[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

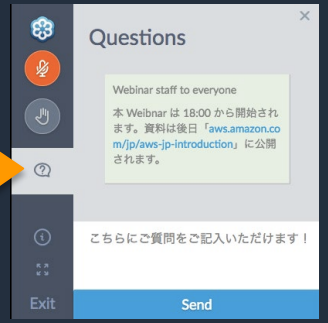
AWS Black Belt Online Seminar とは

「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

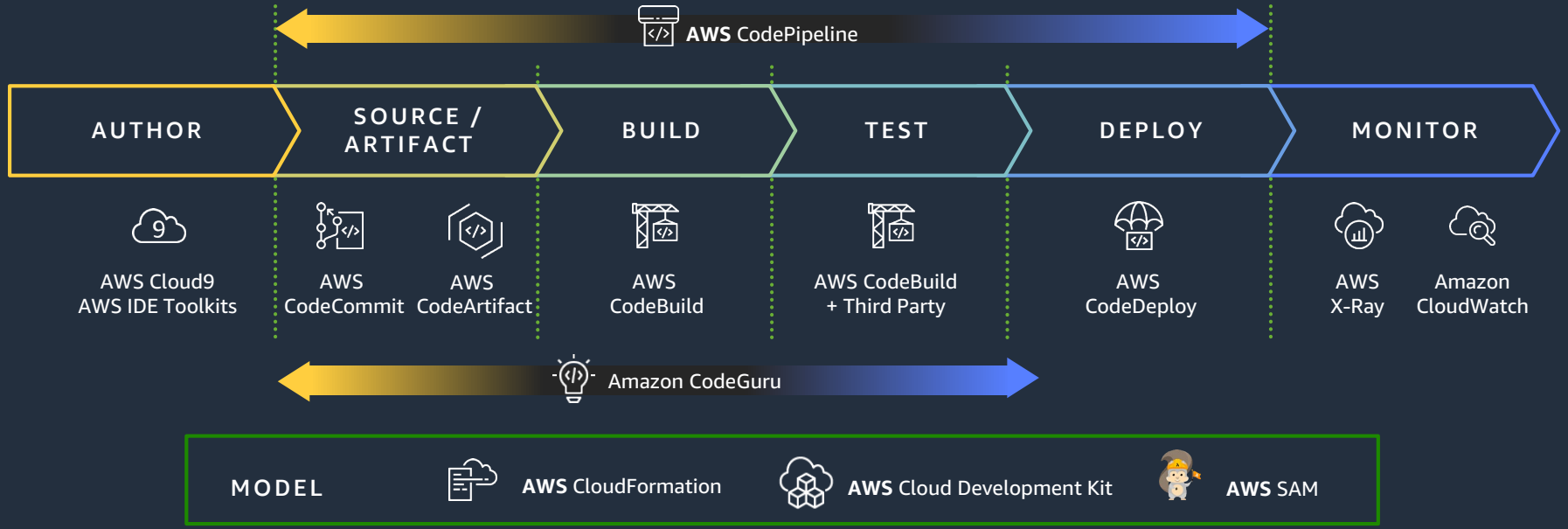
- 本セミナーで学習できること
    - AWS CodeDeploy

- 対象者
    - アーキテクトの方
    - 技術者の方

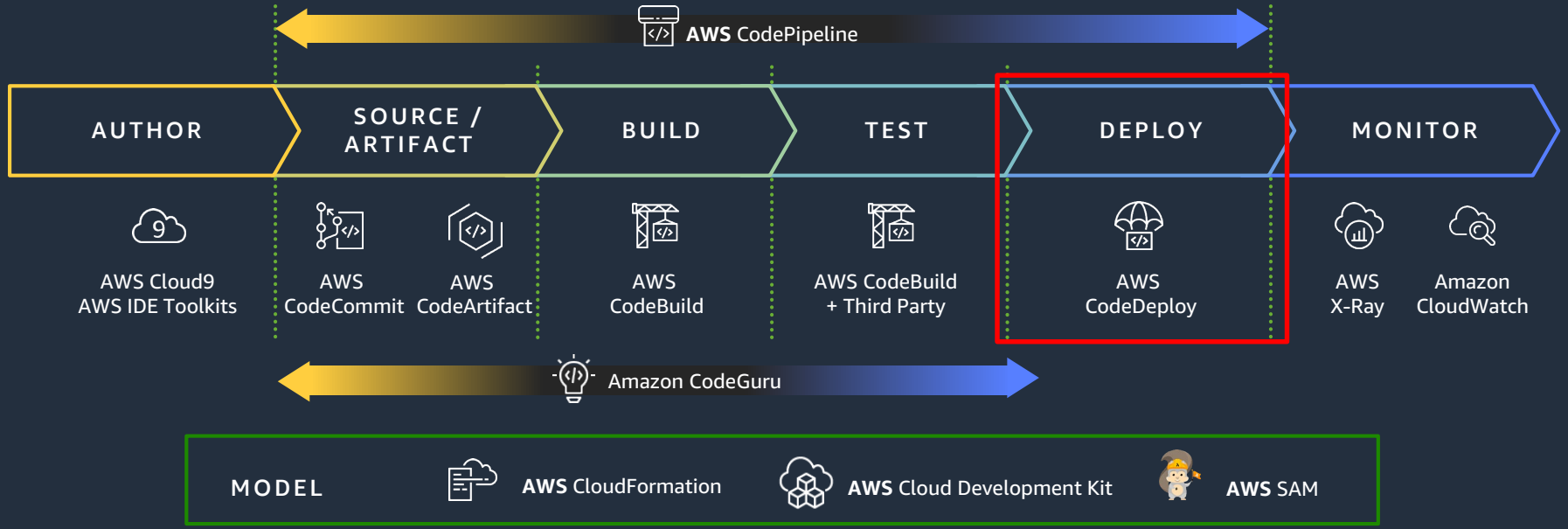
本日のアジェンダ

- 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

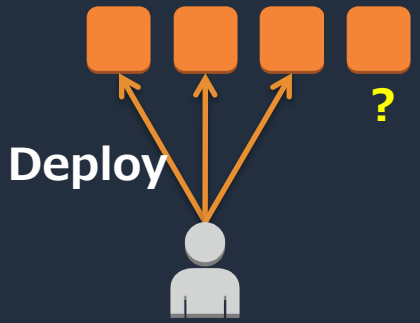


AWS CodeDeploy

- 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

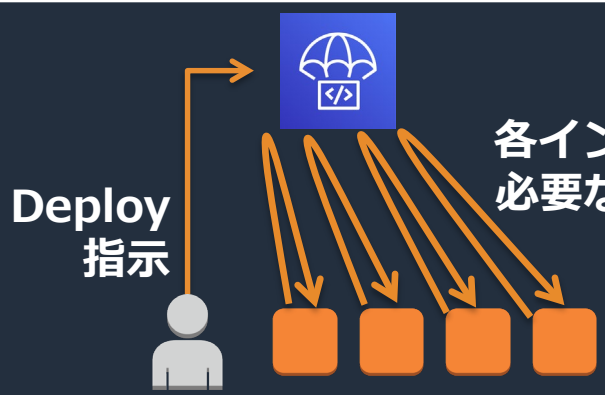
Pull 型のデプロイ手段

× 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

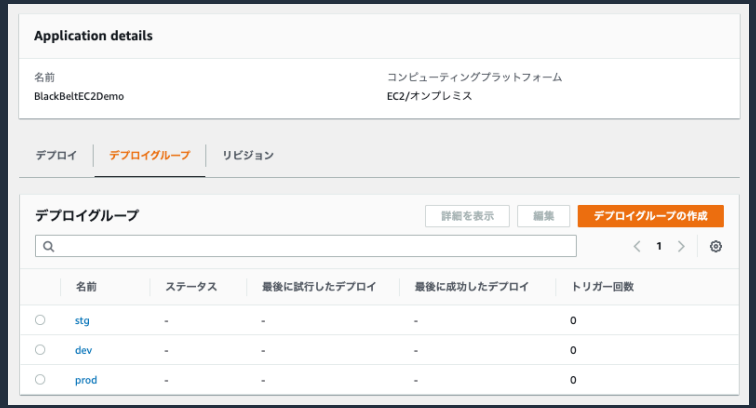
アプリケーションとコンピューティングプラットフォーム

- 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

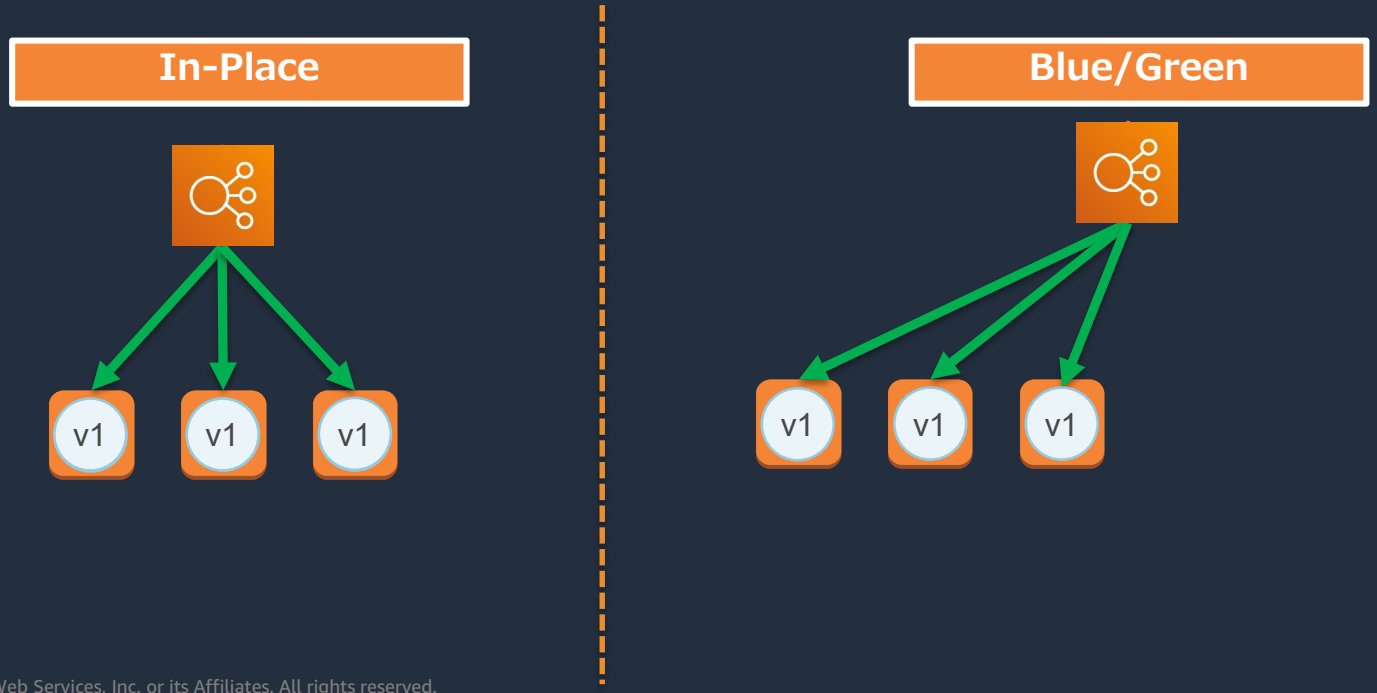
デプロイグループ

- 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



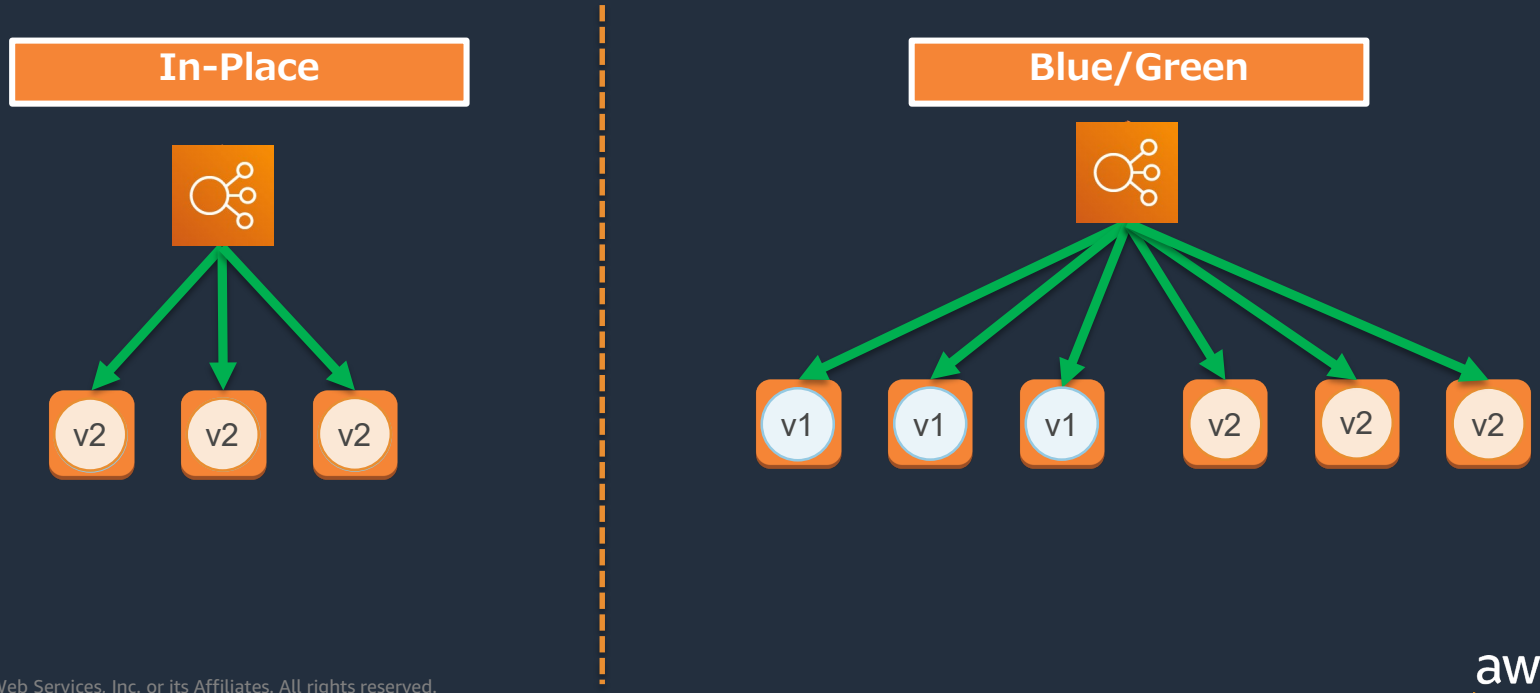
デプロイタイプ

v1 がデプロイされた状態



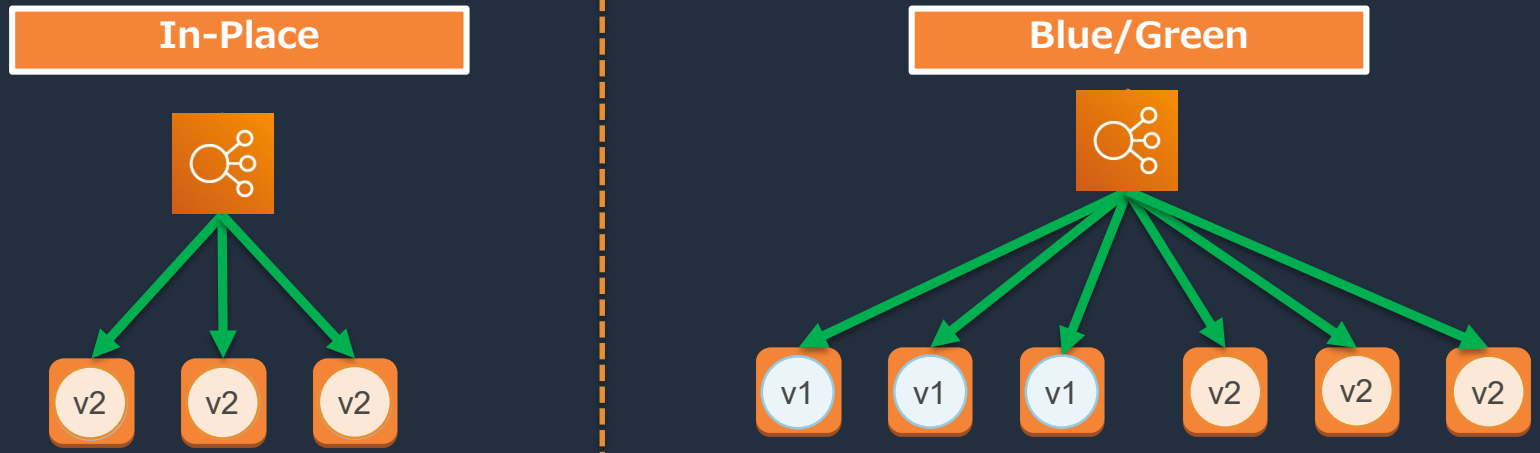
デプロイタイプ

v2 をデプロイ



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



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）

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

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

aws

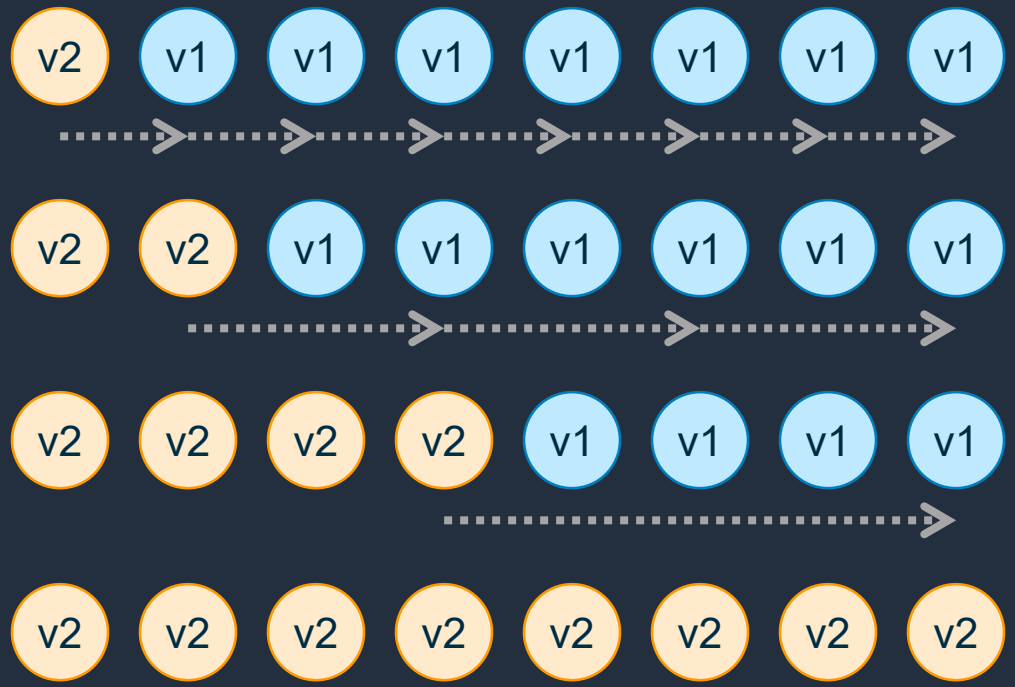
デプロイ設定 EC2/オンプレミス

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



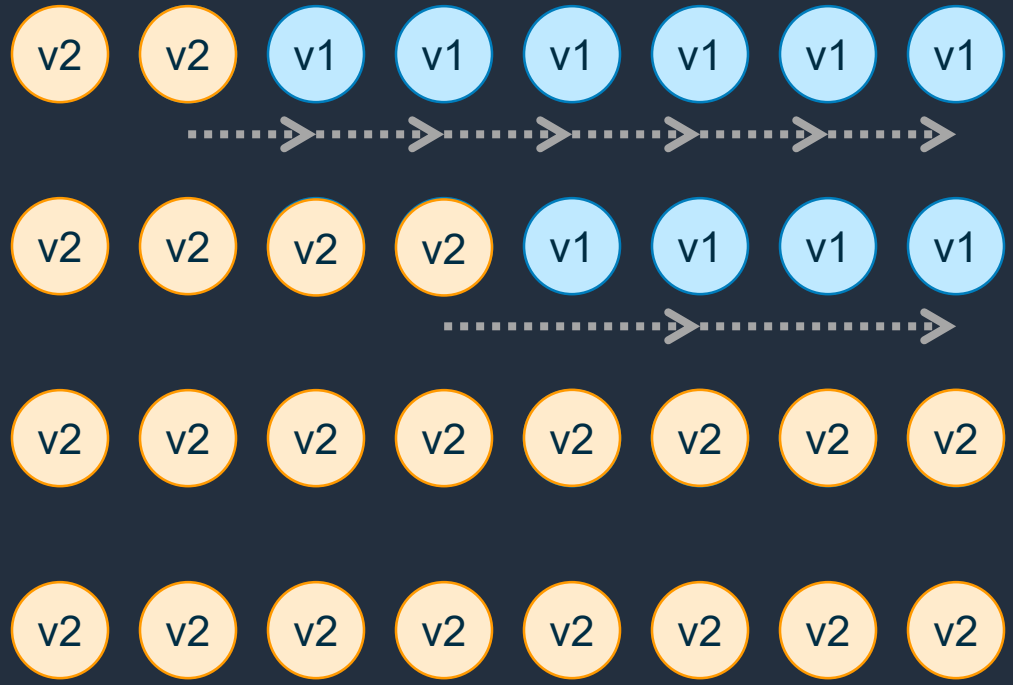
