**Lab Steps**

**Task 1: Sign in to AWS Management Console**

1. On the AWS sign-in page, Enter **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button.
2. 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**

In this task, we are going to set up Visual Studio Code, which is a source code editor. It is used in this lab to write and manage the Terraform configuration files.

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

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

mkdir task\_10008

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

cd task\_10008

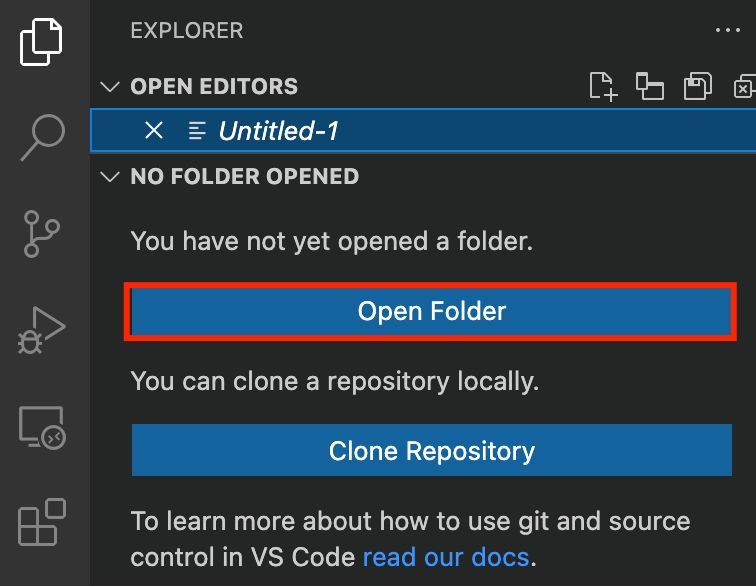
9. Get the location of the present working directory by running the below command:

pwd

10. Note down the location, as you will open the same in the next steps.

11. Now click on the first icon Explorer present on the left sidebar.

12. Click on the button called Open folder and navigate to the location of folder **task\_10008**.



13. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task\_10008 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.

14. Visual Studio Code is now ready to use.

**Task 3: Create a variables 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\_10008** 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\_10008** 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 = "Region of AWS VPC"

}

5. In the above content, you are declaring a variable called, access\_key, secret\_key, and region with a short description of all 3.

6. After pasting the above contents, save the file by pressing **ctrl + S**.

7. Now expand the folder **task\_10008** and click on the **New File** icon to add the file.

8. Name the file as **terraform.tfvars** and press **Enter** to save it.

9. Paste the below content into the **terraform.tfvars** file.

region      = "us-east-1"

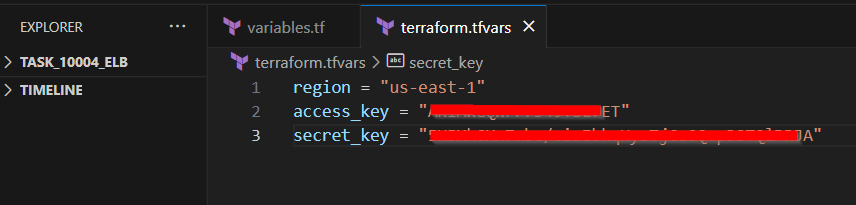
access\_key  = "<YOUR AWS CONSOLE ACCESS ID>"

secret\_key  = "<YOUR AWS CONSOLE SECRET KEY>"

10. In the above code, you are defining the dynamic values of variables declared earlier.

11. Replace the values of access\_key and secret\_key by copying from the lab page.

12. After replacing the values of access\_key and secret\_key, save the file by pressing **Ctrl + S**.



**Task 4: Create EC2 and it’s AMI 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\_10008** 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

}

4. In the above code, you are defining the provider as aws.

5. Next, we want to tell Terraform to create a Security Group within AWS EC2, and populate it with rules to allow traffic on specific ports. In our case, we are allowing the ssh and tcp port 80 (HTTP).

6. We also want to make sure the instance can connect outbound on any port, so we’re including an egress section below as well.

7. Paste the below content into the **main.tf** file after the provider.

resource "aws\_security\_group" "ec2sg" {

  name = "EC2-SG"

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  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"]

  }

}

8. Let's add another set of code after security group creation where you will create EC2 instance.

9. In the below code, we have defined the Amazon Linux 2 AMI. The AMI ID mentioned above is for the us-east-1 region.

10. We have mentioned the instance type as t2.micro. The security group ID is automatically taken by using the variable which we have set during the creation process.

