



[AWS Black Belt Online Seminar]

AWS CodeDeploy

サービスカットシリーズ

Solutions Architect 松本 雅博
2021/1/26

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



過去資料
<https://amzn.to/JPArchive>



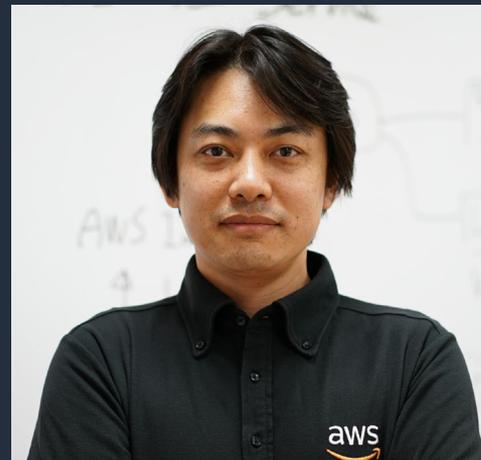
自己紹介

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



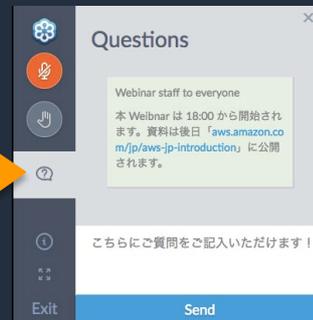
AWS Black Belt Online Seminar とは

「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2021年1月26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.

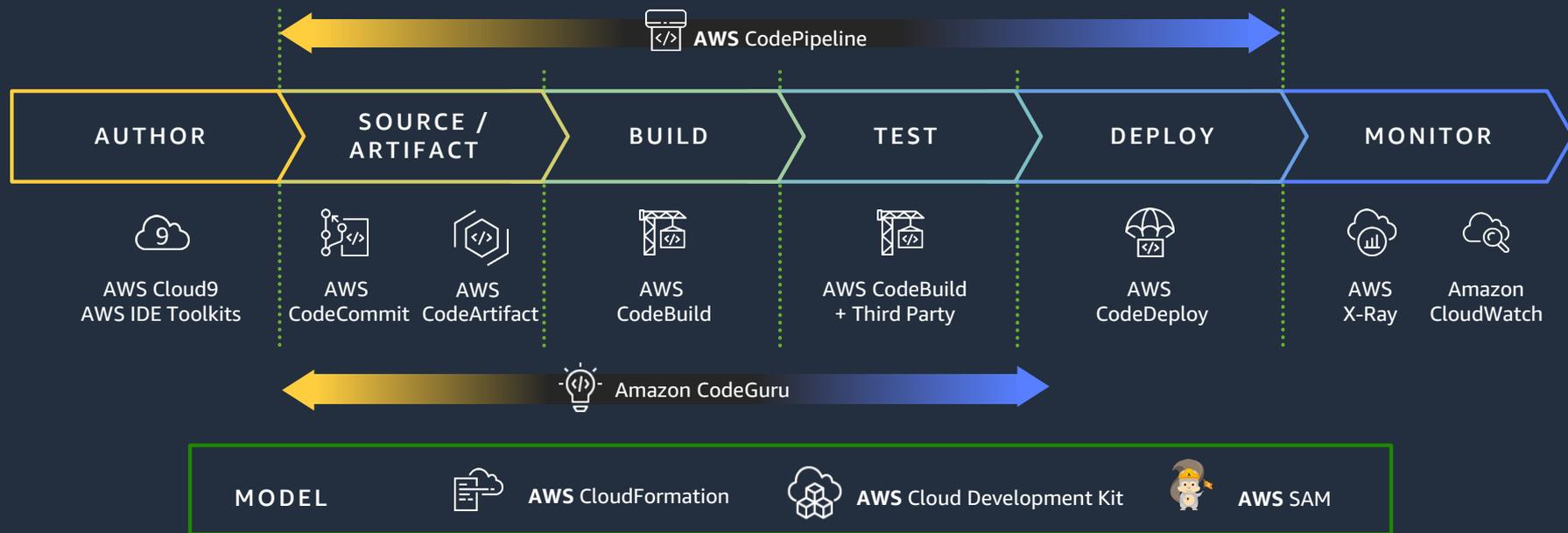
本セミナーの概要

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

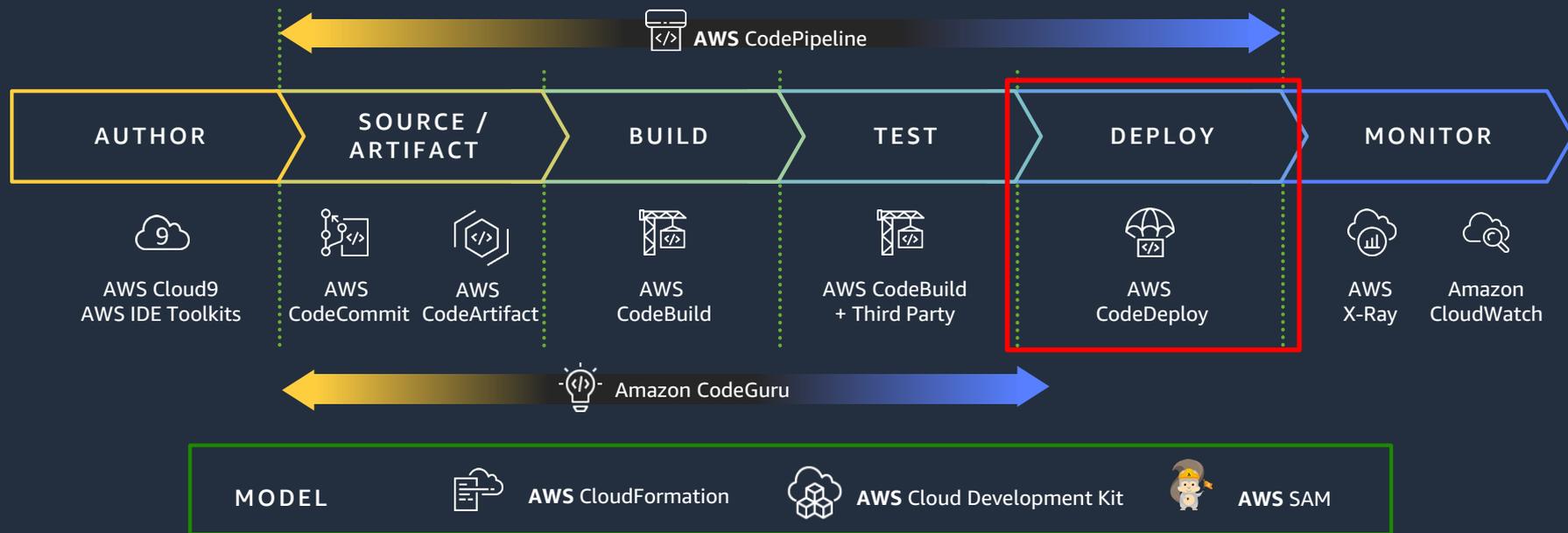
本日のアジェンダ

- AWS CodeDeploy 概要
- EC2 / オンプレミスへのデプロイ
- Lambda へのデプロイ
- ECS へのデプロイ
- 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



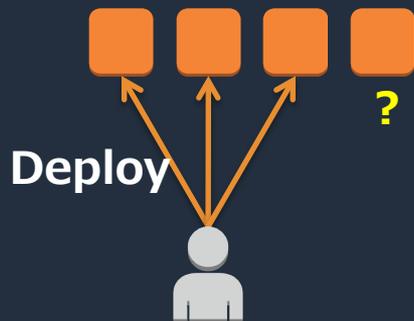
AWS CodeDeploy



- ソフトウェアのデプロイを自動化する、フルマネージド型のサービス
- Amazon EC2、AWS Lambda、オンプレミス サーバー、コンテナへの自動的なデプロイメント
- アプリケーションの複雑なアップデートの実施
- アプリケーションのデプロイ中のダウンタイムを回避
- エラーを検知すると自動的にロールバックを実行

Pull 型のデプロイ手段

× Push型



Deploy先サーバを知っている必要がある

○ Pull型



どんなサーバが立っているか意識する必要はない

自動化を実現するには、Pull 型のデプロイが推奨される。

AWS CodeDeploy の主要コンポーネント

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

アプリケーションとコンピューティングプラットフォーム

- アプリケーション
 - デプロイするアプリケーションを一意に識別する名前
- コンピューティングプラットフォーム
 - アプリケーションがデプロイされるプラットフォーム
 - EC2/オンプレミス
 - AWS Lambda
 - Amazon ECS

アプリケーションの設定

アプリケーション名
アプリケーション名の入力

BlackBeltDemo

(100 文字以内)

コンピューティングプラットフォーム
コンピューティングプラットフォームを選択する

EC2/オンプレミス

AWS Lambda

Amazon ECS

アプリケーションの作成

デプロイグループ

- デプロイ環境の定義
- Auto Scaling グループ
- タグのグループ
 - EC2 インスタンス、オンプレミスインスタンス
- ECS サービス

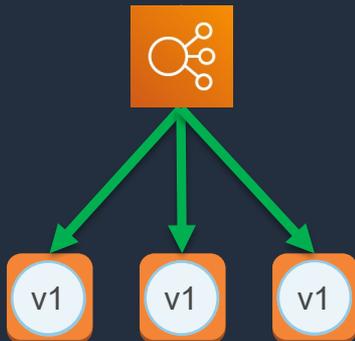
The screenshot displays the AWS IAM console interface for an application named 'BlackBeltEC2Demo'. The 'Application details' section shows the name and platform (EC2/On-premises). Below this, there are tabs for 'Deploy', 'Deployment Groups', and 'Versions', with 'Deployment Groups' currently selected. The 'Deployment Groups' section includes a search bar, a 'Details' button, a 'Cancel' button, and a 'Create Deployment Group' button. A table lists three deployment groups: 'stg', 'dev', and 'prod', each with a status of '-' and zero trigger counts.

	名前	ステータス	最後に試行したデプロイ	最後に成功したデプロイ	トリガー回数
<input type="radio"/>	stg	-	-	-	0
<input type="radio"/>	dev	-	-	-	0
<input type="radio"/>	prod	-	-	-	0

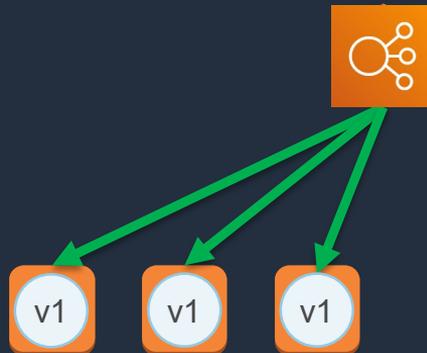
デプロイタイプ

v1 がデプロイされた状態

In-Place



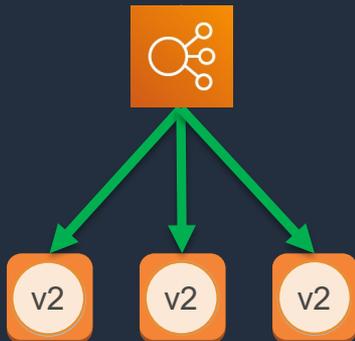
Blue/Green



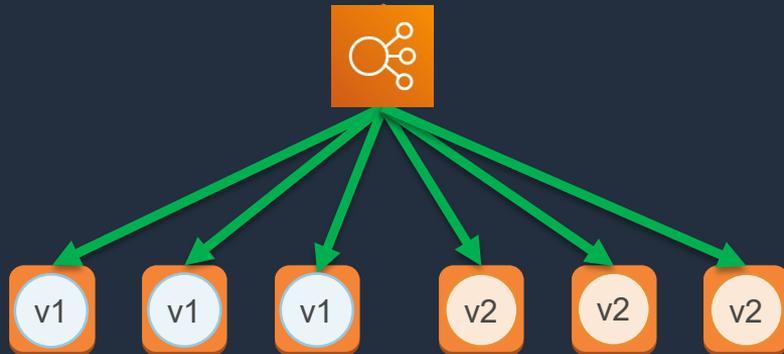
デプロイタイプ

v2 をデプロイ

In-Place



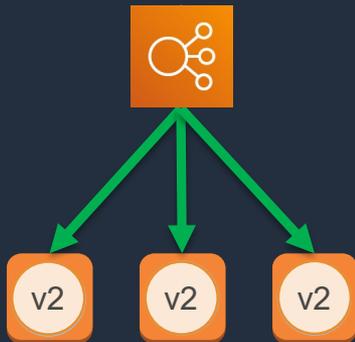
Blue/Green



デプロイタイプ

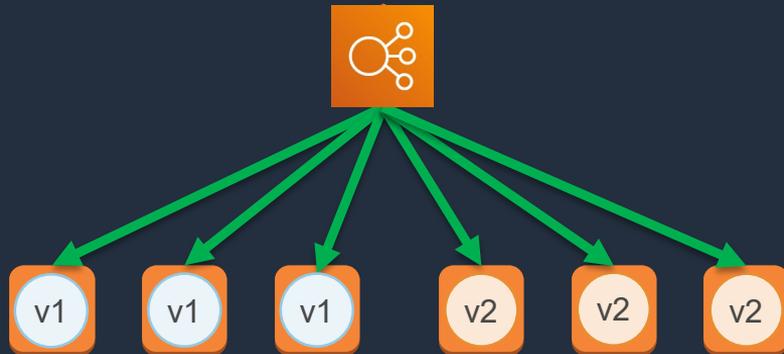
EC2 の場合、In-Place もしくは Blue/Green
Lambda, ECS の場合は Blue/Green

