MAY 15, 2020

INSTALLING, MANAGING, AND RUNNING SONARQUBE COMMUNITY ON EC2

IMPROVING QUALITY OF SOURCE CODE

MARUDHAMARAN GUNASEKARAN

Contents

Installing the SonarQube Server 2

Installing Prerequisites 2

Installing Java 2

Configuring Java 3

Installing Postgresql 3

Configuring Default Postgres Identity 4

Configuring Postgres User for SonarQube 6

Configuring Authentication Methods for Postgres 7

Configuring Data Directory for Postgresql 9

Configuring Postgres Database and a Postgres User for SonarQube 11

Downloading and Installing SonarQube server 11

Creating SonarQube server identity 12

Configuring SonarQube server 12

Change System level memory and file system limits for SonarQube Server 13

Creating systemd files to run ServerQube as a background service 14

Running a SonarQube Scan with Sonar Runner 15

Git cloning required respositories 15

Installing and Configuring node.js to run analysis via sonar-scanner 15

Installing and Configuring sonar-scanner 16

Running source code analysis 16

Updating SonarQube Server 17

Administering SonarQube Server 17

Creating and Managing users 17

Enforcing Authentication 18

Troubleshooting 18

Installing the SonarQube Server

Installing Prerequisites

SonarQube has the following prerequisites

1. Java – as the runtime
2. Postgres – as the data store
3. Elastic Search – as the index engine
4. Node.js – required by sonar-scanner to scan javascript code
5. Sonar-scanner – the cli used to scan the source code in a directory

Detailed: https://docs.sonarqube.org/latest/requirements/requirements/

Installing Java

Verify the current JDK version

java -version

Prerequisites mention java version 11, so if java runtime (jre) is version 11, then the prerequisite is met.

However, the current version of java according to the date of this writing is version 14, so install version 14 of openjdk.

Download the latest openjdk from https://openjdk.java.net/install/

wget https://download.java.net/java/GA/jdk14.0.1/664493ef4a6946b186ff29eb326336a2/7/GPL/openjdk-14.0.1_linux-x64_bin.tar.gz

tar -xvf openjdk-14.0.1_linux-x64_bin.tar.gz

cd in to jdk-14.0.1

ls to see if bin directory exists

cd in to bin

ls to see if java exists

./java -version

should display the current version, that is 14.

Recursively copy the *jdk* to */usr/lib/jvm*

sudo cp -r jdk-14.0.1 "/usr/lib/jvm"

Verify copy

ls "/usr/lib/jvm/jdk-14.0.1/" -l

Remote the temporary unzip and download directories

rm -rf jdk-14.0.1

rm openjdk-14.0.1_linux-x64_bin.tar.gz

Configuring Java

Adjust the alternatives to set the current version of java (openjdk 14)

alternatives --list

sudo alternatives --install "/usr/bin/java" java "/usr/lib/jvm/jdk-14.0.1/bin/java" 3

sudo alternatives --config java

Choose the java 14 from alternatives

Later, verify

java -version

openjdk version "14.0.1" 2020-04-14

OpenJDK Runtime Environment (build 14.0.1+7)

OpenJDK 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)

Also set alternatives for jar and javac to the latest openjdk 14, just in case

sudo alternatives --install /usr/bin/jar jar /usr/lib/jvm/jdk-14.0.1/bin/jar 1

sudo alternatives --set jar /usr/lib/jvm/jdk-14.0.1/bin/jar

sudo alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-14.0.1/bin/javac 1

sudo alternatives --set javac /usr/lib/jvm/jdk-14.0.1/bin/

Installing Postgresql

The current version of postgres is 12 according to the date of this writing.

The postgres builds from postgres itself is not supported on amazon linux - https://www.postgresql.org/message-id/flat/15768-35c3af8405f5e346%40postgresql.org

Hence, fall back to version 11, as supplied in the amazon linux extras bundle - https://aws.amazon.com/amazon-linux-2/faqs/#Amazon_Linux_Extras

amazon-linux-extras | grep "post"

NOTE: The livepatch extra is in public preview, not meant for production use

```
 5  postgresql9.6          available    \
 6  postgresql10           available    [ =10  =stable ]
41  postgresql11           available    [ =11  =stable ]
```

sudo yum clean metadata
sudo yum install postgresql

sudo yum list postgresql*

sudo yum install postgresql-server
sudo yum install postgresql-contrib

Configuring Default Postgres Identity

The default identify that ships with the package to run postgresql is *postgres*. Change it to *postgresuser*

sudo adduser postgresuser

sudo passwd postgresuser

/database/ is the mount for database files, so the default directory for postgres to be changed

