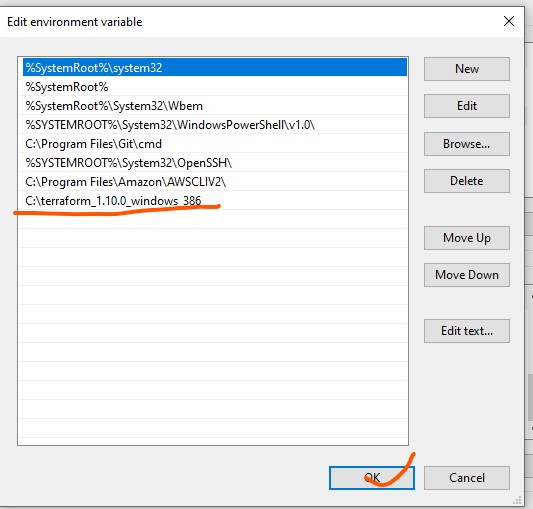
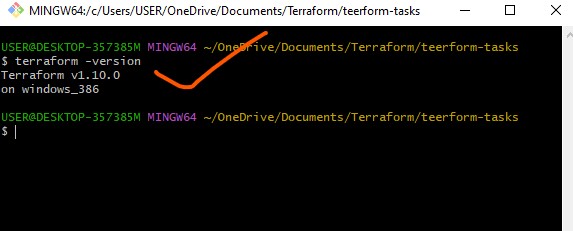
**1) Install Terraform on your PC**

Installing Terraform on windows

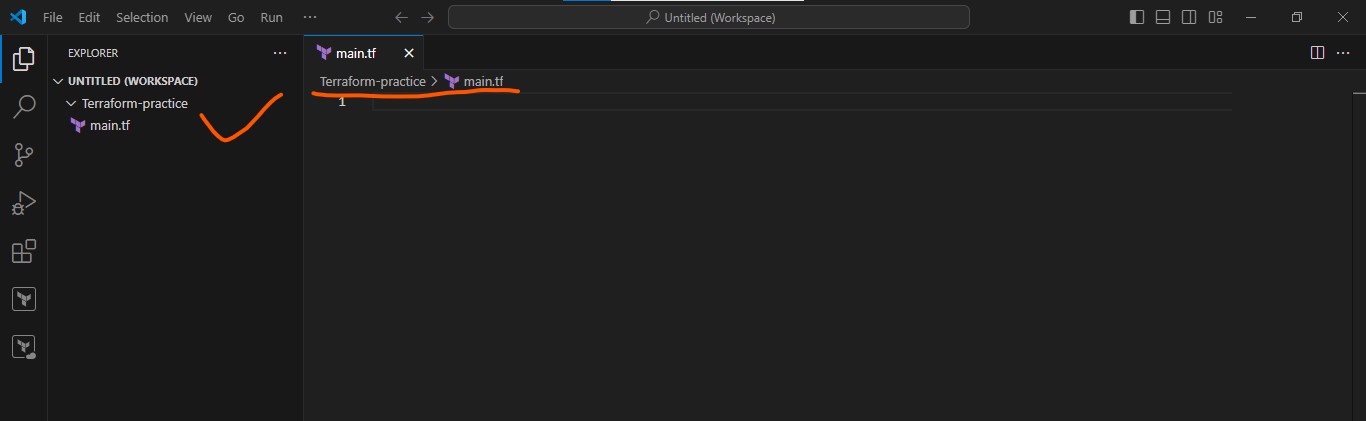
Extract the Terraform to ‘Local disk C’. Copy the path of Terraform in local disk C and paste the path in environmental variables in local machine and save the path.



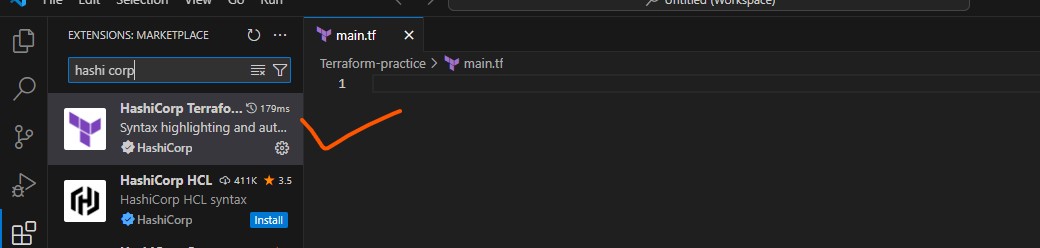
Check the version of the Terraform using git bash on local machine



Install vs code in local machine

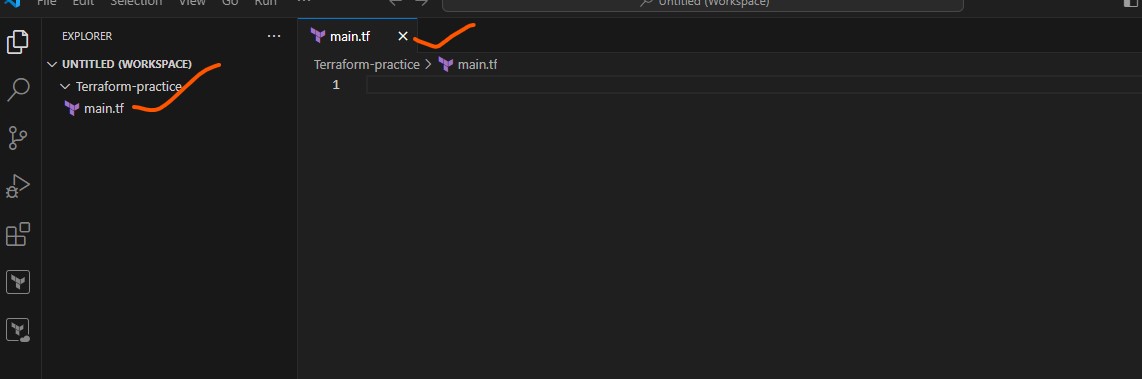


Install the Terraform extension in Vs code by using hash corp in Extensions in VS code



Open new folder in vs code where you work is deployed mention it as the Terraform-folder in local machine and save.

Create a main.tf folder .tf is the extension for the Terraform in visual studio code

