Xerris Bootcamp Series

AWS Lambda Workshop

xerris

ACCOLITE DIGITAL

Transforming The Future, Now

Workshop Objectives

This workshop provides hands-on experience with the following:

- Install DotNet Core
- Install Amazon Lambda Tools & Templates for DotNet
- Introduction to the DotNet Core CLI
- Introduction to the AWS CLI
- Create and deploy an AWS Lambda
- Connect the AWS API Gateway to your Lambda

Install DotNet Core

- Ensure we have the latest version of the DotNet Core framework.

- From the terminal window or PowerShell console, enter:
  dotnet –version

Download/install the latest
https://dotnet.microsoft.com/download

Amazon Lambda Tools

Amazon has created AWS Lambda Tools for DotNet Core.

https://github.com/aws/aws-lambda-dotnet

Installing Amazon.Lamda.Tools

From the command line enter:

dotnet tool install -g Amazon.Lambda.Tools

Updating Amazon.Lamda.Tools

From the command line enter:

dotnet tool update -g Amazon.Lambda.Tools

Amazon Lambda Tools

Amazon has created AWS Lambda Templates for DotNet Core.

https://github.com/aws/aws-lambda-dotnet

Installing Amazon.Lamda.Templates
From the command line, enter:

dotnet new install **Amazon.Lambda.Templates**

After the installation is complete, enter the following:

dotnet new **lambda.EmptyFunction** -h

Amazon Lambda Templates

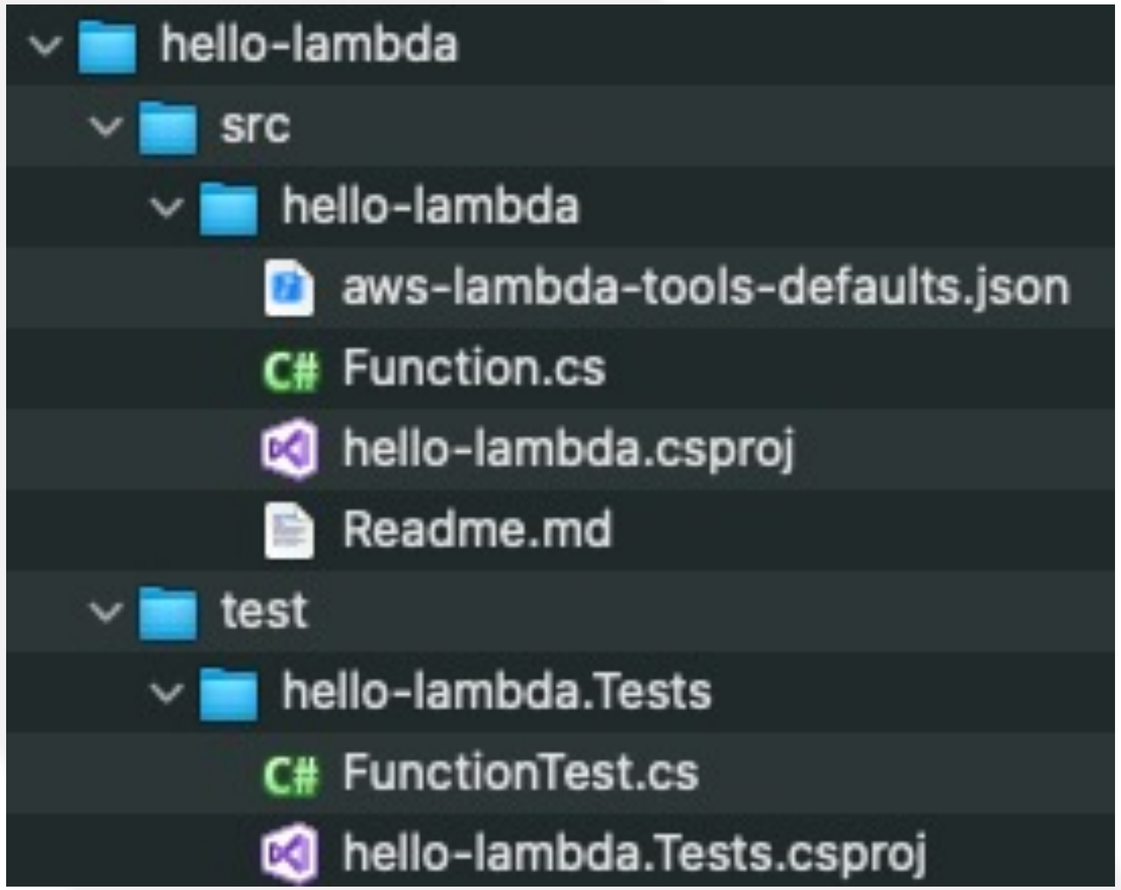
From the command line enter
dotnet lambda list

