Name:- Arnav .S. Sawant D15A Roll-no:- 53

Basic Infrastructure Management with Terraform

Introduction:-

Case Study Overview

This case study focuses on the automation of infrastructure provisioning using Terraform on Amazon Web Services (AWS). Specifically, it involves creating an EC2 (Elastic Compute Cloud) instance and an S3 (Simple Storage Service) bucket, while also demonstrating how to store the EC2 instance's IP address in the S3 bucket. This approach not only streamlines resource management but also enhances reproducibility and version control of the infrastructure.

Key Feature and Application

The key feature of this case study is the use of Terraform's Infrastructure as Code (IaC) capabilities. Terraform allows users to define infrastructure in a declarative manner, enabling version control, easier collaboration, and automated deployments. This practical use case exemplifies how teams can manage cloud resources efficiently and effectively without the need for manual intervention, minimizing human error.

Step-by-Step Explanation:-

Step 1: Initial Setup

1. **Prerequisites:**
    - Ensure that Terraform is installed on your local machine. You can download it from the official Terraform website.
    - Set up an AWS account if you don't have one already.
    - Configure your AWS credentials using the AWS CLI.

Create a Project Directory: Open your terminal and create a directory for your Terraform project:

```
mkdir terraform-aws-ec2-s3
cd terraform-aws-ec2-s3
```

Or **you can also make 2 separate folders for EC2 and S3**

Step 2: Write Terraform Configuration

Name:- Arnav .S. Sawant D15A Roll-no:- 53

1. **Create a `main.tf` File in aws-ec2 folder:** Create a file named `main.tf` in your project directory and add the following Terraform configuration:
   Code:-
   ```
   resource "aws_instance" "myserver" {
      ami = "ami-0866a3c8686eaeeba"
      instance_type = "t2.micro"
      tags = {
         Name = "sample server ad-dev"
      }
   }
   ```

   Same way create an output.tf and variables.tf in the **aws-ec2 folder**
   Code:- #output.tf
   ```
   output "ec2_public_ip" {
      value = aws_instance.myserver.public_ip
   }

   output "ec2_instance_type" {
      value = aws_instance.myserver.instance_type
   }

   #variables.tf
   variable "instance_type" {
      description = "value"
      type = string
      default = "t2.micro"
   }
   ```

2. **Create a `main.tf` File in aws-s3 folder:** Create a file named `main.tf` in your project directory and add the following Terraform configuration:
   Code:-
   ```
   terraform {
      required_providers {
      aws = {
         source = "hashicorp/aws"
         version = "5.64.0"
         }
      }
   }

   resource "aws_s3_bucket"
   "demo-bucket-advdevops-practical-2024-sawant-53-d15a" {
      bucket = "demo-bucket-advdevops-practical-2024-sawant-53-d15a"
   }
   ```

```
resource "aws_s3_object" "bucket-data" {
    bucket =
aws_s3_bucket.demo-bucket-advdevops-practical-2024-sawant-53-d15a.bucket
    source = "./myfile.txt"
    key = "newfile.txt"
}
```

Step 3: Initialize Terraform

Run the following command to initialize Terraform, which downloads the necessary provider plugins:

```
terraform init
```

Step 4: Apply the Configuration

Apply the configuration to create the resources defined in `main.tf`:

```
terraform apply
```

You will see a prompt to confirm the changes; type `yes` to proceed. Terraform will provision the EC2 instance and S3 bucket.

Step 5: Verify the Resources

After the provisioning completes, you can verify that the EC2 instance and S3 bucket have been created successfully in the AWS Management Console.

Step 6: Output the EC2 IP Address

The IP address of the EC2 instance will be displayed in the terminal as per the output defined in `main.tf`. You can also check the contents of the S3 bucket to see the `newfile.txt` file containing the IP address.

Guidelines:-

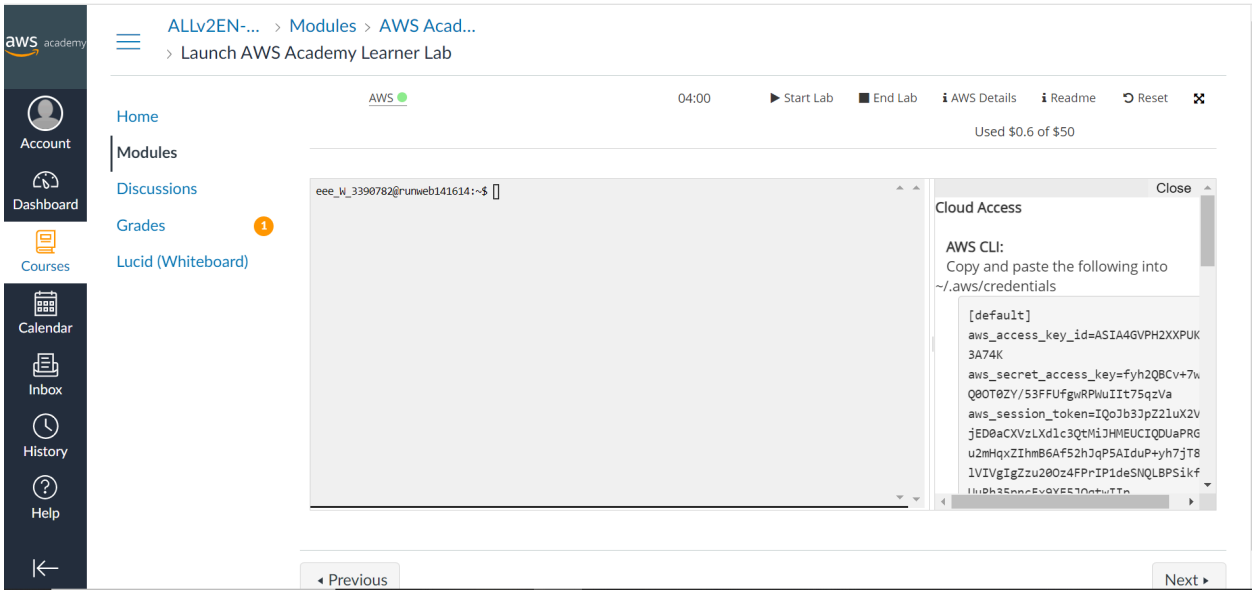
- **Best Practices:**
  - Use version control (e.g., Git) for managing your Terraform code.
  - Regularly check your AWS resource usage to avoid unexpected costs.
  - Tag your resources for better organization and management.
- **Common Errors:**
  - Ensure that the S3 bucket name is unique globally.

