TASKS ON TERRAFORM

- **NAME: S.MUZZAMMIL HUSSAIN**
- **DATE: 24/07/2025**
- **BATCH: 11**
- **NO.OF TASKS: 1**

Task 1. Explore Terraform Meta arguments (depends_on, count, for_each, provider, lifecycle[create_before_destroy, prevent_destroy, ignore_changes, replace_triggered_by])

Terraform supports several **meta-arguments**, which are special arguments we can add to resource, data, and module blocks to alter how resources are managed. The main meta-argument types are:

- **depends_on**
  Specifies explicit dependencies between resources or modules when Terraform cannot infer them by attribute references. This ensures correct ordering for resource creation or destruction.
- **count**
  Allows you to create multiple instances of a resource by specifying the desired number. Each instance gets an index.
- **for_each**
  Iterates over a map or set, creating a separate instance of the resource for each element. Use this when we want to create resources keyed by a unique label.
- **provider**
  Assigns a specific provider configuration or alias to a resource or module. This is essential in multi-provider or multi-region setups.
- **lifecycle**
  A nested block that customizes how resources are created, updated, or destroyed. Key options inside lifecycle include:
  - **create_before_destroy**
  - **prevent_destroy**
  - **ignore_changes**
  - **replace_triggered_by**

These meta-argument types can be used individually or in combination (with some constraints, such as not using both count and for_each in the same block) to control resource behavior, dependency, repetition, and lifecycle management more flexibly.

Summary Table

Meta-Argument	Purpose
depends_on	Explicit dependencies and ordering
count	Create multiple identical resources
for_each	Create resources for each element in a collection
provider	Assign specific provider configuration
lifecycle	Advanced lifecycle controls (create_before_destroy, etc.)

Each meta-argument has rules and best practices for use; for example, use count for simple multiples, for_each for keyed collections, and depends_on only when implicit dependencies aren't sufficient.

1. Depends on

Explanation of **implicit (direct) dependencies** and **explicit (indirect) dependencies** in Terraform resource configuration, illustrated with our local_file resource examples:

Terraform Resource Dependencies: Implicit vs Explicit

In Terraform, managing the order in which resources are created, updated, or destroyed is critical when one resource depends on another. Dependencies can be:

- **Implicit (Direct) Dependencies**: Automatically inferred by Terraform when one resource references attributes of another resource.
- **Explicit (Indirect) Dependencies**: Manually declared using the depends_on meta-argument to enforce dependency even when no direct reference is present.

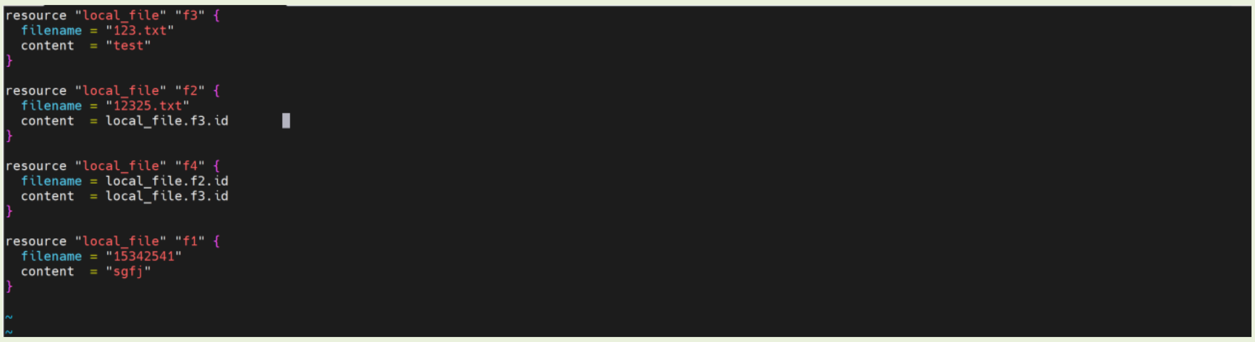
➢ **Implicit (Direct) Dependency**

When a resource block references an attribute of another resource, Terraform automatically creates a dependency between them. This ensures the referenced resource is created or updated before the dependent resource.

Example:

Create a **main.tf** file using **vi main.tf** add the following configuration:

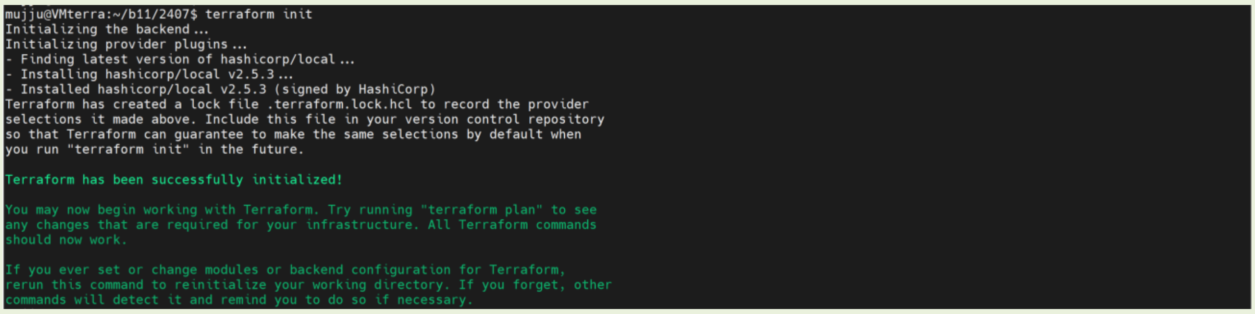
```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"
}

resource "local_file" "f2" {
  filename = "12325.txt"
  content  = local_file.f3.id        # Direct reference creates implicit dependency on f3
}

resource "local_file" "f4" {
  filename = local_file.f2.id        # Depends on f2
  content  = local_file.f3.id        # Also depends on f3
}

resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"
}
```



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"
}

resource "local_file" "f2" {
  filename = "12325.txt"
  content  = local_file.f3.id
}

resource "local_file" "f4" {
  filename = local_file.f2.id
  content  = local_file.f3.id
}

resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"
}

~
~
```

Initialize Terraform

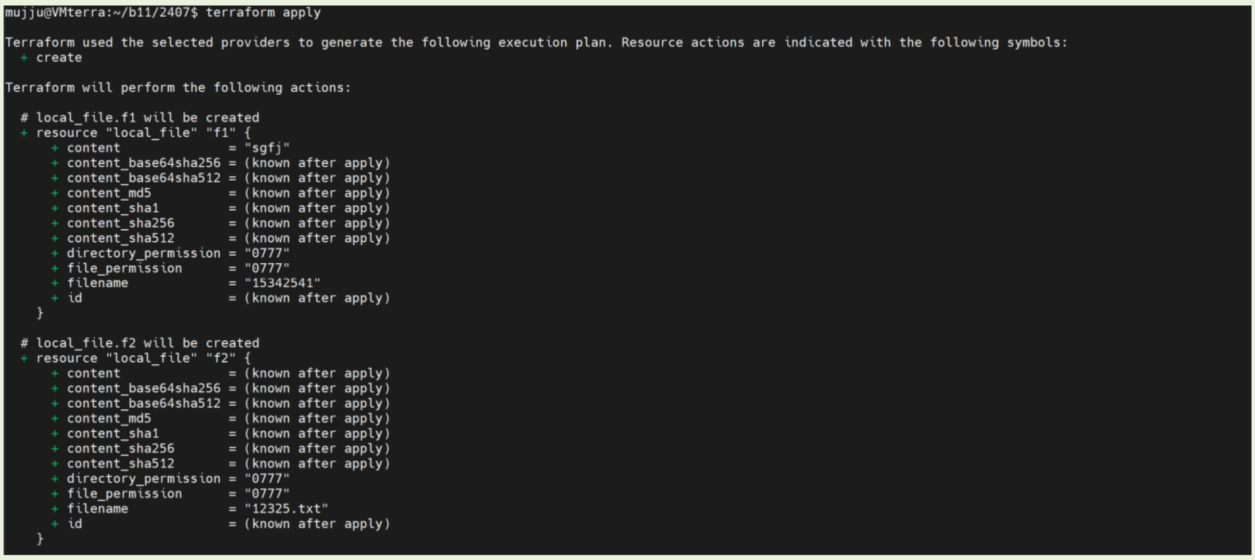
terraform init



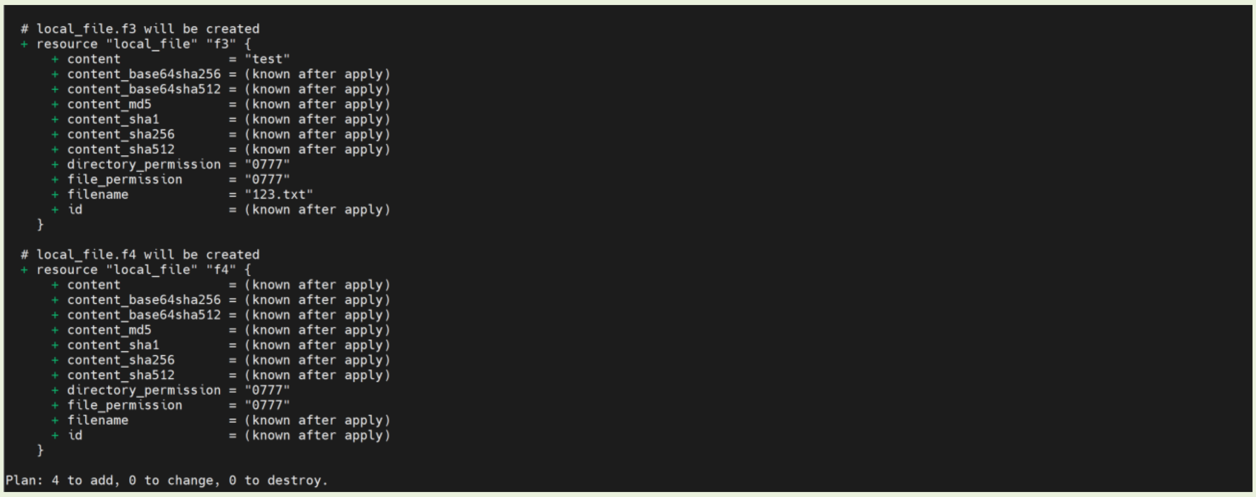
```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
```

Run Terraform apply

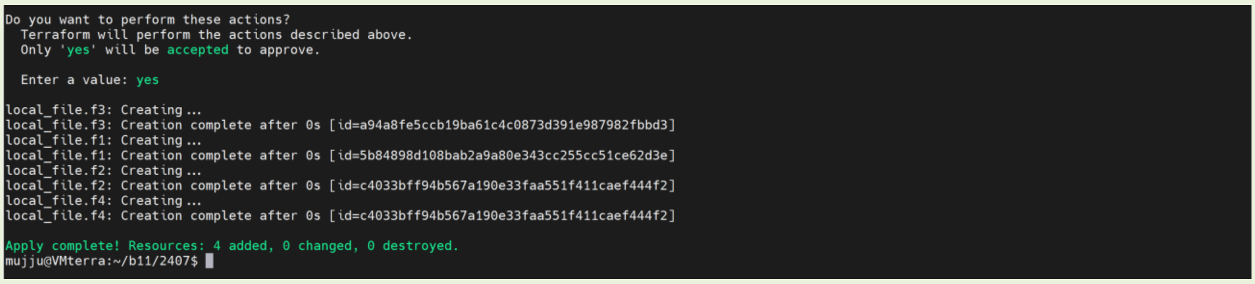
terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f1 will be created
  + resource "local_file" "f1" {
      + content              = "sgfj"
      + 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             = "15342541"
      + id                   = (known after apply)
    }

  # local_file.f2 will be created
  + resource "local_file" "f2" {
      + content              = (known after apply)
      + 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             = "12325.txt"
      + id                   = (known after apply)
    }
```

