このコンテンツは公開から3年以上経過しており内容が古い可能性があります
最新情報についてはサービス別資料もしくはサービスのドキュメントをご確認ください

[AWS Black Belt Online Seminar]

AWS Cloud Development Kit (CDK)

Archive

サービスカットシリーズ

Senior Solutions Architect
大村 幸敬

2020/03/03

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

過去資料
https://amzn.to/JPArchive

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

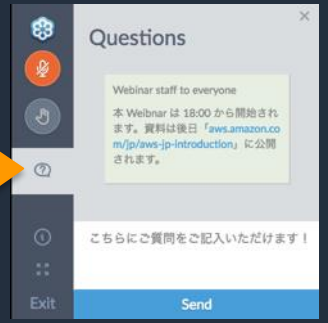
AWS Black Belt Online Seminar とは

「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

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

内容についての注意点

- 本資料では2020年03月03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.

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

大村 幸敬 (おおむら ゆきたか)

シニア ソリューションアーキテクト

- これからクラウドを使いはじめる エンタープライズ企業をサポート
- 運用系サービス & DevOps系サービス

好きなAWSのサービス： **AWS CLI, AWS CDK**

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

本セッションでお伝えすること

- CDKとは何か、なぜ必要なのか
- 各言語の AWS CDK の使いはじめかた
- CDKのコンセプトとTips
- CDKをより深く使うための最新情報

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Agenda

1. AWSの環境を構築するには
2. AWS CDK (Cloud Development Kit) の紹介
3. How to use AWS CDK
   TypeScript / Python / Java / C#
4. コアコンセプト
5. Tips & Trics
6. DiveDeep
7. 情報源

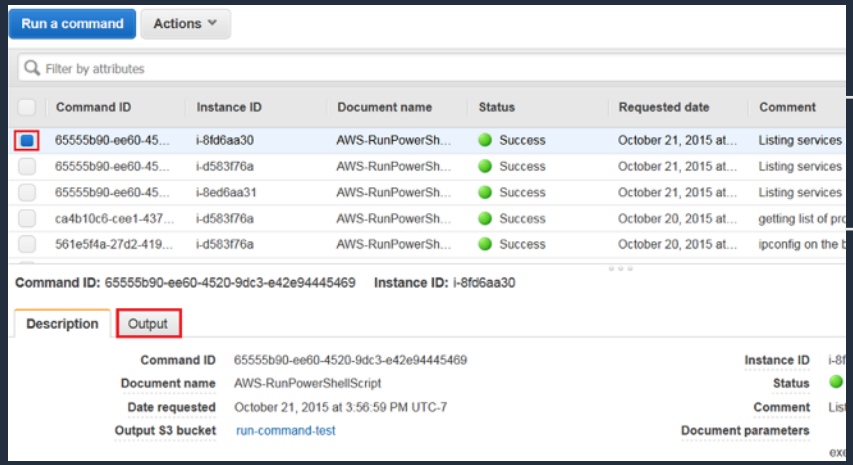
© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

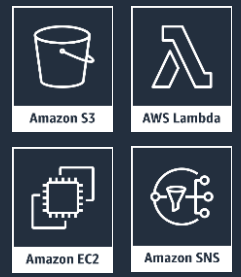
AWS の環境を構築するには

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

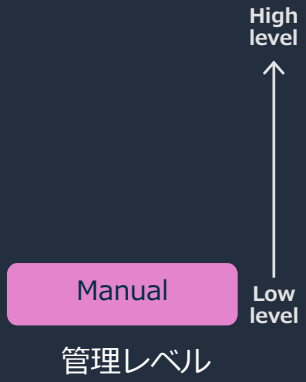
手動操作（マネジメントコンソール）

操作手順書が別途必要





○ 始めるのは簡単
× 繰り返し可能ではない
× エラーが起きやすい
× 時間がかかる

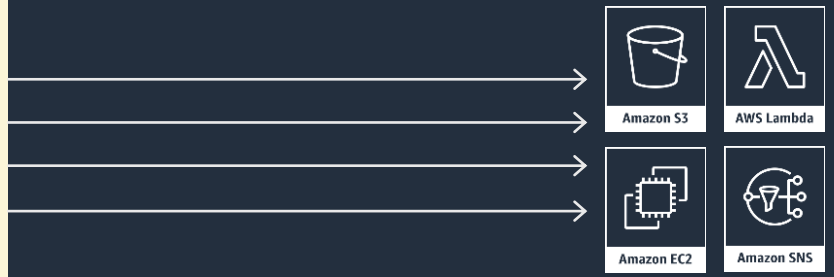


© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

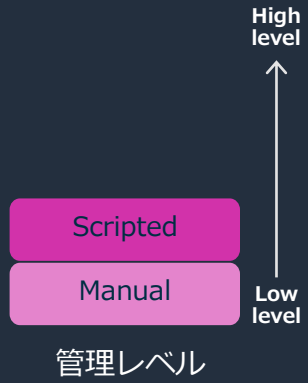
スクリプト (SDK, CLI)

操作手順の定義が可能

```ruby
require 'aws-sdk-ec2'

ec2 = Aws::EC2::Resource.new(region: 'us-west-2')

instance = ec2.create_instances({
  image_id: 'IMAGE_ID',
  min_count: 1,
  max_count: 1,
  key_name: 'MyGroovyKeyPair',
  security_group_ids: ['SECURITY_GROUP_ID'],
  instance_type: 't2.micro',
  placement: {
    availability_zone: 'us-west-2a'
  },
  subnet_id: 'SUBNET_ID',
  iam_instance_profile: {
    arn: 'arn:aws:iam::' + 'ACCOUNT_ID' + ':instance-profile/aws-opsworks-ec2-role'
  }
})
```



? APIコールが失敗したら何が起こる？
? どうやってアップデートする？
? リソースが準備完了なのはどうやって知る？
? どうやってロールバックする？



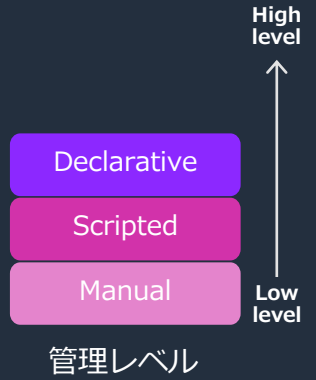
© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

プロビジョニング ツール (CloudFormationなど)

あるべき状態の定義が可能

CloudFormationや
Terraformで
デプロイ

○自動化が容易
○再生成可能
× ツール固有の記述方式
× 抽象化なし、詳細な記述



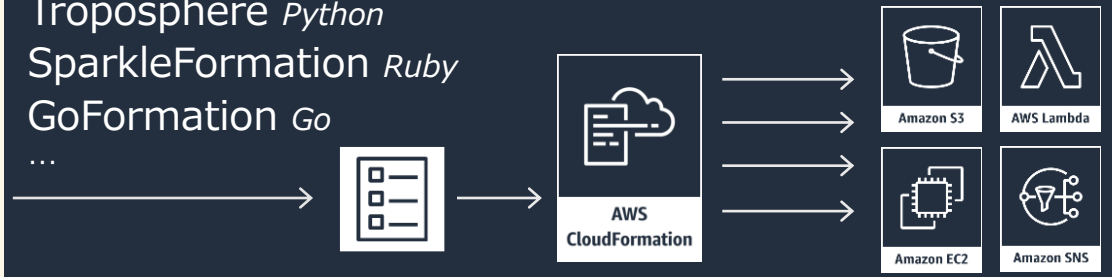
© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Document Object Models (DOMs)

あるべき状態の定義がコードで可能

```
from troposphere import Template
from troposphere.ec2 import VPC, Subnet, InternetGateway

t = Template()

VPC = t.add_resource(
    VPC(
        'VPC',
        CidrBlock='10.0.0.0/16',
        Tags=Tags(
            Application=ref_stack_id)))

subnet = t.add_resource(
    Subnet(
        'Subnet',
        CidrBlock='10.0.0.0/24',
        VpcId=Ref(VPC),
        Tags=Tags(
            Application=ref_stack_id)))
```

