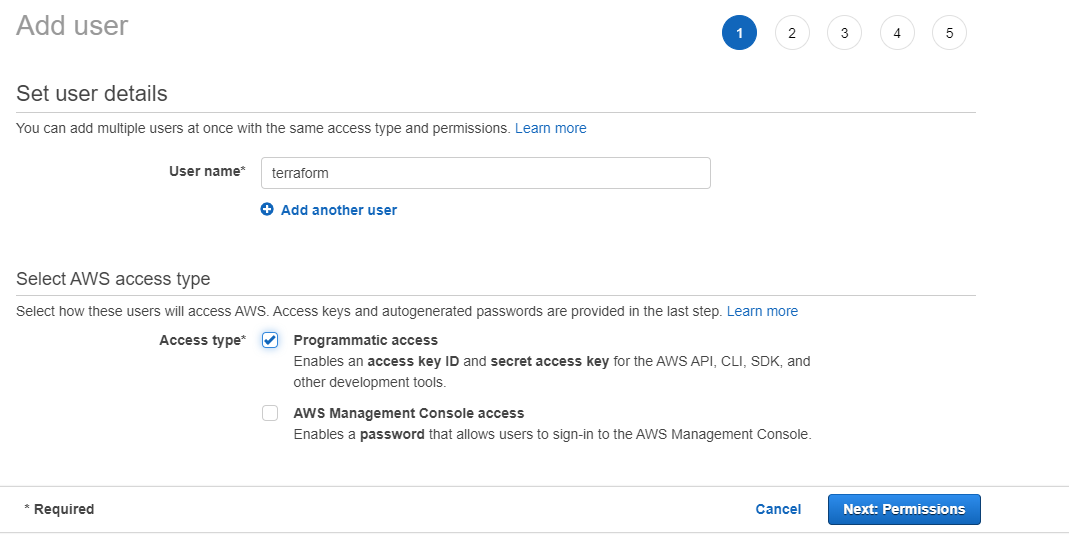
Download Terraform from official site

<https://www.terraform.io/downloads.html>

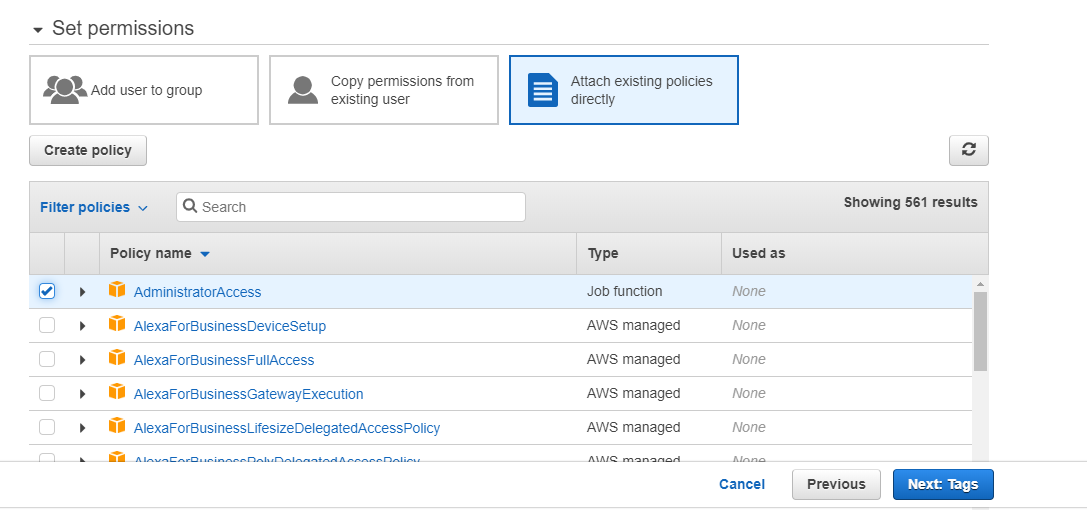
Set Up patch variable for this as per doc and check your tf version



