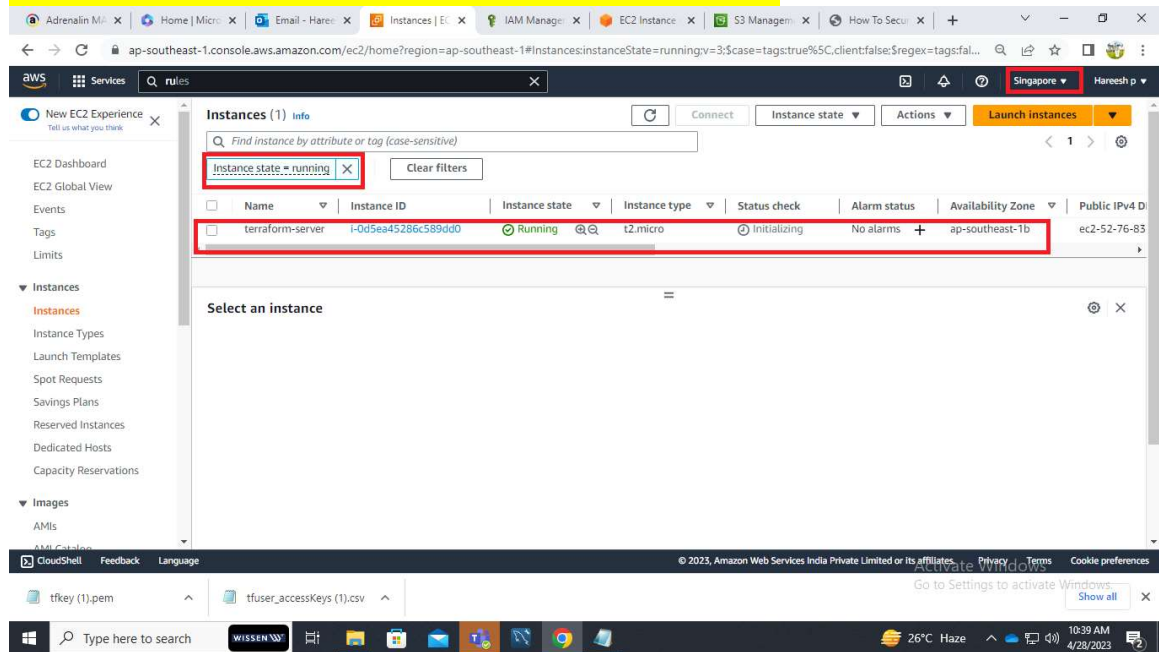
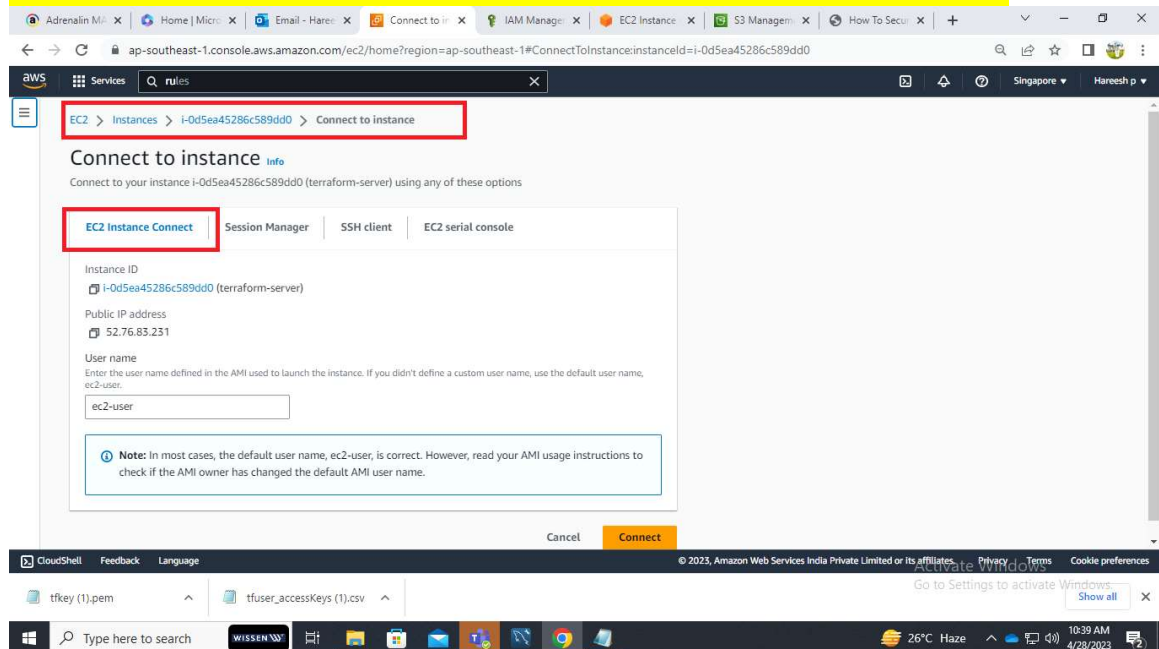