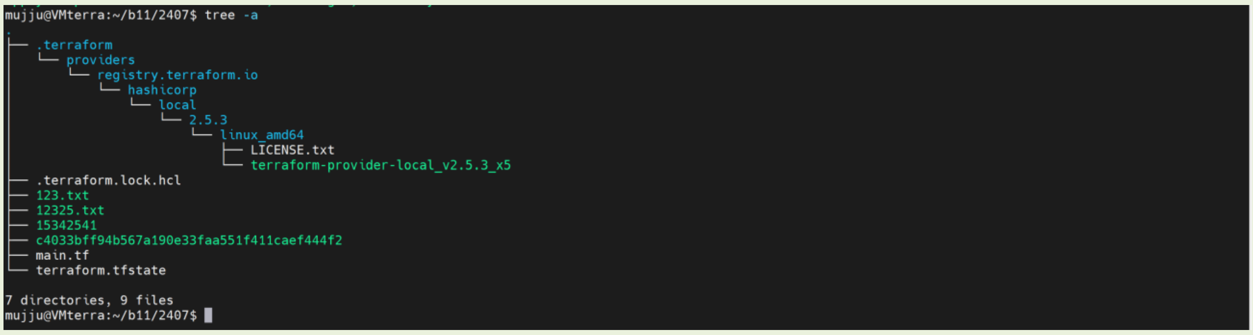


```
  # local_file.f3 will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "123.txt"
      + id                   = (known after apply)
    }

  # local_file.f4 will be created
  + resource "local_file" "f4" {
      + content              = (known after apply)
      + 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             = (known after apply)
      + id                   = (known after apply)
    }

Plan: 4 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.f3: Creating...
local_file.f3: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f1: Creating...
local_file.f1: Creation complete after 0s [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]
local_file.f2: Creating...
local_file.f2: Creation complete after 0s [id=c4033bff94b567a190e33faa551f411caef444f2]
local_file.f4: Creating...
local_file.f4: Creation complete after 0s [id=c4033bff94b567a190e33faa551f411caef444f2]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$ 
```

Check the contents using **tree -a**



```
mujju@VMterra:~/b11/2407$ tree -a
.
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               └── local
│                   └── 2.5.3
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-local_v2.5.3_x5
├── .terraform.lock.hcl
├── 123.txt
├── 12325.txt
├── 15342541
├── c4033bff94b567a190e33faa551f411caef444f2
├── main.tf
└── terraform.tfstate

7 directories, 9 files
mujju@VMterra:~/b11/2407$
```

What happens here:

- **local_file.f2** depends on **local_file.f3** because it references **local_file.f3.id.**
- **local_file.f4** depends on both **local_file.f2** and **local_file.f3**.
- **local_file.f1** has no dependencies.

Terraform automatically understands the necessary resource creation order based on these references.

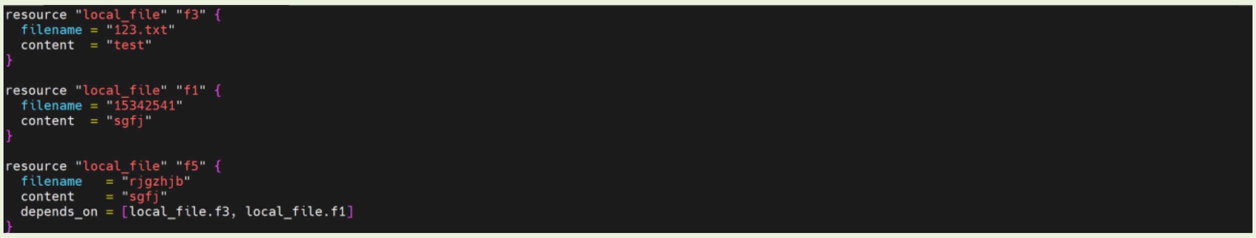
➢ Explicit (Indirect) Dependency Using depends_on

Sometimes when we want to force a resource to depend on another even if there is no direct reference between their attributes. For this, Terraform provides the depends_on meta-argument that explicitly declares dependencies.

Example:

Create a **main.tf** file using **vi main.tf** add the following configuration

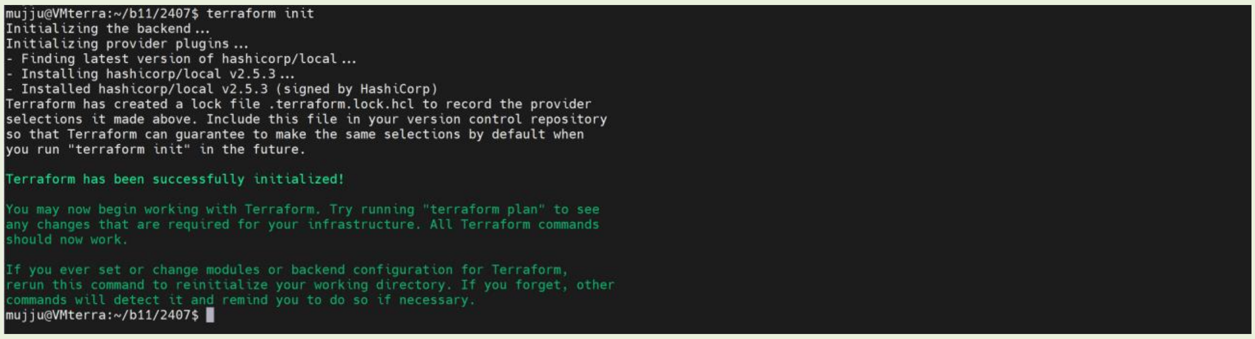
```
 resource "local_file"  "f3" {
  filename = "123.txt"
  content  = "test"
}

resource "local_file"  "f1" {
  filename = "15342541"
  content  = "sgfj"
}

resource "local_file"  "f5" {
  filename   = "rjgzhjb"
  content    = "sgfj"
  depends_on = [local_file.f3, local_file.f1]   # Explicitly depends on f3 and f1
}
```



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"
}

resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"
}

resource "local_file" "f5" {
  filename   = "rjgzhjb"
  content    = "sgfj"
  depends_on = [local_file.f3, local_file.f1]
}
```

Initialize Terraform

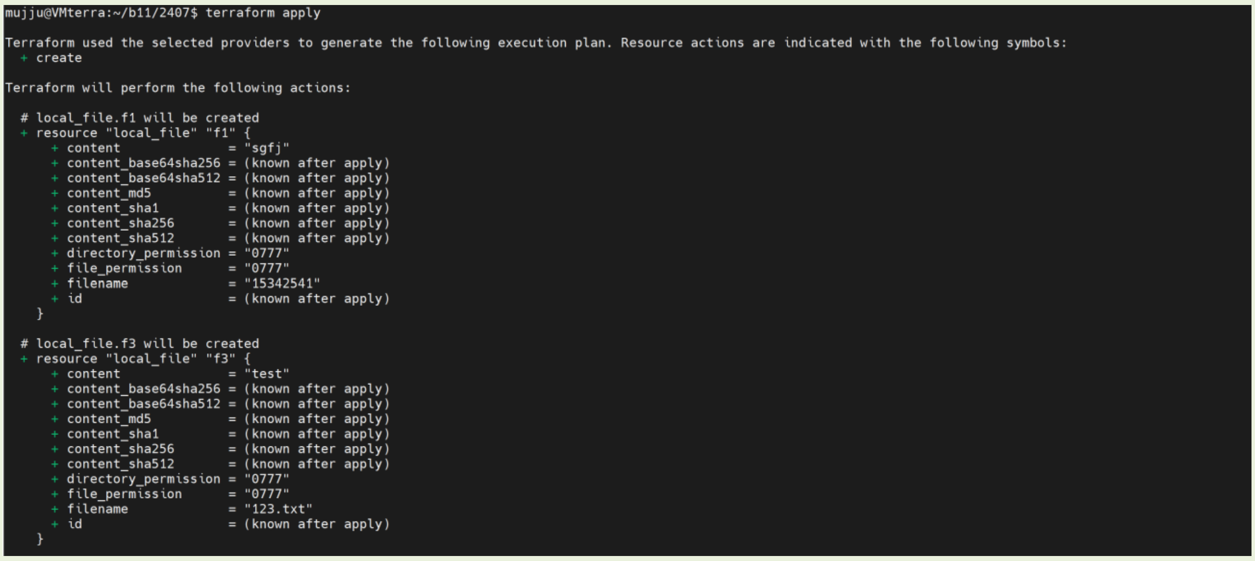
terraform init



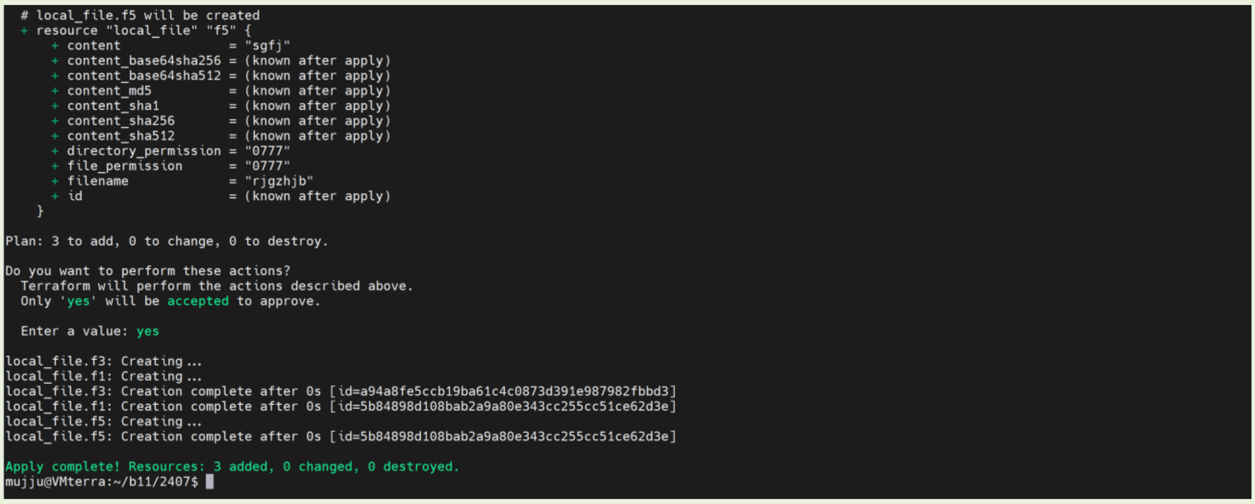
```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
mujju@VMterra:~/b11/2407$
```

Run Terraform apply

terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f1 will be created
  + resource "local_file" "f1" {
      + content              = "sgfj"
      + 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             = "15342541"
      + id                   = (known after apply)
    }

  # local_file.f3 will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "123.txt"
      + id                   = (known after apply)
    }
```



```
  # local_file.f5 will be created
  + resource "local_file" "f5" {
      + content              = "sgfj"
      + 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             = "rjgzhjb"
      + id                   = (known after apply)
    }

Plan: 3 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.f3: Creating...
local_file.f1: Creating...
local_file.f3: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f1: Creation complete after 0s [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]
local_file.f5: Creating...
local_file.f5: Creation complete after 0s [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

What happens here:

- **local_file.f5** will only be created after both **local_file.f3** and **local_file.f1** are completely created, regardless of whether f5 references any attribute from those resources directly or not.

Summary Table

Dependency Type	How Declared	Example Purpose	Terraform Behavior
Implicit (Direct)	Reference resource attribute	content = local_file.f3.id	Terraform infers creation order automatically
Explicit (Indirect)	Use depends_on meta-argument	Resource must wait for others even if no attributes are referenced (e.g., coordination or ordering needs)	Terraform enforces dependency explicitly

Additional Notes

- In general, we prefer **implicit dependencies** because they keep our configuration clean and declarative.
- Use **explicit dependencies** only when necessary to manage ordering in cases not naturally inferred by attribute references.
- Overuse of depends_on can complicate our configuration and lead to longer apply times.

Looping Over Resources in Terraform: count vs for_each

Terraform provides two primary ways to create multiple instances of a resource from collections:

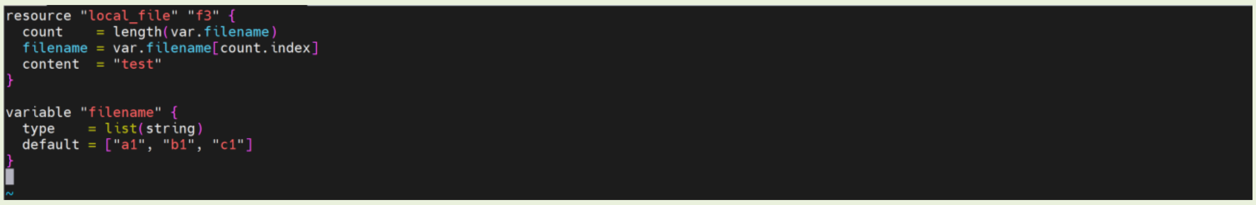
2. Using count

- **count** accepts an integer, typically the length of a list, and creates that many instances of the resource.
- Within the block, you access the current index with **count.index**.
- Best for simple lists or when we only need an index number.

Example:

Create a **main.tf** file using **vi main.tf** add the following configuration

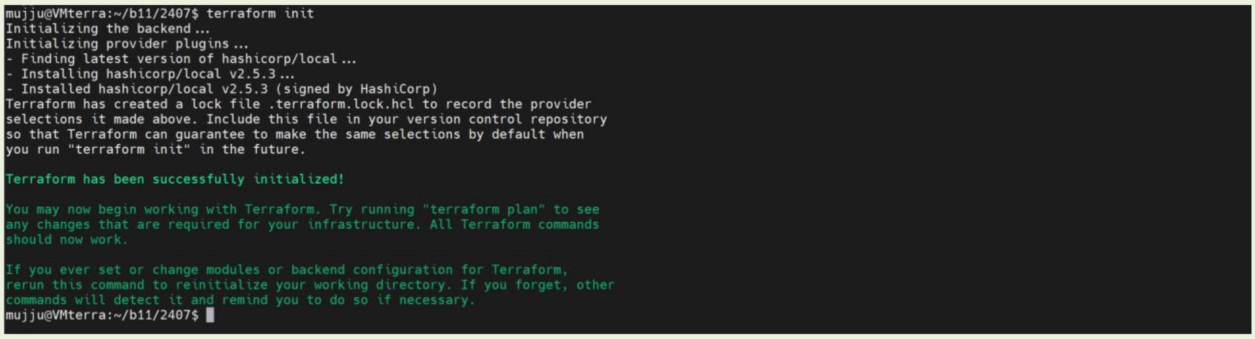
```
resource "local_file" "f3" {
  count    = length(var.filename)
  filename = var.filename[count.index]
  content  = "test"
}

