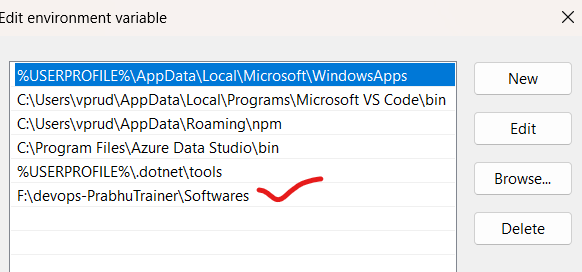
1. Software

Download terraform -> <https://developer.hashicorp.com/terraform/install>

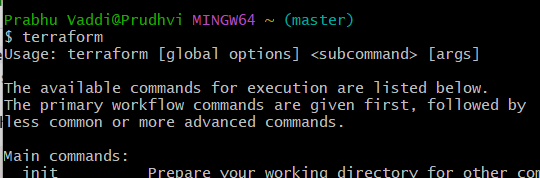
Download aws cli => <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>



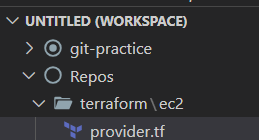
1. Setup Environment variable for terraform.exe path.



1. Check terraform version in git bash or cmd



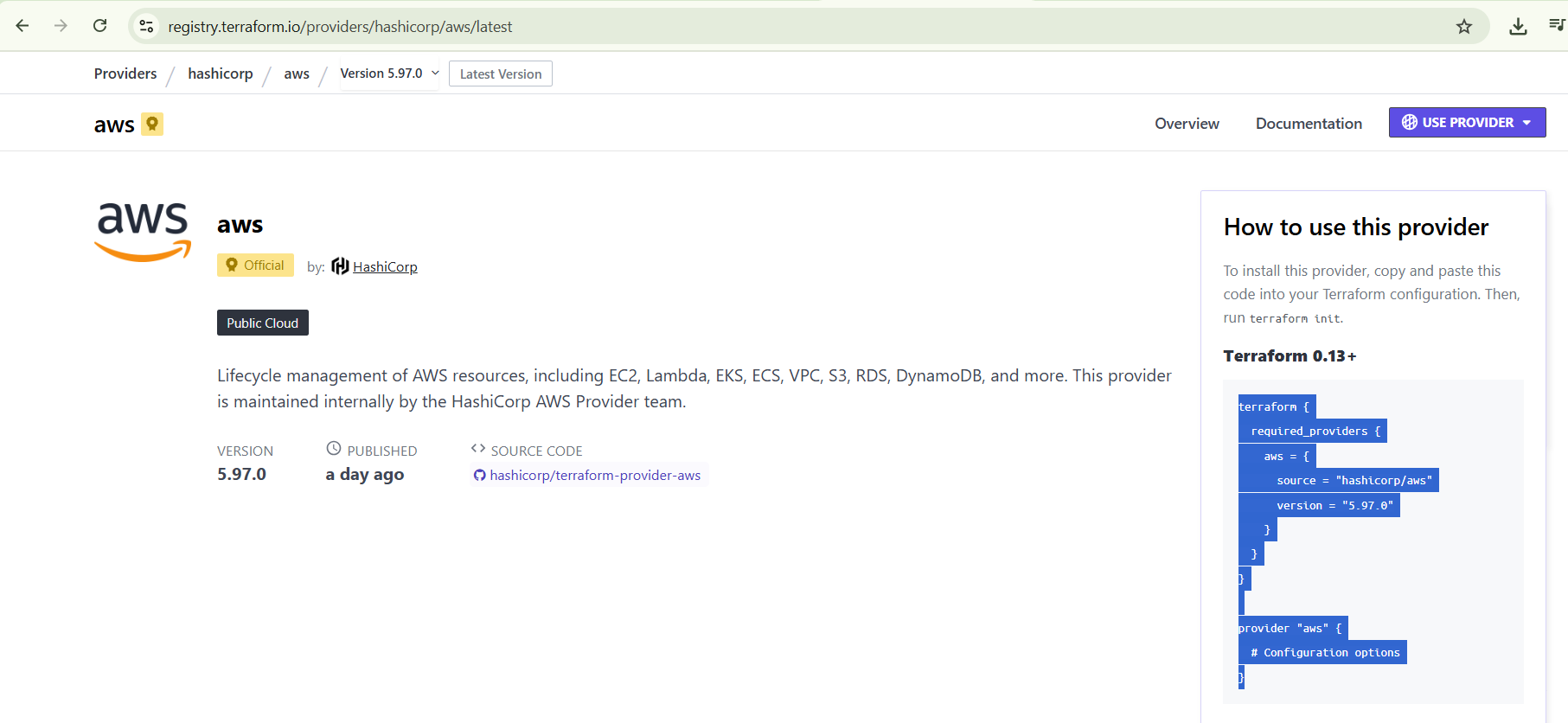
1. Install Visual studio code and create terraform/ec2 folder with provider.tf file

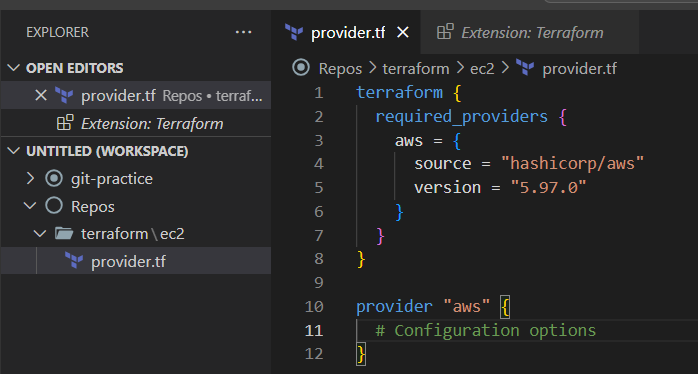


1. Providers : aws

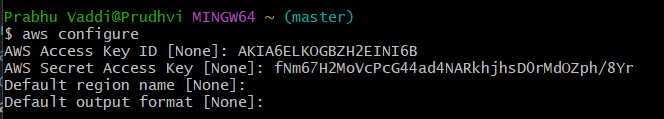
<https://registry.terraform.io/providers/hashicorp/aws/latest>

Copy the code & paste into provider.tf in vs code

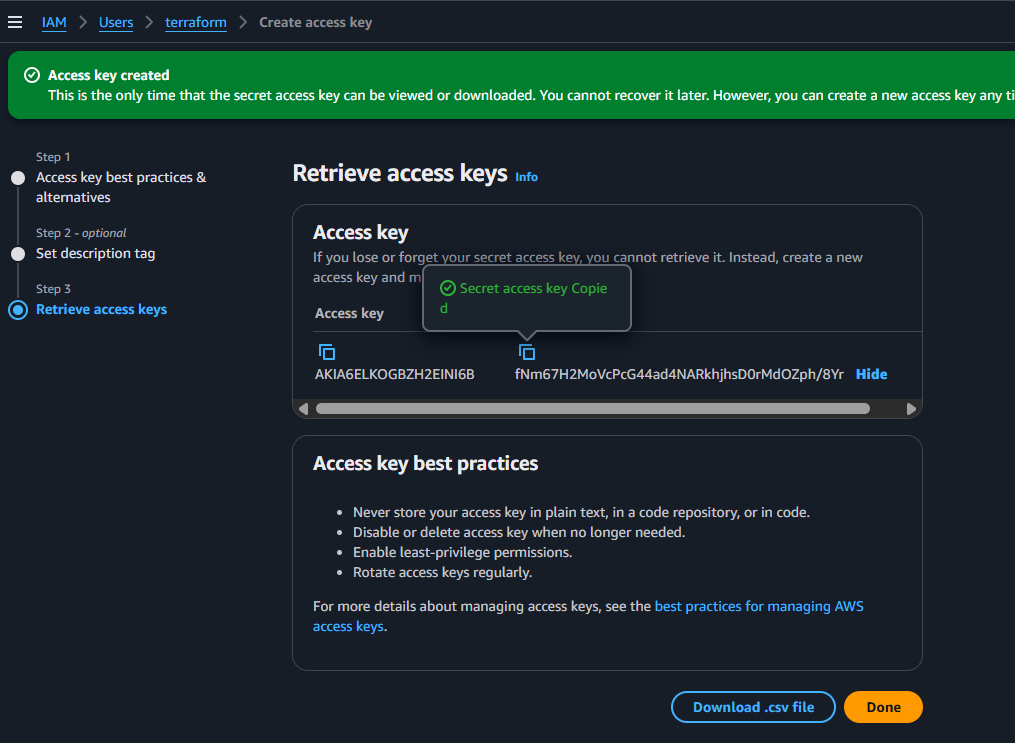




1. Set AWS configure cmd in git bash.



1. Provide access and secret key from aws console.
2. Create a user-terraform using IAM
3. Policies- administrationaccess
4. Once, user created=> security credentials=> create secret/access key



1. Edit the provider.tf with region etc.,



1. Create a new file under ec2=> ec2.tf

Security\_Group creation

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group>

**Notes**

1. Name
2. Description
3. Ingress =>incoming traffic mandatory
4. Egress => outgoing traffic mandatory

**Security Group Code**

resource "aws\_security\_group" "allow\_ssh\_terraform" {

    name = "allow\_sshh" #allow\_ssh is already there in my account

    description = "Allow port no 22 for ssh access"

    #usually we allow everything in egress

    egress {

    from\_port        = 0

    to\_port          = 0

    protocol         = "-1"

    cidr\_blocks      = ["0.0.0.0/0"]

    ipv6\_cidr\_blocks = ["::/0"]

  }

  ingress {

    from\_port        = 22

    to\_port          = 22

    protocol         = "tcp"

    cidr\_blocks      = ["0.0.0.0/0"] #allow from everyone

    ipv6\_cidr\_blocks = ["::/0"]

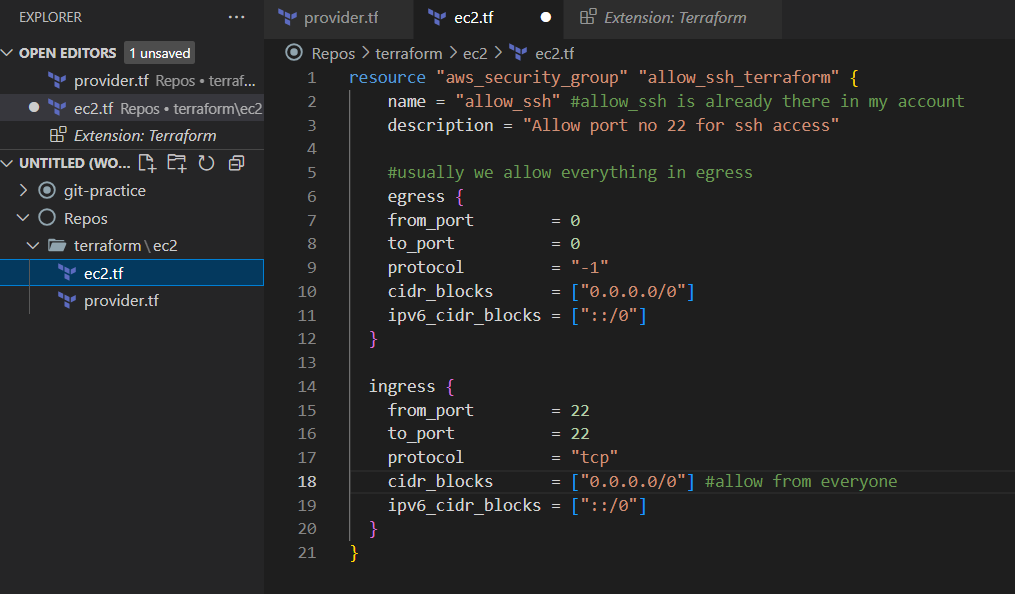
  }

    tags = {

    Name = "allow\_sshh"

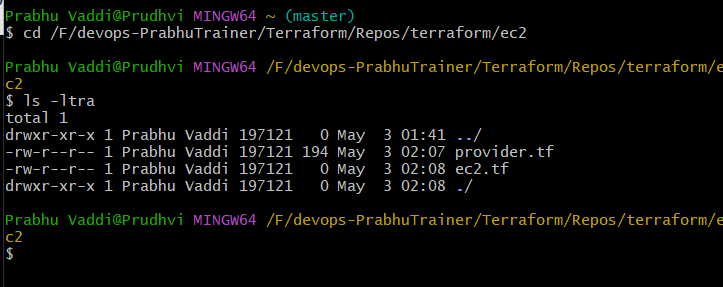
  }

}

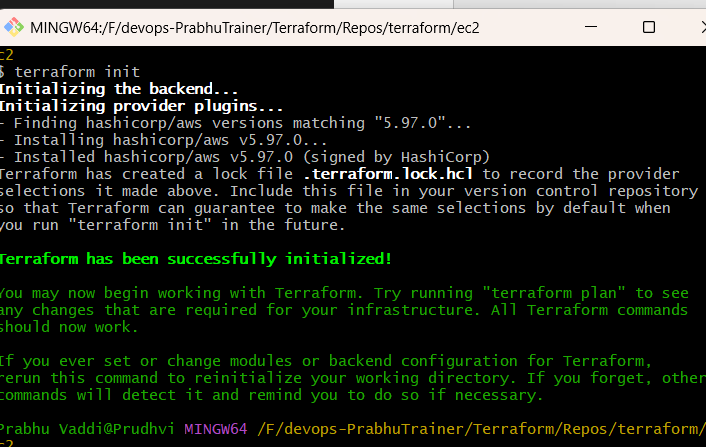


1. Terraform init => connect with provider and download it.

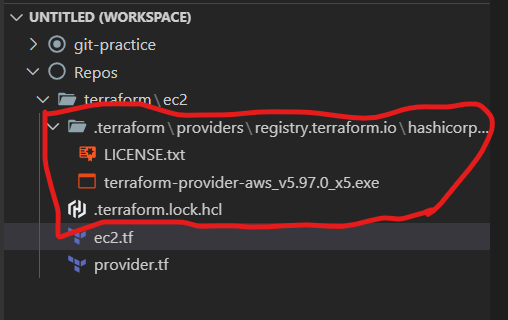
Whereever tf files, that directory is eligible to run the terraform init operation



CMD : terraform init



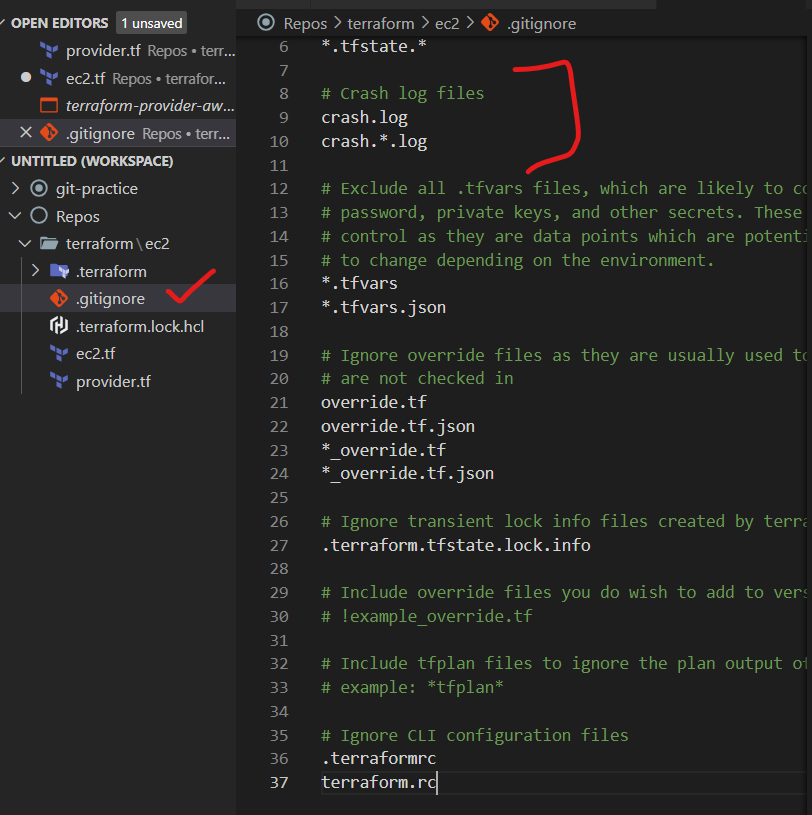
Open vs code=> few files added/download



1. GIT IGNORE => Mandatory
2. These files are not push into git repo. So, terraform .gitignore helps for us.

<https://github.com/github/gitignore/blob/main/Terraform.gitignore>

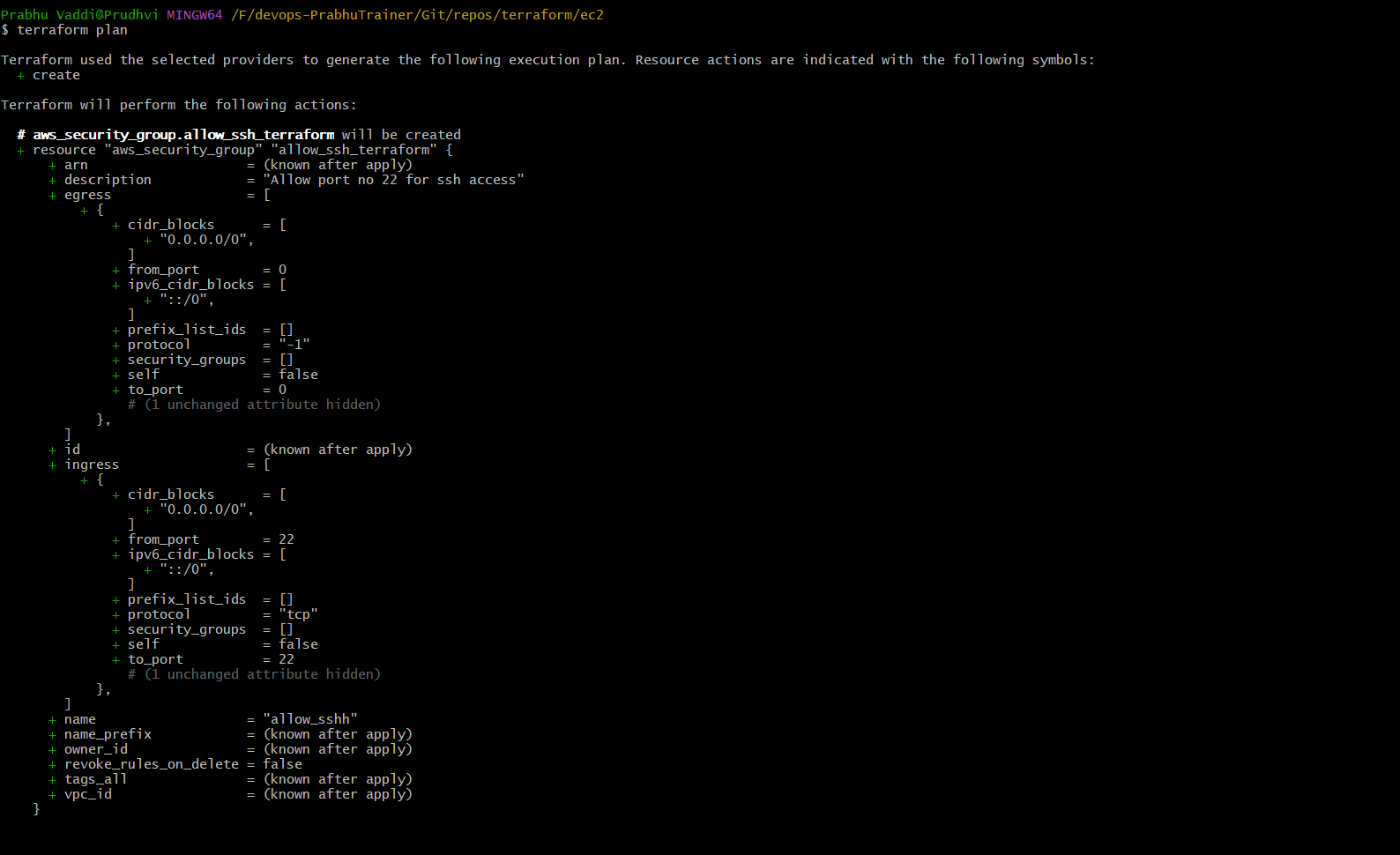
Note : Create a file => .gitignore=> Copy & paste into vs code

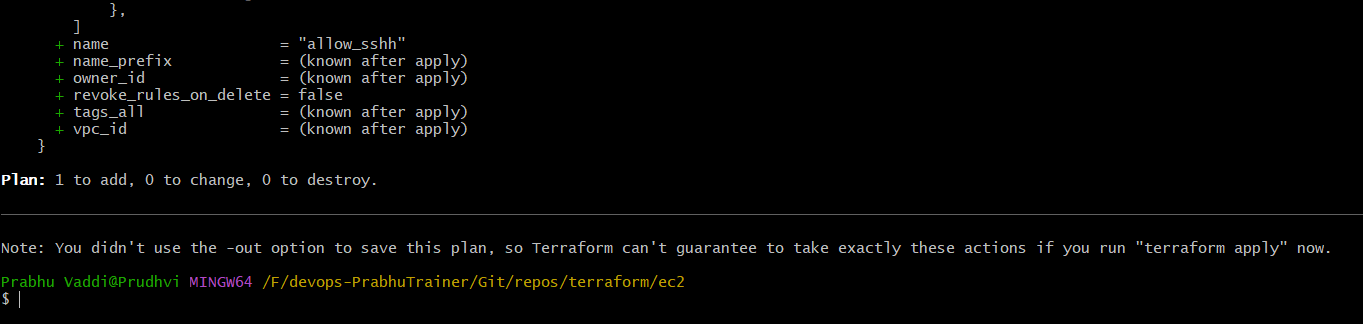


PLAN

1. Tell resources using plan.

Cannot create resources, it is just plan





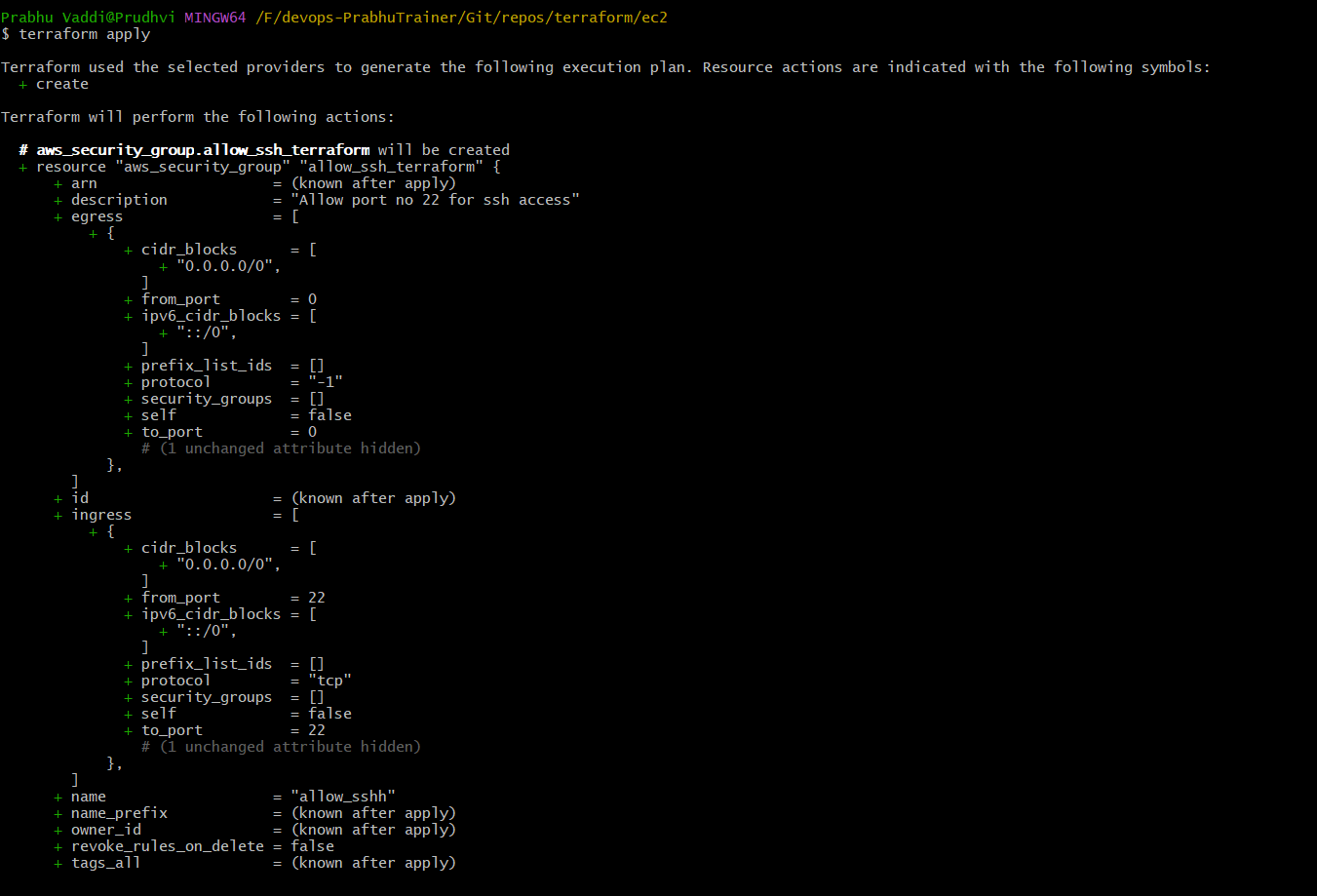
APPLY

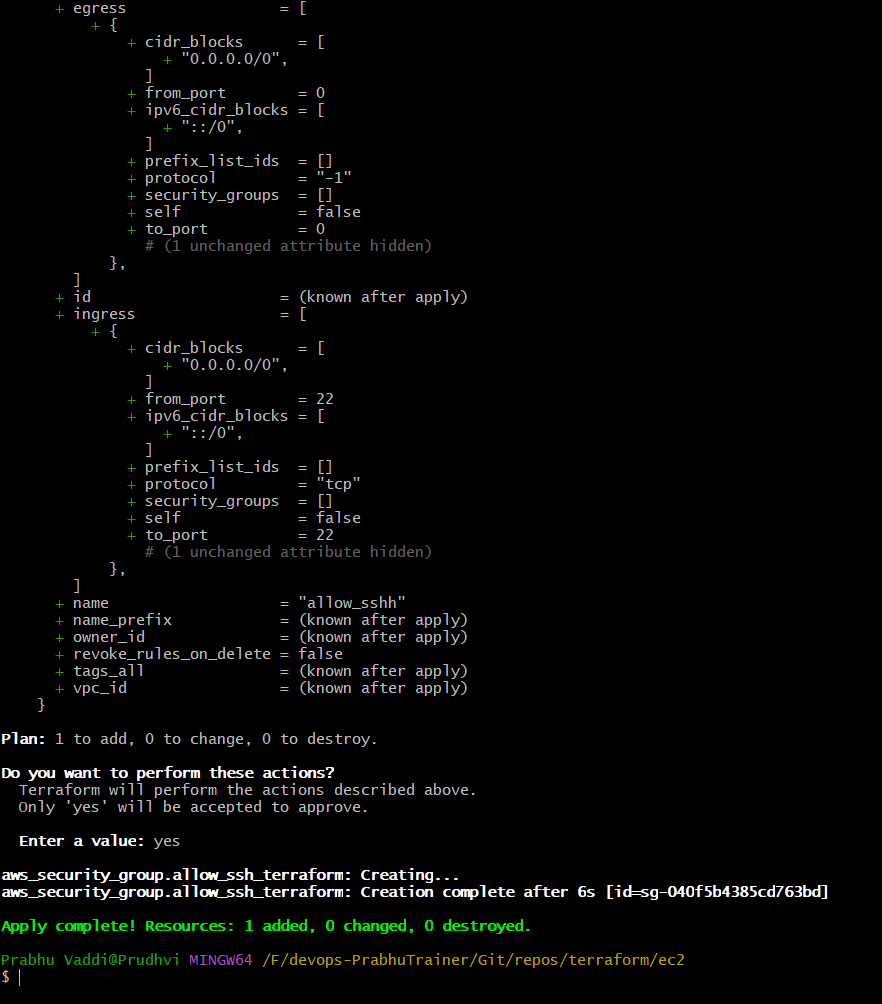
1. Tell resources using apply=> effect the resources.

