2018

SOFTWARE INDUSTRILISATION

DOCKER REPORT

BHRIGU MAHAJAN

EPITA | PARIS

INDEX

Docker

Docker is a computer program that performs operating-system-level virtualization. Docker is used to run software packages called containers.
One container runs a web server and web application, while a second container runs a database server that is used by the web application. Containers are isolated from each other and use their own set of tools and libraries; they can communicate through well-defined channels.
All containers use the same kernel and are therefore more lightweight than virtual machines.
Containers are created from images which specify their precise contents. Images are often created by combining and modifying standard images downloaded from repositories.

A container platform is a complete solution that allows organizations to solve multiple problems across a diverse set of requirements. It is more than a piece of technology and orchestration - it delivers sustainable benefits throughout your organization by providing all the pieces an enterprise operation requires including security, governance, automation, support and certification over the entire application lifecycle. Docker Enterprise Edition (EE) is an enterprise-ready container platform that enables IT leaders to choose how to cost-effectively build and manage their entire application portfolio at their own pace, without fear of architecture and infrastructure lock-in.

Why Docker?

- Isolation
- Lightweight
- Simplicity
- Workflow
- Community

Docker Platform = Docker Engine + Docker Daemon

Docker Engine

- Docker Daemon
- Docker CLI

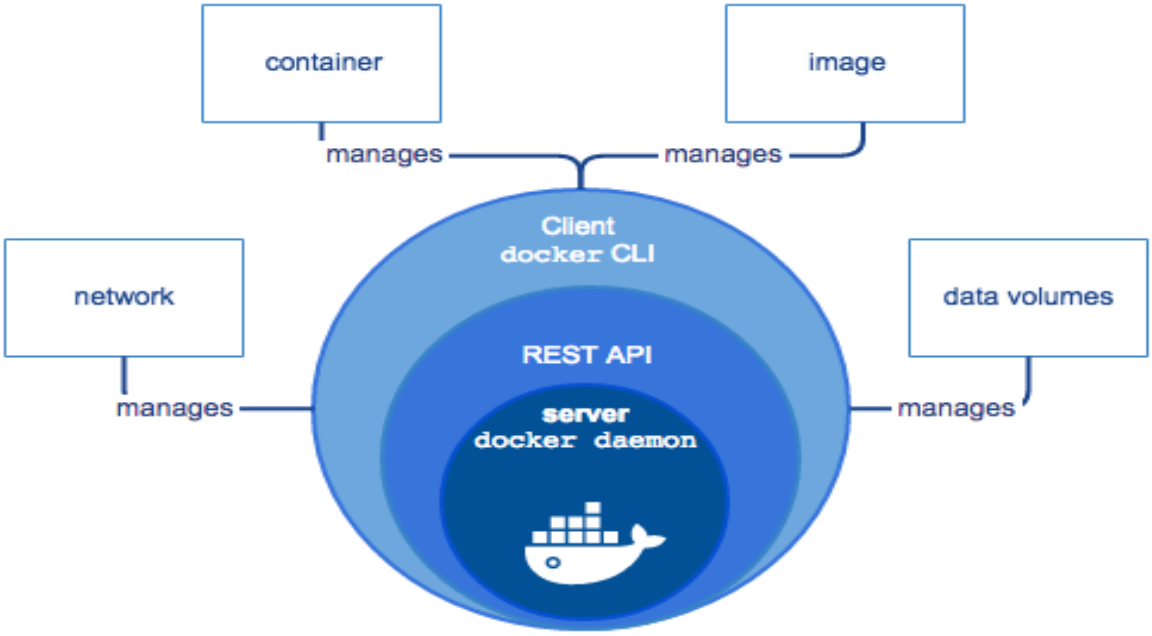
Docker Daemon

- Builds Images
- Runs and Manages Containers
- RESTful API

Docker CLI

• **docker build**	Build an image from a Docker file
• **docker images**	List all images on a Docker host
• **docker run**	Run an image
• **docker ps**	List all running and stopped instances
• **docker stop**	Stop a running instance
• **docker rm**	Remove an instance
• **docker rmi**	Remove an image

