[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

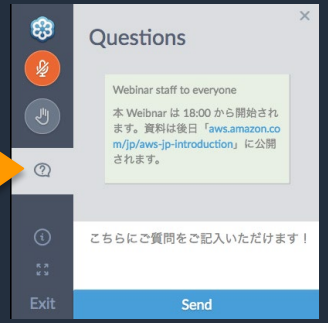
AWS Black Belt Online Seminar とは

「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います。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

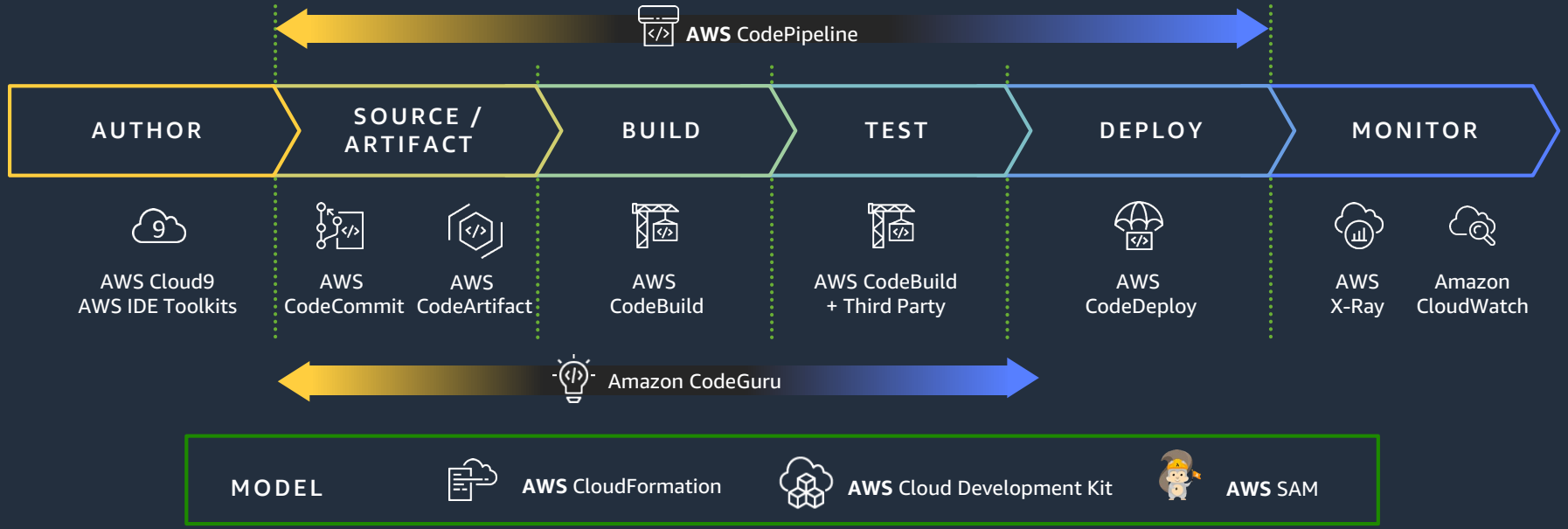
- 本セミナーで学習できること
    - AWS CodeBuild

- 対象者
    - アーキテクトの方
    - 技術者の方

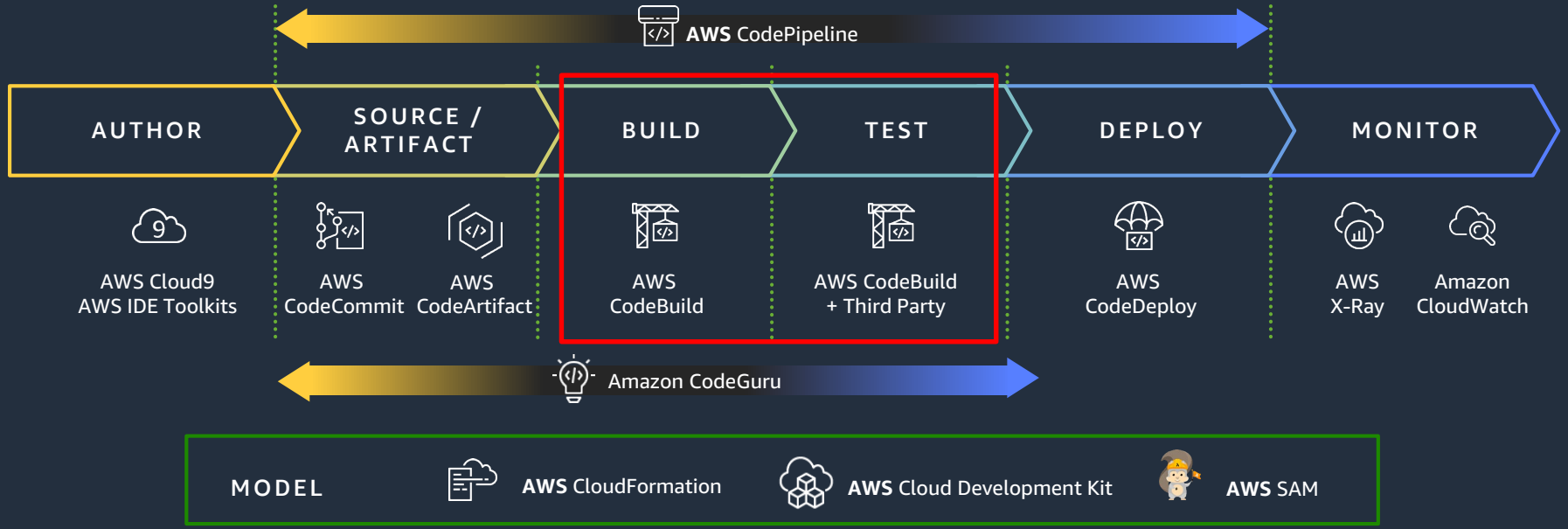
本日のアジェンダ

