DevOps (Lab 1 and 2 Commands)

VM address: http://172.17.5.39:

doyendev.click

INSTALLATION REQUIRED:

Docker, node, net-tools, git

SETUP SSH:

- ssh appadmin@172.17.5.39
- P@ssw0rd

CREATE A NEW USER: (so that it may not conflict with other team members)

- sudo useradd -s /bin/bash -d /home/yourName / -m -G sudo yourName
- sudo passwd yourName

INSTALL NODE: (If not install; for verification type node --version)

- curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
- export NVM_DIR="$HOME/.nvm" && [-s "$NVM_DIR/nvm.sh"] && \. "$NVM_DIR/nvm.sh" && [-s "$NVM_DIR/bash_completion"] && \. "$NVM_DIR/bash_completion"
  OR
- source ~/.bashrc
- nvm install 18
- nvm use 18

INSTALL DOCKER:

- RUN apt update && curl -fsSL https://get.docker.com | sh
- sudo usermod -aG docker $USER(yourName)
- newgrp docker (no logout)
  PASSWORDLESS LOGIN:
- [PASTE your .pub key here]
- nano /home/$USER(yourName)/.ssh/authorized_keys
  (IF .SSH FOLDER DOES NOT EXIST)
- mkdir .ssh
- nano /home/$USER/.ssh/authorized_keys
- paste the public key here, so next time you can login without password
- or (known_hosts)

INSTALLING JENKINS:

- anktest@labserver:/home/anktest$ mkdir jenkins
- anktest@labserver:/home/anktest/jenkins$ nano Dockerfile
  Paste jenkins's Dockerfile data to pull image of Jenkins from docker repo)

```
#Dockerfile
FROM jenkins/jenkins:lts
USER root
RUN mkdir -p /tmp/download && \
 curl -L https://download.docker.com/linux/static/stable/x86_64/docker-18.03.1-ce.tgz | tar -xz -C
/tmp/download && \
 rm -rf /tmp/download/docker/dockerd && \
 mv /tmp/download/docker/docker* /usr/local/bin/ && \
 rm -rf /tmp/download && \
 groupadd -g 999 docker && \
 usermod -aG staff,docker Jenkins

USER jenkins
```

BUILD JENKINS IMAGE AND RUN IT IN CONTAINER:

- anktest@labserver:/home/anktest/jenkins$ sudo docker build -t jenkins-lab-anktest .
  - at the time of building the pulled image of Jenkins just give a unique tag/name to image e.g. jenkins-lab-anktest
  - now as you build with image tag name jenkins-lab-anktest, so run the image with this tag name

- anktest@labserver:/home/anktest/jenkins$ docker run -dit --restart=always -P -v jenkins_home:/var/jenkins_home_anktest -v /var/run/docker.sock:/var/run/docker.sock jenkins-lab-anktest
  - -P = it will assign port by itself which you can see by doing docker ps
  - here jenkins-lab-anktest is the image tag name which you have selected while executing build command
  - To avoid Jenkins pswd override to your group member make "jenkins_home_anktest" unique as well.

- anktest@labserver:/home/anktest/jenkins$ docker ps
  - from above command get the port on which your Jenkins in running
  - go to browser type your server address e.g. http://172.17.5.3x:portnumber
  - After, it will ask for password to verify Jenkins have been setup properly.
  - exec below command

- anktest@labserver:/home/anktest/jenkins$ sudo docker exec -it yourContainerId bash
  - here yourContainerId is the container id on which Jenkins running e.g. 443f5957500a (see it with docker ps command)

