



# [AWS Black Belt Online Seminar]

## AWS CodeBuild

サービスカットシリーズ

Solutions Architect 松本 雅博  
2020/11/25

AWS 公式 Webinar  
<https://amzn.to/JPWebinar>



過去資料  
<https://amzn.to/JPArchive>



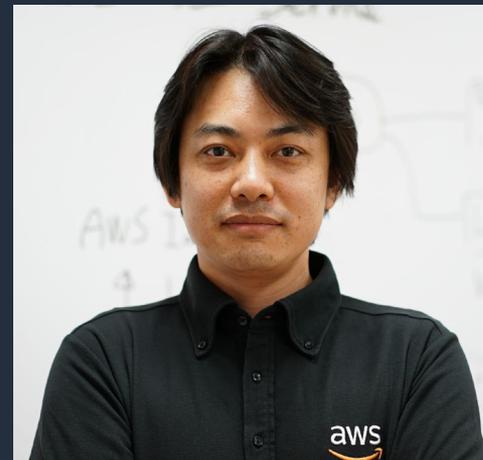
# 自己紹介

松本 雅博（まつもと まさひろ）

技術統括本部 西日本ソリューション部  
ソリューションアーキテクト

関西を中心に、西日本のお客様をご支援  
好きなサービス

- Code シリーズ
- AWS CloudFormation, AWS Cloud Development Kit



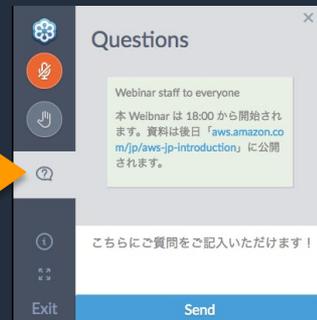
# AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾンウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

## 質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問はお答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください  
#awsblackbelt

# 内容についての注意点

- 本資料では2020年11月25日現在のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様には別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