- 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



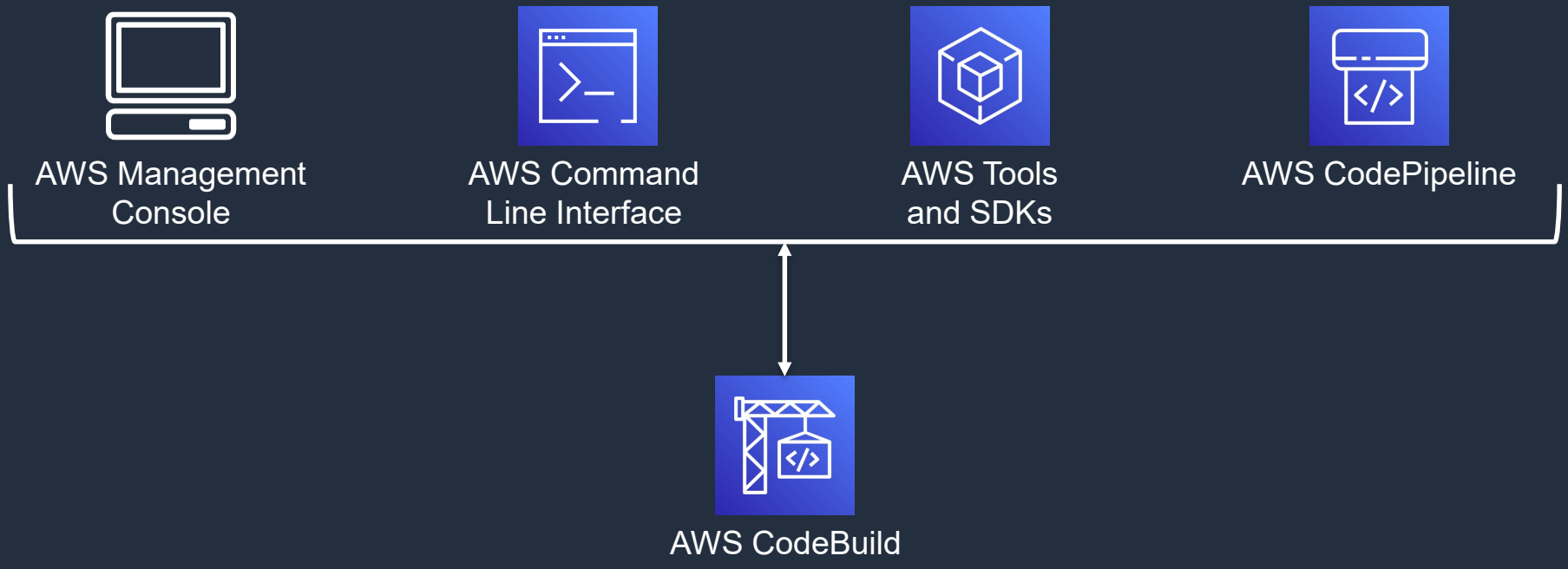
AWS CodeBuild

- 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

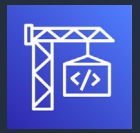
AWS CodeBuild

- 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 CodeBuild の仕組み



AWS CodeBuild

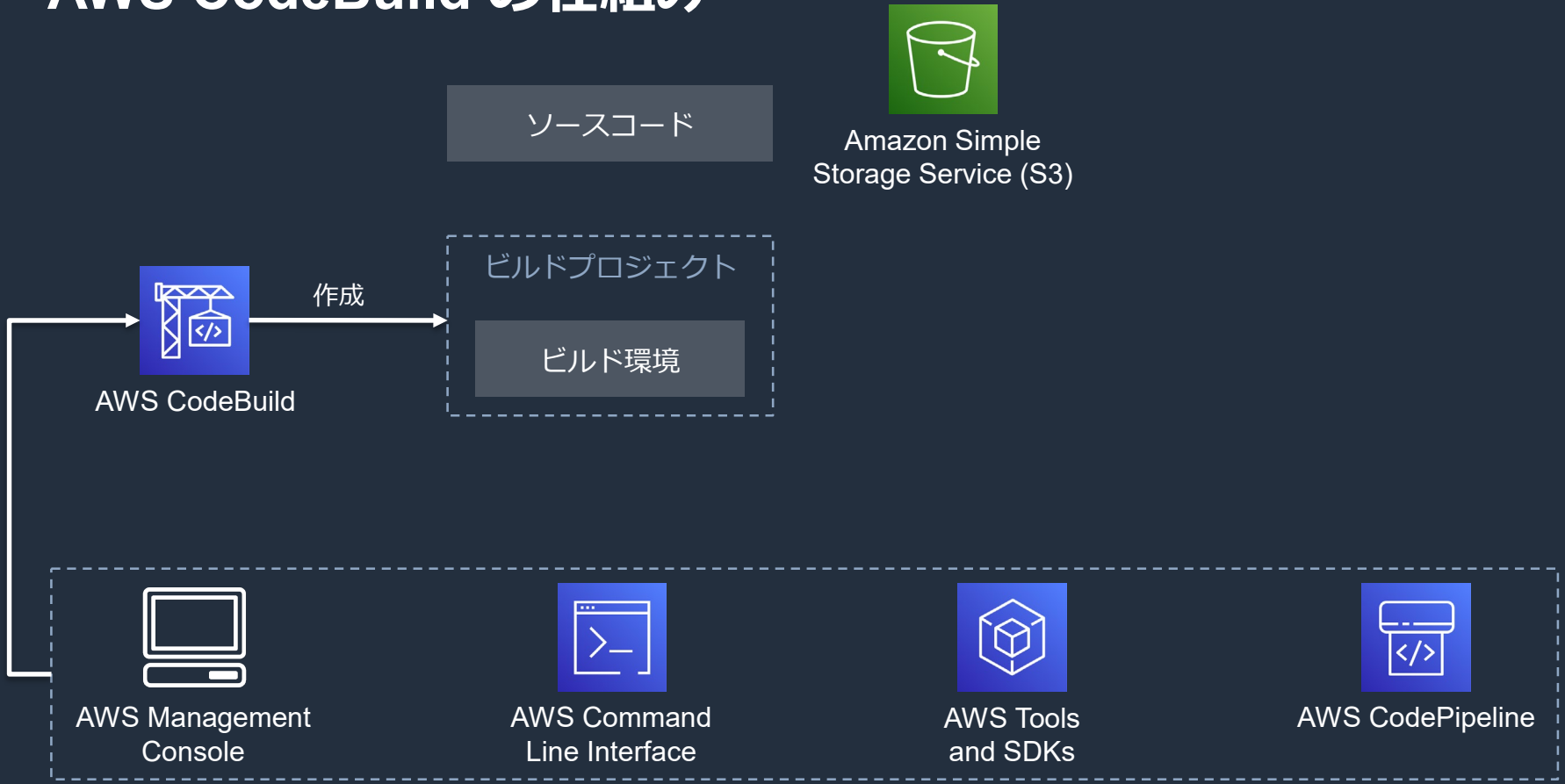
AWS Management Console

AWS Command Line Interface

