



# [AWS Black Belt Online Seminar] AWS CodeStar & AWS CodePipeline

サービスカットシリーズ

Solutions Architect 山口 弘樹  
2020/11/11

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



過去資料  
<https://amzn.to/JPArchive>



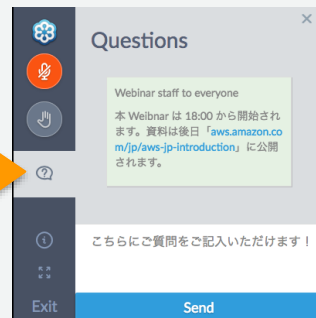
# AWS Black Belt Online Seminar とは

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

## 質問を投げることができます！

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

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



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

# 内容についての注意点

- 本資料では2020年11月11日時点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.

# 自己紹介

## ➤ 名前

山口 弘樹 (やまぐち ひろき) yamaghi@

## ➤ 所属

アマゾン ウェブ サービス ジャパン株式会社  
パートナーソリューションアーキテクト

## ➤ 経歴

国内 Sler で主に PM / PL や上級 SE を担当  
Spring Boot を使った MSA システム設計、アジャイル開発の推進

## ➤ 好きなAWSサービス

AWS Code シリーズ



# 本セミナーの概要

## □ 本セミナーで学習できること

- ❖ AWS CodeStar、AWS CodePipelineの機能概要
- ❖ CI/CD を構築する際の一般的な構成

## □ 対象者

- ❖ AWS CodeStar、AWS CodePipelineにご興味がある方
- ❖ これから CI/CD を検討している方、または既にCI/CD を導入している方
  - 定期的に機能拡充を行っているアプリケーションエンジニア、または開発を支えるインフラエンジニア
  - 機能拡充頻度は低いが CI/CD を通してプロセスの自動化を図りたい方

# 本日のアジェンダ

- CI/CD が必要となる背景
- AWS CodeStar の機能説明
- AWS CodePipeline の機能説明
- よくあるご相談
- まとめ

# 本日のアジェンダ

- CI/CD が必要となる背景
- AWS CodeStar の機能説明
- AWS CodePipeline の機能説明
- よくあるご相談
- まとめ

# 急速なイノベーションがビジネスを進化させる



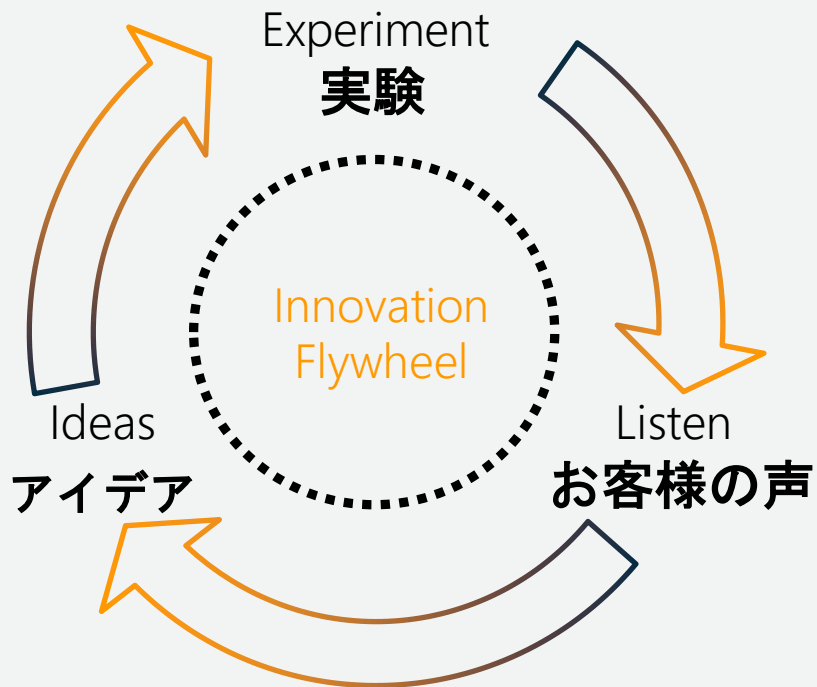
新たなマーケット

新たな顧客価値

新たなデジタル製品とサービス



# 実験がイノベーションを加速する



# ソフトウェア開発手法の潮流の変化

複雑で変化の激しい問題に対応し、可能な限り価値の高いプロダクトを生産的かつ創造的に届けるための開発方法論が求められるようになった

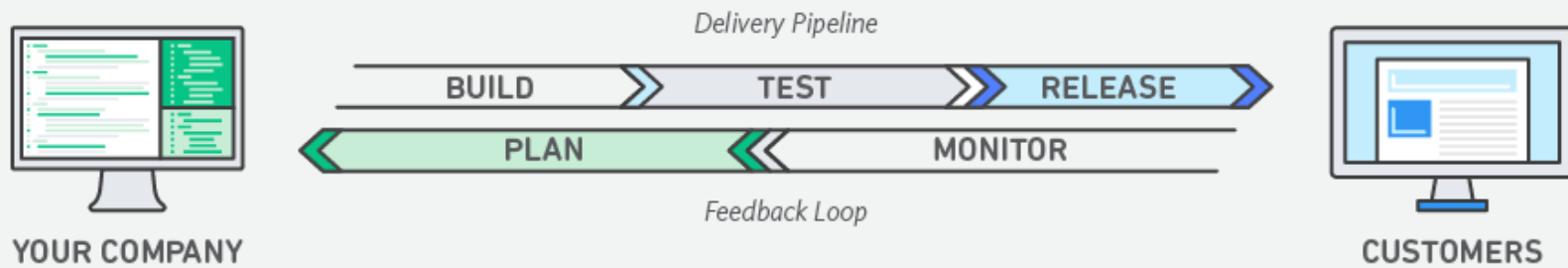
- スクラムやXPなどアジャイル型開発が浸透
- デザイン思考などユーザーを理解し、仮説立てて問題解決する設計手法



<http://agilemanifesto.org/iso/ja/manifesto.html>

# AWS が考える DevOps

## ソフトウェア開発のライフサイクル



1way:リードタイム短縮 2way:Feedbackを元に改善 3way:継続的な実験と学び

CI/CDを利用して高品質な機能を迅速にリリースする  
CI/CDを自動化しているチームは  
コードをより速く、自信を持って書いている

30x

頻繁なデプロイ

440x

リードタイムの短縮

60x

ミスが減少

-21%

予期しない作業の削減

+44%

新しい作業

<https://puppet.com/resources/whitepaper/2017-state-of-devops-report>

# リリースプロセスに求められる 4 つのフェーズ



- ソースコードを  
チェックイン
- バージョン管理、  
ブランチ管理
- 新しいコードの  
ピアレビュー



- Java、C#などの  
コードのコンパイル
- ユニットテスト
- スタイルチェッカー
- コードメトリック
- コンテナイメージの  
作成



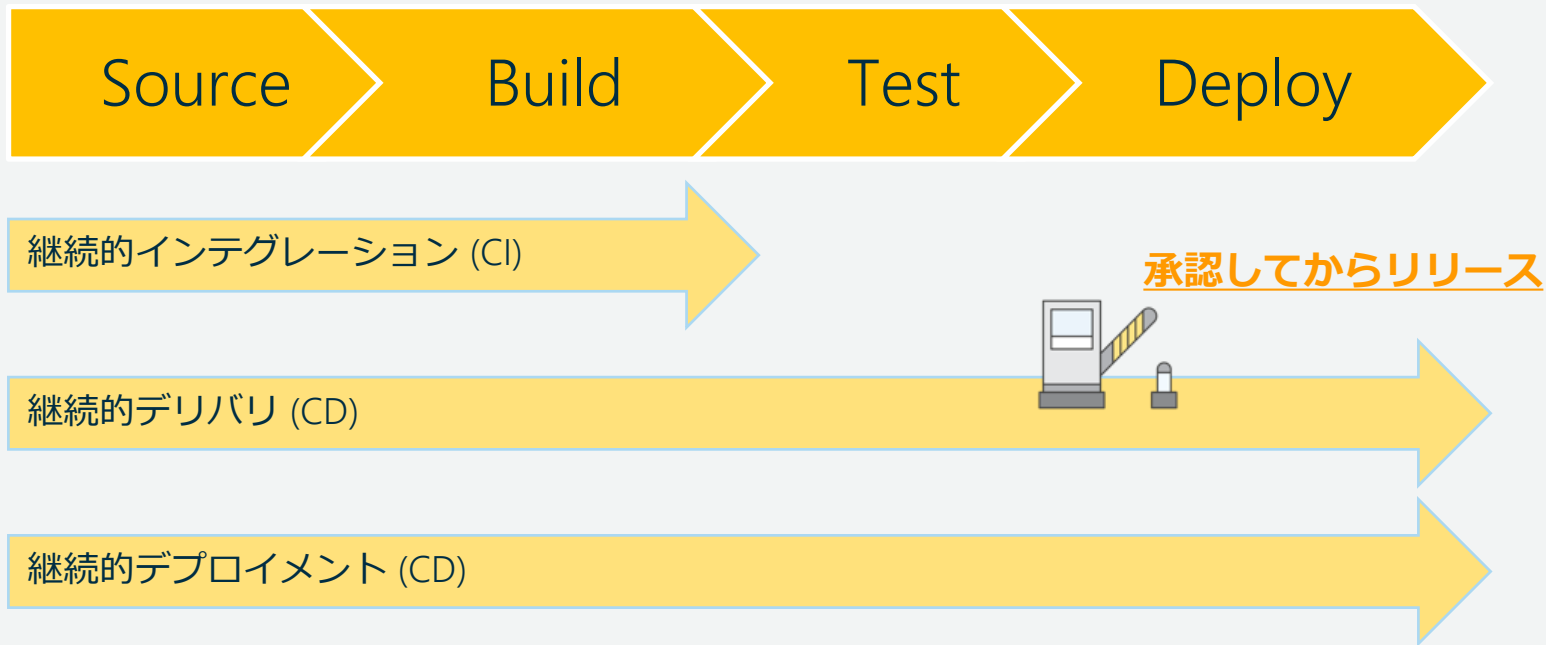
- 他のシステムと  
の統合テスト
- ロードテスト
- UIテスト
- 侵入テスト



