Automation with Jenkins and Kubernetes

Task Overview:

1. Create a container image that has Linux and another basic configuration required to run Slave for Jenkins. (example here we require kubectl to be configured)

2. When we launch the job it should automatically starts job on slave based on the label provided for dynamic approach.

3. Create a job chain of job1 & job2 using build pipeline plugin in Jenkins

4. **Job1**: Pull the GitHub repo automatically when some developers push repo to GitHub and perform the following operations as:

4.1 Create the new image dynamically for the application and copy the application code into that corresponding docker image

4.2 Push that image to the docker hub (Public repository)

(GitHub code contain the application code and Dockerfile to create a new image)

5. **Job2 (Should be run on the dynamic slave of Jenkins configured with Kubernetes kubectl command)**: Launch the application on the top of Kubernetes cluster performing following operations:

5.1 If launching first time then create a deployment of the pod using the image created in the previous job. Else if deployment already exists then do rollout of the existing pod making zero downtime for the user.

5.2 If Application created first time, then Expose the application. Else do not expose it.

Before I started to explain the task, one more thing is set up an environment for a Dynamically created cluster for Jenkins...

Why we need Dynamically created cluster...??

When you run multiple jobs at a same time in Jenkins, they need much computing resources for complete the job. But it is not possible to create hundreds of jobs at the same time in only one master server. That's why we need some setup to

connect multiple resources as slave so they easily distribute the job and perform at same time. This setup is known as a cluster.

We normally see two types of clusters: one is static and the other is dynamic...

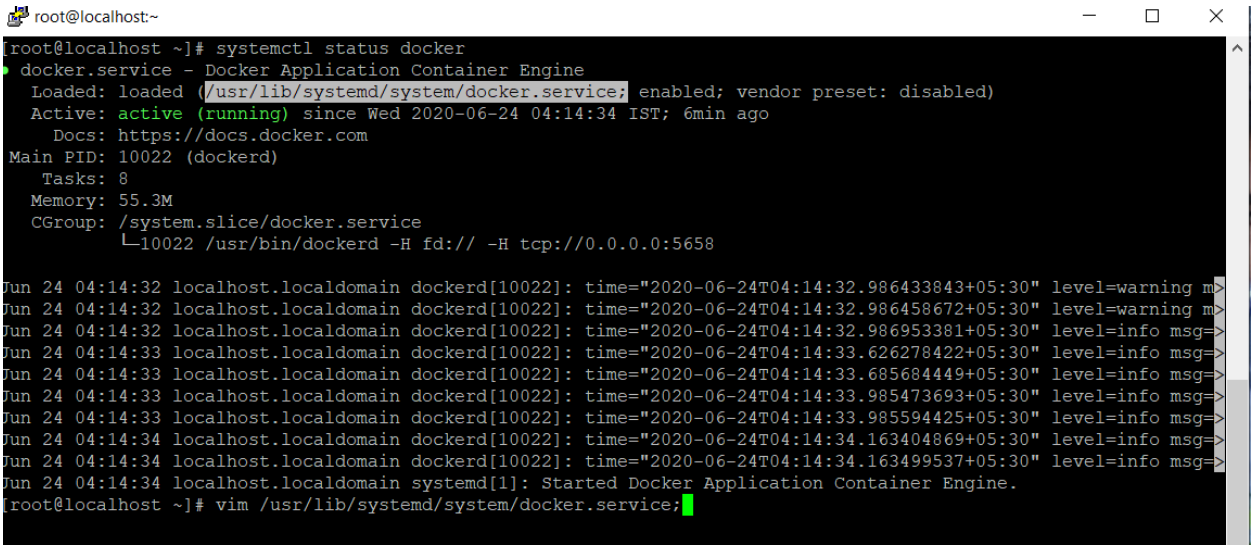
Static Cluster: For creating a static cluster, we need active permanent resources so that when you run your job in Jenkins, they run the job without any delay. But here is one drawback you can see that there is pure wastage of resources since machines are running continuously with no use until required.

Dynamic Cluster: We can overcome the above drawback by using dynamic cluster. We normally use container here, so they easily launch when you need to run the job and terminate after it. No wastage of resources, the flexibility of creating the job and running the environment when we needed.

Here I implemented the dynamic cluster for completing the project...

So first we see how to create a dynamic cluster in Jenkins using Docker containers.

- I use two VM's here, one for docker service and another for docker client.
- In docker service, you need to do some changes...



