

AWS

S U M M I T

AWSマネージドサービスで実現する CI/CDパイプライン

Yuki, Chiba, AWS Solutions Architect

2017. 6. 2





- ✿ Name : 千葉 悠貴 (ちば ゆうき)
- ✿ Role : Solutions Architect
- ✿ Segment : Internet Media
- ✿ Favorite Service : CloudFormation

本セッションのテーマ

継続的インテグレーション(CI)と継続的デリバリー(CD)

- ❖ 本セッションに含まれるもの
 - ❖ CI/CDを実現する“ツールとしての”AWSサービスの概要
 - ❖ 継続的デリバリーにフォーカスしたテクニックと実装例
- ❖ 本セッションに含まれないもの
 - ❖ What's CI/CD? Why CI/CD?
 - ❖ DevOpsの文化的な話
 - ❖ 取り上げる個々のAWSサービスの機能詳細

CI/CDパイプラインを実現する AWSのマネージドサービス

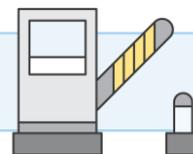
CI/CDパイプライン



継続的インテグレーション

継続的デリバリー

継続的デプロイメント



CI/CDパイプラインを実現するAWSサービス

ソースコードの
バージョン管理



AWS CodeCommit

ビルド自動化



AWS CodeBuild

デプロイ自動化



AWS CodeDeploy



AWS CloudFormation



AWS Elastic Beanstalk

ワークフロー管理



AWS CodePipeline



AWS CodeStar

AWS Code Services



AWS CodeCommit

- ❖ セキュア、スケーラブルなGit互換のリポジトリサービス
 - ❖ スタンダードなGit Toolからアクセス可能
 - ❖ PushなどのイベントをトリガーにSNS/Lambdaを呼び出し可能
-



AWS CodeBuild

- ❖ スケーラビリティに優れたビルドサービス
 - ❖ ソースのコンパイル、テスト、パッケージ生成をサポート
 - ❖ Dockerイメージの作成も可能
-



AWS CodeDeploy

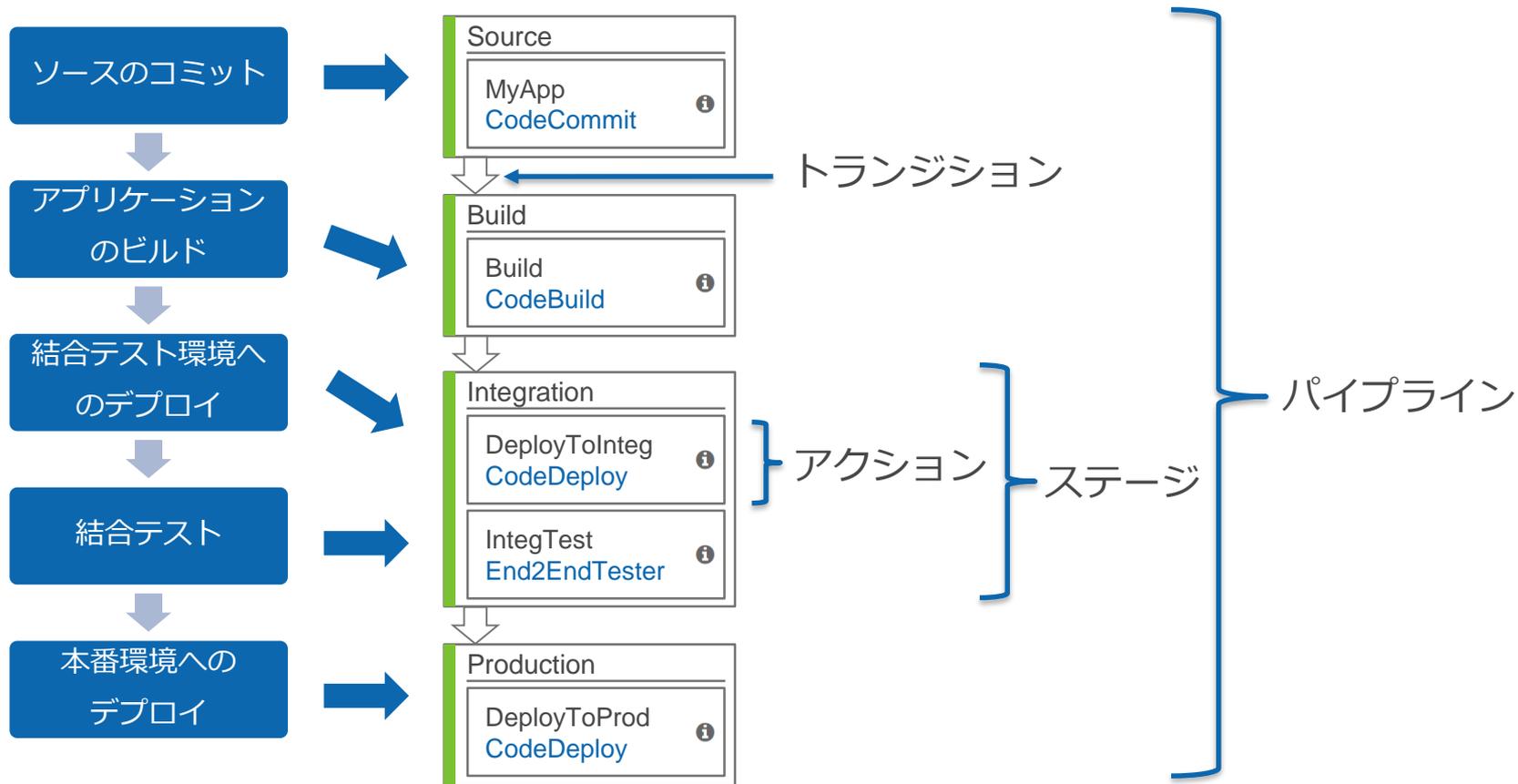
- ❖ S3またはGitHub上のコードをあらゆるインスタンスにデプロイ
 - ❖ デプロイを安全に実行するための様々な機能を提供
 - ❖ In-place(ローリング) およびBlue/Greenのデプロイをサポート
-



AWS CodePipeline

- ❖ リリースプロセスのモデル化と見える化を実現
 - ❖ カスタムアクションによる柔軟なパイプライン作成が可能
 - ❖ 様々なAWSサービスや3rdパーティ製品との統合をサポート
-

CodePipelineでのリリースプロセスモデル化





CICDDemoPipeline

View progress and manage your pipeline.

[Edit](#) [Release change](#)

Source

Source ⓘ
AWS CodeCommit

✔ **Succeeded** 3 hours ago
5dc0200

Source: version 1.0

Build

CodeBuild ⓘ
AWS CodeBuild

✔ **Succeeded** 3 hours ago
[Details](#)

Source: version 1.0

Integration

DeployToInteg ⓘ
AWS CodeDeploy

✔ **Succeeded** 3 hours ago
[Details](#)

AWS CodeStar

AWS CodeStar

AWS上にアプリケーションをすばやく開発・ビルド・デプロイ



AWS上での開発をわずか数分間で開始

チームをまたがった開発をセキュアに

ソフトウェアデリバリの管理を容易に

様々なプロジェクトテンプレートから選択

Step1 : プロジェクトテンプレート選択

AWS CodeStar ▶ Create project



Filter

Application category

- Web application
- Web service
- Static Website

Programming languages

- Ruby
- Node.js
- Java
- Python
- PHP
- HTML 5

AWS services

- AWS Elastic Beanstalk
- Amazon EC2
- AWS Lambda

Choose a project template

Start a new software project on AWS in minutes using a project template. [Help me choose](#)

<h3>Ruby on Rails</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)	<h3>Ruby on Rails</h3> <ul style="list-style-type: none">Web applicationAmazon EC2 (runs on virtual servers that you manage)	<h3>Java Spring</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)	<h3>Java Spring</h3> <ul style="list-style-type: none">Web applicationAmazon EC2 (runs on virtual servers that you manage)
<h3>Node.js</h3> <ul style="list-style-type: none">Web applicationAWS Lambda (running serverless)	<h3>Node.js</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)	<h3>Node.js</h3> <ul style="list-style-type: none">Web applicationAmazon EC2 (runs on virtual servers that you manage)	<h3>Python (Django)</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)
<h3>Python (Django)</h3> <ul style="list-style-type: none">Web applicationAmazon EC2 (runs on virtual servers that you manage)	<h3>Express.js</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)	<h3>Express.js</h3> <ul style="list-style-type: none">Web applicationAmazon EC2 (runs on virtual servers that you manage)	<h3>PHP (Laravel)</h3> <ul style="list-style-type: none">Web applicationAWS Elastic Beanstalk (runs in a managed application environment)
<h3>PHP (Laravel)</h3>	<h3>Ruby (Sinatra)</h3>	<h3>Ruby (Sinatra)</h3>	<h3>Java Spring</h3>

Step1 : プロジェクトテンプレート選択

AWS CodeStar ▶ Create project



Project name

AWS SF Summit

Project ID ⓘ

aws-sf-summit

[Edit](#)

AWS CodeStar includes all of the tools and services you need for a development project.

This project includes an AWS CodePipeline connected with the following tools:



Source



Build



Test



Deploy



Monitoring



AWS CodeCommit



AWS CodeBuild



AWS CloudFormation



Amazon CloudWatch

AWS CodeStar would like permission to administer AWS resources on your behalf. [Learn more](#)

[Previous](#)

[Create Project](#)

Step2 : 開発ツールセットアップ

AWS CodeStar ▶ Create project



Provisioning 0% complete

Choose how you want to edit your project code

You can always change this choice after the project has been created



Visual Studio

Configure the AWS Toolkit for Visual Studio to edit your AWS CodeStar project code in Microsoft Visual Studio 2015 (or higher.)



Eclipse

Configure the AWS Toolkit for Eclipse to edit your AWS CodeStar project code in Eclipse.



Command line tools

Edit AWS CodeStar project code by connecting directly to your project's Git source repository.

Clone repository URL

HTTPS ▾



We're creating your repository. It should be ready soon.

Copy

[Credential details](#)

Previous

Skip

Complete!

AWS CodeStar ▶ AWS SF Summit

Important: You must connect to your project's repository before you can start working on the code. [I have already done this](#) [Connect Tools](#)

Add tile

Welcome to AWS SF Summit! Close

Let us help you get started.



[Learn about AWS CodeStar](#)



[Setup your team](#)



[Configure issue tracking](#)

Team wiki tile

Edit this tile to save your own project links, code samples and notes to share with your team. You can use [markdown](#) to **format** your text.

Some other things to try in your project...