Troposphere *Python*
SparkleFormation *Ruby*
GoFormation *Go*
…

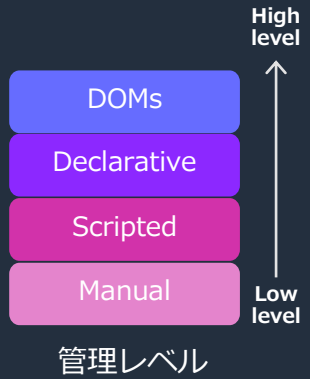


CFn テンプレートの生成

CFnでデプロイ

○ リアルコード *if*文、*for* ループ、*IDE*利用可能
○ あるべき状態の定義
× 抽象化は組み込まれていない

例えば、*Troposphere*で*VPC*を作成するには*128*行必要

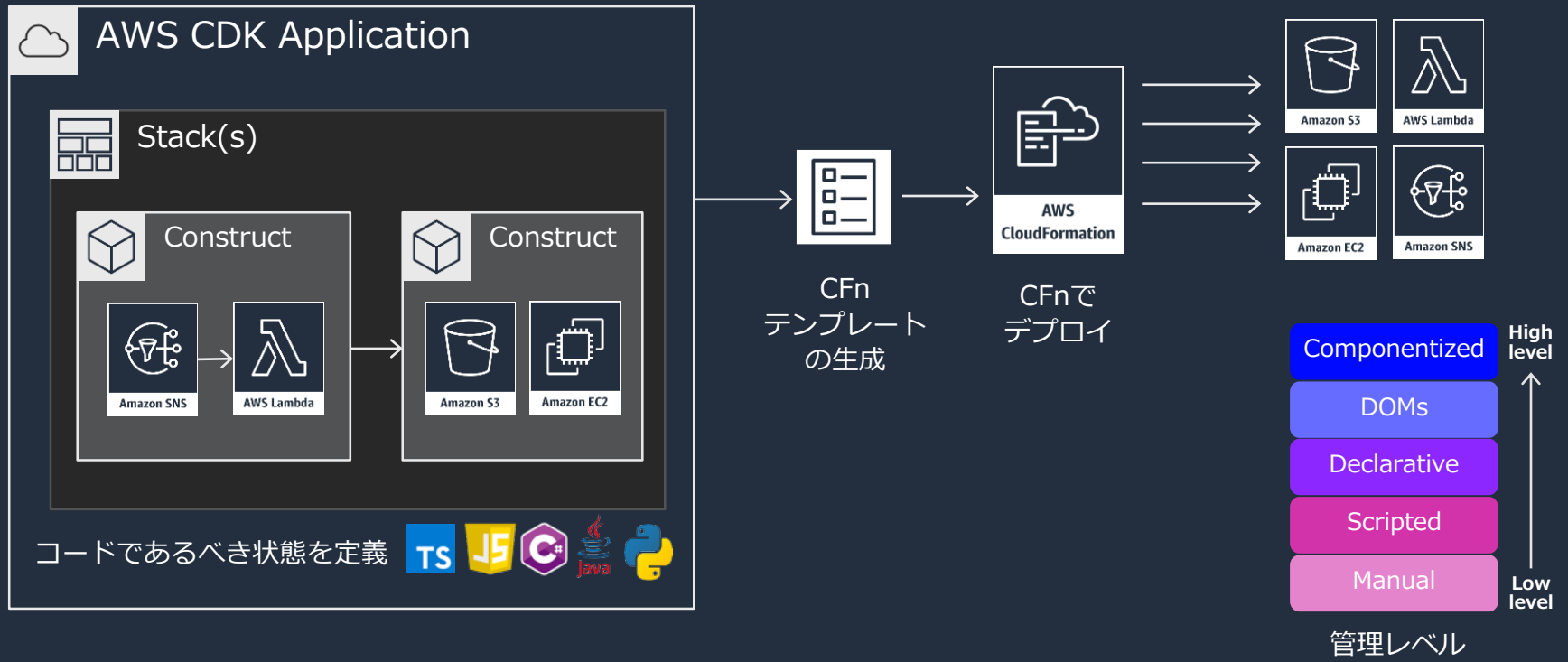


管理レベル

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CDK

あるべき状態の定義がコードで可能+抽象化



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CDK
(Cloud Development Kit)

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

AWS CDK

AWSの環境を 一般のプログラミング言語で記述できるツールキット

- ソースコードからCloudFormationテンプレートを生成
- AWSのベストプラクティスが定義されたライブラリ (Construct) によって少ないコードで記述可能
- オープンソースで開発されておりユーザが拡張可能
- 2018年8月 から Developer Preview
  2019年7月11日 TypeScript, Pythonが 一般利用可能 (GA) に
  2019年11月25日 Java, .NETがGA

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CDK 環境要件 ※2020/03/03 現在

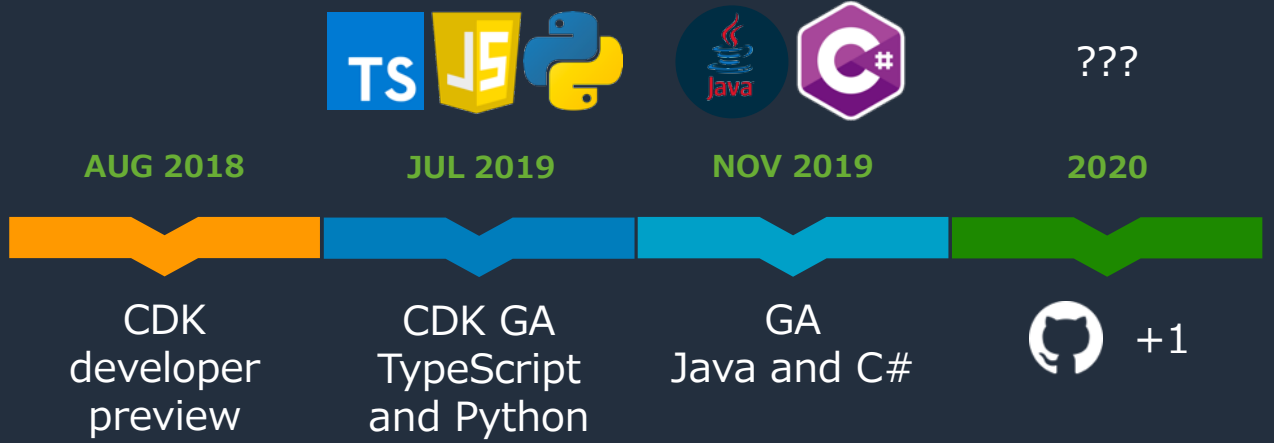
CDK
- Node.js (>= 10.3.0)

開発言語
- TypeScript, JavaScript
  (TypeScript >= 2.7)
- Python
  (Python >= 3.6)
- Java
  (Java 8 and Maven 3.5.4)
- C#
  (.NET Core >= 3.0)

一般利用可能:
TypeScript, Python
Java, C# (.NET)

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Multi-language libraries



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDKの特徴（CloudFormation に比較して）

CDKのメリット

- 一般のプログラミング言語が使える
  - 制御構文のほか、クラスや継承など抽象化の概念が利用できる
  - 言語によりエディタによる型チェック、サジェスト、API仕様の参照が可能
- 一般にコード量が少なくなる
- テストコードが記述できる
- 複数スタック間の依存関係が記述できる
- バックエンドがCFnである（豊富な実績とツールの安定性）

CDKから見た CloudFormation (CFn) の位置付け

- CDKを使った場合も最終的なデプロイはCFnで行われる
- 詳細な設定の記述には CFn と同様の記述が必要
- 従来の資産や SAM(Serverless Application Model) の活用はCFn

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

How to use AWS CDK

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

How to use AWS SDK

1. CDK 開発環境の準備
2. Demo - TypeScript編
3. Demo - Python編
4. 紹介 - Java編
5. 紹介 - C#編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK 開発環境の準備

1. 事前に必要な環境
   - AWS CLI
   - Node.js 10.3.0 以上の環境を用意
     - Windows: MSIインストーラ / Mac: homebrew & nodebrew

2. CDKの導入
   - "npm install -g aws-cdk"

3. 各言語の開発環境を用意（例: TypeScript）
   - お好きなエディタ （例：VisualStudio Code）

4. 初期コードを生成
   - "cdk init app --language=typescript"
   - サンプルコード付き → "cdk init sample-app --language=typescript"

5. Let's coding!

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK コードのデプロイ

1. CDKデプロイ管理用の環境（S3バケット）を作成（初回のみ）
   - "cdk bootstrap"

2. ビルドとCFnテンプレートの生成
   - ビルド - "npm run build など"
   - "cdk synth mystack"

3. デプロイ
   - "cdk deploy mystack"

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

CDK 開発環境の準備

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Demo

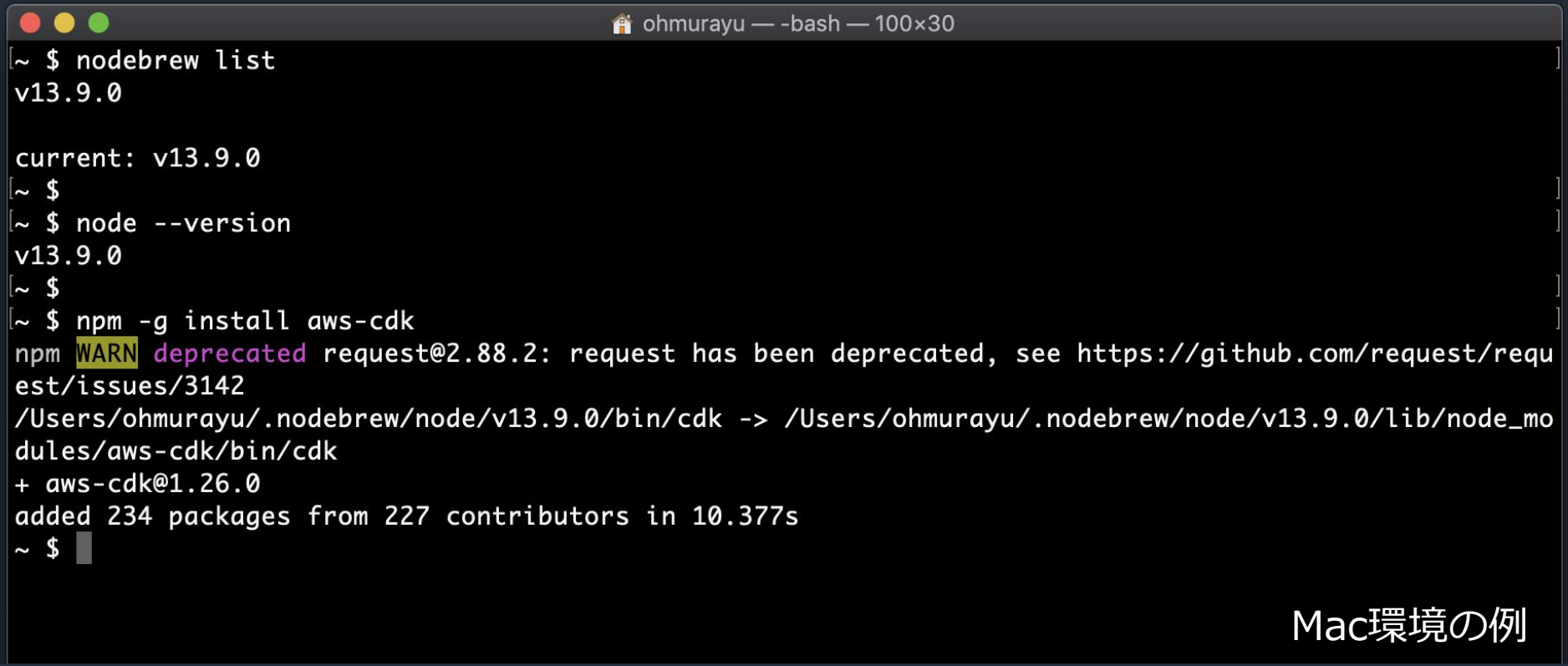
環境の準備

- Node.js 導入済み

1. CDK インストール
2. cdk init （初期コードの生成）
3. コードの確認
4. AWS CLI認証情報とリージョンの指定
5. cdk bootstrap

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDKインストール



Mac環境の例

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CLI 認証情報とリージョンの指定

CDKは内部でAWS CLIを使用するため、
CLIに渡す認証情報とリージョンの両方を指定する必要がある

方法は以下の3つ（優先順）

1. cdk コマンドに --profileを指定
    - ~/.aws/config に記載したプロファイルを使用
      aws_access_key_id, aws_secret_access_key, region の指定が必要

