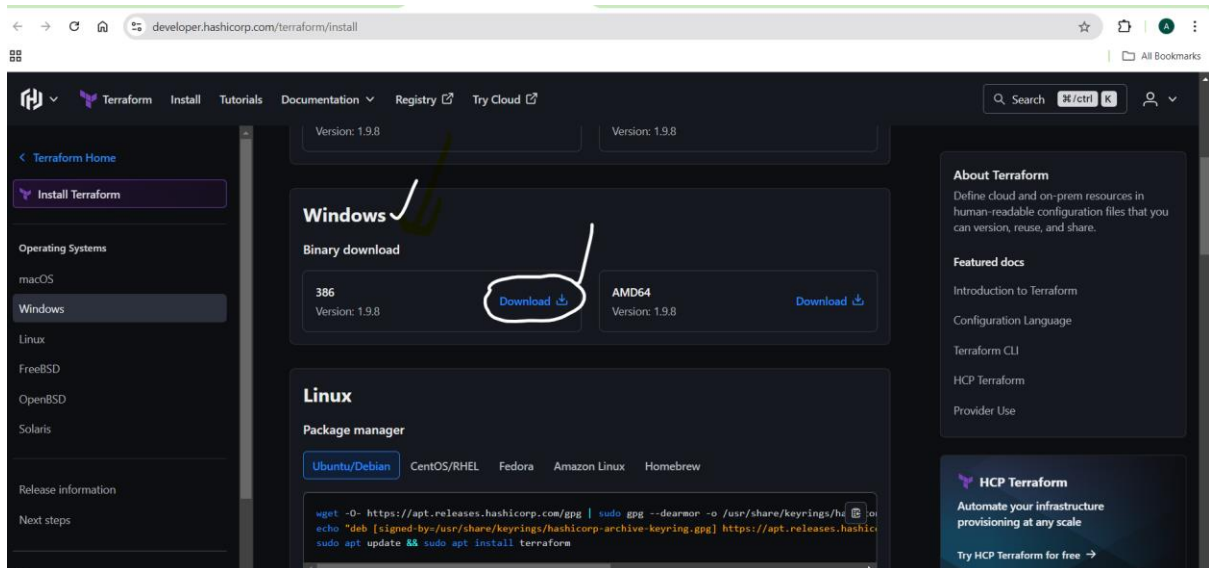


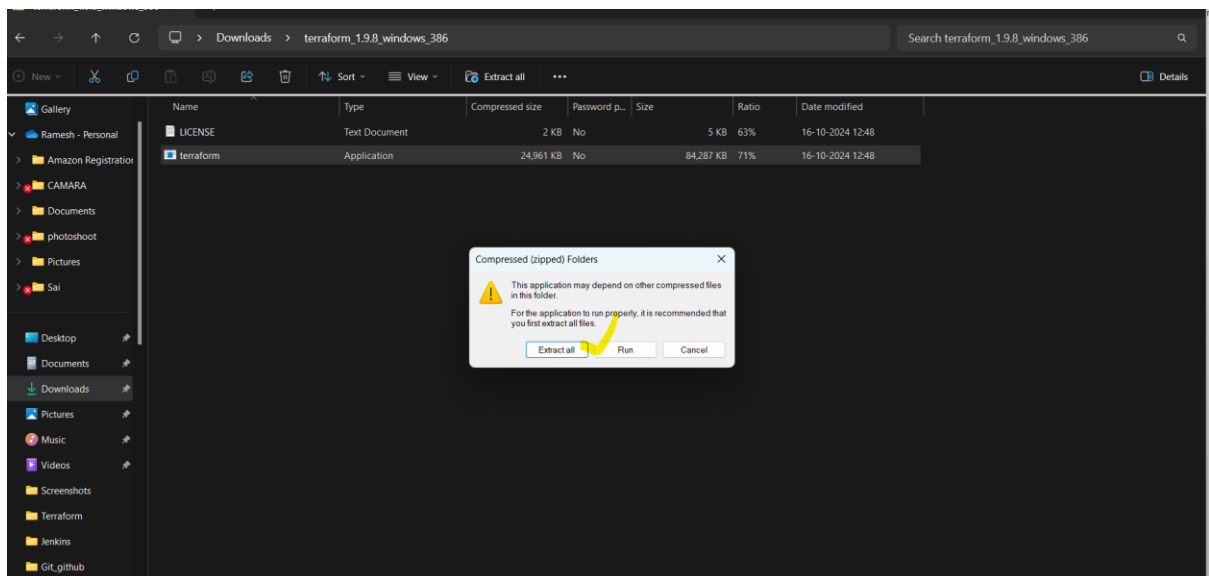
1) Install Terraform on your PC

To install terraform in our PC below steps we can follow.

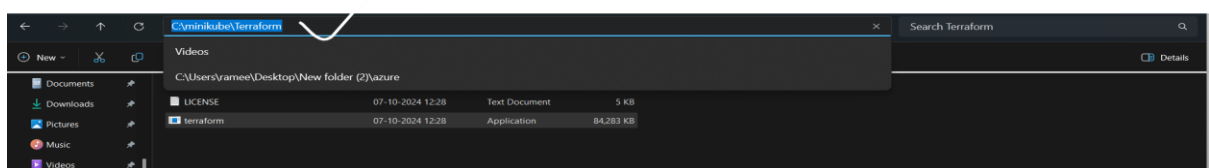
- Just click on this Link -- <https://developer.hashicorp.com/terraform/install>
- You see the below interface scroll down below there you find Windows.
- Just click on downloads.



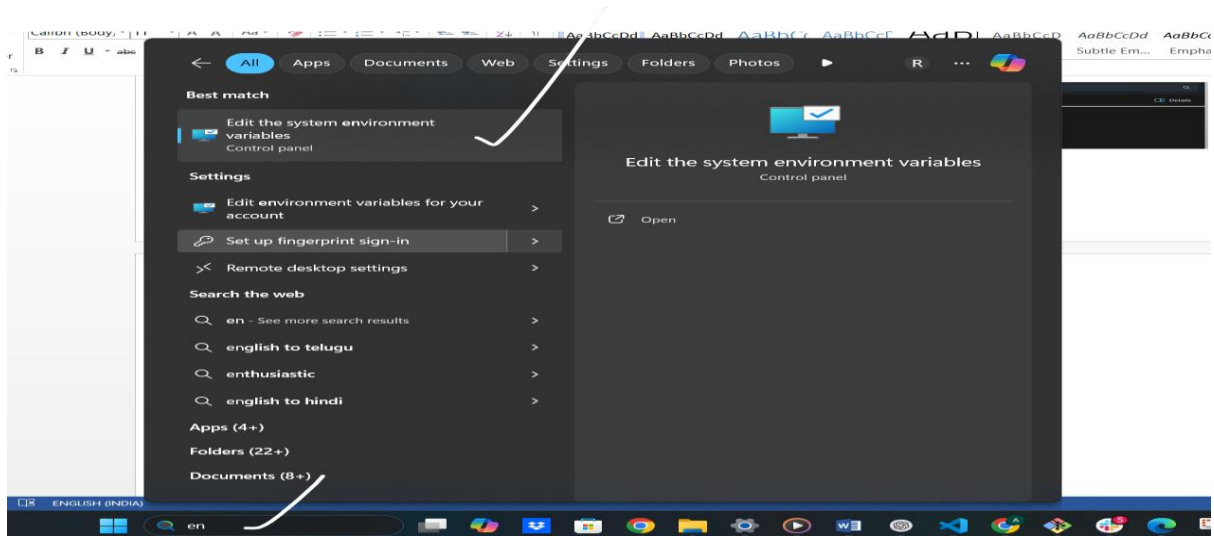
Extract the terraform folder.



