Terraform-01 and Terraform-02

1) Install Terraform on your PC

--terraform download-windows-extract file-copy and paste in c-open gitbash-expose environmentvariables-save it now check it

```
MINGW64:/c/terraform

naren@narendar MINGW64 /c/terraform
$ terraform -v
Terraform v1.11.4
on windows_386
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\naren> cd C:\
PS C:\> cd terroform_basic
PS C:\terroform_basic> terraform -v
Terraform v1.11.4
on windows_386
PS C:\terroform_basic>
```

2) Execute all the templates shown in video.

--connect to visual studio code

--create template_basic folder

-Create main.tf file

Then do

a) basic terraform template with local-file provider

```
main.tf
C: > terroform_basic > main.tf > resource "local_file" "pet" > content
1  resource "local_file" "pet" {
2  filename = " pets.txt"
3  content = "I love pets!"
4  }
5
```

-execution

first do init

```
PS C:\terroform_basic> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.2...
- Installed hashicorp/local v2.5.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

do plan and check

```
PS C:\terroform_basic> terraform plan

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

Terraform will perform the following actions:

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

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

then do apply

```
PS C:\terroform_basic> terraform apply

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

Terraform will perform the following actions:

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

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

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

  Enter a value: yes

local_file.pet: Creating...
local_file.pet: Creation complete after 0s [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]

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

--if we want to delete above template delete by using below command

```
PS C:\terroform_basic> terraform destroy
local_file.pet: Refreshing state... [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # local_file.pet will be destroyed
  - resource "local_file" "pet" {
      - content              = "I love pets!" -> null
      - content_base64sha256 = "4LBQ/UV/Gxp6DWNEp+upXEfPrTTpUCpfiAkOpJixJEc=" -> null
      - content_base64sha512 = "wKb93aeKXXwWVvqgU6yfOuiKC6dLDLLWp7dclsVtlgenVx4/BUpKiEnsdA6EbHgUmSDTp/+Si9wFeXH8/46Eiw==" -> null
      - content_md5          = "3dd08189b6cbd661de6b25ed369f5746" -> null
      - content_sha1         = "7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68" -> null
      - content_sha256       = "e0b050fd457f1b1a7a0d6344a7eba95c47cfad34e9502a5f88090ea498b12447" -> null
      - content_sha512       = "c0a6fddda78a5d7c1656faa053ac9f3ae88a0ba74b0cb2d6a7b75c96c56d9607a7571e3f054a4a8849ec740e846c78149920d3a7ff928bdc057971fcff8e848b" -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = " pets.txt" -> null
      - id                   = "7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68" -> null
    }

Plan: 0 to add, 0 to change, 1 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.pet: Destroying... [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]
local_file.pet: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
```

b) multiple provider (random provider & local file provider)

--TF template

```
main.tf
C: > terroform_basic > main.tf > ...
resource "local_file" "pet" {
filename = " pets.txt"
content = "I love pets!"
}
resource "random_pet" "mypet" {
prefix = "MR"
separator = "."
length = "1"
}
```

--execution

-init

-plan

-then apply random name created

```
PS C:\terroform_basic> terraform apply

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

Terraform will perform the following actions:

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

```
  # random_pet.mypet will be created
  + resource "random_pet" "mypet" {
      + 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

local_file.pet: Creating...
random_pet.mypet: Creating...
random_pet.mypet: Creation complete after 0s [id=MR.raccoon]
local_file.pet: Creation complete after 0s [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]

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

3)Note down below points

Terraform Init
Terraform Plan
Terraform Apply
Terraform Provider

Terraform Init

- Initializes the working directory with Terraform configuration files.

-Downloads provider plugins.

-Sets up backend configuration (if defined).

```
PS C:\terroform_basic> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.2...
- Installed hashicorp/local v2.5.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Terraform Plan

- Creates an execution plan by comparing current infrastructure with desired state.

- Helps you review what will change before applying.

```
PS C:\terroform_basic> terraform plan

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

Terraform will perform the following actions:

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

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

Terraform Apply

-Applies the changes to reach the desired state as defined in the configuration.

-Prompts for approval before execution unless -auto-approve is used.

