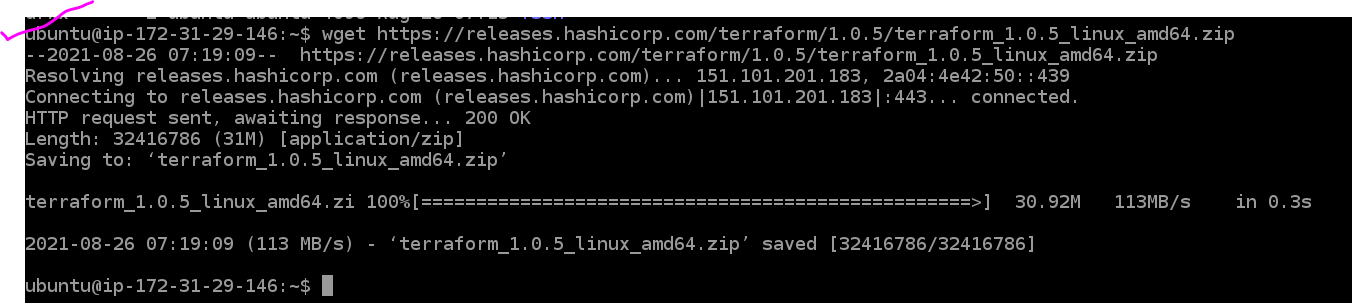
**Terraform :-**

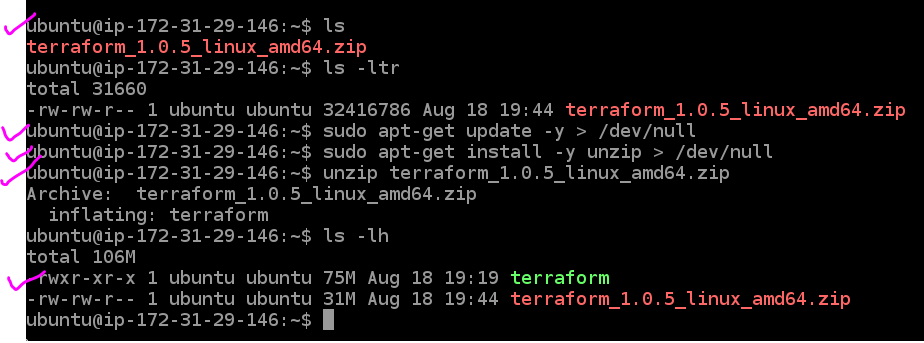
First provision an Ubuntu EC2 instance on AWS and connect to it using Putty.

Go to : <https://www.terraform.io/downloads.html> and select your OS. In my case I am selecting:-

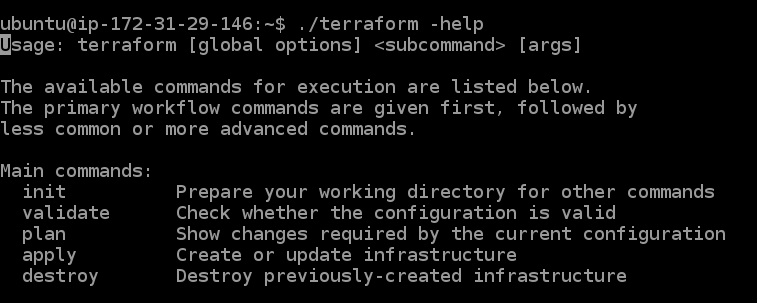
<https://releases.hashicorp.com/terraform/1.0.5/terraform_1.0.5_linux_amd64.zip>

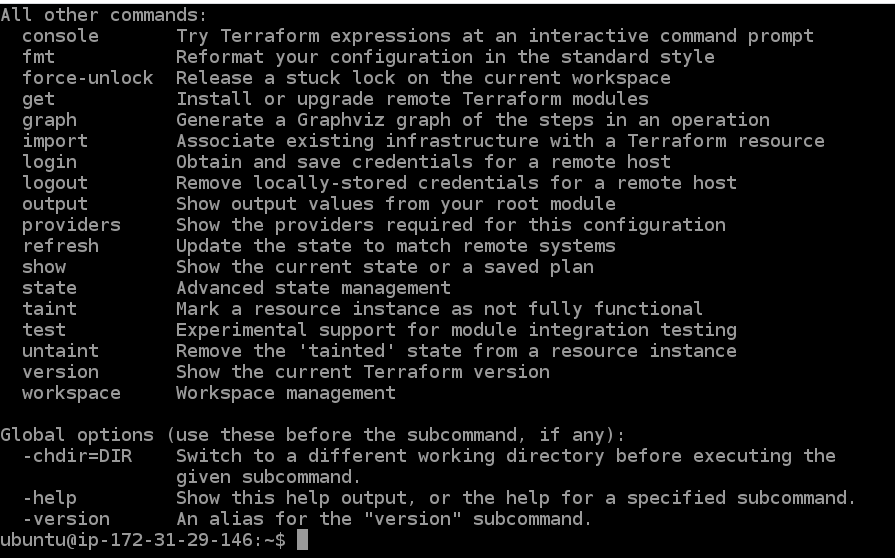


Run the below commands as highlighted:-

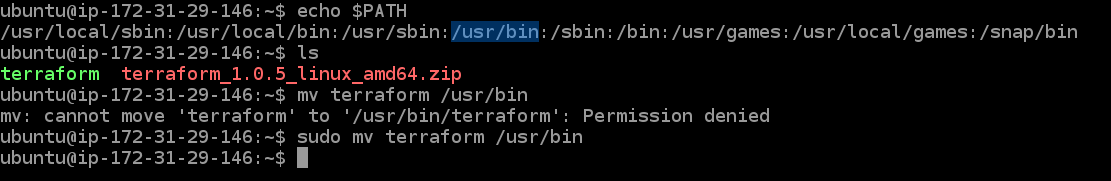


Invoking **terraform -help**



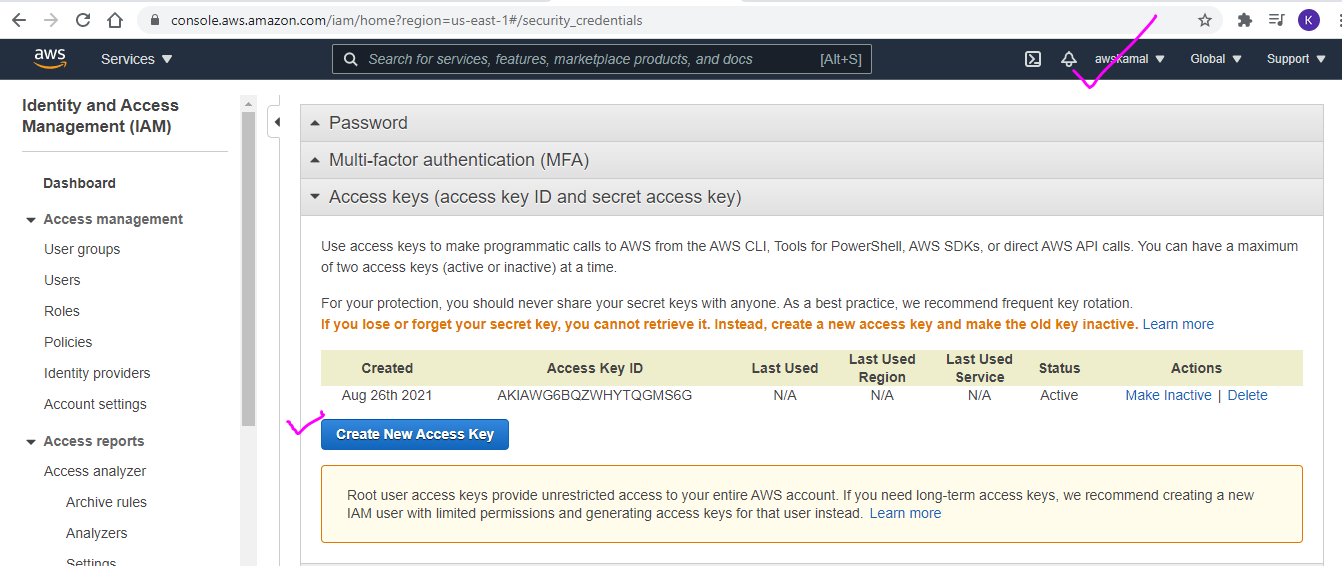


Moving the terraform file to PATH variable folders. Because we don’t want to execute terraform as ./terraform.