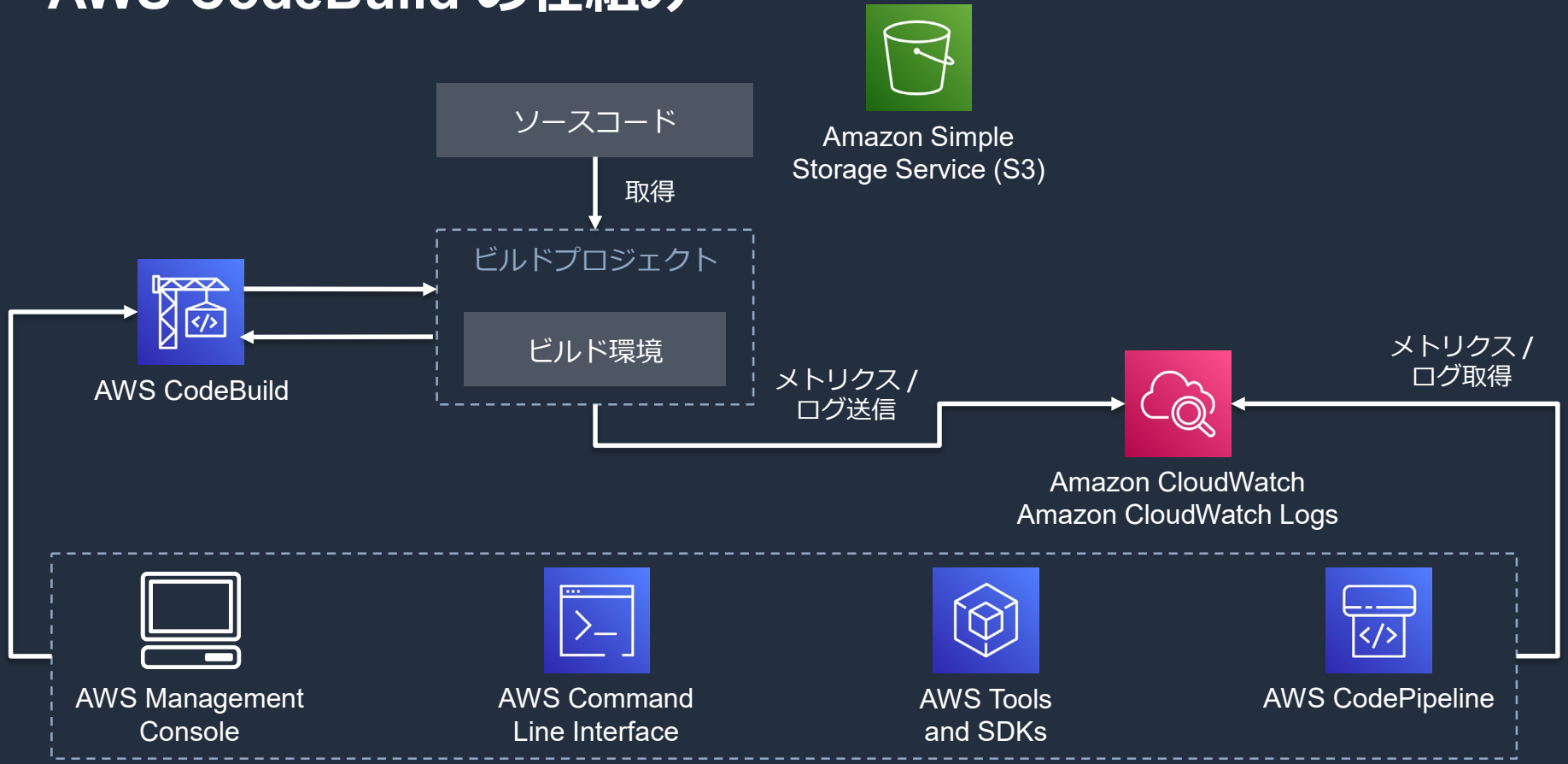
AWS Tools and SDKs

AWS CodePipeline

AWS CodeBuild の仕組み



AWS CodeBuild の仕組み



AWS CodeBuild の仕組み

AWS CodeBuild の仕組み

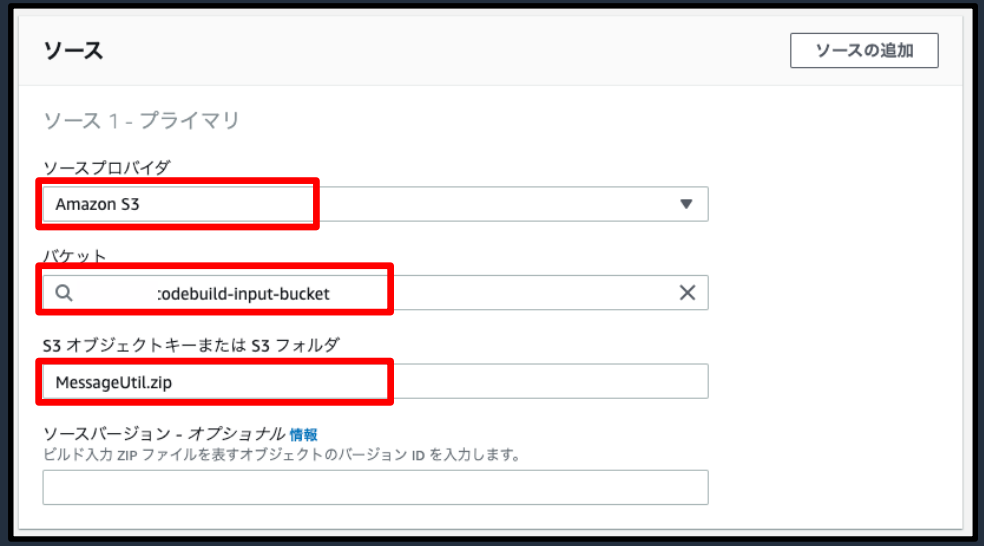
AWS CodeBuild プロジェクトの作成

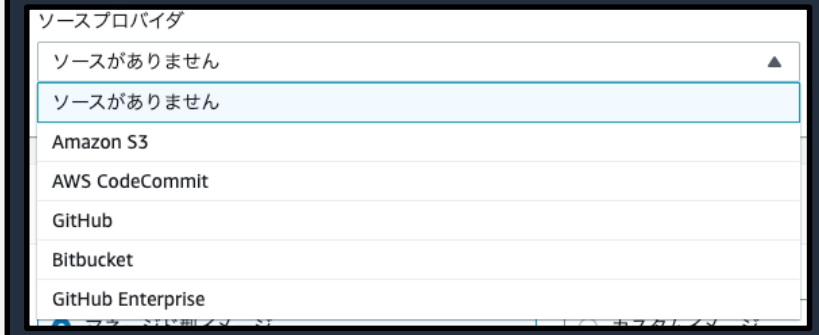
AWS CodeBuild プロジェクトの作成

プロジェクト名を入力

AWS CodeBuild プロジェクトの作成

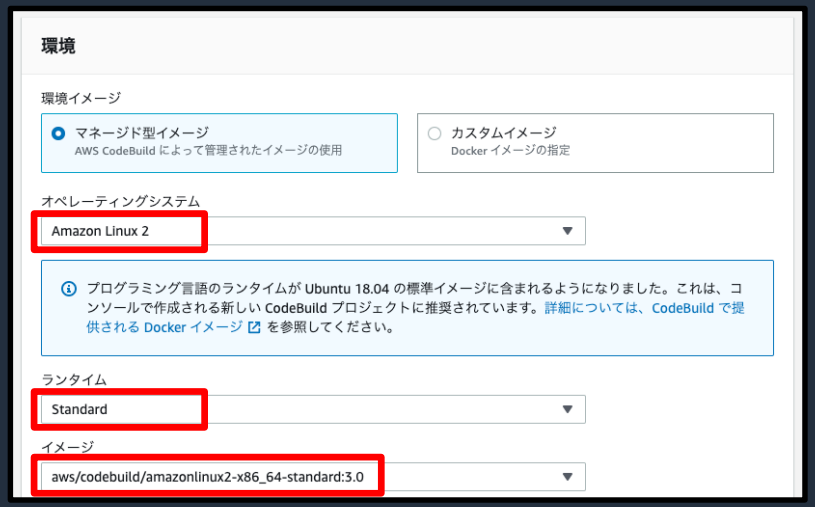
ソースとして Amazon S3 を選択し、
バケット名と オブジェクトキーを入力