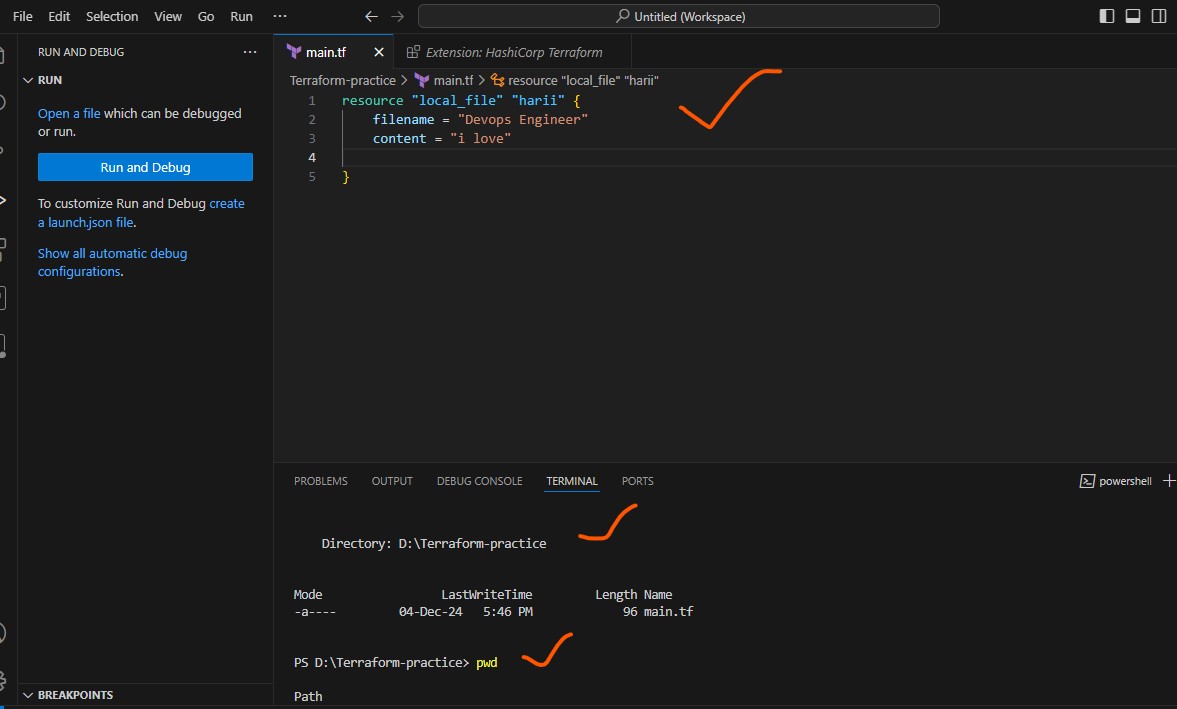


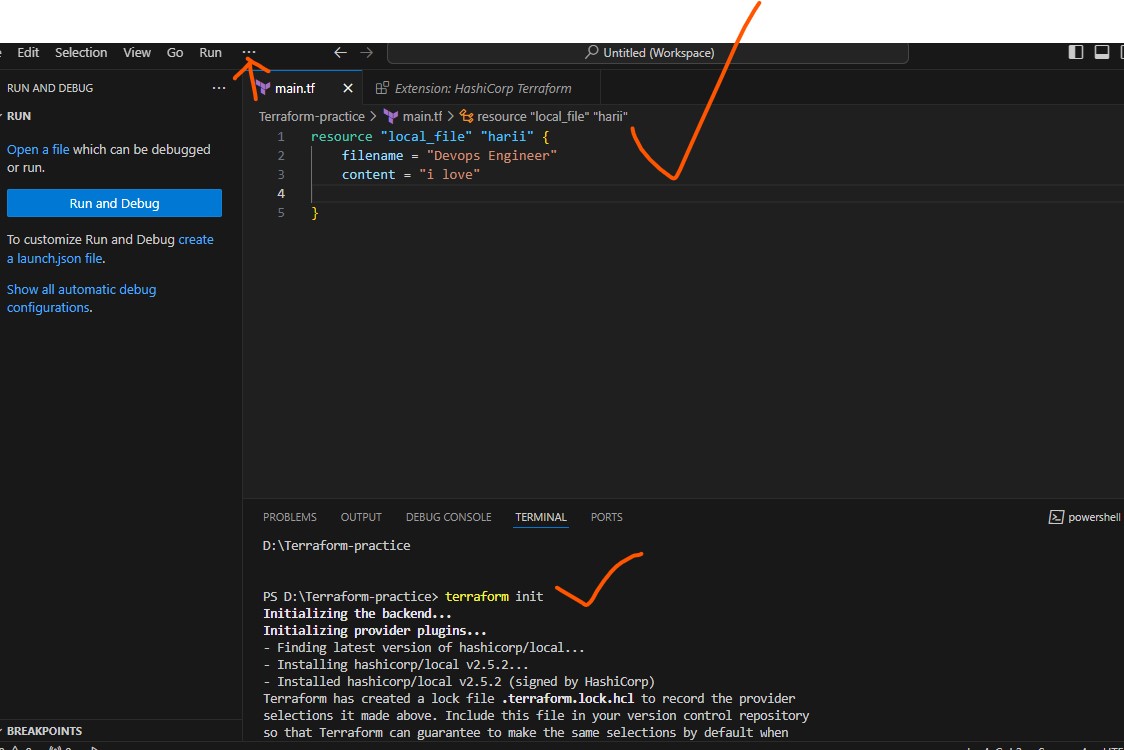
Write a piece of code in *main.tf* extension folder for the Terraform

**2) Execute all the templates shown in video.**

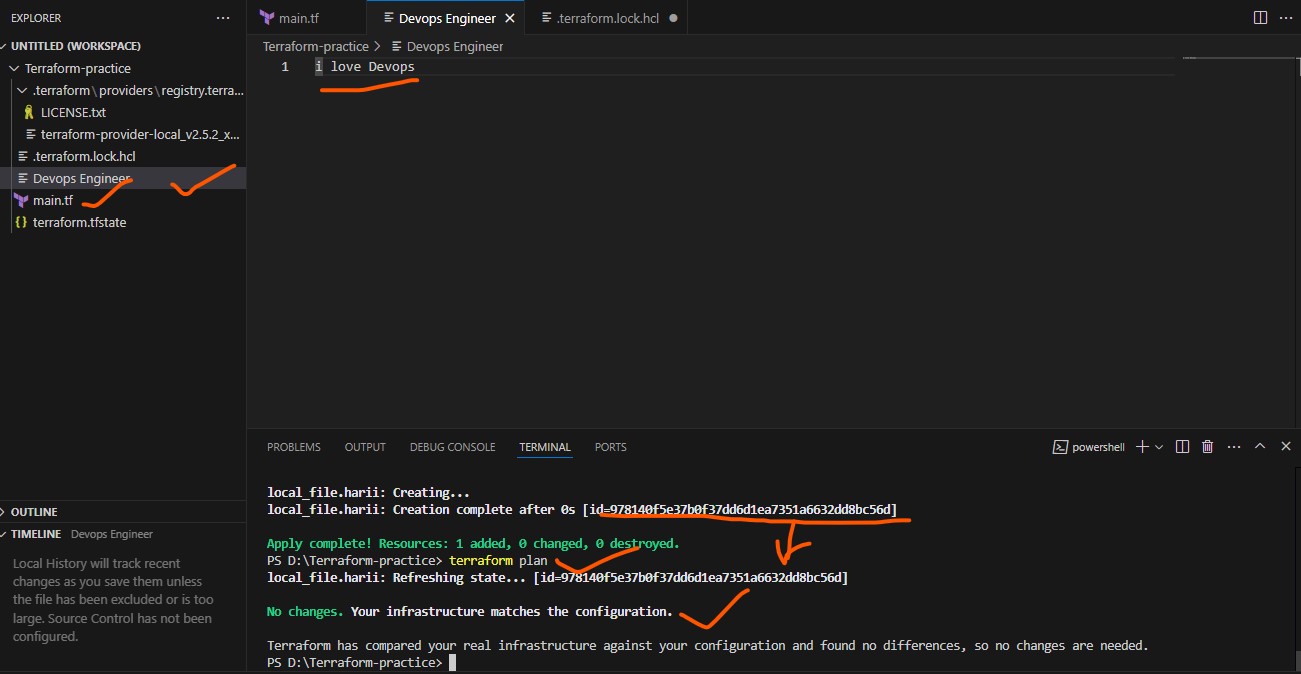
After create new terminal for the main.tf

