****

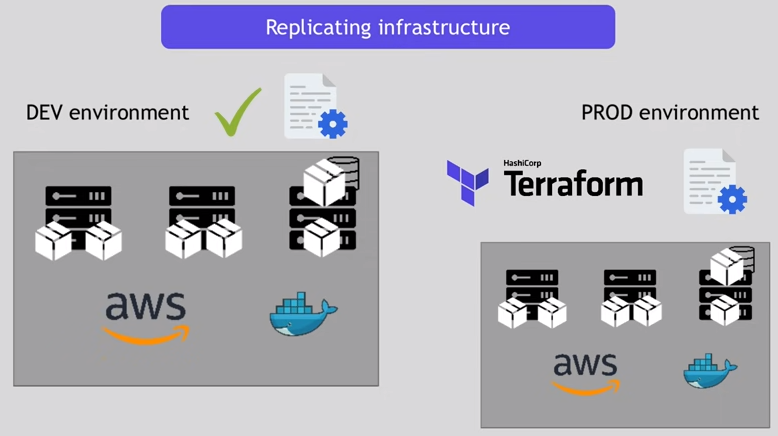
**Terraform** is HashiCorp's infrastructure as code(IaC) tool. It lets you define resources and infrastructure in human-readable, declarative configuration files, and manages your infrastructure's lifecycle.

Why **IaC** model?To Automate the process of building infrastructure or update or remove.

Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface. IaC allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version, reuse, and share.

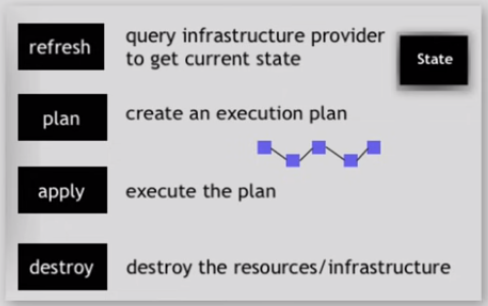
Why Terraform?

* Free/open source
* Hashicorp developer
* Script: HCL easy to write and understand.
* Support multi provider (Aws, Azure, GCP, Oracle, Docker, Alibaba...)
* Support any platform (Bare Metal, Virtual, Cloud)
* State-full configuration
* Helps replicating infrastructure



How terraform works?

* Terraform needs two inputs **TF config** and **TF state** given to Central terraform server(Core) which will communicate to several **providers** via plugins when initialize terraform.
* Next it will validate and do dry run (simulate mode) before applying to real time.
* Then finally apply/execute config (.***tf***)
* After creation it will create ***terraform.tfstate*** file which hold all created resource information like instance ID, IP address, key-pair, etc. Always it checks current state (tfstate) with desired state (config file)
* If it is stale you do destroy.



How to setup terraform on AWS linux ?

* Download terraform repo to linux
* Add terraform repo to local linux
* yum install terraform -y-----
* Create iam user to allow terraform as administrator access?
* go to iam -- create user-- programmatic --set permission (administrator access) copy access key and secret key..

Process for every script execution:

1. create directory (project2)
2. create config file (.tf)
3. Example: vi main.tf

provider"aws"

{access\_key = ""secret\_key=""region="us-east-2"

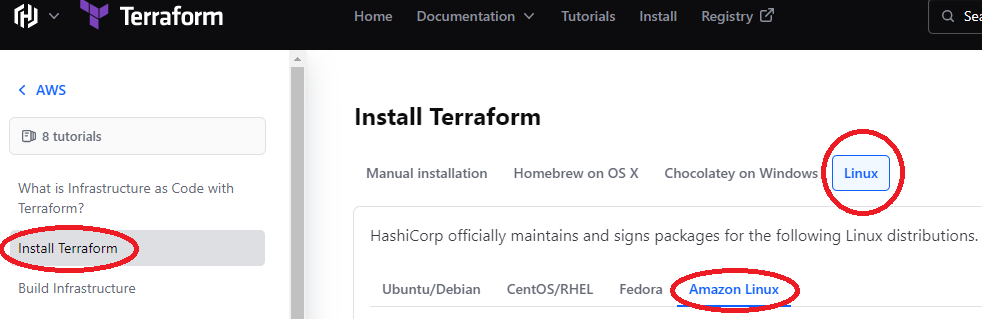
}

1. terraform init
2. terraform fmt
3. terraform validate (syntax errors)
4. terraform plan ( dry run)
5. terraform apply
6. it creates ***terraform.tfstate***
7. terraform destroy