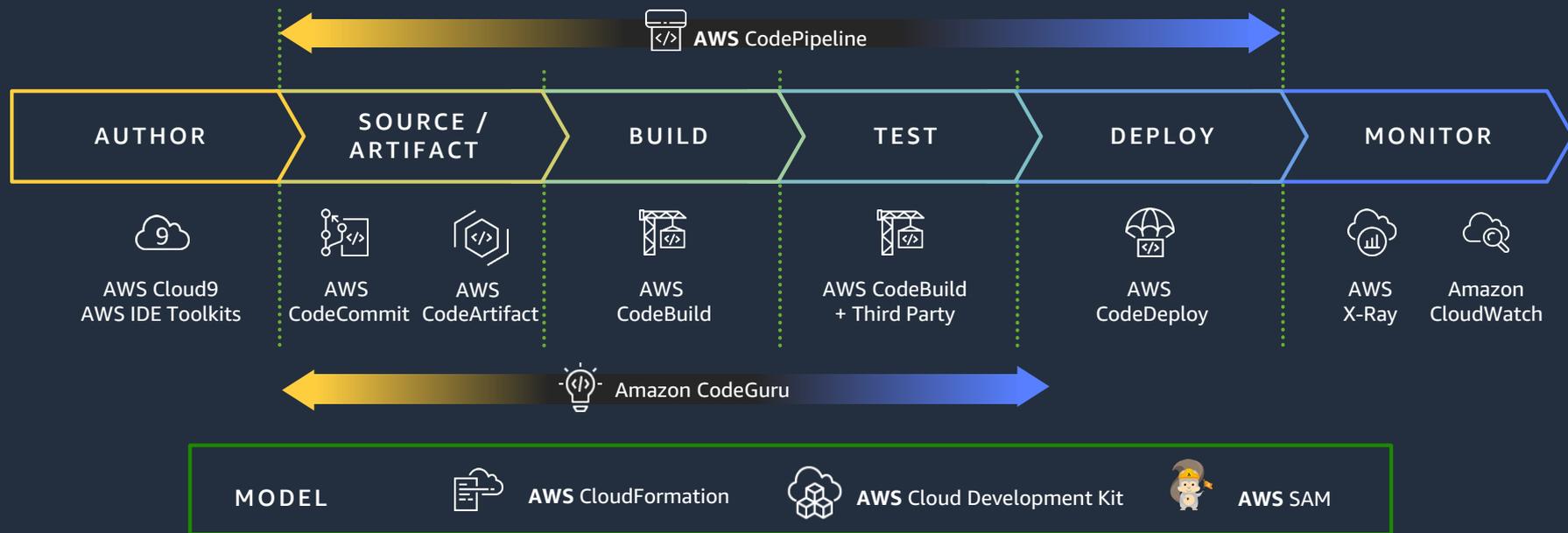
# 本セミナーの概要

- 本セミナーで学習できること
  - AWS CodeBuild
- 対象者
  - アーキテクトの方
  - 技術者の方

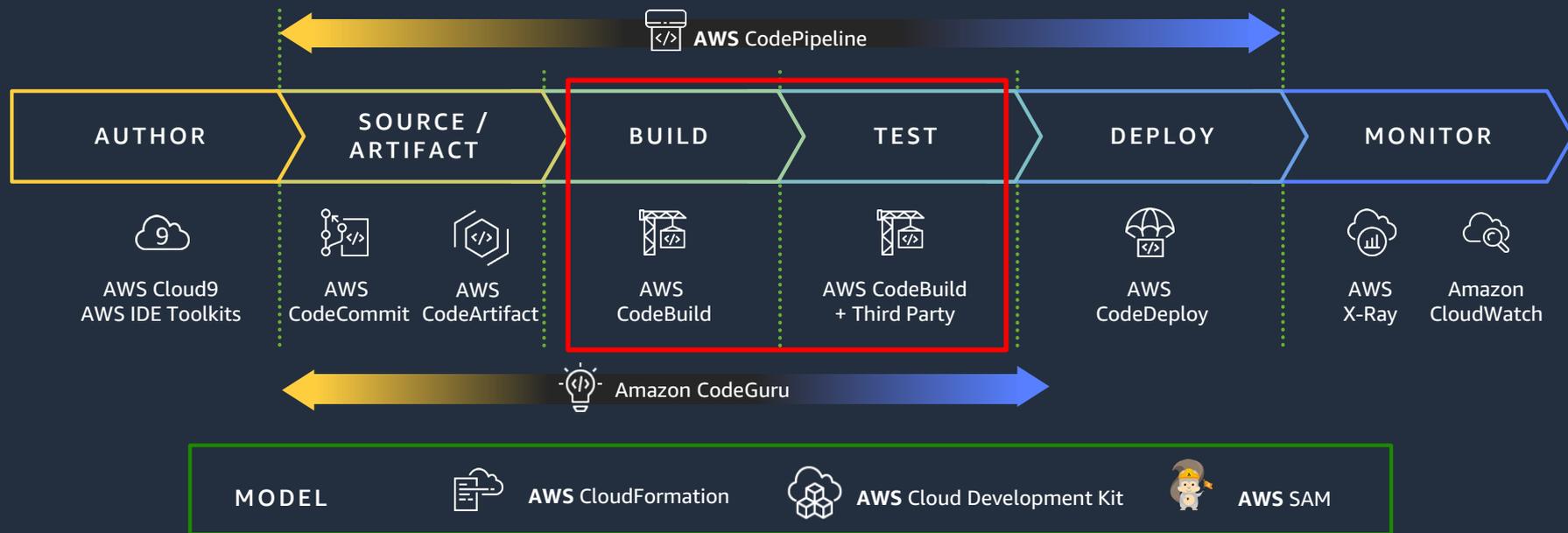
# 本日のアジェンダ

- AWS CodeBuild 概要
- buildspec.yml について
- 機能紹介
- まとめ

# ソフトウェア開発に関連する AWS サービス



# ソフトウェア開発に関連する AWS サービス



# AWS CodeBuild



- フルマネージドなビルドサービスでソースコードのコンパイル、テスト実行、ソフトウェアパッケージの作成を実行
- 継続的なスケールと同時複数ビルドプロセス
- サーバーの管理は不要
- 利用した分のみの支払い（分単位の課金）
- Amazon CloudWatch によるモニタリング可能

# AWS CodeBuild



- 一貫した**イミュータブルな環境**のために個々のビルドを新規 Docker コンテナで実行
- すべてのオフィシャルな AWS CodeBuild イメージに Docker と AWS CLI をインストール済み
- ニーズに応じて Docker イメージを作成することによって**カスタムなビルド環境**を提供可能

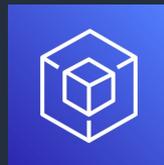
# AWS CodeBuild の実行方法



AWS Management  
Console



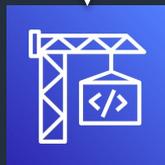
AWS Command  
Line Interface



AWS Tools  
and SDKs



AWS CodePipeline



AWS CodeBuild

# AWS CodeBuild の仕組み



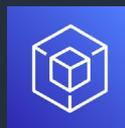
AWS CodeBuild



AWS Management  
Console



AWS Command  
Line Interface



AWS Tools  
and SDKs



AWS CodePipeline

# AWS CodeBuild の仕組み



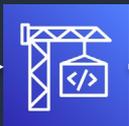
Amazon Simple Storage Service (S3)

ソースコード

作成

ビルドプロジェクト

ビルド環境



AWS CodeBuild



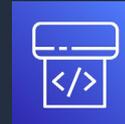
AWS Management Console



AWS Command Line Interface



AWS Tools and SDKs



AWS CodePipeline

# AWS CodeBuild の仕組み



# AWS CodeBuild の仕組み



# AWS CodeBuild の仕組み



# AWS CodeBuild プロジェクトの作成

デベロッパー用ツール > CodeBuild > ビルドプロジェクト

ビルドプロジェクト 情報

  通知 ▼    編集 ▼   **ビルドプロジェクトを作成する**

 お客様のプロジェクト ▼ < 1 > 

名前 ▼	ソースプロバイダ	リポジトリ	最新のビルドステータス
結果がありません 表示する結果はありません。			

# AWS CodeBuild プロジェクトの作成

## プロジェクト名を入力

開発者用ツール > CodeBuild > ビルドプロジェクト > ビルドプロジェクトを作成する

### ビルドプロジェクトを作成する

#### プロジェクトの設定

プロジェクト名

プロジェクト名は 2~255 文字にする必要があります。アルファベット (A~Z、a~z)、番号 (0~9)、特殊文字 (- と \_) を含めることができます。

説明 - オプション

ビルドバッジ - オプション

ビルドバッジを有効にする

▶ 追加設定

タグ

# AWS CodeBuild プロジェクトの作成

ソースとして Amazon S3 を選択し、  
バケット名と オブジェクトキーを入力

ソース ソースの追加

ソース 1 - プライマリ

ソースプロバイダ

Amazon S3

バケット

Q :odebuild-input-bucket X

S3 オブジェクトキーまたは S3 フォルダ

MessageUtil.zip

ソースバージョン - オプション情報  
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 を入力します。

ソースプロバイダ

ソースがありません ▲

ソースがありません

Amazon S3

AWS CodeCommit

GitHub

Bitbucket

GitHub Enterprise

# AWS CodeBuild プロジェクトの作成

オペレーティングシステムに Amazon Linux 2 を選択し、ランタイムとイメージを指定する

**環境**

環境イメージ

マネージド型イメージ  
AWS CodeBuild によって管理されたイメージの使用

カスタムイメージ  
Docker イメージの指定

オペレーティングシステム

Amazon Linux 2

**ランタイム**

Standard

イメージ

aws/codebuild/amazonlinux2-x86\_64-standard:3.0

④ プログラミング言語のランタイムが Ubuntu 18.04 の標準イメージに含まれるようになりました。これは、コンソールで作成される新しい CodeBuild プロジェクトに推奨されています。詳細については、CodeBuild で提供される Docker イメージ を参照してください。

イメージのバージョン

このランタイムバージョンには常に最新のイメージを使用してください

環境タイプ

Linux

特権付与

Docker イメージを構築するか、ビルドで昇格されたアクセス権限を取得するには、このフラグを有効にします

サービスロール

新しいサービスロール  
アカウントでサービスロールを作成

既存のサービスロール  
アカウントから既存のサービスロールを選択

ロール名

codebuild-codebuild-demo-project-service-role

サービスロール名の入力

▶ 追加設定  
タイムアウト、証明書、VPC、コンピューティングタイプ、環境変数、ファイルシステム

# AWS CodeBuild プロジェクトの作成

## ビルド仕様は buildspec ファイルを使用するを選択

### Buildspec

#### ビルド仕様

buildspec ファイルを使用する

build コマンドを YAML 形式の buildspec ファイルに保存

ビルドコマンドの挿入

ビルドプロジェクト設定としてビルドコマンドを保存

#### Buildspec 名 - オプション

CodeBuild のデフォルトでは、buildspec.yml という名前のファイルがソースコードルートディレクトリで検索されます。buildspec ファイルに別の名前または場所を使用している場合は、ここにソースルートからのパス (buildspec-two.yml や configuration/buildspec.yml など) を入力します。

