## Lab 6-03: Deploy a New Web API Application to the Lambda Service Using an AWS Template

### Service Introduction

AWS Lambda is a serverless computing service that lets you run code in response to events without provisioning or managing servers. AWS CloudFormation allows you to use templates to model and set up your AWS resources, so you can spend less time managing those resources and more time focusing on your applications.

### Problem

Your team is developing a new web API application and needs a quick, scalable, and cost-effective way to deploy it. Manually managing the underlying infrastructure is time-consuming and prone to errors, hindering development.

### Solution

Use an AWS CloudFormation template to deploy your web API application to AWS Lambda. The CloudFormation template will define all the necessary resources, such as API Gateway, Lambda functions, IAM roles, and other dependencies. This approach ensures a consistent and repeatable deployment process, allowing you to focus on developing your application while AWS handles the infrastructure setup and scaling.

#### Task 1: Configure Access Keys for the cloud\_user Account

|  |
| --- |
| 1. In the search bar at the top of the AWS console, enter **IAM** to search for the IAM service. 2. From the results, choose **IAM.**      1. On the left navigation menu, choose **Users.** 2. Click on cloud\_user.      1. Click the **Security credentials** tab and go to **Manage console access.**      1. Scroll down to the Access keys. 2. Click the **Create access key.**      1. From the available use cases, select **Other** and click **Next.**      1. Click **Create access key.**      1. On the Retrieve access keys page, you will see the Access key and Secret access key. Click Show to reveal the Secret access key.      1. Copy both keys into a text file, as you will need them shortly. Alternatively, you can click **Download .csv file** to save them to your computer.      1. On your computer, run the following command to configure the AWS CLI with the access keys you just created:   aws configure --profile cloud\_user   1. When prompted, enter the Access key ID and Secret access key you noted in the earlier step. 2. For the Default region name, enter us-east-1. 3. For the Default output format, enter json.      1. You now have a profile called cloud\_user configured for the AWS CLI. You can use this profile to run AWS CLI commands using the credentials of the cloud\_user account. |

#### Task 2: Create an S3 Bucket for the CloudFormation Stack

|  |
| --- |
| **Note:** To deploy applications using the .NET Lambda tooling, create an S3 bucket to store the deployment artifacts. S3 bucket names must be unique across all AWS users. For this lab, use your account number as the prefix for the bucket name.   1. In the top right corner of the AWS console, click on your username (formatted as cloud\_user @ followed by a number) to view your Account ID. It will be in the format of 123456789012.      1. Open a terminal window on your computer. 2. Use the following command to create the S3 bucket, replacing <ACCOUNT\_ID> with the account ID you got a moment ago):   aws s3api create-bucket --bucket <ACCOUNT\_ID>demobucket --profile cloud\_user   1. You will see output similar to:   {  "Location": "/XXXXXXXXdemobucket"  }     1. You now have the necessary components in AWS to deploy your application. |

#### Task 3: Create a .NET Web API Application Using an AWS Template

|  |
| --- |
| 1. Open **PowerShell** or another **console/terminal** window on your computer. 2. Create the new project using the following command:   dotnet new serverless.AspNetCoreWebAPI --name AspNetCoreWebAPI   1. Change to the new project directory using the following command:   cd AspNetCoreWebAPI/src/AspNetCoreWebAPI     1. You can open the project in your preferred IDE to review the code. |

#### Task 4: Deploy the Application to the AWS Lambda Service

|  |
| --- |
| 1. Now that you have created the application and the S3 bucket, you can deploy it to the Lambda service.      1. Deploy the project you created earlier using the following command:   dotnet lambda deploy-serverless --stack-name AspNetCoreWebAPI --s3-bucket XXXXXXXXdemobucket --profile cloud\_user  **Note:** This will take a few minutes to deploy. Wait until the deployment completes to continue to the next step.     1. When the deployment is complete, you will see output that looks like the following:   Output Name Value  --------------------------- --------------------------------------------------  ApiURL https://XXXXXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/   1. The application is deployed and ready to be invoked. |

#### Task 5: Invoke the Endpoints on the Application

|  |
| --- |
| 1. Copy the ApiURL from the output and paste it into a new browser tab. You should see a welcome message displayed.      1. Then, append /api/values to the end of the URL and open it. You should see a JSON response containing the values value1 and value2. |