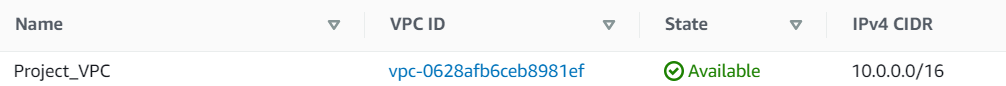
AWS Project

PART 1-Cloudformation

VPC Creation:

Creation of VPC



 "MyVPC": {

          "Type": "AWS::EC2::VPC",

          "Properties": {

            "CidrBlock":

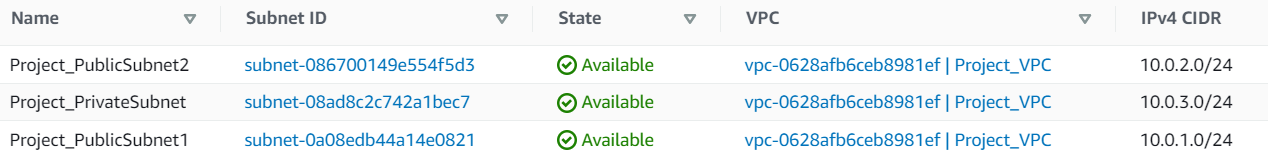
            { "Ref": "VpcCIDR" },

            "Tags": [{"Key":"Name","Value":"Project\_VPC"}]

          }

        },

Creation of Public & Private Subnet



"PublicSubnet1": {

          "Type": "AWS::EC2::Subnet",

          "Properties": {

            "AvailabilityZone": "us-east-1a",

            "VpcId": {"Ref": "MyVPC"},

            "CidrBlock":{"Ref":"PublicSubnet1CIDR"},

            "MapPublicIpOnLaunch": "true",

            "Tags": [{"Key":"Name","Value":"Project\_PublicSubnet1"}]

          }

        },

 "PrivateSubnet1": {

          "Type": "AWS::EC2::Subnet",

          "Properties": {

            "AvailabilityZone": "us-east-1b",

            "VpcId": { "Ref": "MyVPC" },

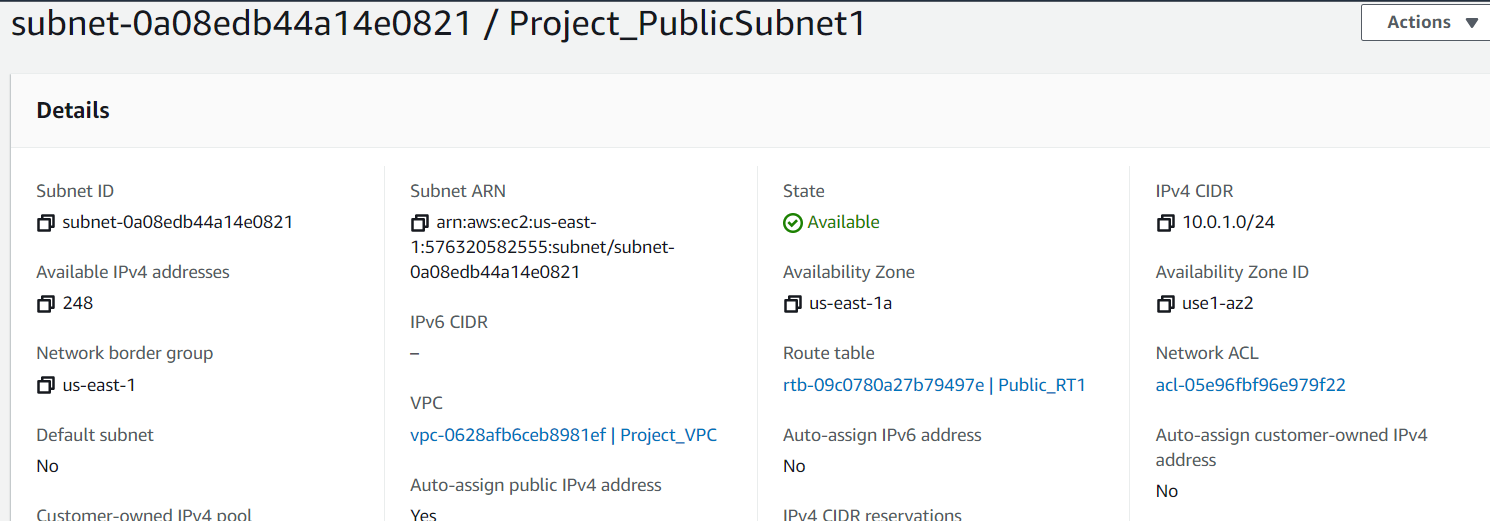
            "CidrBlock": { "Ref": "PrivateSubnetCIDR" },