#### バッチ設定

ビルドのグループを 1 つの実行として実行できます。バッチ設定は、ビルドの開始時にアドバンスオプションでも使用できます。

バッチ設定を定義 - オプション

ビルドバッチの開始時にバッチ設定を定義または上書きすることもできます。

#### ビルドコマンド

```
1 version: 0.2
2
3 #env:
4 #variables:
5 # key: "value"
6 # key: "value"
7 #parameter-store:
8 # key: "value"
9 # key: "value"
10 #secrets-manager:
11 # key: secret-id:json-key:version-stage:version-id
12 # key: secret-id:json-key:version-stage:version-id
13 #exported-variables:
14 # - variable
15 # - variable
16 #git-credential-helper: yes
17 #batch:
18 #fast-fail: true
19 #build-list:
20 #build-matrix:
21 #build-graph:
22 #phases:
```

1 行に切り替える

# AWS CodeBuild プロジェクトの作成

アーティファクトのタイプに Amazon S3 を選択し、  
バケット名を入力する

### アーティファクト

アーティファクトの追加

アーティファクト 1 - プライマリ

タイプ

テストを実行するか Docker イメージを Amazon ECR にプッシュする場合は、[アーティファクトなし] を選択できます。

バケット名

名前

出力アーティファクトを含むバケット内のフォルダまたは圧縮ファイルの名前です。フォルダを使用するか圧縮ファイルを使用するかは、詳細設定にあるアーティファクトのパッケージから選択します。名前が設定されない場合は、デフォルト名が使われます。

セマンティックバージョンングの有効化  
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

パス - オプション  
ビルド出力 ZIP ファイルまたはフォルダのパス。

例: MyPath/MyArtifact.zip。

名前空間のタイプ - オプション

[ビルド ID] を選択して、ビルド出力 ZIP ファイルまたはフォルダへのパスにビルド ID を挿入します (例: MyPath/MyBuildID/MyArtifact.zip)。それ以外の場合、[なし] を選択します。

アーティファクトのパッケージ化

なし  
アーティファクトファイルはバケットにアップロードされます。

Zip  
AWS CodeBuild は、アーティファクトを圧縮ファイルに保存し、指定されたバケットにアップロードします。

アーティファクト暗号化の削除  
アーティファクトを使用して静的なウェブサイトを開示したり、他のユーザーとコンテンツを共有したりする場合は、暗号化を削除します。

▶ 追加設定  
キャッシュ、暗号化キー

# AWS CodeBuild プロジェクトの作成

## CloudWatch Logs へログを送信する

### ログ

CloudWatch

**CloudWatch Logs - オプション**  
このオプションをチェックすると、ビルド出力ログが CloudWatch にアップロードされます。

グループ名

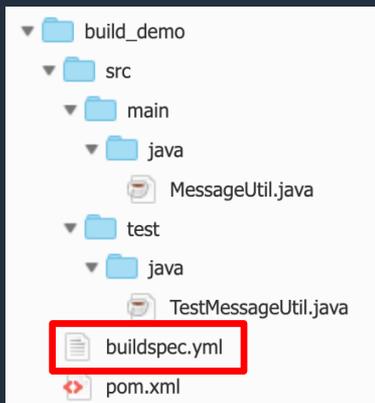
ストリーム名

S3

**S3 ログ - オプション**  
このオプションをチェックすると、ビルド出力ログが S3 にアップロードされます。

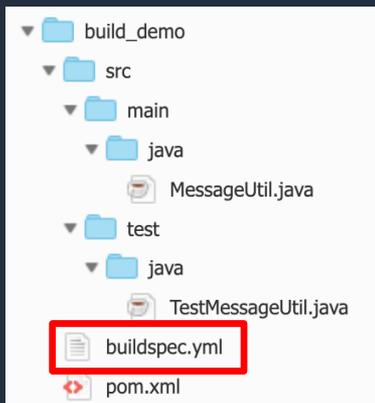
キャンセル

# ソースの作成



```
buildspec.yml x (+)
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
```

# ソースの作成



buildspec.yml

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

buildspec のバージョン

ランタイムに Java を指定

jar ファイルを作成

ビルド出力を定義

# ビルドの実行

デベロッパー用ツール > CodeBuild > ビルドプロジェクト > codebuild-demo-project

## codebuild-demo-project

通知 ▼ 共有 編集 ▼ ビルドプロジェクトの削除 **ビルドを開始**

### 設定

ソースプロバイダ Amazon S3	プライマリリポジトリ -codebuild-input-bucket/MessageUtil.zip <a href="#">🔗</a>	アーティファクトのアップロード場所 -codebuild-output-bucket	ビルドバッジ 無効
-----------------------	---	---	--------------

**ビルド履歴** | バッチ履歴 | ビルドの詳細 | ビルドのトリガー | メトリクス

### ビルド履歴

🔄 ビルドの停止 アーティファクトの表示 ログの表示 ビルドの削除 ビルドの再試行

< 1 > ⚙️

■	ビルドの実行	ステータス	ビルド番号	ソースバージョン	送信者	期間	完了済み
結果がありません 表示する結果はありません。							

**ビルドを開始**

# ビルドの実行

🕒 ビルドが開始されました

次のビルドが正常に開始されました: codebuild-demo-project:778cbeef-7710-4d6c-b59d-f75ce10a3477



デベロッパー用ツール > CodeBuild > ビルドプロジェクト > codebuild-demo-project > codebuild-demo-project:778cbeef-7710-4d6c-b59d-f75ce10a3477

codebuild-demo-project:778cbeef-7710-4d6c-b59d-f75ce10a3477

ビルドの停止

ビルドの再試行

## ビルドステータス

ステータス

進行中

イニシエータ

Admin/

ビルド ARN

arn:aws:codebuild:us-west-  
:build/codebuild-demo-  
project:778cbeef-7710-4d6c-b59d-  
f75ce10a3477

解決済みソースバージョン

-

開始時刻

11月 25, 2020 11:02 午前 (UTC+9:00)

終了時刻

-

ビルド番号

19

ビルドログ

フェーズ詳細

レポート

環境変数

ビルドの詳細

リソース使用率

ビルドログの最後の 1000 行を表示しています。 [ログ全体の表示](#)

