**POC- Problem statement**

**\*\*\*\*\*\*\*\*\*\*\*\*\*JENKINS CI POC \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Below are the Stages that are required for the CI pipeline**

**1. master slave architecture ( using jnlp protocol)**

**2. Make sure that the job is running on slave**

**3. Integration of github with jenkins & the commit id from github must be displayed on Jenkins & configure webhook**

**4. Generate code coverage report using cobertura plugin**

**5. Integrate sonarqube with jenkins**

**6. Create customize gate for sonarqube & if make sure if the quality gate fails the job should not trigger**

**7. build docker image**

**8. push the docker image to docker hub**

**Install JAVA ON MASTER**

sudo amazon-linux-extras install java-openjdk11 -y

sudo update-alternatives --config java

vim ~/.bash\_profile

JAVA\_HOME="/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86\_64/bin/java"

source ~/.bash\_profile

echo $JAVA\_HOME

**Install Jenkins on MASTER**

sudo wget -O /etc/yum.repos.d/jenkins.repo \

https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum upgrade -y

sudo yum install jenkins java-11-openjdk-devel -y

sudo systemctl daemon-reload

sudo service jenkins start

sudo service jenkins status

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

**sudo systemctl status Jenkins**

**sudo systemctl start jenkins.service**

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

OPTIONAL

sudo yum install httpd -y **( global config error path error maven http error 403) (ERROR1)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Enter the public ip of the master Jenkins server**

Public\_ip\_of\_master:8080

Get the initial admin password of Jenkins

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Install suggested plugins

**Create a jenkins user named master from jenkins dashboard**

**Install git and maven on master & slave**

sudo yum install git -y

sudo yum install maven -y

**Repo clone and pushed to github**

<https://github.com/akshaykamthe651/Tecton.git>

**Setup master slave**

**Slave setup on master**

Manage jenkins → Manage node & clouds → New Node → give node name linux\_node1 →

add to permanent

**Remote root directory**

/home/ec2-user/

**Tick on ( use websocket) & save**

Click on the slave node & right click on agent.jar then copy link address

**On slave machine**

Download the agent.jar file at /home/ec2-user using wget

wget http://54.198.69.61:8080/jnlpJars/agent.jar

**Join from slave using below command**

java -jar agent.jar -jnlpUrl http://54.198.69.61:8080/computer/linux\_slave1/jenkins-agent.jnlp

-secret aa224035b5f55c28c926847bcf9bf60d4cf2dcf75b53be7fb350581a0413b874 -workDir

"/home/ec2-user/"

java -jar agent.jar -jnlpUrl http://54.147.245.0:8080/computer/linux\_node1/jenkins-agent.jnlp

-secret a4d1e2ae0a49dc202811fe9b1a34accf06247077e31b64278306ba562ba1624e -workDir

"/home/ec2-user/"

Check if the node/slave is connected

**Configure global config**

Manage jenkins → Global tool configuration

**Jdk**

Add jdk

Name: java

**Untick install automatically**

**Java\_home =/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86\_64**

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86\_64

**Git**

**To find the path of git below is the command**

Whereis git

/usr/bin/git **(ERROR1 optional)**

**Maven add maven**

**Name :** maven

**Untick install automatically**

**To find the path of maven below is the command**

Mvn -v

/usr/share/maven

Name : maven

Path : /usr/share/maven

**Click Save**

**Install docker on linux master-slave machine**

sudo amazon-linux-extras install docker –y

sudo service docker start

sudo docker info

sudo chmod 777 /var/run/docker.sock

//compulsory run this command otherwise docker login get failed during image push

**SETUP SONAR ON NEW SERVER**

**Launch at least instance type t2.medium because the minimum ram required for sonar is**

**4 gb**

**Install java**

sudo amazon-linux-extras install java-openjdk11 -y

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Check java path**

sudo update-alternatives --config java

/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86\_64/bin/java

**Set environment variable**

vim ~/.bash\_profile

JAVA\_HOME=”/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86\_64/bin/java”

source ~/.bash\_profile

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Install sonarqube**

**download binary of sonar**

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.1.0.31237.zip

unzip sonarqube-8.1.0.31237.zip

cd sonarqube-8.1.0.31237/bin/linux-x86-64/

./sonar.sh console

Login using admin as username & password

http://50.17.69.175:9000/ ( public ip of sonar server : 9000 )

