**TERRAFORM**

https://github.com/zealvora/terraform-beginner-to-advanced-resource

Tools used to deploy infrastructure as a code are Terraform, Cloudformation, Ansible, Heat, Saltstack, Chef, Puppet etc. Here, Ansible, Chef and Puppet are primarily used to manage software on the existing servers.

**Pros**

* Open source
* Terraform can integrate with the configuration management tools such as Ansible to configure the services.
* Terraform supports multiple providers (cloud providers and others too) such as AWS, Azure, Akamai, Alibaba, Google cloud platform, Helm, Kubernetes, grafana etc.
* Easily extensible with the plugins

**Installation**

* Terraform - downloads - windows - download the file and extract the .exe file - move the .exe file to c drive.
* To run the terraform globally (not restricted to c drive) - system properties - envirnment variales - path - edit - browze - select the terraform.exe file
* For a linux machine, select run the commands specifiles

**Setup and installation in visual studio**

* Extensions are the add-ons used to allow you the required customizations and enhance the experience. (Without the terraform extension installed, the entered contents will be monochromatic and with the extensions it will be colorful)

**Authentication and authorization**

* It is required for the terraform to access aws (or any other providers) and create the required resources.
* Access credentials for different providers are,

i. AWS - Access and secret access key

ii. GitHub - Tokens

iii. Kubernetes - Kubeconfig file, credential config

iv. Digital Ocean - Tokens

**Providers**

- Terraform supports multiple providers such as aws, azure, gcp, k8s etc

- Provider plugin is used to manage an external api

**Terraform workflow**

1. **Terraform init :**

* It downloads the provider plugin specified in the provider block and stores it in the .terraform directory locally.
* The .terraform folder contains the modules and provide folders.
* .terraform.lock.hcl will be created which is maintained automatically by "terraform init".

1. **Terraform plan:**

* It is used for the dry run and get the details of the resources that are being alterd.

1. **Terraform apply**
2. **Terraform destroy**

**a. Virtual machine through terraform**

* It is the creation of the ec2 instance from terraform.
* All the configurations are specified in the resource block.
* Resource block contains,

i. ami --> which can be obtained from the aws console. It differs for each region.

ii.instance type --> it is t2.micro or any other

**Terraform state file management**

|  |  |  |
| --- | --- | --- |
| **Feature** | **terraform.tfstate** | **terraform.tfstate.backup** |
| Purpose | Current state file | Backup of the state file |
| When is it created? | Created automatically when you run terraform apply for the first time | Created automatically before Terraform makes any changes to the state file |
| When is it updated? | Updated every time you run terraform apply | Overwritten every time Terraform makes changes to the state file |
| Where is it stored? | Stored in the same directory as your Terraform configuration files | Stored in the same directory as the state file |

**What?**

* Terraform state file is created by terraform when terraform apply is executed.
* It contains all the resource information in the form of the meta-data.
* It is a recorder file which records all the steps that are executed once the file is created.

**Need**

* Can pull the state file from the remote and create the infrastructure even if the state file is not stored in the local.
* A single state file is to be maintained even when multiple developers are working. Hence it should be stored remotely.

**When?**

* It is created in the local workspace. Hence called as a local file.
* Once the plan or apply command is executed, the terraform state file is created first the the resources are created in the aws/azure.

**Store the state file remotely**

To store the state file in the s3 bucket, we need to add a terraform block in the main.tf file where the s3 bucket details will be mentioned.

terraform {

backend "s3" {

bucket = "mybucket"

key = "path/to/my/key/terraform.tfstate"

region = "us-east-1"

}

}

* Bucket is the bucket name and key specifies the path to the directory to store the state file.
* Bucket is created and terraform apply is executed.
* Once the remote storage is applied, the local state file will not be updated.

1. **Terraform state pull:**

* It is used to get the updated state file from the remote to the local (i.e, s3 bucket).

1. **Terraform state push:**

* It is used to push the updated state file from local to the remote (i.e, s3 bucket).
* But not a best practice as it will alter the changes that are stored by other dedvelopers.

**Terraform state locking**