DVS Technologies Aws & Devops

Compiled and Scrutinized by

Mr. Shaan Shaik

(Senior DevOps Lead)

DVS Technologies

Words To The Students

Though we have taken utmost efforts to present you this book error free, but still it may contain some errors or mistakes. Students are encouraged to bring, if there are any mistakes or errors in this document to our notice. So that it may be rectified in the next edition of this document.

"Suppressing your doubts is Hindering your growth".

We urge you to work hard and make use of the facilities we are providing to you, because there is no substitute for hard work. We wish you all the best for your future.

"The grass isn't greener on the other side; the grass is greener where you water it."

You and your suggestions are valuable to us; Help us to serve you better. In case of any suggestions, grievance, or complaints, please feel free to write us your suggestions, grievance and feedback on the following

Dvs.training@gmail.com

1. Infrastructure as a Code (IaC)

Terraform Introduction ::

Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation

If you search the Internet for "infrastructure-as-code", it's pretty easy to come up with a list of the most popular tools:

Chef
Puppet
Ansible
SaltStack
CloudFormation
Terraform

All the above tools helps us to manage our infrastructure in the form of code.

But question is simple why ?

should we choose "terraform" than all these. If you observe all the above tools are open source and they have their own communities and the contribution & one more thing is they all are enterprise tools.

Even we have "cloud formation" for automating the things with AWS than terraform. but question remains same why terraform ??

Configuration Management vs Orchestration ::

The above mentioned tools except "CloudFormation & Terraform" all other tools are basically configuration management tools.

Which means that they are used to manage and install the s/w or helps to maintain a state of the particular machine.

But "Terraform" & "CloudFormation" are the Orchestration tools which means that they are designed to provison the machines & their infrastructure. Once the machine is builded you can use the configuration management tools for performing your task.

Mutable Infrastructure vs Immutable Infrastructure ::

DVS Technologies

Mutable --> Configuration management tools

using configuration management tools, we can deploy the new software versions based up on the environment we are choosing. But if you observe each server will be having a separate version based up on the environment.

Immutable --> Orchestration tools

But if you are choosing the Orchestration tools you can simply maintain all the servers with a single version
of OS. It's simple create a simple OS image and start deploy the servers as per the requirement and all your
old machines will be replaced with the newly builded machines and all the machines will have same version of
the package installed !!

Procedural vs Declarative ::

Procedural approach means if you want to achieve something you need to mention the things in an programmatic approach. "Chef & Ansible" works on the same.

But in Declarative approach you no need to worry about flow it will automatically gets the respective information based up the resource what we are choosing.

For example if you want to create 10 servers with app version v1 then the code for different tools will be like below.

using Ansible : (Procedural approach)

```
- ec2:
    count: 10
    ami: app-v1
    instance_type: t2.micro
```

DVS Technologies

Using terraform : (Declarative)

```
resource "aws_instance" "example" {
  count = 10
  ami = "ami-v1"
  instance_type = "t2.micro"
}
```

Till now its fine no much changes in both the configuration. But question is what will happen if the load is high and if you want to add 5 more servers.

Using Ansible you need to specify the code like below.

```
- ec2:
        count: 15
        ami: app-v1
        instance_type: t2.micro
```

Soon after executing this code, you will get a 15 more servers along with 10 machines so total will be 25 servers. But your desire state is to have only 5 machines without changing the code. Which means that you need to again re-write the entire code and find the previous machines and has to do all the other stuff.

Using Terraform you need to specify the code like below.

```
resource "aws_instance" "exampe" {
        count = 15
        ami = "ami-v1"
        instance_type = "t2.micro"
        }
```

Now what Terraform will do it, it won't create 15 more servers it will simply create 5 servers because it is well aware of the current state whatever it is having. Hence you no need to break you head to write new code.

DVS Technologies

Disadvantages:

Of course, there are downsides to declarative languages too. Without access to a full programming language, your expressive power is limited. For example, some types of infrastructure changes, such as a rolling, zero-downtime deployment, are hard to express in purely declarative terms. Similarly, without the ability to do "logic" (e.g. if-statements, loops), creating generic, reusable code can be tricky (especially in CloudFormation).

Client/Server Architecture vs Client-Only Architecture ::

Chef,puppet & salt stack are purely based on "Client/Server". Which indeed there are many hiccups when you are dealing with these tools, like

1. you need to install the client in all the machines in order to get the desired state as per the requirement
2. you should require a manageable server which should give the instructions to the client machines.
3. you will get all the issues with the network, client,management and etc.

Ansible, CloudFormation & Terraform are purely client-only Architecture, which in deed you no need to install any agents as part of your machines in order to do the management.

CloudFormation is also client/server, but AWS handles all the server details so transparently, that as an end user, you only have to think about the client code. The Ansible client works by connecting directly to your servers over SSH

Terraform uses cloud provider APIs to provision infrastructure, so there are no new authentication mechanisms beyond what you're using with the cloud provider already, and there is no need for direct access to your servers

Final Conclusion:

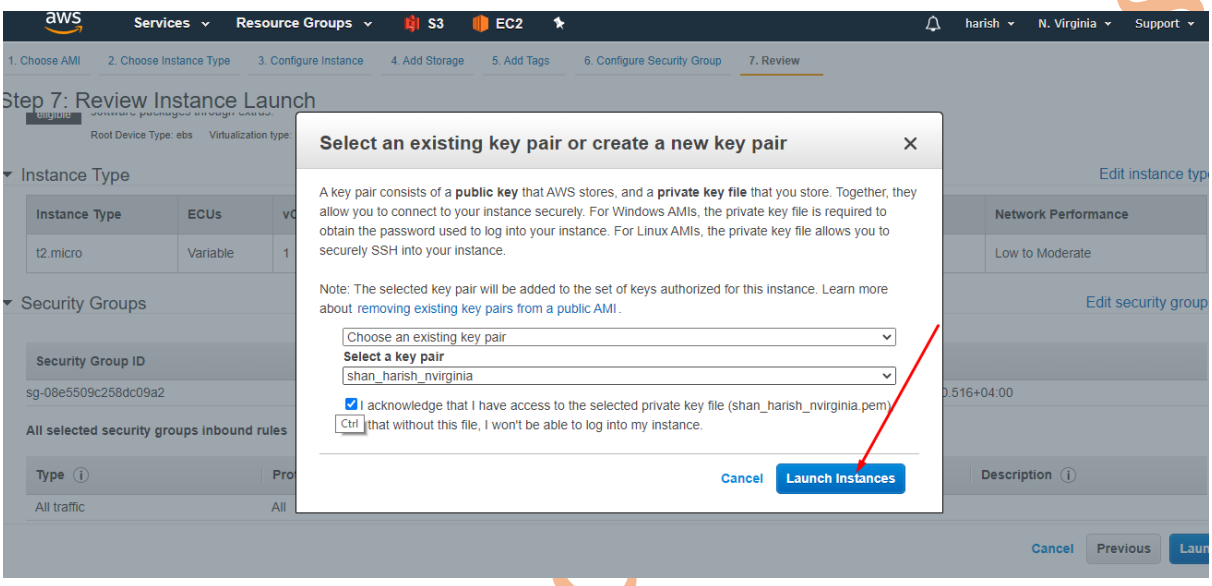
Of course, Terraform isn't perfect. It's younger and less mature than all the other tools on the list: whereas Puppet came out in 2005, Chef in 2009, SaltStack and CloudFormation in 2011, and Ansible in 2012,
Terraform came out just 4 years ago, in 2014.

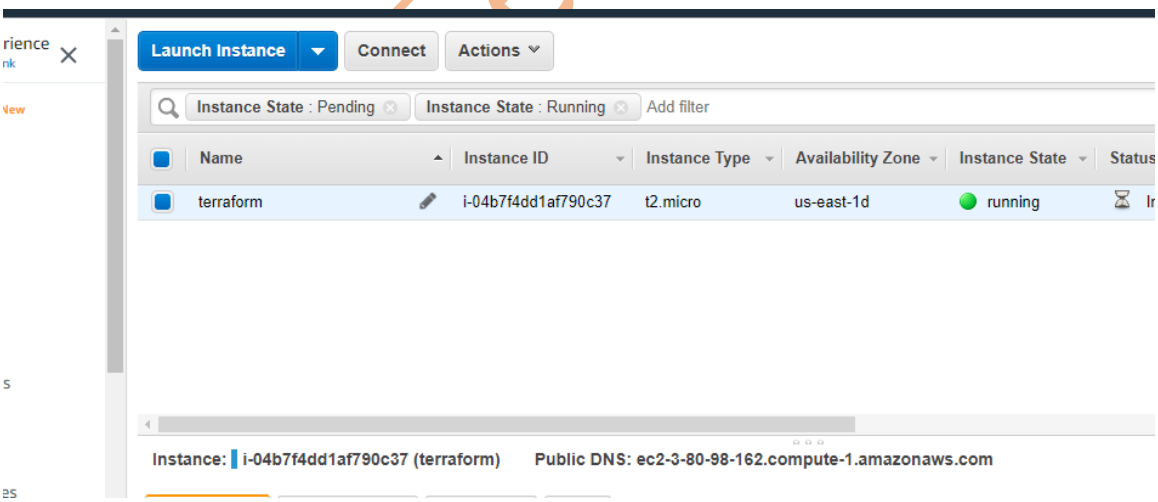
Bugs are relatively common (e.g. there are over 800 open issues with the label "bug"), although the vast majority are harmless eventual consistency issues that go away when you Re-run Terraform

DVS Technologies

2. Installation and Configuration

Create one Ec2





Configure your Aws Cli as specified below:

```
root@ip-172-31-26-113:~
[ec2-user@ip-172-31-26-113 ~]$ sudo su -
[root@ip-172-31-26-113 ~]# aws configure
AWS Access Key ID [None]: [redacted]URKISNRQ2HQN
AWS Secret Access Key [None]: [redacted]/kujPvneAXQc8R3LtiBAMKAOwEioz
Default region name [None]: us-east-1
Default output format [None]: json
[root@ip-172-31-26-113 ~]#
```

Let's Install the terraform:

sudo curl -O
https://releases.hashicorp.com/terraform/0.11.13/terraform_0.11.13_linux_amd64.zip
sudo yum install -y unzip
sudo unzip terraform_0.11.13_linux_amd64.zip -d /usr/local/bin/

```
Default output format [None]: json
[root@ip-172-31-26-113 ~]# sudo curl -O https://releases.hashicorp.com/terraform/0.11.13/terraform_0.11.13_linux_amd64.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 20.1M  100 20.1M    0     0  41.0M      0 --:--:-- --:--:-- --:--:-- 40.9M
[root@ip-172-31-26-113 ~]# sudo yum install -y unzip
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package unzip-6.0-20.amzn2.x86_64 already installed and latest version
Nothing to do
[root@ip-172-31-26-113 ~]# sudo unzip terraform_0.11.13_linux_amd64.zip -d /usr/local/bin/
Archive:  terraform_0.11.13_linux_amd64.zip
  inflating: /usr/local/bin/terraform
[root@ip-172-31-26-113 ~]# terraform -v
Terraform v0.11.13

Your version of Terraform is out of date! The latest version
is 0.13.3. You can update by downloading from www.terraform.io/downloads.html
[root@ip-172-31-26-113 ~]#
```

.

3. Working with terraform

Let's Create a key pair as specified below.

```
[root@terraform ~]# mkdir basics
[root@terraform ~]# cd basics/
[root@terraform basics]# ls -l
total 0
[root@terraform basics]# vi main.tf
[root@terraform basics]#
[root@terraform basics]#
[root@terraform basics]#
[root@terraform basics]# cat main.tf
provider "aws" {
        region= "us-east-1"
}

resource "aws_key_pair" "deployer" {
  key_name   = "dvsbatch5"
  public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZBp5vizP
vaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fs1NfRPW5mYAM49f4fhtxP
b5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE6lymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/06niW
rOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
}
```

Init:

```
[root@terraform basics]# ls -al
total 4
drwxr-xr-x 2 root root  21 Sep 19 10:10 .
dr-xr-x--- 6 root root 190 Sep 19 10:09 ..
-rw-r--r-- 1 root root 523 Sep 19 10:10 main.tf
[root@terraform basics]# terraform init .

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "aws" (2.70.0)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.70"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

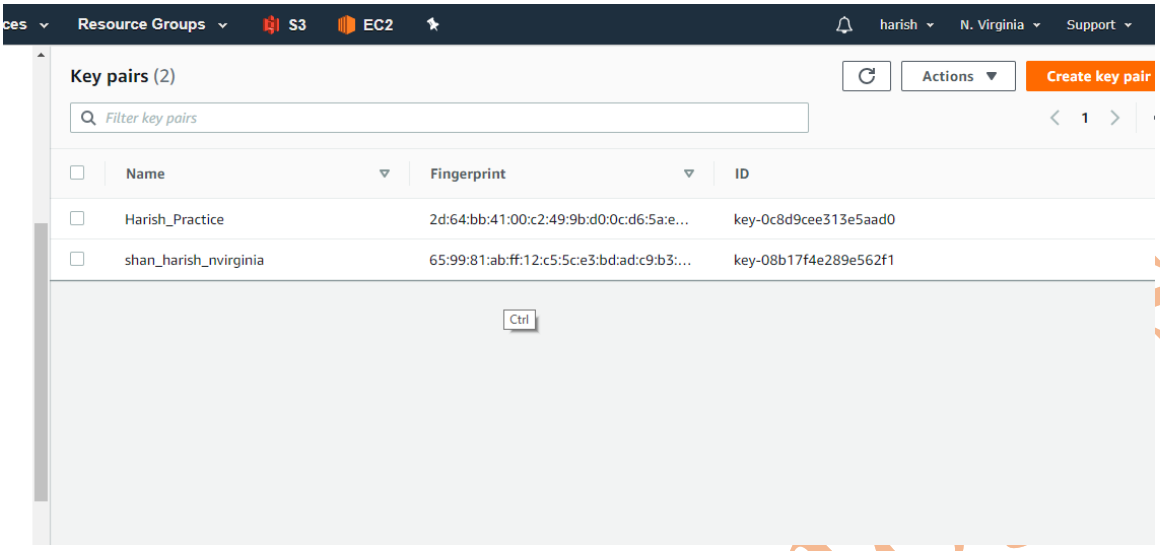
```
[root@terraform basics]# pwd
/root/basics
[root@terraform basics]# ls -la
total 4
drwxr-xr-x 3 root root  39 Sep 19 10:11 .
dr-xr-x--- 6 root root 190 Sep 19 10:09 ..
-rw-r--r-- 1 root root 523 Sep 19 10:10 main.tf
drwxr-xr-x 3 root root  21 Sep 19 10:11 .terraform
[root@terraform basics]# cat main.tf
provider "aws" {
        region= "us-east-1"
}