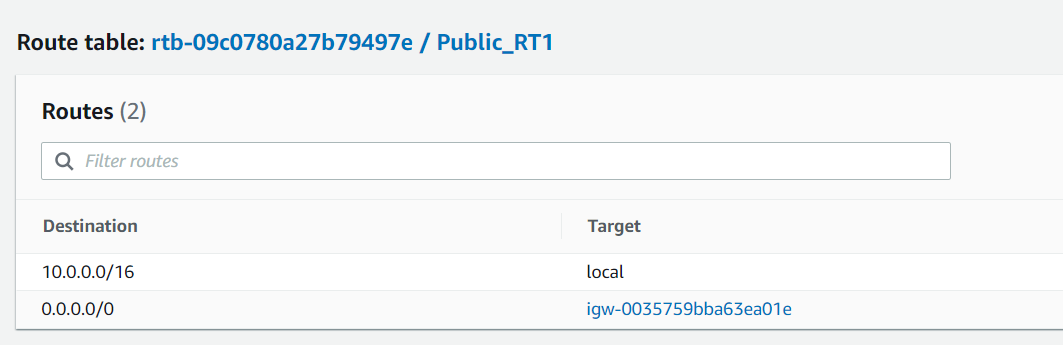
            "Tags": [{"Key":"Name","Value":"Project\_PrivateSubnet"}]

          }

        },

Public - Route table, route and association of route table to the subnet:





"PublicRouteTable1": {

          "Type": "AWS::EC2::RouteTable",

          "Properties": {

            "VpcId": { "Ref": "MyVPC" },

            "Tags": [{"Key":"Name","Value":"Public\_RT1"}]

          }

        },

        "PublicRoute1": {

          "Type": "AWS::EC2::Route",

          "Properties": {

            "RouteTableId": { "Ref": "PublicRouteTable1" },

            "DestinationCidrBlock": "0.0.0.0/0",

            "GatewayId": { "Ref": "InternetGateway1" }

            }

        },

        "PublicRouteAssociation1": {

          "Type": "AWS::EC2::SubnetRouteTableAssociation",

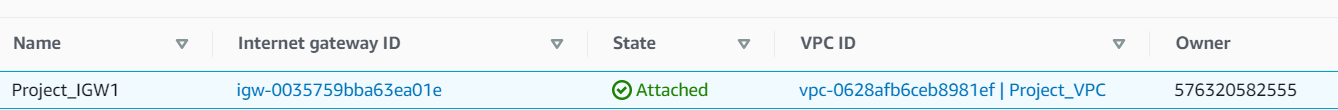
          "Properties": {

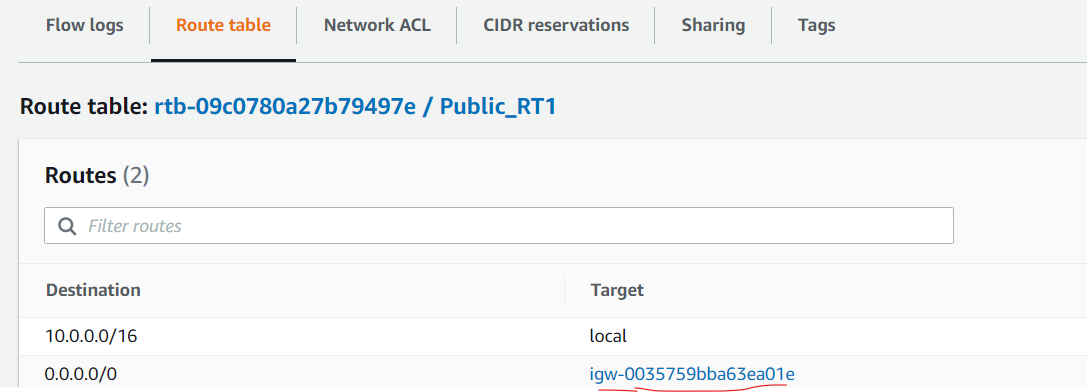
            "SubnetId": { "Ref": "PublicSubnet1" },

            "RouteTableId": { "Ref": "PublicRouteTable1" }

          }

Internet Gateway creation and attachment to public subnet





 "InternetGateway1": {

          "Type": "AWS::EC2::InternetGateway",

          "Properties": {

            "Tags": [{"Key":"Name","Value":"Project\_IGW1"}]

