

AWS のインフラを管理する方法

～ 管理でおこりがちな問題を AWS CDK で解決してみましよう ～

水流 洋人

アマゾン ウェブ サービス ジャパン合同会社
AWS プロフェッショナルサービス本部
クラウドアプリケーションアーキテクト

自己紹介



水流 洋人 (つる ひろひと)

クラウド アプリケーション アーキテクト

様々な業種のお客様に、AWS 上での
モダンアプリケーション開発の導入や人材育成を支援

好きな AWS サービス

- AWS CDK
- AWS Lambda
- Amazon DynamoDB

本セッションの対象となる方

- 下記に該当する、インフラ担当者・アプリ担当者
 - ✓ AWS を利用して間もない
 - ✓ AWS の環境構築やリソースの運用に課題感を持っている

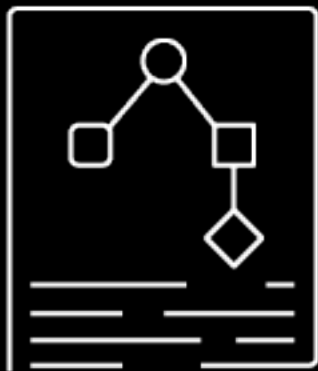
アジェンダ

- AWS のインフラ管理でおこりがちな問題と解決法
- AWS でのインフラコード管理方法と比較
- AWS CDK の始め方
- AWS CDK の便利な使い方
- まとめ

AWS のインフラ管理で おこりがちな問題と解決法

従来のインフラ管理

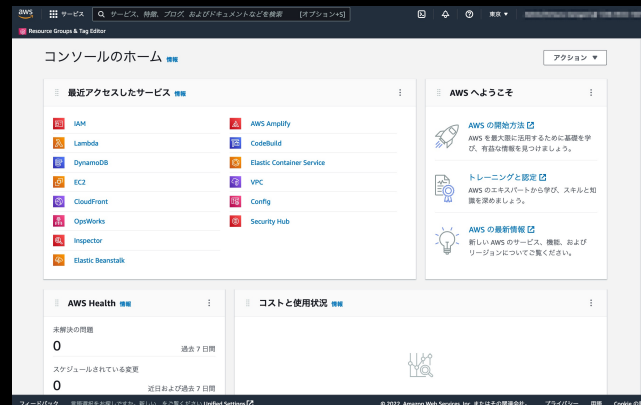
設計



ドキュメント化



構築



システムの要件にあわせて
インフラ、リソースの構成
を設計

設計した内容を設計書や
パラメータシートとして
ドキュメント化

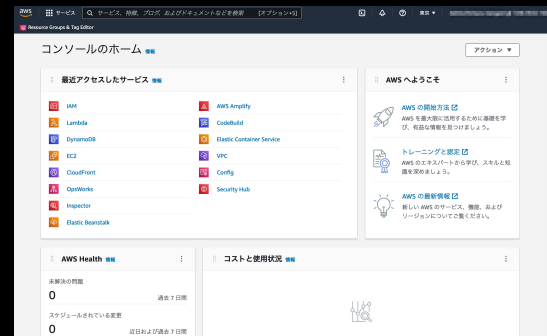
設計書やパラメータシート
をみながら、マネジメント
コンソールからリソースを
手動で作成

おこりがちな問題

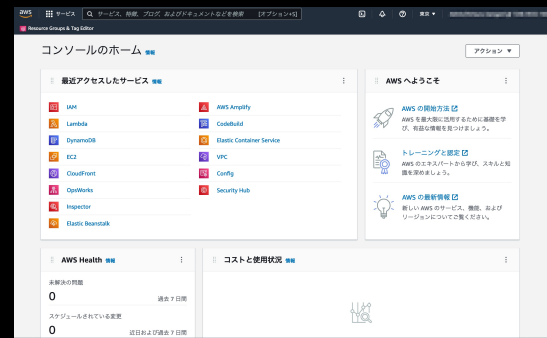
設計書
パラメータシート



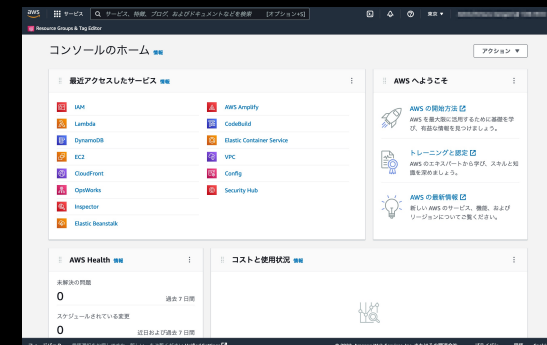
開発環境



ステージング環境



本番環境



特定の環境でアプリケーションが動かない原因

ヒューマンエラー

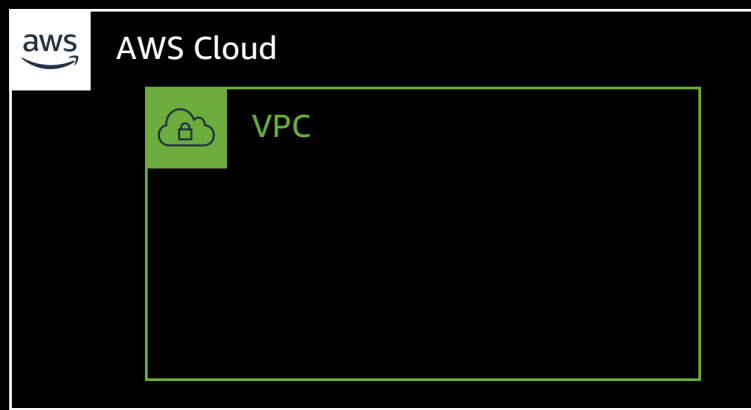
設計書やパラメータシートに従って画面操作したつもりでも、設定誤りや、設定漏れを起こすことはある



特定の環境でアプリケーションが動かない原因

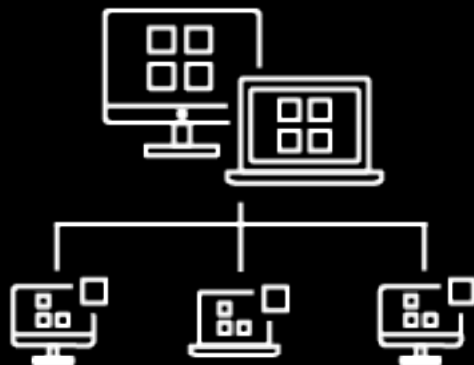
ドキュメント誤り

- (例) ・ ステージング環境で外部システムとの連携テスト中に接続できない問題が発生
- ・ 調査や検証の結果ネットワークセキュリティの設定の問題と判明し、設定の変更をおこなった



特定の環境でアプリケーションが動かない原因

ドキュメント誤り (例)



≠



- 行った設定変更を、他の環境にも反映させることを優先
- ドキュメントへの反映を忘れてしまい、ドキュメントと環境との差異が発生

リソースをコードで管理するという解決法

Infrastructure as Code (IaC) と呼ぶ



- コードからリソースの生成を行うため、**ミスが起こりにくい**
- コードの変更履歴から、**いつ、誰が、**
どんな修正を行ったかを追跡しやすい

リソースをコードで管理するうえでの考慮点



- 環境の変更はマネジメントコンソールから直接行わずに、必ずコードを修正して、コードから反映を行う
- IaC のコードをコードリポジトリで管理する

AWS でのインフラコード 管理方法と比較

インフラをコードで管理するサービス・ツール



AWS CloudFormation



AWS Serverless Application
Model (AWS SAM)



