**LOAD BALANCER WITH AUTOSCALING WITH THE RULE 70-30 WEIGHTED POLICY**

1.with the weighted routing we can switch the traffic between versions of your application

2.This configuration allows up to control the distribution of traffic to our application

**1.create a template--nginx-tg**

**2.create a template--apache-tg**

**3.create a autoscaling group(nginx)**

1.name ------ASG-nginx

2.template---nginx-lt

3.az and subnets----1a,1b,1c

4.capacity

desired-------1

minimum----1

maximum----2

5.add sns

6.tag---Name-nginx

7.create a autoscaling group

**4.create a autoscaling group(apache)**

1.name ------ASG-apache

2.template---apache-lt

3.az and subnets----1a,1b,1c

4.capacity

desired-------1

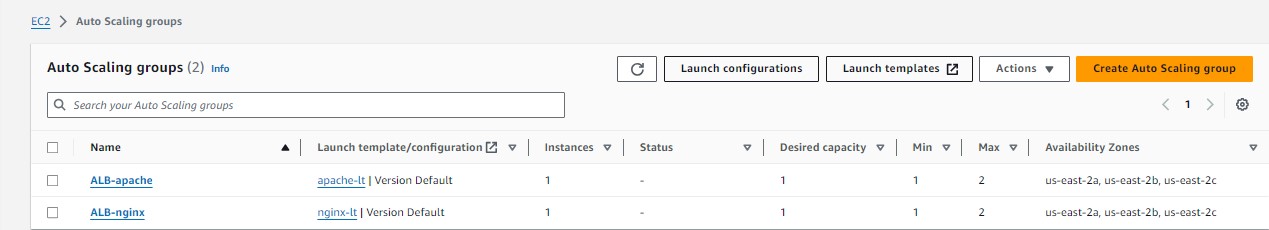
minimum----1

maximum----2

5.add sns

6.tag---Name-apache

7.create a autoscaling group



**5.create a target group (nginx)**

1.name------tg-nginx

2.type-------instance

3.path---------/

4.register-------nginx

**6.create a target group (apache)**

1.name------tg-apache

2.type-------instance

3.path---------/

4.register-------apache

**7.create a load balancer**

1.name------ALB

2.default action-------tg-nginx

3.create a target group

**9.click on created ALB**

1.select listner------click on listner

2.action--------------edit listner

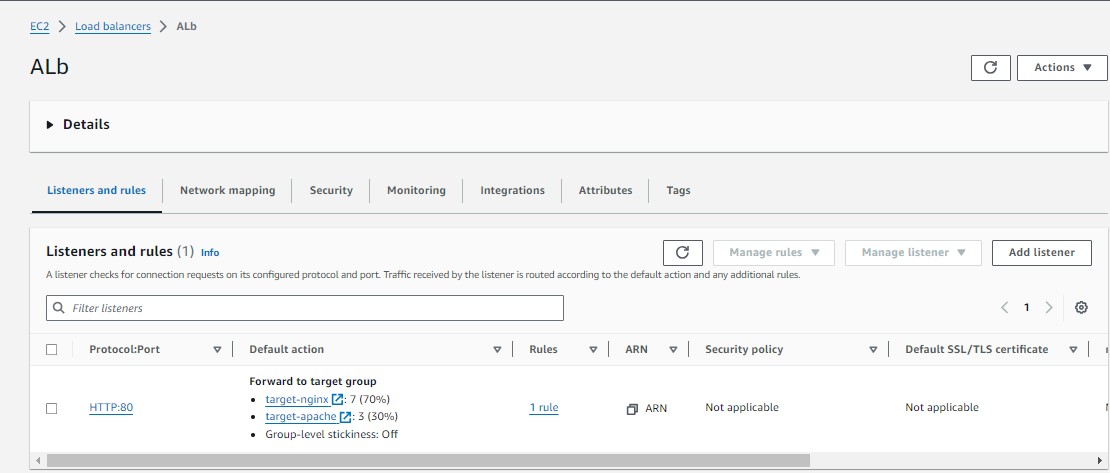
3.add target group-----tg-apache

4.add the weighted policy

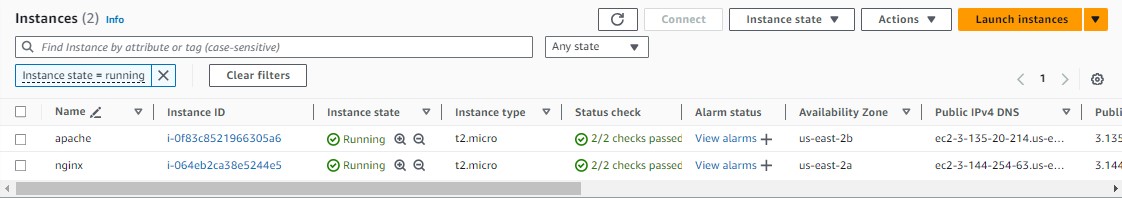
1. tg-nginx---------7 --------------70%

2.tg-apache-------3 --------------30%

5.save the changes



**10.check the instances that we create**



**10.click on created instance – nginx---connect**

1.sudo apt update -y

2.sudo apt install nginx -y

3.sudo systemctl start nginx

**11.click on created instance apache----connect**

1.sudo apt update -y

2.sudo apt install apache2 -y

Sudo systemctl start apache2

**12.click on application load balancer that you created**

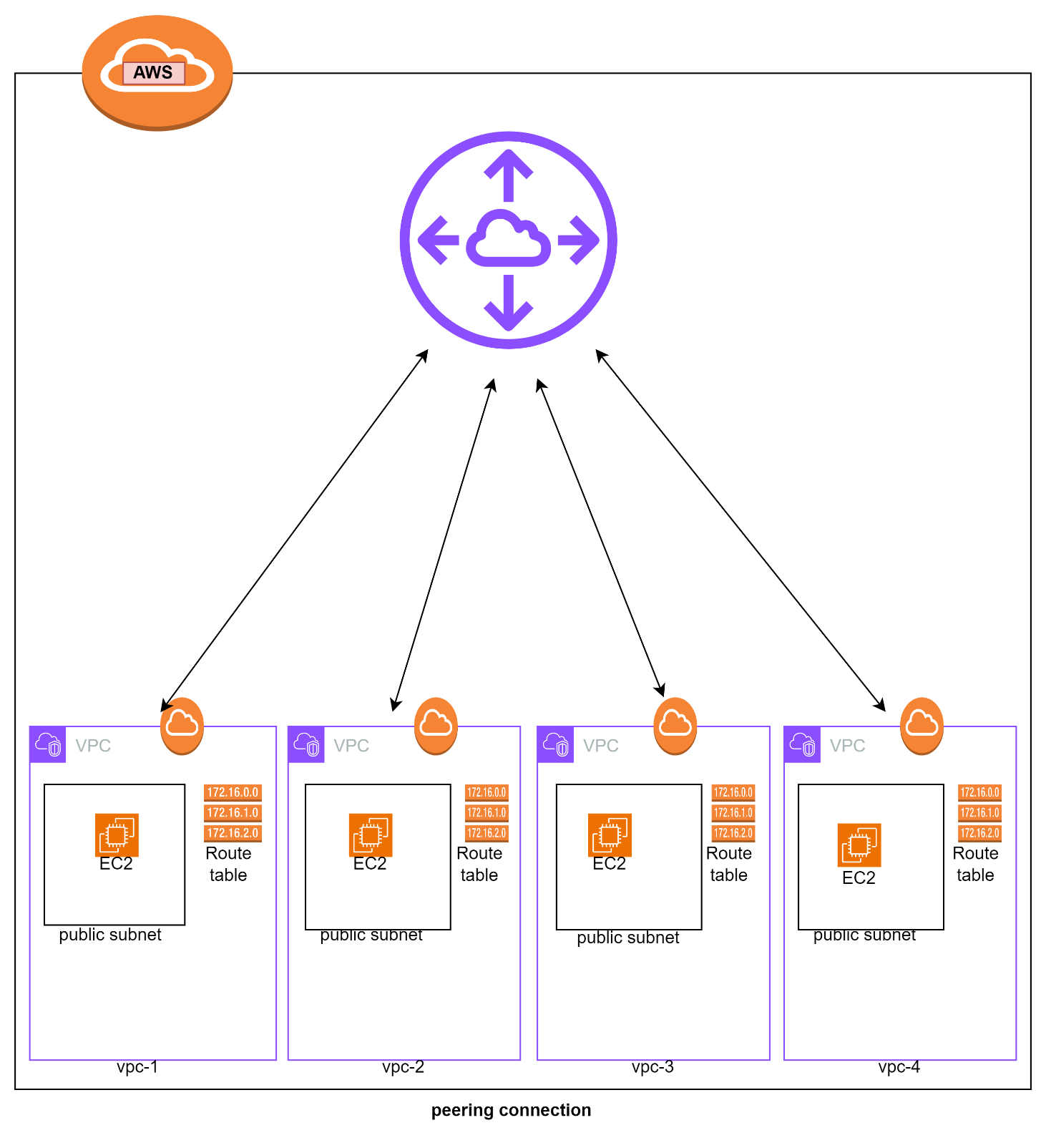
1.copy the dns and paste

**13.the traffic is moving on about 70% in nginx while in apache2 it should about 30%**

===========================================================================

**IMPLEMENT PEERING BETWEEN 4 VPC**

**1.create a vpc1-vpc2-vpc2 and vpc3**



**vpc1:-**

1.create a vpc-1 -----------------------12.0.0.0/16

2.create a subnet--(subnet-vpc-1)-------12.0.1.0/24

3.create a IGW and attach to the vpc-1

4.create a route table--(rt-vpc-1)

associate---subnet-vpc-1----IGW

5.create a instance

1.name---------ec2-vpc-1

2.machine------amazon-linux

3.type---------t2.micro

4.key----------ec2-vpc-1 key

5.user-data

#!/bin/bash

sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd.service

sudo echo "<h1> this is my 1st vpc </h1>" > /var/www/html/index.html

sudo systemctl restart httpd.service

sudo systemctl enable httpd.service

**vpc2:-**

1.create a vpc-2 -----------------------13.0.0.0/16

2.create a subnet--(subnet-vpc-2)-------13.0.1.0/24

3.create a IGW and attach to the vpc-2

4.create a route table--(rt-vpc-2)

associate---subnet-vpc-2----IGW

5.create a instance

1.name---------ec2-vpc-2

2.machine------amazon-linux

3.type---------t2.micro

4.key----------ec2-vpc-2 key

5.user-data

#!/bin/bash

sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd.service

sudo echo "<h1> this is my 2nd vpc </h1>" > /var/www/html/index.html

sudo systemctl restart httpd.service

sudo systemctl enable httpd.service

**vpc3:-**

1.create a vpc-3 -----------------------14.0.0.0/16