2. 環境変数
    - AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_DEFAULT_REGION を指定

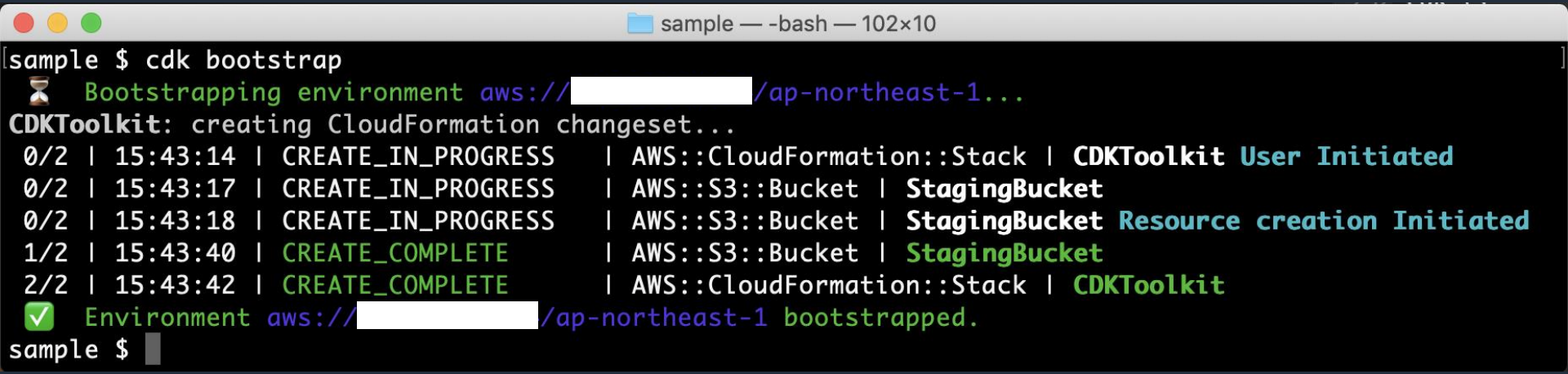
3. AWS CLI のデフォルトプロファイル

https://docs.aws.amazon.com/ja_jp/cdk/latest/guide/getting_started.html#getting_started_credentials

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

cdk bootstrap

- CDKがデプロイのために使用するS3バケットを作成
- 対象アカウント、リージョンにつき1回だけ実施



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Demo - TypeScript編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

デモの流れ

1.最初のデプロイ

1. npm run build
2. cdk synth
3. cdk deploy

2.コード変更とデプロイ

1. テストの実施(OK)
2. テストコード変更(NG)
3. コード修正と再テスト(OK)
4. cdk diff
5. cdk deploy

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

デモの流れ

1.最初のデプロイ

1. npm run build
2. cdk synth
3. cdk deploy

2.コード変更とデプロイ

1. テストの実施(OK)
2. テストコード変更(NG)
3. コード修正と再テスト(OK)
4. cdk diff
5. cdk deploy

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Demo - Python編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

デモの流れ

1. 環境の準備

- Node.js, Python3 導入済み

1. cdk init
2. コードの確認

2.最初のデプロイ

1. cdk synth
2. cdk bootstrap
3. cdk deploy

3.コード変更とデプロイ

1. コード編集
2. cdk diff
3. cdk deploy

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

デモの流れ

1. 環境の準備
   - Node.js, Python3 導入済み
   1. cdk インストール
   2. cdk init
   3. コードの確認

2. 最初のデプロイ
   1. cdk synth
   2. cdk bootstrap
   3. cdk deploy

3. コード変更とデプロイ
   1. コード編集
   2. cdk diff
   3. cdk deploy

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

紹介 - Java編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

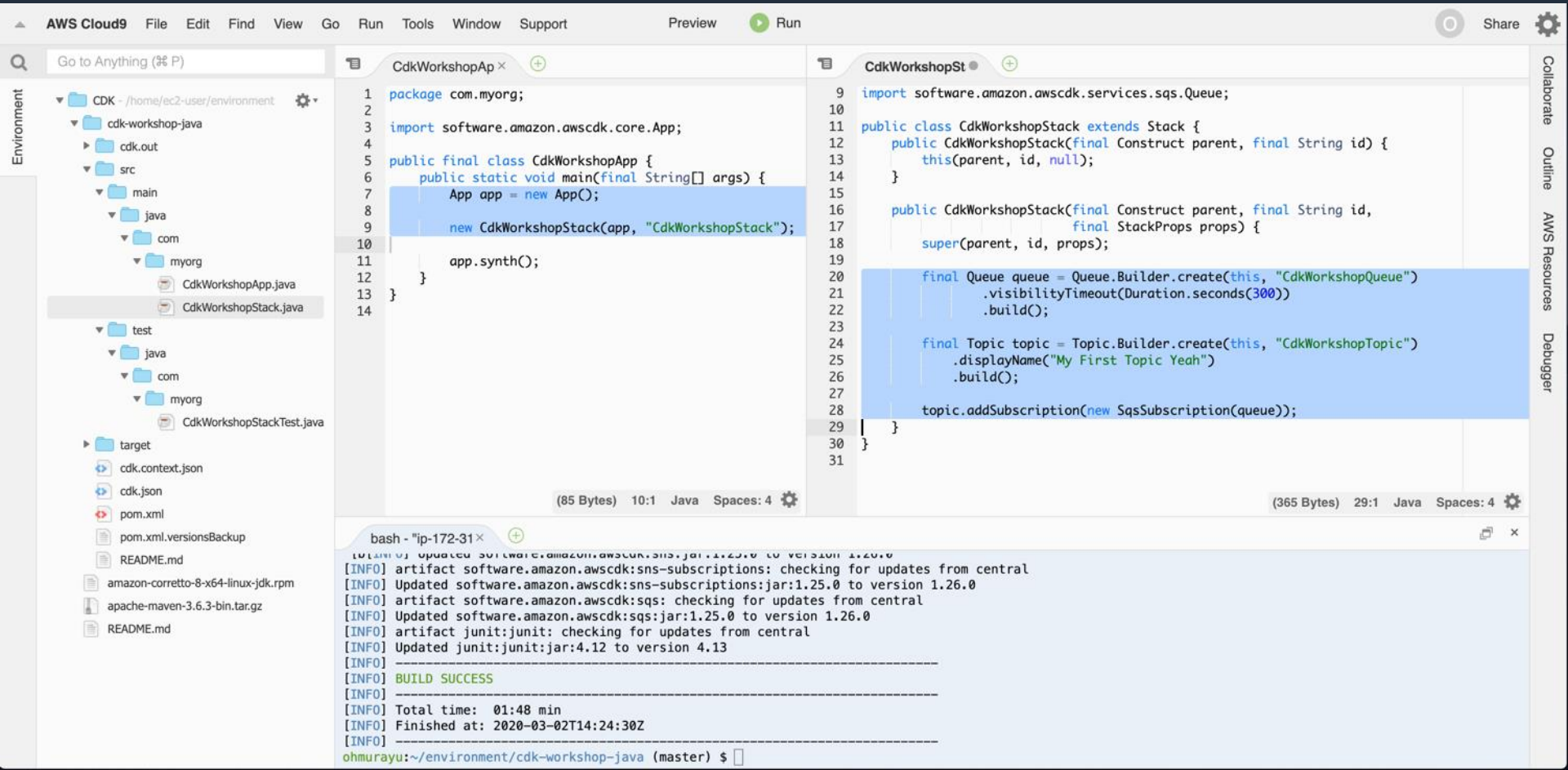
操作の流れ

1.環境の準備
1. 作業環境の準備(Cloud9)
2. cdk インストール
3. cdk init
4. コードの確認
5. JDK, Maven 導入

2.最初のデプロイ
1. cdk synth
2. cdk bootstrap
3. mvn package
4. cdk deploy

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Cloud9を使った CDK Javaコードの編集



Cloud9 環境への JDK, Mavenの導入

```
$ wget https://corretto.aws/downloads/latest/amazon-corretto-8-x64-linux-jdk.rpm
$ sudo yum localinstall amazon-corretto-8-x64-linux-jdk.rpm

$ wget http://ftp.kddilabs.jp/infosystems/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
$ tar xvf apache-maven-3.6.3-bin.tar.gz
$ sudo mv apache-maven-3.6.3 /opt
$ vi ~/.bash_profile
# --- 以下を追加
export PATH=/opt/apache-maven-3.6.3/bin:$PATH
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-amazon-corretto
# ---
$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment Corretto-8.242.08.1 (build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM Corretto-8.242.08.1 (build 25.242-b08, mixed mode)

$ javac -version
javac 1.8.0_242

$ mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/apache-maven-3.6.3
Java version: 1.8.0_242, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-1.8.0-amazon-corretto/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.14.165-103.209.amzn1.x86_64", arch: "amd64", family: "unix"
```

Java ビルドとデプロイ

```
# 依存ライブラリの最新化
$ mvn versions:use-latest-versions

# Stackの一覧
$ cdk ls

# ビルド
$ mvn package

# デプロイ
$ cdk deploy
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

```
ohmurayu:~/environment/cdk-workshop-java (master) $ cdk ls
CdkWorkshopStack
ohmurayu:~/environment/cdk-workshop-java (master) $ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.myorg:cdk-workshop >------------------------
[INFO] Building cdk-workshop 0.1
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ cdk-workshop ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/cdk-workshop-java/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ cdk-workshop ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ cdk-workshop ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/cdk-workshop-java/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ cdk-workshop ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/ec2-user/environment/cdk-workshop-java/target/test-classes
[INFO] /home/ec2-user/environment/cdk-workshop-java/src/test/java/com/myorg/CdkWorkshopStackTest.java: /ho
me/ec2-user/environment/cdk-workshop-java/src/test/java/com/myorg/CdkWorkshopStackTest.java uses or overri
des a deprecated API.
[INFO] /home/ec2-user/environment/cdk-workshop-java/src/test/java/com/myorg/CdkWorkshopStackTest.java: Rec
ompile with -Xlint:deprecation for details.
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ cdk-workshop ---
[INFO] Surefire report directory: /home/ec2-user/environment/cdk-workshop-java/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.myorg.CdkWorkshopStackTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.424 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ cdk-workshop ---
[INFO] Building jar: /home/ec2-user/environment/cdk-workshop-java/target/cdk-workshop-0.1.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.873 s
[INFO] Finished at: 2020-03-02T14:36:00Z
[INFO] ------------------------------------------------------------------------
ohmurayu:~/environment/cdk-workshop-java (master) $ cdk deploy
CdkWorkshopStack: deploying...