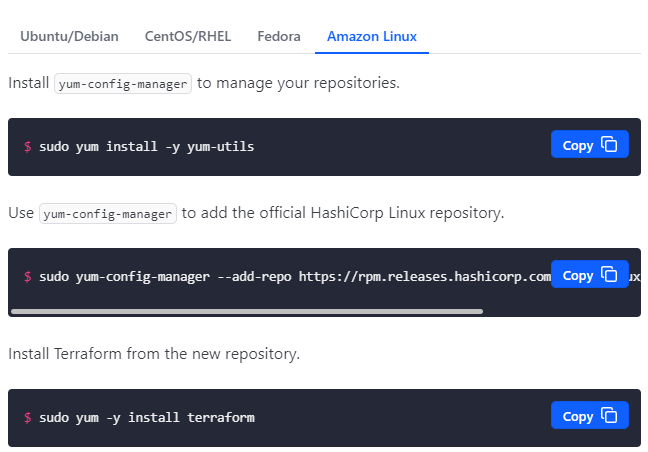
Let us now see it practically.

1. **Installing terraform**.

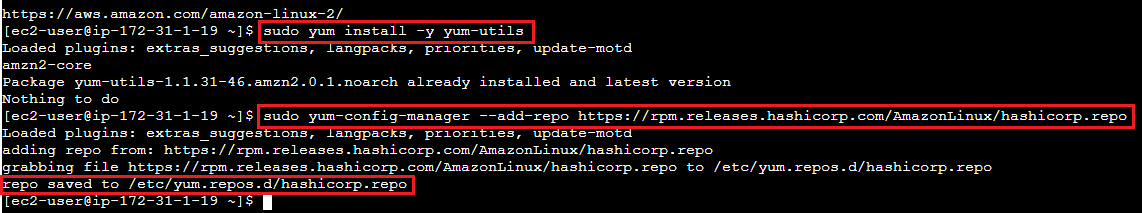
Launch an EC2 instance (AWS Linux platform) and connect to the instance. Go to <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/> to open the link install terraform. Scroll down to select Linux subsequently Amazon Linux as shown below.



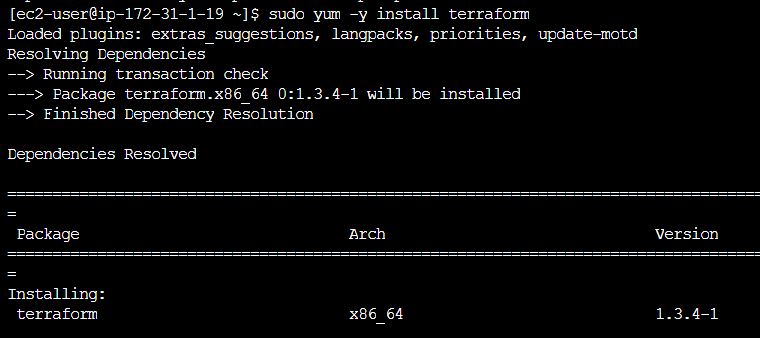
The *yum-util* is a package that allow you to manage your repositories. As we download the repositories and add to the Linux repos.



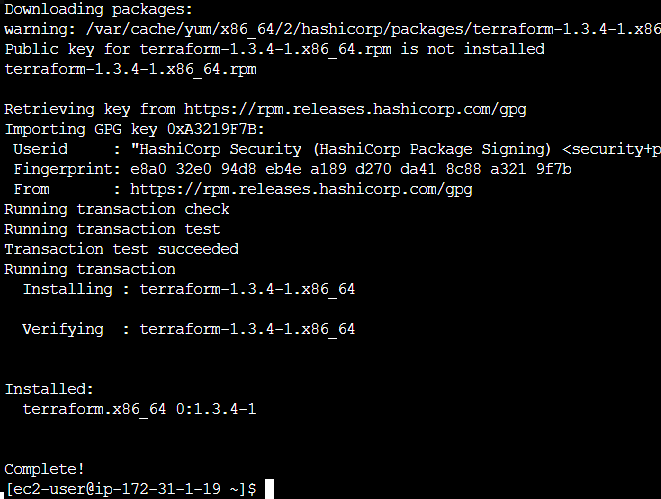
Copy and paste the above three commands consecutively in the command prompt.