resource "aws_key_pair" "deployer" {
  key_name   = "dvsbatch5"
  public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZBp5viz
vaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f4fhtx
b5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE61ymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/06ni
rOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
}
[root@terraform basics]# ls -l .terraform/plugins/linux_amd64/
total 154456
-rwxr-xr-x 1 root root        79 Sep 19 10:11 lock.json
-rwxr-xr-x 1 root root 158158848 Sep 19 10:11 terraform-provider-aws_v2.70.0_x4
[root@terraform basics]#
```

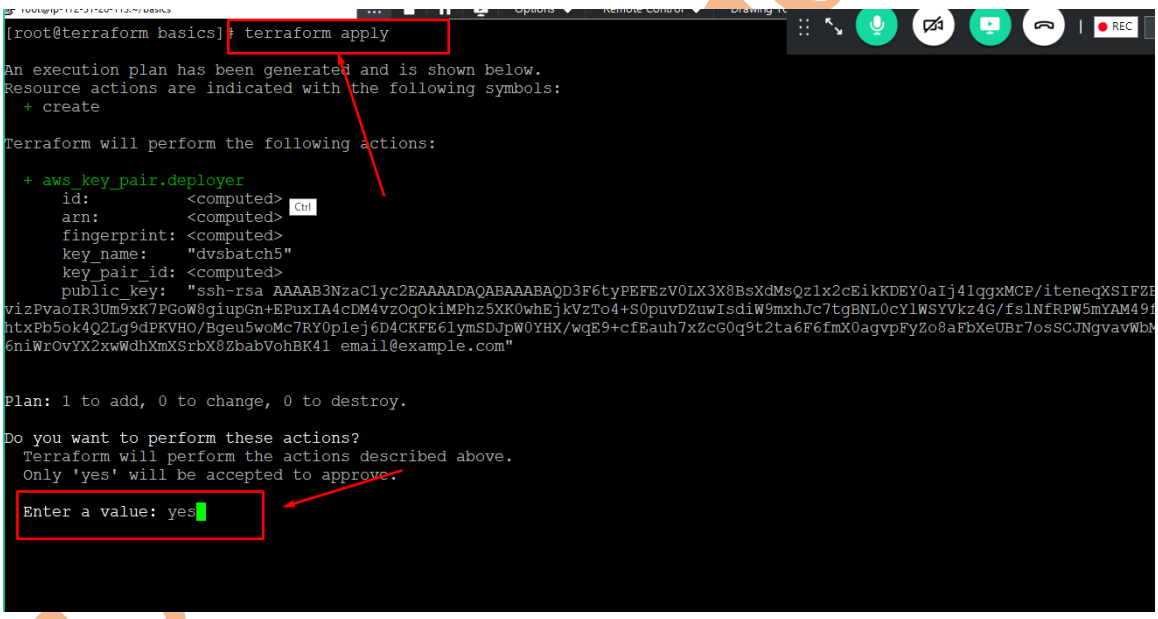
Plan:

```
[root@terraform basics]# terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_key_pair.deployer
      id:          <computed>
      arn:         <computed>
      fingerprint: <computed>
      key_name:    "dvsbatch5"
      key_pair_id: <computed>
      public_key:  "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZBp
vizPvaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f4
htxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE61ymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/
6niWrOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"


Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------
```

Apply:

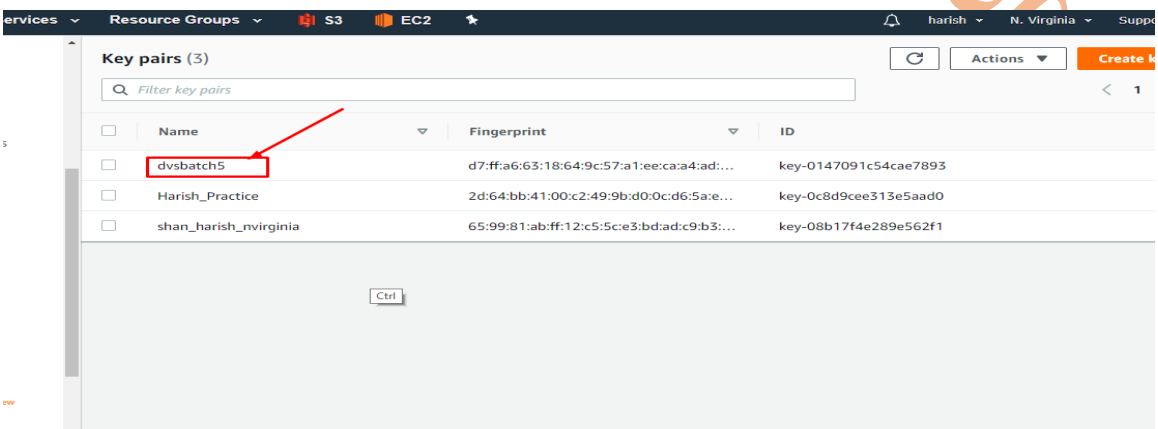
Before:


Resource Groups
S3
EC2
harish
N. Virginia
Support
Key pairs (2)
Actions
Create key pair
Filter key pairs
Name
Fingerprint
ID
Harish_Practice
2d:64:bb:41:00:c2:49:9b:d0:0c:d6:5a:e...
key-0c8d9cee313e5aad0
shan_harish_nvirginia
65:99:81:ab:ff:12:c5:5c:e3:bd:ad:c9:b3:...
key-08b17f4e289e562f1


[root@terraform basics]# terraform apply
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:
+ aws_key_pair.deployer
id: <computed>
arn: <computed>
fingerprint: <computed>
key_name: "dvsbatch5"
key_pair_id: <computed>
public_key: "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZE
vizPvaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f
htxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE6lymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM
6niWrOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes

```
aws_key_pair.deployer: Creating...
  arn:         "" => "<computed>"
  fingerprint: "" => "<computed>"
  key_name:    "" => "dvsbatch5"
  key_pair_id: "" => "<computed>"
  public_key:  "" => "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEi
p5vizPvaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBN
4fhtxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE6lymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0a
/06niWrOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
aws_key_pair.deployer: Creation complete after 0s (ID: dvsbatch5)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@terraform basics]#
```

After:

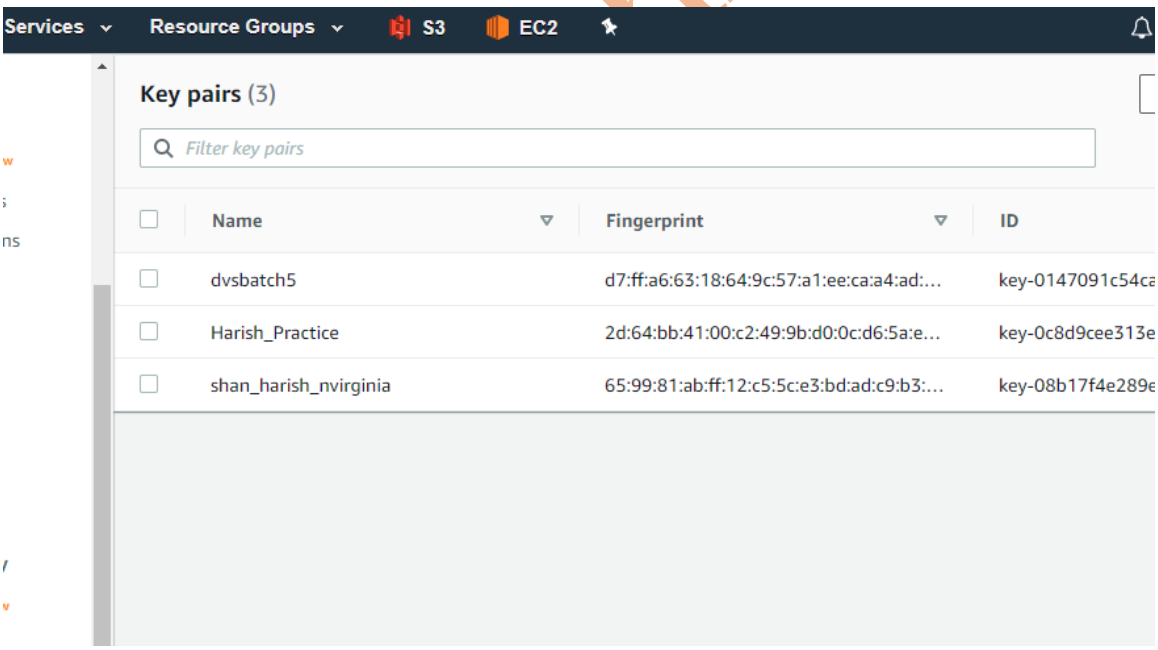


Terraform state file:

```
[root@terraform basics]# ls -al
total 12
drwxr-xr-x 3 root root   96 Sep 19 10:22 .
dr-xr-x--- 6 root root  190 Sep 19 10:09 ..
-rw-r--r-- 1 root root  523 Sep 19 10:10 main.tf
drwxr-xr-x 3 root root   21 Sep 19 10:17 .terraform
-rw-r--r-- 1 root root 1714 Sep 19 10:22 terraform.tfstate
-rw-r--r-- 1 root root  318 Sep 19 10:22 terraform.tfstate.backup
[root@terraform basics]# cat terraform.tfstate
{
    "version": 3,
    "terraform_version": "0.11.13",
    "serial": 3,
    "lineage": "1cb8d07e-9521-c777-bab1-1f12d519f66c",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {
                "aws_key_pair.deployer": {
                    "type": "aws_key_pair",
                    "depends_on": [],
                    "primary": {
                        "id": "dvsbatch5",
                        "attributes": {
                            "arn": "arn:aws:ec2:us-east-1:907814406801:key-pair/dvsbatch5"
                            "fingerprint": "d7:ff:a6:63:18:64:9c:57:a1:ee:ca:a4:ad:c2:81:6
                            "id": "dvsbatch5",
```

Destroy:

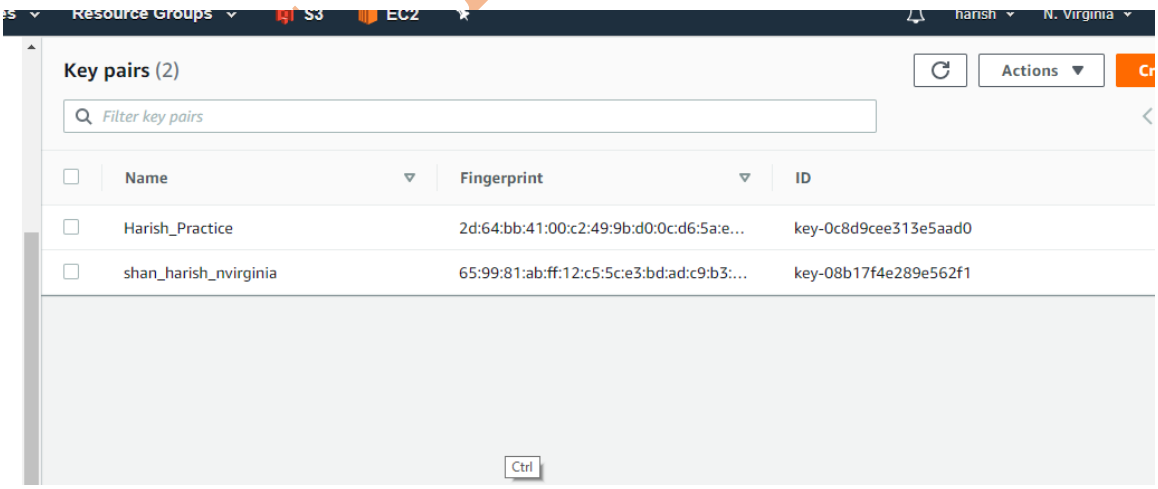
Before:



```
[root@terraform basics]# terraform destroy
aws_key_pair.deployer: Refreshing state... (ID: dvsbatch5)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  - aws_key_pair.deployer


Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

```
aws_key_pair.deployer: Destroying... (ID: dvsbatch5)
aws_key_pair.deployer: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
[root@terraform basics]#
```

After:



```
[root@terraform basics]#
[root@terraform basics]# terraform show

[root@terraform basics]# cat terraform.tfstate
{
    "version": 3,
    "terraform_version": "0.11.13",
    "serial": 4,
    "lineage": "1cb8d07e-9521-c777-bab1-1f12d519f66c",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {},
            "depends_on": []
        }
    ]
}
[root@terraform basics]#
```

```
[root@terraform basics]# terraform apply -auto-approve
aws_key_pair.deployer: Creating...
  arn:         "" => "<computed>"
  fingerprint: "" => "<computed>"
  key_name:    "" => "dvsbatch5"
  key_pair_id: "" => "<computed>"
  public_key:  "" => "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSI
p5vizPvaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM
4fhtxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE61ymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvav
/06niWrOvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
aws_key_pair.deployer: Creation complete after 0s (ID: dvsbatch5)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@terraform basics]# terraform show
aws_key_pair.deployer:
  id = dvsbatch5
  arn = arn:aws:ec2:us-east-1:907814406801:key-pair/dvsbatch5
  fingerprint = d7:ff:a6:63:18:64:9c:57:a1:ee:ca:a4:ad:c2:81:62
  key_name = dvsbatch5
  key_pair_id = key-0174df53ada20b44f
  public_key = ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZBp5vi
aoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f4fht
5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE61ymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/06n
OvYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com
```

