NATIONAL INSTITUTE OF TECHNOLOGY, DELHI

Project Report

On

Using A Cloud Service (AWS)

Network Programming
(CSB-351)

Submitted By
Mahendra soni
171210036

Content

Introduction

In 2006, Amazon Web Services (AWS) started to offer IT services to the market in the form of web services, which is now a days known as cloud computing. With this cloud, we need not plan for servers and other IT infrastructure which takes up much of time in advance. Instead, these service scan instantly spin up hundreds or thousands of servers in minutes and deliver results faster. We pay only for what we use with no up-front expenses and no long-term commitments, which makes AWS cost efficient. Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers multitude of businesses in 190 countries around the world.

What is Cloud Computing

Cloud computing is an internet-based computing service in which large groups of remote servers are networked to allow centralized data storage, and online access to computer services or resources.

Using cloud computing, organizations can use shared computing and storage resources rather than building, operating, and improving infrastructure on their own.

Cloud computing is a model that enables the following features.

- Users can provision and release resources on-demand.
- Resources can be scaled up or down automatically, depending on the load.
- Resources are accessible over a network with proper security.
- Cloud service providers can enable a pay-as-you-go model, where customers are charged based on the type of resources and per usage.

Types of Clouds

There are three types of clouds – Public, Private, and Hybrid cloud.

Public Cloud

In public cloud, the third-party service providers make resources and services available to their customers via Internet. Customer's data and related security is with the service providers' owned infrastructure.

Private Cloud

A private cloud also provides almost similar features as public cloud, but the data and services are managed by the organization or by the third party only for the customer's organization. In this type of cloud, major control is over the infrastructure so security related issues are minimized.

Hybrid Cloud

A hybrid cloud is the combination of both private and public cloud. The decision to run on private or public cloud usually depends on various parameters like sensitivity of data and applications, industry certifications and required standards, regulations, etc.

Cloud Service Models

There are three types of service models in cloud – IaaS, PaaS, and SaaS.

IaaS

IaaS stands for **Infrastructure as a Service**. It provides users with the capability to provision processing, storage, and network connectivity on demand. Using this service model, the customers can develop their own applications on these resources.

PaaS

PaaS stands for **Platform as a Service**. Here, the service provider provides various services like databases, queues, workflow engines, e-mails, etc. to their customers. The customer can then use these components for building their own applications. The services, availability of resources and data backup are handled by the service provider that helps the customers to focus more on their application's functionality.

SaaS

SaaS stands for **Software as a Service**. As the name suggests, here the third-party providers provide end-user applications to their customers with some administrative capability at the application level, such as the ability to create and manage their users. Also some level of customizability is possible such as the customers can use their own corporate logos, colors, etc.

Advantages of Cloud Computing

Here is a list of some of the most important advantages that Cloud Computing has to offer –

- **Cost-Efficient** – Building our own servers and tools is time-consuming as well as expensive as we need to order, pay for, install, and configure expensive hardware, long before we need it. However, using cloud computing, we only pay for the amount we use and when we use the computing resources. In this manner, cloud computing is cost efficient.

- **Reliability** – A cloud computing platform provides much more managed, reliable and consistent service than an in-house IT infrastructure. It guarantees 24x7 and 365 days of service. If any of the server fails, then hosted applications and services can easily be transited to any of the available servers.

- **Unlimited Storage** – Cloud computing provides almost unlimited storage capacity, i.e., we need not worry about running out of storage space or increasing our current storage space availability. We can access as much or as little as we need.

- **Backup & Recovery** – Storing data in the cloud, backing it up and restoring the same is relatively easier than storing it on a physical device. The cloud service providers also have enough technology to recover our data, so there is the convenience of recovering our data anytime.

- **Easy Access to Information** – Once you register yourself in cloud, you can access your account from anywhere in the world provided there is internet connection at that point. There are various storage and security facilities that vary with the account type chosen.

Disadvantages of Cloud Computing

Although Cloud Computing provides a wonderful set of advantages, it has some drawbacks as well that often raise questions about its efficiency.

Security issues

Security is the major issue in cloud computing. The cloud service providers implement the best security standards and industry certifications, however, storing data and important files on external service providers always bears a risk.

AWS cloud infrastructure is designed to be the most flexible and secured cloud network. It provides scalable and highly reliable platform that enables customers to deploy applications and data quickly and securely.

Technical issues

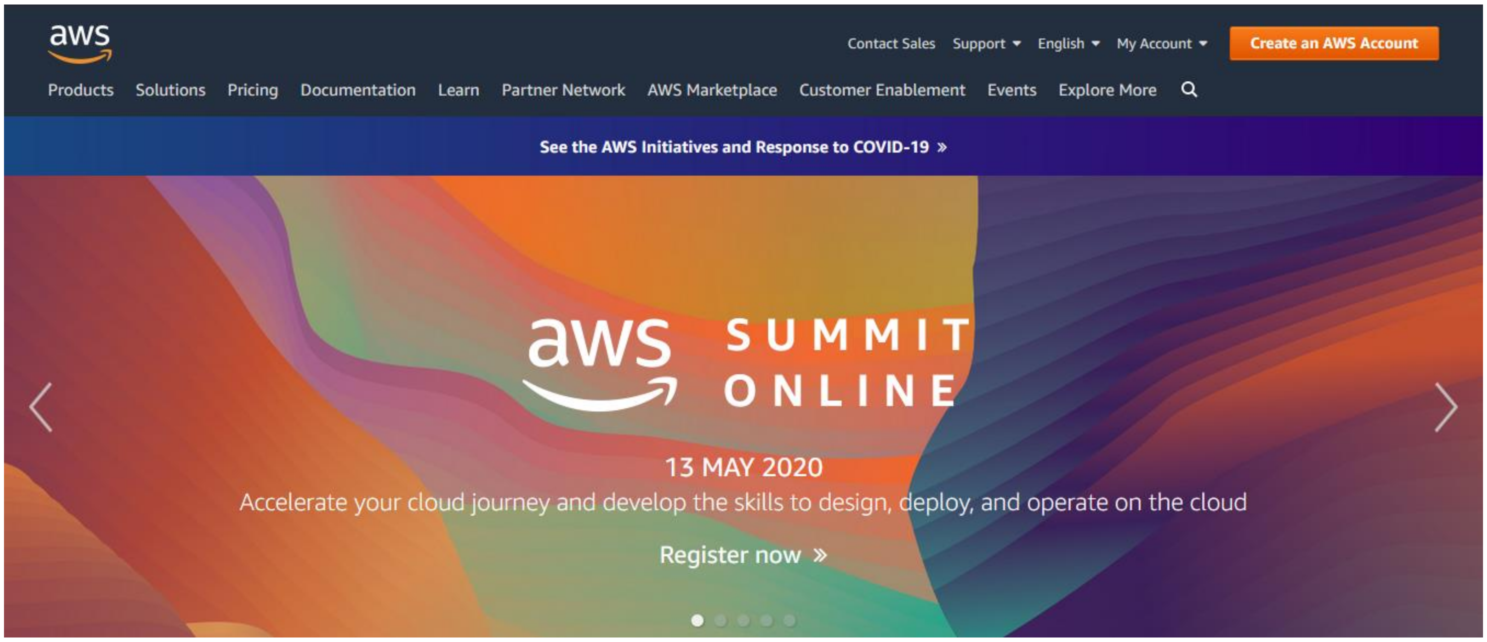
As cloud service providers offer services to number of clients each day, sometimes the system can have some serious issues leading to business processes temporarily being suspended. Additionally, if the internet connection is offline then we will not be able to access any of the applications, server, or data from the cloud.

