

## コードを使った AWS 運用のはじめかた

木村 友則

アマゾン ウェブ サービス ジャパン合同会社  
技術統括本部 テクニカルソリューション部  
ソリューションアーキテクト



## 木村 友則 (きむら とものり)

ソリューションアーキテクト

- 業種・業態を問わず、様々なお客様の構成検討を支援
- 前職では、AWS のユーザーの立場として、インフラエンジニアに従事

好きな AWS サービス

- AWS CLI
- AWS CDK

# 本セッションの対象となる方

- AWS を利用して間もない、又はまだ使っていない
- 手作業での運用に課題感を持っている
- 以上に該当する、インフラ担当者・アプリ担当者

# アジェンダ

- コードを使った運用とは？
- ユースケース別 AWS サービス・ツールを使った運用
- 各サービス・ツールの Tips
- コスト
- まとめ

# コードを使った運用とは？

# AWS における運用にはなにがあるか



## マネジメントコンソール



開発・運用チーム



ガバナンス  
の確立



コンプライアンス  
の実現



監視 & 観測

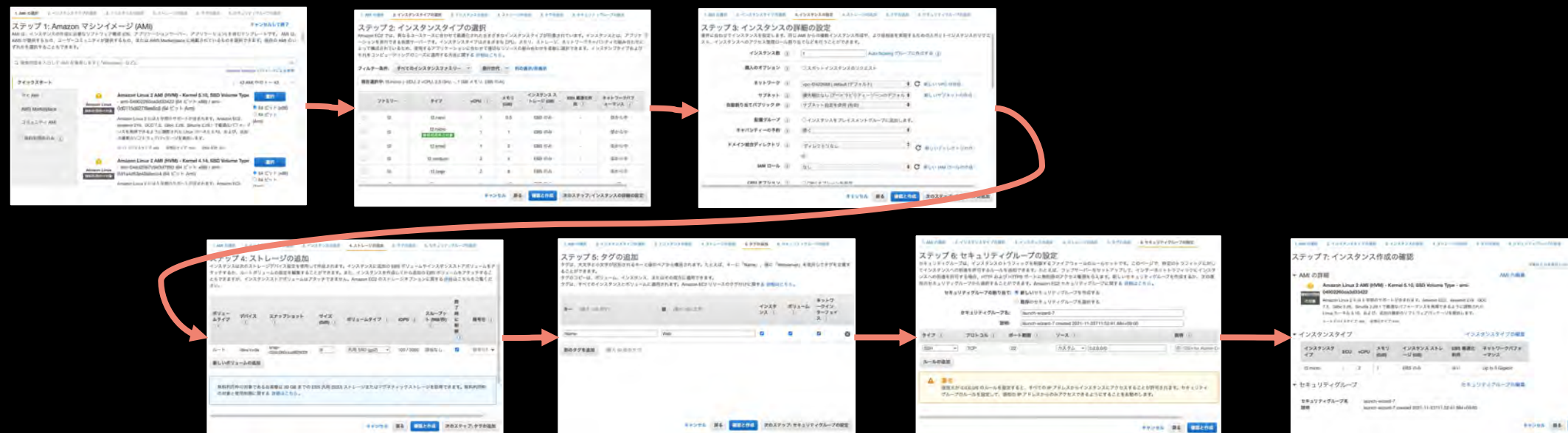


オペレーション



プロビジョニング  
と統合

# 例えば Amazon EC2 を構築する場合



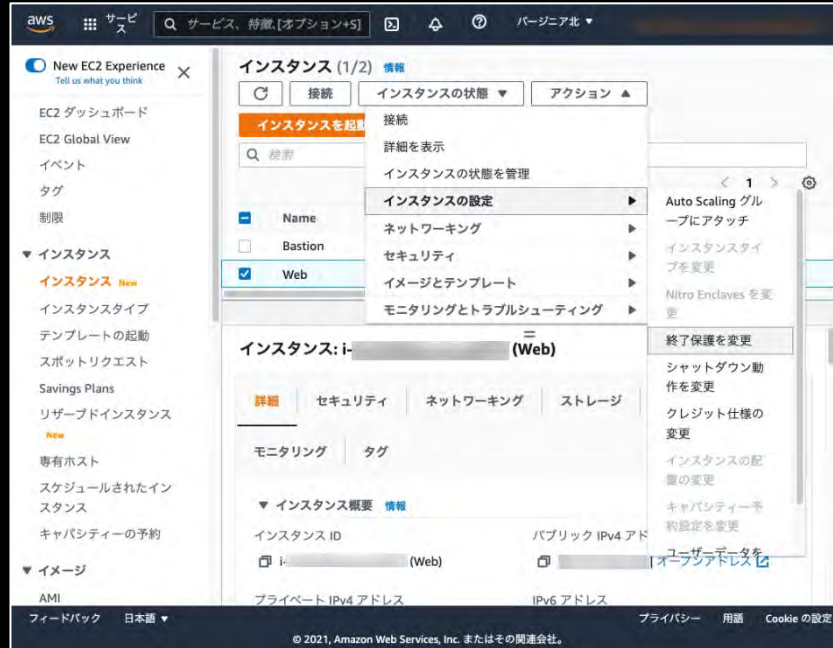
7つのステップ、数分で終わる操作。  
これを10台、100台と繰り返すとしたら...?

