



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

Devops Jenkins Assignment

Submitted to:

Dr.Uma S

Submitted by:

Advay Aggarwal (18BCS002)

Perumalla Tushar(18BCS065)

Meghana Hadimani(18BCS052)

Samana B S(18BCS088)

Rahul Pryadarshini(18BCS074)

Setting up CI/CD Jenkins pipeline for kubernetes

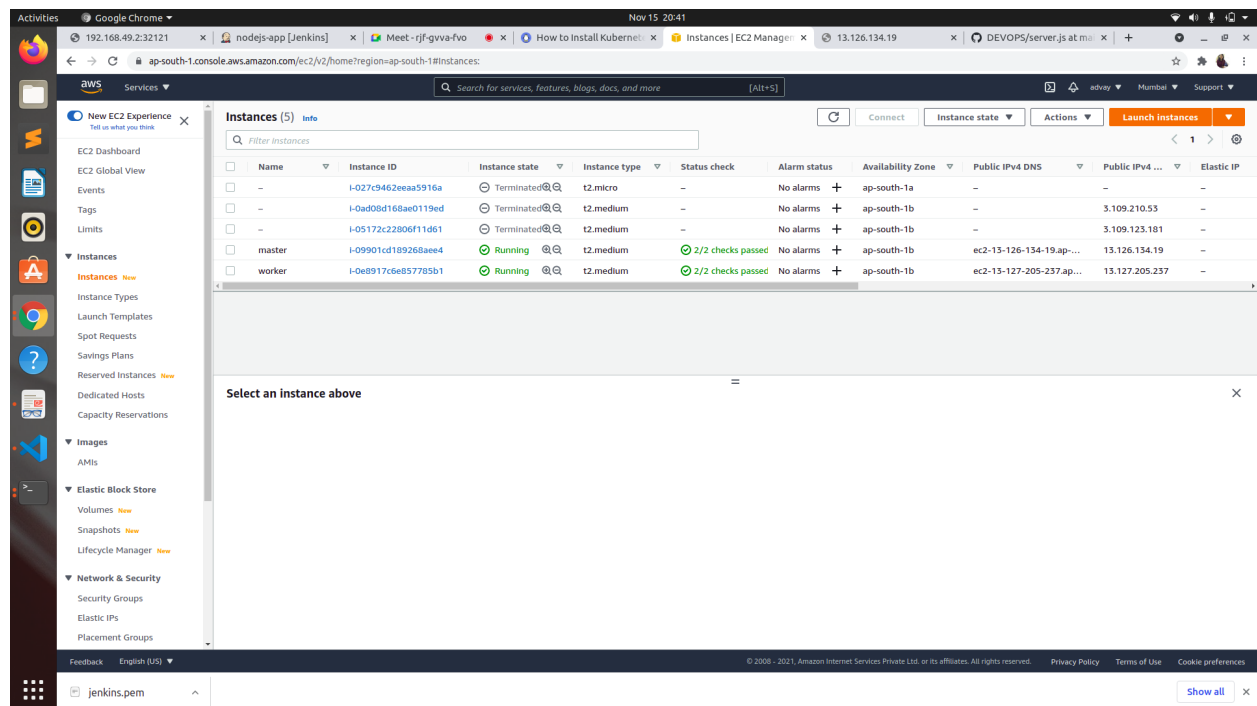
Tools and Technologies used :

- Github
- Docker and Docker hub
- Jenkins
- Kubernetes Cluster

Prerequisites:

- NodeJS v8+
- 2 AWS Ec2 Ubuntu instances of size t2.medium and 15 GB of volumes attached.
- Install Docker and kubernetes on AWS Instances

AWS Instance:



Step - 1 : Setting Up kubernetes Cluster

We have used the kubeadm tool to set up the kubernetes cluster.

Setting up a kubernetes cluster containing 2 nodes master and worker.

1. Install docker and add docker daemon after that enable and start the docker on both the nodes
2. Install kubernetes(Kubeadm,Kubelet and Kubectl) on both the nodes
3. Initialize the master node using kubeadm. Output of this command would be a key , through which worker nodes can join the kubernetes cluster. Copy the token and save it somewhere.

```
Then you can join any number of worker nodes by running the following on each as root:
kubeadm join 172.31.7.209:6443 --token etipst.0p05m5f584805wdk \
--discovery-token-ca-cert-hash sha256:2a0d9a8414faeda13f4693a33d746f68ea0c64d0f5ce8b03345c233810cb4351
```

