**Terraform**

* Terraform is a third party open source infrastructure as a code(**IaaS**) platform from HashiCorp that allows you to build, change and version the infrastructure safely and efficiently in fraction of seconds.
* The language that we use to write infra as code in terraform is **HCL**(HashiCorpLanguage) and the file extension is .tf
* Terraform supports multiple cloud providers.

**Install Terraform on Linux:**

**sudo yum-config-manager --add-repo** [**https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**](https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo)

**sudo yum -y install terraform**

**Install Terraform on Windows:**

[**https://releases.hashicorp.com/terraform/1.9.3/terraform\_1.9.3\_windows\_amd64.zip**](https://releases.hashicorp.com/terraform/1.9.3/terraform_1.9.3_windows_amd64.zip)

unzip, add terraform exe file path into environment variables of PATH, to access terraform from anywhere.

**Install AWS CLI on Windows to connect with Terraform:**

**https://awscli.amazonaws.com/AWSCLIV2.msi**

check the aws installed or not ***aws --version***

**link AWS CLI & Terraform by configuring a user:**

1. Create a IAM user in AWS, provide Administrator policy, and get the access & secret key of the user.
2. ***aws configure:*** enter access and secret key.

**Terraform Components:**

1. **Providers:**

To interact with multiple cloud providers, SaaS and other APIs, we use providers in Terraform.

1. **Resources:**

Resources are the most important elements in the terraform. Each resource describes one or more infrastructure objects such as Virtual Machines, Security groups, DNS records, buckets, SaaS component.

1. **Variables:**

Variables used to avoid hardcode values, the name itself say varies the values. A variable in terraform makes our configurations more dynamic during the deployments. A separate file with the name **variables.tf** needs to created in working area to use all the variables in our **main.tf** file.

1. **Statefile:**

After each deployment completes, terraform creates a state file called “**terraform.tfstate**” to keep track of current state of infrastructure. It compares the current state and desired state before deploying/destroying the resources. Whatever the changes Terraform detected by the comparison it creates/updates/destroys those only as remaining are already present and duplicating them again leads to errors/issues.

1. **Provisioners:**

Provisioners provide the ability to perform additional steps/tasks during a resource creation/destroy. Same like Ansible but not a replacement for configuration management tools.

**Terraform Commands:**

1. ***terraform init***: it initializes/downloads the provider that you mentioned in tf file into your local system by creating a “.terraform” directory.
2. ***terraform validate***: it validates the compilation.
3. ***terraform plan***: it shows the preview of infra what is going to be created after the comparison.
4. ***terraform apply***: it creates/updates the infra that gets from the comparison and generates a statefile to store the status current infra got created**.**
5. ***terraform destroy***: it destroys/deletes the entire infra whatever is there in the statefile.