デプロイ設定 EC2/オンプレミス

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



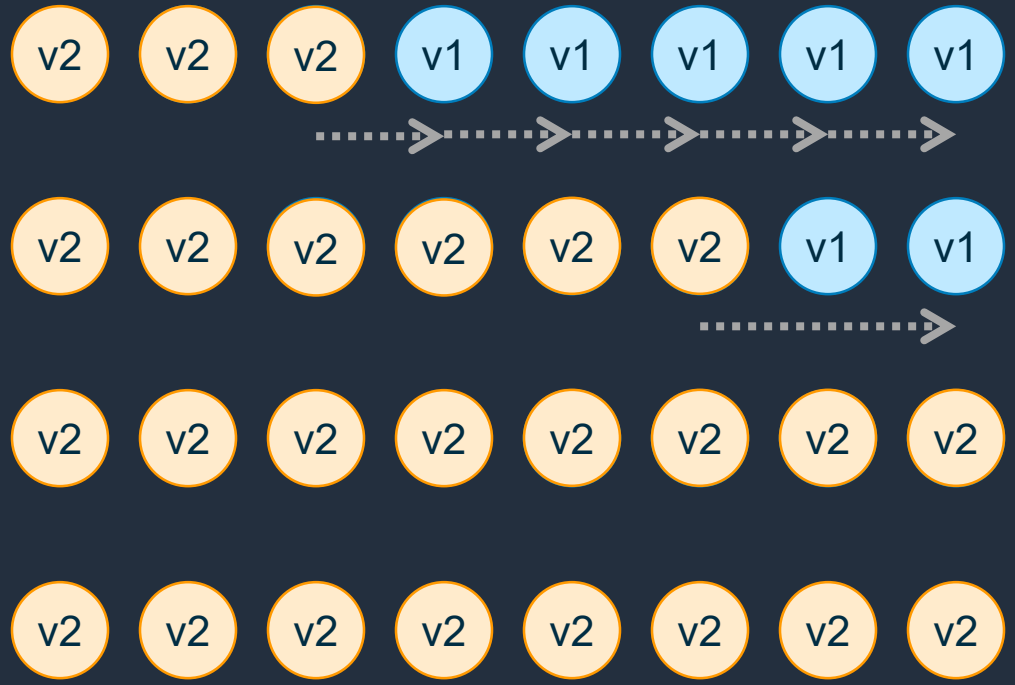
デプロイ設定 EC2/オンプレミス

One-at-a-time
Min. healthy hosts = 99%

[Custom]
Min. healthy hosts = 75%

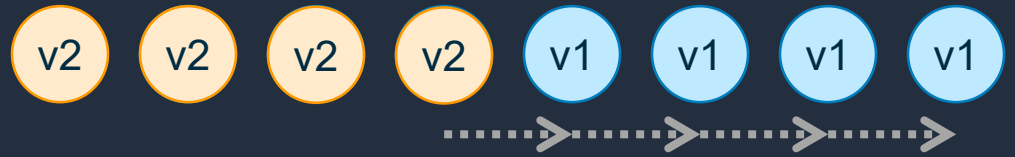
Half-at-a-time
Min. healthy hosts = 50%

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%



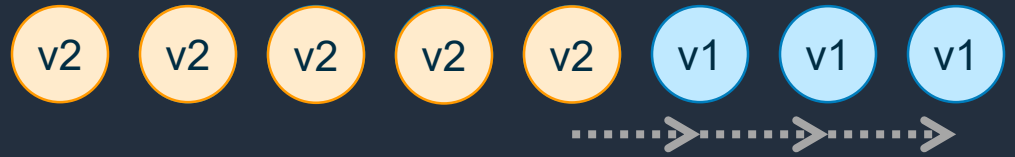
[Custom]
Min. healthy hosts = 75%

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%



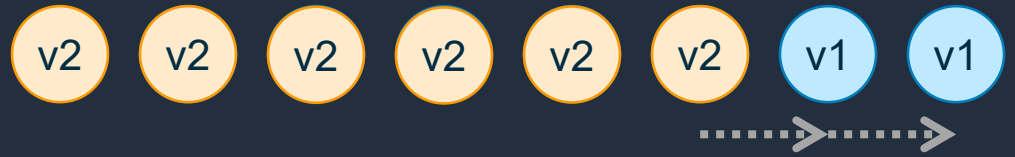
[Custom]
Min. healthy hosts = 75%

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%



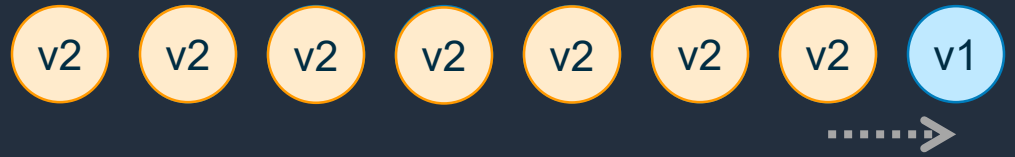
[Custom]
Min. healthy hosts = 75%

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

aws

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

aws

デプロイ設定 Lambda / ECS

Linear
Step 25%, Interval 10min

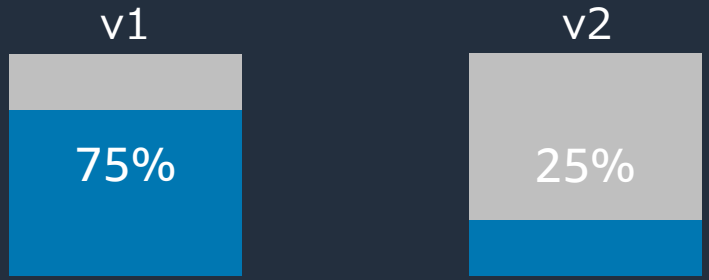
Canary
Step 25%, Interval 10min

All-at-once

デプロイ設定 Lambda / ECS

Linear
Step 25%, Interval 10min

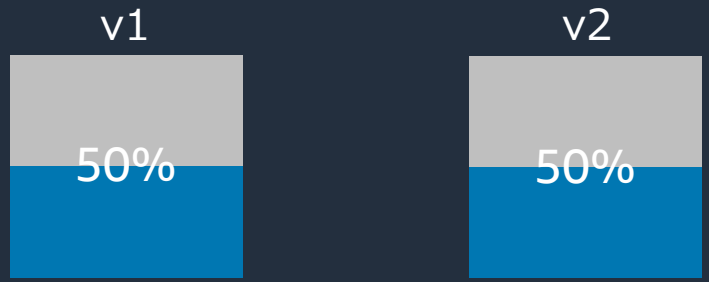
Canary
Step 25%, Interval 10min



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

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

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

- CodeDaploy に付与する IAM ロール
- CodeDaploy 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ァイル

