**1) Watch terraform-03 video.**

**2) Execute the script shown in video**.

* **We are using Variable.tf file**

In **main.tf** we need to add resources and variables

resource "local\_file" "my-pets" {

  filename = var.filename

  content = var.content

}

resource "random\_pet" "petname" {

  prefix = var.prefix

  separator = "."

  length = "1"

}

In variable**.tf** we need to add resources

variable "filename" {

default = "pets.txt"

type = string

description = "file name is pet"

}

variable "content" {

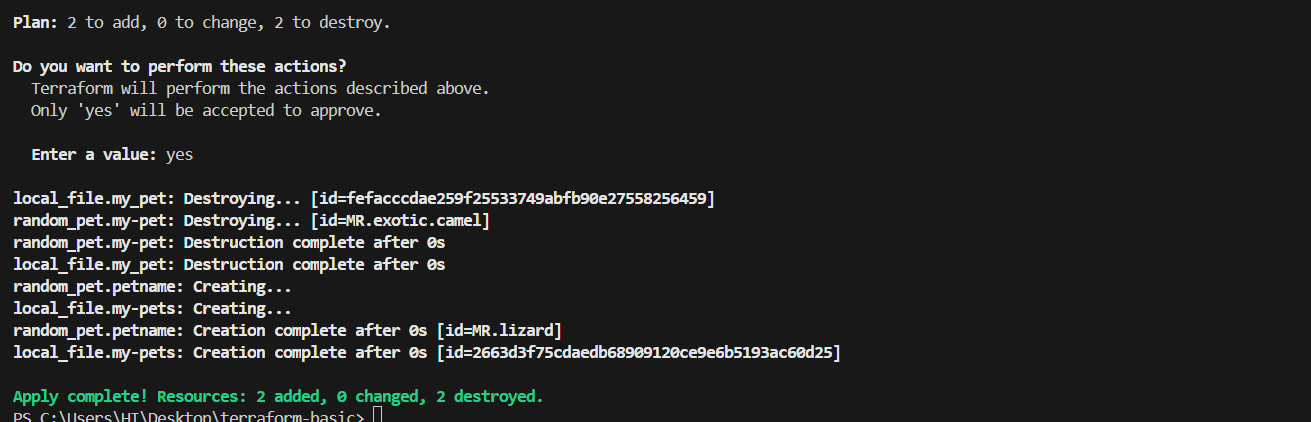
default = "we love nature"

}

variable "prefix" {

default = "MR"

}



* **We are using interactive mode**

\* In variable.tf file not given any input

resource "local\_file" "my-pets" {

  filename = var.filename

  content = var.content

}

resource "random\_pet" "petname" {

  prefix = var.prefix

  separator = "."

  length = "1"

}

variable "filename" {

}

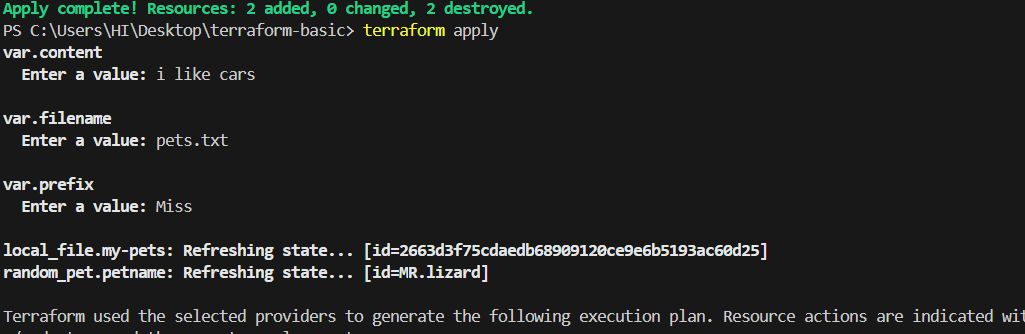
variable "content" {

}

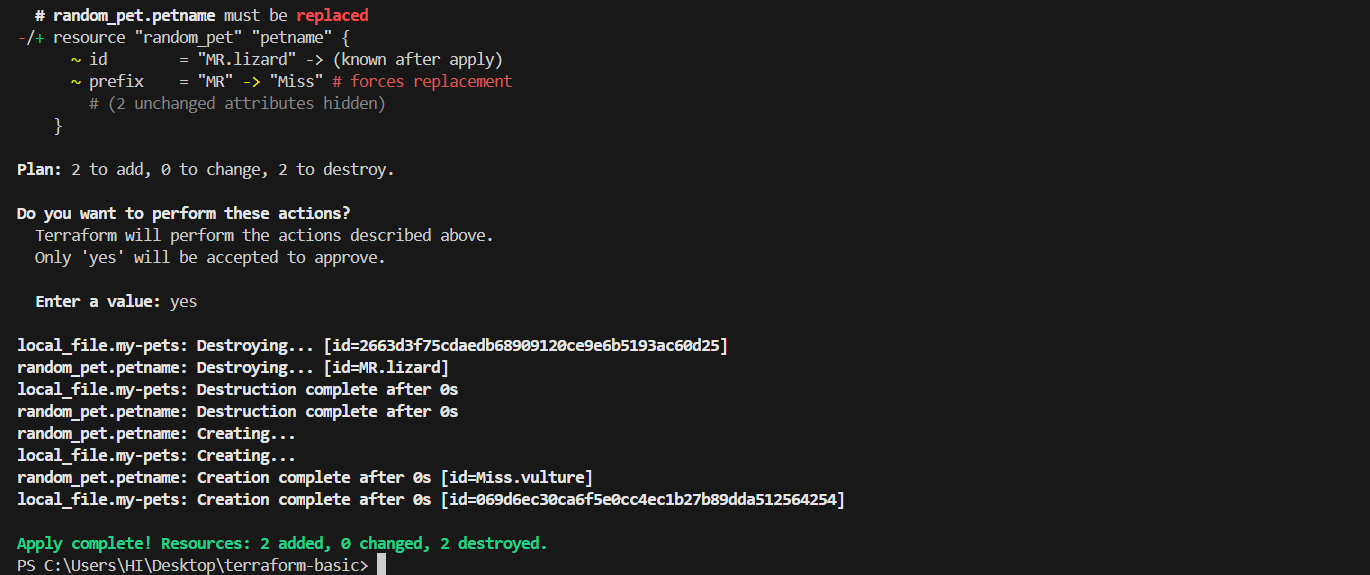
variable "prefix" {

}

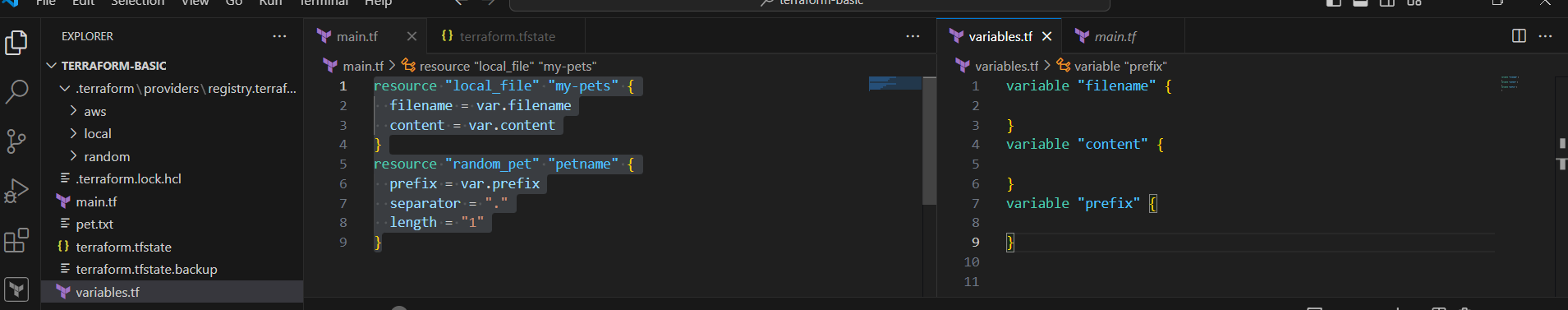
\* After **apply** to **terraform ,** we can enter manually



\* I will destroy previous infrastructure and then create new infrastructure with random name