2.create a subnet--(subnet-vpc-3)-------14.0.1.0/24

3.create a IGW and attach to the vpc-3

4.create a route table--(rt-vpc-3)

associate---subnet-vpc-3----IGW

5.create a instance

1.name---------ec2-vpc-3

2.machine------amazon-linux

3.type---------t2.micro

4.key----------ec2-vpc-3 key

5.user-data

#!/bin/bash

sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd.service

sudo echo "<h1> this is my 3rd vpc </h1>" > /var/www/html/index.html

sudo systemctl restart httpd.service

sudo systemctl enable httpd.service

**vpc4:-**

1.create a vpc-4 -----------------------15.0.0.0/16

2.create a subnet--(subnet-vpc-4)-------15.0.1.0/24

3.create a IGW and attach to the vpc-4

4.create a route table--(rt-vpc-4)

associate---subnet-vpc-4----IGW

5.create a instance

1.name---------ec2-vpc-4

2.machine------amazon-linux

3.type---------t2.micro

4.key----------ec2-vpc-4 key

5.user-data

#!/bin/bash

sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd.service

sudo echo "<h1> this is my 4th vpc </h1>" > /var/www/html/index.html

sudo systemctl restart httpd.service

sudo systemctl enable httpd.service

**2.create a transit getway**

1.name-------tg-for-vpc1-vpc2-vpc3-vpc4

2.create a transit getway

**3.then create transit getway attachments-for-vpc-1**

1.name----------tg-attachment-vpc-1

2.tg ID---------(select)

3.type----------vpc

4.vpc ID--------(vpc-1)

5.create a tg attachments

**4.then create transit getway attachments-for-vpc-2**

1.name----------tg-attachment-vpc-2

2.tg ID---------(select)

3.type----------vpc

4.vpc ID--------(vpc-2)

5.create a tg attachments

**5.then create transit getway attachments-for-vpc-3**

1.name----------tg-attachment-vpc-3

2.tg ID--------(select)

3.type---------vpc

4.vpc ID-------(vpc-3)

5.create a tg attachments

**6.then create transit getway attachments-for-vpc-4**

1.name----------tg-attachment-vpc-4

2.tg ID--------(select)

3.type---------vpc

4.vpc ID-------(vpc-4)

5.create a tg attachments

**7.then updates the routs of vpc-1 in vpc-1 we can add (vpc2-vpc3-vpc4)**

1.add route------13.0.0.0/16 ----------transit getway-------------save

2.add route------14.0.0.0/16-----------transit getway-------------save

3.add route------15.0.0.0/16-----------transit getway-------------save

**8.then update the routs of vpc-2 in vpc-2 we can add (vpc1-vpc3-vpc4)**

1.add route------12.0.0.0/16 ----------transit getway-------------save

2.add route------14.0.0.0/16-----------transit getway-------------save

3.add route------15.0.0.0/16-----------transit getway-------------save

**9.then update the routs of vpc-3 in vpc-3 we can add (vpc1-vpc2-vpc4)**

1.add route------12.0.0.0/16 ----------transit getway-------------save

2.add route------13.0.0.0/16-----------transit getway-------------save

3.add route------15.0.0.0/16-----------transit getway-------------save

**10.then update the routs of vpc-4 in vpc-4 we can add (vpc1-vpc2-vpc3)**

1.add route------12.0.0.0/16 ----------transit getway-------------save

2.add route------13.0.0.0/16-----------transit getway-------------save

3.add route------14.0.0.0/16-----------transit getway-------------save

**11.then test the instances**

**12.instance----ec2-vpc-1----connect**

1.vim vpc-1 key.pem

2.vim vpc-2 key.pem

3.vim vpc-3 key.pem

4.vim vpc-4 key.pem

chmod 400 vpc-1 key.pem

chmod 400 vpc-2 key.pem

chmod 400 vpc-3 key.pem

chmod 400 vpc-4 key.pem

**1.ssh -i vpc-1 key.pem ubuntu@(ip of vpc-1)**

curl (private ip of vpc-2)

curl (private ip of vpc-3)

curl (private ip of vpc-4)

