


WindowsでAWS CDK(C#)の開発環境を整えてみた

#Cloud Development Kit #C++ #AWS CDK #AWS

 Takuya Shibata

2019.08.22



しばたです。

先月[TypeScriptとPythonでの利用がGA](#)したAWS Cloud Development Kit (CDK)ですが、Developers.IOで静かな？ブームとなっている様で多くの社員が記事を書いています。このブームに乗る形で私もWindows環境で言語にC#を選び試してみました。

内容的には先日梶原が書いたこちらの記事の影響を受けており、そこそこ内容が被るかもしれませんが。

[WindowsでAWS CDK\(Java\)の開発環境を整えてみた](#)

注意事項

本日(2019年8月22日)時点でTypeScriptとPythonでの利用はGAしてますが、C#での利用はDeveloper Previewです。

本記事の内容は将来的に変更される可能性がある点にご留意ください。

参考資料

参考資料として.NET(C#)のCDKが登場した際のAWS Developers Blogの記事とGitHubリポジトリのドキュメントを使用しました。

- [AWS CDK for .NET](#)
- [aws/aws-cdk](#)

AWS Developers Blogの記事について、当時はC#の言語サポートが **dotnet** という形で記載されていますが、現在は **csharp** と改善されていますので適宜差し替えてご覧ください。

当時の記述

```
cdk init app --language dotnet
```

現在の記述 : C# は csharp に改善

```
cdk init app --language csharp
```

また、基本的な手順としては以下の公式ドキュメントが役に立ちます。

- [Getting Started With the AWS CDK](#)

環境構築

今回は私の開発PC(64bit版 Windows 10 May 2019 Update (1903))を検証環境とし、C#でCDKを利用するに際して以下のツールをインストールしています。

- Node.js
- AWS CDK CLI
- .NET Core SDK

また、開発環境としてGit + Visual Studio Code + C#拡張を使用しています。本記事では上述のツールのインストール手順は紹介しますがVisual Studio Codeの設定については触れません。Gitを使いC#のコードをよしなにコンパイルできる環境がある前提で話を進めていきます。

Node.js

CDKの基本となる **cdk** コマンドはNode.js製ですのでWindowsにNode.jsをインストールします。Node.js 8.11以上のバージョンである必要があります。今回は執筆時点の最新バージョンであるNode.js 12.8.1をインストールしました。

PowerShellコンソールを開き、以下のコマンドを実行すると既定の設定でサイレントインストールできます。

```
# MSIインストーラーをダウンロードしてサイレントインストール
$msiPath = Join-Path $env:TEMP 'node-v12.8.1-x64.msi'
Invoke-WebRequest -Uri 'https://nodejs.org/dist/v12.8.1/node-v12.8.1-x64.msi' -OutFile $msiPath
# 全てデフォルト設定のままサイレントインストール
msiexec.exe /i $msiPath /passive
```

インストール後コンソールを再起動するとPATH環境変数が更新され、 `node` コマンドや `npm` コマンドが実行可能になります。

```
C:\> node --version
v12.8.1
C:\> npm --version
6.10.2
```

AWS CDK CLI

`cdk` コマンドを利用するためにAWS CDK CLIをnpmからインストールします。 PowerShellコンソールから以下のコマンドを実行します。

```
# npmからインストール
npm i -g aws-cdk
```

今回はVer.1.5.0がインストールされました。

```
C:\> cdk --version
1.5.0 (build c020efa)
```

.NET Core SDK

C#のコードをコンパイルするために.NET Core SDKをインストールします。 要件としては.NET Core 2.0以降である必要があります。 今回は執筆時点で最新のインストーラー版SDKをインストールしました。

PowerShellコンソールを開き、以下のコマンドを実行するとサイレントインストールできます。 インストーラー版SDKはマシン全体に設定が反映されるため要管理者権限です。

```
# 現在最新の.NET Core SDKのバージョンを取得
$commitHash, $version = -split (Invoke-RestMethod -Uri https://dotnetcli.azureedge.net/dotnet/Sdk/$version/dotnetcli.msixbundle
# MSIインストーラーをダウンロードしてサイレントインストール
$msiPath = Join-Path $env:TEMP "dotnet-sdk-$version-win-x64.exe"
Invoke-WebRequest -Uri "https://dotnetcli.azureedge.net/dotnet/Sdk/$version/dotnetcli.msixbundle" -OutFile $msiPath
& $msiPath /install /passive
```

