Session 01 – Installing terraform

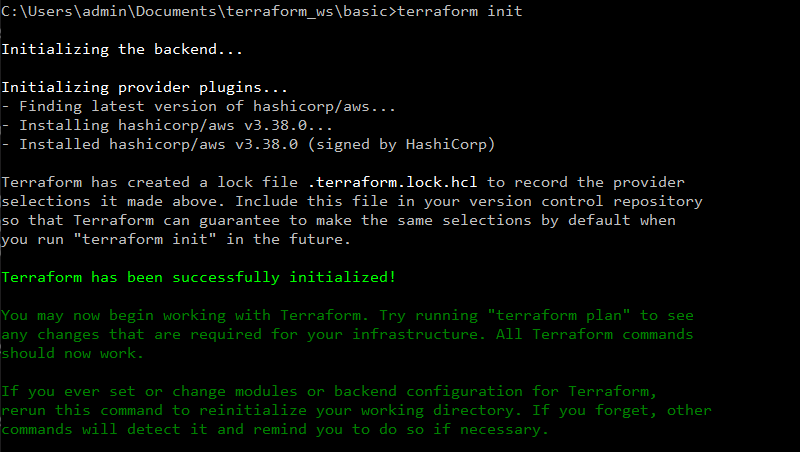
* If you are using only one cloud like AWS, so you can use cloud formation.
* But when you want to use multiple cloud in your organization then we have to use tool like terraform.
  + Resource = which cloud
* Terraform is a Infrastructure as a code tool.
  + It will setup a complete infrastructure.
* All the clouds have their own services.
  + Some cloud have some service cheaper, while other cloud has other service cheaper.
  + We also have to check about security and some other things.
  + So we use different different cloud all at once.
  + Here we can use terraform.
* Using terraform you can easily switch between multiple clouds.
* You can download terraform from, <https://www.terraform.io/downloads.html>

Session 02 –

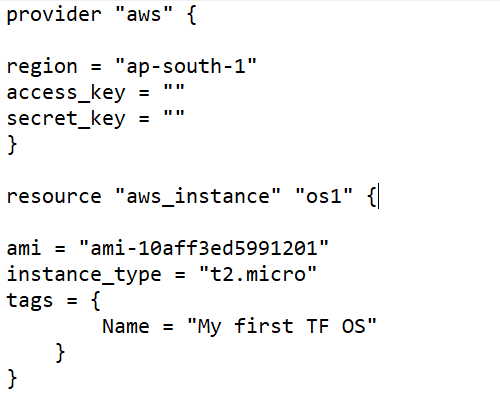
* If you want to get some service from providers (cloud), then you have to download a plugin.
* You have to tell your provider and save file with tf extension.



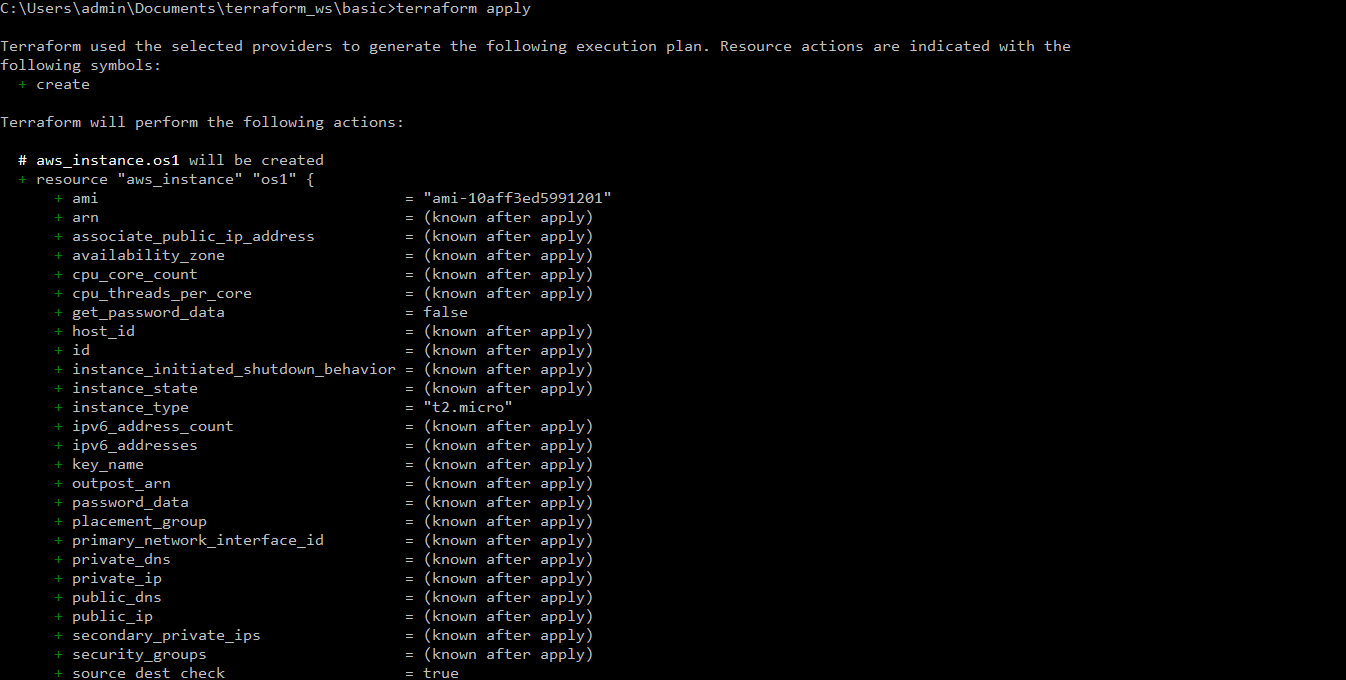
* + You have to initialize this file.
  + It will download a plugin if not available.
    - For this you have to use **terraform init** command.
    - This command you have to run only once per directory.
  + In your current folder it will search for .tf extension.



