Installing and configuring SonarQube with an Nginx reverse proxy on an AWS EC2 Ubuntu instance

SonarQube is an open-source platform used for continuous inspection of code quality to perform static code analysis and identify bugs, vulnerabilities, and code smells. Nginx can be configured as a reverse proxy to provide a public-facing URL for SonarQube, which typically runs on a non-standard port.

Nginx is an open-source, high-performance web server that functions as a reverse proxy, load balancer, and content cache. It is designed to handle large volumes of traffic and many simultaneous connections, serving static content, improving application performance by caching and managing requests, and enhancing security by acting as a barrier between clients and backend servers

Steps to follow:

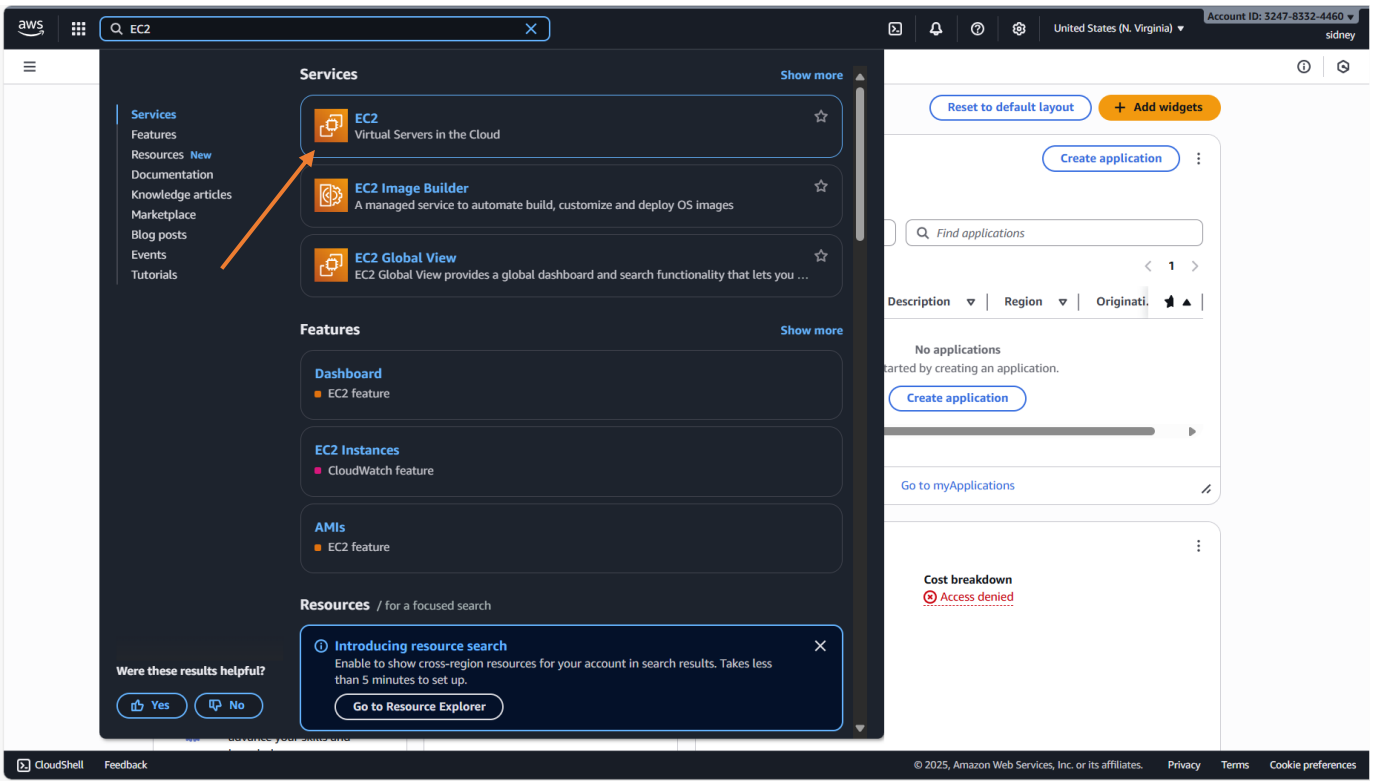
1. **Prepare the AWS EC2 Ubuntu Instance:**
    - Launch an EC2 instance with an Ubuntu Server AMI (e.g., Ubuntu Server 22.04 LTS).
    - Ensure the instance type meets SonarQube's memory requirements (at least 2GB RAM for small instances).
    - Configure the security group to allow inbound traffic on port 22 (SSH), 9000 (SonarQube direct access, if needed for initial setup), 80 (HTTP), and 443 (HTTPS for Nginx).
    - Connect to the instance via SSH.
2. **Install Prerequisites on Ubuntu:**
    - Update package lists.
    - Install Java (OpenJDK 17 or higher is recommended for recent SonarQube versions):
    - Install PostgreSQL (SonarQube requires a database).
    - Create a PostgreSQL user and database for SonarQube:
    - Modify Kernel System Limits for SonarQube.
        - Increase file descriptor limits.
3. **Install SonarQube:**
    - Create a dedicated user for SonarQube.
    - Download the latest SonarQube LTS version from the official website.
    - Configure SonarQube to connect to PostgreSQL.
    - Create a systemd service for SonarQube.
    - Enable and start SonarQube.
4. **Configure Nginx Reverse Proxy:**
    - Install Nginx.
    - Add the DNS records
    - Create a new Nginx configuration file for SonarQube
    - Enable the Nginx site and restart Nginx
    - Make sure your configuration file has no syntax errors
    - Enable the Nginx site and restart Nginx
5. **Secure with SSL using Certbot:**
    - Install Certbot
    - Obtain and install an SSL certificate on the subdomain
    - Verify if Certificate deployed successfully

STEP 1: Prepare the AWS EC2 Ubuntu Instance

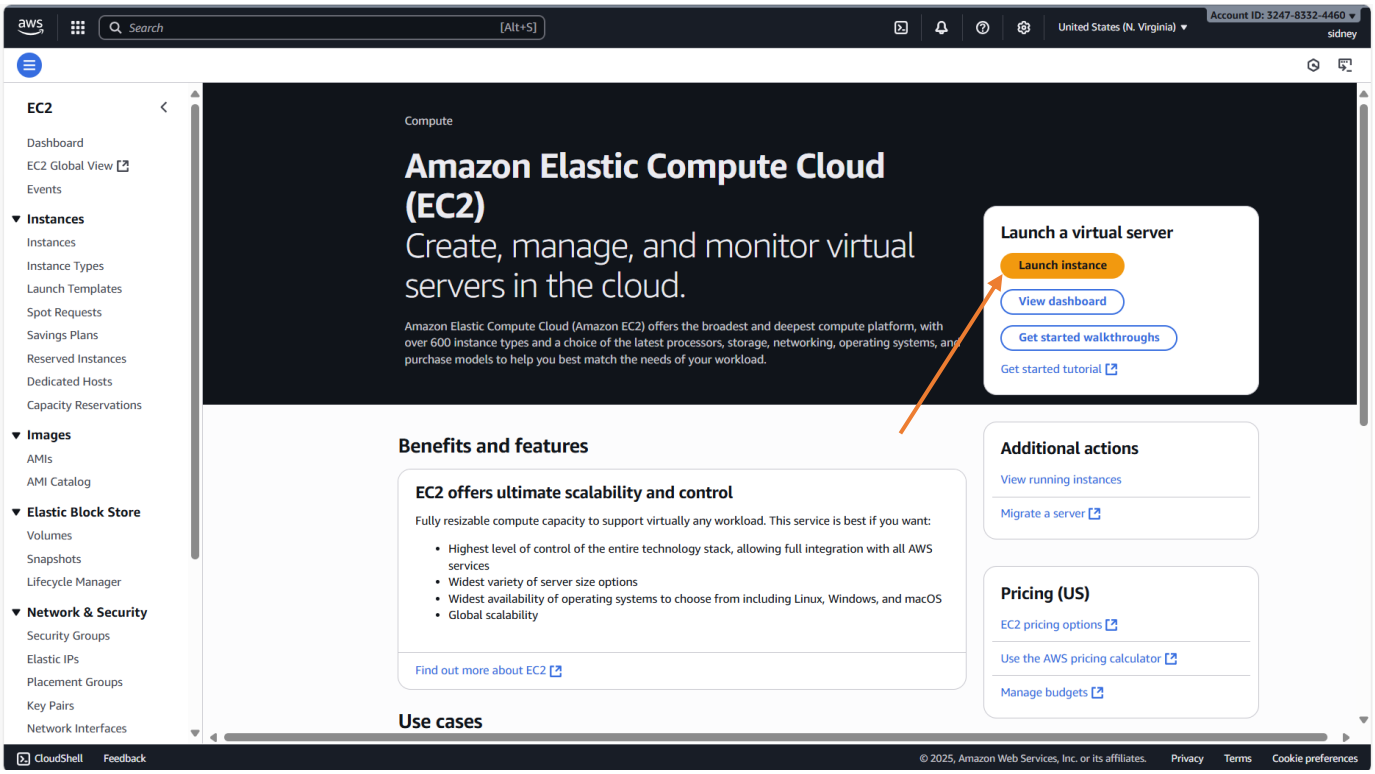
We will launch an ubuntu EC2 instance, enable ports 22 for SSH, port 9000 for SonarQube, port 80 for HTTP and port 43 for HTTPS and later on SSH connect to it to install our dependencies.

PART 1: Launch an Ubuntu EC2 instance

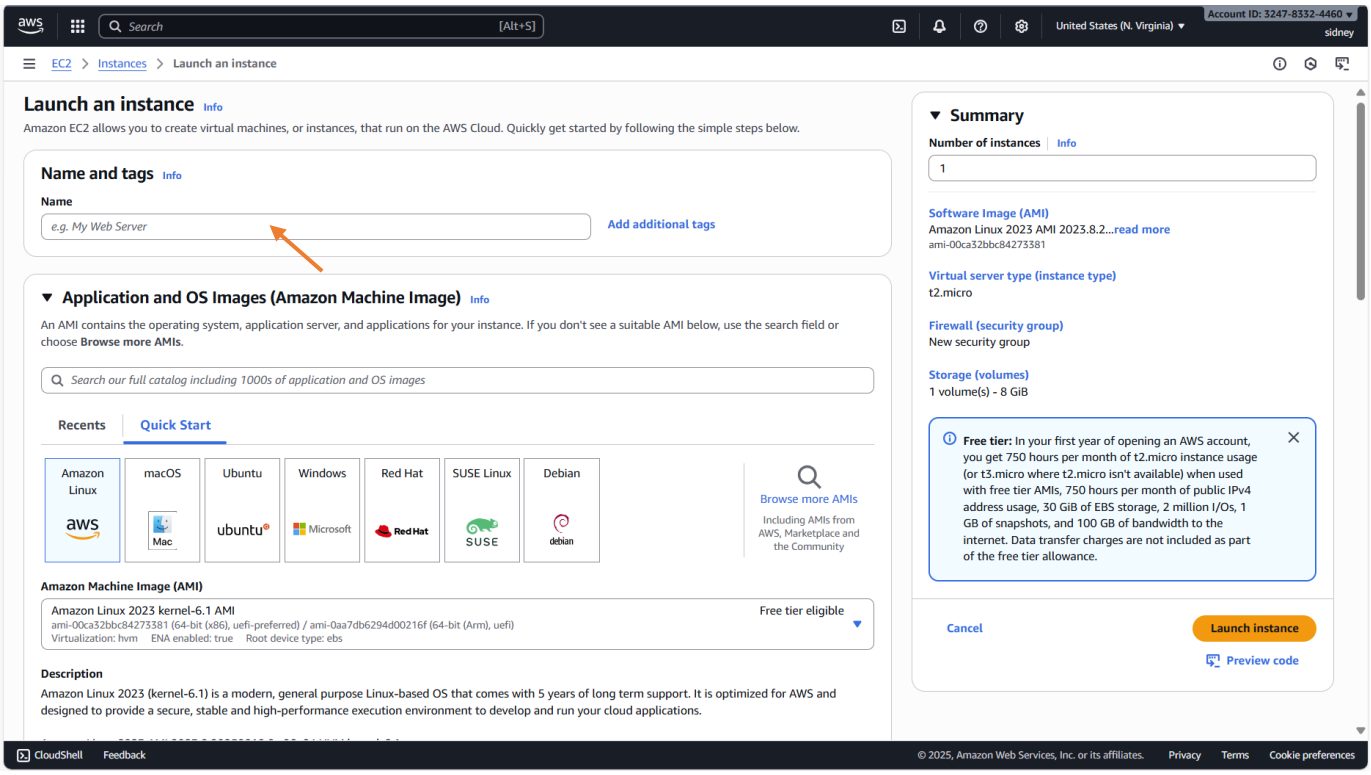
Go to AWS Management console and search for "**EC2**" Instance.



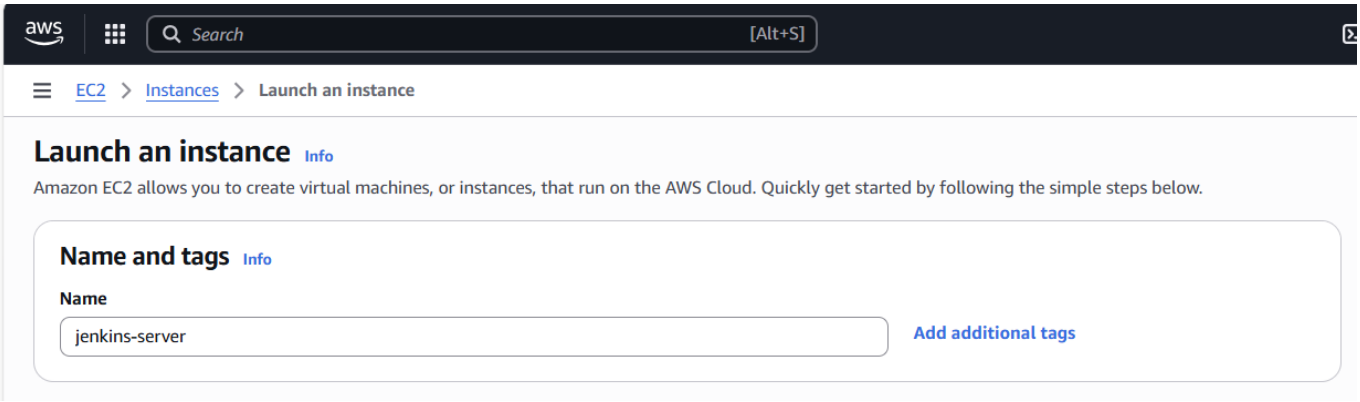
Click on "**EC2**"



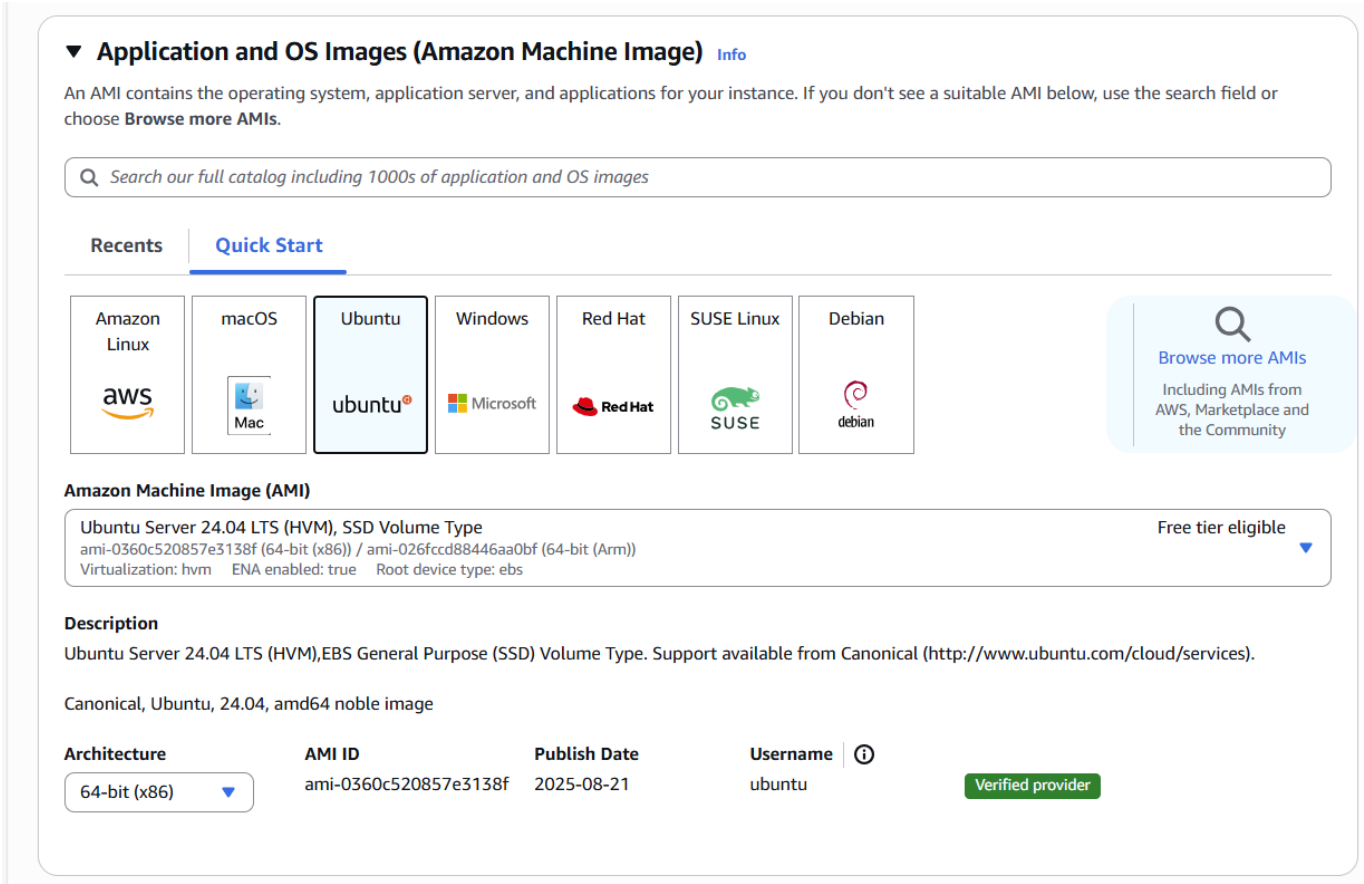
Click on “**Launch Instance**”



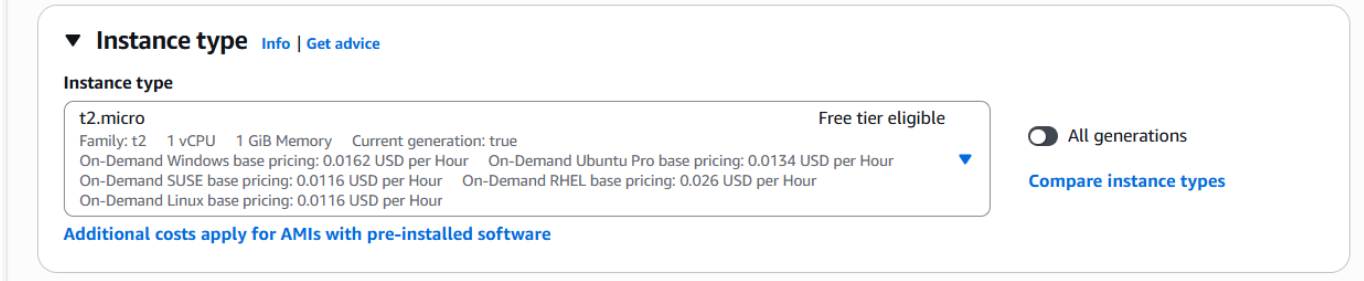
We will call the instance “**sonar-server**”



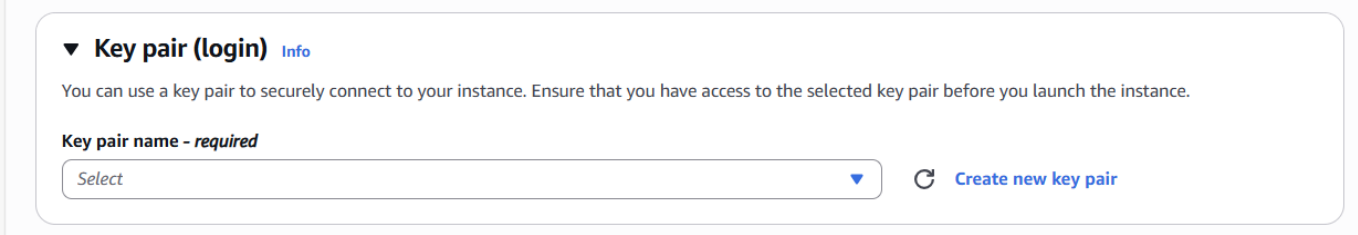
Scroll down to "**AMI**", select "**ubuntu**"



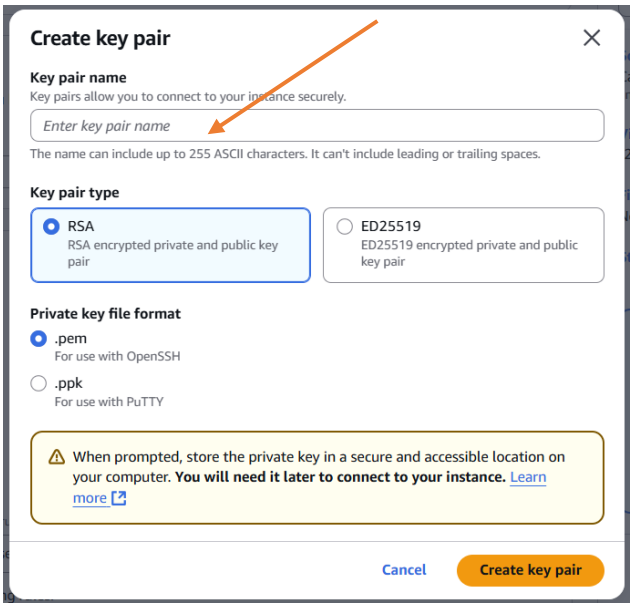
Scroll down to "**Instance Type**" and select "**t2.medium**"



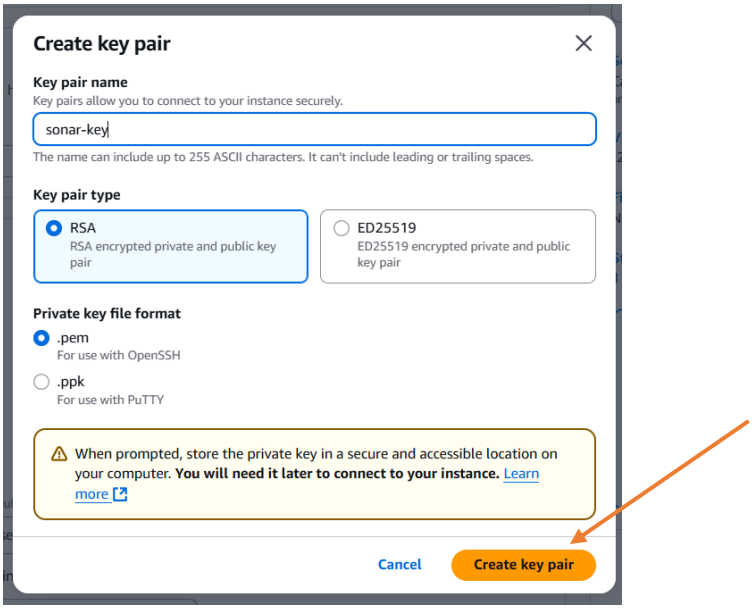
Scroll down to “**Key Pair**”



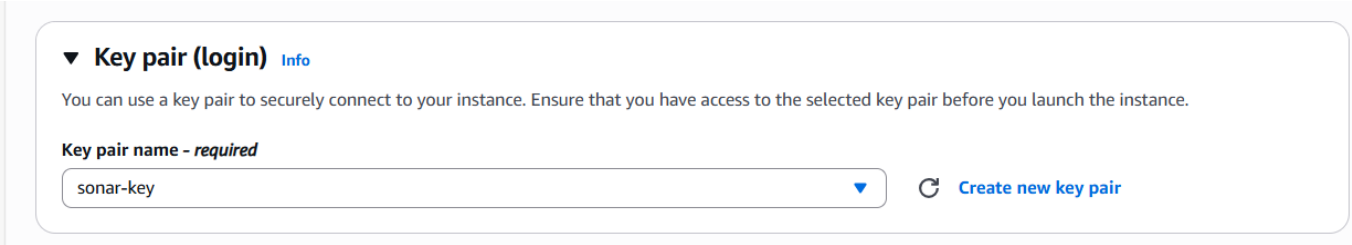
Click on “**Create new key pair**”



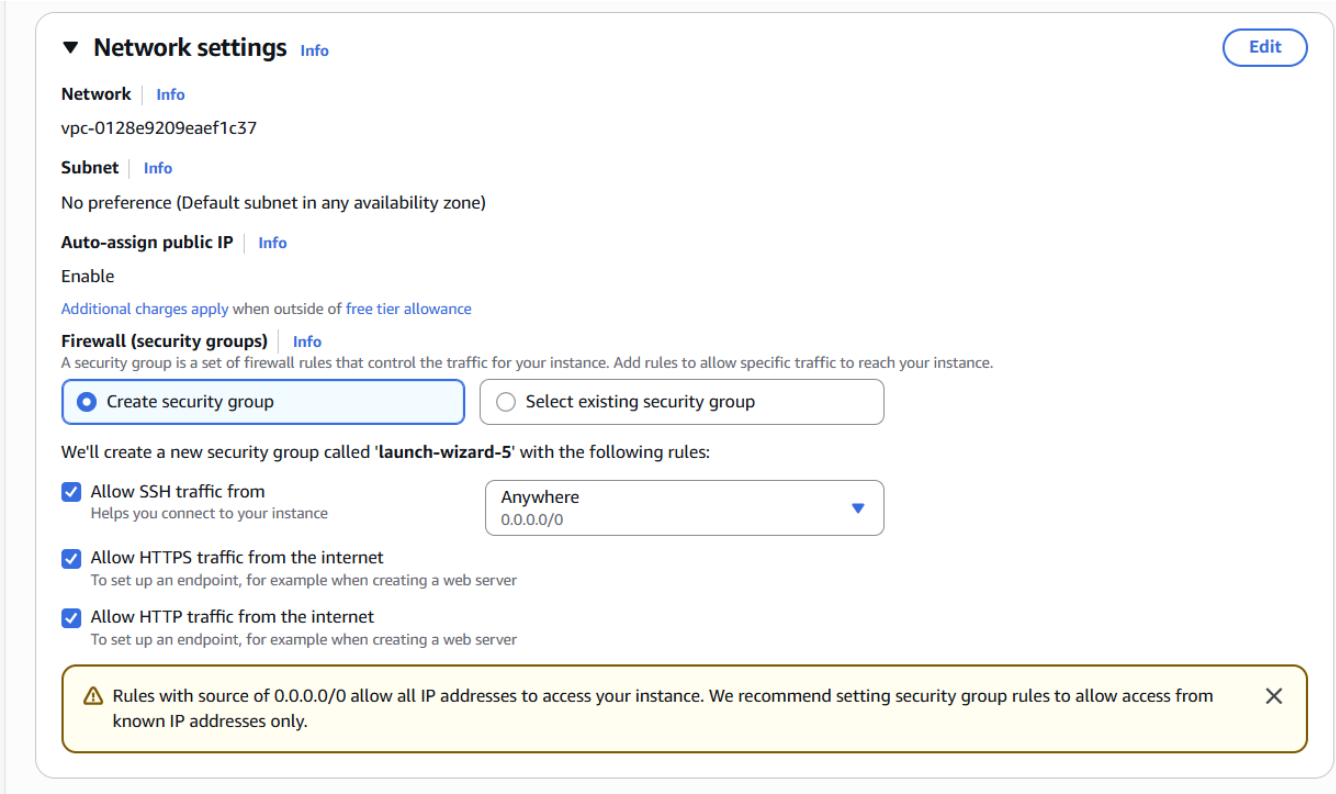
Give the key pair a name, I will call it “**sonar-key**”



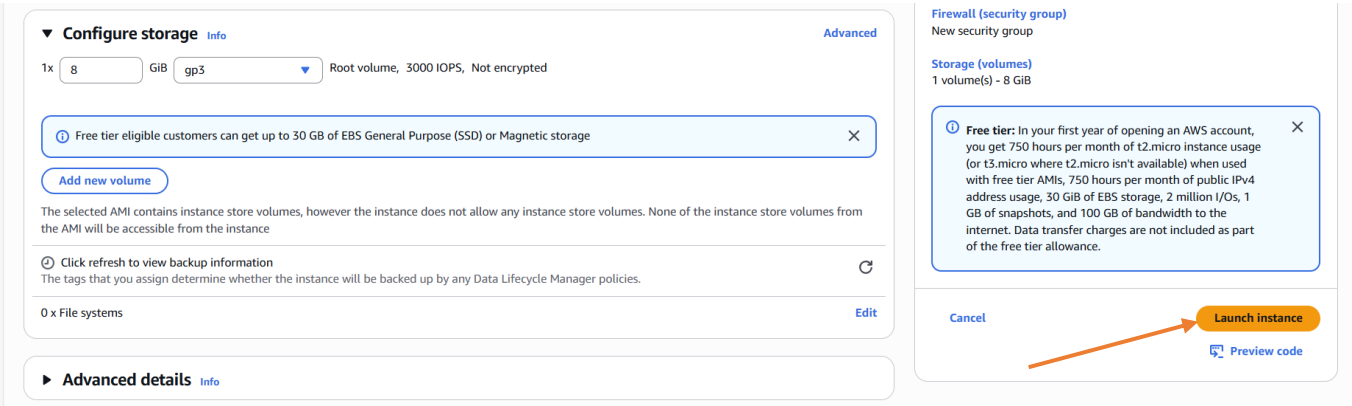
Click on “**Create key pair**”



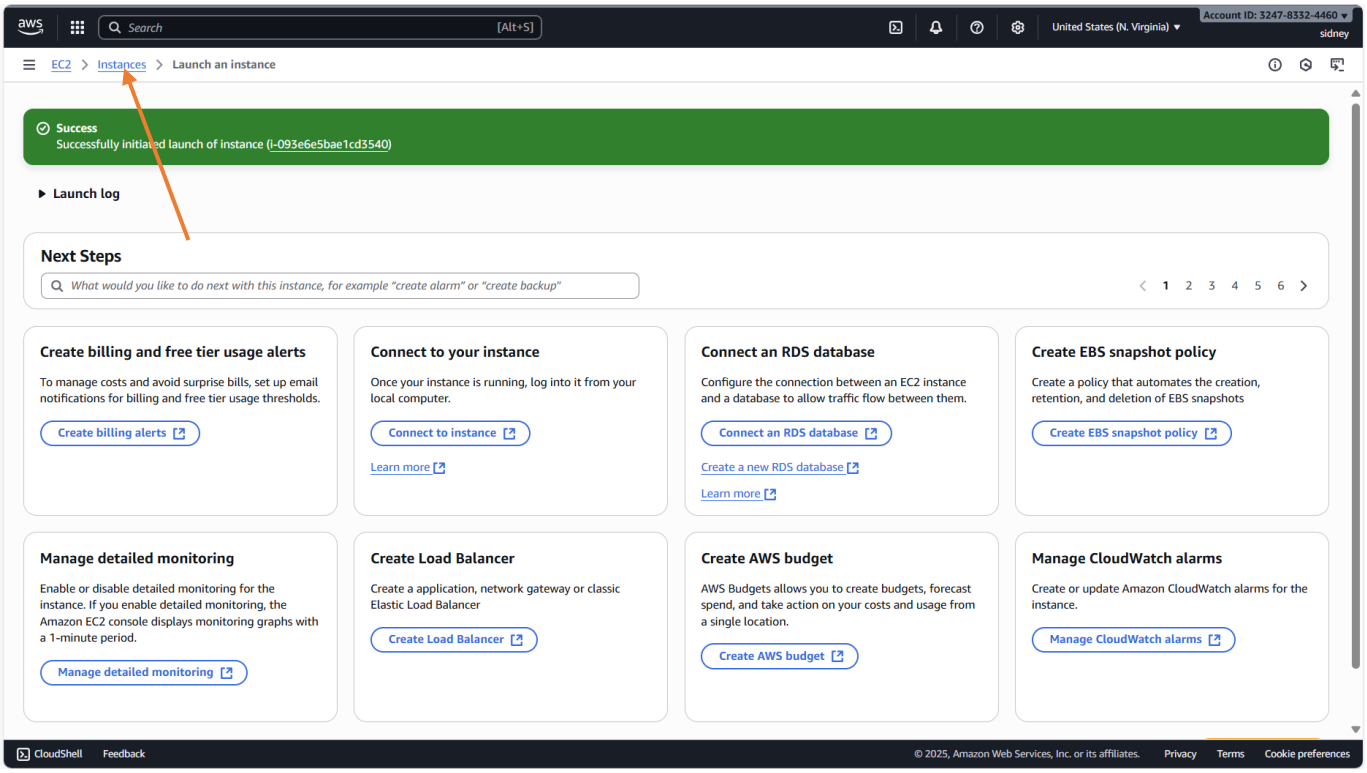
Scroll down to “**Network Settings**”. Select “**Allow HTTP traffic from the internet**”, “**Allow HTTP traffic from the internet** and “**Allow SSH traffic from**”.