variable "filename" {
  type    = list(string)
  default = ["a1", "b1", "c1"]
}
```



```
resource "local_file" "f3" {
  count    = length(var.filename)
  filename = var.filename[count.index]
  content  = "test"
}

variable "filename" {
  type    = list(string)
  default = ["a1", "b1", "c1"]
}

~
```

Initialize Terraform

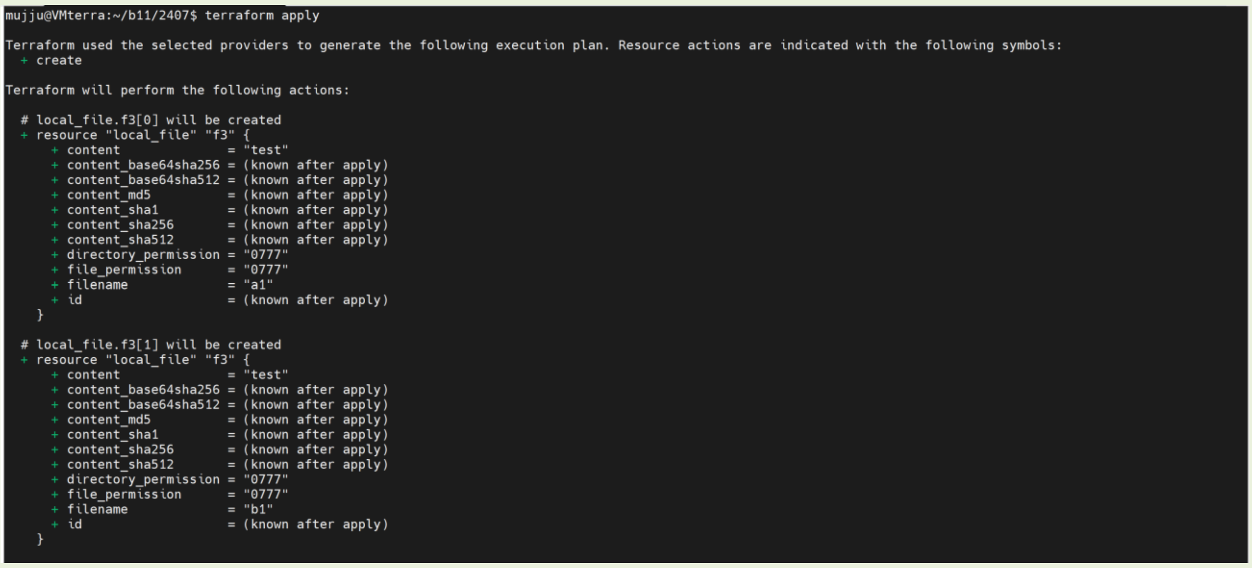
terraform init



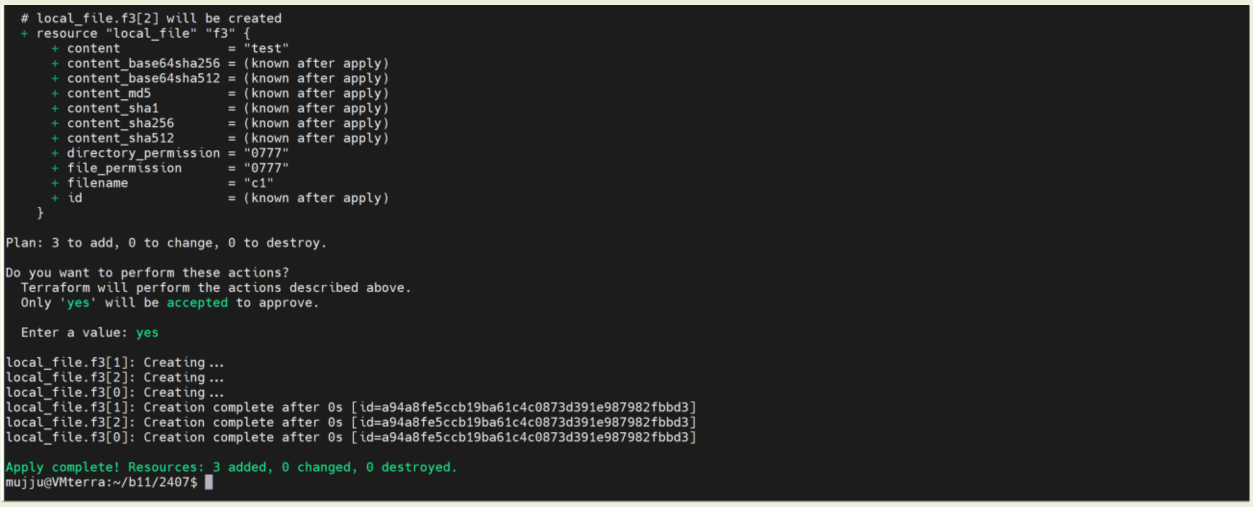
```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
mujju@VMterra:~/b11/2407$
```

Run Terraform apply

terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f3[0] will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "a1"
      + id                   = (known after apply)
    }

  # local_file.f3[1] will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "b1"
      + id                   = (known after apply)
    }
```



```
  # local_file.f3[2] will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "c1"
      + id                   = (known after apply)
    }

Plan: 3 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.f3[1]: Creating...
local_file.f3[2]: Creating...
local_file.f3[0]: Creating...
local_file.f3[1]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[2]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[0]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

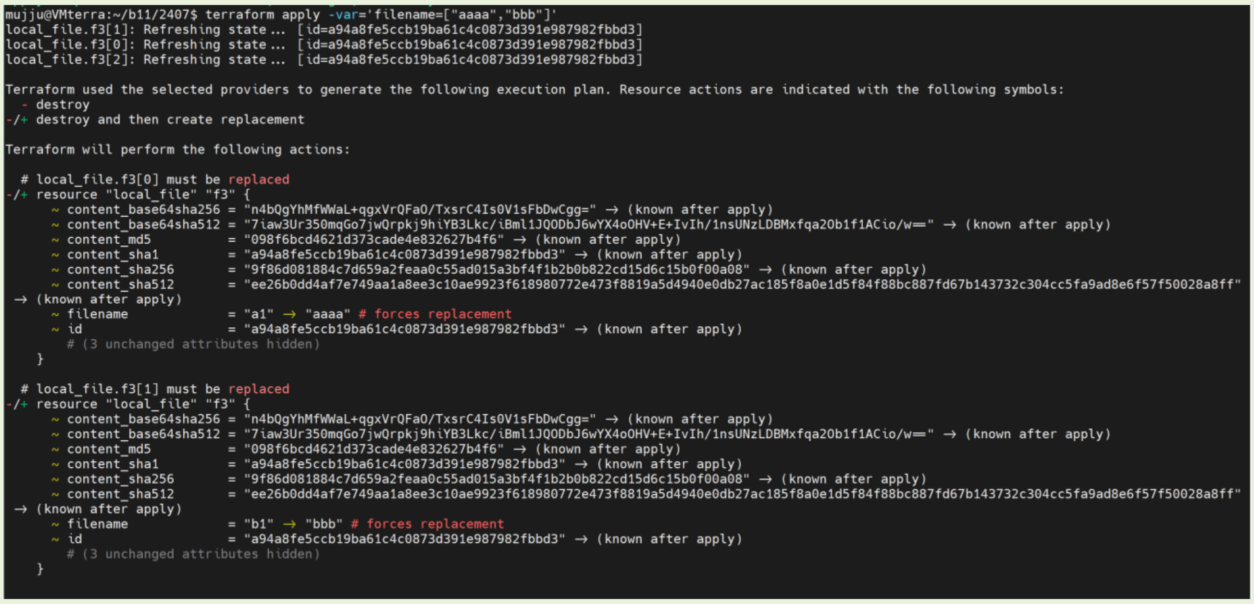
What happens:

- local_file.f3 creates 3 files: "a1", "b1", and "c1".
- Each resource instance's filename picks the element from the list at the current count.index.

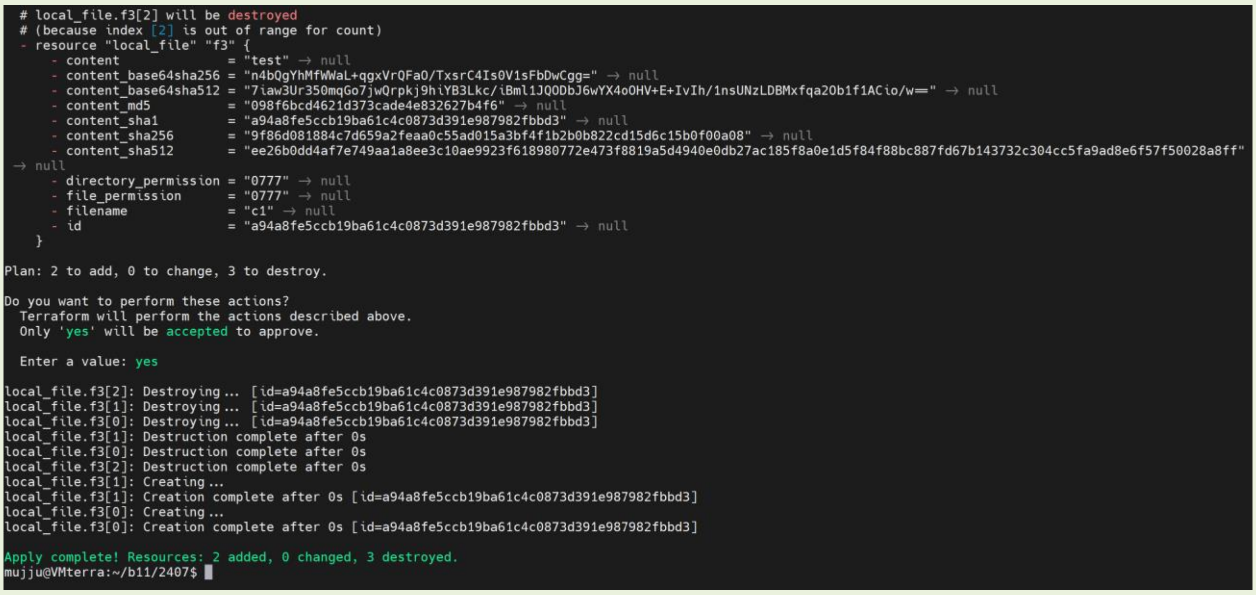
→Run the command

terraform apply -var='filename=["aaaa","bbb"]'

The command terraform apply -var='filename=["aaaa","bbb"]' is used to **apply the infrastructure changes defined in our Terraform configuration while overriding the variable filename with the list ["aaaa", "bbb"] at runtime**.