- [Access your application](#)
- Read "What do I do next?" in README.md in project source repository
- [Add team members](#)
- Setup issue tracking (under "Extensions")
- [Customize project dashboard](#)
- [View AWS CodeStar documentation](#)
- [Visit the AWS CodeStar Forum](#)

Commit history: aws-sf-summit master

Application endpoints

<https://pq2a7xcdh0.execute-api.us-west-2.amazonaws.com/Prod/>

Continuous deployment AWS CodePipeline

[Release change](#)

Source

4/18/2017, 5:07:03 PM
ApplicationSource CodeCommit
Succeeded

[Commit history](#)

セットアップ内容

- ❖ サンプルアプリケーション
 - ❖ EC2 or Beanstalk or Lambda
- ❖ CodeCommitリポジトリ
- ❖ CodeBuildビルドプロジェクト
- ❖ デプロイツール
 - ❖ CodeDeploy or Beanstalk or CloudFormation
- ❖ CodePipeline 継続的デプロイメントパイプライン
- ❖ CloudWatchメトリクス
- ❖ プロジェクトダッシュボード



Project Dashboard

AWS CodeStar ▶ SampleProject

Dashboard

Code

Build

Deploy

Pipeline

Team

Extensions

Project

Add tile

Commit history: sampleproject

master

CodeCommit

- version 1.0 release
Chiba committed 5 days ago
5e5e79d
- test commit
Chiba committed 5 days ago
8947424
- Initial commit of sample code made during project creation in AWS CodeStar
AWS CodeStar committed 5 days ago
d7bd8cd

Connect

AWS CodeCommit details

Application endpoints

<http://sampleprojectapp.minjvawq4m.us-east-1.elasticbeanstalk.com>

Appエンドポイント

Application activity

CPUtilization

Amazon CloudWatch

CloudWatchメトリクス

Amazon CloudWatch details

JIRA

Track work items and issues for your AWS CodeStar projects with Atlassian JIRA integration.

JIRA統合メニュー

Connect

Continuous deployment

AWS CodePipeline

Release change

Source

2017/5/22 20:03:21
ApplicationSource CodeCommit
Succeeded

Commit history

Build

2017/5/22 20:05:37
CodeBuild CodeBuild
Succeeded

Application

2017/5/22 20:06:45
EBStack ElasticBeanstalk
Succeeded

CodePipeline

Endpoint(1)

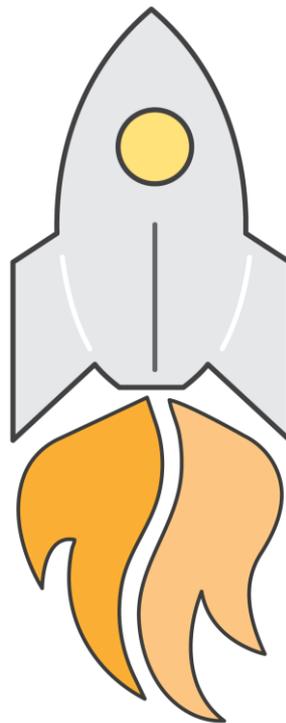
AWSサービスを利用した 継続的デリバリーテクニック

継続的デリバリー？

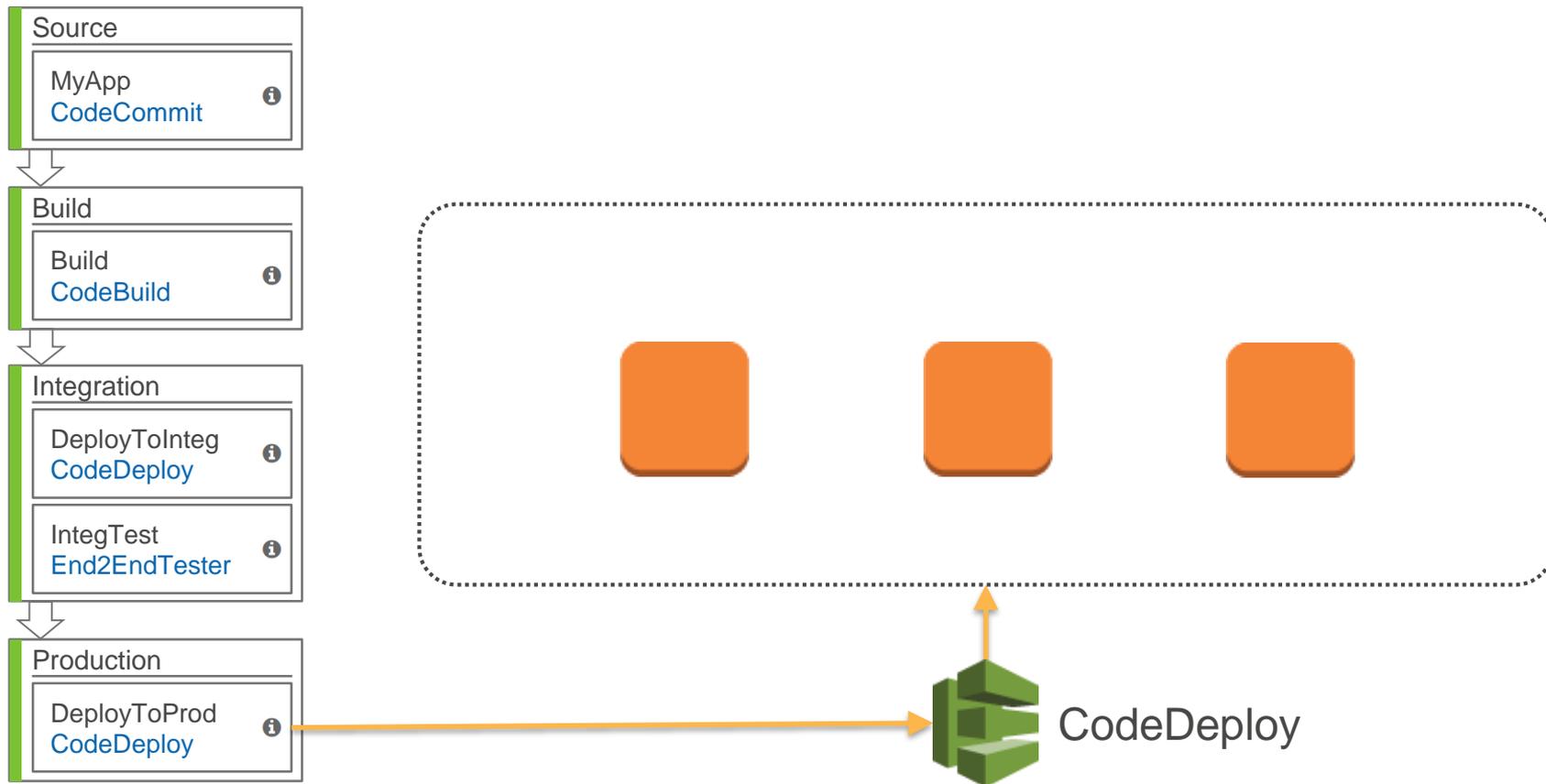
- ❖ プロダクション環境 = ビジネスをヘルシーに保つ
 - ❖ リリース作業による問題を減らす = 安全にデプロイする
 - ❖ 問題が起きた際は素早く検知し影響範囲を少なくする
- ❖ リリース判断 / タイミングをビジネスに合わせる
 - ❖ ビジネス影響が大きいリリースはマニュアル承認ステップを設ける
 - ❖ ビジネス判断でリリースを遅らせることも
- ❖ 継続的デリバリー ⇒ 継続的デプロイメント
 - ❖ いつでもプロダクションリリース可能な状態にしておく
 - ❖ ビルド～テスト～デプロイの自動化は必須事項
 - ❖ 安全性が担保されたらマニュアルステップを自動化する