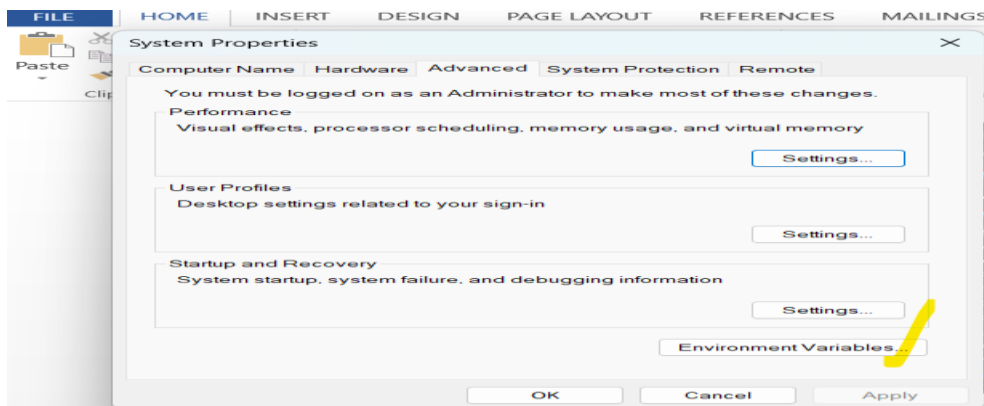
- Then go to the location of Terraform in C drive.
- Now this path I want to pass inn environment variables.



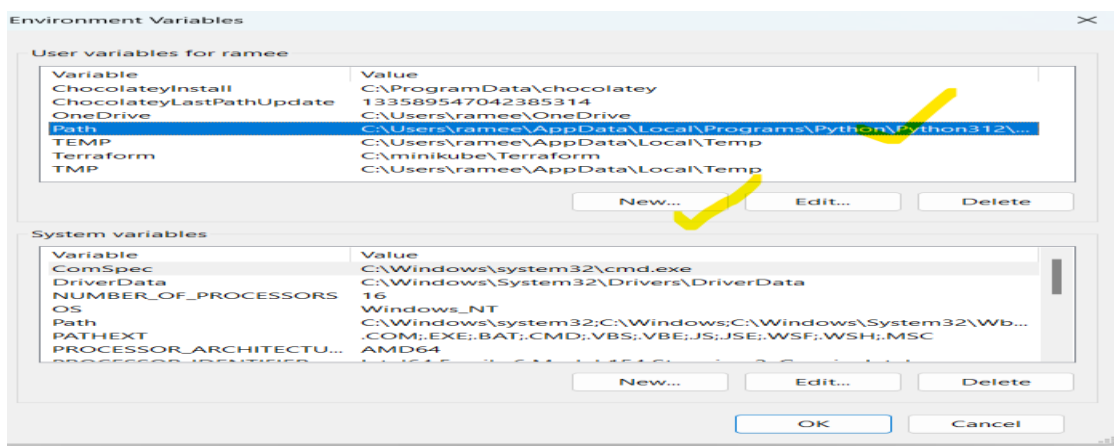
Search for system environments and click on that.

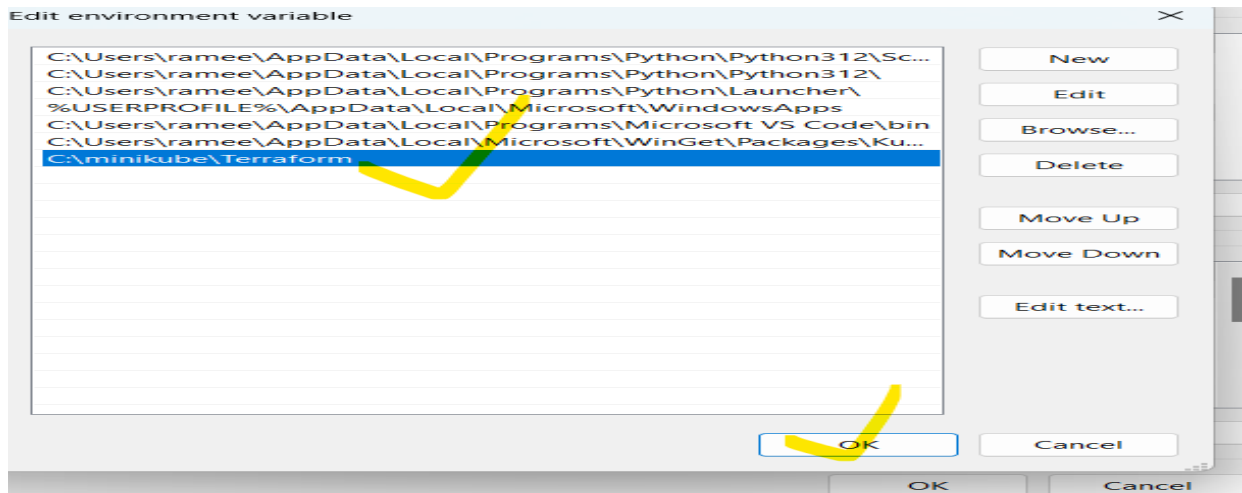


After that click on environmental variables



Select the Path and click on new.





Now path is Setup.

Now I went to the git bash check the terraform version.

CMD: terraform -v

```
ramee@Ramesh MINGW64 ~ (master)
$ terraform -v
Terraform v1.9.7
on windows_386

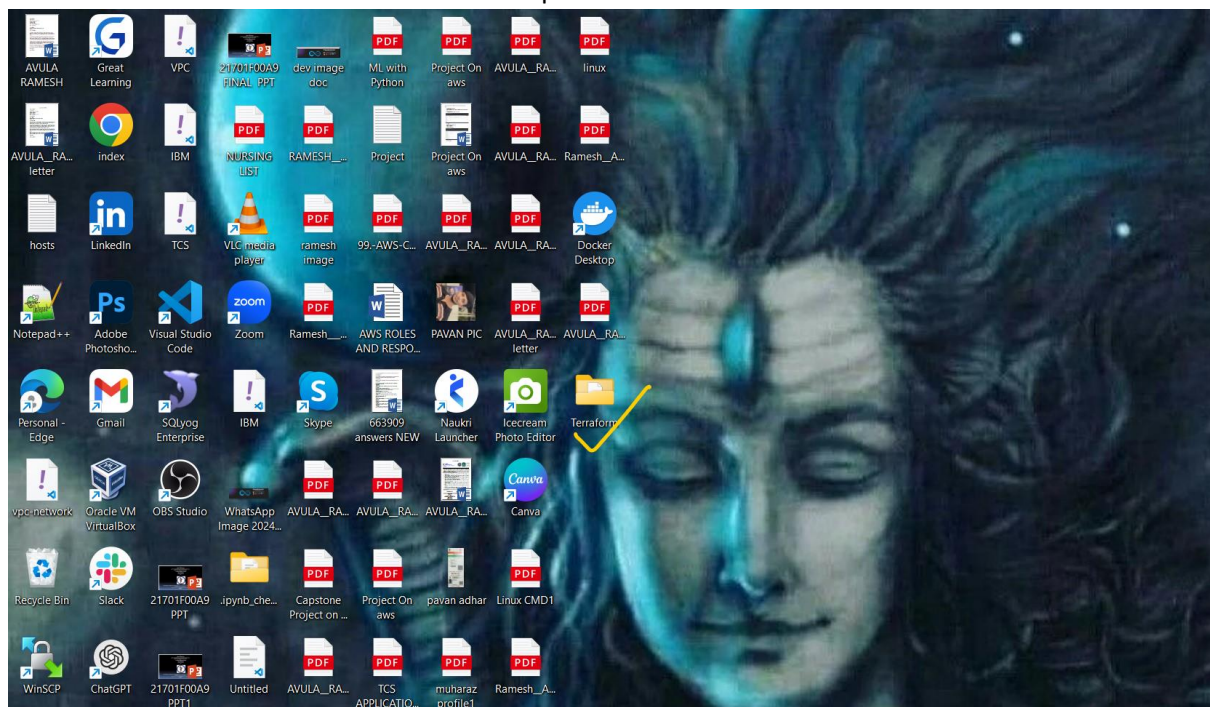
Your version of Terraform is out of date! The latest version
is 1.9.8. You can update by downloading from https://www.terraform.io/downloads.
html

ramee@Ramesh MINGW64 ~ (master)
$ |
```

2) Execute all the templates shown in video.