Create a directory /pgsql/data under /database/

sudo mkdir /database/pgsql/data -p

Then change the ownership of /database/pgsql/data to the user postgresuser

sudo chown -R postgresuser:postgresuser /database/pgsql/data

Also change the ownership of the default install directory

sudo chown -R postgresuser:postgresuser /var/lib/pgsql

Delete the default postgres user

sudo userdel -r postgres

Look at the default system unit file for postgresql and extend the unit file configuration to override the default postgres user to *postgresuser*

cat /lib/systemd/system/postgresql.service

Extending systemd service unit files are extensively documented at - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files#sect-Managing_Services_with_systemd-Unit_File_Modify

Create a directory *postgresql.service.d* under */etc/systemd/system*

sudo mkdir /etc/systemd/system/postgresql.service.d/

Create a file named *overrideuser.conf*

sudo touch /etc/systemd/system/postgresql.service.d/overrideuser.conf

Edit the *overrideuser.conf* to provide *User* and *Group* information as the newly created *postgresuser*.

sudo vi /etc/systemd/system/postgresql.service.d/overrideuser.conf

[Service]
User=postgresuser
Group=postgresuser

:wq!

The configuration files from configuration directories in #/etc/systemd/system/ take precedence over the default unit files in #/usr/lib/systemd/system/.

Change the ownership of the /var/run/postgresql where the postgres will create a socket when running

sudo chown -R postgresuser:postgresuser /var/run/postgresql

/var/run/postgresql directory will be erased on every reboot, hence the directory need to be created during reboot with the ownership of the newly created *postgresuser*.

sudo vi /usr/lib/tmpfiles.d/postgresql.conf

d /var/run/postgresql 0755 postgresuser postgresuser -

:wq!

Failing to correctly modify the ownership and configuration of the /var/run/postgressql might results in errors like below: sudo systemctl status postgresql.service -l # FATAL: could not create lock file #"/var/run/postgresql/.s.PGSQL.5432.lock": Permission denied

After the above changes, Enable postgresql.service to be run on system start, then reload the systemd daemon, start the postgresql.service, check the status to see if the service is running.

sudo systemctl enable postgresql.service

sudo systemctl daemon-reload

sudo systemctl restart postgresql.service

sudo systemctl status postgresql.service -l

Now postgresql.service should be running under the newly created user identify *postgresuser*.

Configuring Postgres User for SonarQube

Login as *postgresuser*

su – postgresuser

psql --version

should throw a fatal error that “postgresuser” does not exist. This expected behavior us because postgresql creates a template db for every user. So create a create template db for the user *postgresuser*.

psql

createdb postgresuser

SELECT version();

version

PostgreSQL 11.5 on x86_64-koji-linux-gnu, compiled by gcc (GCC) 7.3.1 20180712 (Red Hat 7.3.1-6), 64-bit

(1 row)

exit to logout of psql prompt

exit to logout of postgresuser

Configuring Authentication Methods for Postgres

Review the authentication methods:

https://www.postgresql.org/docs/11/auth-methods.html

https://www.postgresql.org/docs/11/auth-password.html

postgresql by default users ident and trust for local networks and the default authentication methods are found at

sudo cat /var/lib/pgsql/data/pg_hba.conf

The password storage mechanism for postgresql users is configured at

sudo cat /var/lib/pgsql/data/postgresql.conf

Starting postgresql11, there is support for SHA-256 based passwords with the configuration *scram-sha-256*. So, change the default password hashing mechanism for postgresql users at *postgresql.conf*.

sudo vi /var/lib/pgsql/data/postgresql.conf

password_encryption = 'scram-sha-256'

:wq!

Change the authentication settings in pg_hba.conf

Change *peer* to *trust* and *ident* to *scram-sha-256*

sudo vi /var/lib/pgsql/data/pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            ident
host    replication     all             ::1/128                 ident
```

Reload and start service after authentication change.

sudo systemctl status postgresql.service -l

sudo systemctl daemon-reload

sudo systemctl restart postgresql.service

sudo systemctl status postgresql.service -l

Verify authentication changes by logging in to psql and checking version.

su - postgresuser

psql --version

select * from pg_shadow;

List users, and change the password for the database user named *postgresuser*.

\du+

\password

After password change the ps_shadow table should show a password for the postgresuser starting with scram-sha-256.

select * from pg_shadow;

Exit from pgsql, and postgresuser.

Change the *pg_hba.conf* to enforce scram-sha-256 for local connections as well.

sudo vi /var/lib/pgsql/data/pg_hba.conf

Change trust to scram-sha-256

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     scram-sha-256
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     scram-sha-256
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256
```

:wq!