開発環境を基に、検証や本番環境の複製をするケースなど



# 例えば Amazon EC2 の設定を変更する場合

#AWSBuilders



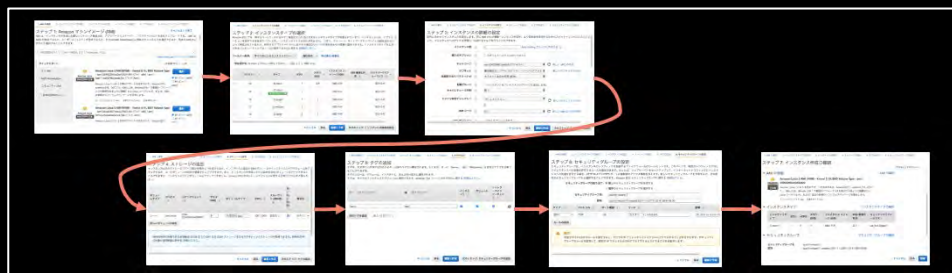
1つのステップで終わる操作。  
これを10台、100台繰り返すとしたら...？

手作業で設定していると、細かい項目の徹底を忘れがち



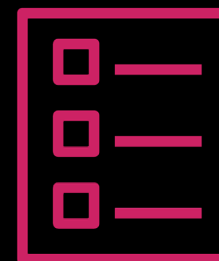
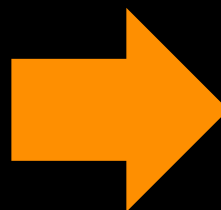
# マネジメントコンソールからでも可能だが...

反復操作は、スクリプトやテンプレートなどの  
コードを使うことで効率的な運用が可能になる



Amazon EC2 の構築操作 × 台数分

マネジメントコンソール

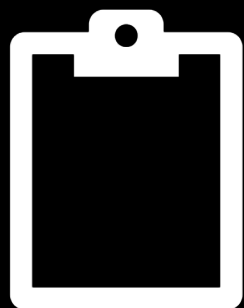


コード作成 → 実行 × 1回

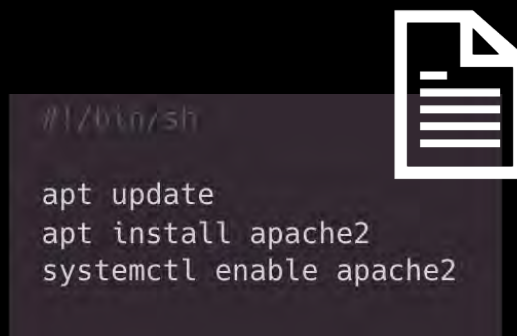
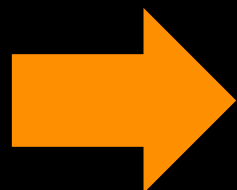
スクリプトやテンプレート

# コードを使った運用とは - 1. 手続きを列挙

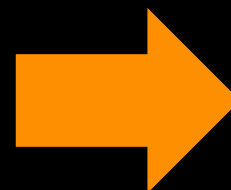
- 反復操作の手続きを列挙してスクリプトにする
  - 手順に従って順に処理を行うため、作業手順書などから移行しやすい
  - 日々繰り返す操作や1回のみのイベントなどに適する
  - 命令的・手続き的なアプローチ
  - 例：〇〇をインストールして、サービスとして有効化する



手順書



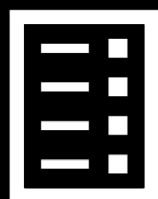
スクリプト



変更の適用

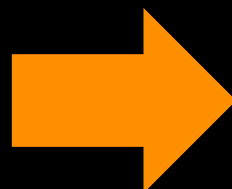
# コードを使った運用とは - 2. 状態を宣言

- 定義ファイル・テンプレートで状態を宣言する
  - どうあるべきかを定めることに注力することができ、手段や実行順序はツールが適切に選択する
  - 環境構築などの継続的な維持が必要で複雑な操作に適する
  - 宣言的なアプローチ
  - 例：〇〇 がインストールされていてサービスとして動いている



```
---  
- name: Install Apache  
  ansible.builtin.apt:  
    name: apache2  
    update_cache: yes  
    state: latest  
  
- name: Enable service httpd  
  ansible.builtin.service:  
    name: apache2  
    enabled: yes
```