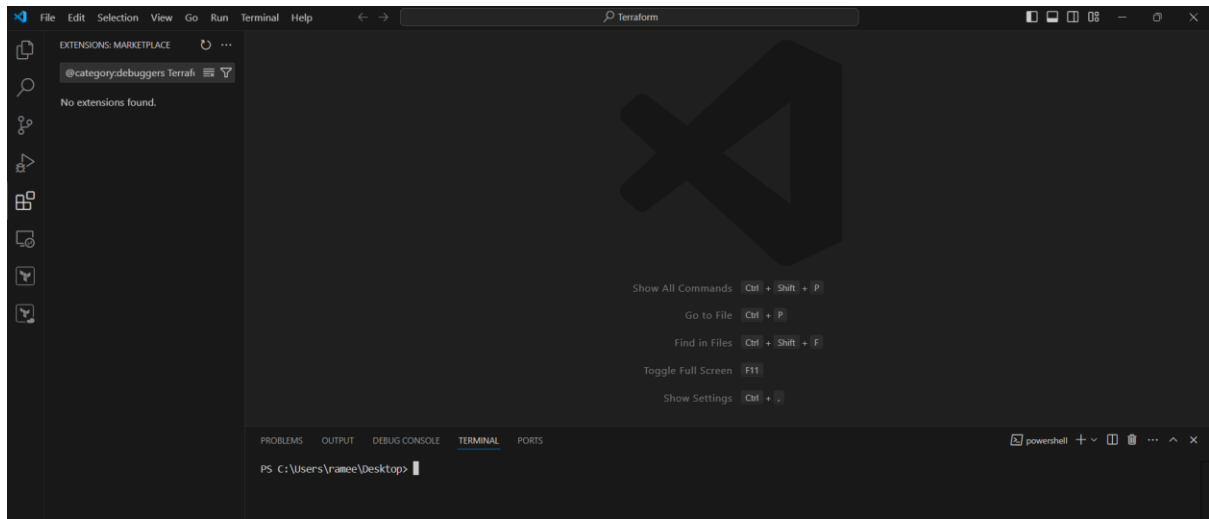
To execute the all templates created one separated folder.

Now we have to create one folder on Desktop with name Terraform.

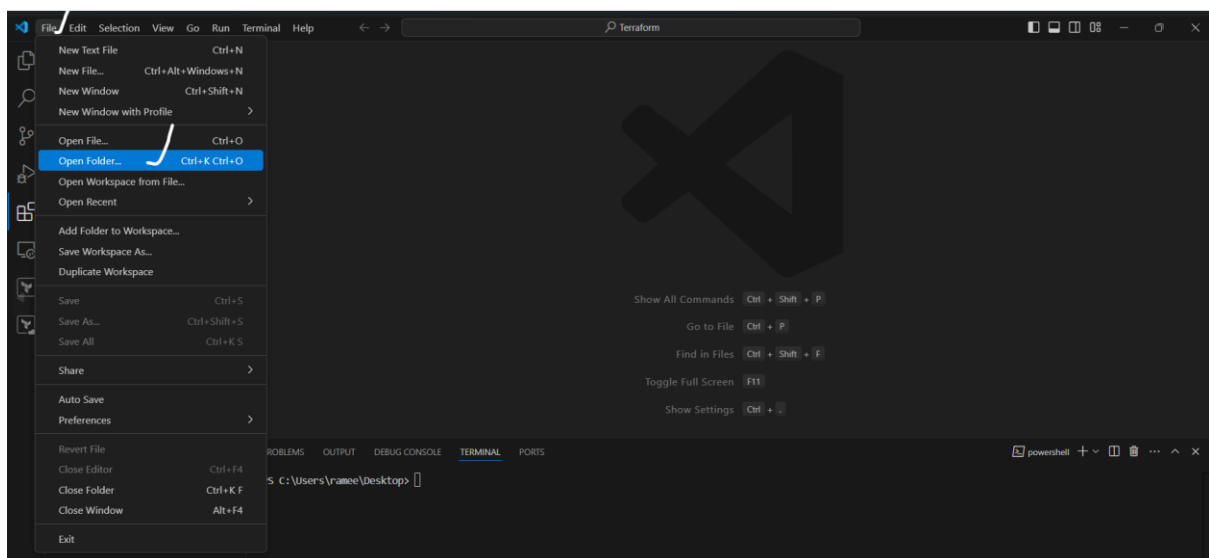


Now go to the Visual code.

You see the below that interface in visual code.

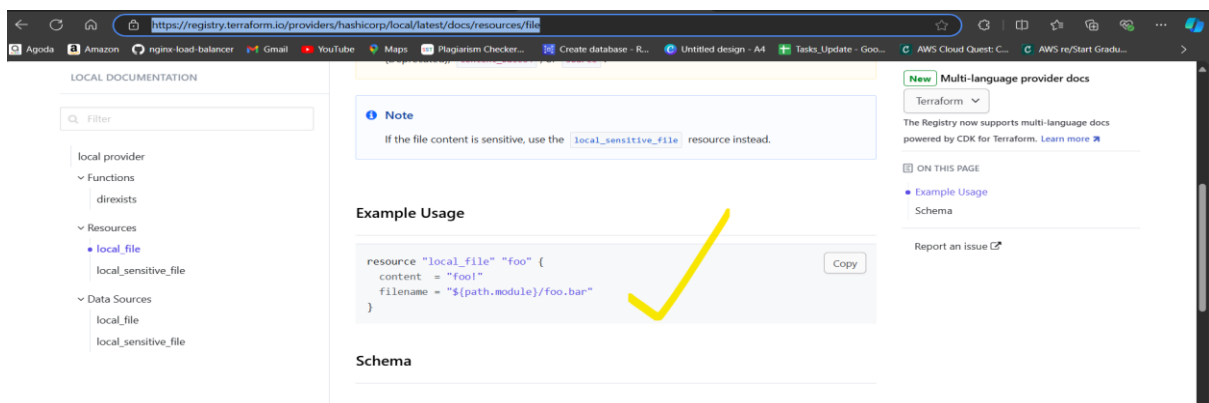


Now we have to click on the file and open folder.