Navigate to admin logo which is at top right corner → click on my account

→ security → enter token name : sonaradmin → click on generate & copy token

d1ced1005c661658d60e9cfcfb8eae0002232cd0

**Go to jenkins master server**

Install sonar plugin

Managejenkins → manage plugins → available plugins → search plugin

SonarQube Scanner

& install without restart

Manage Jenkins → Configure system

SonarQube servers

**Tick**

Environment variables Enable injection of SonarQube server configuration as build environment

variables

Click Add sonarqube

Name : sonarqube

http://publicip\_of\_sonarqube:9000

http://34.203.248.145:9000

**Add credentials for sonarqube**

Server authentication token

Add → jenkins → secret text in kind → Secret -- paste the token that was copied from sonar

server → Description : sonar token →

click on none & select sonar token from drop down menu

**Tools Path configuration**

**Manage jenkins → manage nodes & clouds → click on settings of node → tools location**

**add →**

**GIT**

**To find the path of git below is the command**

Whereis git ( Execute this command on slave node)

/usr/bin/git

**JAVA**

**To find the path of git below is the command**

update alternatives command ( Execute this command on slave node)

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86\_64

**MAVEN**

**To find the path of maven below is the command**

mvn -v ( Execute this command on slave node)

/usr/share/maven

CREATE A PIPELINE PROJECT

Name for the project(Add script )

pipeline{

environment {

scannerHome = tool 'sonarscanner'

registry = "akshaykamthe651/jenkins-ci-declerative-pipeline"

registryCredential = "dockerhub"

dockerImage = ''

}

agent any

tools{

git 'git'

maven 'maven'

jdk 'java'

}

stages{

stage('checkout'){

steps{

git 'https://github.com/akshaykamthe651/Tecton.git'

echo 'inside checkout'

}

}

stage('build'){

steps{

sh 'mvn clean install -f pom.xml'

echo 'inside build'

}

}

stage('Code Quality'){

steps{

withSonarQubeEnv('sonarqube'){

sh 'mvn -f pom.xml sonar:sonar'

}

}

}

stage('Sonarqube') {

steps {

withSonarQubeEnv(installationName: 'sonarqube',credentialsId:'token-2') {

sh "mvn clean test sonar:sonar -Dsonar.login=c84ae0d0584513b7db6de2df37e0705ec6720ab8"

echo 'inside sonar '

}

timeout(time: 10, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

echo 'inside sonar environment'

}

}

}

stage("Build-package"){

steps{

sh "mvn clean install"

}

}

stage('Building docker image') {

steps {

script {

dockerImage = docker.build registry + ":$BUILD\_NUMBER"

}

}

}

stage('push docker image') {

steps {

script {

docker.withRegistry( '', registryCredential ) {

dockerImage.push()

}

}

}

}

stage('Publish Test Coverage Report') {

steps {

step([$class: 'JacocoPublisher',

execPattern: '\*\*/\*\*.exec',

classPattern: '\*\*/classes',

sourcePattern: '\*\*/src/main/java',

sourceInclusionPattern: '\*/\*.java',

])