```
mujju@VMterra:~/b11/2407$ terraform apply -var='filename=["aaaa","bbb"]'
local_file.f3[1]: Refreshing state... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[0]: Refreshing state... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[2]: Refreshing state... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

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

Terraform will perform the following actions:

  # local_file.f3[0] must be replaced
-/+ resource "local_file" "f3" {
      ~ content_base64sha256 = "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=" → (known after apply)
      ~ content_base64sha512 = "7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w==" → (known after apply)
      ~ content_md5          = "098f6bcd4621d373cade4e832627b4f6" → (known after apply)
      ~ content_sha1         = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
      ~ content_sha256       = "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08" → (known after apply)
      ~ content_sha512       = "ee26b0dd4af7e749aa1a8ee3c10ae9923f618980772e473f8819a5d4940e0db27ac185f8a0e1d5f84f88bc887fd67b143732c304cc5fa9ad8e6f57f50028a8ff"
 → (known after apply)
      ~ filename             = "a1" → "aaaa" # forces replacement
      ~ id                   = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
        # (3 unchanged attributes hidden)
    }

  # local_file.f3[1] must be replaced
-/+ resource "local_file" "f3" {
      ~ content_base64sha256 = "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=" → (known after apply)
      ~ content_base64sha512 = "7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w==" → (known after apply)
      ~ content_md5          = "098f6bcd4621d373cade4e832627b4f6" → (known after apply)
      ~ content_sha1         = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
      ~ content_sha256       = "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08" → (known after apply)
      ~ content_sha512       = "ee26b0dd4af7e749aa1a8ee3c10ae9923f618980772e473f8819a5d4940e0db27ac185f8a0e1d5f84f88bc887fd67b143732c304cc5fa9ad8e6f57f50028a8ff"
 → (known after apply)
      ~ filename             = "b1" → "bbb" # forces replacement
      ~ id                   = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
        # (3 unchanged attributes hidden)
    }
```



```
  # local_file.f3[2] will be destroyed
  # (because index [2] is out of range for count)
  - resource "local_file" "f3" {
      - content              = "test" → null
      - content_base64sha256 = "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=" → null
      - content_base64sha512 = "7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w==" → null
      - content_md5          = "098f6bcd4621d373cade4e832627b4f6" → null
      - content_sha1         = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → null
      - content_sha256       = "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08" → null
      - content_sha512       = "ee26b0dd4af7e749aa1a8ee3c10ae9923f618980772e473f8819a5d4940e0db27ac185f8a0e1d5f84f88bc887fd67b143732c304cc5fa9ad8e6f57f50028a8ff"
 → null
      - directory_permission = "0777" → null
      - file_permission      = "0777" → null
      - filename             = "c1" → null
      - id                   = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → null
    }

Plan: 2 to add, 0 to change, 3 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.f3[2]: Destroying... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[1]: Destroying... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[0]: Destroying... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[1]: Destruction complete after 0s
local_file.f3[0]: Destruction complete after 0s
local_file.f3[2]: Destruction complete after 0s
local_file.f3[1]: Creating...
local_file.f3[1]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3[0]: Creating...
local_file.f3[0]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 2 added, 0 changed, 3 destroyed.
mujju@VMterra:~/b11/2407$
```

Here's what happens when we run:

terraform apply -var='filename=["aaaa","bbb"]'

What happens with our command:

- -var='filename=["aaaa","bbb"]' sets var.filename to the list ["aaaa", "bbb"].
- count = length(var.filename) means **count = 2**.
- Terraform will create:
    - **local_file.f3 with filename = "aaaa"**
    - **local_file.f3 with filename = "bbb"**

Terraform will:

- Show a plan to create 2 resources (one for each filename).
- Ask for confirmation.
- On approval, create two local files named "aaaa" and "bbb" (each with content "test").

Summary Table

Instance	filename
local_file.f3	aaaa
local_file.f3	bbb

- We can use this var-override syntax for any variable.
- If we were using **for_each** instead, the logic would be nearly identical: one resource instance per unique filename.

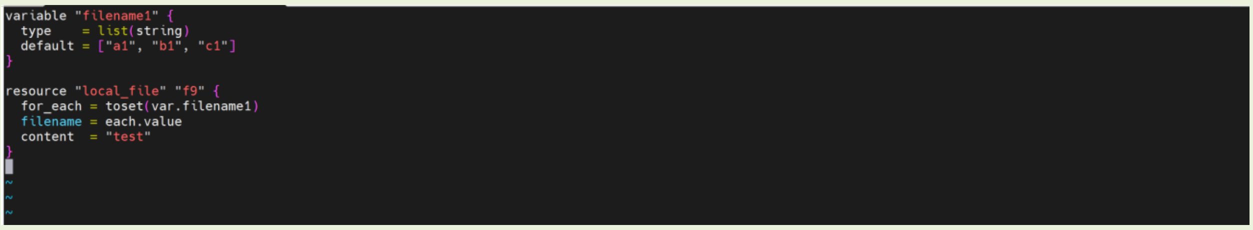
3. Using for_each

- **for_each** iterates over a set, map, or list of strings.
- The resource instances have access to each.key (if the collection is a map) or each.value (if the collection is a set or list).
- Preferred when you want to address resources by a key or when the input is a map.
- Ensures guarantees about resource instance keys/names that help with stable resource targeting.

Example:

Create a **main.tf** file using **vi main.tf** add the following configuration

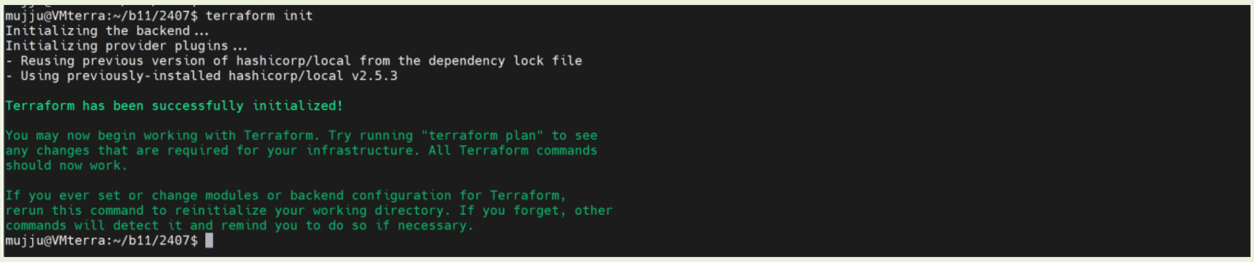
```
variable "filename1" {
  type    = list(string)
  default = ["a1", "b1", "c1"]
}

resource "local_file" "f9" {
  for_each = toset(var.filename1)
  filename = each.value
  content  = "test"
}
```



```
variable "filename1" {
  type    = list(string)
  default = ["a1", "b1", "c1"]
}

resource "local_file" "f9" {
  for_each = toset(var.filename1)
  filename = each.value
  content  = "test"
}
~
~
~
```

Initialize Terraform

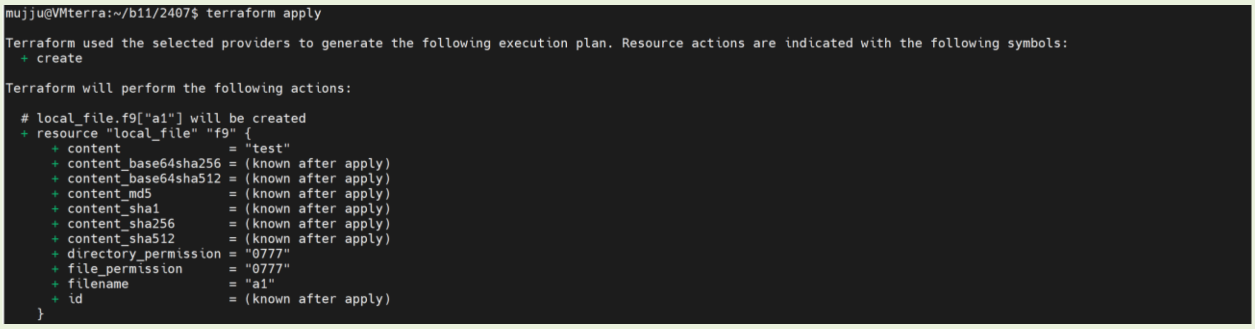
terraform init



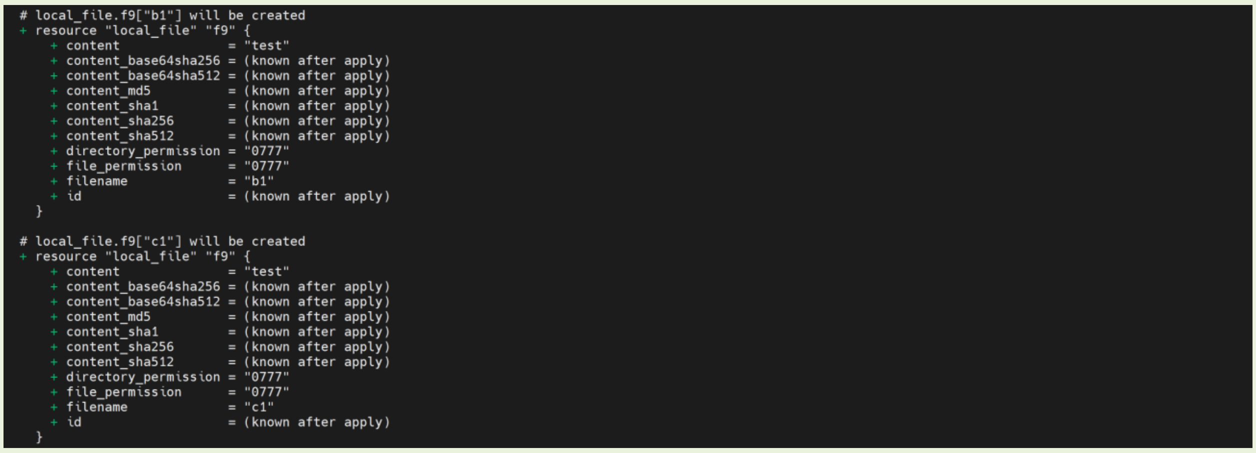
```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Using previously-installed hashicorp/local v2.5.3