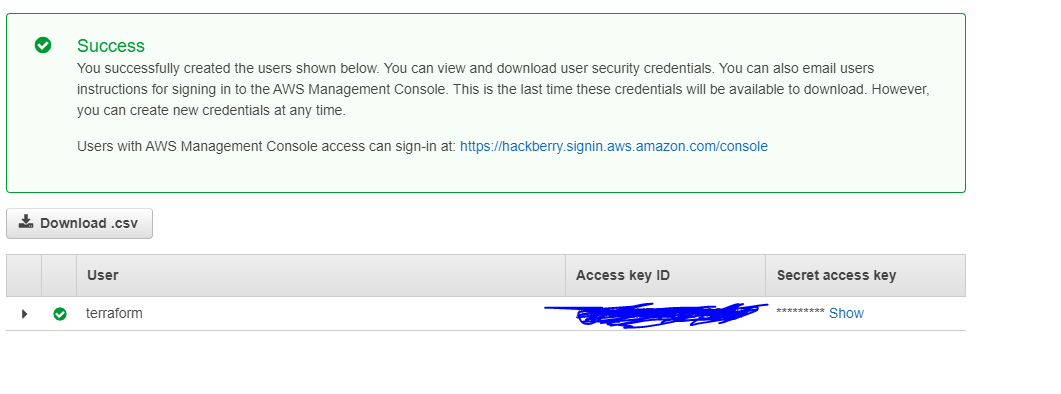
Create a terraform user to integrate AWS with terraform. I have created a user terraform.



Provide Administrator access policy to this user

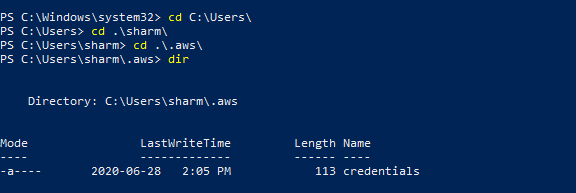


Download Access key id. Create a credential in order to authenticate with AWS by default.

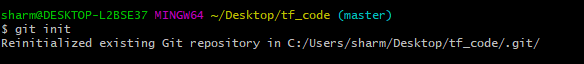


Create a .aws folder inside the user folder and save your cred.(You can choose any other folder)

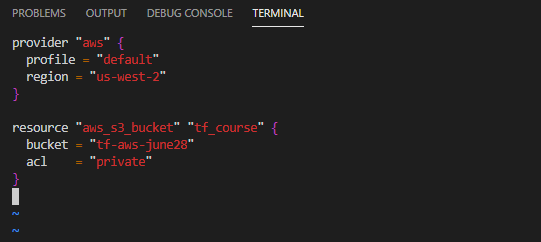
Save your cred inside the c\users\username\.aws



I’ll use Git here to integrate terra with AWS so ensure you have Git installed to follow along. You can also use other command line as well to work with.Create a folder & Initialize your git repos



Create a file and add the below tf code to create a bucket in aws.



provider "aws" { # Terraform is used to create, manage, and update infrastructure resources such as physical machines, VMs, network switches, containers, and more. Almost any infrastructure type can be represented as a resource in Terraform.

A provider is responsible for understanding API interactions and exposing resources. Providers generally are an IaaS (e.g. Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g. Heroku), or SaaS services (e.g. Terraform Cloud, DNSimple, Cloudflare).

profile = "default" # this is your default profile to authenticate to aws using your saved cred

region = "us-west-2"

}

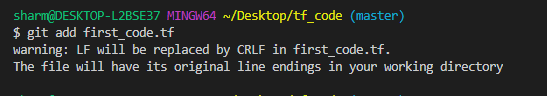
resource "aws\_s3\_bucket" "tf\_course" { # what resources you are using need to be place here. in this ex s3. resource name is tf\_course this is how terraform define

bucket = "tf-aws-june28" # this is consider as arguments in terraform and we use this in "". this is how aws define this name

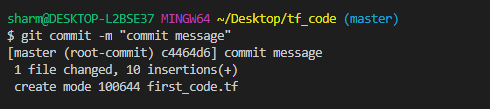
acl = "private" # an argument to define it is a private bucket