s3 Bucket Creation using terraform

step-1: Login to aws management console create a ec2 instance



step-2: connect the terminal via ec2 connect (browser based) to run the commands



step-3: Login as a root user and create a hostname

- install the terraform in amazon linux machine follow the link and run those commands:

1. sudo yum install -y yum-utils

2. sudo yum-config-manager --add-repo

<https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>

3. sudo yum -y install terraform

4. terraform --version

step-4: create a directory and change in to that directory

step-5: create a main.tf and provider.tf files to run the below scripts

main.tf

```
resource "aws_s3_bucket" "aws-bucket" {
```

```
    bucket = "terraformnewawsbuckettttttt"
```

```
    acl    = "private"
```

```
    versioning {
```

```
        enabled = "false"
```

```
    }
```

```
    force_destroy = "false"
```

```
}
```

```
variable "aws_access_key" {}
```

```
variable "aws_secret_key" {}
```

provider.tf

```
provider "aws" {
```

```
    access_key = var.aws_access_key
```

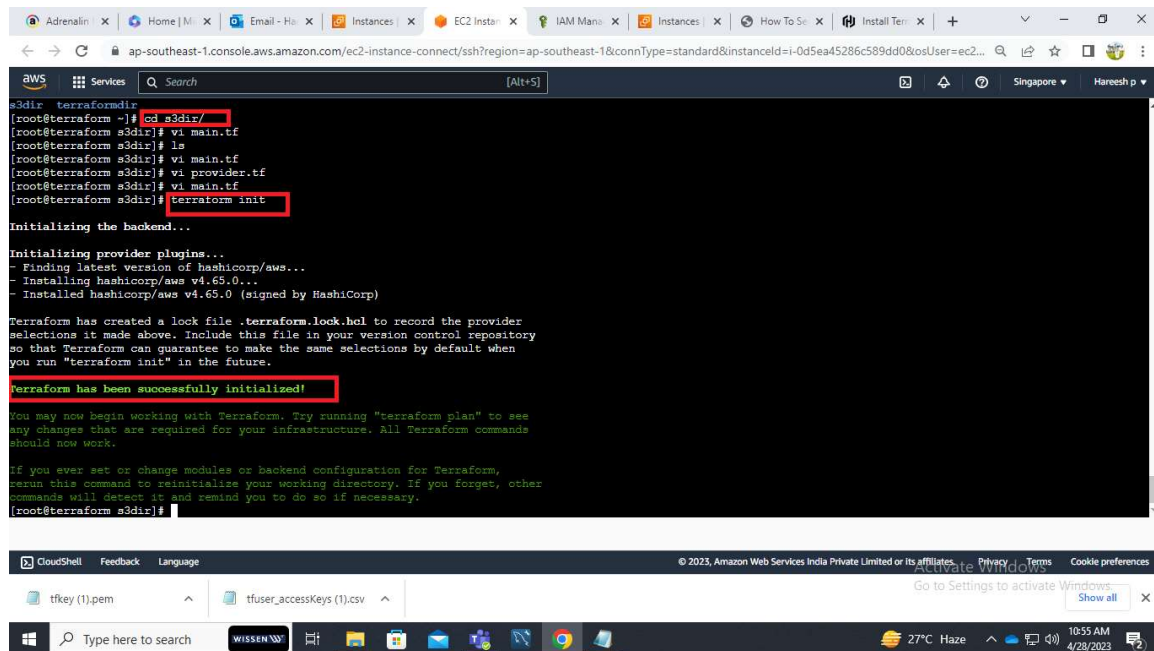
```
secret_key = var.aws_secret_key

region    = "ap-southeast-2"

}
```