Docker Architecture



Docker Applications

- ✓ GITLAB
- ✓ JENKINS
- ✓ NEXUS
- ✓ SONARQUBE

Docker-Compose.yml

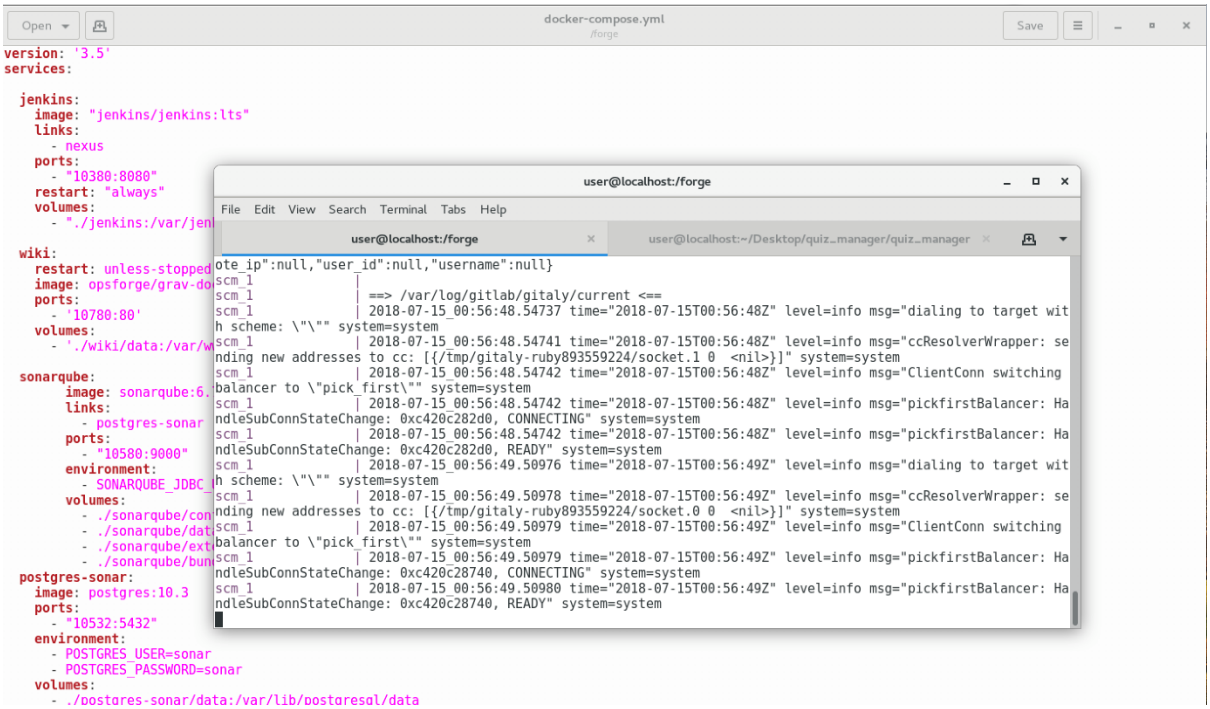
```
version: '3.5'
services:

jenkins:
image: "jenkins/jenkins:lts"
links:
- nexus
ports:
- "10380:8080"
restart: "always"
volumes:
- "./jenkins:/var/jenkins_home"

wiki:
restart: unless-stopped
image: opsforge/grav-docker:latest
ports:
- '10780:80'
volumes:
- './wiki/data:/var/www/html'
```

```
sonarqube:
image: sonarqube:6.7.3
links:
- postgres-sonar
ports:
- "10580:9000"
environment:
- SONARQUBE_JDBC_URL=jdbc:postgresql://postgres-sonar:5432/sonar
volumes:
- ./sonarqube/conf:/opt/sonarqube/conf
- ./sonarqube/data:/opt/sonarqube/data
- ./sonarqube/extensions:/opt/sonarqube/extensions
- ./sonarqube/bundled-plugins:/opt/sonarqube/lib/bundled-plugins
postgres-sonar:
image: postgres:10.3
ports:
- "10532:5432"
environment:
- POSTGRES_USER=sonar
- POSTGRES_PASSWORD=sonar
volumes:
- ./postgres-sonar/data:/var/lib/postgresql/data

nexus:
image: sonatype/nexus:oss
ports:
- "10680:8081"
volumes:
- ./nexus:/sonatype-work
scm:
image: 'gitlab/gitlab-ce:latest'
restart: always
ports:
- '10280:80'
- '443:443'
- '10222:22'
volumes:
- './scm/config:/etc/gitlab'
- './scm/logs:/var/log/gitlab'
- './scm/data:/var/opt/gitlab'
```

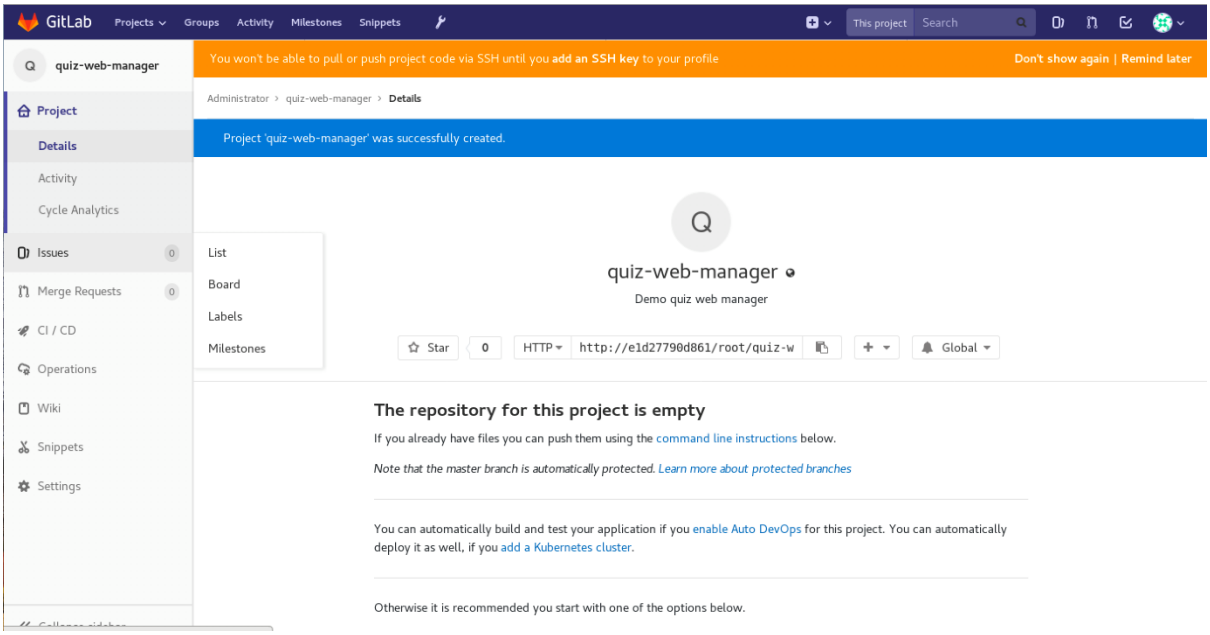
GITLAB

GitLab provides official Docker images to allowing you to easily take advantage of the benefits of containerization while operating your GitLab instance.

