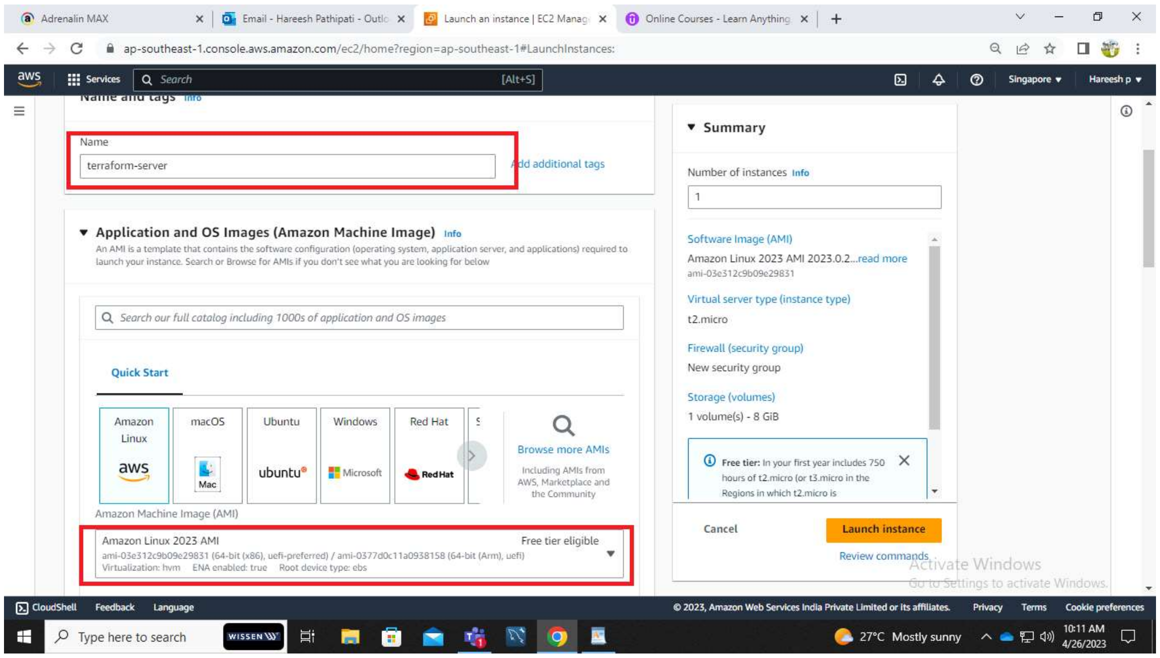
creation of ec2 instances and s3 Bucket through terraform

- Sign in to AWS Management Console
- Navigate to ec2 dashboard create an ec2 instance for installing Terraform



Name: terraform-server

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Quick Start: Amazon Linux, macOS, Ubuntu, Windows, Red Hat

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-03e312c9b09e29831 (64-bit (x86), uefi-preferred) / ami-0377d0c11a0938158 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.0.2...read more
ami-03e312c9b09e29831

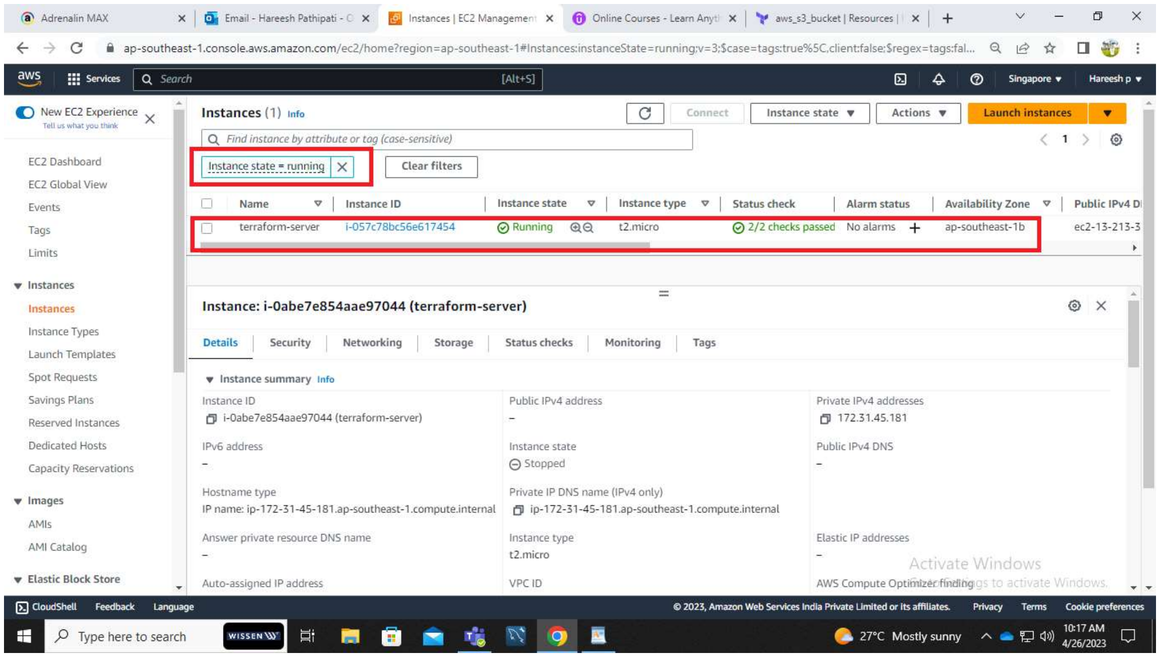
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is