Initialize the Terraform init in the terminal check the path of the file



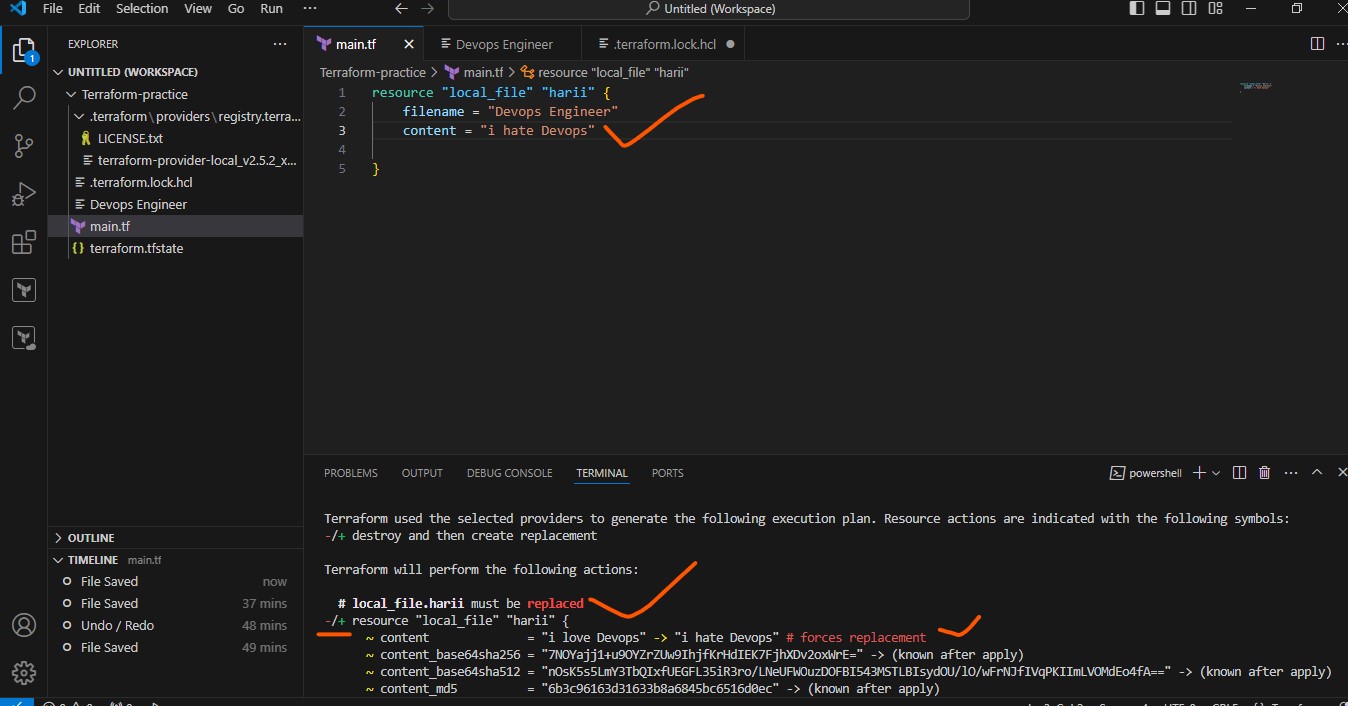


In the terminal type as Terraform plan 🡪 to preview the changes in the code

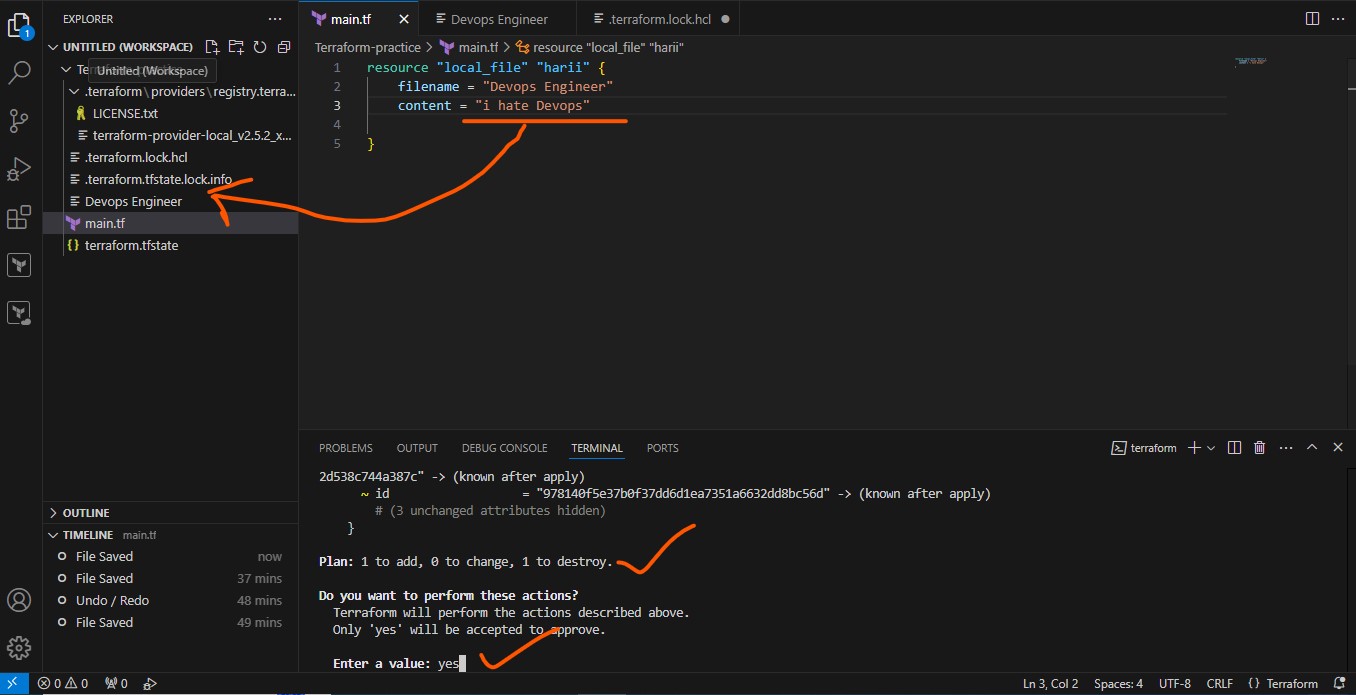
After type Terraform apply 🡪 to execute the code 🡪 automatically it creates a id for the local\_file  


The code is replaced and save the content 🡪 Terraform plan to review changes

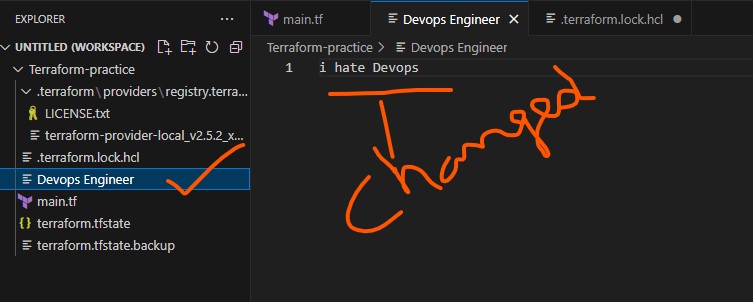
In Terraform plan we can see that the ‘~’ tilt symbol is for the changes has been reviewed in the code and after that Terraform apply to make the changes in the code



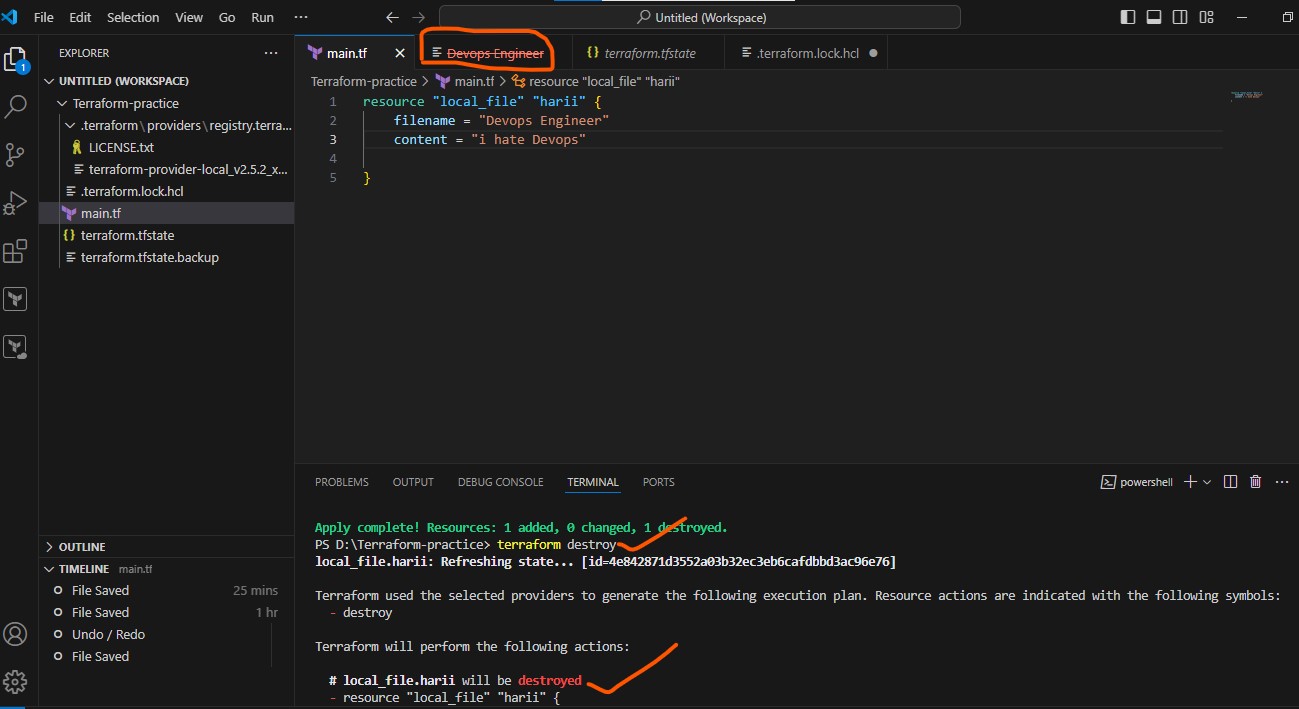
After executing Terraform apply command in the terminal 🡪 we can see the details as plan: 1 to add, 0 to change, 1: destroy and type 🡪 yes in the command line to make changes.



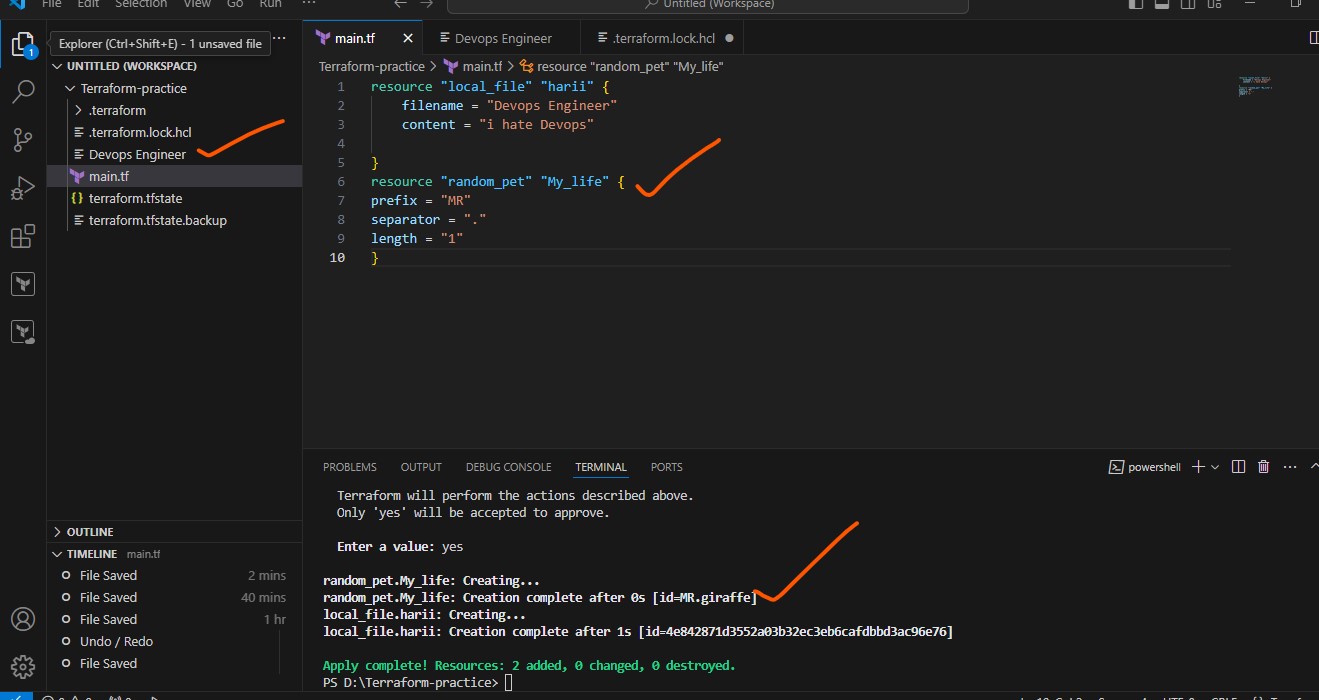
And the changes that happen in the code will be seen in Devops Engineer it’s above the main.tf – the main file



