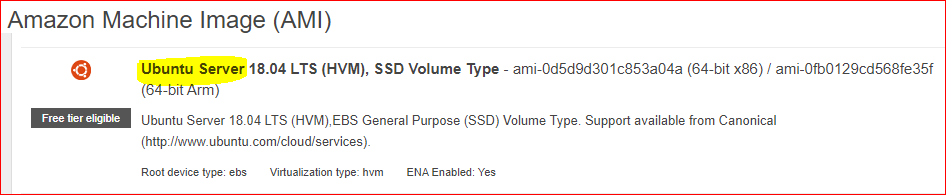
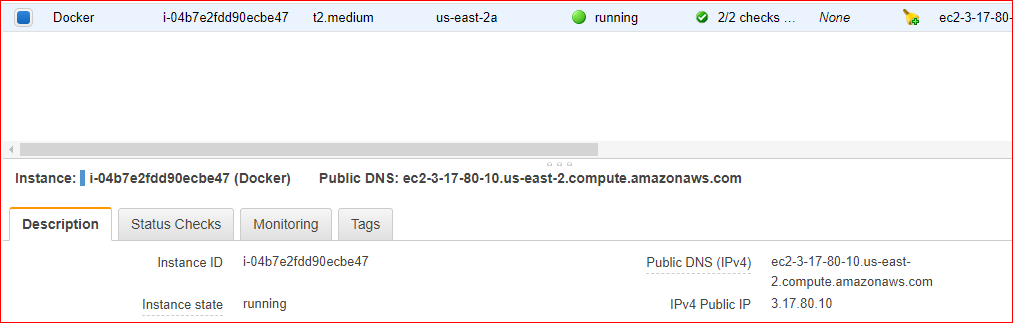
**Assignment**

# CICD tools installation and Creating pipeline Using Docker

**Prerequisites for installing Docker:**

1. **Create an EC2 instance t2-medium– Ubuntu in AWS:**





1. **Create an account on Docker Hub**

<https://hub.docker.com/>

-------------------------------------------------------------------------------------------------------------

**Install Docker Engine – Community Edition:**

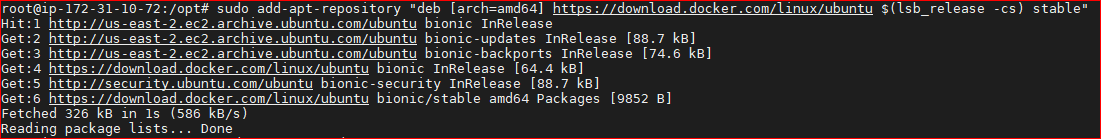
1. Add GPC of the official docker repository:

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –**



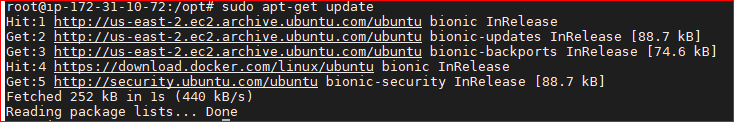
1. Add Docker repository to APT source:

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**



1. Update the package database:

**sudo apt-get update**



1. Now install Docker CE by using below command:

**sudo apt-get install -y docker-ce**

Now verify that the docker is installed successfully by using below command:

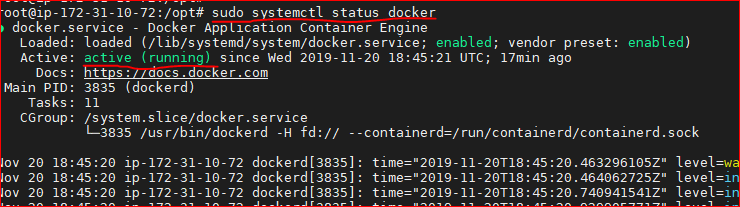
**docker - -version**

We should be able to see the version as below, then it is successfully installed:



1. To check if docker is running:

**sudo systemctl status docker**



-------------------------------------------------------------------------------------------------------------

**Jenkins Image Installation from docker:**

1. First create volume by using below URL:

Volumes are created to persists data generated by the docker container. Volumes data are store on remote host or cloud provider and it exists outside the life cycle of a container, so even if the container is killed or removed, the volume still exists.

root@ip-172-31-12-32:/opt# **docker volume create tomcat**

tomcat

root@ip-172-31-12-32:/opt# **docker inspect tomcat**

**[inspect command is used to verify that the volume is created as shown below]**

[

{

"CreatedAt": "2019-11-22T06:05:48Z",

"Driver": "local",

"Labels": null,

"Mountpoint": "/var/lib/docker/volumes/tomcat/\_data",

"Name": "tomcat",

"Options": null,

"Scope": "local"

}

]

1. Install the latest jenkins

**docker pull jenkins/jenkins:lts**

* Docker pull command pulls an image or a repository from a docker registry

**docker run -d --name=jenkins -p 8090:8080 -p 9093:9093 -p 9094:9094 -v tomcat:/var/workspace -u root jenkins/jenkins:lts**

**8090 – jenkins**

**9093 – INGFavAccount (backend)**

**9094 – INGFavBank (backend)**

**Options used:**

**--d 🡪** detaches the process and runs it in the background, otherwise we won’t be able to use the terminal until we killed it.

**--name=jenkins 🡪** it is the name given to the container. If the name is not given, then the random name will be assigned to the container.

**-p 8090:8080 🡪** sets up a port forward. Jenkins container is listening to 8080 by default. This flag maps the 8080 to port 8090 on the host system.

**-v tomcat:/var/workspace** 🡪 creates a volume name tomcat

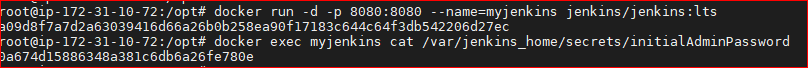
**-u root** 🡪 running jenkins container as root

**jenkins/jenkins:lts 🡪** spedifies that the container should be build from jenkins/jenkins:lts image available in docker hub

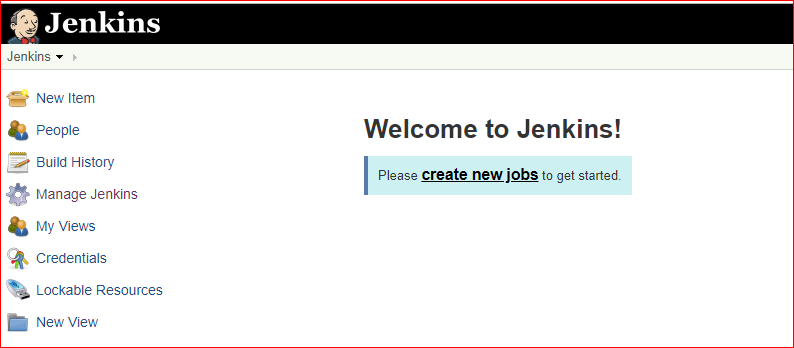
<http://3.17.77.132:9080/>

**docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword**

[to get the password for installing jenkins]

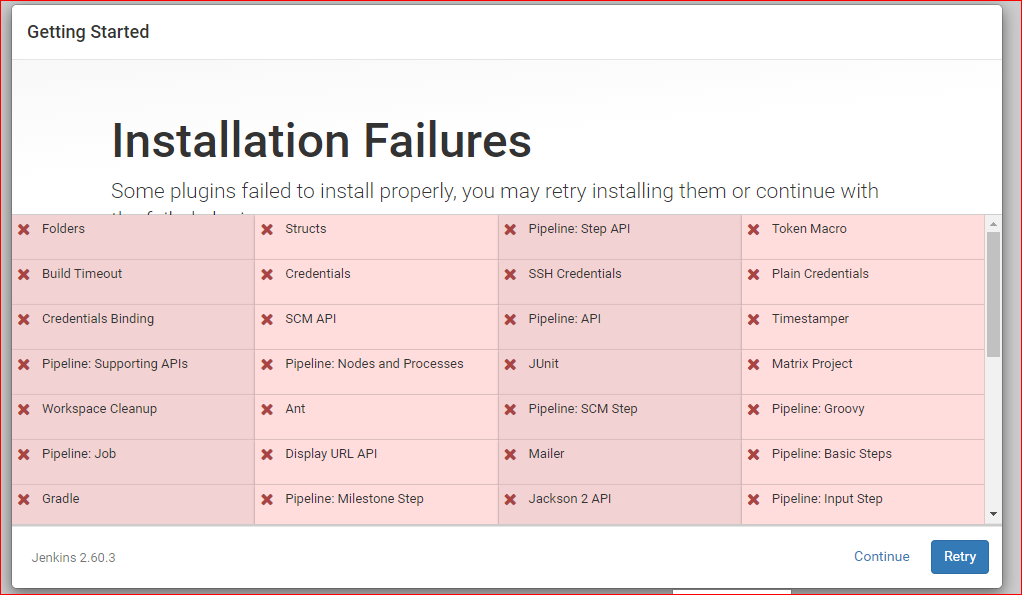


<http://ubuntu-IP:8080/>



**Jenkins Installation failure:**

Sometimes we encountered the below failure because it is not the latest jenkins image and had some issue with the one we installed, always look for latest to download if you get the below error:



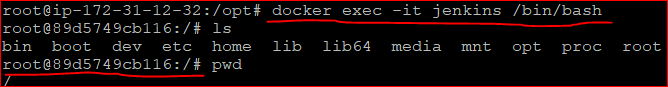
-------------------------------------------------------------------------------------------------------------

**Install maven and nodejs inside jenkins container**

1. First go inside jenkins container by using below command:

**docker exec –it jenkins /bin/bash or docker exec –it jenkins /bin/sh**

[docker exec command runs a new command in the running container]