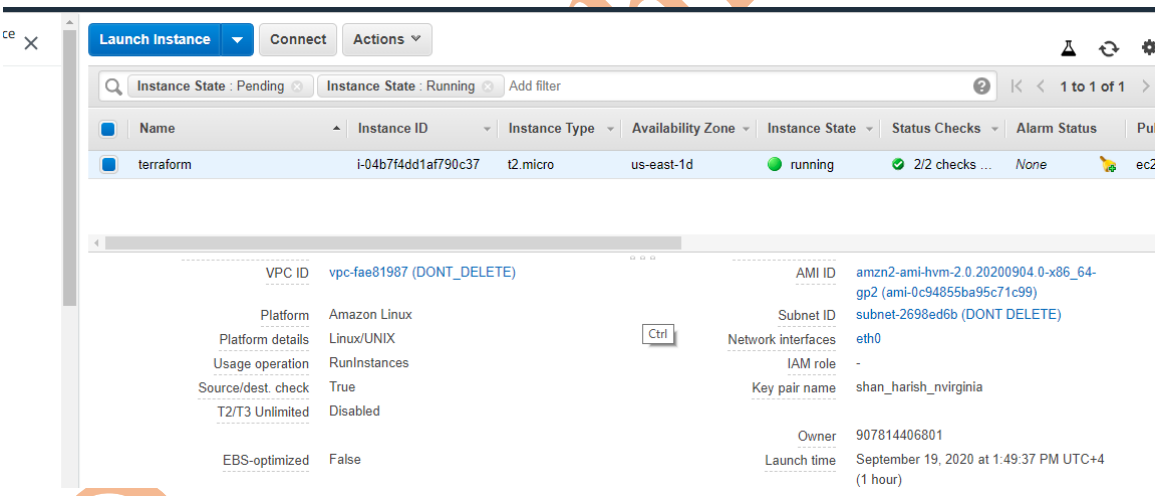
```
[root@terraform basics]# terraform providers
.
└── provider.aws

[root@terraform basics]#
```

4. Working with variables

Let's create a server with static variables & later convert them to variables

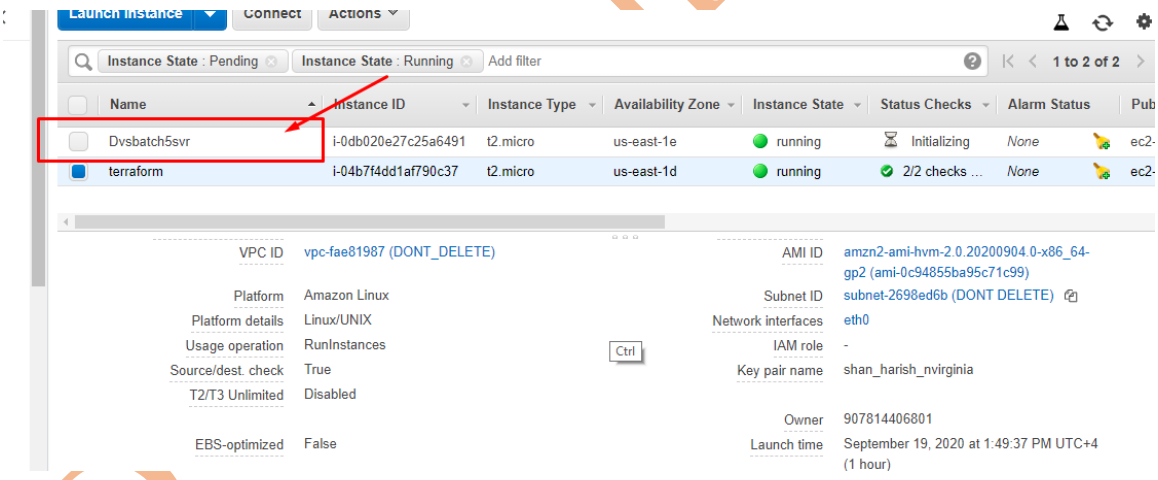
```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}
resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "Dvsbatch5svr"
  }
}
[root@terraform basics]# terraform init .

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.
```

Before:



```
[root@terraform basics]# terraform apply -auto-approve
aws_instance.myec2: Creating...
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
  get_password_data:            "" => "false"
  host_id:                      "" => "<computed>"
  instance_state:               "" => "<computed>"
  instance_type:                "" => "t2.micro"
  ipv6_address_count:           "" => "<computed>"
  ipv6_addresses.#:             "" => "<computed>"
  key_name:                     "" => "shan_harish_nvirginia"
  metadata_options.#:           "" => "<computed>"
  network_interface.#:          "" => "<computed>"
  network_interface_id:         "" => "<computed>"
  outpost_arn:                  "" => "<computed>"
  password_data:                "" => "<computed>"
  placement_group:              "" => "<computed>"
  primary_network_interface_id: "" => "<computed>"
```

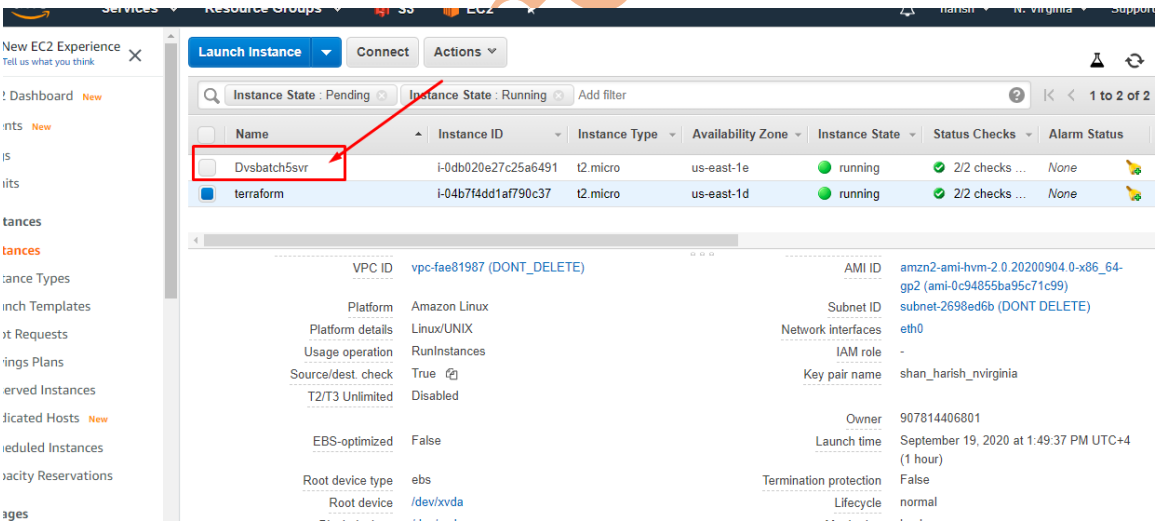
```
aws_instance.myec2: Still creating... (30s elapsed)
aws_instance.myec2: Creation complete after 31s (ID: i-0db020e27c25a6491)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@terraform basics]#
```



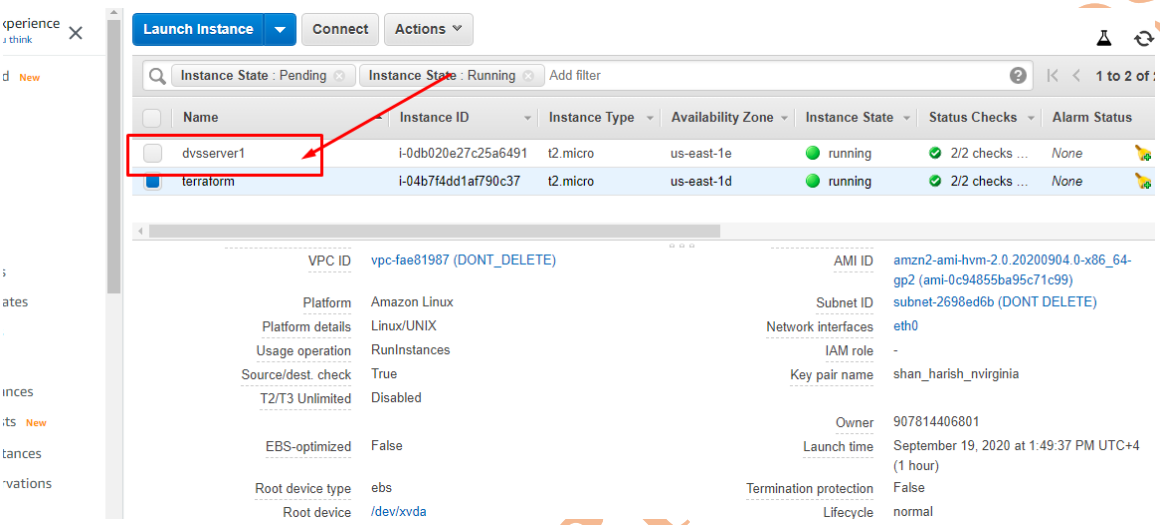
1. **Default variables:**

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {

        default = "dvsserver1"

}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}
[root@terraform basics]#
```

Before:



```
}
[root@terraform basics]# terraform apply -auto-approve
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)
aws_instance.myec2: Modifying... (ID: i-0db020e27c25a6491)
  tags.Name: "Dvsbatch5svr" => "dvsserver1"
aws_instance.myec2: Modifications complete after 1s (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[root@terraform basics]#
```



2. **Dynamic data for the variables:**

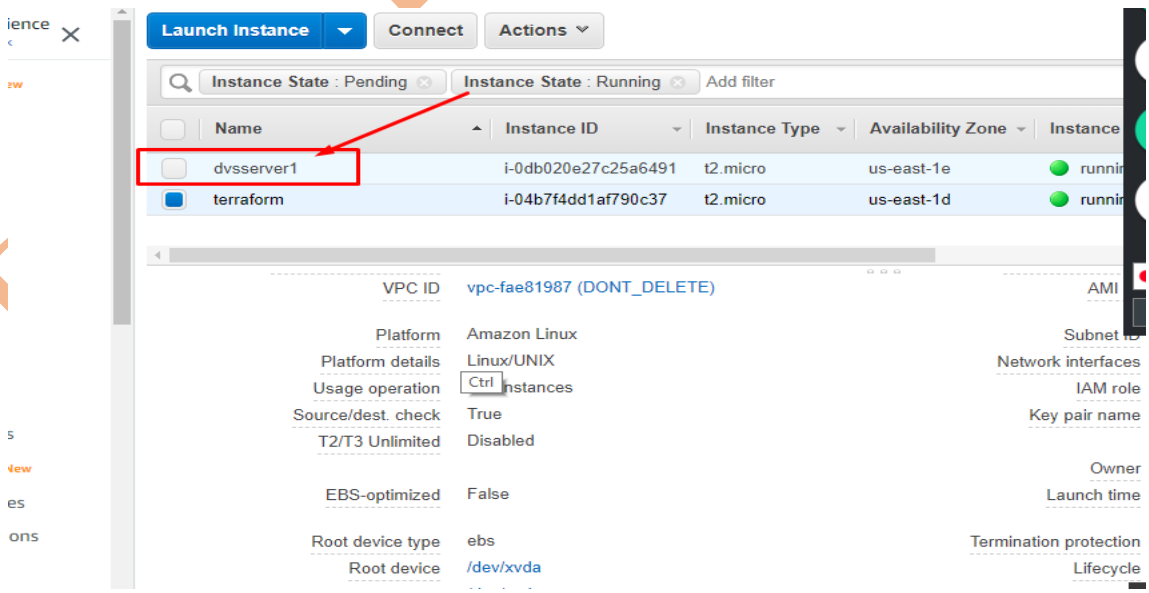
Simply remove the default value section like below

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {


}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}
[root@terraform basics]# terraform apply -auto-approve
var.server_tag
  Enter a value: dvsnewserver

aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)
aws_instance.myec2: Modifying... (ID: i-0db020e27c25a6491)
  tags.Name: "dvsserver1" => "dvsnewserver"
aws_instance.myec2: Modifications complete after 1s (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[root@terraform basics]#
```

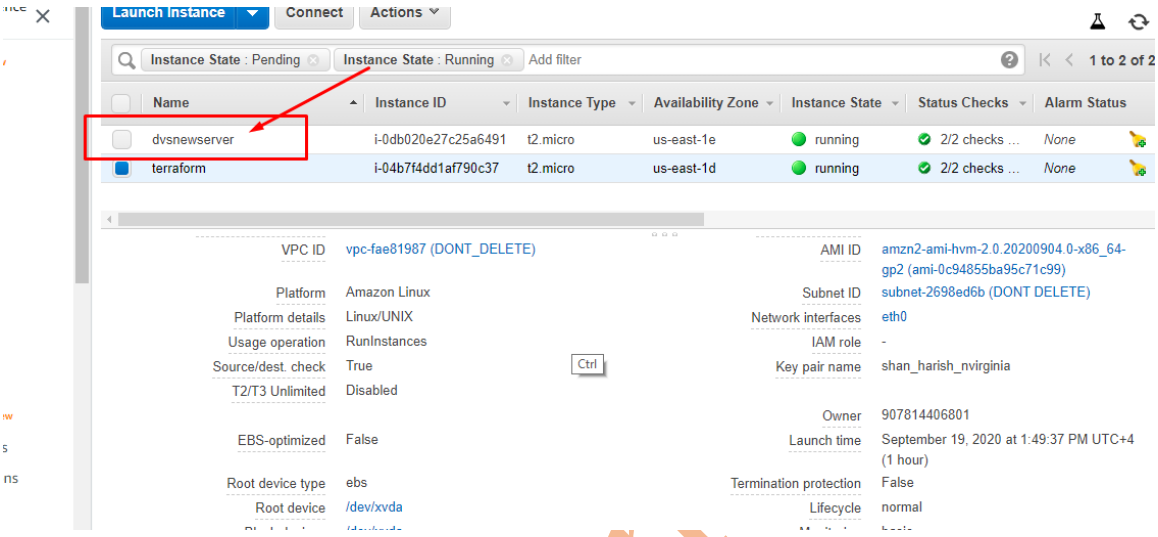
Don't declare default value

pgm will ask for a value

Before:



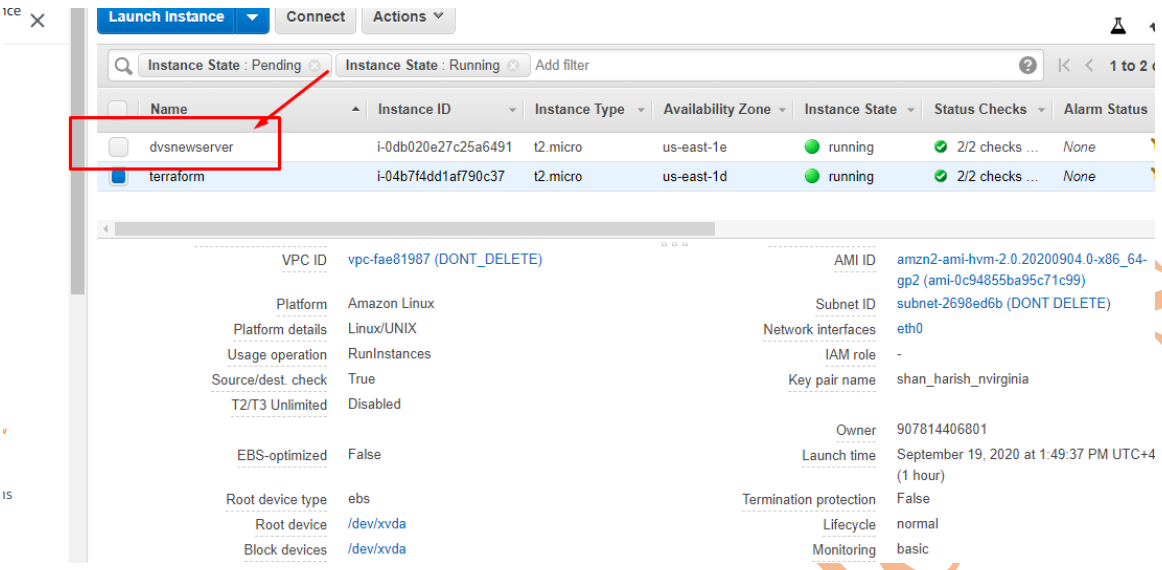
After:



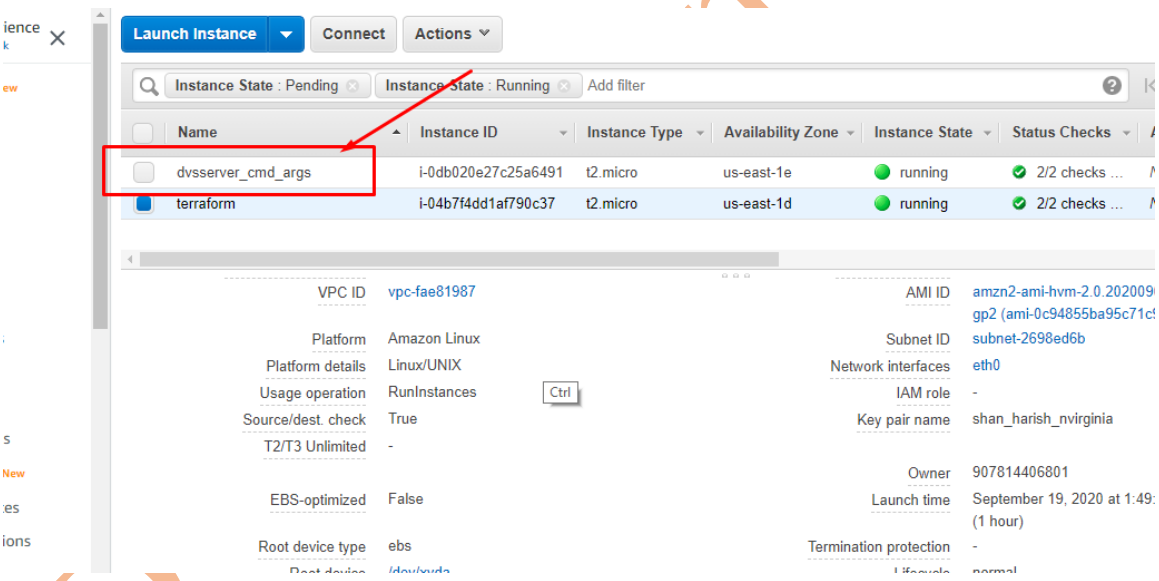
3. **Passing variables from Commandline:**

```
}
[root@terraform basics]# terraform apply -auto-approve -var server_tag=dvsserver_cmd_args
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)
aws_instance.myec2: Modifying... (ID: i-0db020e27c25a6491)
  tags.Name: "dvsnewserver" => "dvsserver_cmd_args"
aws_instance.myec2: Modifications complete after 0s (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[root@terraform basics]#
```

Before:



After:



4. **Variables via files:**

```
[root@terraform basics]# vi myvars.tfvars
[root@terraform basics]# cat myvars.tfvars
server_tag="dvsvarsfromfiles"
[root@terraform basics]# terraform apply -auto-approve -var-file=myvars.tfvars
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)
aws_instance.myec2: Modifying... (ID: i-0db020e27c25a6491)
  tags.Name: "dvsserver_cmd_args" => "dvsvarsfromfiles"
aws_instance.myec2: Modifications complete after 1s (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[root@terraform basics]#
```

Example file for your variables:

```
[root@terraform basics]# cat myvars.tfvars
server_tag="dvsvarsfromfiles"
myvar1="myvalue"
myvar2="myvalue2"
[root@terraform basics]#
```

Output Section:

```
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {


}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}

/* Output Section */

output "mypub_dns" {

        value = "${}"
}
```

aws_instance.myec2

"${aws_instance.myec2.public_dns}"

"${ . "

"${

```
root@terraform:~/basics
aws_instance.myec2:
  id = i-0db020e27c23a6491
  ami = ami-0c94855ba95c71c99
  arn = arn:aws:ec2:us-east-1:907814406801:instance/i-
  associate_public_ip_address = true
  availability_zone = us-east-1e
  cpu_core_count = 1
  cpu_threads_per_core = 1
  credit_specification.# = 1
  credit_specification.0.cpu_credits = standard
  disable_api_termination = false
  ebs_block_device.# = 0
  ebs_optimized = false
  ephemeral_block_device.# = 0
  get_password_data = false
  hibernation = false
  iam_instance_profile =
  instance_state = running
  instance_type = t2.micro
  ipv6_address_count = 0
  ipv6_addresses.# = 0
  key_name = shan_harish_nvirginia
  metadata_options.# = 1
  metadata_options.0.http_endpoint = enabled
  metadata_options.0.http_put_response_hop_limit = 1
  metadata_options.0.http_tokens = optional
  monitoring = false
  network_interface.# = 0
  outpost_arn =
```

- availability_zone

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {


}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}

/* Output Section */

output "mypub_dns" {
        value = "${aws_instance.myec2.public_dns}"
}
```

```
/* Output Section */

output "mypub_dns" {
        value = "${aws_instance.myec2.public_dns}"
}
[root@terraform basics]# terraform apply -auto-approve -var-file=myvars.tfvars
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns = ec2-100-26-178-178.compute-1.amazonaws.com
[root@terraform basics]#
```

Other example:

```
  }
}

/* Output Section */

output "mypub_dns_harish" {
        value = "${aws_instance.myec2.public_dns},${aws_instance.myec2.availability_zone}"
}
[root@terraform basics]# terraform apply -auto-approve -var-file=myvars.tfvars
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns_harish = ec2-100-26-178-178.compute-1.amazonaws.com,us-east-1e
[root@terraform basics]#
```

DVS Technologies

5. Segregating my code

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {


}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}

/* Output Section */

output "mypub_dns_harish" {
        value = "${aws_instance.myec2.public_dns},${aws_instance.myec2.availability_zone}"
}
[root@terraform basics]#
```

providers → main.tf

variables → variables.tf

resource → main.tf

outputs.tf

output

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

/*Comment section -- Variables */
variable "server_tag" {


}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}

/* Output Section */

output "mypub_dns_harish" {
        value = "${aws_instance.myec2.public_dns},${aws_instance.myec2.availability_zone}"
}
```

* Combination of all sections *

Breaking them in to individual sections:

```
[root@terraform basics]# cat variables.tf
/*Comment section -- Variables */
variable "server_tag" {

        default = "dvsvarsfromfiles"
}
[root@terraform basics]#
```

```
}
[root@terraform basics]# cat outputs.tf
/* Output Section */

output "mypub_dns_harish" {
        value = "${aws_instance.myec2.public_dns},${aws_instance.myec2.availability_zone}"
}

[root@terraform basics]#
```

```
[root@terraform basics]# cat main.tf
provider "aws" {

        region = "us-east-1"

}

resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "shan_harish_nvirginia"
  tags = {
    Name = "${var.server_tag}"
  }
}

[root@terraform basics]#
```

Final testing post code alignment

```
[root@terraform basics]# terraform apply -auto-approve
aws_instance.myec2: Refreshing state... (ID: i-0db020e27c25a6491)

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns_harish = ec2-100-26-178-178.compute-1.amazonaws.com,us-east-1e
[root@terraform basics]#
```

6. Maps & lookups

```
[root@terraform basics]# cat variables.tf
/*Comment section -- Variables */


variable "env" {

        "description" = "dev or prod"

}

variable "my_key_name" {

"default" = {

        "dev" = "dev-key"
        "prod" = "prod-key"
        }
}

variable "my_server_name" {

"default" = {
        "dev" = "dev-server"
        "prod" = "prod-server"

        }
}
[root@terraform basics]#
```

"${var.env}"

env → var.env

"${lookup(var.my_key_name, var.env)}"

"${lookup(var.my_server_name, var.env)}"

${lookup(var.my_server_name,var.env)}"
${lookup(var.my_key_name,var.env)}"

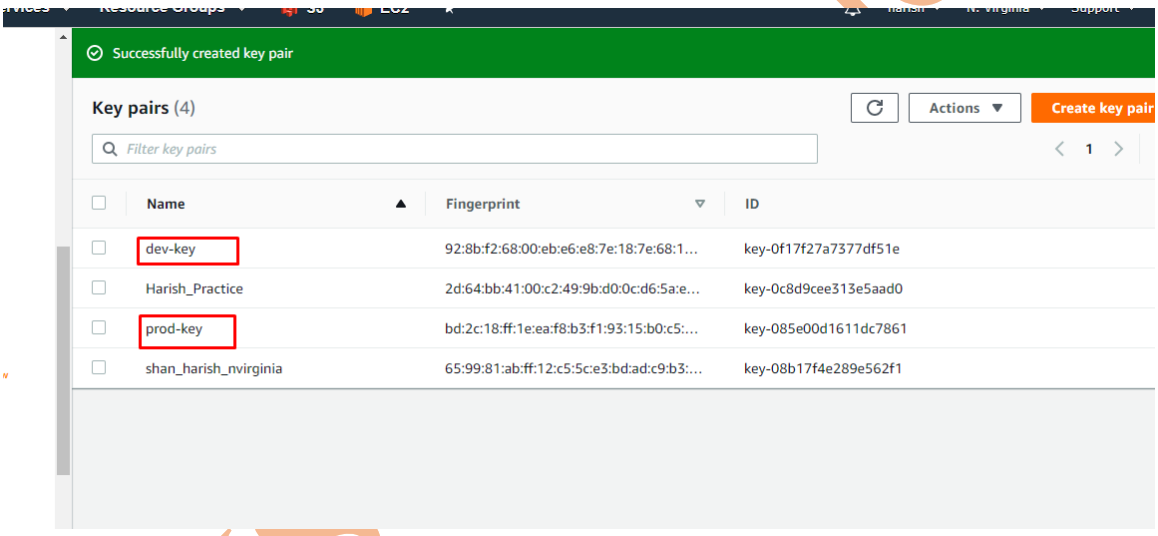
```
[root@terraform basics]#
[root@terraform basics]# cat variables.tf
/*Comment section -- Variables */


variable "env" {

        "description" = "dev or prod"

}

variable "my_key_name" {

"default" = {

        "dev" = "dev-key"
        "prod" = "prod-key"
        }
}

variable "my_server_name" {

"default" = {
        "dev" = "dev-server"
        "prod" = "prod-server"

        }
}
[root@terraform basics]#
```

```
[root@terraform basics]# cat main.tf
provider "aws" {

region = "us-east-1"

}


resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "${lookup(var.my_key_name,var.env)}"
  tags = {
    Name = "${lookup(var.my_server_name,var.env)}"
  }
}
```

```
[root@terraform basics]# cat outputs.tf
/* Output Section */

output "mypub_dns_harish" {
        value = "${aws_instance.myec2.public_dns},${aws_instance.myec2.availability_zone}"
}

[root@terraform basics]#
```

Make sure that you are creating the keypairs:

Dev-key
Prod-key




```
[root@terraform basics]# terraform plan -var env=dev
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

```
    instance_type:                  t2.micro
    ipv6_address_count:             <computed>
    ipv6_addresses.#:               <computed>
    key_name:                       "dev-key"
    metadata_options.#:             <computed>
    network_interface.#:            <computed>
    network_interface_id:           <computed>
    outpost_arn:                    <computed>
    password_data:                  <computed>
    placement_group:                <computed>
    primary_network_interface_id:   <computed>
    private_dns:                    <computed>
    private_ip:                     <computed>
    public_dns:                     <computed>
    public_ip:                      <computed>
    root_block_device.#:            <computed>
    security_groups.#:              <computed>
    source_dest_check:              "true"
    subnet_id:                      <computed>
    tags.%:                         "1"
    tags.Name:                      "dev-server"
    tenancy:                        <computed>
    volume_tags.%:                  <computed>
    vpc_security_group_ids.#:       <computed>
```