Cancel | Launch instance | Review commands



Instances (1) Info

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
terraform-server	i-057c78bc56e617454	Running	t2.micro	2/2 checks passed	No alarms	ap-southeast-1b	ec2-13-213-3

Instance: i-0abe7e854aae97044 (terraform-server)

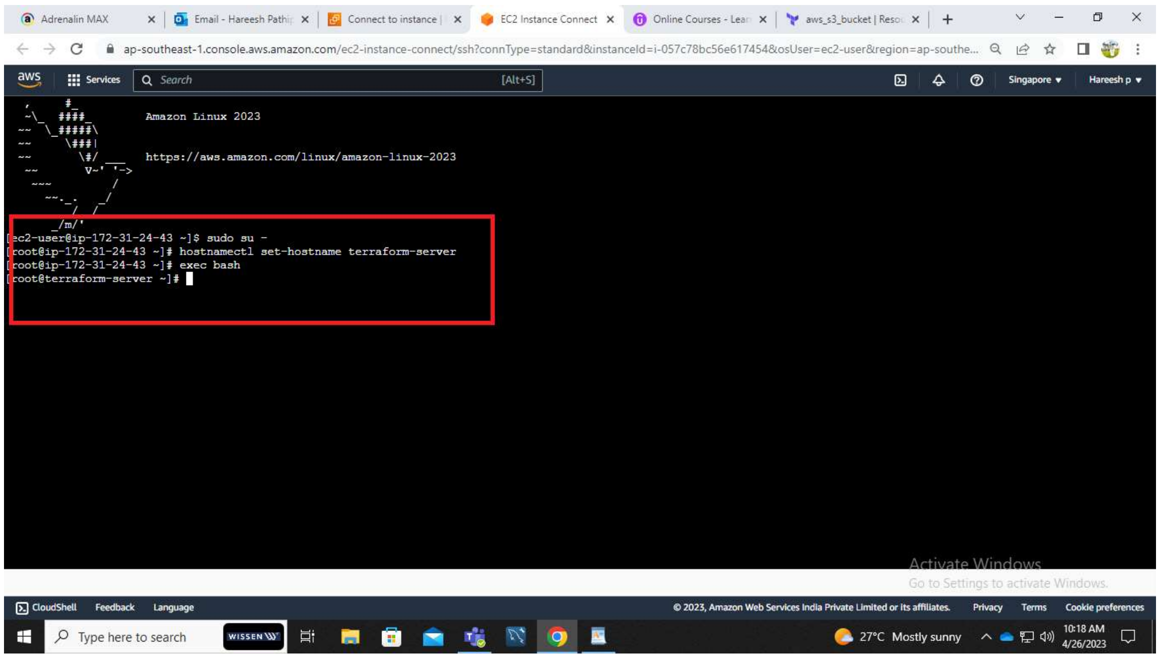
Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0abe7e854aae97044 (terraform-server)	–	172.31.45.181

IPv6 address: –
Instance state: Stopped
Public IPv4 DNS: –
Hostname type: IP name: ip-172-31-45-181.ap-southeast-1.compute.internal
Private IP DNS name (IPv4 only): ip-172-31-45-181.ap-southeast-1.compute.internal
Answer private resource DNS name: –
Instance type: t2.micro
Elastic IP addresses: –
Auto-assigned IP address
VPC ID
AWS Compute Optimizer finding

