

[Open in app](#)[Sign up](#)[Sign in](#)[Search](#)

How to install and configure SonarQube on AWS EC2 Ubuntu 22.04 and 20.04 (Full Setup)?

DeshDeepakDhobi (DD) · [Follow](#)

9 min read · May 28

[Listen](#)[Share](#)

Humans do make mistakes and when it comes to mistakes in the codes it is too much to manage and handle and there comes SonarQube to rescue.

SonarQube is a code quality assurance tool used for continuous inspection of code quality to perform automatic reviews with analysis of code to detect bugs and code smells on multiple programming languages and generate an analysis report to ensure code reliability.



Let's start the installation of the latest version (10.0) of SonarQube on Ubuntu 22.04 LTS.

Note: The same steps also work on the AWS EC2 Ubuntu 20.04.

Server Specification (Please choose as per your requirement):

OS = Ubuntu 22.04 LTS

CPU: 2 vCPU

RAM: 4 GB

Storage: 20 GB

And view the following documentation for the Prerequisites and Hardware recommendations.

Prerequisites and overview

The prerequisites for installing SonarQube.

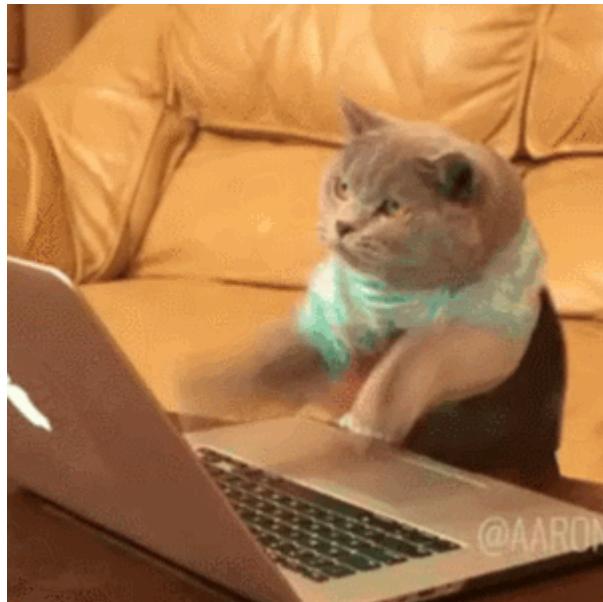
docs.sonarqube.org

Security Ports to open for the AWS EC2 server:

- i) HTTP = 80
- ii) HTTPS = 443
- iii) SSH = 22 (open to only required IP)
- iv) Custom port = 9000

Type	Protocol	Port range	Source
HTTPS	TCP	443	0.0.0.0/0
Custom TCP	TCP	9000	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	27.34.96.72/32

Let's Start Intsalling and configuring things.



Let's start installing things.

1) SSH into the AWS EC2 Ubuntu Server

```
ssh -i [keyname.pem] [OSname]@[PublicIP_of_EC2Server]
```

For example (for me its):

```
ssh -i helloworld.pem ubuntu@55.63.86.254
```

2) Let's rejuvenate our Ubuntu server

```
sudo apt update
```

```
ubuntu@ip-172-31-20-61:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
0% [5 Packages 14.1 MB/14.1 MB 100%]
```

Update the server

```
sudo apt upgrade -y
```

```
ubuntu@ip-172-31-20-61:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  apt binutils binutils-common binutils-x86-64-linux-gnu ca-certificates dpkg libbinutils libctf-nobfd0 libctf0 libncurses6
  libncursesw6 libtinfo6 mdadm mokutil ncurses-base ncurses-bin ncurses-term python3-apport python3-problem-report
  python3-software-properties software-properties-common
21 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
13 standard LTS security updates
Need to get 6419 kB of archives.
After this operation, 34.8 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dpkg amd64 1.21.1ubuntu2.2 [1239 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-bin amd64 6.3-2ubuntu0.1 [184 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-base all 6.3-2ubuntu0.1 [20.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libncursesw6 amd64 6.3-2ubuntu0.1 [147 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libncurses6 amd64 6.3-2ubuntu0.1 [111 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtinfo6 amd64 6.3-2ubuntu0.1 [105 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ca-certificates all 20230311ubuntu0.22.04.1 [155 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-problem-report all 2.20.11-0ubuntu82.5 [11.1 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-apport all 2.20.11-0ubuntu82.5 [88.9 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apport all 2.20.11-0ubuntu82.5 [133 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libctf0 amd64 2.38-4ubuntu2.2 [103 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libctf-nobfd0 amd64 2.38-4ubuntu2.2 [107 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 binutils-x86-64-linux-gnu amd64 2.38-4ubuntu2.2 [2328 kB]
]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libbinutils amd64 2.38-4ubuntu2.2 [660 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 binutils amd64 2.38-4ubuntu2.2 [3186 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 binutils-common amd64 2.38-4ubuntu2.2 [222 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mdadm amd64 4.2-0ubuntu2 [464 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mokutil amd64 0.6.0-2~22.04.1 [27.2 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-term all 6.3-2ubuntu0.1 [267 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 software-properties-common all 0.99.22.7 [14.1 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-software-properties all 0.99.22.7 [28.8 kB]
Fetched 6419 kB in 0s (33.1 MB/s)
Preconfiguring packages ...
```

