Hiren
Rathod

Software Industrialization

EPITA 2020

Documentation

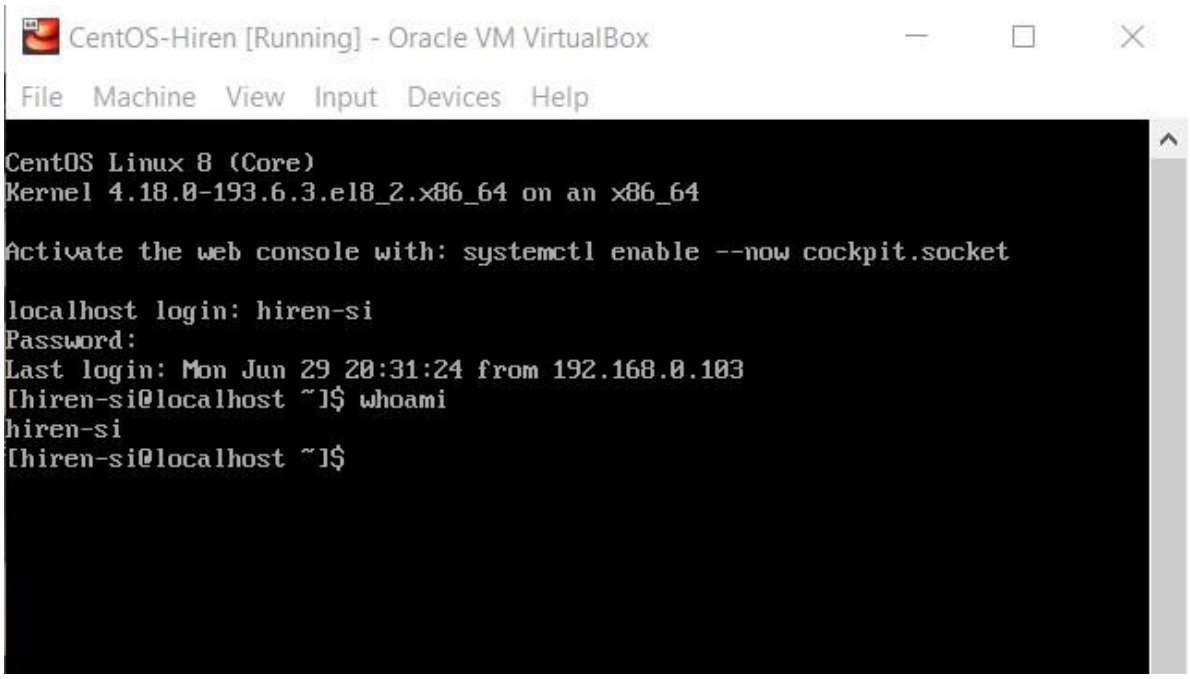
CentOS Installation

We will make a use of CentOS 8 DVD. iOS for the process of software industrialization by installing it using virtual box environment.

You can download DVD iOS for virtual box by clicking here...

Before that you need to download and setup VirtualBox environment for running the CentOS operating system as a server or UIX

You can follow the installation instruction by using the following Link



Install Docker Engine on CentOS

We will make a use o to install Docker Engine, you need a maintained version of CentOS 7. Archived versions aren't supported or tested.

If you already have an old docker version you need to remove it from the version and start it by fresh.

You can remove old version by using following command

```
$ sudo yum remove docker \
        docker-client \
        docker-client-latest \
        docker-common \
        docker-latest \
        docker-latest-logrotate \
        docker-logrotate \
        docker-engine
```

Installation Process:

Step 1:

You first need to set up the docker repository

Run the following commands into your machine

```
$ sudo yum install -y yum-utils

$ sudo yum-config-manager \
        --add-repo \
        https://download.docker.com/linux/centos/docker-ce.repo
```

Step 2: Install Docker Engine

Run the command in your machine:

```
$ sudo yum install docker-ce docker-ce-cli containerd.io
```

Step 3: Install Docker Compose

Run this command to download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Apply executable permission to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

To test the docker-compose is installed or not run the following command:

- docker-compose --version

Step 4: Configuration and Testing for Docker And Docker-Compose.

1. Start a Docker service:
   - sudo systemctl start docker
2. Verify the docker engine by using following command:
   - sudo docker run hello-world

Step 5: Jenkins Setup

We will setup Jenkins using docker-compose commands

To pull the image from the repo configure the ports and bindings the volumes from host machine to our machine:

Step 1:

Create a directory using:

“mkdir software-industrialization”

And change the directory by using

“cd software-industrialization”

Step 2:

Create another directory to save Jenkins data into it :

“mkdir Jenkins”

And give that directory owner permission recursively

“chown 1000:1000 Jenkins -R”

then

Create a docker-compose.yml file

“vi docker-compose.yml”

And

Add the following lines in file

```
version: '3.5'
services:
        jenkins:
                image: "jenkins/jenkins:lts"
                ports:
                  - "10380:8080"
                restart: "always"
                volumes:
                  - "./jenkins:/var/jenkins_home"
```

- save the file by using following commands
  press ESC key
  SHIFT + ; key
  wq and hit ENTER

Step 2:

Run the following command to execute Jenkins:

- docker-compose up -d jenkins

Step 3:

Go to your browser and in URL box type your IP address and following port that you have added in docker-compose.yml file

Eg: 192.168.0.102:10380

Jenkins will start……

Step 4:

It will ask for admin passcode:

Use: “docker -f logs jenkins”

It will show you passcode in a format

aksdj234513jasjh322421

copy paste the passcode and click on next button

- register your-self by adding username passcode
- Click on setup Jenkins and here you go.. it will start Jenkins and push you to Jenkins dashboard

Step 6: Sonatype/Nexus Setup

Step 1: by using the same procedure as we did before for Jenkins

Create another directory for binding nexus data into it:

"mkdir nexus"

And give that directory owner permission recursively

"chown 999 nexus -R"

Add the following lines to your docker-compose.yml file:

```
nexus:
        image: sonatype/nexus3
        ports:
          - "10680:8081"
        volumes:
          - "./nexus:/nexus-data"
```

- Save the file go to the directory where docker-compose.yml is and run the following command:
  "docker-compose up -d nexus"
- Visit browser and add you ipaddress add port to run nexus live
  "192.168.0.102: 10680"
- After the setup is up it will ask for username and password used by default credentials
  Username: admin
  Password: goto nexus folder there you'll find admin.password file copy that password and paste it there to log in.
- Your nexus is setup and running…

Step 7: SonarQube Setup

Step 1: by using the same procedure as we did before for Jenkins

Create another directory to save SonarQube data into it :

"mkdir sonarqube"