Install maven

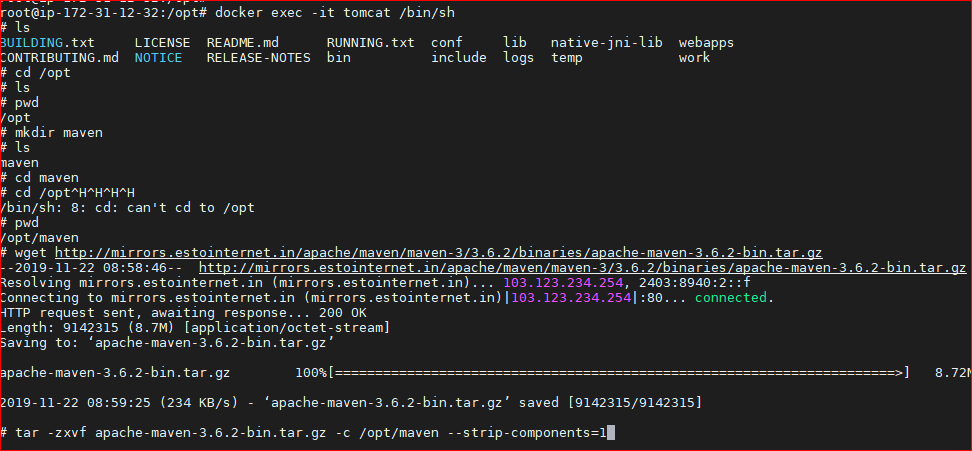
**wget** [**http://apachemirror.wuchna.com/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz**](http://apachemirror.wuchna.com/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz)

**cd /opt**

**mkdir maven**

**tar -zxvf apache-maven-3.6.2-bin.tar.gz -C /opt/maven --strip-components=1**

**cd maven**



Install nodejs

**cd /opt**

**apt-get -y update**

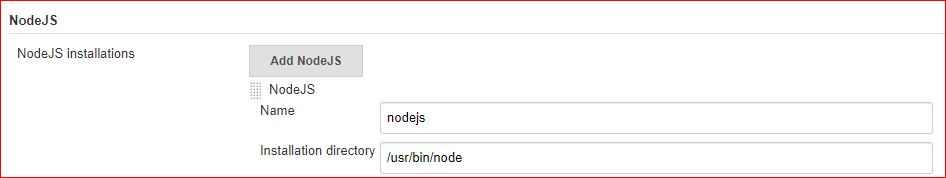
**curl -sL https://deb.nodesource.com//setup\_10.x | bash**

**apt-get install -y nodejs**



**To integrate with Jenkins:**

**Manage Jenkins🡪Global Tools Configuration**



-------------------------------------------------------------------------------------------------------------

**Install tomcat container**

1. Run command pulls the jenkins image from the docker hub and executes it:

**docker run -d --name tomcat -v tomcat:/usr/local/tomcat/webapps -p 8888:8888 tomcat**

1. Copy the files to be modified from container to host and once changes are done copy them back to the container by using the below commands:
2. **Context.xml**

Copy from container to host:

**docker cp tomcat:/usr/local/tomcat/webapps/host-manager/META-INF/context.xml context.xml**

**sudo vi context.xml**

<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->



Copy from host to container:

**docker cp context.xml tomcat:/usr/local/tomcat/webapps/host-manager/META-INF/context.xml**

1. **Context.xml**

Copy from container to host:

**docker cp tomcat:/usr/local/tomcat/webapps/manager/META-INF/context.xml context.xml**

**sudo vi context.xml**

<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->

Copy from host to the container:

**docker cp context.xml tomcat:/usr/local/tomcat/webapps/manager/META-INF/context.xml**

1. **tomcat-users.xml**

Copy from container to host:

**docker cp tomcat:/usr/local/tomcat/conf/tomcat-users.xml tomcat-users.xml**

**sudo vi tomcat-users.xml**

**Add below roles and user credentials**

<role rolename="admin-gui"/>

<role rolename="admin-script"/>

<role rolename="manager-gui"/>

<role rolename="manager-status"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<user username="admin" password="admin" roles="manager-gui,admin-gui,manager-script"/>

Copy from host to the container:

**docker cp tomcat-users.xml tomcat:/usr/local/tomcat/conf/tomcat-users.xml**

1. **server.xml**

Copy from container to host

**docker cp tomcat:/usr/local/tomcat/conf/server.xml server.xml**

**vi server.xml**

<Connector port="8888" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

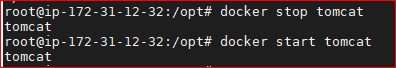
Copy back to container

**docker cp server.xml tomcat:/usr/local/tomcat/conf/server.xml**

1. Stop and start the container

**docker stop tomcat**

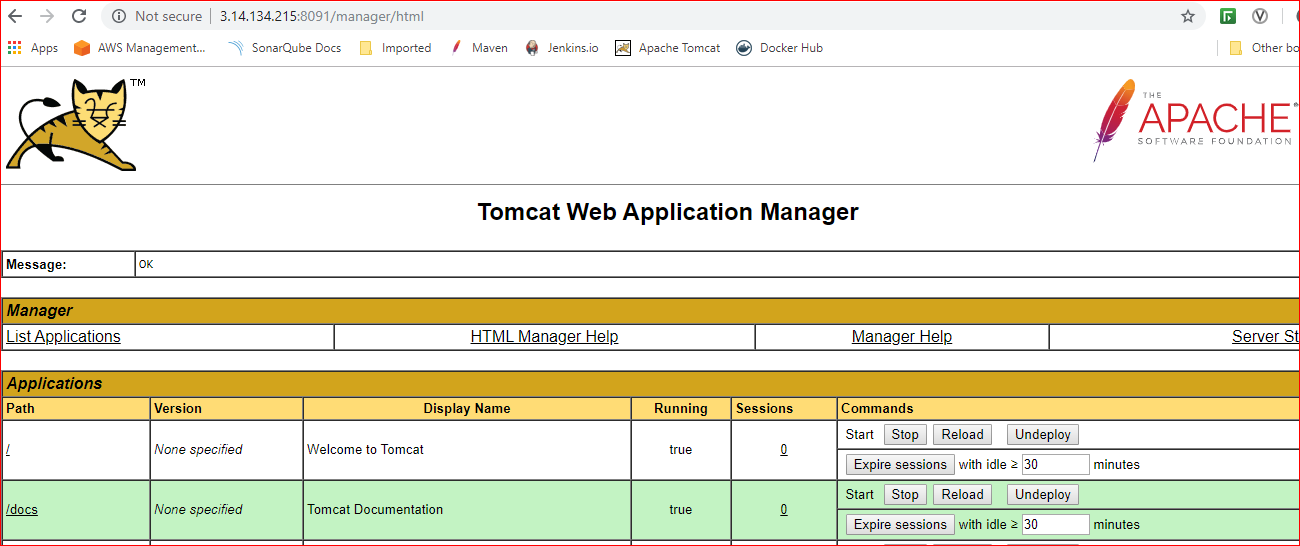
**docker start tomcat**



1. Now check if the tomcat is up and running by going to the below URL:

<http://3.14.134.215:8888/>

[http://Ubuntu\_IP:8888/]



1. Add below details in settings.xml file:

Go inside jenkins container:

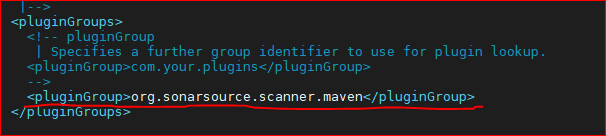
**docker exec -it jenkins /bin/sh**

**cd /opt/maven/conf**

**vi settings.xml**

**add below pluginGroup and server details:**

**<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>**

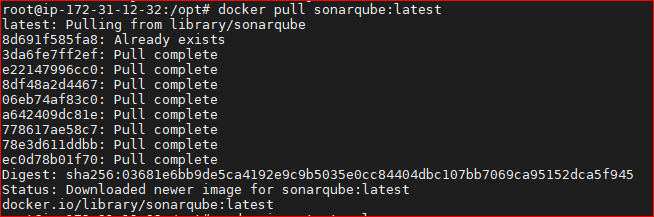


-------------------------------------------------------------------------------------------------------------