    - it will navigate to the bash terminal and here enter command below

- jenkins@443f5957500a:/$ cat /var/jenkins_home/secrets/initialAdminPassword

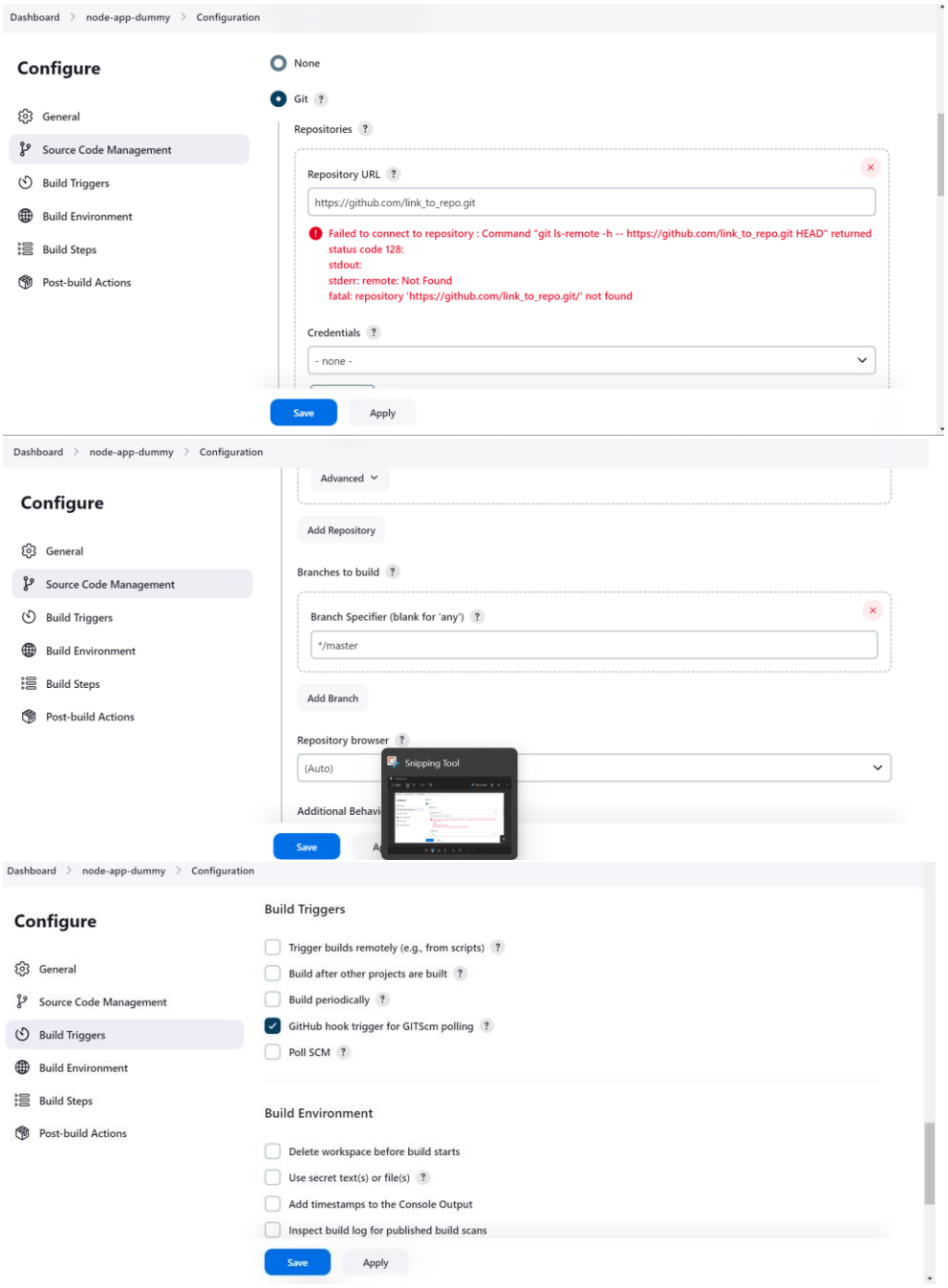
    - this will return a password, copy it and paste it to Jenkins password required field.

CREATING CI/CD PIPELINE:

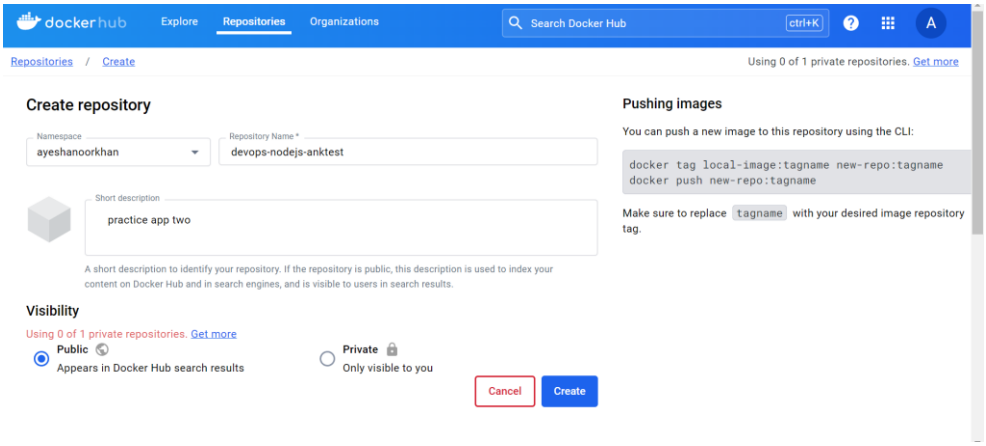
- Go to Jenkins, navigate to Dashboard>Manage Extensions>Available Plugins
- Search for CloudBees Docker Build and Publish Plugins, install it.
- Now we need a dummy project to create its CI/CD Pipeline
- Clone a dummy repo "https://github.com/Khhafeez47/node-js-sample.git"
- git clone https://github.com/Khhafeez47/node-js-sample.git
- create a repo on github
- login with your credentials in terminal.
    - git config --global user.name yourGitHubUsername
    - git config --global user.email yourGitHubEmail
    - set the password in credential manager>windows credentials of github.com to PAT (which can be generated from github one time)(not sure whether works in xshell or not)
- After you clone the dummy project remove the git init folder from it and reintinalized with:
    - git init
    - git add README.md
    - git commit -m "first commit"
    - git branch master
    - git remote add origin https://github.com/link_to_repo.git or
    - git remote set-url origin https://github.com/link_to_repo.git
    - git push -u origin master
- Now you have got the dummy project locally with your read, write access. Also you have successfully pushed to that newly created github repository.
- Not sure about this step cd /var/run

JENKINS DASHBOARD:

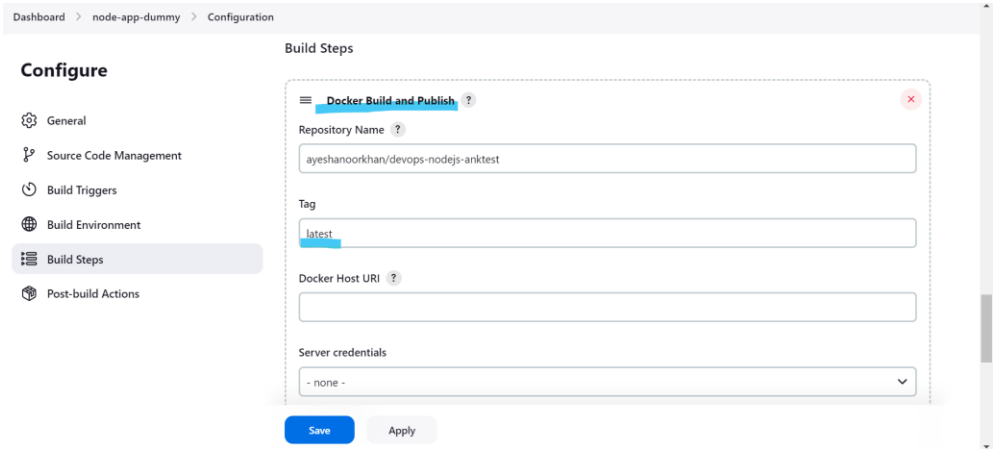
- Create a New Item, give it a name and select "Free Style Project".
- Now follow the below steps:



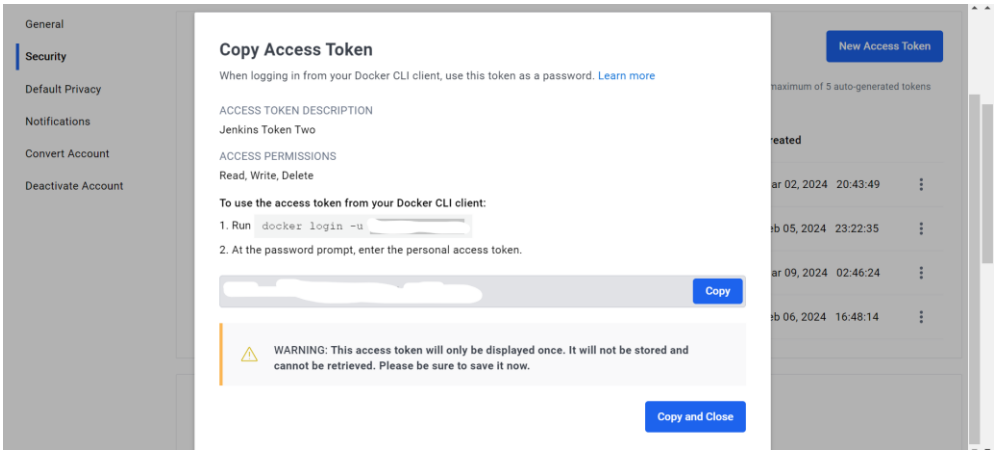
- Now create a repository for your current dummy project on Docker Account as well:
- Login/Signup to your Docker Account, click on “create new repo” and click “create”.