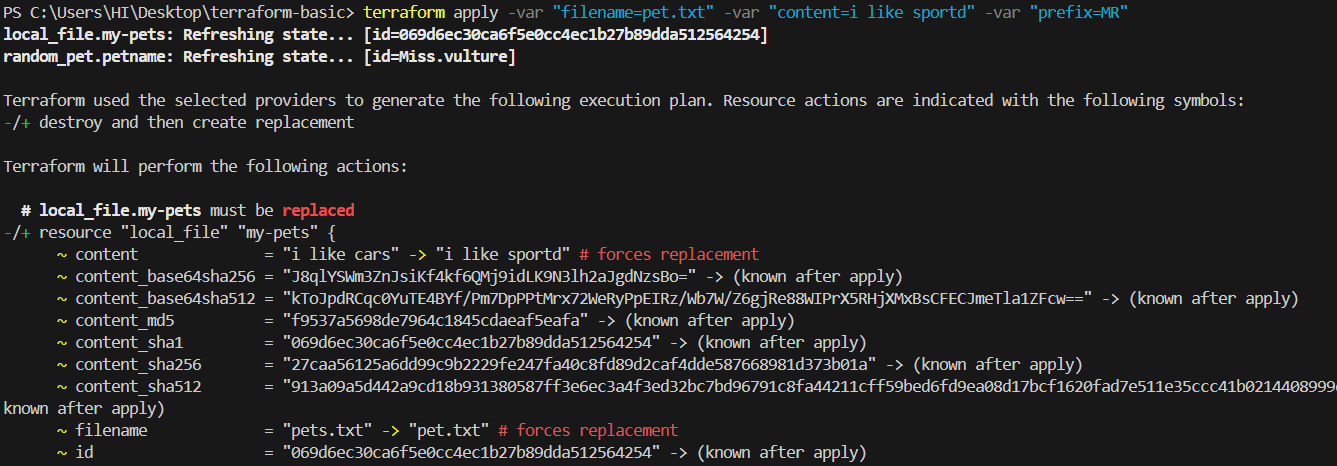


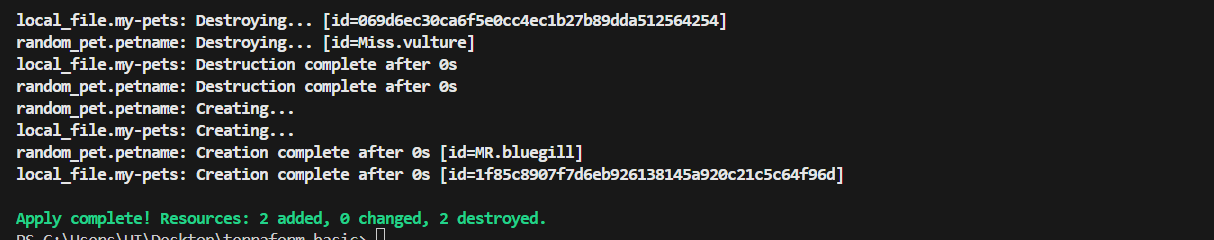
* **Here am using Command Line Flags(CLF)**



By using below command line flags to create a file with random name

terraform apply -var "filename=pet.txt" -var "content=i like sportd" -var "prefix=MR"





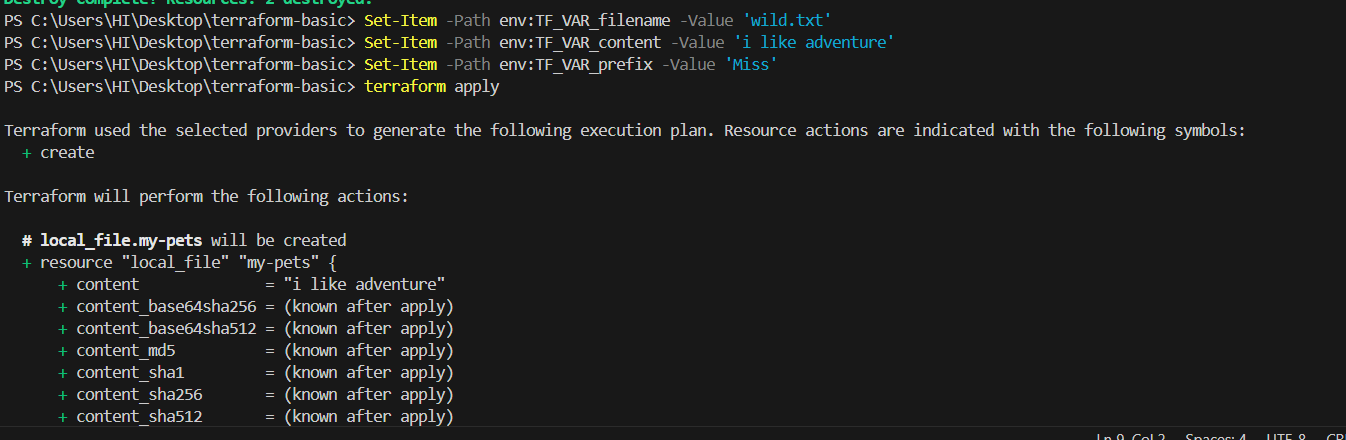
* **Now am using Environment variables for** temperary basis

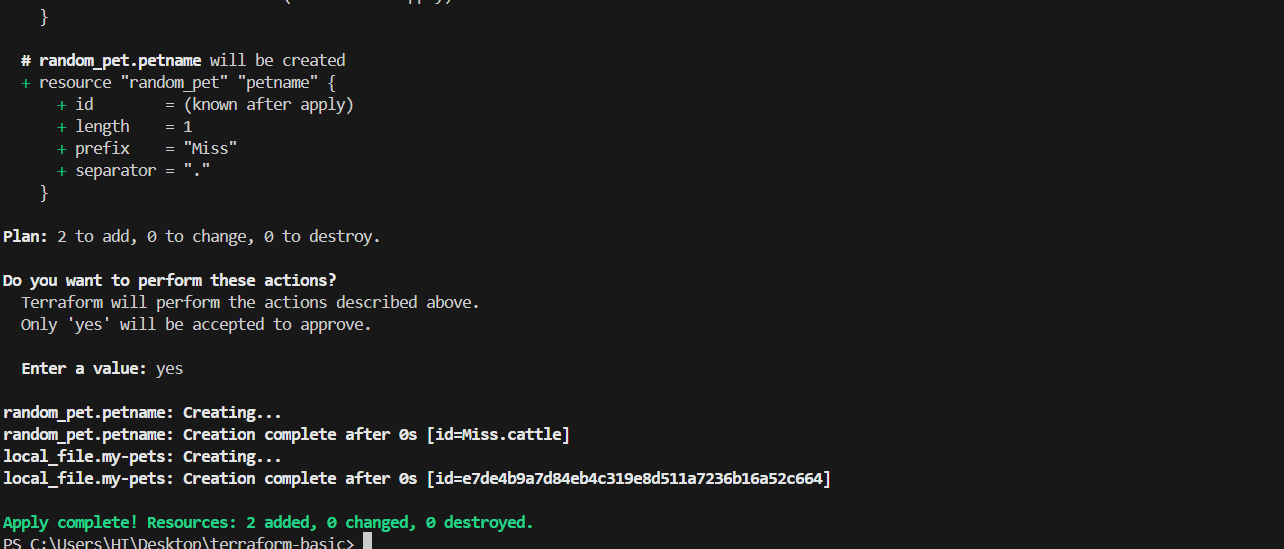
Set-Item -Path env:TF\_VAR\_filename -Value 'wild.txt'

Set-Item -Path env:TF\_VAR\_content -Value 'i like adventure'