**Installing SonarQube**

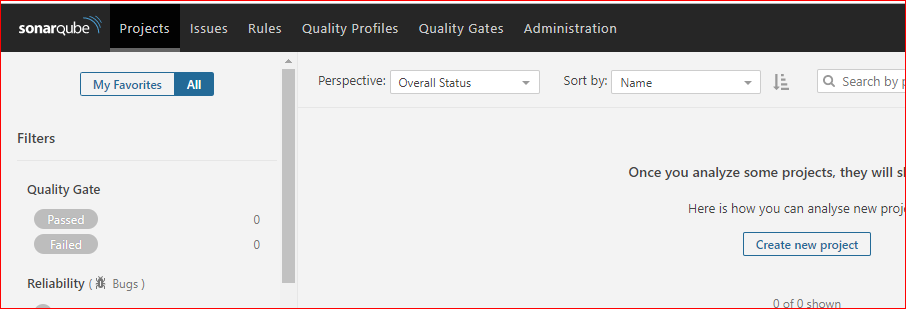
**docker pull sonarqube:latest**

**docker run -d --name sonarqube -p 9000:9000 sonarqube [-d option indicates it is ruuning in detached mode]**



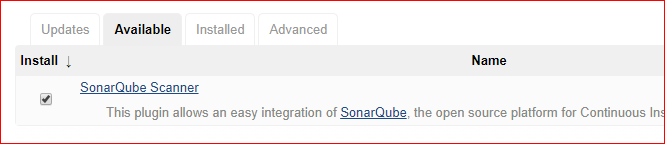


<http://3.14.134.215:9000/>

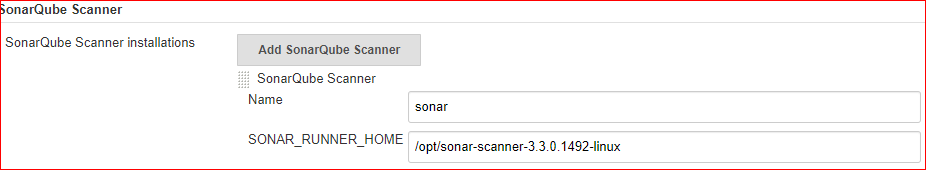


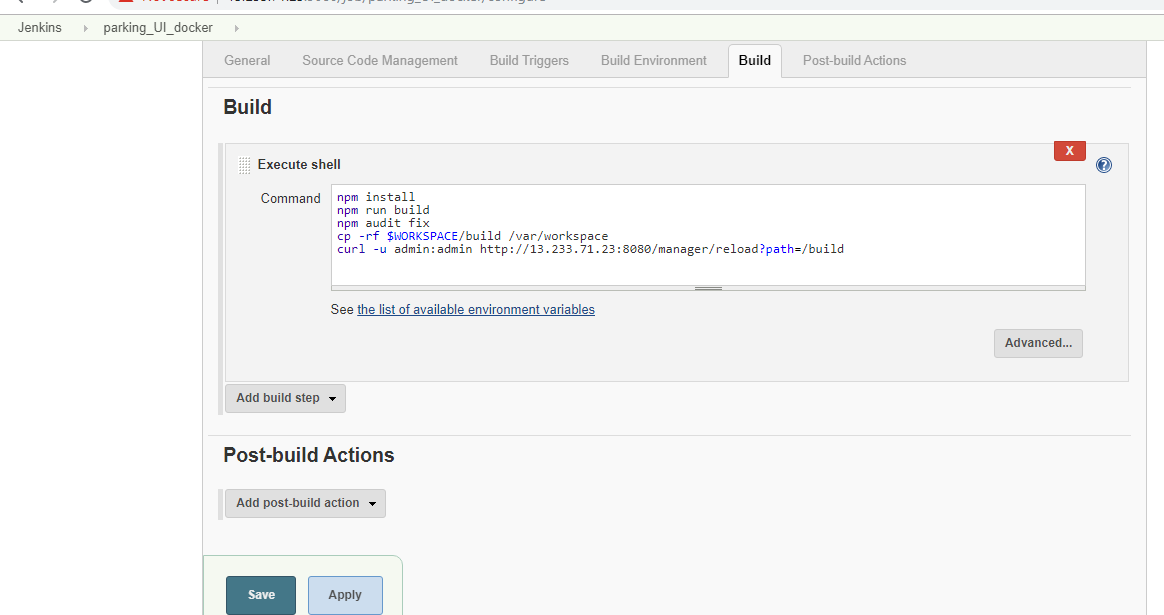
To integrate with Jenkins, add below plugins in jenkins:

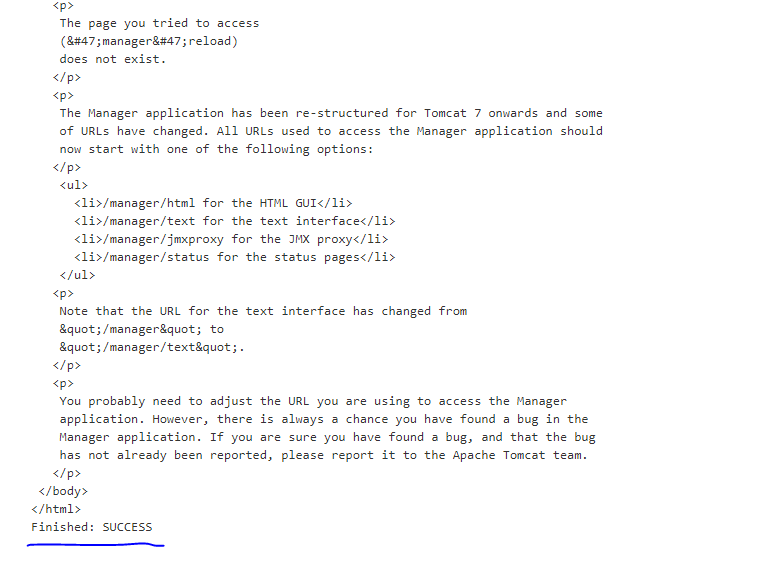
Manage Jenkins🡪Manage Plugins

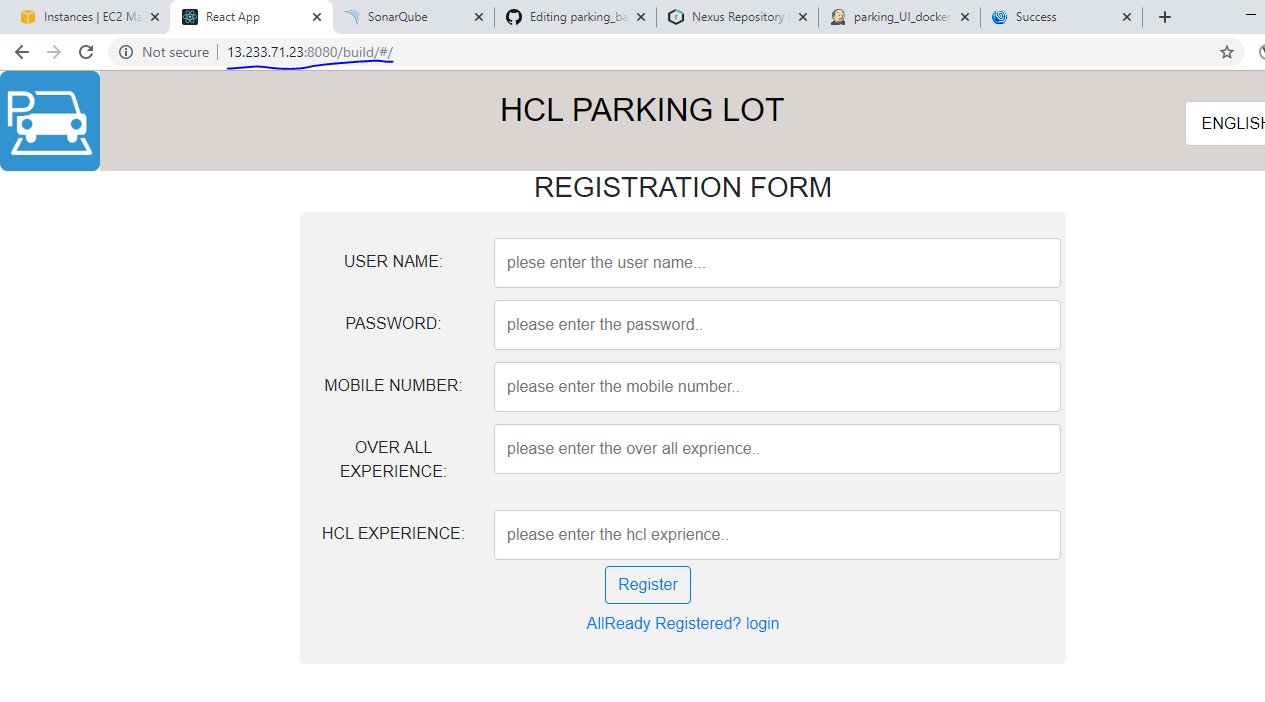


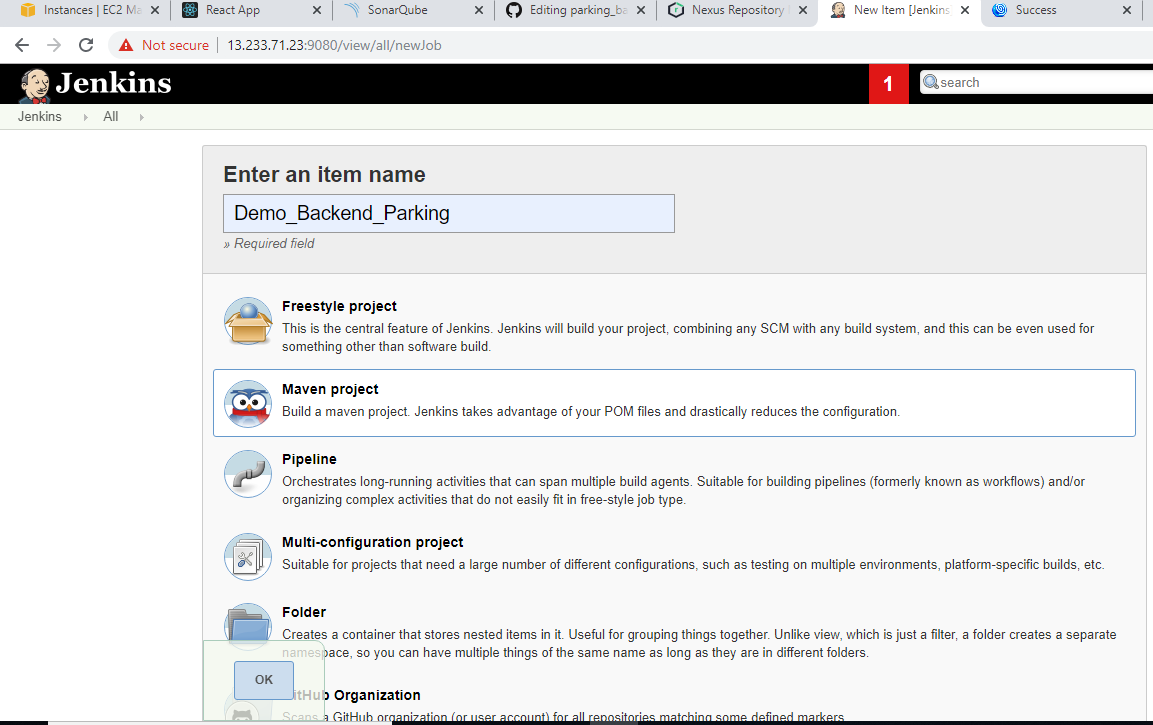
Manage Jenkins🡪Global Tools Configurations





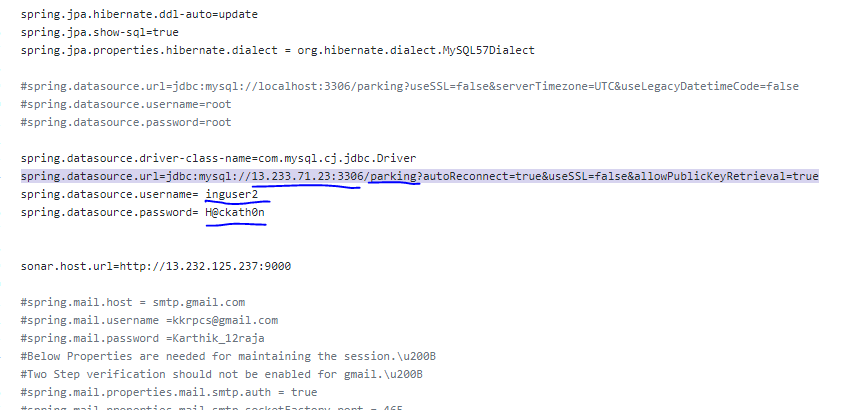






**Application.properties**

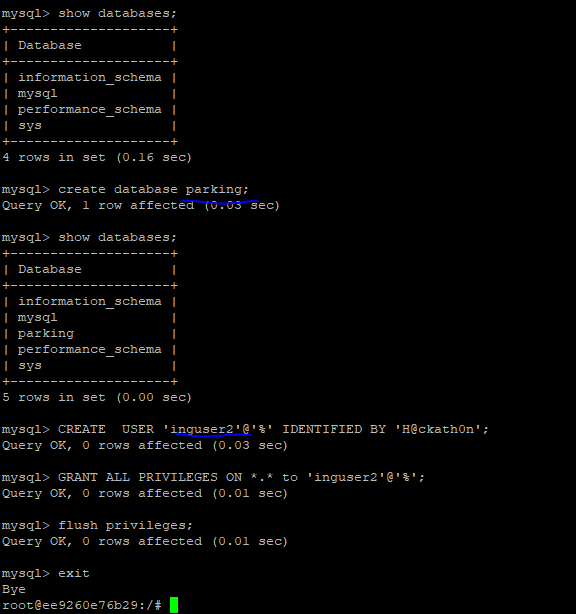
**spring.datasource.url=jdbc:mysql://13.233.71.23:3306/parking?autoReconnect=true&useSSL=false&allowPublicKeyRetrieval=true**