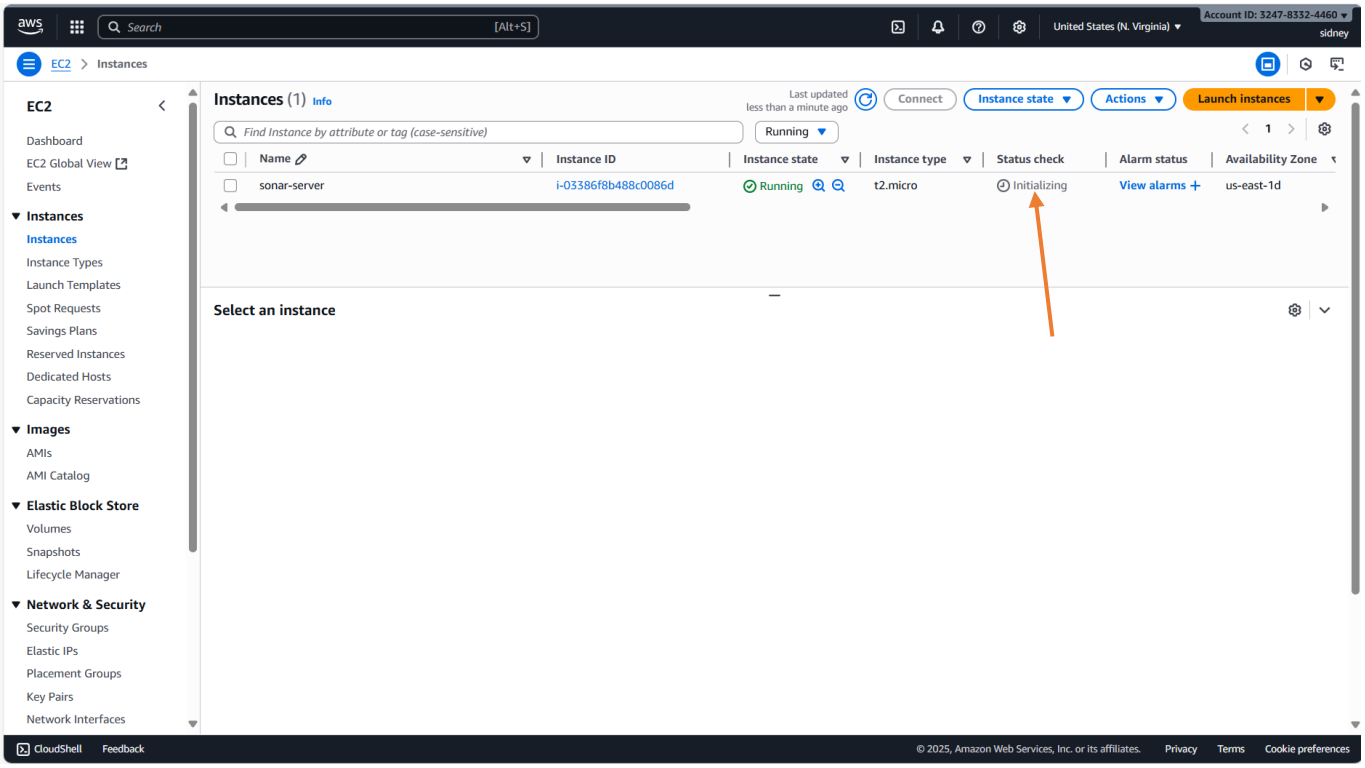
Scroll down to the end



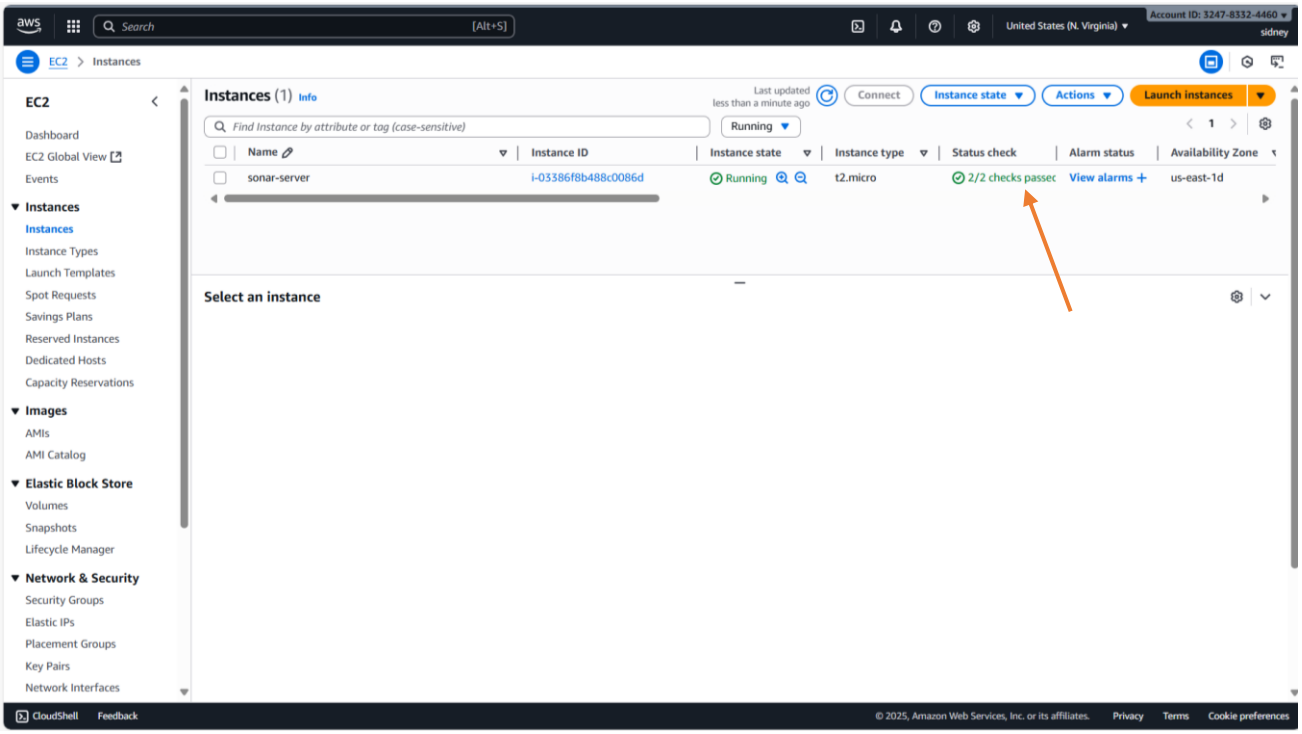
Click on “**Launch Instance**”



Click on "**Instances**"



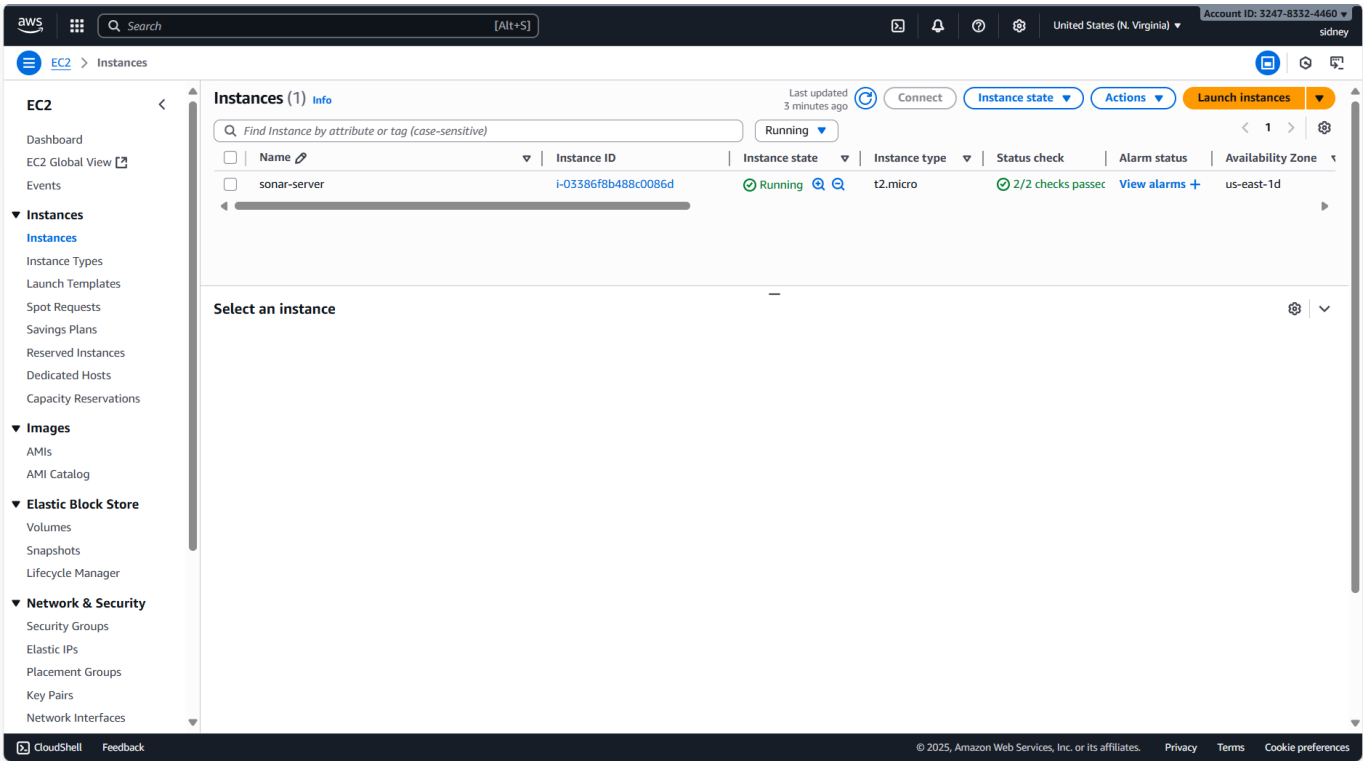
You can see that our just created instance is initializing. Wait for it to pass the "**2/2 check**"



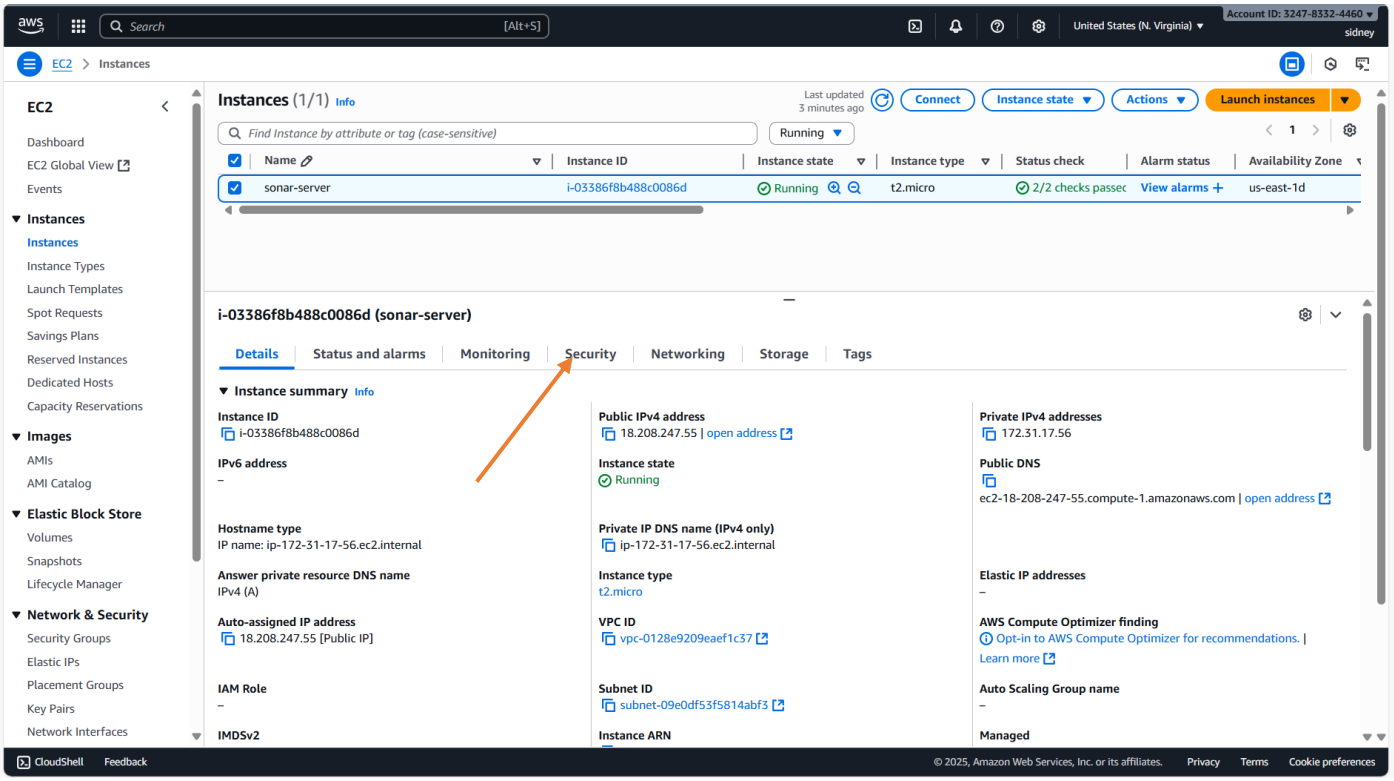
The instance has passed the "**2/2 check**"

PART 2: Add Port 9000 for SonarQube

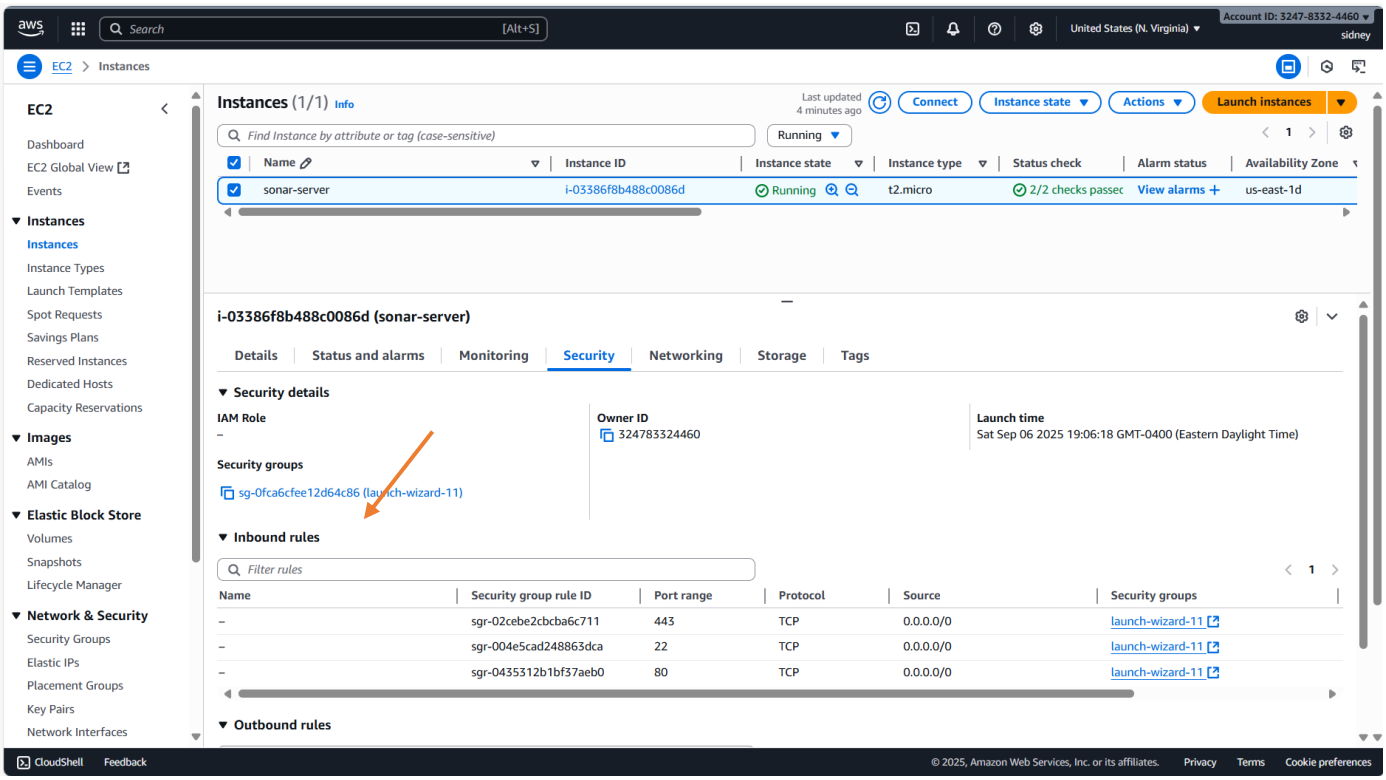
We will add port 9000 that will be used to access SonarQube through the browser. Now, we have to add port 9000 to enable access SonarQube from our browser. Go to our EC2 instance



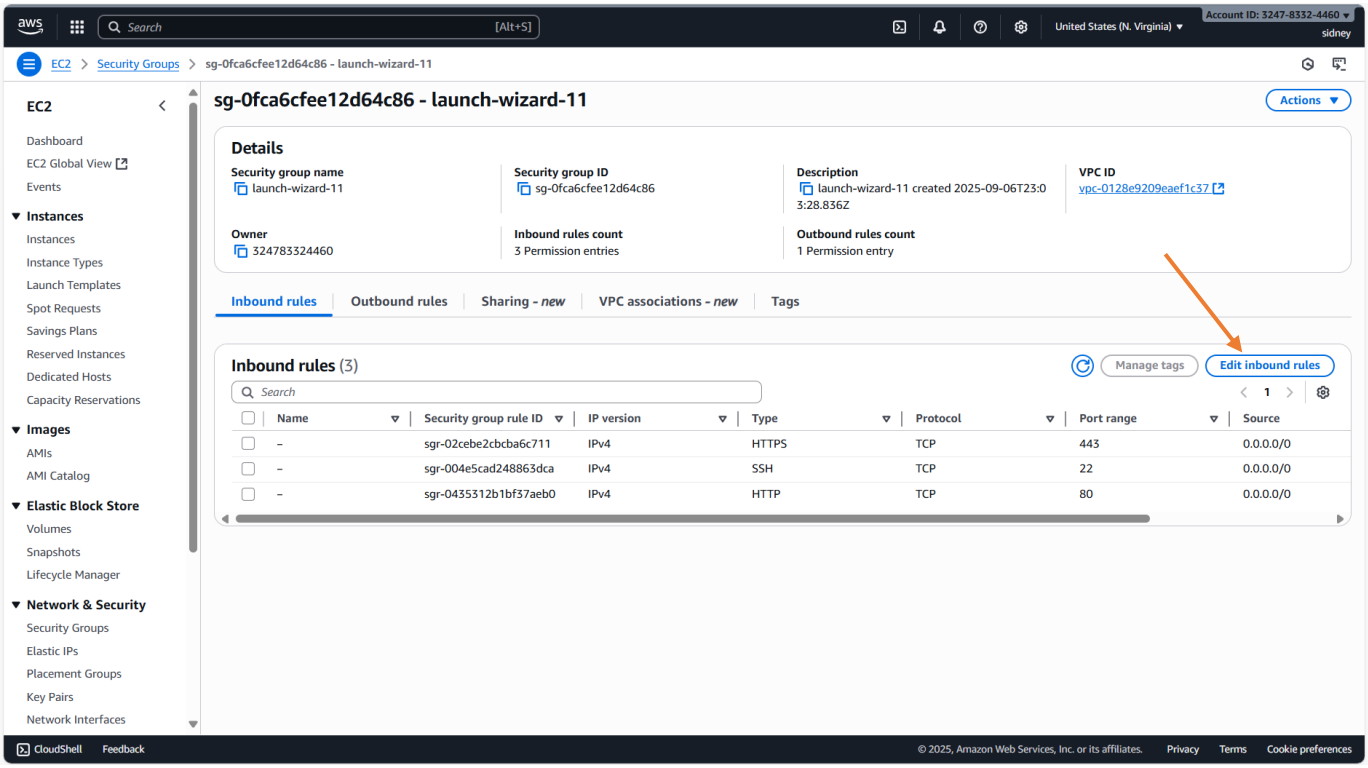
Select the EC2 instance



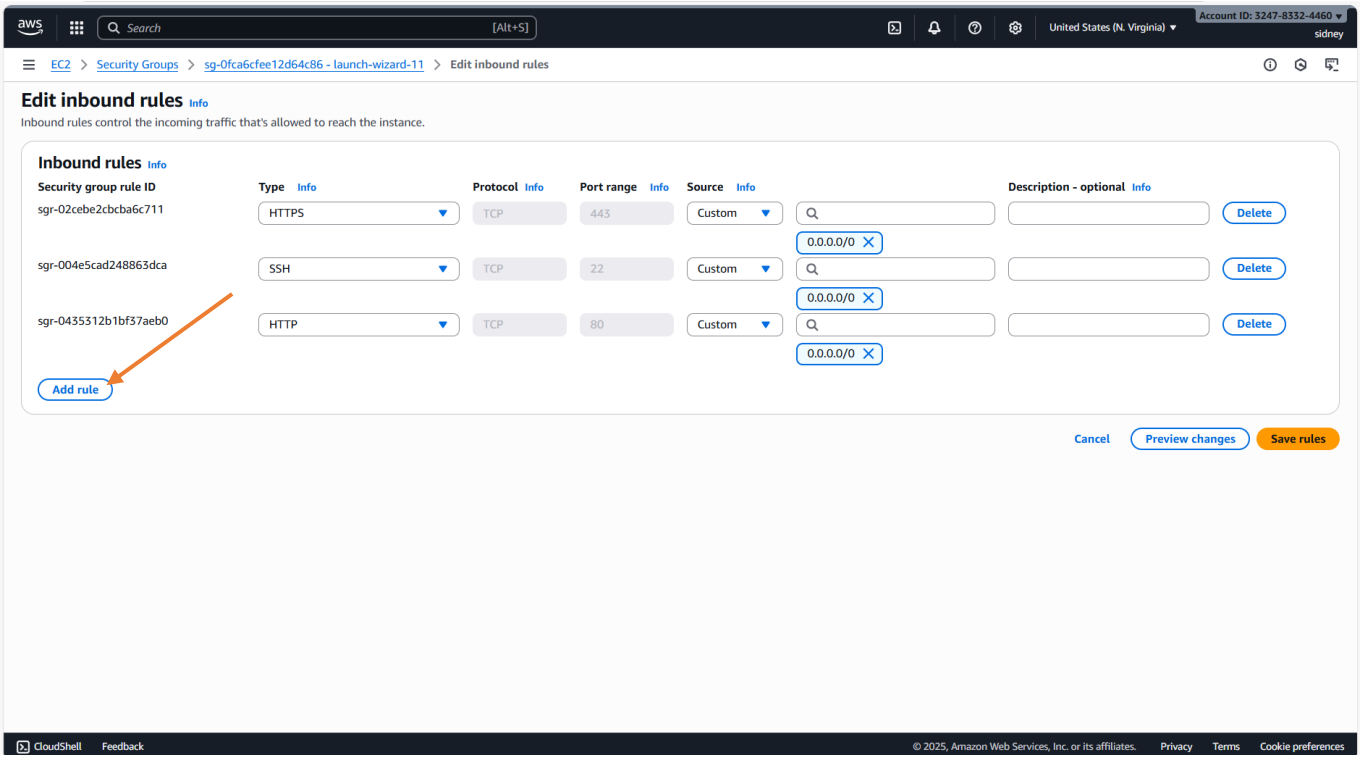
Click on the "**Security**" tab



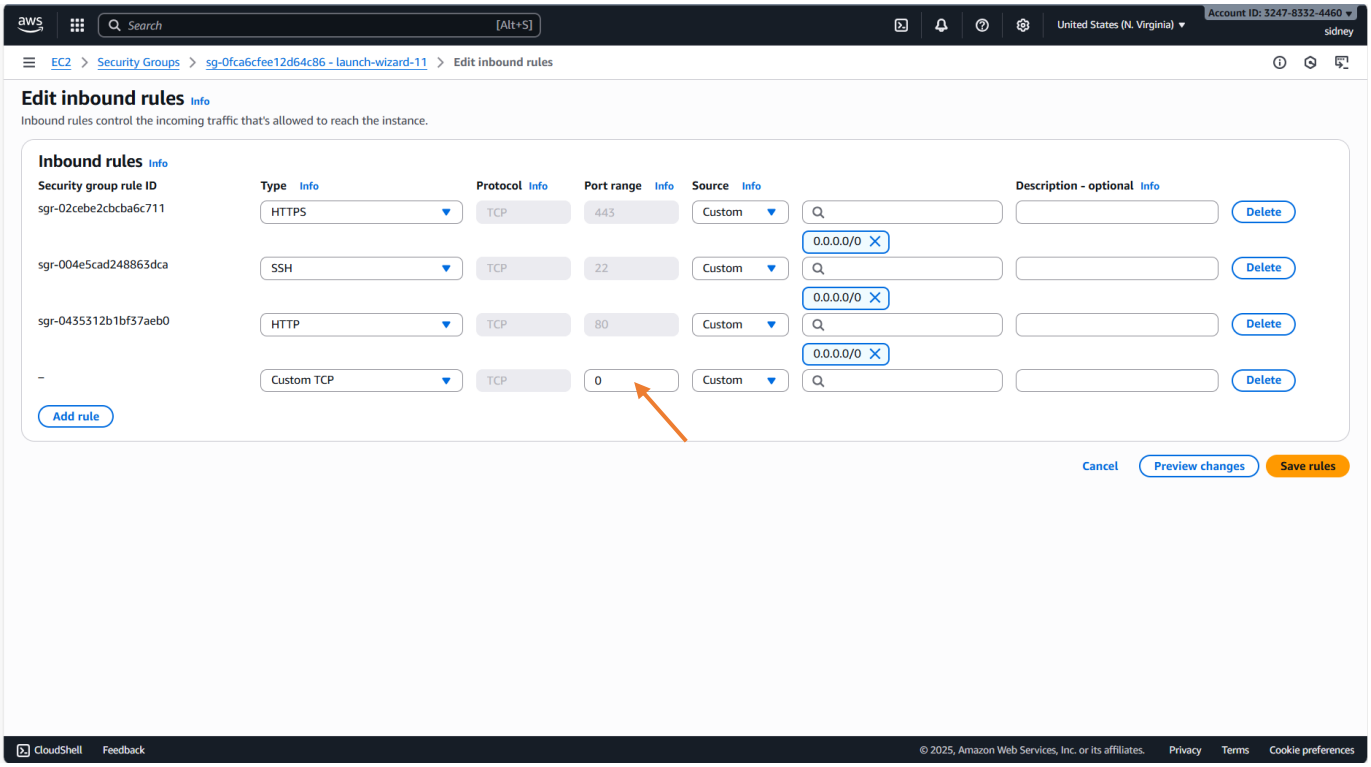
Click on the "**Security Group**" url



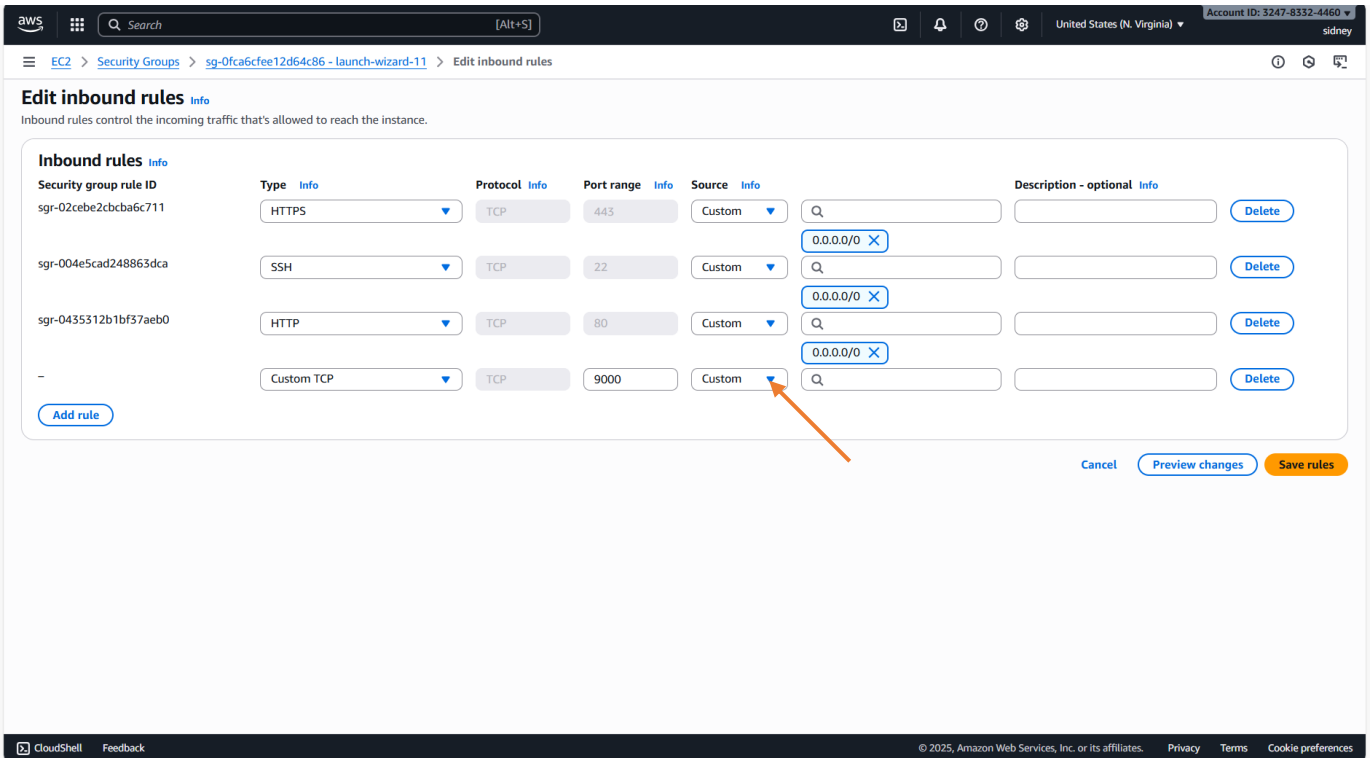
Click on "**Edit Inbound rules**"



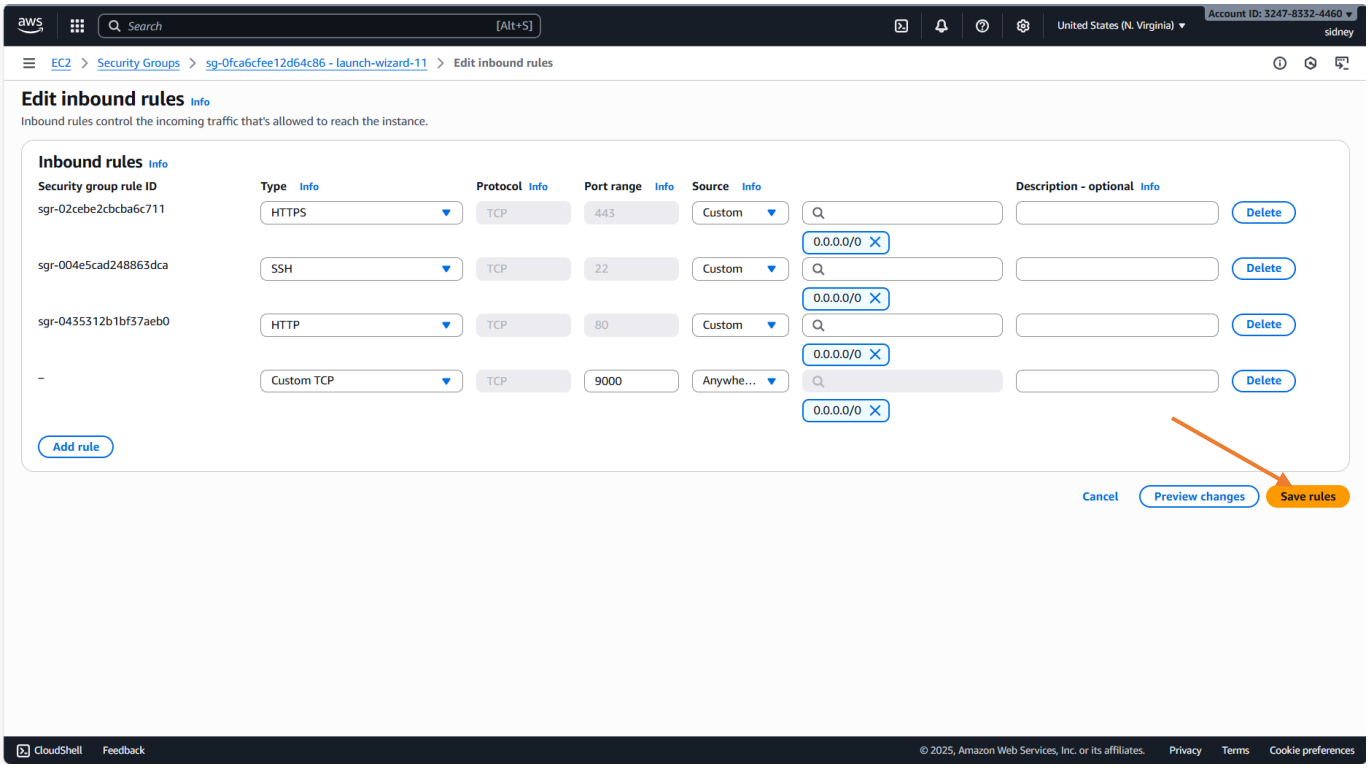
Click on "**Add Rule**"



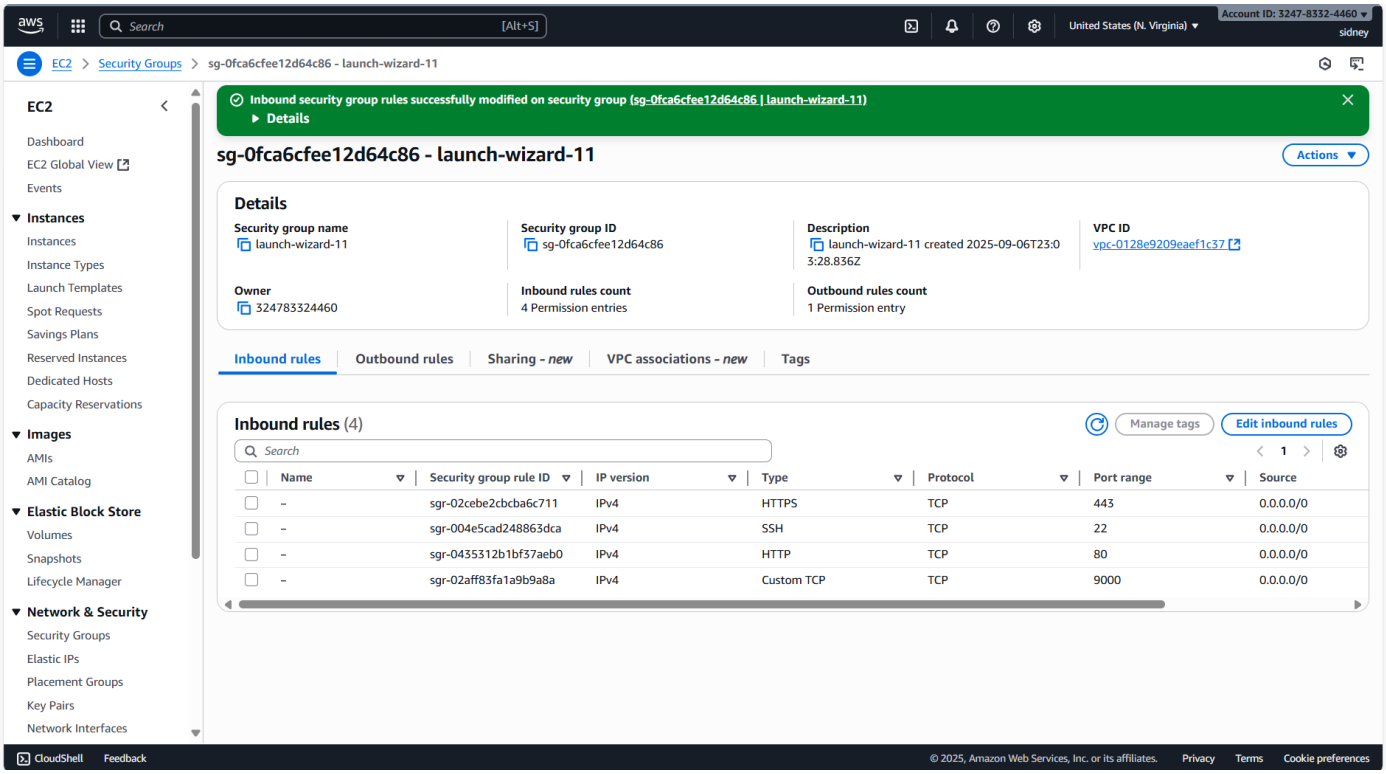
Click on the drop down on “**Port Range**” and enter the value “**9000**”



Click on the drop down on “**source**” and select “**Anywhere-IPv4**”



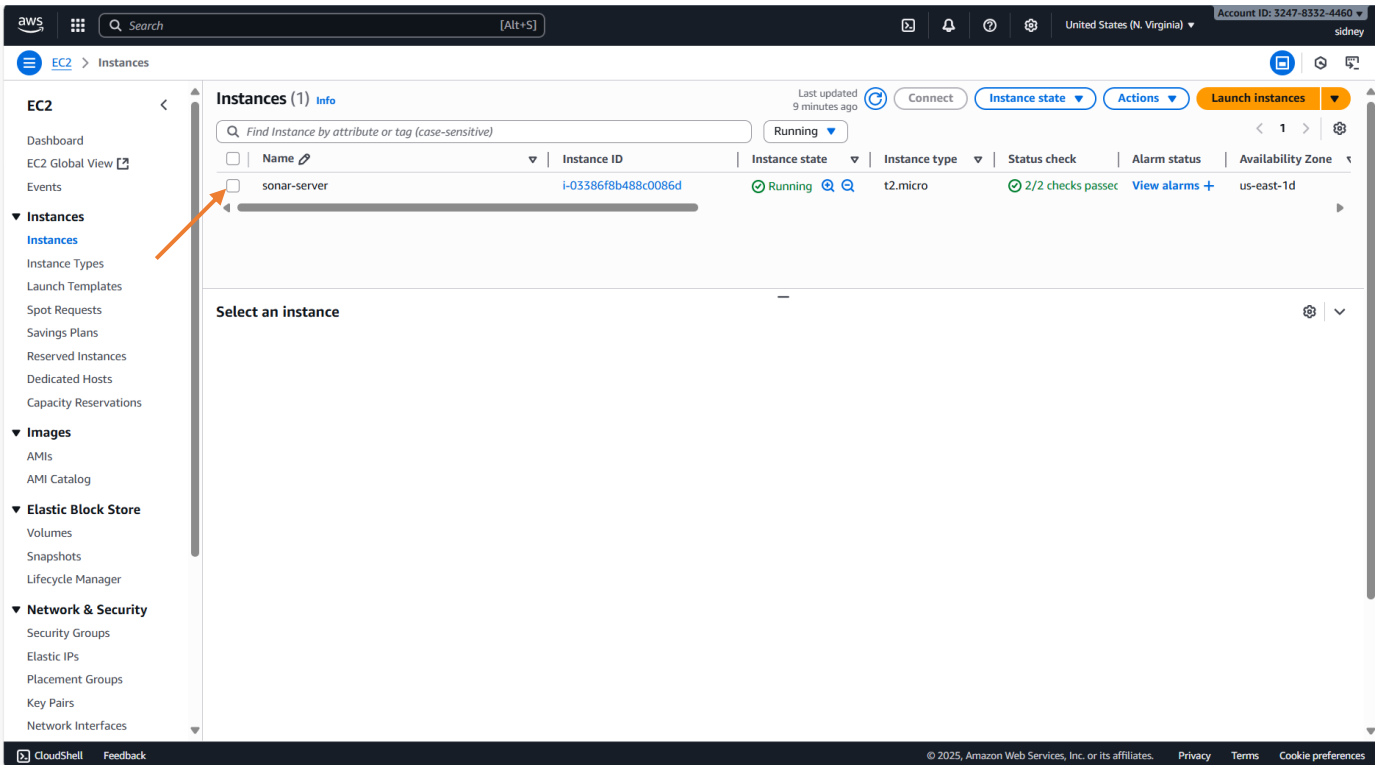
Click on “**Save rules**”



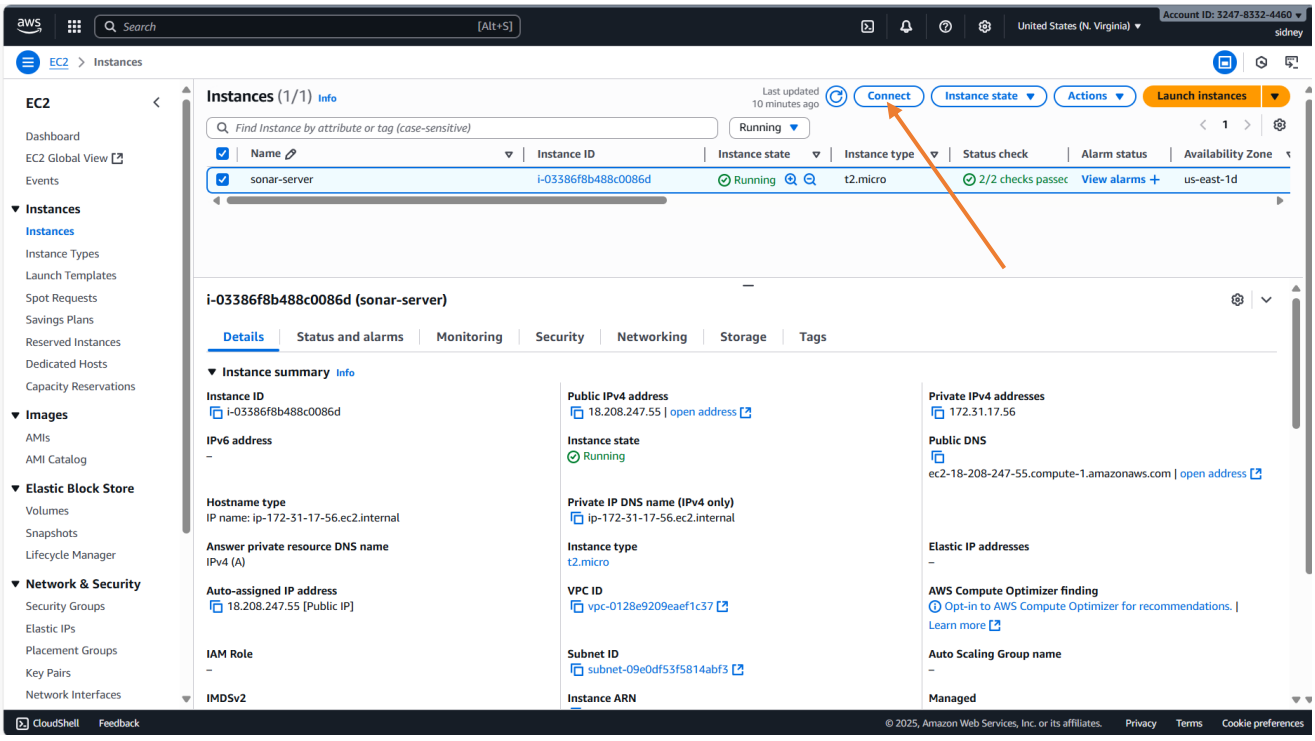
Port 9000 has been added.

