17-03-2024

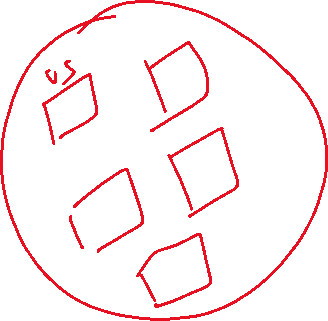
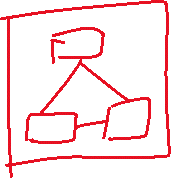
**AWS:**

* 32 geographical ‘Regions’.

Data Centres

Availability Zone

Regions



* Certifications:

AWS Solutions architect

AWS certified SysOps Administrator

AWS Certified Developer

* Agenda:

1. Understanding Region, AZ
2. AWS Services
3. Understanding of EC2
4. Practical experience of EC2
5. Click on:

Services -> Compute -> EC2 -> Launch Instance

* 1. Name and tags: KhushiServer1
  2. AMI: Microsoft Windows Server 2022 Base
  3. Instance Type: t2.micro, 1CPU, 1GB
  4. Key Pair: KhushiServer1key.pem
  5. Network settings -> Edit
     1. VPC - required Info: default
     2. Subnet Info: No preference
     3. Auto-assign public IP Info: Enable
     4. Firewall (security groups)

-> Create Security Group

-> Security group name – required : KhushiServer1SG1

-> Description – required

-> Inbound Security Group Rules:

Type Info: rdp, Protocol: TCP, Port range: 3389

Source IP: Anywhere

* 1. Storage configuration
  2. Launch Instance

1. Web server creation and setup
2. Deploy static website using windows server

Public IP: 3.110.27.35

DNS: ec2-3-110-27-35.ap-south-1.compute.amazonaws.com

Username: Administrator

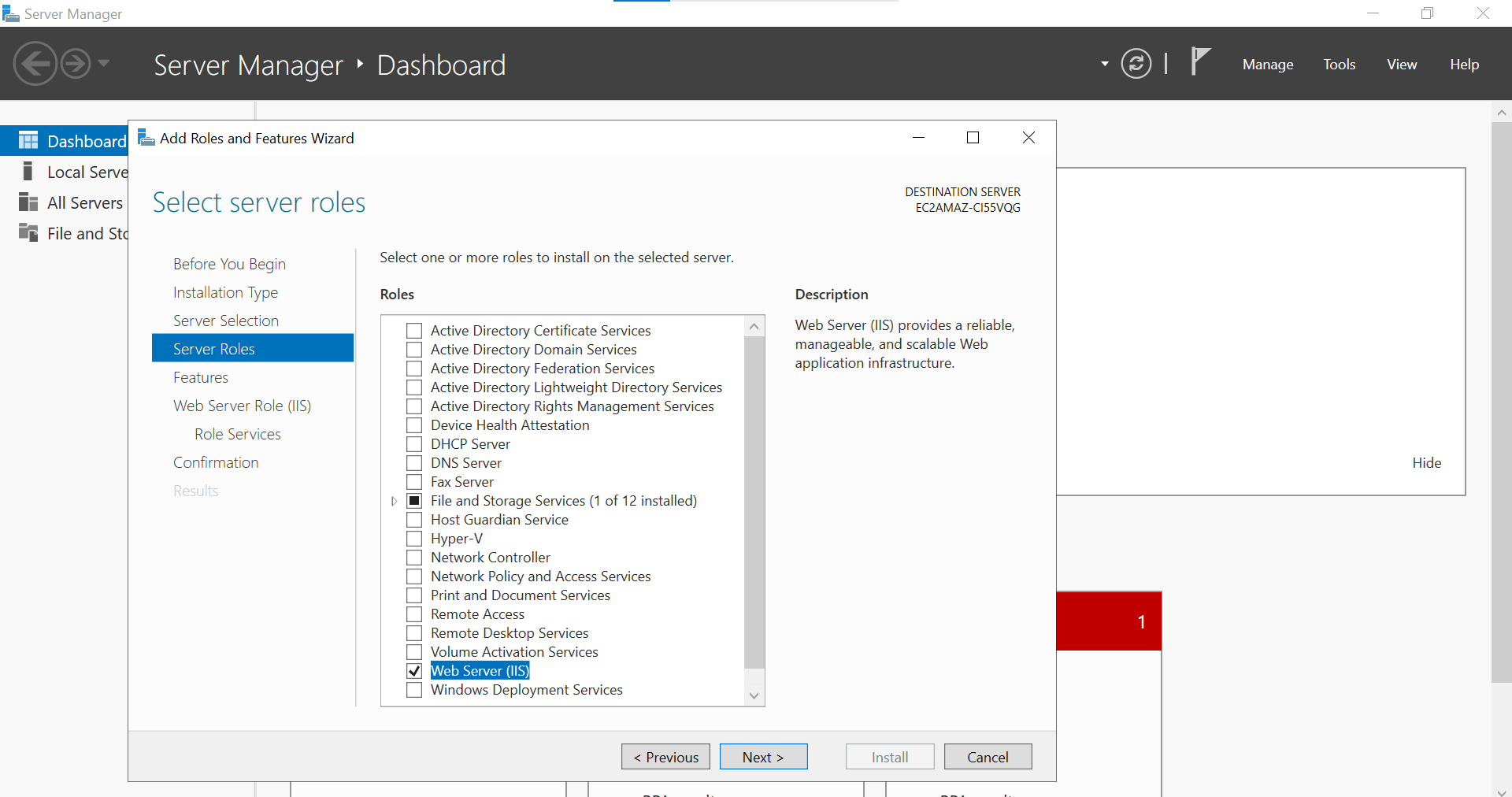
Password: KICX(h2kS(SQh;O@LR9C5o?uhW\*z&W4y

Select EC2 Instance -> connect -> rdp client -> get password -> decrypt

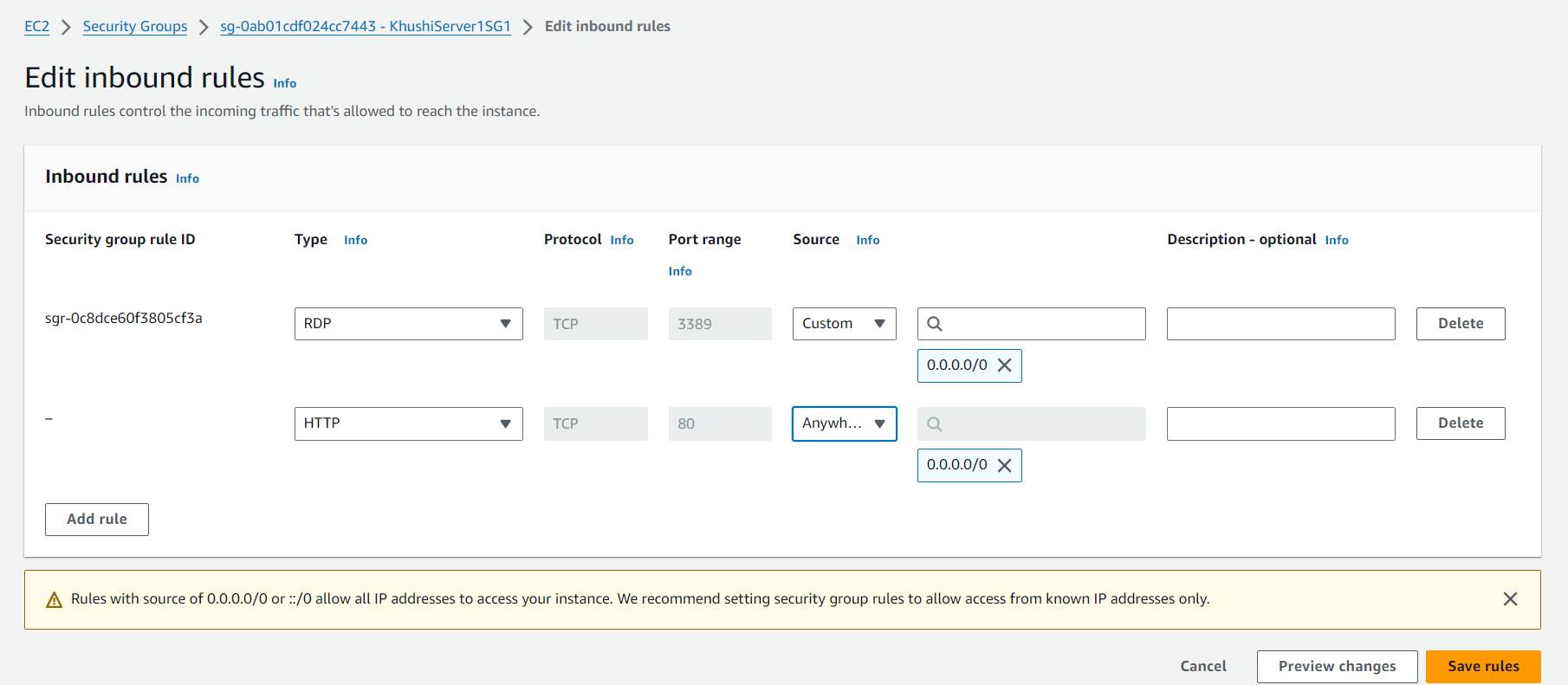
Port: 3389

Open “Remote Desktop Connect” from your local PC -> Public IP Of instance -> Connect-> Username & Password

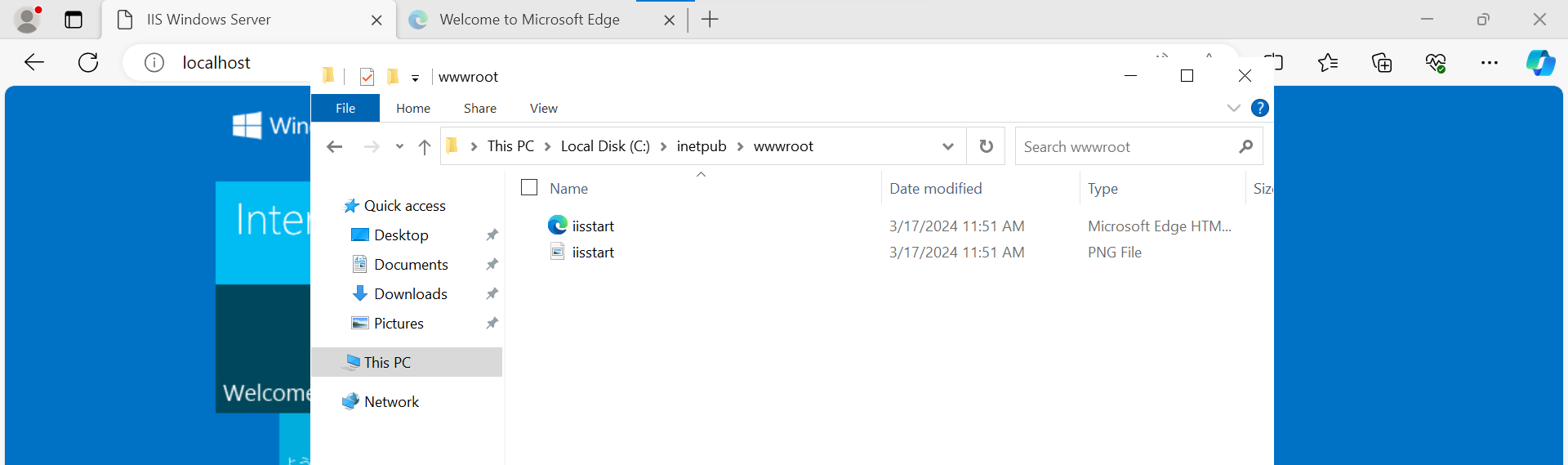
Open control panel in remote windows, -> programs-> turn windows feature on/off



To access the web from anywhere: change this setting, add inbound rule



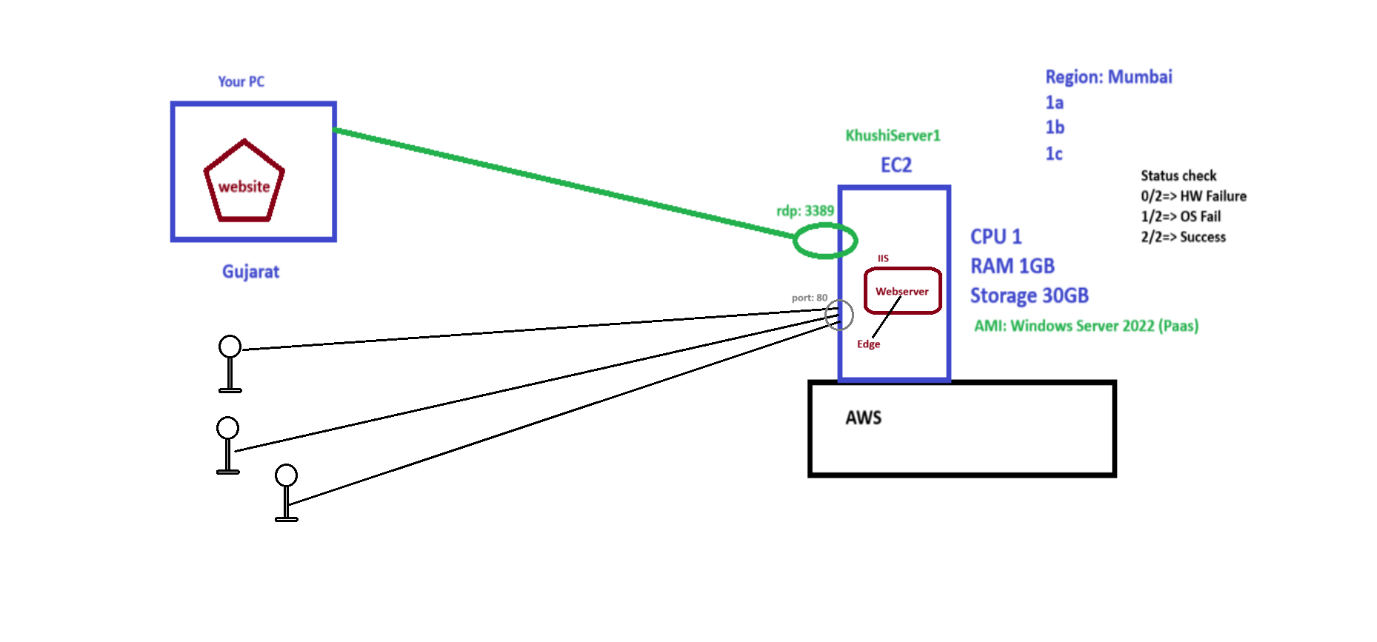
Open remote desktop:



Modify or create the html file/website according to you!