And give that directory owner permission recursively

"chown 999 sonarqube -R"

then

Add the following lines to your docker-compose.yml file:

```
sonarqube:
        image: "sonarqube:lts"
        ports:
          - "10580:9000"
        volumes:
          - ./sonarqube/data:/opt/sonarqube/data
```

- Save the file go to the directory where docker-compose.yml is and run the following command:
  "docker-compose up -d sonarqube"
- Visit browser and add you ipaddress add port to run nexus live
  "192.168.0.102: 10580"
- After the setup is up it will ask for username and password used by default credentials
  Username: admin
  Password: admin
- Your sonarqube it setup and running…

Step 8: Gitea Setup

Step 1: by using the same procedure

Create another directory for binding gitea data into it :

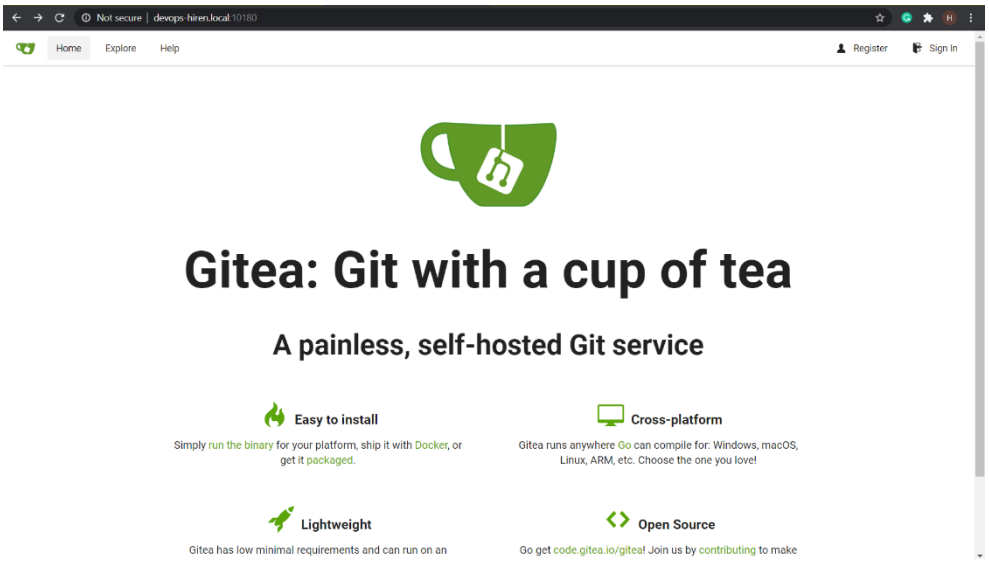
“mkdir gitea”

then

Add the following lines to your docker-compose.yml file:

```
gitea:
    image: "gitea/gitea:latest"
    ports:
      - "10122:22"
      - "10180:3000"
    volumes:
      - ./gitea:/data
```

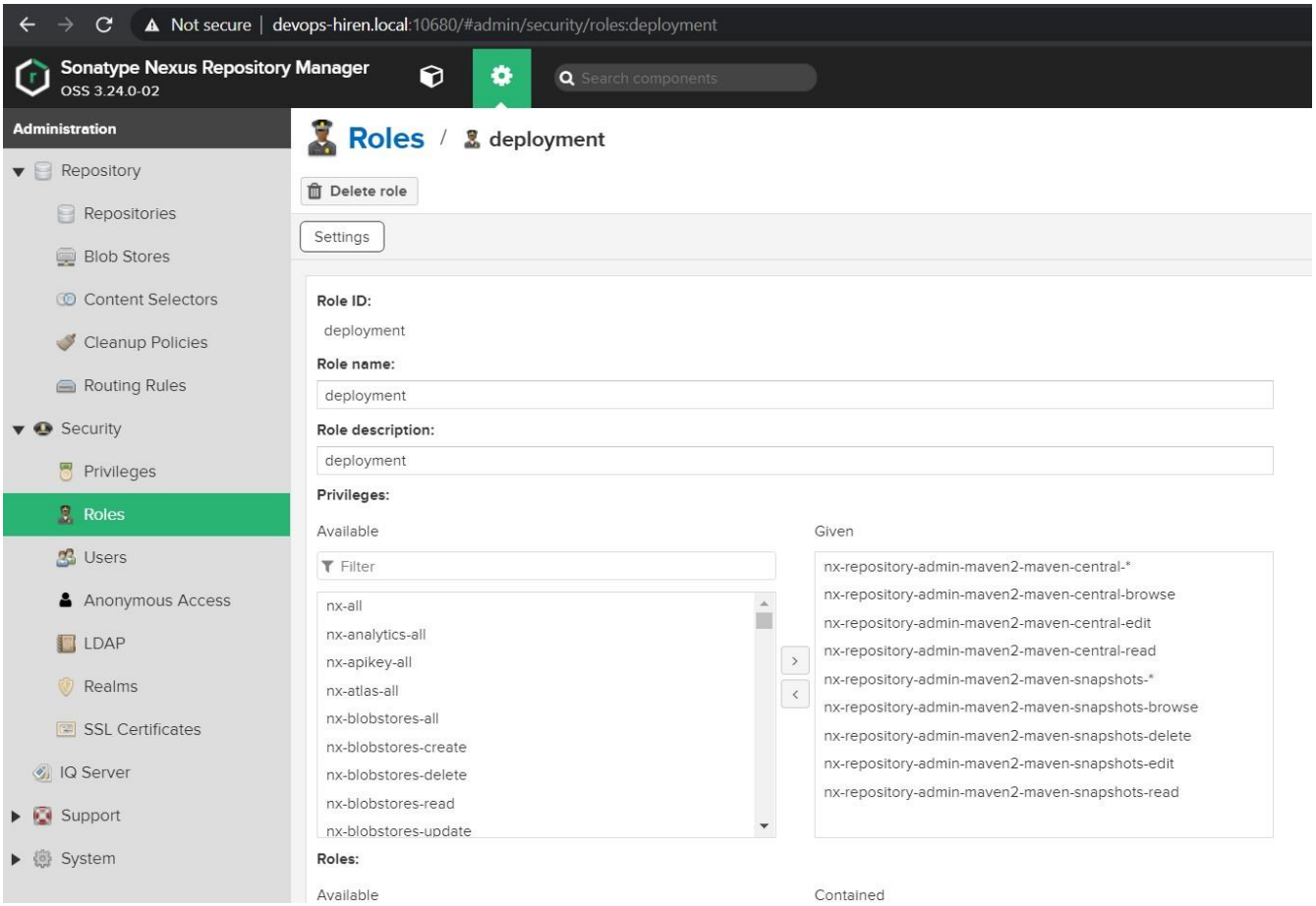
- Save the file go to the directory where docker-compose.yml is and run the following command: “docker-compose up -d gitea”
- Visit browser and add you ipaddress add port to run nexus live “192.168.0.102: 10180”
- Your gitea is setup and running…