4. Using the token copied earlier run this command on worker node :
`sudo kubeadm join 172.31.7.219:6443 --token etipst.0p05m5f584805wdk --discovery-token-ca-cert-hashsha256:2a0d9a8414faeda13f4693a33d746f68ea0c64d0f5ce8b03345c233810cb4351`
to get following output

```
ubuntu@ip-172-31-1-54:~$ sudo kubeadm join 172.31.7.209:6443 --token etipst.0p05m5f584805wdk --discovery-token-ca-cert-hash sha256:2a0d9a8414faeda13f4693a33d746f68ea0c64d0f5ce8b03345c233810cb4351
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
 * Certificate signing request was sent to apiserer and a response was received.
 * The kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

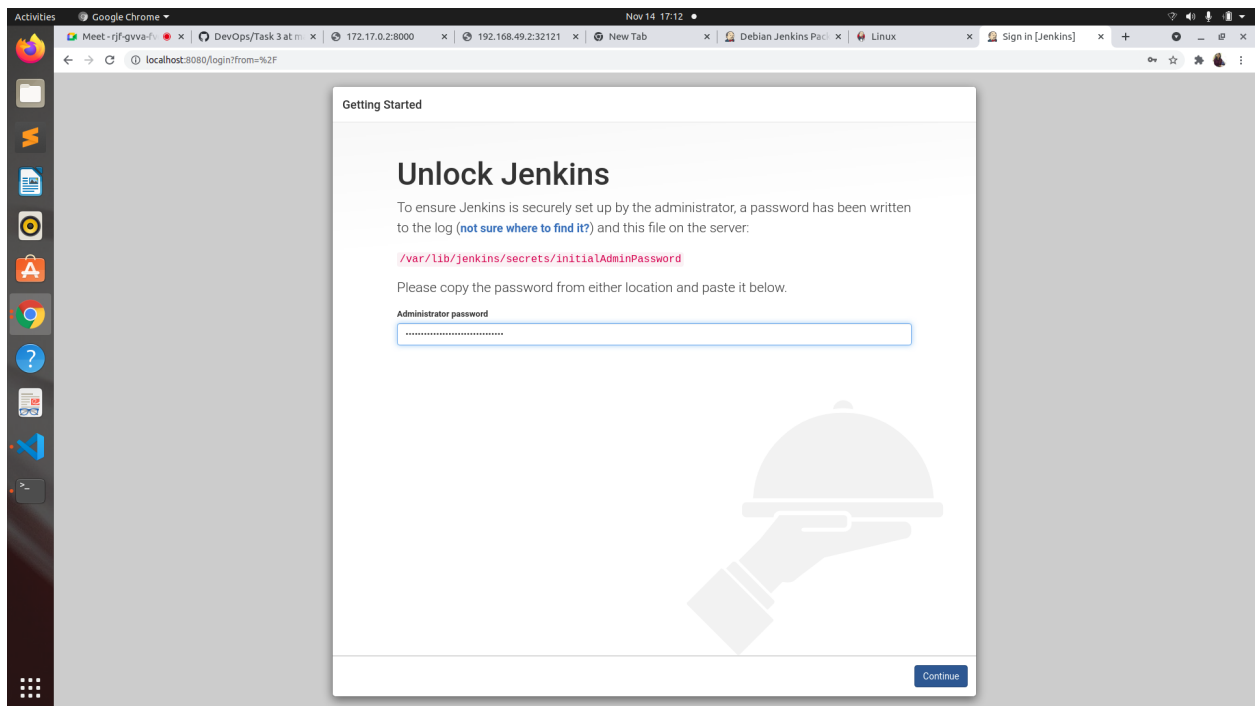
ubuntu@ip-172-31-1-54:~$ client_loop: send disconnect: Broken pipe
meghana@meghana-X510UNR:~/Downloads$
```

5. When we run `kubectl get nodes` on master node Now there would be 2 nodes one master and one newly joined worker node newly joined node's role name would be <none> to label it as worker Using the token copied earlier

```
Last login: Mon Nov 15 13:55:07 2021 from 49.37.191.73
ubuntu@ip-172-31-7-209:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-1-54      Ready    <none>    92s   v1.22.3
ip-172-31-7-209     Ready    control-plane,master   22m   v1.22.3
ubuntu@ip-172-31-7-209:~$
```

Step - 2 : Setting up jenkins locally

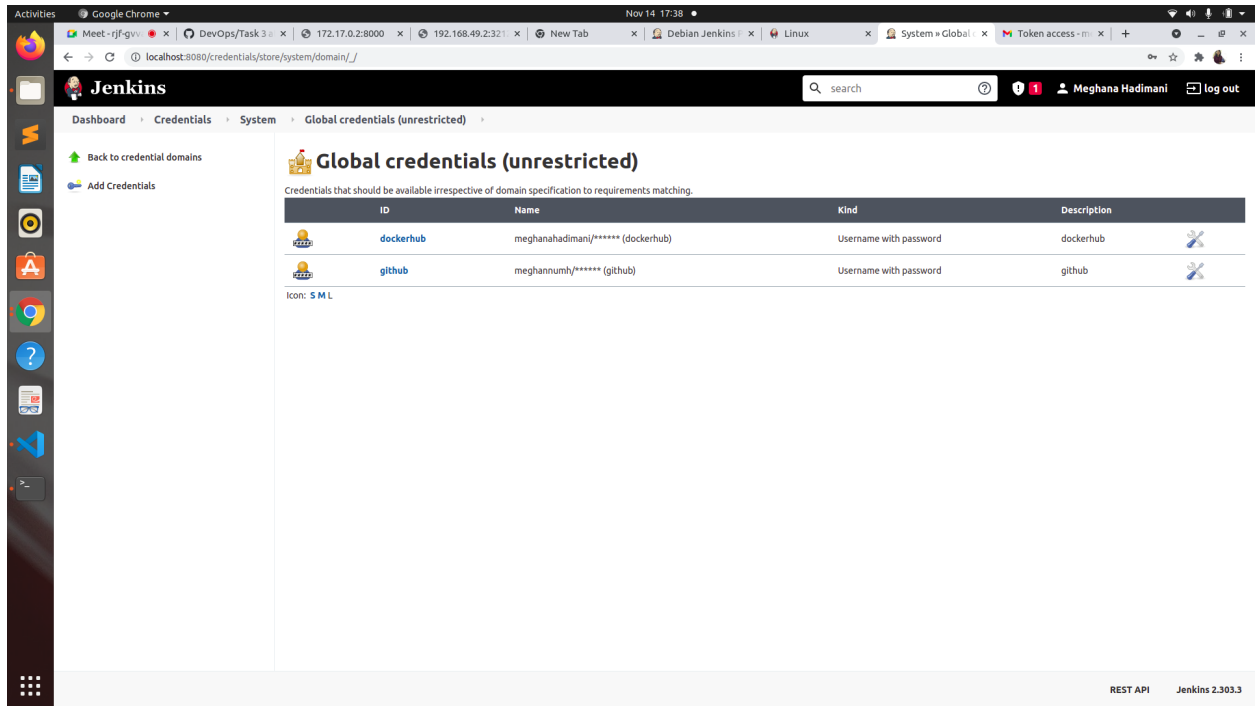
1. Install Java and Jenkins. Run the jenkins file, the first time it asks for an admin password. Run `cat /var/lib/jenkins/secrets/initialAdminPassword` and copy the password and paste it.