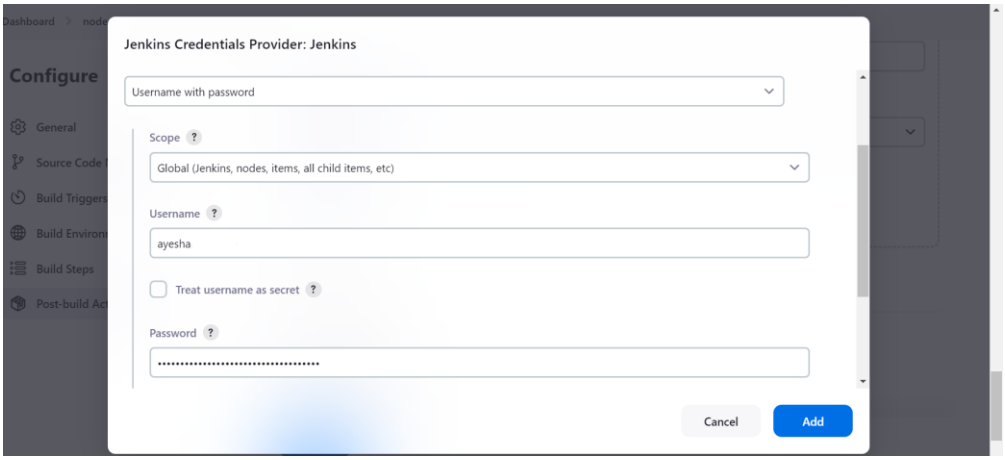
- In build step, select a new build step i.e. “Docker Build and Publish” and provide the details as below:
    - Add the name of newly created repo on docker in repo name field.
    - Tag should “latest”.
    - Registry Credential should have the credential of your Docker Hub account.
    - For Docker Hub Credential, go to My Account in Docker Hub, select security, select new access token and get the token and username from there.



Generating token from docker hub



Providing docker hub token to Jenkins login

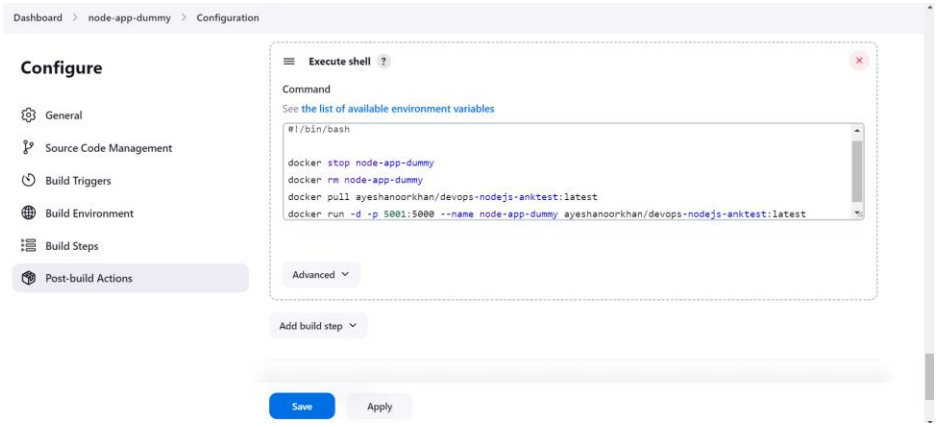


- Click again on “Add build step” and select “Execute Shell” provide the below detail on it.