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.
mujju@VMterra:~/b11/2407$
```

Run Terraform apply

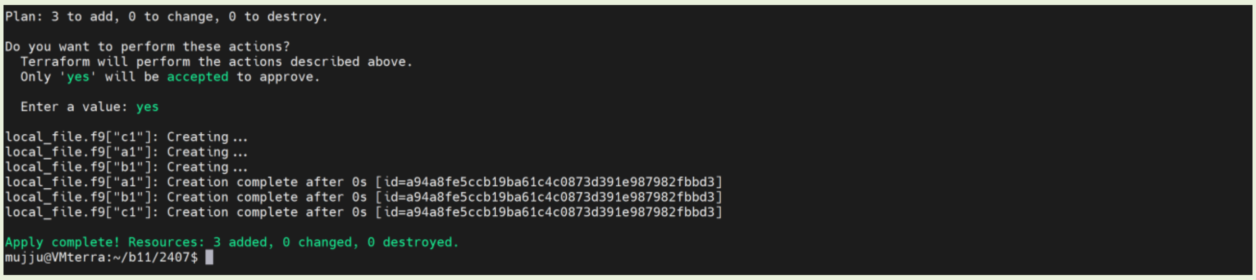
terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f9["a1"] will be created
  + resource "local_file" "f9" {
      + content              = "test"
      + 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             = "a1"
      + id                   = (known after apply)
    }
```



```
  # local_file.f9["b1"] will be created
  + resource "local_file" "f9" {
      + content              = "test"
      + 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             = "b1"
      + id                   = (known after apply)
    }

  # local_file.f9["c1"] will be created
  + resource "local_file" "f9" {
      + content              = "test"
      + 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             = "c1"
      + id                   = (known after apply)
    }
```



```
Plan: 3 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.f9["c1"]: Creating...
local_file.f9["a1"]: Creating...
local_file.f9["b1"]: Creating...
local_file.f9["a1"]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f9["b1"]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f9["c1"]: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

What happens:

- **toset(var.filename1)** converts the list to a set to remove duplicates and provide a unique key for each element.
- Terraform creates one **local_file.f9** resource per unique filename string in the set.
- Accessible as **local_file.f9["a1"], local_file.f9["b1"],** etc. — the keys come from the each.value in this case.

When to use count vs for_each

Feature	count	for_each
Collection type	Only integer count	Supports sets, lists, maps
Accessing elements	Indexed by count.index	By key each.key and value each.value
Stable resource keys	No (index changes on reordering)	Yes (keys stay the same if names/keys stable)
Use case example	Simple lists for identical resources	Resources keyed by meaningful string or map keys

Key Notes

- Avoid mixing count and for_each on the same resource block.
- Prefer for_each when resource uniqueness and stable identity matter.
- Use count for simple counts without need for identity keys.

4. Providers

The **provider meta-argument** in Terraform is used within a resource or module block to explicitly specify which provider configuration that resource or module should use, overriding Terraform's default provider selection behavior.

Key Points about the provider Meta-Argument

- By default, Terraform selects a provider configuration based on the resource type's prefix. For example, a resource type starting with aws_ uses the default AWS provider configuration.
- When we have **multiple configurations of the same provider**, such as managing resources across multiple regions or accounts, you define these additional provider configurations with **aliases**.
- The provider meta-argument lets you assign a specific **aliased** provider configuration to a resource or module. This controls exactly which provider instance manages that resource.

Example:

```
# Default provider configuration for AWS in us-west-1
provider "aws" {
  region = "us-west-1"
}

# Alternate provider configuration for AWS in us-west-2
provider "aws" {
  alias  = "usw2"
  region = "us-west-2"
}

resource "aws_instance" "example" {
  # Use the 'usw2' aliased provider here instead of default
  provider = aws.usw2

  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t3.micro"
}
```

In this example, the aws_instance.example resource will be created using the AWS provider configured for us-west-2 region, not the default us-west-1 provider.

Important Details

- The value of the provider meta-argument must follow the format <PROVIDER>.<ALIAS> (e.g., aws.usw2).
- It should **not be quoted** and cannot be an arbitrary expression because Terraform needs to resolve it during the dependency graph construction.
- Resources have an implicit dependency on their assigned provider to ensure proper initialization order.

Why Use the provider Meta-Argument?

- **Multi-region or multi-account setups:** When our infrastructure spans across multiple regions or accounts, each with distinct provider configurations.
- **Custom provider configurations:** When the resource needs to use a provider with special settings or credentials differing from the default.

Summary Table

Aspect	Description
Purpose	Assign a specific provider configuration to a resource/module
Format	<PROVIDER>.<ALIAS>, e.g., aws.usw2
Default Behavior	Provider inferred from resource type prefix, uses default config
Use case	Multi-region, multi-account, or special provider setups
Restrictions	Must be a direct provider alias reference, no expressions allowed

This provider meta-argument gives granular control over provider usage in complex Terraform projects.

5. Life cycles

Terraform lifecycle Meta-Argument — Explanation of Key Options

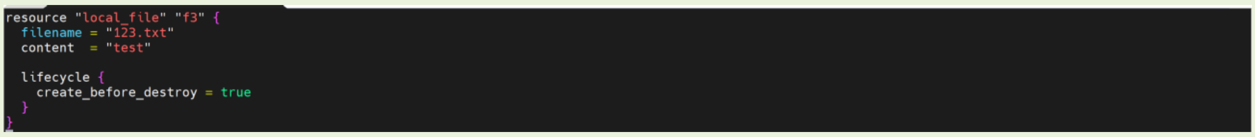
1. create_before_destroy

It create replacements before deleting the existing resource (reduces downtime).

Example:

Create a **main.tf** file using **vi main.tf** add the following configuration

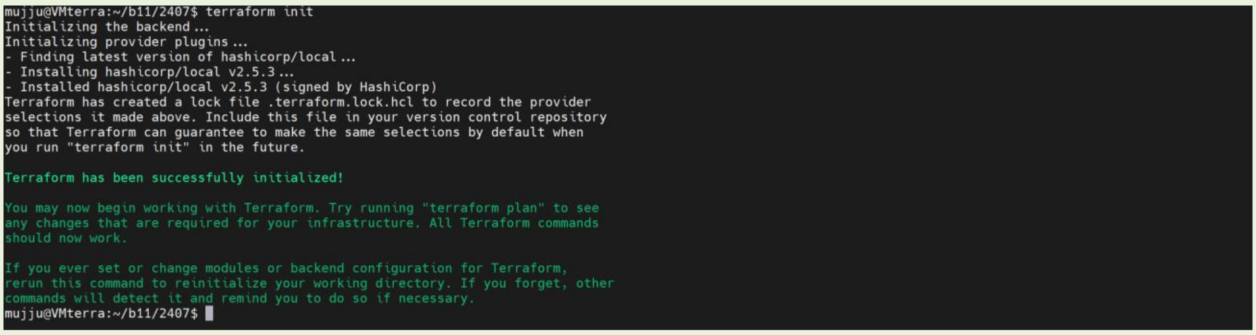
```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    create_before_destroy = true
  }
}
```



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    create_before_destroy = true
  }
}
```

Initialize Terraform

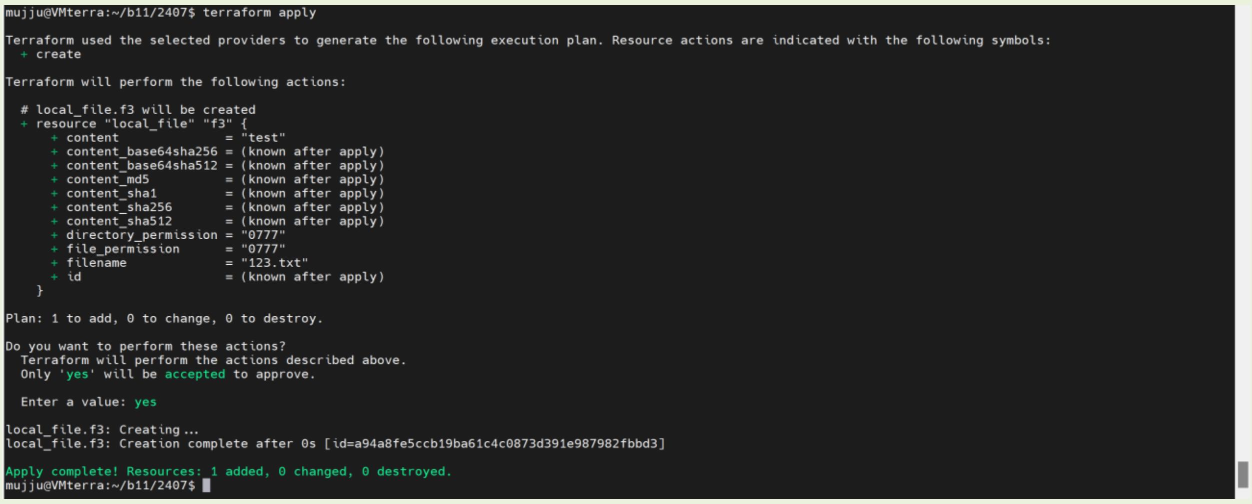
terraform init



```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (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.
mujju@VMterra:~/b11/2407$
```

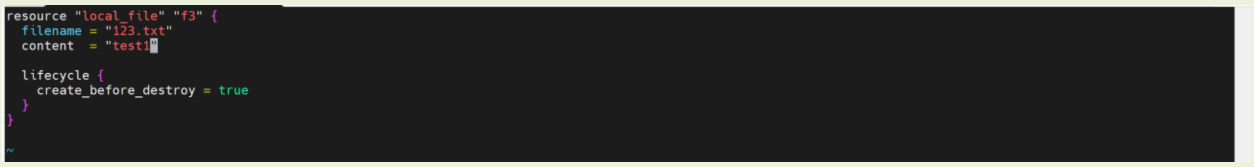
Run Terraform apply

terraform apply

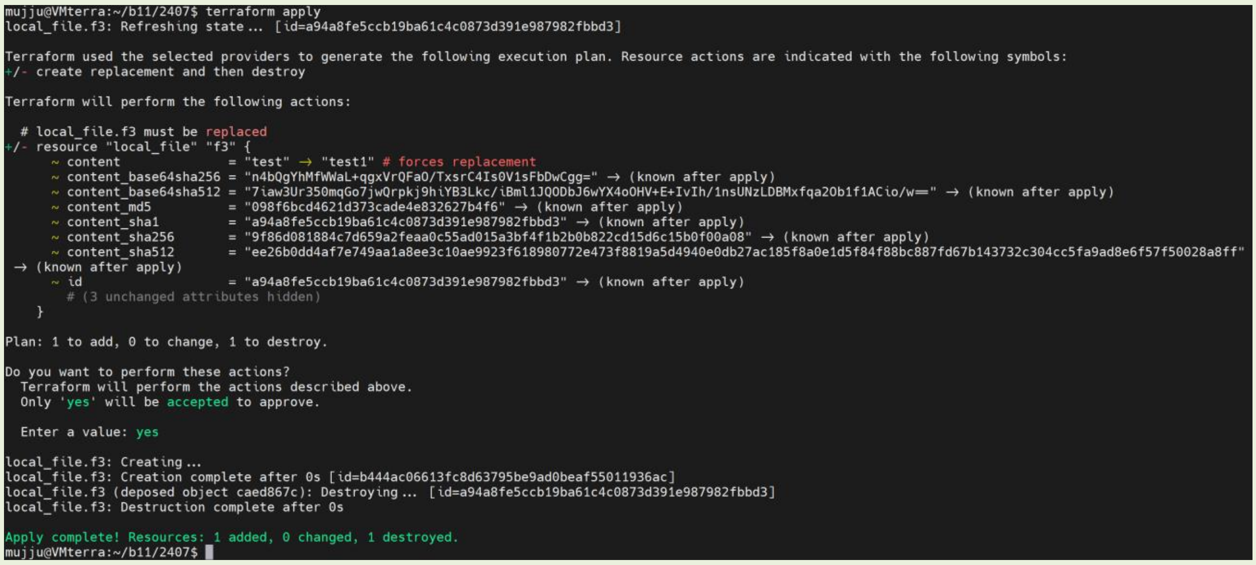


```
mujju@VMterra:~/b11/2407$ terraform apply

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.f3 will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "123.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.f3: Creating...
local_file.f3: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

When we make some changes in **main.tf file** and then run **terraform apply** it will first create the new resource and then deletes the older one as we have provided the condition **create before destroy**



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test1"

  lifecycle {
    create_before_destroy = true
  }
}
~
```



```
mujju@VMterra:~/b11/2407$ terraform apply
local_file.f3: Refreshing state... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

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

Terraform will perform the following actions:

  # local_file.f3 must be replaced
+/- resource "local_file" "f3" {
      ~ content              = "test" → "test1" # forces replacement
      ~ content_base64sha256 = "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=" → (known after apply)
      ~ content_base64sha512 = "7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w==" → (known after apply)
      ~ content_md5          = "098f6bcd4621d373cade4e832627b4f6" → (known after apply)
      ~ content_sha1         = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
      ~ content_sha256       = "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08" → (known after apply)
      ~ content_sha512       = "ee26b0dd4af7e749aa1a8ee3c10ae9923f618980772e473f8819a5d4940e0db27ac185f8a0e1d5f84f88bc887fd67b143732c304cc5fa9ad8e6f57f50028a8ff"
 → (known after apply)
      ~ id                   = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (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.f3: Creating...
local_file.f3: Creation complete after 0s [id=b444ac06613fc8d63795be9ad0beaf55011936ac]
local_file.f3 (deposed object caed867c): Destroying... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3: Destruction complete after 0s

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
mujju@VMterra:~/b11/2407$
```

