**SysOps Project**

**Part-1** Automation

**Step 1 : CloudFormation Requirement:**

* Create a VPC using CloudFormation - with one public subnet, one private subnet.
* Create IGW
* Create Routing Tables

{

"AWSTemplateFormatVersion": "2010-09-09",

"Resources": {

"MyVPC": {

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock": "10.0.0.0/16",

"EnableDnsSupport": "true",

"EnableDnsHostnames": "true",

"InstanceTenancy": "default"

}

},

"InternetGateway": {

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

},

"VPCGatewayAttachment": {

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

"Properties": {

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

"InternetGatewayId": {"Ref" : "InternetGateway"}

}

},

"Sub1": {

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

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

"CidrBlock": "10.0.1.0/24",

"MapPublicIpOnLaunch": "true"

}

},

"Sub2": {

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2b",

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

"CidrBlock": "10.0.2.0/24",

"MapPublicIpOnLaunch": "false"

}

},

"RouteTable": {

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

"Properties": {

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

}

},

"InternetRoute": {

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

"DependsOn": "VPCGatewayAttachment",

"Properties": {

"DestinationCidrBlock": "0.0.0.0/0",

"GatewayId": {"Ref" : "InternetGateway"},

"RouteTableId": {"Ref": "RouteTable"}

}

},

"Sub1RouteTableAssociation": {

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

"Properties": {

"RouteTableId": {"Ref" : "RouteTable"},

"SubnetId": {"Ref": "Sub1"}

}

},

"Sub2RouteTableAssociation": {

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

"Properties": {

"RouteTableId": {"Ref" : "RouteTable"},

"SubnetId": {"Ref": "Sub2"}

}

}

},

"Outputs": {

"VpcId": {

"Value": {"Ref":"MyVPC"},

"Export": {"Name": {"Fn::Sub": "${AWS::StackName}-VpcId"}}

},

"PublicSubnetID": {

"Value": {"Ref":"Sub1"},

"Export": {"Name": {"Fn::Sub":"${AWS::StackName}-PublicSubnetID"}}

},

"PrivateSubnetID":{

"Value": {"Ref":"Sub2"},

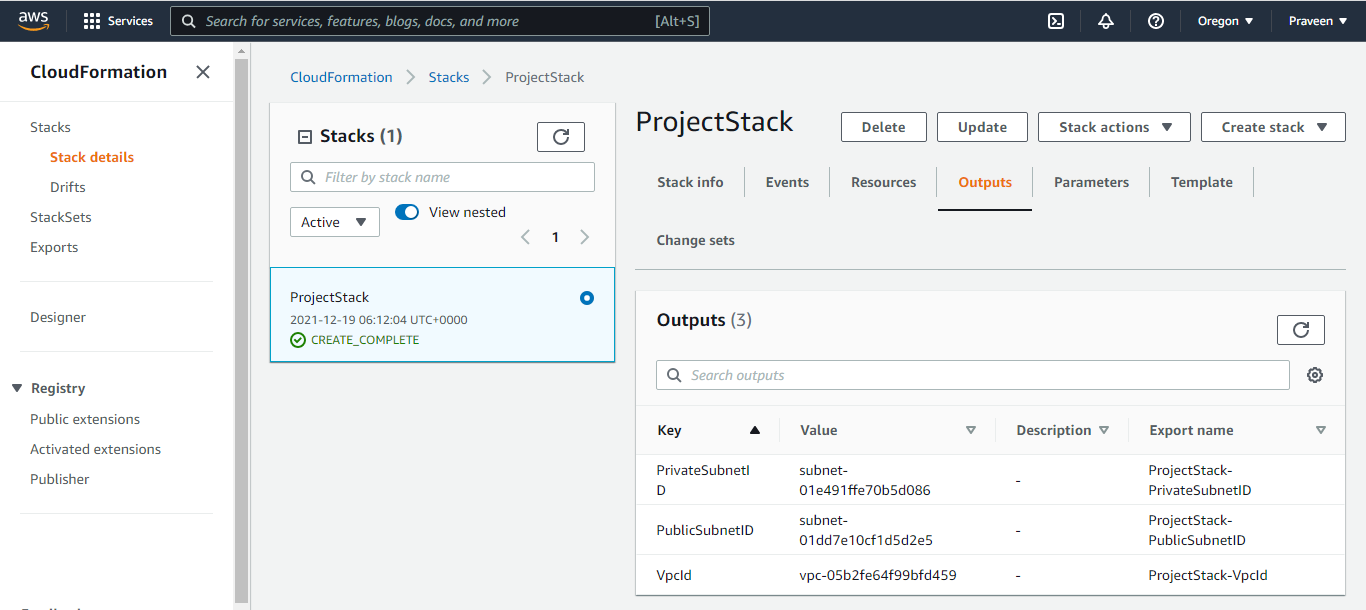
"Export": {"Name": {"Fn::Sub":"${AWS::StackName}-PrivateSubnetID"}}

}

}

}

Output:



**Step 2 : Terraform Requirement**

* In public subnet - Auto Scaling group with min cap as 2 and max as 5, AMI- ubuntu 18.04
* Frontend with a layer 4 Load balancer
* In private subnet - One EC2 with Ubuntu

Asg.tf:

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

name = "asg-public"

launch\_configuration = "${aws\_launch\_configuration.web.name}"

min\_size = 2

desired\_capacity = 2

max\_size = 5

vpc\_zone\_identifier = [data.aws\_cloudformation\_export.publicsubnetid.value]

target\_group\_arns = ["${aws\_lb\_target\_group.test.arn}"]

health\_check\_type = "ELB"

force\_delete = true

lifecycle {

create\_before\_destroy = true

}

tag {

key = "Name"

value = "EC2"

propagate\_at\_launch = true

}

}

Elb.tf:

resource "aws\_elb" "elb" {

name = "elb"

subnets = [data.aws\_cloudformation\_export.publicsubnetid.value]

listener {

instance\_port = 80

instance\_protocol = "http"

lb\_port = 80

lb\_protocol = "http"

}

health\_check {

healthy\_threshold = 2

unhealthy\_threshold = 2

timeout = 5

target = "TCP:22"

interval = 30

}

}

resource "aws\_lb\_target\_group" "test" {

name = "elb-tg"

port = "80"

protocol = "TCP"

vpc\_id = data.aws\_cloudformation\_export.vpcid.value

}

Data.tf:

data "aws\_ami\_ids" "ubuntu" {

filter {

name = "name"

values = ["ubuntu/images/hvm-ssd/ubuntu-trusty-18.04-amd64-server-\*"]

}

filter {

name = "virtualization-type"

values = ["hvm"]

}

owners = ["193367416928"] # Canonical

}

data "aws\_cloudformation\_export" "vpcid" {

name = "ProjectStack-VpcId"

}

data "aws\_cloudformation\_export" "publicsubnetid" {

name = "ProjectStack-PublicSubnetID"

}

data "aws\_cloudformation\_export" "privatesubnetid" {

name = "ProjectStack-PrivateSubnetID"

}

Launch.tf:

resource "aws\_launch\_configuration" "web" {

name\_prefix = "web-"

image\_id = "ami-074251216af698218" # Ubuntu 18.04, SSD Volume Type

instance\_type = "t2.micro"

key\_name = "labkp1"

security\_groups = [

"${aws\_security\_group.demosg1.id}"

]

associate\_public\_ip\_address = true

user\_data = <<-EOF

#!/bin/sh

sudo apt-get update

sudo apt-get install unzip tree

sudo apt-get install -y python-pip

pip install boto

EOF

lifecycle {

create\_before\_destroy = true

}

}

Private.tf:

resource "aws\_instance" "PrivateLinux" {

ami = "ami-074251216af698218"

instance\_type = "t2.micro"

key\_name = "labkp1"

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

vpc\_security\_group\_ids = [aws\_security\_group.demosg1.id]

associate\_public\_ip\_address = false

user\_data = <<-EOF

#!/bin/sh

sudo apt-get update

sudo apt-get install unzip tree

sudo apt-get install -y python-pip

pip install boto

EOF

tags = {

"Name" : "PrivateLinux"

}

}

Variables.tf:

#region

variable "region" {

type = string

default = "us-west-2"

}

#ami id

variable "ami\_id" {

type = map

default = {

"us-west-2" = "ami-074251216af698218" }

}

#instance type

variable "instance\_type" {

type = string

default = "t2.micro"

}

#existing key-pair name

variable "key\_name" {

type = string

default = "labkp1"

}

# public subnet id

variable "publicsubnetid" {

type = string

default = "subnet-0170dda0fd63ba961"

}

#private subnet id

variable "privatesubnetid" {

type = string

default = "subnet-0572e3a654e2def0b"

}

#vpc id

variable "vpcid" {

type = string

default = "vpc-0c582012bec0e2e7b"

}

variable "http\_port" {

description = "The port to use for HTTP traffic. Defaults to 80"

default = "80"

}

variable "ssh\_port" {

description = "The port to use for HTTP traffic. Defaults to 22"

default = "22"

}

variable "https\_port" {

description = "The port to use for HTTPS traffic. Defaults to 443"

default = "443"

}

variable "instance\_tenancy" {

description = "A tenancy option for instances launched into the VPC"

type = string

default = "default"

}

variable "enable\_dns\_hostnames" {

description = "Should be true to enable DNS hostnames in the VPC"

type = bool

default = true

}

variable "enable\_dns\_support" {

description = "Should be true to enable DNS support in the VPC"

type = bool

default = true

}

variable "enable\_classiclink" {

description = "Should be true to enable ClassicLink for the VPC. Only valid in regions and accounts that support EC2 Classic."