Launch terminal and go to directory forge.
Switch to root user
Run ‘Service docker start’
Run ‘setenforce 0’ to enter permissive mode
Run ‘docker-compose up scm’
Docker will fire pull command: *docker pull gitlab/gitlab-ce*

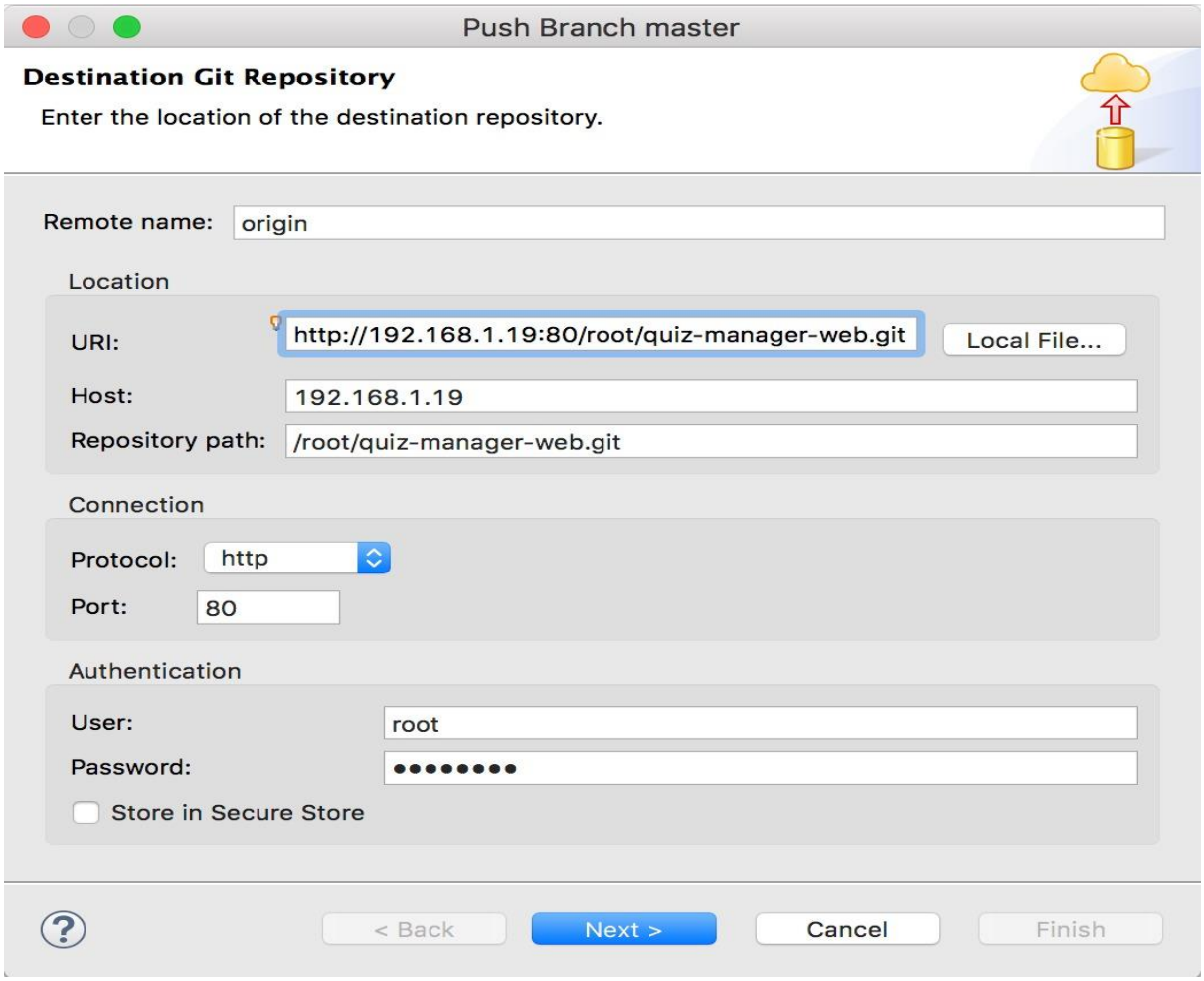


Once GITLAB installed successfully create a repository in Gitlab by accessing URL *localhost:10280*