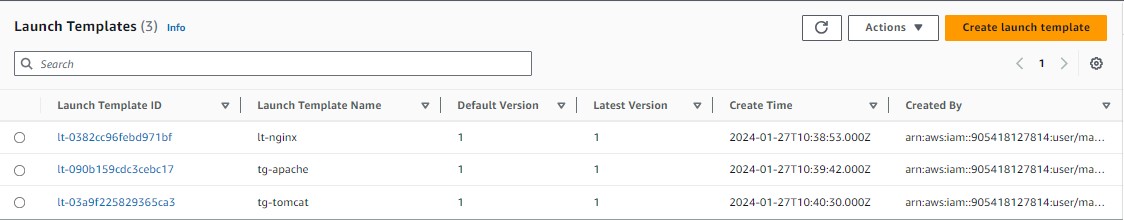
===========================================================================

**LOAD BALANCER WITH AUTOSCALING WITH THE RULE 40-40-20 WEIGHTED POLICY**

**1.create a template--nginx-tg**

**2.create a template--apache-tg**

**3.create a template—tomcat-tg**



**4.create a autoscaling group(nginx)**

1.name ------ASG-nginx

2.template---nginx-lt

3.az and subnets----1a,1b,1c

4.capacity

desired-------1

minimum----1

maximum----2

5.add sns

6.tag---Name-nginx

7.create a autoscaling group

**5.create a autoscaling group(apache)**

1.name ------ASG-apache

2.template---apache-lt

3.az and subnets----1a,1b,1c

4.capacity

desired-------1

minimum----1

maximum----2

5.add sns

6.tag---Name-apache

7.create a autoscaling group

**6.create a autoscaling group(tomcat)**

1.name ------ASG-tomcat

2.template---tomcat-lt

3.az and subnets----1a,1b,1c

4.capacity

desired-------1

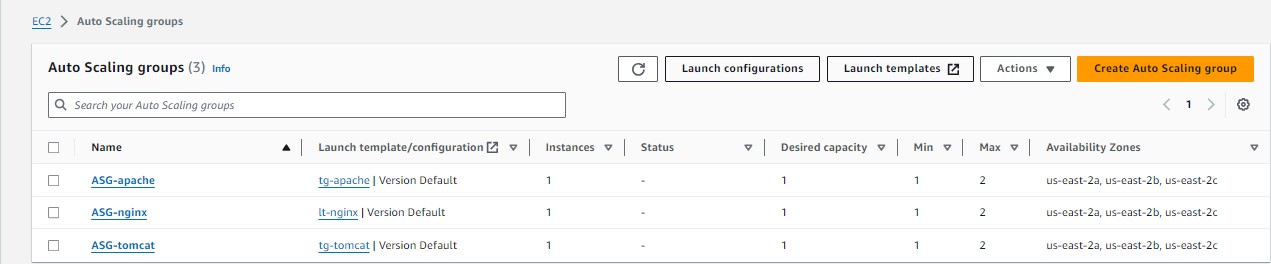
minimum----1

maximum----2

5.add sns

6.tag---Name-tomcat

7.create a autoscaling group



**7.create a target group (nginx)**

1.name------tg-nginx

2.type-------instance

3.path---------/

4.register-------nginx

**8.create a target group (apache)**

1.name------tg-apache

2.type-------instance

3.path---------/

4.register-------apache

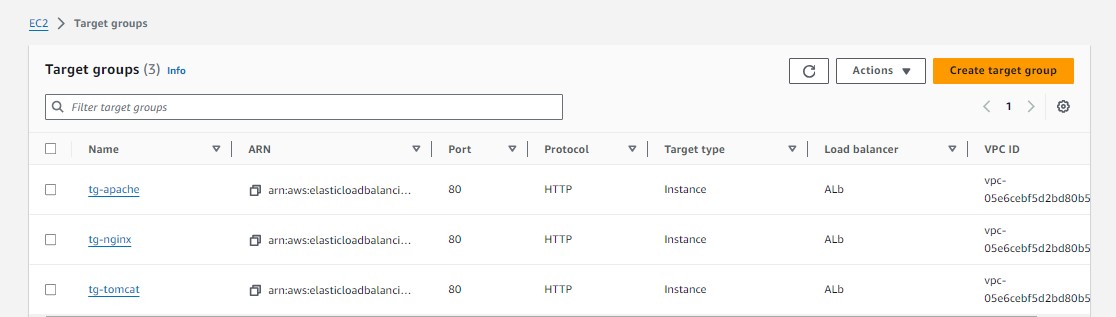
9. **create a target group (tomcat)**

1.name------tg-tomcat

2.type-------instance

3.path---------/

4.register-------tomcat-----8080----include as pending below



**10.create a load balancer**

1.name-----------------ALB

2.default action-------tg-nginx

3.create a target group

**11.click on created ALB**

**1.select listner------click on listner**

**2.action--------------edit listner**

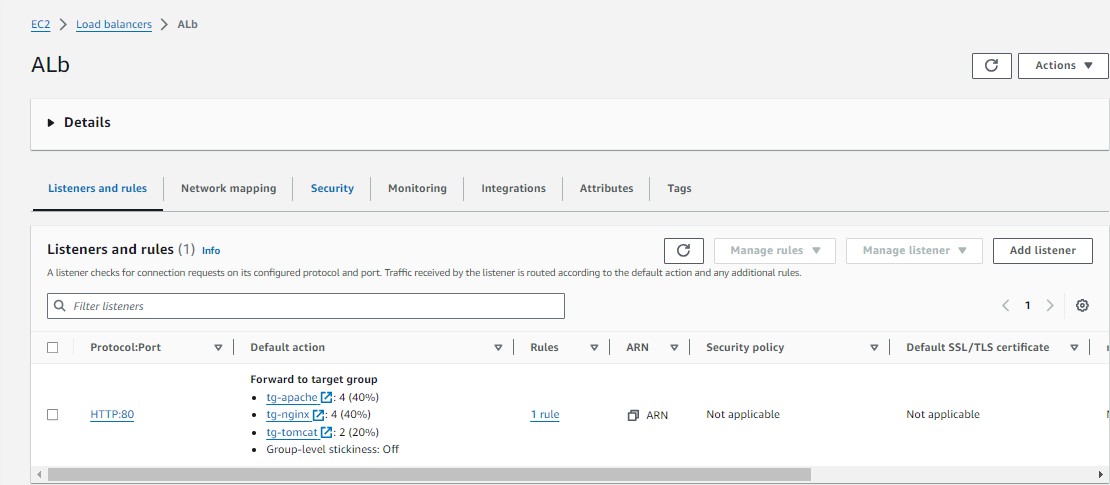
**3.add target group-----tg-apache and tg-tomcat**

