AWS CDK using Typescript [take-away]

- Exploring CDK lifecycle
- Troubleshooting with cloudformation stacks
- Auto deploy code with github actions
- Solving known issues with official github aws-cdk
- Exploring s3 , ec2 , Iam , VPC etc in aws using CDK
- 4 Projects

Day 1: TypeScript and CDK Basics

Session 1: TypeScript Foundations

- Why TypeScript for AWS CDK?
- Setting up the environment for TypeScript.
- TypeScript essentials:
  - Types, interfaces, and classes.
  - Modules and namespaces.
  - Async/await and error handling.
- Hands-on exercises:
  - Write a simple TypeScript program.
  - Create reusable modules.

Session 2: AWS CDK Introduction

- Overview of Infrastructure as Code (IaC).
- CDK architecture and core concepts:
  - Stacks, constructs, and apps.
- Hands-on:
  - Set up your first CDK project using TypeScript.
  - Deploy a basic S3 bucket.

Session 3: CDK CLI Commands and Workflow

- Exploring CDK commands:
  - `cdk init`, `cdk diff`, `cdk deploy`, `cdk synth`.
- Understanding CDK deployment process.
- Hands-on:
  - Practice the CDK command lifecycle with simple projects.

Day 2: AWS CDK with GitHub Integration

Session 1: Working with GitHub Repositories

- Introduction to GitHub basics:
  - Creating and managing repositories.
  - Version control with Git for CDK projects.
- Hands-on:
  - Push a CDK project to a GitHub repository.

Session 2: CI/CD with GitHub Actions

- Writing GitHub Actions workflows for CDK:
  - Auto-deploying CDK stacks.
  - Managing environment-specific deployments.
- Hands-on:
  - Create and test a simple GitHub Actions workflow for CDK.

Session 3: Building Real-World CDK Stacks

- Advanced constructs for AWS services:
  - S3 with lifecycle policies.
  - IAM roles and policies.
  - EC2 with custom VPC configurations.
- Hands-on:
  - Build and deploy a stack with multiple AWS services.

Day 3: Advanced CDK Features and CloudWatch

Session 1: Scaling with CDK Constructs

- Reusable custom constructs.
- Using environment variables and secrets.
- Multi-region and multi-account setups.
- Hands-on:
  - Build reusable constructs for shared configurations.

Session 2: Debugging with CloudFormation and CloudWatch

- Synthesizing CloudFormation templates from CDK.
- Troubleshooting with CloudFormation logs.
- Monitoring deployments with CloudWatch.
- Hands-on:
  - Investigate and fix deployment issues using CloudFormation and CloudWatch logs.

Session 3: Advanced CDK Commands and Customization

- Advanced CDK commands: `cdk destroy`, `cdk bootstrap`.
- CDK aspects and metadata.
- Hands-on:
  - Customize a deployment with metadata and advanced CLI options.

Day 4: Deploying AWS Services with CDK

Session 1: Deep Dive into AWS Services

- S3: Versioning, encryption, and lifecycle policies.
- EC2: Instance types, VPCs, and security groups.
- IAM: Custom roles and policies.
- Hands-on:
  - Deploy a stack integrating S3, EC2, and IAM.

Session 2: Cost and Performance Optimization

- Optimizing stacks for cost and efficiency.
- Using AWS pricing calculators.
- Hands-on:
  - Analyze and optimize a high-cost stack.

Session 3: Multi-Service Applications

- Deploying complex applications with multiple services.
- Best practices for scaling and performance.
- Hands-on:
  - Deploy an application stack integrating S3, EC2, VPC, and Lambda.

Day 5: Advanced Scenarios and Troubleshooting

Session 1: Advanced Features and Automation

- Using CDK Pipelines for CI/CD.
- Integrating CDK with Secrets Manager and Parameter Store.
- Automating deployments with `cdk deploy` options.
- Hands-on:
  - Build and deploy a pipeline for a multi-service application.

Session 2: Debugging and Troubleshooting

- Debugging TypeScript issues in CDK.
- Common deployment errors and solutions.
- Using GenAI for troubleshooting.
- Hands-on:
    - Resolve a real-world deployment issue.

Session 3: Final Project and Wrap-Up

- Final project:
    - Build and deploy a scalable web application stack.
- Recap of key concepts.
- Q&A and additional resources.

Projects:

Project 1: Event-Driven Serverless Application with S3 and Lambda

Objective: Create a serverless application to process and store files uploaded to an S3 bucket.

Steps:

1. Use AWS CDK to create the following resources:
    - An S3 bucket with versioning and lifecycle rules.
    - A Lambda function triggered by S3 object creation events.
    - A DynamoDB table to store metadata of processed files.
2. Write a Lambda function in TypeScript to:
    - Read file metadata from the S3 event.
    - Store metadata in the DynamoDB table.
3. Test the application by uploading sample files and verifying the metadata in DynamoDB.

Key Features:

- Real-world use of Lambda and S3 integration.
- Hands-on experience with CDK constructs for S3, Lambda, and DynamoDB.

Project 2: Automated Alerts for EC2 Instances

Objective: Set up a monitoring and alerting system for EC2 instances using CloudWatch and SNS.

Steps:

1. Use AWS CDK to create:
    - EC2 instances in a custom VPC.
    - CloudWatch alarms to monitor CPU utilization.
    - An SNS topic for sending email notifications.
2. Write a Lambda function (optional) to automate scaling based on CPU usage.
3. Configure email subscriptions to receive alerts.
4. Deploy and test the setup by simulating high CPU usage.

Key Features:

- Practical implementation of CloudWatch for monitoring.
- Alerting using SNS.
- Hands-on with EC2 and VPC configurations.

Project 3: CI/CD Pipeline for Lambda Deployment with GitHub Actions

Objective: Automate the deployment of a Lambda function using GitHub Actions and CDK Pipelines.

Steps:

1. Set up a GitHub repository for the project.
2. Create a Lambda function in TypeScript and deploy it using CDK.
3. Write a GitHub Actions workflow to:
    - Lint and test the Lambda code.
    - Automatically deploy changes to AWS when a PR is merged.
4. Use CDK Pipelines for multi-environment deployment (e.g., dev and prod).

Key Features:

- End-to-end CI/CD pipeline using GitHub Actions.
- Real-world application of CDK Pipelines.
- Automation of Lambda deployments.

Project 4: Scalable Web Application with Monitoring and Alerts

Objective: Deploy a scalable web application with EC2, ALB, and CloudWatch monitoring.

Steps:

1. Use AWS CDK to create:
    - A custom VPC with subnets.
    - An Auto Scaling Group for EC2 instances.
    - An Application Load Balancer (ALB).
2. Deploy a simple web application (e.g., a Node.js or Python app) on the EC2 instances.
3. Set up CloudWatch logs and alarms to monitor:
    - Application errors.
    - ALB response times.
4. Create SNS alerts for critical issues.
5. (Optional) Use GitHub Actions for infrastructure updates.

Key Features:

- Real-world deployment of a multi-service web application.
- Scaling with EC2 and ALB.
- Monitoring and troubleshooting with CloudWatch.