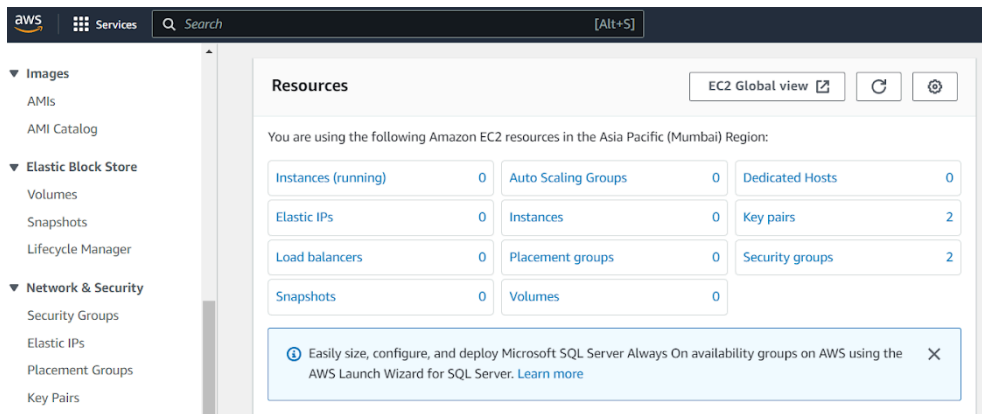


ASSIGNMENT – 10

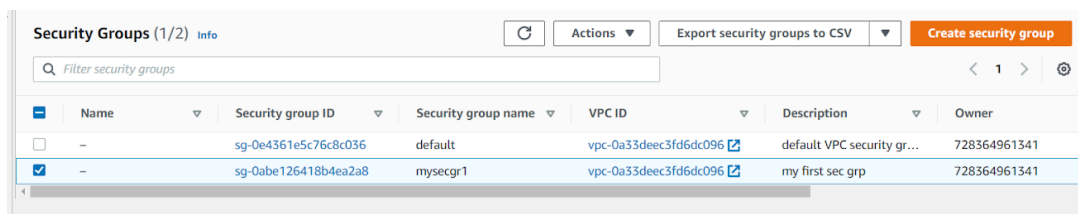
Problem Statement: Deploy project from GitHub to EC2 by creating new security group and user data.

Procedure:

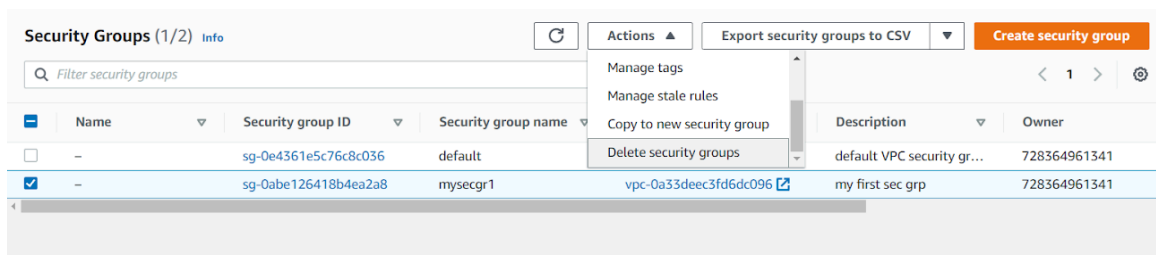
1. Sign in to your AWS account.
2. Go to your EC2 dashboard
3. Scroll down and Click on Security Groups option on the left side nav bar under Network & Security option.



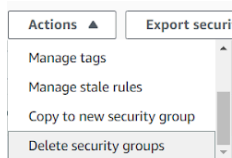
4. Select all the Security Groups other than the one named “default”.



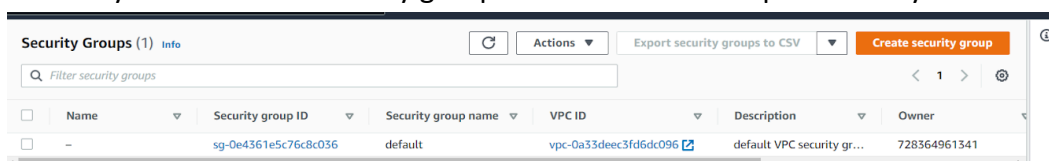
5. Then Click on the Actions button.



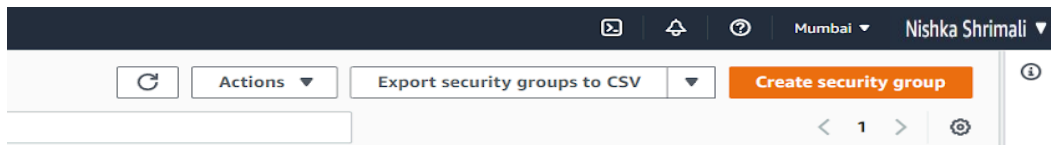
6. Scroll-Down the dropdown list until you find the “delete all security groups” option. Click on it.