}

Use git add to add this file in local git repo

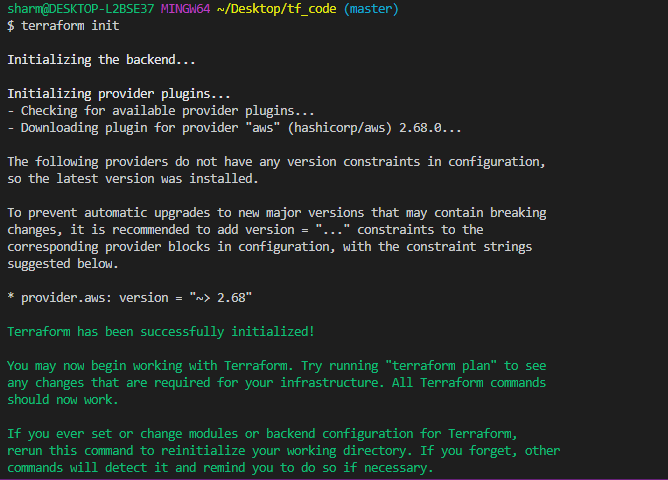


Commit your repo using git commit and it it ask for id provide your github account details

\

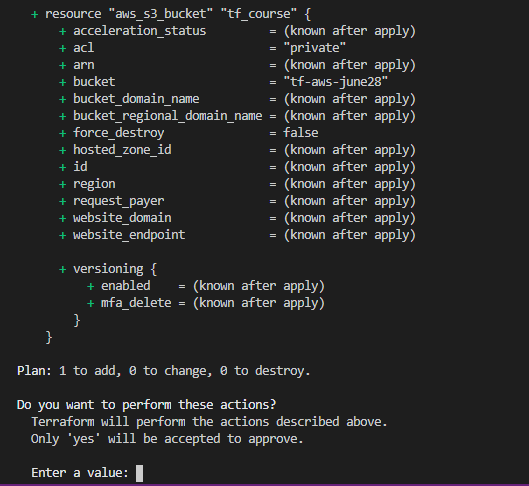
Initialize your tf directory.

Once initialize we are ready to deploy our code to aws

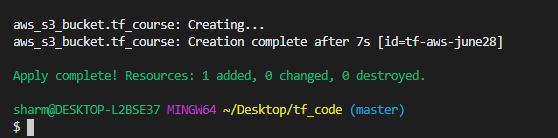


Now use terraform apply. This will ask you to complete the changes on AWS, put yes, if you don’t want this to ask you yes or no, you can simply write in the beginning as