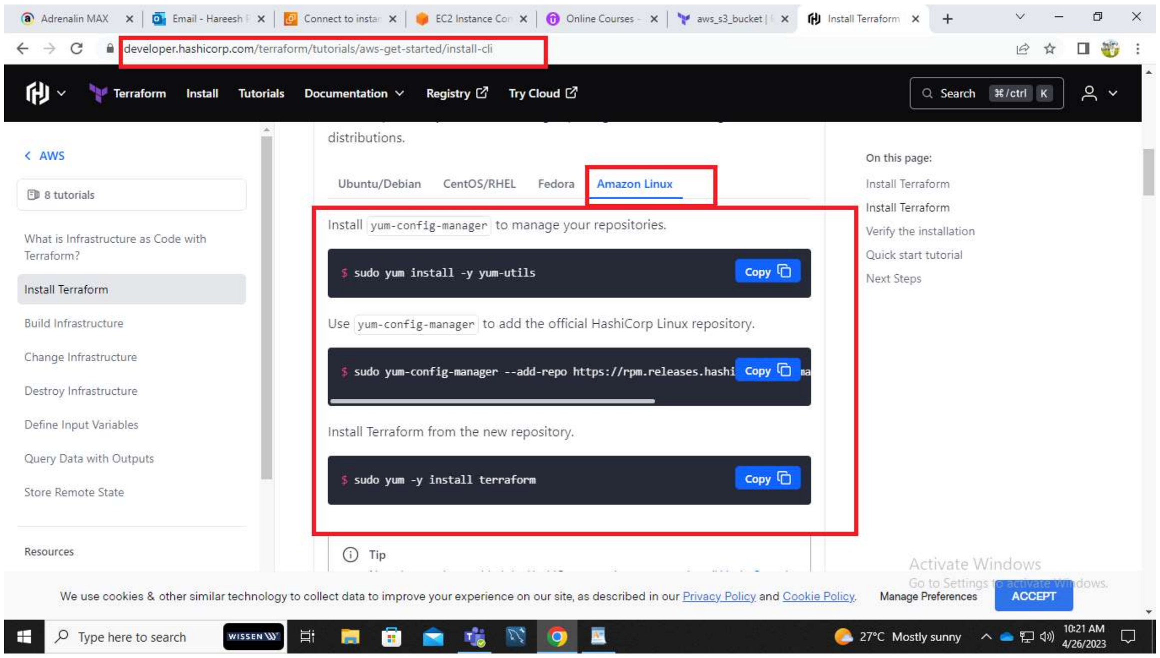
- connect to terminal via ec2 connect



```
 ,     #_
 ~\_  ####_        Amazon Linux 2023
~~  \_#####\
~~     \###|
~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
 ~~       V~' '->
  ~~~         /
    ~~._.   _/
       _/ _/
     _/m/'
[ec2-user@ip-172-31-24-43 ~]$ sudo su -
[root@ip-172-31-24-43 ~]# hostnamectl set-hostname terraform-server
[root@ip-172-31-24-43 ~]# exec bash
[root@terraform-server ~]# 
```

- open the Terraform.io For terraform installation in Amazon linux machine

- Follow the commands to install terraform in linux machine



developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

Ubuntu/Debian CentOS/RHEL Fedora Amazon Linux

Install `yum-config-manager` to manage your repositories.

```
$ sudo yum install -y yum-utils
```

Use `yum-config-manager` to add the official HashiCorp Linux repository.

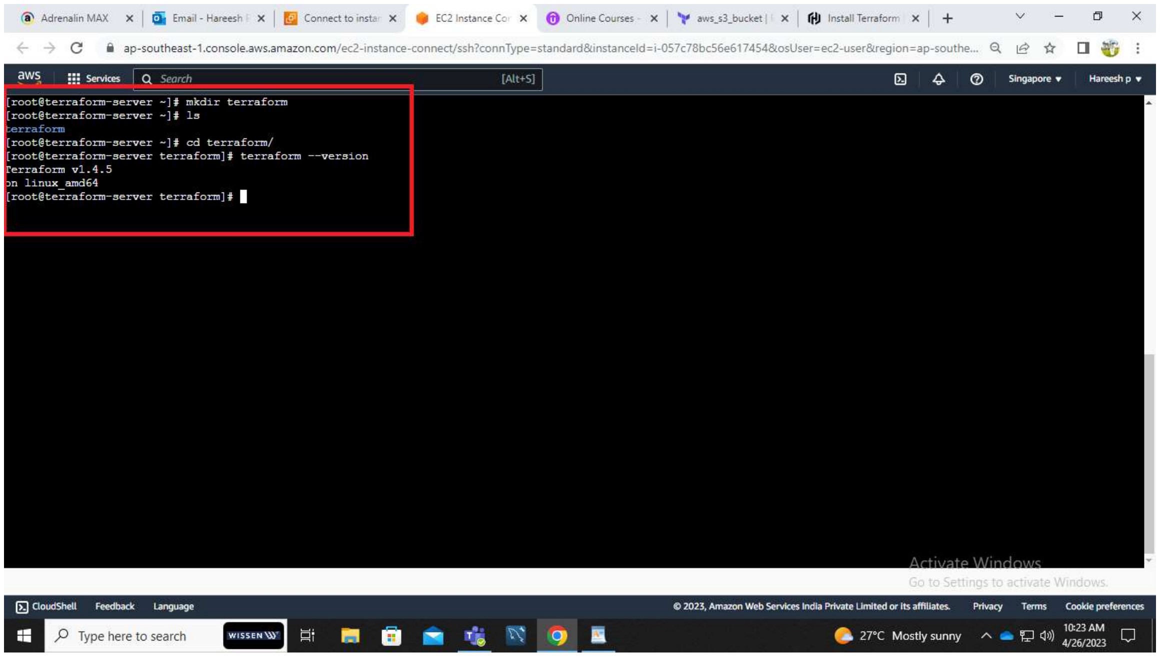
```
$ sudo yum-config-manager --add-repo https://rpm.releases.hashi
```

Install Terraform from the new repository.

