**AWS CDK** is an open-source software development framework that allows you to define your cloud application resources using familiar programming languages, like JavaScript, TypeScript, Python, Java, and C#. The code you write gets transpiled into CloudFormation (CFN) templates, and creates the infrastructure using AWS CloudFormation.

**Prerequisites:**

**Install and configure AWS CLI**

For AWS CDK to interact with your AWS Account, it needs credentials and the AWS CLI installed and configured. If you haven't done so, please follow the Setting Up Your AWS Environment guide to set up the CLI on your workstation. When you're setting up your AWS CLI, remember to use an AWS account with administrator-level access.

**Choose and install an IDE**

One of the benefits of AWS CDK is that you can use your favourite development environment and have a rich experience when exploring the hundreds of different services and features of AWS. We highly recommend you use an IDE that supports code-completion and syntax highlighting for your language of choice:

* Visual Studio Code (VSCode)
* AWS Cloud9
* Atom
* vim
* emacs
* WebStorm
* PyCharm

**Check your language framework**

The AWS CDK has first-class support for TypeScript, JavaScript, Python, Java, C#, and Go. Other JVM and .NET CLR languages may also be used, at least in theory, but we are unable to offer support for them at this time.

All AWS CDK developers, even those working in Python, Java, or C#, need Node.js 10.13.0 or later. All supported languages use the same backend, which runs on Node.js.

**Working with the AWS CDK in TypeScript:**

<https://docs.aws.amazon.com/cdk/v2/guide/work-with-cdk-typescript.html>

**Working with the AWS CDK in JavaScript:**

<https://docs.aws.amazon.com/cdk/v2/guide/work-with-cdk-javascript.html>

**Working with the AWS CDK in Python:**

<https://docs.aws.amazon.com/cdk/v2/guide/work-with-cdk-python.html>

**Working with the AWS CDK in Java:**

<https://docs.aws.amazon.com/cdk/v2/guide/work-with-cdk-java.html>

**Working with the AWS CDK in C#:**

<https://docs.aws.amazon.com/cdk/v2/guide/work-with-cdk-csharp.html>

**1. Install AWS CDK**

Use the NPM package manager in your terminal to install AWS CDK and TypeScript globally on your system:

**npm install -g aws-cdk typescript**

Once you’ve installed AWS CDK you can validate that you’re running on the latest version by running the following command in the terminal:

**cdk –version**

**2. Install AWS CLI and configure an AWS profile**

The AWS CLI is a command line tool that allows you to interact with AWS services in your terminal. Depending on if you’re running Linux, macOS, or Windows the installation goes like this:

# macOS install method:

**brew install awscli**

# Windows install method:

**wget https://awscli.amazonaws.com/AWSCLIV2.msi**

**msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi**

# Linux (Ubuntu) install method:

**sudo apt install awscli**

In order to access your AWS account with the AWS CLI, you first need to configure an AWS Profile. There are 2 ways of configuring a profile:

* **Access and secret key credentials from an IAM user**
* **AWS Single Sign-on (SSO) user**

**Then configure the AWS profile on the AWS CLI as follows:**

**➜ aws configure**

AWS Access Key ID [None]: <insert\_access\_key>

AWS Secret Access Key [None]: <insert\_secret\_key>

Default region name [None]: <insert\_aws\_region>

Default output format [json]: json

Your was credentials are storen in ~/.aws/credentials and as you can validate that that you AWS profile is working by running the command:

**➜ aws sts get-caller-identity**

{

"UserId": "AIDA5BRFSNF24CDMD7FNY",

"Account": "012345678901",

"Arn": "arn:aws:iam::012345678901:user/test-user"

}

**3. Create a new AWS CDK TypeScript Project**

Now that we’ve configured our profile and installed the packages, it’s time to create an AWS CDK TypeScript project where you’re going to build the Amazon S3 Bucket construct.

You can generate a new AWS CDK TypeScript project by running the following command in an empty directory:

**➜ cdk init sample-app --language=typescript**

**4. Create an Amazon S3 Bucket construct in AWS CDK**

import \* as cdk from 'aws-cdk-lib';

import \* as iam from 'aws-cdk-lib/aws-iam';

import \* as kms from 'aws-cdk-lib/aws-kms';

import \* as s3 from 'aws-cdk-lib/aws-s3';

import { Construct } from 'constructs';

export class S3BucketStack extends cdk.Stack {

constructor(scope: Construct, id: string, props?: cdk.StackProps) {

super(scope, id, props);

const s3Bucket = new s3.Bucket(this, 'exampleBucket', {

objectOwnership: s3.ObjectOwnership.BUCKET\_OWNER\_ENFORCED,

blockPublicAccess: s3.BlockPublicAccess.BLOCK\_ALL,

encryptionKey: new kms.Key(this, 's3BucketKMSKey'),

});

s3Bucket.grantRead(new iam.AccountRootPrincipal());

}

}

const app = new cdk.App();

new S3BucketStack(app, 'S3BucketStack');

app.synth();

**5. Synthesize your Amazon S3 Bucket in AWS CDK**

The Amazon S3 Bucket construct has been created in a stack. Now you can generate the CloudFormation template by running AWS CDK Synthesize:

**➜ cdk synth**

**6. Deploy your Amazon S3 Bucket to AWS Cloud using AWS CDK**

To deploy the S3 bucket to your AWS account, run the deploy command:

**➜ cdk deploy**

**7. Destroy Your Amazon S3 Bucket Using AWS CDK**

To clean up the AWS resources that were created for this project, run the destroy command:

**➜ cdk destroy**