024年4月18日
AWSCodeBuildAdminA ccess と AWSCodeBu	CodeBuild は、チャットアプ リケーションで Amazon Q	2023 年 5 月 16 日

AWS CodeBuild

変更	説明	日付
ildDeveloperAccess - 既存のポリシーに対する更新	Developer を使用する追加の 通知タイプをサポートするた めに、これらのポリシーにア クセス許可を追加しました。 AWSCodeBuildAdminA ccess および AWSCodeBu ildDeveloperAccess ポリシーが変更され、 アクセス許可 chatbot:L istMicrosoftTeamsC hannelConfiguratio ns が追加されました。	
CodeBuild が変更の追跡を開 始しました	CodeBuild は、 AWS 管理ポ リシーの変更の追跡を開始し ました。	2021 年 5 月 16 日

カスタマー管理ポリシーの例

このセクションでは、 AWS CodeBuild アクションのアクセス許可を付与するユーザーポリシー例を 示しています。これらのポリシーは、CodeBuild API、 AWS SDKs AWS CLI。コンソールを使用す る場合は、コンソールに固有の追加のアクセス許可を付与する必要があります。詳細については、 「AWS CodeBuild コンソールの使用に必要な許可」を参照してください。

以下のサンプル IAM ポリシーを使用して、ユーザーとロールに対して CodeBuild へのアクセスを制限できます。

トピック

- ビルドプロジェクトに関する情報の取得をユーザーに許可する
- フリートに関する情報の取得をユーザーに許可
- レポートグループに関する情報の取得をユーザーに許可する
- レポートに関する情報の取得をユーザーに許可する
- ビルドプロジェクトの作成をユーザーに許可する
- フリートの作成をユーザーに許可

- ・ レポートグループの作成をユーザーに許可する
- フリートの削除をユーザーに許可
- レポートグループの削除をユーザーに許可する
- レポートの削除をユーザーに許可する
- ビルドプロジェクトの削除をユーザーに許可する
- ビルドプロジェクト名の一覧表示をユーザーに許可する
- ビルドプロジェクトに関する情報の変更をユーザーに許可する
- フリートの変更をユーザーに許可
- ・レポートグループの変更をユーザーに許可する
- ビルドに関する情報の取得をユーザーに許可する
- ビルドプロジェクトのビルド ID の一覧表示をユーザーに許可する
- ビルド ID の一覧表示をユーザーに許可する
- フリートのリスト取得をユーザーに許可
- レポートグループの一覧表示をユーザーに許可する
- レポートの一覧表示をユーザーに許可する
- レポートグループのレポートの一覧表示をユーザーに許可する
- レポートのテストケースの一覧表示をユーザーに許可する
- ・ ビルドの実行開始をユーザーに許可する
- ・ ビルドの停止試行をユーザーに許可する
- ビルドの削除試行をユーザーに許可する
- CodeBuild が管理する Docker イメージに関する情報の取得をユーザーに許可する
- フリートサービスロールのアクセス許可ポリシーを追加することをユーザーに許可
- <u>VPC ネットワークインターフェイスの作成に必要な AWS サービスへの CodeBuild アクセスを許</u> 可する
- ・ <u>拒否ステートメントを使用して、がソースプロバイダーから切断 AWS CodeBuild されないように</u>
 する

ビルドプロジェクトに関する情報の取得をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウ ントで、名前が my で始まるビルドプロジェクトについての情報をユーザーが取得するのを許可しま す。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:BatchGetProjects",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

フリートに関する情報の取得をユーザーに許可

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンのフ リートに関する情報をユーザーが取得できるようにします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:BatchGetFleets",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
        }
    ]
}
```

レポートグループに関する情報の取得をユーザーに許可する

次のポリシーステートメント例では、us-east-2 リージョンにある 123456789012 アカウントの レポートグループについての情報をユーザーが取得するのを許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
        "Effect": "Allow",
        "Action": "codebuild:BatchGetReportGroups",
        "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
]
```

}

レポートに関する情報の取得をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンのレ ポートに関する情報をユーザーが取得できるようにします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:BatchGetReports",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

ビルドプロジェクトの作成をユーザーに許可する

以下のポリシーステートメントの例では、名前は問いませんが、us-east-2 リージョンだけにある 123456789012 アカウントで、特定の CodeBuild サービスロールのみを使用したビルドプロジェク トの作成をユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:CreateProject",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
        }
    ]
}
```

次のポリシーステートメントの例では、任意の名前でビルドプロジェクトを作成することをユーザー に許可しています。ただし、123456789012 アカウントで us-east-2 リージョンに限り、指定さ れた CodeBuild サービスロールのみを使用して作成する必要があります。また、ユーザーは指定さ れたサービスロールを でのみ使用でき AWS CodeBuild 、他の AWS サービスでは使用できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole",
      "Condition": {
          "StringEquals": {"iam:PassedToService": "codebuild.amazonaws.com"}
      }
    }
  ]
}}
```

フリートの作成をユーザーに許可

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでフ リートを作成することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:CreateFleet",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
        }
    ]
}
```

レポートグループの作成をユーザーに許可する

次のポリシーステートメントの例では、123456789012 アカウントの us-east-2 リージョンにレ ポートグループを作成することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:CreateReportGroup",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

フリートの削除をユーザーに許可

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでフ リートを削除することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:DeleteFleet",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
        }
    ]
}
```

レポートグループの削除をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンからレ ポートグループを削除することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:DeleteReportGroup",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

レポートの削除をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンからレ ポートを削除することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:DeleteReport",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

ビルドプロジェクトの削除をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウ ントで、名前が my で始まるビルドプロジェクトをユーザーが削除するのを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:DeleteProject",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

ビルドプロジェクト名の一覧表示をユーザーに許可する

以下のポリシーステートメントの例では、同じアカウントのビルドプロジェクト名のリストをユー ザーが取得するのを許可します。

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

```
"Action": "codebuild:ListProjects",
    "Resource": "*"
}
]
}
```

ビルドプロジェクトに関する情報の変更をユーザーに許可する

次のポリシーステートメントの例では、名前は問いませんが、us-east-2 リージョンだけにある 123456789012 アカウントで、特定の AWS CodeBuild のサービスロールのみを使用したビルドプ ロジェクトに関する情報をユーザーが変更するのを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": "codebuild:UpdateProject",
          "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
        },
        {
          "Effect": "Allow",
          "Action": "iam:PassRole",
          "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
        }
    ]
}
```

フリートの変更をユーザーに許可

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでフ リートを変更することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:UpdateFleet",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
        }
    ]
```

}

レポートグループの変更をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでレ ポートグループを変更することをユーザーに許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
        "Effect": "Allow",
        "Action": "codebuild:UpdateReportGroup",
        "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
]
}
```

ビルドに関する情報の取得をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウ ントで、my-build-project および my-other-build-project という名前のビルドプロジェク トについての情報をユーザーが取得するのを許可します。

ビルドプロジェクトのビルド ID の一覧表示をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウ ントで、my-build-project および my-other-build-project という名前のビルドプロジェク トのビルド ID リストをユーザーが取得するのを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": "codebuild:ListBuildsForProject",
        "Resource": [
            "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
            "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
        ]
     }
]
```

ビルド ID の一覧表示をユーザーに許可する

以下のポリシーステートメントの例では、同じアカウントのすべてのビルド ID のリストをユーザー が取得するのを許可します。

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

フリートのリスト取得をユーザーに許可

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 のリージョンでフ リートのリストを取得することをユーザーに許可します。

{

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

レポートグループの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウント us-east-2 のリージョンでレ ポートグループリストを取得することをユーザーに許可します。

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

レポートの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 のリージョンでレ ポートリストを取得することをユーザーに許可します。

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

レポートグループのレポートの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 のリージョンでレ ポートグループのレポートリストを取得することをユーザーに許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
        "Effect": "Allow",
        "Action": "codebuild:ListReportsForReportGroup",
        "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
]
```

レポートのテストケースの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでレ ポートのテストケースのリストを取得することをユーザーに許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
        "Effect": "Allow",
        "Action": "codebuild:DescribeTestCases",
        "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
]
}
```

ビルドの実行開始をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンの 123456789012 というアカウン トの、my という名前で始まるビルドプロジェクトのビルドをユーザーが実行する許可を与えます。

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

```
"Action": "codebuild:StartBuild",
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
}
]
}
```

ビルドの停止試行をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウ ントで、名前が my で始まるビルドの停止を試みるのをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:StopBuild",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

ビルドの削除試行をユーザーに許可する

次のポリシーステートメントの例では、123456789012 アカウントで us-east-2 リージョンに限 り、名前が my で始まるビルドプロジェクトでのビルドの削除試行をユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:BatchDeleteBuilds",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

CodeBuild が管理する Docker イメージに関する情報の取得をユーザーに許可する

次のポリシーステートメントの例では、CodeBuild で管理するすべての Docker イメージに関する情 報を取得することをユーザーに許可します。

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

フリートサービスロールのアクセス許可ポリシーを追加することをユーザーに許可

次のリソースポリシーステートメントの例では、フリートサービスロールの VPC アクセス許可ポリ シーを追加することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVpcCreateNI",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
        "arn:aws:ec2:region:account-id:security-group/security-group-id-1",
        "arn:aws:ec2:region:account-id:network-interface/*"
      1
    },
    {
      "Sid": "CodeBuildFleetVpcPermission",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:DeleteNetworkInterface"
      ],
```

```
"Resource": "*"
    },
    {
      "Sid": "CodeBuildFleetVpcNIPermission",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission"
      ],
      "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:Subnet": [
            "arn:aws:ec2:region:account-id:subnet/subnet-id-1"
          ]
        }
      }
    }
  ]
}
```

次のリソースポリシーステートメントの例では、フリートサービスロールにカスタム Amazon マシ ンイメージ (AMI) アクセス許可ポリシーを追加することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeImages",
            "Resource": "*"
        }
    ]
}
```

次の信頼ポリシーステートメントの例では、フリートサービスロールのアクセス許可ポリシーを追加 することをユーザーに許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildFleetVPCTrustPolicy",
```

```
"Effect": "Allow",
   "Principal": {
        "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "account-id"
        }
    }
    }
]
```

VPC ネットワークインターフェイスの作成に必要な AWS サービスへの CodeBuild アクセスを許可 する

次のポリシーステートメントの例では、2 つのサブネットを持つ VPC にネットワークインターフェ イスを作成する AWS CodeBuild アクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission"
      ],
      "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
      "Condition": {
```

```
"StringEquals": {
    "ec2:AuthorizedService": "codebuild.amazonaws.com"
    },
    "ArnEquals": {
        "ec2:Subnet": [
            "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
            "arn:aws:ec2:region:account-id:subnet/subnet-id-2"
        ]
        }
    }
    }
}
```

拒否ステートメントを使用して、 がソースプロバイダーから切断 AWS CodeBuild されないようにす る

以下のポリシーステートメントの例では、拒否ステートメントを使用して AWS CodeBuild によるソースプロバイダーの切断を防ぎます。ソースプロバイダーと接続するに は、codebuild:Persist0AuthToken および codebuild:ImportSourceCredentialsの逆 である codebuild:Delete0AuthToken を使用します。詳細については、「<u>AWS CodeBuild コン</u> ソールがソースプロバイダーに接続するために必要なアクセス許可」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "codebuild:Delete0AuthToken",
            "Resource": "*"
        }
    ]
}
```

AWS CodeBuild アクセス許可リファレンス

AWS CodeBuild ポリシーで AWS全体の条件キーを使用して、条件を表現できます。リストについて は、IAM ユーザーガイドの「<u>利用可能なキー</u>」を参照してください。

アクションは、ポリシーの Action フィールドで指定します。アクションを指定するに は、API オペレーション名 (例えば、codebuild: や codebuild:CreateProject) の前に codebuild:StartBuild プレフィックスを使用します。単一のステートメ ントに複数のアクションを指定するには、コンマで区切ります (例えば、"Action": ["codebuild:CreateProject", "codebuild:StartBuild"])。

ワイルドカード文字の使用

ポリシーの Resource フィールドでリソース値として、ワイルドカード文字 (*) を使用して、また は使用せずに ARN を指定します。ワイルドカードを使用して複数のアクションまたはリソースを 指定することができます。たとえば、codebuild:* は、すべての CodeBuild アクションを指定 し、codebuild:Batch* は、Batch という単語で始まるすべての CodeBuild アクションを指定し ます。次の例では、my で始まる名前のすべてのビルドプロジェクトへのアクセスを許可します。

arn:aws:codebuild:us-east-2:123456789012:project/my*

CodeBuild API オペレーションおよびコミットされたコードのアクションで必要なアクセス権限

BatchDeleteBuilds

アクション:codebuild:BatchDeleteBuilds

ビルドを削除するのに必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* BatchGetBuilds

アクション:codebuild:BatchGetBuilds

ビルドに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* BatchGetProjects

アクション:codebuild:BatchGetProjects

ビルドプロジェクトに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* BatchGetReportGroups

アクション:codebuild:BatchGetReportGroups

レポートグループに関する情報を取得するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

BatchGetReports

アクション:codebuild:BatchGetReports

レポートに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

BatchPutTestCases 1

アクション:codebuild:BatchPutTestCases

テストレポートを作成または更新するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

CreateProject

アクション:codebuild:CreateProject、iam:PassRole

ビルドプロジェクトを作成するのに必要です。

リソース:

• arn:aws:codebuild:region-ID:account-ID:project/project-name

• arn:aws:iam::account-ID:role/role-name

CreateReport ¹

アクション:codebuild:CreateReport

テストレポートを作成するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

CreateReportGroup

アクション:codebuild:CreateReportGroup

レポートグループを作成するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

CreateWebhook

アクション:codebuild:CreateWebhook

ウェブフックを作成するために必要です。

リソース:arn:aws:codebuild:*region-ID:account-ID*:project/*project-name*

DeleteProject

アクション:codebuild:DeleteProject

CodeBuild プロジェクトを削除するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:project/project-name

DeleteReport

アクション:codebuild:DeleteReport

レポートを削除するのに必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

DeleteReportGroup

アクション:codebuild:DeleteReportGroup

レポートグループを削除するのに必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

DeleteSourceCredentials

アクション:codebuild:DeleteSourceCredentials

GitHub、GitHub Enterprise Server、または Bitbucket リポジトリの認証情報が含まれている一連の SourceCredentialsInfo オブジェクトを削除するために必要です。

リソース: *

DeleteWebhook

アクション:codebuild:DeleteWebhook

ウェブフックを作成するために必要です。

リソース:arn:aws:codebuild:*region-ID:account-ID*:project/*project-name*

DescribeTestCases

アクション:codebuild:DescribeTestCases

ページ分割されたテストケースのリストを返すために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

ImportSourceCredentials

アクション:codebuild:ImportSourceCredentials

GitHub、GitHub Enterprise Server、または Bitbucket リポジトリの認証情報が含まれている一連の SourceCredentialsInfo オブジェクトをインポートするために必要です。

リソース:*

InvalidateProjectCache

アクション:codebuild:InvalidateProjectCache

プロジェクトのキャッシュをリセットするために必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* ListBuildBatches

アクション:codebuild:ListBuildBatches

ビルドバッチ ID のリストを取得するために必要です。

リソース:*

ListBuildBatchesForProject

アクション:codebuild:ListBuildBatchesForProject

特定のプロジェクトのビルドバッチ ID のリストを取得するために必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* ListBuilds

アクション:codebuild:ListBuilds

ビルド ID のリストを取得するのに必要です。

リソース: *

ListBuildsForProject

アクション:codebuild:ListBuildsForProject

ビルドプロジェクトのビルド ID のリストを取得するために必要です。

リソース:arn:aws:codebuild:*region-ID:account-ID*:project/*project-name*

ListCuratedEnvironmentImages

アクション:codebuild:ListCuratedEnvironmentImages

AWS CodeBuildによって管理されるすべての Docker イメージに関する情報を取得するのに必要 です。

リソース: * (必須ですが、アドレスで呼び出せる AWS リソースは参照しません)

ListProjects

アクション:codebuild:ListProjects

ビルドプロジェクト名のリストを取得するのに必要です。

リソース:*

ListReportGroups

アクション:codebuild:ListReportGroups

レポートグループのリストを取得するために必要です。

リソース: *

ListReports

アクション:codebuild:ListReports

レポートリストを取得するために必要です。

リソース: *

ListReportsForReportGroup

アクション:codebuild:ListReportsForReportGroup

レポートグループのレポートのリストを取得するために必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:report-group/*report*group-name

RetryBuild

アクション:codebuild:RetryBuild

ビルドを再試行するのに必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* StartBuild

アクション:codebuild:StartBuild

ビルドの実行を開始するために必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* StopBuild

アクション:codebuild:StopBuild

実行中のビルドを停止しようとするのに必要です。

リソース: arn:aws:codebuild:*region-ID:account-ID*:project/*project-name* UpdateProject

アクション: codebuild:UpdateProject、iam:PassRole

ビルドに関する情報を変更するのに必要です。

リソース:

• arn:aws:codebuild:region-ID:account-ID:project/project-name

• arn:aws:iam::account-ID:role/role-name

UpdateProjectVisibility

アクション: codebuild:UpdateProjectVisibility、iam:PassRole

プロジェクトのビルドの公開可視性を変更するために必要です。

リソース:

- arn:aws:codebuild:region-ID:account-ID:project/project-name
- arn:aws:iam::account-ID:role/role-name

UpdateReport ¹

アクション:codebuild:UpdateReport

テストレポートを作成または更新するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

UpdateReportGroup

アクション:codebuild:UpdateReportGroup

レポートグループを更新するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:report-group/report-

group-name

UpdateWebHook

アクション:codebuild:UpdateWebhook

Webhook を更新するために必要です。

リソース: arn:aws:codebuild:region-ID:account-ID:project/project-name

¹ アクセス許可にのみ使用されます。このアクションに API はありません。

タグを使用した AWS CodeBuild リソースへのアクセスのコントロール

IAM ポリシーステートメントの条件は、CodeBuild プロジェクトベースのアクションに対するアク セス許可を指定するために使用できる構文の一部です。プロジェクトに関連付けられたタグに基づ いてプロジェクトに対するアクションを許可または拒否するポリシーを作成し、これらのポリシー を、ユーザーの管理用に設定した IAM グループに適用できます。コンソールまたは を使用してプロ ジェクトにタグを適用する方法については AWS CLI、「」を参照してください<u>でビルドプロジェク</u> トを作成する AWS CodeBuild。CodeBuild SDK を使用したタグの適用については、CodeBuild API リファレンスの「CreateProject」および「タグ」を参照してください。。タグを使用して AWS リ ソースへのアクセスを制御する方法については、IAM ユーザーガイドの「リソースタグを使用した AWS リソースへのアクセスの制御」を参照してください。

▲ Important

リザーブドキャパシティ機能を使用すると、ソースファイル、Docker レイヤー、buildspec で指定されキャッシュされたディレクトリなどを含む、フリートインスタンスにキャッシュ されたデータに、同じアカウント内の他のプロジェクトからアクセスできます。これは設計 によるもので、同じアカウント内のプロジェクトがフリートインスタンスを共有できるよう にしています。

Example 例 1: リソースタグに基づいてプロジェクトに対する CodeBuild アクションを制限する

次の例では、キー BatchGetProjects とキー値 Environment のタグが付いているプロジェクト に対するすべての Production アクションを拒否します。ユーザーの管理者は、この IAM ポリシー をマネージド型のユーザーポリシーに加えて、承認されないユーザーにアタッチする必要がありま す。aws:ResourceTag 条件キーを使用して、リソースへのアクセスをリソースタグに基づいてコ ントロールします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Example 例 2: リクエストタグに基づいてプロジェクトに対する CodeBuild アクションを制限する

次のポリシーでは、リクエスト内のタグのキーが CreateProject で、キー値が Environment で ある場合、ユーザーに Production アクションへのアクセス許可を拒否します。さらに、このポリ シーでは、aws:TagKeys 条件キーを使用して、リクエスト内のタグのキーが UpdateProject で ある場合に、Environment を許可しないことにより、これらの承認されないユーザーにプロジェク トの変更を禁止します。管理者は、これらのアクションの実行を承認されないユーザーに、マネー ジド型のユーザーポリシーに加えて、この IAM ポリシーをアタッチする必要があります。この aws:RequestTag 条件キーを使用して、IAM リクエストで渡すことができるタグをコントロールし ます

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:CreateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:RequestTag/Environment": "Production"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:UpdateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["Environment"]
        }
      }
    }
  ]
}
```

Example 例 3: リソースタグに基づいてレポートグループのアクションを拒否または許可する

これらのリソースに関連付けられた AWS タグに基づいて CodeBuild リソース (プロジェクトお よびレポートグループ) に対するアクションを許可または拒否するポリシーを作成し、ユーザー を管理するために設定した IAM グループにそれらのポリシーを適用できます。たとえば、 AWS タグキーStatusとキー値が のレポートグループですべての CodeBuild アクションを拒否するポ リシーを作成しSecret、そのポリシーを一般的な開発者 (###) 用に作成した IAM グループに適 用できます。次に、上記のタグ付けされたレポートグループに対して作業する開発者が一般的な *Developers* グループのメンバーではなく、代わりに制限されたポリシーが適用されていない別の IAM グループ (SecretDevelopers) に属していることを確認する必要があります。

以下の例では、キー Status およびキー値 Secret でタグ付けされたレポートグループに対するす べての CodeBuild アクションを拒否します。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "codebuild:BatchGetReportGroups,"
        "codebuild:CreateReportGroup",
        "codebuild:DeleteReportGroup",
        "codebuild:ListReportGroups",
        "codebuild:ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
       1
      "Resource" : "*",
      "Condition" : {
         "StringEquals" : "aws:ResourceTag/Status": "Secret"
        }
    }
  ]
}
```

Example 例 4: リソースタグに基づいて AWSCodeBuildDeveloperAccess への CodeBuild アクション を制限する

特定のタグが付けられていないすべてのレポートグループおよびプロジェクトに対する CodeBuild アクションを許可するポリシーを作成できます。たとえば、以下のポリシーでは、指定したタグが 付けられたものを除くすべてのレポートグループとプロジェクトに <u>AWSCodeBuildDeveloperAccess</u> と同等のアクセス許可を付与します。

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

```
"Effect": "Allow",
         "Action": [
            "codebuild:StartBuild",
            "codebuild:StopBuild",
            "codebuild:BatchGet*",
            "codebuild:GetResourcePolicy",
            "codebuild:DescribeTestCases",
            "codebuild:List*",
            "codecommit:GetBranch",
            "codecommit:GetCommit",
            "codecommit:GetRepository",
            "codecommit:ListBranches",
            "cloudwatch:GetMetricStatistics",
            "events:DescribeRule",
            "events:ListTargetsByRule",
            "events:ListRuleNamesByTarget",
            "logs:GetLogEvents",
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets"
         ],
         "Resource": "*",
         "Condition": {
            "StringNotEquals": {
               "aws:ResourceTag/Status": "Secret",
               "aws:ResourceTag/Team": "Saanvi"
            }
         }
      }
   ]
}
```

コンソールでのリソースの表示

AWS CodeBuild コンソールには、サインインしている AWS リージョンの AWS アカウントのリポ ジトリのリストを表示するListRepositoriesアクセス許可が必要です。このコンソールには、大 文字と小文字を区別しない検索をリソースに対して迅速に実行するための [Go to resource (リソー スに移動)] 機能も含まれています。この検索は、サインインしている AWS リージョンのアカウント AWS で実行されます。次のリソースは、以下のサービス全体で表示されます。

- AWS CodeBuild: ビルドプロジェクト
- ・ AWS CodeCommit: リポジトリ
- ・ AWS CodeDeploy: アプリケーション

・ AWS CodePipeline: パイプライン

この検索をすべてのサービスのリソースにわたって実行するには、次のアクセス権限が必要です。

- CodeBuild: ListProjects
- CodeCommit: ListRepositories
- CodeDeploy: ListApplications
- CodePipeline: ListPipelines

あるサービスに対するアクセス権限がない場合、そのサービスのリソースに関して結果は返されませ ん。表示のアクセス権限がある場合でも、表示に対する明示的な Deny が設定されているリソースに ついては、結果が返されません。

AWS CodeBuild のコンプライアンス検証

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として AWS CodeBuild のセキュリティと AWS コンプライアンスを評価します。これらのプログラムに は、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラム AWS の範囲内のサービスのリストについては、<u>AWS コンプラ</u> <u>イアンスプログラムの範囲内のサービス</u>を参照してください。一般的な情報については、「<u>AWS コ</u> ンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細について は、AWS 「Artifact でのレポートのダウンロード」を参照してください。

CodeBuild 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。CodeBuild の使用が HIPAA、PCI、FedRAMP などの標準への準拠の対象である場合、 は以下に役立つリソース AWS を 提供します。

- セキュリティとコンプライアンスのクイックスタートガイド これらのデプロイガイドでは、 アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いた ベースライン環境をデプロイする手順について説明します AWS。
- Architecting for HIPAA Security and Compliance ホワイトペーパー このホワイトペーパーでは、 企業が AWS を使用して HIPAA 準拠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。

- <u>AWS コンプライアンスリソース</u> このワークブックとガイドのコレクションは、お客様の業界や
 地域に適用される場合があります。
- <u>AWS Config</u> この AWS サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。
- <u>AWS Security Hub</u> を使用して AWS CodeBuild、セキュリティのベストプラクティスに関連するの使用状況をモニタリングします<u>AWS Security Hub</u>。Security Hub は、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub を使用して CodeBuild リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「<u>AWS CodeBuild</u>コントロール」を参照してください。

の耐障害性 AWS CodeBuild

AWS グローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心に 構築されています。AWS リージョンは、低レイテンシー、高スループット、および高度に冗長な ネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。 アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイル オーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリ ティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障 害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラスト ラクチャ」を参照してください。

インフラストラクチャセキュリティ in AWS CodeBuild

マネージドサービスである AWS CodeBuild は、 AWS グローバルネットワークセキュリティで保護 されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法につい ては、<u>AWS「 クラウドセキュリティ</u>」を参照してください。インフラストラクチャセキュリティの ベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で CodeBuild 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。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットア クセスキーを使用して署名する必要があります。または<u>AWS Security Token Service</u> (AWS STS) を 使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

CodeBuild でソースプロバイダにアクセスする

GitHub または GitHub Enterprise Server では、個人用アクセストークン、Secrets Manager シー クレット、接続、または OAuth アプリケーションを使用してソースプロバイダーにアクセスしま す。Bitbucket では、アクセストークン、アプリパスワード、Secrets Manager シークレット、接 続、または OAuth アプリのいずれかを使用して、ソースプロバイダーにアクセスします。

トピック

- Secrets Manager シークレットにトークンを作成して保存
- CodeBuild の GitHub および GitHub Enterprise Server アクセス
- CodeBuild での Bitbucket アクセス
- CodeBuild での GitLab アクセス

Secrets Manager シークレットにトークンを作成して保存

Secrets Manager を使用してアクセストークンを保存する場合は、既存のシークレット接続を使用するか、新しいシークレットを作成できます。新しいシークレットを作成するには、次の手順に従います。

AWS Management Console

で Secrets Manager シークレットを作成するには AWS Management Console

- 1. [ソースプロバイダ] には、[Bitbucket]、[GitHub]、または [GitHub Enterprise] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - [デフォルトソース認証情報] を選択し、アカウントのデフォルトソース認証情報を使用 して、すべてのプロジェクトに適用します。
- a. ソースプロバイダに接続していない場合は、[デフォルトソース認証情報を管理] を 選択します。
- b. [認証情報タイプ] では、[CodeConnections] 以外の認証情報タイプを選択します。
- c. [サービス] では、[Secrets Manager] を選択し、[シークレット] で [新しいシーク レット] を選択します。
- d. [シークレット名] で、シークレットの名前を入力します。
- e. [シークレットの説明 オプション] に、シークレットの説明を入力します。
- f. 選択したソースプロバイダに応じて、トークンまたはユーザー名とアプリパスワー ドを入力し、[保存]を選択します。
- [カスタムソース認証情報]を選択し、カスタムソース認証情報を使用してアカウントの デフォルト設定を上書きします。
 - a. [認証情報タイプ]では、[CodeConnections] 以外の認証情報タイプを選択します。
 - b. [接続]で、[シークレットを作成]を選択します。
 - c. [シークレット名] で、シークレットの名前を入力します。
 - d. [シークレットの説明 オプション] に、シークレットの説明を入力します。
 - e. 選択したソースプロバイダに応じて、トークンまたはユーザー名とアプリパスワードを入力し、[作成]を選択します。

AWS CLI

で Secrets Manager シークレットを作成するには AWS CLI

ターミナル (Linux/macOS/Unix) またはコマンドプロンプト (Windows) を開きます。 AWS
 CLI を使用して Secrets Manager create-secret コマンドを実行します。

Key=codebuild:source:provider,Value=<provider>

CodeBuild が受け入れる Secrets Manager シークレットは、CodeBuild プロジェクトと同じ アカウントと AWS リージョンにあり、次の JSON 形式である必要があります。

```
{
    "ServerType": ServerType,
    "AuthType: AuthType,
    "Token": string,
    "Username": string // Optional and is only used for Bitbucket app
password
    }
```

フィールド	有効値	説明
ServerType	GITHUB GITHUB_ENTERPRISE BITBUCKET	Secrets Manager シーク レットのサードパーティー ソースプロバイダです。
AuthType	PERSONAL_ACCESS_TO KEN BASIC_AUTH	認証情報で使用される アクセストークンのタ イプです。GitHub で は、PERSONAL_ACCESS _TOKEN のみが有効で す。BASIC_AUTH は Bitbucket アプリパスワード にのみ有効です。
トークン	string	GitHub または GitHub Enterprise では、これは個 人用アクセストークンで す。Bitbucket の場合、これ はアクセストークンまたは Bitbucket アプリパスワード のいずれかです。

フィールド	有効値	説明
ユーザーネーム	string	AuthType が BASIC_AUT H の場合の Bitbucket ユー ザー名です。その他のタ イプのソースプロバイダで は、このパラメータは有効 ではありません。

さらに、CodeBuild は、シークレットに次のリソースタグを使用して、プロジェクトを作成 または編集するときにシークレットを簡単に選択できるようにします。

タグキー	タグ値	説明
codebuild:source:provider	GitHub github_enterprise bitbucket	このシークレットの対 象となるプロバイダを CodeBuild に伝えます。
codebuild:source:type	personal_access_token basic_auth	このシークレットのアク セストークンのタイプを CodeBuild に伝えます。

CodeBuild の GitHub および GitHub Enterprise Server アクセス

GitHub の場合、個人用アクセストークン、OAuth アプリ、Secrets Manager シークレット、または GitHub アプリ接続を使用して、ソースプロバイダにアクセスできます。GitHub Enterprise Server の 場合、個人用アクセストークン、Secrets Manager シークレット、GitHub アプリ接続を使用して、 ソースプロバイダにアクセスできます。

トピック

- ・ <u>GitHub および GitHub Enterprise Server の GitHub アプリ接続</u>
- GitHub および GitHub Enterprise Server アクセストークン
- <u>GitHub OAuth アプリ</u>

GitHub および GitHub Enterprise Server アクセス

GitHub および GitHub Enterprise Server の GitHub アプリ接続

GitHub アプリを使用して CodeBuild に接続できます。GitHub アプリ接続は <u>AWS CodeConnections</u> を通じてサポートされています。

ソースプロバイダアクセスを使用すると、<u>CreateWebhook</u>を使用して <u>GitHub ウェブフックイベン</u> <u>ト</u>をサブスクライブすることでビルドをトリガーしたり、CodeBuild で <u>チュートリアル: CodeBuild</u> がホストする GitHub Actions ランナーを設定 を使用したりできます。

Note

CodeConnections は、使用できるリージョンが CodeBuild よりも限られていま す。CodeBuild では、クロスリージョン接続を使用できます。オプトインリージョン で作成された接続は、他のリージョンでは使用できません。詳細については、「<u>AWS</u> CodeConnections エンドポイントとクォータ」を参照してください。

トピック

- ステップ 1: GitHub アプリへの接続を作成 (コンソール)
- ステップ 2: 接続を使用するために CodeBuild プロジェクト IAM ロールにアクセスを許可
- ・ ステップ 3: 新しい接続を使用するように CodeBuild を設定
- GitHub アプリに関する問題のトラブルシューティング

ステップ 1: GitHub アプリへの接続を作成 (コンソール)

以下のステップを使用して、CodeBuild コンソールで GitHub 内のプロジェクト用に接続を追加しま す。

GitHub への接続を作成するには

 「デベロッパーツールユーザーガイド」にある「<u>Create a connection to GitHub</u>」の手順に従っ てください。 Note

アカウントで既存の接続を作成または使用する代わりに、別の AWS アカウントから共有さ れた接続を使用できます。詳細については、<u>AWS 「アカウントとの接続の共有</u>」を参照し てください。

ステップ 2: 接続を使用するために CodeBuild プロジェクト IAM ロールにアクセスを許可

CodeBuild プロジェクト IAM ロールに、接続によって提供された GitHub トークンを使用するための アクセスを許可できます。

CodeBuild プロジェクトの IAM ロールにアクセスを許可するには

- CodeBuild プロジェクトの <u>CodeBuild が他の AWS のサービスとやり取りすることを許可</u> の手順に従って、CodeBuild プロジェクトの IAM ロールを作成します。
- 手順に従って、次の IAM ポリシーを CodeBuild プロジェクトロールに追加して、接続へのアク セスを許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
        "Effect": "Allow",
        "Action": [
           "codeconnections:GetConnectionToken",
           "codeconnections:GetConnection"
        ],
        "Resource": [
           <connection-arn>
        ]
      }
   ]
}
```

ステップ 3: 新しい接続を使用するように CodeBuild を設定

接続をアカウントレベルの認証情報として設定し、プロジェクトで使用できます。

AWS Management Console

でアカウントレベルの認証情報として接続を設定するには AWS Management Console

- 1. [ソースプロバイダー] で [GitHub] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - [デフォルトソース認証情報]を選択し、アカウントのデフォルトソース認証情報を使用して、すべてのプロジェクトに適用します。
 - a. GitHub に接続していない場合は、[デフォルトソース認証情報を管理] を選択します。
 - b. [認証情報タイプ] では、[GitHub アプリ] を選択します。
 - c. [接続]で、既存の接続を使用するか、新規の接続を作成することを選択します。
 - [カスタムソース認証情報]を選択し、カスタムソース認証情報を使用してアカウントの デフォルト設定を上書きします。
 - a. [認証情報タイプ] では、[GitHub アプリ] を選択します。
 - b. [接続]で、既存の接続を使用するか、新規の接続を作成することを選択します。

AWS CLI

でアカウントレベルの認証情報として接続を設定するには AWS CLI

 ターミナル (Linux/macOS/Unix) またはコマンドプロンプト (Windows) を開きます。 AWS CLI を使用して import-source-credentials コマンドを実行し、--token接続に --authtype、--server-type、 を指定します。

以下のコマンドを使用します。

aws codebuild import-source-credentials --auth-type CODECONNECTIONS --servertype GITHUB --token <connection-arn>

CodeBuild プロジェクトに複数のトークンを設定することもできます。詳細については、「<u>複数の</u> トークンをソースレベルの認証情報として設定」を参照してください。

GitHub アプリに関する問題のトラブルシューティング

以下の情報は、GitHub アプリの一般的な問題のトラブルシューティングに役立ちます。

トピック

- AWS Connector for GitHub アプリを望ましくないリージョンにインストールする
- GitHub アプリ接続にリポジトリへのアクセス権限がありません。
- ・ AWS サービスの IAM ロールに必要な IAM アクセス許可がありません。

AWS Connector for GitHub アプリを望ましくないリージョンにインストールする

問題: GitHub Marketplace から AWS Connector for GitHub をインストールしましたが、接続が望ま しくないリージョンで作成されました。GitHub ウェブサイトでアプリを再設定しようとしても、ア プリが既に GitHub アカウントにインストールされているため、動作しません。

考えられる原因: アプリは GitHub アカウントに既にインストールされているため、アプリのアクセ ス許可のみを再設定できます。

推奨される解決策: インストール ID を使用して目的のリージョンに新しい接続を作成できます。

- <u>https://console.aws.amazon.com/codesuite/settings/connections</u> で CodeConnections コンソー ルを開き、 AWS コンソールナビゲーションバーのリージョンセレクターを使用して目的のリー ジョンに移動します。
- 2. 「デベロッパーツールユーザーガイド」にある「<u>Create a connection to GitHub</u>」の手順に従っ てください。

Note

AWS Connector for GitHub アプリはインストール済みであるため、新しいアプリをイン ストールする代わりに選択できます。

GitHub アプリ接続にリポジトリへのアクセス権限がありません。

問題: CodeBuild や CodePipeline などの接続を使用する AWS サービスは、リポジトリにアクセス できないか、リポジトリが存在しないことを報告します。考えられるエラーメッセージには、次のよ うなものがあります。

- Authentication required for primary source.
- Unable to create webhook at this time. Please try again later.
- Failed to create webhook. GitHub API limit reached. Please try again later.