- 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": "*"
    }
  ]
}
```


aws

EC2 / オンプレミス へのデプロイ

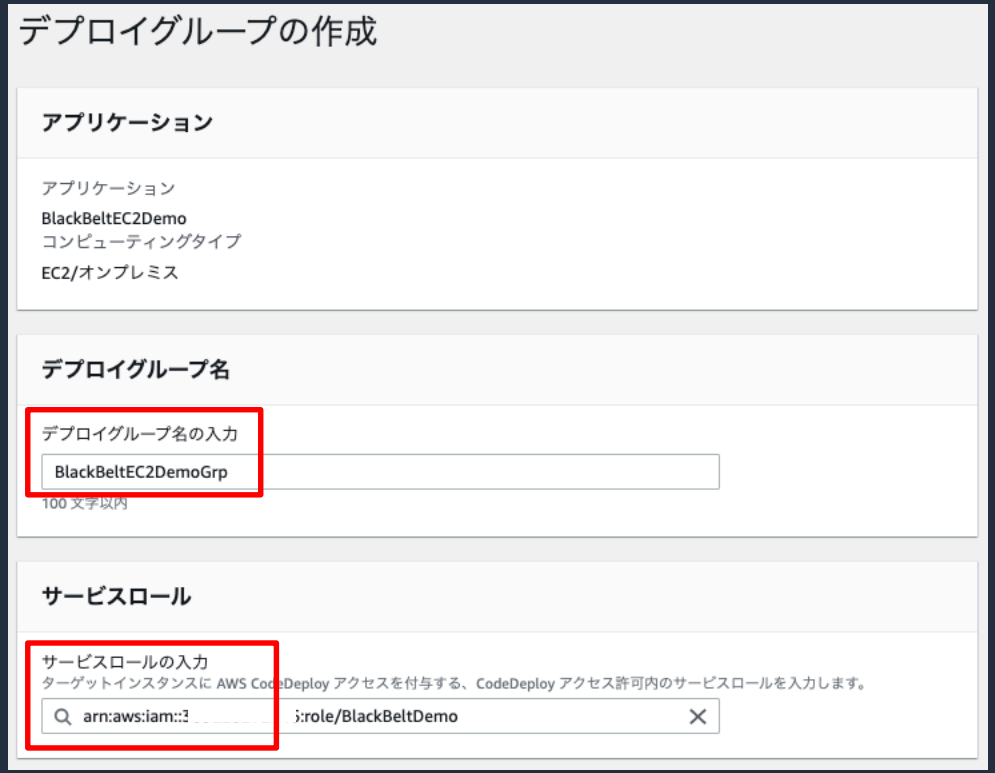
CodeDeploy EC2/オンプレミス デプロイメント

- 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

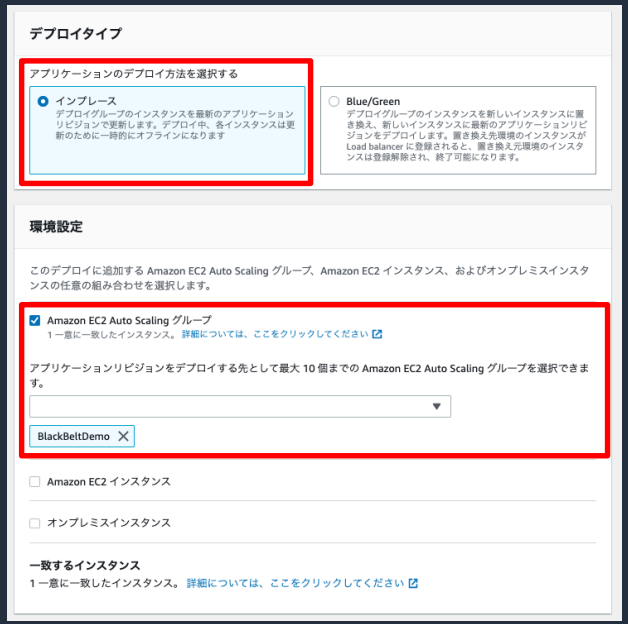
アプリケーションの作成

デプロイグループの作成

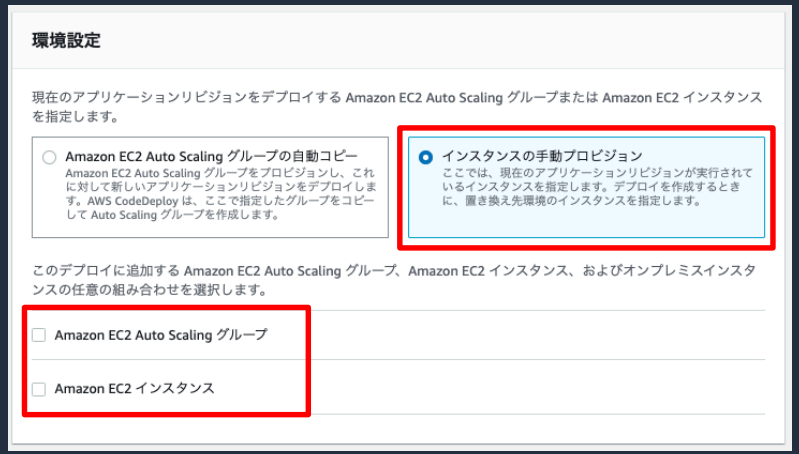
デプロイグループの作成



デプロイグループの作成

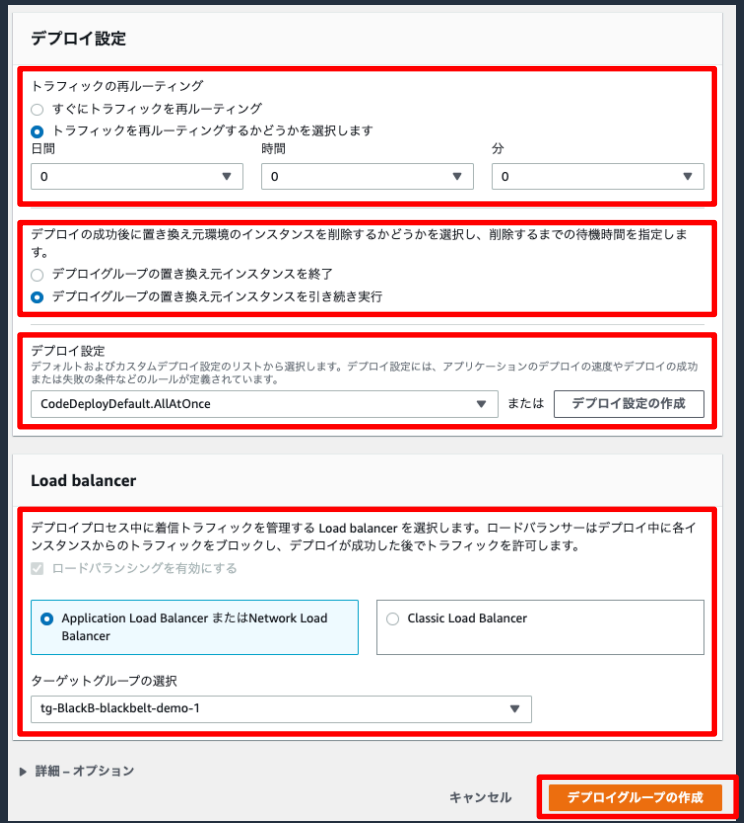


デプロイグループの作成

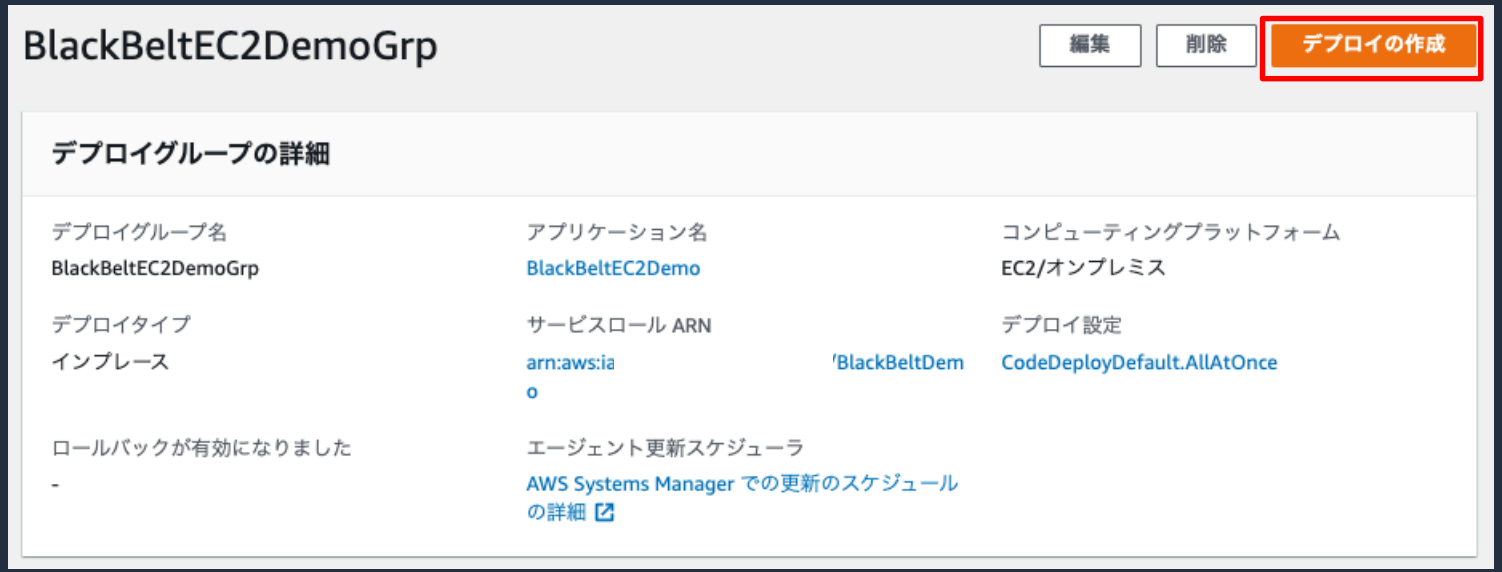


デプロイグループの作成

デプロイグループの作成

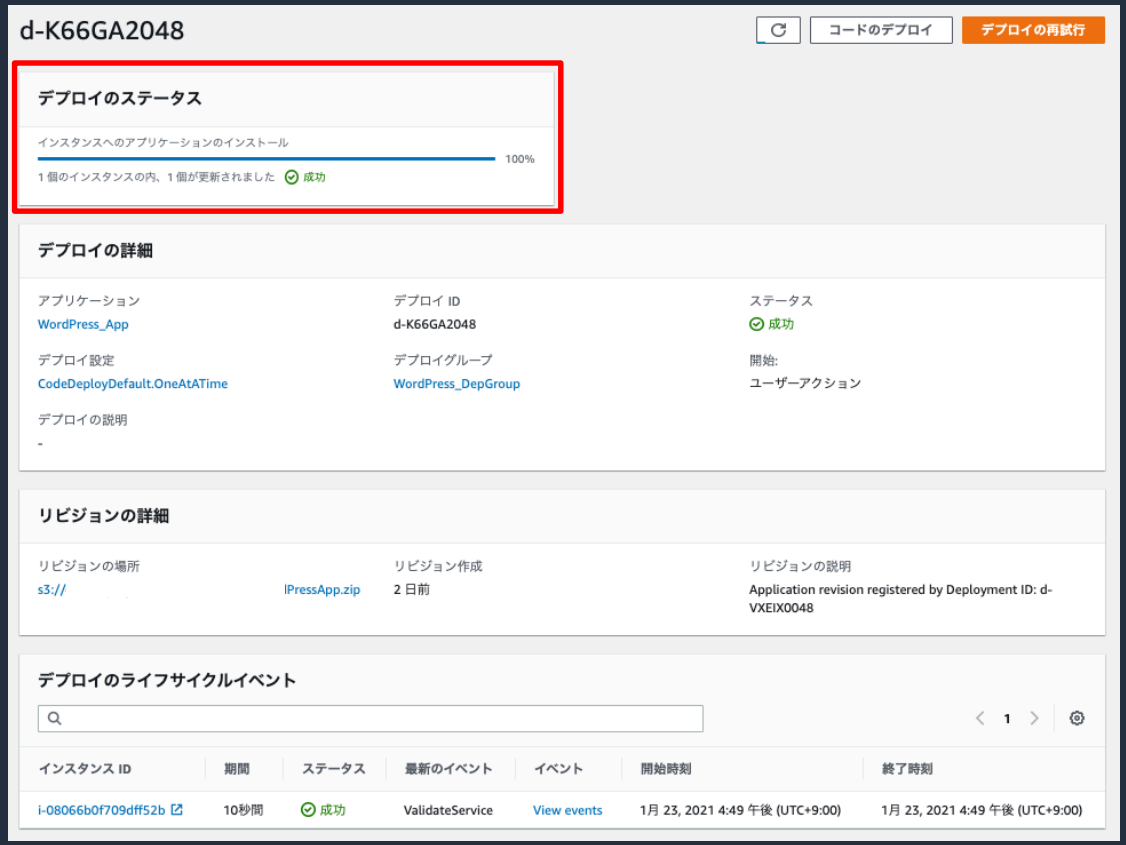


デプロイの作成



デプロイの作成

デプロイ実行



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
```


