Terraform

1. Download Terraform - www.terraform.io/downloads.html
2. IAM Permissions
    - Add a user and give programmatic access
    - Attach administrator access policy
3. Download AWS CLI and configure through command prompt for Windows.
    - Give the user credentials and region
4. Install Atom editor
    - Install terraform plugins - language-terraform and terraform-fmt

Single line comment in terraform - // or #
Multi-line comment in terraform - /* <....> */

Initializing Terraform Project

Create a file for the provider. Here, I have created it with the name provider.tf. Specify the provider name eg. aws and the region name where it needs to deploy the resources. Create these files within a folder in a specific drive/directory in your computer. Go to the command prompt, navigate to the particular directory and run terraform init to initialize the terraform project.

provider.tf

```
1  provider "aws" {
2    region = "us-east-1"
3  }
4
```

```
C:\Users\adars>D:

D:\>cd New_Terraform

D:\New_Terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v3.37.0...
- Installed hashicorp/aws v3.37.0 (self-signed, key ID 34365D9472D7468F)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

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.

D:\New_Terraform>
```

Terraform init will download the plugins for the provider, here in our case - AWS

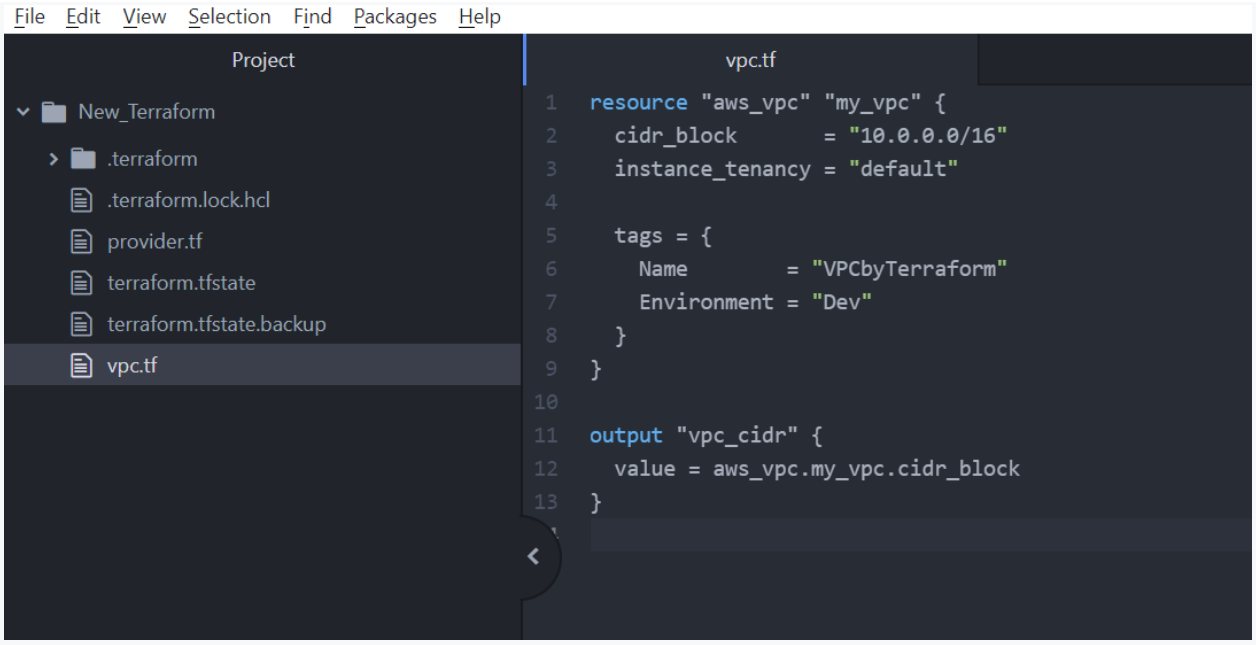
Resource-

Basic usage:

```
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}
```

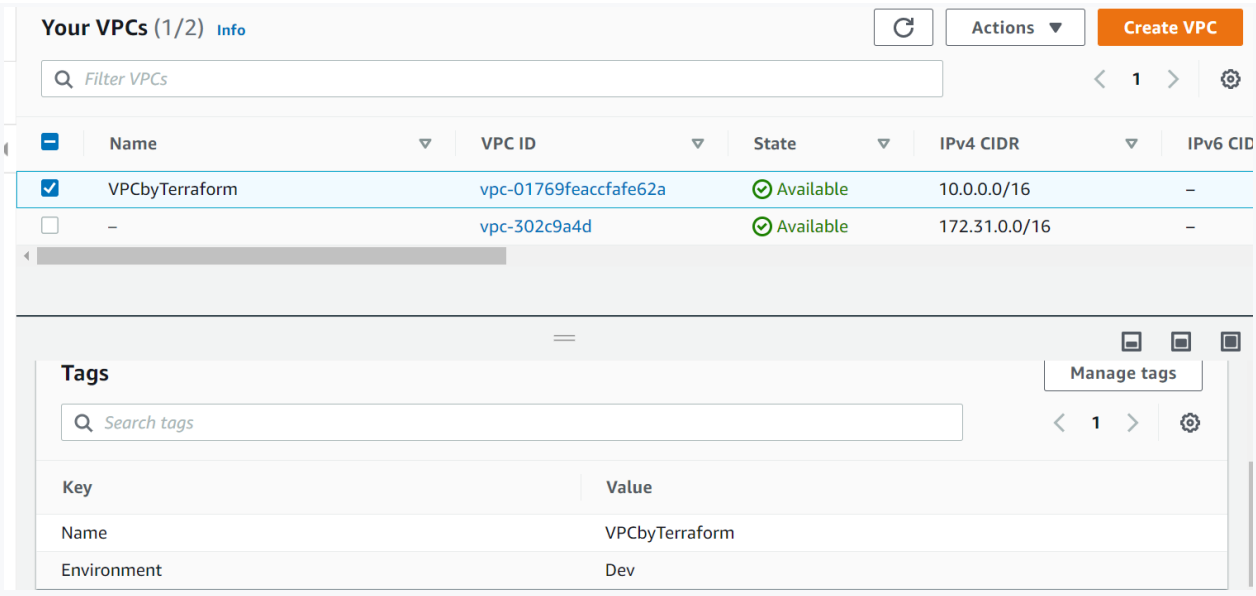
Basic usage with tags:

```
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "main"
  }
}
```



Command Prompt

```
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = (known after apply)
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Environment" = "Dev"
          + "Name"        = "VPCbyTerraform"
        }
      + tags_all                         = {
          + "Environment" = "Dev"
          + "Name"        = "VPCbyTerraform"
        }
    }

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

Changes to Outputs:
  + vpc_cidr = "10.0.0.0/16"

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_vpc.my_vpc: Creating...
aws_vpc.my_vpc: Still creating... [10s elapsed]
aws_vpc.my_vpc: Creation complete after 13s [id=vpc-01769feaccfafe62a]

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

Outputs:

vpc_cidr = "10.0.0.0/16"

D:\New_Terraform>
```

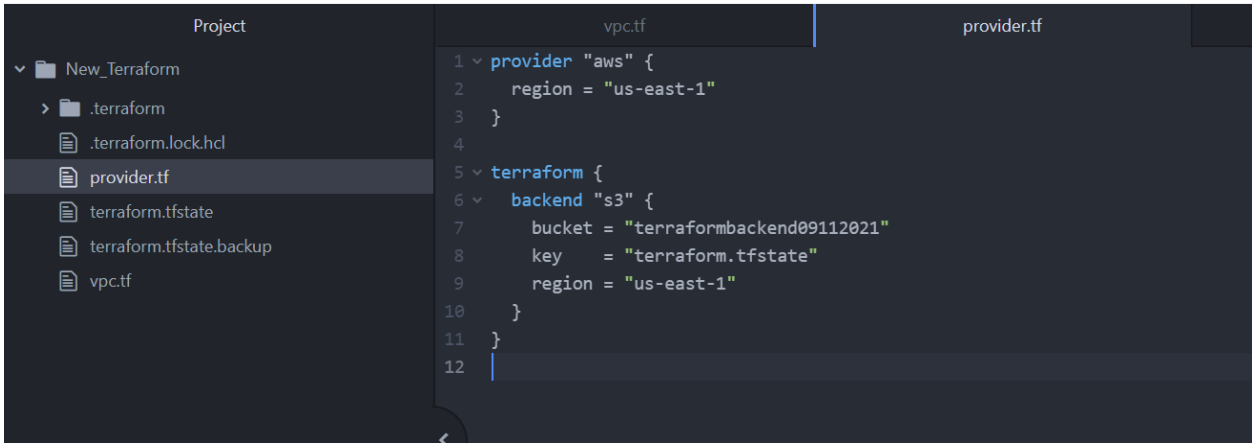


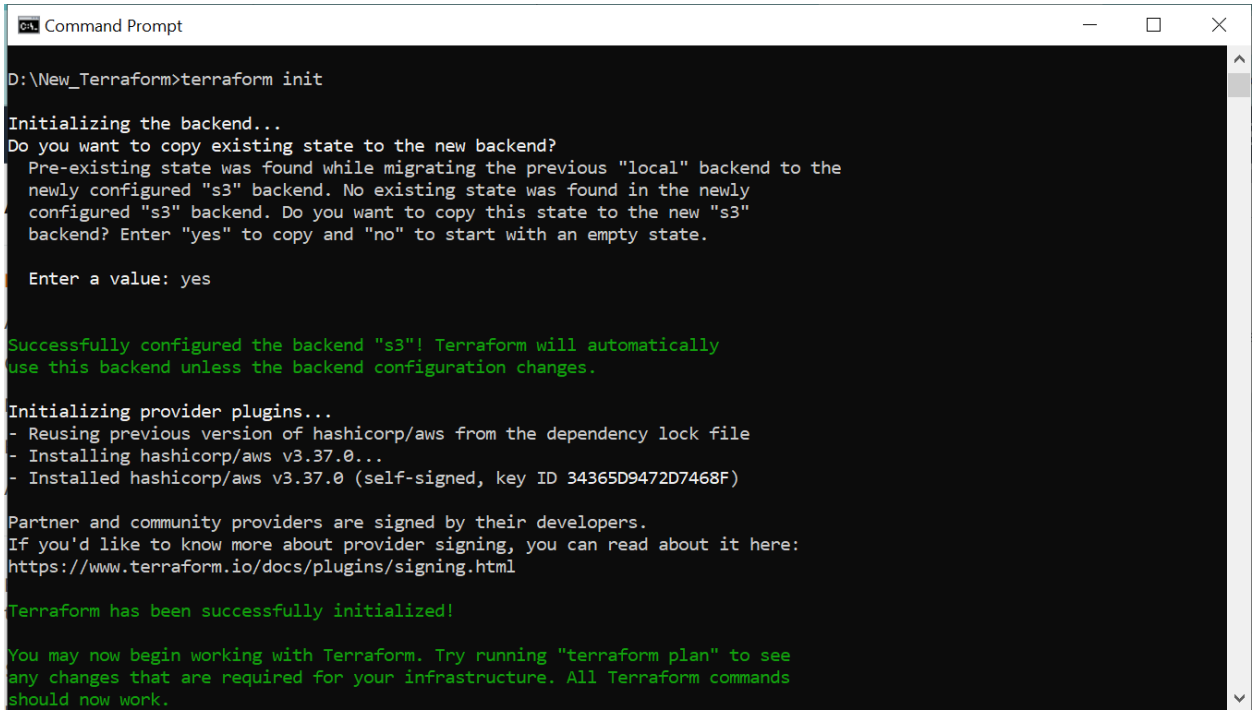
Terraform Local State File : Terraform Local state file maintains the record of all the resources created by Terraform. Therefore, whenever we create/modify/update/delete any resource terraform draws the reference from here. If in case this file is deleted, terraform loses the reference and creates the resources newly and doesn't have any control over the previous one.

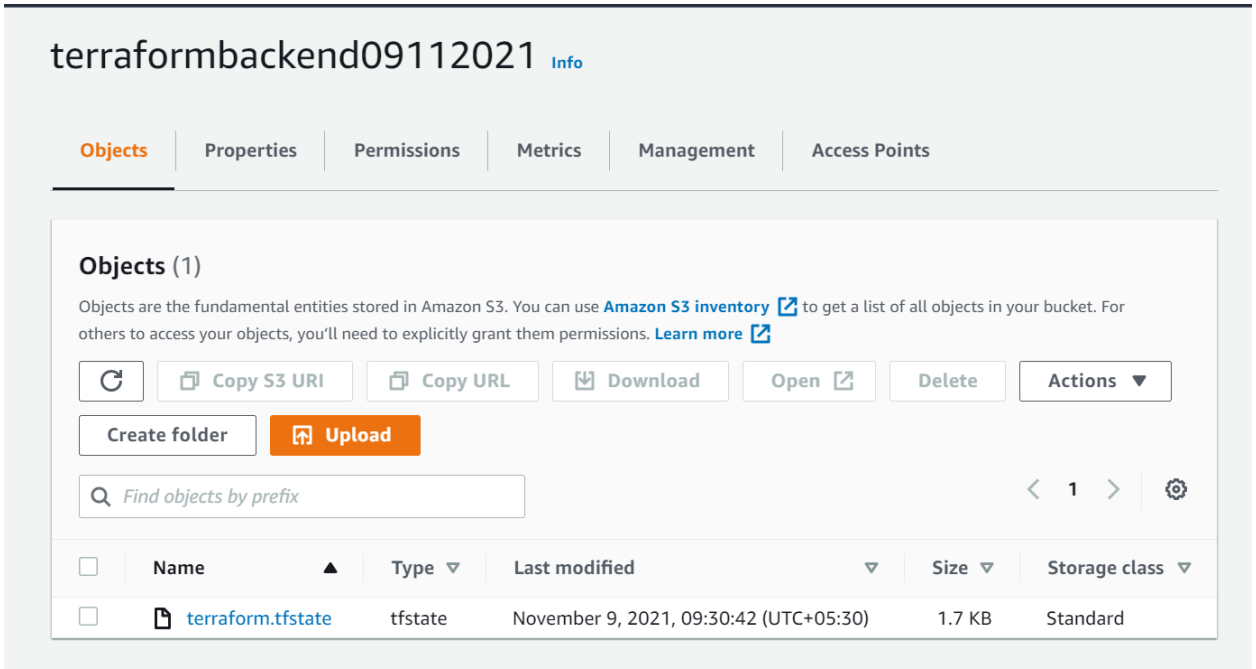
By default, this file is called " terraform. tfstate" and is stored locally in JSON format but can also store it remotely. It is created by Terraform the first time the terraform plan command is run and will use it each time it is run to compare its state with that of the target infrastructure and return the preview of the changes to be made.

