**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: Create a Key Pair

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 **Key Pairs** on the left navigation panel and click on button.
4. Name : Enter **WhizKey**
5. Private key file format : Select **.pem**
6. **NOTE: Don't change the Key Name and key format**
7. Keep Rest things default and click on button.

Task 3: 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\_10122

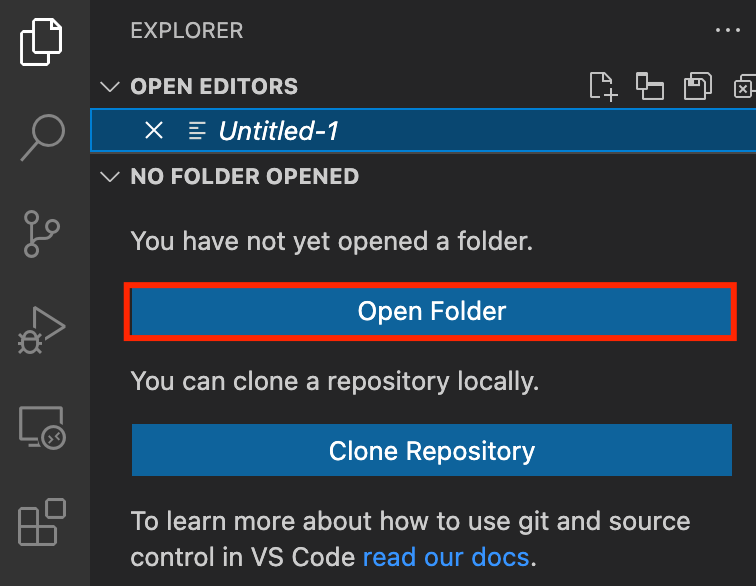
1. Change your present working directory to use the newly created folder by running the below command:

cd task\_10122

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\_10122**.



1. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task\_10122 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 4: 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\_10122** 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\_10122** 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\_10122** 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.

Graphical user interface

Description automatically generated

Task 5: Create a VPC 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\_10122** 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 VPC for EC2
3. To create a VPC Paste the below content into the **main.tf** file after the provider

|  |
| --- |
| # Create a VPC  resource "aws\_vpc" "vpc" {    cidr\_block = "10.0.0.0/16"    tags = {      Name = "MyVPC"    }  } |

Text

Description automatically generated

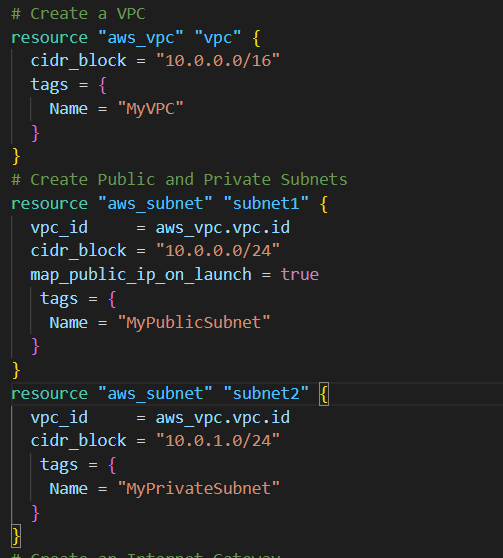
1. Save the file by pressing **Ctrl + S**.

Task 6: Create Public and Private Subnets in main.tf file

In this task you will create Public and Private Subnets for EC2 in main.tf file

1. To create Public and Private subnets paste the below code in mian.tf file after the vpc code

|  |
| --- |
| # Create Public and Private Subnets  resource "aws\_subnet" "subnet1" {    vpc\_id     = aws\_vpc.vpc.id    cidr\_block = "10.0.0.0/24"    map\_public\_ip\_on\_launch = true     tags = {      Name = "MyPublicSubnet"    }  }  resource "aws\_subnet" "subnet2" {    vpc\_id     = aws\_vpc.vpc.id    cidr\_block = "10.0.1.0/24"     tags = {      Name = "MyPrivateSubnet"    }  } |