```
PS C:\terroform_basic> terraform apply

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

Terraform will perform the following actions:

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

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

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

  Enter a value: yes

local_file.pet: Creating...
local_file.pet: Creation complete after 0s [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]

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

Terraform Provider

- Providers are plugins that allow Terraform to interact with APIs (e.g., AWS, Azure, Google Cloud, etc.).

- Defined in configuration files using the provider block.

- Providers are downloaded during terraform init.

```
PS C:\terroform_basic> terraform providers

Providers required by configuration:
.
└── provider[registry.terraform.io/hashicorp/local]

Providers required by state:

    provider[registry.terraform.io/hashicorp/local]
```

Terroform format

-The terraform fmt command is used to automatically format your Terraform configuration files to follow the canonical style.

main.tf

C: > terroform_basic > main.tf > ...

```
resource "local_file" "pet" {
  filename = " pets.txt"
  content  = "I love pets!"
}
resource "random_pet" "mypet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

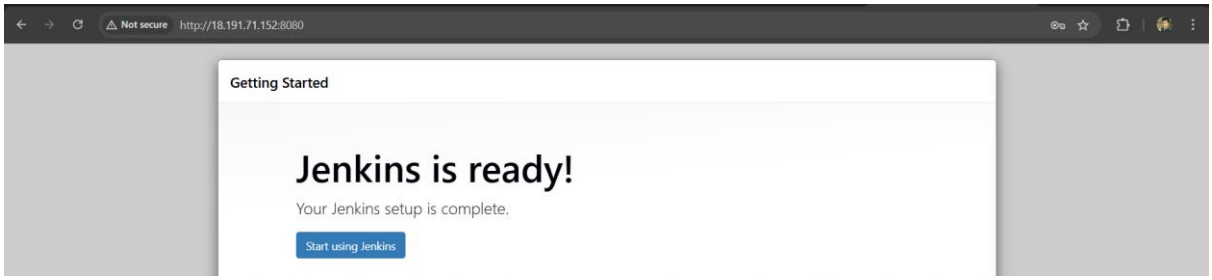
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\terroform_basic> terraform fmt
main.tf
PS C:\terroform_basic>
```

4)Integrate a sample Terraform template in jenkins.