**4.add the weighted policy**

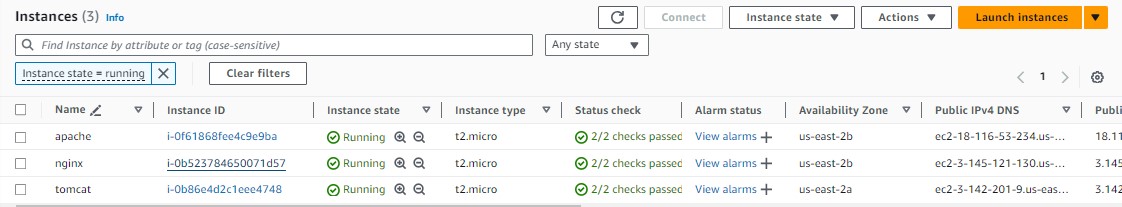
**1. tg-nginx----------4---------------40%**

**2.tg-apache--------4 --------------40%**

**3.tg-tomcat---------2--------------20%**



**12.check the instances that we create**



**13.click on created instance – nginx---connect**

1.sudo apt update -y

2.sudo apt install nginx -y

3.sudo systemctl start nginx

**14.click on created instance--apache----connect**

1.sudo apt update -y

2.sudo apt install apache2 -y

Sudo systemctl start apache2

**15.click on created instance--tomcat---connect**

1.sudo apt update -y

2.apt install java-openjdk-11 -y (apt install default-jre -y) / (apt install default-jdk -y)

3.wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.85/bin/apache-tomcat-9.0.85.tar.gz

4.tar -xvf apache-tomcat-9.0.85.tar.gz

5.cd apache-tomcat-9.0.85

6.cd bin/

7. ./catalina.sh start

**12.click on application load balancer that you created**

1.copy the dns and paste

**13.the traffic is on nginx 40%,apache 40% and in tomcat about 20%**

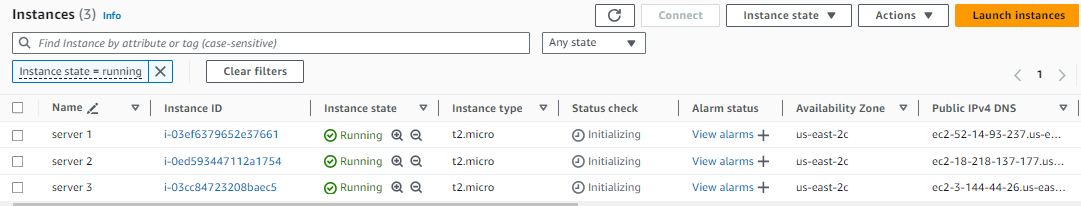
===========================================================================

**NGINX AS A LOAD BALANCER**

1.create a instance [load balancer(server 1)

2.create a instance server 2

3.create a instance server 3



4.access the instance (server 2)

1.sudo apt udpate -y

2.sudo apt install nginx -y

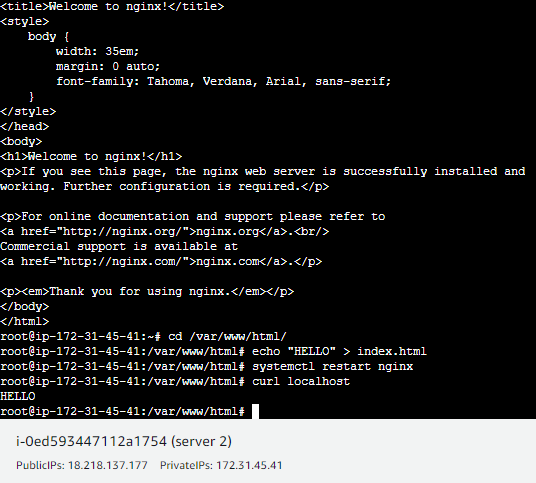
3.sudo systemctl start nginx

4.sudo systemctl enable nginx

5.cd /var/www/html/

6.echo "HELLO" > index.html

7.sudo systemctl restart nginx



5.access the instance (server 3)

1.sudo apt udpate -y

2.sudo apt install nginx -y

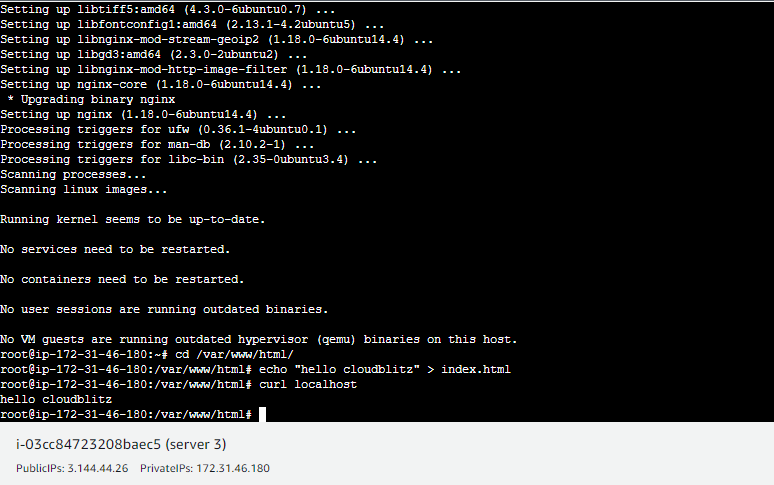
3.sudo systemctl start nginx

4.sudo systemctl enable nginx

5.cd /var/www/html/

6.echo "hello cloudblitz" > index.html

7.sudo systemctl restart nginx



4.access the instance [load balancer(server 1)

1.sudo apt udpate -y

2.sudo apt install nginx -y

3.sudo systemctl start nginx

4.sudo systemctl enable nginx

5.then go to nginx conf file- vim /etc/nginx/nginx.conf

6.paste this in nginx.conf

events {

worker\_connections 768;

}

http {

upstream myapp

{

server Server\_1\_Address weight=1;

server Server\_2\_Address weight=1;

}

#These are my settings SHROBON

server {

listen 80;

location / {

proxy\_pass http://myapp ;

}

}

}

7. systemctl restart nginx

8. hit the loadbalancer ip in new tab

===========================================================================

**CREATE A DATABASE (RDS)-AND CREATE TWO USERS**

**1.create a instance(rds)**

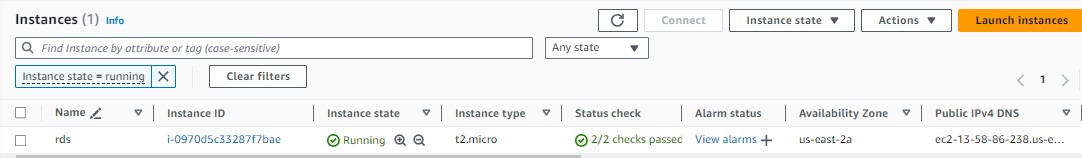
1.name--------rds

2.machine-----amazon linux

3.type-----------t2.micro

4.sg--------------http/https/ssh/mysql-aurora(3306)