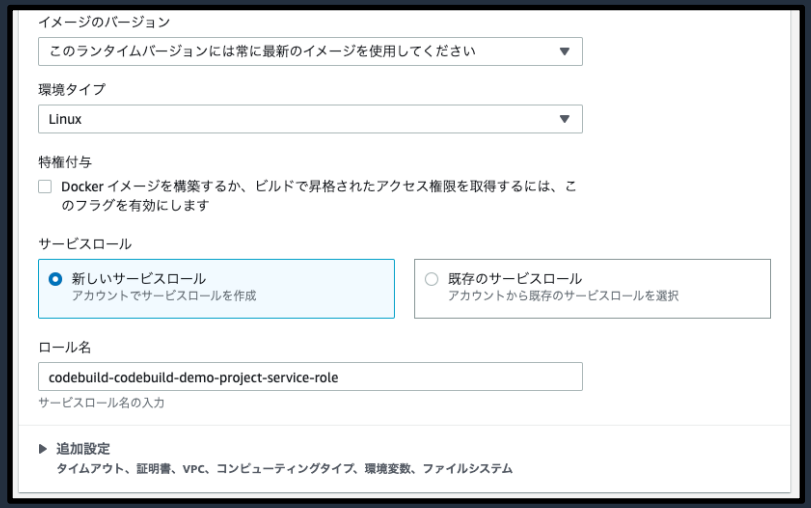




AWS CodeBuild プロジェクトの作成

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





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

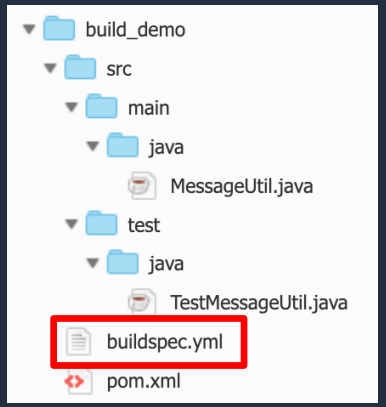
AWS CodeBuild プロジェクトの作成

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

AWS CodeBuild プロジェクトの作成

CloudWatch Logs へログを送信する

ソースの作成

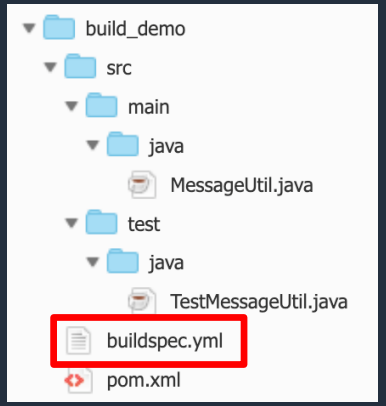


buildspec.yml

```yaml
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