考えられる原因: GitHub アプリを使用していて、ウェブフックのアクセス許可スコープを付与してい ない可能性があります。

推奨される解決策: 必要なアクセス許可スコープを付与するには、「<u>Navigating to the GitHub App</u> you want to review or modify」の指示に従って、インストールされたアプリを設定します。アク セス許可セクションには、アプリにウェブフックアクセス許可がないことが表示され、新しくリ クエストされたアクセス許可を確認するオプションがあります。新しいアクセス許可を確認して 承諾します。詳細については、「<u>Approving updated permissions for a GitHub App</u>」を参照して ください。

考えられる原因: 接続は想定どおりに機能していましたが、突然リポジトリにアクセスできなくなり ました。

考えられる解決策: まず、「<u>authorizations</u>」と「<u>installations</u>」を確認してから、GitHub アプリが 承認されてインストールされていることを確認します。GitHub アプリのインストールが一時停 止されている場合は、一時停止を解除する必要があります。GitHub アプリが <u>UAT (ユーザーアク</u> セストークン) 接続に対して承認されていない場合、または <u>IAT (インストールアクセストークン)</u> 接続に対してインストールされていない場合、既存の接続は使用できなくなるため、新しい接続 を作成する必要があります。GitHub アプリを再インストールしても、古いインストールに関連付 けられた以前の接続は復元されないことに注意してください。

考えられる解決策: 接続が UAT 接続の場合は、複数の CodeBuild ビルドの同時実行で使用されて いるなど、接続が同時に使用されていないことを確認してください。これは、有効期限が切れる トークンが接続によって更新された場合、GitHub が以前に発行された UAT を直ちに無効にする ためです。CodeBuild の複数の同時ビルドに UAT 接続を使用する必要がある場合は、複数の接続 を作成し、各接続を個別に使用できます。

考えられる解決策: UAT 接続が過去 6 か月間使用されていない場合、接続は GitHub によって無 効になります。これを修正するには、新しい接続を作成します。

考えられる原因: アプリをインストールせずに UAT 接続を使用していた可能性があります。

推奨される解決策: UAT 接続を作成する際に、接続を GitHub アプリのインストールに関連付け る必要はありませんが、リポジトリにアクセスできるようにするためにはインストールが必要で す。手順に従って<u>インストールを確認</u>し、GitHub アプリがインストールされていることを確認し ます。インストールされていない場合は、<u>GitHub アプリのページ</u>に移動してアプリをインストー ルします。UAT のアクセスの詳細については、「<u>About user access tokens</u>」を参照してくださ い。 AWS サービスの IAM ロールに必要な IAM アクセス許可がありません。

問題: 次のいずれかのエラーメッセージが表示されます。

- Access denied to connection <connection-arn>
- Failed to get access token from <connection-arn>

推奨される解決策:通常、CodePipeline や CodeBuild などの AWS サービスとの接続を使用しま す。AWS サービスに IAM ロールを付与すると、AWS サービスはロールのアクセス許可を使用して ユーザーに代わって動作できます。IAM ロールに必要なアクセス許可が付与されていることを確認 してください。必要な IAM アクセス許可の詳細については、「デベロッパーツールコンソールユー ザーガイド」の<u>「接続を使用するための CodeBuild プロジェクト IAM ロールアクセスの付与</u>」およ び<u>AWS CodeStar 「通知と CodeConnections のアイデンティティとアクセス管理</u>」を参照してくだ さい。

GitHub および GitHub Enterprise Server アクセストークン

アクセストークンの前提条件

開始する前に、GitHub アクセストークンへの適切なアクセス許可スコープを追加する必要がありま す。

GitHub では、個人用アクセストークンに次のスコープが必要です。

- repo: プライベートリポジトリのフルコントロールを許可します。
- repo:status: パブリックおよびプライベートリポジトリのコミットステータスへの読み取り/書き込みアクセスを許可します。
- admin:repo_hook: リポジトリフックのフルコントロールを許可します。このスコープは、トークンに repo スコープがある場合は必要ありません。
- admin:org_hook: 組織フックの完全な制御を付与します。このスコープは、組織のウェブフック機能を使用している場合にのみ必要です。

詳細については、GitHub ウェブサイトの <u>Understanding Scopes for OAuth Apps</u> を参照してくださ い。

きめ細かい個人用アクセストークンを使用している場合、ユースケースによっては、個人用アクセス トークンに次のアクセス許可が必要になる場合があります。

- Contents: Read-only: プライベートリポジトリへのアクセスを付与します。このアクセス許可は、 プライベートリポジトリをソースとして使用している場合に必要です。
- Commit statuses: Read and write: コミットステータスを作成するアクセス許可を付与します。このアクセス許可は、プロジェクトにウェブフックが設定されている場合、または [ビルドのステータスを報告] 機能が有効になっている場合に必要です。
- Webhooks: Read and write: ウェブフックを管理するアクセス許可を付与します。このアクセス許可は、プロジェクトにウェブフックが設定されている場合に必要です。
- Pull requests: Read-only: プルリクエストにアクセスするアクセス許可を付与します。このアクセ ス許可は、ウェブフックにプルリクエストイベントに対する FILE_PATH フィルタがある場合に必 要です。
- Administration: Read and write: このアクセス許可は、CodeBuild でセルフホスト型の GitHub Actions ランナー機能を使用している場合に必要です。詳細については、「<u>Create a registration</u> <u>token for a repository</u>」と「<u>チュートリアル: CodeBuild がホストする GitHub Actions ランナーを</u> 設定」を参照してください。

Note

組織リポジトリにアクセスする場合は、アクセストークンのリソース所有者として組織を指 定する必要があります。

詳細については、GitHub ウェブサイトの「<u>Permissions required for fine-grained personal access</u> tokens」を参照してください。

GitHub をアクセストークンで接続する(コンソール)

コンソールを使用し、アクセストークンを使用してプロジェクトを GitHub に接続するには、プロ ジェクトを作成するときに以下の操作を実行します。詳細については、<u>ビルドプロジェクトの作成</u> (コンソール) を参照してください。

- 1. [ソースプロバイダー] で [GitHub] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - アカウント認証情報を使用して、アカウントのデフォルトのソース認証情報をすべてのプロジェクトに適用することを選択します。
 - a. GitHub に接続していない場合は、アカウント認証情報の管理を選択します。

- b. [認証情報タイプ]では、[個人用アクセストークン]を選択します。
- サービスにアカウントレベルの認証情報を使用することを選択した場合は、トークンの保存 に使用するサービスを選択し、以下を実行します。
 - a. Secrets Manager を使用する場合は、既存のシークレット接続を使用するか、新しい シークレットを作成し、保存を選択します。新しいシークレットの作成方法の詳細につ いては、「<u>Secrets Manager シークレットにトークンを作成して保存</u>」を参照してくだ さい。
 - b. CodeBuild を使用する場合は、GitHub の個人用アクセストークンを入力し、保存を選択します。
- このプロジェクトのオーバーライド認証情報を使用するを選択して、カスタムソース認証情報を使用してアカウントの認証情報設定を上書きします。
 - a. 入力された認証情報リストから、個人用アクセストークンの下にあるオプションのいず れかを選択します。
 - b. 説明で新しい個人用アクセストークン接続の作成を選択して、新しい個人用アクセス トークンを作成することもできます。

GitHub をアクセストークンで接続する(CLI)

アクセストークンを使用してプロジェクトを GitHub AWS CLI に接続するには、次の手順に従いま す。 AWS CLI で を使用する方法については AWS CodeBuild、「」を参照してください<u>コマンドラ</u> インリファレンス。

1. import-source-credentials コマンドを実行します。

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 形式のデータが出力に表示されます。 AWS CLI がインストールされているロー カルコンピュータまたはインスタンス上の場所にあるファイル(など*import-sourcecredentials.json*)にデータをコピーします。コピーされたデータを次のように変更して、 結果を保存します。

```
{
    "serverType": "server-type",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
```

```
ユーザーガイド
```

```
"token": "token",
"username": "username"
}
```

以下に置き換えます。

- server-type: 必須値。この認証情報に使用されるソースプロバイダー。有効な値は GITHUB、BITBUCKET、GITHUB_ENTERPRISE、GITLAB、GITLAB_SELF_MANAGED で す。
- auth-type: 必須値。リポジトリへの接続に使用される認証のタイプ。有効な値 は、OAUTH、BASIC_AUTH、PERSONAL_ACCESS_TOKEN、CODECONNECTIONS、SECRETS_ です。GitHub では、PERSONAL_ACCESS_TOKEN のみが許可されます。BASIC_AUTH は、Bitbucket アプリパスワードでのみ許可されます。
- should-overwrite:オプションの値。リポジトリソースの認証情報が上書きされないよう にするには、false に設定します。リポジトリソースの認証情報を上書きするには、true に 設定します。デフォルト値は true です。
- token: 必須値。GitHub または GitHub Enterprise Server の場合、これは個人用アクセストークンです。Bitbucket の場合、これは個人用アクセストークンまたはアプリパスワードです。認証タイプが CODECONNECTIONS の場合、これは接続 ARN です。認証タイプが SECRETS_MANAGER の場合、これはシークレット ARN です。
- username: オプションの値。このパラメーターは、GitHub および GitHub エンタープライズ サーバーソースプロバイダーでは無視されます。
- アカウントをアクセストークンに接続するには、ステップ1で保存した import-sourcecredentials.json ファイルが含まれるディレクトリに切り替え、もう一度 import-sourcecredentials コマンドを実行します。

aws codebuild import-source-credentials --cli-input-json file://import-sourcecredentials.json

JSON 形式のデータが、Amazon リソースネーム (ARN) を持つ出力に表示されます。

{
 "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}

Note

同じサーバータイプと認証タイプを持つ import-source-credentials コマンドを 2 回目に 実行した場合、保存されたアクセストークンが更新されます。

アカウントがアクセストークンに接続されたら、CodeBuild を使用して create-project プロ ジェクトを作成できます。詳細については、「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」を参照 してください。

3. 接続されたアクセストークンを表示するには、list-source-credentials コマンドを実行します。

aws codebuild list-source-credentials

JSON 形式 sourceCredentialsInfos オブジェクトが出力に表示されます。

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
            "serverType": "server-type",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObjectには、接続されたソース認証情報のリストが含まれています。

- authTypeは、認証情報により使用される認証のタイプです。これ は、OAUTH、BASIC_AUTH、PERSONAL_ACCESS_TOKEN、CODECONNECTIONS、または SECRETS_MANAGERです。
- serverTypeは、ソースプロバイダーのタイプです。これ は、GITHUB、GITHUB_ENTERPRISE、BITBUCKET、GITLAB、または GITLAB_SELF_MANAGEDです。
- ・ arn は、トークンの ARN です。
- 4. ソースプロバイダーから切断してそのアクセストークンを削除するには、その ARN を使用して delete-source-credentials コマンドを実行します。

aws codebuild delete-source-credentials --arn arn-of-your-credentials

削除された認証情報の ARN とともに JSON 形式のデータが返されます。

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

GitHub OAuth アプリ

OAuth を使用して GitHub に接続 (コンソール)

コンソールを使用して、OAuth アプリでプロジェクトを GitHub に接続するには、プロジェクトを 作成するときに以下の操作を実行します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u> ル)」を参照してください。

- 1. [ソースプロバイダー] で [GitHub] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - アカウント認証情報を使用して、アカウントのデフォルトのソース認証情報をすべてのプロジェクトに適用することを選択します。
 - a. GitHub に接続していない場合は、アカウント認証情報の管理を選択します。
 - b. 認証情報タイプで、OAuth アプリを選択します。
 - サービスにアカウントレベルの認証情報を使用することを選択した場合は、トークンの保存 に使用するサービスを選択し、以下を実行します。
 - a. Secrets Manager を使用する場合は、既存のシークレット接続を使用するか、新しい シークレットを作成するかを選択し、保存を選択します。新しいシークレットの作成方 法の詳細については、「<u>Secrets Manager シークレットにトークンを作成して保存</u>」を 参照してください。
 - b. CodeBuild の使用を選択した場合は、保存を選択します。
 - このプロジェクトのオーバーライド認証情報を使用するを選択して、カスタムソース認証情 報を使用してアカウントの認証情報設定を上書きします。
 - a. 入力された認証情報リストから、OAuth アプリのオプションのいずれかを選択します。

b. 説明で新しい Oauth アプリケーショントークン接続の作成を選択して、新しい OAuth アプリケーショントークンを作成することもできます。

承認された OAuth アプリを確認するには、GitHub の「<u>Applications</u>」に移動し、<u>aws-codesuite</u> が所 有する AWS CodeBuild(*region*)という名前のアプリケーションがリストされていることを確認 します。

CodeBuild での Bitbucket アクセス

Bitbucket では、アクセストークン、アプリパスワード、OAuth アプリ、または Bitbucket 接続のいずれかを使用して、ソースプロバイダにアクセスします。

トピック

- Bitbucket アプリ接続
- Bitbucket アプリのパスワードまたはアクセストークン
- Bitbucket OAuth アプリ

Bitbucket アプリ接続

Bitbucket を使用して CodeBuild に接続できます。Bitbucket アプリ接続は <u>AWS CodeConnections</u> を 通じてサポートされています。

CodeConnections は、使用できるリージョンが CodeBuild よりも限られていま す。CodeBuild では、クロスリージョン接続を使用できます。オプトインリージョン で作成された接続は、他のリージョンでは使用できません。詳細については、「<u>AWS</u> <u>CodeConnections エンドポイントとクォータ</u>」を参照してください。

トピック

- <u>ステップ 1: Bitbucket への接続を作成 (コンソール)</u>
- ステップ 2: 接続を使用するために CodeBuild プロジェクト IAM ロールにアクセスを許可
- ・ <u>ステップ 3:</u> 新しい接続を使用するように CodeBuild を設定

Note

ステップ 1: Bitbucket への接続を作成 (コンソール)

以下のステップを使用して、CodeBuild コンソールで Bitbucket 内のプロジェクト用に接続を追加し ます。

Bitbucket への接続を作成するには

• 「デベロッパーツールユーザーガイド」にある「<u>Create a connection to Bitbucket</u>」の手順に 従ってください。

Note

アカウントで既存の接続を作成または使用する代わりに、別の AWS アカウントから共有さ れた接続を使用できます。詳細については、<u>AWS 「アカウントとの接続の共有</u>」を参照し てください。

ステップ 2: 接続を使用するために CodeBuild プロジェクト IAM ロールにアクセスを許可

CodeBuild プロジェクト IAM ロールに、接続によって提供された Bitbucket トークンを使用するためのアクセス許可を付与できます。

CodeBuild プロジェクトの IAM ロールにアクセスを付与するには

- CodeBuild プロジェクトの <u>CodeBuild が他の AWS のサービスとやり取りすることを許可</u> の手順に従って、CodeBuild プロジェクトの IAM ロールを作成します。
- 手順に従って、次の IAM ポリシーを CodeBuild プロジェクトロールに追加して、接続へのアク セスを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": [
            "codeconnections:GetConnectionToken",
            "codeconnections:GetConnection"
        ],
        "Resource": [
            <connection-arn>
```

ステップ 3: 新しい接続を使用するように CodeBuild を設定

接続をアカウントレベルの認証情報として設定し、プロジェクトで使用できます。

AWS Management Console

でアカウントレベルの認証情報として接続を設定するには AWS Management Console

- 1. [ソースプロバイダー] で、[Bitbucket] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - [デフォルトソース認証情報]を選択し、アカウントのデフォルトソース認証情報を使用 して、すべてのプロジェクトに適用します。
 - a. Bitbucket に接続していない場合は、[デフォルトソース認証情報を管理] を選択します。
 - b. [認証情報タイプ] では、[CodeConnections] を選択します。
 - c. [接続]で、既存の接続を使用するか、新規の接続を作成するかを選択します。
 - [カスタムソース認証情報]を選択し、カスタムソース認証情報を使用してアカウントの デフォルト設定を上書きします。
 - a. [認証情報タイプ] では、[CodeConnections] を選択します。
 - b. [接続] で、既存の接続を使用するか、新規の接続を作成するかを選択します。

AWS CLI

でアカウントレベルの認証情報として接続を設定するには AWS CLI

 ターミナル (Linux/macOS/Unix) またはコマンドプロンプト (Windows) を開きます。 AWS CLI を使用して import-source-credentials コマンドを実行し、--token接続に --authtype、--server-type、 を指定します。

以下のコマンドを使用します。

aws codebuild import-source-credentials --auth-type CODECONNECTIONS --servertype BITBUCKET --token <connection-arn>

CodeBuild プロジェクトで複数のトークンを設定する方法の詳細については、「<u>複数のトークンを</u> ソースレベルの認証情報として設定」を参照してください。

Bitbucket アプリのパスワードまたはアクセストークン

前提条件

開始する前に、Bitbucket アプリのパスワードまたはアクセストークンへの適切なアクセス許可ス コープを追加する必要があります。

Bitbucket では、アプリパスワードまたはアクセストークンに次のスコープが必要です。

- repository:read: 承認側ユーザーがアクセスできるすべてのリポジトリへの読み取りアクセスを許可します。
- pullrequest:read: プルリクエストの読み取りアクセスを許可します。プロジェクトに Bitbucket
 ウェブフックがある場合、アプリパスワードまたはアクセストークンにこのスコープが必要です。
- webhook: Webhook へのアクセスを許可します。プロジェクトにウェブフックペレーションがある場合、アプリパスワードまたはアクセストークンにこのスコープが必要です。

詳細については、Bitbucket ウェブサイトの「<u>Scopes for Bitbucket Cloud REST API</u>」と「<u>OAuth on</u> Bitbucket Cloud」を参照してください。

アプリケーションパスワードで Bitbucket へ接続する(コンソール)

コンソールを使用し、アクセストークンを使用してプロジェクトを Bitbucket に接続するには、プロ ジェクトを作成するときに以下の操作を実行します。詳細については、<u>ビルドプロジェクトの作成</u> (コンソール) を参照してください。

- 1. [ソースプロバイダー] で、[Bitbucket] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - アカウント認証情報を使用して、アカウントのデフォルトのソース認証情報をすべてのプロジェクトに適用することを選択します。
 - a. Bitbucket に接続していない場合は、アカウント認証情報の管理を選択します。

- b. [認証情報タイプ]では、[アプリパスワード]を選択します。
- サービスにアカウントレベルの認証情報を使用することを選択した場合は、トークンの保存 に使用するサービスを選択し、以下を実行します。
 - a. Secrets Manager を使用する場合は、既存のシークレット接続を使用するか、新しい シークレットを作成するかを選択し、保存を選択します。新しいシークレットの作成方 法の詳細については、「<u>Secrets Manager シークレットにトークンを作成して保存</u>」を 参照してください。
 - b. CodeBuild を使用する場合は、Bitbucket のユーザー名とパスワードを入力し、保存を 選択します。
- このプロジェクトのオーバーライド認証情報を使用するを選択して、カスタムソース認証情 報を使用してアカウントの認証情報設定を上書きします。
 - a. 入力された認証情報リストから、アプリパスワードの下にあるオプションのいずれかを 選択します。
 - b. 説明で新しいアプリパスワード接続の作成を選択して、新しいアプリパスワードトーク ンを作成することもできます。

アクセストークンを使用して Bitbucket に接続 (コンソール)

コンソールを使用して、アクセストークンでプロジェクトを Bitbucket に接続するには、プロジェク トを作成するときに以下の操作を実行します。詳細については、「<u>ビルドプロジェクトの作成 (コン</u> ソール)」を参照してください。

- 1. [ソースプロバイダー] で、[Bitbucket] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - アカウント認証情報を使用して、アカウントのデフォルトのソース認証情報をすべてのプロジェクトに適用することを選択します。
 - a. Bitbucket に接続していない場合は、アカウント認証情報の管理を選択します。
 - b. [認証情報タイプ] では、[個人用アクセストークン] を選択します。
 - ・ サービスにアカウントレベルの認証情報を使用することを選択した場合は、トークンの保存 に使用するサービスを選択し、以下を実行します。
 - a. Secrets Manager を使用する場合は、既存のシークレット接続を使用するか、新しい シークレットを作成するかを選択し、保存を選択します。新しいシークレットの作成方

法の詳細については、「<u>Secrets Manager シークレットにトークンを作成して保存</u>」を 参照してください。

- b. CodeBuild を使用する場合は、Bitbucket の個人用アクセストークンを入力し、保存を 選択します。
- このプロジェクトのオーバーライド認証情報を使用するを選択して、カスタムソース認証情 報を使用してアカウントの認証情報設定を上書きします。
 - a. 入力された認証情報リストから、個人用アクセストークンの下にあるオプションのいず れかを選択します。
 - b. 説明で新しい個人用アクセストークン接続の作成を選択して、新しい個人用アクセス トークンを作成することもできます。

アプリパスワードまたはアクセストークンを使用して Bitbucket に接続 (CLI)

を使用して、アプリケーションパスワードまたはアクセストークンを使用してプロジェクトを Bitbucket AWS CLI に接続するには、次の手順に従います。 AWS CLI で を使用する方法については AWS CodeBuild、「」を参照してくださいコマンドラインリファレンス。

1. import-source-credentials コマンドを実行します。

aws codebuild import-source-credentials --generate-cli-skeleton

JSON 形式のデータが出力に表示されます。 AWS CLI がインストールされているロー カルコンピュータまたはインスタンス上の場所にあるファイル(など*import-sourcecredentials.json*) にデータをコピーします。コピーされたデータを次のように変更して、 結果を保存します。

```
{
    "serverType": "BITBUCKET",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
    "token": "token",
    "username": "username"
}
```

以下に置き換えます。

- server-type: 必須値。この認証情報に使用されるソースプロバイダー。有効な値は GITHUB、BITBUCKET、GITHUB_ENTERPRISE、GITLAB、GITLAB_SELF_MANAGED で す。
- auth-type: 必須値。リポジトリへの接続に使用される認証のタイプ。有効な値 は、OAUTH、BASIC_AUTH、PERSONAL_ACCESS_TOKEN、CODECONNECTIONS、SECRETS_ です。GitHub では、PERSONAL_ACCESS_TOKEN のみが許可されます。BASIC_AUTH は、Bitbucket アプリパスワードでのみ許可されます。
- should-overwrite:オプションの値。リポジトリソースの認証情報が上書きされないよう にするには、false に設定します。リポジトリソースの認証情報を上書きするには、true に 設定します。デフォルト値は true です。
- token: 必須値。GitHub または GitHub Enterprise Server の場合、これは個人用アクセストークンです。Bitbucket の場合、これは個人用アクセストークンまたはアプリパスワードです。認証タイプが CODECONNECTIONS の場合、これは接続 ARN です。認証タイプが SECRETS_MANAGER の場合、これはシークレット ARN です。
- username: オプションの値。このパラメーターは、GitHub および GitHub エンタープライズ サーバーソースプロバイダーでは無視されます。
- アプリパスワードまたはアクセストークンを使用してアカウントに接続するには、ステップ1 で保存した import-source-credentials.json ファイルが含まれるディレクトリに切り替 え、もう一度 import-source-credentials コマンドを実行します。

aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json

JSON 形式のデータが、Amazon リソースネーム (ARN) を持つ出力に表示されます。

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

Note

同じサーバータイプと認証タイプを持つ import-source-credentials コマンドを 2 回目に 実行した場合、保存されたアクセストークンが更新されます。 アカウントがアプリパスワードに接続されたら、CodeBuild を使用して create-project プロ ジェクトを作成できます。詳細については、「<u>ビルドプロジェクトの作成 (AWS CLI)</u>」を参照 してください。

 接続されたアプリパスワードまたはアクセストークンを表示するには、list-source-credentials コ マンドを実行します。

```
aws codebuild list-source-credentials
```

JSON 形式 sourceCredentialsInfos オブジェクトが出力に表示されます。

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
            "serverType": "BITBUCKET",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObjectには、接続されたソース認証情報のリストが含まれています。

- authTypeは、認証情報により使用される認証のタイプです。これ は、OAUTH、BASIC_AUTH、PERSONAL_ACCESS_TOKEN、CODECONNECTIONS、または SECRETS_MANAGERです。
- serverType は、ソースプロバイダーのタイプです。これ は、GITHUB、GITHUB_ENTERPRISE、BITBUCKET、GITLAB、または GITLAB_SELF_MANAGED です。
- arn は、トークンの ARN です。
- 4. ソースプロバイダから切断して、そのアプリパスワードまたはアクセストークンを削除するに は、その ARN を使用して delete-source-credentials コマンドを実行します。

aws codebuild delete-source-credentials --arn arn-of-your-credentials

削除された認証情報の ARN とともに JSON 形式のデータが返されます。

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

Bitbucket OAuth アプリ

OAuth を使用して Bitbucket に接続 (コンソール)

コンソールを使用して、OAuth アプリでプロジェクトを Bitbucket に接続するには、プロジェクトを 作成するときに以下の操作を実行します。詳細については、「<u>ビルドプロジェクトの作成 (コンソー</u> ル)」を参照してください。

- 1. [ソースプロバイダー] で、[Bitbucket] を選択します。
- 2. [認証情報] で、次のいずれかを実行します。
 - アカウント認証情報を使用して、アカウントのデフォルトのソース認証情報をすべてのプロジェクトに適用することを選択します。
 - a. Bitbucket に接続していない場合は、アカウント認証情報の管理を選択します。
 - b. 認証情報タイプで、OAuth アプリを選択します。
 - サービスにアカウントレベルの認証情報を使用することを選択した場合は、トークンの保存 に使用するサービスを選択し、以下を実行します。
 - a. Secrets Manager を使用する場合は、既存のシークレット接続を使用するか、新しい シークレットを作成するかを選択し、保存を選択します。新しいシークレットの作成方 法の詳細については、「<u>Secrets Manager シークレットにトークンを作成して保存</u>」を 参照してください。
 - b. CodeBuild の使用を選択した場合は、保存を選択します。
 - このプロジェクトのオーバーライド認証情報を使用するを選択して、カスタムソース認証情 報を使用してアカウントの認証情報設定を上書きします。
 - a. 入力された認証情報リストから、OAuth アプリのオプションのいずれかを選択します。
 - b. 説明で新しい Oauth アプリトークン接続の作成を選択して、新しい OAuth アプリトー クンを作成することもできます。

承認された OAuth アプリを確認するには、Bitbucket の「<u>Application authorizations</u>」に移動し、AWS CodeBuild(<u>region</u>)という名前のアプリケーションがリストされていることを確認します。

CodeBuild での GitLab アクセス

GitLab では、GitLab 接続を使用してソースプロバイダにアクセスします。

トピック

CodeBuild を GitLab に接続

CodeBuild を GitLab に接続

接続を使用すると、 を使用してサードパーティープロバイダーを AWS リソースに関連付ける設定 を承認および確立できます AWS CodeConnections。サードパーティのリポジトリをビルドプロジェ クトのソースとして関連付けるには、接続を使用します。

CodeBuild で GitLab または GitLab セルフマネージドソースプロバイダを追加するには、次のいずれ かを選択できます。

- CodeBuild コンソールの [ビルドプロジェクトを作成] ウィザードまたは [ソースを編集] ページを 使用して、[GitLab] または [GitLab セルフマネージド] プロバイダオプションを選択します。ソー スプロバイダを追加するには、「GitLab (コンソール) への接続を作成する」を参照してくださ い。このコンソールは、接続リソースの作成に役立ちます。
- 接続リソースを作成するには、CLIを使用します。CLIを使用して接続リソースを作成するには、 「GitLab (CLI) への接続を作成する」を参照してください。

Note

[設定] からデベロッパーツール コンソールを使用して、接続を作成することもできます。[<u>接</u> 続を作成する] を参照してください。

Note

この接続のインストールを GitLab で承認すると、アカウントにアクセスしてデータを処理す るアクセス許可を当社のサービスに付与したものとみなされます。また、アプリケーション をアンインストールすれば、アクセス許可をいつでも取り消すことができます。

GitLab への接続を作成する

このセクションでは、GitLab を CodeBuild に接続する方法について説明します。GitLab 接続の詳細 については、「CodeBuild を GitLab に接続」を参照してください。

開始する前に:

• GitLab でアカウントを作成しておく必要があります。

Note

接続は、接続の作成と承認に使用されたアカウントで所有するリポジトリへのアクセスだ けを提供します。

Note

GitLab で、自分が所有者ロールを持っているリポジトリへの接続を作成すると、その接続 を CodeBuild などのリソースを含むリポジトリで使用できます。グループ内のリポジトリ では、グループの所有者である必要はありません。

ビルドプロジェクトのソースを指定するには、GitLab にリポジトリを作成しておく必要があります。

トピック

- GitLab (コンソール) への接続を作成する
- GitLab (CLI) への接続を作成する

GitLab (コンソール) への接続を作成する

以下のステップを使用して、CodeBuild コンソールを使用して GitLab 内のプロジェクト (リポジト リ) 用に接続を追加します。 Note

アカウントで既存の接続を作成または使用する代わりに、別の AWS アカウントから共有さ れた接続を使用できます。詳細については、<u>AWS 「アカウントとの接続の共有</u>」を参照し てください。

ビルドプロジェクトを作成または編集するには

- 1. CodeBuild コンソールにサインインします。
- 2. 次のいずれかを選択します。
 - [ビルドプロジェクトを作成]を選択します。
 ビルドプロジェクトの作成 (コンソール) の手順に従って最初の画面を完了し、[ソース] セクションの [プロバイダ] で [GitLab] を選択します。
 - 既存のビルドプロジェクトを編集するこを選択します。[編集]、[ソース]の順に選択します。[ソースを編集]ページの[ソースプロバイダ]で、[GitLab]を選択します。
- 3. 次のいずれかを選択します。
 - [接続] で、[デフォルト接続] を選択します。デフォルト接続は、すべてのプロジェクトにデ フォルトの GitLab 接続を適用します。
 - [接続] で、[カスタム接続] を選択します。カスタム接続は、アカウントのデフォルト設定を上 書きするカスタム GitLab 接続を適用します。
- 4. 次のいずれかを行います:
 - [デフォルト接続] または [カスタム接続] でプロバイダへの接続をまだ作成していない場合 は、[新しい GitLab 接続を作成] を選択します。ステップ 5 に進んで、接続を作成します。
 - ・[接続] でプロバイダへの接続を既に作成している場合は、その接続を選択します。ステップ
 10 に進みます。

Note

GitLab 接続が作成される前にポップアップウィンドウを閉じた場合は、ページを更新す る必要があります。

5. GitLab リポジトリへの接続を作成するには、[プロバイダーを選択する] で、[GitLab] を選択しま す。[接続名] に、作成する接続の名前を入力します。 [GitLab に接続] を選択します。

Developer Tools > Connections > Create connection	
Create a connection Info	
Create GitLab connection Info	
Connection name	
► Tags - optional	
	Connect to GitLab

- 6. GitLab のサインインページが表示されたら、認証情報を使用してログインし、[サインイン] を 選択します。
- 7. 初めて接続を承認する場合は、承認ページが表示され、GitLab アカウントにアクセスするための接続の承認を求めるメッセージが表示されます。

[承認] を選択します。

Authorize AWS Connector for GitLab to use your account?

An application called AWS Connector for GitLab is requesting access to your GitLab account. This application was created by Amazon AWS. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.

This application will be able to:

· Access the authenticated user's API

Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.

- Read the authenticated user's personal information
 Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- Read Api

Grants read access to the API, including all groups and projects, the container registry, and the package registry.

• Allows read-only access to the repository Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

Allows read-write access to the repository Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

- 8. ブラウザは接続コンソールページに戻ります。[GitLab 接続設定] の [接続名] に新しい接続が表 示されます。
- 9. [接続]を選択してください。

GitLab 接続が正常に作成されると、上部に成功のバナーが表示されます。

- 10. [ビルドプロジェクトを作成] ページの [デフォルト接続] または [カスタム接続] のドロップダウ ンリストで、接続 ARN がリストされていることを確認します。リストされていない場合は、更 新ボタンを選択して表示します。
- 11. [リポジトリ] で、プロジェクトのパスと名前空間を指定して、GitLab 内のプロジェクトの名前を選択します。例えば、グループレベルのリポジトリの場合は、リポジトリ名を group-name/repository-name の形式で入力します。パスと名前空間の詳細については、<u>https://docs.gitlab.com/ee/api/projects.html#get-single-project</u>で path_with_namespace フィールドを参照してください。GitLab の名前空間の詳細については、<u>https://docs.gitlab.com/ee/user/namespace</u>/を参照してください。

Note

GitLab 内のグループでは、プロジェクトのパスと名前空間を手動で指定する必要が あります。例えば、グループ mygroup 内のリポジトリの名前が myrepo の場合は、 「mygroup/myrepo」と入力します。プロジェクトのパスと名前空間は GitLab の URL で見つけることができます。

12. [ソースバージョン - オプション] で、プルリクエスト ID、ブランチ、コミット ID、コミット ID、タグ、または参照およびコミット ID を入力します。詳細については、「<u>を使用したソース</u> バージョンサンプル AWS CodeBuild」を参照してください。

Note

811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

13. [Git のクローンの深さ - オプション] で、指定されるコミット数で履歴が切り捨てられた浅いク ローンを作成できます。完全クローンを希望する場合には、[Full (完全)] を選択します。 14. [ビルドステータス - オプション] で、ビルドの開始と終了のステータスをソースプロバイダに報告する場合は、[ビルドの開始時と終了時にソースプロバイダにビルドステータスを報告] を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられ たユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き 込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、 「ソースプロバイダーのアクセス」を参照してください。

GitLab (CLI) への接続を作成する

AWS Command Line Interface (AWS CLI)を使用して接続を作成できます。

これを行うには、create-connection コマンドを使用します。

A Important

AWS CLI または を通じて作成された接続 AWS CloudFormation は、デフォルトで PENDINGステータスです。CLI または との接続を作成したら AWS CloudFormation、コン ソールを使用して接続を編集し、ステータスを にしますAVAILABLE。

接続を作成するには

 「デベロッパーツールユーザーガイド」にある「<u>Create a connection to GitLab (CLI)</u>」の手順に 従ってください。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス)が、別のサービス (呼び出し対象サービス)を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで <u>aws:SourceArn</u>および <u>aws:SourceAccount</u> グローバル条件コンテキス トキーを使用して、 がリソースに別のサービス AWS CodeBuild に付与するアクセス許可を制 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。

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には、グローバルコンテキスト条件キーaws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字(*)を使用します。例えば、arn:aws:codebuild:*:123456789012:*。

aws : SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合 は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

aws:SourceArn の値は CodeBuild プロジェクトの ARN でなければなりません。

次の例では、CodeBuild 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                     "aws:SourceArn": "arn:aws:codebuild:region-ID:account-
ID:project/project-name"
                }
            }
        }
    ]
}
```