5.create a instance



**2.create a database (rds)**

1.standard

2.engine-------------mariadb

3.tempalte-----------free tier

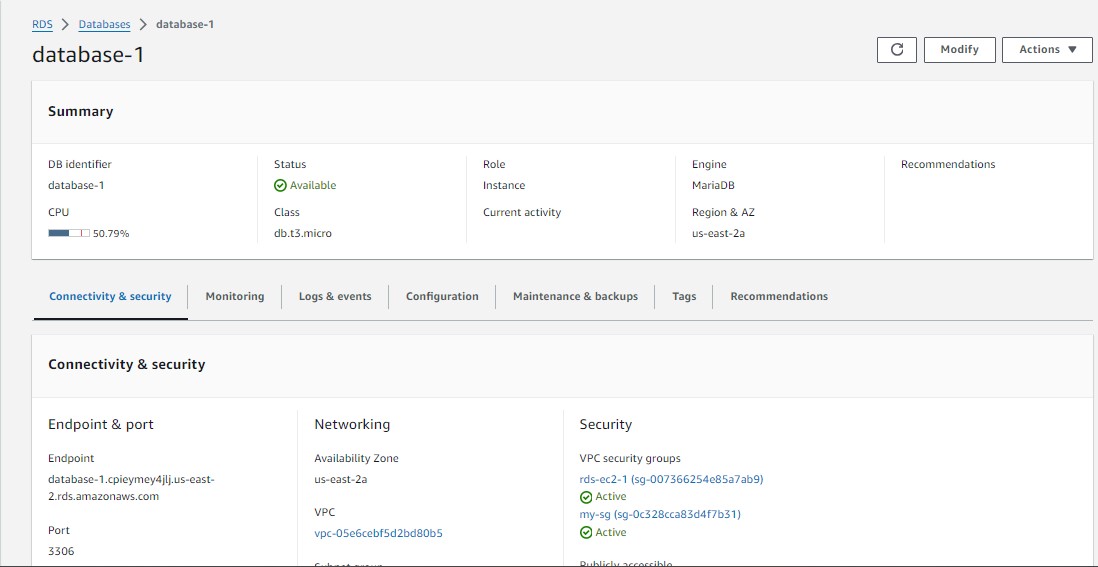
4.master username----admin

5.master password----admin123

6.connect to EC2 ----rds

7.existing vpc sg----(attach sg)

8.additional conf----student



**3.access the instance (rds)**

1.yum install mariadb\* -y

2.systemctl start mariadb

3.mysql -u admin -h (endpoint) -padmin123

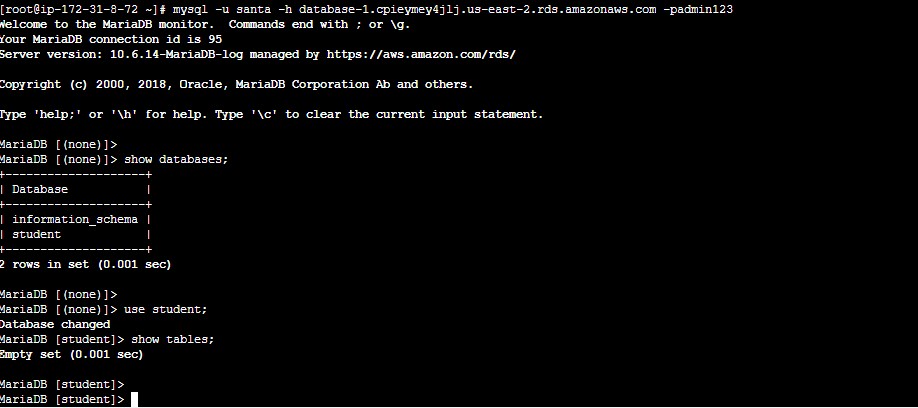
1.create user santa identified by 'admin123';

2.create user banta identified by 'admin123';

3.GRANT ALL PRIVILEGES ON student.\* TO 'santa';

4.ctrl+c

**4.** **mysql -u santa -h (endpoint) -padmin123**



1.show databses;

2.use student;

1.create table student\_info (serial\_no int,student\_name varchar(500),address

varchar(500),mobile\_no bigint);

2.insert into student\_info values (4,'shubham','palasgaon bai',7875863620);

3.insert into student\_info values (2,'bhushan','palasgaon bai',7066329821);

4.insert into student\_info values (3,'vaibhav','palasgaon bai',7378488213);

5.insert into student\_info values (1,'mayur','palasgaon bai',7378644515);

6.insert into student\_info values (5,'sameer', 'palasgaon bai',9090442312);

7.insert into student\_info values (6,'swapnil', 'palasgaon bai',9090442312);

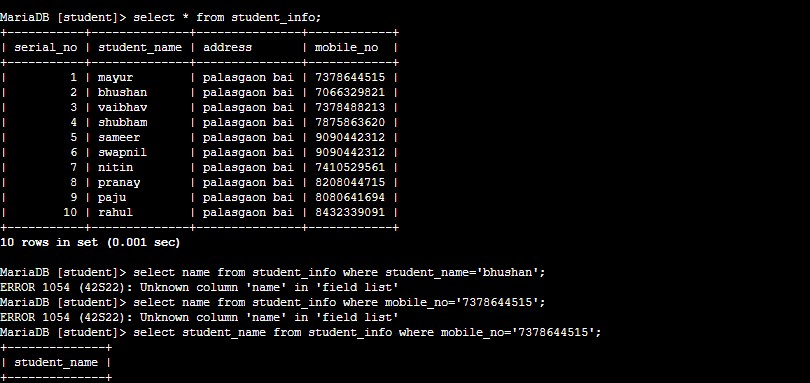
8.insert into student\_info values (7,'nitin', 'palasgaon bai',7410529561);

9.insert into student\_info values (8,'pranay', 'palasgaon bai',8208044715);

10.insert into student\_info values (9,'paju', 'palasgaon bai',8080641694);

11.insert into student\_info values (10,'rahul', 'palasgaon bai',8432339091);

3.select \* from student\_info;



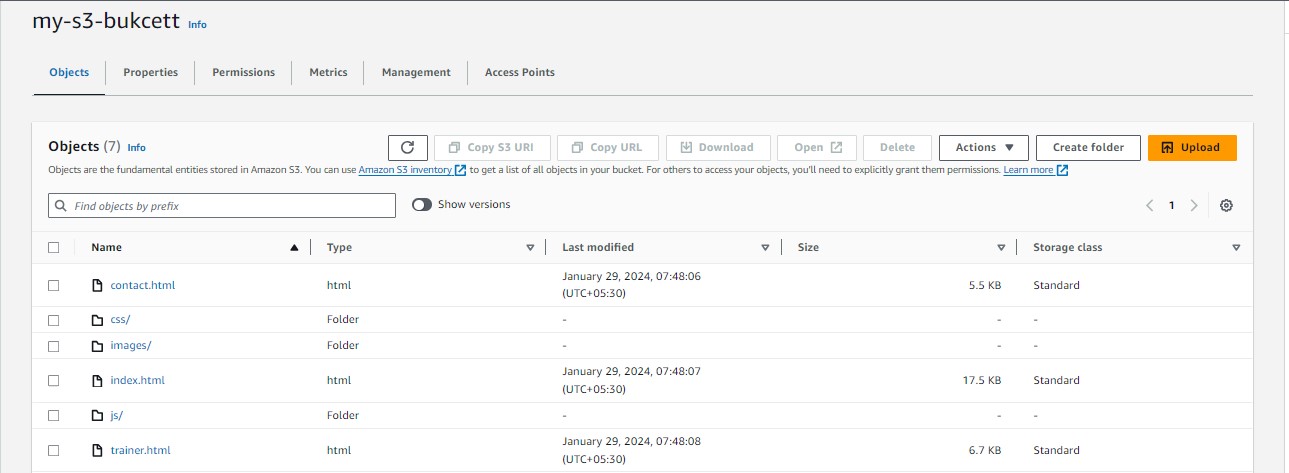
4.select student\_name from student\_info where mobile\_no='7378644515';