you will be shown a summary of this command

```
Project Home: https://github.com/aws/aws-extensions-for-dotnet-cli, https://github.com/aws/aws-lambda-dotnet



Commands to deploy and manage AWS Lambda functions:

        deploy-function         Command to deploy the project to AWS Lambda
        invoke-function         Command to invoke a function in Lambda with an optional input
        list-functions          Command to list all your Lambda functions
        delete-function         Command to delete a Lambda function
        get-function-config     Command to get the current runtime configuration for a Lambda function
        update-function-config  Command to update the runtime configuration for a Lambda function

Commands to deploy and manage AWS Serverless applications using AWS CloudFormation:

        deploy-serverless       Command to deploy an AWS Serverless application
        list-serverless         Command to list all your AWS Serverless applications
        delete-serverless       Command to delete an AWS Serverless application

Commands to publish and manage AWS Lambda Layers:

        publish-layer           Command to publish a Layer that can be associated with a Lambda function
        list-layers             Command to list Layers
        list-layer-versions     Command to list versions for a Layer
        get-layer-version       Command to get the details of a Layer version
        delete-layer-version    Command to delete a version of a Layer

Other Commands:

        package                 Command to package a Lambda project either into a zip file or docker image if
ith either deploy-function command or with another tool.
        package-ci              Command to use as part of a continuous integration system.
        push-image              Build Lambda Docker image and push the image to Amazon ECR.

To get help on individual commands execute:
        dotnet lambda help <command>
```

Creating a DotNet Lambda

- From the command line enter
  dotnet new lambda --list
- Scroll up and notice the Lambda templates available
- Create a Lambda using the **EmptyFunction** template
  dotnet new lambda**.EmptyFunction**
  –n hello-lambda-your-name



Creating a DotNet Solution

- In the root folder, create a DotNet Solution
  dotnet new sln
- Add the Lambda and Test DotNet Projects
- Create a folder called src
- Within the src folder, create a hello-lambda folder
- Create a class library project
  dotnet new classlib –f net6.0
- Within the src folder, create a hello-lambda-test folder
- Create a class library project
  dotnet new classlib –f net6.0

- Add these projects to your solution from your project root folder
  dotnet sln hello-lambda.sln add -s .
  src/hello-lambda/hello-lambda.csproj

  dotnet sln hello-lambda.sln add -s .
  src/hello-lambda-tests/hello-lambda-tests.csproj

Solution Explorer for VS Code

- In **VS Code,** go to the extensions tab and search for **dotnet solution.**
- Install the **vscode-solution-explorer** by **Fernando Escolar**.
- Locate the Solution Explorer icon in the menu bar and open it.
- Spend a bit of time navigating around the solution explorer and see what features it provides.
- Right-click on the **hello-lambda-tests** projects and run the tests.

Using the DotNet CLI

- From the command-line console, try to build the solution
  dotnet **build**

- Try running the tests
  dotnet **test**

- Play around with the Lambda code
- Update the test to it fails
- See the output from the command-line console

Deploying your Lambda

- Navigate to the src/hello-lambda

  dotnet lambda deploy-function hello-lambda –profile <profile-name>

- Choose to create a new role when promoted.
- Go to the AWS Management Console to the Lambda section.
- Find your newly deployed lambda and run it.

xerris

Connect your Lambda to the AWS API Gateway

- Log into the AWS Management Console
  - Select REST API -> BUILD
  - Choose NEW API
  - Provide a name and description
  - Action -> Create Method
    - Hook the POST method to your Lambda
  - Action -> Deploy API
    - Choose [New Stage] and name it **DEV**
  - Get the **URL** from the deployed Lambda and try it out in **Postman**.

xerris

LAB: Create your own AWS Lambda

- Objectives
  - Practical experience using the dotnet CLI
  - Practical experience using the Amazon DotNet Templates
  - Practical experience using the Amazon DotNet tools

  Create your own Lambda
  - Using the Amazon DotNet templates, create a basic Lambda project
  - Create a solution and add the projects to the solution.
  - Update your Lambda to return your name
  - Using Amazon DotNet Tools, deploy your Lambda

xerris