aws

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

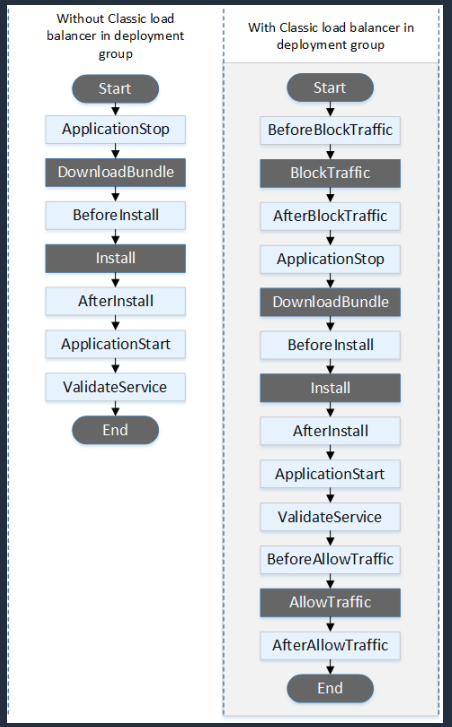
Hook用スクリプトの指定

- 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

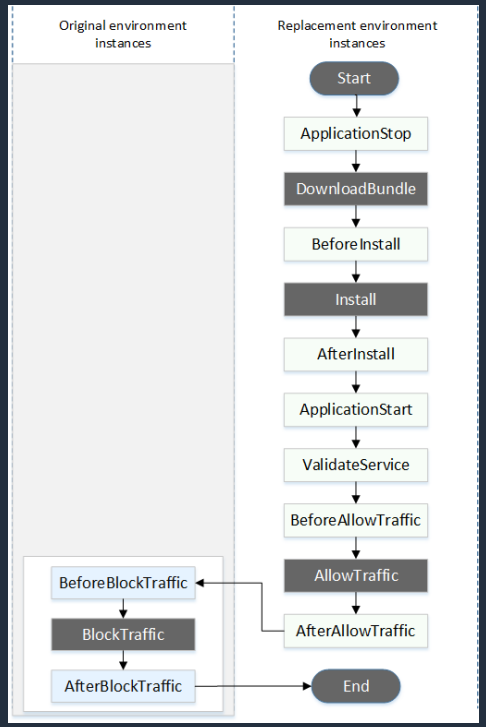
aws

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

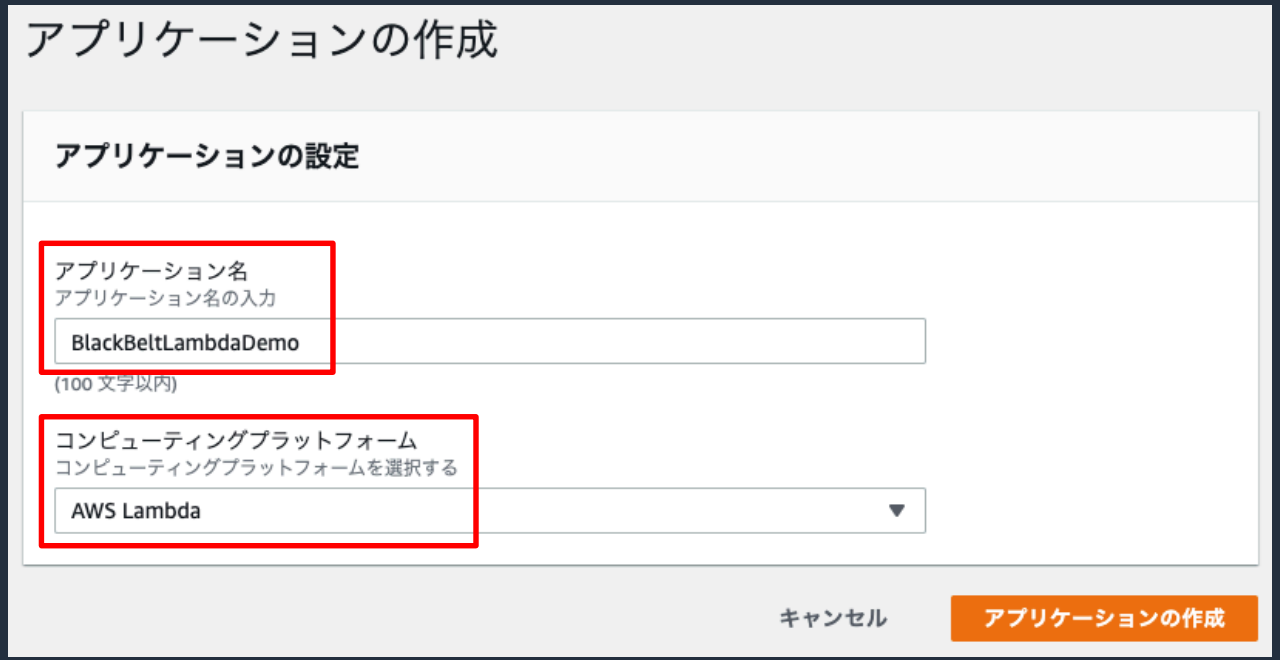
- オープンソースで利用可能なので、他の環境にも適用可能
  - https://github.com/aws/aws-codedeploy-agent

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

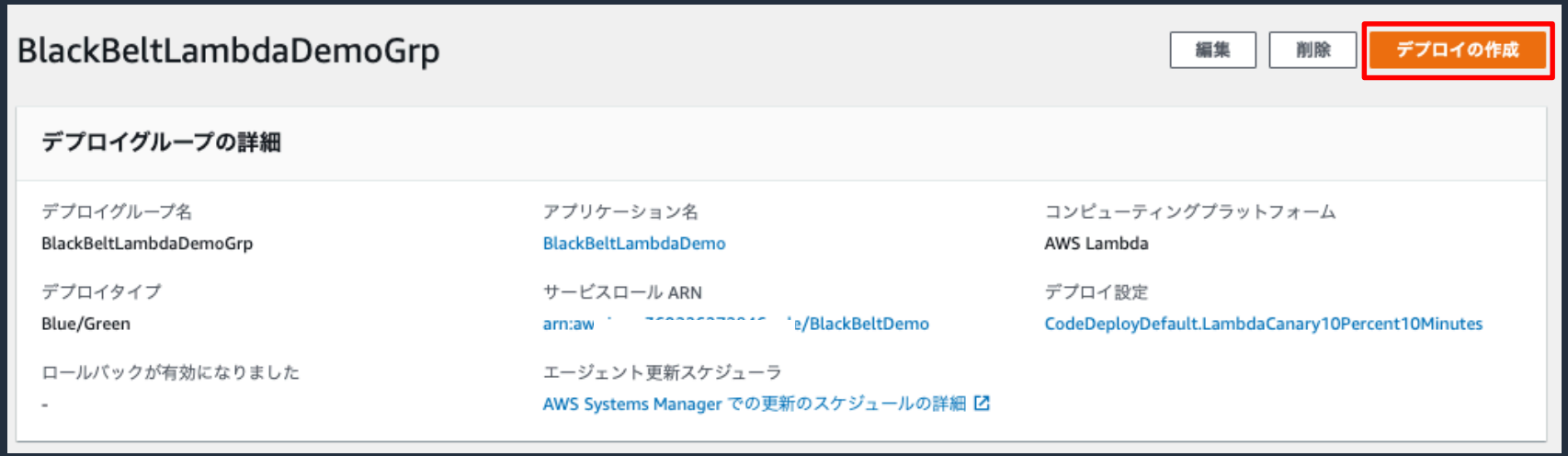
アプリケーションの作成



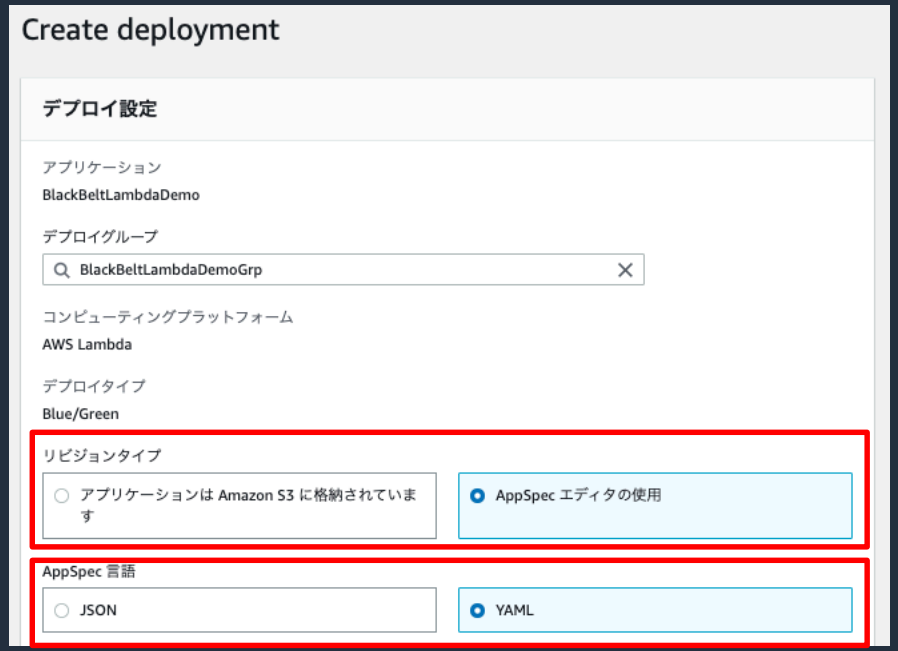
デプロイグループの作成

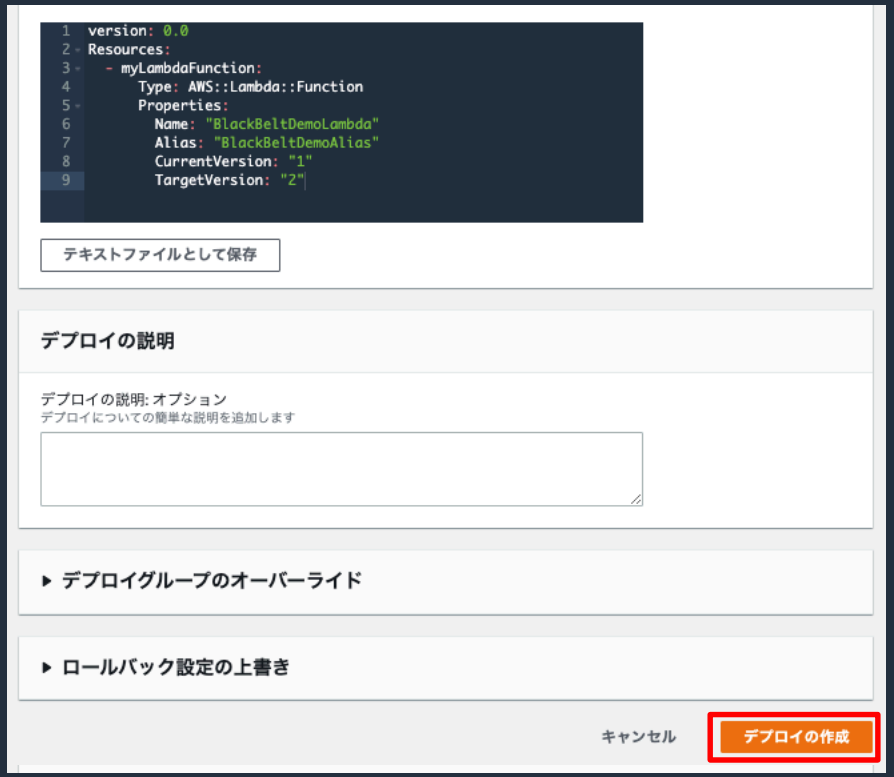
デプロイグループの作成

デプロイの作成

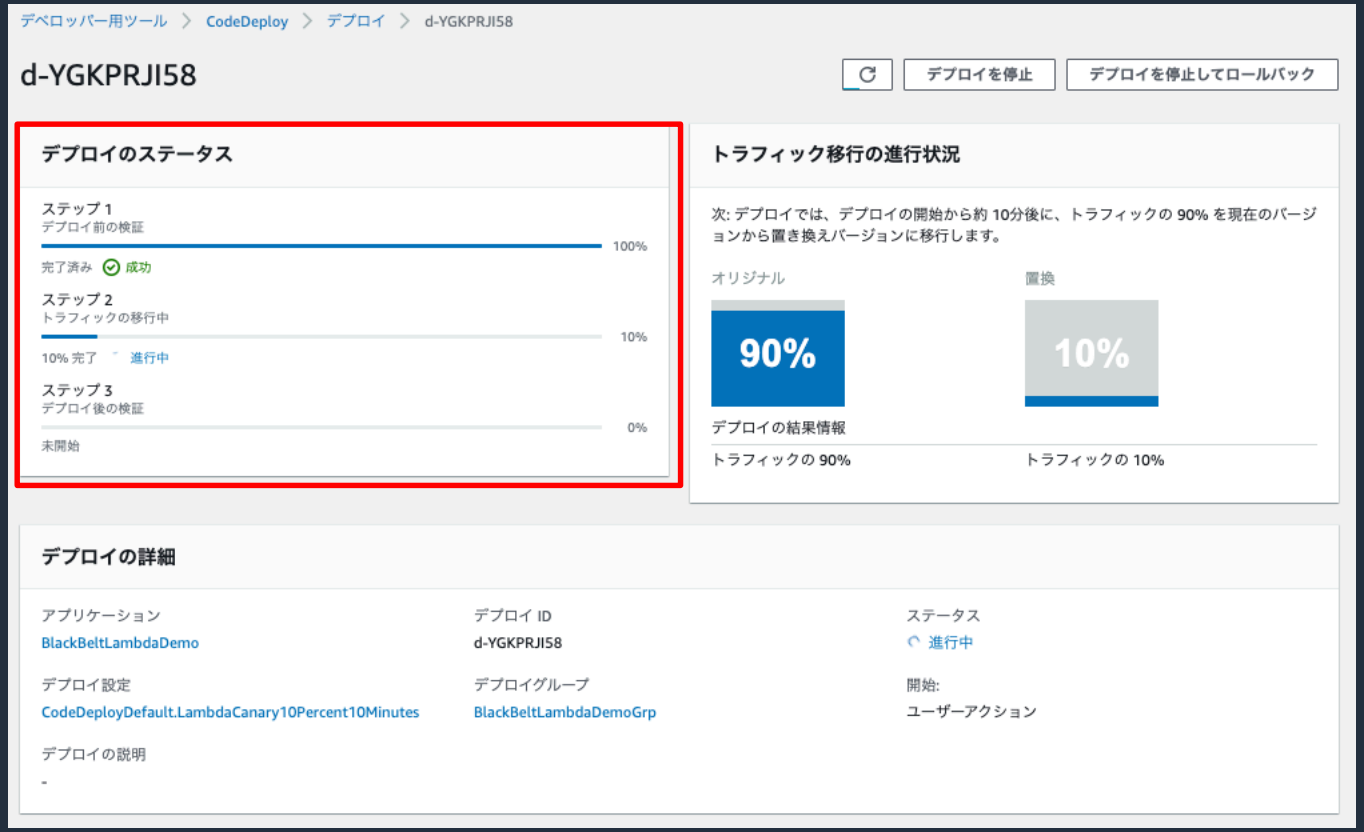


デプロイの作成





デプロイ実行中



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”
Hooks:
  - BeforeAllowTraffic: "BeforeTrafficShift”
  - AfterAllowTraffic:  "AfterTrafficShift"
```

デプロイするLambdaの指定

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
    Properties:
      DeploymentPreference:
        Type: Canary10Percent10Minutes
        Alarms:
          - !Ref ErrorsAlarm
        Hooks:
          PreTraffic: !Ref PreTrafficHook
          PostTraffic: !Ref PostTrafficHook
```