Upgrade the server

3) Install OpenJDK 17

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

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

```
ubuntu@ip-172-31-20-61:~$ sudo apt install -y openjdk-17-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
adwaita-icon-theme alsamixer-conf alsound-conf at-spi2-core ca-certificates-java
dconf-gsettings-backend dconf-service fontconfig fontconfig-config fonts-dejavu-core
fonts-dejavu-extra gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme
humanity-icon-theme java-common libasound2 libasound2-data libatk-bridge2.0-0
libatk-wrapper-java libatk-wrapper-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
libavahi-client3 libavahi-common-data libavahi-common3 libcairo-gobject2 libcairo2
libcurl2 libdatrie1 libdconf1 libdeflate0 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2
libdrm-radeon1 libfontconfig1 libfontenc1 libgail-common libgail18 libgd-pixbuf-2.0-0
libgd-pixbuf2.0-bin libgd-pixbuf2.0-common libgif7 libgl1 libgl1-amber-dri
libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0
libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice-dev libice6 libjbig0 libjpeg-turbo8
libjpeg8 liblcms2-2 libllvm15 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0
libpcaccess0 libpcslite1 libpixman-1-0 libpthread-stubs0-dev librsvg2-2 librsvg2-common
libsensors-config libsensors5 libsm-dev libsm6 libthai-data libthai0 libtiff5 libwebp7
libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
libxcb-present0 libxcb-render0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0
libxcb1-dev libxcomposite1 libxcursor1 libxdamage1 libxdmcp-dev libxfixes3 libxft2 libxi6
libxinerama1 libxkbfile1 libxml2 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 openjdk-17-jdk-headless openjdk-17-jre
openjdk-17-jre-headless session-migration ubuntu-mono x11-common x11proto-dev
xorg-sgml-doctools xtrans-dev
suggested packages:
```

Installing JDK 17

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

```
java -version
```

```
ubuntu@ip-172-31-20-61:~$ java -version
openjdk version "17.0.7" 2023-04-18
OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu122.04.2)
OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu122.04.2, mixed mode, sharing)
ubuntu@ip-172-31-20-61:~$
```

Checking the Java version

4) Install and Configure PostgreSQL

i) 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'
```

```
ubuntu@ip-172-31-20-61:~$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" /etc/apt/sources.list.d/pgdg.list'
deb http://apt.postgresql.org/pub/repos/apt/ jammy-pgdg main /etc/apt/sources.list.d/pgdg.list
ubuntu@ip-172-31-20-61:~$
```

ii) Add the PostgreSQL signing key.

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

iii) Install PostgreSQL.

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

```
ubuntu@ip-172-31-20-61:~$ wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8))
.
OK
ubuntu@ip-172-31-20-61:~$ sudo apt install postgresql postgresql-contrib -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl postgresql-14 postgresql-client-14 postgresql-client-common
  postgresql-common ssl-cert sysstat
Suggested packages:
  postgresql-doc postgresql-doc-14 isag
The following NEW packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl postgresql postgresql-14 postgresql-client-14
  postgresql-client-common postgresql-common postgresql-contrib ssl-cert sysstat
0 upgraded, 14 newly installed, 0 to remove and 0 not upgraded.
Need to get 42.4 MB of archives.
After this operation, 161 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libcommon-sense-perl amd64 3.75-2build1 [21.1 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libjson-perl all 4.04000-1 [81.8 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libtypes-serialiser-perl
```

Installing PostgreSQL and its dependencies

iv) Enable the database server to start automatically on reboot.

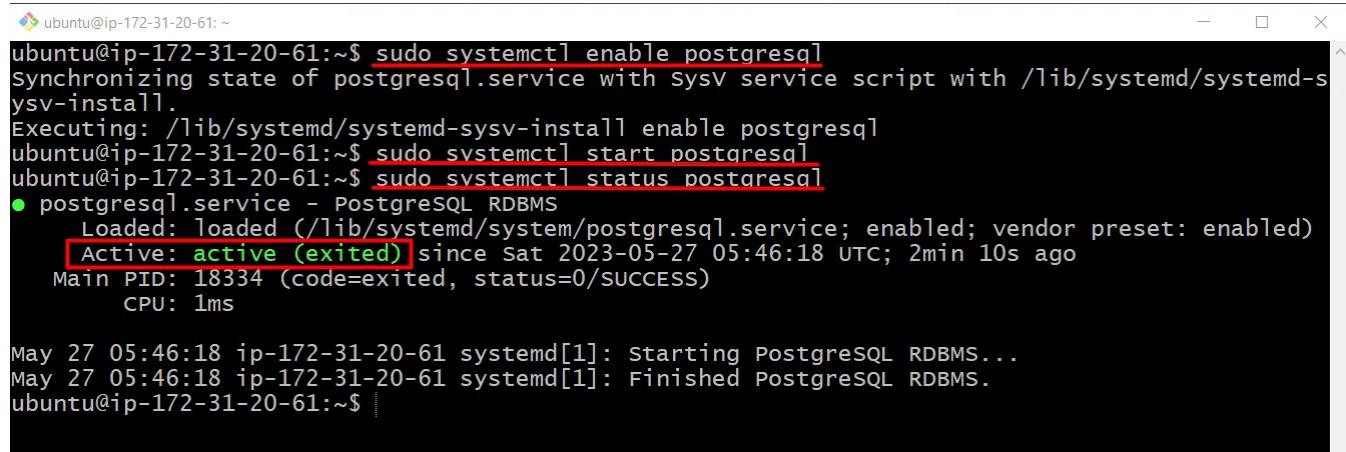
```
sudo systemctl enable postgresql
```

v) Start the database server.