```
$ sudo yum -y install terraform
```

- Confirm the installation of Terraform by checking the version.

- create a directory for working of terraform



```
[root@terraform-server ~]# mkdir terraform
[root@terraform-server ~]# ls
terraform
[root@terraform-server ~]# cd terraform/
[root@terraform-server terraform]# terraform --version
Terraform v1.4.5
on linux_amd64
[root@terraform-server terraform]#
```

- Apply the terraform init - initializes a working directory containing Terraform configuration files

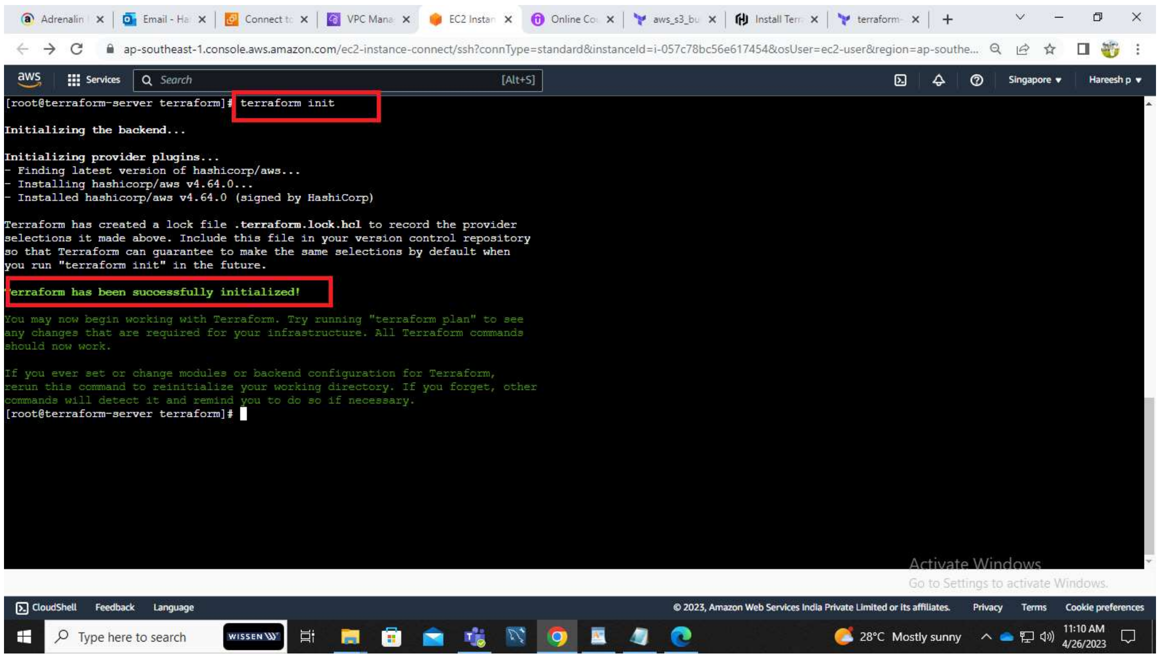
- Create a main.tf ,variable.tf files - inside the files we run the code to build an infrastructure .

main.tf --