**S3 WITH CLOUDFRONT BUCKET SHOULD BE PRIVATE**

**1.download a css template**

**2.create a s3 bucket which is private**

1.add these template in s3 bucket



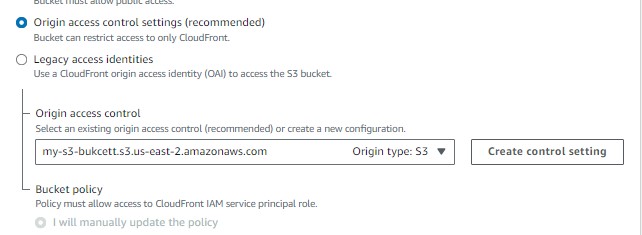
**3.create a cloudfront**

1.create a distribution

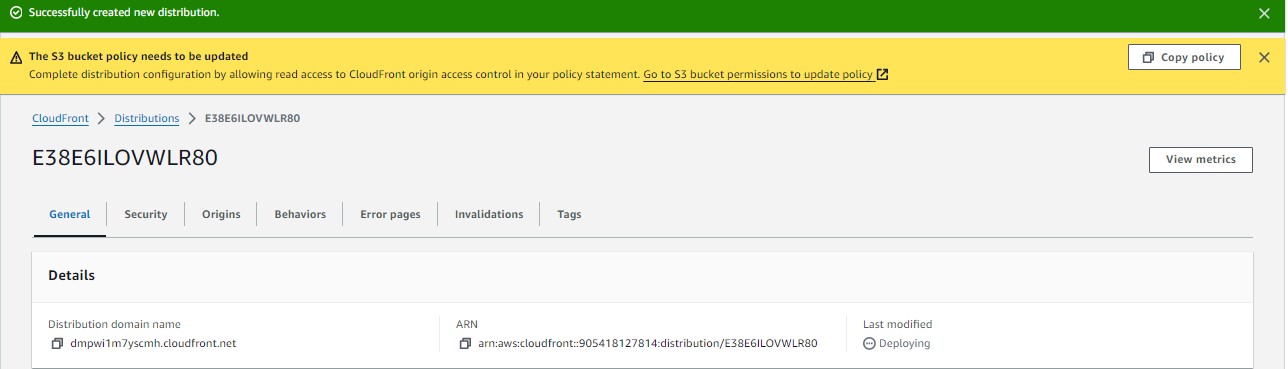
2.choose destination----s3 bucket

3.click on Origin access control settings------and create a control setting------create

4.create a distribution



**4.then you get a notification and copy these policy to s3 bucket**



**5.copy the dns and hit on web**

{ (dns)/index.html }



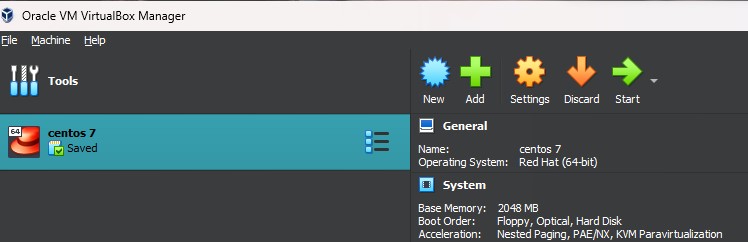
================================================================

Project-1

**CREATE A DATABASE IN CENTOS 7 AND MIGRATE THESE DATABASE IN CLOUD**

**1.download a virtual box.**

**2.install a centos7 machine in virtual box.**



**3.install mariadb on centos7**

1.yum install mariadb-server -y

2.systemctl start mariadb,service

**4.set the password**

mysql\_secure\_installation

**5.enter in mariadb**

1.mysql -u root -h localhost -pmayur

1.show database;

2.create database student\_data;

3.use student\_data;

4.create table student\_info (name varchar(10),roll\_no int,status varchar(10));

5.show tables;

6.describe student\_info;

7.insert into student\_info values (‘user1’,100,’fail’);

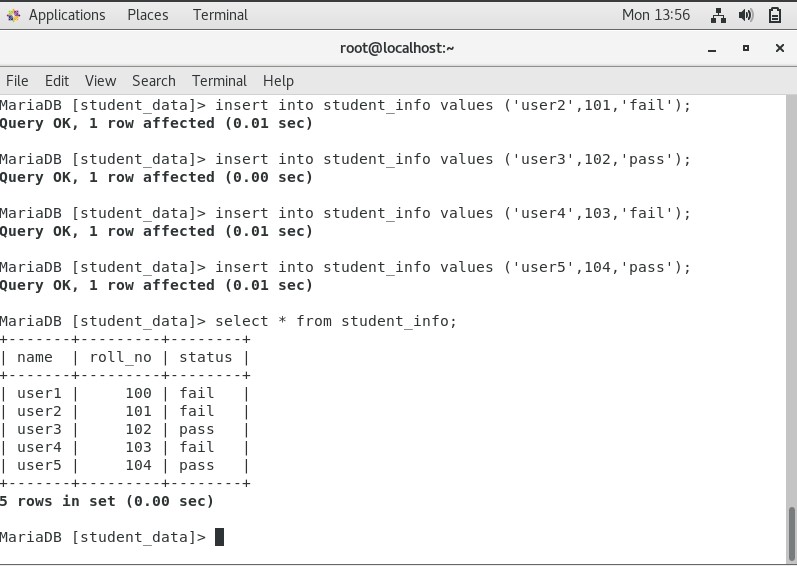
insert into student\_info values (‘user2’,101,’pass’);

insert into student\_info values (‘user3’,102,’fail’);

insert into student\_info values (‘user4’,103,’pass’);

insert into student\_info values (‘user5’,104,’fail’);

8.select \* from student\_info;



**6.to create a backup**

1.mysqldump -u root -h localhost -pmayur student\_data >abc.bkp

2.cat abc.bkp



**7.Then transfer these file in local machine. You can use scp either any command I am**

**using samba server to transfer these files in my local machine**

1.install samba server in your centos7

1.yum install samba -y

2.mkdir /access

cp abc.bkp /access

3.useradd new

4.passwd new

5.vim /etc/samba/smb.conf

[access\_dir]

path = /access

writable = yes

browseable = yes

Public = yes

Write list = new

Read list = new

Valid users = new

6.give directory permission to new user

Setfacl -m u:new:rwx /access

7.set samba password for new

Password: mayur

8.change selinux to permissive

Setenforce permissive

9.download samba-client

Yum install samba-client

Systemctl start smb

Systemctl start nmb

10.check we can access samba or not

smbclient -U new //localhost/access\_dir

**11.attach samba to windows----cross connectivity**

1.go to network status

2.change adapter option

3.virtual box only adapter ---------------------------to see ip

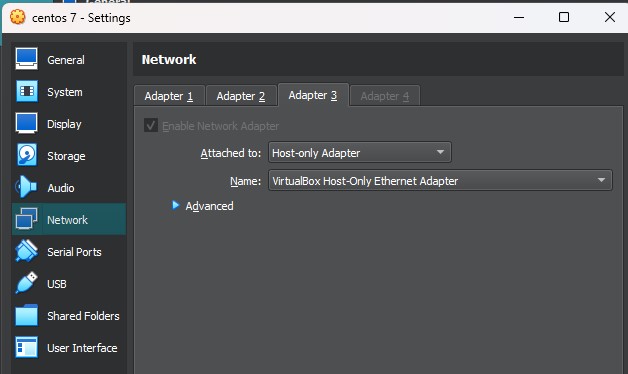
192.168.56.1 -------------------------------------ip address