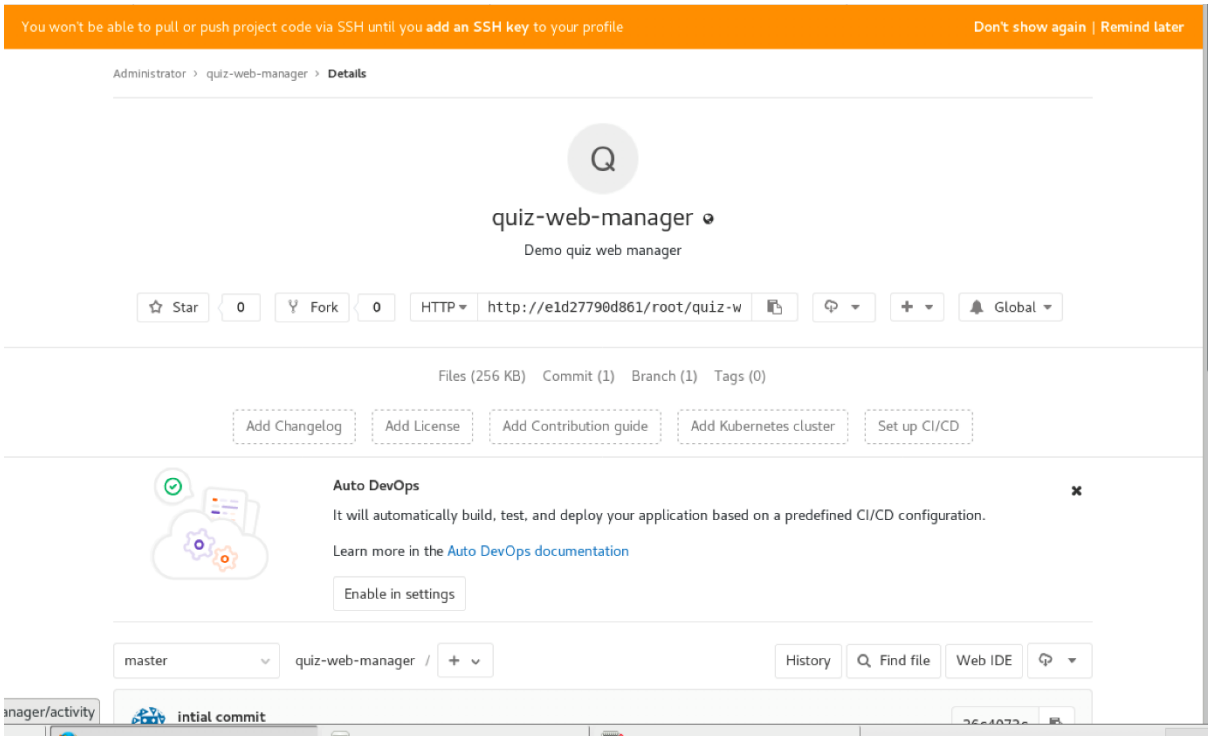


Then copy URL from virtual machine and push code from eclipse to that URL

- Replace localhost by IP address
- Change external port to 80
- Provide repository path



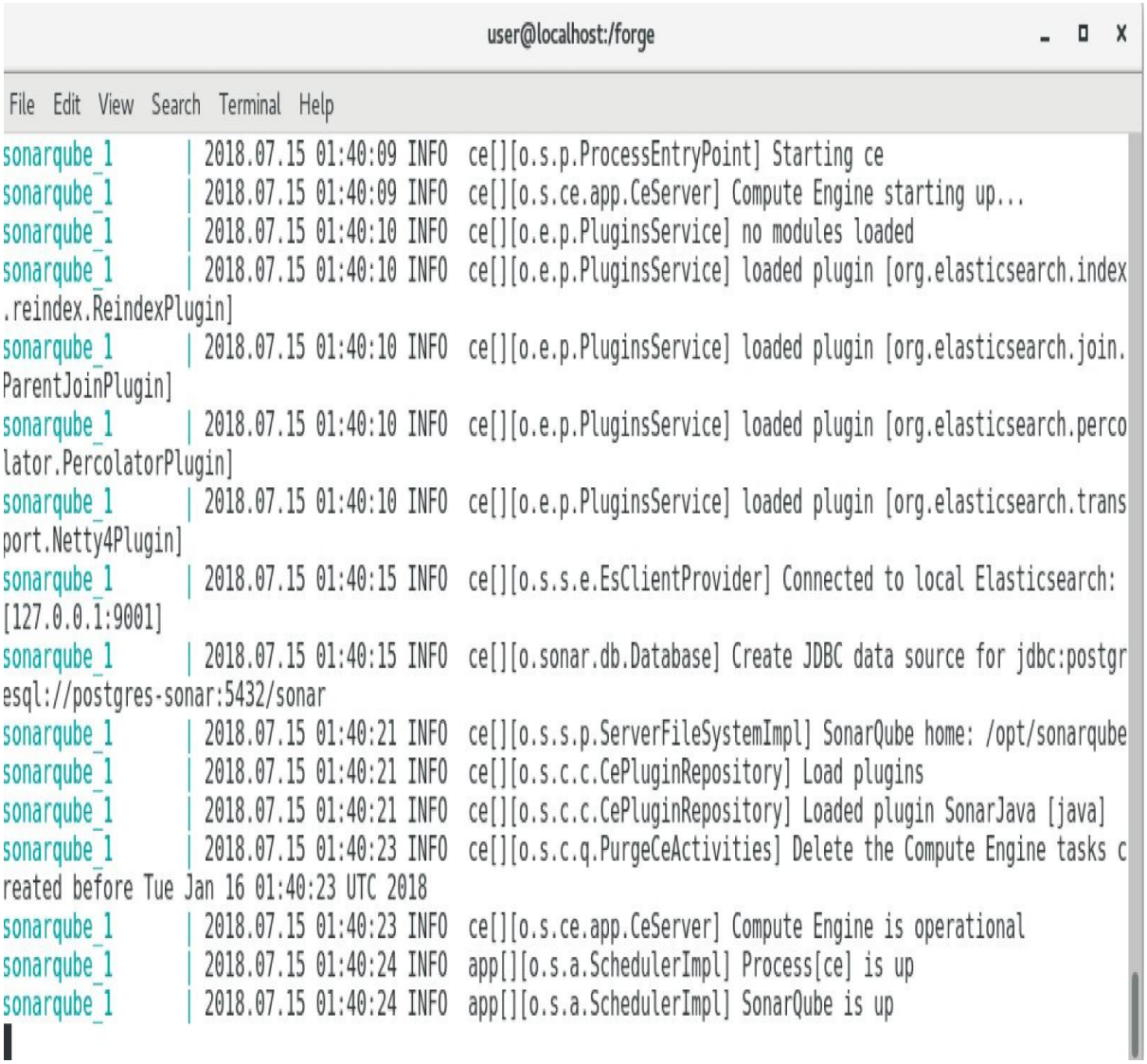
Code is pushed from the host to the GITLAB server



