

Case Study 13

Basic Infrastructure Management with Terraform

Name: Aryan Deepak Saraf

Class: D15B

Roll no.: 53

- **Concepts Used:** Terraform, AWS EC2, and S3.
- **Problem Statement:** "Use Terraform to create an EC2 instance and an S3 bucket. Store the EC2 instance's IP address in the S3 bucket."
- **Tasks:**
 - Write a simple Terraform script to provision an EC2 instance and an S3 bucket.
 - Use Terraform outputs to extract the EC2 instance's IP address.
 - Store the IP address in a text file in the S3 bucket.

SOLUTION

Set up AWS Learner Lab for Terraform

Log into your **AWS Learner Lab**.

Go to **Details** of your session to view **Access Key**, **Secret Key**, and **Session Token**. These credentials will be used for authenticating with AWS services via Terraform.

For temporary credentials, such as those from AWS Learner Lab, use the following environment variables after configuring your profile:

```
export AWS_ACCESS_KEY_ID="your_access_key"
```

```
export AWS_SECRET_ACCESS_KEY="your_secret_key"
```

```
export AWS_SESSION_TOKEN="your_session_token"
```

The screenshot shows the AWS Learner Lab interface. The top bar includes a status indicator (green dot), a timer (03:53), and buttons for 'Start Lab', 'End Lab', 'AWS Details', 'Readme', and 'Reset'. Below the bar, it shows 'Used \$3.3 of \$50'. The main area is split into two panes. The left pane is a terminal window with the following commands and output:

```
eee_W_3413286@runweb141421:~$ export AWS_ACCESS_KEY_ID="ASIAT43BMNECTFPJHXRm"
eee_W_3413286@runweb141421:~$ export AWS_SECRET_ACCESS_KEY="MeaeeEsc9wFeV9nz879HJ/91A
+tpG20PBkxas/0Q"
eee_W_3413286@runweb141421:~$ export AWS_SESSION_TOKEN="IqoJb3JpZ21uX2VjECMaCXVzLXd1c
3QtMiJGMEQCFkZ5sImgSNAvY4wtuyNh877IrrqVKAMuMQGBuZSlqdkNAiBU46xNNYm1VqWn68HfkdYhRAk9cB
jpebSI5my2VOTZGcQ9AgI"
m8MtLbcaTvZhtmlK1DCg09hc1J1deMmZ18UnGMM29b6HcVpInZ7sPHa+bXFRd4EejLI0T5JoML7oad0TtIH6
eFl+reWIothBgCfXt2qczPVjwNMZy6mA5pFmzUPFcAGRY7iuiimG1KLCHTIPmsIIVc+H6IR5DoXYZmNuUuVGyH
sJjmbMUBve1k0x9TjKqwbBtJMRa6HY2mo6wmiG+9+SpL2sUTUQme9b2J410yHUBq9HQQ6zndwzb6Et8dcrIRu
G2G2KatNY/ybWZspt3d9JeSzFux+1M+eKXsh9x8jru32IltT/VDi7xE0GA0Fg4JVLda5EqS4pQ7hm0rUmvrU
9o1TsFrCwxGRcyGwJMMrr2LgGOp4Bj1kn1uj1S3+1jSHYu/NwB0z5LwFFnL6KiF5xkIWb9Qu6pYLDAs6W2KHr
Arvb63LXKFTU5nhrQ8i65oGsR+ig10qVUVZNTQ9L8nROTxdexCsIy1XA+peW7dvf0IEhWLPgVv+BwdeCp4n+d
dkGpRU1bsUtyXnp8F7VhExcGJ0b3QqyTcWAtyMcVtW80eUrxefZrVJdKpJ2bBRUVb127I="
eee_W_3413286@runweb141421:~$
```

The right pane is titled 'Cloud Access' and contains the following text:

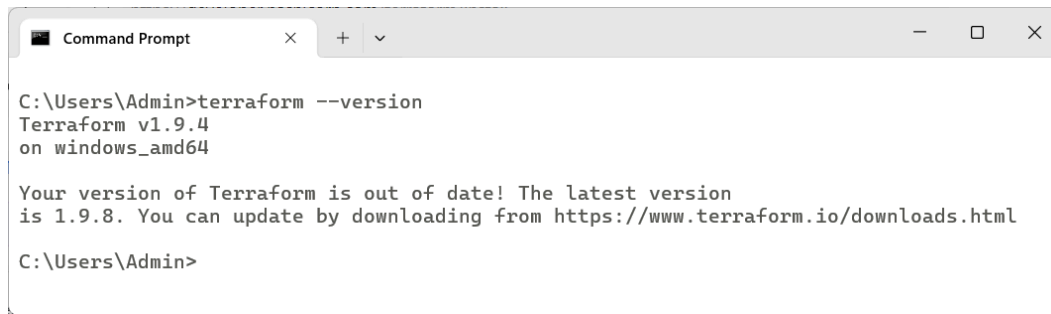
AWS CLI:
Copy and paste the following into
~/.aws/credentials

```
[default]
aws_access_key_id=ASIAT43BMNECT
PJHXRm
aws_secret_access_key=MeaeeEsc9
FeV9nz879HJ/91A+tpG20PBkxas/0Q
aws_session_token=IqoJb3JpZ21uX
VjECMaCXVzLXd1c3QtMiJGMEQCFkZ5
ImgSNAvY4wtuyNh877IrrqVKAMuMQGBu
SlqdkNAiBU46xNNYm1VqWn68HfkdYhR.
k9cBjpebSI5my2VOTZGcQ9AgI
```

Install Terraform on Your Local System

If you haven't installed Terraform, follow the instructions below:

- Download Terraform: Terraform Download.
- Install Terraform and verify installation by running `terraform --version`.



```
Command Prompt
C:\Users\Admin>terraform --version
Terraform v1.9.4
on windows_amd64

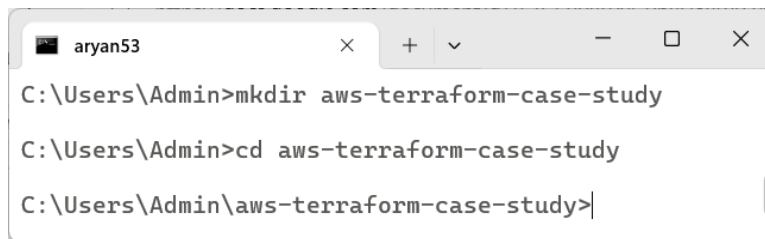
Your version of Terraform is out of date! The latest version
is 1.9.8. You can update by downloading from https://www.terraform.io/downloads.html
C:\Users\Admin>
```