Name:- Arnav .S. Sawant D15A Roll-no:- 53

- Validate your Terraform configuration using `terraform validate` before applying or you figure out the error in “terraform plan”
- Make sure to check official documentation so that it would be easy to write refer it and write the code.

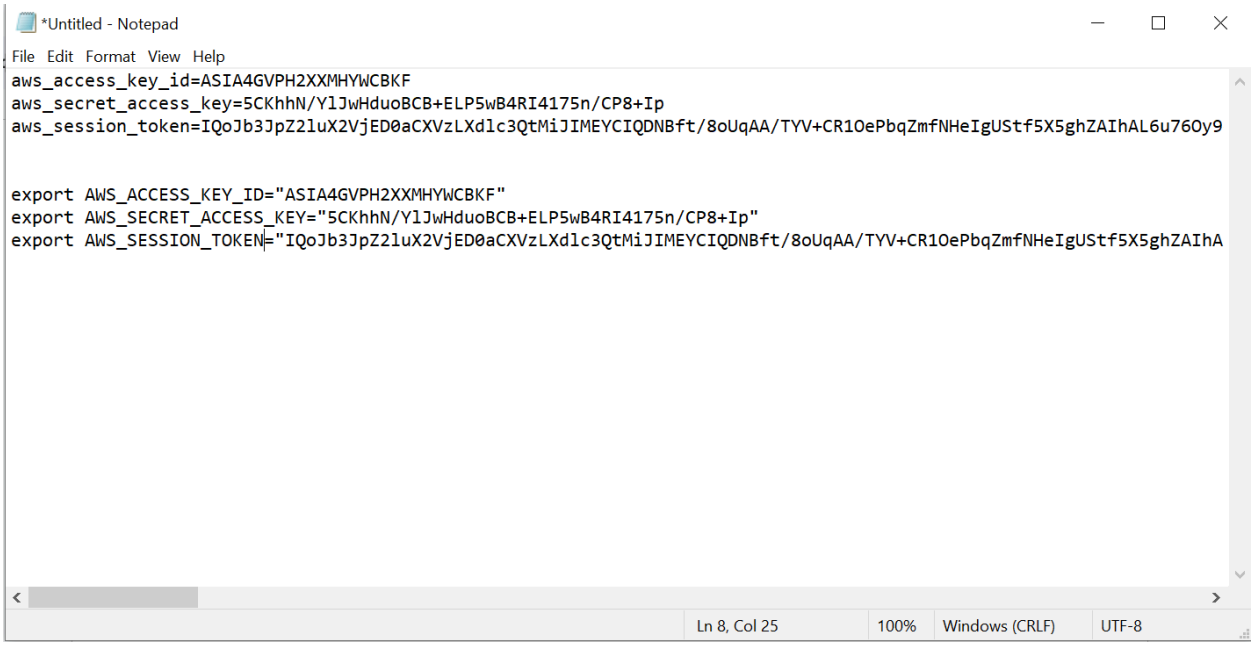
Screenshots:-

1. Turn on the aws-academy lab and click on AWS Details to get the access key, secret key, session token