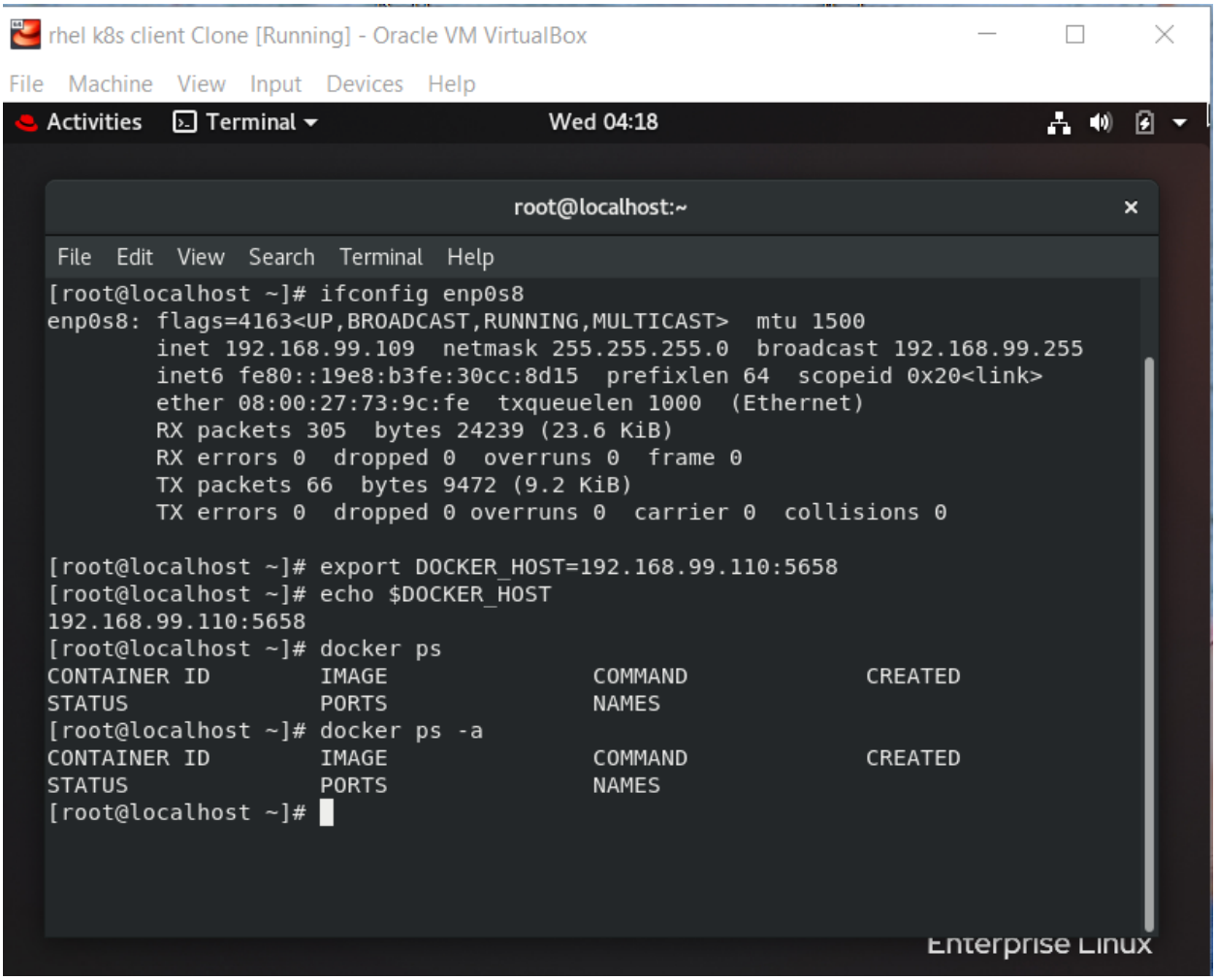
```
root@localhost:~
[root@localhost ~]# vim /usr/lib/systemd/system/docker.service;
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target firewalld.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:5658
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not supports it.
# Only systemd 226 and above support this option.
TasksMax=infinity

# set delegate yes so that systemd does not reset the cgroups of docker containers
```

```
root@localhost:~
[root@localhost ~]# vim /usr/lib/systemd/system/docker.service;
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
[root@localhost ~]#
```