Plan => stage

Apply => commit

CMD : terraform apply

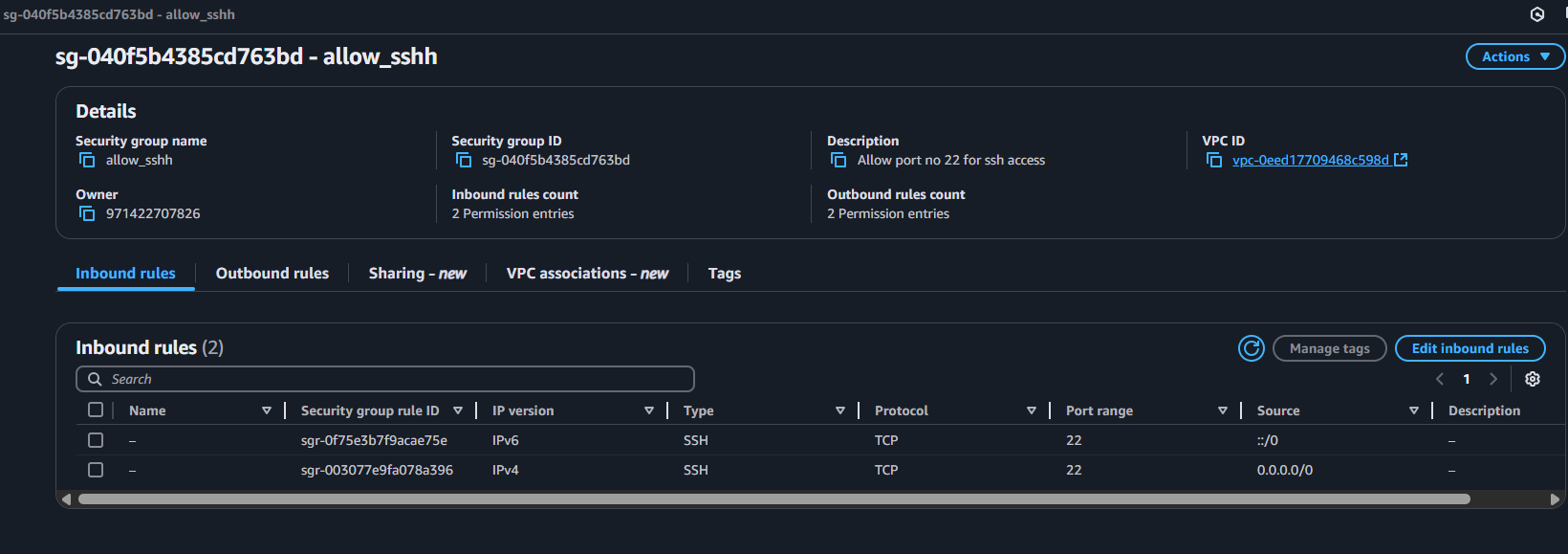




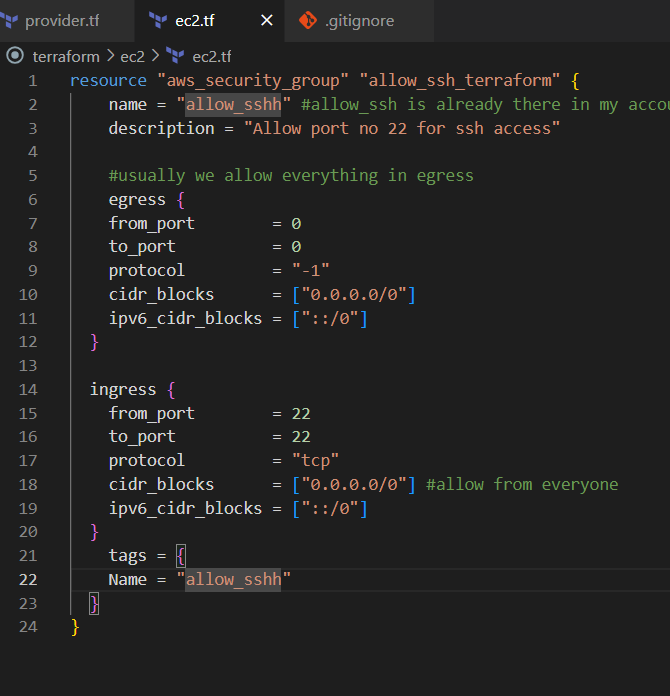
Successfully created security group & ec2

1. Validate the Resources in AWS

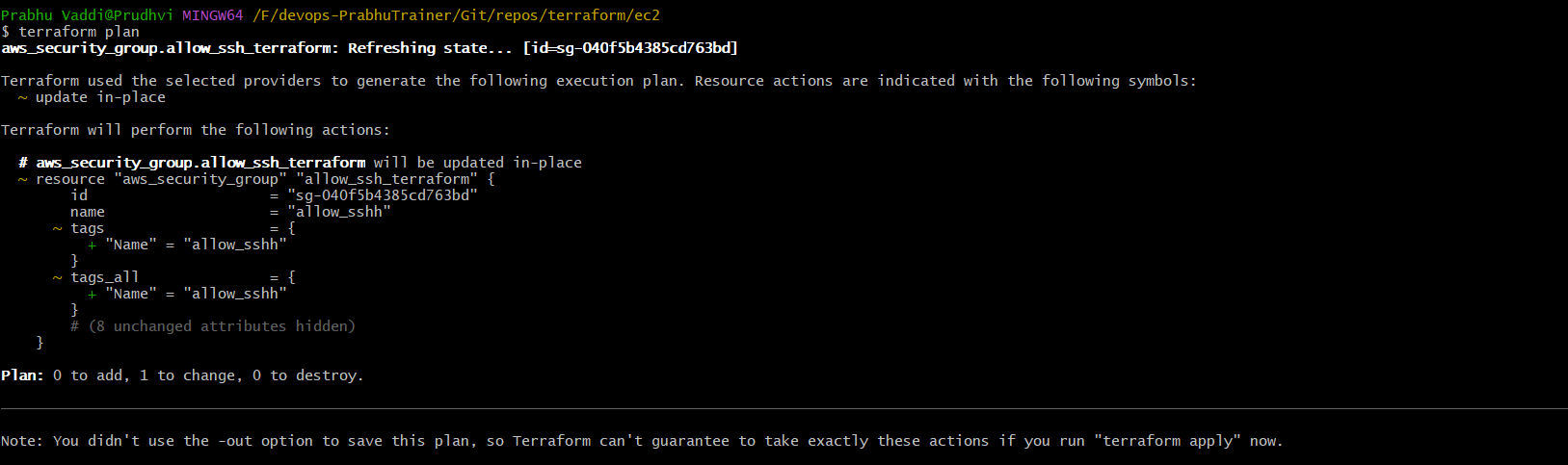
Check Security group



1. Modify the security group with tag => Name : allow\_sshh. Next => plan & apply.

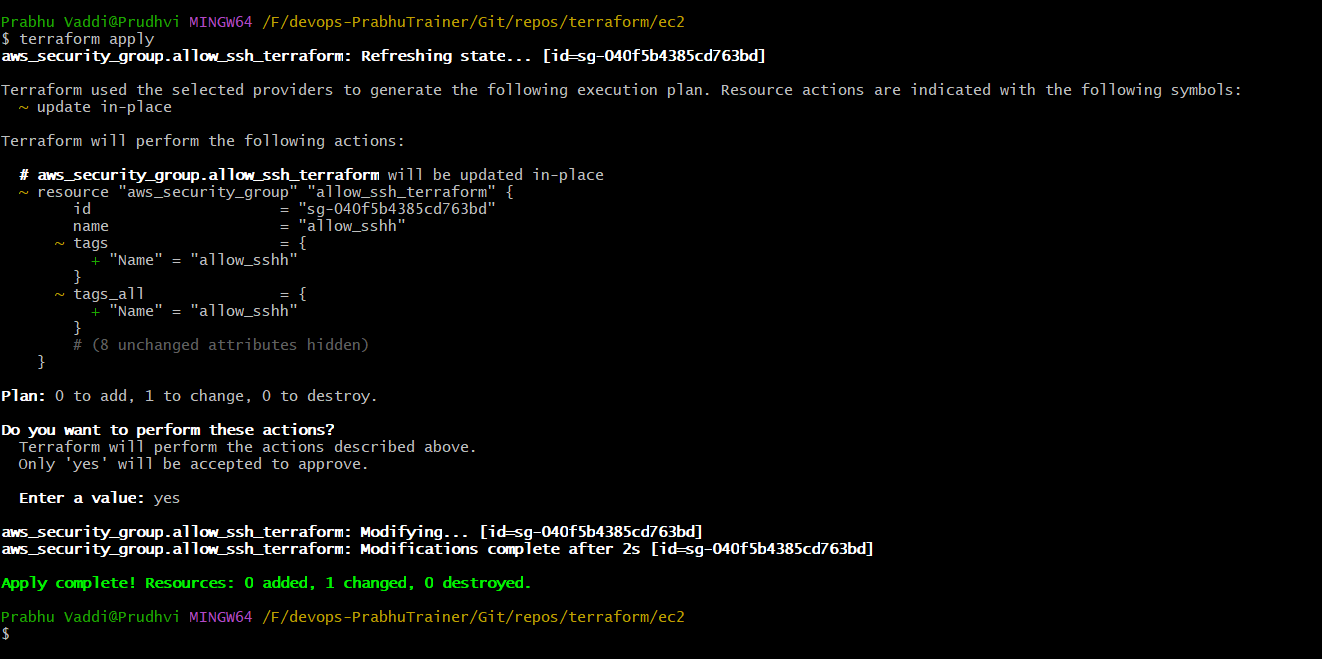


1. Plan in Gitbash

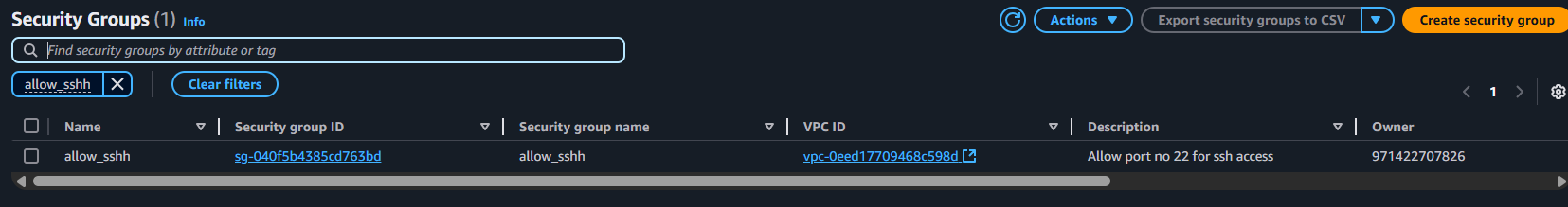


1. CMD: terraform apply -auto-approve

Note : no need to provide “yes”



1. Validate => SG



1. Added EC2 Instance

Code

resource "aws\_instance" "terraform" {

  ami           = "ami-09c813fb71547fc4f"

  instance\_type = "t3.micro"

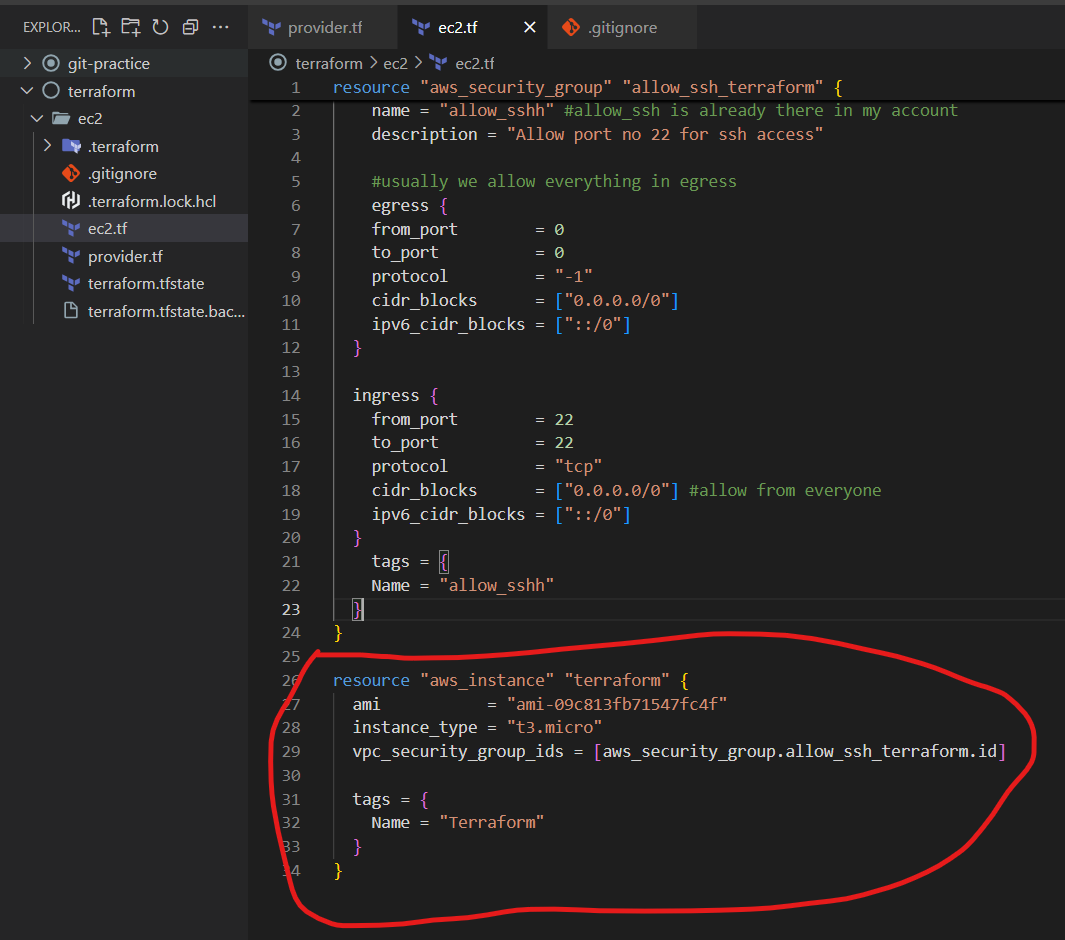
  vpc\_security\_group\_ids = [aws\_security\_group.allow\_ssh\_terraform.id]

  tags = {

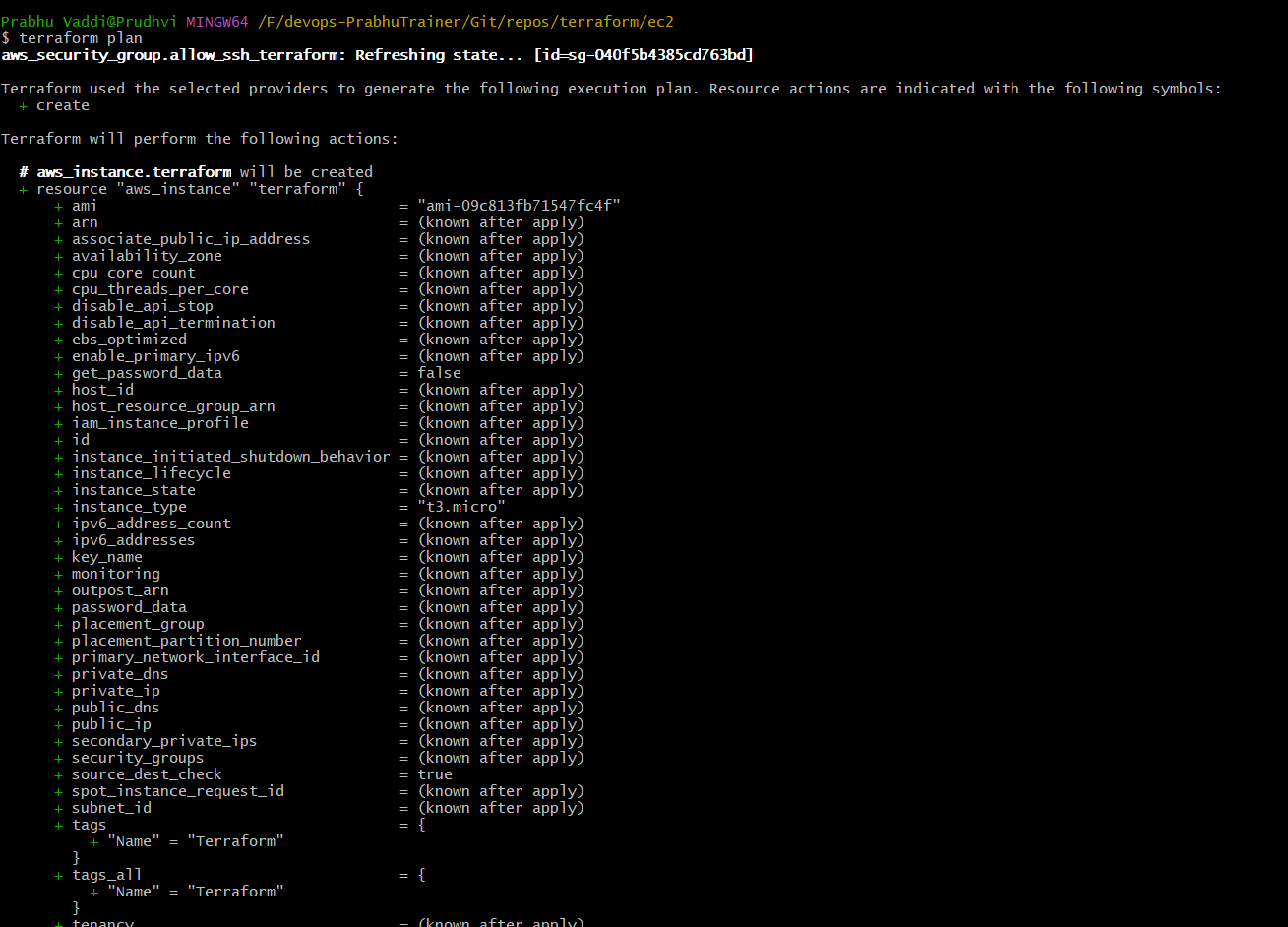
    Name = "Terraform"

