######################################################################

**1. create a simple AWS CDK app in TypeScript that deploys an S3 bucket.**

######################################################################

Prerequisites:

1. AWS Account: You'll need an AWS account to deploy resources.
2. Node.js and npm: AWS CDK requires Node.js and npm.
3. AWS CLI: You'll need the AWS CLI to configure credentials and bootstrap your environment.
4. TypeScript: Ensure you have TypeScript installed globally.

**Step1: install aws cli in your local machine**

Verify the installation: aws --version

**Step 2: Create IAM User with CLI Access**

1. Log in to the AWS Management Console.

2. Navigate to the IAM (Identity and Access Management) service.

3. Create a new user:

Click on Users > Add user.

Enter a username (e.g., aws-cli-user).

Select Programmatic access.

4.Set permissions:

Attach existing policies directly.

For testing purposes, you can select the AdministratorAccess policy, but in a production environment, you should follow the principle of least privilege.

Complete the process and download the .csv file containing the Access Key ID and Secret Access Key.

**Step 3: Configure AWS CLI**

1. Open a terminal or command prompt.

2. Run the aws configure command: aws configure

Sample Data Demo

C:\Users\psasirek>aws configure

AWS Access Key ID [None]: AKIA6ODU6OMLYQMRIEX5

AWS Secret Access Key [None]: VR+JcCtsWxyVwevA6tdXjIb+KJGYVPFfcfUr6dSj

Default region name [None]: us-east-1

Default output format [None]: json

3. Enter the following information when prompted:

AWS Access Key ID: Enter the Access Key ID from the .csv file.

AWS Secret Access Key: Enter the Secret Access Key from the .csv file.

Default region name: Enter your preferred region (e.g., us-west-2).

Default output format: Enter your preferred output format (e.g., json).

Step 4: Verify Configuration

aws s3 ls

Steo 6:

Create a new CDK project in your local machine

mkdir my-cdk-project

cd my-cdk-project

cdk init app --language typescript

Install dependencies.

npm install

Add the S3 construct.

==============================================================

Open lib/my-cdk-project-stack.ts and add the following code:

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

import { Construct } from 'constructs';

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

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

export class MyprojectStack extends cdk.Stack {

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

super(scope, id, props);

new s3.Bucket(this, 'MyFirstBucket', {

versioned: true

});

}

}

Step 7. Bootstrap your AWS environment.

cdk bootstrap

Step 8: Deploy your stack.

cdk deploy

Explanation:

**cdk init:** Creates a new CDK project with the necessary files and structure.

**npm install**: Installs the required dependencies, including the AWS CDK libraries.

**aws-cdk-lib/aws-s3**: Imports the S3 construct library.

**new s3.Bucket**: Creates a new S3 bucket with versioning enabled.

**cdk bootstrap**: Initializes your AWS environment for CDK deployments.

**cdk deploy**: Deploys your stack to AWS, creating the S3 bucket.

**Detailed Explanation:**

1. import \* as cdk from 'aws-cdk-lib': This imports the AWS CDK library, which provides the base classes and methods for defining infrastructure as code.

2. import { Construct } from 'constructs': This imports the Construct class, which is a building block of AWS CDK applications.

3. import \* as s3 from 'aws-cdk-lib/aws-s3': This imports the AWS CDK S3 library, which provides classes and methods for defining Amazon S3 resources.

4. export class MyprojectStack extends cdk.Stack {

export class MyprojectStack: This defines a new class named MyprojectStack.

By exporting it, the class can be imported and used in other files.

extends cdk.Stack: This indicates that MyprojectStack is a subclass of cdk.Stack. In AWS CDK, a stack is a unit of deployment, which can contain multiple AWS resources.

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

This is the constructor for the MyprojectStack class.

It takes three parameters:

**scope:** The parent construct. In most cases, this is the application or another construct that this stack is part of.

**id:** A unique identifier for this stack.

**props**: Optional properties for the stack, which can include settings like environment and stack name.

super(scope, id, props): This calls the constructor of the parent class (cdk.Stack) with the provided arguments.

4. Define an S3 Bucket

new s3.Bucket(this, 'MyFirstBucket', {

versioned: true

});