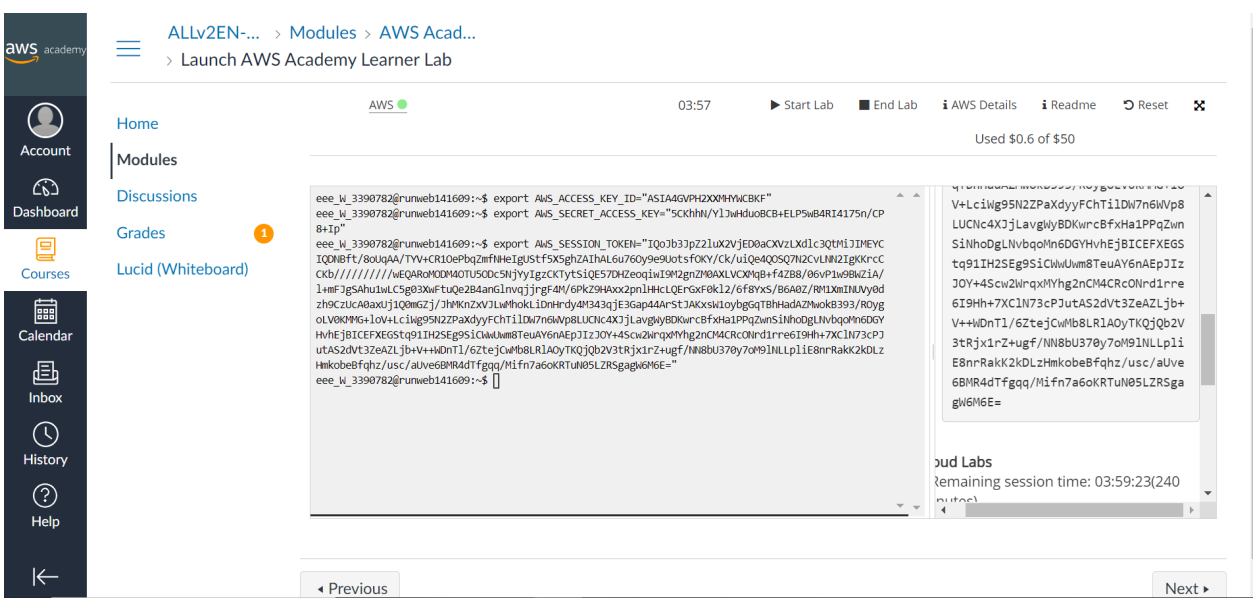


2. Copy all those details in notepad, and make sure to do it like this export ←Name→=”←actual key→”.

Name:- Arnav .S. Sawant D15A Roll-no:- 53

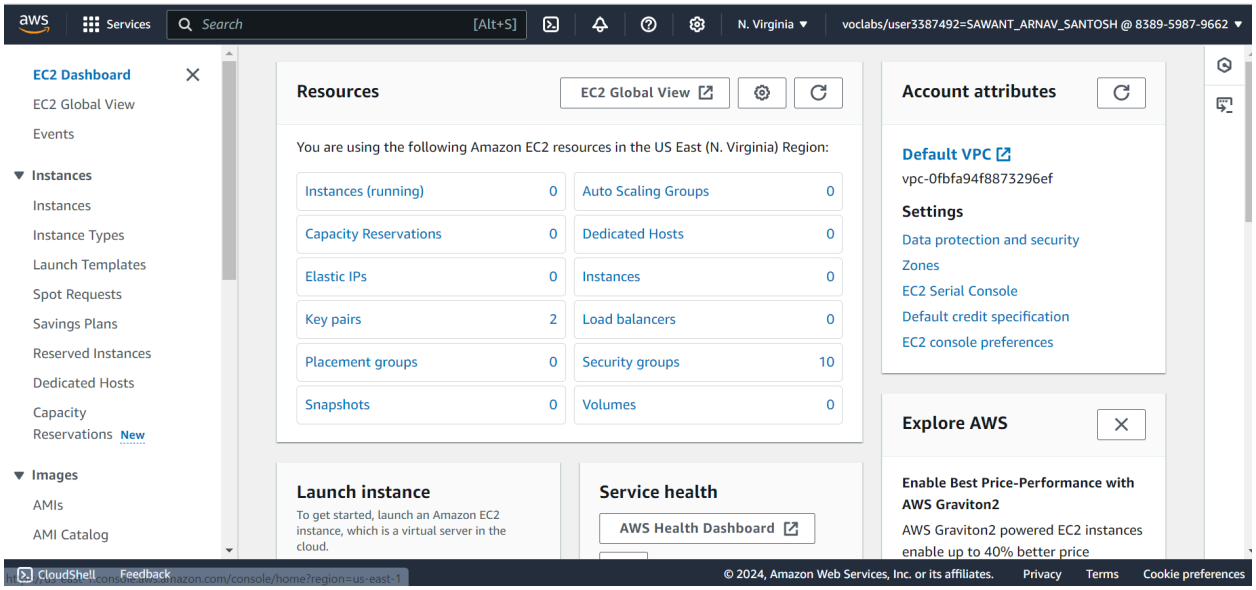


3. Execute those commands on the CLI.

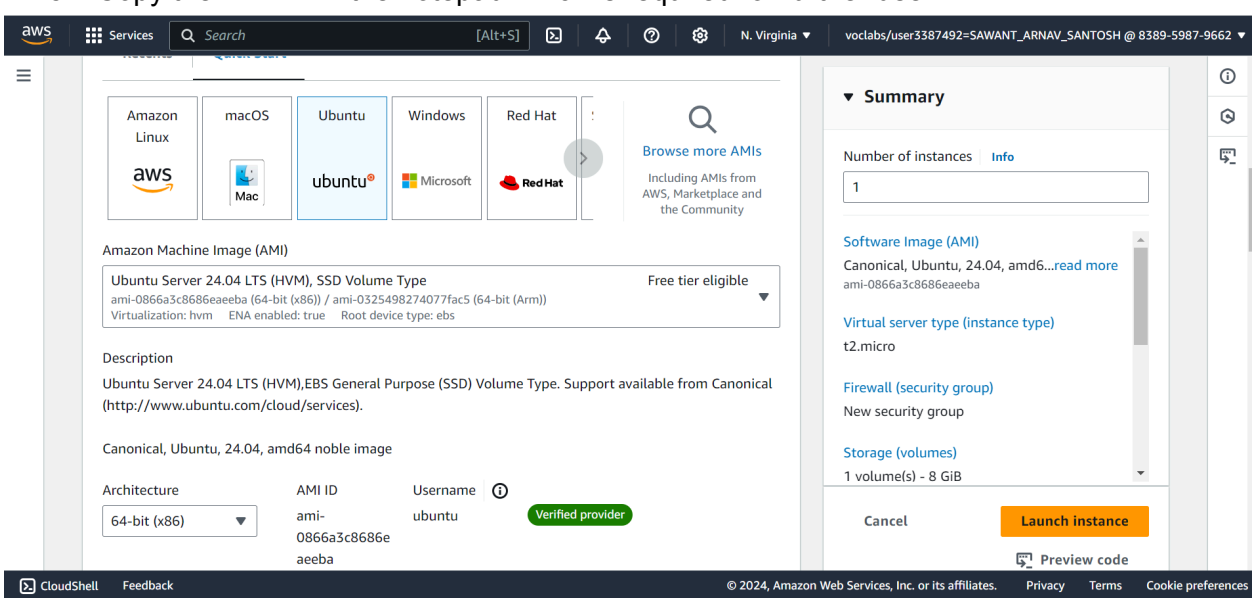


4. Go to your EC2 instance and check for any running or existing instances, **here in the photo there are no running instances**.