PART 4: SSH Connect to the EC2 instance

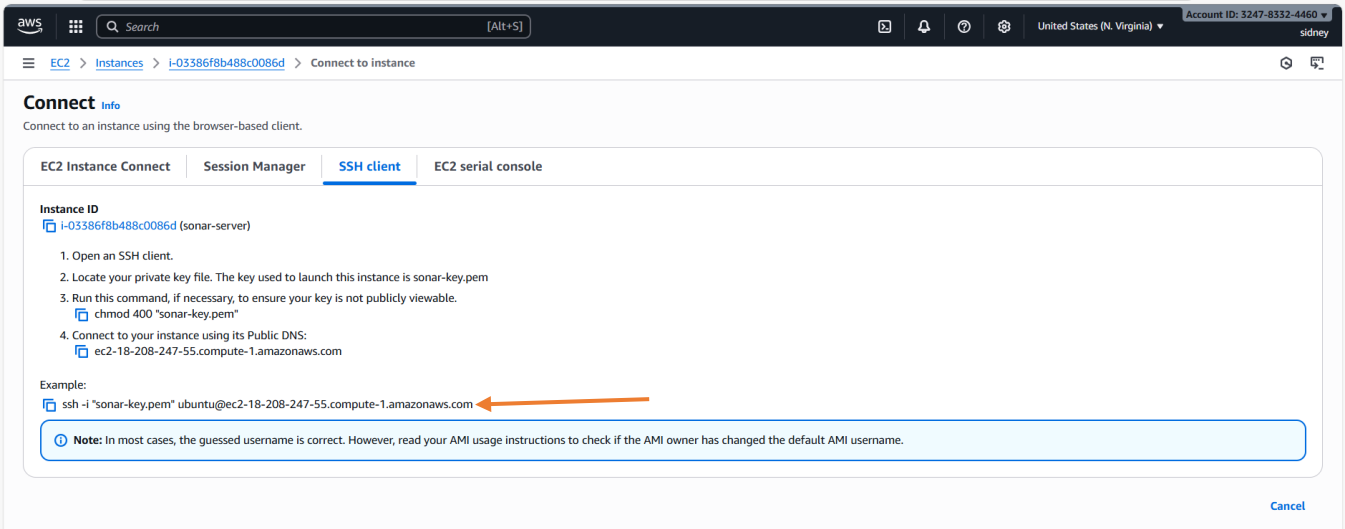
Now, let us connect to our EC2 instance through SSH



Select the instance

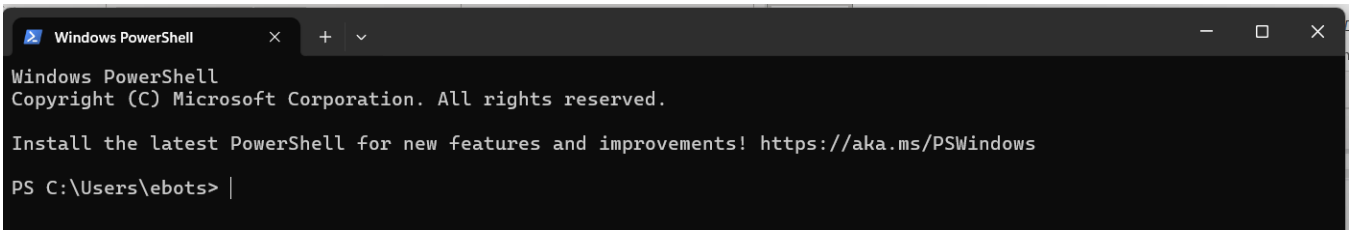


Click on “**Connect**”



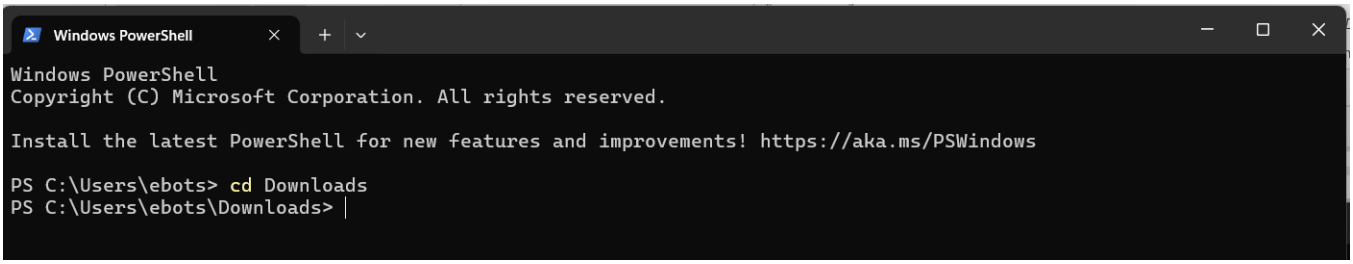
Copy this command and open PowerShell

ssh -i "sonar-key.pem" ubuntu@ec2-18-208-247-55.compute-1.amazonaws.com



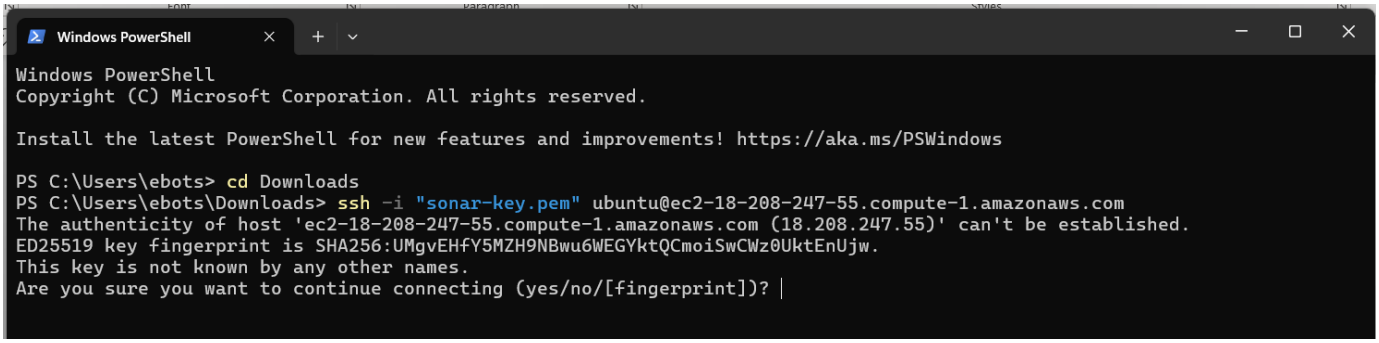
Navigate to the "**Downloads**" folder where our "**.pem**" file is stored by using the command:

cd Downloads

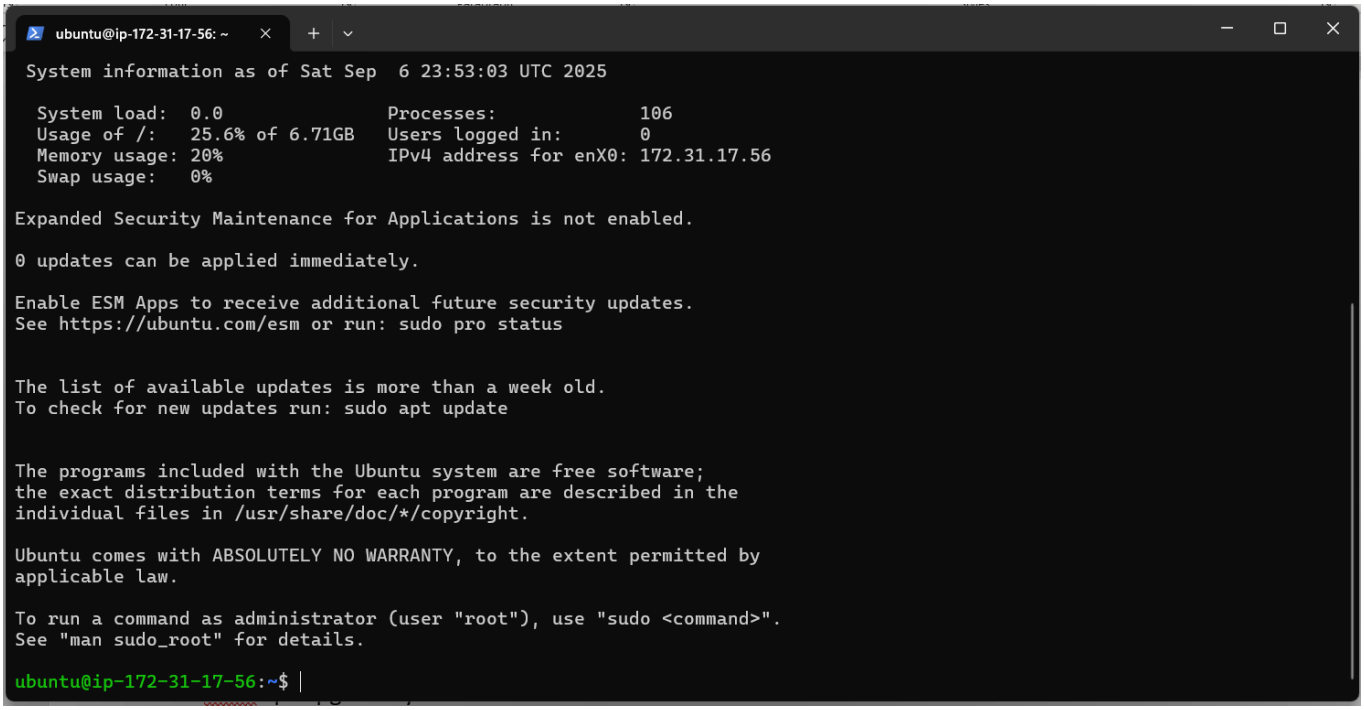


Then run the copied command:

ssh -i "sonar-key.pem" ubuntu@ec2-18-208-247-55.compute-1.amazonaws.com



Type “**yes**” and press ENTER



We have successfully SSH connected to our EC2 instance

Clear the terminal using the command:

```
clear
```

Grand Root user access using the command:

```
sudo su
```

STEP 2: Install Prerequisites on Ubuntu:

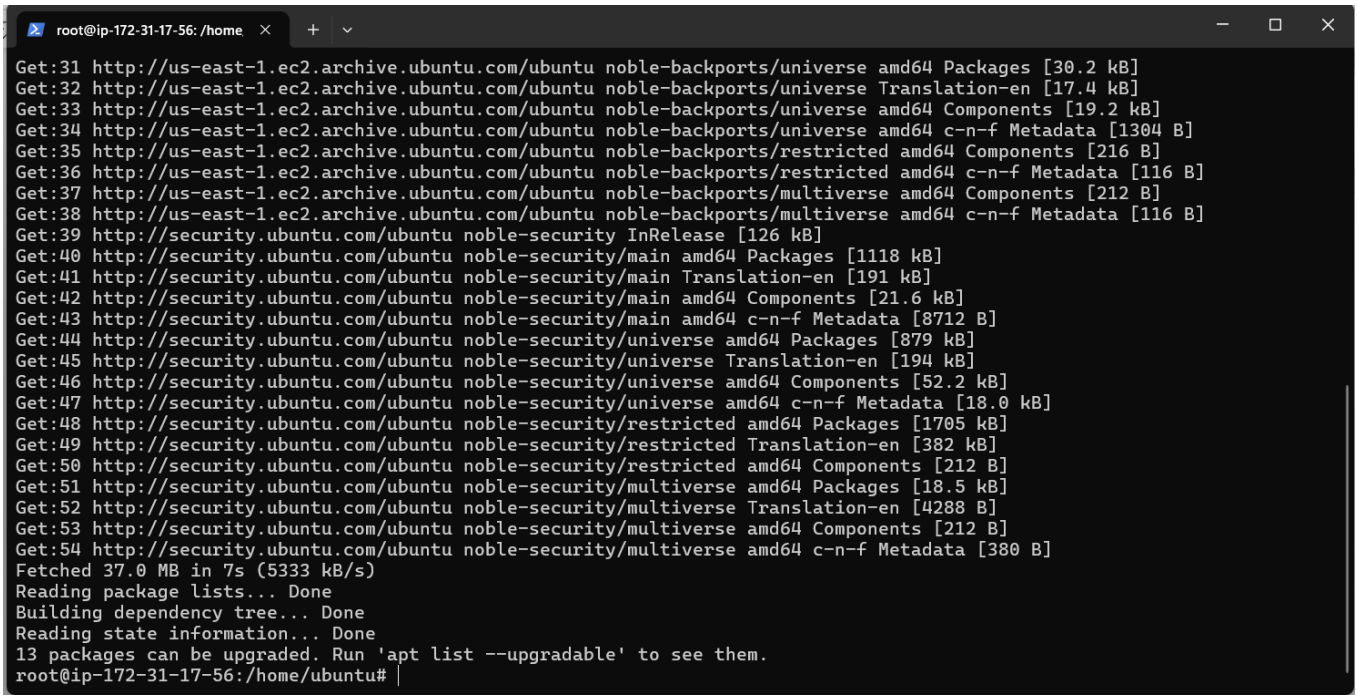
Here we will Update package lists, Install Java (OpenJDK 17 or higher is recommended for recent SonarQube versions), Install PostgreSQL (SonarQube requires a database), Create a PostgreSQL user and database for SonarQube, adjust kernel parameters for SonarQube, and Increase file descriptor limits.

Part 1: Update package lists

We will update the package and upgrade the package list here

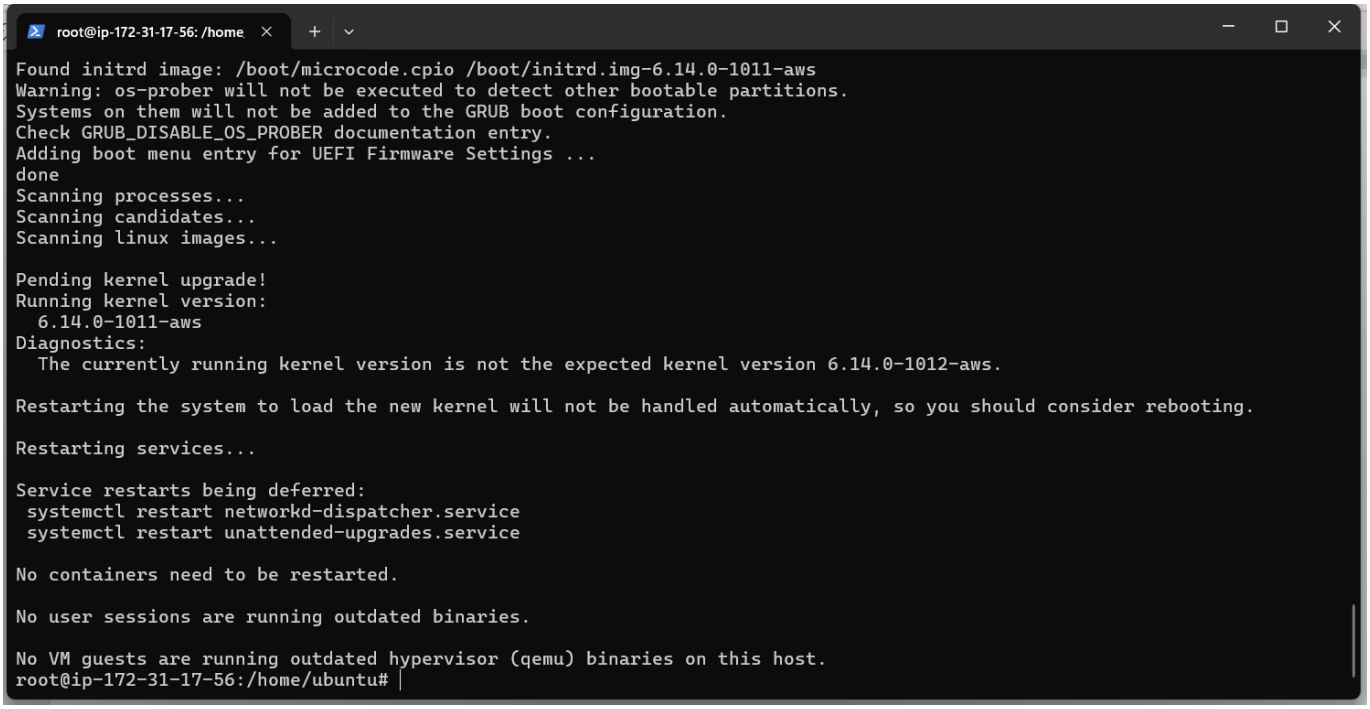
Update packages on server

```
sudo apt update
```



Update the server

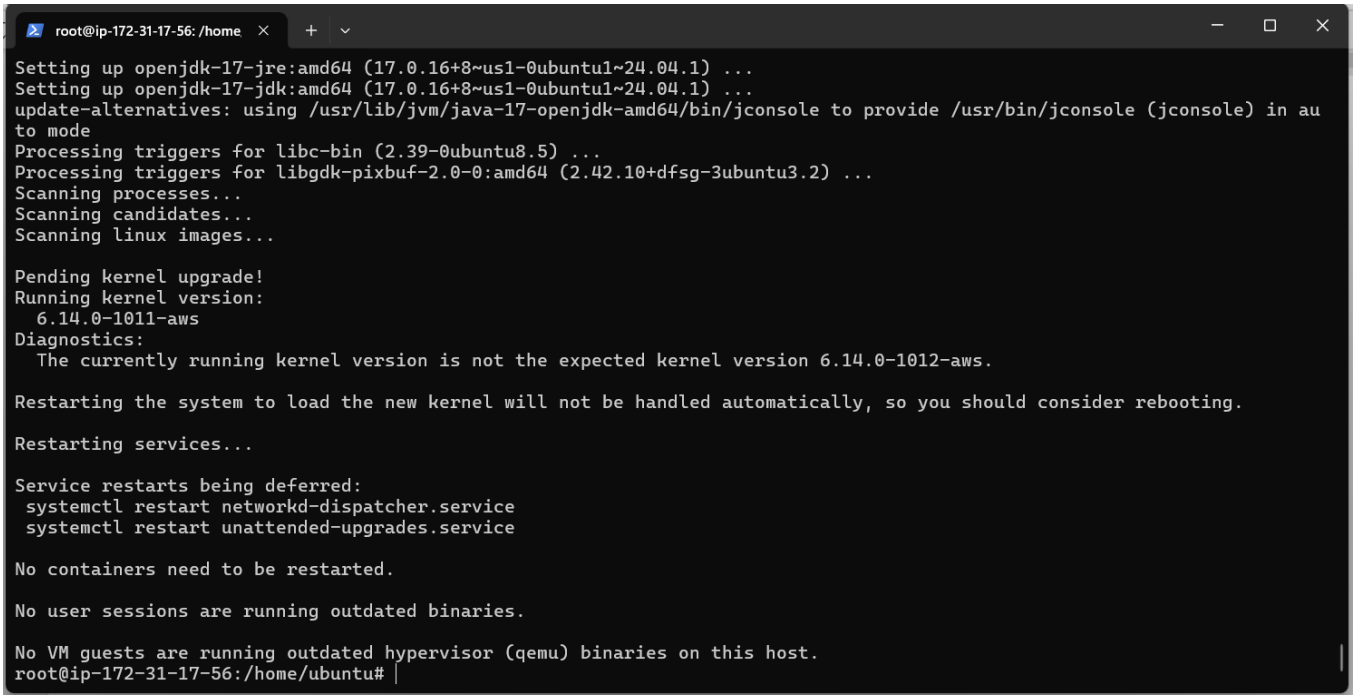
```
sudo apt upgrade -y
```



Part 2: Install OpenJDK 17

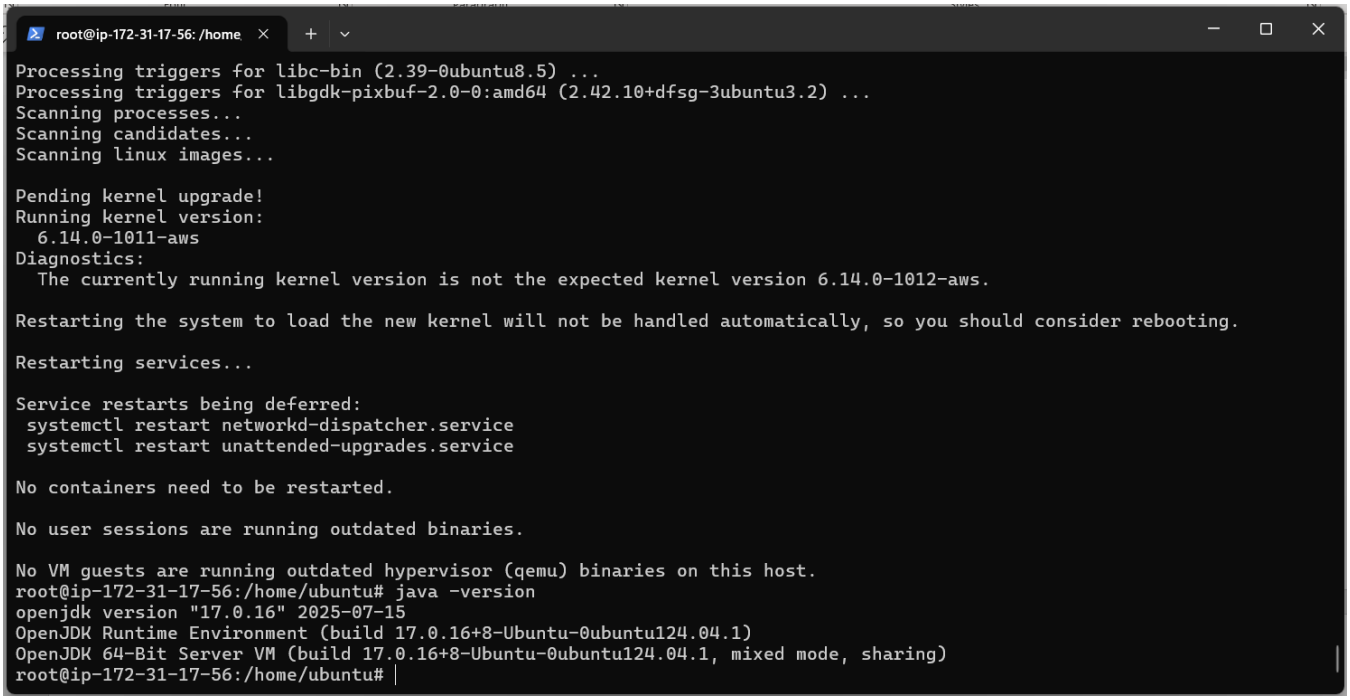
Install OpenJDK 17 (needed for the latest version of SonarQube (version 10.0).

```
sudo apt install -y openjdk-17-jdk
```



Let's check the installed version of Java. VALIDATION IS IMPORTANT.

```
java -version
```



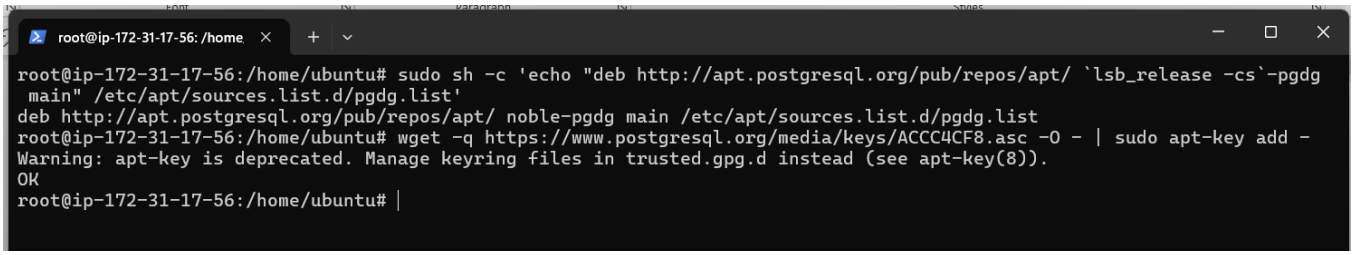
Part 3: Install and Configure PostgreSQL

Add the PostgreSQL repository

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" /etc/apt/sources.list.d/pgdg.list'
```

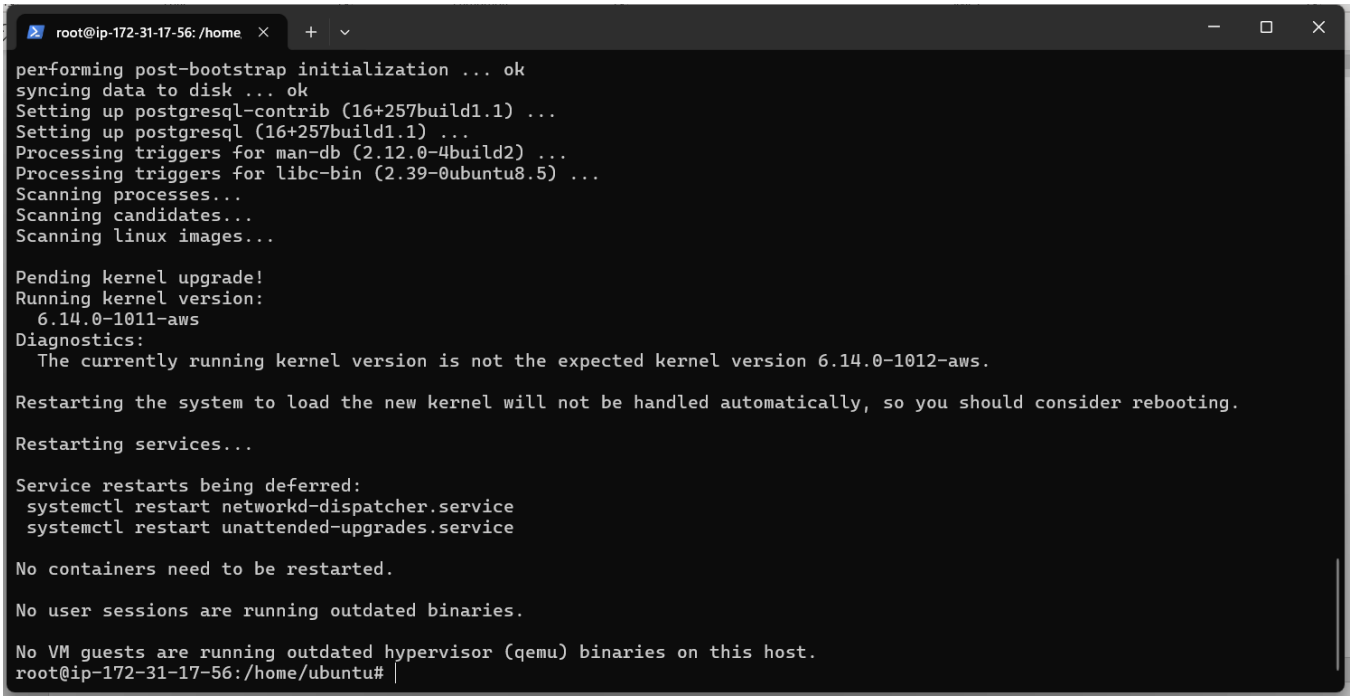
Add the PostgreSQL signing key

```
wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -
```



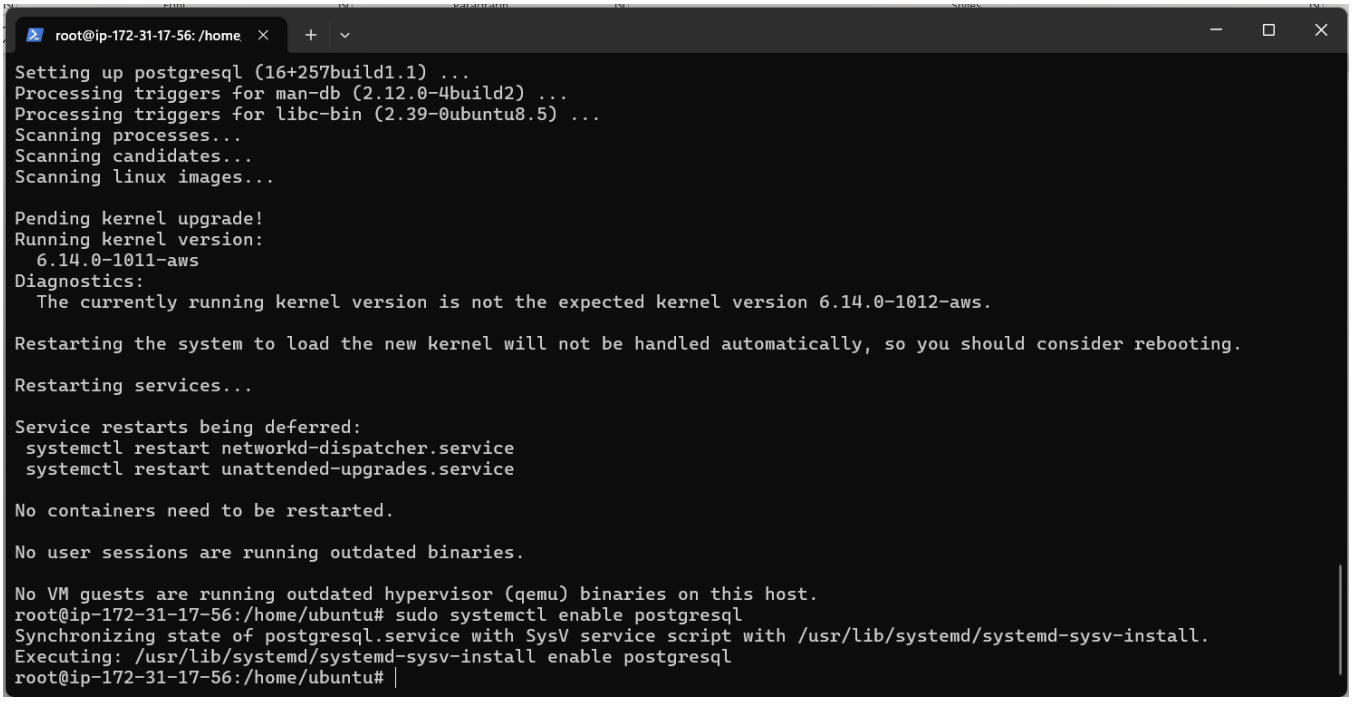
Install PostgreSQL

