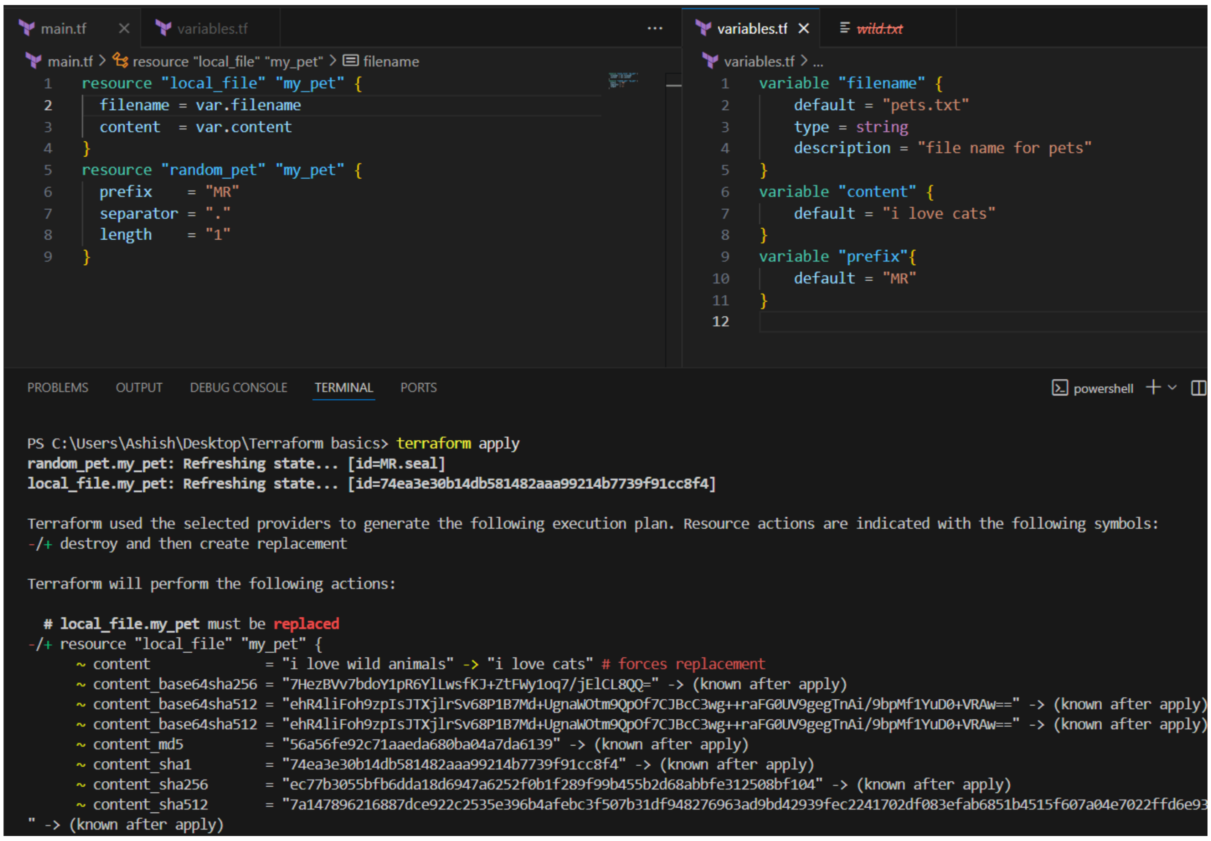
1. Watch the Terraform-03 video.

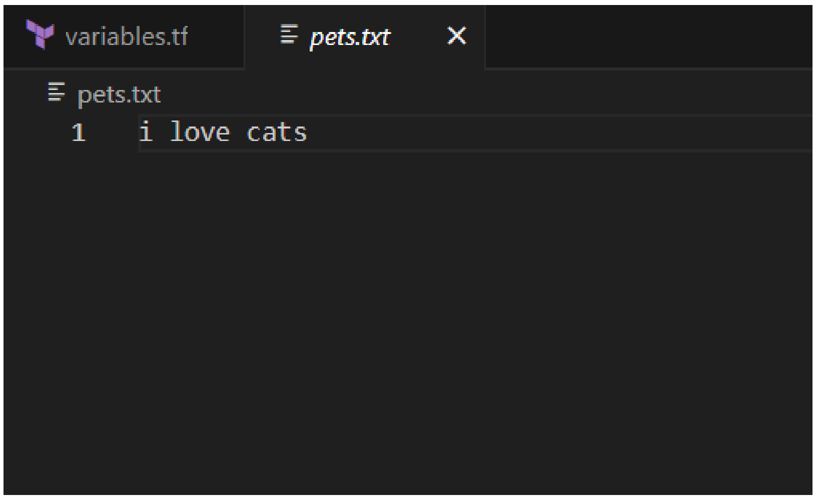
2. Execute the Script shown in the video.



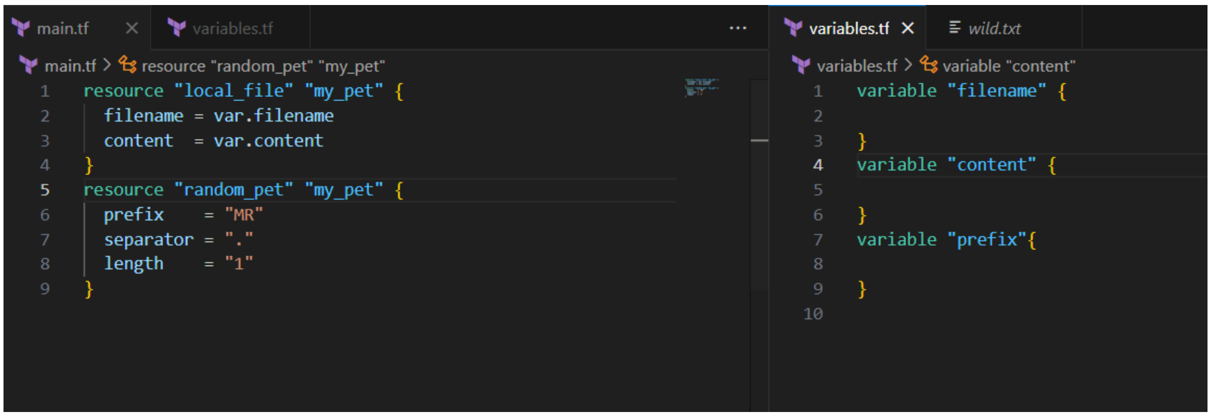
```
resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}
resource "random_pet" "my_pet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

```
variable "filename" {
    default = "pets.txt"
    type = string
    description = "file name for pets"
}
variable "content" {
    default = "i love cats"
}
variable "prefix"{
    default = "MR"
}
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
random_pet.my_pet: Refreshing state... [id=MR.seal]
local_file.my_pet: Refreshing state... [id=74ea3e30b14db581482aaa99214b7739f91cc8f4]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # local_file.my_pet must be replaced
-/+ resource "local_file" "my_pet" {
      ~ content              = "i love wild animals" -> "i love cats" # forces replacement
      ~ content_base64sha256 = "7HezBVv7bdoY1pR6YlLwsfKJ+ZtFWy1oq7/jElCL8QQ=" -> (known after apply)
      ~ content_base64sha512 = "ehR4liFoh9zpIsJTXjlrSv68P1B7Md+UgnaWOtm9QpOf7CJBcC3wg++raFG0UV9gegTnAi/9bpMf1YuD0+VRAw==" -> (known after apply)
      ~ content_base64sha512 = "ehR4liFoh9zpIsJTXjlrSv68P1B7Md+UgnaWOtm9QpOf7CJBcC3wg++raFG0UV9gegTnAi/9bpMf1YuD0+VRAw==" -> (known after apply)
      ~ content_md5          = "56a56fe92c71aaeda680ba04a7da6139" -> (known after apply)
      ~ content_sha1         = "74ea3e30b14db581482aaa99214b7739f91cc8f4" -> (known after apply)
      ~ content_sha256       = "ec77b3055bfb6dda18d6947a6252f0b1f289f99b455b2d68abbfe312508bf104" -> (known after apply)
      ~ content_sha512       = "7a147896216887dce922c2535e396b4afebc3f507b31df948276963ad9bd42939fec2241702df083efab6851b4515f607a04e7022ffd6e93
" -> (known after apply)
```



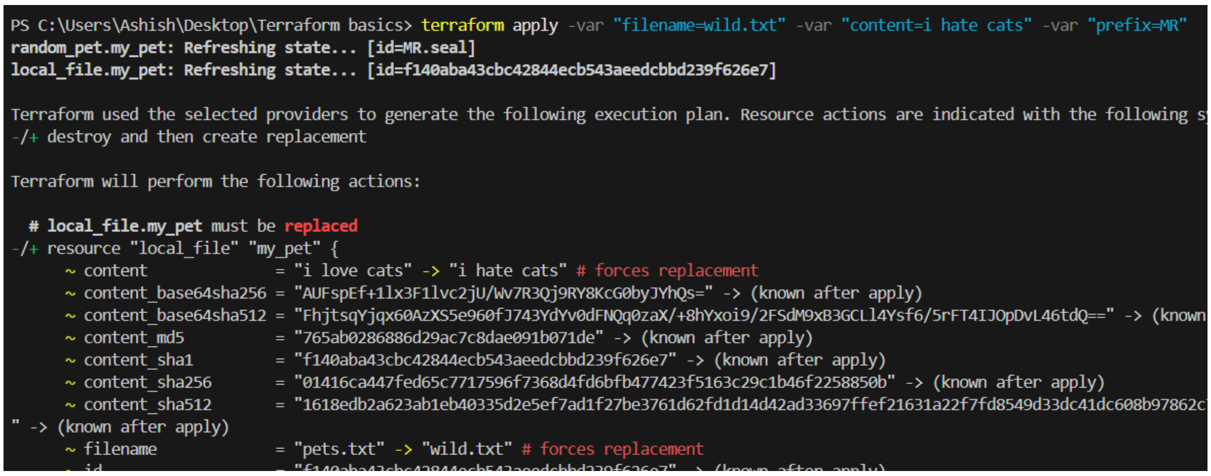
```
i love cats
```


main.tf
variables.tf
main.tf > resource "random_pet" "my_pet"
variables.tf
wild.txt
variables.tf > variable "content"

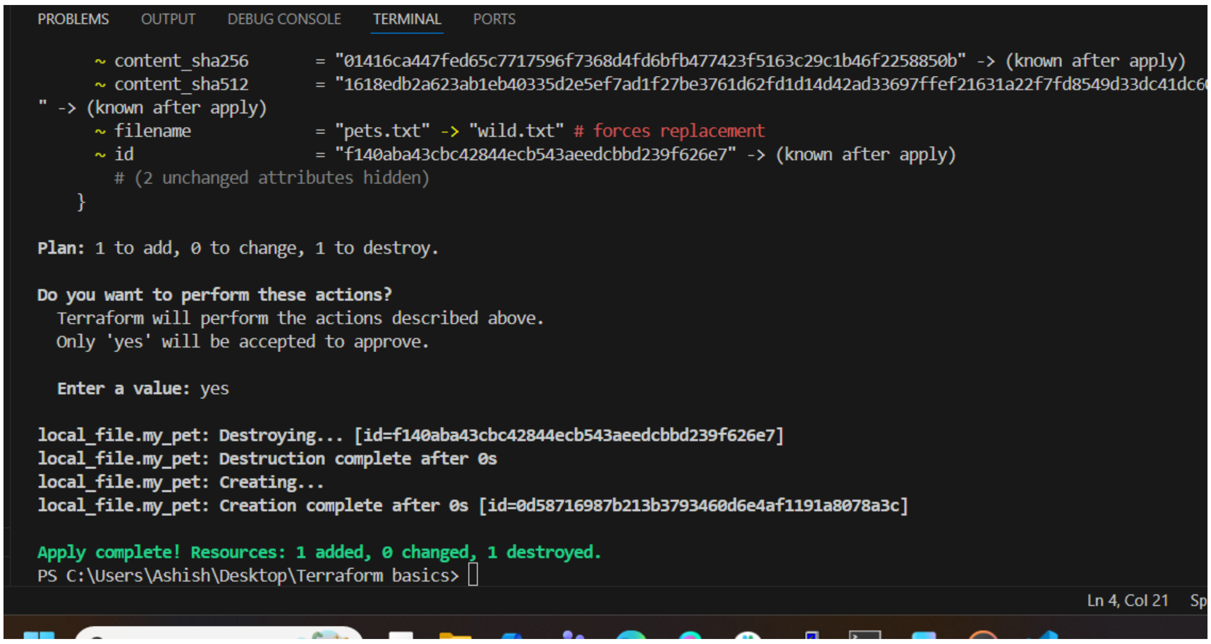
```
resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}
resource "random_pet" "my_pet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

```
variable "filename" {

}
variable "content" {

}
variable "prefix"{

}
```

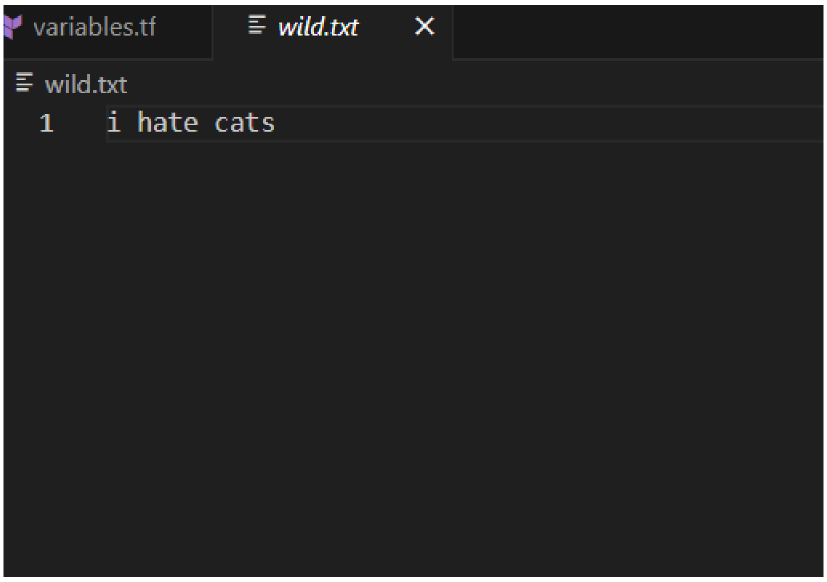
- terraform apply -var "filename=wild.txt" -var "content=i hate cats" -var "prefix=MR"



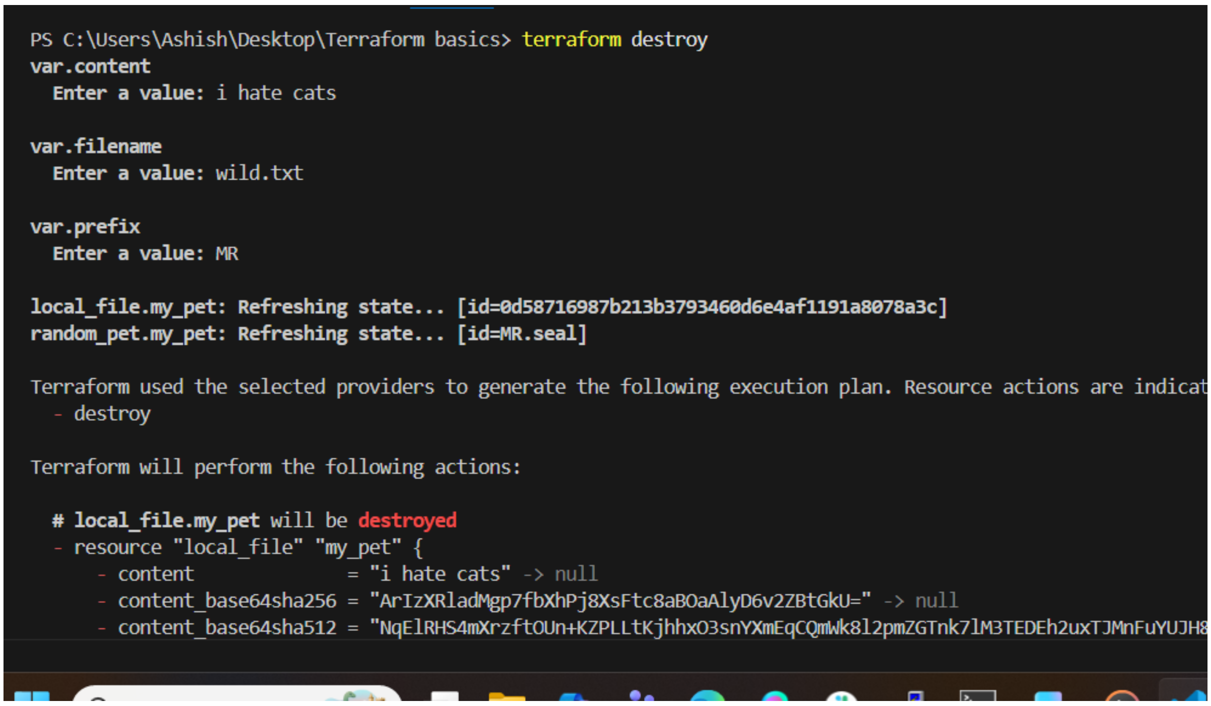
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply -var "filename=wild.txt" -var "content=i hate cats" -var "prefix=MR"
random_pet.my_pet: Refreshing state... [id=MR.seal]
local_file.my_pet: Refreshing state... [id=f140aba43cbc42844ecb543aeedcbbd239f626e7]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following s
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # local_file.my_pet must be replaced
-/+ resource "local_file" "my_pet" {
      ~ content              = "i love cats" -> "i hate cats" # forces replacement
      ~ content_base64sha256 = "AUFspEf+1lx3F1lvc2jU/Wv7R3Qj9RY8KcG0byJYhQs=" -> (known after apply)
      ~ content_base64sha512 = "FhjtsqYjqx60AzXS5e960fJ743YdYv0dFNQq0zaX/+8hYxoi9/2FSdM9xB3GCLl4Ysf6/5rFT4IJOpDvL46tdQ==" -> (known
      ~ content_md5          = "765ab0286886d29ac7c8dae091b071de" -> (known after apply)
      ~ content_sha1         = "f140aba43cbc42844ecb543aeedcbbd239f626e7" -> (known after apply)
      ~ content_sha256       = "01416ca447fed65c7717596f7368d4fd6bfb477423f5163c29c1b46f2258850b" -> (known after apply)
      ~ content_sha512       = "1618edb2a623ab1eb40335d2e5ef7ad1f27be3761d62fd1d14d42ad33697ffef21631a22f7fd8549d33dc41dc608b97862c
" -> (known after apply)
      ~ filename             = "pets.txt" -> "wild.txt" # forces replacement
```



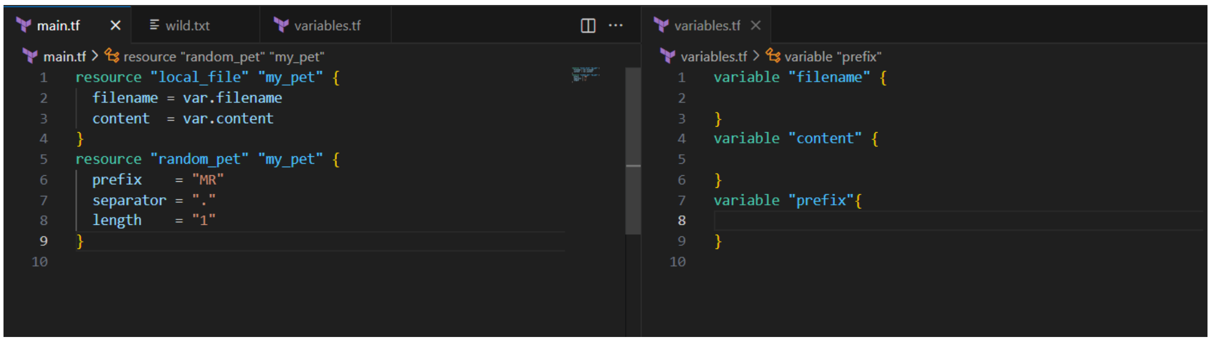
```
      ~ content_sha256       = "01416ca447fed65c7717596f7368d4fd6bfb477423f5163c29c1b46f2258850b" -> (known after apply)
      ~ content_sha512       = "1618edb2a623ab1eb40335d2e5ef7ad1f27be3761d62fd1d14d42ad33697ffef21631a22f7fd8549d33dc41dc6
" -> (known after apply)
      ~ filename             = "pets.txt" -> "wild.txt" # forces replacement
      ~ id                   = "f140aba43cbc42844ecb543aeedcbbd239f626e7" -> (known after apply)
        # (2 unchanged attributes hidden)
    }