   CdkWorkshopStack (no changes)

Stack ARN:
arn:aws:cloudformation:ap-northeast-1:340935377354:stack/CdkWorkshopStack/f9329b50-5394-11ea-9d9b-0a50f898
64ca
```

紹介 - C#編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK for C# 開発環境

- OS: Windows
- IDE: Visual Studio 2019 + .NET Framework
- 実行環境: .NET Standard 2.1 compatible implementation
    - .NET Core v3.0 or later (v3.1 or later preferred)
    - .NET Framework v4.6.1 or later
    - Mono v5.4 or later on Linux or Mac OS X
- パッケージマネージャ: NuGet

https://docs.aws.amazon.com/ja_jp/cdk/latest/guide/work-with-cdk-csharp.html

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

操作の流れ

1.環境の準備

- Node.js, .NET Core 導入済み

1. cdk インストール
2. cdk init
3. コードの確認

2.最初のデプロイ

1. dotnet build src
2. cdk synth
3. cdk bootstrap
4. cdk deploy

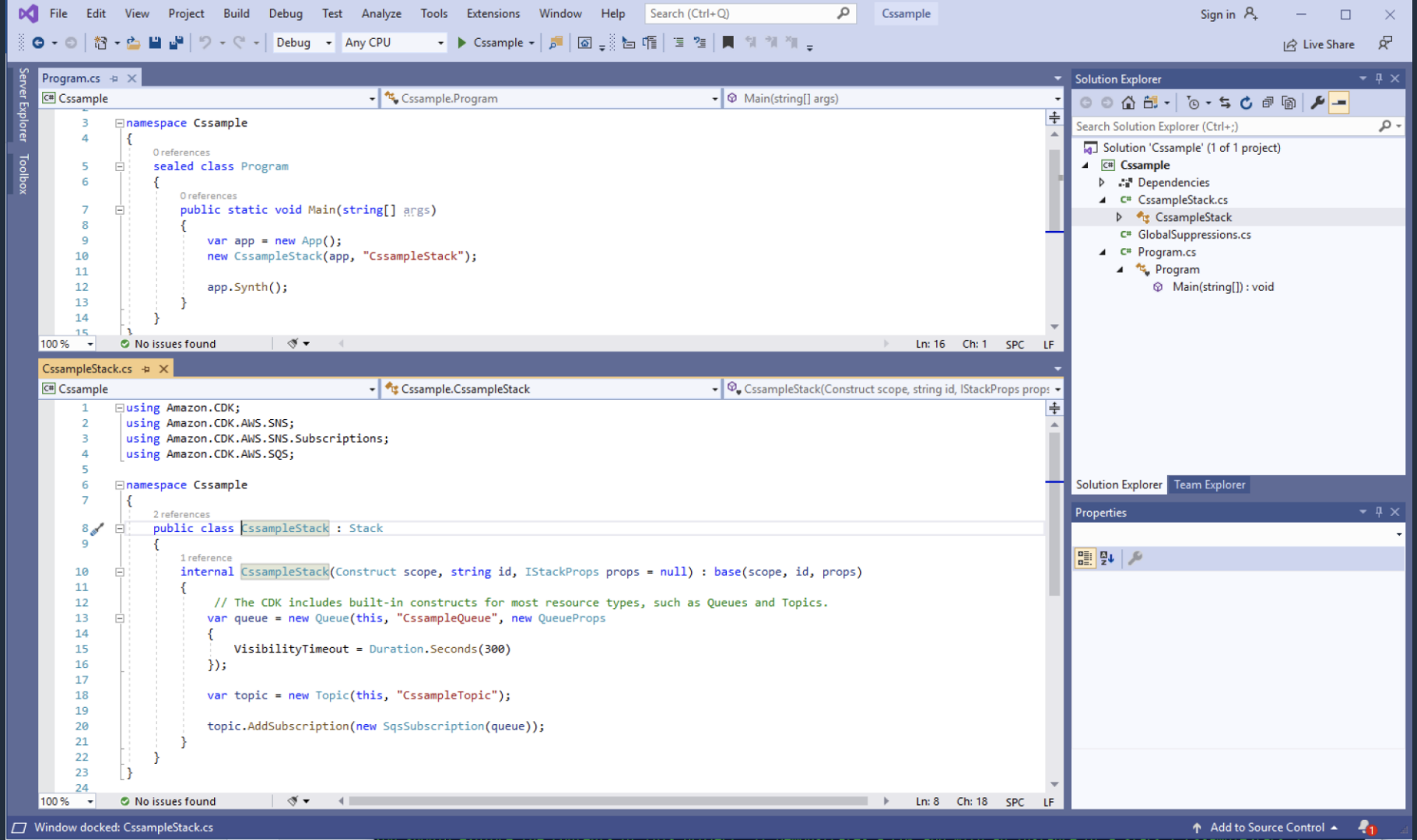
© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

```
H:\cssample>cdk init sample-app --language=csharp
Applying project template sample-app for csharp
Project `Cssample\Cssample.csproj` added to the solution.
Initializing a new git repository...
'git' is not recognized as an internal or external command,
operable program or batch file.
Unable to initialize git repository for your project.
# Welcome to your CDK C# project!

You should explore the contents of this project. It demonstrates a CDK app with an instance of a stack (`CssampleStack`)
which contains an Amazon SQS queue that is subscribed to an Amazon SNS topic.
The `cdk.json` file tells the CDK Toolkit how to execute your app.

It uses the [.NET Core CLI](https://docs.microsoft.com/dotnet/articles/core/) to compile and execute your project.

## Useful commands

* `dotnet build src` compile this app
* `cdk ls`           list all stacks in the app
* `cdk synth`       emits the synthesized CloudFormation template
* `cdk deploy`      deploy this stack to your default AWS account/region
* `cdk diff`        compare deployed stack with current state
* `cdk docs`        open CDK documentation

Enjoy!
```

→初期コードの中にソリューションファイル(.sln)があるので VisualStudioで開く

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Cssample
Sign in
Debug
Any CPU
Cssample
Live Share
Server Explorer
Toolbox
Program.cs
Cssample
Cssample.Program
Main(string[] args)
namespace Cssample
{
0 references
sealed class Program
{
0 references
public static void Main(string[] args)
{
var app = new App();
new CssampleStack(app, "CssampleStack");
app.Synth();
}
}
100 %
No issues found
Ln: 16 Ch: 1 SPC LF
CssampleStack.cs
Cssample
Cssample.CssampleStack
CssampleStack(Construct scope, string id, IStackProps props
using Amazon.CDK;
using Amazon.CDK.AWS.SNS;
using Amazon.CDK.AWS.SNS.Subscriptions;
using Amazon.CDK.AWS.SQS;
namespace Cssample
{
2 references
public class CssampleStack : Stack
{
1 reference
internal CssampleStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props)
{
// The CDK includes built-in constructs for most resource types, such as Queues and Topics.
var queue = new Queue(this, "CssampleQueue", new QueueProps
{
VisibilityTimeout = Duration.Seconds(300)
});
var topic = new Topic(this, "CssampleTopic");
topic.AddSubscription(new SqsSubscription(queue));
}
}
}
100 %
No issues found
Ln: 8 Ch: 18 SPC LF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Cssample' (1 of 1 project)
Cssample
Dependencies
CssampleStack.cs
CssampleStack
GlobalSuppressions.cs
Program.cs
Program
Main(string[]) : void
Solution Explorer
Team Explorer
Properties
Window docked: CssampleStack.cs
Add to Source Control

```
H:\cssample>dotnet build src
Microsoft (R) Build Engine version 16.4.0+e901037fe for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 80.59 ms for H:\cssample\src\Cssample\Cssample.csproj.
  Cssample -> H:\cssample\src\Cssample\bin\Debug\netcoreapp3.0\Cssample.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:09.15

H:\cssample>cdk ls
CssampleStack
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

```
H:\cssample>cdk deploy
This deployment will make potentially sensitive changes according to your current security approval level (--requir
e-approval broadening).
Please confirm you intend to make the following modifications:

IAM Statement Changes
```