継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート

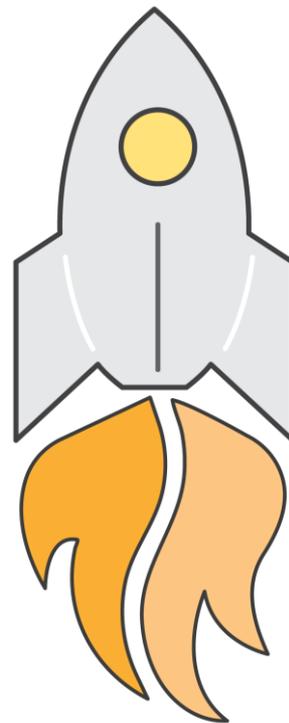


リリースプロセス：ベースライン



継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート



本番環境の継続的な監視

Requirement

本番環境のサービス停止を意識する

- ❖ サービスは内部起因／外部起因でいつでも停止し得る
- ❖ 内部監視では健全でもユーザーからアクセスできない可能性も

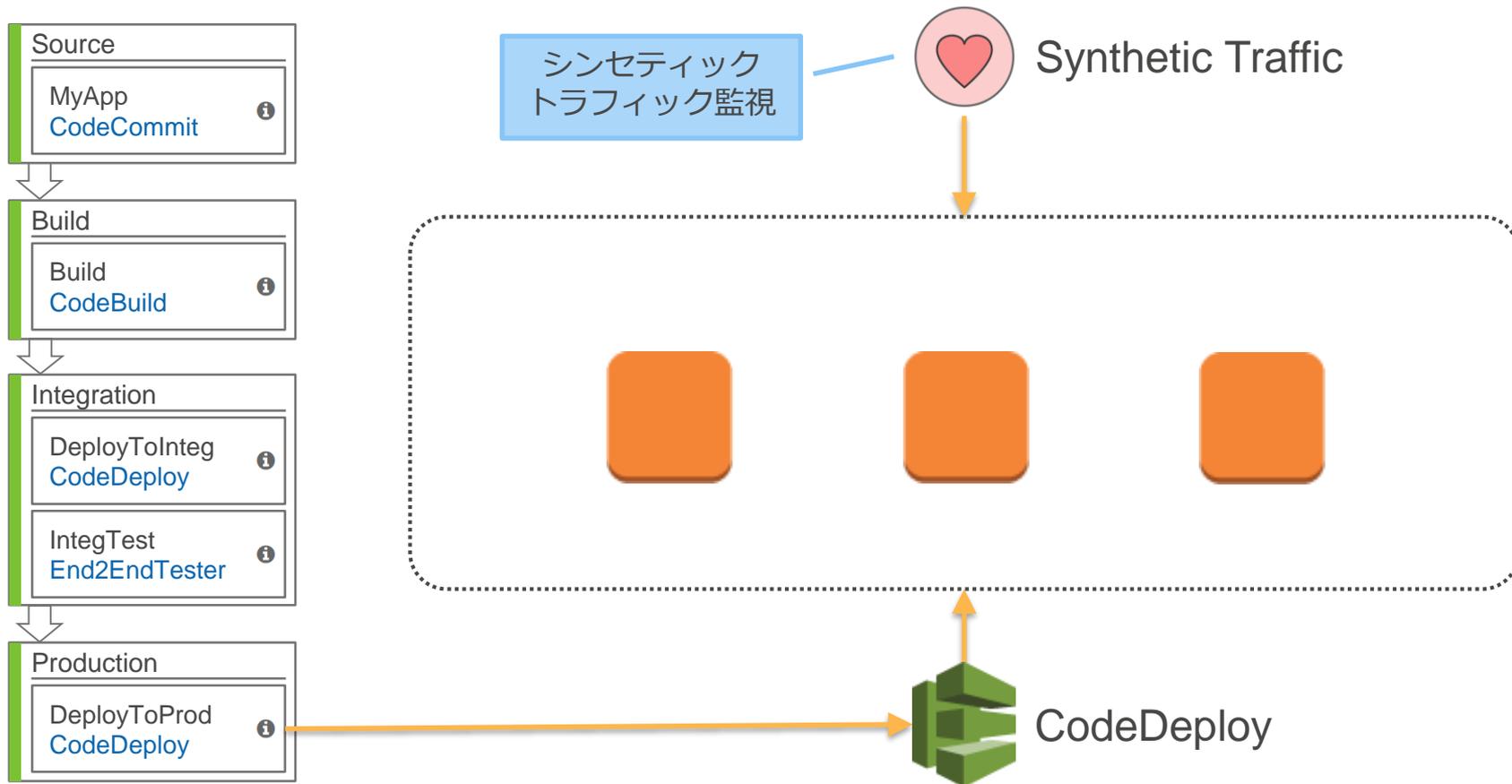
Implement

シンセティックトラフィックで本番環境を常に監視をする

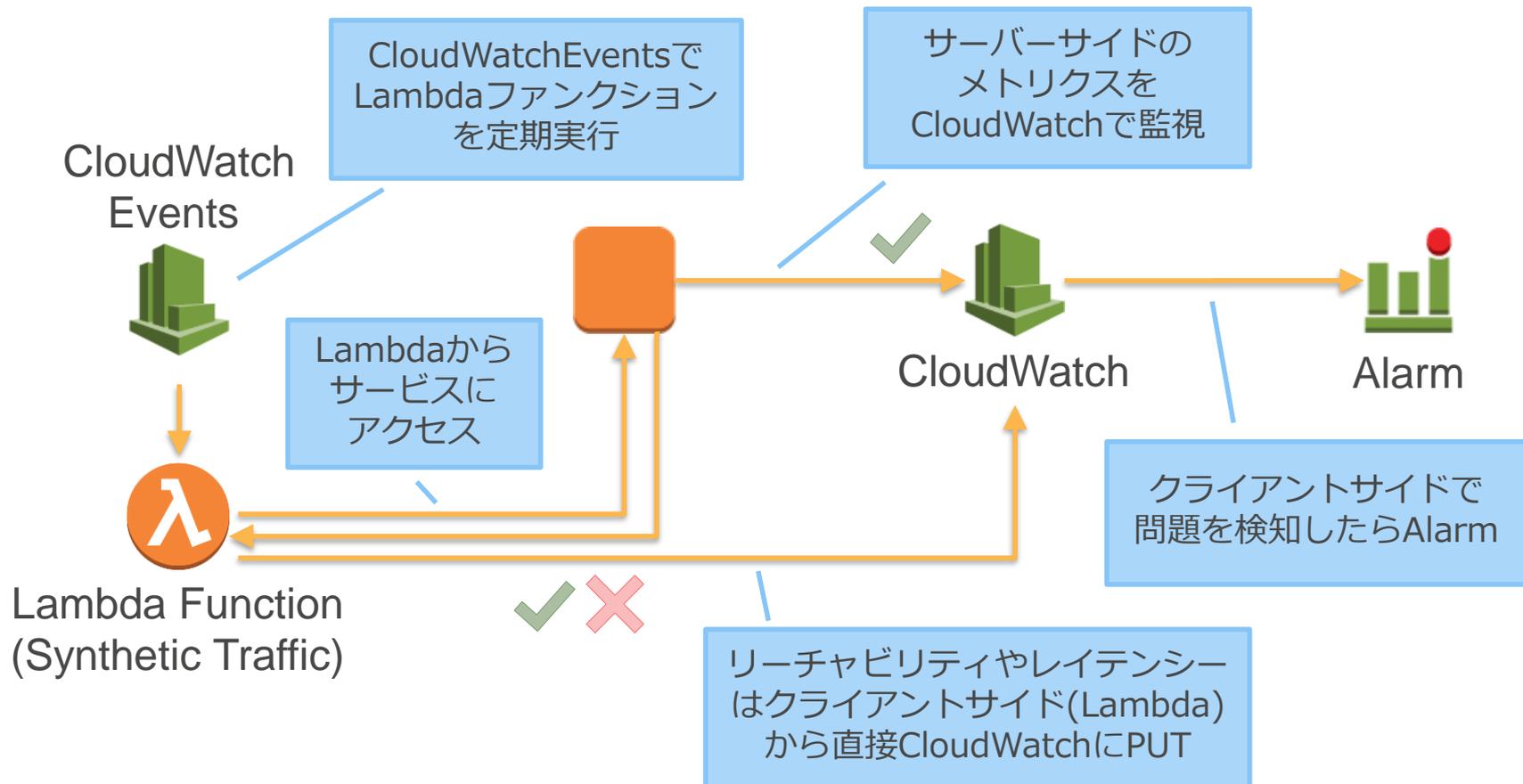
- ❖ 接続性のチェック
- ❖ クライアントレイテンシーの測定
- ❖ 重要なビジネスファンクション(API/UI)のテスト

シンセティックトラフィック=リアルユーザーをシミュレートしたトラフィック

リリースプロセス + シンセティックトラフィック

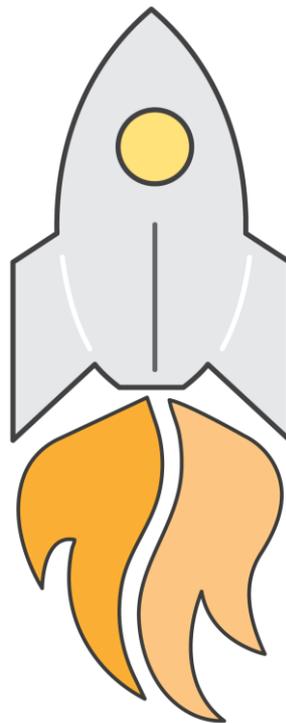


Reference Architecture – シンセティックトラフィック



継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート



デプロイメントヘルスチェック

Requirement

デプロイメント時の失敗・問題をすばやく検知する

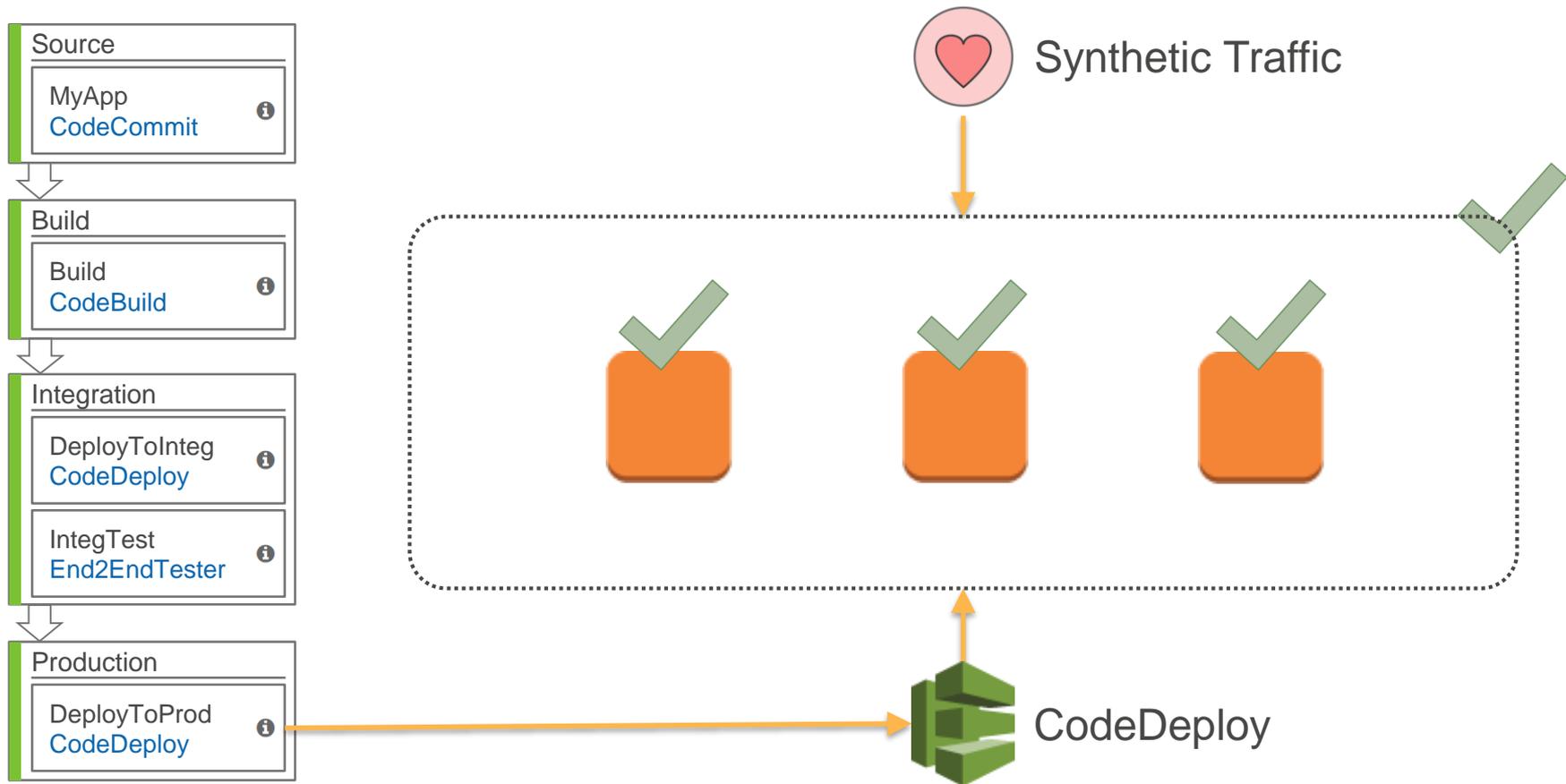
- ❖ ユニットテストが正常でもデプロイ時に異常になるケースもあり得る
- ❖ 単一ホストの問題か、全体に波及する問題かを区別する

Implement

デプロイ時のヘルスチェックとロールバックを実装する

1. 各ホストごとにデプロイ後のセルフヘルスチェックを実施する
2. 問題のあるホスト数が閾値を超えた場合デプロイを失敗にする
3. デプロイ失敗時はロールバックする

リリースプロセス + デプロイヘルスチェック

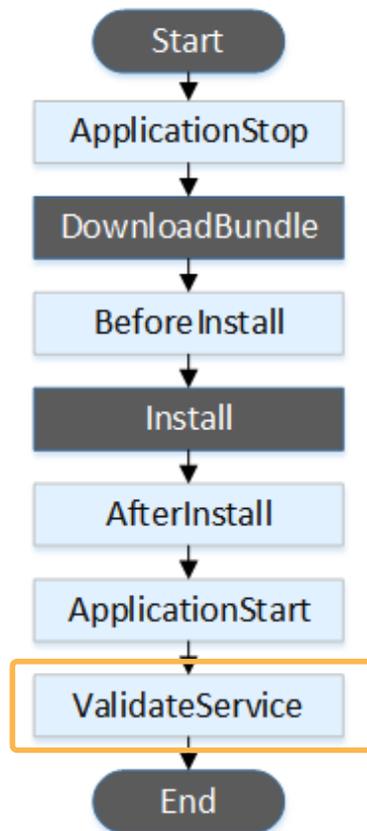


CodeDeploy - ValidateServiceフック

- ❖ CodeDeployではValidateServiceフックを利用してアプリ開始後にセルフヘルスチェックスクリプトの実行が可能

```
version: 0.0
os: linux
files:
- source: source
  destination: /webapps/my_app
hooks:
  ApplicationStart:
  - location: scripts/start.sh
    timeout: 3600
  ValidateService:
  - location: scripts/validate_running_service.sh
    timeout: 3600
    runas: codedeployuser
```