高度なトピック

このセクションでは、経験豊富な AWS CodeBuild ユーザーに役立ついくつかの高度なトピックを示 します。

トピック

- ユーザーに CodeBuild とのやり取りを許可
- CodeBuild が他の AWS のサービスとやり取りすることを許可
- カスタマーマネージドキーを使用してビルド出力を暗号化
- を使用して CodeBuild を操作する AWS CLI
- のコマンドラインリファレンス AWS CodeBuild
- ・ AWS SDKs とツールのリファレンス AWS CodeBuild
- AWS SDK でのこのサービスの使用
- AWS CodeBuild エンドポイントを指定する
- AWS CodeBuild で AWS CodePipeline を使用してコードをテストし、ビルドを実行する
- Codecov AWS CodeBuild でを使用する
- Jenkins AWS CodeBuild でを使用する
- サーバーレスアプリケーションで AWS CodeBuild を使用する
- AWS CodeBuild for Windows のサードパーティー通知
- CodeBuild 条件キーを IAM サービスロール変数として使用してビルドアクセスを制御する

ユーザーに CodeBuild とのやり取りを許可

AWS CodeBuild 「」のステップに従って <u>コンソールを使用した開始方法</u>に初めてアクセスする場合、このトピックの情報は不要である可能性が高くなります。ただし、CodeBuild を引き続き使用すると、組織内の他のユーザーやグループに CodeBuild とやり取りする機能を付与することが必要になる場合があります。

IAM ユーザーまたはグループが とやり取りできるようにするには AWS CodeBuild、CodeBuild への アクセス許可を付与する必要があります。このセクションでは、IAM コンソールまたは AWS CLIで これを行う方法について説明します。

AWS ルートアカウント (非推奨) または AWS アカウントの管理者ユーザーを使用して CodeBuild に アクセスする場合は、これらの指示に従う必要はありません。 AWS ルートアカウントと管理者ユーザーの詳細については、<u>「 ユーザーガイド」の AWS アカウン</u> <u>ト 「ルートユーザー</u>」と<u>「最初の AWS アカウント ルートユーザーとグループの作成</u>」を参照して ください。

IAM グループまたはユーザーに CodeBuild アクセス許可を追加するには (コンソール)

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

次のいずれか AWS Management Console を使用して、 に既にサインインしている必要があり ます。

- AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「<u>AWS</u> アカウント ルートユーザー」を参照してください。
- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- ・以下の最小アクションセットを実行するアクセス許可を持つ AWS アカウントのユーザー。

```
iam:AttachGroupPolicy
iam:AttachUserPolicy
iam:CreatePolicy
iam:ListAttachedGroupPolicies
iam:ListAttachedUserPolicies
iam:ListGroups
iam:ListPolicies
iam:ListUsers
```

詳細については、ユーザーガイドの「IAM ポリシーの概要」を参照してください。

- 2. ナビゲーションペインで、ポリシー を選択してください。
- IAM グループまたは IAM ユーザーにアクセス AWS CodeBuild 許可のカスタムセットを追加す るには、この手順のステップ 4 に進みます。

IAM グループや IAM ユーザーにデフォルトの CodeBuild アクセス許可セットを追加するには、 [Policy Type]、[AWS Managed] の順に選択し、以下の操作を行います。

CodeBuild へのフルアクセス許可を追加するには、[AWSCodeBuildAdminAccess] という名前のボックスを選択し、[ポリシーアクション]、[アタッチ] の順に選択します。対象の IAM グループやユーザーの横にあるボックスを選択し、[Attach Policy] (ポリシーのアタッチ) を選択します。AmazonS3ReadOnlyAccess ポリシーおよび IAMFullAccess ポリシーに対して、この操作を繰り返します。

- ビルドプロジェクトの管理を除くすべてについて CodeBuild へのアクセス許可に追加する には、[AWSCodeBuildDeveloperAccess] という名前のボックスを選択し、[Policy Actions] (ポリシーアクション)、[Attach] (アタッチ) の順に選択します。対象の IAM グループや ユーザーの横にあるボックスを選択し、[Attach Policy] (ポリシーのアタッチ) を選択しま す。AmazonS3ReadOnlyAccess ポリシーに対して、この操作を繰り返します。
- CodeBuild への読み取り専用アクセス許可を追加するには、[AWSCodeBuildReadOnlyAccess] という名前のボックスを選択します。対象の IAM グループやユーザーの横にあるボックスを 選択し、[Attach Policy] (ポリシーのアタッチ)を選択します。AmazonS3ReadOnlyAccess ポ リシーに対して、この操作を繰り返します。

これで、IAM グループまたはユーザーに CodeBuild へのデフォルトのアクセス許可セットが追加されました。この手順の残りの手順をスキップします。

- 4. [ポリシーの作成]を選択します。
- 5. [Create Policy] ページで、[Create Your Own Policy] の横にある [Select] を選択します。
- 6. [ポリシーの確認] ページの [ポリシー名] に、ポリシーの名前 (CodeBuildAccessPolicy など) を入力します。別の名前を使用する場合は、この手順全体でそれを使用してください。
- 7. [ポリシードキュメント] に、次のように入力し、[ポリシーの作成] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID:role/role-name"
    },
    {
```

```
"Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

このポリシーは、すべての CodeBuild アクションと、潜在的に多数の AWS リソースへ のアクセスを許可します。アクセス許可を特定の CodeBuild アクションに限定するに は、CodeBuild ポリシーステートメントの codebuild:* の値を変更します。詳細に ついては、「<u>Identity and Access Management</u>」を参照してください。特定の AWS リ ソースへのアクセスを制限するには、Resource オブジェクトの値を変更します。詳細 については、「<u>Identity and Access Management</u>」を参照してください。

8. ナビゲーションペインで、[Groups] または [Users] を選択します。

- 9. グループまたはユーザーのリストで、CodeBuild アクセス許可を追加する IAM グループまたは IAM ユーザーの名前を選択します。
- 10. グループの場合は、グループ設定ページの [アクセス許可] タブで [管理ポリシー] を展開し、[ポ リシーのアタッチ] を選択します。

ユーザーの場合は、ユーザー設定ページの [Permissions] タブで、[Add permissions] を選択します。

11. グループの場合は、[Attach Policy] (ポリシーのアタッチ) ページで [CodeBuildAccessPolicy]、[Attach Policy] (ポリシーのアタッチ) の順に選択します。

ユーザーの場合は、[Add permissions] (アクセス許可の付与) ページで [Attach existing policies directly] (既存のポリシーを直接アタッチ) を選択します。[CodeBuildAccessPolicy] を選択し、 [Next: Reivew] (次のステップ: 確認)、[Add permissions] (アクセス権限の追加) の順にクリック します。

IAM グループまたはユーザーに CodeBuild アクセス許可を追加するには (AWS CLI)

- 前の手順で説明したように、IAM エンティティの1つに対応する AWS アクセスキーと AWS シークレットアクセスキー AWS CLI で が設定されていることを確認します。詳細について は、<u>AWS Command Line Interfaceユーザーガイド</u>の「AWS Command Line Interface のセット アップ」を参照してください。
- IAM グループまたは IAM ユーザーにアクセス AWS CodeBuild 許可のカスタムセットを追加す るには、この手順のステップ 3 に進みます。

IAM グループまたは IAM ユーザーに、CodeBuild アクセス許可のデフォルトセットを追加する には以下を実行します。

IAM グループまたはユーザーのどちらにアクセス許可を追加するかに応じて、以下のいずれかのコマンドを実行します。

aws iam attach-group-policy --group-name group-name --policy-arn policy-arn

aws iam attach-user-policy --user-name user-name --policy-arn policy-arn

コマンドは 3 回実行する必要があります。group-name または user-name は IAM グループ名ま たはユーザー名に置き換え、policy-arn は 1 回ごとに以下の各ポリシー Amazon リソースネーム (ARN) に置き換えてください。
- CodeBuild にフルアクセス許可を追加するには、以下のポリシー ARN を使用します。
 - arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
 - arn:aws:iam::aws:policy/IAMFullAccess
- ビルドプロジェクトの管理以外のすべてに対して CodeBuild にアクセス許可を追加するには、次のポリシー ARN を使用します。
 - arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
- CodeBuild に読み取り専用アクセス許可を追加するには、以下のポリシー ARN を使用しま す。
 - arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess

これで、IAM グループまたはユーザーに CodeBuild へのデフォルトのアクセス許可セットが追加されました。この手順の残りの手順をスキップします。

 AWS CLI がインストールされているローカルワークステーションまたはインスタンスの空の ディレクトリに、 put-group-policy.json または という名前のファイルを作成しますputuser-policy.json。別のファイル名を使用する場合は、この手順全体でそれを使用してくだ さい。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
```

```
],
      "Resource": "arn:aws:iam::account-ID:role/role-name"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    Ł
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

このポリシーは、すべての CodeBuild アクションと、潜在的に多数の AWS リソースへ のアクセスを許可します。アクセス許可を特定の CodeBuild アクションに限定するに は、CodeBuild ポリシーステートメントの codebuild:* の値を変更します。詳細に ついては、「<u>Identity and Access Management</u>」を参照してください。特定の AWS リ ソースへのアクセスを制限するには、関連するResourceオブジェクトの値を変更しま す。詳細については、「<u>Identity and Access Management</u>」または特定の AWS サービ スのセキュリティドキュメントを参照してください。

ファイルを保存したディレクトリに移動し、以下のいずれかのコマンドを実行します。CodeBuildGroupAccessPolicy および CodeBuildUserAccessPolicy に異なる値を使用できます。異なる値を使用する場合は、ここでそれらを使用してください。

IAM グループの場合:

aws iam put-group-policy --group-name group-name --policy-name CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json

ユーザーの場合:

aws iam put-user-policy --user-name user-name --policy-name CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json

前述のコマンドで、group-name または user-name は、対象の IAM グループまたはユーザーの 名前に置き換えます。

CodeBuild が他の AWS のサービスとやり取りすることを許可

AWS CodeBuild 「」のステップに従って <u>コンソールを使用した開始方法</u>に初めてアクセスする場合、このトピックの情報は不要である可能性が高くなります。ただし、CodeBuild を引き続き使用するときは、CodeBuild が他の AWS サービスとやり取りすることを許可するなどの操作が必要になる場合があります。

CodeBuild がユーザーに代わって依存 AWS サービスとやり取りできるようにするには、 AWS CodeBuild サービスロールが必要です。CodeBuild または AWS CodePipeline コンソールを使用し て、CodeBuild サービスロールを作成できます。詳細については、以下を参照してください。

- ビルドプロジェクトの作成 (コンソール)
- CodeBuild を使用するパイプラインを作成する (CodePipeline コンソール)
- CodeBuild ビルドアクションをパイプラインに追加する (CodePipeline コンソール)
- ・ビルドプロジェクトの設定の変更 (コンソール)

これらのコンソールを使用する予定がない場合のために、このセクションでは、IAM コンソールま たは AWS CLIを使用して CodeBuild サービスロールを作成する方法について説明します。

A Important

CodeBuild は、ユーザーのために実行されるすべての操作でサービスロールを使用します。 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<u>最小特権</u>を付与することを確 認します。

このページで説明されているサービスロールには、CodeBuild を使用するのに必要な最小権 限を付与するポリシーが含まれています。ユースケースに応じて、さらに許可を追加する必 要がある場合があります。

CodeBuild サービスロールを作成するには (コンソール)

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

次のいずれかを使用して、コンソールに既にサインインしている必要があります。

- AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「<u>AWS</u> <u>アカウント ルートユーザー</u>」を参照してください。
- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- ・以下の最小アクションセットを実行するアクセス許可を持つ AWS アカウントのユーザー。

```
iam:AddRoleToInstanceProfile
iam:AttachRolePolicy
iam:CreateInstanceProfile
iam:CreatePolicy
iam:CreateRole
iam:GetRole
iam:ListAttachedRolePolicies
iam:ListPolicies
iam:ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy
```

詳細については、ユーザーガイドの「IAM ポリシーの概要」を参照してください。

{

- 2. ナビゲーションペインで、ポリシーを選択してください。
- 3. [ポリシーの作成]を選択します。
- 4. [Create Policy] ページで、[JSON] を選択します。
- 5. [JSON ポリシー] に、次のように入力し、[ポリシーの確認] を選択します。

```
"Version": "2012-10-17",
"Statement": [
 {
    "Sid": "CloudWatchLogsPolicy",
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CodeCommitPolicy",
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull"
    ],
    "Resource": "*"
  },
  {
    "Sid": "S3GetObjectPolicy",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": "*"
 },
  {
    "Sid": "S3PutObjectPolicy",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject"
    ],
    "Resource": "*"
```

```
},
    {
      "Sid": "ECRPullPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRAuthPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
 ]
}
```

Note

このポリシーには、潜在的に多数の AWS リソースへのアクセスを許可するステートメ ントが含まれています。特定の AWS リソース AWS CodeBuild へのアクセスを制限す るには、 Resource 配列の値を変更します。詳細については、 AWS サービスのセキュ リティドキュメントを参照してください。

[ポリシーの確認] ページで、[ポリシー名] にポリシー名 (CodeBuildServiceRolePolicy など) を入力し、[ポリシーの作成] を選択します。

- 7. ナビゲーションペインで [Roles (ロール)]を選択します。
- 8. [Create role] を選択します。
- 9. [ロールの作成] ページで、[AWS のサービス] が選択された状態で、[CodeBuild]、[次の手順: ア クセス許可] の順に選択します。
- 10. [Attach permissions policies (アクセス権限ポリシーをアタッチする)] ページで、 [CodeBuildServiceRolePolicy]、[Next: Review (次へ: 確認)] の順に選択します。
- 11. [Create role and review (ロールの作成と確認)] ページで、[ロール名] にロールの名前 (**CodeBuildServiceRole** など) を入力し、[ロールの作成] を選択します。

CodeBuild サービスロールの作成 (AWS CLI)

- 前の手順で説明したように、IAM エンティティの1つに対応する AWS アクセスキーと AWS シークレットアクセスキー AWS CLI で が設定されていることを確認します。詳細について は、<u>AWS Command Line Interfaceユーザーガイド</u>の「AWS Command Line Interface のセット アップ」を参照してください。
- AWS CLI がインストールされているローカルワークステーションまたはインスタンスの空の ディレクトリに、 create-role.jsonと という名前の2つのファイルを作成しますputrole-policy.json。別のファイル名を選択した場合は、この手順全体でそれを使用してくだ さい。

create-role.json:

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

}

Note

<u>「混乱した代理」問題</u>に対して自分を守るために aws:SourceAccount および aws:SourceArn 条件キーを使用することをお勧めします。例えば、前述の信頼ポリ シーを次の条件ブロックで編集できます。aws:SourceAccount は CodeBuild プロ ジェクトの所有者で、aws:SourceArn は CodeBuild プロジェクトの ARN です。

サービスロールを AWS アカウントに制限する場合、 create-role.json は次のようになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                     "aws:SourceAccount": [
                        "account-ID"
                     ]
                }
            }
        }
    ]
}
```

サービスロールを特定の CodeBuild プロジェクトに制限する場合、create-role.json は次のようになります。

Note

CodeBuild プロジェクトの名前が不明である、または名前を決定しておらず、特定の ARN パターンに信頼ポリシーの制限が必要な場合は、ARN の該当部分をワイルドカー ド (*) に置き換えることができます。プロジェクトを作成した後は、信頼ポリシーを更 新できます。

put-role-policy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
```

```
"Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

このポリシーには、潜在的に多数の AWS リソースへのアクセスを許可するステートメ ントが含まれています。特定の AWS リソース AWS CodeBuild へのアクセスを制限す るには、 Resource 配列の値を変更します。詳細については、 AWS サービスのセキュ リティドキュメントを参照してください。 上記のファイルを保存したディレクトリに移動し、以下の2つのコマンドをこの順番で1つず つ実行します。CodeBuildServiceRole と CodeBuildServiceRolePolicy には異なる値 を使用する場合は、ここでそれらを使用してください。

aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document
file://create-role.json

aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json

カスタマーマネージドキーを使用してビルド出力を暗号化

AWS CodeBuild 「」のステップに従って <u>コンソールを使用した開始方法</u>に初めてアクセスする場合、このトピックの情報は不要である可能性が高くなります。ただし、CodeBuild を引き続き使用す ると、ビルドアーティファクトの暗号化などが必要になる場合があります。

がビルド出力アーティファクトを暗号化 AWS CodeBuild するには、KMS キーにアクセスする必要 があります。デフォルトでは、CodeBuild は AWS アカウントの Amazon S3 AWS マネージドキー に を使用します。

を使用しない場合は AWS マネージドキー、カスタマーマネージドキーを自分で作成して設定する必要があります。このセクションでは、IAM コンソールを使用してこれを行う方法を説明します。

カスタマー管理のキーの詳細については、AWS KMS デベロッパーガイドの「<u>AWS Key</u> Management Service の概念およびキーの作成」を参照してください。

CodeBuild で使用するカスタマー管理のキーを設定するには、AWS KMS 開発者ガイドの「<u>キーポ</u> <u>リシーの変更</u>」の手順に従ってください。次に、キーポリシーに以下のステートメント (###BEGIN ADDING STATEMENTS HERE### と ###END ADDING STATEMENTS HERE### の間)を追加しま す。省略記号(...)は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立 ちます。ステートメントを削除しないでください、また、これらの省略記号をキーポリシーに入力し ないでください。

```
{
    "Version": "2012-10-17",
    "Id": "...",
    "Statement": [
    ### BEGIN ADDING STATEMENTS HERE ###
    {
```

```
"Sid": "Allow access through Amazon S3 for all principals in the account that are
authorized to use Amazon S3",
     "Effect": "Allow",
     "Principal": {
       "AWS": "*"
     },
     "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
     ],
     "Resource": "*",
     "Condition": {
       "StringEquals": {
         "kms:ViaService": "s3.region-ID.amazonaws.com",
         "kms:CallerAccount": "account-ID"
       }
     }
   },
   {
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::account-ID:role/CodeBuild-service-role"
     },
     "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
     ],
     "Resource": "*"
   },
   ### END ADDING STATEMENTS HERE ###
   {
     "Sid": "Enable IAM User Permissions",
     . . .
   },
   {
     "Sid": "Allow access for Key Administrators",
     . . .
   },
```

```
{
    "Sid": "Allow use of the key",
    ...
    },
    {
        "Sid": "Allow attachment of persistent resources",
        ...
     }
  ]
}
```

- *region-ID*は、CodeBuildに関連付けられた Amazon S3 バケットがある AWS リージョンの ID を表します (例: us-east-1)。
- ・##### IDは、AWS カスタマー管理のキーを所有する アカウントのID を表します。
- CodeBuild-service-role は、このトピックの前半で作成または識別した CodeBuild サービス ロールの名前を表します。
 - Note

IAM コンソールを使用してカスタマーマネージドキーを作成または設定するには、まず次の いずれか AWS Management Console を使用して にサインインする必要があります。

- AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの 「アカウントルートユーザー」を参照してください。
- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- カスタマーマネージドキーを作成または変更するアクセス許可を持つ AWS アカウントのユーザー。詳細については、「AWS KMS デベロッパーガイド」の「AWS KMS コン ソールを使用するために必要なアクセス許可」を参照してください。

を使用して CodeBuild を操作する AWS CLI

AWS CodeBuild 「」のステップに従って <u>コンソールを使用した開始方法</u>に初めてアクセスする場 合、このトピックの情報は不要である可能性が高くなります。ただし、CodeBuild を引き続き使用す るときは、CodeBuild コンソール、CodePipeline CodePipeline コンソール、または AWS SDKs の代 わりに (または追加で)、ユーザーが を使用して CodeBuild と AWS CLI やり取りすることを許可す るなどの操作が必要になる場合があります。

をインストールして設定するには AWS CLI、 AWS Command Line Interface ユーザーガイド<u>の「 の</u> セットアップ AWS Command Line Interface」を参照してください。

をインストールしたら AWS CLI、次のタスクを実行します。

1. 次のコマンドを実行して、 AWS CLI のインストールが CodeBuild をサポートしているかどうか を確認します。

aws codebuild list-builds

成功すると、次のような情報が出力に表示されます。

空の角括弧は、まだビルドを実行していないことを示しています。

 エラーが出力された場合は、現在のバージョンの AWS CLI をアンインストールしてから、最新 バージョンをインストールする必要があります。詳細については、<u>AWS CLIユーザーガイド</u>の 「<u>AWS Command Line Interfaceのアンインストール</u>」および「AWS Command Line Interface のインストール」を参照してください。

のコマンドラインリファレンス AWS CodeBuild

AWS CLI には、 を自動化するためのコマンドが用意されています AWS CodeBuild。このトピック の情報は、<u>AWS Command Line Interface ユーザーガイド</u>と <u>AWS CodeBuildのAWS CLI リファレン</u> <u>ス</u>の補足として使用します。

お探しのものではありませんか。 AWS SDKs「」を参照してください<u>AWS SDKsとツールのリファ</u> <u>レンス</u>。 CodeBuild

このトピックの情報を使用するには、「」で説明されているように、 をインストール AWS CLI し、CodeBuild で使用するように設定しておく必要があります<u>を使用して CodeBuild を操作する</u> AWS CLI。 を使用して CodeBuild のエンドポイントを指定する AWS CLI には、「」を参照してください<u>AWS</u> CodeBuild エンドポイントを指定する (AWS CLI)。

次のコマンドでは、CodeBuild のコマンドのリストを取得できます。

aws codebuild help

次のコマンドでは、CodeBuild コマンドに関する情報を取得できます。*command-name* はコマンド 名です。

aws codebuild *command-name* help

CodeBuild のコマンドは以下の通りです。

- batch-delete-builds: CodeBuild の 1 つ以上のビルドを削除します。詳細については、「ビルドの削除 (AWS CLI)」を参照してください。
- batch-get-builds: CodeBuild の複数のビルドに関する情報を取得します。詳細については、 「ビルドの詳細の表示 (AWS CLI)」を参照してください。
- batch-get-projects: 指定された1つ以上のビルドプロジェクトに関する情報を取得します。
 詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」を参照してください。
- create-project: ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの 作成 (AWS CLI)」を参照してください。
- delete-project: ビルドプロジェクトを削除します。詳細については、「ビルドプロジェクトの 削除 (AWS CLI)」を参照してください。
- 1ist-builds: CodeBuild でのビルドの Amazon リソースネーム (ARN) をリスト表示します。詳細については、「ビルド ID の一覧表示 (AWS CLI)」を参照してください。
- list-builds-for-project: 指定されたビルドプロジェクトに関連付けられているビルド ID の リストを取得します。詳細については、「ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)」を参照してください。
- list-curated-environment-images: ビルドに使用できる CodeBuild によって管理される Docker イメージのリストを取得します。詳細については、「<u>CodeBuild に用意されている Docker</u> イメージ」を参照してください。
- list-projects: ビルドプロジェクト名のリストを取得します。詳細については、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。
- ・ start-build: ビルドの実行を開始します。詳細については、「<u>ビルドの実行 (AWS CLI)</u>」を参 照してください。

- stop-build: 停止されたビルドの実行を停止しようとします。詳細については、「ビルドの停止 (AWS CLI)」を参照してください。
- update-project: 指定されたビルドプロジェクトに関する情報を変更します。詳細については、 「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。

AWS SDKs とツールのリファレンス AWS CodeBuild

1 つの AWS SDKs またはツールを使用して自動化するには AWS CodeBuild、次のリソースを参照し てください。

を使用して CodeBuild AWS CLI を実行する場合は、「」を参照してください<u>コマンドラインリファ</u> レンス。

でサポートされている AWS SDKsとツール AWS CodeBuild

次の AWS SDKs とツールは CodeBuild をサポートしています。

- <u>AWS SDK for C++</u>。詳細については、<u>http://sdk.amazonaws.com/cpp/api/LATEST/</u> <u>namespace_aws_1_1_code_build.html</u> SDK for C++ API リファレンスのAWS Aws::CodeBuild 名前 空間のセクションを参照してください。
- <u>AWS SDK for Go</u>。詳細については、AWS SDK for Go API リファレンスの <u>CodeBuild</u> セクション を参照してください。
- <u>AWS SDK for Java</u>。詳細については、<u>AWS SDK for Java API リ</u>
 <u>ファレンス</u>の com.amazonaws.services.codebuild および
 com.amazonaws.services.codebuild.model セクションを参照してください。
- <u>ブラウザでのAWS SDK for JavaScript</u> および <u>Node.js でのAWS SDK for JavaScript</u>。詳細に ついては、「クラス:」を参照してください AWS。SDK for JavaScript API リファレンスの <u>CodeBuild</u> セクション。AWS JavaScript
- <u>AWS SDK for .NET</u>。詳細については、「AWS SDK for .NET API リファレンス」の 「<u>Amazon.CodeBuild</u>」および「<u>Amazon.CodeBuild.Model</u>」名前空間セクションを参照してくだ さい。
- <u>AWS SDK for PHP</u>。詳細については、AWS SDK for PHP API リファレンスの<u>名前空間 Aws</u> <u>\CodeBuild</u> セクションを参照してください。
- <u>AWS SDK for Python (Boto3)</u>。詳細については、Boto 3 ドキュメントの「<u>CodeBuild</u>」セクションを参照してください。

- <u>AWS SDK for Ruby</u>。詳細については、AWS SDK for Ruby API リファレンスの「<u>モジュール</u>: Aws::CodeBuild 」セクションを参照してください。
- <u>AWS Tools for PowerShell</u>。詳細については、<u>AWS CodeBuild Tools for PowerShell Cmdlet リ</u>ファレンスの「AWS」セクションを参照してください。

AWS SDK でのこのサービスの使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。 各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、 コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	<u>AWS CLI コード例</u>
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	<u>AWS SDK for Java コード例</u>
AWS SDK for JavaScript	<u>AWS SDK for JavaScript コード例</u>
AWS SDK for Kotlin	<u>AWS SDK for Kotlin コード例</u>
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	<u>AWS SDK for PHP コード例</u>
AWS Tools for PowerShell	<u>Tools for PowerShell のコード例</u>
AWS SDK for Python (Boto3)	<u>AWS SDK for Python (Boto3) コード例</u>
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	<u>AWS SDK for Rust コード例</u>
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	<u>AWS SDK for Swift コード例</u>

このサービスに固有の例については、「<u>SDK を使用した CodeBuild のコード例 AWS SDKs</u>」を参照 してください。

🚯 可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィード バックを送信)] リンクから、コードの例をリクエストしてください。

AWS CodeBuild エンドポイントを指定する

AWS Command Line Interface (AWS CLI) またはいずれかの AWS SDKs を使用して、 が使用する エンドポイントを指定できます AWS CodeBuild。エンドポイントは、CodeBuild が使用可能なリー ジョンごとに存在します。リージョンのエンドポイントに加えて、4 つのリージョンに連邦情報処理 標準 (FIPS) エンドポイントがあります。FIPS エンドポイントの詳細については、「<u>FIPS 140-2 の</u> 概要」を参照してください。

エンドポイントの指定はオプションです。使用するエンドポイントを CodeBuild に明示的に指示 しない場合、サービスは AWS アカウントが使用するリージョンに関連付けられたエンドポイント を使用します。CodeBuild では、FIPS エンドポイントがデフォルトで使用されることはありませ ん。FIPS エンドポイントを使用するには、次のいずれかのメソッドを使用して、CodeBuild と関連 付ける必要があります。

Note

エイリアスまたはリージョン名を使用して、 AWS SDK を使用してエンドポイントを指定で きます。を使用する場合は AWS CLI、完全なエンドポイント名を使用する必要があります。

CodeBuild で使用可能なエンドポイントについては、「<u>CodeBuild のリージョンとエンドポイント</u>」 を参照してください。

トピック

- <u>AWS CodeBuild エンドポイントを指定する (AWS CLI)</u>
- AWS CodeBuild エンドポイントを指定する (AWS SDK)

AWS CodeBuild エンドポイントを指定する (AWS CLI)

を使用して、CodeBuild コマンドの --endpoint-url引数を使用して AWS CodeBuild 、 がアクセ スされるエンドポイント AWS CLI を指定できます。たとえば、「米国東部 (バージニア北部) リー ジョン」で連邦情報処理標準 (FIPS) エンドポイントを使用して、プロジェクトビルド名のリストを 取得するには、このコマンドを実行します。

aws codebuild list-projects --endpoint-url https://codebuild-fips.useast-1.amazonaws.com

エンドポイントの先頭に https:// を追加します。

--endpoint-url AWS CLI 引数は、すべての AWS サービスで使用できます。この引数およびその 他の AWS CLI 引数の詳細については、AWS CLI 「 コマンドリファレンス」を参照してください。

AWS CodeBuild エンドポイントを指定する (AWS SDK)

AWS SDK を使用して、 AWS CodeBuild がアクセスされるエンドポイントを指定できます。この例 では <u>AWS SDK for Java</u>を使用していますが、他の AWS SDKs を使用してエンドポイントを指定で きます。

AWSCodeBuild クライアントを作成する場合は、withEndpointConfiguration メソッドを使用 します。以下の形式を使用します。

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
    "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

AWSCodeBuildClientBuilder については、「<u>AWSCodeBuildClientBuilder クラス</u>」を参照して ください。

withCredentialsの認証情報のタイプは、AWSCredentialsProviderを使用する必要があります。詳細については、AWS「認証情報の使用」を参照してください。

エンドポイントの先頭に https:// を追加しないでください。

非 FIPS エンドポイントを指定する場合は、実際のエンドポイントではなくリージョンを使用しま す。例えば、米国東部 (バージニア北部) リージョンのエンドポイントを指定するには、完全なエン ドポイント名 (codebuild.us-east-1.amazonaws.com) ではなく、us-east-1 を使用できま す。

FIPS エンドポイントを指定する場合は、エイリアスを使用して、コードを簡素化することができま す。FIPS エンドポイントのみ、エイリアスが含まれます。他のエンドポイントは、リージョンまた は完全名を使用して指定する必要があります。

利用できる 4 つの FIPS エンドポイントごとのエイリアスを以下のテーブルに示します。

リージョン 名	リージョン	エンドポイント	エイリアス
米国東部 (バージニ ア北部)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1- fips
米国東部 (オハイオ)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2- fips
米国西部 (北カリ フォルニア)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1- fips
米国西部 (オレゴン)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2- fips

エイリアスを使用して、米国西部 (オレゴン) リージョンの FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

米国東部 (バージニア北部) リージョンの非 FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
    "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

アジアパシフィック (ムンバイ) リージョンの非 FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
    "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

AWS CodeBuild で AWS CodePipeline を使用してコードをテスト し、ビルドを実行する

AWS CodePipeline を使用してコードをテストし、 でビルドを実行することで、リリースプロセス を自動化できます AWS CodeBuild。

次の表に示しているのは、タスクとその実行に使用できるメソッドです。これらのタスクを AWS SDK で達成する方法については、このトピックの対象外です。

タスク	使用可能なアプ ローチ	このトピックで説明するアプローチ
CodeBuild でビ ルドを自動化す る CodePipeline を使用して、継 続的な配信 (CD)	・ CodePipeline コンソール ・ AWS CLI ・ AWS SDKs	 CodePipeline コンソールの使用 AWS CLIの使用 このトピックの情報は、AWS SDK を使用するように 調整できます。詳細については、AWS CodePipeline の API リファレンス の CreatePipeline またはア

タスク	使用可能なアプ ローチ	このトピックで説明するアプローチ
パイプラインを 作成する		マゾン ウェブ サービスのツールの <u>SDK</u> セクションで プログラミング言語の create-pipeline アクショ ンドキュメントを参照してください。
既存の CodePipeline の パイプラインに CodeBuild での テストおよびビ ルドの自動化を 追加する	 CodePipeline コンソール AWS CLI AWS SDKs 	 CodePipeline コンソールを使用してビルドの自動化を 追加する CodePipeline コンソールを使用してテストの自動化を 追加する では AWS CLI、このトピックの情報を適応させ て、CodeBuild ビルドアクションまたはテストアク ションを含むパイプラインを作成できます。詳細につ いては、AWS CodePipeline ユーザーガイドのパイプ ラインを編集する (AWS CLI) およびCodePipeline のパ イプライン構造リファレンスを参照してください。 このトピックの情報は、AWS SDK を使用するように 調整できます。詳細については、AWS CodePipeline の API リファレンス の UpdatePipeline またはア マゾン ウェブ サービスのツールの SDK セクション からプログラミング言語の update-pipeline アク ションドキュメントを参照してください。

トピック

- 前提条件
- CodeBuild を使用するパイプラインを作成する (CodePipeline コンソール)
- CodeBuild を使用するパイプラインの作成 (AWS CLI)
- ・ <u>CodeBuild ビルドアクションをパイプラインに追加する (CodePipeline コンソール)</u>
- ・ CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)

前提条件

1. ビルドを計画するの質問に答えます。

 AWS ルートアカウントまたは管理者ユーザーの代わりに ユーザーを使用して CodePipeline に アクセスする場合は、という名前の管理ポリシーAWSCodePipelineFullAccessをユーザー (またはユーザーが属する IAM グループ) にアタッチします。 AWS ルートアカウントの使用は お勧めしません。このポリシーは、CodePipeline でパイプラインを作成するためのアクセス許 可をユーザーに付与します。詳細については、ユーザーガイドの「管理ポリシーをアタッチす る」を参照してください。

Note

ポリシーをユーザー (またはユーザーが属する IAM グループ) にアタッチする IAM エン ティティは、ポリシーをアタッチするために IAM でのアクセス許可を持っている必要が あります。詳細については、ユーザーガイドの「<u>IAM ユーザー、グループ、および認証</u> <u>情報を管理するためのアクセス許可の委任</u>」を参照してください。

 AWS アカウントに CodePipeline のサービスロールがまだない場合は、作成しま す。CodePipeline はこのサービスロールを使用して、 AWS CodeBuildユーザーに代わって を含 む他の AWS サービスとやり取りします。たとえば、 を使用して CodePipeline サービスロール AWS CLI を作成するには、IAM create-role コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role
    --assume-role-policy-document '{"Version":"2012-10-17","Statement":
    {"Effect":"Allow","Principal":
    {"Service":"codepipeline.amazonaws.com"},"Action":"sts:AssumeRole"}}'
```