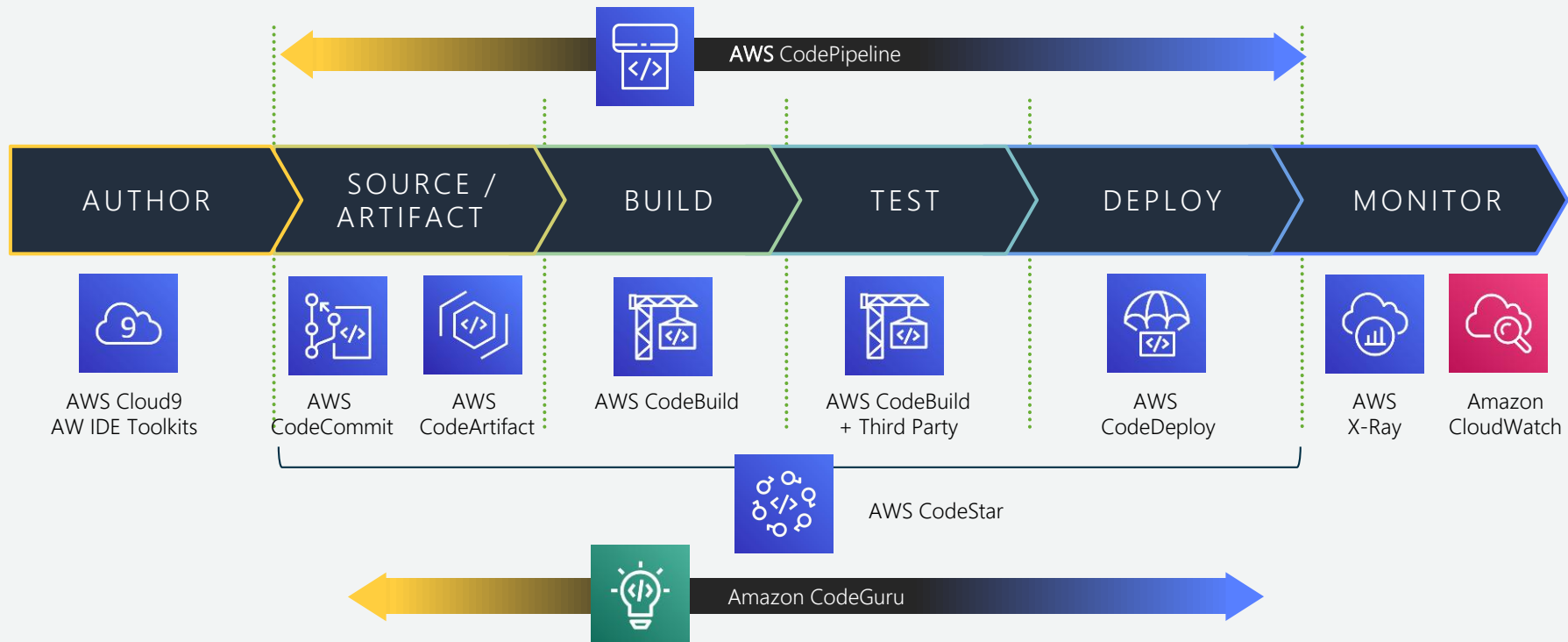
- 本番環境に  
デプロイ



# リリースプロセスのレベル



# CI/CD の実現を容易にするAWS Code サービス



# AWS を活用した CI/CD の特徴



マネージドサービスによる  
高い可用性と信頼性



AWSサービスとの  
シームレスな連携



サードパーティーツールとの  
インテグレーション



任意のアクションは  
コードベースで定義可能



承認アクションと  
ステージ移行



低コストと  
無料利用枠



# 本日のアジェンダ










































- CI/CD が必要となる背景
- **AWS CodeStar の機能説明**
- AWS CodePipeline の機能説明
- よくあるご相談
- まとめ

# AWS CodeStar の概要



- ✓ 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 デリバリーの管理を**容易に**
- ✓ 様々な**プロジェクトテンプレート**から選択

# 豊富なプロジェクトテンプレート

 Go アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Lambda サーバーレスで実行する	 Node.js アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Lambda サーバーレスで実行する	 Python アプリケーションタイプ ウェブサービス  AWSのサービス  AWS Lambda サーバーレスで実行する	 Express.js アプリケーションタイプ ウェブサービス  AWSのサービス  AWS Lambda サーバーレスで実行する	 HTML アプリケーションタイプ 静的ウェブサイト  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 Python (Flask) アプリケーションタイプ ウェブサービス  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 Python (Flask) アプリケーションタイプ ウェブサービス  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する
 Express.js アプリケーションタイプ ウェブサービス  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 Express.js アプリケーションタイプ ウェブサービス  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Java Spring アプリケーションタイプ ウェブサービス  AWSのサービス  AWS Lambda サーバーレスで実行する	 Java Spring アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Node.js アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 Node.js アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Express.js アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する
 Express.js アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Java Spring アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 PHP (Laravel) アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Python (Django) アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 PHP (Laravel) アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する	 Ruby on Rails アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS Elastic Beanstalk マネージド型アプリケーション 環境で実行する	 Python (Django) アプリケーションタイプ ウェブアプリケーション  AWSのサービス  AWS EC2 管理している仮想サーバーで 実行する

(※テンプレートの一部のみ記載)

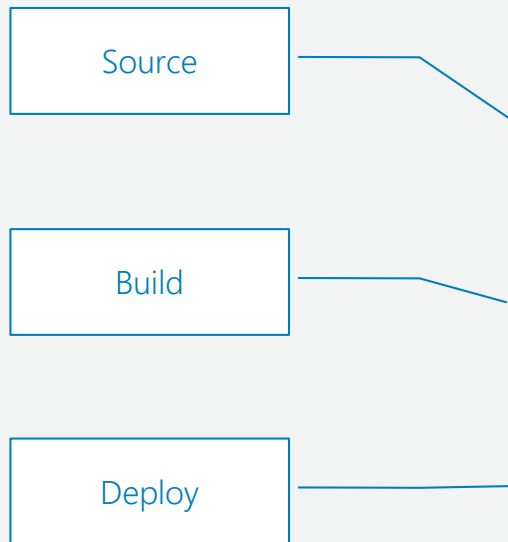
プロジェクトテンプレートを選ぶと CI/CD 環境が自動で作成される

# プロジェクト ダッシュボード

The screenshot shows the AWS Project Dashboard for 'MyAppA'. The top navigation bar includes tabs for '概要' (Overview), 'IDE', 'リポジトリ' (Repository), 'パイプライン' (Pipeline), 'モニタリング' (Monitoring), and '問題' (Issues). A callout box labeled 'Repository エンドポイント' points to the 'リポジトリ' tab. Another callout box labeled 'デリバリー パイプライン' points to the 'パイプライン' tab. A third callout box labeled '統合メニュー' points to the 'モニタリング' tab. A fourth callout box labeled 'App エンドポイント' points to the '問題' tab. A button labeled 'アプリケーションの表示' is also visible. Below the navigation bar, the 'プロジェクトのアクティビティ' section is expanded, showing details for the 'myappa-Pipeline'. It lists the latest action as 'Deploy: CodeDeploy を使用する Deploy' completed 1 day ago with a status of '成功しました'. A callout box labeled '開発ツールとの連携' points to this section. To the right, a 'CPUUtilization' graph shows a line chart with a blue area fill, with a callout box labeled 'CloudWatch メトリクス' pointing to it. The graph has a 'ダッシュボードに追加' button and a refresh icon. The x-axis shows time from 23:30 to 02:00, and the y-axis shows CPU utilization from 0 to 2.67.

ダッシュボードを通して、リポジトリやパイプラインが可視化

# パイプラインの自動生成



MyAppA アプリケーションの表示

概要 IDE リポジトリ **パイプライン** モニタリング 問題

パイプラインの詳細 編集 履歴の表示 変更をリリースする

パイプライン myappa-pipeline 最新のアクション Deploy: CodeDeploy を使用する Deploy ステータス 成功しました 8 日前

**Source**

ApplicationSource AWS CodeCommit ①  
成功しました - 8 日前 92676169

**Build**

PackageExport AWS CodeBuild ①  
成功しました - 8 日前 詳細

**Deploy**

GenerateChangeSet AWS CloudFormation ①  
成功しました - 8 日前 詳細

ステージとして、ソース・ビルド・デプロイが構築される

# 開発ツールとの連携

## ▼ プロジェクトコードにアクセス 情報

プロジェクトを統合開発環境 (IDE) に接続して、ソースコードの変更を修正、テスト、プッシュします。[ドキュメントを開く](#)

## ブラウザで使用できるツール



### AWS Cloud9

AWS Cloud9 を使用して、AWS CodeStar が作成する AWS に完全に統合された開発環境でソースコードを記述、実行、編集します。

環境を作成



### AWS CodeCommit

CodeCommit を使用して、ソースコードのクローンを作成したり、認証情報を管理したりすることなく、コンソールで直接ソースコードを編集およびアップロードします。

AWS CodeCommit で編集する

## ローカルで使用できるツール



### コマンドラインインターフェイス

コマンドラインを使用して、ソースコードを直接プロジェクトの Git リポジトリから編集します。

[指示を表示](#)



### Eclipse

AWS Toolkit for Eclipse を使用して、Eclipse でソースコードを編集します。

[指示を表示](#)