          }

        },

        "InternetGatewayAttachment1": {

          "Type": "AWS::EC2::VPCGatewayAttachment",

          "Properties": {

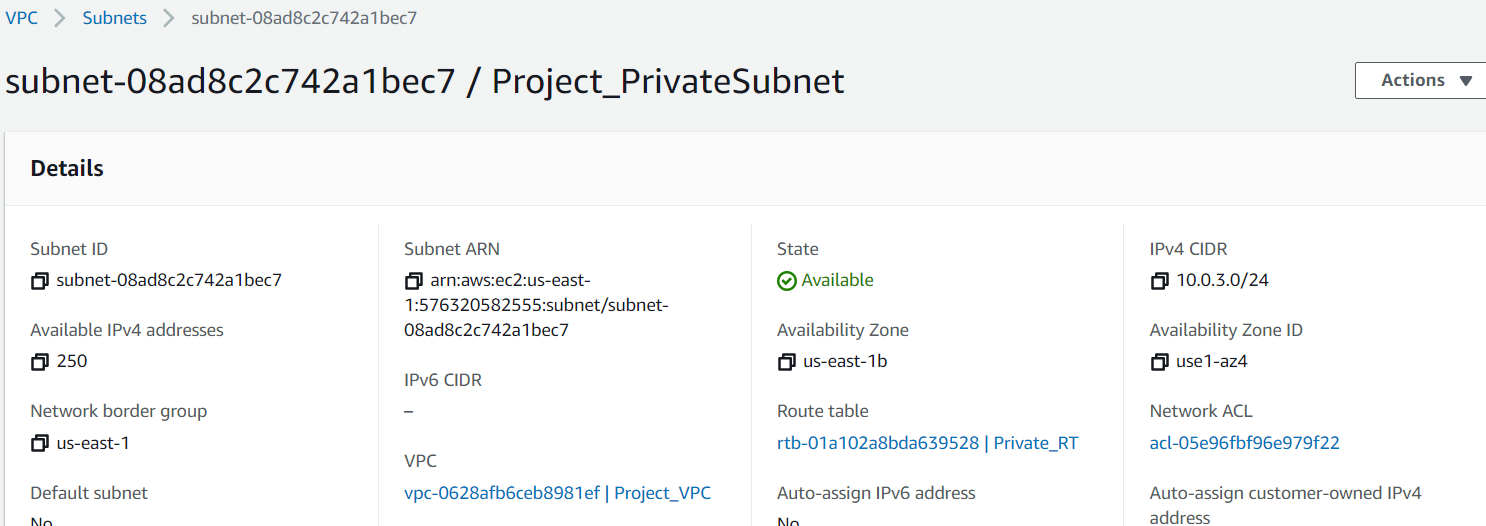
            "VpcId": { "Ref": "MyVPC" },

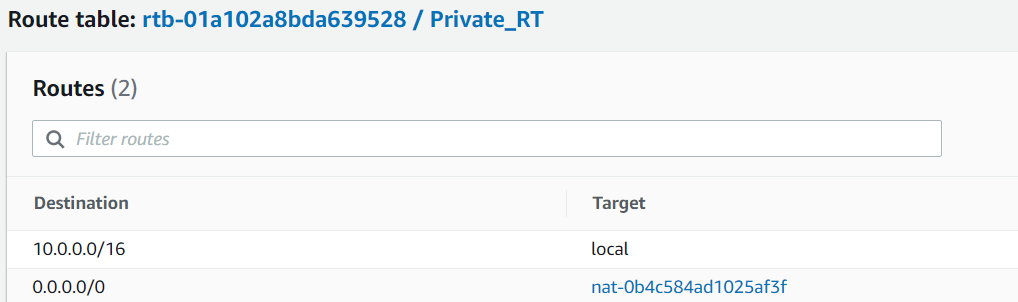
            "InternetGatewayId": { "Ref": "InternetGateway1" }

          }

        },

Private - Route table, route and association of route table to the subnet





"PrivateRouteTable": {

          "Type": "AWS::EC2::RouteTable",

          "Properties": {

            "VpcId": { "Ref": "MyVPC" },

            "Tags": [{"Key":"Name","Value":"Private\_RT"}]

