SCENARIO:-

Mr Paulo is an IT Administrator of Alchemist group Pvt. Ltd. His organization is passionate about adopting IaaS using public cloud service providers. Their majority of clients are e-commerce and OTP service providers. Initially, they want to set up two virtual windows servers using Amazon EC2 which can be resizable and provide compute capacity along with a web-scale cloud computing solution.

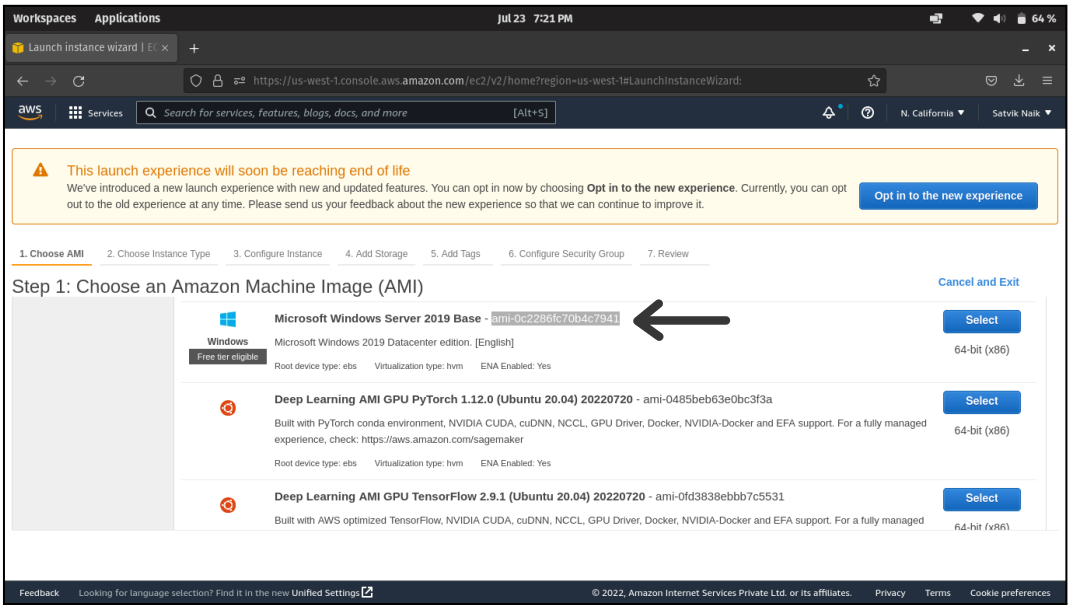
Paulo is planning to create IAAS as below for E-Commerce clients. You are required to provide the solution to Paulo with proper step by step demonstration. Consider the following attached scenario and perform the following tasks using AWS EC2 Service:

- Launch a web server with termination protection enabled
- Monitor Your EC2 instance
- Modify the security group that your web server is using to allow HTTP access
- Resize your Amazon EC2 instance to scale
- Explore EC2 limits
- Test termination protection
- Terminate your EC2 instance

AIM:- To automate above Infrastructure using Infrastructure As A Code - Terraform.

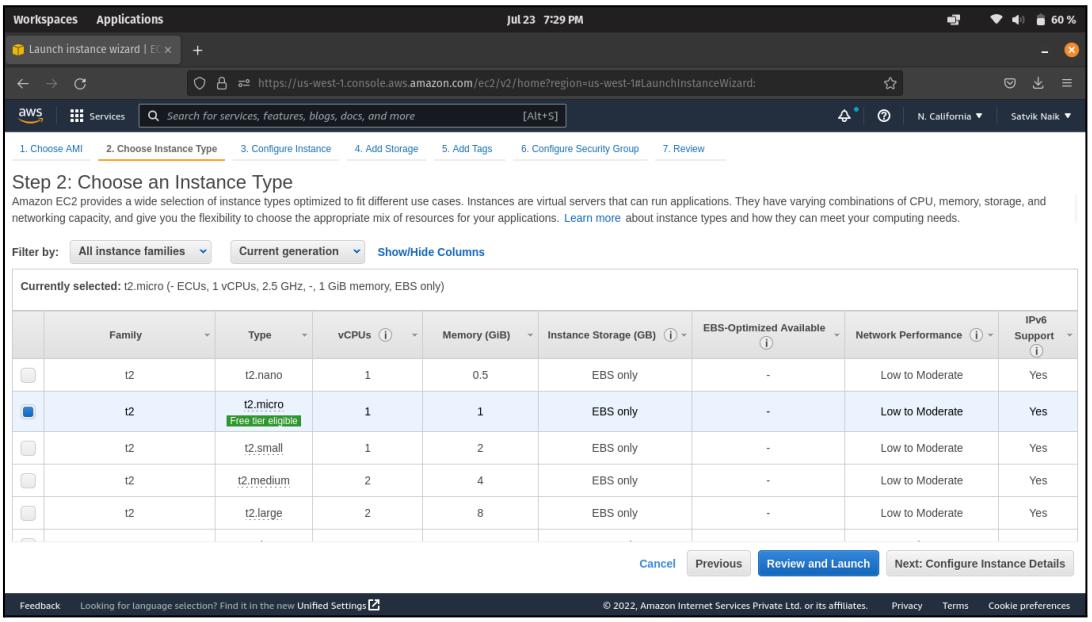
Steps:-

1. **Getting pre-requisite details.**
   To create an instance, we need some pre-requisite details of the resource such as follows:
   - **ami**
     - To know the ami of the server you must simply redirect to ***Instances > Launch Instance***.
     - There you will get the list of various ami's.



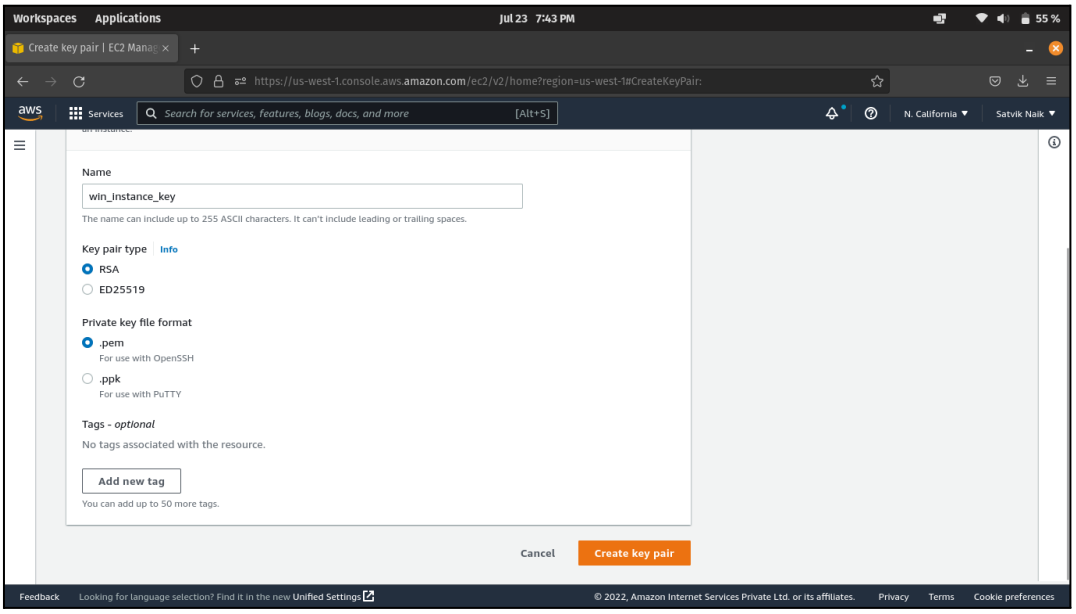
- As I want to create a windows server instance I am selecting the windows server ami.