### Visual Studio for Windows

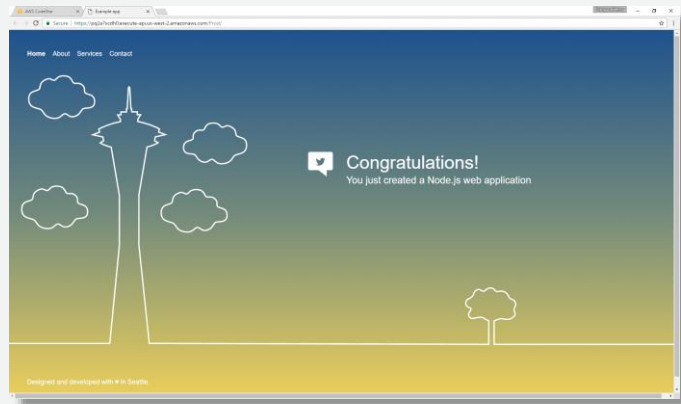
AWS Toolkit for Visual Studio を使用して、Microsoft Visual Studio 2015 以降でソースコードを編集します。

[指示を表示](#)

開発ツールとして AWS Cloud9 も同時にデプロイできる

# セットアップ内容

- プロジェクトテンプレートの選択
  - EC2 or Beanstalk or Lambda
- プロジェクトの設定
  - リポジトリ (CodeCommit、GitHub)
- プロジェクトの編集
  - プロジェクトダッシュボード
  - CodePipeline 継続的デプロイメントパイプライン
  - CloudWatchメトリクス



# AWS CodeStar の操作: プロジェクトを作成

## プロジェクトテンプレートの選択 情報

テンプレート フィルタをクリアする

AWS のサービス、アプリケーションタイプ、およびプログラミング言語でフィルタリングします。

< 1 2 3 4 5 6 >

<p><b>node</b> ハローワールドスキル <input checked="" type="radio"/></p> <p>アプリケーションタイプ <input type="radio"/> Alexa スキル</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>	<p><b>Python</b> ハローワールドスキル <input type="radio"/></p> <p>アプリケーションタイプ <input type="radio"/> Alexa スキル</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>	<p><b>Go</b> <input type="radio"/></p> <p>アプリケーションタイプ ウェブアプリケーション</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>
<p><b>node</b> Node.js <input type="radio"/></p> <p>アプリケーションタイプ ウェブアプリケーション</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>	<p><b>Python</b> Python <input type="radio"/></p> <p>アプリケーションタイプ ウェブサービス</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>	<p><b>Express</b> Express.js <input type="radio"/></p> <p>アプリケーションタイプ ウェブサービス</p> <p>AWS のサービス <input checked="" type="radio"/> AWS Lambda サーバーレスで実行する</p>

## プロジェクトの設定 情報

### プロジェクトの詳細

プロジェクト名

プロジェクト ID  
この ID は、リソース ARN と他の AWS リソース用に生成された名前を追加されます。  
  
プロジェクト ID は 2~15 文字以内で、文字で始まる必要があり、小文字、数字、ダッシュのみを含めることができます。

### プロジェクトのリポジトリ

リポジトリプロバイダーを選択する

CodeCommit  
プロジェクトの新しい AWS CodeCommit リポジトリを使用します。

GitHub  
プロジェクトに新しい GitHub ソースリポジトリを使用します (既存の GitHub アカウントが必要です)。

リポジトリ名  
  
リポジトリ名には、文字、数字、ダッシュ、アンダースコア、ピリオドのみを含めることができます。「.git」で終わることはできません。

1. テンプレートからサンプルプロジェクトを選択
2. プロジェクト名、リポジトリ情報などを入力し、あとは数分間待つだけ



# AWS CodeStar の操作: チームメンバを追加・削除

チームメンバー 情報 編集 削除 チームメンバーの追加

🔍

< 1 > ⚙️

	名前	Eメール	ロール	リモートアクセス
<input type="radio"/>	yamaghi	yamaghi@amazon.co.jp	所有者	はい

チームメンバーの詳細

ユーザー  
他の AWS CodeStar プロジェクトの既存のチームメンバー、または IAM ユーザー。

または

表示名

Eメールアドレス

プロジェクトロール

リモートアクセス  
 プロジェクトインスタンスへの SSH アクセスを許可します。

キャンセル

チームメンバーの追加

- 既存のチームメンバー、またはIAMユーザーからユーザーの追加が可能
- 表示名、メールアドレス、ロールを指定して簡易にプロジェクトメンバーを作成できる

# AWS CodeStar の料金

AWS CodeStar に対する追加料金は発生しません

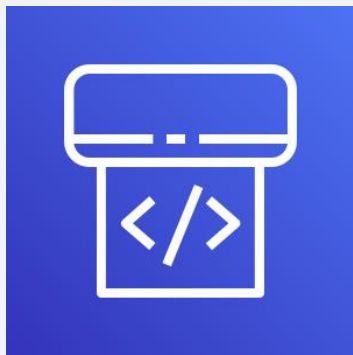
- CodeStar プロジェクトでプロビジョニングされた AWS リソース (例: Amazon EC2 インスタンス、AWS Lambda の実行、Amazon EBS ボリューム、Amazon S3 バケット) に対してのみ料金が発生します。

<https://aws.amazon.com/jp/codestar/pricing/>

# 本日のアジェンダ

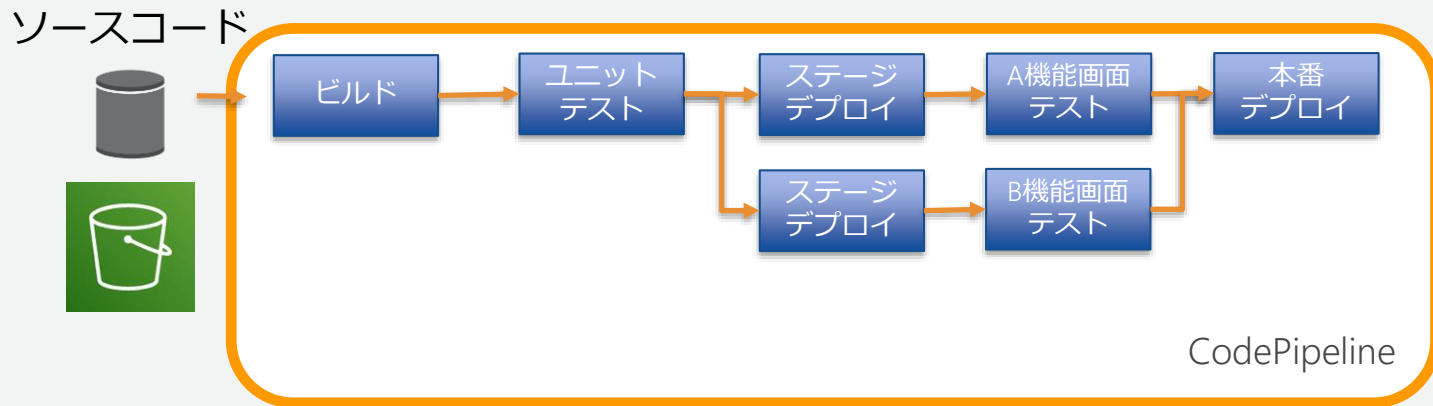
- CI/CD が必要となる背景
- AWS CodeStar の機能説明
- **AWS CodePipeline の機能説明**
- よくあるご相談
- まとめ

# AWS CodePipeline 概要



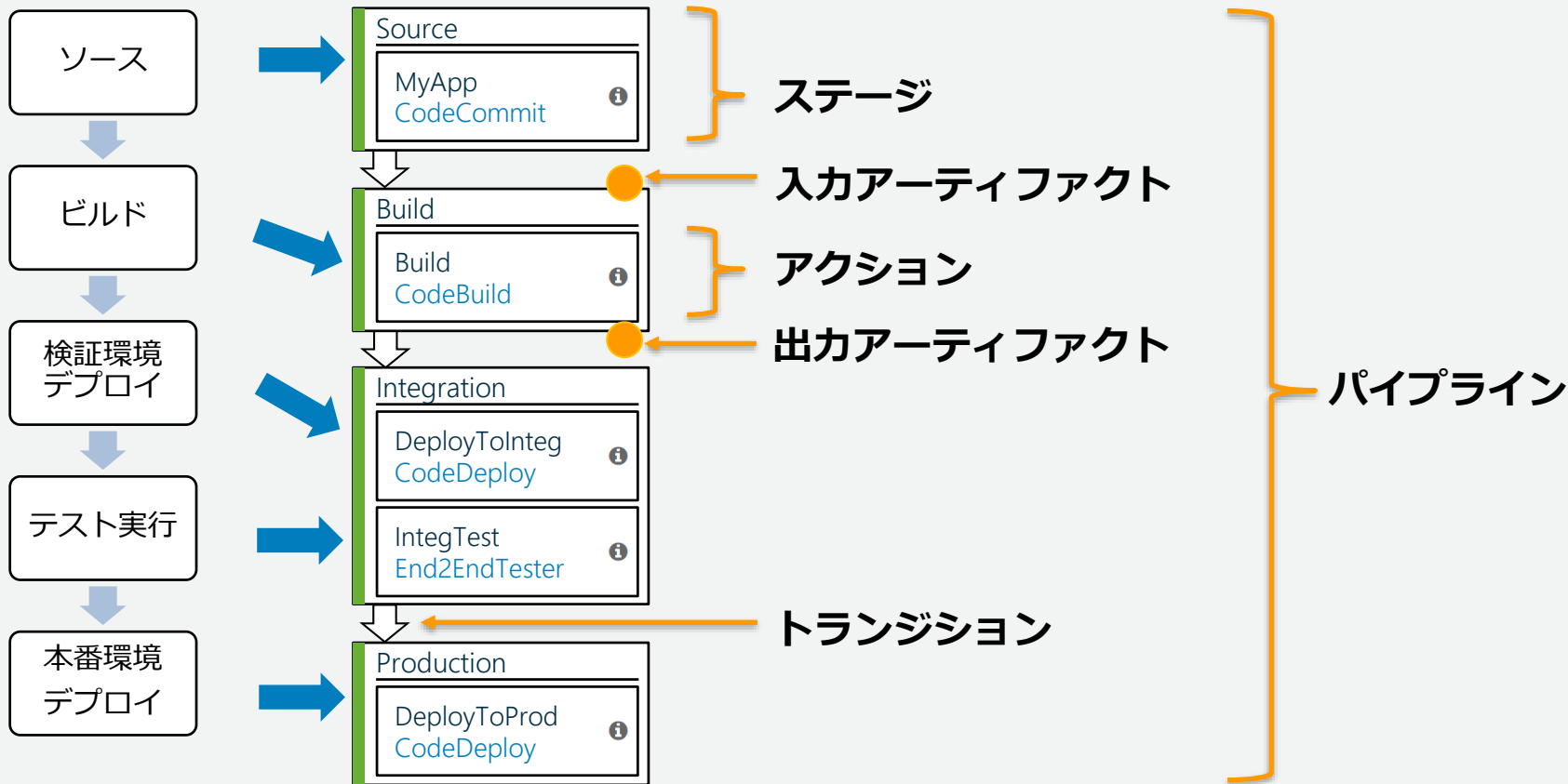
- ✓ 素早く信頼性の高いアプリケーション更新のための**継続的デリバリーサービス**を提供
- ✓ ソフトウェアリリースプロセスを**モデル化**することにより視覚化
- ✓ コードが変更されるとコードのビルド、テスト、デプロイを実施
- ✓ **AWSサービス**や**サードパーティーのツール**と**統合**