テンプレート



変更の適用

# なぜコードを使って運用を行うのか

コード化する手間をかけても得られるメリットが大きい

- だれもが簡単に何度でも再現できる
- 再現可能であるため使い捨てが可能になる
- 変更を安全に適用できる

同じ環境を、迅速に、繰り返し作成可能になり、  
フィードバックを素早く繰り返せる、変化に強い仕組みになる

# 効率的な運用を行うために何ができるか

## ■ 手続きを列挙



### AWS Command Line Interface (CLI)

コマンドで AWS リソースを操作

## ■ 状態を宣言



### AWS CloudFormation

AWS 環境をテンプレートで管理



### AWS Cloud Development Kit (CDK)

AWS 環境を一般のプログラミング言語で記述可能なツールキット  
CloudFormation テンプレートを生成し、CloudFormation で構築実行

## ■ シェル環境

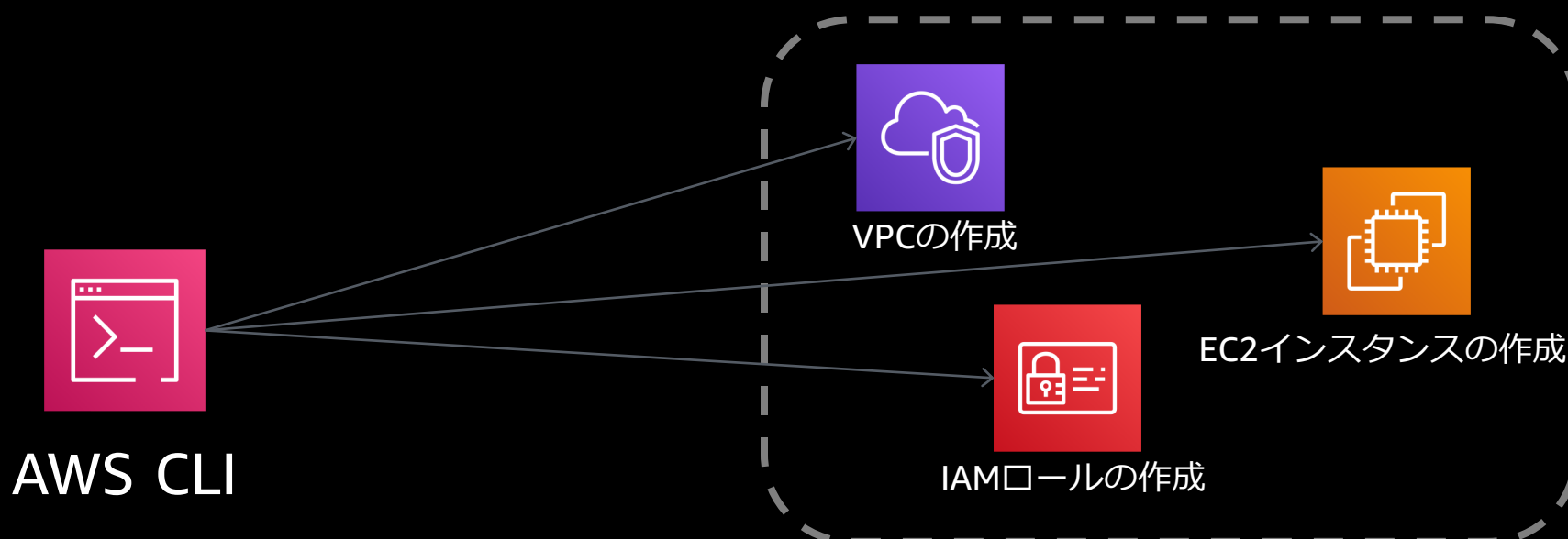


### AWS CloudShell

マネジメントコンソールで  
AWS CLI を利用可能

# AWS CLI

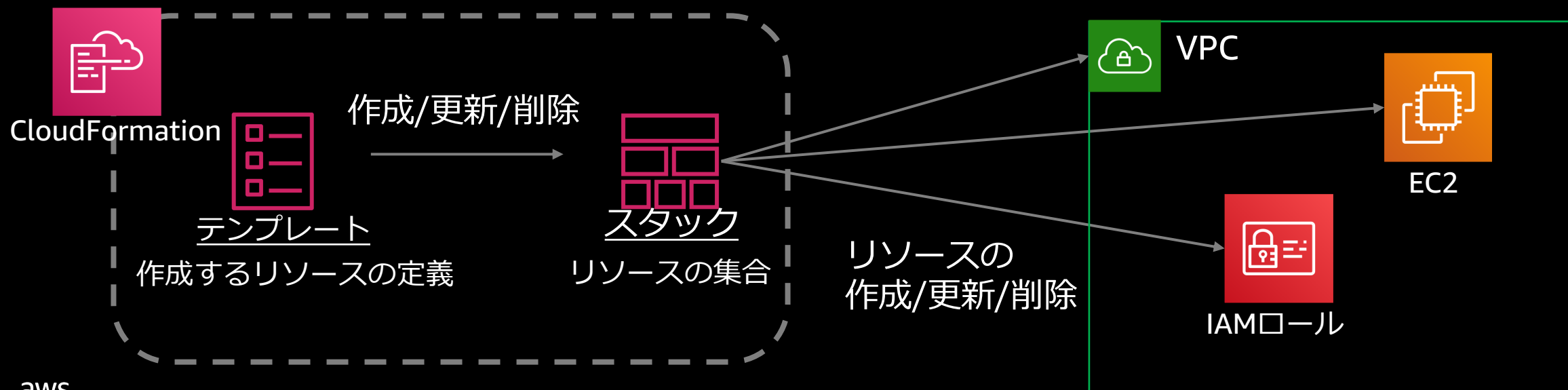
- AWS の各種サービスをコマンドで操作できる OSS の統合ツール
- AWS の API 実装に追従しており、ほぼ全ての AWS サービス・API を操作可能 ※1
- Windows 、 Linux 、 macOS プラットフォームに対応



※1 新サービス・機能进行操作するには、AWS CLI の更新が必要です。定期的な更新を推奨します。

# AWS CloudFormation

- AWS 環境のテンプレート (YAML または JSON) による管理を実現する
- テンプレートで定義した環境を作成/更新/削除できる
- AWS CLI でも作成/更新/削除は可能だがアプローチが異なる
- 複数リソースの依存関係、構築順序は CloudFormation が解決する



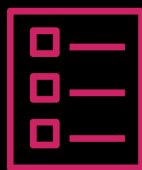
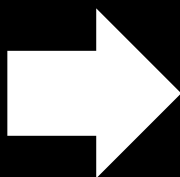


# AWS CDK

- AWS 環境を一般のプログラミング言語で記述できるツールキット
- 記述したコードから CloudFormation テンプレート を生成し構築を実行する
- 一般のプログラミング言語であるため、テキストエディタによる強力な記述サポートの恩恵を受けられる