Terraform destroy is used to delete the files in the code

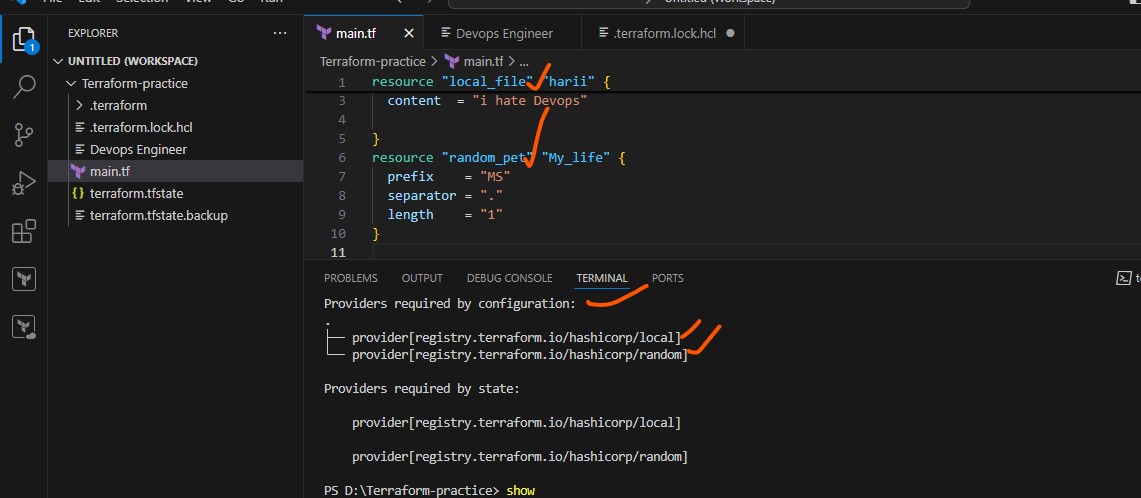


Give the resource as random\_pet 🡪 this gives the id as a MR.giraffe

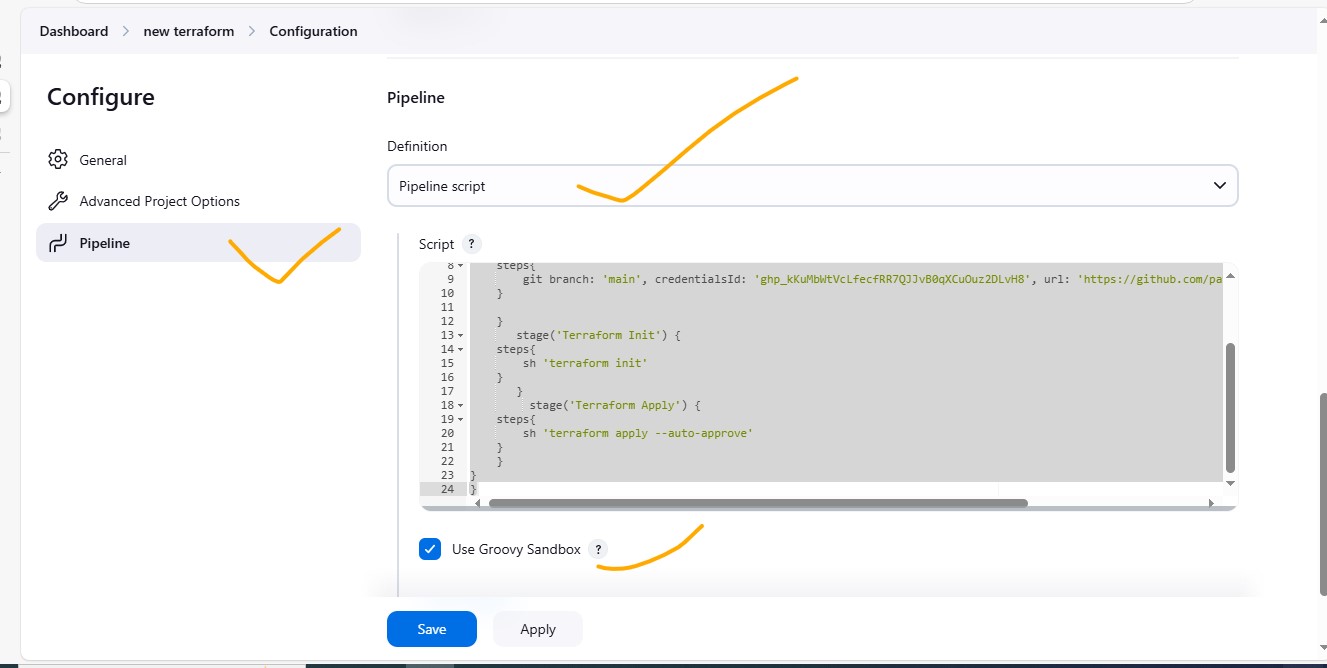