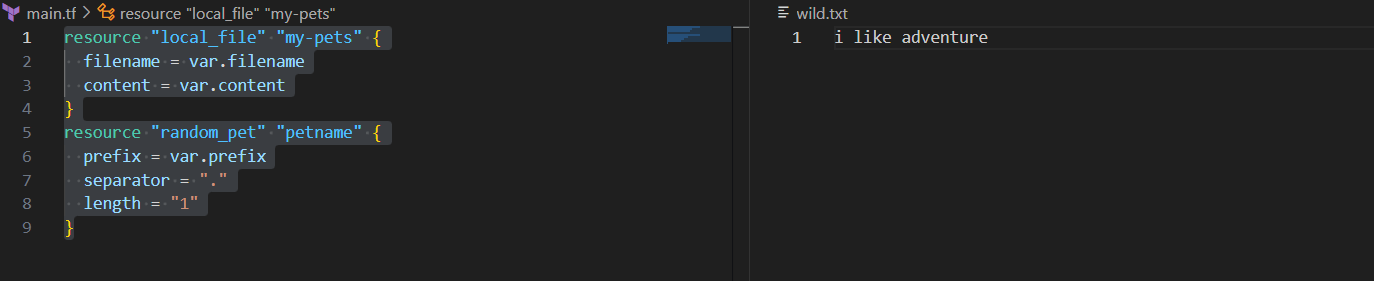
Set-Item -Path env:TF\_VAR\_prefix -Value 'Miss'

terraform apply



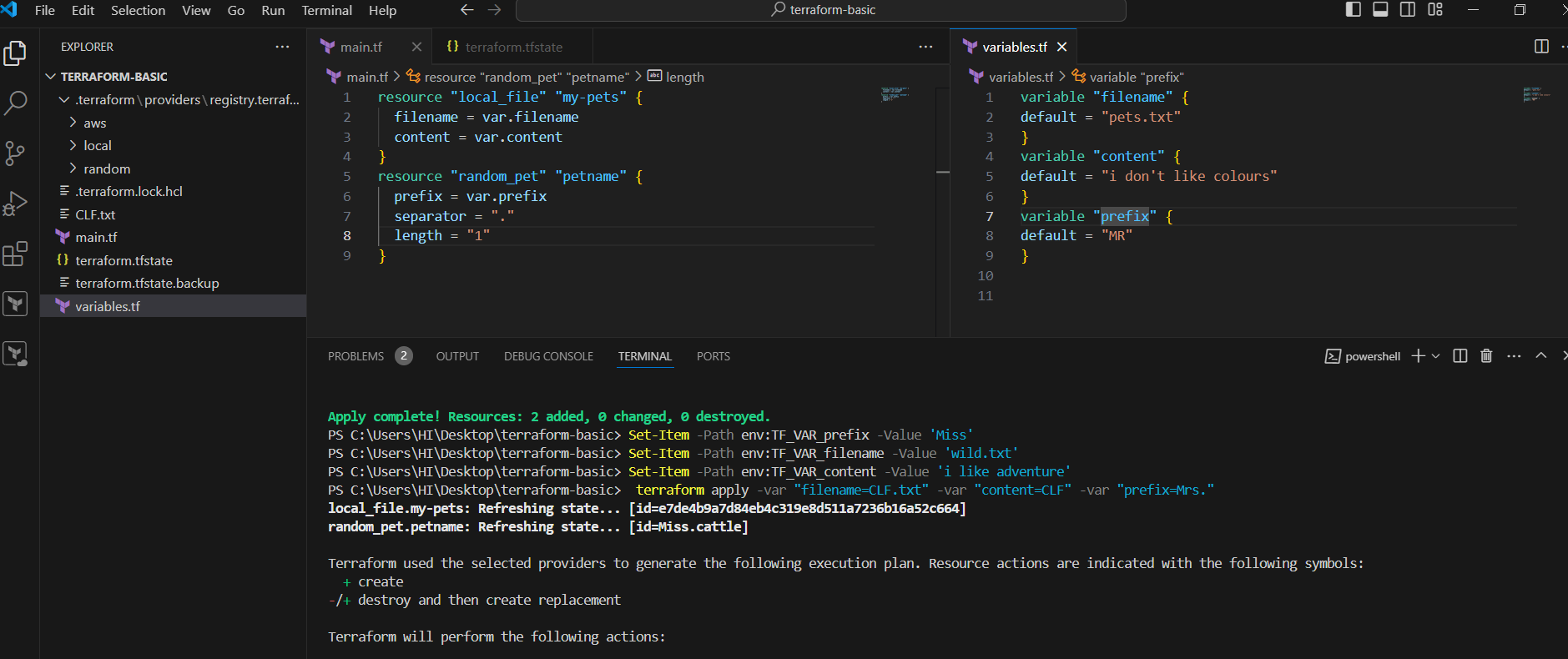


\* Now I can see the output(content)



* **Now am using Variable Definition file**

**\*** here multiple variables are applied at the same time



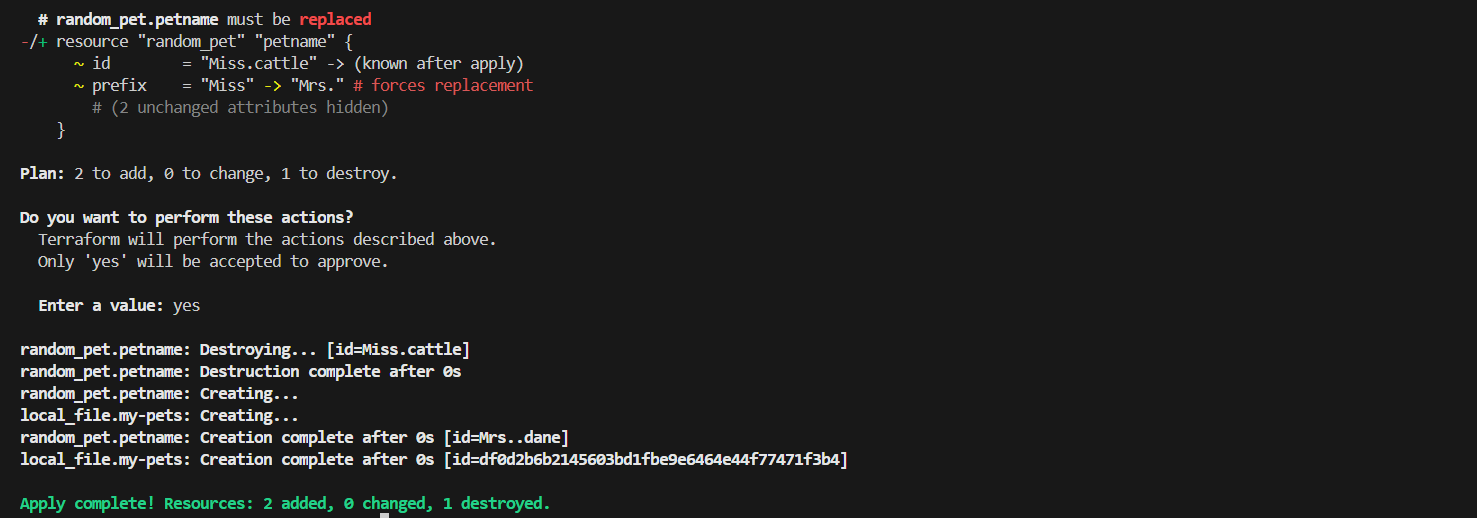
\* At this moment it will checks for **precedence order**

environment variables

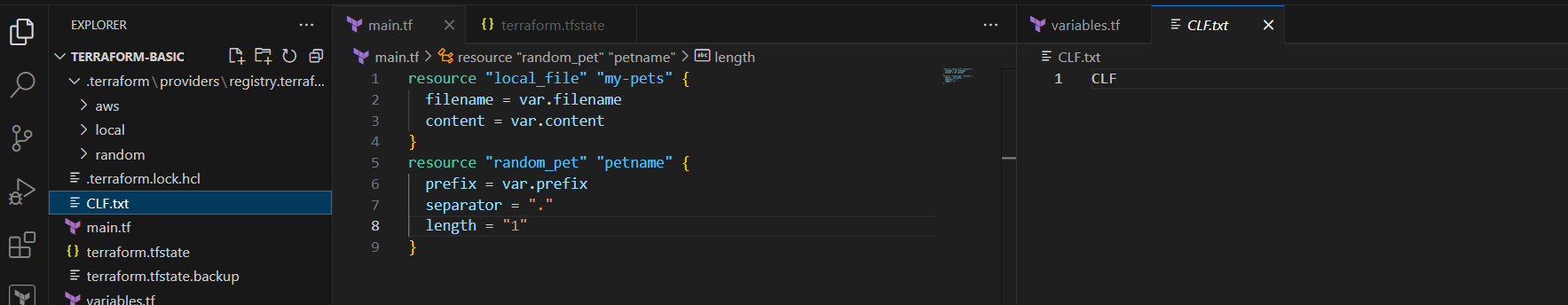
Terraform.tfvars

-var or -var-file (CLF)