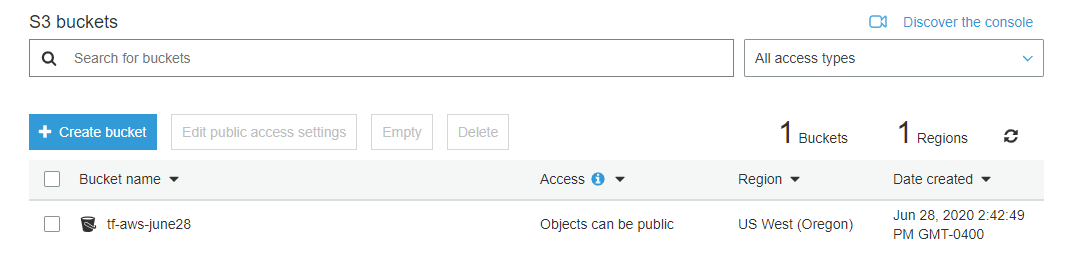
**Terraform apply -auto-approve**

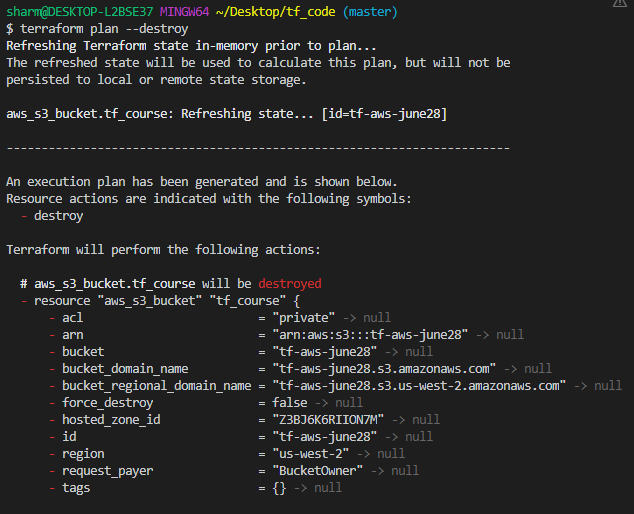


As soon as you hit yes within a sec this will deploy a bucket in your aws console

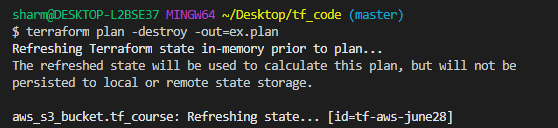


Login and check in your console

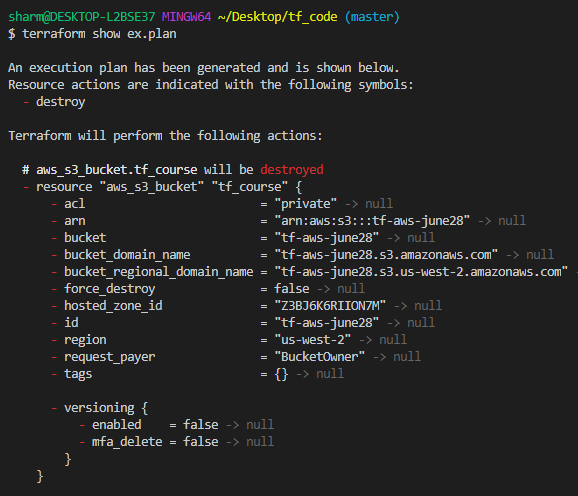




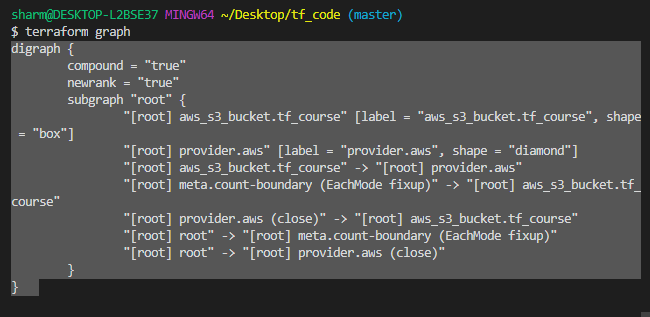
You can also save this plan to a file to see what exactly it is going to do