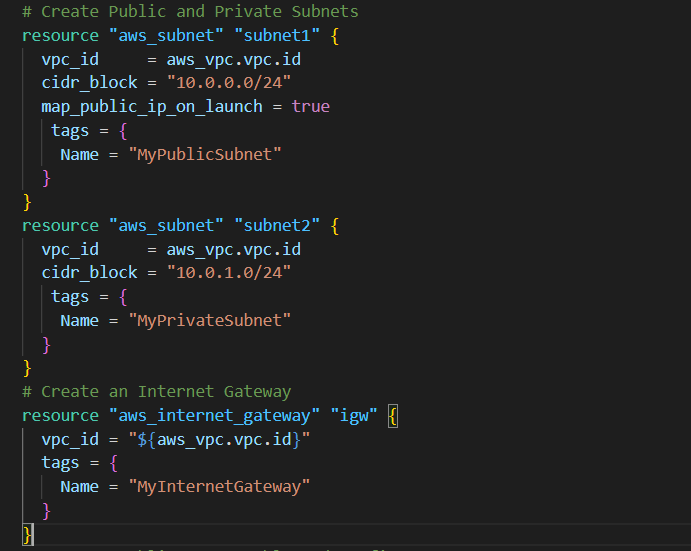
1. Save the file by pressing **Ctrl + S.**

Task 7: Create Internet Gateway in main.tf file

In this task you will create an Internet Gateway and Attach it to VPC in main.tf file

1. To create an Internet Gateway paste the below code in main.tf file after subnets code

|  |
| --- |
| # Create an Internet Gateway  resource "aws\_internet\_gateway" "igw" {    vpc\_id = "${aws\_vpc.vpc.id}"    tags = {      Name = "MyInternetGateway"    }  } |



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

Task 8: Create Public Route Table and Configure it in main.tf file

In this task you will create a route table in main.tf file and attach a public subnet to it. Instances launched within this subnet will have access to the Internet.

1. To create a Public Route Table paste the below code in main.tf file after the gateway code

|  |
| --- |
| #Create a Public Route table and Configure  resource "aws\_route\_table" "routetable" {    vpc\_id = aws\_vpc.vpc.id    route {      cidr\_block = "0.0.0.0/0"      gateway\_id = aws\_internet\_gateway.igw.id    }  }  resource "aws\_route\_table\_association" "subnetassociation" {    subnet\_id = aws\_subnet.subnet1.id    route\_table\_id = aws\_route\_table.routetable.id  } |



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

Task 9 : Launch Public and Private Instances in main.tf file

In this Task you will launch two Instances Public and Private in main.tf file

1. Firstly, We’ll create a security group for EC2 Instances to create a security group paste below code in main.tf file after the route table code

|  |
| --- |
| # Create Security group for EC2  resource "aws\_security\_group" "ec2sg" {    name        = "MyEC2Server\_SG"    description = "Security Group to allow traffic to EC2"    vpc\_id      = aws\_vpc.vpc.id    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"]    }    tags = {      Name = "whiz\_sg"    }  } |

1. Now to Launch  EC2 Instances add another block of code just below the security group code into the **main.tf** file

|  |
| --- |
| ############ Launching first EC2 Instance ############  resource "aws\_instance" "instance" {      ami = "ami-02e136e904f3da870"      instance\_type = "t2.micro"      vpc\_security\_group\_ids = ["${aws\_security\_group.ec2sg.id}"]      subnet\_id     = aws\_subnet.subnet1.id      associate\_public\_ip\_address = true      key\_name = "WhizKey"      tags = {          Name = "MyPublicEC2Server"      }      depends\_on = [aws\_security\_group.ec2sg]  }  ############ Launching second EC2 Instance ############  resource "aws\_instance" "instance2" {      ami = "ami-02e136e904f3da870"      instance\_type = "t2.micro"      vpc\_security\_group\_ids = ["${aws\_security\_group.ec2sg.id}"]      subnet\_id     = aws\_subnet.subnet2.id      associate\_public\_ip\_address = false      key\_name = "WhizKey"      tags = {          Name = "MyPrivateEC2Server"      }      depends\_on = [aws\_security\_group.ec2sg]  } |

1. This code will create two EC2 instances Public and Private with the specified configuration .The Public instance will have a public IP address, while the Private instance will not.
2. 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\_10122** 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.instance.id  }  output "ec2\_2" {    value       = aws\_instance.instance2.id  } |

1. In the above code, we will extract id’s of Public and Private EC2 Instances 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\_10122

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

**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

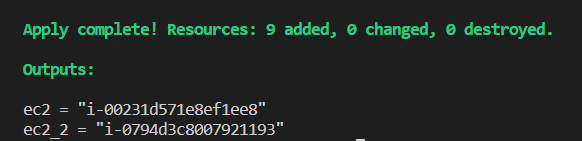
Text

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**.