AWS Cloud Development
Kit (AWS CDK)

インフラをコードで管理するサービス・ツール



AWS CloudFormation

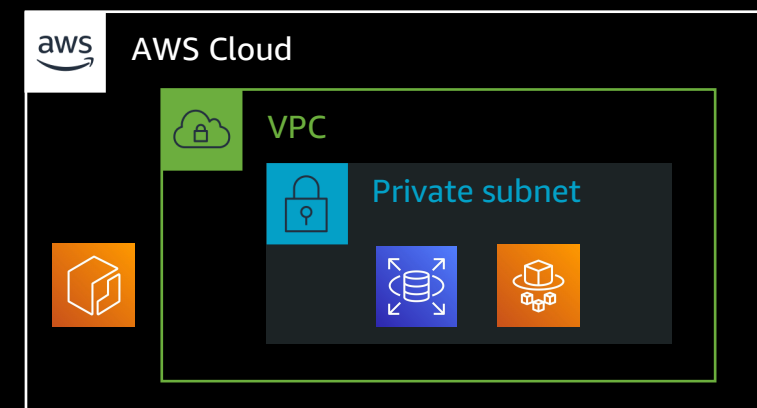
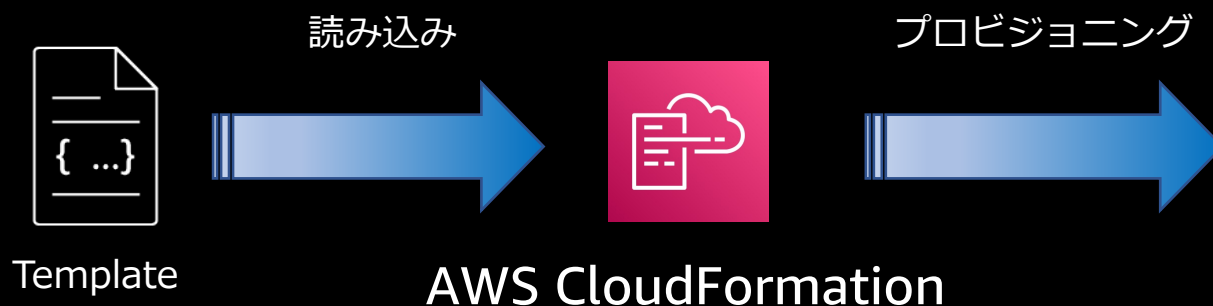


AWS Serverless Application
Model (AWS SAM)



AWS Cloud Development
Kit (AWS CDK)

- JSON や YAML 形式のテンプレートファイルから、AWS リソースをプロビジョニングできるサービス



インフラをコードで管理するサービス・ツール



AWS CloudFormation

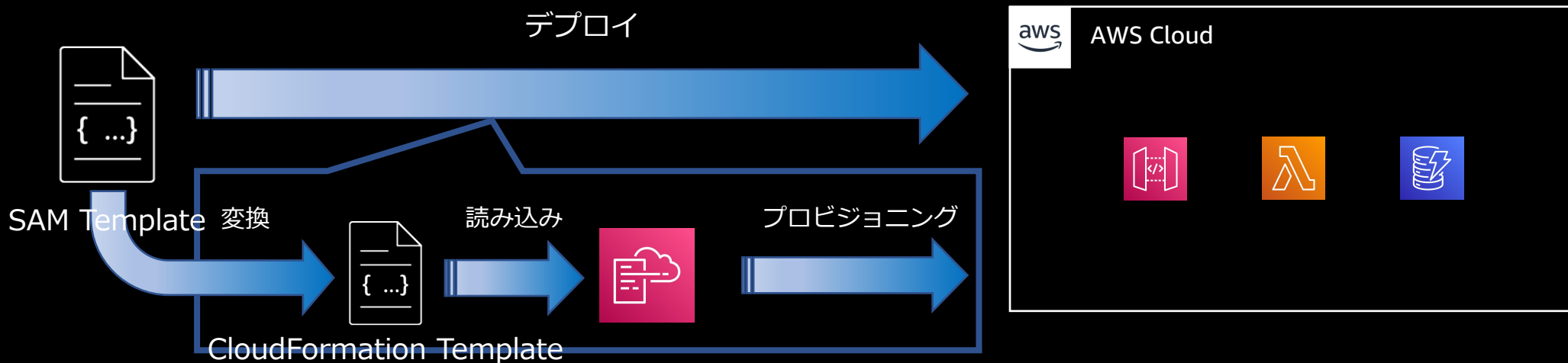


AWS Serverless Application
Model (AWS SAM)



AWS Cloud Development
Kit (AWS CDK)

- サーバーレスアプリケーションを構築するためのオープンソースのフレームワーク



インフラをコードで管理するサービス・ツール



AWS CloudFormation

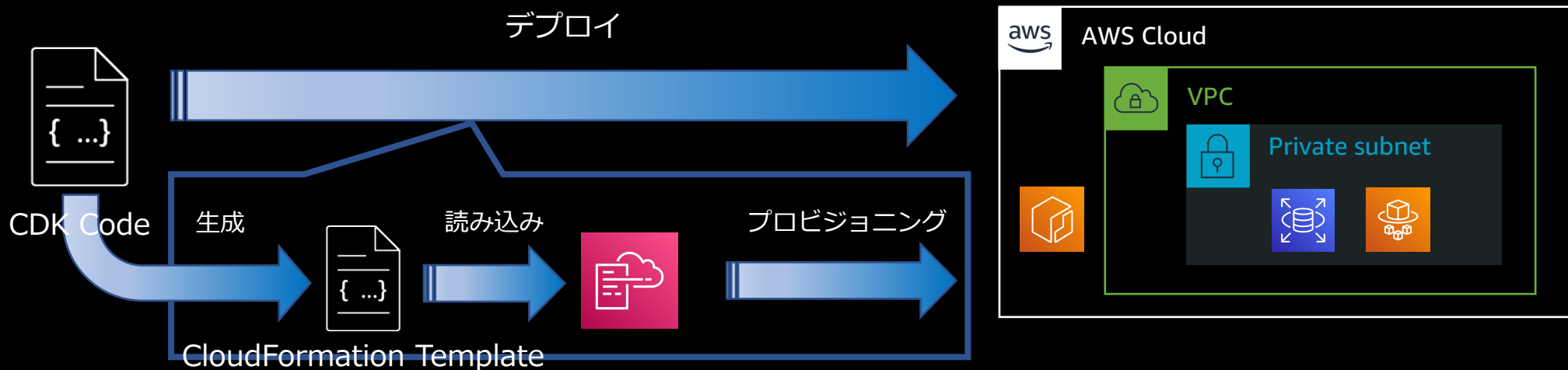


AWS Serverless Application
Model (AWS SAM)



AWS Cloud Development
Kit (AWS CDK)

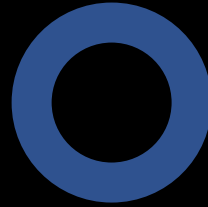
- プログラミング言語で AWS リソースを定義することができるオープンソースのフレームワーク



サービス・ツールごとのメリット・デメリット



AWS CloudFormation



- サービスとしての安定稼働の実績があり、多くの情報・ノウハウをインターネットから取得することができる
- テンプレートを作成するだけで、専用のツールをインストールすることなく簡単に始められる