**Pom.xml**

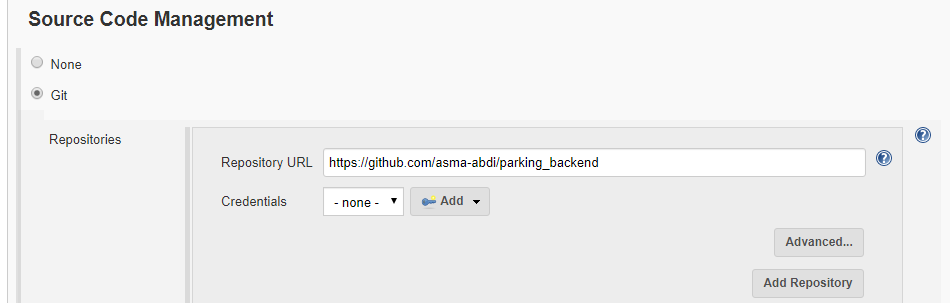


**Db:**



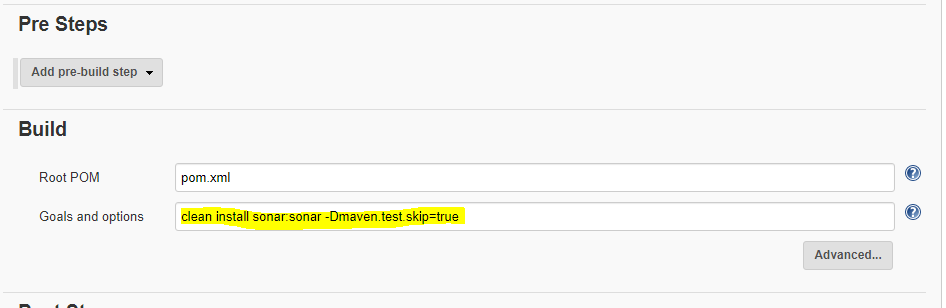
**SCM**

<https://github.com/asma-abdi/parking_backend>



**Pre-steps**

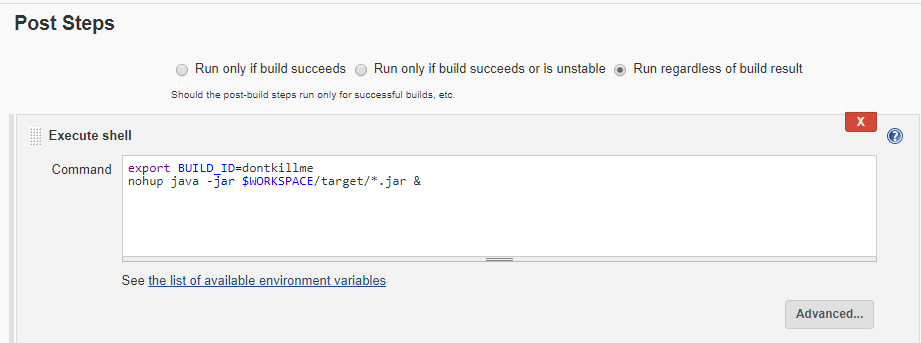
**clean install sonar:sonar -Dmaven.test.skip=true**



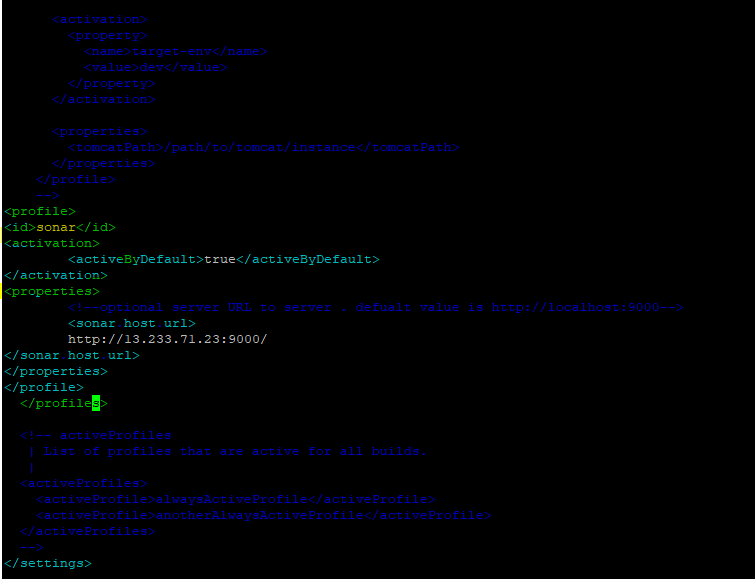
**Post-Steps**

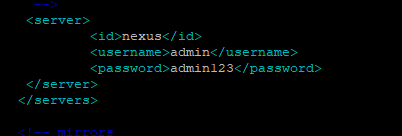
**export BUILD\_ID=dontkillme**

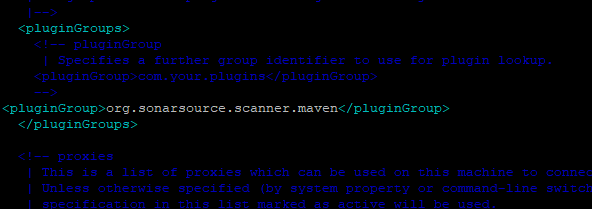
**nohup java -jar $WORKSPACE/target/\*.jar &**