          }

        },

        "DefaultPrivateRoute1":{

            "Type": "AWS::EC2::Route",

            "Properties": {

              "RouteTableId": { "Ref": "PrivateRouteTable" },

              "DestinationCidrBlock": "0.0.0.0/0",

              "NatGatewayId": { "Ref": "NATGateway1" }

            }

        },

        "PrivateRouteAssociation": {

          "Type": "AWS::EC2::SubnetRouteTableAssociation",

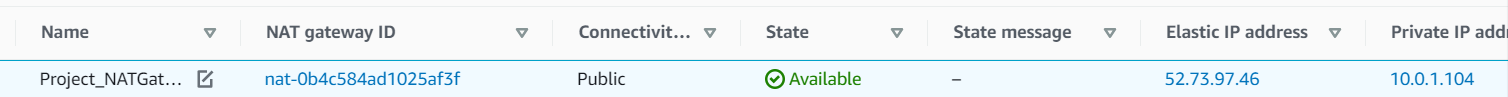
          "Properties": {

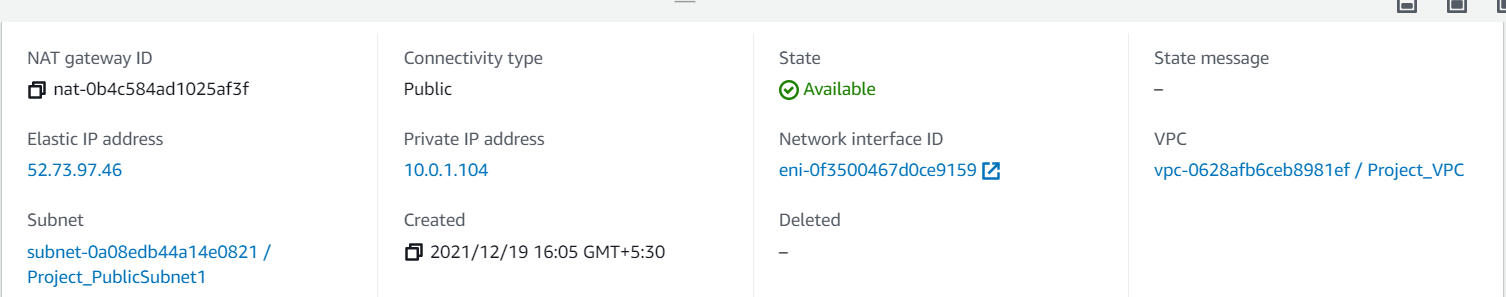
            "SubnetId": { "Ref": "PrivateSubnet1" },

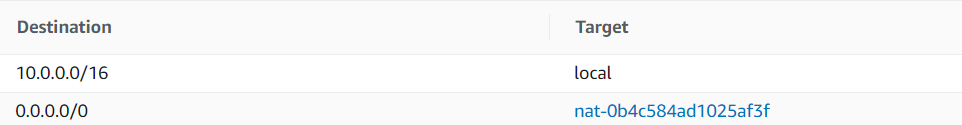
            "RouteTableId": { "Ref": "PrivateRouteTable" }

          }

NAT Gateway creation and altered private route table to attach to it







        "NatGateway1EIP": {

            "Type": "AWS::EC2::EIP",

            "DependsOn": "InternetGatewayAttachment1",

            "Properties": {

            "Domain": "MyVPC"

          }

        },

        "NATGateway1": {

            "Type": "AWS::EC2::NatGateway",

            "Properties":{

              "AllocationId": {"Fn::GetAtt": ["NatGateway1EIP", "AllocationId"] },

              "SubnetId": { "Ref": "PublicSubnet1" },

              "Tags":[{"Key":"Name","Value":"Project\_NATGatway1"}]

            }

        },

        "DefaultPrivateRoute1":{

            "Type": "AWS::EC2::Route",

            "Properties": {

              "RouteTableId": { "Ref": "PrivateRouteTable" },

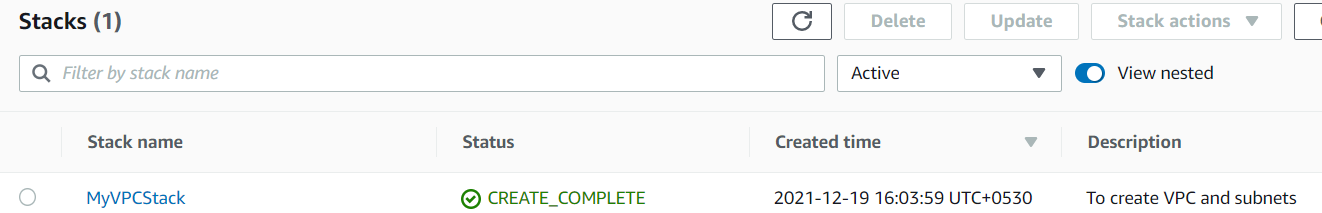
              "DestinationCidrBlock": "0.0.0.0/0",