1. **Note**: This process will take around 2-5 minutes.
2. The ID’s of Public and Private EC2 Instances Created by terraform will be visible there.

Task 13: Check the resources in AWS Console

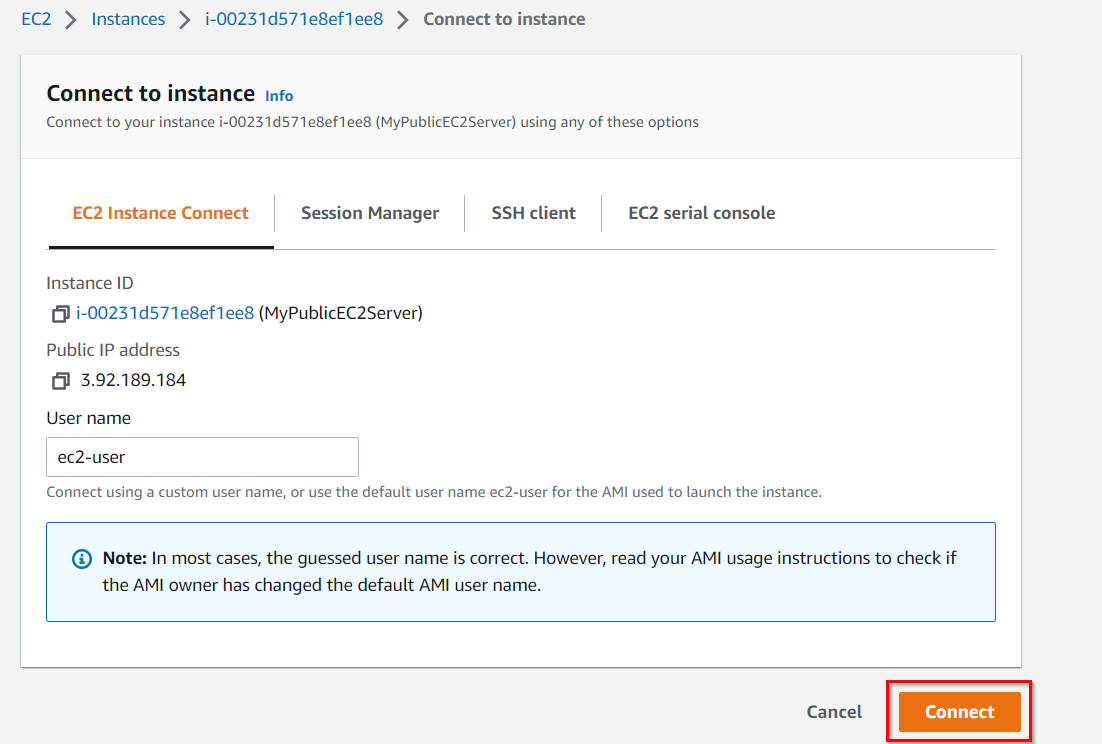
1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **EC2** by clicking on **Services** on the top, then click on **EC2** in the **Compute** section.
3. Click on the **Instances** on the left navigation panel. You can see the Instances created successfully.

Graphical user interface, text, application

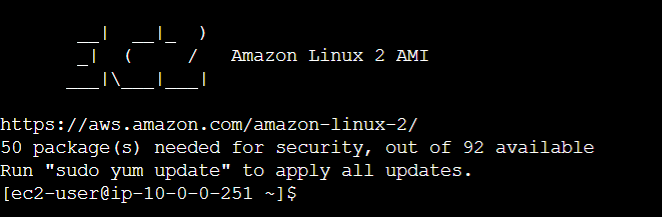
Description automatically generated

Task 14 : SSH into Public and Private EC2 Instance and Test Internet Connectivity

1. Navigate to the EC2 Dashboard and select the **MyPublicEC2Server**
2. Click on button to SSH into the EC2 instance
3. In Connect to Instance page select EC2 Instance Connect tab and click on **Connect** button



