M346, C# Lambda-Function.

Thema:
Serverless - Lamda

Lernziele
- Sie erstellen eine C# Lambda-Function als IaC

Umgebung:
LP-22.04 / AWS Learner Lab

Aufgabe 1: Installation von Templates und Tools | Partnerarbeit

aws cli:
Überprüfen Sie, ob die aws cli in der Version >= 2.x.x installiert ist.

```
aws --version
```

Falls die aws cli noch nicht installiert ist, installieren sie sie gemäss Anleitung in M346

Wichtig!
Übernehmen Sie die aktuellen AWS-Credentials bei jedem Gebrauch in Datei ~/.aws/credentials

.NET 6
Prüfen Sie, ob der dotnet sdk 6.0.x installiert ist.

```
dotnet --list-sdks
```

Falls dotnet sdk 6.0.x nicht aufgelistet wird, installieren sie ihn wie folgt:

```
sudo apt-get update
```

```
sudo apt-get install -y dotnet-sdk-6.0
```

Amazon Tools
Installieren Sie die neusten Amazon Lambda Templates:

```
dotnet new --install Amazon.Lambda.Templates
```

Installieren Sie nun die Amazon Lambda Tools.

```
dotnet tool install -g Amazon.Lambda.Tools
```

Falls eine Fehlermeldung, dass die Tools bereist installiert sind, führen Sie einen Update durch.

```
dotnet tool update -g Amazon.Lambda.Tools
```

Damit der Pfad auf die dotnet tools (z.B. dotnet lambda) in der Bash bekannt sind, muss die Umgebungsvariable PATH gesetzt werden. Der folgende Befehl erstellt die Datei ~/.bash_profile mit dem Pfad auf die dotnet tools.

```
cat << \EOF >> ~/.bash_profile
# Add .NET Core SDK tools
export PATH="$PATH:/home/vmadmin/.dotnet/tools"
EOF
```

Wichtig!
Der Pfad ist erst bekannt, nachdem Sie sich in der VM abgemeldet und neu angemeldet haben.

Aufgabe 2: Lamda Function mit Auruf über HTTP-Request | Partnerarbeit

Erstellen Sie ein .NET Projekt für eine Lambda-Function:

```
dotnet new lambda.EmptyFunction -n HelloGbs
```

Navigieren Sie ins neu erstellt Projektverzeichnis «HelloGbs/src/HelloGbs» und fügen Sie dem Projekt das Nuget-Package Amazon.Lambda.APIGatewayEvents hinzu:

```
dotnet add package Amazon.Lambda.APIGatewayEvents
```

Fügen Sie in Function.cs und FunctionTest.cs ganz zuoberst noch folgendes using-Statement ein:

```
using Amazon.Lambda.APIGatewayEvents;
```

Ersetzen Sie in Function.cs die Methode FunctionHandler mit folgendem Code:

```
public string FunctionHandler(APIGatewayHttpApiV2ProxyRequest request,
ILambdaContext context)
{
   context.Logger.LogInformation(request.RawQueryString);
   return "Yeahhhhhh:" + request.RawQueryString;
}
```

Ersetzen Sie in FunctionTest.cs Die TestUpperFunction mit folgendem Code:

```
public void TestFunction()
{
   var function = new Function();
   var context = new TestLambdaContext();
   var apiGatewayRequest = new APIGatewayHttpApiV2ProxyRequest();
   apiGatewayRequest.RawQueryString ="test";
   var result = function.FunctionHandler(apiGatewayRequest, context);
   Assert.Equal("Yeahhhhhh:test", result);
}
```

Navigieren Sie ins Testverzeichnis «HelloGbs/test/HelloGbs.Tests» und führen Sie die Tests aus:

```
dotnet test
```

Navigieren Sie ins Projektverzeichnis «HelloGbs/src/HelloGbs» und deployen Sie Ihre Lamda-Function:

```
dotnet lambda deploy-function --function-role LabRole gbslambda
```

Damit die Lamda-Function von Aussen her aufrufbar ist, muss noch eine Function-URL erstellt werden:

```
LAMBDA_URL=$(aws lambda create-function-url-config \
--function-name gbslambda \
--auth-type NONE \
--query 'FunctionUrl' \
--output text)

aws lambda add-permission \
--function-name gbslambda \
--statement-id AuthNone \
--action lambda:InvokeFunctionUrl \
--principal "*" \
--function-url-auth-type NONE

echo "Lamda-URL: $LAMBDA_URL?param1=x&param2=y"
```