	Resource	Effect	Action	Principal	Condition
+	${CssampleQueue.Arn}	Allow	sqs:SendMessage	Service:sns.amazonaws.com	"ArnEquals": { "aws:SourceArn": "${CssampleTopic}" }

```
(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

Do you wish to deploy these changes (y/n)? y
CssampleStack: deploying...
CssampleStack: creating CloudFormation changeset...
 0/6 | 12:39:58 AM | CREATE_IN_PROGRESS   | AWS::CloudFormation::Stack | CssampleStack User Initiated
 0/6 | 12:40:02 AM | CREATE_IN_PROGRESS   | AWS::SNS::Topic            | CssampleTopic (CssampleTopic390681D2)
 0/6 | 12:40:02 AM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata         | CDKMetadata
 0/6 | 12:40:03 AM | CREATE_IN_PROGRESS   | AWS::SQS::Queue            | CssampleQueue (CssampleQueue33CBF296)
 0/6 | 12:40:03 AM | CREATE_IN_PROGRESS   | AWS::SNS::Topic            | CssampleTopic (CssampleTopic390681D2) Resource
 creation Initiated
 0/6 | 12:40:03 AM | CREATE_IN_PROGRESS   | AWS::SQS::Queue            | CssampleQueue (CssampleQueue33CBF296) Resource
 creation Initiated
 0/6 | 12:40:04 AM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata         | CDKMetadata Resource creation Initiated
 1/6 | 12:40:05 AM | CREATE_COMPLETE      | AWS::SQS::Queue            | CssampleQueue (CssampleQueue33CBF296)
 2/6 | 12:40:05 AM | CREATE_COMPLETE      | AWS::CDK::Metadata         | CDKMetadata
 3/6 | 12:40:14 AM | CREATE_COMPLETE      | AWS::SNS::Topic            | CssampleTopic (CssampleTopic390681D2)
 3/6 | 12:40:16 AM | CREATE_IN_PROGRESS   | AWS::SQS::QueuePolicy      | CssampleQueue/Policy (CssampleQueuePolicy2C8F7
1F4)
 3/6 | 12:40:16 AM | CREATE_IN_PROGRESS   | AWS::SNS::Subscription     | CssampleQueue/CssampleStackCssampleTopic13926C
CA (CssampleQueueCssampleStackCssampleTopic13926CCAE4201416)
 3/6 | 12:40:16 AM | CREATE_IN_PROGRESS   | AWS::SQS::QueuePolicy      | CssampleQueue/Policy (CssampleQueuePolicy2C8F7
1F4) Resource creation Initiated
 4/6 | 12:40:17 AM | CREATE_COMPLETE      | AWS::SQS::QueuePolicy      | CssampleQueue/Policy (CssampleQueuePolicy2C8F7
1F4)
 4/6 | 12:40:17 AM | CREATE_IN_PROGRESS   | AWS::SNS::Subscription     | CssampleQueue/CssampleStackCssampleTopic13926C
CA (CssampleQueueCssampleStackCssampleTopic13926CCAE4201416) Resource creation Initiated
 5/6 | 12:40:17 AM | CREATE_COMPLETE      | AWS::SNS::Subscription     | CssampleQueue/CssampleStackCssampleTopic13926C
CA (CssampleQueueCssampleStackCssampleTopic13926CCAE4201416)
 6/6 | 12:40:19 AM | CREATE_COMPLETE      | AWS::CloudFormation::Stack | CssampleStack

 ✅  CssampleStack
```

How to use AWS SDK まとめ

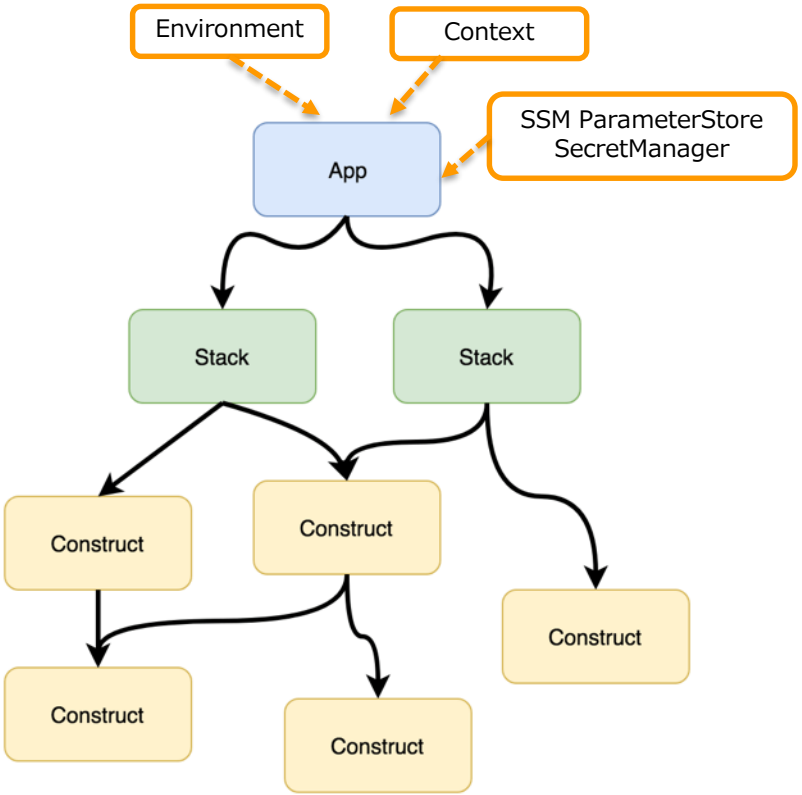
1. CDK 開発環境の準備
2. Demo - TypeScript編
3. Demo - Python編
4. 紹介 - Java編
5. 紹介 - C#編

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK コアコンセプト

- App
- Stack
- Construct
- Environment
- Context
- SSM ParameterStore / SecretManager

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



CDK アプリケーションの構成

App

- CloudFormationテンプレートの生成とデプロイに利用する最上位要素
- 複数のStackとその依存関係を定義

Stack

- CloudFormation Stack に該当しデプロイ可能な最小単位
- リージョンとアカウントを保持

Construct

- Stackに作成されるAWSリソース
- 標準でAWS Construct Libraryを提供
- 独自に定義したり配布することが可能

Construct

AWS CDK アプリの基本ビルディングブロック

- Cloudコンポーネントを表し、CloudFormationがリソースを作成するために必要なすべてをカプセル化

- 単一のリソースを表現したり（Amazon S3 Bucketなど）、
複数のAWS CDKリソースをまたがるハイレベル コンポーネントを
表現することも可能

- 標準で提供する AWS Constructs Libraryのほか、
既存Constructを継承して独自のConstructを定義したり
作成したConstructをパッケージとして配布することが可能

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS Constructs Library

AWS CDK が標準で提供する Construct のライブラリ

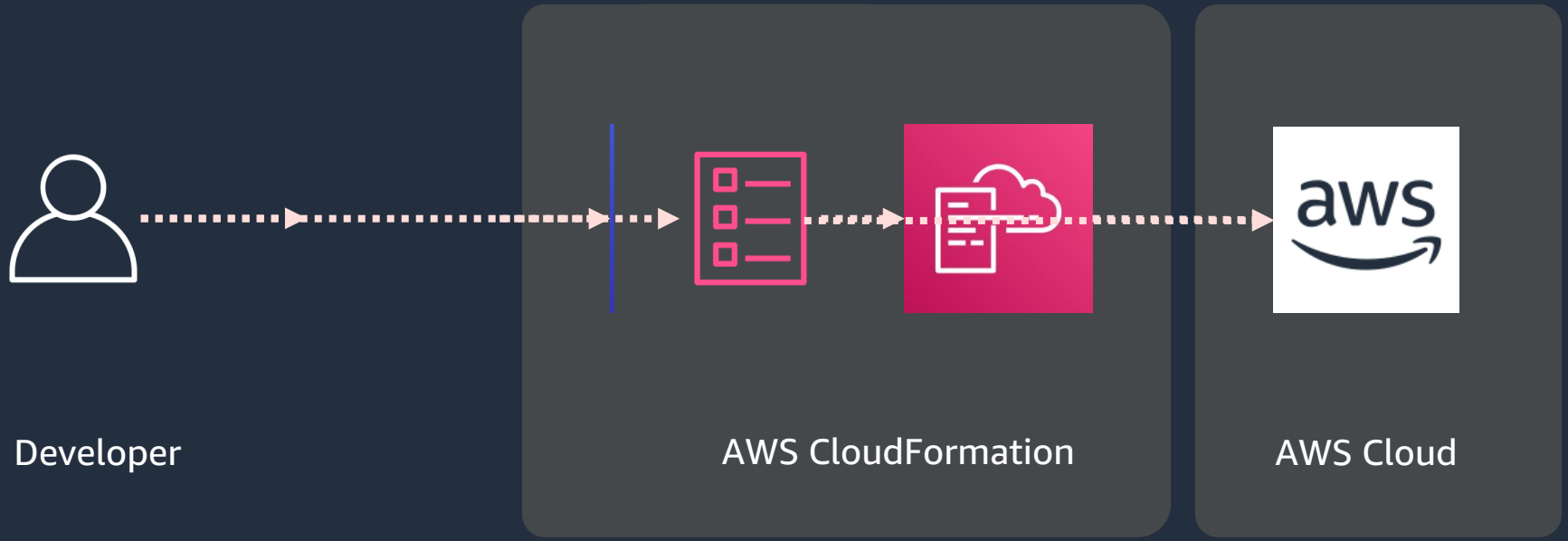
- High-level constructs (L2)
  - デフォルト値や便利なメソッドを定義したAWSリソースを表すクラス
  - 例）クラス s3 は メソッド s3.Bucket.addLifeCycleRule() を持つ

- Low-level constructs (L1)
  - CloudFormationリソースおよびプロパティと1:1で対応（自動生成される）
  - CfnXXXという名前 （例：s3.CfnBucket は AWS::S3::Bucket を意味）
  - すべてのプロパティを明示的に設定する必要がある

- Patterns
  - 複数のリソースを含む一般的な構成パターンを事前に定義したもの
  - aws-ecs-patterns.LoadBalancedFargateServiceなど

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS Construct Library



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CloudFormation resources

AWS CloudFormation template language

```yaml
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
  MyUser:
    Type: AWS::IAM::User
  MyUserPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt: [ MyBucket, Arn ]
              - Fn::Sub: "${MyBucket.Arn}/*"
        Version: "2012-10-17"
      PolicyName: MyUserPolicy
      Users:
        - Ref: MyUser
```

AWS CloudFormation resources

"L1"