CodeDeploy設定ファイル : AppSpec.yml



CodeDeploy - Minimum Healthy Hosts

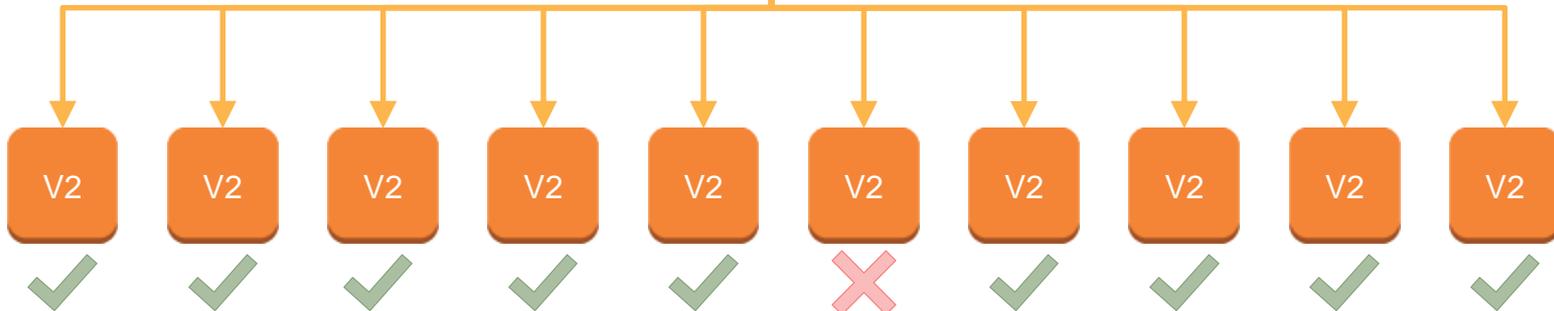
- ❖ デプロイ対象全体の何割が正常であればデプロイを成功とみなすかを定義できる

MHH 66%, 10 hosts:

1 failure – 90% healthy



ELB



Production Fleet

CodeDeploy - Minimum Healthy Hosts

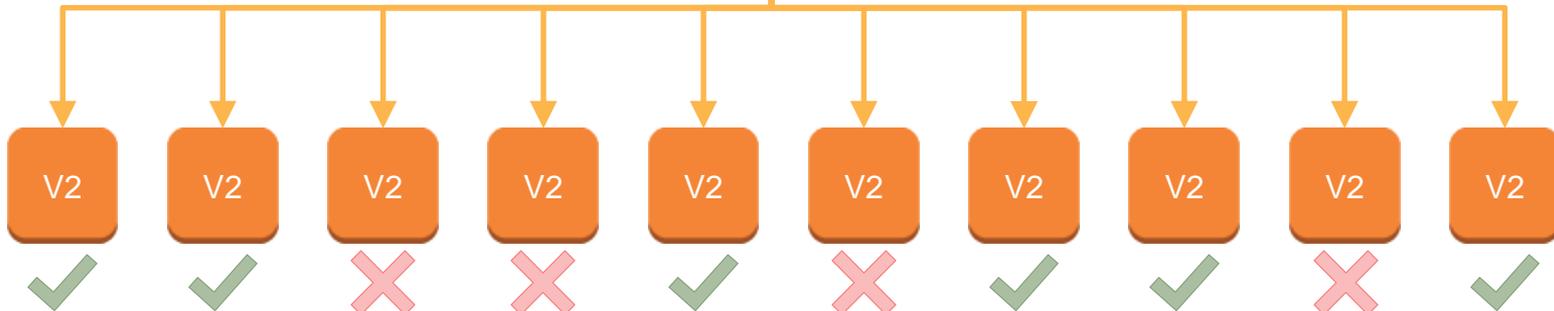
- ❖ デプロイ対象全体の何割が正常であればデプロイを成功とみなすかを定義できる

MHH 66%, 10 hosts:

4 failure – 60% healthy



ELB



Production Fleet

CodeDeploy - ロールバック

- ✿ デプロイメントが失敗した際に自動的に前のリビジョンにロールバックするようにDeployment Groupを設定する

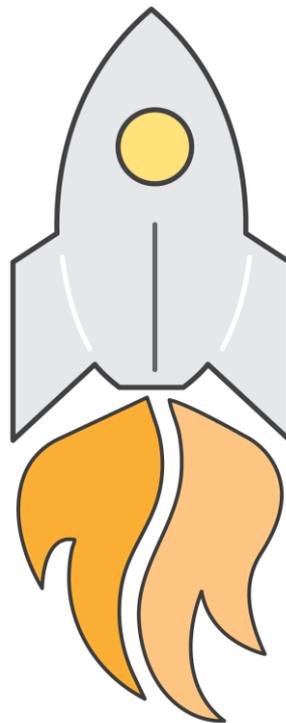
Rollbacks

Enable deployment rollbacks for this deployment group.

- Roll back when a deployment fails
- Roll back when alarm thresholds are met

継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート



デプロイ単位のセグメンテーション

Requirement

健全でないアプリのデプロイ範囲を広げない

- ❖ 問題範囲を制限しビジネス影響を最小限にする
- ❖ 部分的に数時間～数日間運用してから範囲を広げる

Implement

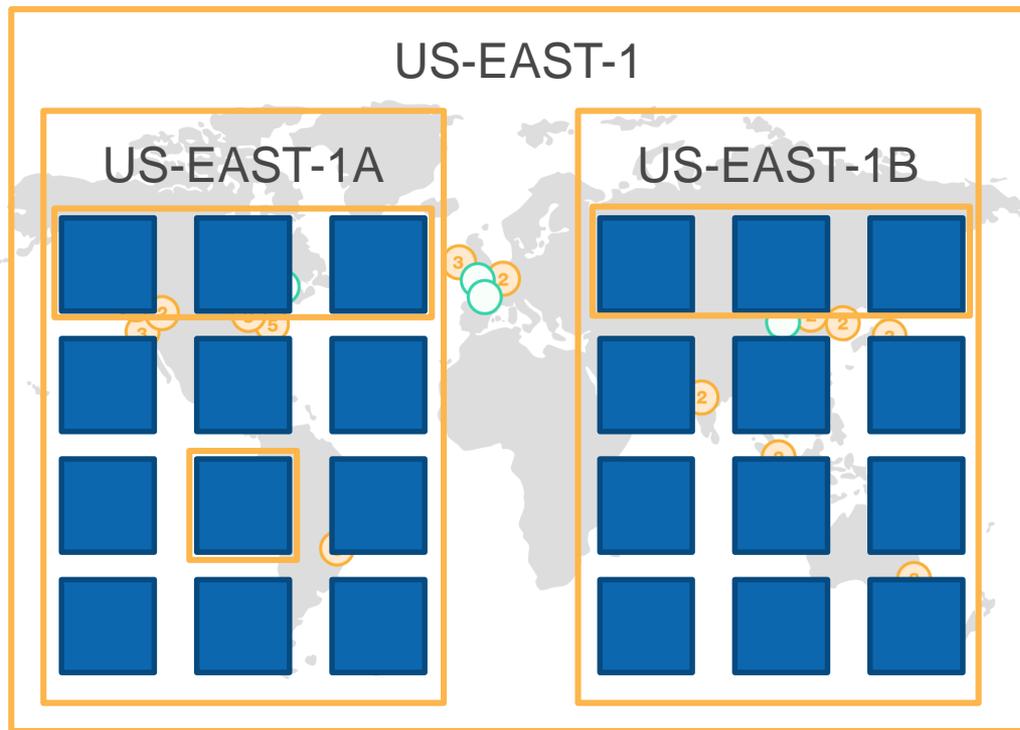
セグメント分割することでデプロイのリスクを低減する

1. 本番環境を複数のセグメントに分割する
2. 一つのセグメントを対象にデプロイする
3. 各セグメントに対してポストデプロイテストを実施する
4. 各セグメントに対し2と3を繰り返す

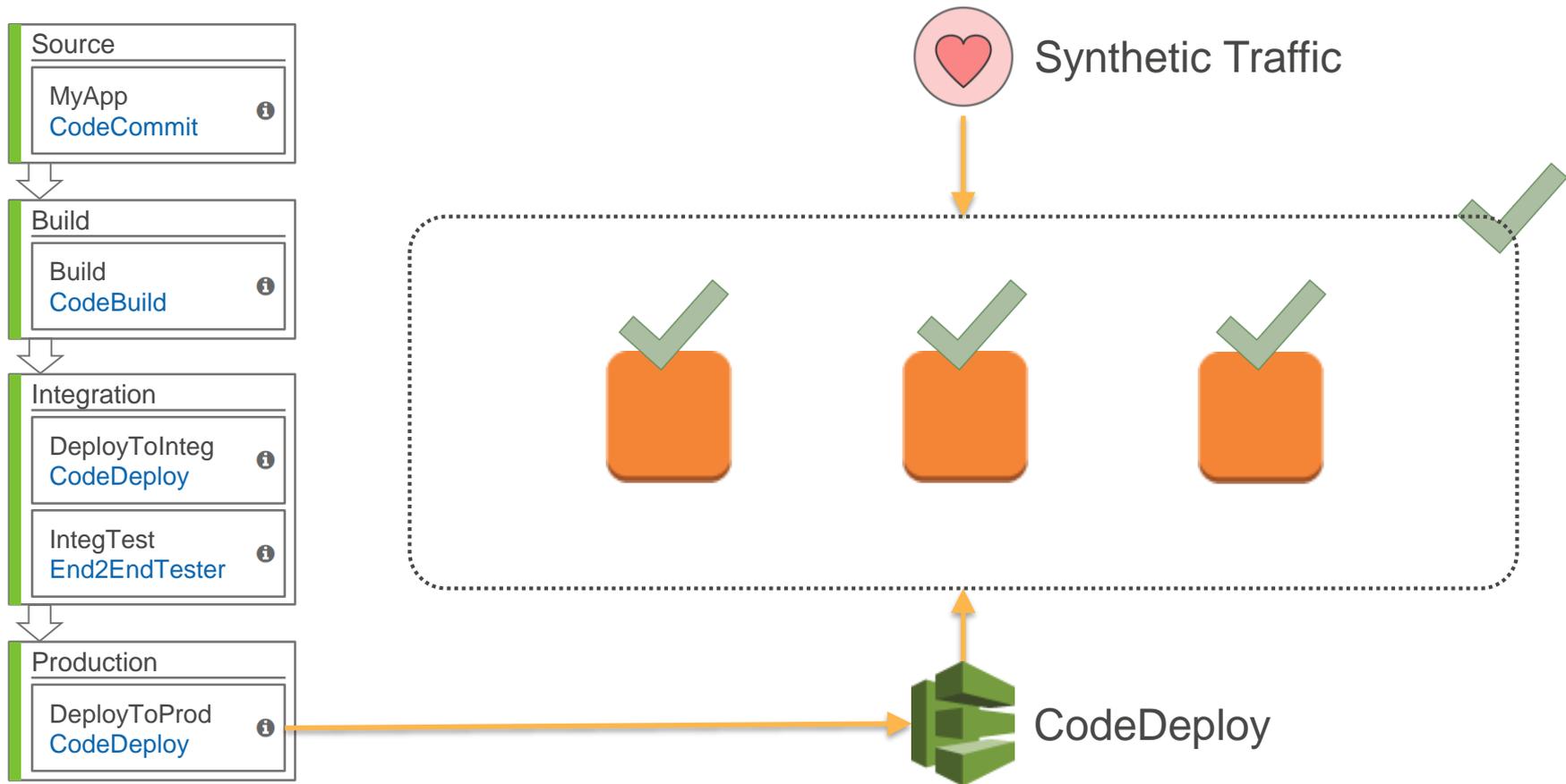
複数セグメントにデプロイ単位を分割

セグメントタイプ:

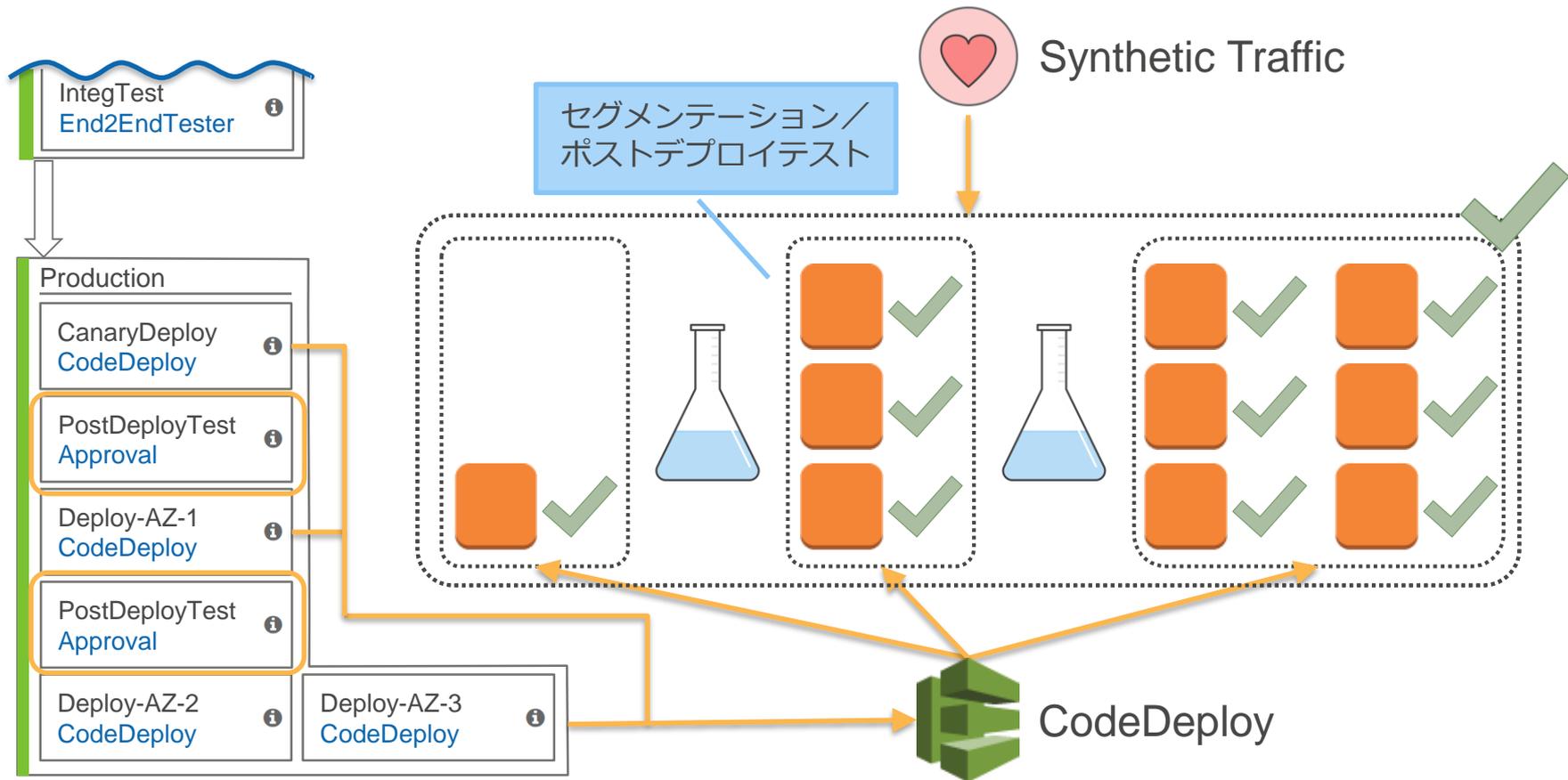
- リージョン
- Availability Zone
- サブゾーン
- 特定ホスト
(カナリア)



リリースプロセス + デプロイセグメンテーション



リリースプロセス + デプロイセグメンテーション



CodeDeploy - デプロイメントグループ

- ❖ セグメントごとにデプロイメントグループを作成
 - ❖ タグ
 - ❖ Auto Scaling Group

Edit Deployment Group: myapplication_us-east-1a ?

Change the name, deployment configuration, service role ARN, and add or remove instances from your deployment group.

Deployment Group Name*

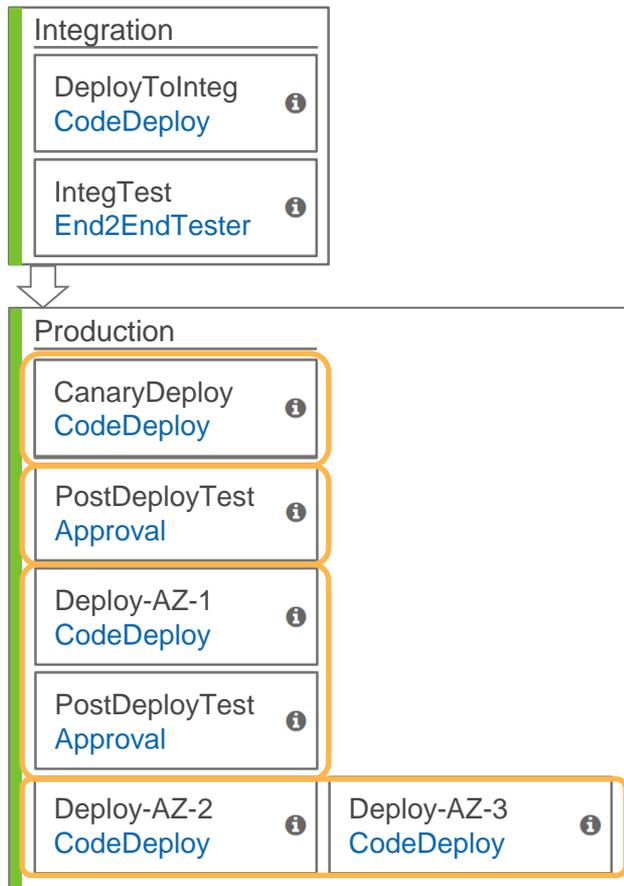
Add Instances

Locate and add existing instances to this deployment group by searching for their tags or Auto Scaling group names.

Search by Tags ⓘ

	Tag Type	Key	Value	Instances	
1	Auto Scaling Group ▼	CodePipelineDemo-AutoScaling-Rails-Distributor-Custom-Action-▼		1	✕
2	Amazon EC2 ▼	▼	▼		✕

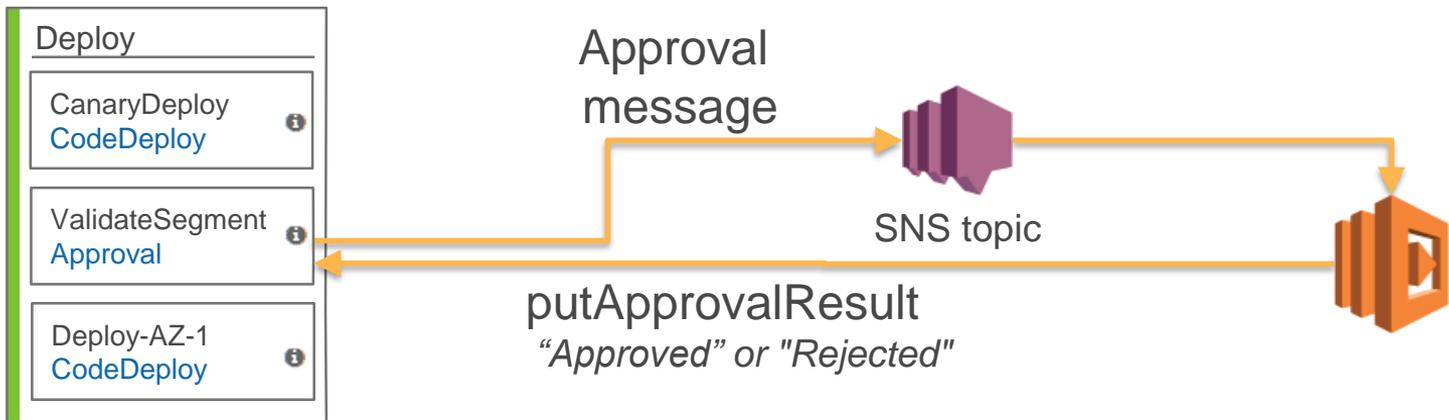
CodePipeline - アクションの分割



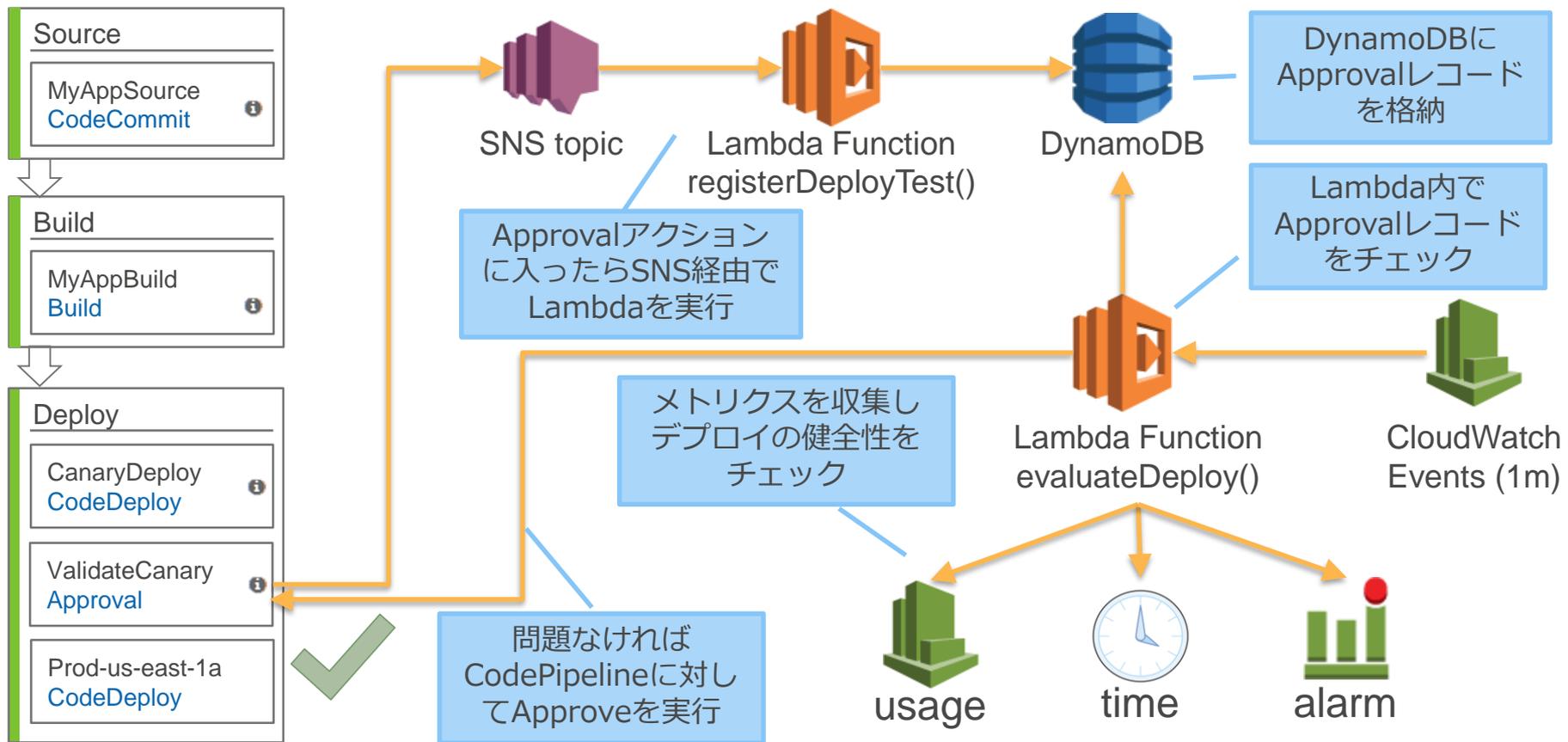
1. 最小セグメントへのデプロイ
2. ポストデプロイテスト
3. 1 AZへのデプロイ
4. ポストデプロイテスト
5. 残りのAZへのデプロイ