Not easy to switch service providers

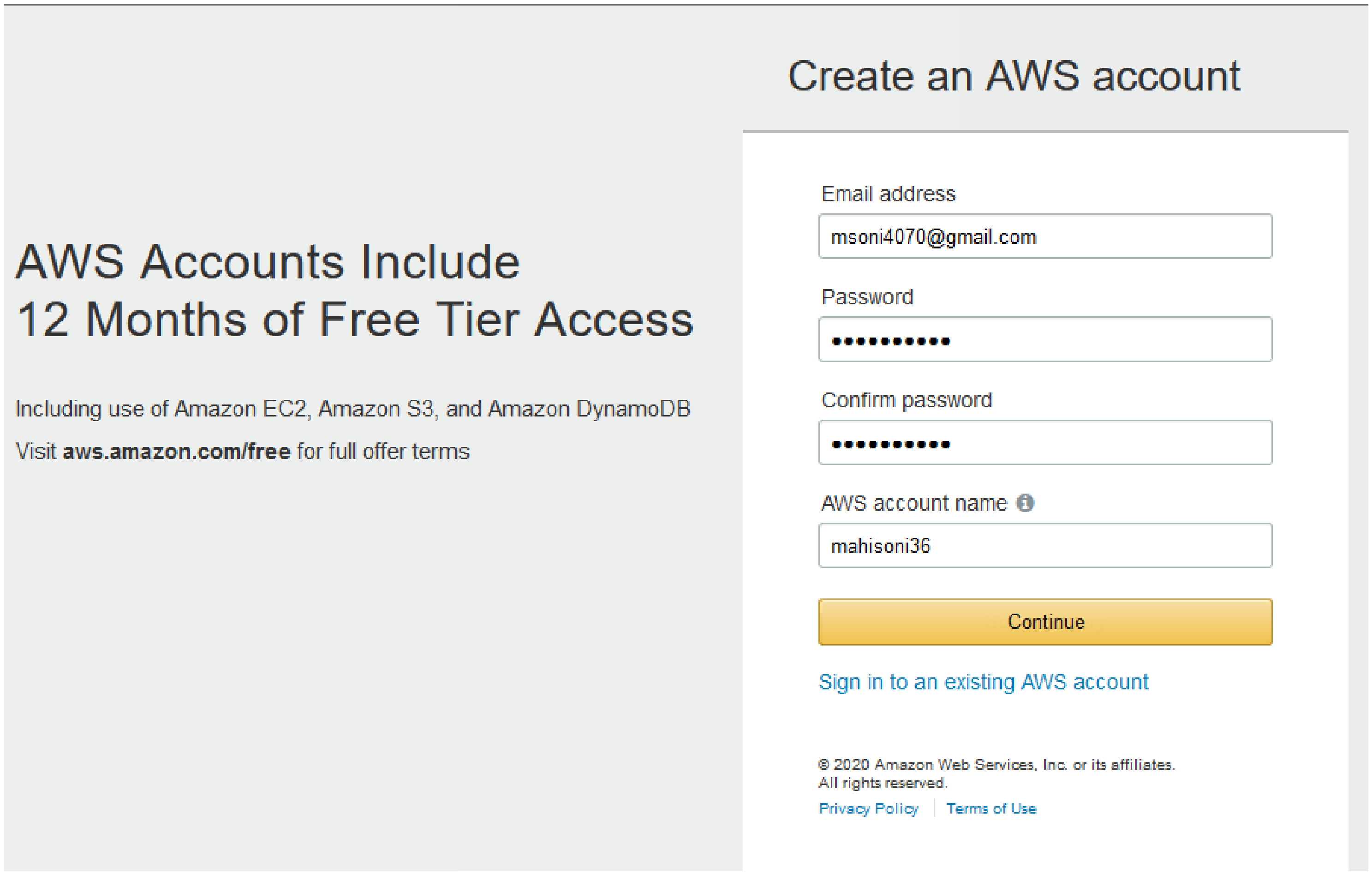
Cloud service providers promises vendors that the cloud will be flexible to use and integrate, however switching cloud services is not easy. Most organizations may find it difficult to host and integrate current cloud applications on another platform. Interoperability and support issues may arise such as applications developed on Linux platform may not work properly on Microsoft Development Framework (.Net).

Creating an Account on AWS

1) **Visit AWS Website**



2) **Select Create an AWS Account and fill the necessary fields**



Create an AWS account

AWS Accounts Include
12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB

Visit **aws.amazon.com/free** for full offer terms

Email address: msoni4070@gmail.com

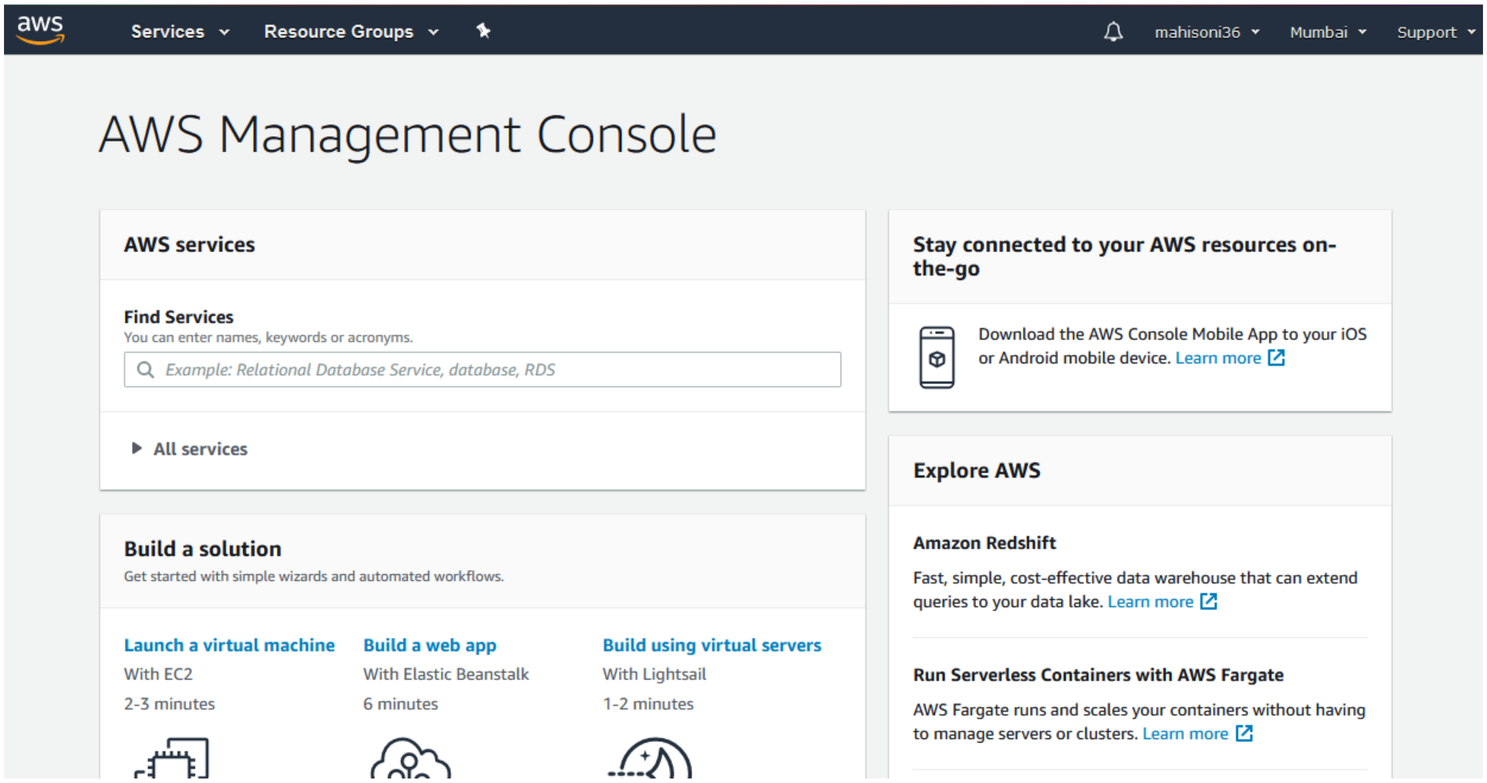
Password: ●●●●●●●●●●●

Confirm password: ●●●●●●●●●●●

AWS account name: mahisoni36

© 2020 Amazon Web Services, Inc. or its affiliates.
All rights reserved.

3) **After filing all the essential details you will be directed to your AWS Management Console page.**



