1. **Setting up the project and the development environment**

**Solutions:**

**1.1 Install Node.js and its package manager (npm)**

**OS: Ubuntu 18.04**

Latest supported node js runtime for AWS Lambda function is 12.x so we will install node js 12.x version. There are tons of ways to install node js, but here we will consider installing node js from NodeSource repository. Follow these steps.

1. Enable the NodeSource repository by running the following [curl](https://linuxize.com/post/curl-command-examples/)

|  |
| --- |
| $ curl -sL https://deb.nodesource.com/setup\_12.x | sudo -E bash - |

1. Once the NodeSource repository is enabled, install Node.js and npm by typing:

|  |
| --- |
| $ sudo apt install nodejs |

1. Now verify that node js and npm were installed successfully.

|  |
| --- |
| $ echo "Node Version" $(node -v) && echo "Npm Version" $(npm -v) |

Output

|  |
| --- |
| Node Version v12.16.3 Npm Version 6.14.4 |

**1.2. Install AWS CLI tools in your machine.**

Install AWS CLI (version 2) by typing following command.

|  |
| --- |
| curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip" unzip awscliv2.zip sudo ./aws/install |

If you are using a different operating system or have already running AWS CLI 1.x version then you can find more details to install and upgrade to the latest version here. <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install>

**1.3 Install Serverless Framework (npm package) to help you organize and deploy your application to AWS by typing below command.**

|  |
| --- |
| **$ npm install -g serverless** |

Once the installation process is done, Verify that serverless is installed successfully by running the following command in your terminal:

|  |
| --- |
| serverless --version |

**1.4 Creating AWS Access Keys**

The Serverless Framework needs access to your cloud provider account so that it can create and manage resources on your behalf.

Follow these steps to create an IAM user for the Serverless Framework:

1. Login to your AWS account and go to the Identity & Access Management (IAM) page.
2. Click on Users and then Add user. Enter a name in the first field to remind you this User is related to the Serverless Framework, called it **serverless-admin**. Enable Programmatic access by clicking the checkbox. Click Next to go through to the Permissions page. Click on Attach existing policies directly. Search for and select AdministratorAccess then click Next: Review. Check to make sure everything looks good and click Create user.
3. View and copy the API Key & Secret to a temporary place. You'll need it in the next step.

**1.5 Configure AWS account credentials on your machine**

Serverless provides a convenient way to configure AWS profiles with the help of the serverless config credentials command.

|  |
| --- |
| **$ serverless config credentials --provider aws --key key\_which\_you\_have\_noted\_in\_previous\_steps --secret secret\_which\_you\_have\_noted\_in\_previous\_steps** |

Alternatively, you can configure the credentials using aws-cli

|  |
| --- |
| $ aws configure AWS Access Key ID [None]: **key\_which\_you\_have\_noted\_in\_previous\_steps** AWS Secret Access Key [None]: **key\_which\_you\_have\_noted\_in\_previous\_steps** Default region name [None]: us-west-1 Default output format [None]: ENTER |

**1.6 Setup HelloWorld serverless template and invoke locally and deploy Lambda function.**

Run the following command to create HelloWorld service.

|  |
| --- |
| # Create service with nodeJS template in the folder ./HelloWorld $ serverless create --template aws-nodejs --path HelloWorld |

Once the command runs successfully you should see the following files under HelloWorld folder.

* serverless.yml
* handler.js

Now invoke **hello** lambda function, defined in severless.yml

|  |
| --- |
| $ serverless invoke local --function hello |

After the successful local invocation of **hello** lambda function, It will print below the JSON response.

|  |
| --- |
| {  "statusCode": 200,  "body": "{\n \"message\": \"Go Serverless v1.0! Your function executed successfully!\",\n \"input\": \"\"\n}" } |

Now, upload and deploy your Lambda function by typing serverless deploy command.

|  |
| --- |
| $ serverless deploy |

Once deployment is successful you should see below output

|  |
| --- |
| Serverless: Packaging service... Serverless: Excluding development dependencies... Serverless: Creating Stack... Serverless: Checking Stack create progress... ........ Serverless: Stack create finished... Serverless: Uploading CloudFormation file to S3... Serverless: Uploading artifacts... Serverless: Uploading service helloworld.zip file to S3 (389 B)... Serverless: Validating template... Serverless: Updating Stack... Serverless: Checking Stack update progress... ............... Serverless: Stack update finished... Service Information service: helloworld stage: dev region: us-east-1 stack: helloworld-dev resources: 6 api keys:  None endpoints:  None functions:  hello: helloworld-dev-hello layers:  None Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing. |

Now, Invoked deployed version of lambda function by typing below command.

|  |
| --- |
| $ serverless invoke --function hello |

You should see the output below.

|  |
| --- |
| {  "statusCode": 200,  "body": "{\n \"message\": \"Go Serverless v1.0! Your function executed successfully!\",\n \"input\": {}\n}" } |

Now, let's create a **ReadMe** file and mention the details of deliverables for this milestone.

**Node Version:**

|  |
| --- |
| Node Version v12.16.3 Npm Version 6.14.4 |

**AWS CLI Version:**

|  |
| --- |
| aws-cli/2.0.10 Python/3.7.3 Linux/5.3.0-1017-aws botocore/2.0.0dev14 |

**Serverless CLI and framework version:**

|  |
| --- |
| Framework Core: 1.69.0 Plugin: 3.6.9 SDK: 2.3.0 Components: 2.30.5 |

**Lambda function Invocation response:**

|  |
| --- |
| {  "statusCode": 200,  "body": "{\n \"message\": \"Go Serverless v1.0! Your function executed successfully!\",\n \"input\": \"\"\n}" } |