- **instance_type**
  - After selecting the ami on clicking on ***'Next'***, you will get an option to choose an instance type.

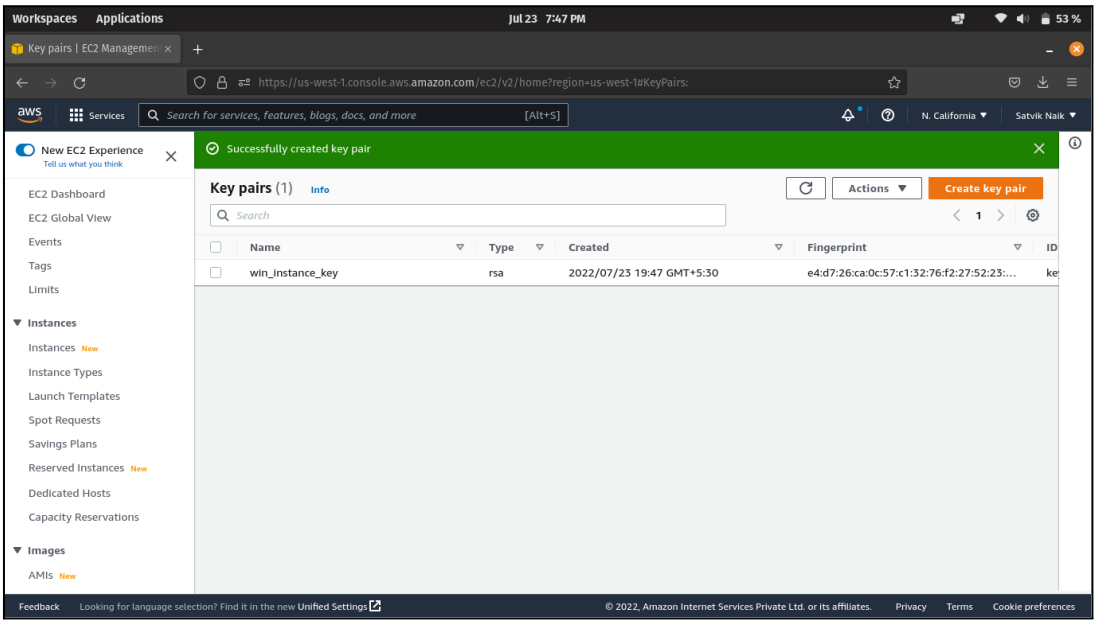


  - As we are working in a ***free tier account***, we will always select ***t2.micro*** as instance type.

- **key_name**
  - To create a key pair redirect to ***Key Pairs > Create key pair***.



  - Give ***key_name***, select ***key pair type*** & ***private key file format***. Click on ***'Create key pair'***.



- Key pair created successfully.

2. Configuring terraform code.

- **CODE**:-

```
terraform {
 required_providers {
   aws = {
     source  = "hashicorp/aws"
     version = "~> 3.0"
   }
 }
 required_version = ">= 0.13.5"
}

# Configure the AWS Provider
provider "aws" {
 region     = "us-east-1"
 access_key = "ACCESS_KEY"
 secret_key = "SECRET_KEY"
}

# Create a Instance
resource "aws_instance" "vm" {
 ami                     = "ami-05912b6333beaa478"
 instance_type           = "t2.micro"
 key_name                = "win_instance_key"
 security_groups         = ["${aws_security_group.allow_rdp_http.name}"]
 disable_api_termination = true
 monitoring              = true
```

```
  tags = {
    Name = "Windows Server 2019"
  }
}

#creating a security group to allow RDP & HTTP access.
resource "aws_security_group" "allow_rdp_http" {
 name        = "allow_rdp_http"
 description = "Allows HTTP access"

 ingress {
   from_port   = 3389
   to_port     = 3389
   protocol    = "tcp"
   cidr_blocks = ["0.0.0.0/0"]
 }

 ingress {
   from_port   = 80
   to_port     = 80
   protocol    = "tcp"
   cidr_blocks = ["0.0.0.0/0"]
 }
}
```

- ***disable_api_termination*** If true, enables EC2 Instance Termination Protection.

- ***monitoring*** If true, the launched EC2 instance will have detailed monitoring enabled.

3. Create an instance via terraform.

Terraform has 5 main commands, which are as follows:

- **terraform init**
  - This prepares your working directory for other commands.

```
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E1$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.0"...
- Installing hashicorp/aws v3.75.2...
- Installed hashicorp/aws v3.75.2 (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.
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E1$
```

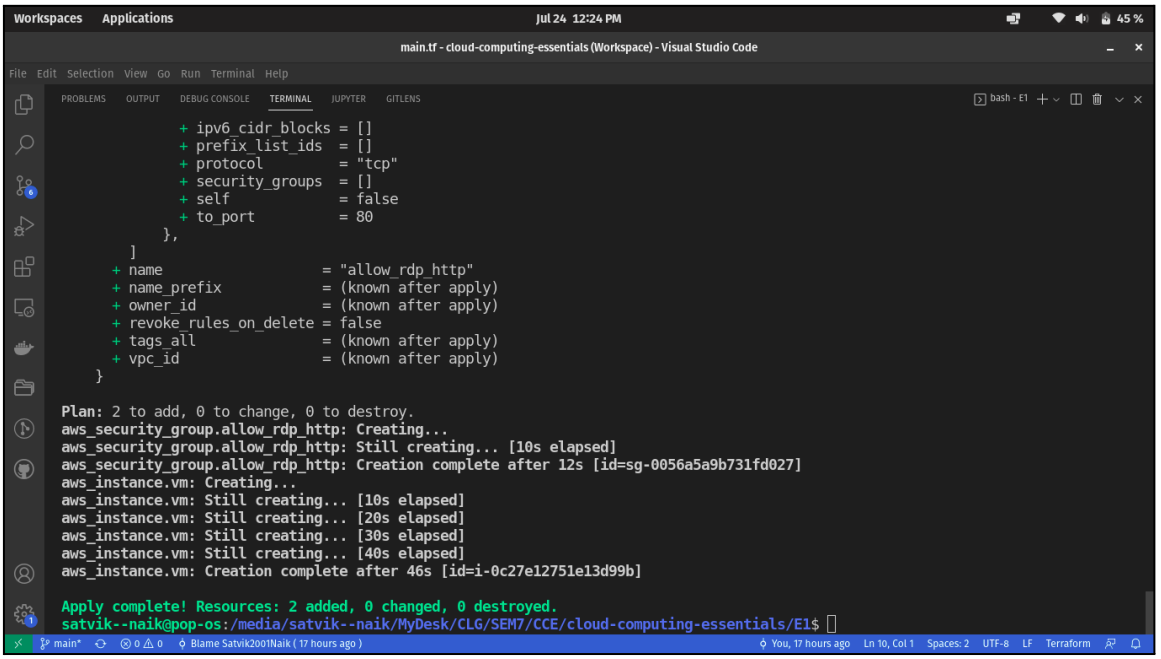
- **terraform validate**
  - Checks whether the configuration is valid or not.

```
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E1$ terraform validate
Success! The configuration is valid.

satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E1$
```