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
2-3 minutes

Build a web app
With Elastic Beanstalk
6 minutes

Build using virtual servers
With Lightsail
1-2 minutes

Stay connected to your AWS resources on-the-go

Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

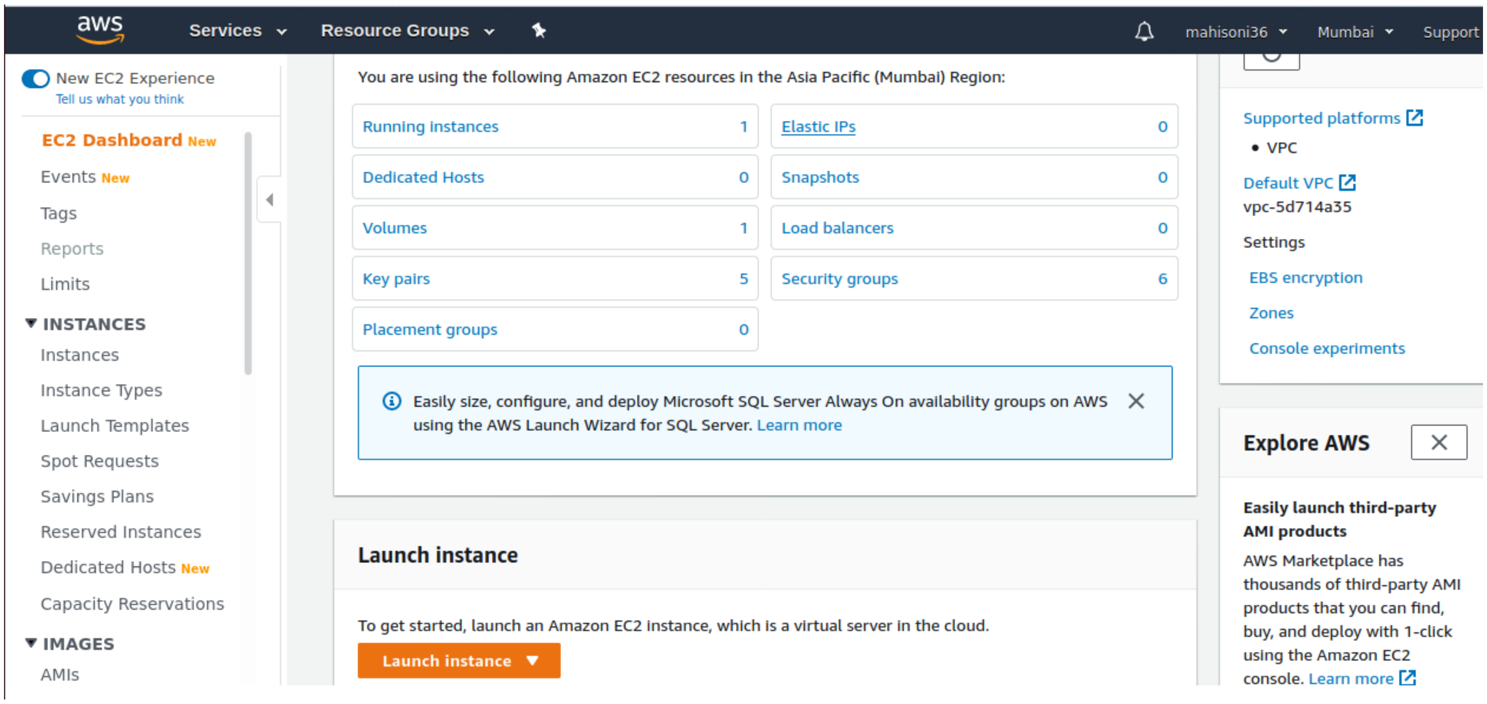
Explore AWS

Amazon Redshift
Fast, simple, cost-effective data warehouse that can extend queries to your data lake. Learn more

Run Serverless Containers with AWS Fargate
AWS Fargate runs and scales your containers without having to manage servers or clusters. Learn more

Creating Ec2 Instance (Server)

In the find Services field on the AWS Management Console page type EC2, you will be directed to the following page.



You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Running instances	1	Elastic IPs	0
Dedicated Hosts	0	Snapshots	0
Volumes	1	Load balancers	0
Key pairs	5	Security groups	6
Placement groups	0		

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

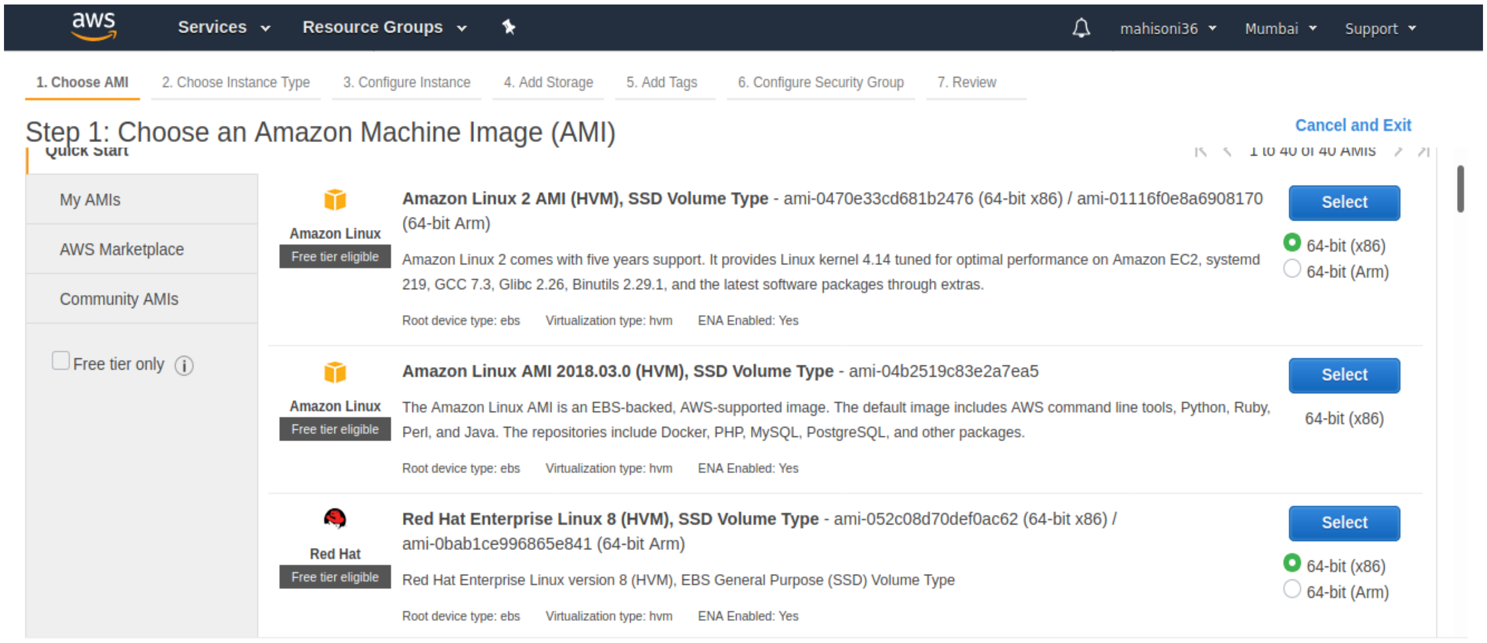
Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Easily launch third-party AMI products

AWS Marketplace has thousands of third-party AMI products that you can find, buy, and deploy with 1-click using the Amazon EC2 console. Learn more

Select **Launch Instance** option, you will be directed to the following page



Step 1: Choose an Amazon Machine Image (AMI)

Amazon Linux — Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0470e33cd681b2476 (64-bit x86) / ami-01116f0e8a6908170 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select — [✓ 64-bit (x86)] 64-bit (Arm)

Amazon Linux — Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-04b2519c83e2a7ea5

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select — 64-bit (x86)

Red Hat — Free tier eligible

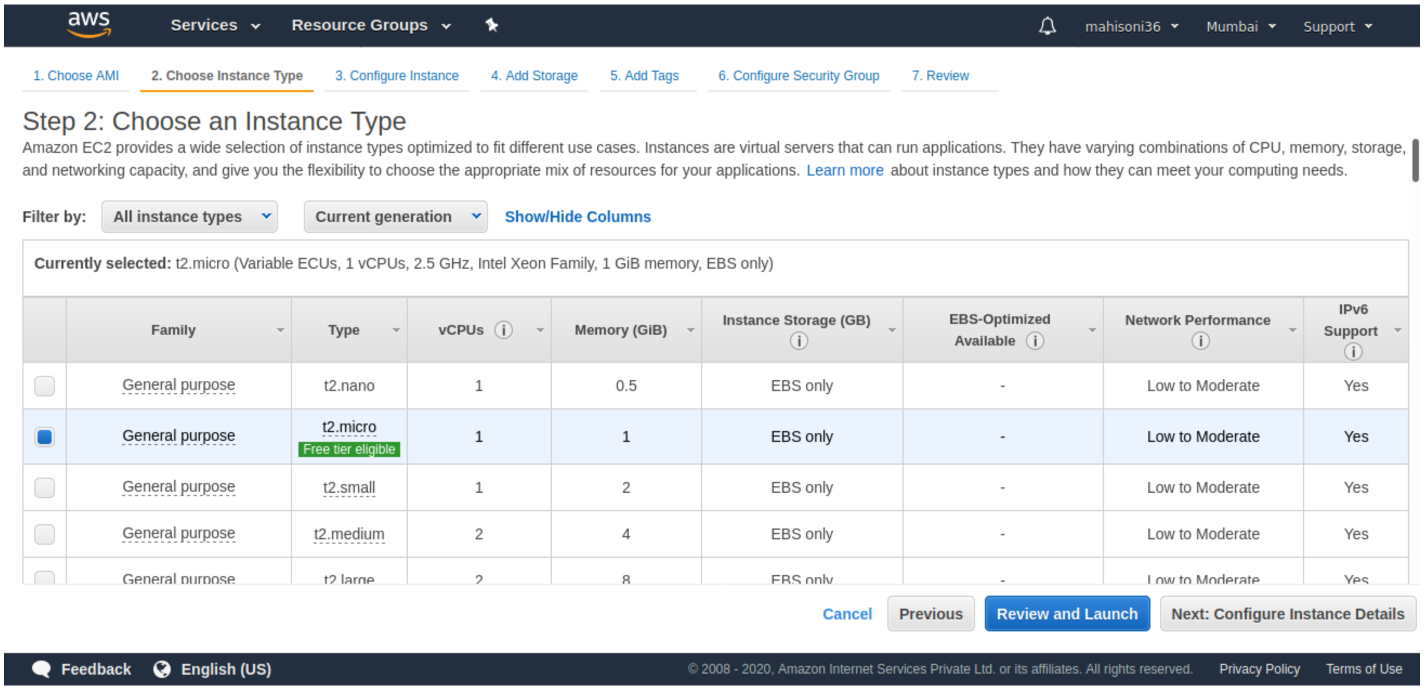
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-052c08d70def0ac62 (64-bit x86) / ami-0bab1ce996865e841 (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select — [✓ 64-bit (x86)] 64-bit (Arm)