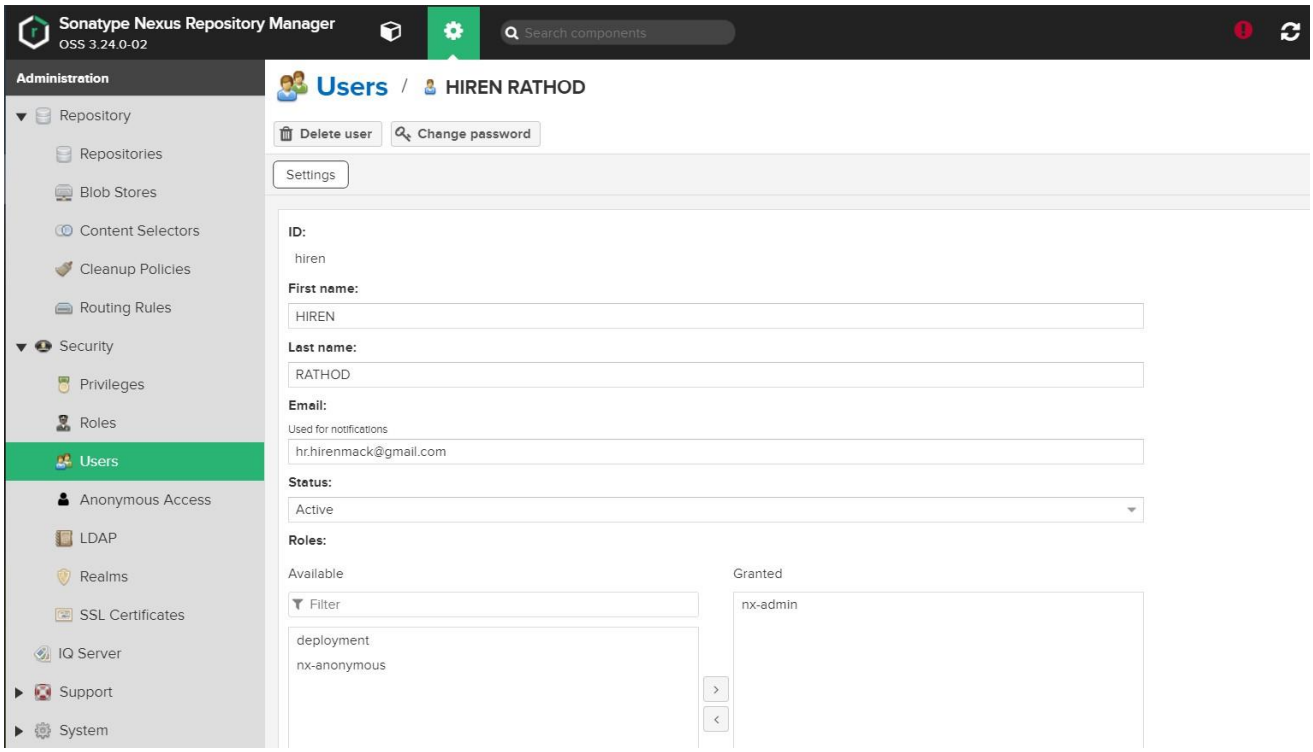
⇨ Building, Testing and deploying java maven project with Jenkins and Nexus3

Step 1:

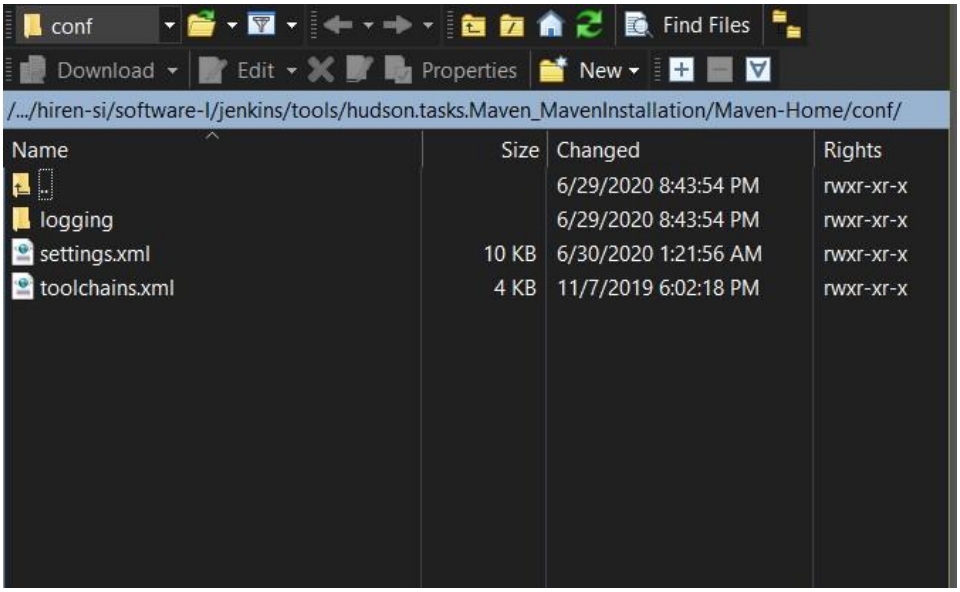
- Open nexus and Jenkins in your browser on the following ports which we binded in our docker-compose.yml file and login with the following credentials as shown above.
- To deploy your maven project and push it to nexus repository manager you need to do the following procedure first:
  - Create your role in nexus



- Create a user and add your role to the user to handle all your work



- Browsers to Jenkins-data folder in CentOS => goto tools => config => open settings.xml file