Terraform Remote State File

Terraform by default stores the state file in the local. That is fine when one developer is working on the code but it is a problem when multiple developers contribute to the code in the project. Hence, we maintain the state file in S3 as a remote state file. For that we need to create an S3 bucket, maintain the below code and run terraform init.







Terraform deletes the state file from the local once the remote state file is created. It is a good practice to enable versioning for the S3 bucket as it will maintain every version of the state file and is convenient if we want to fall back to any of the previous versions.

Once we create the remote state file it is very important to lock it as multiple entries concurrently by many developers may result in inconsistency. Locking enables a lock on the code once a

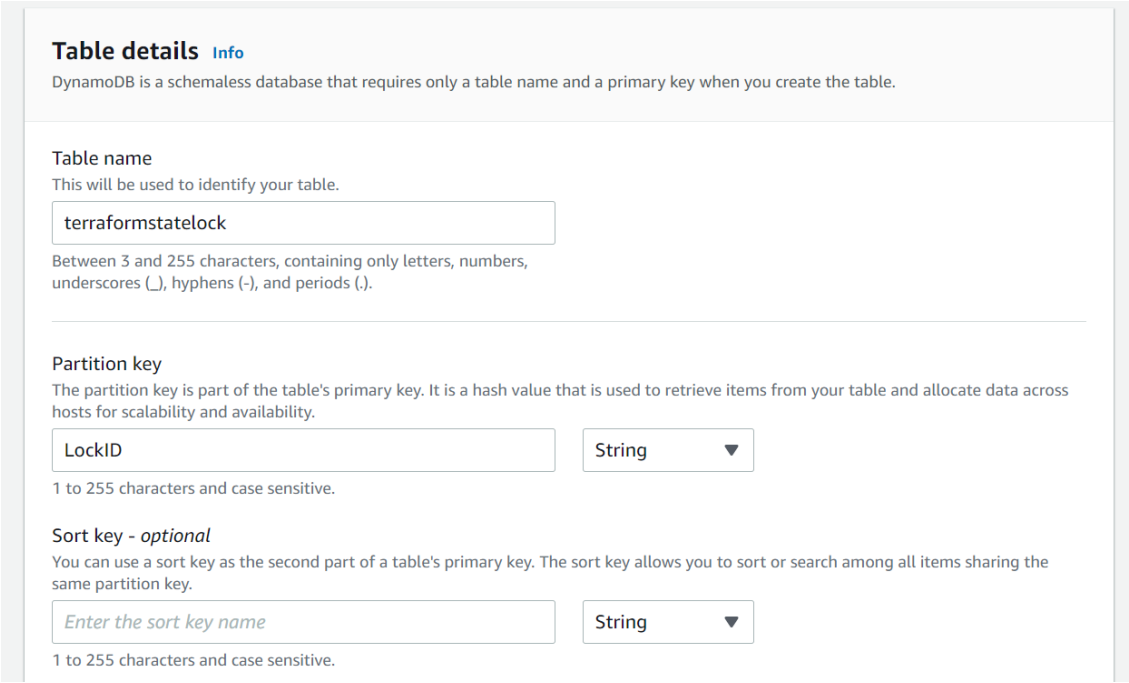
developer is making any changes and prohibits other changes by other developers during that point of time.

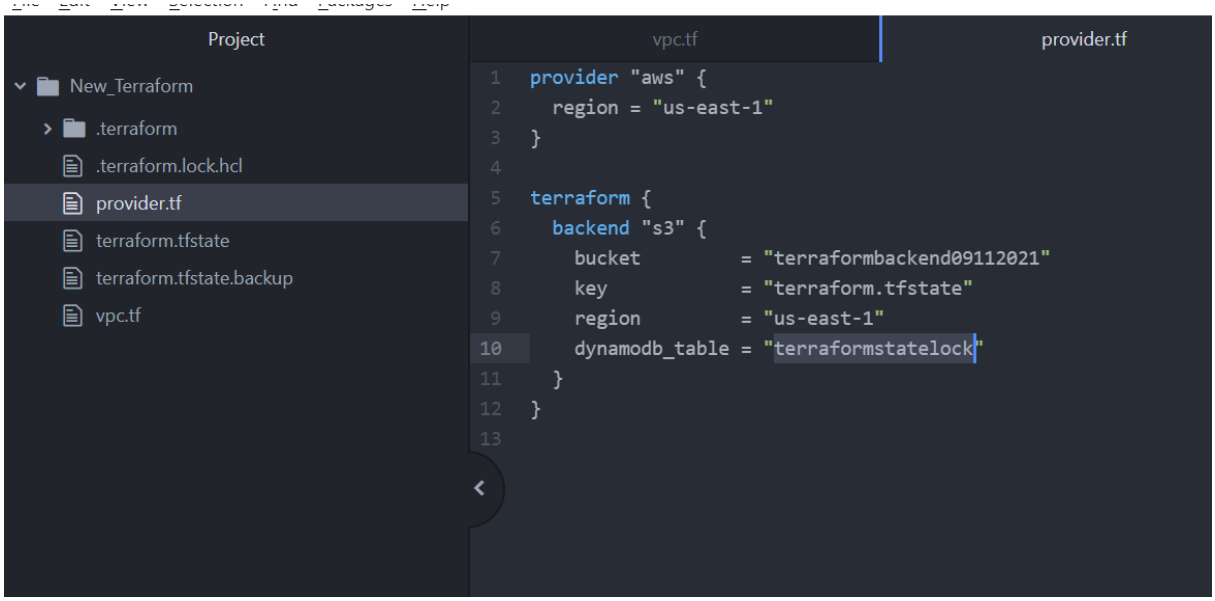
DynamoDB State Locking

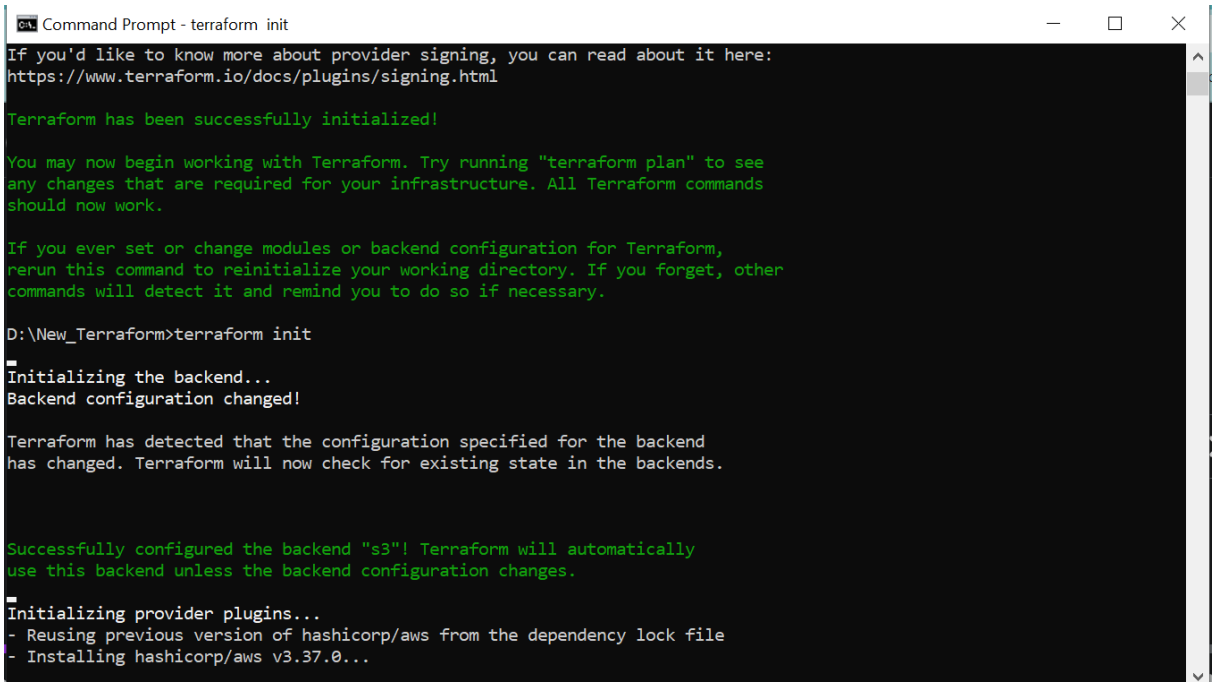
The following configuration is optional:

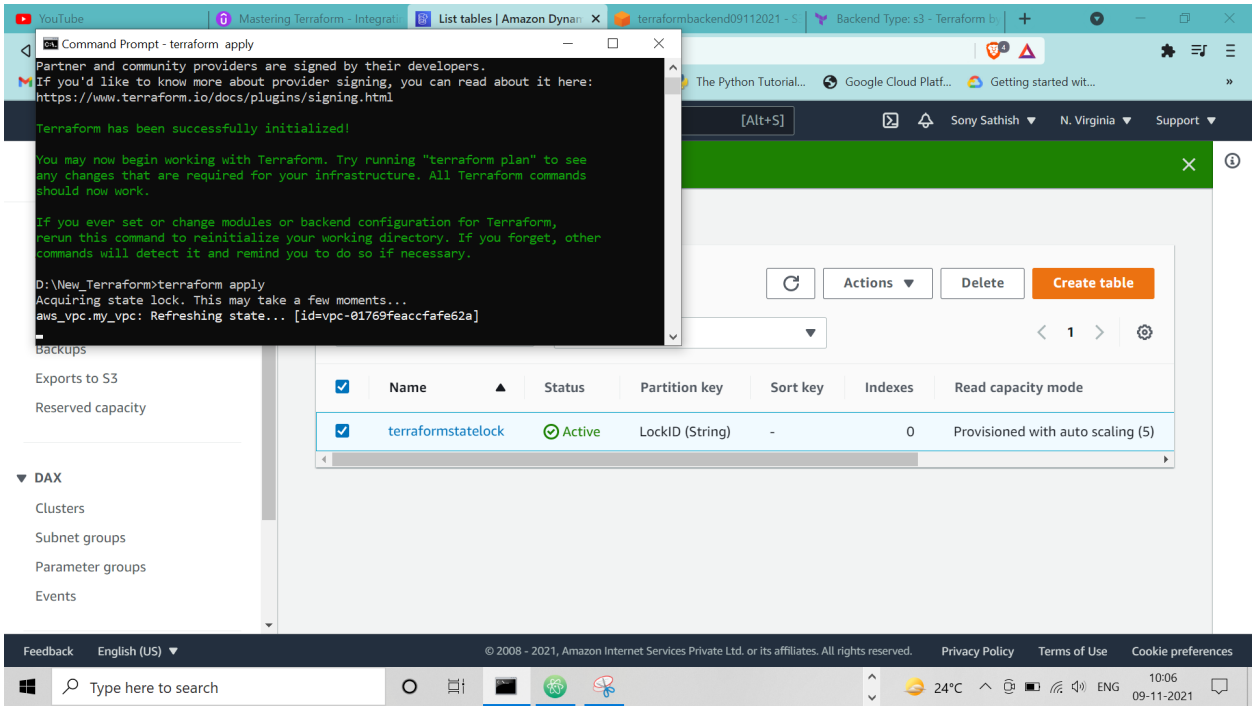
dynamodb_endpoint - (Optional) Custom endpoint for the AWS DynamoDB API. This can also be sourced from the AWS_DYNAMODB_ENDPOINT environment variable.

dynamodb_table - (Optional) Name of DynamoDB Table to use for state locking and consistency. The table must have a primary key named **LockID with type of string**. If not configured, state locking will be disabled.



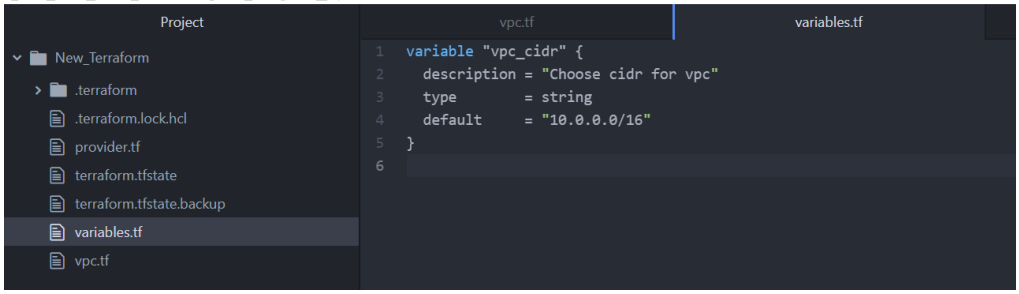

Project
New_Terraform
.terraform
.terraform.lock.hcl
provider.tf
terraform.tfstate
terraform.tfstate.backup
vpc.tf
vpc.tf
provider.tf
1 provider "aws" {
2 region = "us-east-1"
3 }
4
5 terraform {
6 backend "s3" {
7 bucket = "terraformbackend09112021"
8 key = "terraform.tfstate"
9 region = "us-east-1"
10 dynamodb_table = "terraformstatelock"
11 }
12 }
13


Command Prompt - terraform init
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html
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.
D:\New_Terraform>terraform init
Initializing the backend...
Backend configuration changed!
Terraform has detected that the configuration specified for the backend
has changed. Terraform will now check for existing state in the backends.
Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v3.37.0...