Set up Terraform Project Folder

Create a project folder and navigate to it:

`mkdir aws-terraform-case-study`

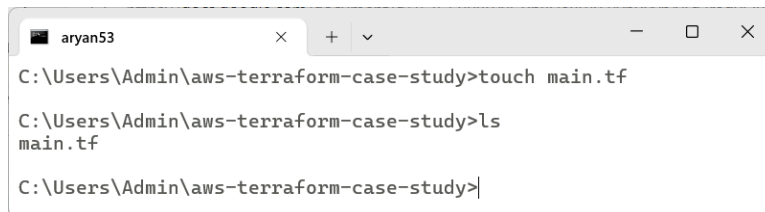
`cd aws-terraform-case-study`



```
aryan53
C:\Users\Admin>mkdir aws-terraform-case-study
C:\Users\Admin>cd aws-terraform-case-study
C:\Users\Admin\aws-terraform-case-study>
```

Create a Terraform configuration file (`main.tf`):

`touch main.tf`



```
aryan53
C:\Users\Admin\aws-terraform-case-study>touch main.tf
C:\Users\Admin\aws-terraform-case-study>ls
main.tf
C:\Users\Admin\aws-terraform-case-study>
```

Write the Terraform Configuration (`main.tf`)

```
provider "aws" {
  region    = "us-east-1"
  access_key = "ASIAT43BMNECTFPJHXRm"
  secret_key = "MeaeE9wFeV9nz879HJ/91A+tpG2OPBkaxs/0Q"
  token     =
"IQoJb3JpZ2luX2VjECMaCXVzLXdlc3QtMiJGMEQCIFkZ5xImgSNAvY4wtuyNh877IrkVKAMuMQ
GBuzSlqdkNAiBU46xNNYM1VqWN68HfkdYhRAk9cBjpebSI5my2VOTZGCq9AgiM////////8BEAEa
DDI2ODEwMjgyMjE0OSIMxekfF09zUH3KbgowKpEC8YcGvAI8MtLbcaTvZhtm1KIDCg09hc1Jlde
MNmZ18UnGMM29b6HcVplnZ7sPHa+bXfRd4EejLI0T5joML7oaD0TtIH6eFI+reWlotHbGcFXt2qczP
VjwNMZy6mA5pFmzUPFcAGRY7iuiMGIKLCHTIPms1IVc+H6IR5DoXYZmNuUuVGyHsjimBMUB
ve1k0x9TjkqwbBtJMRa6HY2mo6wmiG+9+SpL2sUTUQme9b2J4l0yHU8q9HQQ6zndwzb6Et8dcRiRuG
2G2KatNY/ybWXZspt3d9JeSzfUx+IM+eKXsh9x8jru32IltT/VDi7xE06A0fG4JVLda5EqS4pQ7hm0rU
mvrU9o1TsFrCwxGRCyGWJMMrr2LgGOp4Bjln1uj1S3+lJSHYu/NwBOz5LwfFnL6KiF5xkiWb9Qu6
pYLdAs6W2KHrArvb63LXKfTUsnhrQ8i65oGsR+iG10qVUVZNTQ9L8nROTXdexCsIylXA+pew7dvf
```

```
0IEhWLpGVv+BwdeCp4n+ddkGpRUibsUtYxnp8F7VhExcGJ0b3QqyTcWAtyMCvTMW80eUrxeFZrV
JdKpJZbBRUVbl27I="
}

# Create an S3 bucket
resource "aws_s3_bucket" "aryan53_case_study_bucket" {
  bucket = "my-ec2-ip-bucket-${random_id.bucket_id.hex}"
}

resource "random_id" "bucket_id" {
  byte_length = 4
}

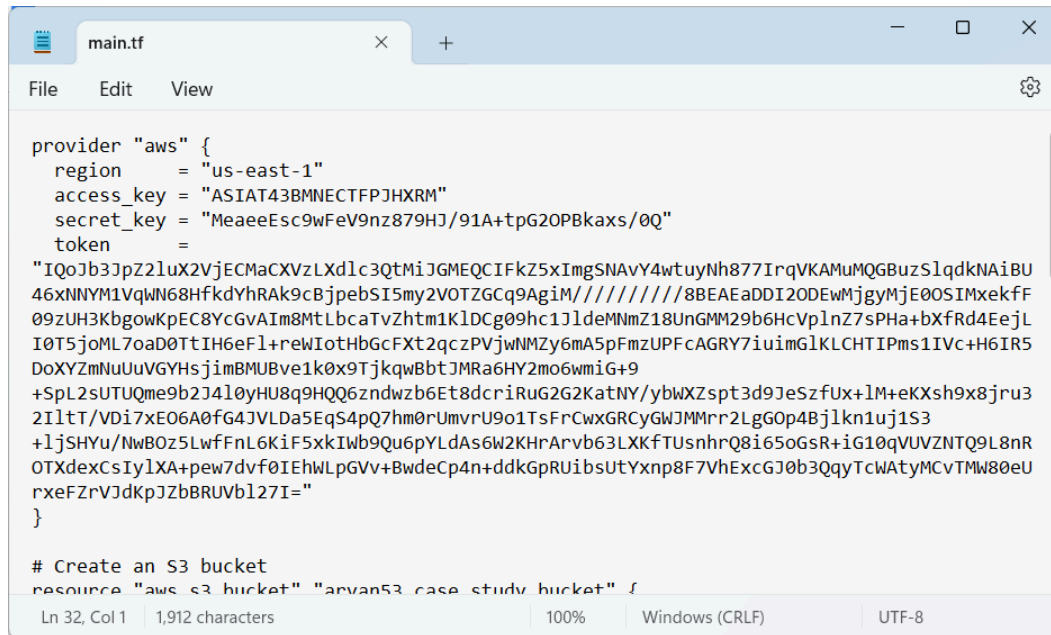
# Create a local file with the EC2 instance's IP address
resource "local_file" "ec2_ip_file" {
  content = aws_instance.aryan_instance.public_ip
  filename = "ec2_instance_ip.txt"
}

# Upload the EC2 instance IP to the S3 bucket using aws_s3_object
resource "aws_s3_object" "ec2_ip_object" {
  bucket = aws_s3_bucket.aryan53_case_study_bucket.bucket
  key    = "ec2_instance_ip.txt"
  source = local_file.ec2_ip_file.filename
  acl    = "private"
}

# Create an EC2 instance
resource "aws_instance" "aryan_instance" {
  ami          = "ami-06b21ccaeff8cd686" # Amazon Linux 2 AMI
  instance_type = "t2.micro"

  tags = {
    Name = "Terraform-EC2"
  }
}

# Output the public IP of the EC2 instance
output "instance_public_ip" {
  value = aws_instance.aryan_instance.public_ip
}
```



```

provider "aws" {
  region      = "us-east-1"
  access_key  = "ASIAT43BMNECTFPJHXRm"
  secret_key  = "MeaeEeEsc9wFeV9nz879HJ/91A+tpG20PBkaxs/0Q"
  token       =
  "IQoJb3JpZ2luX2VjECMaCXVzLXdlc3Q0tMiJGMEQCIFkZ5xImgSNAvY4wtuyNh877IrrqVKAMuMQGBuzSlqdkNAiBU
  46xNNYm1VqWN68HfkdYhRAK9cBjpebSI5my2VOTZGCq9Agim////////8BEAEaDDI20DEwMjgyMjE0SImxekfF
  09zUH3KbgowKpEC8YcGvAIm8MtLbcaTvZhtm1KlDCg09hc1JldeMNMZ18UnGMM29b6HcVplnZ7sPHA+bXfRd4EejL
  I0T5j0mL7oaD0TtIH6eFl+reWIotHbGcFxt2qcZPVjwNMZY6mA5pFmzUPFcAGRY7iuiMGkLCHTIPms1IVc+H6IR5
  DoXYZmNuUuVGyHsjimBMUBve1k0x9TjkqWbBtJMRa6HY2mo6wmig+9
  +Spl2sUTUQme9b2J4l0yHU8q9HQQ6zndwzb6Et8dcRiRuG2G2KatNY/ybWXXZspt3d9JeSzfUx+lm+eKXsh9x8jru3
  2IltT/VDi7xE06A0f64JVLda5EqS4pQ7hm0rUmvru9o1TsFrCwxGRCyGWJMMrr2LgG0p4Bjln1uj1S3
  +ljSHYu/NwB0z5LwffnL6KiF5xkIwb9Qu6pYLDAs6W2KhrArvb63LXKftUsnhrQ8i65oGsR+iG10qVUVZNTQ9L8nR
  OTXdexCsIylXA+pew7dvf0IEhWLPgVv+BwdeCp4n+ddkGpRUibSutYxnp8F7VhExcGJ0b3QqyTcWAtyMCvTMW80eU
  rxeFZrVJdKpJZbBRUVbl27I="
}

# Create an S3 bucket
resource "aws_s3_bucket" "aryanS3_case_study_bucket" {