Next install the suggested plugins After installing the dashboard opens up Go to Manage Jenkins -> Manage Plugins -> Available and install plugins for Nodejs, Docker, Kubernetes, Github, Kubernetes CLI.

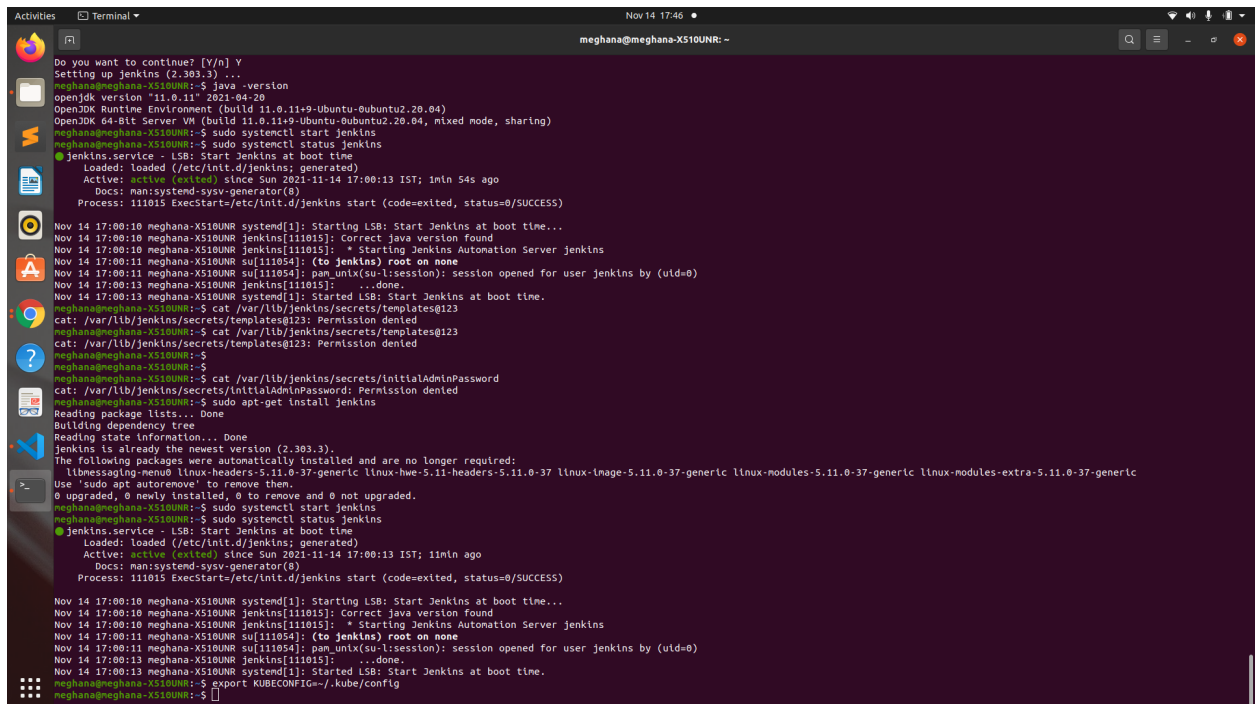
2. Configure Docker Hub and github credentials. Go to Manage Jenkins -> Manage credentials and add a credential for docker hub with your username and password. Create a credential id (which will be used later) and description. Similarly add github credentials, i.e. username

and Personal Access Token.



3. Run the following command to create an environment variable named KUBECONFIG and provide the .kube/config path. Jenkins goes to this path to execute kubectl commands.

```
export KUBECONFIG=~/.kube/config
```



Install docker on jenkins server and add current ubuntu user and jenkins to docker group