AWS CloudFormation template language

```
const bucket = new CfnBucket(this, 'MyBucket');
const user = new CfnUser(this, 'MyUser');
new CfnPolicy(this, 'MyUserPolicy', {
  policyName: 'MyUserPolicy',
  policyDocument: new PolicyDocument({
    statements: [new PolicyStatement({
      actions: [
        's3:GetObject*',
        's3:GetBucket*',
        's3:List*'],
      resources: [
        bucket.bucketArn,
        `${bucket.bucketArn}/*`]
    })]
  }),
  users: [user.userName],
});
```

AWS Construct Library

"L2"

AWS CloudFormation resources

"L1"

AWS CloudFormation template language

```
const bucket = new s3.Bucket(this, 'MyBucket');
const user = new iam.User(this, 'MyUser');

bucket.grantRead(user);
```

Environment

- デプロイ先となるアカウントとリージョンの組み合わせ
- デプロイ先は明示的に指定することを推奨
- cdkコマンドで --profile オプションを指定した場合は CDK_DEFAULT_ACCOUNT, CDK_DEFAULT_REGION で参照可能

```
// コード内で明示的に指定する場合
const envEU = { account: '2383838383', region: 'eu-west-1' };
const envUSA = { account: '8373873873', region: 'us-west-2' };
new MyFirstStack(app, 'first-stack-us', { env: envUSA, encryption: false });
new MyFirstStack(app, 'first-stack-eu', { env: envEU, encryption: true });

// --profile で指定された情報を使う場合
new MyDevStack(app, 'dev', {
  env: { account: process.env.CDK_DEFAULT_ACCOUNT,
         region: process.env.CDK_DEFAULT_REGION
}});
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Context

- CDKコードを実行する際の各種パラメータ（Dev/Prod, VPC IDなど）をKey-Value型で定義、参照する仕組み
  - 最新のAMI IDが変わるなどして、意図せぬ変更が発生することを防ぐ
  - Construct が情報を持ち、this.node.tryGetContext('key') のようにアクセス
  - cdk context で参照可能

```
$ cat cdk.json
{
  "app": "npx ts-node bin/sample1.26.ts"
}
$ cat cdk.context.json
{
  "@aws-cdk/core:enableStackNameDuplicates": "true",
  "aws-cdk:enableDiffNoFail": "true"
}
$ cdk context
Context found in cdk.json:
┌───┬──────────────────────────────────────────┬──────────┐
│ # │ Key                                      │ Value    │
├───┼──────────────────────────────────────────┼──────────┤
│ 1 │ @aws-cdk/core:enableStackNameDuplicates  │ "true"   │
├───┼──────────────────────────────────────────┼──────────┤
│ 2 │ aws-cdk:enableDiffNoFail                 │ "true"   │
└───┴──────────────────────────────────────────┴──────────┘
Run cdk context --reset KEY_OR_NUMBER to remove a context key. It will be refreshed on the next CDK
synthesis run.
```

- 複数の定義方法
  - 現在の環境から情報を取得
  - cdk --context key=value で指定
  - cdk.json ファイルに指定
  - ~/.cdk.json ファイルに指定
  - コードでconstruct.node.setContext()で指定

- CDK コードで既存環境から情報を参照可能。
  その場合 プロジェクトの cdk.context.json ファイルに保存される
  - stack.availabilityZones(), HostedZone.fromLookup(), Vpc.fromLookup() など

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

SSM ParameterStore / Secret Managerから値を取得

cdk synth(CFnテンプレート生成)時またはデプロイ時に値を取得する

- SSM ParameterStore
    - aws-ssm.StringParameter.valueFromLookup()
        - cdk synth実施時に値を取得する
    - aws-ssm.StringParameter.valueForStringParameter()
        - デプロイ時にプレーンテキストを取得する（CFnパラメータへ設定）
    - aws-ssm.StringParameter.valueForSecureStringParameter()
        - デプロイ時にセキュア文字列を取得する（CFn Dynamic Referenceとして）
        - 渡せる場所は CFnの仕様に従うので注意（RDSパラメータなど）
- SecretManager
    - aws-secretsmanager.Secret.fromSecretAttributes()
        - デプロイ時にセキュア文字列を取得する（CFn Dynamic Referenceとして）

https://docs.aws.amazon.com/cdk/latest/guide/get_ssm_value.html
https://docs.aws.amazon.com/cdk/latest/guide/get_secrets_manager_value.html

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

パラメータの例: 既存VPCに DB接続情報を持つ Fargateを立てる

←App

```
const app = new cdk.App();
const fargate_stack = new FargateStack(app, 'FargateStack', {
  env: {
    account: process.env.CDK_DEFAULT_ACCOUNT,
    region: process.env.CDK_DEFAULT_REGION
  }});
```

↓Stack

```
export class FargateStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const vpc = ec2.Vpc.fromLookup(this, 'defVPC', {
      vpcName: this.node.tryGetContext('vpcName')
    });
    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });
    const user = ssm.StringParameter.fromStringParameterAttributes(this, 'SSMUser', {
      parameterName: '/MySystem/Dev/DB/User'
    });
    const password = ssm.StringParameter.fromSecureStringParameterAttributes(this, 'SSMPassword', {
      parameterName: '/MySystem/Prod/DB/Password',
      version: 1
    });
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "MyFargateService", {
      cluster: cluster,
      image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample"),
      environment: { 'user': user.stringValue },
      secrets: { 'password': ecs.Secret.fromSsmParameter(password) }
    });
  }
}
```

↓cdk.json

```
{
  "app": "npx ts-node bin/basestack.ts",
  "context": {
    "vpcName": "myvpc"
  }
}
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserve

パラメータの例: 既存VPCに DB接続情報を持つ Fargateを立てる

```
const app = new cdk.App();
const fargate_stack = new FargateStack(app, 'FargateStack', {
  env: {
    account: process.env.CDK_DEFAULT_ACCOUNT,
    region: process.env.CDK_DEFAULT_REGION
  }});
```

←App

Environment:
デフォルト環境の情報取得

↓Stack

```
export class FargateStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const vpc = ec2.Vpc.fromLookup(this, 'defVPC', {
      vpcName: this.node.tryGetContext('vpcName')
    });
    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });
    const user = ssm.StringParameter.fromStringParameterAttributes(this, 'SSMUser', {
      parameterName: '/MySystem/Dev/DB/User'
    });
    const password = ssm.StringParameter.fromSecureStringParameterAttributes(this, 'SSMPassword', {
      parameterName: '/MySystem/Prod/DB/Password',
      version: 1
    });
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "MyFargateService", {
      cluster: cluster,
      image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample"),
      environment: { 'user': user.stringValue },
      secrets: { 'password': ecs.Secret.fromSsmParameter(password) }
    });
  }
}
```

Environment+Context で
既存環境からVPCを取得

SSMの平文パラメータは
CFn パラメータ化（デプロイ時に確定）

SSMのSecureStringは
CFn DynamicReference化（デプロイ時に確定）

SSMパラメータをTaskDefinitionの
environment と secret に渡す

↓cdk.json

```
{
  "app": "npx ts-node bin/basestack.ts",
  "context": {
    "vpcName": "myvpc"
  }
}
```

Context:
VPCの名前を指定

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserve

CDK Tips&Trics

- アクセス許可指定のイディオム
- 複数スタックの管理
- テスト
- CDKのバージョンアップ
- 困った時は

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK アクセス許可設定のイディオム

- Principals
  - 操作主体（User, Role, サービス, アカウントなど）

- S3 や DynamoDB などアクセス対象となる Construct は アクセス許可を与えるため grant から始まるメソッドを持つ
  - 例） grantRead() , grantWrite(), grantReadWrite()

- PricipalやIGrantable インタフェースを実装した オブジェクトを渡すと適切な IAM Role/Policy が設定される
  - 例) `s3bucket.grantRead(lambdaFunction);`

https://docs.aws.amazon.com/cdk/latest/guide/permissions.html

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

CDK 複数スタックの管理

- スタック間でConstructを参照すればOK
- スタックの依存順序が定義可

```
const app = new cdk.App();
const vpc_stack = new VPCStack(app, 'VPCStack');
const alb_stack = new ALBStack(app, 'ALBStack', vpc_stack.vpc);
alb_stack.addDependency(vpc_stack);
```

依存関係のあるスタックの実装例↑

デプロイの例（ALBStack を作ると VPCStack も作られる）→

```
basestack $ cdk deploy ALBStack
Including dependency stacks: VPCStack
VPCStack
VPCStack: deploying...
VPCStack: creating CloudFormation changeset...
  0/25 | 17:27:47 | CREATE_IN_PROGRESS   | AWS::
 24/25 | 17:30:21 | CREATE_COMPLETE      | AWS::

 ✅ VPCStack

Stack ARN:
arn:aws:cloudformation:ap-northeast-1:340935377
ALBStack
ALBStack: deploying...
ALBStack: creating CloudFormation changeset...
 0/2 | 17:30:32 | CREATE_IN_PROGRESS    | AWS::C
 2/4 | 17:43:28 | CREATE_IN_PROGRESS    | AWS::E
2/4 Currently in progress: ALBStack, alb8A8B13C2

 ✅ ALBStack
```

自動生成されたエクスポート値 →

エクスポート名	エクスポート値	スタックの名前
VPCStack:ExportsOutputRefTheVPC92636AB00B2A4A70	vpc-0b9bbbcda361a209f	VPCStack

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK のテスト (DeveloperPreview - TypeScriptのみ)

"Testing infrastructure with the AWS Cloud Development Kit (CDK)"

- https://aws.amazon.com/jp/blogs/developer/testing-infrastructure-with-the-aws-cloud-development-kit-cdk/

- Snapshot tests (golden master tests)
  - あるべきCFnテンプレート全体を用意し、CDKで作成したテンプレートが一致することを確認する
  - CDKの開発では「Integration test」として使用
- Fine-grained assersions
  - CDKで作成したテンプレートの一部をチェックし、指定したリソースが特定のプロパティを持つことを確認する
  - expect(stack).toHaveResource('AWS::SQS::Queue', {プロパティ}) の形で検証
- Validation tests
  - Construct に与えられたパラメータが正しいことを検証するなど一般的なユニットテスト
  - expect(...).toThrowError() など