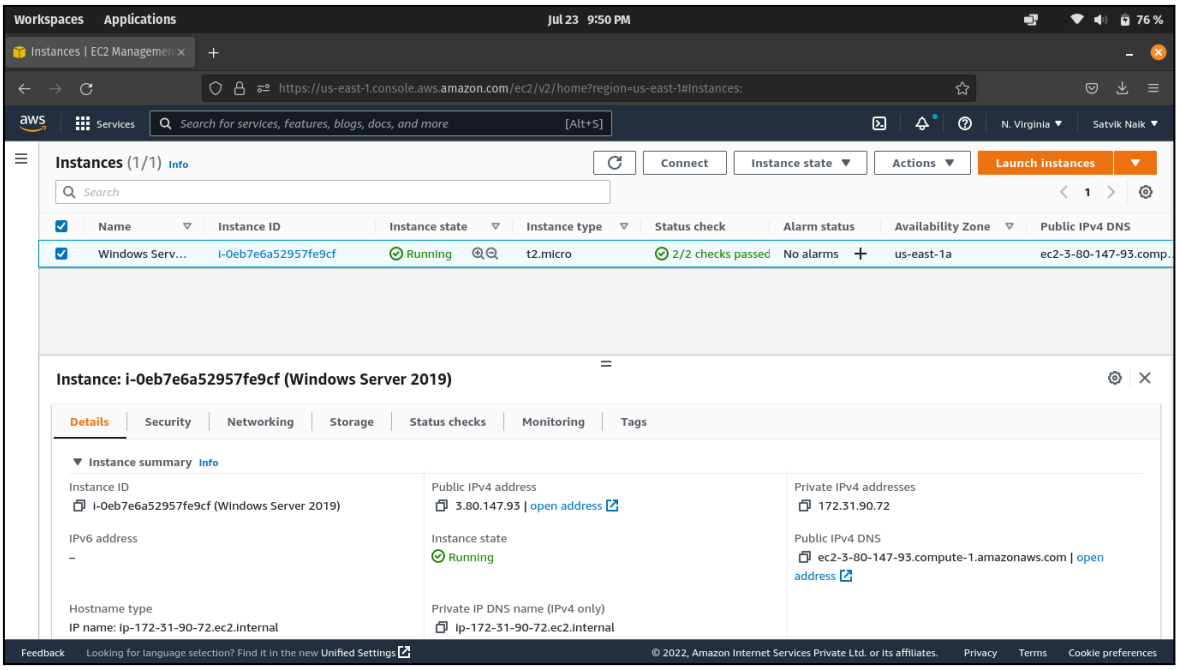
- **terraform plan**
  - It shows changes required by the current configuration.

```
      }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 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.
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E1$
```

- **terraform apply**
  - Creates or updates infrastructure.



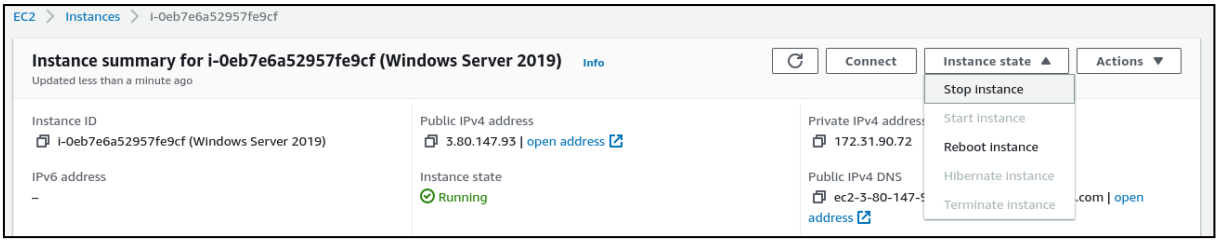
- After applying the code, the instance has been successfully ***created*** & is in a ***Running*** state.



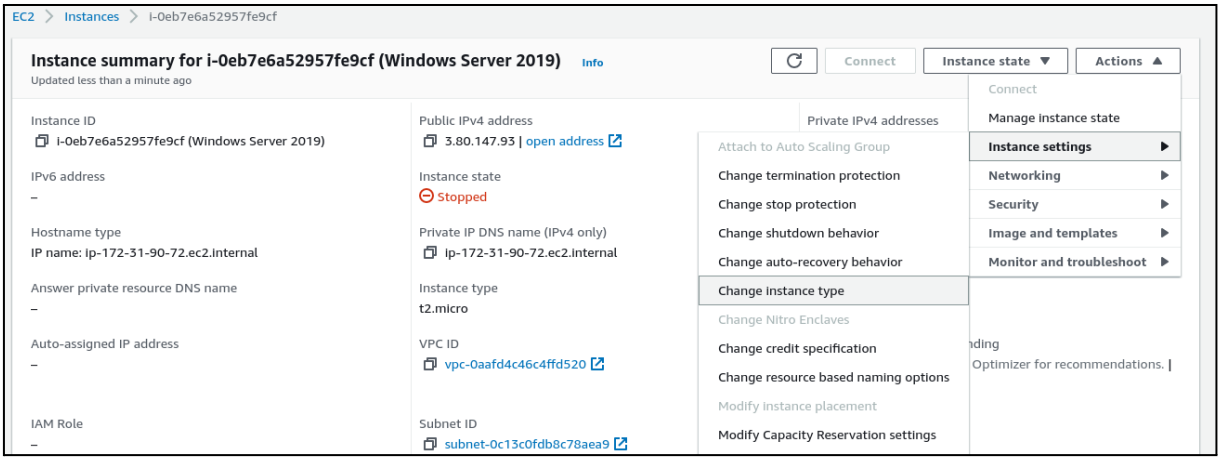
4. Resizing Amazon EC2 instance to scale.

- In order to resize amazon EC2 instance to scale, you must ***stop the existing instance*** and then you should ***change the instance type***.