```
sudo apt install postgresql postgresql-contrib -y
```



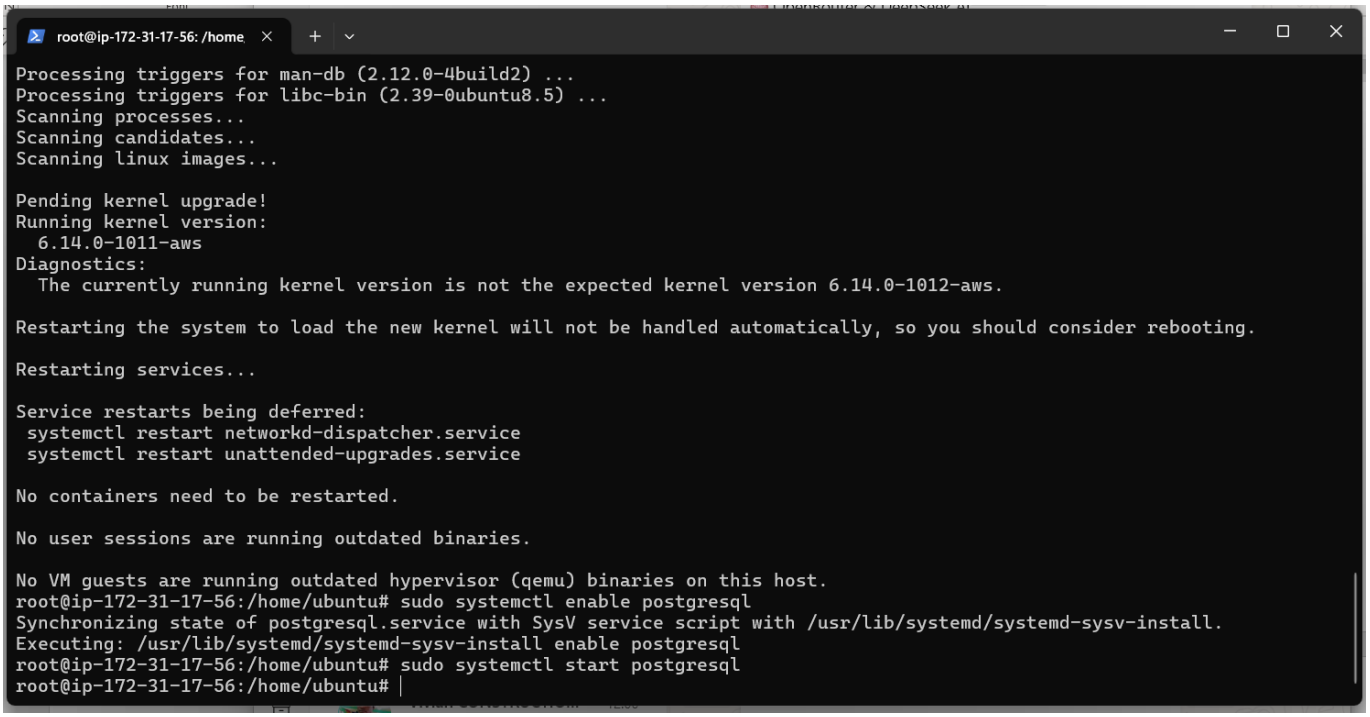
Enable the database server to start automatically on reboot

```
sudo systemctl enable postgresql
```



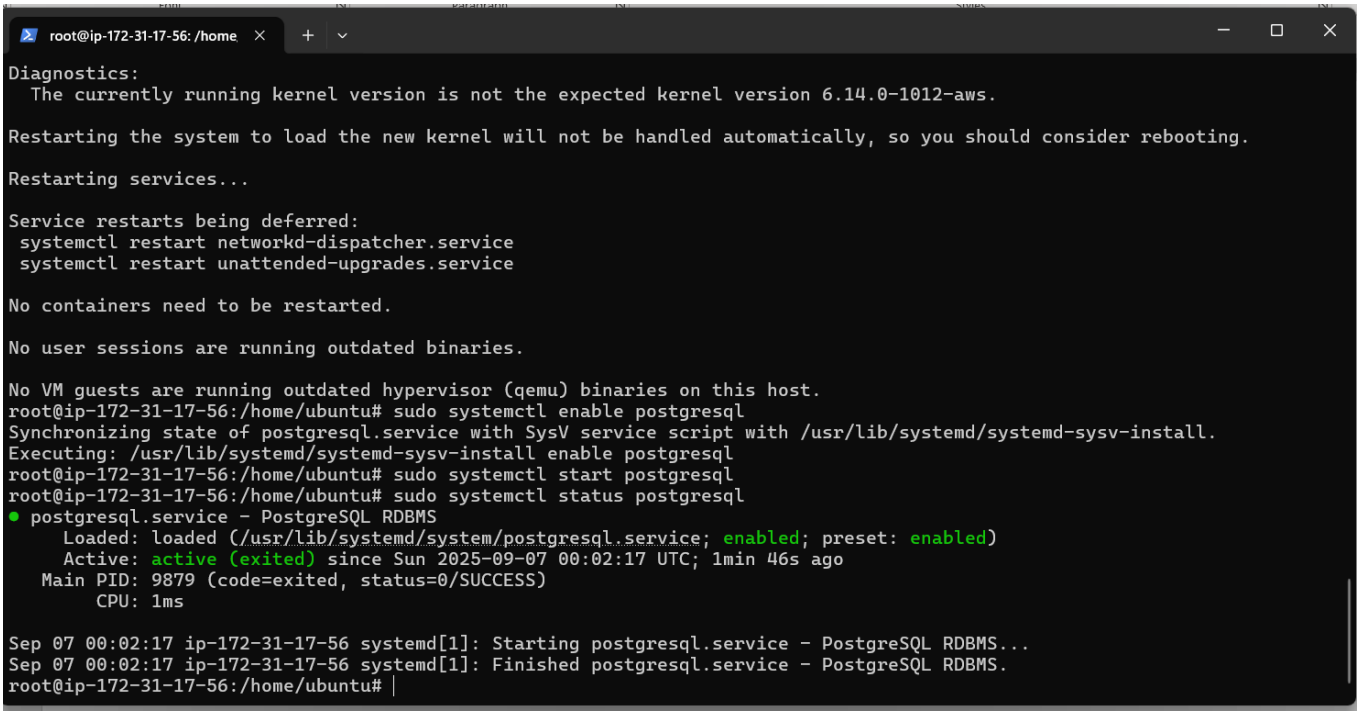
Start the database server

```
sudo systemctl start postgresql
```



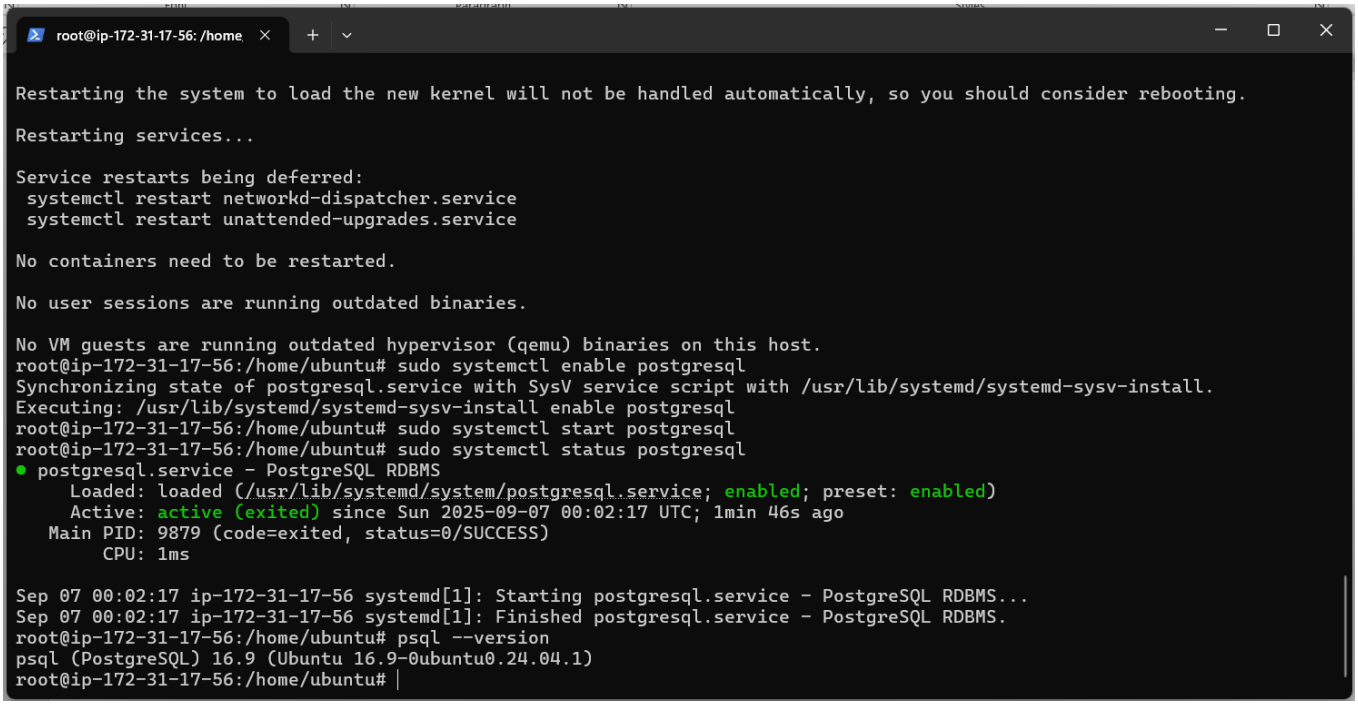
Check the status of the database server

```
sudo systemctl status postgresql
```



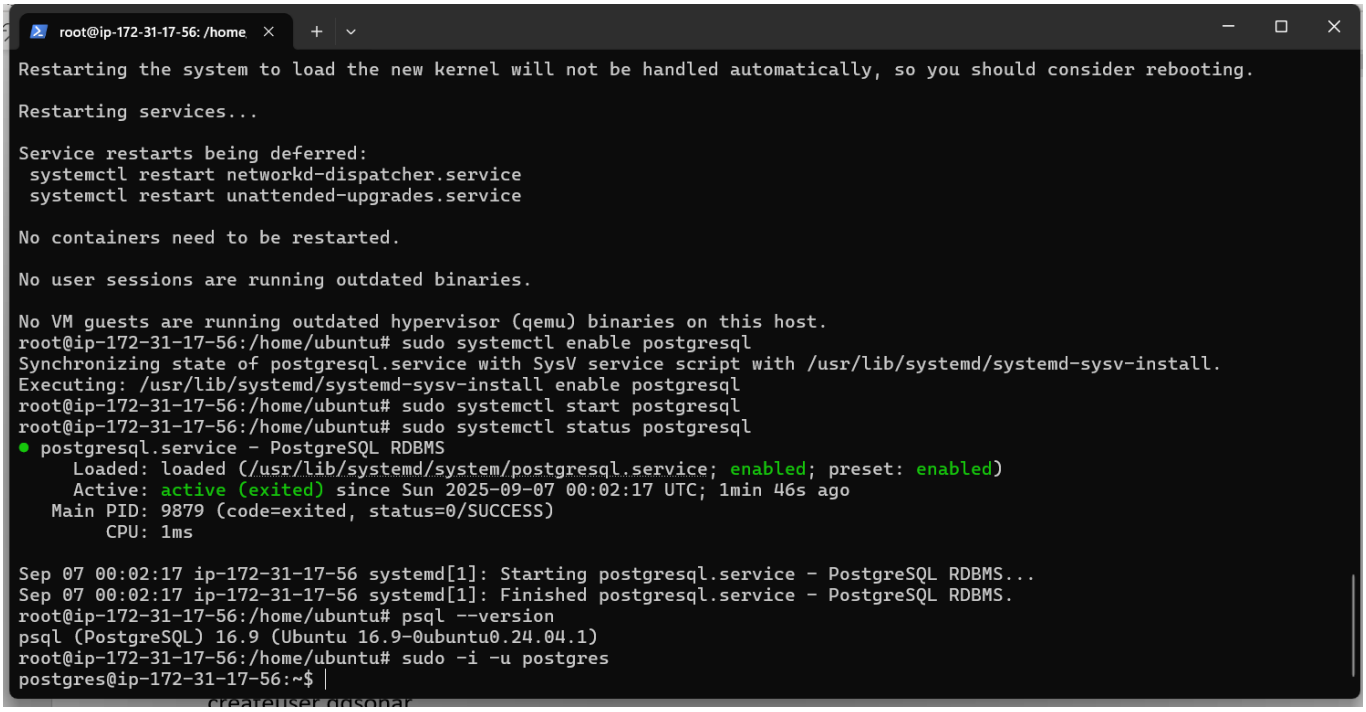
Let's check the installed version of the install Postgres DB. VALIDATION IS IMPORTANT

```
psql --version
```



Switch to the Postgres user

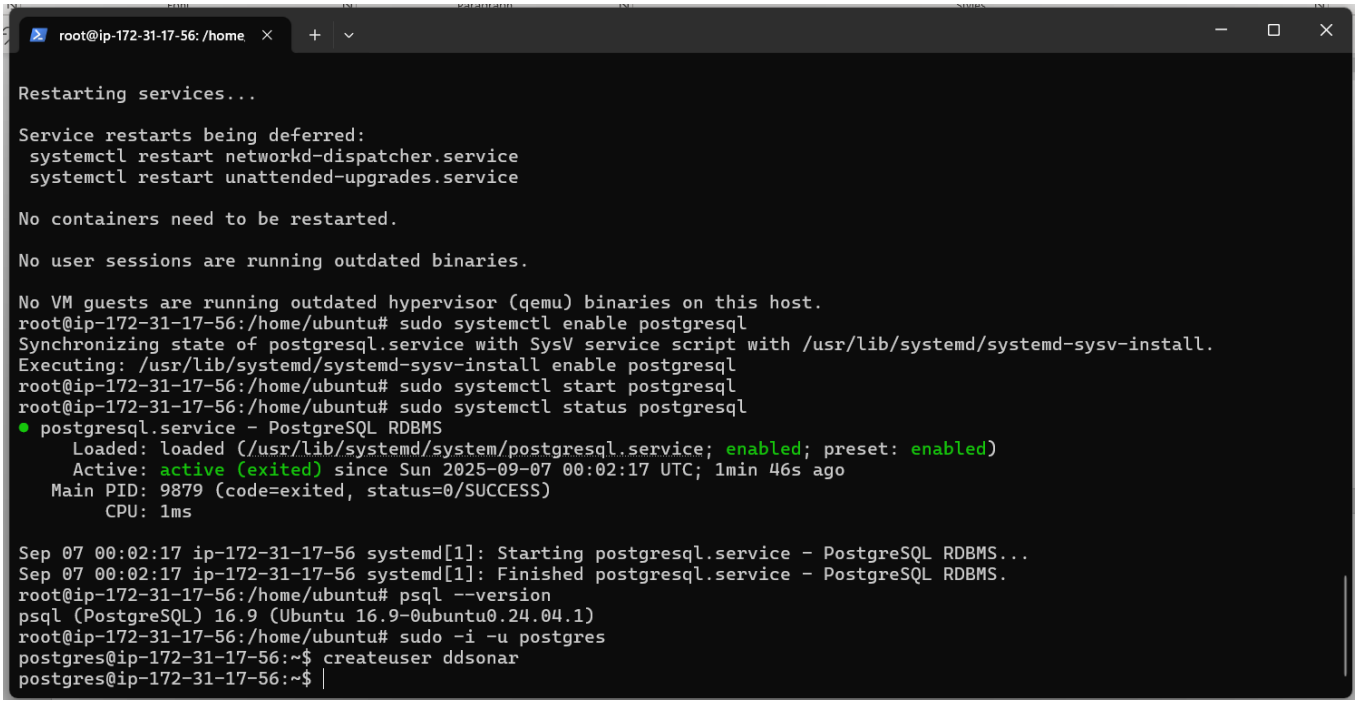
```
sudo -i -u postgres
```



Part 4: Create a database user named ddsonar

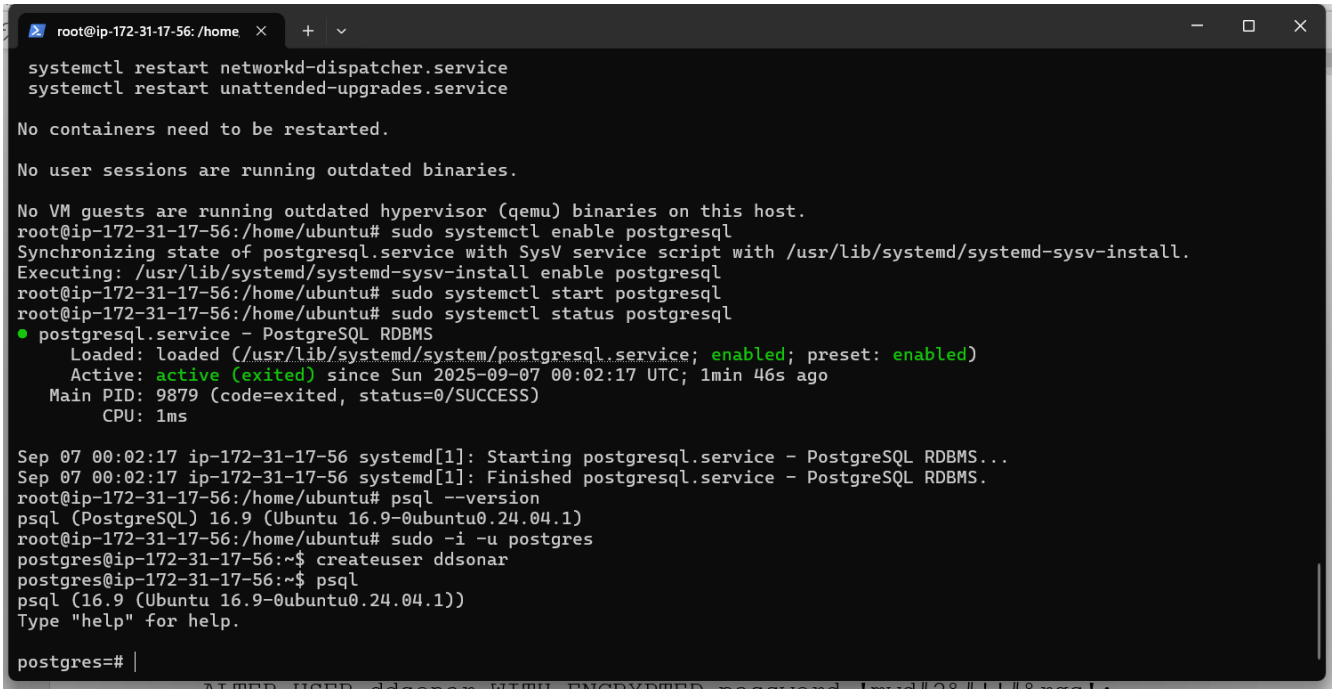
Note: You can provide the name of your own but make a note of it, as we will be needing this name in further steps.

```
createuser ddsonar
```



Log in to PostgreSQL

```
psql
```

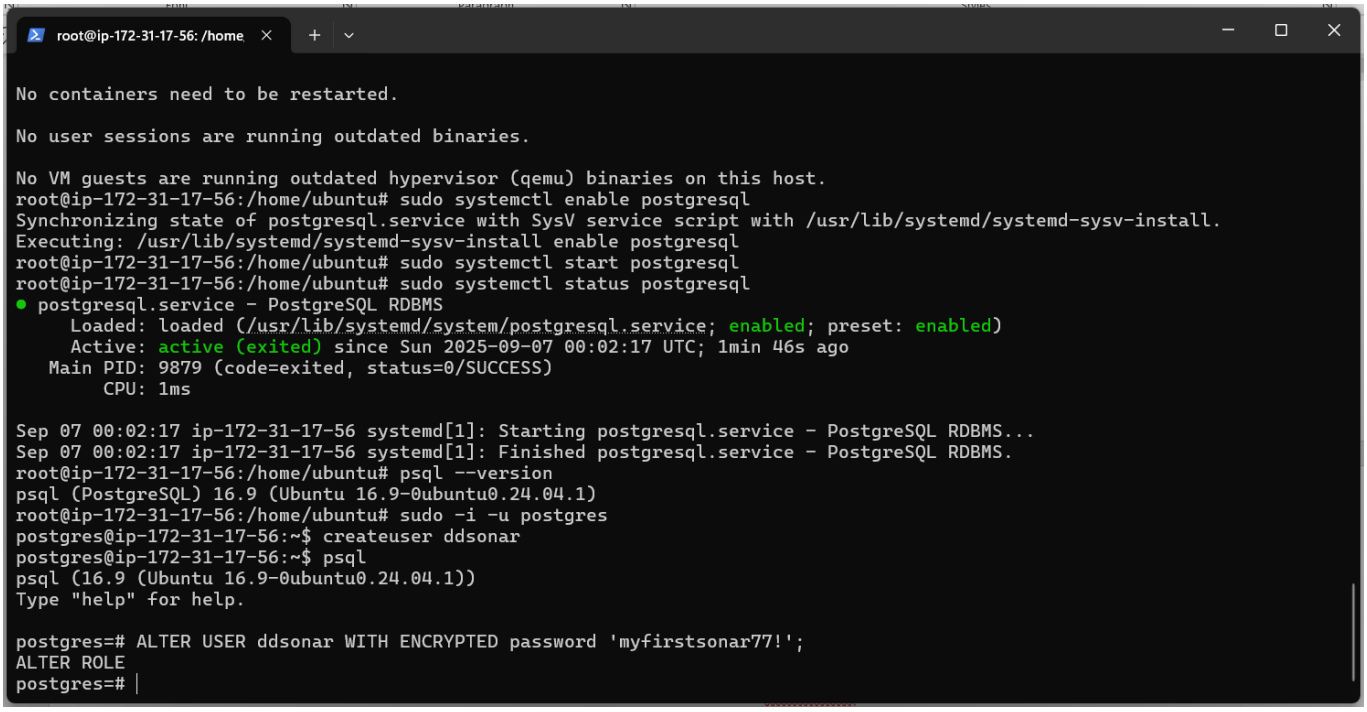


Set a password for the ddsonar user. Use a strong password in place of my_strong_password.

```
ALTER USER [Created_user_name] WITH ENCRYPTED password 'my_strong_password';
```

For example (In my case it will be):

```
ALTER USER ddsonar WITH ENCRYPTED password 'myfirstsonar77!';
```

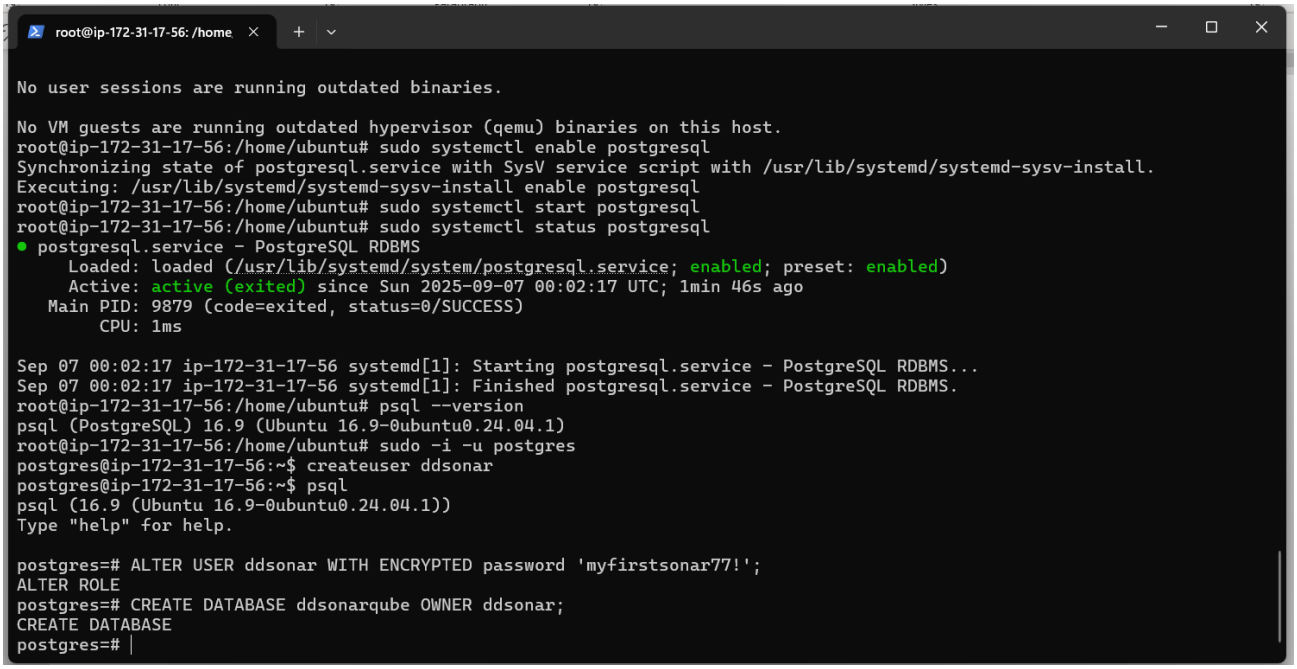


Create a SonarQube database and set the owner to ddsonar.

```
CREATE DATABASE [database_name] OWNER [Created_user_name];
```

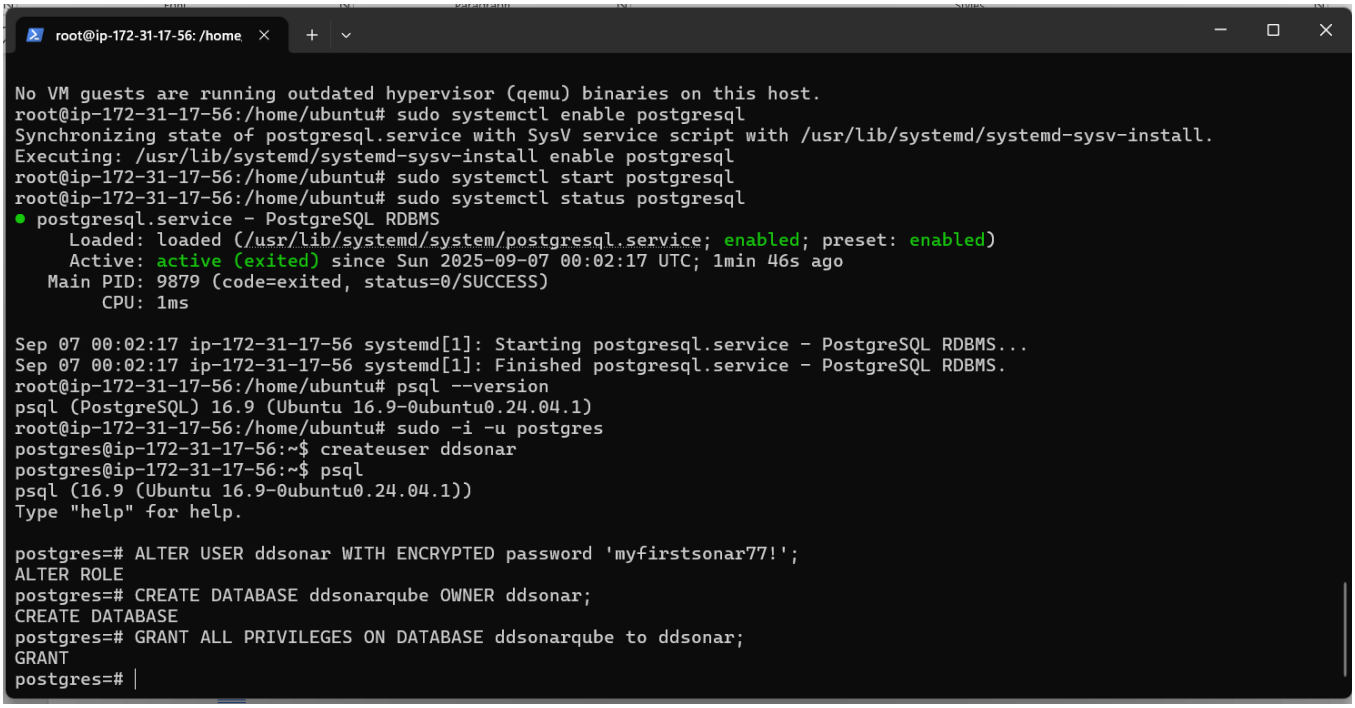
For example (In our case it will be) — feel free to give an awesome database name as per your requirement:

```
CREATE DATABASE ddsonarqube OWNER ddsonar;
```



Grant all the privileges on the ddsonarqube database to the ddsonar user.

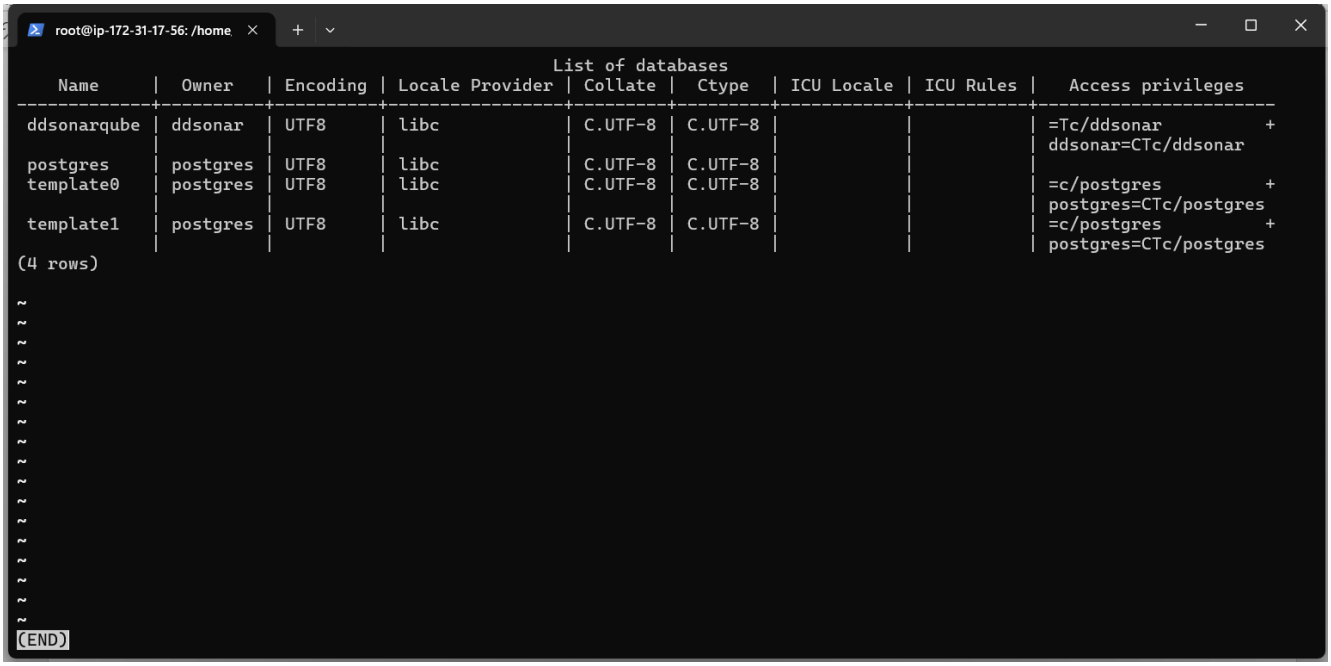
```
GRANT ALL PRIVILEGES ON DATABASE ddsonarqube to ddsonar;
```



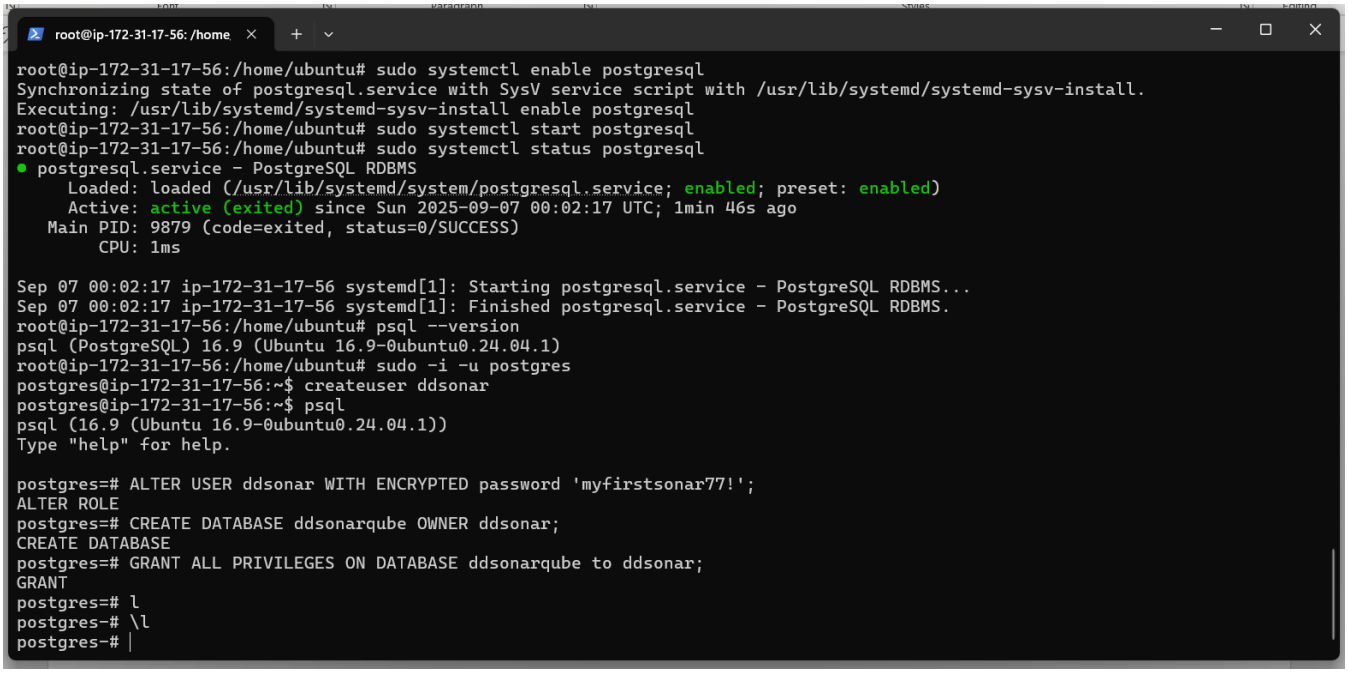
Let's check the created user and the database.

a) To check the created database using the command:

```
\l
```

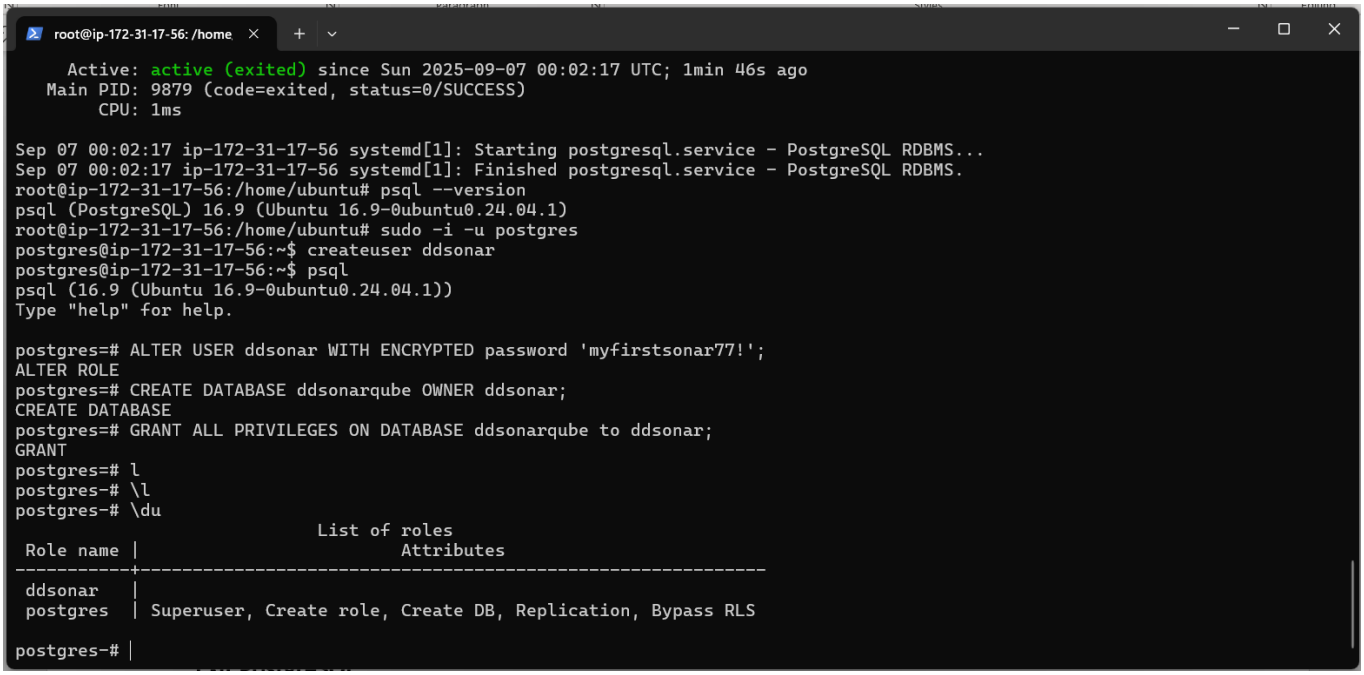


Press “**q**”



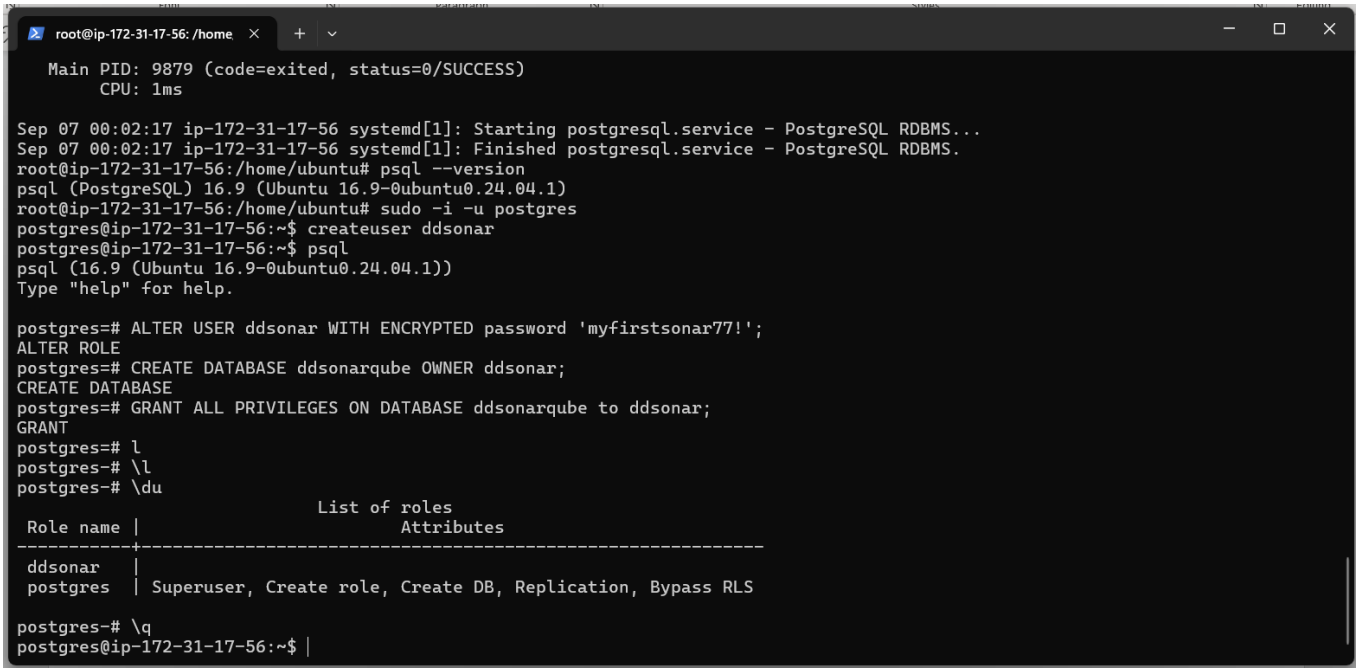
b) To check the created database user

```
\du
```



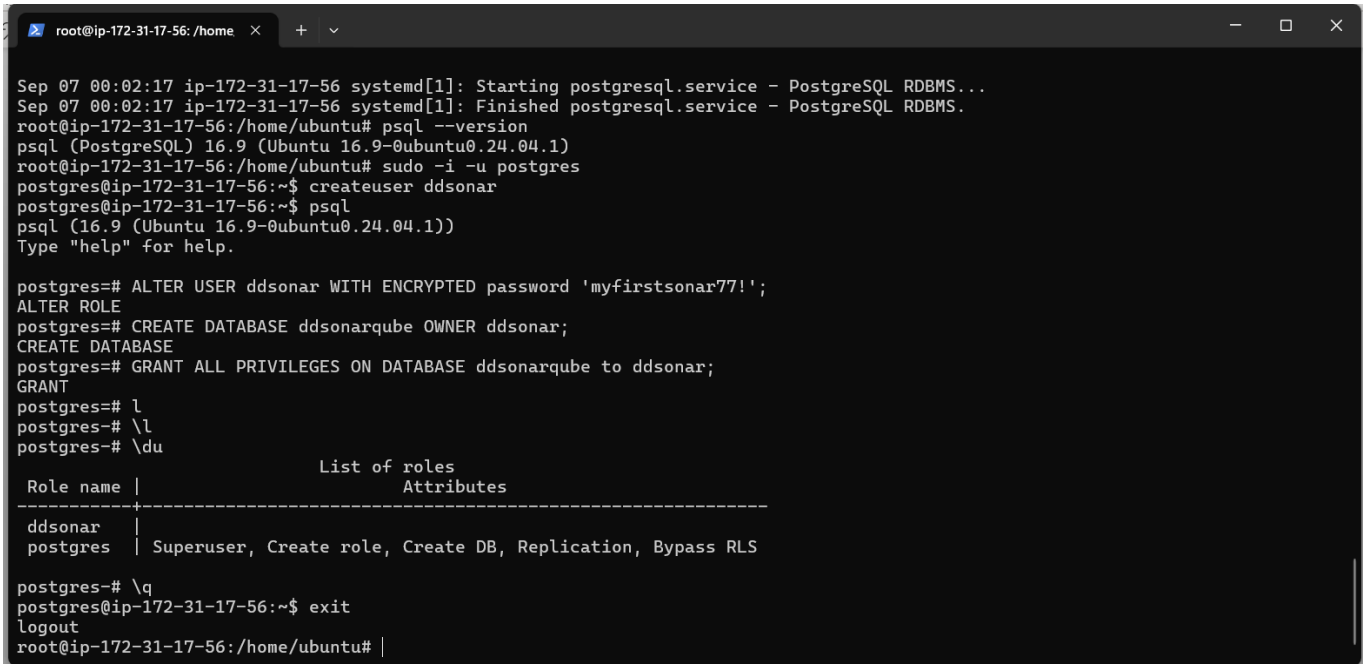
Exit PostgreSQL.

```
\q
```



Return to your non-root sudo user account.

```
exit
```

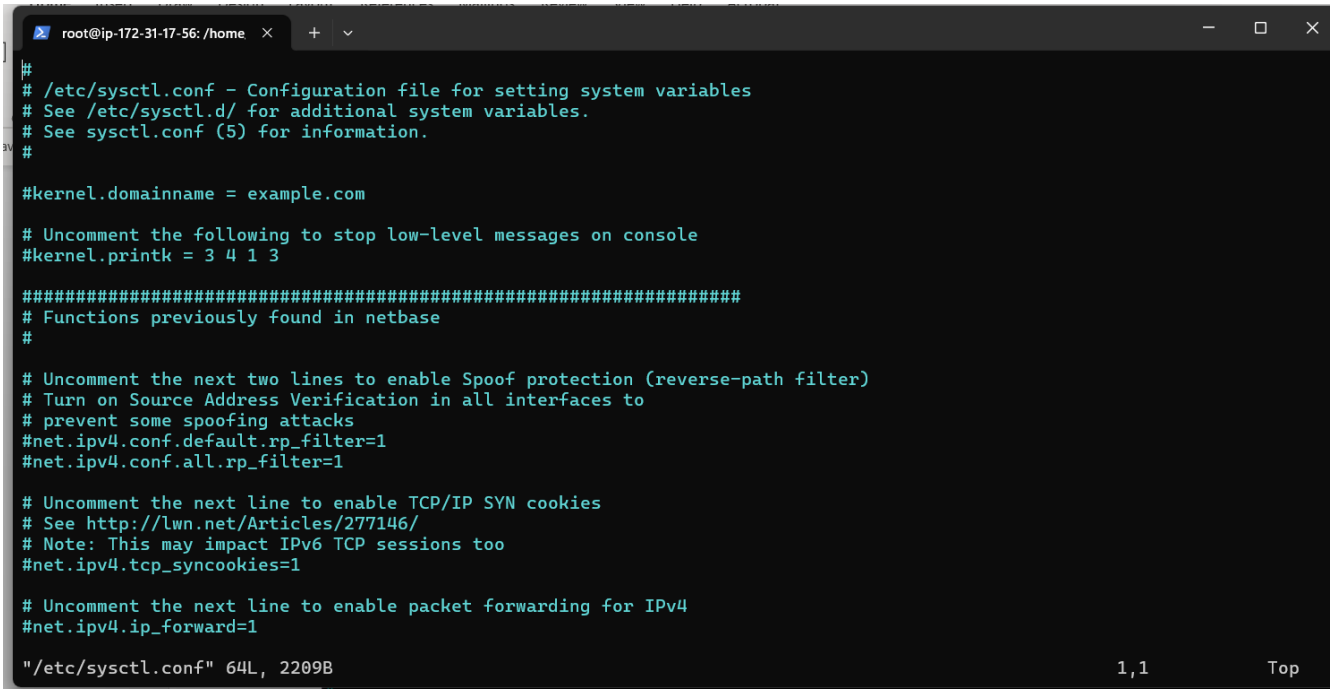


Part 5: Modify Kernel System Limits

SonarQube uses Elasticsearch to store its indices in an MMap FS directory. It requires some changes to the system defaults.

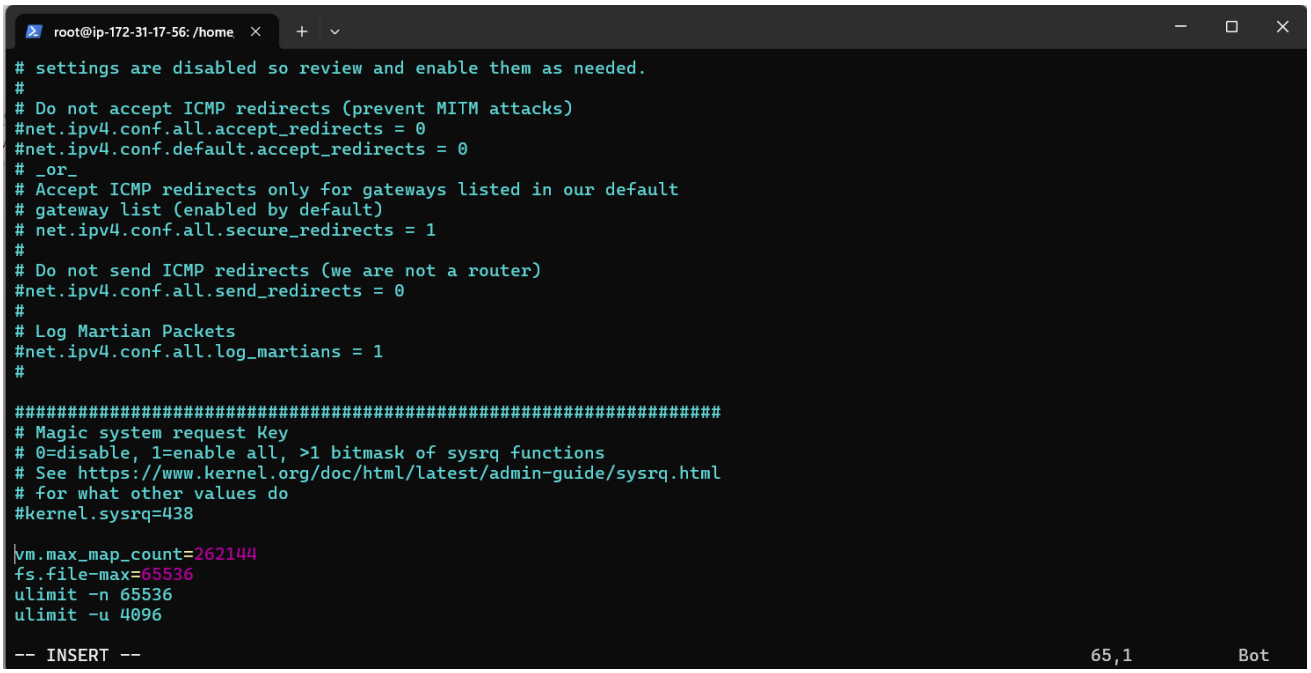
Edit the sysctl configuration file.

```
sudo vim /etc/sysctl.conf
```



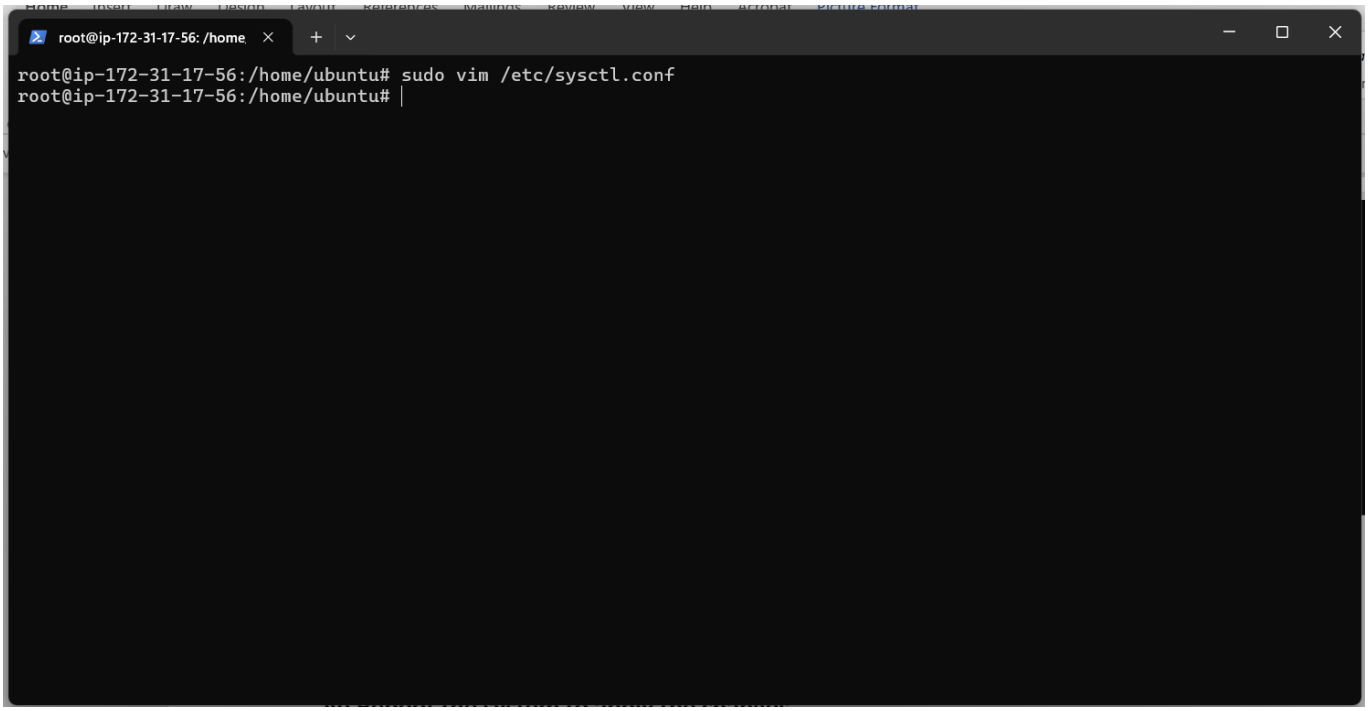
Increase file descriptor limits

```
vm.max_map_count=262144
fs.file-max=65536
ulimit -n 65536
ulimit -u 4096
```



Save and exit the file

Press ESC, followed by :wq and press ENTER


root@ip-172-31-17-56: /home
root@ip-172-31-17-56:/home/ubuntu# sudo vim /etc/sysctl.conf
root@ip-172-31-17-56:/home/ubuntu#

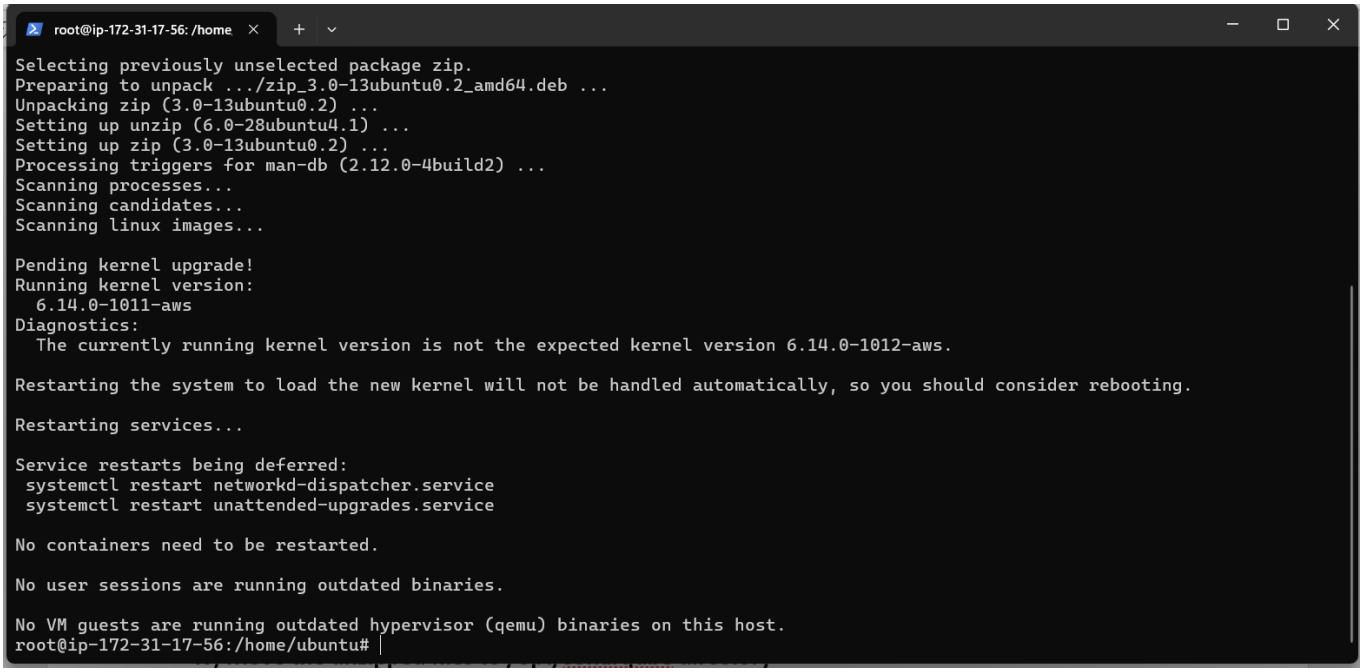
STEP 3: Install SonarQube

We will Download the latest SonarQube LTS version from the official website, Create a dedicated user for SonarQube, Configure SonarQube to connect to PostgreSQL, Create a systemd service for SonarQube, Enable and start SonarQube,

Part 1: Download and Install SonarQube

Install the zip utility, which is needed to unzip the SonarQube files.

```
sudo apt install zip -y
```



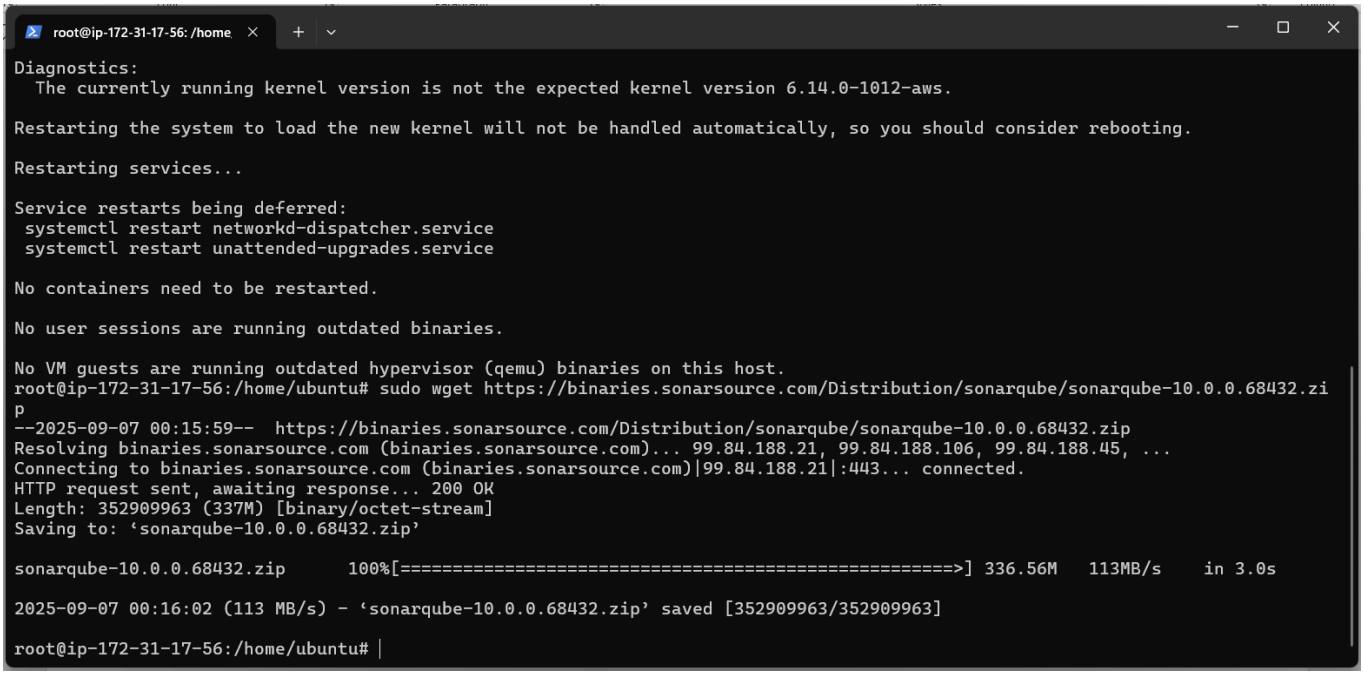
Locate the latest download URL from the SonarQube official download page.

Download the SonarQube distribution files. (You can download the latest SonarQube distribution using the following link)

https://www.sonarsource.com/products/sonarqube/downloads/

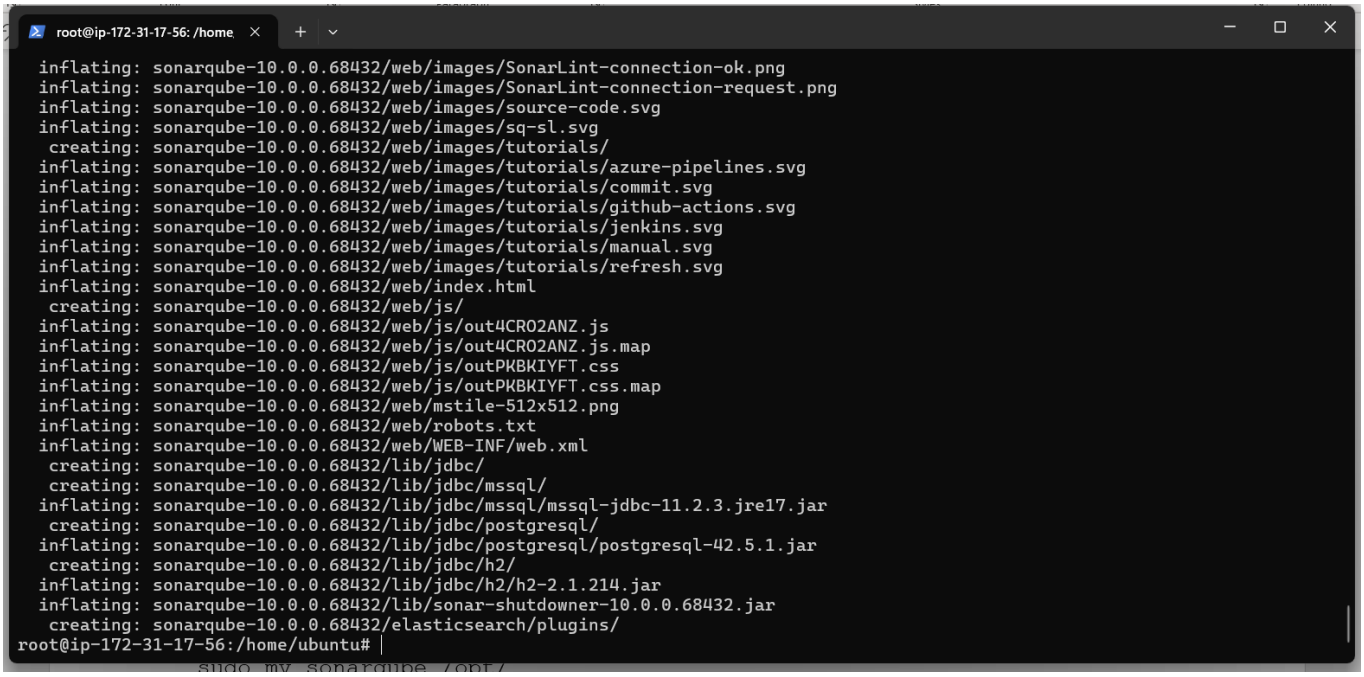
Here we are installing the latest version of SonarQube 10.0 community edition (free one)

```
sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.0.0.68432.zip
```



Unzip the downloaded file.

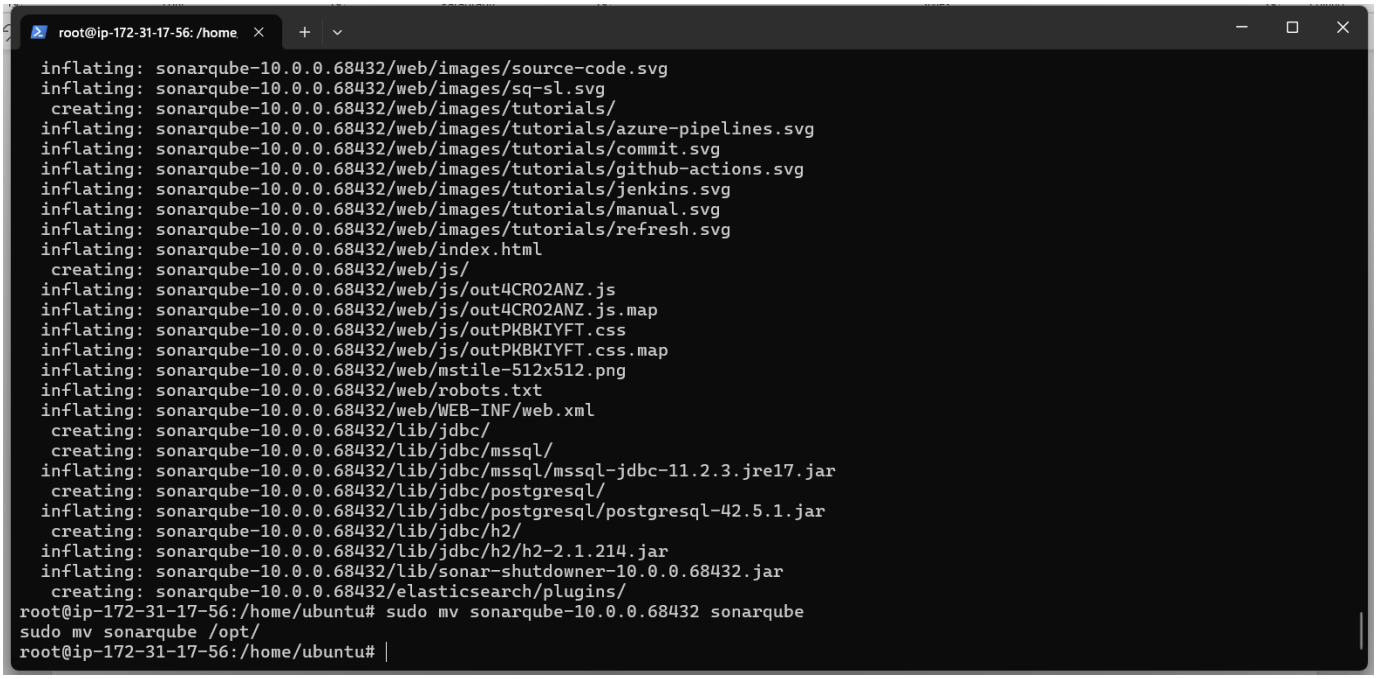
```
sudo unzip sonarqube-10.0.0.68432.zip
```



Move the unzipped files to /opt/sonarqube directory

```
sudo mv sonarqube-10.0.0.68432 sonarqube
```

```
sudo mv sonarqube /opt/
```



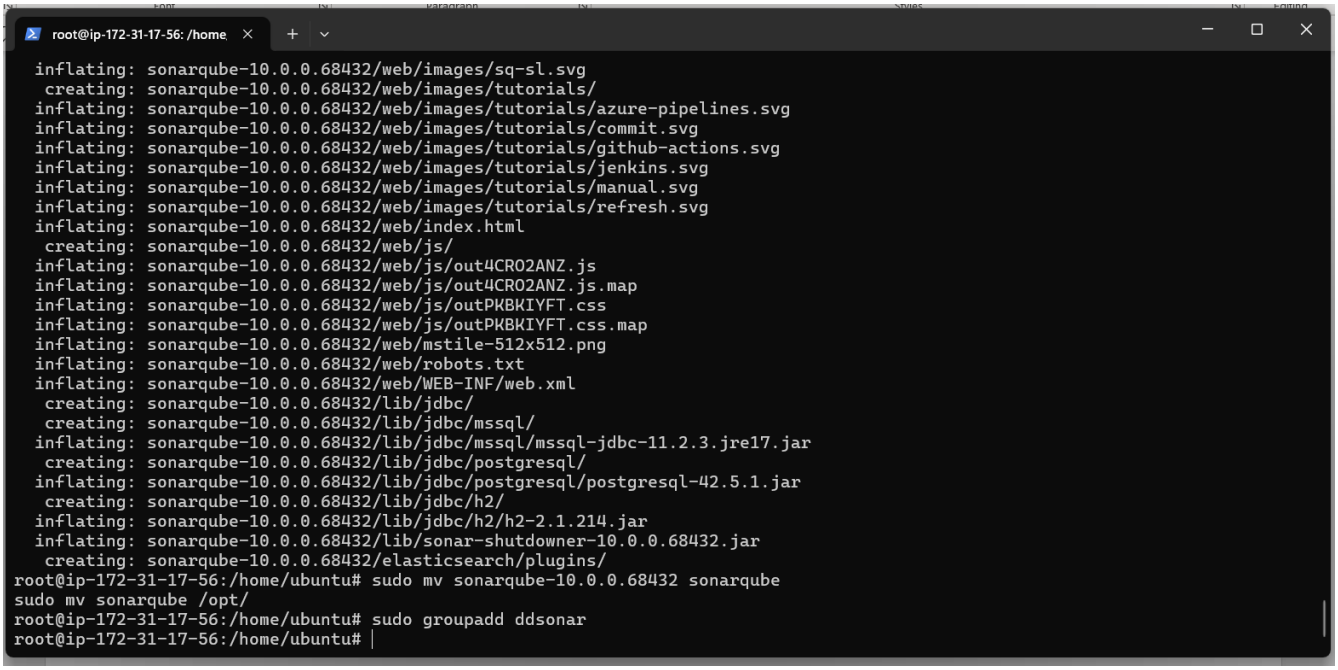
Part 2: Add SonarQube Group and User

Create a dedicated user and group for SonarQube, which can not run as the root user.

Note: You can give any name for the sonar user and group. I have here given the user and group name to be the same i.e **ddsonar**.

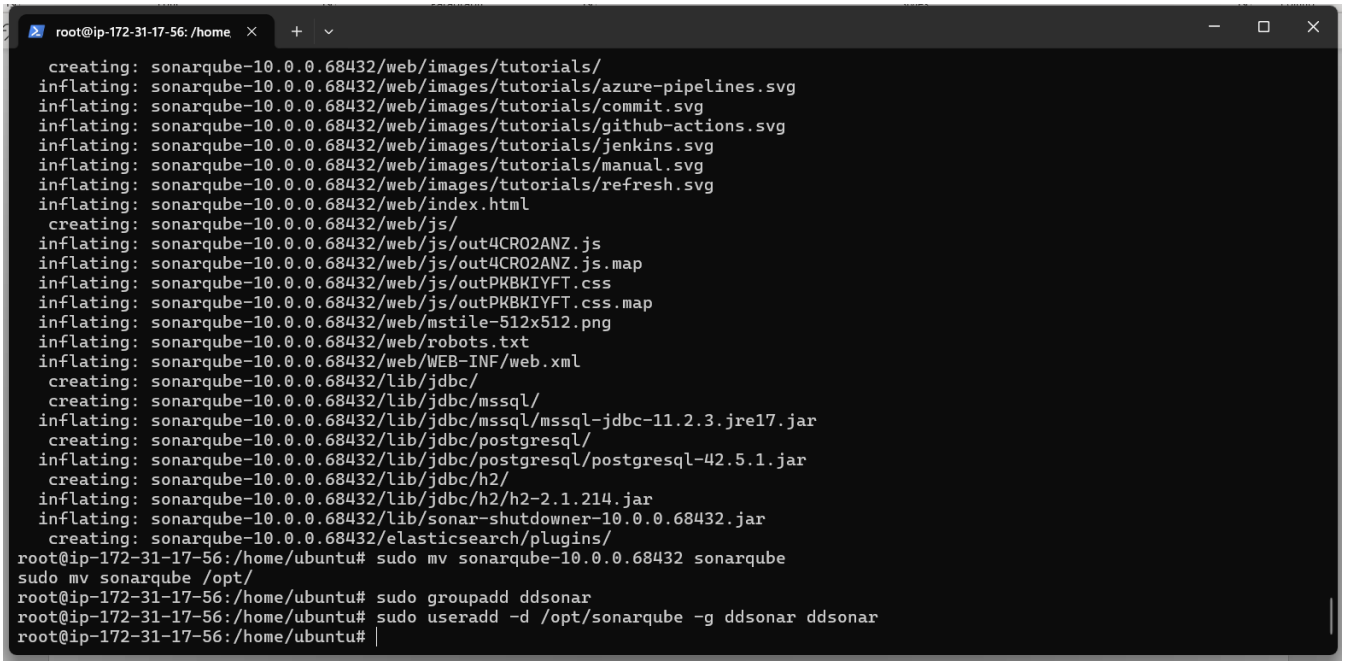
Create a sonar group.

```
sudo groupadd ddsonar
```



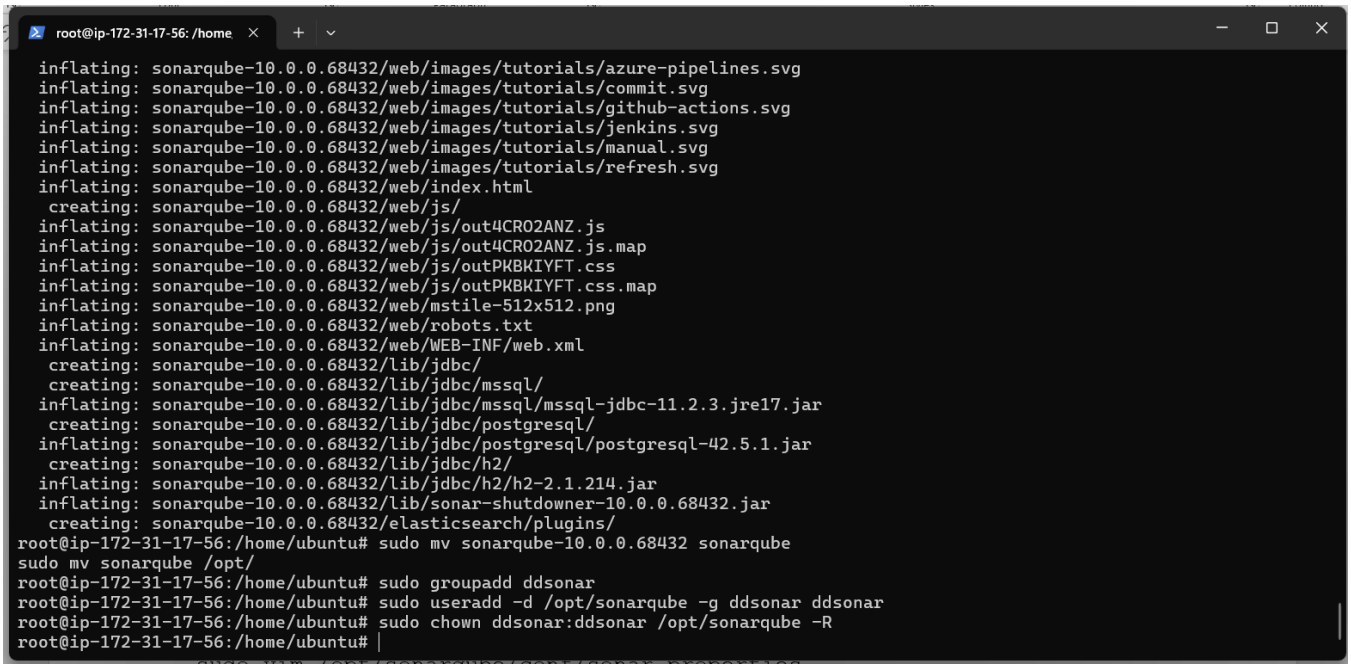
Create a sonar user and set /opt/sonarqube as the home directory.

