terraform cli

* refresh: is used to update the local state (\*.tfstate) with resources created.

terraform refresh -var 'accesskey=<youraccesskey>' -var 'secretkey=<your-secret-key>' .

* taint: is used to mark resources for recreation during next apply

terraform taint <resource-type>.<resource-name>

terraform apply .

# example

terraform taint aws\_subnet.subnet3

# subnet 3 will be marked as tainted

terraform apply -var 'accesskey=<youraccesskey>' -var 'secretkey=<your-secret-key>' .

# since subnet3 is marked as tainted, it will be recreated.

* untaint: if you have tainted any resource by mistake/for any other reason and if you want it to be removed from taint then use untaint.
* graph: Generate a graph from your template. Ensure you install dot from graphviz. [Refer Here](https://www.graphviz.org/download/)

choco install graphviz -y

terraform graph | dot -Tsvg > graph.svg

Terraform outputs

* Output is result of the infra provisioning which can be shared
* To Create outputs, create a new file called as outputs.tf with following content

output "vpc-id" {

value = "${aws\_vpc.myvpc.id}"

}

output "subnet1-id" {

value = "${aws\_subnet.subnet1.id}"

}

* Now execute terraform apply and observe the output

Terraform modules

* Module is reusable terraform configuration.
* Terraform community shares many modules for resuse in [Terraform Registry](https://registry.terraform.io/)
* Terraform templates written by us can also be reused as modules.

Using Terraform module from Local folder

* Create a new directory called as moduledemo
* in the module demo create one file main.tf with following content

module "<name>" {

source = "../hello-tf"

accesskey = ""

secretkey = ""

}

* Now execute

terraform init

terraform apply .

* In hello-tf all the variables will be arguments to module and all the outputs will be attributes of module.
* Any varaible without default is required argument and variable with default is optional argument.

## Data-Sources

* For creating ec2 machine, we need to provide
  + subnet-id
  + security group
  + key value pair
* One approach create every thing and use the attributes.
* If we want to use existing subnet-id, security group and key-value pair, we need to know ids, for this terraform has data sources which can query the information from providers.
* Every provider gives various data sources much like resources.
* Lets create a simple data source to pull the information of default subnet and create a new subnet

data "aws\_vpc" "default" {

default = true

}

resource "aws\_subnet" "extra" {

cidr\_block = "172.31.48.0/20"

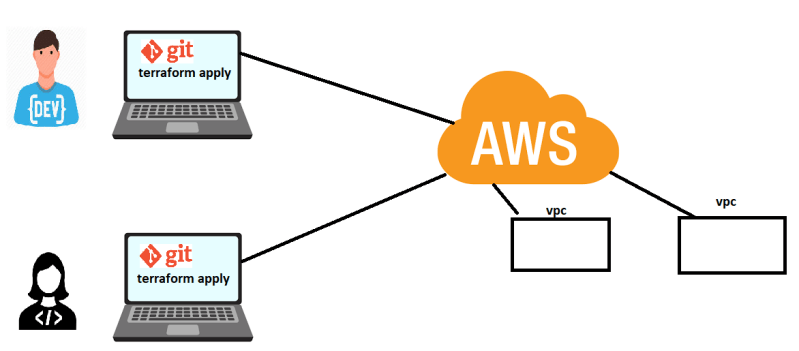
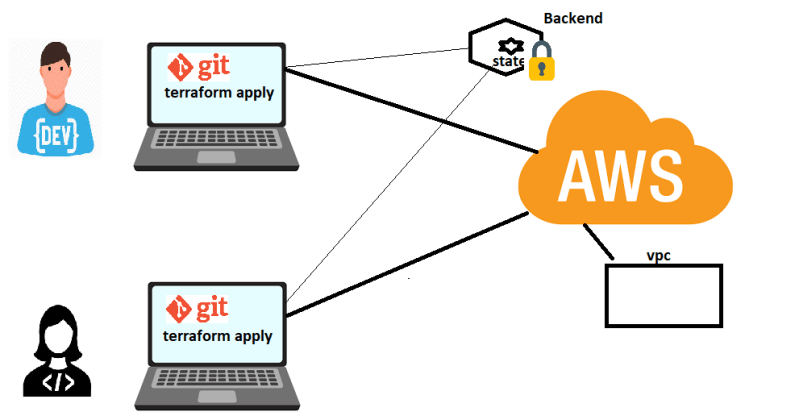
vpc\_id = "${data.aws\_vpc.default.id}"

}

## Terraform provisioning

* Execution of scripts/ansible/chef after creation of Virtual Machines is supported by terraform provisioners. [Refer](https://www.terraform.io/docs/provisioners/index.html)

## Backends

* Two tf developers have same terraform script and they have applied terraform, it creates two different resources, as the state file is stored on individual developers laptop 
* Now, if we want to restrict these two developers in such a way, whenever they execute terraform it should not create two different but one resource.
* Terraform supports backends, to store state remotely and terraform also supports locking feature to avoid simultaneous access to terraform state. 
* [Refer](https://www.terraform.io/docs/backends/index.html)

