**Providers:**

Here in Terraform we have Provides, which are just like plugins used to talk to set of API’s

**Terraform Init:**

The moment we hit the **Terraform init** for deployment, what it does is, looks for all configurations details in .tf file like provider details and download required plugins to interact with AWS API.

A screenshot of a computer

Description automatically generated

**Terraform Plan:**

Very next step after Terraform init is **Terraform Plan**, with executes a dry run, and gives us a detailed view of what the .tf script gonna do, what resources will be created, any impact we have all these details it will show to us, and moreover its like sanity test, even thought it is optional its best practice to always make a habit to execute Terraform Plan.

**Terraform apply:**

This command is used finally to execute the .tf script to create resources, as per the details explained in Terraform Plan

* In order to update the existing resource:

1. Update the code in .tf and save it
2. Execute command “Terraform plan” > it shows what are the change will be done.
3. Execute command “Terraform apply” to create resource

* To remove the any part of configuration process remains same

**Note:** “+” means resource is created, “-” means resource is removed/deleted, “~”(orange colour) means configurations changed

**Terraform Destroy:**

Terraform destroy will delete all the resource created earlier, so please make sure before you delete the resource, as it first shows what resources are going to delete and asks for confirmation.

**Additional files created while executing the terraform scripts:**

“. terraform\plugins\windows\*\*” this folder and its files in below screen shot will be created the moment we execute “Terraform Init”, it gathers all the required plugins in this folder which need to be used(which are defined in .tf file).

“.tstate/.tfstate.backup” this file keeps tracking on current .tf files creations/modifications/deletions(this files helps us to create/remove/delete on go)

A screenshot of a computer

Description automatically generated

Terraform Project1:

Terraform Commands:

Terraform state list

A screen shot of a computer program

Description automatically generated

Terraform state show <resourcename>

A computer screen shot of a program

Description automatically generated

**Terraform Output:**

**Terraform output cmd to see the .tf script output**

#Prints the public server IP

output "server\_public\_ip"{

    value = aws\_eip.one.public\_ip

}

output "server\_private\_ip" {

  value = aws\_instance.web-server-instance.private\_ip

}

output "server\_id" {

  value = aws\_instance.web-server-instance.id

}

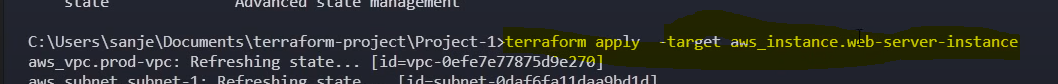
**Target Resources:**

* **It deletes only web server resource**

A screen shot of a computer

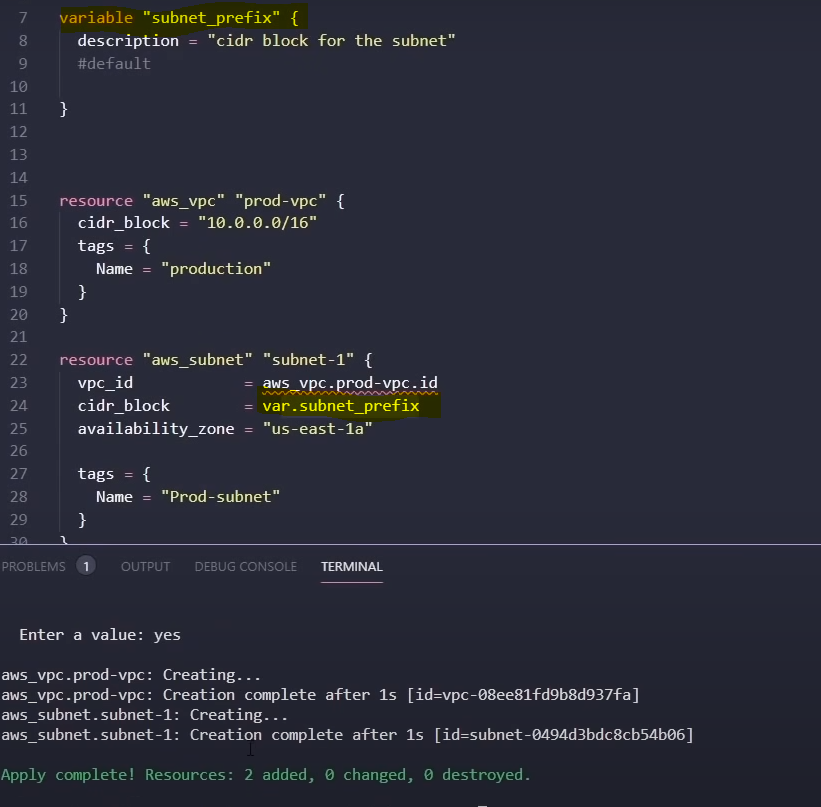
Description automatically generated

* **It recreates deleted web server resource**

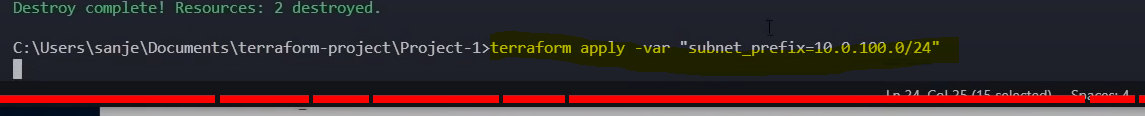


**Variables:**

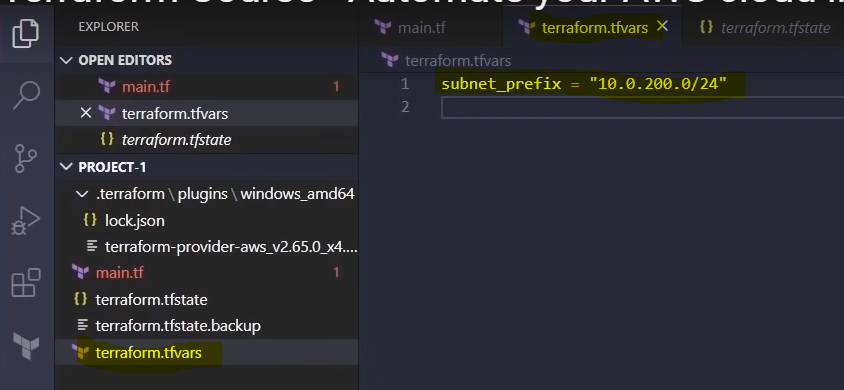
* **Defining a Variable, this prompts for use input for value, when to execute “terraform apply”**



* Assigning a variable via command line



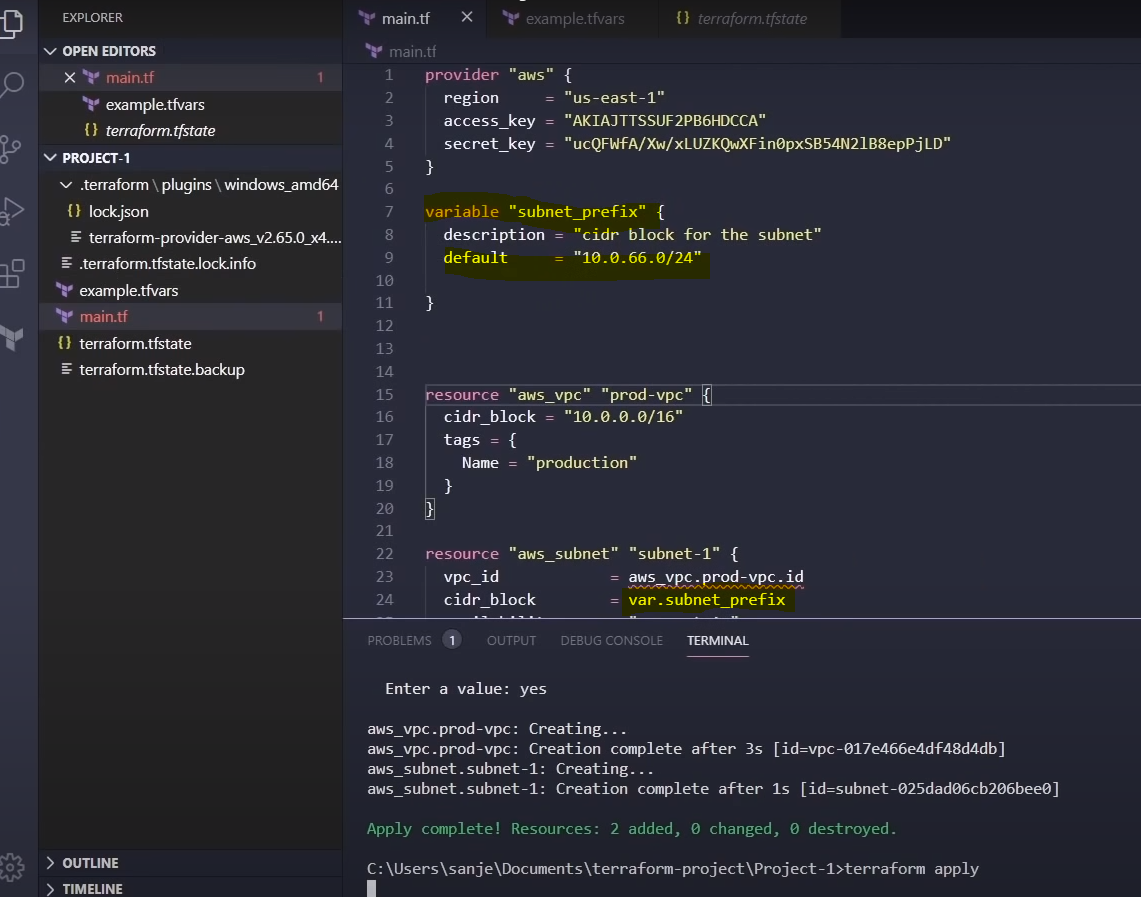
* But in real time scenarios we use separate file to define variables like below “terraform.tfvars” and then just execute “terraform apply” it picks the value from .tfvars file.



* **Other way you can also pass explicitly the variable file name as shown below**



* **Other way around is you can pass default value directly in main.tf file**



* **One more thing we need to observe is in “.tf” file we define a variable and on .tfvars file we assign a value.**