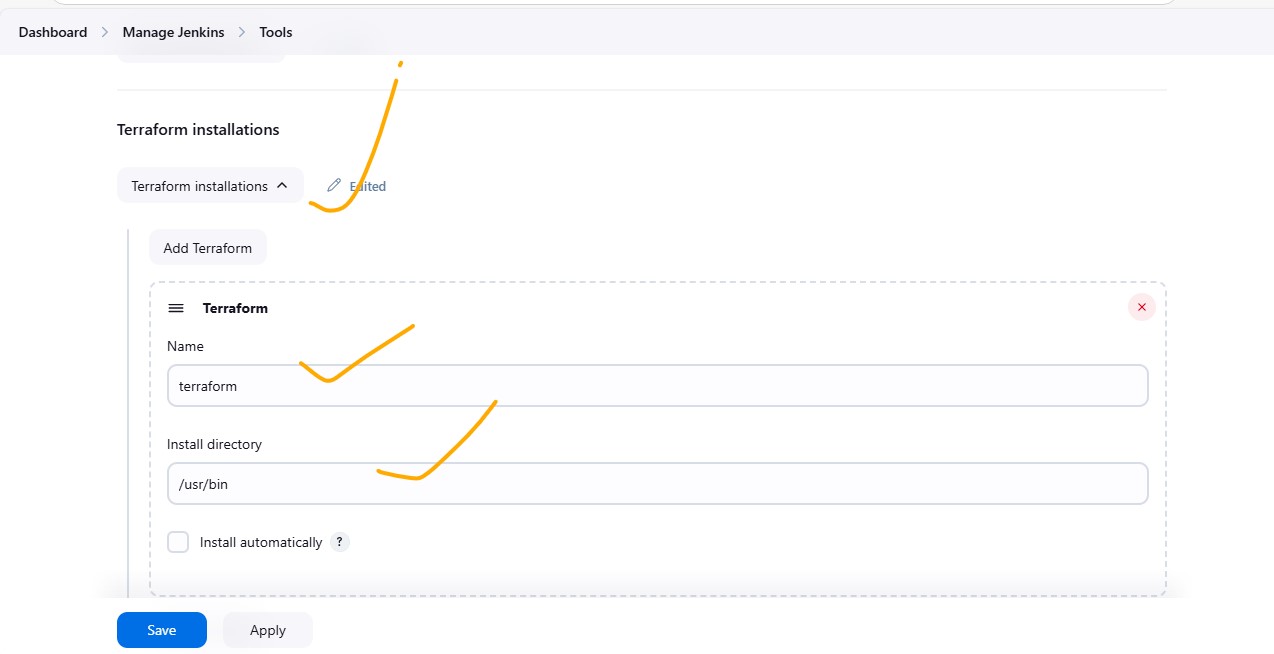
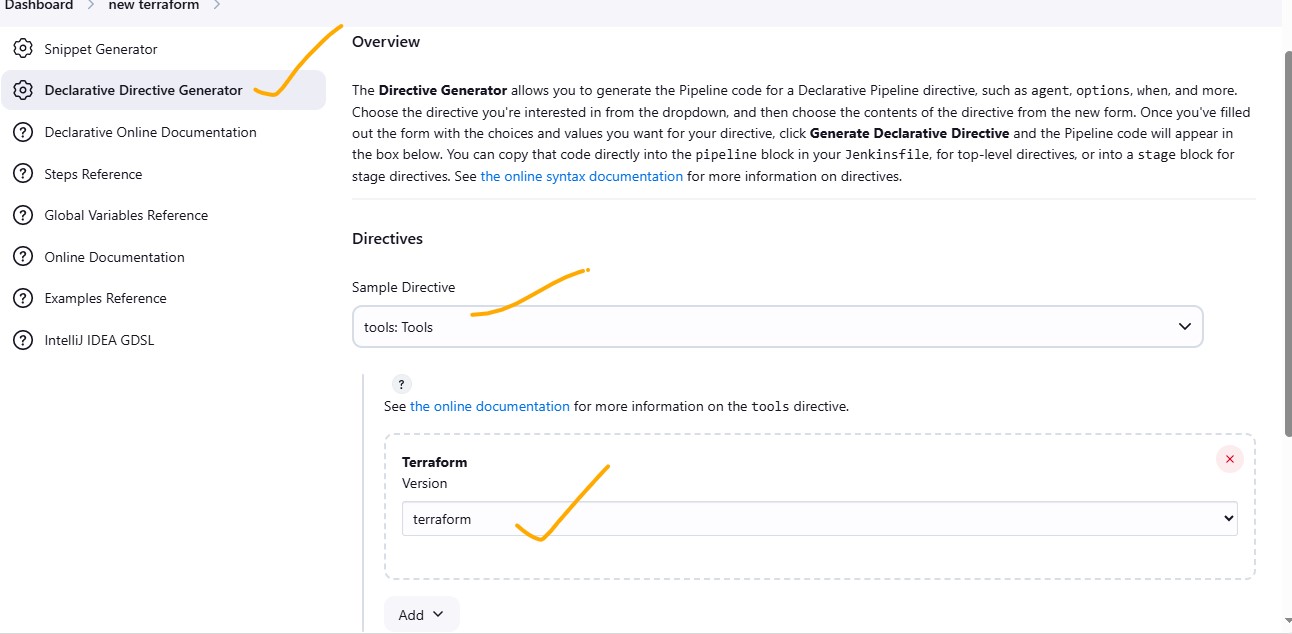
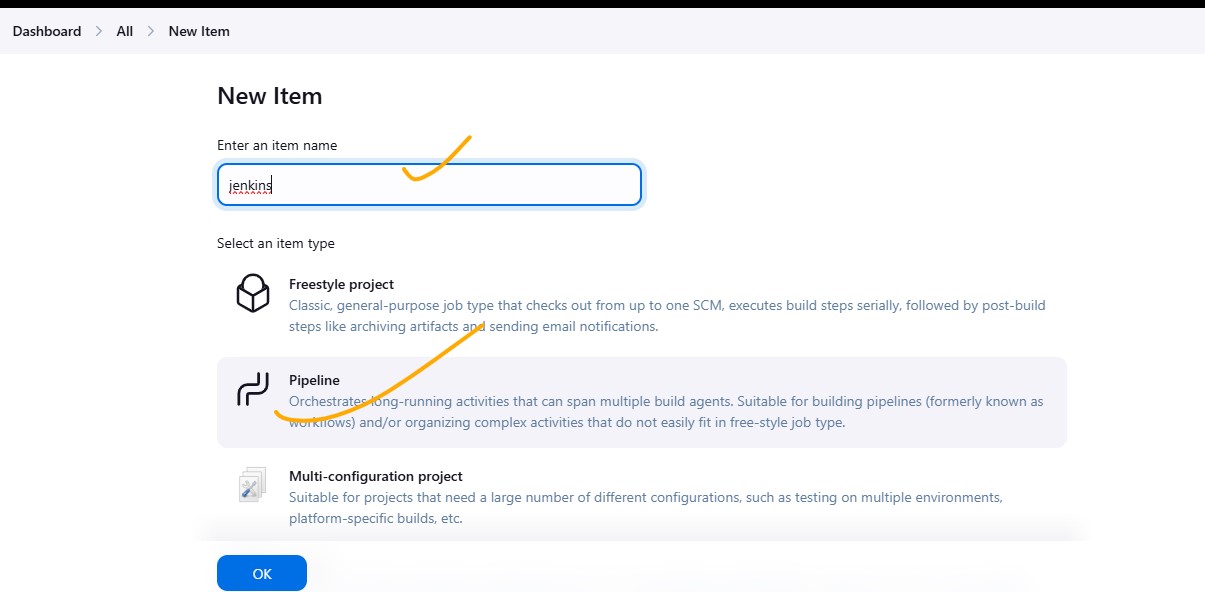
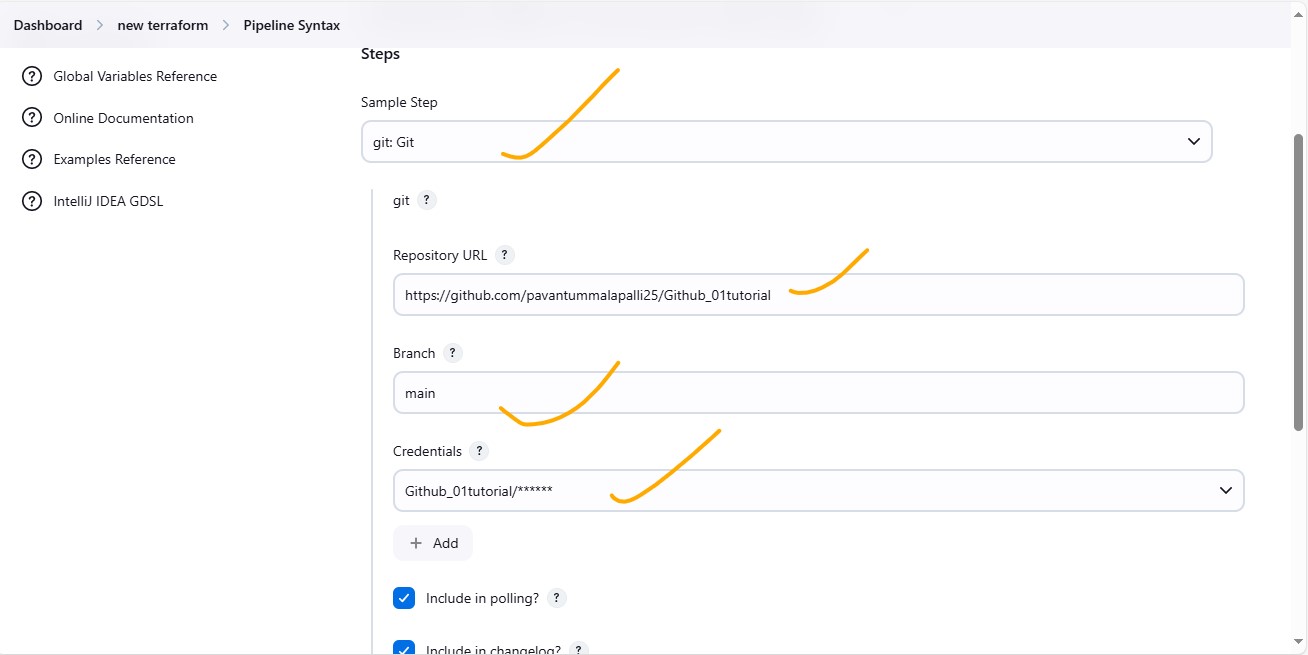
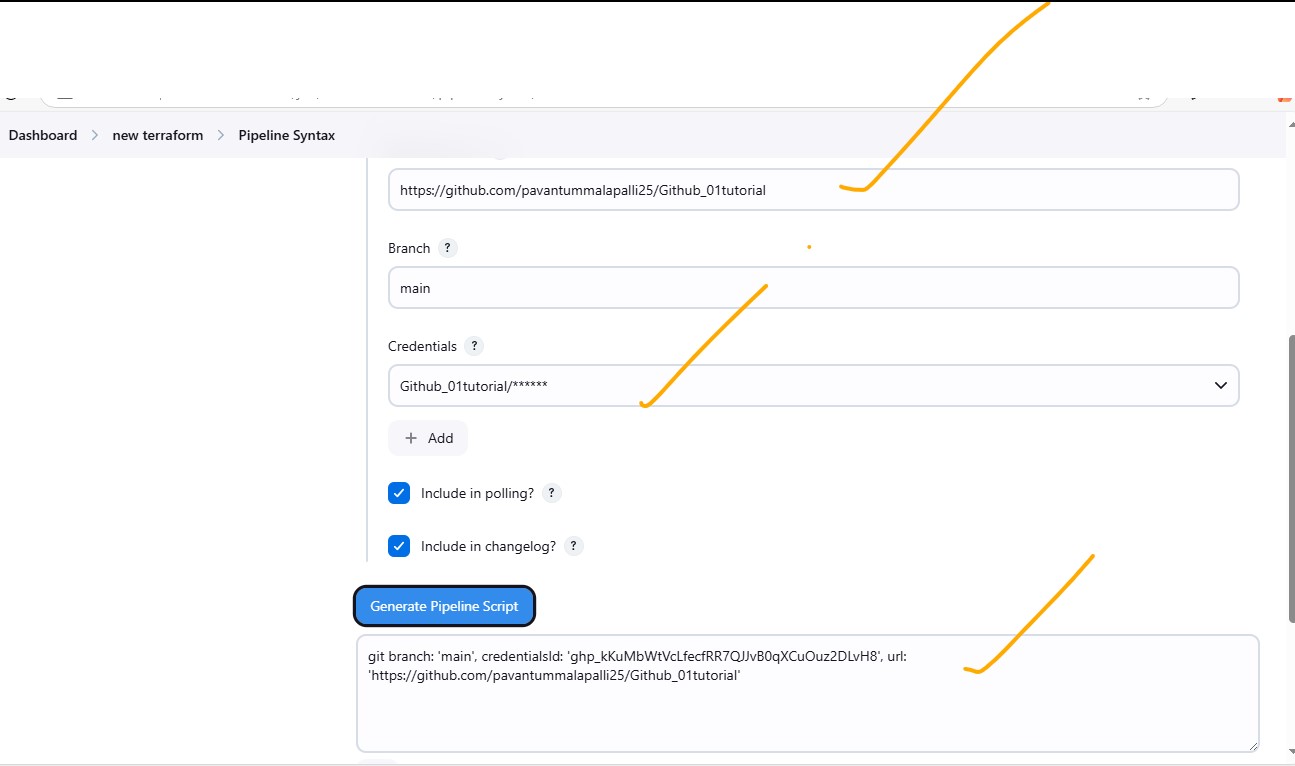