type = bool

default = false

}

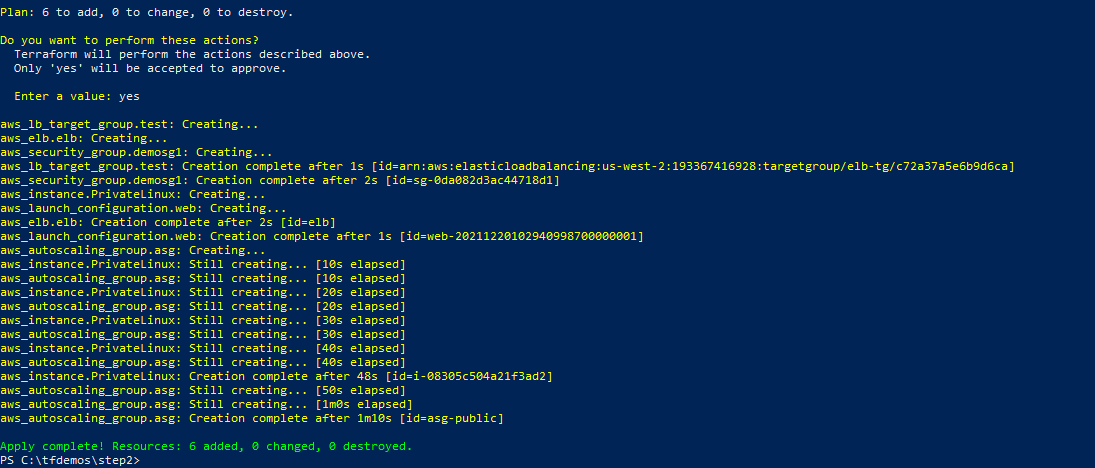
variable "tags" {

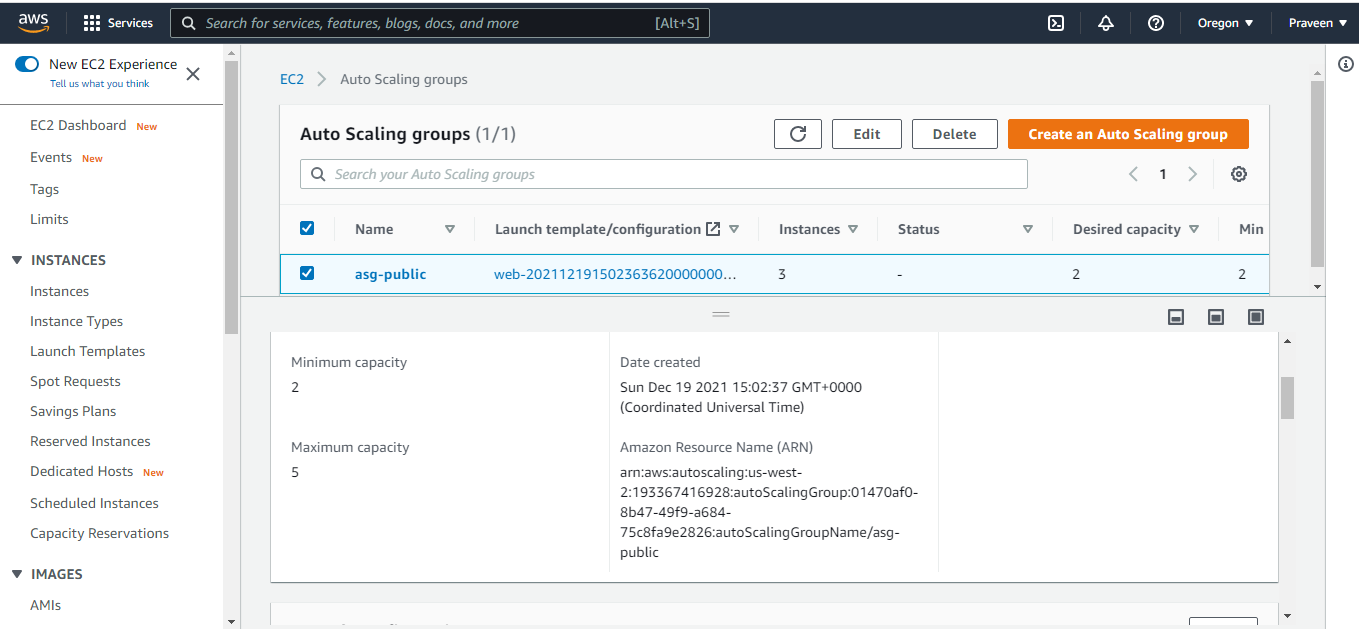
description = "A map of tags to add to all resources"