Choose a **Machine Image** (i chose Amazon linux 2 AMI), select next after completion and you will be further directed to the following page.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

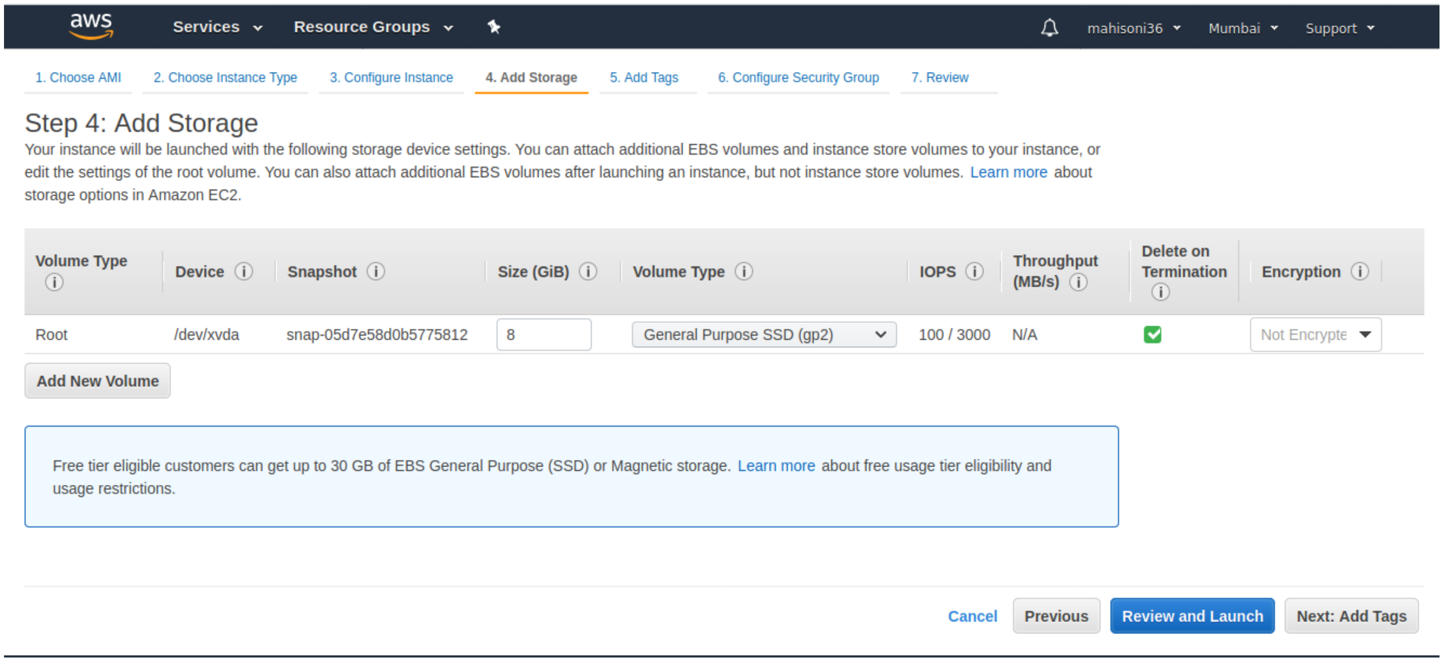
Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes

Cancel
Previous
Review and Launch
Next: Configure Instance Details
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Choose an **Instance Type** and select Next



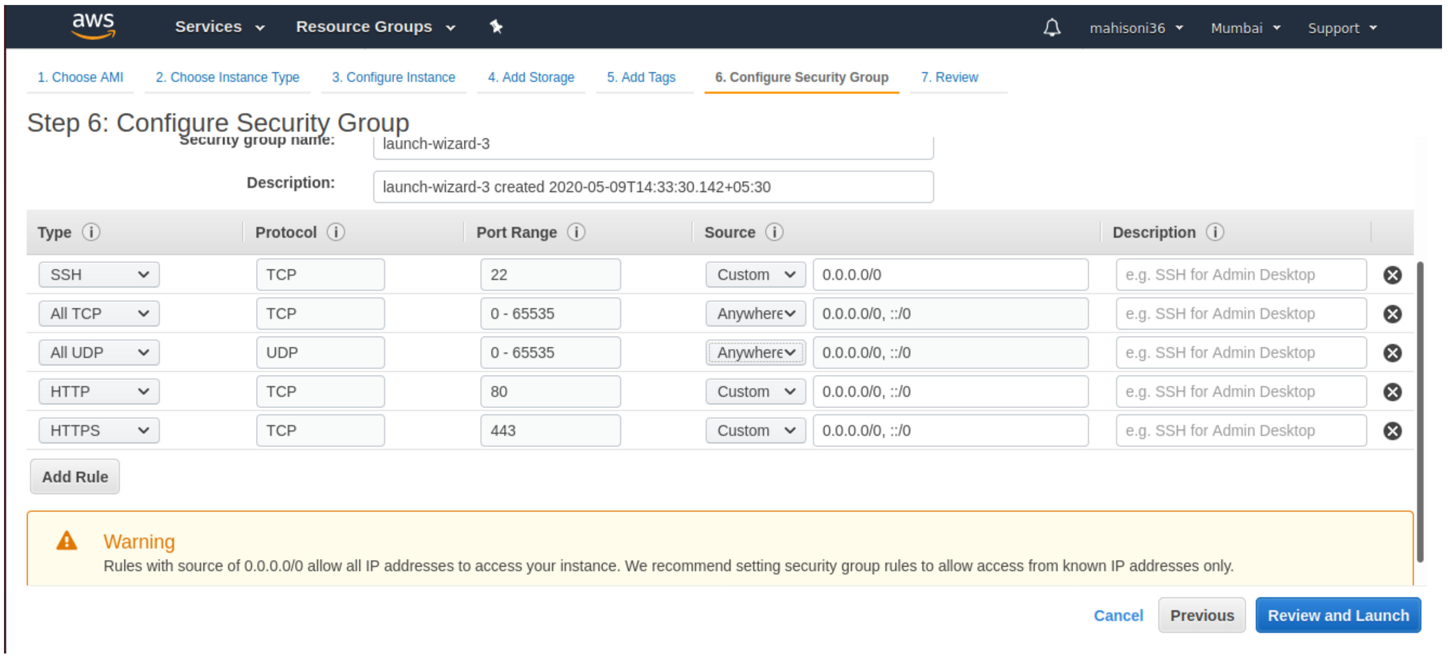
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-05d7e58d0b5775812	8	General Purpose SSD (gp2)	100 / 3000	N/A	✓	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Add Storage allows you to configure your machine’s storage. Select next after completion



Step 6: Configure Security Group

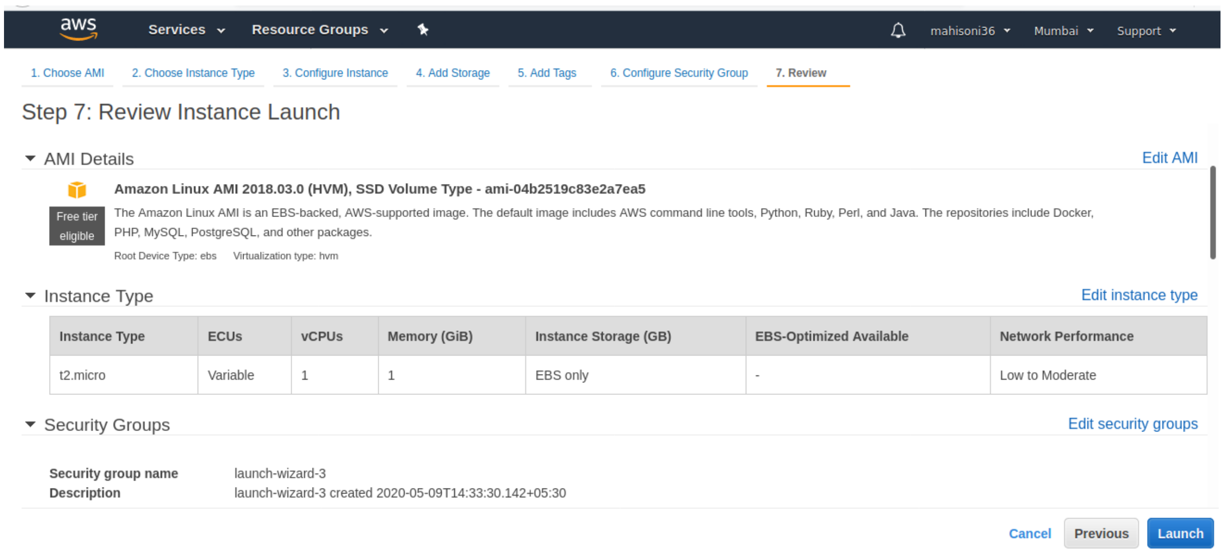
Security group name: launch-wizard-3

Description: launch-wizard-3 created 2020-05-09T14:33:30.142+05:30

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
All TCP	TCP	0 - 65535	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
All UDP	UDP	0 - 65535	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure Security Group (I added four different types of Configuration (All TCP, ALL UDP, HTTP, HTTPS). Click on Review and launch.



