**Lab Steps**

Task 1: Sign in to AWS Management Console

1. Click on the button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

* Leave the **Account ID** as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
* Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username** and **Password** in AWS Console and click on the **Sign in** button

1. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1**.

Task 2: Setup Visual Studio Code

1. Open the visual studio code.
2. If you have already installed and using Visual studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.
4. Open Terminal by selecting View from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.

Graphical user interface, application

Description automatically generated

1. Once the terminal is ready, let us navigate to the **Desktop**.

cd Desktop

1. Create a new folder by running the below command.

mkdir task\_10106\_api

1. Change your present working directory to use the newly created folder by running the below command:

cd task\_10106\_api

1. Get the location of the present working directory by running the below command:

pwd

1. Note down the location, as you will open the same in the next steps.
2. Now click on the first icon Explorer present on the left sidebar.
3. Click on the button called Open folder and navigate to the location of folder **task\_10106\_api**
4. (Optional) Click on Authorize button for allowing Visual Studio Code to use the **task\_10096\_api** folder. This will only be asked when you have been using Visual Studio code for a while as you are allowing a new folder to be accessed by VSC.
5. Visual Studio Code is now ready to use.

Task 3: Create a variable file

In this task, you will create a variable file where you will declare all the global variables with a short description and a default value.

1. To create a variable file, expand the folder **task\_10106\_api** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press Enter to save it.
3. **Note: Don't change the location of the new file, keep it default, i.e. inside the task\_10106\_api folder.**
4. Paste the below contents in **variables.tf** file.

|  |
| --- |
| variable "access\_key" {      description = "Access key to AWS console"  }  variable "secret\_key" {      description = "Secret key to AWS console"  }  variable "region" {      description = "AWS region"  } |

1. In the above content, you are declaring a variable called, access\_key, secret\_key, and region with a short description of all 3.
2. After pasting the above contents, save the file by pressing **ctrl + S**.
3. Now expand the folder **task\_10106\_api** and click on the **New File** icon to add the file.
4. Name the file as **terraform.tfvars** and press Enter to save it.
5. Paste the below content into the **terraform.tfvars** file.

|  |
| --- |
| region = "us-east-1"  access\_key = "<YOUR\_ACCESS\_KEY>"  secret\_key = "<YOUR\_SECRET\_KEY>" |

1. In the above code, you are defining the dynamic values of variables declared earlier.
2. Replace the values of access\_key and secret\_key by copying from the lab page.
3. After replacing the values of access\_key and secret\_key, save the file by pressing **Ctrl + S**.

Graphical user interface

Description automatically generated

Task 4: Create an IAM role for the Lambda in the main.tf file

 In this task, you will create a **main.tf** file where you will add details of the provider and resources.

1. To create a **main.tf** file, expand the folder **task\_10106\_api** and click on the New File icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

|  |
| --- |
| provider "aws" {      region     = "${var.region}"      access\_key = "${var.access\_key}"      secret\_key = "${var.secret\_key}"  } |

1. In the above code, you are defining the provider as **AWS**.
2. Next, we want to tell Terraform to create an IAM role for the Lambda function.
3. In AWS Console, navigate to the **IAM** by clicking on **Services** on the top and search for **IAM**.
4. **Click** on the **Policies** on the left navigation panel and search **lambda\_Rolepolicy**. **Copy the ARN** and save it in notepad.  
   A picture containing timeline

   Description automatically generated
5. Paste the following code into the **main.tf** file.

|  |
| --- |
| resource "aws\_iam\_role" "iam\_for\_lambda" {    name = "iam\_for\_lambda"    assume\_role\_policy = <<EOF  {    "Version": "2012-10-17",    "Statement": [      {        "Action": "sts:AssumeRole",        "Principal": {          "Service": "lambda.amazonaws.com"        },        "Effect": "Allow",        "Sid": ""      }    ]  }  EOF  } |

1. We will also describe the policy to the IAM role created. This policy will provide Lambda function access to CloudWatch logs.
2. Paste the below content into the **main.tf** file after the provider. Paste the policy arn copied earlier.

|  |
| --- |
| resource "aws\_iam\_role\_policy\_attachment" "lambda\_policy" {    role       = aws\_iam\_role.iam\_for\_lambda.name    policy\_arn = "<PASTE\_POLICY\_ARN\_HERE>"  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S.**

Task 5: Create a lambda function in main.tf file

In this task we are going to create a lambda function.