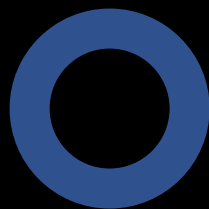


- 繰り返しを表現することができないので、類似構成のリソースのプロビジョニングが大変
- 記述量が多い

サービス・ツールごとのメリット・デメリット



AWS SAM



- サーバーレスアプリケーションに最適化されており、開発からデプロイまでを統合管理することが容易
- デプロイ前に、ローカルテストに便利なツールが含まれている



- 繰り返しを表現することができないので、類似構成のリソースのプロビジョニングが大変
- サーバーレスアプリケーションのデプロイ、サービスのプロビジョニングに特化されている

サービス・ツールごとのメリット・デメリット



AWS CDK






- 使い慣れたプログラミング言語でリソース定義が可能
- デプロイ前に、プロビジョニングするリソース構成のテストが可能



- プログラミング言語を覚える必要がある

サービス・ツールごとのメリット・デメリット

	メリット	デメリット
AWS CloudFormation 	<ul style="list-style-type: none">2011 年から提供されており、サービスとしての安定稼働の実績があり、多くの情報・ノウハウをインターネットから取得することができるテンプレートを作成するだけで、ツールを追加でインストールすることなく簡単にはじめられる	<ul style="list-style-type: none">繰り返しを表現することができないので、類似構成のリソースのプロビジョニングが大変記述量が多い
AWS SAM 	<ul style="list-style-type: none">サーバーレスアプリケーションに最適化されており、開発からデプロイまでを統合管理することが容易デプロイ前に、ローカルテストに便利なツールが含まれている	<ul style="list-style-type: none">繰り返しを表現することができないので、類似構成のリソースのプロビジョニングが大変サーバーレスアプリケーションのデプロイ、サービスのプロビジョニングに特化されている
AWS CDK 	<ul style="list-style-type: none">使い慣れたプログラミング言語でリソース定義が可能デプロイ前に、プロビジョニングするリソース構成のテストが可能	<ul style="list-style-type: none">プログラミング言語を覚える必要がある

AWS CDK の始め方

AWS CDK でリソース構築するまでの流れ

1. AWS CLI のインストール
2. AWS CDK CLI のインストール
3. AWS アカウントのブートストラップ
4. CDK プロジェクトの作成
5. インフラ作成コードの実装
6. デプロイ

1. AWS CLI のインストール



- AWS CDK の使用には、AWS コマンドラインインタフェース (CLI) が必要
- 事前に、AWS のクレデンシヤル (認証情報) をクライアントに設定しておく

参考: https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/getting-started-install.html

2. AWS CDK CLI のインストール



- AWS CDK CLI のインストールには、Node 及び Node Package Manager (npm) のインストールが事前に必要

```
npm install -g aws-cdk
```



AWS Cloud9

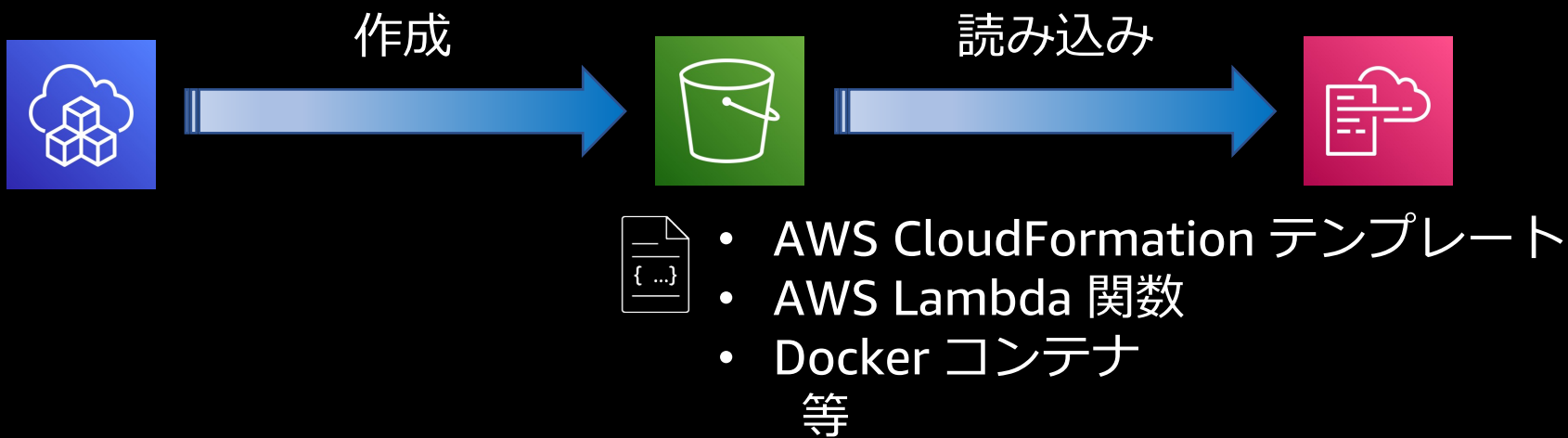
AWS Cloud9 では、AWS CDK がすぐに使用できる状態になっているため便利

3. AWS アカウントのブートストラップ



- AWS CDK がデプロイプロセスを実行するために必要なリソースを予め作成する

cdk bootstrap



4. CDK プロジェクトの作成



- プロジェクトの雛形を作成する

```
cdk init --language [ 使用言語 ]
```

AWS CDK で使用言語オプションに指定可能な言語 (2022/06 現在)

オプション名	言語
javascript	JavaScript
typescript	TypeScript
java	Java
csharp	C#
go	Go

5. インフラ作成コードの実装



(VPC を作成する TypeScript でのコード記述例)

```
const vpc = new Vpc(this, 'cdk-vpc',  
{  
  cidr: '10.0.0.0/23',  
  maxAzs: 2,  
  subnetConfiguration: [{  
    name: 'public-subnet-',  
    cidrMask: 24,  
    subnetType: SubnetType.PUBLIC  
  }]  
});
```

- 10.0.0.0/23 のCIDR レンジ
(512 個のIP アドレス)で

5. インフラ作成コードの実装



(VPC を作成する TypeScript でのコード記述例)

```
const vpc = new Vpc(this, 'cdk-vpc',  
{  
  cidr: '10.0.0.0/23',  
  maxAzs: 2,  
  subnetConfiguration: [{  
    name: 'public-subnet-',  
    cidrMask: 24,  
    subnetType: SubnetType.PUBLIC  
  }]  
});
```

- 10.0.0.0/23 のCIDR レンジ (512 個のIP アドレス) で
- 2 つのアベイラビリティゾーン に対して

5. インフラ作成コードの実装



(VPC を作成する TypeScript でのコード記述例)

```
const vpc = new Vpc(this, 'cdk-vpc',  
{  
  cidr: '10.0.0.0/23',  
  maxAzs: 2,  
  subnetConfiguration: [{  
    name: 'public-subnet-',  
    cidrMask: 24,  
    subnetType: SubnetType.PUBLIC  
  }]  
});
```

- 10.0.0.0/23 のCIDR レンジ (512 個のIP アドレス) で
- 2 つのアベイラビリティゾーン に対して
- CIDR レンジ /24 (256 個の IPアドレス) のパブリック サブネット