Windows の場合:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-
role-policy-document "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":
\"Allow\",\"Principal\":{\"Service\":\"codepipeline.amazonaws.com\"},\"Action\":
\"sts:AssumeRole\"}}"
```

Note

この CodePipeline のサービスロールを作成する IAM エンティティは、サービスロール を作成するために IAM のアクセス許可を持っている必要があります。 CodePipeline サービスロールを作成した後、または既存のサービスロールを識別した後、AWS CodePipeline ユーザーガイドのデフォルトの CodePipeline サービスロールポリシーを確認す <u>る</u>で説明されているように、デフォルトの CodePipeline サービスロールポリシーをサービス ロールに追加する必要があります (ロールのポリシーの一部になっていない場合)。

Note

この CodePipeline サービスロールのポリシーを追加する IAM エンティティは、サービ スロールポリシーをサービスロールに追加するために IAM のアクセス許可を持っている 必要があります。

 CodeCommit、Amazon S3、Bitbucket、GitHub など、CodeBuild と CodePipeline でサポートさ れているリポジトリタイプにソースコードを作成してアップロードします。ソースコードには buildspec ファイルが含まれている必要がありますが、このトピックの後半でビルドプロジェク トを定義するときにそのファイルを宣言できます。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

▲ Important

パイプラインを使用してビルド済みのソースコードをデプロイする場合、ビルド出力 アーティファクトには、使用するデプロイシステムとの互換性が必要です。

 については AWS OpsWorks、AWS OpsWorks「ユーザーガイド」の「アプリケー ションソース」および「での CodePipelineの使用 AWS OpsWorks」を参照してくだ さい。

CodeBuild を使用するパイプラインを作成する (CodePipeline コンソール)

CodeBuild を使用してソースコードをビルドおよびデプロイするパイプラインを作成するには、次の 手順を実行します。

ソースコードのみをテストするパイプラインを作成するには、以下の操作を行います。

 次の手順を使用してパイプラインを作成し、パイプラインからビルドステージとベータステージ を削除します。次に、このトピックの「<u>CodeBuild テストアクションをパイプラインに追加する</u> (<u>CodePipeline コンソール</u>」の手順を使用して、CodeBuild を使用するテストアクションをパイプ ラインに追加します。 このトピックの他の手順のいずれかを使用してパイプラインを作成した後、このトピックの 「<u>CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)</u>」の手順を 使用して CodeBuild を使用するテストアクションをパイプラインに追加します。

CodePipeline でパイプライン作成ウィザードを使用して、CodeBuild を使用するパイプラインを作 成するには

- 1. 以下を使用して にサインイン AWS Management Console します。
 - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカ ウントルートユーザー」を参照してください。
 - AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
 - 以下の最小限のアクションセットを使用するアクセス許可を持つ AWS アカウントのユー ザー。

codepipeline:* iam:ListRoles iam:PassRole s3:CreateBucket s3:GetBucketPolicy s3:GetObject s3:ListAllMyBuckets s3:ListBucket s3:PutBucketPolicy codecommit:ListBranches codecommit:ListRepositories codedeploy:GetApplication codedeploy:GetDeploymentGroup codedeploy:ListApplications codedeploy:ListDeploymentGroups elasticbeanstalk:DescribeApplications elasticbeanstalk:DescribeEnvironments lambda:GetFunctionConfiguration lambda:ListFunctions opsworks:DescribeStacks opsworks:DescribeApps opsworks:DescribeLayers

- 2. AWS CodePipeline コンソールを <u>https://console.aws.amazon.com/codesuite/codepipeline/</u> home://https://https://https://https://https
- AWS リージョンセレクタで、ビルドプロジェクト AWS リソースがある AWS リージョンを選 択します。これは、CodeBuild がサポートされている AWS リージョンである必要があります。 詳細については、「Amazon Web Services 全般のリファレンス」の「<u>AWS CodeBuild</u>」を参照 してください。
- パイプラインを作成する CodePipeline 情報ページが表示されたら、[Create pipeline] (パイプ ラインの作成) を選択します。[Pipelines (パイプライン)] ページが表示された場合は、[Create pipeline (パイプラインの作成)] を選択します。
- [Step 1: Choose pipeline settings (ステップ 1: パイプラインの設定の選択)] ページで、[Pipeline name (パイプライン名)] にパイプラインの名前を入力します (例: CodeBuildDemoPipeline)。別の名前を選択した場合は、この手順全体でそれを使用してください。
- 6. [Role name (ロール名)] として、以下のいずれかの操作を行います。

[New service role (新しいサービスロール)] を選択し、[Role Name (ロール名)] に、新しいサービ スロールの名前を入力します。

[Existing service role] (既存のサービスロール) を選択し、このトピックの前提条件の一部として 作成または特定した CodePipeline サービスロールを選択します。

- 7. [Artifact store (アーティファクトストア)] で、次のいずれかの操作を行います。
 - デフォルトの場所を選択して、パイプラインに選択した AWS リージョンのパイプラインのデ フォルトとして指定された S3 アーティファクトバケットなど、デフォルトのアーティファク トストアを使用します。
 - パイプラインと同じ AWS リージョンに S3 アーティファクトバケットなど、作成した既存の
 アーティファクトストアがある場合は、カスタムロケーションを選択します。

Note

これはパイプラインのソースコードのソースバケットではありません。パイプラインの アーティファクトストアです。S3 バケットなどの個別のアーティファクトストアは、パ イプラインと同じ AWS リージョン内のパイプラインごとに必要です。

8. [Next (次へ)] を選択します。

- 9. [Step 2: Add source stage (ステップ 2: ソースステージの追加)] ページの [ソースプロバイダ] で、次のいずれかの操作を行います。
 - ソースコードの保存先が S3 バケットである場合は、[Amazon S3] を選択します。[バケット] で、ソースコードが含まれている S3 バケットを選択します。[S3 オブジェクトキー] に、 ソースコードを含むファイルの名前 (例: *file-name*.zip) を入力します。[Next (次へ)] を選 択します。
 - ソースコードが AWS CodeCommit リポジトリに保存されている場合は、CodeCommit を選 択します。[Repository name] で、ソースコードが含まれているリポジトリの名前を選択しま す。[ブランチ名] で、ビルドするソースコードのバージョンが含まれているブランチの名前を 選択します。[Next (次へ)] を選択します。
 - ソースコードが GitHub リポジトリに保存されている場合は、[GitHub] を選択します。
 [Connect to GitHub] を選択し、手順に従って GitHub に対して認証します。[Repository] で、
 ソースコードが含まれているリポジトリの名前を選択します。[ブランチ] で、ビルドするソースコードのバージョンが含まれているブランチの名前を選択します。

[Next (次へ)] を選択します。

- 10. [Step 3: Add build stage] (ステップ 3: ビルドステージを追加する) ページで、[Build provider] (ビ ルドプロバイダー) として [CodeBuild] を選択します。
- 11. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロ ジェクトの名前を選択し、この手順の次のステップにスキップします。

新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「<u>ビルドプロジェクトの</u> 作成 (コンソール)」の手順に従ってから、この手順に戻ります。

既存のビルドプロジェクトを選択した場合、ビルド出力アーティファクトの設定がすでに定義 されている必要があります (ただし、CodePipeline によってビルド出力の設定が上書きされま す)。詳細については、「<u>ビルドプロジェクトの設定の変更 (コンソール)</u>」を参照してくださ い。

Important

CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されま す。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通 じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金 されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを 無効にすることをお勧めします。 AWS CodeBuild コンソールで、[Webhook] ボックス をオフにします。詳細については、「<u>ビルドプロジェクトの設定の変更 (コンソール)</u>」 を参照してください。

- 12. [Step 4: Add deploy stage (ステップ 4: デプロイステージの追加)] ページで、次のいずれかの操 作を行います。
 - ビルド出力アーティファクトをデプロイしない場合は、[Skip (スキップ)]を選択し、プロンプトが表示されたら、これを選択したことを確認します。
 - ビルド出力アーティファクトをデプロイする場合は、[Deployment provider (デプロイプロバ イダ)] でデプロイプロバイダを選択し、次にプロンプトに応じて設定を指定します。

[Next (次へ)] を選択します。

- 13. [確認] ページで、選択内容を確認し、[パイプラインの作成] を選択します。
- パイプラインが正常に実行されたら、ビルド出力アーティファクトを取得できます。CodePipeline コンソールにパイプラインを表示した状態で、[Build] (ビルド) アクションで ツールヒントを選択します。[Output artifact] の値をメモします (例: MyAppBuild)。

Note

ビルド出力アーティファクトを取得するには、CodeBuild コンソールのビルドの詳細 ページで [Build artifacts] (ビルドアーティファクト) リンクを選択することもできます。 このページを表示するには、この手順の残りのステップを省略して、「<u>ビルドの詳細の</u> <u>表示 (コンソール)</u>」を参照してください。

- 15. <u>https://console.aws.amazon.com/s3/</u> で Amazon S3 コンソールを開きます。
- 16. バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名は、codepipeline-*region-ID-random-number*の形式に従う必要があります。を使用してCodePipeline get-pipeline コマンド AWS CLIを実行し、バケットの名前を取得できます。*my-pipeline-name*はパイプラインの表示名です。

aws codepipeline get-pipeline --name my-pipeline-name

出力では、pipeline オブジェクトには artifactStore オブジェクトが含まれ、それには、 バケットの名前と location の値が含まれます。

- 17. パイプラインの名前と一致するフォルダを開きます (パイプライン名の長さによってはフォルダ 名が切り詰められている場合があります)。次に、前に書き留めた [出力アーティファクト] の値 と一致するフォルダを開きます。
- 18. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タ イムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操 作できるように、必要に応じて、ファイルに.zip 拡張子を付けます。) ビルド出力アーティ ファクトは、展開されたファイルの内容に含まれます。
- 19. CodePipeline にビルド出力アーティファクトをデプロイするよう指示した場合は、デプロイプ ロバイダの説明を活用してデプロイターゲットのビルド出力アーティファクトを取得します。

CodeBuild を使用するパイプラインの作成 (AWS CLI)

CodeBuild を使用してソースコードをビルドするパイプラインを作成するには、次の手順を実行します。

を使用して、ビルドされたソースコードをデプロイするパイプライン、またはソースコードをテスト するパイプライン AWS CLI を作成するには、AWS CodePipeline 「 ユーザーガイド」の<u>「パイプラ</u> <u>インの編集 (AWS CLI)</u>」および<u>CodePipeline パイプライン構造リファレンス</u>」の手順を適用できま す。

 CodeBuild でビルドプロジェクトを作成または識別します。詳細については、「ビルドプロジェ クトの作成」を参照してください。

A Important

ビルドプロジェクトは、ビルド出力アーティファクトの設定を定義する必要があります (ただし、CodePipeline によって上書きされます)。詳細については、「artifacts」で <u>ビルドプロジェクトの作成 (AWS CLI)</u>の説明を参照してください。

- このトピックで説明されている IAM エンティティのいずれかに対応する AWS アクセスキーと AWS シークレットアクセスキー AWS CLI で が設定されていることを確認します。詳細につい ては、AWS Command Line Interface ユーザーガイドの <u>AWS Command Line Interfaceのセット</u> アップを参照してください。
- パイプラインの構造を表す JSON 形式のファイルを作成します。ファイルに createpipeline.json のような名前を付けます。たとえば、この JSON 形式の構造では、S3 入力バ ケットを参照するソースアクションと CodeBuild を使用するビルドアクションを使用してパイ プラインを作成します。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::<account-id>:role/<AWS-CodePipeline-service-role-
name>",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
              "S3Bucket": "<bucket-name>",
              "S3ObjectKey": "<source-code-file-name.zip>"
            },
            "runOrder": 1
          }
        ]
      },
      {
        "name": "Build",
        "actions": [
          {
            "inputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "name": "Build",
            "actionTypeId": {
              "category": "Build",
              "owner": "AWS",
```

```
"version": "1",
              "provider": "CodeBuild"
            },
            "outputArtifacts": [
              {
                "name": "default"
              }
            ],
            "configuration": {
              "ProjectName": "<build-project-name>"
            },
            "runOrder": 1
          }
        ]
      }
    ],
    "artifactStore": {
      "type": "S3",
      "location": "<CodePipeline-internal-bucket-name>"
    },
    "name": "<my-pipeline-name>",
    "version": 1
 }
}
```

この JSON 形式のデータは以下のようになっています。

- roleArnの値は、前提条件の一部として作成または特定した CodePipeline のサービスロールの ARN と一致する必要があります。
- S3Bucket の S30bjectKey と configuration の値は、ソースコードの保存先が S3 バ ケットであることを前提としています。その他のソースコードのリポジトリタイプの設定につ いては、AWS CodePipeline ユーザーガイドの <u>CodePipeline のパイプライン構造リファレン</u> <u>ス</u>を参照してください。
- ProjectNameの値は、この手順の前半で作成した CodeBuild ビルドプロジェクトの名前です。
- locationの値は、このパイプラインで使用する S3 バケットの名前です。詳細については、AWS CodePipeline ユーザーガイドの CodePipelineのアーティファクトストアとして使用する S3 バケットのポリシーを作成するを参照してください。
- name の値は、このパイプラインの名前です。すべてのパイプラインの名前はアカウントに対して一意である必要があります。

このデータはソースアクションとビルドアクションのみを示していますが、テスト、ビルド出 カアーティファクトのデプロイ、 AWS Lambda 関数の呼び出しなどに関連するアクティビティ のアクションを追加できます。詳細については、AWS CodePipeline ユーザーガイドの <u>AWS</u> CodePipeline のパイプライン構造リファレンスを参照してください。

4. JSON ファイルが保存されているフォルダに切り替え、 CodePipeline の「<u>create-pipeline</u>」コマ ンドを実行し、ファイル名を指定します。

aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json

Note

CodeBuild がサポートされている AWS リージョンでパイプラインを作成する必要が あります。詳細については、「Amazon Web Services 全般のリファレンス」の「<u>AWS</u> CodeBuild」を参照してください。

JSON 形式のデータが出力に表示され、CodePipeline がパイプラインを作成します。

5. パイプラインのステータスに関する情報を取得するには、パイプラインの名前を指定して CodePipeline の get-pipeline-state コマンドを実行します。

aws codepipeline get-pipeline-state --name <my-pipeline-name>

出力で、ビルドが成功したことを確認する情報を探します。省略記号 (...) は、簡潔にするた めに省略されたデータを表すために使用されます。

```
{
    ...
    "stageStates": [
        ...
        {
            "actionStates": [
               {
               "actionName": "CodeBuild",
               "latestExecution": {
                "status": "SUCCEEDED",
               ...
        },
        }
    }
}
```

. } }

このコマンドをあまりに早く実行すると、ビルドアクションに関する情報が表示されないことが あります。パイプラインがビルドアクションの実行を終了するまで、このコマンドを複数回実行 する必要があります。

6. ビルドが成功したら、次の手順に従ってビルド出力アーティファクトを取得します。<u>https://</u> console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

Note

ビルド出力アーティファクトを取得するには、CodeBuild コンソールの関連するビルド の詳細ページで [ビルドアーティファクト] リンクを選択することもできます。このペー ジを表示するには、この手順の残りのステップを省略して、「<u>ビルドの詳細の表示 (コ</u> ンソール)」を参照してください。

バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名は、codepipeline-<region-ID>-<random-number>の形式に従う必要があります。バケット名は、create-pipeline.jsonファイルから取得するか、CodePipelineのget-pipelineコマンドを実行して取得できます。

aws codepipeline get-pipeline --name <pipeline-name>

出力では、pipeline オブジェクトには artifactStore オブジェクトが含まれ、それには、 バケットの名前と location の値が含まれます。

- 8. パイプラインの名前と一致するフォルダを開きます (例:<pipeline-name>)。
- 9. そのフォルダで、default という名前のフォルダを開きます。
- 10. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タ イムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操 作できるように、必要に応じて、ファイルに.zip 拡張子を付けます。) ビルド出力アーティ ファクトは、展開されたファイルの内容に含まれます。

CodeBuild ビルドアクションをパイプラインに追加する (CodePipeline コン ソール)

- 1. 以下を使用して にサインイン AWS Management Console します。
 - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカ ウントルートユーザー」を参照してください。
 - AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
 - ・以下の最小アクションセットを実行するアクセス許可を持つ AWS アカウントのユーザー。

codepipeline:* iam:ListRoles iam:PassRole s3:CreateBucket s3:GetBucketPolicy s3:GetObject s3:ListAllMyBuckets s3:ListBucket s3:PutBucketPolicy codecommit:ListBranches codecommit:ListRepositories codedeploy:GetApplication codedeploy:GetDeploymentGroup codedeploy:ListApplications codedeploy:ListDeploymentGroups elasticbeanstalk:DescribeApplications elasticbeanstalk:DescribeEnvironments lambda:GetFunctionConfiguration lambda:ListFunctions opsworks:DescribeStacks opsworks:DescribeApps opsworks:DescribeLayers

- 2. CodePipeline コンソール (<u>http://console.aws.amazon.com/codesuite/codepipeline/home</u>) を開き ます。
- AWS リージョンセレクタで、パイプラインがある AWS リージョンを選択します。このリージョンでは、CodeBuild をサポートしている必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「CodeBuild」を参照してください。
- 4. [Pipelines (パイプライン)] ページで、パイプラインの名前を選択します。

5. パイプラインの詳細ページの [ソース] アクションで、ツールヒントを選択します。[Output artifact (出力アーティファクト)] の値 (例: MyApp) をメモします。

Note

この手順では、[Source] ステージと [Beta] ステージの間のビルドステージ内にビルド アクションを追加する方法について説明します。ビルドアクションを別の場所に追加す る場合は、ビルドアクションを追加する場所の直前のアクションのツールヒントを選択 し、[Output artifact (出力アーティファクト)] の値をメモします。

- 6. [編集]を選択します。
- 7. [ソース] と [ベータ] ステージの間で、[Add (追加)] を選択します。

Note

この手順では、パイプラインの [Source] ステージと [Beta] ステージの間にビルドアク ションを追加する方法について説明します。既存のステージにビルドアクションを追加 するには、ステージで [Edit stage (ステージを編集)] を選択し、この手順のステップ 8 に進みます。ビルドステージを別の場所に追加するには、目的の場所で [Add stage (ス テージの追加)] を選択します。

Edit: Source	Edit stage
Source S3	Ġ
+ Add sta	age

- [Stage name (ステージ名)] に、ビルドステージの名前 (例: Build) を入力します。別の名前を選 択する場合は、この手順全体でそれを使用します。
- 9. 選択したステージの中で、[アクションの追加]を選択します。

Note

この手順では、ビルドステージの中にビルドアクションを追加する方法について説明 します。ビルドアクションを別の場所に追加するには、目的の場所で [Add action (アク ションの追加)] を選択します。まず、ビルドアクションを追加する既存のステージで [Edit stage (ステージを編集)] アイコンを選択する必要があります。

- 10. [アクションの編集] の [アクション名] に、アクションの名前 (例: CodeBuild) を入力します。 別の名前を選択する場合は、この手順全体でそれを使用します。
- 11. [Action provider] (アクションプロバイダー) で、[CodeBuild] を選択します。
- 12. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロ ジェクトの名前を選択し、この手順の次のステップにスキップします。

新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「<u>ビルドプロジェクトの</u> 作成 (コンソール)」の手順に従ってから、この手順に戻ります。

既存のビルドプロジェクトを選択した場合、ビルド出力アーティファクトの設定がすでに定義 されている必要があります (ただし、CodePipeline によってビルド出力の設定が上書きされま す)。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドプロジェク トの設定の変更 (コンソール)」にある「アーティファクト」の説明を参照してください。

Important

CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されま す。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通 じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金 されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを 無効にすることをお勧めします。CodeBuild コンソールで、[Webhook] ボックスをオフ にします。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照 してください

- 13. [入力アーティファクト] で、この手順で前に書き留めた出力アーティファクトを選択します。
- 14. [出力アーティファクト] で、出力アーティファクトの名前を入力します (例: MyAppBuild)。
- 15. [Add action] を選択します。
- 16. [Save (保存)]、[Save (保存)] の順に選択し、パイプラインの変更を保存します。
- 17. [Release change] を選択します。
- パイプラインが正常に実行されたら、ビルド出力アーティファクトを取得できます。CodePipeline コンソールにパイプラインを表示した状態で、[Build] (ビルド) アクションで ツールヒントを選択します。[Output artifact] の値をメモします (例: MyAppBuild)。

Note

ビルド出力アーティファクトを取得するには、CodeBuild コンソールのビルドの詳細 ページで [Build artifacts] (ビルドアーティファクト) リンクを選択することもできます。 このページの内容を表示するには、「<u>ビルドの詳細の表示 (コンソール)</u>」を参照して、 この手順のステップ 31 に進みます。

- 19. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
- 20. バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名 は、codepipeline-*region-ID-random-number*の形式に従う必要があります。を使用して CodePipeline get-pipeline コマンド AWS CLI を実行し、バケットの名前を取得できます。

aws codepipeline get-pipeline --name my-pipeline-name

出力では、pipeline オブジェクトには artifactStore オブジェクトが含まれ、それには、 バケットの名前と location の値が含まれます。

- 21. パイプラインの名前と一致するフォルダを開きます (パイプライン名の長さによってはフォルダ 名が切り詰められている場合があります)。次に、この手順で前に書き留めた [出力アーティファ クト] の値と一致するフォルダを開きます。
- 22. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タ イムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操 作できるように、必要に応じて、ファイルに.zip 拡張子を付けます。) ビルド出力アーティ ファクトは、展開されたファイルの内容に含まれます。
- 23. CodePipeline にビルド出力アーティファクトをデプロイするよう指示した場合は、デプロイプ ロバイダの説明を活用してデプロイターゲットのビルド出力アーティファクトを取得します。

CodeBuild テストアクションをパイプラインに追加する (CodePipeline コン ソール)

1. 以下を使用して にサインイン AWS Management Console します。

- AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカウントルートユーザー」を参照してください。
- AWS アカウントの管理者ユーザー。詳細については、「ユーザーガイド」の「最初の AWS アカウント ルートユーザーとグループの作成」を参照してください。
- ・以下の最小アクションセットを実行するアクセス許可を持つ AWS アカウントのユーザー。

codepipeline:* iam:ListRoles iam:PassRole s3:CreateBucket s3:GetBucketPolicy s3:GetObject s3:ListAllMyBuckets s3:ListBucket s3:PutBucketPolicy codecommit:ListBranches codecommit:ListRepositories codedeploy:GetApplication codedeploy:GetDeploymentGroup codedeploy:ListApplications codedeploy:ListDeploymentGroups elasticbeanstalk:DescribeApplications elasticbeanstalk:DescribeEnvironments lambda:GetFunctionConfiguration lambda:ListFunctions opsworks:DescribeStacks opsworks:DescribeApps opsworks:DescribeLayers

- 2. CodePipeline コンソール (<u>http://console.aws.amazon.com/codesuite/codepipeline/home</u>) を開き ます。
- AWS リージョンセレクタで、パイプラインがある AWS リージョンを選択します。これ は、CodeBuild がサポートされている AWS リージョンである必要があります。詳細について は、「Amazon Web Services 全般のリファレンス」の「<u>AWS CodeBuild</u>」を参照してくださ い。
- 4. [Pipelines (パイプライン)] ページで、パイプラインの名前を選択します。
- 5. パイプラインの詳細ページの [ソース] アクションで、ツールヒントを選択します。[Output artifact (出力アーティファクト)] の値 (例: MyApp) をメモします。

Note

この手順では、[Source] ステージと [Beta] ステージの間のテストステージ内にテストア クションを追加する方法について説明します。テストアクションを別の場所に追加する 場合は、直前のアクションにマウスポインタを合わせ、[Output artifact] の値をメモしま す。

6. [編集]を選択します。

7. [ソース] ステージのすぐ後で、[Add stage (ステージの追加)] を選択します。

Note

この手順では、パイプラインの [Source] ステージの直後にテストステージを追加する方 法を示します。既存のステージにテストアクションを追加するには、ステージで [Edit stage (ステージを編集)] を選択し、この手順のステップ 8 に進みます。テストステー ジを別の場所に追加するには、目的の場所で [Add stage (ステージの追加)] を選択しま す。

Edit: Source	Edit stage
Source 53	١
+ Add stag	e

- 8. [ステージ名] に、テストステージの名前 (例: Test) を入力します。別の名前を選択する場合 は、この手順全体でそれを使用します。
- 9. 選択されたステージで、[アクションの追加]を選択します。

Note

この手順は、テストステージにテストアクションを追加する方法を示します。テストア クションを別の場所に追加するには、目的の場所で [Add action (アクションの追加)] を 選択します。まず、テストアクションを追加する既存のステージで [Edit (編集)] アイコ ンを選択する必要があります。

- 10. [アクションの編集] の [アクション名] に、アクションの名前 (例: Test) を入力します。別の名 前を選択する場合は、この手順全体でそれを使用します。
- 11. [Action provider] (アクションプロバイダー) の [Test] (テスト) で、[CodeBuild] を選択します。
- 12. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロ ジェクトの名前を選択し、この手順の次のステップにスキップします。

新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「<u>ビルドプロジェクトの</u> <u>作成 (コンソール)</u>」の手順に従ってから、この手順に戻ります。

Important

CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されま す。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通 じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金 されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを 無効にすることをお勧めします。CodeBuild コンソールで、[Webhook] ボックスをオフ にします。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照 してください

- 13. [入力アーティファクト] で、この手順で前に書き留めた [出力アーティファクト] の値を選択しま す。
- 14. (オプション) テストアクションで出力アーティファクトを生成し、それに応じてビルド仕様を 設定する場合は、出力アーティファクトに割り当てる値を[出力アーティファクト] に入力しま す。
- 15. [Save] を選択します。
- 16. [Release change] を選択します。

- パイプラインが正常に実行された後、テスト結果を取得できます。パイプラインの [Test] (テスト) ステージで [CodeBuild] ハイパーリンクを選択し、CodeBuild コンソールで関連するビルドプロジェクトのページを開きます。
- 18. ビルドプロジェクトページの [Build history] エリアで、[Build run] ハイパーリンクを選択します。
- 19. ビルドの実行ページの [Build logs] (ビルドログ) エリアで、[View entire log] (ログ全体の表示) ハ イパーリンクを選択し、Amazon CloudWatch コンソールでビルドログを開きます。
- 20. ビルドログをスクロールして、テスト結果を表示します。

Codecov AWS CodeBuild で を使用する

Codecov は、コードのテストカバレッジを測定するツールです。Codecov は、コード内のどのメ ソッドとステートメントがテストされていないかを識別します。その結果に基づいて、コードの品質 を向上させるためのテストの記述先を判断します。Codecov は、 CodeBuild でサポートされている ソースリポジトリのうち、3 つ (GitHub、GitHub Enterprise Server、Bitbucket) でサポートされてい ます。ビルドプロジェクトで GitHub Enterprise Server を使用する場合は、Codecov Enterprise を使 用する必要があります。

Codecov を統合した CodeBuild プロジェクトのビルドを実行すると、リポジトリ内のコードを分 析する Codecov レポートが Codecov にアップロードされます。ビルドログには、レポートへのリ ンクが含まれています。次のサンプルでは、Python および Java ビルドプロジェクトを Codecov と統合する方法を示します。Codecov でサポートされている言語のリストについては、<u>Codecov</u> Supported Languages を参照してください。

Codecov とビルドプロジェクトの統合

Codecov をビルドプロジェクトに統合するには、次の手順に従います。

Codecov とビルドプロジェクトを統合するには

- <u>https://codecov.io/signup</u>に移動し、GitHub または Bitbucket ソースリポジトリにサイン アップします。GitHub Enterprise を使用する場合は、Codecov ウェブサイトの「<u>Codecov</u> Enterprise」を参照してください。
- 2. Codecov に、カバレッジ対象のリポジトリを追加します。
- 3. トークン情報が表示されたら、[Copy] を選択します。

Overview Overview	 ↔ Commits 	& Branches	ាំ Pulls	ឿ Compare	¢¦ ⁸ Settings	
Let's get your project covered.						
No repository activation required. Simply upload a report and the project activates automatically.						
STEP 1 - COPY TOKEN						

🖻 Copy

4.	コピーしたトークンを、CODEC	OV_TOKEN という名前の環境変数としてビルドプロジェクトに
	追加します。詳細については、	「ビルドプロジェクトの設定の変更 (コンソール)」を参照してく
	ださい。	

5. リポジトリ内に my_script.sh という名前のテキストファイルを作成します。このファイルに 次の内容を入力します。

#/bin/bash bash <(curl -s https://codecov.io/bash) -t \$CODECOV_TOKEN</pre>

Upload Token

6. ビルドプロジェクトの用途に応じて [Python] タブまたは [Java] タブを選択し、次の手順に従い ます。

Java

1. 次の JaCoCo プラグインをリポジトリ内の pom.xml に追加します。

<build></build>					
<plugins></plugins>					
<plugin></plugin>					
<groupid>org.jacoco</groupid>					
<artifactid>jacoco-maven-plugin</artifactid>					
<version>0.8.2</version>					
<executions></executions>					
<execution></execution>					
<goals></goals>					
<goal>prepare-agent</goal>					
<execution></execution>					
<id>report</id>					
<phase>test</phase>					

```
<goals>
<goal>report</goal>
</goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

 buildspec ファイルに次のコマンドを入力します。詳細については、「<u>buildspec の構</u> 文」を参照してください。

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

buildspec ファイルに次のコマンドを入力します。詳細については、「<u>buildspec の構文</u>」を 参照してください。

```
build:
    pip install coverage
    coverage run -m unittest discover
postbuild:
    echo 'Connect to CodeCov'
    bash my_script.sh
```

 ビルドプロジェクトのビルドを実行します。プロジェクト用に生成された Codecov レポートへのリンクがビルドログに表示されます。リンクを使用して Codecov レポートを表示します。詳細については、<u>AWS CodeBuild ビルドを手動で実行する</u>および<u>を使用した AWS CodeBuild API</u> <u>コールのログ記録 AWS CloudTrail</u>を参照してください。ビルドログの Codecov 情報は、次のように表示されます。

Codecov とビルドプロジェクトの統合



レポートは次のように表示されます。



Jenkins AWS CodeBuild で を使用する

の Jenkins プラグインを使用して AWS CodeBuild 、CodeBuild を Jenkins ビルドジョブと統合でき ます。Jenkins ビルドノードにビルドジョブを送信する代わりに、プラグインを使用してビルドジョ ブを CodeBuild に送信します。これにより、Jenkins ビルドノードのプロビジョニング、設定、およ び管理が不要になります。

トピック

- Jenkinsの設定
- プラグインをインストールする
- プラグインを使用

Jenkins の設定

AWS CodeBuild プラグインで Jenkins をセットアップする方法と、プラグインのソースコード をダウンロードする方法については、「https://<u>https://github.com/awslabs/aws-codebuild-jenkins-</u> plugin.」を参照してください。

プラグインをインストールする

Jenkins サーバーをセットアップ済みで、 AWS CodeBuild プラグインのみをインストールする場合 は、Jenkins インスタンスの Plugin Manager で **CodeBuild Plugin for Jenkins** を検索しま す。

プラグインを使用

VPC の外部からのソース AWS CodeBuild で を使用するには

- CodeBuild コンソールでプロジェクトを作成します。詳細については、「ビルドプロジェクトの 作成 (コンソール)」を参照してください。
 - ビルドを実行する AWS リージョンを選択します。
 - ・ (オプション) CodeBuild ビルドコンテナによる VPC のリソースへのアクセスを許可するよう に Amazon VPC 設定を指定します。
 - プロジェクトの名前を書き留めます。これはステップ3で必要になります。
 - ・ (オプション) ソースリポジトリが CodeBuild でネイティブにサポートされていない場合は、 プロジェクトの入力ソースタイプとして Amazon S3 を設定できます。
- 2. IAM コンソールで、Jenkins プラグインで使用するユーザーを作成します。
 - ユーザーの認証情報を作成するときに、[Programmatic Access (プログラムによるアクセス)]
 を選択します。

• 次のようなポリシーを作成し、このポリシーをユーザーにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:logs:{{region}}:{{awsAccountId}}:log-group:/aws/
codebuild/{{projectName}}:*"],
      "Action": ["logs:GetLogEvents"]
   },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{inputBucket}}"],
      "Action": ["s3:GetBucketVersioning"]
   },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{inputBucket}}/{{inputObject}}"],
      "Action": ["s3:PutObject"]
   },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{outputBucket}}/*"],
      "Action": ["s3:GetObject"]
   },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:codebuild:{{region}}:{{awsAccountId}}:project/
{{projectName}}"],
      "Action": ["codebuild:StartBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"]
    }
 ]
}
```

- 3. Jenkins で自由形式のプロジェクトを作成します。
 - [Configure] (設定) ページで、[Add build step] (ビルドステップの追加)、[Run build on CodeBuild] (CodeBuild でビルドを実行) を選択します。
 - ビルドステップを設定します。

- [Region (リージョン)]、[Credentials (認証情報)]、および [Project Name (プロジェクト名)] の値を入力します。
- [Use Project source (プロジェクトソースを使用)] を選択します。
- ・ 設定を保存し、Jenkins からビルドを実行します。
- [Source Code Management (ソースコードの管理)] で、ソースの取得方法を選択しま す。Jenkins サーバーでの GitHub プラグイン (またはソースリポジトリプロバイダ用の Jenkins プラグイン) のインストールが必要になる場合があります。
 - [設定] ページで、[ビルドステップの追加] を選択し、[AWS CodeBuildでビルドを実行] を選択 します。
 - ビルドステップを設定します。
 - [Region (リージョン)]、[Credentials (認証情報)]、および [Project Name (プロジェクト名)] の値を入力します。
 - [Use Jenkins source (Jenkins ソースを使用)] を選択します。
 - ・ 設定を保存し、Jenkins からビルドを実行します。

Jenkins パイプライン AWS CodeBuild プラグインでプラグインを使用するには

 Jenkins パイプラインプロジェクトページで、スニペットジェネレーターを使用して、パイプラ インにステップとして CodeBuild を追加するパイプラインスクリプトを生成します。次のよう なスクリプトが生成されます。

awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2', sourceControlType: 'jenkins'

サーバーレスアプリケーションで AWS CodeBuild を使用する

AWS Serverless Application Model (AWS SAM)は、サーバーレスアプリケーションを構築するた めのオープンソースフレームワークです。詳細については、GitHub の <u>AWS serverless application</u> <u>model</u> リポジトリを参照してください。

を使用して AWS CodeBuild 、標準に準拠したサーバーレスアプリケーションをパッケージ化 AWS SAM およびデプロイできます。デプロイのステップで、CodeBuild は AWS CloudFormationを使用 できます。CodeBuild と を使用してサーバーレスアプリケーションの構築とデプロイを自動化する には AWS CloudFormation、 を使用できます AWS CodePipeline。 詳細については、AWS Serverless Application Model 開発者ガイドの「<u>サーバーレスアプリケーショ</u> ンをデプロイする」を参照してください。

関連リソース

- の開始方法については AWS CodeBuild、「」を参照してください<u>コンソール AWS CodeBuild を</u> 使用した の開始方法。
- CodeBuild の問題のトラブルシューティングについては、「<u>トラブルシューティング AWS</u> <u>CodeBuild</u>」を参照してください。
- ・ CodeBuild のクォータについては、「<u>のクォータ AWS CodeBuild</u>」を参照してください。

AWS CodeBuild for Windows のサードパーティー通知

CodeBuild for Windows のビルドを使用すると、複数のサードパーティー製のパッケージやモジュー ルを使用して、構築済みのアプリケーションを Microsoft Windows オペレーティングシステムで実行 したり、複数のサードパーティー製品と相互運用したりできます。指定したサードパーティー製の パッケージやモジュールの使用に適用されるサードパーティーの法的条項を以下に示します。

トピック

- <u>1) 基本 Docker イメージ windowsservercore</u>
- 2) Windows ベースの Docker イメージ Choco
- 3) Windows ベースの Docker イメージ git --version 2.16.2
- 4) Windows ベースの Docker イメージ microsoft-build-tools --version 15.0.26320.2
- 5) Windows ベースの Docker イメージ nuget.commandline --version 4.5.1
- 7) Windows ベースの Docker イメージ netfx-4.6.2-devpack
- 8) Windows ベースの Docker イメージ visualfsharptools, v 4.0
- 9) Windows ベースの Docker イメージ netfx-pcl-reference-assemblies-4.6
- 10) Windows ベース Docker イメージ visualcppbuildtools v 14.0.25420.1
- 11) Windows ベースの Docker イメージ microsoft-windows-netfx3-ondemand-package.cab
- <u>12) Windows ベースの Docker イメージ dotnet-sdk</u>
- 1) 基本 Docker イメージ windowsservercore

(ライセンス条項は https://https://hub.docker.com/_/microsoft-windows-servercore.)