SonarQube

SonarQube is an open source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, and security vulnerabilities. SonarQube can record metrics history and provides evolution graphs. SonarQube's provides fully automated analysis and integration with Maven, Ant, Gradle, MSBuild and continuous integration tools like Jenkins, Hudson, etc.

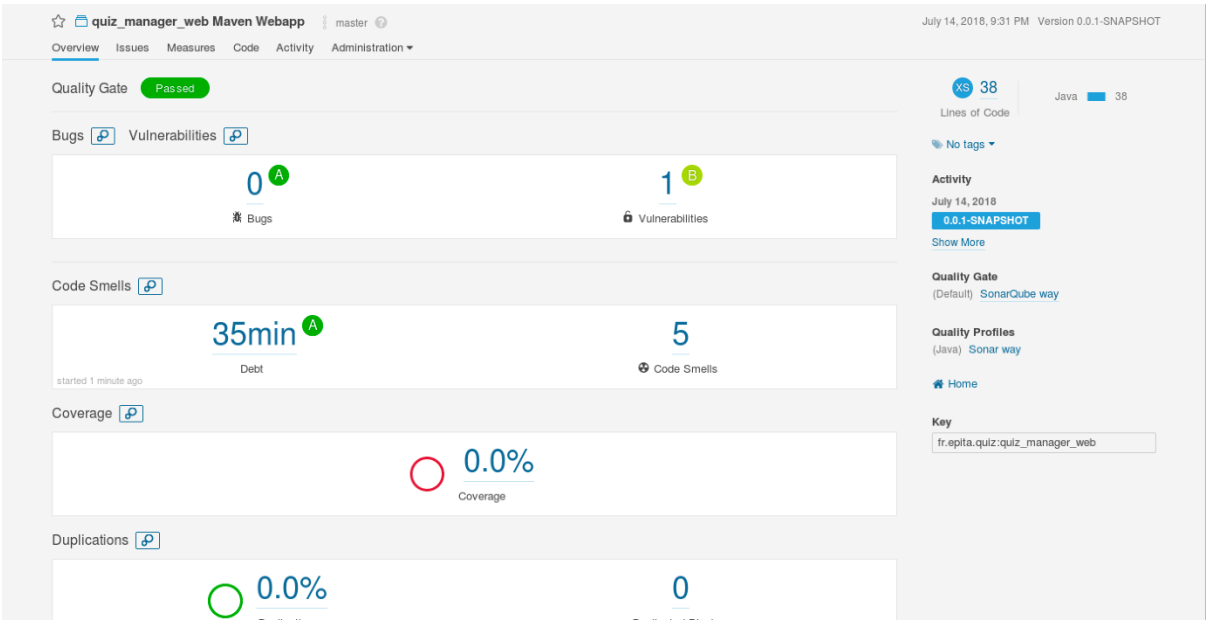
Launch terminal and go to directory forge.
Switch to root user
Run ‘Service docker start’
Run ‘setenforce 0’ to enter permissive mode
Run ‘docker-compose up sonarqube’



Wait till the Sonarqube server is up then add properties tag in pom.xml

```
<artifactId>quiz_manager_web</artifactId>
<packaging>war</packaging>
<version>0.0.1-SNAPSHOT</version>
<name>quiz_manager_web Maven Webapp</name>
<url>http://maven.apache.org</url>
<build>
    <finalName>quiz_manager_web</finalName>
</build>

<properties>

    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>

    <servlet.version>3.1.0</servlet.version>
    <sonar.host.url>http://sonarqube:9000</sonar.host.url>
    <sonar.exclusions>**bootstrap**</sonar.exclusions>

</properties>
<distributionManagement>
    <repository>
        <id>releases</id>
        <url>http://nexus:8081/nexus/content/repositories/releases</url>
    </repository>
    <snapshotRepository>
        <id>snapshots</id>
        <url>http://nexus:8081/nexus/content/repositories/snapshots</url>
    </snapshotRepository>
</distributionManagement>
<dependencies>

    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>${servlet.version}</version>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>fr.epita.quiz</groupId>
        <artifactId>quiz_manager</artifactId>
        <version>0.0.1-SNAPSHOT</version>

    </dependency>
```