1. Termination of Instance

* Select instance -> actions -> instance setting -> change termination protection
* Select Instance -> Instance state -> Terminate



--------------------------------------------------------------------------------------------------------------------------------------

Create react app

npm run build

npm install -g serve

serve -s build

* Create zip folder of the build folder
* AWS Amplify -> Amplify hosting -> Deploy without Git provider -> App name: cloudApp, Environment name: Staging, drag & drop
* Copy the link
* Actions -> delete app

(100ms api – for google meet to integrate in your react application)

--------------------------------------------------------------------------------------------------------------------------------------

* Create EC2 instance using Linux

1. Services -> Compute -> EC2 -> Launch Instance
   1. Name and tags: KhushiServer2
   2. AMI: Ubuntu Server 22.04 LTS
   3. Instance Type: t2.micro, 1CPU, 1GB
   4. Key Pair: KhushiServer2key.pem

- Max, linux, git -> pem key

- Sftp -> ppt key

* 1. Network settings -> Edit
     1. VPC - required Info: default
     2. Subnet Info: No preference
     3. Auto-assign public IP Info: Enable
     4. Firewall (security groups)

-> Create Security Group

-> Security group name – required : KhushiServer2SG2

-> Description – required

-> Inbound Security Group Rules:

Type Info: ssh, Protocol: TCP, Port range: 22

Source IP: Anywhere

* 1. Storage configuration: 10 gb
  2. Launch Instance
* IPV4: 13.201.127.175
* User: ubuntu

for red het linux/ amazon linux : ec2 ubuntu

* Key/Password: KhushiServer2key.pem
* Port: 22
* Commands:
* curl ifconfig.me
* ls \*.pem
* ls -la \*.pem
* chmod 400 KhushiServer2key.pem
* ls -la \*.pem
* ssh -i "KhushiServer2key.pem" [ubuntu@13.201.127.175](mailto:ubuntu@13.201.127.175)

(ssh -i “key.pem” username@ip)

* sudo apt update
* sudo su
* apt update
* cat /etc/passwd
* adduser Khushi

password: 1234

* cd /etc/ssh
* vi sshd\_config

(make below changes)

PasswordAuthentication yes

kbdInteractiveauthentication yes

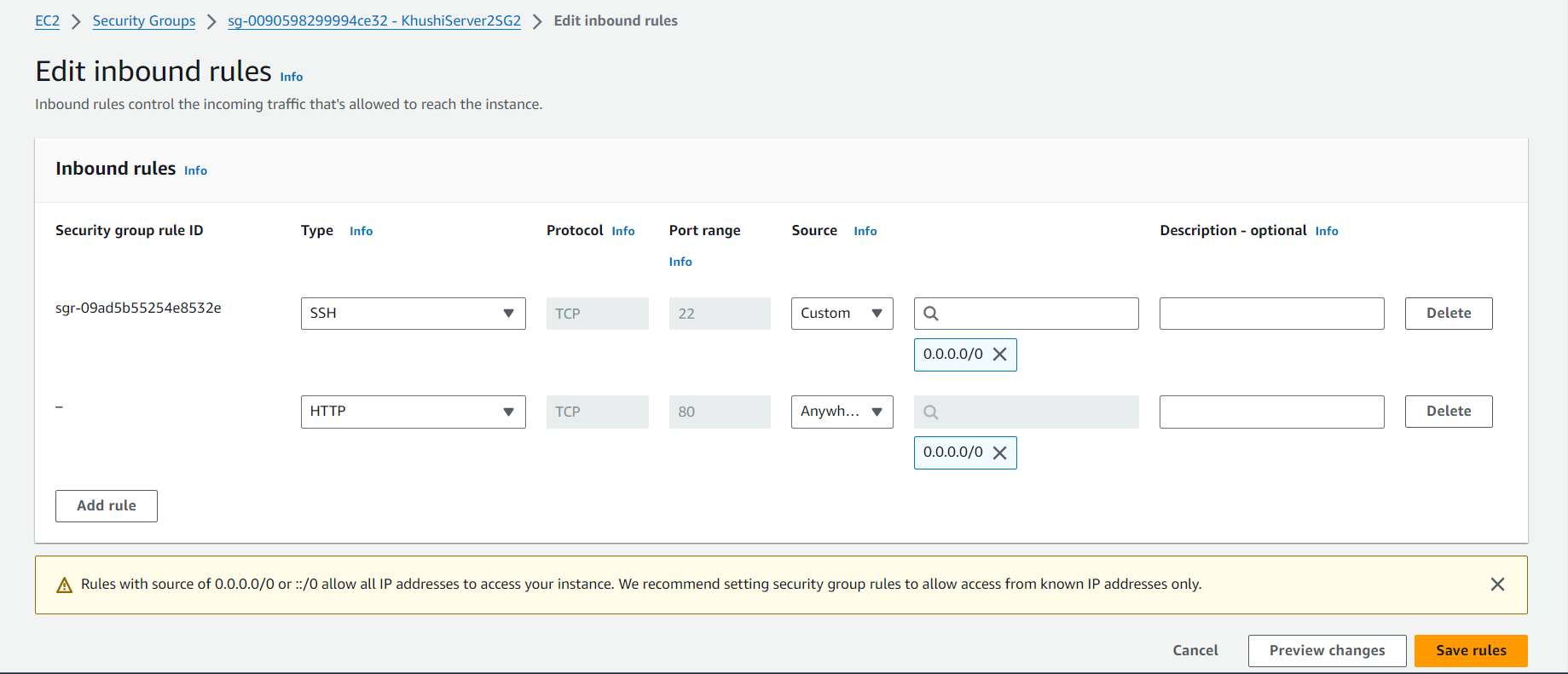
(press `Esc` -> :wq)

* systemctl restart sshd
* systemctl restart ssh
* ssh khushi@ 13.201.127.175 (windows terminal)
* tmux
* tmux ls
* vi abc.c
* cat abc.c
* ssh khushi@ 13.201.127.175 (windows terminal)
* tmux attach -t 0
* to check public ip: curl ifconfig.me
* to check private ip:

sudo apt install net-tools

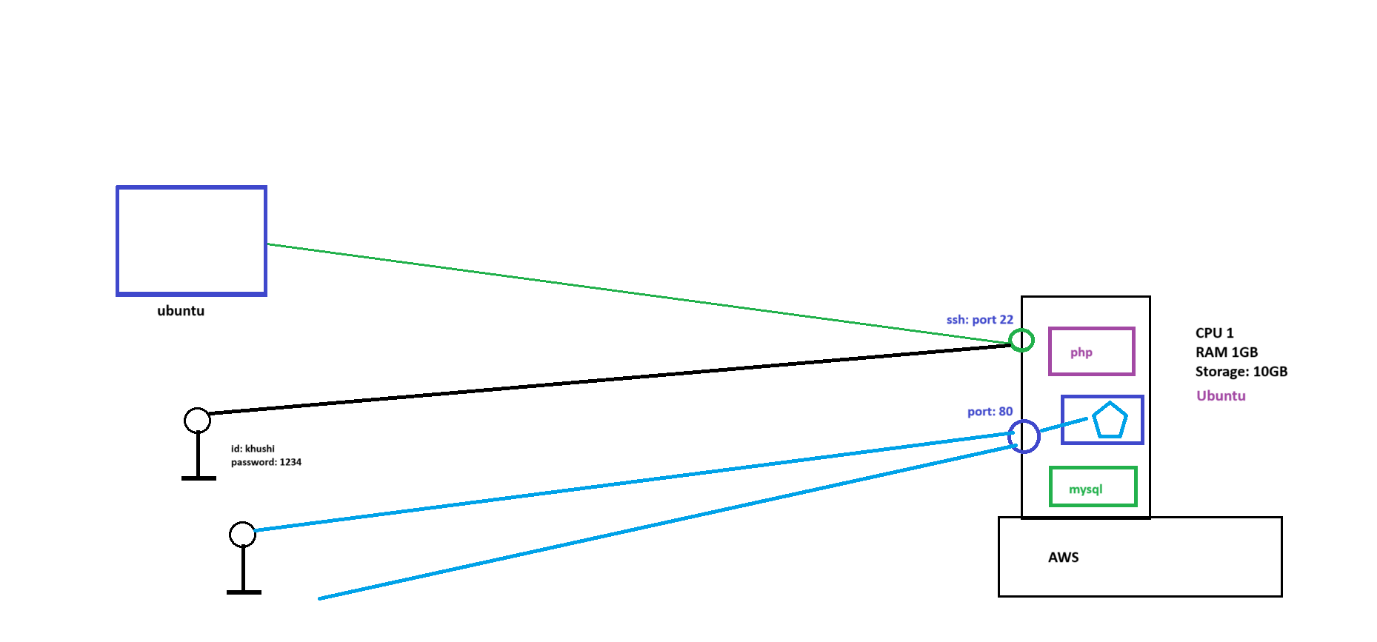
ifconfig

* sudo systemctl status apache2
* sudo apt install apache2



Check on browser : <http://13.201.127.175/>

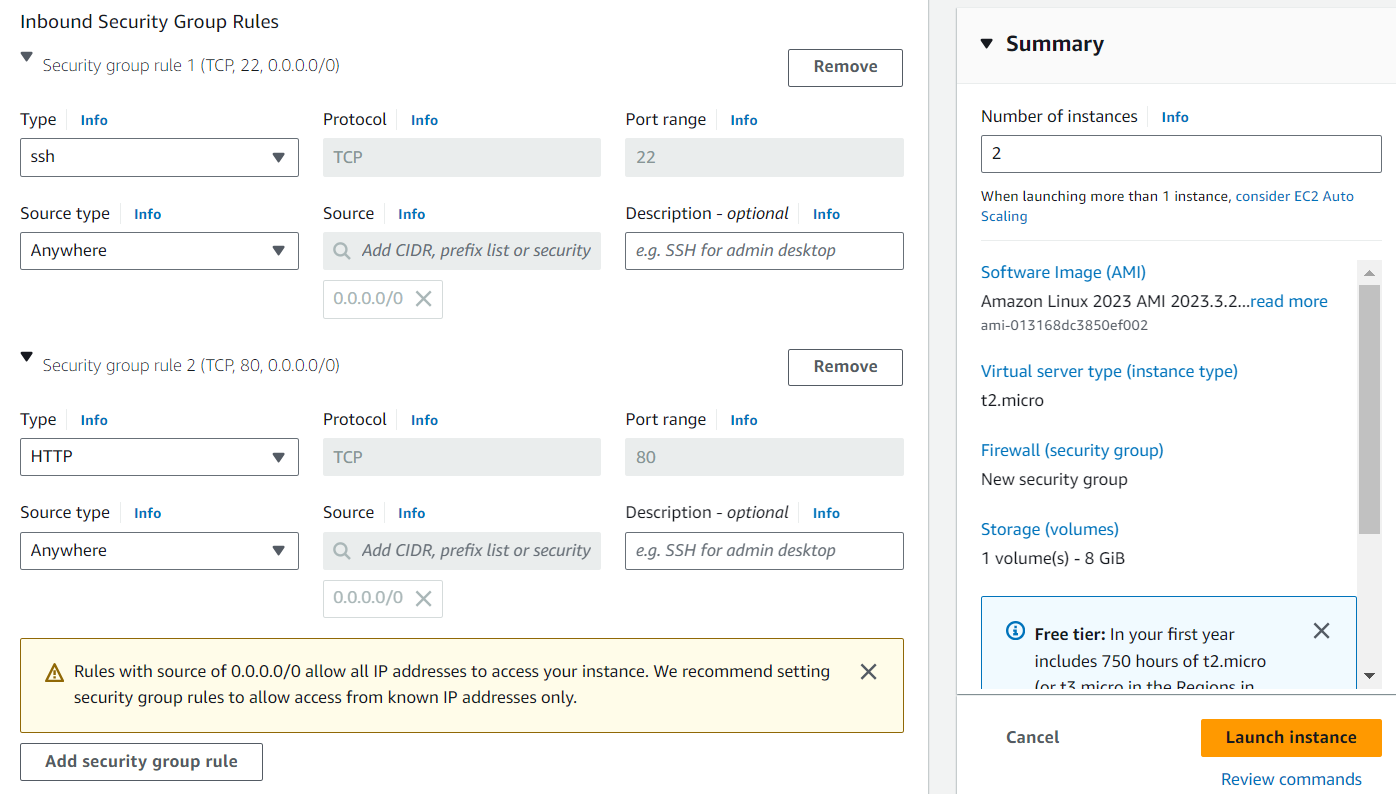
* sudo su
* cd /var/www/html/
* ls
* echo “hello Khushi here. welcome” > index.html
* rm index.html
* apt install unzip
* wget <https://www.free-css.com/assets/files/free-css-templates/download/page294/woody.zip>
* unzip filename.zip
* cp \* -r ../../html/
* sudo apt install php-cli
* apt install php-mysql



-----------------------------------------------------------------------------------

18-03-2024

* Load Balancing:
* Create 2 EC2 instances



* Select the instance -> connect -> SSH -> ssh -i "Ins1key.pem" [ec2-user@ec2-3-111-245-36.ap-south-1.compute.amazonaws.com](mailto:ec2-user@ec2-3-111-245-36.ap-south-1.compute.amazonaws.com)
* Paste it into the gitbash
* C:\Users\Khushi Desai\Documents\PDPU\Sem6\cloud lab\workshop>ssh -i "Ins1key.pem" [ec2-user@ec2-3-111-245-36.ap-south-1.c](mailto:ec2-user@ec2-3-111-245-36.ap-south-1.c)
* Sudo su
* Yum update
* Yum install httpd
* Systemctl status httpd
* Systemctl start httpd
* Cd /var/www/html
* Ls
* Echo “server1” > index.html
* Same steps for second instance!
* Now we have two separate machines
* In AWS Console:
* Load balancing -> Target groups -> Create target group

1. Specify group details:
   1. Basic configuration:
      1. Choose a target type: Instances
      2. Target group name: TG1
      3. Protocol: Port => HTTP, 80
      4. IP Address type: IPv4
      5. VPC
   2. Health checks
      1. Advance health check settings:

* Health check port: Traffic port
* Healthy threshold: 2
* Unhealthy threshold: 2
* Timeout:2
* Interval: 5
* Success codes: 200

1. Register targets:

Select both the targets -> include as pending below -> create

* Load balancing -> load balancers -> create load balancer -> Application load balancer

1. Basic configuration:
   1. Load balancer name: myLB1
   2. Scheme: Internet-facing
   3. IP address type : IPv4
2. Network mapping:
   1. VPC
   2. Mappings: 1a & 1b
3. Security group:

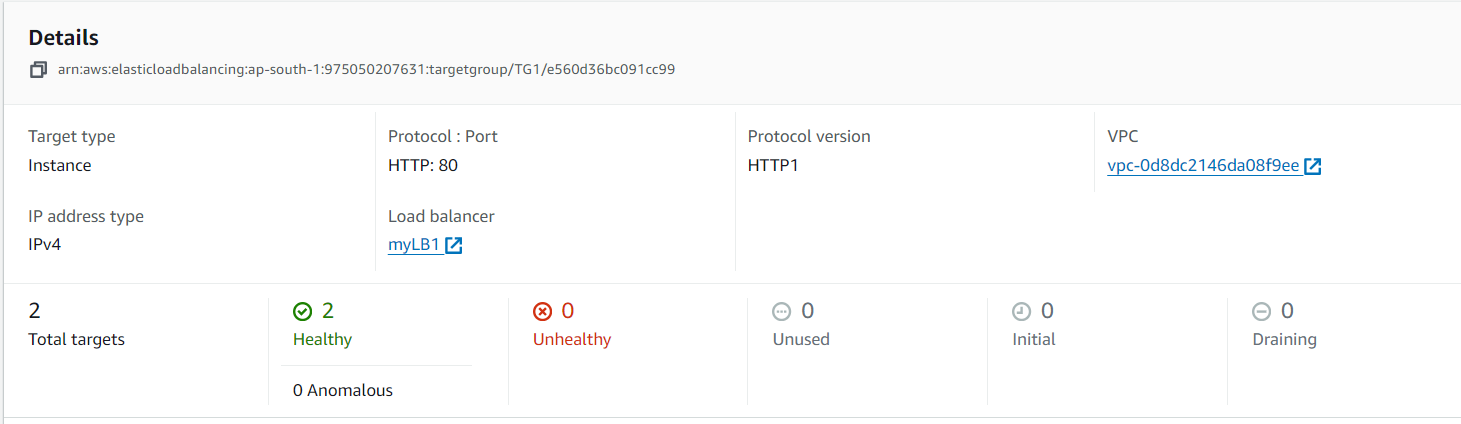
Select the one in which port no 80 is on