7. Now only the “default” security group remains and we keep it that way.



8. Now click on the “Create Security Group” button.



9. Now start by giving a name to the security group and giving its description (anything).
Let the VPC remain unchanged.

 The screenshot shows the 'Create security group' page in the AWS console. Under the 'Basic details' section, the 'Security group name' field contains 'mysec1' and the 'Description' field also contains 'mysec1'. The 'VPC' dropdown menu is set to 'vpc-0a33deec3fd6dc096'. A note states: 'A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.'

10. Next, we will add Inbound Rules. Start adding by clicking the Add rule button. These include:

a) SSH

 This screenshot shows the configuration for the first inbound rule. The 'Type' is 'SSH', 'Protocol' is 'TCP', and 'Port range' is '22'. The 'Source' is set to 'Anywh...' with a search bar containing '0.0.0.0/0'. There is a 'Delete' button on the right.

b) HTTP

 This screenshot shows the configuration for the second inbound rule. The 'Type' is 'HTTP', 'Protocol' is 'TCP', and 'Port range' is '80'. The 'Source' is set to 'Anywh...' with a search bar containing '0.0.0.0/0'. There is a 'Delete' button on the right.

c) HTTPS

 This screenshot shows the configuration for the third inbound rule. The 'Type' is 'HTTPS', 'Protocol' is 'TCP', and 'Port range' is '443'. The 'Source' is set to 'Anywh...' with a search bar containing '0.0.0.0/0'. There is a 'Delete' button on the right.

d) Custom TCP

 This screenshot shows the configuration for the fourth inbound rule. The 'Type' is 'Custom TCP', 'Protocol' is 'TCP', and 'Port range' is '4000'. The 'Source' is set to 'Anywh...' with a search bar containing '0.0.0.0/0'. There is a 'Delete' button on the right.

The last one with custom TCP has a specific port range that we require to connect to our project. It has been specified in our index.js file (refer Ass9).

Now the final Inbound Rules section should look like this.

 This screenshot shows the final 'Inbound rules' section. It lists four rules: SSH (TCP, 22), HTTP (TCP, 80), HTTPS (TCP, 443), and Custom TCP (TCP, 4000). Each rule has a 'Source' of 'Anywh...' and a search bar containing '0.0.0.0/0'. Each rule has a 'Delete' button. At the bottom left, there is an 'Add rule' button.

11. Next outbound rules and all other sections remain unchanged. Now Click on the create security group button.

Outbound rules [Info](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Destination [Info](#) Description - optional [Info](#)

All traffic [▼](#) All All Custom [▼](#) [X](#) [Delete](#)

[Add rule](#)

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags

[Cancel](#) [Create security group](#)

12. Now go back to the security groups list and click on the security group ID of the newly created Security Group.

Security Groups (2) [Info](#)

<input type="checkbox"/>	Name ▼	Security group ID ▼	Security group name ▼
<input type="checkbox"/>	–	sg-0493398d43b761e55	mysec1
<input type="checkbox"/>	–	sg-0e4361e5c76c8c036	default

Security group name [mysec1](#) Security group ID [sg-0493398d43b761e55](#) Description [mysec1](#) VPC ID [vpc-0a33deec3fd6dc096](#)

Owner [728364961341](#) Inbound rules count 4 Permission entries Outbound rules count 1 Permission entry

[Inbound rules](#) [Outbound rules](#) [Tags](#)

[You can now check network connectivity with Reachability Analyzer](#) [Run Reachability Analyzer](#) [X](#)

Inbound rules (4) [Manage tags](#) [Edit inbound rules](#)

<input type="checkbox"/>	Name ▼	Security group rule... ▼	IP version ▼	Type ▼	Protocol ▼	Port range
<input type="checkbox"/>	–	sgr-0b77b32c36bf02194	IPv4	HTTPS	TCP	443
<input type="checkbox"/>	–	sgr-02f9143435809d0...	IPv4	SSH	TCP	22
<input type="checkbox"/>	–	sgr-08f9ba0e0aeeceae64	IPv4	HTTP	TCP	80
<input type="checkbox"/>	–	sgr-0d92a3e25bf3add37	IPv4	Custom TCP	TCP	4000

After clicking we can view the inbound rules that we added during its creation.

13. Now we go to the instances section from the left side nav bar.
14. Now we Create a new EC2 instance. Click on the Launch Instance button.