In-Place



既存ノードのアセットを更新
(1 台ずつ行う場合は
ローリングデプロイとも言う)

Blue/Green



新規にノードを構築し、デプロイ・テスト後に
リクエストの振り分け先を変更

デプロイ設定

- どのようにデプロイするかを定義したもの
- デプロイする割合やデプロイ成功、失敗の条件が異なる
- 独自のデプロイ設定を作成することも可能

The image displays a collection of AWS CodeDeploy deployment configuration cards. Each card represents a different deployment strategy and platform. The configurations are as follows:

- CodeDeployDefault.OneAtATime**: コンピューティングプラットフォーム EC2/オンプレミス。正常なホストの最小数値 1。
- CodeDeployDefault.HalfAtATime**: コンピューティングプラットフォーム EC2/オンプレミス。
- CodeDeployDefault.AllAtOnce**: コンピューティングプラットフォーム EC2/オンプレミス。
- CodeDeployDefault.LambdaAllAtOnce**: コンピューティングプラットフォーム AWS Lambda。設定タイプ リニア。1回にすべて。
- CodeDeployDefault.LambdaLinear10PercentEvery1Minute**: コンピューティングプラットフォーム AWS Lambda。設定タイプ リニア。ステップ 10%。間隔 1分。
- CodeDeployDefault.LambdaLinear10PercentEvery2Minutes**: コンピューティングプラットフォーム AWS Lambda。設定タイプ リニア。ステップ 10%。間隔 2分。
- CodeDeployDefault.LambdaLinear10PercentEvery3Minutes**: コンピューティングプラットフォーム AWS Lambda。設定タイプ リニア。ステップ 10%。間隔 3分。
- CodeDeployDefault.ECSAllAtOnce**: コンピューティングプラットフォーム Amazon ECS。設定タイプ リニア。1回にすべて。
- CodeDeployDefault.ECSLinear10PercentEvery1Minute**: コンピューティングプラットフォーム Amazon ECS。設定タイプ リニア。ステップ 10%。間隔 1分。
- CodeDeployDefault.ECSLinear10PercentEvery3Minutes**: コンピューティングプラットフォーム Amazon ECS。設定タイプ リニア。ステップ 10%。間隔 3分。
- CodeDeployDefault.ECSCanary10Percent5Minutes**: コンピューティングプラットフォーム Amazon ECS。設定タイプ Canary。ステップ 10%。間隔 5分。

デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

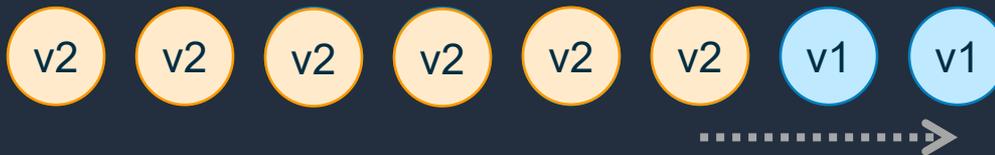
One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

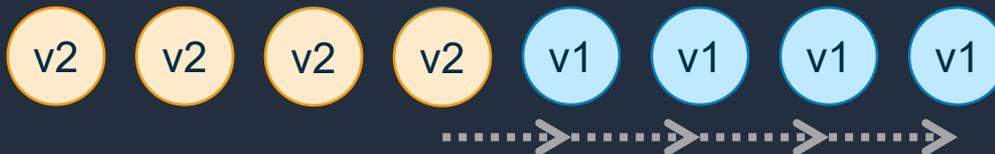
Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

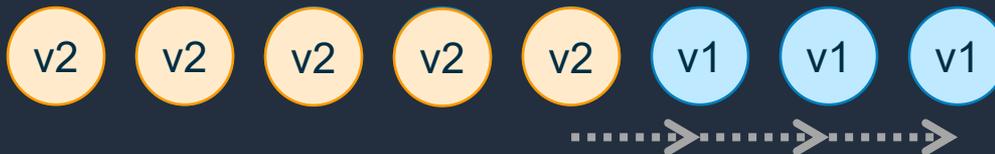
Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

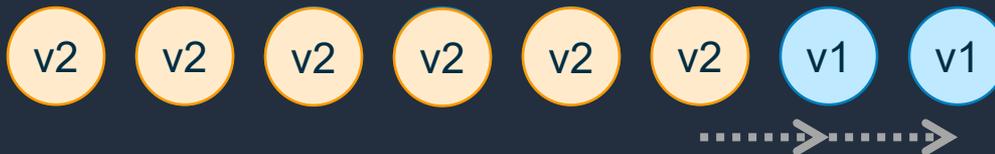
Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

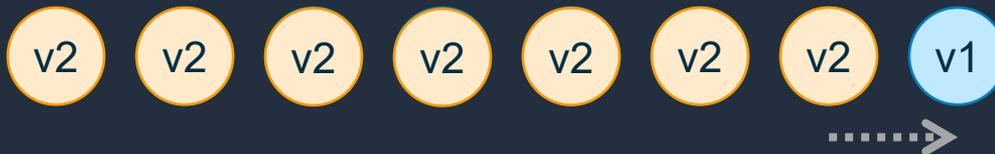
Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

Min. healthy hosts = 0



デプロイ設定 EC2/オンプレミス

One-at-a-time

Min. healthy hosts = 99%



[Custom]

Min. healthy hosts = 75%



Half-at-a-time

Min. healthy hosts = 50%



All-at-once

Min. healthy hosts = 0



デプロイ設定 Lambda / ECS

Linear

Step 25%, Interval 10min

v1

100%

v2

0%

Canary

Step 25%, Interval 10min

v1

100%

v2

0%

All-at-once

v1

100%

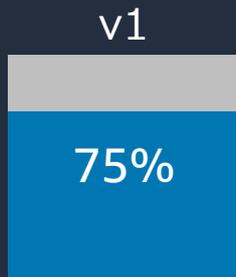
v2

0%

デプロイ設定 Lambda / ECS

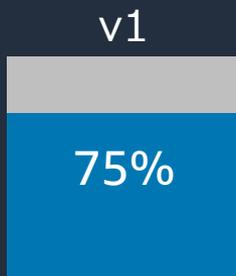
Linear

Step 25%, Interval 10min

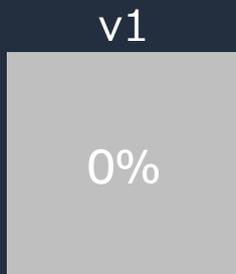


Canary

Step 25%, Interval 10min



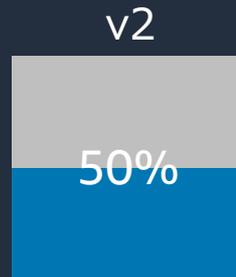
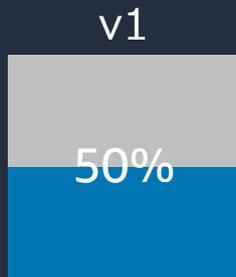
All-at-once



デプロイ設定 Lambda / ECS

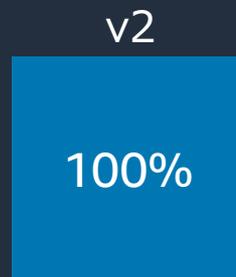
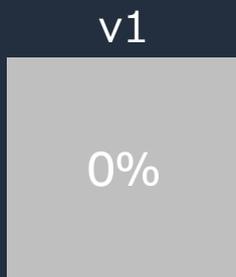
Linear

Step 25%, Interval 10min

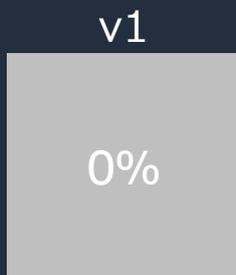


Canary

Step 25%, Interval 10min



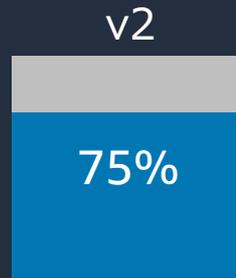
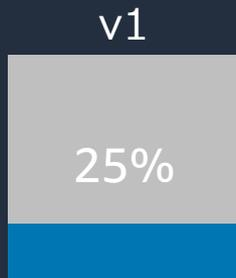
All-at-once



デプロイ設定 Lambda / ECS

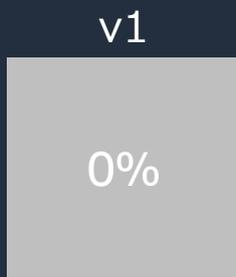
Linear

Step 25%, Interval 10min

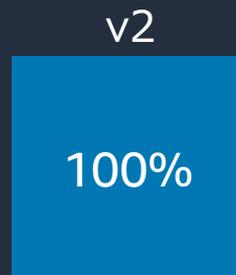
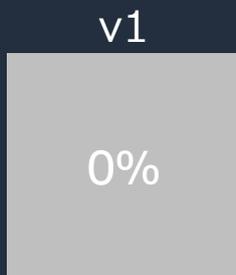


Canary

Step 25%, Interval 10min



All-at-once



デプロイ設定 Lambda / ECS

Linear

Step 25%, Interval 10min

v1

0%

v2

100%

Canary

Step 25%, Interval 10min

v1

0%

v2

100%

All-at-once

v1

0%

v2

100%

リビジョンとターゲットリビジョン

- リビジョン
 - EC2
 - ソースコード、Webページ、スクリプト等と AppSpec ファイルをまとめたアーカイブ
 - Lambda
 - Lambda デプロイ用の AppSpec ファイル
 - ECS
 - ECS デプロイ用の AppSpec ファイル
- ターゲットリビジョン
 - リポジトリにアップロードした直近のリビジョン
 - デプロイグループへデプロイする対象
 - 自動デプロイで取得されるリビジョン

サービスロール

- CodeDeploy に付与する IAM ロール
- CodeDeploy から AWS リソースを操作するために必要
- 管理ポリシー
 - AWSCodeDeployRole
 - AWSCodeDeployRoleForLambda
 - AWSCodeDeployRoleForLambdaLimited
 - AWSCodeDeployRoleForECS
 - AWSCodeDeployRoleForECSLimited

IAM インスタンスプロフィール

- EC2 インスタンスに付与する IAM ロール
- S3 から配布物を取得できるようにする

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

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

EC2 / オンプレミス へのデプロイ

CodeDeploy **EC2/オンプレミス** デプロイメント

- AWS CodeDeploy Agent が導入された、EC2インスタンス、オンプレミスインスタンスへデプロイ
- EC2 インスタンスには、**In-Place**、**Blue/Green** デプロイが可能
- オンプレミスインスタンスへは、**In-Place** デプロイのみ可能
- デプロイグループに Auto Scaling Group を指定することで、**スケールアウト時に最新のリビジョンが自動でデプロイされる**
- **ライフサイクルイベントへ Hook を指定し、スクリプトを実行可能**
- Hookが失敗した場合やAmazon CloudWatchアラームを検知した場合は**数秒で迅速にロールバック**

アプリケーションの作成

アプリケーションの作成

アプリケーションの設定

アプリケーション名
アプリケーション名の入力

BlackBeltEC2Demo

(100 文字以内)