Reload the daemon, and restart the postgresql.service.

sudo systemctl daemon-reload
sudo systemctl restart postgresql.service
sudo systemctl status postgresql.service -l

As a last change, login to postgres and delete the default user database ‘postgres’.

su - postgresuser

psql

Enter the newly created password for the database identify *postgresuser*.

List all databases

\l

drop database postgres;

Continue to change the default data directory for postgresql.

Configuring Data Directory for Postgresql

In the psql prompt, show data_directory displays the current data directory.

show data_directory;

data_directory

/var/lib/pgsql/data

(1 row)

Exit psql and *postgresuser*.

\q

exit

Stop the *postgresql.service*

sudo systemctl stop postgresql.service
sudo systemctl status postgresql.service -l

Move the default data directory to */database* mount point preserving the permissions and attributes.

sudo rsync -av /var/lib/pgsql /database

Change the default data directory configuration in */var/lib/pgsql/data/postgresql.conf*

sudo vi /var/lib/pgsql/data/postgresql.conf

data_directory = '/database/pgsql/data'

:wq!

After changing the default data directory. Login to *psql* as *postgresuser* and check *show data_directory*;

su - postgresuser

psql

show data_directory;

data_directory

/database/pgsql/data

(1 row)

When the default data directory is moved, override the default systemd unit file configuration to enforce */database/pgsql/data*.

sudo touch /etc/systemd/system/postgresql.service.d/overridedatadirectory.conf

sudo vi /etc/systemd/system/postgresql.service.d/overridedatadirectory.conf

[Service]

# Location of database directory
Environment=PGDATA=***/database/pgsql/data***
Environment=PGLOG=***/database/pgsql/data/pgstartup.log***

:wq!

Reload daemon, and restart the *postgresql.service*.

sudo systemctl daemon-reload
sudo systemctl restart postgresql.service
sudo systemctl status postgresql.service -l

Configuring Postgres Database and a Postgres User for SonarQube

Create a database named sonarqubedb and a user named sonaruser for SonarQube server.

su - postgresuser

psql

create database sonarqubedb;

create user sonarqubeuser with encrypted password 'ReferToPasswordManager';

Grant all privileges on database *sonarqubedb* to *sonarqubeuser.*

grant all privileges on database sonarqubedb to sonarqubeuser;

Downloading and Installing SonarQube server

/var/lib/ is the installation directory for sonarqube server. So cd in to /var/lib/

cd /var/lib

Obtain the latest version from https://www.sonarqube.org/downloads/

sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.3.1.34397.zip

Unzip and move contents to the sonarqube directory

sudo unzip sonarqube-8.3.1.34397.zip
sudo mv sonarqube-8.3.1.34397/* sonarqube

Remove the default unzip directory and the downloaded .zip file source

sudo rm -r sonarqube-8.3.1.34397
sudo rm -r sonarqube-8.3.1.34397.zip

Create sonarqube data directory for elastic search

sudo mkdir /database/sonarqube
sudo mkdir /database/sonarqube/data
sudo mkdir /database/sonarqube/temp
sudo mkdir /database/sonarqube/logs

Creating SonarQube server identity

Create a user named *sonaruser* and change ownership of the sonarqube data directory to the *sonaruser*.

sudo useradd sonaruser
sudo passwd sonaruser

sudo chown -R sonaruser:sonaruser /database/sonarqube /var/lib/sonarqube

Configuring SonarQube server

Edit sonarqube configuration file present at */var/lib/sonarqube/conf/sonar.properties* for postgresql database and webserver settings. Webserver needs to run on port *8080*.

sudo vi /var/lib/sonarqube/conf/sonar.properties

And Change the below properties. The properties are spread across the file from the top till the bottom of the file.

sonar.jdbc.username=sonarqubeuser
sonar.jdbc.password=ReferToPasswordManager
sonar.jdbc.url=jdbc:postgresql://localhost/sonarqubedb

sonar.web.port=8080

sonar.path.logs=/database/sonarqube/logs

sonar.path.data=/database/sonarqube/data
sonar.path.temp=/database/sonarqube/temp

:wq!

Change System level memory and file system limits for SonarQube Server

SonarQube server has prescribed minimum memory and file system limits as described in the prerequisites – https://docs.sonarqube.org/latest/requirements/requirements/.