CodePipeline - Approvalアクション

- ❖ Approvalアクションによりマニュアル承認ステップを追加できる
- ❖ SNSにメッセージが飛ぶため、Lambdaと連携し自動承認も可能
 - ❖ Approvalアクションのタイムアウト期間は7日間あるため、長時間のヘルスチェックの結果を確認してから次のステップに進むということができる
 - ❖ タイムアウトが短くても良い処理の場合、直接Lambdaを実行するInvokeアクションも利用可能（タイムアウトは1時間）

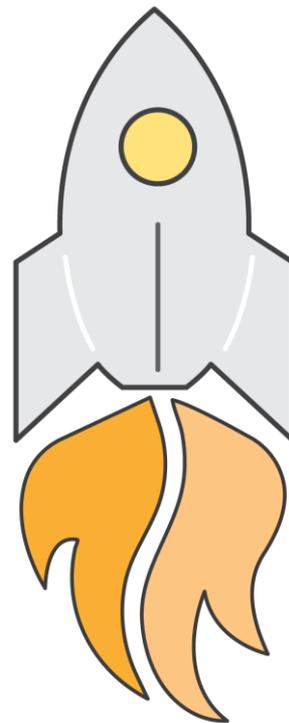


Reference Architecture - ポストデプロイテスト



継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート



プロダクション昇格の無効化

Requirement

デプロイ先環境の健全性を確認してからデプロイする

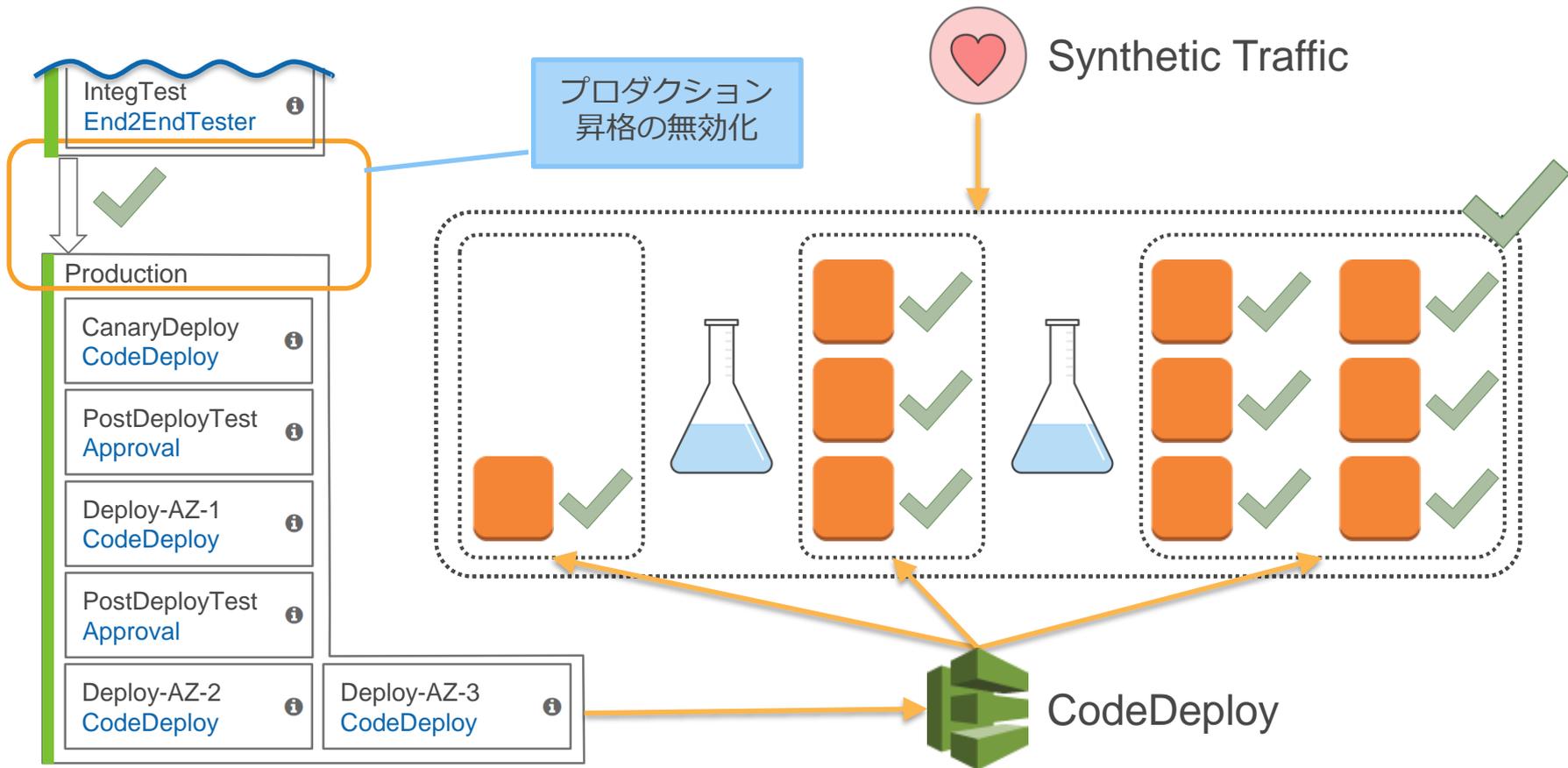
- ✿ パイプラインはデプロイ先が健全かは意識しない
- ✿ 健全でない環境にデプロイを続けると問題が拡大する

Implement

本番環境に問題が起きている場合はリリースを停止する

1. シンセティックトラフィックで本番環境を監視する
2. 問題が見つかったらパイプラインのトランジションを無効化する

リリースプロセス + プロダクション昇格の無効化



CodePipeline - トランジションの有効化/無効化

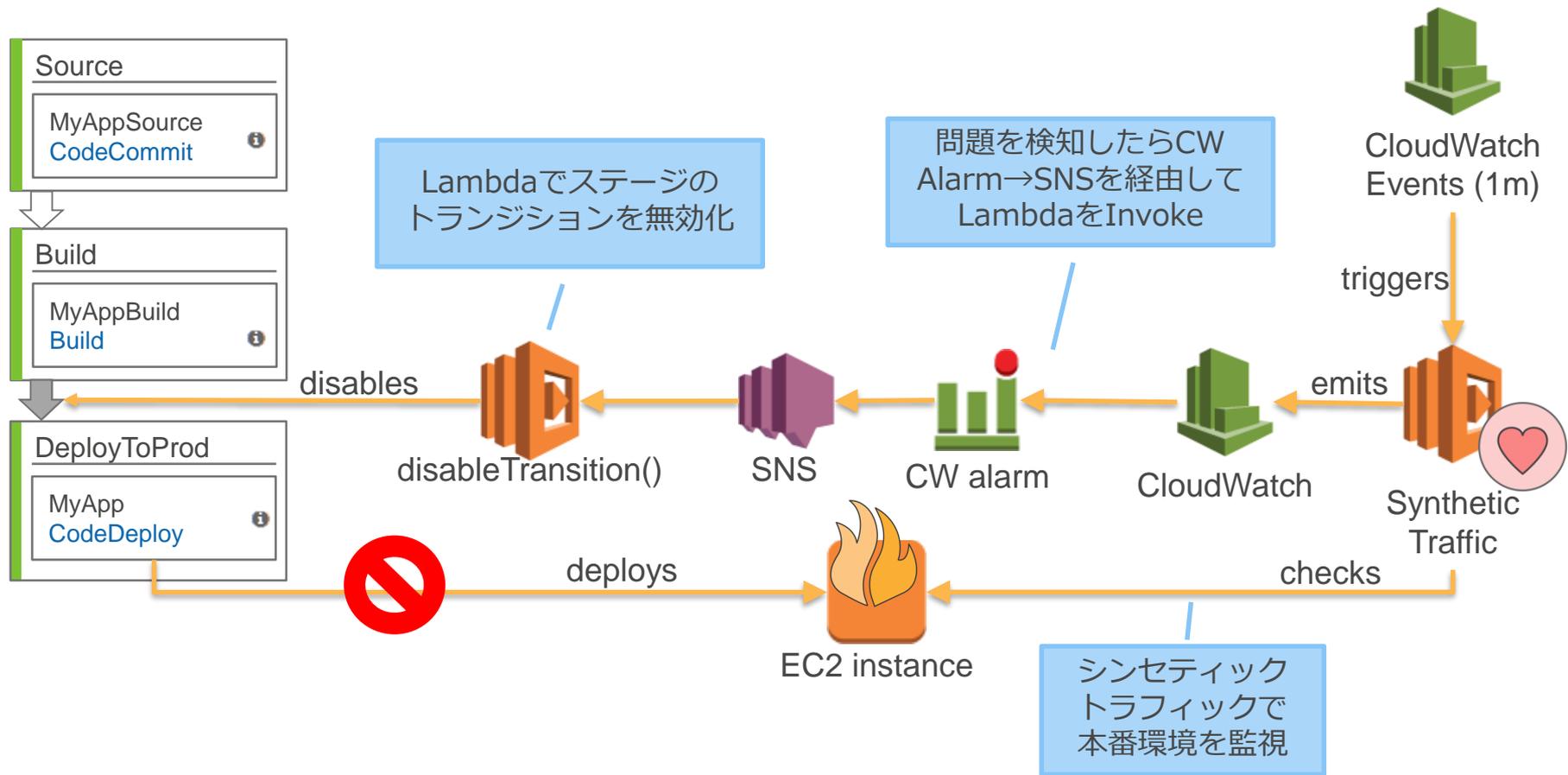
❖ ステージ間のトランジションを一時的に無効化できる

The image illustrates the process of temporarily disabling and then enabling a transition in an AWS CodePipeline. It is divided into two parts by a large orange arrow pointing from left to right.