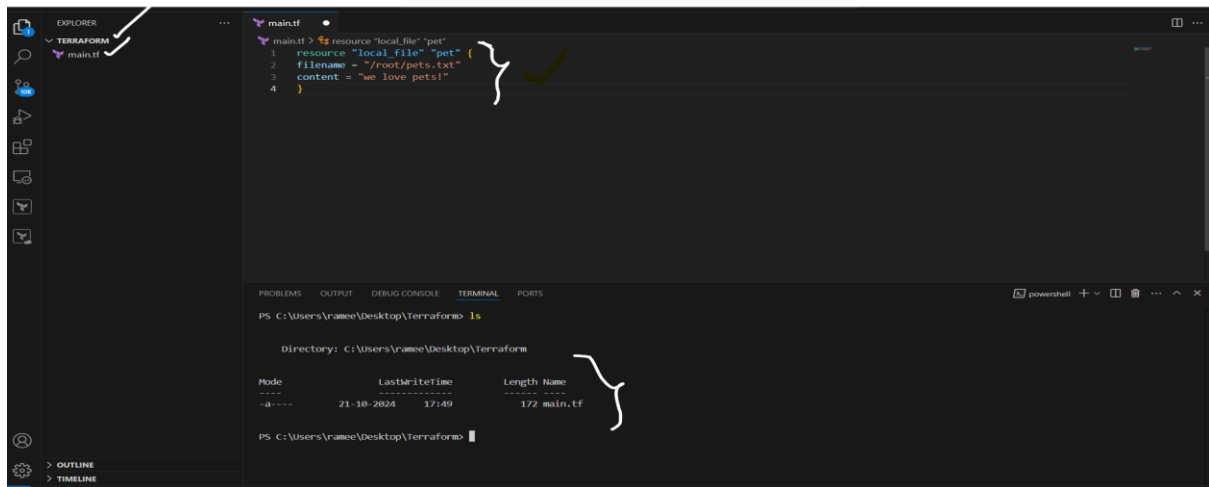
To create local template go to the terraform registry for reference.

Use the link -- [local_file | Resources | hashicorp/local | Terraform | Terraform Registry](https://registry.terraform.io/providers/hashicorp/local/latest/docs/resources/file)



Here I am selected the terraform folder and created main.tf file.

In the main.tf i am created one simple template for local only.

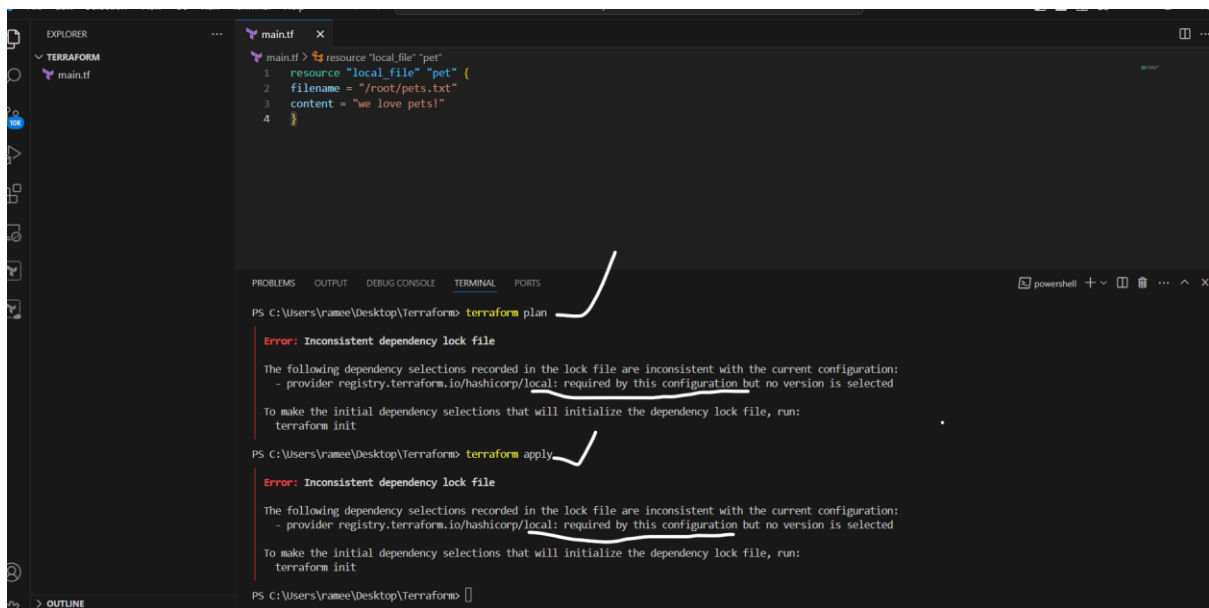


Now try to execute this template.

To check the blue print off the output I am entered the CMD: terraform plan

To create the output CMD: terraform apply.

Issue: provider registry.terraform.io/hashicorp/local: required by this configuration but no version is selected.

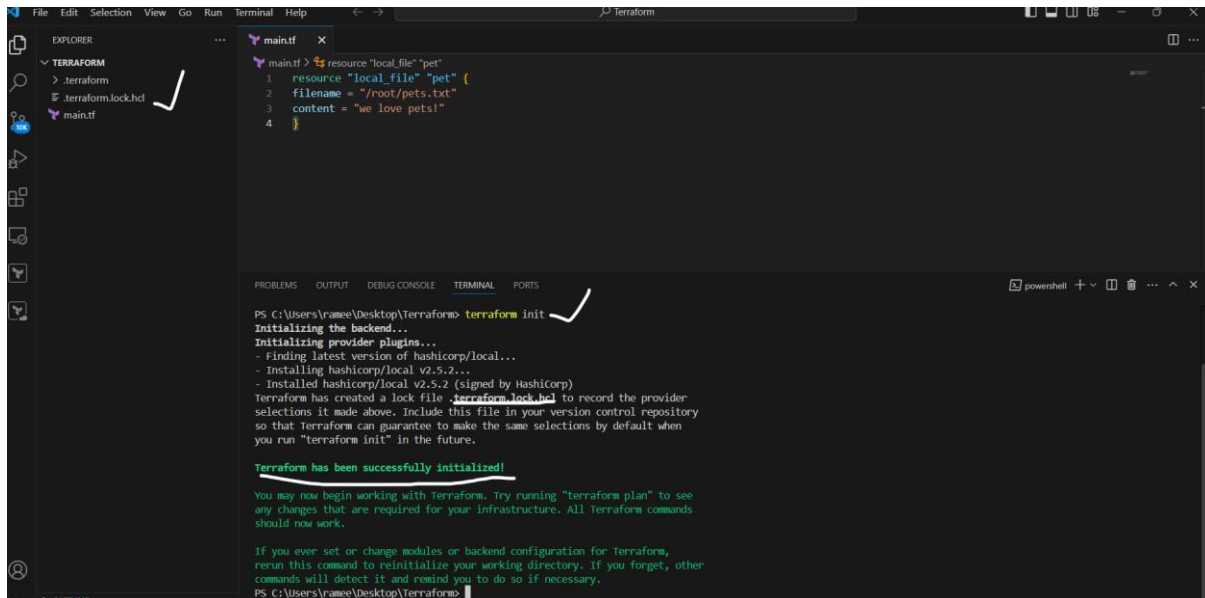


To solve this issue we need to initialize the repository and download the dependencies.

Using this CMD: `terraform init` ---- to initialize the repository and download the dependencies.

After that you see `.terraform.lock.hcl` file.