11. We have added the user data to install the apache server and add a html page.

12. We have provided tags for the EC2 instance.

resource "aws\_instance" "ec2instance" {

  ami              = "ami-0c101f26f147fa7fd"

  instance\_type    = "t2.micro"

  security\_groups  = ["${aws\_security\_group.ec2sg.name}"]

  user\_data = <<-EOF

    #!/bin/bash

    sudo su

    yum update -y

    yum install httpd -y

    echo "<html><h1> Welcome to my labs </h1></html>" >> /var/www/html/index.html

    systemctl start httpd

    systemctl enable httpd

  EOF

  tags = {

    Name = "MyEC2Server"

  }

}

13. Let's add another set of code after EC2 Instances creation where you will create the ec2 ami using the ec2 instance created.

resource "aws\_ami\_from\_instance" "ec2ami" {

  name               = "MyEC2Image"

  source\_instance\_id = aws\_instance.ec2instance.id

}

**Task 5: Create an Output file**

In this task, you will create an **output.tf** file where you add details of the output you want to display.

1. To create an **output.tf** file, expand the folder **task\_10008** 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 "ami\_id" {

  value = aws\_ami\_from\_instance.ec2ami.id

}

4. In the above code, we will extract the ami id of the created EC2 AMI and display it once the resources are created.

**Task 6: 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\_10008

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

terraform version

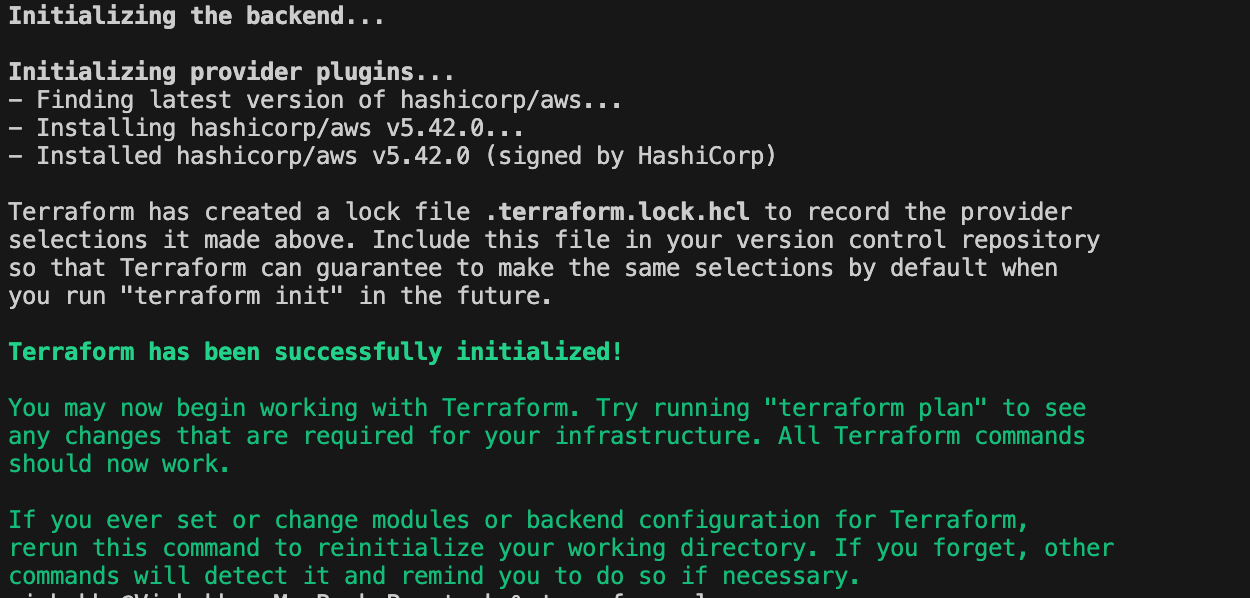
4. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, to install terraform follow the official guide link provided in the Prerequisite section above.