1. Firstly you will download [**lambda\_function.zip**](https://labresources.whizlabs.com/d5d10ebfafc0e7c907b2bb679200b042/lambda_function.zip) and upload it locally to your folder **task\_10106\_api.**
2. Name the zip file as **lambda\_function.zip**.

Text

Description automatically generated

1. To create a lambda function, paste the following contents in the **main.tf** file

|  |
| --- |
| resource "aws\_lambda\_function" "test\_lambda" {    filename      = "lambda\_function.zip"    function\_name = "WhizlabsAPI"    role          = aws\_iam\_role.iam\_for\_lambda.arn    handler       = "lambda\_function.lambda\_handler"    runtime = "python3.8"    source\_code\_hash = filebase64sha256("lambda\_function.zip")  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S.**

Task 6: Create a REST API,it's method and resource in main.tf file

In this task we are going to create a Rest API and its method and resource and paste it in the **main.tf** file.

1. To create a REST API, add another block just below the upload object code in **main.tf** file.

|  |
| --- |
| resource "aws\_api\_gateway\_rest\_api" "testAPI" {    name        = "WhizAPI"    description = "This is my API for demonstration purposes"   endpoint\_configuration {      types            = ["REGIONAL"]   }  } |

1. In the above code ,we have declared the name , description and type of the rest api created.
2. To create a resource for the Rest API , paste the following contents in the **main.tf** file.

|  |
| --- |
| resource "aws\_api\_gateway\_resource" "testresource" {    parent\_id   = aws\_api\_gateway\_rest\_api.testAPI.root\_resource\_id    path\_part   = "whizapi"    rest\_api\_id = aws\_api\_gateway\_rest\_api.testAPI.id  } |

1. In the above code, we have declared the path part , parent id and the REST API associated with it.
2. For creating the method , paste the following code in the **main.tf** file:

|  |
| --- |
| resource "aws\_api\_gateway\_method" "testMethod" {    rest\_api\_id   = aws\_api\_gateway\_rest\_api.testAPI.id    resource\_id   = aws\_api\_gateway\_resource.testresource.id    http\_method   = "GET"    authorization = "NONE"  } |

1. To create a method response , paste the following code in the **main.tf** file.

|  |
| --- |
| resource "aws\_api\_gateway\_method\_response" "response\_200" {    rest\_api\_id = aws\_api\_gateway\_rest\_api.testAPI.id    resource\_id = aws\_api\_gateway\_resource.testresource.id    http\_method = aws\_api\_gateway\_method.testMethod.http\_method    status\_code = "200"    response\_models = {         "application/json" = "Empty"     }  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S**.

Task 7: Create a gateway integration and deploy the API in the main.tf file

1. To create a gateway integration , paste the following code in the **main.tf** file.

|  |
| --- |
| resource "aws\_api\_gateway\_integration" "MyDemoIntegration" {    rest\_api\_id          = aws\_api\_gateway\_rest\_api.testAPI.id    resource\_id          = aws\_api\_gateway\_resource.testresource.id    http\_method          = aws\_api\_gateway\_method.testMethod.http\_method    integration\_http\_method     = "POST"     uri = aws\_lambda\_function.test\_lambda.invoke\_arn    type                 = "AWS"    passthrough\_behavior    = "WHEN\_NO\_TEMPLATES"    } |

1. In the above code, we have declared the rest api, resource , method and the lambda function invoke arn.
2. To add the gateway integration response , paste the following in the **main.tf** file

|  |
| --- |
| resource "aws\_api\_gateway\_integration\_response" "MyDemoIntegrationResponse" {    rest\_api\_id = aws\_api\_gateway\_rest\_api.testAPI.id    resource\_id = aws\_api\_gateway\_resource.testresource.id    http\_method = aws\_api\_gateway\_method.testMethod.http\_method    status\_code = aws\_api\_gateway\_method\_response.response\_200.status\_code    response\_templates = {         "application/json" = ""     }     depends\_on = [      aws\_api\_gateway\_integration.MyDemoIntegration    ]  } |