}

}

}

}

Install plugins

1.docker

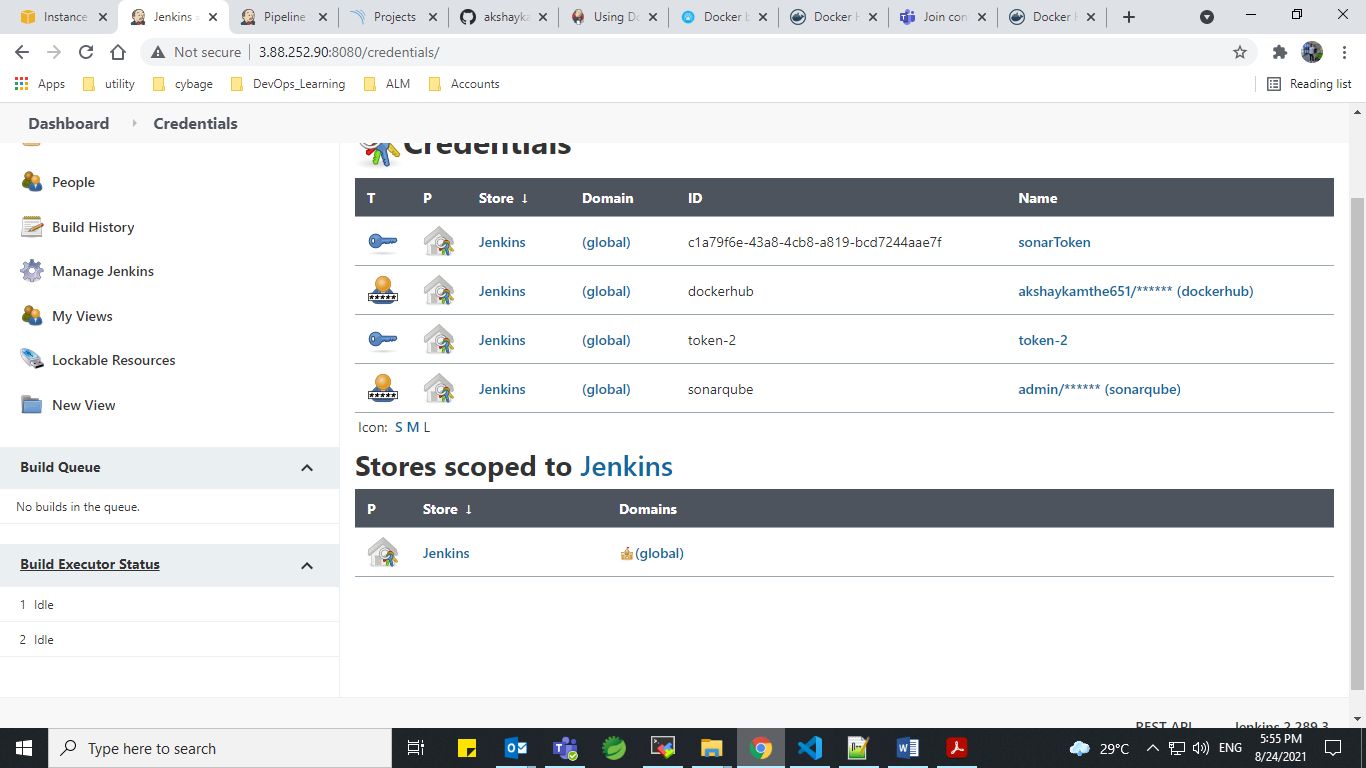
2.docker pipeline

3.cloudbees docker

4.docker build-step

5.jacoco

Add Credentials-> manage Jenkins-> manage