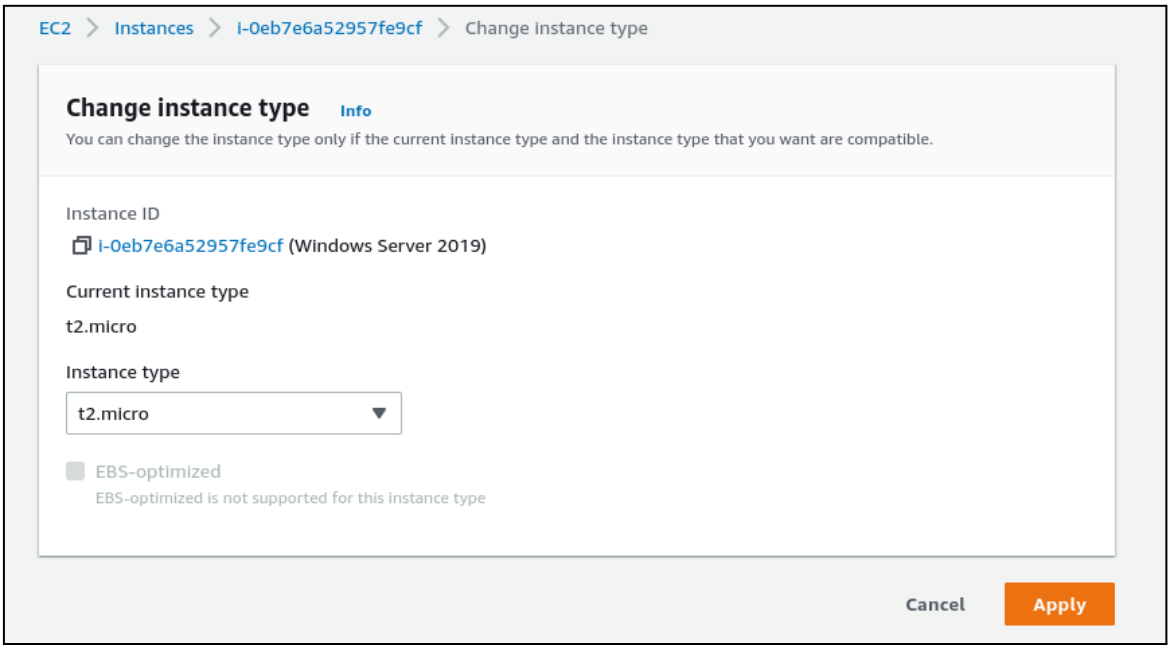
- Stop instance



- Change instance type



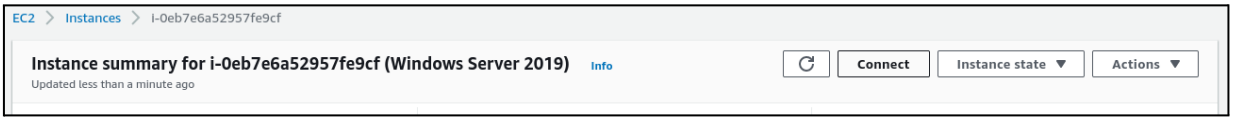
- By default t2.micro is selected, for scaling other instance types could be selected and further applied. (i.e. t2.xlarge, t2.2xlarge etc…)



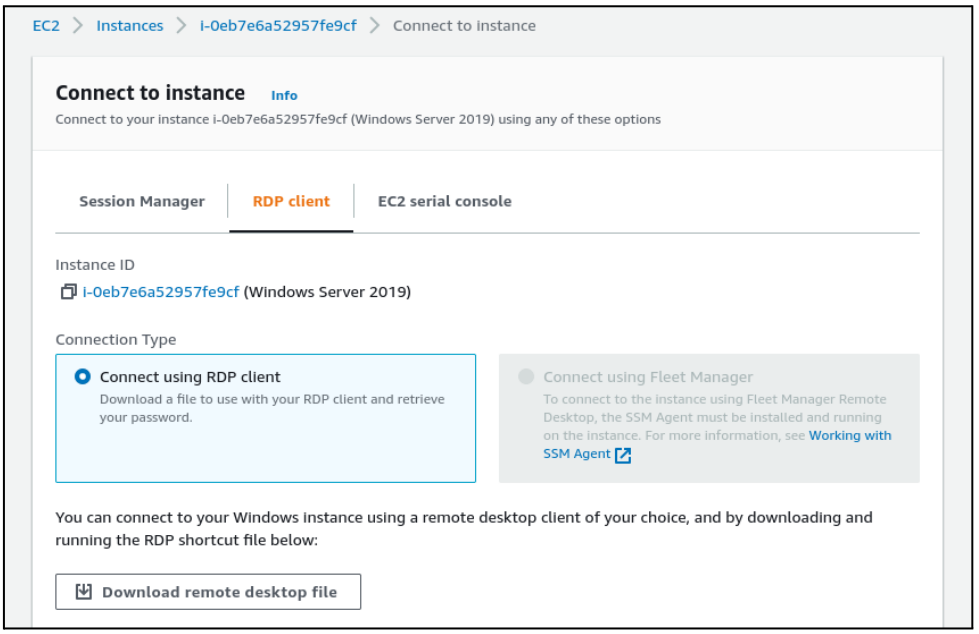
- Select the particular instance type and click on ***'Apply'*** & then start your instance.

5. Connect to instance via RDP.

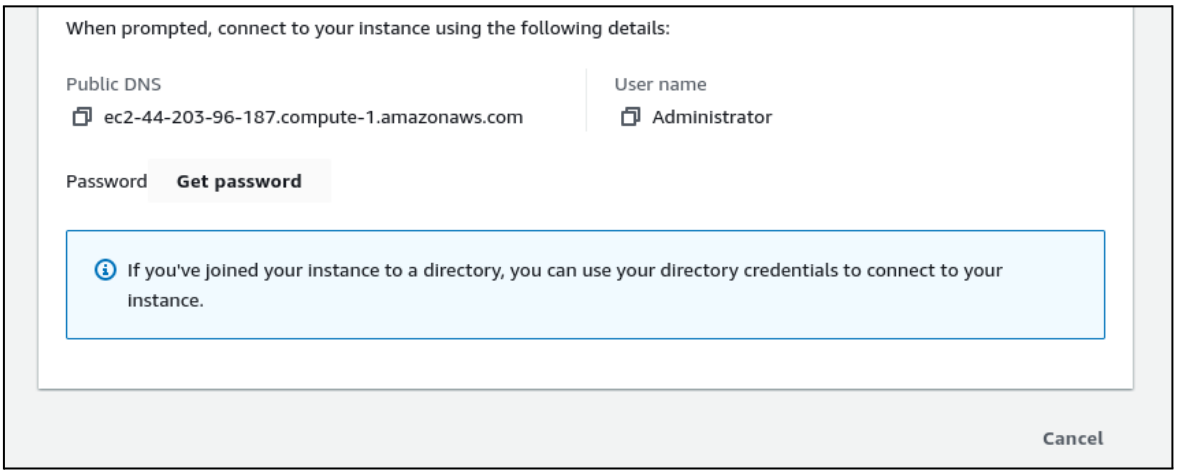
- Go to the instance and click on ***'Connect'***.