255.255.255.0 -----------------------------------netmask

4.In centos machine

1.add the adapter

-Host only adapter-



5.then in centos machine the the host-adapter ip that starts with 192

192.168.56.108

6.then ping these ip in window

192.168.56.108

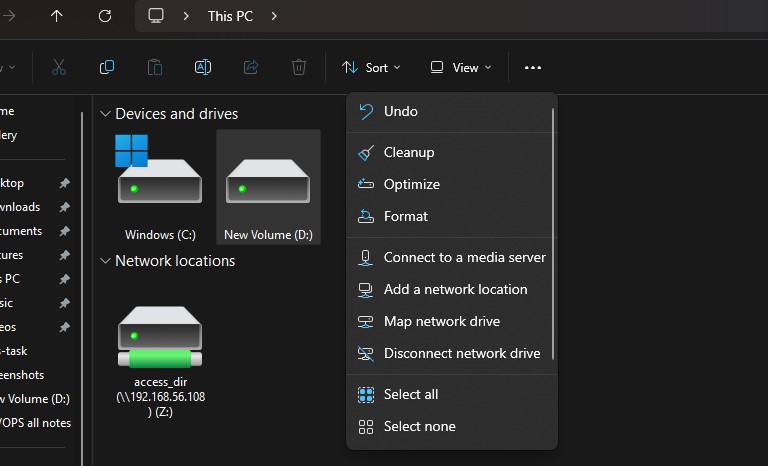
7.and ping your local machine ip in centos

192.168.56.1

8.firewall-cmd --add-service=samba

9.then go to desktop

1.map network drive



2.\\192.168.56.108\access

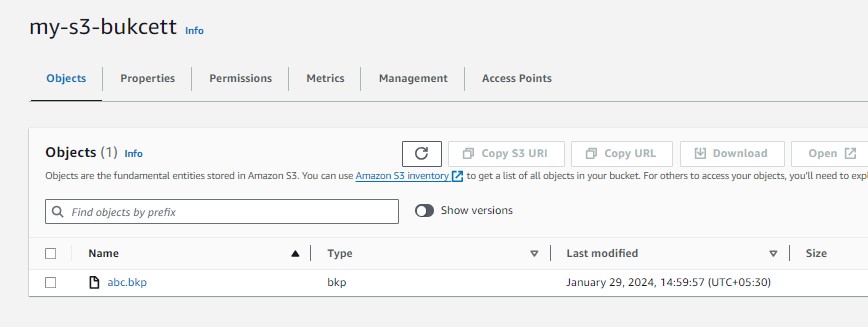
Give the username--------new

Password ------------------mayur

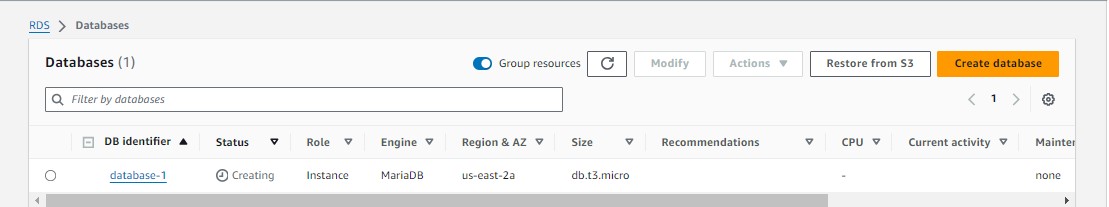
3.you can connect to the linux

1.then you can access all the files that are in access copy these abc.bkp file

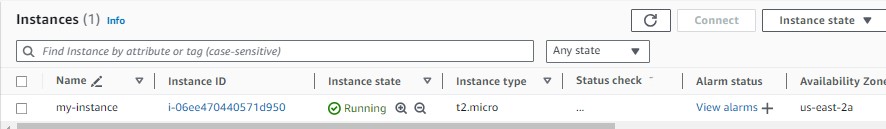
**8.create a s3 bucket and add abc.bkp file in s3 bucket**



**9.create a RDS(relational database)**



**10.create a instance**



**11.Then create a role that have s3 bucket full access and attach to the instance**

**12.access the instance**

1.aws s3 ls

3.aws s3 ls s3://my-s3-bucket

3.aws s3 cp s3://my-s3-bucket/abc.bkp /root

4.ls

abc.bkp

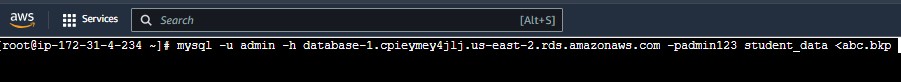
**13.Then install mariadb in server**

1.Yum install mariadb -y

2.Systemctl start mariadb.service

**14.Retrive these backup in rds**

1.mysql -u admin -h (endpoint) -padmin123 student\_data <abc.bkp



**15.Then check the database**

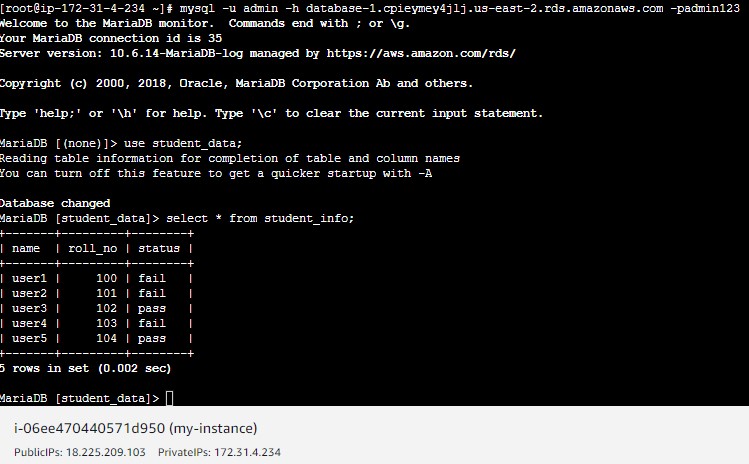
1.mysql -u admin -h(endpoint) -padmin123

1.show database;

2.use student\_data;

3.describe student\_info;

4.select \* from student\_info;



**PROJECT:-2**

**Migrate on premises virtual machine on cloud**

**1.set up tomcat on virtual machine(ubuntu)**

1.sudo apt upgrade -y

1.apt install openjdk-11-jdk -y

2.wget <https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.85/bin/apache-tomcat->

