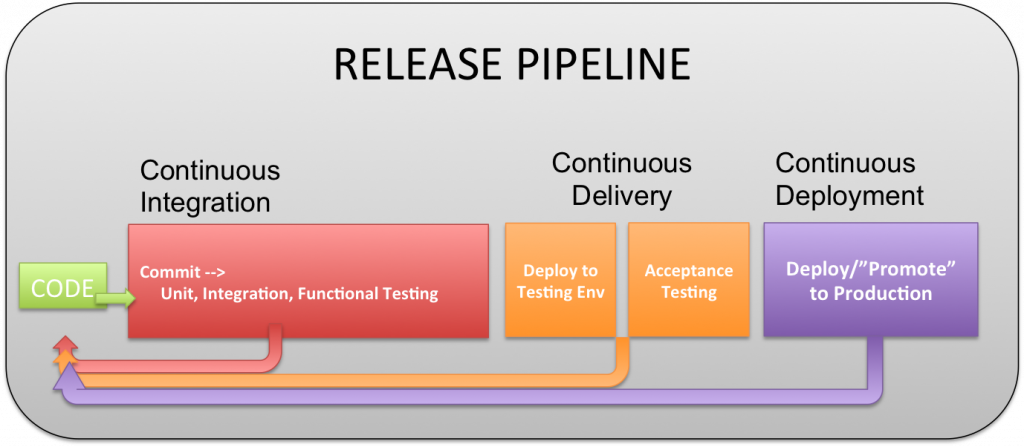
Jenkins 106

Jenkins Release pipeline



Read more about Jenkins pipeline from here: https://www.jenkins.io/doc/book/pipeline/

=> Way to create simple generic Jenkins pipeline

Step 1: Check if Jenkins Pipeline plugin is install in jenkins server ( mostly it will be installed)

Step 2: Go to create a new jenkins job, click on `new item`

Step 3: enter item name and select item type as `Pipeline` project.

Step 4: click on `Ok`

Step 5: Under `Pipeline`, enter the following script:

pipeline {

agent any

environment {

NAME = 'Ankit'

LASTNAME = 'Rajput'

}

stages {

stage('Environment Var') {

steps {

retry(3) { // Adding a retry of 3 times in case this step fails.

sh 'echo "We will use environment variables $NAME $LASTNAME"'

}

}

}

stage('Build') {

steps {

sh 'echo "Under Building stage, step 1: Building ..."'

sh '''

echo "Under Building stage, step 2: I can do more stuff here ..."

'''

}

}

stage('Test') {

steps {

retry(3) { // Adding a retry of 3 times in case this step fails.

sh 'echo "Testing ..."'

}

}

}

stage('Deploy') {

steps {

timeout(time: 3, unit: 'SECONDS') { // Adding a timeout section, if this step doesn't execute in 3 seconds, then it gets aborted.

echo 'Deploying ...'

}

}

}

}

post {

always {

echo 'I will always get executed !!'

}

success {

echo 'I will only get executed if this job is a success'

}

failure {

echo 'I will only get executed if this job is a failure'

}

unstable {

echo 'I will only get executed if this job is unstable'

}

}

}

Step 6: Save and build this jenkins pipeline.

=> Way to use sensitive password credentials in Jenkins pipeline

Step 1: Go to `manage jenkins` and click on `Credentials`.

