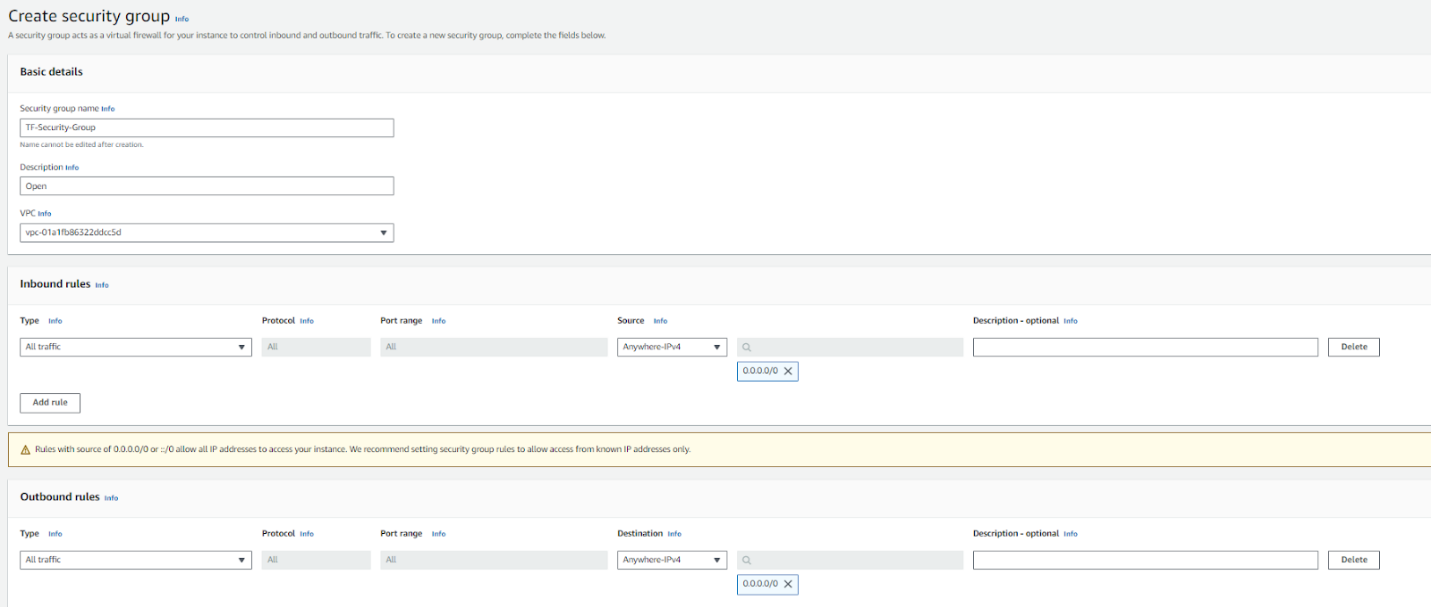
**Terraform**

**Installation**

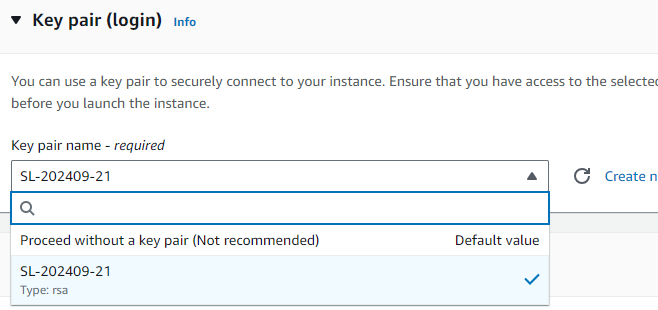
Setup TF Controller server

**Create Security Group**

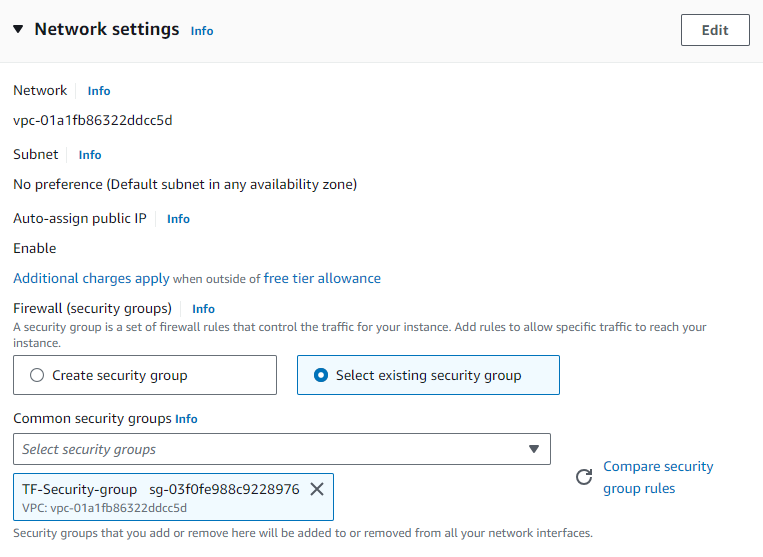


**Create Server**

Create New Keypair



Choose the created Security Group



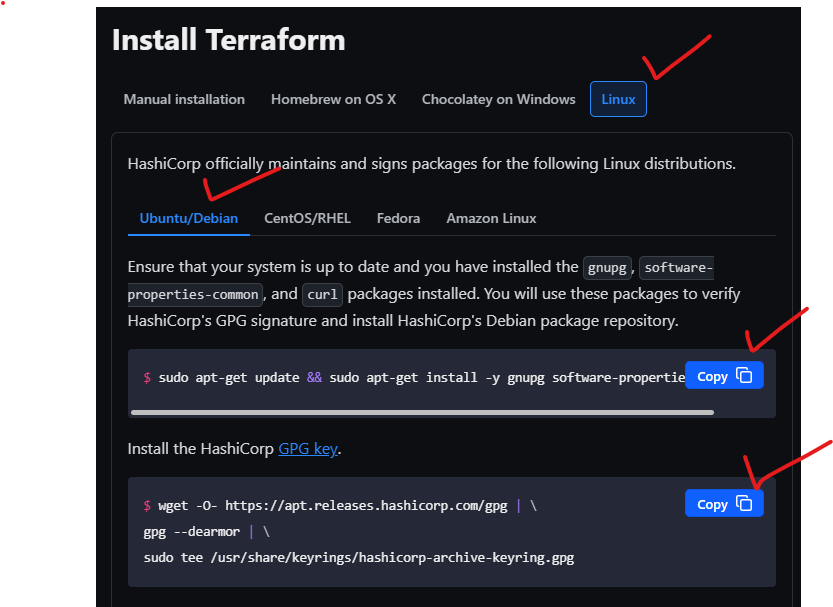
Click On Create

Install Terraform

Follow the Installation guide from the URL

<https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>.

*Note: Please choose Linux Tab in the guide*

**

*Note: If you encounter below error, run the suggested commands*

|  |
| --- |
| E: Could not get lock /var/lib/dpkg/lock-frontend. It is held by process 7809 (unattended-upgr)  N: Be aware that removing the lock file is not a solution and may break your system.  E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it? |