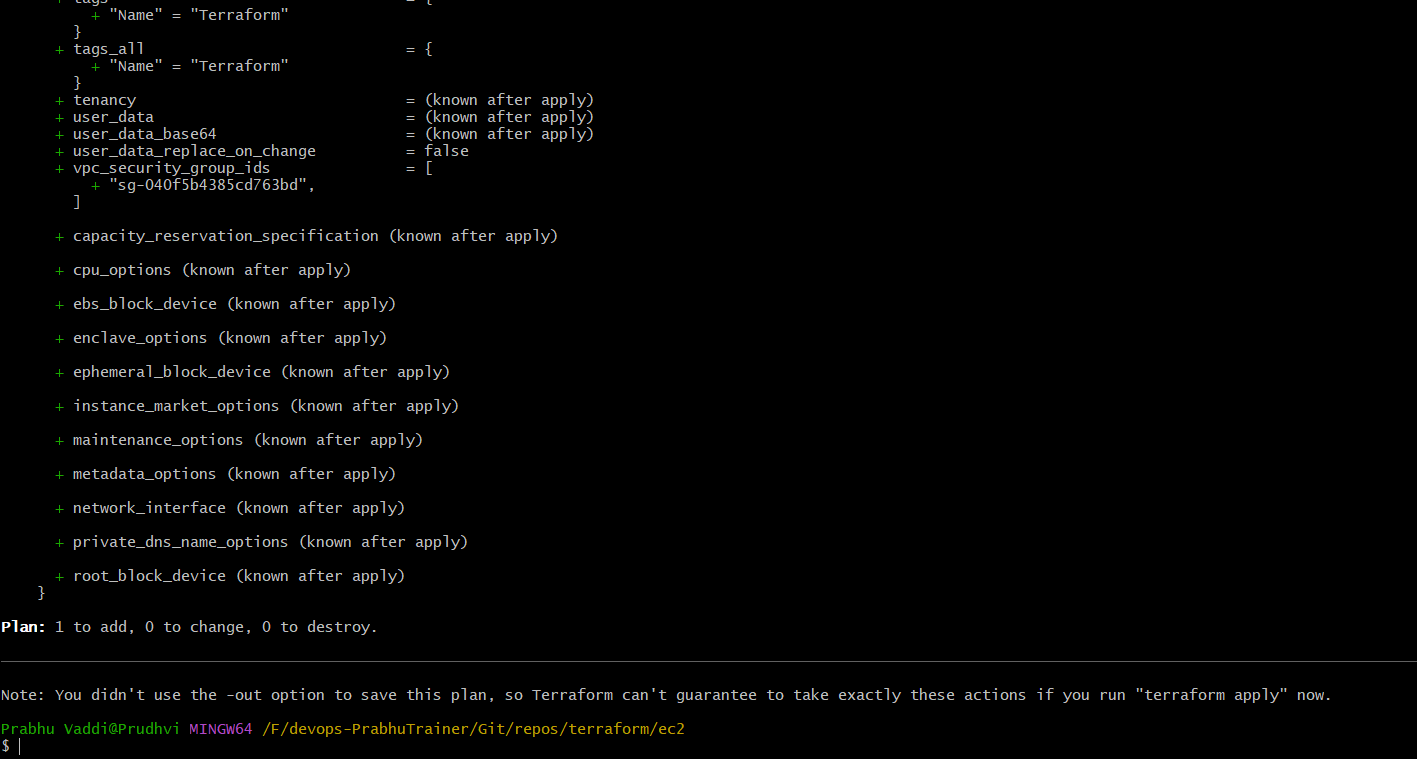
  }

}

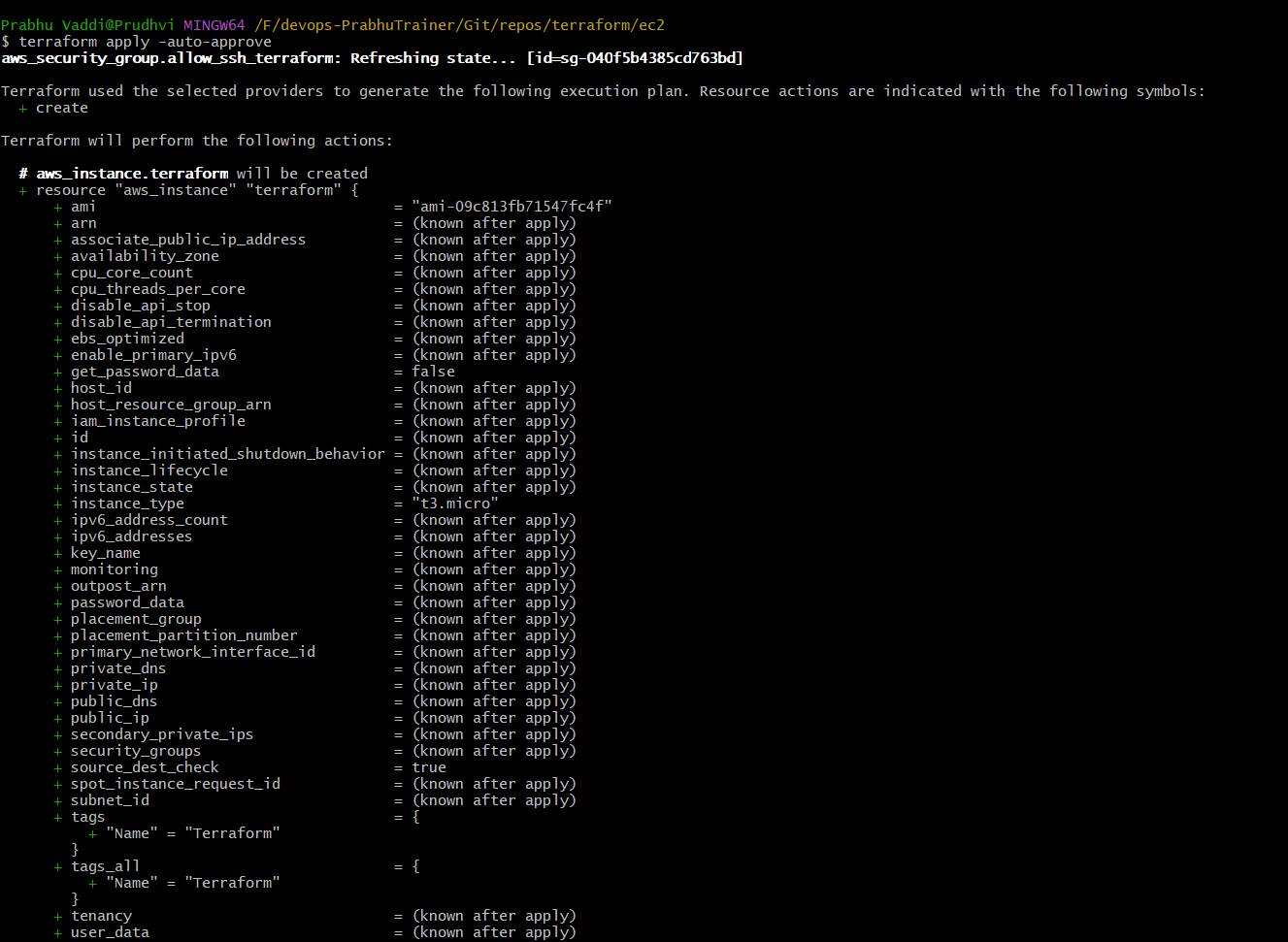


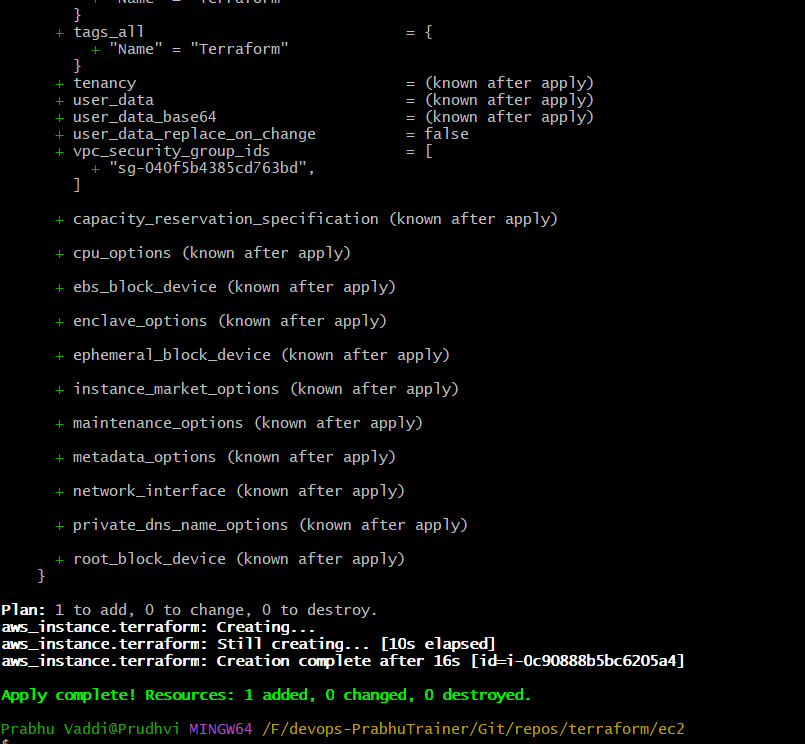
1. Plan & Apply auto save



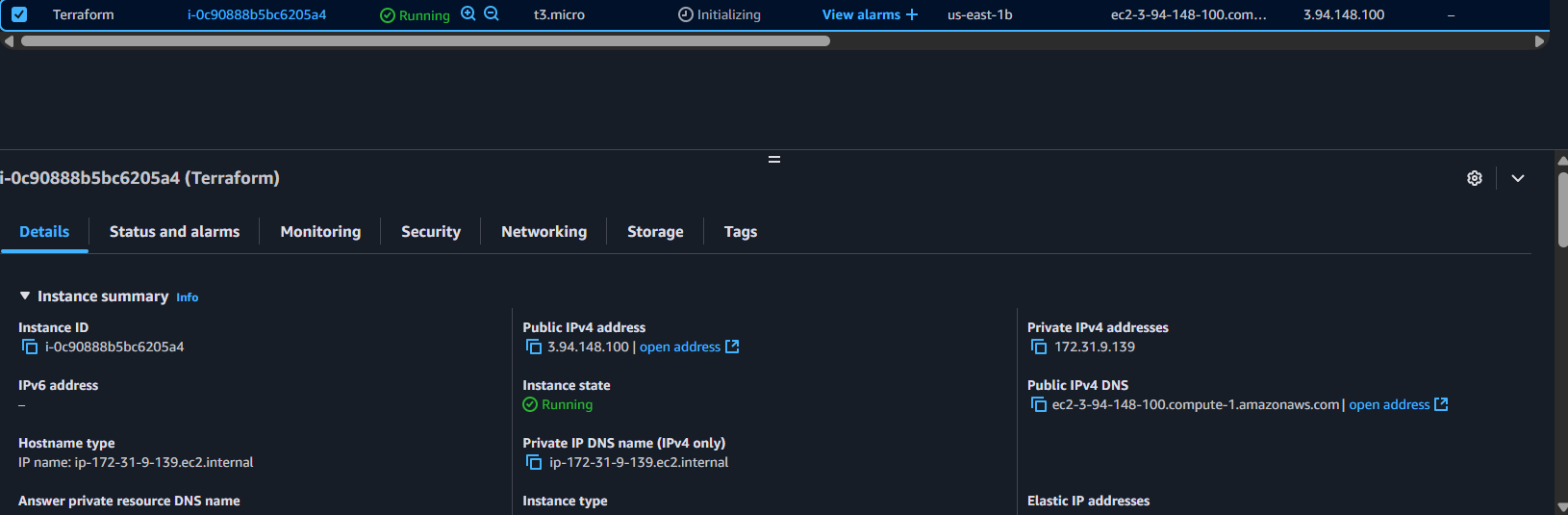


NEXT, Apply -auto-approve

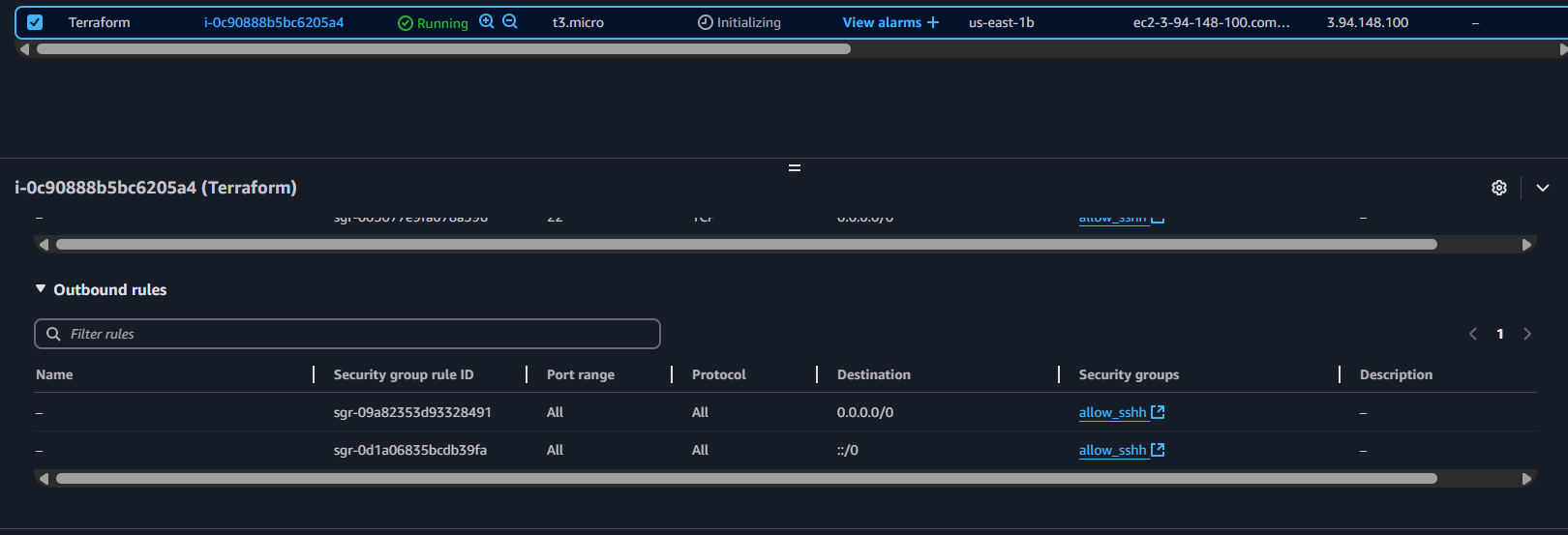




1. Validate the EC2 Instance

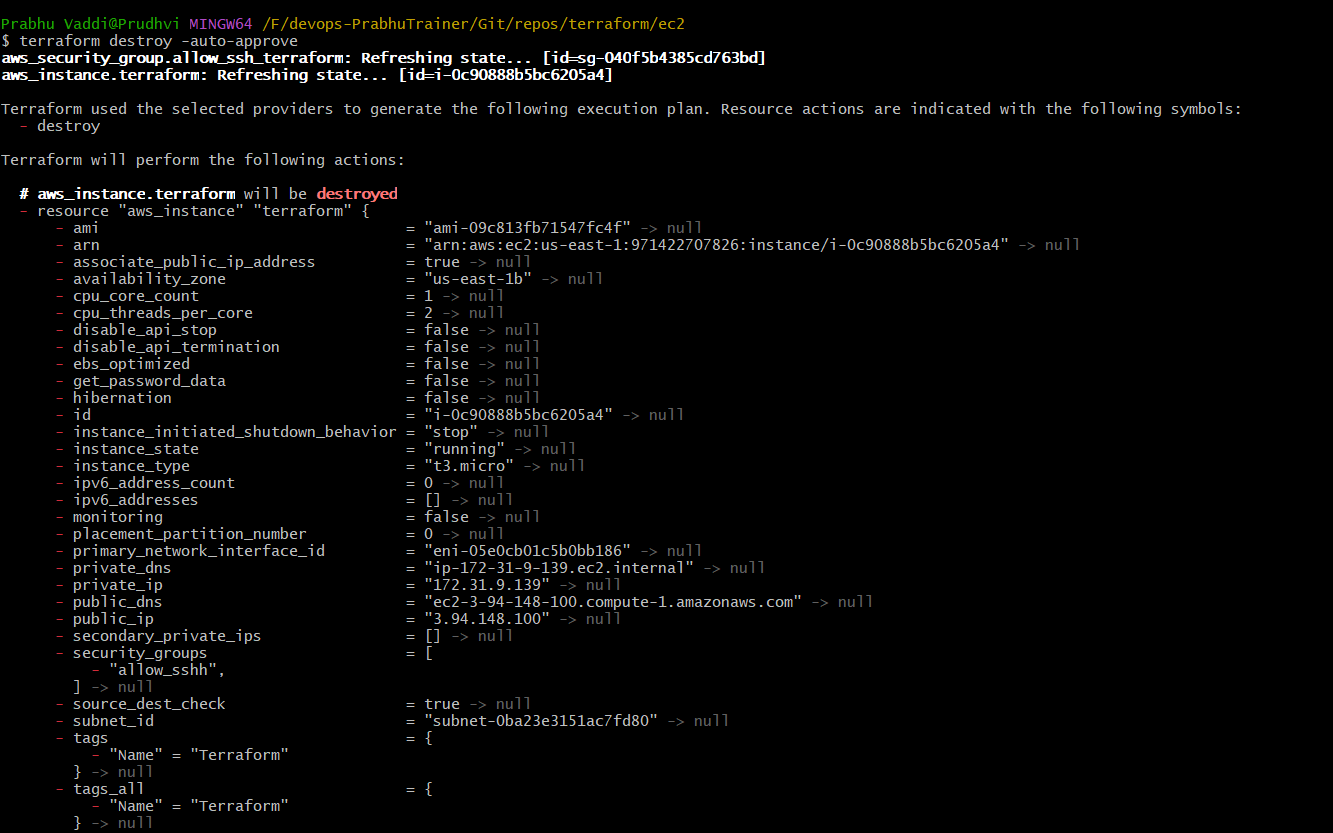


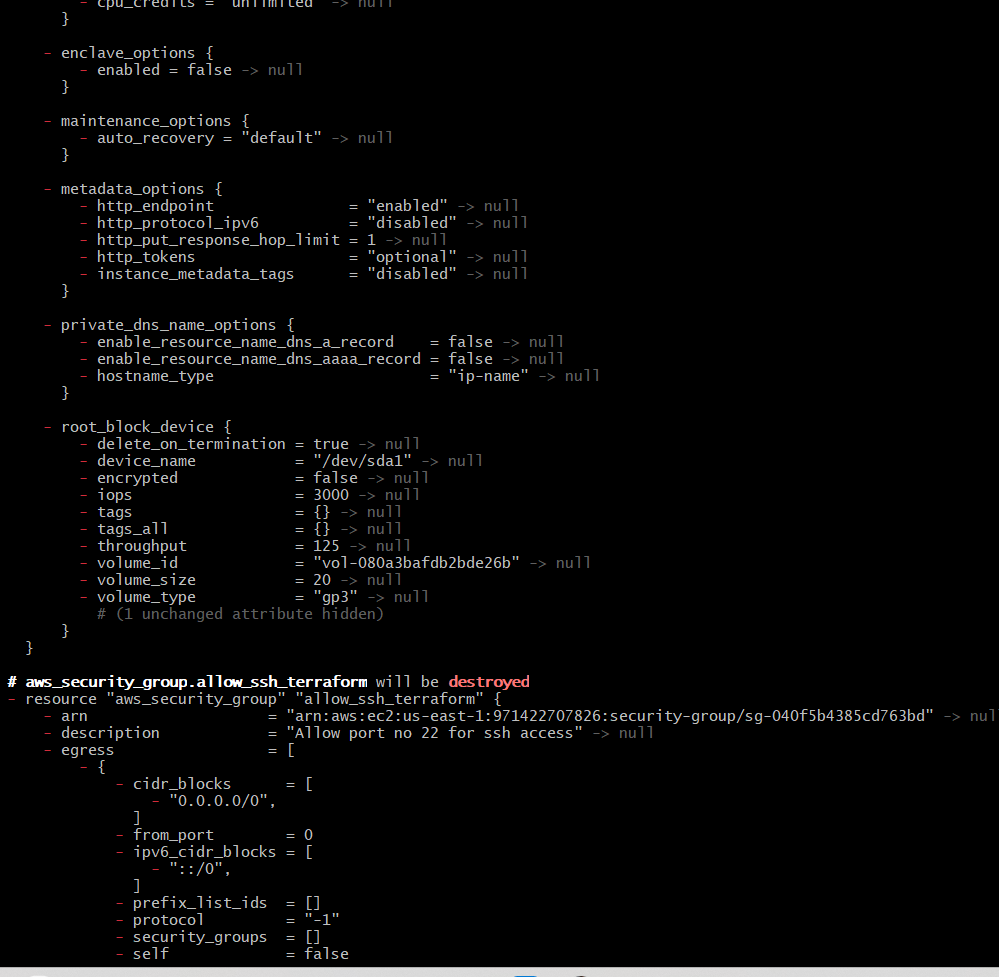
Security Group validation.

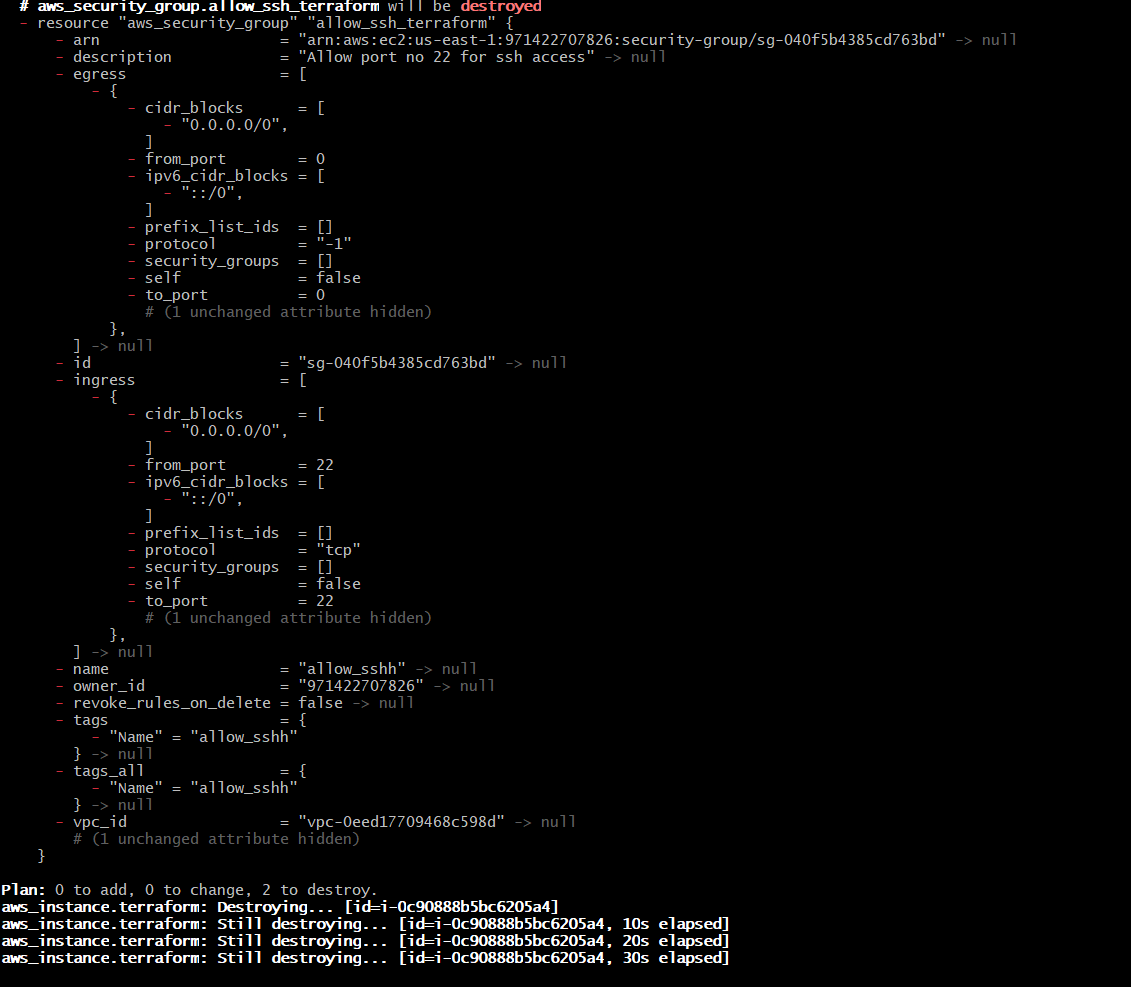


DESTROY

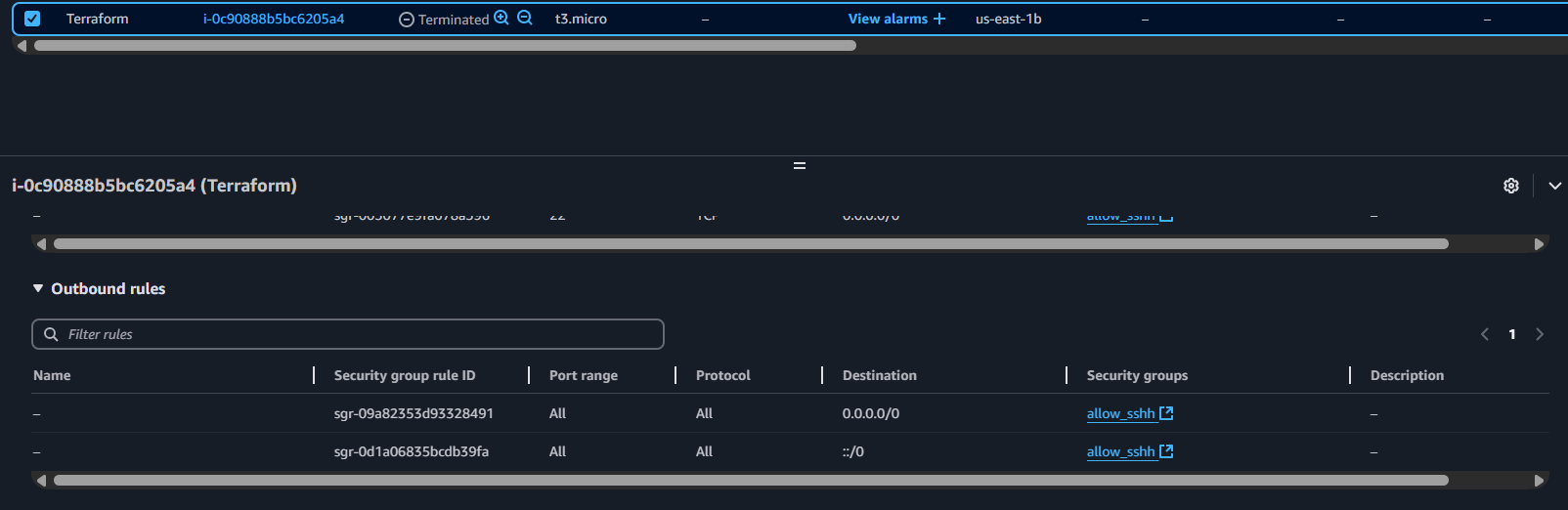
1. CMD : terraform destroy -auto-approve







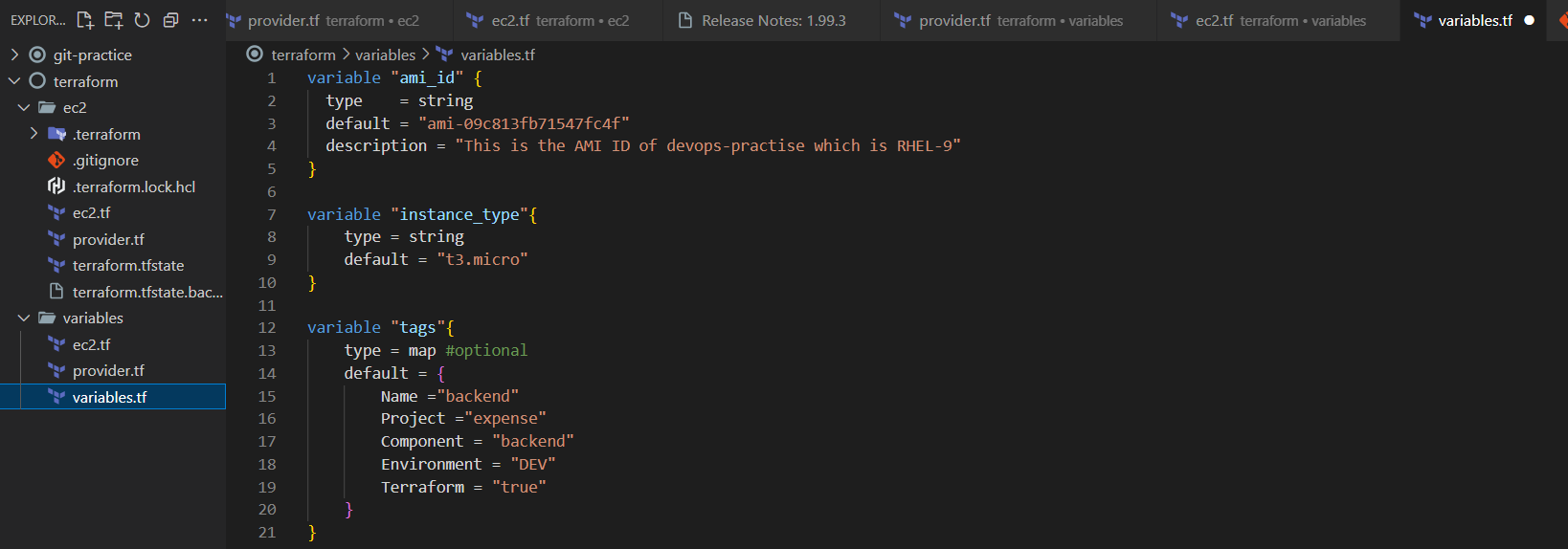
1. Validate the resource after destroy in aws.

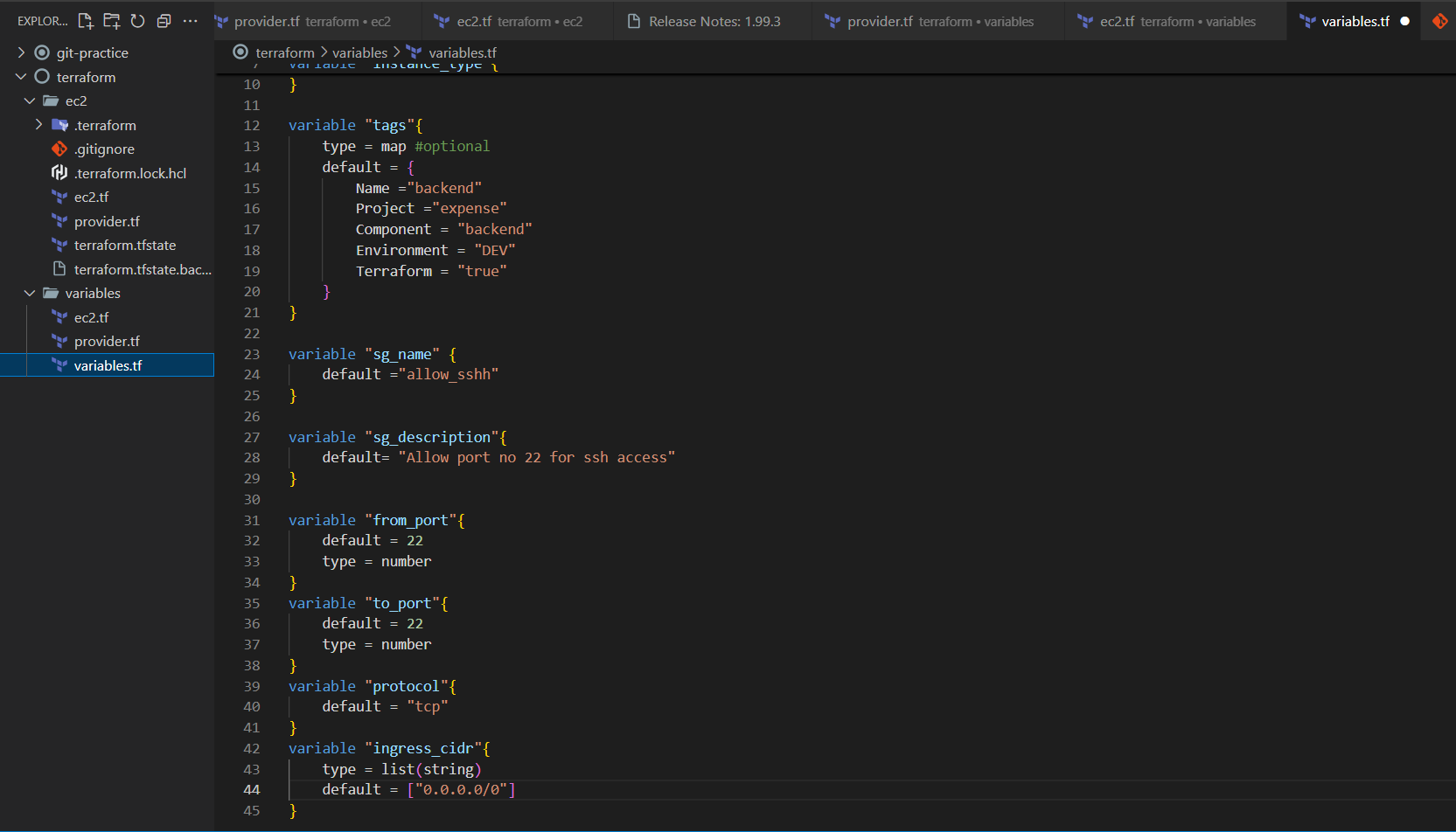


Note: EC2 is depedent on Security Group

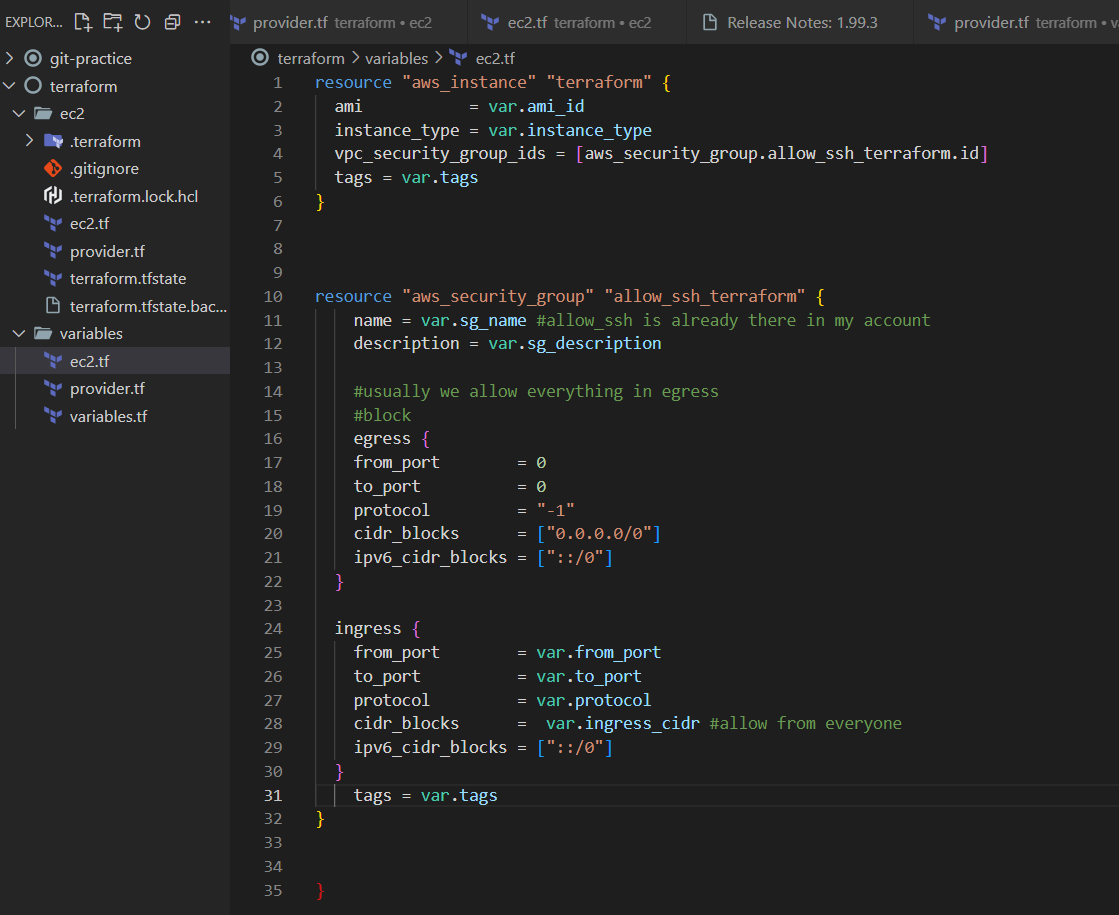
1. Variables
2. Conditions
3. Data Types
4. Loops
5. Functions
6. Variables

URL : <https://developer.hashicorp.com/terraform/language/values/variables>



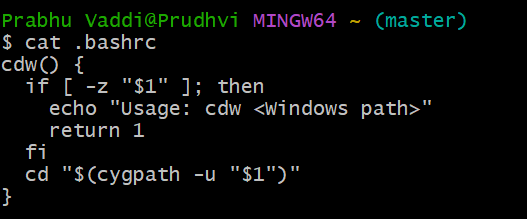


Change static value to dynamic value in ec2.tf.



