

## 1. Watch the Terraform-03 video.

## 2. Execute the Script shown in the video.

main.tf

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

variables.tf

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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 = "7HezBW7bd0Y1pR6YLwsfkJ+ZtFWy1oq7/jElCL8Q==" -> (known after apply)
  ~ content_base64sha512 = "ehr4liFoh9zpisJTXj1rSv68P1B7Md+Ugnal0tm9Qp0f7CJBcC3wg+rfaFG0UV9gegTnAi/9bpMf1YuD0+VRAw==" -> (known after apply)
  ~ content_base64sha512 = "ehr4liFoh9zpisJTXj1rSv68P1B7Md+Ugnal0tm9Qp0f7CJBcC3wg+rfaFG0UV9gegTnAi/9bpMf1YuD0+VRAw==" -> (known after apply)
  ~ content_md5        = "56a456fe92c71aaeda680ba04a7da6139" -> (known after apply)
  ~ content_sha1       = "74ea3e30b14db581482aaa99214b7739f91cc8f4" -> (known after apply)
  ~ content_sha256     = "ec77b3055bfb6dda18d6947a6252f0b1f289ff99b455b2d68abbfe312508bf104" -> (known after apply)
  ~ content_sha512     = "7a147896216887dce922c2535e396b4afebc3f507b31df948276963ad9bd42939fec2241702df083efab6851b4515f607a04e7022ffd6e93
" -> (known after apply)
```

variables.tf

pets.txt

```
i love cats
```

```

main.tf
...
variables.tf

```

```

resource "random_pet" "my_pet" {
  prefix = "MR"
  separator = "."
  length = "1"
}

resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}

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 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 cats" -> "i hate cats" # forces replacement
    ~ content_base64sha256 = "AUfspEf+1lx3F1lvC2ju/W7R3Qj9RY8KCG0byJYhQs=" -> (known after apply)
    ~ content_base64sha512 = "FhjtsqYjx60AzxS5e960fJ743YdVv0dFNQq0zaX/+shYxoi9/2FsdM9xB3GCL14Ysf6/5rFT4IJOpDvL46tDQ==" -> (known after apply)
    ~ content_md5      = "765ab0286886d29ac7c8dae091b071de" -> (known after apply)
    ~ content_sha1     = "f140aba43cbc42844ecb543aeedcbbd239f626e7" -> (known after apply)
    ~ content_sha256   = "01416ca447fed65c7717596f7368d4fd6bf477423f5163c29c1b46f2258850b" -> (known after apply)
    ~ content_sha512   = "1618edb2a623ab1eb40335d2e5ef7ad1f27be3761d62fd1d14d42ad33697ffef21631a22f7fd8549d33dc41dc608b97862c"
  " -> (known after apply)
    ~ filename         = "pets.txt" -> "wild.txt" # forces replacement
    ~ id               = "f140aba43cbc42844ecb543aeedcbbd239f626e7" -> (known after apply)
}

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

~ content_sha256      = "01416ca447fed65c7717596f7368d4fd6bf477423f5163c29c1b46f2258850b" -> (known after apply)
~ content_sha512      = "1618edb2a623ab1eb40335d2e5ef7ad1f27be3761d62fd1d14d42ad33697ffef21631a22f7fd8549d33dc41dc608b97862c"
" -> (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>

```

```

variables.tf
var.content
var.filename
var.prefix

wild.txt
1 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 indicated by colors.
- 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 = "ArIzXRladMgp7fbxhPj8XsFtc8aB0aAlyD6v2ZBtGkU=" -> null
    - content_base64sha512 = "NqElRHS4mXrzftOUn+KZPLLtKjhnxO3snYXmEqCQmlwk8l2pmZGTnk7lMBTEDh2uxTJMnFuYUJH8

```

```

main.tf
resource "local_file" "my_pet" {
  filename = var.filename
  content  = var.content
}

resource "random_pet" "my_pet" {
  prefix   = "MR"
  separator = "."
  length   = "1"
}

variables.tf
variable "filename" {
}
variable "content" {
}
variable "prefix" {
}

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```
# 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 indicated by colors.
+ 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"
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