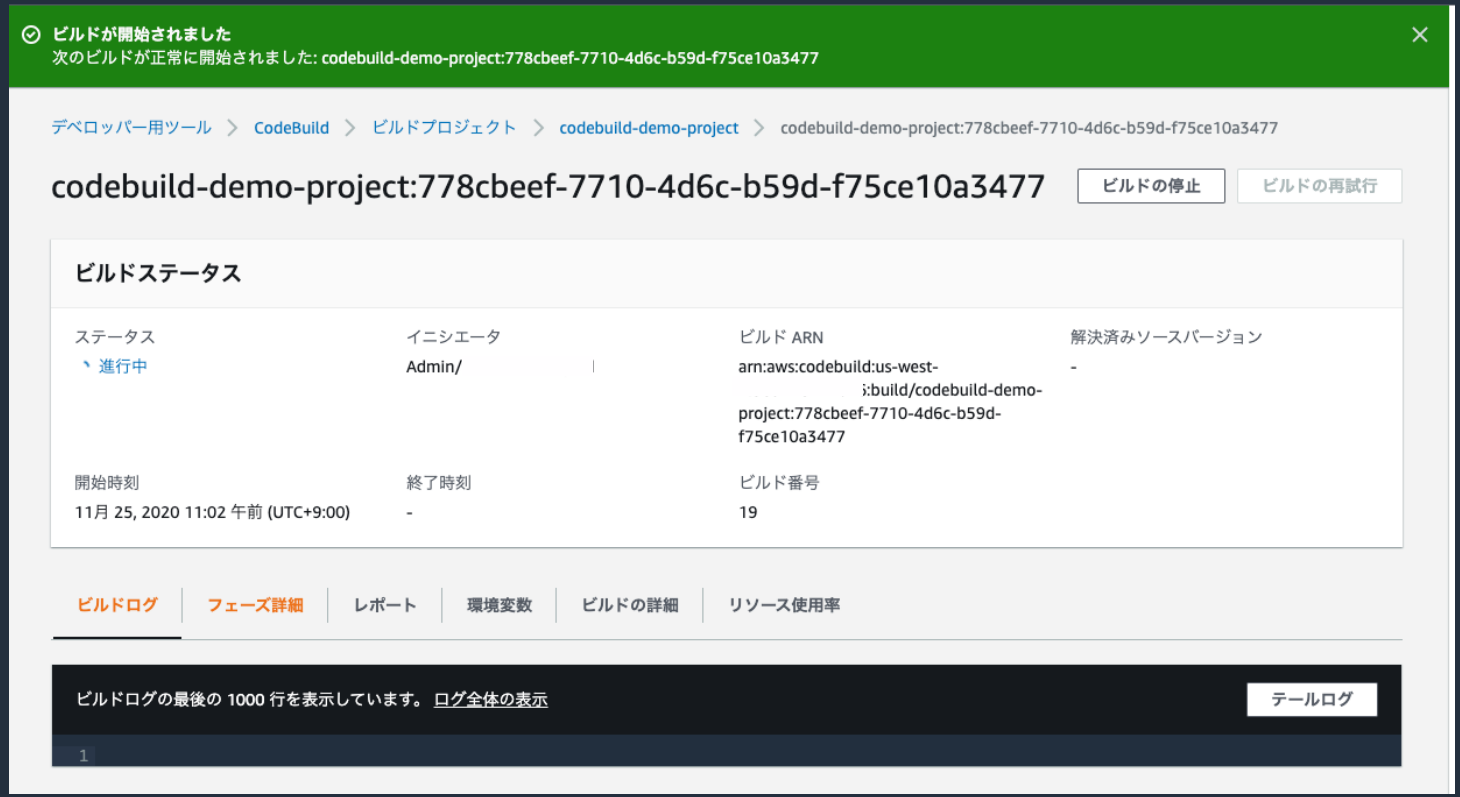
jar ファイルを作成

ビルド出力を定義

aws

ビルドの実行

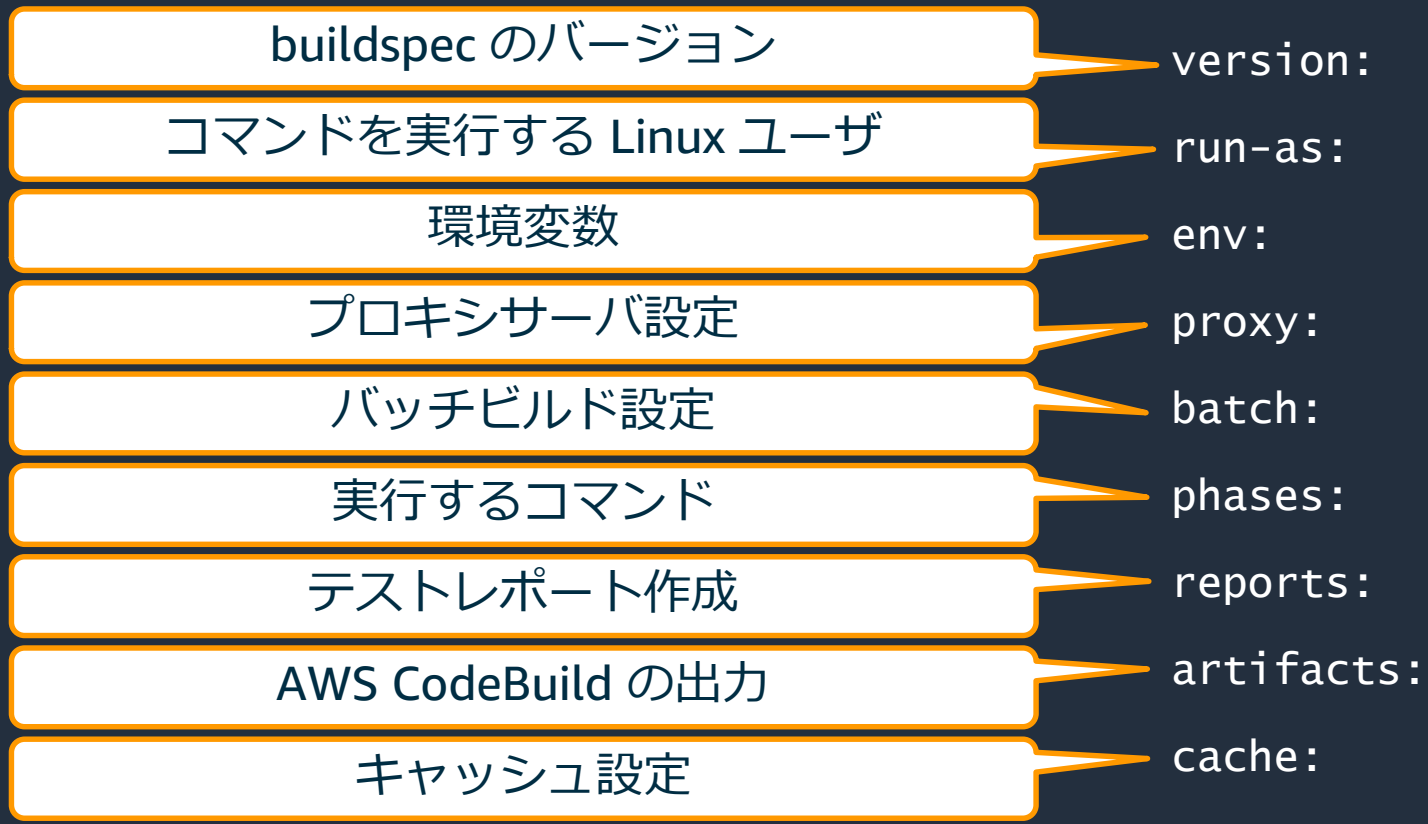
ビルドの実行



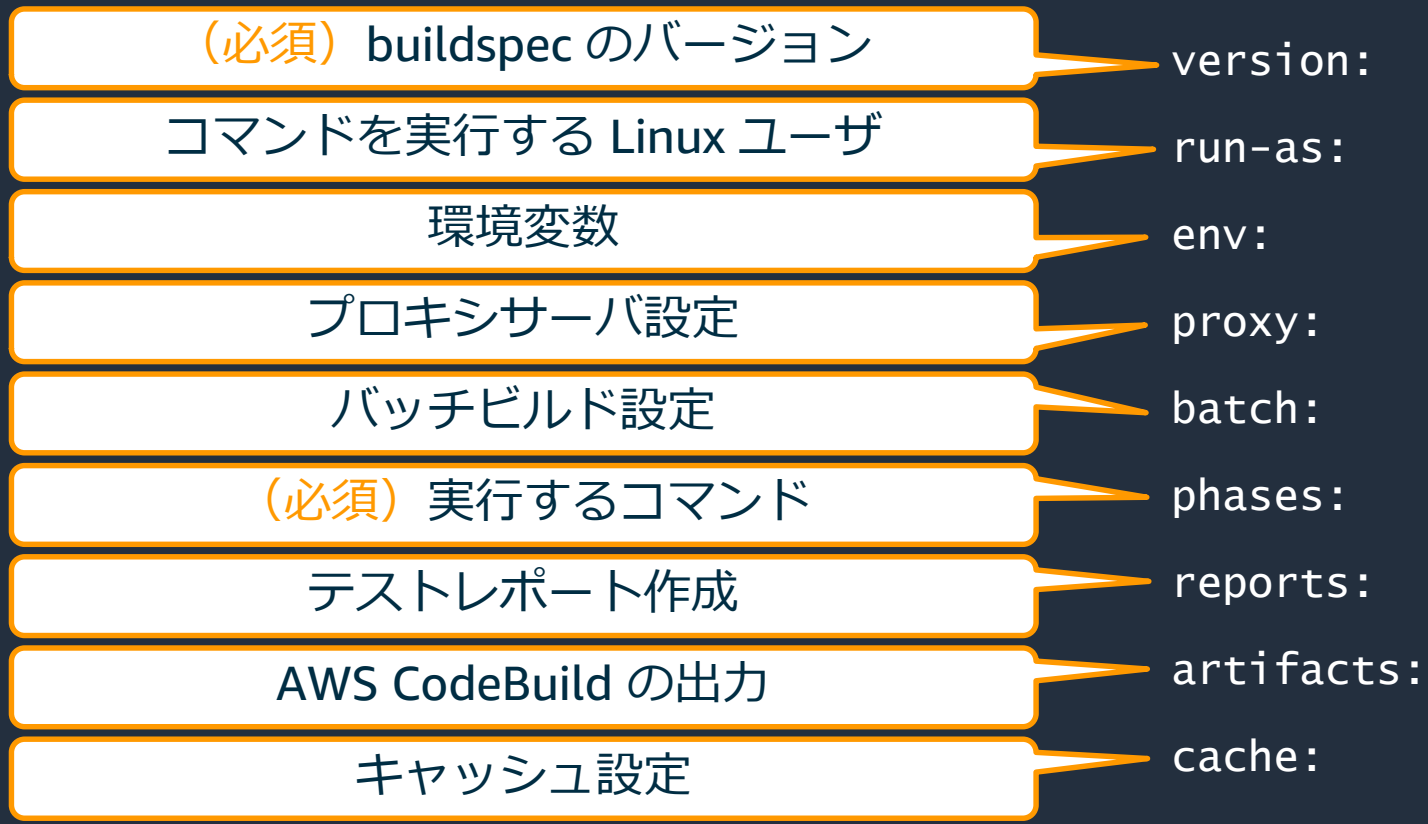
本日のアジェンダ

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

buidspec.yml の構成



buidspec.yml の構成



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

aws

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

aws

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

aws

batch

バッチビルド設定

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