```

Key parts of the configuration:

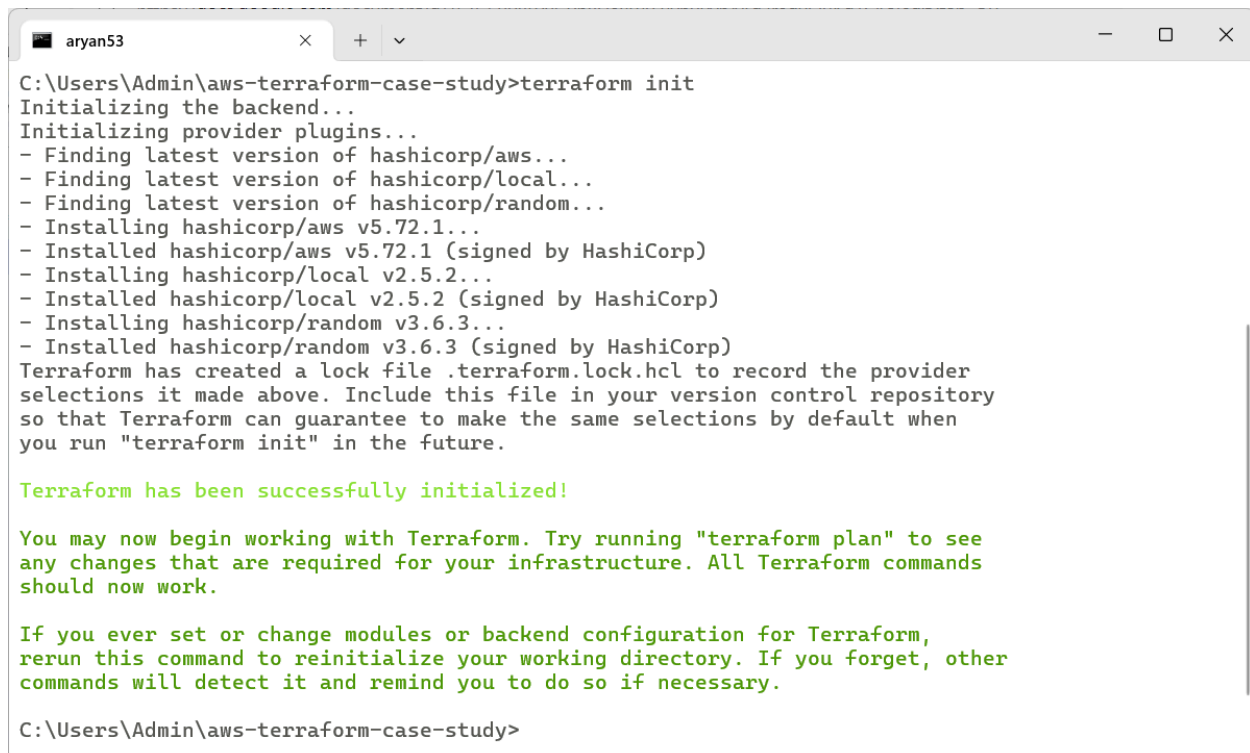
- **Provider Block:** Connects Terraform to AWS.
- **EC2 Instance:** Provisions a basic `t2.micro` instance using the specified AMI (Amazon Linux 2).
- **S3 Bucket:** Creates an S3 bucket with a unique name.
- **Local File:** Saves the public IP of the EC2 instance into a text file (`ec2_instance_ip.txt`).
- **S3 Object:** Uploads the text file containing the EC2 instance's IP address to the S3 bucket.