1. Once you SSH into the instance,follow the below steps.



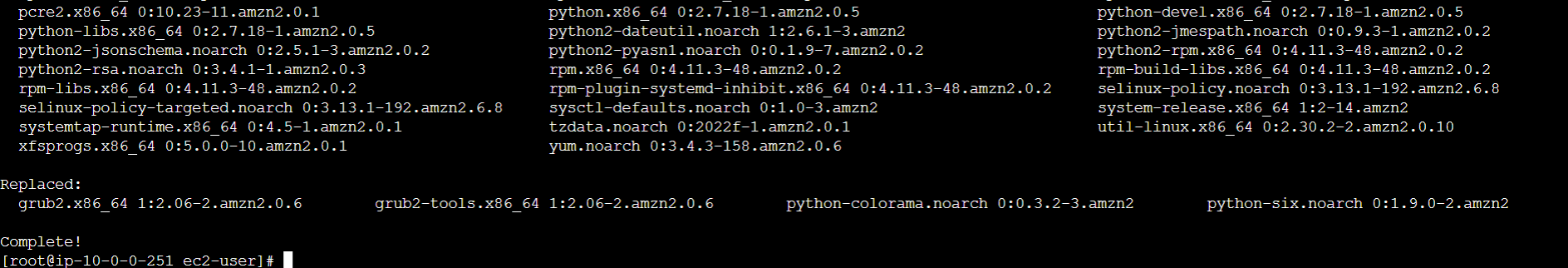
1. Switch to root user:

sudo su

1. Run the updates using the following command:

yum -y update

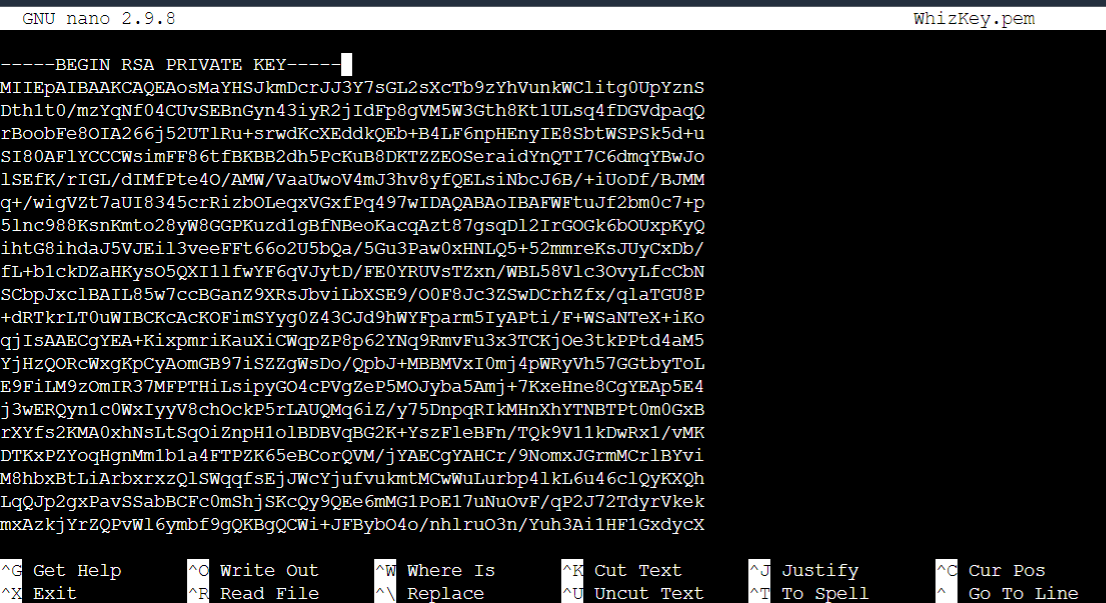
1. Since the Internet Gateway **MyIGW**is connected to **MyPublicSubnet**, updates will be completed successfully.



