**TERRAFORM**

* Terraform is a IAAC tool that allows you to build, change and version infrastructure safely and efficiently.
* Download the terraform exe file and we need to execute it from the download path only then the terraform commands will work if not it wont work, in order to make use of it from anywhere with in cmd prompt we need to set the environmental variables (we need to give the path of the located exe)
* We need to navigate to the code file directory and we need to give **terraform init** so that it downloads the required providers plugins for it.
* Terraform plan : its a command which is used to read the tf files and will show us the required changes which are going to happen on it.
* Terraform apply : which is used to apply the changes with in the main.tf file and will starts deploying
* Terraform destroy : which is used to delete all the resource which has been created it will fetch the data from the state file and will delete accordingly.
* Terraform validate : it just validates the syntax of the HCL code
* Components : providers , resources , variables , statefiles , provisioners, backends, modules, data sources , service principals.
* Providers : it relies on plugins called providers to interact with the cloud providers & SAAS & API
* Resources : each resource block defines one or more infrastructure objects like VM and vnets
* Variables : it used to create the variable.tf file needs to be created in the working directory to store all variables for the used main.tf config file.
* Statefile : after the deployment is finessed terraform creates the statefile to keep the track of current state of the infrastructure.  
  Primary aim is to compare the file from the current state vs desired state. Using this file
* Provisioners : it helps us to run the additional steps or task when a resource is created or destroyed.

Variables

* The primary aim of the variable is to make the code dynamic so no need to pass the contect within the main.tf file we can create a variable.tf file and can use the syntax as   
  **Variable “rgname” {  
  Type = string  
  Description = “used to create the resource group ”  
  Default = “somevalue”  
  }**
* The new content can be passed through the **terraform.tfvars** file where we have make modifications directly within the file.
* Inorder to get the data from the variable and the terraform.tfvar file to that of the main file we can use the notation within the main.tf file **name = “${var.rgname}” location =”${var.rglocation}”**
* If we accidentally delete the state file in the terraform how do we fix it
* Initially we need to open the main.tf file and then we need to note all the resource group names and followed by the resource id which is present with in the console  
  **Terraform import azure\_rg\_create.rg1 /subscriptions/029dqbd092988bcasc998772/**  
  Then we need to execute each commands one by one and then the terraform state file will be created and all the resources will be imported with in the statefile.(we need to repeat the steps)  
    
   Terraform Backend
* Inorder to achieve that backend if the state file is deleted we can store that state file within the storage account   
  Login to azure console >> create a storage account >> we can choose the code for the authentication method types as the ad authentication or the Access keys …etc
* Copy the terraform command for any of the authentication type and then modify the accesskey within the azure consone   
  Terraform init (which would initialize the backend command)- then it will upload the state file to the blob which we created within the console.  
  When we do any modifications with in the main.tf file and then apply or plan thus it authenticates within the statefile present in the cloud and will change the **lease status to locked** once appy or plan is done it will release the **leased state** to unlock (which wont accept the parallel executions)  
    
   MODULES
* If you want to follow the template based deployment we can follow modularized approach.
* We can add the module file in place of variable.tfvar which we can simple use the module syntax followed by the source =”./modules”
* Terraform destroy –target=module.modulename (which is used to delete that particular module)   
    
   DATA SOURCEs
* We can uses these data sources where if we can create a resource group using the other language apart from terraform. these documentation is available in terraform documentation only.
* Datasources in terraforms are used to get info about resource external to terraform and use them to setup your terraform resource.
* If we use the terraform destroy it only delete the resources which was created by only terraform
* Locals are those inorder to use it simple way of notation initially we need to declare the at the beginning of the code we need to use it throughout the code. **local.rg\_info.name**  
  **Local {  
  Rg\_info = azurerm\_resource\_group.rg1**

**}**

* Locals can be referred or called with the same file or with in the code only.
* Terraform workspace list : which is used to show the workspace list
* Terraform workspace new name : to create a new workspace command and switch as well
* Terraform workspace select name : in order to switch between the workspaces
* Terraform apply –var-file dev.tfvars : to deploy only in the dev environment in the currentworkspace