By using this orders it will create the file



\*Output



* **Now am using Resource attribute reference (Implicity dependency)**

\* Here am adding the random name into content

resource "local\_file" "my-pets" {

  filename = "pets.txt"

  content = "The monkey name is ${random\_pet.petname.id}"

}

resource "random\_pet" "petname" {

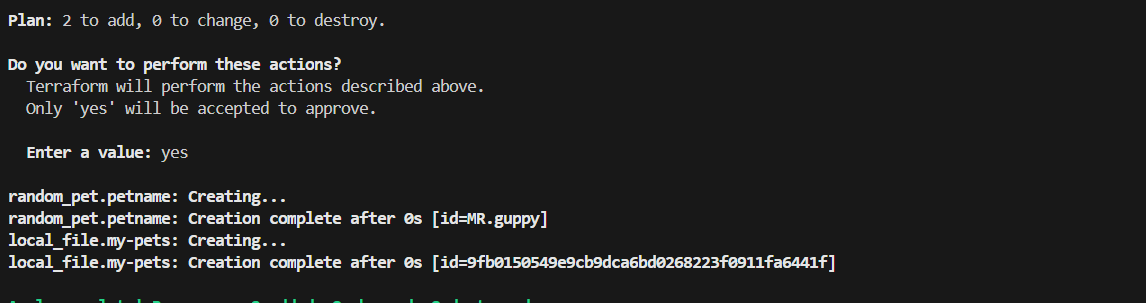
  prefix = "MR"

  separator = "."

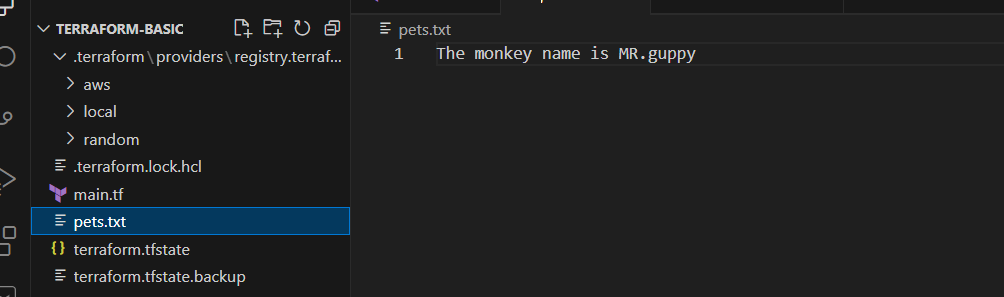
  length = "1"

}

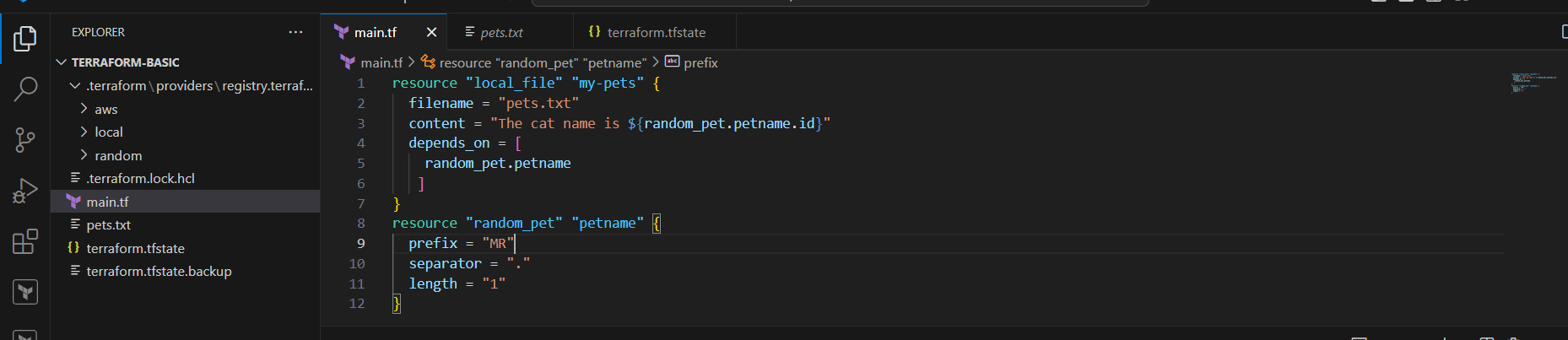
terraform apply



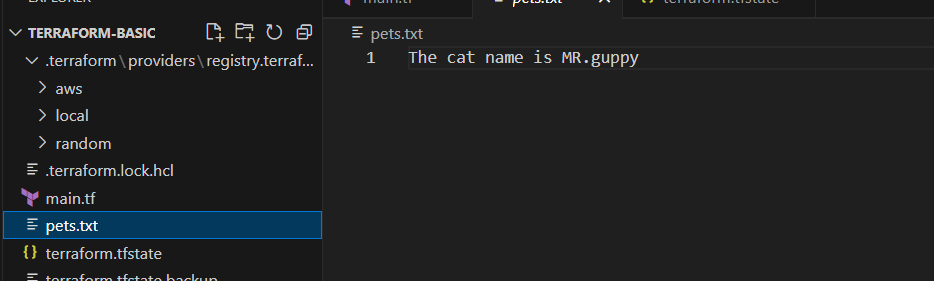
\* The random name has been added into content



\* Now am using **explicity dependency** by adding **depends on module**



\*terraform will understand to that dependency



* **Output**

resource "local\_file" "my-pets" {

  filename = "pets.txt"

  content = "The cat name is ${random\_pet.petname.id}"

  depends\_on = [

    random\_pet.petname

   ]

}

resource "random\_pet" "petname" {

  prefix = "MR"

  separator = "."

  length = "1"