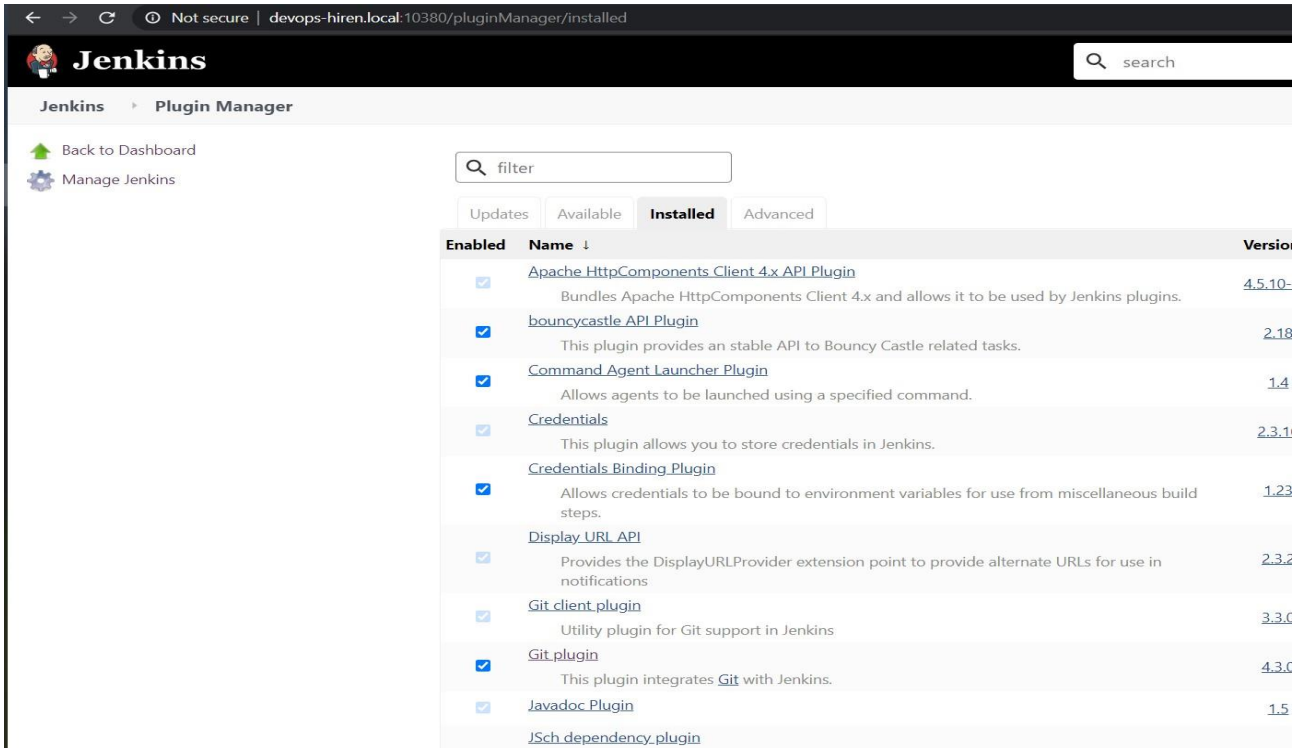


- Add user ID and password that you created in nexus into the file as shown below

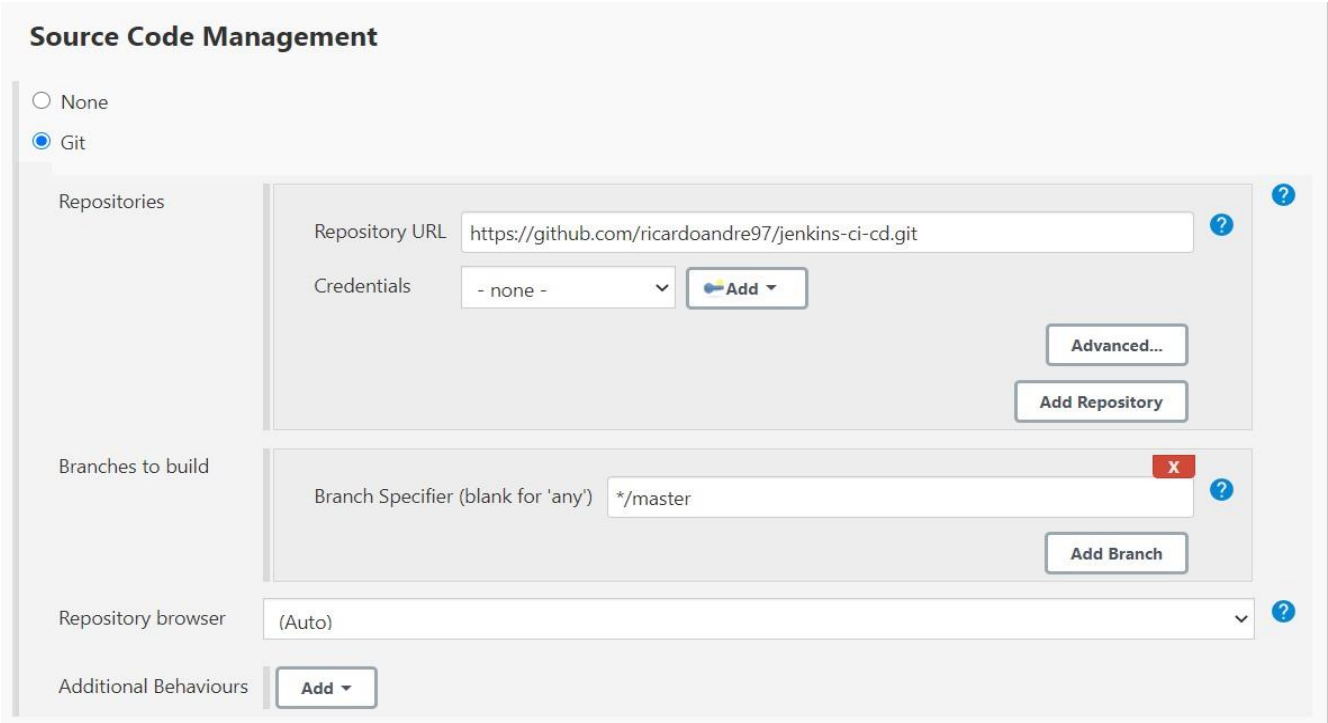
```
<servers>
  <!-- server
   | Specifies the authentication information to use wh
   | a unique name within the system (referred to by th
   |
   | NOTE: You should either specify username/password
   |       used together.
   |-->
  <server>
    <id>hirenRepo</id>
    <username>hiren</username>
    <password>Hiren@54321</password>
  </server>
```

- Go to Jenkins Add a new job as a maven project.
- Go to manage plugins section in Jenkins And install the required plugins from available section in it.

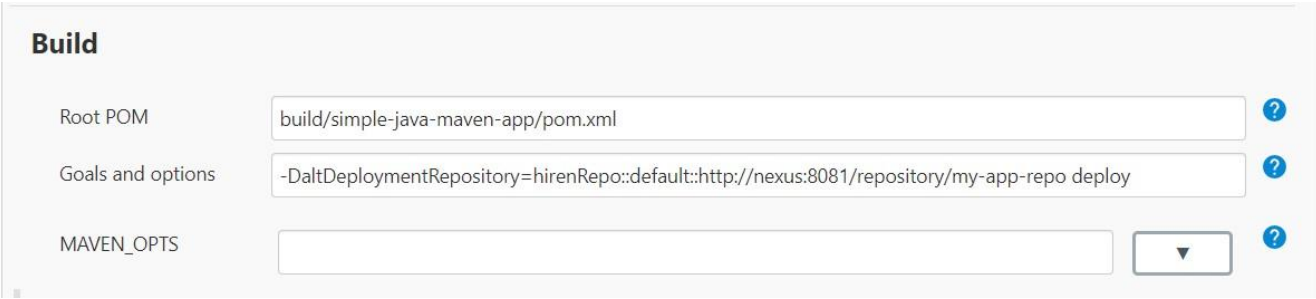
1. Git & Maven



- Click on the project and open config project
- In the SCM section select Git and add your GitHub URL in it.