```
sudo systemctl start postgresql
```

vi) Check the status of the database server

```
sudo systemctl status postgresql
```



A terminal window showing the status of the PostgreSQL service. The command `sudo systemctl status postgresql` is run, displaying the service status. The output shows the service is active (exited) since May 27 2023, with a main PID of 18334 and 1ms CPU usage. Log messages indicate the service is starting and finished.

```
ubuntu@ip-172-31-20-61:~$ sudo systemctl enable postgresql
Synchronizing state of postgresql.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
ubuntu@ip-172-31-20-61:~$ sudo systemctl start postgresql
ubuntu@ip-172-31-20-61:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
    Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
    Active: active (exited) since Sat 2023-05-27 05:46:18 UTC; 2min 10s ago
      Main PID: 18334 (code=exited, status=0/SUCCESS)
        CPU: 1ms

May 27 05:46:18 ip-172-31-20-61 systemd[1]: Starting PostgreSQL RDBMS...
May 27 05:46:18 ip-172-31-20-61 systemd[1]: Finished PostgreSQL RDBMS.
ubuntu@ip-172-31-20-61:~$
```

Postgresql status check

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

```
psql --version
```

```
ubuntu@ip-172-31-20-61:~$ psql --version
psql (PostgreSQL) 14.8 (Ubuntu 14.8-0ubuntu0.22.04.1)
ubuntu@ip-172-31-20-61:~$
```

Postgresql version check

viii) Switch to the Postgres user.

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

ix) 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
```

x) Log in to PostgreSQL.

```
psql
```

```
ubuntu@ip-172-31-20-61:~$ sudo -i -u postgres
postgres@ip-172-31-20-61:~$ createuser ddsonar
postgres@ip-172-31-20-61:~$ psql
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
Type "help" for help.
```

```
postgres=#
```

SSH into the Postgresql

xi) 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 'mwd#2%#!#%rgs';
```

xii) 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;
```

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

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

```
postgres=# CREATE DATABASE ddsonarqube OWNER ddsonar;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE ddsonarqube TO ddsonar;
GRANT
postgres=#

```

Create a database and grant it ALL PRIVILEGES.

xiv) Let's check the created user and the database.

a) To check the created database

```
\l
```

```
postgres-# \l
              List of databases
   Name    |  Owner   | Encoding | Collate | Ctype | Access privileges
+-----+-----+
ddsonarqube | ddsonar | UTF8     | C.UTF-8 | C.UTF-8 | =Tc/ddsonar
               |          |          |          |          | ddsonar=CTc/ddsonar
+-----+
postgres     | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres
template0    | postgres | UTF8     | C.UTF-8 | C.UTF-8 | postgres=CTc/postgres
template1    | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres
               |          |          |          |          | postgres=CTc/postgres
(4 rows)
postgres-#
```

Checking the created Database name

b) To check the created database user

```
\du
```

```
postgres-# \du
              List of roles
 Role name | Attributes | Member of
+-----+-----+
ddsonar    |           | {}
postgres    | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

checking the created database user

xv) Exit PostgreSQL.

```
\q
```

xvi) Return to your non-root sudo user account.