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

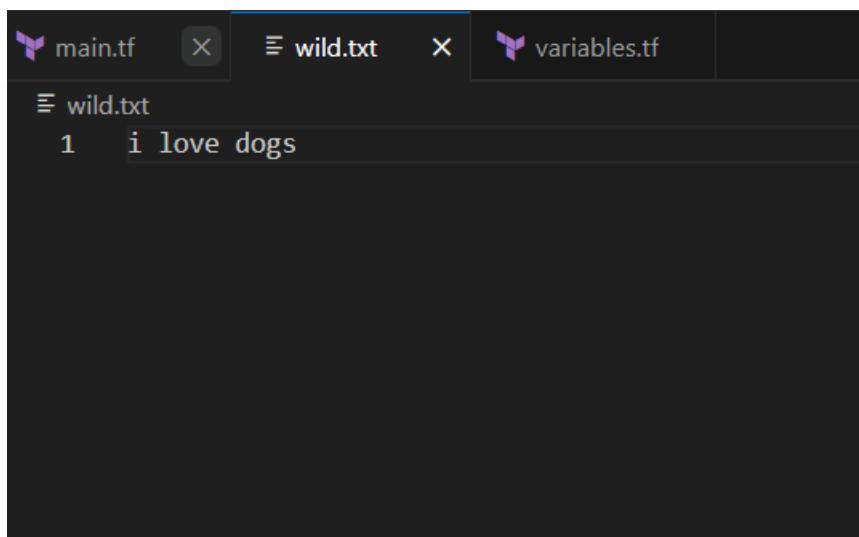
**Plan:** 2 to add, 0 to change, 0 to destroy.

**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.**



main.tf    X    wild.txt    X    variables.tf

wild.txt

```
1 i love dogs
```

- Giving 3 types of variables at the same time

```

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

variables.tf
variable "content" {
  default = "Hello World"
}
variable "filename" {
  default = "pets.txt"
}
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"

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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"

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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)
}
```

Ln 4, Col 1

PROBLEMS    OUTPUT    DEBUG CONSOLE    **TERMINAL**    PORTS

```
+ 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.

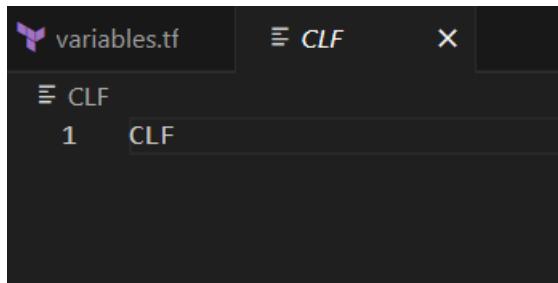
Enter a value: yes

```
local_file.my_pet: Creating...
local_file.my_pet: Creation complete after 0s [id=df0d2b6b2145603bd1fbe9e6464e44f7747]
```

**Apply complete! Resources: 1 added, 0 changed, 0 destroyed.**

Then CLF variable is created

- Because of precedence order flags variable will be executed.



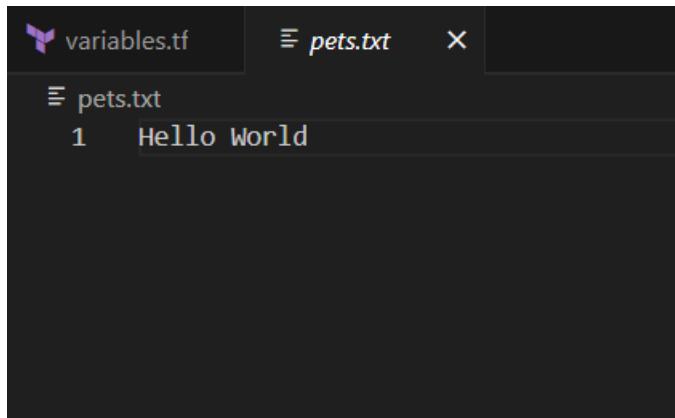
Destroy all and open from back

main.tf    variables.tf

```
main.tf > resource "random_pet" "my_pet"
1 resource "local_file" "my_pet" {
2   content  = var.content
3 }
4 resource "random_pet" "my_pet" {
5   prefix   = "MR"
6   separator = "."
7   length    = "1"
8 }
9 }
```

variables.tf    pets.txt

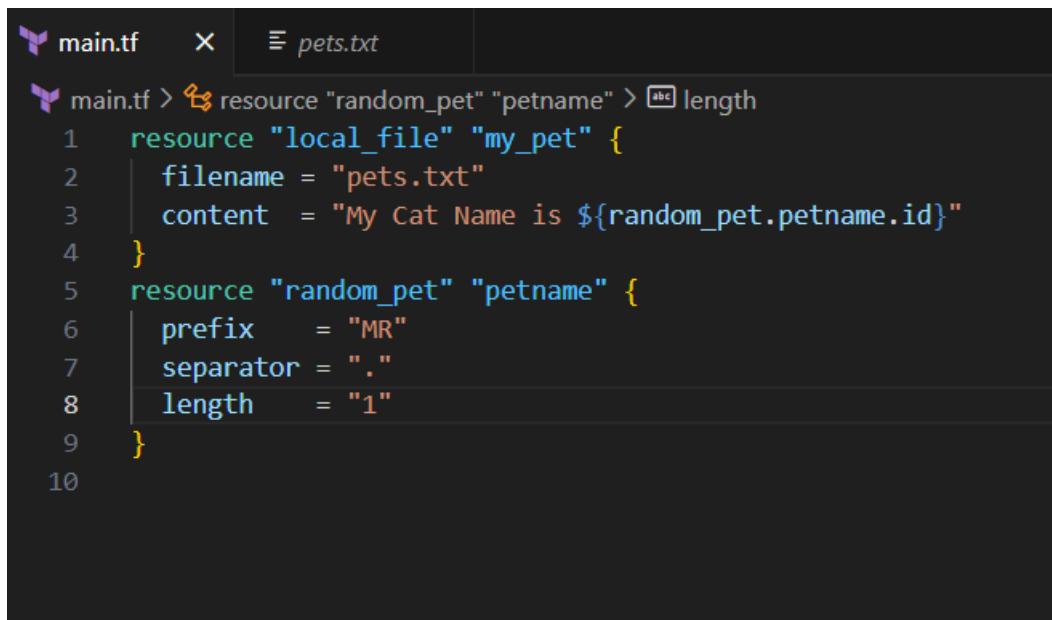
```
variables.tf > variable "content"
1 variable "filename" {
2   default = "pets.txt"
3 }
4 variable "content" {
5   default = "Hello World"
6 }
7 variable "prefix"{
8   default = "MR"
9 }
```