```
provider "aws" {
  access_key = var.aws_access_key
  secret_key = var.aws_secret_key
  region     = var.region
}
resource "aws_instance" "os" {
  count         = 2
  ami           = var.ami
  instance_type = var.instance_type
  availability_zone = var.availability_zone
  tags = {
    Name = "terraform-instance"
  }
}
resource "aws_s3_bucket" "terraform-bucket" {
  bucket = "s3-newterraform-bucket"
  tags = {
    Name        = "terraform-bucket"
    Environment = "Dev"
  }
}
```

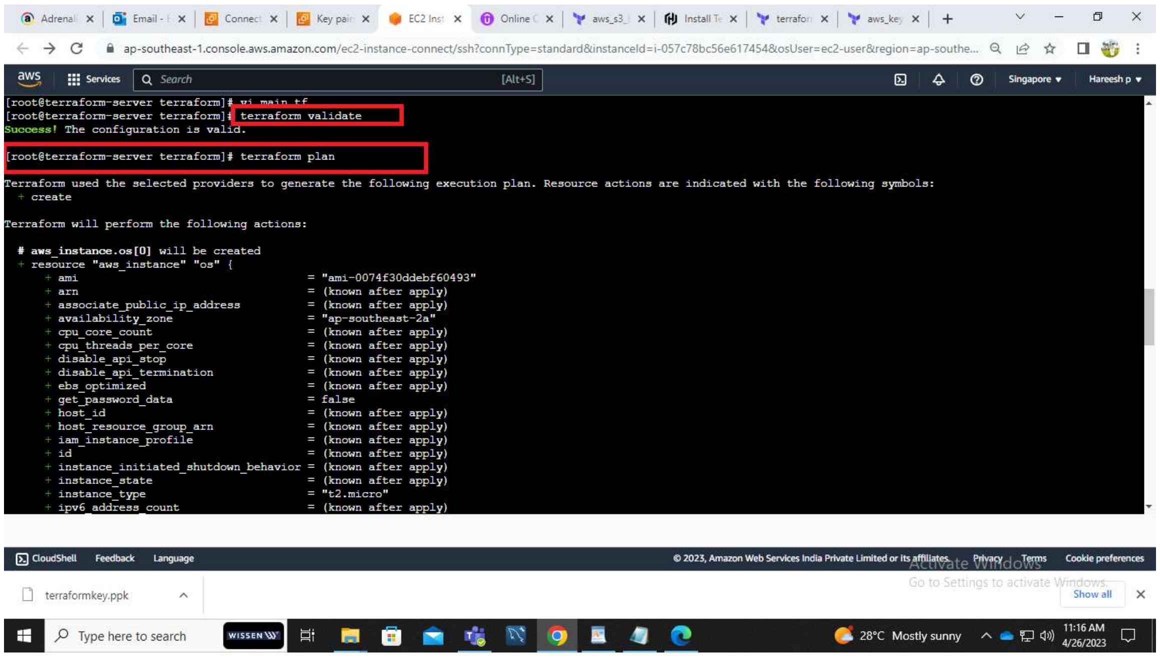
variable.tf --

```
variable "aws_access_key" {
  default = "AKIA5FBOF72JUGIIAOUM"
}
variable "aws_secret_key" {
  default = "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"
}
variable "region" {
  default = "ap-southeast-2"
}
variable "ami" {
  default = "ami-0074f30ddebf60493"
}

variable "instance_type" {

  default = "t2.micro"

}

variable "availability_zone" {

  default = "ap-southeast-2a"