credentials-> add credentials



Add credentials for

1.SonarToken

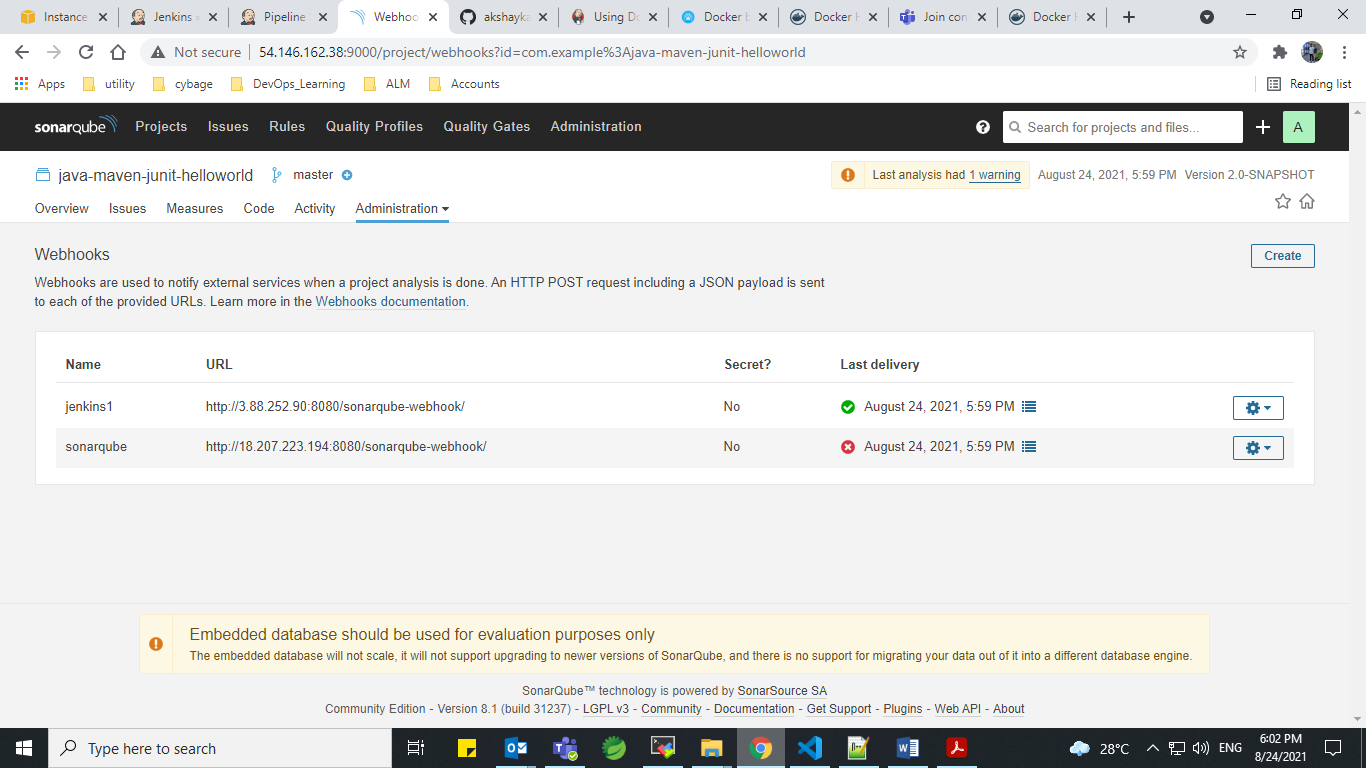
2. dockerhub

3.sonarqube

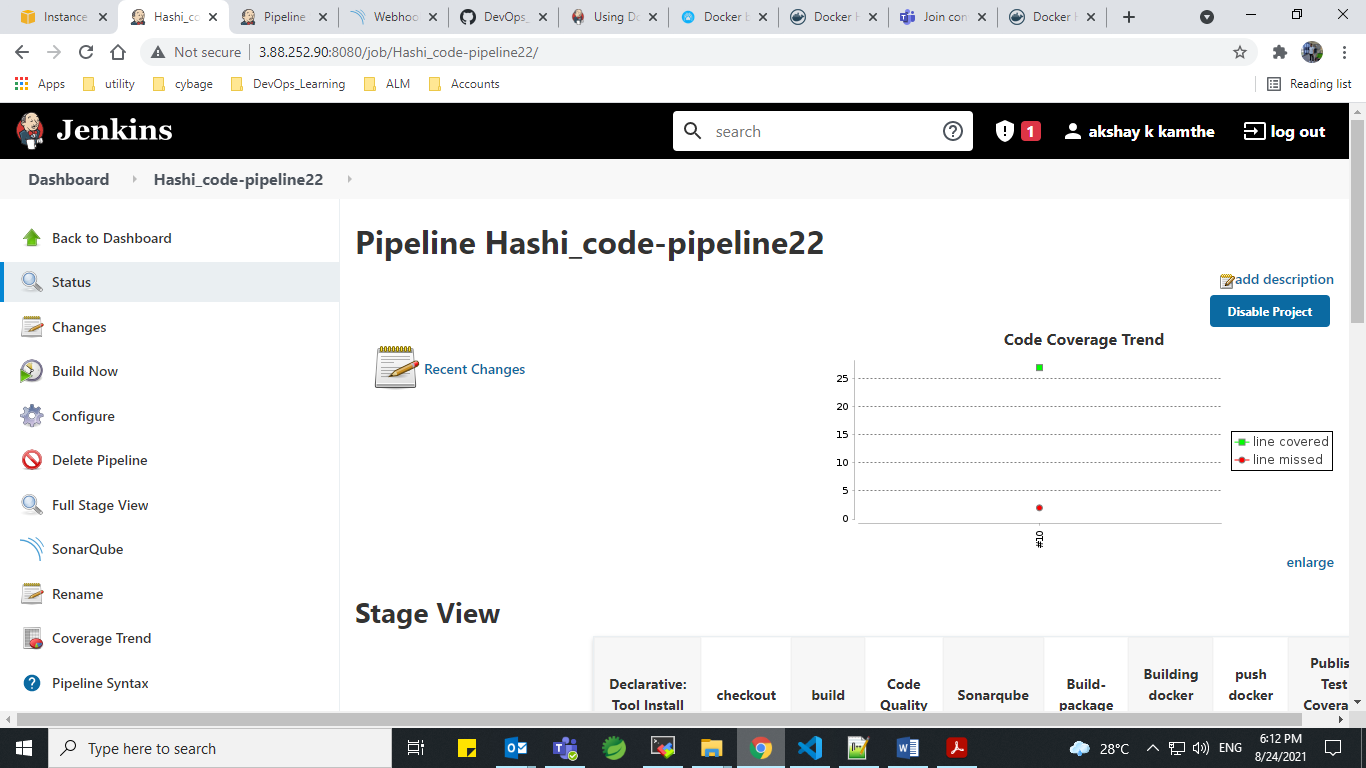