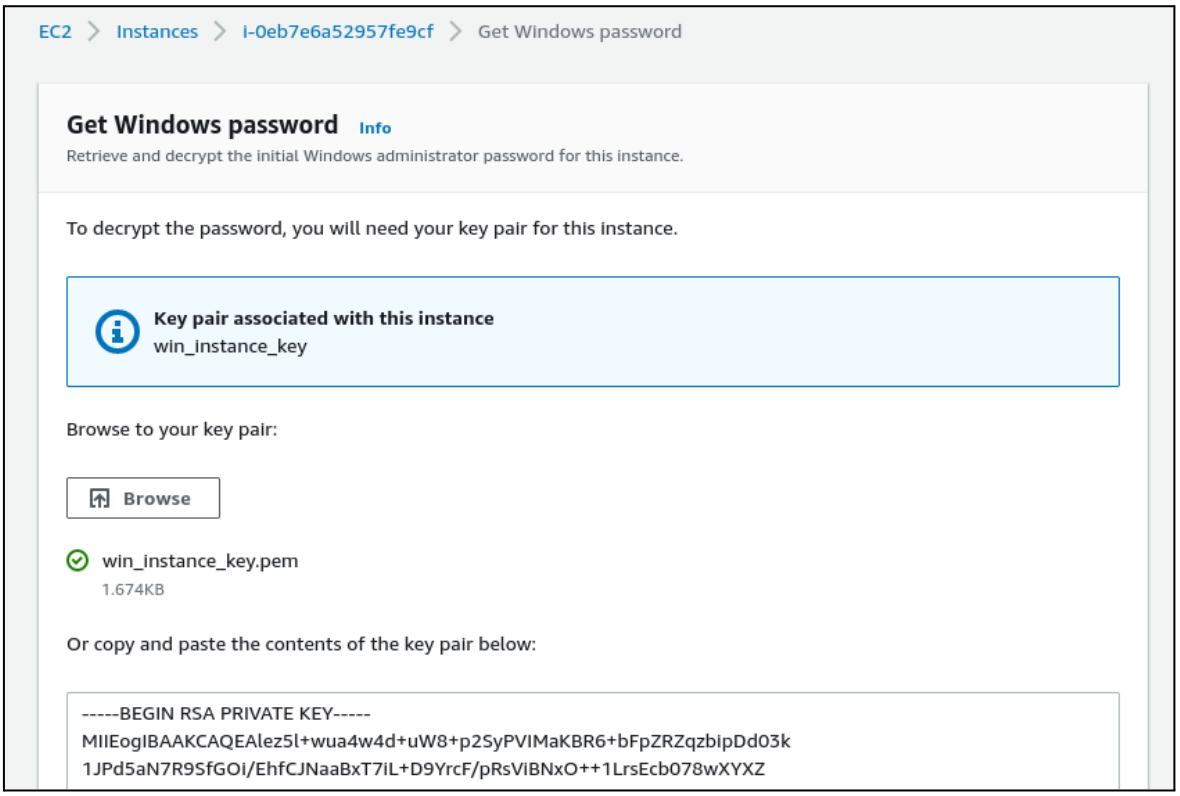
- Select RDP client.

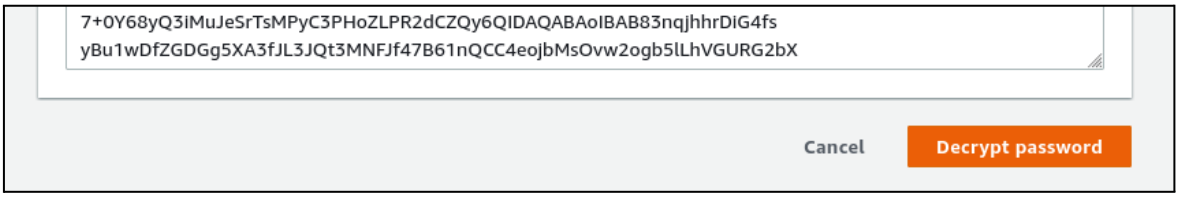


- Click on 'Get Password'.



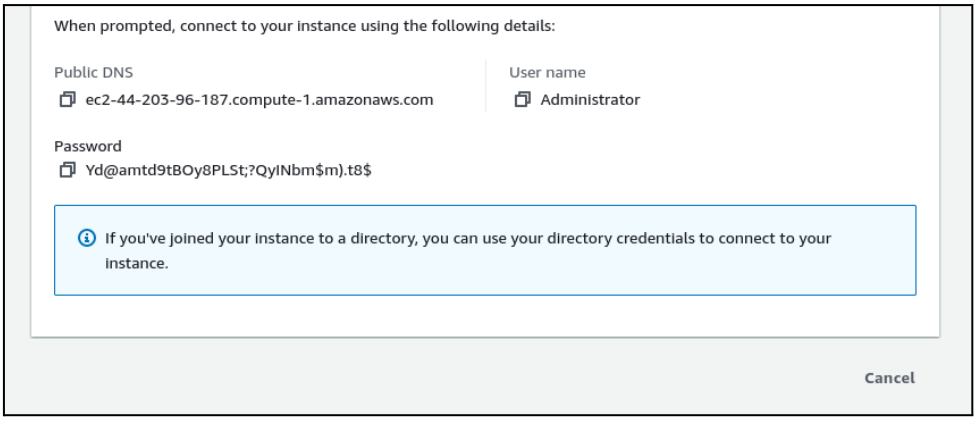
- Browse the key file which was created earlier to decrypt the password.



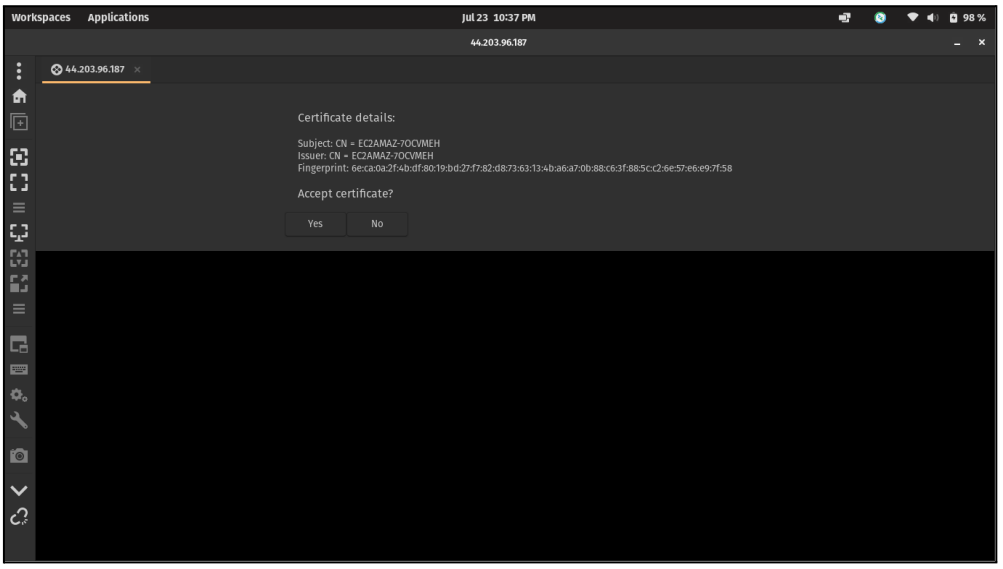


Click on ***'Decrypt Password'***.

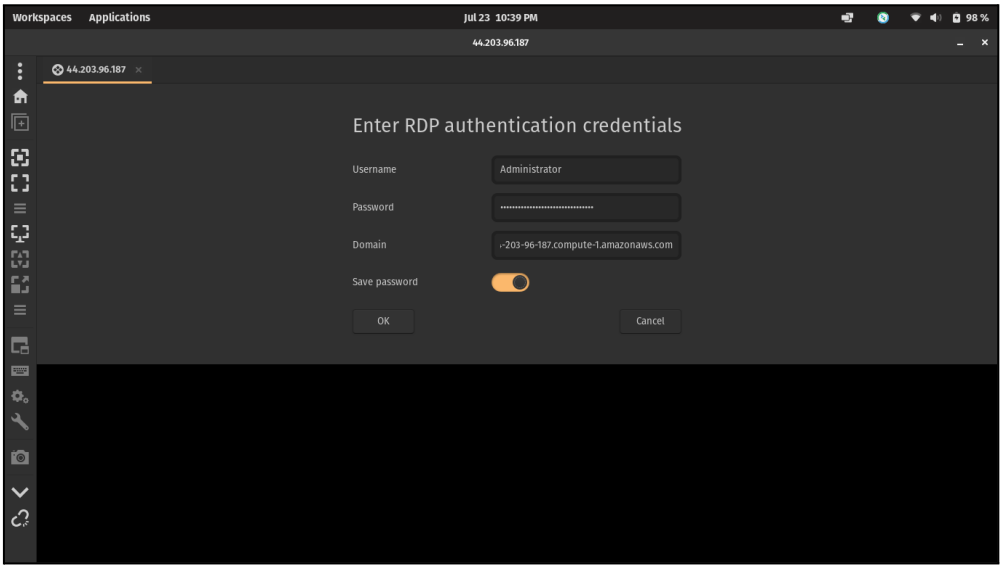
- You will get another password which will help you while connecting to the instance via RDP.