Name:- Arnav .S. Sawant D15A Roll-no:- 53

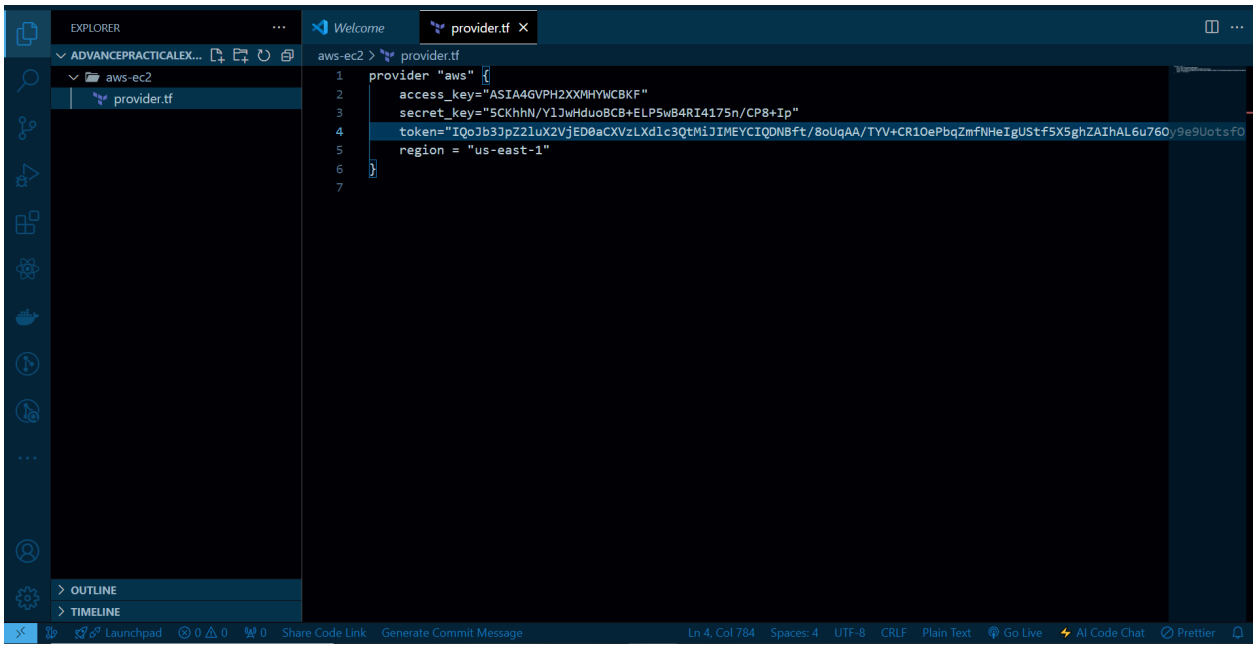


5. Copy the AMI ID in the notepad which is required for further use



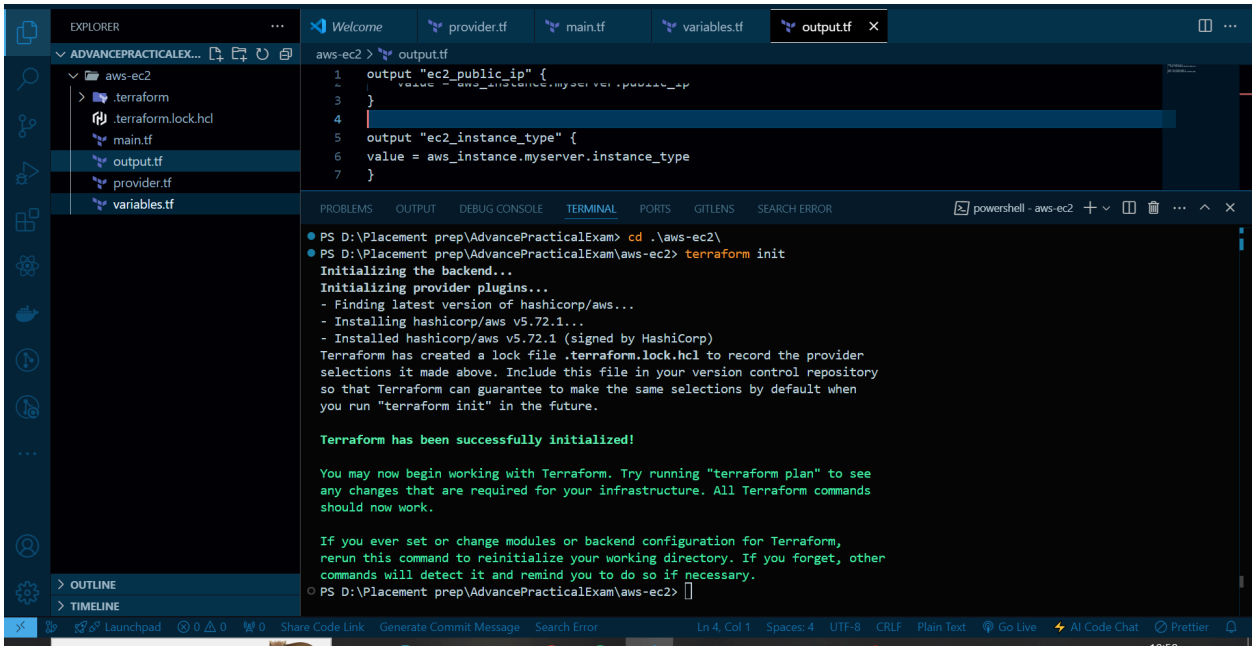
6. Paste the content of the notepad in provider.tf of the aws-ec2 folder.