# 開発のスタイルに合ったパイプラインを自在に構築



パイプラインは開発チームやサービスにより異なる  
スムーズなデプロイメントにはパイプラインは欠かせない

# CodePipeline の定義



# CodePipeline の操作: パイプラインの作成

- パイプライン名を指定する
- 任意のサービスロールを指定する
- 高度な設定（アーティファクトストア、暗号化キー）はオプション

パイプラインの設定を選択する 情報

### パイプラインの設定

パイプライン名  
パイプライン名を入力します。作成後にパイプライン名を編集することはできません。

100 文字を超えることはできません

サービスロール

新しいサービスロール  
アカウントでサービスロールを作成

既存のサービスロール  
アカウントから既存のサービスロールを選択

ロール名

サービスロール名の入力

AWS CodePipeline がサービスロールを作成できるようになるため、この新しいパイプラインでの使用が可能になります。

▶ 高度な設定

キャンセル

次に

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>

# CodePipeline の操作: パイプラインの作成

- ソースプロバイダーを指定する
- ソースプロバイダーに紐づく情報を指定する（例: AWS CodeCommitのリポジトリ名やブランチ名）

ソースステージを追加する 情報

### ソース

ソースプロバイダー  
ここでは、パイプラインの入力アーティファクトを保存します。プロバイダ接続の詳細を指定します。

AWS CodeCommit

リポジトリ名  
ソースコードをプッシュしたところで既に作成したリポジトリを選択します。

MyAppA

ブランチ名  
リポジトリのブランチを選択します

master

検出オプションを変更する  
検出モードを選択して、ソースコードに変更が発生したときにパイプラインを自動的に開始させます。

Amazon CloudWatch Events (推奨)  
Amazon CloudWatch Events を使用して、変更が発生したときにパイプラインを自動的に開始させます。

AWS CodePipeline  
AWS CodePipeline を使用して、変更を定期的を確認する

キャンセル

戻る

次に

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>



# CodePipeline の操作: パイプラインの作成

- ビルドステージはオプション  
(コンパイル言語の場合は指定)
- ビルドプロバイダーを指定する
- ビルドプロバイダーに紐づく情報を  
指定する (例: リージョン、アプリ  
ケーション名など)

ビルドステージを追加する 情報

### 構築する - オプション

プロバイダーを構築する  
これはビルドプロジェクトのツールです。オペレーティングシステム、ビルドスベックファイル、および出力ファイル名のようなビルドアーティファクトの詳細を提供してください。

AWS CodeBuild

リージョン  
アジアパシフィック (東京)

プロジェクト名  
AWS CodeBuild コンソールで既に作成したビルドプロジェクトを選択します。または AWS CodeBuild コンソールでビルドプロジェクトを作成してこのタスクに戻ります。

myappa × または プロジェクトを作成する

環境変数 - オプション  
CodeBuild 環境変数のキー、値、タイプを選択します。[value] フィールドで、CodePipeline によって生成された変数を参照できます。 [詳細](#)

環境変数の追加

ビルドタイプ

単一ビルド  
単一ビルドをトリガー

バッチビルド  
複数のビルドを1つの実行としてトリガーします。

キャンセル

戻る

ビルドステージをスキップ

次に

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>

# CodePipeline の操作: パイプラインの作成

- デプロイステージはオプション  
(CDを行う場合は指定)
- デプロイプロバイダーを指定する
- デプロイプロバイダーに紐づく情報を  
指定する (例: リージョン、アプリ  
ケーション名など)

デプロイステージを追加する 情報

### デプロイ - オプション

デプロイプロバイダー  
インスタンスにデプロイする方法を選択します。プロバイダーを選択し、プロバイダーの設定の詳細を入力します。

AWS CodeDeploy ▼

リージョン  
アジアパシフィック (東京) ▼

アプリケーション名  
AWS CodeDeploy コンソールで既に作成したアプリケーションを選択します。または AWS CodeDeploy コンソールでアプリケーションを作成してこのタスクに戻ります。

Q myappa X

デプロイグループ  
AWS CodeDeploy コンソールで既に作成したデプロイグループを選択します。または AWS CodeDeploy コンソールでデプロイグループを作成してこのタスクに戻ります。

Q myappa-Env X

キャンセル 戻る 導入手順をスキップ **次に**

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>

# CodePipeline の操作: パイプラインの作成

レビュー 情報

- パイプライン構成を確認し、作成を選択

ステップ 1: パイプラインの設定を選択する

## パイプラインの設定

パイプライン名

MyAppA

Artifact の場所

新しい Amazon S3 バケットがパイプラインのデフォルトアーティファクトストアとして作成されました

サービスロール名

AWSCodePipelineServiceRole-ap-northeast-1-MyAppA

ステップ 4: デプロイステージを追加する

## アクションプロバイダーをデプロイする

Deploy アクションプロバイダー

AWS CodeDeploy

ApplicationName

myappa

DeploymentGroupName

myappa-Env

キャンセル

戻る

パイプラインを作成する

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>

# CodePipeline の操作: パイプラインの編集

- パイプラインの編集で、作成したステージやアクションを編集可能
- 後からアクションやステージを追加可能

Editing: MyAppA

削除する

キャンセル

保存する

編集する: Source キャンセル 完了

+ アクショングループを追加する

Source ⓘ + アクションの追加

☑ ×

+ アクショングループを追加する

+ ステージを追加する

編集する: Build ステージを編集する

:

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html>

# CodePipeline の操作: パイプラインの実行開始、停止


- パイプラインの開始・停止が可能。
- リリースに問題が含まれていることに気付いた場合、それ以上のデプロイからリリースをすぐに停止可能に。

The screenshot displays the AWS CodePipeline console for a pipeline named 'MyAppA'. At the top, there are several action buttons: '通知' (Notifications), '編集する' (Edit), '実行を停止' (Stop execution), 'パイプラインをクローンする' (Clone pipeline), and '変更をリリースする' (Release change). Below these buttons, the pipeline is shown in a '進行中' (In progress) state. The 'Source' stage is currently active, with a status of '実行中' (In progress) and a sub-status of '実行しませんでした' (Did not execute). The 'Build' stage is also shown, with a status of '進行中' (In progress) and a sub-status of '進行中 - 今' (In progress - now). A '移行を無効にする' (Disable transition) button is visible between the two stages. The console also shows the pipeline execution ID and the source commit information.

<https://aws.amazon.com/jp/about-aws/whats-new/2020/01/aws-codepipeline-enables-stopping-pipeline-executions/>

# CodePipeline の操作: パイプラインの実行履歴を表示

- パイプラインで最後に実行されたアクション、ステージ間のトランジション、失敗した結果など、詳細を表示する
- 実行履歴は最大12ヶ月間保存



実行履歴 情報

実行を停止 詳細を表示

Q

実行 ID	ステータス	ソースリビジョン	期間	完了しました
5ee7bbdd-e525-4f10-a044-38466f2d3e8a	⊗ 失敗しました	Source - 92d76169: Initial commit by AWS CodeCommit	1 分間 55秒間	10月 31, 2020 2:15 午後 (UTC+9:00)
ea8dc1fd-c137-4e67-a67e-691a97a2dc76	⊗ 失敗しました	Source - 92d76169: Initial commit by AWS CodeCommit	1 分間 8秒間	10月 31, 2020 2:14 午後 (UTC+9:00)
c21fc67c-2a9c-4b13-9db4-a16762a2dc30	⊗ 失敗しました	Source - 92d76169: Initial commit by AWS CodeCommit	1 分間 8秒間	10月 30, 2020 11:50 午前 (UTC+9:00)

