**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.
6. 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\_10126

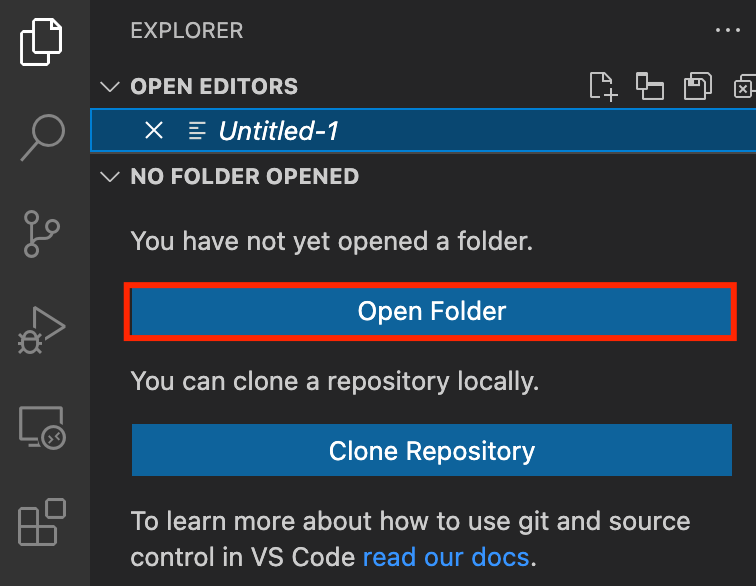
1. Change your present working directory to use the newly created folder by running the below command:

cd task\_10126

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\_10126**.



1. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task\_10126 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.
2. Visual Studio Code is now ready to use.

Task 3: Create a variable file

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

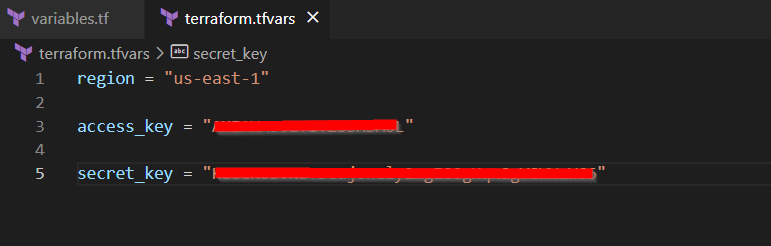
1. To create a variable file, expand the folder **task\_10126** and click on the **New** **File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task\_10126** folder**.**
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\_10126** 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 AWS CONSOLE ACCESS ID>"  secret\_key = "<YOUR AWS CONSOLE 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.



Task 4: Launch an EC2 in main.tf file

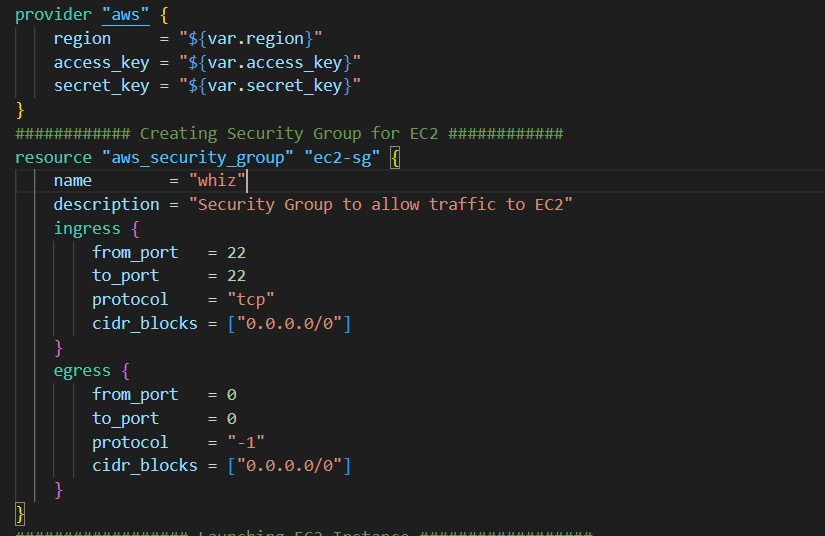
In this task, you will create a **main.tf** file where you will add details of the provider and resources.