--create EC2 instance

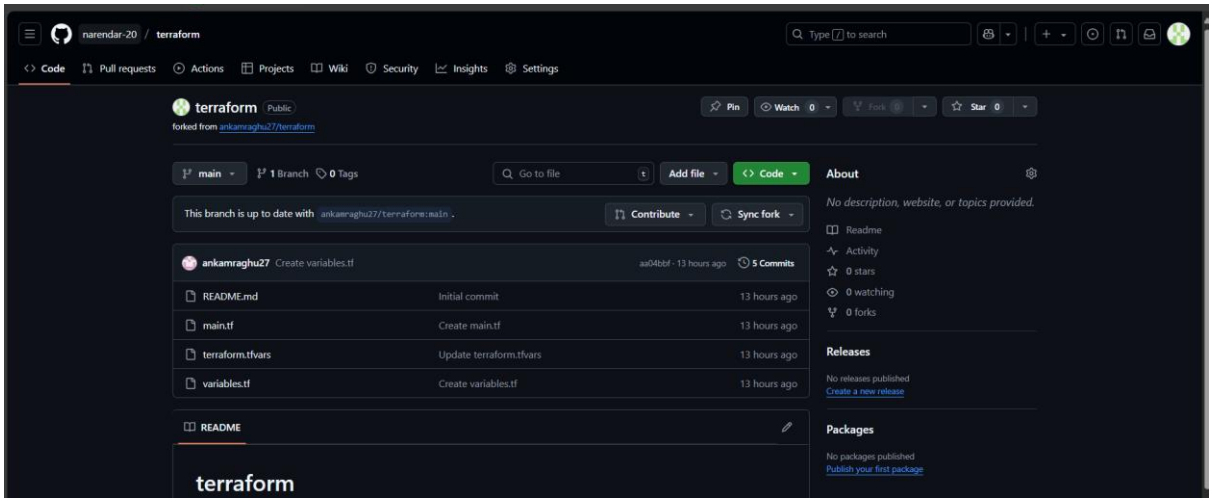
--install Jenkins



--install terraform in Jenkins server along with git

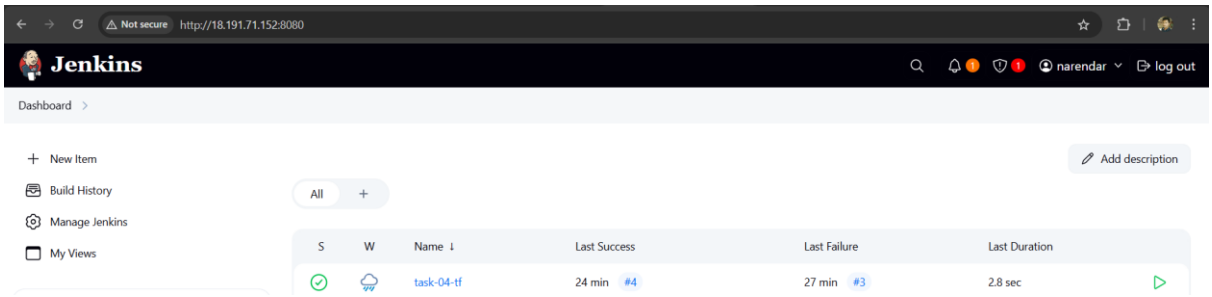
```
Complete!
[root@ip-172-31-7-208 ~]# terraform -v
Terraform v1.11.4
on linux_amd64
```

--create repo in git hub



https://github.com/narendar-20/terraform.git

--create a Jenkins job

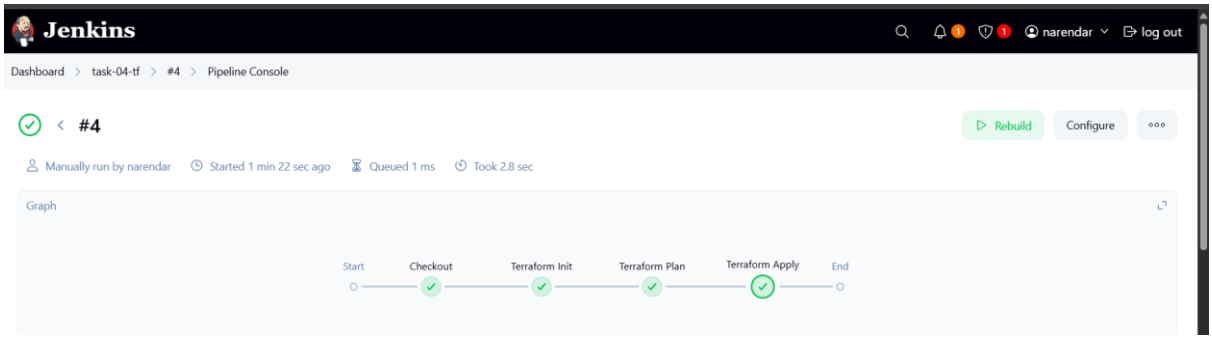


--add terraform pipeline

```
pipeline {
   agent any

   environment {
```

```
        TF_IN_AUTOMATION = "true"
    }

    stages {
        stage('Checkout') {
            steps {
                git url: 'https://github.com/narendar-20/terraform.git', branch: 'main'
            }
        }

        stage('Terraform Init') {
            steps {
                sh 'terraform init'
            }
        }

        stage('Terraform Plan') {
            steps {
                sh 'terraform plan'
            }
        }

        stage('Terraform Apply') {
            steps {
                sh 'terraform apply -auto-approve'
            }
        }
    }
}
```

--build the job



```
22
23   # random_pet.mypet will be created
24   + resource "random_pet" "mypet" {
25       + id        = (known after apply)
26       + length    = 1
27       + prefix    = "MR"
28       + separator = "."
29     }
30
31 Plan: 2 to add, 0 to change, 0 to destroy.
32 random_pet.mypet: Creating...
33 random_pet.mypet: Creation complete after 0s [id=MR.kite]
34 local_file.pet_file: Creating...
35 local_file.pet_file: Creation complete after 0s [id=fefacccdae259f25533749abfb90e27558256459]
36
37 Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

--check in Jenkins server