```
sudo useradd -d /opt/sonarqube -g ddsonar ddsonar
```



Grant the sonar user access to the /opt/sonarqube directory.

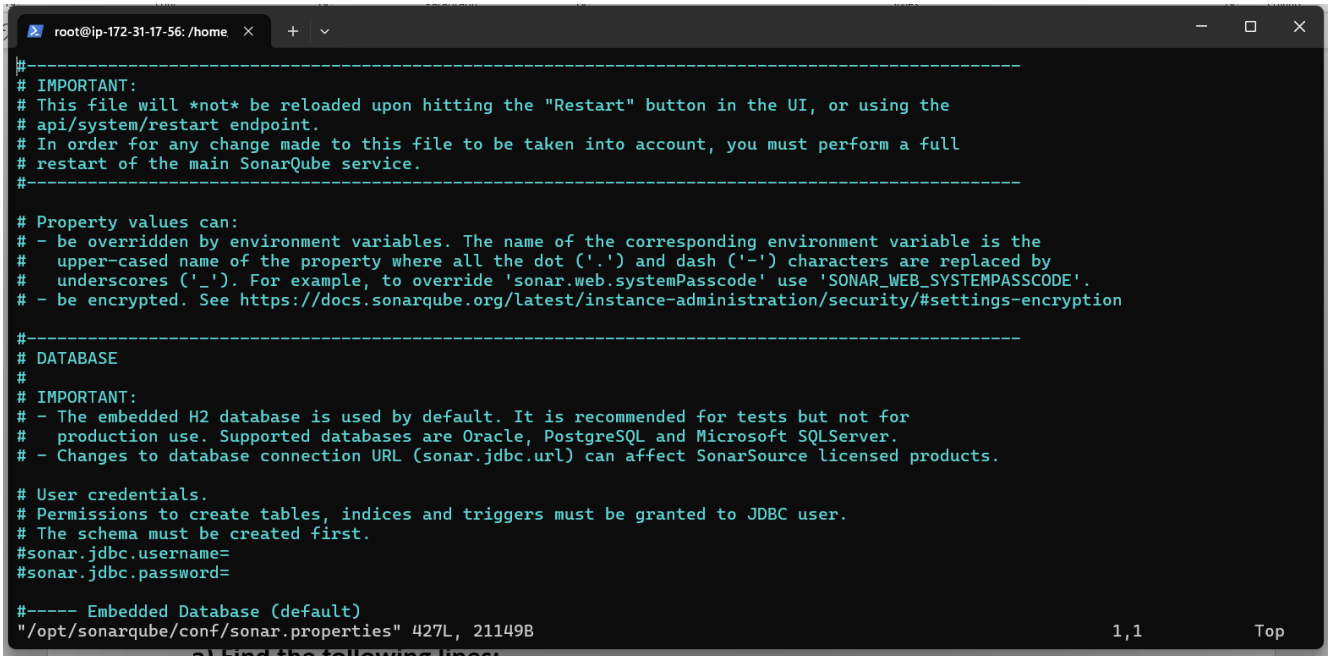
```
sudo chown ddsonar:ddsonar /opt/sonarqube -R
```



Part 3: Configure SonarQube to connect with PostgreSQL

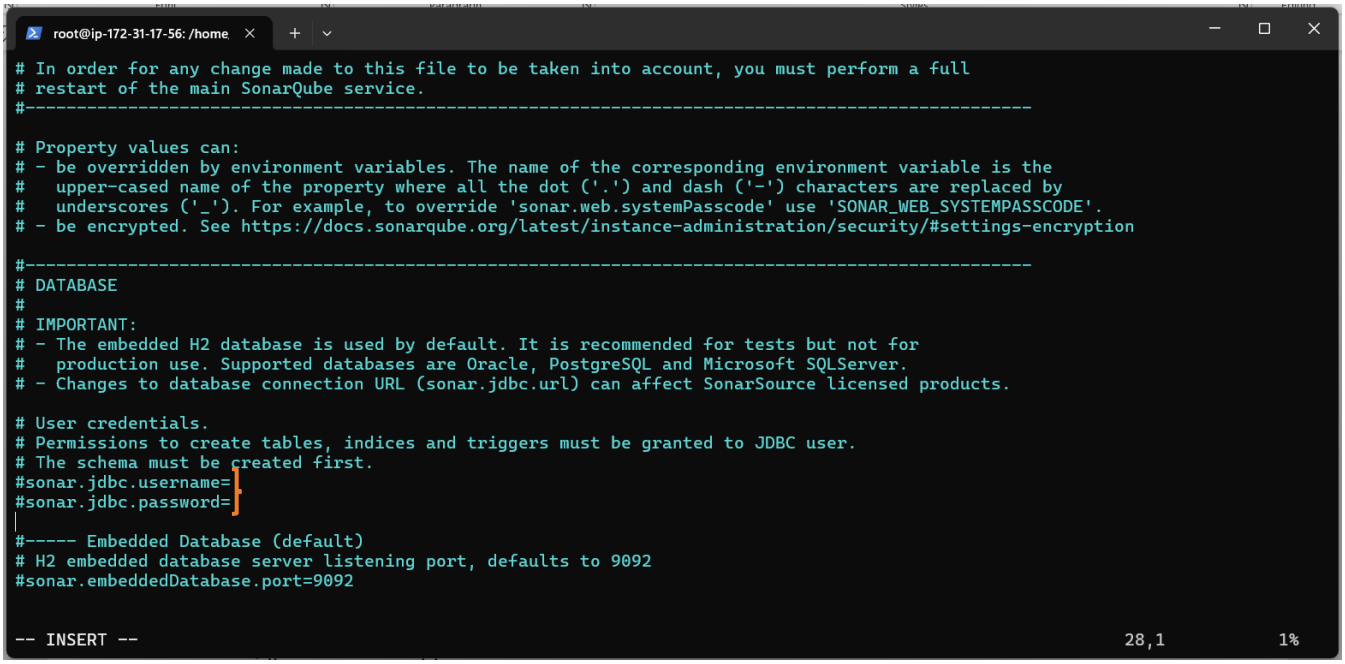
Edit the SonarQube configuration file.

```
sudo vim /opt/sonarqube/conf/sonar.properties
```



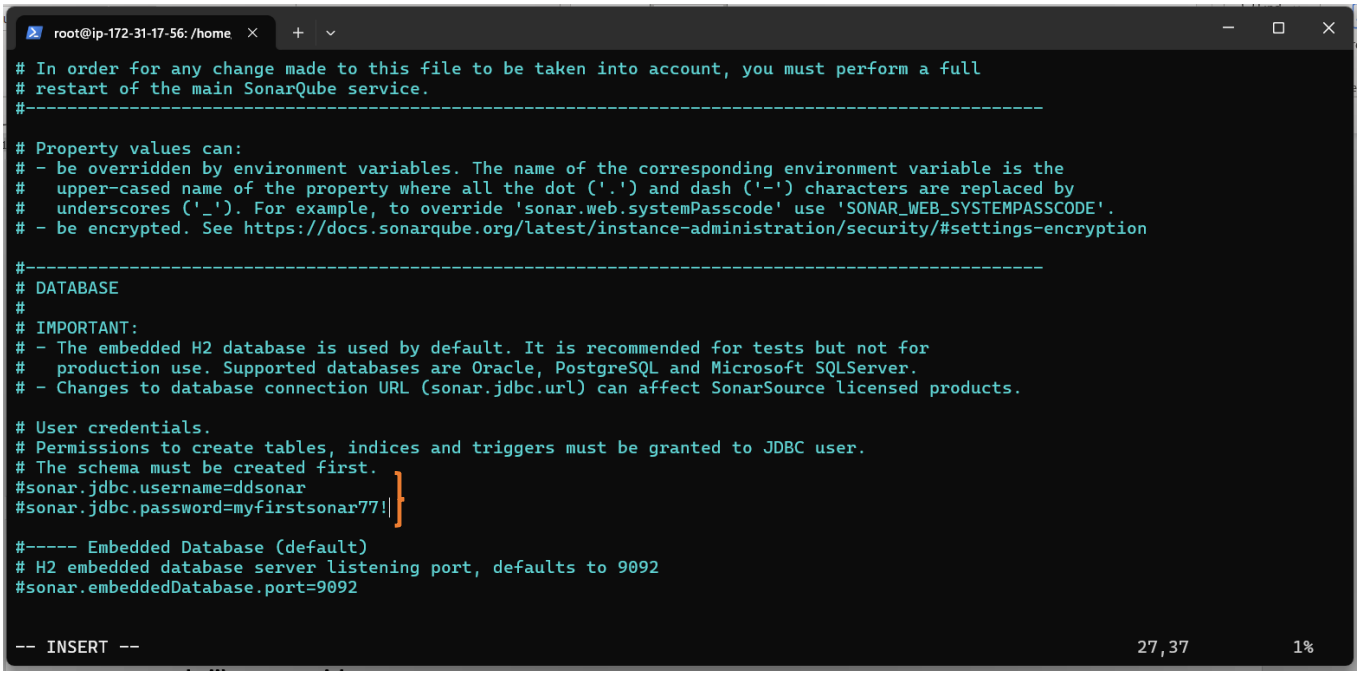
Find the following lines:

```
#sonar.jdbc.username=
#sonar.jdbc.password=
```



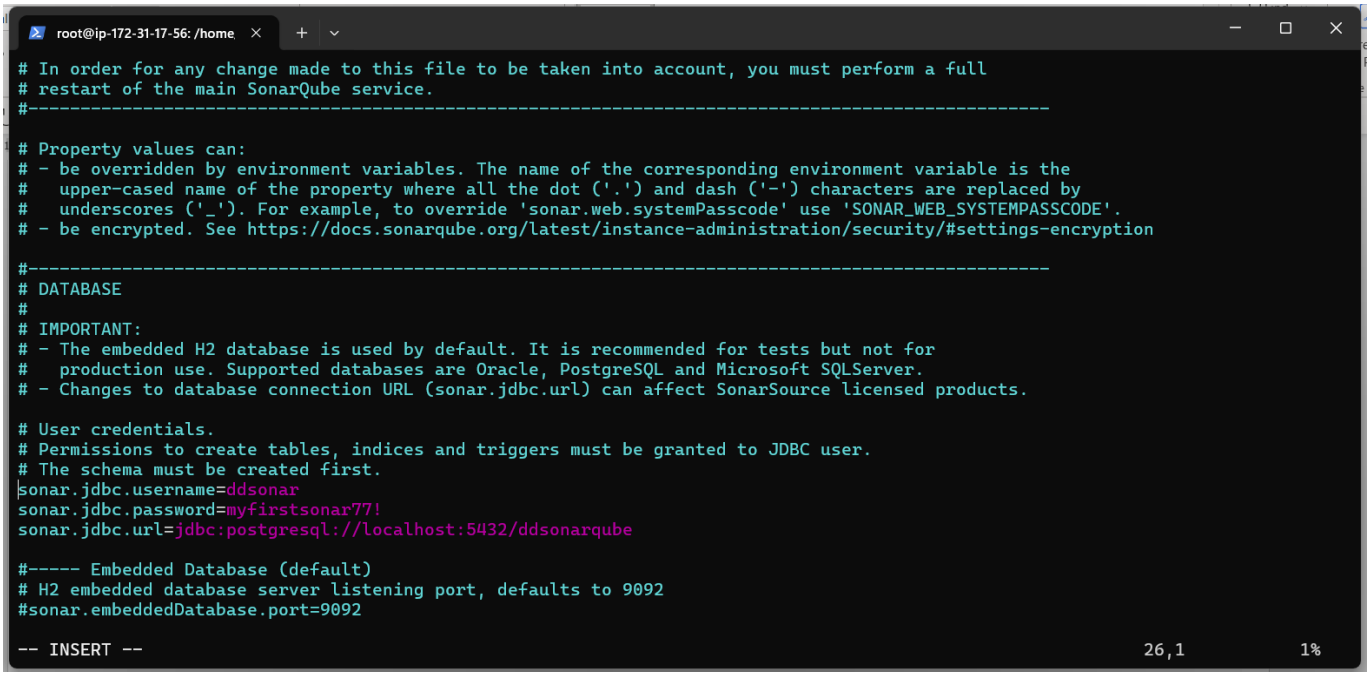
Uncomment the lines, and add the database user and Database password you created in Step 3. For me, it's:

```
sonar.jdbc.username=ddsonar
sonar.jdbc.password=myfirstsonar77!
```



Below these two lines, add the following line of code.

`sonar.jdbc.url=jdbc:postgresql://localhost:5432/ddsonarqube`



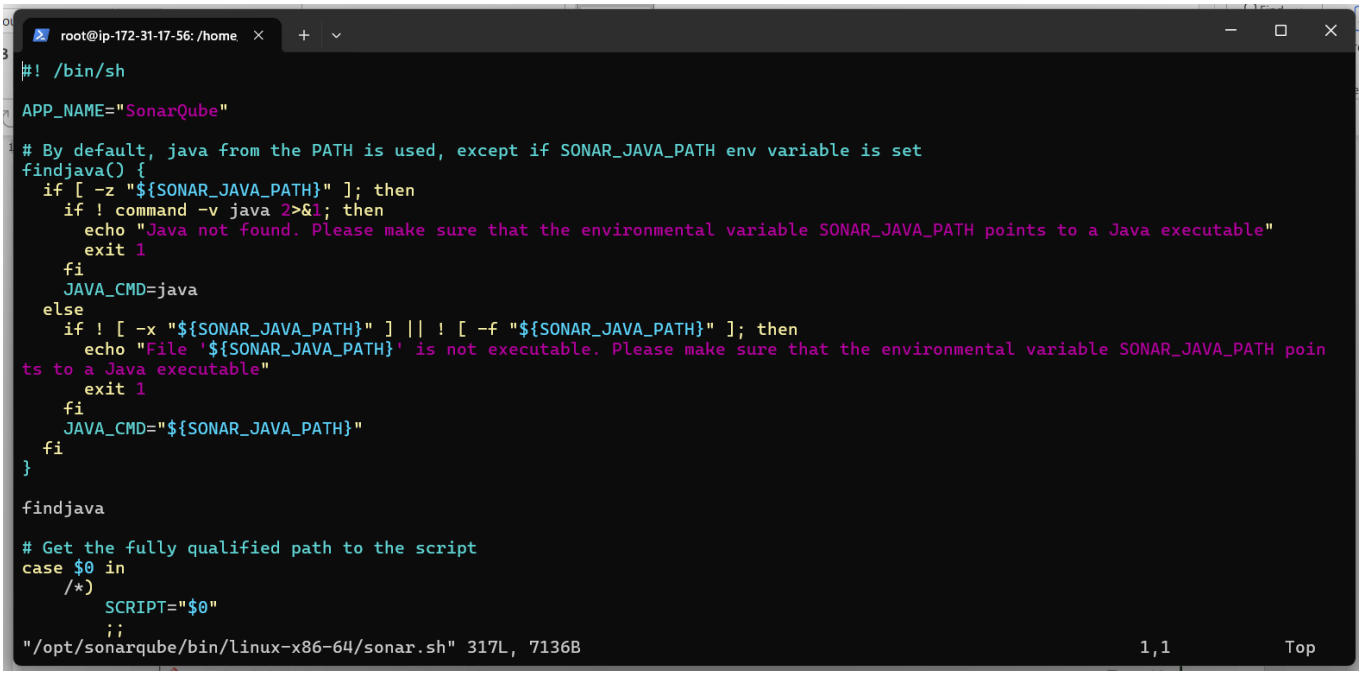
Here, ddsonarqube is the database name created.

Save and exit the file.

Press ESC, followed by :wq and press ENTER

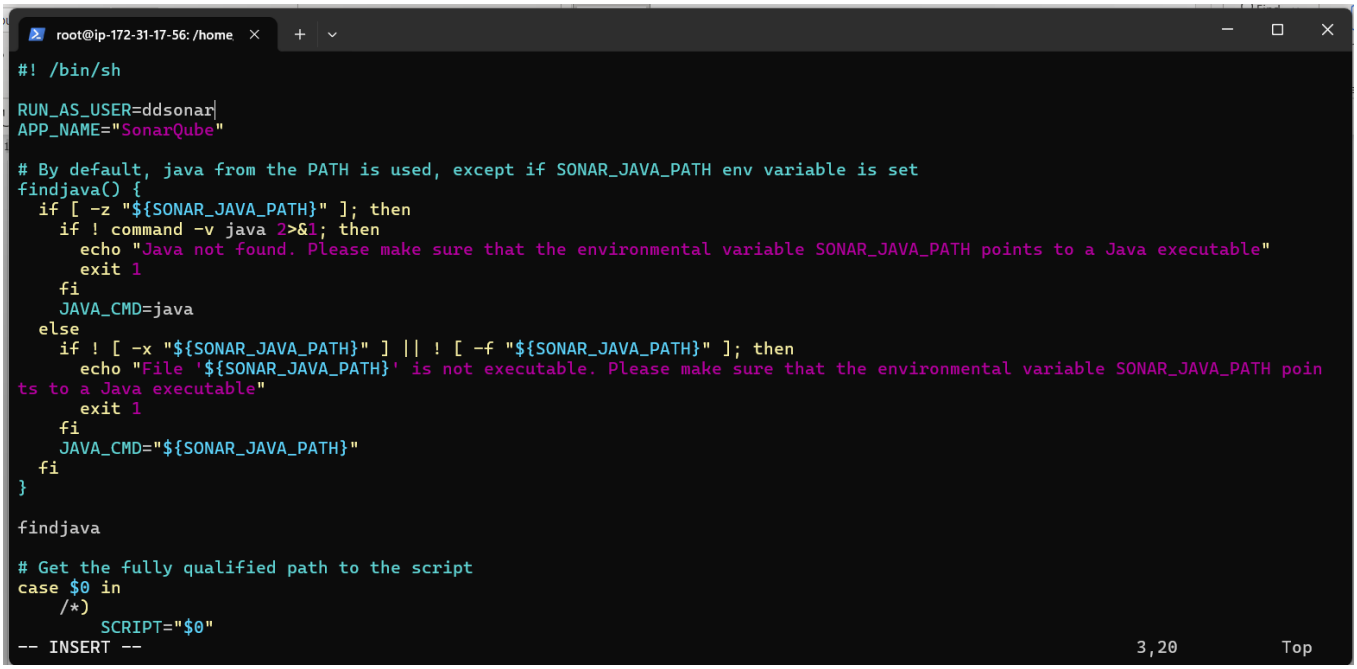
Edit the sonar script file.

`sudo vim /opt/sonarqube/bin/linux-x86-64/sonar.sh`



Add the following line

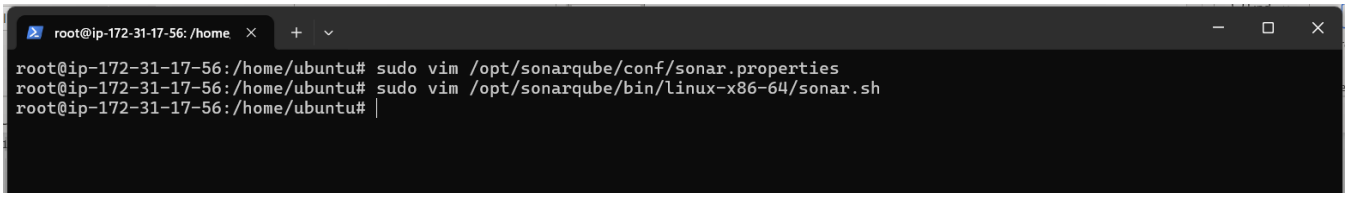
`RUN_AS_USER=ddsonar`



Here, ddsonar is the name of the user that we have created in step number 6 (ii).

Save and exit the file.

Press ESC, followed by :wq and press ENTER



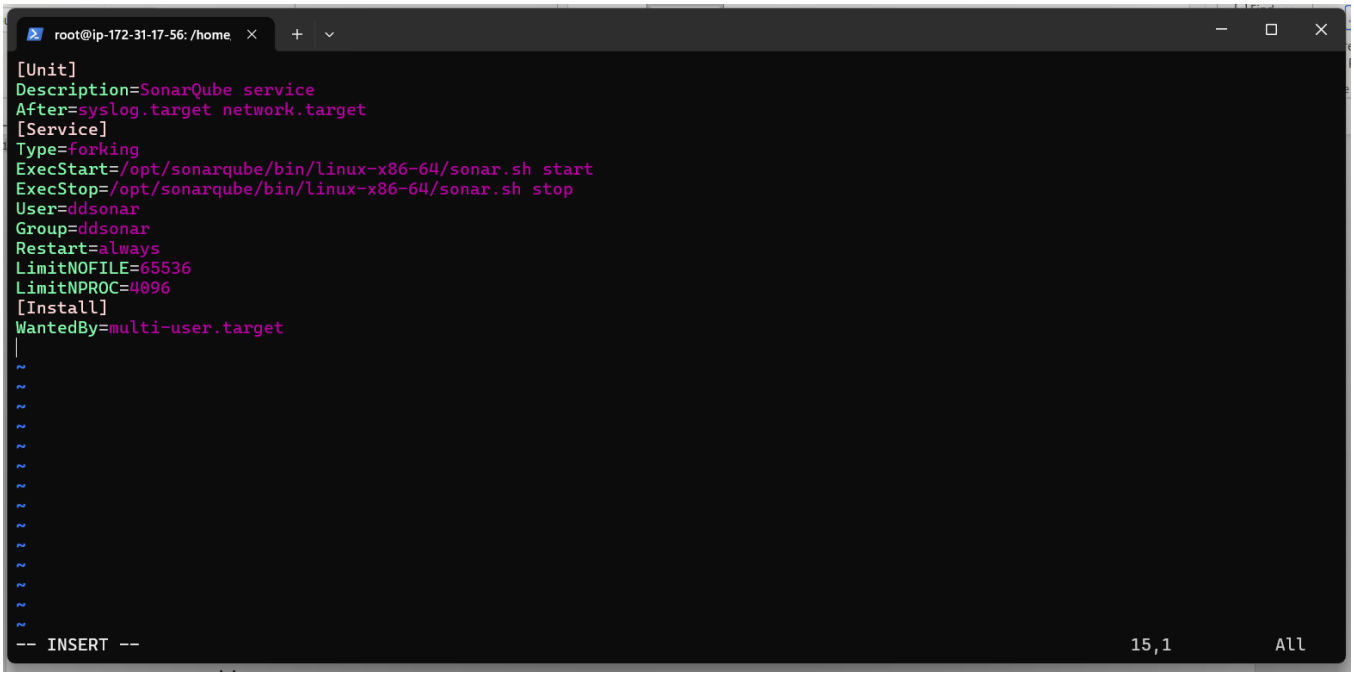
Part 4: Create Systemd service for SonarQube

Create a systemd service file to start SonarQube at system boot.

```
sudo vim /etc/systemd/system/sonar.service
```

Paste the following lines to the file.

```
[Unit]
Description=SonarQube service
After=syslog.target network.target
[Service]
Type=forking
ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop
User=ddsonar
Group=ddsonar
Restart=always
LimitNOFILE=65536
LimitNPROC=4096
[Install]
WantedBy=multi-user.target
```



Note: Here in the above script, make sure to change the User and Group section with the value that you have created. For me its:

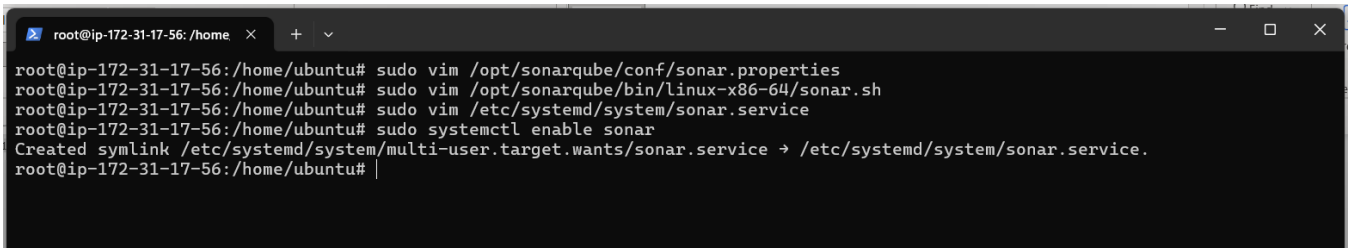
```
User=ddsonar
Group=ddsonar
```

Save and exit the file.

Press ESC, followed by :wq then press ENTER

Part 5: Enable the SonarQube service to run at system startup

```
sudo systemctl enable sonar
```

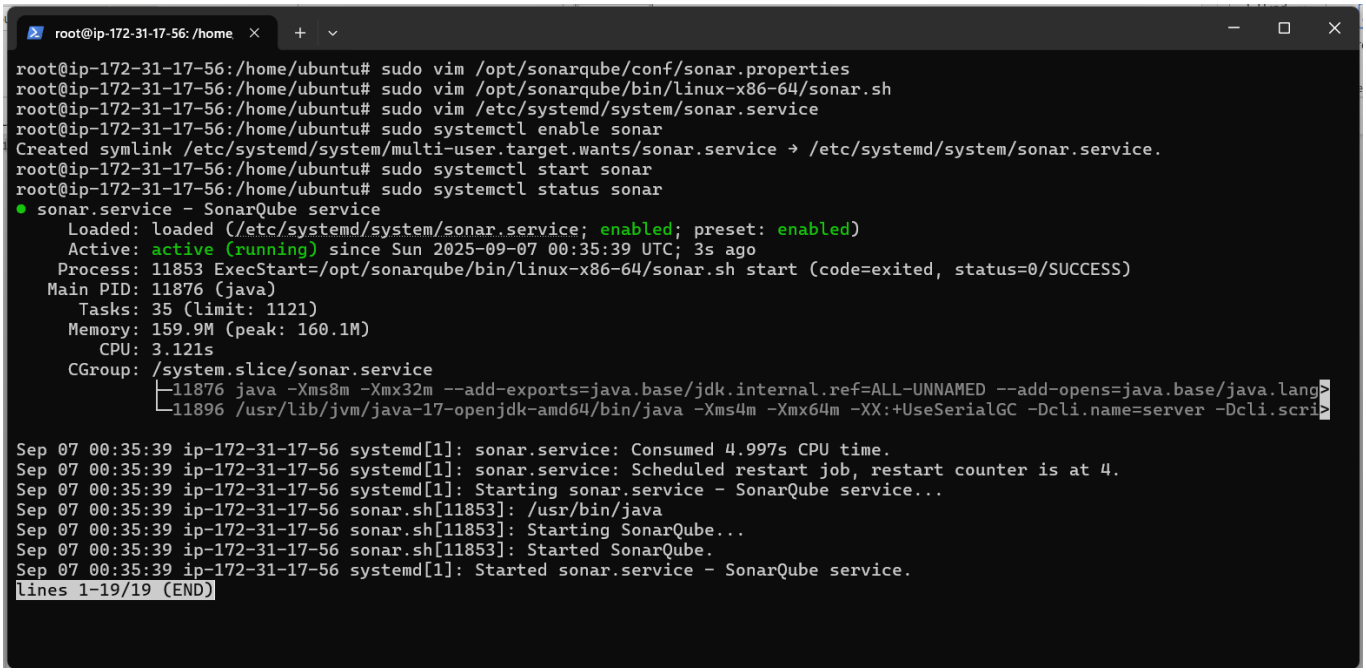


Start the SonarQube service

```
sudo systemctl start sonar
```

Check the service status

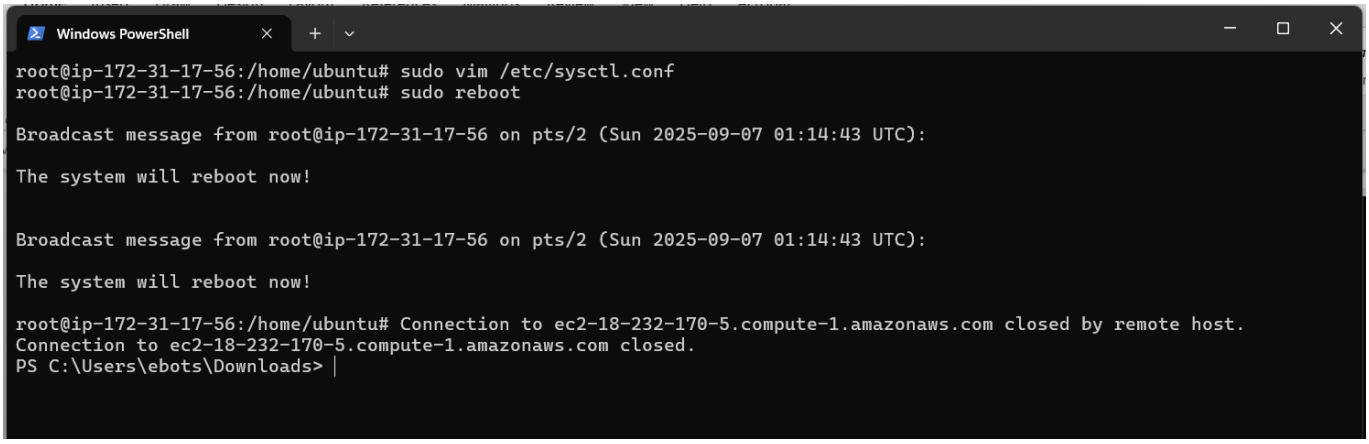
```
sudo systemctl status sonar
```



It is up and running. Then type “q” to exit this mode

Reboot the system to apply the changes.