3) Note down below points, Terraform Init Terraform Plan Terraform Apply Terraform Provider.

🡪*Terraform providers* is used to print the tree of the providers used in the configuration



4) Integrate a sample Terraform template in jenkins.



6) Execute all the templates shown in video

**Terraform life cycle**

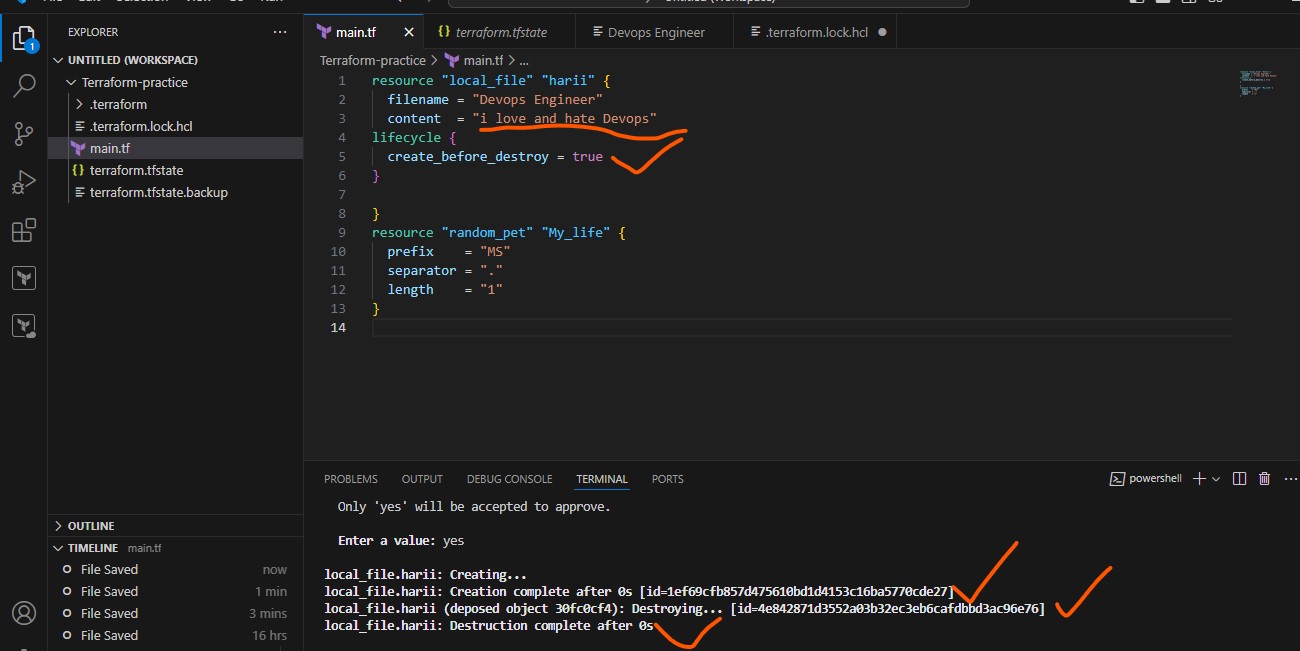
1. **Create\_before\_destroy = true**

lifecycle {

create\_before\_destroy = true

}

This life cycle first create a new version of the local file and eventually it will automatically deleting the older version of the file

.

1. **Prevent\_destroy**:-

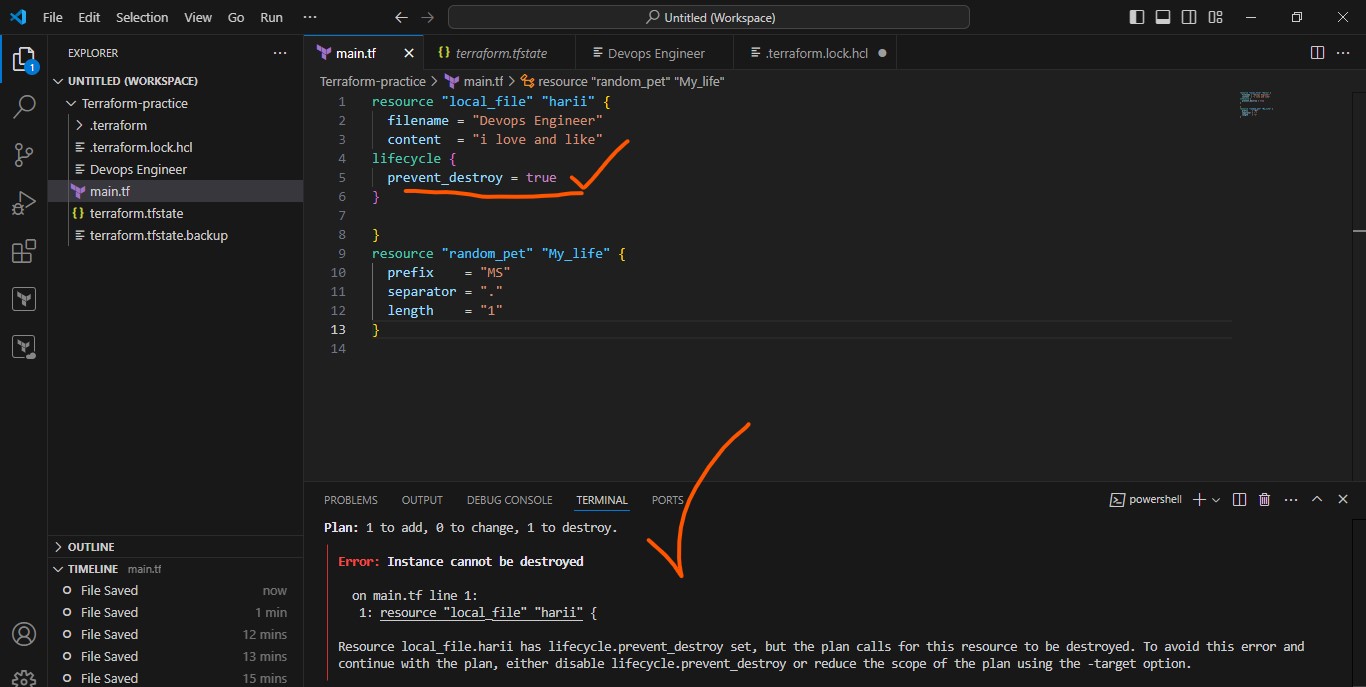
lifecycle {

Prevent\_destroy = true

}

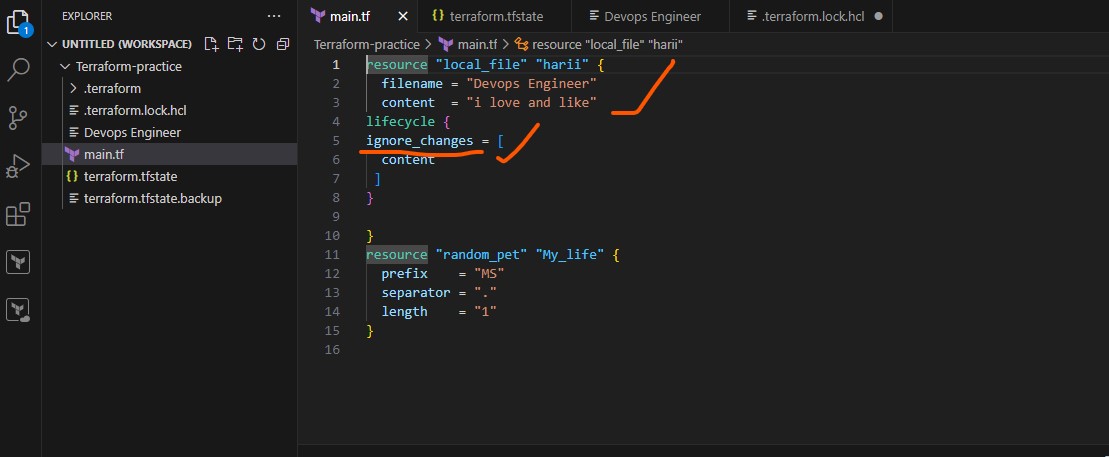
Prevents the file to be destroyed when someone get deleting the file.