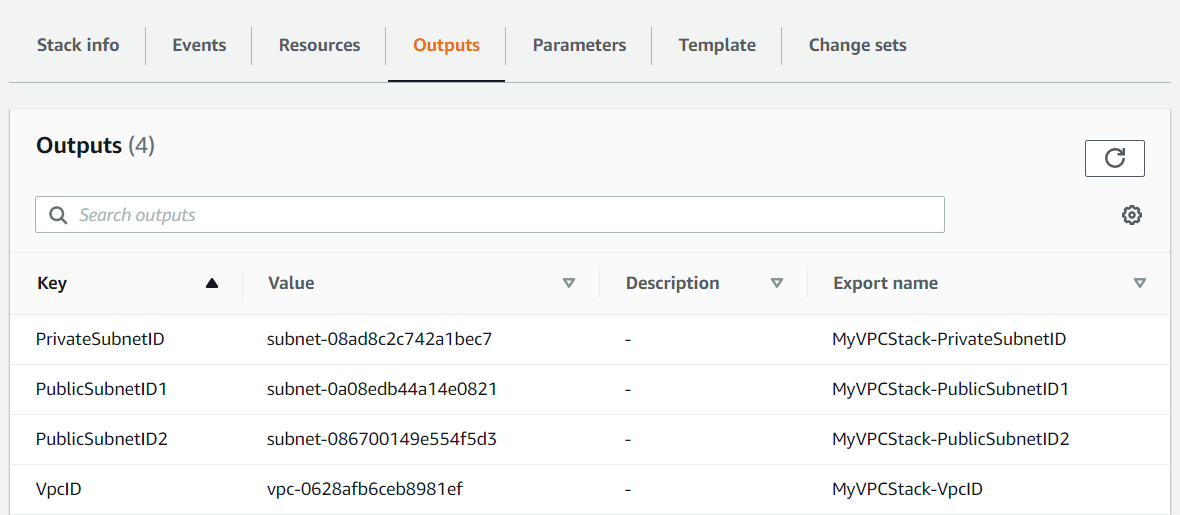
              "NatGatewayId": { "Ref": "NATGateway1" }

            }

        }

Cloud formation stack creation for the script execution

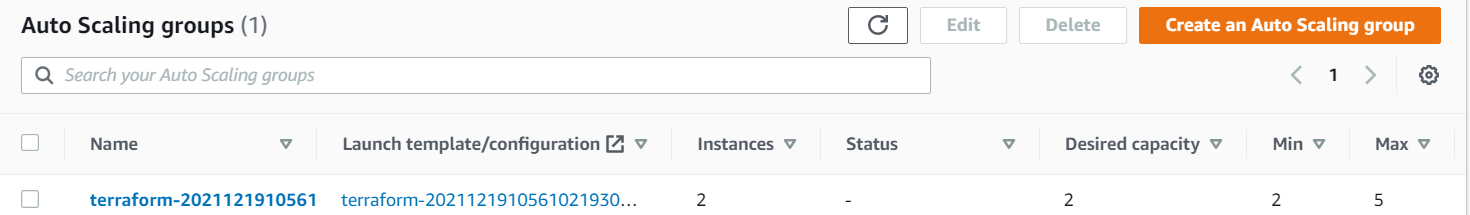




PART 1-Terraform

Auto Scaling group creation for public subnet EC2:

-> Min-2; Max-5  
-> AMI - Ubuntu18.04



resource "aws\_autoscaling\_group" "asg" {

  launch\_configuration = aws\_launch\_configuration.LC.name

  min\_size = 2

  max\_size = 5

  desired\_capacity = 2

  health\_check\_grace\_period = 300

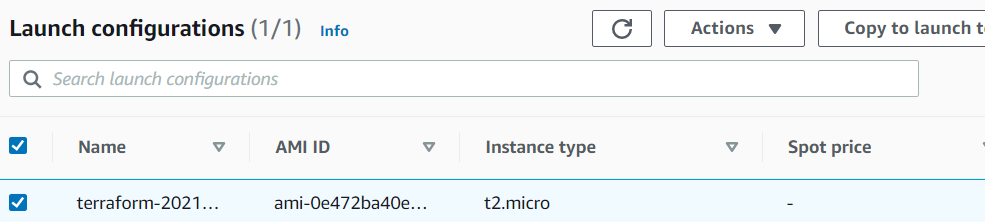
  load\_balancers = ["${aws\_elb.ELB.id}"]

  health\_check\_type = "EC2"

  vpc\_zone\_identifier = aws\_elb.ELB.subnets

}

Launch configuration of EC2 – ubuntu18.04



resource "aws\_launch\_configuration" "LC" {

  image\_id               = "${var.aws\_ami}"