```
exit
```

```
postgres-# \q
postgres@ip-172-31-20-61:~$ exit
logout
ubuntu@ip-172-31-20-61:~$ |
```

Returning to the non-root user

5) 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.
```

```
ubuntu@ip-172-31-20-61:~$ sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.0.0.68432.zip
--2023-05-27 06:58:03-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.0.0.68432.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.191.75, 99.84.191.87, 99.84.191.23, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.191.75|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 352909963 (337M) [binary/octet-stream]
Saving to: 'sonarqube-10.0.0.68432.zip'

sonarqube-10.0.0.68432. 100%[=====] 336.56M 119MB/s in 2.8s
2023-05-27 06:58:06 (119 MB/s) - 'sonarqube-10.0.0.68432.zip' saved [352909963/352909963]
ubuntu@ip-172-31-20-61:~$ ls
sonarqube-10.0.0.68432.zip
ubuntu@ip-172-31-20-61:~$
```

Downloading the latest version (10.0) of SonarQube distribution files.

iii) Unzip the downloaded file.

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

iv) Move the unzipped files to /opt/sonarqube directory

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

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

6) 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.

i) Create a sonar group.

```
sudo groupadd ddsonar
```

- ii) Create a sonar user and set `/opt/sonarqube` as the home directory.

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

- iii) Grant the sonar user access to the `/opt/sonarqube` directory.

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

```
ubuntu@ip-172-31-20-61:~$ sudo groupadd ddsonar
ubuntu@ip-172-31-20-61:~$ sudo useradd -d /opt/sonarqube -g ddsonar ddsonar
ubuntu@ip-172-31-20-61:~$ sudo chown ddsonar:ddsonar /opt/sonarqube -R
ubuntu@ip-172-31-20-61:~$
```

Creating sonar user and group

7) Configure SonarQube

- i) Edit the SonarQube configuration file.

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

- a) Find the following lines:

`#sonar.jdbc.username=`

`#sonar.jdbc.password=`

- b) Uncomment the lines, and add the database user and Database password you created in *Step 4 (xi and xii)*. For me, it's:

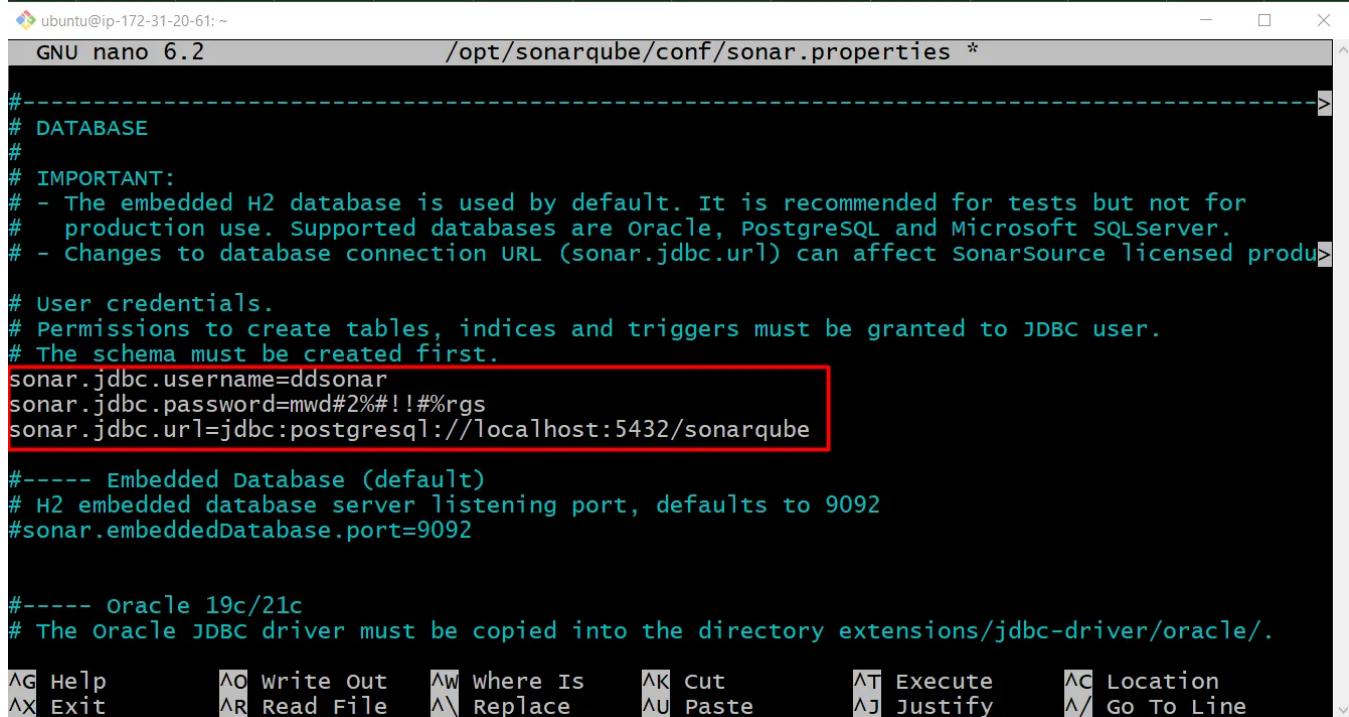
`sonar.jdbc.username=ddsonar`

```
sonar.jdbc.password=mwd#2%#!%rgs
```