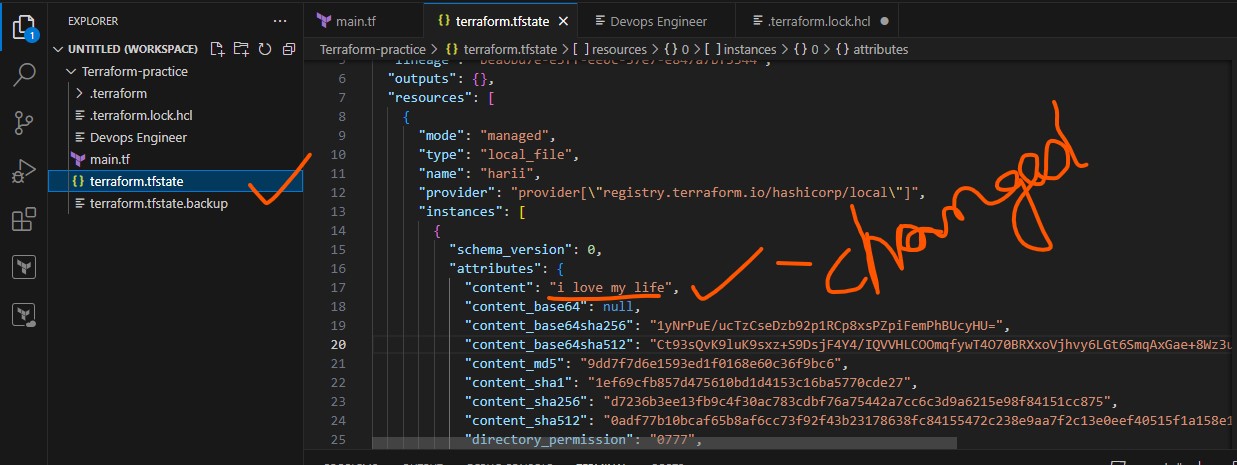
Someone is destroying the file and automatically it prevents, such cases it has been used.



**3. Ignore\_changes**

It will ignore the content of the file that you changed in the state file. It only execute the main.tf file

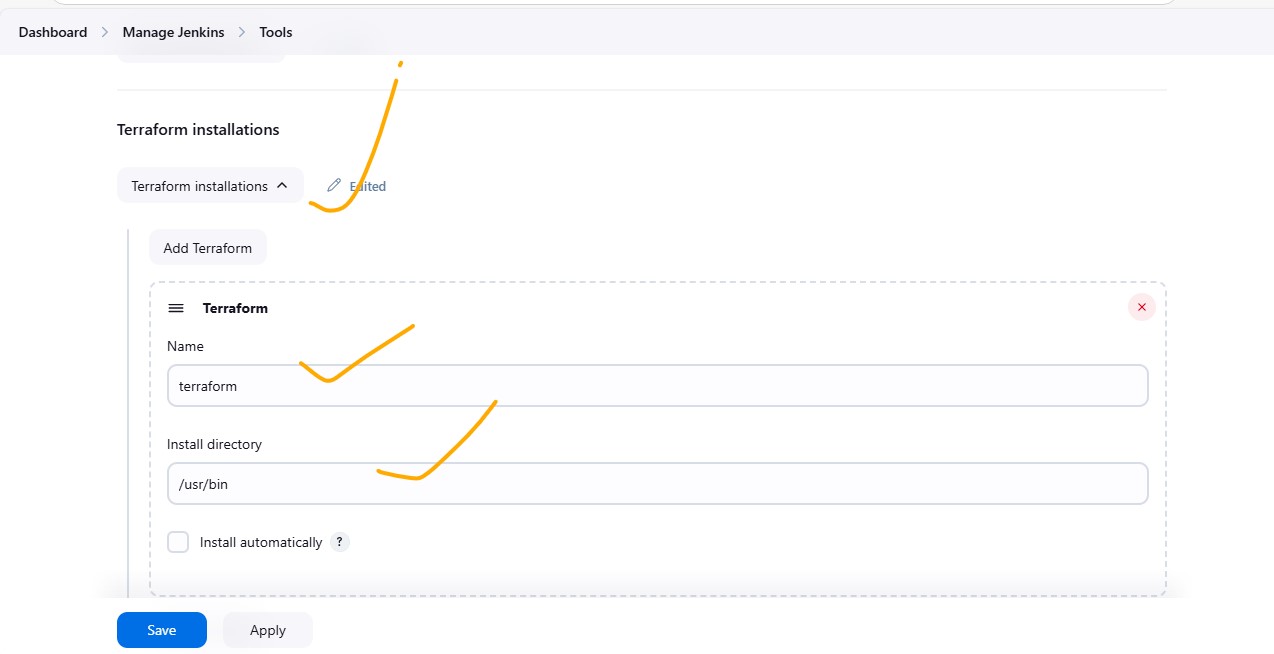
****

****

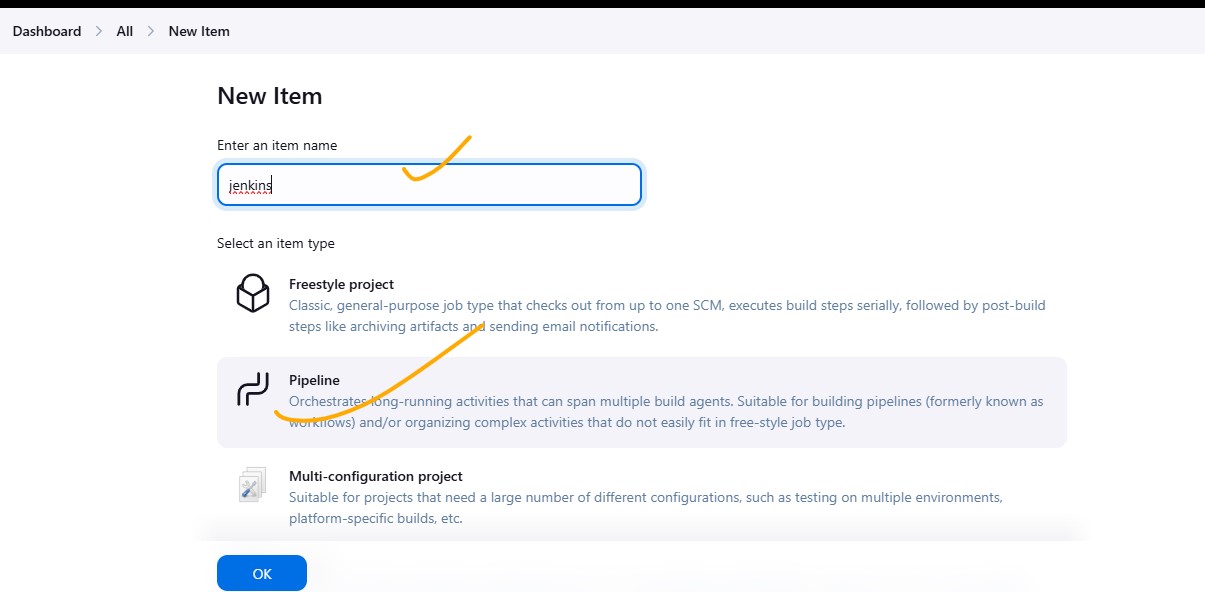
7) Integrate terraform in jenkins using Terraform plugin.

Install Terraform plugin in plugins

Go to manage jenkins and select tools. Below select Terraform installations and add Terraform and install directory as */usr/bin* and save. Configure the Terraform and select it.



Create a New item in jenkins and select the pipe line and ok



Below the configuration, select the pipeline script

Pipeline script {

agent any {

}

}

And give the code by using pipeline syntax as given below and paste the code in script.

