Terraform Command

.gitignore file

```
*.terraform
*.pem
*.tfvars
*.tfplan
*.tfstate
*.tfstate.backup
*.lock.info
.terraform
.DS_Store
```

Download provider plugins to interact with the API

```
terraform init
```

To see all the changes before applying with terraform apply command

```
terraform plan
```

Create resources

```
terraform apply
```

To skip yes while running terraform apply command

```
terraform apply --auto-approve
```

To skip yes while running terraform destroy command

```
terraform destroy --auto-approve
```

To see all Common commands in terraform or help

```
terraform
```

Check all state Subcommands

```
terraform state
```

List resources in the state

```
terraform state list
```

List all resources in the state file or see the contents of your tfstate files

```
terraform state show
```

List a specific resource in the state

```
terraform state show <resource name>
terraform state show aws_internet_gateway.igw
```

Command for Formatting

```
terraform fmt
```

Commands used for State Management

```
terraform state list
terraform state mv aws_instance.webapp aws_instance.myec2
terraform state rm aws_instance.myec2
```

Command for Validating

```
terraform validate
```

Delete all resources created by terraform

- This will destroy all the resource this the Terraform state file
- if you want to destroy a particular resource, you can either command that resource in the state file and run Terraform apply or you can use `terraform destroy --target <resource name>` to do that.

  This command `terraform destroy --target <resource name>` will remove the resource in the state file and if your `terraform apply again`, it will create the resource again.

  The best way to destroy the resource is to remove it in `.tf` file or to command it in `.tf` file and run terraform apply command.

```
terraform destroy
```

To destroy a particular resource

```
terraform destroy --target <resource name>
terraform destroy --target aws_security_group.allow-web
```

Lifecycle with destroy

- if a lifecycle is set to `true` for any resources in terraform, it will not be deleted while running terraform destroy.

  it must be set to `false` before you can destroy it.

  This just prevents someone from deleting the resource on the console because if we comment on the resource or remove it from the configuration file and run `terraform destroy`, it will still destroy it.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_eip" "name" {
  vpc      = true
  lifecycle {
    prevent_destroy = "true"
  }
}
provider "aws" {
  region = "us-east-1"
}

/*
resource "aws_eip" "name" {
  vpc      = true
  lifecycle {
    prevent_destroy = "true"
  }
}
*/
```

OR

```
provider "aws" {
  region = "us-east-1"
}
```

Comment in terraform

We use /* and */ to comment in terrform

```
/*
This is a comment in terraform
*/

provider "aws" {
    profile = "default"
    region = "us-east-1"
}
/*
resource "aws_s3_bucket" "terraform_s3" {
    bucket = "terraform-bucket-232"
    acl = "private"
}
*/
```

Refresh Command (to refresh the current state)

When we run terraform apply command, it refreshes the state file first to match the desired state to the current state. We can also run the `terraform refresh` command to refresh the state file.

```
terraform refresh
```

The terraform workspace list command is used to list all existing workspaces.

- The command will list all existing workspaces. The current workspace is indicated using an asterisk (*) marker.

```
terraform workspace list
```

The terraform workspace select command is used to choose a different workspace to use for further operations.

```
terraform workspace select [NAME]
terraform workspace select default
Switched to workspace "default".
```

The terraform workspace new command is used to create a new workspace

```
terraform workspace new [NAME]
terraform workspace new example
Created and switched to workspace "example"!
```

The terraform workspace delete command is used to delete an existing workspace.

- -force - Delete the workspace even if its state is not empty. Defaults to false.

```
terraform workspace delete [NAME]
terraform workspace delete example
Deleted workspace "example".
```

The terraform workspace show command is used to output the current workspace.

- The command will display the current workspace.

```
terraform workspace show
```

Assign variable while running terraform apply

```
terraform plan -var="instancetype=t2.small"
```

To deploy resources in the dev environment while using the workspace.

```
terraform apply -var-file=dev.tfvars
OR
terraform apply -var-file=dev.tfvars
```

To deploy resources in the prod environment while using workspace

```
terraform apply -var-file=prod.tfvars
OR
terraform apply -var-file prod.tfvars
```

To destroy resources in the prod environment while using workspace

```
terraform destroy -var-file=prod.tfvars
OR
terraform destroy -var-file prod.tfvars
```

To destroy resources in the dev environment while using workspace

```
terraform destroy -var-file=dev.tfvars
OR
terraform destroy -var-file dev.tfvars
```

To plan resources in the dev environment while using workspace

```
terraform plan -var-file=dev.tfvars
OR
terraform plan -var-file dev.tfvars
```

Manually unlock the state for the defined configuration.

- the force-unlock command requires a unique lock ID. Terraform will output this lock ID if unlocking fails

```
 terraform force-unlock LOCK_ID [DIR]
```

Command: taint

- The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply.

```
terraform taint [options] address
terraform taint aws_security_group.allow_all
```

Command: untaint

- Manually unmark a resource as tainted

```
terraform untaint [options] name
terraform untaint aws_security_group.allow_all
```

Command: validate

- It is used to check if the configuration is valid

```
terraform validate
```

Force and destroy resources (example: s3 with content)

```
terraform destroy --force
```

Plan destroy

```
terraform plan -destroy
```

Save a plan

```
terraform plan -out example.tfplan
terraform apply example.tfplan
```

Variables

- Terraform will automatically load all files which match `terraform.tfvars` or `*.tfvars` or `.auto.tfvars` from the current directory.

  Other files can be passed explicitly using `-var-file` flag

```
terraform plan -var-file=secrets.tfvars -var-file=dmz.tfvars -
target=module.directory-service -out plan
```

To set auto-approve of apply and destroy for the current section

```
export TF_CLI_ARGS_destroy="-auto-approve"
export TF_CLI_ARGS_apply="-auto-approve"
```

To remove auto-approve of apply and destroy for the current section

```
export TF_CLI_ARGS_destroy=" "
export TF_CLI_ARGS_apply=" "
```

From each Module

```
terragrunt plan
terragrunt apply
terragrunt output
terragrunt destroy
```

From a root module

```
terragrunt plan-all
terragrunt apply-all
terragrunt output-all
terragrunt destroy-all
```