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

Review the current system prerequisites using the commands below and adjust the limits accordingly.

sysctl vm.max_map_count
sysctl fs.file-max
ulimit -n
ulimit -u

Adjust the below values only in the minimum configuration is not met by the current operating system.

sudo touch /etc/sysctl.d/99-sonarqube.conf

sudo vi /etc/sysctl.d/99-sonarqube.conf

vm.max_map_count=262144
fs.file-max=35536

:wq!

sudo touch /etc/security/limits.d/99-sonarqube.conf

sudo vi /etc/security/limits.d/99-sonarqube.conf

sonaruser - nofile 65536
sonaruser - nproc 4096

:wq!

After the system limit changes, a reboot is required for settings to take effect. So, reboot

sudo reboot

Creating systemd files to run ServerQube as a background service

Create a file named sonarqube.service under /etc/systemd/system with the contents as described below:

sudo vi /etc/systemd/system/sonarqube.service

[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=simple
User=sonaruser
Group=sonaruser
PermissionsStartOnly=true
ExecStart=/bin/nohup java -Xms32m -Xmx32m -Djava.net.preferIPv4Stack=true -jar /var/lib/sonarqube/lib/sonar-application-8.3.1.34397.jar
StandardOutput=syslog
LimitNOFILE=65536
LimitNPROC=4096
TimeoutStartSec=5
Restart=always
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target

Please note the ExecStart parameter takes the .jar file with the version number as argument. During upgrades, this systemd unit file needs it's ExecStart parameter to be updated accordingly with the more recent version of sonarqube.

The above service file is the most suitable way to run sonarqube. However for testing purposes, sometimes if sonarqube needed to be run with the */var/lib/sonarqube/bin/linux-x86-64/sonar.sh* file, then the below parameter inside the */var/lib/sonarqube/bin/linux-x86-64/sonar.sh* file needs to be changed as *sonaruser*. ***#RUN_AS_USER=sonaruser***

Enable the service file on start, start the service, and check status to see if sonarqube is up and running.

sudo systemctl enable sonarqube.service
sudo systemctl start sonarqube.service
sudo systemctl status sonarqube.service -l

Verify if sonar is running on localhost:8080 by wget.

sudo wget http://localhost:8080

Running a SonarQube Scan with Sonar Runner

Review the Sonar Runner options at https://docs.sonarqube.org/latest/analysis/overview/

Use sonar-scanner as referenced at https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

Git cloning required respositories

sudo mkdir -p /database/bitbucket/fantasytravel/

cd /database/bitbucket/fantasytravel/

sudo git clone all or the desired repositories repositories

Installing and Configuring node.js to run analysis via sonar-scanner

Since node.js is not a part of the Amazon Extras, review the installation instructions at https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html

Get the latest version of node.js from

https://github.com/nvm-sh/nvm/blob/master/README.md

sudo wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
. ~/.nvm/nvm.sh
nvm install node

Verify node.js installation by checking the version.

node -e "console.log('Running Node.js ' + process.version)"

Enable node.js for sudoers by creating a symbolic link at /usr/bin

whereis node

node: /home/do.maran/.nvm/versions/node/v14.2.0/bin/node

sudo ln -s /home/do.maran/.nvm/versions/node/v14.2.0/bin/node /usr/bin/node

Reboot to apply changes

sudo reboot

Installing and Configuring sonar-scanner

Get the latest version of sonar-scanner from
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

cd /database

sudo wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.3.0.2102-linux.zip

sudo unzip sonar-scanner-cli-4.3.0.2102-linux.zip

sudo mv sonar-scanner-4.3.0.2102-linux sonar-scanner

Point sonar-scanner to the running url of sonarqube server. The sonar.host.url needs to be reachable from the sonar-scanner location. So, sonar-scanner can not only be run from the same EC2 instance, but also from anywhere in side the VLAN, VPN where the sonarqube server - https://sonarqube.fantasynetwork.com/ is accessible.

sudo vi /database/sonar-scanner/conf/sonar-scanner.properties

sonar.host.url=http://localhost:8080

:wq!

Make sure sonar-scanner is working by invoking its help.

/database/sonar-scanner/bin/sonar-scanner -h

Remove the downloaded .zip file

sudo rm -fR /database/sonar-scanner-cli-4.3.0.2102-linux.zip

Running source code analysis

From the root directory of a source code, sonar-runner needs to be invoked to run a source code analysis.

sonar-runner requires basic properties such as project key – which is unique in a sonarqube server.

The basic scan properties can be a part of the `sonar-project.properties` file in the root directory of a source code, or the basic scan properties can also be passed on to *sonar-scanner* via the command line.