Name:- Arnav .S. Sawant D15A Roll-no:- 53

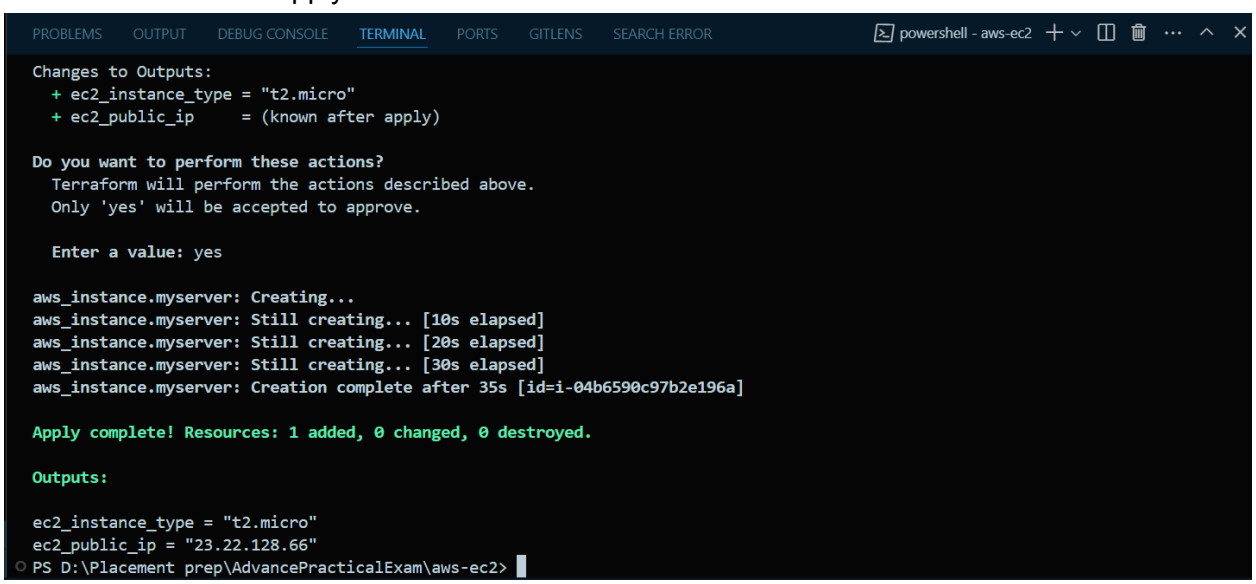


7. Write the code of output.tf and variables.tf in the respective file and on terminal write “terraform init”.

Name:- Arnav .S. Sawant D15A Roll-no:- 53

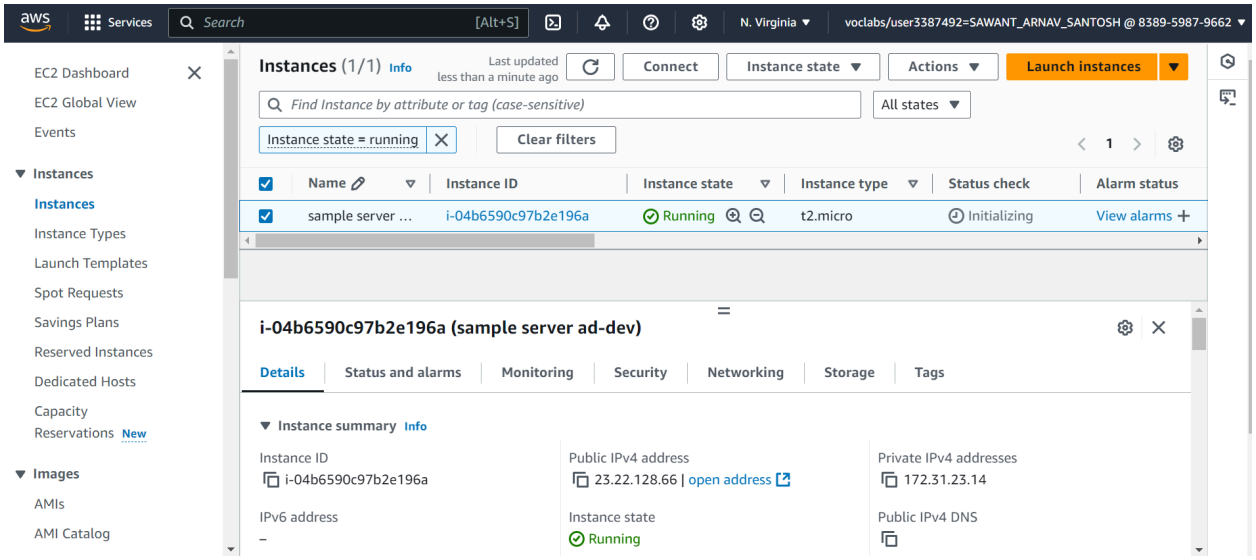


8. Then execute the command “terraform plan” , check for errors , if there are no errors go for “terraform apply”



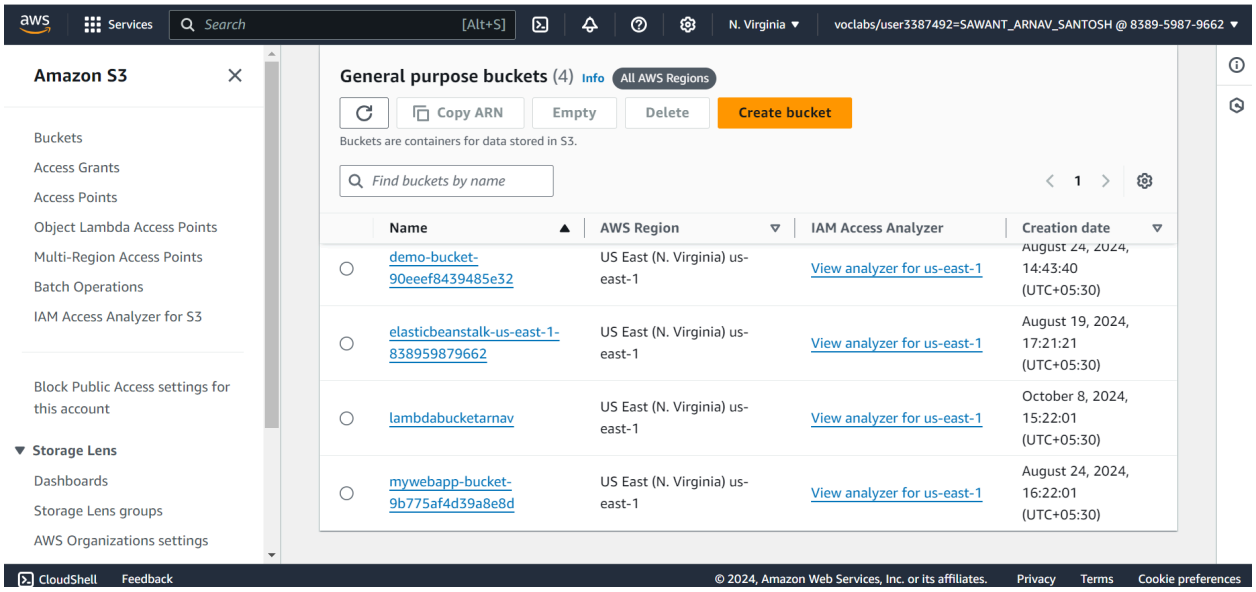
9. Now you can see that your ec2-instance has been created