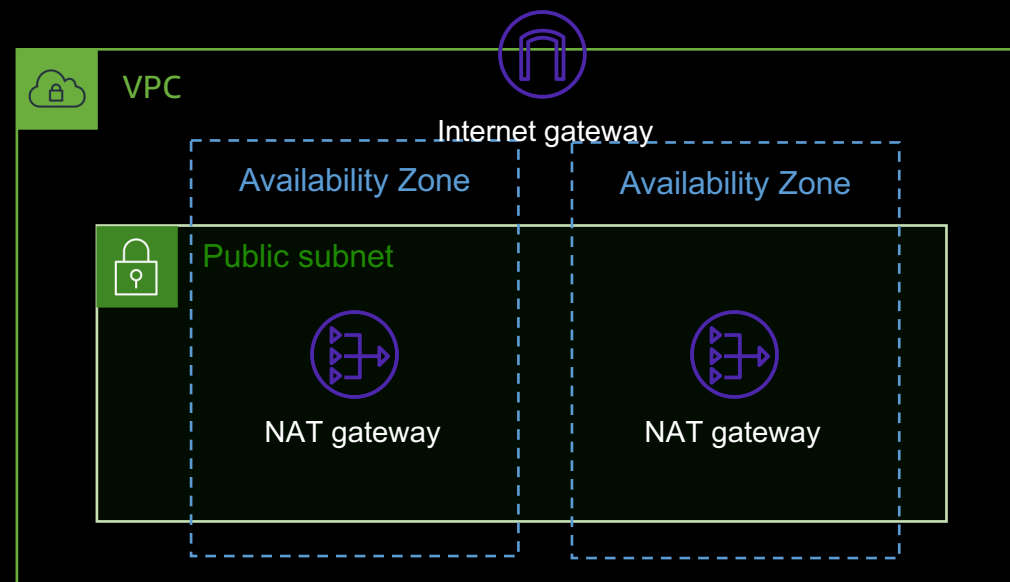
を構築

5. インフラ作成コードの実装



(VPC を作成する TypeScript でのコード記述例)

```
const vpc = new Vpc(this, 'cdk-vpc',  
{  
  cidr: '10.0.0.0/23',  
  maxAzs: 2,  
  subnetConfiguration: [{  
    name: 'public-subnet-',  
    cidrMask: 24,  
    subnetType: SubnetType.PUBLIC  
  }]  
});
```



わずかなコードで、必要なリソースの作成、関連付けが可能

5. インフラ作成コードの実装



(VPC を作成する TypeScript)

```
const vpc = new Vpc(this, 'cdk-vpc',  
{  
  cidr: '10.0.0.0/23',  
  maxAzs: 2,  
  subnetConfiguration: [{  
    name: 'public-subnet-',  
    cidrMask: 24,  
    subnetType: SubnetType.PUBLIC  
  }]  
});
```

わずかなコードで、必要なリソースを生成する

生成される CloudFormation テンプレート

```
{  
  "Resources": {  
    "cdkvpc58C641FA": {  
      "Type": "AWS::EC2::VPC",  
      "Properties": {  
        "CidrBlock": "10.0.0.0/23",  
        "EnableDnsHostnames": true,  
        "EnableDnsSupport": true,  
        "InstanceTenancy": "default",  
        "Tags": [  
          {  
            "Key": "Name",  
            "Value": "CdkStack/cdk-vpc"  
          }  
        ]  
      },  
      "Metadata": {  
        "aws:cdk:path": "CdkStack/cdkvpc/Resource"  
      }  
    },  
    :  
  }  
}
```