Initialize and Deploy with Terraform

a. Initialize the Terraform Working Directory

Before applying the configuration, initialize the directory:

`terraform init`



```

C:\Users\Admin\aws-terraform-case-study>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/random...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
- Installing hashicorp/local v2.5.2...
- Installed hashicorp/local v2.5.2 (signed by HashiCorp)
- Installing hashicorp/random v3.6.3...
- Installed hashicorp/random v3.6.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\Admin\aws-terraform-case-study>

```

This command downloads the necessary provider plugins (AWS in this case).

b. Generate a Plan

To preview the actions that Terraform will take:

`terraform plan`

```
aryan53
C:\Users\Admin\aws-terraform-case-study>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource
actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.aryan_instance will be created
+ resource "aws_instance" "aryan_instance" {
+   ami                    = "ami-0c55b159cbfafa1f0"
+   arn                    = (known after apply)
+   associate_public_ip_address = (known after apply)
+   availability_zone       = (known after apply)
+   cpu_core_count          = (known after apply)
+   cpu_threads_per_core    = (known after apply)
+   disable_api_stop        = (known after apply)
+   disable_api_termination = (known after apply)
+   ebs_optimized           = (known after apply)
```

```
aryan53
+ byte_length = 4
+ dec         = (known after apply)
+ hex         = (known after apply)
+ id          = (known after apply)
}

Plan: 5 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ instance_public_ip = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to t
ake
exactly these actions if you run "terraform apply" now.

C:\Users\Admin\aws-terraform-case-study>
```

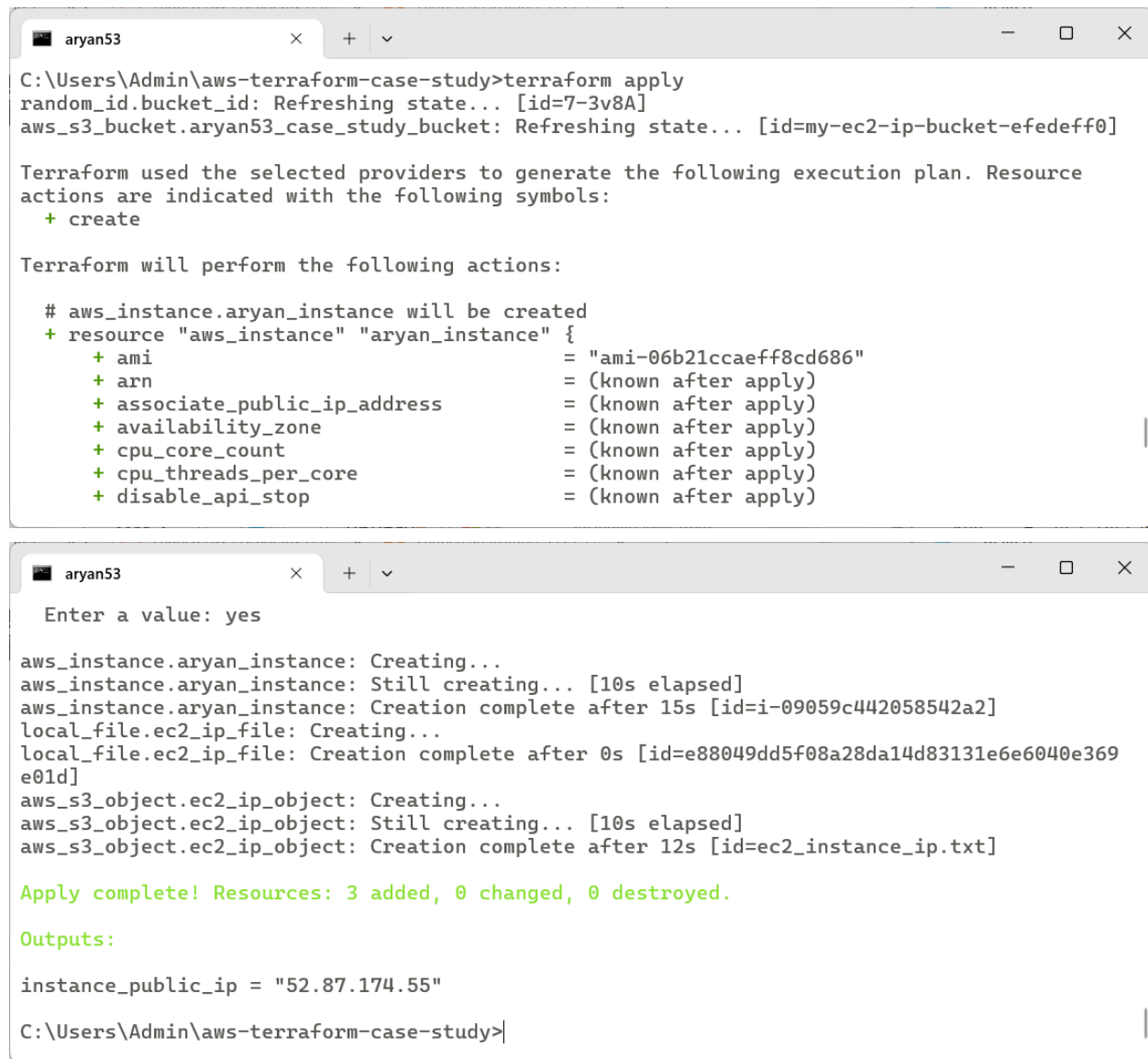
It will show you the resources Terraform plans to create.

c. Apply the Terraform Configuration

Run the following command to provision the EC2 instance and S3 bucket:

`terraform apply`

Type `yes` when prompted to confirm.



The first screenshot shows the execution of the Terraform apply command. It displays the state refresh for 'random_id.bucket_id' and 'aws_s3_bucket.aryan53_case_study_bucket'. It then shows the execution plan, indicating that an AWS instance will be created with various attributes like ami, arn, and public IP address. The second screenshot shows the successful completion of the apply command. It lists the resources created: 'aws_instance.aryan_instance', 'local_file.ec2_ip_file', and 'aws_s3_object.ec2_ip_object'. It also shows the public IP address of the instance: '52.87.174.55'.

```
C:\Users\Admin\aws-terraform-case-study>terraform apply
random_id.bucket_id: Refreshing state... [id=7-3v8A]
aws_s3_bucket.aryan53_case_study_bucket: Refreshing state... [id=my-ec2-ip-bucket-efedeff0]

Terraform used the selected providers to generate the following execution plan. Resource
actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.aryan_instance will be created
+ resource "aws_instance" "aryan_instance" {
  + ami                  = "ami-06b21ccaef8cd686"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count       = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop     = (known after apply)

Enter a value: yes

aws_instance.aryan_instance: Creating...
aws_instance.aryan_instance: Still creating... [10s elapsed]
aws_instance.aryan_instance: Creation complete after 15s [id=i-09059c442058542a2]
local_file.ec2_ip_file: Creating...
local_file.ec2_ip_file: Creation complete after 0s [id=e88049dd5f08a28da14d83131e6e6040e369e01d]
aws_s3_object.ec2_ip_object: Creating...
aws_s3_object.ec2_ip_object: Still creating... [10s elapsed]
aws_s3_object.ec2_ip_object: Creation complete after 12s [id=ec2_instance_ip.txt]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

instance_public_ip = "52.87.174.55"

C:\Users\Admin\aws-terraform-case-study>
```

d. Verify the IP Address in S3

- Once the infrastructure is created, the EC2 instance's IP address will be saved in the file `ec2_instance_ip.txt` and uploaded to the S3 bucket.

