SonarQube Installation/Configuration on Docker

```
Latest:

# Create Network to link/connect the Sonarqube container with Postgres database

docker network create newnet1 --ip-range 192.168.0.0/24 --subnet 192.168.0.0/16


# Create a Volume for data on host

docker volume create mypgvol1


# Starting a postgres database container with the external volume named, “mypgvol1” so that the database is persistent even
# in case of the removal of the container. It mounts the postgres’s default database path on to the external volume.
# Assigns the fixed IP & hostname

docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword --network newnet1 -d postgres


docker run --name postgres -v mypgvol1:/var/lib/postgresql/data -e
POSTGRES_PASSWORD=password --network newnet1 --ip 192.168.0.2 -p 5432:5432 -h dbhost -d
postgres
```

Note:
This image has been configured with the below:

```
ENTRYPOINT ["docker-entrypoint.sh"]
CMD ["postgres"]
```

```
# restart container if it stops. if it is manually stopped,
# it is restarted only when Docker daemon restarts or the container itself is manually
# restarted.

docker update --restart always postgres


# Setup postgres database user-sonarqube

>> docker exec -it postgres  psql postgres -U postgres

create user sonarqube with encrypted password 'sonarqube';
```

Per this configuration, “postgres” will be passed as a parameter to the script “docker-entrypoint.sh”.

```
# connecting directly to postgres container via “psql” client tool as a database
# user-sonarqube
```

```
docker exec -it postgres  psql postgres -U sonarqube
```

```
# Connecting to Postgres via  a client container.
# Start a NEW client container with postgres image and invoke psql client & connect to the postgres db server
# Syntax: docker run -it --rm --network <network_name> <image_name> <command_to_be_executed_with_options>
# Syntax of psql: psql -h <DB host IP/container_name> -U <connect_as_user>

docker run -it --rm --network newnet1 postgres psql -h postgres -U postgres
```

```
Note:
The above container gets removed as soon as its exit at the end because of "--rm" usage.
```

```
# Create SonarQube Container
######################

# Create the required volumes first.

docker volume create --name sonarqube_data
docker volume create --name sonarqube_extensions
docker volume create --name sonarqube_logs

#Use the same network as the postgres db (newnet1)

Get the IP address of Postgres container

>> docker inspect postgres


# SONAR_JDBC_URL=jdbc:postgresql:<host IP:port#>/<database_name>


docker run -d --name sonarqube   --network newnet1 --ip=192.168.0.3 -p 9000:9000 -v
sonarqube_data:/opt/sonarqube/data     -v sonarqube_extensions:/opt/sonarqube/extensions  -e
SONAR_JDBC_URL=jdbc:postgresql://192.168.0.2:5432/postgres  -e SONAR_JDBC_USERNAME=sonarqube
-e SONAR_JDBC_PASSWORD=sonarqube    -v sonarqube_logs:/opt/sonarqube/logs sonarqube
```

```
# restart container if it stops if it is manually stopped
# it is restarted only when Docker daemon restarts or the container itself is manually
# restarted.

docker update --restart always sonarqube
```

It had failed to startup with the below error:

```
2021.09.25 07:40:55 INFO  es[][o.e.b.BootstrapChecks] explicitly enforcing bootstrap checks
```

ERROR: [1] bootstrap checks failed. You must address the points described in the following [1] lines before starting Elasticsearch.
*bootstrap check failure [1] of [1]: **max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]***

To fix the issue, the kernel parameter's value needs to be increased on the Docker host as below.

1) Add the following line to /etc/sysctl.conf:

vm.max_map_count=262144

2) Reload the config as root:

sysctl -p

3) Check the new value:

cat /proc/sys/vm/max_map_count

The sonarqube container will derive this from the Host's kernal.

Can you change kernel parameters of a Docker Container?

https://stackoverflow.com/questions/54845095/cannot-run-sysctl-command-in-dockerfile

Since Docker containers share the host system's kernel and its settings, a Docker container usually can't run sysctl at all. (You especially can't disable security-critical settings like this one.) You can set a limited number of sysctls on a container-local basis with docker run --sysctl, but the one you mention isn't one of these.*(such as, vm.max_map_count)*

Furthermore, you also can't force changes like this in a Dockerfile. A Docker image only contains a filesystem and some associated metadata, and not any running processes or host-system settings. Even if this RUN sysctl worked, if you rebooted your system and then launched a container from the image, that setting would be lost.

to read the logs being generated continuously, as the logs are getting generated - use the "-f" option.

```
docker logs -f sonarqube
```

<<Remove the sonarqube container and rerun the same command that creates it.>>

When you want to make any changes to the running container (regardless of the value passed to its -
ENTRYPOINT or CMD in the container's image), use "docker exec".

For example, to change a (changeable) kernel parameter value. To persist, commit the container after making the changes.

```
docker exec -it sonarqube /bin/bash
```

Configure project through the web app (www.<host_ip>:9000)

1) create a new project and a token (for any sonarqube user, for example, Administrator) through the sonarqube application. (http://165.232.182.235:9000/admin/users)

2) Create *sonar-project.properties* @current working directory where from sonarqube runner will be invoked.

   Sample:

   sonar.projectKey=pcl
   *sonar.exclusions=**/*.java*

Running sonar scanner client on the application(available at the given path) to generate reports
Define the variables used from command line

```
docker run --rm     --network newnet1 -e SONAR_HOST_URL="http://${SONARQUBE_URL}:9000"
-e SONAR_LOGIN="79a86d0053f7ebbb781032d0dfecb0f935942dc6"    -v
"/home/ubuntu/spring-petclinic:/usr/src"     sonarsource/sonar-scanner-cli
```