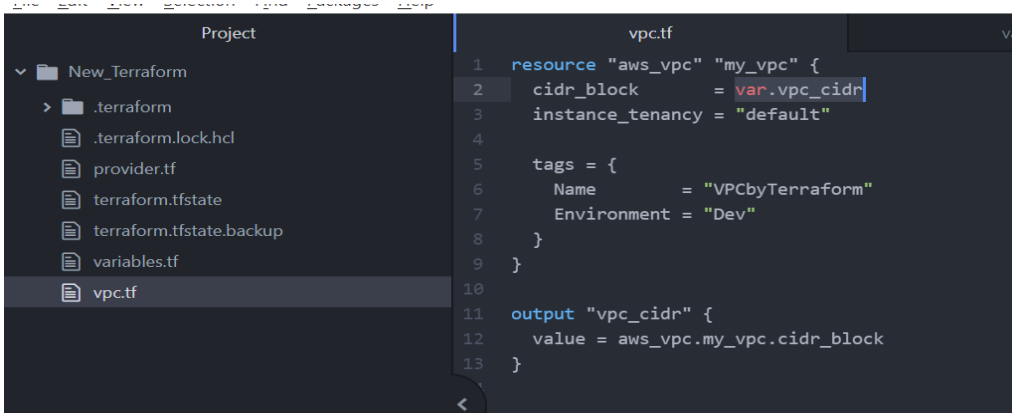


Terraform Variables

Terraform variables ensure good maintenance and reusability of the code. Here, we have maintained the variables in the file variables.tf.



In order to access the variables, we need to pass the variable value like below.



In order to override the variable value of the variables file in the command line -

terraform apply -var “vpc_cidr=10.20.0.0/16” -auto-approve (auto-approve is to approve without giving yes as input every time)

When there is a need to give multiple variables, then it is not possible to supply every variable at the command line giving -var option. In that case we maintain those variables in .tfvars file.

For eg. dev.tfvars contain the below data

vpc_cidr = “10.20.0.0/16”

vpc_cidr = “10.30.0.0/16”

Now at the command line supply the file

terraform apply -var-file=dev.tfvars -auto-approve

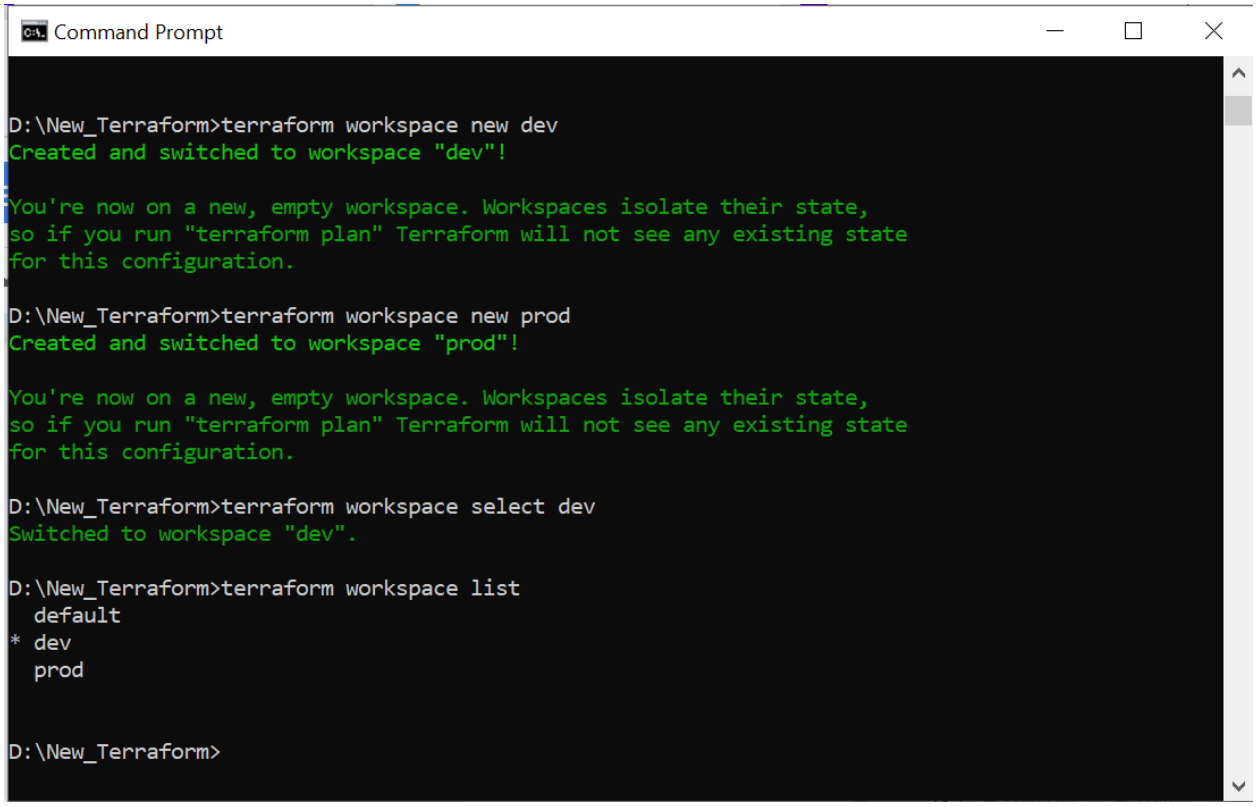
Terraform Workspaces:

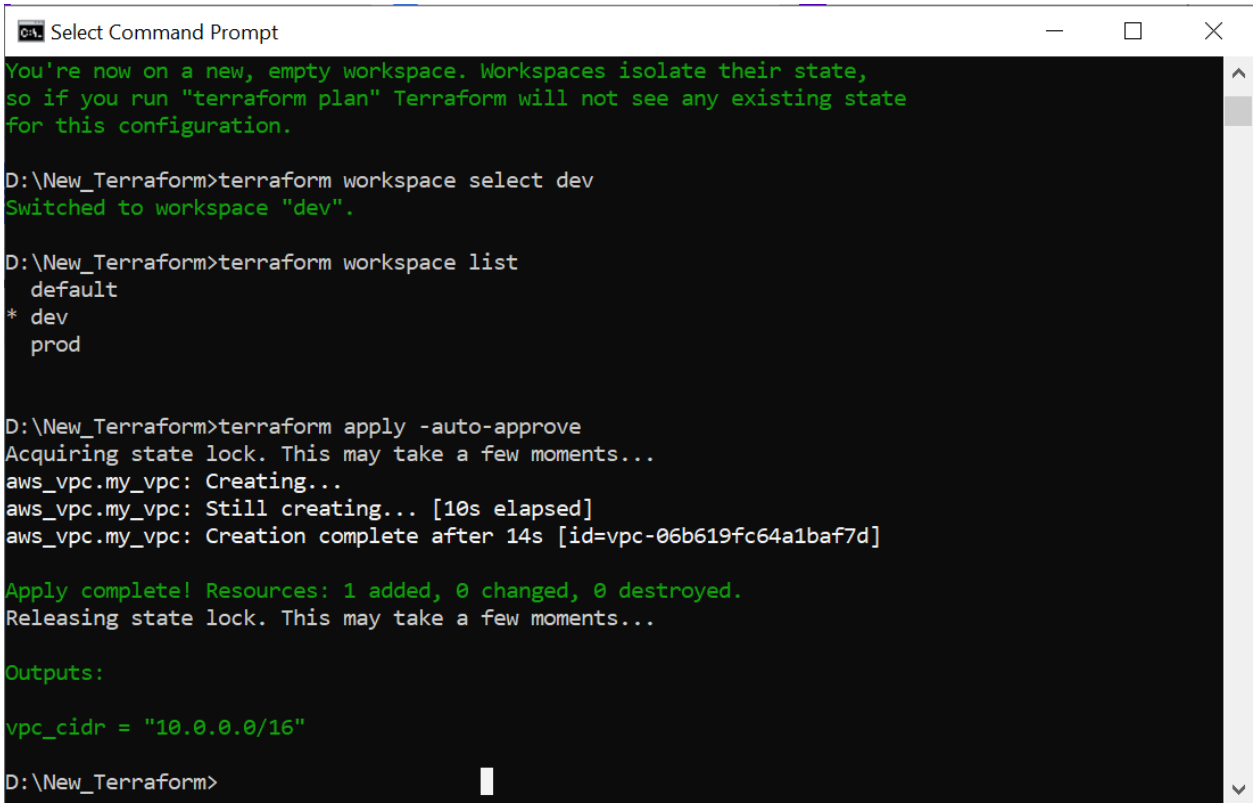
A workspace is basically an environment for the terraform to deploy resources. By default terraform has a workspace by the name “default”. For example below we have maintained 2 different workspaces called dev and prod.Terraform maintains separate state files for different workspaces.

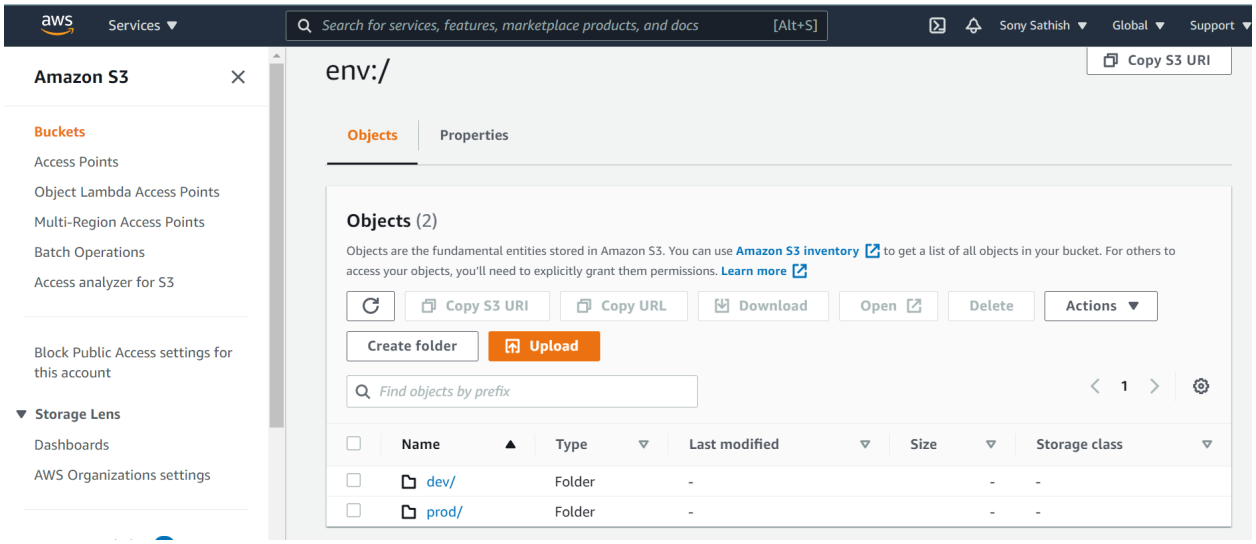
To create a new workspace - **terraform workspace new <env>**

To select the workspace - **terraform workspace select <env>**

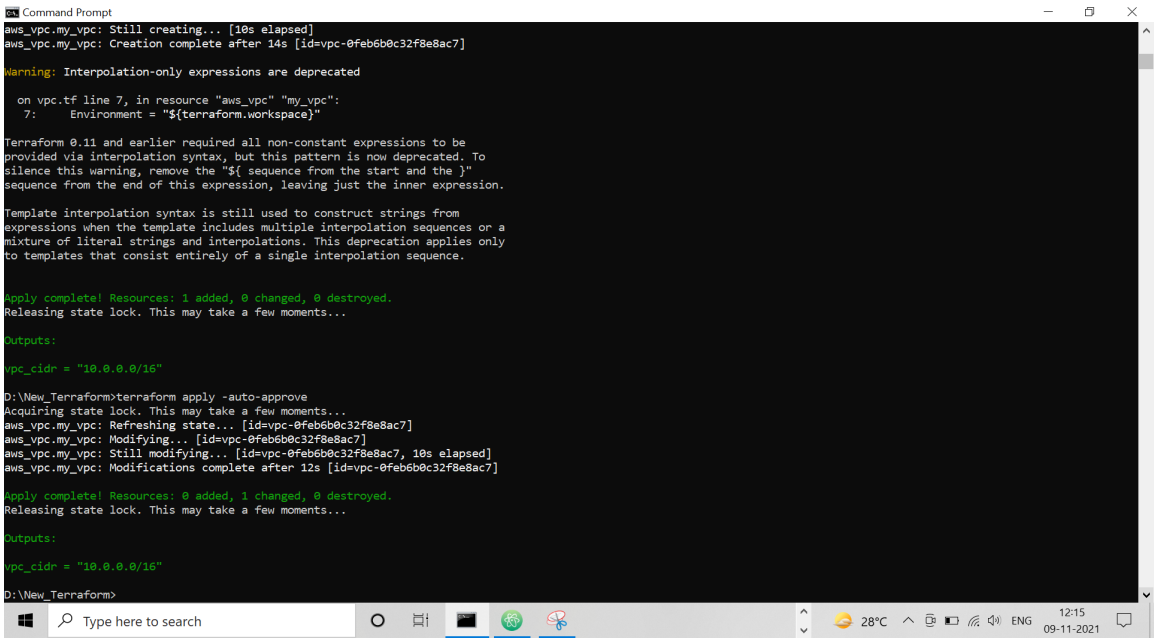
terraform apply -auto-approve


Command Prompt
D:\New_Terraform>terraform workspace new dev
Created and switched to workspace "dev"!
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.
D:\New_Terraform>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.
D:\New_Terraform>terraform workspace select dev
Switched to workspace "dev".
D:\New_Terraform>terraform workspace list
default
* dev
prod
D:\New_Terraform>