Step 7: Review Instance Launch

AMI Details

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-04b2519c83e2a7ea5

Free tier eligible

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

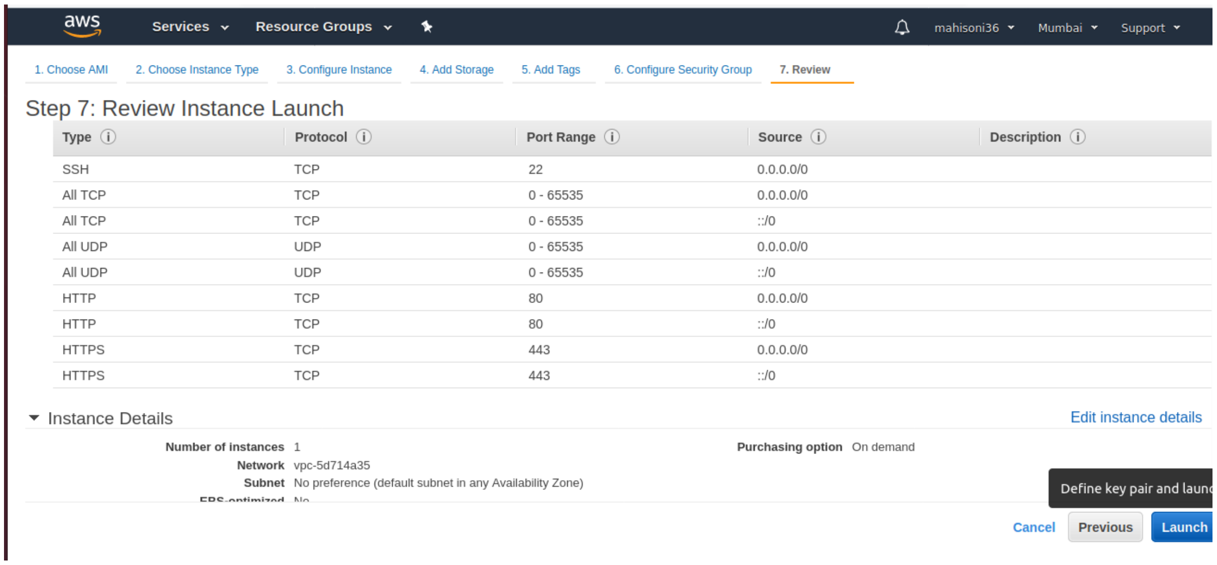
Root Device Type: ebs   Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security group name	launch-wizard-3
Description	launch-wizard-3 created 2020-05-09T14:33:30.142+05:30



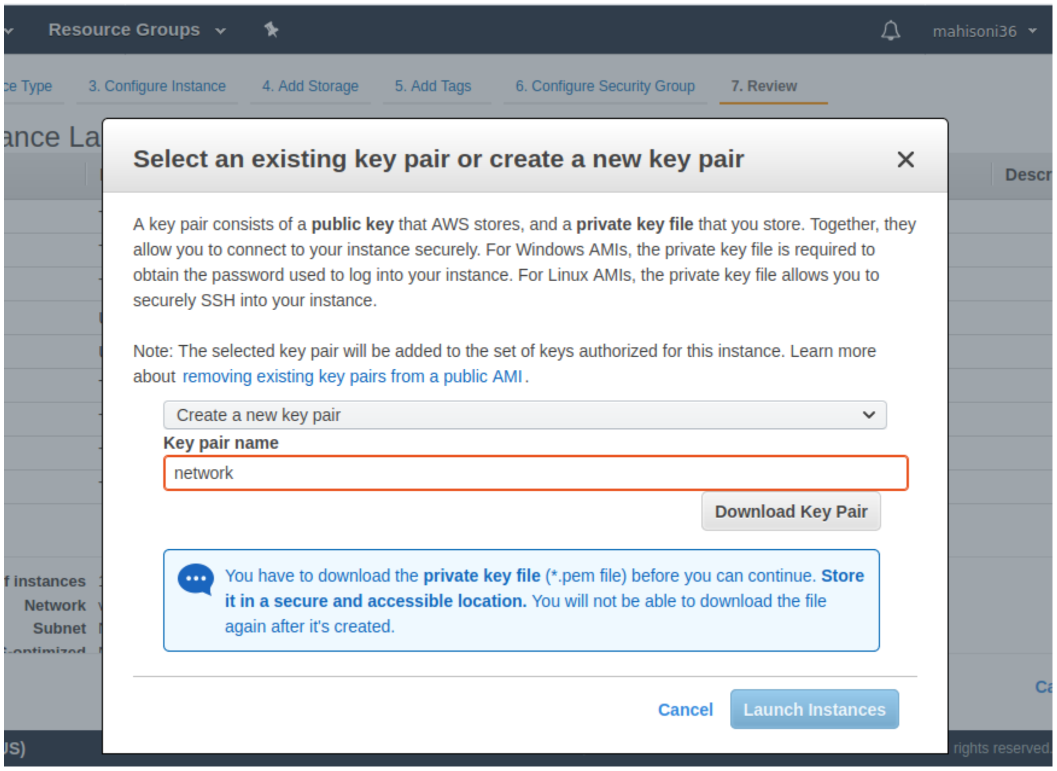
Step 7: Review Instance Launch

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
All TCP	TCP	0 - 65535	0.0.0.0/0	
All TCP	TCP	0 - 65535	::/0	
All UDP	UDP	0 - 65535	0.0.0.0/0	
All UDP	UDP	0 - 65535	::/0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	
HTTPS	TCP	443	0.0.0.0/0	
HTTPS	TCP	443	::/0	

Instance Details

Number of instances	1	Purchasing option	On demand
Network	vpc-5d714a35		
Subnet	No preference (default subnet in any Availability Zone)		
EBS-optimized	No		

This page lets you review all your selections and lets you make changes before launching the instance (server). Click **Launch** after reviewing all the selections.
After selecting launch a dialog box pops up which asks you to generate a key-value pair which will be used to log into the server. Make a **key value** pair and download the **key** on your system.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

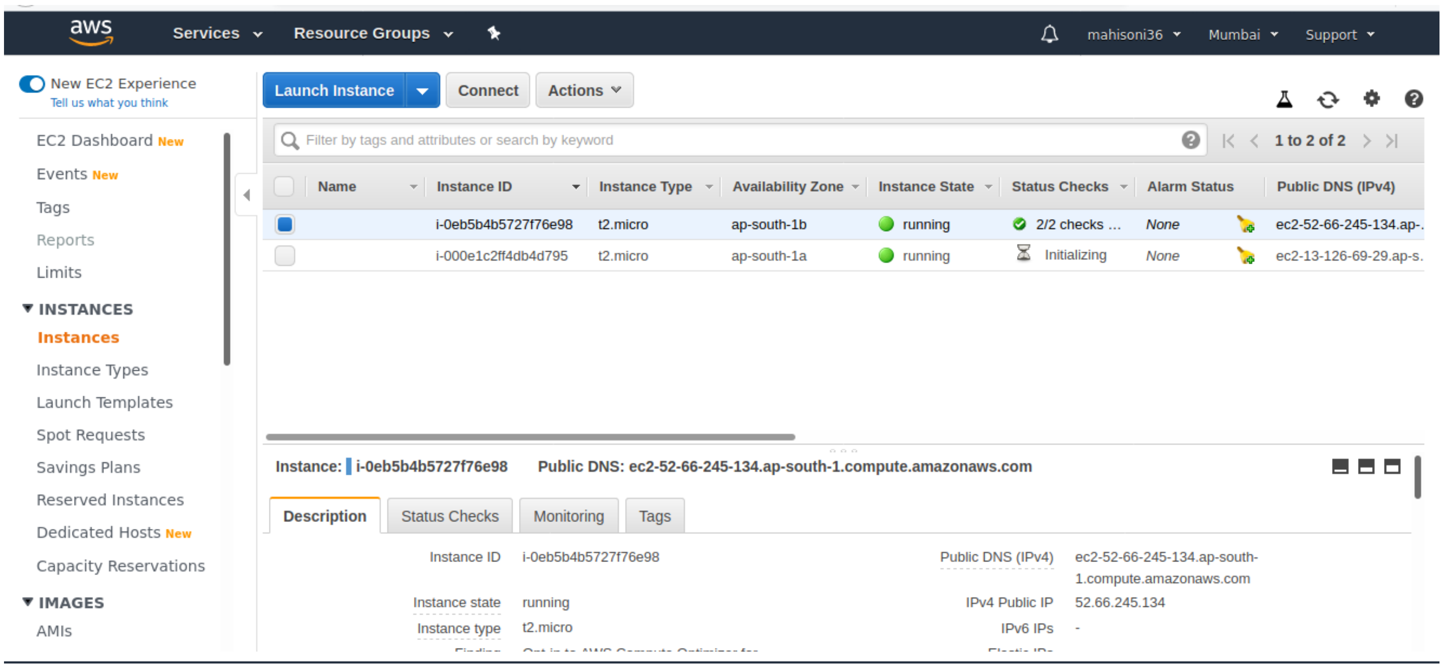
Key pair name: network

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel | Launch Instances

Click on **Launch Instances** after completion of the above steps. Wait for a while and you will be directed to your running instance page.



