**============ Harbor Setup ===========**

sudo su

apt update && sudo apt -y full-upgrade

apt install apt-transport-https ca-certificates curl software-properties-common

curl -s https://api.github.com/repos/goharbor/harbor/releases/latest | grep browser\_download\_url | cut -d '"' -f 4 | grep '\.tgz$' | wget -i -

OR

wget https://github.com/goharbor/harbor/releases/download/v2.10.0/harbor-online-installer-v2.10.0.tgz

tar -xvf harbor-online-installer-v2.10.0.tgz

cd harbor

ls -al

cd ..

**Generating SSL Certificate & Key**

To generate, you need OpenSSL installed. You can generate by any other way also, if you're clueless, just follow along.

Navigate to the directory you want the credentials to be generated and execute the following command. Replace registry.technonext.com or 127.0.0.1 with your hostname.

mkdir ssl\_cert

cd ssl\_cert

openssl req -x509 \

-sha256 -days 356 \

-nodes \

-newkey rsa:2048 \

-subj "/CN=registry.technonext.com/C=BD/L=Dhaka" \

-keyout rootCA.key -out rootCA.crt

cd ..

nano harbor.yml.tmpl

1. Change these parameters

2. - "hostname" (IP or Domain)

3. - "harbor\_admin\_password"

4. certificate: /Your\_Location/ssl\_cert/rootCA.crt

5. private\_key: /Your\_Location/ssl\_cert/rootCA.key

Save as this filename (drop the .tmpl)

harbor.yml

Verify configurations

cat harbor.yml

**Install Docker and Docker Compose before execute the install.sh file**

**Install Harbor**

./install.sh

docker image ls

docker container ls

Test Harbor web access

http://IP\_ADDRESS or Hostname

OR

https://IP\_ADDRESS or Hostname

**The below lines should be added to skip the http/https certification authentication error.**

\*\*\*on the remote machine as we don’t have a DNS server to resolve

root@herbour-documentation-testing2:/home/mehedi# vim /etc/docker/daemon.json

{

"insecure-registries" : ["192.168.169.241"]

}

Now Pull an Image with command # Docker Pull nginx

Before push this image to the local repo have to add an tag with it

# Docker image tag naginx 192.168.169.241/harbor/nginx

**Note: We have to create a project first on Harbor server. (Here harbor is the project name)**

Now we can push our image with the command

# Docker push 192.168.169.241/harbor/nginx

**Vulnerability Scanning With Harbor**

Harbor provides static analysis of vulnerabilities in images through the open source projects [Trivy](https://github.com/aquasecurity/trivy) and [Clair](https://github.com/coreos/clair).

To use Trivy or Clair or both, you must enable Trivy, Clair, or both when you install your Harbor instance (by appending installation options --with-trivy, --with-clair, or both).

./install.sh –with-trivy

docker image ls

docker container ls

To use Harbor as central registry through Jenkins to store docker image on central registry We need harbor Jenkins and remote server to do this.

To do CI/CD from Jenkins we need to add credentials for all the servers,

Credentials needed for Harbor registry here **User: Harbor** and **Password: Harbor12345**

We build image on the Jenkins server and we need to add credentials for ssh-key based authentication to Jenkins to access remote-server and harbor registry from Jenkins server.

**Here is the pipeline script given below:**

pipeline {

agent any

environment {

image= "192.168.169.241/test/newalpine:$BUILD\_NUMBER"

SSH\_CREDENTIALS = 'remote-server'

Server\_ip = '192.168.169.242'

DOCKER\_REGISTRY = '192.168.169.241'

DOCKER\_CREDENTIALS\_ID = 'harbor-regis'

}

stages {

stage('docker build') {

steps {

sh "docker images"

sh "cd /home/mehedi/docker && docker build -t ${image} ."

}

}

stage('Harbor Logging for Build') {

steps {

withCredentials([usernamePassword(credentialsId: DOCKER\_CREDENTIALS\_ID, usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]){

sh "docker login -u ${DOCKER\_USERNAME} -p ${DOCKER\_PASSWORD} ${DOCKER\_REGISTRY}"

}

}

}

stage('Harbor push') {

steps {

withCredentials([usernamePassword(credentialsId: DOCKER\_CREDENTIALS\_ID, usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]){

sh "cd /home/mehedi/docker/ && docker tag newalpine ${DOCKER\_REGISTRY}/test/${image}"

sh "cd /home/mehedi/docker/ && docker push ${image}"

}

}

}

stage('Harbor Logging for Deploy') {

steps {

withCredentials([usernamePassword(credentialsId: DOCKER\_CREDENTIALS\_ID, usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]){

sshagent(credentials: [SSH\_CREDENTIALS]) {

sh "ssh -o StrictHostKeyChecking=no mehedi@${Server\_ip} 'docker login -u ${DOCKER\_USERNAME} -p ${DOCKER\_PASSWORD} ${DOCKER\_REGISTRY}'"

}

}

}

}

stage('Deploy to Remote Server') {

steps {

script {

sshagent(credentials: [SSH\_CREDENTIALS]) {

sh "ssh -o StrictHostKeyChecking=no mehedi@${Server\_ip} 'cd /home/mehedi/docker/ && docker rmi -f ${DOCKER\_REGISTRY}/test/${image} && docker rm -f newalpine'"

sh "ssh -o StrictHostKeyChecking=no mehedi@${Server\_ip} 'cd /home/mehedi/docker && docker pull ${image}'"

sh "ssh -o StrictHostKeyChecking=no mehedi@${Server\_ip} 'docker run -d -p 8000:8000 --name newalpine --restart always ${image} '"

}

}

}

}

}

}