Left Part (Disabling the Transition):
The pipeline view shows two stages: 'IntegTest' (AWS Lambda) and 'Production' (AWS CodeDeploy). A dialog box titled 'Disable transition' is open, displaying the message: 'You are about to disable the transition between Integration and Production.' Below this, there is a text input field for the reason, containing the word 'test'. A 'Disable' button is highlighted with an orange border.

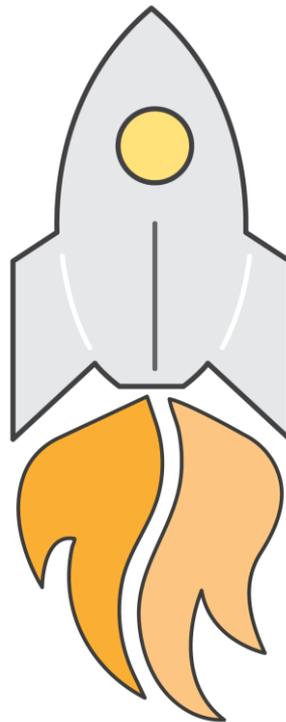
Right Part (Enabling the Transition):
The pipeline view is the same, but the transition between 'IntegTest' and 'Production' is now disabled, indicated by a greyed-out arrow. A dialog box titled 'Enable transition' is open, displaying the message: 'You are about to enable the transition from Integration to Production.' Below this, there is a text input field for the reason, containing the word 'test' and the text 'Disabled: just now'. An 'Enable' button is highlighted with an orange border.

Reference Architecture – プロダクション昇格の無効化



継続的デリバリーテクニック

1. 本番環境の継続的監視
2. デプロイメントヘルスチェック
3. デプロイ単位のセグメンテーション
4. プロダクション昇格の無効化
5. BlackDayゲート



BlackDayゲート

Requirement

ビジネス的にセンシティブな時期／時間を考慮する

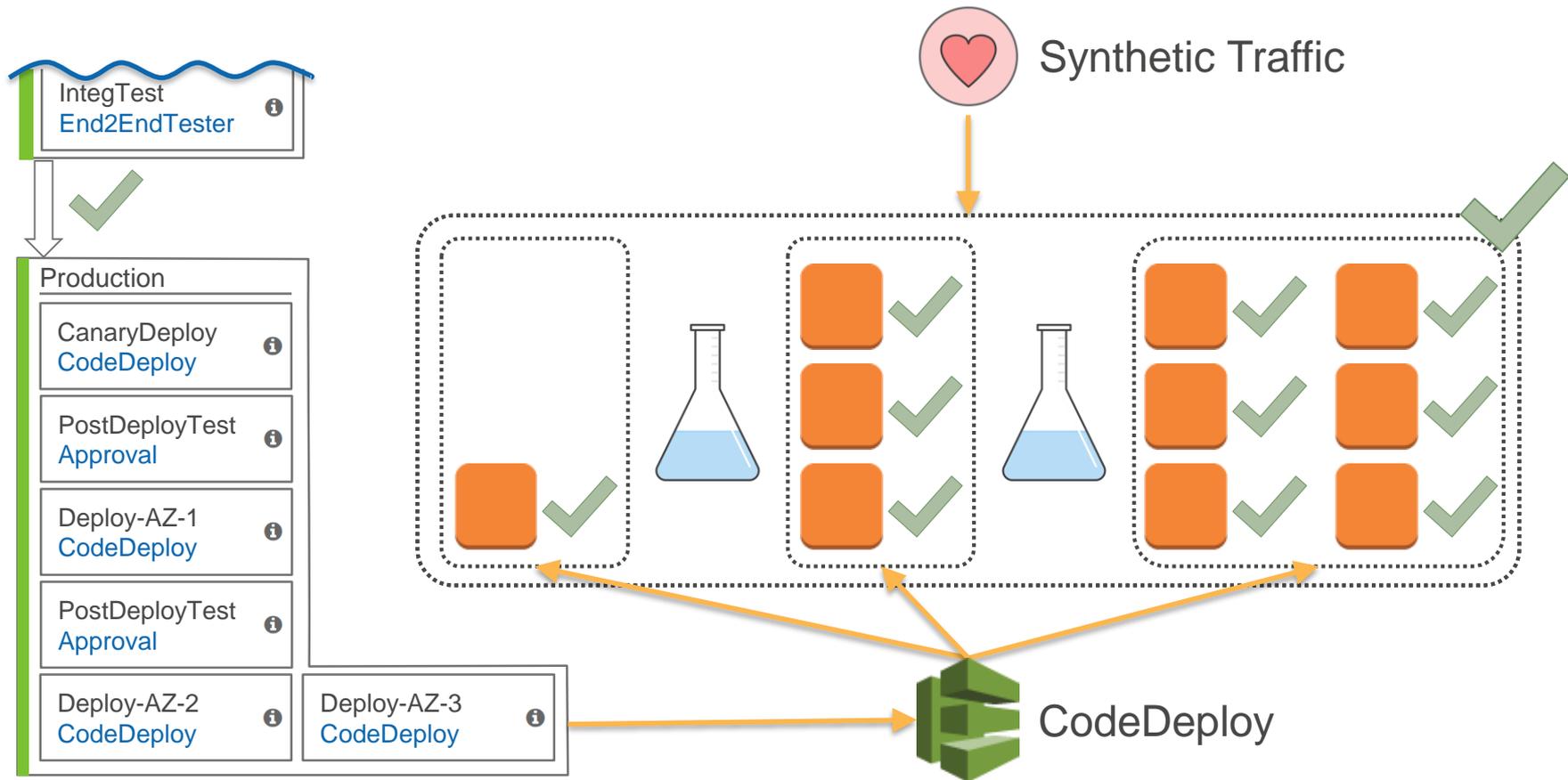
- ❖ キャンペーン期間中など、ビジネス的にセンシティブなタイミングに問題が起きると大きな機会損失につながる

Implement

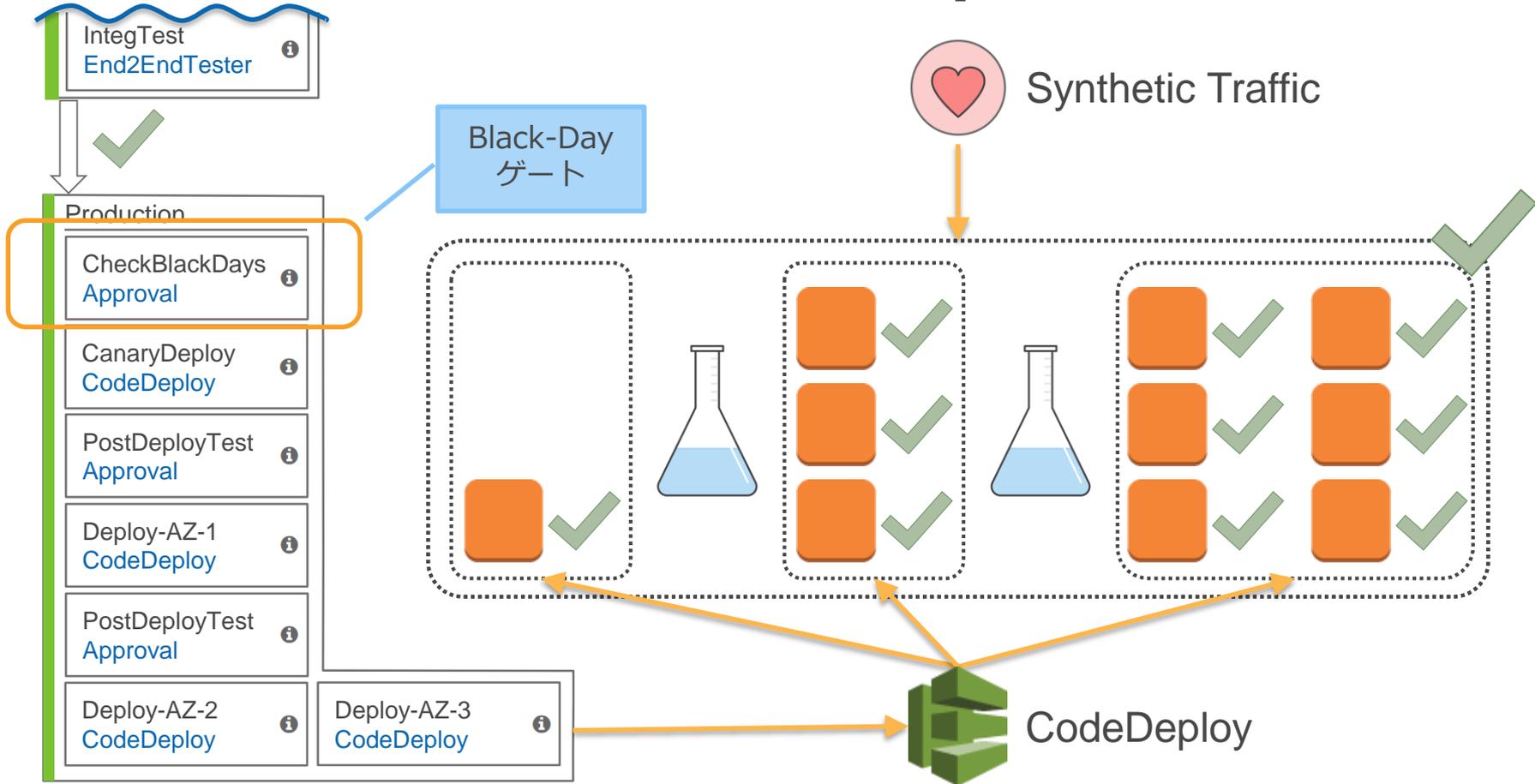
センシティブな時間(BlackDay)はデプロイを停止する

1. パイプラインにBlackDayカレンダーを導入する
2. BlackDay中はApproveアクションでデプロイを止める
3. BlackDay期間以外は自動的にApproveする