テールログ

1

# 本日のアジェンダ

- AWS CodeBuild 概要
- `buildspec.yml` について
- 機能紹介
- まとめ

# buildspec.yml の構成

buildspec のバージョン	version:
コマンドを実行する Linux ユーザ	run-as:
環境変数	env:
プロキシサーバ設定	proxy:
バッチビルド設定	batch:
実行するコマンド	phases:
テストレポート作成	reports:
AWS CodeBuild の出力	artifacts:
キャッシュ設定	cache:

# buildspec.yml の構成

(必須) buildspec のバージョン	version:
コマンドを実行する Linux ユーザ	run-as:
環境変数	env:
プロキシサーバ設定	proxy:
バッチビルド設定	batch:
(必須) 実行するコマンド	phases:
テストレポート作成	reports:
AWS CodeBuild の出力	artifacts:
キャッシュ設定	cache:

# version

buildspec のバージョン

**version:** 0.2

- 0.2 の使用を推奨
  - 0.1 も引き続き利用可能
- バージョン 0.2 での変更点
  - `environment_variables` → `env`
  - `plaintext` → `variables`
  - `artifacts` の `type` プロパティ廃止
  - 同一インスタンスで全てのビルドコマンドを実行

# version

version 0.1 は各コマンドが独立して実行される

phases:

build:

commands:

```
{ - SERVICE_NAME=CodeBuild }  
{ - echo ${SERVICE_NAME} }
```

version: 0.1 の場合

Running command echo \${SERVICE\_NAME}



version: 0.2 の場合

Running command echo \${SERVICE\_NAME}

CodeBuild

# run-as

コマンドを実行する Linux ユーザ

`run-as`: Linux-user-name

phases:

  build:

`run-as`: Linux-user-name-02

- Linux 環境でのみ使用可能
- 指定したユーザに読み取りおよび実行権限を付与
- 指定しない場合、すべてのコマンドが root ユーザで実行される
- Phases ブロックでオーバーライド可能

# env

## 環境変数

### env:

shell: shell-tag

variables:

key: "value"

parameter-store:

key: "value"

exported-variables:

- variable

secrets-manager:

key: secret-id:json-key:version-stage:version-id

git-credential-helper: no | yes

- 環境変数には以下を指定可能
  - プレーンテキスト
  - AWS Systems Manager Parameter Store の値
  - AWS Secrets Manager の値
- 利用するシェル
  - Bash, /bin/sh
  - Powershell.exe, cmd.exe
- Git 認証ヘルパーを利用するか

# proxy

## プロキシサーバ設定

### proxy:

upload-artifacts: no | yes

logs: no | yes

- アーティファクトのアップロード時、CloudWatch Logs へのログ送信時にプロキシサーバを利用するか個別に定義可能
- 明示的なプロキシサーバを利用する場合、環境変数の設定が必要
  - HTTP\_PROXY, HTTPS\_PROXY, NO\_PROXY

# batch

## バッチビルド設定

### batch:

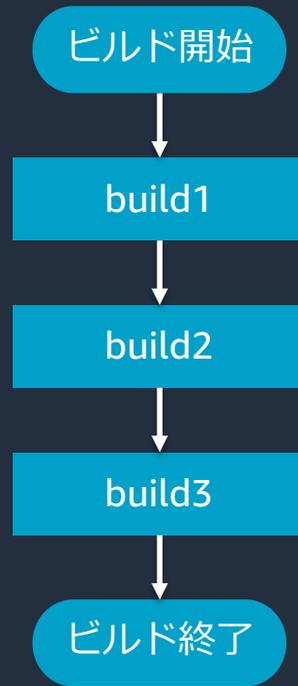
```
fast-fail: false | true
# build-list:
# build-matrix:
# build-graph:
```

- プロジェクトの同時実行と協調実行の定義を行う
- バッチビルドには 3 つのタイプがある
  - build-graph
  - build-list
  - build-matrix

# batch/build-graph

バッチ内の他のタスクに依存する一連のタスクを定義

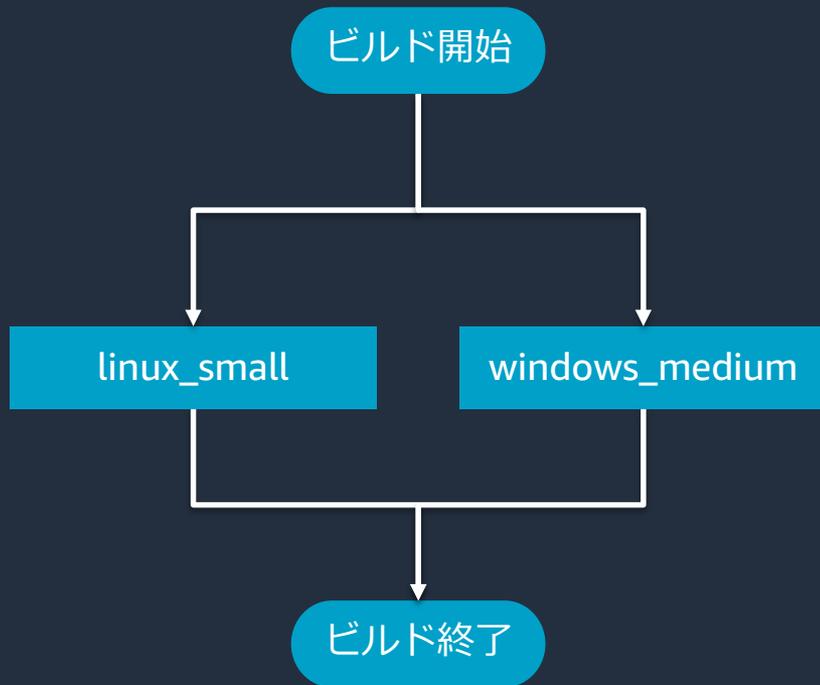
```
batch:  
  fast-fail: false  
  build-graph:  
    - identifier: build1  
      env:  
        compute-type: BUILD_GENERAL1_SMALL  
    - identifier: build2  
      env:  
        compute-type: BUILD_GENERAL1_MEDIUM  
      depend-on:  
        - build1  
    - identifier: build3  
      env:  
        compute-type: BUILD_GENERAL1_LARGE  
      depend-on:  
        - build2
```



# batch/build-list

同時に実行されるタスクを定義

```
batch:  
  fast-fail: false  
  build-list:  
    ignore-failure: true  
    - identifier: linux_small  
      env:  
        compute-type: BUILD_GENERAL1_SMALL  
    - identifier: windows_medium  
      env:  
        type: WINDOWS_SERVER_2019_CONTAINER  
        image: aws/codebuild/windows-base:2019-1.0  
        compute-type: BUILD_GENERAL1_MEDIUM
```