The `.terraform.lock.hcl` file ensures consistent provider versions across environments, preventing unexpected changes from version upgrades.



```
File Edit Selection View Go Run Terminal Help
Terraform

EXPLORER
TERRAFORM
> .terraform
.terraform.lock.hcl
main.tf

main.tf
1 resource "local_file" "pet" {
2   filename = "/root/pets.txt"
3   content = "we love pets!"
4 }
```

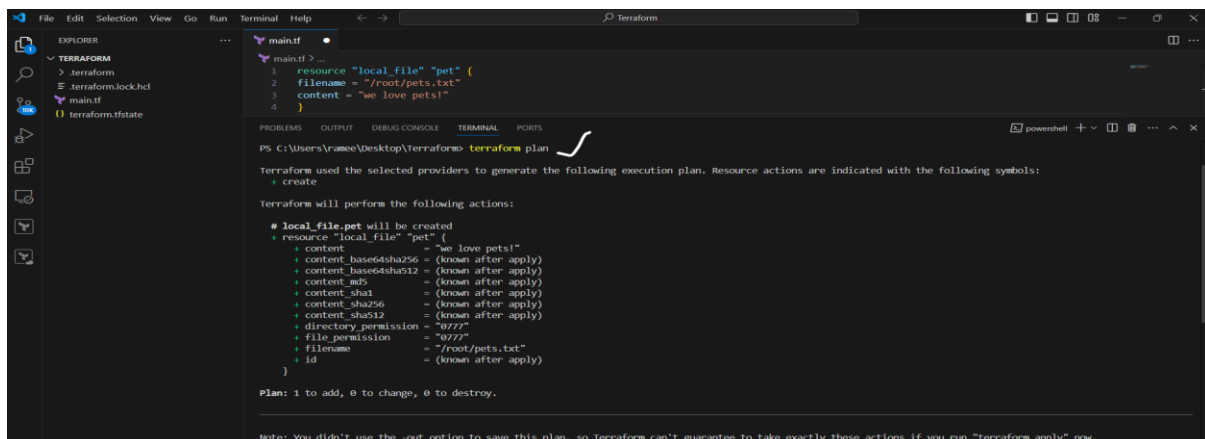
```
PS C:\Users\ramee\Desktop\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.2...
- Installed hashicorp/local v2.5.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
PS C:\Users\ramee\Desktop\Terraform>
```

After that I am enter the commands then it will executed.



```
File Edit Selection View Go Run Terminal Help
Terraform

EXPLORER
TERRAFORM
> .terraform
.terraform.lock.hcl
main.tf
terraform.tfstate

main.tf
1 resource "local_file" "pet" {
2   filename = "/root/pets.txt"
3   content = "we love pets!"
4 }
```

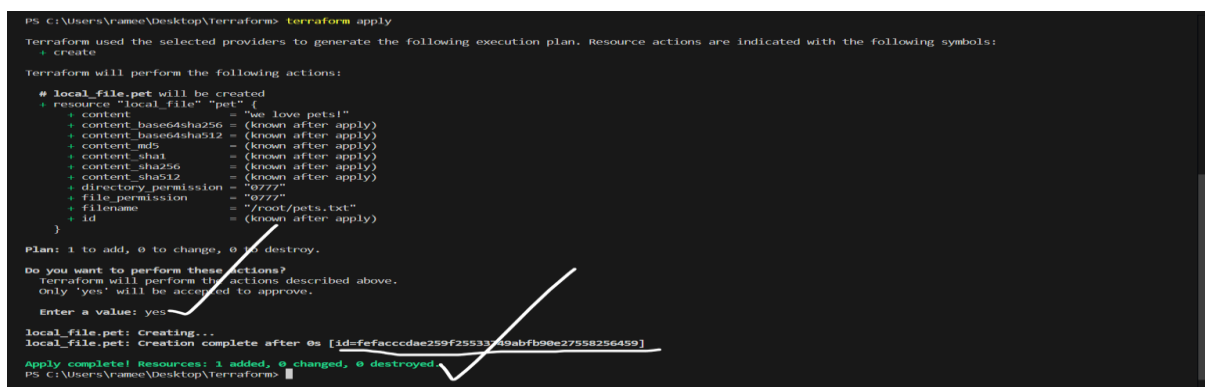
```
PS C:\Users\ramee\Desktop\Terraform> terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# local_file.pet will be created
+ resource "local_file" "pet" {
+   content = "we love pets!"
+   content_base64sha256 = (known after apply)
+   content_base64sha512 = (known after apply)
+   content_md5 = (known after apply)
+   content_sha1 = (known after apply)
+   content_sha256 = (known after apply)
+   content_sha512 = (known after apply)
+   directory_permission = "0777"
+   file_permission = "0777"
+   filename = "/root/pets.txt"
+   id = (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.
```



```
PS C:\Users\ramee\Desktop\Terraform> terraform apply
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:

# local_file.pet will be created
+ resource "local_file" "pet" {
+   content = "we love pets!"
+   content_base64sha256 = (known after apply)
+   content_base64sha512 = (known after apply)
+   content_md5 = (known after apply)
+   content_sha1 = (known after apply)
+   content_sha256 = (known after apply)
+   content_sha512 = (known after apply)
+   directory_permission = "0777"
+   file_permission = "0777"
+   filename = "/root/pets.txt"
+   id = (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

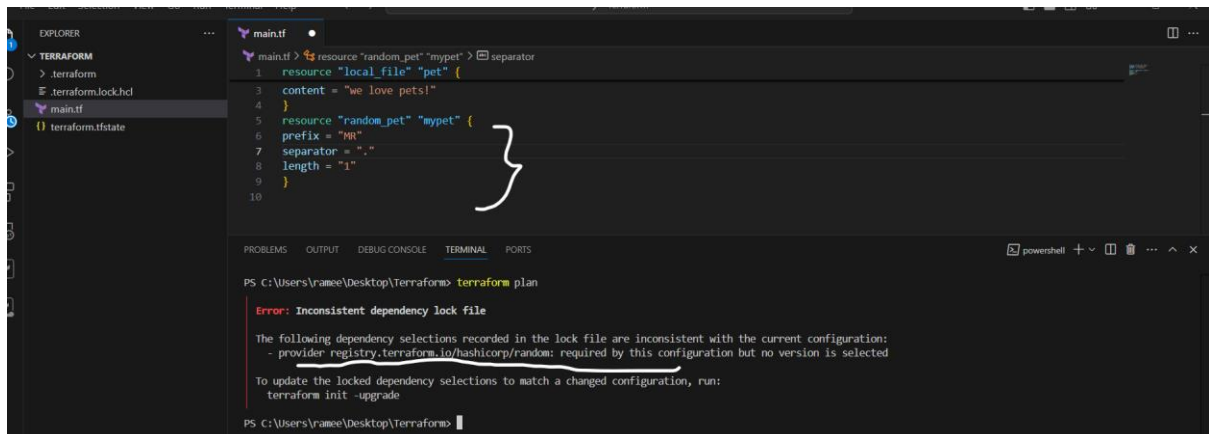
Enter a value: yes
local_file.pet: Creating...
local_file.pet: Creation complete after 0s [id=fe6acccdae259f2553359abfb90e27558256459]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed
PS C:\Users\ramee\Desktop\Terraform>
```

Again I am creating another resource provider as Random

After adding the resource we need to save.

Again I a traied to see the plan but I am not able to see.