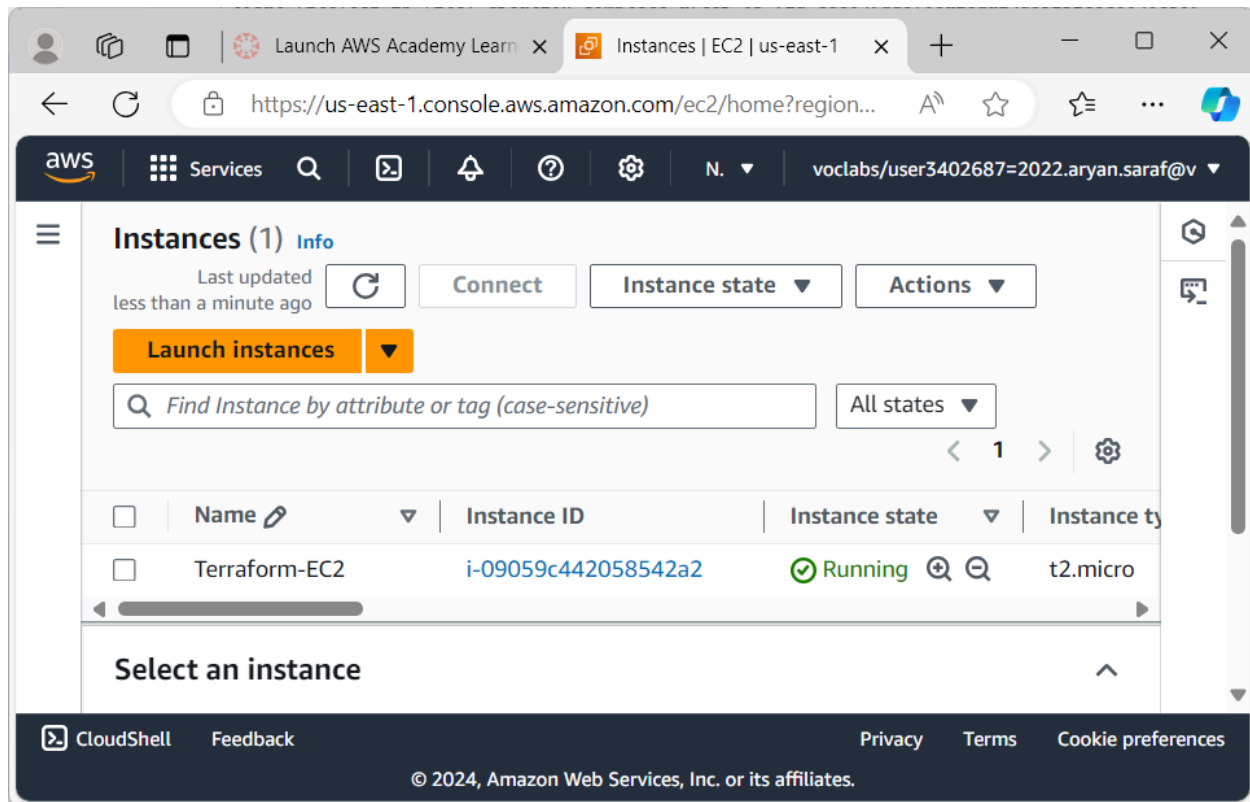
Verify the Results

After the Terraform script has completed, verify the following:

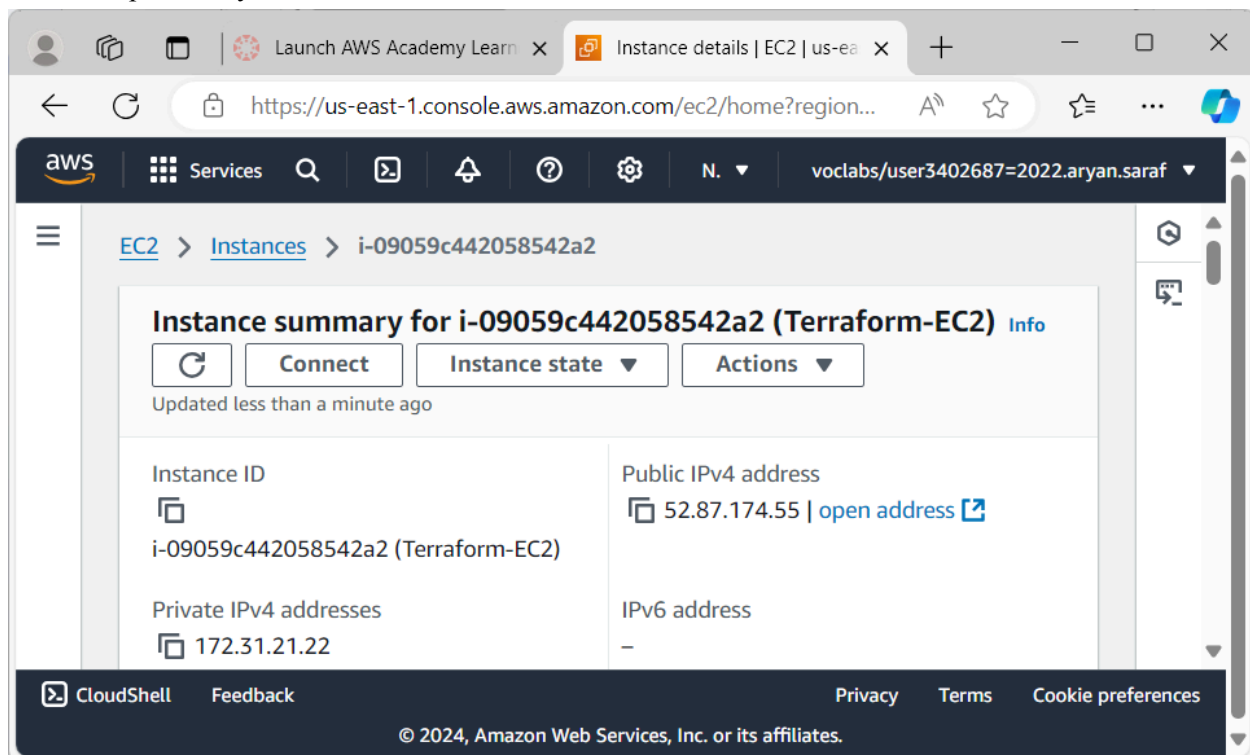
Check the S3 bucket: Open the AWS Console and navigate to the S3 service. Verify that the EC2 instance's public IP is stored in the `ec2_instance_ip.txt` file.