# batch/build-matrix

さまざまな環境と並行して実行されるタスクを定義

batch:

**build-matrix:**

static:

ignore-failure: false

env:

type: LINUX\_CONTAINER

privileged-mode: true

dynamic:

env:

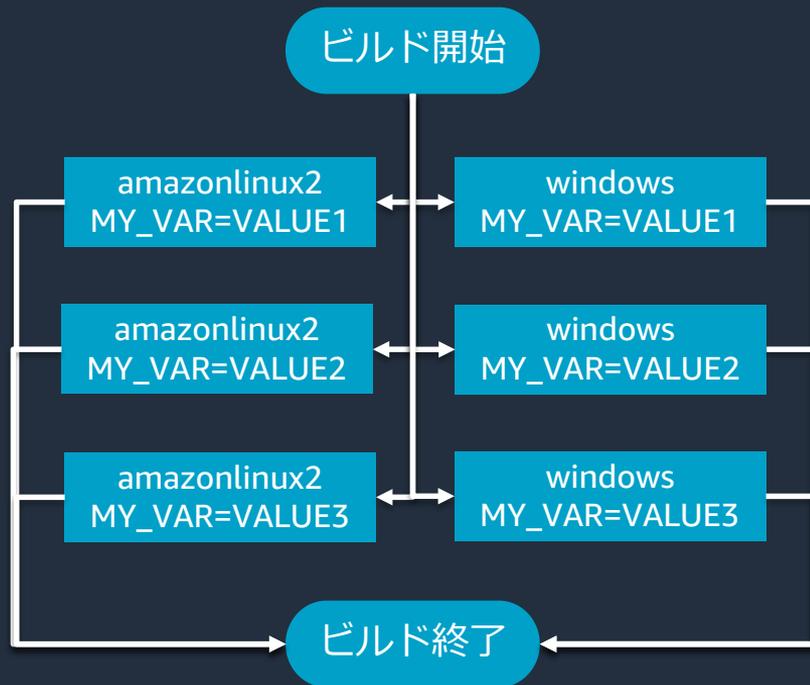
image:

- aws/codebuild/amazonlinux2-x86\_64-standard:3.0
- aws/codebuild/windows-base:2019-1.0

variables:

**MY\_VAR:**

- VALUE1
- VALUE2
- VALUE3



# phases

## ビルドの各段階で CodeBuild が実行するコマンド記述する

### phases:

#### install:

runtime-versions:

runtime: version

runtime: version

#### pre\_build:

#### build:

#### post\_build:

run-as: Linux-user-name

commands:

- command

- command

finally:

- command

- command

#### install

パッケージのインストールのみに利用することを推奨

runtime-versions でランタイムを指定可能

#### pre\_build

ビルド前に実行するコマンドを記述

Amazon ECR へのサインイン、npm の依存関係インストールなど

#### build

ビルド中に実行するコマンドを記述

#### post\_build

ビルド後に実行するコマンドを記述

ビルドアーティファクトを jar, war にする,

Docker イメージを Amazon ECR へ Push など

# reports

## テストレポートの作成

### reports:

report-group-name-or-arn:

files:

- location

- location

base-directory: location

discard-paths: no | yes

file-format: report-format

- テストレポート とコードカバレッジレポートの2種類

# reports

テストレポート ( JunitXMLの例 )

reports:

junit-report:

files:

- 'target/surefire-reports/TEST-TestMessageUtil.xml'

file-format: JUNITXML

- 以下のフォーマットに対応
  - Cucumber JSON, JUnitXML, NUnitXML, NUnit3 XML, TestNGXML, Visual Studio TRX

# reports

## テストレポート



テストケース

任意のステータス ▼ 詳細を表示

Q

テストケース	ステータス	プレフィックス	メッセージ	期間
testError02	失敗	TestMessageUtil	org.junit.ComparisonFail...	0
testError01	失敗	TestMessageUtil	org.junit.ComparisonFail...	0.013秒間
testOK	成功	TestMessageUtil	-	0
testPrintMessage	成功	TestMessageUtil	-	0
testSalutationMessage	成功	TestMessageUtil	-	0.012秒間



# reports

コードカバレッジレポート ( JaCoCoXML の例 )

reports:

```
jacoco-report:
```

```
  files:
```

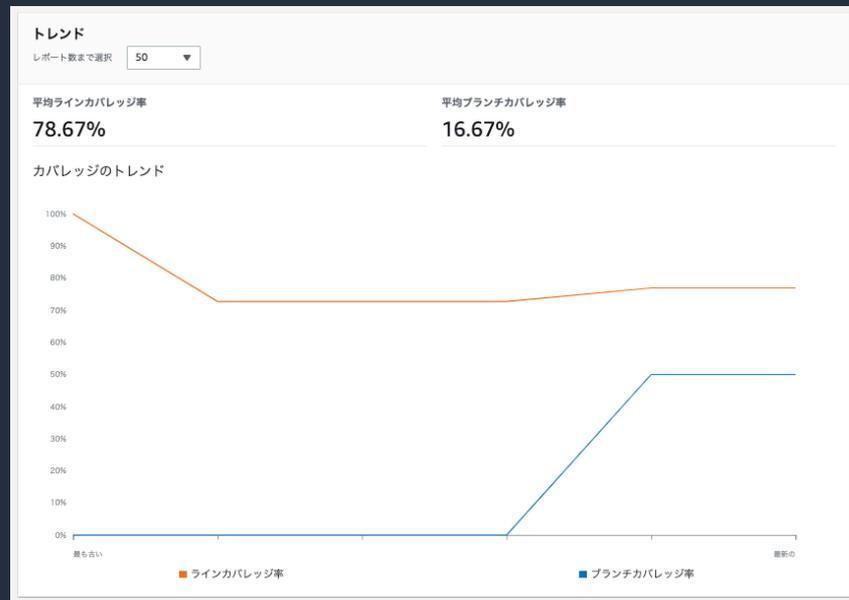
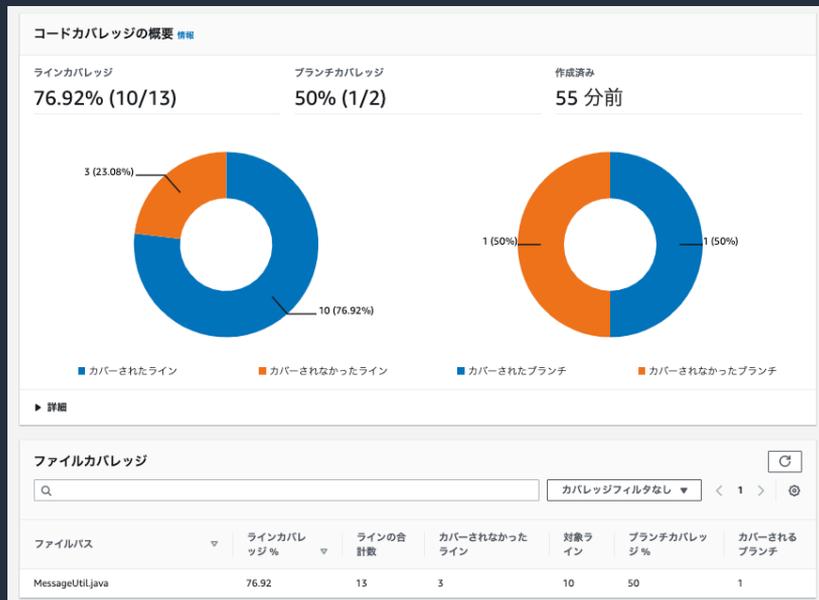
```
    - 'target/site/jacoco/jacoco.xml'
```

```
  file-format: 'JACOCOXML'
```