**Settings.xml**



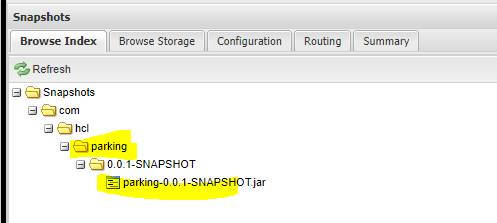




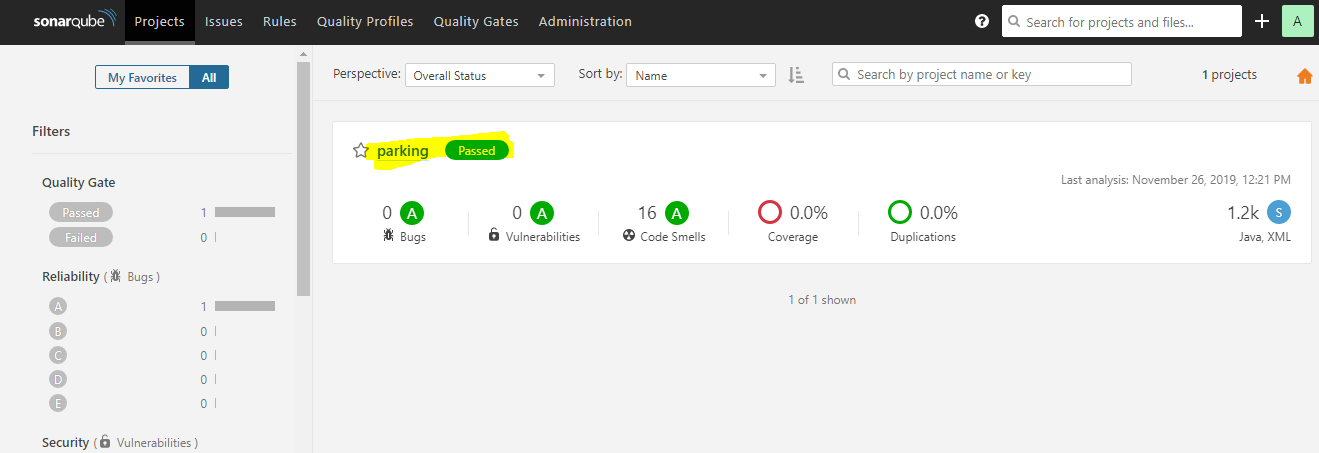
* **Run the Build-**



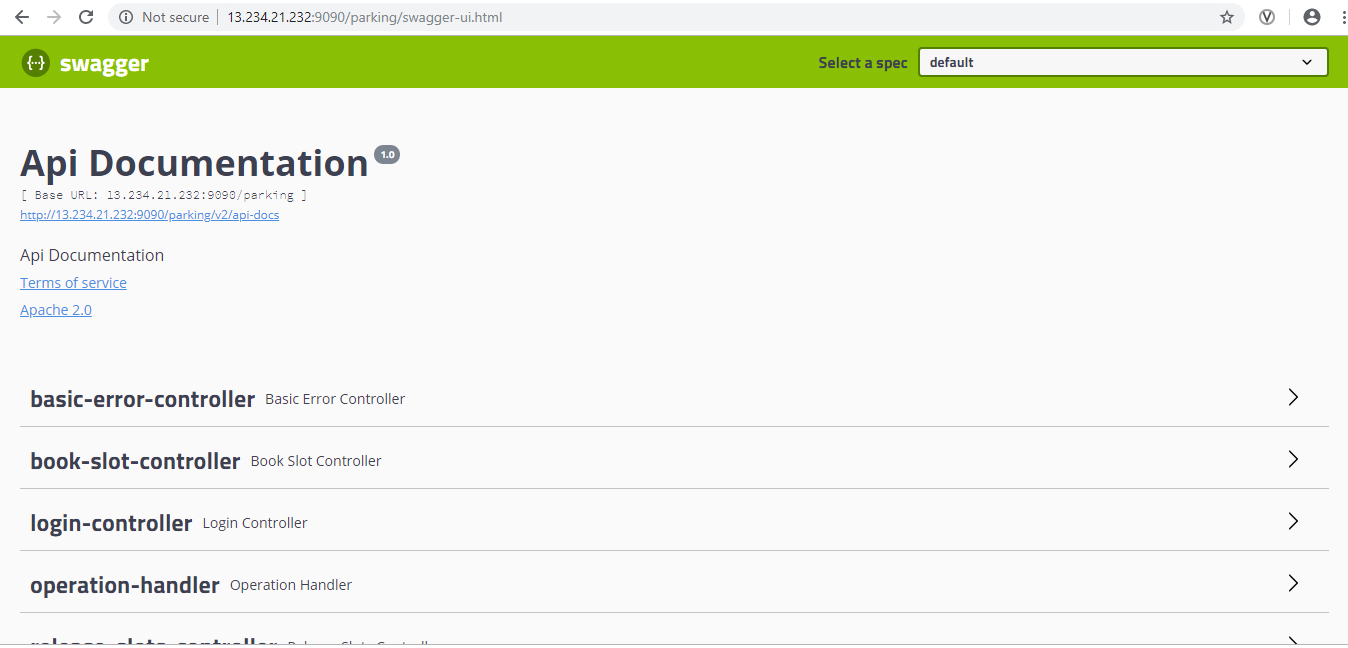
**Nexus:**



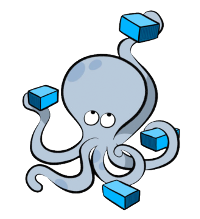
**Sonarqube:**



**Swagger**



**Docker/Compose**



Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services. Then, using a single command, you create and start all the services from your configuration. To learn more about all the features of Compose see [the list of features](https://github.com/docker/docker.github.io/blob/master/compose/overview.md#features).

Compose is great for development, testing, and staging environments, as well as CI workflows. You can learn more about each case in [Common Use Cases](https://github.com/docker/docker.github.io/blob/master/compose/overview.md#common-use-cases).

Using Compose is basically a three-step process.

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Lastly, run docker-compose up and Compose will start and run your entire app.

A docker-compose.yml looks like this:

version: '2'

services:

web:

build: .

ports:

- "5000:5000"

volumes:

- .:/code

redis:

image: redis

## **Docker Compose File**

Docker Compose File is a YAML file which contains details about the services, networks and volumes for setting up the Docker application.

Run the below command to find the version of your Docker Engine.

|  |  |
| --- | --- |
| 1 | docker -v |

Executing the command will return the version running at your host. Based on the version of Docker Engine at your host, download an appropriate version Docker Compose.

* Docker –v

🡪 Docker version 18.09.7, build 2d0083d



## **Install Docker Compose**

To download Compose, run the below set of commands.

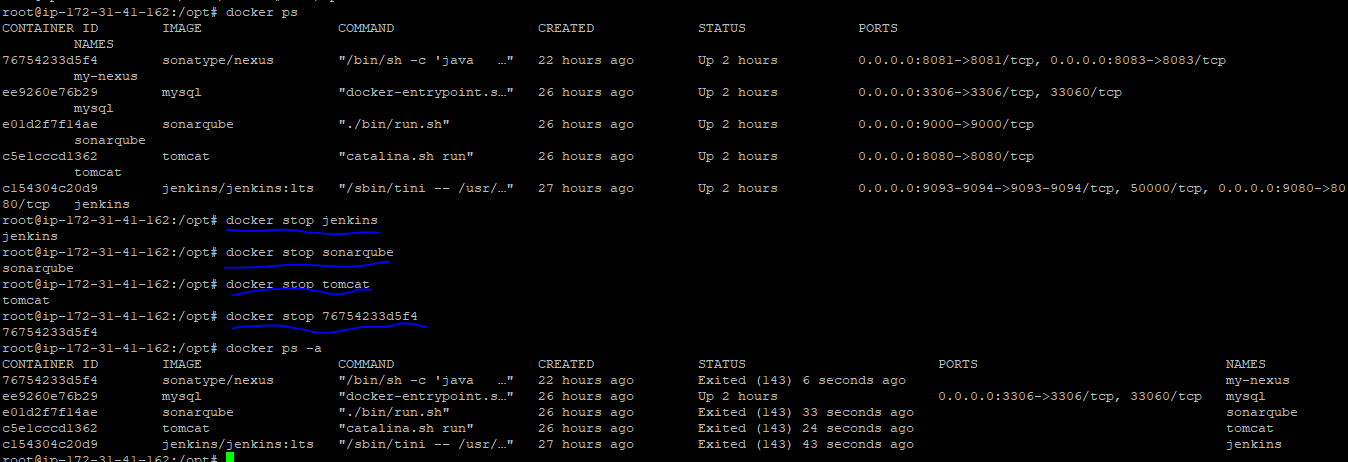
🡪sudo curl -L <https://github.com/docker/compose/releases/download/1.16.1/docker-compose->`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

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

* Write the YML file

🡪sudo vi /usr/local/bin/docker-compose.yml

* Before writing docker-compose.yml file, need to stop already running containers:



* ubuntu@ip-172-31-41-162:~$ sudo vi /usr/local/bin/docker-compose.yml

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

version: '2'

services:

jenkins:

image: jenkins/jenkins:lts

container\_name: jenkins1

ports:

- "8091:8080"

volumes:

- tomcat1:/var/workspace

tomcat:

image: tomcat

container\_name: tomcat1

ports:

- "8092:8080"

volumes:

- tomcat1:/usr/local/tomcat/webapps

mysql:

image: mysql

container\_name: mysql1

ports:

- "3307:3306"

environment:

- MYSQL\_ROOT\_PASSWORD=H@ckath0n

sonarqube:

image: sonarqube

container\_name: sonarqube1

ports:

- "9001:9000"

nexus:

image: sonatype/nexus

container\_name: my-nexus1

ports:

- 8081:8081

volumes:

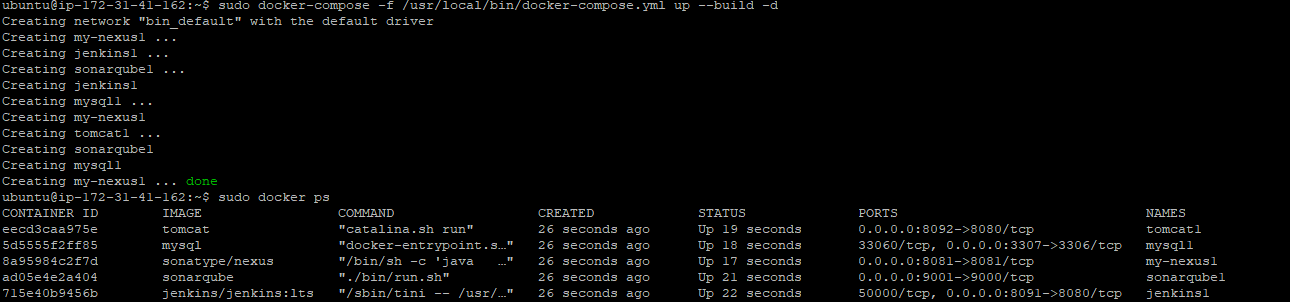
tomcat1:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Now we are at the conclusion of setting up a Compose. To start a Docker Compose and spin the all containers with all services, we need to simply execute the below commandsfrom the directory where the Docker Compose File (YAML file) is present:

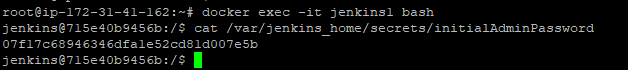
🡪ubuntu@ip-172-31-41-162:~$ sudo docker-compose -f /usr/local/bin/docker-compose.yml up --build –d