Instances [Info](#) [Connect](#) [Instance state](#) [▼](#) [Actions](#) [▼](#) [Launch instances](#) [▼](#)

<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼	Public IPv4 DNS
No instances								
You do not have any instances in this region								

Now,







- a) Give the name, Select Ubuntu as OS, Select a keypair or generate a new one if none is available.

[Launch an instance](#) [Info](#)

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)
 An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux  macOS  Ubuntu  Windows  Red Hat  S  [Browse more AMIs](#)
 Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

▼ **Key pair (login)** [Info](#)
 You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

debkey2 ▼ [Create new key pair](#)

- b) Then under Network settings select the Select Existing Security Group option.
 c) Now under the security groups dropdown menu select the one we just created.

Firewall (security groups) [Info](#)
 A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Create security group ☒ Select existing security group

Security groups [Info](#)

Select security groups ▼

mysec1 sg-0493398d43b761e55 ✕
 VPC: vpc-0a33deec3fd6dc096

[Compare security group rules](#)

- d) Now scroll down and click on the Advanced Details option.
 e) Now again scroll-down to the newly appeared sub-sections until you find User Data section.
- f) Write the following commands in the given box. Remember this user data is given to execute the given commands once the server starts. So essentially, we can provide all commands that we entered in our Assignment 9 previously and execute them without connecting to our server itself!! They will be executed sequentially.
- **#!/bin/bash**
 - **apt-get update**
 - **apt-get install -y nginx**
 - **systemctl start nginx**
 - **systemctl enable nginx**
 - **apt-get install -y git**
 - **curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -**
 - **apt-get install -y nodejs**

Now, here is a caveat. We have created a private repository in GitHub. So, whenever we run the git clone command it asks for our username and password. Hence this cannot be executed directly through our User Data instructions. We have to connect manually and enter all commands starting from the git clone command.

g) Now we click on the launch instance button.

User data - optional [Info](#)
Enter user data in the field.

```
#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
```

☐ User data has already been base64 encoded

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-02eb7a4783e7e9317

Virtual server type (instance type)
t2.micro

Firewall (security group)
mysec1

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is

CancelLaunch instance

[Review commands](#)

15. Now we Click on the ‘Instance Id’ link of our newly created server in our Instances list.

Instances (1) [Info](#)

RefreshConnectInstance state ▼Actions ▼Launch instance

Find instance by attribute or tag (case-sensitive)

<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼
<input type="checkbox"/>	debserver1	i-0a6ab24417f81fffb	Running	t2.micro	Initializing	No alarms +	ap-south-1a

16. Now click on the connect button

Instance summary for [i-0a6ab24417f81fffb \(debserver1\)](#) [Info](#)

RefreshConnectInstance state ▼Actions ▼

Updated less than a minute ago

Instance ID i-0a6ab24417f81fffb (debserver1)	Public IPv4 address 3.110.134.34 open address	Private IPv4 addresses 172.31.41.246
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-110-134-34.ap-south-1.compute.amazonaws.com open address

17. Again, click on the connect button

Connect to instance [Info](#)
Connect to your instance [i-0a6ab24417f81fffb \(debserver1\)](#) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
[i-0a6ab24417f81fffb \(debserver1\)](#)

Public IP address
[3.110.134.34](#)

User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.

Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Connect

18. After this anew Tab will open with a Bash Terminal that is of our remote EC2 server!
Here we can type all our required commands that we used to type in a similar terminal by connecting to our remote server through our Bitwise SSH client software in our previous assignments.

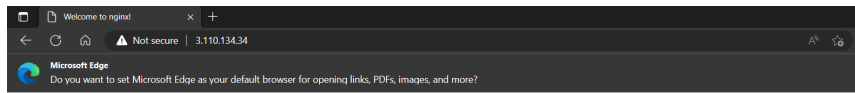
19. Now copy and paste the Public IPv4 address of your EC2 instance in another browser.

[i-0a6ab24417f81fffb \(debserver1\)](#)

Public IPv4 address copied

[3.110.134.34](#) | [open address](#)

Instance state
Running



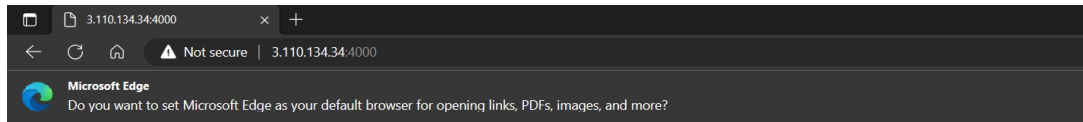
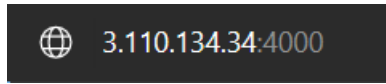
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

20. Now append the port no. 4000 (for our case) to the IP address in the browser with a “:” sign.



We have successfully Deployed a project from GitHub to EC2 by creating a new Security group and User Data.