ライセンス: この Windows コンテナー用のコンテナー OS イメージを要求および使用する場合、次の追加ライセンス条項を承認、理解し、同意するものと見なされます。

Microsoft ソフトウェア追加ライセンス条項

コンテナー OS イメージ

Microsoft Corporation (またはお住まいの地域に基づいていずれかの関連会社) (以下 "Microsoft") は、 このコンテナー OS イメージの追加機能 (以下 "追加機能") のライセンスをユーザーに付与します。 この追加機能と、基になるホストオペレーティングシステムソフトウェア (以下 "ホストソフトウェ ア") を組み合わせて使用するライセンスは、ホストソフトウェアでのコンテナー機能の実行を支援す る目的でのみ付与されます。この追加機能の使用には、ホストソフトウェアのライセンス条項が適用 されます。ホストソフトウェアのライセンスを持っていない場合、使用することはできません。ライ センスが有効なホストソフトウェアのコピーがある場合にのみ、この追加機能を使用できます。

その他のライセンス要件と使用権

前述の条項に従って追加機能を使用すると、特定の追加機能コンポーネントを含むコンテナーイメー ジ (以下 "コンテナーイメージ") が作成または変更される場合があります。明確にしておくと、コ ンテナーイメージは、仮想マシンまたは仮想アプライアンスイメージとは別のものです。Microsoft は、本ライセンス条項に従い、以下の条件で、そのような追加機能コンポーネントを再配布する限定 的な権利をユーザーに付与します。

(i) 追加機能コンポーネントは、ユーザーのコンテナーイメージ内かつユーザーのコンテナーイメージの一部としてのみ使用できます。

(ii) 追加機能とは実質的に異なる重要な主機能がコンテナーイメージにある場合に限り、コンテナー イメージ内の追加機能コンポーネントを使用できます。

(iii) エンドユーザーが追加機能コンポーネントを使用する場合に適切にライセンスが付与されるよう に、本ライセンス条項 (または Microsoft やホスト側で必須とする同様の条項) をコンテナーイメージ に含めることに同意します。

Microsoft は、本条項で明記されていないその他のすべての権利を有します。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下 の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

この Windows コンテナー用のコンテナー OS イメージの追加ライセンス条項の一部として、基 になる Windows Server ホストソフトウェアのライセンス条項 (<u>https://www.microsoft.com/en-us/</u> useterms.) も適用されます。

2) Windows ベースの Docker イメージ – Choco

(ライセンス条項の参照先: https://github.com/chocolatey/choco/blob/master/LICENSE)

Copyright 2011 - Present RealDimensions Software, LLC

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファ イルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所 から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布さ れるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布さ れます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してくださ い。

3) Windows ベースの Docker イメージ – git --version 2.16.2

(ライセンス条項の参照先: https://chocolatey.org/packages/git/2.16.2)

GNU 一般公衆ライセンス、バージョン 2 に基づいてライセンスされます。参照先: <u>https://</u> www.gnu.org/licenses/old-licenses/gpl-2.0.html

4) Windows ベースの Docker イメージ – microsoft-build-tools --version 15.0.26320.2

(ライセンス条項の参照先: https://www.visualstudio.com/license-terms/mt171552/)

MICROSOFT VISUAL STUDIO 2015 拡張機能、VISUAL STUDIO SHELLS および C++ 再頒布可能 パッケージ

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社)とお客 様との契約を構成します。本ライセンス条項は、上記のソフトウェア (以下「本ソフトウェア」とい います) に適用されます。本ライセンス条項は、別途のライセンス条項が付属している場合を除き、 本ソフトウェアに関連するマイクロソフトのサービスまたは更新プログラムにも適用されます。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

- インストールおよび使用に関する権利お客様は、本ソフトウェアの任意の数の複製をインストールして使用することができます。
- 2. 特定のコンポーネントに関する条件
 - a. ユーティリティ。このソフトウェアには、<u>https://docs.microsoft.com/en-us/visualstudio/</u> productinfo/2015-redistribution-vs の [ユーティリティリスト] の一部の項目が含まれている場合 があります。お客様は、本ソフトウェアと共に開発したアプリケーションやデータベースをデ バッグおよび展開するために、ソフトウェアに含まれている場合、お客様自身または他のサー ドパーティー製マシンにこれらのアイテムをコピーしてインストールすることができます。 ユーティリティは一時的な使用を目的として設計されていること、マイクロソフトは本ソフト ウェアの他のコンポーネントと切り離してユーティリティにパッチを適用したり、ユーティリ ティを更新したりできない場合があること、および一部のユーティリティはその性質上、その ユーティリティがインストールされているコンピュータに他者がアクセスできるようにするこ とが可能であることに注意してください。このため、お客様は、お客様のアプリケーションお よびデータベースのデバッグまたは展開が終了した後で、お客様がインストールしたすべての ユーティリティを削除する必要があります。マイクロソフトは、お客様が任意のコンピュータ にインストールしたユーティリティの第三者による使用またはアクセスについて責任を負いま せん。
 - b. マイクロソフトプラットフォーム本ソフトウェアには、Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office、および Microsoft SharePointのコンポーネントが含まれていることがあります。これらのコンポーネン トには、本ソフトウェアに付属しているマイクロソフトの「Licenses」フォルダーに規定され ている、別途のライセンス条項および固有の製品サポートポリシーが適用されます。ただし、 関連するインストールディレクトリにこれらのコンポーネントのライセンス条項も含まれてい る場合は当該ライセンス条項が適用されます。
 - c. 第三者のコンポーネント 本ソフトウェアには、別途の法的通知を含みまたは別の契約が適用さ れる第三者のコンポーネントが含まれている場合があり、これらについては本ソフトウェアに 付属する ThirdPartyNotices ファイルに規定されています。かかるコンポーネントには、他の 契約が適用される場合でも、以下の保証の免責、損害賠償の制限および除外も適用されます。 本ソフトウェアには、ソースコードの公開義務が適用されるオープンソースライセンスに基づ いてライセンスが許諾されるコンポーネントも含まれている場合があります。該当する場合、 これらのライセンスの複製は、ThirdPartyNotices ファイルに含まれています。お客様は、当該 オープンソースライセンスで求められているとおり、5.00 米ドルの郵便為替または小切手を 次の宛先に送付することにより、対応するソースコードをマイクロソフトから取得することが できます: Source Code Compliance Team, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052。支払いの備考欄には、下記の1つまたは複数のコンポーネントのソースコードを 記載してください。

- Remote Tools for Visual Studio 2015
- Standalone Profiler for Visual Studio 2015
- IntelliTraceCollector for Visual Studio 2015
- Microsoft VC++ Redistributable 2015
- Multibyte MFC Library for Visual Studio 2015
- Microsoft Build Tools 2015
- Feedback Client
- Visual Studio 2015 Integrated Shell
- Visual Studio 2015 Isolated Shell

マイクロソフトは、ソースコードの複製を <u>http://thirdpartysource.microsoft.com</u> で公開するこ ともあります。

- 3. データ。本ソフトウェアは、お客様およびお客様による本ソフトウェアの使用に関する情報を収 集し、マイクロソフトに送信することがあります。マイクロソフトはこの情報を、サービスの提 供ならびにマイクロソフトの製品およびサービスの向上を目的として使用することがあります。 お客様は、製品付属の文書に説明されているとおり、これらの情報収集の多くを停止することが できますが、すべてを停止することはできません。また、本ソフトウェアにある特定の機能を使 用すると、お客様がお客様のアプリケーションのユーザーからデータを収集できる場合がありま す。お客様は、これらの機能を使用する場合、お客様のアプリケーションのユーザーに適切な通 知を提供するなど、適用される法令を遵守しなければなりません。データの収集および使用の詳 細については、「<u>https://privacy.microsoft.com/en-us/privacystatement</u>」のヘルプドキュメントお よびマイクロソフトのプライバシーに関する声明を参照してください。本ソフトウェアを使用し た場合、お客様はこれらの規定に同意したものとみなされます。
- 4. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありま せん。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。そ の他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項 の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された 方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた 使用方法を制限する技術的制限に従うものとします。以下の行為は禁じられています。
 - 本ソフトウェアの技術的な制限を回避すること。
 - 本ソフトウェアのリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルを実行または試行すること。ただし、本ソフトウェアに含まれる場合がある一定のオープンソースコンポーネントの使用に適用される第三者のライセンス条項により求められている場合を除きます。

- 本ソフトウェアの Microsoft またはサプライヤーの告知を削除、最小化、ブロックまたは修正すること。
- 法律に違反する方法で本ソフトウェアを使用すること。
- 本ソフトウェアを共有、公開、レンタル、もしくはリースすること、本ソフトウェアを第三者 が使用できるようにスタンドアロンのホスト型ソリューションとして提供すること。
- 5. 輸出規制 お客様は、本ソフトウェアに適用されるすべての国内法および国際法 (輸出対象国、エ ンドユーザーおよびエンドユーザーによる使用に関する制限を含みます) を遵守しなければなりま せん。輸出規制の詳細については、(aka.ms/exporting) を参照してください。
- 5. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本 ソフトウェアに関してサポートサービスを提供しない場合があります。
- 7. 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネット ベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびサポートサー ビスについてのお客様とマイクロソフトとの間の完全なる合意を構成します。
- 8. 準拠法 お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解釈および契約 違反への主張は、米国ワシントン州法に準拠するものとします。他の主張については、お客様が 所在する地域の法律に準拠します。お客様が本ソフトウェアを他の国で入手した場合は、当該地 域の法律を準拠法とします。
- 9. 消費者の権利、地域による違い本契約は、特定の法的な権利を規定したものです。お客様は、地域や国によっては、消費者権利を含め、その他の権利を有する場合があります。Microsoftとお客様との関係とは別に、お客様が本ソフトウェアを取得した当事者に関する権利を有する場合もあります。本契約は、お客様の地域または国の法令が権利の変更を許容しない場合、それらのその他の権利を変更しないものとします。たとえば、お客様が本ソフトウェアを以下のいずれかの地域で取得した場合、または強行的な国の法令が適用される場合には、以下の規定がお客様に適用されます。
 - a. オーストラリア お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条 項は、それらの権利に影響を与えることを意図するものではありません。
 - b. カナダ。本ソフトウェアをカナダで取得した場合、自働更新機能をオフにするか、お使いの機器をインターネットから取り外すか(ただし、インターネットに再接続すると、本ソフトウェアは更新プログラムのチェックとインストールを再開します)、または本ソフトウェアをアンインストールすることにより、更新受信を停止することができます。製品付属の文書がある場合は、当該文書にお客様の特定のデバイスまたはソフトウェアの更新をオフにする方法が記載されていることもあります。
 - c. ドイツおよびオーストリア

- i. 保証 正規にライセンスを取得したソフトウェアは、本ソフトウェアに付属するマイクロソフトの資料の記載に実質的に従って動作します。ただし、マイクロソフトは、ライセンスを取得したソフトウェアに関して契約上の保証は一切いたしません。
- ii. 限定責任 故意、重過失、製品責任法に基づく請求があった場合、および死亡、人的または物的損傷があった場合、Microsoft は、制定法に従って責任を負うものとします。前掲条項(ii) を条件とし、Microsoft が軽過失に該当する契約違反をして、同義務を履行することは本契約の正当な履行に資するものであって、同義務の違反は本契約の目的および当事者が常に拠り所とする本契約への準拠を損なう(いわゆる「基本的義務」に違反する)可能性がある場合、Microsoft は当該の軽過失についてのみ責任を負うものとします。その他の軽過失については、マイクロソフトは責任を負いません。
- 10.保証の免責: 本ソフトウェアは、「現状有姿のまま」ライセンス供与されます。本ソフトウェアの 使用に伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いた しません。お客様の地域の法律によって認められる範囲において、マイクロソフトは、商品性、 特定目的に対する適合性、および侵害の不存在に関する黙示の保証責任を負いません。
- 11損害賠償に関する制限および除外 YOU CAN RECOVER FROM MICROSOFT およびその供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、逸失利 益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負いませ ん。この制限は、(a) 本ソフトウェア、サービス、第三者のインターネットのサイト上のコンテン ツ (コードを含みます) または第三者のアプリケーションに関連した事項、および (b) 契約違反、 保証違反、厳格責任、過失、または不法行為等の請求 (適用される法令により認められている範囲 において) に適用されます。

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場 合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制 限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合がありま す。

EULA ID: VS2015_Update3_ShellsRedist_<ENU>

5) Windows ベースの Docker イメージ – nuget.commandline --version 4.5.1

(ライセンス条項の参照先: https://github.com/NuGet/Home/blob/dev/LICENSE.txt)

Copyright (c) .NET Foundation。All rights reserved。

5) Windows ベースの Docker イメージ – nuget.commandline --version 4.5.1

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファ イルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所 から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布さ れるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布さ れます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してくださ い。

7) Windows ベースの Docker イメージ – netfx-4.6.2-devpack

Microsoft ソフトウェア追加ライセンス条項

.NET FRAMEWORK AND ASSOCIATED LANGUAGE PACKS FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) は、本追加ソフトウェアのラ イセンスをお客様に供与します。Microsoft Windows operating system ソフトウェア (以下「本ソフ トウェア」といいます) を使用するためのラインセンスを取得している場合は、本追加ソフトウェア を使用できます。本ソフトウェアのライセンスを取得していない場合は、本追加ソフトウェアを使用 することはできません。お客様は、本ソフトウェアの有効なライセンス取得済みの複製 1 部ごとに 本追加ソフトウェアを使用できます。

以下のライセンス条項は、本ソフトウェアの追加の使用条件について説明しています。これらの条項 と本ソフトウェアのライセンス条項が本追加ソフトウェアの使用に適用されます。両者の間に矛盾が ある場合は、本追加ライセンス条項が適用されます。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下 の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

頒布可能コード。本追加ソフトウェアは頒布可能コードで構成されています。「頒布可能コード」とは、お客様が開発されたプログラムに含めて頒布することができるコードです。ただし、お客様は以下の条件に従うものとします。

- a. 使用および頒布の権利
 - お客様は、本追加ソフトウェアをオブジェクトコード形式で複製し、頒布することができます。
 - <u>第三者による頒布。</u>お客様は、お客様のプログラムの頒布者に対して、お客様のプログラムの一部として頒布可能コードの複製および頒布を許可することができます。
- b. 頒布の条件 お客様は、お客様が頒布するすべての頒布可能コードにつき、以下に従わなければ なりません
 - お客様のプログラムにおいて頒布可能コードに重要な新しい機能を追加すること
 - .lib というファイル名拡張子が付いた頒布可能コードの場合は、リンカーによってその頒布可
 能コードを実行した結果だけをお客様のプログラムと共に頒布すること
 - セットアッププログラムに含まれる頒布可能コードを、改変されていないセットアッププロ グラムの一部としてのみ頒布すること
 - お客様のアプリケーションの頒布者およびエンドユーザーに、本ライセンス条項と同等以上 に頒布可能コードを保護する条項に同意させること
 - お客様のアプリケーションにお客様名義の有効な著作権表示を行うこと
 - お客様のプログラムの頒布または使用に関するクレームについて、マイクロソフトを免責、
 保護、補償すること (弁護士費用についての免責、保護、補償も含む)
- c. 頒布の制限 以下の行為は禁じられています
 - 頒布可能コードの著作権、商標または特許の表示を改変すること
 - お客様のプログラムの名称の一部にマイクロソフトの商標を使用したり、お客様のプログラムがマイクロソフトから由来したり、マイクロソフトが推奨しているように見せかけること
 - Windows プラットフォーム以外のプラットフォームで実行する目的で頒布可能コードを頒布 すること
 - 頒布可能コードを悪質、詐欺的または違法なプログラムに組み込むこと
 - 除外ライセンスの適用対象となるような方法で頒布可能コードのソースコードを改変または 頒布すること。「除外ライセンス」とは、使用、改変または頒布の条件として以下を義務付 けるライセンスです。
 - コードをソースコード形式で公表または頒布すること
 - 他者が改変する権利を有すること
- 2. 本追加ソフトウェアのサポート サービス マイクロソフトは、本ソフトウェアに対し

て、<u>www.support.microsoft.com/common/international.aspx</u>に記載するサポートサービスを提供します。

8) Windows ベースの Docker イメージ – visualfsharptools, v 4.0

(ライセンス条項の参照先: https://github.com/dotnet/fsharp/blob/main/License.txt)

Copyright (c) Microsoft Corporation。All rights reserved。

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファ イルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所 から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布さ れるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布さ れます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してくださ い。

9) Windows ベースの Docker イメージ – netfx-pcl-reference-assemblies-4.6

Microsoft ソフトウェアライセンス条項

MICROSOFT .NET PORTABLE CLASS LIBRARY REFERENCE ASSEMBLIES - 4.6

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) とお客 様との契約を構成します。本ライセンス条項をお読みください。本ライセンス条項は、上記のソフト ウェア (以下「本ソフトウェア」といいます) に適用されます。本ライセンス条項は、本ソフトウェ アに関連するマイクロソフトの以下の各項目にも適用されます。

- 更新,
- 追加プログラム
- インターネットベースのサービス
- サポートサービス

ただし、上記項目に別途のライセンス条項が付属している場合を除きます。別途のライセンス条項が 付属している場合は、それらの別途のライセンス条項が適用されます。

本ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。これらの 条項に同意されない場合、本ソフトウェアは使用しないでください。 本ライセンス条項を遵守することを条件として、お客様には以下の永続的な権利が許諾されます。

- インストールおよび使用に関する権利 お客様は、お客様のプログラムを設計、開発およびテスト するために、本ソフトウェアの任意の数の複製をインストールして使用することができます。
- 2. その他のライセンス要件と使用権
 - a. 頒布可能コード。お客様は、以下の条項を遵守することを条件として、お客様が開発した開発 者ツールプログラムで本ソフトウェアを配布し、お客様のプログラムのユーザーに対してポー タブルライブラリを開発して任意のデバイスまたはオペレーティングシステムで使用すること を許可できます。
 - i. 使用および頒布の権利 本ソフトウェアは「頒布可能コード」です。
 - <u>頒布可能コード</u>。お客様は、本ソフトウェアをオブジェクトコード形式で複製し、頒布することができます。
 - <u>第三者による頒布。</u>お客様は、お客様のプログラムの頒布者に対して、お客様のプログラムの一部として頒布可能コードの複製および頒布を許可することができます。
 - ii. 頒布の条件 お客様は、お客様が頒布するすべての頒布可能コードにつき、以下に従わなけれ ばなりません
 - お客様のプログラムにおいて頒布可能コードに重要な新しい機能を追加すること。
 - お客様のアプリケーションの頒布者およびユーザーに、本ライセンス条項と同等以上に頒 布可能コードを保護する条項に同意させること
 - お客様のアプリケーションにお客様名義の有効な著作権表示を行うこと。
 - お客様のプログラムの頒布または使用に関するクレームについて、マイクロソフトを免 責、保護、補償すること(弁護士費用についての免責、保護、補償も含む)

iii. 頒布の制限 以下の行為は禁じられています

- 頒布可能コードの著作権、商標または特許の表示を改変すること
- お客様のプログラムの名称の一部にマイクロソフトの商標を使用したり、お客様のプログ ラムがマイクロソフトから由来したり、マイクロソフトが推奨しているように見せかける こと
- 頒布可能コードを悪質、詐欺的または違法なプログラムに組み込むこと
- 除外ライセンスの適用対象となるような方法で頒布可能コードを改変または頒布すること。「除外ライセンス」とは、使用、改変または頒布の条件として以下を義務付けるライセンスです。

⁹⁾ Windows ベースのDockerをシージスettlepcl-ド語誌e-as公式表表なは頒布すること

• 他者が改変する権利を有すること

- 3. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありません。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。その他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた使用方法を制限する技術的制限に従うものとします。以下の行為は禁じられています。
 - 本ソフトウェアの技術的な制限を回避すること。
 - 本ソフトウェアをリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルすること。ただし、この制限にもかからわず、適用される法によって明示的に許可される場合を除きます。
 - 本ソフトウェアを公開して第三者に複製させること。
 - 本ソフトウェアをレンタル、リースまたは貸与すること。
- 4. フィードバック お客様は、本ソフトウェアに関するフィードバックを提供できます。本ソフト ウェアに関するフィードバックをお客様がマイクロソフトに提供した場合は、方法および目的を 問わず、そのフィードバックを無償で使用、公開、および商用利用する権利をお客様がマイクロ ソフトに付与したものと見なされます。また、そのフィードバックが含まれているマイクロソフ トのソフトウェアまたはサービスの特定部分が、第三者の製品、テクノロジ、およびサービスに よって使用される場合またはその部分との連携が行われる場合に必要となる特許権についても、 お客様が無償で付与したものと見なされます。お客様のフィードバックをソフトウェアまたは ドキュメントに含めるために、マイクロソフトから第三者に対して、そのソフトウェアまたはド キュメントの使用許諾が必要となるようなライセンスが適用されるフィードバックは、お客様か ら提供されないものとします。これらの権利は、本契約の終了後も継続するものとします。
- 5. 第三者への譲渡本ソフトウェアの最初のユーザーは、本ソフトウェアおよび本契約を第三者に譲渡できます。譲渡する前に、当該の第三者は、本契約が本ソフトウェアの譲渡と使用に適用されることに同意する必要があります。最初のユーザーは、本ソフトウェアを譲渡する前に、本ソフトウェアをデバイスからアンインストールする必要があります。最初のユーザーは、一切の複製を保持しないものとします。
- 6. 輸出規制 本ソフトウェアは、アメリカ合衆国の輸出に関する規制の対象となります。お客様は、 本ソフトウェアに適用されるすべての国内外の輸出に関する法および規制を遵守しなければなり ません。これらの法には、輸出対象国、エンドユーザーおよびエンドユーザーによる使用に関す る制限が含まれます。詳細については、www.microsoft.com/exporting を参照してください。
- 7. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本 ソフトウェアに関してサポートサービスを提供しない場合があります。

- 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネット ベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびマイクロソフ トが提供するすべてのサポートサービスについてのお客様とマイクロソフトとの間の完全なる合 意を構成します。
- 9. 準拠法
 - a. アメリカ合衆国。お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解 釈および契約違反への主張は、抵触法にかかわらず、米国ワシントン州法に準拠するものとし ます。他の主張については、消費者保護法、公正取引法、および違法行為に基づく主張も含め て、お客様が所在する地域の法律に準拠します。
 - b. 米国以外 お客様が本ソフトウェアを他の国で入手した場合は、当該国の法律を準拠法としま す。
- 10法的効力 本契約は、特定の法的な権利を規定したものです。お客様は、国の法律によっては、そ の他の権利を有する場合があります。また、お客様が本ソフトウェアを取得された第三者に関す る権利を有する場合もあります。本ライセンス条項は、お客様の国の法律がその法律に基づく権 利の変更を許容しない場合、それらの権利を変更しないものとします。
- 11.保証の免責: 本ソフトウェアは "現状のまま" ライセンス供与されます。本ソフトウェアの使用に 伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いたしません。本ライセンス条項では変更できない、お客様の地域の法律による追加の消費者の権利または 法定保証が存在する場合があります。お客様の地域の法律によって認められる範囲において、マ イクロソフトは、商品性、特定目的に対する適合性、および侵害の不存在に関する黙示の保証責 任を負いません。

オーストラリア限定 – お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセン ス条項は、それらの権利に影響を与えることを意図するものではありません。

12.救済手段および損害賠償の制限および除外 YOU CAN RECOVER FROM MICROSOFT およびそ の供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、 逸失利益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負 いません。

この制限は、以下に適用されるものとします。

- ・ 本ソフトウェア、サービス、第三者のインターネットサイト上のコンテンツ (コードを含みます) または第三者のプログラムに関連した事項
- 契約違反、保証違反、無過失責任、過失または不法行為 (適用法で許可されている範囲において)

9) Windows ベースの Docker イメージ – netfx-pcl-reference-assemblies-4.6

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合があります。

10) Windows ベース Docker イメージ — visualcppbuildtools v 14.0.25420.1

(ライセンス条項の参照先: https://www.visualstudio.com/license-terms/mt644918/)

MICROSOFT VISUAL C++ 構築ツール

Microsoft ソフトウェアライセンス条項

MICROSOFT VISUAL C++ 構築ツール

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社)とお客様との契約を構成します。本ライセンス条項は、上記のソフトウェア (以下「本ソフトウェア」といいます) に適用されます。本ライセンス条項は、別途のライセンス条項が付属している場合を除き、 本ソフトウェアに関連するマイクロソフトのサービスまたは更新プログラムにも適用されます。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

- 1. インストールおよび使用に関する権利
 - a. 1 人のユーザーが、アプリケーションの開発およびテストを行うために、本ソフトウェアの複 製を使用することができます。
- 2. データ。本ソフトウェアは、お客様およびお客様による本ソフトウェアの使用に関する情報を収 集し、マイクロソフトに送信することがあります。マイクロソフトはこの情報を、サービスの提 供ならびにマイクロソフトの製品およびサービスの向上を目的として使用することがあります。 お客様は、製品付属の文書に説明されているとおり、これらの情報収集の多くを停止することが できますが、すべてを停止することはできません。また、本ソフトウェアにある特定の機能を使 用すると、お客様がお客様のアプリケーションのユーザーからデータを収集できる場合がありま す。お客様は、これらの機能を使用する場合、お客様のアプリケーションのユーザーに適切な通 知を提供するなど、適用される法令を遵守しなければなりません。データの収集および使用の詳 細については、ヘルプドキュメントおよびマイクロソフトのプライバシーに関する声明を参照し

てください: <u>http://go.microsoft.com/fwlink/?LinkID=528096</u>。本ソフトウェアを使用した場合、お 客様はこれらの規定に同意したものとみなされます。

- 3. 特定のコンポーネントに関する条件
 - a. ビルドサーバー 本ソフトウェアには、BuildServer.TXT ファイルに一覧されている複数の Build Server コンポーネントと、本 Microsoft ソフトウェアライセンス条項に続く BuildeServer リス トに一覧されているファイルが含まれている場合があります。これらの項目が本ソフトウェア に含まれている場合は、これらを複製してビルドコンピューターにインストールすることがで きます。お客様およびお客様の組織内の他のユーザーは、お客様のアプリケーションのコンパ イル、構築、検証、およびアーカイブと、構築プロセスの一環としての品質テストやパフォー マンステストを実行する目的に限り、ビルドコンピューターでこれらの項目を使用することが できます。
 - b. マイクロソフトプラットフォーム本ソフトウェアには、Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office、および Microsoft SharePointのコンポーネントが含まれていることがあります。これらのコンポーネン トには、本ソフトウェアに付属しているマイクロソフトの「Licenses」フォルダーに規定され ている、別途のライセンス条項および固有の製品サポートポリシーが適用されます。ただし、 関連するインストールディレクトリにこれらのコンポーネントのライセンス条項も含まれてい る場合は当該ライセンス条項が適用されます。
 - c. 第三者のコンポーネント 本ソフトウェアには、別途の法的通知を含みまたは別の契約が適用さ れる第三者のコンポーネントが含まれている場合があり、これらについては本ソフトウェアに 付属する ThirdPartyNotices ファイルに規定されています。かかるコンポーネントには、他の契 約が適用される場合でも、以下の保証の免責、損害賠償の制限および除外も適用されます。
 - d. パッケージマネージャー 本ソフトウェアには、他のマイクロソフトや第三者のソフトウェア パッケージをダウンロードしてお客様のアプリケーションで使用できるようにするパッケージ マネージャー (Nuget など) が含まれている場合があります。これらのパッケージには、独自の ライセンスが適用され、本契約は適用されません。マイクロソフトは、第三者のパッケージの 頒布、使用許諾、または保証の提供は行いません。
- 4. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありません。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。その他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた使用方法を制限する技術的制限に従うものとします。詳細については、「<u>https://docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1-installation-and-use-rights</u>」を参照してください。以下の行為は禁じられています。

- 本ソフトウェアの技術的な制限を回避すること。
- 本ソフトウェアのリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルを実行または試行すること。ただし、本ソフトウェアに含まれる場合がある一定のオープンソースコンポーネントの使用に適用される第三者のライセンス条項により求められている場合を除きます。
- Microsoft またはサプライヤーの告知を削除、最小化、ブロックまたは修正すること。
- 法律に違反する方法で本ソフトウェアを使用すること。
- 本ソフトウェアを共有、公開、レンタル、もしくはリースすること、本ソフトウェアを第三者 が使用できるようにスタンドアロンのホスト型ソリューションとして提供すること。
- 5. 輸出規制 お客様は、本ソフトウェアに適用されるすべての国内法および国際法 (輸出対象国、エ ンドユーザーおよびエンドユーザーによる使用に関する制限を含みます) を遵守しなければなりま せん。輸出規制の詳細については、(aka.ms/exporting) を参照してください。
- 5. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本 ソフトウェアに関してサポートサービスを提供しない場合があります。
- 7. 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネット ベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびサポートサー ビスについてのお客様とマイクロソフトとの間の完全なる合意を構成します。
- 8. 準拠法 お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解釈および契約 違反への主張は、米国ワシントン州法に準拠するものとします。他の主張については、お客様が 所在する地域の法律に準拠します。お客様が本ソフトウェアを他の国で入手した場合は、当該地 域の法律を準拠法とします。
- 9. 消費者の権利、地域による違い本契約は、特定の法的な権利を規定したものです。お客様は、地域や国によっては、消費者権利を含め、その他の権利を有する場合があります。Microsoftとお客様との関係とは別に、お客様が本ソフトウェアを取得した当事者に関する権利を有する場合もあります。本契約は、お客様の地域または国の法令が権利の変更を許容しない場合、それらのその他の権利を変更しないものとします。たとえば、お客様が本ソフトウェアを以下のいずれかの地域で取得した場合、または強行的な国の法令が適用される場合には、以下の規定がお客様に適用されます。
 - オーストラリアお客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではありません。
 - カナダ。本ソフトウェアをカナダで取得した場合、自働更新機能をオフにするか、お使いの機器をインターネットから取り外すか (ただし、インターネットに再接続すると、本ソフトウェアは更新プログラムのチェックとインストールを再開します)、または本ソフトウェアをアンインストールすることにより、更新受信を停止することができます。製品付属の文書がある場合

は、当該文書にお客様の特定のデバイスまたはソフトウェアの更新をオフにする方法が記載さ れていることもあります。

- ドイツおよびオーストリア
 - 保証 正規にライセンスを取得したソフトウェアは、本ソフトウェアに付属するマイクロソフトの資料の記載に実質的に従って動作します。ただし、マイクロソフトは、ライセンスを取得したソフトウェアに関して契約上の保証は一切いたしません。
 - 限定責任 故意、重過失、製品責任法に基づく請求があった場合、および死亡、人的または物 的損傷があった場合、Microsoft は、制定法にしたがって責任を負うものとします。

前掲条項 (ii) を条件とし、Microsoft が軽過失に該当する契約違反をして、同義務を履行する ことは本契約の正当な履行に資するものであって、同義務の違反は本契約の目的および当事 者が常に拠り所とする本契約への準拠を損なう (いわゆる「基本的義務」に違反する) 可能性 がある場合、Microsoft は当該の軽過失についてのみ責任を負うものとします。その他の軽過 失については、マイクロソフトは責任を負いません。

- 10法的効力 本契約は、特定の法的な権利を規定したものです。お客様は、地域または国の法律に よっては、その他の権利を有する場合があります。本ライセンス条項は、お客様の地域や国の法 律がその法律に基づく権利の変更を許容しない場合、それらの権利を変更しないものとします。 前項の制限にかかわらず、オーストラリアの場合、お客様は、オーストラリア消費者法に基づく 法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではあ りません。
- 11.保証の免責: 本ソフトウェアは "現状のまま" ライセンス供与されます。本ソフトウェアの使用に 伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いたしませ ん。お客様の地域の法律によって認められる範囲において、マイクロソフトは、商品性、特定目 的に対する適合性、および侵害の不存在に関する黙示の保証責任を負いません。
- 12損害賠償に関する制限および除外 YOU CAN RECOVER FROM MICROSOFT およびその供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、逸失利 益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負いませ ん。

この制限は、(a) 本ソフトウェア、サービス、第三者のインターネットのサイト上のコンテンツ (コードを含みます) または第三者のアプリケーションに関連した事項、および (b) 契約違反、保証 違反、厳格責任、過失、または不法行為等の請求 (適用される法令により認められている範囲にお いて) に適用されます。

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場 合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制 限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合がありま す。

11) Windows ベースの Docker イメージ – microsoft-windows-netfx3ondemand-package.cab

Microsoft ソフトウェア追加ライセンス条項

MICROSOFT .NET FRAMEWORK 3.5 SP1 FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) は、本追加ソフトウェアのラ イセンスをお客様に供与します。本追加ソフトウェアの基となるマイクロソフト Windows オペレー ティングシステムソフトウェア (以下「本ソフトウェア」といいます) を使用するためのラインセン スを取得している場合は、本追加ソフトウェアを使用できます。本ソフトウェアのライセンスを取得 していない場合は、本追加ソフトウェアを使用することはできません。お客様は、本ソフトウェアの 有効なライセンス取得済みの複製1部ごとに本追加ソフトウェアの複製を使用できます。

以下のライセンス条項は、本ソフトウェアの追加の使用条件について説明しています。これらの条項 と本ソフトウェアのライセンス条項が本追加ソフトウェアの使用に適用されます。両者の間に矛盾が ある場合は、本追加ライセンス条項が適用されます。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下 の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

- 1. 本追加ソフトウェアのサポート サービス マイクロソフトは、本ソフトウェアに対し て、<u>www.support.microsoft.com/common/international.aspx</u> に記載するサポートサービスを提供し ます。
- 2. MICROSOFT .NET のベンチマークテスト 本ソフトウェアには、Windows オペレーティング システムの .NET Framework、Windows Communication Foundation、Windows Presentation Foundation、および Windows Workflow Foundation のコンポーネント (以下「.NET コンポーネン ト」といいます) が含まれています。お客様は、これらのコンポーネントの内部ベンチマークテス トを実施することができます。お客様は、http://go.microsoft.com/fwlink/?LinkID=66406 に記載さ

れた条件に従うことを条件に、.NET コンポーネントのベンチマークテストの結果を開示すること ができます。

マイクロソフトと別段の合意があっても、お客様がかかるベンチマークテストの結果を公表した 場合、マイクロソフトは、<u>http://go.microsoft.com/fwlink/?LinkID=66406</u>の条件と同じ条件に従う ことを条件に、.NET コンポーネントと競合するお客様の製品についてマイクロソフトが実施した ベンチマークテストの結果を公表する権利を有します。

12) Windows ベースの Docker イメージ – dotnet-sdk

(https://https://github.com/dotnet/core/blob/main/LICENSE.TXT.)