Select Command Prompt
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.
D:\New_Terraform>terraform workspace select dev
Switched to workspace "dev".
D:\New_Terraform>terraform workspace list
default
* dev
prod
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc: Creating...
aws_vpc.my_vpc: Still creating... [10s elapsed]
aws_vpc.my_vpc: Creation complete after 14s [id=vpc-06b619fc64a1baf7d]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
Outputs:
vpc_cidr = "10.0.0.0/16"
D:\New_Terraform>



In order to print the current workspace dynamically in the tags, we need to pass the workspace value like below -



Terraform loops:

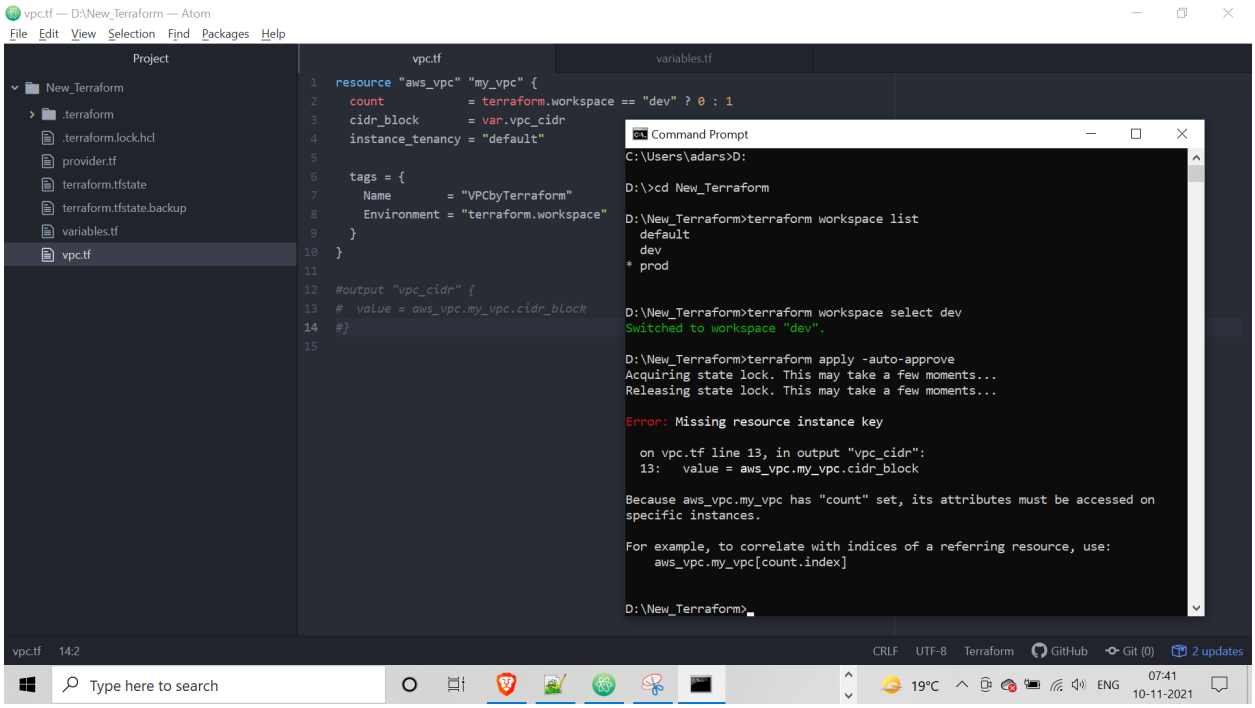
In order to deploy multiple resources of the same type, we use loops. Loops are executed using count meta-argument. Count = 0 doesn't deploy anything or destroys if anything already present

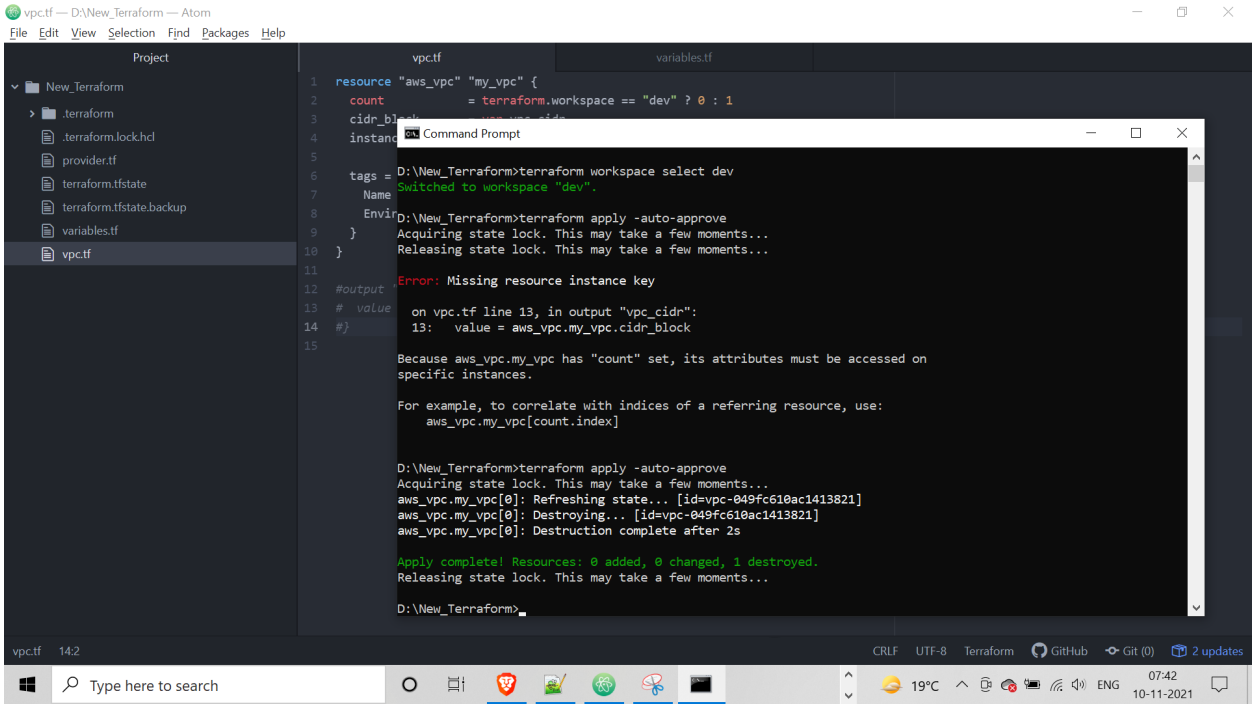
```
resource "aws_instance" "server" {
  count = 4 # create four similar EC2 instances

  ami           = "ami-a1b2c3d4"
  instance_type = "t2.micro"

  tags = {
    Name = "Server ${count.index}"
  }
}
```

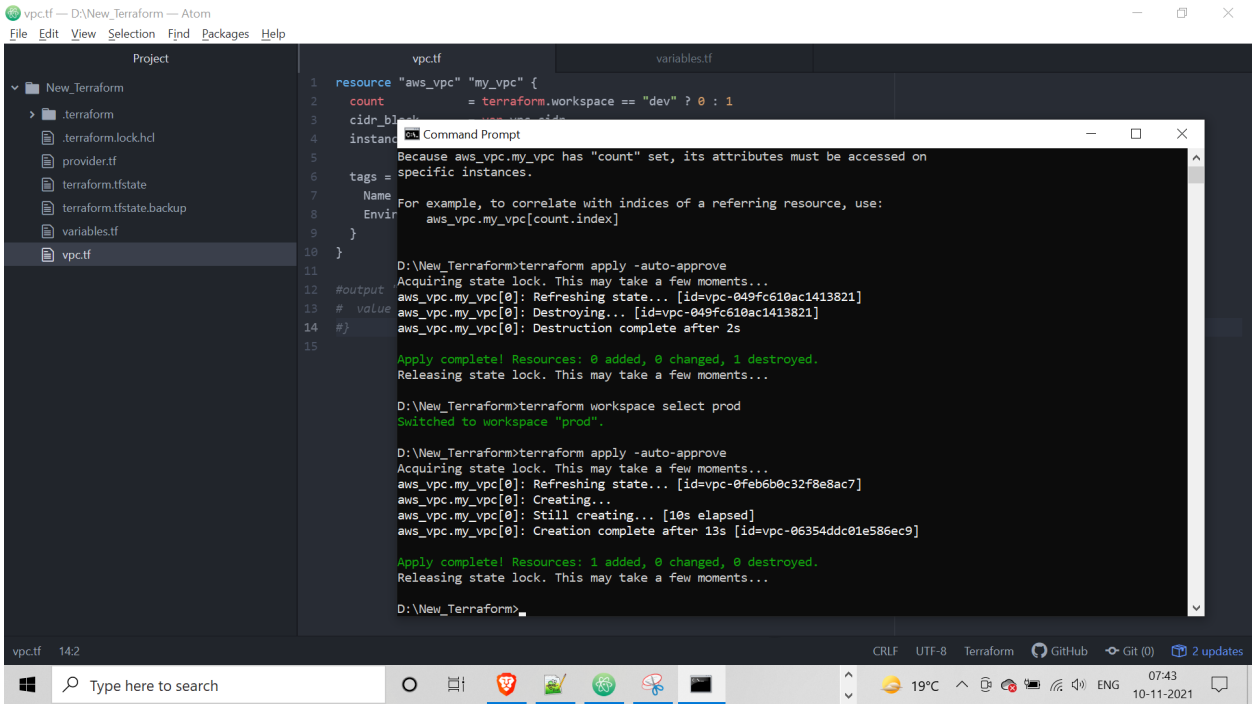
Terraform conditions:

In terraform we can create certain resources conditionally. For example if we want any resource only in prod and not in dev then that can be done using terraform conditions. Here, if the workspace is dev then we are provisioning 0 resources or destroying resources if there are any but provisioning the resources in any workspace environment other than dev.

```
resource "aws_vpc" "my_vpc" {
  count             = terraform.workspace == "dev" ? 0 : 1
```

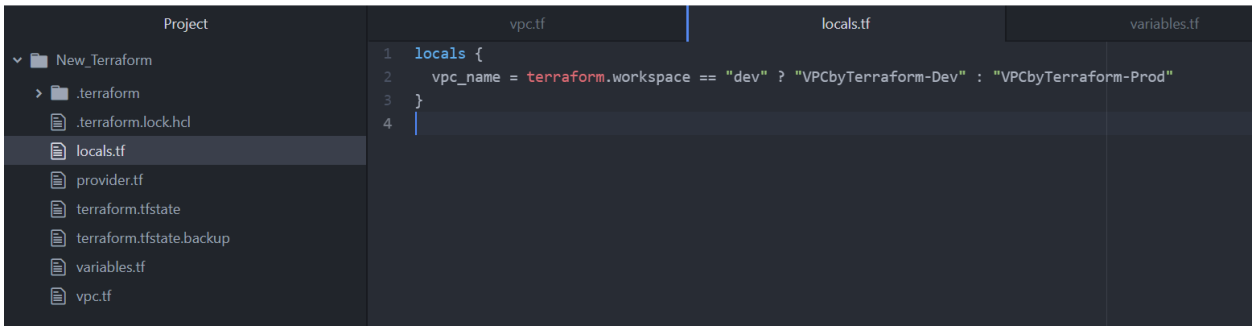

vpc.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
provider.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
variables.tf
resource "aws_vpc" "my_vpc" {
count = terraform.workspace == "dev" ? 0 : 1
cidr_block = var.vpc_cidr
instance_tenancy = "default"
tags = {
Name = "VPCbyTerraform"
Environment = "terraform.workspace"
}
}
#output "vpc_cidr" {
value = aws_vpc.my_vpc.cidr_block
#}
Command Prompt
C:\Users\adars>D:
D:\>cd New_Terraform
D:\New_Terraform>terraform workspace list
default
dev
* prod
D:\New_Terraform>terraform workspace select dev
Switched to workspace "dev".
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
Releasing state lock. This may take a few moments...
Error: Missing resource instance key
on vpc.tf line 13, in output "vpc_cidr":
13: value = aws_vpc.my_vpc.cidr_block
Because aws_vpc.my_vpc has "count" set, its attributes must be accessed on
specific instances.
For example, to correlate with indices of a referring resource, use:
aws_vpc.my_vpc[count.index]
D:\New_Terraform>
vpc.tf 14:2
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
19°C ENG 07:41 10-11-2021

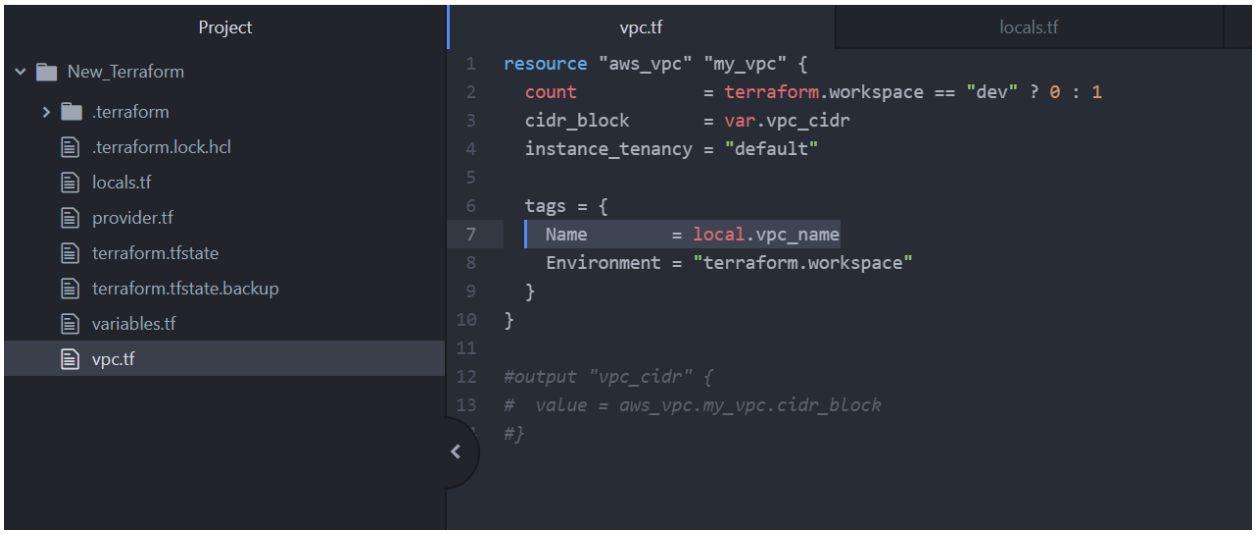

vpc.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
provider.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
variables.tf
Command Prompt
D:\New_Terraform>terraform workspace select dev
Switched to workspace "dev".
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
Releasing state lock. This may take a few moments...
Error: Missing resource instance key
on vpc.tf line 13, in output "vpc_cidr":
13: value = aws_vpc.my_vpc.cidr_block
Because aws_vpc.my_vpc has "count" set, its attributes must be accessed on
specific instances.
For example, to correlate with indices of a referring resource, use:
aws_vpc.my_vpc[count.index]
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc[0]: Refreshing state... [id=vpc-049fc610ac1413821]
aws_vpc.my_vpc[0]: Destroying... [id=vpc-049fc610ac1413821]
aws_vpc.my_vpc[0]: Destruction complete after 2s
Apply complete! Resources: 0 added, 0 changed, 1 destroyed.
Releasing state lock. This may take a few moments...
D:\New_Terraform>
vpc.tf 14:2
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
19°C ENG 07:42 10-11-2021

