

## TASKS ON TERRAFORM

❖ **NAME: S.MUZZAMMIL HUSSAIN**

❖ **DATE: 18/07/2025**

❖ BATCH: 11

❖ NO.OF TASKS: 1

## Task 1. In Terraform configuration make a mistake in configuration, run validate, correct it and execute

Here is a practical **example of making a mistake in a Terraform configuration for Azure, detecting it with terraform validate, correcting it, and executing the deployment:**

We created a directory named **b11**, in **b11** we created another directory with name **1807**, here we create a file named **main.tf** using **vi main.tf**

```
mujju@VMterra:~$ mkdir b11
mujju@VMterra:~$ cd b11
mujju@VMterra:~/b11$ mkdir 1807
mujju@VMterra:~/b11$ cd 1807
mujju@VMterra:~/b11/1807$ vi main.tf
mujju@VMterra:~/b11/1807$
```

## 1. Write a Terraform Configuration with an Error

Suppose we create a file named abc.txt.

We create a file **main.tf** but with a typo(mistake) and save it using **esc:wq**

```
resource "local_file" "filecreation" {
  filename = "abc.txt"
  content  = "This is a sample file created by Terraform."
}
```

```
resource "local_file" "filecreation" {
  filename = "abc.txt"
  content  = "This is a sample file created by Terraform."
}
```

## 2. Validate the Configuration

In your terminal, run the following:

```
terraform init
terraform validate
```

You will receive an error:

```
mujju@VMterra:~/b11/1807$ 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 made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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:~/b11/1807$ terraform validate

Error: Missing required argument

  on main.tf line 1, in resource "local_file" "filecreation":
   1: resource "local_file" "filecreation" {

The argument "filename" is required, but no definition was found.

Error: Unsupported argument

  on main.tf line 2, in resource "local_file" "filecreation":
   2:   filename = "abc.txt"

An argument named "filename" is not expected here. Did you mean "filename"?
mujju@VMterra:~/b11/1807$
```

### terraform init

#### Purpose:

Initializes a new or existing Terraform configuration.

#### What It Does:

- Downloads provider plugins (e.g., azurearm, aws, google).
- Sets up the .terraform directory with backend settings, provider plugins, and modules.
- Prepares your environment for other Terraform commands like plan and apply.

#### When to Use:

- The first time you set up a new Terraform configuration.
- Anytime you add, remove, or change a provider or module in your configuration.

### terraform validate

#### Purpose:

Validates whether the syntax and structure of your Terraform configuration are correct.

#### What It Does:

- Checks for basic syntax errors.
- Ensures all required arguments and blocks are present.
- Does **NOT** check real-world resource availability (for that, use terraform plan).

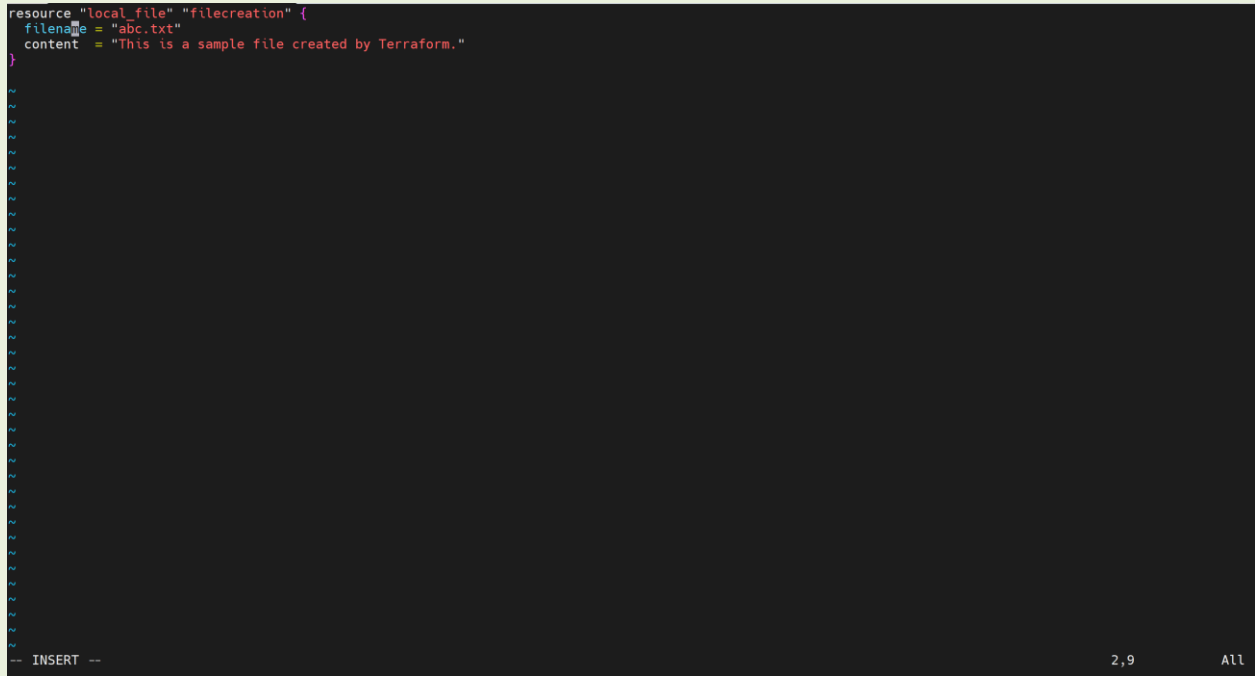
#### When to Use:

- Before running terraform apply to catch mistakes early.
- In CI/CD pipelines as a pre-check step.

### 3. Fix the Error

Correct the typo in **main.tf** by running command **vi main.tf** and correct the mistake as follows

```
resource "local_file" "filecreation" {  
  filename = "abc.txt"  
  content  = "This is a sample file created by Terraform."  
}
```

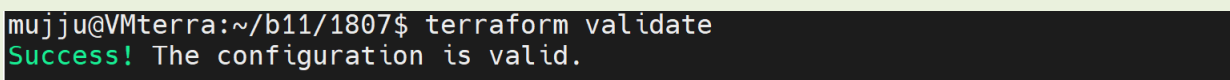


```
resource "local_file" "filecreation" {  
  filename = "abc.txt"  
  content  = "This is a sample file created by Terraform."  
}
```

### 4. Re-validate

Run the command

**terraform validate**



```
mujju@VMterra:~/b11/1807$ terraform validate  
Success! The configuration is valid.
```

Success! The configuration is valid.

### 5. Plan and Apply

**terraform plan**

- Think of it as a **preview** or **dry run**.
- It shows:
  - What Terraform wants to create, change, or delete.
- **No changes are made** to our real environment when running plan.
- Helps spot mistakes before applying.

Run the command

## terraform plan

```
mujju@VMterra:~/b11/1807$ 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.filecreation will be created
+ resource "local_file" "filecreation" {
  + content           = "This is a sample file created by Terraform."
  + 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          = "abc.txt"
  + id                = (known after apply)
}

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

## terraform apply

- This is the **action** step.
- Terraform uses our .tf files to actually:
  - Create, modify, or delete real resources (VMs, networks, etc.).
- You'll usually get a confirmation prompt before changes happen.

Run the command

## terraform apply

Type yes when prompted to confirm

```
mujju@VMterra:~/b11/1807$ 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.filecreation will be created
+ resource "local_file" "filecreation" {
  + content           = "This is a sample file created by Terraform."
  + 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          = "abc.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.filecreation: Creating ...
local_file.filecreation: Creation complete after 0s [id=cf104155daf4d1ebd2d73766dff6b362967bc489]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
mujju@VMterra:~/b11/1807$ █
```

File abc.txt is created

```
mujju@VMterra:~/b11/1807$ ls
abc.txt  main.tf  terraform.tfstate
mujju@VMterra:~/b11/1807$ █
```

To delete the resource created we use

## terraform destroy

- Deletes all resources defined in your current Terraform configuration.
- Prompts for confirmation before actually destroying anything.

```
mujju@VMterra:~/b11/1807$ terraform destroy
local_file.filecreation: Refreshing state... [id=cf104155daf4d1ebd2d73766dff6b362967bc489]

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.filecreation will be destroyed
- resource "local_file" "filecreation" {
  - content          = "This is a sample file created by Terraform." → null
  - content_base64sha256 = "jMPzASd0+57CYMunN95Nel2m6e3431tIeUK7eJ2IOXY=" → null
  - content_base64sha512 = "TweSt86/hgyRR/a8rEyRkT+SdA0JuVRif6kJV63Zy71oF2os21w8DFLIL03Um9vpBzxRsN0ZUsN9dexDTv8yDw=" → null
  - content_md5        = "1be2e220860f9945s79d192ddffdb8fda" → null
  - content_sha1       = "cf104155daf4d1ebd2d73766dff6b362967bc489" → null
  - content_sha256     = "8cc3f3012774fb9ec2616ba737de4d7a5da6e9edf8dc8b4879493b7896483976" → null
  - content_sha512     = "4f0792b41ebf860c9147f6bcac4c91913f92740389b954627fa90957add9cbbd68176a2cdb5c3c0c52c82f4dd49bdbe9073c51b0d39952c37d75ec434eff320f" → null
  - directory_permission = "0777" → null
  - file_permission     = "0777" → null
  - filename            = "abc.txt" → null
  - id                  = "cf104155daf4d1ebd2d73766dff6b362967bc489" → 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.filecreation: Destroying... [id=cf104155daf4d1ebd2d73766dff6b362967bc489]
local_file.filecreation: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
mujju@VMterra:~/b11/1807$
```