**PUSH THE CODE INTO GIT**

1. Create a nano file with name .bashrc



Ctrl+X and Y then Enter

It helps to convert the path backward slash to forwardslash.

**CDW Function**

cdw "F:\devops-PrabhuTrainer\Git\repos\terraform"

PUSHING THE CODE from Local to GIT Repo

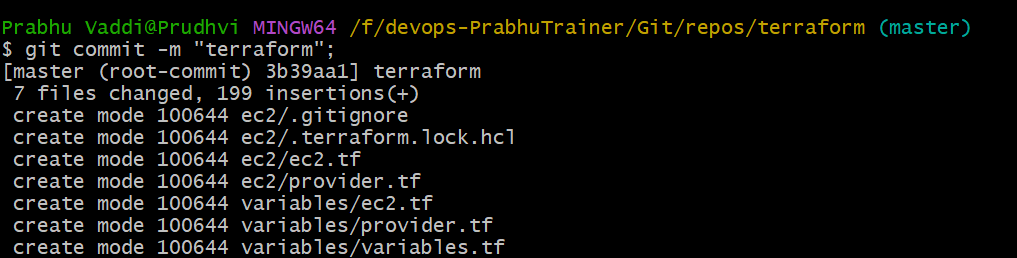
1. Git init
2. ls -ltra



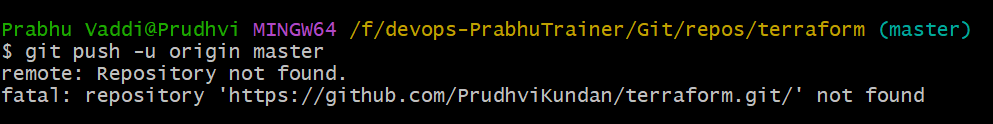
1. git status , git add .



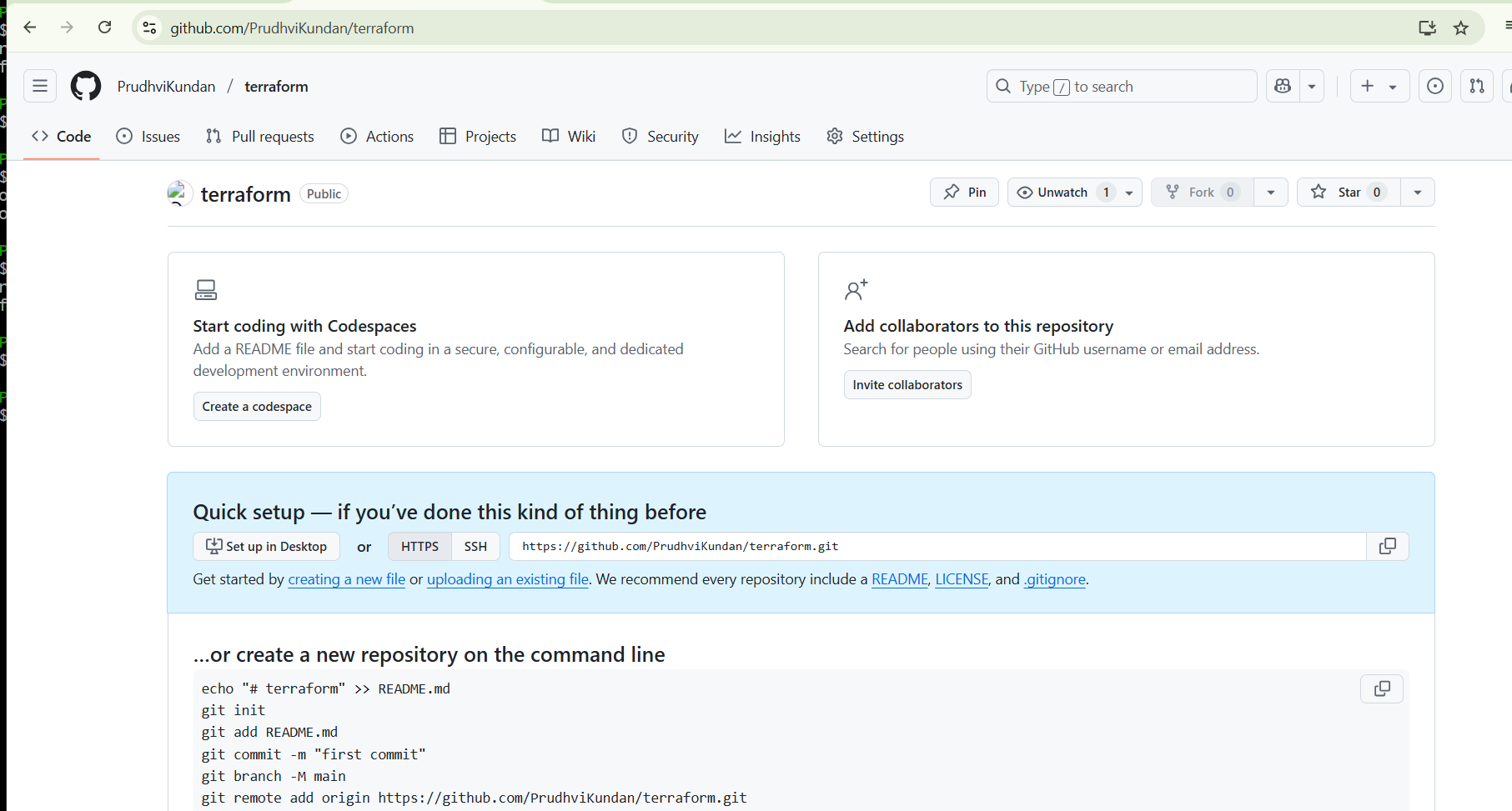
1. git commit



1. git push => NO Repository exist in GITHUB



1. Create a repository in GIT HUB.

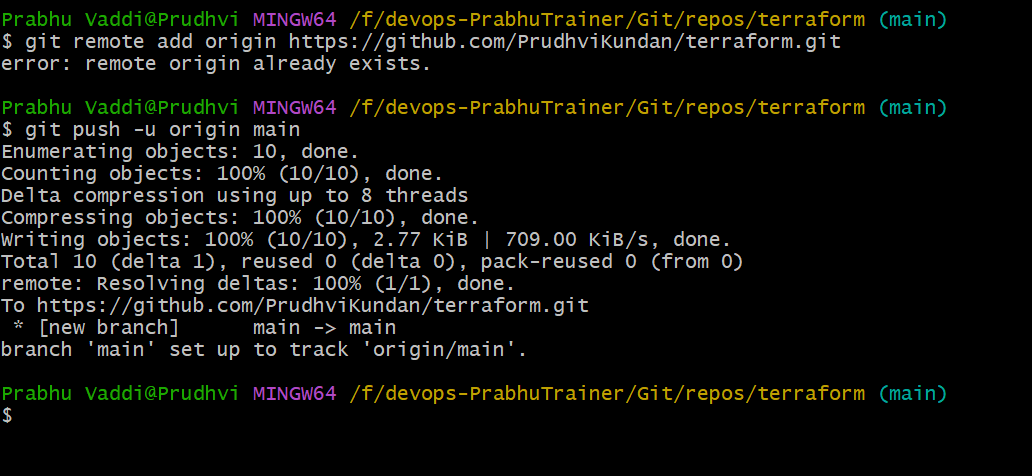


1. Follow this steps in GIT BASH

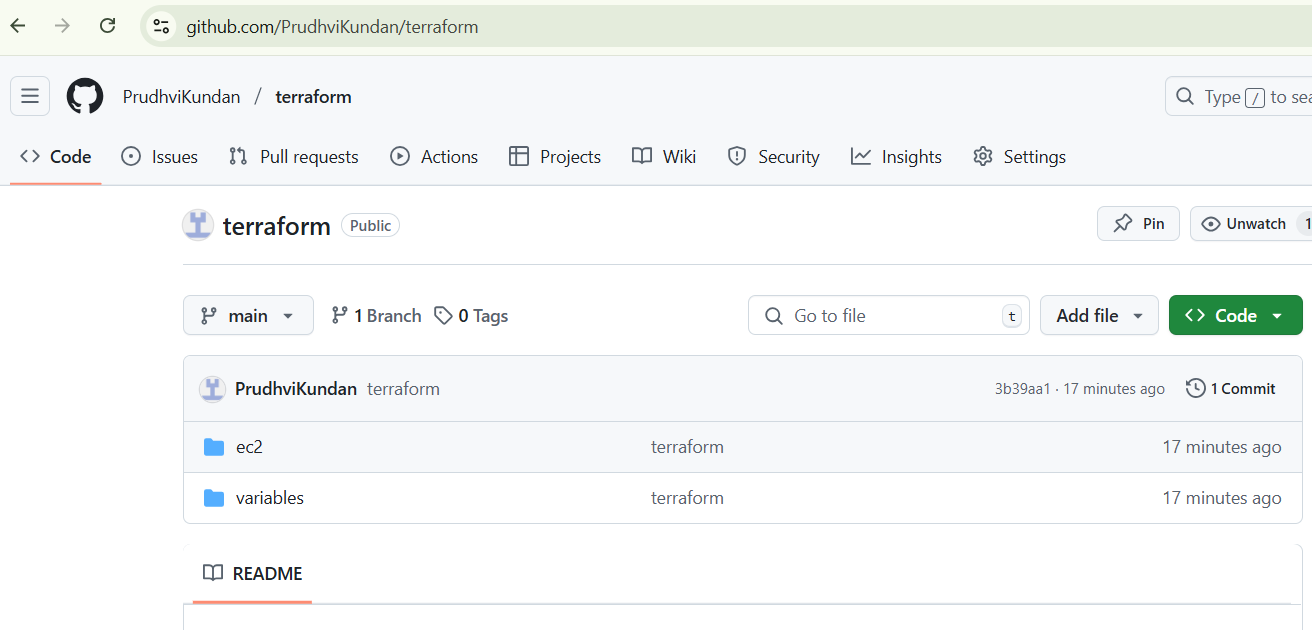
git branch -M main

git remote add origin https://github.com/PrudhviKundan/terraform.git

git push -u origin main

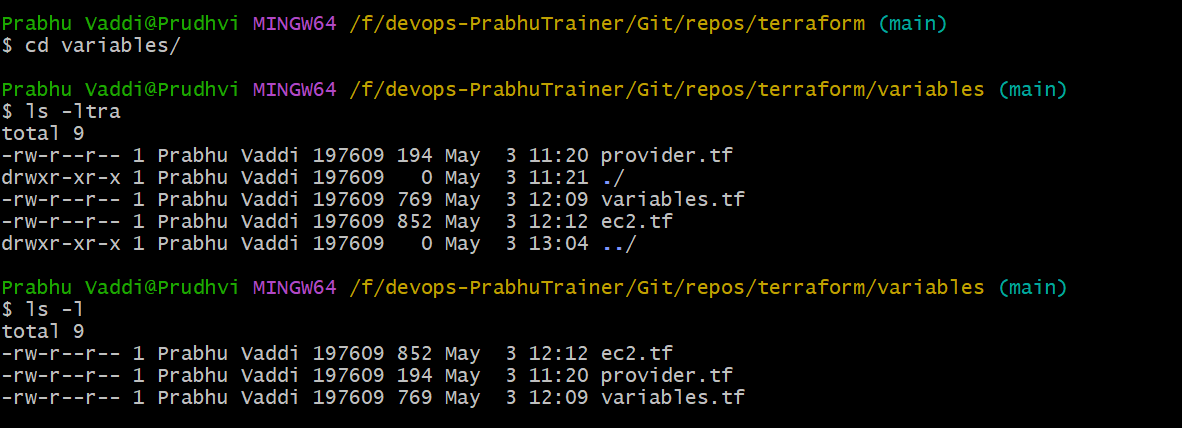


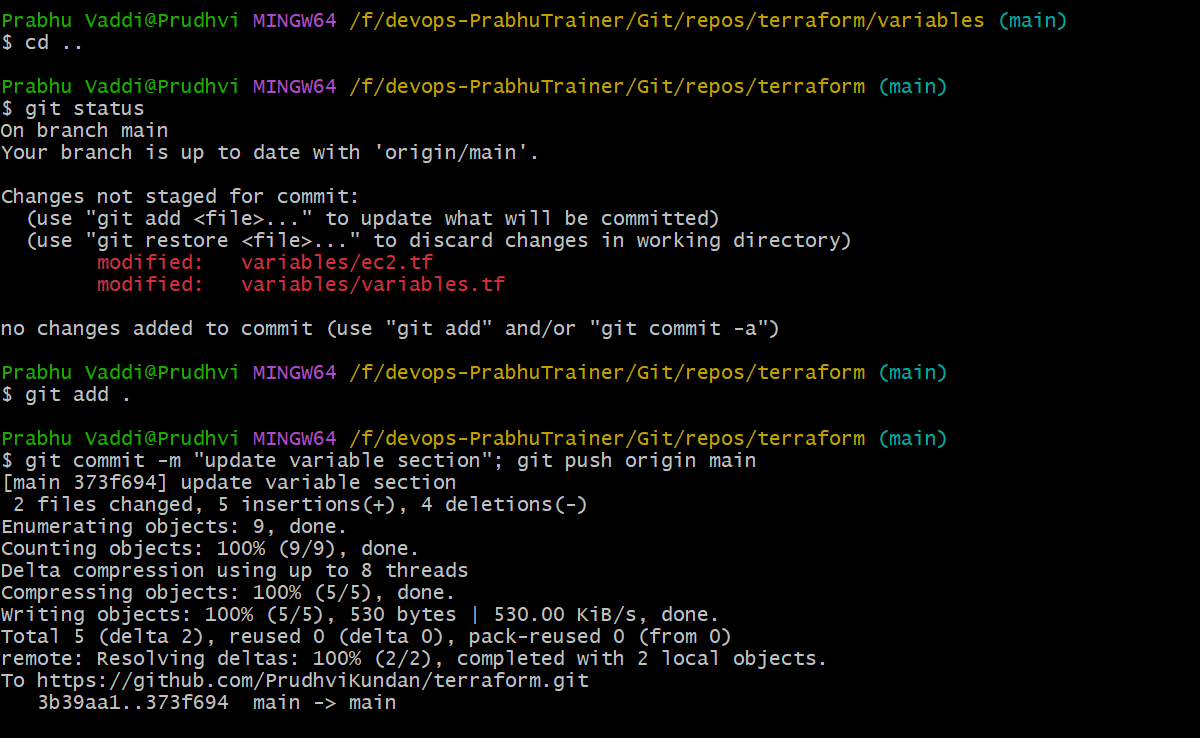
**Validate GIT Repo - Terraform**



**Variables**

In GIT BASH=> Terraform initialization for variables section



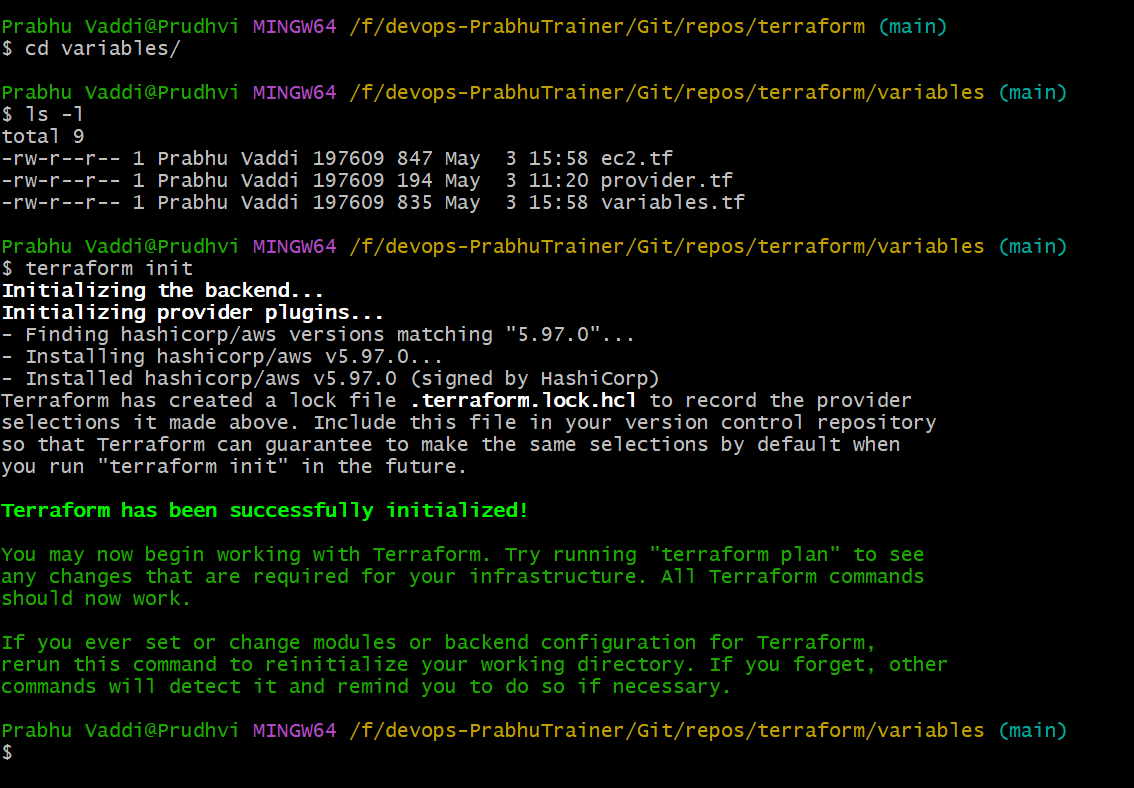


**Terraform Plan for variable section**

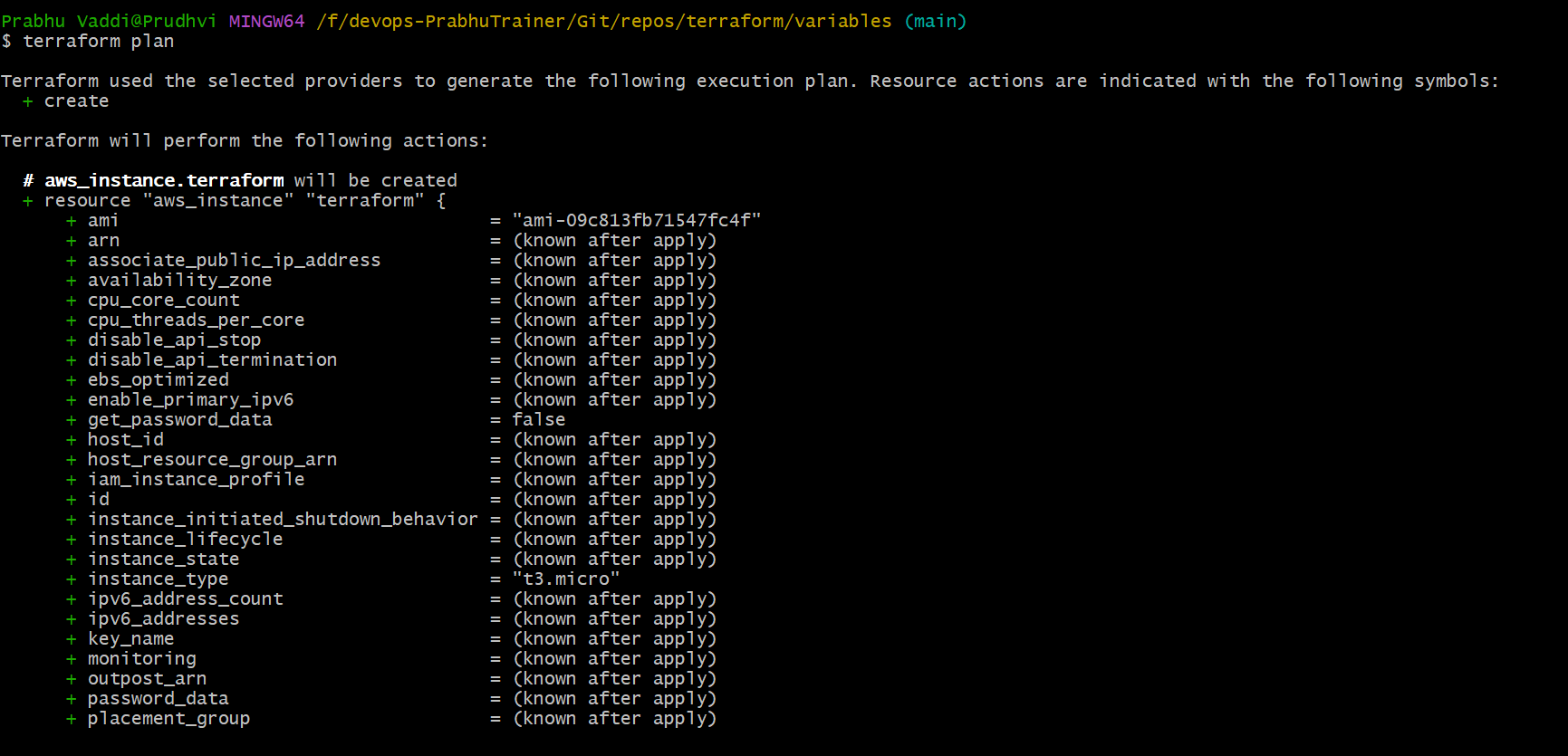
Once, code deployed into git. Next, Changes are happened to different directory (variables).

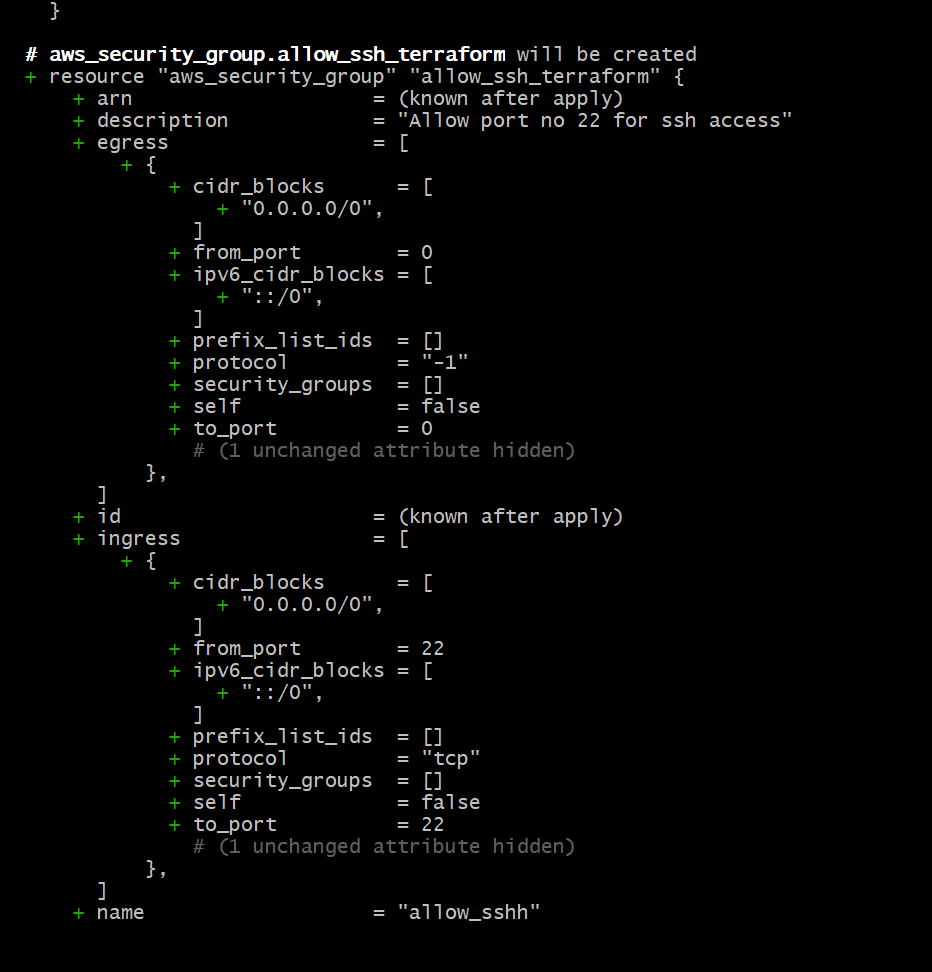
Start the terraform commands.