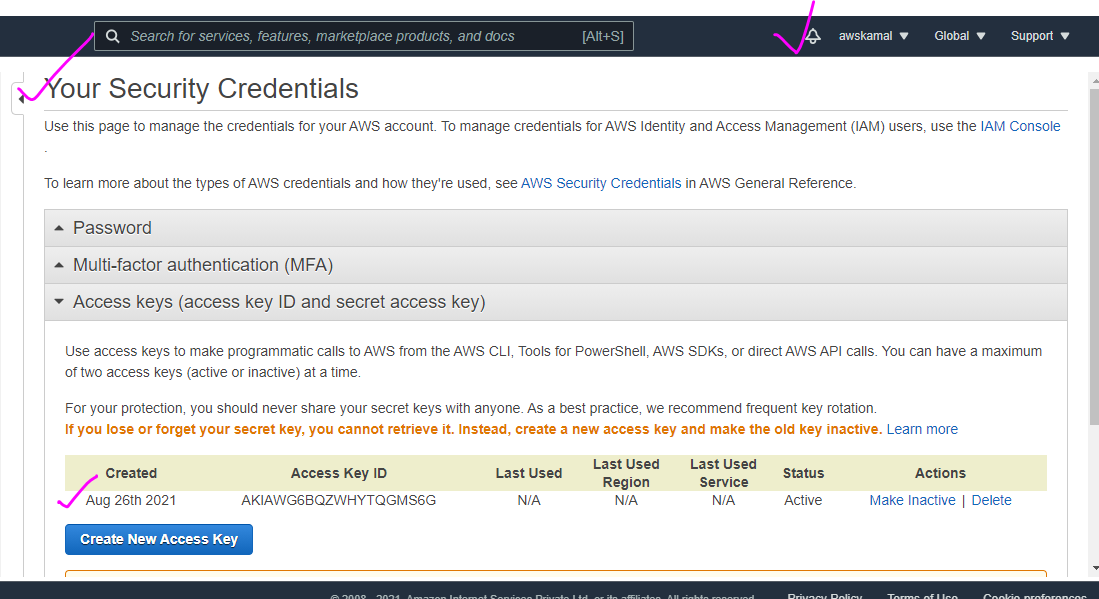


You would need your access key and secret access key from AWS:-

Go to your aws console > under your account > My Security Credentials

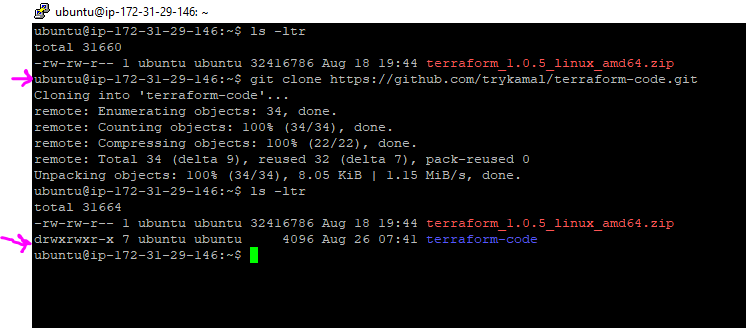


Create or use an existing Access key and secret access key

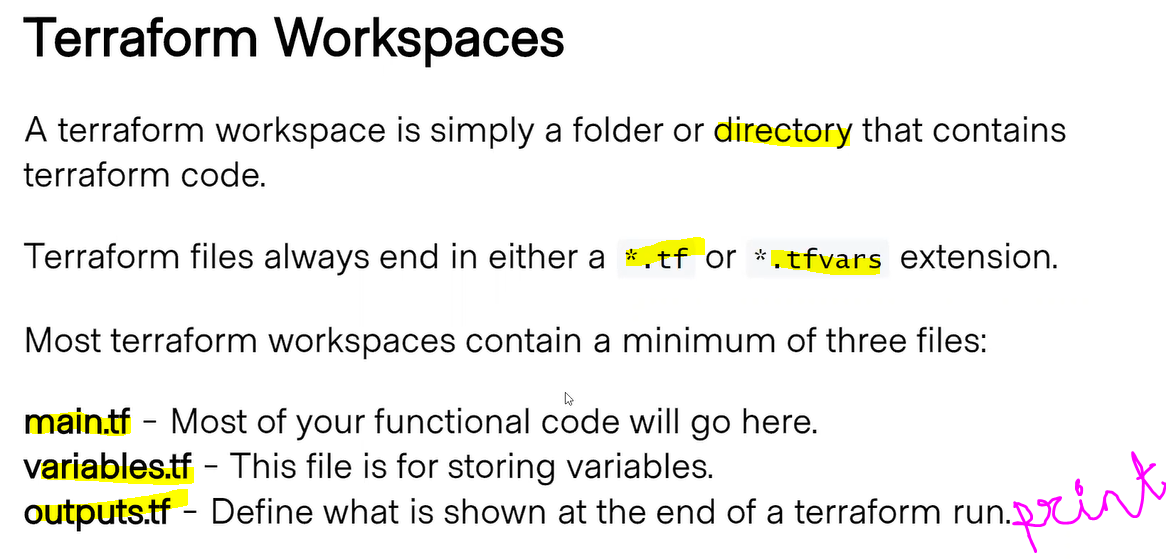


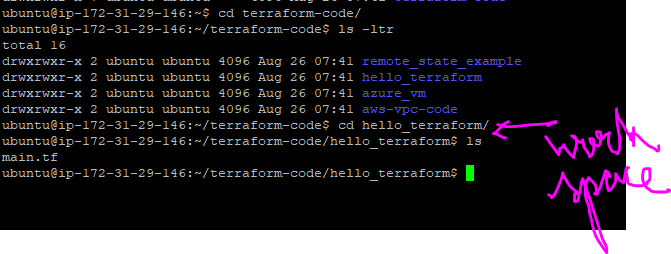
You can download the terraform code from my gitlab repo and run **git clone** command:

<https://github.com/trykamal/terraform-code.git>

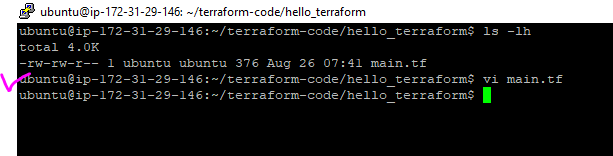


Notes:-

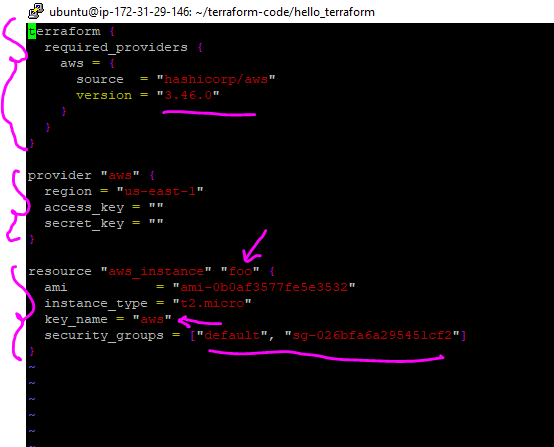




Lets go inside the simple code and understand:



Lets understand each block:



*terraform {*

*required\_providers {*

*aws = {*

*source = "hashicorp/aws"*

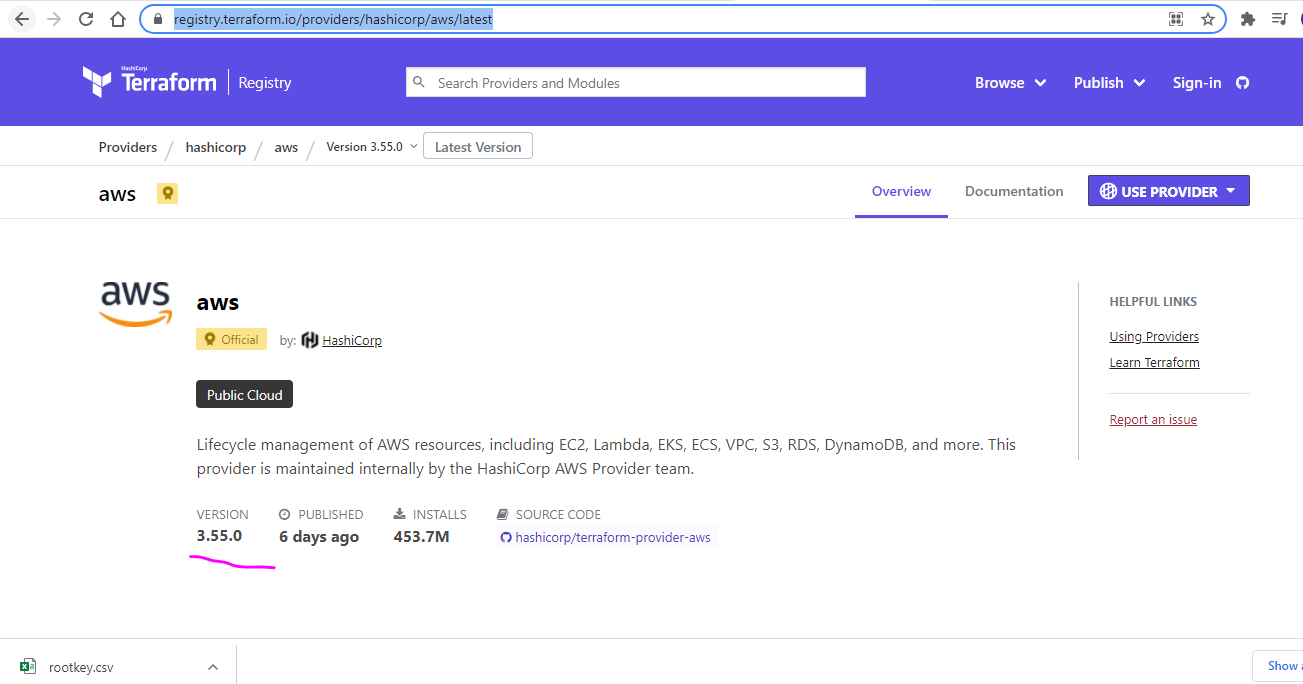
*version = "3.46.0"*

*}*

*}*

*}*

* This is for terraform initialization. Meaning we are telling terraform that we want to provision AWS resources. This is mandatory.
* You can get “latest” version from <https://registry.terraform.io/providers/hashicorp/aws/latest>



*provider "aws" {*

*region = "us-east-1"*

*access\_key = ""*

*secret\_key = ""*

*}*

* Provide the provider information = aws
* Region = any region (not availability zone) of your choice
* Access key = which you downloaded earlier
* Secret\_key = as part of access key