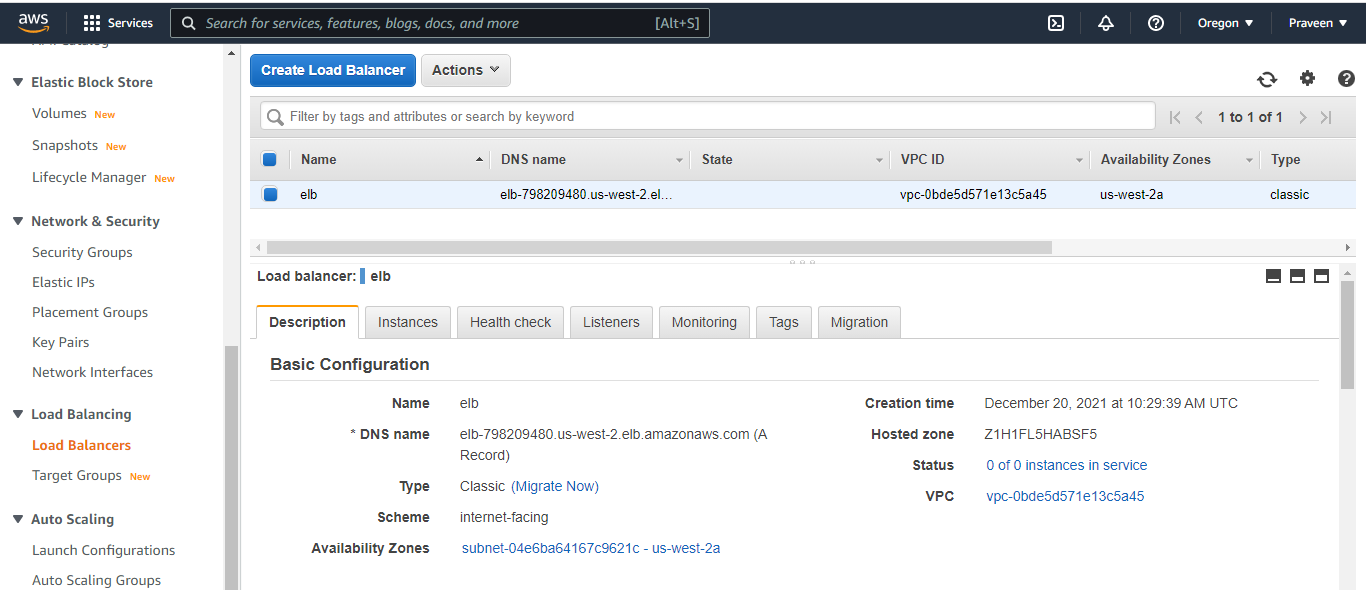
type = string

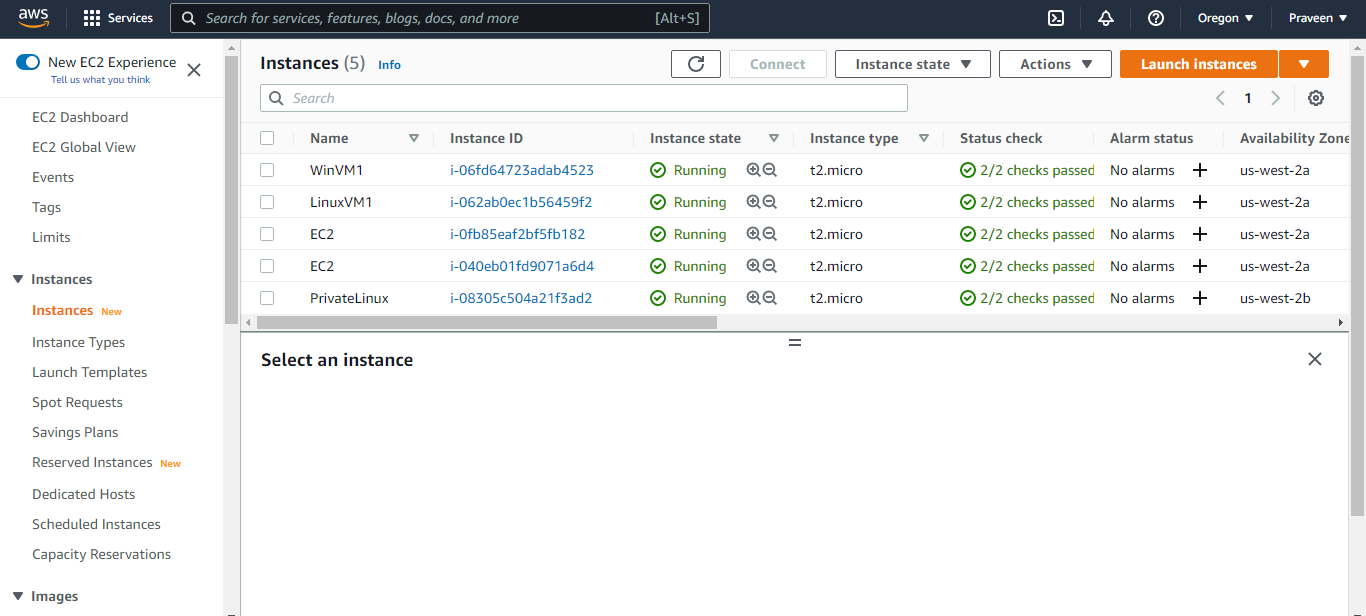
default = "Vpc-custom-demo"

}









**Step 3: Ansible Requirement:**

• On the ASG and EC2 install Apache and on the Apache copy index.html

Dynamic Inventory:

[inventory]

enable\_plugins = aws\_ec2

[defaults]

inventory = /home/ubuntu/demos/aws\_ec2.yaml

host\_key\_checking = false

remote\_user = ubuntu

ask\_pass = false

private\_key\_file = /home/ubuntu/demos/labkp1.pem

Aws\_ec2.yaml:

---

plugin: aws\_ec2

aws\_access\_key: AKIAS2BMYUBQEYQLKC3T

aws\_secret\_key: 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

keyed\_groups:

- key: tags

prefix: tag

- prefix: instance\_type

key: instance\_type

Apache.yaml:

---

- hosts: tag\_Name\_EC2

connection: ssh

become: true

tasks:

- name: installing apache server

apt:

name: apache2

update\_cache: true

state: present

- name: create webpage

copy:

src: /home/ubuntu/demos/index.html

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

- name: start apache

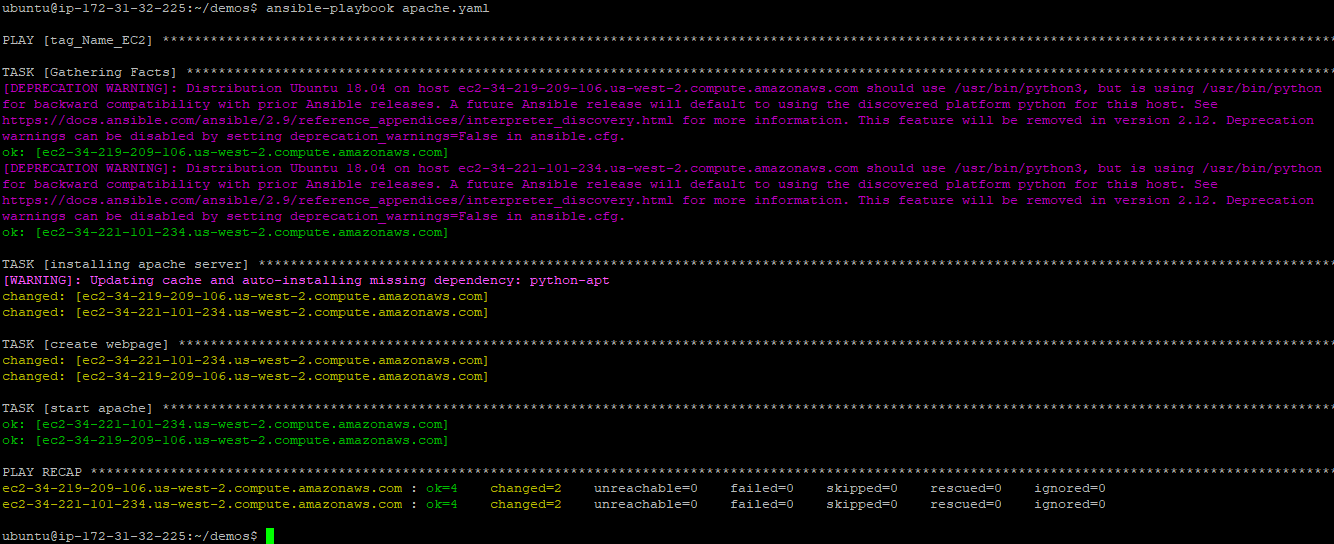
service:

name: apache2

state: started

Index.html:

<h1>Welcome to SysOps Server</h1>







• On private subnet EC2 install MySql

MySQL.yaml:

---

- hosts: tag\_Name\_PrivateLinux

connection: ssh

become: true

tasks:

- name: installing MySQL packages

apt:

name: ['mysql-server', 'mysql-client', 'python-mysqldb']

state: present

update\_cache: true

- name: start the MySQL service

service:

name: mysql

state: started

enabled: true

---

- hosts: tag\_Name\_PrivateLinux

connection: ssh

become: true

tasks:

- name: installing MySQL packages

apt:

update\_cache: yes

name:

- mysql-server

- mysql-client

- python-mysqldb

state: present

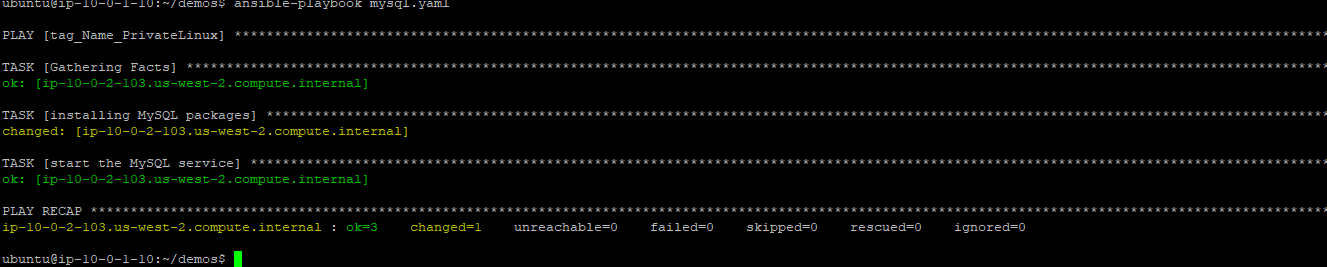
- name: start the MySQL service

service:

name: mysql

state: started

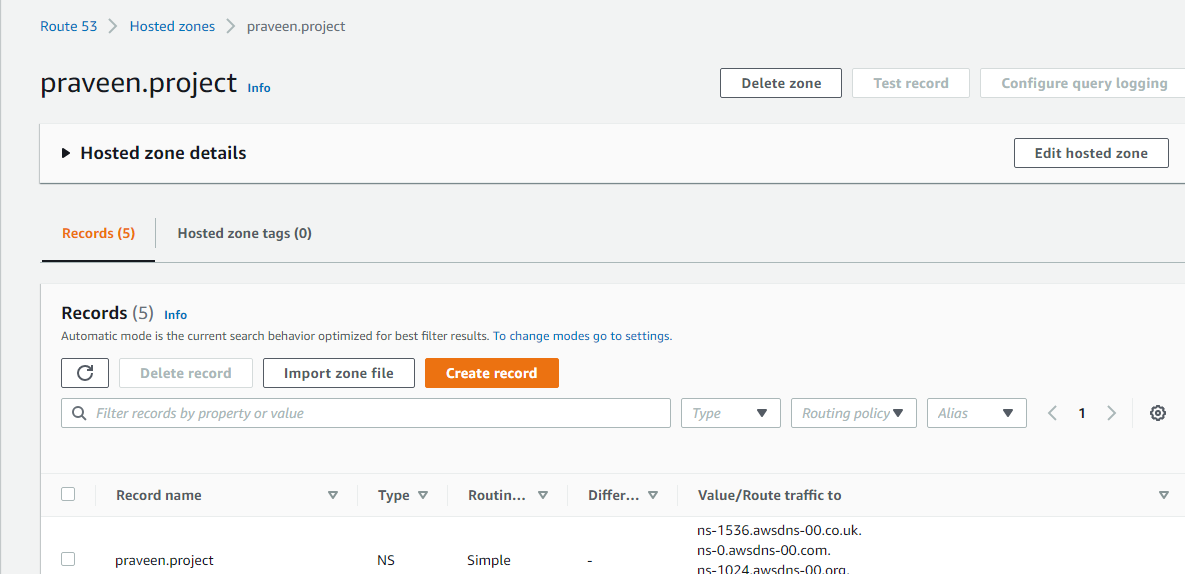
enabled: true

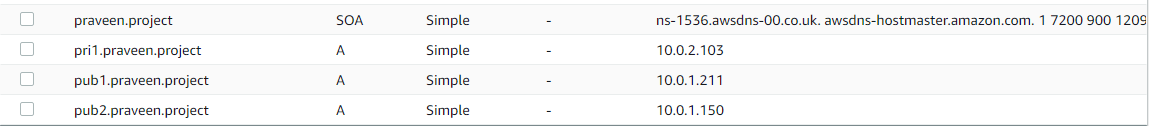


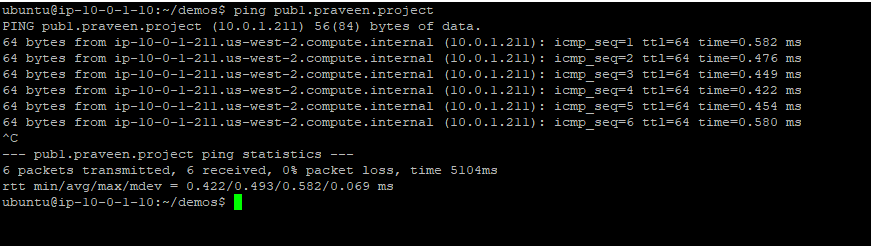
**Part-2:** Secure Communication and Security Management for Application (Manual Configuration)

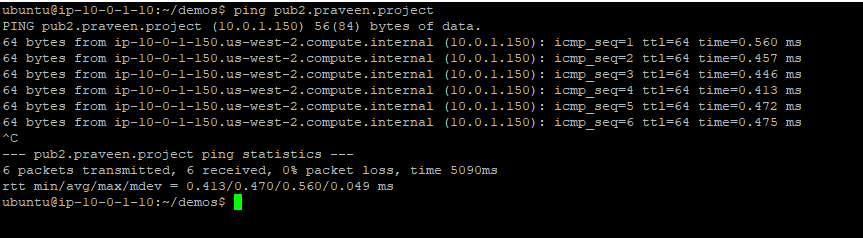
**Requirement:**

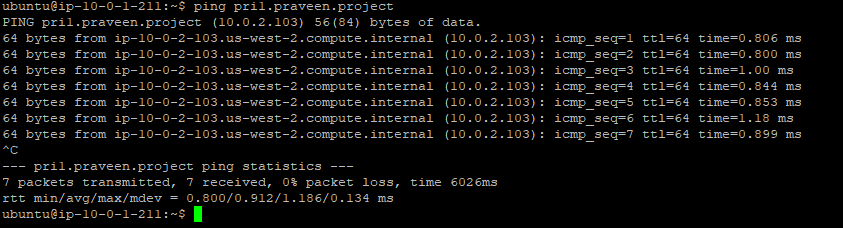
• Route53 Configuration for VPC internal communication