c) 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.



```
ubuntu@ip-172-31-20-61: ~
GNU nano 6.2          /opt/sonarqube/conf/sonar.properties *

#
# DATABASE
#
# IMPORTANT:
# - The embedded H2 database is used by default. It is recommended for tests but not for
#   production use. Supported databases are Oracle, PostgreSQL and Microsoft SQLServer.
# - Changes to database connection URL (sonar.jdbc.url) can affect SonarSource licensed produ>
# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC user.
# The schema must be created first.
sonar.jdbc.username=ddsonar
sonar.jdbc.password=mwd#2%#!%rgs
sonar.jdbc.url=jdbc:postgresql://localhost:5432/ddsonarqube

----- Embedded Database (default)
# H2 embedded database server listening port, defaults to 9092
#sonar.embeddedDatabase.port=9092

----- Oracle 19c/21c
# The oracle JDBC driver must be copied into the directory extensions/jdbc-driver/oracle/.

^G Help      ^O Write Out    ^W Where Is     ^K Cut        ^T Execute     ^C Location
^X Exit      ^R Read File    ^M Replace     ^U Paste      ^J Justify     ^L Go To Line
```

Adding the required line of details in the file sonar.properties.

d) Save and exit the file.

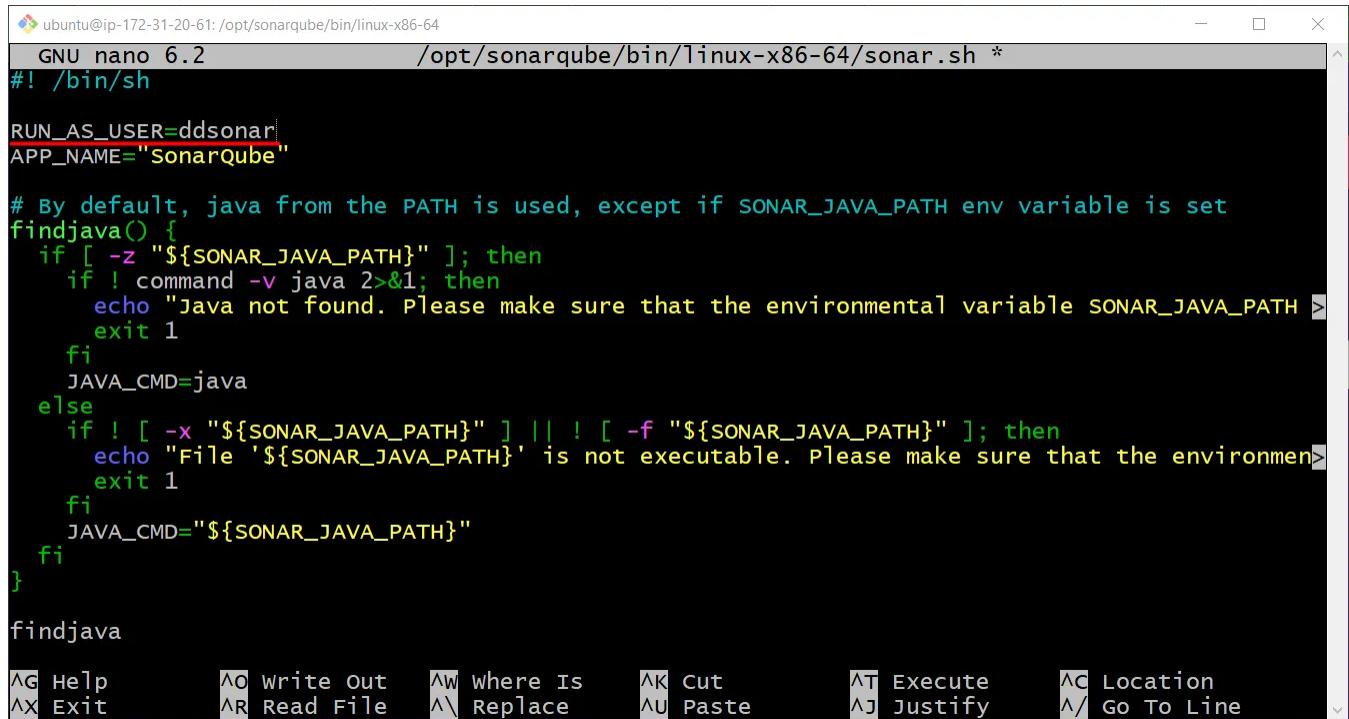
ii) Edit the sonar script file.