- 以下のフォーマットに対応
  - Clover XML, Cobertura XML, JaCoCoXML, SimpleCovJSON

# reports

## コードカバレッジレポート



# reports

## コードカバレッジレポート

- ラインカバレッジ ( C0 : 命令網羅率 )
  - $\text{line coverage} = (\text{total lines covered}) / (\text{total number of lines})$
  - コードのうち、テストで実行された行の割合
- ブランチカバレッジ ( C1 : 分岐網羅率 )
  - $\text{branch coverage} = (\text{total branches covered}) / (\text{total number of branches})$
  - 分岐のうち、テストで実行された分岐の割合

# artifacts

## CodeBuild の出力を定義

### artifacts:

files:

- location

name: artifact-name

discard-paths: no | yes

base-directory: location

secondary-artifacts:

artifactIdentifier:

files:

- location

name: secondary-artifact-name

discard-paths: no | yes

base-directory: location

- アーティファクトの名、アーティファクトに含めるサブディレクトリとファイルを指定
- secondary-artifacts を利用して、複数のビルド出力アーティファクトを定義することも可能

# cache

## キャッシュの設定

cache:

paths:

- path
- path

キャッシュタイプ
キャッシュなし ▼
Amazon S3
ローカル
キャッシュなし ▲

- キャッシュタイプはビルドプロジェクトに対して設定
  - キャッシュなし、S3、ローカル
- S3 キャッシュと、ローカルキャッシュのカスタムキャッシュは buildspec でキャッシュ対象を指定する

# cache

## S3 キャッシュの設定

cache:

paths:

- \${ディレクトリ}
- \${ファイル}

The screenshot shows the configuration interface for an S3 cache. It includes a dropdown menu for 'キャッシュタイプ' (Cache Type) set to 'Amazon S3'. Below it is a 'キャッシュバケット' (Cache Bucket) field containing 'lebuild-output-bucket'. There is an empty field for 'キャッシュパスのプレフィックス - オプションル' (Cache Path Prefix - Optional). A section for 'キャッシュのライフサイクル - オプションル' (Cache Lifecycle - Optional) contains a field with '0' and a button '+ 有効期限を追加する' (Add Expiry). A small note explains that the lifecycle prefix is applied to all objects or a subset of objects in the cache bucket.

- S3 キャッシュでは個別のファイルを指定可能

# cache

## ローカル キャッシュの設定

cache:

paths:

- \${ディレクトリ}

キャッシュタイプ

ローカル ▼

少なくとも 1 つのオプションを選択してください。

- DockerLayerCache**  
Docker-in-Docker ストレージをインスタンスストレージにマウントし、ビルドでキャッシュされた Dockerfile レイヤーを再利用できるようにします。
- SourceCache**  
インスタンスストレージで .git メタデータを保存します。将来のビルドでは、デルタコミットをプルするだけで済みます。
- CustomCache**  
有効な場合、実行時に buildspec.yml から追加のキャッシュパスを読み取ります。

- ローカルキャッシュのカスタムキャッシュはディレクトリのみ指定可能

# cache

## ローカル キャッシュ

- ソースキャッシュ
  - Git メタデータをキャッシュ
  - コミット間の変更のみが Pull される
- Docker レイヤーキャッシュ
  - Docker レイヤーをキャッシュ
  - Linux 環境のみ使用可能
- カスタムキャッシュ
  - 上記に該当しない場合
  - 個々のファイルは指定できず、ディレクトリのみ指定可能

# 本日のアジェンダ

- AWS CodeBuild 概要
- buildspec.yml について
- 機能紹介
- まとめ

# 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、ローカルマシンでビルド可能

- Git と Docker が必要

ビルドイメージの設定

```
git clone https://github.com/aws/aws-codebuild-docker-images.git
cd aws-codebuild-docker-images/ubuntu/standard/4.0
docker build -t aws/codebuild/standard:4.0 .
docker pull amazon/aws-codebuild-local:latest --disable-content-trust=false
```

CodeBuild エージェントのイメージとスクリプトを取得

```
docker pull amazon/aws-codebuild-local:latest --disable-content-trust=false
wget https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/master/local_builds/codebuild_build.sh
chmod +x codebuild_build.sh
```

# 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、ローカルマシンでビルド可能

- スクリプトはビルドイメージを起動し、カレントディレクトリにあるプロジェクトでビルドを実行する
- ビルドプロジェクトの場所を指定するには `-s` オプションを利用する
- 出力ディレクトリの指定には `-a` オプションを利用する

カレントディレクトリのプロジェクトでビルド

```
./codebuild_build.sh -i aws/codebuild/standard:4.0 -a <output directory>
```

プロジェクトの場所を指定してビルド

```
./codebuild_build.sh -i aws/codebuild/standard:4.0 -s <build project directory> -a <output directory>
```

# AWS Session Manager でビルド環境へアクセスする

マネジメントコンソール、CLI を介して Linux, Windows のビルド環境にアクセス

- `buildspec.yml` に `codebuild-breakpoint` を埋め込み、ビルド設定でセッション接続を有効にする
- Session Manager でビルド環境にアクセスし、調査を実施
- ビルドを再開するには `codebuild-resume` を実行する