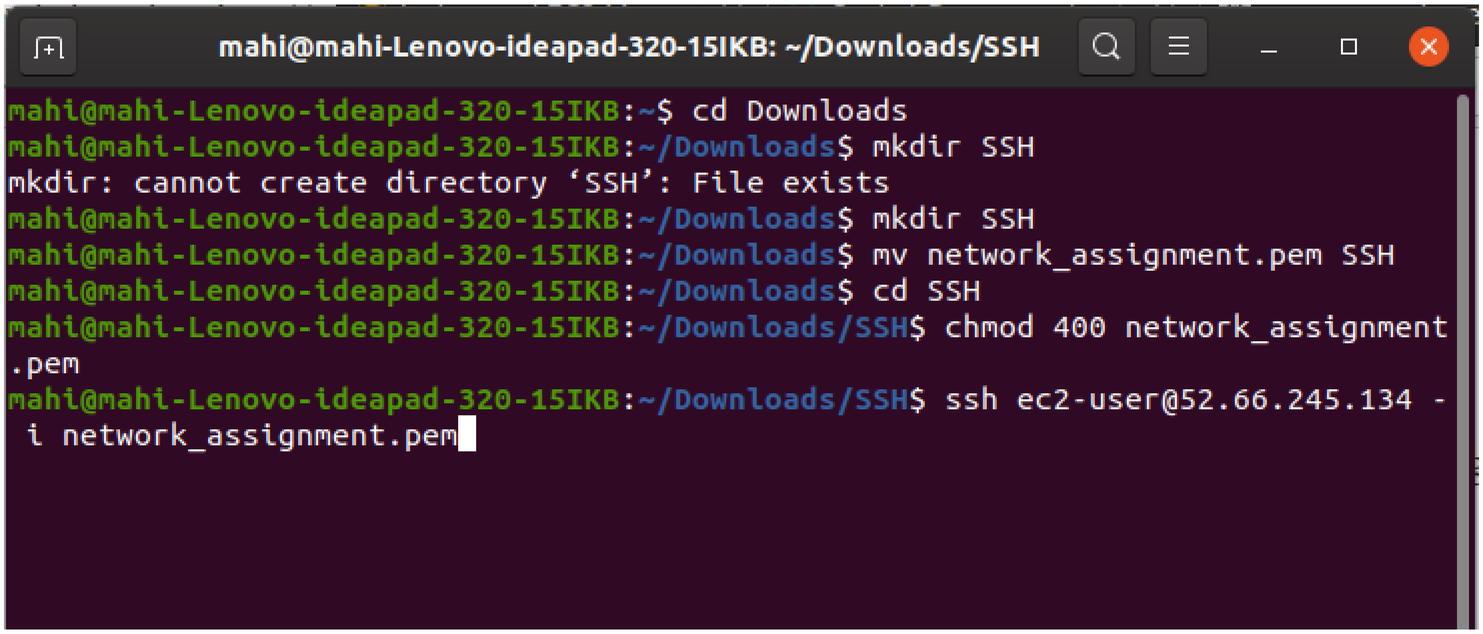
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
	i-0eb5b4b5727f76e98	t2.micro	ap-south-1b	● running	✓ 2/2 checks ...	None	ec2-52-66-245-134.ap-.
	i-000e1c2ff4db4d795	t2.micro	ap-south-1a	● running	Initializing	None	ec2-13-126-69-29.ap-s.

Instance: i-0eb5b4b5727f76e98 Public DNS: ec2-52-66-245-134.ap-south-1.compute.amazonaws.com

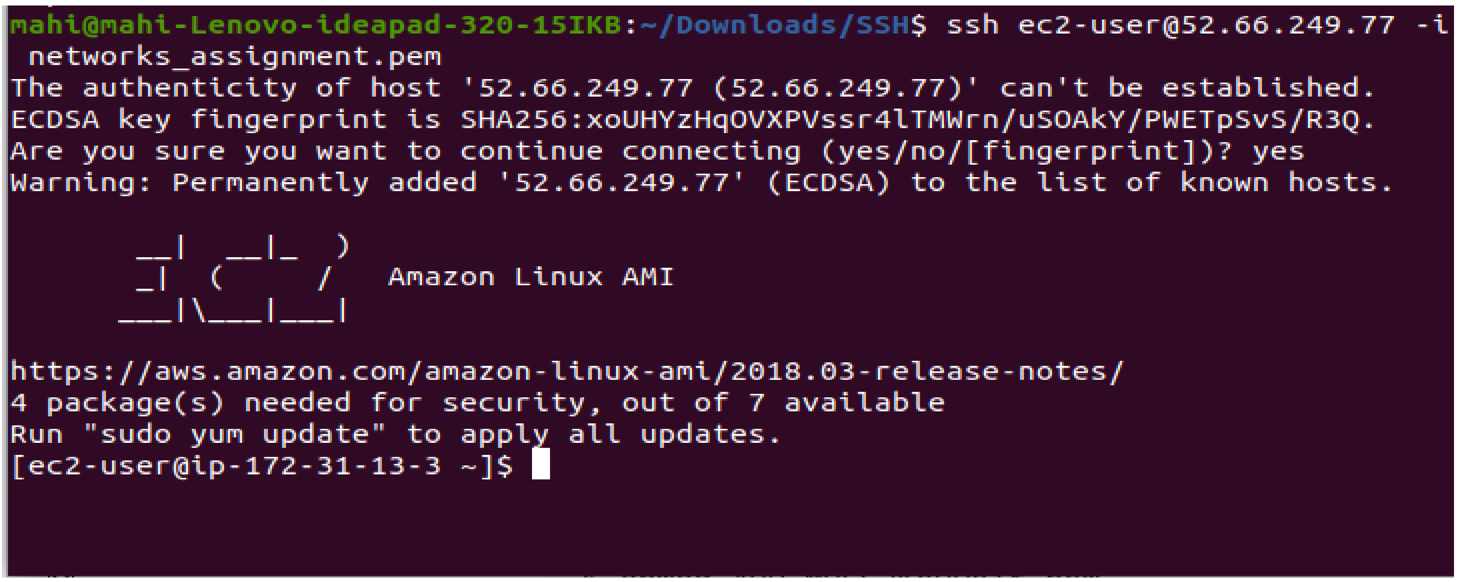
Instance ID	i-0eb5b4b5727f76e98	Public DNS (IPv4)	ec2-52-66-245-134.ap-south-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	52.66.245.134
Instance type	t2.micro	IPv6 IPs	-

Connecting Instance to Local Machine (computer)

- ✓ Open Terminal in Ubuntu.
- ✓ Type the following command to open the folder where downloaded is **key-file** exist.
    - o cd Downloads
- ✓ Create a new Directory **SSH**
    - o mkdir SSH
- ✓ Move the key-file into SSH
    - o mv network_assignment.pem SSH
- ✓ Now run the instance with following command-
    - o ssh ec2-user@52.66.245.134 -i network_assignment.pem



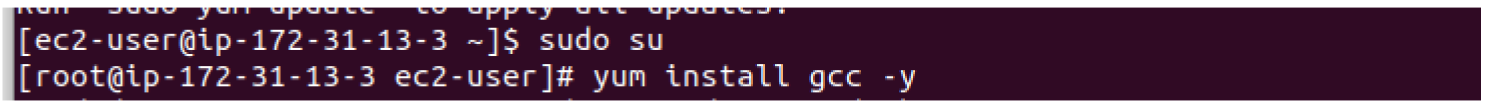
```
mahi@mahi-Lenovo-ideapad-320-15IKB:~$ cd Downloads
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads$ mkdir SSH
mkdir: cannot create directory ‘SSH’: File exists
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads$ mkdir SSH
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads$ mv network_assignment.pem SSH
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads$ cd SSH
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads/SSH$ chmod 400 network_assignment
.pem
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads/SSH$ ssh ec2-user@52.66.245.134 -
 i network_assignment.pem
```



```
mahi@mahi-Lenovo-ideapad-320-15IKB:~/Downloads/SSH$ ssh ec2-user@52.66.249.77 -i
 networks_assignment.pem
The authenticity of host '52.66.249.77 (52.66.249.77)' can't be established.
ECDSA key fingerprint is SHA256:xoUHYzHqOVXPVssr4lTMWrn/uSOAkY/PWETpSvS/R3Q.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '52.66.249.77' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-13-3 ~]$ 
```