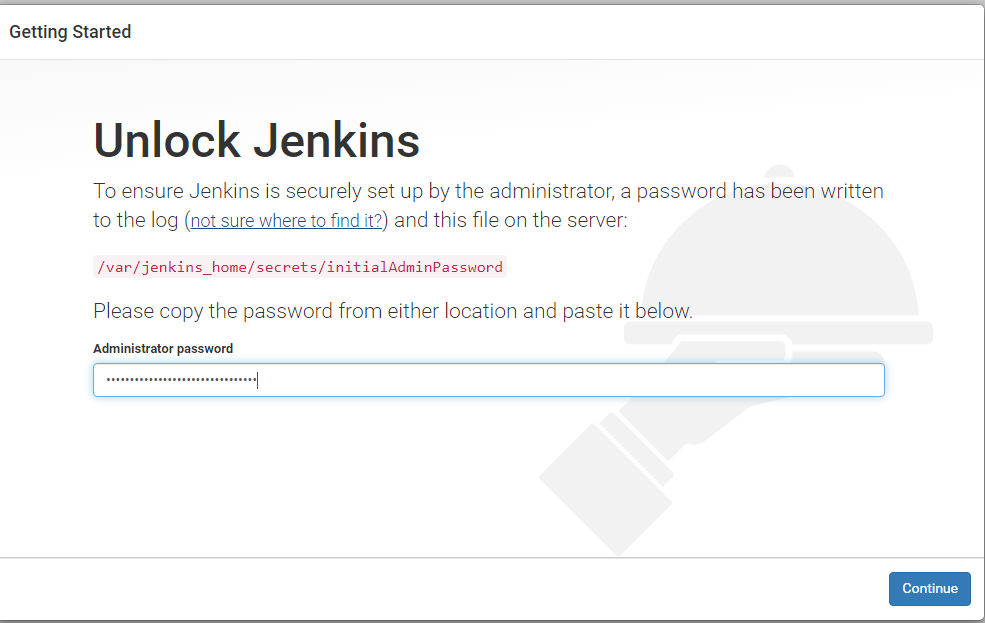
🡪docker ps

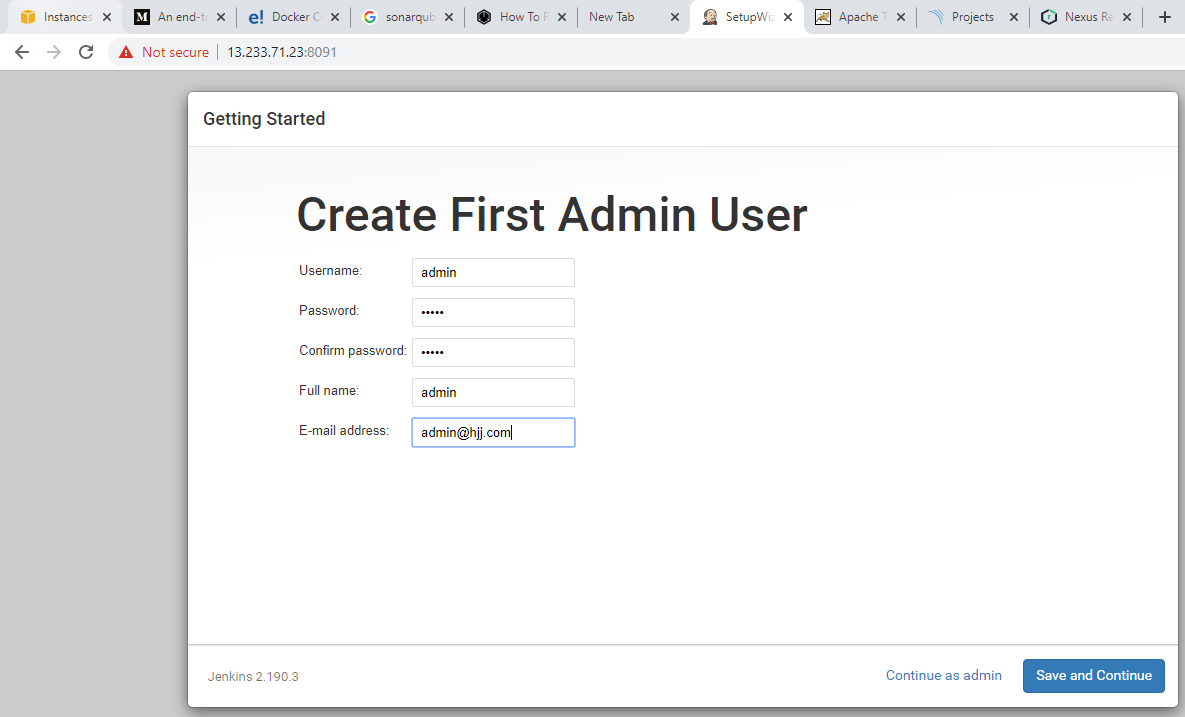


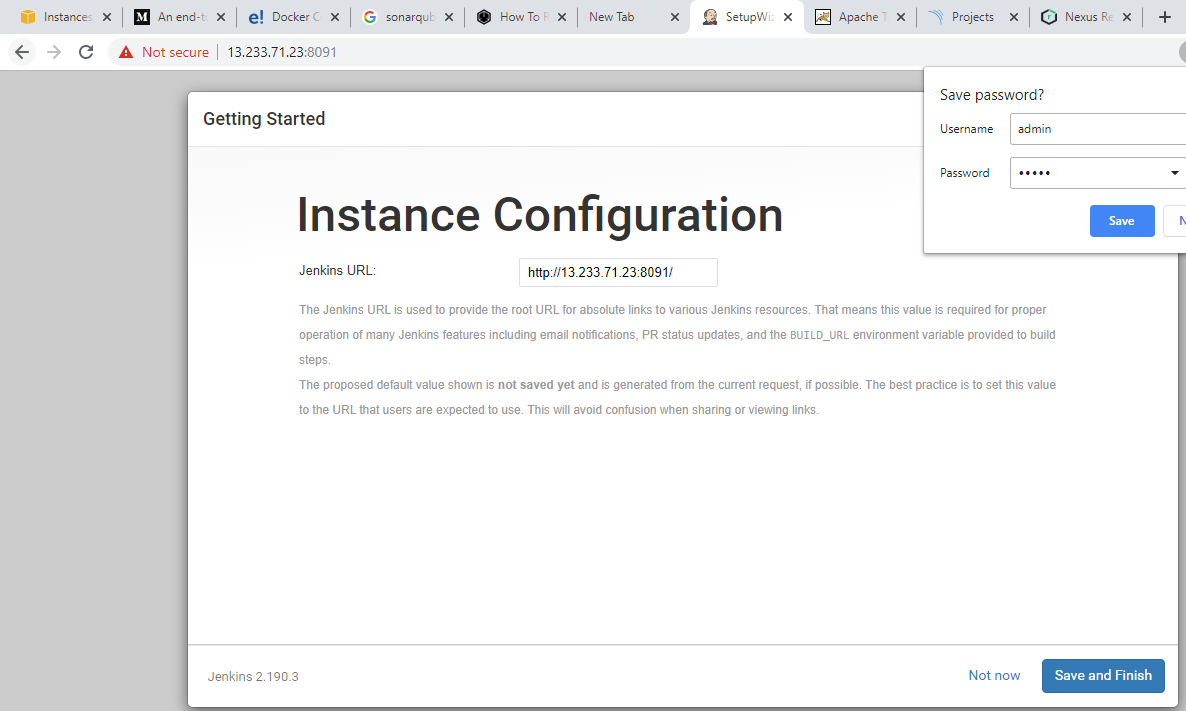
* **Now will check that all containers are up and running one-by-one-**