What create_before_destroy = true does:

- When a change to this resource requires replacing it (e.g., changing filename, which often triggers replacement for a file resource), Terraform will **create the new resource before destroying the old one**.
- This helps **avoid downtime or gaps**, ensuring the new file exists before the old one is deleted.
- For local_file resources, this could mean creating the new file with the updated filename/content before removing the old file.

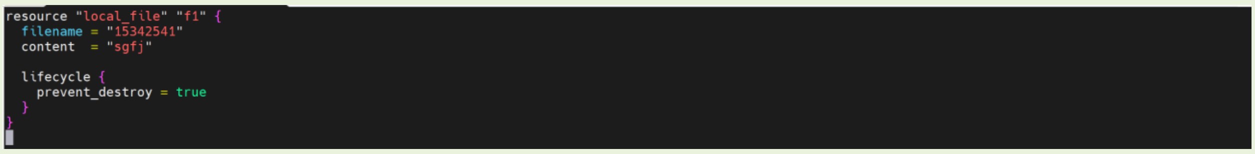
- This option only applies when a resource must be replaced (not updated in-place).
- It helps ensure zero downtime, but for some resource types it may not be meaningful if the resource itself cannot exist simultaneously (which typically isn't the case for local files).
- Without this, Terraform will destroy the old resource first, then create the new one.

2. prevent_destroy

- **Definition:** Prevents Terraform from destroying the resource, causing Terraform to error if the resource would need to be deleted.

Create a **main.tf** file using **vi main.tf** add the following configuration

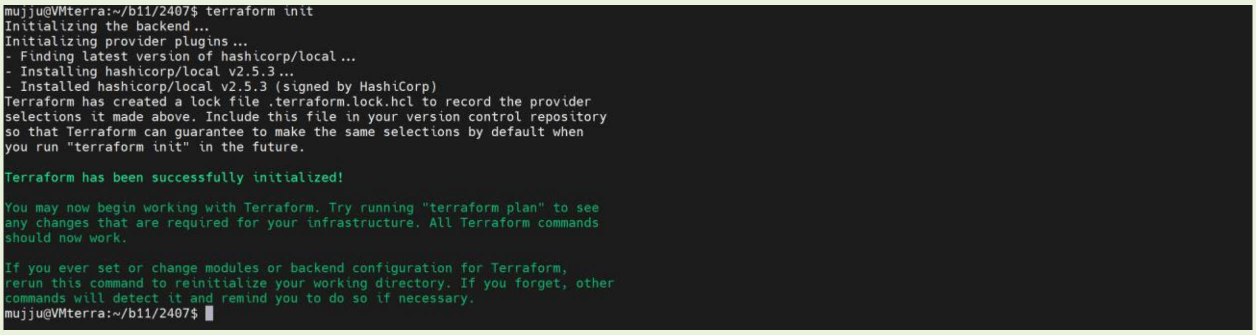
```
resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"

  lifecycle {
    prevent_destroy = true
  }
}
```



```
resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"

  lifecycle {
    prevent_destroy = true
  }
}
```

Initialize Terraform

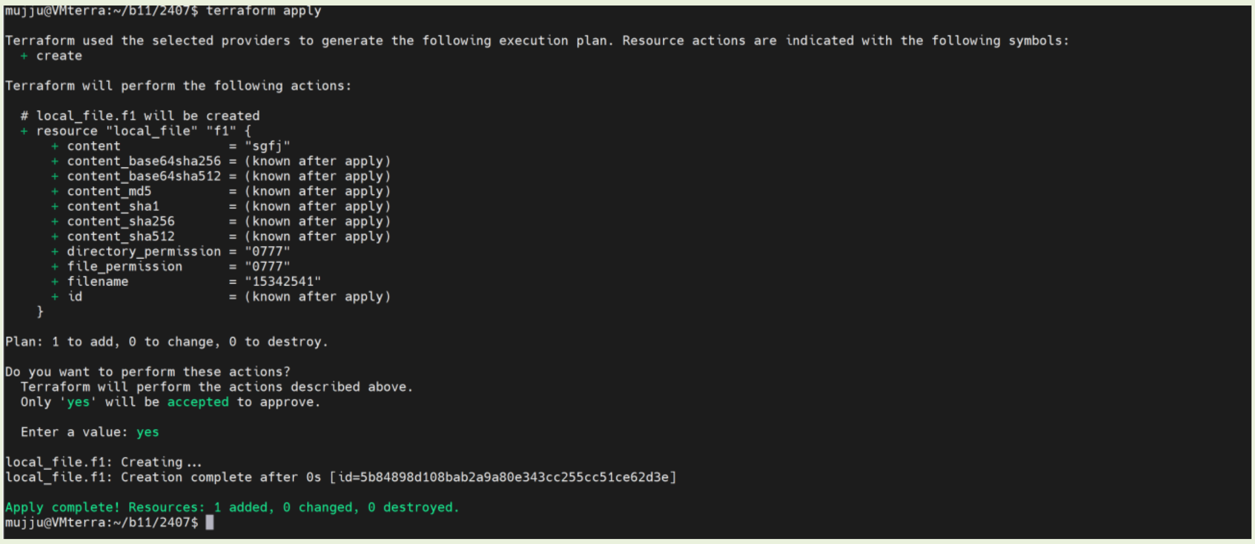
terraform init



```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (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.
mujju@VMterra:~/b11/2407$
```

Run Terraform apply

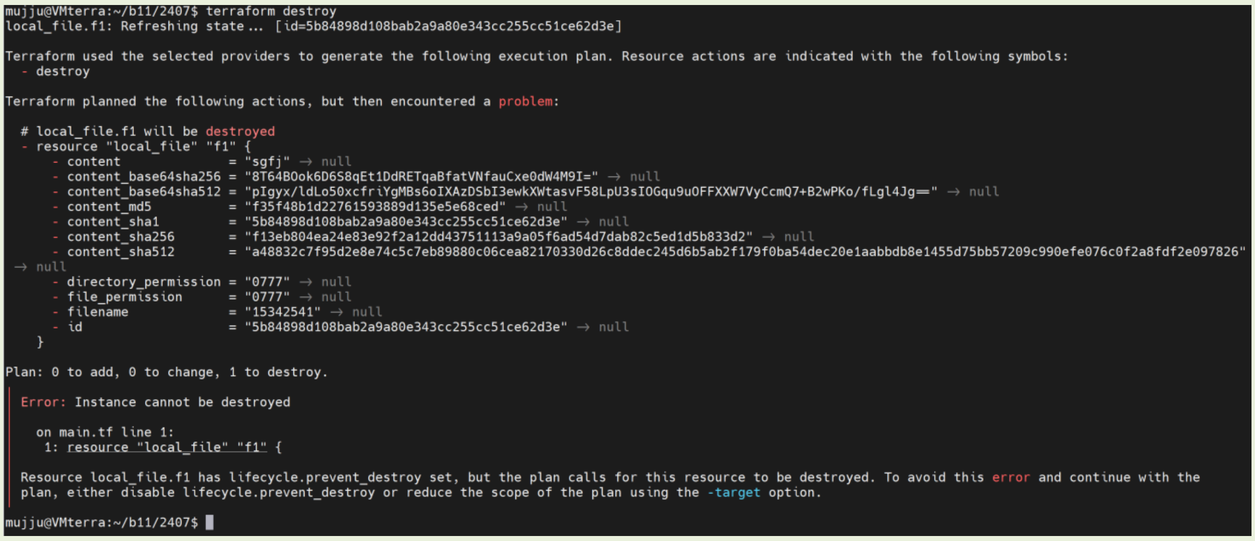
terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f1 will be created
  + resource "local_file" "f1" {
      + content              = "sgfj"
      + 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             = "15342541"
      + 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.f1: Creating...
local_file.f1: Creation complete after 0s [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

Run Terraform destroy

terraform destroy



```
mujju@VMterra:~/b11/2407$ terraform destroy
local_file.f1: Refreshing state... [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]

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

Terraform planned the following actions, but then encountered a problem:

  # local_file.f1 will be destroyed
  - resource "local_file" "f1" {
      - content              = "sgfj" -> null
      - content_base64sha256 = "8T64BOok6D6S8qEt1DdRETqaBfatVNfauCxe0dW4M9I=" -> null
      - content_base64sha512 = "pIgyx/ldLo50xcfriYgMBs6oIXAzDSbI3ewkXWtasvF58LpU3sIOGqu9uOFFXXW7VyCcmQ7+B2wPKo/fLgl4Jg==" -> null
      - content_md5          = "f35f48b1d22761593889d135e5e68ced" -> null
      - content_sha1         = "5b84898d108bab2a9a80e343cc255cc51ce62d3e" -> null
      - content_sha256       = "f13eb804ea24e83e92f2a12dd43751113a9a05f6ad54d7dab82c5ed1d5b833d2" -> null
      - content_sha512       = "a48832c7f95d2e8e74c5c7eb89880c06cea82170330d26c8ddec245d6b5ab2f179f0ba54dec20e1aabbdb8e1455d75bb57209c990efe076c0f2a8fdf2e097826"
 -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = "15342541" -> null
      - id                   = "5b84898d108bab2a9a80e343cc255cc51ce62d3e" -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.
│
│ Error: Instance cannot be destroyed
│
│   on main.tf line 1:
│    1: resource "local_file" "f1" {
│
│ Resource local_file.f1 has lifecycle.prevent_destroy set, but the plan calls for this resource to be destroyed. To avoid this error and continue with the
│ plan, either disable lifecycle.prevent_destroy or reduce the scope of the plan using the -target option.

mujju@VMterra:~/b11/2407$ █
```

What prevent_destroy = true does:

- It **prevents Terraform from destroying this resource** accidentally.
- If we try to run terraform destroy or remove this resource from our configuration, Terraform will fail the operation and show an error.
- This is a safeguard to protect critical or important resources from being deleted unintentionally.

Use Cases:

- When the resource represents something important that we don't want to lose by mistake.
- Useful in production environments or for resources where accidental destruction would be costly or harmful.

What happens in practice?

- If you attempt to delete the resource or perform actions that require replacement (which includes destroying the resource), Terraform will refuse unless we remove prevent_destroy or explicitly override it.

Protects important or critical resources that should never be destroyed without explicit manual action.