今回はVer.2.2.301がインストールされました。

```
C:\> dotnet --version
2.2.301
```

AWS認証情報

CDKでは認証情報にAWS CLIと同じ `~/.aws/config` および `~/.aws/credentials` ファイル (Windowsの場合は `%USERPROFILE%\aws\config` 、 `%USERPROFILE%\aws\credentials`)を使用します。設定すべき内容はAWS CLIと同様で、プロファイル設定をすると `cdk` コマンドの `--profile` パラメーターで使うことができます。

```
# ~/.aws/config 設定例
[profile test-profile]
region=ap-northeast-1

# ~/.aws/credentials 設定例
[test-profile]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

注意事項

[CDKのドキュメント](#)に例示されている様に、アクセスキーの機密情報も `~/.aws/config` にまとめて記述することもできますが、この場合は空ファイルで良いので `~/.aws/credentials` ファイルを作成しておいてください。 `~/.aws/credentials` が無いと `cdk` コマンド実行時に

AWS region must be configured either when you configure your CDK stack or through the environment

といったエラーを吐いてしまいます。

- 参考 : [cdk deploy issue](#)

以上で環境構築は完了です。

やってみた

ここから実際にCDKを使ってみます。

プロジェクト作成

最初にCDKのプロジェクトを作成します。 任意のディレクトリ(本記事では `C:\temp\cdksample`) に移動し `cdk init app` コマンドを実行すると空のプロジェクトを作成できます。 言語はC#を使いたいので `--language csharp` も合わせて指定します。

```
cdk init app --language csharp
```

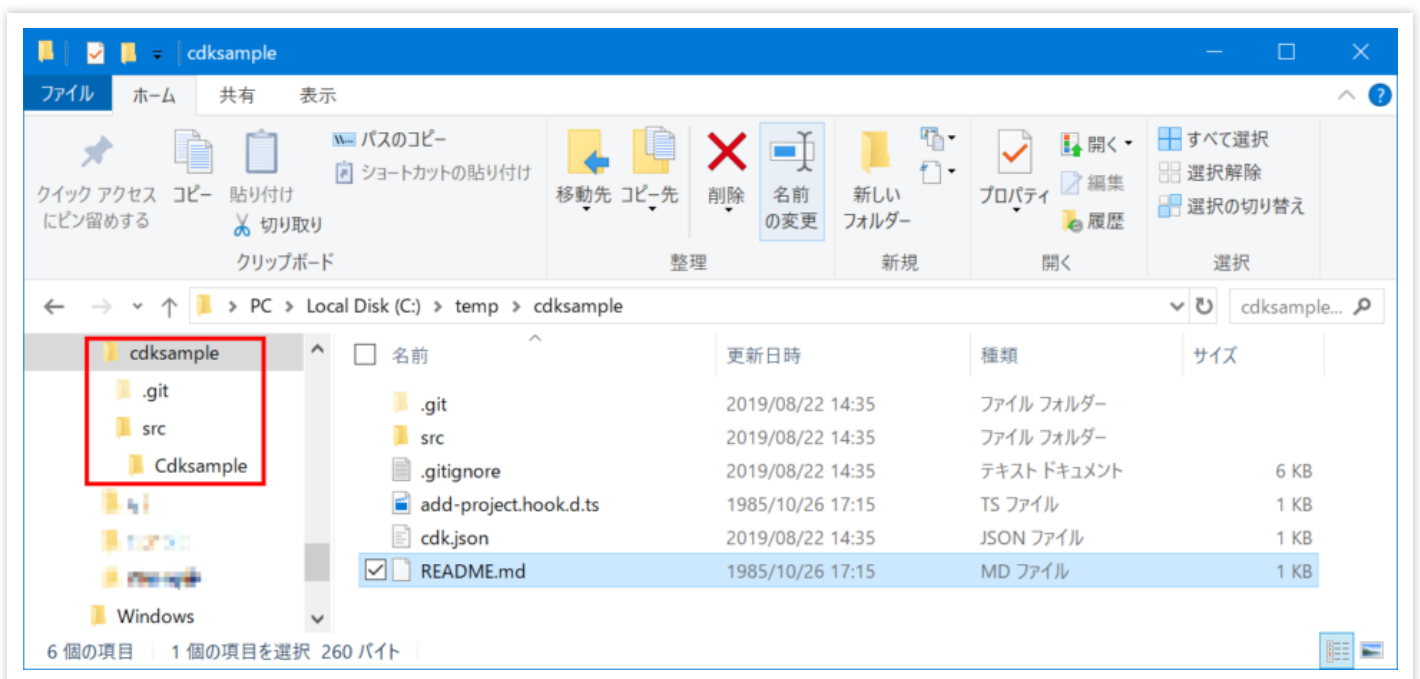
```

posh-git ~ cdksample [master]
C:\temp\cdksample> cdk init app --language csharp
Applying project template app for csharp
v\W\F\N\g\`Cdksample\Cdksample.csproj` [V\...B
Initializing a new git repository...
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in add-project.hook.d.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in cdk.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/Cdksample.sln.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/Cdksample/Cdksample.csproj.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/Cdksample/CdksampleStack.cs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/Cdksample/Program.cs.
The file will have its original line endings in your working directory
# Useful commands

* `dotnet build src` compile this app
* `cdk deploy`      deploy this stack to your default AWS account/region
* `cdk diff`        compare deployed stack with current state
* `cdk synth`       emits the synthesized CloudFormation template
C:\temp\cdksample [master]>

```

画面表示が一部文字化けし改行コードの自動変換が入るもののディレクトリ配下に必要なファイルは生成されます。



カレントディレクトリに **cdk.json** が生成され、C#ソース一式は **src** ディレクトリ配下に生成されます。treeコマンドの結果以下のようになります。