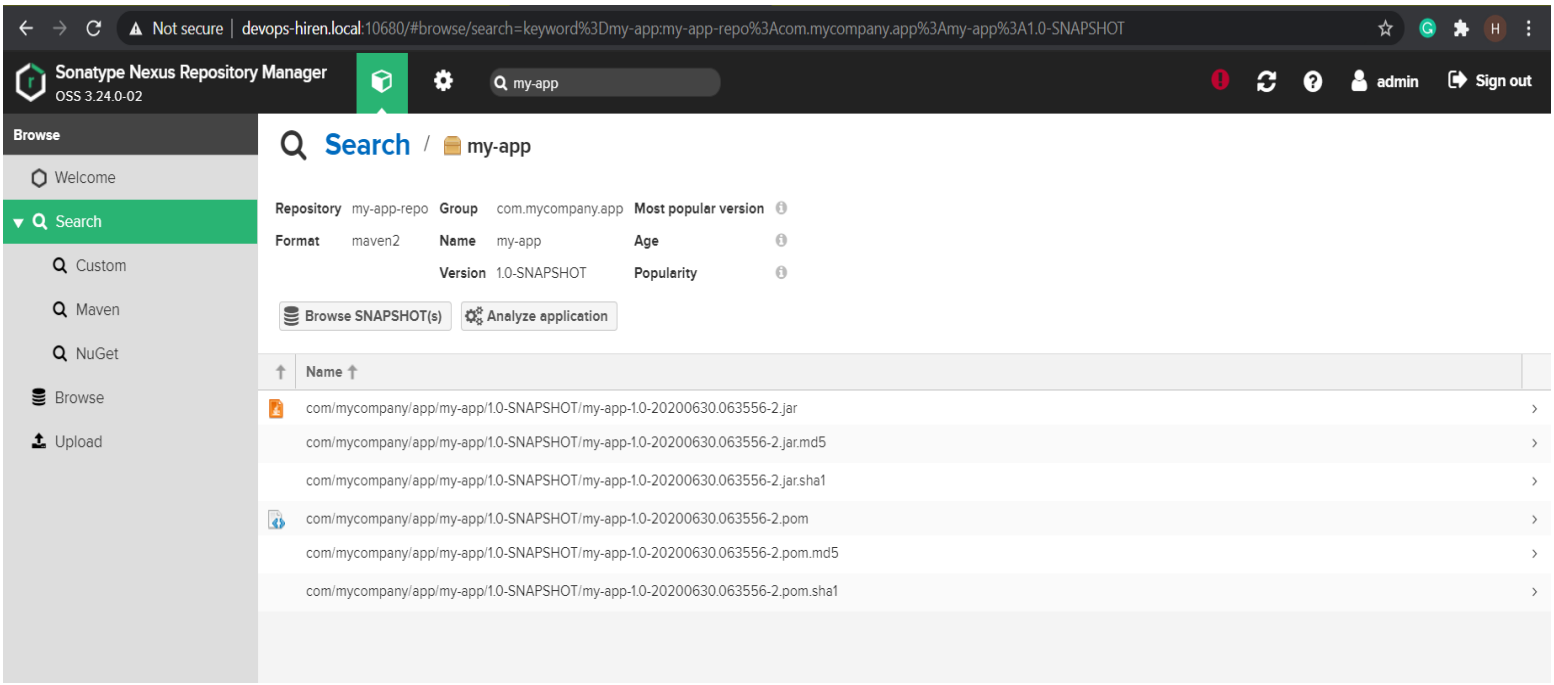
- Go to Build section and add the following details



- In the image above hirenRepo is the ID that we created in settings.xml file for nexus repository
- http://nexus:8081/repository/my-app-repo is the path where we want to push the deployed jar/war/pom file

- "Deploy" is a maven command to deploy the project.