1. Listeners and routing:

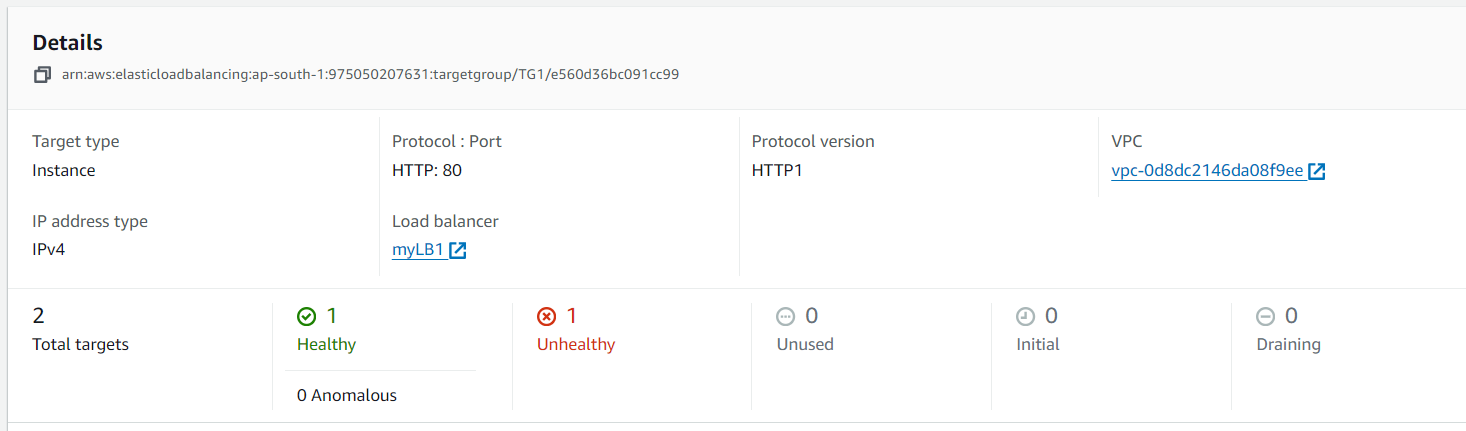
Select the target group

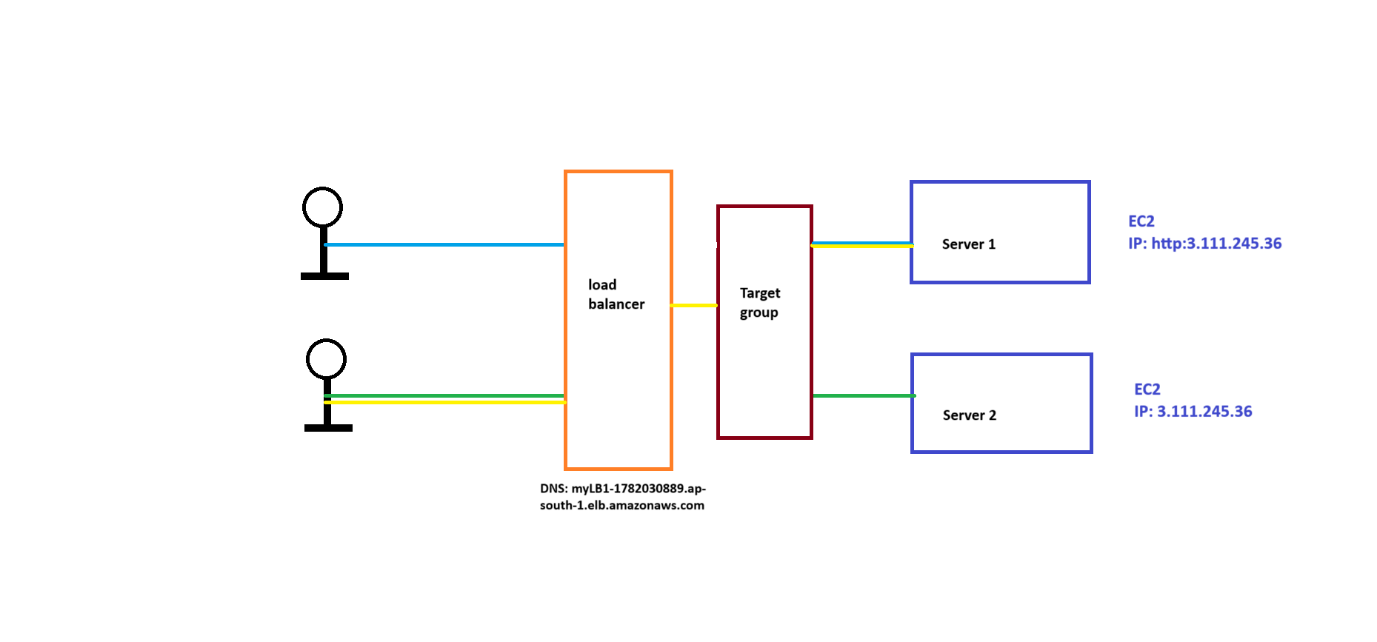
1. For all other options make them default
2. Create

* Check in EC2-> Load balancing -> target groups:



* Systemctl stop http

****

****

**--------------------------------------------------------------------------------------------------------------------------**

* File sharing: SFTP

Third party file sharing : WinSCP

* Install WinSCP
* Select SFTP
* Add ip address in username
* For password:

Advanced setting -> authentication -> browse the key

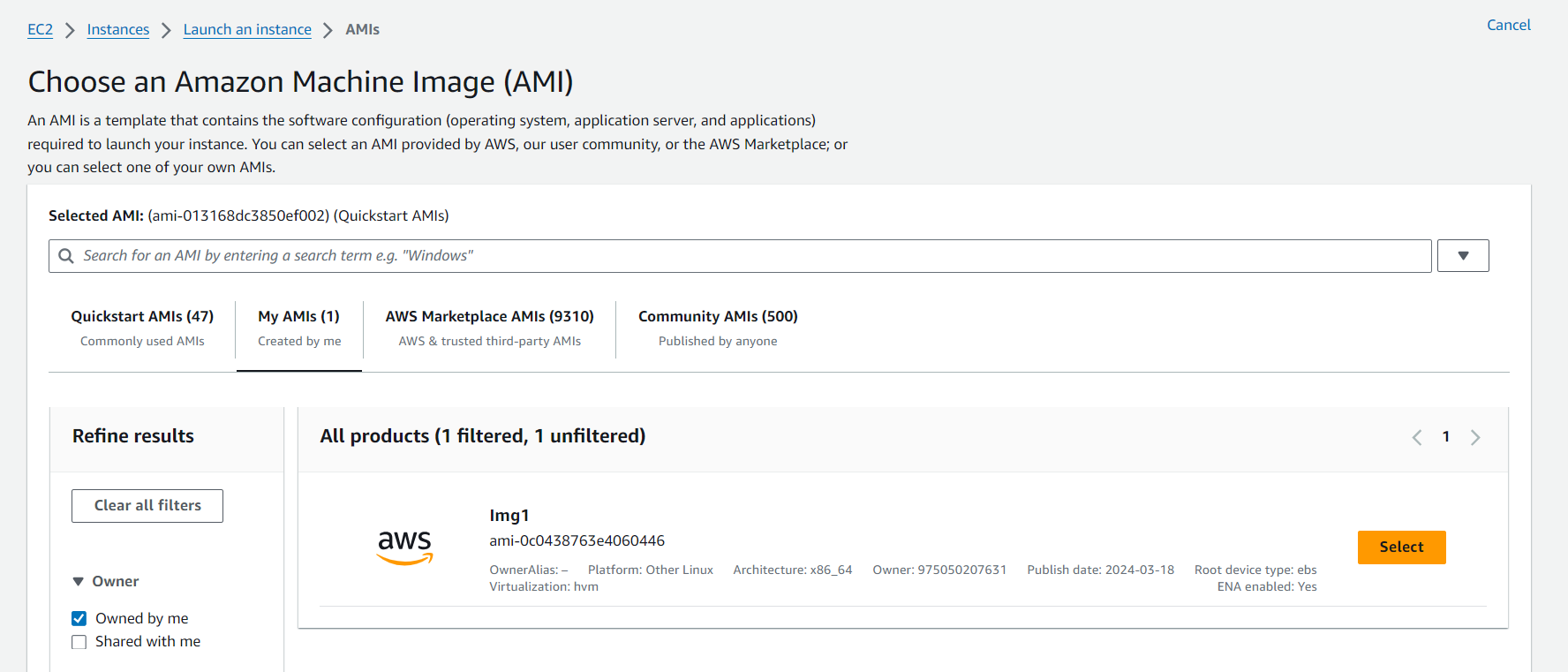
Add or move your file

Mv : used for rename & move file

* In gitbash
* Login
* Ls
* Mv foldername filename
* Pwd
* Cp filename /var/www/html
* Cd /var/www/html
* Ls
* Mv filename index.html

--------------------------------------------------------------------------------------------------------------------------

* Auto-scaling:
* Scale up-down
* Scale in-scale out
* Select the instance -> actions -> image and template -> create image
* Give the image name, description -> create image
* Check the AMIs in the left pannel
* Create a new instance using created AMI

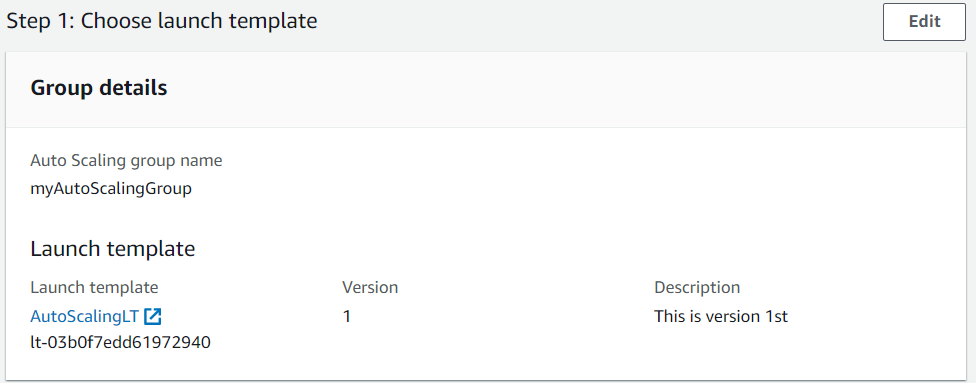


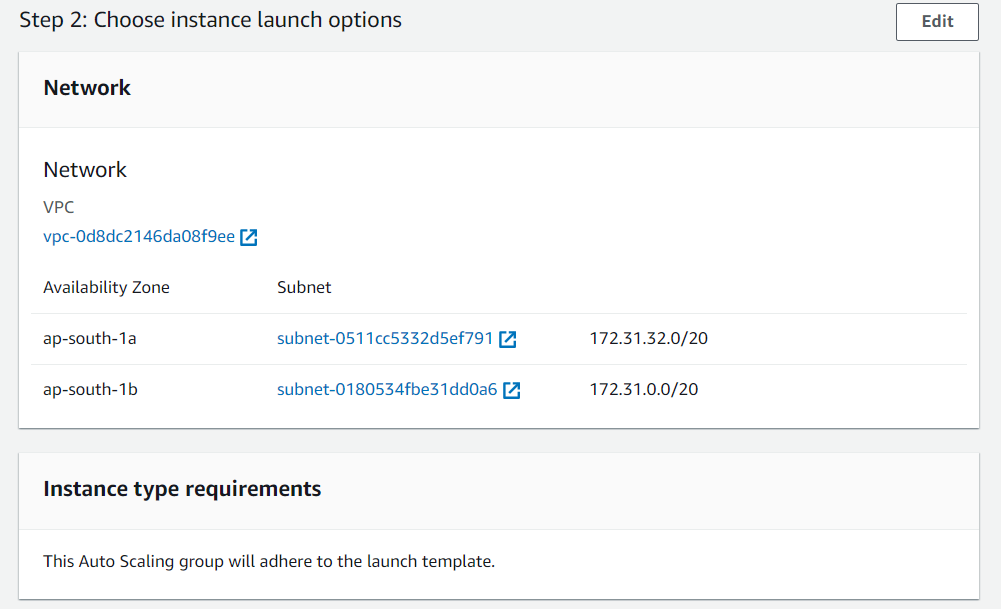
Select security group in which port 80 is running

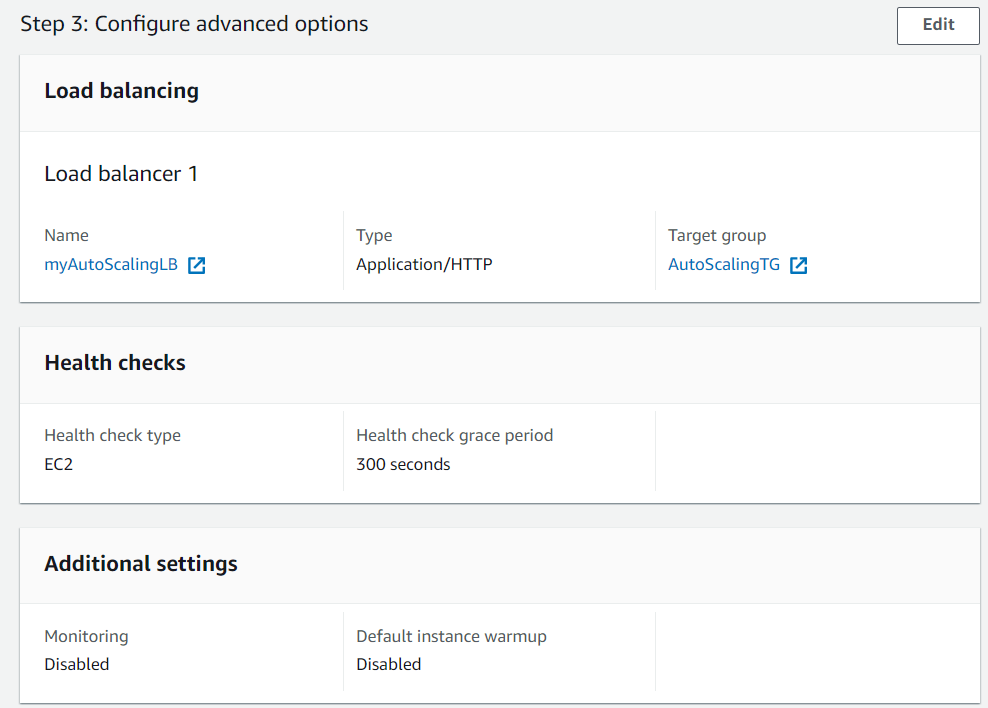
* Select that instance and connect -> EC2 instance connect -> connect -> systemctl start httpd
* Check in the new browser with your new instance’s ip

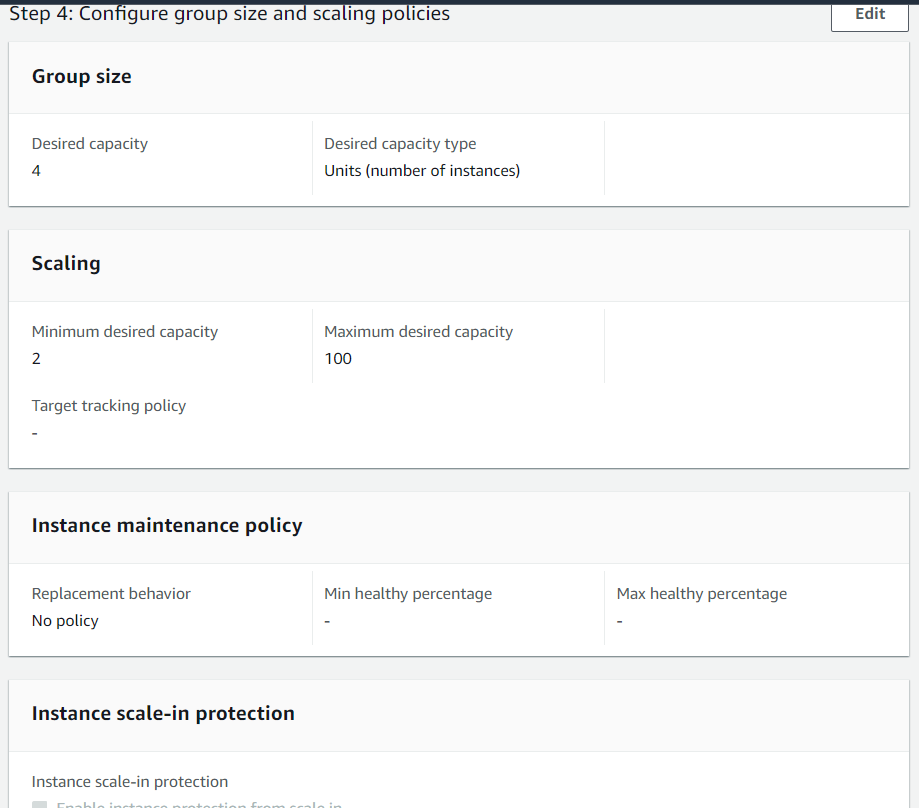
----------

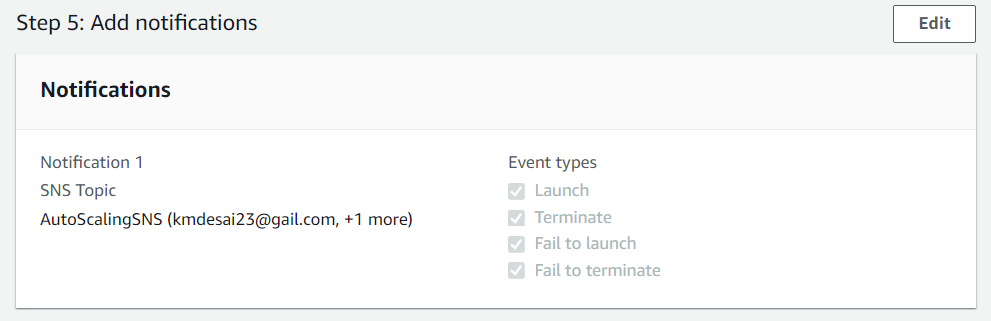
* Create one EC2 instance
* Create web server with a website (woody)
* Create one image
* Create a target group
* Create a load balancer
* Create a template
* SNS service
  1. Name
  2. Create topic:
     + Type: Standard
     + Create topic
  3. Create subscription
     + Protocol: email
     + Add your email
     + Confirm your email by opening your gmail
* Create autoscaling group