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
    }, dom
  );
  // Setup AutoScaling policy
  const scaling = service.service.autoScaleTask
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    );
}
```



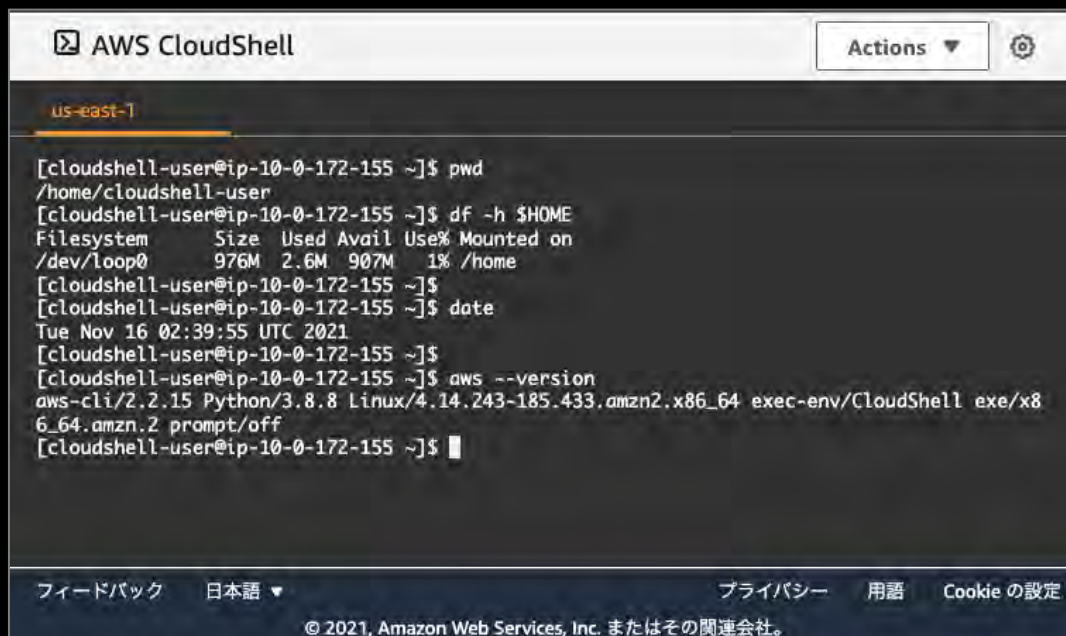
CloudFormation テンプレートを生成し、  
CloudFormation によって構築を実行する

AWS のベストプラクティスをデフォルト値として内包するため  
少ないコード量で表現することが可能

生成される CloudFormation テンプレートによって  
実際にデプロイされるものを確認可能

# AWS CloudShell

- マネジメントコンソール上で利用可能なシェル環境
- AWS CLI、Python、Node.js 等がインストール済み
- ホームディレクトリ※1に保存されたファイルは、同リージョンの将来のセッションで再利用可能※2

The screenshot shows the AWS CloudShell interface. At the top, there's a header bar with the AWS CloudShell logo, an 'Actions' dropdown menu, and a settings icon. Below the header, the terminal window shows the following content:  
- A tab labeled 'us-east-1'.  
- The terminal prompt is '[cloudshell-user@ip-10-0-172-155 ~]\$'.  
- The first command is 'pwd', which outputs '/home/cloudshell-user'.  
- The second command is 'df -h \$HOME', which outputs a table of disk usage for the /home directory.  
- The third command is 'date', which outputs 'Tue Nov 16 02:39:55 UTC 2021'.  
- The fourth command is 'aws --version', which outputs 'aws-cli/2.2.15 Python/3.8.8 Linux/4.14.243-185.433.amzn2.x86\_64 exec-env/CloudShell exe/x86\_64.amzn.2 prompt/off'.  
- The terminal ends with the prompt '[cloudshell-user@ip-10-0-172-155 ~]\$' and a cursor.  
At the bottom of the interface, there's a footer bar with links for 'フィードバック' (Feedback), '日本語' (Japanese), 'プライバシー' (Privacy), '用語' (Glossary), and 'Cookie の設定' (Cookie Settings). Below these links is the copyright notice '© 2021, Amazon Web Services, Inc. またはその関連会社。'.

※1 最大1GB

※2 最終セッションから120日まで

# ユースケース別 **AWS** サービス・ツールを使った運用

# ユースケース

1. 手元 PC やオンプレミスから AWS リソースを操作したい
2. 環境設定なしで AWS CLI を実行したい
3. AWS リソースをテンプレートで定義する
4. AWS リソースをプログラミング言語で定義する

# 1. 手元 PC やオンプレミスから **AWS** リソースを操作したい

例えば...

- EC2 インスタンスの属性を確認・変更したい
- オンプレミスのサーバーから Amazon S3 にデータをバックアップしたい

このような場合には AWS CLI が最適

EC2 インスタンスの属性を確認・変更する

```
$ EC2ID=${インスタンスID}  
$ aws ec2 describe-instance-attribute --attribute disableApiTermination --instance-id ${EC2ID}  
$ aws ec2 modify-instance-attribute --disable-api-termination --instance-id ${EC2ID}
```

S3 ヘデータをバックアップする

```
$ aws s3 sync ./original-data s3://${S3バケット名}/
```

## 2. 環境設定なしで **AWS CLI** を実行したい

例えば...

- AWS CLI を使いたいが、環境設定が面倒
- PCに、IAM (Identify and Access Management) ユーザーのアクセスキーを置きたくない

このような場合には AWS CloudShell が最適