Name:- Arnav .S. Sawant D15A Roll-no:- 53



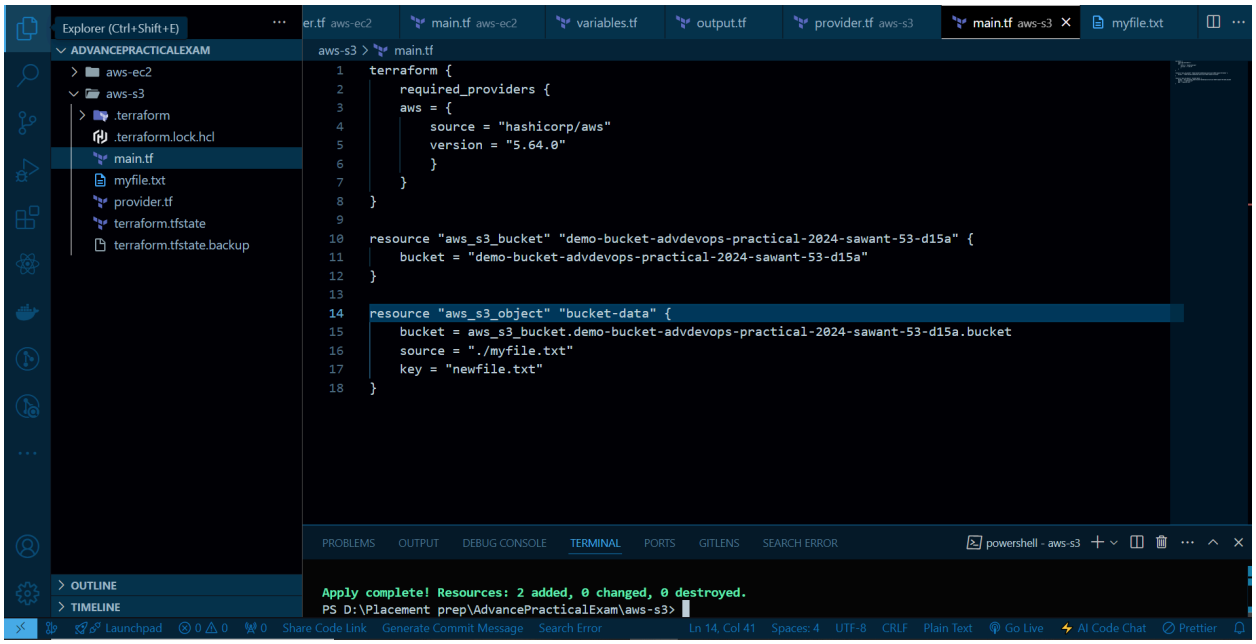
Now the part of s3

10. These are the list of s3 buckets before writing the terraform script to create an s3 bucket

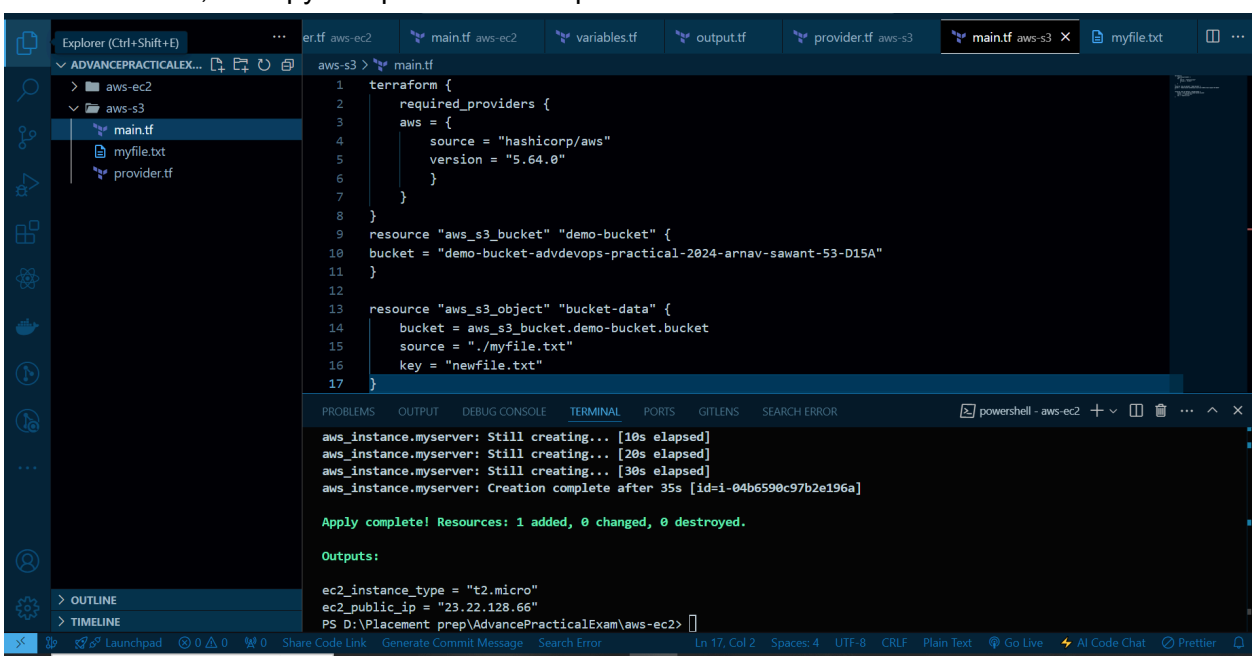


Name:- Arnav .S. Sawant D15A Roll-no:- 53