# CodePipeline の操作: パイプラインの実行結果の通知

- リポジトリ、ビルドプロジェクト、デプロイ、およびパイプラインのイベントに関する通知を送信する
- 通知はAmazon SNSで送信される

## 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。

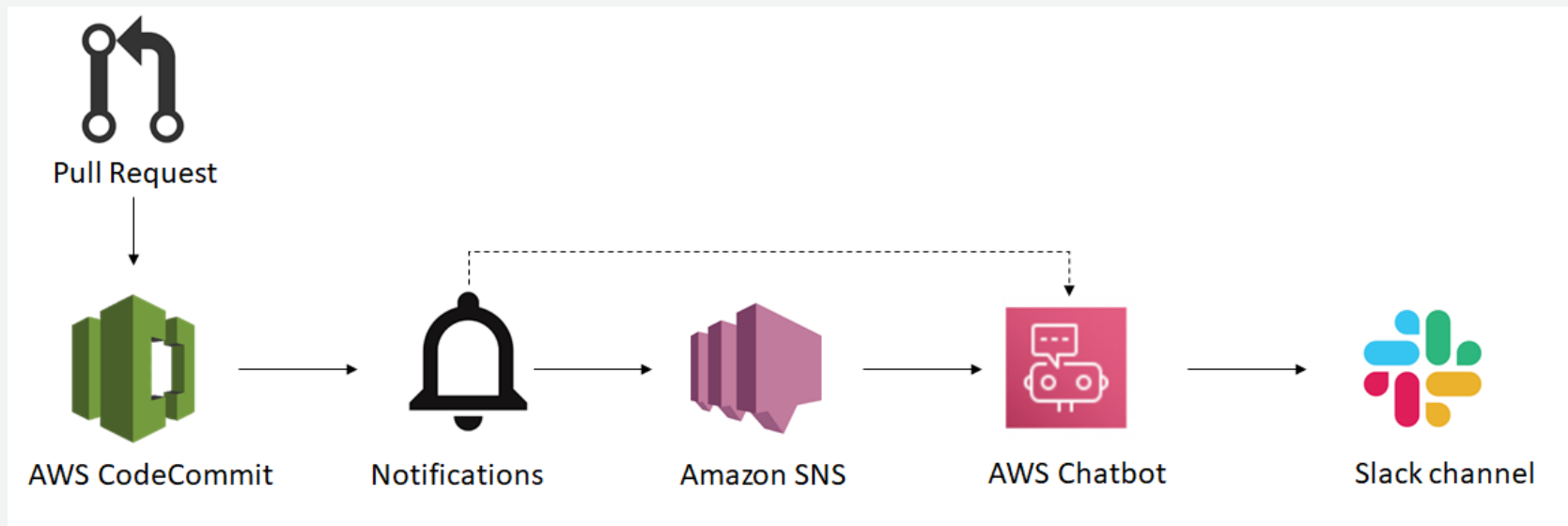
ベーシック  
リソースイベントで提供される情報のみが含まれます。

### 通知をトリガーするイベント

Action execution	Stage execution	Pipeline execution	Manual approval
<input type="checkbox"/> Succeeded	<input type="checkbox"/> Started	<input type="checkbox"/> Failed	<input type="checkbox"/> Failed
<input type="checkbox"/> Failed	<input type="checkbox"/> Succeeded	<input type="checkbox"/> Canceled	<input type="checkbox"/> Needed
<input type="checkbox"/> Canceled	<input type="checkbox"/> Resumed	<input type="checkbox"/> Started	<input type="checkbox"/> Succeeded
<input type="checkbox"/> Started	<input type="checkbox"/> Canceled	<input type="checkbox"/> Resumed	
	<input type="checkbox"/> Failed	<input type="checkbox"/> Succeeded	
		<input type="checkbox"/> Superseded	

<https://docs.aws.amazon.com/dtconsole/latest/userguide/welcome.html>

# 参考資料: ビルドやテスト結果をSlack等に通知する



<https://aws.amazon.com/jp/blogs/news/receive-aws-developer-tools-notifications-over-slack-using-aws-chatbot/>



# 参考資料: ビルドやテスト結果をSlack等に通知する

- AWS Chatbot を利用して AWS 開発者用ツールの通知を Slack で受け取る方法  
<https://aws.amazon.com/jp/blogs/news/receive-aws-developer-tools-notifications-over-slack-using-aws-chatbot/>
- ウェブフックを使用して Amazon SNS メッセージを Amazon Chime、Slack や Microsoft Teams に発行する方法を教えてください。  
<https://aws.amazon.com/jp/premiumsupport/knowledge-center/sns-lambda-webhooks-chime-slack-teams/>

# AWS CodePipeline がサポートしているプロバイダー

## • Source

- AWS CodeCommit
- Amazon ECR
- Amazon S3
- Bitbucket
- GitHub
- GitHub Enterprise Server

## • Build

- AWS CodeBuild
- CloudBees
- Jenkins
- TeamCity

## • Test

- AWS CodeBuild
- AWS Device Farm
- BlazeMeter
- Jenkins
- GhostInspector
- Micro Focus StormRunner Load
- Novovola
- Runscope

[https://docs.aws.amazon.com/ja\\_jp/codepipeline/latest/userguide/integrations-action-type.html](https://docs.aws.amazon.com/ja_jp/codepipeline/latest/userguide/integrations-action-type.html)

<https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html>

# AWS CodePipeline がサポートしているプロバイダー

- Deploy

- AWS CodeDeploy
- AWS CloudFormation
- Amazon S3
- Amazon ECS
- Amazon ECS (Blue/Green)
- Elastic Beanstalk
- AWS AppConfig
- AWS OpsWorks
- AWS Service Catalog
- Amazon Alexa
- XebiaLabs

- Approval

- Manual

- Invoke

- AWS Lambda
- AWS Step Functions

[https://docs.aws.amazon.com/ja\\_jp/codepipeline/latest/userguide/integrations-action-type.html](https://docs.aws.amazon.com/ja_jp/codepipeline/latest/userguide/integrations-action-type.html)  
<https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html>

# AWS CodePipeline と統合可能な製品



**GitHub**

cloud  
bees



**BlazeMeter**

**XebiaLabs**  
Deliver Faster



**Jenkins**

**Ghost Inspector**

**TeamCity**

**MICRO  
FOCUS**

**Runscope**

<https://aws.amazon.com/jp/codepipeline/product-integrations/>

# Custom アクション

デフォルトで用意されていない独自のアクションを追加可能

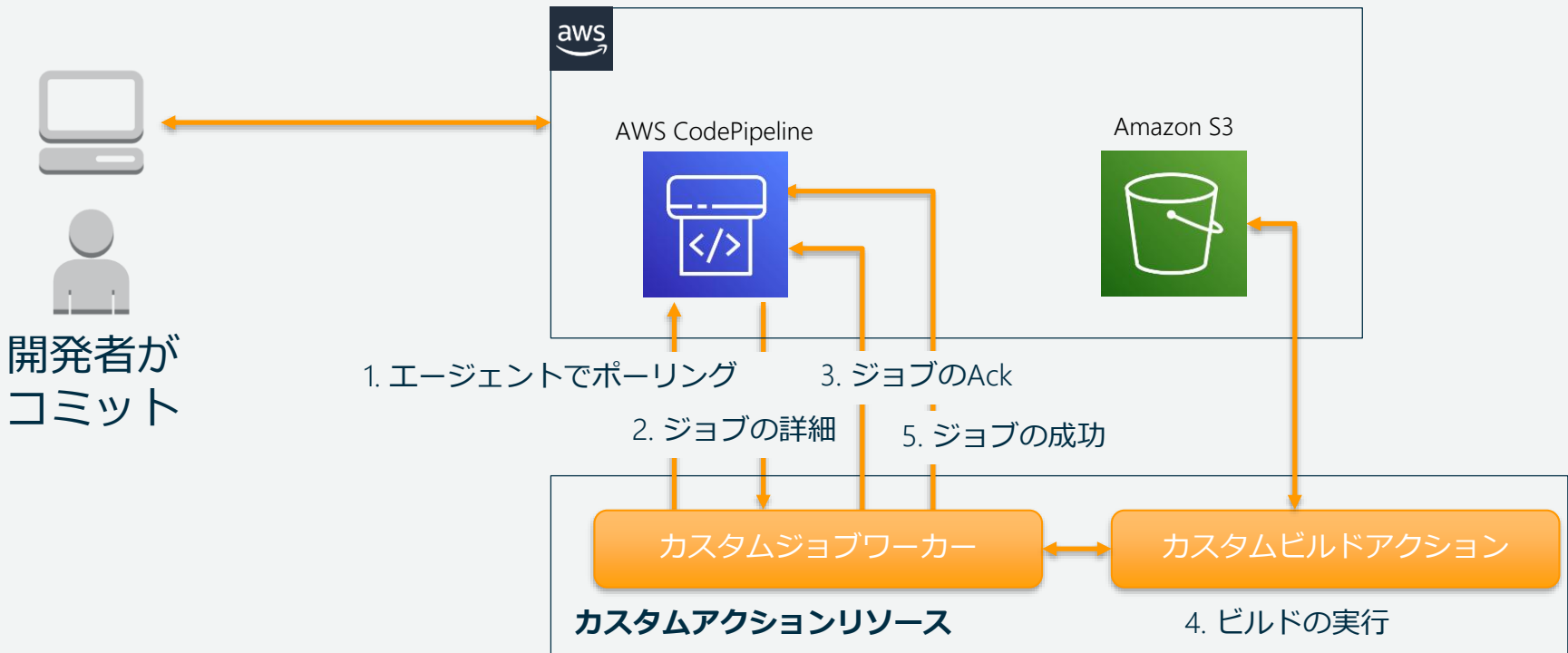
AWS CLI を使ってパイプラインにカスタムアクションを作成

Job Worker の作成が必要

- Customアクションに対するAWS CodePipelineのジョブリクエストをポーリング
- Custom Jobの実行
- 結果のステータスをAWS CodePipelineに返す
- AWS CodePipelineから参照可能なパブリックなエンドポイントを公開

[http://docs.aws.amazon.com/ja\\_jp/codepipeline/latest/userguide/actions-create-custom-action.html](http://docs.aws.amazon.com/ja_jp/codepipeline/latest/userguide/actions-create-custom-action.html)