コンピューティングプラットフォーム
コンピューティングプラットフォームを選択する

EC2/オンプレミス

キャンセル

アプリケーションの作成

デプロイグループの作成

BlackBeltEC2Demo

通知 ▼ アプリケーションの削除

Application details

名前	コンピューティングプラットフォーム
BlackBeltEC2Demo	EC2/オンプレミス

デプロイ | **デプロイグループ** | リビジョン

デプロイグループ

詳細を表示 編集 **デプロイグループの作成**

< 1 > ⚙️

名前	ステータス	最後に試行したデプロイ	最後に成功したデプロイ
<p>デプロイグループはありません</p> <p>CodeDeploy を使ってアプリケーションをデプロイする前に、デプロイグループを作成する必要があります。</p> デプロイグループの作成			

デプロイグループの作成

デプロイグループの作成

アプリケーション

アプリケーション

BlackBeltEC2Demo

コンピューティングタイプ

EC2/オンプレミス

デプロイグループ名

デプロイグループ名の入力

BlackBeltEC2DemoGrp

100文字以内

サービスロール

サービスロールの入力

ターゲットインスタンスに AWS CodeDeploy アクセスを付与する、CodeDeploy アクセス許可内のサービスロールを入力します。

arn:aws:iam:::role/BlackBeltDemo

デプロイグループの作成

デプロイタイプ

アプリケーションのデプロイ方法を選択する

インプレース

デプロイグループのインスタンスを最新のアプリケーションリビジョンで更新します。デプロイ中、各インスタンスは更新のために一時的にオフラインになります

Blue/Green

デプロイグループのインスタンスを新しいインスタンスに置き換え、新しいインスタンスに最新のアプリケーションリビジョンをデプロイします。置き換え先環境のインスタンスが Load balancer に登録されると、置き換え先環境のインスタンスは登録解除され、終了可能になります。

環境設定

このデプロイに追加する Amazon EC2 Auto Scaling グループ、Amazon EC2 インスタンス、およびオンプレミスインスタンスの任意の組み合わせを選択します。

Amazon EC2 Auto Scaling グループ

1 一意に一致したインスタンス。詳細については、[ここをクリックしてください](#)

アプリケーションリビジョンをデプロイする先として最大 10 個までの Amazon EC2 Auto Scaling グループを選択できます。

BlackBeltDemo X

Amazon EC2 インスタンス

オンプレミスインスタンス

一致するインスタンス

1 一意に一致したインスタンス。詳細については、[ここをクリックしてください](#)

環境設定

このデプロイに追加する Amazon EC2 Auto Scaling グループ、Amazon EC2 インスタンス、およびオンプレミスインスタンスの任意の組み合わせを選択します。

Amazon EC2 Auto Scaling グループ

Amazon EC2 インスタンス

0 一意に一致したインスタンス。詳細については、[ここをクリックしてください](#)

このデプロイグループには EC2 インスタンス用に最大 3 個のタググループを追加できます。
1 つのタググループ: タググループによって識別される任意のインスタンスのデプロイ先。
複数のタググループ: すべてのタググループによって識別されるインスタンスのみのデプロイ先。

タググループ 1

キー

値 - オプション

Q Name X

Q WebServer X

タグの削除

タグの追加

+ タググループの追加

オンプレミスインスタンス

環境設定

このデプロイに追加する Amazon EC2 Auto Scaling グループ、Amazon EC2 インスタンス、およびオンプレミスインスタンスの任意の組み合わせを選択します。

Amazon EC2 Auto Scaling グループ

Amazon EC2 インスタンス

オンプレミスインスタンス

このデプロイグループには EC2 インスタンス用に最大 3 個のタググループを追加できます。
1 つのタググループ: タググループによって識別される任意のインスタンスのデプロイ先。
複数のタググループ: すべてのタググループによって識別されるインスタンスのみのデプロイ先。

タググループ 1

キー

値 - オプション

Name

WebServer

タグの削除

タグの追加

+ タググループの追加

デプロイグループの作成

デプロイタイプ

アプリケーションのデプロイ方法を選択する

- インプレース
デプロイグループのインスタンスを最新のアプリケーションリビジョンで更新します。デプロイ中、各インスタンスは更新のために一時的にオフラインになります

- Blue/Green
デプロイグループのインスタンスを新しいインスタンスに置き換え、新しいインスタンスに最新のアプリケーションリビジョンをデプロイします。置き換え先環境のインスタンスが Load balancer に登録されると、置き換え先環境のインスタンスは登録解除され、終了可能になります。

環境設定

現在のアプリケーションリビジョンをデプロイする Amazon EC2 Auto Scaling グループまたは Amazon EC2 インスタンスを指定します。

- Amazon EC2 Auto Scaling グループの自動コピー
Amazon EC2 Auto Scaling グループをプロビジョンし、これに対して新しいアプリケーションリビジョンをデプロイします。AWS CodeDeploy は、ここで指定したグループをコピーして Auto Scaling グループを作成します。

- インスタンスの手動プロビジョン
ここでは、現在のアプリケーションリビジョンが実行されているインスタンスを指定します。デプロイを作成するときに、置き換え先環境のインスタンスを指定します。

現在のアプリケーションリビジョンがデプロイされている Amazon EC2 Auto Scaling グループを選択します。

Q BlackBeltDemo



環境設定

現在のアプリケーションリビジョンをデプロイする Amazon EC2 Auto Scaling グループまたは Amazon EC2 インスタンスを指定します。

- Amazon EC2 Auto Scaling グループの自動コピー
Amazon EC2 Auto Scaling グループをプロビジョンし、これに対して新しいアプリケーションリビジョンをデプロイします。AWS CodeDeploy は、ここで指定したグループをコピーして Auto Scaling グループを作成します。

- インスタンスの手動プロビジョン
ここでは、現在のアプリケーションリビジョンが実行されているインスタンスを指定します。デプロイを作成するときに、置き換え先環境のインスタンスを指定します。

このデプロイに追加する Amazon EC2 Auto Scaling グループ、Amazon EC2 インスタンス、およびオンプレミスインスタンスの任意の組み合わせを選択します。

Amazon EC2 Auto Scaling グループ

Amazon EC2 インスタンス

デプロイグループの作成

AWS Systems Manager を使用したエージェント設定 情報

AWS Systems Manager は CodeDeploy エージェントをすべてのインスタンスにインストールし、設定した頻度で更新します。



AWS Systems Manager が CodeDeploy エージェントをインストールする前に、必須の前提条件を完了します。

AWS Systems Manager エージェントがすべてのインスタンスにインストールされていることを確認し、必須の IAM ポリシーをアタッチします。 [詳細はこちら](#)

AWS CodeDeploy エージェントのインストール

- なし
- 1 回のみ
- 今すぐ更新し、更新をスケジュール

ベーシックスケジューラ

Cron 式

14

日間



デプロイグループの作成

デプロイ設定

トラフィックの再ルーティング

すぐにトラフィックを再ルーティング

トラフィックを再ルーティングするかどうかを選択します

日間 時間 分

0 0 0

デプロイの成功後に置き換え元環境のインスタンスを削除するかどうかを選択し、削除するまでの待機時間を指定します。

デプロイグループの置き換え元インスタンスを終了

デプロイグループの置き換え元インスタンスを引き続き実行

デプロイ設定

デフォルトおよびカスタムデプロイ設定のリストから選択します。デプロイ設定には、アプリケーションのデプロイの速度やデプロイの成功または失敗の条件などのルールが定義されています。

CodeDeployDefault.AllAtOnce または

Load balancer

デプロイプロセス中に着信トラフィックを管理する Load balancer を選択します。ロードバランサーはデプロイ中に各インスタンスからのトラフィックをブロックし、デプロイが成功した後でトラフィックを許可します。

ロードバランシングを有効にする

Application Load Balancer または Network Load Balancer

Classic Load Balancer

ターゲットグループの選択

tg-BlackB-blackbelt-demo-1

▶ 詳細 - オプション

キャンセル

デプロイの作成

BlackBeltEC2DemoGrp

[編集](#)[削除](#)[デプロイの作成](#)

デプロイグループの詳細

デプロイグループ名
BlackBeltEC2DemoGrp

アプリケーション名
BlackBeltEC2Demo

コンピューティングプラットフォーム
EC2/オンプレミス

デプロイタイプ
インブレース

サービスロール ARN

arn:aws:iam:
o

'BlackBeltDem

デプロイ設定

CodeDeployDefault.AllAtOnce

ロールバックが有効になりました

-

エージェント更新スケジューラ

[AWS Systems Manager](#) での更新のスケジュール
の詳細 [🔗](#)

デプロイの作成

Create deployment

デプロイ設定

アプリケーション

BlackBeltEC2Demo

デプロイグループ

BlackBeltEC2DemoGrp

コンピューティングプラットフォーム

EC2/オンプレミス

デプロイタイプ

インプレース

リビジョンタイプ

アプリケーションは Amazon S3 に格納されています

アプリケーションは GitHub に格納されています

リビジョンの場所

リビジョンが保存されている Amazon S3 バケットをコピーして貼り付けます

s3:// essApp.zip

s3://bucket-name/folder/object.[zip|tar|tgz]

リビジョンファイルの種類

.zip

デプロイの説明

デプロイの説明: オプション

デプロイについての簡単な説明を追加します

デプロイの説明

デプロイの説明: オプション

デプロイについての簡単な説明を追加します

追加のデプロイ動作設定

ApplicationStop ライフサイクルイベントの障害 - オプション

デプロイグループの名前の入力

このインスタンス上のライフサイクルイベントの障害で、デプロイを失敗させない

コンテンツオプション - オptional

デプロイ時に、ターゲットインスタンスのファイルとアプリケーションリビジョンのファイルが同じ名前である場合、対処方法を選択します

デプロイの失敗

エラーが報告され、デプロイのステータスが [Failed] に変更されます。

コンテンツの上書き

アプリケーションリビジョンのファイルは、インスタンスのターゲットの場所にコピーされ、以前のファイルと置き換えられます。

コンテンツの保持

アプリケーションリビジョンのファイルは、インスタンスにコピーされません。既存のファイルはターゲットの場所に保持され、新しいデプロイの一部として扱われます。

▶ デプロイグループのオーバーライド

▶ ロールバック設定の上書き

キャンセル

デプロイの作成

デプロイ実行

d-K66GA2048

🔄 コードのデプロイ デプロイの再試行

デプロイのステータス

インスタンスへのアプリケーションのインストール

100%

1個のインスタンスの内、1個が更新されました 🟢 成功

デプロイの詳細

アプリケーション WordPress_App	デプロイ ID d-K66GA2048	ステータス 🟢 成功
デプロイ設定 CodeDeployDefault.OneAtATime	デプロイグループ WordPress_DepGroup	開始: ユーザーアクション
デプロイの説明 -		

リビジョンの詳細

リビジョンの場所 s3://	リビジョン作成 2 日前	リビジョンの説明 Application revision registered by Deployment ID: d-VXEIX0048
-----------------------------------	-----------------	---

デプロイのライフサイクルイベント

< 1 > ⚙️

インスタンス ID	期間	ステータス	最新のイベント	イベント	開始時刻	終了時刻
i-08066b0f709dff52b	10秒間	🟢 成功	ValidateService	View events	1月 23, 2021 4:49 午後 (UTC+9:00)	1月 23, 2021 4:49 午後 (UTC+9:00)

リビジョンの構成

- フォルダ構成
 - appspec.yml (必須)
 - ビルド済みの成果物
 - その他の配布物
 - Hook スクリプト
- アップロード先
 - Amazon S3
 - GitHub のリポジトリ

```
/
├─ appspec.yml
├─ config/
│  └─ config.xml
├─ deploy_hooks/
│  └─ start-tomcat.sh
│  └─ stop-tomcat.sh
└─ target/
   └─ hello.war
```