```
variables.tf      pets.txt      X  
pets.txt  
1 Hello World
```

## Destroy variable.tf

## Make changes in main.tf



```
main.tf      X      pets.txt  
main.tf > resource "random_pet" "petname" > length  
1   resource "local_file" "my_pet" {  
2     filename = "pets.txt"  
3     content  = "My Cat Name is ${random_pet.petname.id}"  
4   }  
5   resource "random_pet" "petname" {  
6     prefix    = "MR"  
7     separator = "."  
8     length    = "1"  
9   }  
10
```

```

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

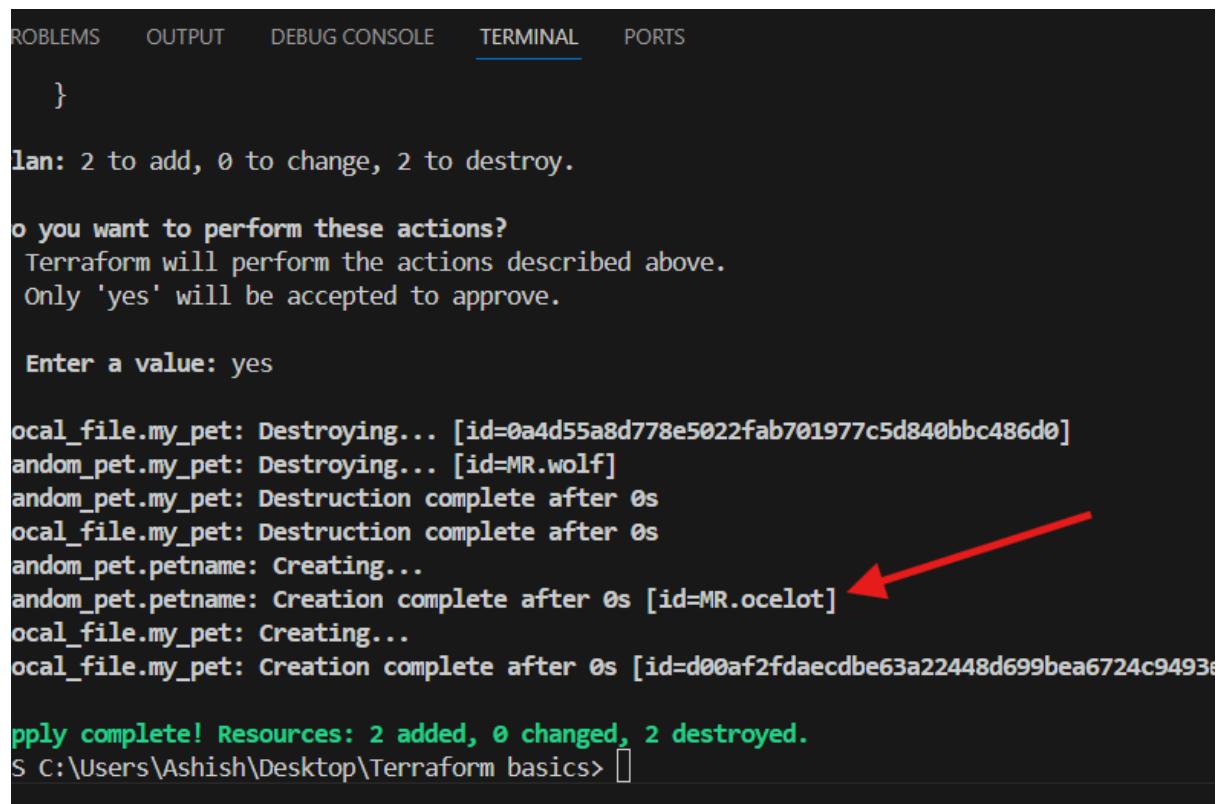
Terraform used the selected providers to generate the following execution plan.
+ create
- destroy
-/+ destroy and then create replacement