```
[root@ip-172-31-7-208 ~]# terraform -v
Terraform v1.11.4
on linux_amd64
[root@ip-172-31-7-208 ~]# ll
total 0
[root@ip-172-31-7-208 ~]# cd /var/lib/jenkins/workspace
[root@ip-172-31-7-208 workspace]# ls
task-04-tf  task-04-tf@tmp
[root@ip-172-31-7-208 workspace]# cd task-04-tf
[root@ip-172-31-7-208 task-04-tf]# ls
README.md  main.tf  pets.txt  terraform.tfstate  terraform.tfvars  variables.tf
[root@ip-172-31-7-208 task-04-tf]#
```

i-06add05e8e88f0610 (jenkins)

PublicIPs: 18.191.71.152 PrivateIPs: 172.31.7.208

5) Watch the terraform-02 video.

--completed

6) Execute all the templates shown in video.

a) create_before_destroy

--template

```
main.tf > ...
resource "local_file" "pet" {
  filename = " pets.txt"
  content  = "I love dog"
  lifecycle {
    create_before_destroy = true
  }
}
resource "random_pet" "mypet" {
  prefix    = "MR"
  separator = "."
  length    = "1"
}
```

--execution

Its creating before destroy

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

Plan: 1 to add, 0 to change, 1 to destroy.
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.pet: Creating...
local_file.pet: Creation complete after 0s [id=d933fb7152a99816ca16f99eb573a6bc34eeef82]
local_file.pet (deposed object b8cdab7a): Destroying... [id=7e4db4fbfdbb108bdd04692602bae3e9bd1e1b68]
local_file.pet: Destruction complete after 0s

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

b) prevent_destroy

--template

```
Welcome   main.tf   .terraform.lock.hcl
main.tf > resource "local_file" "pet" > lifecycle
1   resource "local_file" "pet" {
2     filename = " pets.txt"
3     content  = "I love dog"
4     lifecycle {
5       prevent_destroy = true
6     }
7   }
8   resource "random_pet" "mypet" {
9     prefix    = "MR"
10    separator = "."
11    length    = "1"
12  }
13
```

--execution

Its not destroyed gave an error

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

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

Error: Instance cannot be destroyed

  on main.tf line 1:
   1: resource "local_file" "pet" {
```

c) ignore_changes

--first make changes in state file

```
EXPLORER
TERROFORM_BASIC
  .terraform
  pets.txt
  .terraform.lock.hcl
  main.tf
  terraform.tfstate
  terraform.tfstate.backup

Welcome   main.tf   terraform.tfstate
terraform.tfstate > [ ] resources > {} 0 > [ ] instances > {} 0 > {} attributes
7   "resources": [
8     {
10      "type": "local_file",
11      "name": "pet",
12      "provider": "provider[\"registry.terraform.io/hashicorp/local\"]",
13      "instances": [
14        {
15          "schema_version": 0,
16          "attributes": {
17            "content": "I love cat",
18            "content_base64": null,
19            "content_base64sha256": "H4n8ARFuizazPoskiLeX/zAqgWVm1b/N9tUILbxmJhE=",
20            "content_base64sha512": "D/oSkMsivTfagv8VmQTqRhKJ2nAoO/Qdpi6UXEprc1CcDW
21            "content_md5": "f207f0440c5a66afb5fbab224b02c53e",
22            "content_sha1": "d933fb7152a99816ca16f99eb573a6bc34eeef82",
```

--and don't do any changes in actual main.tf template

```
main.tf > resource "local_file" "pet"
1   resource "local_file" "pet" {
2     filename = " pets.txt"
3     content  = "I love dog"
4     lifecycle {
5       ignore_changes = [
6         content
7       ]
8     }
9   }
10  resource "random_pet" "mypet" {
11    prefix    = "MR"
12    separator = "."
13    length    = "1"
14  }
15
```

--execution

No changes happened here because of ignore_changes lifecycle

```
PS C:\terroform_basic> terraform apply
local_file.pet: Refreshing state... [id=d933fb7152a99816ca16f99eb573a6bc34eeef82]
random_pet.mypet: Refreshing state... [id=MR.seal]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

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

VARIABLES

-----apply multiple variables

--templates

Welcome | main.tf | variables.tf | terraform.tfstate

main.tf > resource "local_file" "pet" > content

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