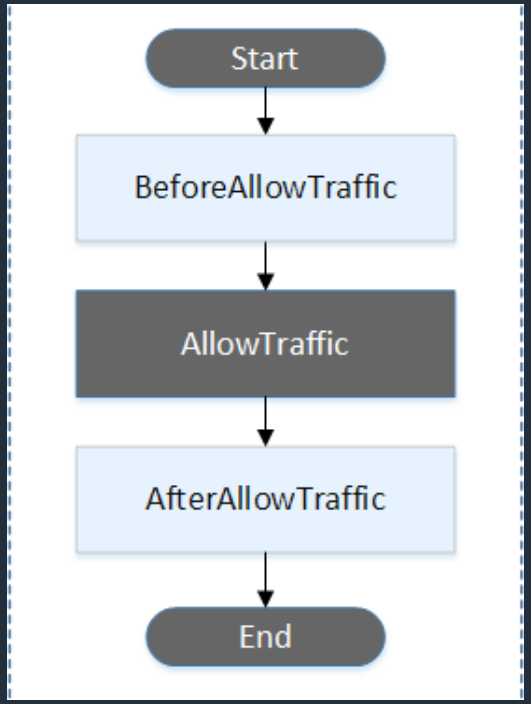
SAMによるLambda関数の定義

カナリアデプロイの定義

CloudWatchアラームの定義

Hook用Lambda関数の指定

デプロイライフサイクルイベント



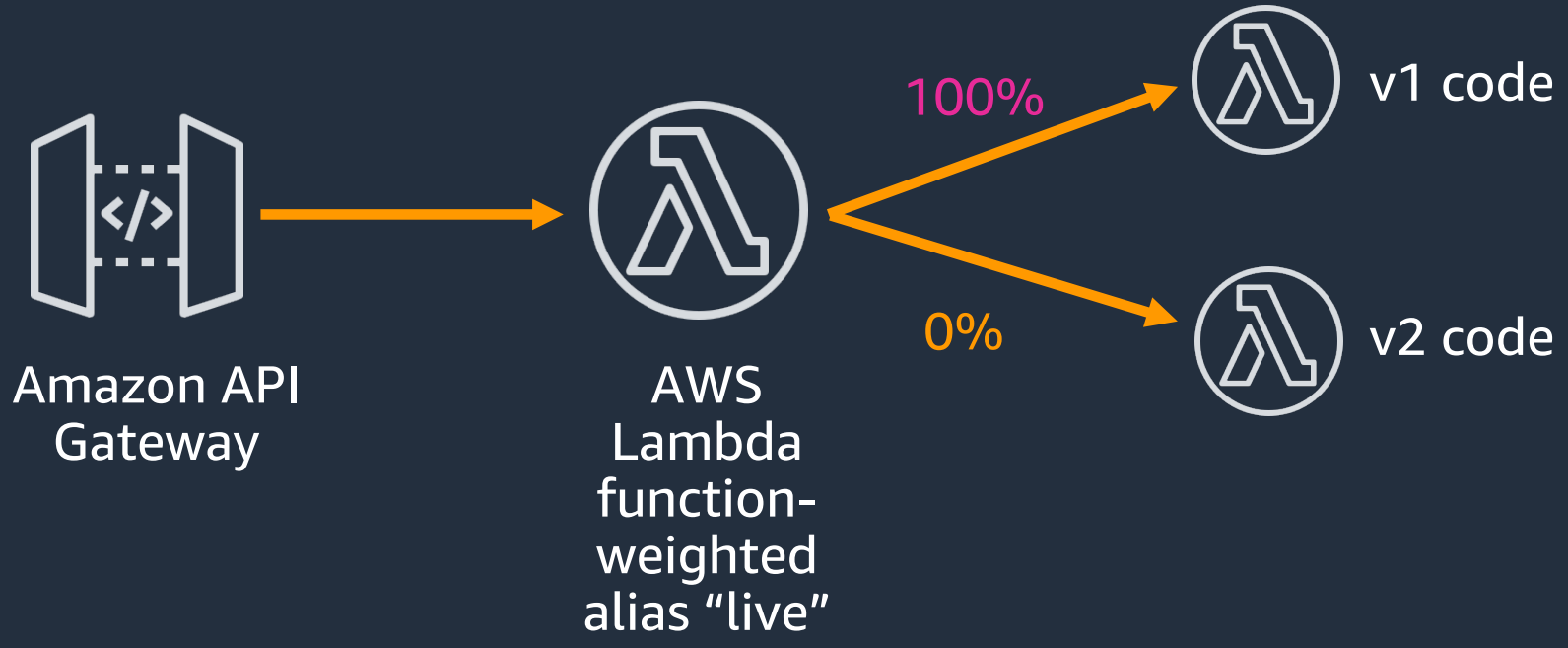
Hook スクリプト 実行可能箇所

- BeforeAllowTraffic
- AfterAllowTraffic

CodeDeploy Lambda カナリアデプロイメント

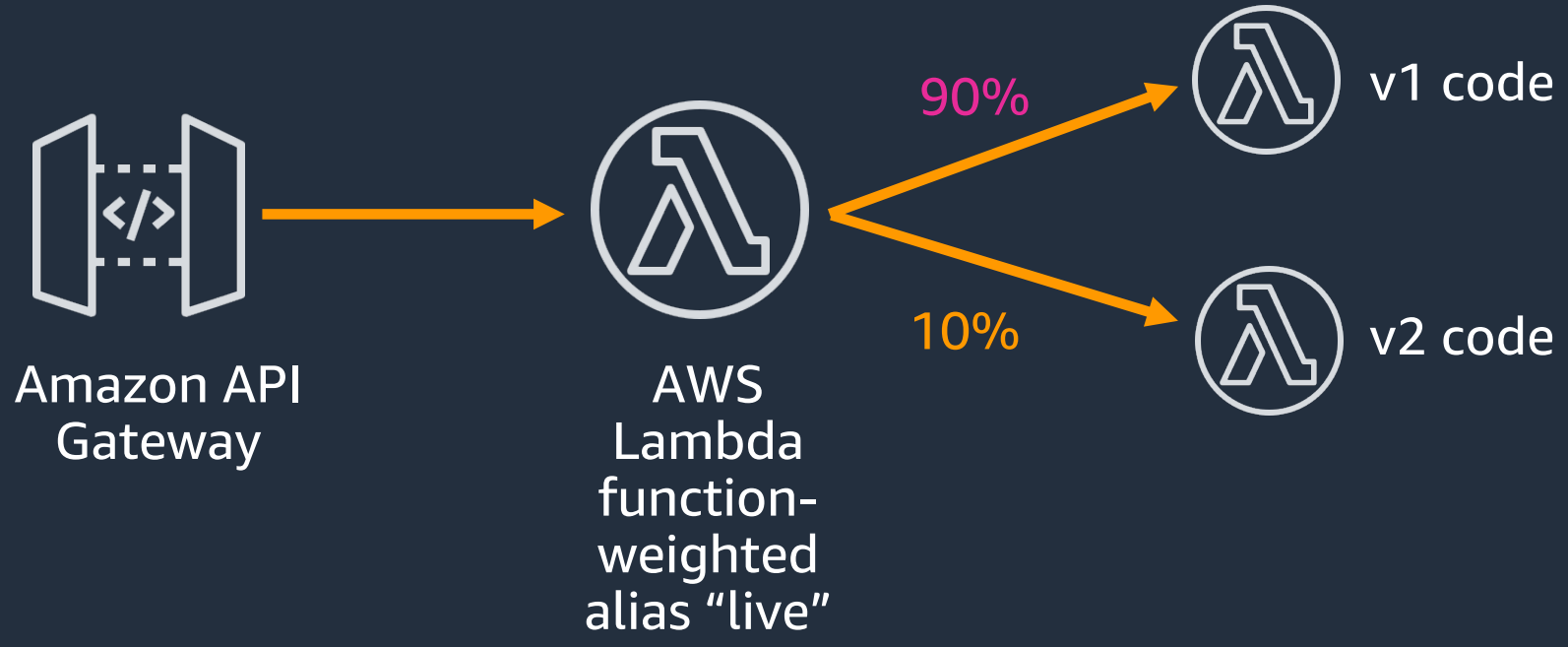
CodeDeploy Lambda カナリアデプロイメント

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

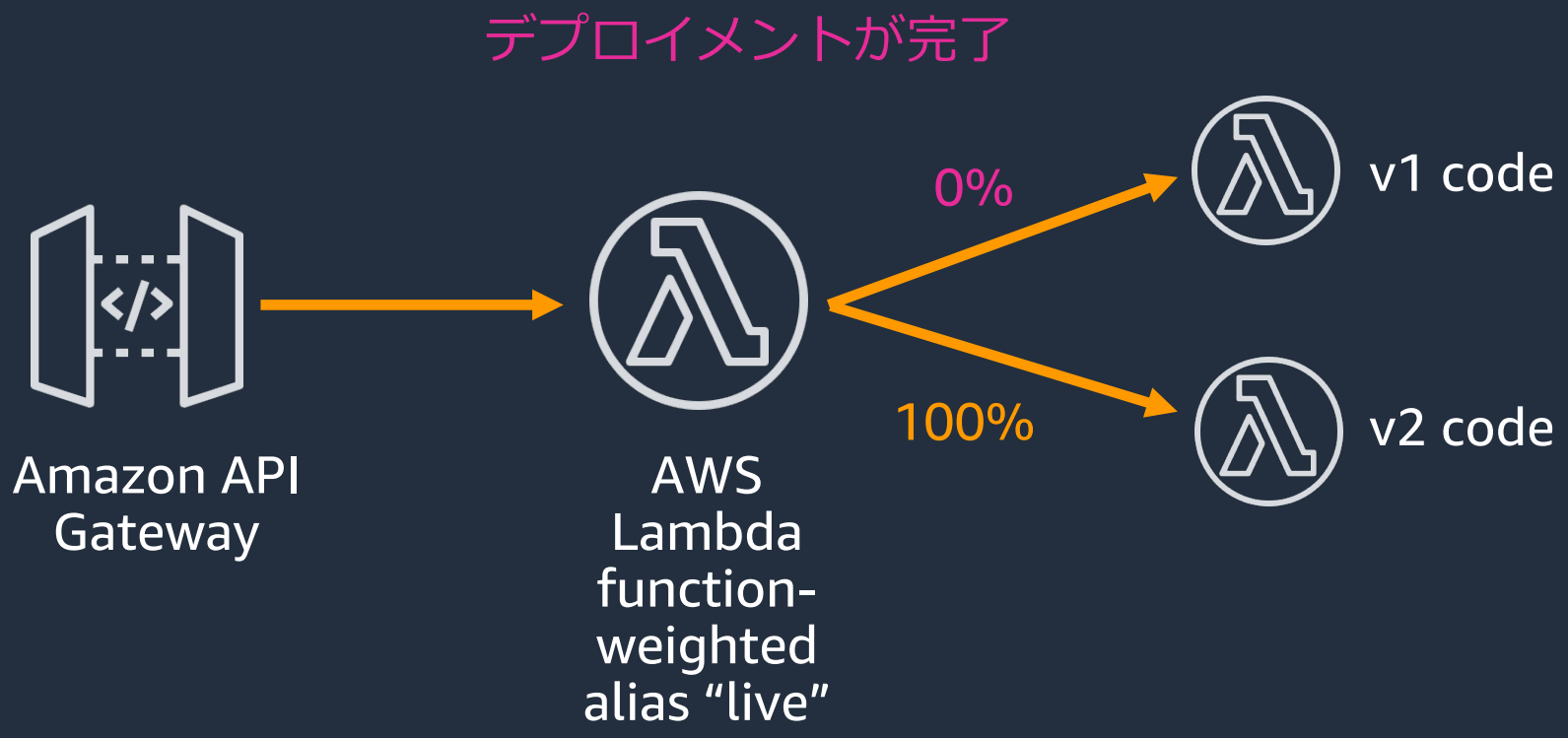


CodeDeploy Lambda カナリアデプロイメント

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



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

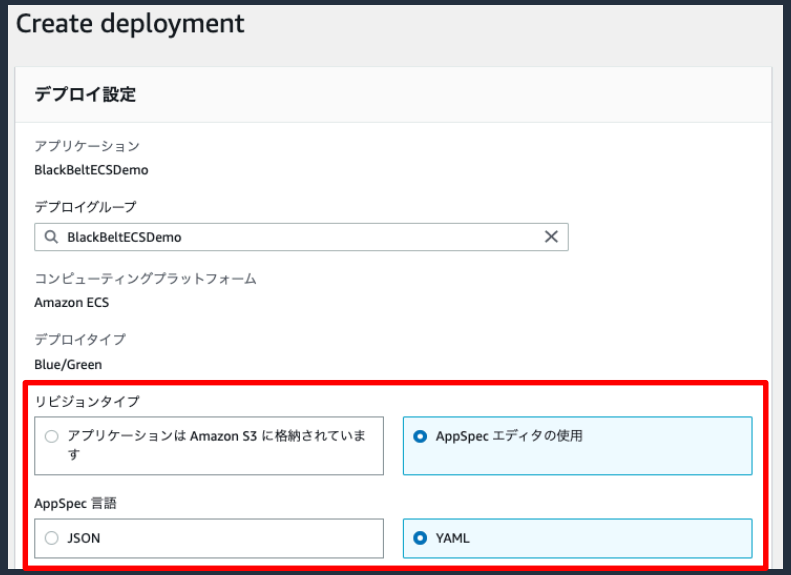
アプリケーションの作成

デプロイグループの作成

