**AWS CDK Primer Resume**

**Module 1: AWS CDK Intro**

**Summary:** This part simply describes DevOps as a methodology on how groups can better manage their software suites to deliver CI more efficiently compared to traditional methods. We set up the AWS CLI and briefly describe why to use this setup as opposed to several alternatives. We learn the intricacies of the Cloud Development Kit, how the components interact with each other and the benefits of a singular cloud-based infrastructure. We put a capstone onto the project by deploying a quick sample PHP app using AWS CloudFormation.

**Core Technologies:** AWS CLI, AWS CDK,AWS CloudFormation, Amazon Elastic Container Services (ECS), AWS Fargate

**Module 2: AWS CDK Core Components**

**Summary:** We get a more intimate look at constructs, which are the core components of CDK applications at the smallest scale. Constructs contain the code that AWS CloudFormation uses to create a component, then creates Stacks from Constructs. We learn about the CDK Construct hierarchy within constructs. We learn how to better appropriate the AWS CDK in order to synchronize the workflow we normally use onto the CloudFormation configuration.

**Core Technologies:** AWS CDK, NPM, Node, AWS Construct Library, AWS CDK App Constructs, nested Stacks

**Module 3: AWS CDK Concepts**

**Summary:** Identifiers are used to keep the scope of Constructs properly segmented and for CloudFormation to properly reference and distribute Constructs. We learn about the CDK Construct hierarchy within constructs. We learn how to better appropriate the AWS CDK in order to synchronize the workflow we normally use onto the CloudFormation configuration. We learn how to better appropriate the AWS CDK in order to synchronize the workflow we normally use onto the CloudFormation configuration. Environments are a way to define the area where our stack operates in. Contexts are specialized data pairs intended to be accessed through specific child components or constructs but not the entire stack.Finally, Assets are the actual images and documents that are integrated into the CDK for use on the stack or app.

**Core Technologies:** AWS Construct Library, AWS CDK App Constructs, nested Stacks, AWS CDK App Constructs, nested Stacks

**Module 4: AWS CDK Toolkit**

**Summary:** We can use the Toolkit as a more convenient way to input into our applications. With a variety of commands and inputs, it bears similarities to a terminal based approach. The Toolkit is designed to funnel into AWS CloudFormation. Additionally, we learn how to configure our IDE to utilize the Toolkit for use in AWS CloudFormation. Testing your code is also done through the CDK.

**Core Technologies:** AWS CDK Toolkit, TypeScript, Node.js AWS Libraries, AWS CloudFormation, CDK Bootstrap, AWS Solutions Constructs, Jest

**Module 5: AWS CDK Development**

**Summary:** We learn how to configure our IDE to utilize the Toolkit for use in AWS CloudFormation. Testing your code is also done through the CDK. We conclude this section with a brief discussion on the best practices involved on creating software through the AWS CDK.

**Core Technologies:** AWS CloudFormation, CDK Bootstrap, AWS Solutions Constructs, Jest

**Module 6: Troubleshooting**

**Summary:** This module discusses the core issue of troubleshooting, especially in cases of code inexplicably breaking between updates. We mediate between different package version dependencies.It also covers fixing some of the more common issues users of AWS can run into like synth error or deployment destruction errors.

**Core Technologies:** AWS Construct Library, NPM