```
sudo reboot
```

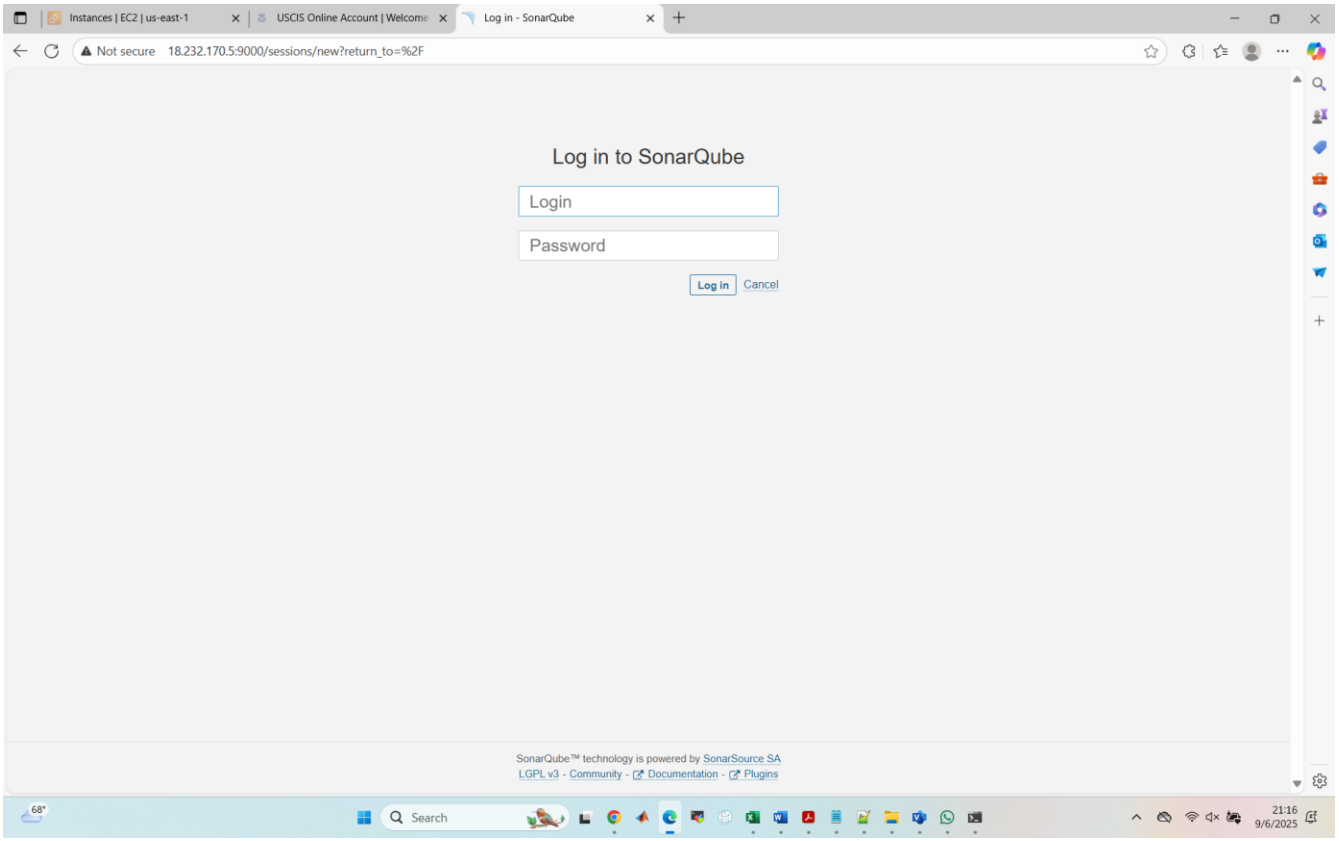


STEP 5: Access SonarQube Web Interface

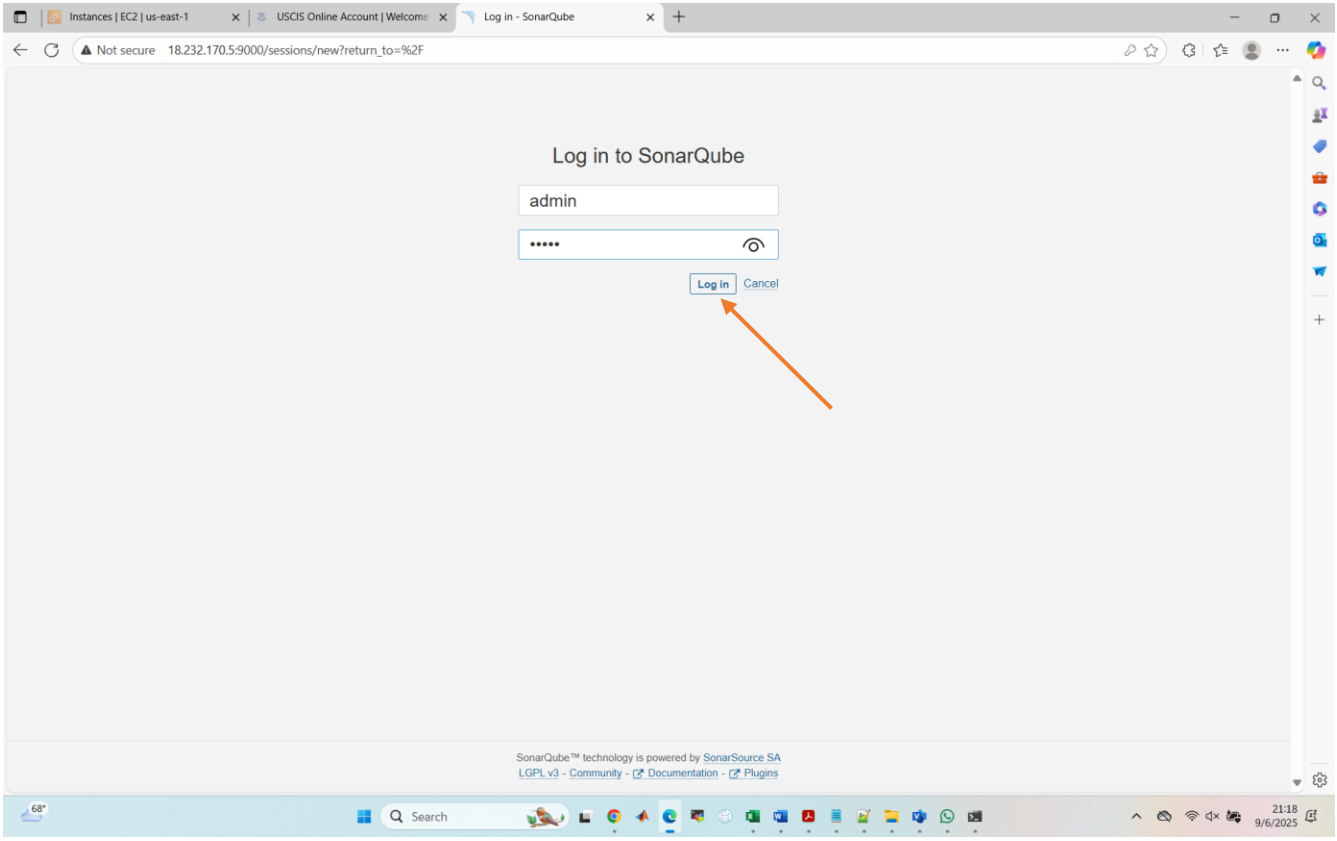
Access SonarQube in a web browser at your server's IP address on port 9000.

For example, http://Public IPv4 Address:9000

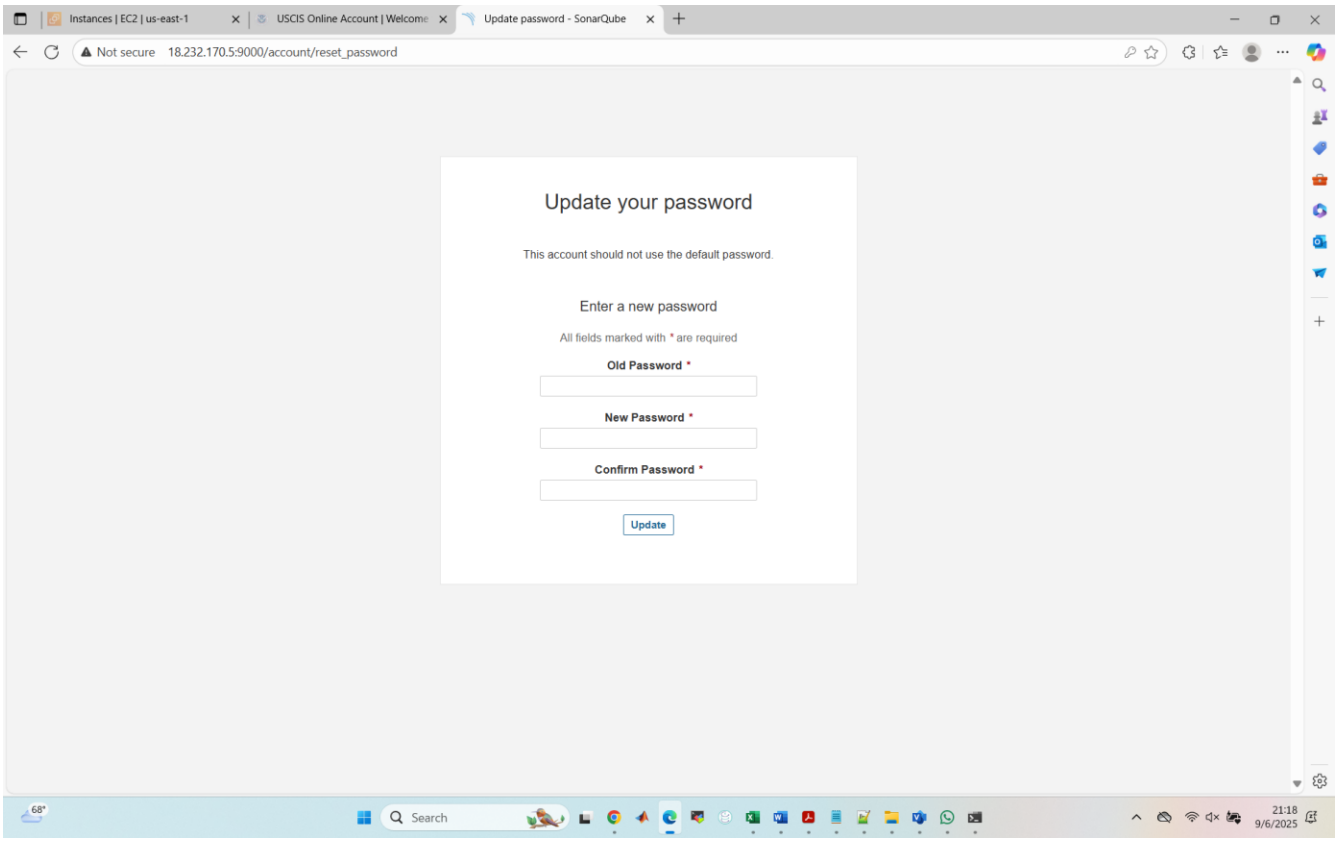
That is **http://18.232.170.5:9000**



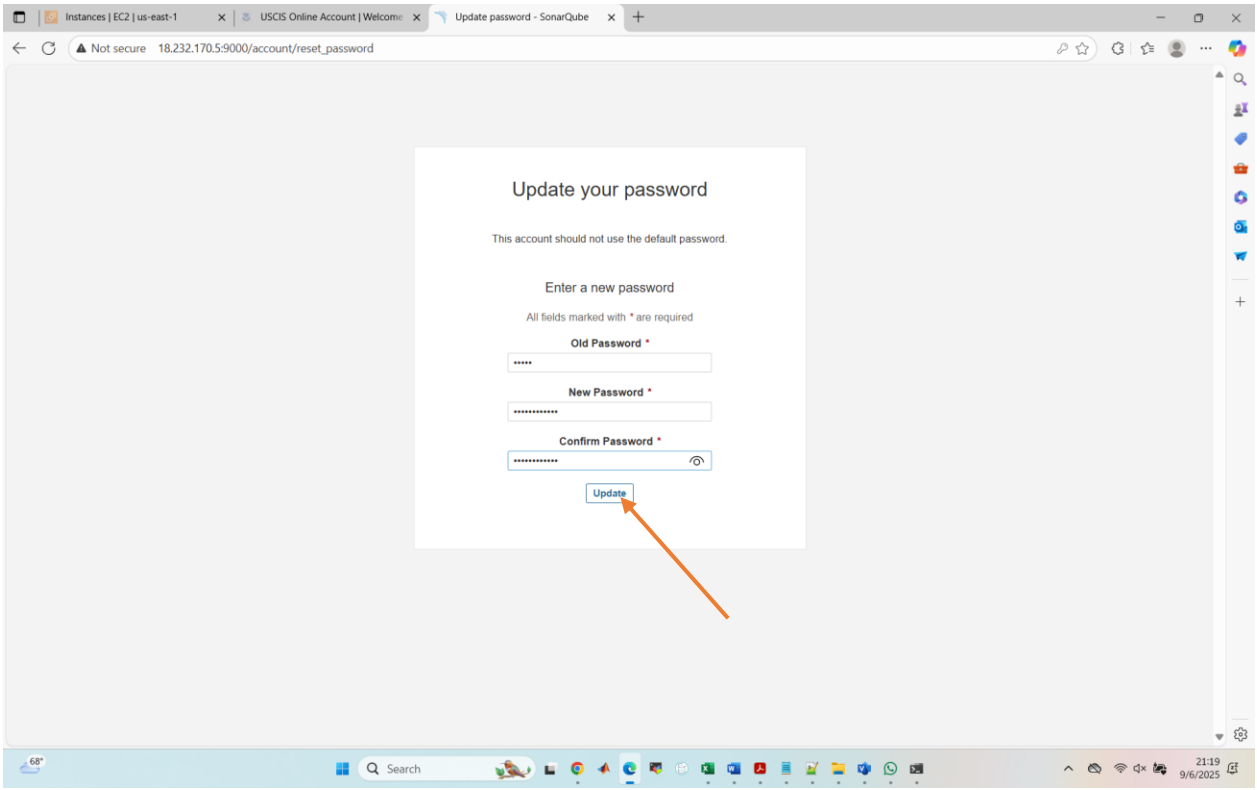
Note: the default username and password are **admin** and **admin** respectively.



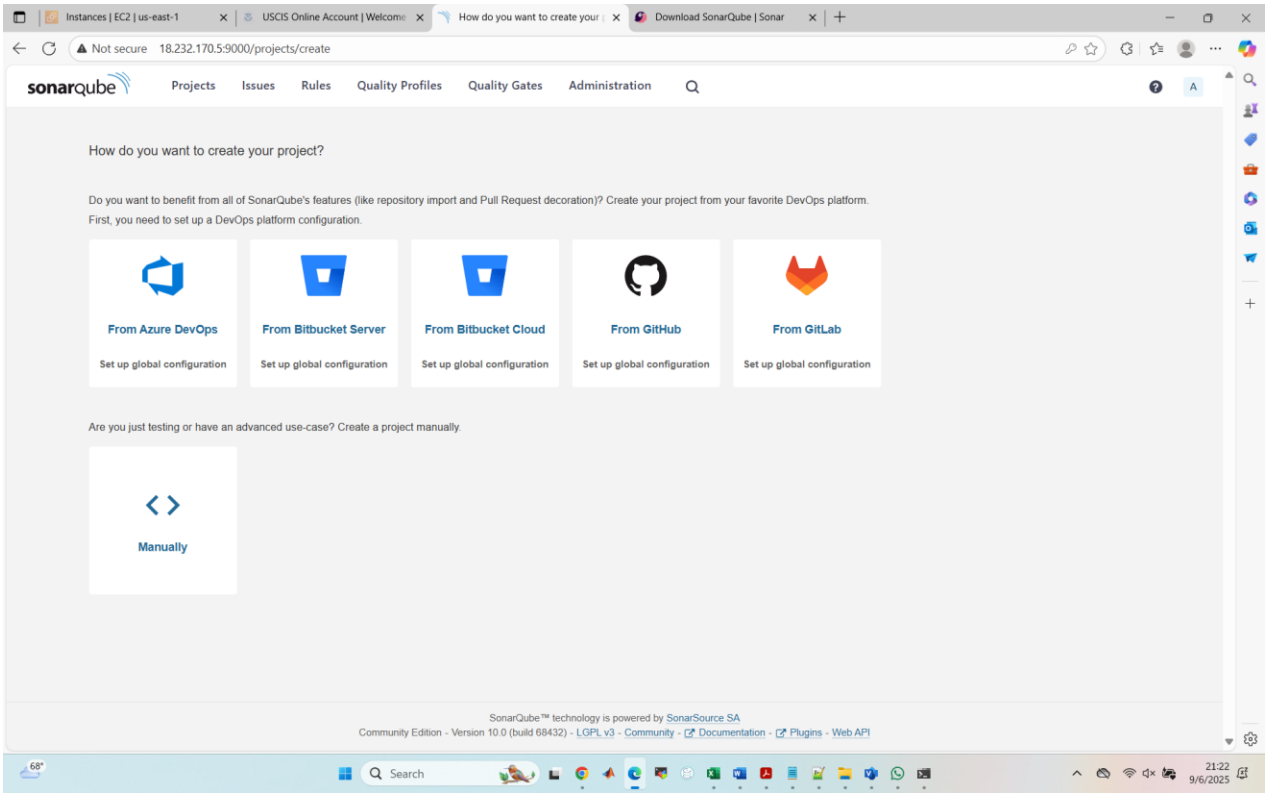
Click on “Login”. In the next step, SonarQube will prompt you to change your password. CHANGE THE PASSWORD.



Change the Old password with a New one.



Click on "**Update**"



We have successfully installed the SonarQube Community version 10.0 on AWS EC2 Ubuntu.

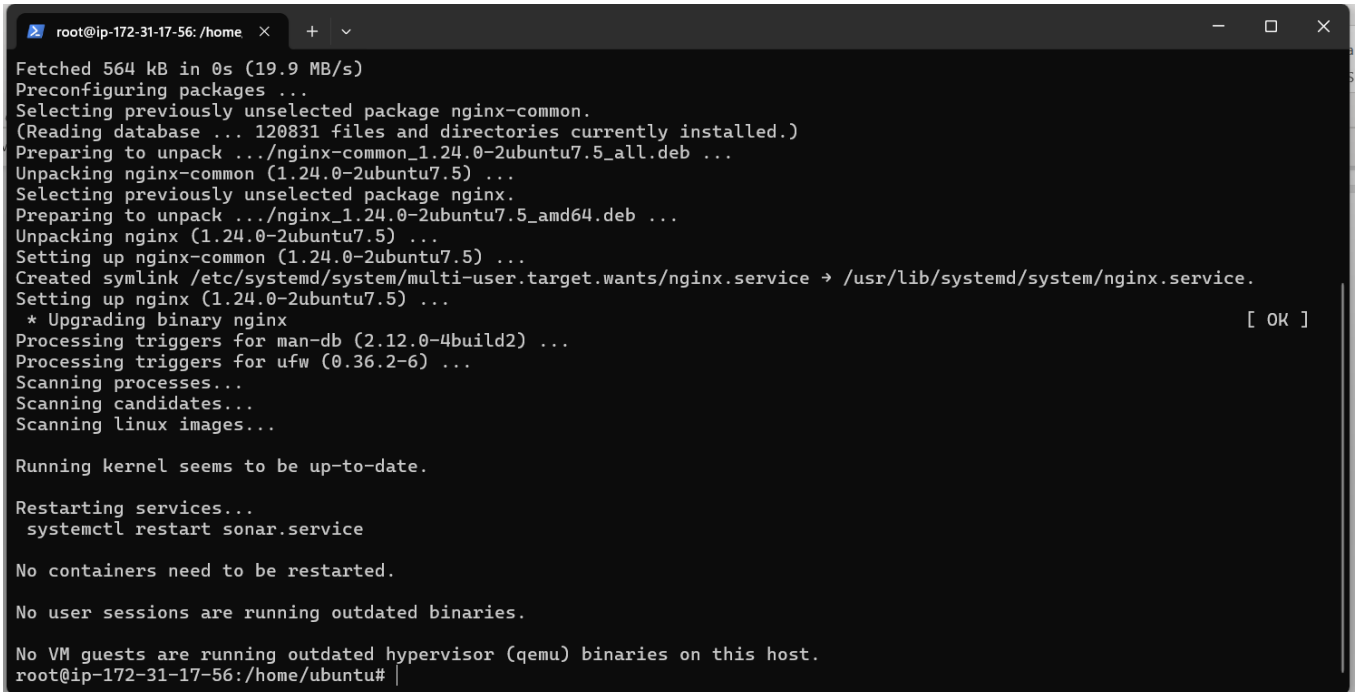
STEP 4: Configuring the Reverse Proxy using Nginx

Now we want a domain name to take over the IP of the server so that instead of hitting the IP we can use the domain name that is TLS enabled. Let's see how to do so.

I will be using the sub-domain name **sonarqube.sosoebot.org**. This is mine. Please use yours.

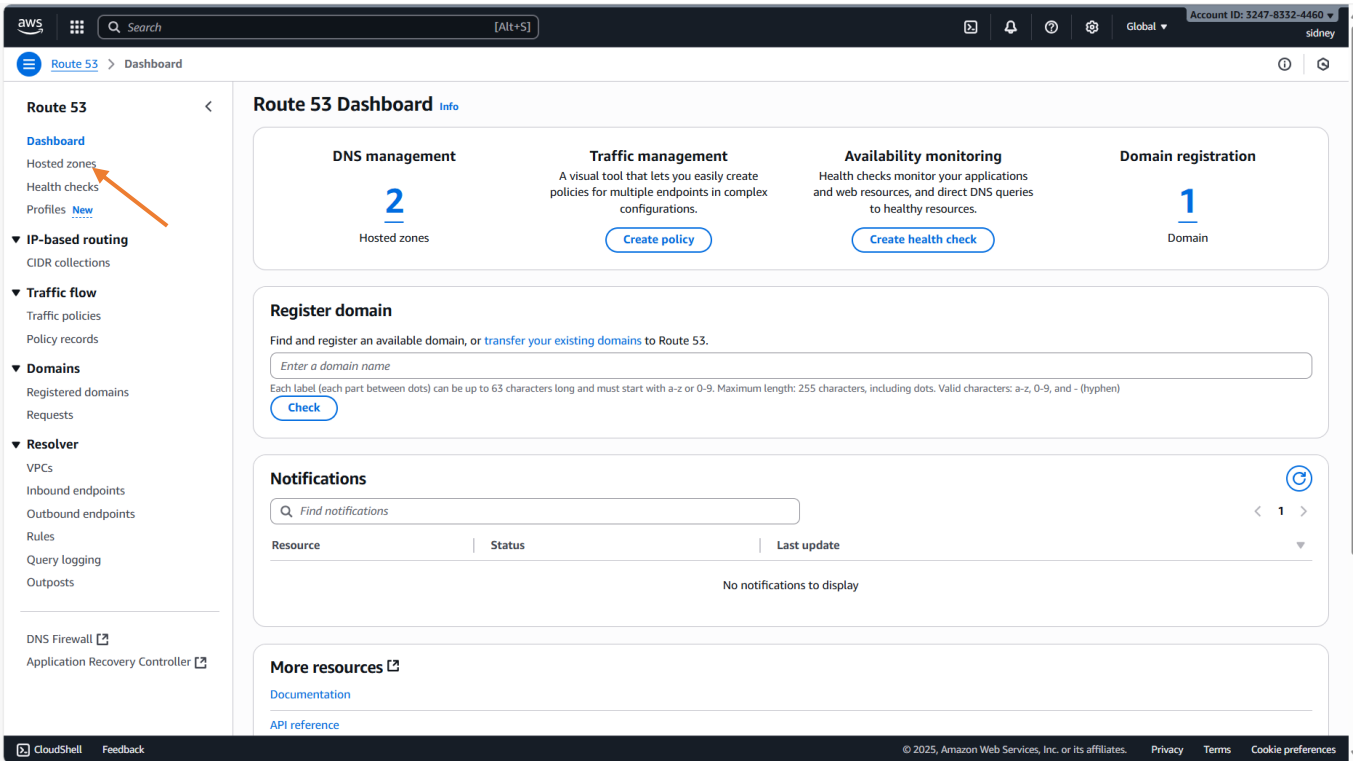
Part 1: Install Nginx

```
sudo apt install nginx -y
```

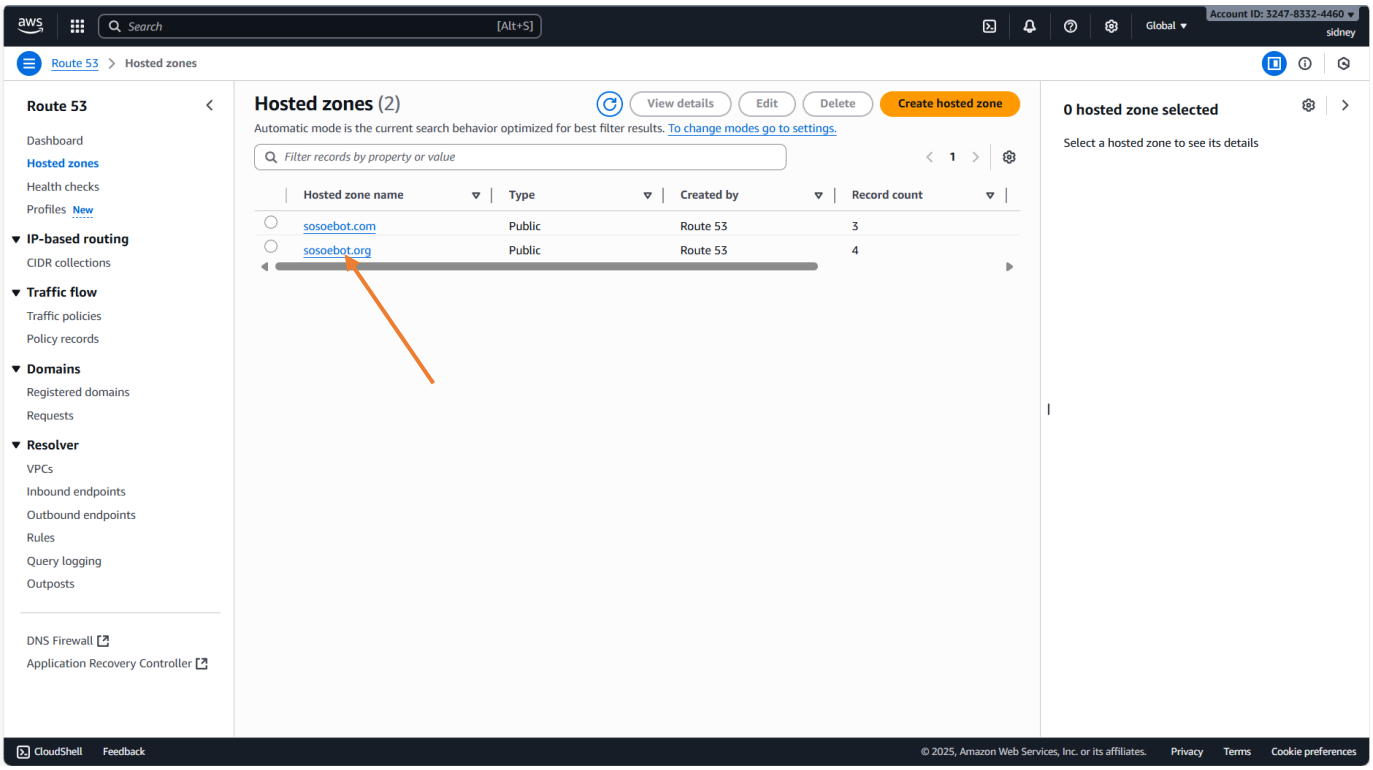


Part 2: Add the DNS records

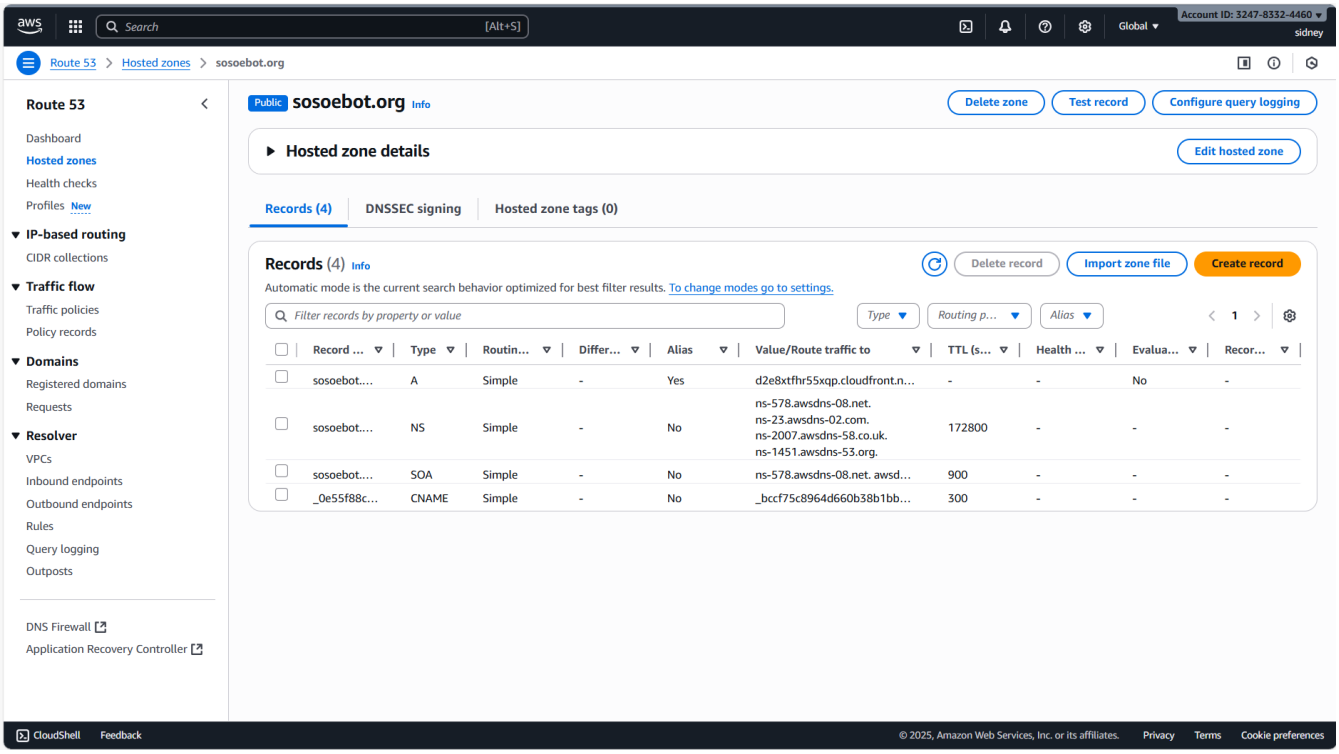
My domain is hosted on the AWS Route53, thus I will be adding the records in there. Go to AWS Console and search for "**Route53**"



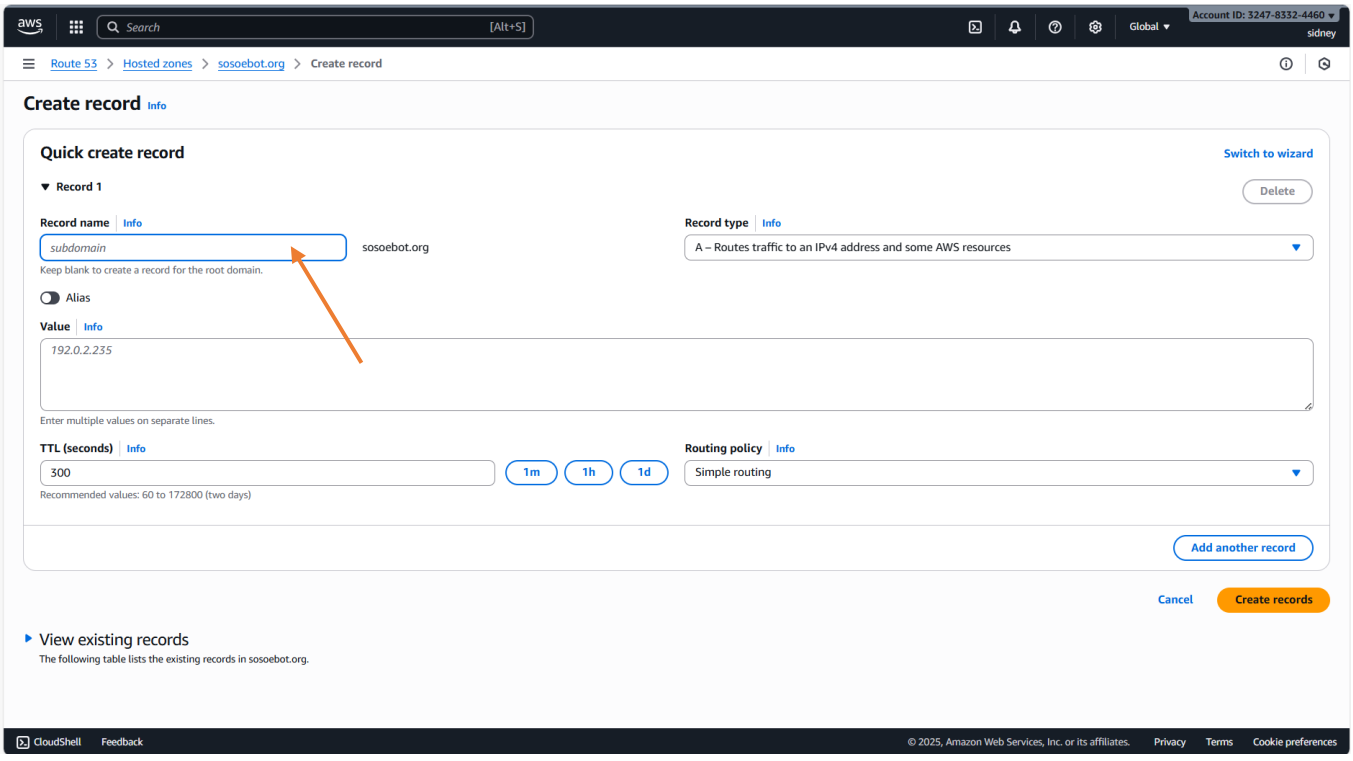
Adding the IP of the server on the DNS zone. Click on "**Hosted Zones**"



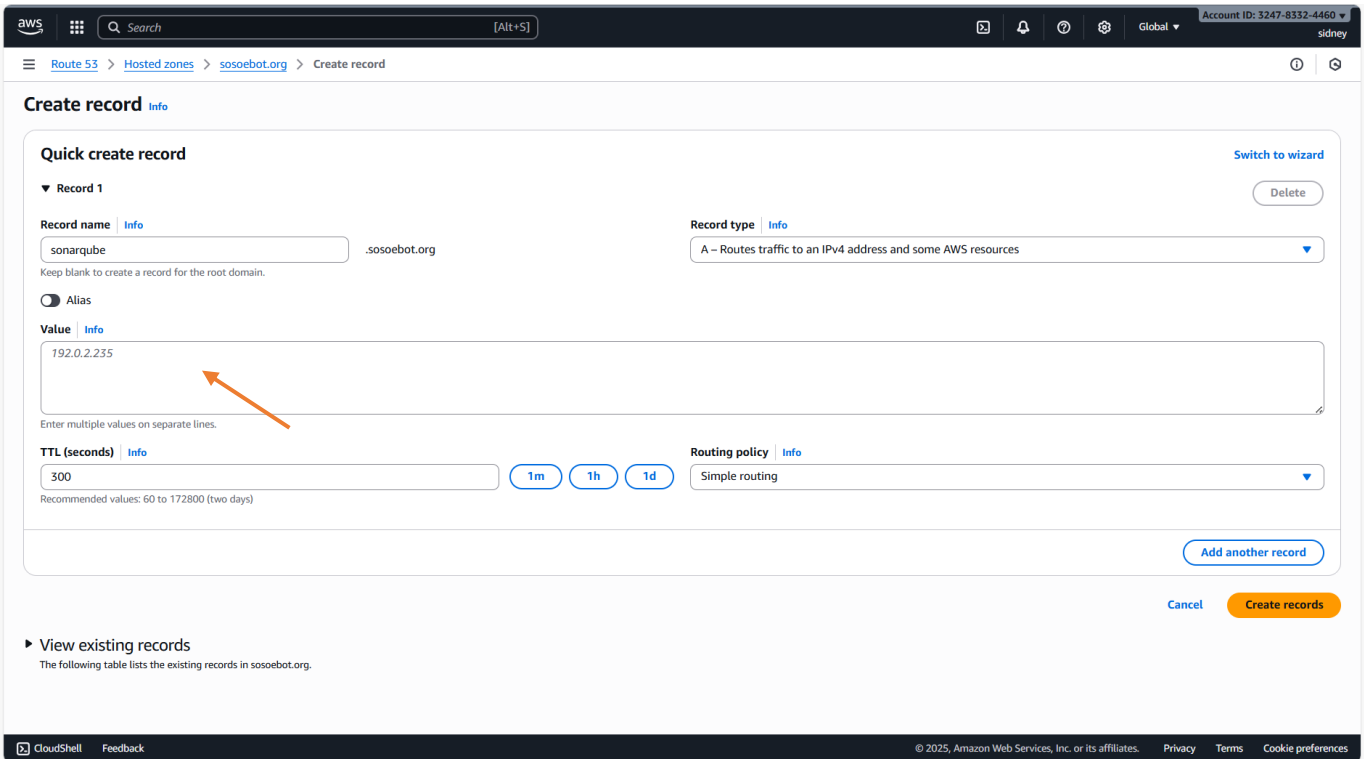
And now our domain sonarqube.sosoebot.org is ready to be used for the nginx configuration setup. Click on "**sosoebot.org**"



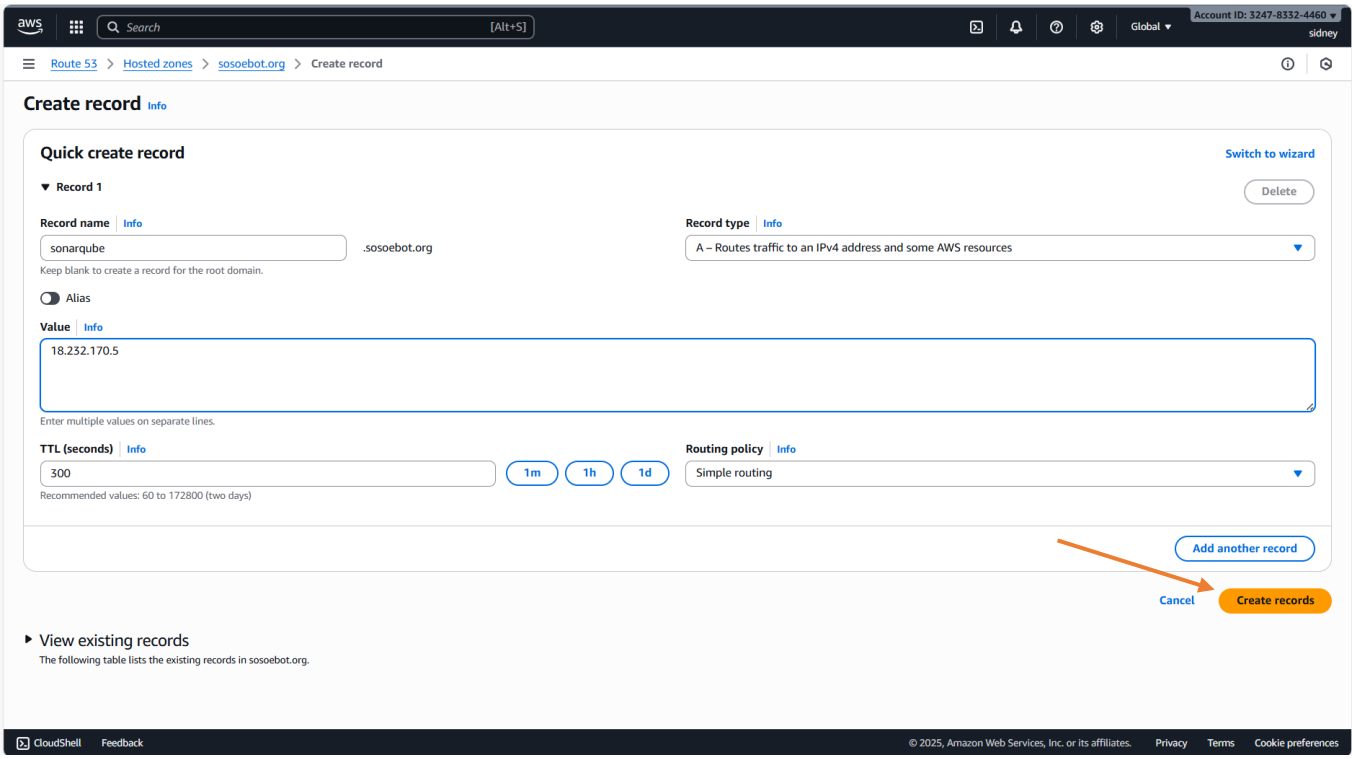
Click on “**Create Record**”



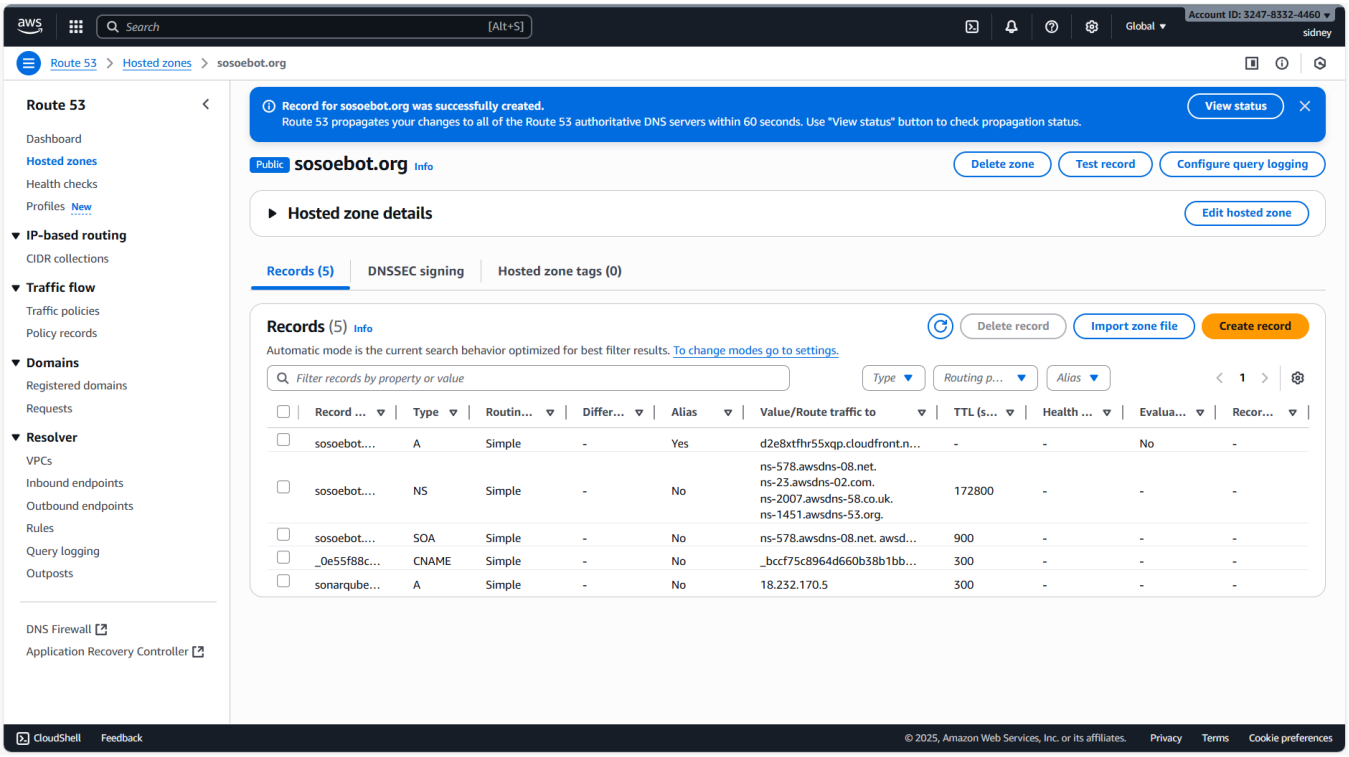
Enter “**sonarqube**”



Add the IP address

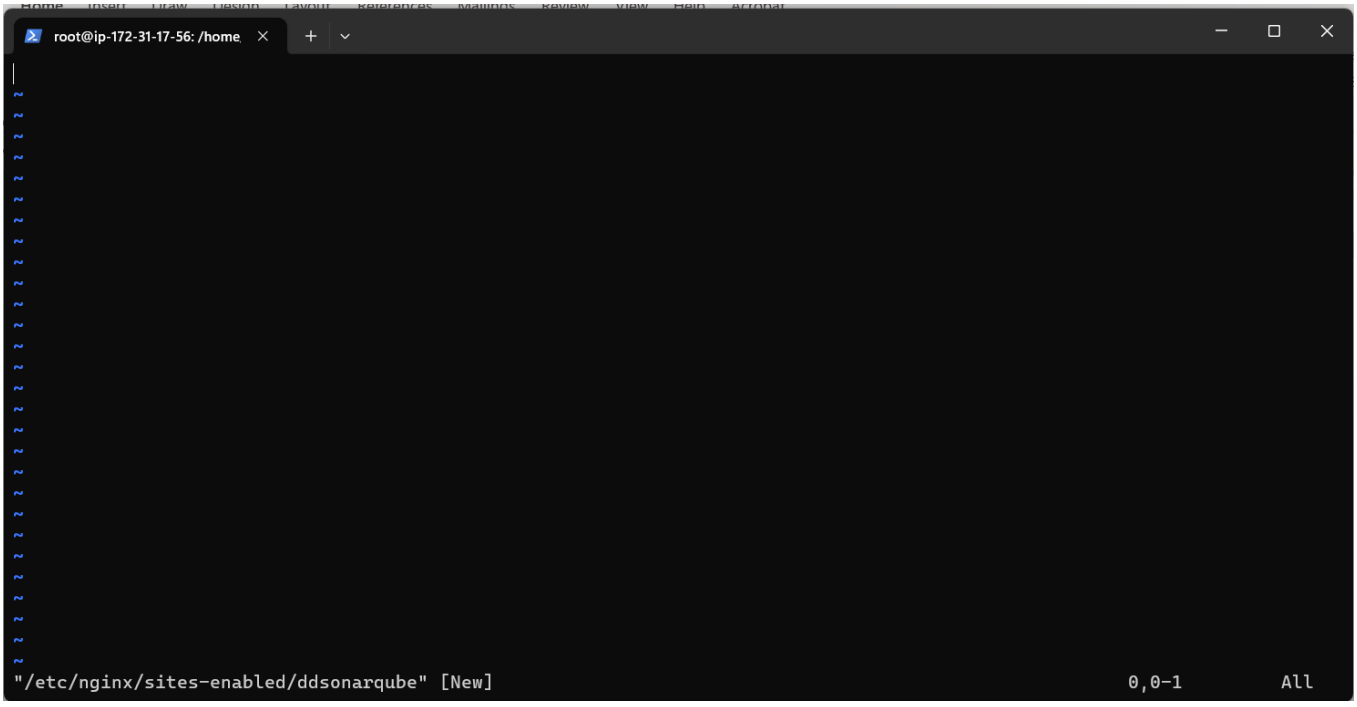


Click on “**Create records**”



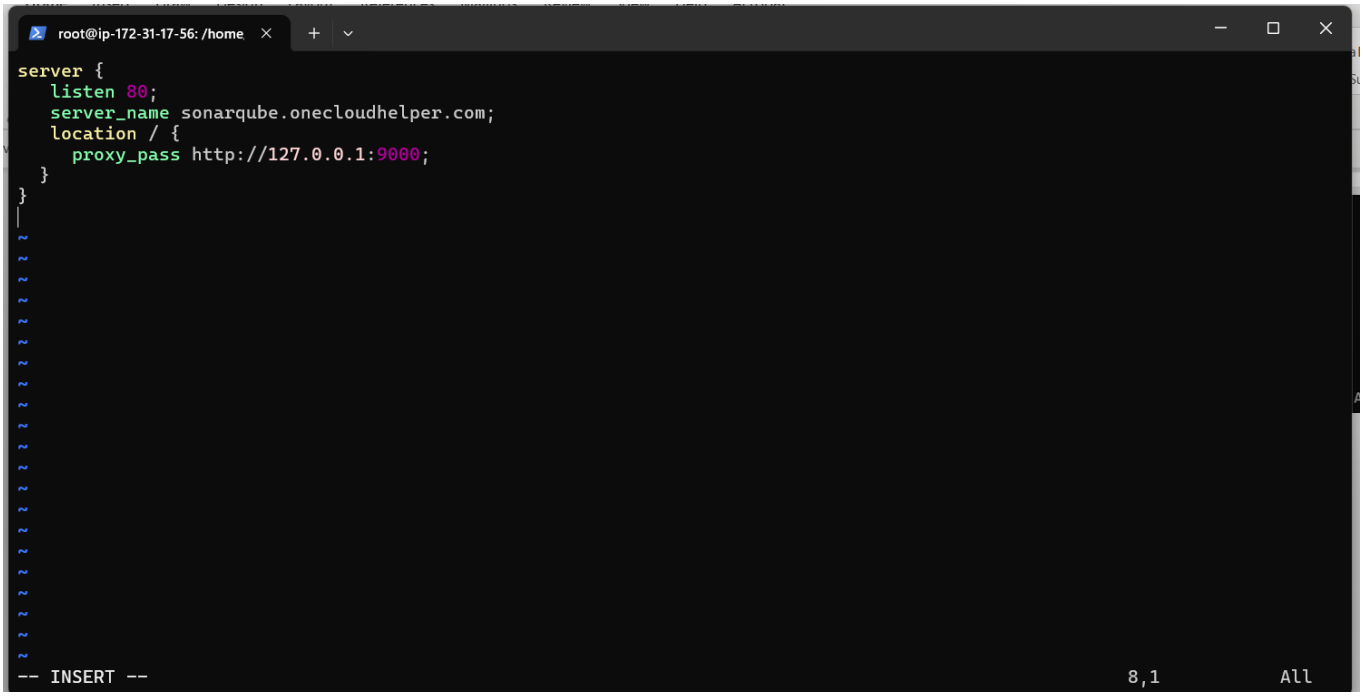
Part 3: Create a new Nginx configuration file for the site