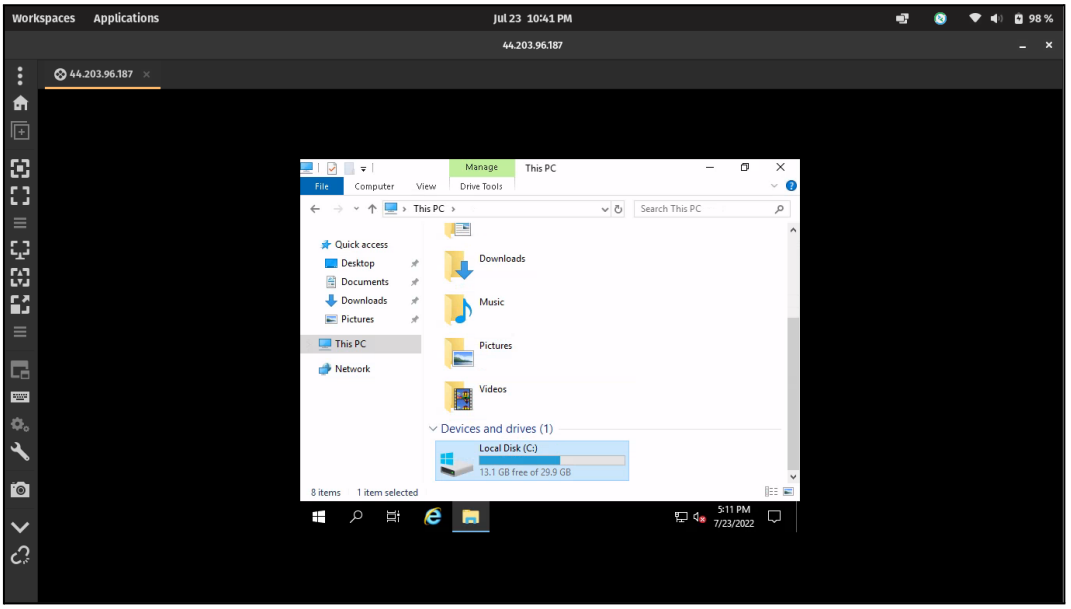
- Use the ***public IP*** of the instance to get connected & accept the certificate.



- Enter username, password & domain.

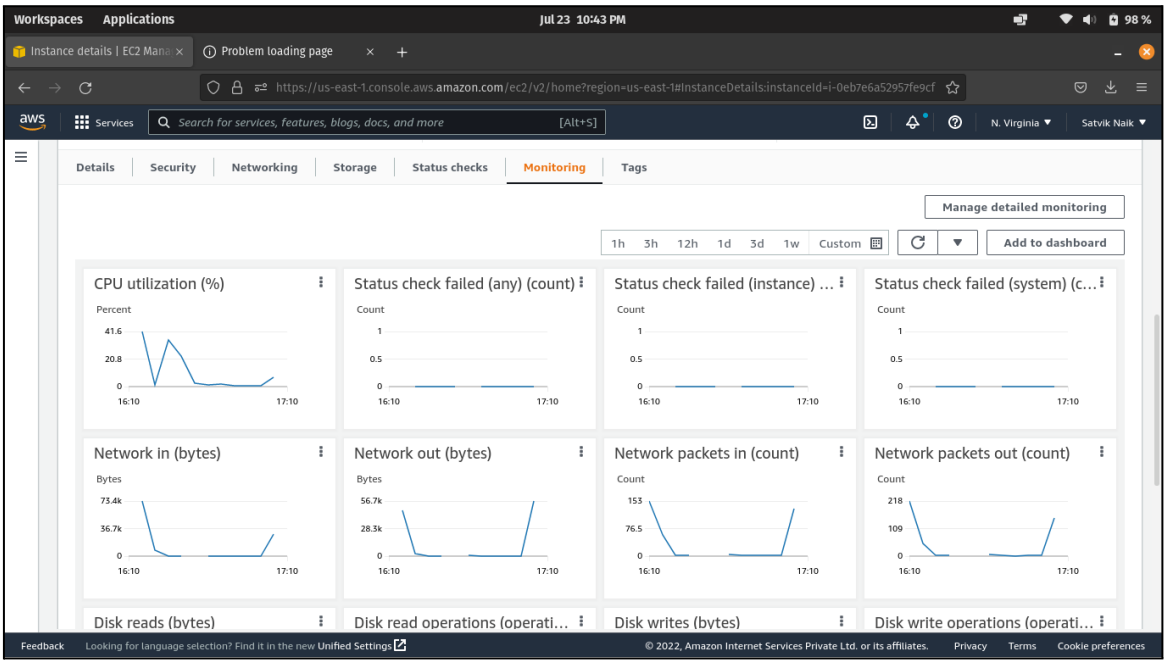


- Successfully accessing the instance via RDP.