```

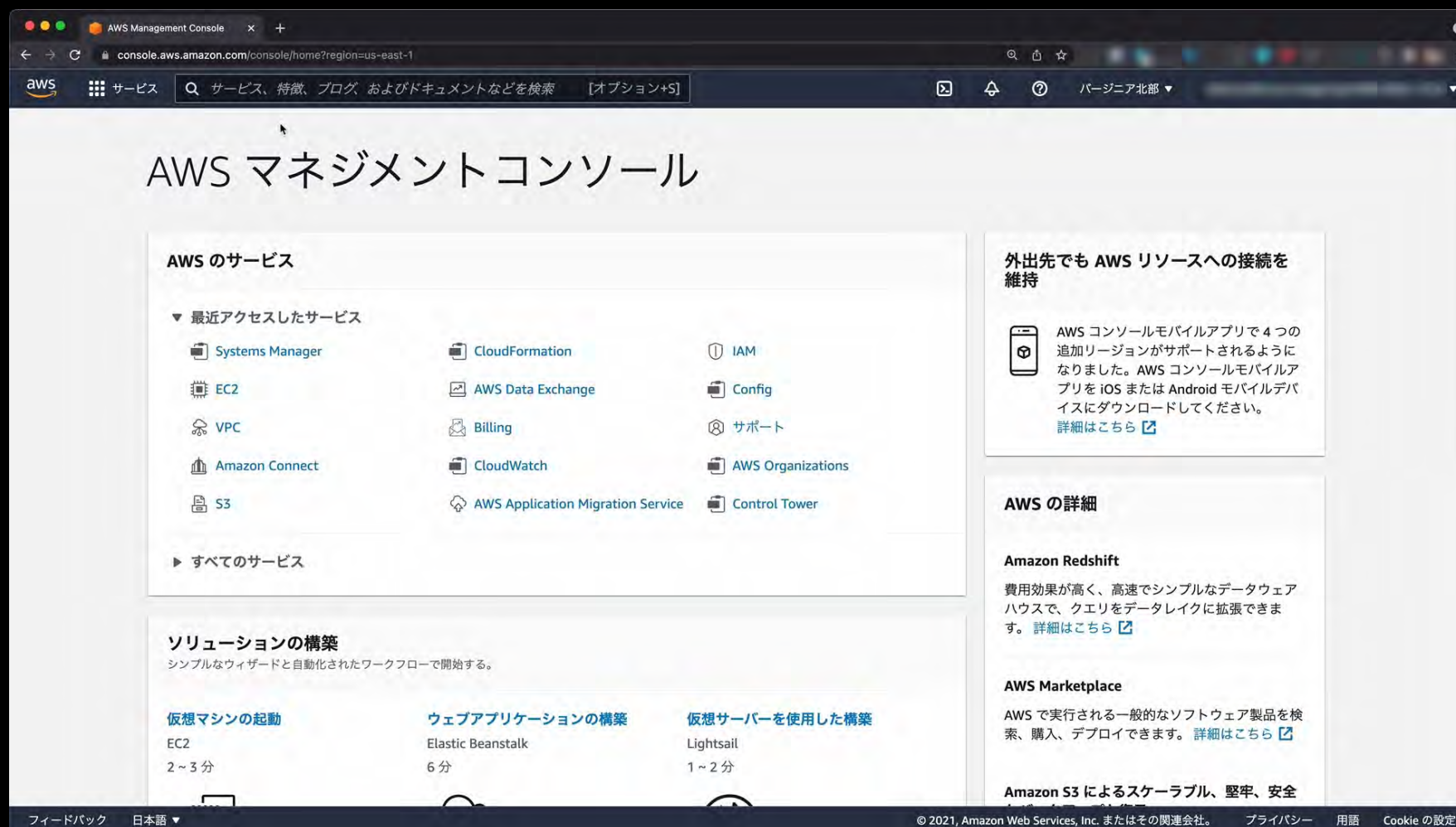
AWS CloudShell
us-east-1
[cloudshell-user@ip-10-0-172-155 ~]$ pwd
/home/cloudshell-user
[cloudshell-user@ip-10-0-172-155 ~]$ df -h $HOME
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop0      976M  2.6M  907M   1% /home
[cloudshell-user@ip-10-0-172-155 ~]$ date
Tue Nov 16 02:39:55 UTC 2021
[cloudshell-user@ip-10-0-172-155 ~]$ aws --version
aws-cli/2.2.15 Python/3.8.8 Linux/4.14.243-185.433.amzn2.x86_64 exec-env/CloudShell exe/x86_64.amzn.2 prompt/off
[cloudshell-user@ip-10-0-172-155 ~]$