step-6: Apply the below commands to run the terraform

1.terraform init - Prepare your working directory for other command



The screenshot shows a terminal window with the following commands and output:

```
s3dir terraformdir
[root@terraform ~]# cd s3dir/
[root@terraform s3dir]# vi main.tf
[root@terraform s3dir]# ls
[root@terraform s3dir]# vi main.tf
[root@terraform s3dir]# vi provider.tf
[root@terraform s3dir]# vi main.tf
[root@terraform s3dir]# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.65.0...
- Installed hashicorp/aws v4.65.0 (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 s3dir]#
```

2. terraform validate - it will Check whether the configuration is valid

3. terraform fmt - it will reformat your configuration in the standard style

4. terraform plan - it will Show changes required by the current configuration

```
[root@terraform s3dir]# terraform validate
Warning: Argument is deprecated

  with aws_s3_bucket.aws-bucket,
  on main.tf line 1, in resource "aws_s3_bucket" "aws-bucket":
   1: resource "aws_s3_bucket" "aws-bucket" {}

Use the aws_s3_bucket_versioning resource instead
(and one more similar warning elsewhere)

Success! The configuration is valid, but there were some validation warnings as shown above.

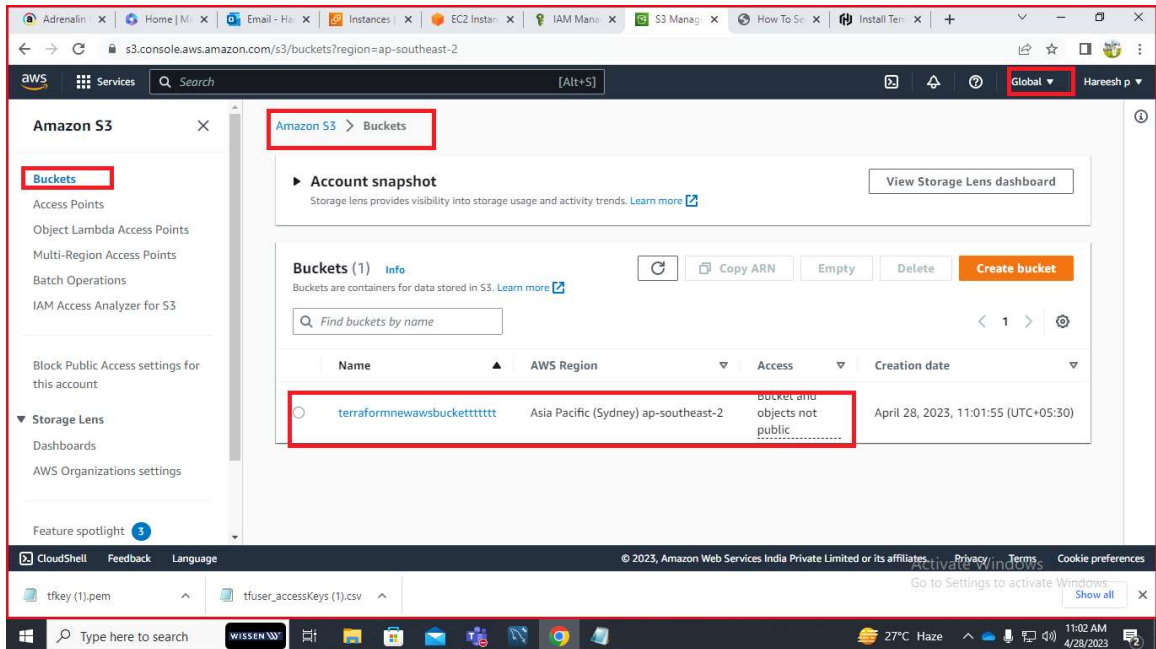
[root@terraform s3dir]# terraform fmt
[root@terraform s3dir]# terraform plan
var.aws_access_key
Enter a value:
```

5. terraform apply -auto-approve - it will Create or update infrastructure

```
[root@terraform s3dir]# terraform apply -auto-approve
var.aws_access_key
Enter a value:
```

step-6: Navigate to s3 dash board and check the bucket created or Not

Here the s3 bucket is created successfully



Apply terraform destroy - Here Destroy previously-created infrastructure