1. First time – terraform init



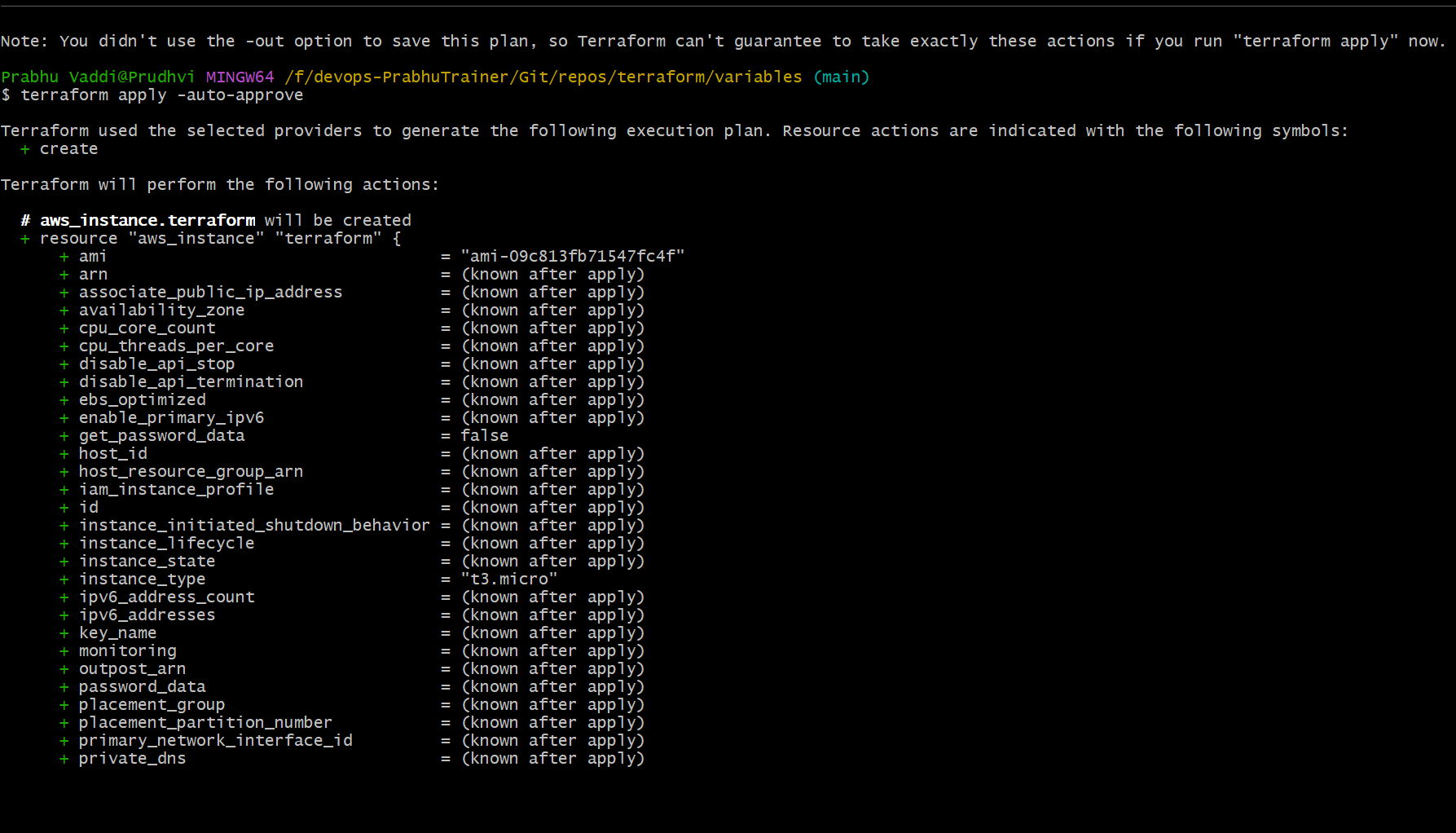
Terraform Plan => Just staging

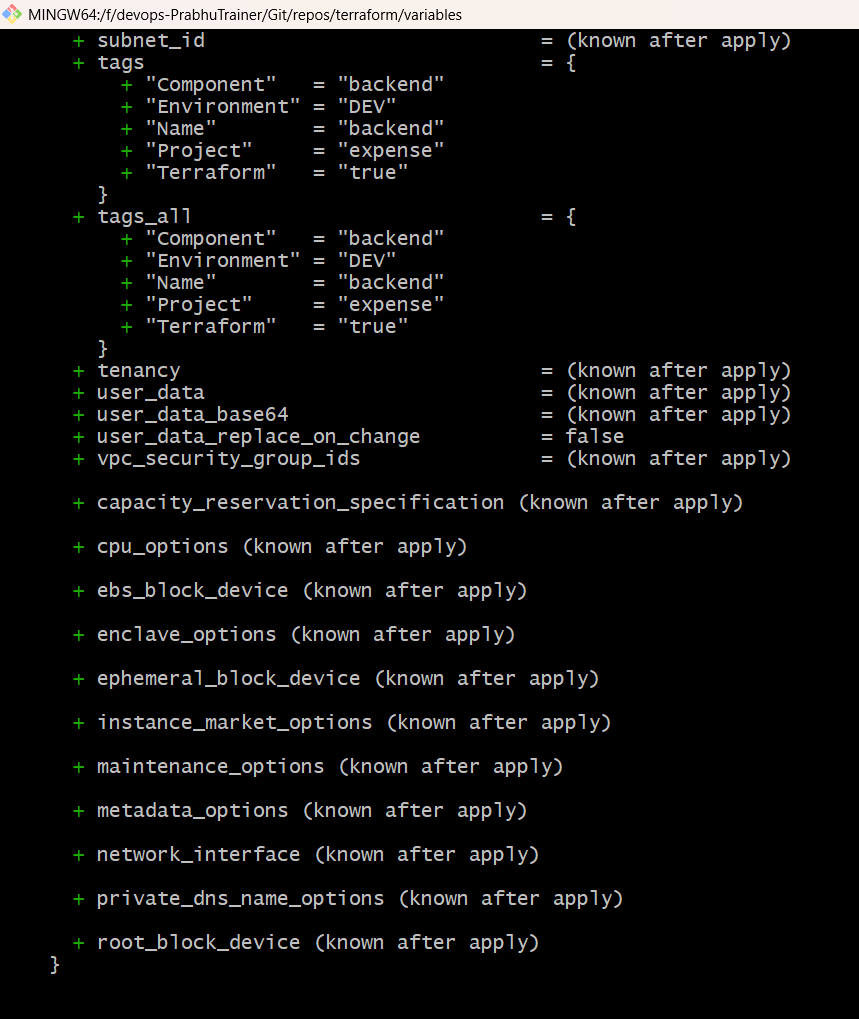


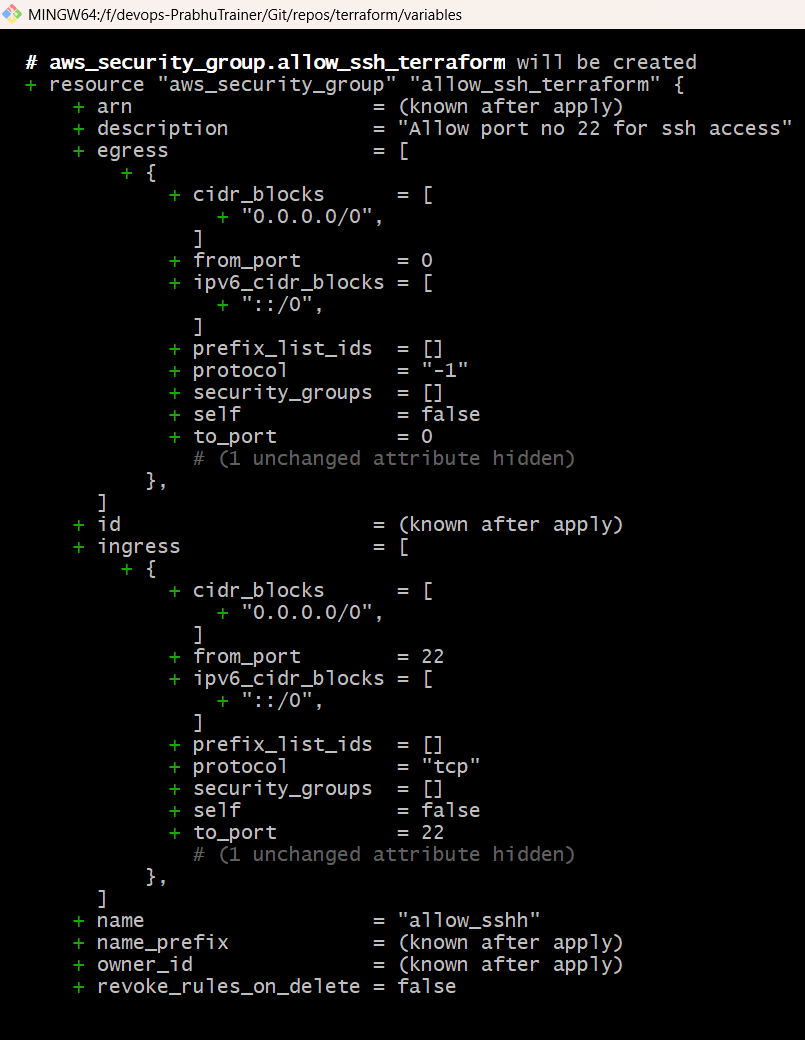


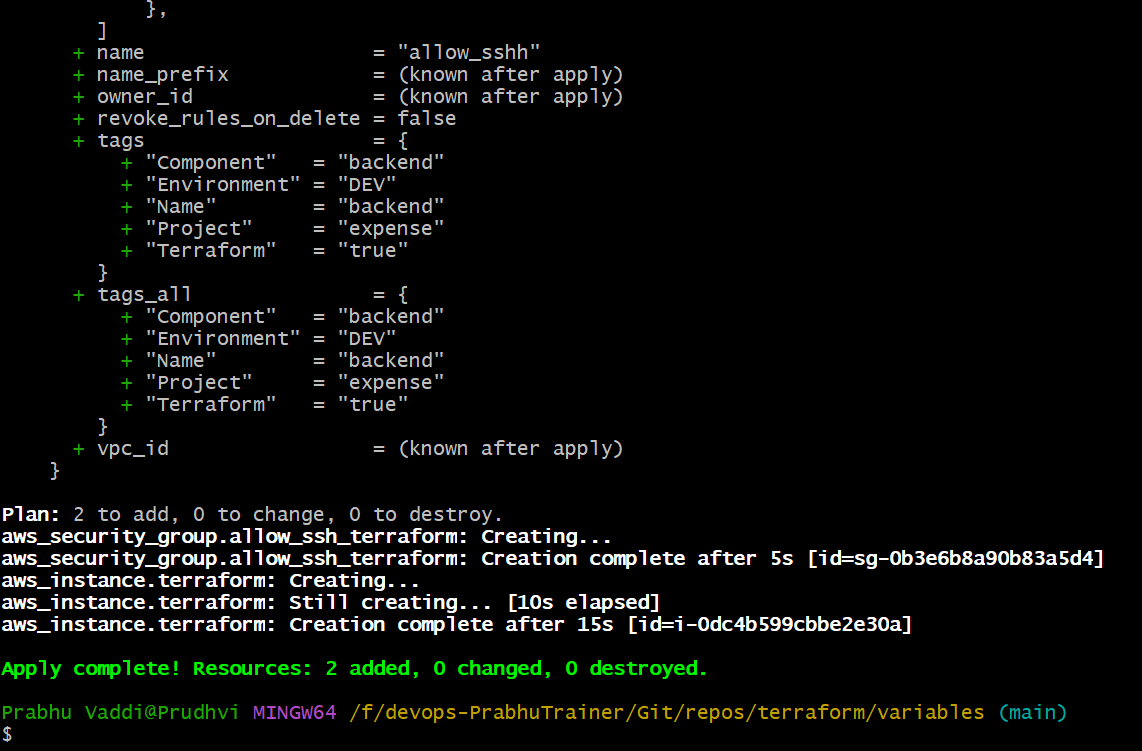


1. Terraform apply -auto-approve

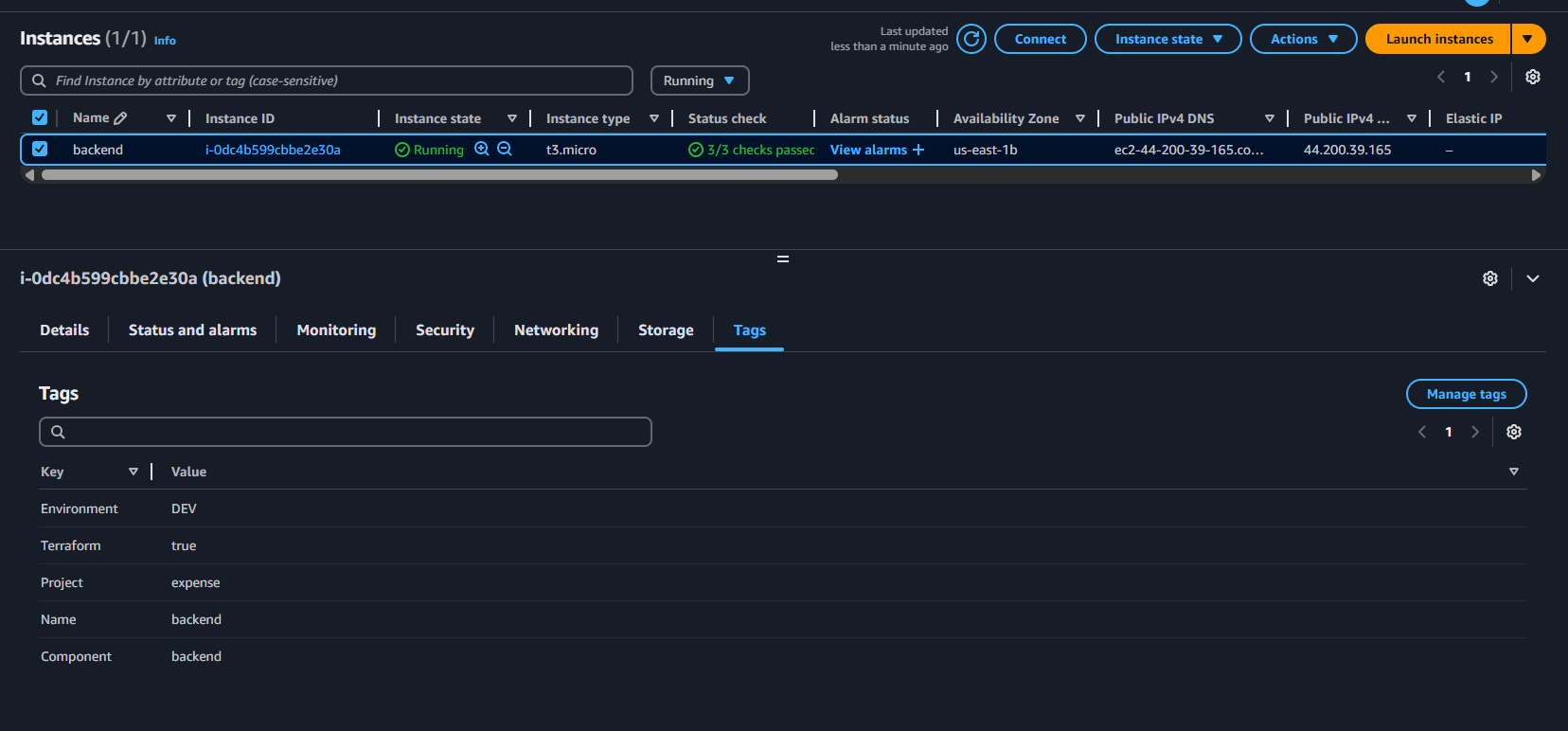




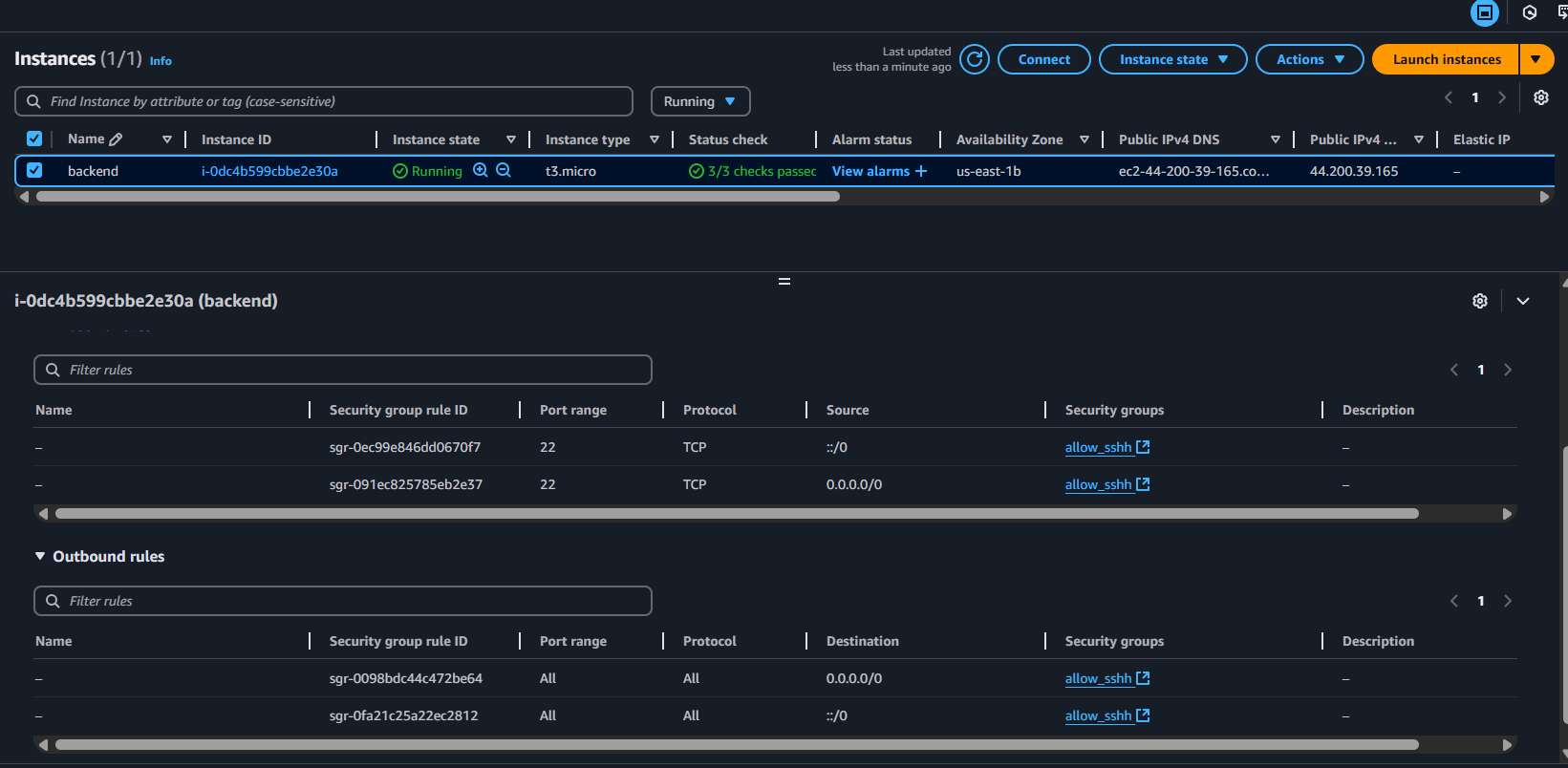




1. Validate in AWS Console



**Security**

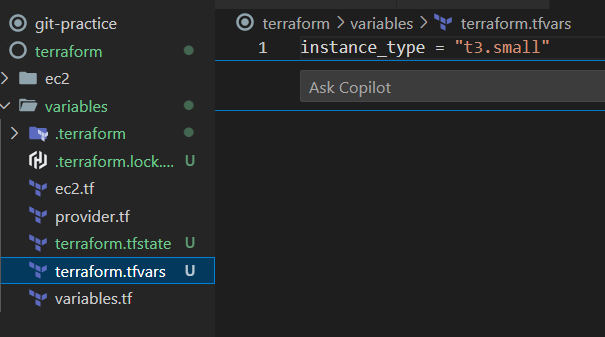
****

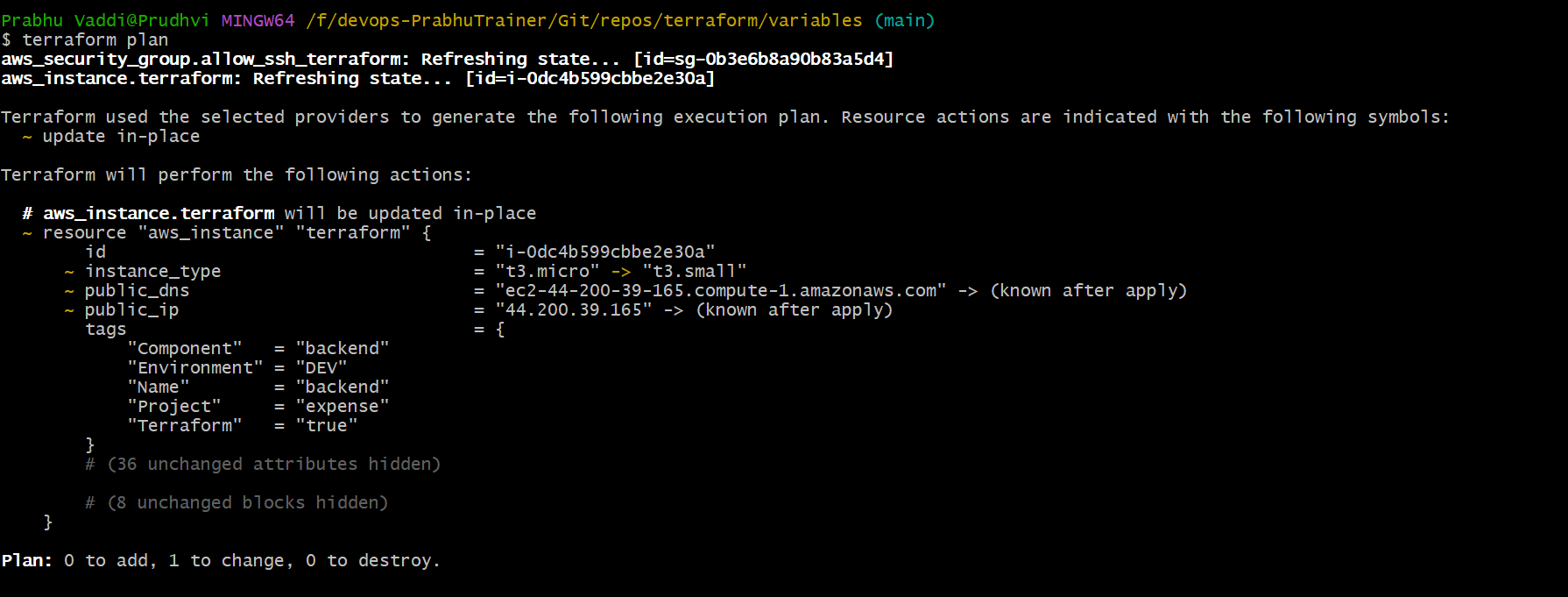
**Terraform.tfvars –** using this file, we can override the default values in variables or else we can set the values

First Preference – tfvars

Second preference – default

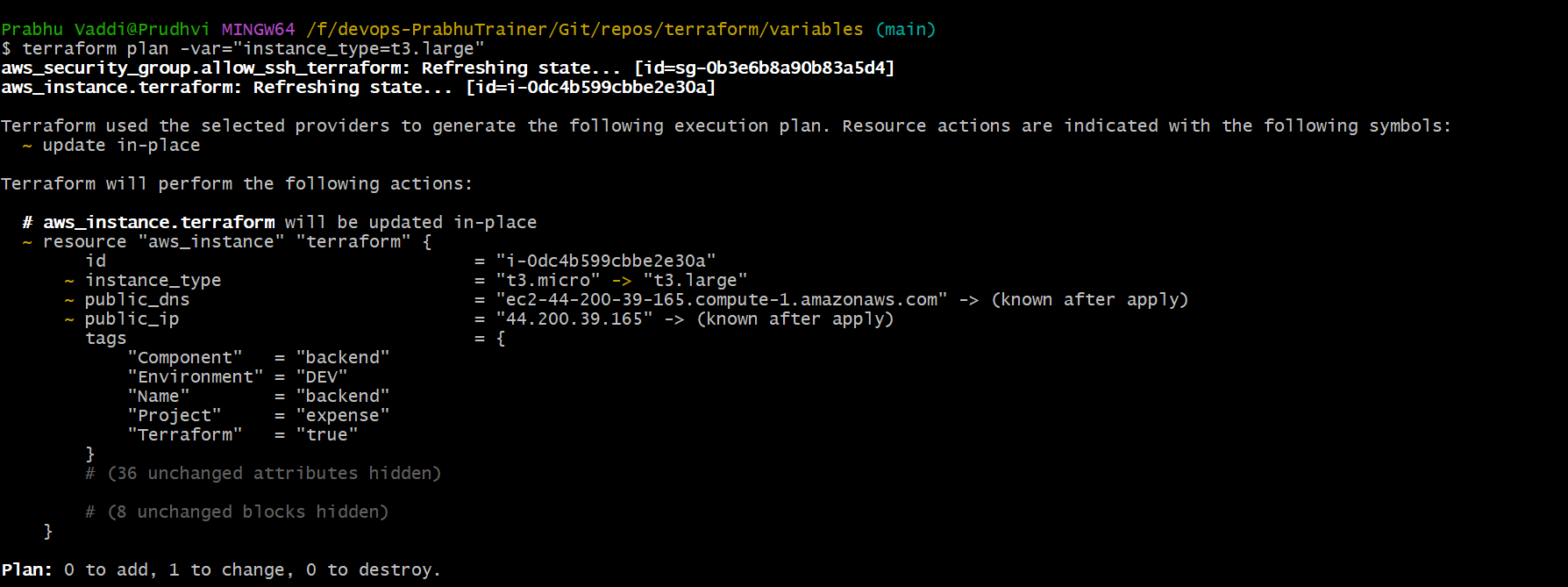
1. Commandline – 1st Preference
2. TFVARS – 2nd Preference
3. Environmental variables – 3rd Preference
4. Default – 4th Preference