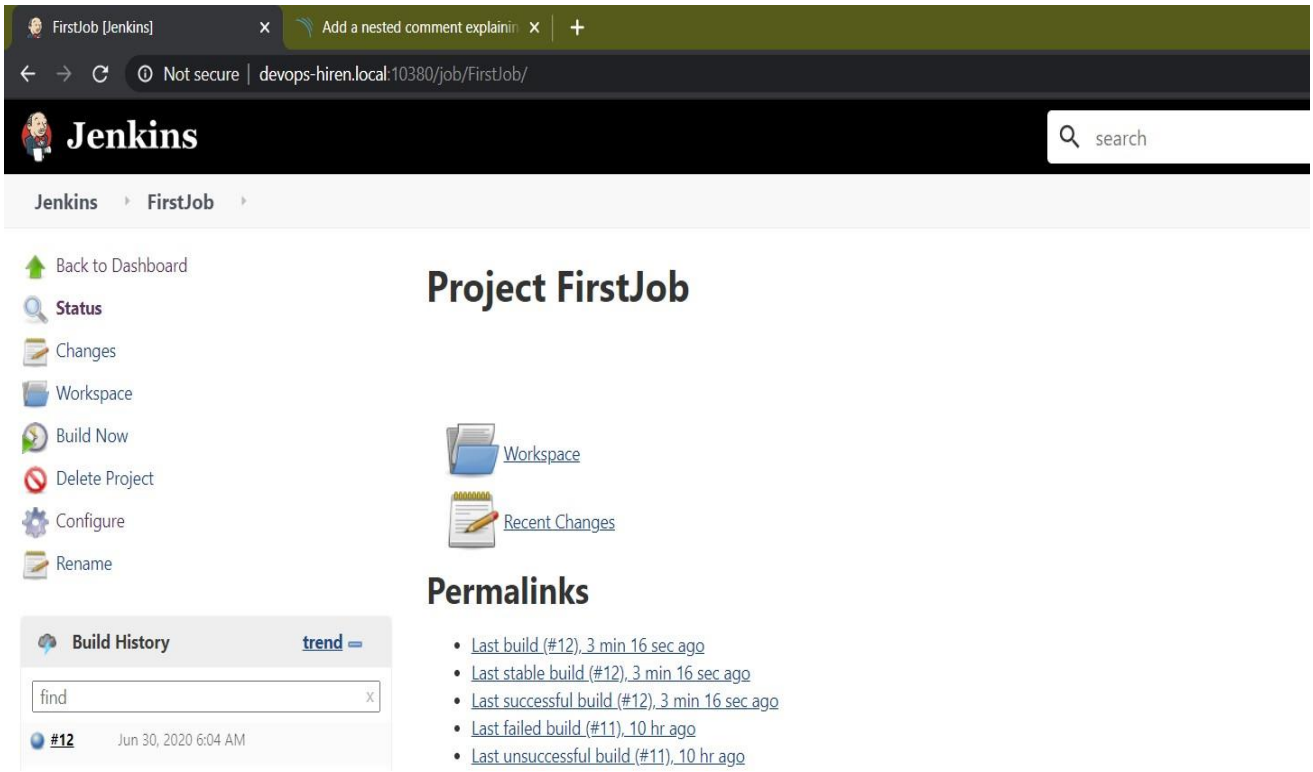
⇨ Result



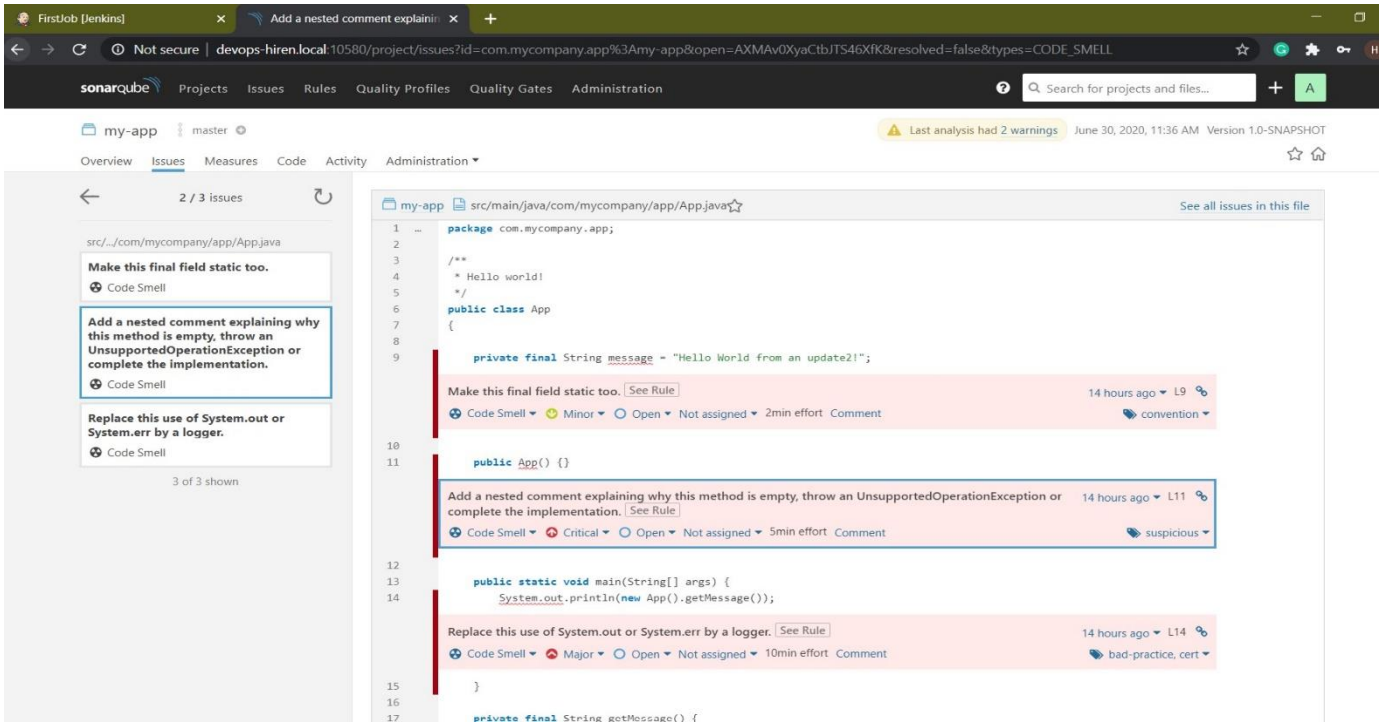
⇨ Building, Testing, analyzing your Java Maven project using Jenkins and SonarQube.

- Open sonarqube and Jenkins in your browser on the following ports which we binded in our docker-compose.yml file and login with the following credentials as shown above.

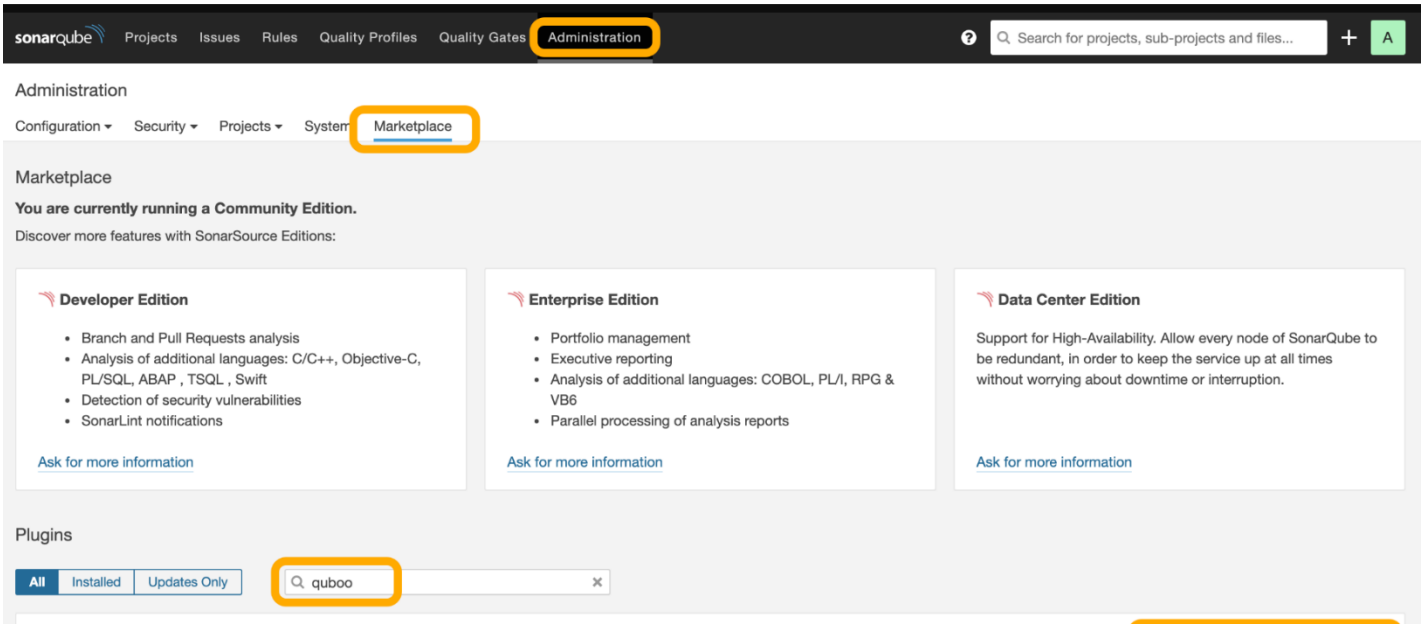
Jenkins:



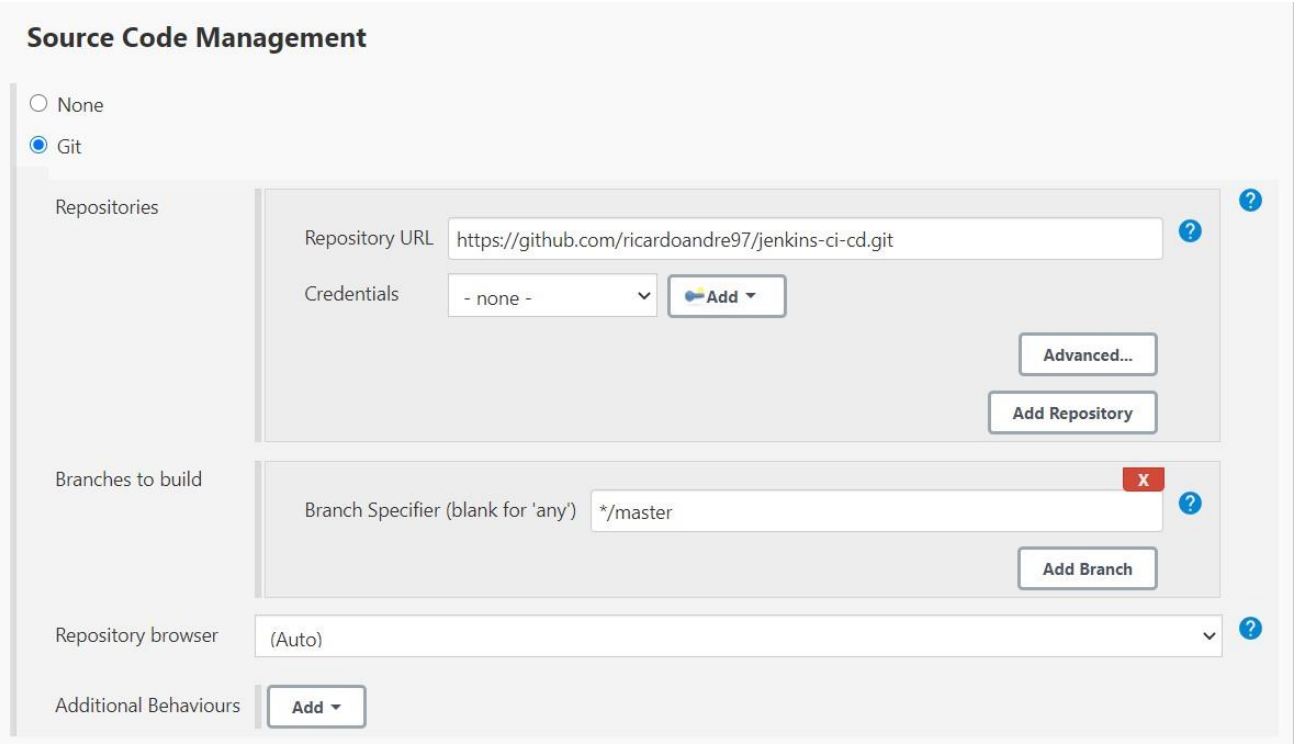
SonarQube



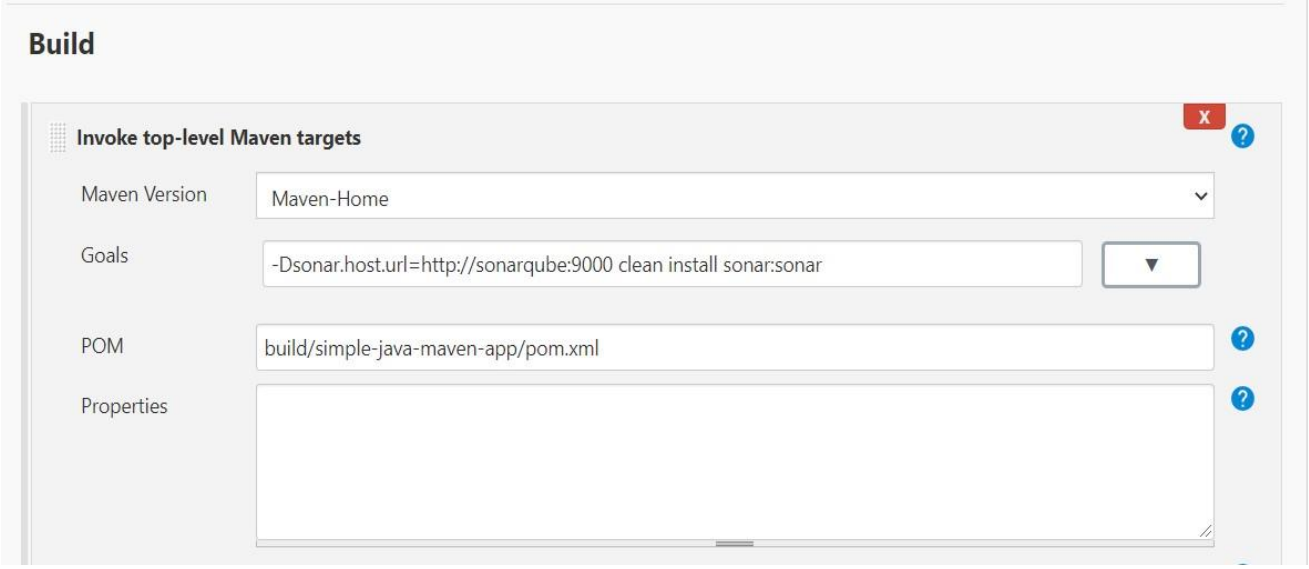
- Click on Administration tab then Click on Marketplace and install the required plugins.
- For example for maven I have installed "SonarJava & Bugs "



- Click on the project and open config project
- In the SCM section select Git and add your GitHub URL in it.