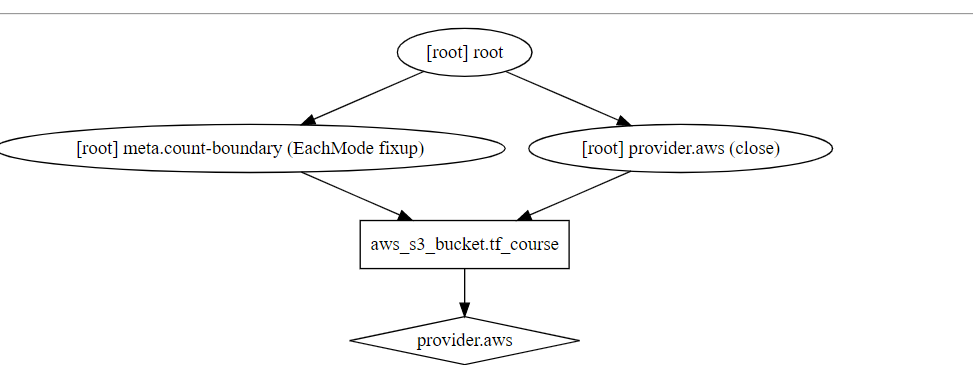


Use terrafor show command to check the result



**Terraform graph** - so i have one S3 bucket in my account and lets see what we get when we generate the graph for it



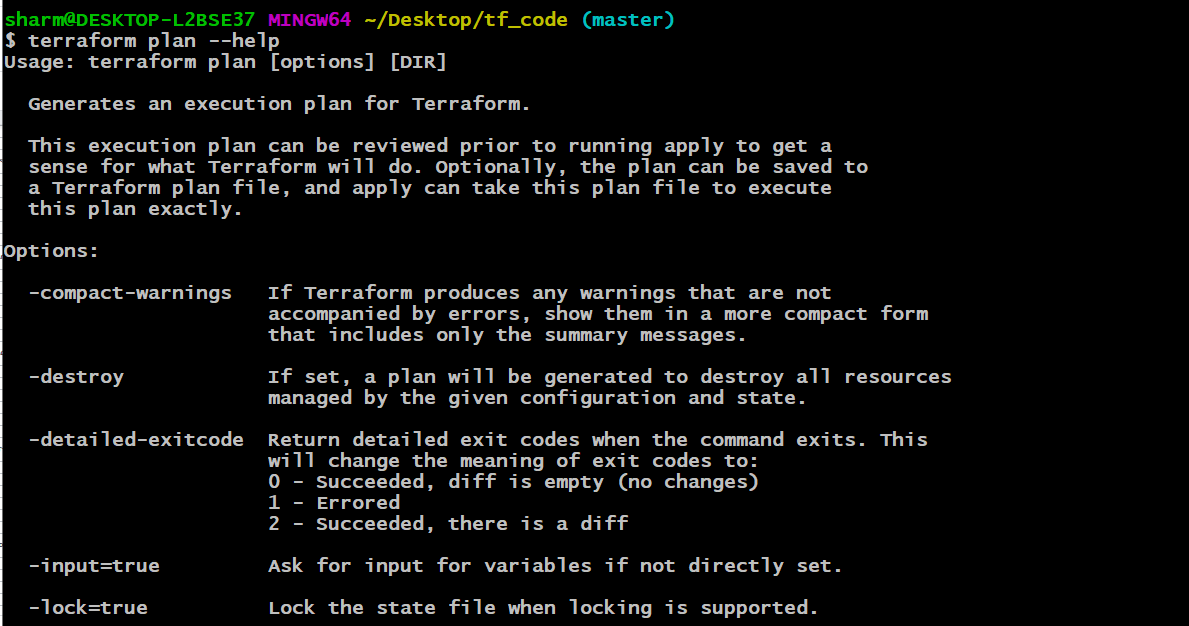


**How does TF work ?**

IaC - This is an Infrastructure as Code. We can define resources based on tf code. You can build your infrastructure using a code, share it with others. You need to plan your execution well.

Code> State> Plan > Apply = Always remember this.

Terraform plan --help = this will give you what options we can use with plan



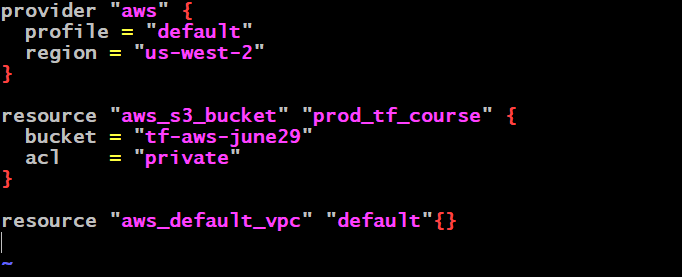
**Resources** - Building blocks of Terraform code