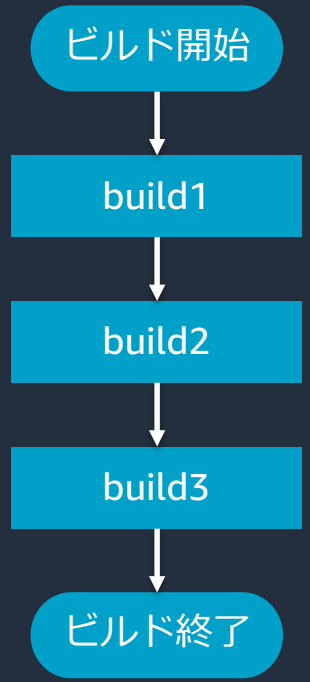
- プロジェクトの同時実行と協調実行の定義を行う
- バッチビルドには 3 つのタイプがある
    - build-graph
    - build-list
    - build-matrix

aws

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

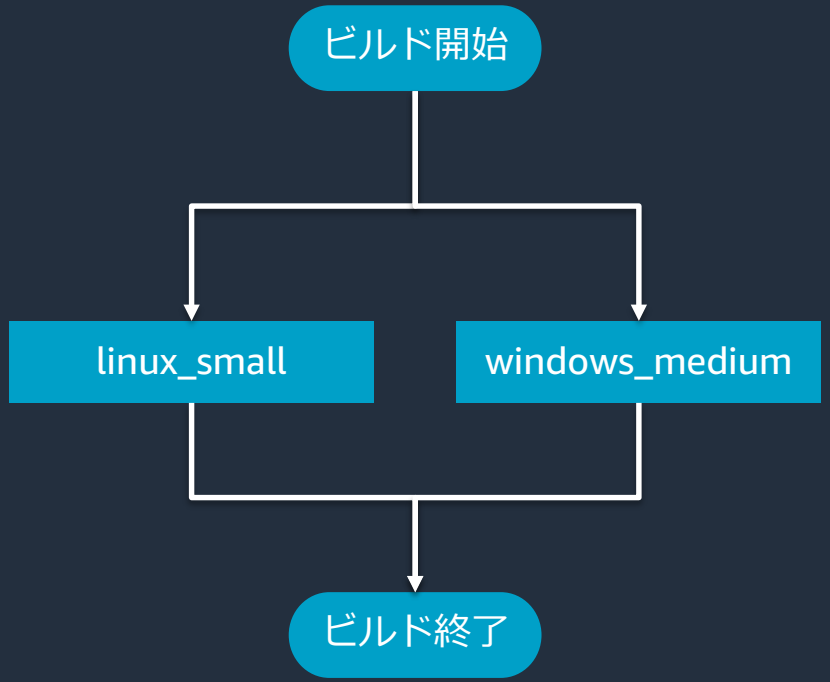
```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM
      depend-on:
        - build1
    - identifier: build3
      env:
        compute-type: BUILD_GENERAL1_LARGE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