* You have to give resource name (tag) with every resource.



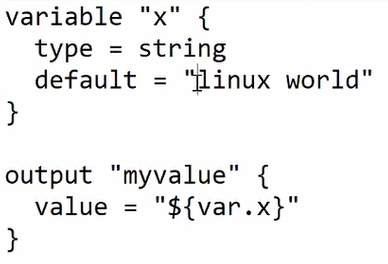
* You can check your code and plan using **terraform plan** command.
* You can apply this using **terraform apply**.
  + Both commands give you same (plan) output, but apply will ask for your confirmation to run resource.



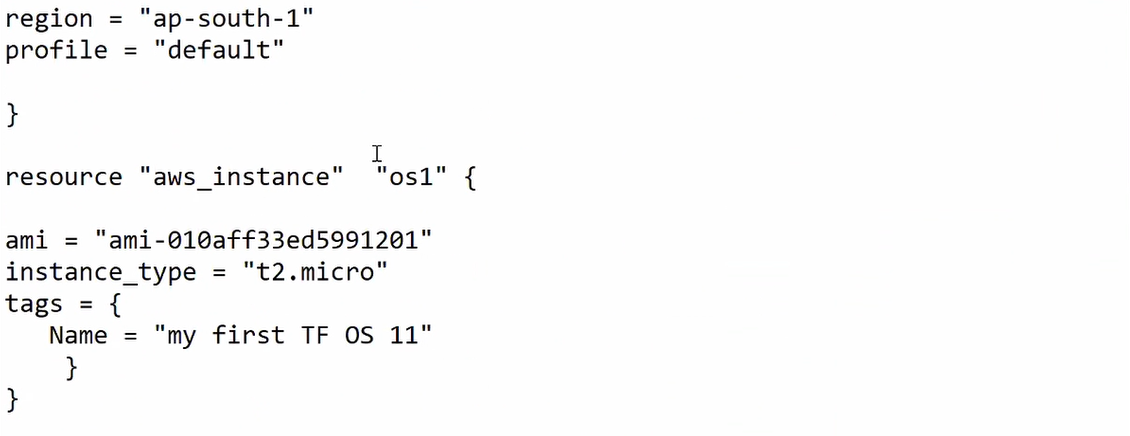
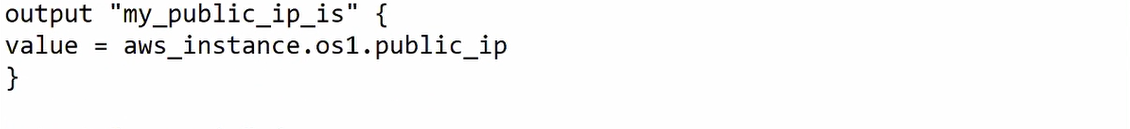
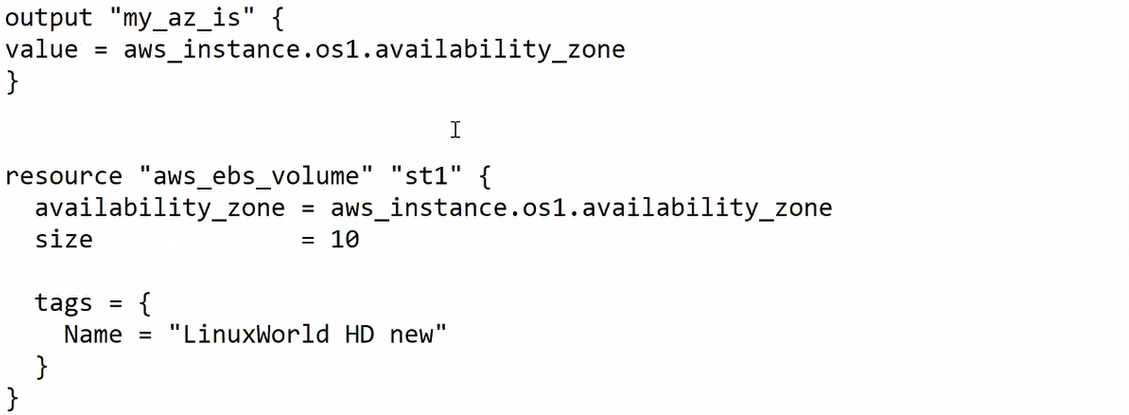
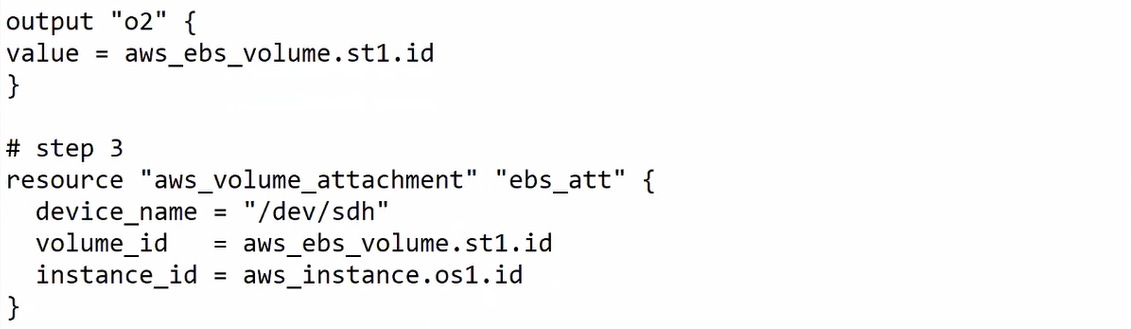
* Terraform is **declarative language**.
  + There are lots of benefits of declarative language.
  + If resource is already available it will not touch that resource and go to next resource.
    - This nature is known as **idempotence nature**, if thing is there I will not apply.
* Here you can also change your instance type just by changing one word.

Session – 3

* If we have multiple file with extension tf, then terraform will run all the files.
* Terraform user define variable syntax.