MIT ライセンス (MIT)

Copyright (c) Microsoft Corporation

本ソフトウェアおよび関連するドキュメントファイル (以下「本ソフトウェア」といいます) のコ ピーを入手した誰に対しても、以下の条件を遵守することを条件として、本ソフトウェアを無料で無 制限に利用する権限を許可します。この権限には、本ソフトウェアを使用、複製、変更、マージ、公 開、頒布、サブライセンス、およびコピーを販売する権利と、本ソフトウェアの提供先のユーザーが 上記の権利を行使することを許可する権利が含まれますが、これらに限定されません。

上記の著作権表示とこの許可表示を、本ソフトウェアのすべてのコピーまたはかなりの部分に含める ものとします。

本ソフトウェアは「現状有姿」で提供され、明示または黙示を問わず、商品性、特定目的への適合 性、非侵害性の保証を含むいかなる種類の保証も伴いません。本ソフトウェアの使用またはその他の 取り扱いによって、あるいはこれに関連して生じたいかなる要求、損害、またはその他の法的責任に ついては、契約や不法行為などのいかなる場合においても、著者または著作権所有者はその責任を負 いません。

CodeBuild 条件キーを IAM サービスロール変数として使用してビ ルドアクセスを制御する

CodeBuild ビルド ARN を使用すると、コンテキストキーを使用して CodeBuild サービスロー ルのリソースアクセスの範囲を絞り込むことで、CodeBuildリソースアクセスを制限できま す。CodeBuild の場合、ビルドアクセス動作の制御に使用できるキーは codebuild:buildArnおよ び ですcodebuild:projectArn。ビルドプロジェクトの ARN を使用すると、リソースへの呼び出 しが特定のビルドプロジェクトから送信されたかどうかを確認できます。これを確認するには、IAM アイデンティティベースのポリシーで codebuild:buildArnまたは codebuild:projectArn条 件キーを使用します。

ポリシーで codebuild:buildArnまたは codebuild:projectArn条件キーを使用するには、任 意の ARN 条件演算子で条件として含めます。キーの値は、有効な ARN に解決される IAM 変数であ る必要があります。以下のポリシー例では、\${codebuild:projectArn} IAM 変数のプロジェク ト ARN を持つビルドプロジェクトへのアクセスのみが許可されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket-name/${codebuild:projectArn}/*"
        }
}
```

SDK を使用した CodeBuild のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で CodeBuild を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があ ります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアク ションは、関連するシナリオで確認できます。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください<u>AWS SDK</u> <u>でのこのサービスの使用</u>。このトピックには、使用開始方法に関する情報と、以前の SDK バージョ ンの詳細も含まれています。

コードの例

- SDK を使用した CodeBuild の基本的な例 AWS SDKs
 - SDK を使用した CodeBuild のアクション AWS SDKs
 - AWS SDK または CLI CreateProjectで を使用する
 - AWS SDK または CLI ListBuildsで を使用する
 - AWS SDK または CLI ListProjectsで を使用する
 - AWS SDK または CLI StartBuildで を使用する

SDK を使用した CodeBuild の基本的な例 AWS SDKs

次のコード例は、 SDKs AWS CodeBuild で AWS の基本を使用する方法を示しています。

例

- SDK を使用した CodeBuild のアクション AWS SDKs
 - AWS SDK または CLI CreateProjectでを使用する
 - AWS SDK または CLI ListBuildsで を使用する
 - AWS SDK または CLI ListProjectsでを使用する
 - AWS SDK または CLI StartBuildでを使用する

SDK を使用した CodeBuild のアクション AWS SDKs

次のコード例は、 AWS SDKs で個々の CodeBuild アクションを実行する方法を示しています。それ ぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されて います。

以下の例には、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。詳細な一覧については、 「AWS CodeBuild API リファレンス」を参照してください。

例

- AWS SDK または CLI CreateProjectで を使用する
- AWS SDK または CLI ListBuildsで を使用する
- AWS SDK または CLI ListProjectsで を使用する
- AWS SDK または CLI StartBuildで を使用する

AWS SDK または CLI CreateProjectで を使用する

以下のコード例は、CreateProject の使用方法を示しています。

CLI

AWS CLI

例 1: AWS CodeBuild ビルドプロジェクトを作成するには

次の create-project 例は、S3 バケットのソースファイルを使用して CodeBuild ビルドプ ロジェクトを作成します。

```
aws codebuild create-project \
    --name "my-demo-project" \
    --source "{\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-
input-bucket/my-source.zip\"}" \
    --artifacts {"\"type\": \"S3\",\"location\": \"codebuild-us-
west-2-123456789012-output-bucket\""} \
    --environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/
standard:1.0\",\"computeType\": \"BUILD_GENERAL1_SMALL\"}" \
    --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role"
```

出力:

```
{
    "project": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-
project",
        "name": "my-cli-demo-project",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "lastModified": 1556839783.274,
        "badge": {
            "badgeEnabled": false
        },
        "queuedTimeoutInMinutes": 480,
        "environment": {
            "image": "aws/codebuild/standard:1.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "type": "LINUX_CONTAINER",
            "imagePullCredentialsType": "CODEBUILD",
            "privilegedMode": false,
            "environmentVariables": []
        },
        "artifacts": {
            "location": "codebuild-us-west-2-123456789012-output-bucket",
            "name": "my-cli-demo-project",
            "namespaceType": "NONE",
            "type": "S3",
            "packaging": "NONE",
            "encryptionDisabled": false
        },
        "source": {
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
            "insecureSsl": false
        },
        "timeoutInMinutes": 60,
        "cache": {
            "type": "NO_CACHE"
        },
        "created": 1556839783.274
    }
```

}

例 2: パラメータの JSON 入力ファイルを使用して AWS CodeBuild ビルドプロジェクトを作 成するには

次の create-project 例では、JSON 入力ファイルにすべての必須パラメータを渡して CodeBuild ビルドプロジェクトを作成します。--generate-cli-skeleton parameter の みを含むコマンドを実行して、入力ファイルテンプレートを作成します。

aws codebuild create-project --cli-input-json file://create-project.json

入力 JSON ファイル create-project.json には、以下の内容が含まれます。

```
{
    "name": "codebuild-demo-project",
    "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "artifacts": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-output-bucket"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:1.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "serviceIAMRole"
}
```

出力:

```
{
    "project": {
        "name": "codebuild-demo-project",
        "serviceRole": "serviceIAMRole",
        "tags": [],
        "artifacts": {
            "packaging": "NONE",
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-output-bucket",
            "
```

```
"name": "message-util.zip"
        },
        "lastModified": 1472661575.244,
        "timeoutInMinutes": 60,
        "created": 1472661575.244,
        "environment": {
            "computeType": "BUILD_GENERAL1_SMALL",
            "image": "aws/codebuild/standard:1.0",
            "type": "LINUX_CONTAINER",
            "environmentVariables": []
        },
        "source": {
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
        },
        "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
        "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
    }
}
```

詳細については、AWS「 CodeBuild <u>ユーザーガイド」の「ビルドプロジェクトの作成 (AWS</u> <u>CLI)</u>」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「<u>CreateProject</u>」を参照してく ださい。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、<u>AWS コード例リポ</u> ジトリでの設定と実行の方法を確認してください。

プロジェクトを作成します。

import {
 ArtifactsType,
```
CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";
// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});
  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of
 "computeType" and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-
ref-available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
```

```
type: SourceType.GITHUB,
      // The location of the repository that contains the source code to be
built.
      location: githubUrl,
    },
  }),
 );
console.log(response);
     {
//
//
        '$metadata': {
          httpStatusCode: 200,
11
11
          requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
11
          extendedRequestId: undefined,
          cfId: undefined,
11
11
          attempts: 1,
11
          totalRetryDelay: 0
11
        },
11
        project: {
11
          arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxx:project/MyCodeBuilder',
11
          artifacts: {
11
            encryptionDisabled: false,
11
            location: 'xxxxx-xxxxxx-xxxxx',
11
            name: 'MyCodeBuilder',
11
            namespaceType: 'NONE',
11
            packaging: 'NONE',
//
            type: 'S3'
11
          },
11
          badge: { badgeEnabled: false },
11
          cache: { type: 'NO_CACHE' },
//
          created: 2023-08-18T14:46:48.979Z,
11
          encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxx:alias/aws/s3',
11
          environment: {
            computeType: 'BUILD_GENERAL1_SMALL',
11
11
            environmentVariables: [],
11
            image: 'aws/codebuild/standard:7.0',
            imagePullCredentialsType: 'CODEBUILD',
11
11
            privilegedMode: false,
11
            type: 'LINUX_CONTAINER'
11
          },
11
          lastModified: 2023-08-18T14:46:48.979Z,
11
          name: 'MyCodeBuilder',
11
          projectVisibility: 'PRIVATE',
11
          queuedTimeoutInMinutes: 480,
11
          serviceRole: 'arn:aws:iam::xxxxxxxxxxrrole/CodeBuildAdmin',
```

```
11
           source: {
 11
             insecureSsl: false,
             location: 'https://...',
 11
 //
             reportBuildStatus: false,
 11
             type: 'GITHUB'
 11
           },
           timeoutInMinutes: 60
 11
         }
 //
 11
       }
 return response;
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の 「<u>CreateProject</u>」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください<u>AWS SDK</u> <u>でのこのサービスの使用</u>。このトピックには、使用開始方法に関する情報と、以前の SDK バージョ ンの詳細も含まれています。

AWS SDK または CLI ListBuildsで を使用する

以下のコード例は、ListBuildsの使用方法を示しています。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、<u>AWS コード例リポ</u> ジトリでの設定と実行の方法を確認してください。

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);
    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }
        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
 codeBuildClient.ListBuilds(
                listBuildsRequest);
        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {
                std::cout << "Information about each build:" << std::endl;</pre>
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
 codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);
                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
 getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." <</pre>
 std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;</pre>
                    }
                } else {
                    std::cerr << "Error getting builds"</pre>
                              << getBuildsOutcome.GetError().GetMessage() <<
 std::endl;
                    return false;
```

```
}
            } else {
                 std::cout << "No builds found." << std::endl;</pre>
            }
            // Get the next token for pagination.
            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"</pre>
                       << listBuildsOutcome.GetError().GetMessage()
                       << std::endl;
            return false;
        }
    } while (!nextToken.
            empty()
            );
    return true;
}
```

• API の詳細については、「AWS SDK for C++ API リファレンス」の「<u>ListBuilds</u>」を参照し てください。

```
CLI
```

AWS CLI

AWS CodeBuild ビルド IDs。

次の list-builds の例では、昇順にソートされた CodeBuild ID のリストを取得します。

aws codebuild list-builds --sort-order ASCENDING

出力には、より多くの出力が利用できることを示す nextToken 値が含まれます。

```
"nextToken": "4AEA6u7J...The full token has been omitted for
brevity...MzY2OA==",
   "ids": [
      "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
      "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
           ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
    ]
}
```

このコマンドを再度実行し、前のレスポンスの nextToken 値をパラメータとして指定して、 出力の次の部分を取得します。レスポンスに nextToken 値が返されなくなるまで繰り返しま す。

```
aws codebuild list-builds --sort-order ASCENDING --next-
token 4AEA6u7J...The full token has been omitted for brevity...MzY20A==
```

出力の次の部分:

```
{
    "ids": [
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
        ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
    ]
}
```

詳細については、AWS 「 CodeBuild ユーザーガイド<u>」のIDs のリストを表示する (AWS</u> <u>CLI)</u>」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「<u>ListBuilds</u>」を参照してく ださい。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください<u>AWS SDK</u> <u>でのこのサービスの使用</u>。このトピックには、使用開始方法に関する情報と、以前の SDK バージョ ンの詳細も含まれています。

AWS SDK または CLI ListProjectsで を使用する

以下のコード例は、ListProjects の使用方法を示しています。

C++

SDK for C++

1 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、<u>AWS コード例リポ</u> ジトリ</mark>での設定と実行の方法を確認してください。

```
//! List the CodeBuild projects.
/*!
 \param sortType: 'SortOrderType' type.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
 sortType,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);
    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);
    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;
    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }
        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(),
 projects.end());
```

• API の詳細については、「AWS SDK for C++ API リファレンス」の「<u>ListProjects</u>」を参照 してください。

CLI

AWS CLI

AWS CodeBuild ビルドプロジェクト名のリストを取得するには。

次の list-projects の例では、CodeBuild ビルドプロジェクトを名前で昇順にソートした リストを取得します。

aws codebuild list-projects --sort-by NAME --sort-order ASCENDING

出力には、より多くの出力が利用できることを示す nextToken 値が含まれます。

```
{
    "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
    "projects": [
        "codebuild-demo-project",
```

このコマンドを再度実行し、前のレスポンスの nextToken 値をパラメータとして指定して、 出力の次の部分を取得します。レスポンスに nextToken 値が返されなくなるまで繰り返しま す。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
{
    "projects": [
        "codebuild-demo-project100",
        "codebuild-demo-project101",
        ... The full list of build project names has been omitted for brevity ...
        "codebuild-demo-project122"
]
}
```

詳細については、AWS 「 CodeBuild ユーザーガイド<u>」の「ビルドプロジェクト名のリストを</u> 表示する (AWS CLI)」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「<u>ListProjects</u>」を参照して ください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください<u>AWS SDK</u> <u>でのこのサービスの使用</u>。このトピックには、使用開始方法に関する情報と、以前の SDK バージョ ンの詳細も含まれています。

AWS SDK または CLI StartBuildで を使用する

以下のコード例は、StartBuild の使用方法を示しています。

C++

SDK for C++

```
1 Note
```

GitHub には、その他のリソースもあります。用例一覧を検索し、<u>AWS コード例リポ</u> ジトリでの設定と実行の方法を確認してください。

```
//! Start an AWS CodeBuild project build.
/*!
 \param projectName: A CodeBuild project name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);
    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);
    Aws::CodeBuild::Model::StartBuildOutcome outcome =
 codeBuildClient.StartBuild(
            startBuildRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;</pre>
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()</pre>
                  << std::endl;
    }
    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()</pre>
                  << std::endl;
    }
    return outcome.IsSuccess();
}
```

• API の詳細については、「AWS SDK for C++ API リファレンス」の「<u>StartBuild</u>」を参照し てください。

CLI

AWS CLI

AWS CodeBuild ビルドプロジェクトのビルドの実行を開始するには。

次の start-build の例では、指定した CodeBuild プロジェクトのビルドを開始します。ビ ルドは、タイムアウトするまでにビルドをキューに入れることができる分数に関するプロジェ クト設定と、プロジェクトのアーティファクト設定の両方を上書きします。

```
aws codebuild start-build \
    --project-name "my-demo-project" \
    --queued-timeout-in-minutes-override 5 \
    --artifacts-override {"\"type\": \"S3\",\"location\":
    \"arn:aws:s3:::artifacts-override\",\"overrideArtifactName\":true"}
```

出力:

```
{
    "build": {
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "buildStatus": "IN_PROGRESS",
        "buildComplete": false,
        "projectName": "my-demo-project",
        "timeoutInMinutes": 60,
        "source": {
            "insecureSsl": false,
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
        },
        "queuedTimeoutInMinutes": 5,
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "currentPhase": "QUEUED",
        "startTime": 1556905683.568,
        "environment": {
            "computeType": "BUILD_GENERAL1_MEDIUM",
            "environmentVariables": [],
```

```
"type": "LINUX_CONTAINER",
            "privilegedMode": false,
            "image": "aws/codebuild/standard:1.0",
            "imagePullCredentialsType": "CODEBUILD"
        },
        "phases": [
            {
                "phaseStatus": "SUCCEEDED",
                "startTime": 1556905683.568,
                "phaseType": "SUBMITTED",
                "durationInSeconds": 0,
                "endTime": 1556905684.524
            },
            {
                "startTime": 1556905684.524,
                "phaseType": "QUEUED"
            }
        ],
        "logs": {
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logEvent:group=null;stream=null"
        },
        "artifacts": {
            "encryptionDisabled": false,
            "location": "arn:aws:s3:::artifacts-override/my-demo-project",
            "overrideArtifactName": true
        },
        "cache": {
            "type": "NO_CACHE"
        },
        "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
        "initiator": "my-aws-account-name",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
    }
}
```

```
詳細については、AWS 「 CodeBuild <u>ユーザーガイド」の「ビルドの実行 (AWS CLI)</u>」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「<u>StartBuild</u>」を参照してく ださい。 AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください<u>AWS SDK</u> <u>でのこのサービスの使用</u>。このトピックには、使用開始方法に関する情報と、以前の SDK バージョ ンの詳細も含まれています。

トラブルシューティング AWS CodeBuild

このトピックの情報を使用して、問題を特定、診断、対処します。CodeBuild のビルドのログ記録と モニタリングを行い、問題のトラブルシューティングを行う方法については、「<u>ログ記録とモニタリ</u> ング」を参照してください。

トピック

- Apache Maven が間違ったリポジトリのアーティファクトを参照している
- ビルドコマンドがデフォルトでルートとして実行される
- ファイル名に英語以外の文字が含まれているとビルドが失敗する場合があります。
- Amazon EC2 パラメータストアからパラメータを取得する際にビルドが失敗する場合がある
- CodeBuild コンソールでブランチフィルタにアクセスできない
- ビルドの成功または失敗を表示できない
- ビルドステータスがソースプロバイダに報告されない
- Windows Server Core 2019 プラットフォームの基本イメージが見つからず、選択できない
- buildspec ファイルの以前のコマンドが、以降のコマンドで認識されない
- エラー: キャッシュのダウンロード時に「アクセスが拒否される」
- エラー:「BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE」カスタムビルドイメージを使用した 場合
- エラー:「ビルドの完了前にビルドコンテナの停止が検出されました。ビルドコンテナがメモリ不 足のため停止したか、Docker イメージがサポートされていません」 エラーコード: 500」
- <u>Error: "Cannot connect to the Docker daemon" when running a build (ビルドの実行時に「Docker</u> デーモンに接続できません」)
- エラー:ビルドプロジェクトを作成または更新する際、「CodeBuild は sts:AssumeRole の実行を 許可されていません」
- エラー:「GetBucketAcl の呼び出しエラー: バケット所有者が変更されたか、サービスロールに は、呼び出された s3:GetBucketAcl へのアクセス許可がありません」
- <u>エラー: ビルドの実行時に「アーティファクトのアップロードに失敗しました: 無効な ARN」</u>
- エラー:「Git のクローンに失敗しました: 'your-repository-URL' にアクセスできません: SSL 証明 書の問題: 自己署名証明書」
- エラー: ビルドの実行時に「アクセスしようとしているバケットは、指定されたエンドポイントを 使用してアドレス指定する必要があります」

- エラー: "このビルドイメージではランタイムバージョンを少なくとも1つ選択する必要があります。"
- ・ビルドキューのビルドが失敗するとエラー「QUEUED:INSUFFICIENT_SUBNET」が表示される
- エラー:「キャッシュをダウンロードできません: RequestError: 次の理由により送信リクエスト が失敗しました: x509: システムのルートをロードできませんでした。ルートが指定されていません」
- エラー:「S3 から証明書をダウンロードできません。AccessDenied"
- エラー: 「認証情報を見つけることができません」
- ・ プロキシサーバーで CodeBuild を実行しているときの RequestError タイムアウトエラー
- ・ ビルドイメージ内に Bourne シェル (sh) が必要
- エラー: CodeBuild コンソールを開くときに「JobWorker ID を確認できません」と表示される。
- ビルドの開始の失敗
- ローカルにキャッシュされたビルドの GitHub メタデータへのアクセス
- AccessDenied: レポートグループのバケット所有者が S3 バケットの所有者と一致しません。
- エラー: CodeConnections で CodeBuild プロジェクトを作成するときに、「認証情報に必要な権限 スコープが1つ以上ありません」
- エラー: Ubuntu install コマンドでビルドするときに「すみません、ターミナルがまったくリクエス トされていません - 入力を取得できません」

Apache Maven が間違ったリポジトリのアーティファクトを参照し ている

問題: AWS CodeBuildが提供する Java ビルド環境で Maven を使用すると、Maven は <u>https://</u> <u>repo1.maven.org/maven2</u>://www.a.jp の安全な中央 Maven リポジトリからビルドとプラグインの依 存関係を取得します。これは、ビルドプロジェクトの pom.xml ファイルが別の場所を使用すると明 示的に宣言した場合でも発生します。

考えられる原因: settings.xmlCodeBuild が提供する Java ビルド環境には、ビルド環境の / root/.m2 ディレクトリに事前にインストールされている という名前のファイルが含まれてい ます。この settings.xml には次の宣言が含まれています。これにより、Maven が常に https:// <u>repo1.maven.org/maven2</u> で、セキュアな中央 Maven リポジトリからビルドおよびプラグインの依 存関係を引き出すように指示されます。

```
<settings>
 <activeProfiles>
    <activeProfile>securecentral</activeProfile>
 </activeProfiles>
 <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
 </profiles>
</settings>
```

推奨される解決策: 以下を実行してください。

- 1. settings.xml ファイルをソースコードに追加します。
- Consettings.xml ファイルでは、前述の settings.xml 形式をガイドとして使用して、Maven が代わりにビルドとプラグインの依存関係を取得するリポジトリを宣言します。
- ビルドプロジェクトの install フェーズで、settings.xml ファイルをビルド環境の / root/.m2 ディレクトリにコピーするよう CodeBuild に指示します。たとえば、この動作を示 す buildspec.yml ファイルの次のスニペットを考えてみましょう。

```
version 0.2
phases:
    install:
        commands:
            - cp ./settings.xml /root/.m2/settings.xml
```

ビルドコマンドがデフォルトでルートとして実行される

Issue: AWS CodeBuild ルートユーザーとしてビルドコマンドを実行します。これは、関連するビル ドイメージの Dockerfile によって USER インストラクションが別のユーザーに設定された場合でも発 生します。

原因: CodeBuild は、デフォルトですべてのビルドコマンドをルートユーザーとして実行します。

推奨される解決策:なし。

ファイル名に英語以外の文字が含まれているとビルドが失敗する場 合があります

問題: 英語以外の文字 (漢字など) を含むファイル名のファイルを使用するビルドを実行すると、ビル ドが失敗します。

考えられる原因: AWS CodeBuild によって提供されるビルド環境は、デフォルトのロケールが POSIX に設定されています。POSIX ローカリゼーション設定は、米国英語以外の文字を含む CodeBuild やファイル名との互換性が低く、関連するビルドが失敗する可能性があります。

推奨される解決策: buildspec ファイルの pre_build セクションに次のコマンドを追加します。こ れらのコマンドは、ビルド環境のローカライゼーション設定として米国英語 UTF-8 を使用するよ う設定します。これは、米国英語以外の文字を含む CodeBuild やファイル名と高い互換性がありま す。

Ubuntu ベースのビルド環境の場合:

pre_build: commands:

- export LC_ALL="en_US.UTF-8"
- locale-gen en_US en_US.UTF-8

- dpkg-reconfigure -f noninteractive locales

Amazon Linux ベースのビルド環境の場合:

pre_build: commands: - export LC_ALL="en_US.utf8"

Amazon EC2 パラメータストアからパラメータを取得する際にビ ルドが失敗する場合がある

問題: ビルドが Amazon EC2 パラメータストアに保存されている 1 つ以上のパラメータの値を取得 しようとすると、DOWNLOAD_SOURCE フェーズで「Parameter does not exist」というエラー が発生し、ビルドが失敗する。

考えられる原因: ビルドプロジェクトが依存するサービスロールに ssm:GetParametersアクショ ンを呼び出すアクセス許可がないか、ビルドプロジェクトが によって生成 AWS CodeBuild され、 ssm:GetParametersアクションの呼び出しを許可するサービスロールを使用しますが、パラメー タには で始まらない名前があります/CodeBuild/。

推奨される解決策:

 CodeBuild によって生成されていないサービスロールの場合は、その定義を更新して CodeBuild が ssm:GetParameters アクションを呼びだせるようにします。たとえば、次のポリシーステー トメントでは、ssm:GetParameters アクションを呼び出して /CodeBuild/ で始まる名前を持 つパラメータを取得できます。

• CodeBuild によって生成されたサービスロールの場合は、その定義を更新して、 Amazon EC2 パ ラメータストア内の /CodeBuild/ で始まる名前以外の名前のパラメータに CodeBuild がアクセ スできるようにします。たとえば、次のポリシーステートメントでは、ssm:GetParameters ア クションを呼び出して指定された名前のパラメータを取得できます。

CodeBuild コンソールでブランチフィルタにアクセスできない

問題: AWS CodeBuild プロジェクトを作成または更新するときに、ブランチフィルターオプション をコンソールで使用することはできません。

考えられる原因: ブランチフィルタオプションは廃止されました。このオプションはウェブフック フィルタグループに置き換えられています。これにより、CodeBuild の新しいビルドをトリガーする ウェブフックイベントの制御が強化されます。

推奨される解決策: ウェブフックフィルタの導入前に作成したブランチフィルタを移行するには、 正規表現 HEAD_REF で ^refs/heads/*branchName*\$ フィルタを使用してウェブフックフィ ルタグループを作成します。たとえば、ブランチフィルタの正規表現が ^branchName\$ の場 合、HEAD_REF フィルタに入力する更新後の正規表現は ^refs/heads/branchName\$ です。詳細 については、「<u>Bitbucket ウェブフックイベント</u>」および「<u>GitHub ウェブフックイベントのフィルタ</u> リング (コンソール)」を参照してください。

ビルドの成功または失敗を表示できない

問題: 再試行されたビルドの成功または失敗を確認できない。

考えられる原因: ビルドのステータスを報告するオプションが有効になっていません。

推奨される解決策: CodeBuild プロジェクトを作成または更新するときは、[ビルドのステータ スを報告] を有効にします。このオプションは、ビルドをトリガーするときに CodeBuild にス テータスを報告するように指示します。詳細については、AWS CodeBuild API リファレンスの 「ReportBuildStatus」を参照してください。

ビルドステータスがソースプロバイダに報告されない

問題: GitHub や Bitbucket などのソースプロバイダーへのビルドステータスのレポートを許可した後 で、ビルドステータスが更新されない。

考えられる原因: ソースプロバイダーに関連付けられたユーザーに、リポジトリへの書き込みアクセ ス許可がありません。

推奨される解決策: ソースプロバイダーにビルドステータスを報告できるようにするには、ソースプ ロバイダーに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があり ます。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。 詳細については、「ソースプロバイダーのアクセス」を参照してください。

Windows Server Core 2019 プラットフォームの基本イメージが見 つからず、選択できない

問題: Windows Server Core 2019 プラットフォームの基本イメージを検索または選択できない。

考えられる原因: このイメージをサポートしていない AWS リージョンを使用している。

推奨される解決策: Windows Server Core 2019 プラットフォームの基本イメージがサポートされて いる、次のいずれかの AWS リージョンを使用します。

- 米国東部(バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- 欧州 (アイルランド)

buildspec ファイルの以前のコマンドが、以降のコマンドで認識されない

問題: buildspec ファイルの1つ以上のコマンドの結果が、同じ buildspec ファイルの以降のコマンド で認識されない。たとえば、コマンドによってローカル環境変数が設定される場合がありますが、後 で実行されるコマンドがそのローカル環境変数の値を取得できない可能性があります。 考えられる原因: buildspec ファイルバージョン 0.1 では、 AWS CodeBuild はビルド環境のデフォル トシェルの各インスタンスで各コマンドを実行します。つまり、各コマンドは他のすべてのコマンド とは独立して実行されます。デフォルトでは、以前のコマンドの状態に依存する単一のコマンドを実 行することはできません。

推奨される解決策: buildspec バージョン 0.2 を使用してください。これにより、問題が解決されま す。何らかの理由で buildspec バージョン 0.1 を使用する必要がある場合は、シェルコマンド連鎖演 算子 (Linux の && など) を使用して、複数のコマンドを 1 つのコマンドにまとめることをお勧めしま す。または、複数のコマンドを含むソースコードにシェルスクリプトを組み込み、そのシェルスク リプトを buildspec ファイルの 1 つのコマンドから呼び出します。詳細については、「ビルド環境の シェルとコマンド」および「ビルド環境の環境変数」を参照してください。

エラー: キャッシュのダウンロード時に「アクセスが拒否される」

問題: キャッシュが有効になっているビルドプロジェクトでキャッシュをダウンロードしようとする と、Access denied エラーが発生する。

考えられる原因:

- ビルドプロジェクトの一部としてキャッシングが設定されています。
- キャッシュは、InvalidateProjectCache API により最近無効化されています。
- CodeBuild を使用しているサービスロールには、キャッシュを保持する S3 バケットへの s3:Get0bjectアクセス許可と s3:Put0bject アクセス許可がありません。

推奨される解決策: キャッシュ設定を更新した直後に初めて使用する場合、このエラーが表示さ れるのは普通です。このエラーが解消されない場合は、キャッシュを保持する S3 バケットへの s3:Get0bject アクセス許可と s3:Put0bject アクセス許可が、サービスロールに付与されてい るかどうかを確認する必要があります。詳細については、「Amazon S3 デベロッパーガイド」の 「S3 アクセス許可の指定」を参照してください。

エラー: 「BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE」カス タムビルドイメージを使用した場合

問題: カスタムビルドイメージを使用するビルドを実行しようとすると、ビルドは BUILD_CONTAINER_UNABLE_T0_PULL_IMAGE というエラーで失敗する。 考えられる原因: ビルドイメージの全体的な非圧縮サイズが、ビルド環境のコンピューティングタイ プの使用可能ディスクスペースよりも大きい。ビルドイメージのサイズを確認するには、Docker を 使用して docker images REPOSITORY:TAG コマンドを実行します。コンピューティングタイプ で使用可能なディスク容量のリストについては、「ビルド環境のコンピューティングモードおよびタ イプ」を参照してください。

推奨される解決策: 使用可能なディスク容量の大きなコンピューティングタイプを使用するか、 カスタムビルドイメージのサイズを縮小します。

考えられる原因: AWS CodeBuild Amazon Elastic Container Registry (Amazon ECR) からビルドイ メージをプルするアクセス許可がありません。

推奨される解決策: CodeBuild がカスタムビルドイメージをビルド環境にプルできるよう

に、Amazon ECR のリポジトリの権限を更新します。詳細については、「<u>Amazon ECR のサン</u> プル」を参照してください。

考えられる原因: リクエストした Amazon ECR イメージは、 AWS アカウントが使用している AWS リージョンでは使用できません。

推奨される解決策: AWS アカウントが使用しているリージョンと同じ AWS リージョンにある Amazon ECR イメージを使用します。

考えられる原因:パブリックインターネットアクセスのない VPC でプライベートレジストリを使用し ています。CodeBuild は、VPC 内のプライベート IP アドレスからイメージを取得できません。詳細 については、「<u>CodeBuild AWS Secrets Manager のサンプルを含むプライベートレジストリ</u>」を参 照してください。

推奨される解決策: VPC でプライベートレジストリを使用する場合は、VPC にパブリックイン ターネットアクセスがあることを確認してください。

考えられる原因: エラーメッセージに「toomanyrequests」が含まれており、イメージを Docker Hub から取得した場合、このエラーは Docker Hub のプル制限に達したことを意味します。

推奨される解決策: Docker Hub のプライベートレジストリを使用するか、Amazon ECR からイ メージを取得します。プライベートレジストリの使用の詳細については、「<u>CodeBuild AWS</u> <u>Secrets Manager のサンプルを含むプライベートレジストリ</u>」を参照してください。Amazon ECR の使用方法の詳細については、「<u>CodeBuild の Amazon ECR サンプル</u>」を参照してくださ い。

エラー: 「ビルドの完了前にビルドコンテナの停止が検出されま した。ビルドコンテナがメモリ不足のため停止したか、Docker イ メージがサポートされていません」 エラーコード: 500」

問題: で Microsoft Windows または Linux コンテナを使用しようとすると AWS CodeBuild、このエ ラーは PROVISIONING フェーズ中に発生します。

考えられる原因:

- コンテナ OS バージョンが CodeBuild でサポートされていません。
- HTTP_PROXY、HTTPS_PROXY、またはその両方がコンテナで指定されます。

推奨される解決策:

- Microsoft Windows の場合は、Windows コンテナを、コンテナ OS バージョン microsoft/ windowsservercore:10.0.x (microsoft/windowsservercore:10.0.14393.2125 など) で使用します。
- Linux の場合は、Docker イメージの HTTP_PROXY 設定と HTTPS_PROXY 設定をクリアするか、ビルドプロジェクトで VPC 設定を指定します。

Error: "Cannot connect to the Docker daemon" when running a build (ビルドの実行時に「Docker デーモンに接続できません」)

問題:ビルドが失敗し、「Cannot connect to the Docker daemon at unix:/var/run/ docker.sock. Is the docker daemon running?」のようなエラーがビルドログに表示され る。

考えられる原因: 特権モードでビルドを実行していません。

推奨される解決策: このエラーを修正するには、特権モードを有効にし、次の手順を使用して buildspec を更新する必要があります。

特権モードでビルドを実行するには、次の手順に従います。

- 1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
- ナビゲーションペインで [ビルドプロジェクト] を選択し、該当するビルドプロジェクトを選択します。

エラー: 「ビルドの完了前にビルドコンテナの停止が検出されました。ビルドコンテナがメモリ不足のた め停止したか、Docker イメージがサポートされていません」 エラーコード: 500」

- 3. [Edit] (編集) から [Environment] (環境) を選択します。
- 4. [Additional configuration (追加設定)] を選択します。
- 5. [特権付与]から、[Docker イメージをビルドする場合、またはビルドで昇格された権限を取得す る場合は、このフラグを有効にする]を選択します。
- 6. [Update environment (環境の更新)] を選択します。
- 7. [ビルドの開始]を選択してビルドを再度実行します。

また、コンテナ内で Docker デーモンを起動する必要があります。buildspec の install フェーズ は、以下のようなものです。

phases:				
install:				
commands:				
- nohup /usr/local/bin/dockerdhost=unix:///var/run/docker.sock				
<pre>host=tcp://127.0.0.1:2375storage-driver=overlay2 &</pre>				
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"				

buildspec ファイルで参照される OverlayFS ストレージドライバーの詳細については、Docker ウェ ブサイトの「Use the OverlayFS storage driver」を参照してください。

Note

基本オペレーティングシステムが Alpine Linux である場合は、buildspec.yml で -t 引数 を timeout に追加します。

- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"

を使用して Docker イメージを構築および実行する方法の詳細については AWS CodeBuild、「」を 参照してください<u>CodeBuild のカスタム Docker イメージのサンプル</u>。

エラー:ビルドプロジェクトを作成または更新する際、

「CodeBuild は sts:AssumeRole の実行を許可されていません」

問題:ビルドプロジェクトを作成または更新しようとすると、エラー 「Code:InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::*account-ID*:role/*service-role-name*」 が発生する。

考えられる原因:

- AWS Security Token Service (AWS STS)は、ビルドプロジェクトを作成または更新しようとしている AWS リージョンで非アクティブ化されています。
- ビルドプロジェクトに関連付けられた AWS CodeBuild サービスロールが存在しない
 か、CodeBuild を信頼するための十分なアクセス許可がありません。
- ビルドプロジェクトに関連付けられた AWS CodeBuild サービスロールの大文字と小文字が、実際の IAM ロールと一致しません。

推奨される解決策:

- ビルドプロジェクトを作成または更新しようとしている AWS リージョンで AWS STS がアクティブ化されてい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「AWS Uージョン AWS STS での のアクティブ化と非アクティブ化」を参照してください。
- ターゲット CodeBuild サービスロールが AWS アカウントに存在することを確認します。コンソー ルを使用していない場合は、ビルドプロジェクトを作成または更新したときにサービスロールの Amazon リソースネーム (ARN) のスペルを間違えていないことを確認してください。IAM ロール では大文字と小文字が区別されるため、IAM ロールの大文字と小文字が正しいことを確認してく ださい。
- 対象の CodeBuild サービスロールに、CodeBuild を信頼するための十分な権限があることを確認 します。詳細については、CodeBuild が他の AWS のサービスとやり取りすることを許可の「信頼 関係のポリシーステートメント」を参照してください。

エラー: 「GetBucketAcl の呼び出しエラー: バケット所有者が変更 されたか、サービスロールには、呼び出された s3:GetBucketAcl へ のアクセス許可がありません」

問題: ビルドを実行すると、S3 バケット所有者や GetBucketAc1 アクセス許可の変更に関するエ ラーが発生する。

考えられる原因: s3:GetBucketAcl および s3:GetBucketLocation のアクセス許可を IAM ロー ルに追加しています。これらのアクセス許可は、プロジェクトの S3 バケットを保護し、バケットに アクセスできるユーザーを自分に限定します。これらのアクセス許可を追加した後で、S3 バケット の所有者が変更されています。

推奨される解決策: S3 バケットの所有者が自分であることを確認し、IAM ロールへのアクセス許可 を追加し直します。詳細については、「<u>S3 バケットへの安全なアクセス</u>」を参照してください。

エラー: ビルドの実行時に「アーティファクトのアップロードに失 敗しました: 無効な ARN」

問題:ビルドを実行すると、UPLOAD_ARTIFACTSビルドフェーズが失敗し、「Failed to upload artifacts: Invalid arn」というエラーが表示される。

考えられる原因: S3 出力バケット(がビルドからの出力 AWS CodeBuild を保存するバケット) が CodeBuild ビルドプロジェクトとは異なる AWS リージョンにある。

推奨される解決策: ビルドプロジェクトの設定を更新して、ビルドプロジェクトと同じ AWS リー ジョンにある出力バケットを参照します。

エラー: 「Git のクローンに失敗しました: **'your-repository-**URL' にアクセスできません: SSL 証明書の問題: 自己署名証明書」

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因: ソースリポジトリには自己署名証明書がありますが、S3 バケットから証明書をビ ルドプロジェクトの一部としてインストールする選択をしていません。

推奨される解決策:

- プロジェクトを編集します。[証明書] で [S3 から証明書をインストールする] を選択します。[証明 書のバケット] では、SSL 証明書が保存されている S3 バケットを選択します。[証明書のオブジェ クトキー] に、S3 オブジェクトキーの名前を入力します。
- プロジェクトを編集します。GitHub Enterprise Server プロジェクトリポジトリに接続するときの SSL 警告を無視するには、[Insecure SSL (安全でない SSL)]を選択します。

Note

[Insecure SSL] はテストのみに使用することが推奨されます。本番環境では使用しないで ください。

エラー: ビルドの実行時に「アクセスしようとしているバケット は、指定されたエンドポイントを使用してアドレス指定する必要が あります」

問題:ビルドを実行すると、DOWNLOAD_SOURCE ビルドフェーズが失敗し、「The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint」というエラーが表示される。

考えられる原因: 構築済みのソースコードは S3 バケットに保存され、そのバケットはビルドプロ ジェクトとは異なる AWS AWS CodeBuild リージョンにあります。

推奨される解決策: 構築済みのソースコードが含まれているバケットを指すように、ビルドプロジェ クトの設定を更新します。バケットがビルドプロジェクトと同じ AWS リージョンにあることを確認 します。

エラー: "このビルドイメージではランタイムバージョンを少なくと も 1 つ選択する必要があります。"

問題:ビルドを実行すると、DOWNLOAD_SOURCEビルドフェーズが失敗し、「YAML_FILE_ERROR: This build image requires selecting at least one runtime version」というエ ラーが表示される。

考えられる原因: ビルドでバージョン 1.0 以降の Amazon Linux 2 (AL2) 標準イメージ、またはバー ジョン 2.0 以降の Ubuntu 標準イメージが使用されていますが、buildspec ファイルでランタイムが 指定されていません。

推奨される解決策: aws/codebuild/standard:2.0 CodeBuild マネージド型イメージを使用する 場合は、buildspec ファイルの runtime-versions セクションでランタイムバージョンを指定する 必要があります。たとえば、PHP を使用するプロジェクトでは、次の buildspec ファイルを使用し ます。

version: 0.2
phases:
 install:
 runtime-versions:
 php: 7.3

build:					
commands:					
- phpversion					
artifacts:					
files:					
- README.md					

Note

runtime-versions セクションを指定して、Ubuntu 標準イメージ 2.0 以降や Amazon Linux 2 (AL2) 標準イメージ 1.0 以降以外のイメージを使用した場合は、ビルドで 「Skipping install of runtimes. Runtime version selection is not supported by this build image」の警告が表示されます。

詳細については、「Specify runtime versions in the buildspec file」を参照してください。

ビルドキューのビルドが失敗するとエラー

「QUEUED:INSUFFICIENT_SUBNET」が表示される

問題: ビルドキューのビルドが QUEUED: INSUFFICIENT_SUBNET のようなエラーで失敗する。

考えられる原因: VPC に指定された IPv4 CIDR ブロックが、リザーブド IP アドレスを使用してい る。各サブネット CIDR ブロックの最初の 4 つの IP アドレスと最後の IP アドレスは使用できず、 インスタンスに割り当てることができません。たとえば、CIDR ブロック 10.0.0.0/24 を持つサブ ネットの場合、次の 5 つの IP アドレスが予約されます。

- 10.0.0.0: ネットワークアドレスです。
- 10.0.0.1: VPC ルーター AWS 用に によって予約されています。
- 10.0.0.2:予約者 AWS。DNS サーバーの IP アドレスは、常に VPC ネットワークのベースに 2 を付加したものですが、各サブネット範囲のベースに 2 を付加したアドレスも予約されていま す。複数の CIDR ブロックを持つ VPC の場合、DNS サーバーの IP アドレスはプライマリ CIDR にあります。詳細については、Amazon VPC ユーザーガイドの「<u>Amazon DNS サーバー</u>」を参照 してください。
- 10.0.0.3: 将来の使用 AWS のために によって予約されています。
- 10.0.0.255: ネットワークブロードキャストアドレスです。VPC でのブロードキャストはサポートされていません。このアドレスは予約されています。

推奨される解決策: VPC でリザーブド IP アドレスが使用されていることを確認します。予約済みの IP アドレスを、予約されていないアドレスに置き換えます。詳細については、Amazon VPC ユー ザーガイドのVPC とサブネットのサイズ設定を参照してください。

エラー: 「キャッシュをダウンロードできません: RequestError: 次の理由により送信リクエストが失敗しました: x509: システムの ルートをロードできませんでした。ルートが指定されていません」

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因: ビルドプロジェクトの一部としてキャッシュを設定し、有効期限が切れたルート証 明書を含む古い Docker イメージを使用しています。

推奨される解決策: AWS CodeBuild プロジェクトで使用されている Docker イメージを更新しま す。詳細については、「CodeBuild に用意されている Docker イメージ」を参照してください。

エラー : 「S3 から証明書をダウンロードできませ

ん。AccessDenied"

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因:

- ・ 証明書に正しくない S3 バケットが選択されています。
- 証明書に誤ったオブジェクトキーが入力されています。

推奨される解決策:

- プロジェクトを編集します。[証明書のバケット] では、SSL 証明書が保存されている S3 バケット を選択します。
- プロジェクトを編集します。[証明書のオブジェクトキー] に、S3 オブジェクトキーの名前を入力します。

エラー:「認証情報を見つけることができません」

問題: を実行したり AWS CLI、 SDK を使用した AWS り、ビルドの一部として別の同様のコン ポーネントを呼び出したりしようとすると、 AWS CLI、 AWS SDK、または コンポーネントに直接 関連するビルドエラーが発生します。たとえば、「Unable to locate credentials」などのビ ルドエラーが発生する場合があります。

考えられる原因:

- ビルド環境の AWS CLI、 AWS SDK、または コンポーネントのバージョンに互換性がありません AWS CodeBuild。
- Docker を使用するビルド環境内で Docker コンテナを実行しており、コンテナはデフォルトで AWS 認証情報にアクセスできません。

推奨される解決策:

- ビルド環境の AWS CLI、 AWS SDK、または コンポーネントの次のバージョン以上であることを 確認します。
 - AWS CLI: 1.10.47
 - AWS SDK for C++: 0.2.19
 - AWS SDK for Go: 1.2.5
 - AWS SDK for Java: 1.11.16 [「]」
 - AWS SDK for JavaScript: 2.4.7
 - AWS SDK for PHP: 3.18.28 [[]]
 - AWS SDK for Python (Boto3): 1.4.0
 - AWS SDK for Ruby: 2.3.22
 - Botocore: 1.4.37
 - CoreCLR: 3.2.6-beta
 - Node.js: 2.4.7
- ビルド環境で Docker コンテナを実行する必要があり、コンテナに AWS 認証情報が必要な場合 は、ビルド環境からコンテナに認証情報を渡す必要があります。buildspec ファイルでは、以下の ような Docker run コマンドを組み込みます。この例では、aws s3 1s コマンドを使用して、使 用可能な S3 バケットを一覧表示します。-e オプションは、コンテナが AWS 認証情報にアクセ スするために必要な環境変数を渡します。

docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI yourimage-tag aws s3 ls

- Docker イメージを構築していて、ビルドに AWS 認証情報が必要な場合 (Amazon S3 からファイ ルをダウンロードする場合など)、次のようにビルド環境から Docker ビルドプロセスに認証情報 を渡す必要があります。
 - 1. Docker イメージ用ソースコードの Dockerfile に、次の ARG インストラクションを指定します。

ARG AWS_DEFAULT_REGION ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI

 buildspec ファイルでは、以下のような Docker build コマンドを組み込みます。--buildarg オプションは、Docker ビルドプロセスが AWS 認証情報にアクセスするために必要な環境 変数を設定します。

docker build --build-arg AWS_DEFAULT_REGION=\$AWS_DEFAULT_REGION --build-arg AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=\$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI t your-image-tag .

プロキシサーバーで CodeBuild を実行しているときの RequestError タイムアウトエラー

問題: 次のいずれかのような RequestError エラーが表示される。

- RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeoutCloudWatch Logs の。
- Error uploading artifacts: RequestError: send request failed caused by: Put https://your-bucket.s3.your-aws-region.amazonaws.com/*: dial tcp 52.219.96.208:443: connect: connection refusedAmazon S3 の。

考えられる原因:

- ssl-bump が適切に設定されていません。
- ・ 組織のセキュリティポリシーで ssl_bump を使用することが許可されていません。

・ buildspec ファイルに、proxy 要素を使用して指定されたプロキシ設定がありません。

推奨される解決策:

- ss1-bump が適切に設定されていることを確認します。プロキシサーバーに Squid を使用している場合は、「明示的なプロキシサーバーとしての Squid の設定」を参照してください。
- Amazon S3 および CloudWatch Logs のプライベートエンドポイントを使用するには、次の手順に 従います。
 - プライベートサブネットのルーティングテーブルで、インターネット用トラフィックをプロキ シサーバーにルーティングする追加済みのルールを削除します。詳細については、「Amazon VPC ユーザーガイド」の「VPC でサブネットを作成する」を参照してください。
 - プライベート Amazon S3 エンドポイントと CloudWatch Logs エンドポイントを作成し、 それらを Amazon VPC のプライベートサブネットに関連付けます。詳細については、 「Amazon VPC ユーザーガイド」の「VPC エンドポイントサービス」を参照してください。
 - Amazon VPC の [プライベート DNS 名を有効にする] が選択されていることを確認します。
 詳細については、 Amazon VPC ユーザーガイド のインターフェイスエンドポイントの作成 参照してください。
- 明示的なプロキシサーバーに ssl-bump を使用しない場合は、proxy 要素を使用して buildspec ファイルにプロキシ設定を追加します。詳細については、「<u>明示的なプロキシサーバーでの</u> CodeBuild の実行」および「buildspec の構文」を参照してください。