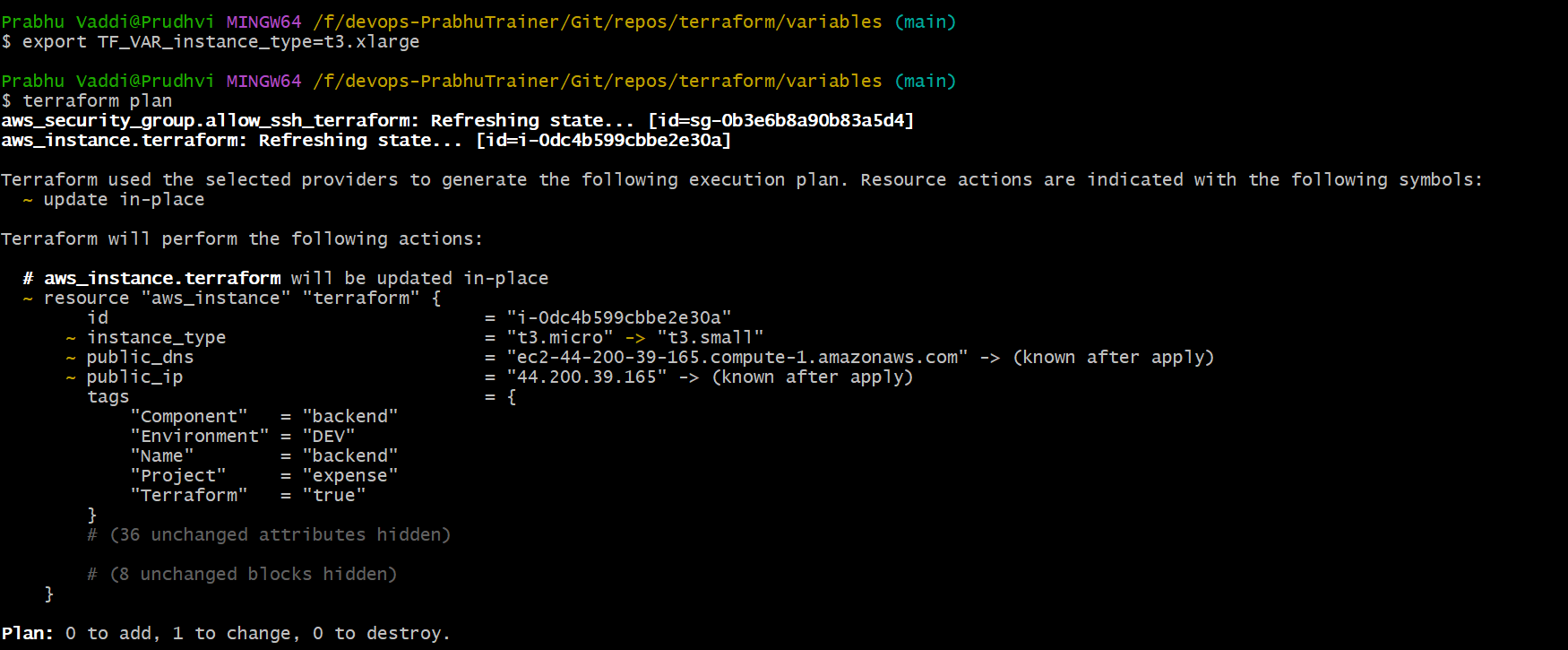
**Command Line**

**CMD : terraform plan -var="instance\_type=t3.large"**

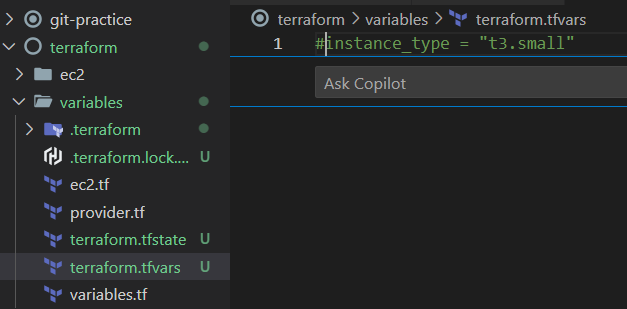
****

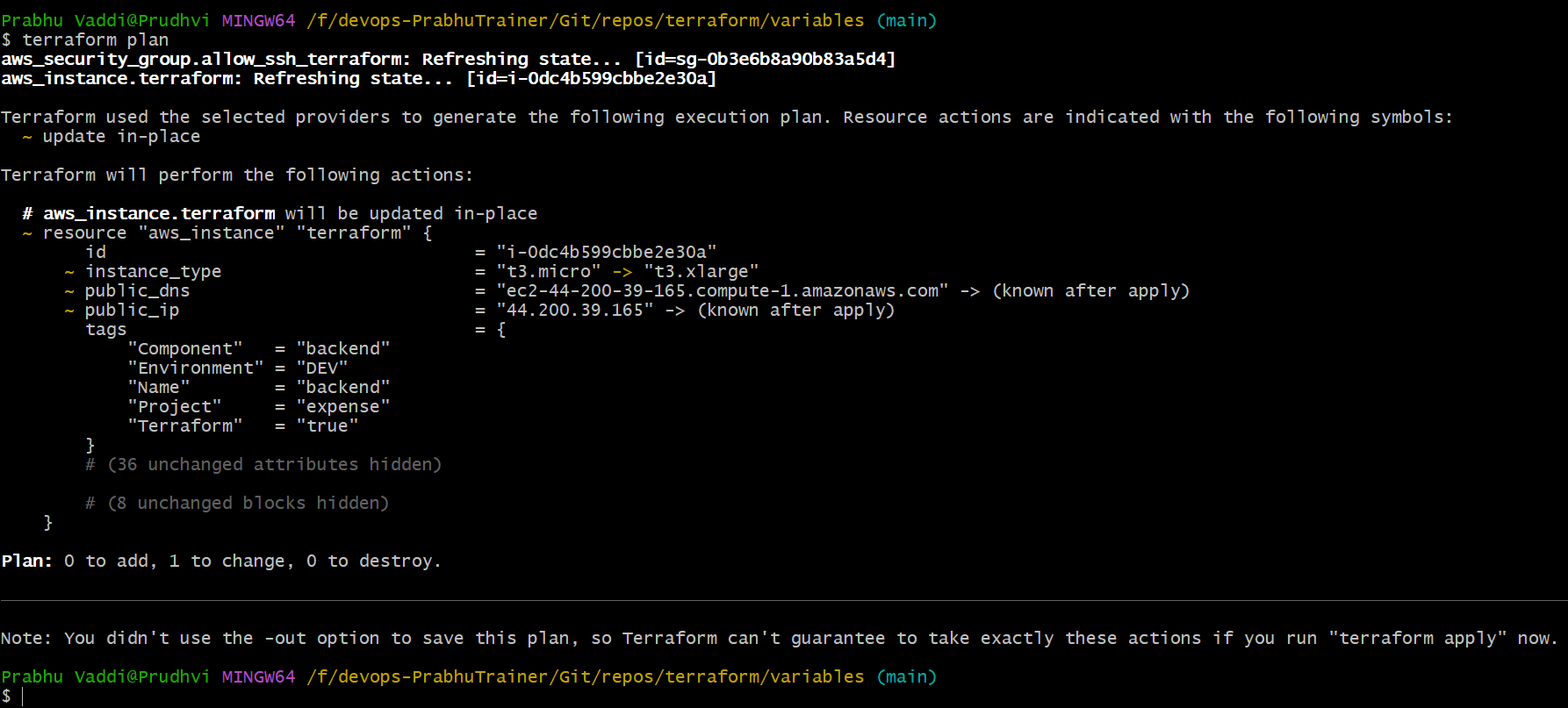
**Environmental variable**

**CMD :** export TF\_VAR\_instance\_type =t3.xlarge

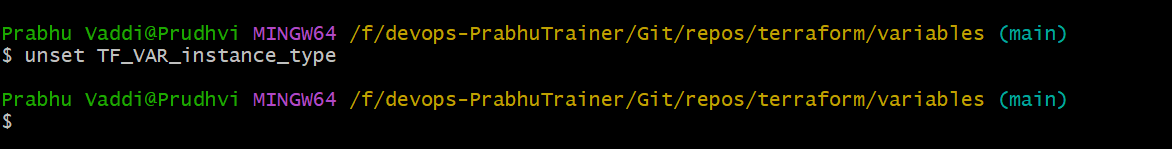
****

1. **DEFAULT : committed tfvars in code**

****

****

**Comment Instance\_Type & Unset Environment variable**

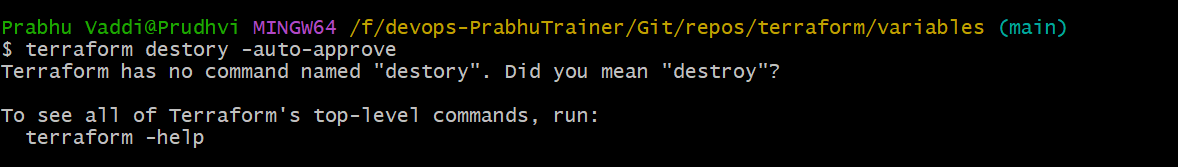
****

**It is checking all places , but nothing found. So, it is asking enter the value**

**des**

**DESTORY**

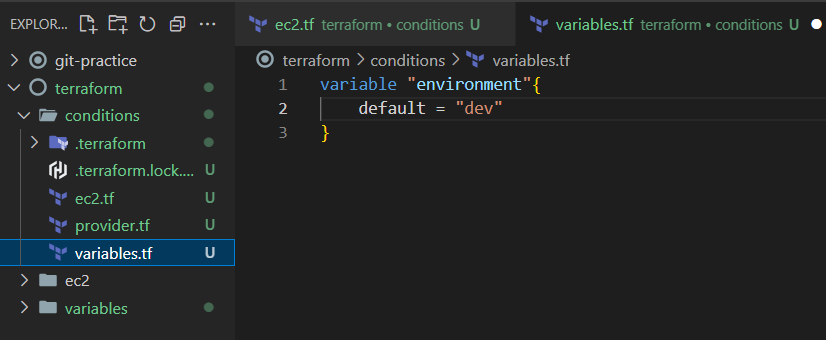
1. Terraform destroy -auto-approve

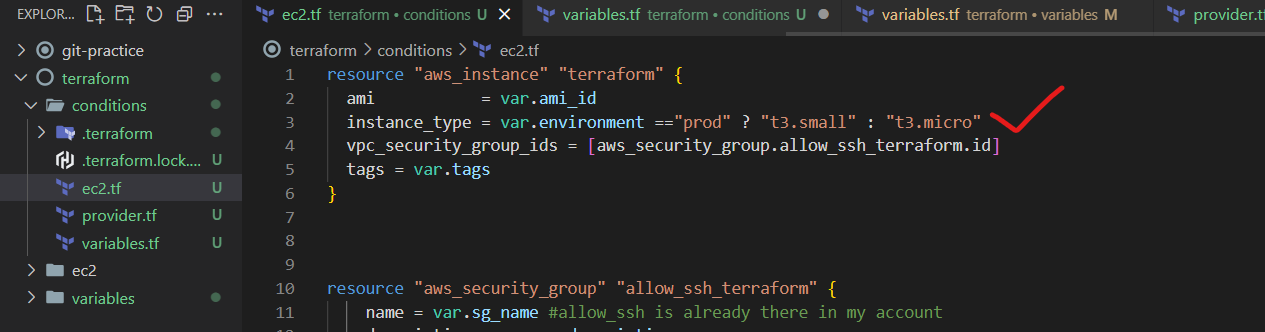
****

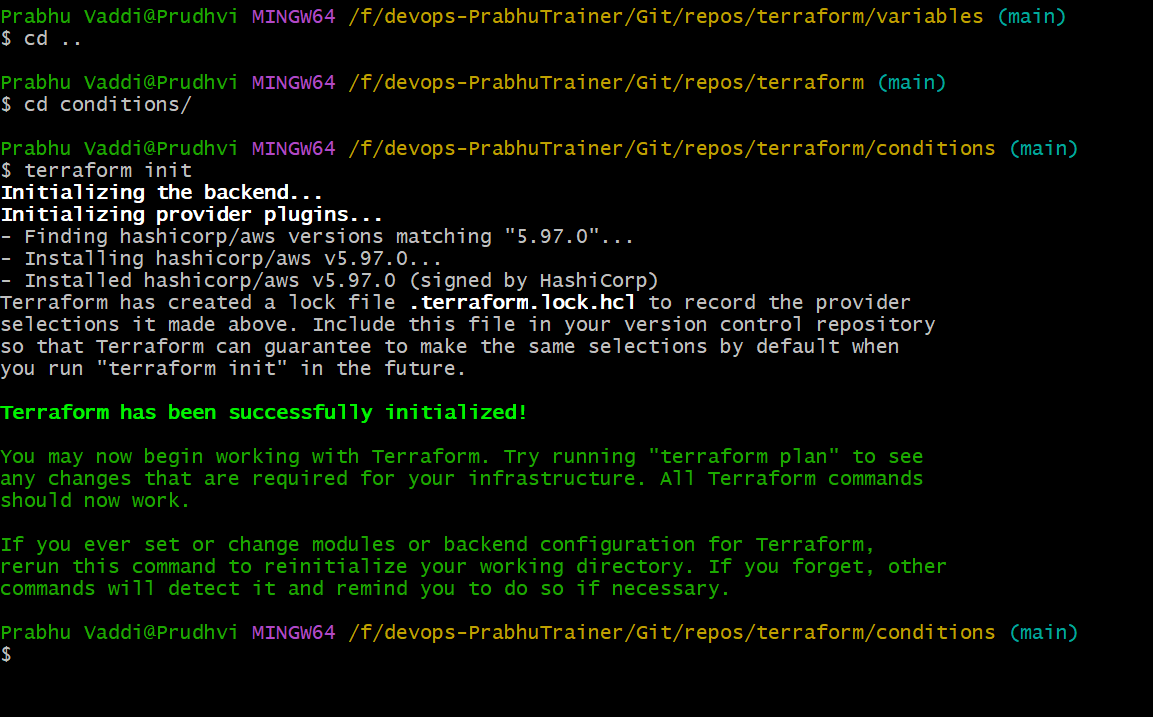
**Validate EC2 instance in AWS. Nothing is running after destroy**

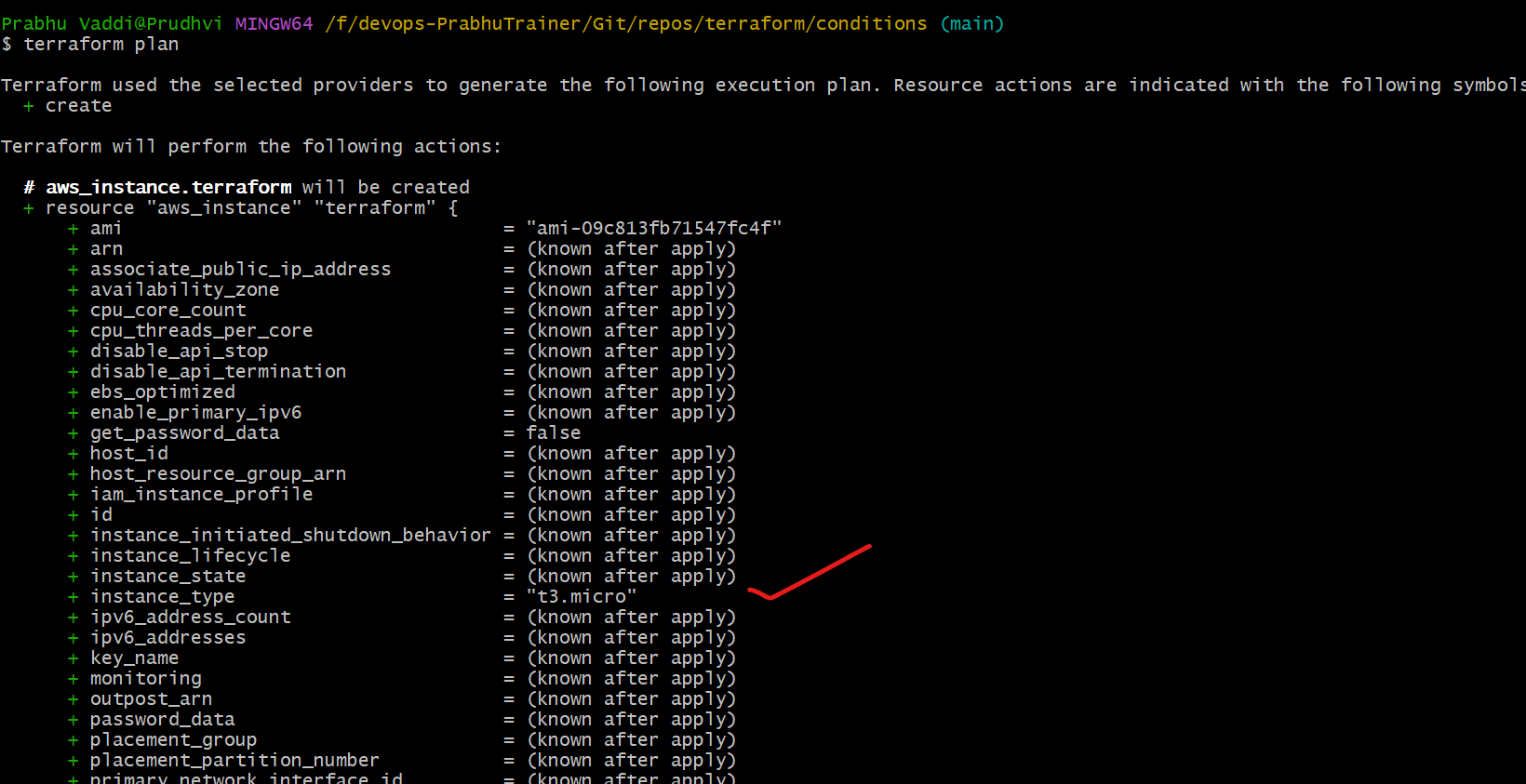
****

CONDITIONS

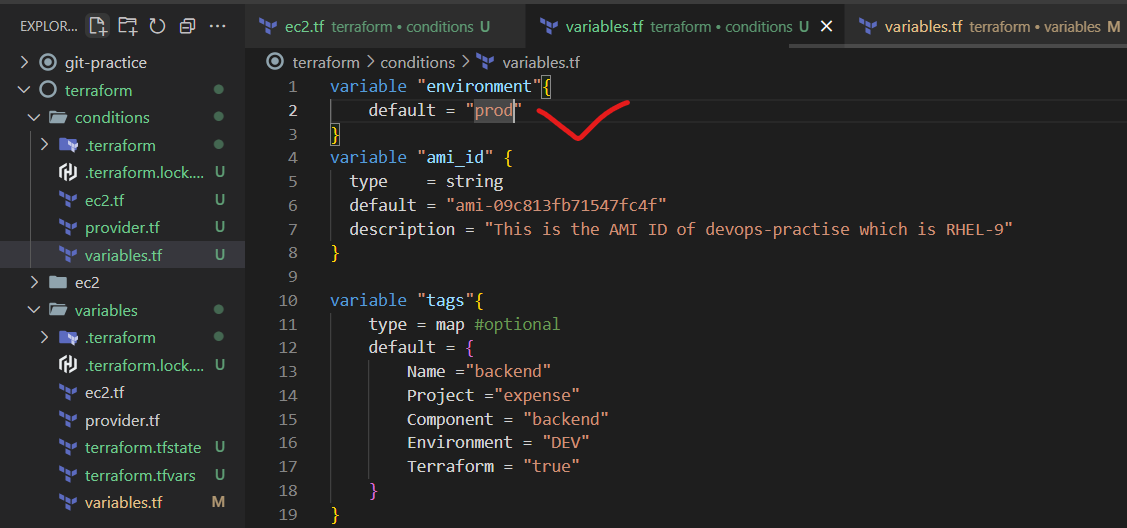


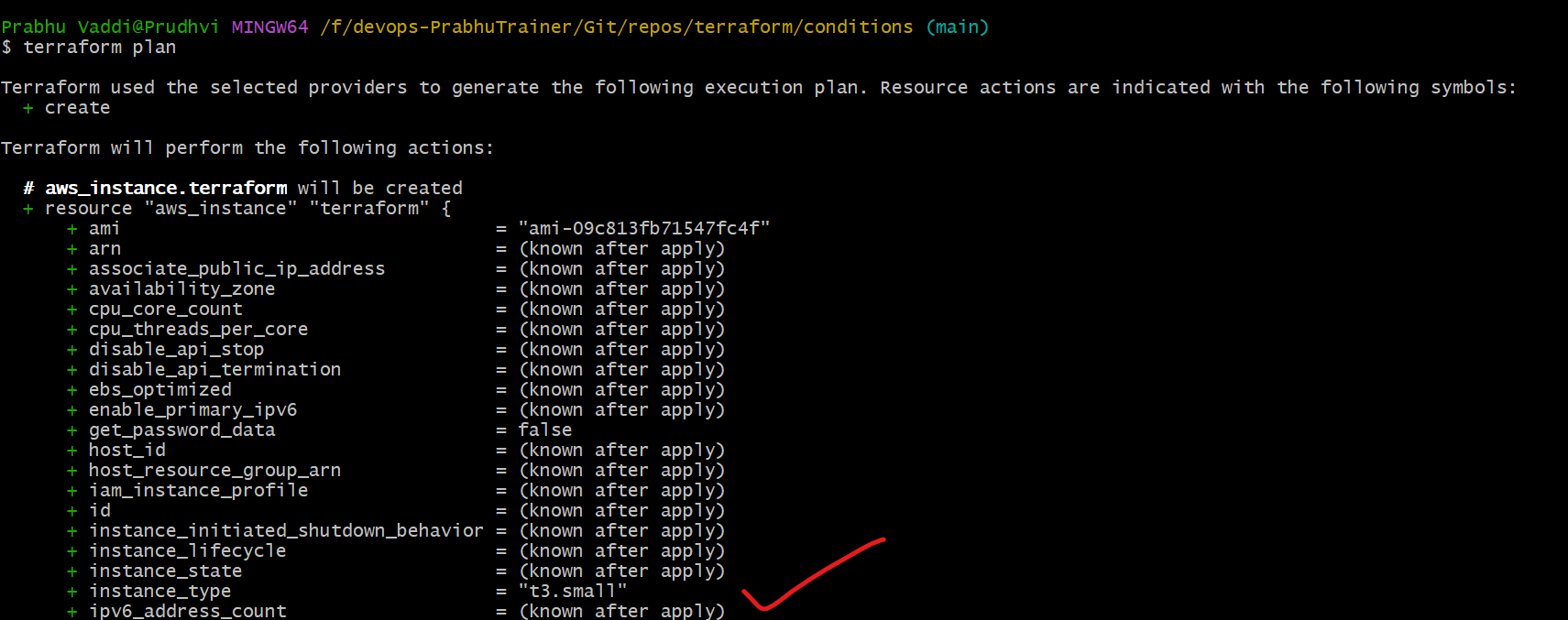






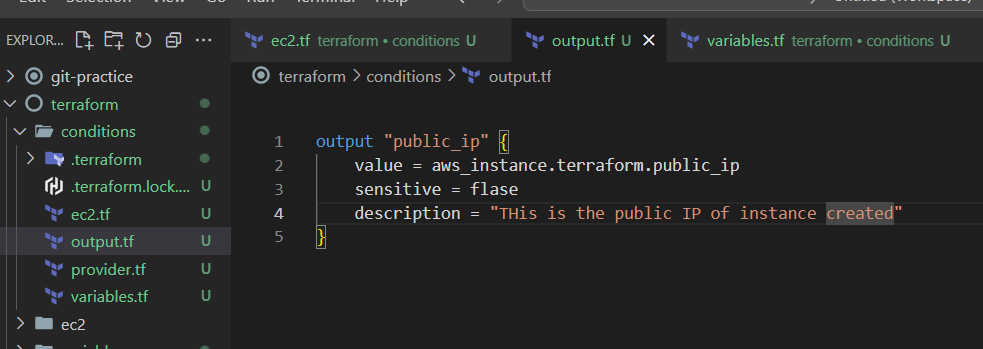
**ENV – PROD : t3.small**

****

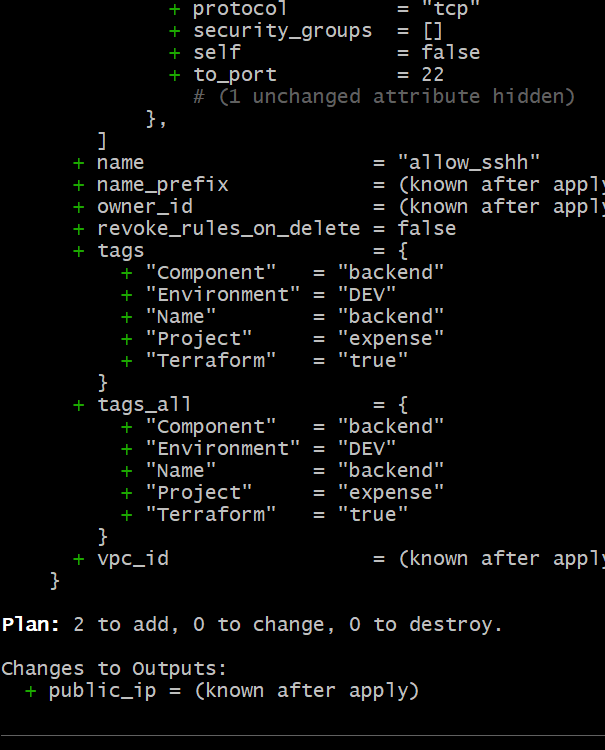
****

**OUTPUTS**

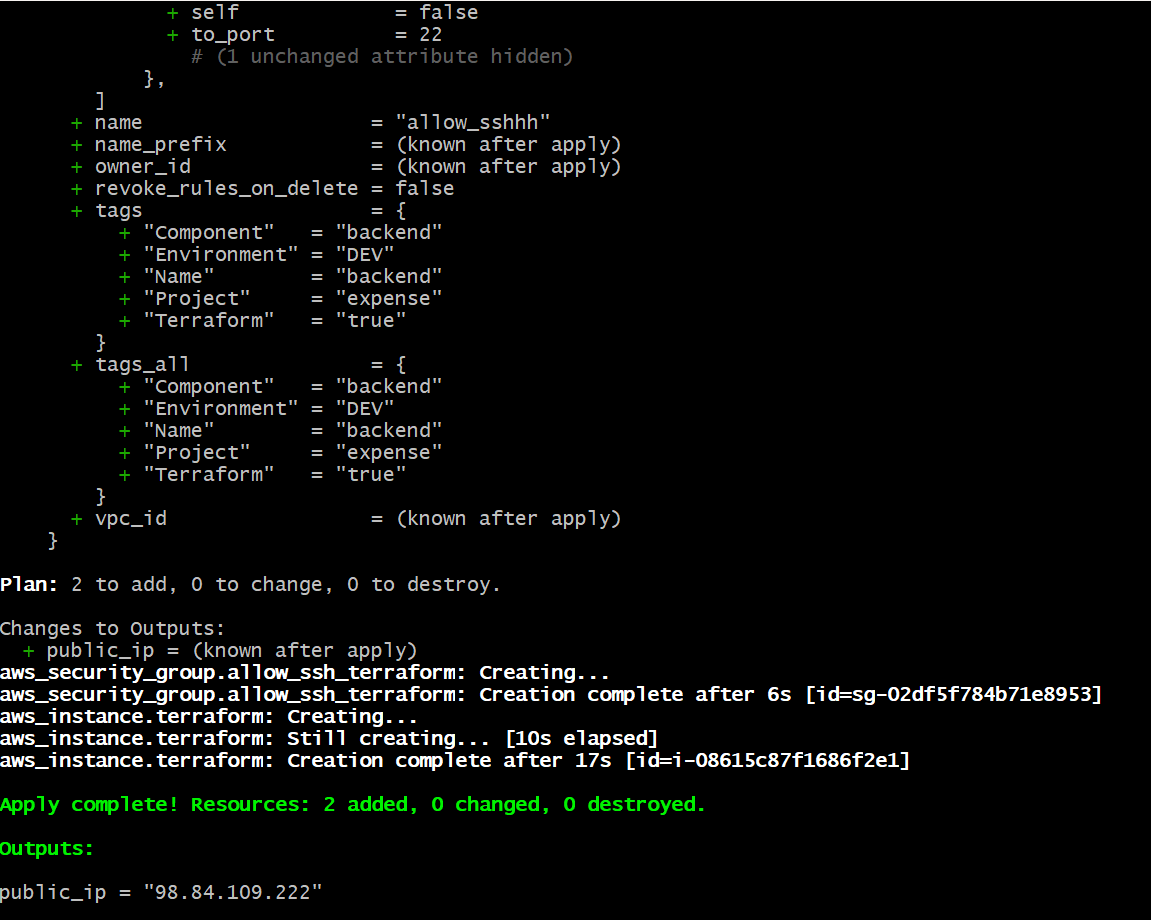
**We can use terraform output things, whereever u want**