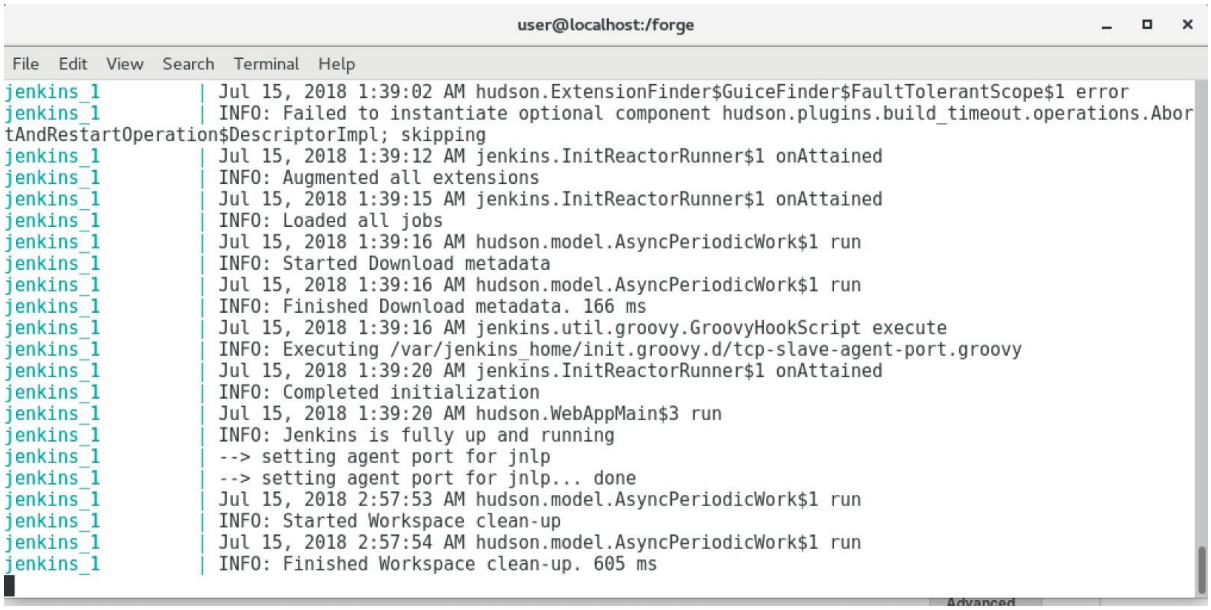
Then Deploy a build on Jenkins (configure Jenkins before build as steps mentioned in next upcoming pages).
Once build is successfully deployed hit Sonar URL at localhost:10580/ and check the code status.



JENKINS

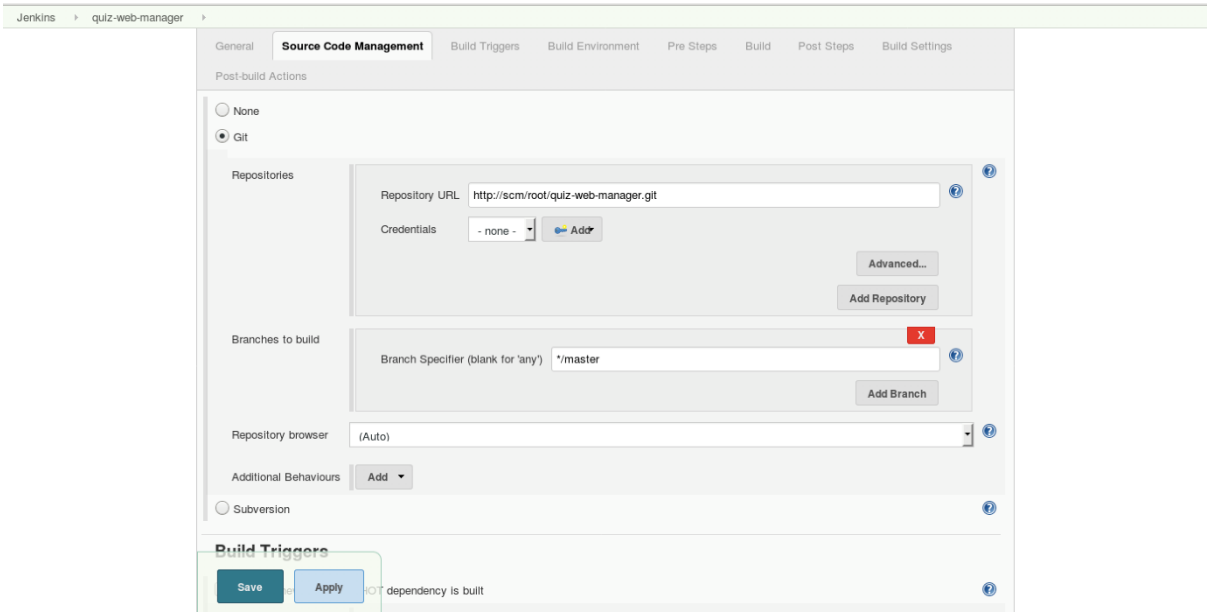
Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands. Jenkins is a highly extensible product whose functionality can be extended through the installation of plugins.