```

C:.\
| .gitignore
| add-project.hook.d.ts
| cdk.json
| README.md

```

```
|
|_src
|  Cdksample.sln
|_Cdksample
  Cdksample.csproj
  CdksampleStack.cs
  Program.cs
```

ソースコードについて

生成されたソースはエントリポイントである `Program.cs` と空のスタックである `CdksampleStack.cs` があり、それぞれ以下の様になっています。

- `Program.cs`

```
using Amazon.CDK;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Cdksample
{
    class Program
    {
        static void Main(string[] args)
        {
            var app = new App(null);
            new CdksampleStack(app, "CdksampleStack", new StackProps());
            app.Synth();
        }
    }
}
```

- `CdksampleStack.cs`

```
using Amazon.CDK;

namespace Cdksample
{
    public class CdksampleStack : Stack
    {
```

```
public CdksampleStack(Construct parent, string id, IStackProps props) : I
{
    // The code that defines your stack goes here
}
}
```

C#のCDKにはまだ公式のサンプルソースが公開されていないため、本記事では単純にVPCを1つだけ定義する様に `CdksampleStack.cs` を修正します。

- 修正した `CdksampleStack.cs`

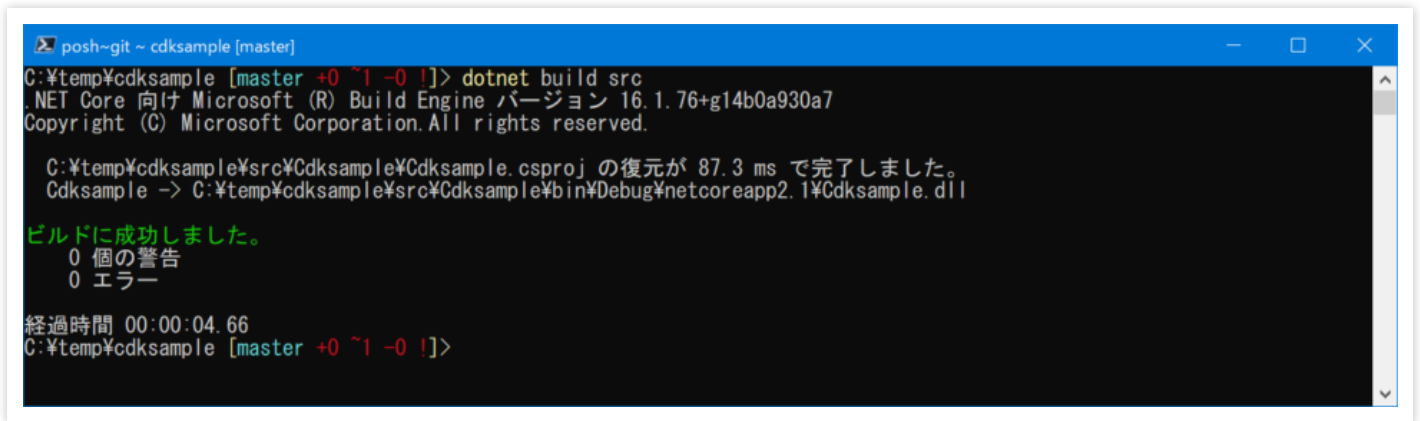
```
using Amazon.CDK;
using Amazon.CDK.AWS.EC2;

namespace Cdksample
{
    public class CdksampleStack : Stack
    {
        public CdksampleStack(Construct parent, string id, IStackProps props) : I
        {
            // Create a VPC
            var vpc = new Vpc(this, "SampleVPC", new VpcProps
            {
                Cidr = "172.18.0.0/16"
            });
        }
    }
}
```

ソースのビルド + CloudFormation Templateの生成(synth)

ソースを修正したらビルドしてアセンブリを生成します。こちらは普通のC#のビルドですので `dotnet build` コマンドで行います。ソリューションファイル `Cdksample.sln` が `.\src\` フォルダにありますので実際に実行するコマンドは以下のようになります。

```
# ソースのビルドは普通のC#のビルドと同じ
# ソリューションファイルが src フォルダにあるので dotnet build src コマンドを実施
dotnet build src
```

```

posh-git ~ cdksample [master]
C:\temp\cdksample [master +0 ~1 -0 !]> dotnet build src
.NET Core 向け Microsoft (R) Build Engine バージョン 16.1.176+g14b0a930a7
Copyright (C) Microsoft Corporation. All rights reserved.