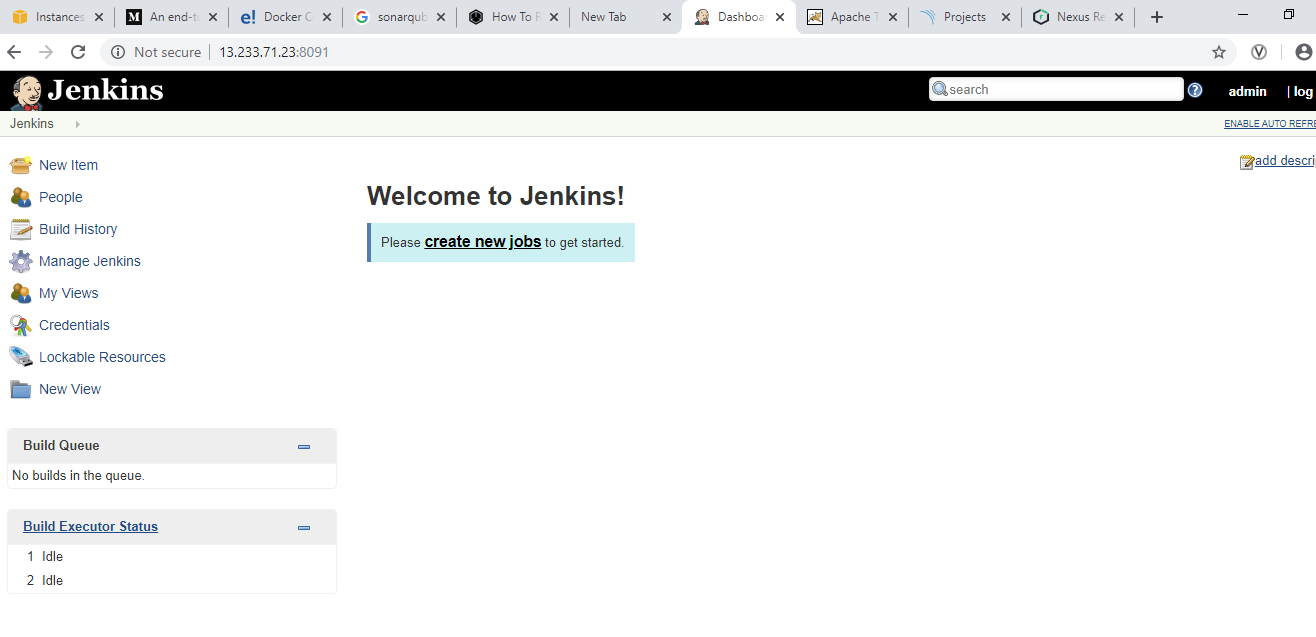
1. **Jenkins**



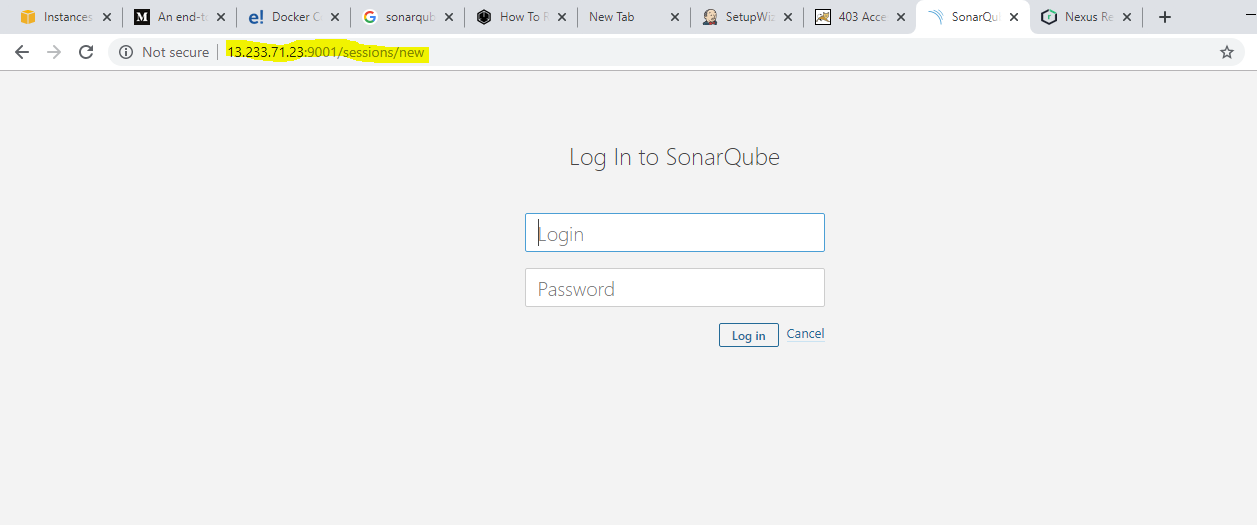




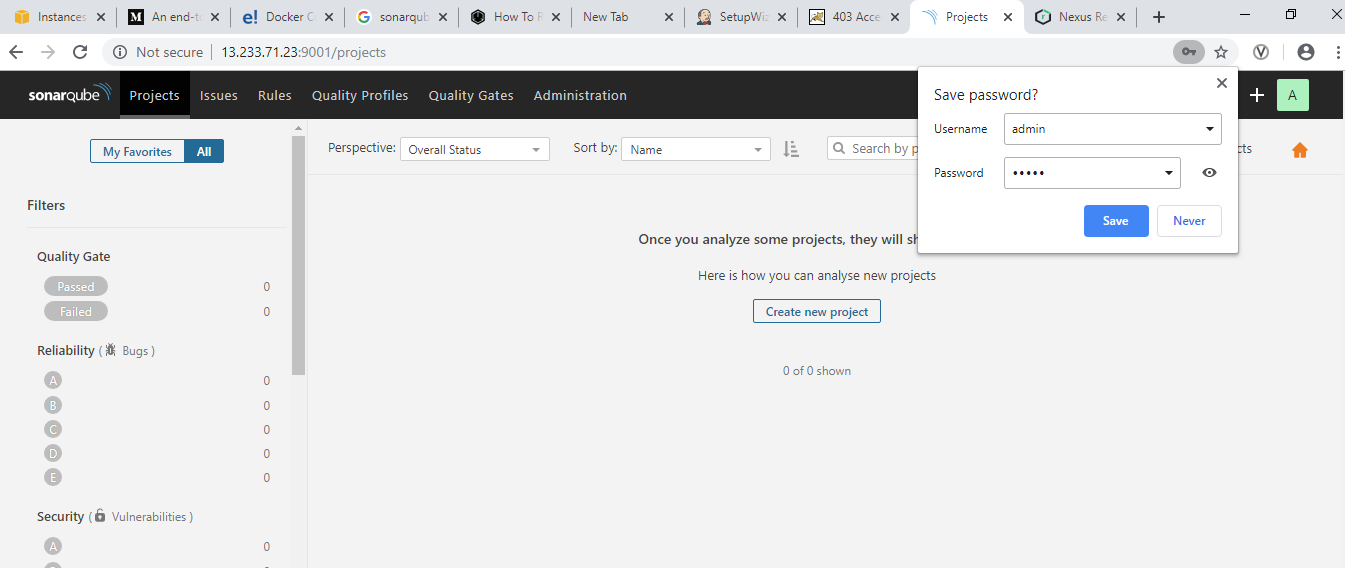




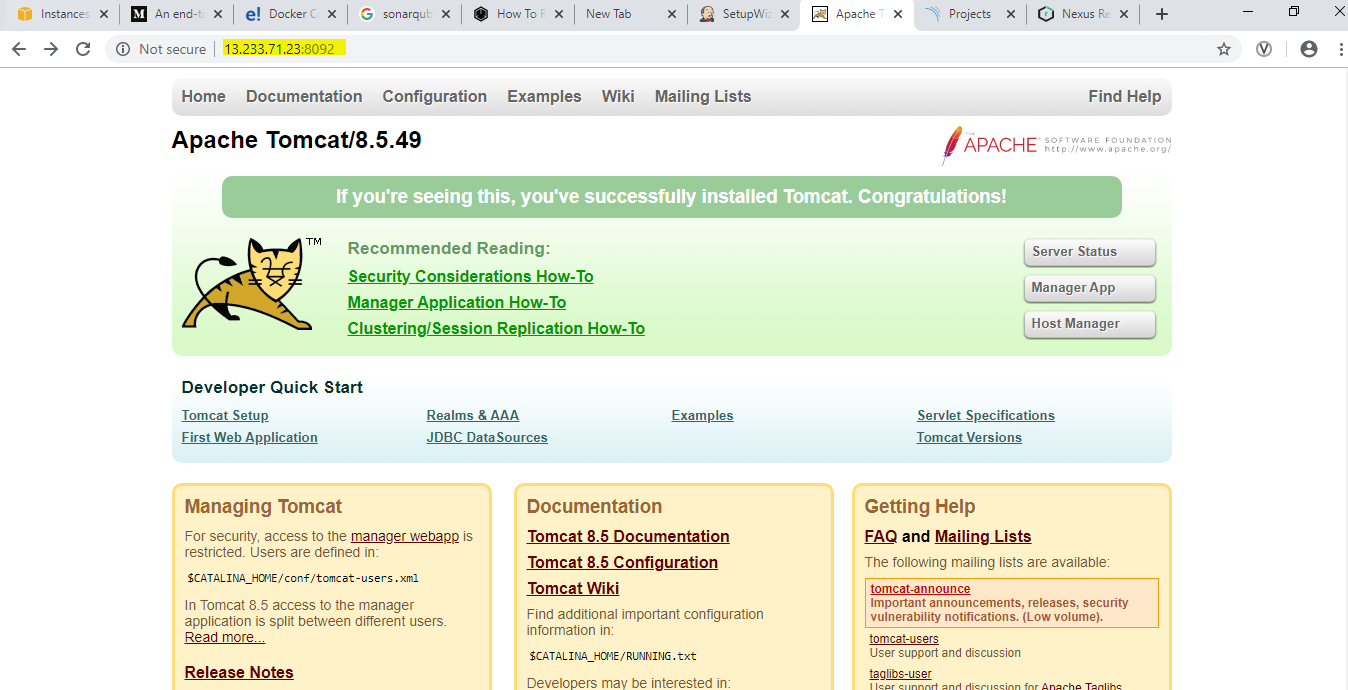
1. Sonarqube



**Dashboard**

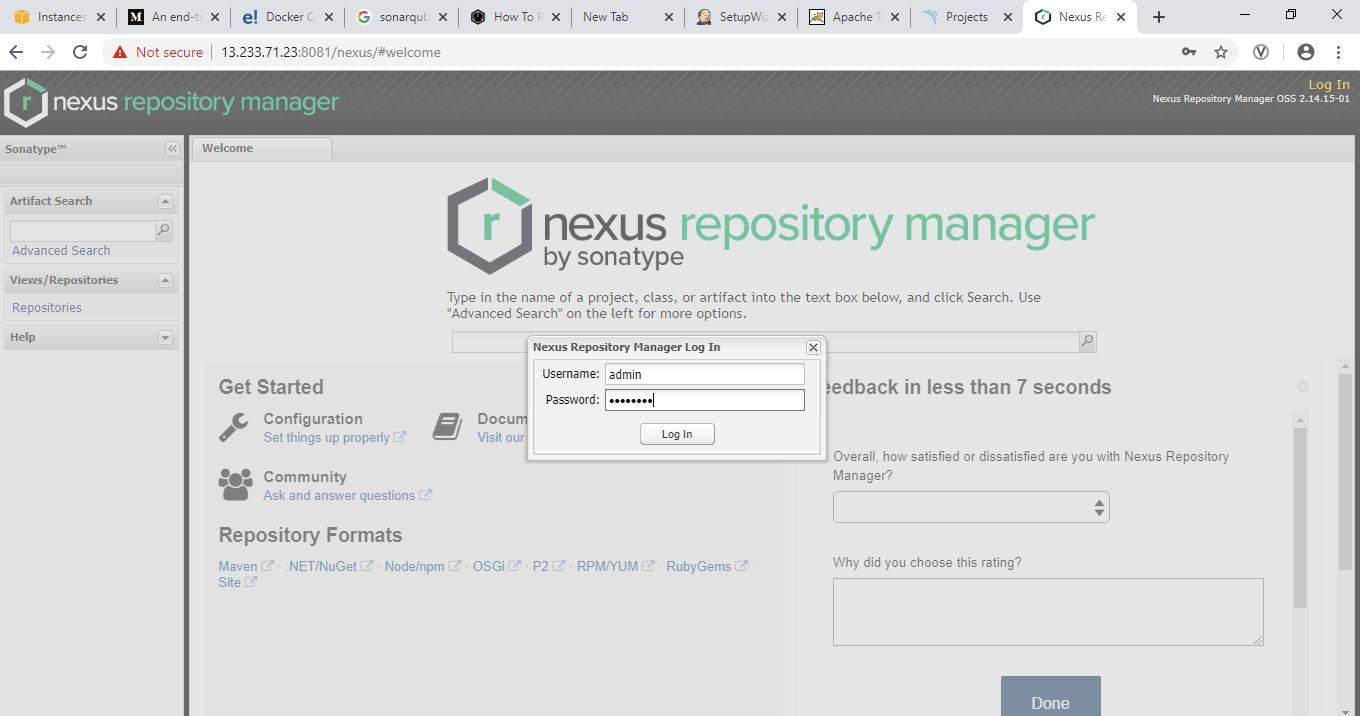


1. **Tomcat**



**NOTE: here are able to see the tomcat dashboard but for login we need to change context.xml file, uers.xml and server.xml as well, then only will be able to see the manager-app**

1. **Nexus Dashboard**



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Configure the tomcat files context.xml users.xml via docker file**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**🡪create a directory mkdir mydockerbuild**

**🡪 cd mydockerbuild/**

**🡪 vi Dockerfile**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**FROM tomcat**

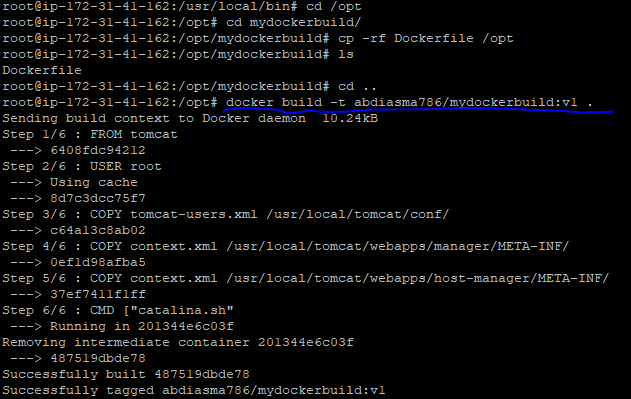
**USER root**

**COPY tomcat-users.xml /usr/local/tomcat/conf/**

**COPY context.xml /usr/local/tomcat/webapps/host-manager/META-INF/**

**CMD ["catalina.sh","run"]**

* **cp -rf Dockerfile /opt**
* **cd ..**
* **docker build -t abdiasma786/mydockerbuild:v1 .**



* **Now login on dockerhub and push the docker file to yours repository—**

**🡪root@ip-172-31-41-162:/opt# docker login**

**Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.**

**Username: abdiasma786**

**Password:**

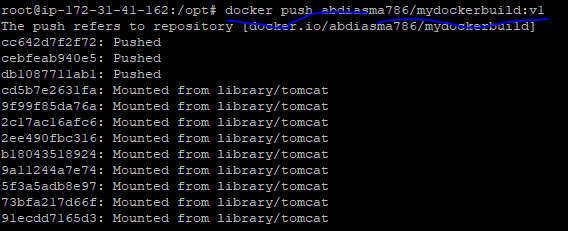
**WARNING! Your password will be stored unencrypted in /root/.docker/config.json.**