*resource "aws\_instance" "foo" {*

*ami = "ami-0b0af3577fe5e3532"*

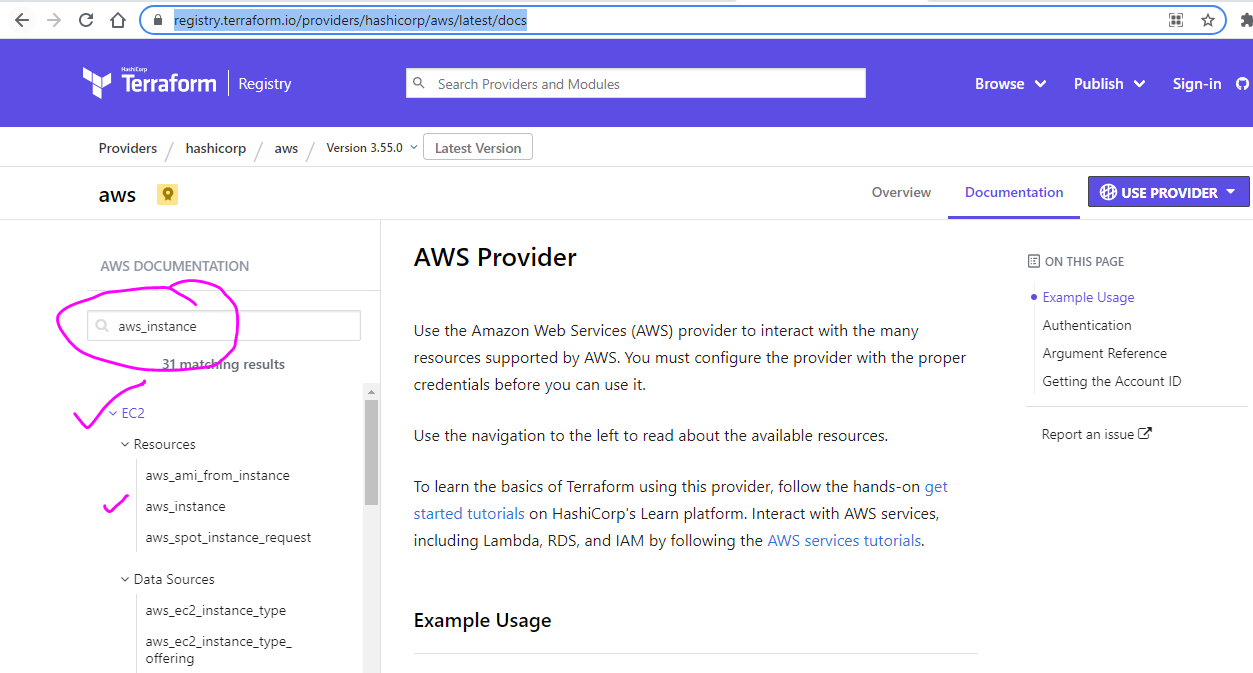
*instance\_type = "t2.micro"*

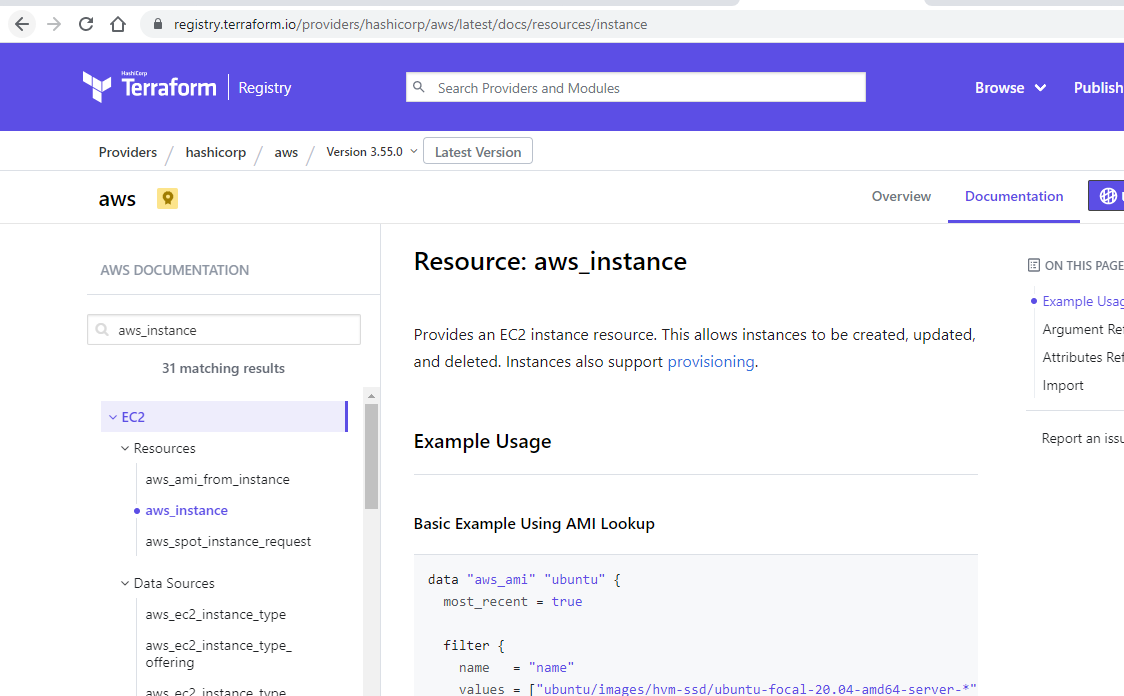
*key\_name = "aws"*

*security\_groups = ["default", "sg-026bfa6a295451cf2"]*

*}*

Resource = aws\_instance ( for ec2, for others like S3/vpc/alb/etc you refer to terraform documentation. <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>)





resource "aws\_instance" "foo" {

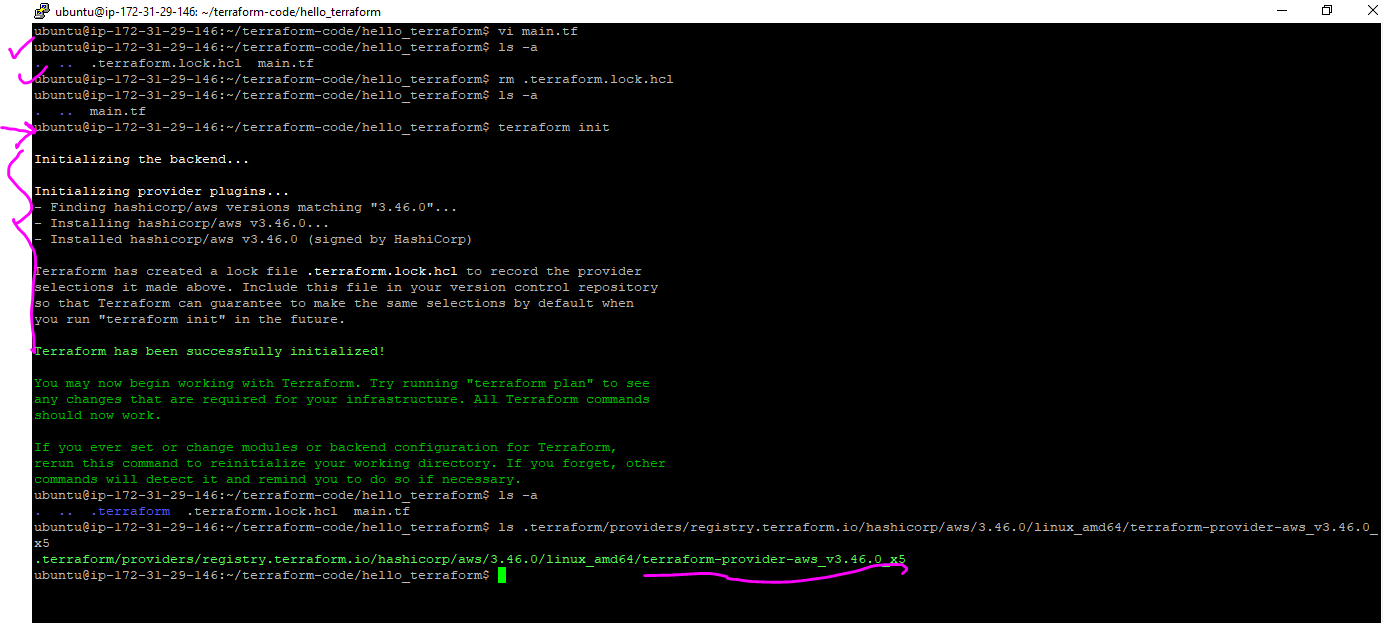
* Foo is just a name , you can have whatever you want. Note, its not a tag.
* ami = "ami-0b0af3577fe5e3532" -- the ami you want
* instance\_type = "t2.micro" -> type of instance
* key\_name = "aws" -> the .ppk file name or the key that you have downloaded earlier. In my case it will be NVkeyPairLenovo
* security\_groups = ["default", "sg-026bfa6a295451cf2"]
* }

The first step is to run => terraform init.

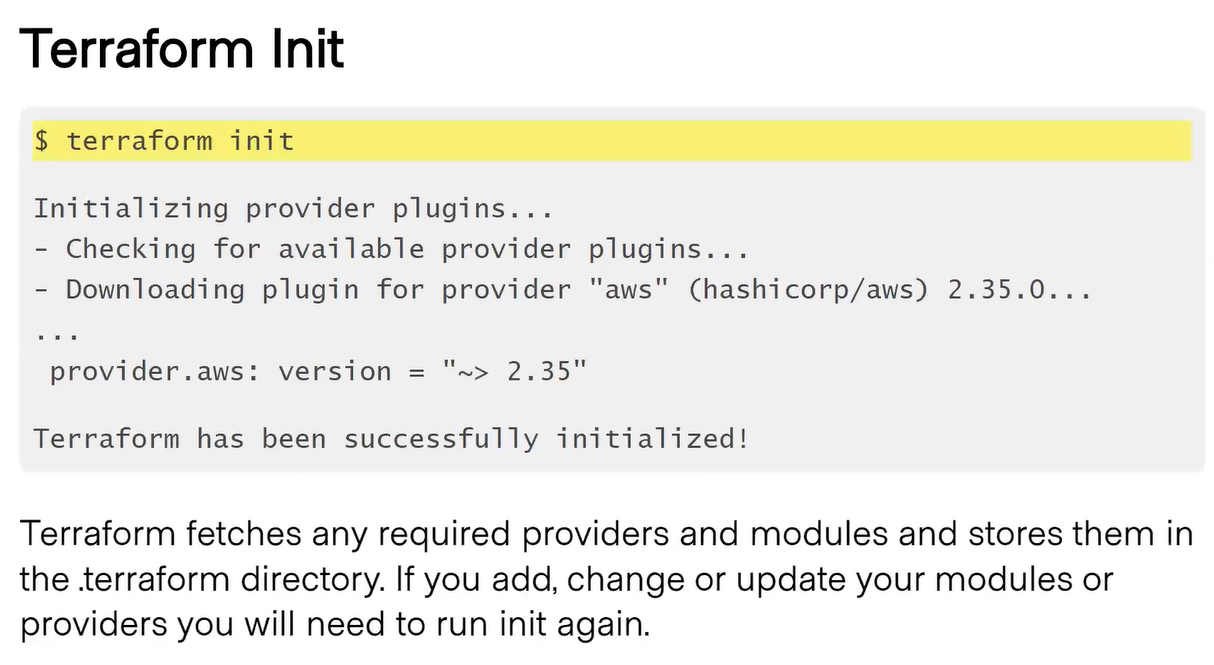
\*\* remember to delete the file .terraform.lock.hcl

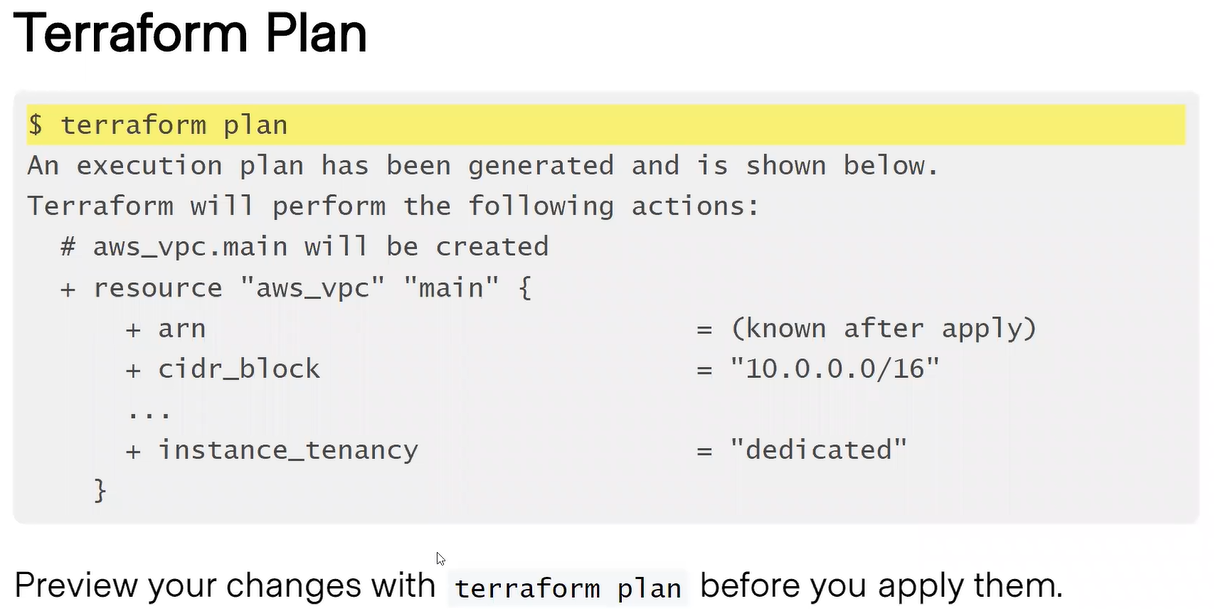
Else terraform init will not work properly