(You can also use the output shown in the Terraform terminal to verify the public IP.)

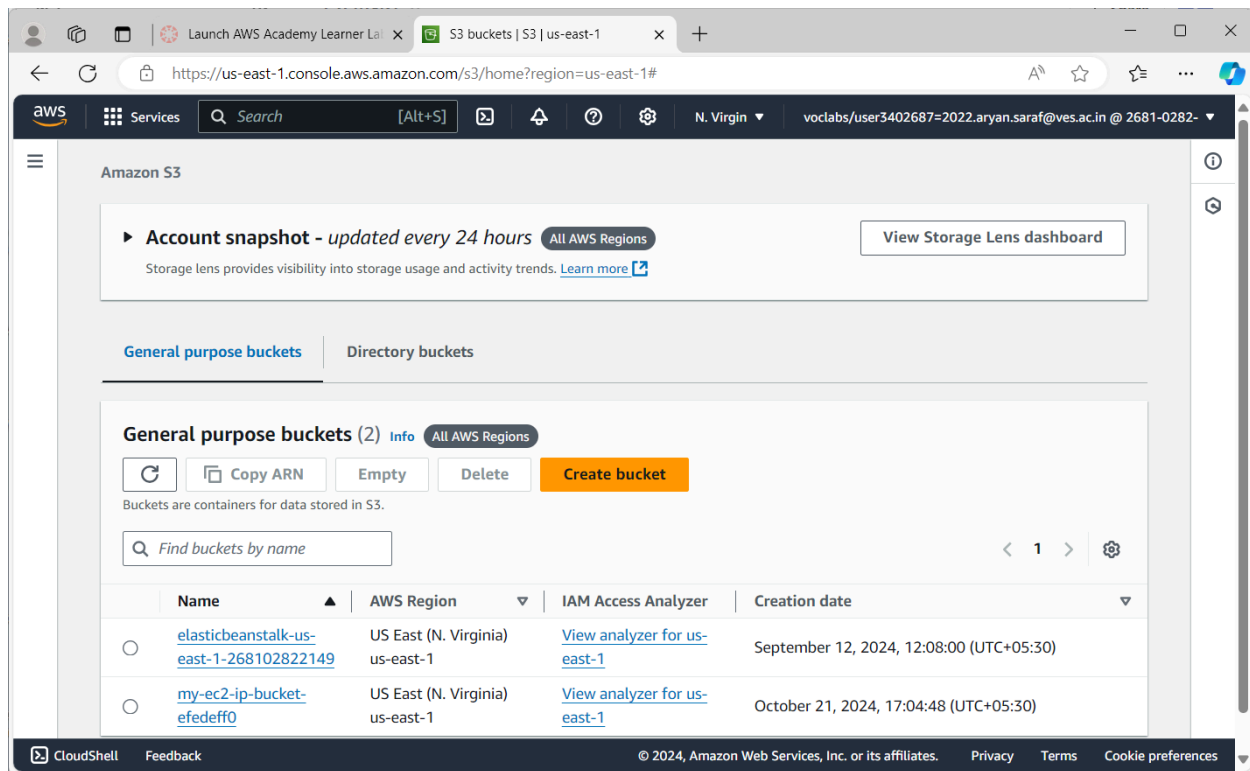
Go to the EC2 instances and check whether our specified instance through terraform is same or not.



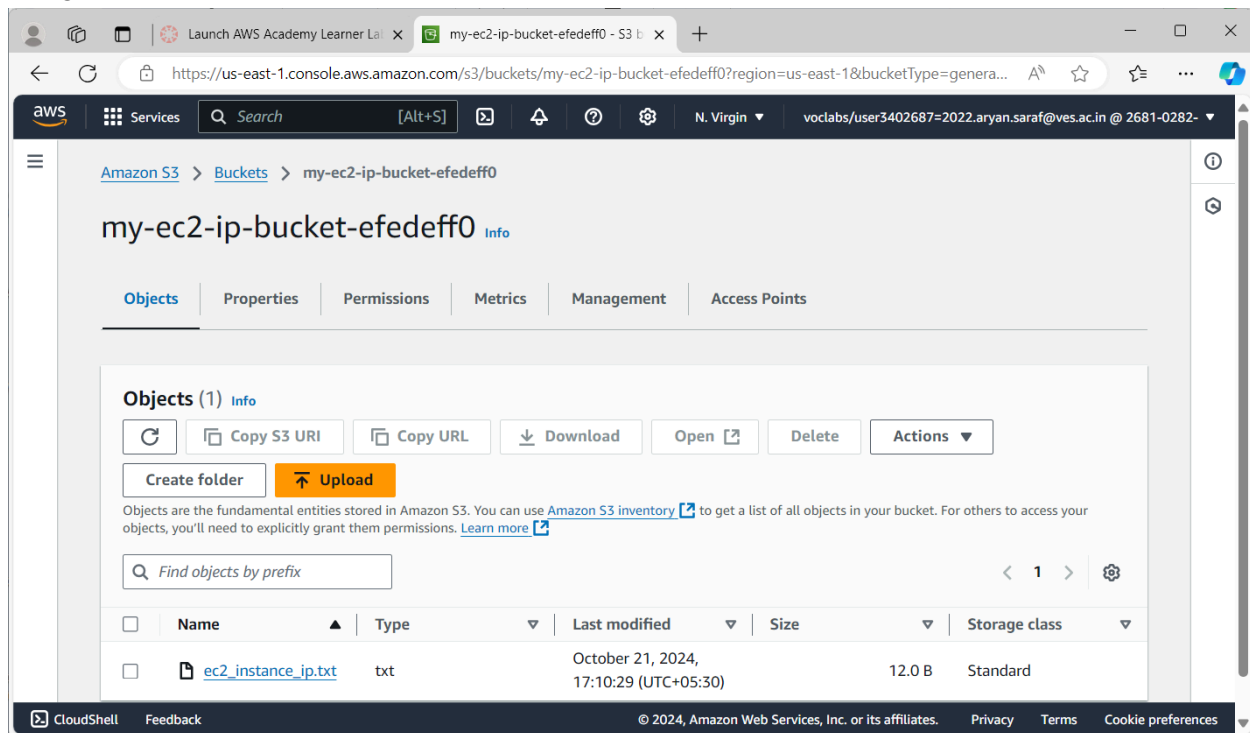
Check the public key of the EC2 instance.