Step - 3: Setting up code

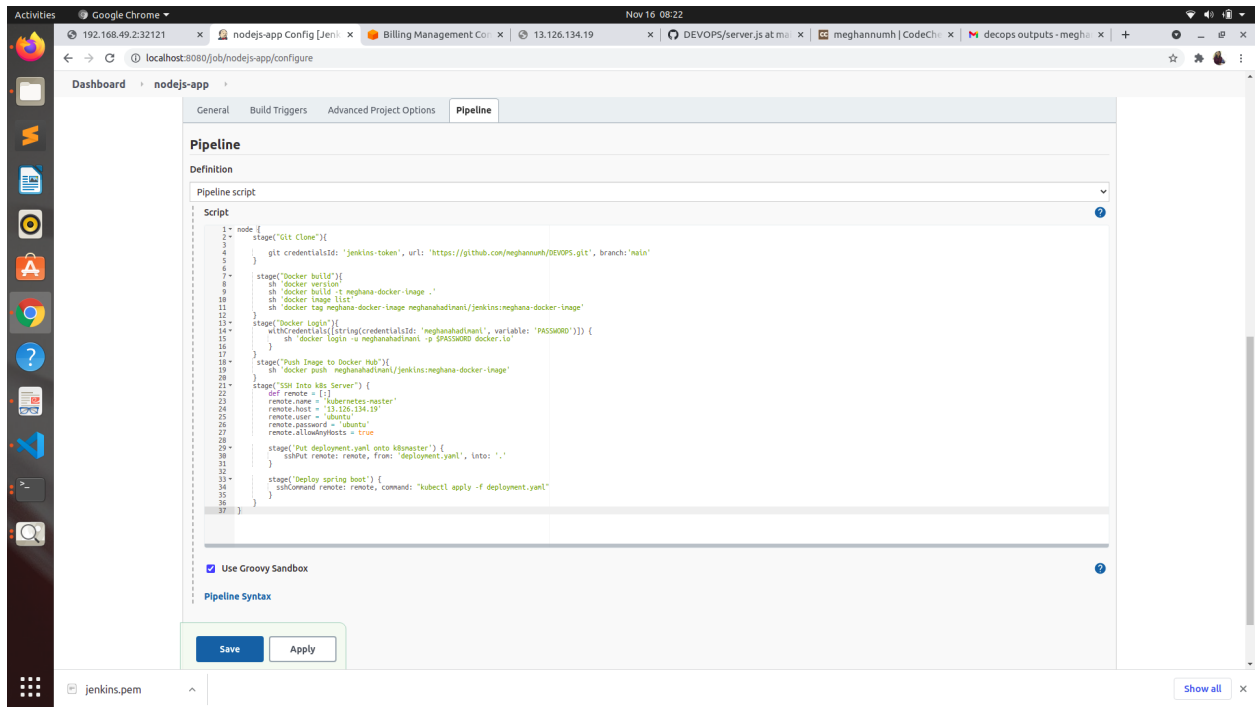
We made a simple node.js application, code is on github. [Code link](#)

[Deployment file link](#)

Step - 4: Building the pipeline

1. Create a jenkins pipeline. Add this code to create stages which include:
 - A. Git clone
 - B. Docker Build
 - C. Docker login
 - D. Push image to docker hub
 - E. Adding ssh into k8 server
 - F. Copy Deployment.yaml file to kubernetes master
 - G. Create the deployment and service on kubernetes

Pipeline code:



The screenshot shows the Jenkins Pipeline configuration page for a project named 'nodes-app'. The 'Pipeline' tab is selected, and the 'Definition' section is expanded. The 'Script' tab is active, displaying a Groovy script for a pipeline. The script includes stages for cloning the repository, building the Docker image, logging into Docker Hub, pushing the image, and deploying to a Kubernetes cluster. The 'Save' button is highlighted.

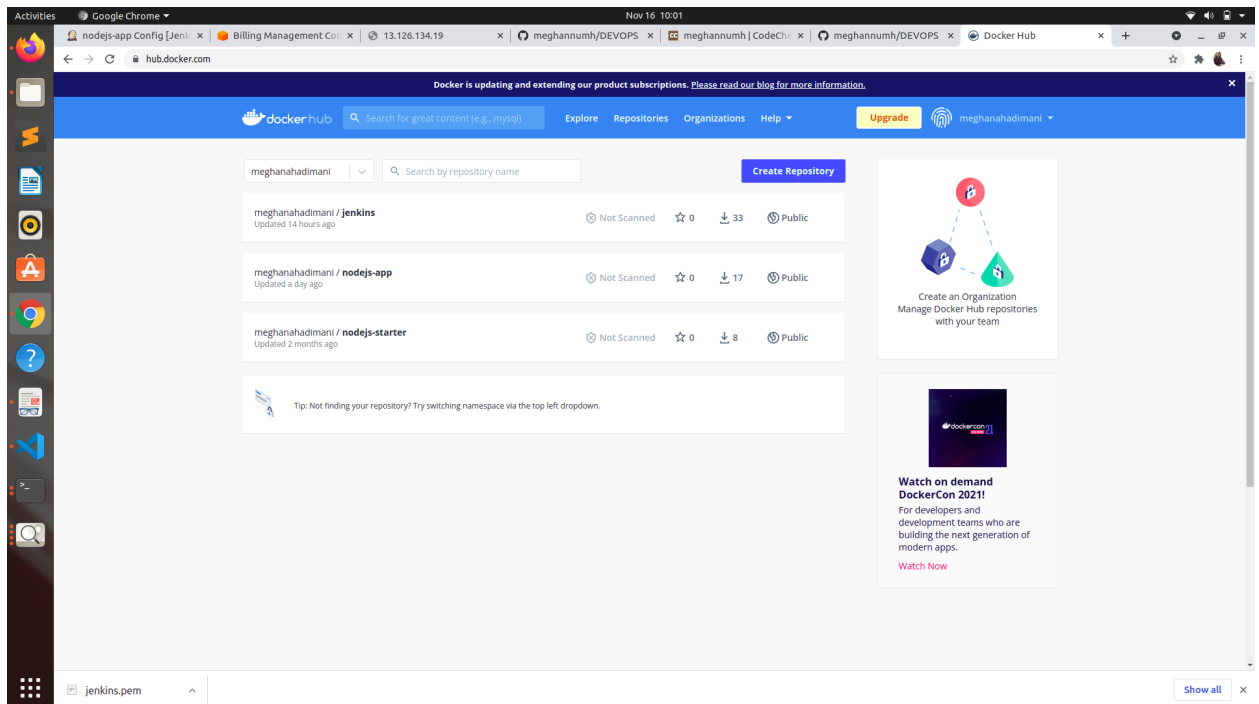
```
1- node {
2-   stage('Git Clone'){
3-     git credentialsId: 'jenkins-token', url: 'https://github.com/meghanumh/DEVOPS.git', branch: 'main'
4-   }
5-   stage('Docker build'){
6-     sh 'docker version'
7-     sh 'docker build -t meghana-docker-image .'
8-     sh 'docker image ls'
9-     sh 'docker tag meghana-docker-image meghanaadimani/jenkins:meghana-docker-image'
10-   }
11-   stage('Docker Login'){
12-     withCredentials([string(credentialsId: 'meghanadimani', variable: 'PASSWORD')]) {
13-       sh 'docker login -u meghanaadimani -p $PASSWORD docker.io'
14-     }
15-   }
16-   stage('Push Image to Docker Hub'){
17-     sh 'docker push meghanaadimani/jenkins:meghana-docker-image'
18-   }
19-   stage('SSH Into k8s Server') {
20-     def remote = {}
21-     remote.name = 'kubernetes-master'
22-     remote.host = '13.126.134.19'
23-     remote.user = 'ubuntu'
24-     remote.password = 'ubuntu'
25-     remote.allowHosts = true
26-     stage('Put deployment.yaml onto k8s master') {
27-       sshPut remote: remote, from: 'deployment.yaml', into: '.'
28-     }
29-     stage('Deploy spring boot') {
30-       sshCommand remote: remote, command: 'kubectl apply -f deployment.yaml'
31-     }
32-   }
33- }
34- }
```