# AWS Session Manager でビルド環境へアクセスする

codebuild-breakpoint コマンドでブレイクポイントをセット

```
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`
      - codebuild-breakpoint
```

# AWS Session Manager でビルド環境へアクセスする

開発者用ツール > CodeBuild > ビルドプロジェクト > codebuild-demo-project > 新しいビルドの開始

## ビルドを開始

高度なビルドの上書き

### ビルド設定

プロジェクト

codebuild-demo-project ▼

ビルドタイプ

単一ビルド  
単一ビルドをトリガー

バッチビルド  
Webhook は、複数のビルドを 1 つの実行としてトリガーします

# Session Manager でビルド環境へアクセスする

環境

現在の環境イメージ  
aws/codebuild/amazonlinux2-x86\_64-standard:3.0

イメージの上書き

セッション接続  
インタラクティブなブラウザベースのシェルを使用してビルドマシンのコンテナを管理するように AWS Session Manager をセットアップします。セッション接続を有効にするための AWS Session Manager 権限が必要です。 [詳細](#)

セッション接続の有効化

サービスロール  
アカウントから既存のサービスロールを選択

arn:aws:iam:::role/service-role/codebuild-codebuild-demo-proj X

AWS CodeBuild にこのサービスロールの編集を許可し、このビルドプロジェクトでの使用を可能にする

追加設定  
タイムアウト、証明書、コンピューティングタイプ、環境変数

S3

S3 ログ - オプション  
このオプションをチェックすると、ビルド出力ログが S3 にアップロードされます。

キャンセル

チェックを入れると必要な権限が  
ロールに追加される

# Session Manager でビルド環境へアクセスする

デベロッパーツール > CodeBuild > ビルドプロジェクト > codebuild-demo-project > codebuild-demo-project:acb78081-72fc-47a2-b42e-1fb6acf3cabf

codebuild-demo-project:acb78081-72fc-47a2-b42e-1fb6acf3cabf ビルドの停止 ビルドの再試行

### ビルドステータス

ステータス	イニシエータ	ビルド ARN	解決済みソースバージョン
<span>🔄</span> 進行中	Admin/n rd	arn:aws:codebuild:us-west-1fb6acf3cabf	-
開始時刻	終了時刻	ビルド番号	
11月 25, 2020 9:54 午前 (UTC+9:00)	-	10	

[ビルドログ](#) | [フェーズ詳細](#) | [レポート](#) | [環境変数](#) | [ビルドの詳細](#) | [リソース使用率](#)

ビルドログの最後の 1000 行を表示しています。 [ログ全体の表示](#) テールログ

1

# Session Manager でビルド環境へアクセスする

デベロッパー用ツール > CodeBuild > ビルドプロジェクト > codebuild-demo-project > codebuild-demo-project:acb78081-72fc-47a2-b42e-1fb6acf3cabf

codebuild-demo-project:acb78081-72fc-47a2-b42e-1fb6acf3cabf ビルドの停止 ビルドの再試行

### ビルドステータス

ステータス	イニシエータ	ビルド ARN	解決済みソースバージョン
<a href="#">進行中</a>	Admin/	arn:aws:codebuild:us-west-	39fa00d86d4afe5f26b934b82e915974ac27420c
		1100d13cab1	
開始時刻	終了時刻	ビルド番号	<b>AWS セッションマネージャー</b> <b>AWS セッションマネージャー</b> <a href="#">🔗</a>
11月 25, 2020 9:54 午前 (UTC+9:00)	-	10	

```
3912 [Container] 2020/11/25 00:55:26 Phase complete: BUILD State: SUCCEEDED
3913 [Container] 2020/11/25 00:55:26 Phase context status code: Message:
3914 [Container] 2020/11/25 00:55:26 Entering phase POST_BUILD
3915 [Container] 2020/11/25 00:55:26 Running command echo Build completed on `date`
3916 Build completed on Wed Nov 25 00:55:26 UTC 2020
3917
3918 [Container] 2020/11/25 00:55:26 Running command codebuild-breakpoint
3919 2020/11/25 00:55:26 Build is paused temporarily and you can use codebuild-resume command in the session to resume this build
3920
```

# Session Manager でビルド環境へアクセスする

codebuild-resume コマンドでビルドを再開

セッション ID:	21bd63af338f8b	インスタンス
<pre>sh-4.2# ls -al total 24 drwxr-xr-x 4 root root 4096 Nov 25 00:57 . drwxr-xr-x 3 root root 4096 Nov 25 00:55 .. -rw-r--r-- 1 root root 402 Nov 25 00:55 buildspec.yml -rw-r--r-- 1 root root 1492 Nov 25 00:55 pom.xml drwxr-xr-x 4 root root 4096 Nov 25 00:55 src drwxr-xr-x 9 root root 4096 Nov 25 00:55 target sh-4.2# ls -al target total 40 drwxr-xr-x 9 root root 4096 Nov 25 00:55 . drwxr-xr-x 4 root root 4096 Nov 25 00:57 .. drwxr-xr-x 2 root root 4096 Nov 25 00:55 classes drwxr-xr-x 3 root root 4096 Nov 25 00:55 generated-sources drwxr-xr-x 3 root root 4096 Nov 25 00:55 generated-test-sources drwxr-xr-x 2 root root 4096 Nov 25 00:55 maven-archiver drwxr-xr-x 3 root root 4096 Nov 25 00:55 maven-status -rw-r--r-- 1 root root 2224 Nov 25 00:55 messageUtil-1.0.jar drwxr-xr-x 2 root root 4096 Nov 25 00:55 surefire-reports drwxr-xr-x 2 root root 4096 Nov 25 00:55 test-classes sh-4.2#</pre>		

ビルドを再開する

```
codebuild-resume
```

# ビルドバッジ

埋め込み可能なイメージ（バッジ）として動的に生成されるステータス画像

 AWS CodeBuild **unknown**

 AWS CodeBuild **in progress**

 AWS CodeBuild **passing**

 AWS CodeBuild **failing**

- ビルドプロジェクトに対して生成されるパブリック可能な URL で認証不要
- CodeCommit や GitHub のマークダウンファイルに埋め込み、ビルドステータスを表示することが可能

# ステータスの通知

通知ルールを設定してビルド状況の通知を受けることが可能

### 通知ルールの設定

通知名

詳細タイプ  
通知に必要な詳細レベルを選択します。 [通知とセキュリティの詳細](#)