CodeDeploy EC2/オンプレミス appspec

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: "*.html"
    owner: root
    group: root
    mode: 755
hooks:
  BeforeInstall:
    - location: Scripts/deregister_from_elb.sh
  AfterInstall:
    - location: Scripts/install_dependencies.sh
  ApplicationStart:
    - location: Scripts/scripts/start_httpd.sh
  ValidateService:
    - location: Scripts/scripts/register_with_elb.sh
```

アプリケーションや設定ファイルをディレクトリへコピー

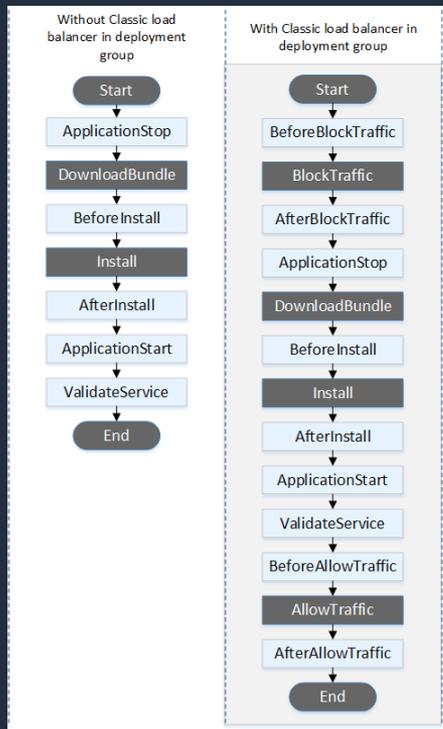
特定のディレクトリとファイルのアクセス許可を設定

Hook用スクリプトの指定

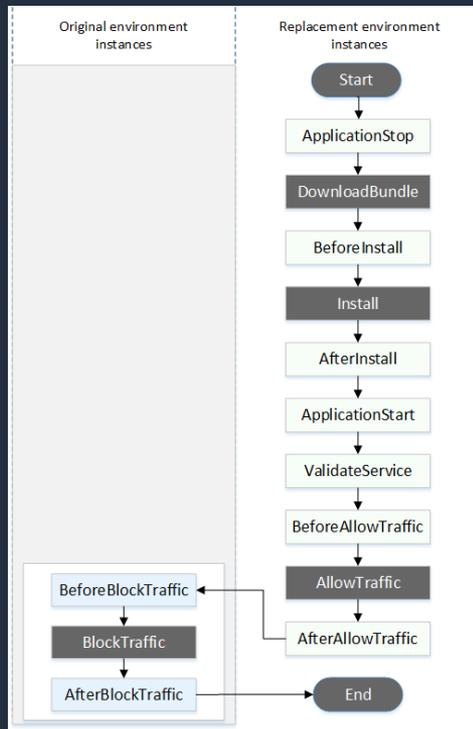
- インストール前後の処理
- アプリケーションの開始
- デプロイ成功の確認

デプロイライフサイクルイベント

In-Place



Blue/Green



Hook スクリプト 実行可能箇所

- ApplicationStop
- BeforeInstall
- AfterInstall
- ApplicationStart
- ValidateService
- BeforeAllowTraffic
- AfterAllowTraffic
- BeforeBlockTraffic
- AfterBlockTraffic

AWS CodeDeploy Agent

- デプロイ対象となるインスタンスで実行
- サポートする Amazon EC2 AMI OS
 - Amazon Linux 2018.03.x、2017.03.x、2016.09.x、2016.03.x、2014.09.x
 - Amazon Linux 2 (ARM、x86)
 - Ubuntu Server 20.04 LTS、19.10、18.04 LTS、16.04 LTS、14.04 LTS
 - Microsoft Windows Server 2019、2016、2012 R2、および 2008 R2
 - Red Hat Enterprise Linux (RHEL) 7.x

AWS CodeDeploy Agent

- サポートされているオンプレミス OS
 - Ubuntu Server 20.04 LTS、19.10、18.04 LTS、16.04 LTS、14.04 LTS
 - Microsoft Windows Server 2019、2016、2012 R2、および 2008 R2
 - Red Hat Enterprise Linux (RHEL) 7.x
- オープンソースで利用可能なので、他の環境にも適用可能
 - <https://github.com/aws/aws-codedeploy-agent>

AWS Lambda のデプロイ

CodeDeploy **Lambda** デプロイメント

- AWS Lambda の**関数重み付けエイリアス**を利用したトラフィックのシフト
- **カナリアデプロイ** (10分間 10% のトラフィックをシフト、その後残り全部もシフト) や**リニアデプロイ**(毎10分ごとに10%ずつシフト)を選択可能
- **Validation Hook**は各ステージへのデプロイ時のテストを有効化
- Hookが失敗した場合やAmazon CloudWatchアラームを検知した場合は数秒で**迅速にロールバック**

アプリケーションの作成

アプリケーションの作成

アプリケーションの設定

アプリケーション名
アプリケーション名の入力

BlackBeltLambdaDemo

(100 文字以内)

コンピューティングプラットフォーム
コンピューティングプラットフォームを選択する

AWS Lambda

キャンセル

アプリケーションの作成

デプロイグループの作成

BlackBeltLambdaDemo

通知 アプリケーションの削除

Application details

名前	コンピューティングプラットフォーム
BlackBeltLambdaDemo	AWS Lambda

デプロイ | デプロイグループ | リビジョン

デプロイグループ

詳細を表示 編集 デプロイグループの作成< 1 > 設定

名前	ステータス	最後に試行したデプロイ	最後に成功したデプロイ
<p>デプロイグループはありません</p> <p>CodeDeploy を使ってアプリケーションをデプロイする前に、デプロイグループを作成する必要があります。</p> デプロイグループの作成			

デプロイグループの作成

デプロイグループの作成

アプリケーション

アプリケーション
BlackBeltLambdaDemo
コンピューティングタイプ
AWS Lambda

デプロイグループ名

デプロイグループ名の入力

100 文字以内

サービスロール

サービスロールの入力
ターゲットインスタンスに AWS CodeDeploy アクセスを付与する、CodeDeploy アクセス許可内のサービスロールを入力します。

デプロイ設定

デプロイ設定
デフォルトおよびカスタムデプロイ設定のリストから選択します。デプロイ設定には、アプリケーションのデプロイの速度やデプロイの成功または失敗の条件などのルールが定義されています。

CodeDeployDefault.LambdaCanary10Percent10Minutes ▼ または

▶ 詳細 - オプション

キャンセル

デプロイの作成

BlackBeltLambdaDemoGrp

[編集](#)[削除](#)[デプロイの作成](#)

デプロイグループの詳細

デプロイグループ名

BlackBeltLambdaDemoGrp

アプリケーション名

[BlackBeltLambdaDemo](#)

コンピューティングプラットフォーム

AWS Lambda

デプロイタイプ

Blue/Green

サービスロール ARN

[arn:aws:iam::755260000000:role/BlackBeltDemo](#)

デプロイ設定

[CodeDeployDefault.LambdaCanary10Percent10Minutes](#)

ロールバックが有効になりました

-

エージェント更新スケジューラ

[AWS Systems Manager での更新のスケジュールの詳細](#)

デプロイの作成

Create deployment

デプロイ設定

アプリケーション

BlackBeltLambdaDemo

デプロイグループ

Q BlackBeltLambdaDemoGrp X

コンピューティングプラットフォーム

AWS Lambda

デプロイタイプ

Blue/Green

リビジョンタイプ

アプリケーションは Amazon S3 に格納されていま
す

AppSpec エディタの使用

AppSpec 言語

JSON

YAML

```
1 version: 0.0
2 Resources:
3   - myLambdaFunction:
4     Type: AWS::Lambda::Function
5     Properties:
6       Name: "BlackBeltDemoLambda"
7       Alias: "BlackBeltDemoAlias"
8       CurrentVersion: "1"
9       TargetVersion: "2"
```

テキストファイルとして保存

デプロイの説明

デプロイの説明: オプション

デプロイについての簡単な説明を追加します

▶ デプロイグループのオーバーライド

▶ ロールバック設定の上書き

キャンセル

デプロイの作成

デプロイ実行中

デベロッパー用ツール > CodeDeploy > デプロイ > d-YGKPRJI58

d-YGKPRJI58



デプロイを停止

デプロイを停止してロールバック

デプロイのステータス

ステップ 1

デプロイ前の検証



完了済み 成功

ステップ 2

トラフィックの移行中



10% 完了 進行中

ステップ 3

デプロイ後の検証



未開始

トラフィック移行の進行状況

次: デプロイでは、デプロイの開始から約 10分後に、トラフィックの 90% を現在のバージョンから置き換えバージョンに移行します。

オリジナル



デプロイの結果情報

トラフィックの 90%

置換



トラフィックの 10%

デプロイの詳細

アプリケーション

BlackBeltLambdaDemo

デプロイ ID

d-YGKPRJI58

ステータス

進行中

デプロイ設定

CodeDeployDefault.LambdaCanary10Percent10Minutes

デプロイグループ

BlackBeltLambdaDemoGrp

開始:

ユーザーアクション

デプロイの説明

-

CodeDeploy **Lambda** デプロイメント (appspec)

AppSpec File の例

```
version: 0.0
```

```
Resources:
```

```
- myLambdaFunction:
```

```
  Type: AWS::Lambda::Function
```

```
  Properties:
```

```
    Name: "myLambdaFunction"
```

```
    Alias: "myLambdaFunctionAlias"
```

```
    CurrentVersion: "1"
```

```
    TargetVersion: "2"
```

デプロイするLambdaの指定

```
Hooks:
```

```
- BeforeAllowTraffic: "BeforeTrafficShift"
```

```
- AfterAllowTraffic: "AfterTrafficShift"
```

Hook用Lambda関数の指定

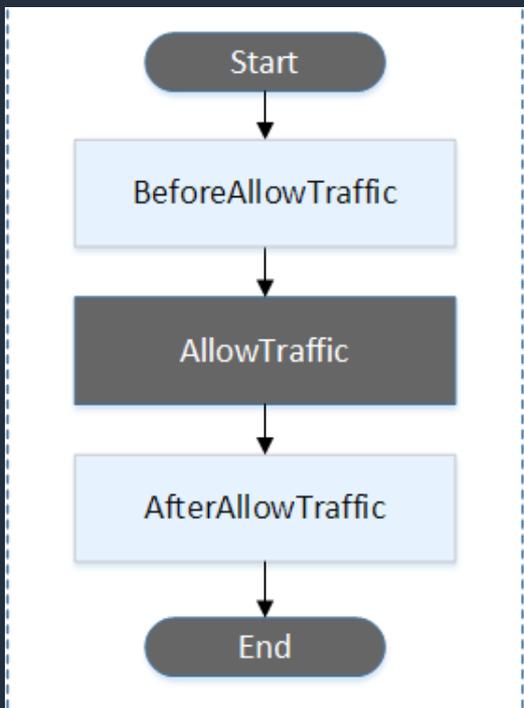
CodeDeploy **Lambda** デプロイメント (SAM)

サーバーレスアプリケーションのテンプレート

Resources:

GetFunction:	
{ Type: AWS::Serverless::Function	} SAMによるLambda関数の定義
Properties:	
{ DeploymentPreference:	} カナリアデプロイの定義
Type: Canary10Percent10Minutes	
Alarms:	
- !Ref ErrorsAlarm	} CloudWatchアラームの定義
Hooks:	
PreTraffic: !Ref PreTrafficHook	} Hook用Lambda関数の指定
PostTraffic: !Ref PostTrafficHook	

デプロイライフサイクルイベント



Hook スクリプト 実行可能箇所

- BeforeAllowTraffic
- AfterAllowTraffic

CodeDeploy **Lambda** カナリアデプロイメント



CodeDeploy **Lambda** カナリアデプロイメント

V2コードがトラフィックを受け取る前にHook関数が実行される



CodeDeploy **Lambda** カナリアデプロイメント

警告によるロールバックに備えて10分間待機



CodeDeploy **Lambda** カナリアデプロイメント

デプロイメントが完了



Amazon ECS のデプロイ

CodeDeploy ECS blue/greenデプロイメント

- Greenタスクをプロビジョニングし、ロードバランサーのトラフィックを切り替え
- 検証Hookによって各ステージのデプロイメントでテストを有効化
- Hookが失敗した場合やAmazon CloudWatchアラームを検知した場合は数秒でBlueタスクに迅速にロールバック

アプリケーションの作成

アプリケーションの作成

アプリケーションの設定

アプリケーション名
アプリケーション名の入力

BlackBeltECSDemo

(100 文字以内)

コンピューティングプラットフォーム
コンピューティングプラットフォームを選択する

Amazon ECS

キャンセル

アプリケーションの作成

デプロイグループの作成

BlackBeltECSDemo

通知 アプリケーションの削除

Application details

名前	コンピューティングプラットフォーム
BlackBeltECSDemo	Amazon ECS

デプロイ | デプロイグループ | リビジョン

デプロイグループ

詳細を表示 編集 デプロイグループの作成 < 1 > ⚙️

名前	ステータス	最後に試行したデプロイ	最後に成功したデプロイ
<p>デプロイグループはありません</p> <p>CodeDeploy を使ってアプリケーションをデプロイする前に、デプロイグループを作成する必要があります。</p> デプロイグループの作成			

デプロイグループの作成

デプロイグループの作成

アプリケーション

アプリケーション
BlackBeltECSDemo
コンピューティングタイプ
Amazon ECS

デプロイグループ名

デプロイグループ名の入力

BlackBeltECSDemo

100文字以内

サービスロール

サービスロールの入力

ターゲットインスタンスに AWS CodeDeploy アクセスを付与する、CodeDeploy アクセス許可内のサービスロールを入力します。

arn:aws

emo

環境設定

ECS クラスター名を選択します

BlackBeltDemo

ECS サービス名を選択します

blackbelt-demo-fargate

Load balancer

Load balancerを選択

BlackBeltDemoALB

本稼働リスナーポート

HTTP: 80

テストリスナーポート - オプション

トラフィックによって再ルーティングされる前に置き換えバージョンをテストする場合は、テストリスナーが必要です

HTTP: 8080

ターゲットグループ 1 の名前

tg-BlackB-blackbelt-demo-1

ターゲットグループ 2 の名前

tg-BlackB-blackbelt-demo-2

デプロイ設定

トラフィックの再ルーティング

トラフィックがすぐに置き換え先環境に再ルーティングされるか、再ルーティングプロセスの開始まで待機状態になるかを選択します。

すぐにトラフィックを再ルーティング

トラフィックを再ルーティングするタイミングを指定します

デプロイ設定

デフォルトおよびカスタムデプロイ設定のリストから選択します。デプロイ設定には、アプリケーションのデプロイの速度やデプロイの成功または失敗の条件などのルールが定義されています。

CodeDeployDefault::ECSAllAtOnce

または デプロイ設定の作成

元のバージョンの終了

元のタスクセットを終了する前に CodeDeploy が待機する時間を指定します。終了処理が開始すると、手動で、または自動的にロールバックすることはできません。

日間

時間

分

0

1

0

▶ 詳細 - オプション

キャンセル

デプロイグループの作成

デプロイの作成

BlackBeltECSDemo

編集 削除 デプロイの作成

デプロイグループの詳細

デプロイグループ名 BlackBeltECSDemo	アプリケーション名 BlackBeltECSDemo	コンピューティングプラットフォーム Amazon ECS
デプロイタイプ Blue/Green	サービスロール ARN a /BlackBeltDemo	デプロイ設定 CodeDeployDefault.ECSAllAtOnce
ロールバックが有効になりました -	エージェント更新スケジュール AWS Systems Manager での更新のスケジュールの詳細 🔗	

環境設定

ECS クラスター名 BlackBeltDemo 🔗	ECS サービス名 blackbelt-demo-fargate 🔗
---	---

Load Balancing

ターゲットグループ 1 の名前 tg-BlackB-blackbelt-demo-1	ターゲットグループ 2 の名前 tg-BlackB-blackbelt-demo-2
本稼働リスナー ARN arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/application/BlackBeltDemoALB/97903f7b2868606c/343d6a0515cee288	テストリスナー ARN arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/application/BlackBeltDemoALB/97903f7b2868606c/e8a30260dabc523a

デプロイの作成

Create deployment

デプロイ設定

アプリケーション
BlackBeltECSDemo

デプロイグループ

BlackBeltECSDemo

コンピューティングプラットフォーム
Amazon ECS

デプロイタイプ
Blue/Green

リビジョンタイプ

アプリケーションは Amazon S3 に格納されていま
す

AppSpec エディタの使用

AppSpec 言語

JSON

YAML