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     = "2c74fd17edaf80e8447b0d46741ee243b7eb74dd2149a0a

```

## A petname has created MR.ocelot



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

}

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

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

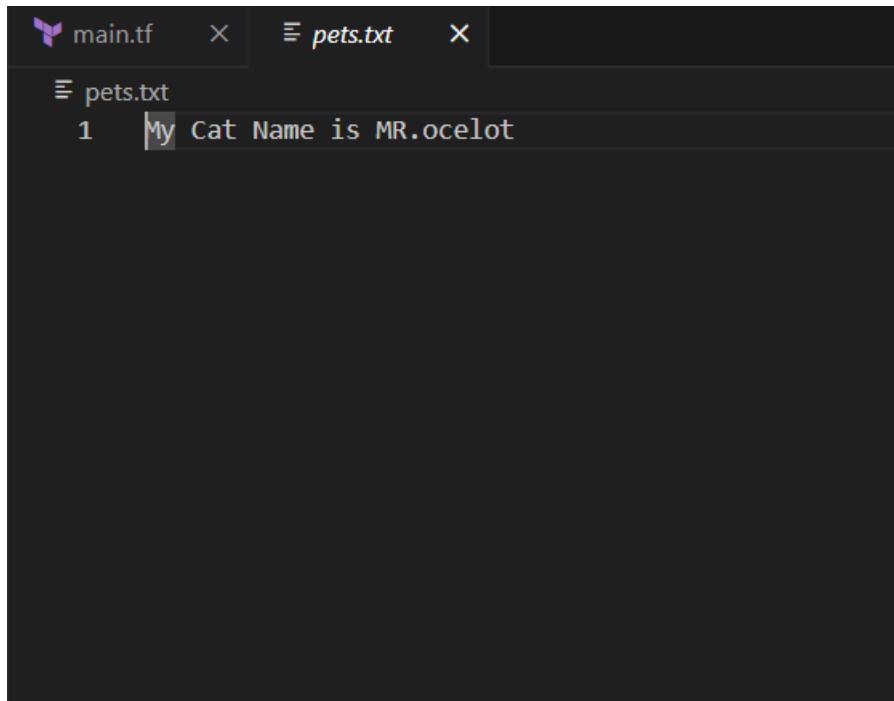
Enter a value: yes

local_file.my_pet: Destroying... [id=0a4d55a8d778e5022fab701977c5d840bbc486d0]
random_pet.my_pet: Destroying... [id=MR.wolf]
random_pet.my_pet: Destruction complete after 0s
local_file.my_pet: Destruction complete after 0s
random_pet.petname: Creating...
random_pet.petname: Creation complete after 0s [id=MR.ocelot] ←
local_file.my_pet: Creating...
local_file.my_pet: Creation complete after 0s [id=d00af2fdaecdbe63a22448d699bea6724c9493a

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