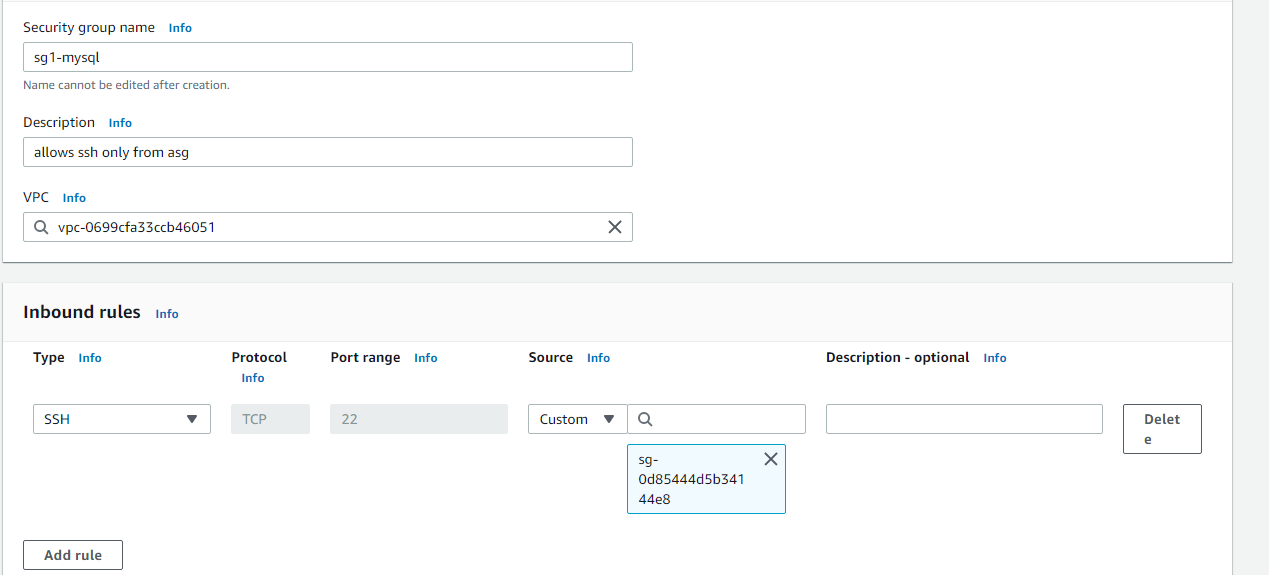




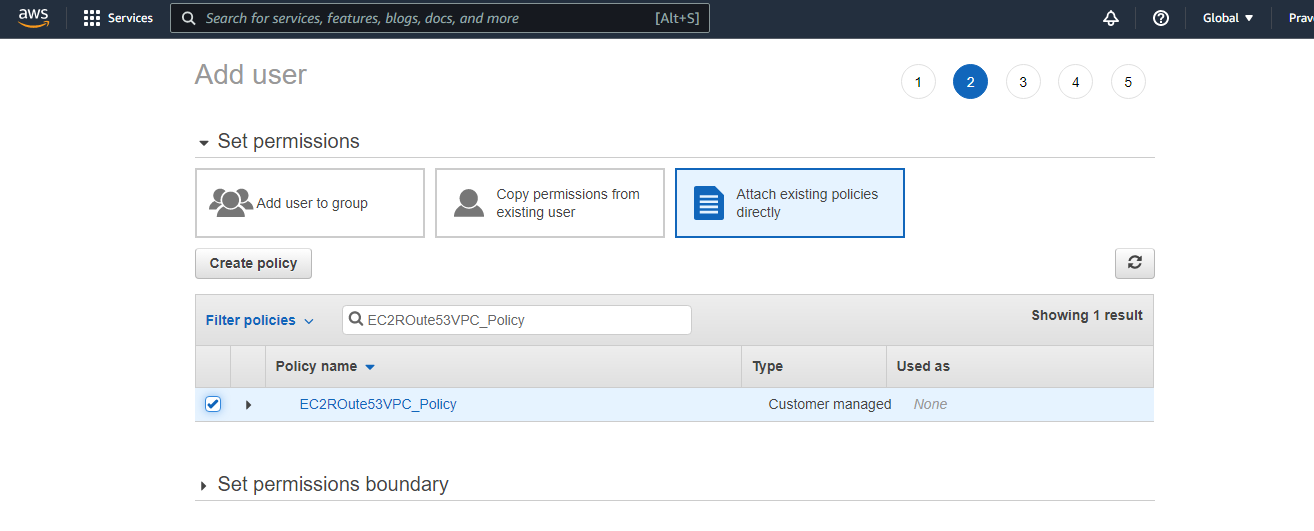


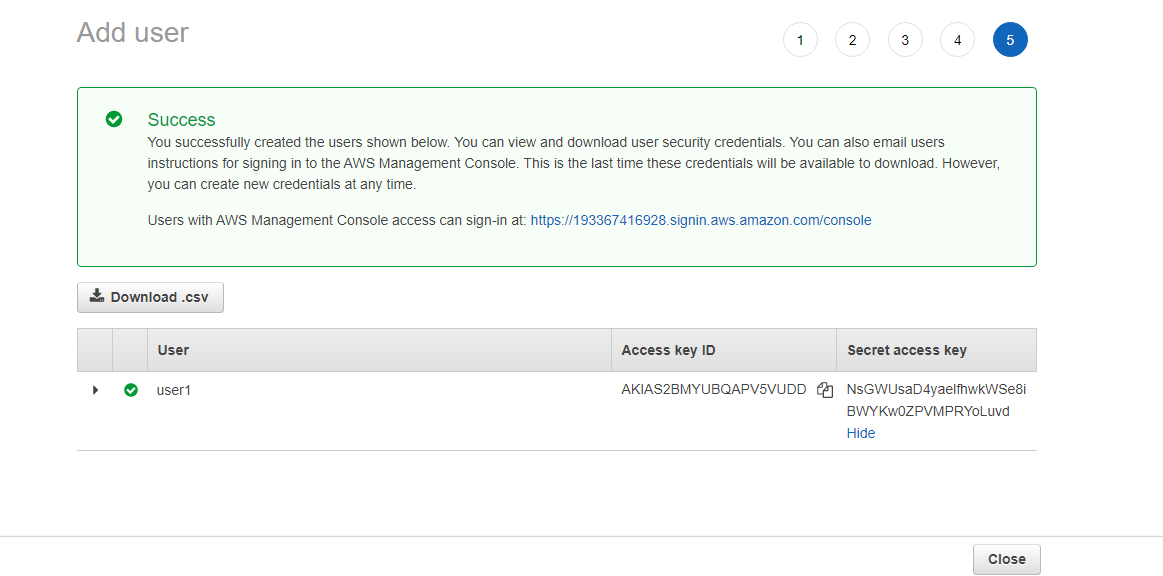


• MySQL DB (EC2) should be accessible only from ASG (Webserver Cluster) Hint: Security Groups



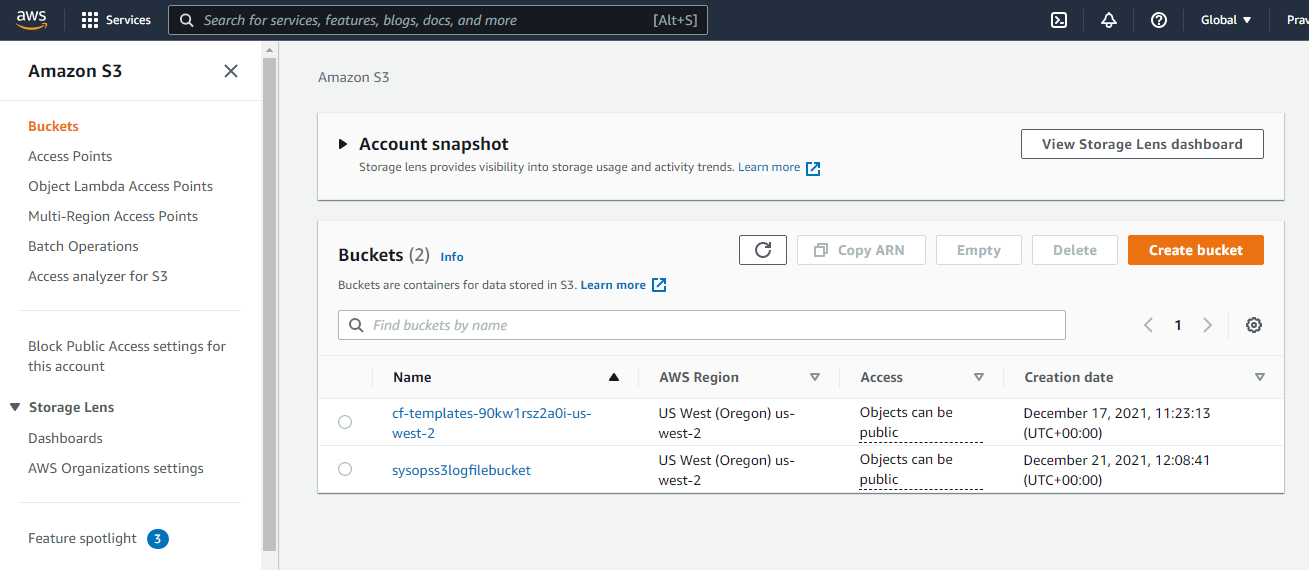
• A user to manage application infrastructure (VPC, Route53, EC2) Hint : IAM Policies





• S3 Bucket to store log files

• All EC2 instances should have permission to access S3 Hints: IAM Roles Note: All internal Communication between VMs must be using host name not through IP

****

**Part 3 -** Monitoring App and App Infrastructure (Manual Configuration)

Required to create a CloudWatch Dashboard to monitor :

o Network Traffic (Network I/O)

o EC2 instances- CPU, Disk, Memory Usage

o Load balancer Traffic