```
1 version: 0.0
2- Resources:
3- - TargetService:
4     Type: AWS::ECS::Service
5     Properties:
6     TaskDefinition: "ar" 46:task-c
7     LoadBalancerInfo:
8     ContainerName: "blackbelt-demo-fargate"
9     ContainerPort: 80
10
```

テキストファイルとして保存

デプロイの説明

デプロイの説明: オプション
デプロイについての簡単な説明を追加します

▶ デプロイグループのオーバーライド

▶ ロールバック設定の上書き

キャンセル

デプロイの作成

デプロイ実行

d-1K9KHV5X8 🔄 デプロイを停止 デプロイを停止してロールバック

デプロイのステータス

ステップ 1:
置き換えタスクセットのデプロイ 50%
🔄 進行中

ステップ 2:
テストトラフィックルーティングのセットアップ 0%
未開始

ステップ 3:
本稼働トラフィックを置き換えタスクセットに再ルーティング中 0%
未開始

ステップ 4:
1 時間 0 分を待機 0%
未開始

ステップ 5:
元のタスクセットの終了 0%
未開始

トラフィック移行の進行状況

オリジナル 100.0% 置き換え 0.0%

元のタスクセットがトラフィックを処理しています

置き換えタスクセットがトラフィックを処理していません

デプロイの詳細

アプリケーション	デプロイ ID	ステータス
BlackBeltECSDemo	d-1K9KHV5X8	🔄 進行中
デプロイ設定	デプロイグループ	開始:
CodeDeployDefault.ECSAllAtOnce	BlackBeltECSDemo	ユーザーアクション
デプロイの説明		-

CodeDeploy **ECS** appspec

version: 0.0

Resources:

```
- TargetService:
  Type: AWS::ECS::Service
  Properties:
    TaskDefinition: "my_task_definition:8"
    LoadBalancerInfos:
      ContainerName: "SampleApp"
      ContainerPort: 80
```

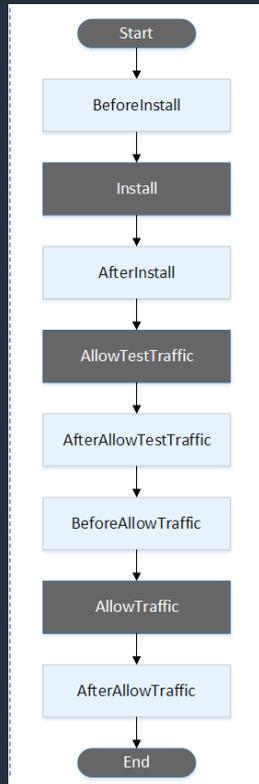
ターゲットタイプの指定
タスク定義
ロードバランサの定義

Hooks:

```
- BeforeInstall: "BeforeNewRevisionInstallation"
- AfterInstall: "AfterNewRevisionInstallation"
- AfterAllowTestTraffic: "AfterTestTrafficShift"
- BeforeAllowTraffic: "BeforeTrafficShift"
- AfterAllowTraffic: "AfterTrafficShift"
```

フック関数の定義
※実際にはLambdaのARNを指定

デプロイライフサイクルイベント



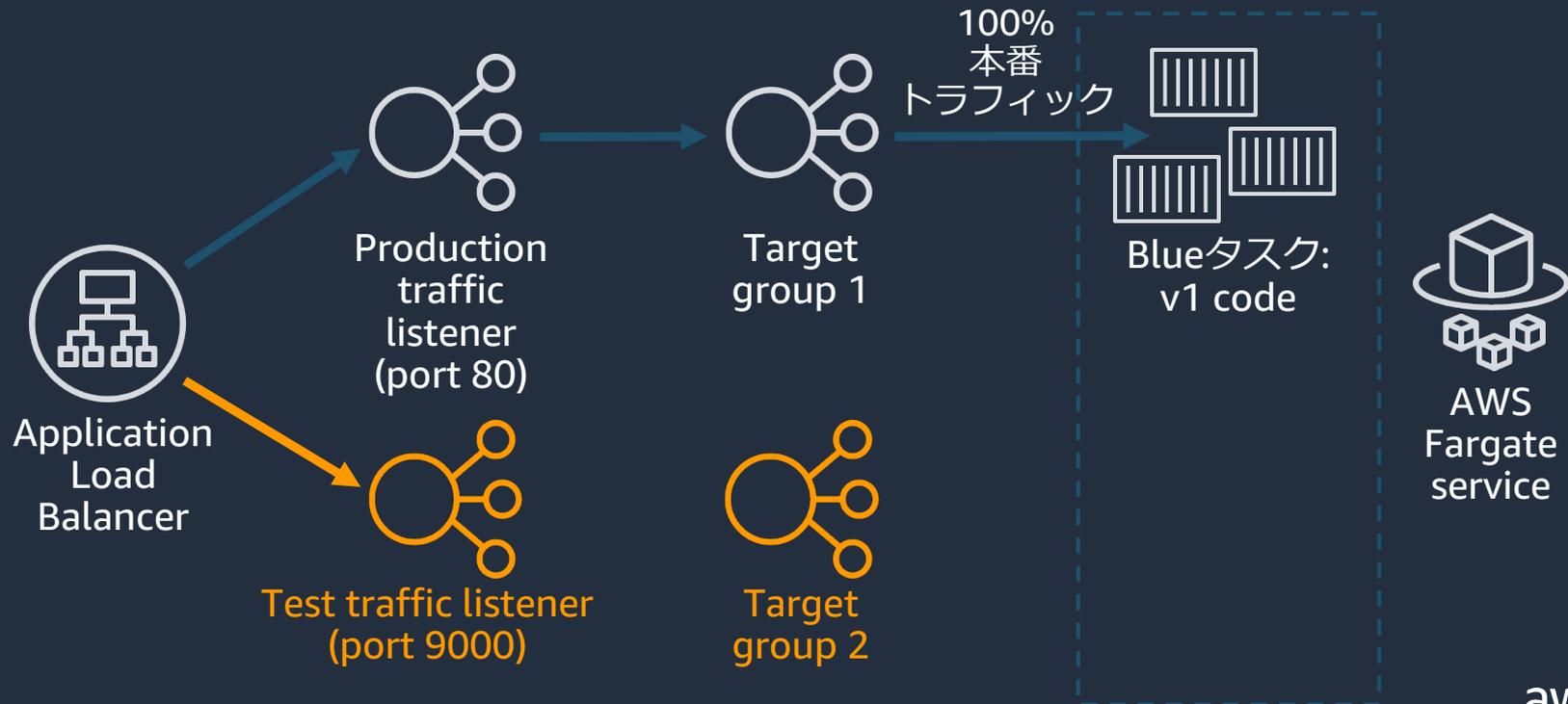
Hook スクリプト 実行可能箇所

- BeforeInstall
- AfterInstall
- AfterAllowTestTraffic
- BeforeAllowTraffic
- AfterAllowTraffic

CodeDeploy ECS blue/greenデプロイメント

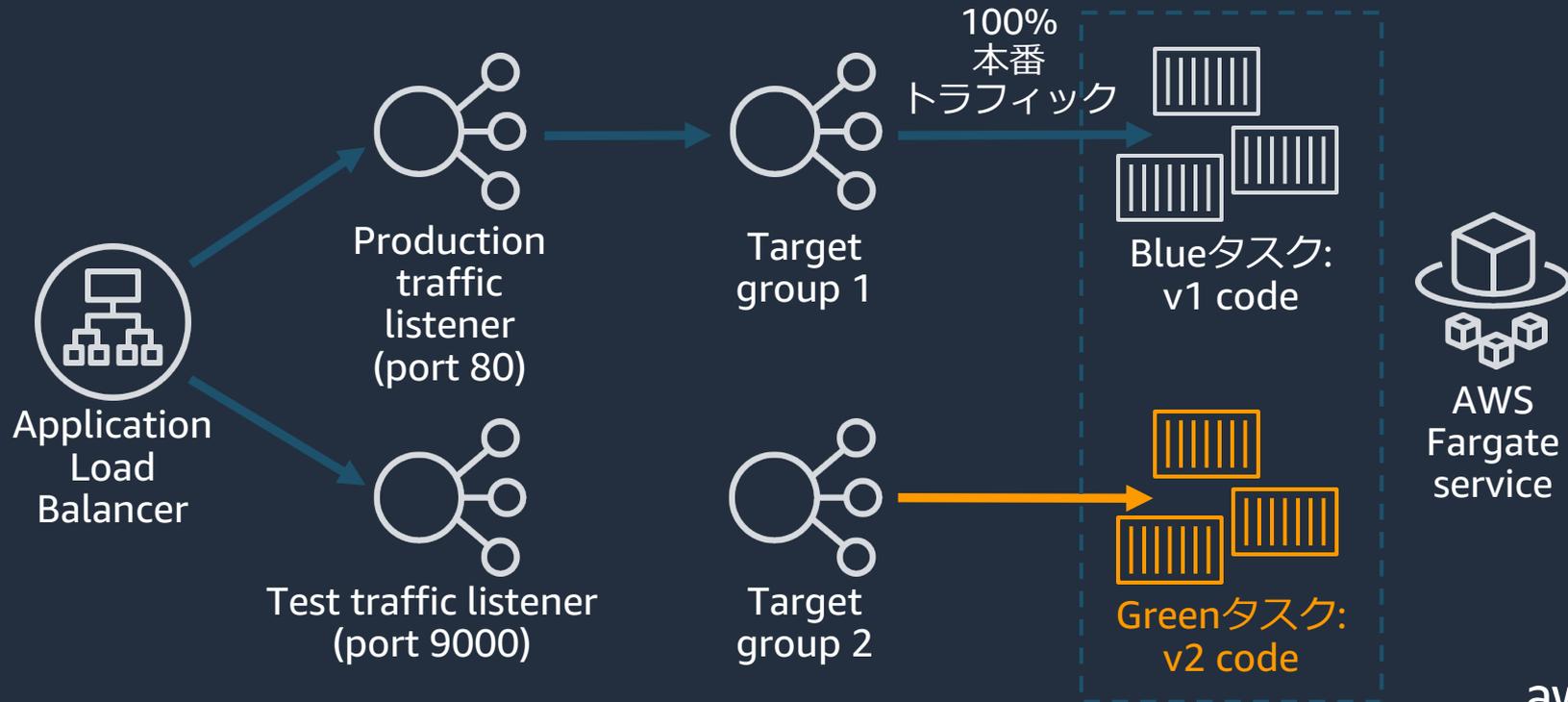


CodeDeploy ECS blue/greenデプロイメント



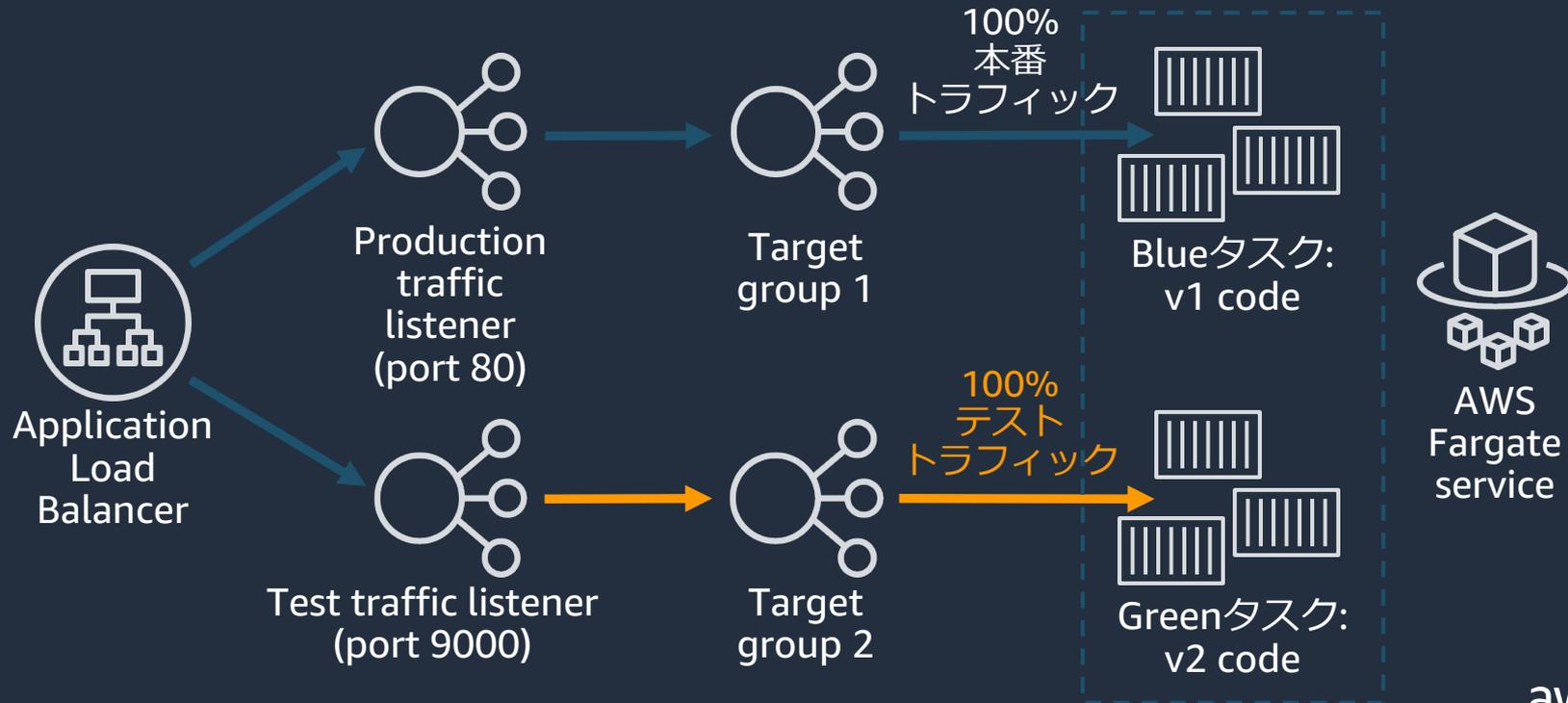
CodeDeploy ECS blue/greenデプロイメント

greenタスクを配置



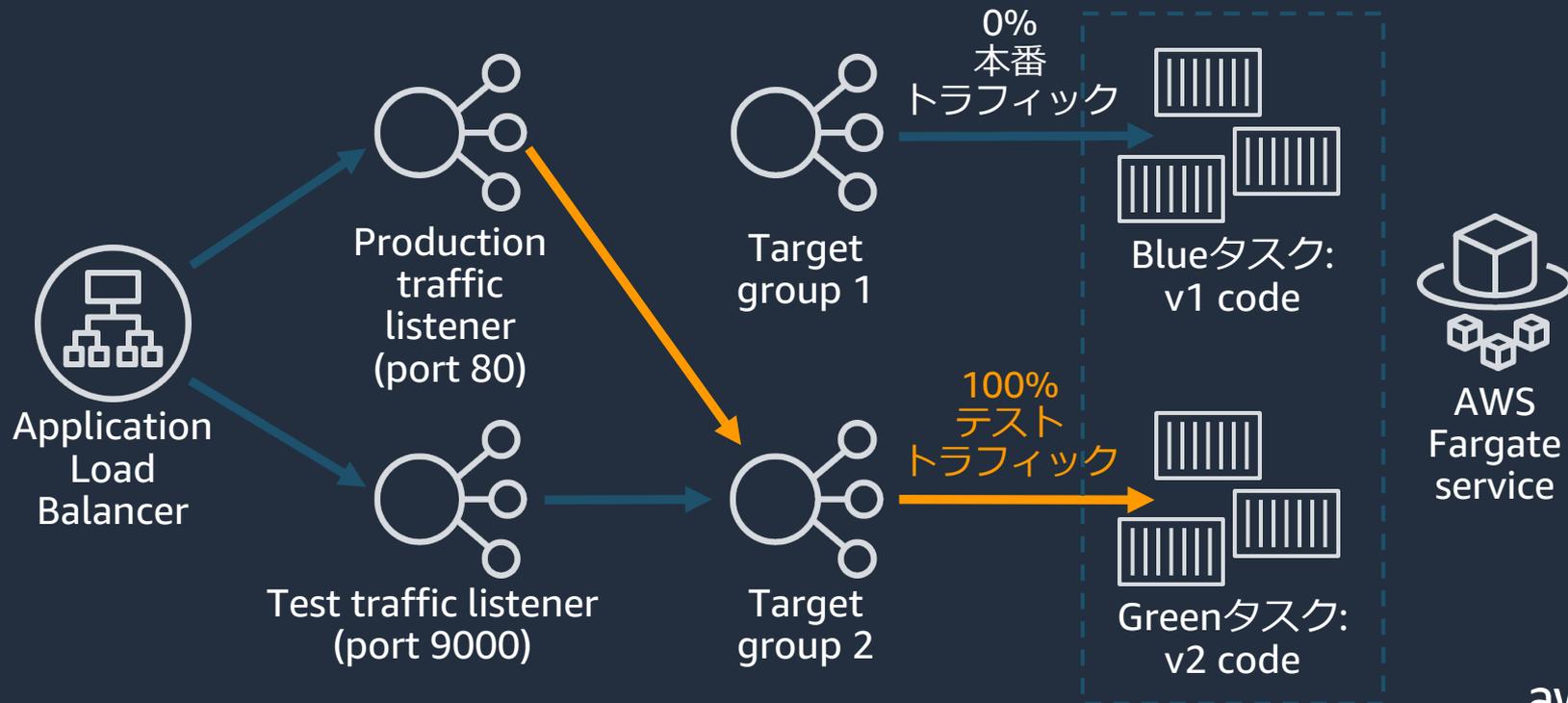
CodeDeploy ECS blue/greenデプロイメント

greenタスクが本番用トラフィックを受け取る前にテストエンドポイントでhookが実行される



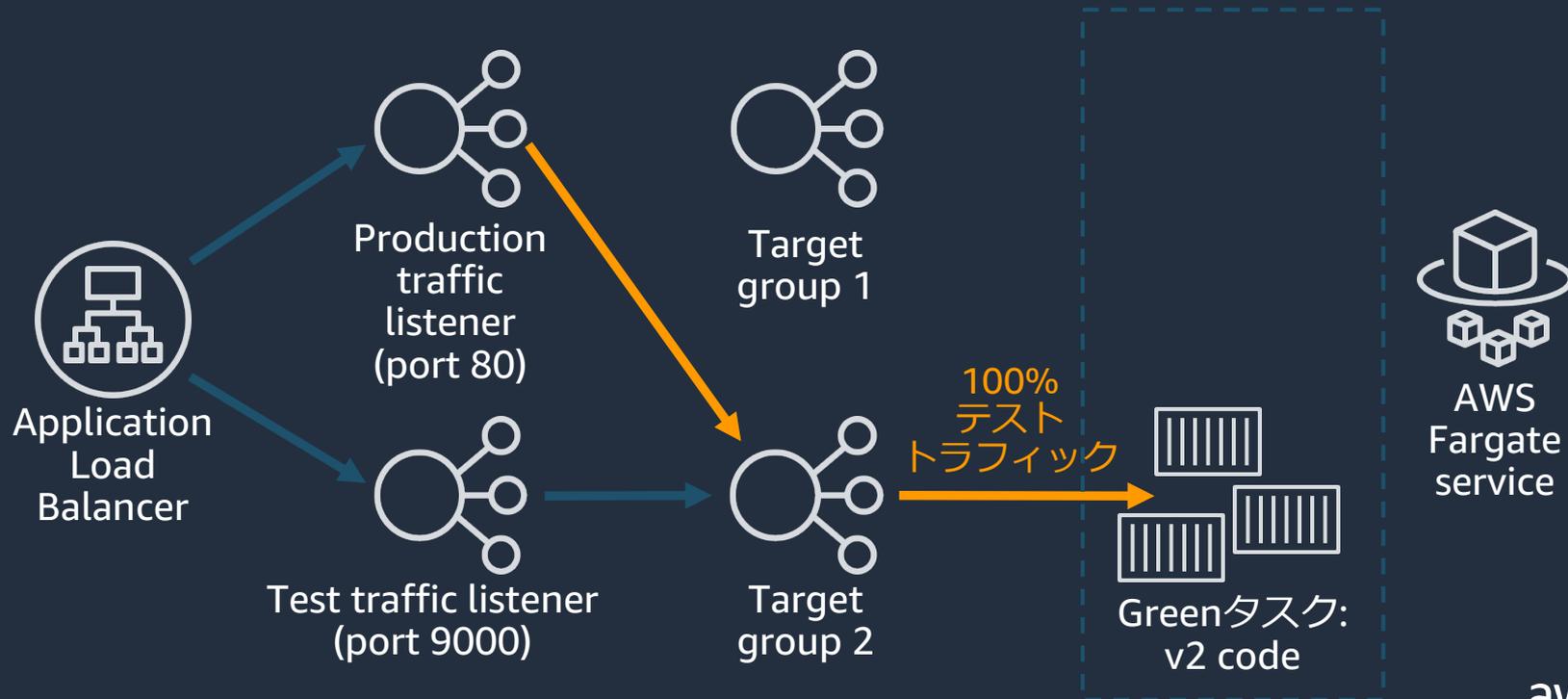
CodeDeploy ECS blue/greenデプロイメント

Greenタスクへのトラフィックを切り替え。
もし失敗を検知した場合はロールバック



CodeDeploy ECS blue/greenデプロイメント

blueタスクをドレイニング



重要 : デプロイメントのためのコンテナイメージのタグ付け

- Dockerタグはデプロイ時のみではなく個々のコンテナの開始時にも利用
- latest や prod タグはスケールアウトイベント発生時に未テストのコードを本番環境へデプロイする結果を招く
- デプロイメントにはユニークでイミュータブルなタグを利用すべき

デプロイメントのためのコンテナイメージのタグ付け



AWS
Fargate
service



Amazon
ECR
repository

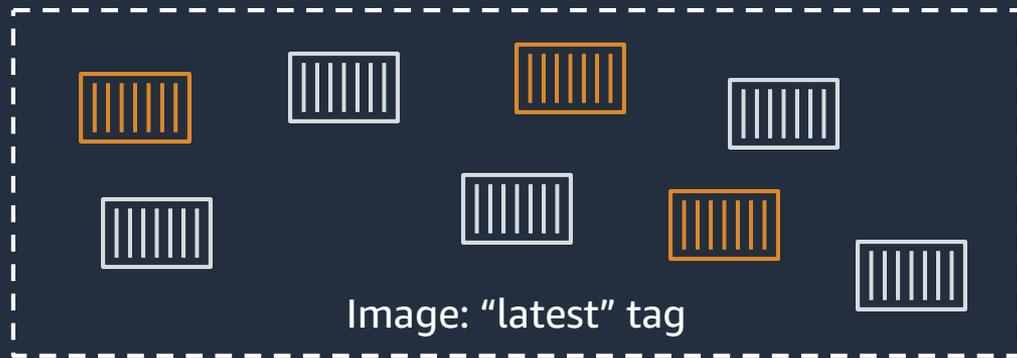
デプロイメントのためのコンテナイメージのタグ付け

ビルドが新しい latest イメージをプッシュ

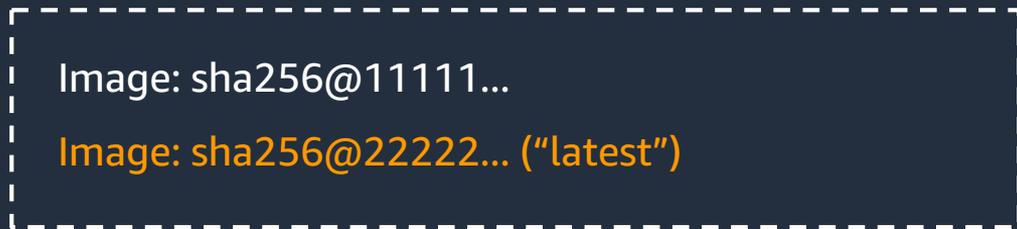


デプロイメントのためのコンテナイメージのタグ付け

サービスがスケールアウトし新しいタスクを実行開始



AWS
Fargate
service



Amazon
ECR
repository

デプロイメントのためのコンテナイメージのタグ付け

イミュータブルなタグをデプロイ

SHA256 digestを利用