# Custom アクション



# AWS Lambda ファンクションの実行

AWS CodePipeline のアクションとして Lambda を呼び出し可能

Lambda ファンクションの利用例:

- AWS CloudFormation を使用して、任意のタイミングでリソース作成・削除
- Lambda ファンクション で CNAME をスワップすることでゼロダウンタイムでのバージョンアップを ElasticBeanstalk で実行
- AMI スナップショットを作成することで構築、デプロイする前にリソースのバックアップを作成
- IRC クライアントにメッセージをポストするなど、サードパーティーの製品と統合

<https://docs.aws.amazon.com/codepipeline/latest/userguide/action-reference-Lambda.html>

# AWS Step Function の実行

AWS CodePipelineのアクションとして AWS Step Function を呼び出し可能

リリースプロセスの一部として複雑なワークフローを呼び出すことができる

AWS Step Function の利用例:

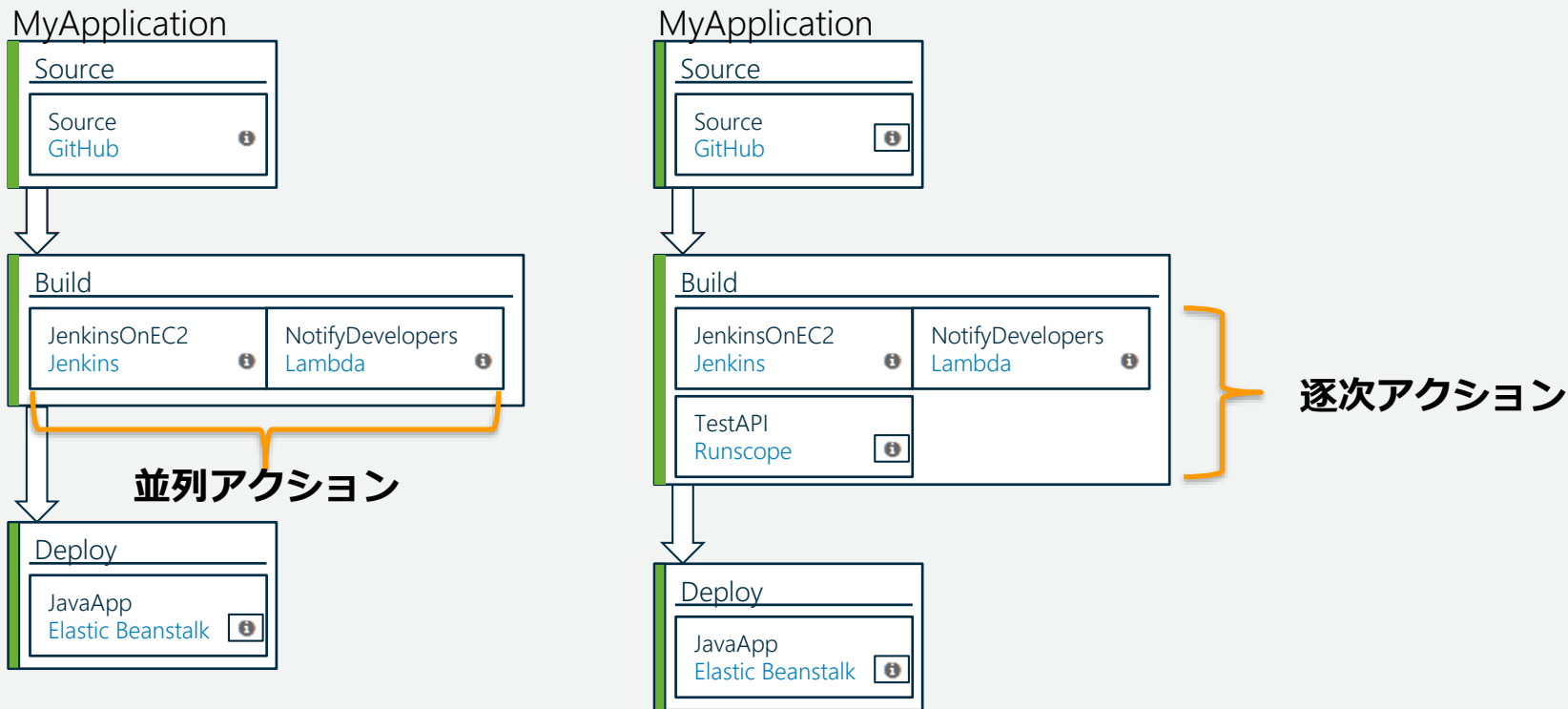
- 条件付き分岐、エラー処理、および非同期タスクを行う処理
- 他の AWS のサービスとの連携
- シンプルなリリースパイプラインを維持し、複雑なワークフローの動作を Step Functions ステートマシンエンジンに委任する

<https://docs.aws.amazon.com/codepipeline/latest/userguide/action-reference-StepFunctions.html>





# 並列アクションと逐次アクション



# 実行時のアクション間の変数引き渡し

パイプライン内でアクション間の変数受け渡しが可能に。

従来はアクション間で受け渡しは出来なかったが、変数を元に評価をすることで、よりダイナミックなアクションが記述可能となった。

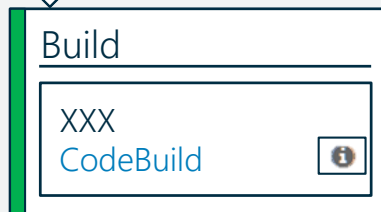
利用例:

- CodeCommit ソースアクションが出力するコミットメッセージを元に、手動認証操作のためのカスタムメッセージを定義する
- エンドポイント URL の生成をデプロイステージで行い、その後続くテストアクションでテストする
- カスタムワークフローにおいては、Lambda アクションが出力する変数を、それに呼応するアクションで自由に使用できる

<https://aws.amazon.com/jp/about-aws/whats-new/2019/11/aws-codepipeline-enables-passing-variables-between-actions-at-execution-time/>

# 構成例: Amazon ECS での CI/CD パイプライン

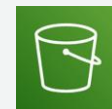
MyApplication



AWS CodeCommit

2. ソースアーティファクトを格納

ソースアーティファクト



Amazon S3

4. ソースアーティファクト  
を取得

5. ビルドアーティファクト  
を格納



Amazon ECR  
ビルドアーティファクト



AWS CodeBuild

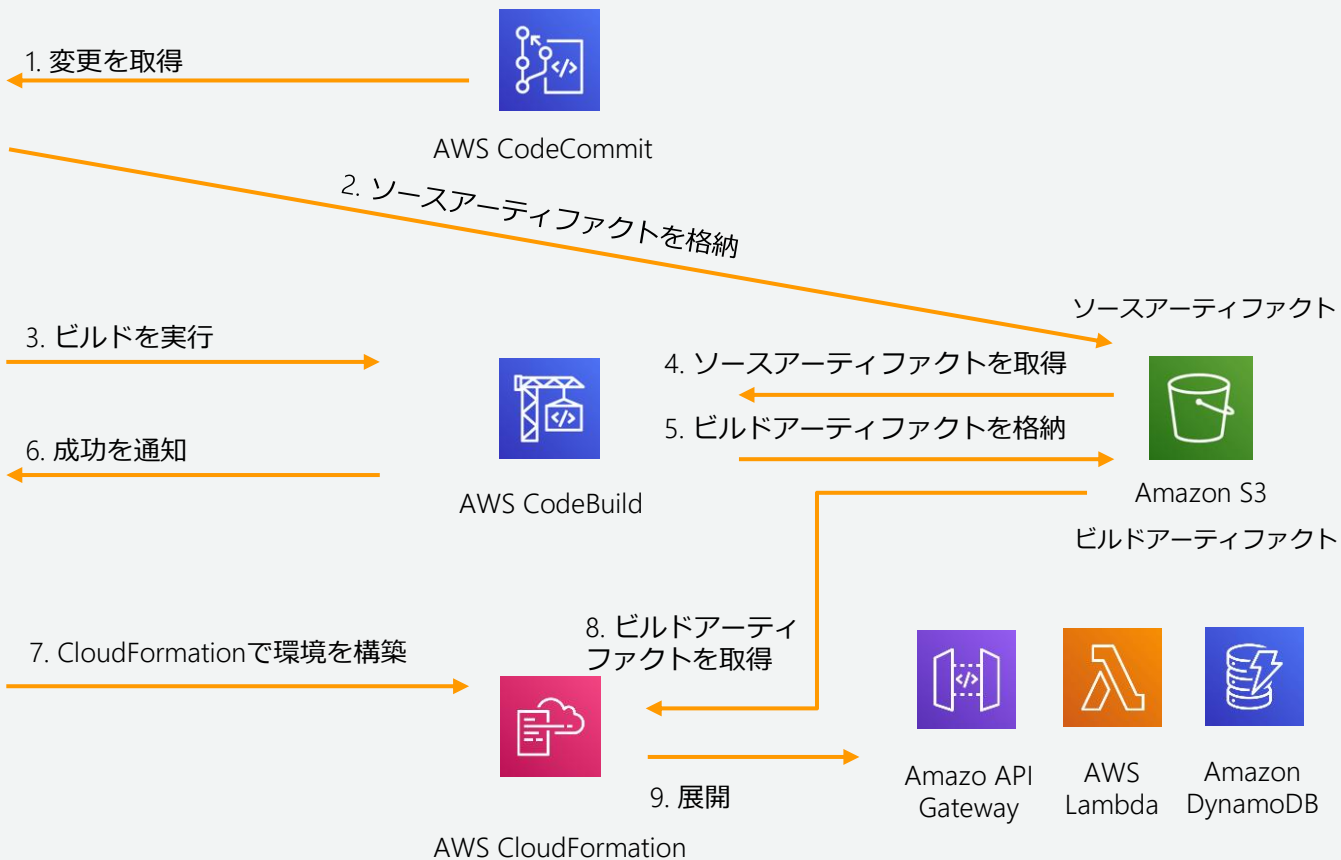
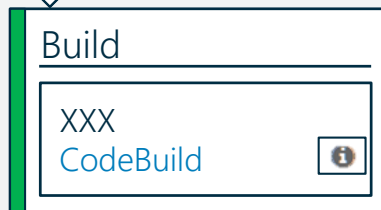