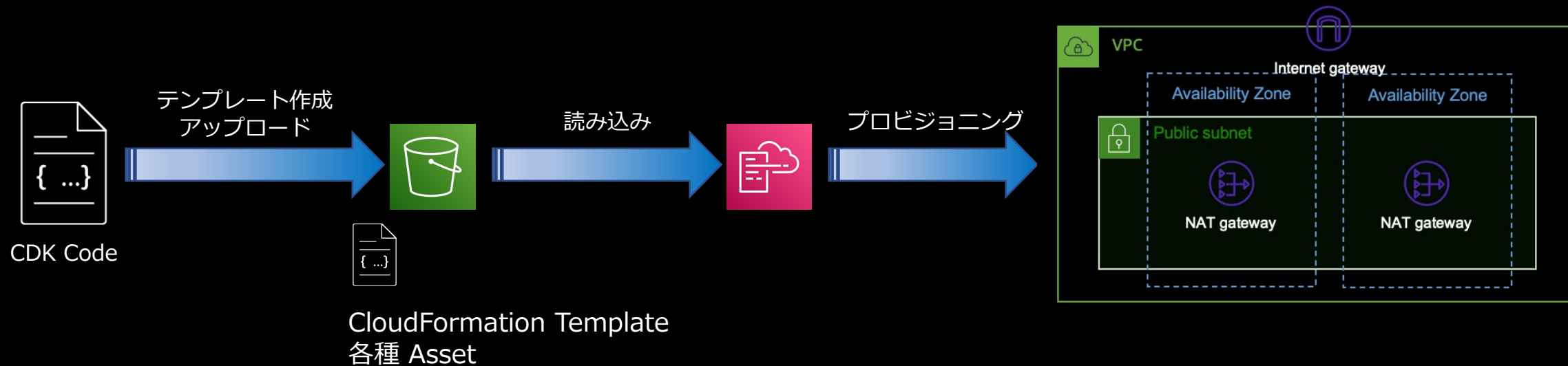
約450 行

6. デプロイ



- コードに従って、AWS 環境上にリソースが構築される

cdk deploy



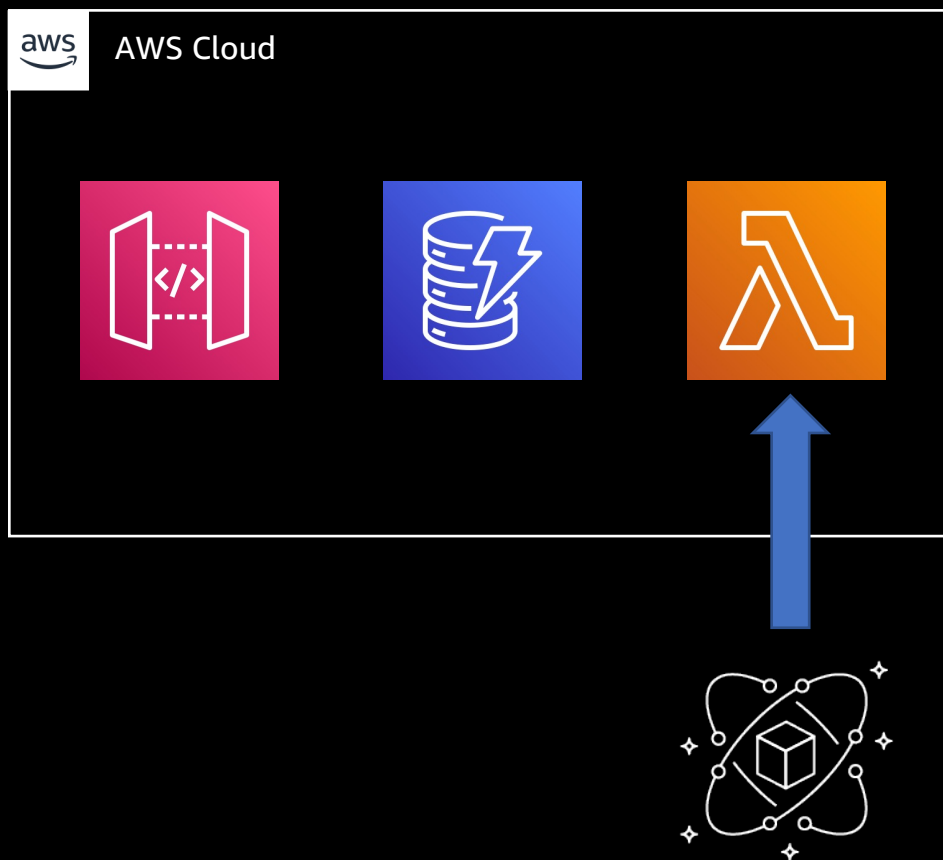
AWS CDK の便利な使い方

AWS CDK の便利な使い方

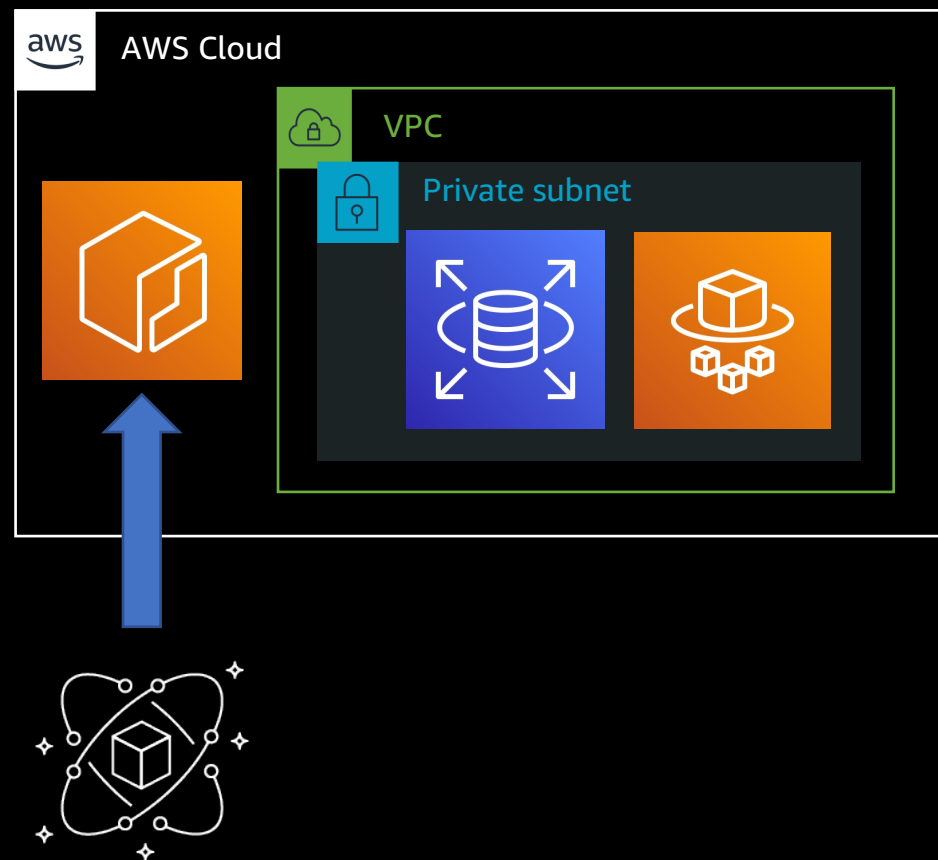
- インフラと同時にアプリケーションもデプロイ
- 類似構成の複数のリソースを短いコードで作成
- 条件によるリソース設定値の制御
- デプロイ前のテスト実行