```
{  
  "name": "sample-app",  
  "image": "amazon/amazon-ecs-  
    sample@sha256:3e39d933b1d948c92309bb583b5a1f3d28f0119e1551ca1fe538ba414a41af48d"  
}
```

Build IDを利用

```
{  
  "name": "sample-app",  
  "image": "amazon/amazon-ecs-sample:build-b2085490-359f-4eaf-8970-6d1e26c354f0"  
}
```

デプロイメントのためのコンテナイメージのタグ付け

ビルド時にイミュータブルなタグを生成

SHA256 digest

```
export IMAGE_URI=`docker inspect --format='{{index .RepoDigests 0}}' my_image:$IMAGE_TAG`
```

サンプル結果：

```
amazon/amazon-ecs-sample@sha256:3e39d933b...
```

Build ID

```
export IMAGE_TAG=build-`echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}'`
```

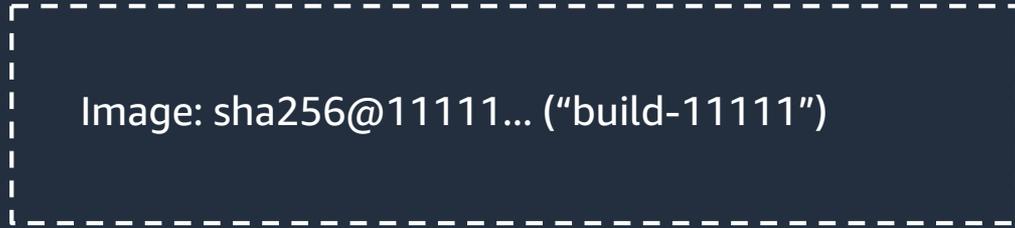
サンプル結果：

```
build-b2085490-359f-4eaf-8970-6d1e26c354f0
```

デプロイメントのためのコンテナイメージのタグ付け



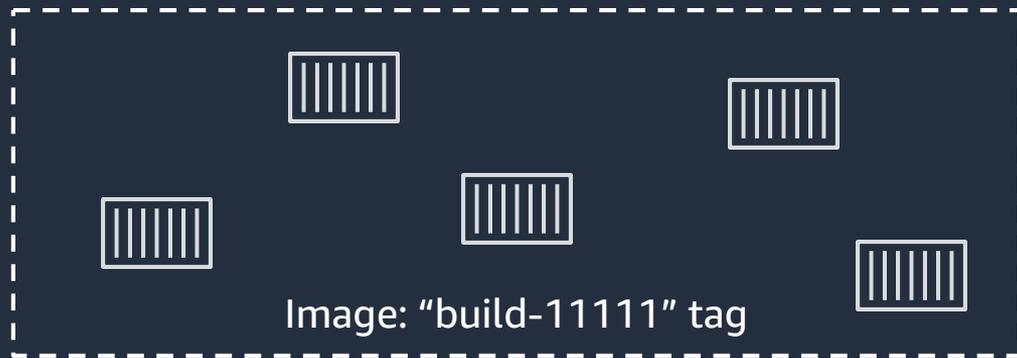
AWS
Fargate
service



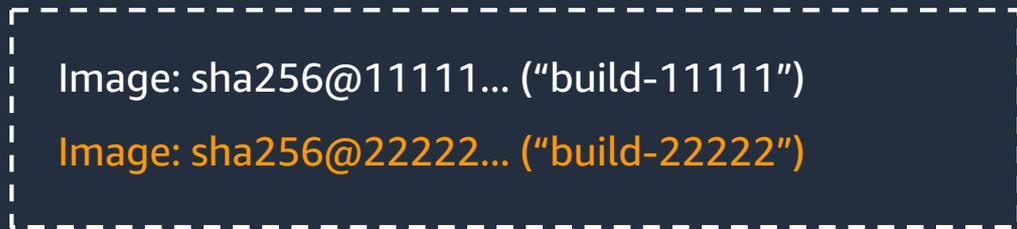
Amazon
ECR
repository

デプロイメントのためのコンテナイメージのタグ付け

ビルドが新しいビルドIDのイメージタグをプッシュ



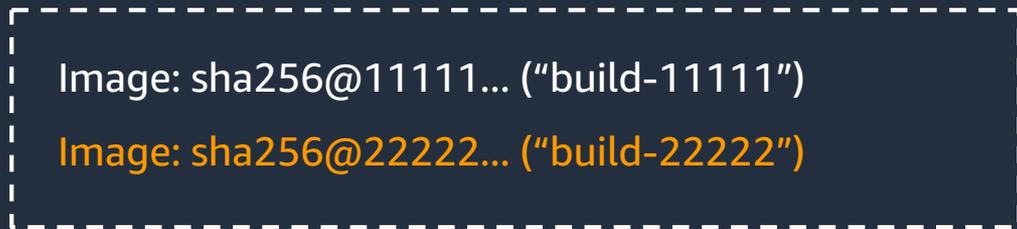
AWS
Fargate
service



Amazon
ECR
repository

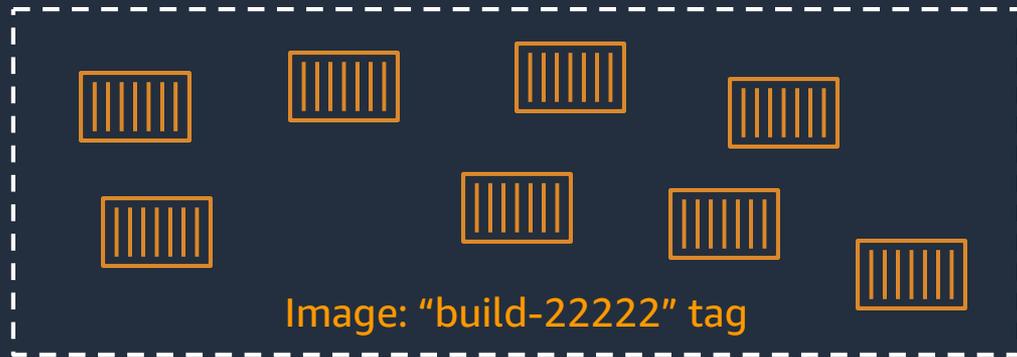
デプロイメントのためのコンテナイメージのタグ付け

サービスがスケールアウトし、新しいタスクを実行しようとする

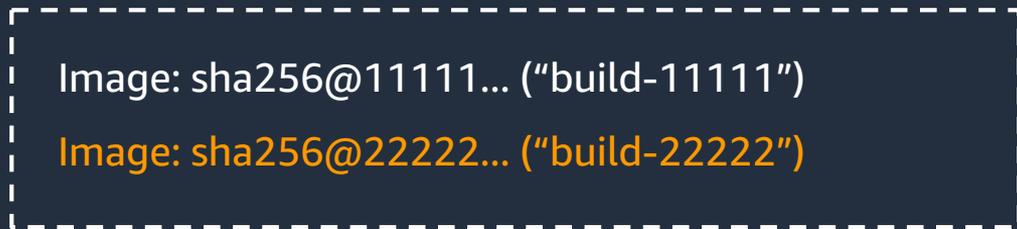


デプロイメントのためのコンテナイメージのタグ付け

デプロイメントはサービス定義を更新し、タスクを置き換え



AWS
Fargate
service



Amazon
ECR
repository

機能紹介

Validation Hook Lambda

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
  //Read the DeploymentId from the event payload.
  var deploymentId = event.DeploymentId;

  //Read the LifecycleEventHookExecutionId from the event payload
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

  /*
   Enter validation tests here.
  */

  // Prepare the validation test results with the deploymentId and
  // the lifecycleEventHookExecutionId for AWS CodeDeploy.
  var params = {
    deploymentId: deploymentId,
    lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
    status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
  };

  // Pass AWS CodeDeploy the prepared validation test results.
  codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
  });
};
```

} テストコード

} CodeDeploy に結果を連携

CloudWatch アラームによるデプロイ停止

メトリクス やログを監視し、デプロイを停止することが可能

デプロイアラームを作成する

このデプロイグループのデプロイを自動的に停止するアラームを追加します。アラームは最大 10 個追加できます。

Amazon CloudWatch アラーム

① まず最初に Amazon CloudWatch でアラームを作成する必要があります。メトリックおよびそのしきい値を指定した後、デプロイグループにアラームを追加します。

サービスロールに、cloudwatch:DescribeAlarms 権限を与える必要があります。

キャンセル **アラームの追加**

アラーム

アラームの削除 アラームの追加

名前
<input type="radio"/> BlackBeltDemoAlarm

- アラーム設定を無視する
デプロイプロセス中は、Amazon CloudWatch アラームを確認するステップを省略します
- アラームの状態が利用できない場合でも、デプロイを続行する
Amazon Cloudwatch からアラームデータを取得できない場合でも、デプロイ実行を許可します

ステータスの通知

通知ルール、デプロイトリガーを設定してデプロイ状況の通知を受けることが可能

通知ルールの作成

通知ルールは、リソースで発生するイベントへのサブスクリプションを設定します。こうしたイベントが発生すると、指定したターゲットに通知が送信されます。通知の設定は、[設定] の環境設定で管理できます。 [情報](#)

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

何も選択しない

Deployment

Succeeded

Failed

Started

デプロイトリガーを作成する

Amazon Simple Notification Service (SNS) のトピックにサブスクライブするトリガーを作成して、このデプロイグループのデプロイおよびインスタンスイベントに関する通知を受け取ります。最大 10 個のトリガーを作成できます。

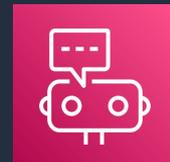
トリガー名

イベント

- デプロイの開始
- デプロイの成功
- デプロイの失敗
- デプロイの停止
- デプロイの準備が整いました
- デプロイのロールバック
- インスタンスの開始
- インスタンスの成功
- インスタンスの失敗
- インスタンスの準備が整いました