**Configure a credential helper to remove this warning. See**

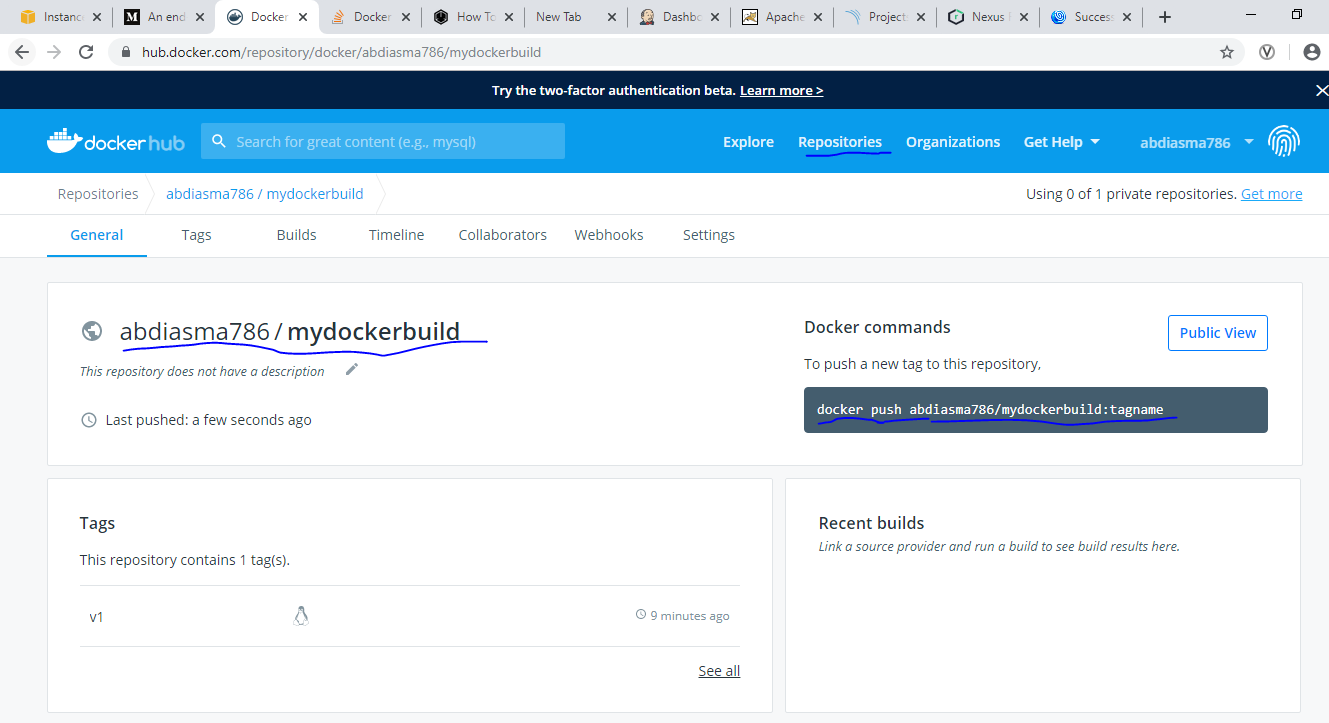
**https://docs.docker.com/engine/reference/commandline/login/#credentials-store**

**Login Succeeded**

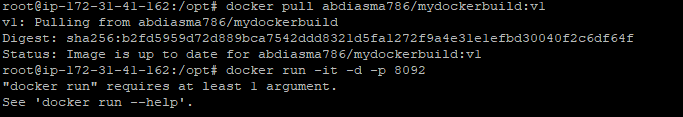
* **root@ip-172-31-41-162:/opt# docker push abdiasma786/mydockerbuild:v1**



* **go to your docker hub account🡪repository🡪”mydockerbuild”🡪**



* **Now pull the same image from docker hub-**
* **root@ip-172-31-41-162:/opt# docker pull abdiasma786/mydockerbuild:v1**



* **Now Run the custom image –**

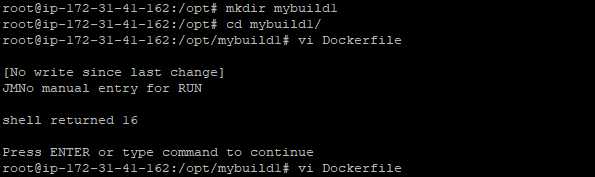
**🡪root@ip-172-31-41-162:/opt# docker run -it -d -p 8080:8080 --name mytomcat2 abdiasma786/mydockerbuild:v1 bash**

* **Check for the tomcat page is up/not**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Create a docker file for custom jenkins image to include maven/node js installation and avoid manual configuration and push this image to docker hub**

* **root@ip-172-31-41-162:/opt# mkdir mybuild1**
* **root@ip-172-31-41-162:/opt# cd mybuild1/**
* **root@ip-172-31-41-162:/opt/mybuild1# vi Dockerfile**



**Docker file**

**FROM jenkins/jenkins:lts**

**USER root**

**MAINTAINER asma**

**RUN apt-get update**

**RUN curl -sL https://deb.nodesource.com//setup\_10.x | bash**

**RUN apt-get install -y nodejs**

**RUN mkdir /opt/maven/**

**RUN wget http://mirrors.estointernet.in/apache/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz**

**RUN tar -zxvf apache-maven-3.6.2-bin.tar.gz -C /opt/maven --strip-components=1**

**EXPOSE 8080**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* **root@ip-172-31-41-162:/opt/mybuild1# cp -rf Dockerfile /opt**
* **root@ip-172-31-41-162:/opt/mybuild1# cd ..**
* **Build --**

**root@ip-172-31-41-162:/opt# docker build -t abdiasma786/mybuild1:v1 .**

**Sending build context to Docker daemon 11.78kB**

**Step 1/10 : FROM jenkins/jenkins:lts**

**---> 22b8b9a84dbe**

**Step 2/10 : USER root**

**---> Running in dfb99bb03a18**

**Removing intermediate container dfb99bb03a18**

**---> 818883f98d46**

**Step 3/10 : MAINTAINER asma**

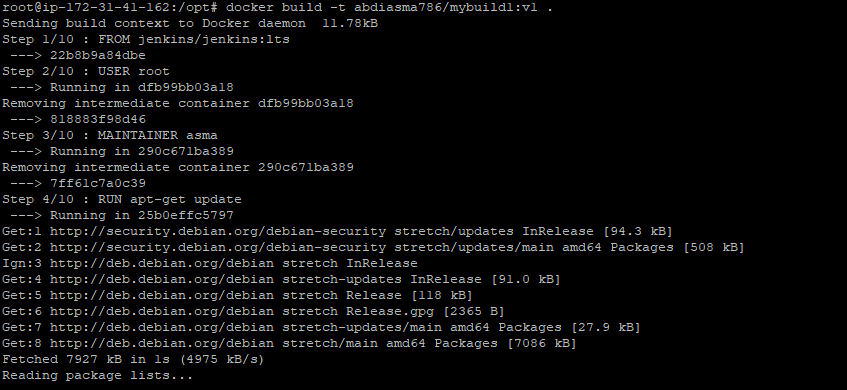
**---> Running in 290c671ba389**

**Removing intermediate container 290c671ba389**

**---> 7ff61c7a0c39**

**Step 4/10 : RUN apt-get update**

**---> Running in 25b0effc5797**



* **Login to docker hub**

**root@ip-172-31-41-162:/opt# docker login**

**Authenticating with existing credentials...**

**WARNING! Your password will be stored unencrypted in /root/.docker/config.json.**

**Configure a credential helper to remove this warning. See**

**https://docs.docker.com/engine/reference/commandline/login/#credentials-store**

**Login Succeeded**

* **Push the image to docker hub**

**root@ip-172-31-41-162:/opt# docker push abdiasma786/mybuild1:v1**

**The push refers to repository [docker.io/abdiasma786/mybuild1]**

**2b01e9e7ae09: Pushed**

**074913e21d5b: Pushed**

**d8714ae2c365: Pushed**

**fc5770127822: Pushed**

**094329e9f322: Pushed**

**4c515ae1172c: Pushed**

**e0485b038afa: Mounted from jenkins/jenkins**

**2950fdd45d03: Mounted from jenkins/jenkins**

**cfc53f61da25: Mounted from jenkins/jenkins**

**29c489ae7aae: Mounted from jenkins/jenkins**

**473b7de94ea9: Mounted from jenkins/jenkins**

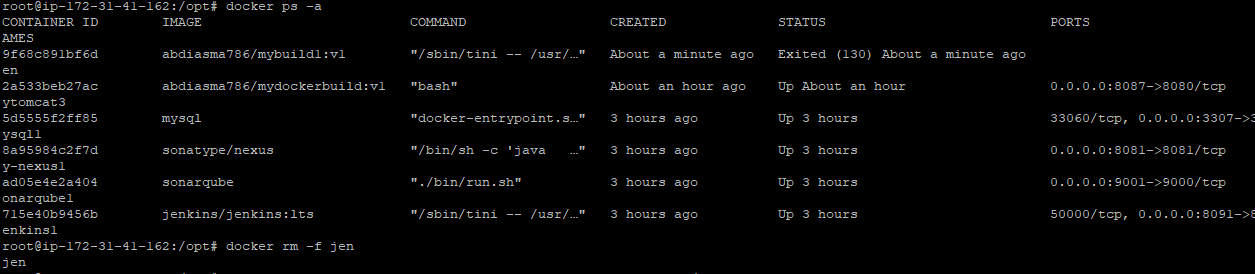
**6ce697717948: Mounted from jenkins/jenkins**

**0fb3a3c5199f: Mounted from jenkins/jenkins**

**23257f20fce5: Mounted from jenkins/jenkins**

**b48320151ebb: Mounted from jenkins/Jenkins**

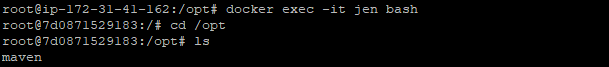
* **Docker ps –a**



* **root@ip-172-31-41-162:/opt# docker run -it -d -p 8090:8080 --name jen abdiasma786/mybuild1:v1** 

**Chexk for maven inside Jenkins congtainer-**

* **root@ip-172-31-41-162:/opt# docker exec -it jen bash**
* **root@7d0871529183:/# cd /opt**
* **root@7d0871529183:/opt# ls**



* **Check for maven inside Jenkins congtainer-**