1. To create a **main.tf** file, expand the folder **task\_10126** 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}"  } |

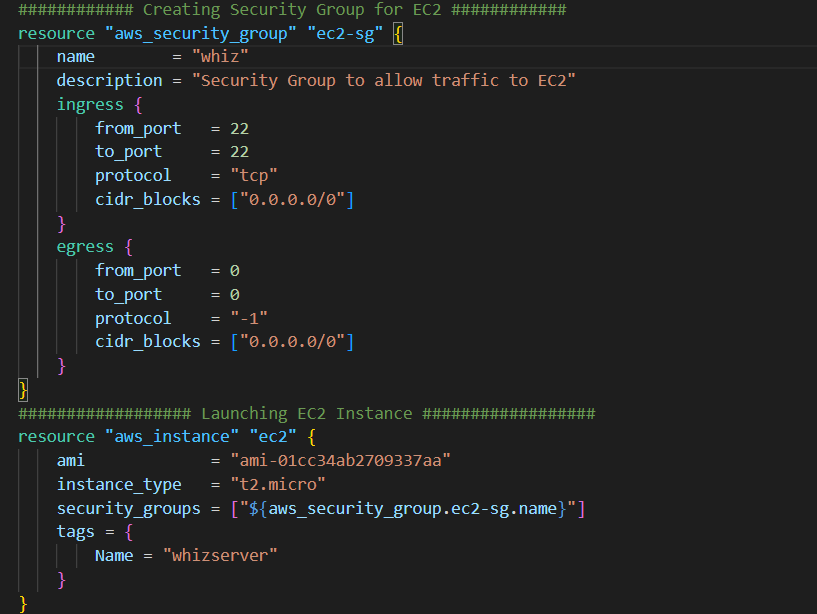
1. In the above code, you are defining the provider as aws.
2. Next, we want to tell Terraform to create a security group for EC2 Instance
3. To create a security group. Paste the below content into the **main.tf** file after the provider

|  |
| --- |
| ############ Creating Security Group for EC2 ############  resource "aws\_security\_group" "ec2-sg" {      name        = "whiz"      description = "Security Group to allow traffic to EC2"      ingress {          from\_port   = 22          to\_port     = 22          protocol    = "tcp"          cidr\_blocks = ["0.0.0.0/0"]      }      egress {          from\_port   = 0          to\_port     = 0          protocol    = "-1"          cidr\_blocks = ["0.0.0.0/0"]      }  } |



1. Next, we’ll tell terraform to Launch an EC2
2. To launch an EC2 Instance. Paste the below content into the **main.tf** file after the security group code.

|  |
| --- |
| ################## Launching EC2 Instance ##################  resource "aws\_instance" "ec2" {      ami             = "ami-01cc34ab2709337aa"      instance\_type   = "t2.micro"      security\_groups = ["${aws\_security\_group.ec2-sg.name}"]      tags = {          Name = "whizserver"      }  } |



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

Task 5: Create an SNS Topic in main.tf file

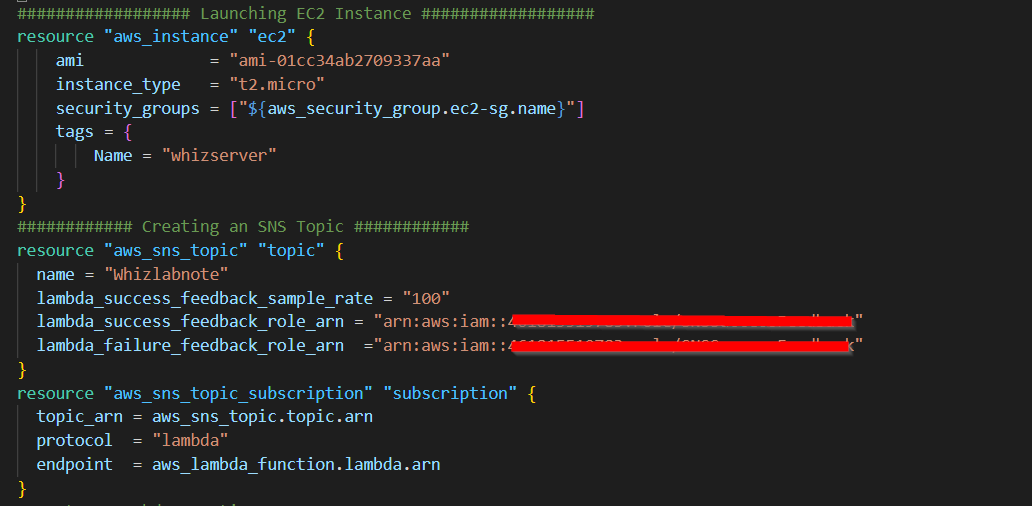
In this task you will create an SNS Topic and lambda subscription in main.tf file