  instance\_type          = "t2.micro"

  security\_groups        = ["${aws\_security\_group.SG.id}"]

  associate\_public\_ip\_address = true

  key\_name               = "${var.keyname}"

  lifecycle {

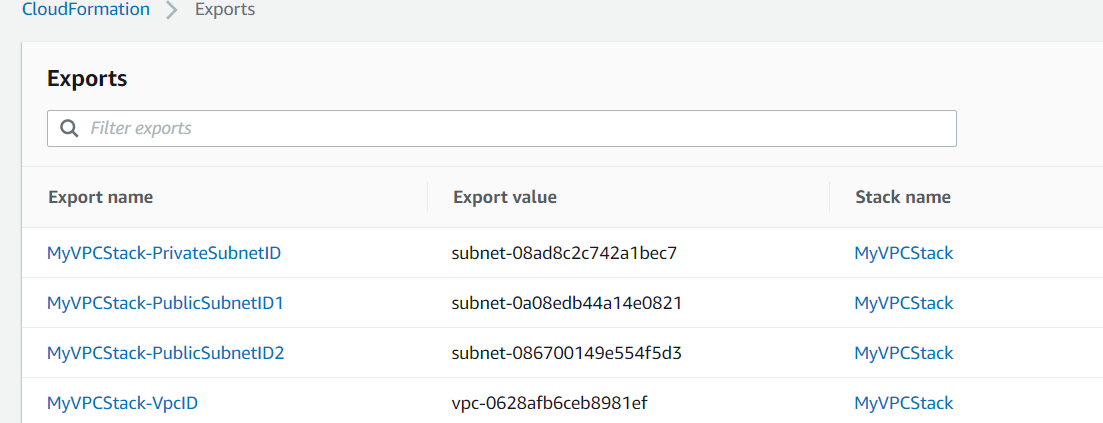
    create\_before\_destroy = true

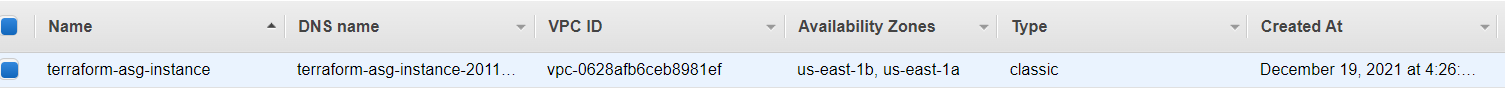
  }

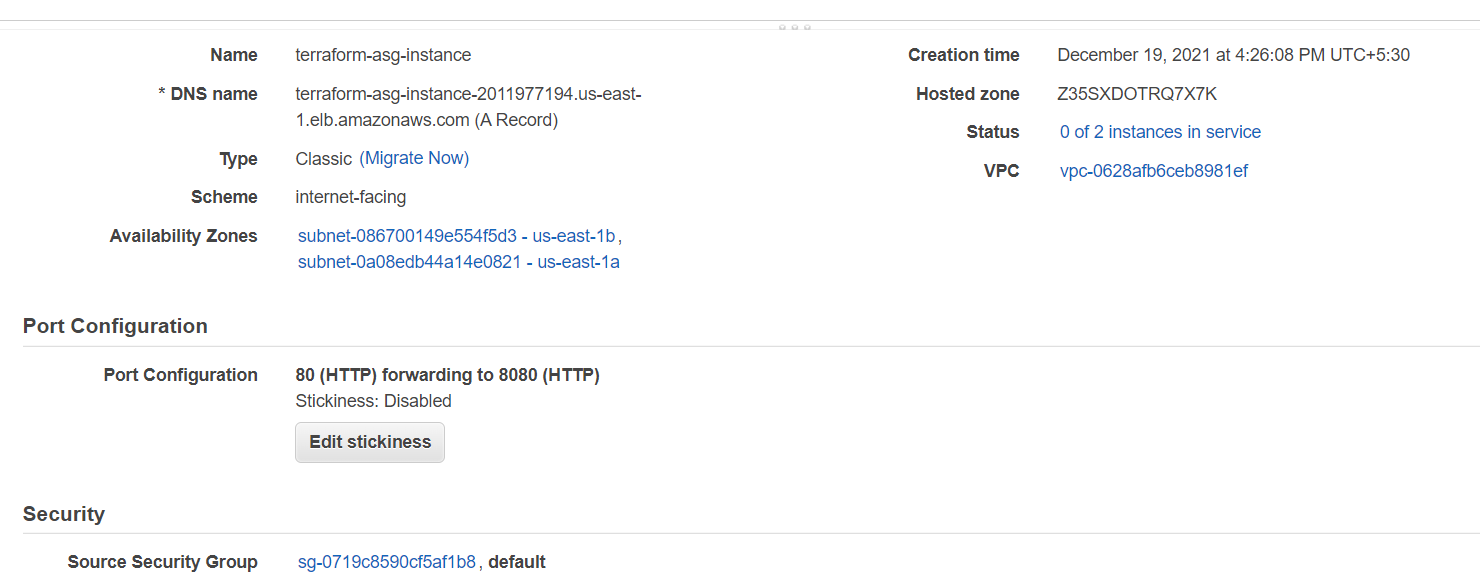
}

ELB creation and the required security group rules.

* Used cloudformation export function to get the values from cloudformation to terraform script







data "aws\_cloudformation\_export" "PublicSubnetID1" {

  name = "MyVPCStack-PublicSubnetID1"

}

data "aws\_cloudformation\_export" "PublicSubnetID2" {

  name = "MyVPCStack-PublicSubnetID2"

}

#data "aws\_availability\_zones" "all" { }

## Security Group for ELB

resource "aws\_security\_group" "elb\_sg" {

  name = "terraform-elb-sg"

  egress {

    from\_port = 0

    to\_port = 0

    protocol = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port = 80

    to\_port = 80

    protocol = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

## Creating ELB

resource "aws\_elb" "ELB" {

  name = "terraform-asg-instance"

  subnets = [data.aws\_cloudformation\_export.PublicSubnetID1.value,data.aws\_cloudformation\_export.PublicSubnetID2.value ]

  #availability\_zones = ["${data.aws\_availability\_zones.all.names}"]

  health\_check {

    healthy\_threshold = 2

    unhealthy\_threshold = 2

    timeout = 3

    interval = 30

    target = "HTTP:8080/"

  }

  listener {

    lb\_port = 80

    lb\_protocol = "http"