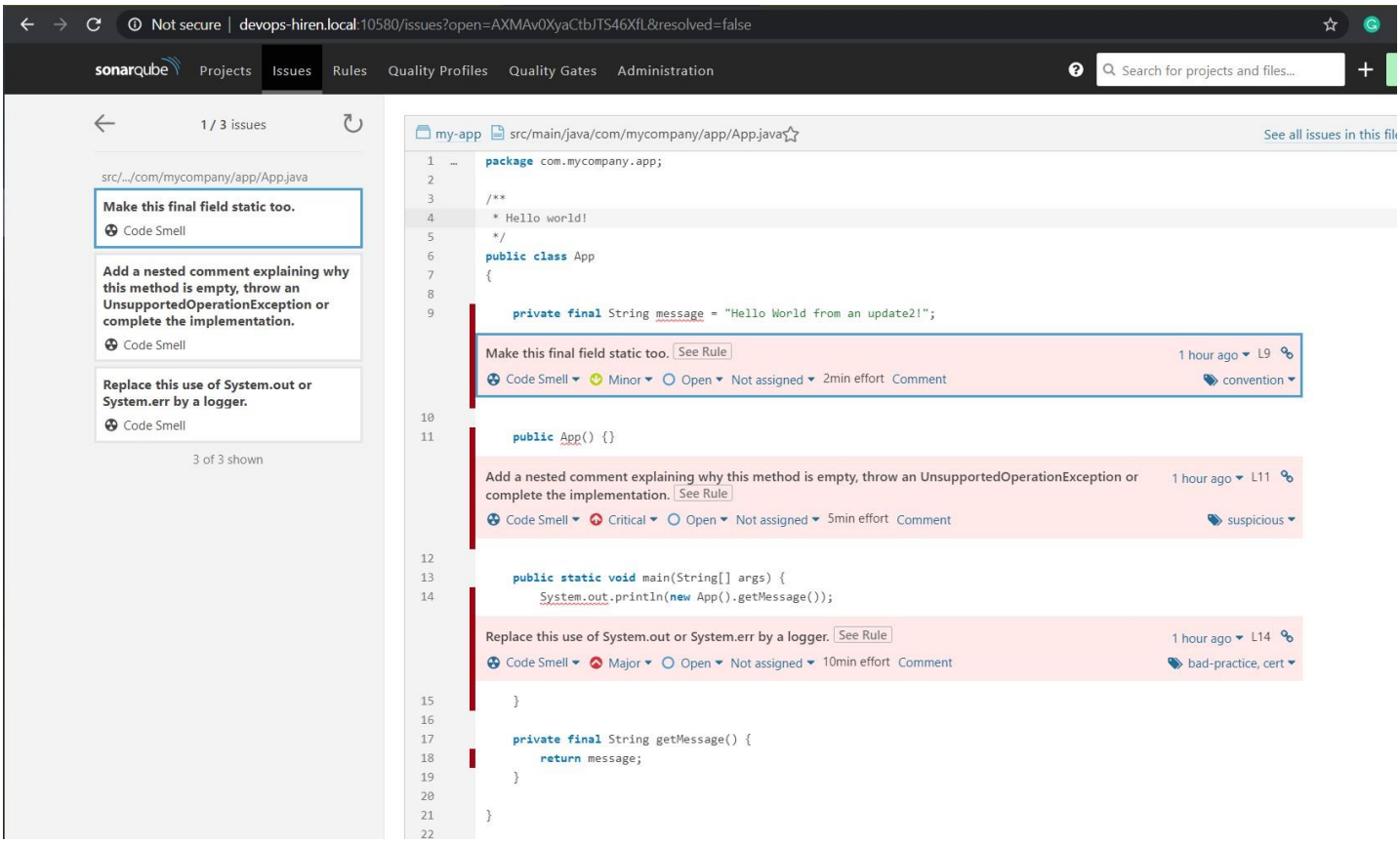


- Go to Build section and add the following details



- Save the project and click on Build to see the magic

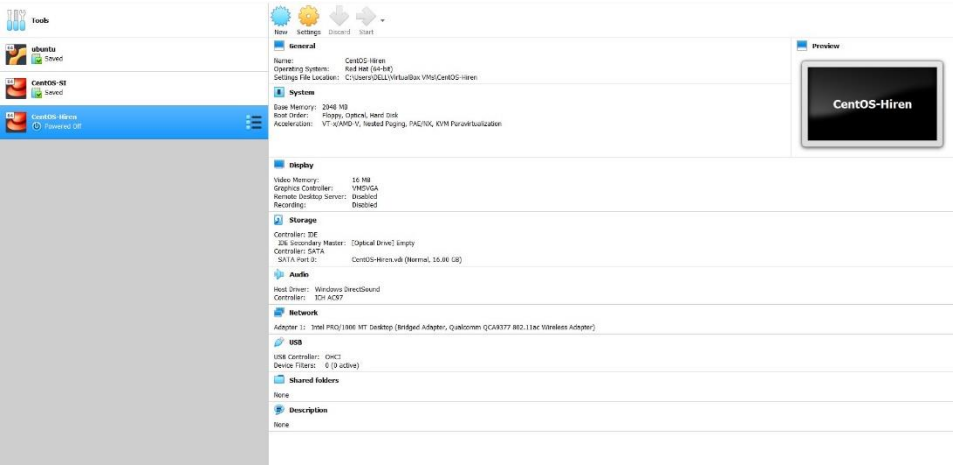
- Visit sonarqube and hit refresh and you can see your project is uploaded and analyzed



Whole Docker-Compose YAML FILE Review

hiren-si@localhost:~/software-I

```yaml
version: '3.5'
services:
  jenkins:
    image: "jenkins/jenkins:lts"
    ports:
            - "10380:8080"
    restart: "always"
    volumes:
            - "./jenkins:/var/jenkins_home"

  nexus:
          image: "sonatype/nexus3"
          ports:
                  - "10680:8081"
          volumes:
                  - "./nexus:/nexus-data"
  gitea:
          image: "gitea/gitea:latest"
          ports:
                  - "10122:22"
                  - "10180:3000"
          volumes:
                  - "./gitea:/data"
  sonarqube:
          image: "sonarqube:lts"

          ports:
                  - "10580:9000"

          volumes:
                  - "./sonarqube/conf:/opt/sonarqube/conf"

                  - "./sonarqube/data:/opt/sonarqube/data"

                  - "./sonarqube/logs:/opt/sonarqube/logs"

                  - "./sonarqube/extensions:/opt/sonarqube/extensions"

                  - "./sonarqube/bundled-plugins:/opt/sonarqube/lib/bundled-plugins"
~
~
```



Thank You