11. Write the code in “main.tf” for bucket and remember to make the name of the bucket globally unique

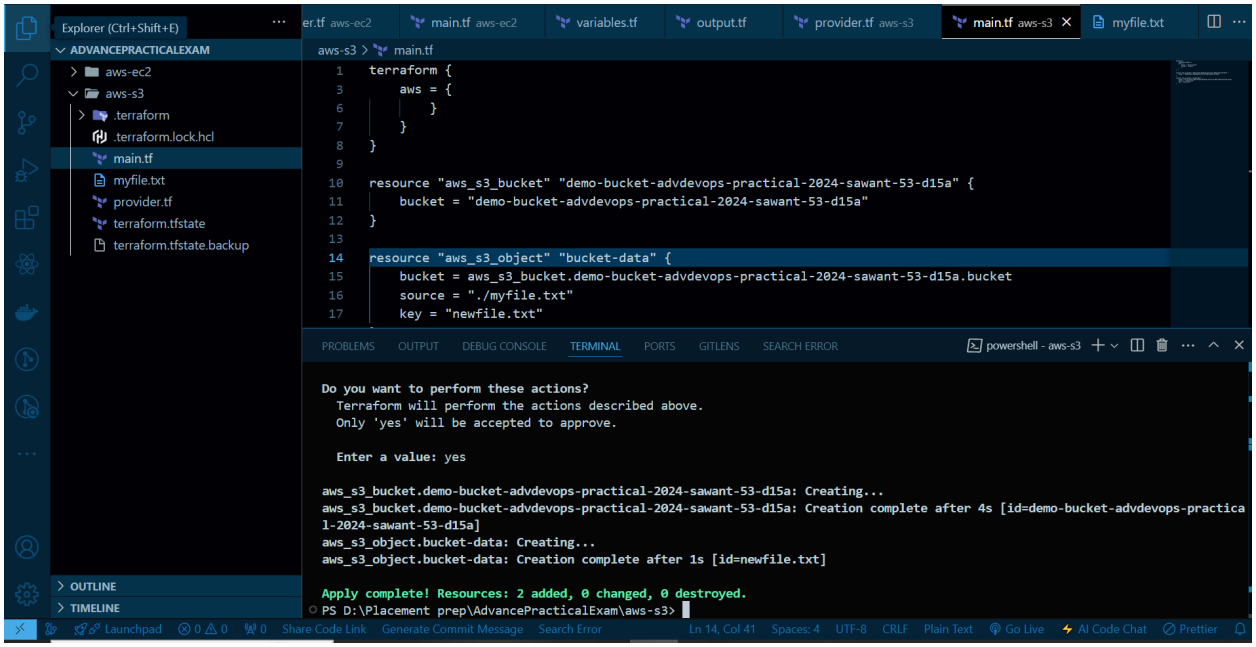


12. Now after the “terraform apply” command for EC2 the public ip is been listed in the terminal, so copy the ip address and paste in the .txt file created in the aws-s3 folder