The screenshot shows a VS Code editor with a file named `main.tf` containing the following Terraform configuration:

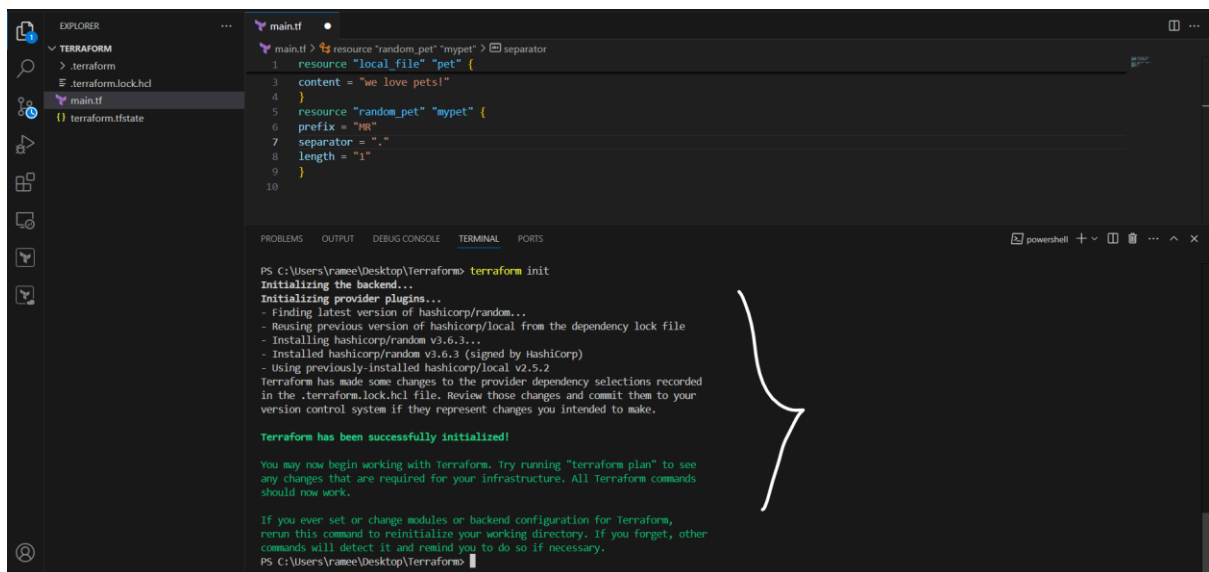
```
1 resource "local_file" "pet" {
2   content = "we love pets!"
3 }
4
5 resource "random_pet" "mypet" {
6   prefix = "my"
7   separator = "-"
8   length = "1"
9 }
10
```

A white curly brace is drawn around the `random_pet` resource block. Below the code, the terminal window shows the output of the `terraform plan` command:

```
PS C:\Users\ramee\Desktop\Terraform> terraform plan
Error: Inconsistent dependency lock file
The following dependency selections recorded in the lock file are inconsistent with the current configuration:
- provider registry.terraform.io/hashicorp/random: required by this configuration but no version is selected
To update the locked dependency selections to match a changed configuration, run:
terraform init -upgrade
PS C:\Users\ramee\Desktop\Terraform>
```

The provider changed so gain I need to initialize the random provider.

So we need to use the CMD: `terraform init` – it will download the all dependencies of random provider.



The screenshot shows the VS Code editor with the same `main.tf` file. The terminal window now shows the output of the `terraform init` command:

```
PS C:\Users\ramee\Desktop\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Reusing previous version of hashicorp/random from the dependency lock file
- Installing hashicorp/random v3.6.3...
- Installed hashicorp/random v3.6.3 (signed by HashiCorp)
- Using previously-installed hashicorp/local v2.5.2
Terraform has made some changes to the provider dependency selections recorded in the .terraform.lock.hcl file. Review those changes and commit them to your version control system if they represent changes you intended to make.

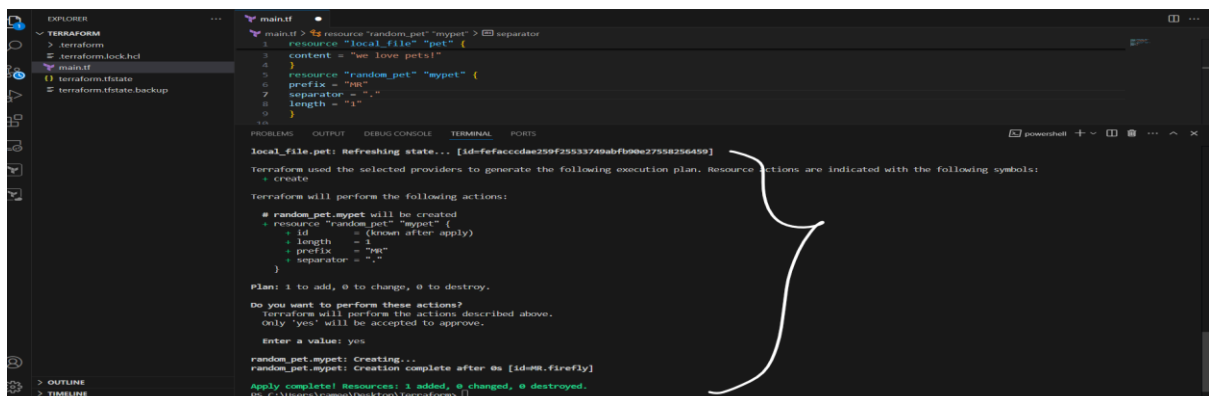
Terraform has been successfully initialized!

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.
PS C:\Users\ramee\Desktop\Terraform>
```

A white curly brace is drawn around the success message and the instructions for using Terraform.

Now you can run apply.



The screenshot shows the VS Code editor with the same `main.tf` file. The terminal window now shows the output of the `terraform apply` command:

```
PS C:\Users\ramee\Desktop\Terraform> terraform apply
local_file.pet: Refreshing state... [id=fe6cccdde259f2553749abf00e2758256459]
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:

# random_pet.mypet will be created
+ resource "random_pet" "mypet" {
+   id           = (known after apply)
+   length       = 1
+   prefix       = "my"
+   separator    = "-"
+ }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

random_pet.mypet: Creating...
random_pet.mypet: Creation complete after 0s [id=dm.firefly]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform>
```

A white curly brace is drawn around the plan and the confirmation prompt.

Now it's successfully Done.

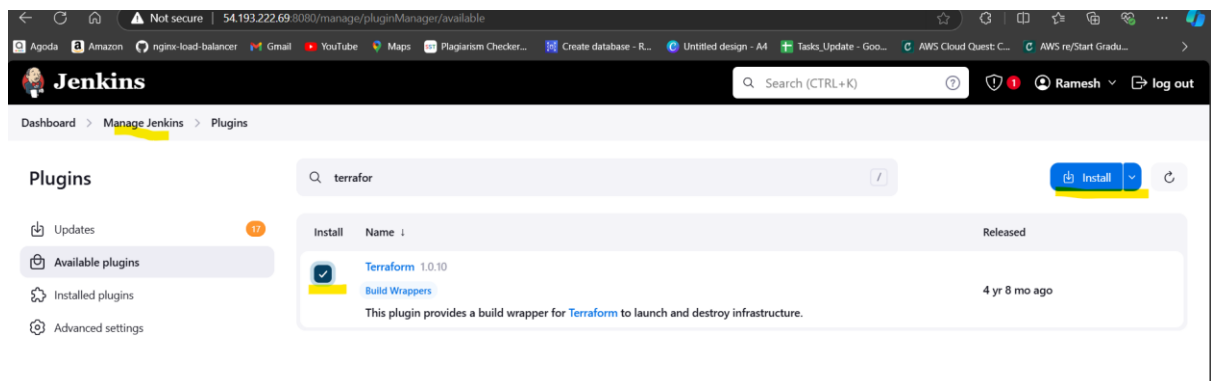
3) Note down below points,

- **Terraform init:** Terraform Init initializes the working directory, downloads necessary plugins, and sets up backend configuration to manage the state file.
- **Terraform Plan:** Terraform Plan creates a detailed preview of the changes that will be made, helping us verify the resources to be added, modified, or destroyed without making any actual changes.
- **Terraform Apply:** Terraform Apply executes the changes identified in the plan, provisioning or modifying the resources. It updates the state file to keep track of the infrastructure.
- **Terraform Provider:** Terraform Providers are plugins that allow Terraform to interact with specific cloud platforms like AWS or Azure. They define the resources and services Terraform can manage.