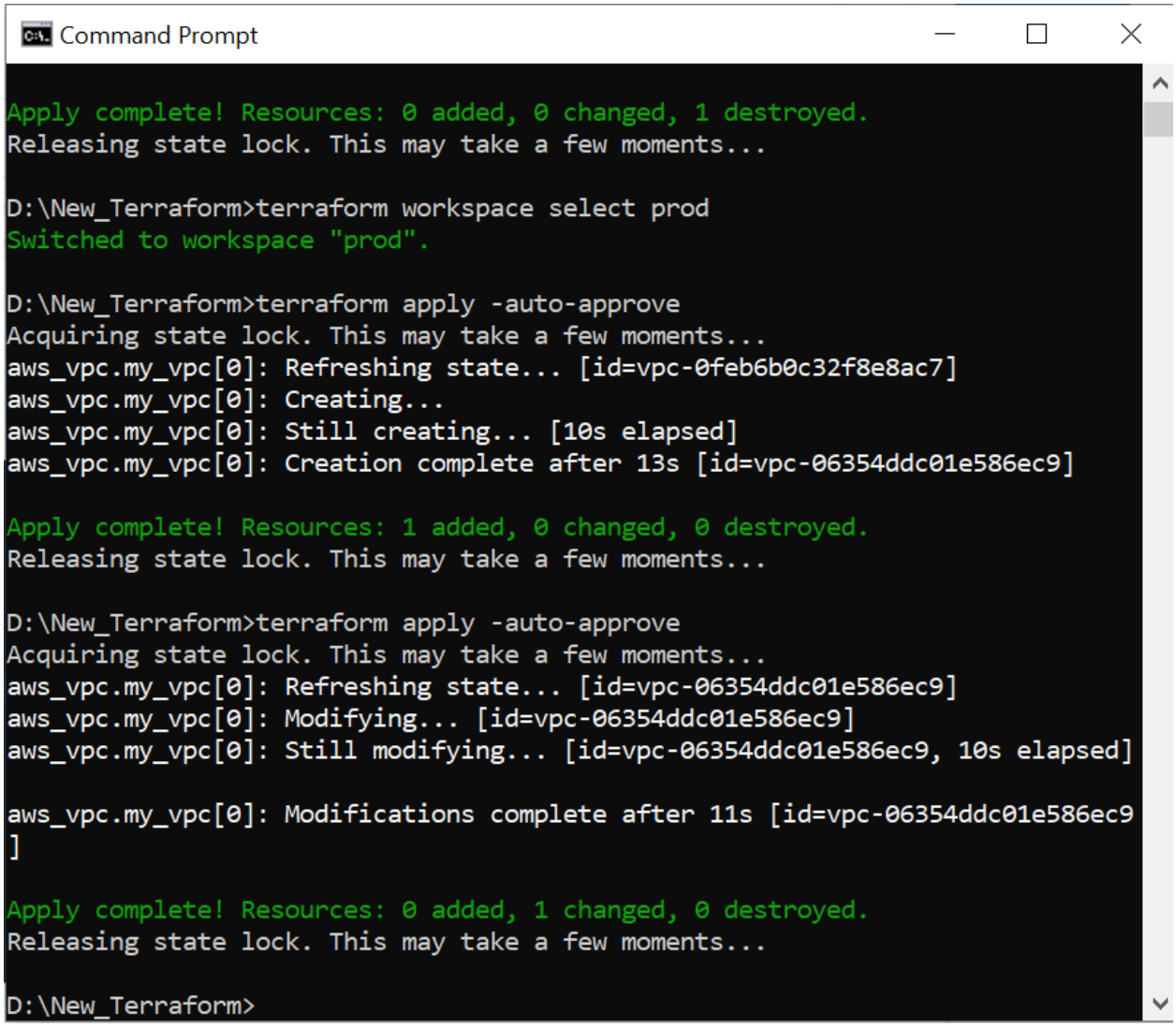


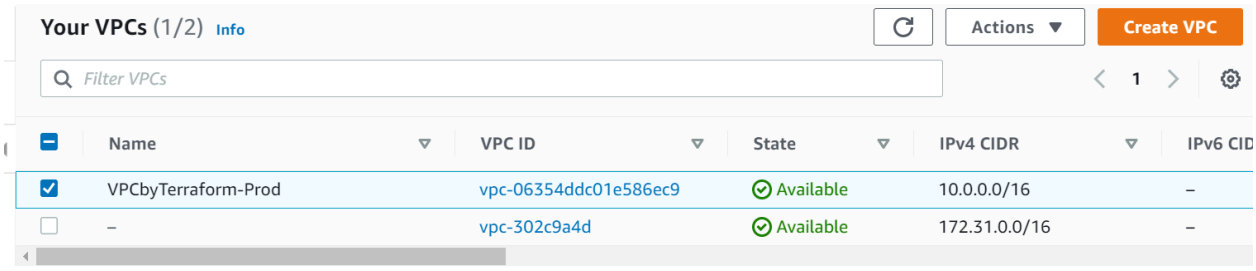
Terraform Locals

If we have any expression which is repeatedly used in the code then we can declare them as a part of the local variable and refer them in the code so that in the future if you have to change the expression, you don't have to change it at several places.



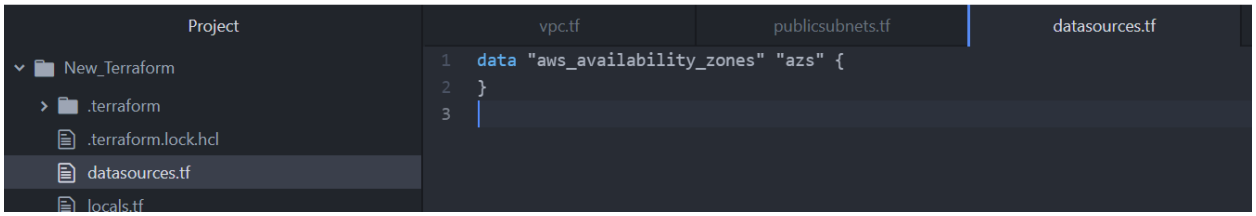

Project
New_Terraform
.terraform
.terraform.lock.hcl
locals.tf
provider.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
locals.tf
resource "aws_vpc" "my_vpc" {
count = terraform.workspace == "dev" ? 0 : 1
cidr_block = var.vpc_cidr
instance_tenancy = "default"
tags = {
Name = local.vpc_name
Environment = "terraform.workspace"
}
}
#output "vpc_cidr" {
value = aws_vpc.my_vpc.cidr_block
#}

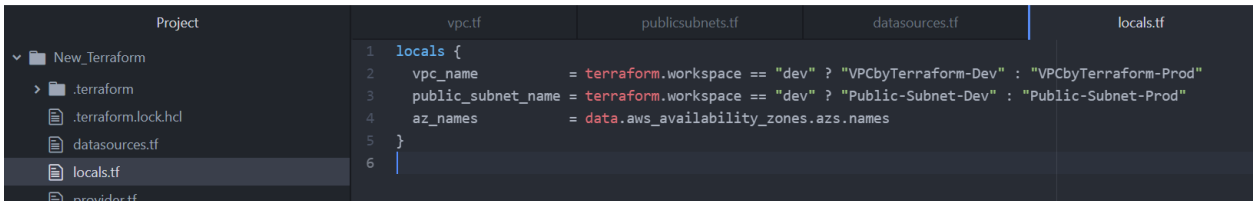

Command Prompt
Apply complete! Resources: 0 added, 0 changed, 1 destroyed.
Releasing state lock. This may take a few moments...
D:\New_Terraform>terraform workspace select prod
Switched to workspace "prod".
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc[0]: Refreshing state... [id=vpc-0feb6b0c32f8e8ac7]
aws_vpc.my_vpc[0]: Creating...
aws_vpc.my_vpc[0]: Still creating... [10s elapsed]
aws_vpc.my_vpc[0]: Creation complete after 13s [id=vpc-06354ddc01e586ec9]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc[0]: Refreshing state... [id=vpc-06354ddc01e586ec9]
aws_vpc.my_vpc[0]: Modifying... [id=vpc-06354ddc01e586ec9]
aws_vpc.my_vpc[0]: Still modifying... [id=vpc-06354ddc01e586ec9, 10s elapsed]
aws_vpc.my_vpc[0]: Modifications complete after 11s [id=vpc-06354ddc01e586ec9
]
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
D:\New_Terraform>



Datasources:

Datasources help us to import certain information which is declared outside of the terraform configuration. For example, if we want to get the most recent ami then we can do it with the help of datasource. Likewise, it also helps us to get all the availability zones of the region where we are provisioning the resources.