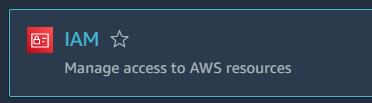
The ‘$sudo yum-config-manager ---’ command saves the file to local ‘/etc/yum.repos.d/hashicorp.repo’



Executing ‘$sudo yum –y install terraform’ command



Terraform installation complete.

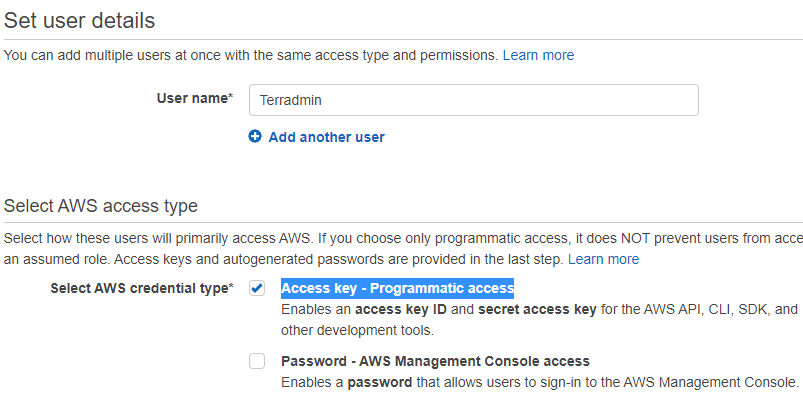
1. **Create IAM role**

Open AWS IAM service to create a User.

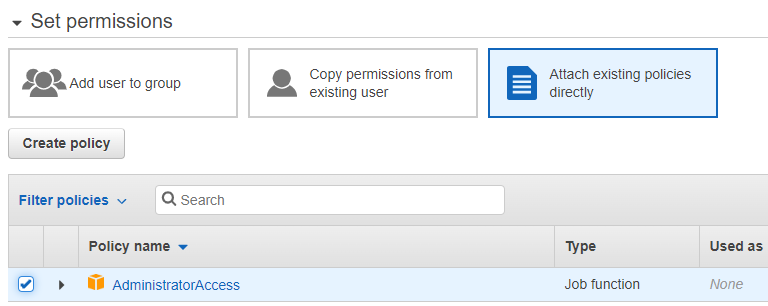


Click on Users link under Access Management group from the navigation pane. Click

Give user name and select **Access key - Programmatic access** as credential type and click Next.

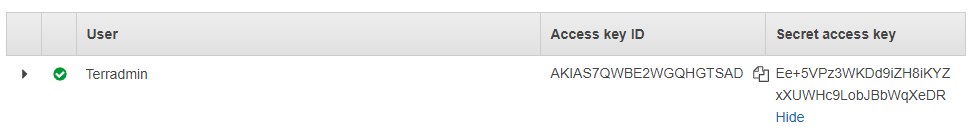


Select *Attach existing policies directly* to select *AdministratorAccess*. Click next.





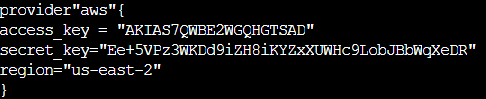
Click next and click It generates the Access and Secret Keys.



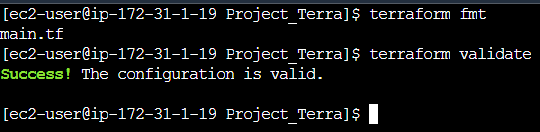
1. **Create main.tf**

Now create a directory for instance, ***Project\_Terra*** and open it to save a terraform script file ***main.tf***.