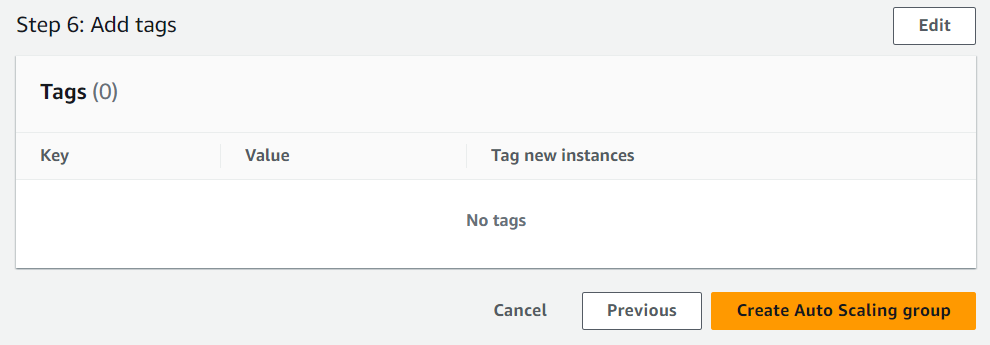




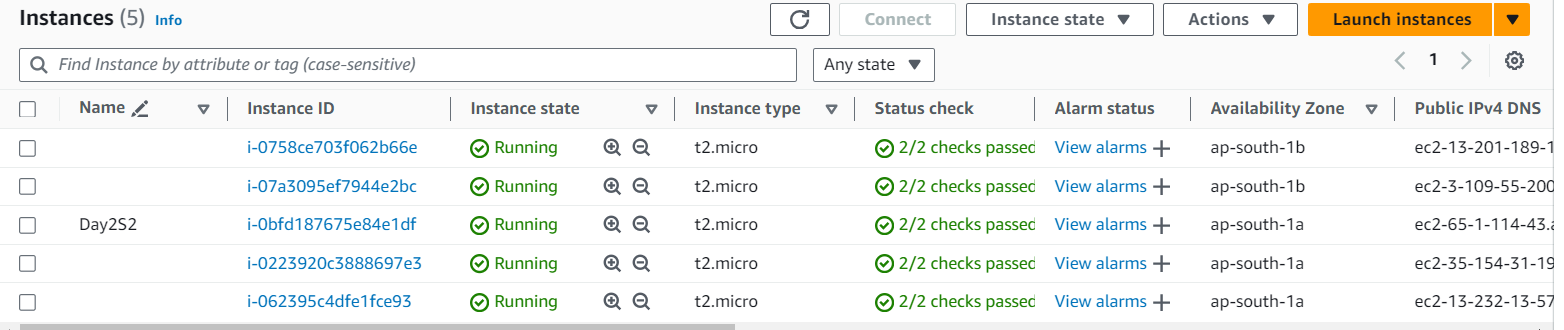




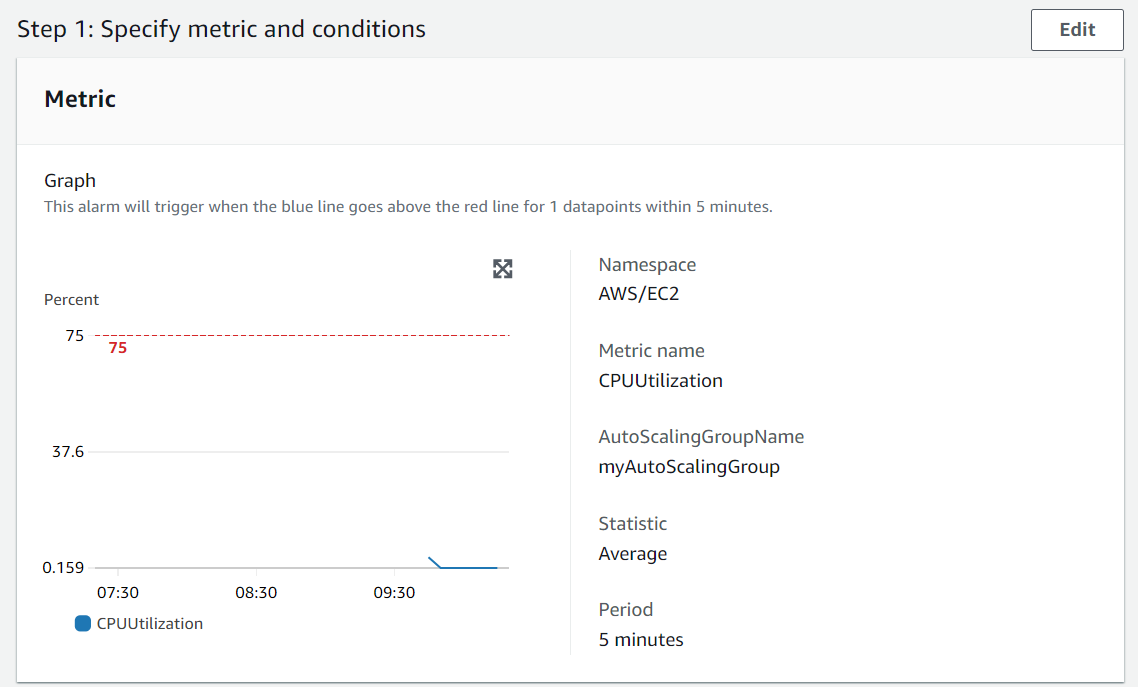


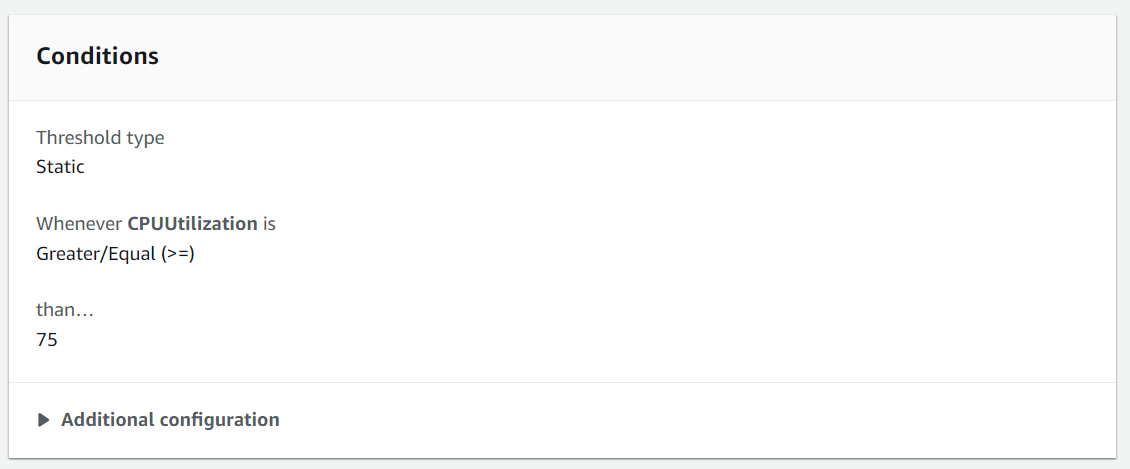


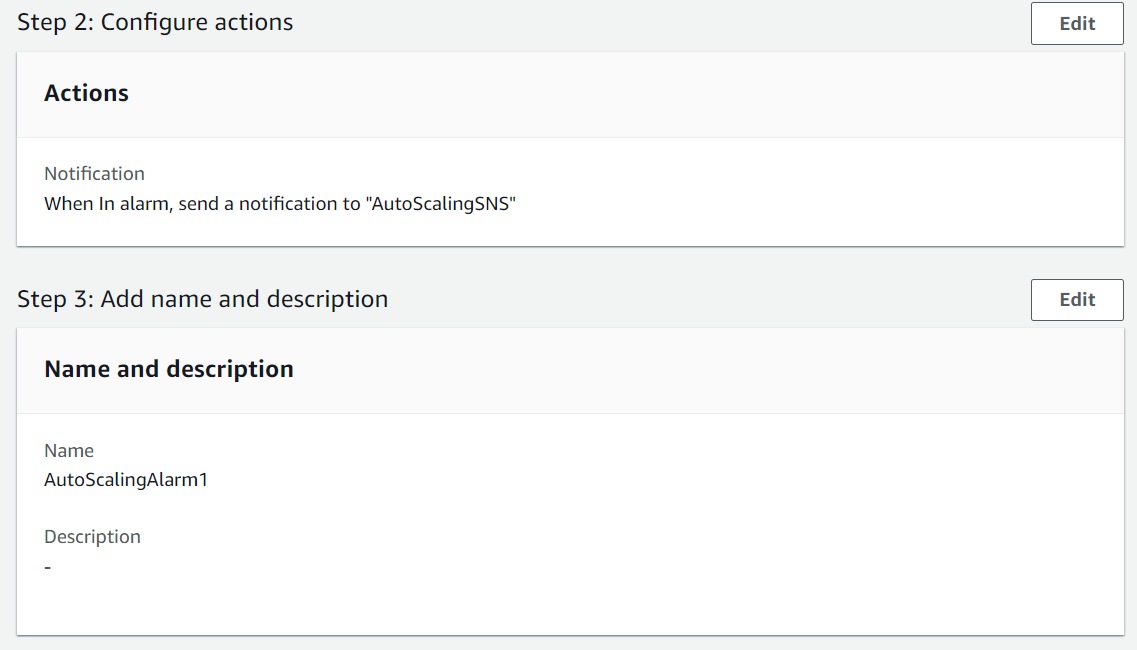
* Now check your instances



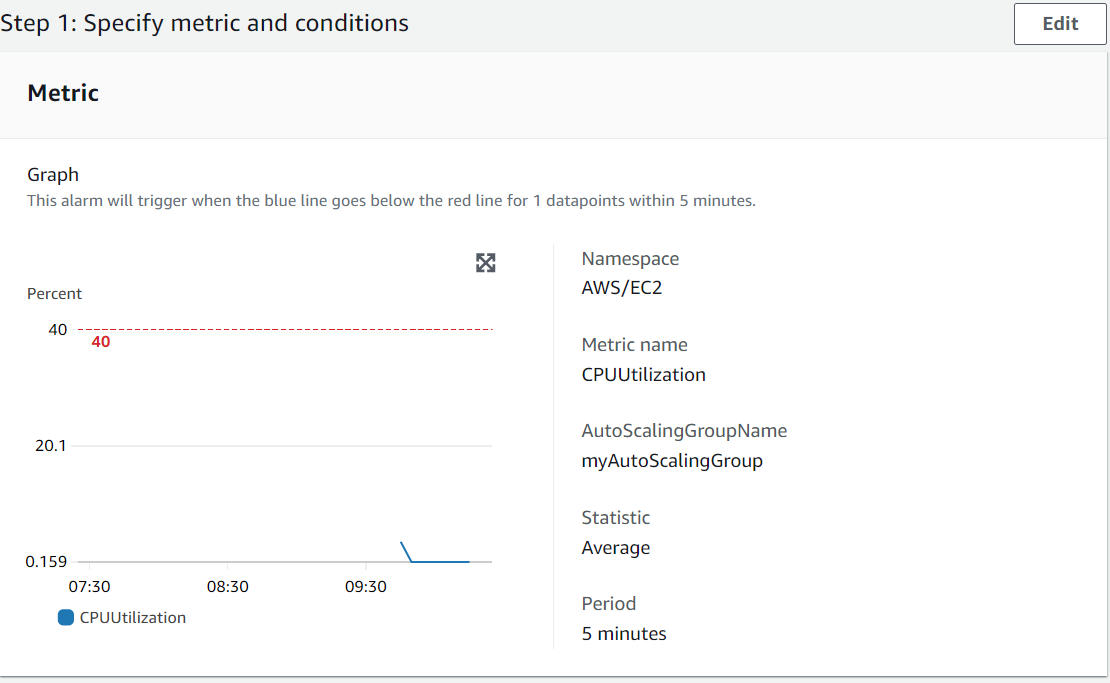
* Type below commands in gitbash
* htop
* top
* Cloud watch monitoring service:

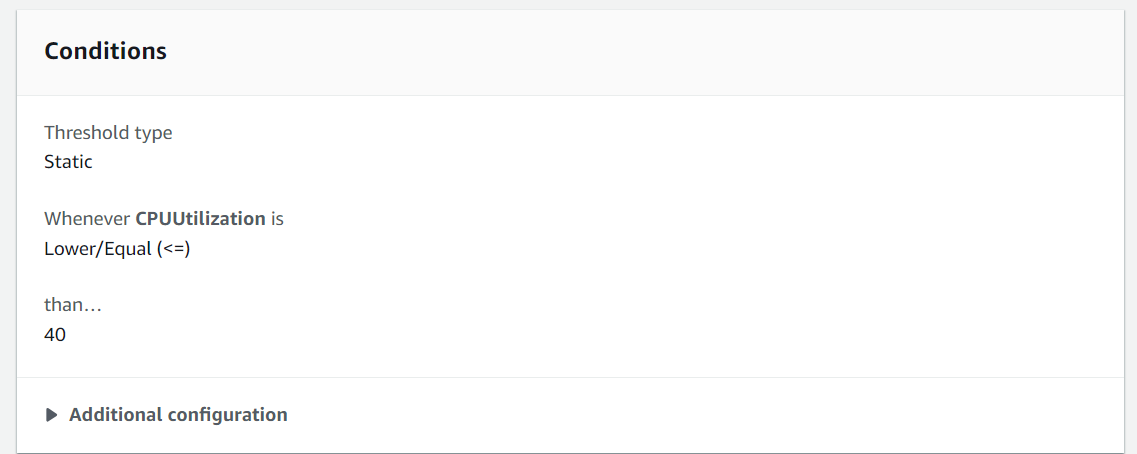


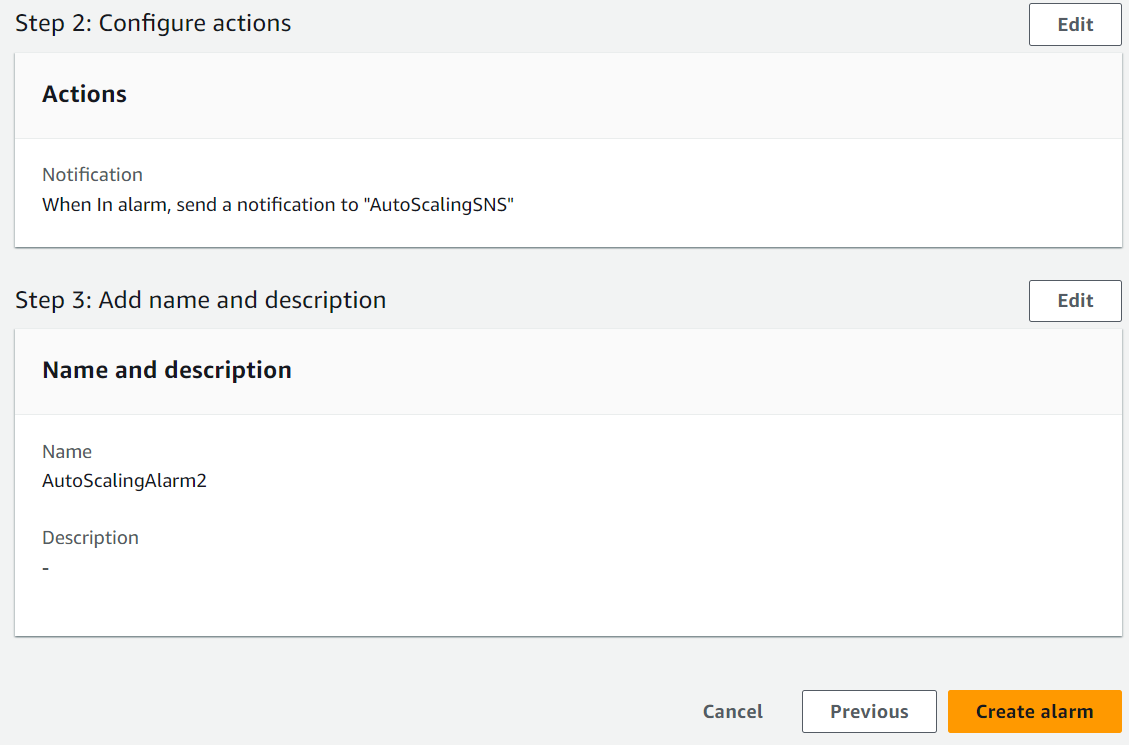




* Create alarm2



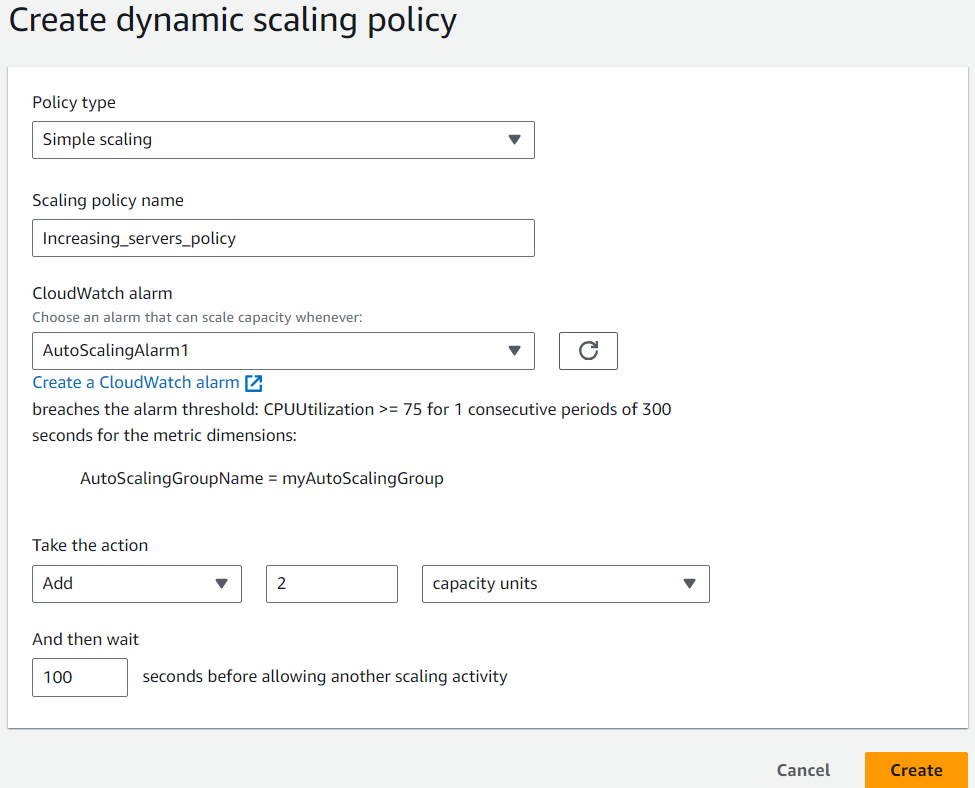




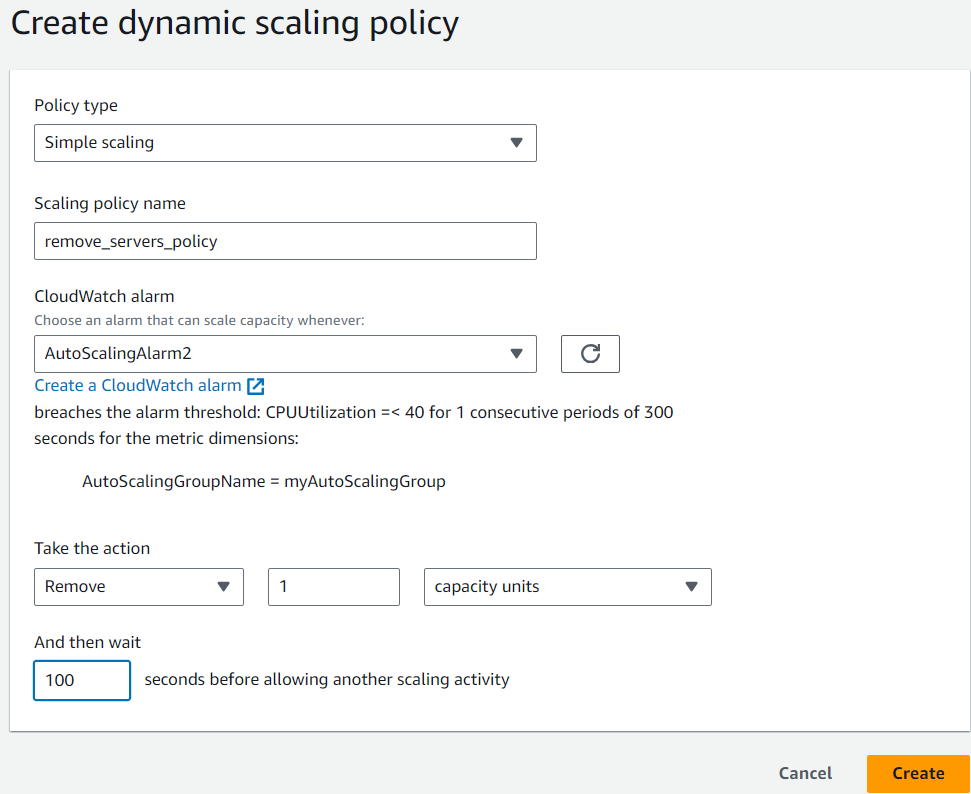
* EC2 -> Autoscaling groups -> select the group

Automatic scaling -> Create dynamic scaling policy

Create One dynamic policy for alarm1 (overloaded)



Create One dynamic policy for alarm2 (underloaded)



Check email notifications

Delete any instance, it will automatically create one if it is less than min instance value.(here 2)

* Deleting process:

Autoscaling group

Load balancer

Target group

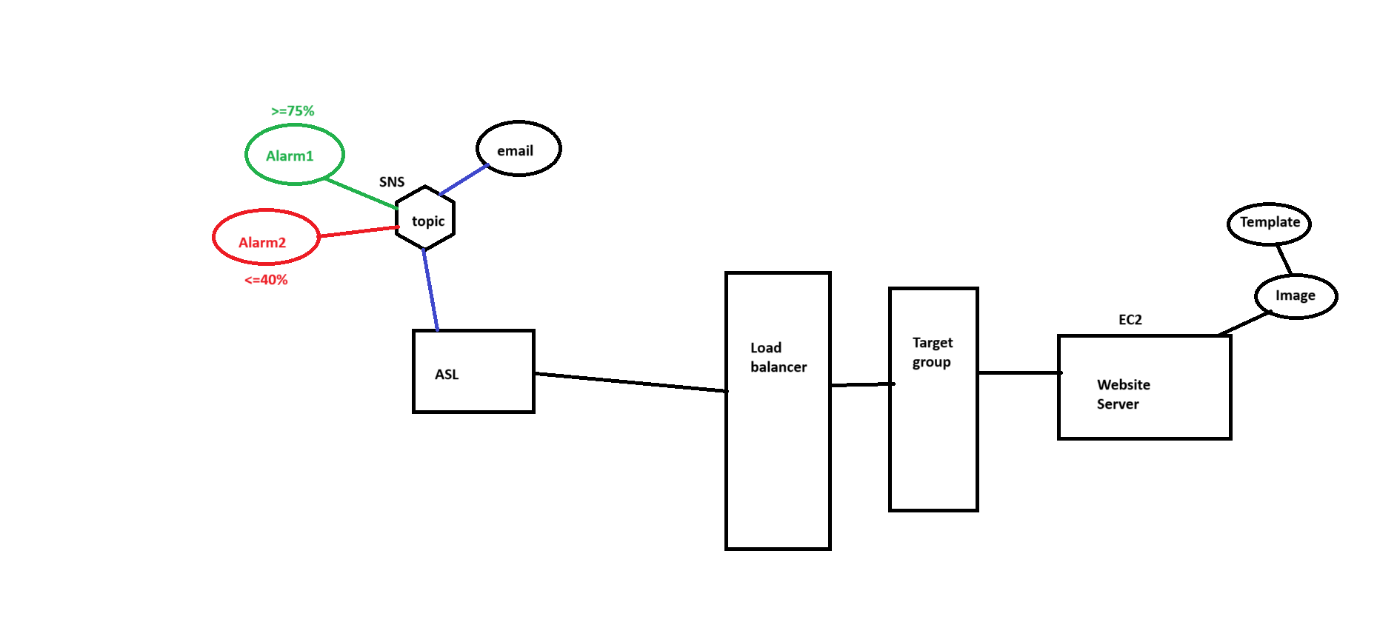
Instances

Sns

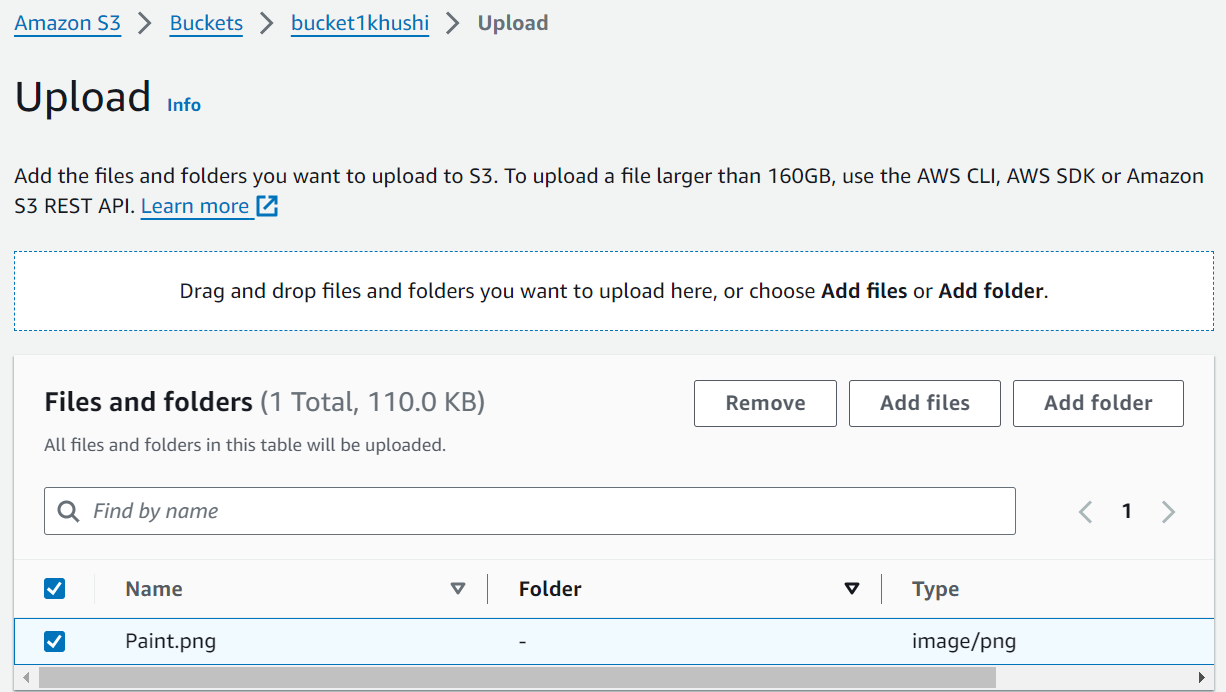
Cloud watch

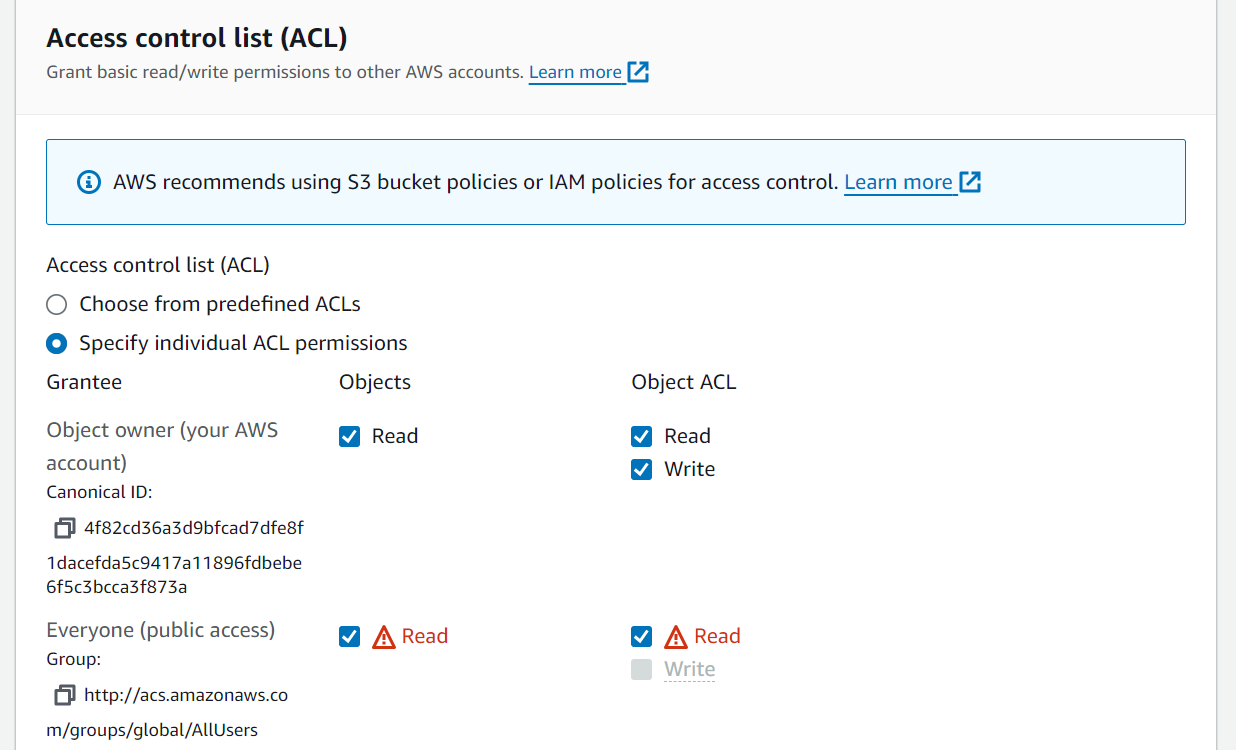
Template

Image



* Cloud trial
* S3:
* Create bucket ->
  1. General configuration: give bucket name
  2. Object Ownership: ACLs enabled
  3. Block Public Access settings for this bucket: uncheck the boxes
  4. Create bucket
* Object upload:
* Click on the bucket -> upload -> add files