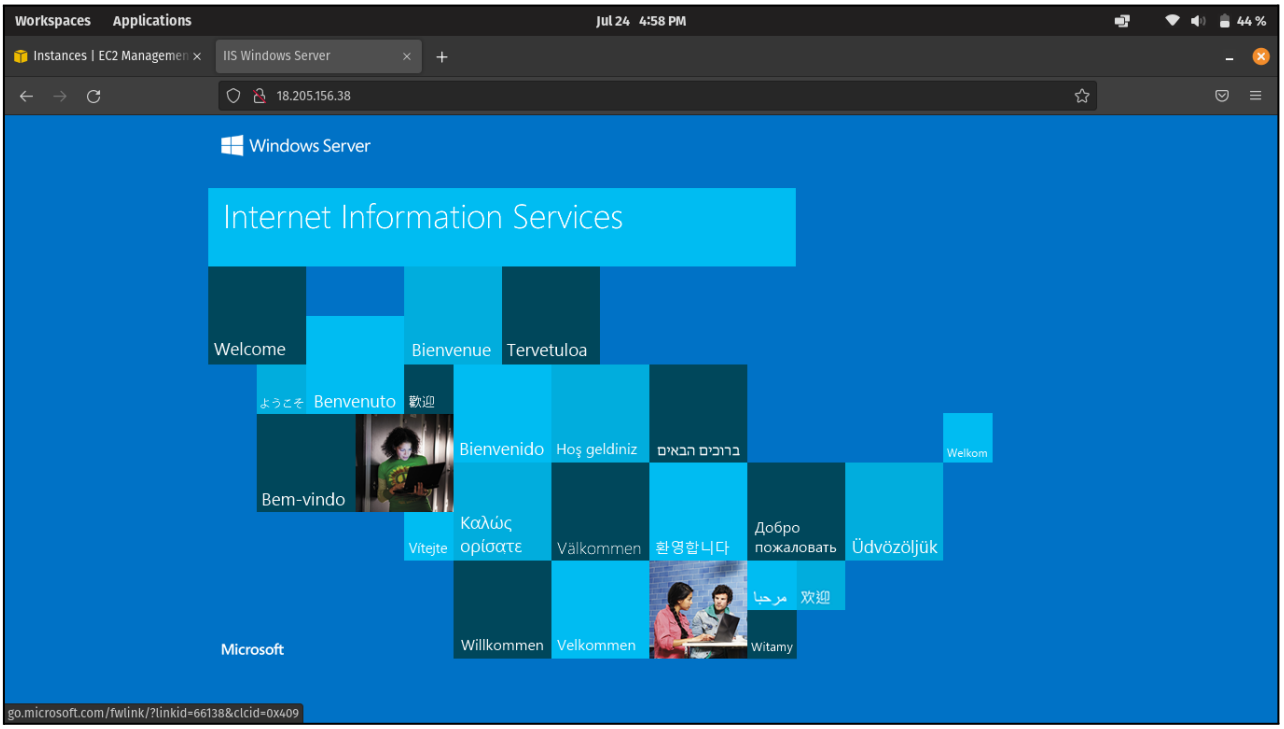


6. Monitoring EC2 Instance.

- EC2 instances can be monitored by visualizing various parameters such as disk R/W operations. CPU usage, network packets etc…

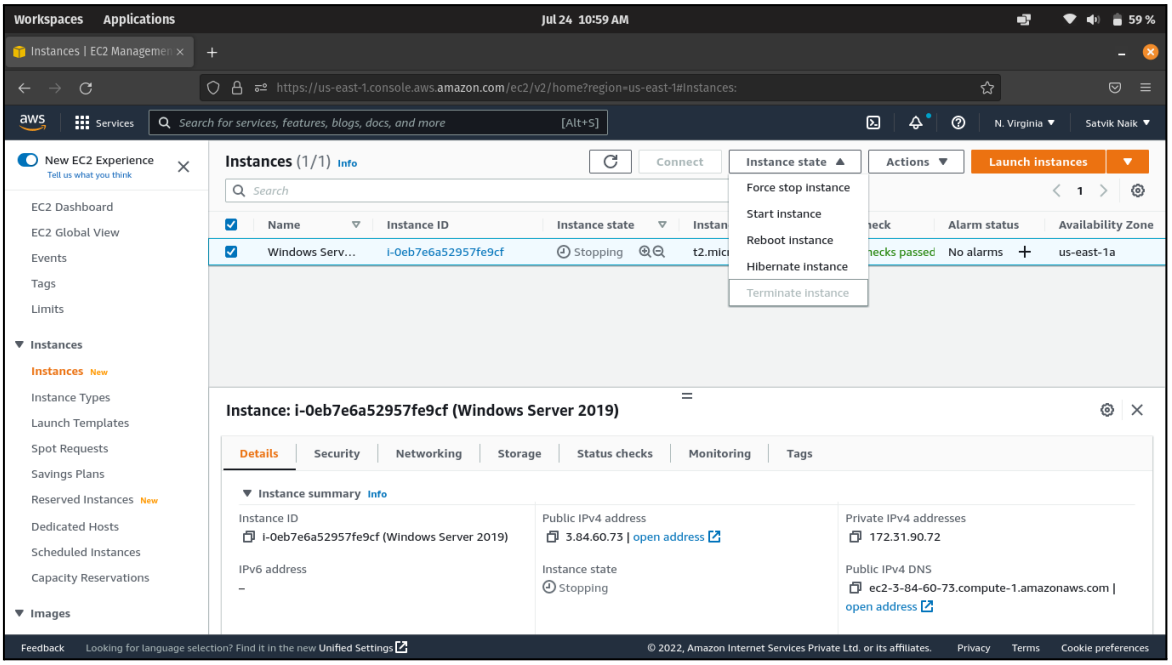


7. Modify security group to allow HTTP access.

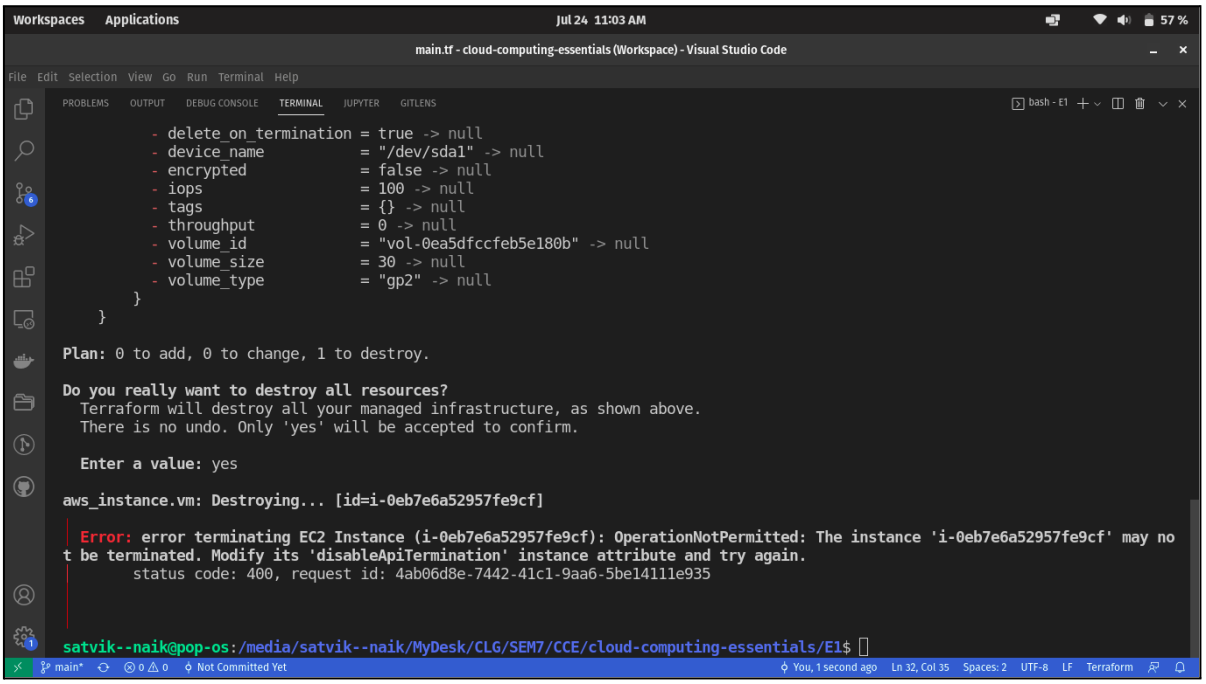


8. Test termination protection.

- In case you try to terminate the instance, it will prompt you with a message showing ***"Termination protection is enabled for one or more selected instances."***