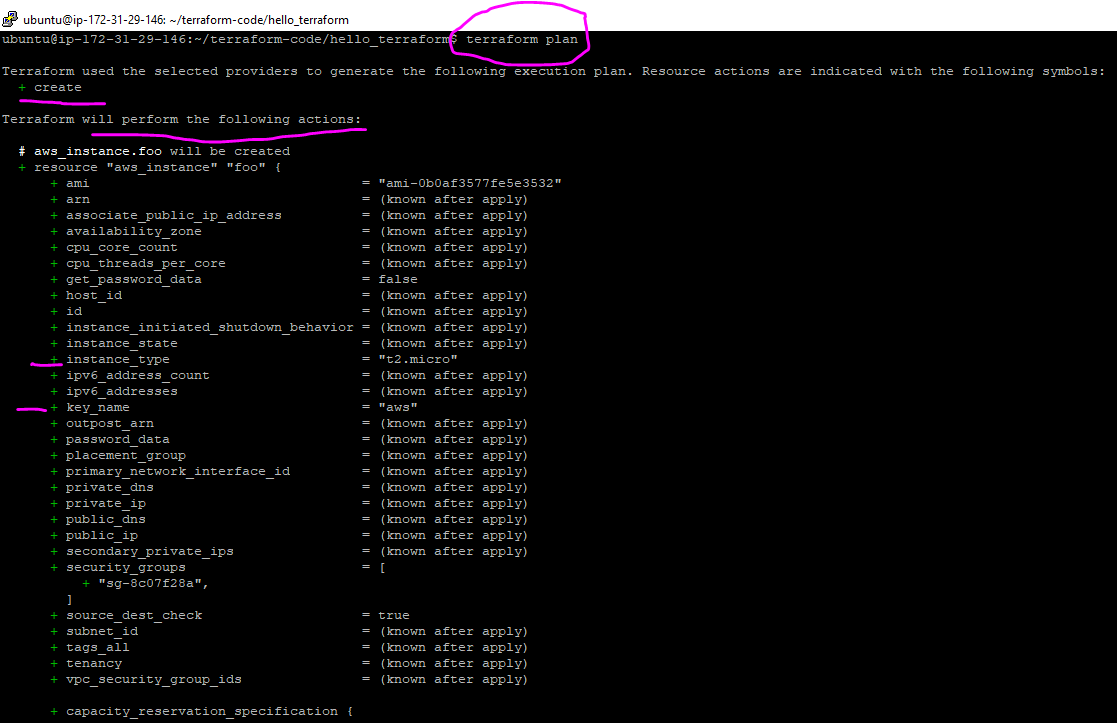
Understand that the mail.tf is readable to you but not to aws , so terraform init will download all you need. In particular the file terraform-provider-aws\_v3.46.0\_x5

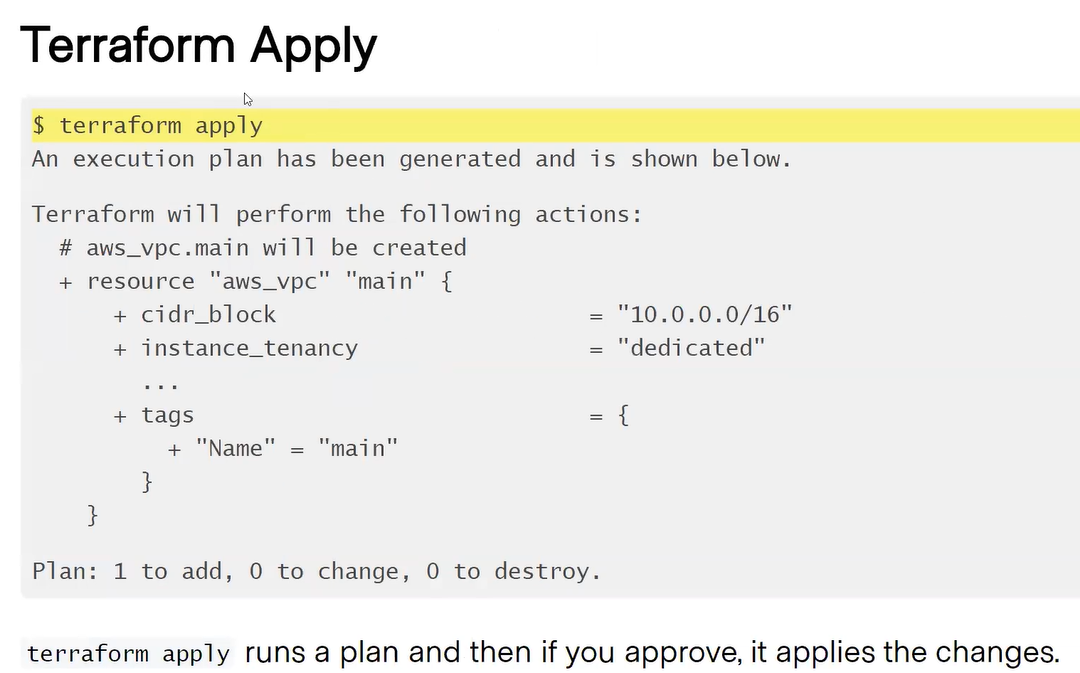


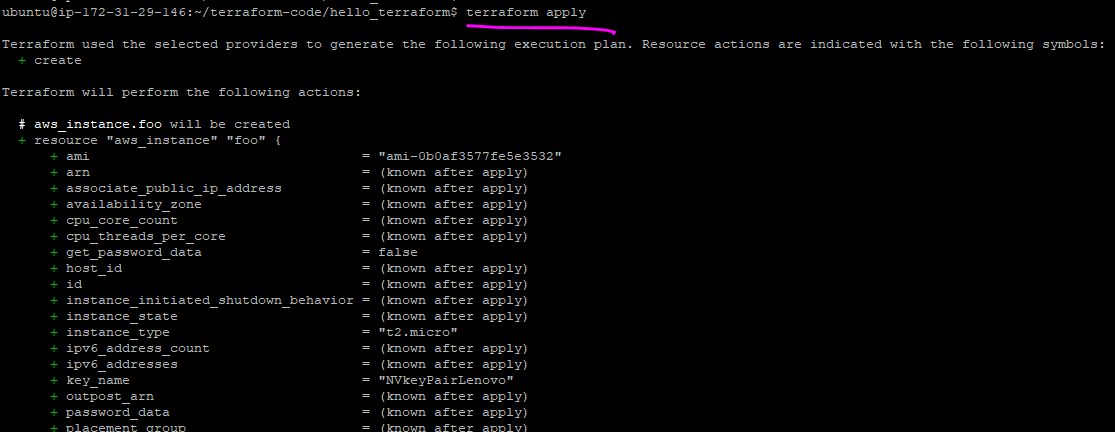




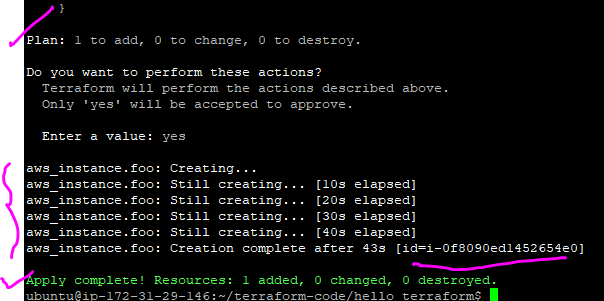




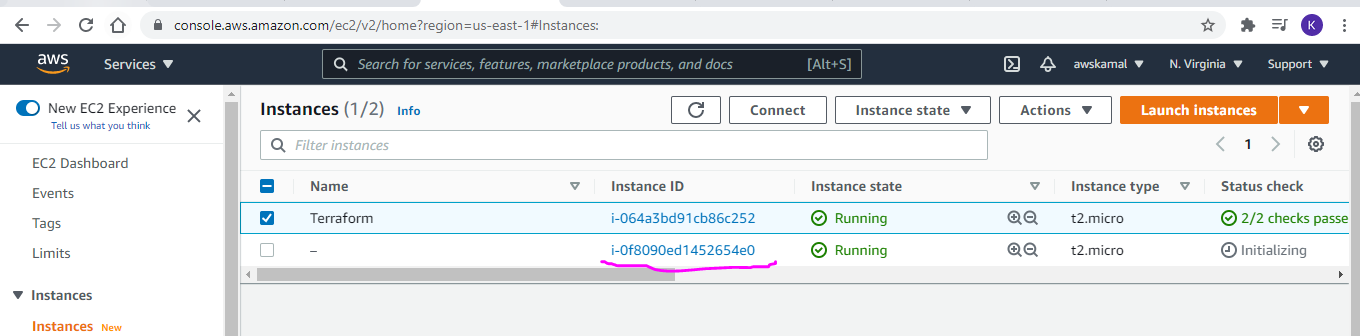




It will ask one time “yes” which you have to type in.

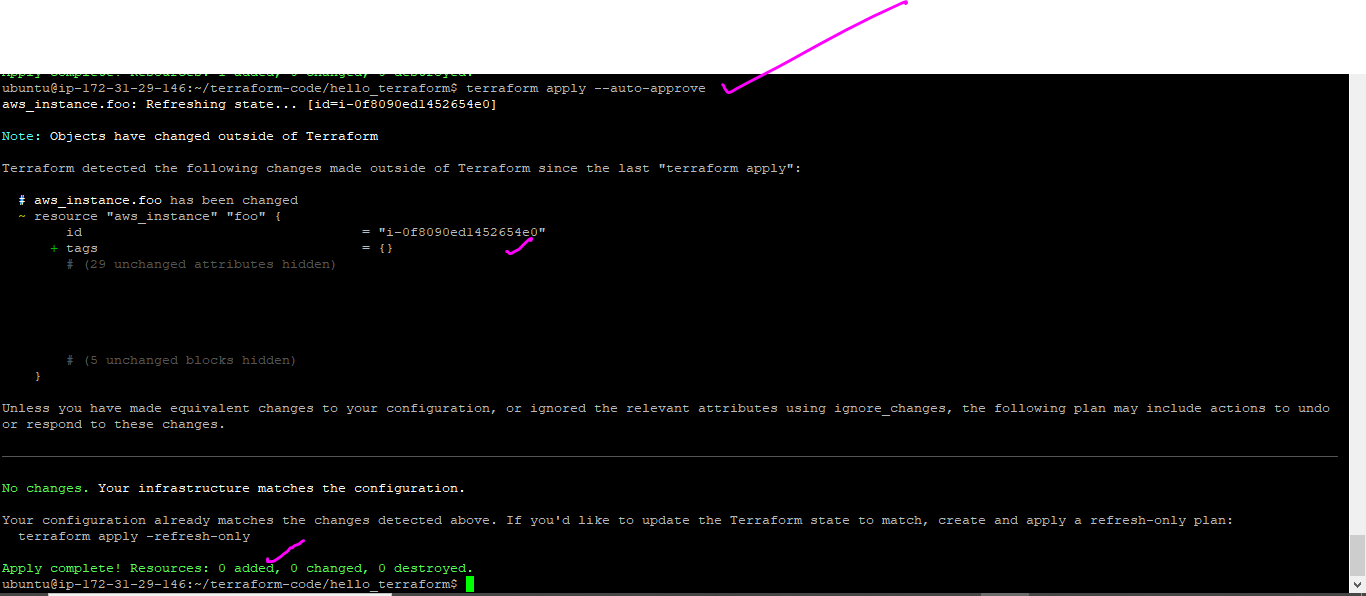


See that the resource has been added to EC2 dashboard.