**Task 7: Apply terraform configurations**

1. Initialize Terraform by running the below command,

terraform init

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



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

terraform plan

4. Review the whole generated plan.

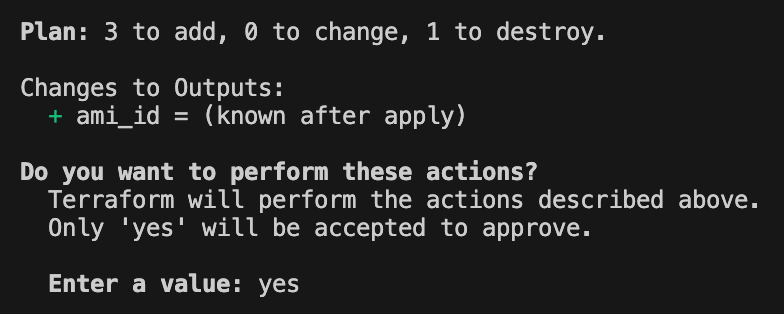
5. To create all the resources declared in main.tf configuration file, run the below command,

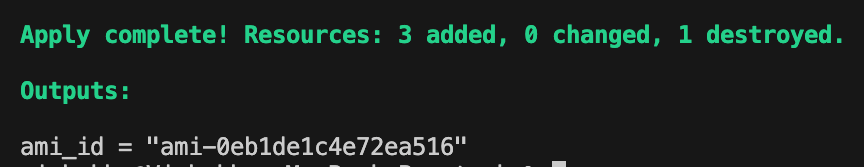
terraform apply

6. You will be able to see the resources which will be created, approve the creation of all the resources by entering **yes**.

7. It may take up to 5-6 minutes for the terraform apply command to create the resources.

8. Id of the ami created by terraform will be visible there.





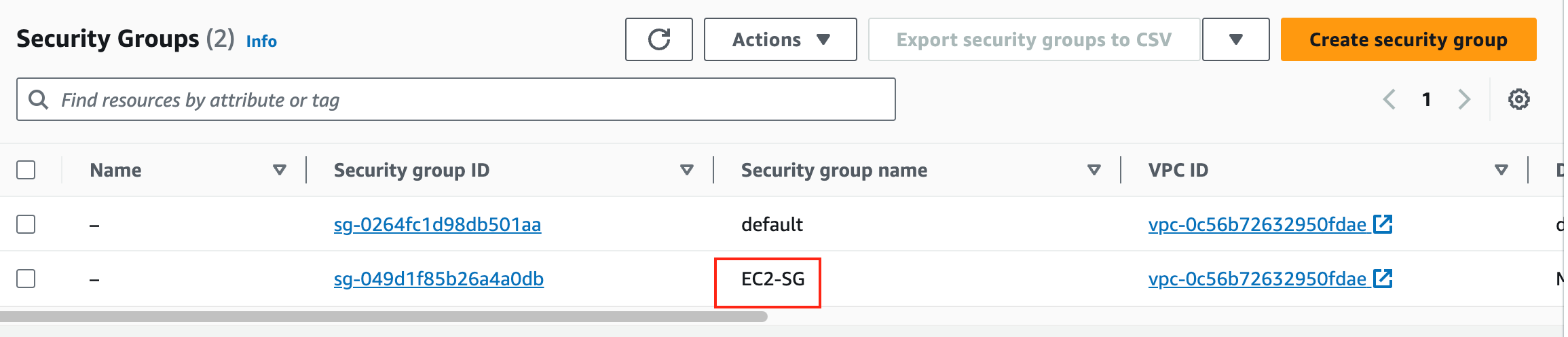
**Task 8: Check the resources in AWS Console**

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.

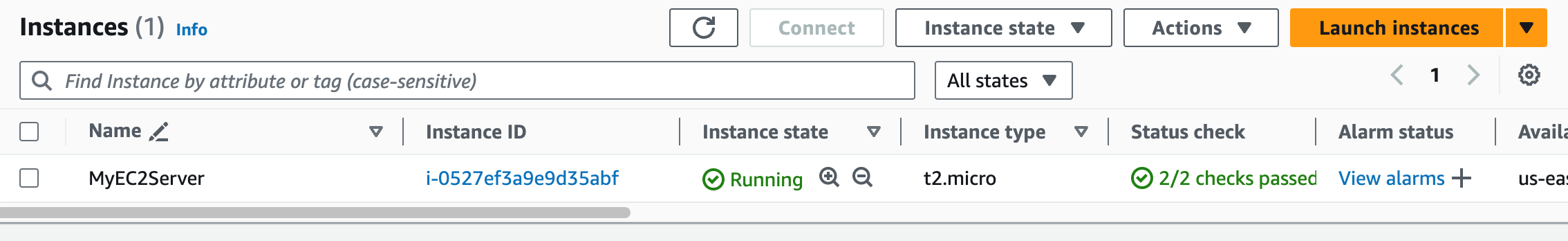
3. Navigate to **Security Groups** under **Network & Security** on the left panel.