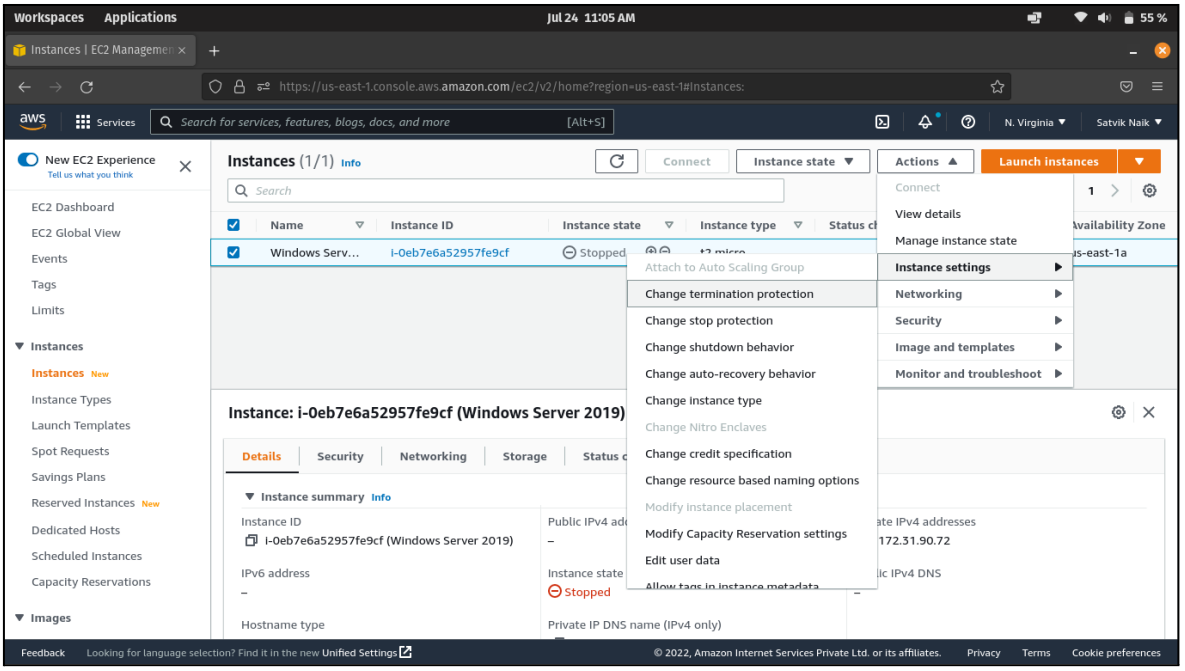


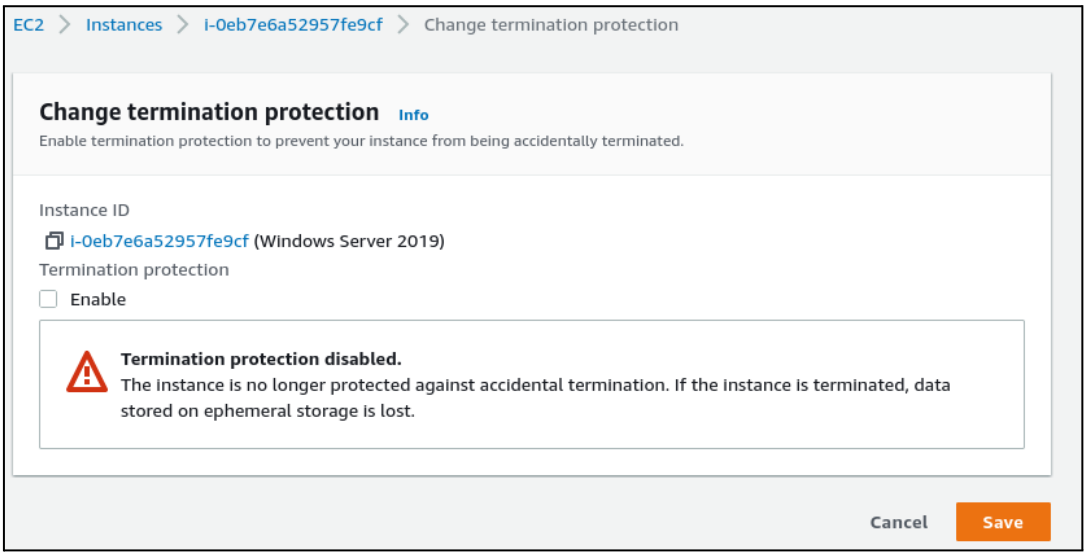
- In case if you try to do it via ***'terraform destroy'*** command then also it will show you the following error message. (i.e. OperationNotPermitter).



9. Terminate your EC2 Instance.

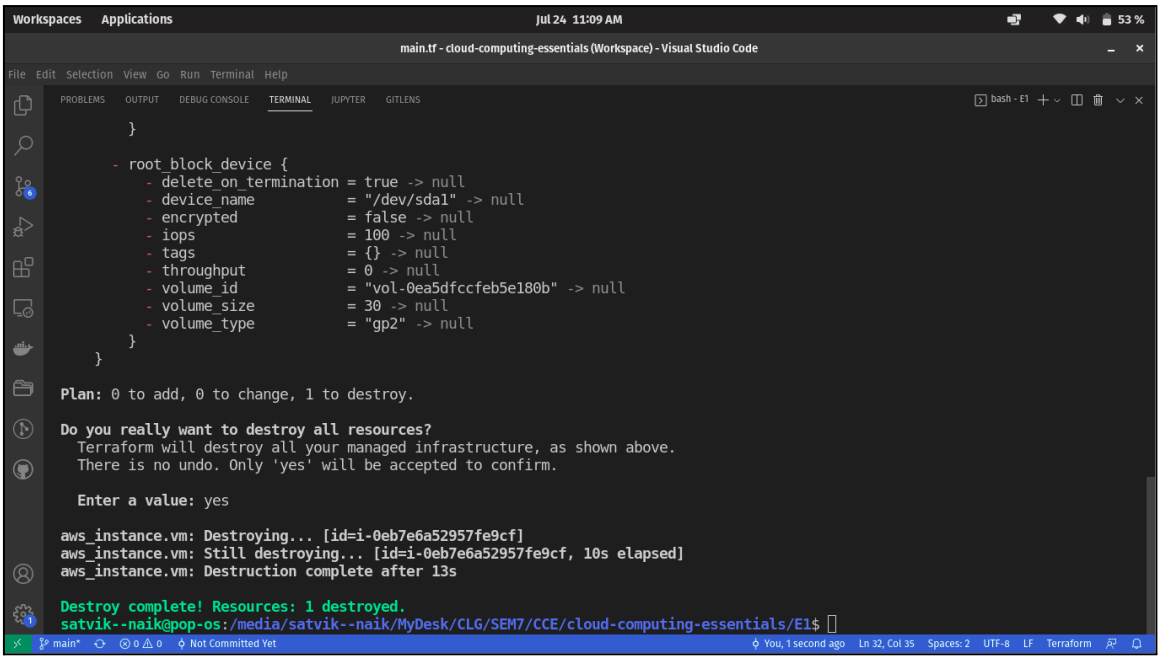
- In order to terminate your instance, first disable the termination protection.





Click on ***'Save'***.

- Now on trying to give the ***'terraform destroy'*** command, it will terminate the instance.



10. EC2 limits.

- 20 instances per region.
- 5 Elastic IP Addresses.
- Price variations.