Use Groovy Sandbox

Pipeline Syntax

Save Apply

Pushing image to docker hub:



The screenshot shows the Docker Hub repository page for the user 'meghanadimani'. The page lists three repositories: 'jenkins', 'nodes-app', and 'nodes-starter'. The 'nodes-app' repository is highlighted. The page also includes a 'Create Repository' button and a 'Watch on demand DockerCon 2021!' banner.

meghanadimani / jenkins

Updated 14 hours ago

Not Scanned

0 stars

33 downloads

Public

meghanadimani / nodes-app

Updated a day ago

Not Scanned

0 stars

17 downloads

Public

meghanadimani / nodes-starter

Updated 2 months ago

Not Scanned

0 stars

8 downloads

Public

Tip: Not finding your repository? Try switching namespace via the top left dropdown.

Create Repository

Create an Organization

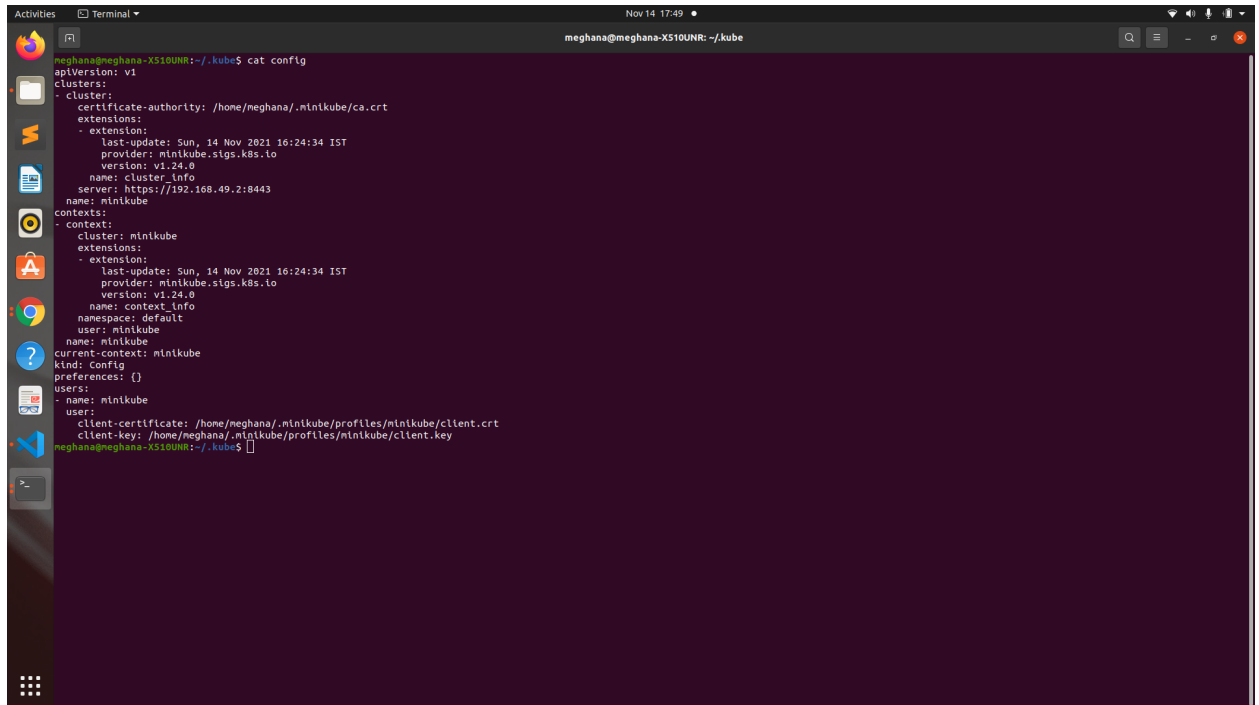
Manage Docker Hub repositories with your team

Watch on demand DockerCon 2021!