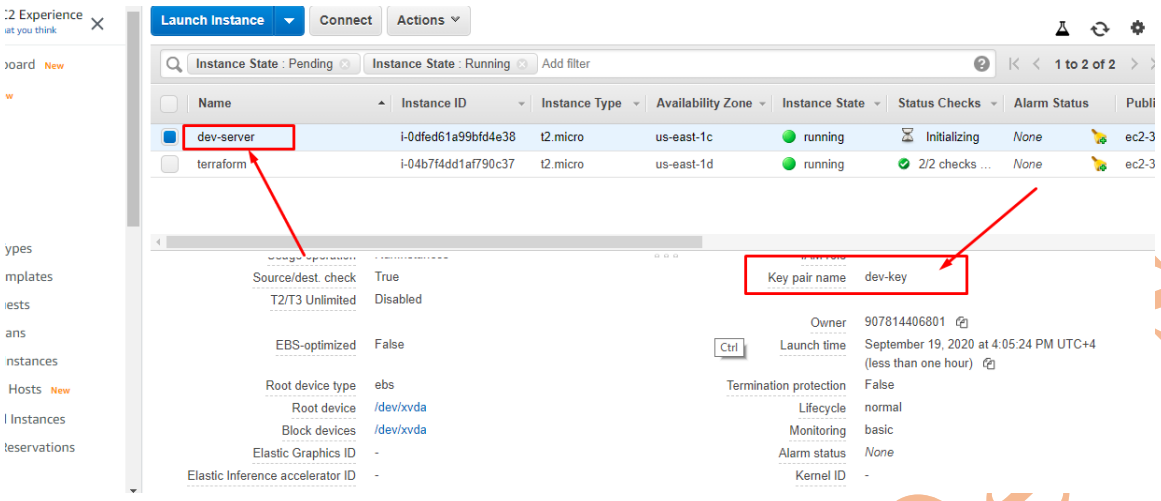
```
[root@terraform basics]#
[root@terraform basics]# terraform plan -var env=prod
Refreshing Terraform state in memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_instance.myec2
      id:                           <computed>
      ami:                          "ami-0c94855ba95c71c99"
      arn:                          <computed>
```

```
      ipv6_address_count:           <computed>
      ipv6_addresses.#:             <computed>
      key_name:                     "prod-key"
      metadata_options.#:           <computed>
      network_interface.#:          <computed>
      network_interface_id:         <computed>
      outpost_arn:                  <computed>
      password_data:                <computed>
      placement_group:              <computed>
      primary_network_interface_id: <computed>
      private_dns:                  <computed>
      private_ip:                   <computed>
      public_dns:                   <computed>
      public_ip:                    <computed>
      root_block_device.#:          <computed>
      security_groups.#:            <computed>
      source_dest_check:            "true"
      subnet_id:                    <computed>
      tags.%:                       "1"
      tags.Name:                    "prod-server"
      tenancy:                      <computed>
      volume_tags.%:                <computed>
      vpc_security_group_ids.#:     <computed>
```

Final testing:

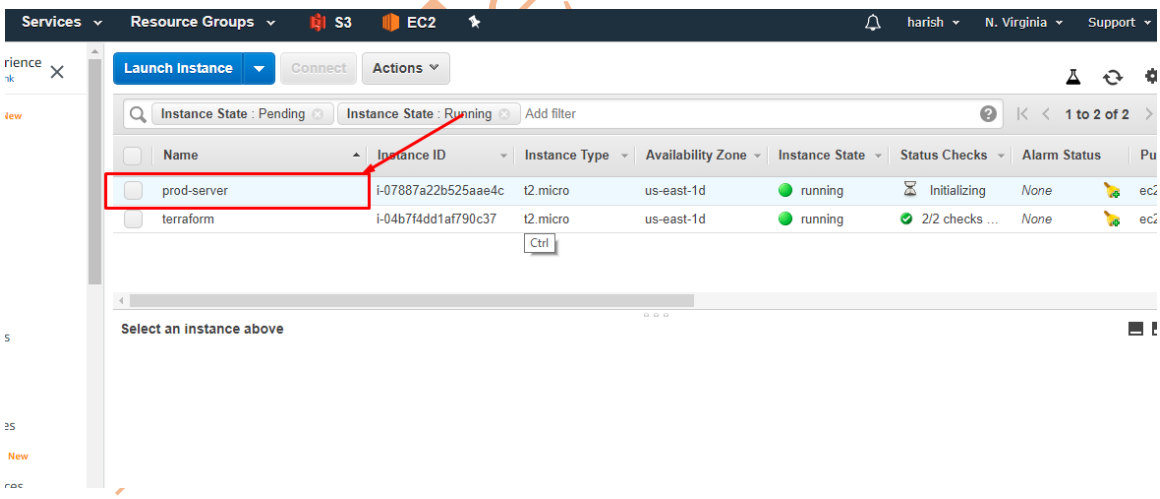
```
[root@terraform basics]# terraform apply -auto-approve -var env=dev
aws_instance.myec2: Creating...
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
```

```
aws_instance.myec2: Still creating... (30s elapsed)
aws_instance.myec2: Creation complete after 32s (ID: i-0dfed61a99bfd4e38)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns_harish = ec2-3-86-211-225.compute-1.amazonaws.com,us-east-1c
[root@terraform basics]#
```



Prod Testing:

```
[root@terraform basics]#
[root@terraform basics]# terraform apply -auto-approve -var env=prod
aws_instance.myec2: Refreshing state... (ID: i-0dfed61a99bfd4e38)
aws_instance.myec2: Destroying... (ID: i-0dfed61a99bfd4e38)
aws_instance.myec2: Still destroying... (ID: i-0dfed61a99bfd4e38, 10s elapsed)
aws_instance.myec2: Still destroying... (ID: i-0dfed61a99bfd4e38, 20s elapsed)
aws_instance.myec2: Destruction complete after 29s
aws_instance.myec2: Creating...
```



Note: Only one server will be up at a time because of tf state file

7. Working with Workspace

Now let's do the below to overcome our issue with maps & lookups

```
[root@terraform basics]# pwd
/root/basics
[root@terraform basics]# ls -l
total 28
-rw-r--r-- 1 root root  268 Sep 19 12:00 main.tf
-rw-r--r-- 1 root root   65 Sep 19 11:12 myvars.tfvars
-rw-r--r-- 1 root root  144 Sep 19 11:39 outputs.tf
-rw-r--r-- 1 root root  319 Sep 19 13:07 terraform.tfstate
-rw-r--r-- 1 root root 4765 Sep 19 13:07 terraform.tfstate.backup
-rw-r--r-- 1 root root  271 Sep 19 11:53 variables.tf
```

Listing workspace :

```
[root@terraform basics]# terraform workspace list
* default
```

Creating Workspace:

```
[root@terraform basics]# terraform workspace new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate the
```

List & Checking the current workspace:

```
for this configuration.
[root@terraform basics]# terraform workspace list
  default
* dev

[root@terraform basics]# terraform workspace show
dev
[root@terraform basics]#
```

Working with Dev workspace:

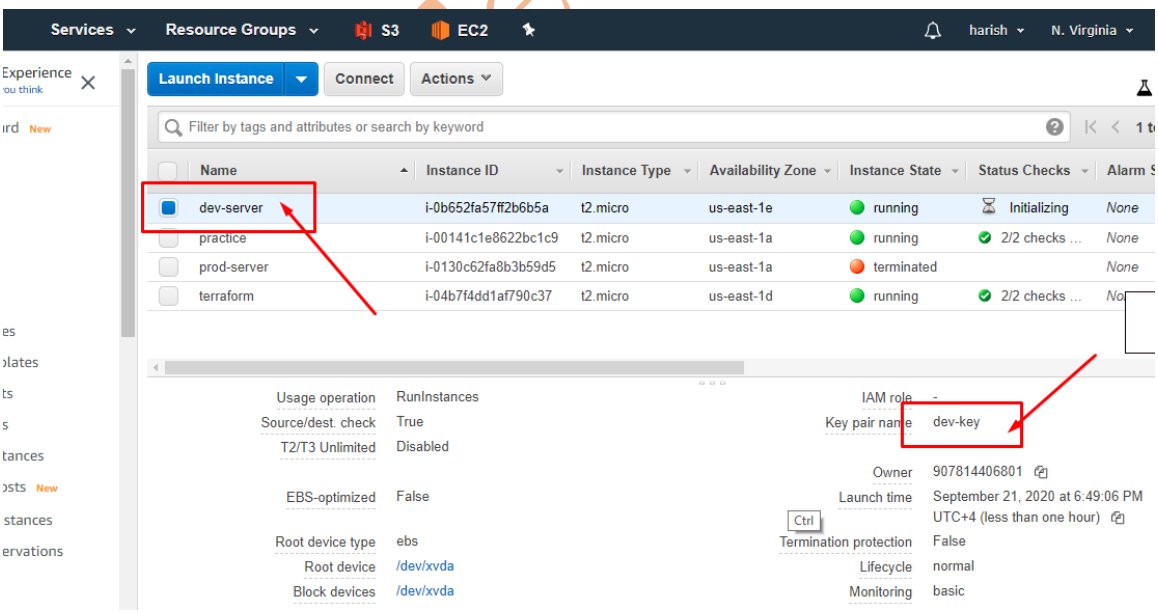
```
[root@terraform basics]# terraform workspace show
dev
[root@terraform basics]# ls -l
total 28
-rw-r--r-- 1 root root  268 Sep 19 12:00 main.tf
-rw-r--r-- 1 root root   65 Sep 19 11:12 myvars.tfvars
-rw-r--r-- 1 root root  144 Sep 19 11:39 outputs.tf
-rw-r--r-- 1 root root  319 Sep 19 13:07 terraform.tfstate
-rw-r--r-- 1 root root 4765 Sep 19 13:07 terraform.tfstate.backup
drwxr-xr-x 3 root root   17 Sep 21 14:44 terraform.tfstate.d
-rw-r--r-- 1 root root  271 Sep 19 11:53 variables.tf
[root@terraform basics]# ls -l terraform.tfstate.d/dev/
total 0
[root@terraform basics]#
```

```
[root@terraform basics]# terraform show
No state.
[root@terraform basics]# terraform apply -auto-approve -var env=dev
aws_instance.myec2: Creating...
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
```

```
  ipv6_addresses.#:              "" => "<computed>"
  key_name:                      "" => "dev-key"
  metadata_options.#:            "" => "<computed>"
  network_interface.#:           "" => "<computed>"
  network_interface_id:          "" => "<computed>"
  outpost_arn:                   "" => "<computed>"
  password_data:                 "" => "<computed>"
  placement_group:               "" => "<computed>"
  primary_network_interface_id:  "" => "<computed>"
  private_dns:                   "" => "<computed>"
  private_ip:                    "" => "<computed>"
  public_dns:                    "" => "<computed>"
  public_ip:                     "" => "<computed>"
  root_block_device.#:           "" => "<computed>"
  security_groups.#:             "" => "<computed>"
  source_dest_check:             "" => "true"
  subnet_id:                     "" => "<computed>"
  tags.%:                        "" => "1"
  tags.Name:                     "" => "dev-server"
  tenancy:                       "" => "<computed>"
  volume_tags.%:                 "" => "<computed>"
  vpc_security_group_ids.#:      "" => "<computed>"
aws_instance.myec2: Still creating... (10s elapsed)
aws_instance.myec2: Still creating... (20s elapsed)
```

```
aws_instance.myec2: Creation complete after 32s (ID: i-0b652fa57ff2b6b5a)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns_harish = ec2-34-204-101-93.compute-1.amazonaws.com,us-east-1e
[root@terraform basics]#
```

```
[root@terraform basics]#
[root@terraform basics]# terraform show
aws_instance.myec2:
  id = i-0b652fa57ff2b6b5a
  ami = ami-0c94855ba95c71c99
  arn = arn:aws:ec2:us-east-1:907814406801:instance/i-0b652fa57ff2b6b5a
  associate_public_ip_address = true
  availability_zone = us-east-1e
  cpu_core_count = 1
  cpu_threads_per_core = 1
  credit_specification.# = 1
  credit_specification.0.cpu_credits = standard
  disable_api_termination = false
  ebs_block_device.# = 0
  ebs_optimized = false
```



```
[root@terraform basics]# cat ./terraform.tfstate
{
    "version": 3,
    "terraform_version": "0.11.13",
    "serial": 20,
    "lineage": "1cb8d07e-9521-c777-bab1-1f12d519f66c",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {},
            "depends_on": []
        }
    ]
}
```

default workspace
no data

But if you check in your dev workspace you will find the new tf state file like below.

```
}
[root@terraform basics]# terraform workspace show
dev
[root@terraform basics]# ls -l terraform.tfstate.d/dev/terraform.tfstate
-rw-r--r-- 1 root root 4766 Sep 21 14:49 terraform.tfstate.d/dev/terraform.tfstate
[root@terraform basics]# more terraform.tfstate.d/dev/terraform.tfstate
{
    "version": 3,
    "terraform_version": "0.11.13",
    "serial": 1,
    "lineage": "89a3d2e6-175d-4beb-bb34-041fc9aa5196",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {
                "mypub_dns_harish": {
                    "sensitive": false,
```

Let's work on prod workspace:

```
[root@terraform basics]# terraform workspace list
  default
* dev

[root@terraform basics]# terraform workspace  new  prod
Created and switched to workspace "prod"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
[root@terraform basics]# terraform workspace show
prod
[root@terraform basics]# ls -l terraform.tfstate.d/
dev/  prod/
[root@terraform basics]# ls -l terraform.tfstate.d/prod/
total 0
[root@terraform basics]# ls -l terraform.tfstate.d/dev/
total 8
-rw-r--r-- 1 root root 4766 Sep 21 14:49 terraform.tfstate
[root@terraform basics]#
```

prod dir with no statefile [As of now]

dev tfstate file

```
-rw-r--r-- 1 root root 4766 Sep 21 14:49 terraform.tfstate
[root@terraform basics]# terraform apply -auto-approve -var env=prod
aws_instance.myec2: Creating
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
  get_password_data:            "" => "false"
  host_id:                      "" => "<computed>"
  instance_state:               "" => "<computed>"
  instance_type:                "" => "t2.micro"
```

```
  get_password_data:                "" => "false"
  host_id:                          "" => "<computed>"
  instance_state:                   "" => "<computed>"
  instance_type:                    "" => "t2.micro"
  ipv6_address_count:               "" => "<computed>"
  ipv6_addresses.#:                 "" => "<computed>"
  key_name:                         "" => "prod-key"
  metadata_options.#:               "" => "<computed>"
  network_interface.#:              "" => "<computed>"
  network_interface_id:             "" => "<computed>"
  outpost_arn:                      "" => "<computed>"
  password_data:                    "" => "<computed>"
  placement_group:                  "" => "<computed>"
  primary_network_interface_id:     "" => "<computed>"
  private_dns:                      "" => "<computed>"
  private_ip:                       "" => "<computed>"
  public_dns:                       "" => "<computed>"
  public_ip:                        "" => "<computed>"
  root_block_device.#:              "" => "<computed>"
  security_groups.#:                "" => "<computed>"
  source_dest_check:                "" => "true"
  subnet_id:                        "" => "<computed>"
  tags.%:                           "" => "1"
  tags.Name:                        "" => "prod-server"
  tenancy:                          "" => "<computed>"
  volume_tags.%:                    "" => "<computed>"
  vpc_security_group_ids.#:         "" => "<computed>"
aws_instance.myec2: Still creating... (10s elapsed)
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

mypub_dns_harish = ec2-54-84-81-75.compute-1.amazonaws.com,us-east-1a
[root@terraform basics]#
```