```

Welcome | main.tf | variables.tf | terraform.tfstate

variables.tf > variable "separator"

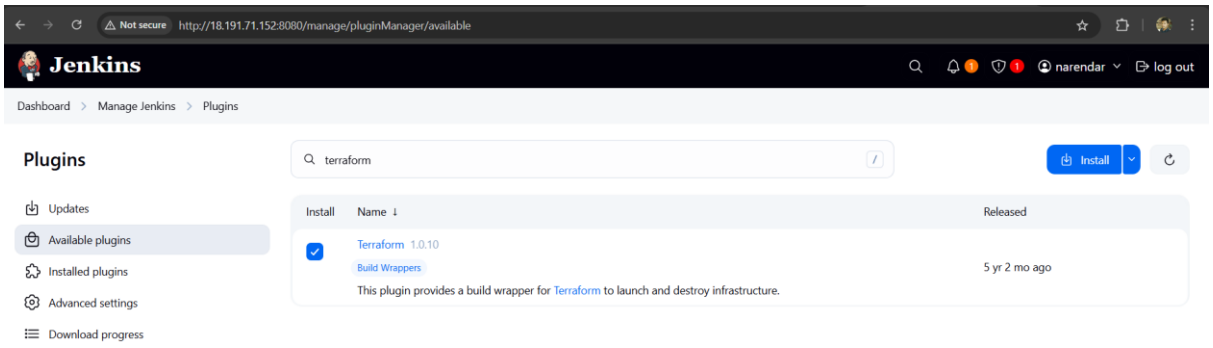
```
variable "filename" {
default ="root/pets.txt"
}
variable "content" {
default ="we love pets!"
}
variable "prefix" {
default ="MR"
}
variable "separator" {
default ="."
}
variable "length" {
default ="1"
}
```

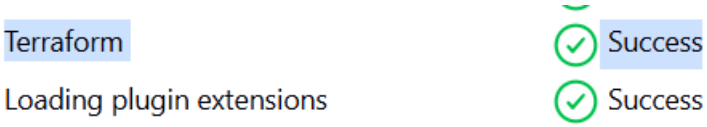
--execution

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

      ~ content_sha512       = "0ffa1290cb22bd37da82ff159904ea461289da70283bf41da62e945c4a6b73509c0d648f8516
e4e4413b6aae98471" -> (known after apply)
      ~ id                   = "d933fb7152a99816ca16f99eb573a6bc34eeef82" -> (known after apply)
        # (3 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.pet: Destroying... [id=d933fb7152a99816ca16f99eb573a6bc34eeef82]
local_file.pet: Destruction complete after 0s
local_file.pet: Creating...
local_file.pet: Creation complete after 0s [id=fefacccdae259f25533749abfb90e27558256459]

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

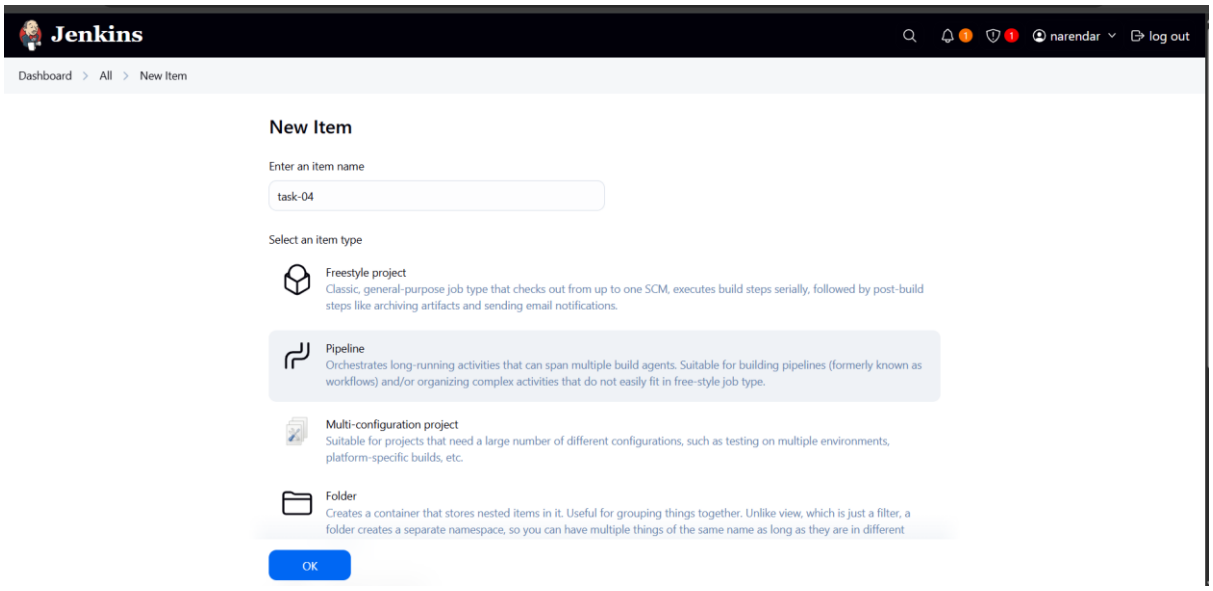
7) Integrate terraform in jenkins using Terraform plugin.