#!/bin/bash

docker stop node-app-dummy
docker rm node-app-dummy
docker pull ayeshanoorkhan/devops-nodejs-anktest:latest
docker run -d -p 5001:5000 --name node-app-dummy ayeshanoorkhan/devops-nodejs-anktest:latest

(remember to replace docker repo name and jenkin item name with your unique selected names :D)



- Click on save.
- On Left Sidebar, select Build Now, to test the build
- REMEMBER:
  - Chmod 777 (Read Write Edit) error may arise
  - If error occurs “port is occupied” then let it decide port by itself and remove specifying port from Execute Shell command, means : instead of -p 5001:5000 do -P
  - To check on which port the build is running? Just do the command “docker ps” and find your containerized app deployment port there

ADDING DNS TO JENKINS AND DUMMY NODEAPP PROJECT

- anktest@labserver:/home$ sudo apt install nginx-full -y
- To get Free SSL: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04
- anktest@labserver:/home$ systemctl status nginx (To check whether nignx is running or not)
- anktest@labserver:/home$ cd /etc/nginx
- anktest@labserver:/etc/nginx $ cd sites-enabled
- anktest@labserver:/etc/nginx/sites-available$
  The file which you want to enable are in sites-enabled
- anktest@labserver:/etc/nginx/sites-available$ sudo nano jenkins

  Paste the command below

```
server {
listen 80;
server_name ayesha.doyendev.online;

location / {
proxy_pass http://localhost:32779; # Change to your backend server address
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}
}
```

  Above server_name is your sub-domain on Route 53 and localhost:32779 is the port on which container of Jenkins is running, remember to change your sub-domain and port on which app container is running

- anktest@labserver:/etc/nginx/sites-available$ cd ../sites-enabled
- anktest@labserver:/etc/nginx/sites-enabled$ sudo ln -s /etc/nginx/sites-available/nodeapp /etc/nginx/sites-enabled/

  above command (for activating the site)

- anktest@labserver:/etc/nginx/sites-enabled$ sudo nginx -t (to make sure for correct syntax)
- anktest@labserver:/etc/nginx/sites-enabled$ systemctl restart nginx

- anktest@labserver:/etc/nginx/sites-enabled$ systemctl reload nginx
- anktest@labserver:/etc/nginx/sites-enabled$ systemctl status nginx

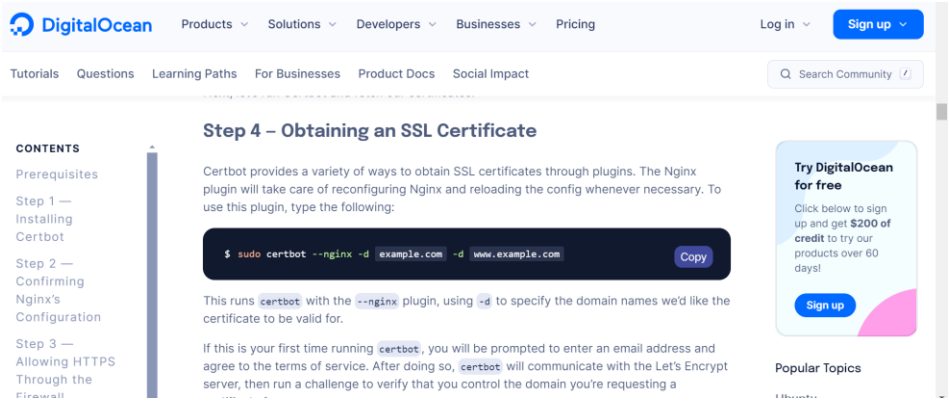
  Repeat the above steps for nodeapp (“Hello World App”) as well but create a new sub-domain for it on route 53 and correct port of its container in sudo nano nodeapp step

  http://ayesha.doyendev.online/ check jenkins url
  http://nodeapp.doyendev.online/ checkunder nodeapp url

ADDING SSL LAYER TO JENKINS AND DUMMY NODEAPP

Since we deployed/ or given the DNS/ domain name to our repos. Now we want to make it from http (not secure) to https (ssl certified)

For that, we are using Lets Encrypt:



- anktest@labserver:/home$ sudo apt install certbot python3-certbot-nginx
- Obtaining ssl certificate for sub domains
- anktest@labserver:/home$ sudo certbot --nginx -d ayesha.doyendev.online
  (Without https url daley to wo https pr hi redirect hojae select 2 for it)
  SSL failed as we have Private IP and let’s encrypt is for Public Ips