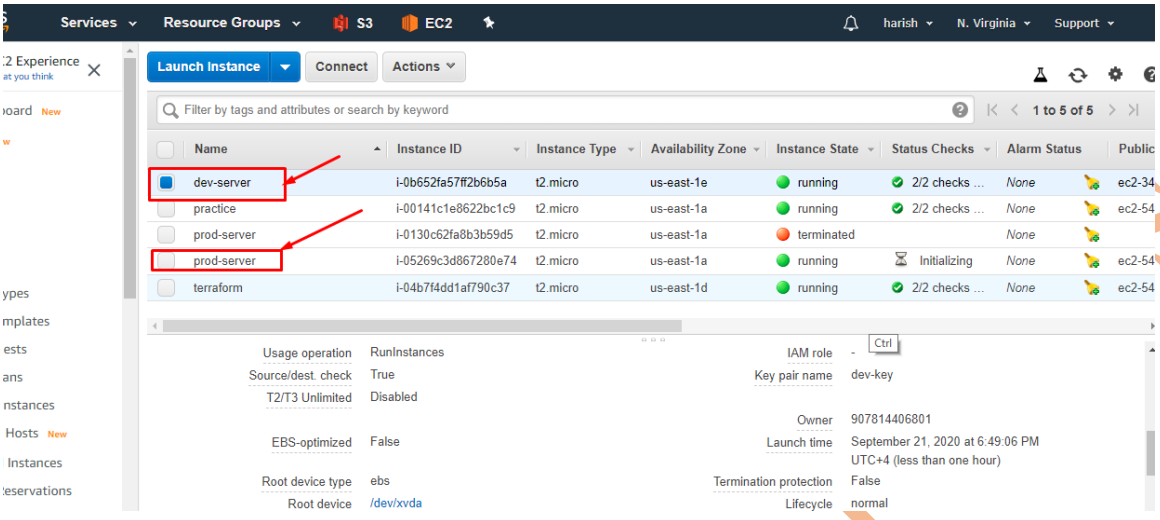
Now let's verify our prod tfstate file:

```
ypub_dns_harish = ec2-54-84-81-75.compute-1.amazonaws.com,us-east-1a
root@terraform basics]# ls -l terraform.tfstate.d/prod/terraform.tfstate
rw-r--r-- 1 root root 4764 Sep 21 14:58 terraform.tfstate.d/prod/terraform.tfstate
root@terraform basics]# ls -l terraform.tfstate.d/dev/terraform.tfstate
rw-r--r-- 1 root root 4766 Sep 21 14:49 terraform.tfstate.d/dev/terraform.tfstate
root@terraform basics]# ls -l terraform.tfstate
rw-r--r-- 1 root root 319 Sep 19 13:07 terraform.tfstate
root@terraform basics]#
```

prod

dev

default



Now if you observe we can have both the environments from the same code only change we applied is just adding new workspace for the same directory.

Switching between workspaces:

```
[root@terraform basics]# terraform workspace list
  default
  dev
* prod

[root@terraform basics]# terraform workspace select default
Switched to workspace "default".
[root@terraform basics]# terraform workspace list
* default
  dev
  prod

[root@terraform basics]# terraform workspace select dev
Switched to workspace "dev".
[root@terraform basics]# terraform workspace list
  default
* dev
  prod

[root@terraform basics]# terraform workspace show
dev
[root@terraform basics]#
```

Destroy the environments:

```
[root@terraform basics]# terraform workspace show
dev
[root@terraform basics]# terraform destroy -auto-approve -var env=dev
aws_instance.myec2: Refreshing state... (ID: i-0b652fa57ff2b6b5a)
aws_instance.myec2: Destroying... (ID: i-0b652fa57ff2b6b5a)
aws_instance.myec2: Still destroying... (ID: i-0b652fa57ff2b6b5a, 10s elapsed)
aws_instance.myec2: Still destroying... (ID: i-0b652fa57ff2b6b5a, 20s elapsed)
aws_instance.myec2: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
[root@terraform basics]# terraform workspace select prod
Switched to workspace "prod".
[root@terraform basics]# terraform workspace show
prod
[root@terraform basics]# terraform destroy -auto-approve -var env=prod
aws_instance.myec2: Refreshing state... (ID: i-05269c3d867280e74)
aws_instance.myec2: Destroying... (ID: i-05269c3d867280e74)
aws_instance.myec2: Still destroying... (ID: i-05269c3d867280e74, 10s elapsed)
aws_instance.myec2: Still destroying... (ID: i-05269c3d867280e74, 20s elapsed)
aws_instance.myec2: Destruction complete after 29s

Destroy complete! Resources: 1 destroyed.
[root@terraform basics]#
```

null_resource:

```
[root@terraform basics]# ls -l
total 28
-rw-r--r-- 1 root root  268 Sep 19 12:00 main.tf
-rw-r--r-- 1 root root   65 Sep 19 11:12 myvars.tfvars
-rw-r--r-- 1 root root  144 Sep 19 11:39 outputs.tf
-rw-r--r-- 1 root root  319 Sep 19 13:07 terraform.tfstate
-rw-r--r-- 1 root root 4765 Sep 19 13:07 terraform.tfstate.backup
drwxr-xr-x 4 root root   29 Sep 21 14:54 terraform.tfstate.d
-rw-r--r-- 1 root root  271 Sep 19 11:53 variables.tf
```

No file messages.txt

```
[root@terraform basics]# cat main.tf
provider "aws" {

region = "us-east-1"

}


resource "aws_instance" "myec2" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  key_name = "${lookup(var.my_key_name,var.env)}"
  tags = {
    Name = "${lookup(var.my_server_name,var.env)}"
  }
}

resource "null_resource" "null_id" {
  provisioner "local-exec" {
    command = "echo ${aws_instance.myec2.tags.Name},${aws_instance.myec2.public_ip} >> servers.txt"
  }
}

[root@terraform basics]#
```

capturing server info to a file

```
[root@terraform basics]# terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "null" (2.1.2)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
```

```
commands will detect it and remind you to do so if necessary.
[root@terraform basics]# terraform apply -auto-approve -var env=dev
aws_instance.myec2: Creating...
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
```

```
[root@terraform basics]# ls -l
total 32
-rw-r--r-- 1 root root  441 Sep 21 15:10 main.tf
-rw-r--r-- 1 root root   65 Sep 19 11:12 myvars.tfvars
-rw-r--r-- 1 root root  144 Sep 19 11:39 outputs.tf
-rw-r--r-- 1 root root   25 Sep 21 15:11 servers.txt
-rw-r--r-- 1 root root 5370 Sep 21 15:11 terraform.tfstate
-rw-r--r-- 1 root root  319 Sep 21 15:11 terraform.tfstate.backup
drwxr-xr-x 4 root root   29 Sep 21 14:54 terraform.tfstate.d
-rw-r--r-- 1 root root  271 Sep 19 11:53 variables.tf
[root@terraform basics]# cat servers.txt
dev-server,54.160.77.244
[root@terraform basics]# .
```

server info.

8. working with Modules

Let's create individual elements i.e, S3,VPC,EC2

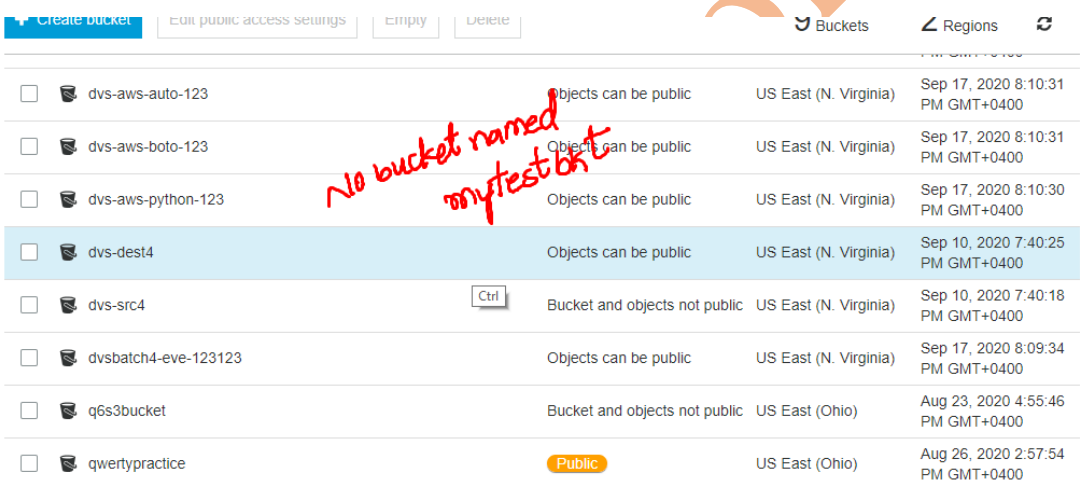
Creation of S3:

```
[root@terraform ~]# mkdir moduels
[root@terraform ~]# cd moduels/
[root@terraform moduels]# mkdir s3
[root@terraform moduels]# cd s3/
[root@terraform s3]# pwd
/root/moduels/s3
[root@terraform s3]# ls -l
total 0
[root@terraform s3]# vi main.tf
[root@terraform s3]# cat main.tf
provider "aws" {
        region = "us-east-1"
}

resource "random_id" "myrandid" {
  byte_length = 2
}

/*variable*/

variable "bucket_name" {
```

DVS Technologies

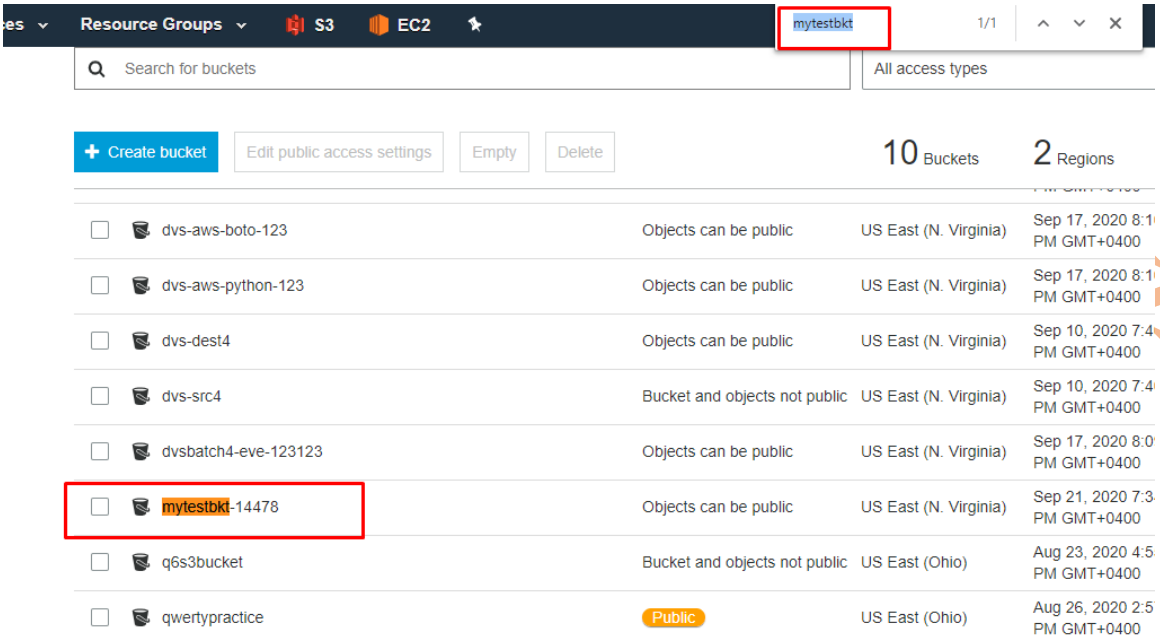
```
}

resource "aws_s3_bucket" "mys3bucket" {
  bucket = "${var.bucket_name}-${random_id.myrandid.dec}"
  tags = {
    Name        = "${var.bucket_name}-${random_id.myrandid.dec}"

  }
}
```

Before applying below:



terraform init .
terraform apply -auto-approve -var bucket_name="mytestbkt"

Post changes:



After Alignment my code look like below:

```
[root@terraform s3]# cat variables.tf
variable "bucket_name" {}
[root@terraform s3]# cat main.tf
provider "aws" {
        region = "us-east-1"
}

resource "random_id" "myrandid" {
  byte_length = 2
}

resource "aws_s3_bucket" "mys3bucket" {
  bucket = "${var.bucket_name}-${random_id.myrandid.dec}"
  tags = {
    Name        = "${var.bucket_name}-${random_id.myrandid.dec}"

  }
}
[root@terraform s3]# vi myvars.tfvars
[root@terraform s3]# cat myvars.tfvars
bucket_name="mytestbkt"
[root@terraform s3]# terraform apply -auto-approve -var-file=myvars.tfvars
random_id.myrandid: Refreshing state... (ID: 014)
aws_s3_bucket.mys3bucket: Refreshing state... (ID: mytestbkt-14478)

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[root@terraform s3]#
```

Total three file main.tf, outputs.tf, myvars.tfvars

Let's work with our Ec2:

```
[root@terraform ec2]# touch {variables.tf,main.tf,myvars.tfvars}
[root@terraform ec2]# vi myvars.tfvars
[root@terraform ec2]# cat myvars.tfvars
ec2_ami="ami-0c94855ba95c71c99"
ec2_keypair="shan_harish_nvirginia"
ec2_type="t2.micro"
region="us-east-1"
ec2_name="mytestserver"
[root@terraform ec2]# vi variables.tf
[root@terraform ec2]# cat variables.tf
variable "ec2_name" {}
variable "ec2_keypair" {}
variable "ec2_ami" {}
variable "ec2_type" {}
variable "region" {}
[root@terraform ec2]# vi main.tf
[root@terraform ec2]# cat main.tf
provider "aws" {

        region = "${var.region}"
}


resource "aws_instance" "myec2" {
  ami           = "${var.ec2_ami}"
  instance_type = "${var.ec2_type}"
  key_name = "${var.ec2_keypair}"
  tags = {
    Name = "${var.ec2_name}"
  }
}
[root@terraform ec2]#
```