```

- AWS CLI がインストール済み
- アクセスキー不要、ログイン中のIAM ユーザーの権限(ロール)で実行可能
- どこからでも同じ環境が使える



# デモ: CloudShell で CLI を実行してみる



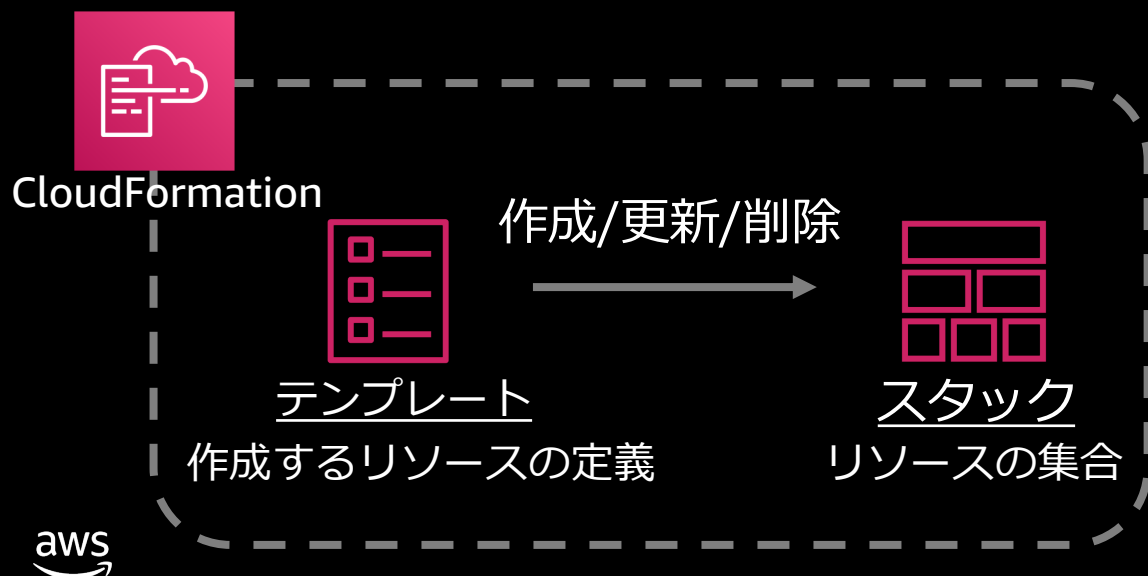


### 3. AWS リソースをテンプレートで定義する

例えば...

- 複数のリソースをまとめて構築したい
- 開発環境を作り、その後に検証・本番環境を複製したい

このような場合には AWS CloudFormation が最適



- リソース同士の依存関係は、CloudFormation が解決
- 単一テンプレートを利用し、パラメータによって構築する内容を調整出来る

## 4. **AWS** リソースをプログラミング言語で定義する

例えば...

- YAML や JSON ではなく、エディタによる記述サポートがある一般的なプログラミング言語で記述したい
- 少ないコード量で表現をしたい

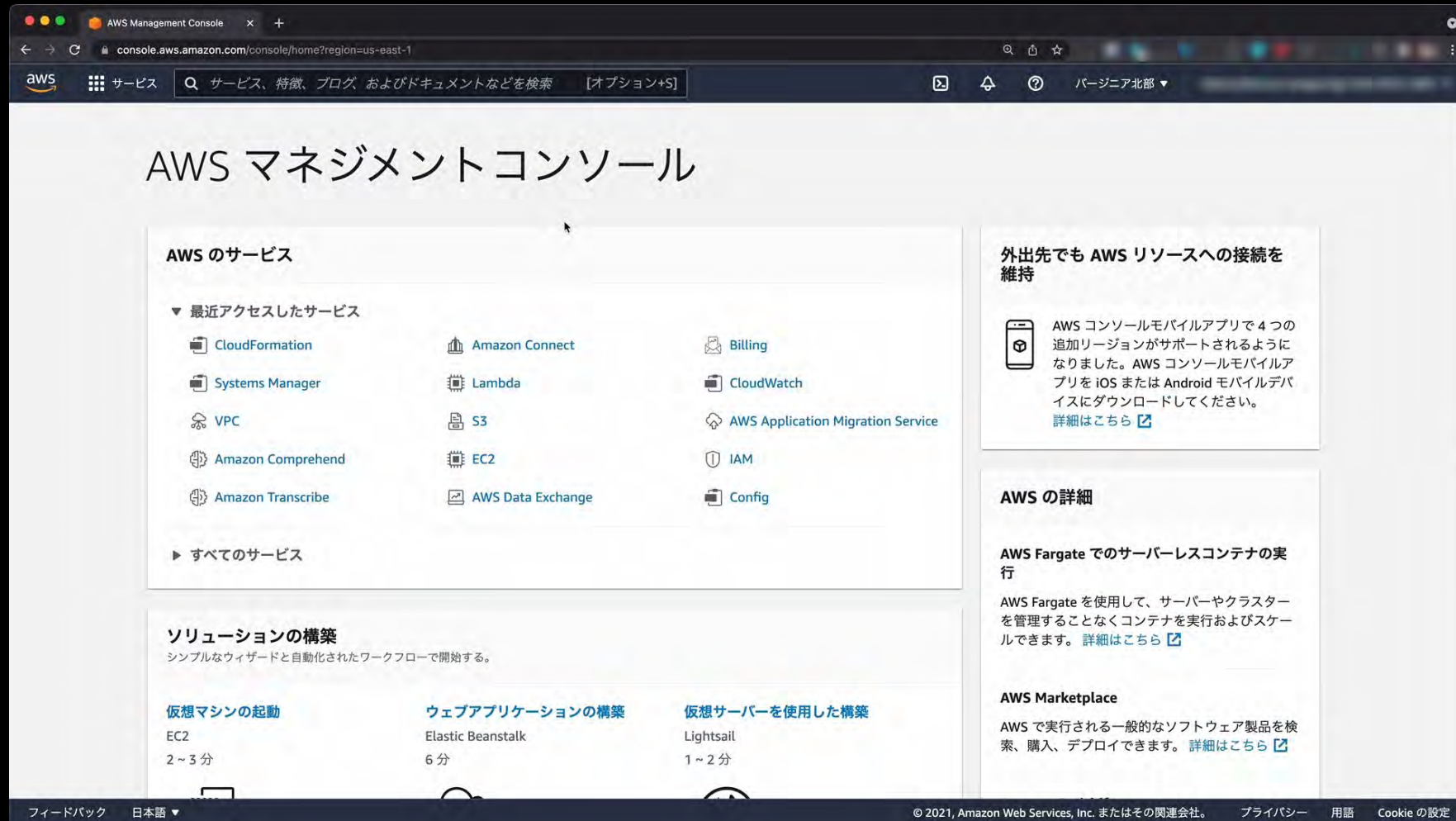
このような場合には AWS CDK が最適