Amazon ECS

8. ビルドアーティ  
ファクトを取得

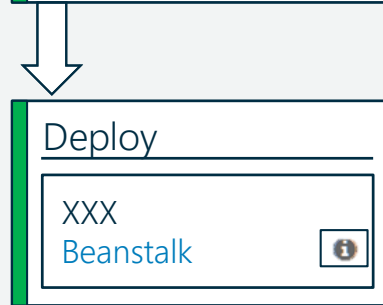
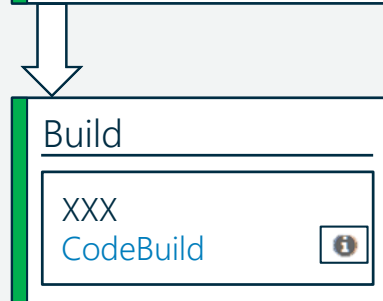
# 構成例: サーバーレスでの CI/CD パイプライン

MyApplication



# 構成例: GitHub, Jenkins との連携

MyApplication

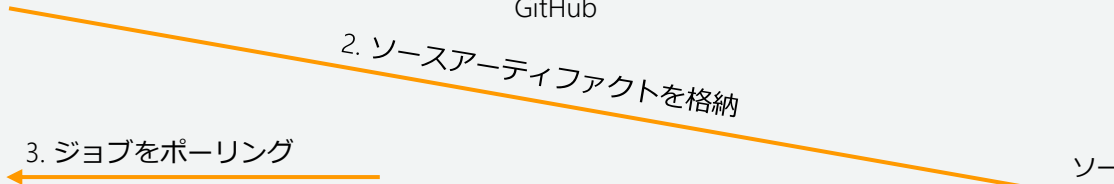


GitHub

1. 変更を取得

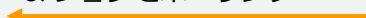


2. ソースアーティファクトを格納

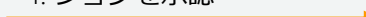


ソースアーティファクト

3. ジョブをポーリング



4. ジョブを承認



7. 成功を通知



5. ソースアーティファクトを取得

6. ビルドアーティファクトを格納



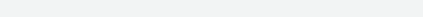
Amazon S3

ビルドアーティファクト

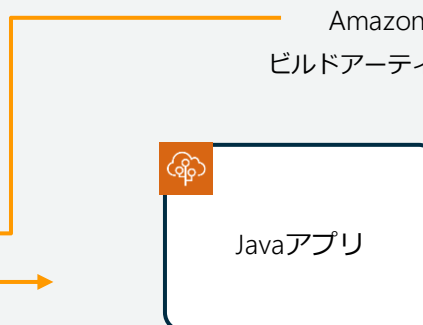


Jenkin

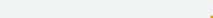
8. AWS Elastic Beanstalkで環境を構築



9. ビルドアーティファクトを取得



10. 展開



AWS Elastic Beanstalk



Javaアプリ

# AWS CodePipeline のクォータ

Resource	デフォルト値
アクションがタイムアウトするまでの時間	Approvalアクション: 7 日間 AWS CloudFormation デプロイアクション: 3 日間 CodeBuild ビルドアクションおよびテストアクション: 8 時間 CodeDeploy および CodeDeploy ECS (blue/green) デプロイアクション: 5 日間 AWS Lambda の呼び出しアクション: 24 時間
AWS アカウントのリージョンあたりの全パイプラインの最大数	300
AWS リージョンごとにソース変更のポーリングに設定するパイプラインの最大数	60
AWS アカウントのリージョンあたりのウェブフックの最大数	300
AWS アカウントのリージョンあたりのカスタムアクションの数	50

[https://docs.aws.amazon.com/ja\\_jp/codepipeline/latest/userguide/limits.html](https://docs.aws.amazon.com/ja_jp/codepipeline/latest/userguide/limits.html)

# AWS CodePipeline の料金

アクティブパイプライン\* 1つに対して、1.00 USD / 月

- 実験を奨励するため、パイプラインは作成後の最初の 30 日間は無料
- アクティブパイプライン\*とは、30日以上存在していて、その月に1つのコード変更が発生したパイプライン
- その月に新しいコード変更がないパイプラインに対しては料金は発生しません
- 1ヶ月に満たない分に対しては按分計算されません

## 無料利用枠

- アクティブなパイプラインを 1 か月あたり 1 つ無料

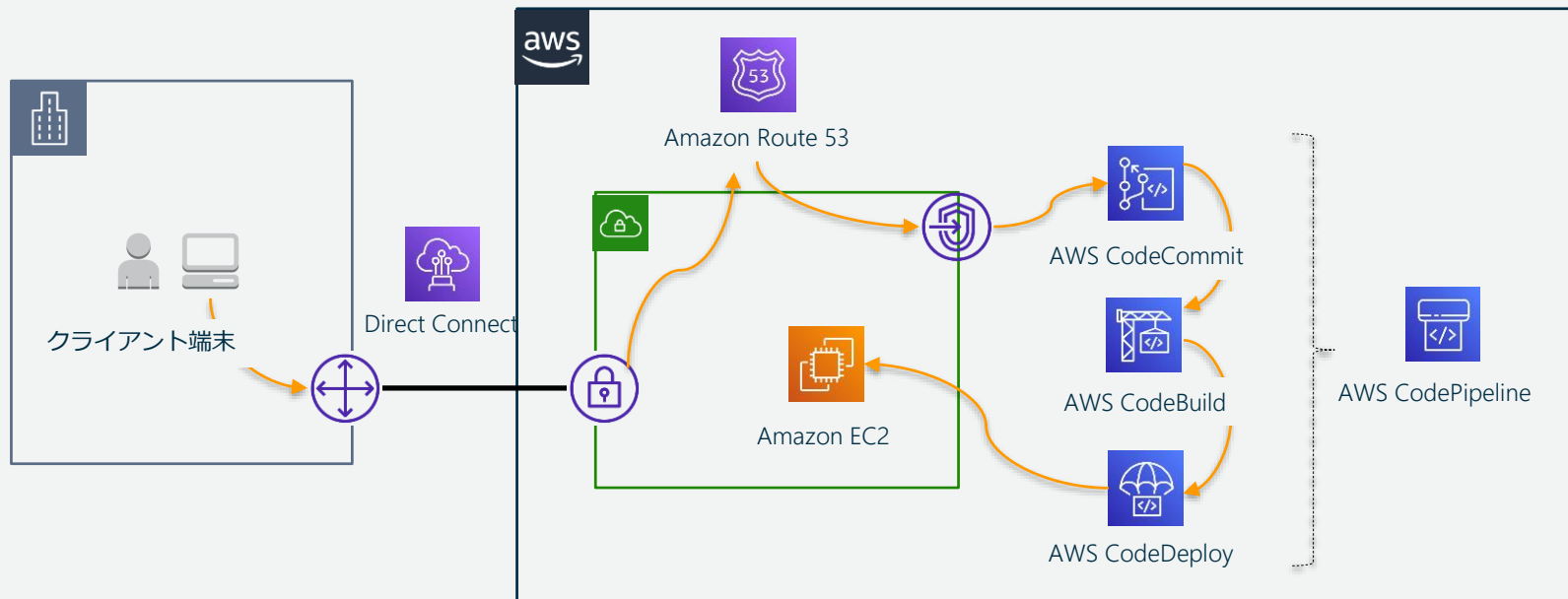
<https://aws.amazon.com/jp/codepipeline/pricing/>



# 本日のアジェンダ

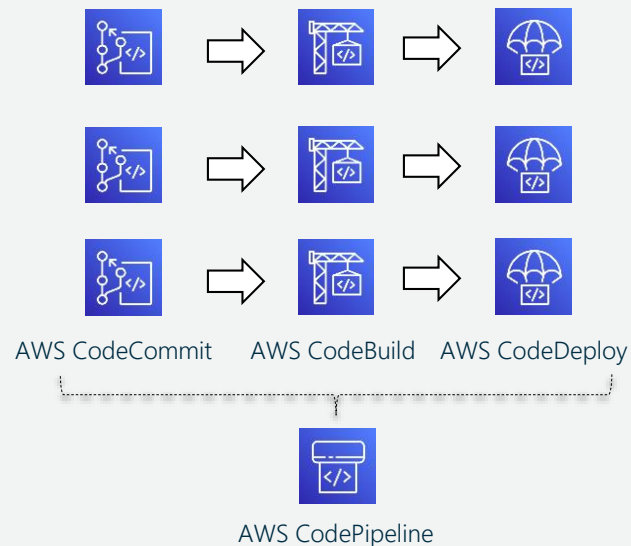
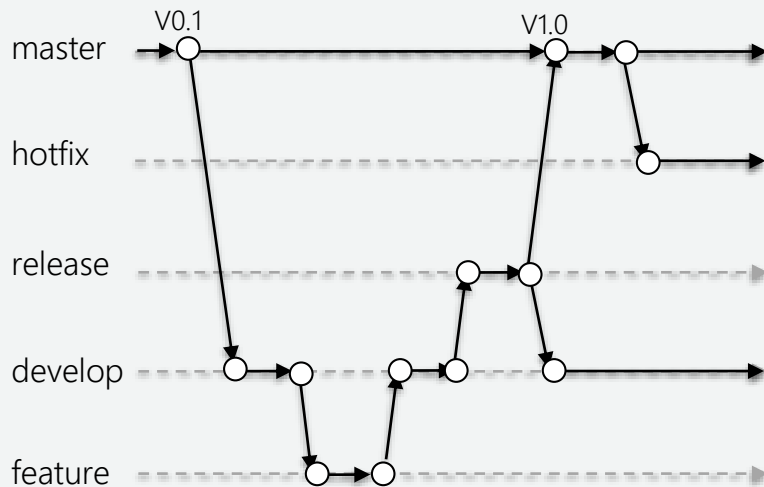
- CI/CD が必要となる背景
- AWS CodeStar の機能説明
- AWS CodePipeline の機能説明
- **よくあるご相談**
- まとめ