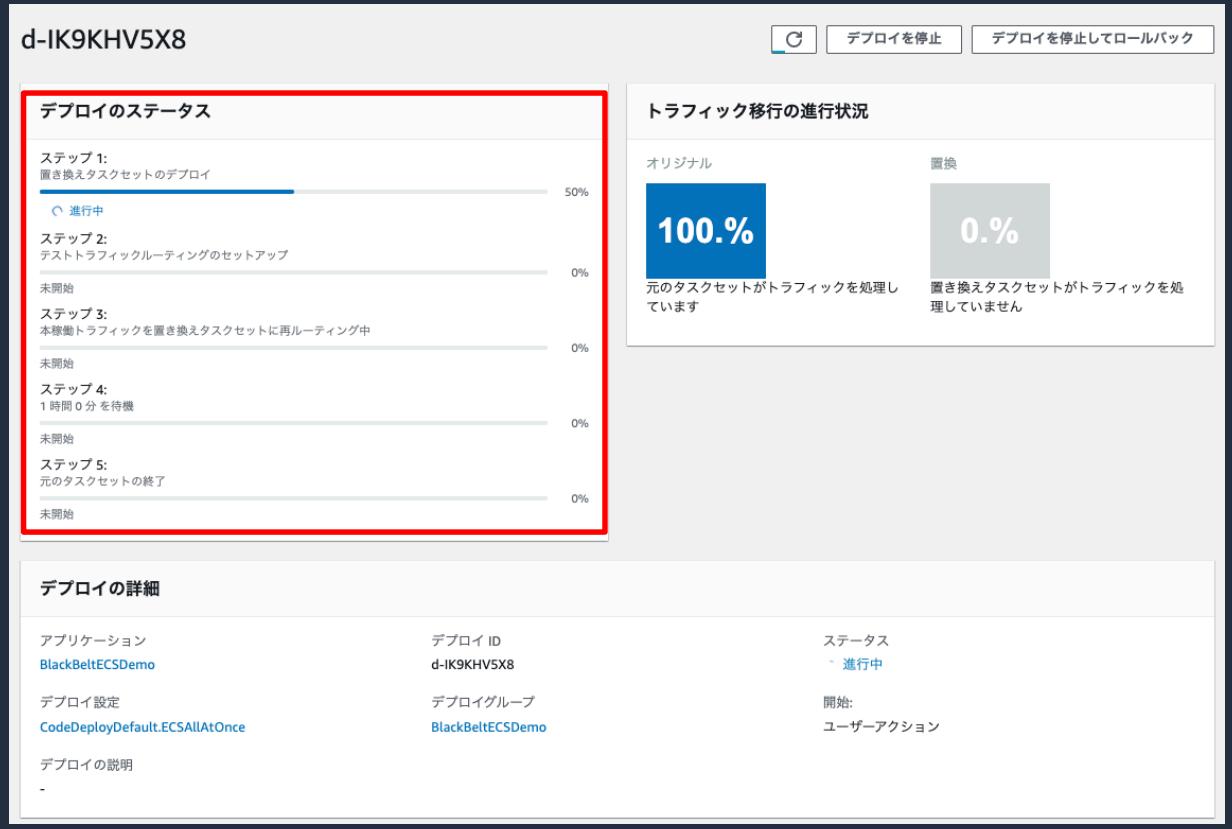
デプロイグループの作成

デプロイの作成

デプロイの作成



デプロイ実行



CodeDeploy ECS appspec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "my_task_definition:8"
        LoadBalancerInfos:
          ContainerName: "SampleApp"
          ContainerPort: 80
```

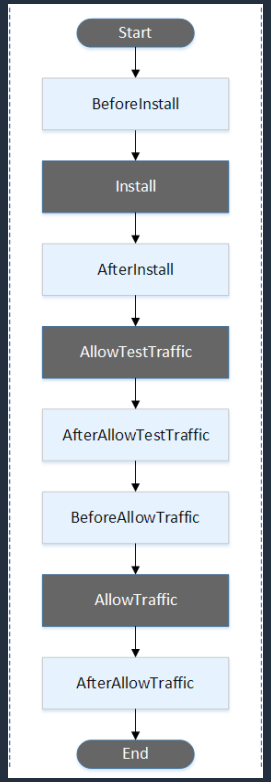
ターゲットタイプの指定
タスク定義
ロードバランサの定義

```
Hooks:
  - BeforeInstall:          "BeforeNewRevisionInstalltion"
  - 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

aws

デプロイライフサイクルイベント

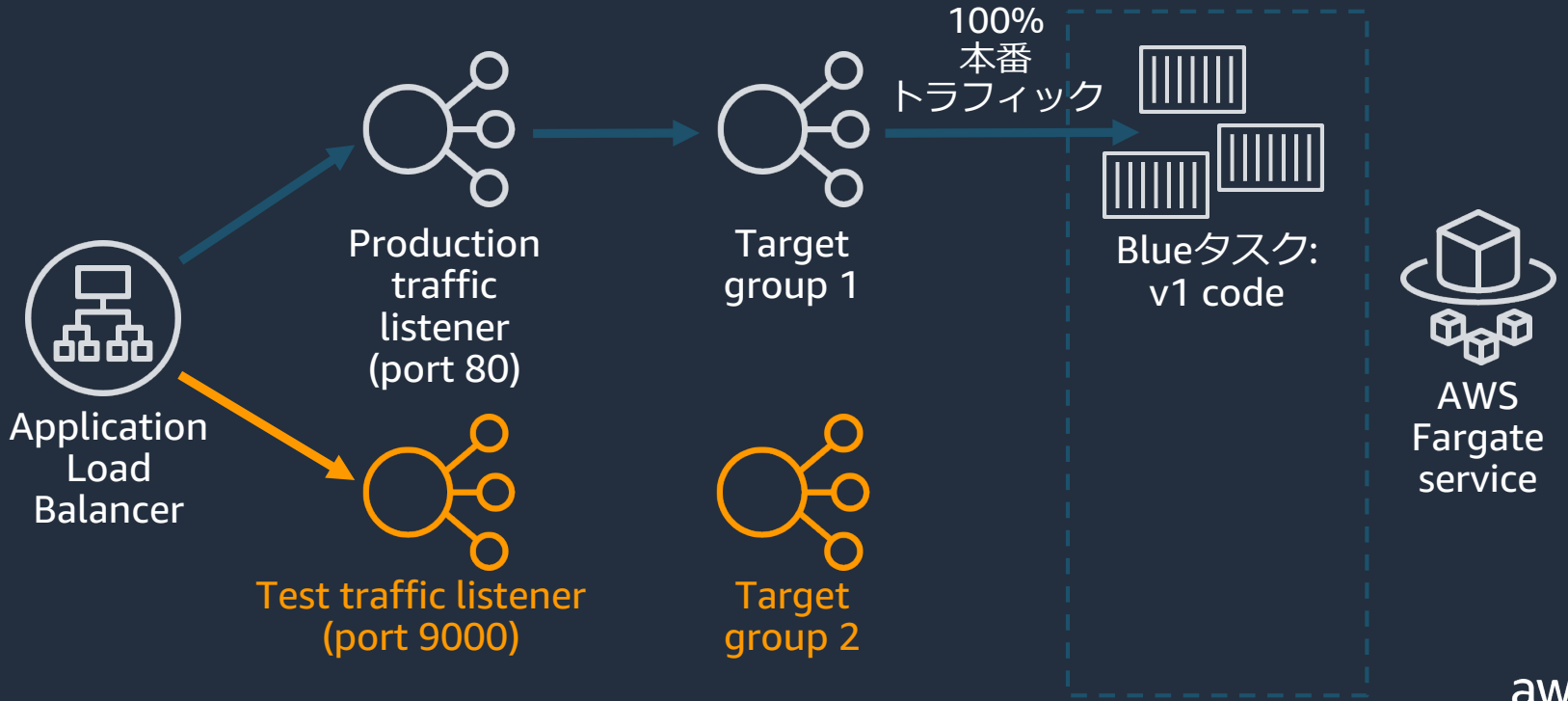


Hook スクリプト 実行可能箇所

- BeforeInstall
- AfterInstall
- AfterAllowTestTraffic
- BeforeAllowTraffic
- AfterAllowTraffic