- Create object

* Click on the created object

Copy the object url

* Upload website folder

Properties -> static website ->enable

Index document: index.html

Error document: error.html

Save

Now you can check the website url

- Paste it into the browser and add /foldername

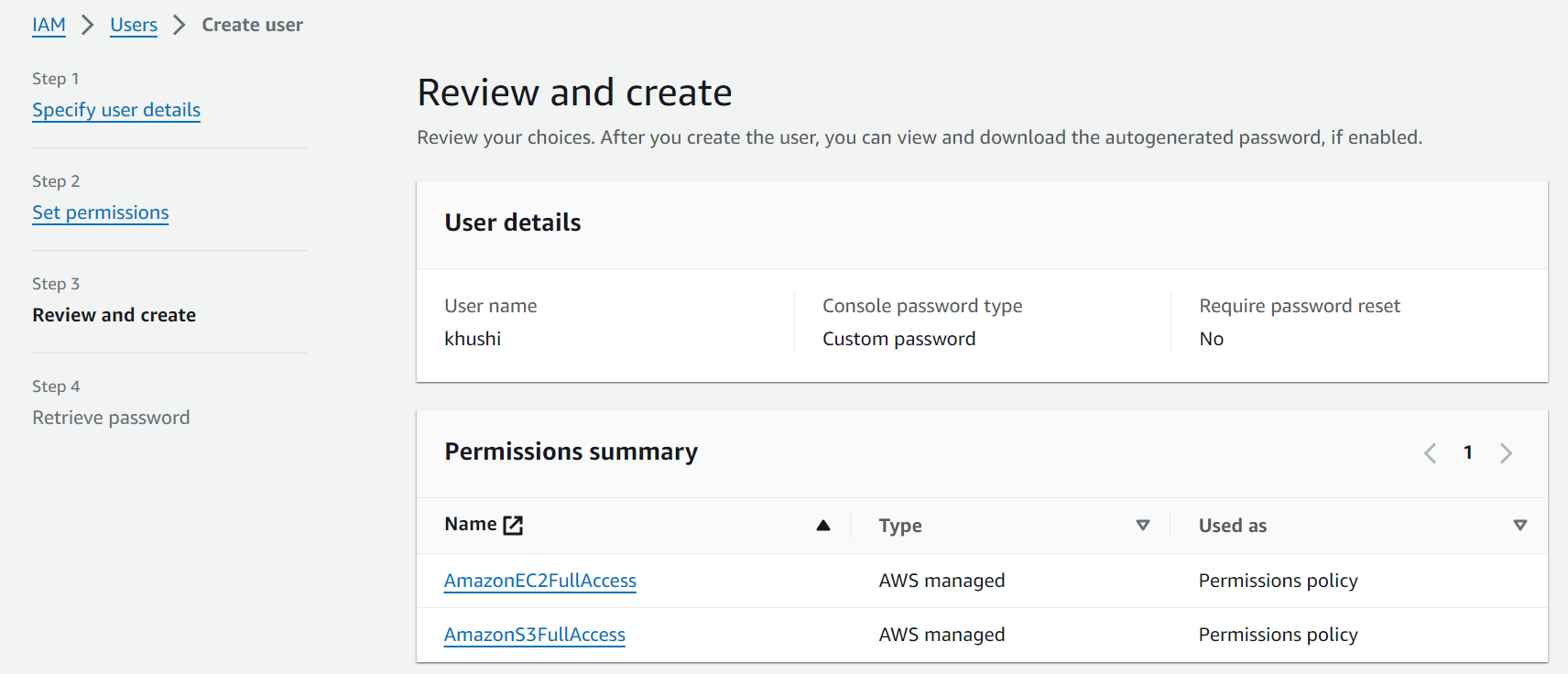
<http://bucket1khushi.s3-website.ap-south-1.amazonaws.com/hostit-html>

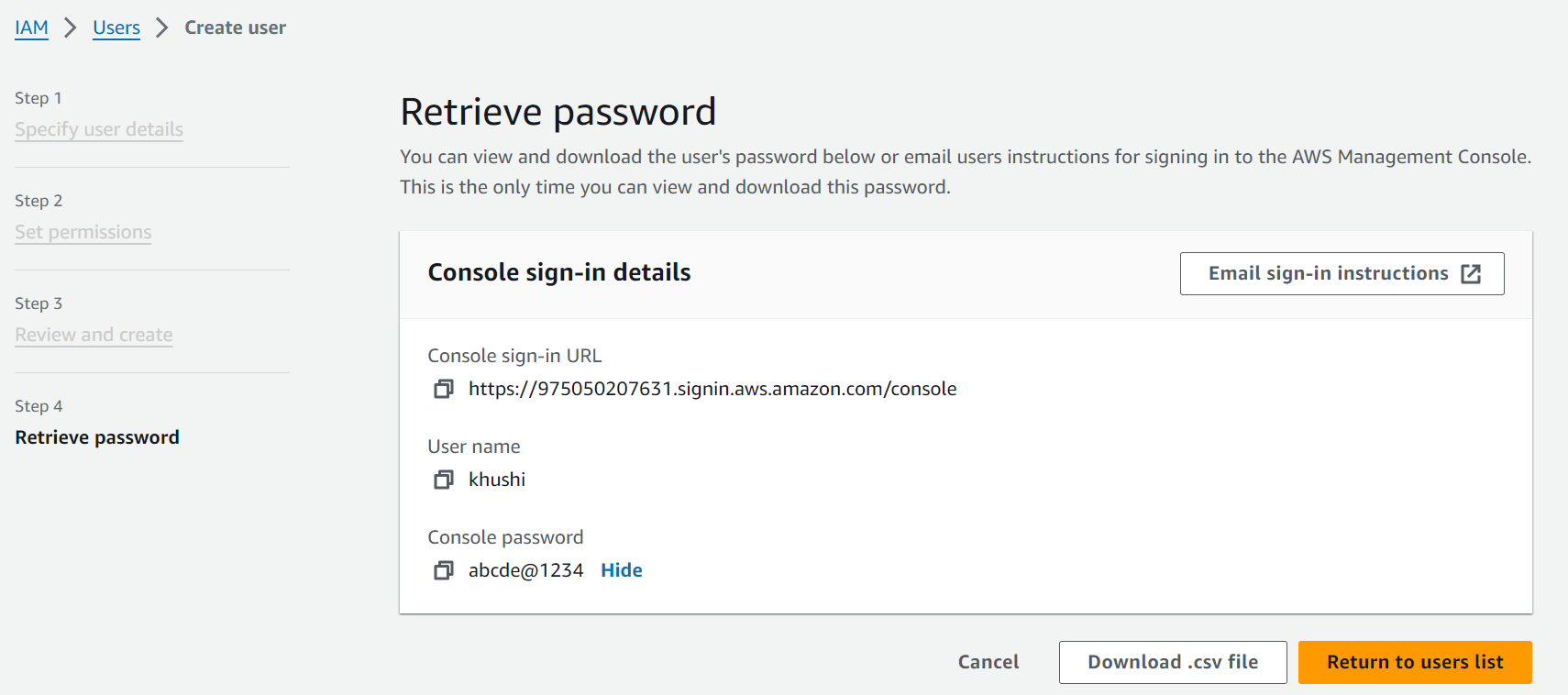
------------------------------------------------------------------------------------

19-03-2024

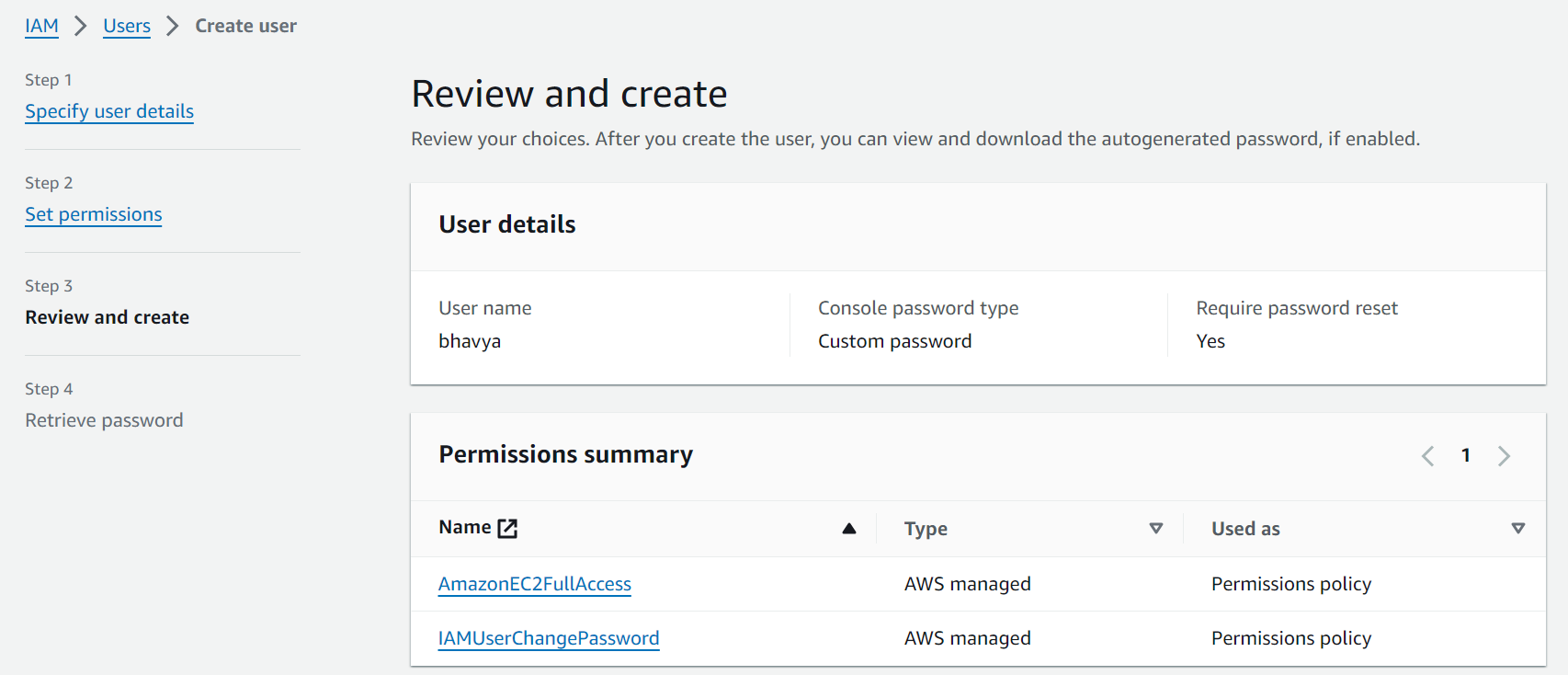
* Create a S3 bucket, enable versioning option during creation.
* Create one object with text file
* By enabling show versions option, you can check versions
* Actions-> edit action -> edit storage class -> standard storage
* Select the object -> delete
* Enable show version option -> you can check that delete marker is created
* If you delete the marker file, then you check that your file is again restored.
* IAM

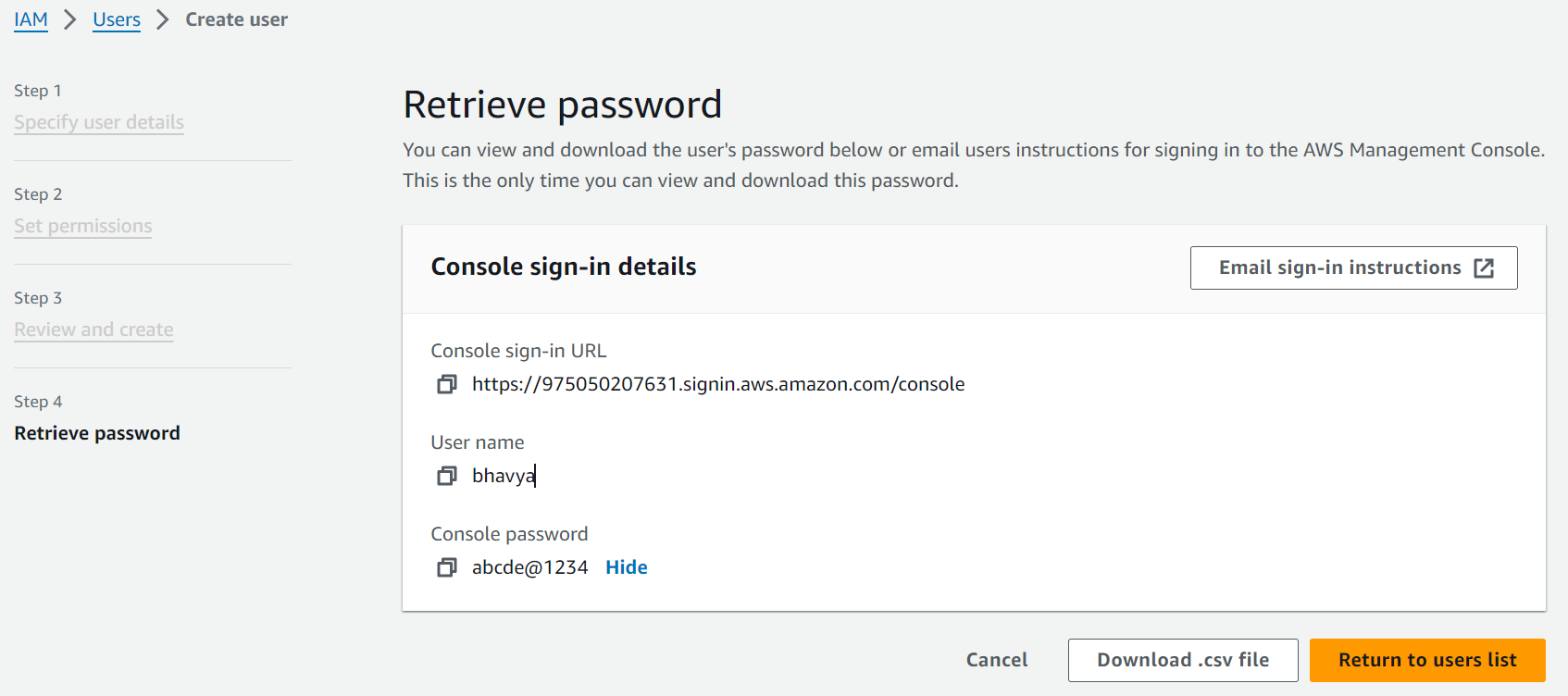
Create 1st user with EC2 & S3 permission:



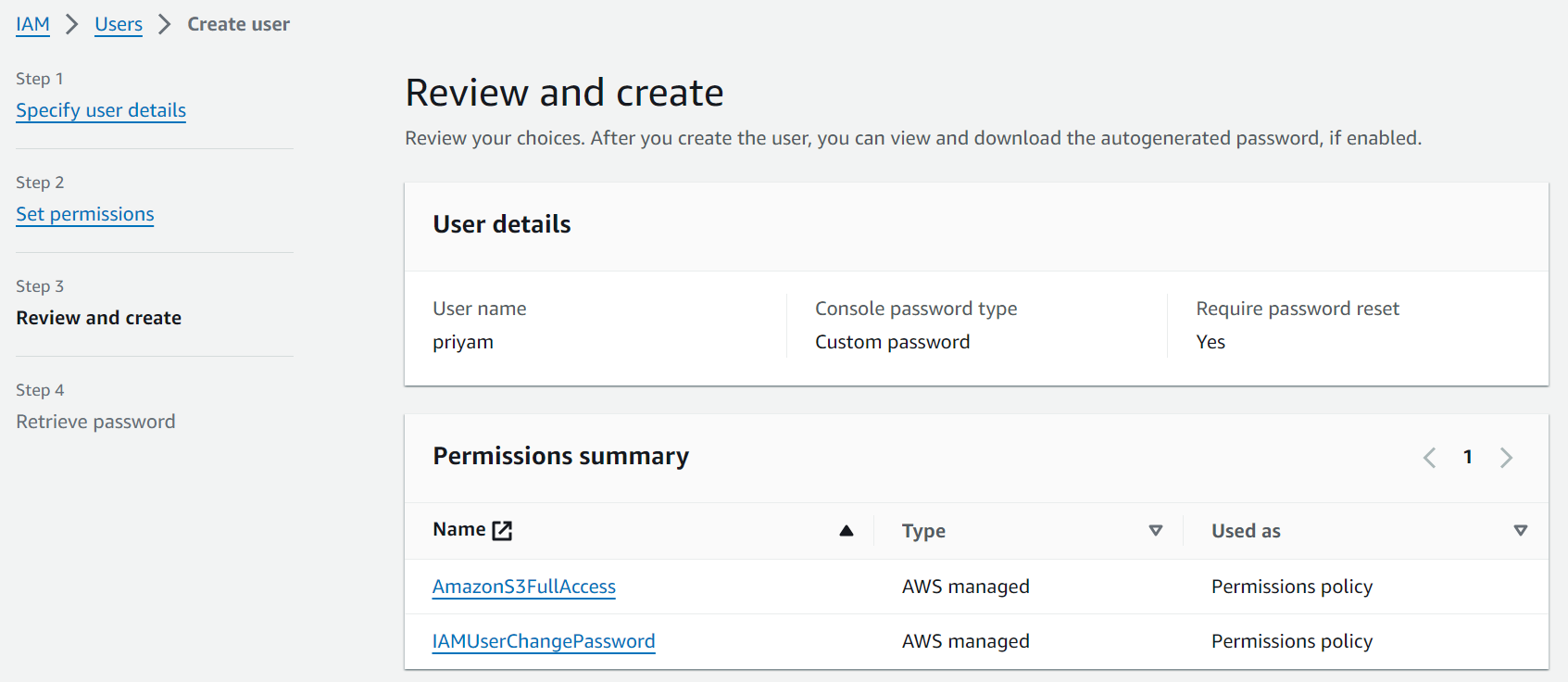


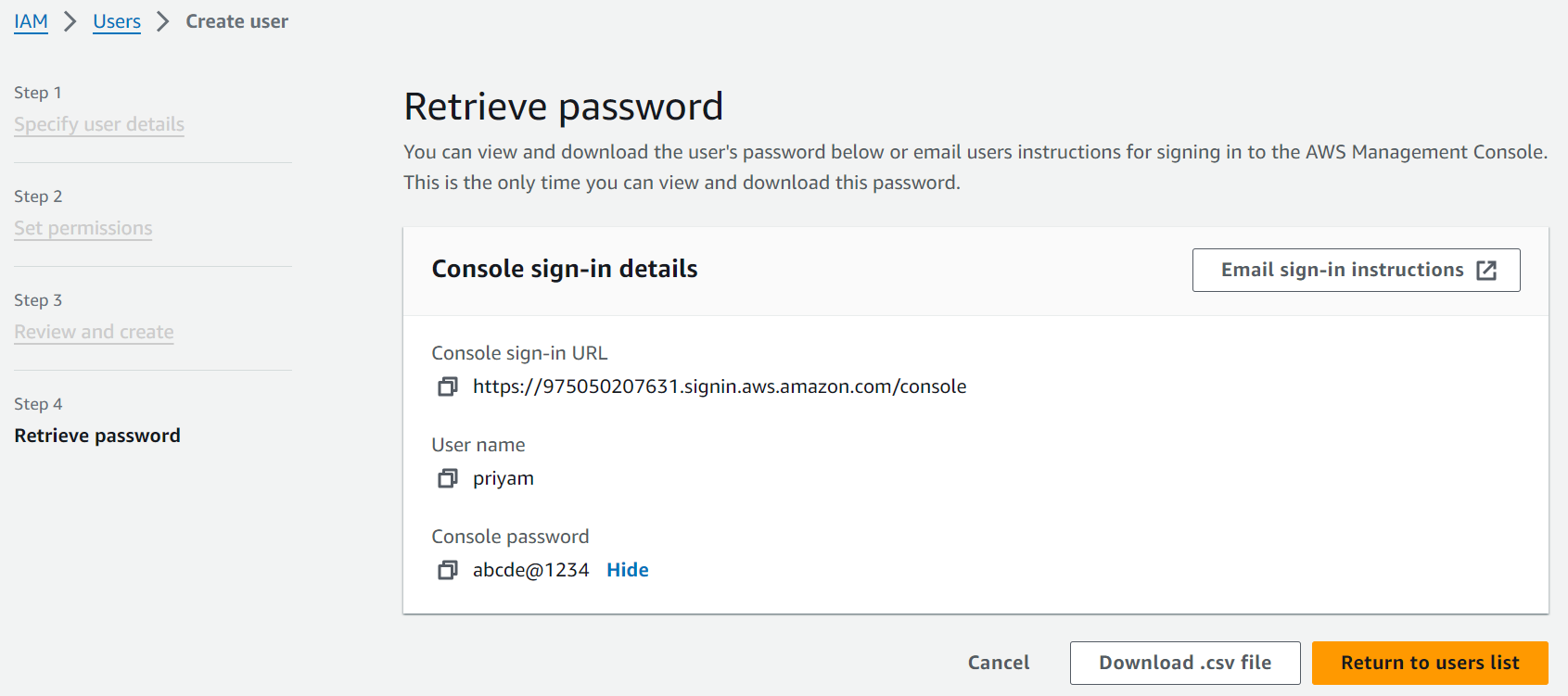
Create 2nd user with EC2 permission:



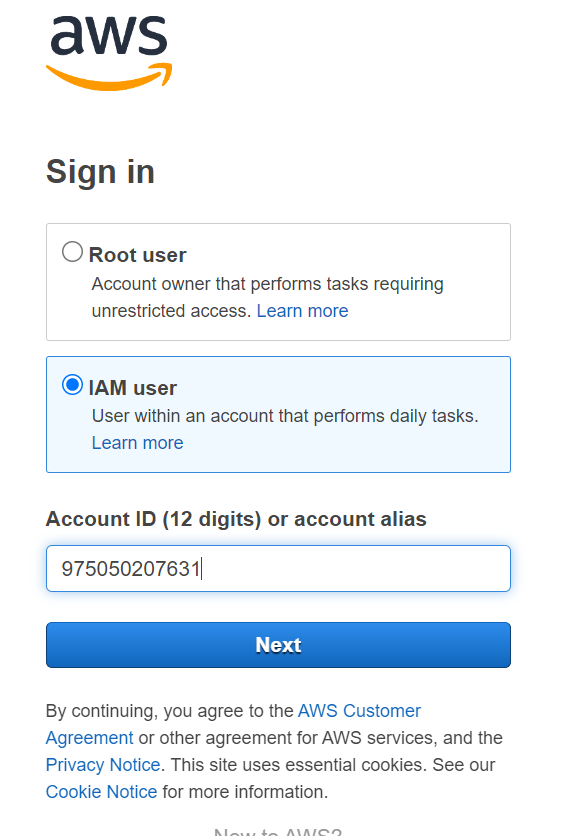
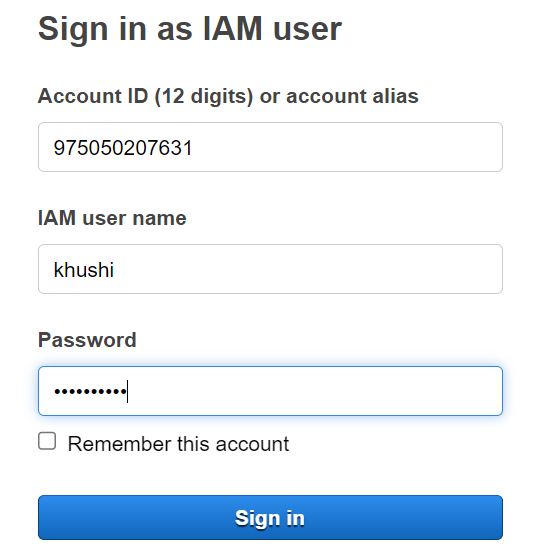


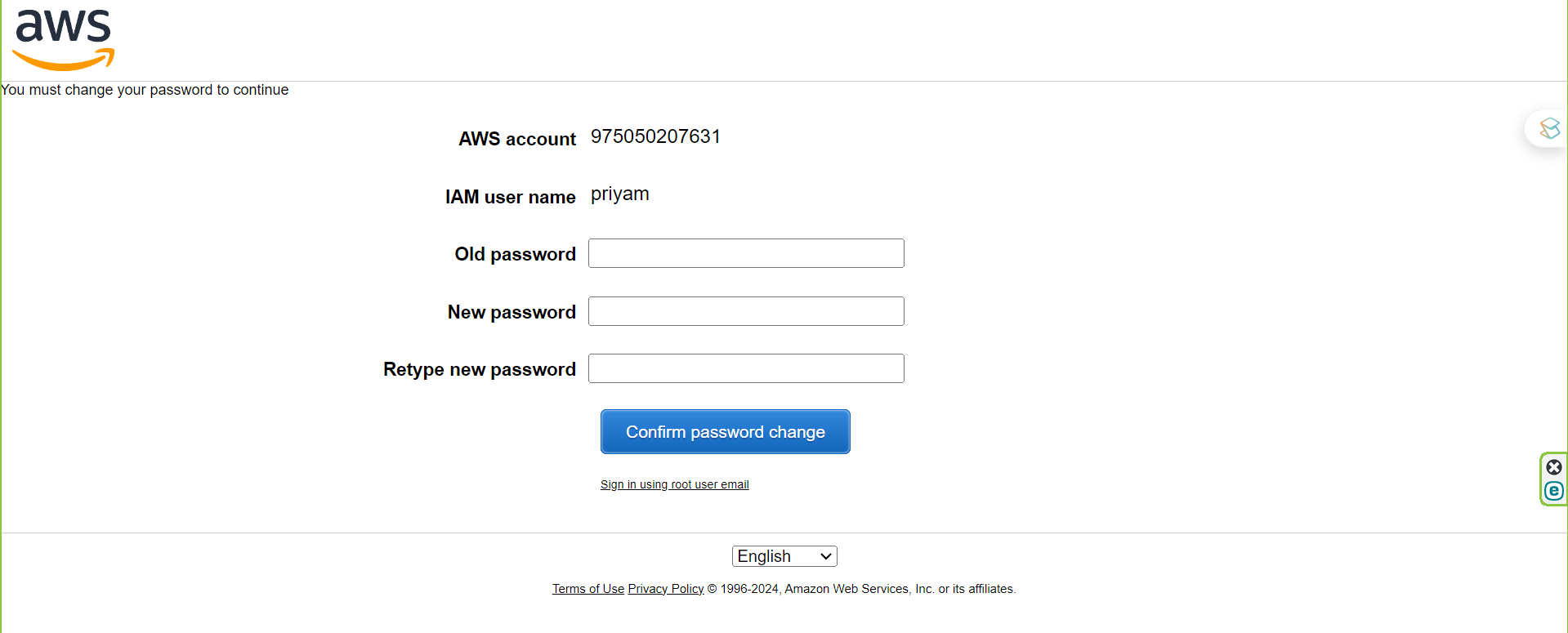
Create 3rd user with S3 permission:





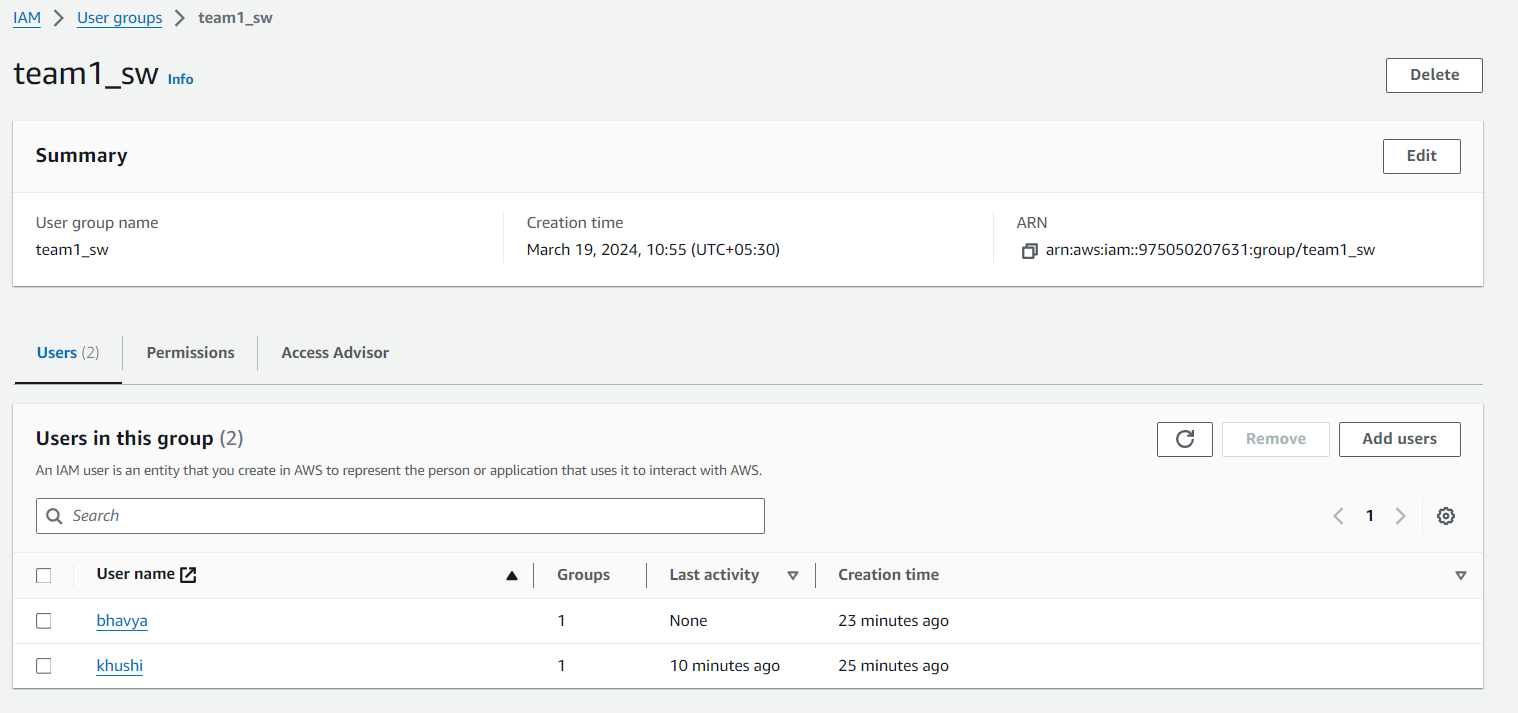
* Open a new browser
* Open aws

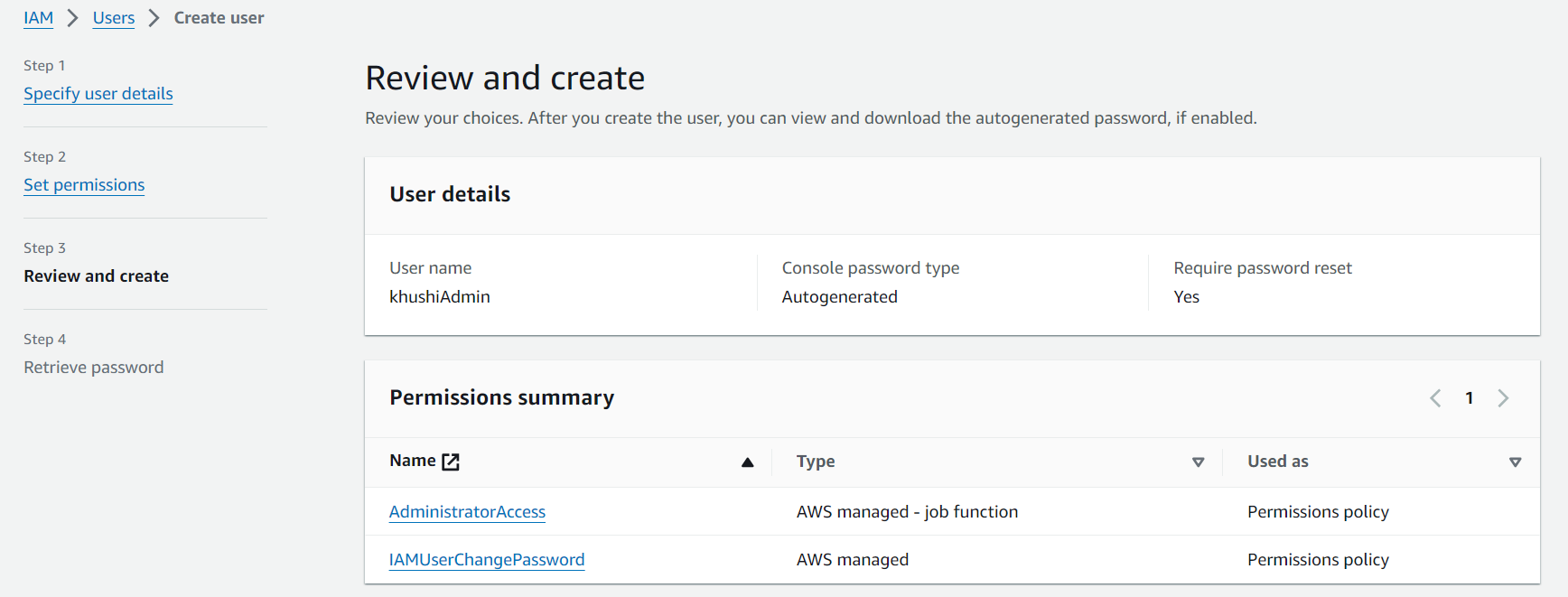


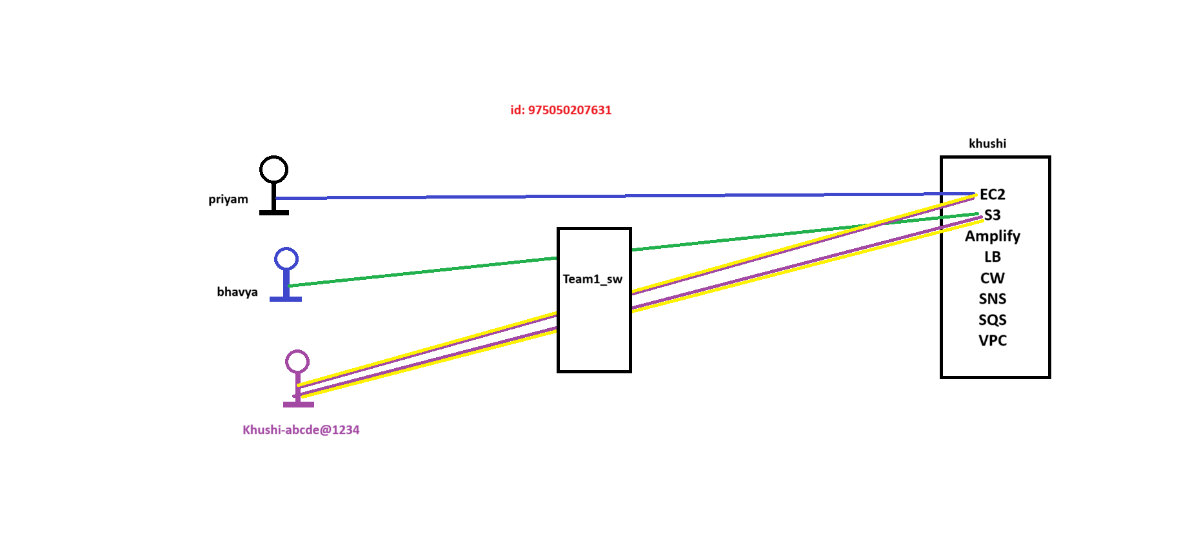
* Create user groups

Team1\_Sw: give SNS & S3 Permission



* Create another user:





* Install AWS CLI

--------------------------------------------------------------------------------------------

* Terraform
* Install terraform for windows
* Extract all
* Open in VS Code
* Create one **main.tf** file

provider "aws" {

  region     = "ap-south-1"

  access\_key = "AKIA6GBMFOGHT4J7WROH"

  secret\_key = "BFxLnwyxIlFkMa+zjYi57dF6c3etAPuMb4+dMmDV"

}

resource "aws\_instance" "EC2server" {

  ami           = "ami-007020fd9c84e18c7"

  instance\_type = "t2.micro"

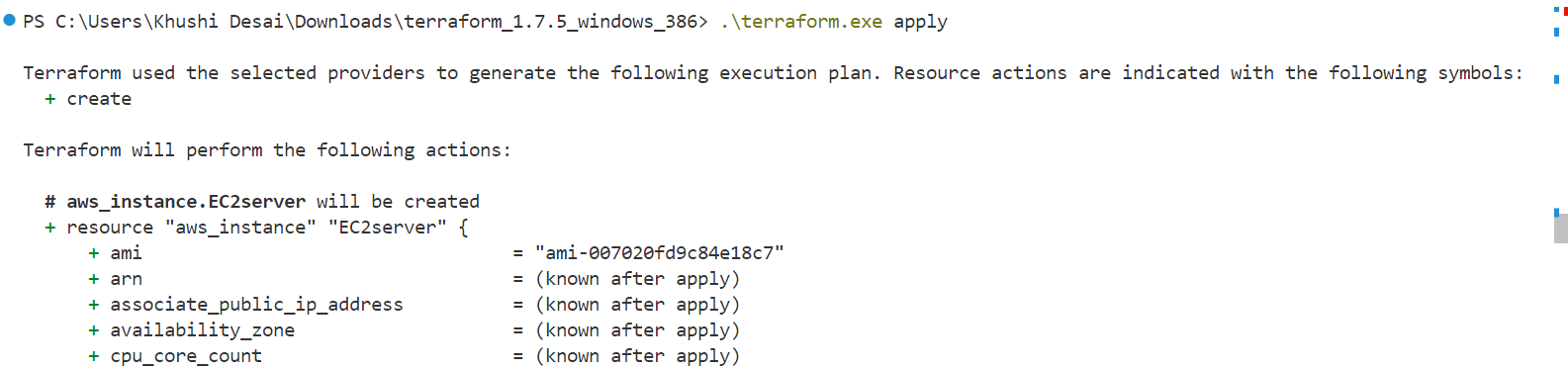
  tags = {

    Name = "khushiCloudServer"

  }

}

* Terminal commands:
  1. .\terraform.exe fmt 🡪 Format document
  2. .\terraform.exe plan 🡪 create plans
  3. .\terraform.exe init 🡪 finds the plugin in AWS
  4. .\terraform.exe plan



* 1. .\terraform.exe apply 🡪 resources are added

Check your aws account, one instance has been created!!

* 1. .\terraform.exe destroy

Only those which are created by me will only be deleted.

* + - * create 5 instances:

provider "aws" {

  region     = "ap-south-1"

  access\_key = "AKIA6GBMFOGHT4J7WROH"

  secret\_key = "BFxLnwyxIlFkMa+zjYi57dF6c3etAPuMb4+dMmDV"

}

resource "aws\_instance" "EC2server" {

  ami           = "ami-007020fd9c84e18c7"

  instance\_type = "t2.micro"

  count         = 5

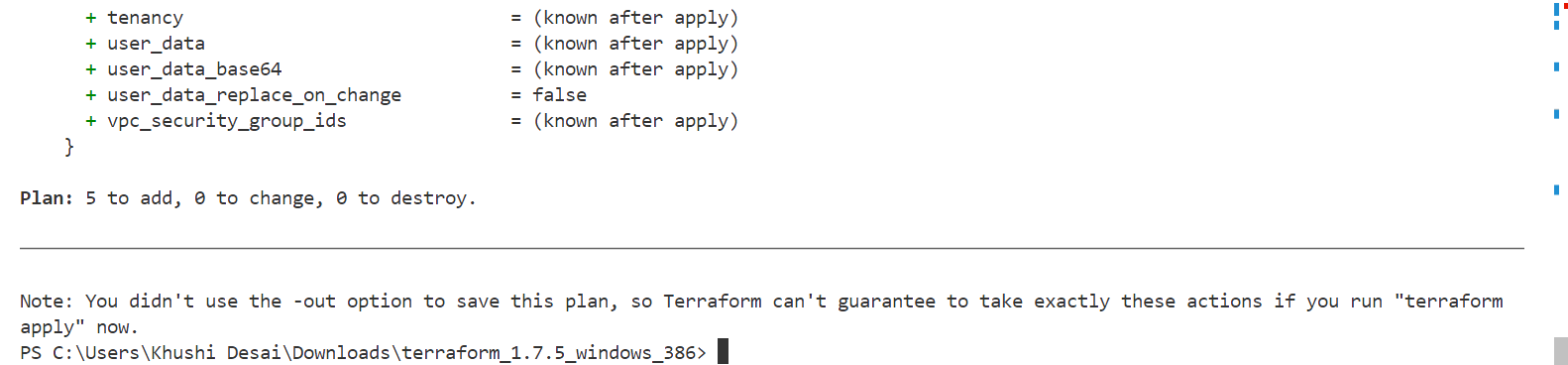
  tags = {

    Name = "khushiCloudServer"

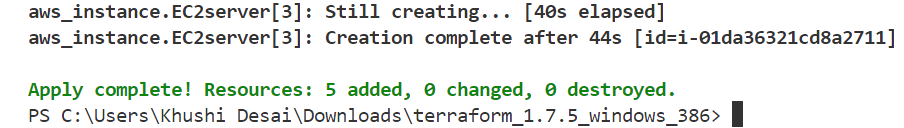
  }

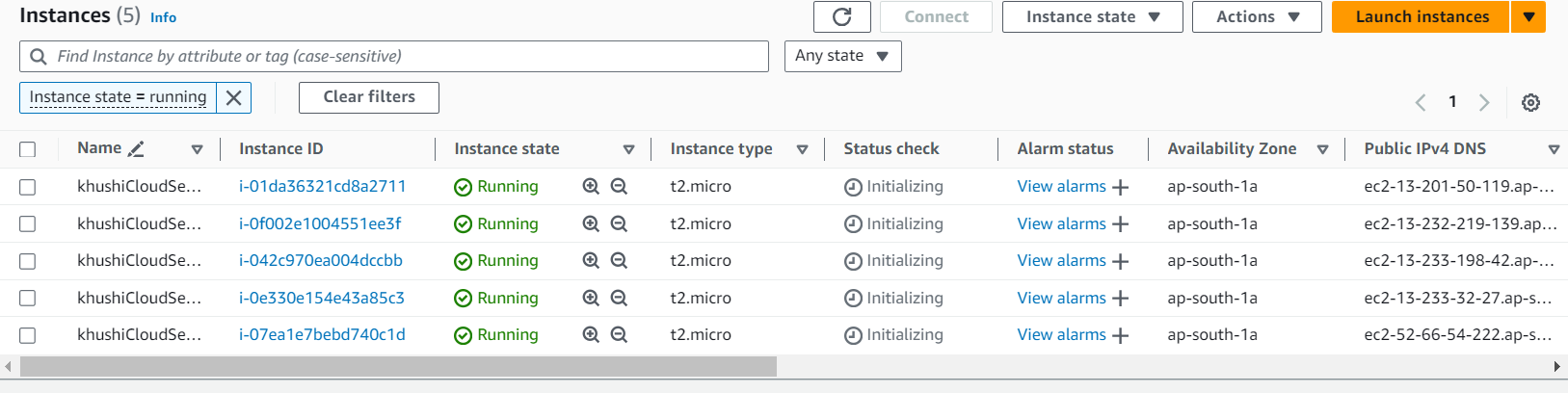
}

1. .\terraform.exe plan



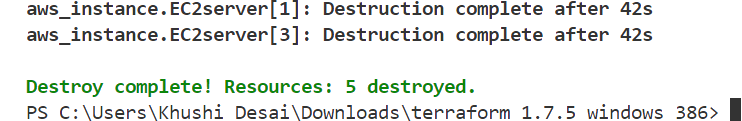
1. .\terraform.exe apply





* **terraform.tfstate** file contains all the summary and other information of the infrastructure.

1. .\terraform.exe destroy



* Mysql
* Open xampp and start services
* Command prompt: login to mysql server:

mysql -u root -h localhost -p

* Create database:(all steps refer somewhere)
* After creating database :

show create table students\G

* MySQL using AWS
* RDS-> Create database
* Standard create
* MYSQL
* Engine version: Mysql 8.0.32
* Templates: Free tier
* Settings ->

DB instance identifier: database-1

Credentials settings:

Self manages

Master password: 12345678

* Storage:

Storage autoscaling -> maximum storage threshold -> 22

* Connectivity:

Public access -> yes

VPC Security group: choose the one having 3306 port no

* Additional configuration:

Backup retention period: 5 days

Choose backup window

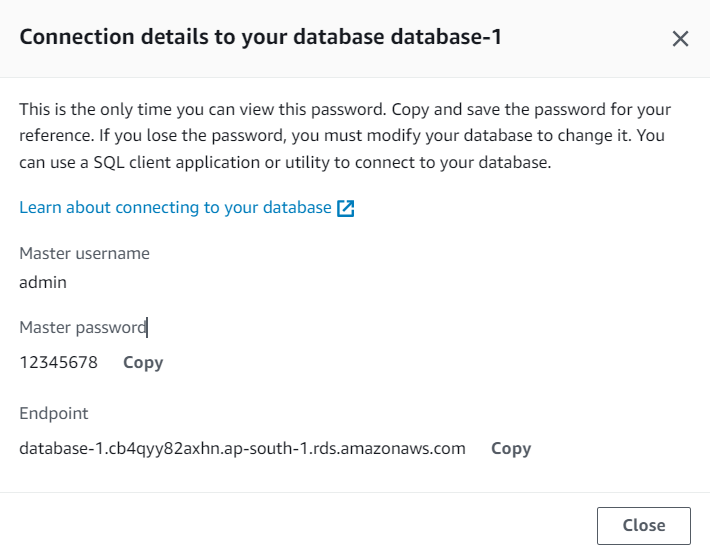
Start time: 22:05 UTC, Duration: 1hr

Maintenance:

Choose window

Start day: Sunday, Start time: 01:00 UTC

* Create database
* Copy endpoint link



* + - * Open terminal:
* mysql -u admin -h database-1.cb4qyy82axhn.ap-south-1.rds.amazonaws.com -p
* password: 12345678
* create database pdpu;
* use pdpu;
* create table student(

sid varchar(10),

name varchar(25),

primary key(sid));

* show databases;
* insert into student values('p01', 'RRR');
* insert into student values('p02', 'KKK');
  + - * Open vscode:
* In nodeAPI folder
* Open bash terminal

🡪 npm init -y

🡪npm i express mysql nodemon

🡪node index.js

🡪open browser: localhost:5000/api/user/

🡪Index.js file:

let express = require('express');

let mysql = require('mysql');

let connection = mysql.createConnection({

    user: "admin",

    password: "12345678",

    host: "database-1.cb4qyy82axhn.ap-south-1.rds.amazonaws.com",

    port: 3306,

    database: "pdpu"

})

connection.connect((err)=>{

    if(err){

        console.log(err);

    }else{

        console.log("Connected to the database");

    }

})

let pdpuApp = express();

pdpuApp.use("/api/user",function(req,res){

    let sqlQuery = "select \* from student";

    connection.query(sqlQuery,(err,results)=>{

        if(err){

            res.send(err.sqlMessage);

        }else{

            res.send(results);

        }

    })

});

pdpuApp.listen(5000,()=>{

    console.log("Server is running on port 5000");

})

* Open AWS
* Create one EC2 instance have Security group for port no: 22,80,5000
* Open bash terminal

🡪 ssh -i "databasekey.pem" ubuntu@ec2-44-212-0-254.compute-1.amazonaws.com

🡪 npm init -y

🡪 touch index.js

🡪 ls

🡪vi index.js

Paste the code of index.js from vscode

🡪npm I express mysql nodemon

🡪node index.js

🡪open browser: <http://44.212.0.254:5000/api/user/>