    - (注) @aws-cdk/assert ライブラリはデベロッパープレビューの状態であり、実装は今後変更される可能性があります

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CDK テストコードの例

Snapshot test code

```
import { SynthUtils } from '@aws-cdk/assert';
import cdk = require('@aws-cdk/core');
import vpcstack = require('../lib/vpc-stack');

test('VPC Stack template integ', () => {
    const app = new cdk.App();
    const stack = new vpcstack.VPCStack(app, 'MyTestStack');
    expect(SynthUtils.toCloudFormation(stack)).toMatchSnapshot();
});
```

Fine-grained test code

```
import { expect as expectCDK, haveResource } from '@aws-cdk/assert';
import cdk = require('@aws-cdk/core');
import Sample = require('../lib/sample-stack');

test('SQS Queue Created', () => {
    const app = new cdk.App();
    const stack = new Sample.SampleStack(app, 'MyTestStack');
    expectCDK(stack).to(haveResource("AWS::SQS::Queue",{
      VisibilityTimeout: 300
    }));
});
```

Snapshot test 実行例

```
basestack $ npm run test

> basestack@0.1.0 test /Users/ohmurayu/work/cdk-devday/basestack
> jest

 PASS  test/vpc-stack.test.ts
  ✓ VPC Stack template integ (212ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 passed, 1 total
Time:        2.743s
Ran all test suites.
```

("npm run test -- -u" で現在のスナップショットを保存)

Fine-grained test 実行例

```
sample $ npm run test

> sample@0.1.0 test /Users/ohmurayu/work/cdk-devday/sample
> jest

 PASS  test/sample.test.ts
  ✓ SQS Queue Created (78ms)
  ✓ SQS Queue Created - Short (41ms)
  ✓ SNS Topic Created (36ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        3.767s
Ran all test suites.
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

バージョンアップ (1/2) - CDK

グローバルスコープのCDKをバージョンアップする

1. CDKバージョン確認

```
$ cdk --version
```

2. アップデート有無のチェック

```
$ sudo npm install -g npm-check-updates

$ npx npm-check-updates -g aws-cdk
  （または ncu -g aws-cdk）

aws-cdk 1.23.0 → 1.26.0
```

3. CDKのアップグレード

```
$ sudo npm install -g aws-cdk
```

4. この後、各言語のローカルパッケージをそれぞれの手順で更新
   （次ページへ）

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

バージョンアップ (2/2) - CDKコードの依存ライブラリ

CDKプロジェクトディレクトリで ローカルパッケージをバージョンアップする

- TypeScript

```
$ npx npm-check-updates -u
$ npm install
```

- Python

```
$ source .env/bin/activate
(.env) $ pip list -o | sed -e '1,2d' | cut -f1 -d' ' | xargs pip install -U
```

- Java

```
$ mvn versions:use-latest-versions
```

- C#

```
$ nuget update
```

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

困った時は

cdk help
- CDKコマンドの使い方を確認

cdk doctor
- トラブルシュートに役立つ情報を収集
- バグレポートを送付する際に利用

```
$ cdk doctor
ℹ️  CDK Version: 1.0.0 (build e64993a)
ℹ️  AWS environment variables:
   - AWS_EC2_METADATA_DISABLED = 1
   - AWS_SDK_LOAD_CONFIG = 1
```

ドキュメントの "TroubleShooting"
- https://docs.aws.amazon.com/ja_jp/cdk/latest/guide/troubleshooting.html

GitHubの Issues
- https://github.com/aws/aws-cdk/issues

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

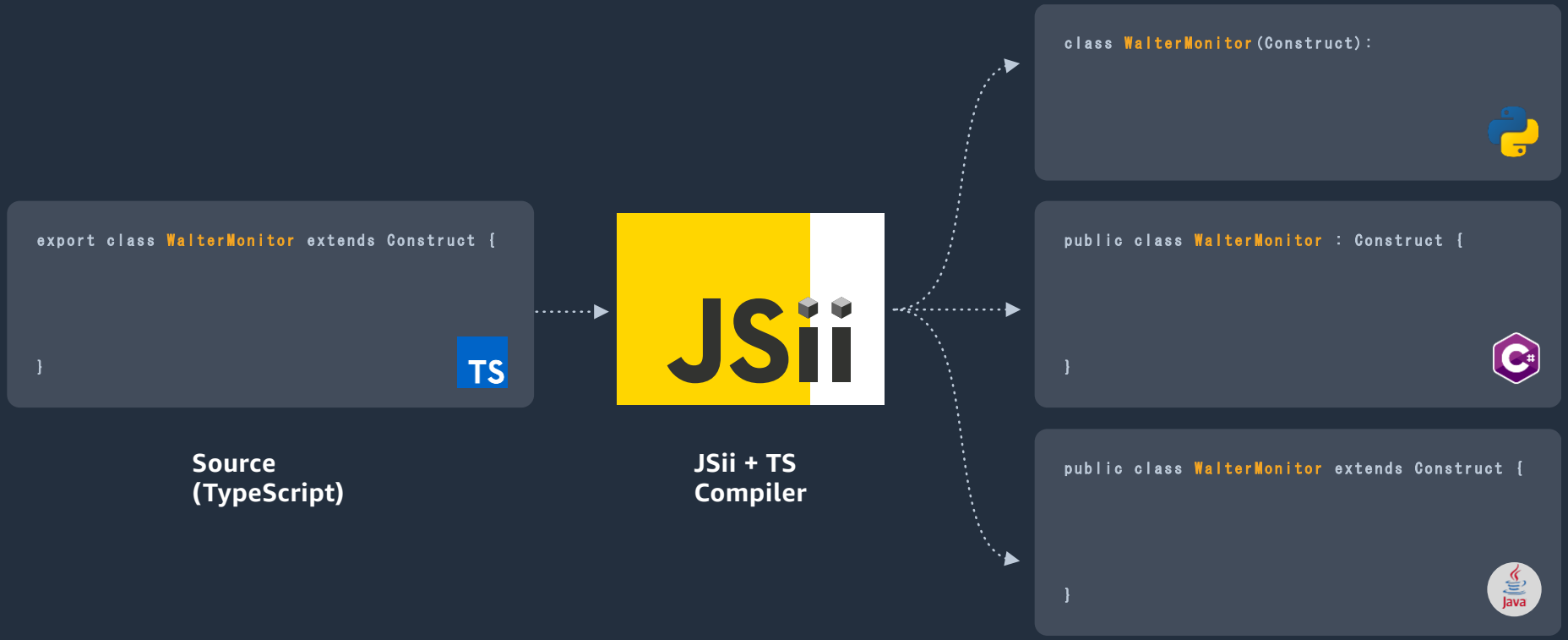
CDK DiveDeep

- jsii
- CDK RFCs
- FeatureFlags
- CDK Node.js Lambda Construct
- 参考）awsome-cdk
- 参考）既存環境からCDKコードを生成する

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Multi-language libraries



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

jsii

JavaScript で書かれたクラスに 他の言語からアクセスするためのライブラリ

- https://github.com/aws/jsii
- "Use javascript modules naturally from every programming language"

- AWSが CDK 実装のために開発しオープンソース化 (Apache License 2.0)
- CDK は TypeScript で実装され、jsii によって
  Python など他の言語でもタイムリーに反映されるようにしている
- jsii は開発やビルドツールでの使用を想定しておりパフォーマンスは限定的

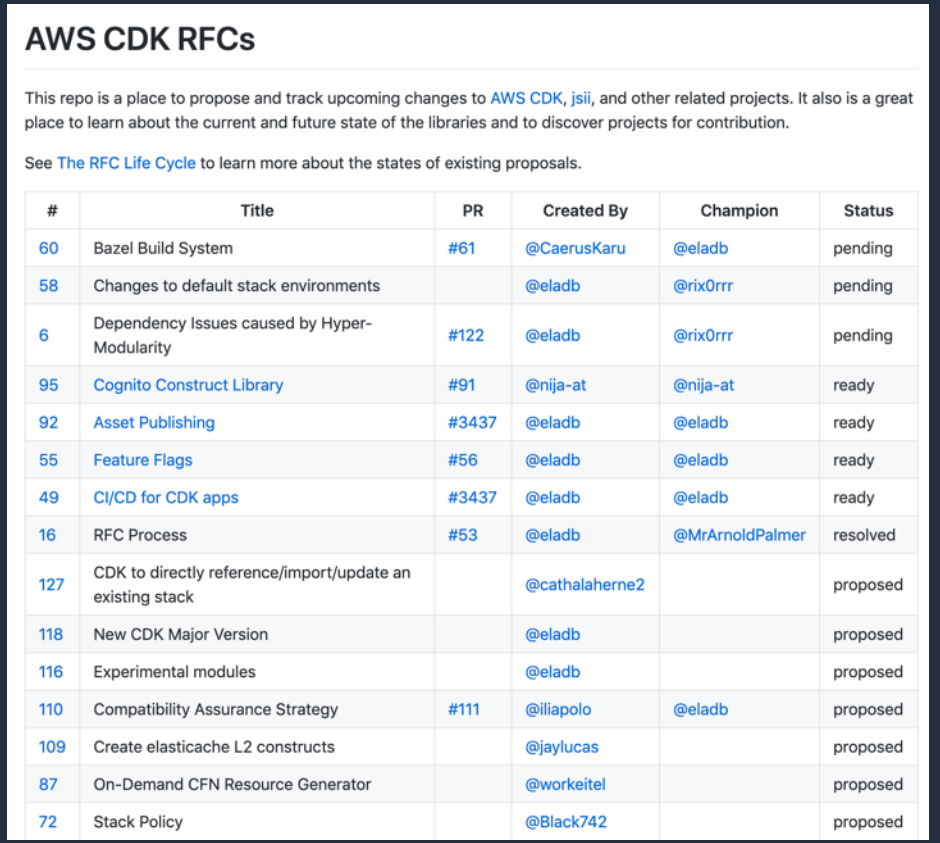
© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

CDK RFCs

https://github.com/aws/aws-cdk-rfcs

- CDKの新規機能の方向性を検討するDesign Docs のリスト
- チェックしておくとContributeできるプロジェクトが見つかるかも



AWS CDK RFCs

This repo is a place to propose and track upcoming changes to AWS CDK, jsii, and other related projects. It also is a great place to learn about the current and future state of the libraries and to discover projects for contribution.

See The RFC Life Cycle to learn more about the states of existing proposals.

#	Title	PR	Created By	Champion	Status
60	Bazel Build System	#61	@CaerusKaru	@eladb	pending
58	Changes to default stack environments		@eladb	@rix0rrr	pending
6	Dependency Issues caused by Hyper-Modularity	#122	@eladb	@rix0rrr	pending
95	Cognito Construct Library	#91	@nija-at	@nija-at	ready
92	Asset Publishing	#3437	@eladb	@eladb	ready
55	Feature Flags	#56	@eladb	@eladb	ready
49	CI/CD for CDK apps	#3437	@eladb	@eladb	ready
16	RFC Process	#53	@eladb	@MrArnoldPalmer	resolved
127	CDK to directly reference/import/update an existing stack		@cathalaherne2		proposed
118	New CDK Major Version		@eladb		proposed
116	Experimental modules		@eladb		proposed
110	Compatibility Assurance Strategy	#111	@iliapolo	@eladb	proposed
109	Create elasticache L2 constructs		@jaylucas		proposed
87	On-Demand CFN Resource Generator		@workeitel		proposed
72	Stack Policy		@Black742		proposed

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

FeatureFlags