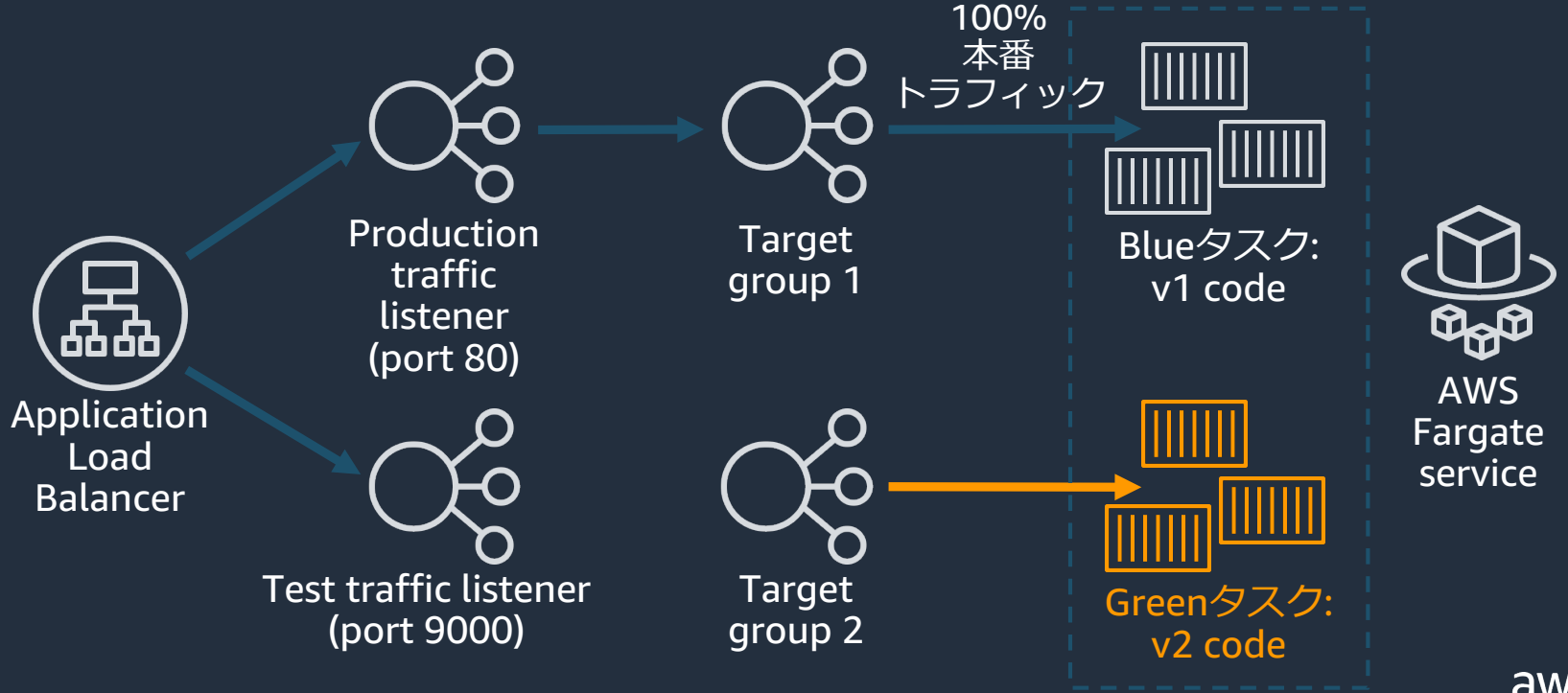
CodeDeploy ECS blue/greenデプロイメント

CodeDeploy ECS blue/greenデプロイメント



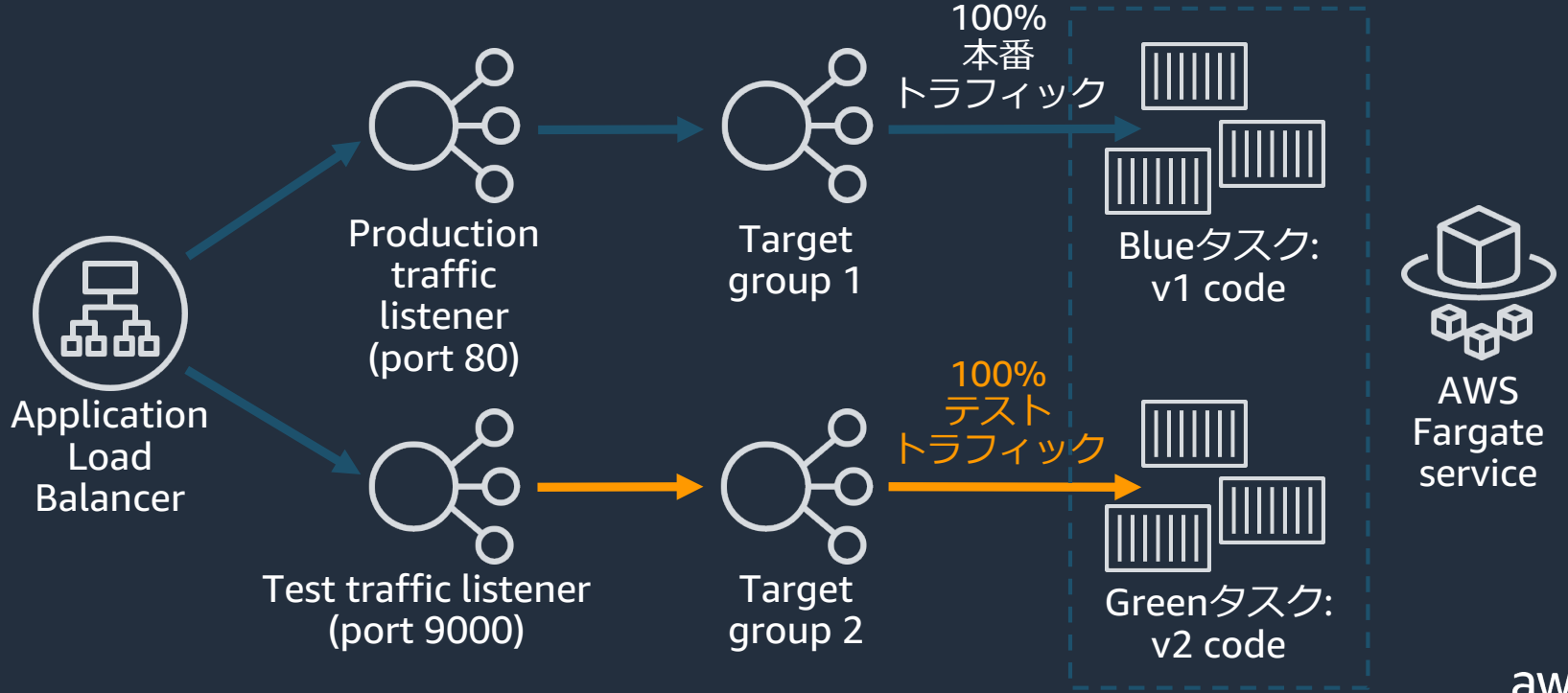
CodeDeploy ECS blue/greenデプロイメント

greenタスクを配置



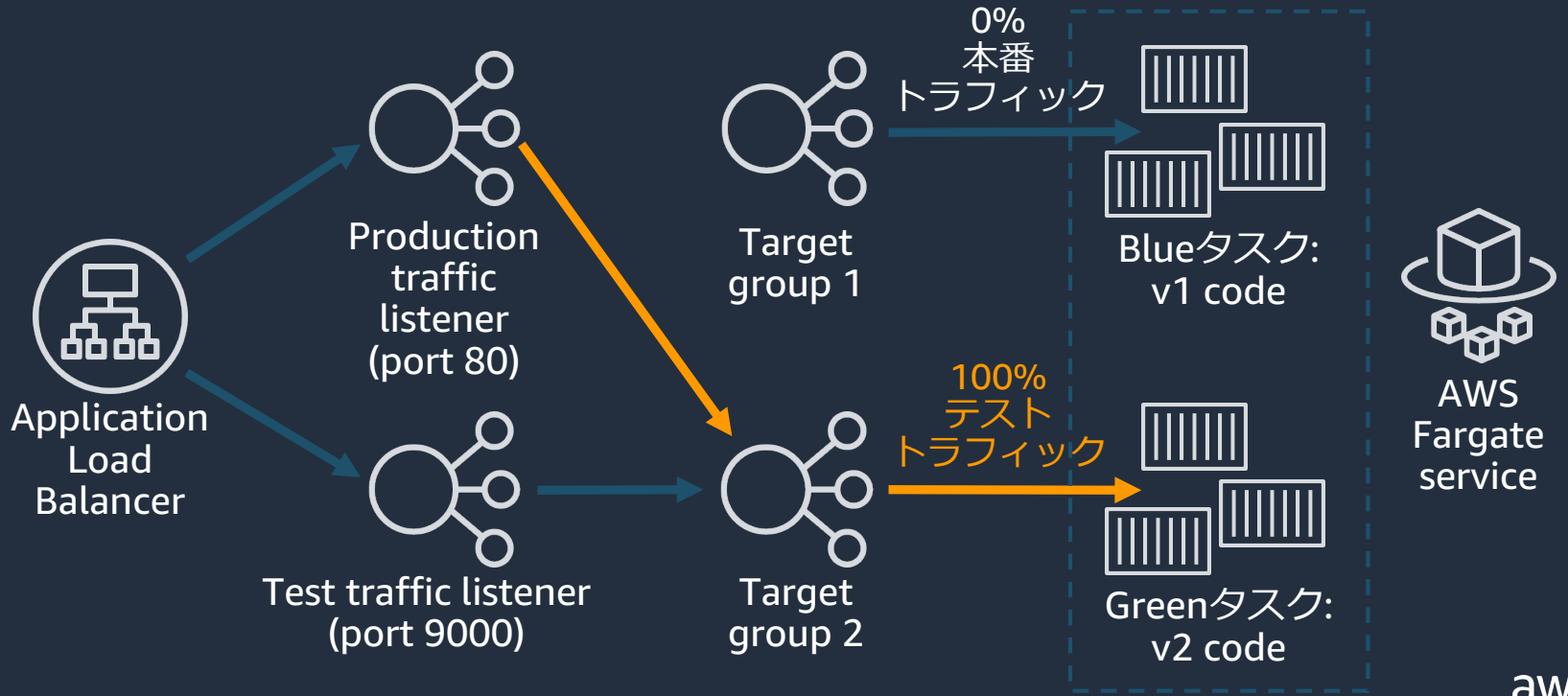
CodeDeploy ECS blue/greenデプロイメント

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

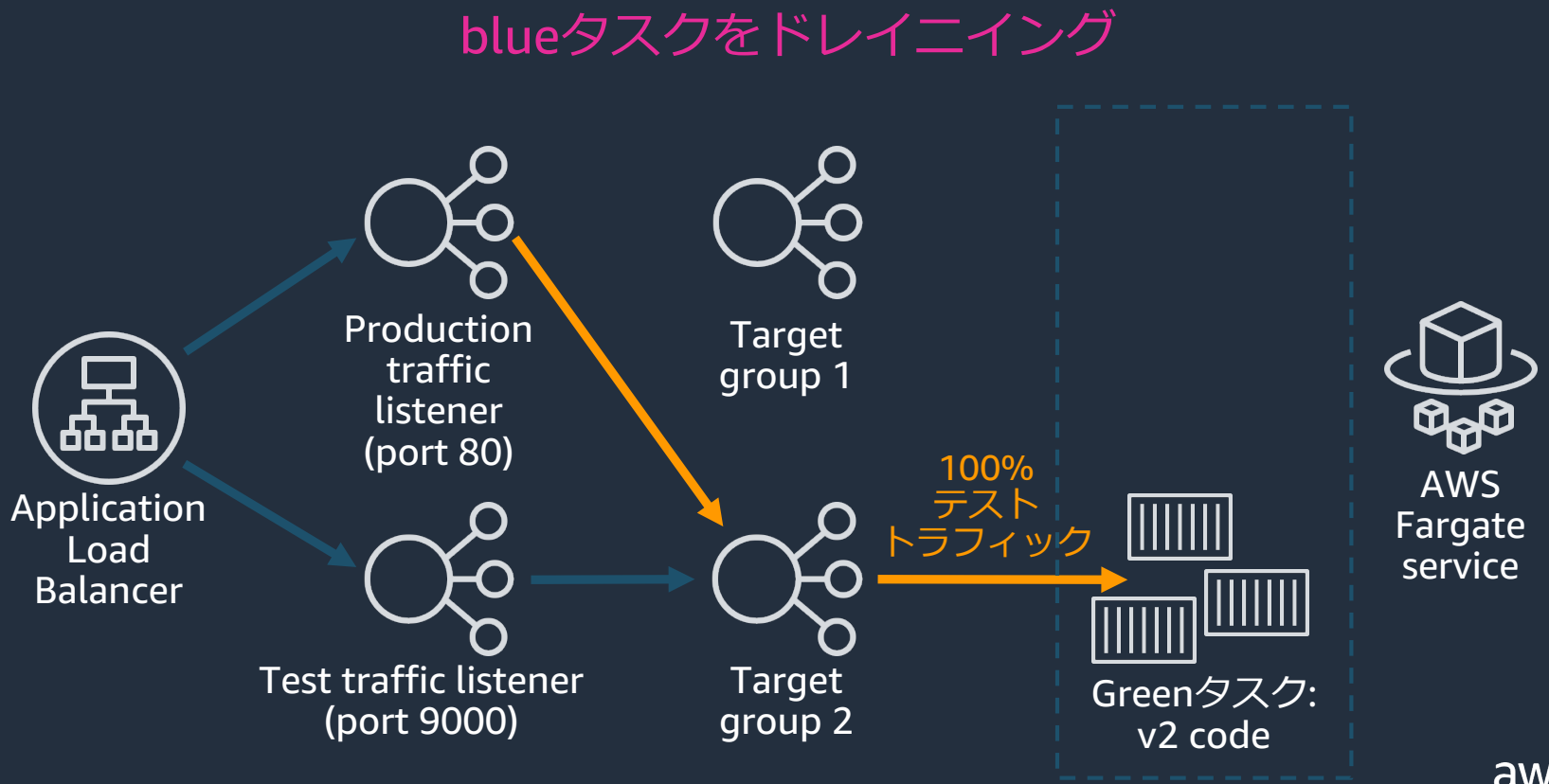


CodeDeploy ECS blue/greenデプロイメント

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



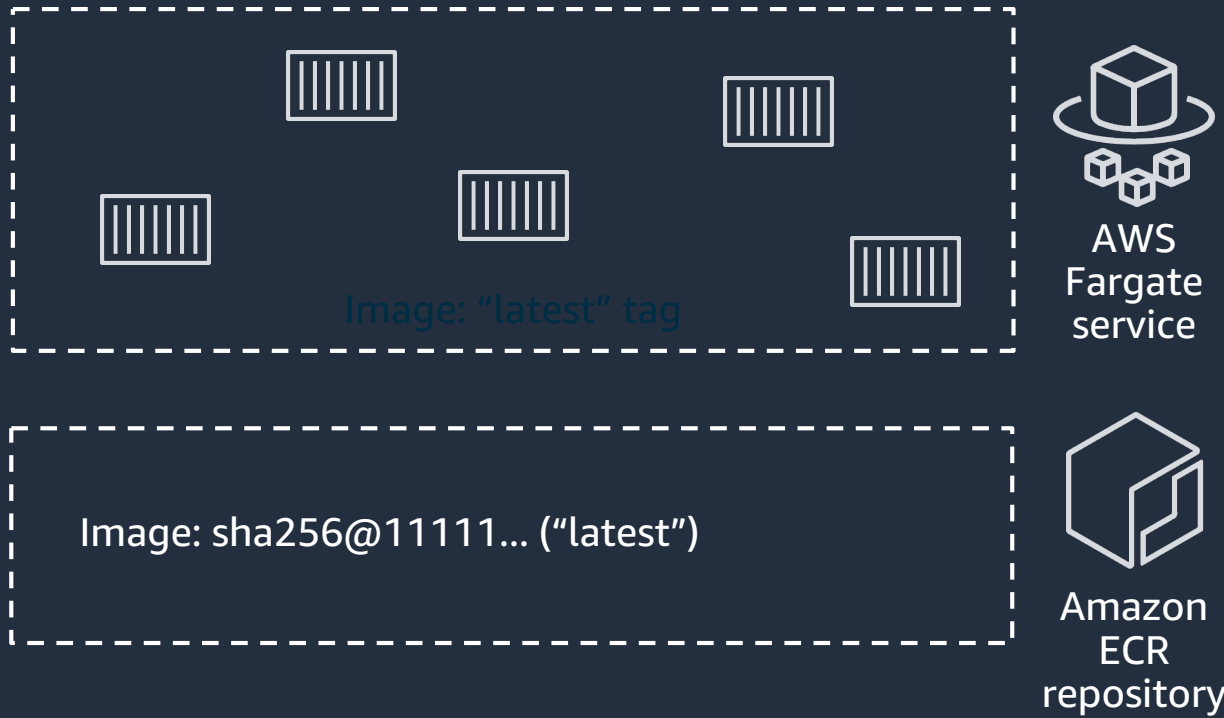
CodeDeploy ECS blue/greenデプロイメント



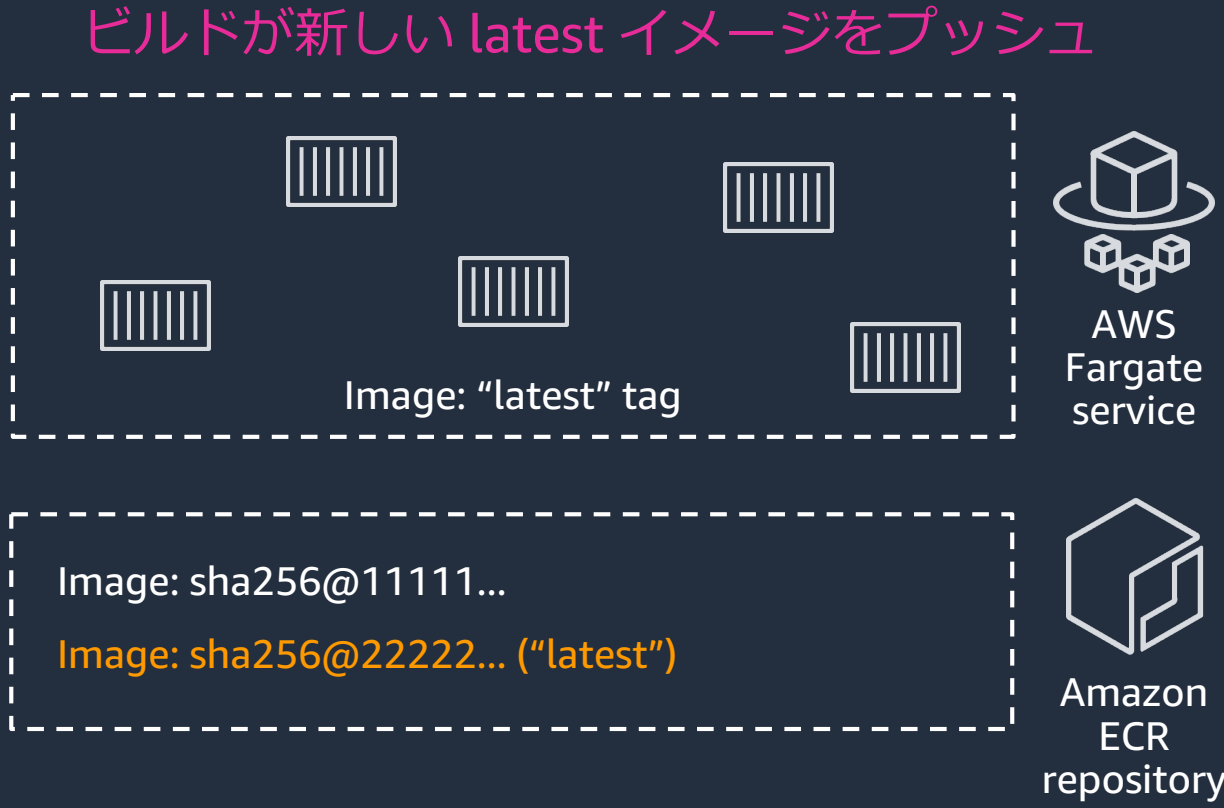
重要：デプロイメントのためのコンテナイメージのタグ付け

- 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



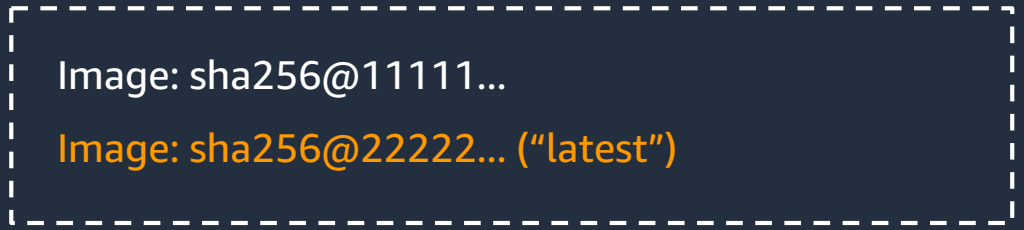
デプロイメントのためのコンテナイメージのタグ付け



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

aws

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
```


aws

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

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