```
const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
  cluster,
  taskImageOptions: {
    image: ecs.ContainerImage.fromAsset('ping'),
  },
  domain
});
// Setup AutoScaling policy
const scaling = service.service.autoScalingAspects().add(new ScalingPolicy({
  scaling.scaleOnCpuUtilization('CpuScaling', {
    targetUtilizationPercent: 50,
    scaleInCooldown: Duration.seconds(60),
    scaleOutCooldown: Duration.seconds(60)
  });
}));
```



※ Go は 2021年11月現在 Developer Preview です

# デモ: CloudFormation で VPC を作成してみる #AWSBuilders



# 各サービス・ツールの Tips

# CLI と CloudFormation/CDK の使い分け

- リソースの構築は CLI でも可能だが、複数リソースの作成では CloudFormation/CDK が最適
- 依存関係や構築順序を CloudFormation が自動的に解決する
- CloudFormation/CDK は何度適用しても同じ結果が得られる

## CloudFormation サンプル

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  CFnVPC:
    Type: AWS :: EC2 :: VPC
    Properties:
      CidrBlock: 10.0.0.0/16
```

## CLI サンプル

```
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

一見 CLIの方が単純だが、何度実行しても同じ結果が得られることの保証や、複数リソースの依存関係を考慮した実行順序の調整は、自身で解決する必要がある

# CloudFormation/CDK との付き合い方

- VPC や IAM などの**変更頻度が低いもの**からの着手がおすすめ
  - 部分的なコード化からでも価値は大きい
- **100% を目指さない**
  - コード化できないもの・しないものがあっても良い
  - 環境操作の手続きを定義する場合には CLI・シェルスクリプトを使う
- **手作業が残っても良い**
  - 反復しない操作ならば、コードにしない選択も良い
  - 例えば、再現不要、期間限定のリソースなど



# 状態を宣言することの大きなメリット

```
AWSTemplateFormatVersion: 2010-09-09

Resources:
  CFnVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      InstanceTenancy: default
      EnableDnsSupport: true
      EnableDnsHostnames: true

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.0/24
      VpcId: !Ref CFnVPC
      AvailabilityZone: !Select [ 0, !GetAZs ]
      MapPublicIpOnLaunch: true

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.2.0/24
      VpcId: !Ref CFnVPC
      AvailabilityZone: !Select [ 0, !GetAZs ]
```

VPC を定義する CloudFormation テンプレート



- **宣言と実体(AWSリソース)が一致している**
  - 実体と差分が出れば検知することができる
  - 手順書で構築した環境は、一致を維持することが難しい
- **差分が分かるので復元も容易**
  - 再適用すれば復元できる
  - テンプレートを実体に合わせることも出来る
- **記述内容はシンプルでわかりやすい**
  - マネジメントコンソールでの操作の代わりに、リソースのプロパティに値を指定するだけ
  - **手間はかかるがメリットは大きい**



# コスト



# コスト

- AWS CLI、AWS CloudFormation、AWS CDK、AWS CloudShell は、それ自体の**コストは基本的に無料**
- 付随して利用するサービスに基づく料金
  - AWS CLI、AWS CloudFormation、AWS CDK で作成した各種リソース
  - AWS CloudShell の利用で発生するデータ転送量 ※1
    - インターネットからのデータ転送(イン) 0 USD/GB
    - インターネットへのデータ転送(アウト) 0.114USD/GB ※2

※1 金額は2021年11月時点の東京リージョンのものです

※2 2021年11月24日に発表されたデータ転送量の値下げにより、100GB/月までは無料で利用できます

# まとめ

# ご紹介した主なサービス/ツール

## ■ 手続きを列挙



### AWS Command Line Interface (CLI)

コマンドで AWS リソースを操作

## ■ 状態を宣言



### AWS CloudFormation

AWS 環境をテンプレートで管理



### AWS Cloud Development Kit (CDK)

AWS 環境を一般のプログラミング  
言語で記述可能なツールキット  
CloudFormation テンプレートを生成  
し、CloudFormation で構築実行

## ■ シェル環境



### AWS CloudShell

マネジメントコンソールで  
AWS CLI を利用可能

# まずはここから始めましょう

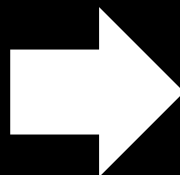
CloudShell で CLI を実行してみませんか？



```
us-east-1

[cloudshell-user@ip-10-0-172-155 ~]$ pwd
/home/cloudshell-user
[cloudshell-user@ip-10-0-172-155 ~]$ df -h $HOME
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop0      976M  2.6M  907M   1% /home
[cloudshell-user@ip-10-0-172-155 ~]$
[cloudshell-user@ip-10-0-172-155 ~]$ date
Tue Nov 16 02:39:55 UTC 2021
[cloudshell-user@ip-10-0-172-155 ~]$
[cloudshell-user@ip-10-0-172-155 ~]$ aws --version
aws-cli/2.2.15 Python/3.8.8 Linux/4.14.243-185.433.amzn2.x86_64 exec-env/CloudShell exe/x86_64.amzn.2 prompt/off
[cloudshell-user@ip-10-0-172-155 ~]$
```

CloudShell を開く



S3 バケットを作ってみる