Plan: 1 to add, 0 to change, 1 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.my_pet: Destroying... [id=f140aba43cbc42844ecb543aeedcbbd239f626e7]
local_file.my_pet: Destruction complete after 0s
local_file.my_pet: Creating...
local_file.my_pet: Creation complete after 0s [id=0d58716987b213b3793460d6e4af1191a8078a3c]

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



```
i hate cats
```

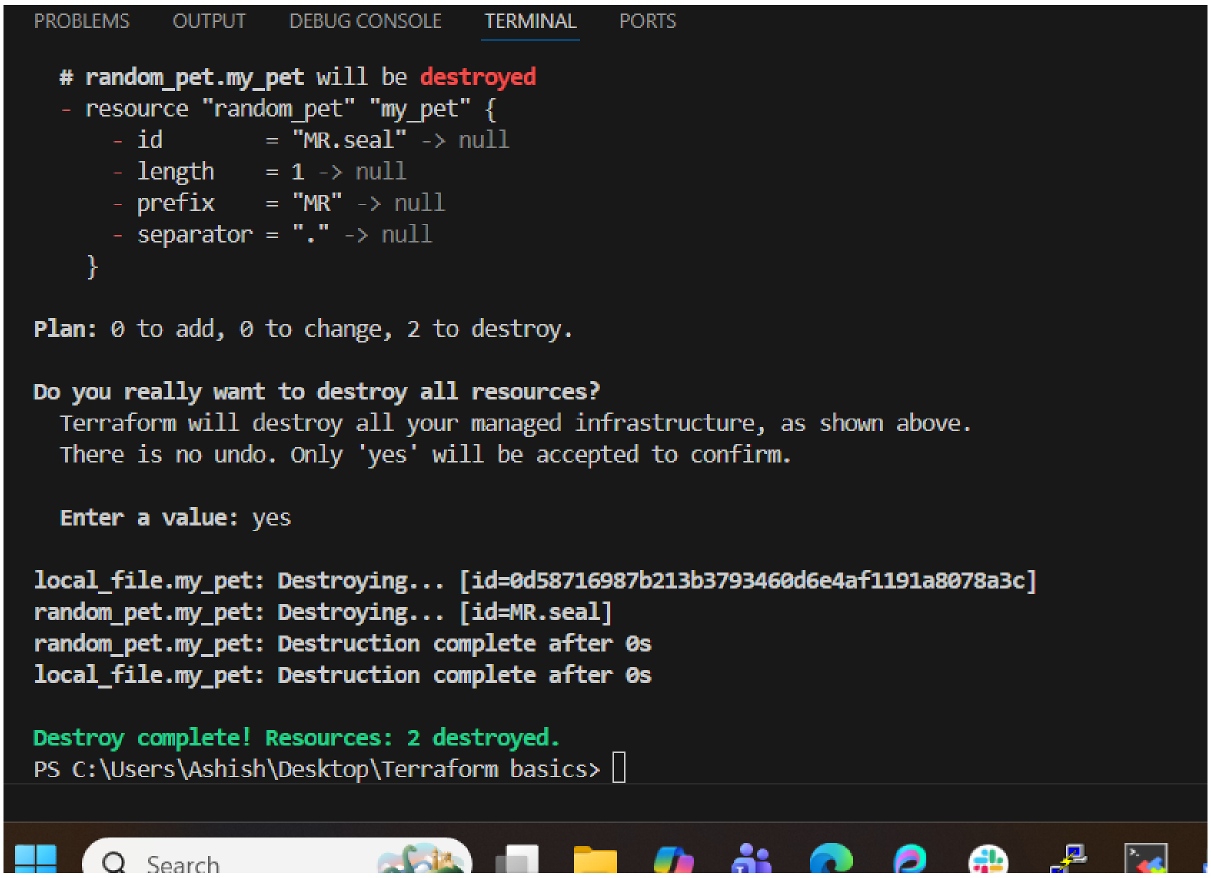


```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform destroy
var.content
  Enter a value: i hate cats

var.filename
  Enter a value: wild.txt

var.prefix
  Enter a value: MR

local_file.my_pet: Refreshing state... [id=0d58716987b213b3793460d6e4af1191a8078a3c]
random_pet.my_pet: Refreshing state... [id=MR.seal]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicat
  - destroy

Terraform will perform the following actions:

  # local_file.my_pet will be destroyed
  - resource "local_file" "my_pet" {
      - content              = "i hate cats" -> null
      - content_base64sha256 = "ArIzXRladMgp7fbXhPj8XsFtc8aBOaAlyD6v2ZBtGkU=" -> null
      - content_base64sha512 = "NqElRHS4mXrzftOUn+KZPLLtKjhhxO3snYXmEqCQmWk8l2pmZGTnk7lM3TEDEh2uxTJMnFuYUJH8
```


main.tf
wild.txt
variables.tf
main.tf > resource "random_pet" "my_pet"
variables.tf
variables.tf > variable "prefix"

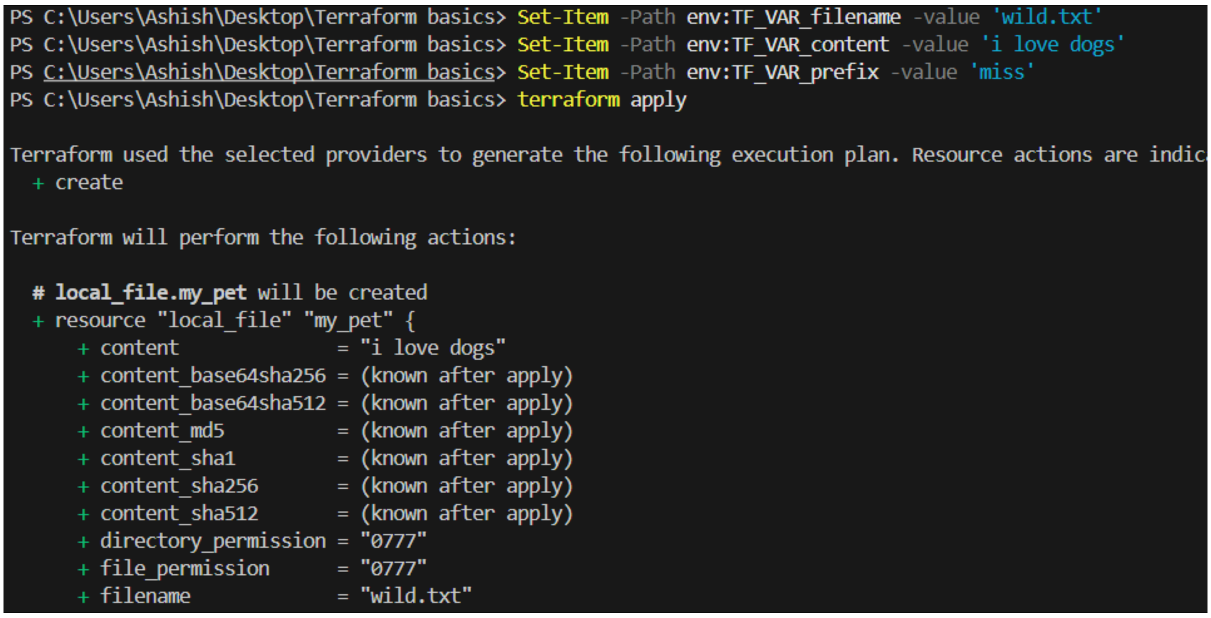
```
resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}
resource "random_pet" "my_pet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

```
variable "filename" {

}
variable "content" {

}
variable "prefix"{

}
```

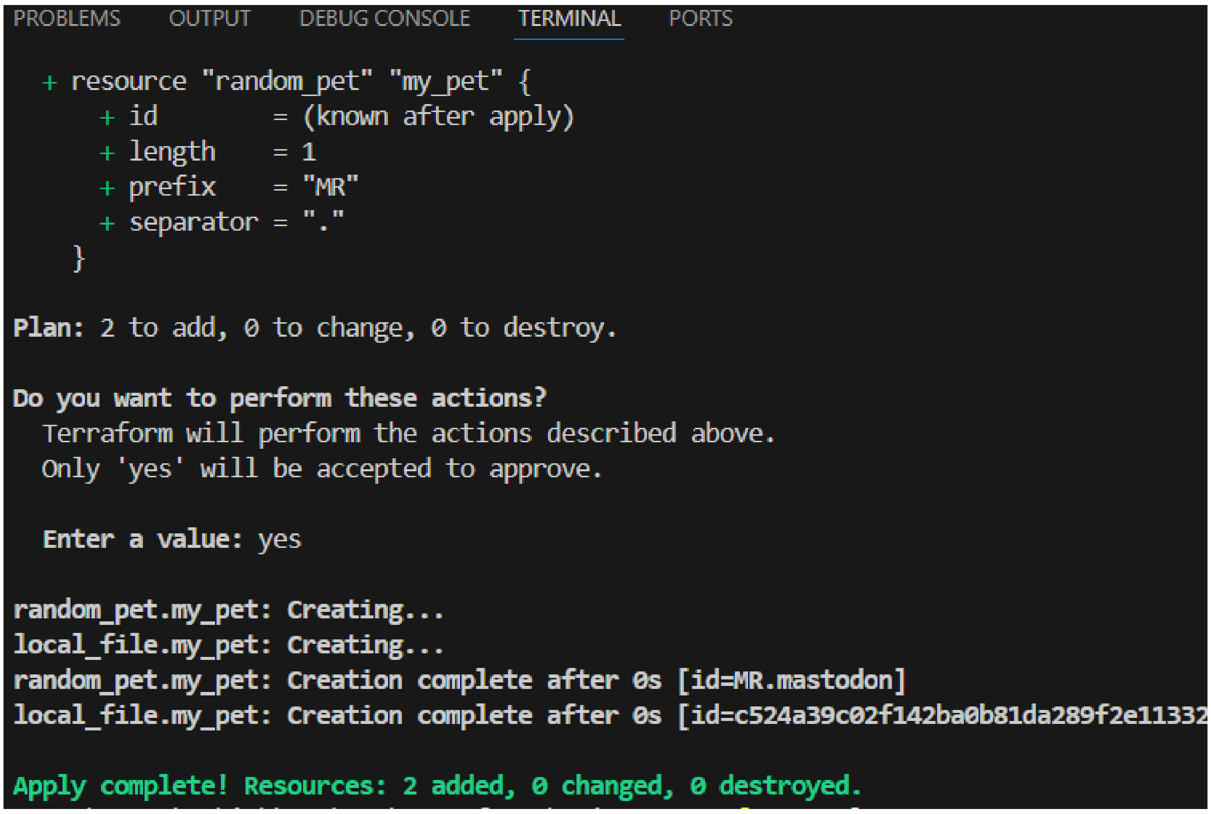


```
  # random_pet.my_pet will be destroyed
  - resource "random_pet" "my_pet" {
      - id        = "MR.seal" -> null
      - length    = 1 -> null
      - prefix    = "MR" -> null
      - separator = "." -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

local_file.my_pet: Destroying... [id=0d58716987b213b3793460d6e4af1191a8078a3c]
random_pet.my_pet: Destroying... [id=MR.seal]
random_pet.my_pet: Destruction complete after 0s
local_file.my_pet: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

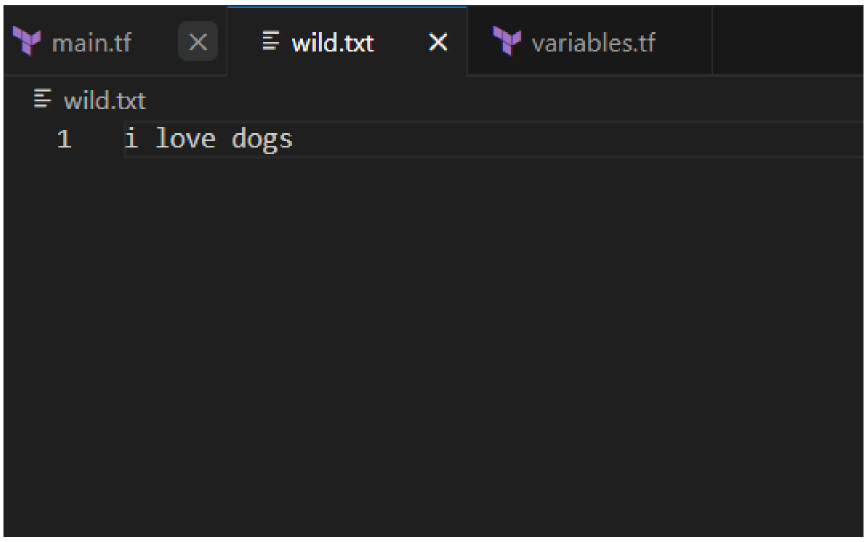
- Set-Item -Path env:TF_VAR_filename -value 'wild.txt'
- Set-Item -Path env:TF_VAR_content -value 'i love dogs'
- Set-Item -Path env:TF_VAR_prefix -value 'miss'



```
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_filename -value 'wild.txt'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_content -value 'i love dogs'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_prefix -value 'miss'
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indic
  + create

Terraform will perform the following actions:

  # local_file.my_pet will be created
  + resource "local_file" "my_pet" {
      + content              = "i love dogs"
      + 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             = "wild.txt"
```

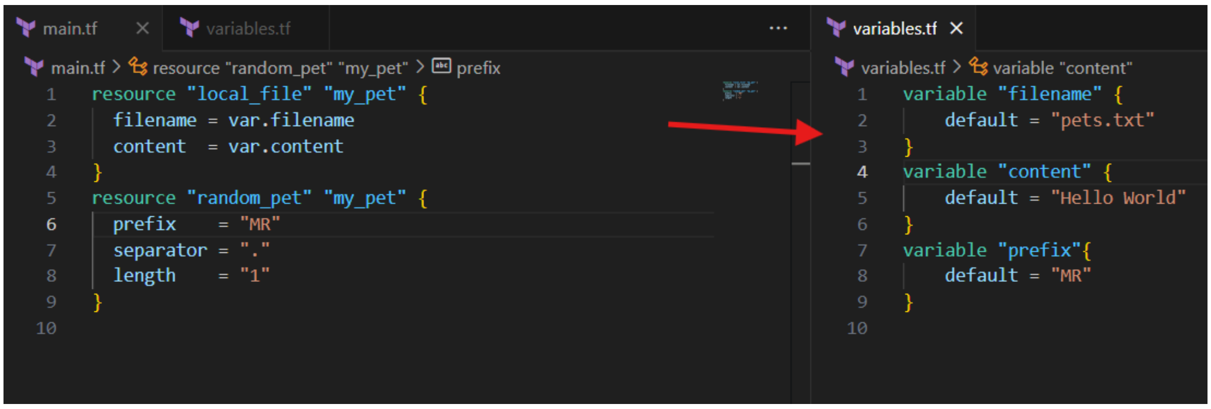


```
  + resource "random_pet" "my_pet" {
      + id        = (known after apply)
      + length    = 1
      + prefix    = "MR"
      + separator = "."
    }

Plan: 2 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.my_pet: Creating...
local_file.my_pet: Creating...
random_pet.my_pet: Creation complete after 0s [id=MR.mastodon]
local_file.my_pet: Creation complete after 0s [id=c524a39c02f142ba0b81da289f2e11332

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```



```
i love dogs
```

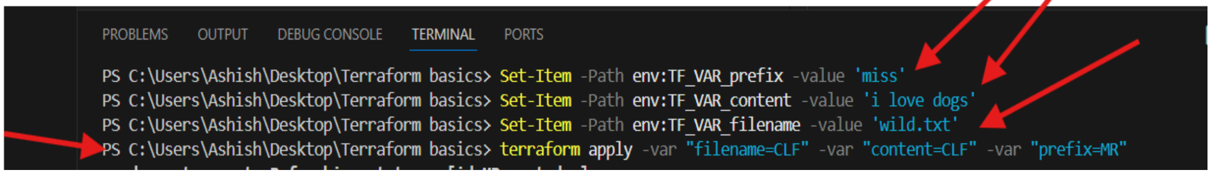
- Giving 3 types of variables at the same time


main.tf
variables.tf
main.tf > resource "random_pet" "my_pet" > prefix
variables.tf > variable "content"

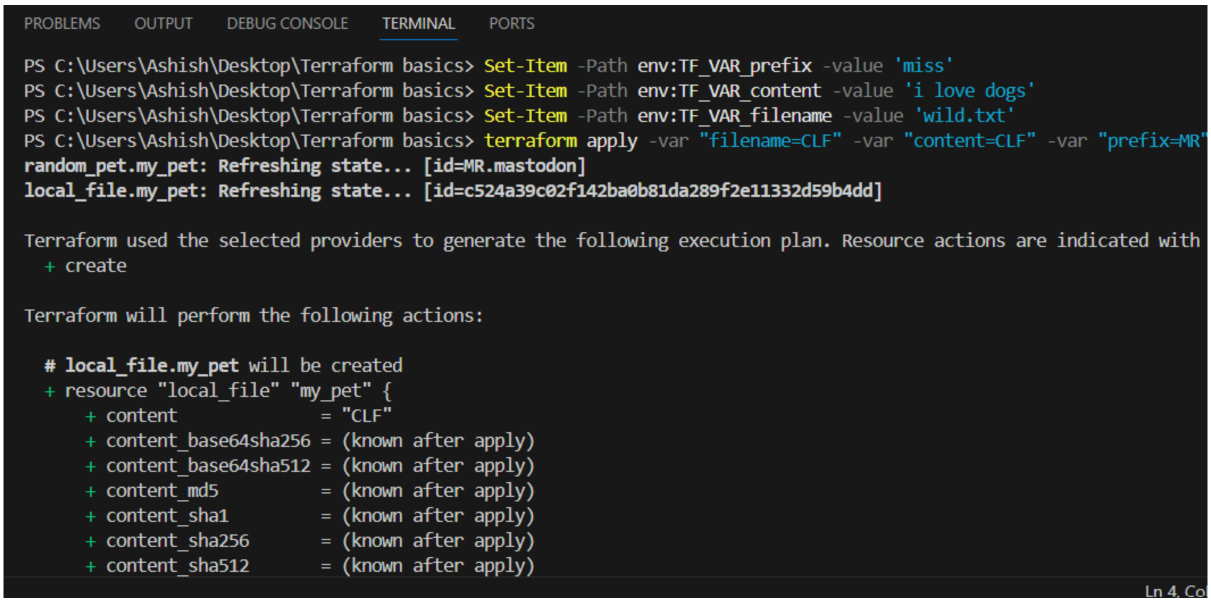
```
resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}
resource "random_pet" "my_pet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

```
variable "filename" {
    default = "pets.txt"
}
variable "content" {
    default = "Hello World"
}
variable "prefix"{
    default = "MR"
}
```