Launch terminal and go to directory forge.
Switch to root user
Run 'Service docker start'
Run 'setenforce 0' to enter permissive mode
Run 'docker-compose up Jenkins'



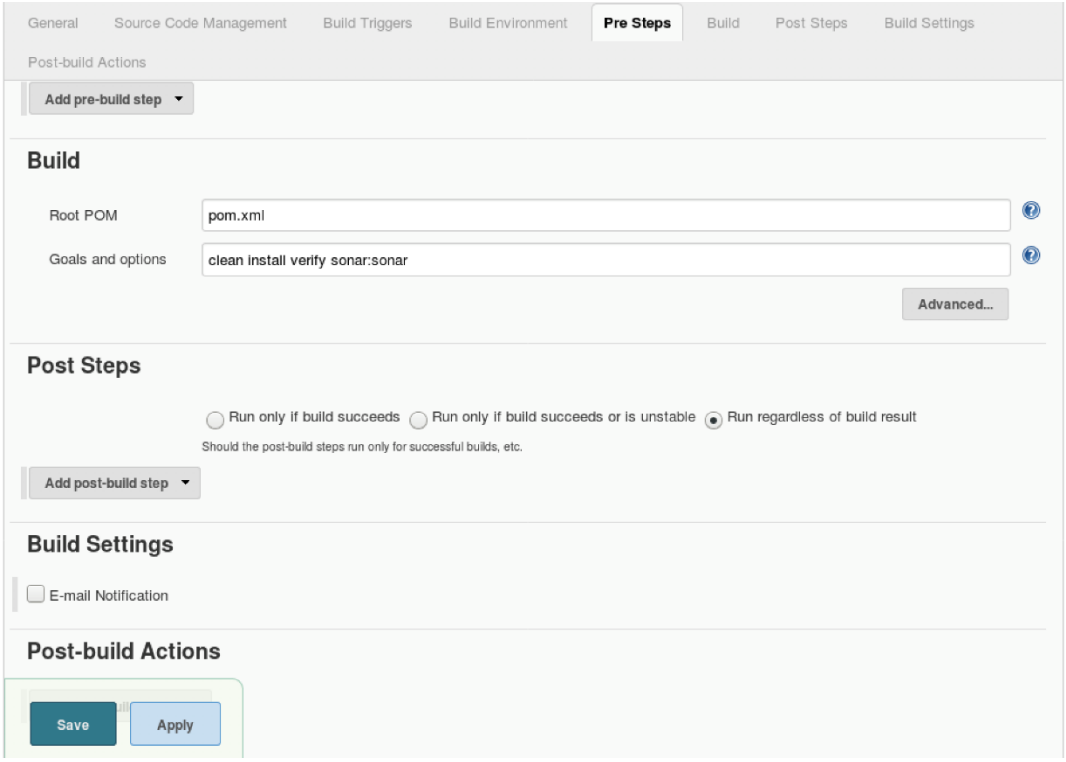
Once Jenkins is fully up and running hit URL on localhost:10280 and Jenkins dashboard will populate.

Install additional plugins if required from Manage Plugin tab in Manage Jenkins tab.
Open created project and select configure tab.
Select git and paste URL of git repository in Source code Management tab.



URL: http://scm/root/quiz-manager-web.git

Switch to Build Environment tab write name oh pom.xml in Root Pom column and in Goals and options column write 'clean install verify sonar: sonar'