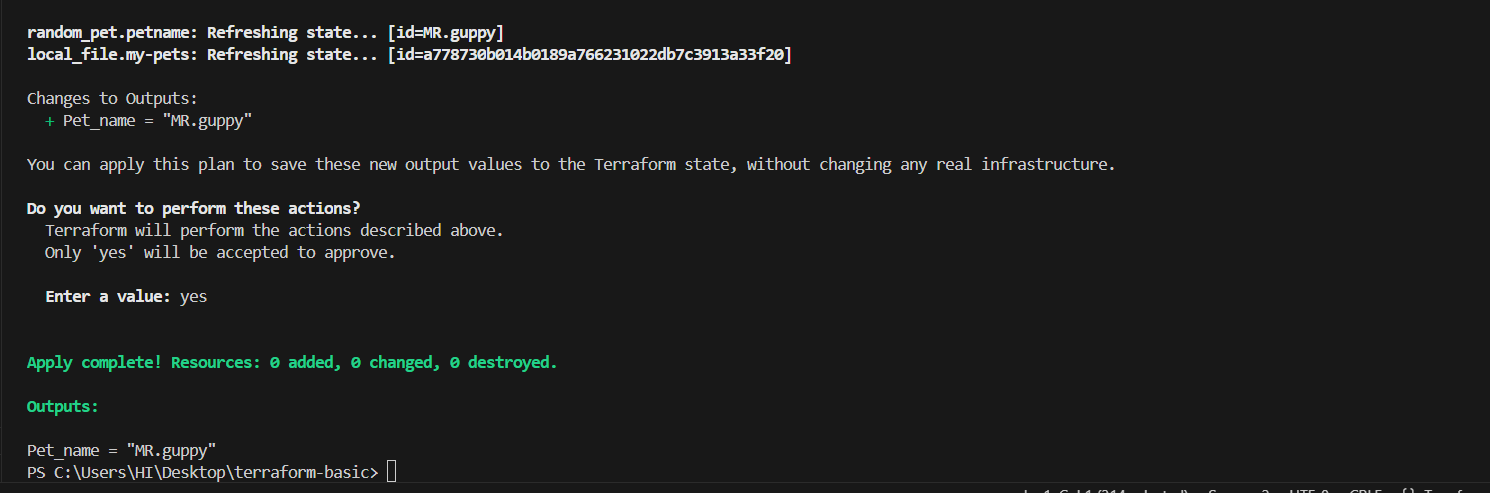
}

output "Pet\_name" {

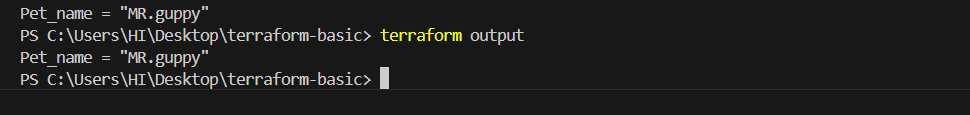
  value = random\_pet.petname.id

}

\* Now will got the output

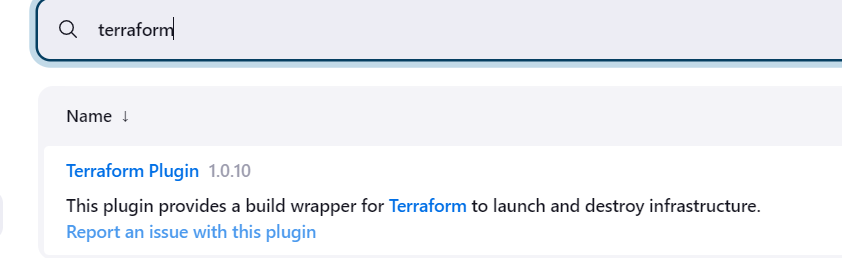


\* we can get the output by using “ **terraform output”**



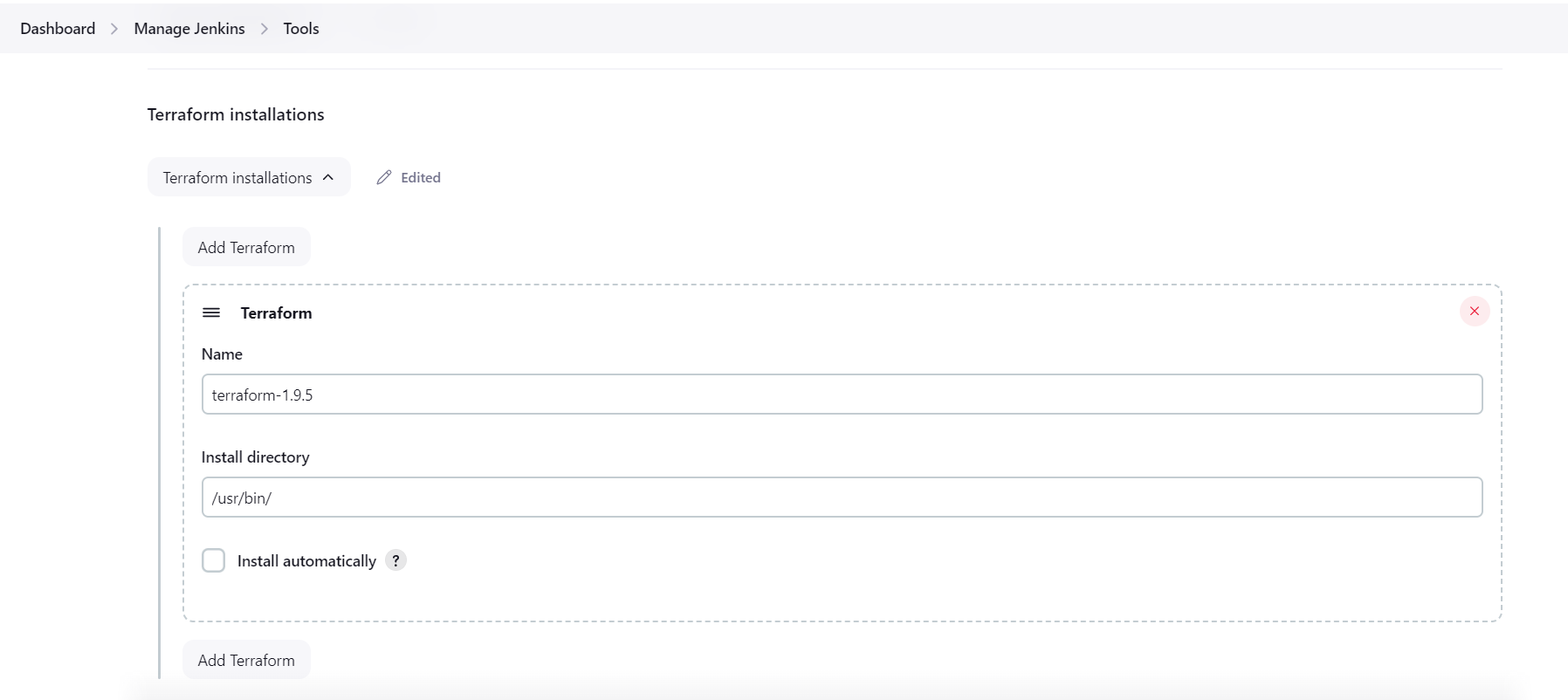
1. **Intergrate terrafrom in jenkins using Terraform plugin.**

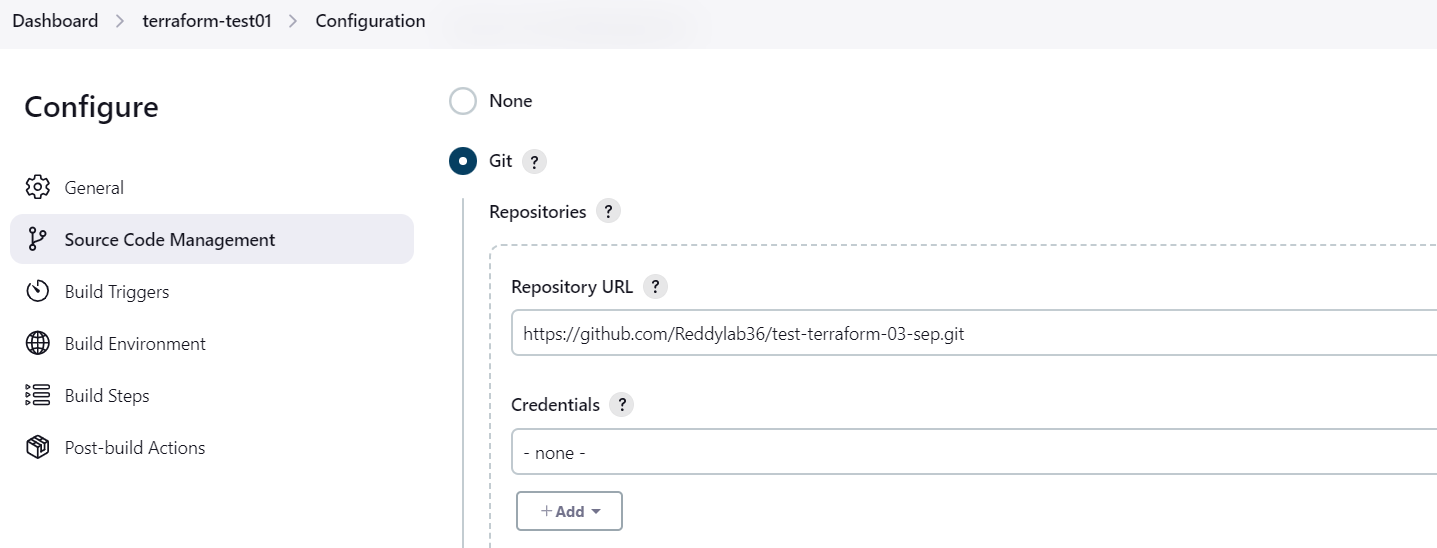
Login to Jenkins server→go to Dashboard→Go to manage jenkins→click on plugin→go to available plugin→search terraform plugin→Install