****

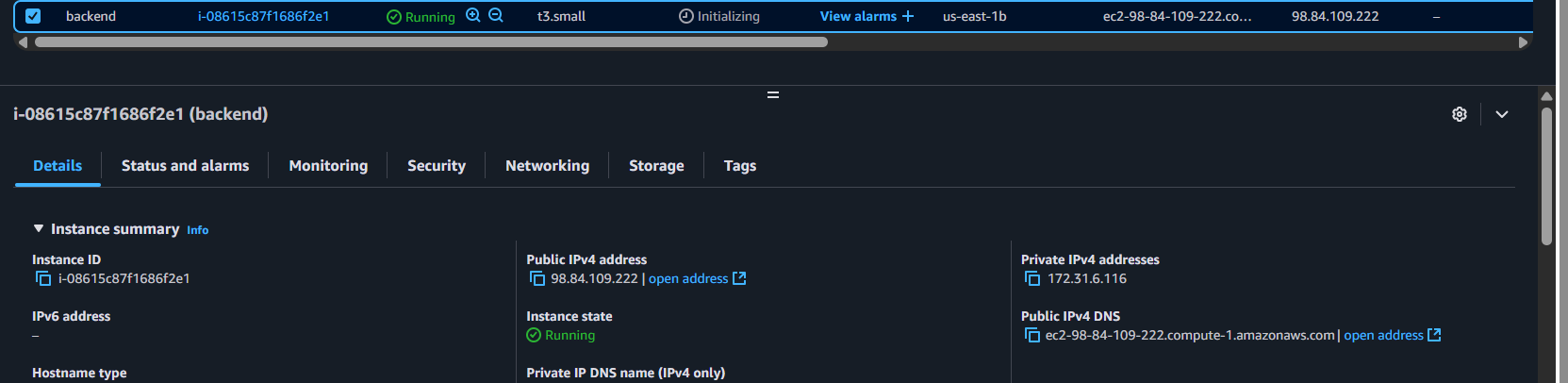
**Terraform plan**

****

**Terraform apply**

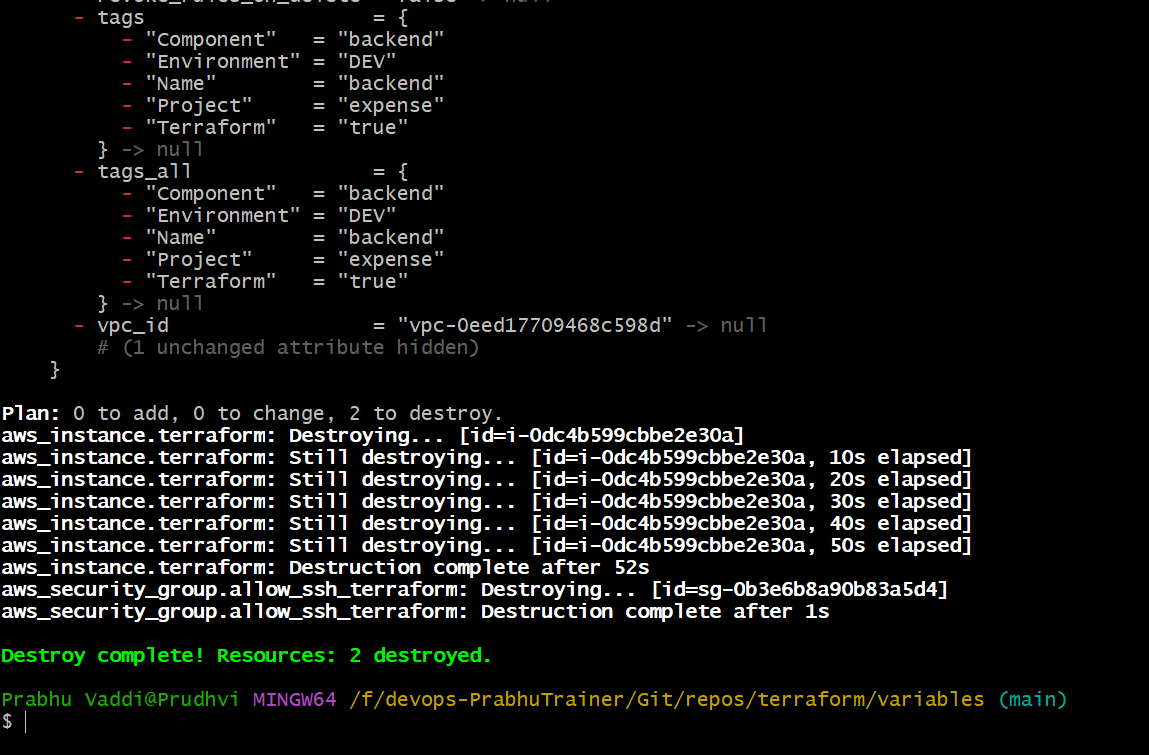
****

**Validate IP Public both are same**

****

**DESTROY Resources**

**terraform destroy -auto-approve**

****

**LOOPS**

**1. Count based loops**

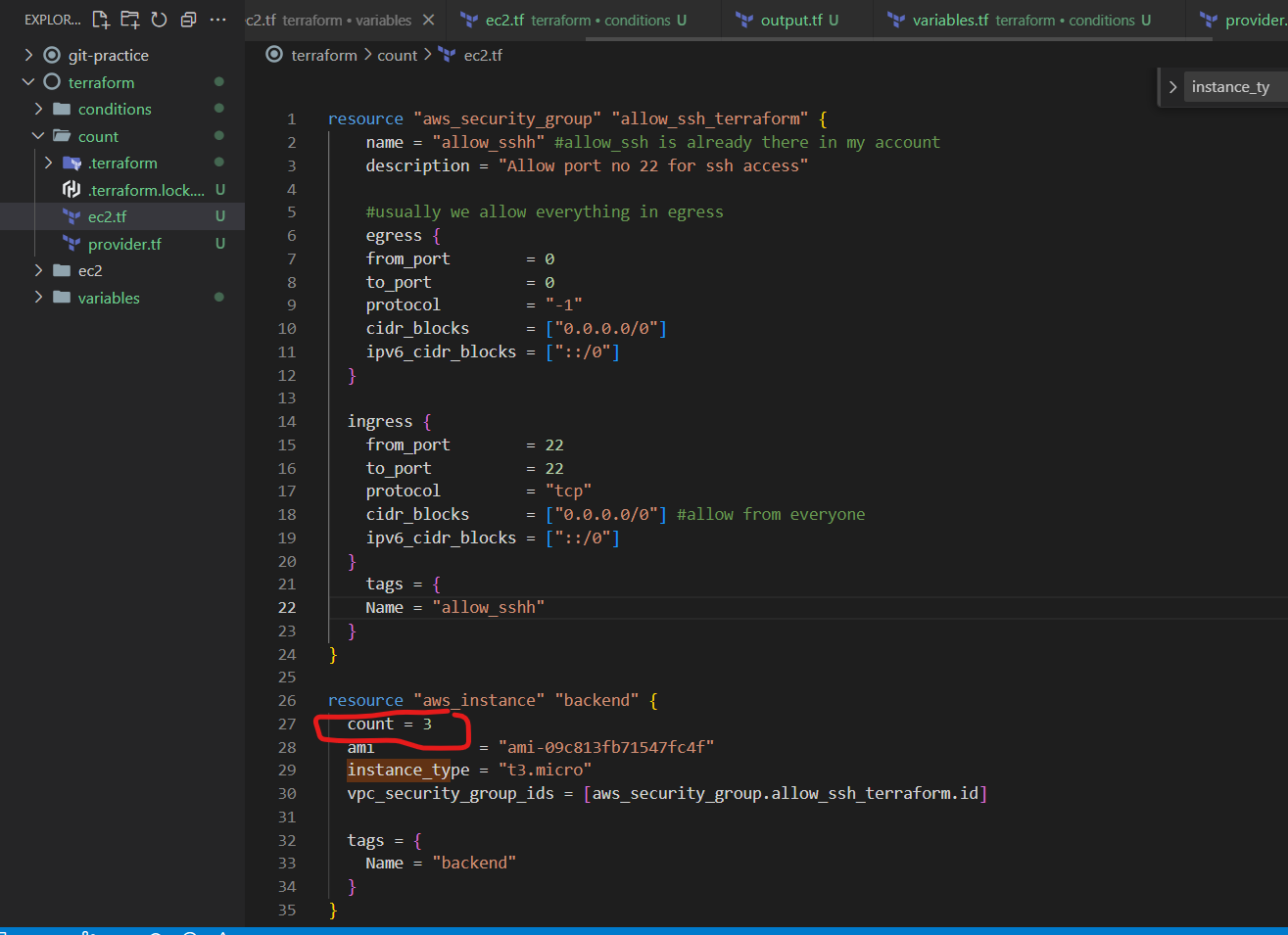
**2. for or for-each loops**

**Folder name – count**

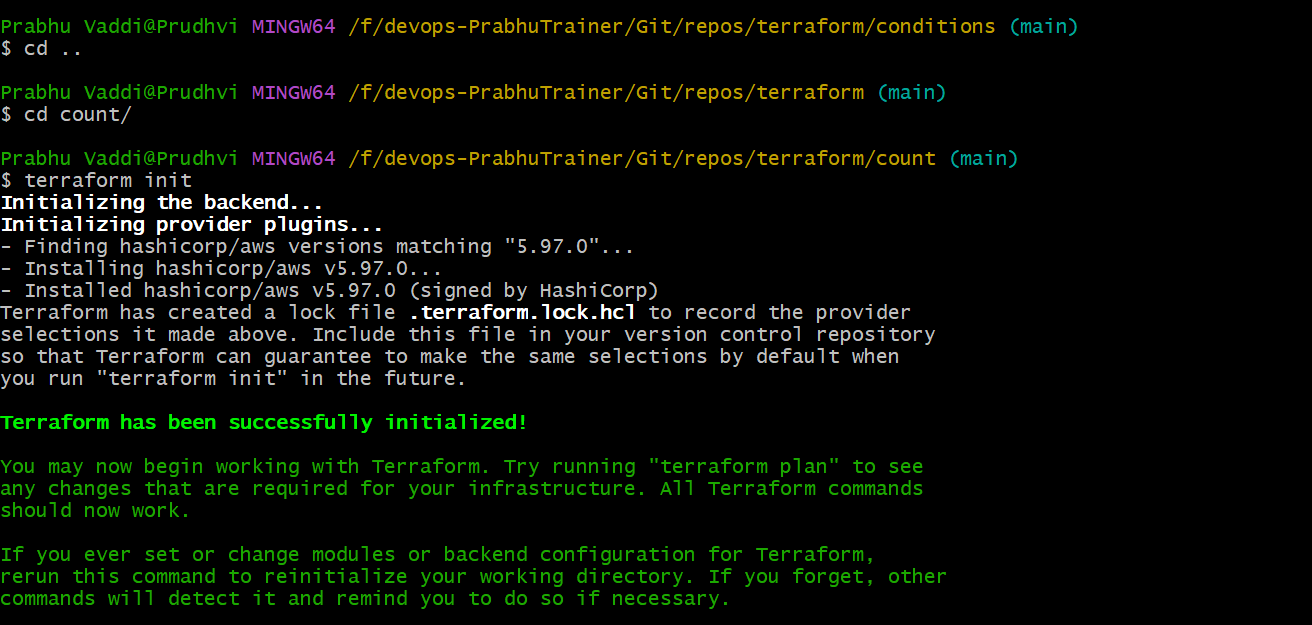
**Count = 3 instances**

**Tag name = backend**

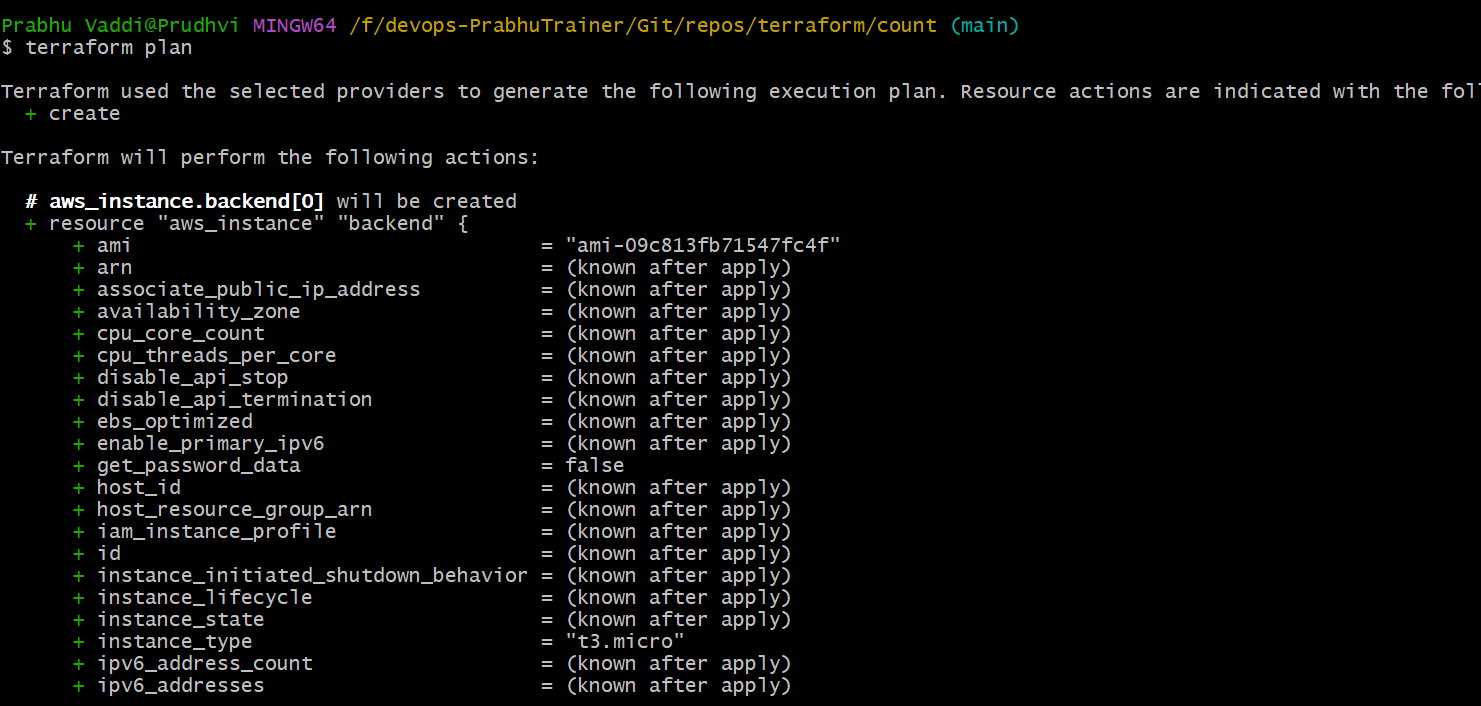
**Resource name = backend in ec2.tf**

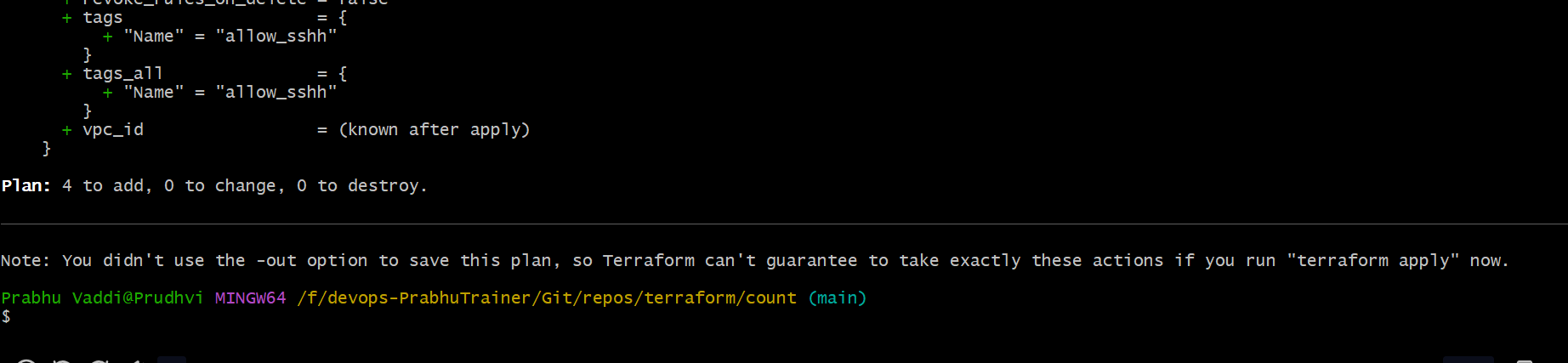
****

Terraform init

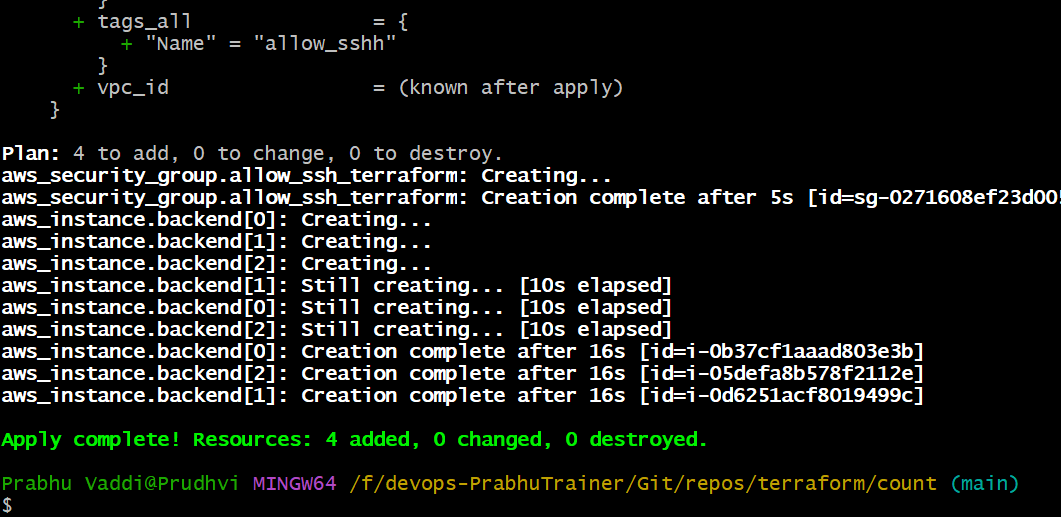


Terraform plan

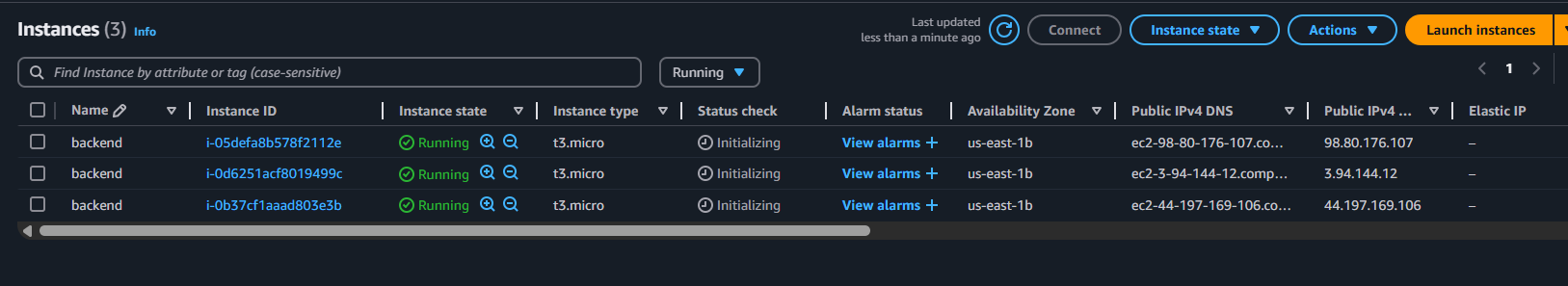




Terraform apply -auto-approve

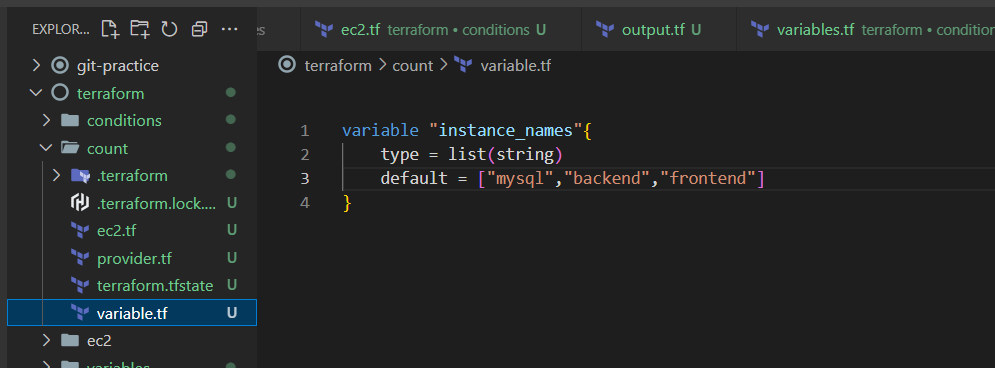


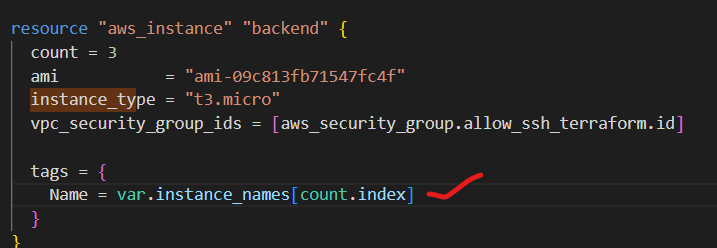
Validate EC2



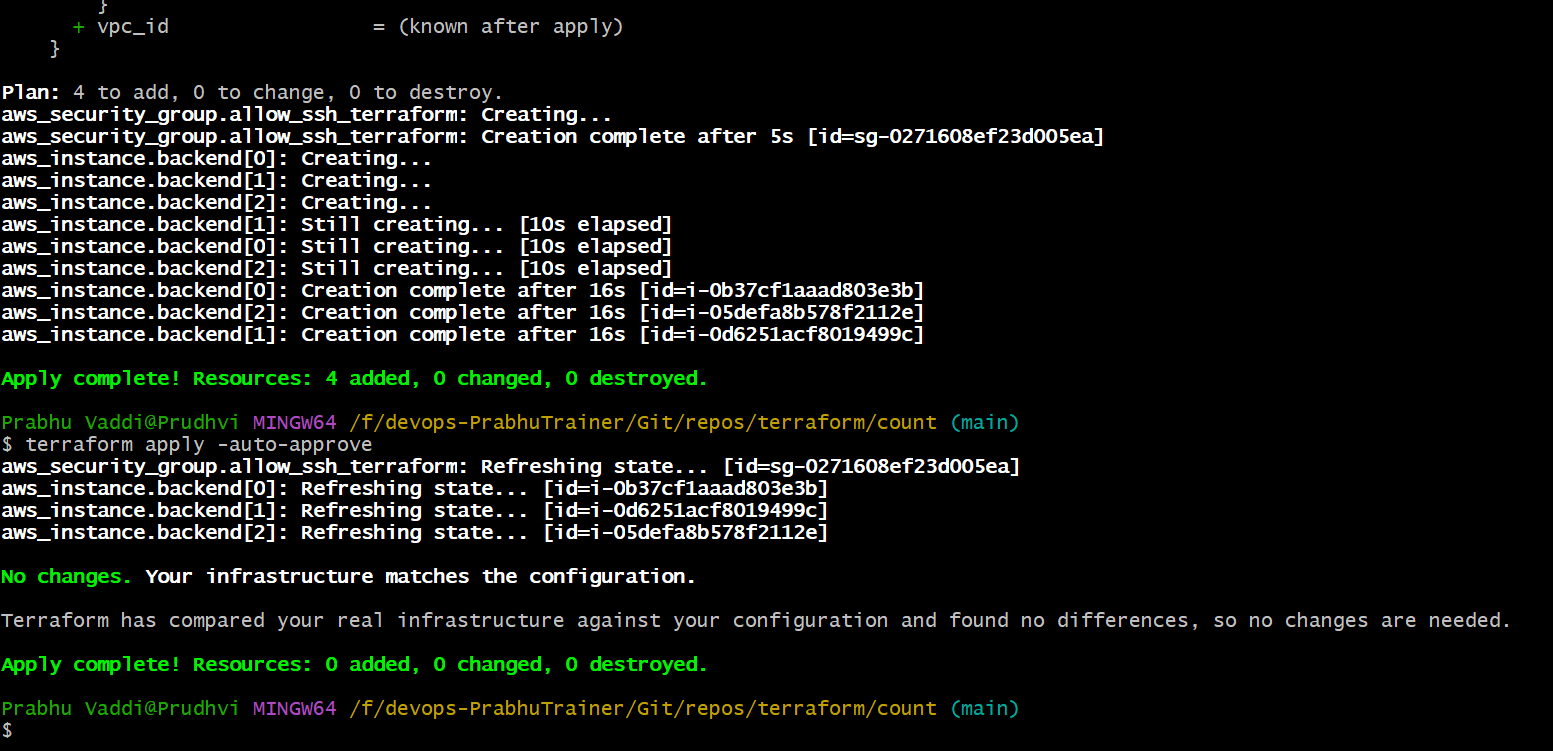
Problem – SAME NAME not correct

Count.Index formula for a list => Create a variable=> use list

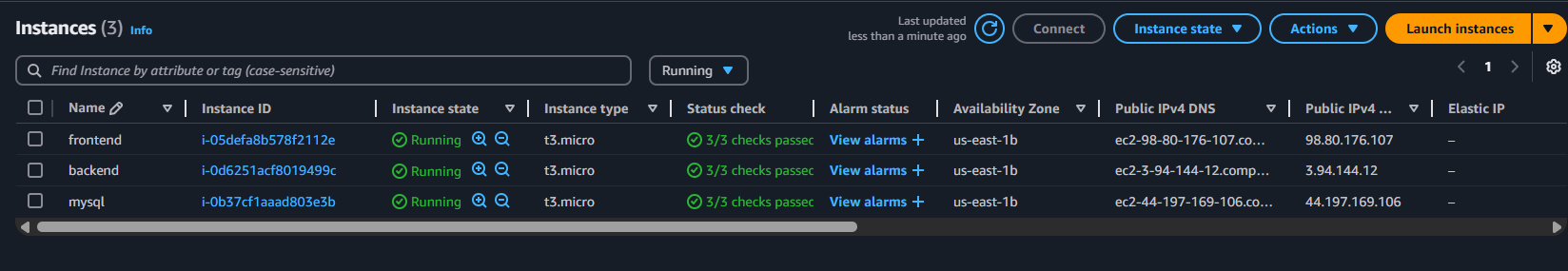




Update the tags using “Terraform apply -auto-approve”