}
```



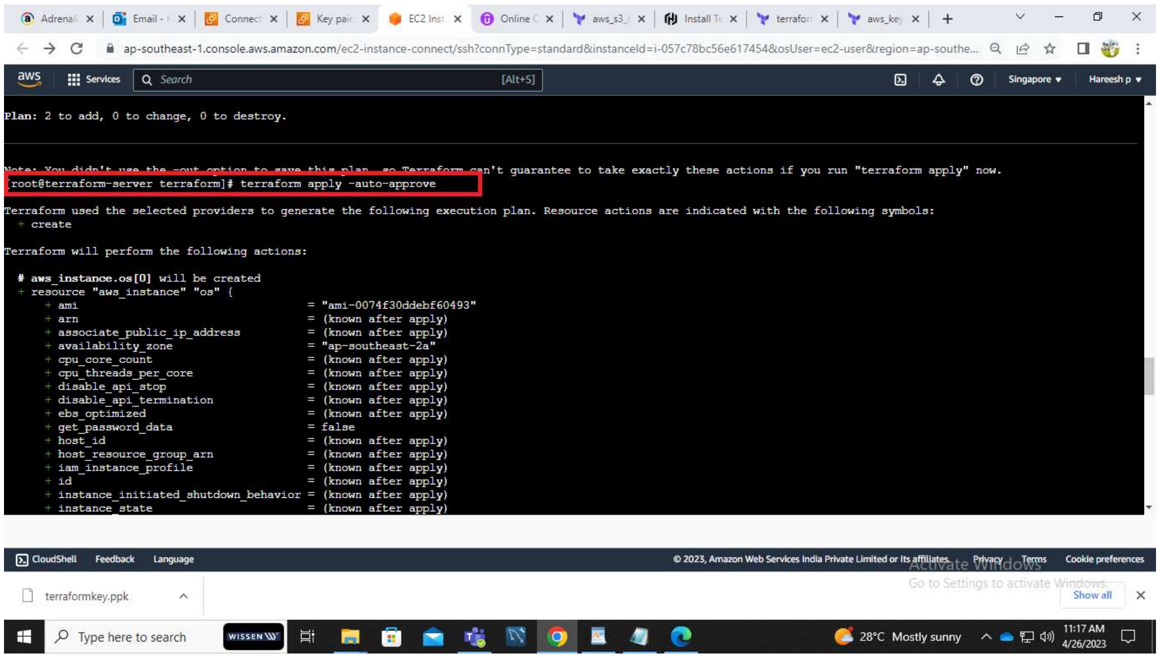
```
[root@terraform-server terraform]# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.64.0...
- Installed hashicorp/aws v4.64.0 (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.
[root@terraform-server terraform]#
```

- Apply terrform validate -it will Check whether the configuration is valid
- Apply terraform fmt - Reformat your configuration in the standard style
- Apply terraform plan - Show changes required by the current configuration



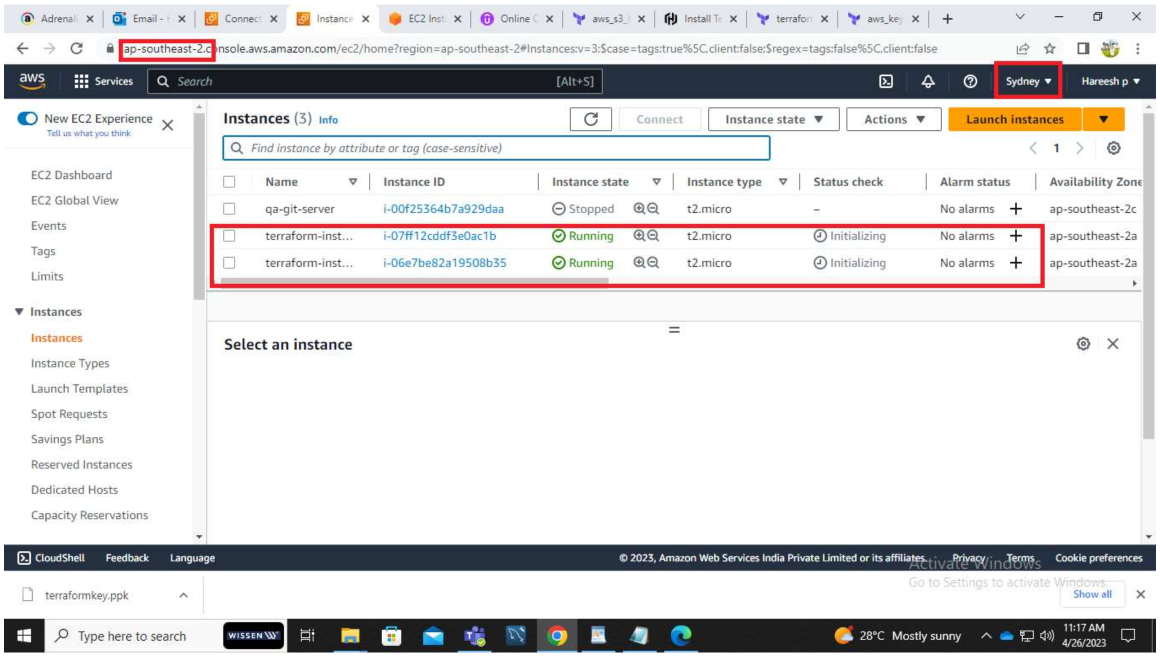
```
[root@terraform-server terraform]# vi main.tf
[root@terraform-server terraform]# terraform validate
Success! The configuration is valid.

[root@terraform-server terraform]# 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.os[0] will be created
  + resource "aws_instance" "os" {
      + ami                                  = "ami-0074f30ddebf60493"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = "ap-southeast-2a"
      + 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)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
```

- Apply terraform apply - Create or update infrastructure



```
Plan: 2 to add, 0 to change, 0 to destroy.