```
$ aws s3 mb s3://${バケット名}
```

S3 バケットのリストを取得してみる

```
$ aws s3 ls
```

作った S3 バケット を削除してみる

```
$ aws s3 rb s3://${バケット名}
```

※ \${バケット名} には、全世界で一意になる名称で指定してください  
例えば「名前-日付」 → 「tkimurz-20220120」 など

※ これらの操作に、AWSの利用料は発生しません

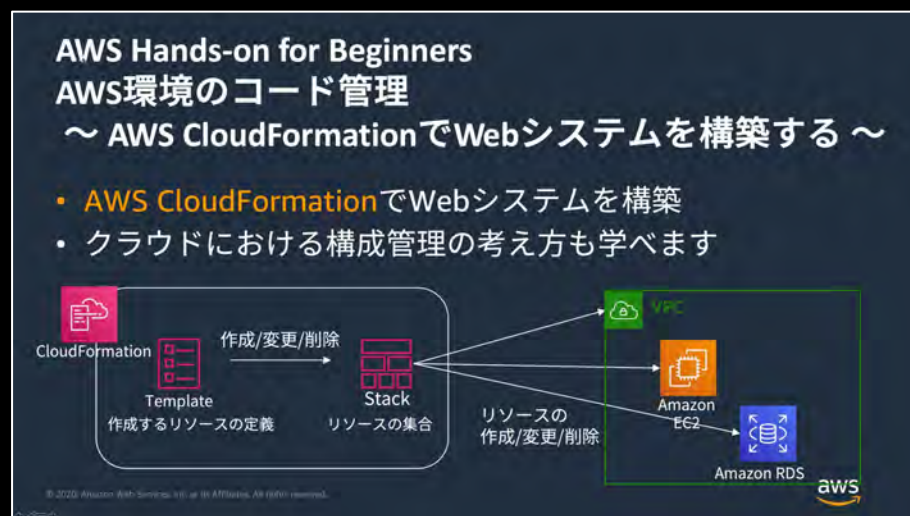
CLIの操作パターン集や詳しい解説はこちら

[https://d1.awsstatic.com/webinars/jp/pdf/services/20190724\\_AWS-BlackBelt\\_AWS%20CLI\\_A.pdf](https://d1.awsstatic.com/webinars/jp/pdf/services/20190724_AWS-BlackBelt_AWS%20CLI_A.pdf)

# Next Step

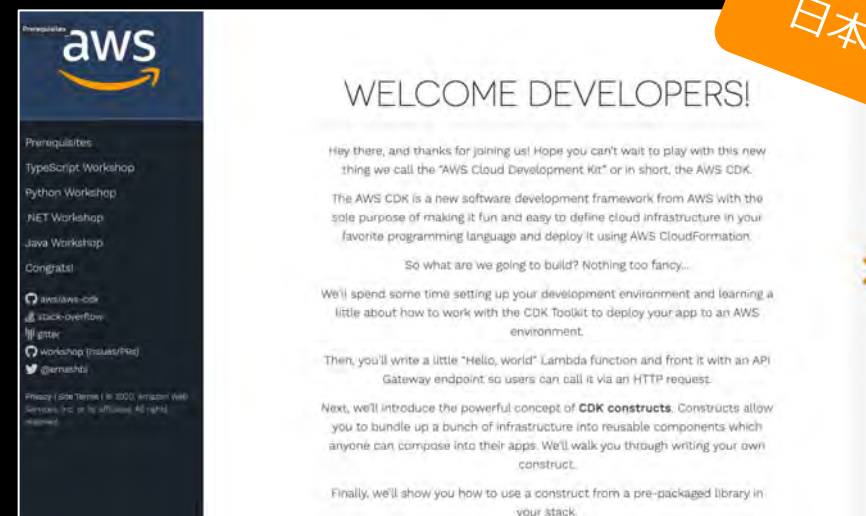
- CloudShell での CLI 利用は、AWS アカウントさえあればすぐに可能
- CloudFormation / CDK の学習用コンテンツがある

「AWS ハンズオン」で検索！



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

「cdk workshop」で検索！



日本語化予定あります

<https://cdkworkshop.com/>

# まとめ

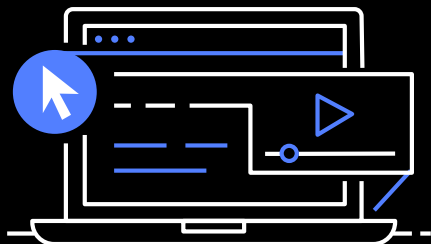
- コードを使うことで効率的な運用ができる
  - スクリプトやテンプレートを使おう
- コード化する手間をかけても得られるメリットが大きい
  - だれもが簡単に何度でも再現できる
  - 宣言(テンプレート)と実体(AWSリソース)が一致している
- すぐにでも始められることがある
  - CloudShell + CLI から始めませんか



# 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「Command Line Interface」「CloudFormation」「CDK」を検索
  - <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
- 「オンプレミスの運用経験をクラウドで活用するには？」
  - <https://resources.awscloud.com/aws-builders-online-series-japanese/bos75-aws-builders-online-series-202107-management>
- 「AWSで始めるInfrastructure as Code」
  - [https://d1.awsstatic.com/events/jp/2021/summit-online/AWS-31\\_AWS\\_Summit\\_Online\\_2021\\_MAD05.pdf](https://d1.awsstatic.com/events/jp/2021/summit-online/AWS-31_AWS_Summit_Online_2021_MAD05.pdf)
  - <https://youtu.be/BoZfgBN6JCs>

# AWS デジタルトレーニング



学習方法を柔軟に選ぶことができます。

550以上の無料デジタルトレーニングでクラウドのスキルを身につけることができ、クラスルームトレーニングでさらに深く学ぶこともできます。

## おすすめのコース

- **[AWS Cloud Practitioner Essentials \(Japanese\)](#)**

このコースは、特定の技術領域ではなく、アマゾン ウェブ サービス (AWS) クラウドを全体的に理解したい方を対象としています。受講者は、AWS クラウドの概念、AWS のサービス、セキュリティ、アーキテクチャ、料金、サポートについて学習し、AWS クラウドについての知識を深めます。このコースは、AWS Certified Cloud Practitioner 試験の準備にも役立ちます。

- **[Getting Started with AWS Security, Identity, and Compliance \(Japanese\)](#)**

このコースでは、AWS のセキュリティテクノロジー、ユースケース、メリット、サービスの概要について説明します。AWS のセキュリティ、アイデンティティ、コンプライアンスのカテゴリにおけるさまざまなサービスを紹介します。このコースを修了すると、クラウド内のセキュリティの重要性を理解し、データの保護に使用する AWS のサービスを選択できるようになります。

- **[Moving to Managed Databases on AWS \(Japanese\)](#)**

リフトアンドシフト戦略は、クラウドを利用しようとする組織が一般的に採用する方法です。このコースでは、AWS ヒーローである Alex DeBrie が、アマゾン ウェブ サービス (AWS) においてセルフマネージドデータベースからフルマネージドデータベースソリューションに移行することのメリットについて説明します。また、AWS クラウドに移行する際に考慮すべき重要な要素についても学習します。最後に、この Moving to Managed Databases on AWS の 8 つのコースのカリキュラムの一環として、一般的なデータベースの移行に関するユースケースのコ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](mailto:aws-apj-marketing@amazon.com)



[twitter.com/awscloud\\_jp](https://twitter.com/awscloud_jp)



[facebook.com/600986860012140](https://facebook.com/600986860012140)



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



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



[twitch.tv/aws](https://twitch.tv/aws)

# Thank you!

木村 友則

アマゾン ウェブ サービス ジャパン合同会社  
技術統括本部 テクニカルソリューション部  
ソリューションアーキテクト