9.0.85.tar.gz

3.tar -xvf apache-tomcat-9.0.85.tar.gz

4.rm -rf apache-tomcat-9.0.85.tar.gz

5.cd apache-tomcat-9.0.85

6.cd bin/

7. ./Catalina.sh start

8.cd

9.firewall-cmd –add-port=8080/tcp

10.curl localhost:8080

**2.set up nginx on virtual machine**

1.apt update -y

2.apt install epel-release -y

3.apt install nginx -y

4.systemctl start nginx

5.systemctl enable nginx

6.vim /etc/nginx/sites-available/default

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass http://localhost:8080;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

# Additional Nginx configurations if needed

}

7.systemctl restart nginx

8.curl localhost

You get tomcat

**##if u get any error while u edit nginx file run the command----{nginx -t} u will know the error**

**3.then export your machine**

1.go to machine(virtual box)

2.click on export appliance

3.give the extension (ovf)(ova)

**4.then go to folder and check properly export or not**

1.in these there are three file should be create

**5.then create s3 bucket (my-s3-bucket)**

1.add these file that you give the extention

**6.create a access key and secret key for your aws cli in windows to configure**

**7.create a IAM role name “vmimport”**

1.create a file trust-policy.json in your desktop(text file)

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": { "Service": "vmie.amazonaws.com" },

"Action": "sts:AssumeRole",

"Condition": {

"StringEquals":{

"sts:Externalid": "vmimport"

}

}

}

]

}

2.go to cmd…configure and create these role

aws iam create-role --role-name vmimport --assume-role-policy-document

<file://trust-policy.json>

note:- if u save these file in desktop then you can also work on these directory

otherwise you can get the full path of these file

**8.create another file name role-policy.json and change the name of your bucket**

{

"Version":"2012-10-17",

"Statement":[

{

"Effect": "Allow",

"Action": [

"s3:GetBucketLocation",

"s3:GetObject",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::my-s3-bucket",

"arn:aws:s3:::my-s3-bucket/\*"

]

},

{

"Effect": "Allow",

"Action": [

"s3:GetBucketLocation",

"s3:GetObject",

"s3:ListBucket",

"s3:PutObject",

"s3:GetBucketAcl"

],

"Resource": [

"arn:aws:s3:::my-s3-bucket",

"arn:aws:s3:::my-s3-bucket/\*"

]

},

{

"Effect": "Allow",

"Action": [

"ec2:ModifySnapshotAttribute",

"ec2:CopySnapshot",

"ec2:RegisterImage",

"ec2:Describe\*"

],

"Resource": "\*"

}

]

}

2.create a policy for these role

aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document

<file://role-policy.json>

**9.you can also move these exported file with command**

aws s3 mv "/home/administrator/Documents/ubuntu.ova" s3://my-s3-bucket

note:- change the bucket name as well as change the path for your exported file

**10. Import the VM image to AWS AMI**

1. After you upload your VM image file to Amazon S3, you can use the AWS CLI to

import the image.

2. Create another file named containers.json with the following policy.

[

{

"Description": "My Server OVA",

"Format": "ova",

"UserBucket": {

"S3Bucket": "my-s3-bucket",

"S3Key": "ubuntu.ova"

}

}

]

NOTE:- you can change the file format as well as bucket name also change the key name

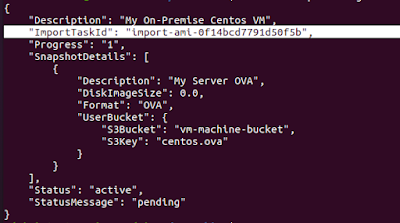
Your key name is your uploaded file name

3.then run the command to import image

aws ec2 import-image --description "My On-Premise Centos vm" --disk-containers

<file://containers.json>

\*\* after importing you will get output like these\*\*



**11.monitor images import task**

\*\*Note: Replace "import-ami-1234567890abcdef0" from above output ImportTaskId\*\*

Run these command to check the status

aws ec2 describe-import-image-tasks --import-task-ids import-ami-0f14bcd7791d50f5b

\*\*Status values include the following\*\*

active — The import task is in progress.

deleting — The import task is being canceled.

deleted — The import task is canceled.

updating — Import status is updating.

validating — The imported image is being validated.

validated — The imported image was validated.

converting — The imported image is being converted into an AMI.

completed — The import task is completed and the AMI is ready to use

NOTE:- after get status of completed you can check the ami

**12. After Completed import image task, Let's check AMI section**

**13. The ami image is seen you can Launch EC2 Instance through that AMI**

**BLUE GREEN DEPLOYMENT**

----Blue Green Deployment----

**Step.1 -first we will launch two instance with different name like blue and green.**

for Blue-instance:(this is our live environment which run on old version)

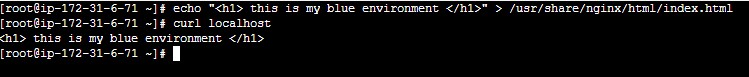
1. sudo -i (for root user access)

2. yum install nginx -y

3. echo “<h1> This is my blue environment </h1>” >/usr/share/nginx/html/index.html

4. systemctl restart nginx

5.systemctl enable nginx



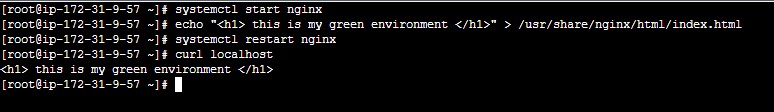
For Green-instance:(this is our green environment which run on new or update version)

1. sudo -i (for root user access)

2. yum install nginx -y

3. echo “<h1> This is green environment </h1>” > /usr/share/nginx/html/index.html 4. systemctl restart nginx

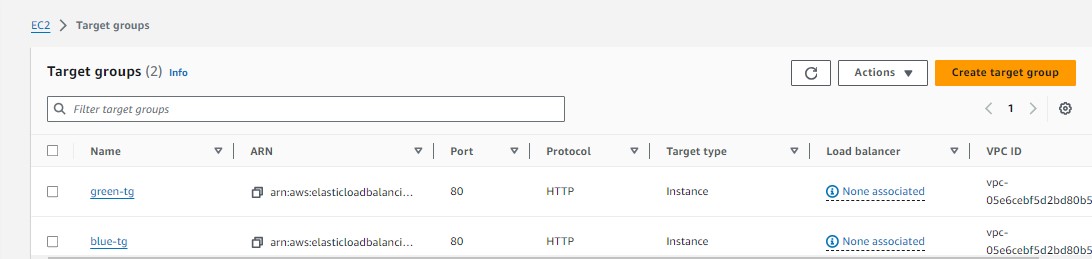
5.systemctl enable nginx



**Step.2 create Two target group**

1. blue-tg: blue-instance will register on this tg.

2. green-tg: green-instance will register on this tg.



**Step.3 Create an application load balancer**

1.first we will add blue-tg on load balancer. and now 100% traffic will come on blue-

tg and load balancer distribute this traffic to its registered instance.

2.now we will add green-tg which is updated and deploy it with load balancer listener

where i have already a listener on this load balancer

3.go to listener and rules and click on listener port and then go to action and click on

edit listener rule scroll down and here is already blue-tg now click on add target

group🡺 add green tg with 20% weight and blue-tg with 80% weigh🡺save changes

4.now hit the load balancer. And see 80% traffic routed on blue while 20% on green

5.if our green environment working good and have no other issue on this enviromnet

we will swicth the all traffic with this green enviromnet.

6.These environment little bit costly because we have already create the servers to

Test The load