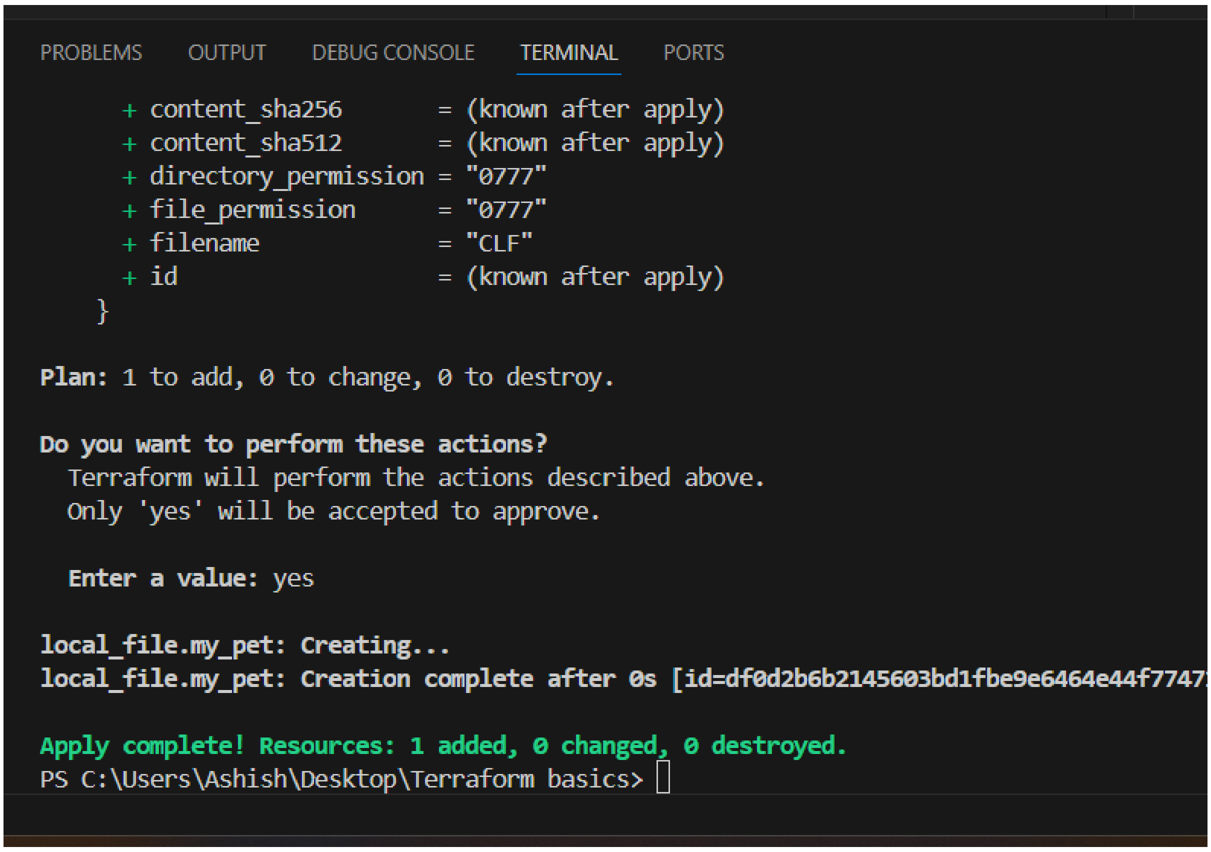
- Set-Item -Path env:TF_VAR_prefix -value 'miss'
- Set-Item -Path env:TF_VAR_content -value 'i love dogs'
- Set-Item -Path env:TF_VAR_filename -value 'wild.txt'
- terraform apply -var "filename=CLF" -var "content=CLF" -var "prefix=MR"



```
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_prefix -value 'miss'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_content -value 'i love dogs'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_filename -value 'wild.txt'
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply -var "filename=CLF" -var "content=CLF" -var "prefix=MR"
```



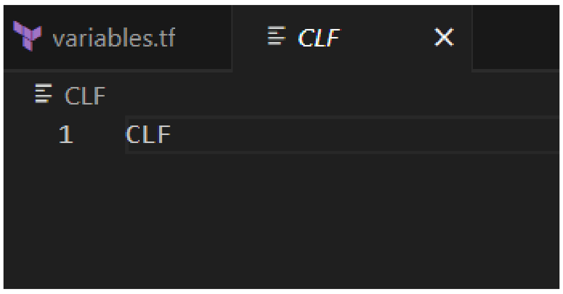
```
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_prefix -value 'miss'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_content -value 'i love dogs'
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_VAR_filename -value 'wild.txt'
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply -var "filename=CLF" -var "content=CLF" -var "prefix=MR"
random_pet.my_pet: Refreshing state... [id=MR.mastodon]
local_file.my_pet: Refreshing state... [id=c524a39c02f142ba0b81da289f2e11332d59b4dd]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
  + create

Terraform will perform the following actions:

  # local_file.my_pet will be created
  + resource "local_file" "my_pet" {
      + content              = "CLF"
      + 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)
```



```
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "CLF"
      + 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.my_pet: Creating...
local_file.my_pet: Creation complete after 0s [id=df0d2b6b2145603bd1fbe9e6464e44f7747

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

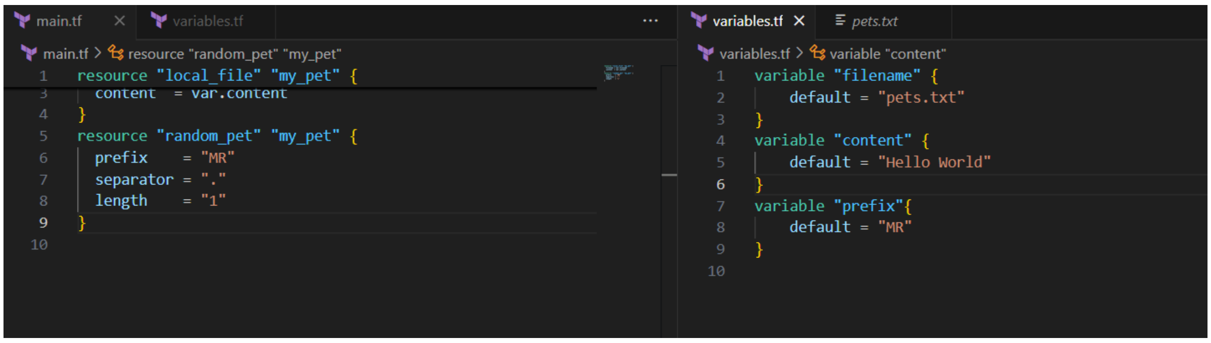
Then CLF variable is created

- Because of precedence order flags variable will be executed.



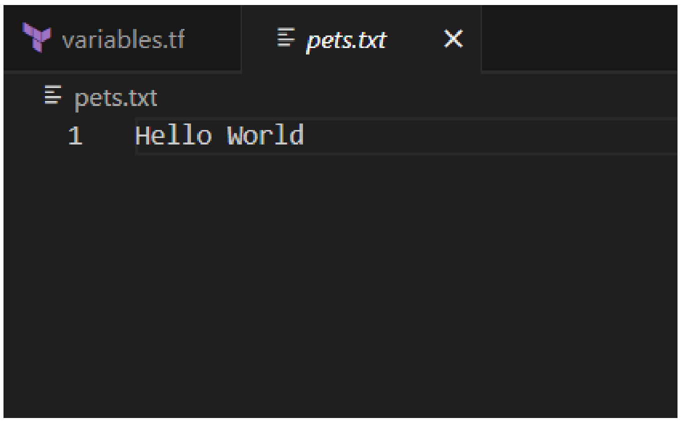
```
CLF
```

Destroy all and open from back


main.tf
variables.tf
main.tf > resource "random_pet" "my_pet"
variables.tf
pets.txt
variables.tf > variable "content"

```
resource "local_file" "my_pet" {
  content  = var.content
}
resource "random_pet" "my_pet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

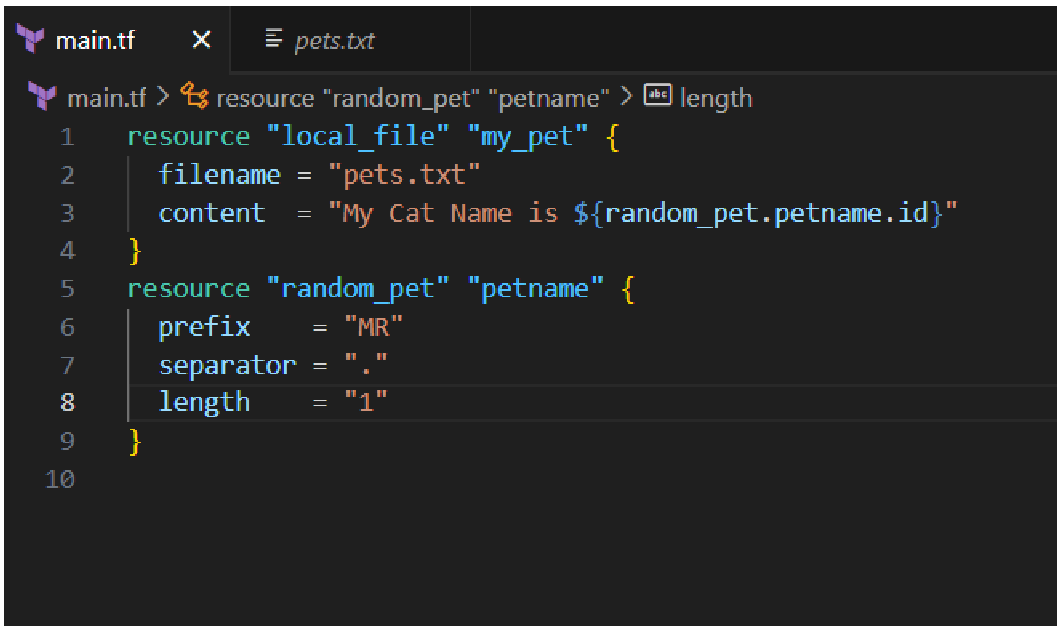
```
variable "filename" {
    default = "pets.txt"
}
variable "content" {
    default = "Hello World"
}
variable "prefix"{
    default = "MR"
}
```



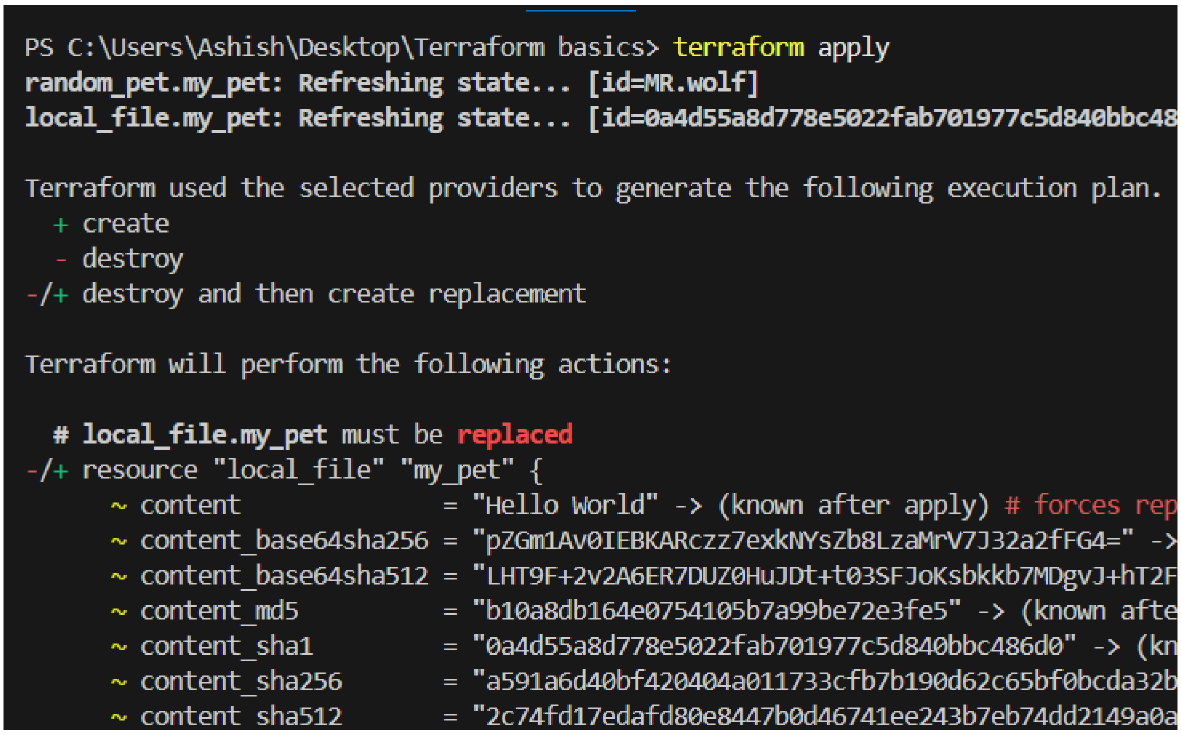
```
Hello World
```

Destroy variable.tf

Make changes in main.tf

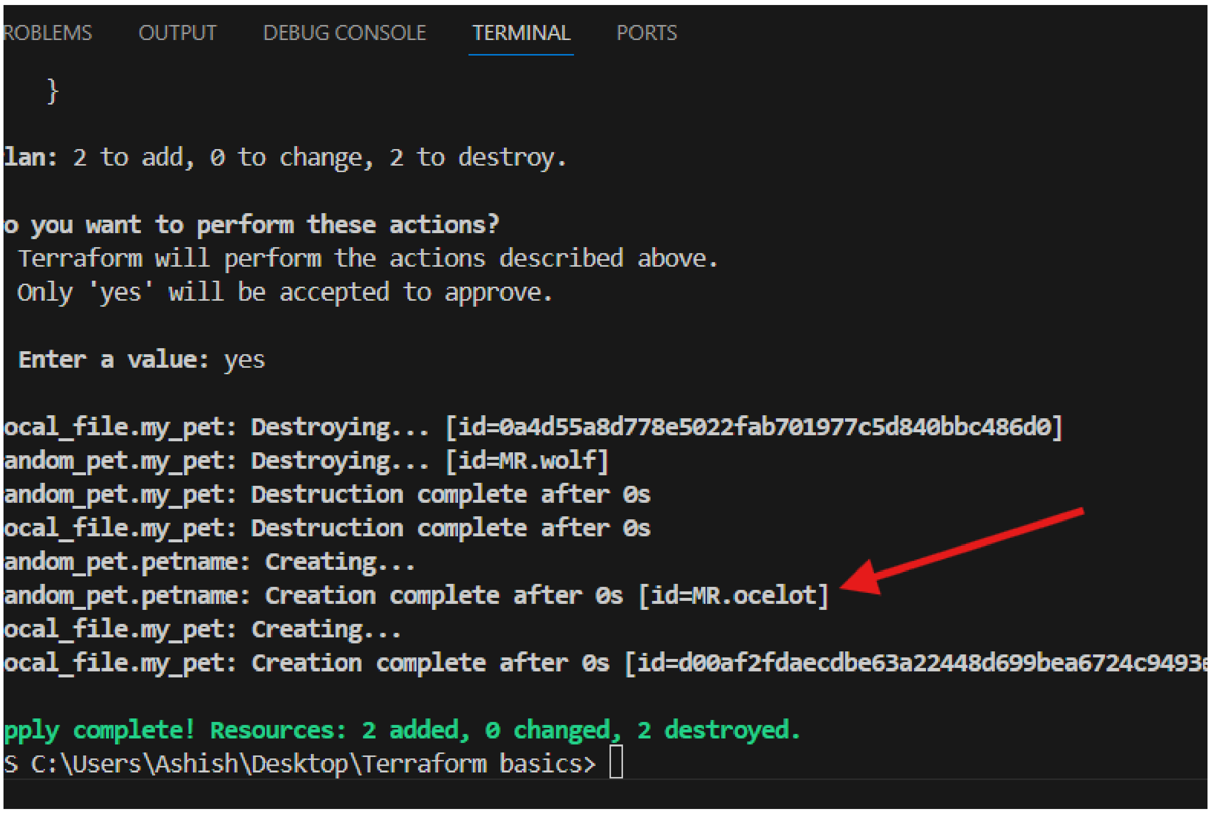


```
resource "local_file" "my_pet" {
  filename = "pets.txt"
  content  = "My Cat Name is ${random_pet.petname.id}"
}
resource "random_pet" "petname" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```



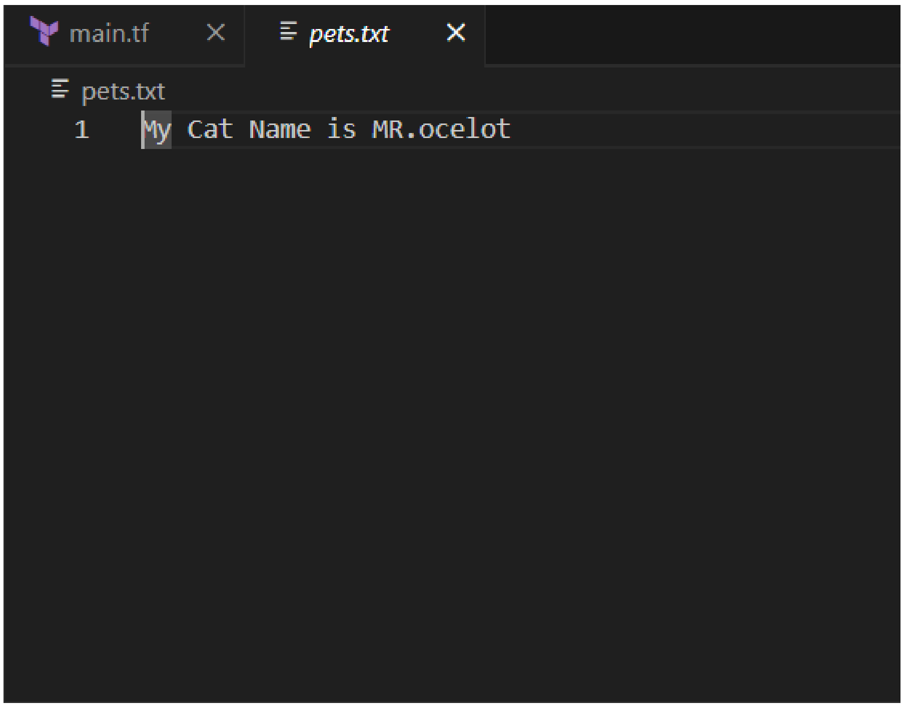
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
random_pet.my_pet: Refreshing state... [id=MR.wolf]
local_file.my_pet: Refreshing state... [id=0a4d55a8d778e5022fab701977c5d840bbc48

Terraform used the selected providers to generate the following execution plan.
  + create
  - destroy
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # local_file.my_pet must be replaced
-/+ resource "local_file" "my_pet" {
      ~ content              = "Hello World" -> (known after apply) # forces rep
      ~ content_base64sha256 = "pZGm1Av0IEBKARczz7exkNYsZb8LzaMrV7J32a2fFG4=" ->
      ~ content_base64sha512 = "LHT9F+2v2A6ER7DUZ0HuJDt+t03SFJoKsbkkb7MDgvJ+hT2F
      ~ content_md5          = "b10a8db164e0754105b7a99be72e3fe5" -> (known afte
      ~ content_sha1         = "0a4d55a8d778e5022fab701977c5d840bbc486d0" -> (kn
      ~ content_sha256       = "a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b
      ~ content_sha512       = "2c74fd17edafd80e8447b0d46741ee243b7eb74dd2149a0a
```