[root@terraform ec2]# touch {variables.tf,main.tf,myvars.tfvars}

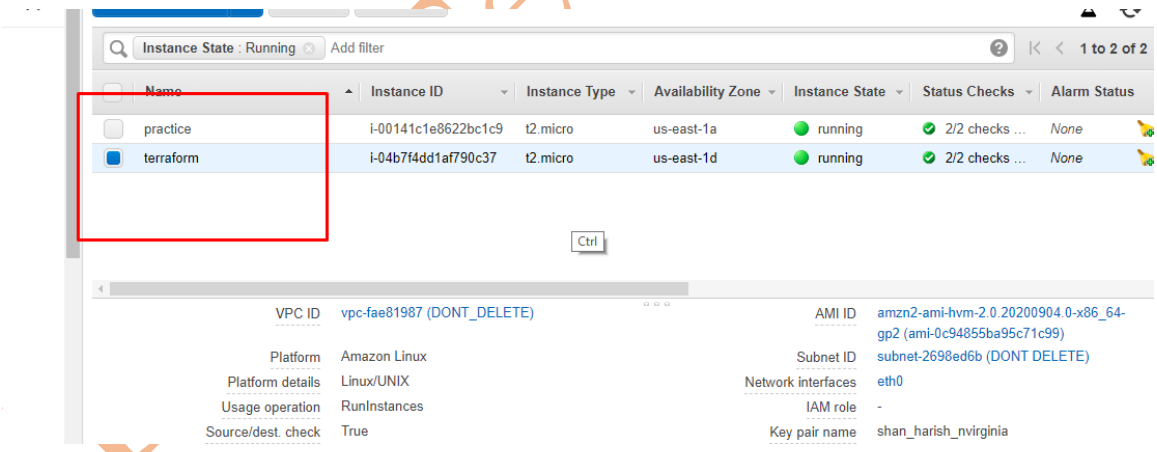
[root@terraform ec2]# vi myvars.tfvars

[root@terraform ec2]# cat myvars.tfvars

ec2_ami="ami-0c94855ba95c71c99"
ec2_keypair="shan_harish_nvirginia"
ec2_type="t2.micro"
region="us-east-1"
ec2_name="mytestserver"
[root@terraform ec2]# vi variables.tf
[root@terraform ec2]# cat variables.tf
variable "ec2_name" {}
variable "ec2_keypair" {}

```
variable "ec2_ami" {}
variable "ec2_type" {}
variable "region" {}
[root@terraform ec2]# vi main.tf
[root@terraform ec2]# cat main.tf
provider "aws" {

      region = "${var.region}"
}


resource "aws_instance" "myec2" {
  ami           = "${var.ec2_ami}"
  instance_type = "${var.ec2_type}"
  key_name = "${var.ec2_keypair}"
  tags = {
    Name = "${var.ec2_name}"
  }
}
[root@terraform ec2]#
```

Before:

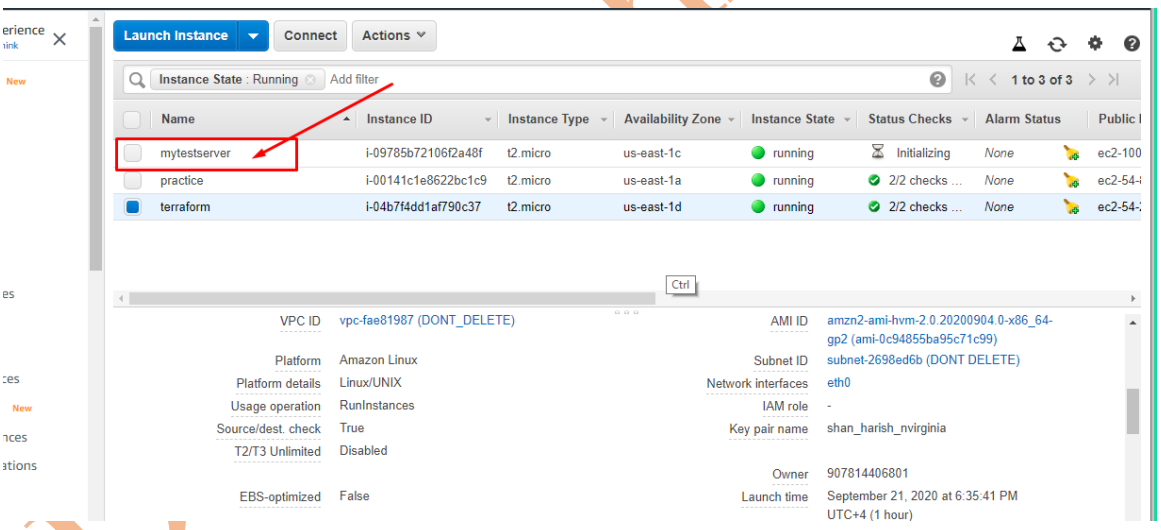


Apply the changes like below:

```
[root@terraform ec2]# terraform init .

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "aws" (2.70.0)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.
```

```
commands will detect it and remind you to do so if necessary.
[root@terraform ec2]#
[root@terraform ec2]# terraform apply -auto-approve -var-file=myvars.tfvars
aws_instance.myec2: Creating...
  ami:                          "" => "ami-0c94855ba95c71c99"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
  get_password_data:            "" => "false"
  host_id:                      "" => "<computed>"
  instance_state:               "" => "<computed>"
  instance_type:                "" => "t2.micro"
```

After Changes:



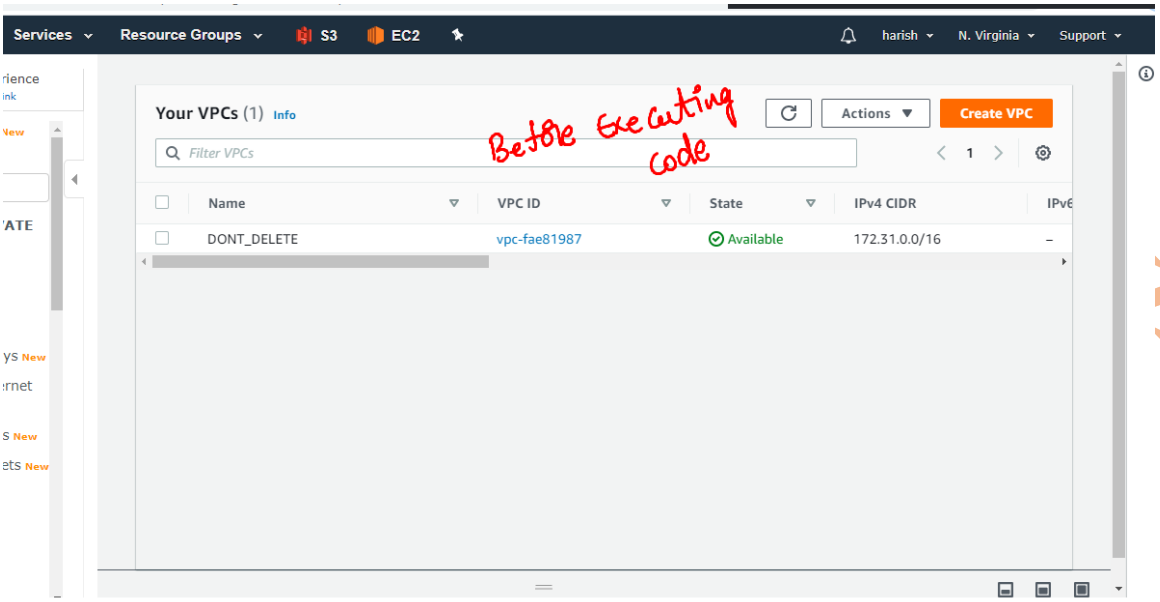
Destroy infrastructure:

```
[root@terraform ec2]# terraform destroy -auto-approve -var-file=myvars.tfvars
aws_instance.myec2: Refresning state... (ID: i-09785b72106f2a48f)
aws_instance.myec2: Destroying... (ID: i-09785b72106f2a48f)
```

Finally let's work on VPC creation:

```
[root@terraform ~]#
[root@terraform ~]# cd moduels/
[root@terraform moduels]# ls -l
total 0
drwxr-xr-x 3 root root 137 Sep 21 16:07 ec2
drwxr-xr-x 3 root root 155 Sep 21 15:59 s3
[root@terraform moduels]# mkdir vpc
[root@terraform moduels]# cd vpc/
[root@terraform vpc]# ls -l
total 0
[root@terraform vpc]# pwd
/root/moduels/vpc
[root@terraform vpc]# ls -l
total 0
[root@terraform vpc]# cd ..
[root@terraform moduels]# pwd
/root/moduels
[root@terraform moduels]# ls -l
total 0
drwxr-xr-x 3 root root 137 Sep 21 16:07 ec2
drwxr-xr-x 3 root root 155 Sep 21 15:59 s3
drwxr-xr-x 2 root root   6 Sep 22 14:48 vpc
[root@terraform moduels]#
```

Before code execution I can see that only one vpc is existing



Let's start working with our code

Code:

[root@terraform vpc]# cat variables.tf

```
/*variable*/

variable "region" {}
variable "vpc_cidr" {}
variable "vpc_name" {}
variable "vpc_igw_name" {}
variable "vpc_route_name" {}
variable "vpc_sub_name" {}
variable "vpc_sub_cidr" {}
variable "vpc_secgrp_name" {}
```

[root@terraform vpc]# cat myvars.tfvars

```
/*myvars.tfvars*/
region="us-east-1"
vpc_name="testvpc"
vpc_cidr="10.60.0.0/16"
vpc_igw_name="testigw"
vpc_route_name="testroute"
vpc_sub_cidr="10.60.10.0/24"
```

```
vpc_sub_name="testsubnet"
vpc_secgrp_name="testsecgroup"
[root@terraform vpc]#
```

[root@terraform vpc]# cat main.tf

```
/*main.tf*/

provider "aws" {
        region = "${var.region}"
}

resource "aws_vpc" "myvpc" {
  cidr_block       = "${var.vpc_cidr}"
  tags = {
    Name = "${var.vpc_name}"
  }
}

resource "aws_internet_gateway" "myigw" {
  vpc_id = "${aws_vpc.myvpc.id}"

  tags = {
    Name = "${var.vpc_igw_name}"
  }
}

resource "aws_route_table" "myroute" {
  vpc_id = "${aws_vpc.myvpc.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.myigw.id}"
  }

  tags = {
    Name = "${var.vpc_route_name}"
  }
}

resource "aws_subnet" "mysubnet" {
  vpc_id     = "${aws_vpc.myvpc.id}"
  cidr_block = "${var.vpc_sub_cidr}"

  tags = {
```

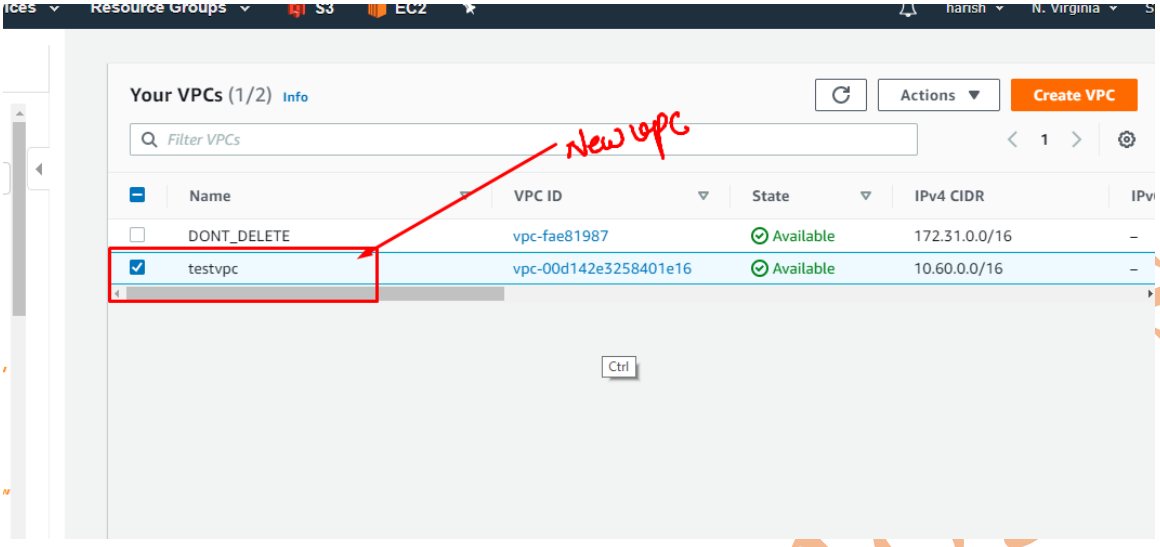
DVS Technologies

```
    Name = "${var.vpc_sub_name}"
  }
}

resource "aws_route_table_association" "myroute_association" {
  subnet_id      = "${aws_subnet.mysubnet.id}"
  route_table_id = "${aws_route_table.myroute.id}"
}


resource "aws_security_group" "mysecgroup" {
  name        = "${var.vpc_secgrp_name}"
  description = "Allow TLS inbound traffic"
  vpc_id      = "${aws_vpc.myvpc.id}"

  ingress {
    description = "mysecgroup"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.vpc_secgrp_name}"
  }
}
```

Execution:

```
terraform init .
terraform apply -auto-approve -var-file=myvars.tfvars
```