--install Terraform plungin in Jenkins GUI

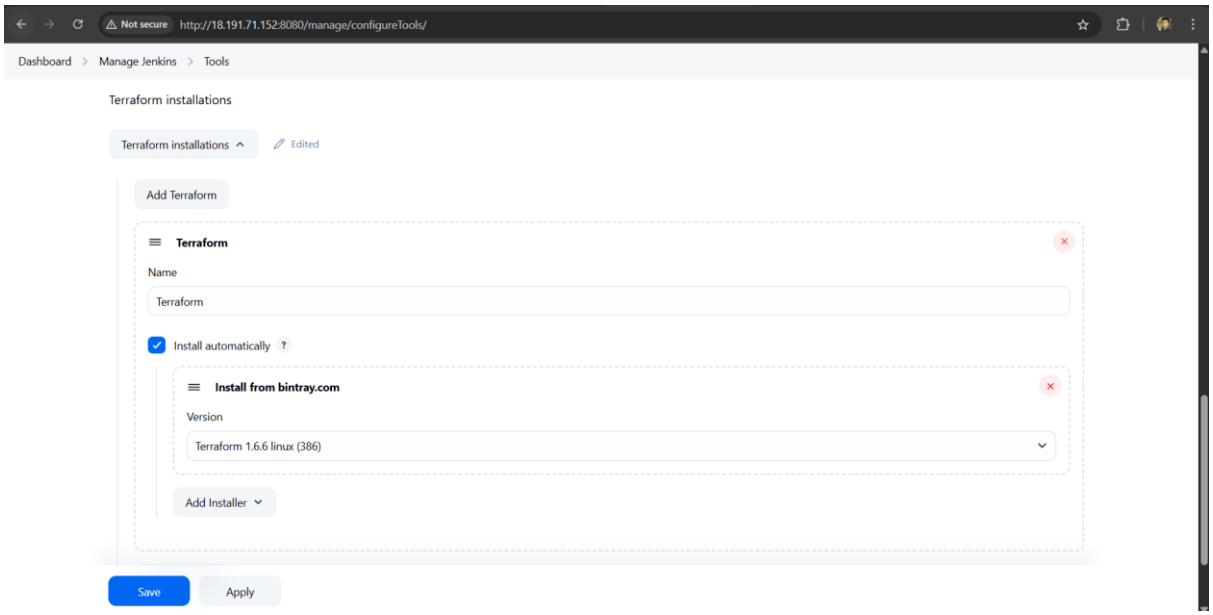




--create a pipeline job



--add the plugin in Jenkins tools



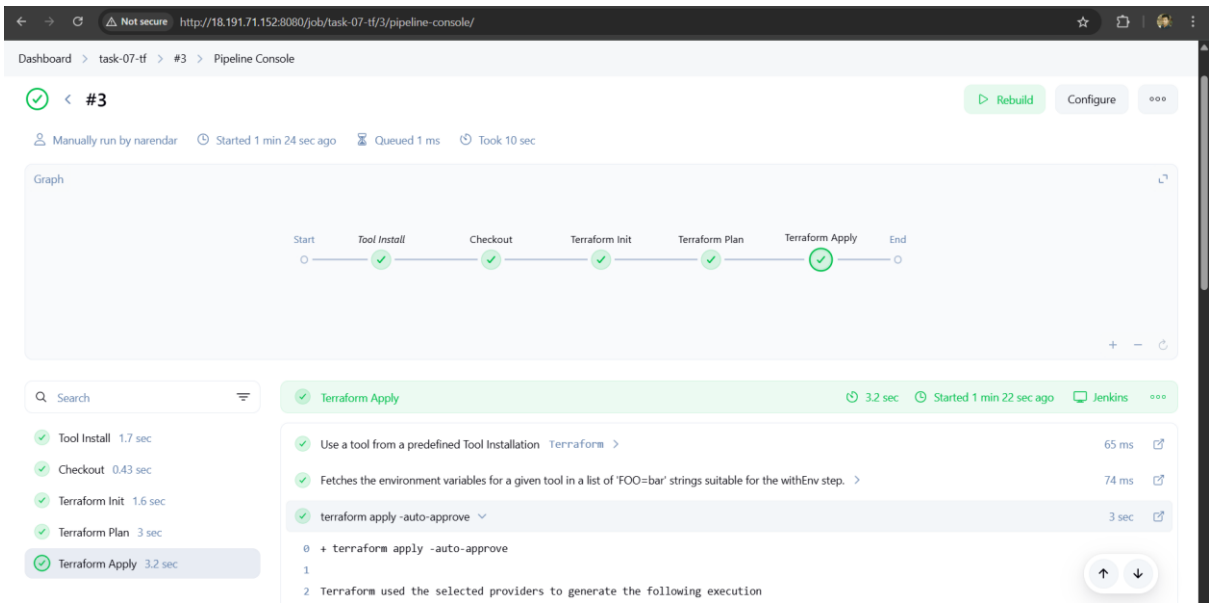
--pipeline add

```
pipeline {

   agent any


   tools {

      terraform 'Terraform'   // Make sure this matches your Jenkins Terraform tool name

   }
```

```
environment {
    TF_ROOT = "${WORKSPACE}"
}

stages {
    stage('Checkout') {
        steps {
            git branch: 'main',
                url: 'https://github.com/narendar-20/terraform.git'
        }
    }

    stage('Terraform Init') {
        steps {
            dir("${TF_ROOT}") {
                sh 'terraform init'
            }
        }
    }

    stage('Terraform Plan') {
        steps {
            dir("${TF_ROOT}") {
                sh 'terraform plan'
            }
        }
    }
```

```
        stage('Terraform Apply') {
            steps {
                dir("${TF_ROOT}") {
                    sh 'terraform apply -auto-approve'
                }
            }
        }
    }
}
```

--build the job



Not secure http://18.191.71.152:8080/job/task-07-tf/3/pipeline-console/