A petname has created MR.ocelot



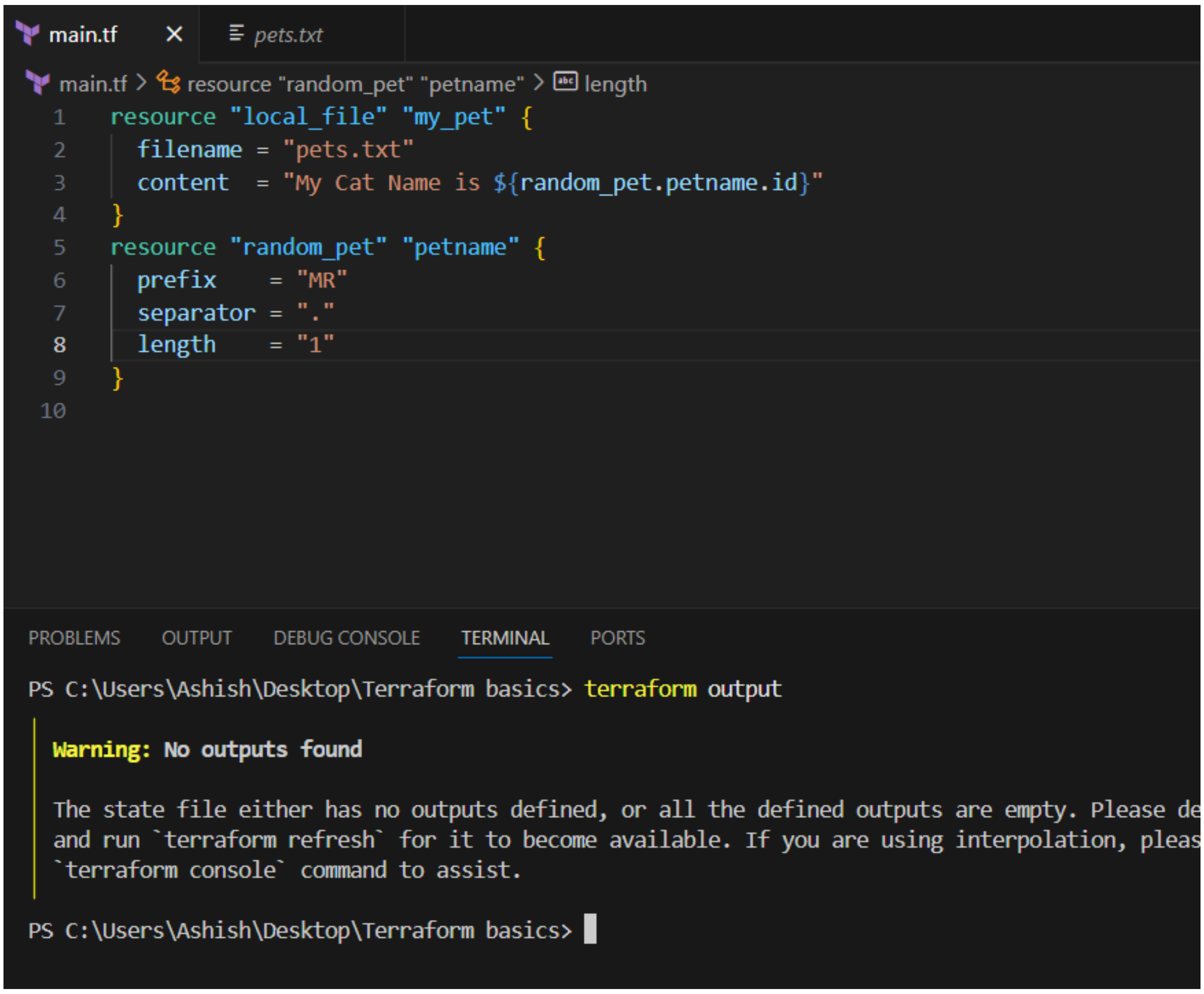
```
   }

lan: 2 to add, 0 to change, 2 to destroy.

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

 Enter a value: yes

ocal_file.my_pet: Destroying... [id=0a4d55a8d778e5022fab701977c5d840bbc486d0]
andom_pet.my_pet: Destroying... [id=MR.wolf]
andom_pet.my_pet: Destruction complete after 0s
ocal_file.my_pet: Destruction complete after 0s
andom_pet.petname: Creating...
andom_pet.petname: Creation complete after 0s [id=MR.ocelot]
ocal_file.my_pet: Creating...
ocal_file.my_pet: Creation complete after 0s [id=d00af2fdaecdbe63a22448d699bea6724c9493

pply complete! Resources: 2 added, 0 changed, 2 destroyed.
S C:\Users\Ashish\Desktop\Terraform basics>
```

If I open pet.txt



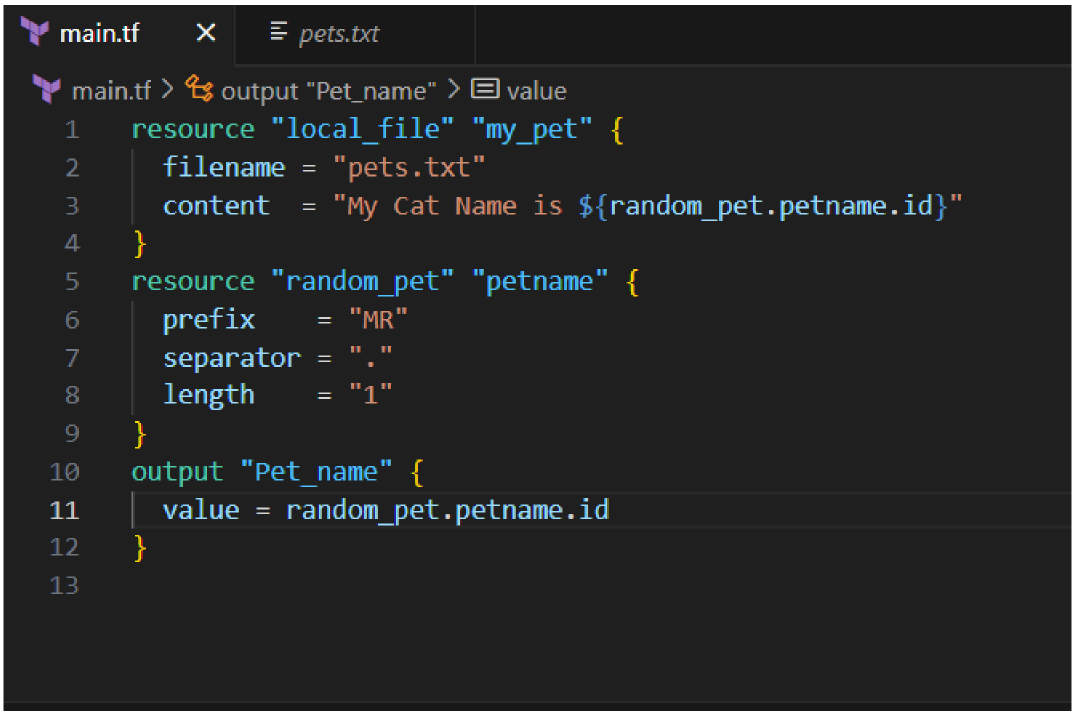
```
My Cat Name is MR.ocelot
```

- terraform output

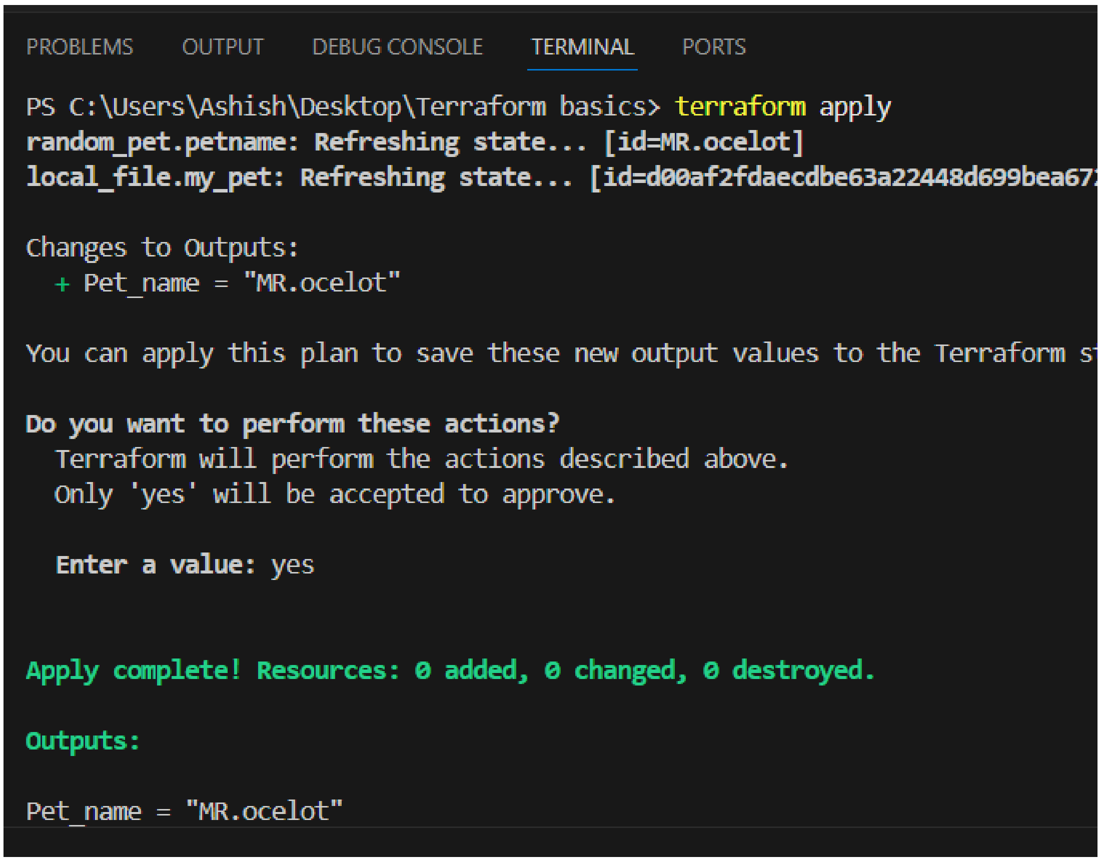


```
resource "local_file" "my_pet" {
  filename = "pets.txt"
  content  = "My Cat Name is ${random_pet.petname.id}"
}
resource "random_pet" "petname" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

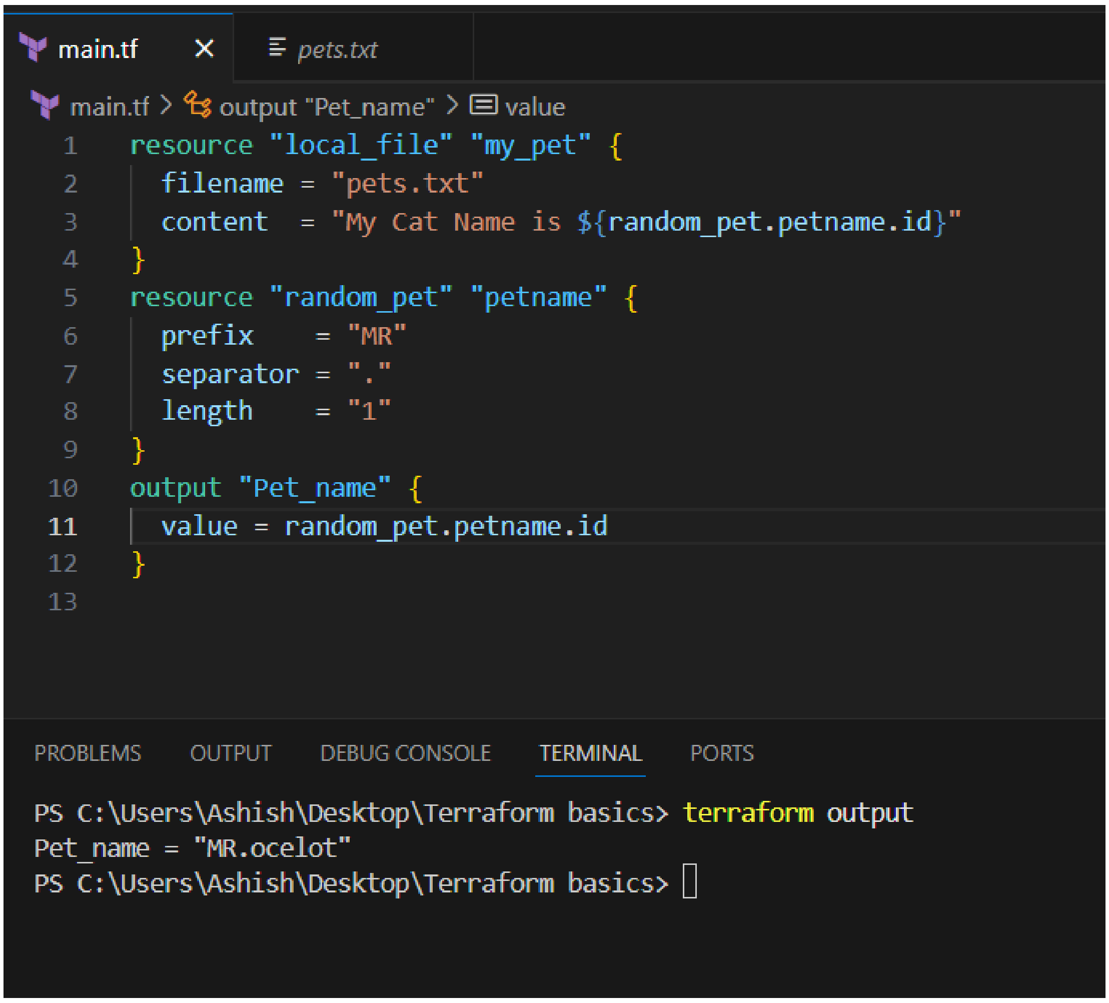
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform output

  Warning: No outputs found

  The state file either has no outputs defined, or all the defined outputs are empty. Please de
  and run `terraform refresh` for it to become available. If you are using interpolation, pleas
  `terraform console` command to assist.

PS C:\Users\Ashish\Desktop\Terraform basics>
```



```
resource "local_file" "my_pet" {
  filename = "pets.txt"
  content  = "My Cat Name is ${random_pet.petname.id}"
}
resource "random_pet" "petname" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
output "Pet_name" {
  value = random_pet.petname.id
}

```



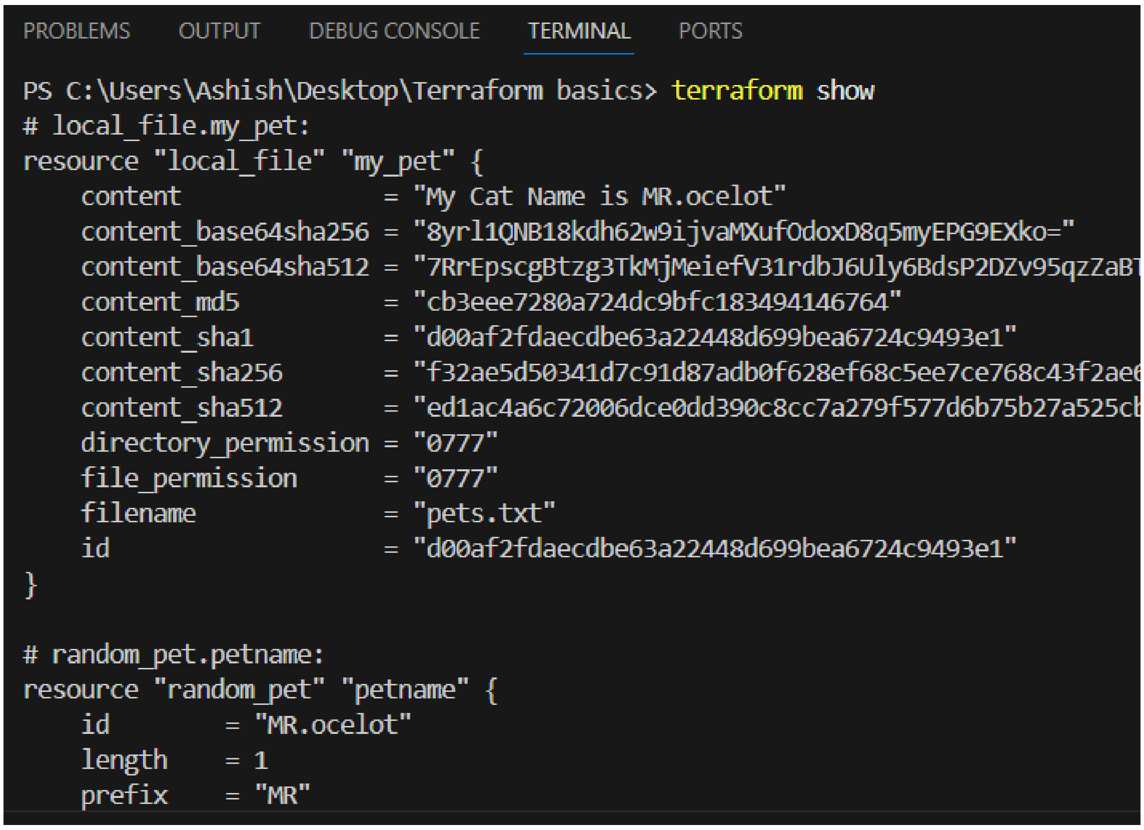
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
random_pet.petname: Refreshing state... [id=MR.ocelot]
local_file.my_pet: Refreshing state... [id=d00af2fdaecdbe63a22448d699bea67

Changes to Outputs:
  + Pet_name = "MR.ocelot"

You can apply this plan to save these new output values to the Terraform s

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


Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

Pet_name = "MR.ocelot"
```



```
resource "local_file" "my_pet" {
  filename = "pets.txt"
  content  = "My Cat Name is ${random_pet.petname.id}"
}
resource "random_pet" "petname" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
output "Pet_name" {
  value = random_pet.petname.id
}

```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform output
Pet_name = "MR.ocelot"
PS C:\Users\Ashish\Desktop\Terraform basics>
```