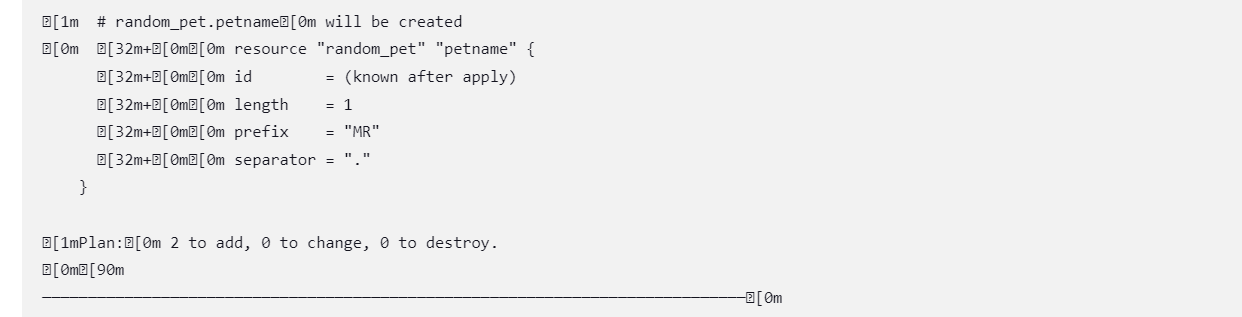


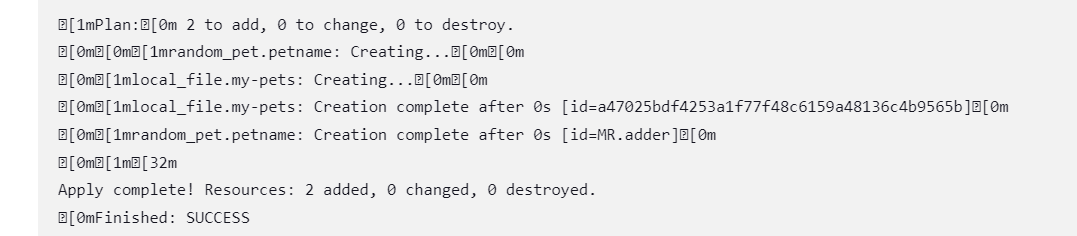
\* Go to Jenkins Dashboard→Manage Jenkins→Tools configuration → click on terraform Installation

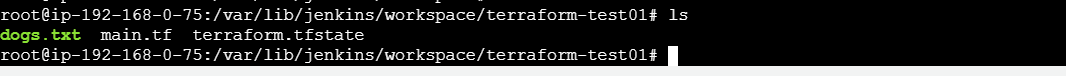
* Then select the terraform version
* Give the path for installed directory









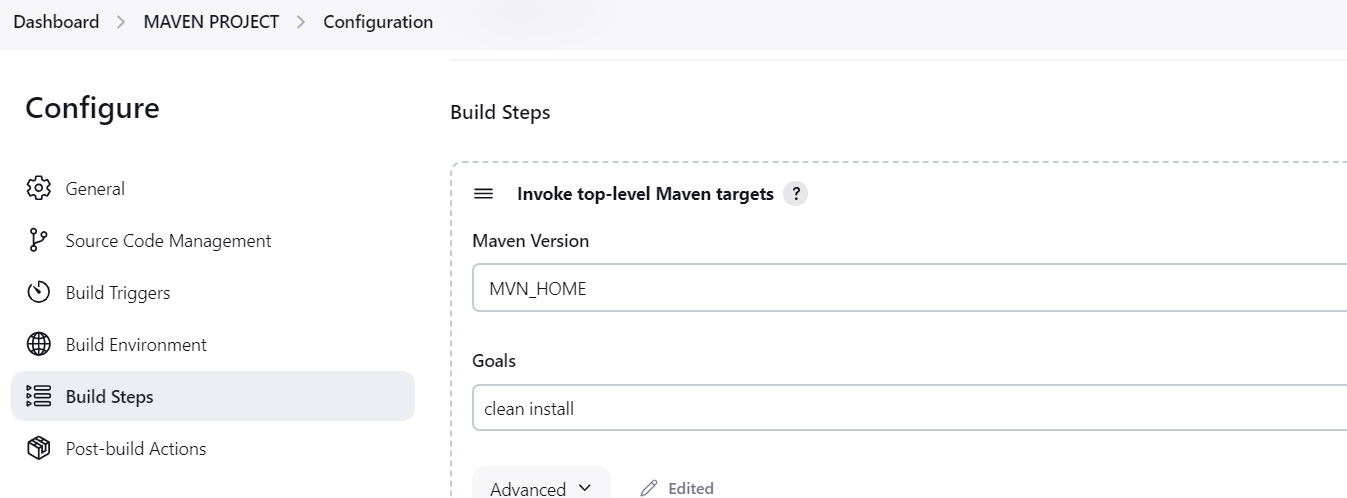


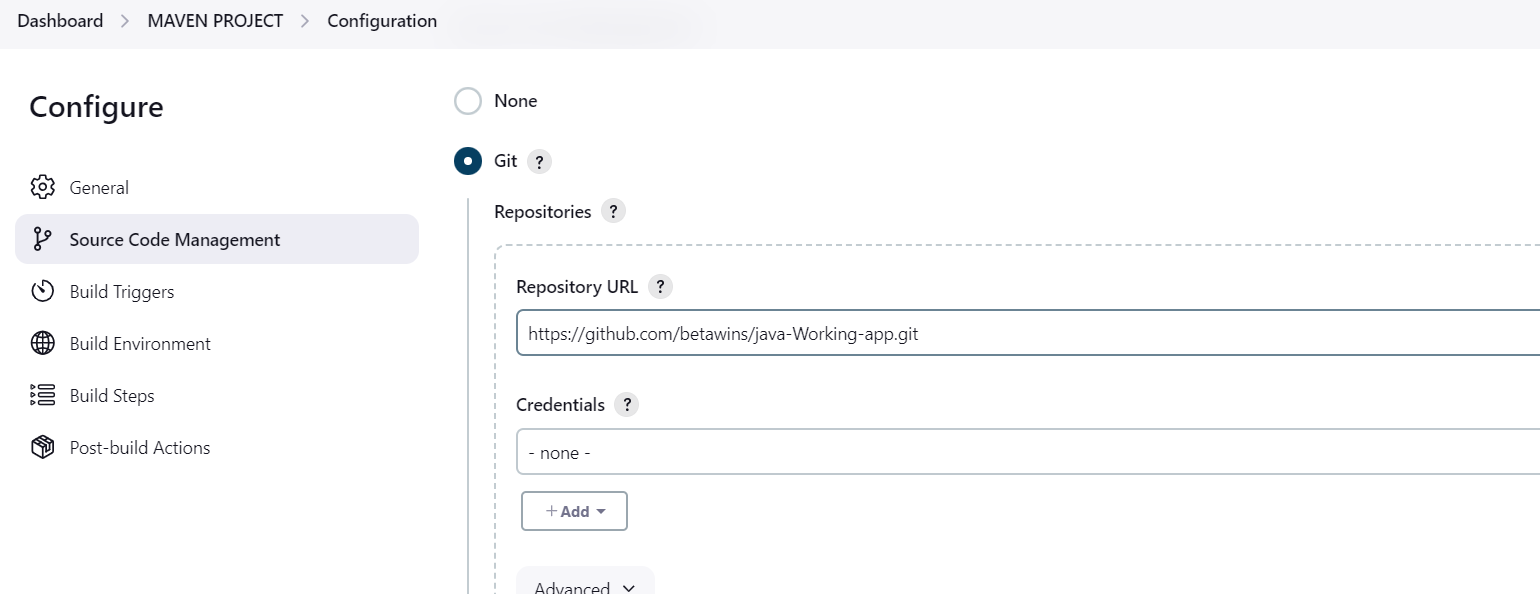
**4) Create one jenkins job using MAVEN PROJECT for the below code with two stages.**