```
{
  "app": "npx ts-node bin/tscdk.ts",
  "context": {
    "@aws-cdk/core:enableStackNameDuplicates": "true"
  }
}
```

- CDKの新バージョンで破壊的変更（breaking changes）が行われた場合に古い仕様のコードも正しく動作可能とするために導入されたフラグ
    - ランタイムコンテキスト (cdk.json または ~/.cdk.json) に設定
    - 設定可能な全ての値は feature.ts に記載
        - https://github.com/aws/aws-cdk/blob/master/packages/@aws-cdk/cx-api/lib/features.ts

- 例） @aws-cdk/core:enableStackNameDuplicates
    - 以前のCDKの仕様ではスタック名が自動生成されるため、同じ名前のスタックを複数作ることができなかった。実際はデプロイ先のリージョンや環境が違う場合など、同名のスタックが必要な場合が多かった。
    - そこで breaking change として同名のスタックを作れるような機能が導入されたが、デフォルトではこれは有効でなく、設定によって有効化されるようにした。（現在は cdk init するとデフォルトで有効なコードが生成される）

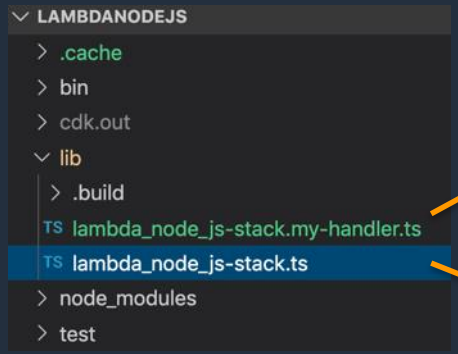
https://docs.aws.amazon.com/cdk/latest/guide/featureflags.html
https://github.com/aws/aws-cdk-rfcs/blob/master/text/0055-feature-flags.md

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

CDK Node.js Lambda Construct (Experimental)

LambdaとCDKのコードをすべてTypeScriptで一括開発&デプロイ



{スタックファイル名}.ts に対し
{スタックファイル名}.{ハンドラ名}.ts が
Lambda本体となる（別途指定も可能）

```ts
import {APIGatewayEvent, APIGatewayProxyHandler} from 'aws-lambda';

export const handler: APIGatewayProxyHandler = async(event: APIGatewayEvent) => {
  const response = {
      statusCode: 200,
      body: JSON.stringify('Hello from Lambda!'),
  };
  return response;
};
```

lambda_node_js-stack.my-hander.ts (Lambdaの実装)

```ts
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda-nodejs';

export class LambdaNodeJsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new lambda.NodejsFunction(this, 'my-handler');
  }
}
```

lambda_node_js-stack.ts (CDKの実装)

https://github.com/aws/aws-cdk/tree/master/packages/%40aws-cdk/aws-lambda-nodejs

参考: https://qiita.com/tetsuya-zama/items/600bb0e187e9dead6e68

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

awesome-cdk（参考）

eladb/awesome-cdk

CDKに関する便利な情報を集めているリポジトリ（個人が運営）

- Open CDK Guide opinionated set of tips and best practices
  - kevinslin/open-cdk
- punchcard type-safe AWS infrastructure
  - punchcard/punchcard
- aws-cdk-pure purely functional CDK
  - fogfish/aws-cdk-pure
- cdk-clj a clojure wrapper for the CDK
  - StediInc/cdk-clj
- cdk-components a collection of higher-level cdk constructs
  - cloudcomponents/cdk-components
- CDK GitHub Action
  - ScottBrenner/aws-cdk-action

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

既存環境からCDKコードを生成する（参考）

- 既存環境からCloudFormationテンプレートを生成
  そこからさらにCDKコードを手で作成する
  - CFn リソースインポート + Drift Detectionでテンプレート作成
  - CDK CloudFormation Disassemblerを使って .ts コードを生成
    L1ライブラリのコードが生成される (experimental)
    https://github.com/aws/aws-cdk/tree/master/packages/cdk-dasm

- 既存環境から直接CDKコードを生成する
  - 3rd Paryオープンソースツールの「Former2」を使用する
    - https://former2.com/
    - 既存環境からCFnテンプレートやCDKコードを生成するツール
    - セキュリティを気にする場合はローカルインストールで利用も可能
  - L1ライブラリのコードが生成される（CFnと同様の内容）

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS CDK の情報源

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

ドキュメントと参考資料

GitHub
- https://github.com/aws/aws-cdk

APIリファレンス
- https://docs.aws.amazon.com/cdk/api/latest/docs/aws-construct-library.html

ドキュメント（Developer Guide） - 更新が早いため英語版の参照をおすすめ
- https://docs.aws.amazon.com/cdk/latest/guide/home.html

- セキュリティ https://docs.aws.amazon.com/cdk/latest/guide/security.html

- バージョンと安定性モデル https://docs.aws.amazon.com/cdk/latest/guide/reference.html

- Assets https://docs.aws.amazon.com/cdk/latest/guide/assets.html
  - ローカルファイルやDockerImageをCDKコードにバンドルする方法

- TroubleShooting https://docs.aws.amazon.com/ja_jp/cdk/latest/guide/troubleshooting.html

ToolKit for VisualStudio Code
- https://aws.amazon.com/jp/visualstudiocode/

循環参照の解決方法（CFnでの考え方）
- https://aws.amazon.com/jp/blogs/infrastructure-and-automation/handling-circular-dependency-errors-in-aws-cloudformation/

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

ハンズオン

- サンプル
  - https://github.com/aws-samples/aws-cdk-examples

- ワークショップ（おすすめ）
  - https://cdkworkshop.com
  - 日本語版: http://bit.ly/cdkworkshopjp

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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。

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

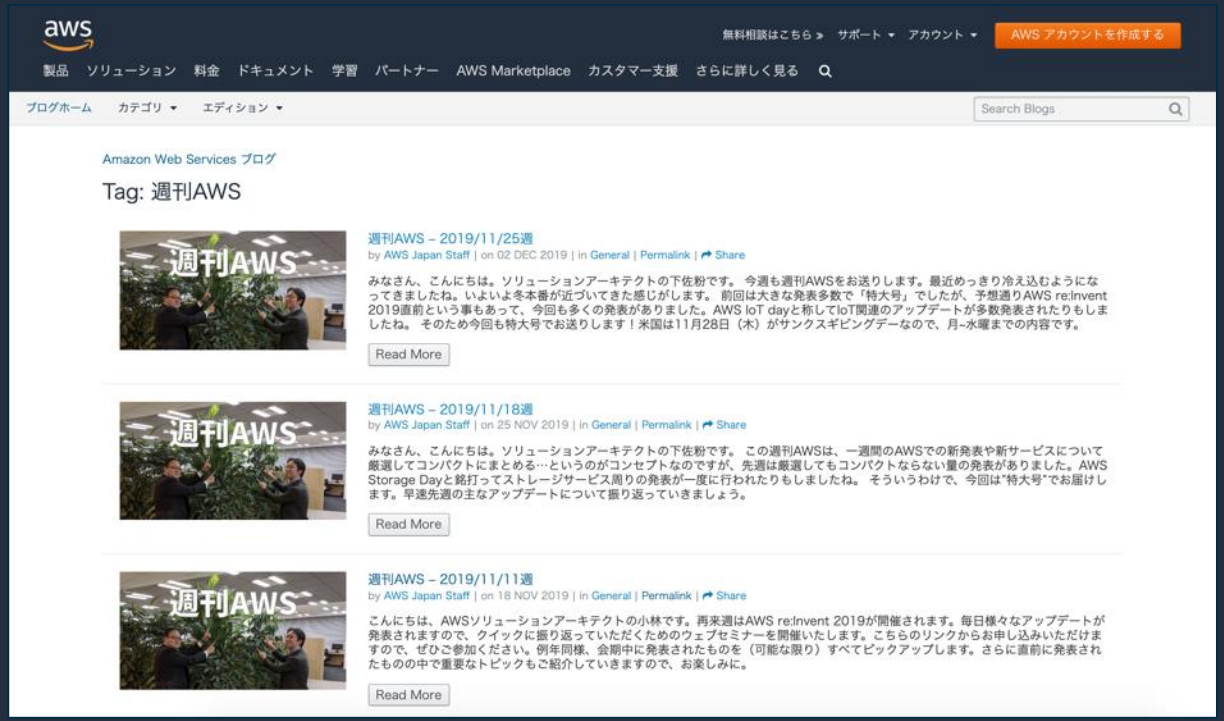
AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

日々のアップデートチェックは週刊AWSで



週刊AWS

© 2020, Amazon Web Services, Inc. or its Affiliates. Al

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]

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.