- terraform show

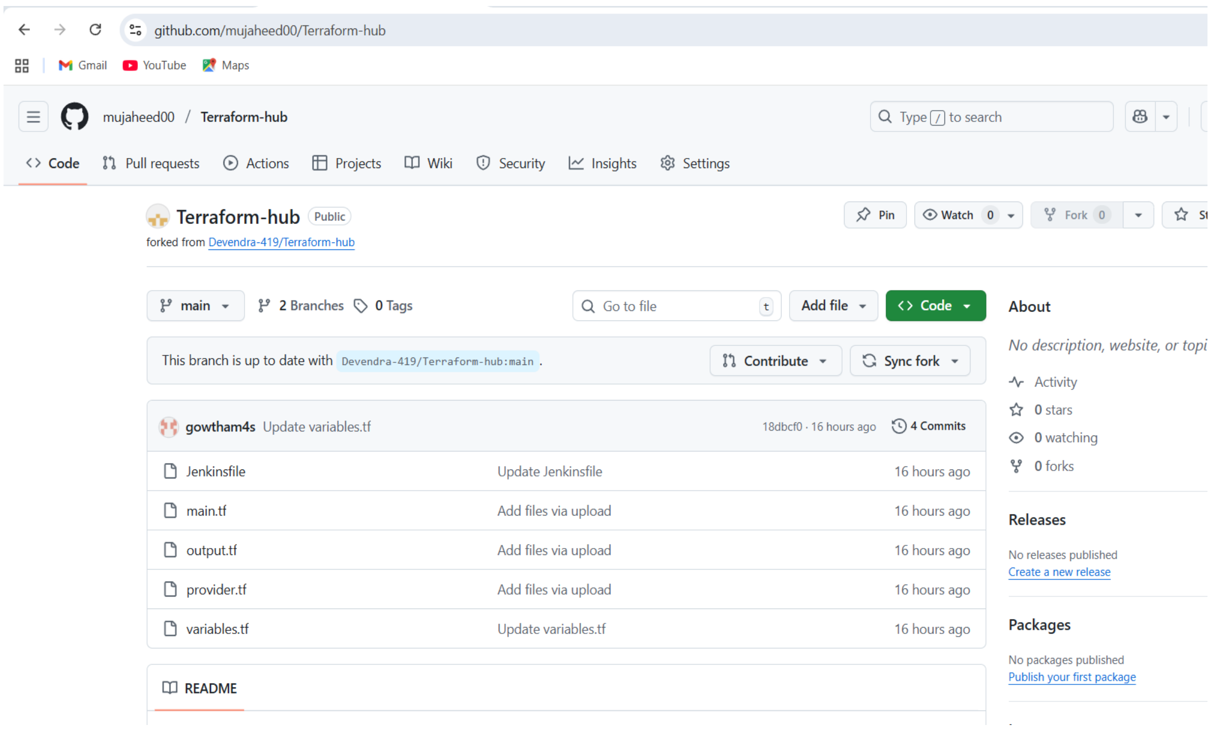


```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform show
# local_file.my_pet:
resource "local_file" "my_pet" {
    content              = "My Cat Name is MR.ocelot"
    content_base64sha256 = "8yrl1QNB18kdh62w9ijvaMXufOdoxD8q5myEPG9EXko="
    content_base64sha512 = "7RrEpscgBtzg3TkMjMeiefV31rdbJ6Uly6BdsP2DZv95qzZaBT
    content_md5          = "cb3eee7280a724dc9bfc183494146764"
    content_sha1         = "d00af2fdaecdbe63a22448d699bea6724c9493e1"
    content_sha256       = "f32ae5d50341d7c91d87adb0f628ef68c5ee7ce768c43f2ae6
    content_sha512       = "ed1ac4a6c72006dce0dd390c8cc7a279f577d6b75b27a525cb
    directory_permission = "0777"
    file_permission      = "0777"
    filename             = "pets.txt"
    id                   = "d00af2fdaecdbe63a22448d699bea6724c9493e1"
}

# random_pet.petname:
resource "random_pet" "petname" {
    id        = "MR.ocelot"
    length    = 1
    prefix    = "MR"
```

3. Integrate Terraform in Jenkins using the Terraform plugin.

Keep all your files in an repository in github.

https://github.com/mujaheed00/Terraform-hub.git


github.com/mujaheed00/Terraform-hub

mujaheed00 / Terraform-hub

Terraform-hub Public
forked from Devendra-419/Terraform-hub

main · 2 Branches · 0 Tags

This branch is up to date with Devendra-419/Terraform-hub:main .

gowtham4s Update variables.tf · 18dbcf0 · 16 hours ago · 4 Commits

Jenkinsfile	Update Jenkinsfile	16 hours ago
main.tf	Add files via upload	16 hours ago
output.tf	Add files via upload	16 hours ago
provider.tf	Add files via upload	16 hours ago
variables.tf	Update variables.tf	16 hours ago

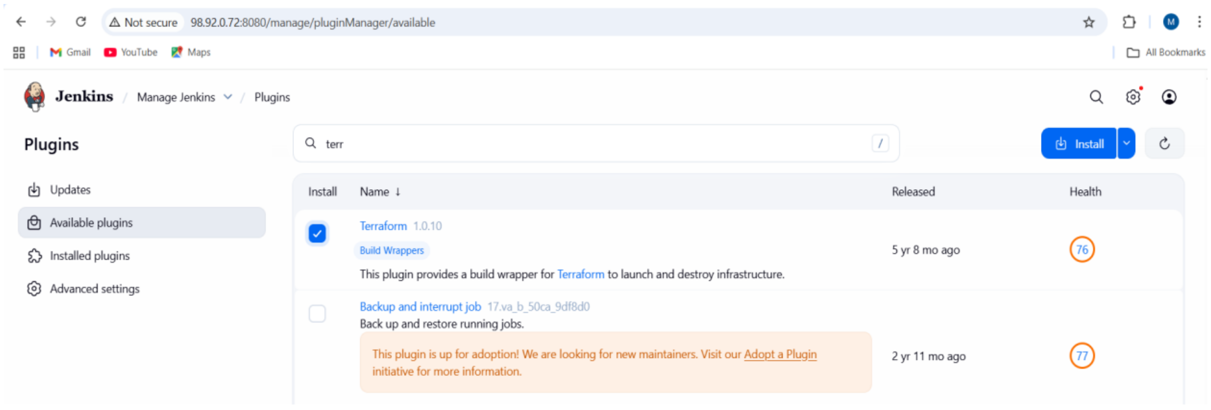
README

About
No description, website, or topi
Activity
0 stars
0 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Go to manage Jenkins and click on plugins, install terraform plugin.


98.92.0.72:8080/manage/pluginManager/available

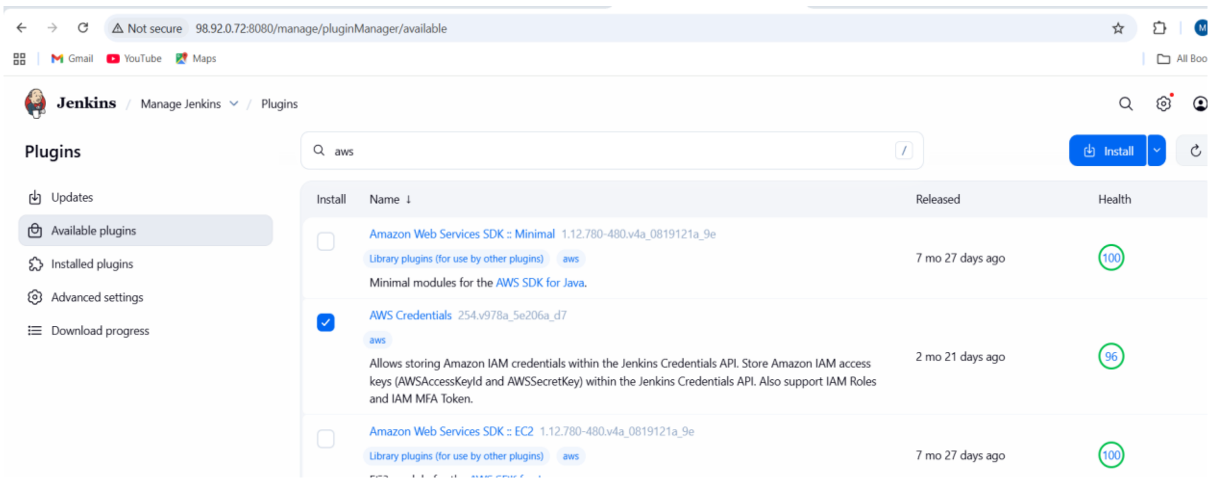
Jenkins / Manage Jenkins / Plugins

Plugins
Updates
Available plugins
Installed plugins
Advanced settings

terr

Install	Name ↓	Released	Health
✓	Terraform 1.0.10 Build Wrappers This plugin provides a build wrapper for Terraform to launch and destroy infrastructure.	5 yr 8 mo ago	76
	Backup and interrupt job 17.va_b_50ca_9df8d0 Back up and restore running jobs. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	2 yr 11 mo ago	77

Install another plugin called aws credentials.


Not secure 98.92.0.72:8080/manage/pluginManager/available

Jenkins / Manage Jenkins / Plugins

Plugins

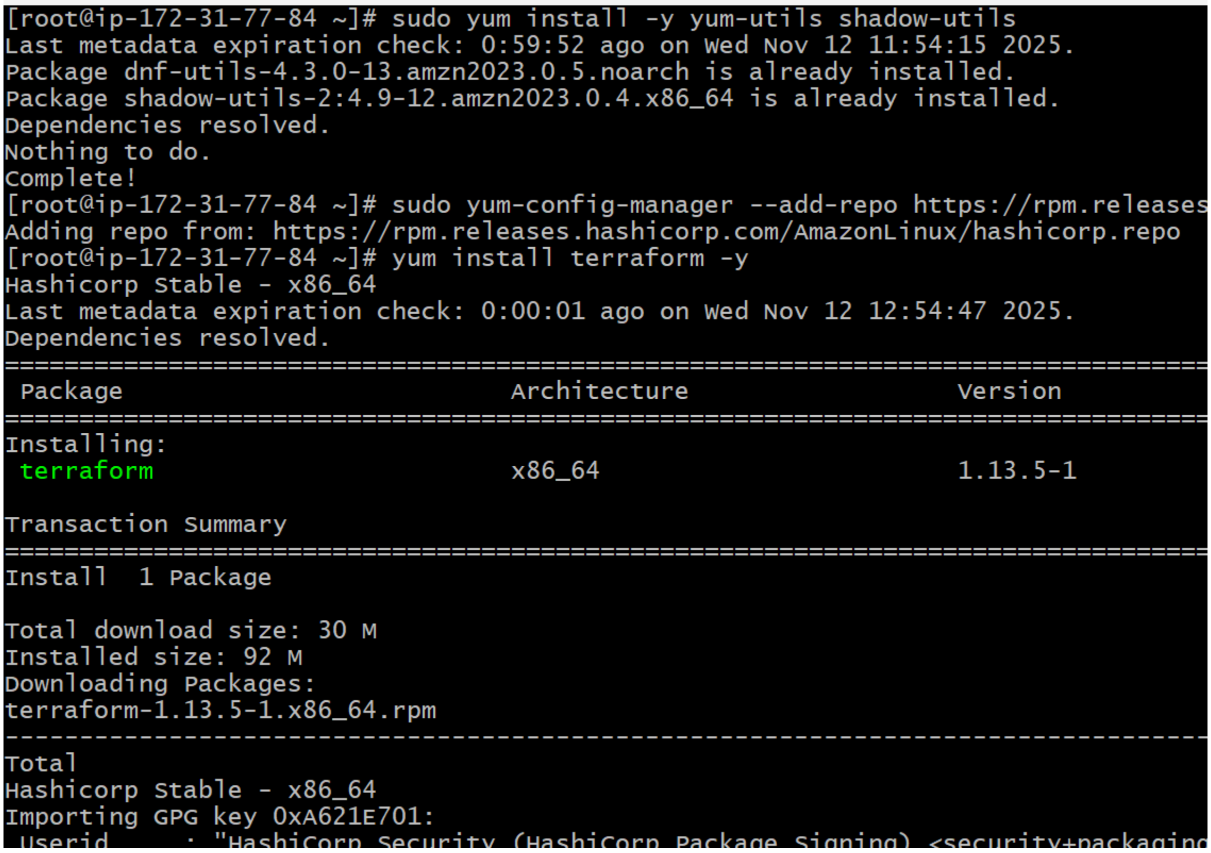
Updates
Available plugins
Installed plugins
Advanced settings
Download progress

aws

Install	Name ↓	Released	Health
☐	Amazon Web Services SDK :: Minimal 1.12.780-480.v4a_0819121a_9e Library plugins (for use by other plugins) aws Minimal modules for the AWS SDK for Java.	7 mo 27 days ago	100
✓	AWS Credentials 254.v978a_5e206a_d7 aws Allows storing Amazon IAM credentials within the Jenkins Credentials API. Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) within the Jenkins Credentials API. Also support IAM Roles and IAM MFA Token.	2 mo 21 days ago	96
☐	Amazon Web Services SDK :: EC2 1.12.780-480.v4a_0819121a_9e Library plugins (for use by other plugins) aws	7 mo 27 days ago	100

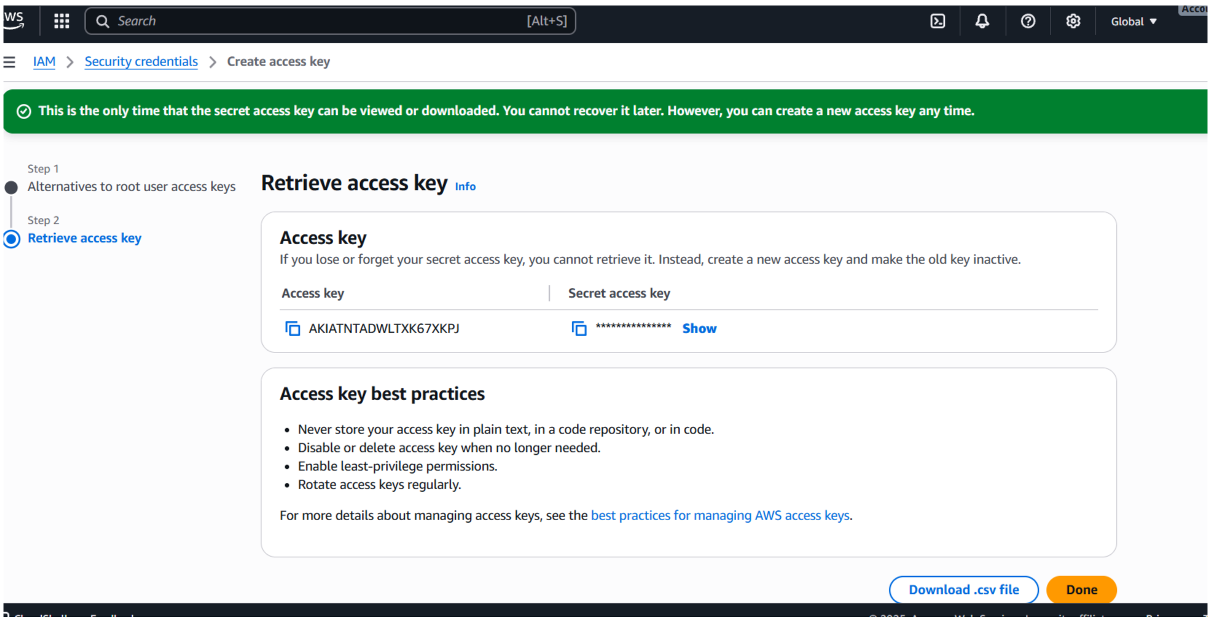
Go to Jenkins server and install terraform by using this commands.

- yum install -y yum-utils shadow-utils
- yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
- yum install terraform



```
[root@ip-172-31-77-84 ~]# sudo yum install -y yum-utils shadow-utils
Last metadata expiration check: 0:59:52 ago on Wed Nov 12 11:54:15 2025.
Package dnf-utils-4.3.0-13.amzn2023.0.5.noarch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-77-84 ~]# sudo yum-config-manager --add-repo https://rpm.releases
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
[root@ip-172-31-77-84 ~]# yum install terraform -y
Hashicorp Stable - x86_64
Last metadata expiration check: 0:00:01 ago on Wed Nov 12 12:54:47 2025.
Dependencies resolved.
================================================================================
 Package                          Architecture                 Version
================================================================================
Installing:
 terraform                        x86_64                       1.13.5-1

Transaction Summary
================================================================================
Install  1 Package

Total download size: 30 M
Installed size: 92 M
Downloading Packages:
terraform-1.13.5-1.x86_64.rpm
--------------------------------------------------------------------------------
Total
Hashicorp Stable - x86_64
Importing GPG key 0xA621E701:
 Userid     : "HashiCorp Security (HashiCorp Package Signing) <security+packaging
```

Create access key and secret key in aws credentials



IAM > Security credentials > Create access key

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Retrieve access key Info

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIATNTADWLTXK67XKPJ	*************** Show

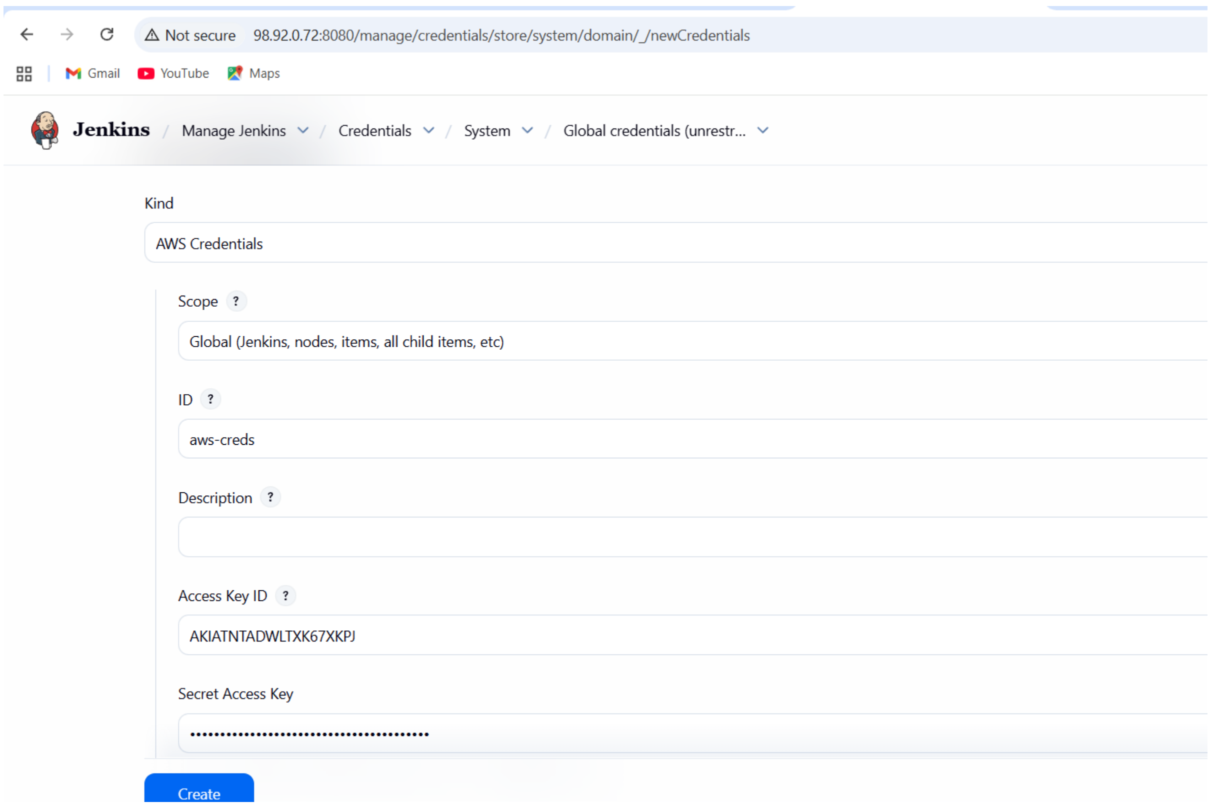
Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

Give the credentials in Jenkins

Go to manage Jenkins , credentials,global credentials select aws credentials and paste access key and secret key.