Dashboard > task-07-tf > #3 > Pipeline Console

```
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "pets.txt"
      + id                   = (known after apply)
    }

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

Plan: 2 to add, 0 to change, 0 to destroy.
random_pet.mypet: Creating...
random_pet.mypet: Creation complete after 0s [id=MR.shrimp]
local_file.pet_file: Creating...
local_file.pet_file: Creation complete after 0s [id=fefacccdae259f25533749abfb90e27558256459]

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

--now check in Jenkins server

```
[root@ip-172-31-7-208 workspace]# cd task-07-tf
[root@ip-172-31-7-208 task-07-tf]# ls
README.md  main.tf  pets.txt  terraform.tfstate  terraform.tfvars  variables.tf
[root@ip-172-31-7-208 task-07-tf]# ll
total 24
-rw-r--r--. 1 jenkins jenkins   11 May  2 09:04 README.md
-rw-r--r--. 1 jenkins jenkins  201 May  2 09:04 main.tf
-rwxr-xr-x. 1 jenkins jenkins   13 May  2 09:09 pets.txt
-rw-r--r--. 1 jenkins jenkins 1894 May  2 09:09 terraform.tfstate
-rw-r--r--. 1 jenkins jenkins   98 May  2 09:04 terraform.tfvars
-rw-r--r--. 1 jenkins jenkins  486 May  2 09:04 variables.tf
```