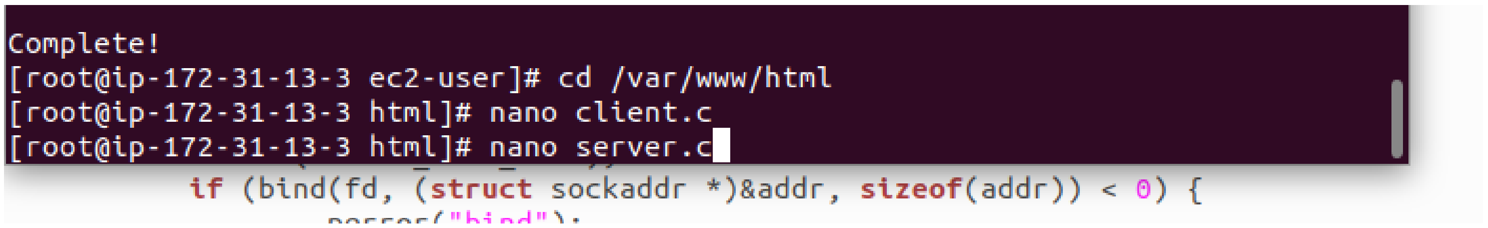
Write code and save in Cloud Storage

Type command **sudo su** to create root user.



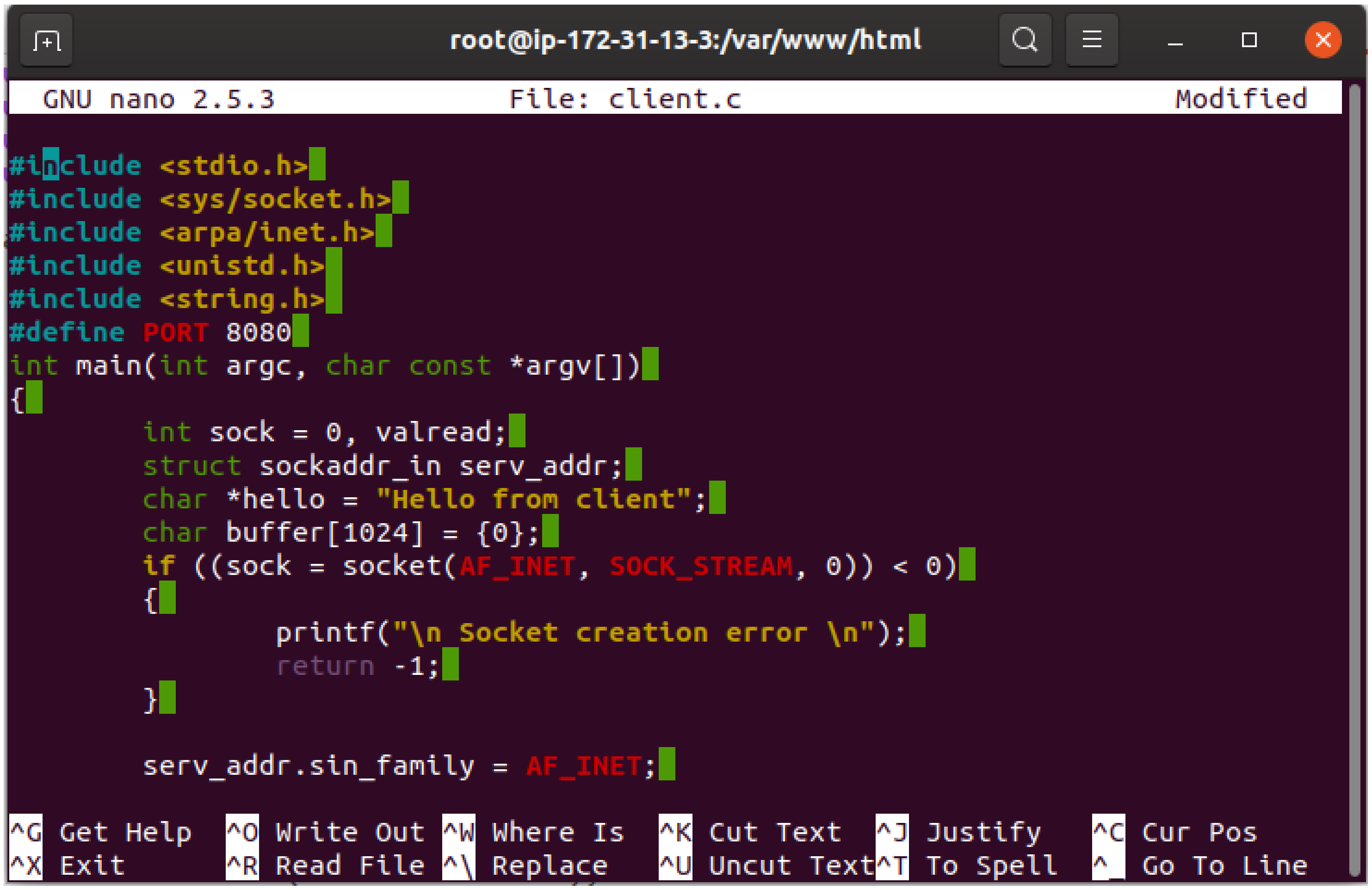
```
[ec2-user@ip-172-31-13-3 ~]$ sudo su
[root@ip-172-31-13-3 ec2-user]# yum install gcc -y
```

- ✓ For write client & server programs. We have to run these given commands
  - Yum install httpd -y
- ✓ To Open directory where we will save client and server side code
  - cd var/www/html
- ✓ To open client.c file
  - nano client.c
- ✓ To open server.c file
  - nano server.c



```
Complete!
[root@ip-172-31-13-3 ec2-user]# cd /var/www/html
[root@ip-172-31-13-3 html]# nano client.c
[root@ip-172-31-13-3 html]# nano server.c
            if (bind(fd, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
```

Below window open when I typed command **nano client.c**
After written complete code I pressed **ctrl +X - > yes - > enter** to save the code file.



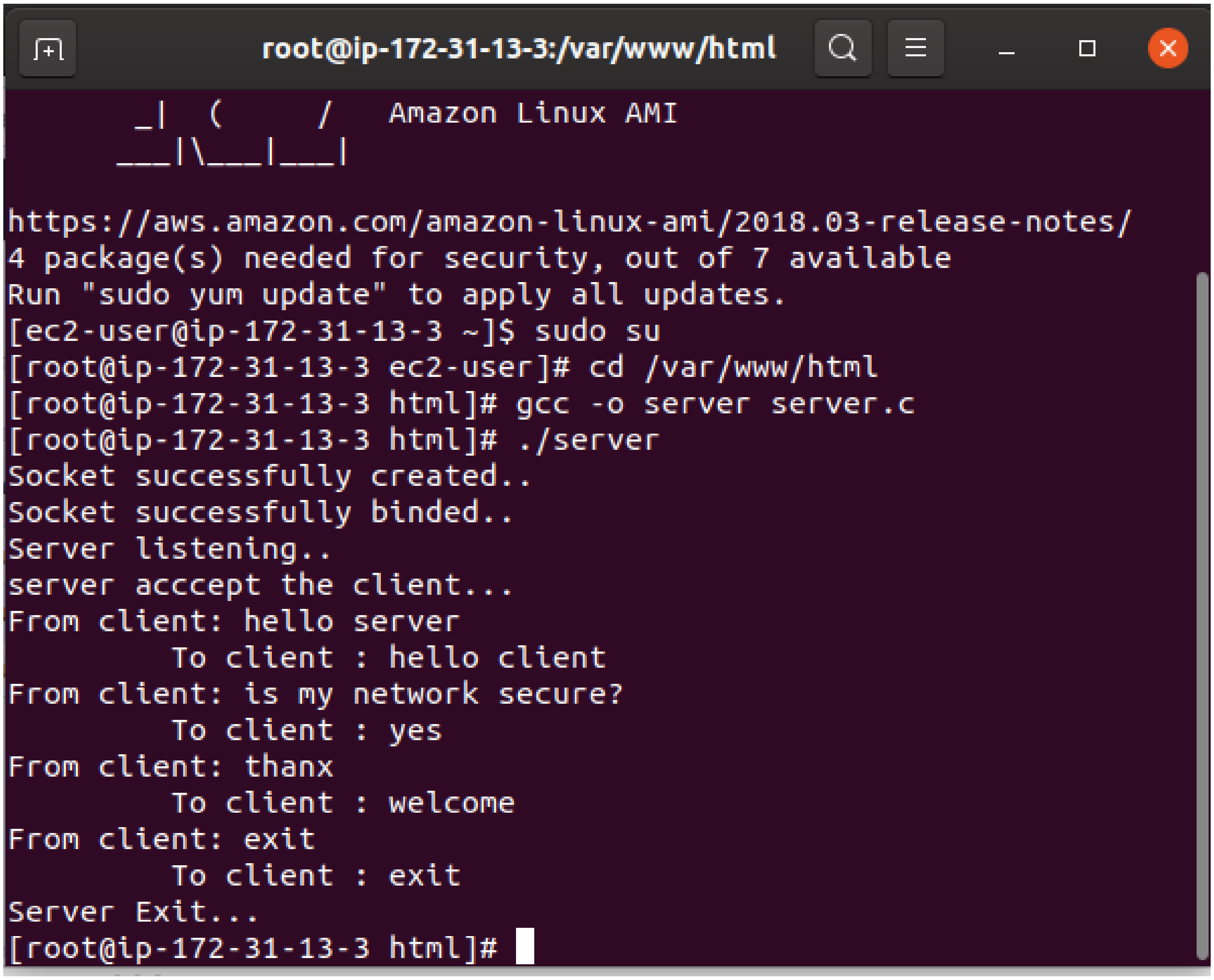
```
  GNU nano 2.5.3              File: client.c                          Modified

#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
        int sock = 0, valread;
        struct sockaddr_in serv_addr;
        char *hello = "Hello from client";
        char buffer[1024] = {0};
        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf("\n Socket creation error \n");
                return -1;
        }

        serv_addr.sin_family = AF_INET;

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell    ^_ Go To Line
```