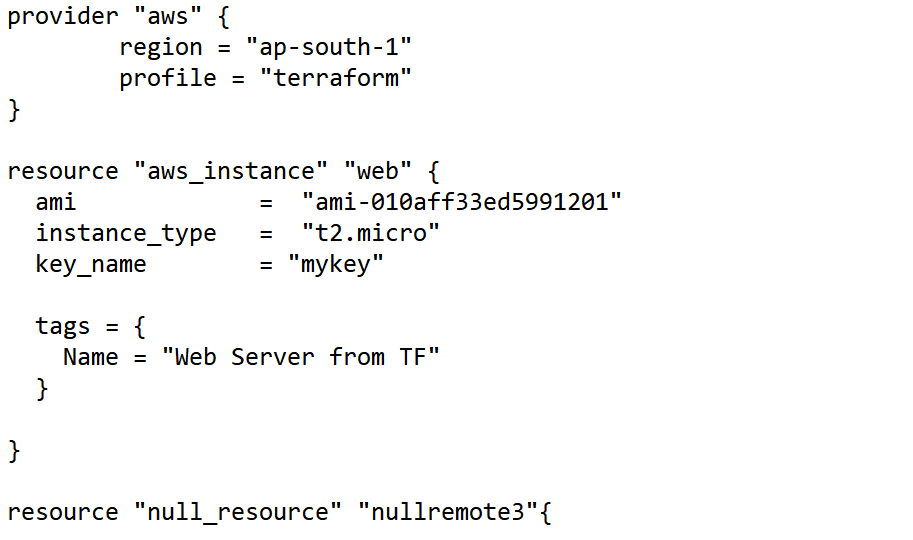
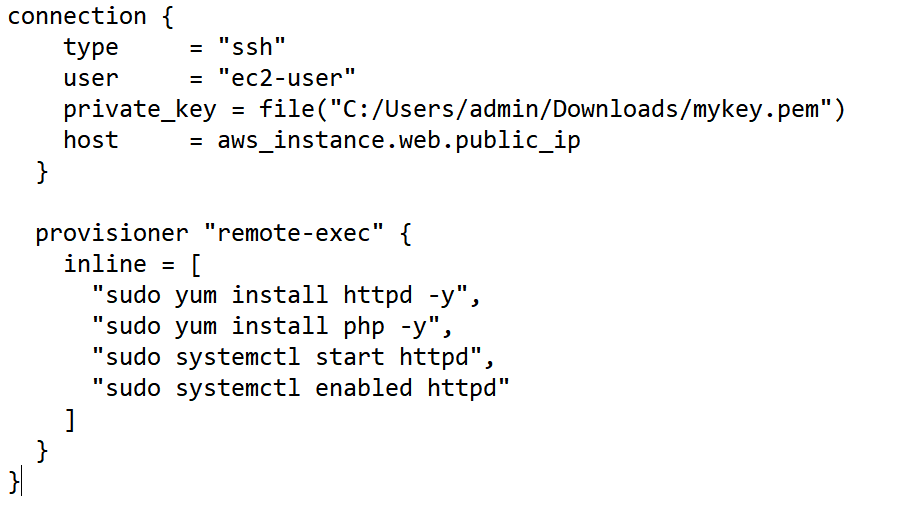
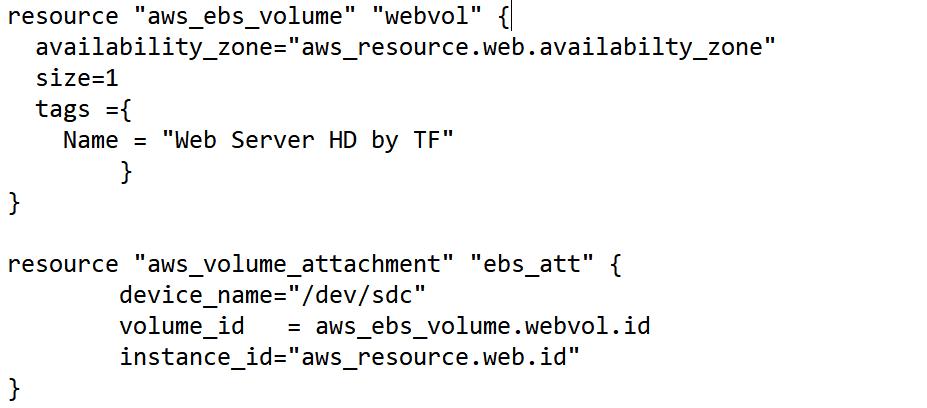


* There are many variables which are system created.
* In last class we have provided secret key access key in the same file, but instead of this we can create a profile using aws configure create profile and use this profile in our code.
* When you launch a instance from terraform, they have all the details stored in variable.
  + For attach a HD both HD should be in same availability zone.
  + Here we can use terraform internal variables.

* If you use **terraform destroy** it will destroy complete project created by this in a proper sequence.
  + In this project we have to first destroy attachment, then ebs, then instance.

Session 04 –

* Terraform will help you to automate the facility provided by the platform.
  + You can write a code to automate things (platform) (provision).
* In this practical we will launch ec2, install apache webserver & store data in external ebs volume.
  + For installing apache webserver we have run some command remotely, for this we can use remote-execution, for connectivity you have to use connection block.
* **Security**\_**groups** is a list so you have write inside square brackets.
* There is mainly two different provisioner for running a program in local or remote.
  + **Local**
  + **Remote-exec**
* For ssh connection we have to use **connection** provisioner.
  + It is a **nested keyword**, so it should be part of any resource (any block).
* Here for connecting to ec2 instance we have to use key stored in somewhere.
  + So we have to retrieve it from some folder.
  + For this you have to use **file** function.
    - If you don’t use anything terraform will feel your path is your secret key.
  + Pic
* Here instead of remote-exec we can use local-exec and write playbook path here.
  + For this we also don’t require connection block because ansible playbook will do this all thing for you.
* Here you have to note one thing that in path you have to do backward slash because they treat / as an **escape sequence**.
* Terraform is for provision they are not meant for configuration management.
  + They can provide you CM but they are not specialized for it.
  + Here we have already launched instance from terraform and then we have add connection and remote-exec block.
* So we can do one thing we can destroy the whole instance and re run whole code.
  + But it is not a good idea, so instead of what we can do is, we can write separate resource for this thing.
  + For this type of use we have special resource called **null\_resource**.
    - You have to note one thing that for running this null\_resource we have to init again, because there is a separate plugin for null\_resource.
* You can pass -auto-approve it will auto run your code, you don’t have to manually write yes every time.   
    