1. Let’s SSH into **MyPrivateEC2Server**from **MyPublicEC2Server**.
2. In order to SSH into **MyPrivateEC2Server***,* first, copy the data from our **WhizKey.pem** on the local machine. (which was downloaded earlier).
3. To copy the contents of the **MyKey.pem**, open the file in a text editor and copy the whole content.
4. To create the MyKey.pem in **MyPublicEC2Server**, run

nano WhizKey.pem

1. In the editor, copy and paste the key that looks similar to the example below:



1. Save the File
   * **Ctrl + X -->Y --> Enter**
   * **Save**
2. Check that the file was created correctly.

ls

**Text

Description automatically generated**

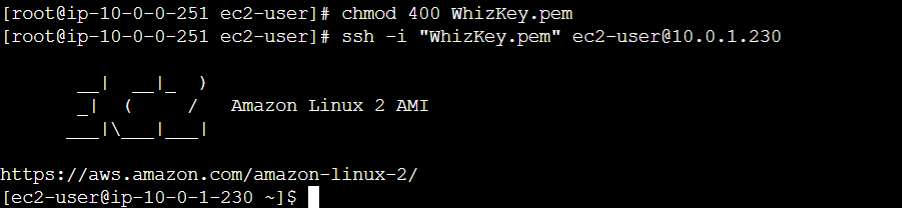
1. Update Permissions for the MyKey.pem

chmod 400 WhizKey.pem

1. Use the Private IP addressof **MyPrivateEC2Server** to SSH.

ssh -i WhizKey.pem ec2-user@<Private IP of Private Instance>

* **Note**: Incase if this message shows **Are you sure you want to continue connecting (yes/no)?** : Enter **yes**

****

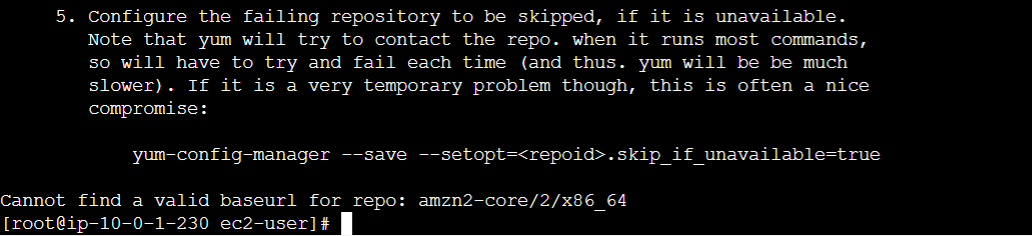
1. Switch to root user

sudo su

1. Run the updates using the following command:

yum -y update

1. Since no internet access is provided for EC2 instances in a private subnet, you will not be able to get updates. After some time, it will fail with the following message.

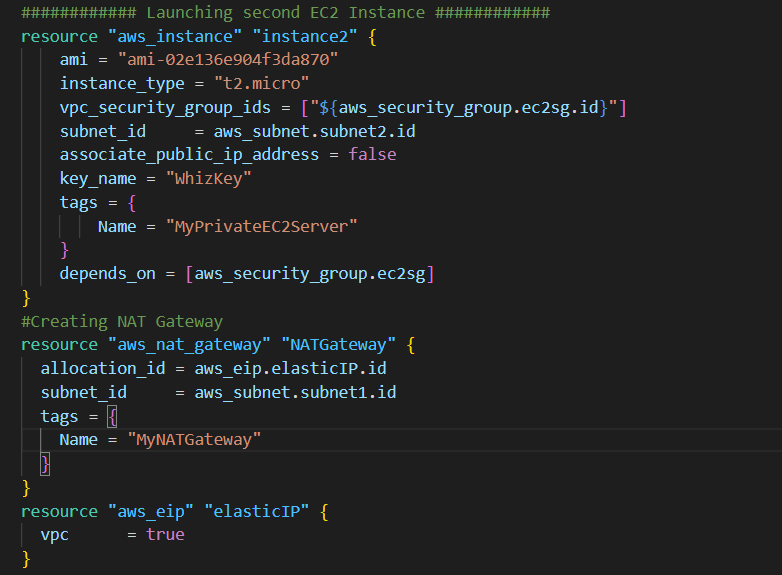
****

Task 15 : Create a NAT Gateway in main.tf file

In this task you will create a NAT Gateway that is used to provide internet access to the Instances inside a private subnet.