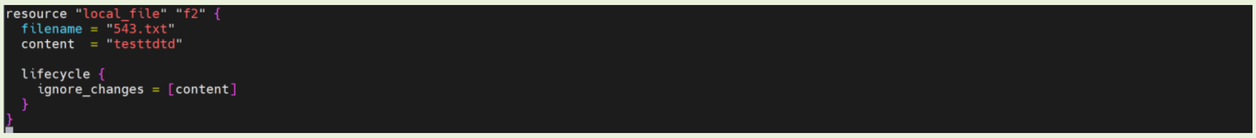
3. ignore_changes

- **Definition:** Tells Terraform to ignore updates made to specified resource attributes after initial creation.
- **Effect on our resource:**
  If we specify ignore_changes = ["content"], and the file content is updated outside Terraform or changed in our configuration, Terraform will **not** consider those differences as requiring an update during apply.
  This is useful if, for example, the file content is managed manually or by another process.
- **Example:**
  If you change content outside Terraform, the next terraform apply will not overwrite it back.
- **Note:**
  Be careful ignoring changes means Terraform will not detect drifts for the specified attributes.

To add the ignore_changes setting in the lifecycle block for our local_file.f2 resource, you specify which attributes Terraform should ignore changes for. For example, if you want Terraform to ignore changes to the content attribute (so changes made outside Terraform won't trigger a resource update), you can define it like this:

Create a **main.tf** file using **vi main.tf** add the following configuration

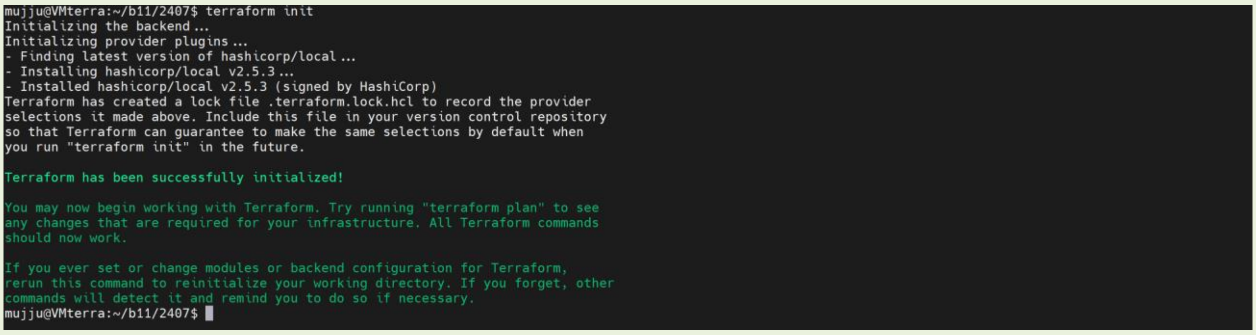
```
resource "local_file" "f2" {
  filename = "543.txt"
  content  = "testtdtd"

  lifecycle {
    ignore_changes = [ content ]
  }
}
```



```
resource "local_file" "f2" {
  filename = "543.txt"
  content  = "testtdtd"

  lifecycle {
    ignore_changes = [content]
  }
}
```

Initialize Terraform

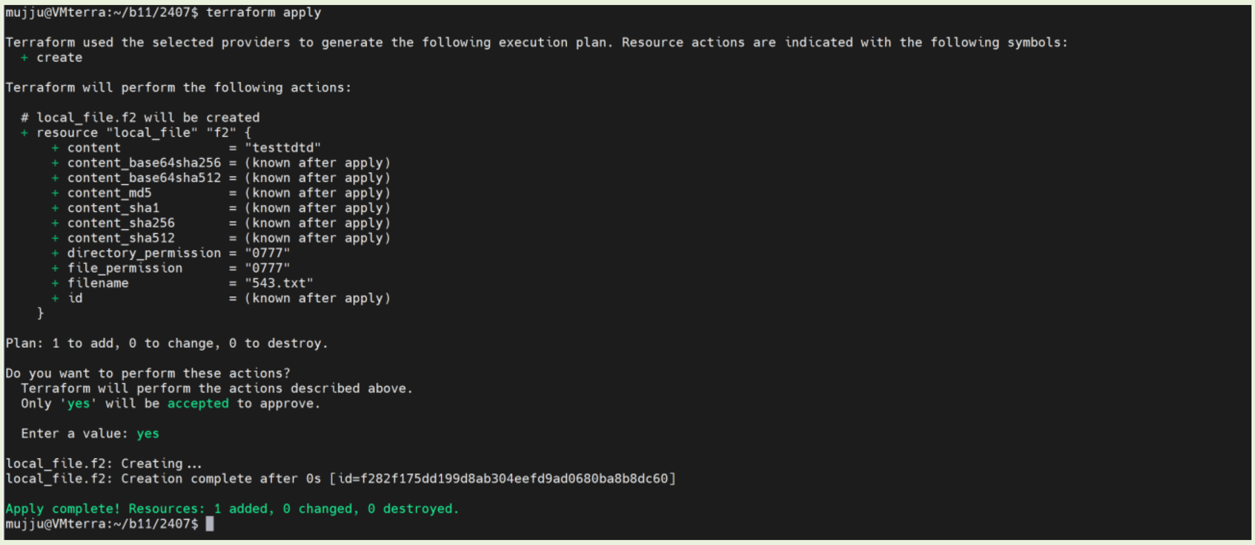
terraform init



```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (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.
mujju@VMterra:~/b11/2407$
```

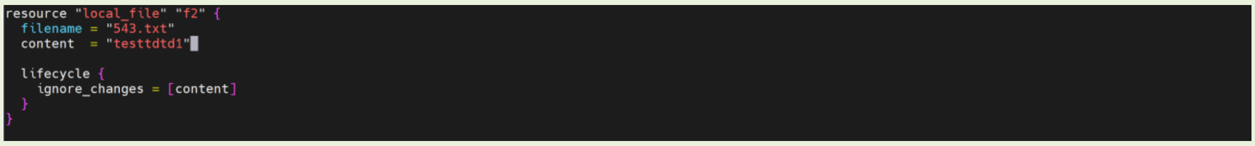
Run Terraform apply

terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f2 will be created
  + resource "local_file" "f2" {
      + content              = "testtdtd"
      + 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             = "543.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.f2: Creating...
local_file.f2: Creation complete after 0s [id=f282f175dd199d8ab304eefd9ad0680ba8b8dc60]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$
```

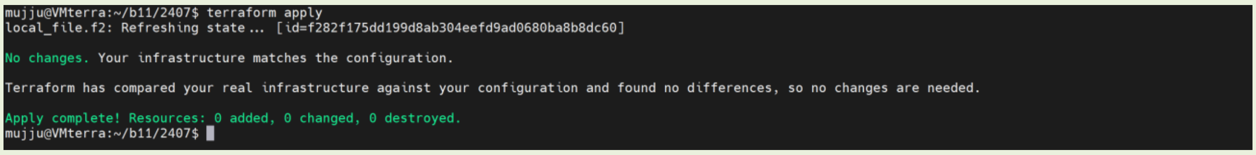
Make some changes in **main.tf** file



```
resource "local_file" "f2" {
  filename = "543.txt"
  content  = "testtdtd1"

  lifecycle {
    ignore_changes = [content]
  }
}
```

Run Terraform apply

terraform apply



```
mujju@VMterra:~/b11/2407$ terraform apply
local_file.f2: Refreshing state... [id=f282f175dd199d8ab304eefd9ad0680ba8b8dc60]

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.
mujju@VMterra:~/b11/2407$
```

Explanation:

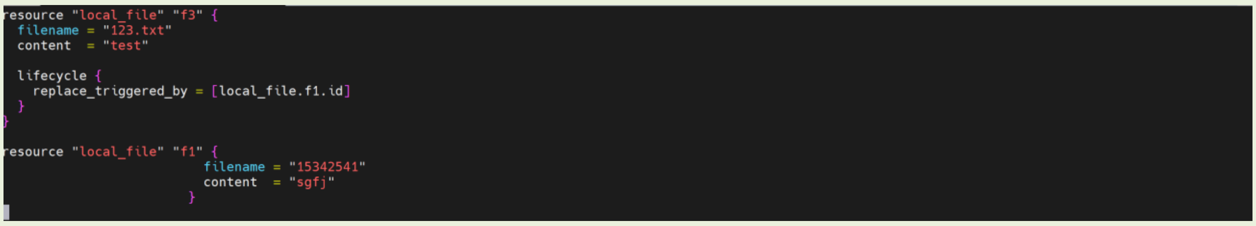
- Terraform will ignore any differences detected in the content field after initial creation.
- This means if the file content changes outside Terraform or we update the content value in our configuration, Terraform will NOT try to update or replace the resource during terraform apply.
- We made a change here **content = “testtdtd1**

4. replace_triggered_by

- **Definition:** Forces replacement of the resource if the referenced resource's attribute changes, even when this resource's own attributes do not change.

Effect on our resource:
Our local_file.f3 resource declaration with the replace_triggered_by lifecycle argument looks correct. Here is a quick explanation of what we have:

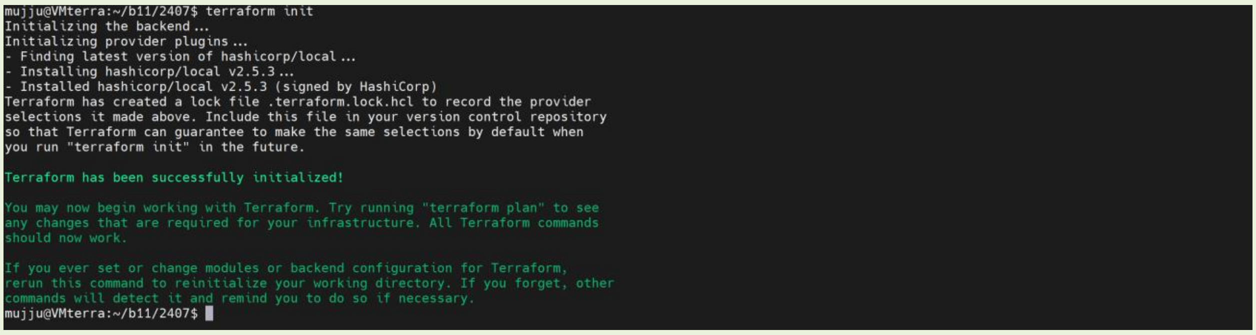
```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    replace_triggered_by = [local_file.f1.id]
  }
}
resource "local_file" "f1" {
  filename = "15342541"
  content  = "sgfj"
}
```



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    replace_triggered_by = [local_file.f1.id]
  }
}

resource "local_file" "f1" {
                         filename = "15342541"
                         content  = "sgfj"
                       }
```