Text

Description automatically generated

1. To deploy the API , add the below contents in the **main.tf** file:

|  |
| --- |
| resource "aws\_api\_gateway\_deployment" "testdep" {    rest\_api\_id = aws\_api\_gateway\_rest\_api.testAPI.id    triggers = {      redeployment = sha1(jsonencode([        aws\_api\_gateway\_resource.testresource.id,        aws\_api\_gateway\_method.testMethod.id,        aws\_api\_gateway\_integration.MyDemoIntegration.id,      ]))    }    depends\_on = [aws\_api\_gateway\_integration.MyDemoIntegration]    lifecycle {      create\_before\_destroy = true    }  } |

Text

Description automatically generated

1. For deployment, there is a need to create a stage as well. To create the stage for the deployment , paste the following contents in the **main.tf** file:

|  |
| --- |
| resource "aws\_api\_gateway\_stage" "teststage" {   deployment\_id = aws\_api\_gateway\_deployment.testdep.id    rest\_api\_id   = aws\_api\_gateway\_rest\_api.testAPI.id     stage\_name    = "whizstage"    } |

Text

Description automatically generated

1. The last thing we need in the **main.tf** file is to provide api gateway the permission to invoke the lambda function. Add the following code in the **main.tf** file:

|  |
| --- |
| resource "aws\_lambda\_permission" "api\_gw" {    statement\_id  = "AllowExecutionFromAPIGateway"    action        = "lambda:InvokeFunction"    function\_name = "${aws\_lambda\_function.test\_lambda.function\_name}"    principal     = "apigateway.amazonaws.com"    source\_arn = "${aws\_api\_gateway\_rest\_api.testAPI.execution\_arn}/\*/\*"  } |

Task 8: Create an Output file

In this task, you will create an output.tf file where you will add details of the provider and resources.

1. To create an **output.tf** file, expand the folder **task\_10106\_api** and click on the **New File** icon to add the file.
2. Name the file as **output.tf** and press **Enter** to save it.
3. Paste the below content into the **output.tf** file. In the above code, we will extract the API Invoke URL.

|  |
| --- |
| output "api\_invoke\_url" {   value = "${aws\_api\_gateway\_stage.teststage.invoke\_url}/${aws\_api\_gateway\_resource.testresource.path\_part}"  } |

Task 9: Confirm the installation of Terraform by checking the version

1. In the Visual Studio Code, open Terminal by selecting View from the Menu bar and choose Terminal.
2. If you are not in the newly created folder change your present working directory by running the below command.

cd task\_10106\_api

1. To confirm the installation of Terraform, run the below command to check the version:

terraform version

1. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 10: Apply terraform configurations

1. Initialize Terraform by running the below command,

terraform init

Text

Description automatically generated

**Note:** terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan

1. To generate the action plans run the below command,

terraform plan

Text

Description automatically generated

1. To create all the resources declared in main.tf configuration file, run the below command,

terraform apply

1. Approve the creation of all the resources by entering **yes**.

Text

Description automatically generated

* **Note: This process will take around 4-5 minutes.**

1. Id’s of all the resources created by terraform will be visible there.

Text

Description automatically generated

1. Copy the **api\_invoke\_url** and **paste** it in the browser to see if the lambda is successfully integrated to the Lambda function.

**Graphical user interface, text, application

Description automatically generated**

Task 11: Check the resources in AWS Console

1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **API Gateway** by clicking on **Services** on the top, then click on **API Gateway** in the **Networking and Content Delivery** section.
3. Click on **APIs** in left navigation menu.

Graphical user interface, application

Description automatically generated

1. Select the **API** and click on **GET**. This is the execution for the API Gateway integrated with lambda function.

Graphical user interface, text, application

Description automatically generated

Task 12: Delete AWS Resources

1. To delete the resources, open Terminal again.
2. Run the below command to delete all the resources. Enter **yes** to confirm the deletion.

terraform destroy

Graphical user interface, text

Description automatically generated

1. You can see the **Destroy complete!** message.

Text

Description automatically generated

**Completion and Conclusion**

* You have succesfully set up the Visual Studio Code editor.
* You have succesfully created variables.tf and terraform.tfvars files.
* You have successfully created an IAM role for the lambda function in the main.tf file.
* You have successfully created a lambda function in the main.tf file
* You have successfully created a rest api, its method and response in the main.tf file
* You have successfully created a gateway integration in the main.tf file.
* You have successfully created an output file.
* You have successfully checked all the resources created by opening the Console.
* You have deleted all the resources.

End Lab