Define the “what” of your infrastructure

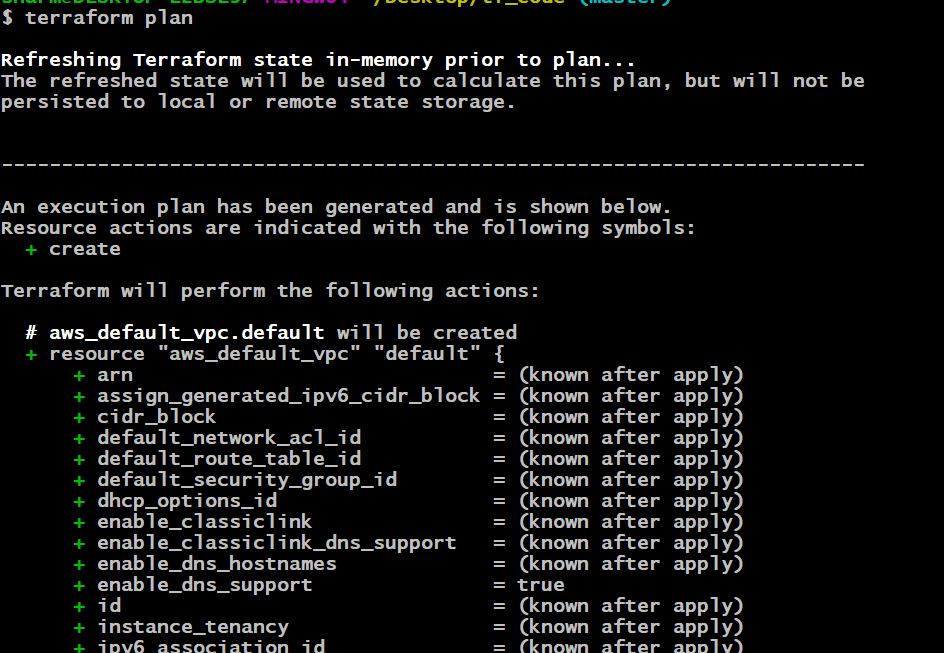
Different settings for every provider like (AWS,Azure,GCP etc..)

As we have destroyed our previous bucks env, Now lets add VPC in our tf code

Create a new prod .tf file and add below

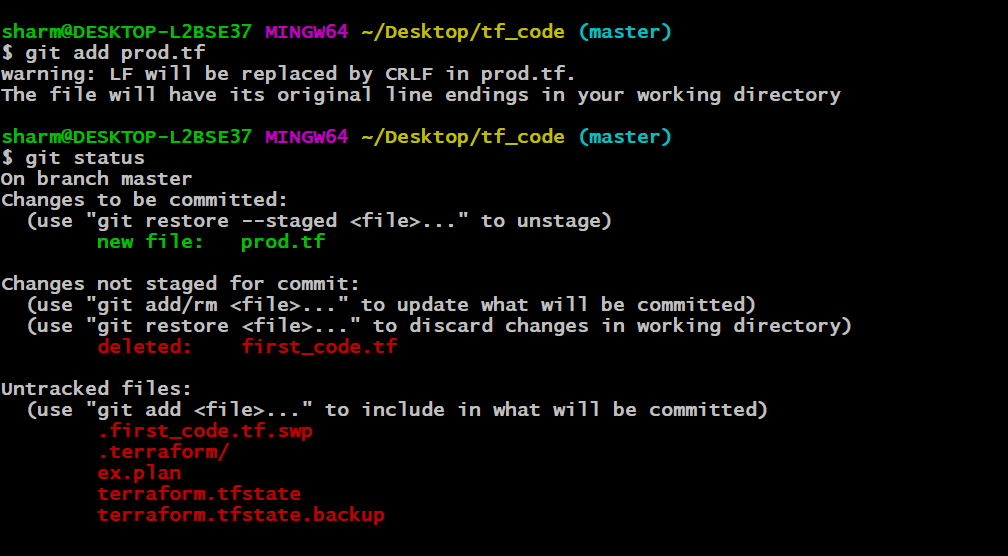


Use terraform plan to see what it is going to do. As you can see using plan we can see this is going to create a default vpc



Add this file using **git add prod.tf** command

And use **git status** to trak the file. As you can see git has found the previous .tf file has been deleted so we need to use git add firs\_code.tf to add this file to repo



Now you can use git commit command to add the vpc changes to it.

