**Terraform init**

* This is the first command for initiating the terraform.
* Terraform init will download the mentioned providers and the backend configuration
* It is also safe to run multiple times
* Whenever there is a change in the provider or backend configuration it is mandatory to initialize the terraform and then execute the code.
* It also sets up local modules and remote modules
* It creates the .terraform directory to store the provider details

Terraform -init upgrade:

* If we upgraded the terraform version then we need to run -upgrade along side with init
* Change in the plugin versions
* Provider details changed

Terraform init -reconfigure:

* If there is a change in the backend configuration then this command is useful
* If we wanted to re-initialize the state then we need to use this command
* -reconfigure focuses on backend configuration adjustments.
* -migrate-state specifically handles state migration between backends.

Terraform init -migrate-state:

1. Terraform will copy the state from the old backend to the new one.

Use cases:

* When you’re transitioning to a new backend (e.g., renaming S3 buckets or adopting a new naming convention).
* It ensures your existing state remains intact while using a different backend.

terraform init -reconfigure:

Purpose: This command reconfigures Terraform based on the backend configuration in your root module.

Effect:

It ignores the existing backend configuration in .terraform and reinitializes Terraform using only the new backend settings.

Useful when you want to change backend settings (e.g., switch from local state to a remote backend) without modifying other aspects of your configuration.

Use Case:

When you need to update backend settings (e.g., changing the Azure storage account or using a different remote backend).

It won’t affect your existing state; it only reconfigures the backend connection.

Terraform init -lock=false:

* The main purpose of removing the lock in a state file, It’s because if we are working in a single project
* Disabling allows faster execution
* Always use this scenario for test purposes
* If we are using state files in local then we can make the best use of this

Terraform lock-timeout

* By default, when Terraform runs commands like terraform apply, it automatically acquires a lock on the state file. If someone else is already running apply, they will hold the lock, and you’ll have to wait.
* The -lock-timeout parameter lets you specify how long Terraform should wait for a lock to be released. For example:
  + -lock-timeout=10m will wait for 10 minutes.
  + [-lock-timeout=0s (zero seconds) causes immediate failure if the lock is already held by another process1](https://www.terraform.io/cli/commands/init)[2](https://stackoverflow.com/questions/53413639/what-is-the-mechanism-of-terraform-state-locking-when-using-google-cloud-platfor).

Terraform init -no-color

Displays without any color

**Terraform validate**

It will check the configuration to validate by parsing the code to check the syntax related errors

**Terraform Plan**

Preview the change that is going to take effect in the azure infrastructure

Generates the execution plan or the blueprint of the resources that are going to be created

**Terraform plan -out (tfplan)**

It will generate the output file which is not a readable format

To execute run terraform apply (tfplan) the machine will take it from here

Compares the current state of your infrastructure to the desired state defined in the terraform configuration file

Analyse the difference between the existing resources and the new configuration specified

**Command : terraform plan -target="azurerm\_storage\_account.st"**

If we wanted to target to a particular resource and then create it using terraform apply.

If we wanted to skip the already created resources and jump straight away to the new resource creation

When you run this in -target then the apply also needs to be passed with -target during the creation

Terraform plan -destroy

Destroy blueprint for the resources

**Terraform plan -var-file terraform.tfvars**

Supported files terraform.tfvars \*.auto.tfvars (include more than one tfvars files)

**Terraform plan -refresh-only**

Safely check the terraform state against the real world infrastructure

What is the diff between terraform plan and terraform plan with refresh-only

The state file will check and compare it with the real world infra

No proposal for any changes, focuses solely on updating the terraform state

**Terraform plan -replace**

This is a updated command for taint. Instructing the terraform to destroy and re-create the resource. Used in terraform apply only

**Terraform Apply**

-auto-apporve : Skip the prompt for the user confirmation

-backup-path: Path to backup the statefile before executing the terraform apply defaults to -state-out=”.backup extension”

**Terraform Destroy**

Terraform destroy -auto-approve

Terraform apply -destroy