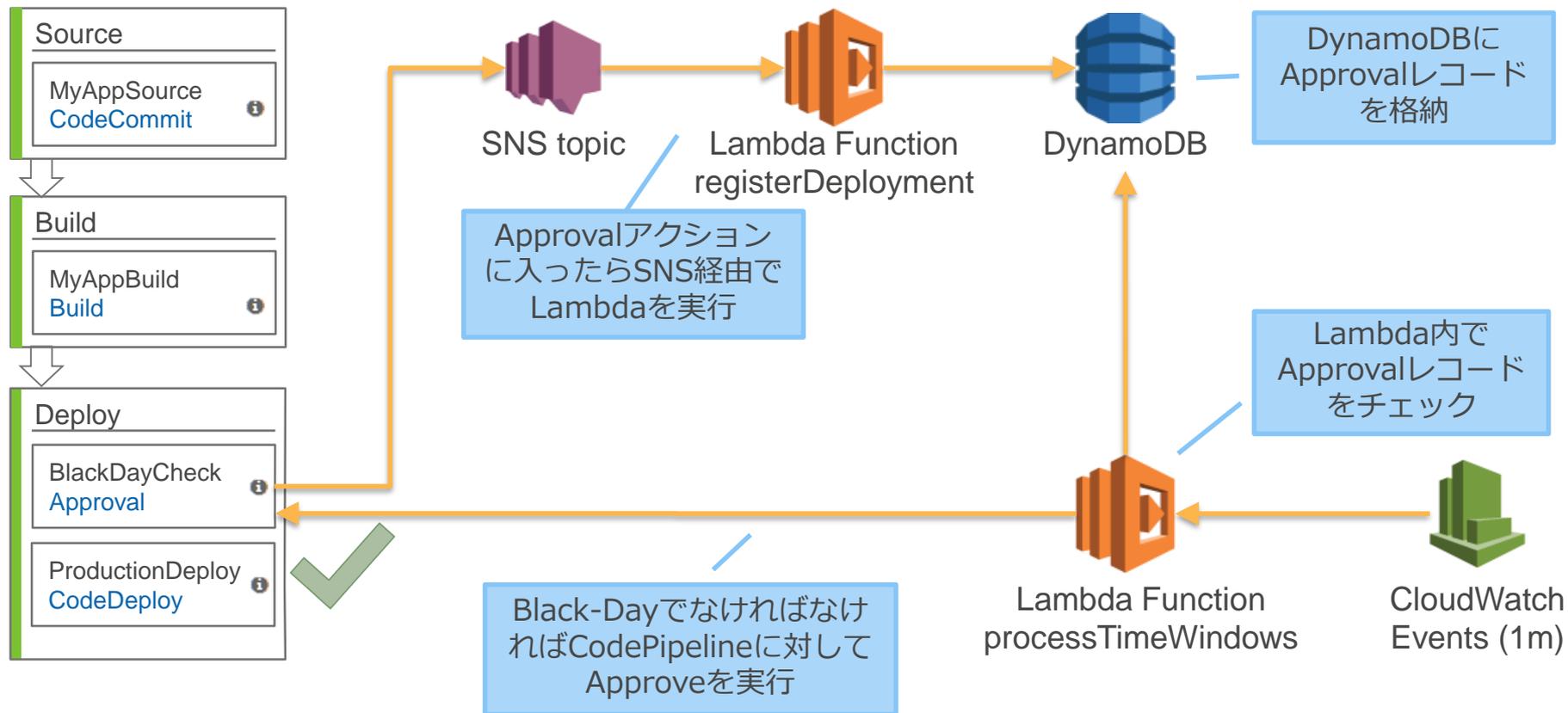
リリースプロセス + BlackDayゲート



リリースプロセス + BlackDayゲート



Reference Architecture – BlackDayゲート



まとめ

CI/CDパイプラインを実現するAWSサービス

ソースコードの
バージョン管理



AWS CodeCommit

ビルド自動化

マネージドサービスを使えば

簡単にCI/CDを実現するツールが利用できる

デプロイ自動化



AWS CodeBuild



AWS CodeDeploy

AWS CloudFormation

AWS Elastic Beanstalk



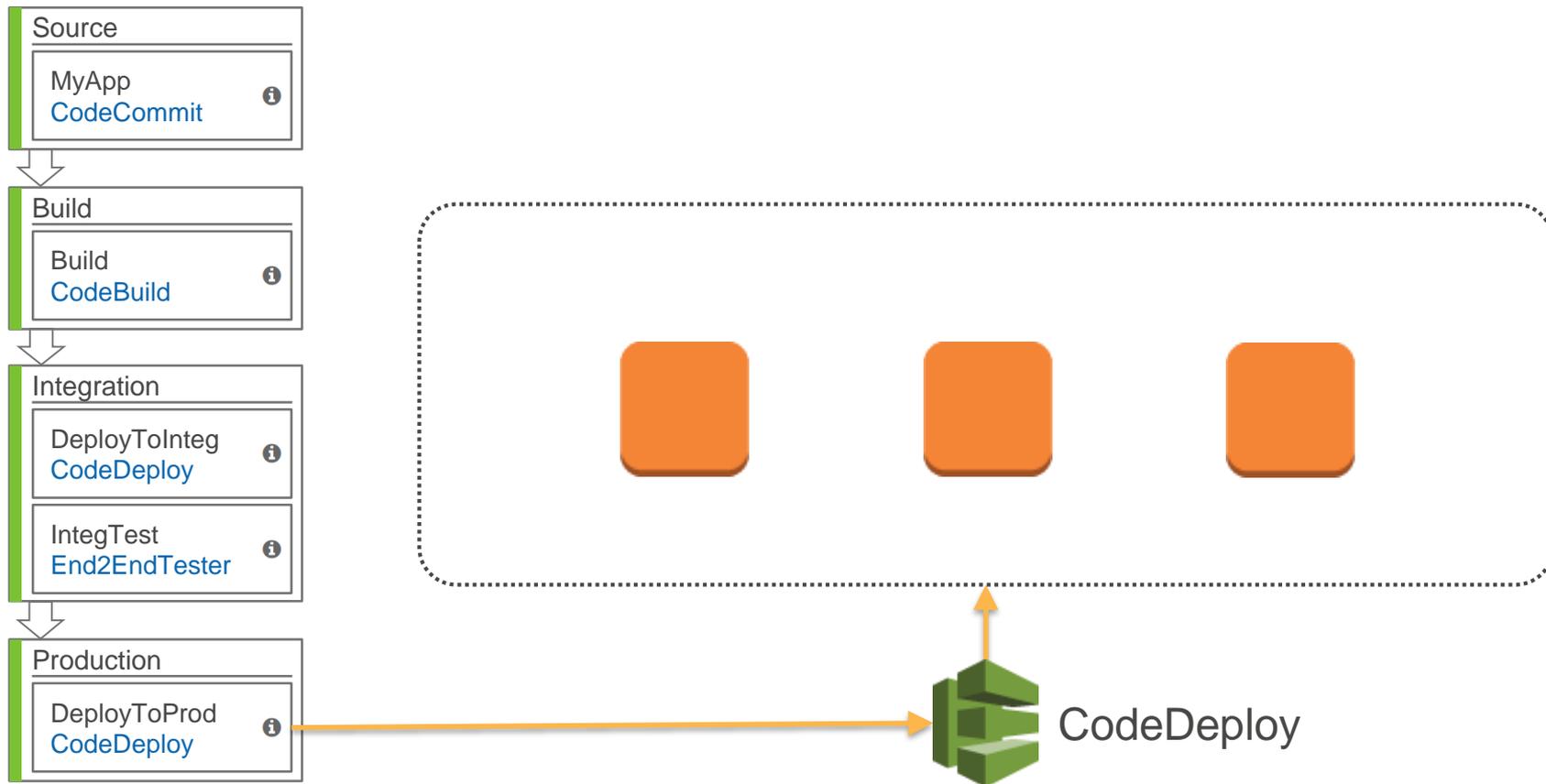
AWS CodeStar

ワークフロー管理

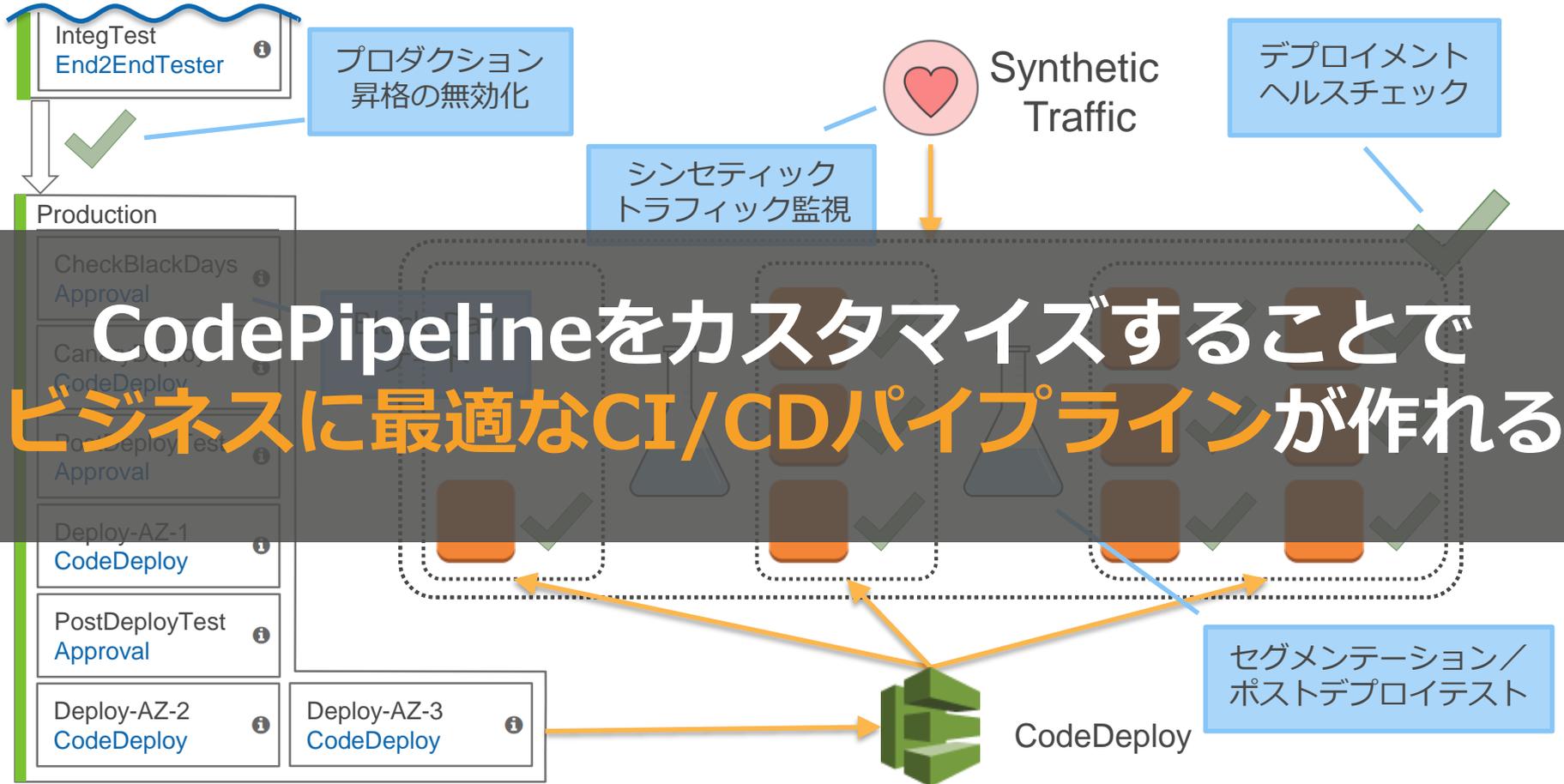


AWS CodePipeline

継続的デリバリープロセス：ベースライン



継続的デリバリープロセス：ゴール



関連リソース

❖ 関連セッション(Dev Day)

- ❖ Docker コンテナへの継続的デプロイメント on AWS

～CodeCommit, CodeBuild, CodePipeline, CloudFormation, ECR, ECS を活用した CI/CD ～

- ❖ DevSecOps on AWS - Policy in Code

- ❖ サーバーレスアプリケーションのための CI/CD パイプライン構築

❖ CI/CDホワイトペーパー

[practicing-continuous-integration-continuous-delivery-on-AWS.pdf](#)

❖ AWS クラウドサービス活用資料集

<https://aws.amazon.com/jp/aws-jp-introduction/>

本セッションのFeedbackをお願いします

受付でお配りしたアンケートに本セッションの満足度やご感想などをご記入ください
アンケートをご提出いただきました方には、もれなく**素敵なAWSオリジナルグッズ**を
プレゼントさせていただきます



アンケートは受付、パミール3FのEXPO展示会場内にて回収させていただきます

AWS

S U M M I T