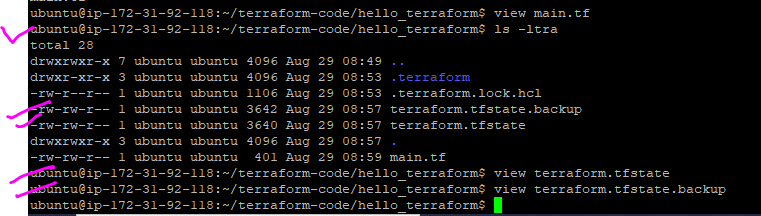


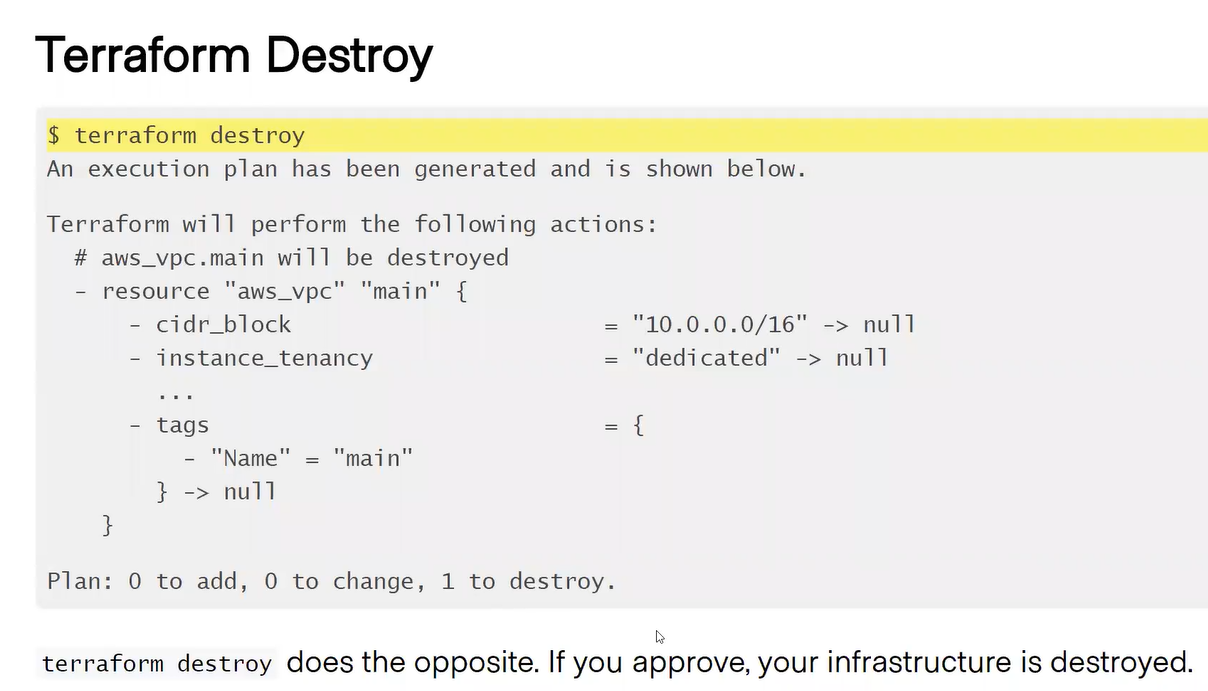
ubuntu@ip-172-31-29-146:~/terraform-code/hello\_terraform$ **terraform apply --auto-approve**

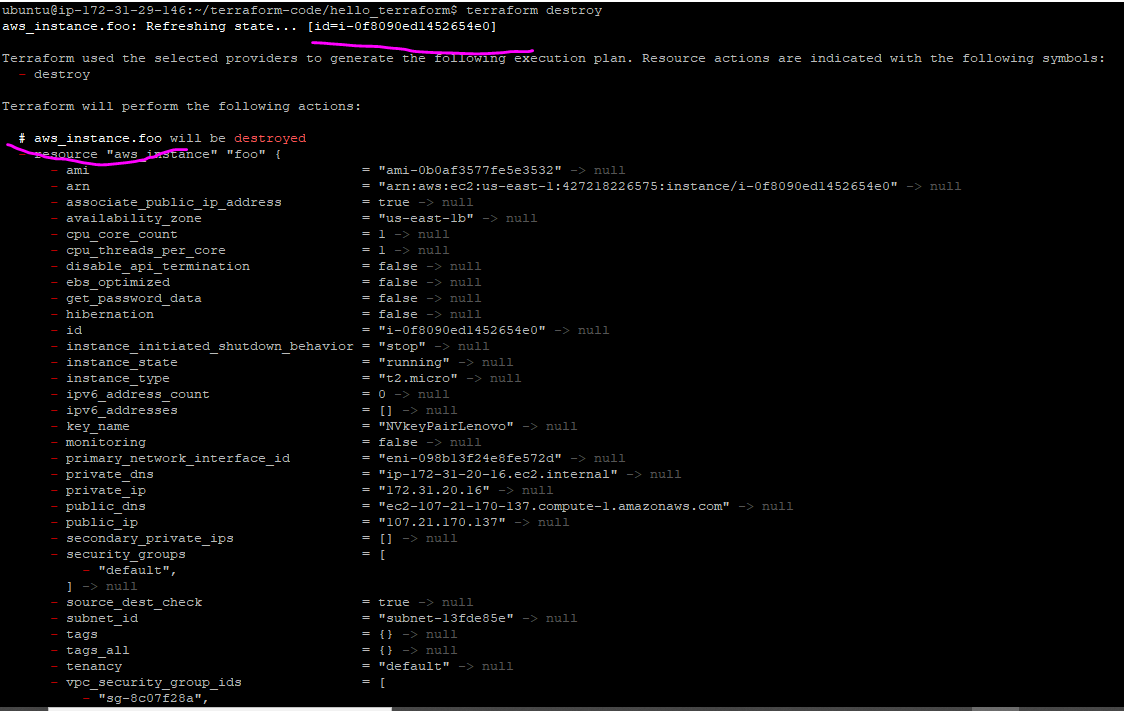
The above command will not ask for prompt. Apply command always run the plan before executing.

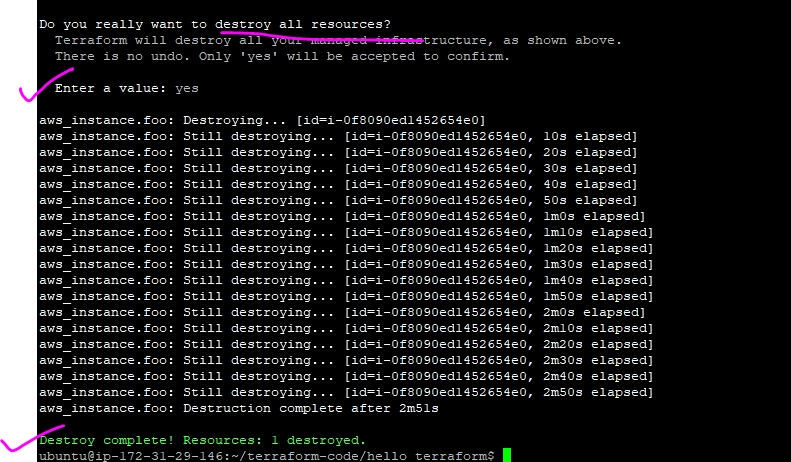


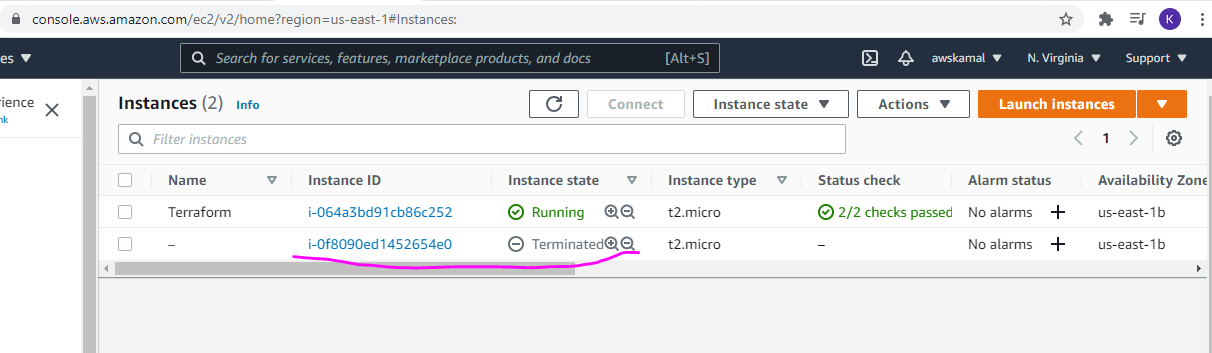
Check the terraform.tfstate file. It has information on the new created instance. You can check the instance id in the terraform.tfstate file.











Terraform validate

* This command “terraform validate” will validate your .tf file for any sysntax error

Terraform fmt

* This command will beautify your code only



\*\* the above code is not formatted

Below code is not beautified or formatted:



"terraform.tfstate"

<https://www.terraform.io/docs/language/state/index.html>

This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.

This state is stored by default in a local file named "terraform.tfstate", but it can also be stored remotely, which works better in a team environment.

In short – Your terraform.tfstate is the file which has the metadata of what it has done so far. If you run terraform apply then this file terraform.tfstate will check if the resource is pre provisioned or not and if not it will provision again else do nothing. Meaning it is the master which does all the checking of provisioning before doing any activity. If you delete the file then terraform will create again and again as it doesn’t know if the resource is already provisioned or not.

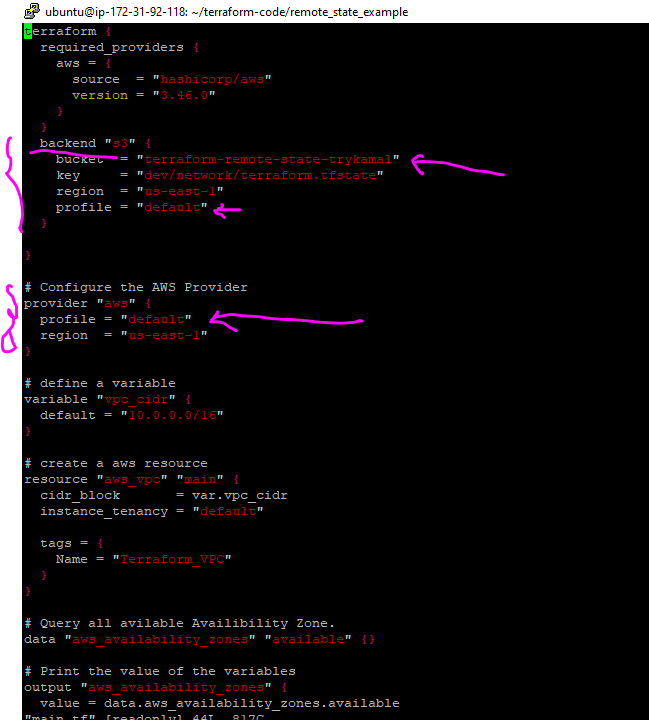
So we keep the file in a safe location called **backend**: <https://www.terraform.io/docs/language/settings/backends/index.html>

This has a major drawback as we are providing the access key and secret key in the same file. Anyone having access to the file may create any number of resources given that the person has the roles associated to him/her.



How to overcome this ? We will see now.

Opening the main.tf file from “/terraform-code/remote\_state\_example” location on your putty



Few things to not before we move forward from the main.tf file.

backend "s3" {

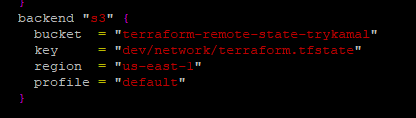
bucket = "terraform-remote-state-trykamal"

key = "dev/network/terraform.tfstate"

region = "us-east-1"

profile = "default" # you have to set your access/secret in home directory.