1. To Create an SNS Topic Role ARN is required.
2. To get Role ARN In Console navigate to **IAM** by clicking on **Services** on the top, then click on **IAM**.
3. Click on the **Roles** on the left navigation panel and search **SNS\_role**
4. Click on **SNS\_role\_<RANDOM\_NUMBER>** role and copy the **ARN** and paste it in notepad
5. Now To create an SNS Topic paste the below code in main.tf file after the EC2 code

|  |
| --- |
| ############ Creating an SNS Topic ############  resource "aws\_sns\_topic" "topic" {    name = "Whizlabnote"    lambda\_success\_feedback\_sample\_rate = "100"    lambda\_success\_feedback\_role\_arn = "ENTER ROLE ARN HERE"    lambda\_failure\_feedback\_role\_arn  ="ENTER ROLE ARN HERE"  } |

1. Replace the **lambda\_success\_feedback\_role\_arn** and **lambda\_failure\_feedback\_role\_arn** with Role ARN copied.
2. Now to create a lambda subscription paste the below code after topic code

|  |
| --- |
| resource "aws\_sns\_topic\_subscription" "subscription" {    topic\_arn = aws\_sns\_topic.topic.arn    protocol  = "lambda"    endpoint  = aws\_lambda\_function.lambda.arn  } |



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

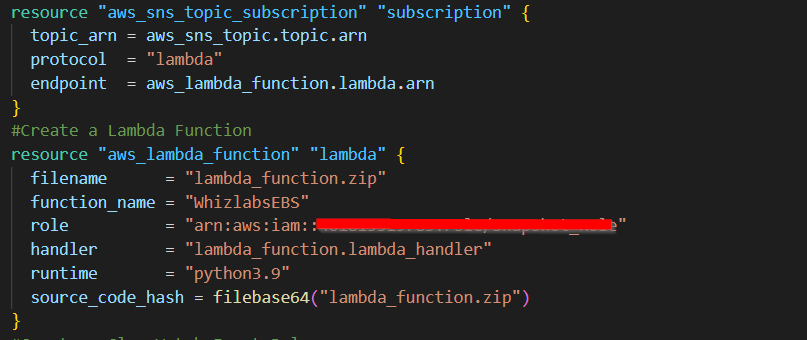
Task 6: Create a Lambda Function in main.tf file

In this task you will create an Lambda Function in main.tf file

1. Firstly you will [download lambda\_function.zip](https://labresources.whizlabs.com/858abe7002bfdae3ad38bcfcf93b55fb/lambda_function_48_54.zip) and upload it locally to your folder **task\_10126**
2. Name the zip file as **lambda\_function.zip**
3. To create a Lambda Function we require Role ARN
4. To get ROLE ARN navigate to **IAM.** Click on the **Roles** on the left navigation panel and search **Lambda\_role**
5. Click on **Lambda\_role\_<RANDOM\_NUMBER>** role and copy the **ARN** and paste it in notepad
6. To create a Lambda Function paste the below code in main.tf file after subscription code

|  |
| --- |
| #Create a Lambda Function  resource "aws\_lambda\_function" "lambda" {    filename      = "lambda\_function.zip"    function\_name = "WhizlabsEBS"    role          = "ENTER ROLE ARN HERE"    handler       = "lambda\_function.lambda\_handler"    runtime       = "python3.9"    source\_code\_hash = filebase64("lambda\_function.zip")  } |