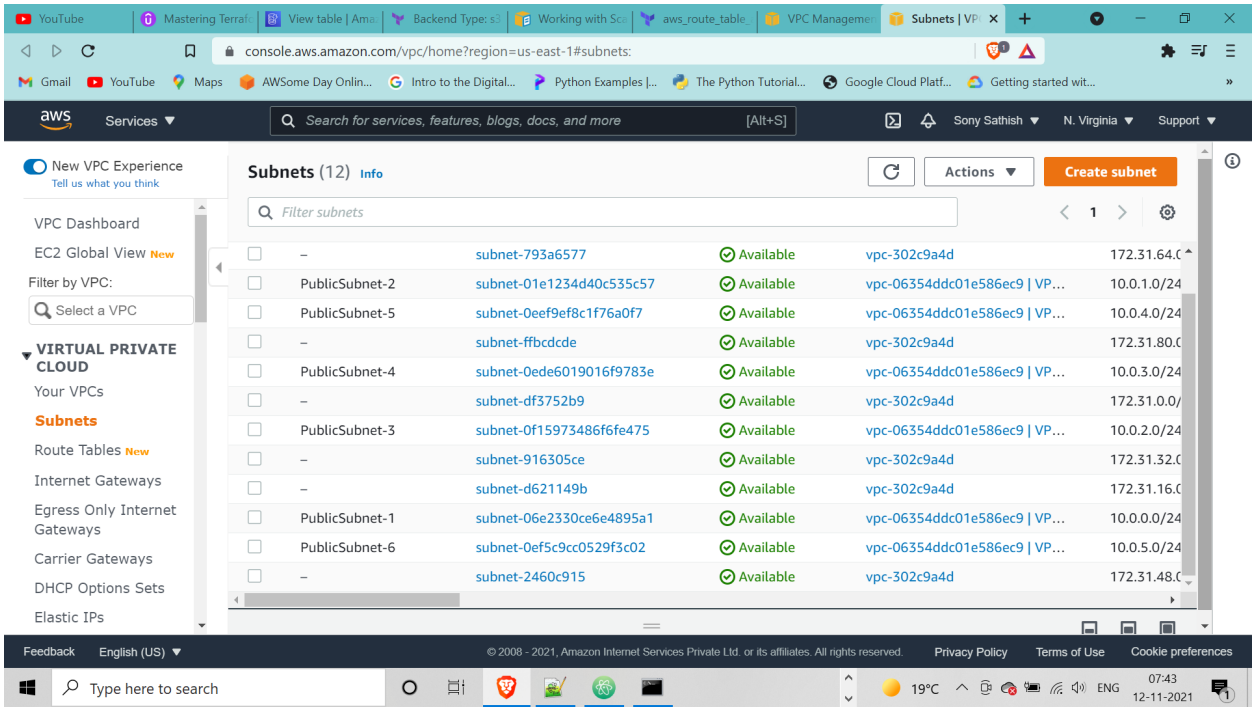




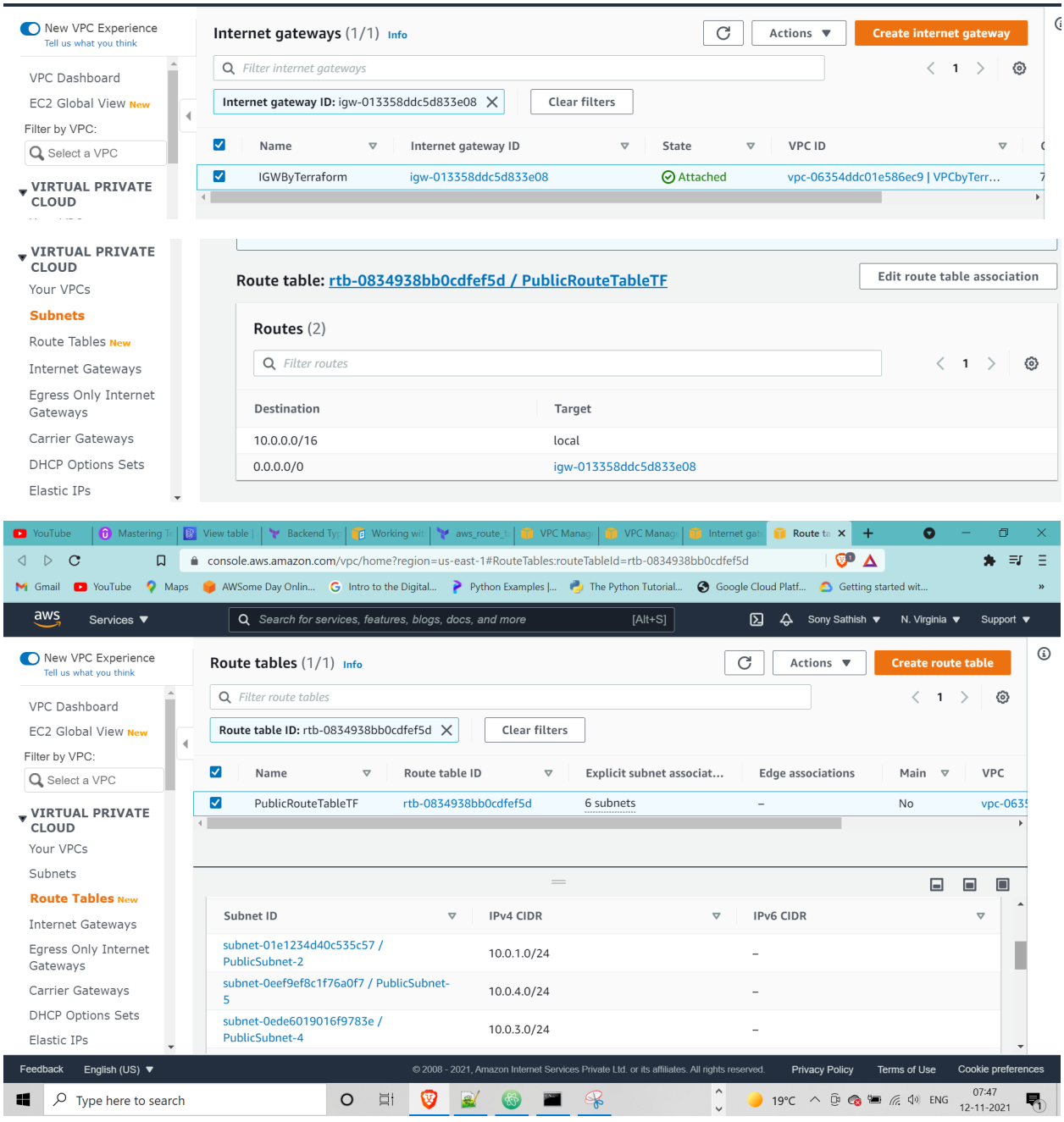
Firstly, we declared the data source in the datasources.tf file. We made the use of local variables for az_names in the locals.tf file. While provisioning the public subnets, we made the use of loops using count so that the public subnet is provisioned into all the availability zones available in us-east-1 region. We passed the length function into count so that it picks all the AZs. In cidr_block we passed 3 parameters into the cidrsubnet function. First parameter is the vpc cidr declared in the variables.tf file. Second parameter is a number which gets added to the cidr notation of the vpc cidr. For eg. Here, the parameter is 8 so it gets added to /16 of the vpc cidr and gives /24 for all the subnets. Lastly, we give count.index so that the subnet ipv4 address goes serially in the way 10.0.0.0/24, 10.0.1.0/24, 10.0.2.0/24, 10.0.3.0/24….till the

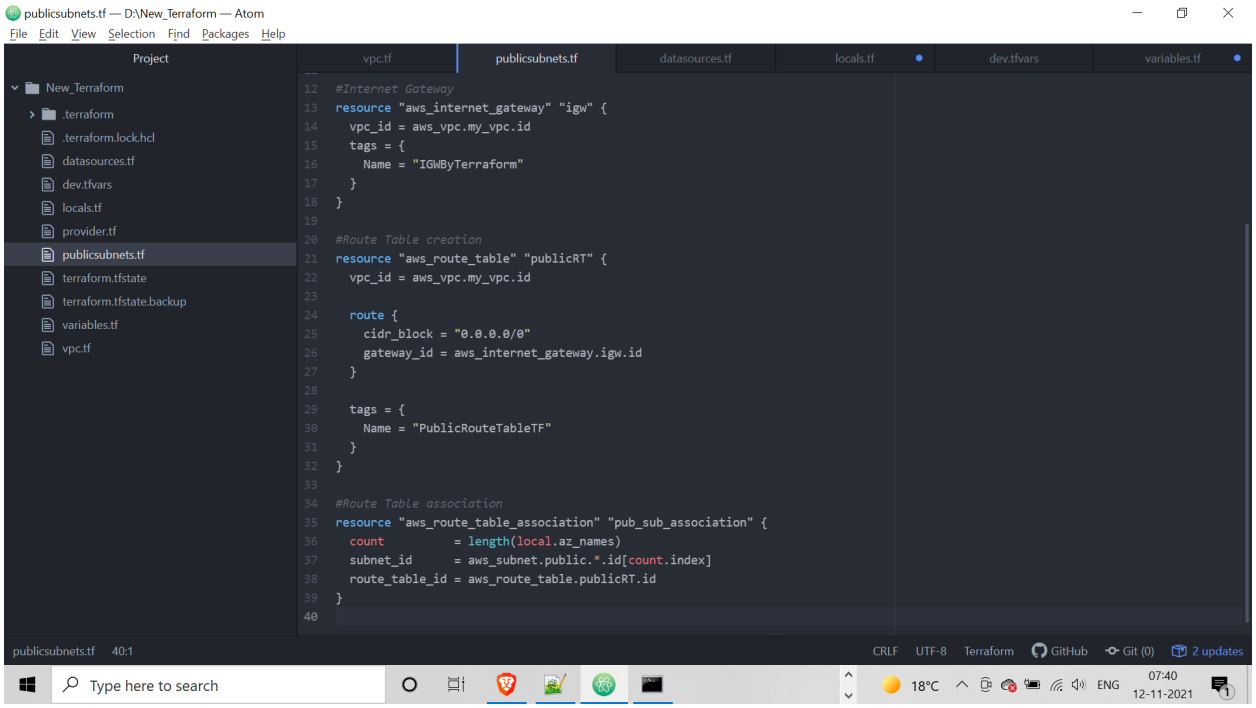
number of AZ is exhausted in the region. Lastly, in the tags we are passing the index with +1 as the first index will always be 0. This way the subnet names will be in the format PublicSubnet-1, PublicSubnet-2, PublicSubnet-3...and so on.

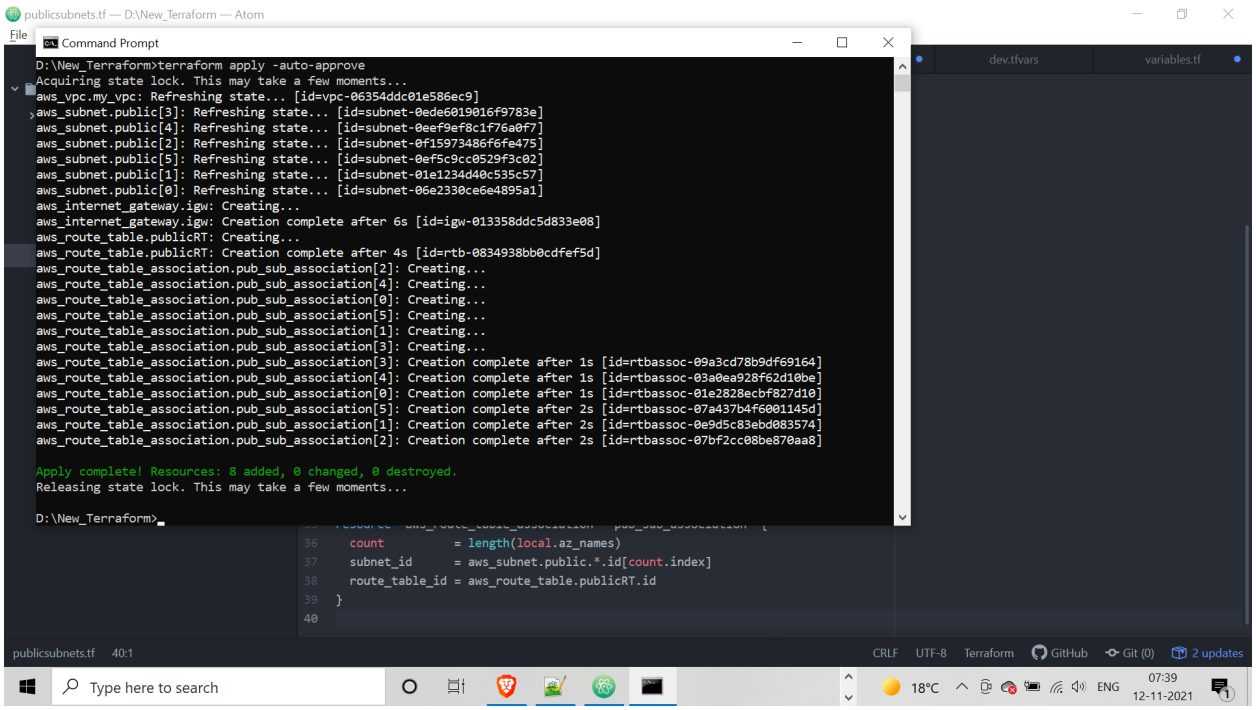
```
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc: Refreshing state... [id=vpc-06354ddc01e586ec9]
aws_subnet.public[4]: Creating...
aws_subnet.public[1]: Creating...
aws_subnet.public[3]: Creating...
aws_subnet.public[5]: Creating...
aws_subnet.public[0]: Creating...
aws_subnet.public[2]: Creating...
aws_subnet.public[3]: Creation complete after 4s [id=subnet-0ede6019016f9783e]
aws_subnet.public[1]: Creation complete after 4s [id=subnet-01e1234d40c535c57]
aws_subnet.public[2]: Creation complete after 4s [id=subnet-0f15973486f6fe475]
aws_subnet.public[4]: Creation complete after 4s [id=subnet-0eef9ef8c1f76a0f7]
aws_subnet.public[5]: Creation complete after 4s [id=subnet-0ef5c9cc0529f3c02]
aws_subnet.public[0]: Creation complete after 4s [id=subnet-06e2330ce6e4895a1]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
```



Internet gateway and route table creation and association







Creating Private Subnets:

Here, slice function helps to deploy only 2 private subnets with index 0 and 1.

count = length(slice(local.az_names, 0, 2))

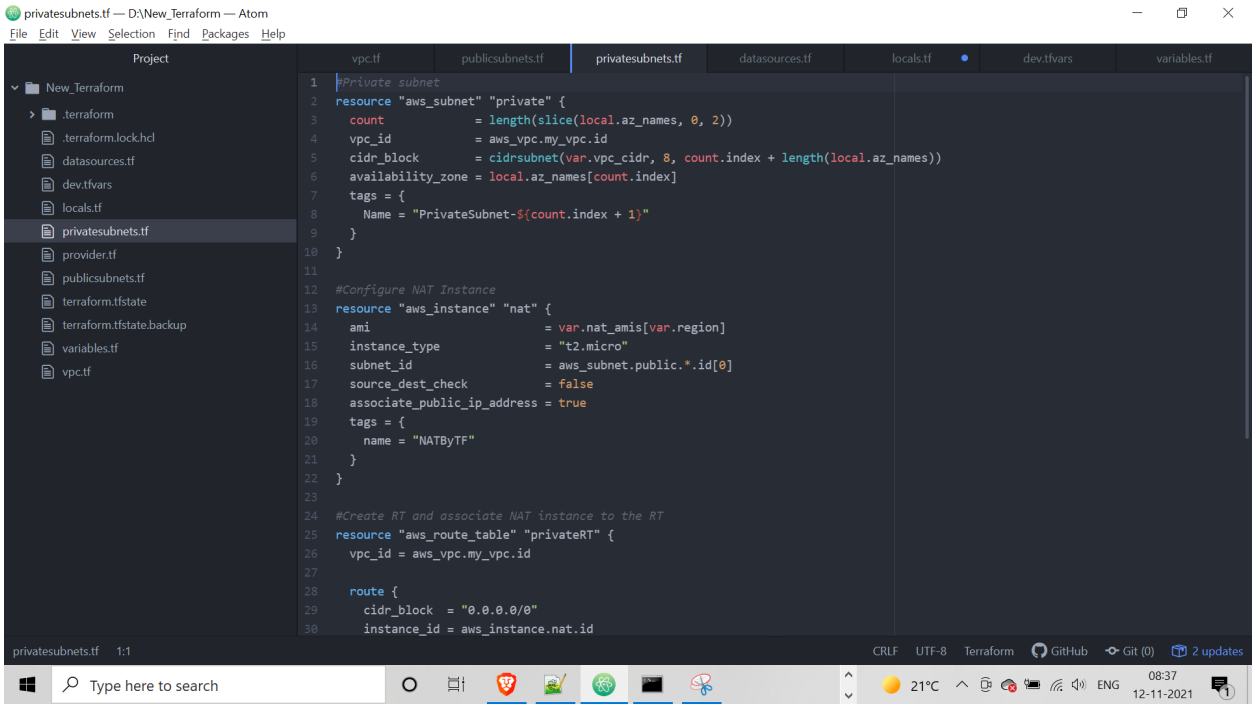
Here, index is picked from where it is left after creating public subnets.