インフラと同時にアプリケーションもデプロイ

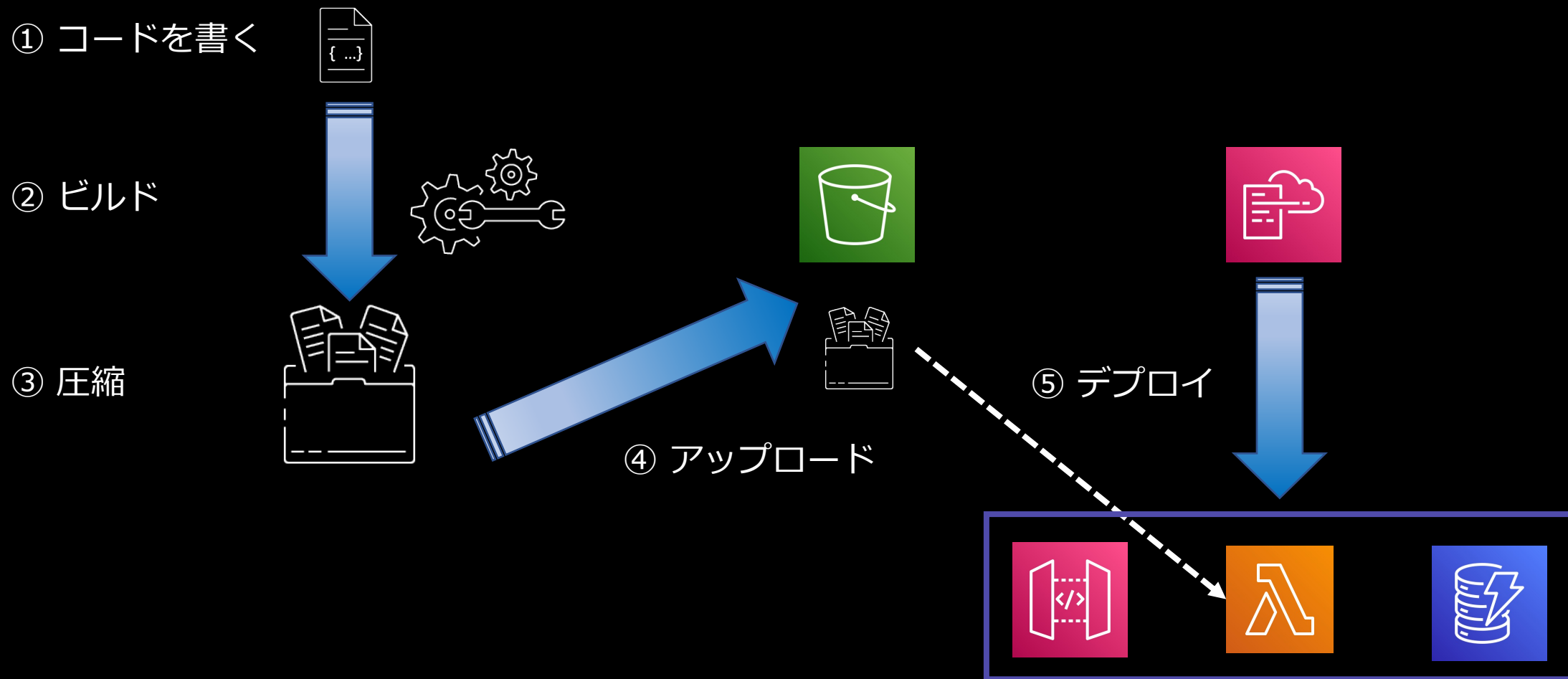
サーバーレスアプリケーション



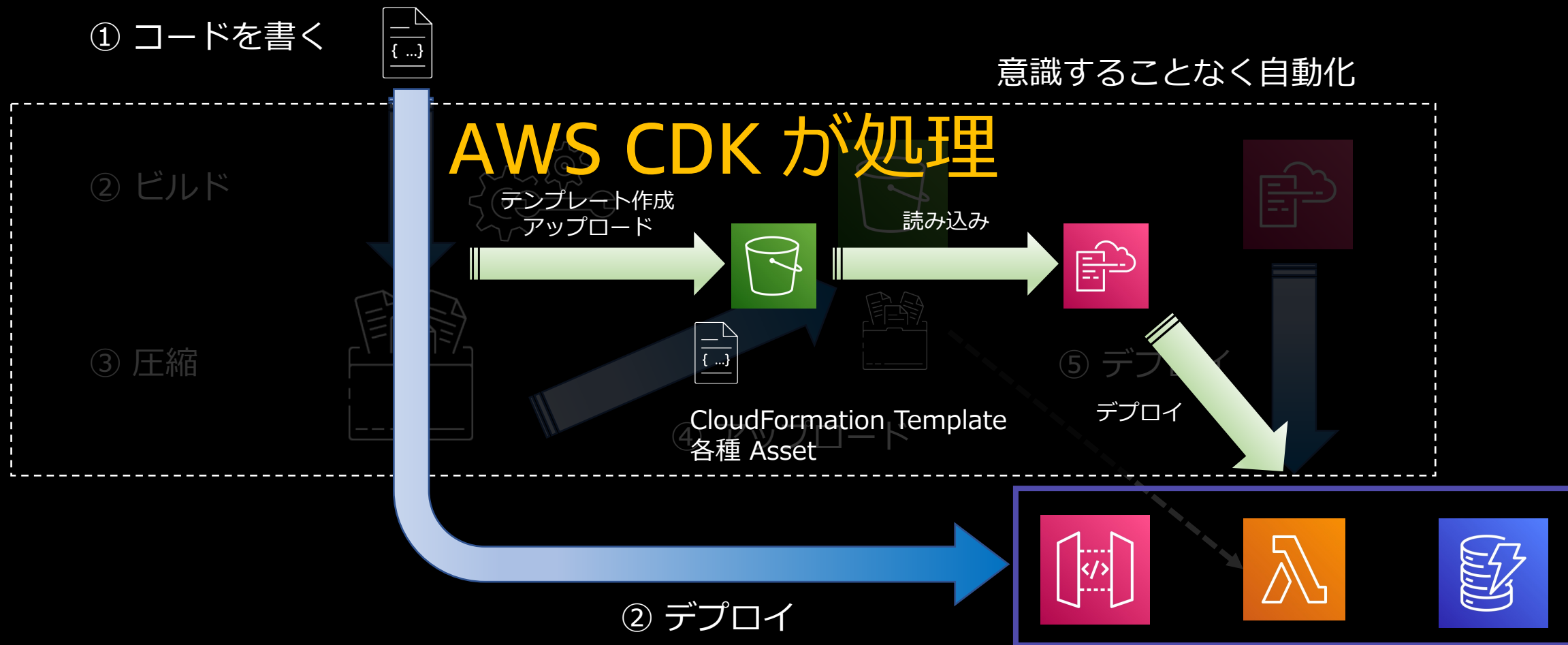
コンテナアプリケーション



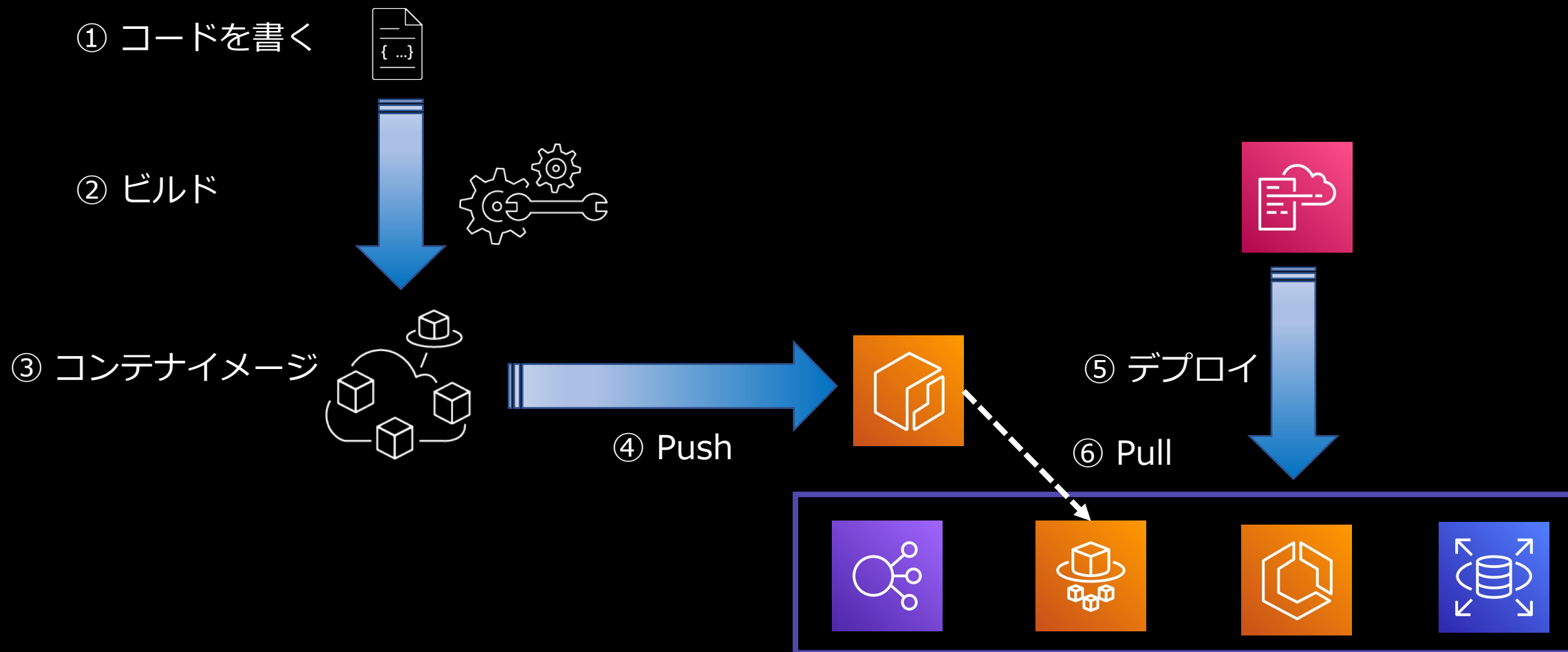
サーバーレスアプリケーションのデプロイ例



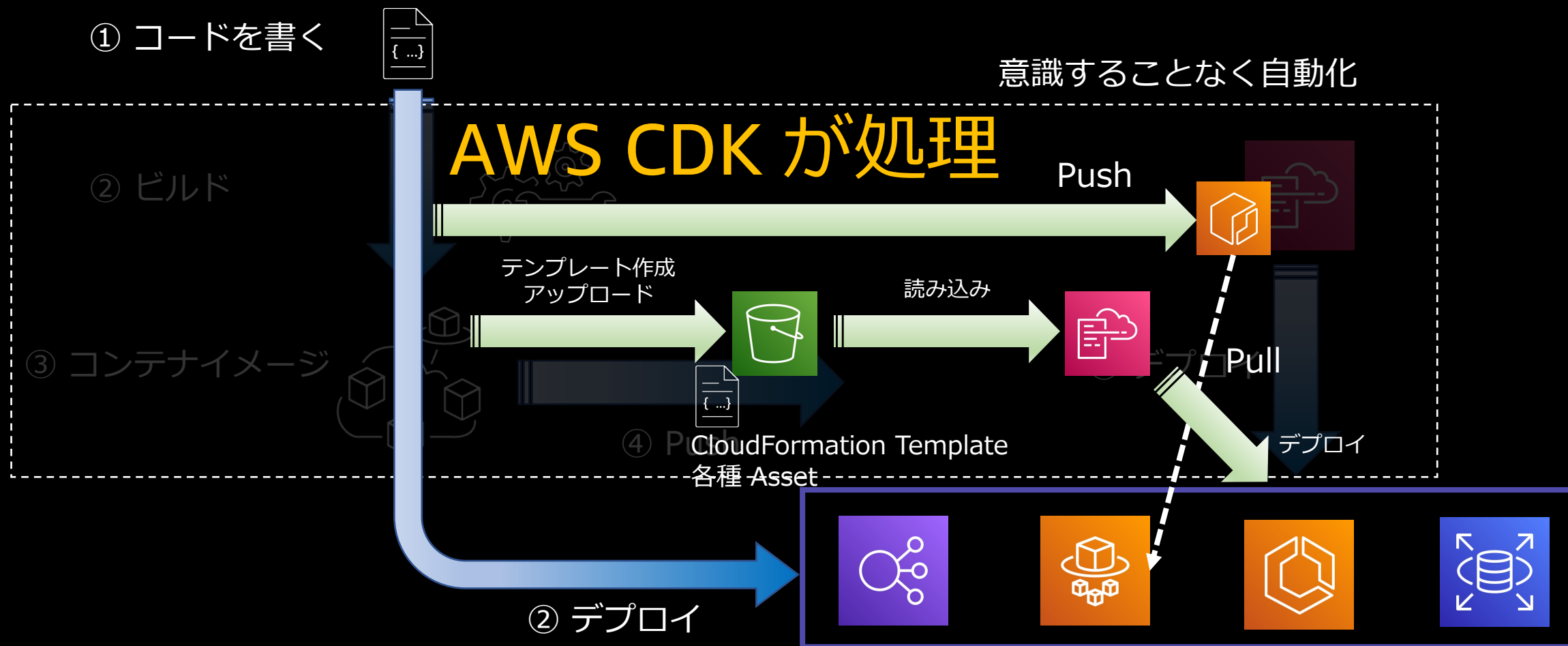
サーバーレスアプリケーションのデプロイ例



コンテナアプリケーションのデプロイ例



コンテナアプリケーションのデプロイ例



類似構成の複数のリソースを短いコードで作成

同じ構成で IP アドレス範囲のみが異なる 3 つの VPC の作成例

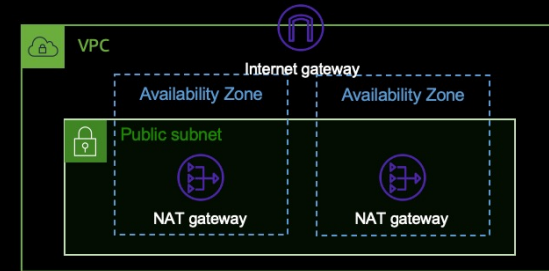