Step 2: Then, click on `System` and then click on `Global credentials`

Step 3: finally, click on `Add Credentials`

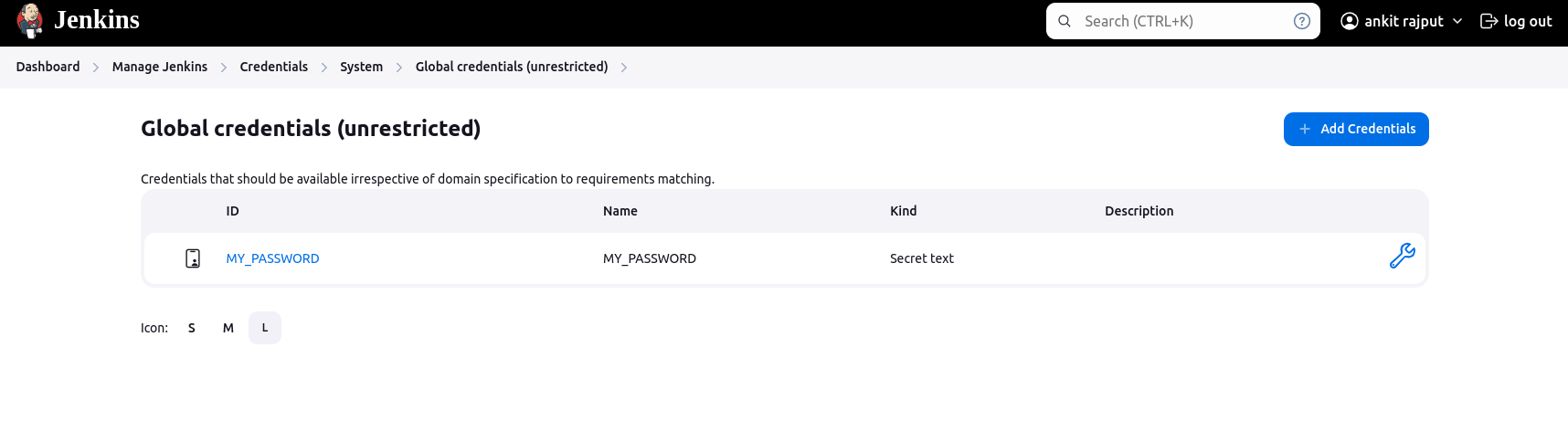
Step 4: Select `Kind` as `Secret text`

Step 5: In `Secret` textbox, enter the secret text

Step 6: In `ID` textbox, enter the ID to reference this secret text.

Step 7: Now, click on `Create`.

Screenshot:



Step 8: Follow the above steps in previous section and use this credential in jenkins pipeline like this:

pipeline {

agent any

environment {

NAME = 'Ankit'

LASTNAME = 'Rajput'

MY\_PASSWORD = credentials('MY\_PASSWORD')

}

stages {

stage('Environment Var') {

steps {

retry(3) { // Adding a retry of 3 times in case if this step fails.

sh 'echo "We will use environment variables $NAME $LASTNAME and my passcode is $MY\_PASSWORD"'

}

}

}

stage('Build') {

steps {

sh 'echo "Under Building stage, step 1: Building ..."'

sh '''

echo "Under Building stage, step 2: I can do more stuff here ..."

'''

}

post {

success {

sh 'echo "this means build stage is successful"'

}

}

}

stage('Test') {

steps {

retry(3) { // Adding a retry of 3 times in case if this step fails.

sh 'echo "Testing ..."'

}

}

}

stage('Deploy') {

steps {

timeout(time: 3, unit: 'SECONDS') { // Adding a timeout section, if this step doesn't execute in 3 seconds, then it gets aborted.

echo 'Deploying ...'

}

}

}

}

post {

always {

echo 'I will always get executed !!'

}

success {

echo 'I will only get executed if this job is a success'

}

failure {

echo 'I will only get executed if this job is a failure'

}

unstable {

echo 'I will only get executed if this job is unstable'

}

}

}

=> Way to integrate the `Blue Ocean` UI in jenkins

Step 1: Go to `manage jenkins` and then click on `Plugins`

Step 2: Then under `Available plugins`, search for `Blue Ocean` and install this plugin.