Initialize Terraform

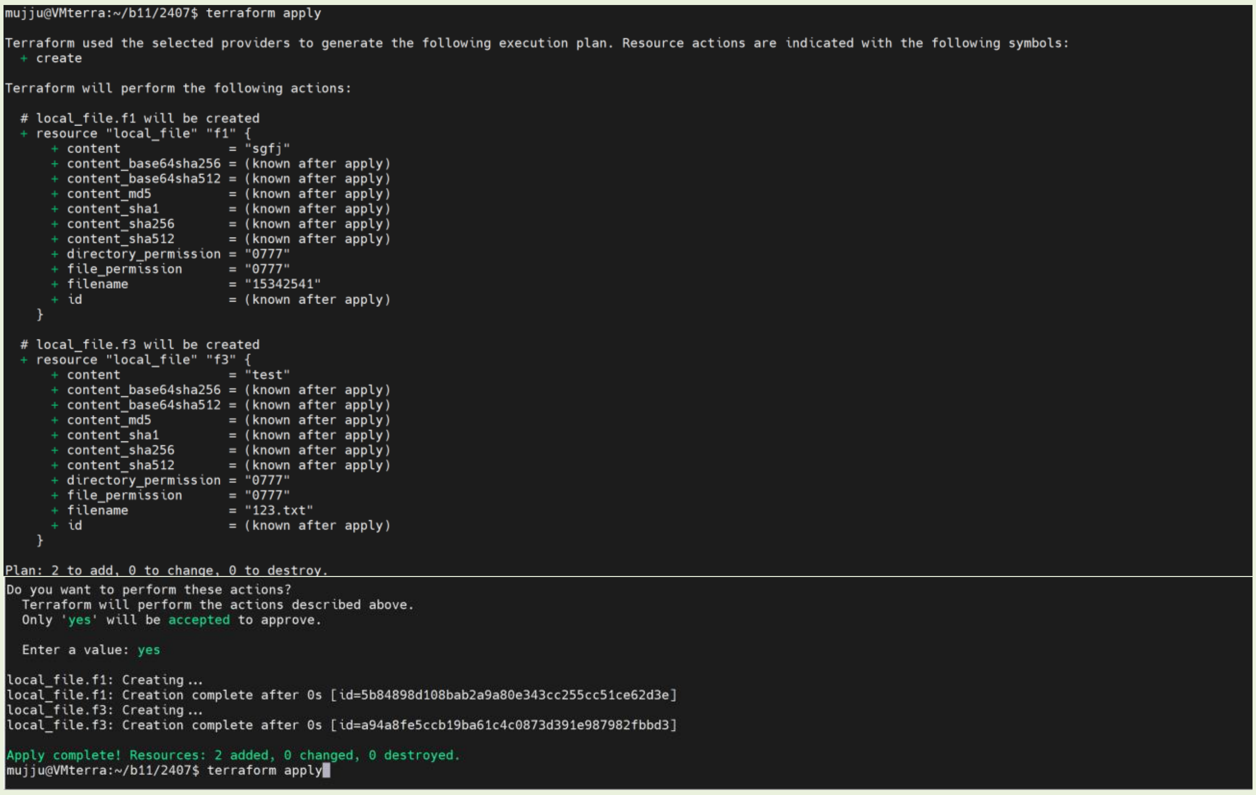
terraform init



```
mujju@VMterra:~/b11/2407$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (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.
mujju@VMterra:~/b11/2407$
```

Run Terraform apply

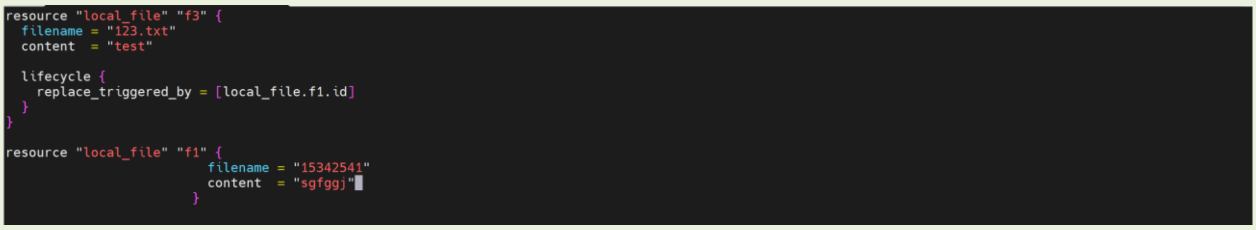
terraform apply



```
mujju@VMterra:~/b11/2407$ 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.f1 will be created
  + resource "local_file" "f1" {
      + content              = "sgfj"
      + 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             = "15342541"
      + id                   = (known after apply)
    }

  # local_file.f3 will be created
  + resource "local_file" "f3" {
      + content              = "test"
      + 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             = "123.txt"
      + id                   = (known after apply)
    }

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.f1: Creating...
local_file.f1: Creation complete after 0s [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]
local_file.f3: Creating...
local_file.f3: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/2407$ terraform apply
```

Make changes in **main.tf** file

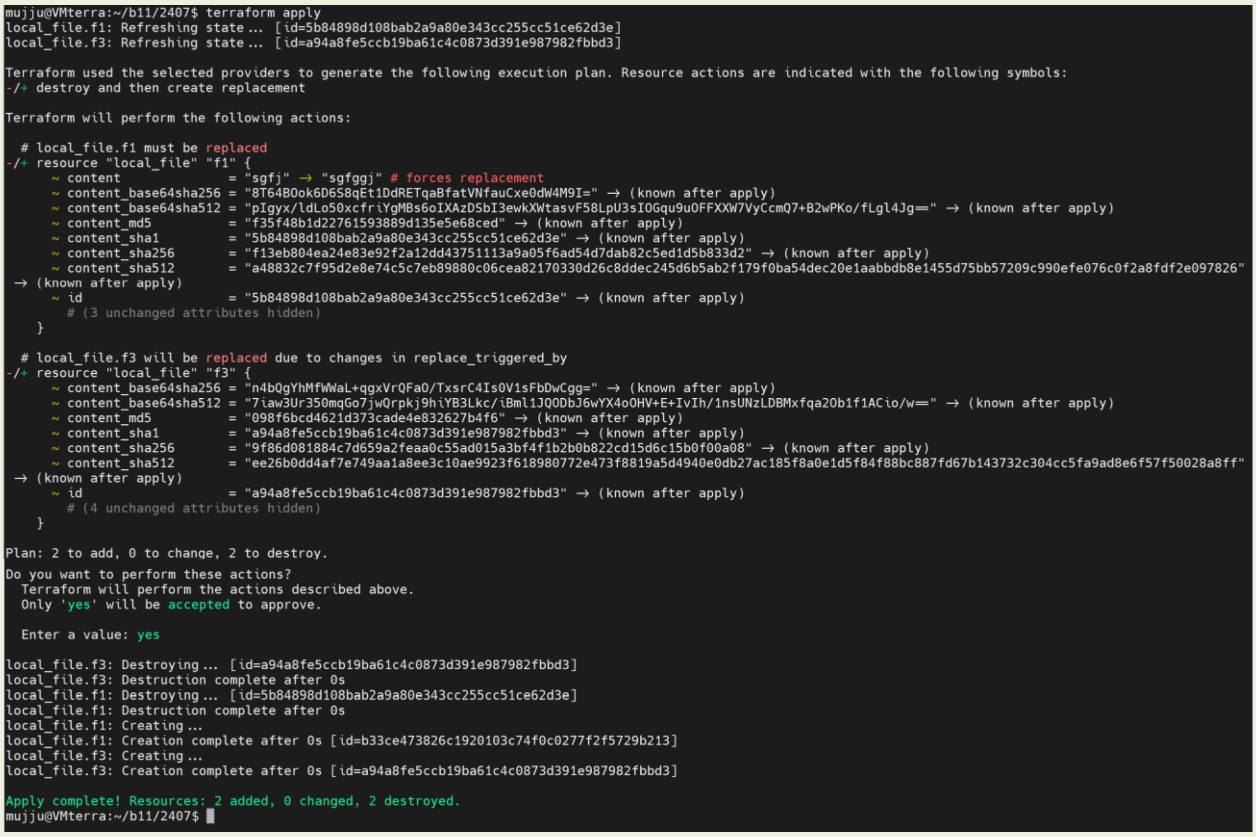
We made changes in resource of **local_file.f1**



```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    replace_triggered_by = [local_file.f1.id]
  }
}

resource "local_file" "f1" {
                          filename = "15342541"
                          content  = "sgfggj"
                        }
```

Run Terraform apply

terraform apply



```
mujju@VMterra:~/b11/2407$ terraform apply
local_file.f1: Refreshing state... [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]
local_file.f3: Refreshing state... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

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.f1 must be replaced
-/+ resource "local_file" "f1" {
      ~ content              = "sgfj" → "sgfggj" # forces replacement
      ~ content_base64sha256 = "8T64BOok6D6S8qEt1DdRETqaBfatVNfauCxe0dW4M9I=" → (known after apply)
      ~ content_base64sha512 = "pIgyx/ldLo50xcfriYgMBs6oIXAzDSbI3ewkXWtasvF58LpU3sIOGqu9uOFFXXW7VyCcmQ7+B2wPKo/fLgl4Jg==" → (known after apply)
      ~ content_md5          = "f35f48b1d22761593889d135e5e68ced" → (known after apply)
      ~ content_sha1         = "5b84898d108bab2a9a80e343cc255cc51ce62d3e" → (known after apply)
      ~ content_sha256       = "f13eb804ea24e83e92f2a12dd43751113a9a05f6ad54d7dab82c5ed1d5b833d2" → (known after apply)
      ~ content_sha512       = "a48832c7f95d2e8e74c5c7eb89880c06cea82170330d26c8ddec245d6b5ab2f179f0ba54dec20e1aabbdb8e1455d75bb57209c990efe076c0f2a8fdf2e097826"
 → (known after apply)
      ~ id                   = "5b84898d108bab2a9a80e343cc255cc51ce62d3e" → (known after apply)
        # (3 unchanged attributes hidden)
    }

  # local_file.f3 will be replaced due to changes in replace_triggered_by
-/+ resource "local_file" "f3" {
      ~ content_base64sha256 = "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=" → (known after apply)
      ~ content_base64sha512 = "7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w==" → (known after apply)
      ~ content_md5          = "098f6bcd4621d373cade4e832627b4f6" → (known after apply)
      ~ content_sha1         = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
      ~ content_sha256       = "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08" → (known after apply)
      ~ content_sha512       = "ee26b0dd4af7e749aa1a8ee3c10ae9923f618980772e473f8819a5d4940e0db27ac185f8a0e1d5f84f88bc887fd67b143732c304cc5fa9ad8e6f57f50028a8ff"
 → (known after apply)
      ~ id                   = "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" → (known after apply)
        # (4 unchanged attributes hidden)
    }

Plan: 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.f3: Destroying... [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]
local_file.f3: Destruction complete after 0s
local_file.f1: Destroying... [id=5b84898d108bab2a9a80e343cc255cc51ce62d3e]
local_file.f1: Destruction complete after 0s
local_file.f1: Creating...
local_file.f1: Creation complete after 0s [id=b33ce473826c1920103c74f0c0277f2f5729b213]
local_file.f3: Creating...
local_file.f3: Creation complete after 0s [id=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3]

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.
mujju@VMterra:~/b11/2407$
```

What this means:

- Terraform will **replace (destroy and recreate)** the **local_file.f3** resource whenever the ID of **local_file.f1** changes.
- The replace_triggered_by meta-argument creates an implicit dependency such that changes to the specified attribute (local_file.f1.id) trigger a resource replacement, even if f3's own configuration did not change.
- This is useful if f3 logically depends on f1, and we want to force its recreation when f1 is replaced.

Important notes:

- **local_file.f1** must already be defined in our Terraform configuration; otherwise, Terraform will error.
- The attribute .id typically changes if f1 itself is recreated.
- We can add multiple triggers if needed, e.g.:

  replace_triggered_by = [local_file.f1.id, local_file.f2.filename]

Summary:

This lifecycle block tells Terraform:

"If the id of local_file.f1 changes, force a replacement of local_file.f3 on the next apply."

Use this to ensure a dependent resource is replaced if another resource it relies on changes.

Summary Table applied to our resources

Lifecycle Option	Action Description	Effect on local_file.f3 Example
create_before_destroy	Create the new resource before destroying the old one	Avoids downtime between file rewrites
prevent_destroy	Blocks resource deletion	Protects the local file from accidental deletion
ignore_changes	Ignore changes to specified attributes (e.g., content)	Terraform won’t update the file content even if changed
replace_triggered_by	Force recreation when a dependent resource attribute changes	f3 will be replaced if f1.id changes

Example full resource with all options applied

```
resource "local_file" "f3" {
  filename = "123.txt"
  content  = "test"

  lifecycle {
    create_before_destroy = true
    prevent_destroy       = true
    ignore_changes        = ["content"]
    replace_triggered_by  = [local_file.f1.id]
  }
}
```

Note: These lifecycle options are independent we can use one or any combination depending on our infrastructure needs.