```

If I open pet.txt

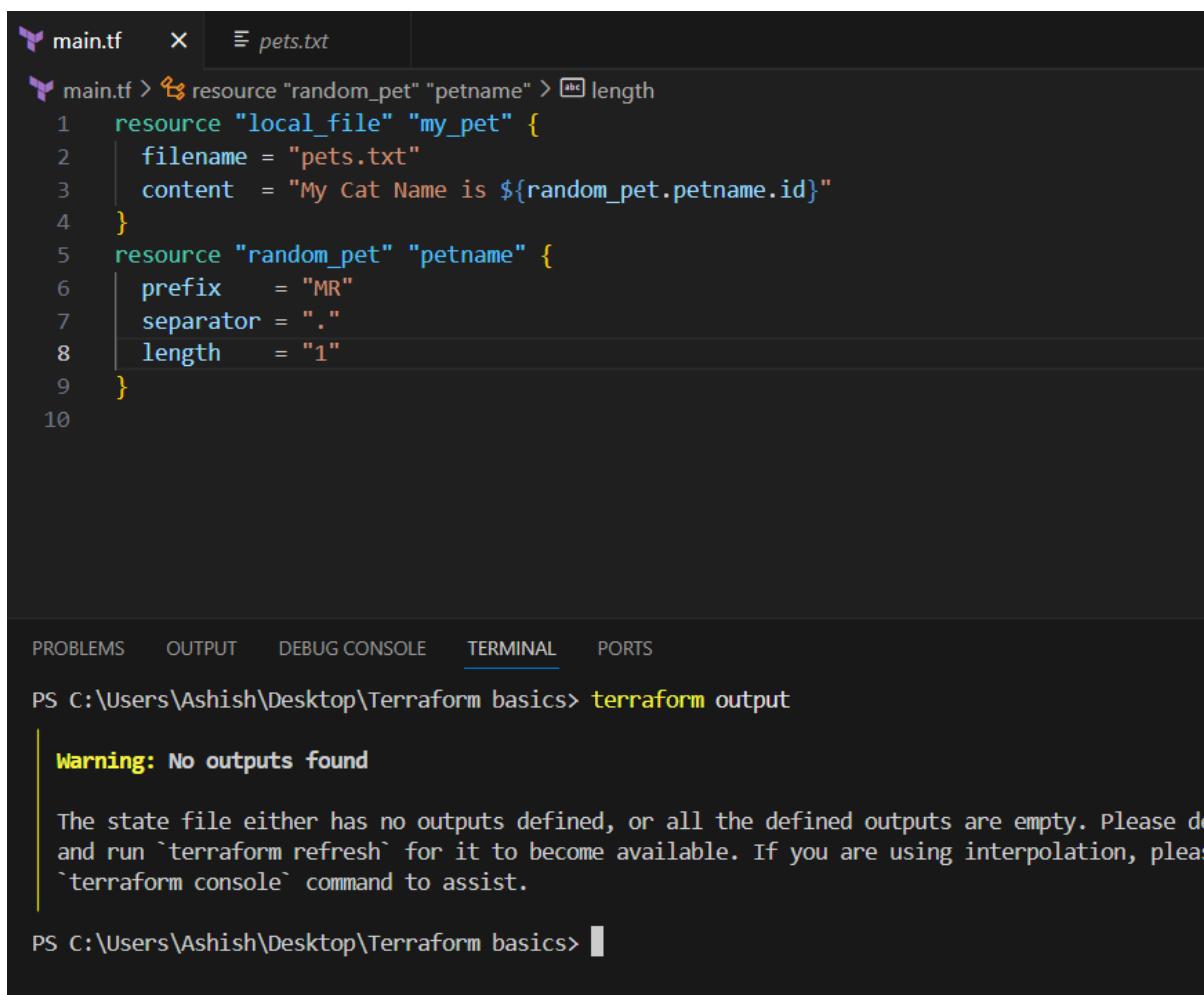


main.tf    X    pets.txt    X

≡ pets.txt

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

- **terraform output**



main.tf    X    pets.txt

main.tf > resource "random\_pet" "petname" > abc length

```
1 resource "local_file" "my_pet" {
2   filename = "pets.txt"
3   content  = "My Cat Name is ${random_pet.petname.id}"
4 }
5 resource "random_pet" "petname" {
6   prefix    = "MR"
7   separator = "."
8   length    = "1"
9 }
10
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

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> █

The screenshot shows a code editor with two tabs: 'main.tf' and 'pets.txt'. The 'main.tf' tab is active and contains the following Terraform code:

```
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
}
```

The screenshot shows the VS Code terminal window with the following output:

```
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 state.

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"
```

The screenshot shows a Visual Studio Code interface with a dark theme. In the top left, there are two tabs: 'main.tf' and 'pets.txt'. The 'main.tf' tab is active, displaying the following Terraform code:

```
1 resource "local_file" "my_pet" {
2   filename = "pets.txt"
3   content  = "My Cat Name is ${random_pet.petname.id}"
4 }
5 resource "random_pet" "petname" {
6   prefix    = "MR"
7   separator = "."
8   length    = "1"
9 }
10 output "Pet_name" {
11   value = random_pet.petname.id
12 }
13
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. Under the TERMINAL tab, the terminal output is shown:

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

- `terraform show`

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```
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>

**Terraform-hub** (Public)  
forked from [Devendra-419/Terraform-hub](#)

**Code** Pull requests Actions Projects Wiki Security Insights Settings

**main** 2 Branches 0 Tags Go to file Add file < Code

This branch is up to date with [Devendra-419/Terraform-hub:main](#).

**Commits**

Author	Commit Message	Time	
gowtham4s	Update variables.tf	18dbcf0 · 16 hours ago	
	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 topics  
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.

**Manage Jenkins** / Plugins

Available plugins

Install Name Released Health

Install	Name	Released	Health
<input checked="" type="checkbox"/>	Terraform 1.0.10	5 yr 8 mo ago	<span>76</span>
<input type="checkbox"/>	Build Wrappers		
This plugin provides a build wrapper for <a href="#">Terraform</a> to launch and destroy infrastructure.			
<input type="checkbox"/>	Backup and interrupt job 17.va_b_50ca_9df8d0	2 yr 11 mo ago	<span>77</span>
Back up and restore running jobs. This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.			

Install another plugin called aws credentials.