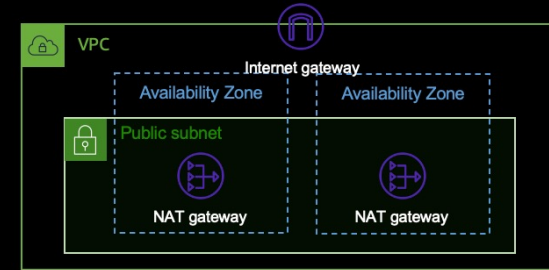
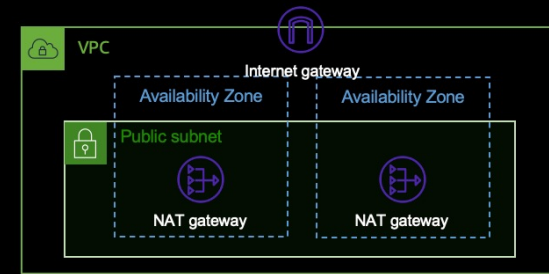
必要な工夫

- 名称や IP アドレス範囲を配列で定義する等して環境ごとに固有の情報を与える

```
for (let i = 1; i <= 3; i++) {
```

```
    // VPC を作成する処理
```

```
}
```



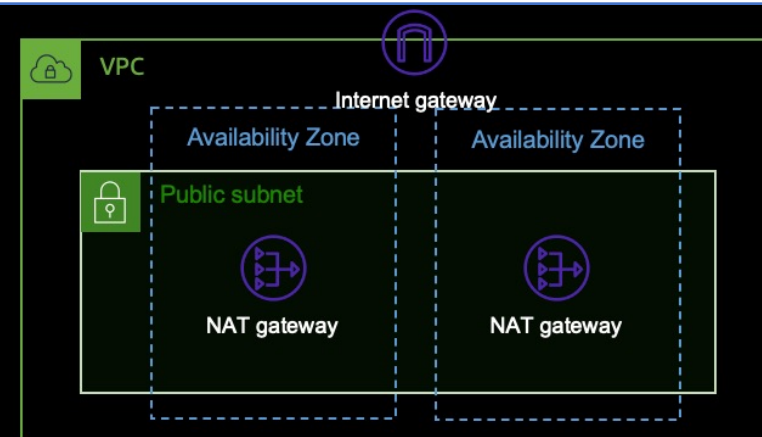
デプロイ前のテスト実行

リソースデプロイ前のテスト実行例



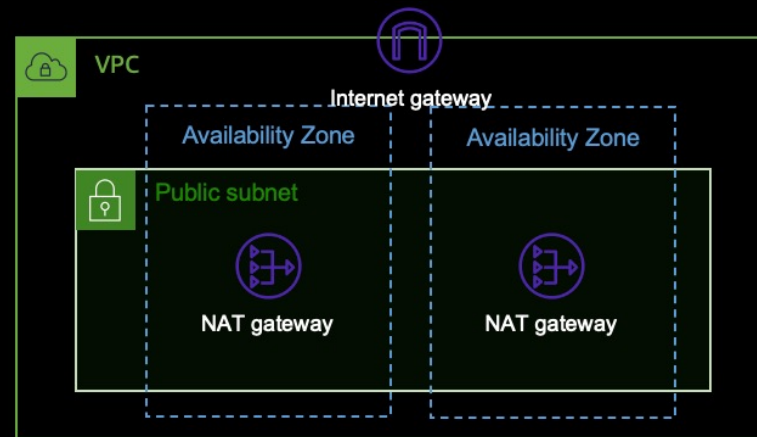
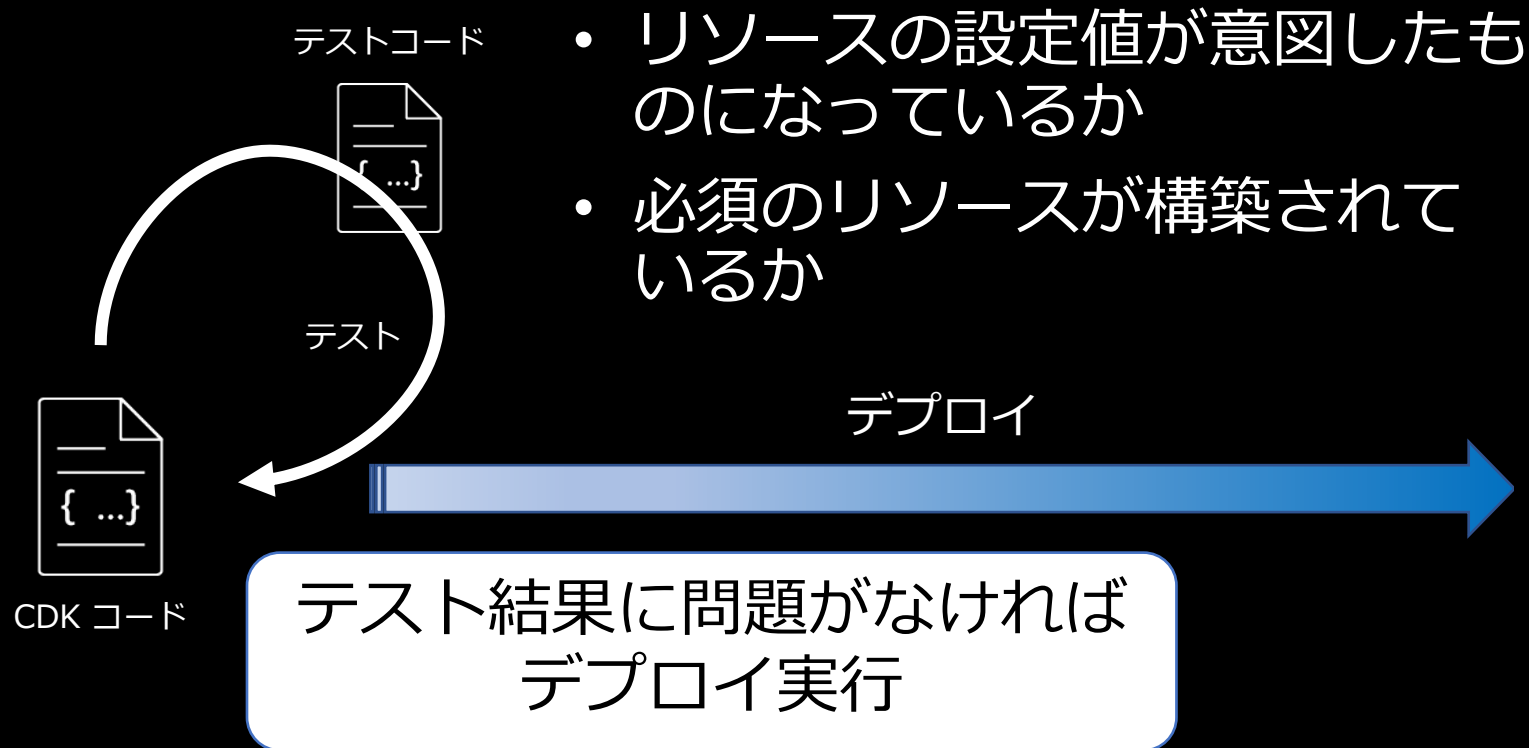
デプロイ