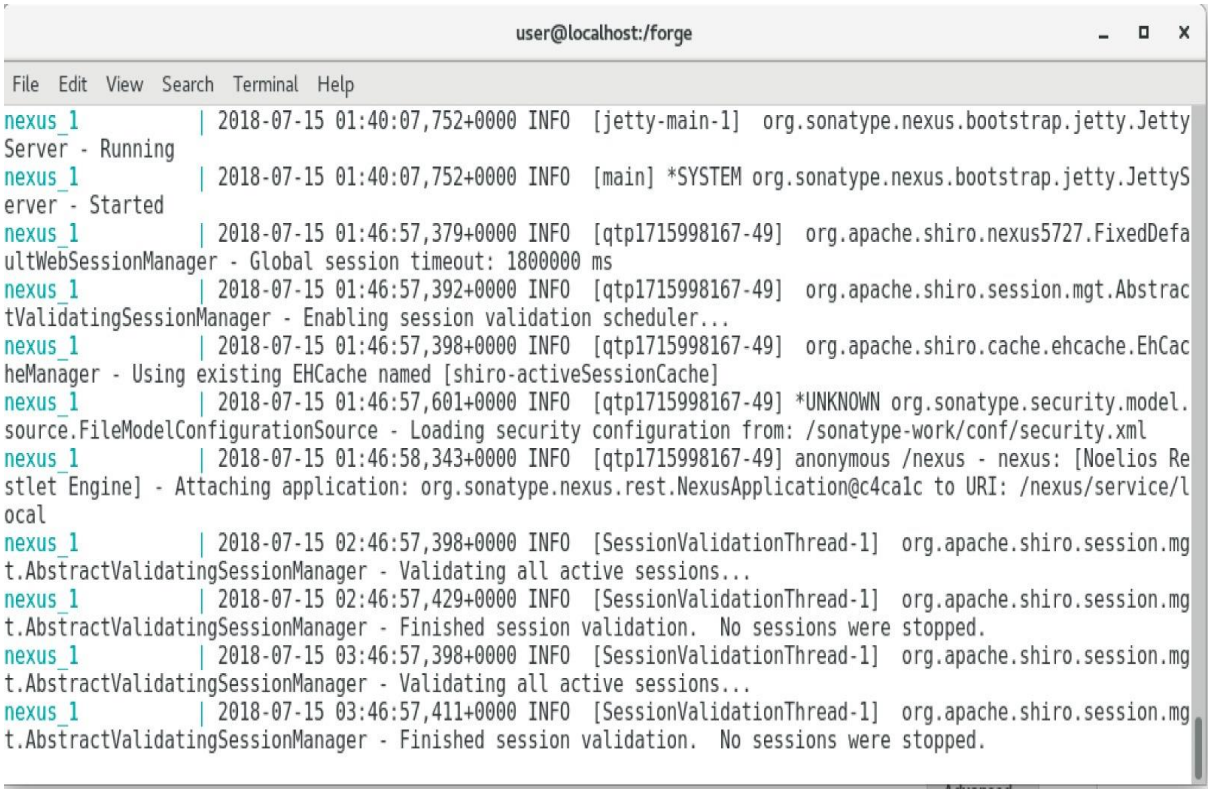


Click apply and build now to build. After successful completion deploy artifacts of project's release or snapshot to Nexus. Analyse sonarqube if required.

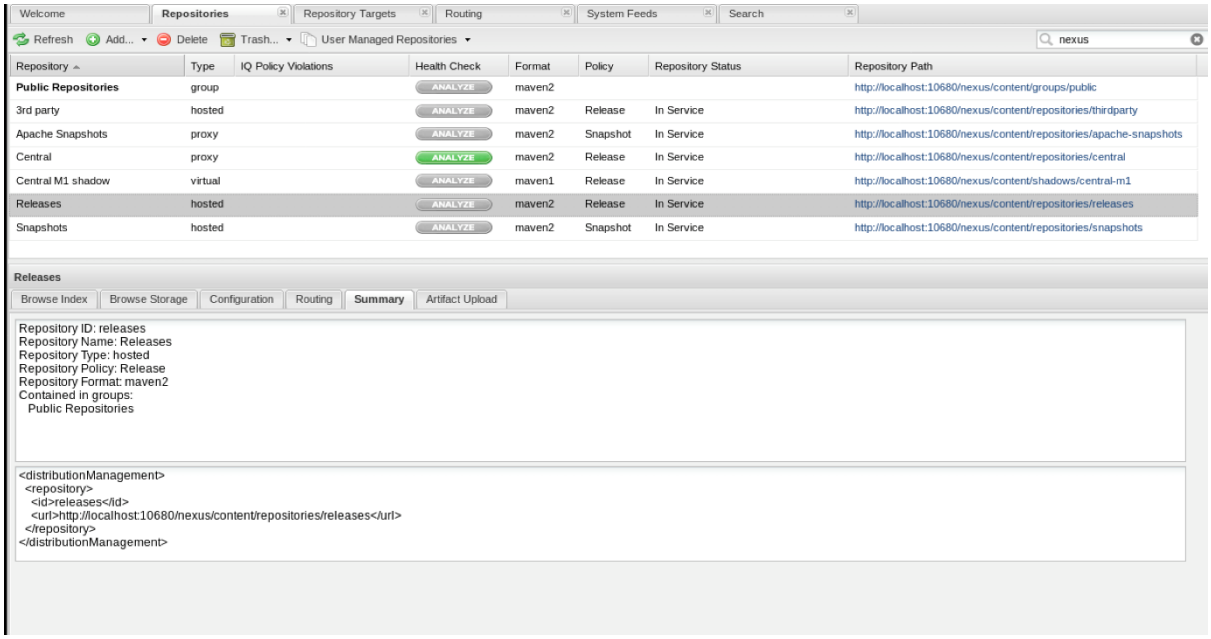
Nexus

Nexus is a repository manager. It allows you to proxy, collect, and manage your dependencies so that you are not constantly juggling a collection of JARs. It makes it easy to distribute your software. Internally, you configure your build to publish artifacts to Nexus and they then become available to other developers. You get the benefits of having your own 'central', and there is no easier way to collaborate.

Launch terminal and go to directory forge.
Switch to root user
Run ‘Service docker start’
Run ‘setenforce 0’ to enter permissive mode
Run ‘docker-compose up nexus’



Once Nexus is fully up and running hit URL on localhost:10680 and sign up for admin account.



Select the Snapshots or Release tab, then after summary tab in the pipeline and copy the dependencies to pom.xml file of the project in eclipse.

Add distributionManagement tag in pom.xml

```
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>quiz_manager_web Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <build>
        <finalName>quiz_manager_web</finalName>
    </build>

    <properties>

        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>

        <servlet.version>3.1.0</servlet.version>
        <sonar.host.url>http://sonarqube:9000</sonar.host.url>
        <sonar.exclusions>**bootstrap**</sonar.exclusions>

    </properties>
    <distributionManagement>
        <repository>
            <id>releases</id>
            <url>http://nexus:8081/nexus/content/repositories/releases</url>
        </repository>
        <snapshotRepository>
            <id>snapshots</id>
            <url>http://nexus:8081/nexus/content/repositories/snapshots</url>
        </snapshotRepository>
    </distributionManagement>
    <dependencies>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>${servlet.version}</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>fr.epita.quiz</groupId>
            <artifactId>quiz_manager</artifactId>
            <version>0.0.1-SNAPSHOT</version>

        </dependency>
```

Configure Nexus with Jenkins at /forge/Jenkins/tools/maven-installation/default/conf/settings.xml location and configure the servers with id of server matching the release and snapshot.