C:\temp\cdksample\src\CdkSample\CdkSample.csproj の復元が 87.3 ms で完了しました。
CdkSample -> C:\temp\cdksample\src\CdkSample\bin\Debug\netcoreapp2.1\CdkSample.dll

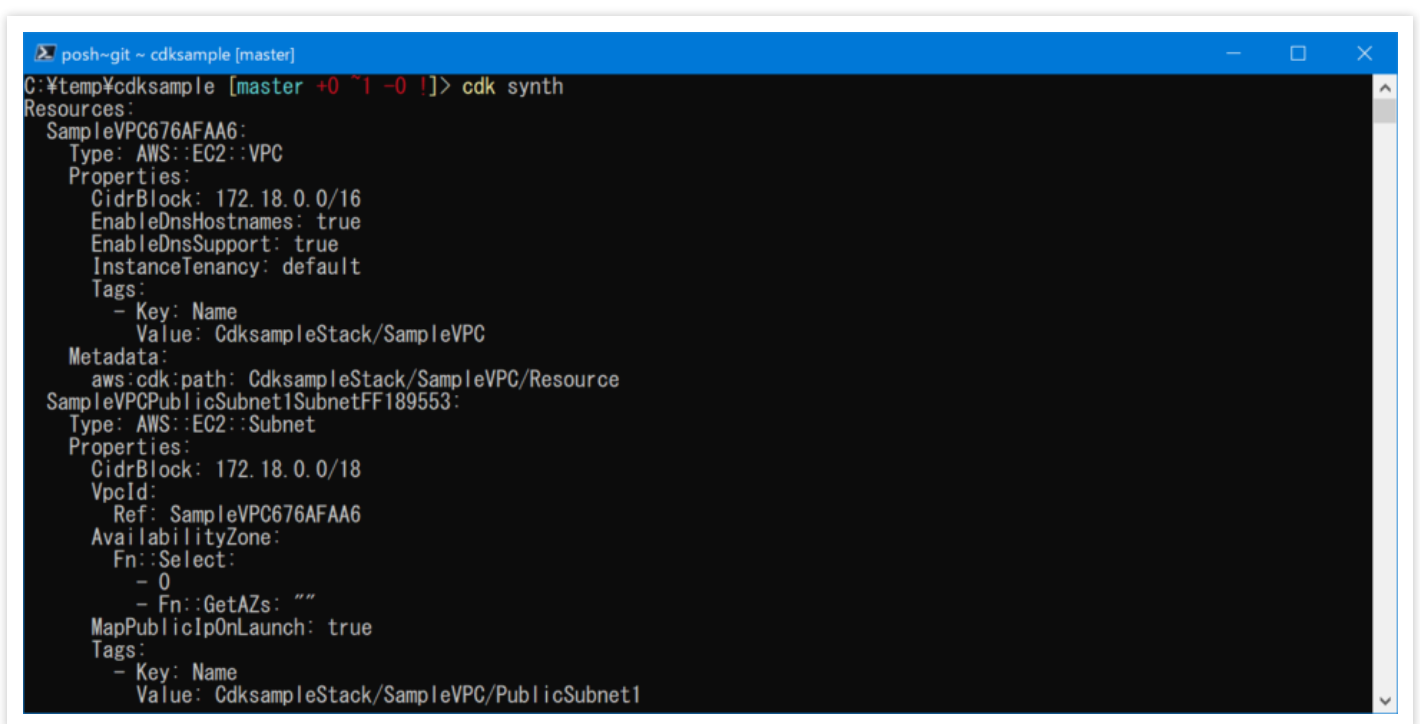
ビルドに成功しました。
0 個の警告
0 エラー

経過時間 00:00:04.66
C:\temp\cdksample [master +0 ~1 -0 !]>

```

エラー無くビルドが完了すればOKです。続けて、 `cdk synth` コマンドを実行することでC#のアセンブリからCloudFormation Templateを生成します。