In pipeline syntax select the declarative directive generator and sample directive as *tools: Tools,* below select Terraform and *generate declarative directive.*

Paste the code in script below the first one.

pipeline {

agent any

tools {

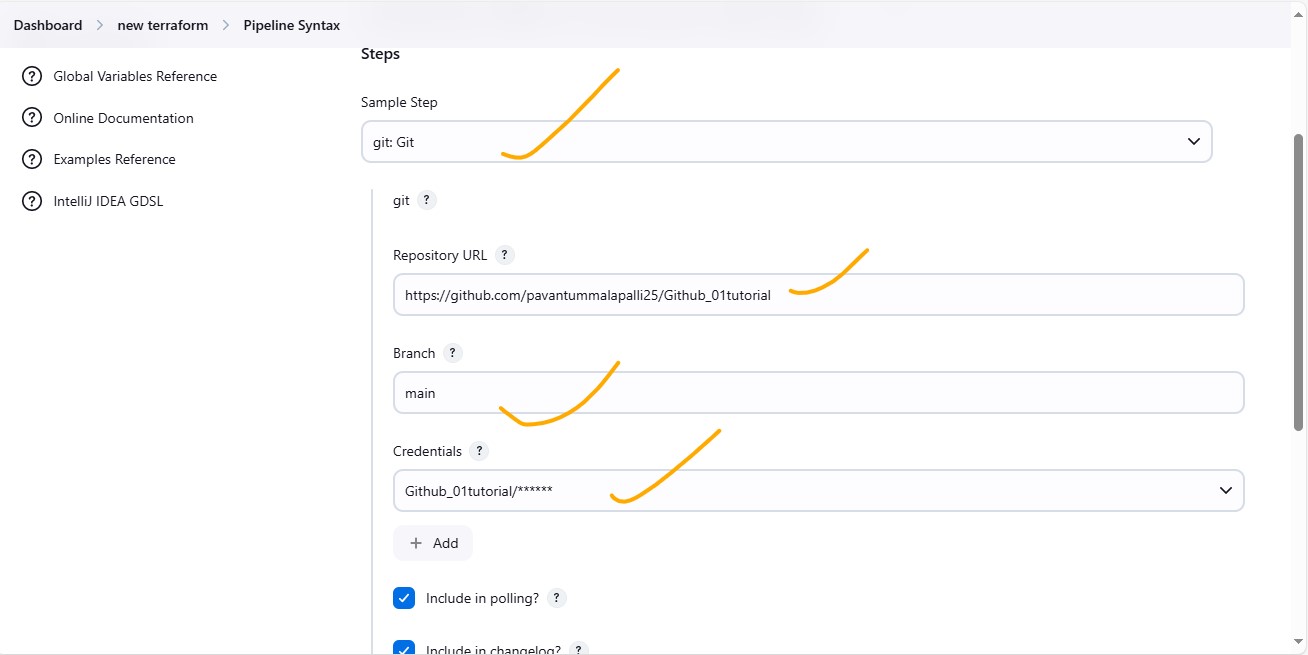
terraform 'terraform'

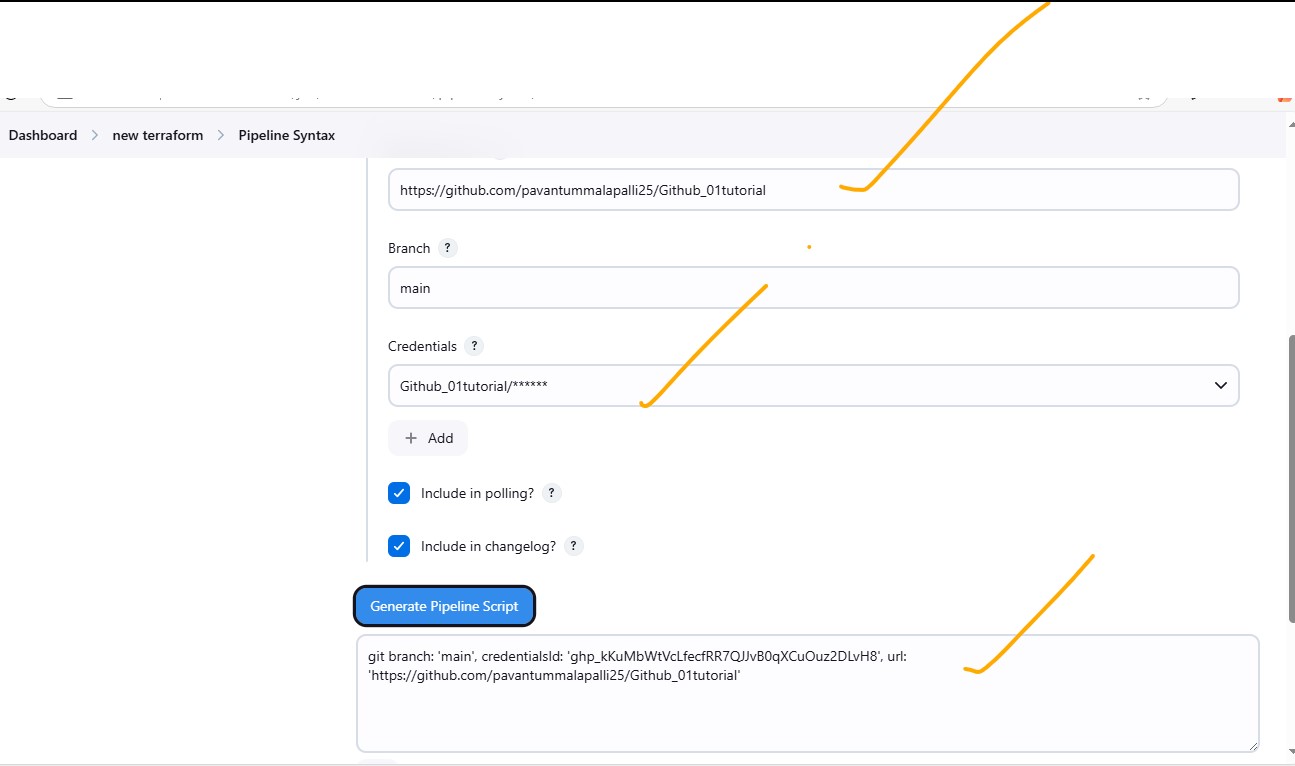
}

}

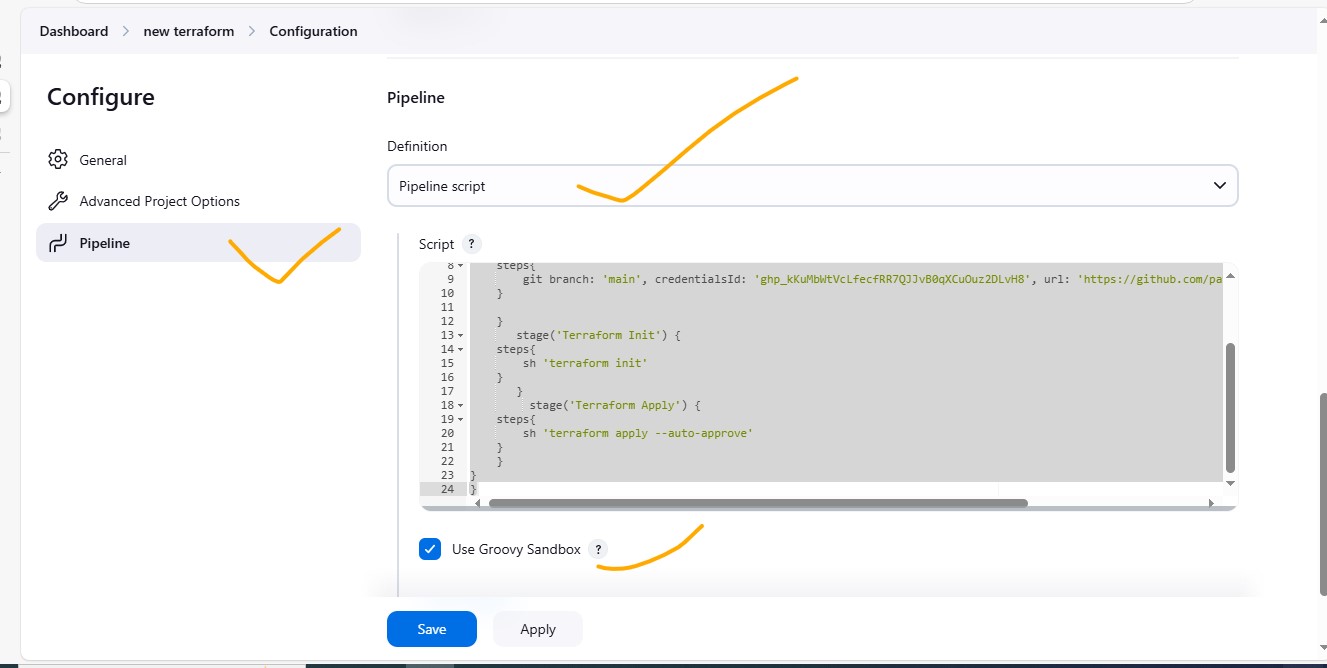
After select git: Git and give the repository url and branch name that exists. Select credentials as Jenkins and the username as git repo name and password as token id that generated in github and save.

Generate pipeline script, copy the code and paste the code to script.





After the git hub, Terraform commands will be given in the script one by one as shown in figure.



And save the configuration

Go to git hub create a new file as main.tf 🡪 and Terraform basic script should be given in the file and save it.

Go to AWS, create a new role as *jenkins-terraform role* and give the role as ec2 administrator and save the role.

Modify the existing role with new role as *jenkins-terraform role.*

Select Build now automatically it will generate the pipeline.

Pipeline overview to get better understanding