DVS Technologies




```
[root@terraform vpc]#
[root@terraform vpc]# terraform destroy -auto-approve -var-file=myvars.tfvars
aws_vpc.myvpc: Refreshing state... (ID: vpc-00d142e3258401e16)
aws_internet_gateway.myigw: Refreshing state... (ID: igw-0450263c8a733c7d5)
aws_subnet.mysubnet: Refreshing state... (ID: subnet-00c4ef65479558bf8)
aws_security_group.mysecgroup: Refreshing state... (ID: sg-01b3f695750bd7a2e)
aws_route_table.myroute: Refreshing state... (ID: rtb-05ff0d02a1e437218)
aws_route_table_association.myroute_association: Refreshing state... (ID: rtbassoc-0a2414acee9ea6f02)
aws_route_table_association.myroute_association: Destroying... (ID: rtbassoc-0a2414acee9ea6f02)
aws_security_group.mysecgroup: Destroying... (ID: sg-01b3f695750bd7a2e)
aws_route_table_association.myroute_association: Destruction complete after 1s
aws_subnet.mysubnet: Destroying... (ID: subnet-00c4ef65479558bf8)
aws_route_table.myroute: Destroying... (ID: rtb-05ff0d02a1e437218)
aws_security_group.mysecgroup: Destruction complete after 1s
aws_route_table.myroute: Destruction complete after 0s
aws_internet_gateway.myigw: Destroying... (ID: igw-0450263c8a733c7d5)
aws_subnet.mysubnet: Destruction complete after 0s
aws_internet_gateway.myigw: Still destroying... (ID: igw-0450263c8a733c7d5, 10s elapsed)
aws_internet_gateway.myigw: Destruction complete after 10s
aws_vpc.myvpc: Destroying... (ID: vpc-00d142e3258401e16)
aws_vpc.myvpc: Destruction complete after 1s

Destroy complete! Resources: 6 destroyed.
[root@terraform vpc]#
```

Code reusability:

Let's create Ec2 using the module

#mkdir user1
#cd user1
[root@terraform user1]# pwd
/root/user1
[root@terraform user1]# cat main.tf
module "myserver" {

source = "/root/moduels/ec2/"

```
ec2_ami = "ami-0c94855ba95c71c99"
ec2_keypair = "dev-key"
ec2_type = "t2.micro"
region = "us-east-1"
ec2_name = "dvsserver1"

}
```

terraform init .
terraform apply -auto-approve

Let's append our vpc:

Variabels to pass for vpc module:

```
[root@terraform user1]# cat /root/moduels/vpc/myvars.tfvars
/*myvars.tfvars*/
region="us-east-1"
vpc_name="testvpc"
vpc_cidr="10.60.0.0/16"
vpc_igw_name="testigw"
vpc_route_name="testroute"
vpc_sub_cidr="10.60.10.0/24"
vpc_sub_name="testsubnet"
vpc_secgrp_name="testsecgroup"
```

Ec2 & VPC module as part of main.tf file:

```
[root@terraform user1]# cat main.tf
module "myserver" {

source = "/root/moduels/ec2/"
ec2_ami = "ami-0c94855ba95c71c99"
ec2_keypair = "dev-key"
ec2_type = "t2.micro"
region = "us-east-1"
ec2_name = "dvsserver1"

}

module "network" {
source = "/root/moduels/vpc/"
region="us-east-1"
```


DVS Technologies

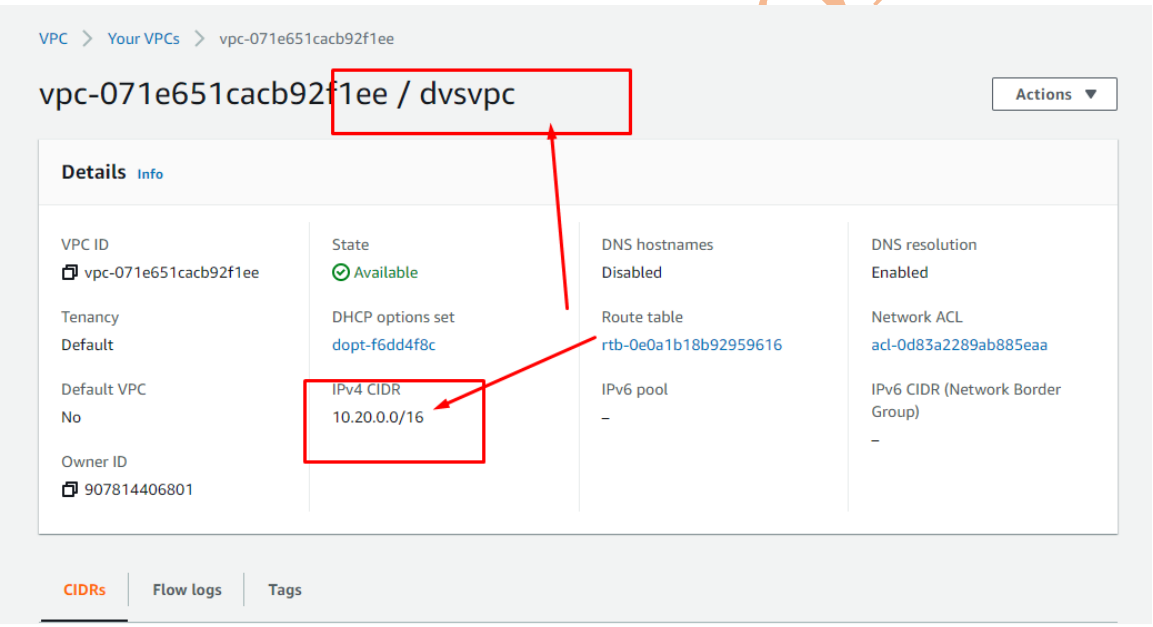
```
vpc_name="dvsvpc"
vpc_cidr="10.20.0.0/16"
vpc_igw_name="dvsigw"
vpc_route_name="dvsroute"
vpc_sub_cidr="10.20.10.0/24"
vpc_sub_name="dvssubnet"
vpc_secgrp_name="dvssecgroup"

}
```

Execution:

```
[root@terraform user1]# terraform init .
Initializing modules...
- module.myserver
- module.network
  Getting source "/root/moduels/vpc/"

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.70"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[root@terraform user1]#
```

```
[root@terraform user1]#
[root@terraform user1]# terraform apply -auto-approve
aws_instance.myec2: Refresning state... (ID: i-0d80f0466fe5cb602)
module.network.aws_vpc.myvpc: Creating...
  arn:                                  "" => "<computed>"
  assign_generated_ipv6_cidr_block:     "" => "false"
  cidr_block:                           "" => "10.20.0.0/16"
  default_network_acl_id:               "" => "<computed>"
  default_route_table_id:               "" => "<computed>"
  default_security_group_id:            "" => "<computed>"
  dhcp_options_id:                      "" => "<computed>"
  enable_classiclink:                   "" => "<computed>"
  enable_classiclink_dns_support:       "" => "<computed>"
  enable_dns_hostnames:                 "" => "<computed>"
  enable_dns_support:                   "" => "true"
  instance_tenancy:                     "" => "default"
  ipv6_association_id:                  "" => "<computed>"
  ipv6_cidr_block:                      "" => "<computed>"
  main_route_table_id:                  "" => "<computed>"
  owner_id:                             "" => "<computed>"
```

Final Verification:



9. Profiles - Accessing Multiple Regions

Please do the below for different profiles

Default Profile:

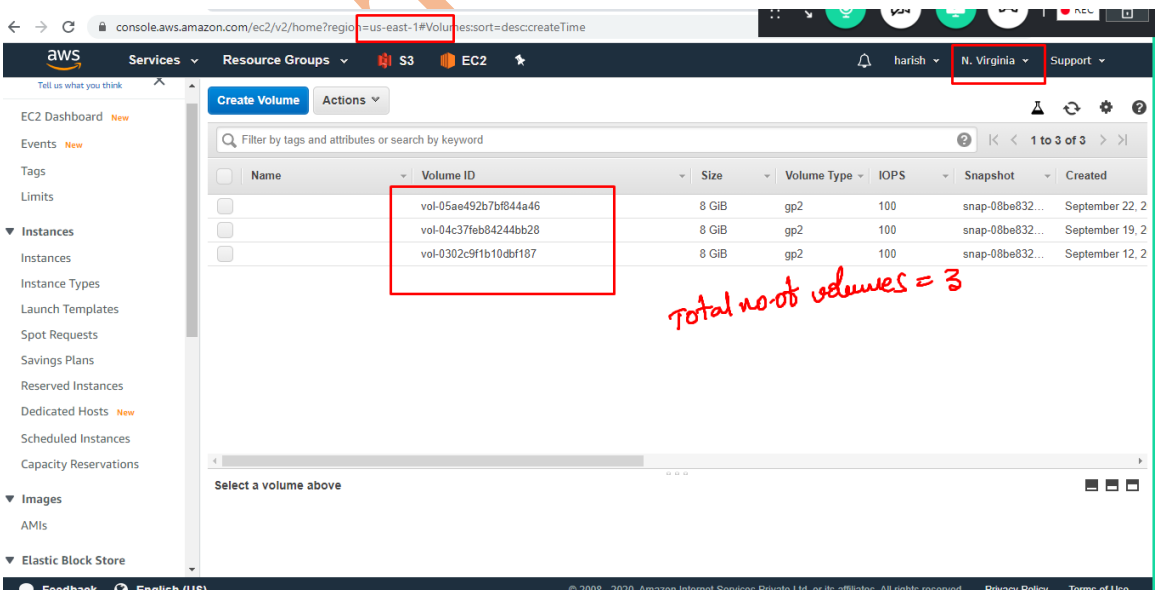
```
[root@terraform user1]# cat ~/.aws/credentials
[default]
aws_access_key_id = AKIA5GXPURKI[redacted]N
aws_secret_access_key = EVmPiSusEQe/kujPvneAXQc8R3Lt[redacted]
[root@terraform user1]# cat ~/.aws/config
[default]
output = json
region = us-east-1
```

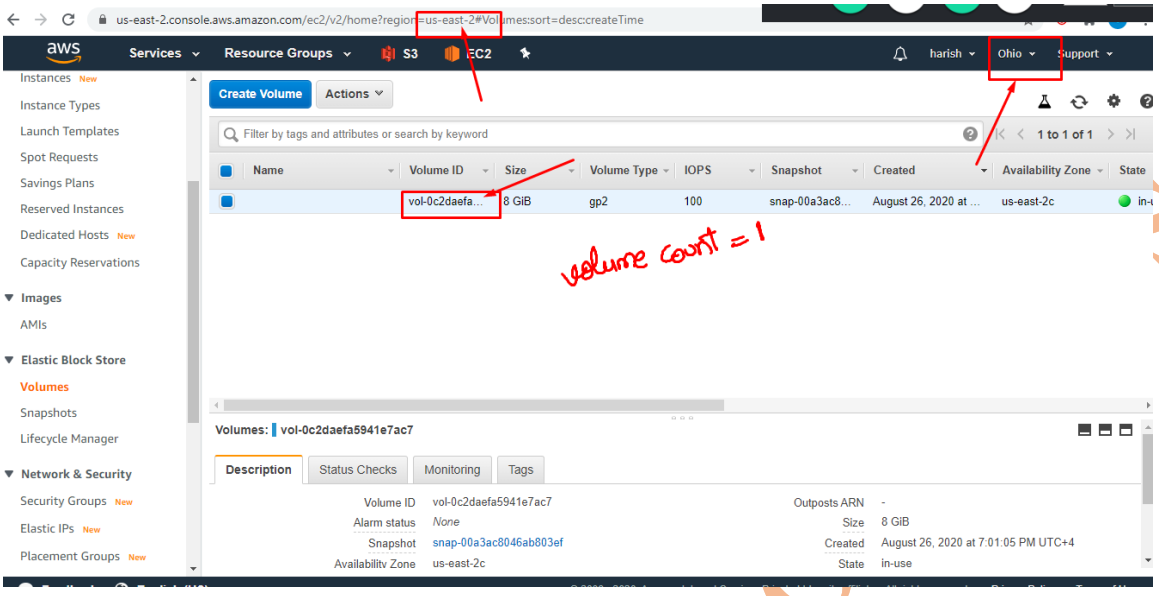
Let's add a new profile i.e us-east-2

```
[root@terraform user1]# aws configure --profile us-east-2
AWS Access Key ID [None]: AKIA5GXPURKISNRQ2HQN
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-2
Default output format [None]: json
[root@terraform user1]# cat ~/.aws/credentials
[default]
aws_access_key_id = AKIA5GXPURKISNRQ2HQN
aws_secret_access_key = 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
[us-east-2]
aws_access_key_id = AKIA5GXPURKISNRQ2HQN
aws_secret_access_key = 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
```

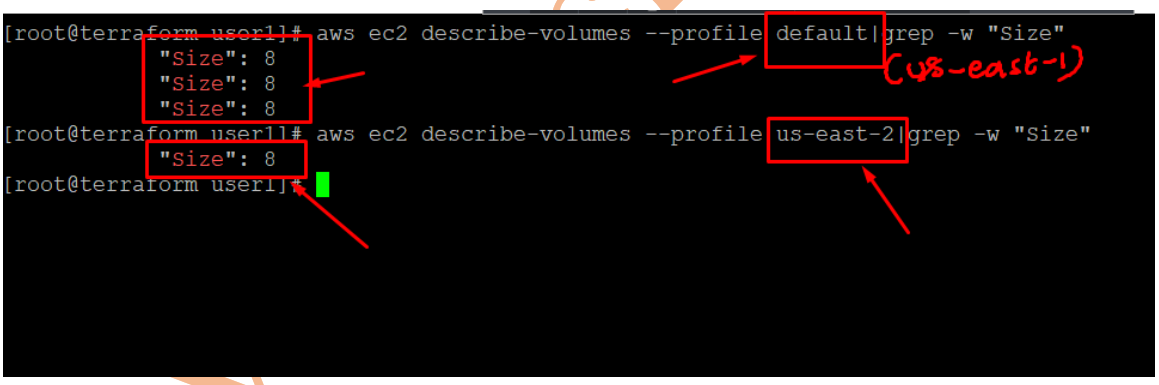
Let's test the profile in different regions:

Us-east-1 & us-east-2





Testing via CLI:



Testing via Terraform:

```
[root@terraform ec2]# vi myvars.tfvars
[root@terraform ec2]# cat myvars.tfvars
ec2_ami="ami-0c94855ba95c71c99"
ec2_keypair="shan_harish_nvirginia"
ec2_type="t2.micro"
region="us-east-2"
ec2_name="mytestserver"
[root@terraform ec2]# cat main.tf
provider "aws" {

        region = "${var.region}"
        profile = "us-east-2"
}


resource "aws_instance" "myec2" {
  ami           = "${var.ec2_ami}"
  instance_type = "${var.ec2_type}"
  key_name = "${var.ec2_keypair}"
  tags = {
    Name = "${var.ec2_name}"
  }
}
[root@terraform ec2]#
```

DVS Technologies