Add SonarServer webhook

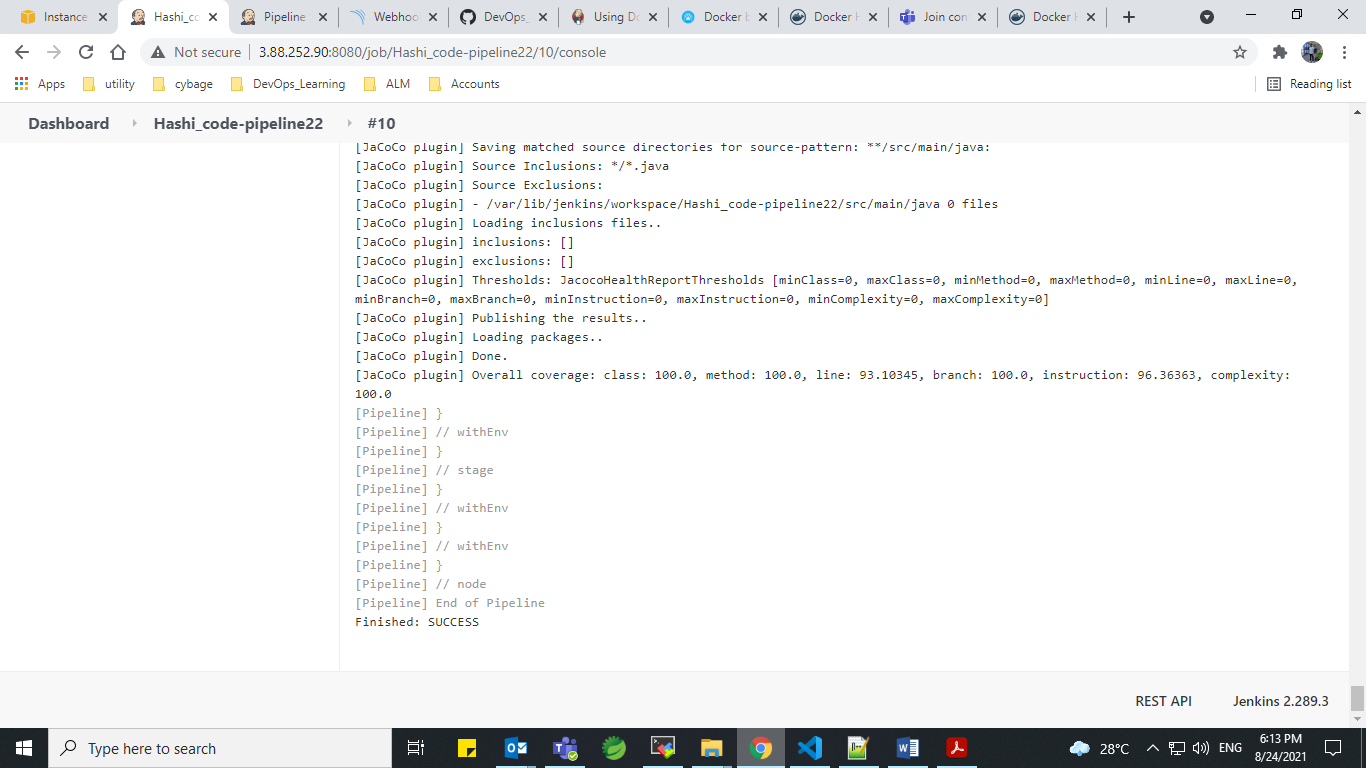
Goto sonar dashboard after first incomplete build project will get updated at dashboard goto project -> administration