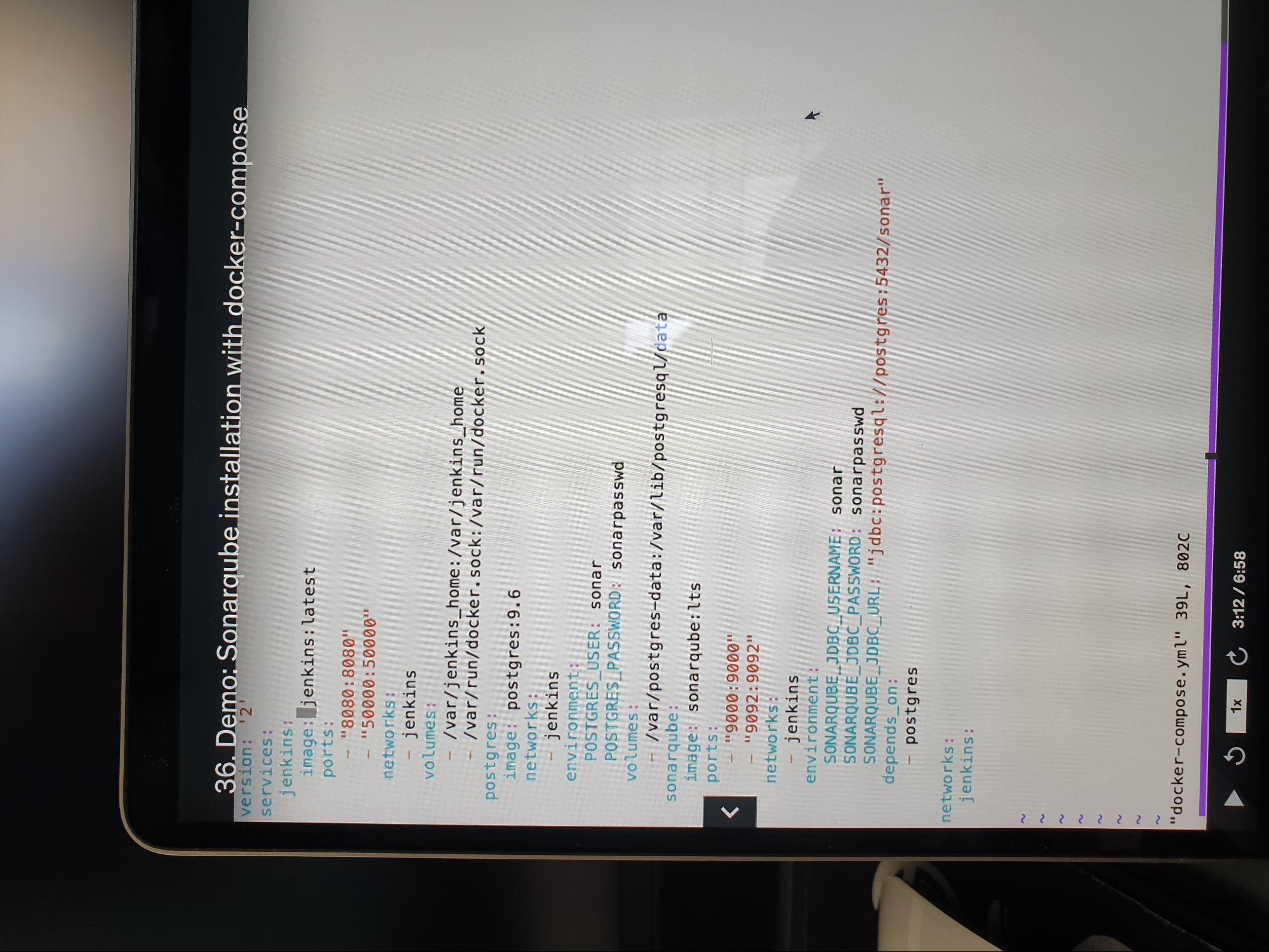
Step 3: Awesome !! now use the blue ocean UI in jenkins.

=> Things to learn further

1. Integrate Sonarqube in the jenkins pipeline

Supporting material

1. Way to install sonarqube via docker



1. Then follow this link: <https://callmezydd.medium.com/unlocking-code-quality-integrating-jenkins-pipeline-with-sonarqube-and-github-7f450f1c90ab>
2. Reference: https://github.com/wardviaene/jenkins-course/blob/master/sonarqube/Jenkinsfile

=> Project

Create a Jenkins pipeline with following things:

1. Create a Gitlab server in docker and push the code in it.
2. That should automatically, trigger a jenkins build in jenkins server hosted by docker
3. Then it should pull the code, build and run test cases.
4. Then the next stage should be `Sonarqube`, where test case results should be published.
5. Then next step, should be pushing the artifact in JFrog artifactory hosted via docker.
6. Finally, last should be creating an application docker image and pushing to dockerHub.

So, anyone can just pull the application docker image and start to use the application.