1. Replace the **role** with Lambda Role ARN copied.



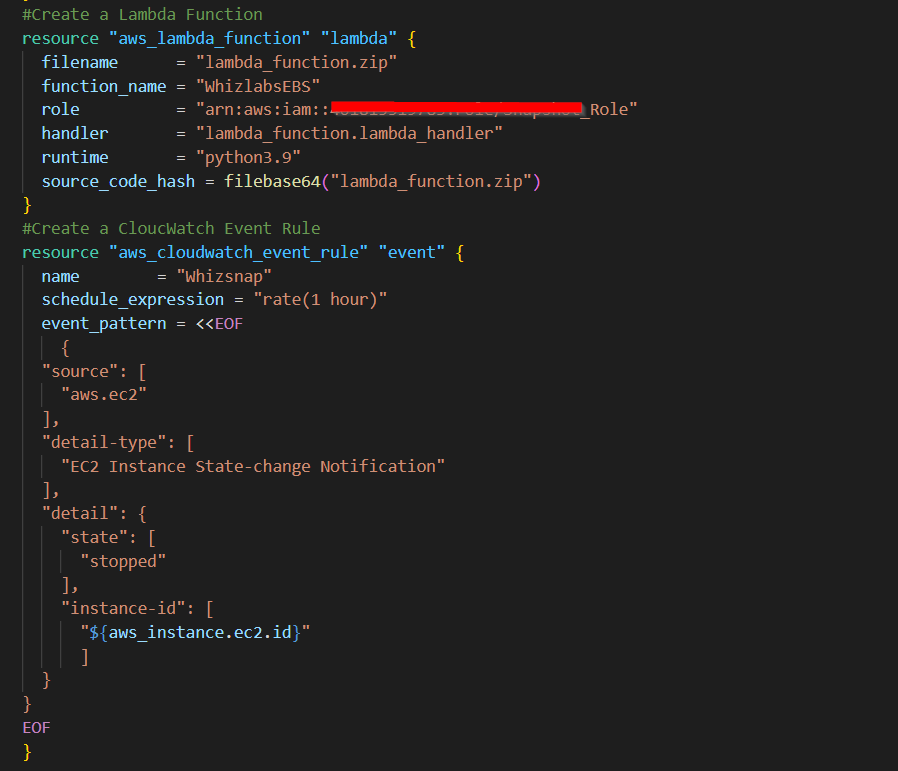
1. Save the file by pressing **Ctrl + S**.

Task 7: Create a CloudWatch Event Rule in main.tf file

In this task you will create a CloudWatch Event Rule in the main.tf file

1. To create a CloudWatch Event Rule paste the below code in main.tf file after the Lambda code

|  |
| --- |
| #Create a CloucWatch Event Rule  resource "aws\_cloudwatch\_event\_rule" "event" {    name        = "Whizsnap"    schedule\_expression = "rate(1 hour)"    event\_pattern = <<EOF      {    "source": [      "aws.ec2"    ],    "detail-type": [      "EC2 Instance State-change Notification"    ],    "detail": {      "state": [        "stopped"      ],      "instance-id": [        "${aws\_instance.ec2.id}"        ]    }  }  EOF  } |



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

Task 8: Add SNS Destination to Lambda in main.tf file

In this task you will add the SNS topic as a destination for the Lambda function when the event is triggered that allows the AWS CloudWatch event rule to trigger specified SNS topic

1. To Add Destination  paste the below code in main.tf file after the Event Rule code

|  |
| --- |
| # Add SNS Destination to Lambda Function  resource "aws\_cloudwatch\_event\_target" "sns" {    rule      = aws\_cloudwatch\_event\_rule.event.name    target\_id = "SendToSNS"    arn       = aws\_sns\_topic.topic.arn  }  resource "aws\_lambda\_function\_event\_invoke\_config" "sns" {    function\_name = aws\_lambda\_function.lambda.function\_name    destination\_config {      on\_failure {        destination = aws\_sns\_topic.topic.arn      }      on\_success {        destination = aws\_sns\_topic.topic.arn      }    }  } |

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

Task 9: Add CloudWatch Event Target to Lambda in main.tf file

In this task you will add the CloudWatch event target to the Lambda function, so that the lambda function can be triggered by the CloudWatch event and give permission for the EventBridge to invoke the function

1. To Add Event Target  paste the below code in main.tf file after the Destination code

|  |
| --- |
| # Add Cloudwatch Event Target to Lambda Function  resource "aws\_cloudwatch\_event\_target" "lambda" {    rule      = aws\_cloudwatch\_event\_rule.event.name    target\_id = "SendToLambda"    arn       = aws\_lambda\_function.lambda.arn  }  resource "aws\_lambda\_permission" "allow\_eventbridge" {    statement\_id  = "AllowExecutionFromEventBridge"    action        = "lambda:InvokeFunction"    function\_name = aws\_lambda\_function.lambda.function\_name    principal     = "events.amazonaws.com"    source\_arn    = aws\_cloudwatch\_event\_rule.event.arn  } |



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

Task 10: 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\_10126** 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.

|  |
| --- |
| output "ec2" {    value       = aws\_instance.ec2.id  }  output "topic" {    value       = aws\_sns\_topic.topic.id  }  output "lambda" {    value       = aws\_lambda\_function.lambda.id  }  output "rule" {    value       = aws\_cloudwatch\_event\_rule.event.id  } |