```
# cdk synthコマンドでCloudFormation Templateを生成
cdk synth
```



```

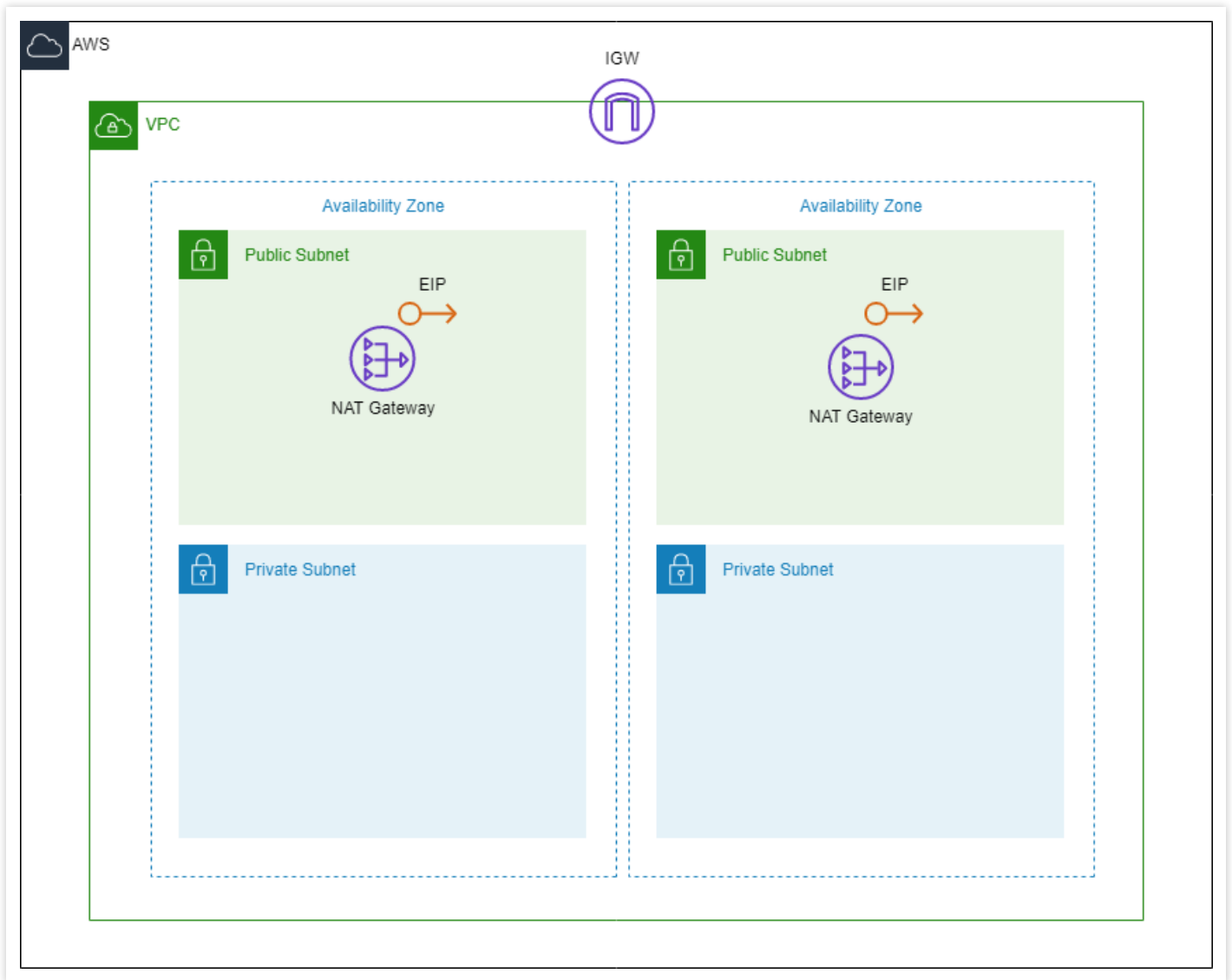
posh-git ~ cdksample [master]
C:\temp\cdksample [master +0 ~1 -0 !]> cdk synth
Resources:
  SampleVPC676AFAA6:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 172.18.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
      InstanceTenancy: default
      Tags:
        - Key: Name
          Value: CdkSampleStack/SampleVPC
    Metadata:
      aws:cdk:path: CdkSampleStack/SampleVPC/Resource
  SampleVPCPublicSubnet1SubnetFF189553:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 172.18.0.0/18
      VpcId:
        Ref: SampleVPC676AFAA6
      AvailabilityZone:
        Fn::Select:
          - 0
          - Fn::GetAZs: ""
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: CdkSampleStack/SampleVPC/PublicSubnet1

```

今回の例で出力されるテンプレートの全容は以下となります。

▶ テンプレート全体を表示

ここで余談なのですが、VPCの定義だけを明示した場合は暗黙的に2つのPublicサブネット、2つのPrivateサブネット、インターネットゲートウェイ(IGW)、NAT Gateway、ルートテーブルなどが生成されます。いわゆるベストプラクティスに沿った高可用性ネットワークをよしなに作ってくれる感じです。



(上記テンプレートのネットワーク構成図)

もちろんサブネット等のリソースを明示した場合は暗黙で作成されるものが減りますが、例えば Publicサブネットを記述した場合はIGWをセットで、Privateサブネットを記述した場合はNAT Gatewayをセットで生成するなど、ネットワーク構成をできる限り意識させないつくりになっている様です。

デプロイ

生成されるCloudFormation Templateに問題が無さそうであれば、 **cdk deploy** コマンドを使い実環境に反映させます。 **--profile** パラメーターで認証に使用するプロファイルを指定します。