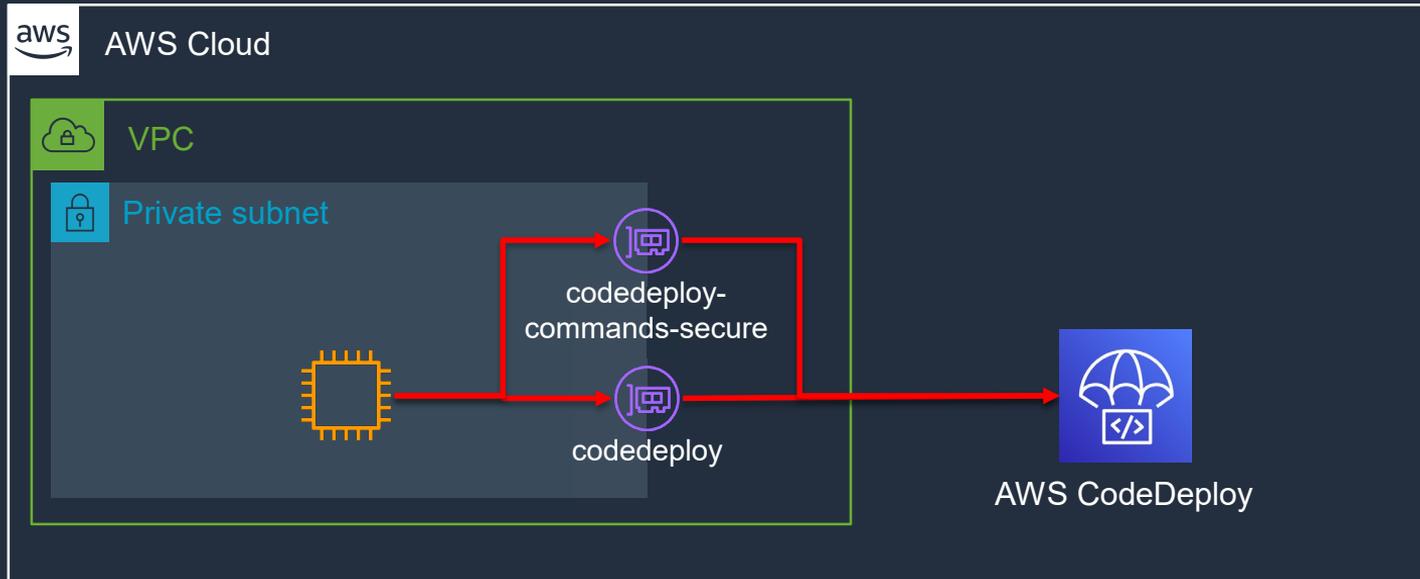
Amazon Simple Notification Service



AWS Chatbot

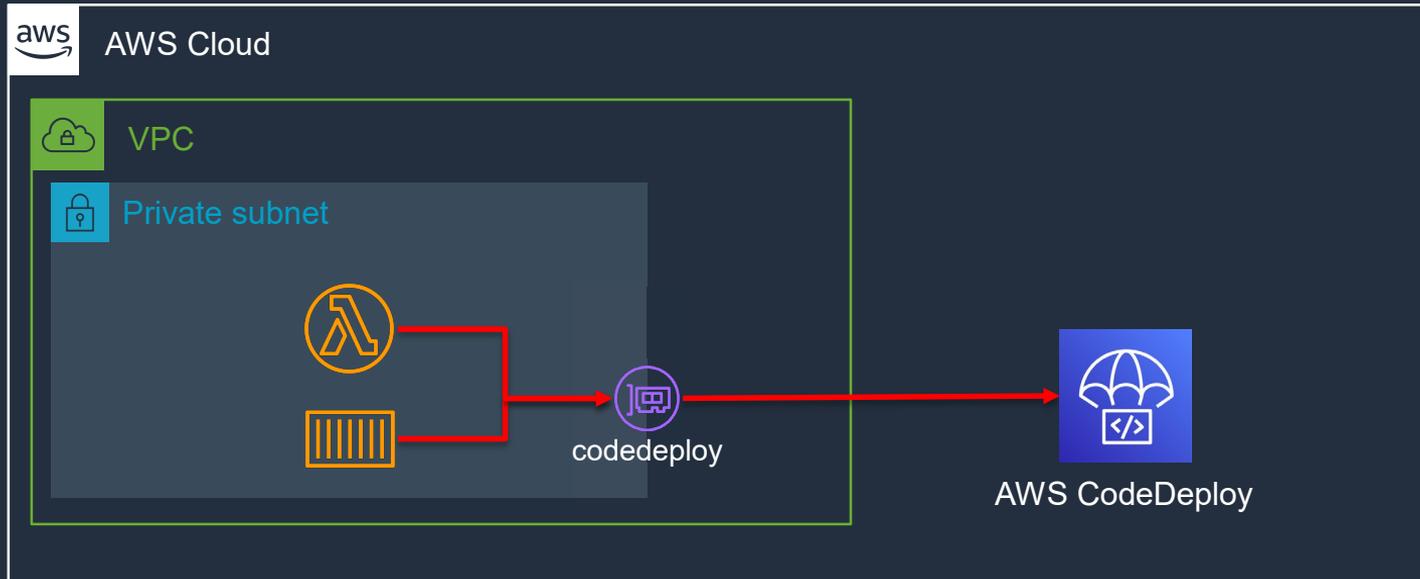
VPC エンドポイントをサポート

- EC2 へのデプロイには `codedeploy`, `codedeploy-commands-secure` の両方が必要



VPC エンドポイントをサポート

- Lambda, ECS へのデプロイには codedeploy のみ必要



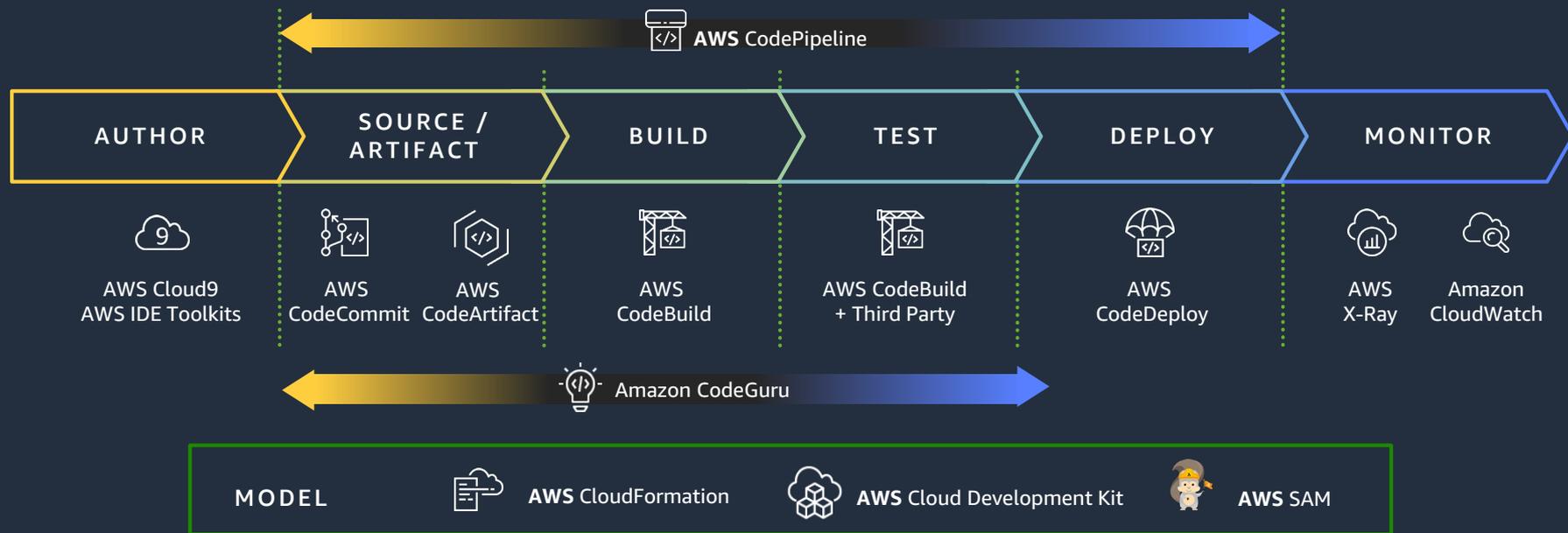
利用料金

- EC2 / Lambda / ECS へのデプロイは**無料**
- オンプレミスのインスタンスに対するデプロイは、
0.02 USD / インスタンス / デプロイ
- 3台のオンプレミスインスタンスへデプロイ
 - 3台 * 0.02 USD = 0.06 USD
- 1台のオンプレミスインスタンスへ 3回デプロイ
 - 1台 * 0.02 USD * 3回 = 0.06 USD

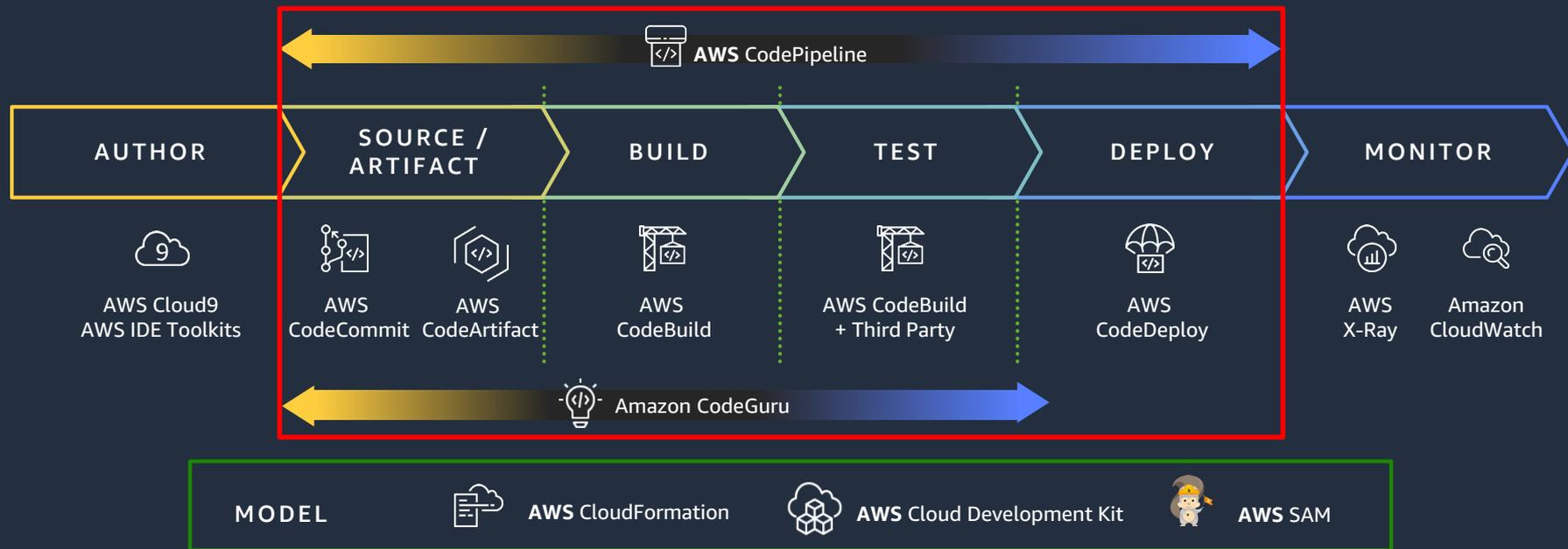
まとめ

- AWS CodeDeploy は、フルマネージドなデプロイサービス
- EC2、オンプレミス、Lambda、ECS へデプロイ可能
- 単体で使うことも、他のサービスと組み合わせることも可能
- 8種類のチュートリアル
 - https://docs.aws.amazon.com/ja_jp/codedeploy/latest/userguide/tutorials.html

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



ソフトウェア開発に関連する AWS サービス

- AWS CodeGuru
 - <https://aws.amazon.com/jp/blogs/news/webinar-bb-amazon-codeguru-2020/>
- AWS CodeStar & AWS CodePipeline
 - https://aws.amazon.com/jp/blogs/news/webinar-bb-awscodestar_awscodepipeline-2020/
- AWS CodeCommit & AWS CodeArtifact
 - https://aws.amazon.com/jp/blogs/news/webinar-bb-aws-codecommit_aws-codeartifact-2020/
- AWS CodeBuild
 - <https://aws.amazon.com/jp/blogs/news/webinar-bb-aws-codebuild-2020/>

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]



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>