# インターネットを経由せずにパイプラインを作るには？



VPC エンドポイントを経由するとインターネットにアクセスすることなくサービスと連携が可能。  
AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy は VPC エンドポイントに対応している。  
クライアント端末から VPC エンドポイント経由で AWS CodeCommit にコミットする。  
AWS CodePipeline 内はAWSサービス間のため閉域網で連携する。

# 開発・検証・本番のパイプラインをどう作れば良いか？



基本的には、ブランチ毎にパイプラインを分けて構築する。  
環境別の差分は環境変数として定義し、CodePipelineから設定する。

<https://aws.amazon.com/jp/blogs/devops/implementing-gitflow-using-aws-codepipeline-aws-codecommit-aws-codebuild-and-aws-codedeploy/>

# 参考資料: AWS Developer Tools を活用した Git-flow

**DOP202-R**

## Implementing GitFlow with AWS Tools

**Ashish Gore**  
Sr. Technical Account Manager  
Amazon Web Services

**Amit Jha**  
Sr. Solutions Architect  
Amazon Web Services

aws  
reInvent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Search...

- Introduction
- Prerequisites
- Master Branch
- Lambda
- Develop Branch
- Feature Branch
- Cleanup
- Conclusion

Privacy | Site Terms | © 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## AWS TOOLS GITFLOW WORKSHOP

### Implementing GitFlow with AWS tools

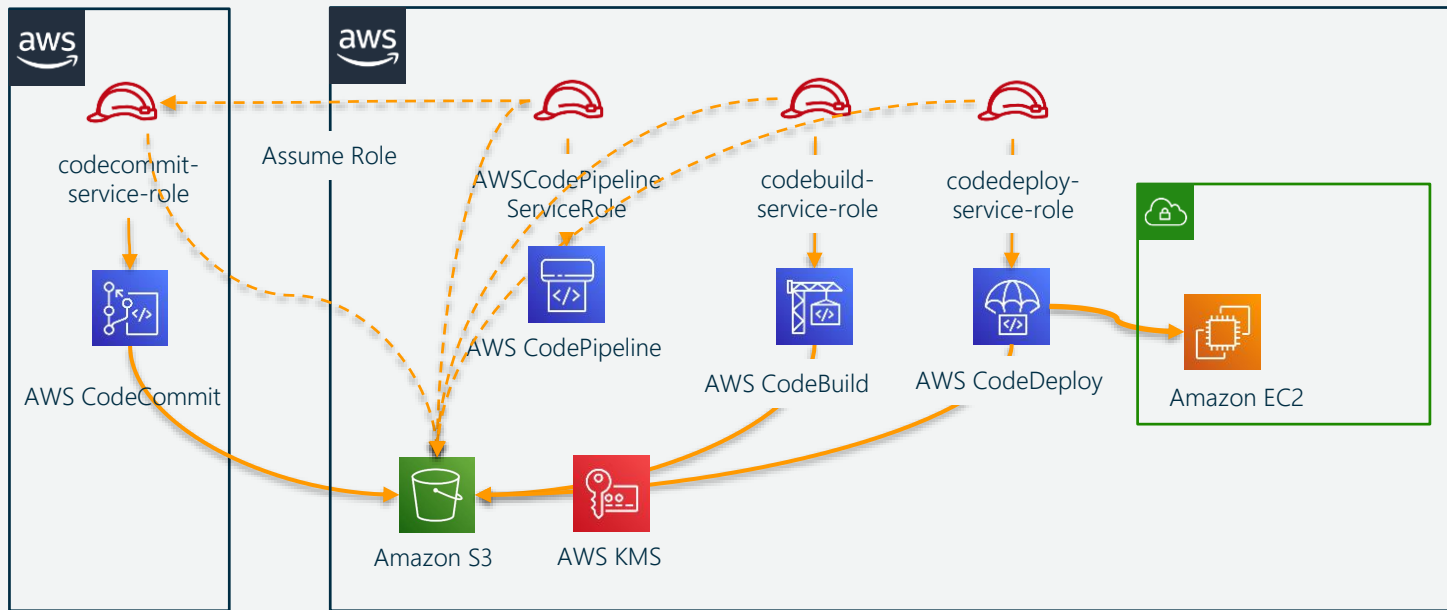
In this workshop, you'll learn about high-level frameworks for how to implement GitFlow using AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, and AWS CodeDeploy. You will also have the opportunity to walk through a prebuilt example and examine how the framework can be adopted for individual use cases.



[https://d1.awsstatic.com/events/reinvent/2019/REPEAT\\_2\\_1\\_mplementing\\_GitFlow\\_with\\_AWS\\_tools\\_DOP202-R2.pdf](https://d1.awsstatic.com/events/reinvent/2019/REPEAT_2_1_mplementing_GitFlow_with_AWS_tools_DOP202-R2.pdf)

<https://gitflow-codetools.workshop.aws/>

# クロスアカウントでパイプラインを作るには？



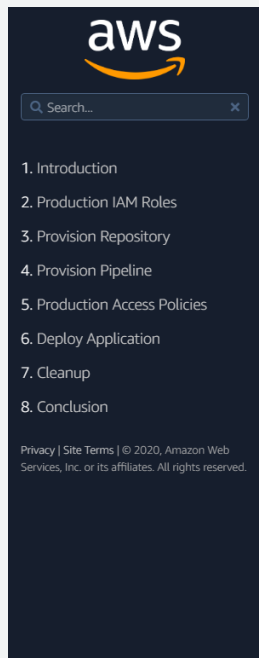
Assume Role でアクセス権限を委任することで連携が可能。

注意点として、アカウントを跨いでアーティファクトを受け渡すため、S3バケットならびにAWS KMSキーをクロスアカウントアクセスを有効にする必要がある。またクロスアカウントパイプラインの制限内でアクセスできるように構成を組む必要がある。

<https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create-cross-account.html>

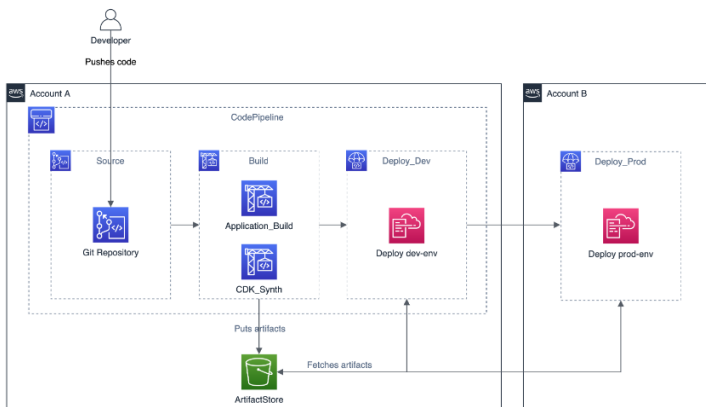
<https://aws.amazon.com/jp/blogs/devops/aws-building-a-secure-cross-account-continuous-delivery-pipeline/>

# 参考資料: Cross-Account CI/CDパイプラインワークショップ



The image shows a dark-themed navigation menu from the AWS console. At the top is the AWS logo and a search bar. Below is a numbered list of eight items: 1. Introduction, 2. Production IAM Roles, 3. Provision Repository, 4. Provision Pipeline, 5. Production Access Policies, 6. Deploy Application, 7. Cleanup, and 8. Conclusion. At the bottom, there is a small footer with 'Privacy | Site Terms | © 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

## BUILDING A CROSS-ACCOUNT CI/CD PIPELINE



<https://cross-account-cicd-pipeline.workshop.aws/>

# 本日のアジェンダ

- CI/CD が必要となる背景
- AWS CodeStar の機能説明
- AWS CodePipeline の機能説明
- よくあるご相談
- **まとめ**

# まとめ

- AWS CodeStar は、テンプレートベースで CI/CD 環境を素早く構築でき、ダッシュボードで一元管理できる。CI/CD 環境を試してみたい方はオススメ。
- AWS CodePipeline は、リリースプロセス全体を一貫したやり方で完全に自動化できる。AWS が揃えているソース、ビルド、デプロイのサービス群とできるほか、サードパーティー製品とも連携ができる。
- 急速なビジネススピードに応えるためには、CI/CDは必須。  
これらサービスを活用することで高品質なソフトウェアをインクリメンタルな開発手法で素早くリリース！



# 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



The screenshot shows the AWS Japanese website header with the AWS logo, navigation links for 'お問い合わせ', 'サポート', '日本語', and 'アカウント', and a 'サインイン' button. Below the header is a navigation bar with links for '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', and 'その他'. The main content area features the title 'AWS クラウドサービス活用資料集トップ' and a paragraph of introductory text. At the bottom, there are four buttons: 'AWS Webinar お申込', 'AWS 初心者向け', '業種・ソリューション別資料', and 'サービス別資料'.

aws

日本担当チームへお問い合わせ サポート 日本語 アカウント

コンソールにサインイン

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

## AWS クラウドサービス活用資料集トップ

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

AWS Webinar お申込 »

AWS 初心者向け »

業種・ソリューション別資料 »

サービス別資料 »

<https://amzn.to/JPArchive>

# AWS Well-Architected 個別技術相談会

## 毎週“W-A個別技術相談会”を実施中

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

- 申込みはイベント告知サイトから

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

AWS イベント で[検索]



AWS Well-Architected



# ご視聴ありがとうございました

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>