意図した構成になっていない



デプロイ前のテスト実行

リソースデプロイ前のテスト実行例



デプロイ前のテスト実行

AWS CDK で行うことができる主なテストの種類

テスト	確認できること
Snapshot	<ul style="list-style-type: none">• 以前に生成した AWS CloudFormation のテンプレートと、現在のコードから生成されるテンプレートを比較し、同じ内容であるか• コードをリファクタリングする場合に、意図しない変更がされていないかを確認する目的で行うことが多いが、エラーになることを前提に、変更した内容がテンプレートに反映されているかを確認する目的で行うこともある
Fine-grained assertions	<ul style="list-style-type: none">• 生成される AWS CloudFormation のテンプレートに意図したリソースが含まれているか• 生成するリソースの設定値が、意図した値になっているか
Validation	<ul style="list-style-type: none">• 無効なパラメータを外部から与えた際にエラーになるか

まとめ

本セッションのまとめ

- インフラの構築には、Infrastructure as Code を選択肢として検討する
 - ✓ 検証してすぐに破棄してしまうような環境では IaC 化の必要がないケースもあるので、ケースにあわせて検討
 - ✓ IaC の導入を検討する際は、ぜひ AWS CDK を候補に

本セッションのまとめ

- CDK のコードをコードリポジトリで管理する
 - ✓ アプリケーションと同じように変更管理の概念を適用することで、いつ、誰が、どんな修正をおこなったか、を追跡することが容易になる

本セッションのまとめ

- AWS CDK の特徴
 - ✓ 使い慣れたプログラミング言語でリソース定義が可能
 - ✓ わずかなコードで、必要なリソースの作成、関連付けが可能
 - ✓ デプロイ前に、プロビジョニングするリソース構成のテストが可能

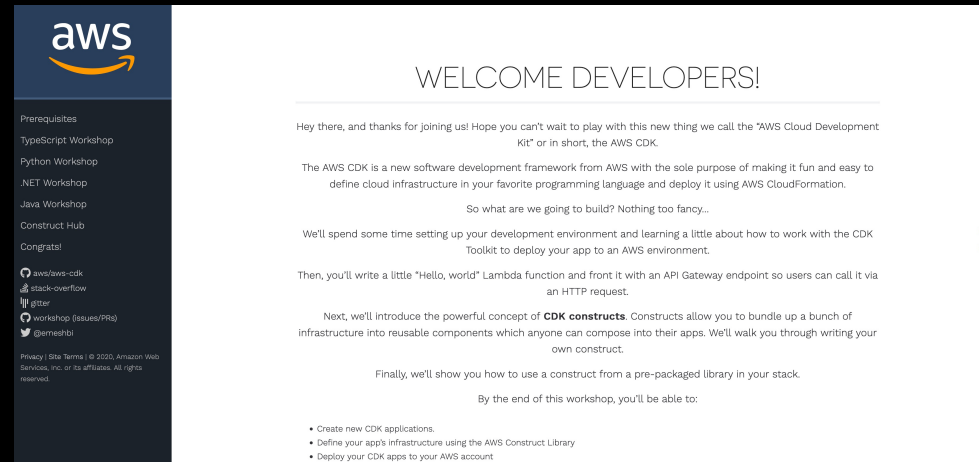
まずは小さな環境からやってみましょう

Next Step

- AWS CDK の無料ワークショップを試してみる

✓ TypeScript / Python / .NET / Java で実施可能

「cdk workshop」で検索



<https://cdkworkshop.com/>

その他関連動画・資料

ご紹介サービスの詳しい解説

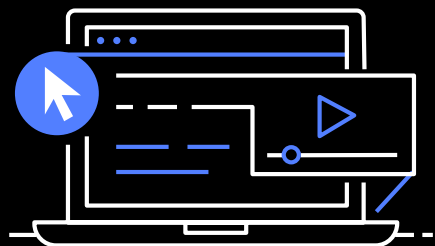
以下ページで「AWS CDK」「AWS CloudFormation」「AWS Command Line Interface」「Amazon S3」「AWS Cloud9」を検索

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

「AWS Serverless Application Model (AWS SAM) とは」

https://docs.aws.amazon.com/ja_jp/serverless-application-model/latest/developerguide/what-is-sam.html

AWS デジタルトレーニング



実力、自信、信頼性を
高め、業界で認められ
た資格で差をつけよう

デジタル学習

- [スキルビルダー](#) - AWS のエキスパートが開発した数百のデジタルトレーニングを自分のスケジュールで学習できます
- [Cloud Quest](#) - AWS Cloud Quest は、実践的なクラウド経験を積み、AWSクラウドのスキルを身につけることができる、初めてで唯一のロールプレイングゲームです

認定試験準備ためのリソース

- [Cloud Practitioner](#) - AWS Certified Cloud Practitioner 取得に役立つリソースをご紹介します
- [Developer – Associate](#) - AWS Certified Developer – Associate 取得に役立つリソースをご紹介します

AWS Builders Online Series にご参加いただきありがとうございます

楽しんでいただけましたか? ぜひアンケートにご協力ください。
本日のイベントに関するご意見/ご感想や今後のイベントについてのご希望や改善のご提案などがございましたら、ぜひお聞かせください。



aws-apj-marketing@amazon.com



twitter.com/awscloud_jp



facebook.com/600986860012140



<https://www.youtube.com/user/AmazonWebServicesJP>



<https://www.linkedin.com/showcase/aws-careers/>



twitch.tv/aws

Thank you!

水流 洋人

アマゾン ウェブサービス ジャパン合同会社
プロフェッショナルサービス本部
クラウド アプリケーション アーキテクト