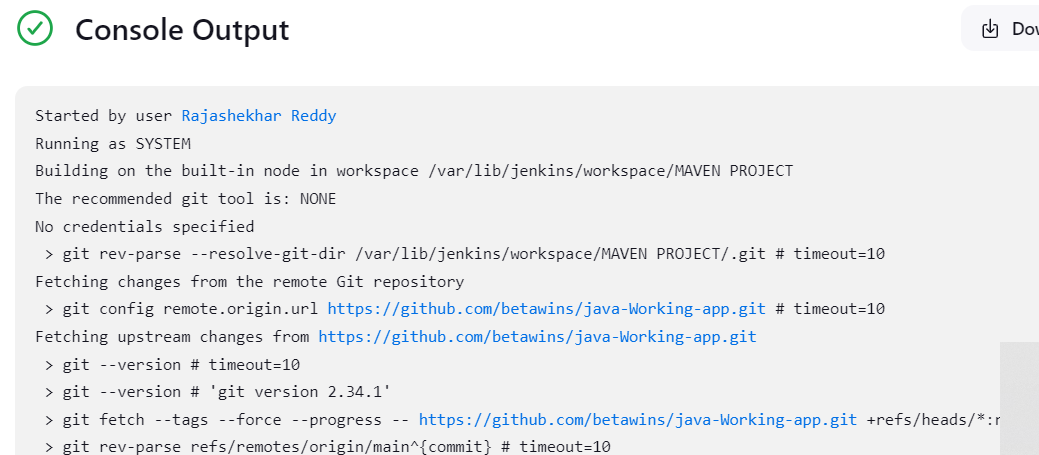
stage 1: Git clone

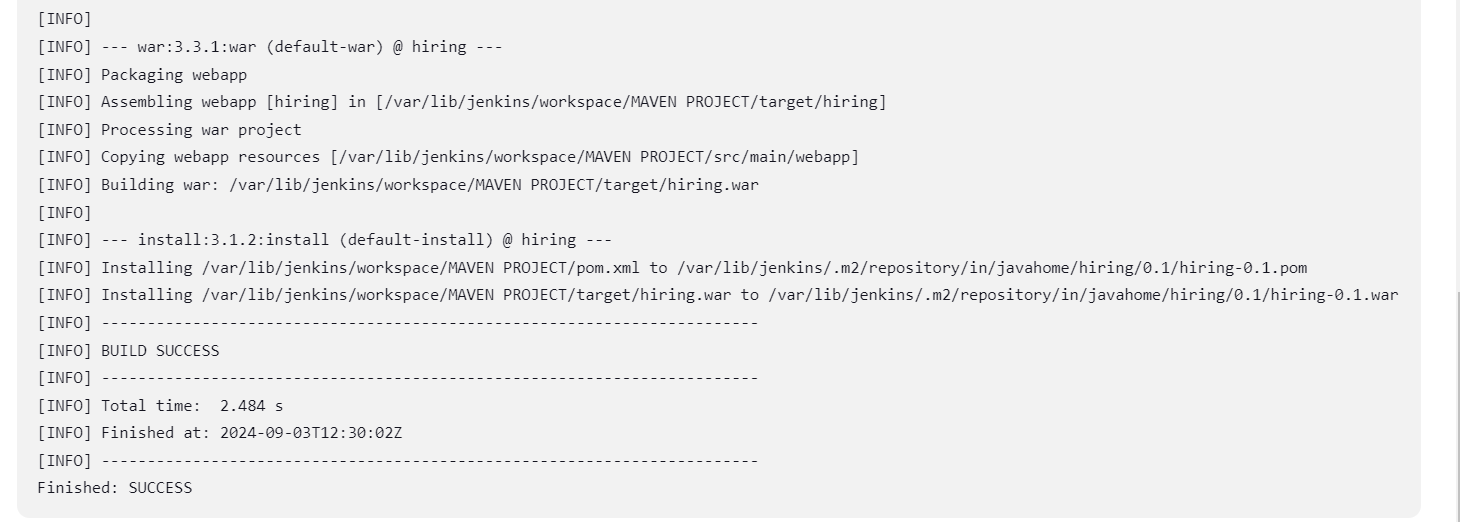
stage 2: Maven Compilation

Code: <https://github.com/betawins/java-Working-app.git>







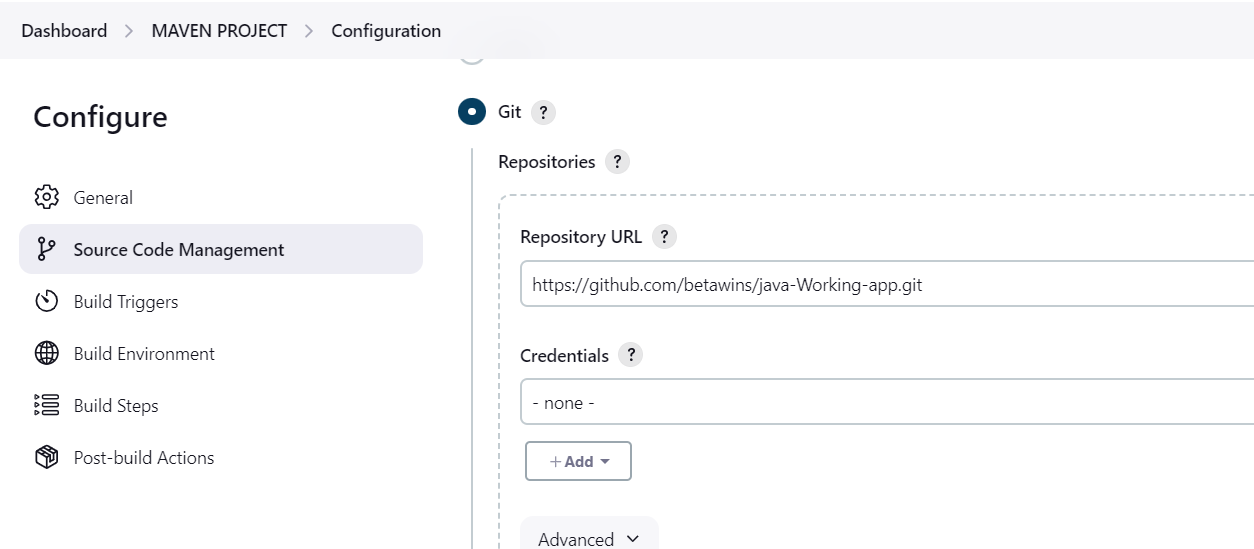


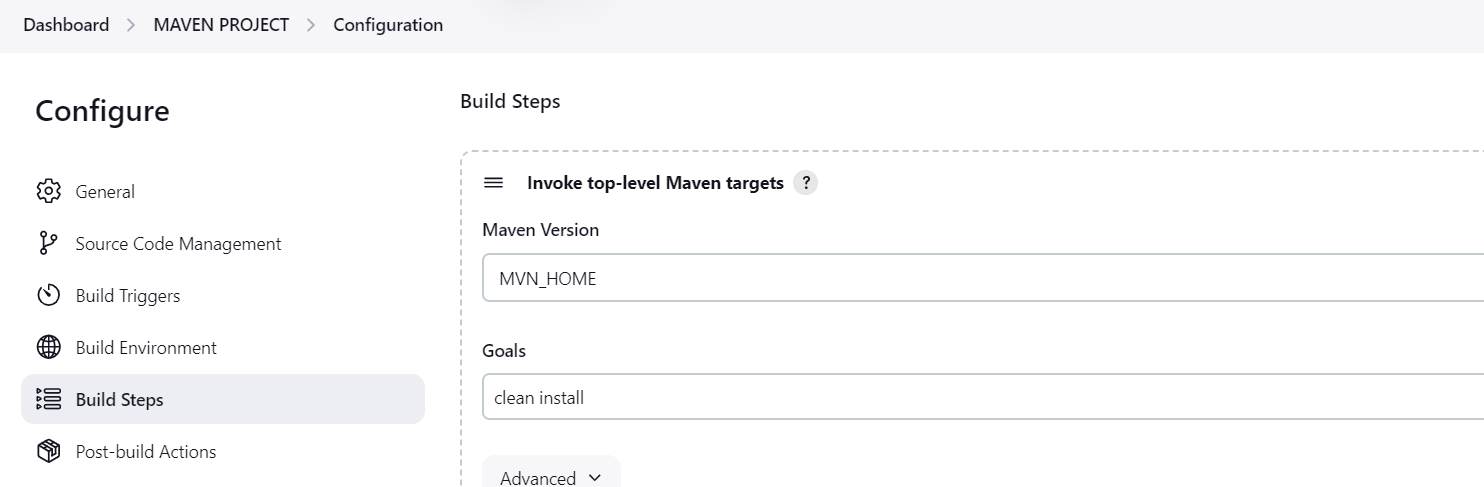
**5) Use the below code and create a parameterized job in jenkins**