Not secure 98.92.0.72:8080/manage/credentials/store/system/domain/_/newCredentials
Gmail
YouTube
Maps
Jenkins
Manage Jenkins
Credentials
System
Global credentials (unrestr...

Kind
AWS Credentials

Scope
Global (Jenkins, nodes, items, all child items, etc)

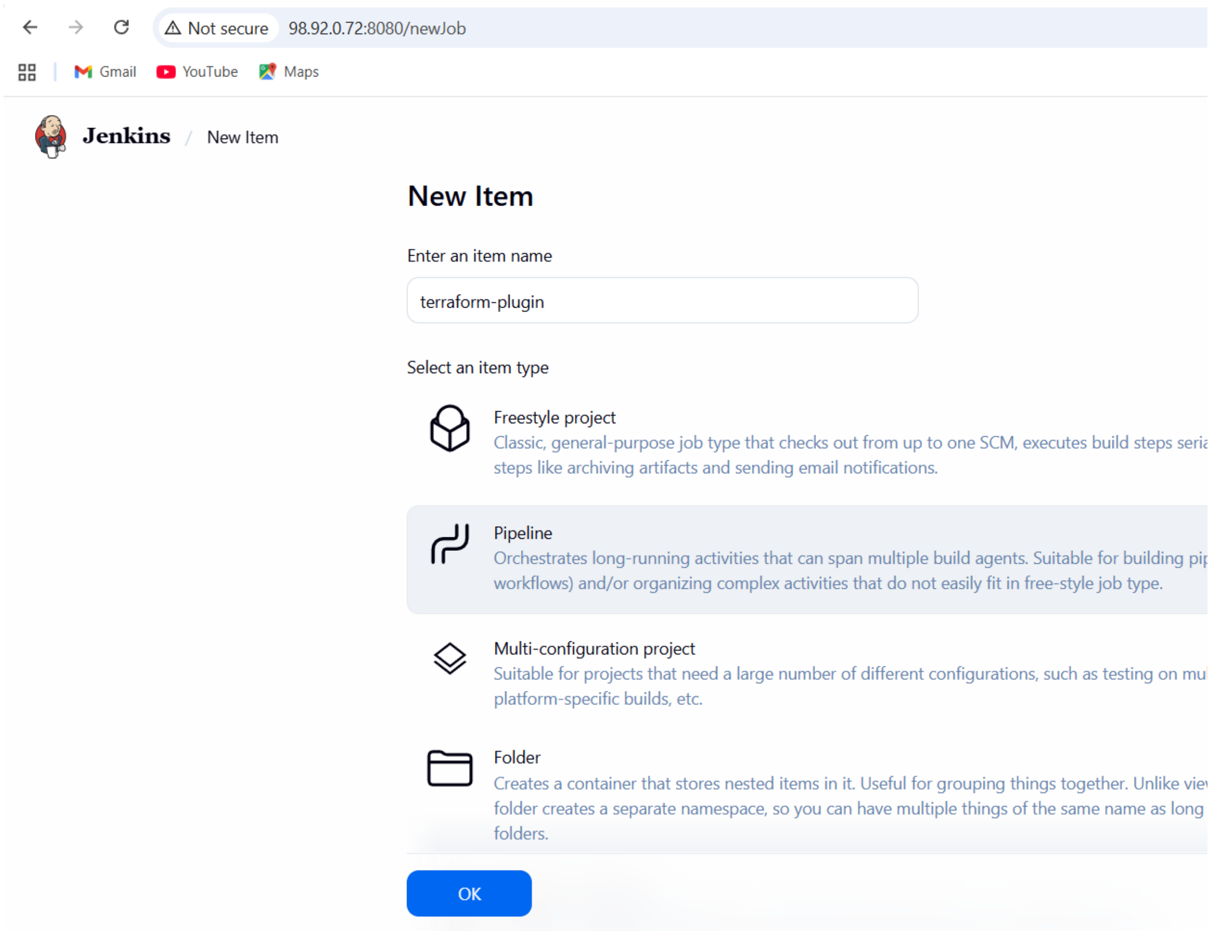
ID
aws-creds

Description

Access Key ID
AKIATNTADWLTXK67XKPJ

Secret Access Key
••

Create a new item and select pipeline.



New Item

Enter an item name
terraform-plugin

Select an item type

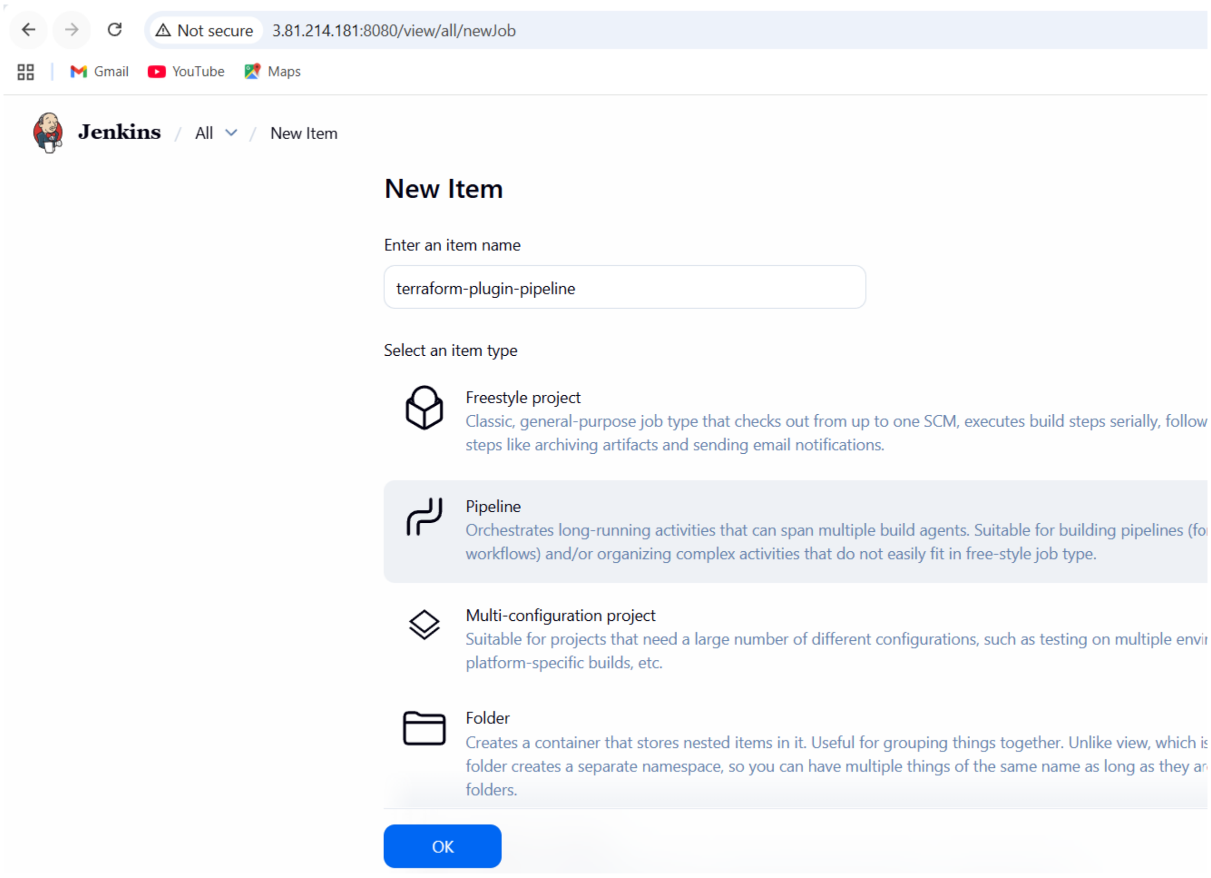
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps seria
steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pi
workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on mu
platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike vie
folder creates a separate namespace, so you can have multiple things of the same name as long
folders.

Select git in configure and give repository URL and branch click on build now.



Not secure 3.81.214.181:8080/view/all/newJob

Gmail YouTube Maps

Jenkins / All / New Item

New Item

Enter an item name

terraform-plugin-pipeline

Select an item type

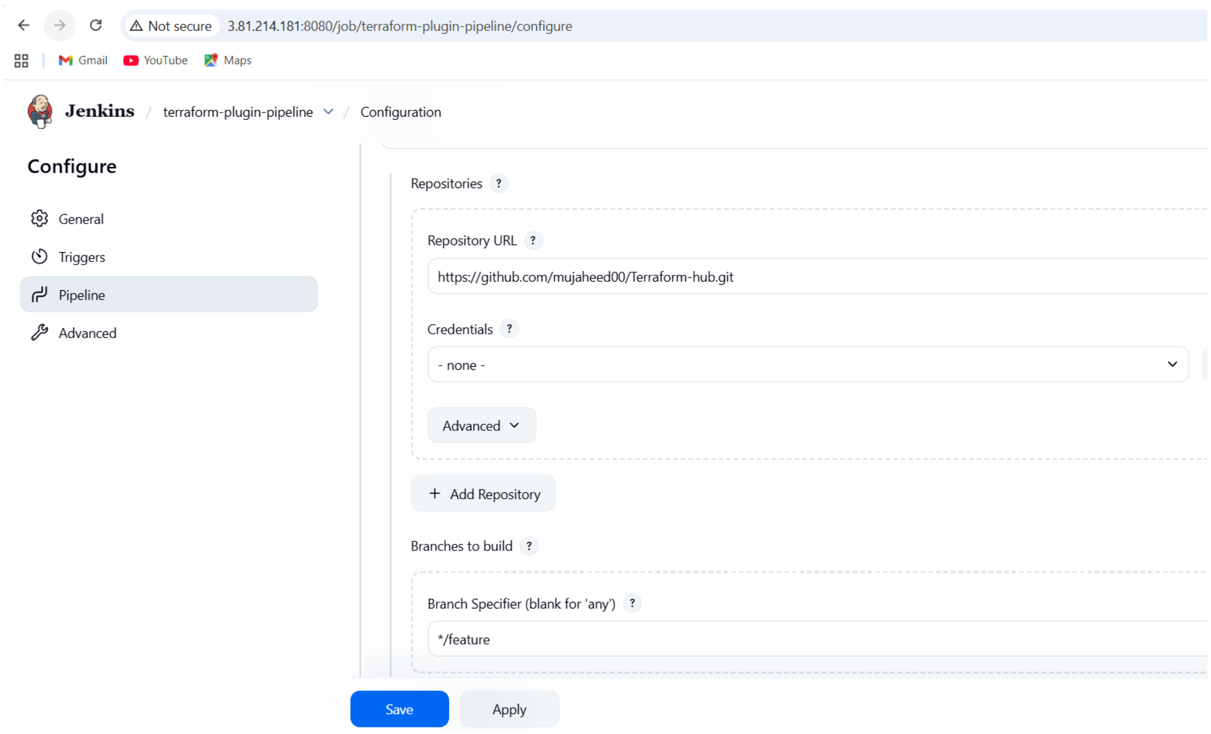
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, follow
steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (fo
workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple envi
platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is
folder creates a separate namespace, so you can have multiple things of the same name as long as they a
folders.

OK



Not secure 3.81.214.181:8080/job/terraform-plugin-pipeline/configure

Gmail YouTube Maps

Jenkins / terraform-plugin-pipeline / Configuration

Configure

General
Triggers
Pipeline
Advanced

Repositories

Repository URL

https://github.com/mujaheed00/Terraform-hub.git

Credentials

- none -

Advanced

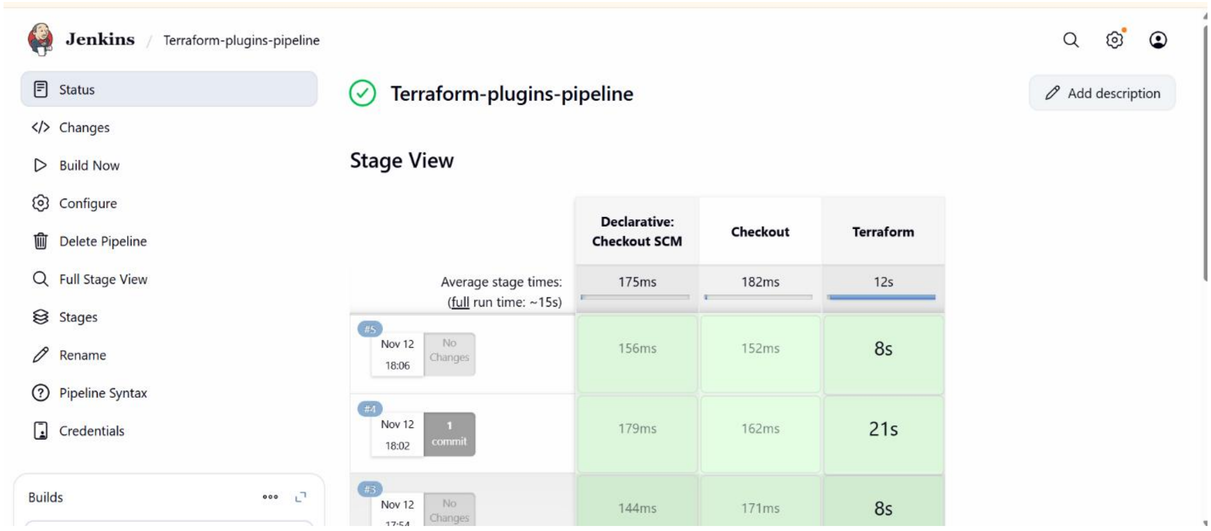
+ Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/feature

Save Apply

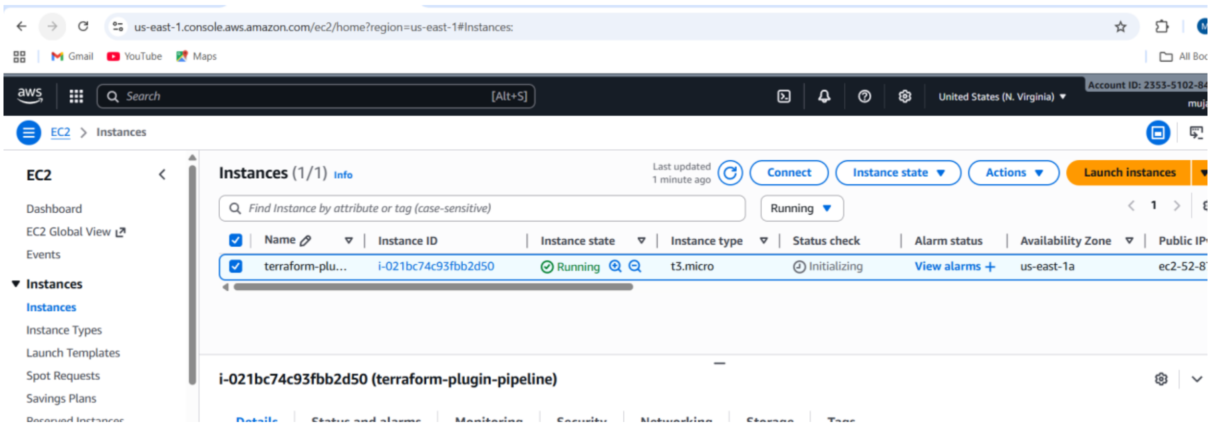


Terraform-plugins-pipeline

Stage View

	Declarative: Checkout SCM	Checkout	Terraform
Average stage times: (full run time: ~15s)	175ms	182ms	12s
#5 Nov 12 18:06 No Changes	156ms	152ms	8s
#4 Nov 12 18:02 1 commit	179ms	162ms	21s
#3 Nov 12 17:54 No Changes	144ms	171ms	8s

It will automatically create an instance.



Instances (1/1) Info

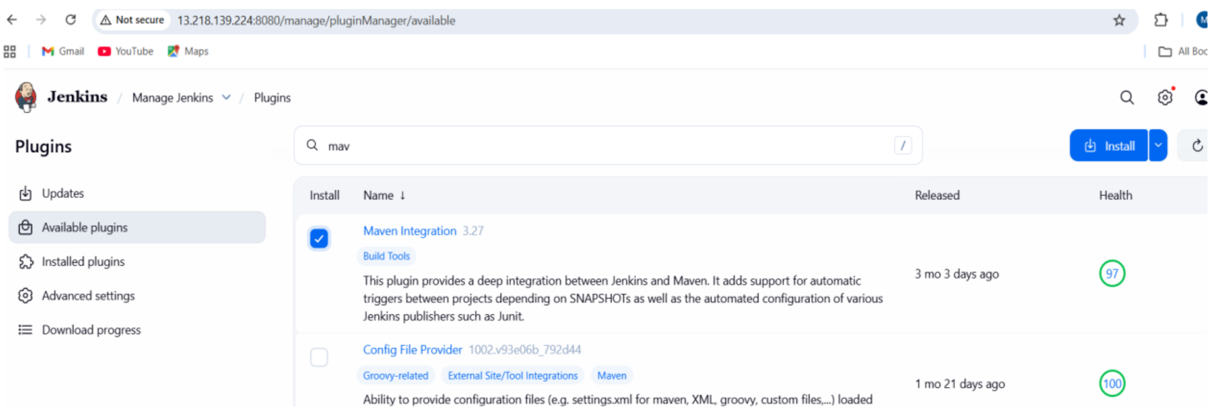
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
✓	terraform-plu...	i-021bc74c93fbb2d50	Running	t3.micro	Initializing	View alarms +	us-east-1a	ec2-52-8

i-021bc74c93fbb2d50 (terraform-plugin-pipeline)

4. Create one Jenkins job using Maven Project for the code below with two stages:

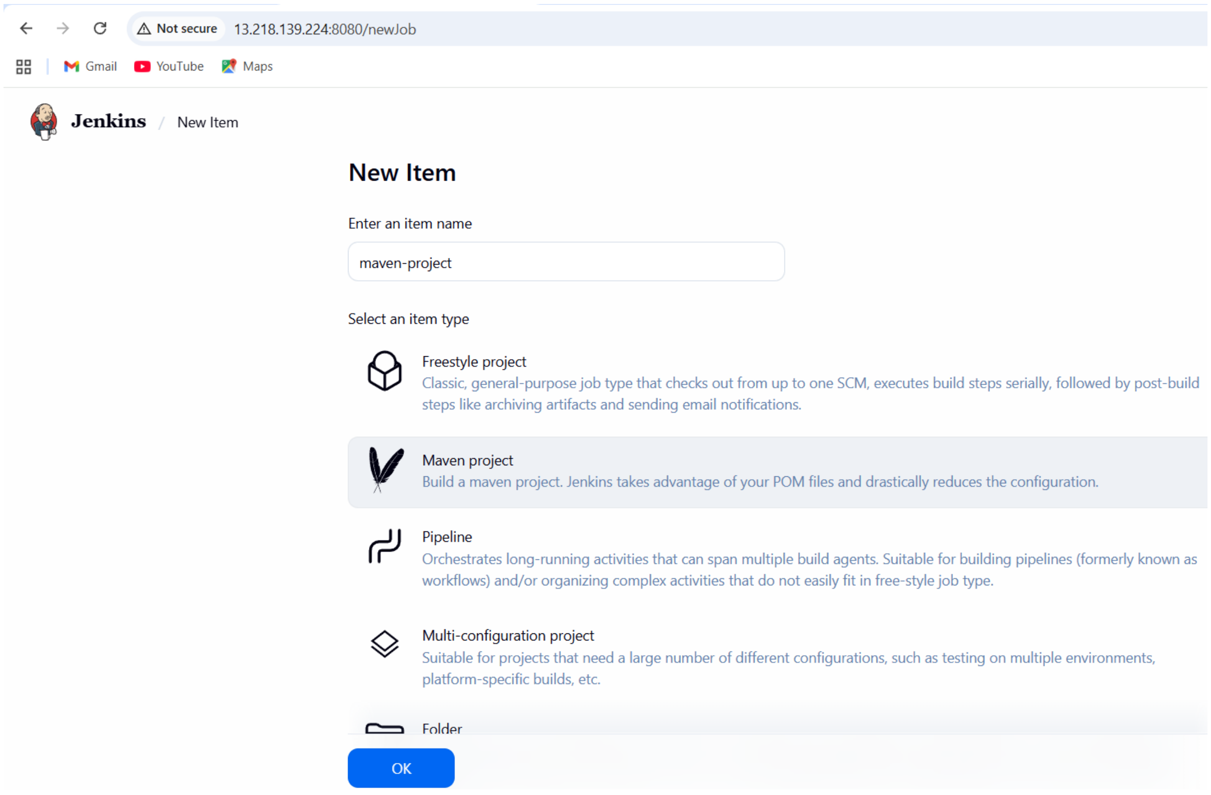
- **Stage 1: Git clone**
- **Stage 2: Maven Compilation Code: https://github.com/betawins/java-Working-app.git**

go to manage Jenkins, plugins,install maven integration plugin.