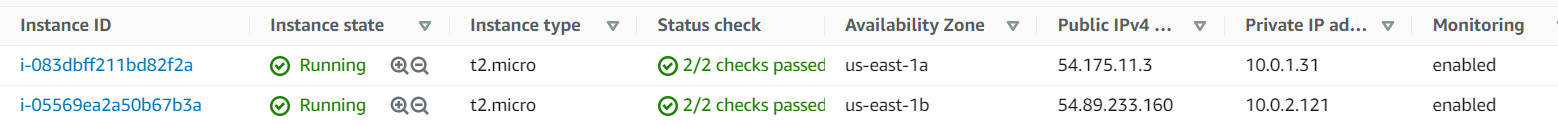
    instance\_port = "8080"

    instance\_protocol = "http"

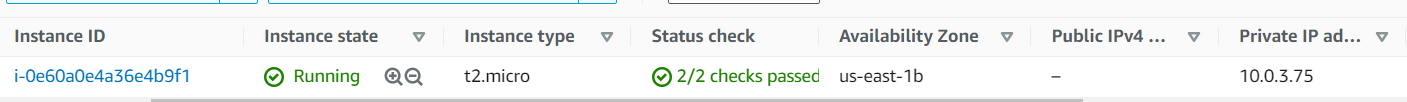
  }

}

Public subnet EC2 instances



In Private subnet EC2 instance creation using terraform



data "aws\_cloudformation\_export" "PrivateSubnetID" {

  name = "MyVPCStack-PrivateSubnetID"

}

resource "aws\_instance" "linuxvm" {

    instance\_type ="t2.micro"

    key\_name = "${var.keyname}"

    count = "${var.count\_instance}"

    ami = "${var.aws\_ami}"

    subnet\_id = data.aws\_cloudformation\_export.PrivateSubnetID.value

    security\_groups = "${[aws\_security\_group.privateSG.id]}"

    tags = {

    Name = "LinuxVM\_Privatesubnet"

    }

}

PART 1-Ansible

Apache installation on public node and copy index.html

* Created two EC2 instances in public subnet and need to install apache on those instances using ansible.
* I’ve considered out of those two EC2 instances one as Ansible controller and one as Ansible node
* Installed Ansible and required packages on ansible controller

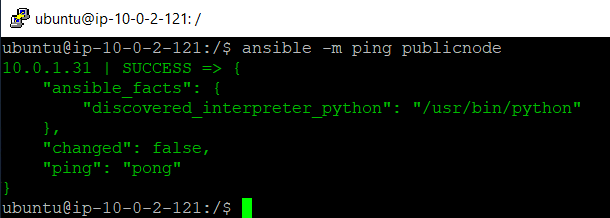
sudo apt-get update  
sudo apt-get upgrade -y  
sudo apt-add-repository ppa:ansible/ansible  
sudo apt-get install ansible -y  
sudo apt-get install python -y

* On ansible node installed python which is a pre-requisite to run ansible scripts
* To make the ansible server and node connection to run scripts for configuration changes:

Generated ssh publickey of ansible controller – ssh-keygen

Copied the value and pasted it on the ~/.ssh/authorizedkeys of ansible node.