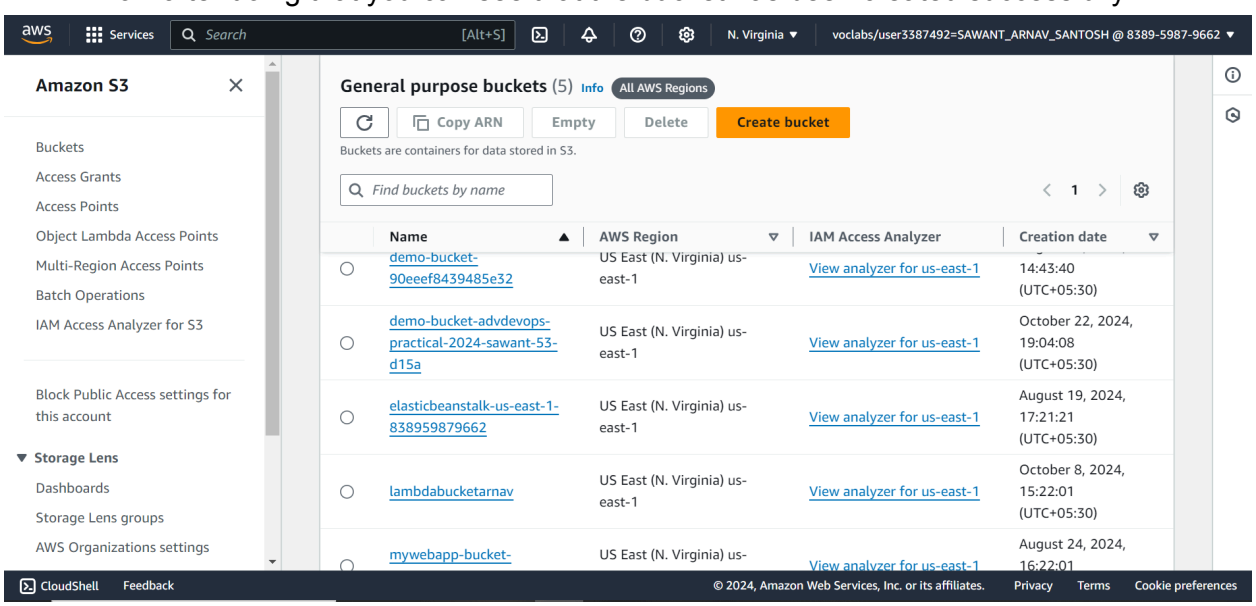


Name:- Arnav .S. Sawant D15A Roll-no:- 53

13. Same go for all the terraform commands and lastly do “terraform apply”

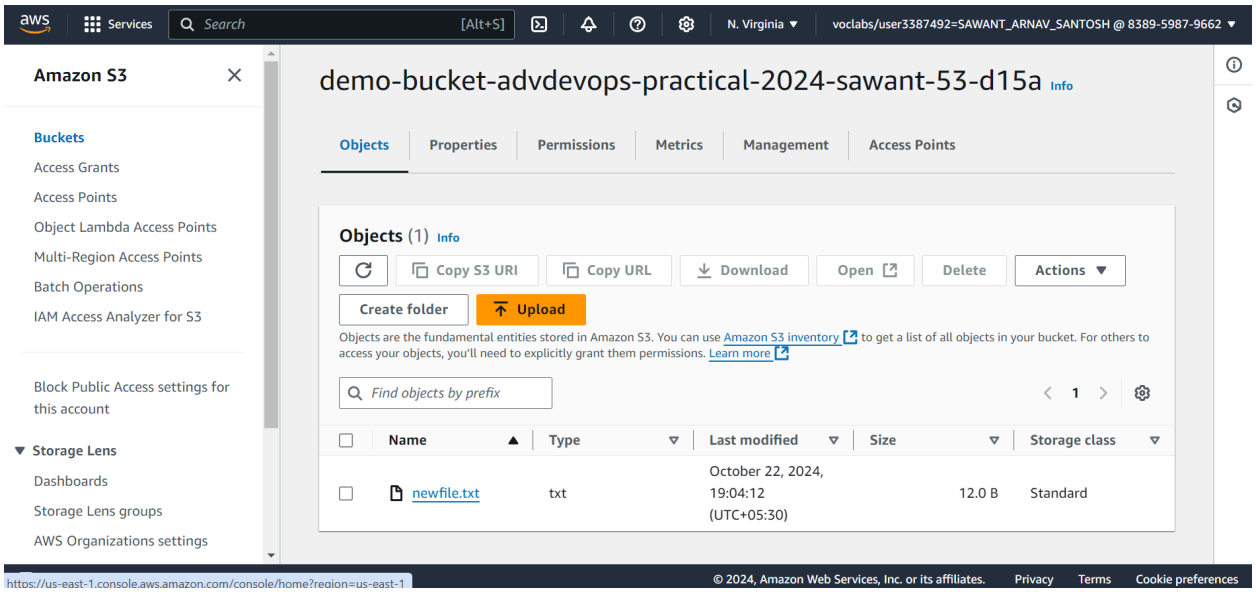


14. Now after doing that you can see that the bucket has been created successfully



Name:- Arnav .S. Sawant D15A Roll-no:- 53

15. Also the newfile.txt has been created and the ip of the ec2 instance has been stored there



Then destroy both the ec2 instance and s3 bucket using command “terraform destroy”

For s3 -

```
PS D:\Placement prep\AdvancePracticalExam\aws-s3> terraform destroy
aws_s3_bucket.demo-bucket-advdevops-practical-2024-sawant-53-d15a: Refreshing state... [id=demo-bucket-advdevops-practical-2024-
sawant-53-d15a]
aws_s3_object.bucket-data: Refreshing state... [id=newfile.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_s3_bucket.demo-bucket-advdevops-practical-2024-sawant-53-d15a will be destroyed
  - resource "aws_s3_bucket" "demo-bucket-advdevops-practical-2024-sawant-53-d15a" {
      - arn                         = "arn:aws:s3:::demo-bucket-advdevops-practical-2024-sawant-53-d15a" -> null
      - bucket                      = "demo-bucket-advdevops-practical-2024-sawant-53-d15a" -> null
      - bucket_domain_name          = "demo-bucket-advdevops-practical-2024-sawant-53-d15a.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "demo-bucket-advdevops-practical-2024-sawant-53-d15a.s3.us-east-1.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
```

For ec2 -

Name:- Arnav .S. Sawant D15A Roll-no:- 53

```
PS D:\Placement prep\AdvancePracticalExam\aws-s3> cd ..
PS D:\Placement prep\AdvancePracticalExam> cd .\aws-ec2\
PS D:\Placement prep\AdvancePracticalExam\aws-ec2> terraform destroy
aws_instance.myserver: Refreshing state... [id=i-04b6590c97b2e196a]

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.myserver will be destroyed
  - resource "aws_instance" "myserver" {
      - ami                                  = "ami-0866a3c8686eaeeba" -> null
      - arn                                  = "arn:aws:ec2:us-east-1:838959879662:instance/i-04b6590c97b2e196a" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-1a" -> null
      - cpu_core_count                       = 1 -> null
```