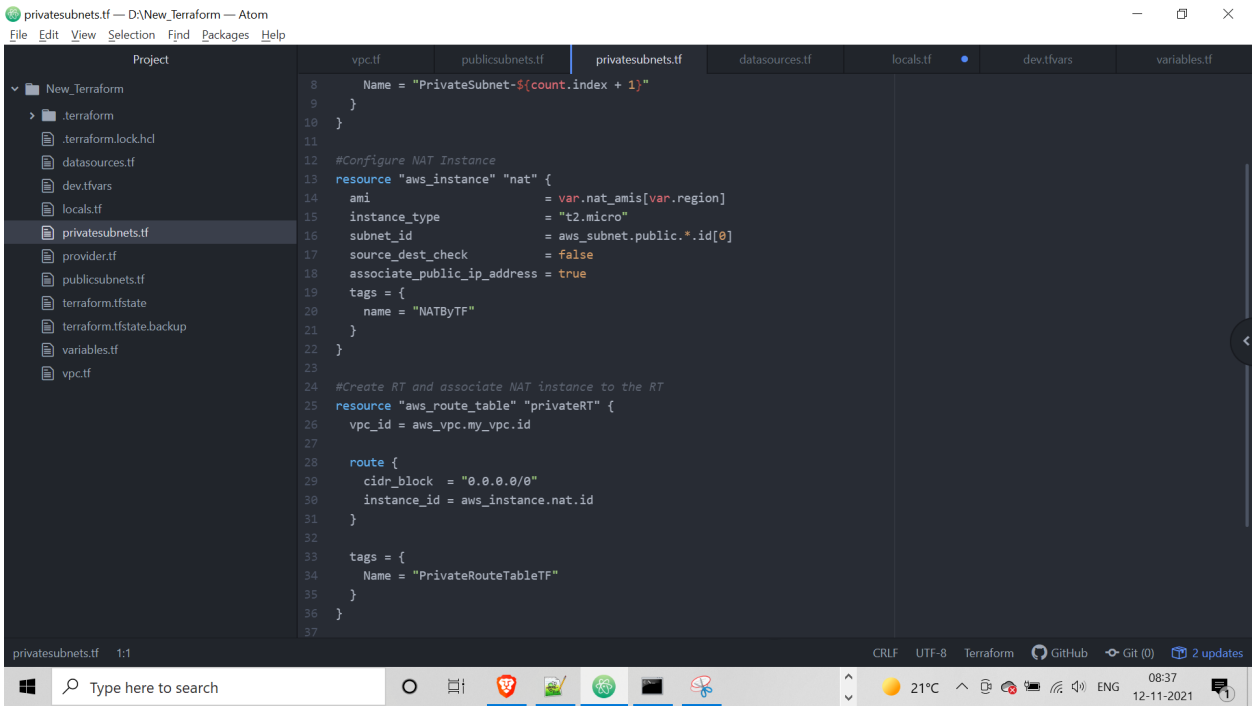
cidr_block = cidrsubnet(var.vpc_cidr, 8, count.index + length(local.az_names))

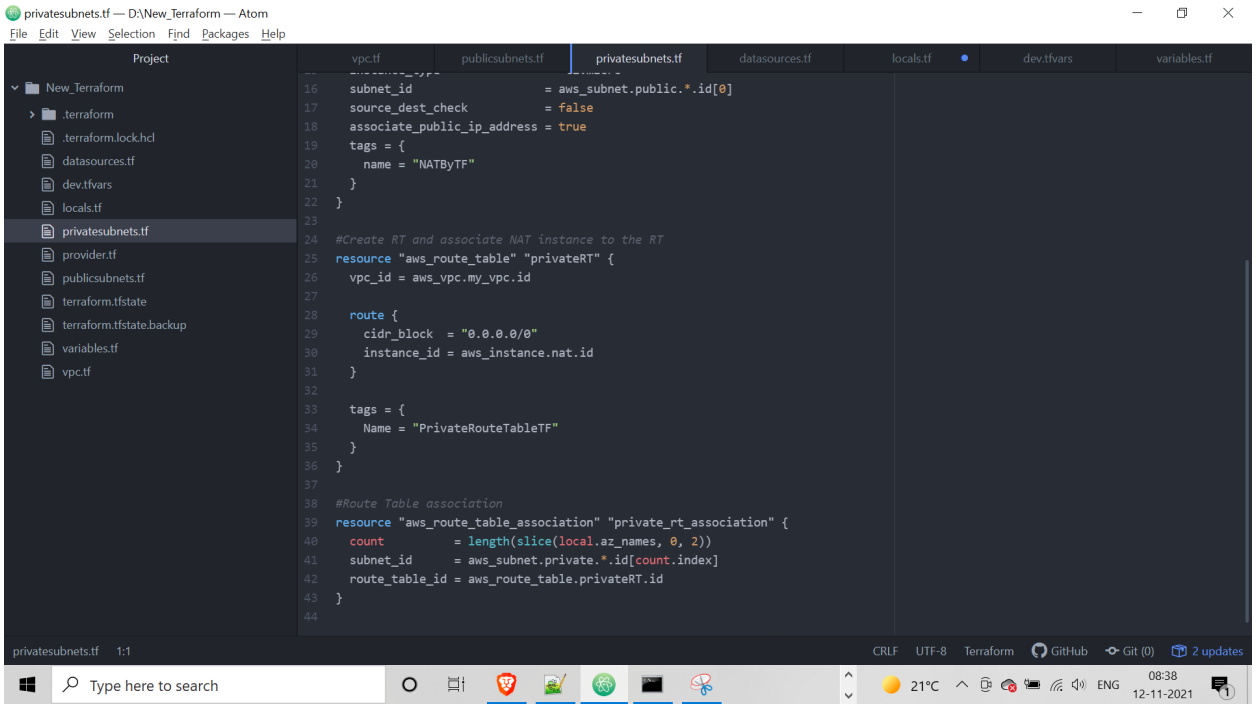

```
D:\New_Terraform>terraform apply -auto-approve
Acquiring state lock. This may take a few moments...
aws_vpc.my_vpc: Refreshing state... [id=vpc-06354ddc01e586ec9]
aws_internet_gateway.igw: Refreshing state... [id=igw-013358ddc5d833e08]
aws_subnet.public[3]: Refreshing state... [id=subnet-0ede6019016f9783e]
aws_subnet.public[1]: Refreshing state... [id=subnet-01e1234d40c535c57]
aws_subnet.public[4]: Refreshing state... [id=subnet-0eef9ef8c1f76a0f7]
aws_subnet.public[5]: Refreshing state... [id=subnet-0ef5c9cc0529f3c02]
aws_subnet.public[2]: Refreshing state... [id=subnet-0f15973486f6fe475]
aws_subnet.public[0]: Refreshing state... [id=subnet-06e2330ce6e4895a1]
aws_route_table.publicRT: Refreshing state... [id=rtb-0834938bb0cdfef5d]
aws_route_table_association.pub_sub_association[4]: Refreshing state... [id=rtbassoc-03a0ea928f62d10be]
aws_route_table_association.pub_sub_association[5]: Refreshing state... [id=rtbassoc-07a437b4f6001145d]
aws_route_table_association.pub_sub_association[2]: Refreshing state... [id=rtbassoc-07bf2cc08be870aa8]
aws_route_table_association.pub_sub_association[1]: Refreshing state... [id=rtbassoc-0e9d5c83ebd083574]
aws_route_table_association.pub_sub_association[3]: Refreshing state... [id=rtbassoc-09a3cd78b9df69164]
aws_route_table_association.pub_sub_association[0]: Refreshing state... [id=rtbassoc-01e2828ecbf827d10]
aws_subnet.private[0]: Creating...
aws_subnet.private[1]: Creating...
aws_subnet.private[1]: Creation complete after 3s [id=subnet-02bd9fdc4406903e4]
aws_subnet.private[0]: Creation complete after 3s [id=subnet-0b9fca7853339ae6c]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
```

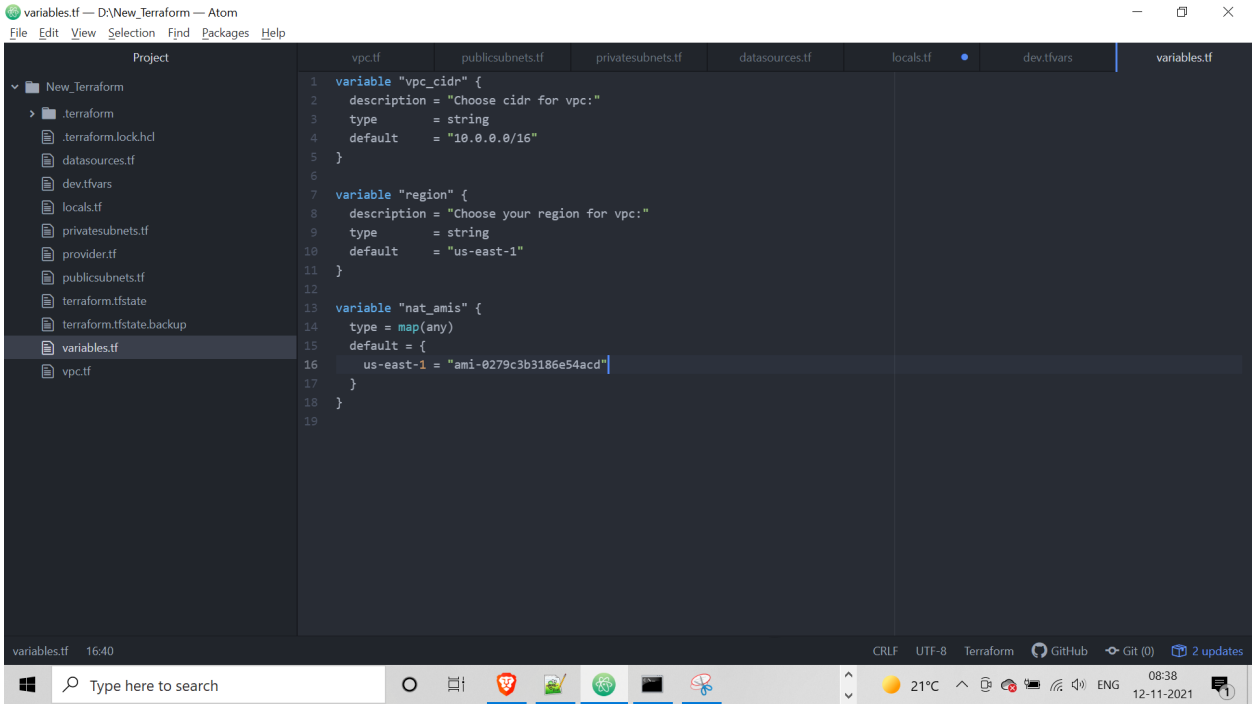
☐	–	subnet-2460c915	Available	vpc-302c9a4d	172.31.48.0
☐	PrivateSubnet-1	subnet-0b9fca7853339ae6c	Available	vpc-06354ddc01e586ec9 | VP...	10.0.6.0/24
☐	–	subnet-793a6577	Available	vpc-302c9a4d	172.31.64.0
☐	–	subnet-ffbcdcde	Available	vpc-302c9a4d	172.31.80.0
☐	PrivateSubnet-2	subnet-02bd9fdc4406903e4	Available	vpc-06354ddc01e586ec9 | VP...	10.0.7.0/24
☐	–	subnet-916305ce	Available	vpc-302c9a4d	172.31.32.0