For developers and development teams who are building the next generation of modern apps.

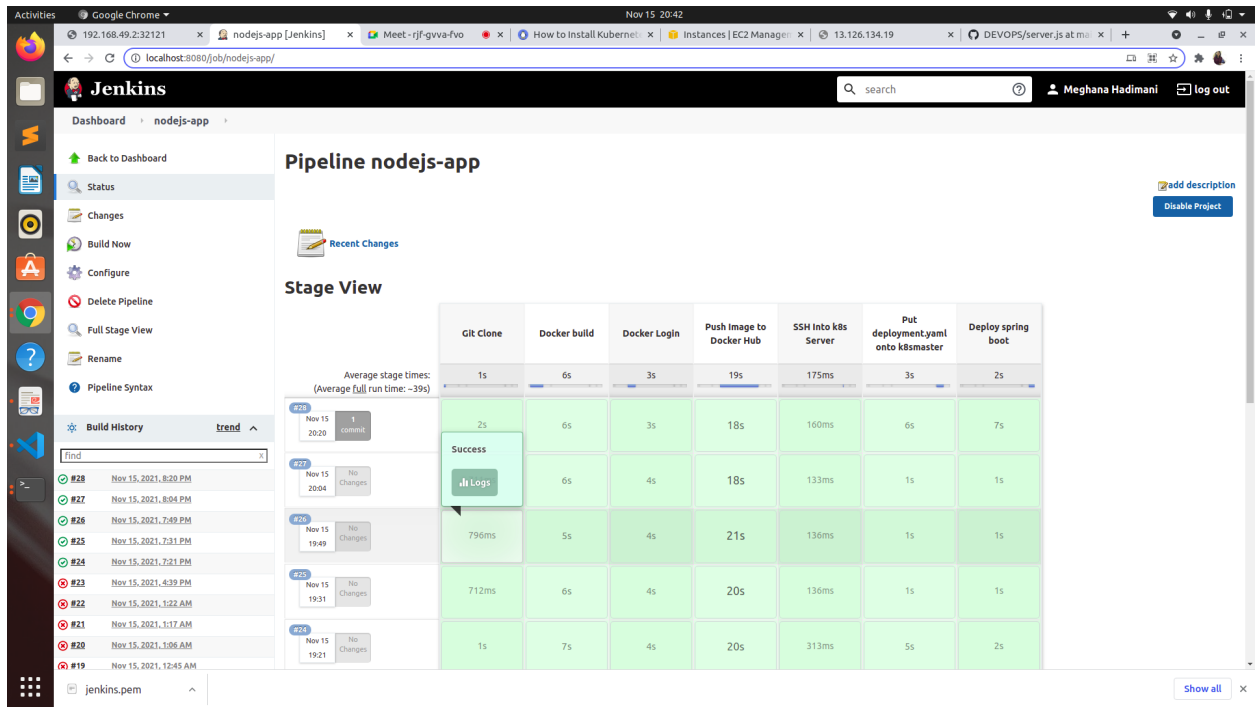
Watch Now

Go to the `~/ .kube` directory and type the command `cat config` to get all the information about the cluster to fill inside the `withKubeConfig` method in the pipeline.



```
meghana@meghana-XS10UNR: ~/.kube$ cat config
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/meghana/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Sun, 14 Nov 2021 16:24:34 IST
        provider: minikube.sigs.k8s.io
        version: v1.24.0
      name: cluster_info
      server: https://192.168.49.2:8443
    name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Sun, 14 Nov 2021 16:24:34 IST
        provider: minikube.sigs.k8s.io
        version: v1.24.0
      name: context_info
      namespace: default
      user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/meghana/.minikube/profiles/minikube/client.crt
    client-key: /home/meghana/.minikube/profiles/minikube/client.key
meghana@meghana-XS10UNR: ~/.kube$
```