    
    
  

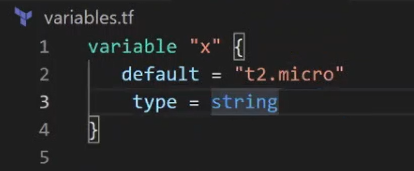
Session 05 –

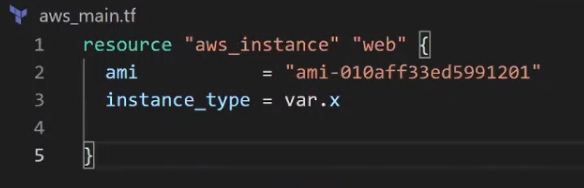
* Instead of creating single, big file we can break this into multiple files.
* Terraform automatically maintain the ordering, like if you want to launch instance ebs and attach it to instance.
  + Then it will first launch both then attach, this thing is known as **inference resources.**
  + But there is some exception where terraform will not run in infer way, so make sure you have organized your data in alphabetical order.
  + So for this you can do one more thing, you can use **depends\_on** keyword.
* When you run apply command, terraform will go to your provider and store his complete details in file called .**tfstate**.
  + This file is match with your code (desired state).
* You can use **terraform validate** to check syntax is proper or not.

Session 06 – Terraform on GCP

Session 07 –

* Instead of writing static (fixed) code, we can pass variable.
  + Eg, we can create variable for instance type.

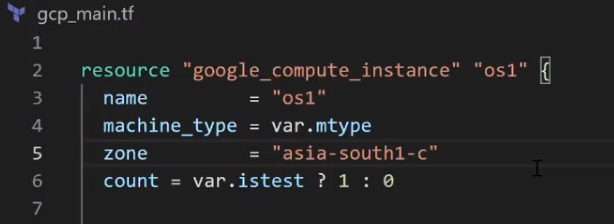




* + Now they use tc2.micro as a default type.
  + But on the fly we can pass different instance type.
  + For this you have ti use this syntax.
    - **Terraform apply --var=”x=t2.medium”**
  + In real world we have hundreds of variable.
    - It is harder to write every variable value here.
    - So instead of this we can use configuration file of terraform.
    - For this we have to create **terraform.tfvars** file in same folder with same name.
    - Here you can simply write all variables with default value.
      * You also have to include this var in variables.tf file.
  + If you have changed file name then we have to use, **terraform apply -var-file=<file>.tfvars**
  + Now IT guys can change config file or on the fly on command line they pass values.
* For Boolean data type we have to use bool as keyword.
* We can use Boolean as a if else.



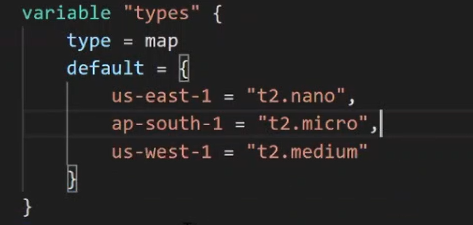
* If you want to run any resource block multiple times, then we can pass keyword called count.
  + You can use **count** keyword in any **resource block**.



* + So here if you pass var value true then it will launch instance in google cloud (testing env).
    - And if you pass false then it will launch instance in aws cloud (production env)
  + Terraform apply -var=”istest=true”
* Suppose you want to choose multiple values for something, then you can use list in variable file.