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

a) 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).



```

ubuntu@ip-172-31-20-61:/opt/sonarqube/bin/linux-x86-64$ nano sonar.sh
GNU nano 6.2          /opt/sonarqube/bin/linux-x86-64/sonar.sh *

#!/bin/sh

RUN_AS_USER=ddsonar
APP_NAME="SonarQube"

# By default, java from the PATH is used, except if SONAR_JAVA_PATH env variable is set
findjava() {
    if [ -z "${SONAR_JAVA_PATH}" ]; then
        if ! command -v java 2>&1; then
            echo "Java not found. Please make sure that the environmental variable SONAR_JAVA_PATH >
            exit 1
        fi
        JAVA_CMD=java
    else
        if ! [ -x "${SONAR_JAVA_PATH}" ] || ! [ -f "${SONAR_JAVA_PATH}" ]; then
            echo "File '${SONAR_JAVA_PATH}' is not executable. Please make sure that the environmen>
            exit 1
        fi
        JAVA_CMD="${SONAR_JAVA_PATH}"
    fi
}

findjava

```

Menu Bar:

- Help**
- Write Out**
- Where Is**
- Cut**
- Execute**
- Location**
- Exit**
- Read File**
- Replace**
- Paste**
- Justify**
- Go To Line**

Adding the RUN_AS_USER=ddsonar in the file sonar.sh

b) Save and exit the file.

8) Setup Systemd service

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

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

ii) Paste the following lines to the file.

```

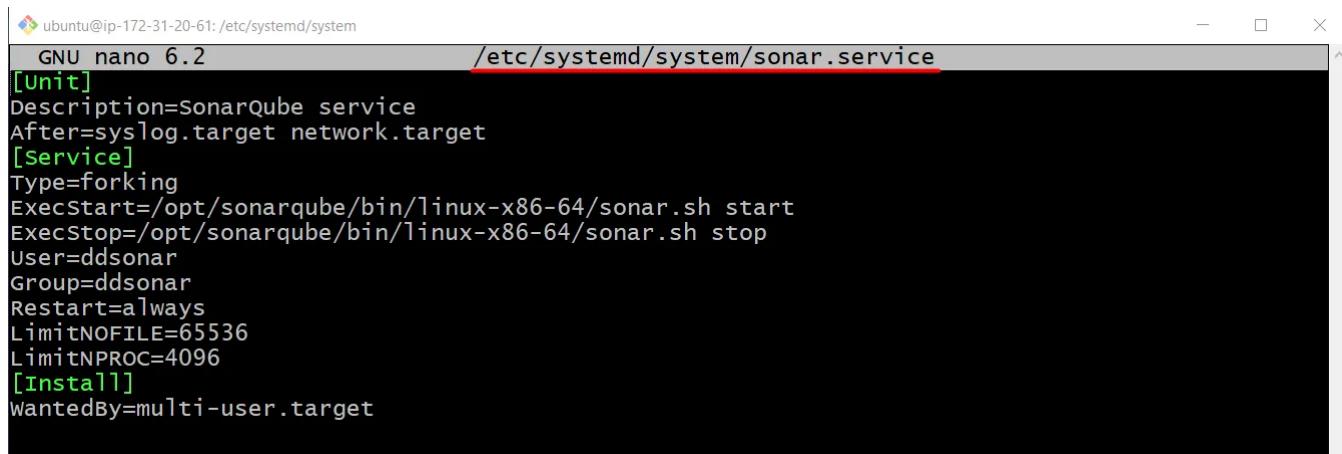
[Unit]
Description=SonarQube service
After=syslog.target network.target
[Service]
Type=forking
ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop
User=ddsonar
Group=ddsonar
Restart=always
LimitNOFILE=65536
LimitNPROC=4096
[Install]
WantedBy=multi-user.target

```

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



```
ubuntu@ip-172-31-20-61:/etc/systemd/system
GNU nano 6.2                               /etc/systemd/system/sonar.service
[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
```

Adding the lines of script to the file sonar.service

iii) Save and exit the file.