Command:

|  |
| --- |
| sudo rm -f /var/lib/dpkg/lock-frontend  sudo rm -f /var/lib/dpkg/lock |

Demo

**Provide Credentials to TF**

Execute below command to so that TF can authenticate to AWS

|  |
| --- |
| export AWS\_ACCESS\_KEY\_ID="Access-key-id" export AWS\_SECRET\_ACCESS\_KEY="secret-key" export AWS\_REGION="us-east-1" |

Create a terraform directory

 (e.g. ~/tf-demo)

|  |
| --- |
| mkdir ~/tf-demo cd ~/tf-demo |

**Define Provider**

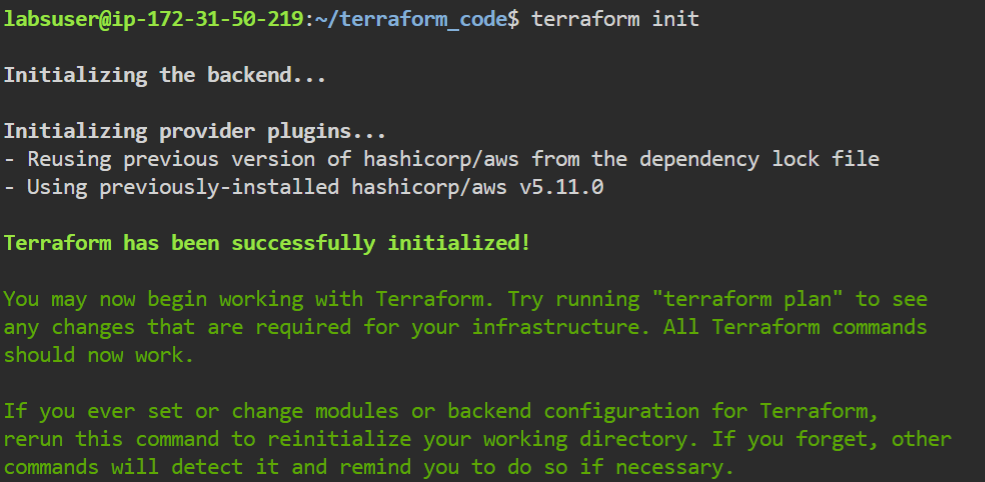
Create manifest in tf directory (e.g. ~/tf-demo/main.tf) with following content

|  |
| --- |
| terraform {   required\_providers {     aws = {       source = "hashicorp/aws"       version = "5.68.0"     }   } } provider "aws" {   # Configuration options } |