The screenshot shows the Jenkins plugin manager interface. The URL in the address bar is 98.92.0.72:8080/manage/pluginManager/available. A search bar at the top contains the text 'aws'. Below it, a table lists three Jenkins plugins:

Install	Name	Released	Health
<input type="checkbox"/>	Amazon Web Services SDK - Minimal 1.12.780-480.v4a_0819121a_9e Library plugins (for use by other plugins) aws	7 mo 27 days ago	100
<input checked="" type="checkbox"/>	AWS Credentials 254.v978a_5e206a_d7 aws	2 mo 21 days ago	96
<input type="checkbox"/>	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](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.hashicorp.com/AmazonLinux/hashicorp.repo
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@hashicorp.com>"
```

## Create access key and secret key in aws credentials

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**

**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
AKIATNTADWLTXKG7XKJP	XXXXXXXXXX <a href="#">Show</a>

**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](#).

[Download .csv file](#) [Done](#)

## 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**

## Create a new item and select pipeline.

← → ⌛ Not secure 98.92.0.72:8080/newJob

Gmail YouTube Maps

 Jenkins / New Item

### 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 serially like archiving artifacts and sending email notifications.

 Pipeline  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (pipelines) 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 platforms or performing platform-specific builds, etc.

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

**OK**

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

New Item

Enter an item name

terraform-plugin-pipeline

Select an item type

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

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

OK

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

The screenshot shows the Jenkins Pipeline interface for the 'Terraform-plugins-pipeline'. On the left, there's a sidebar with various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, Pipeline Syntax, and Credentials. The main area is titled 'Stage View' and displays a grid of three stages. The columns are labeled 'Declarative: Checkout SCM', 'Checkout', and 'Terraform'. Below the columns, it says 'Average stage times: (full run time: ~15s)'. The first stage has a timestamp of Nov 12 18:06 and a status of 'No Changes'. The second stage has a timestamp of Nov 12 18:02 and a status of '1 commit'. The third stage has a timestamp of Nov 12 17:48 and a status of 'No Changes'. The 'Terraform' column shows times of 12s, 8s, and 8s respectively.

It will automatically create an instance.

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for Dashboard, EC2 Global View, Events, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, and Documented Instances. The main content area shows a table of instances. One instance is selected, showing details: Name (terraform-plugin-pipeline), Instance ID (i-021bc74c93fb2d50), Instance state (Running), Instance type (t3.micro), Status check (Initializing), Alarm status (View alarms), Availability Zone (us-east-1a), and Public IP (ec2-52-8-177-248). The instance is also labeled as 'terraform-plugin-pipeline' at the bottom.

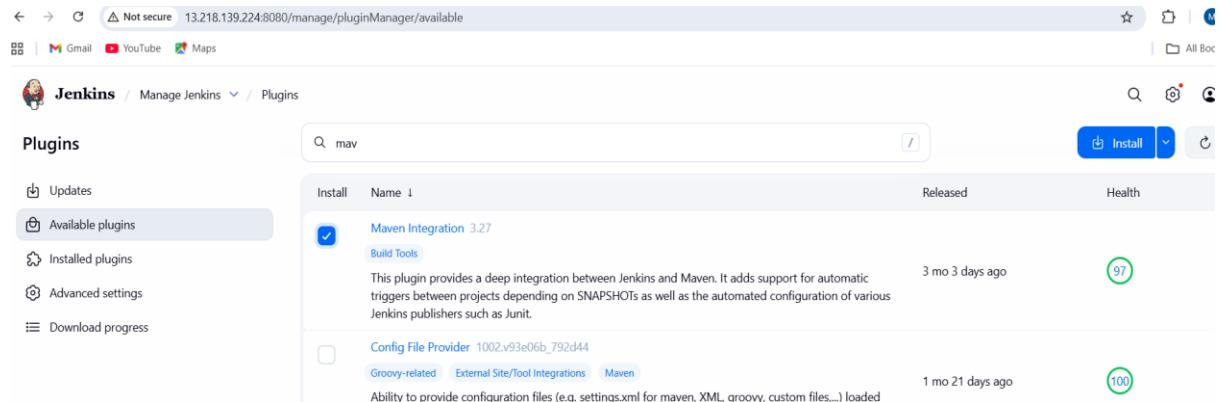
## 4. Create one Jenkins job using Maven Project for the code below with two stages:

- **Stage 1: Git clone**

- **Stage 2: Maven**

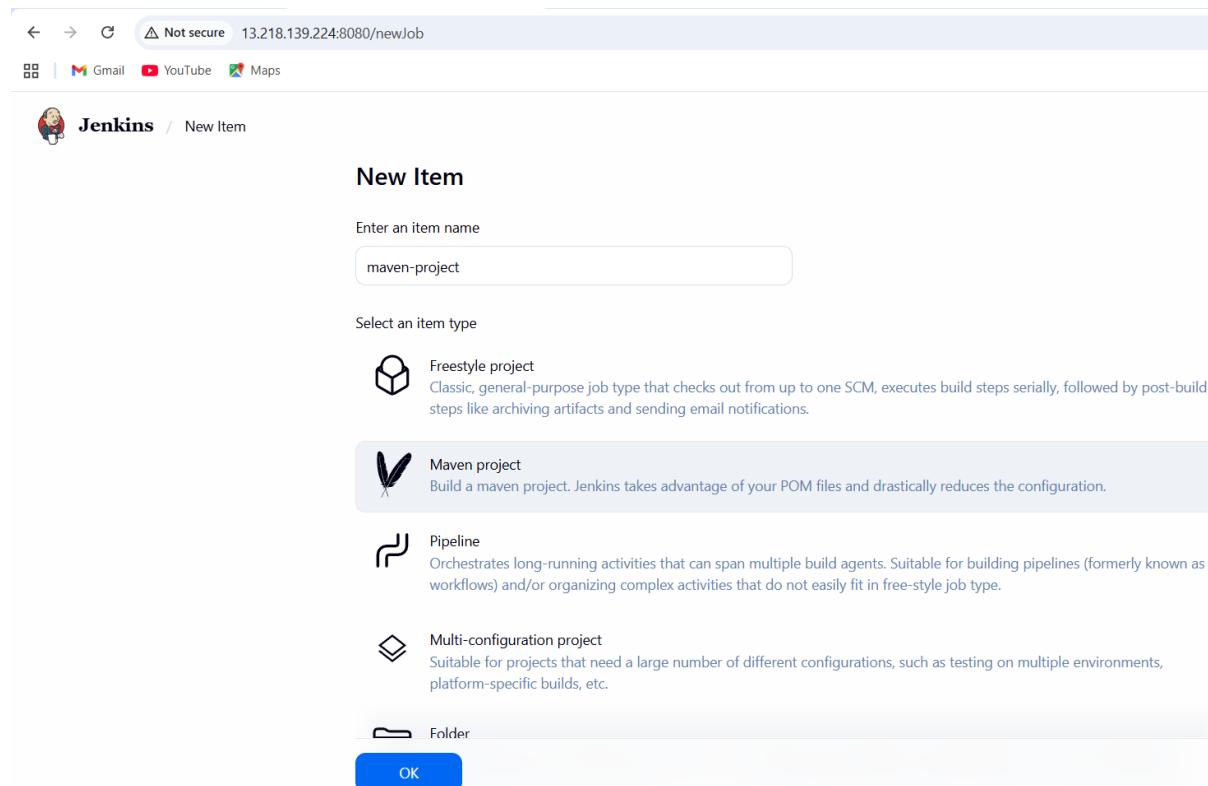
**Compilation Code:** <https://github.com/betawins/java-Working-app.git>

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



The screenshot shows the Jenkins Plugin Manager interface. In the search bar at the top, the text "mav" is typed. Below the search bar, there is a table with columns: Install, Name, Released, and Health. The first row in the table is highlighted with a blue checkmark in the Install column and has a green circular badge with the number "97" next to it. This row corresponds to the "Maven Integration 3.27" plugin, which is described as providing deep integration between Jenkins and Maven. The second row in the table has a green circular badge with the number "100" next to it. This row corresponds to the "Config File Provider" plugin, which allows providing configuration files for Maven, XML, Groovy, and custom files.