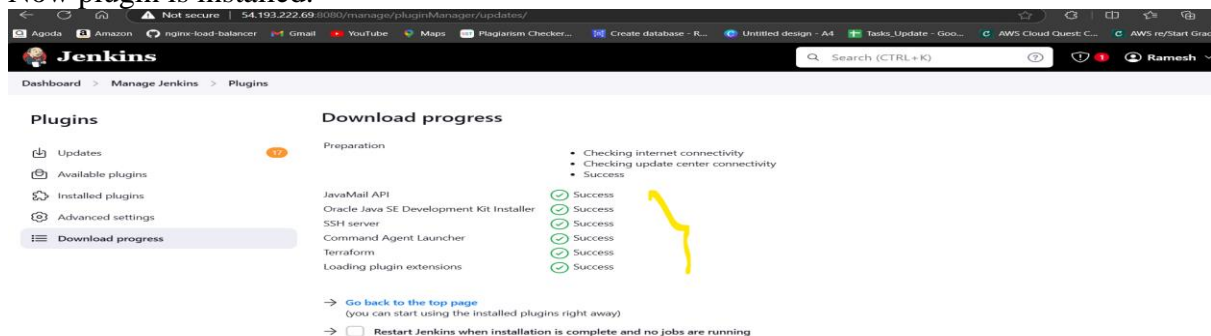
3) Integrate a sample Terraform template in jenkins.

To integrate a sample Terraform template in Jenkins.

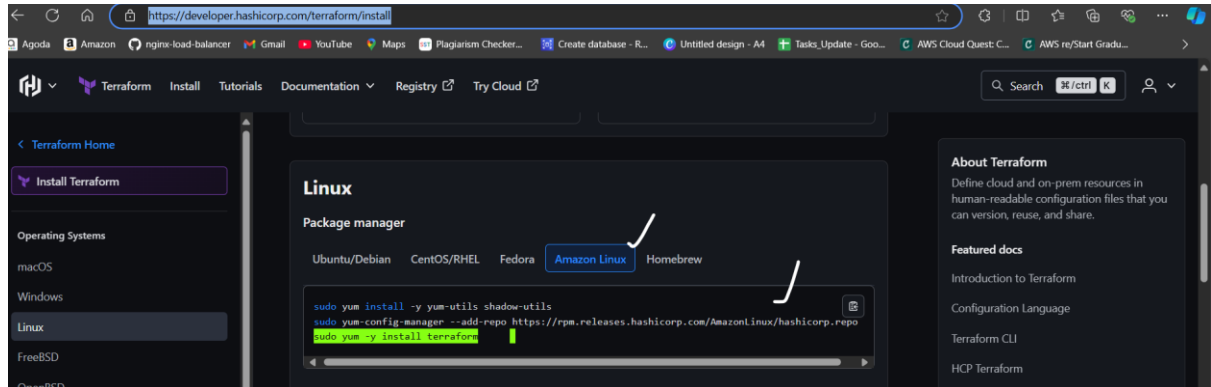
- First we need to launch ec2 server.
- With in the server you need to install Jenkins.
- Go to the Jenkins GUI.
- Go to manage Jenkins > Plugins.
- Click on Available plugins there you need to search terraform plugin.
- Click on check box and install.



Now plugin is installed.



- Now we want to download terraform in the Ec2 server where Jenkins is installed.
- To install terraform click on the Link : [Install | Terraform | HashiCorp Developer](https://developer.hashicorp.com/terraform/install)
- Here I am selecting amazon Linux based on your requirement you can select.



This all commands you need to perform the in the server.

```
[root@ip-172-31-1-15 ~]# sudo yum install -y yum-utils shadow-utils
Last metadata expiration check: 19:20:31 ago on Wed Oct 23 14:58:07 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Package shadow-utils-2.14.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-1-15 ~]# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
[root@ip-172-31-1-15 ~]# sudo yum -y install terraform
Hashicorp Stable - x86_64                                13 MB/s | 1.4 MB   00:00
Dependencies resolved.
=====
Package      Architecture      Version      Repository      Size
=====
Installing:
terraform     x86_64            1.9.8-1      hashicorp       27 M
Transaction Summary
Install 1 Package
Total download size: 27 M
Installed size: 85 M
Downloading Packages:
terraform-1.9.8-1.x86_64.rpm                                49 MB/s | 27 MB   00:00
Total
Hashicorp Stable - x86_64                                49 MB/s | 27 MB   00:00
Importing GPG key 0xA621E701:
  Userid : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
  Fingerprint: 798A EC65 4E5C 1542 8CBE 42EE AA16 FCBC A621 E701
  From    : https://rpm.releases.hashicorp.com/gpg
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Installing     : terraform-1.9.8-1.x86_64                  1/1
  Verifying      : terraform-1.9.8-1.x86_64                  1/1
=====
WARNING:
A newer release of "Amazon Linux" is available.
```

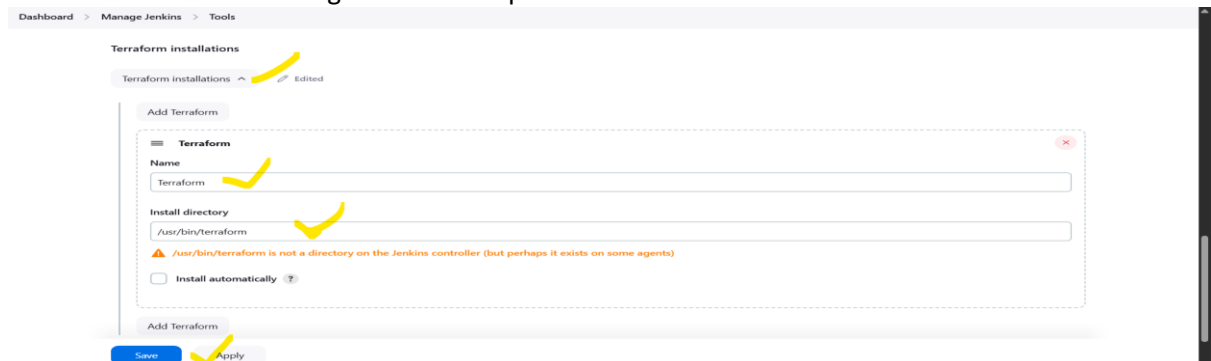
Now terraform is downloaded.

Terraform path you need to give in Jenkins GUI.

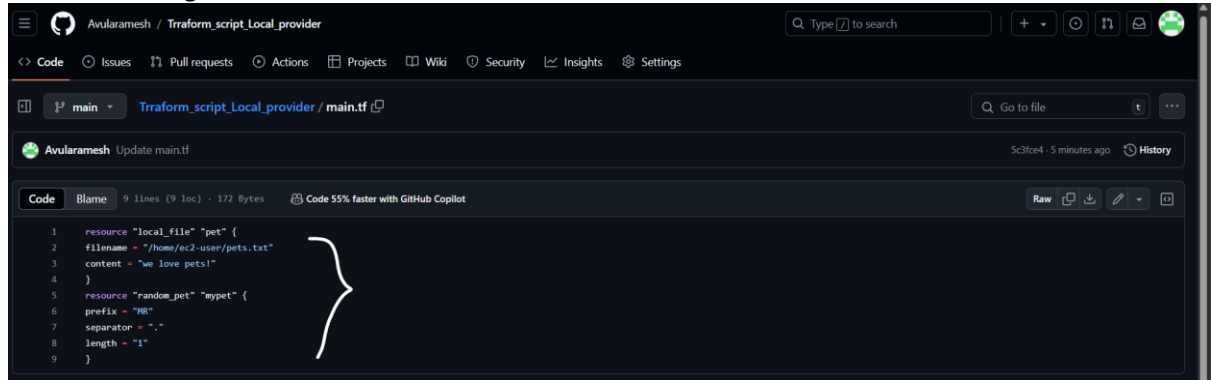
```
[root@ip-172-31-1-15 ~]# which terraform
/usr/bin/terraform
```

Go to the manage jenkins > tools

Click on add terraform----- give name and path of the terraform and save.