iv) Enable the SonarQube service to run at system startup.

```
sudo systemctl enable sonar
```

v) Start the SonarQube service.

```
sudo systemctl start sonar
```

vi) Check the service status.

```
sudo systemctl status sonar
```

```
ubuntu@ip-172-31-20-61:~$ sudo systemctl enable sonar
Created symlink /etc/systemd/system/multi-user.target.wants/sonar.service → /etc/systemd/system/sonar.service.
ubuntu@ip-172-31-20-61:~$ sudo systemctl start sonar
ubuntu@ip-172-31-20-61:~$ sudo systemctl status sonar
● sonar.service - SonarQube service
   Loaded: loaded (/etc/systemd/system/sonar.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2023-05-27 07:19:24 UTC; 9s ago
     Process: 19361 ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start (code=exited, sta...
    Main PID: 19384 (java)
      Tasks: 67 (limit: 4686)
     Memory: 764.7M
        CPU: 17.462s
       CGroup: /system.slice/sonar.service
               └─19384 java -Xms8m -Xmx32m --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED>
                 ├─19409 /usr/lib/jvm/java-17-openjdk-amd64/bin/java -Xms4m -Xmx64m -XX:+UseSerial...
                 └─19471 /usr/lib/jvm/java-17-openjdk-amd64/bin/java -Des.networkaddress.cache.ttl=0 -Dsun.jnu...
May 27 07:19:24 ip-172-31-20-61 systemd[1]: Starting SonarQube service...
May 27 07:19:24 ip-172-31-20-61 sonar.sh[19361]: /usr/bin/java
May 27 07:19:24 ip-172-31-20-61 sonar.sh[19361]: Starting SonarQube...
May 27 07:19:24 ip-172-31-20-61 sonar.sh[19361]: Started SonarQube.
May 27 07:19:24 ip-172-31-20-61 systemd[1]: Started SonarQube service.
Lines 1-18/18 (END)
```

Starting and checking the status of the SonarQube

vii) Hurray, It's up and running.

9) Modify Kernel System Limits

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

i) Edit the `sysctl` configuration file.

```
sudo nano /etc/sysctl.conf
```

ii) Add the following lines.

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

Adding the lines of codes at the file /etc/sysctl.conf

- iii) Save and exit the file.
- iv) Reboot the system to apply the changes.

```
sudo reboot
```

10) Access SonarQube Web Interface

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

For example, <http://IP:9000>

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



Fresh installed SonarQube GUI

ii) Change the Old password with a New one.

Log in with username **admin** and password **admin**. In the next step, SonarQube will prompt you to change your password. **CHANGE THE PASSWORD.**



Change the old password.

iii) And yes, we have successfully installed the SonarQube Community version 10.0 on AWS EC2 Ubuntu 22.

We are inside the SonarQube. Let's get started.

Congratulations, Our SonarQube has been installed successfully.

Remaining Extra Work.



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.onecloudhelper.com. This is mine. Please use yours.

1) Install Nginx

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

Installing Nginx

2) Adding the DNS records.

My domain is hosted on the AWS Route53, thus I will be adding the records in there.

Adding the IP of the server on the DNS zone.

And now our domain **sonarqube.onecloudhelper.com** is ready to be used for the nginx configuration setup.

3) Create a new Nginx configuration file for the site.

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

4) 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 sonarqube.onecloudhelper.com;` with your domain or sub-domain name.

5) Next, make sure your configuration file has no syntax errors.

```
sudo nginx -t
```

Checking the Nginx configuration file for any errors.

6) Enable the Nginx server.

```
sudo systemctl enable nginx
```

7) Restart the nginx server

```
sudo systemctl restart nginx
```

8) Check the status of the nginx server

```
sudo systemctl status nginx
```

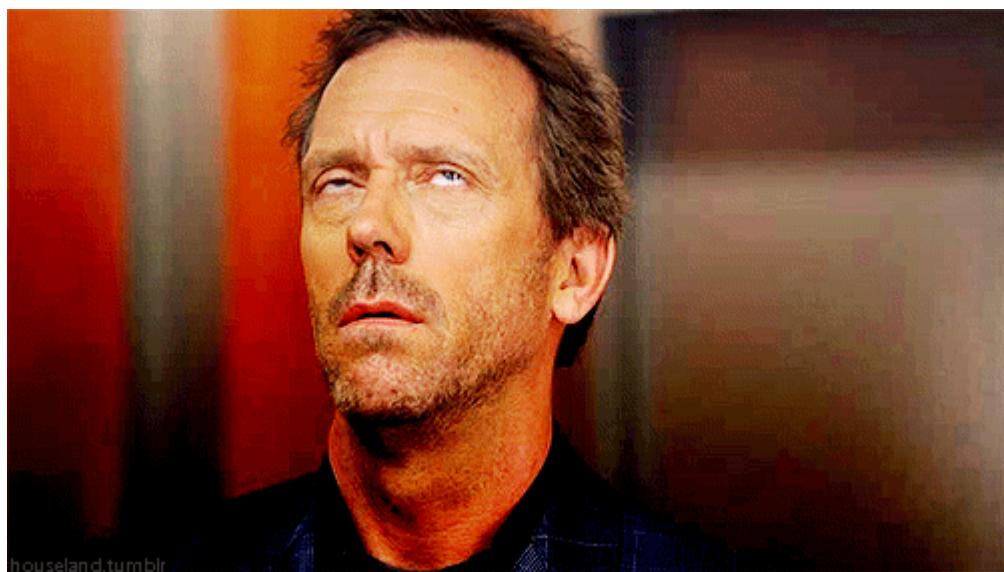
Starting and Enabling the Nginx server

9) Use the configured domain.