```
sudo vim /etc/nginx/sites-enabled/ddsonarqube
```



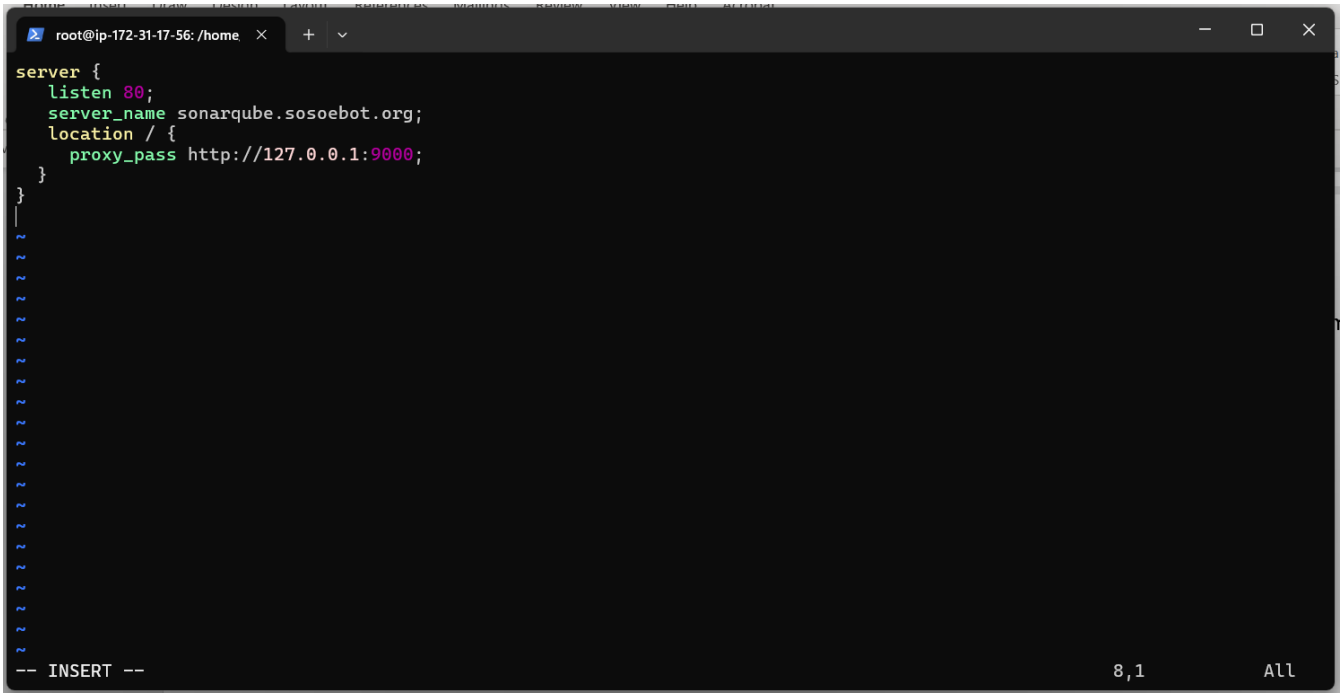
Add this configuration so that Nginx will route incoming traffic to SonarQube.

```
server {
    listen 80;
    server_name sonarqube.onecloudhelper.com;
    location / {
      proxy_pass http://127.0.0.1:9000;
   }
}
```

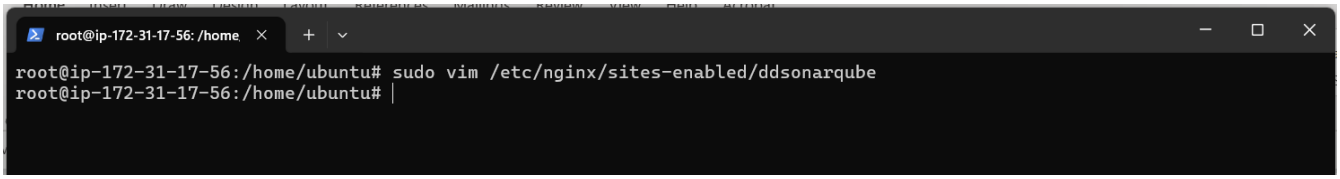


Here, replace the part **server_name** with **sonarqube.sosoebot.org**; that is your domain or sub-domain name.

```
server {
    listen 80;
    server_name sonarqube.sosoebot.org;
    location / {
      proxy_pass http://127.0.0.1:9000;
   }
}
```

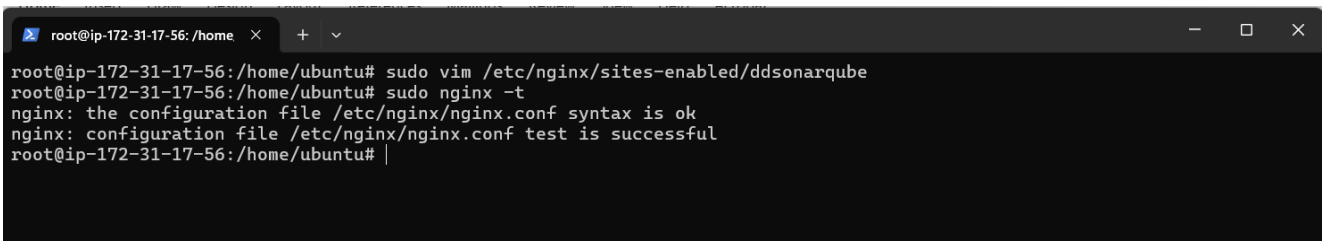


Save it by Pressing ESC, followed by :wq and press ENTER



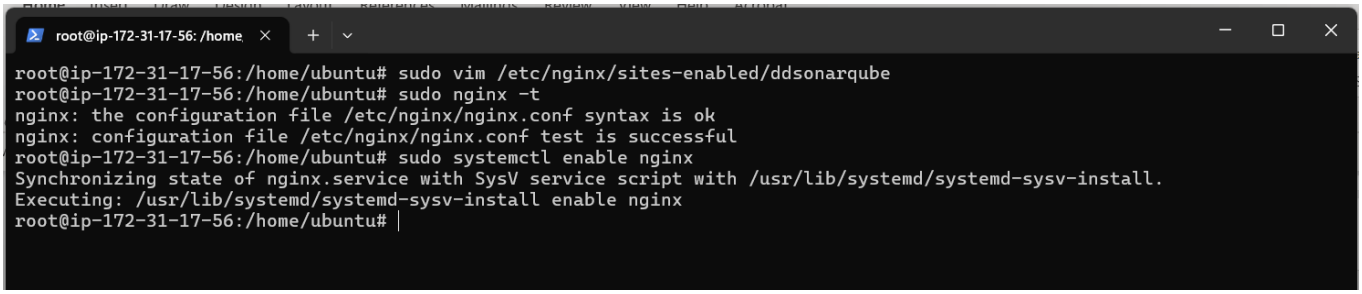
Part 4: Make sure your configuration file has no syntax errors.

`sudo nginx -t`



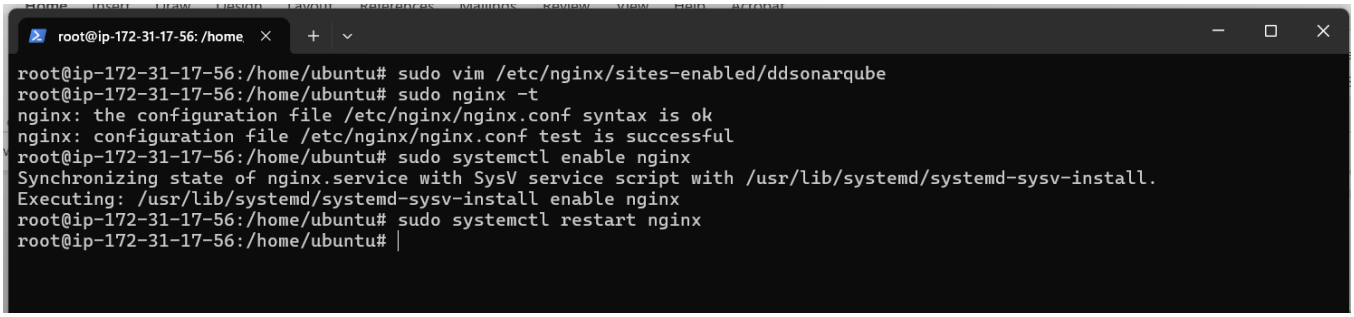
Part 5: Enable the Nginx site and restart Nginx

`sudo systemctl enable nginx`



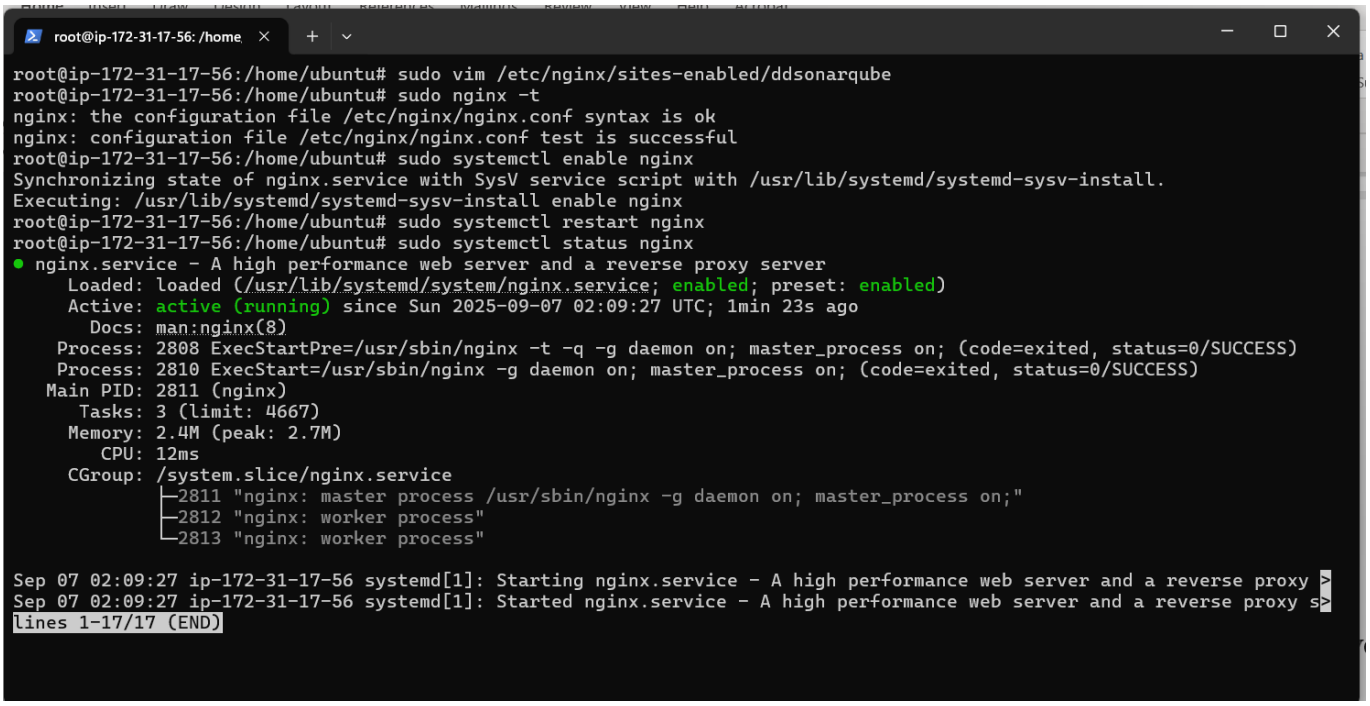
Restart the nginx server

```
sudo systemctl restart nginx
```



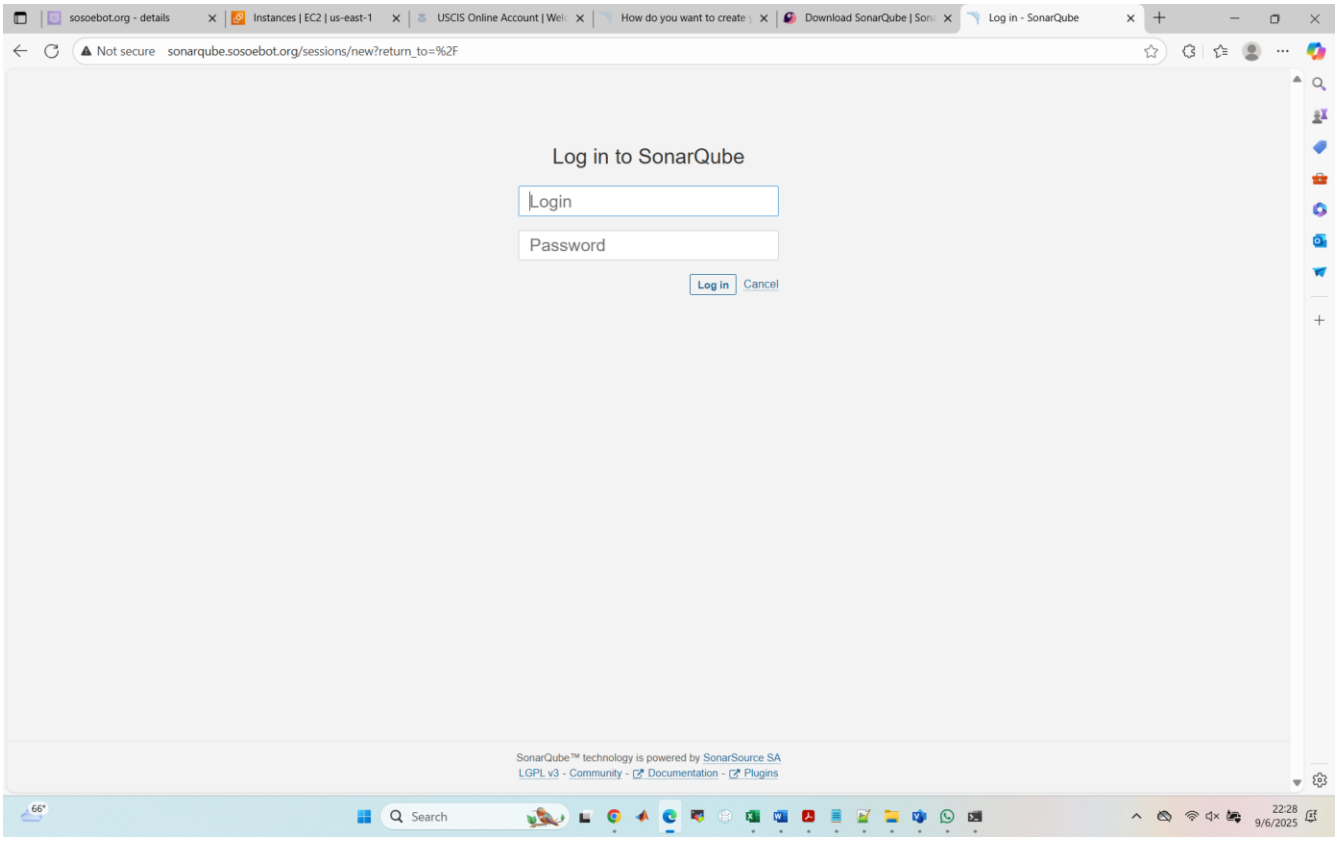
Check the status of the nginx server

```
sudo systemctl status nginx
```



Use the configured domain

You can now visit http://sonarqube.sosoebot.org in your web browser. You'll be greeted with the SonarQube web interface.



The configured domain sonarqube.onecloudhelper.com is working perfectly.

Wait, seems there is a problem ahead, let's solve it. There seems to be a problem ahead

Problem:

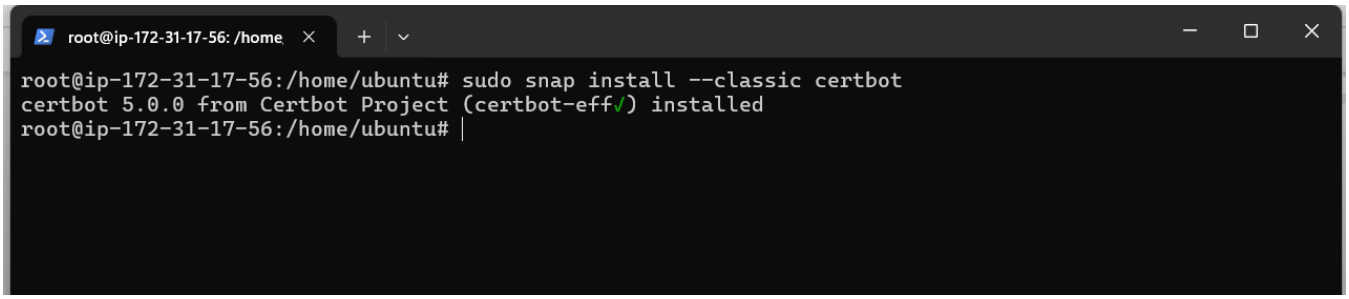
But the domain is working on HTTP protocol and we need to make it secure by making it work with the HTTPS protocol with SSL certificate.

STEP 5: Secure with SSL using Certbot

We will Install SSL certificate for maximum security. We will use the free SSL certificate provided by Certbot for our sub-domain sonarqube.sosoebot.org. Let's see how we will do this.

Part 1: Installing certbot

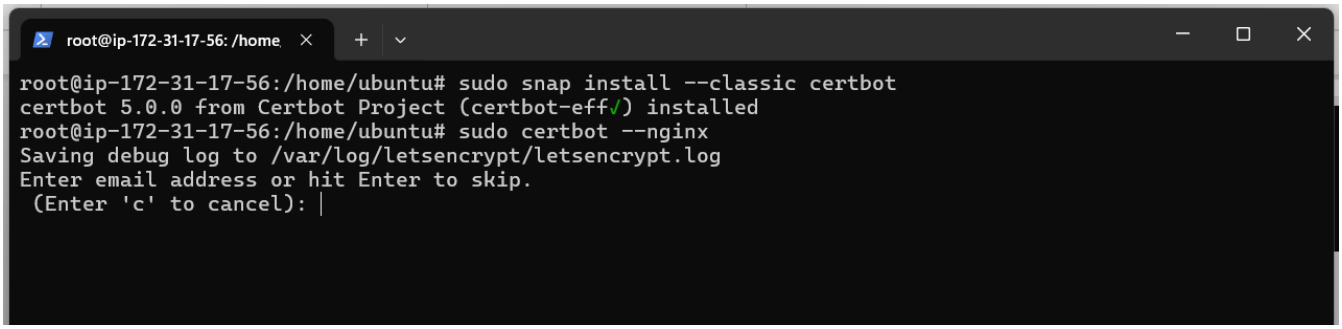
```
sudo snap install --classic certbot
```



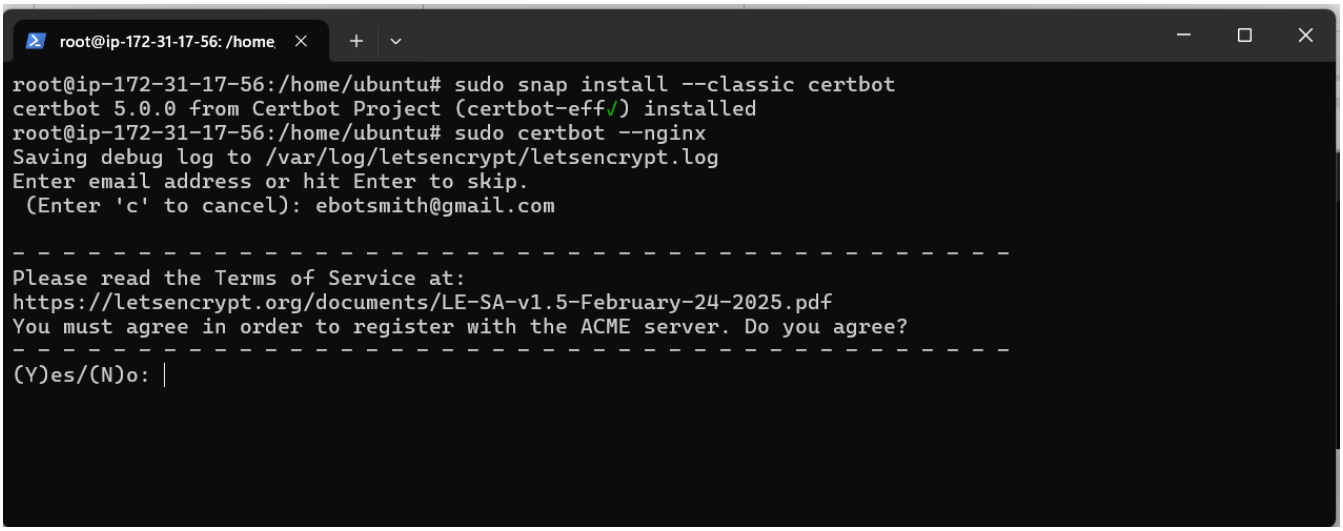
Part 2: Obtain and install an SSL certificate on the subdomain

Generating SSL certificate for the required domain. Please read the asked question carefully and answer them accordingly.

```
sudo certbot --nginx
```



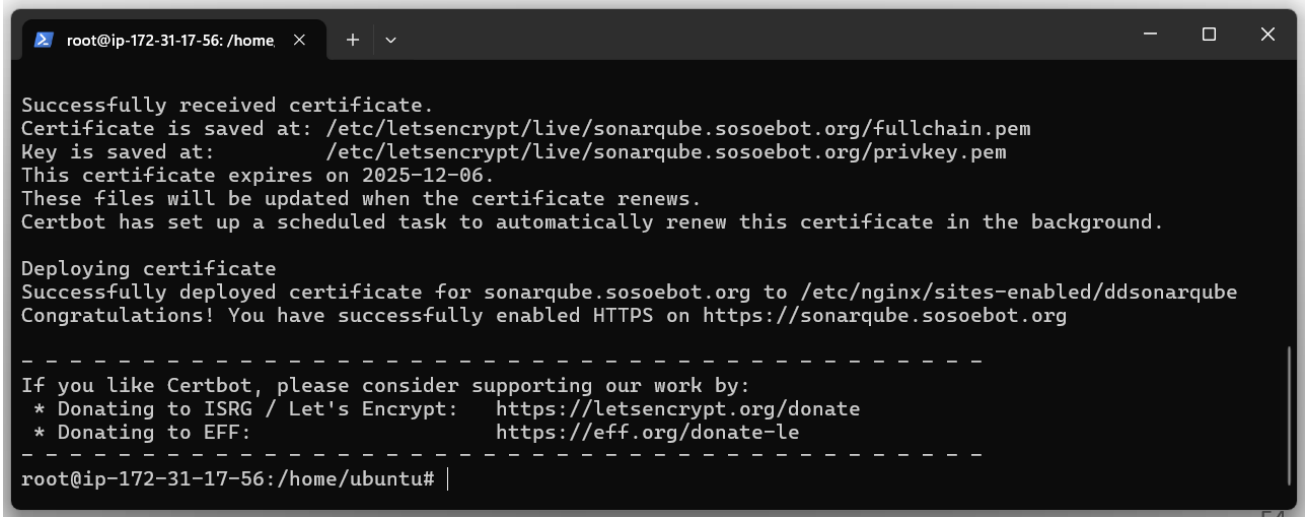
Enter your email address. I will enter “ebotsmith@gmail.com”



Enter “**Y**” and press ENTER

Enter “N” and press ENTER

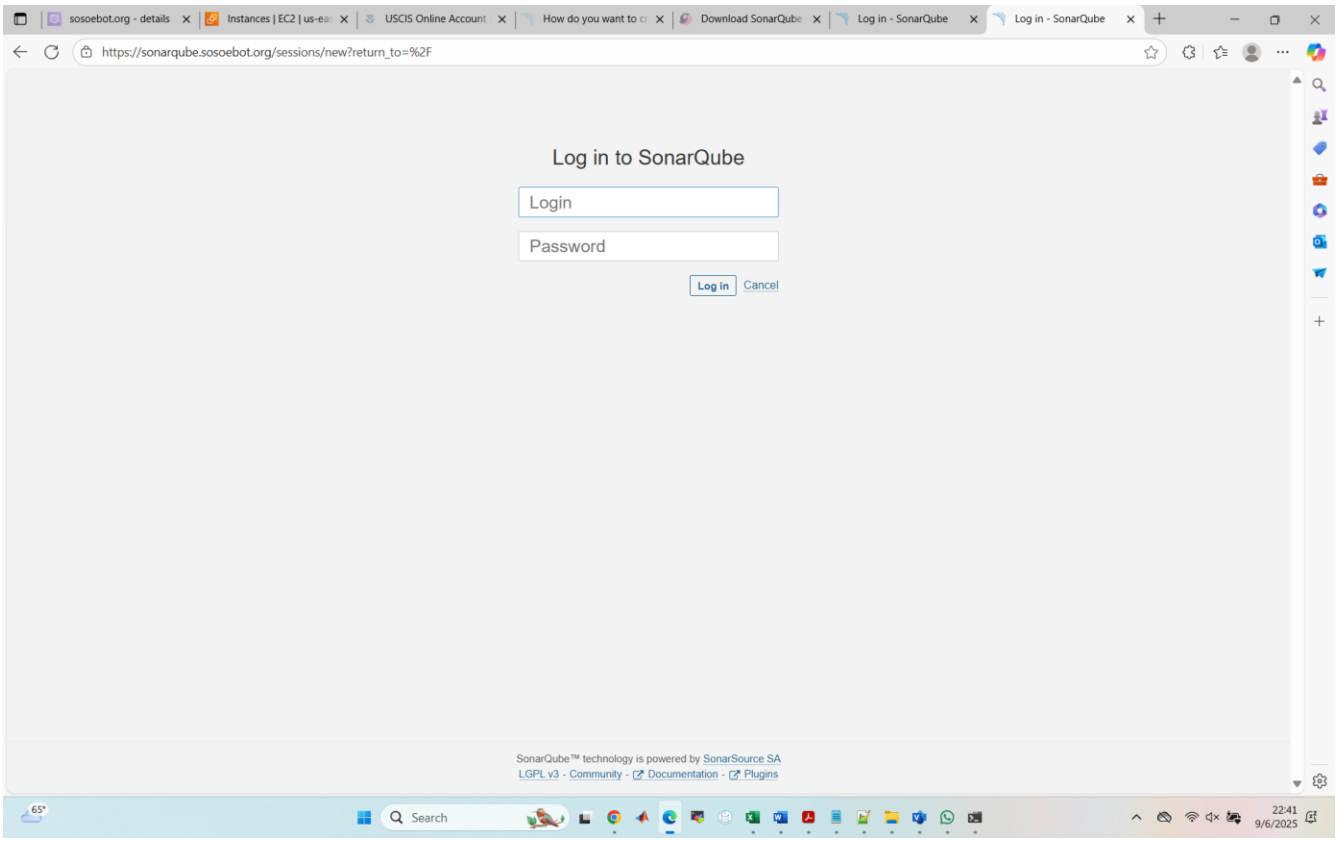
Enter “**1**” and press Enter



Success message that the SSL certificate has been deployed on the domain sonarqube.sosoebot.org

Part 3: Verify if Certificate deployed successfully

Now if we hit the domain **sonarqube.sosoebot.org**, it automatically redirects to HTTPS.



Now you can access SonarQube through your domain name or EC2 public IP address via Nginx (on port 80 or 443 since SSL has been configured.