Install	Name ↓	Released	Health
✓	Maven Integration 3.27 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.	3 mo 3 days ago	97
	Config File Provider 1002.v93e06b_792d44 Groovy-related External Site/Tool Integrations Maven Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded	1 mo 21 days ago	100

Click on item and create it by any name and select type as maven.



New Item

Enter an item name

maven-project

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

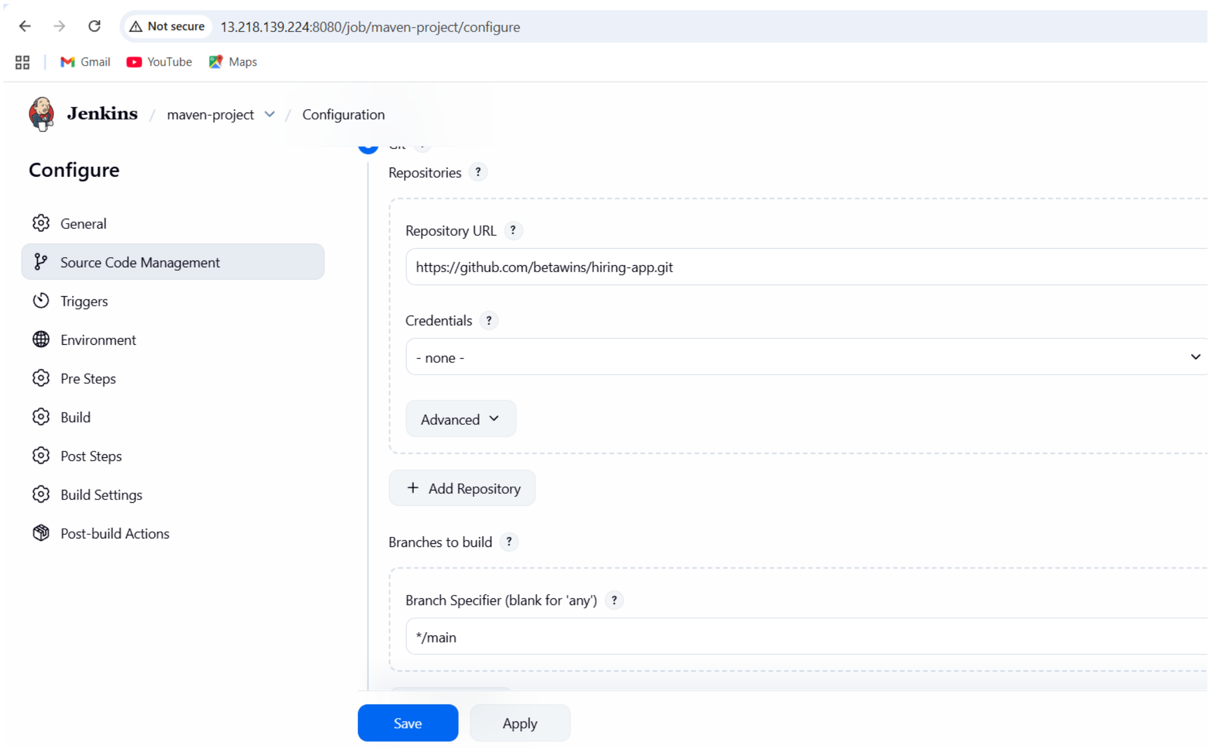
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Go to that job select git and provide url and branch as main clik on save.



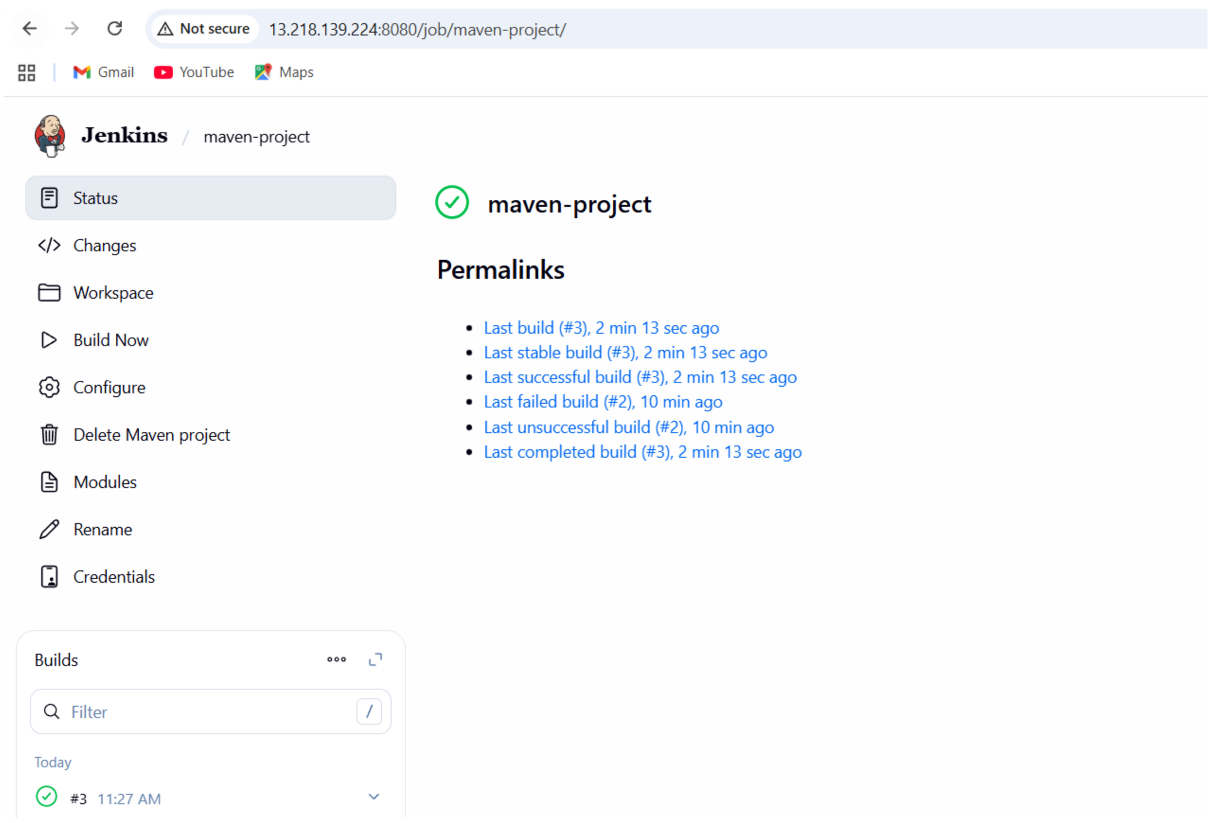
Repositories

Repository URL: https://github.com/betawins/hiring-app.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */main



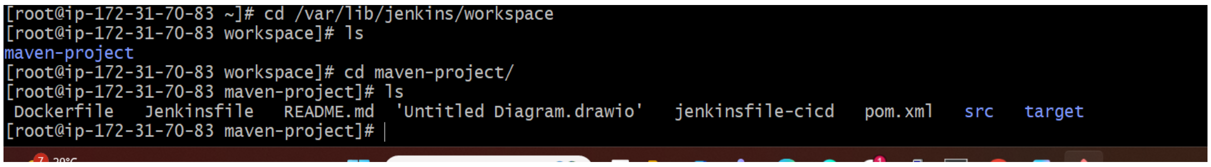
maven-project

Permalinks

- Last build (#3), 2 min 13 sec ago
- Last stable build (#3), 2 min 13 sec ago
- Last successful build (#3), 2 min 13 sec ago
- Last failed build (#2), 10 min ago
- Last unsuccessful build (#2), 10 min ago
- Last completed build (#3), 2 min 13 sec ago

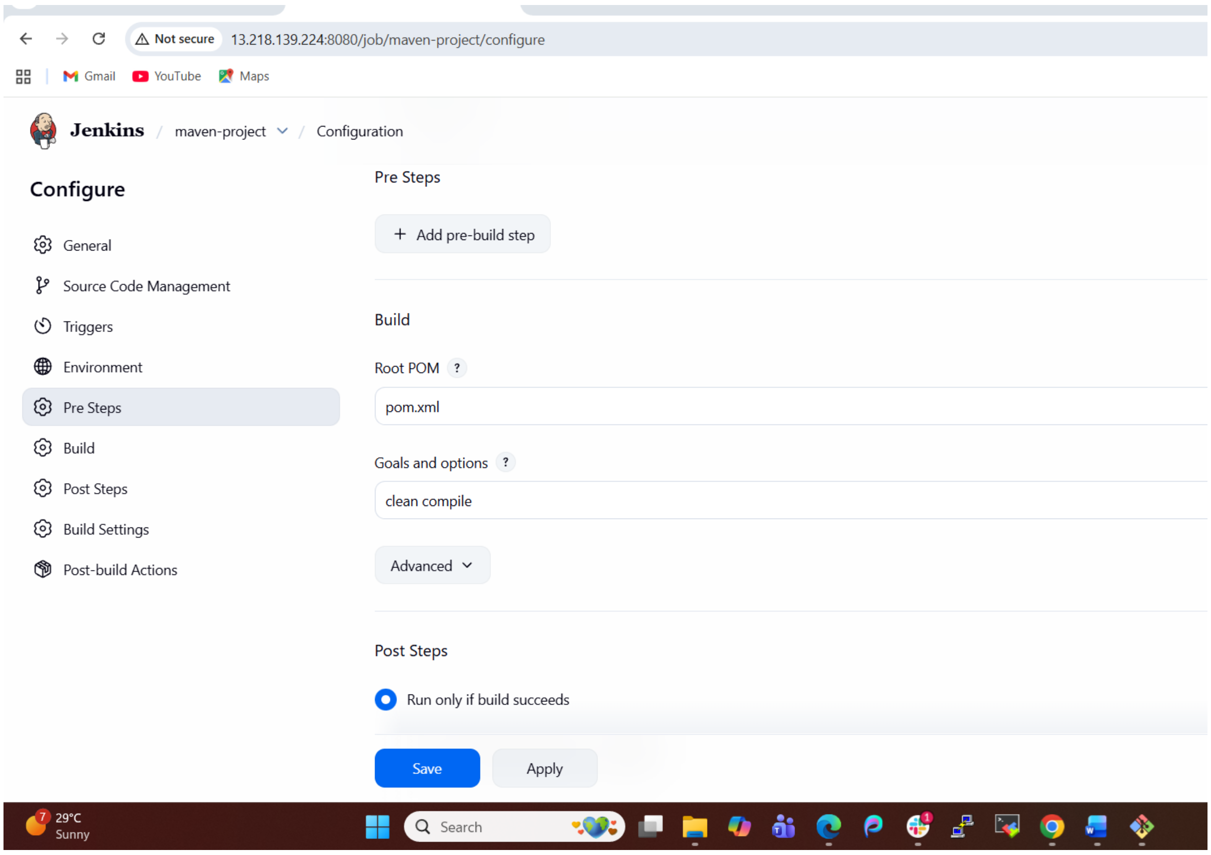
#3 11:27 AM

Git clone completed.



```
[root@ip-172-31-70-83 ~]# cd /var/lib/jenkins/workspace
[root@ip-172-31-70-83 workspace]# ls
maven-project
[root@ip-172-31-70-83 workspace]# cd maven-project/
[root@ip-172-31-70-83 maven-project]# ls
 Dockerfile   Jenkinsfile   README.md  'Untitled Diagram.drawio'   jenkinsfile-cicd   pom.xml   src   target
[root@ip-172-31-70-83 maven-project]#
```

Click on configure go to build and give pom.xml in root POM,in goals give clean compile.

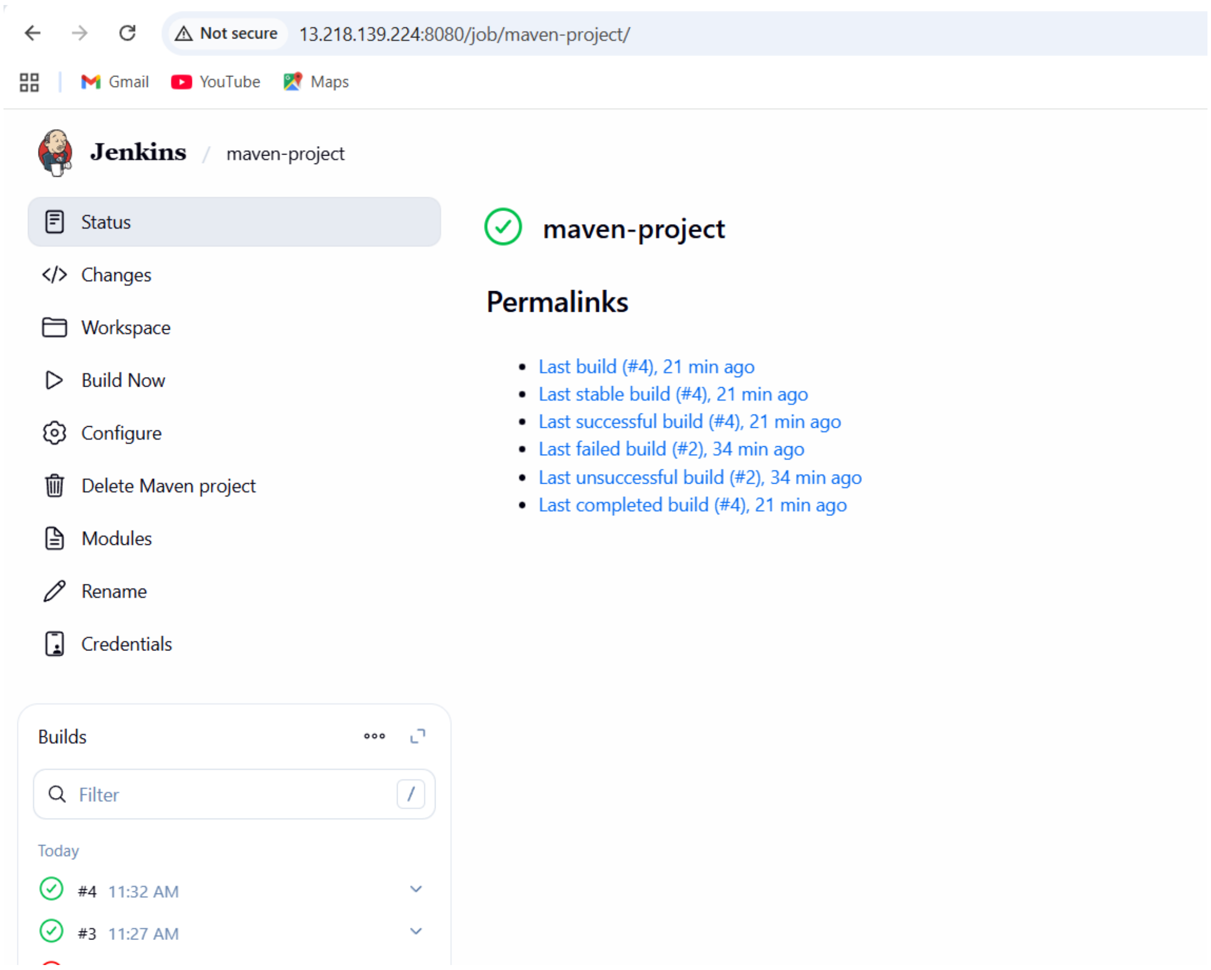


13.218.139.224:8080/job/maven-project/configure

Root POM: pom.xml

Goals and options: clean compile

Run only if build succeeds



maven-project

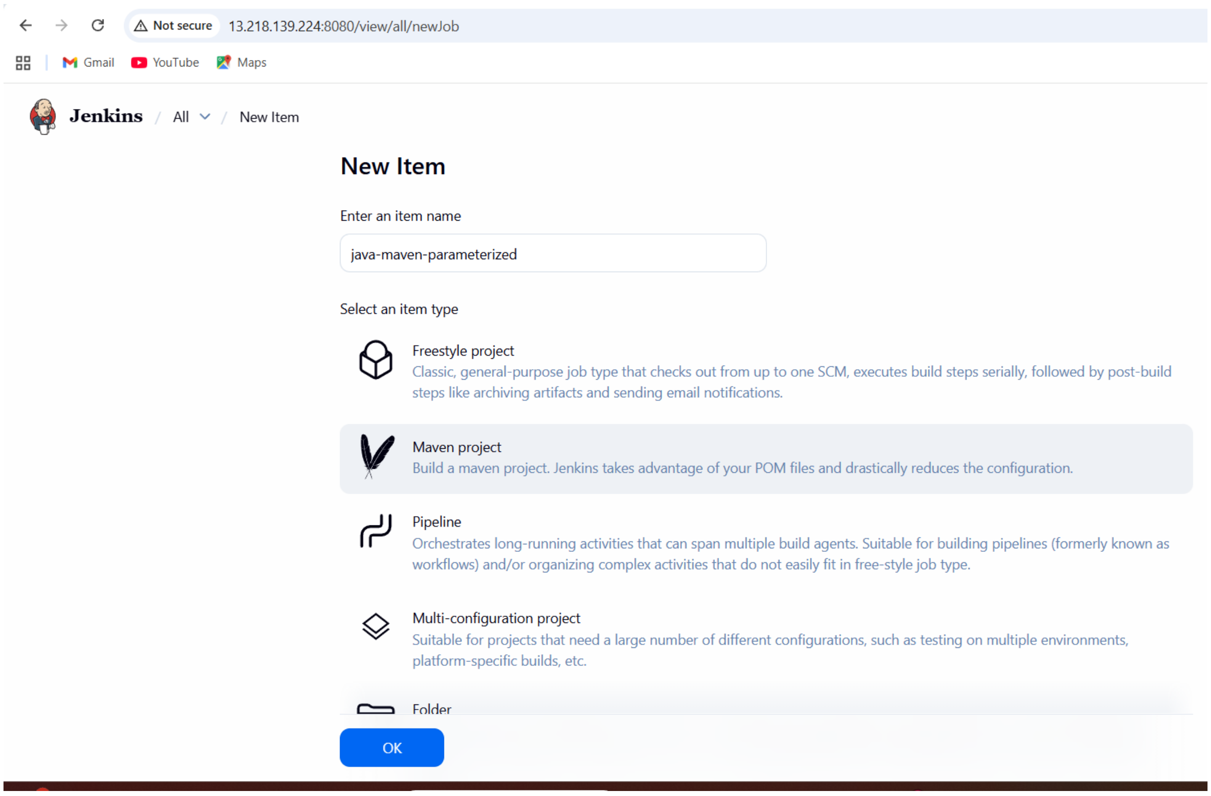
Permalinks

- Last build (#4), 21 min ago
- Last stable build (#4), 21 min ago
- Last successful build (#4), 21 min ago
- Last failed build (#2), 34 min ago
- Last unsuccessful build (#2), 34 min ago
- Last completed build (#4), 21 min ago

5. Use the same code and create a parameterized job in Jenkins with:

- **Stage 1: Git clone**
- **Stage 2: Maven Compilation Code: https://github.com/betawins/java-Working-app.git**

Click on new item give name as java-maven-parameterized and select type as maven parameterized.