4. You will be able to see the security group with the name **EC2-SG** which we have created in the terraform.



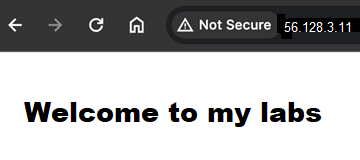
5. Navigate to **Instances** under **Instances** on the left panel.

6. You can see the instance created. You can check the configurations that we applied in the terraform file like security group, instance type, etc.



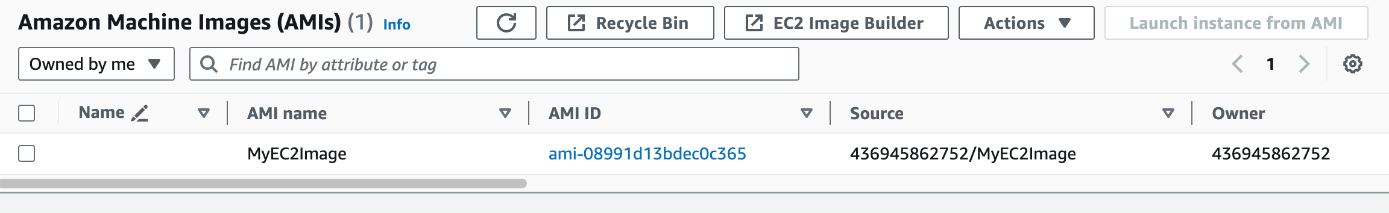
7. Select the MyEC2Server Instance and copy the IPv4 Public IP address of that instance.

8. Enter the IP Address in the Browser.



9. Note: If the Public IP of the instance is not opening any page, Make sure URL Protocol is **http** not https. Navigate to **AMIs** under **Images** on the left panel.

10. You will be able to see the created ami with the name **MyEC2Image** which we have created in the terraform.



**Task 9: Launching the EC2 Instance with the Created AMI using terraform**

1. To Launch an instance using created ami Go back to Visual Studio code and open **main.tf** file

2. Copy the below code and paste it in main.tf file after the EC2 AMI code

resource "aws\_instance" "ec2amiinstance" {

  ami              = aws\_ami\_from\_instance.ec2ami.id

  instance\_type    = "t2.micro"

  security\_groups  = ["${aws\_security\_group.ec2sg.name}"]

  tags = {

    Name = "MyEC2AMIServer"

  }

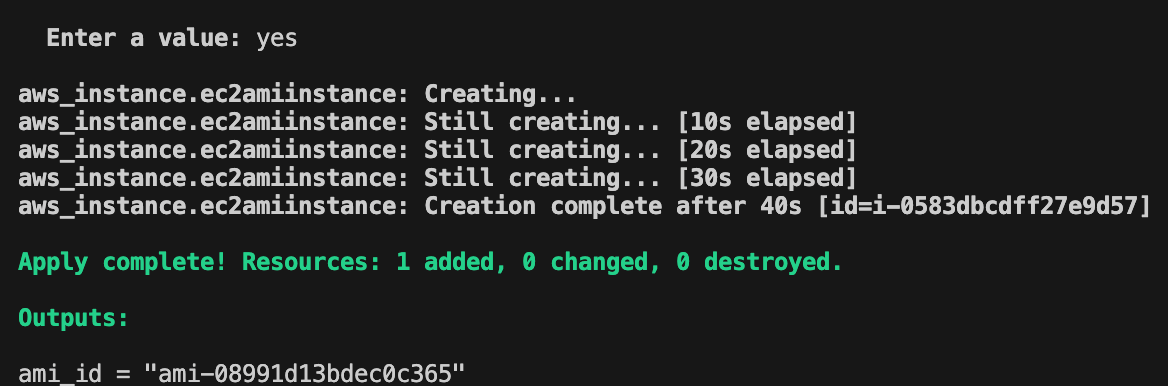
}

3. Save the file by pressing Ctrl + S

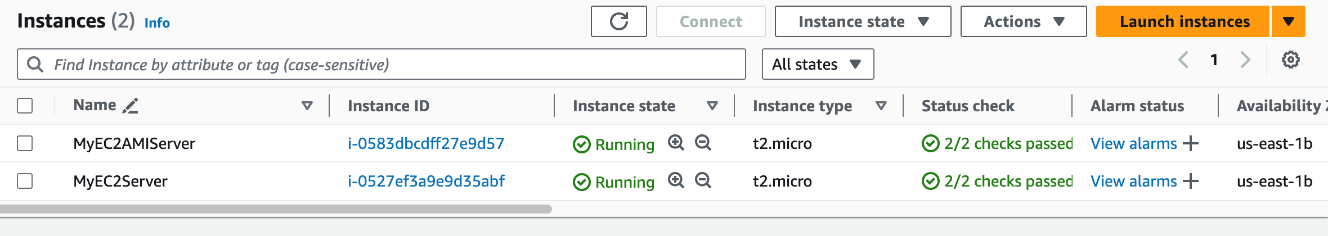
4. To modify instance type run the below command:

terraform apply

5. Approve the modification by entering **yes**.

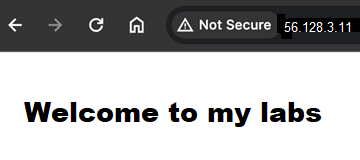


6. Now, Navigate to **Instances** page from left menu you will see 2 Instances running



7. Select the MyEC2AMIServer Instance and copy the IPv4 Public IP address of that instance.

8. Enter the IP Address in the Browser.



9. You will be able to see the HTML page displaying the message Welcome to my labs. This shows that the data in the new instance is the same as the one in the first instance we created.

**Do You Know?**  
The availability of thousands of public AMIs in the AWS Marketplace showcases the flexibility and scalability of the AWS ecosystem. Users can choose from a vast selection of pre-configured AMIs to quickly spin up instances with specific software stacks or use them as a starting point for their own customized environments.

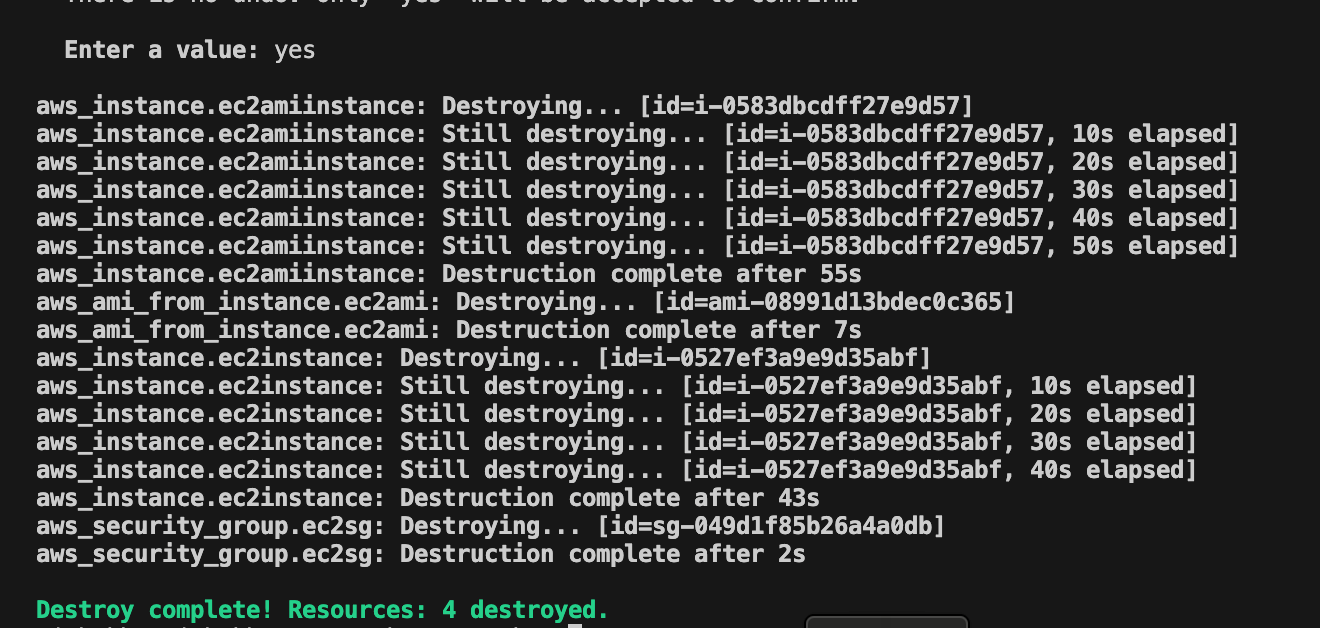
**Task 10: Delete AWS Resources**

1. To delete the resources, open Terminal again.

2. Run the below command to delete all the resources.

terraform destroy

3. Enter yes to confirm the deletion.



**Completion and Conclusion**

* You have set up the Visual Studio Code editor.
* You have created variables.tf and terraform.tfvars files.
* You have created a main.tf file.
* You have executed the terraform configuration commands to create the resources.
* You have checked all the resources created by opening the Console.
* You have deleted all the resources.