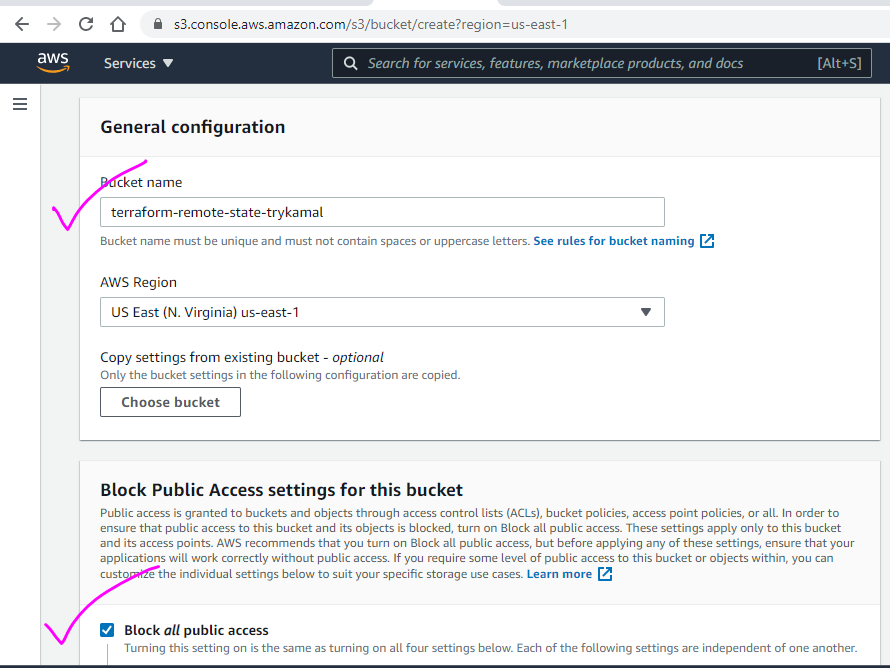
}

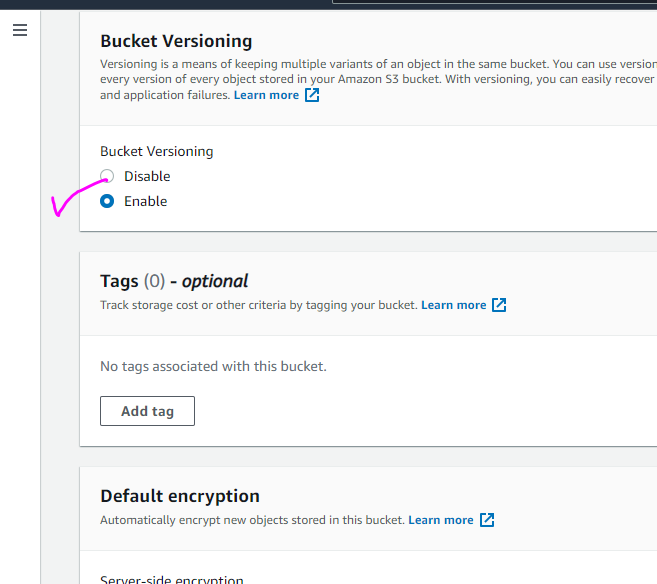


Create a S3 bucket name “terraform-remote-state-trykamal”.

Remember to Block Public access.

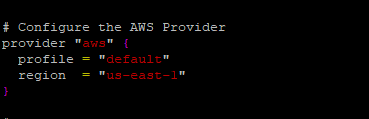
Enable versioning.



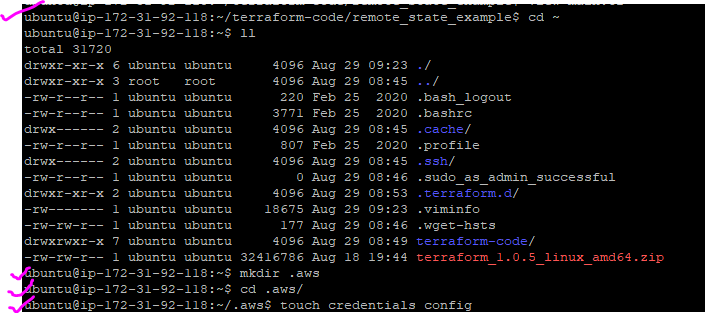


\*\* bucket created successfully.

Now coming back to main.tf file and specifically the provider section.



Note in the profile section we are not giving the access key and secret but instead we will configure in the location as follows.



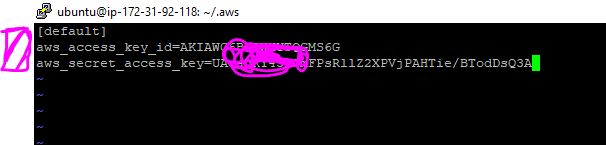
Step 1: go to home directory using cd ~ command

Step2: mkdir .aws

Step 3: cd .aws

Step4: create 2 file , touch credentials config

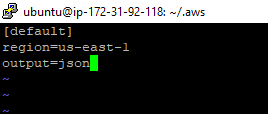
Step 5: vi credentials



The above is the default screenshot :

Check the link: <https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html>

Step 6: vi config

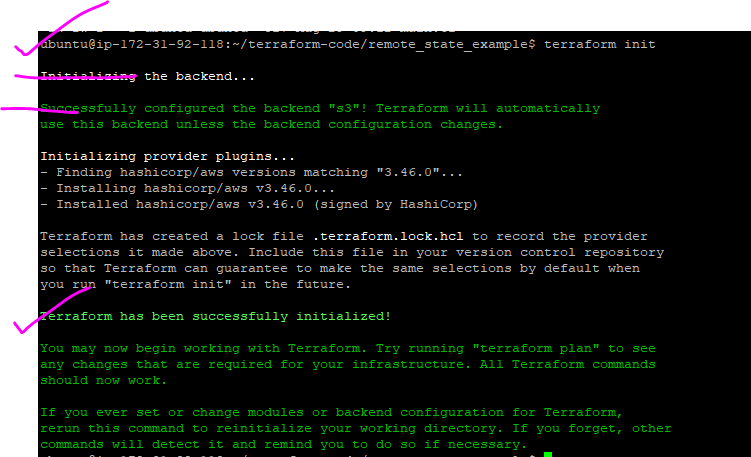


Step 7 : go back to your main.tf code directory

ubuntu@ip-172-31-92-118:~/terraform-code/remote\_state\_example$

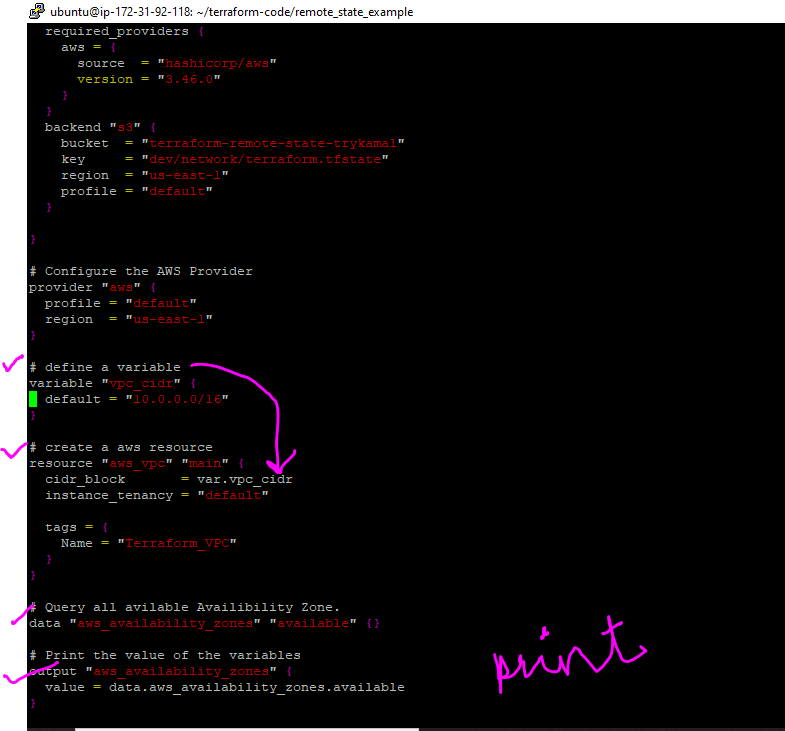
Step 8: run terraform init

If the connectivity is established between your S3 and aws then command will be successful



There are 4 main sections in the main.tf file namely:

Resource,variable,data,output

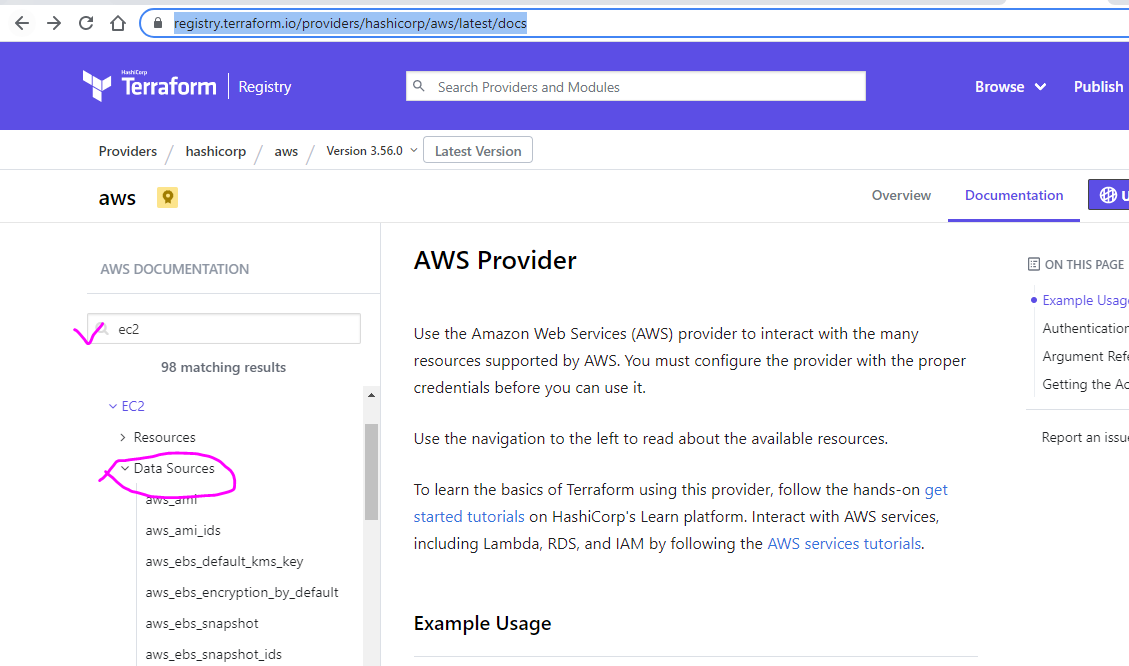


Resource = what you want to create

Variable = variable name and the value. See how the variable is references in the resource section. The syntax is var.<variable name> (eg var.vpc\_cidr) and the default (or value) will be passed. You can pass it as a list as well.

Data = fetching some specific info from the provider. Here I am fetching availability zone info.