Not secure 13.218.139.224:8080/view/all/newJob

Jenkins / All / New Item

New Item

Enter an item name

java-maven-parameterized

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

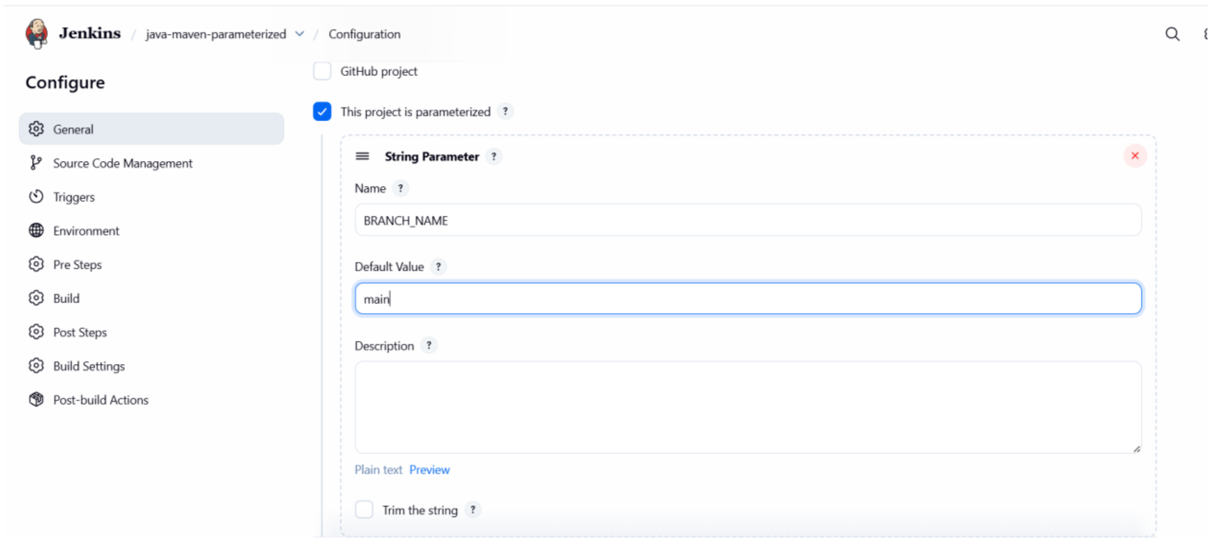
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

OK

Go to job select this job is parameterized and select parameter as string.

- Give name as BRANCH_NAME
- Default value as main


Jenkins / java-maven-parameterized / Configuration

Configure

General
Source Code Management
Triggers
Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions

GitHub project

This project is parameterized ✓

String Parameter

Name: BRANCH_NAME

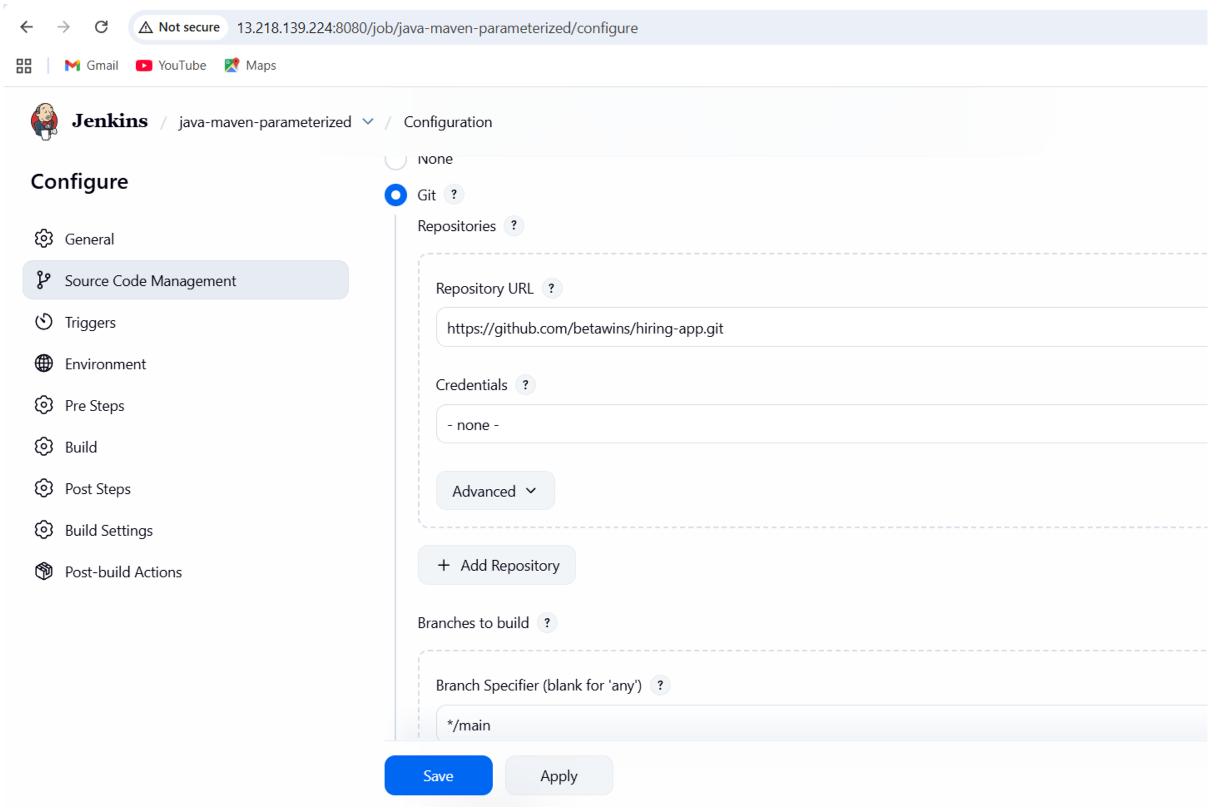
Default Value: main

Description

Plain text Preview

Trim the string

Give git URL and select branch as main.


13.218.139.224:8080/job/java-maven-parameterized/configure

Jenkins / java-maven-parameterized / Configuration

Source Code Management: None [✓ Git]

Repositories

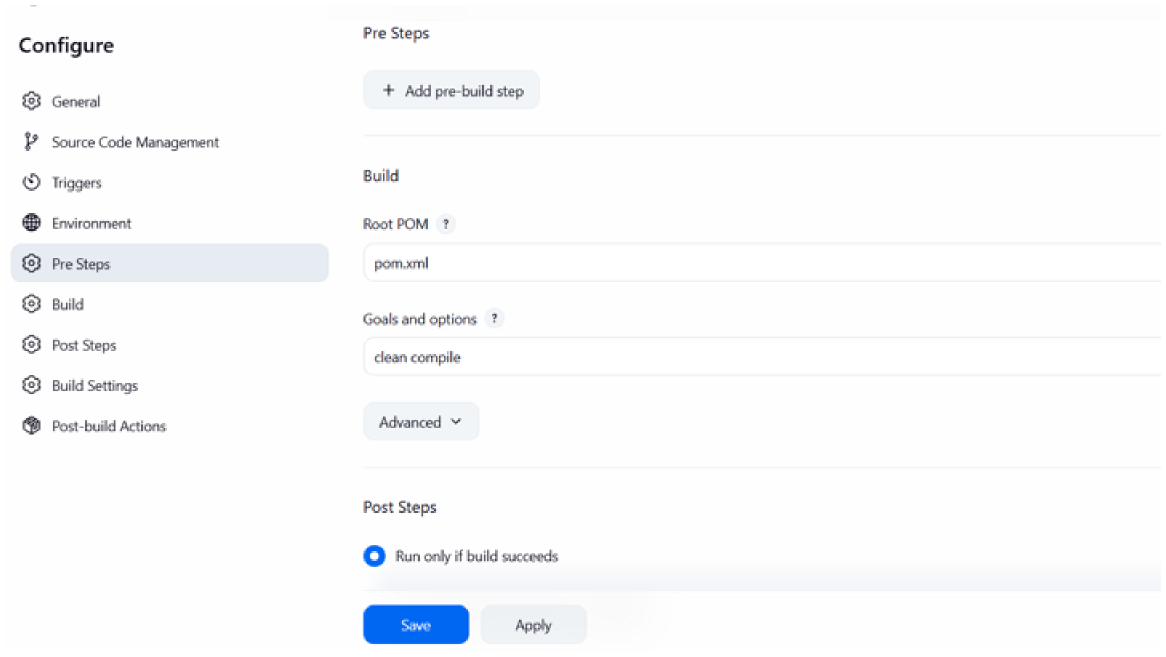
Repository URL: https://github.com/betawins/hiring-app.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */main

Go to build give pom.xml at ROOT POM and in goals and options give clean compile.

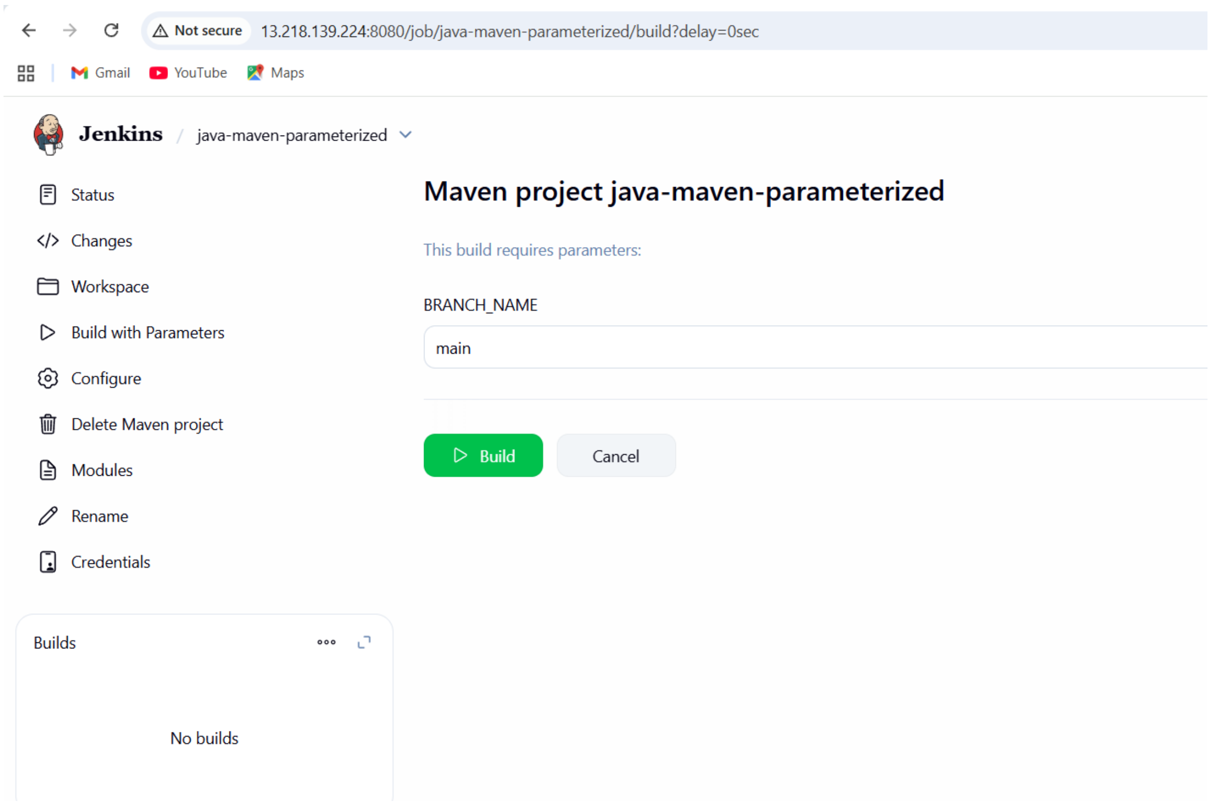

Build

Root POM: pom.xml

Goals and options: clean compile

Post Steps: [✓ Run only if build succeeds]

Click on save and click on build with parameters.


13.218.139.224:8080/job/java-maven-parameterized/build?delay=0sec

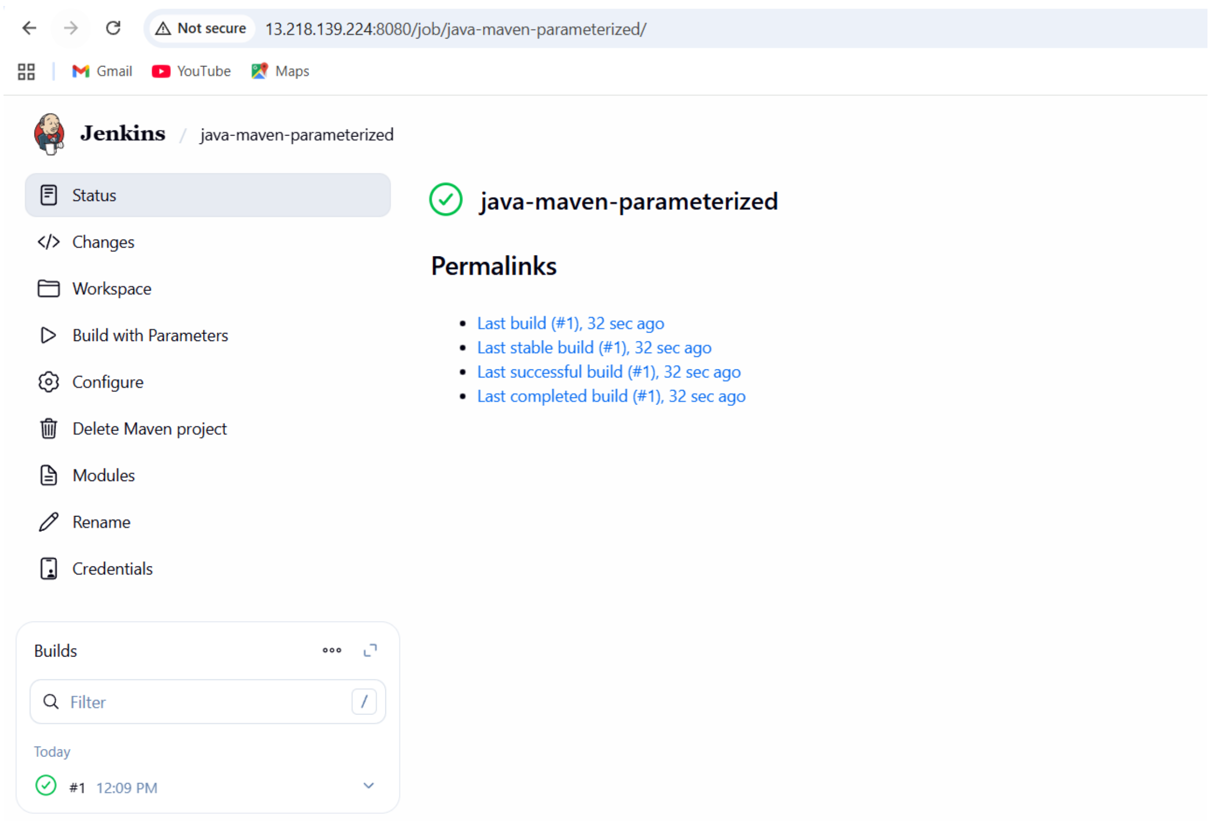
Jenkins / java-maven-parameterized

Maven project java-maven-parameterized

This build requires parameters:

BRANCH_NAME

main


13.218.139.224:8080/job/java-maven-parameterized/

Jenkins / java-maven-parameterized

java-maven-parameterized

Permalinks

- Last build (#1), 32 sec ago
- Last stable build (#1), 32 sec ago
- Last successful build (#1), 32 sec ago
- Last completed build (#1), 32 sec ago

6. What are the global variables in Jenkins?

* In **Jenkins**, *global variables* are built-in environment variables and objects that are **available to every pipeline.**

BUILD_ID	The unique build ID (often same as BUILD_NUMBER).
BUILD_NUMBER	The current build number of the job.
BUILD_TAG	A unique tag like jenkins-${JOB_NAME}-${BUILD_NUMBER}.
BUILD_URL	URL of the current build in Jenkins.
JOB_NAME	Name of the current Jenkins job.
JOB_BASE_NAME	Short name of the job (last part of JOB_NAME).
JOB_URL	URL of the Jenkins job.
WORKSPACE	The workspace directory path for this job on the agent.
NODE_NAME	Name of the node/slave executing the build (master if local).

EXECUTOR_NUMBER	Identifies the executor number on the node.
JENKINS_HOME	Root directory of Jenkins installation.
JENKINS_URL	Base URL of the Jenkins master.
GIT_COMMIT	The Git commit hash currently checked out. *(if using Git SCM)*
GIT_BRANCH	The Git branch being built. *(if using Git SCM)*
GIT_URL	URL of the Git repository. *(if using Git SCM)*
CHANGE_ID, CHANGE_BRANCH, CHANGE_TARGET	Used in multibranch pipelines for pull requests.
BUILD_DISPLAY_NAME	Display name of the build (editable).
BUILD_USER	The user who triggered the build (if plugin installed).
NODE_LABELS	The labels assigned to the build node.

STAGE_NAME	Name of the current pipeline stage (when used inside stage).