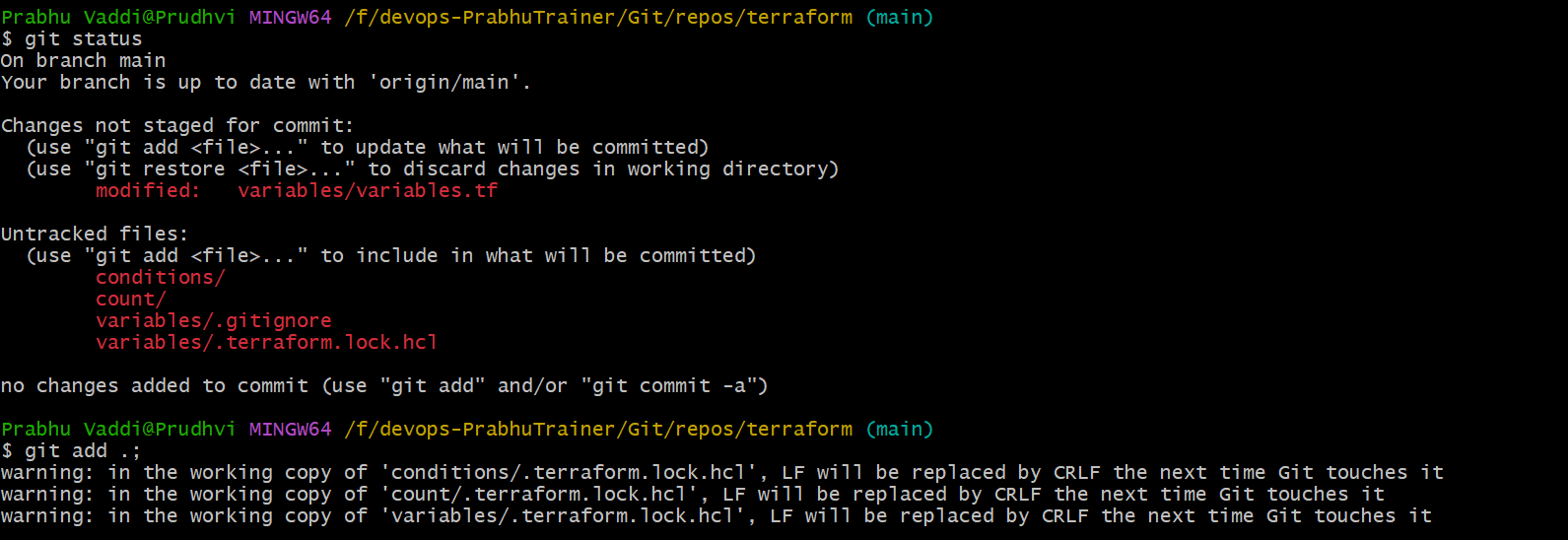


Validate EC2 intances



**GIT COMMIT & PUSH ERROR MESSAGE- Solution**

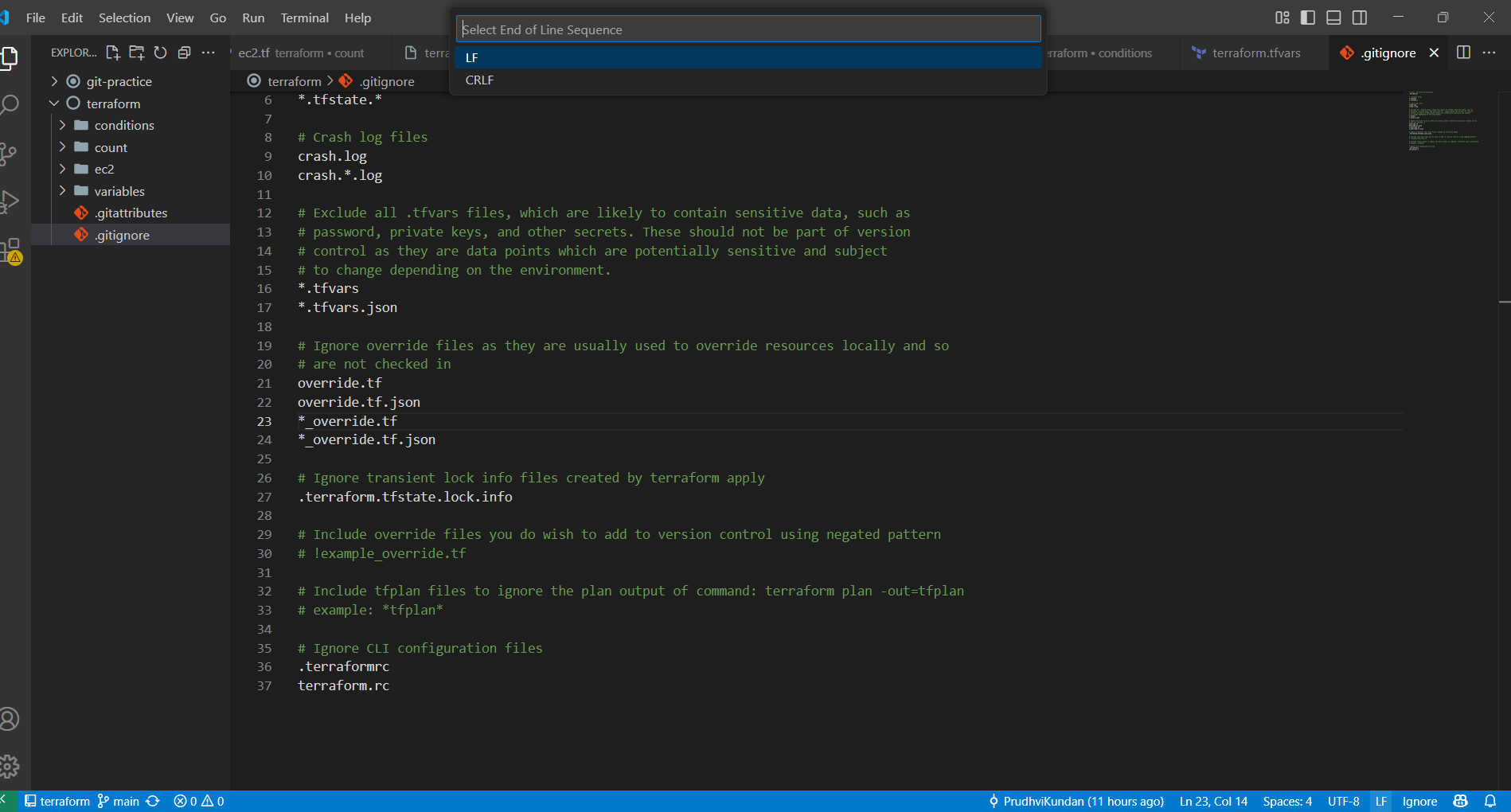
**Warning message**

****

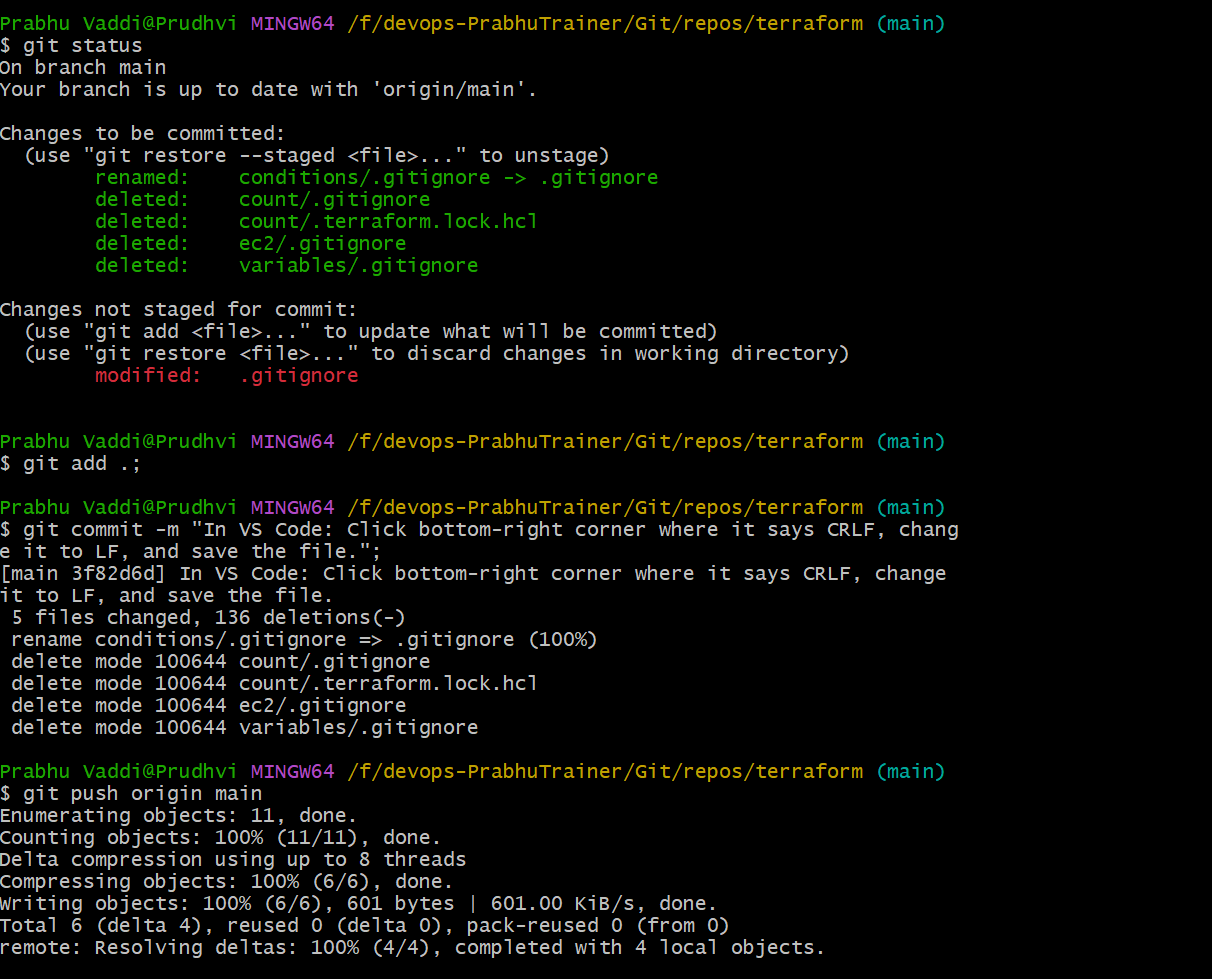
**Git Status**

**Main Option**

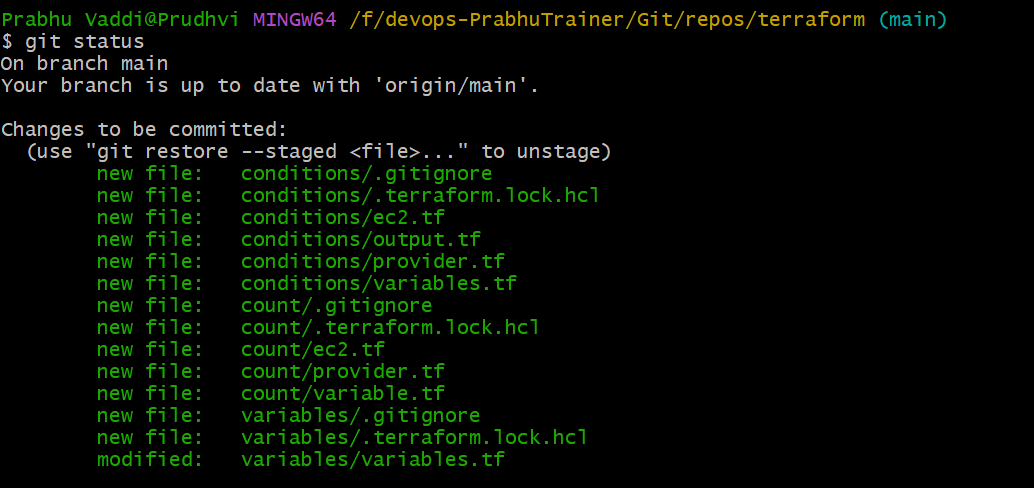
1. Or, use a text editor like **VS Code** or **Notepad++** to convert line endings:
   * In VS Code: Click bottom-right corner where it says CRLF, change it to LF, and save the file.
   * NOTE: .gitignore file exists in Project level.

****

**O/P**

****

**.gitattributes not required.**

****

**Step 1: Set Git to preserve LF endings**

CMD: git config --global core.autocrlf input



**Step 2: Add a .gitattributes file**



**PASTE & Save it: Recommended .gitattributes for Terraform/DevOps Projects**

# Enforce Unix-style LF line endings

\* text=auto eol=lf

# Terraform files

\*.tf text eol=lf

\*.tfvars text eol=lf

\*.hcl text eol=lf

\*.json text eol=lf

# Shell scripts

\*.sh text eol=lf

# YAML/Ansible/Kubernetes files

\*.yml text eol=lf

\*.yaml text eol=lf

# Ignore binary files from line ending checks

\*.zip binary

\*.exe binary

\*.tar binary

\*.gz binary

\*.png binary

\*.jpg binary

\*.jpeg binary

# Prevent .terraform folder from being tracked

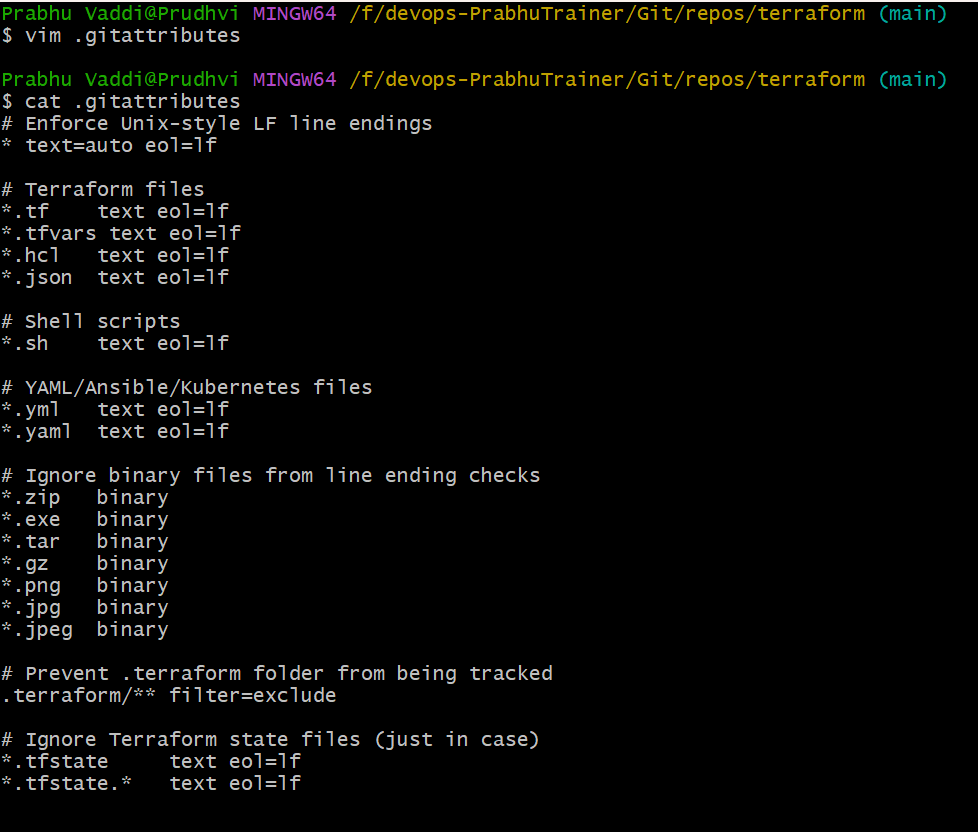
.terraform/\*\* filter=exclude

# Ignore Terraform state files (just in case)

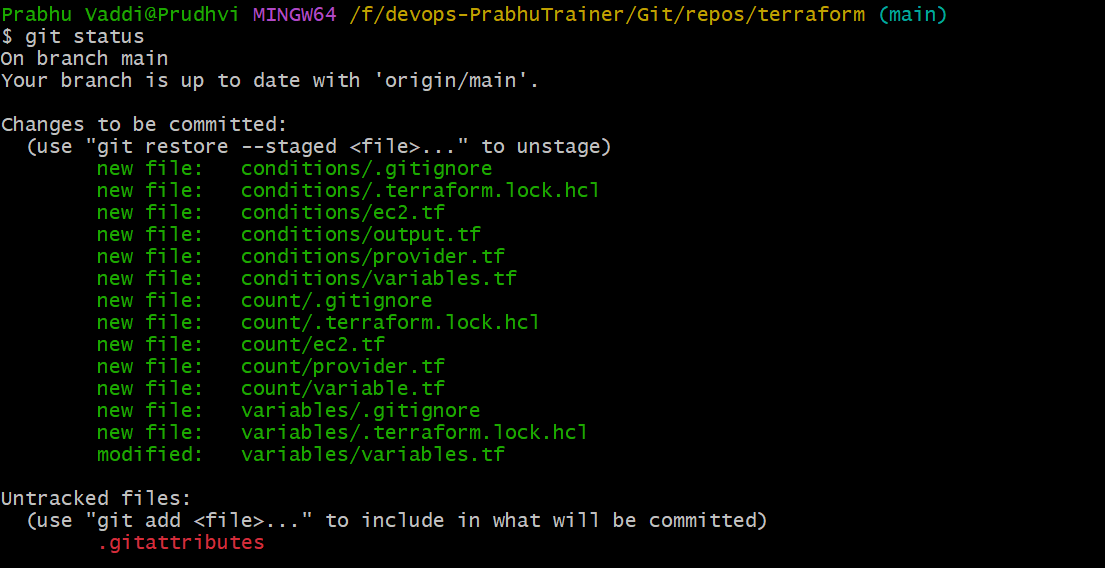
\*.tfstate text eol=lf

\*.tfstate.\* text eol=lf

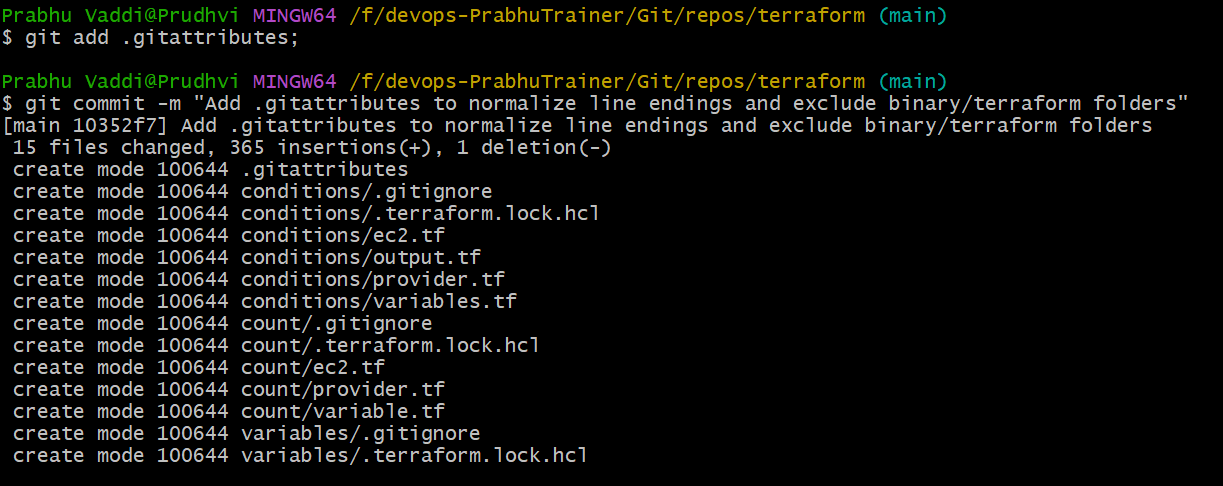
CMD : cat .gitattributes



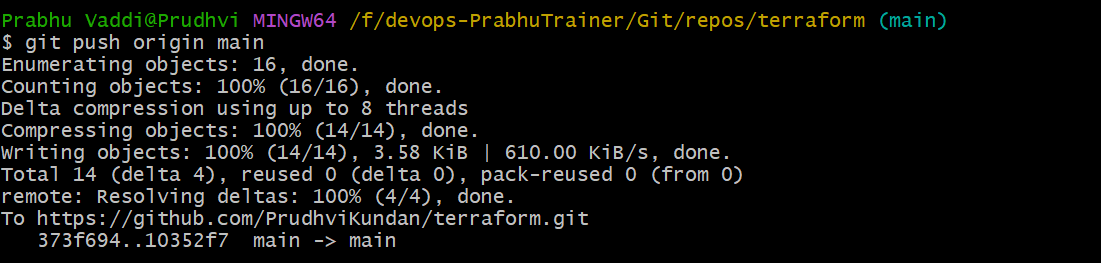
**GIT STATUS**



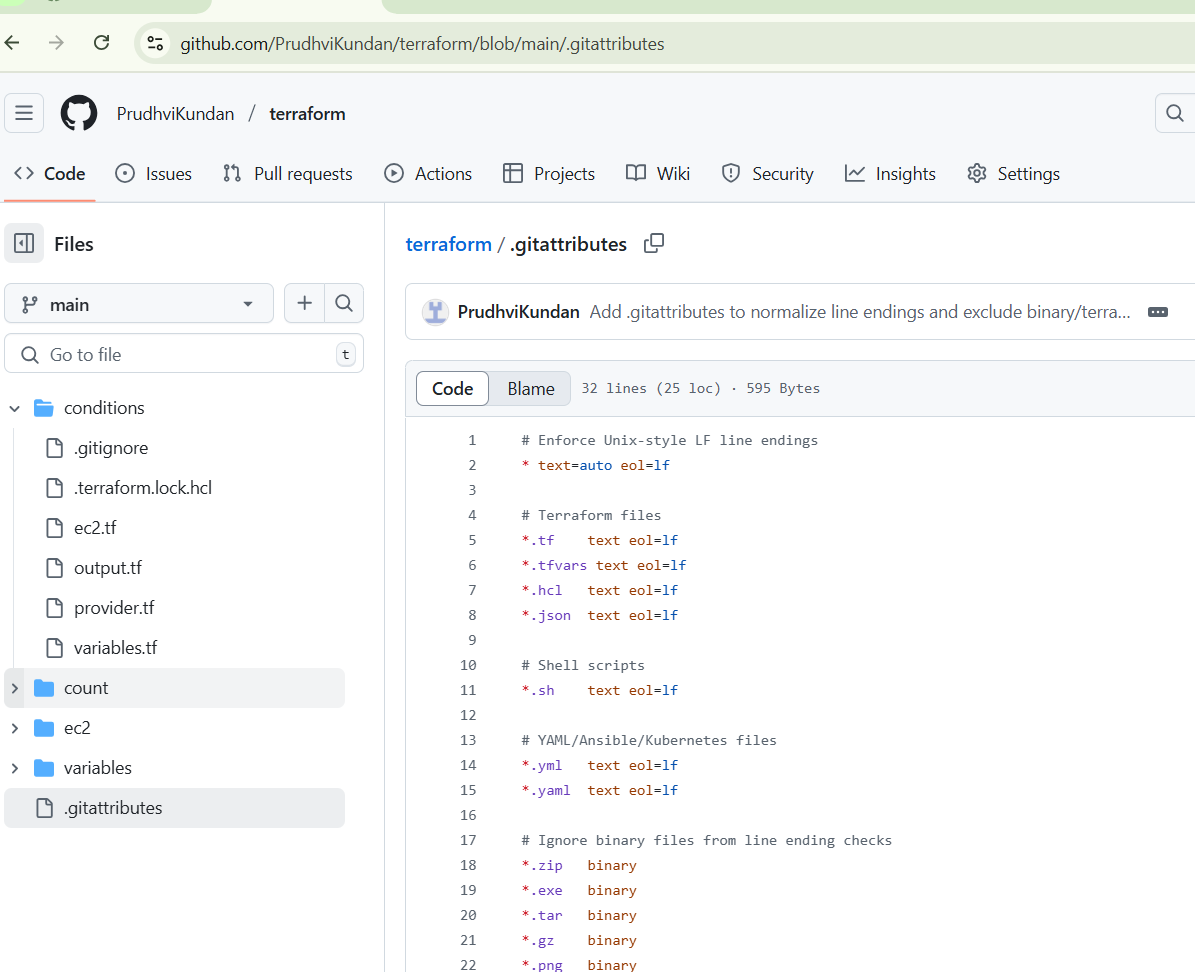
**GIT ADD & COMMIT**



**FINALLY: GIT PUSH**



**VALIDATE GIT REPO -**

****

**FUNCTIONS**