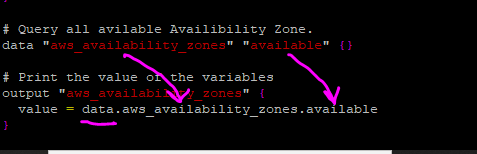
Go to : <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>



Eg: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/ebs_volume>

You can also provide filter condition to narrow down

Output : it is just the print statement. But see the syntax🡪 data.<data name>.<value>

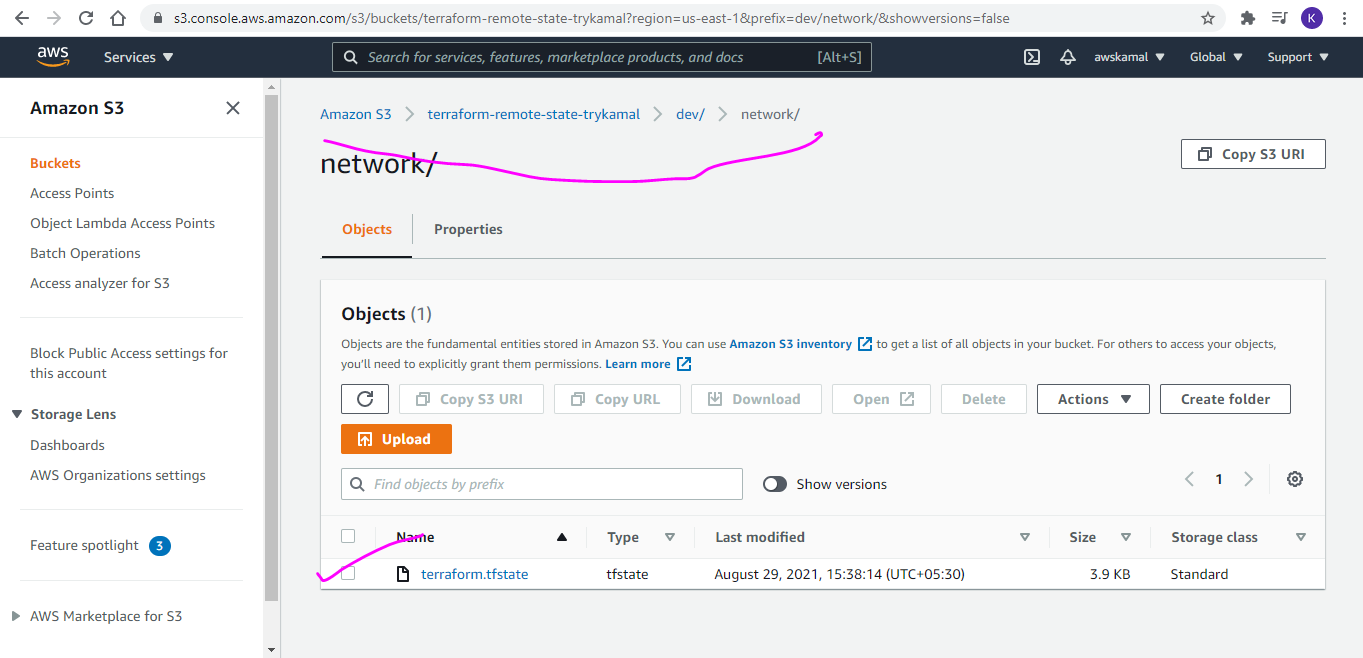


Run terraform plan to see the plan of what it is going to perform.

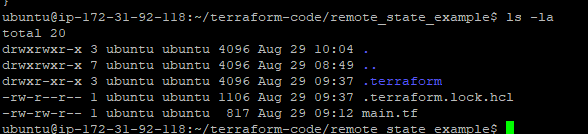
Run terraform apply

Observations:

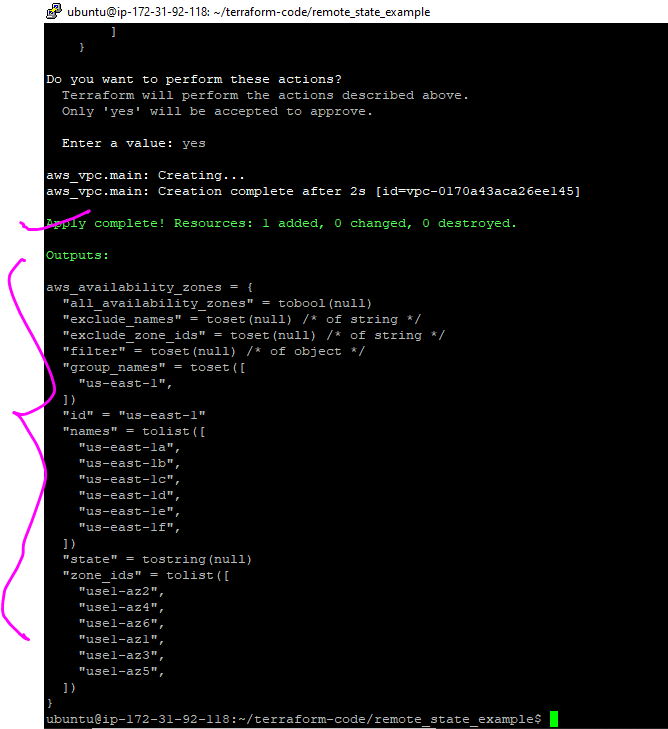
1. State file created remotely to your S3 bucket



Below screenshot doesn’t have terraform.tfstate file as it is stored in S3 bucket.

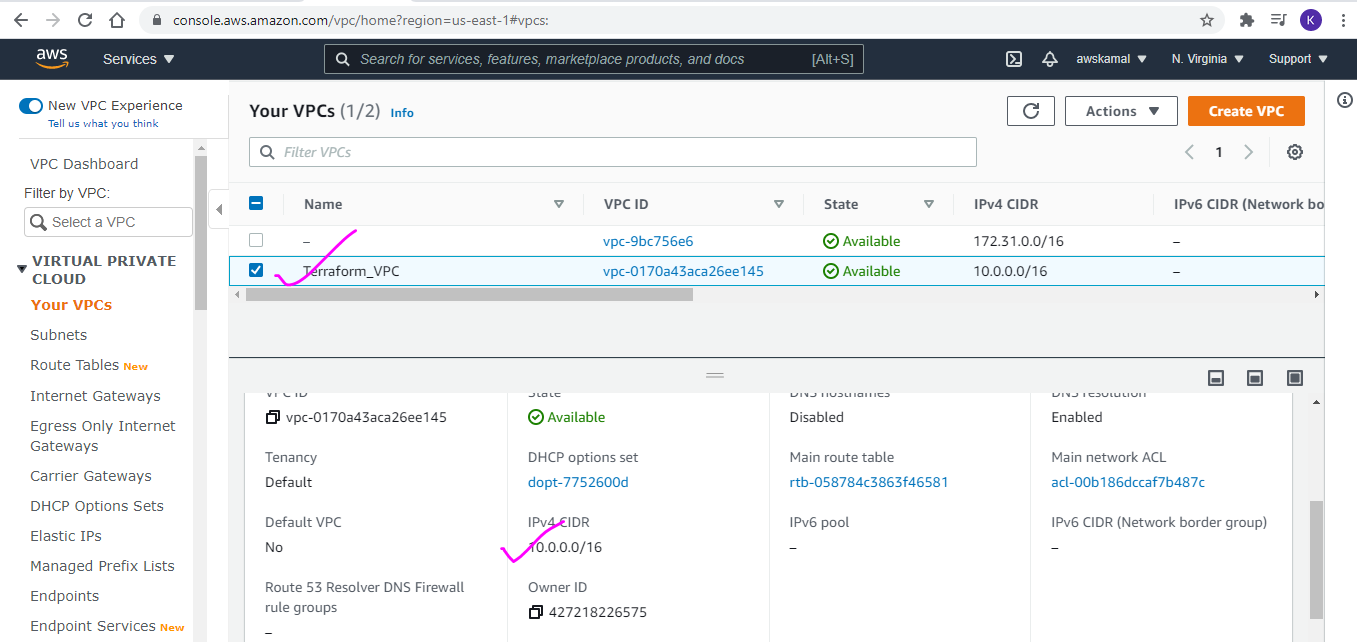


1. Output section has printed you the data



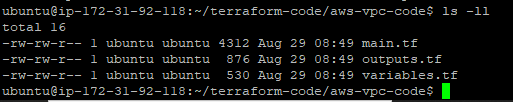
1. terraform import - This command can also create your terraform.tfstate file manually but there is lot of manual work and is not recommended.

See the result of the resource it created



Some complex code:

Industry standard



Main.tf

Output.tf

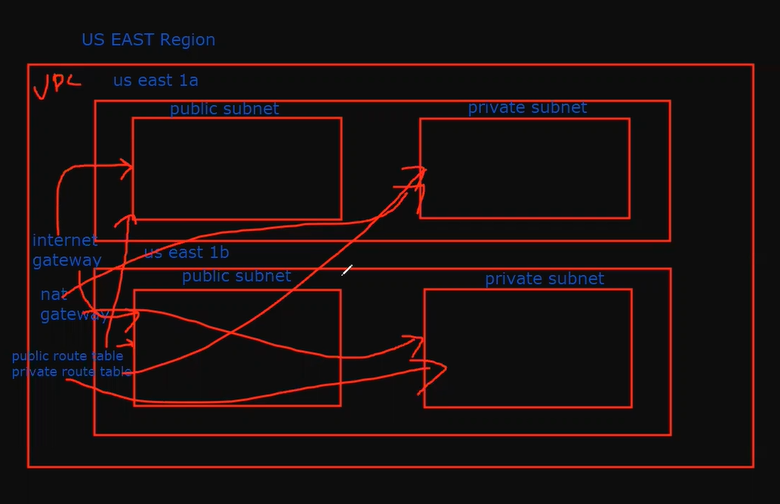
Variables.tf

Variables.tf

<https://www.hashicorp.com/blog/custom-variable-validation-in-terraform-0-13>

<https://www.terraform.io/docs/language/values/variables.html#custom-validation-rules>

Now you can validate your variables. See above link



We are creating a VPC in US EAST region

2 availability zones because of High availability architecture

2 subnets in each AZ ( 1 subnet for Public and 1 for Private)

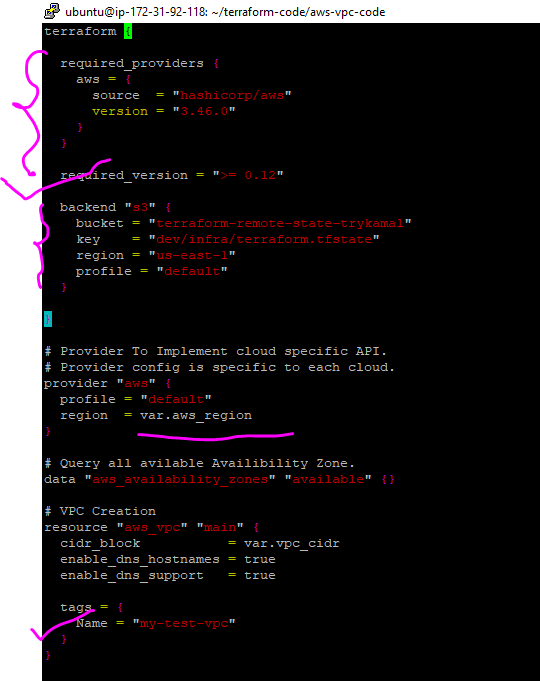
Internet gateway to connect to Public Subnet

NAT gateway to connect to Private Subnet

1 Public Route table to connect to Public Subnet

1 Private Route table to connect to Public Subnet

Vi main.tf



We have some mandatory fields like “required\_providers” & “required\_version”

Below is for my reference where the state file is stored.

backend "s3" {

bucket = "terraform-remote-state-trykamal"

key = "dev/infra/terraform.tfstate"

region = "us-east-1"

profile = "default"

}

Mandatory block:

# Provider To Implement cloud specific API.

# Provider config is specific to each cloud.

provider "aws" {

profile = "default"

region = var.aws\_region

}

# Query all avilable Availibility Zone.

data "aws\_availability\_zones" "available" {}

creating VPC. See the use of var.vpc\_cidr from the variables.tf file

# VPC Creation

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

enable\_dns\_hostnames = true

enable\_dns\_support = true

tags = {

Name = "my-test-vpc"

}

}

Check the internet gateway . we haven’t created the vpc but we have to attach the IG to VPC and see how we are using id to map the vpc. Meaning the id from above step can be used.