1. To create a NAT Gateway 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 Instance code

|  |
| --- |
| #Creating NAT Gateway  resource "aws\_nat\_gateway" "NATGateway" {    allocation\_id = aws\_eip.elasticIP.id    subnet\_id     = aws\_subnet.subnet1.id    tags = {      Name = "MyNATGateway"    }  }  resource "aws\_eip" "elasticIP" {    vpc      = true  } |



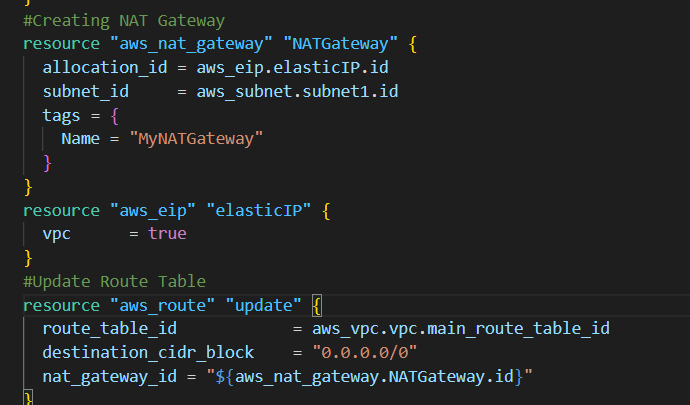
1. Save the file by pressing **Ctrl + S**.

Task 16 : Update Route table and configure NAT Gateway in main.tf file

In this task you will update the route table and add route to NAT Gateway

1. To Perform this add below code in mian.tf file after the NAt gateway code

|  |
| --- |
| #Update Route Table  resource "aws\_route" "update" {    route\_table\_id            = aws\_vpc.vpc.main\_route\_table\_id    destination\_cidr\_block    = "0.0.0.0/0"    nat\_gateway\_id = "${aws\_nat\_gateway.NATGateway.id}"  } |



1. Save the file by pressing **Ctrl + S**.
2. Now the Instances launched within **MyPrivateSubnet** can access the Internet through the NAT Gateway.
3. To modify instance type run the below command:

terraform apply

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

Graphical user interface, text

Description automatically generated

Task 17 : Test Internet connection from Instance inside Private Subnet

1. SSH back into **MyPublicEC2Server**.
2. Switch to root user

sudo su

1. SSH into **MyPrivateEC2Server**

ssh -i WhizKey.pem ec2-user@<Private IP of Private Instance>

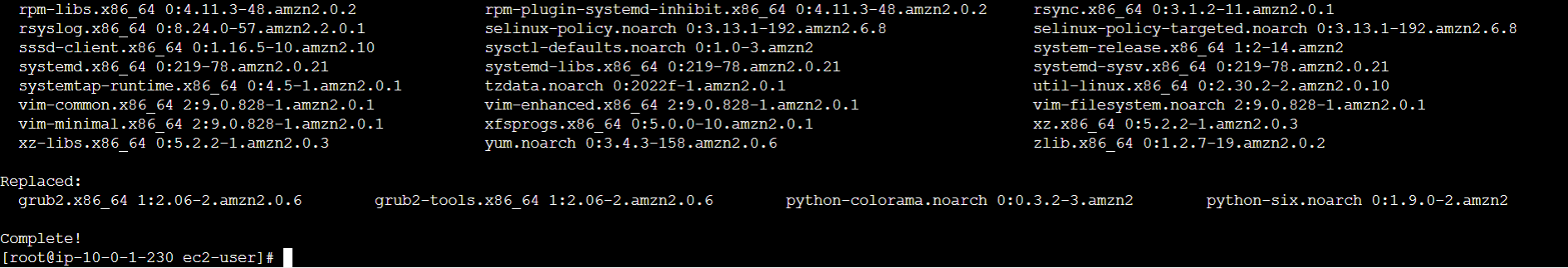
1. Switch to root user

sudo su

1. Run the updates using the following command:

yum -y update

1. You can see that the updates have been completed successfully in the terminal.



1. This shows that **MyPrivateEC2Server**has internet access.

Task 18: 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.
2. **NOTE**: If you are getting the error displayed below, ignore the error and run Destroy command again

A picture containing text

Description automatically generated

**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 create VPC with Public and Private Subnets using terraform
* You have created an Internet Gateway and configured a new route table using terraform
* You have successfully Launched Public and Private EC2 Instances 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 Tested Internet Access in Public EC2 Instance
* To provide Internet access to the EC2 instance in the Private subnet, you created a NAT Gateway and configured a Route table using terraform
* You have successfully Tested Internet Access in Private EC2 Instance
* You have succes