Copy the Access and Secret Keys as below.



Now go to the command prompt and type ***terraform init***. It installs aws terraform plugins from hashicorp.

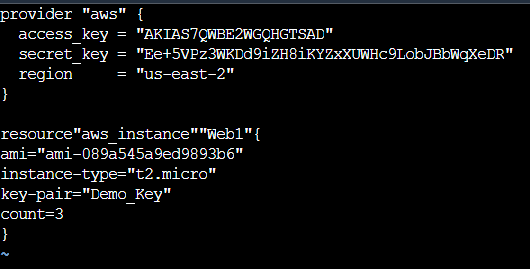


The ***terraform fmt*** command is used to **format your configuration files into a canonical format and style** according to the syntax**.**

The **terraform validate** command validates the syntax.

Now as we set the provider and its credentials, it is time to create the resources/infrastructure.

Open **main.tf** and append the file to add resource block as follows.

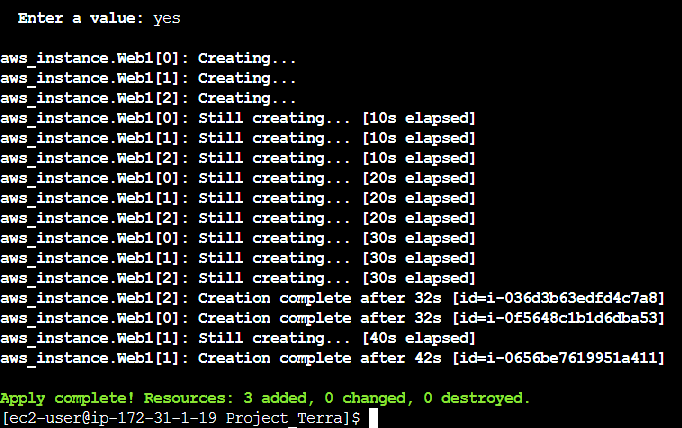


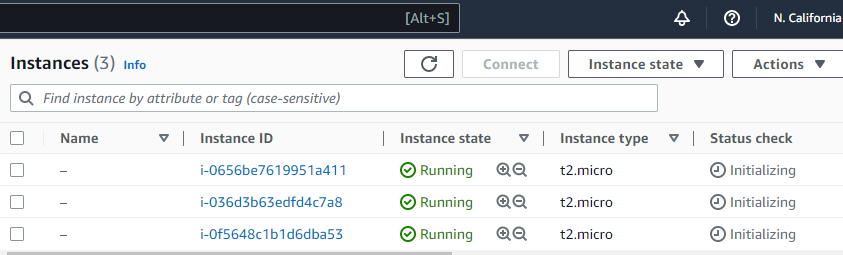
Now run **terraform plan** command for dry run, that is simulation.



Three instances to be added.

Now run the command **terraform apply** to create the resources.

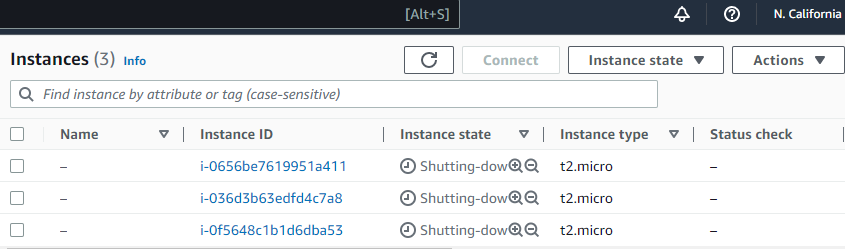




Note that, a **terraform.tfstate** file has been created to save the system state.



The **terraform destroy** command. It destroys all the created resources specified in **.tfstate** file



**A review**

**HCL code structure:** blocktype"resourcetype""resource-name"

{ arguments (key = value)}

**Types of blocks:**

* provider (provider name, credentials, regions..)
* resource (build resources)
* variable
* output
* dynamic
* module
* cloud

**Resource-type:**aws\_vpc, aws\_instance, aws\_subnet

key & value in arguments:

key="string" or numeric value

Difference between Ansible and Terraform.