The below command is an example of running *sonar-runner* for a source code located at *avacado-api-syncer*.

cd /database/bitbucket/fantasytravel/avacado-api-syncer

/database/sonar-scanner/bin/sonar-scanner -Dsonar.projectKey=avacado-api-syncer-8D2is5jQ -Dsonar.projectName=avacado-api-syncer -Dsonar.login=sonar-runner -Dsonar.password=ReferToPasswordManager >> /database/run-sonar-scanner-for-all-projects.log

sudo if permissions are denied to create the log file named */database/run-sonar-scanner-for-all-projects.log.*

sudo bash /database/run-sonar-scanner-for-all-projects.sh

Review the log files at:

cat /database/run-sonar-scanner-for-all-projects.log

Once the analysis is complete, review the results at

https://sonarqube.fantasynetwork.com/

Updating SonarQube Server

Review the recent version at https://www.sonarqube.org/downloads/

Follow the aforementioned steps, but from an update perspective.

After the update, ensure the *sonaruser* and the file permissions on the required directories are set appropriately as outlined.

Administering SonarQube Server

Creating and Managing users

By default, the administrator credentials are admin/admin. This has been changed, refer to the password manager for updated credentials for the default admin user.

sonar-runner is a user created for the *sonar-scanner* cli, which also is required to be an *administrator*.

Refer to the password manager for the credentials.

https://sonarqube.fantasynetwork.com/

Administration ➔ Security ➔Users

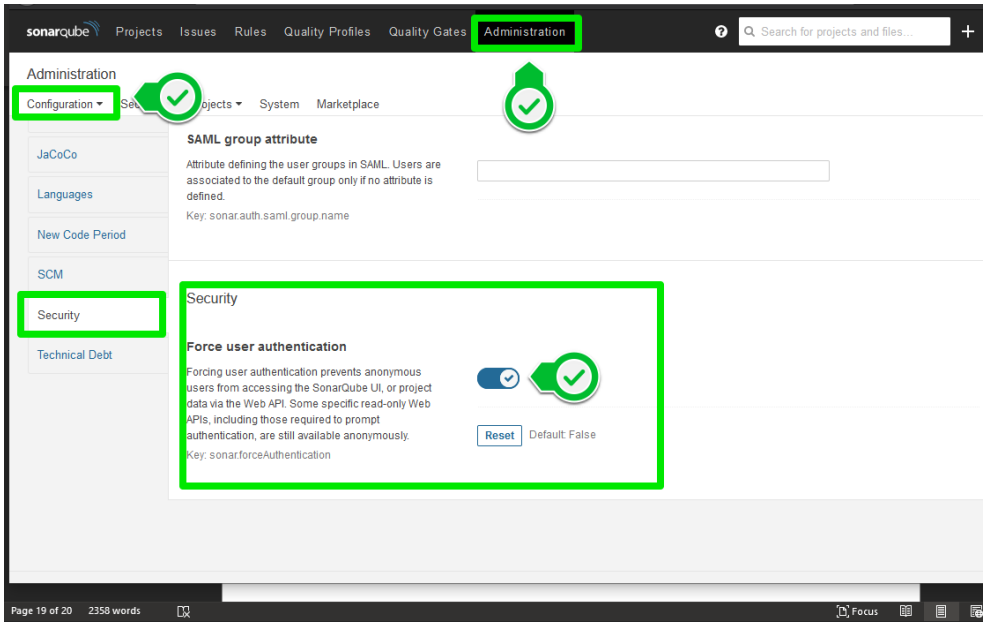
Enforcing Authentication

By default SonarQube web ui has authentication turned off, which means anyone hitting the url https://sonarqube.fantasynetwork.com/ would be able to view the scan results and many other information.

Disable anonymous authentication by enforcing authentication.

https://sonarqube.fantasynetwork.com/

Administration ➔ Configuration ➔ General settings ➔ Security ➔ Force user authentication.



Troubleshooting

Review systemd start up logs by the *status* command.

sudo systemctl status postgresql.service -l

sudo systemctl status sonarqube.service -l

Review postgresql logs at ***/database/pgsql/data/log***

Review sonarqube logs at ***/database/sonarqube/logs***

SonarQube has an elasticsearch subsystem for indexing, the logs are stored with the name ***es.log***

SonarQube has an inbuild java based web server, the logs for the web server are stored at ***web.log***

SonarQube has a compute engine, the logs are stored at ***ce.log***

SonarQube access logs are stored at ***access.log***

Success!