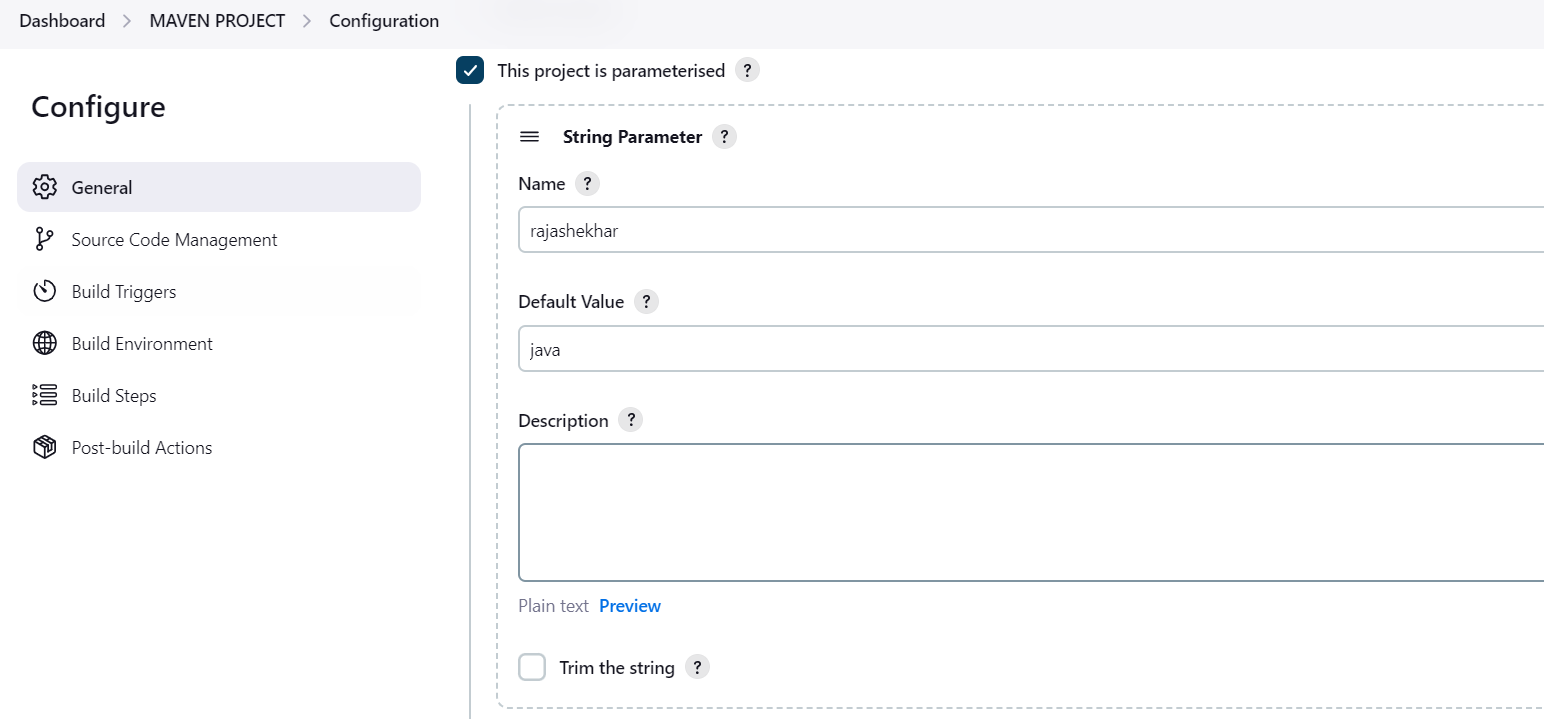
**stage 1: Git clone**

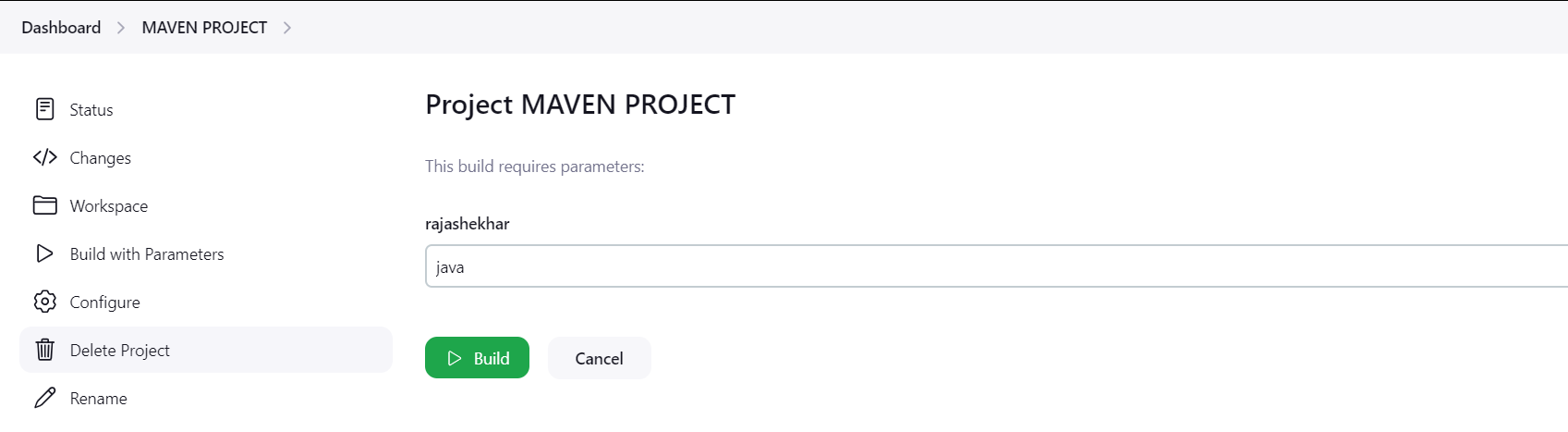
**stage 2: Maven Compilation**

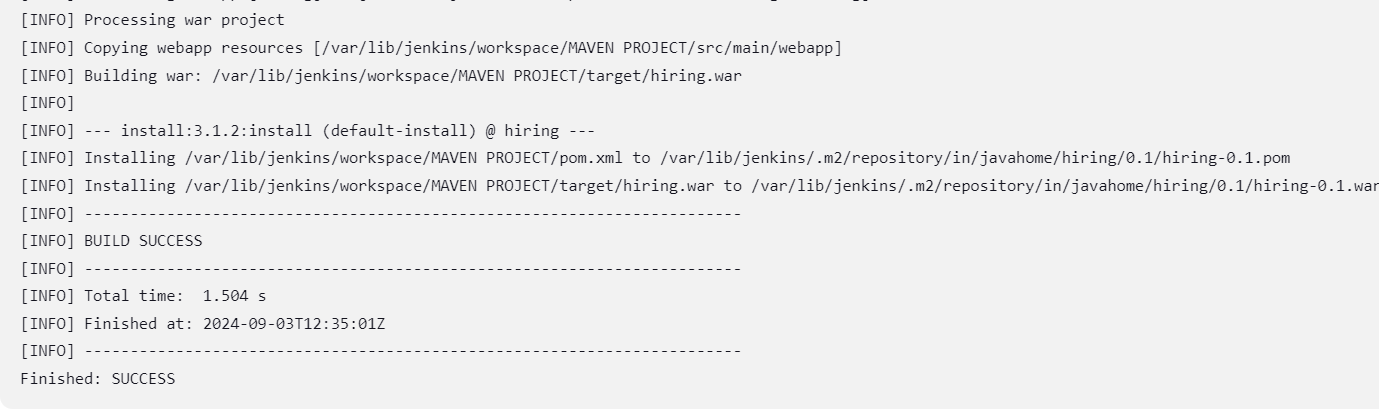
Code: <https://github.com/betawins/java-Working-app.git>











1. **What are the global varaiables in jenkins?**

* The Global YAML Properties Plugin is a universal extension for Jenkins that enhances Jenkins configuration by adding a Global YAML Configuration page. This page allows users to define global properties in YAML format, which is then parsed into a HashMap and can be accessed throughout specific build step
* The variables that are declared outside the given function are known as global variables. These do not stay limited to a specific function- which means that one can use any given function to not only access but also modify the global variables.