```
version: 0.2
proxy:
    upload-artifacts: yes
    logs: yes
phases:
    build:
        commands:
```

ビルドイメージ内に Bourne シェル (sh) が必要

問題: によって提供されていないビルドイメージを使用していて AWS CodeBuild、ビルドが失 敗してメッセージ が表示されるBuild container found dead before completing the build。 考えられる原因: Bourne シェル (sh) がビルドイメージに含まれていません。CodeBuild は、ビルド コマンドとスクリプトを実行するために、sh を必要とします。

推奨される解決策: ビルドイメージに sh が含まれていない場合、イメージを使用するビルドを開始 する前に必ずそれを含めてください (CodeBuild は、ビルドイメージに既に sh を含んでいます)。

警告 (ビルドの実行時): 「ランタイムのインストールをスキップし ます。このビルドイメージではランタイムバージョンの選択はサ ポートされていません」

問題: ビルドを実行すると、この警告がビルドログに表示される。

考えられる原因: ビルドでバージョン 1.0 以降の Amazon Linux 2 (AL2) 標準イメージ、またはバー ジョン 2.0 以降の Ubuntu 標準イメージが使用されておらず、buildspec ファイルの runtimeversions セクションにランタイムが指定されています。

推奨される解決策: buildspec ファイルに runtime-versions セクションが含まれていないことを 確認します。runtime-versions セクションは、Amazon Linux 2 (AL2) 標準イメージ以降または Ubuntu 標準イメージのバージョン 2.0 以降を使用する場合にのみ必要です。

エラー:CodeBuild コンソールを開くときに「JobWorker ID を確 認できません」と表示される。

問題: CodeBuild コンソールを開くと、「JobWorker の ID を確認できません」というエラーメッ セージが表示される。

考えられる原因: コンソールのアクセスに使用されている IAM ロールに、jobId をキーとするタグ があります。このタグキーは CodeBuild 用に予約されており、存在する場合にこのエラーを発生さ せます。

推奨される解決策: jobId キーを持つカスタム IAM ロールタグを変更して、jobIdentifier など の別のキーを持つようにします。

ビルドの開始の失敗

問題: ビルドを開始すると、「ビルドを開始できませんでした」というエラーメッセージが表示され る。 考えられる原因: 同時に実行できるビルド数の上限に達しています。

推奨される解決策: 他のビルドが完了するまで待つか、プロジェクトの同時実行のビルド制限を増や して、ビルドを再度開始します。詳細については、「プロジェクトの設定」を参照してください。

ローカルにキャッシュされたビルドの GitHub メタデータへのアク セス

問題: 状況によっては、キャッシュされたビルドの .git ディレクトリは、ディレクトリではなく、テ キストファイルである場合があります。

考えられる原因: ローカルソースキャッシュがビルドに対して有効になっている場合、CodeBuild は.git ディレクトリの gitlink を作成します。つまり、.git ディレクトリは、単にディレクトリへ のパスを含むテキストファイルです。

推奨される解決策: すべての場合において、次のコマンドを使用して Git メタデータディレクトリを 取得します。このコマンドは、.git のフォーマットに関係なく機能します。

git rev-parse --git-dir

AccessDenied: レポートグループのバケット所有者が S3 バケット の所有者と一致しません。

問題: Amazon S3 バケットにテストデータをアップロードしたときに、CodeBuild がバケットにテス トデータを書き込むことができない。

考えられる原因:

- レポートグループのバケット所有者に指定されたアカウントが、Amazon S3 バケットの所有者と 一致しません。
- サービスロールには、バケットへの書き込みアクセスがありません。

推奨される解決策:

Amazon S3 バケットの所有者と一致するよう、レポートグループのバケット所有者を変更します。

Amazon S3 バケットへの書き込みアクセスを許可するようにサービスロールを変更します。

エラー: CodeConnections で CodeBuild プロジェクトを作成すると きに、「認証情報に必要な権限スコープが 1 つ以上ありません」

問題: CodeConnections を使用して CodeBuild CodeBuild プロジェクトを作成する場合、Bitbucket ウェブフックをインストールするアクセス許可がありません。

考えられる原因:

• 新しいアクセス許可の範囲が Bitbucket アカウントで受け入れられていない可能性があります。

推奨される解決策:

- 新しいアクセス許可を受け入れるには、「必要なアクション」 Bitbucket によって送信され た AWS CodeStar のスコープが変更されている「」という件名の E メールを受信しているはず ですnotifications-noreply@bitbucket.org。E メールには、既存の CodeConnections Bitbucket アプリのインストールにウェブフックのアクセス許可を付与するためのリンクが含まれ ています。
- Eメールが見つからない場合は、に移動するかhttps://bitbucket.org/site/addons/ reauthorize?account=<workspace-name>&addon_key=aws-codestar、ウェブフッ クのアクセス許可を付与するワークスペースhttps://bitbucket.org/site/addons/ reauthorize?addon_key=aws-codestarを選択して、アクセス許可を付与できます。

	aws	$\stackrel{\leftarrow}{\rightarrow}$				
AWS CodeStar requests access This app is hosted at https://codestar-connections.webhooks.aws						
Read your acco	ount informa	tion				
Read and modify your repositories and their pull requests						
Administer your repositories						
Read and modify your repositories' webhooks						
Authorize for wo	orkspace					
				~		
Allow AWS CodeStar to do this?						
This 3rd party vendor has not provided a privacy policy or terms of use.						
Auassian's Privac	y Policy is not a	pplicable to	the use of this Ap	ip.		
				_		
			Grant acces	s Cancel		

エラー: Ubuntu install コマンドでビルドするときに「すみません、 ターミナルがまったくリクエストされていません - 入力を取得でき ません」

問題: GPU コンテナ特権ビルドを実行している場合は、<u>以下の手順</u>で NVIDIA Container Toolkit をインストールしている可能性があります。最新の CodeBuild イメージリリースでは、CodeBuild は最新のubuntu厳選されたイメージnvidia-container-toolkitに Docker をプリインストー ルamazonlinuxして設定します。この手順を実行すると、Ubuntu install コマンドを使用したビルド が失敗し、次のエラーが発生します。

Running command curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | gpg -dearmor --no-tty -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg gpg: Sorry, no terminal at all requested - can't get input curl: (23) Failed writing body

考えられる原因: gpg キーは同じ場所に既に存在します。
推奨される解決策: nvidia-container-toolkitはイメージに既にインストールされています。 このエラーが表示された場合は、buildspec で docker のインストールと再起動プロセスをスキップ できます。

のクォータ AWS CodeBuild

次の表に、現在のクォータを示します AWS CodeBuild。これらのクォータは、特に指定がない限り AWS 、アカウントごとにサポートされている各 AWS リージョンのものです。

Service Quotas

以下は、 AWS CodeBuild サービスのデフォルトのクォータです。

名前	デフォルト	引き上げ可能	説明
プロジェクトごとの関連タグ	サポートされてい る各リージョン: 50	い い え	ビルドプロジェクトに関 連付けることができるタ グの最大数
ビルドプロジェクト	サポートされてい る各リージョン: 5,000	<u>あ</u> り	ビルドするプロジェクト の最大数
ビルドのタイムアウト (分)。	サポートされてい る各リージョン: 2,160	い い え	ビルドの最大タイムアウ ト (分)
ビルドに関する情報に対する同時要求	サポートされてい る各リージョン: 100	い い え	CLI または AWS SDK AWS を使用して、一度 に に関する情報をリクエ ストできるビルドの最大 数。
ビルドプロジェクトに関する情報の同 時要求	サポートされてい る各リージョン: 100	い い え	CLI または AWS SDK AWS を使用して、一度に に関する情報をリクエス

名前	デフォルト	引き上げ可能	説明
			トできるビルドプロジェ クトの最大数。
ARM Lambda/10GB 環境向けの同時実 行ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM Lambda/10GB 環境 向けの同時実行ビルドの 最大数
ARM Lambda/1GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>ぁ</u> り	ARM Lambda/1GB 環境 向けの同時実行ビルドの 最大数
ARM Lambda/2GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM Lambda/2GB 環境 向けの同時実行ビルドの 最大数
ARM Lambda/4GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM Lambda/4GB 環境 向けの同時実行ビルドの 最大数
ARM Lambda/8GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM Lambda/8GB 環境 向けの同時実行ビルドの 最大数
ARM/2XLarge 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM/2XLarge 環境向けの ビルドの同時実行の最大 数
ARM/Large 環境向けのビルドの同時実 行	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM/Large 環境向けのビ ルドの同時実行の最大数

名前	デフォルト	引き上げ可能	説明
ARM/Medium 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM/Medium 環境向けの ビルドの同時実行の最大 数
ARM/Small 環境向けのビルドの同時実 行	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM/Small 環境向けのビ ルドの同時実行の最大数
ARM/XLarge 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 1	<u>あ</u> り	ARM/XLarge 環境向けの ビルドの同時実行の最大 数
Linux GPU Large 環境向けのビルドの同 時実行	サポートされてい る各リージョン: 0	<u>あ</u> り	Linux GPU/Large 環境向 けのビルドの同時実行の 最大数
Linux GPU Small 環境向けのビルドの同 時実行	サポートされてい る各リージョン: 0	<u>あ</u> り	Linux GPU/Small 環境向 けのビルドの同時実行の 最大数
Linux Lambda/10GB 環境向けの同時実 行ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux Lambda/10GB 環境 向けの同時実行ビルドの 最大数
Linux Lambda/1GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux Lambda/1GB 環境 向けの同時実行ビルドの 最大数
Linux Lambda/2GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux Lambda/2GB 環境 向けの同時実行ビルドの 最大数

名前	デフォルト	引き上げ可能	説明
Linux Lambda/4GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux Lambda/4GB 環境 向けの同時実行ビルドの 最大数
Linux Lambda/8GB 環境向けの同時実行 ビルド数	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux Lambda/8GB 環境 向けの同時実行ビルドの 最大数
Linux/2XLarge 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 0	<u>あ</u> り	Linux/2XLarge 環境向け のビルドの同時実行の最 大数
Linux/Large 環境向けのビルドの同時実 行	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux/Large 環境向けのビ ルドの同時実行の最大数
Linux/Medium 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux/Medium 環境向けの ビルドの同時実行の最大 数
Linux/Small 環境向けのビルドの同時実 行	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux/Small 模環境向けの ビルドの同時実行の最大 数
Linux/XLarge 環境向けのビルドの同時 実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Linux/XLarge 環境向けの ビルドの同時実行の最大 数
Windows Server 2019/Large 環境向けの ビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2019/ Large 環境向けのビルド の同時実行の最大数

名前	デフォルト	引き上げ可能	説明
Windows Server 2019/Medium 環境向け のビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2019/ Medium 環境向けのビル ドの同時実行の最大数
Windows Server 2022/2XLarge 環境向 けのビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2022/2XLarge 環境で同時 に実行されるビルドの最 大数
Windows Server 2022/Large 環境向けの ビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2022/ Large 環境で同時に実行 されるビルドの最大数
Windows Server 2022/Medium 環境向け のビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2022/ Medium 環境で同時に実 行されるビルドの最大数
Windows Server 2022/XLarge 環境向け のビルドの同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows Server 2022/ XLarge 環境で同時に実行 されるビルドの最大数
Windows/Large 環境向けのビルドの同 時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows/Large 環境向け のビルドの同時実行の最 大数
Windows/Medium 環境向けのビルドの 同時実行	サポートされてい る各リージョン: 1	<u>あ</u> り	Windows/Medium 環境向 けのビルドの同時実行の 最大数

名前	デフォルト	引き上げ可能	説明
ビルドタイムアウトの最小期間(分単 位)	サポートされてい る各リージョン: 5	い い え	ビルドの最大タイムアウ ト (分)
VPC 設定のセキュリティグループ	サポートされてい る各リージョン: 5	い い え	VPC 設定で利用可能なセ キュリティグループ
VPC 設定のサブネット	サポートされてい る各リージョン: 16	い い え	VPC 設定で利用可能なサ ブネット

Note

内部メトリクスは、同時実行ビルドのデフォルトクォータを決定します。

同時実行ビルドの最大数のクォータは、コンピューティングタイプによって異なります。一部のプ ラットフォームとコンピューティングタイプでは、デフォルトは 20 です。同時ビルドのクォータ の引き上げをリクエストする場合や、「アカウントのアクティブなビルドは X 以上持つことはでき ません」というエラーが発生した場合は、上記リンクでご依頼ください。料金の詳細については、 「AWS CodeBuild の料金」を参照してください。

その他の制限

ビルドプロジェクト

リソース	デフォルト値
ビルドプロジェクトの説明に使用できる文字	すべて
ビルドプロジェクト名に使用できる文字	文字 A-Z および a-z、数字 0-9、特殊文字 - および _
ビルドプロジェクト名の長さ	2~150 文字以内
ビルドプロジェクトの説明の最大長	255 文字
プロジェクトに追加できるレポートの最大数	5
すべての関連ビルドのビルドタイムアウトのた めにビルドプロジェクトで指定できる時間 (分)	5~2,160 (36 時間)

構築数

リソース	デフォルト値
ビルドの履歴が保持される最大時間	1 年
1 つのビルドのビルドタイムアウトのために指 定できる時間 (分)	5~2,160 (36 時間)

コンピューティングフリート

リソース	デフォルト値
コンピューティングフリートの同時実行数	10

AWS	CodeBuild	

リソース	デフォルト値
ARM/Small 環境向けのインスタンスの同時実 行数	1
ARM/Large 環境向けのインスタンスの同時実 行数	1
Linux/Small 環境向けのインスタンスの同時実 行数	1
Linux/Medium 環境向けのインスタンスの同時 実行数	1
Linux/Large 環境向けのインスタンスの同時実 行数	1
Linux/XLarge 環境向けのインスタンスの同時実 行数	1
Linux/2XLarge 環境向けのインスタンスの同時 実行数	0
Linux GPU/Small 環境向けのインスタンスの同 時実行数	0
Linux GPU/Large 環境向けのインスタンスの同 時実行数	0
Windows Server 2019/Medium 環境フリート向 けのインスタンスの同時実行数	1
Windows Server 2019/Large 環境フリート向け のインスタンスの同時実行数	1
Windows Server 2022/Medium 環境フリート向 けのインスタンスの同時実行数	1
Windows Server 2022/Large 環境フリート向け のインスタンスの同時実行数	1

リソース	デフォルト値
Mac ARM/Medium 環境フリート向けのインス タンスの同時実行数	1
Mac ARM/Large 環境フリート向けのインスタ ンスの同時実行数	1

レポート

リソース	デフォルト値
テストレポートの作成後の最大利用期間	30 日間
テストケースメッセージの最大長	5,000 文字
テストケース名の最大長	1,000 文字
AWS アカウントあたりのレポートグループの 最大数	5,000
レポートあたりのテストケースの最大数	500

[タグ]

タグの制限は、CodeBuild ビルドプロジェクトと CodeBuild レポートグループリソースのタグに適用されます。

リソース	デフォルト値
リソースタグのキー名	任意の組み合わせで使用できる文字は、 Unicode 文字、数字、スペース、および許可 されている UTF-8 文字 (1 ~ 127 文字) です。 使用できる文字は次のとおりです: + - = . _ : / @

リソース	デフォルト値
	タグキー名は一意である必要があり、各キーに 使用できる値は 1 つのみです。タグキー名に以 下のことはできません。
	• aws:から始まる
	• 空白文字のみで構成されている
	• 末尾にスペースを使用する
	・ 絵文字、または以下の文字を含める∶? ^ * [\ ~ ! # \$ % & *()> < " ' ` []{};
リソースタグの値	任意の組み合わせで使用できる文字は、 Unicode 文字、数字、スペース、および許可 されている UTF-8 文字 (0 ~ 255 文字) です。 使用できる文字は次のとおりです: + - = . _ : / @
	キーに使用できる値は 1 つのみですが、多数の キーと同じ値を含めることができます。タグの キー値に絵文字や次の文字を含めることはでき ません。 ? ^ * [\ ~ ! # \$ % & * () > < " ' ` [] { } ;

AWS CodeBuild ユーザーガイドのドキュメント履歴

次の表は、の前回のリリース以降のドキュメントの重要な変更点を示しています AWS CodeBuild。 このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

最新の API バージョン: 2016 年 10 月 6 日

変更	説明	日付
<u>新しいコンピューティ</u> <u>ングタイプ: CUSTOM_IN</u> <u>STANCE_TYPE</u>	CodeBuild では、 を使用し て、特定のインスタンスタ イプでリザーブドキャパシ ティフリートを作成できるよ うになりましたCUSTOM_IN STANCE_TYPE 。	2025 年 4 月 23 日
<u>CodeBuild サンドボックスの</u> <u>新しいサポート</u>	新しい CodeBuild サンドボッ クスの使用に関する情報を追 加しました。 <u>CodeBuild サン ドボックスでビルド</u> をデバッ グする」を参照してくださ い。	2025 年 4 月 7 日
<u>新しい Windows 環境タイプ</u>	CodeBuild で Windows XL お よび 2XL 環境タイプがサポー トされるようになりました。 詳細については、「 <u>ビルド環</u> <u>境のコンピューティングタイ</u> <u>プ</u> 」を参照してください。	2025 年 3 月 31 日
<u>Amazon S3 キャッシュの更新</u>	CodeBuild は、Amazon S3 キャッシュの新しいキャッ シュ動作をサポートするよう になりました。	2025 年 3 月 28 日

AWS CodeBuild

<u>新しいコンテンツ: GitHub</u> <u>Actions ランナー設定オプショ</u> ン	CodeBuild では、エンター プライズレベルでの登録 CODEBUILD_CONFIG_G ITHUB_ACTIONS_ENTE RPRISE_REGISTRATIO N_NAME がサポートされるよ うになりました。	2025 年 3 月 11 日
<u>新しいコンテンツ: 新しいウェ</u> <u>ブフックフィルタタイプを追</u> <u>加</u>	新しいウェブフックフィ ルタタイプ (ORGANIZAT ION_NAME)のサポートを追 加します。	2025 年 3 月 11 日
<u>新しいコンテンツ: S3 証</u> <u>明書ストレージを使用した</u> Fastlane による Apple コード 署名のチュートリアル	証明書ストレージに S3 を使 用して CodeBuild の Fastlane で Apple コード署名用の新し いチュートリアルを追加する	2025 年 2 月 5 日
<u>新しいコンテンツ: GitHub 証</u> <u>明書ストレージを使用した</u> Fastlane による Apple コード 署名のチュートリアル	証明書ストレージに GitHub を使用して CodeBuild の Fastlane で Apple コード署名 用の新しいチュートリアルを 追加する	2025 年 2 月 5 日
<u>新しいコンテンツ: Buildkite ラ</u> <u>ンナー</u>	Buildkite ランナーに新しいコ ンテンツを追加する	2025 年 1 月 31 日
<u>新しいコンテンツ: Buildkite 手</u> <u>動ウェブフック</u>	Buildkite 手動ウェブフックの サポートを追加します。	2025 年 1 月 31 日
<u>新しいコンテンツ: バッチビル</u> <u>ド buildspec リファレンス</u>	リザーブドキャパシティフ リートと Lambda 環境での バッチビルドのサポートを追 加します。	2025 年 1 月 8 日
<u>新しいコンテンツ: バッチビル</u> <u>ドで並列テストを実行する</u>	バッチビルドで並列テスト用 の新しいコンテンツを追加し ます。	2025 年 1 月 2 日

<u>新しいコンテンツ: ビルドを自</u> 動的に再試行	CodeBuild は、ウェブフック ビルドの自動再試行をサポー トするようになりました。	2024 年 12 月 18 日
<u>新しいコンテンツ: セルフホス ト型ランナーのプライベート</u> レジストリ認証情報を設定す <u>る</u>	非プライベートレジストリの カスタムイメージを使用す る場合のレジストリ認証情報 の設定のサポートを追加しま す。	2024 年 12 月 13 日
<u>新しいコンテンツ: GitHub</u> <u>Actions ランナー設定オプショ</u> <u>ン</u>	CodeBuild GitHub Actions セ ルフホスト型ランナーで、ラ ンナーを組織レベルで登録し 、特定のランナーグループ ID を設定できるようになりまし た。	2024 年 12 月 12 日
<u>新しいコンテンツ: 障害発生時</u> の属性を追加する RETRY	CodeBuild では、buildspec RETRYで障害発生時の属性を に設定できるようになりまし た。	2024 年 12 月 12 日
<u>新しいコンテンツ: GitLab 手</u> <u>動ウェブフック</u>	GitLab 手動ウェブフックのサ ポートを追加します。	2024 年 12 月 11 日
<u>更新された内容: 更新されたエ</u> <u>イリアス</u>	Linux ベースの標準ランタイム イメージのエイリアスを更新 します。	2024 年 11 月 22 日
更新された内容: CodeBuild が ホストする GitLab ランナーで サポートされているラベルオ ーバーライド	GitLab ランナーのカスタムイ メージラベルオーバーライド のサポートを追加します。	2024 年 11 月 22 日
更新された内容: CodeBuild が ホストする GitHub Actions ラ ンナーでサポートされている ラベルオーバーライド	GitHub Actions ランナーのカ スタムイメージラベルオー バーライドのサポートを追加 します。	2024 年 11 月 22 日

<u>のコンテンツ: AWS 管理 (事前</u> 定義) ポリシーを更新しました AWS CodeBuild	AWSCodeBuildAdminA ccess、AWSCodeBuild DeveloperAccess、およ びAWSCodeBuildReadOn lyAccessポリシーが更新 されました。元のリソー スarn:aws:codebuild: * がに更新されまし たarn:aws:codebuild: *:*:project/* 。	2024 年 11 月 15 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	リザーブドキャパシティフ リートは、ARM EC2、Linux EC2、Windows EC2 などのコ ンテナ以外のビルドをサポー トするようになりました。	2024 年 11 月 12 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	リザーブドキャパシティフ リートが属性ベースのコン ピューティングをサポートす るようになりました。	2024 年 11 月 6 日
<u>新しいコンテンツ: ビルドを自</u> 動的に再試行	CodeBuild では、ビルドの自 動再試行を有効にできるよう になりました。	2024 年 10 月 25 日
<u>新しいコンテンツ: リザーブ ドキャパシティフリートのマ ネージドプロキシサーバーで CodeBuild を実行</u>	リザーブドキャパシティフ リートのプロキシ設定サポー トを追加します。	2024 年 10 月 15 日
<u>新しいコンテンツ: セルフマ</u> <u>ネージド型 GitLab ランナー</u>	セルフマネージド型 GitLab ラ ンナーの新しいコンテンツを 追加します。	2024 年 9 月 17 日
<u>新しいコンテンツ: GitLab グ</u> <u>ループウェブフック</u>	GitLab グループウェブフック のサポートを追加します。	2024 年 9 月 17 日

<u>新しいコンテンツ:</u> INSTALL、PRE_BUILD、 POST_BUILD のフェーズで buildspec コマンドを実行	-with-buildspec のサ ポートを追加します。	2024 年 8 月 20 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	リザーブドキャパシティフ リートが macOS をサポート するようになりました。	2024 年 8 月 19 日
<u>新しいコンテンツ: GitHub ア</u> <u>プリ接続</u>	GitHub アプリ接続のサポート を追加します。	2024 年 8 月 14 日
<u>新しいコンテンツ: Bitbucket</u> <u>アプリ接続</u>	Bitbucket アプリ接続のサポー トを追加します。	2024 年 8 月 14 日
<u>新しいコンテンツ: CodeBuild</u> の複数のアクセストークン	のシークレットから、AWS Secrets Manager または AWS CodeConnections 接続を介し てアクセストークンをサード パーティープロバイダーに調 達するためのサポートを追加 します。	2024 年 8 月 14 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	リザーブドキャパシティフ リートは、ARM Medium、AR M XLarge 、および ARM 2XLarge コンピューティング タイプをサポートするように なりました。	2024 年 8 月 5 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	CodeBuild は、Windows のリ ザーブドキャパシティフリー トの VPC 接続をサポートする ようになりました。	2024 年 8 月 1 日

<u>新しい ARM コンピューティ</u> <u>ングタイプ</u>	CodeBuild が ARM Medium、ARM XLarge、およ び ARM 2XLarge コンピュー ティングタイプをサポートす るようになりました。詳細に ついては、「 <u>ビルド環境のコ</u> <u>ンピューティングタイプ</u> 」を 参照してください。	2024 年 7 月 10 日
<u>更新されたコンテンツ: SHA</u> <u>署名</u>	x86_64 と ARM 用の Secure Hash Algorithm (SHA) 署名を 更新します。	2024 年 6 月 19 日
<u>新しいコンテンツ: GitHub グ</u> <u>ローバルおよび組織のウェブ</u> <u>フック</u>	GitHub グローバルおよび組織 のウェブフックのサポートを 追加します。	2024 年 6 月 17 日
<u>新しいコンテンツ: 新しいウェ</u> <u>ブフックフィルタタイプを追</u> <u>加</u>	新しいウェブフックフィルタ タイプ (REPOSITORY_NAME) のサポートを追加します。	2024 年 6 月 17 日
<u>更新されたディスク容量</u>	ARM Small および ARM Largeコンピューティングタ イプでは、ディスク容量が増 えました。	2024 年 6 月 4 日
<u>新しいコンテンツ: GitHub 手</u> <u>動ウェブフック</u>	GitHub 手動ウェブフックのサ ポートを追加します。	2024 年 5 月 23 日
<u>更新されたコンテンツ: リザー</u> <u>ブドキャパシティ</u>	CodeBuild は、Amazon Linux のリザーブドキャパシティフ リートの VPC 接続をサポート するようになりました。	2024 年 5 月 15 日

<u>更新された内容: Lambda コン</u> <u>ピューティングイメージ</u>	.NET 8 (al-lambda/ aarch64/dotnet8 およ び al-lambda/x86_64/d otnet8)の Lambda サポー トを追加します。	2024 年 5 月 8 日
<u>更新されたクォータ: ビルドタ</u> <u>イムアウト</u>	最大ビルドタイムアウト クォータを 2,160 分 (36 時間) に更新します。	2024 年 5 月 1 日
<u>のコンテンツ: AWS 管理 (事前</u> 定義) ポリシーを更新しました AWS CodeBuild	AWSCodeBuildAdminA ccess、AWSCodeBuild DeveloperAccess、およ び AWSCodeBuildReadOn lyAccess ポリシーが更新され 、ブランド AWS CodeConne ctions 変更が反映されまし た。	2024 年 4 月 30 日
新しいコンテンツ: Bitbucket アプリのパスワードまたはア クセストークン	Bitbucket アクセストークンの サポートを追加します。	2024 年 4 月 11 日
<u>新しいコンテンツ: CodeBuild</u> <u>でレポートを自動的に検出</u>	CodeBuild はレポートの自動 検出をサポートするようにな りました。	2024 年 4 月 4 日
<u>新しいコンテンツ: セルフホス</u> ト型 GitHub Actions ランナー	セルフホスト型 GitHub Actions ランナーの新しいコン テンツを追加します。	2024 年 4 月 2 日
<u>新しいコンテンツ: GitLab 接</u> 続	GitLab および GitHub セルフ マネージド接続のサポートを 追加します。	2024 年 3 月 25 日

<u>新しいコンテンツ: 新しいウェ</u> <u>ブフックイベントとフィルタ</u> タイプを追加	新しいウェブフックイベント (RELEASED と PRERELEAS ED)とフィルタタイプ (TAG_NAME と RELEASE_N AME)のサポートを追加しま す。	2024 年 3 月 15 日
<u>新しいコンテンツ: 新しいウェ</u> <u>ブフックイベント PULL_REQU</u> <u>EST_CLOSED を追加</u>	新しいウェブフックイベント PULL_REQUEST_CLOSED のサポートを追加します。	2024 年 2 月 20 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Windows Server Core 2019 (windows-base:2019- 3.0)のサポートを追加しま す。	2024 年 2 月 7 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Amazon Linux 2023 (a12/ aarch64/standard/3.0) の新しいランタイムのサポー トを追加します。	2024 年 1 月 29 日
<u>新しいコンテンツ: リザーブド</u> <u>キャパシティ</u>	CodeBuild が CodeBuild のリ ザーブドキャパシティフリー トをサポートするようになり ました。	2024 年 1 月 18 日
<u>新しいコンピューティングタ</u> <u>イプ</u>	CodeBuild が Linux XLarge コ ンピューティングタイプをサ ポートするようになりました 。詳細については、「 <u>ビルド</u> 環境のコンピューティングタ <u>イプ</u> 」を参照してください。	2024 年 1 月 8 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Amazon Linux 2 (a12/ standard/5.0)と Ubuntu (ubuntu/standard/7.0) の新しいランタイムのサポー トを追加しました	2023 年 12 月 14 日

<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	新しい Lambda コンピュー ティングイメージのサポート を追加しました	2023 年 12 月 8 日
<u>新しいコンテンツ: AWS</u> Lambda コンピューティング	AWS Lambda コンピューティ ングに新しいコンテンツを追 加する	2023 年 11 月 6 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> ジ	Amazon Linux 2 (a12/stand ard/5.0)のサポートを追加	2023 年 5 月 17 日