* Verified the connectivity of node



Created apache.yaml for installation of apache on public node

---

- hosts: publicnode

  become: yes

  tasks:

    - name: Install latest version of Apache

      apt:

        name: apache2

        state: present

        update\_cache: yes

    - name: create webpage index.html

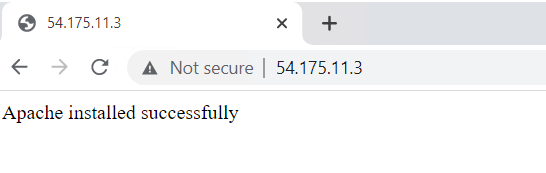
      copy:

        dest: /var/www/html/index.html

        content:

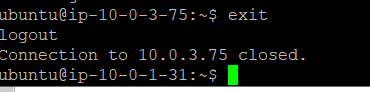
              Apache installed successfully

Copied the index.html with the content “Apache installed successfully”

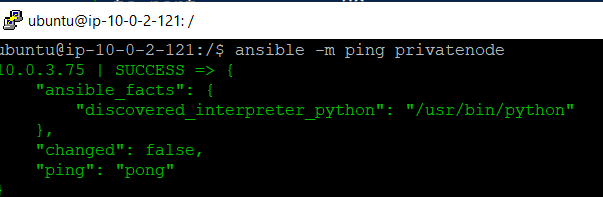


Installation of MySQL on the private node

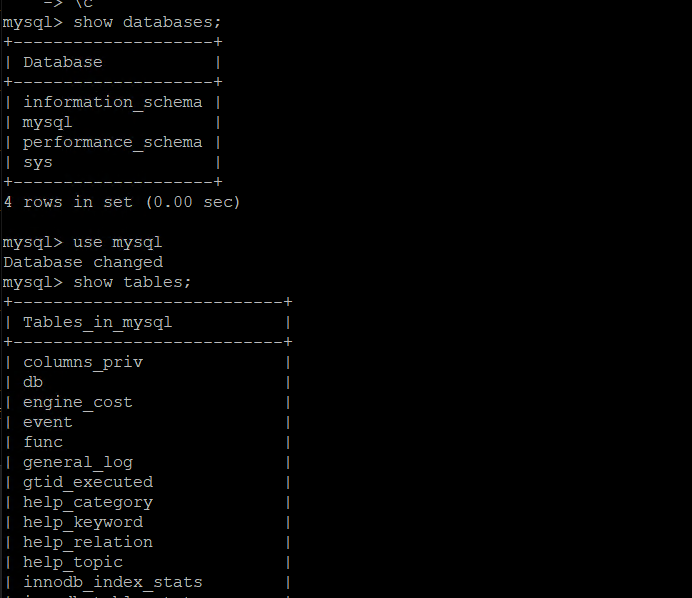
* Copied the linvmkp1.pem file content to the public ansible node
* Verified the private node connection from public node  
  ssh -i "linvmkp.pem" [ubuntu@10.0.3.75](mailto:ubuntu@10.0.3.75)



* Repeated the steps similar to public ansible node to make the connection between ansible server and ansible private node



Prepared the Mysql.yaml file to install mysql on the private node and installed it.

---

- hosts: privatenode

  become: yes

  tasks:

    - name: Install MySQL

      apt: pkg={{ item }} state=latest

      with\_items:

        - mysql-server

        - mysql-client

        - python-mysqldb

    - name: Start the MySQL service

      action: service name=mysql state=started

    - name: Remove the test database

      mysql\_db: name=test state=absent

    - name: Create test user for mysql

      mysql\_user: user="test" host="%" password=admin123 priv=\*.\*:ALL,GRANT

    - name: Ensure anonymous users are not in the database

      mysql\_user: user='' host=$item state=absent

      with\_items:

        - 127.0.0.1

        - ::1

        - localhost

    - name: Update mysql root password for all root accounts

      mysql\_user: name=root host={{item}} password=admin123

      with\_items:

        - 127.0.0.1

        - ::1

        - localhost