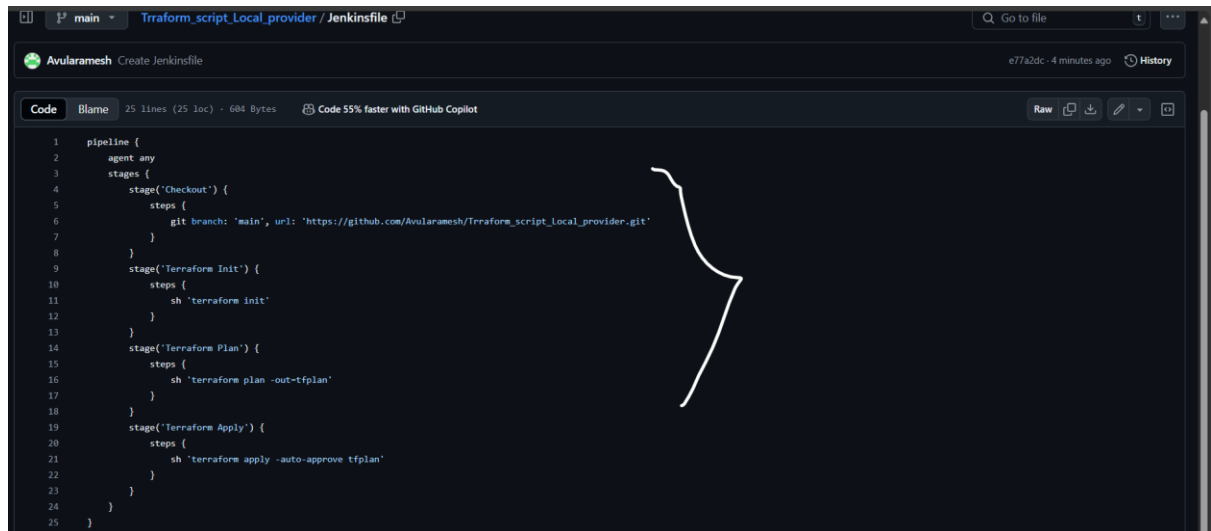


Now I went to github and there I created one main.tf file.



```
1 resource "local_file" "pet" {
2   filename = "/home/ec2-user/pets.txt"
3   content = "we love pets!"
4 }
5 resource "random_pet" "mypet" {
6   prefix = "MR"
7   separator = "-"
8   length = "1"
9 }
```

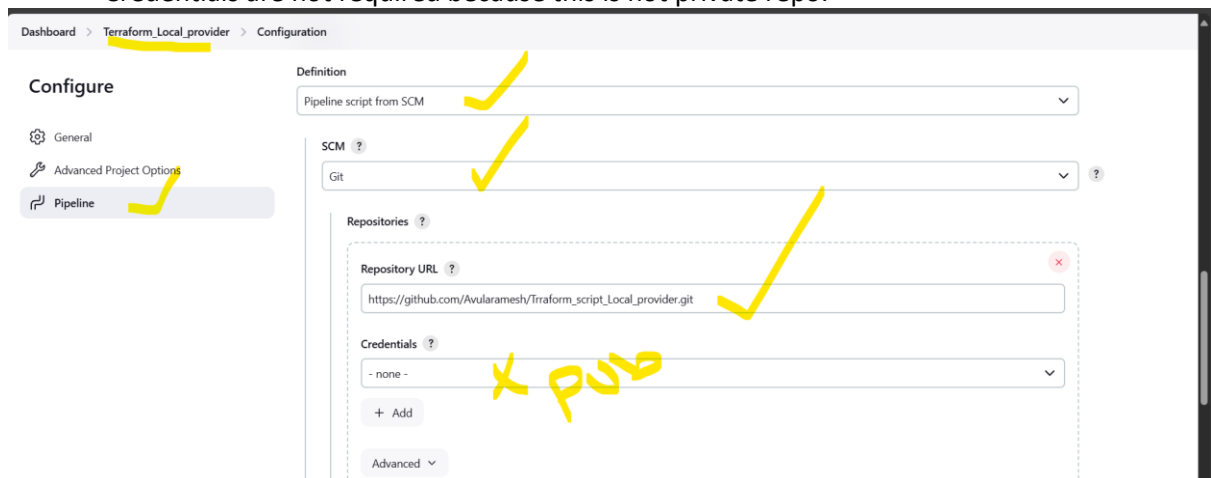
And also created Jenkinsfile.



```
1 pipeline {
2   agent any
3   stages {
4     stage('Checkout') {
5       steps {
6         git branch: 'main', url: 'https://github.com/Avularamesh/Terraform_script_Local_provider.git'
7       }
8     }
9     stage('Terraform Init') {
10      steps {
11        sh 'terraform init'
12      }
13    }
14    stage('Terraform Plan') {
15      steps {
16        sh 'terraform plan -out=tfplan'
17      }
18    }
19    stage('Terraform Apply') {
20      steps {
21        sh 'terraform apply -auto-approve tfplan'
22      }
23    }
24  }
25 }
```

Now I went to the Jenkins GUI and I am created the one pipelinejob.

- The job name given as ---- Terraform_Local_provider
- Select job – pipeline
- Click on ok.
- After that go to the pipeline and select ---- pipeline script from SCM
- Give repo URL
- Credentials are not required because this is not private repo.



Dashboard > Terraform_Local_provider > Configuration

Configure

General

Advanced Project Options

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/Avularamesh/Terraform_script_Local_provider.git

Credentials ?

- none -

+ Add

Advanced

-
- Dashboard > Terraform_Local_provider > Configuration
- ## Configure
- General
 - Advanced Project Options
 - Pipeline**
- Branch Specifier** (blank for any) ?
- Add Branch
- Repository browser** ?
- (Auto)
- Additional Behaviours**
- Add
- Script Path** ?
-
- ☒ **Lightweight checkout** ?
- [Pipeline Syntax](#)
- Save Apply

```
Dashboard > Terraform_Local_provider > #10
```

```
[1] 11:11:11 [1] 1
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Terraform Plan)
[Pipeline] sh
+ terraform plan -out=tfplan
@0m@1mrandom_pet.mypet: Refreshing state... [id=MR.peacock]@0m
@0m@1mlocal_file.pet: Refreshing state... [id=fefacccdae259f25533749abfb90e27558256459]@0m

@0m@1m@32mNo changes.@0m@1m Your infrastructure matches the configuration.@0m

@0mTerraform has compared your real infrastructure against your configuration
and found no differences, so no changes are needed.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Terraform Apply)
[Pipeline] sh
+ terraform apply -auto-approve tfplan
@0m@1m@32m
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
@0m
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

```
#  
~\##### Amazon Linux 2023  
~~\#####\  
~~\####|  
~~\#/  
~~V~' '-> https://aws.amazon.com/linux/amazon-linux-2023  
  
~~~~  
~~~~.  
~~~~./..  
~~~~/_m/' -
```

Last login: Thu Oct 24 10:26:23 2024 from 106.222.229.103
[ec2-user@ip-172-31-1-15 ~]\$ ls
food pets.txt
[ec2-user@ip-172-31-1-15 ~]\$ cat pets.txt
we love pets! [ec2-user@ip-172-31-1-15 ~]\$ █

Task is Done.