-> security ->create webhook

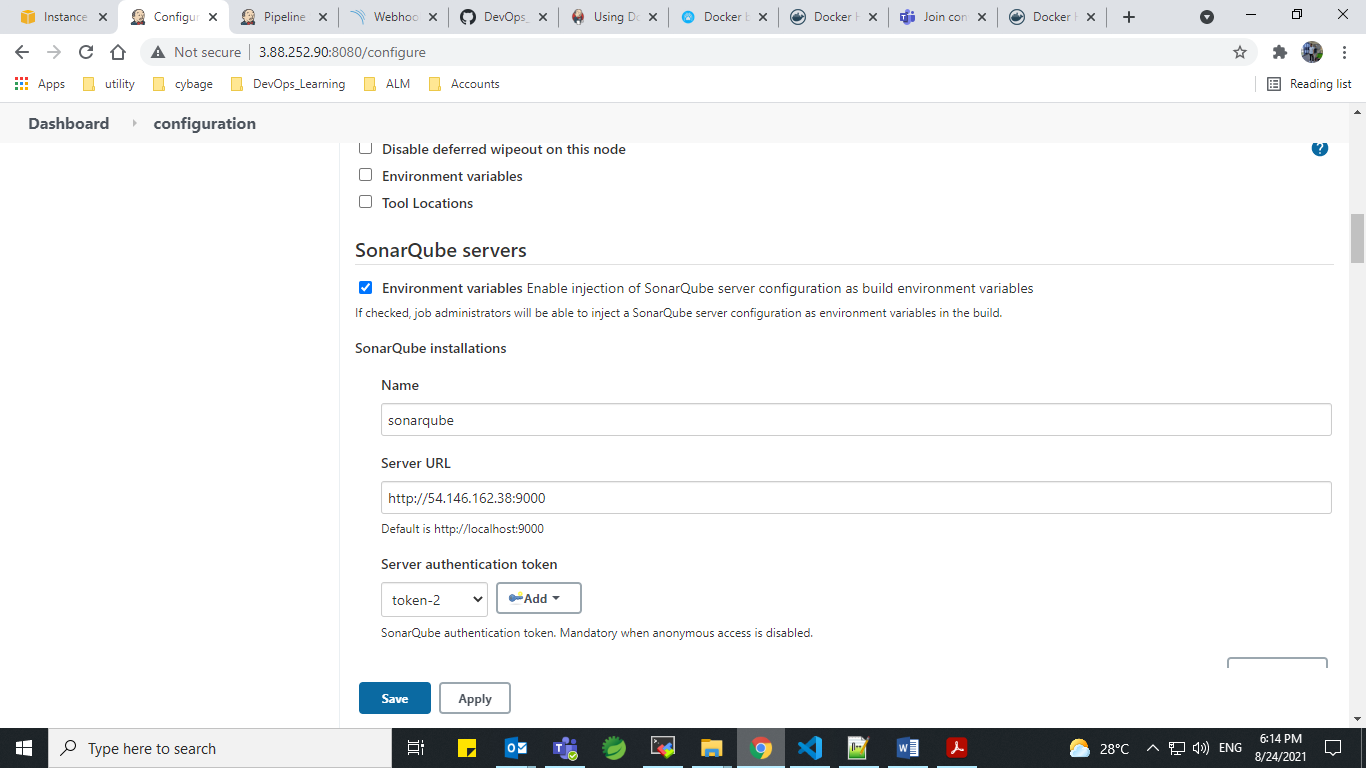


This will triggers & sends response back to Jenkins server of #success status or #Failure .





Sonarqube server (configure system)



After successful build Image get pushed to dockerhub with $BUILD\_NUMBER ex-> 9