You can now visit <http://sonarqube.onecloudhelper.com> 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.

Installing SSL certificate for maximum security

We will use the free SSL certificate provided by Certbot for our sub-domain sonarqube.onecloudhelper.com. Let's see how we will do this.

i) Installing certbot

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

Installing Certbot dependencies

ii) Generate SSL certificate

```
sudo certbot --nginx
```

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

Success message that the SSL certificate has been deployed on the domain sonarqube.onecloudhelper.com

iii) Certificate deployed successfully.

Now if we hit the domain sonarqube.onecloudhelper.com, it automatically redirects to HTTPS.

SSL certificate implemented successfully

AAHHH....

It was a lot of work.



Was it?

Let's recap what we have done till now.

- i) Installed and configured JDK 17 and PostgreSQL.
- ii) Installed and configured SonarQube.
- iii) Installed and configured the Nginx to run the SonarQube on a domain.
- iv) Installed the SSL certificate on the Subdomain.
- v) And finally made it work.

Congratulations, You did it and everything looks perfect, you are ready to go ahead using the awesome SonarQube and start making your code cleaner and smarter.

Thank you and We will meet soon....



Cheers, Cheers, Cheers, Cheers....

Sonarqube

Aws Ec2

Nginx

Lets Encrypt

Cloud Computing



Follow



Written by DeshDeepakDhobi (DD)

57 Followers

AWS Community Builder | 2x - AWS Certified | DevOps Engineer | L2 Support Engineer | Problem solver, enthusiast learner and helper.

More from DeshDeepakDhobi (DD)

 DeshDeepakDhobi (DD)

How to happily SSH/Login into the AWS-managed ECS Fargate Containers?

AWS ECS Fargate is a serverless approach to deploying containers that helps the customers focus on the major development works rather than...

7 min read · Jun 28, 2022



12



 DeshDeepakDhobi (DD) in AWS Tip

How to solve the issue of “Failed” status for AWS ACM certificates?

Problem:

3 min read · Feb 6



1

 DeshDeepakDhobi (DD) in AWS Tip

How to Install, Deploy, and Test Kubernetes (k8s) Cluster on Ubuntu 20.04?

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management.

9 min read · Feb 12



DeshDeepakDhobi (DD) in AWS Tip

How RDP into AWS Windows Instance GUI using AWS Sessions Manager and a few fun commands?

AWS EC2 instances provide us with the SSHing capability using the concept of Public and Private keys.

5 min read · Feb 6



See all from DeshDeepakDhobi (DD)

Recommended from Medium



 Humza Arshad Khan

SonarQube Installation on ubuntu 20.04

How to Install #sonarqube in Ubuntu Linux—Download and Setup SonarQube

3 min read · Oct 19





Pankaj Makhijani in AWS in Plain English

Attach SSL Certificate and Domain to Your EC2 Instance Using Route 53 and AWS ACM

Introduction:

4 min read · Jul 10



15



1



Lists

Staff Picks

535 stories · 539 saves

Stories to Help You Level-Up at Work

19 stories · 369 saves

Self-Improvement 101

20 stories · 1051 saves

Productivity 101

20 stories · 956 saves

 Hopefully Surprising Blog in Stackademic

SonarQube Community Edition: Comprehensive guide for a free personal setup

Supercharge your code with SonarQube!

8 min read · Oct 3



 Thomas Puntillo

Creating a Jenkins container with persistent data using Docker Compose

Hello there! I'm so glad you decided to read this article because today we'll be discussing a tool that will change your life (assuming...)

10 min read · Jul 9



85



1



Theara Seng

Building a Full-Stack Product App with React, Spring Boot, and Docker Compose

In this blog, I am going to share with you a very simple product app that uses React, Spring Boot, and MySQL as the frontend, backend, and...

5 min read · Oct 28



15





Mohsin Rubel

CI/CD Pipeline Setup with Jenkins, SonarQube, Docker, and Nginx

Project Objective: Set up a Jenkins Freestyle Project for a portfolio hosted on GitHub.

Environment: Three VMs — 1st for Jenkins, 2nd for...

3 min read · Sep 12



See more recommendations