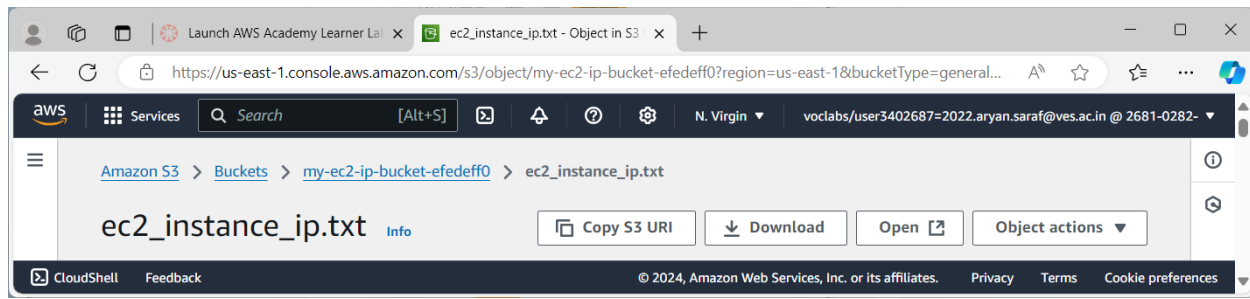
Go to the s3 bucket and check whether our specified s3 bucket through terraform is the same or not.



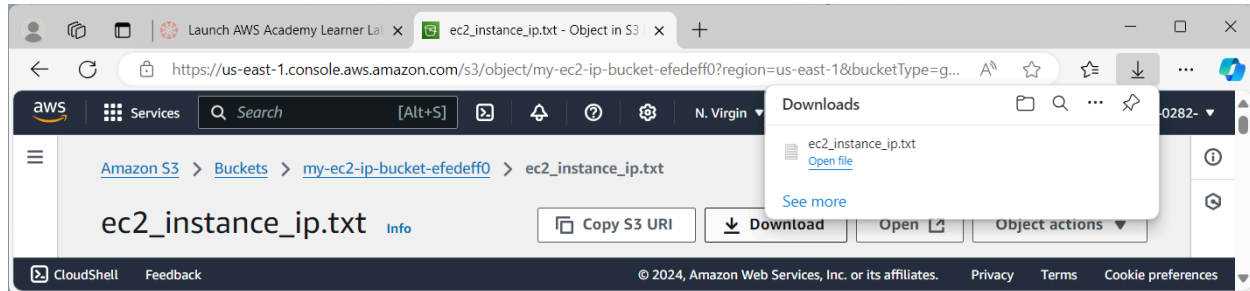
Navigate inside the s3 bucket in which there will be one txt file.



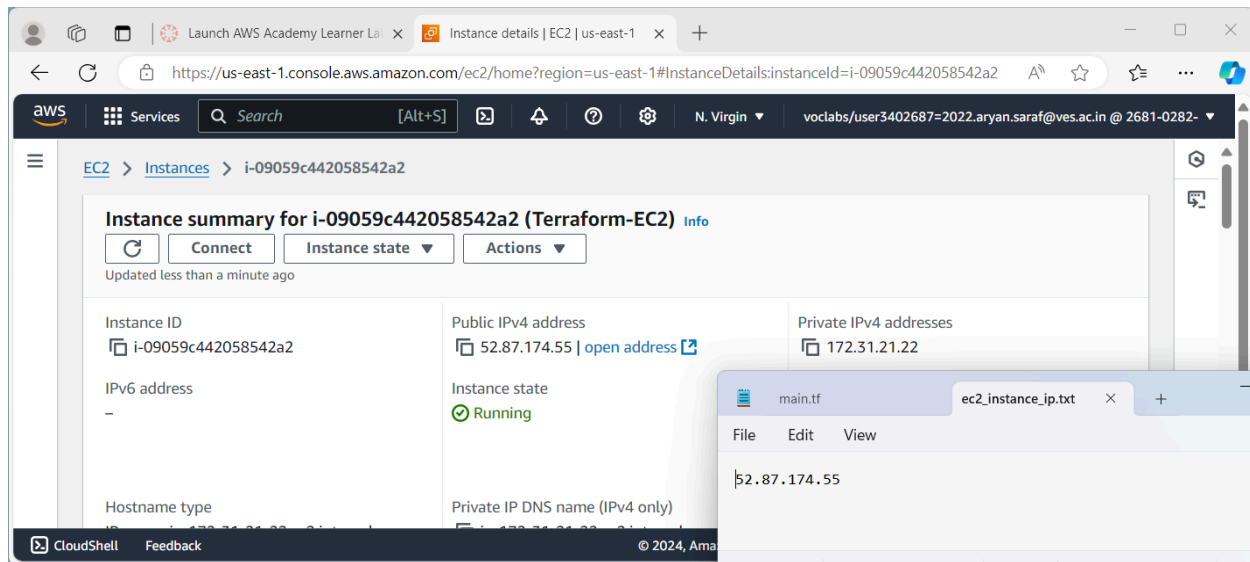
Open txt file.



Download the txt file.



Match the public ip address of the ec2 instance with the ip address in the txt file.



Clean Up Resources

Once you're done with the lab, ensure to delete the resources to avoid any potential charges:

`terraform destroy`

Type `yes` to confirm.

```

C:\Users\Admin\aws-terraform-case-study>terraform destroy
random_id.bucket_id: Refreshing state... [id=7-3v8A]
aws_s3_bucket.aryan53_case_study_bucket: Refreshing state... [id=my-ec2-ip-bucket-efedeff0]
aws_instance.aryan_instance: Refreshing state... [id=i-09059c442058542a2]
local_file.ec2_ip_file: Refreshing state... [id=e88049dd5f08a28da14d83131e6e6040e369e01d]
aws_s3_object.ec2_ip_object: Refreshing state... [id=ec2_instance_ip.txt]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

# aws_instance.aryan_instance will be destroyed
- resource "aws_instance" "aryan_instance" {
  - ami                    = "ami-06b21ccaeff8cd686" -> null
  - arn                    = "arn:aws:ec2:us-east-1:268102822149:instance/i-09059c442058542a2" -> null
  - associate_public_ip_address = true -> null
  - availability_zone         = "us-east-1a" -> null
  - cpu_core_count            = 1 -> null
  - cpu_threads_per_core      = 1 -> null
}

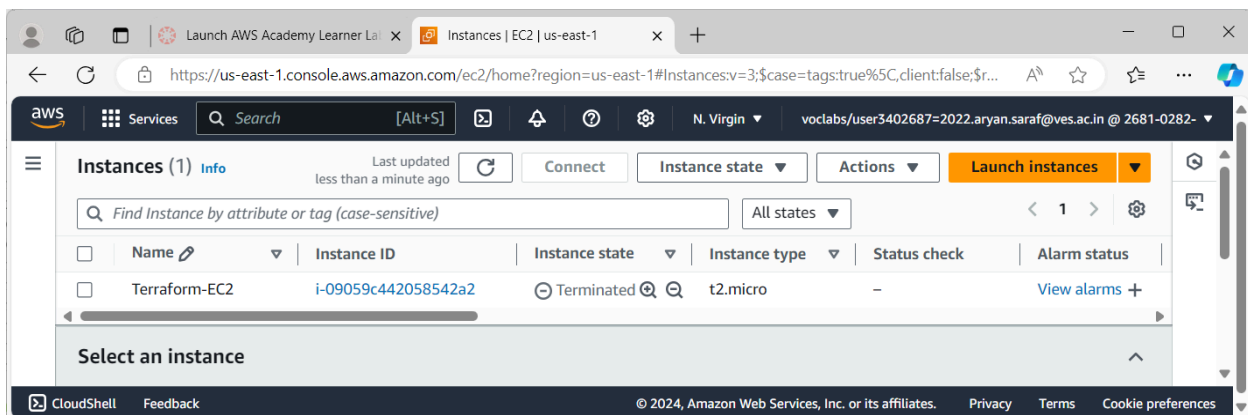
aws_s3_object.ec2_ip_object: Destroying... [id=ec2_instance_ip.txt]
aws_s3_object.ec2_ip_object: Destruction complete after 1s
local_file.ec2_ip_file: Destroying... [id=e88049dd5f08a28da14d83131e6e6040e369e01d]
local_file.ec2_ip_file: Destruction complete after 0s
aws_s3_bucket.aryan53_case_study_bucket: Destroying... [id=my-ec2-ip-bucket-efedeff0]
aws_s3_bucket.aryan53_case_study_bucket: Destruction complete after 1s
aws_instance.aryan_instance: Destroying... [id=i-09059c442058542a2]
aws_s3_bucket.aryan53_case_study_bucket: Destruction complete after 1s
random_id.bucket_id: Destroying... [id=7-3v8A]
random_id.bucket_id: Destruction complete after 0s
aws_instance.aryan_instance: Still destroying... [id=i-09059c442058542a2, 10s elapsed]
aws_instance.aryan_instance: Still destroying... [id=i-09059c442058542a2, 20s elapsed]
aws_instance.aryan_instance: Still destroying... [id=i-09059c442058542a2, 30s elapsed]
aws_instance.aryan_instance: Still destroying... [id=i-09059c442058542a2, 40s elapsed]
aws_instance.aryan_instance: Still destroying... [id=i-09059c442058542a2, 50s elapsed]
aws_instance.aryan_instance: Destruction complete after 52s

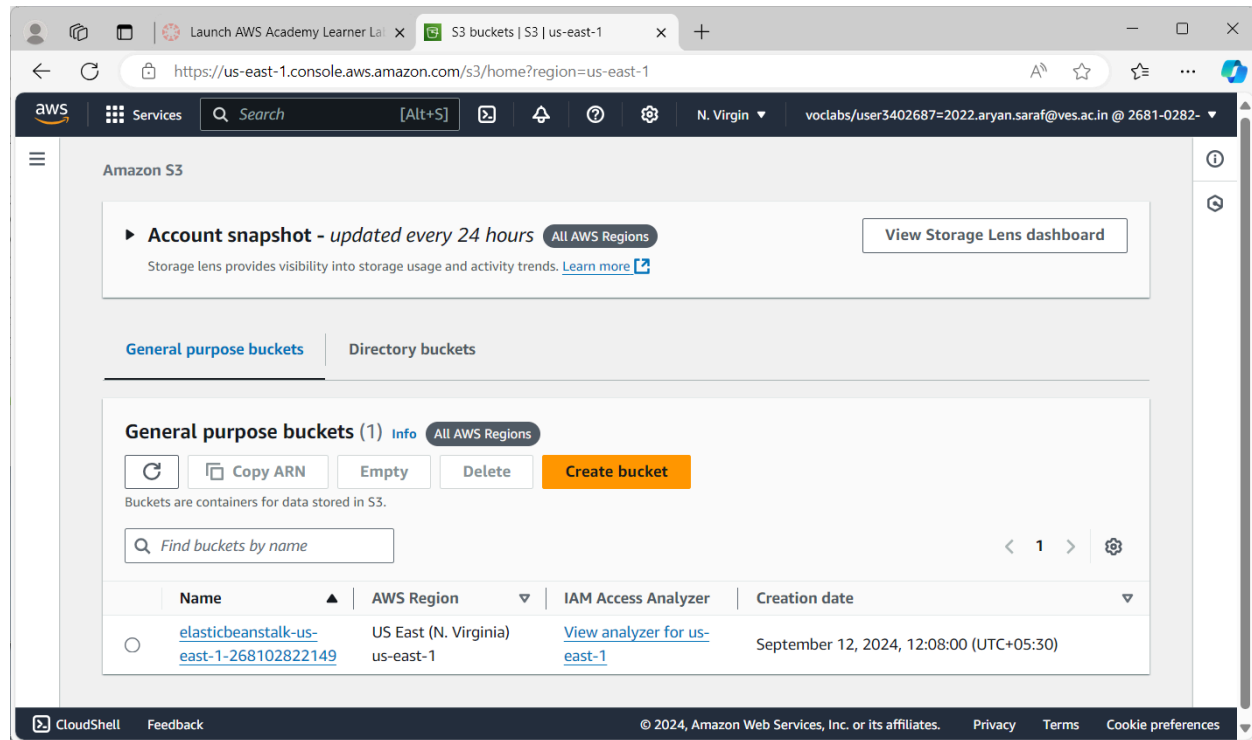
Destroy complete! Resources: 5 destroyed.

C:\Users\Admin\aws-terraform-case-study>

```

Check whether the resources are cleared or not.





Conclusion

This experiment covers important topics related to cloud infrastructure management, automation, and provisioning using Terraform. It emphasizes key AWS services like EC2 and S3, showing how infrastructure can be efficiently automated and deployed at scale.