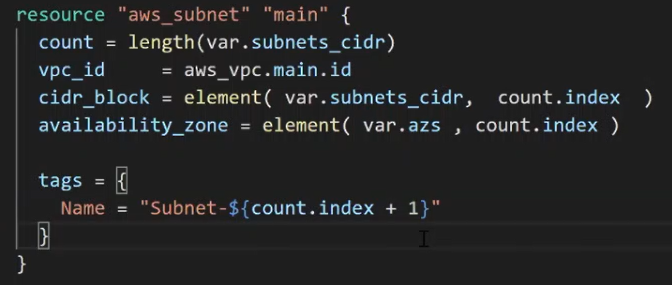
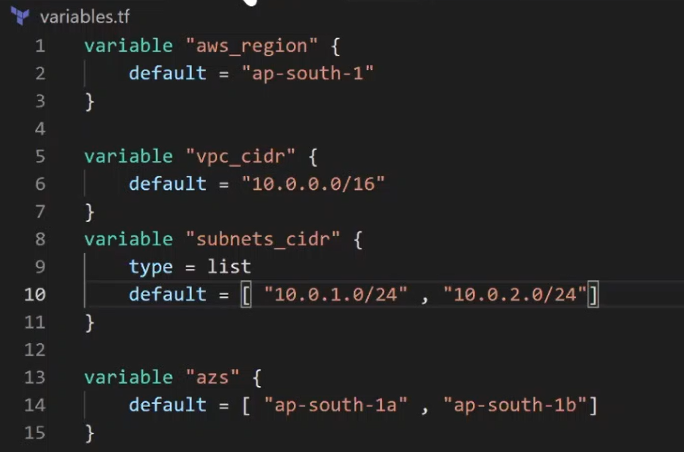


* + And while using this thing you can pass **var.azaws[0]**
* We also have map datatype in terraform.



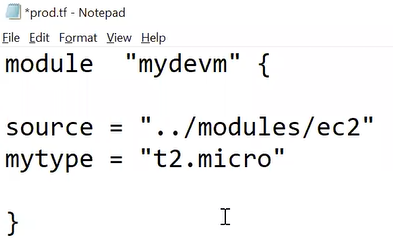


Session 08 – public-private VPC setup

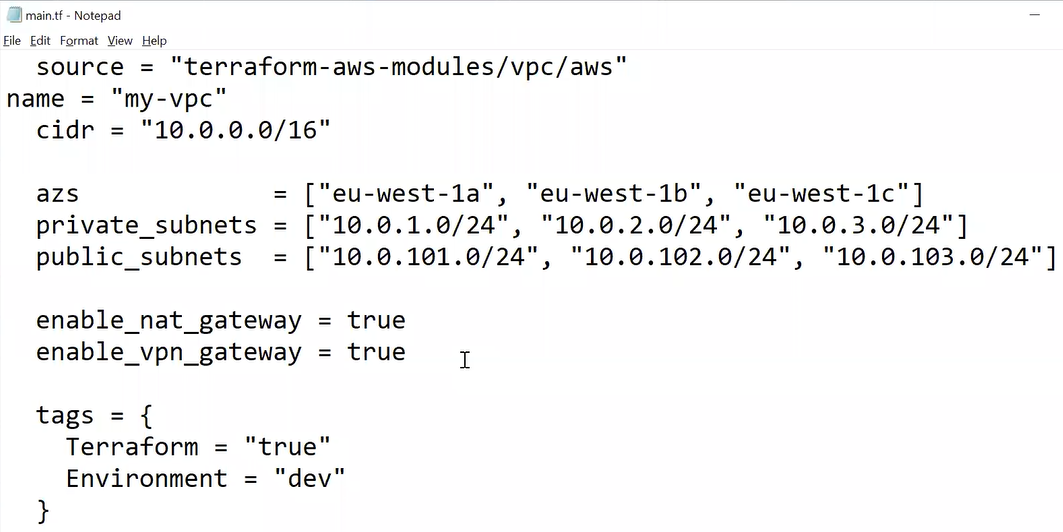
* Built – in function
  + 
  + Here length is built in function.
  + It will go to variable (list) and find length of variable.
* 

Session 09 – module

* We have different setup for prod env and testing env.
* But main.tf and var.tf this both file is same for both env.
  + So we can push this file in git/scm/nas storage and when we want to use this directly pull this file.
  + Both belongs to different env but both are using same standardized folder, same code-base.
  + This centralized folder is known as **module**.
* Normally we give folder name module (can be changed) and keep all modules there.
  + Now create a normal code.
  + Like for ec2, create var.tf & in main.tf add ec2- launch code.
  + Now when any team want to run same code, it will use same module.
  + Here we are using local module, if you have to use module online then you can pass URL in src.



* First time we have to initialize this module before applying it.
  + As seen in above pic we can directly pass variable here.
* This way we can use module from terraform registry.



* You can also use variable from other modules.



Session 10 –

* You can write **terraform console** it will give you live terraform prompt.