Initialize terraform

|  |
| --- |
| terraform init |

expected output:



**S3 Bucket**

Create a file ~/tf-demo/s3.tf with below content

|  |
| --- |
| resource "aws\_s3\_bucket" "example" {   bucket = "your-unique-bucket-name"    tags = {     Name        = "Demo"     Environment = "MyEnv"     Owner       = "Amit"   } } |

**EC2 Instance**

Create a file ~/tf-demo/ec2.tf with below content

|  |
| --- |
| resource "aws\_instance" "myfirstinstance" {   ami           = "ami-0e86e20dae9224db8"   instance\_type = "t2.micro" } |

**Apply the configuration**

|  |
| --- |
| terraform plan terraform apply #(type "yes" to accept) |

**Terraform commands of interest**

Format files for enhance readability

|  |
| --- |
| terraform fmt |

**State Management**

|  |
| --- |
| terraform show #Show all resources terraform state list #List all resource from statefile terraform state show aws\_s3\_bucket.example #List details of a particular resource |

**Resource References**

Create below Resources. Note that the aws\_instance will be referring to other two resources

**Create ssh keypair**

|  |
| --- |
| ssh-keygen -t rsa |

Create and apply below manifests from the TF directory

|  |
| --- |
| resource "aws\_instance" "myfirstinstance" {   ami             = "ami-0e86e20dae9224db8"   instance\_type   = "t2.micro"   security\_groups = [aws\_security\_group.my-sg-1.name]   key\_name        = aws\_key\_pair.deployer.key\_name } resource "aws\_key\_pair" "deployer" {   key\_name   = "deployer-key"   public\_key = file("/home/ubuntu/.ssh/id\_rsa.pub") } resource "aws\_security\_group" "my-sg-1" {   name = "securitygroup-from-tf"   ingress {     from\_port        = 0     to\_port          = 0     protocol         = "-1"     cidr\_blocks      = ["0.0.0.0/0"]     ipv6\_cidr\_blocks = ["::/0"]   }   egress {     from\_port        = 0     to\_port          = 0     protocol         = "-1"     cidr\_blocks      = ["0.0.0.0/0"]     ipv6\_cidr\_blocks = ["::/0"]   }   tags = {     Name = "securitygroup-from-tf"   } } |

**Other Providers**

Let’s take a look at tls and local provides

|  |
| --- |
| terraform {   required\_providers {     tls = {       source = "hashicorp/tls"       version = "4.0.6"     }     local = {       source = "hashicorp/local"       version = "2.5.2"     }    } } provider "aws" {   # Configuration options }  provider "tls" {   # Configuration options }  resource "tls\_private\_key" "rsa-key" {   algorithm = "RSA"   rsa\_bits  = 4096 } resource "local\_file" "pvt-key" {   content  = tls\_private\_key.rsa-key.private\_key\_pem   filename = "/home/ubuntu/.ssh/id\_rsa\_from\_tf" } resource "local\_file" "pub-key" {   content  = tls\_private\_key.rsa-key.public\_key\_openssh   filename = "/home/ubuntu/.ssh/id\_rsa\_from\_tf.pub" } |

**Data Source**

|  |
| --- |
| data "aws\_ami" "my-ami" {   filter {     name   = "name"     values = ["ubuntu/images/hvm-ssd/ubuntu-\*-22.04-amd64-server-\*"]   }   most\_recent = true   owners      = ["099720109477"] }  resource "aws\_instance" "data-source-instance" {   instance\_type = "t2.micro"   ami           = data.aws\_ami.my-ami.id   tags = {     Name = "data-source-instance"   } } |

**Variables and Locals**

|  |
| --- |
| variable "instance-type" {   default = "t2.micro" } variable "ami-id" {   default = "ami-053b0d53c279acc90" } locals {   tagname = "mytag" } resource "aws\_instance" "first-ec2" {   instance\_type = var.instance-type   ami           = var.ami-id   tags = {     Name = local.tagname   } } resource "aws\_instance" "second-ec2" {   instance\_type = var.instance-type   ami           = var.ami-id   tags = {     Name = local.tagname   } } |

The value of the variables can be overridden as follows

|  |
| --- |
| terraform apply -var instance-type=t3.micro |

NOTE: *The value of the locals* ***can not*** *be overridden*

**Multiple Resources**

**Count**

Add below code to multiple.tf

|  |
| --- |
| resource "aws\_instance" "multiple-instances" {   instance\_type = "t2.micro"   ami           = "ami-053b0d53c279acc90"   tags = {     Name = "multiple-${count.index}"   }   count = 3 } |

**for\_each**

|  |
| --- |
| variable "instance-list" {   default = ["dev", "test", "stage", "pre-prod", "prod"] }  resource "aws\_instance" "foreach" {   for\_each = toset(var.instance-list)   instance\_type = "t2.micro"   ami           = "ami-053b0d53c279acc90"   tags = {     Name = each.value   } } |