# Creating Internet Gateway

resource "aws\_internet\_gateway" "gw" {

vpc\_id = **aws\_vpc.main.id**

tags = {

Name = "my-test-igw"

}

}

# Public Route Table

resource "aws\_route\_table" "public\_route" {

vpc\_id = **aws\_vpc.main.id**

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = **aws\_internet\_gateway.gw.id**

}

tags = {

Name = "my-test-public-route"

}

# Private Route Table

resource "aws\_default\_route\_table" "private\_route" {

default\_route\_table\_id = **aws\_vpc.main.default\_route\_table\_id**

route {

nat\_gateway\_id = **aws\_nat\_gateway.my-test-nat-gateway.id #note we haven’t created any #nat gateway above but we can use it. Bcoz terraform will create a graph**

cidr\_block = "0.0.0.0/0"

}

tags = {

Name = "my-test-private-route"

}

}

# Public Subnet

resource "aws\_subnet" "public\_subnet" {

count = 2  **# For Loop. Create 2 subnets**

cidr\_block = var.public\_cidrs[count.index]

vpc\_id = aws\_vpc.main.id

map\_public\_ip\_on\_launch = true

availability\_zone = data.aws\_availability\_zones.available.names[count.index]

tags = {

Name = "my-test-public-subnet.${count.index + 1}" # Incrementing by 1

}

}

Terraform graph: terraform will create a diagraph and you can use the diagraph to view the pictorial representation. Use the site : <http://www.webgraphviz.com/>

ubuntu@ip-172-31-92-118:~/terraform-code/aws-vpc-code$ terraform graph

digraph {

compound = "true"

newrank = "true"

subgraph "root" {

"[root] aws\_default\_route\_table.private\_route (expand)" [label = "aws\_default\_route\_table.private\_route", shape = "box"]

"[root] aws\_eip.my-test-eip (expand)" [label = "aws\_eip.my-test-eip", shape = "box"]

"[root] aws\_internet\_gateway.gw (expand)" [label = "aws\_internet\_gateway.gw", shape = "box"]

"[root] aws\_nat\_gateway.my-test-nat-gateway (expand)" [label = "aws\_nat\_gateway.my-test-nat-gateway", shape = "box"]

"[root] aws\_route\_table.public\_route (expand)" [label = "aws\_route\_table.public\_route", shape = "box"]

"[root] aws\_route\_table\_association.private\_subnet\_assoc (expand)" [label = "aws\_route\_table\_association.private\_subnet\_assoc", shape = "box"]

"[root] aws\_route\_table\_association.public\_subnet\_assoc (expand)" [label = "aws\_route\_table\_association.public\_subnet\_assoc", shape = "box"]

"[root] aws\_security\_group.sg\_private (expand)" [label = "aws\_security\_group.sg\_private", shape = "box"]

"[root] aws\_security\_group.test\_sg (expand)" [label = "aws\_security\_group.test\_sg", shape = "box"]

"[root] aws\_security\_group\_rule.all\_outbound\_access (expand)" [label = "aws\_security\_group\_rule.all\_outbound\_access", shape = "box"]

"[root] aws\_security\_group\_rule.http\_inbound\_access (expand)" [label = "aws\_security\_group\_rule.http\_inbound\_access", shape = "box"]

"[root] aws\_security\_group\_rule.ssh\_inbound\_access (expand)" [label = "aws\_security\_group\_rule.ssh\_inbound\_access", shape = "box"]

"[root] aws\_subnet.private\_subnet (expand)" [label = "aws\_subnet.private\_subnet", shape = "box"]

"[root] aws\_subnet.public\_subnet (expand)" [label = "aws\_subnet.public\_subnet", shape = "box"]

"[root] aws\_vpc.main (expand)" [label = "aws\_vpc.main", shape = "box"]

"[root] data.aws\_availability\_zones.available (expand)" [label = "data.aws\_availability\_zones.available", shape = "box"]

"[root] output.aws\_availability\_zones" [label = "output.aws\_availability\_zones", shape = "note"]

"[root] output.private\_security\_group" [label = "output.private\_security\_group", shape = "note"]

"[root] output.private\_subnet1" [label = "output.private\_subnet1", shape = "note"]

"[root] output.private\_subnet\_ids" [label = "output.private\_subnet\_ids", shape = "note"]

"[root] output.private\_subnets" [label = "output.private\_subnets", shape = "note"]

"[root] output.public\_security\_group" [label = "output.public\_security\_group", shape = "note"]

"[root] output.public\_subnet1" [label = "output.public\_subnet1", shape = "note"]

"[root] output.public\_subnets" [label = "output.public\_subnets", shape = "note"]

"[root] output.vpc\_id" [label = "output.vpc\_id", shape = "note"]

"[root] provider[\"registry.terraform.io/hashicorp/aws\"]" [label = "provider[\"registry.terraform.io/hashicorp/aws\"]", shape = "diamond"]

"[root] var.aws\_region" [label = "var.aws\_region", shape = "note"]

"[root] var.ingress\_ports" [label = "var.ingress\_ports", shape = "note"]

"[root] var.private\_cidrs" [label = "var.private\_cidrs", shape = "note"]

"[root] var.public\_cidrs" [label = "var.public\_cidrs", shape = "note"]

"[root] var.tag\_name" [label = "var.tag\_name", shape = "note"]

"[root] var.vpc\_cidr" [label = "var.vpc\_cidr", shape = "note"]

"[root] aws\_default\_route\_table.private\_route (expand)" -> "[root] aws\_nat\_gateway.my-test-nat-gateway (expand)"

"[root] aws\_eip.my-test-eip (expand)" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"]"

"[root] aws\_internet\_gateway.gw (expand)" -> "[root] aws\_vpc.main (expand)"

"[root] aws\_nat\_gateway.my-test-nat-gateway (expand)" -> "[root] aws\_eip.my-test-eip (expand)"

"[root] aws\_nat\_gateway.my-test-nat-gateway (expand)" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] aws\_route\_table.public\_route (expand)" -> "[root] aws\_internet\_gateway.gw (expand)"

"[root] aws\_route\_table\_association.private\_subnet\_assoc (expand)" -> "[root] aws\_default\_route\_table.private\_route (expand)"

"[root] aws\_route\_table\_association.private\_subnet\_assoc (expand)" -> "[root] aws\_subnet.private\_subnet (expand)"

"[root] aws\_route\_table\_association.public\_subnet\_assoc (expand)" -> "[root] aws\_route\_table.public\_route (expand)"

"[root] aws\_route\_table\_association.public\_subnet\_assoc (expand)" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] aws\_security\_group.sg\_private (expand)" -> "[root] aws\_vpc.main (expand)"

"[root] aws\_security\_group.sg\_private (expand)" -> "[root] var.ingress\_ports"

"[root] aws\_security\_group.test\_sg (expand)" -> "[root] aws\_vpc.main (expand)"

"[root] aws\_security\_group\_rule.all\_outbound\_access (expand)" -> "[root] aws\_security\_group.test\_sg (expand)"

"[root] aws\_security\_group\_rule.http\_inbound\_access (expand)" -> "[root] aws\_security\_group.test\_sg (expand)"

"[root] aws\_security\_group\_rule.ssh\_inbound\_access (expand)" -> "[root] aws\_security\_group.test\_sg (expand)"

"[root] aws\_subnet.private\_subnet (expand)" -> "[root] aws\_vpc.main (expand)"

"[root] aws\_subnet.private\_subnet (expand)" -> "[root] data.aws\_availability\_zones.available (expand)"

"[root] aws\_subnet.private\_subnet (expand)" -> "[root] var.private\_cidrs"

"[root] aws\_subnet.public\_subnet (expand)" -> "[root] aws\_vpc.main (expand)"

"[root] aws\_subnet.public\_subnet (expand)" -> "[root] data.aws\_availability\_zones.available (expand)"

"[root] aws\_subnet.public\_subnet (expand)" -> "[root] var.public\_cidrs"

"[root] aws\_vpc.main (expand)" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"]"

"[root] aws\_vpc.main (expand)" -> "[root] var.vpc\_cidr"

"[root] data.aws\_availability\_zones.available (expand)" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"]"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] aws\_route\_table\_association.private\_subnet\_assoc (expand)"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] aws\_route\_table\_association.public\_subnet\_assoc (expand)"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] aws\_security\_group\_rule.all\_outbound\_access (expand)"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] aws\_security\_group\_rule.http\_inbound\_access (expand)"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] aws\_security\_group\_rule.ssh\_inbound\_access (expand)"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.aws\_availability\_zones"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.private\_security\_group"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.private\_subnet1"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.private\_subnet\_ids"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.private\_subnets"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.public\_security\_group"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.public\_subnet1"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.public\_subnets"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] output.vpc\_id"

"[root] meta.count-boundary (EachMode fixup)" -> "[root] var.tag\_name"

"[root] output.aws\_availability\_zones" -> "[root] data.aws\_availability\_zones.available (expand)"

"[root] output.private\_security\_group" -> "[root] aws\_security\_group.sg\_private (expand)"

"[root] output.private\_subnet1" -> "[root] aws\_subnet.private\_subnet (expand)"

"[root] output.private\_subnet\_ids" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] output.private\_subnets" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] output.public\_security\_group" -> "[root] aws\_security\_group.test\_sg (expand)"

"[root] output.public\_subnet1" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] output.public\_subnets" -> "[root] aws\_subnet.public\_subnet (expand)"

"[root] output.vpc\_id" -> "[root] aws\_vpc.main (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_route\_table\_association.private\_subnet\_assoc (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_route\_table\_association.public\_subnet\_assoc (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_security\_group.sg\_private (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_security\_group\_rule.all\_outbound\_access (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_security\_group\_rule.http\_inbound\_access (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws\_security\_group\_rule.ssh\_inbound\_access (expand)"

"[root] provider[\"registry.terraform.io/hashicorp/aws\"]" -> "[root] var.aws\_region"

"[root] root" -> "[root] meta.count-boundary (EachMode fixup)"

"[root] root" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)"

}

}

ubuntu@ip-172-31-92-118:~/terraform-code/aws-vpc-code$