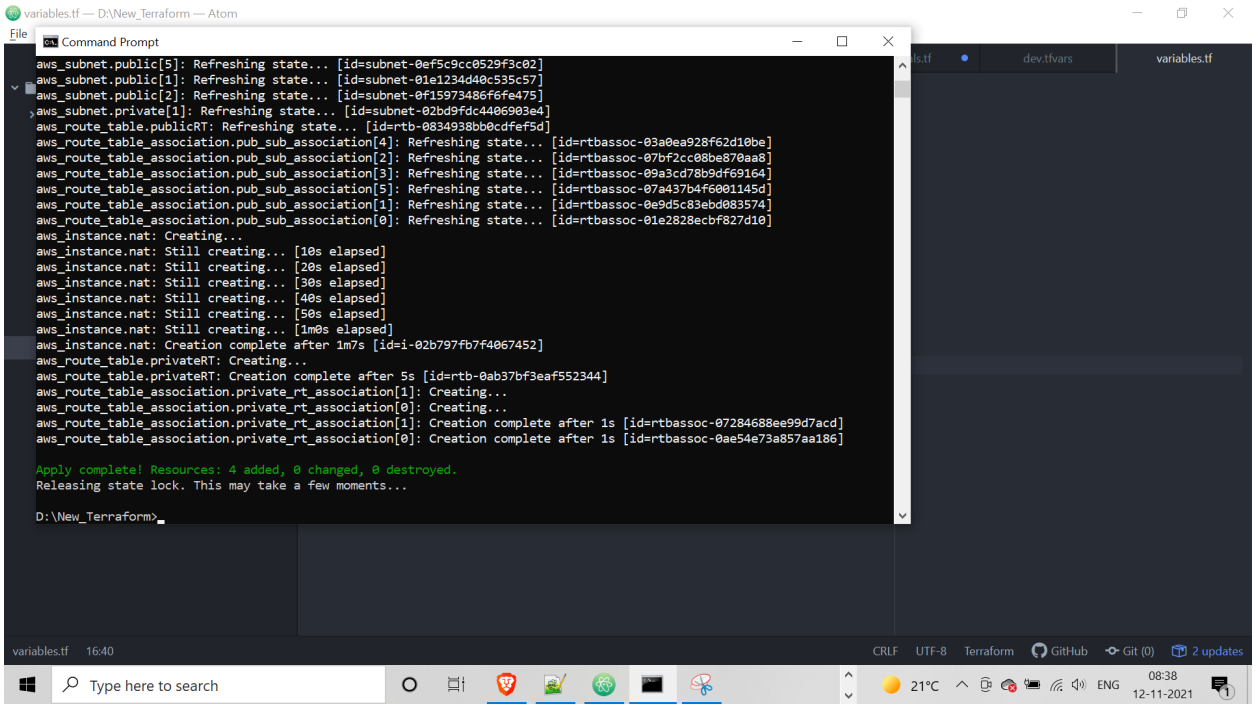
Create NAT instance

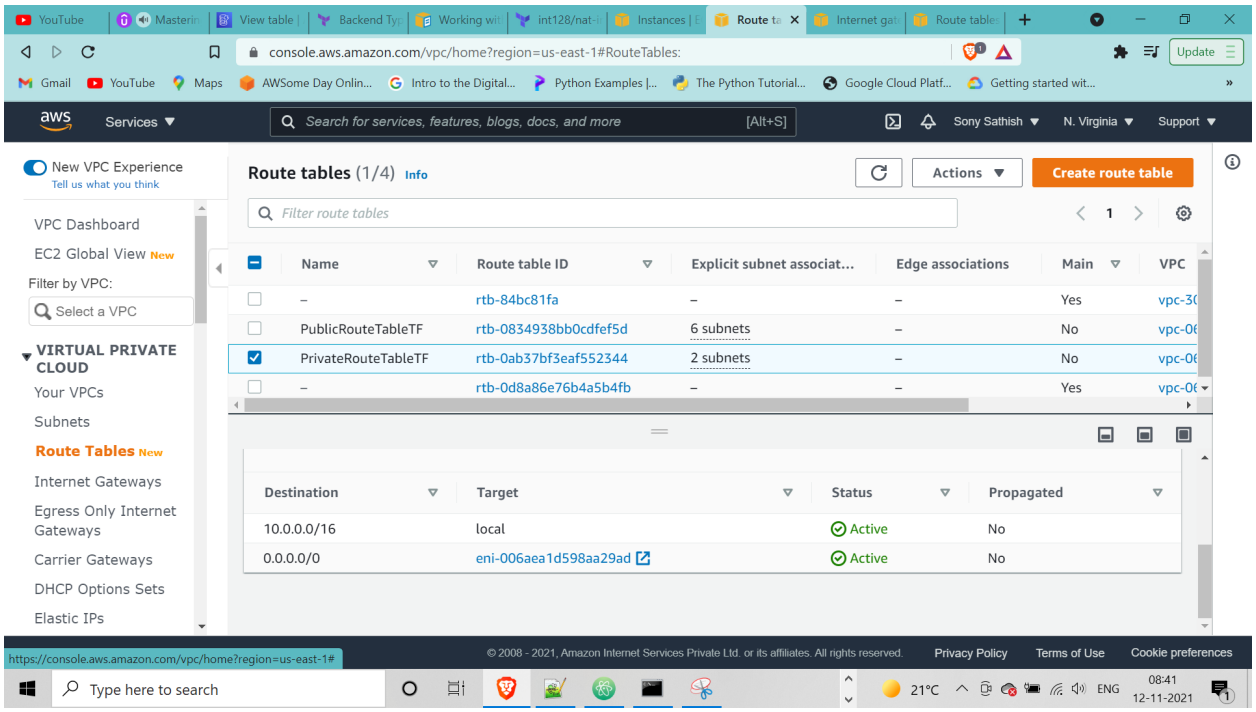

privatesubnets.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
datasources.tf
dev.tfvars
locals.tf
privatesubnets.tf
provider.tf
publicsubnets.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
publicsubnets.tf
privatesubnets.tf
datasources.tf
locals.tf
dev.tfvars
variables.tf
#Private subnet
resource "aws_subnet" "private" {
count = length(slice(local.az_names, 0, 2))
vpc_id = aws_vpc.my_vpc.id
cidr_block = cidrsubnet(var.vpc_cidr, 8, count.index + length(local.az_names))
availability_zone = local.az_names[count.index]
tags = {
Name = "PrivateSubnet-${count.index + 1}"
}
}
#Configure NAT Instance
resource "aws_instance" "nat" {
ami = var.nat_amis[var.region]
instance_type = "t2.micro"
subnet_id = aws_subnet.public.*.id[0]
source_dest_check = false
associate_public_ip_address = true
tags = {
name = "NATByTF"
}
}
#Create RT and associate NAT instance to the RT
resource "aws_route_table" "privateRT" {
vpc_id = aws_vpc.my_vpc.id
route {
cidr_block = "0.0.0.0/0"
instance_id = aws_instance.nat.id
privatesubnets.tf 1:1
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
21°C
ENG
08:37
12-11-2021

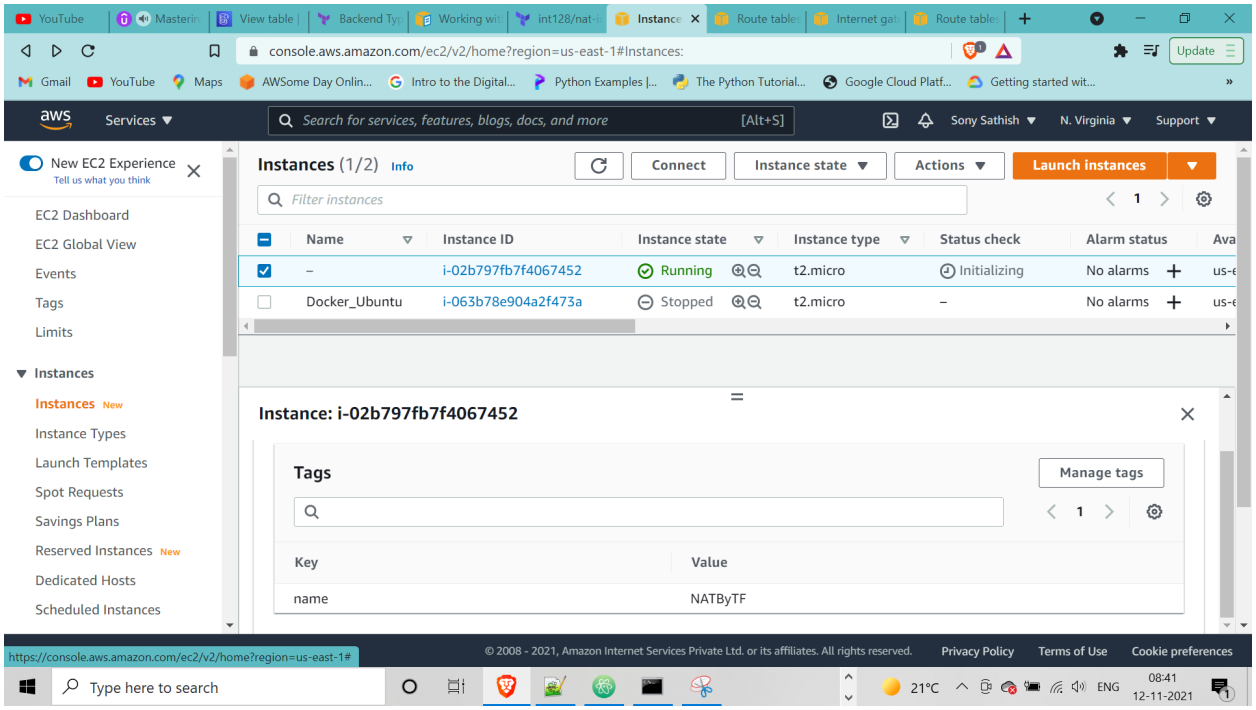

privatesubnets.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
datasources.tf
dev.tfvars
locals.tf
privatesubnets.tf
provider.tf
publicsubnets.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
publicsubnets.tf
privatesubnets.tf
datasources.tf
locals.tf
dev.tfvars
variables.tf
Name = "PrivateSubnet-${count.index + 1}"
}
}
#Configure NAT Instance
resource "aws_instance" "nat" {
ami = var.nat_amis[var.region]
instance_type = "t2.micro"
subnet_id = aws_subnet.public.*.id[0]
source_dest_check = false
associate_public_ip_address = true
tags = {
name = "NATByTF"
}
}
#Create RT and associate NAT instance to the RT
resource "aws_route_table" "privateRT" {
vpc_id = aws_vpc.my_vpc.id
route {
cidr_block = "0.0.0.0/0"
instance_id = aws_instance.nat.id
}
tags = {
Name = "PrivateRouteTableTF"
}
}
privatesubnets.tf 1:1
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
21°C
ENG
08:37
12-11-2021


privatesubnets.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
datasources.tf
dev.tfvars
locals.tf
privatesubnets.tf
provider.tf
publicsubnets.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
publicsubnets.tf
privatesubnets.tf
datasources.tf
locals.tf
dev.tfvars
variables.tf
subnet_id = aws_subnet.public.*.id[0]
source_dest_check = false
associate_public_ip_address = true
tags = {
name = "NATByTF"
}
}
#Create RT and associate NAT instance to the RT
resource "aws_route_table" "privateRT" {
vpc_id = aws_vpc.my_vpc.id
route {
cidr_block = "0.0.0.0/0"
instance_id = aws_instance.nat.id
}
tags = {
Name = "PrivateRouteTableTF"
}
}
#Route Table association
resource "aws_route_table_association" "private_rt_association" {
count = length(slice(local.az_names, 0, 2))
subnet_id = aws_subnet.private.*.id[count.index]
route_table_id = aws_route_table.privateRT.id
}
privatesubnets.tf 1:1
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
21°C
ENG
08:38
12-11-2021


variables.tf — D:\New_Terraform — Atom
File Edit View Selection Find Packages Help
Project
New_Terraform
.terraform
.terraform.lock.hcl
datasources.tf
dev.tfvars
locals.tf
privatesubnets.tf
provider.tf
publicsubnets.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
vpc.tf
vpc.tf
publicsubnets.tf
privatesubnets.tf
datasources.tf
locals.tf
dev.tfvars
variables.tf
variable "vpc_cidr" {
description = "Choose cidr for vpc:"
type = string
default = "10.0.0.0/16"
}
variable "region" {
description = "Choose your region for vpc:"
type = string
default = "us-east-1"
}
variable "nat_amis" {
type = map(any)
default = {
us-east-1 = "ami-0279c3b3186e54acd"
}
}
variables.tf 16:40
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
21°C
ENG
08:38
12-11-2021


variables.tf — D:\New_Terraform — Atom
File
Command Prompt
aws_subnet.public[5]: Refreshing state... [id=subnet-0ef5c9cc0529f3c02]
aws_subnet.public[1]: Refreshing state... [id=subnet-01e1234d40c535c57]
aws_subnet.public[2]: Refreshing state... [id=subnet-0f15973486f6fe475]
aws_subnet.private[1]: Refreshing state... [id=subnet-02bd9fdc4406903e4]
aws_route_table.publicRT: Refreshing state... [id=rtb-0834938bb0cdfef5d]
aws_route_table_association.pub_sub_association[4]: Refreshing state... [id=rtbassoc-03a0ea928f62d10be]
aws_route_table_association.pub_sub_association[2]: Refreshing state... [id=rtbassoc-07bf2cc08be870aa8]
aws_route_table_association.pub_sub_association[3]: Refreshing state... [id=rtbassoc-09a3cd78b9df69164]
aws_route_table_association.pub_sub_association[5]: Refreshing state... [id=rtbassoc-07a437b4f6001145d]
aws_route_table_association.pub_sub_association[1]: Refreshing state... [id=rtbassoc-0e9d5c83ebd083574]
aws_route_table_association.pub_sub_association[0]: Refreshing state... [id=rtbassoc-01e2828ecbf827d10]
aws_instance.nat: Creating...
aws_instance.nat: Still creating... [10s elapsed]
aws_instance.nat: Still creating... [20s elapsed]
aws_instance.nat: Still creating... [30s elapsed]
aws_instance.nat: Still creating... [40s elapsed]
aws_instance.nat: Still creating... [50s elapsed]
aws_instance.nat: Still creating... [1m0s elapsed]
aws_instance.nat: Creation complete after 1m7s [id=i-02b797fb7f4067452]
aws_route_table.privateRT: Creating...
aws_route_table.privateRT: Creation complete after 5s [id=rtb-0ab37bf3eaf552344]
aws_route_table_association.private_rt_association[1]: Creating...
aws_route_table_association.private_rt_association[0]: Creating...
aws_route_table_association.private_rt_association[1]: Creation complete after 1s [id=rtbassoc-07284688ee99d7acd]
aws_route_table_association.private_rt_association[0]: Creation complete after 1s [id=rtbassoc-0ae54e73a857aa186]
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
D:\New_Terraform>
dev.tfvars
variables.tf
variables.tf 16:40
CRLF UTF-8 Terraform GitHub Git (0) 2 updates
Type here to search
21°C ENG 08:38 12-11-2021


console.aws.amazon.com/vpc/home?region=us-east-1#RouteTables:
Gmail YouTube Maps AWSome Day Onlin... Intro to the Digital... Python Examples |... The Python Tutorial... Google Cloud Platf... Getting started wit...
aws Services Search for services, features, blogs, docs, and more [Alt+S] Sony Sathish N. Virginia Support
New VPC Experience
Tell us what you think
VPC Dashboard
EC2 Global View New
Filter by VPC:
Select a VPC
VIRTUAL PRIVATE CLOUD
Your VPCs
Subnets
Route Tables New
Internet Gateways
Egress Only Internet Gateways
Carrier Gateways
DHCP Options Sets
Elastic IPs
Route tables (1/4) Info
Actions
Create route table
Filter route tables
Name
Route table ID
Explicit subnet associat...
Edge associations
Main
VPC
– rtb-84bc81fa – – Yes
PublicRouteTableTF rtb-0834938bb0cdfef5d 6 subnets – No
PrivateRouteTableTF rtb-0ab37bf3eaf552344 2 subnets – No
– rtb-0d8a86e76b4a5b4fb – – Yes
Destination
Target
Status
Propagated
10.0.0.0/16 local Active No
0.0.0.0/0 eni-006aea1d598aa29ad Active No
https://console.aws.amazon.com/vpc/home?region=us-east-1#
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
21°C ENG 08:41 12-11-2021


console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:
Services
Search for services, features, blogs, docs, and more [Alt+S]
Sony Sathish
N. Virginia
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Filter instances
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
– i-02b797fb7f4067452 Running t2.micro Initializing No alarms
Docker_Ubuntu i-063b78e904a2f473a Stopped t2.micro – No alarms
Instance: i-02b797fb7f4067452
Tags
Manage tags
Key
Value
name
NATByTF
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
21°C
ENG
08:41
12-11-2021