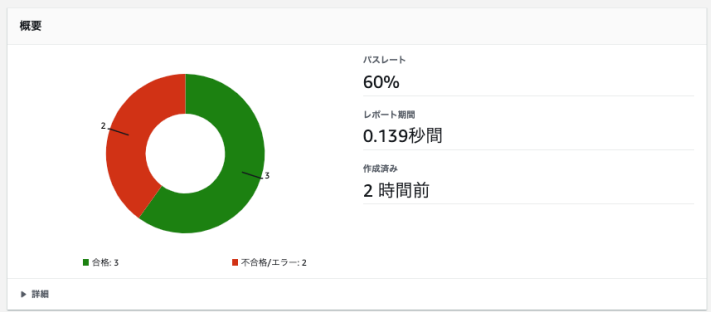
テストレポート (JunitXMLの 例)

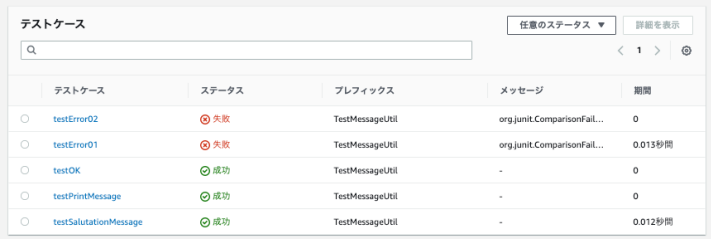
```
reports:
  junit-report:
    files:
      - 'target/surefire-reports/TEST-TestMessageUtil.xml'
    file-format: JUNITXML
```

- 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





reports

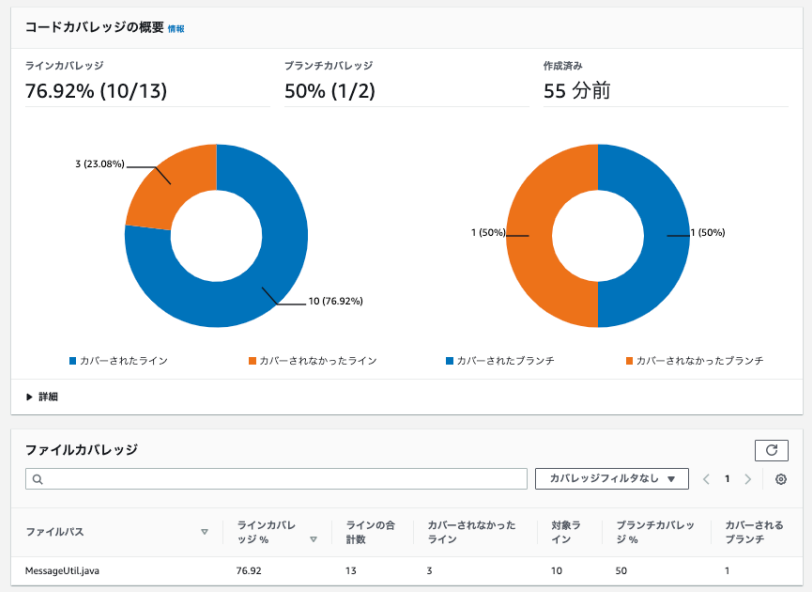
コードカバレッジレポート (JaCoCoXML の例)

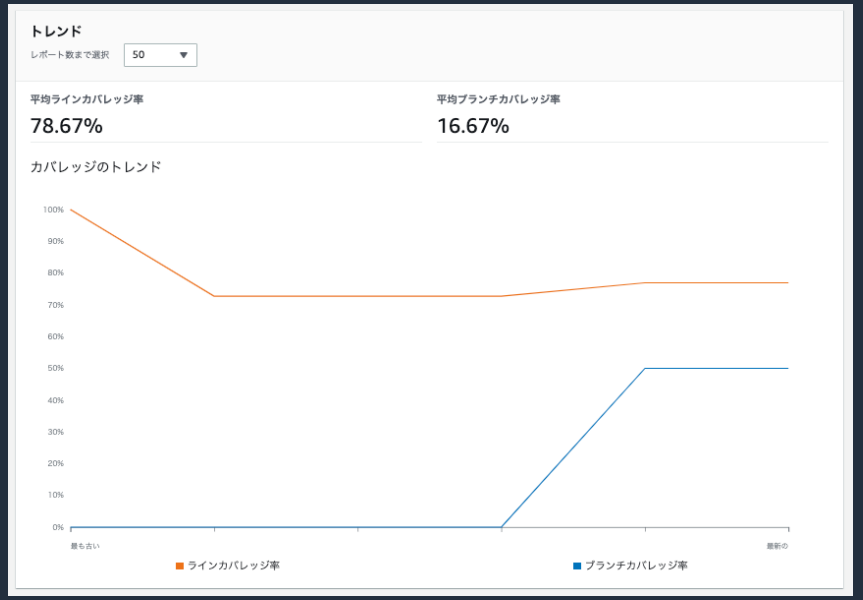
```
reports:
  jacoco-report:
    files:
      - 'target/site/jacoco/jacoco.xml'
    file-format: 'JACOCOXML'
```

- 以下のフォーマットに対応
    - Clover XML, Cobertula 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）
  - line coverage = (total lines covered)/(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）
  - branch coverage = (total branches covered)/(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
```

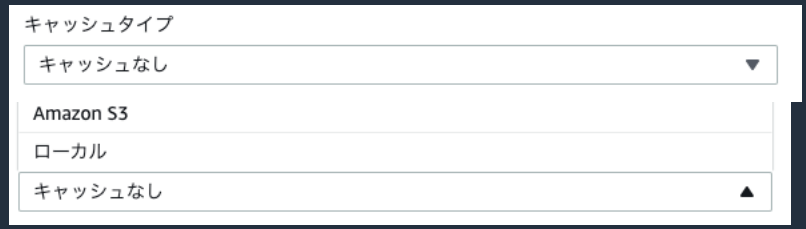
- アーティファクトの名、アーティファクトに含めるサブディレクトリとファイルを指定
- secondary-artifacts を利用して、複数のビルド出力アーティファクトを定義することも可能

aws

cache

キャッシュの設定