AWS Fargate service

Image: sha256@11111... (“build-11111”)

Image: sha256@22222... (“build-22222”)

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

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

Image: sha256@11111... (“build-11111”)

Image: sha256@22222... (“build-22222”)

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

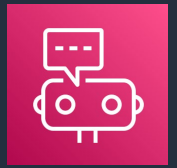
CloudWatch アラームによるデプロイ停止

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

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

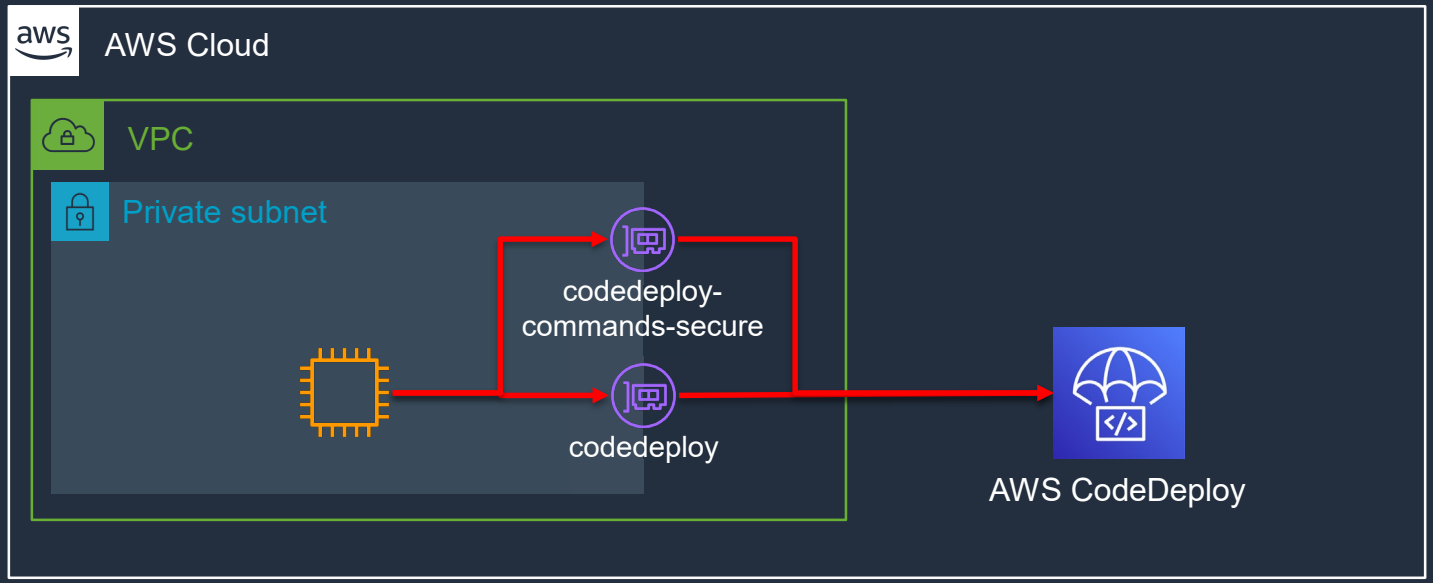
Amazon Simple Notification Service



AWS Chatbot

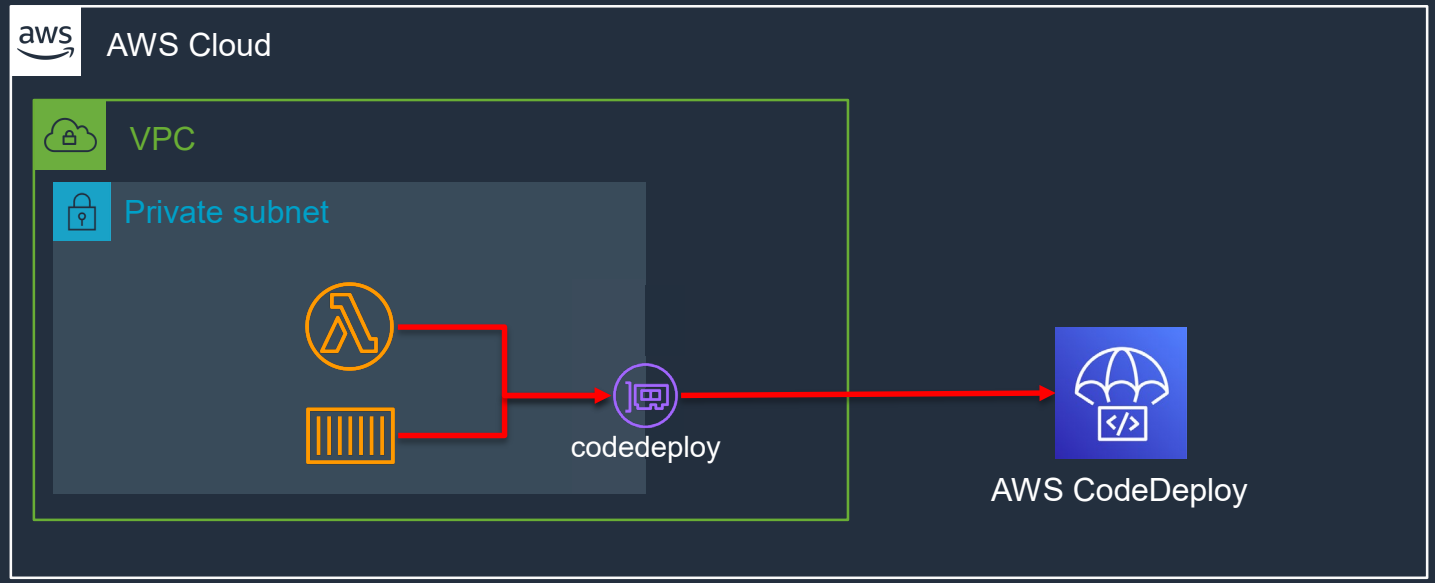
VPC エンドポイントをサポート

- 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



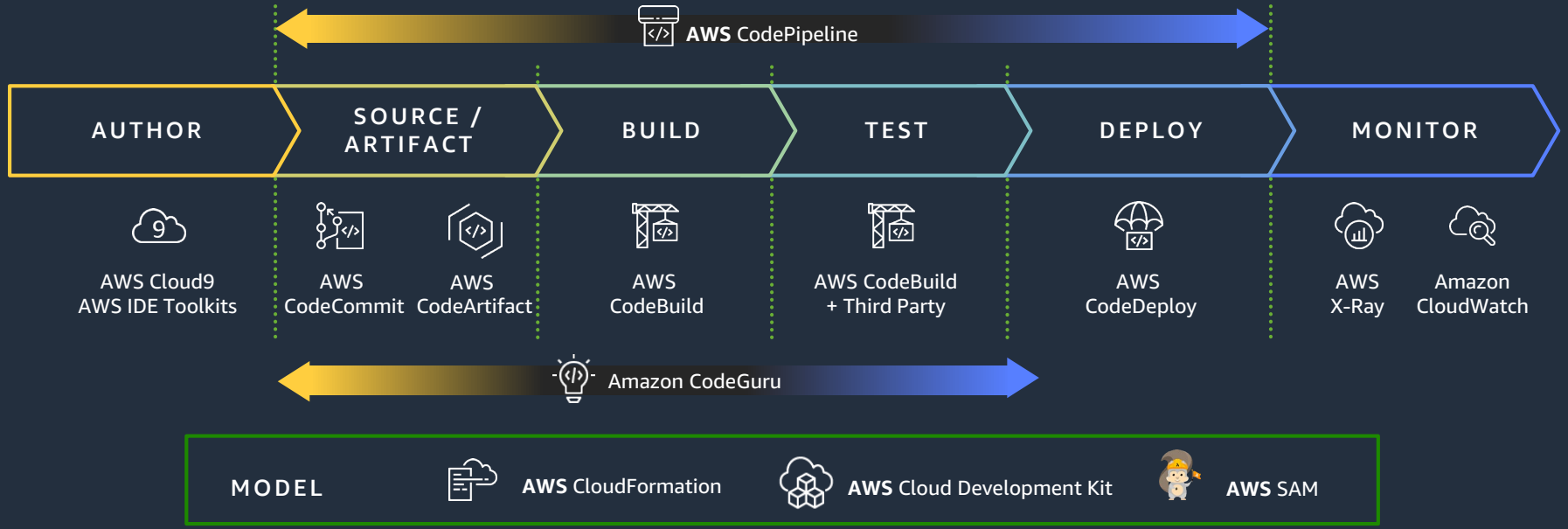
利用料金

- 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

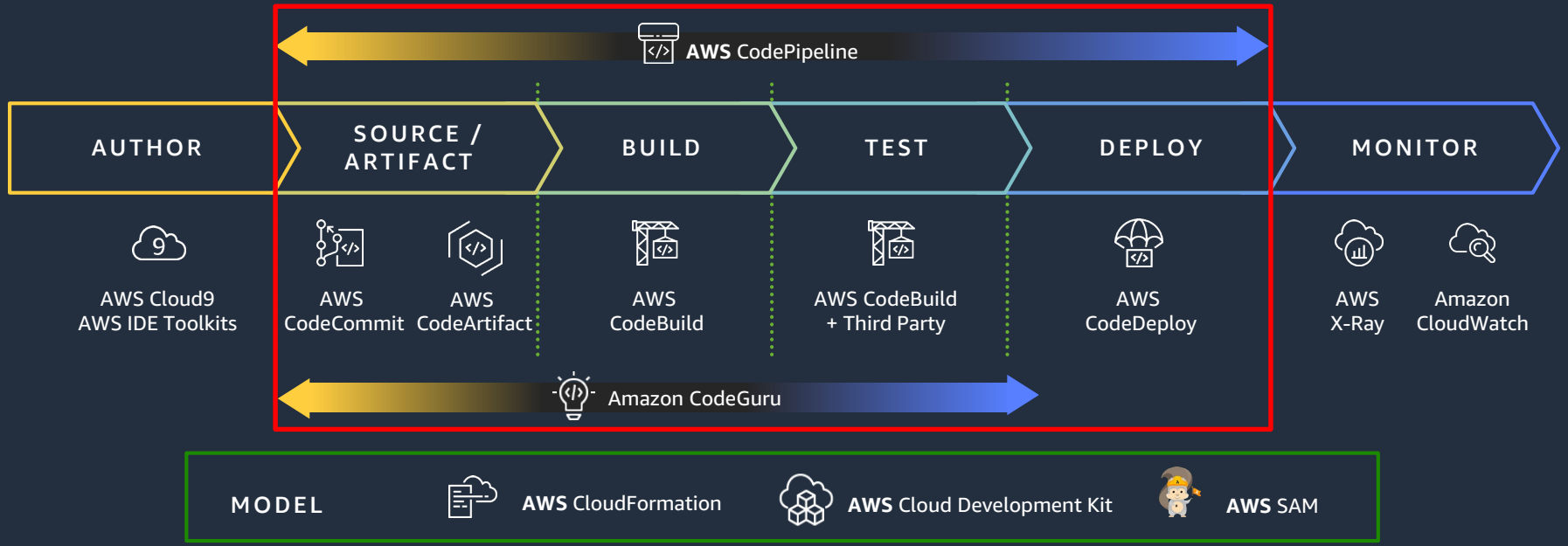
まとめ

- 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

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