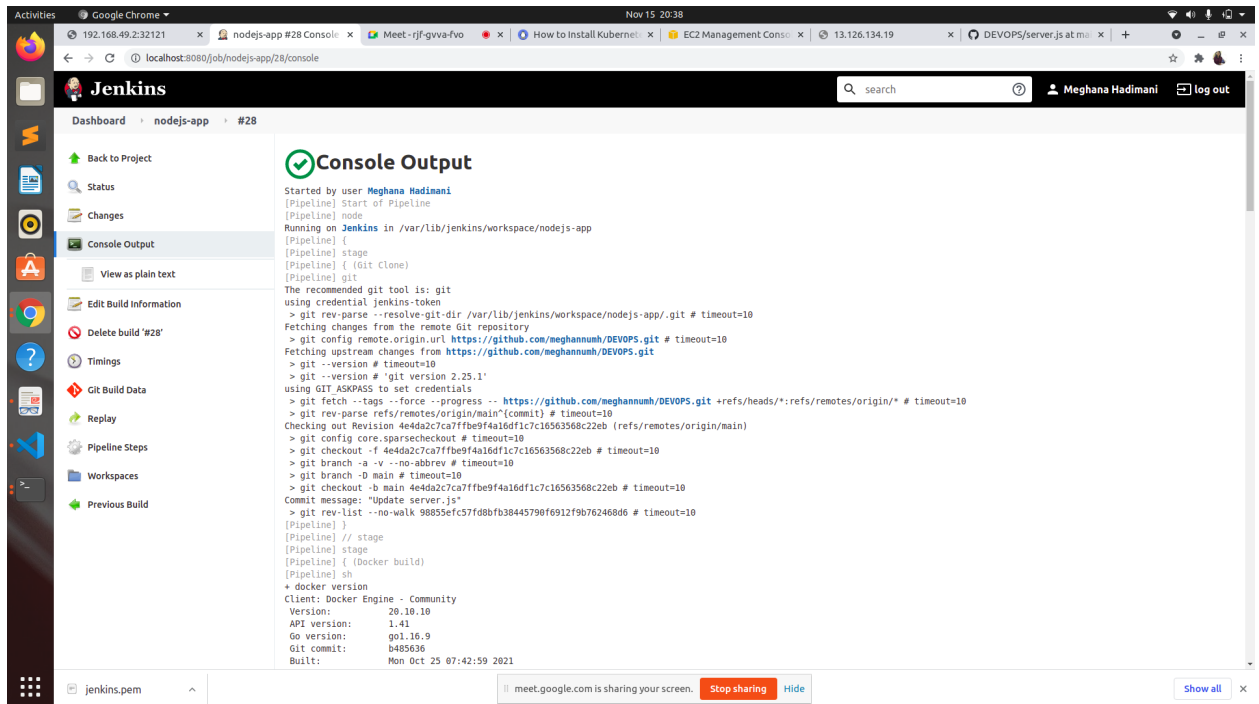

Logs of Deployment on Jenkins:



The screenshot shows the Jenkins web interface for a pipeline named 'nodejs-app'. The 'Stage View' is displayed, showing a table of build stages and their durations. The stages are: Git Clone, Docker build, Docker Login, Push image to Docker Hub, SSH into k8s Server, Put deployment.yaml onto k8smaster, and Deploy spring boot. The table shows the average stage times and the duration of each build run.

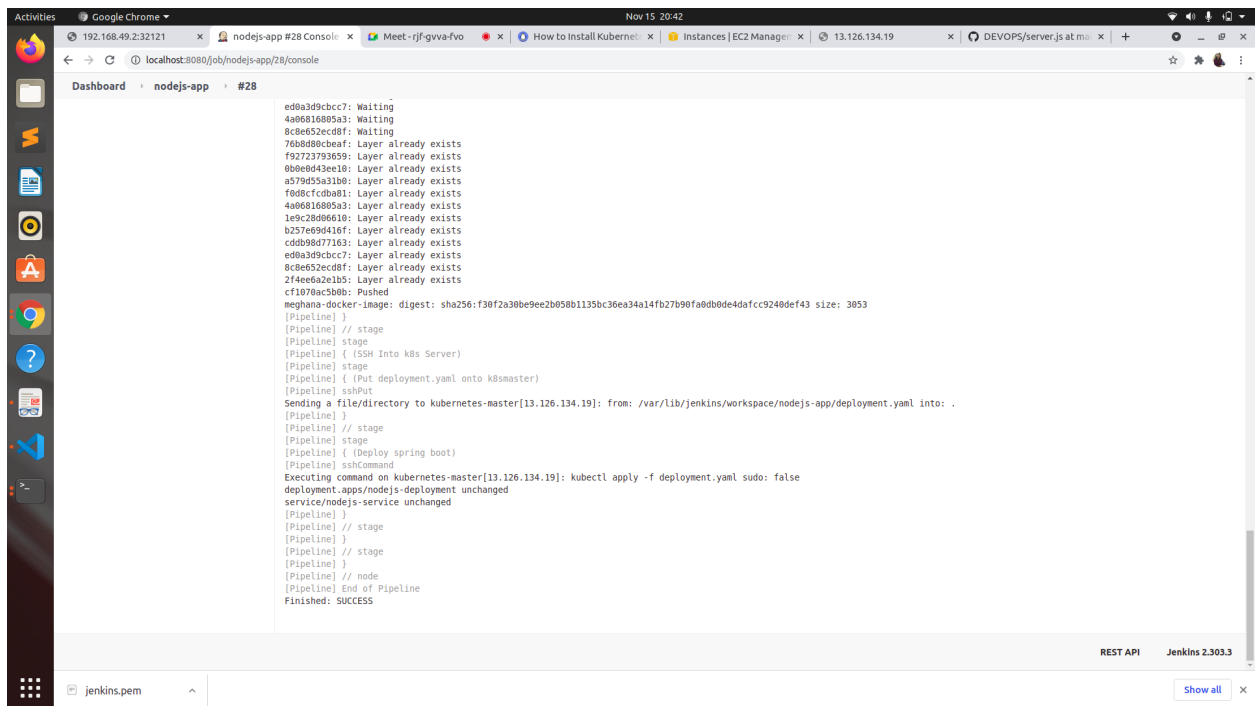
Stage	Git Clone	Docker build	Docker Login	Push image to Docker Hub	SSH into k8s Server	Put deployment.yaml onto k8smaster	Deploy spring boot
Average stage times: (Average full run time: ~39s)	1s	6s	3s	19s	175ms	3s	2s
#28 Nov 15 2020	2s	6s	3s	18s	160ms	6s	7s
#27 Nov 15 2004	Success	6s	4s	18s	133ms	1s	1s
#26 Nov 15 1949	796ms	5s	4s	21s	136ms	1s	1s
#25 Nov 15 1931	712ms	6s	4s	20s	136ms	1s	1s
#24 Nov 15 1921	1s	7s	4s	20s	313ms	5s	2s