```
# CloudFormationスタックを作成し実環境に反映
cdk deploy --profile [your-profile]
```

```

posh-git ~ cdksample [master]
C:\temp\cdksample [master +0 ~1 -0 !]> cdk deploy --profile test-profile
CdkSampleStack: deploying...
CdkSampleStack: creating CloudFormation changeset...
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::EC2::InternetGateway | SampleVPC/IGW (SampleVPCIGW11D9AA06)
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::EC2::VPC | SampleVPC (SampleVPC676AFAA6)
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::EC2::EIP | SampleVPC/PublicSubnet1/EIP (SampleVP
CPublicSubnet1EIP7C23471C)
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::EC2::EIP | SampleVPC/PublicSubnet2/EIP (SampleVP
CPublicSubnet2EIP80228F3)
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::CDK::Metadata | CDKMetadata
0/25 | 16:48:40 | CREATE_IN_PROGRESS | AWS::EC2::InternetGateway | SampleVPC/IGW (SampleVPCIGW11D9AA06)
Resource creation Initiated
0/25 | 16:48:41 | CREATE_IN_PROGRESS | AWS::EC2::VPC | SampleVPC (SampleVPC676AFAA6) Resource
e creation Initiated
0/25 | 16:48:41 | CREATE_IN_PROGRESS | AWS::EC2::EIP | SampleVPC/PublicSubnet2/EIP (SampleVP
CPublicSubnet2EIP80228F3) Resource creation Initiated
0/25 | 16:48:41 | CREATE_IN_PROGRESS | AWS::EC2::EIP | SampleVPC/PublicSubnet1/EIP (SampleVP
CPublicSubnet1EIP7C23471C) Resource creation Initiated
0/25 | 16:48:42 | CREATE_IN_PROGRESS | AWS::CDK::Metadata | CDKMetadata Resource creation Initiat
ed
1/25 | 16:48:42 | CREATE_COMPLETE | AWS::CDK::Metadata | CDKMetadata
2/25 | 16:48:56 | CREATE_COMPLETE | AWS::EC2::EIP | SampleVPC/PublicSubnet1/EIP (SampleVP
CPublicSubnet1EIP7C23471C)
3/25 | 16:48:56 | CREATE_COMPLETE | AWS::EC2::EIP | SampleVPC/PublicSubnet2/EIP (SampleVP
CPublicSubnet2EIP80228F3)
4/25 | 16:48:56 | CREATE_COMPLETE | AWS::EC2::InternetGateway | SampleVPC/IGW (SampleVPCIGW11D9AA06)
5/25 | 16:48:57 | CREATE_COMPLETE | AWS::EC2::VPC | SampleVPC (SampleVPC676AFAA6)
5/25 | 16:48:59 | CREATE_IN_PROGRESS | AWS::EC2::RouteTable | SampleVPC/PrivateSubnet2/RouteTable (

```

```

posh-git ~ cdksample [master]
19/25 | 16:49:35 | CREATE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PublicSubnet1/RouteTableAss
ociation (SampleVPCPublicSubnet1RouteTableAssociation0E1E38CA)
20/25 | 16:49:35 | CREATE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PrivateSubnet1/RouteTableAs
sociation (SampleVPCPrivateSubnet1RouteTableAssociation42FCDD78)
20/25 Currently in progress: SampleVPCPublicSubnet1NATGatewayB71B54D1, SampleVPCPublicSubnet2NATGateway2775C8CB
21/25 | 16:51:22 | CREATE_COMPLETE | AWS::EC2::NatGateway | SampleVPC/PublicSubnet1/NATGateway (S
ampleVPCPublicSubnet1NATGatewayB71B54D1)
22/25 | 16:51:23 | CREATE_COMPLETE | AWS::EC2::NatGateway | SampleVPC/PublicSubnet2/NATGateway (S
ampleVPCPublicSubnet2NATGateway2775C8CB)
22/25 | 16:51:24 | CREATE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet1/DefaultRoute
(SampleVPCPrivateSubnet1DefaultRoute8F7F8183)
22/25 | 16:51:25 | CREATE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet1/DefaultRoute
(SampleVPCPrivateSubnet1DefaultRoute8F7F8183) Resource creation Initiated
22/25 | 16:51:26 | CREATE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet2/DefaultRoute
(SampleVPCPrivateSubnet2DefaultRouteE89BC1AA)
22/25 | 16:51:27 | CREATE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet2/DefaultRoute
(SampleVPCPrivateSubnet2DefaultRouteE89BC1AA) Resource creation Initiated
23/25 | 16:51:41 | CREATE_COMPLETE | AWS::EC2::Route | SampleVPC/PrivateSubnet1/DefaultRoute
(SampleVPCPrivateSubnet1DefaultRoute8F7F8183)
24/25 | 16:51:42 | CREATE_COMPLETE | AWS::EC2::Route | SampleVPC/PrivateSubnet2/DefaultRoute
(SampleVPCPrivateSubnet2DefaultRouteE89BC1AA)
25/25 | 16:51:44 | CREATE_COMPLETE | AWS::CloudFormation::Stack | CdkSampleStack