フル  
リソースまたは通知機能によって提供されるイベントに関する補足情報が含まれます。

ベーシック  
リソースイベントで提供される情報のみが含まれます。

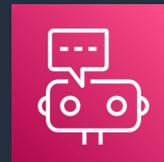
---

### 通知をトリガーするイベント

Build state	Build phase
<input type="checkbox"/> Failed	<input type="checkbox"/> Failure
<input type="checkbox"/> Succeeded	<input type="checkbox"/> Success
<input type="checkbox"/> In-progress	
<input type="checkbox"/> Stopped	



Amazon Simple  
Notification Service



AWS Chatbot

# VPC 内リソースへのアクセス

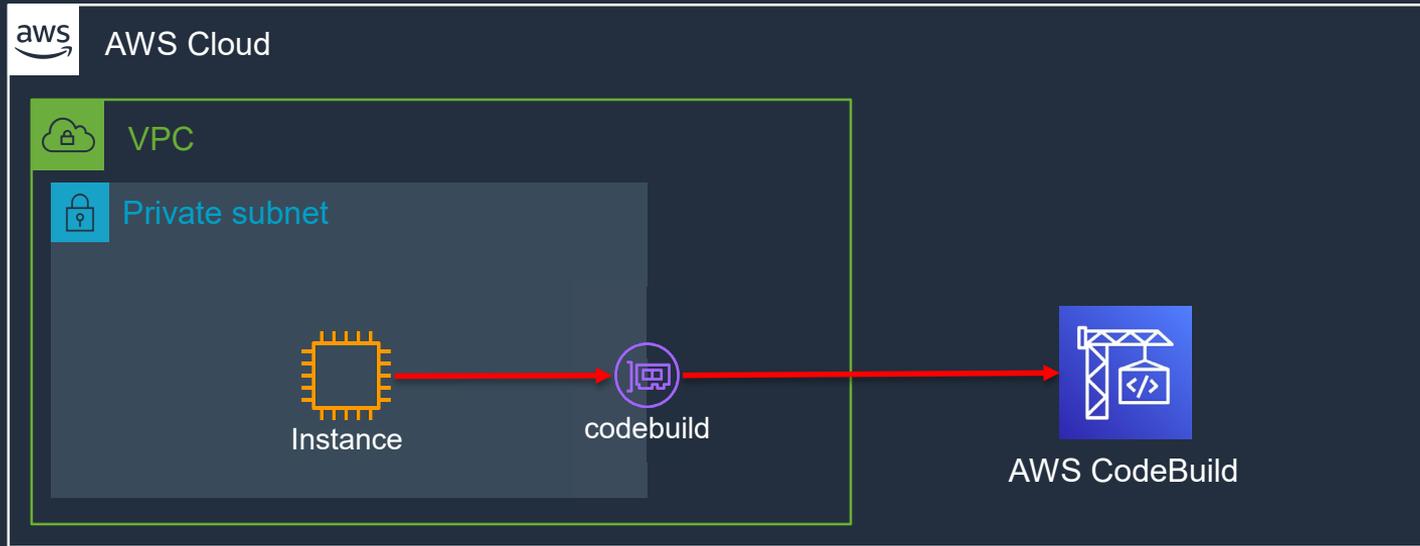
- プライベートサブネット内のリソースへアクセス
- VPCエンドポイント経由でのみアクセスできるように設定されたサービスへのアクセス
- 固定 IP アドレスを必要とする外部サービスへの通信に NAT ゲートウェイ、NAT インスタンスの EIP を使用する

## 注意点

- NAT ゲートウェイ、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利用することはできない

# VPC エンドポイントをサポート

- 連邦情報処理標準(FIPS) 準拠オプションもあり



# 利用料金

## 料金（東京リージョン）

インスタンスタイプ	メモリ	vCPU	Linux ビルド（1分）	Windows ビルド（1分）
general1.small	3 GB	2	0.005USD	該当なし
general1.medium	7 GB	4	0.01USD	
arm1.large	16 GiB	8	0.02USD	
general1.large	15 GB	8	0.02USD	
general1.2xlarge	144 GiB	72	0.25USD	
gpu1.large	244 GiB	32	0.90USD	

general1.small で、1 回 5 分、月に 300 回 ビルドした場合  
 $0.005\text{USD} * 5 \text{分} * 300 \text{回} = 7.5\text{USD}$

無料利用枠

general1.small 100 分（月）

# 本日のアジェンダ

- AWS CodeBuild 概要
- buildspec.yml について
- 機能紹介
- まとめ

# まとめ

- AWS CodeBuild は、フルマネージドなビルドサービスでソースコードのコンパイル、テスト実行、ソフトウェアパッケージの作成を実行可能
- ローカル環境でのビルド、ビルド環境へログインしてのデバッグも可能
- 22種類のユースケースベースのサンプル
  - [https://docs.aws.amazon.com/ja\\_jp/codebuild/latest/userguide/use-case-based-samples.html](https://docs.aws.amazon.com/ja_jp/codebuild/latest/userguide/use-case-based-samples.html)

# Q&A

お答えできなかったご質問については

AWS Japan Blog 「<https://aws.amazon.com/jp/blogs/news/>」にて

後日掲載します。

# AWS の日本語資料の場所「AWS 資料」で検索



日本担当チームへお問い合わせ サポート 日本語 ▾ アカウント ▾

コンソールにサインイン

製品 ソリューション 料金 ドキュメント 学習 パートナー AWS Marketplace その他 🔍

## AWS クラウドサービス活用資料集トップ

アマゾン ウェブ サービス (AWS) は安全なクラウドサービスプラットフォームで、ビジネスのスケールと成長をサポートする処理能力、データベースストレージ、およびその他多種多様な機能を提供します。お客様は必要なサービスを選択し、必要な分だけご利用いただけます。それらを活用するために役立つ日本語資料、動画コンテンツを多数ご提供しております。(本サイトは主に、AWS Webinar で使用した資料およびオンデマンドセミナー情報を掲載しています。)

[AWS Webinar お申込 »](#)

[AWS 初心者向け »](#)

[業種・ソリューション別資料 »](#)

[サービス別資料 »](#)

<https://amzn.to/JPArchive>



# AWS Well-Architected 個別技術相談会

毎週“W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に  
対策などを相談することも可能

- **申込みはイベント告知サイトから**

(<https://aws.amazon.com/jp/about-aws/events/>)

AWS イベント

で[検索]



AWS Well-Architected



# ご視聴ありがとうございました

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>