```
cache:
  paths:
    - path
    - path
```



- 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}
```

- 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}
```

- 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

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

AWS Session Manager でビルド環境へアクセスする

Session Manager でビルド環境へアクセスする

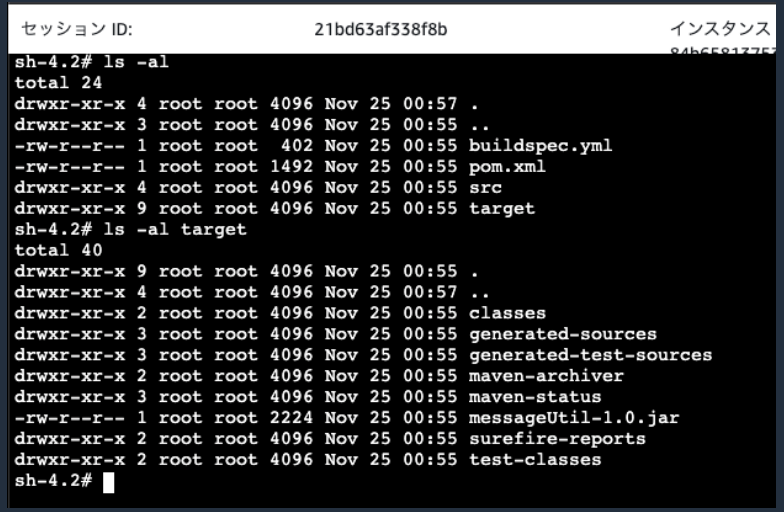
チェックを入れると必要な権限が
ロールに追加される

Session Manager でビルド環境へアクセスする

Session Manager でビルド環境へアクセスする

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

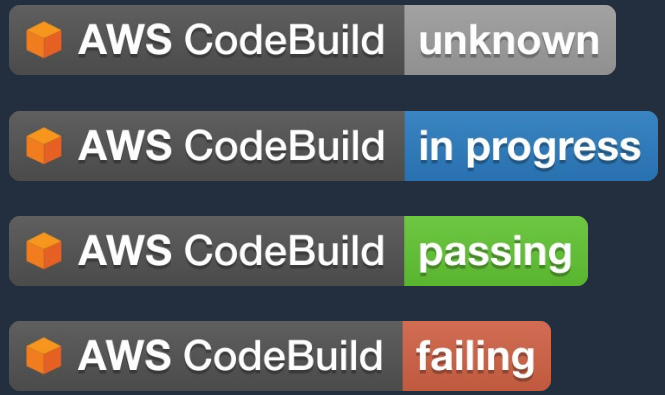


ビルドを再開する

```
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

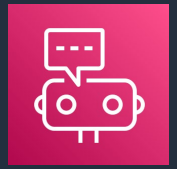


- ビルドプロジェクトに対して生成されるパブリック可能な URL で認証不要
- CodeCommit や GitHub のマークダウンファイルに埋め込み、ビルドステータスを表示することが可能

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

Amazon Simple Notification Service



AWS Chatbot

VPC 内リソースへのアクセス

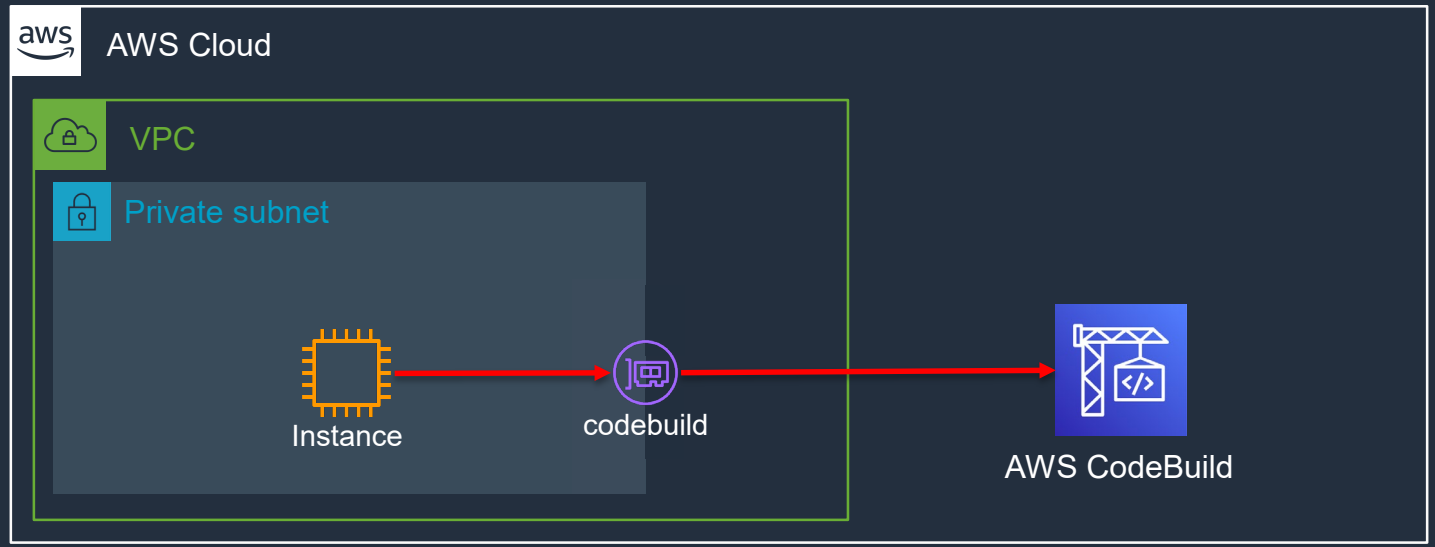
- プライベートサブネット内のリソースへアクセス
- VPCエンドポイント経由でのみアクセスできるように設定されたサービスへのアクセス
- 固定 IP アドレスを必要とする外部サービスへの通信に NAT ゲートウェイ, NAT インスタンスの EIP を使用する

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

0.005USD * 5 分 * 300 回 = 7.5USD

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

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

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]

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