Run codes on Instance

After written both client & server side programs, I typed the command **yum install gcc -y** to run **C** files

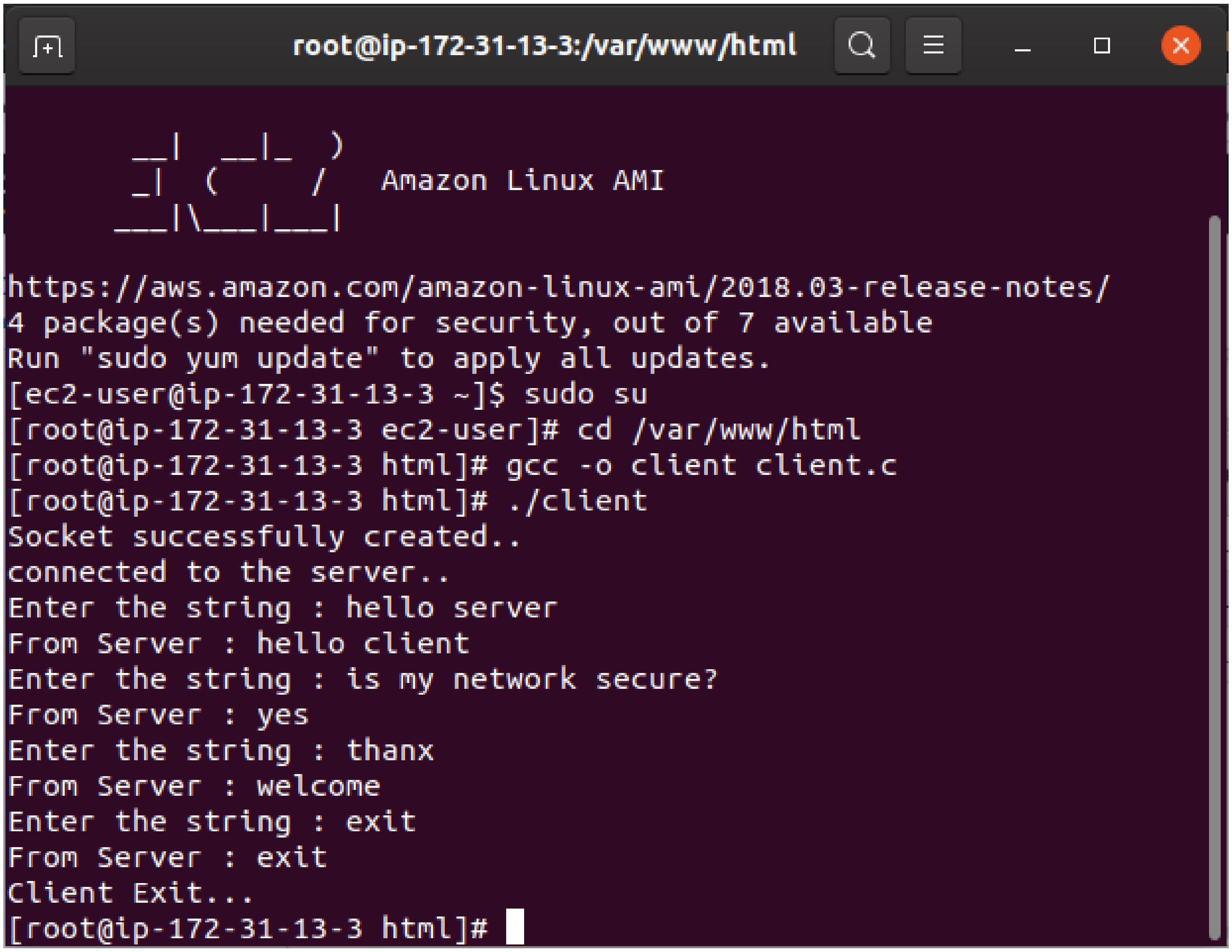
Open second terminal for server side and run following commands

- ✓ Server Terminal
  - cd Downloads/SSH
  - ssh ec2-user@52.66.245.134 -i network_assignment.pem
  - cd /var/www/html
  - gcc -o server server.c
  - ./server



```
      _|  (     /   Amazon Linux AMI
     ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-13-3 ~]$ sudo su
[root@ip-172-31-13-3 ec2-user]# cd /var/www/html
[root@ip-172-31-13-3 html]# gcc -o server server.c
[root@ip-172-31-13-3 html]# ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: hello server
         To client : hello client
From client: is my network secure?
         To client : yes
From client: thanx
         To client : welcome
From client: exit
         To client : exit
Server Exit...
[root@ip-172-31-13-3 html]#
```

- ✓ Client Terminal
    - gcc -o client client.c
    - ./client



```
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-13-3 ~]$ sudo su
[root@ip-172-31-13-3 ec2-user]# cd /var/www/html
[root@ip-172-31-13-3 html]# gcc -o client client.c
[root@ip-172-31-13-3 html]# ./client
Socket successfully created..
connected to the server..
Enter the string : hello server
From Server : hello client
Enter the string : is my network secure?
From Server : yes
Enter the string : thanx
From Server : welcome
Enter the string : exit
From Server : exit
Client Exit...
[root@ip-172-31-13-3 html]# 
```

Source Code

Client Side Code:-

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
        char buff[MAX];
        int n;
        for (; ; ) {
                bzero(buff, MAX);

                read(sockfd, buff, sizeof(buff));
                printf(“From client: %s\t To client : “, buff);
                bzero(buff, MAX);
                n = 0;
                while ((buff[n++] = getchar()) != ‘\n’);

                write(sockfd, buff, sizeof(buff));

                if (strncmp(“exit”, buff, 4) == 0) {
                        printf(“Server Exit…\n”);
                        break;
                }
        }
}
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;

        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf(“socket creation failed…\n”);
                exit(0);
        }
        else
                printf(“Socket successfully created..\n”);
        bzero(&servaddr, sizeof(servaddr));

        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);

        if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
                printf(“socket bind failed…\n”);
                exit(0);
        }
        else
                printf(“Socket successfully binded..\n”);

        if ((listen(sockfd, 5)) != 0) {
                printf(“Listen failed…\n”);
                exit(0);
        }
        else
                printf(“Server listening..\n”);
        len = sizeof(cli);

        connfd = accept(sockfd, (SA*)&cli, &len);
        if (connfd < 0) {
                printf(“server    ccept failed…\n”);
                exit(0);
        }
        else
                printf(“server    ccept the client…\n”);

        func(connfd);

        close(sockfd);
}
```

Server Side Code:-

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
        char buff[MAX];
        int n;
        for (; ; ) {
                bzero(buff, sizeof(buff));
                printf(“Enter the string : “);
                n = 0;
                while ((buff[n++] = getchar()) != ‘\n’)
                        ;
                write(sockfd, buff, sizeof(buff));
                bzero(buff, sizeof(buff));
                read(sockfd, buff, sizeof(buff));
                printf(“From Server : %s”, buff);
                if ((strncmp(buff, “exit”, 4)) == 0) {
                        printf(“Client Exit…\n”);
                        break;
                }
        }
}

int main()
{
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;

        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf(“socket creation failed…\n”);
                exit(0);
        }
        else
                printf(“Socket successfully created..\n”);
        bzero(&servaddr, sizeof(servaddr));

        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr(“127.0.0.1”);
        servaddr.sin_port = htons(PORT);

        if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
                printf(“connection with the server failed…\n”);
                exit(0);
        }
        else
                printf(“connected to the server..\n”);

        func(sockfd);

        close(sockfd);
}
```

<<<END>>>