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
[root@terraform-server terraform]# terraform apply -auto-approve

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.os[0] will be created
  + resource "aws_instance" "os" {
      + ami                                  = "ami-0074f30ddebf60493"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = "ap-southeast-2a"
      + 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)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
```

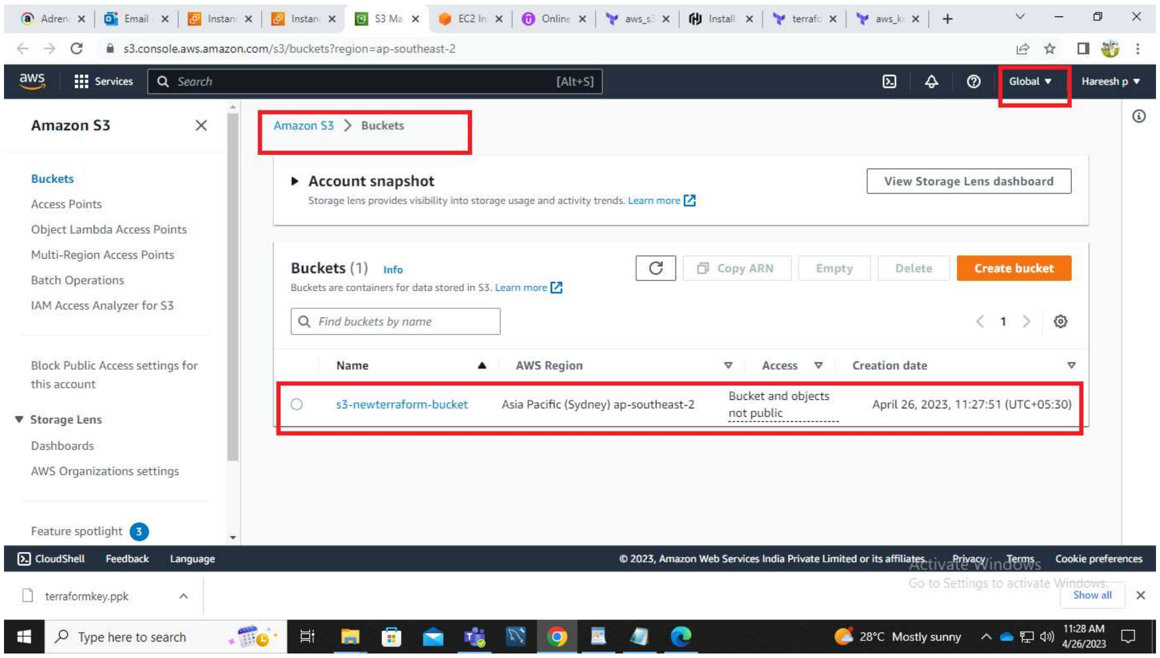
- Navigate to ec2 Dash board and check the resources created or not



Instances (3)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
qa-git-server	i-00f25364b7a929daa	Stopped	t2.micro	–	No alarms	ap-southeast-2c
terraform-inst…	i-07ff12cddf3e0ac1b	Running	t2.micro	Initializing	No alarms	ap-southeast-2a
terraform-inst…	i-06e7be82a19508b35	Running	t2.micro	Initializing	No alarms	ap-southeast-2a

- Navigate to s3 dash board check the resources

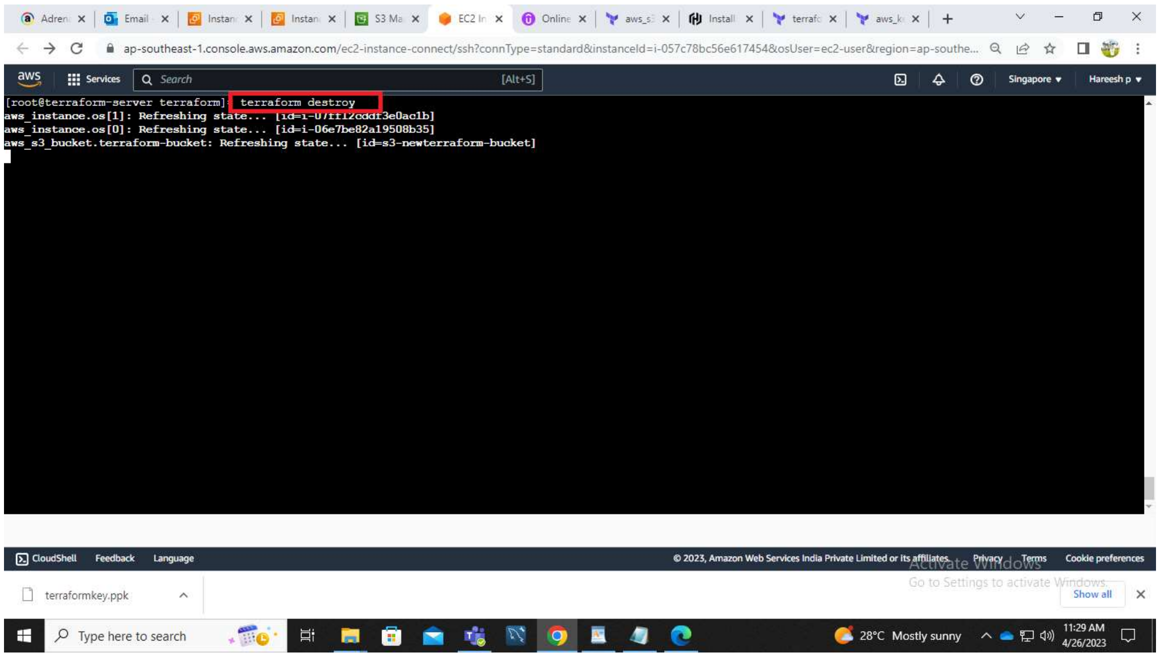


s3.console.aws.amazon.com/s3/buckets?region=ap-southeast-2
Global