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



The screenshot shows the Jenkins "New Item" creation dialog. In the "Enter an item name" field, the text "maven-project" is entered. In the "Select an item type" section, the "Maven project" option is selected, indicated by a blue background. Other options listed include "Freestyle project", "Pipeline", "Multi-configuration project", and "Folder". At the bottom of the dialog is a blue "OK" button.

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

← → ⌂ Not secure 13.218.139.224:8080/job/maven-project/configure

Gmail YouTube Maps

Jenkins / maven-project / Configuration

Configure

General

Source Code Management

Triggers

Environment

Pre Steps

Build

Post Steps

Build Settings

Post-build Actions

Repositories

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

Credentials: - none -

Advanced

+ Add Repository

Branches to build

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

Save Apply

This screenshot shows the Jenkins configuration page for a job named 'maven-project'. The left sidebar lists various configuration sections like General, Triggers, Environment, etc. The 'Source Code Management' section is currently selected and expanded. It contains fields for 'Repository URL' (set to 'https://github.com/betawins/hiring-app.git'), 'Credentials' (set to '- none -'), and an 'Advanced' dropdown. Below these are buttons for '+ Add Repository' and 'Branches to build'. The 'Branch Specifier' field is set to '\*/main'. At the bottom are 'Save' and 'Apply' buttons.

← → ⌂ Not secure 13.218.139.224:8080/job/maven-project/

Gmail YouTube Maps

Jenkins / maven-project

Status

maven-project

</> Changes

Workspace

Build Now

Configure

Delete Maven project

Modules

Rename

Credentials

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

Builds

Filter

Today

#3 11:27 AM

This screenshot shows the main Jenkins project page for 'maven-project'. On the left, there's a sidebar with links like Status, Changes, Workspace, etc. The main area shows a green checkmark icon next to 'maven-project'. Below it is a 'Permalinks' section with a bulleted list of recent builds. At the bottom, there's a 'Builds' section with a 'Filter' input, a date range from 'Today', and a specific build entry for '#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.

The screenshot shows the Jenkins configuration page for a job named "maven-project". The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Pre Steps (selected), Build, Post Steps, Build Settings, and Post-build Actions. The "Pre Steps" section has a "Root POM" field set to "pom.xml" and a "Goals and options" field set to "clean compile". The "Build" section includes an "Advanced" dropdown. The "Post Steps" section has a radio button for "Run only if build succeeds". At the bottom are "Save" and "Apply" buttons. The status bar at the bottom shows a weather icon for 29°C and sunny conditions, along with various system icons.

The screenshot shows a browser window with the address bar displaying 'Not secure 13.218.139.224:8080/job/maven-project/'. Below the address bar are links for Gmail, YouTube, and Maps. The main content area is titled 'Jenkins / maven-project'. On the left, there's a sidebar with various options like Status, Changes, Workspace, Build Now, Configure, Delete Maven project, Modules, Rename, and Credentials. The 'Status' tab is selected, showing a green checkmark and the text 'maven-project'. To the right of the sidebar, under 'Permalinks', is a list of recent builds: 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; and Last completed build (#4), 21 min ago. At the bottom left, there's a 'Builds' sidebar with a 'Filter' input field, showing 'Today' with entries for build #4 at 11:32 AM and build #3 at 11:27 AM.