1. In the above code, we will extract id’s of Resources created to confirm that they are created.

Task 11: 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\_10126

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

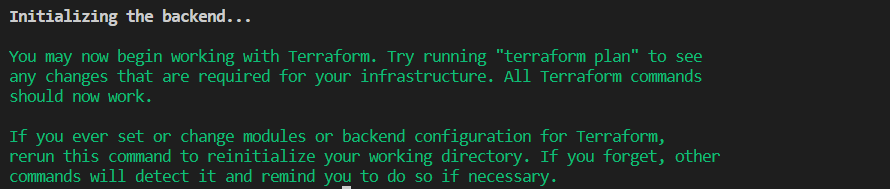
terraform version

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 12: Apply terraform configurations

1. Initialize Terraform by running the below command

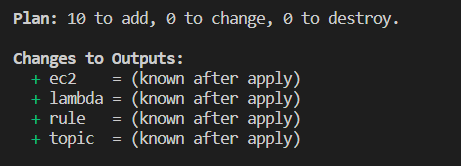
terraform init

****

**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,

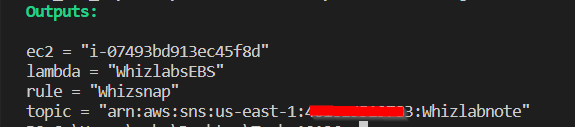
terraform plan



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

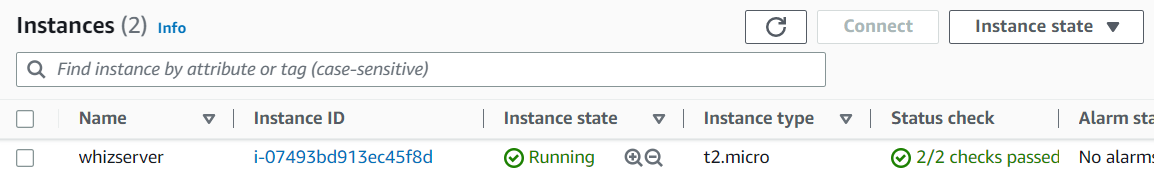
terraform apply

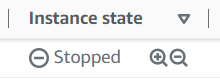
1. Approve the creation of all the resources by entering **yes**.
2. The ID’s of Resources Created by terraform will be visible there.

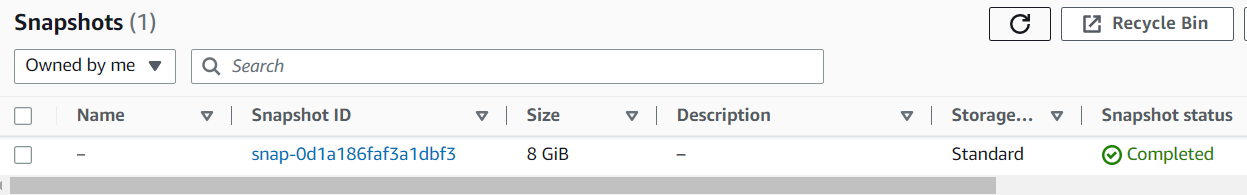


Task 13: Check the resources in AWS Console

1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **EC2** by clicking on **Services** on the top, then click on **EC2** in the **Compute** section.
3. Click on the **Instances** on the left navigation panel. You can see the Instances created successfully.



1. Select the EC2 instance, click on **Instance state** and click on **Stop instance**. Click on the **Stop** button on the pop up window and wait till the state changes to ****
2. To check the snapshot, click on **Snapshots** under **Elastic Block Store** in the left side panel. Please wait for sometime for the snapshot to get created completely and the status will change to **completed**.



Task 14: Delete AWS Resources

1. To delete the resources, open Terminal again.
2. Run the below command to delete all the resources.

terraform destroy

1. Approve the creation of all the resources by entering **yes**. You can see the **Destroy complete!** Message.

**Completion and Conclusion**

* You have successfully set up the Visual Studio Code editor.
* You have successfully created variables.tf and terraform.tfvars files.
* You have successfully Launched EC2 using terraform
* You have successfully created an SNS topic and created lambda subscription using terraform
* You have successfully created Lambda Function using terraform
* You have successfully created a CloudWatch Event Rule and targeted SNS Topic to it using terraform
* You have successfully created output.tf
* You have successfully executed the terraform configuration commands to create the resources.
* You have successfully checked all the resources created by opening the Console.
* You have successfully deleted all the resources.

**End Lab**