Hareesh p
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
Amazon S3 > Buckets
Account snapshot
View Storage Lens dashboard
Copy ARN
Empty
Delete
Create bucket

Buckets (1)

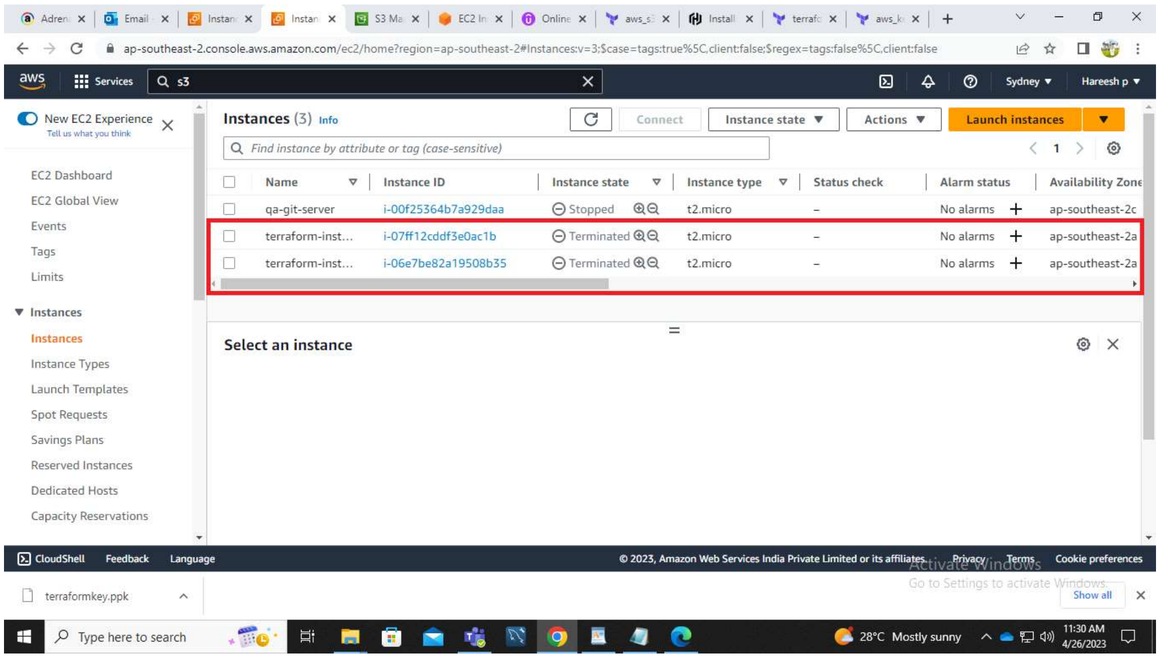
Name	AWS Region	Access	Creation date
s3-newterraform-bucket	Asia Pacific (Sydney) ap-southeast-2	Bucket and objects not public	April 26, 2023, 11:27:51 (UTC+05:30)

- Apply terraform destroy - it will Destroy previously-created infrastructure
- In



```
[root@terraform-server terraform]# terraform destroy
aws_instance.os[1]: Refreshing state... [id=i-07ff12cddf3e0ac1b]
aws_instance.os[0]: Refreshing state... [id=i-06e7be82a19508b35]
aws_s3_bucket.terraform-bucket: Refreshing state... [id=s3-newterraform-bucket]
```

- verify the resources destroyed or not



Instances (3)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
qa-git-server	i-00f25364b7a929daa	Stopped	t2.micro	–	No alarms	ap-southeast-2c
terraform-inst...	i-07ff12cddf3e0ac1b	Terminated	t2.micro	–	No alarms	ap-southeast-2a
terraform-inst...	i-06e7be82a19508b35	Terminated	t2.micro	–	No alarms	ap-southeast-2a