## 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.

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a header bar with a back arrow, forward arrow, and a 'Not secure' warning. Below it is a navigation bar with links for Gmail, YouTube, and Maps. The main title is 'Jenkins / All / New Item'. The page has a heading 'New Item' and a sub-instruction 'Enter an item name' with a text input field containing 'java-maven-parameterized'. A section titled 'Select an item type' lists several options: 'Freestyle project' (selected), 'Maven project' (highlighted in grey), 'Pipeline', 'Multi-configuration project', and 'Folder'. Each option has a brief description below it. At the bottom right is a blue 'OK' button.

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

- Give name as **BRANCH\_NAME**
- Default value as **main**

The screenshot shows the Jenkins 'Configuration' screen for the 'java-maven-parameterized' job. On the left, there's a sidebar with tabs for General, Source Code Management, Triggers, Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The 'General' tab is selected. In the main area, there's a checkbox for 'GitHub project' which is unchecked. Below it is a checked checkbox for 'This project is parameterized'. A 'String Parameter' configuration panel is open, showing a 'Name' field with 'BRANCH\_NAME' and a 'Default Value' field with 'main'. There's also a 'Description' field and a 'Plain text Preview' link. At the bottom of the panel are 'Trim the string' and 'OK' buttons.

Give git URL and select branch as main.

The screenshot shows the Jenkins job configuration page for a job named "java-maven-parameterized". The left sidebar lists configuration sections: General, Source Code Management (selected), Triggers, Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. Under "Source Code Management", "Git" is selected, and the "Repository URL" is set to "https://github.com/betawins/hiring-app.git". The "Branches to build" field contains the value "\*/\*main". At the bottom are "Save" and "Apply" buttons.

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

The screenshot shows the Jenkins job configuration page for a job named "java-maven-parameterized". The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Pre Steps (selected), Build, Post Steps, Build Settings, and Post-build Actions. Under "Pre Steps", the "Root POM" is set to "pom.xml" and the "Goals and options" are set to "clean compile". At the bottom are "Save" and "Apply" buttons.

Click on save and click on build with parameters.

The screenshot shows the Jenkins job configuration page for 'java-maven-parameterized'. On the left, there is a sidebar with various options: Status, Changes, Workspace, Build with Parameters, Configure, Delete Maven project, Modules, Rename, and Credentials. The 'Build with Parameters' option is selected. In the main area, the title is 'Maven project java-maven-parameterized'. It says 'This build requires parameters:' and shows a parameter named 'BRANCH\_NAME' with the value 'main'. There are two buttons at the bottom: a green 'Build' button and a grey 'Cancel' button. Below this, there is a section titled 'Builds' which displays 'No builds'.

The screenshot shows the Jenkins job summary page for 'java-maven-parameterized'. The sidebar on the left is identical to the previous screenshot. In the main area, the job status is shown as 'Status' with a green checkmark and the text 'java-maven-parameterized'. Below this, there is a section titled 'Permalinks' with a bulleted list of links: 'Last build (#1), 32 sec ago', 'Last stable build (#1), 32 sec ago', 'Last successful build (#1), 32 sec ago', and 'Last completed build (#1), 32 sec ago'. At the bottom, there is a 'Builds' section with a search bar labeled 'Filter' and a dropdown menu showing 'Today' and '#1 12:09 PM'.

## 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**.

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

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

**STAGE\_NAME** Name of the current pipeline stage (when used inside stage).