✔ CdkSampleStack

Stack ARN:
arn:aws:cloudformation:ap-northeast-1:~:stack/CdkSampleStack/3c0d7080-c4b1-1~
C:\temp\cdksample [master +0 ~1 -0 !]>

```

(進捗状況の表示がずれてますが処理に問題はありません)

エラー無くコマンドが完了すればOKです。 マネジメントコンソールを確認すると以下の様に CloudFormation Stackが作成され、各種リソースが生成されています。

CloudFormation > スタック: CdksampleStack

CdksampleStack [削除する] [更新する] [スタックアクション] [スタックの作成]

スタックの情報 イベント リソース 出力 パラメータ テンプレート 変更セット

概要

スタック ID	説明
arn:aws:cloudformation:ap-northeast-1:632000000000:stack/CdksampleStack/3c0d7080-c4b1-11eb-b800-0a0b861f0000	-
ステータス	状況の理由
CREATE_COMPLETE	-
ルースタック	親スタック
-	-
作成時刻	削除時刻
2019-08-22 16:48:30 UTC+0900	-
更新時刻	-
2019-08-22 16:48:36 UTC+0900	-
ドリフトステータス	前回のドリフトチェック時刻
NOT_CHECKED	-
削除保護	IAM ロール
無効	-

VPC の作成 [アクション]

search: cdk フィルターの追加

Name	VPC ID	状態	IPv4 CIDR	IPv6 CIDR	DHCP オプションセット	メインルートテーブル
CdksampleStack/Sa...	vpc-0a1e4c...	available	172.18.0.0/16	-	dopt-...	rtb-0396dd...

サブネットの作成 [アクション]

search: cdk フィルターの追加

Name	サブネット ID	状態	VPC	IPv4 CIDR	利用可能な IPv4	IPv6 CIDR
CdksampleStack/SampleVPC/PrivateSubnet1	subnet-0b0e82...	available	vpc-0a1e4c...	172.18.128.0/18	16379	-
CdksampleStack/SampleVPC/PrivateSubnet2	subnet-0954e96...	available	vpc-0a1e4c...	172.18.192.0/18	16379	-
CdksampleStack/SampleVPC/PublicSubnet1	subnet-096f506...	available	vpc-0a1e4c...	172.18.0.0/18	16378	-
CdksampleStack/SampleVPC/PublicSubnet2	subnet-0c6b539...	available	vpc-0a1e4c...	172.18.64.0/18	16378	-

ルートテーブルの作成 [アクション]

search: cdk フィルターの追加

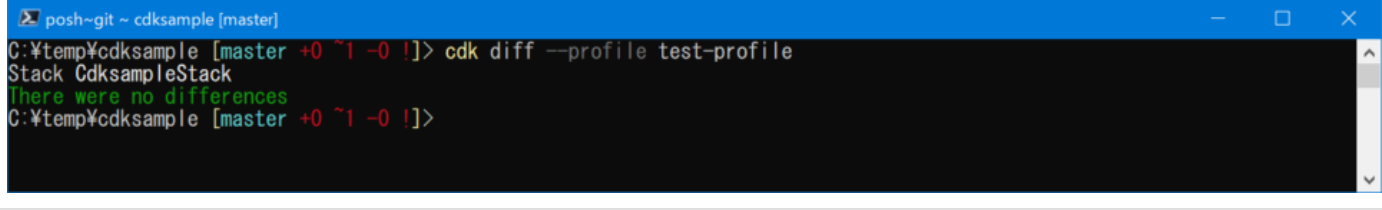
Name	ルートテーブル ID	明示的に関連付けられた	メイン	VPC ID
	rtb-0396dd...	-	はい	vpc-0a1e4c4e...
CdksampleStack/SampleVPC/PrivateSubnet1	rtb-033508...	subnet-0b0e825...	いいえ	vpc-0a1e4c4e...
CdksampleStack/SampleVPC/PrivateSubnet2	rtb-0c5310...	subnet-0954e96...	いいえ	vpc-0a1e4c4e...
CdksampleStack/SampleVPC/PublicSubnet1	rtb-0eeae9...	subnet-096f506...	いいえ	vpc-0a1e4c4e...
CdksampleStack/SampleVPC/PublicSubnet2	rtb-0f9dccb...	subnet-0c6b539...	いいえ	vpc-0a1e4c4e...

(キリがないのでこのくらいで...)

その他コマンド

この他に `cdk diff` コマンドでデプロイされた環境とローカル環境の差分を取得できます。

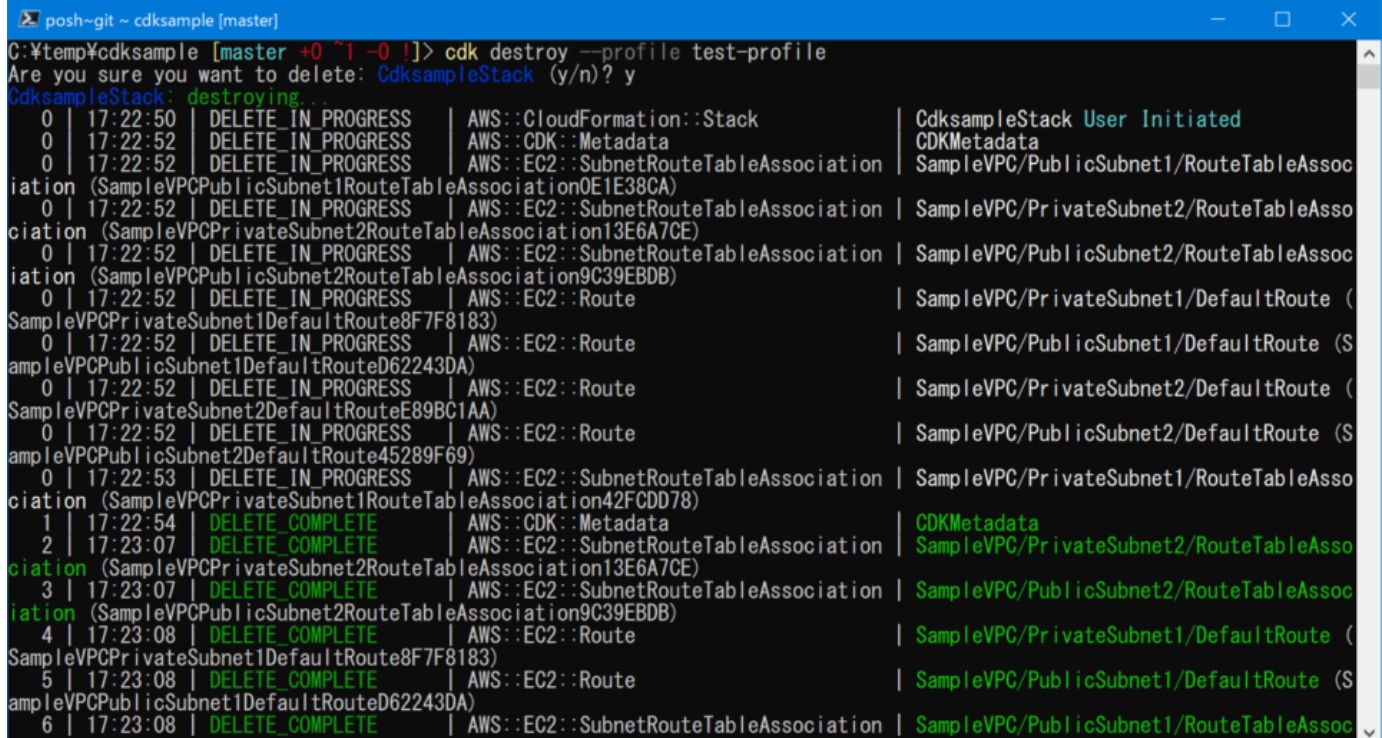
```
# デプロイされた環境とローカル環境の差分を取得
cdk diff --profile [your-profile]
```