<u>CodeBuild のマネージドポリ</u> <u>シーの変更</u>	CodeBuild の AWS マネージ ドポリシーの更新に関する詳 細が利用可能になりました。 詳細については、 <u>CodeBuild</u> <u>updates to AWS managed</u> <u>policies</u> 」を参照してくださ い。	2023 年 5 月 16 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Amazon Linux 2 (al2/stand ard/3.0)のサポートを削 除し、Amazon Linux 2 (al2/ standard/corretto8)と Amazon Linux 2 (al2/stand ard/corretto11)のサ ポートを追加	2023 年 5 月 9 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> ジ	Ubuntu 22.04 のサポートを追 加 (ubuntu/standard/7. 0)	2023 年 4 月 13 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Ubuntu 18.04 (ubuntu/st andard/4.0)と Amazon Linux 2 (al2/aarch64/ standard/1.0)のサポート を削除	2023 年 3 月 31 日

<u>更新されたコンテンツ: VPC</u> <u>制限を削除</u>	以下の制限を削除しま す。VPC で動作するように CodeBuild を設定した場合、 ローカルキャッシュはサポー トされません。2022 年 2 月 28 日以降、構築ごとに新しい Amazon EC2 インスタンスが 使用されるため、VPC の構築 に時間がかかります。	2023 年 3 月 1 日
<u>更新された内容: CodeBuild に</u> <u>用意されている Docker イメー</u> <u>ジ</u>	Ubuntu 18.04 (ubuntu/ standard/3.0)と Amazon Linux 2 (al2/stand ard/2.0)のサポートを削除	2022 年 6 月 30 日
<u>Amazon ECR サンプル: イ</u> <u>メージアクセスの制限</u>	CodeBuild 認証情報を使用 して Amazon ECR イメー ジをプルする場合、特定の CodeBuild プロジェクトへの イメージアクセスを制限で きます。詳細については、 「 <u>Amazon ECR のサンプル</u> 」 を参照してください。	2022 年 3 月 10 日
<u>リージョンサポートの追加</u>	ARM_CONTAINER コンピュー ティングタイプが、アジアパ シフィック (ソウル)、カナダ (中部)、欧州 (ロンドン)、欧 州 (パリ) の各リージョンで サポートされるようになりま した。詳細については、「 <u>ビ</u> <u>ルド環境のコンピューティン</u> <u>グタイプ</u> 」を参照してくださ い。	2022 年 3 月 10 日

<u>新しい VPC 制限</u>	VPC で動作するように CodeBuild を設定した場合、 ローカルキャッシュはサポー トされません。2022 年 2 月 28 日以降、構築ごとに新しい Amazon EC2 インスタンスが 使用されるため、VPC の構築 に時間がかかります。	2022 年 2 月 25 日
<u>バッチレポートモード</u>	CodeBuild では、プロジェク トのソースプロバイダーに バッチビルドステータスを送 信する方法を選択できるよ うになりました。詳細につい ては、「 <u>バッチレポートモー</u> <u>ド</u> 」を参照してください。	2021 年 10 月 4 日
<u>新しいコンピューティングタ</u> <u>イプ</u>	CodeBuild が小さな ARM コ ンピューティングタイプをサ ポートするようになりました 。詳細については、「 <u>ビルド</u> 環境のコンピューティングタ <u>イプ</u> 」を参照してください。	2021 年 9 月 13 日
<u>パブリックビルドプロジェク</u> ト	CodeBuild では、AWS アカウ ントへのアクセスを必要とせ ずに、ビルドプロジェクトの ビルド結果を一般公開できる ようになりました。詳細につ いては、 <u>パブリックビルドプ</u> <u>ロジェクト</u> を参照してくださ い。	2021 年 8 月 11 日

<u>バッチビルドのセッションデ</u> <u>バッグ</u>	CodeBuild は、バッチビルド のセッションデバッグをサ ポートするようになりまし た。詳細については、「 <u>build-</u> graph」および「 <u>build-list</u> 」を 参照してください。	2021 年 3 月 3 日
<u>プロジェクトレベルの同時ビ</u> <u>ルド制限</u>	CodeBuild では、ビルドプロ ジェクトの同時ビルド数を制 限できるようになりました。 詳細については、「 <u>プロジェ</u> <u>クト設定</u> 」および「 <u>concurren</u> <u>tBuildLimit</u> 」を参照してくださ い。	2021 年 2 月 16 日
<u>新しい buildspec プロパティ:</u> <u>s3-prefix</u>	CodeBuild では、アーティ ファクト用の s3-prefix buildspec プロパティで、Am azon S3 にアップロードされ るアーティファクトのパスプ レフィックスを指定できるよ うになりました。詳細につい ては、「 <u>s3-prefix</u> 」を参照し てください。	2021 年 2 月 9 日
<u>新しい buildspec プロパティ:</u> <u>on-failure</u>	CodeBuild では、ビルド フェーズ用の on-failure buildspec プロパティを使用す ることで、ビルドフェーズの 失敗時の処理を指定できるよ うになりました。詳細につい ては、「 <u>on-failure</u> 」を参照し てください。	2021 年 2 月 9 日

<u>新しい buildspec プロパティ:</u> <u>exclude-paths</u>	CodeBuild では、ビルドアー ティファクトからパスを除 外できる、アーティファク ト用の exclude-paths buildspec プロパティが使用 できるようになりました。詳 細については、「 <u>exclude-p</u> <u>aths</u> 」を参照してください。	2021 年 2 月 9 日
<u>新しい buildspec プロパティ:</u> <u>enable-symlinks</u>	CodeBuild では、ZIP アーティ ファクト内のシンボリック リンクを保持できる、アー ティファクト用の enable- symlinks buildspec プロ パティが使用できるようにな りました。詳細については、 「 <u>enable-symlinks</u> 」を参照し てください。	2021 年 2 月 9 日
<u>Buildspec アーティファクト名</u> の強化	CodeBuild では、 「artifacts/name 」プ ロパティを使用して、パス情 報を格納できるようになりま した。詳細については、「 <u>名</u> <u>前</u> 」を参照してください。	2021 年 2 月 9 日
<u>コードのカバレッジレポート</u>	CodeBuild でコードカバレッ ジレポートが提供されるよう になりました。詳細について は、「 <u>コードカバレッジレ</u> <u>ポート</u> 」を参照してくださ い。	2020 年 7 月 30 日

<u>バッチビルド</u>	CodeBuild では、プロジェク トの同時および調整された ビルドの実行がサポートされ るようになりました。詳細に ついては、「 <u>CodeBuild での</u> <u>バッチビルド</u> 」を参照してく ださい。	2020 年 7 月 30 日
<u>Windows Server 2019 イメー</u> ジ	CodeBuild は、Windows Server Core 2019 ビルドイ メージを提供するようにな りました。詳細については、 「 <u>CodeBuild に用意されてい</u> <u>る Docker イメージ</u> 」を参照し てください。	2020 年 7 月 20 日
<u>セッションマネージャー</u>	CodeBuild では、実行中の ビルドを一時停止し、AWS Systems Manager Session Manager を使用してビルド コンテナに接続し、コンテナ の状態を表示できるようにな りました。詳細については、 「 <u>セッションマネージャー</u> 」 を参照してください。	2020 年 7 月 20 日
<u>トピックの更新</u>	CodeBuild では、buildspec ファイル内のビルド環境で使 用するシェルの指定をサポー トするようになりました。詳 細については、「 <u>ビルド仕様</u> に関するリファレンス」を参 照してください。	2020 年 6 月 25 日

<u>テストフレームワークを使用</u> <u>したテストレポート</u>	いくつかのテストフレーム ワークで CodeBuild テスト レポートを生成する方法を説 明するいくつかのトピックを 追加しました。詳細について は、「 <u>テストフレームワーク</u> <u>を使用したテストレポート</u> 」 を参照してください。	2020 年 5 月 29 日
<u>トピックの更新</u>	CodeBuild は、レポートグ ループへのタグの追加を サポートするようになり ました。詳細については、 「 <u>ReportGroup</u> 」を参照して ください。	2020 年 5 月 21 日
<u>テストレポートのサポート</u>	CodeBuild テストレポートの サポートの一般提供が開始さ れました。	2020 年 5 月 21 日
<u>トピックの更新</u>	CodeBuild は、HEAD コミッ トメッセージが指定された式 に一致した場合にのみビルド をトリガーする Github および Bitbucket の create Webhook フィルタの作成をサポート するようになりました。詳 細については、「 <u>GitHub プ</u> <u>ルリクエストと Webhook</u> <u>フィルタのサンプル</u> 」および 「 <u>Bitbucket プルリクエストと</u> Webhook フィルタのサンプル 」を参照してください。	2020年5月6日

<u>新しいトピック</u>	CodeBuild は現在、共有ビル ドプロジェクトおよびレポー トグループリソースをサポー トしています。詳細について は、「 <u>共有プロジェクトの使</u> <u>用</u> 」と「 <u>共有レポートグルー</u> <u>プの使用</u> 」を参照してくださ い。	2019 年 12 月 13 日
<u>新しく更新されたトピック</u>	CodeBuild では現在、ビルド プロジェクト実行中にテスト レポートがサポートされるよ うになりました。詳細につい ては、「テストレポートの使 用」、「テストレポートの作 成」、AWS CLI「サンプルを 使用したテストレポートの作 成」を参照してください。	2019 年 11 月 25 日
<u>トピックの更新</u>	CodeBuild は、現在 Linux GPU と Arm 環境タイプ、 および 2x1arge コンピュー ティングタイプをサポートし ています。詳細については、 「 <u>ビルド環境のコンピュー</u> <u>ティングタイプ</u> 」を参照して ください。	2019 年 11 月 19 日

<u>トピックの更新</u>	CodeBuild は、すべてのビル ドのビルド番号、環境変数の エクスポート、AWS Secrets Manager 統合をサポートす るようになりました。詳細 については、「 <u>buildspec の</u> 構文」の「 <u>エクスポートさ</u> <u>れた変数</u> 」および「 <u>Secrets</u> <u>Manager</u> 」を参照してくださ い。	2019 年 11 月 6 日
<u>新しいトピック</u>	CodeBuild が通知ルールをサ ポートするようになりまし た。通知ルールを使用して、 ビルドプロジェクトの重要な 変更をユーザーに通知できま す。詳細については、「 <u>通知</u> <u>ルールを作成する</u> 」を参照し てください。	2019 年 11 月 5 日
<u>トピックの更新</u>	CodeBuild は、Android バー ジョン 29 と Go バージョン 1.13 のランタイムをサポート するようになりました。詳細 については、「 <u>CodeBuild に</u> <u>用意されている Docker イメー ジ</u> 」および「 <u>buildspec の構</u> <u>文</u> 」を参照してください。	2019 年 9 月 10 日

<u>トピックの更新</u>	プロジェクトを作成するとき に、Amazon Linux 2 (AL2) マ ネージドイメージを選択でき るようになりました。詳細に ついては、「 <u>CodeBuild に用</u> 意されている Docker イメー ジ」および「 <u>CodeBuild 用</u> buildspec ファイルサンプルの ランタイムバージョン」を参 照してください。	2019 年 8 月 16 日
<u>トピックの更新</u>	プロジェクトを作成するとき に、S3 ログの暗号化を無効に (Git ベースのソースリポジト リを使用する場合は Git サブ モジュールを含めるように) 選 択できるようになりました。 詳細については、「 <u>CodeBuild</u> <u>でのビルドプロジェクトの作</u> <u>成</u> 」を参照してください。	2019 年 3 月 8 日
<u>新しいトピック</u>	CodeBuild でローカルキャッ シュがサポートされるよう になりました。ビルドの作 成時、4 つのモードのうち 1 つ以上のモードでローカル キャッシュを指定できます。 詳細については、「 <u>CodeBuild</u> <u>でキャッシングをビルドす</u> <u>る</u> 」を参照してください。	2019 年 2 月 21 日

<u>新しいトピック</u>	CodeBuild では、ビルドをト リガーするイベントを指定 するためのウェブフックイ ベントフィルタグループが サポートされるようになり ました。詳細については、 「 <u>GitHub ウェブフックイベン</u> トのフィルタリング」および 「 <u>Bitbucket ウェブフックイベ</u> ントのフィルタリング」を参 照してください。	2019年2月8日
<u>新しいトピック</u>	CodeBuild ユーザーガイド に、プロキシサーバーでの CodeBuild の使用方法が追加 されました。詳細について は、「 <u>プロキシサーバーで</u> <u>CodeBuild を使用する</u> 」を参 照してください。	2019 年 2 月 4 日
<u>トピックの更新</u>	CodeBuild は、別の AWS アカウントにある Amazon ECR イメージの使用を サポートするようになり ました。この変更を反映 するために、「CodeBuil d の Amazon ECR サンプ ル」、「ビルドプロジェクト の作成」、「CodeBuild サー ビスロールの作成」などのト ピックを更新しています。	2019 年 1 月 24 日

<u>プライベート Docker レジスト</u> <u>リのサポート</u>	CodeBuild では、プライベー トレジストリに保存されてい る Docker イメージをランタ イム環境として使用できるよ うになりました。詳細につい ては、 <u>AWS Secrets Manager</u> 「サンプルを含むプライベー トレジストリ」を参照してく ださい。	2019 年 1 月 24 日
<u>トピックの更新</u>	CodeBuild では、アクセス トークンを使用した GitHub (個人用アクセストークンを使 用) および Bitbucket (アプリパ スワードを使用) リポジトリへ の接続がサポートされるよう になりました。詳細について は、「ビルドプロジェクトの 作成 (コンソール)」と「ソー スプロバイダにアクセストー クンを使用する」を参照して ください。	2018年12月6日
<u>トピックの更新</u>	CodeBuild で、ビルドの各 フェーズの所要時間を測定す る新しいビルドメトリクスが サポートされるようになり ました。詳細については、 「 <u>CodeBuild CloudWatch のメ</u> <u>トリクス</u> 」を参照してくださ い。	2018 年 11 月 15 日

<u>VPC エンドポイントポリシー</u> <u>のトピック</u>	CodeBuild の Amazon VPC エンドポイントでポリシー がサポートされるようにな りました。詳細については、 「 <u>CodeBuild の VPC エンドポ</u> <u>イントポリシーの作成</u> 」を参 照してください。	2018 年 11 月 9 日
更新された内容	新しいコンソールデザインに 関するトピックを更新しまし た。	2018 年 10 月 30 日
<u>Amazon EFS のサンプル</u>	CodeBuild は、プロジェクト の buildspec ファイルでコマ ンドを使用して、ビルド中に Amazon EFS ファイルシステ ムをマウントできます。詳細 については、「 <u>CodeBuild の</u> <u>Amazon EFS のサンプル</u> 」を 参照してください。	2018 年 10 月 26 日
<u>Bitbucket ウェブフック</u>	CodeBuild では、リポジトリ の Bitbucket を使用する際、 ウェブフックをサポートす るようになりました。詳細 については、「 <u>CodeBuild の</u> <u>Bitbucket プルリクエストサン</u> <u>プル</u> 」を参照してください。	2018 年 10 月 2 日
<u>S3 ログ</u>	CodeBuild で、 S3 バケット のログ作成がサポートされ るようになりました。以前 は、CloudWatch Logs を使用 してログを構築できませんで した。詳細については、「 <u>プ</u> <u>ロジェクトを作成する</u> 」を参 照してください。	2018 年 9 月 17 日

<u>複数の入力ソースと複数の出</u> <u>力アーティファクト</u>	CodeBuild は、複数の入力 ソースを使用するプロジェク トをサポートし、複数のアー ティファクトのセットを公 開します。詳細については、 「 <u>複数の入力ソースと出力</u> <u>アーティファクトのサンプ</u> ル」および「 <u>CodePipeline を</u> <u>CodeBuild の複数の入力ソー</u> <u>スおよび出力アーティファク</u> トと統合するサンプル」を参 照してください。	2018年8月30日
<u>セマンティックバージョニン</u> <u>グのサンプル</u>	CodeBuild ユーザーガイドに は、セマンティックバージョ ニングを使用してビルド時に アーティファクト名を作成す る方法を示す、ユースケース ベースのサンプルが用意され ています。詳細については、 「 <u>セマンティックバージョ</u> ニングを使用してビルドアー <u>ティファクトのサンプルに名</u> 前を付ける」を参照してくだ さい。	2018年8月14日

<u>新しい静的ウェブサイトのサ</u> <u>ンプル</u>	CodeBuild ユーザーガイドに は、ビルド出力を S3 バケッ トでホストする方法を示す ユースケースベースのサン プルがあります。このサンプ ルは、最近サポートされた暗 号化されていないビルドアー ティファクトを利用していま す。詳細については、「 <u>ビル</u> <u>ド出力を S3 バケットでホス</u> <u>トする静的ウェブサイトの作</u> <u>成</u> 」を参照してください。	2018年8月14日
<u>セマンティックバージョニン</u> <u>グによるアーティファクト名</u> <u>の上書きのサポート</u>	セマンティックバージョニン グを使用して、CodeBuild が ビルドアーティファクトに名 前を付けるために使用する形 式を指定できるようになり ました。これが役立つのは、 ハードされた名前を持 つビルドアーティファクトが上 に いて、前を使用する前のビル ドアーティファクトが上書 されるためです。たとえば、 ビルーされた場合、アーティ ファクト名にタイムスタンプ を らくしいドアーティファクト に なりにまっため、以前 のビルドのアーティファクト は上書きされません。	2018年8月7日

<u>暗号化されていないビルド</u> <u>アーティファクトのサポート</u>	CodeBuild では、暗号化され ていないビルドアーティファ クトを持つビルドがサポー トされるようになりました。 詳細については、「 <u>ビルドプ</u> <u>ロジェクトの作成 (コンソー</u> <u>ル)</u> 」を参照してください。	2018 年 7 月 26 日
<u>Amazon CloudWatch のメトリ</u> クスとアラームのサポート	CodeBuild では、CloudWatch のメトリクスとアラームとの 統合が提供されるようにな りました。CodeBuild または CloudWatch コンソールを使 用して、プロジェクトレベル とアカウントレベルでビルド をモニタリングします。詳細 については、「ビルドのモニ タリング」を参照してくださ い。	2018年7月19日
<u>ビルドステータスの報告のサ</u> <u>ポート</u>	CodeBuild からソースプロバ イダーに対して、ビルドの開 始と完了のステータスが報告 されるようになりました。詳 細については、「 <u>CodeBuild</u> <u>でのビルドプロジェクトの作</u> <u>成</u> 」を参照してください。	2018 年 7 月 10 日
<u>CodeBuild ドキュメントへの</u> <u>環境変数の追加</u>	[<u>ビルド環境の環境変数</u>] ペー ジが更新され、CODEBUILD _BUILD_ID、CODEBUIL D_LOG_PATH、および CODEBUILD_START_TIME 環境変数が追加されました。	2018 年 7 月 9 日

<u>buildspec ファイルでの</u> <u>finally ブロックのサポート</u>	CodeBuild のドキュメントが 更新され、buildspec ファイ ルにオプションの finally ブロックに関する詳細が追加 されました。finally ブロック 内のコマンドは、常にその対 応する commands ブロック 内のコマンドの実行後に実行 されます。詳細については、 「 <u>buildspec の構文</u> 」を参照し てください。	2018年6月20日
<u>CodeBuild エージェントの更</u> <u>新に関する通知</u>	CodeBuild ドキュメントが更 新され、新しいバージョンの CodeBuild エージェントがリ リースされたときに Amazon SNS で通知を受け取る方法 に関する詳細が追加されまし た。詳細については、「新し い AWS CodeBuild エージェン トバージョンの通知を受信す <u>る</u> 」を参照してください。	2018年6月15日

以前の更新

次の表に、2018 年 6 月以前の「AWS CodeBuild ユーザーガイド」の各リリースにおける重要な変 更点を示します。

変更	説明	日付
Windows ビルドのサポート	CodeBuild で Microsoft Windows Server プラット フォームのビルドがサポー トされるようになりました。 これには、Windows の .NET Core 2.0 のパッケージ済みビ	2018 年 5 月 25 日
変更	説明	日付
-------------------------	---	-----------------
	ルド環境が含まれます。詳細 については、「 <u>CodeBuild の</u> <u>Microsoft Windows サンプルを</u> <u>実行</u> 」を参照してください。	
ビルドのべき等性のサポート	AWS Command Line Interface (AWS CLI) を使用して start-build コマンドを実 行すると、ビルドのべき等性 を確保できます。詳細につい ては、「 <u>ビルドの実行 (AWS</u> <u>CLI)</u> 」を参照してください。	2018 年 5 月 15 日
ビルドプロジェクト設定の上 書き数の増加	ビルドの作成時に上書きでき るビルドプロジェクト設定の 数が増えました。オーバーラ イドは当該ビルドに限られま す。詳細については、「 <u>AWS</u> <u>CodeBuild ビルドを手動で実</u> 行する」を参照してくださ い。	2018年5月15日
VPC エンドポイントのサポー ト	VPC エンドポイントを使用し てビルドのセキュリティを強 化できるようになりました。 詳細については、「 <u>VPC エン</u> <u>ドポイントの使用</u> 」を参照し てください。	2018年3月18日
トリガーのサポート	定期的な間隔でビルドをスケ ジュールするためのトリガー を作成できるようになりまし た。詳細については、「 <u>AWS</u> <u>CodeBuild トリガーの作成</u> 」 を参照してください。	2018年3月28日

変更	説明	日付
FIPS エンドポイントに関する ドキュメント	これで、AWS Command Line Interface (AWS CLI) または AWS SDK を使用してCode Build に 4 つの連邦情報処理 標準 (FIPS) エンドポイントの いずれかを使用するように指 示する方法について説明しま す。詳細については、「AWS <u>CodeBuild エンドポイントを</u> 指定する」を参照してくださ い。	2018年3月28日
AWS CodeBuild アジアパシ フィック (ムンバイ)、欧州 (パリ)、南米 (サンパウロ) で 利用可能に	AWS CodeBuild が、アジアパ シフィック (ムンバイ)、欧州 (パリ)、南米 (サンパウロ) の 各リージョンで利用可能にな りました。詳細については、 「Amazon Web Services 全 般のリファレンス」の「 <u>AWS</u> <u>CodeBuild</u> 」を参照してくださ い。	2018年3月28日
GitHub Enterprise Server のサ ポート	CodeBuild は、GitHub Enterprise Server リポジト リに保存されたソースコー ドからビルドできるようにな りました。詳細については、 「 <u>GitHub Enterprise Server サ</u> <u>ンプルを実行</u> 」を参照してく ださい。	2018 年 1 月 25 日

変更	説明	日付
Git クローンの深さサポート	CodeBuild は、指定されるコ ミット数で切り捨てられる 履歴の浅いクローンの作成を サポートするようになりまし た。詳細については、「 <u>ビル</u> <u>ドプロジェクトの作成</u> 」を参 照してください。	2018年1月25日
VPC サポート	VPC 対応のビルドが VPC 内 のリソースにアクセスできる ようになりました。詳細につ いては、「 <u>VPC サポート</u> 」を 参照してください。	2017 年 11 月 27 日
依存関係のキャッシュのサ ポート	CodeBuild で依存関係の キャッシュがサポートされる ようになりました。これによ り、CodeBuild は、ビルド環 境の特定の再利用可能部分を キャッシュに保存し、これを 複数のビルドにわたって使用 できます。	2017 年 11 月 27 日
ビルドバッジのサポート	CodeBuild でビルドバッジが 使用可能になりました。ビル ドバッジは、埋め込み可能な イメージ (バッジ) として動 的に生成され、プロジェクト の最新ビルドのステータスを 示します。詳細については、 「 <u>ビルドバッジサンプル</u> 」を 参照してください。	2017 年 11 月 27 日

AWS CodeBuild

変更	説明	日付
AWS Config 統合	AWS Config は AWS リソー スとして CodeBuild をサポー トするようになりました。つ まり、サービスは CodeBuild プロジェクトを追跡できま す。詳細については AWS Config、 <u>AWS Config サンプ</u> <u>ル</u> 「」を参照してください。	2017 年 10 月 20 日
GitHub リポジトリで更新され たソースコードの自動的な再 構築	ソースコードを GitHub リポ ジトリに保存している場合 は、コード変更がリポジトリ にプッシュされるたびに AWS CodeBuild でソースコードを 再構築できます。詳細につい ては、「 <u>GitHub プルリクエス</u> トとウェブフックフィルタの サンプルを実行」を参照して ください。	2017 年 9 月 21 日

変更	説明	日付
Amazon EC2 Systems Manager パラメータストアで の新しい方法による重要また は大規模な環境変数の保存と 取得	AWS CodeBuild コンソー ルまたは を使用して AWS CLI、Amazon EC2 Systems Manager パラメータストア に保存されている機密または 大規模な環境変数を取得でき るようになりました。 AWS CodeBuild コンソールを使用 して、これらのタイプの環境 変数を Amazon EC2 Systems Manager パラメータストア に保存できるようになりま した。これまでは、これらの 種類の環境変数を取得するに は、これらの変数をビルド仕 様に含めるか、ビルドコマン ドを実行して AWS CLIを自動 化する以外にありませんでし た。また、これらの種類の環 境変数を保存するには、Am azon EC2 Systems Manager パラメータストアコンソー ルを使用する以外にありませ んでした。詳細については、 「ビルドプロジェクトの作 成」、「ビルドプロジェクトの作 成」、「ビルドプロジェクトの作 成」、「ビルドプロジェクトの た	2017年9月14日
ビルドの削除のサポート	AWS CodeBuildでビルドを削 除できるようになりました。 詳細については、「 <u>ビルドの</u> <u>削除</u> 」を参照してください。	2017 年 8 月 31 日

変更	説明	日付
Amazon EC2 Systems Manager パラメータストアに 保存された重要または大規模 な環境変数をビルド仕様を使 用して取得する新しい方法	AWS CodeBuild では、builds pec を使用して、Amazon EC2 Systems Manager パラ メータストアに保存されてい る機密または大規模な環境変 数を簡単に取得できるように なりました。これまでは、こ れらの種類の環境変数を取得 するには、ビルドコマンドを 実行して AWS CLIを自動化す る以外にありませんでした。 詳細については、「 <u>buildspec</u> の構文」の parameter- store マッピングを参照して ください。	2017年8月10日
AWS CodeBuild が Bitbucket をサポート	CodeBuild は、Bitbucket リ ポジトリに保存されたソース コードから構築できるように なりました。詳細について は、「 <u>ビルドプロジェクト</u> <u>の作成</u> 」および「 <u>ビルドを手</u> <u>動で実行</u> 」を参照してくださ い。	2017 年 8 月 10 日
AWS CodeBuild 米国西部 (北 カリフォルニア)、欧州 (ロン ドン)、カナダ (中部) で利用 可能に	AWS CodeBuild が、米国西 部 (北カリフォルニア)、欧州 (ロンドン)、カナダ (中部) の 各リージョンで利用可能にな りました。詳細については、 「Amazon Web Services 全 般のリファレンス」の「 <u>AWS</u> <u>CodeBuild</u> 」を参照してくださ い。	2017年6月29日

変更	説明	日付
buildspec ファイルの代替の名 前および場所のサポート	ビルドプロジェクトで使用 する buildspec ファイル名と して、ソースコードのルー トにあるデフォルトの名前 (buildspec.yml)の代わり に、別の名前や場所を指定で きるようになりました。詳細 については、「 <u>buildspec ファ</u> <u>イル名とストレージの場所</u> 」 を参照してください。	2017年6月27日
更新されたビルド通知のサン プル	CodeBuild に、Amazon CloudWatch Events および Amazon Simple Notification Service (Amazon SNS) を介 したビルド通知の組み込みサ ポートが組み込まれました。 この新しい動作を反映するた めに、従来の <u>ビルド通知サン</u> <u>プル</u> が更新されています。	2017年6月22日
カスタムイメージの Docker のサンプルを追加	CodeBuild およびカスタム Docker ビルドイメージを使 用して Docker イメージをビ ルドして実行する方法を示 すサンプルを追加しました。 詳細については、「 <u>カスタム</u> <u>Docker イメージのサンプル</u> 」 を参照してください。	2017 年 6 月 7 日

変更	説明	日付
GitHub のプル要求に応じた ソースコードの取得	GitHub リポジトリに保存さ れたソースコードに依存する ビルドを CodeBuild で実行 するときに、ビルドに対する GitHub プル要求 ID を指定で きるようになりました。代わ りに、コミット ID、ブランチ 名、またはタグ名を指定する こともできます。詳細につい ては、「ビルドの実行 (コン ソール)」の [ソースバージョ ン] の値または「ビルドの実行 (AWS CLI)」の sourceVer sion の値を参照してくださ い。	2017年6月6日
ビルド仕様バージョンの更新	新しいバージョンのビル ド仕様形式がリリースされ ました。バージョン 0.2 で は、CodeBuild でデフォルト シェルのインスタンス別に 各ビルドコマンドを実行す る場合の課題に対処していま す。また、バージョン 0.2 で は environment_variab les の名前が env に変更 され、plaintext の名前 が variables 変更されて います。詳細については、「 <u>CodeBuild のビルド仕様に関</u> <u>するリファレンス</u> 」を参照し てください。	2017年5月9日

AWS CodeBuild

変更	説明	日付
GitHub で使用可能なビルドイ メージの Dockerfiles	が提供する多くのビルドイ メージの定義 AWS CodeBuild は、GitHub の Dockerfiles と して利用できます。詳細に ついては、「 <u>CodeBuild に用</u> <u>意されている Docker イメー</u> <u>ジ</u> 」にある表の「定義」列を 参照してください。	2017 年 5 月 2 日
AWS CodeBuild 欧州 (フラ ンクフルト)、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) で利用可能に	AWS CodeBuild が欧州 (フ ランクフルト)、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) の各リージョンで 利用可能になりました。詳細 については、「Amazon Web Services 全般のリファレンス 」の「 <u>AWS CodeBuild</u> 」を参 照してください。	2017年3月21日
CodeBuild の CodePipeline の テストアクションのサポート	CodePipeline のパイプライ ンに CodeBuild を使用するテ ストアクションを追加できる ようになりました。詳細につ いては、「 <u>CodeBuild テスト</u> アクションをパイプラインに 追加する (CodePipeline コン ソール)」を参照してくださ い。	2017年3月8日

変更	説明	日付
buildspec ファイルは、選択し た最上位ディレクトリからの ビルド出力の取得をサポート します。	buildspec ファイルでは、個々 の最上位ディレクトリを指定 して、その内容をビルド出力 アーティファクトに含めるよ うに CodeBuild に指示でき るようになりました。これは 、base-directory マッピ ングを使用して行います。詳 細については、「 <u>buildspec の</u> 構文」を参照してください。	2017年2月8日
組み込み環境変数	AWS CodeBuild には、使用 するビルド用の追加の組み込 み環境変数が用意されていま す。これらには、ビルドを開 始したエンティティを記述す る環境変数、ソースコードリ ポジトリへの URL、ソース コードのバージョン ID など が含まれます。詳細について は、「 <u>ビルド環境の環境変</u> 数」を参照してください。	2017年1月30日
AWS CodeBuild 米国東部 (オ ハイオ) で利用可能に	AWS CodeBuild が米国東部 (オハイオ) リージョンで利用 可能になりました。詳細に ついては、「Amazon Web Services 全般のリファレンス 」の「 <u>AWS CodeBuild</u> 」を参 照してください。	2017 年 1 月 19 日

変更	説明	日付
シェルおよびコマンドの動作 情報	CodeBuild は、ビルド環境の デフォルトのシェルの個別の インスタンスで指定した各コ マンドを実行します。このデ フォルトの動作によって、コ マンドに予期しない悪影響が 生じることがあります。必要 に応じて、このデフォルトの 動作を回避するいくつかの方 法をお勧めします。詳細につ いては、「ビルド環境のシェ ルとコマンド」を参照してく ださい。	2016年12月9日
環境変数の情報	CodeBuild には、ビルドコマ ンドで使用できるいくつか の環境変数が用意されていま す。独自の環境変数を定義す ることもできます。詳細につ いては、「 <u>ビルド環境の環境</u> 変数」を参照してください。	2016 年 7 月 12 日
トラブルシューティング情報	トラブルシューティング情報 が利用できるようになりま した。詳細については、「 <u>ト</u> <u>ラブルシューティング AWS</u> <u>CodeBuild</u> 」を参照してくださ い。	2016 年 5 月 12 日
Jenkins プラグインの初回リ リース	これは、CodeBuild Jenkins プラグインの初回リリー スです。詳細については、 「 <u>Jenkins AWS CodeBuild で</u> <u>を使用する</u> 」を参照してくだ さい。	2016 年 5 月 12 日

変更	説明	日付
ユーザーガイド初回リリース	これは、CodeBuild ユーザー ガイドの初回リリースです。	2016 年 12 月 1 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛 盾がある場合、英語版が優先します。