Output:



The screenshot shows the Jenkins web interface for the 'nodejs-app' pipeline, specifically the console output for build #28. The output shows the execution of the pipeline stages, including Git Clone, Docker build, Docker Login, Push image to Docker Hub, SSH into k8s Server, Put deployment.yaml onto k8smaster, and Deploy spring boot. The output also shows the Docker Engine version and the commit message.

```
Started by user Meghana Hadimani
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/nodejs-app
[Pipeline] {
[Pipeline] stage
[Pipeline] { (git Clone)
[Pipeline] git
The recommended git tool is: git
using credential jenkins-token
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/nodejs-app/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/meghannumh/DEVOPS.git # timeout=10
Fetching upstream changes from https://github.com/meghannumh/DEVOPS.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/meghannumh/DEVOPS.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 4e4da2c7ca7f94a16df1c7c16563568c22eb (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4e4da2c7ca7f94a16df1c7c16563568c22eb # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main 4e4da2c7ca7f94a16df1c7c16563568c22eb # timeout=10
Commit message: "Update server.js"
> git rev-list --no-walk 98855efc57f08bf38445790f6912f9b762468d6 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Docker build)
[Pipeline] sh
+ docker version
Client: Docker Engine - Community
Version: 20.10.10
API version: 1.41
Go version: go1.16.9
git commit: b485636
Built: Mon Oct 25 07:42:59 2021
```



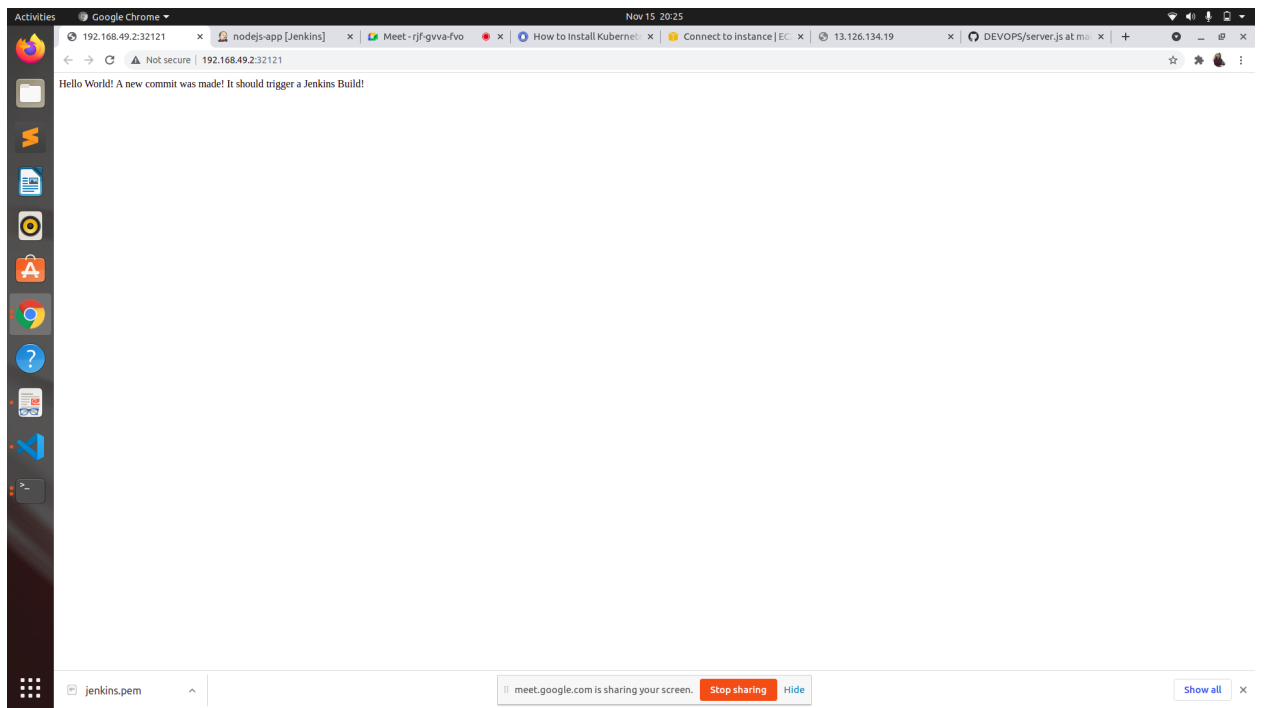
Checking the deployments on kubernetes server

1. To get list of deployments on server run **kubectl get deployments**
2. To get list of services on kubernetes server run **kubectl get services**



node-js service is running on Nodeport, so Deployment is Successful

From the terminal output, node-js-service is running on port number 32121, so open tcp custom 32121 on kubernetes service instance



Application is successfully deployed in kubernetes through Jenkins CI/CD pipeline.