```
posh-git ~ cdksample [master]
C:\temp\cdksample [master +0 ~1 -0 !]> cdk diff --profile test-profile
Stack CdksampleStack
There were no differences
C:\temp\cdksample [master +0 ~1 -0 !]>
```

`cdk destroy` コマンドでCloudFormationスタックの削除を行うことができます。

```
# CloudFormationスタックの削除
cdk destroy --profile [your-profile]
```



```
posh-git ~ cdksample [master]
C:\temp\cdksample [master +0 ~1 -0 !]> cdk destroy --profile test-profile
Are you sure you want to delete: CdksampleStack (y/n)? y
CdksampleStack: destroying...
0 | 17:22:50 | DELETE_IN_PROGRESS | AWS::CloudFormation::Stack | CdksampleStack User Initiated
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::CDK::Metadata | CDKMetadata
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PublicSubnet1/RouteTableAssoc
iation (SampleVPCPublicSubnet1RouteTableAssociation0E1E38CA)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PrivateSubnet2/RouteTableAsso
ciation (SampleVPCPrivateSubnet2RouteTableAssociation13E6A7CE)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PublicSubnet2/RouteTableAssoc
iation (SampleVPCPublicSubnet2RouteTableAssociation9C39EBDB)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet1/DefaultRoute (
SampleVPCPrivateSubnet1DefaultRoute8F7F8183)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PublicSubnet1/DefaultRoute (S
ampleVPCPublicSubnet1DefaultRouteD62243DA)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PrivateSubnet2/DefaultRoute (
SampleVPCPrivateSubnet2DefaultRouteE89BC1AA)
0 | 17:22:52 | DELETE_IN_PROGRESS | AWS::EC2::Route | SampleVPC/PublicSubnet2/DefaultRoute (S
ampleVPCPublicSubnet2DefaultRoute45289F69)
0 | 17:22:53 | DELETE_IN_PROGRESS | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PrivateSubnet1/RouteTableAsso
ciation (SampleVPCPrivateSubnet1RouteTableAssociation42FCDD78)
1 | 17:22:54 | DELETE_COMPLETE | AWS::CDK::Metadata | CDKMetadata
2 | 17:23:07 | DELETE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PrivateSubnet2/RouteTableAsso
ciation (SampleVPCPrivateSubnet2RouteTableAssociation13E6A7CE)
3 | 17:23:07 | DELETE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PublicSubnet2/RouteTableAssoc
iation (SampleVPCPublicSubnet2RouteTableAssociation9C39EBDB)
4 | 17:23:08 | DELETE_COMPLETE | AWS::EC2::Route | SampleVPC/PrivateSubnet1/DefaultRoute (
SampleVPCPrivateSubnet1DefaultRoute8F7F8183)
5 | 17:23:08 | DELETE_COMPLETE | AWS::EC2::Route | SampleVPC/PublicSubnet1/DefaultRoute (S
ampleVPCPublicSubnet1DefaultRouteD62243DA)
6 | 17:23:08 | DELETE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | SampleVPC/PublicSubnet1/RouteTableAssoc
```