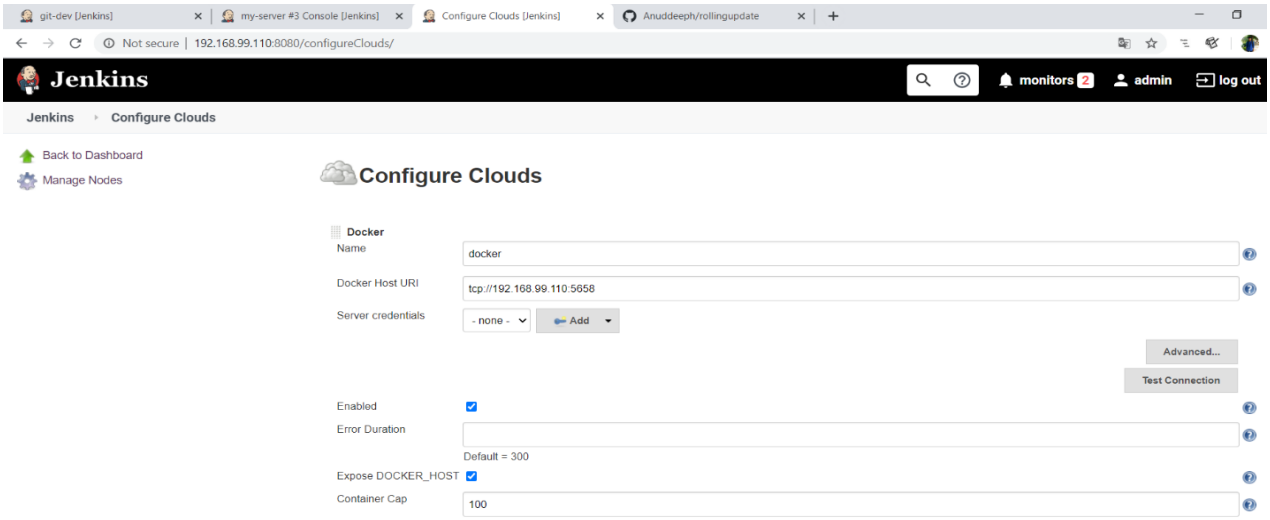
You must do these three steps and your docker service is ready for connect client from another VM.

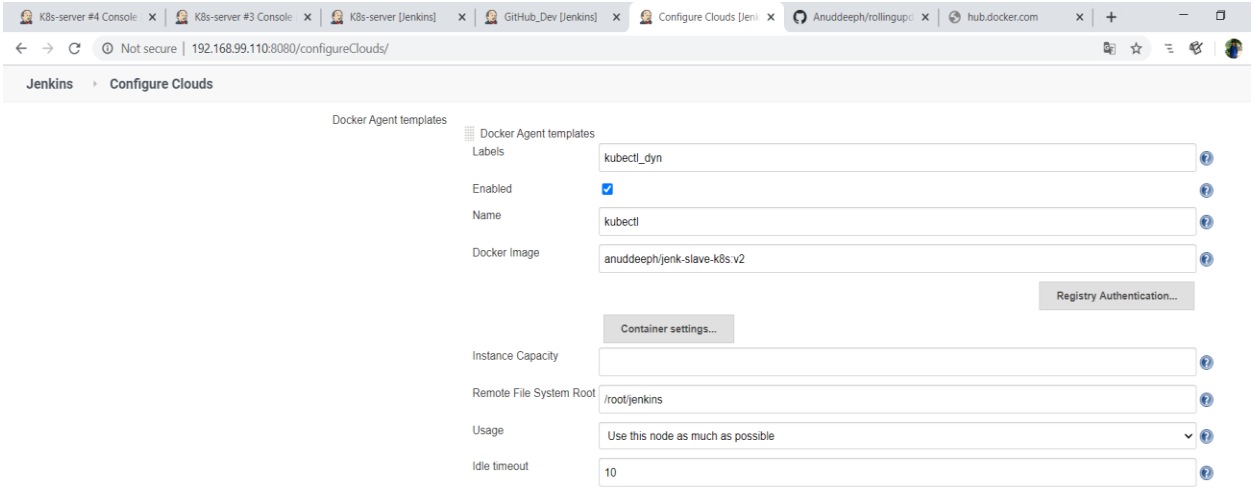
- Now come to another VM and stop the docker services using **systemctl stop docker**. After this you need to export the DOCKER_HOST, so that you can use this VM as a docker client.

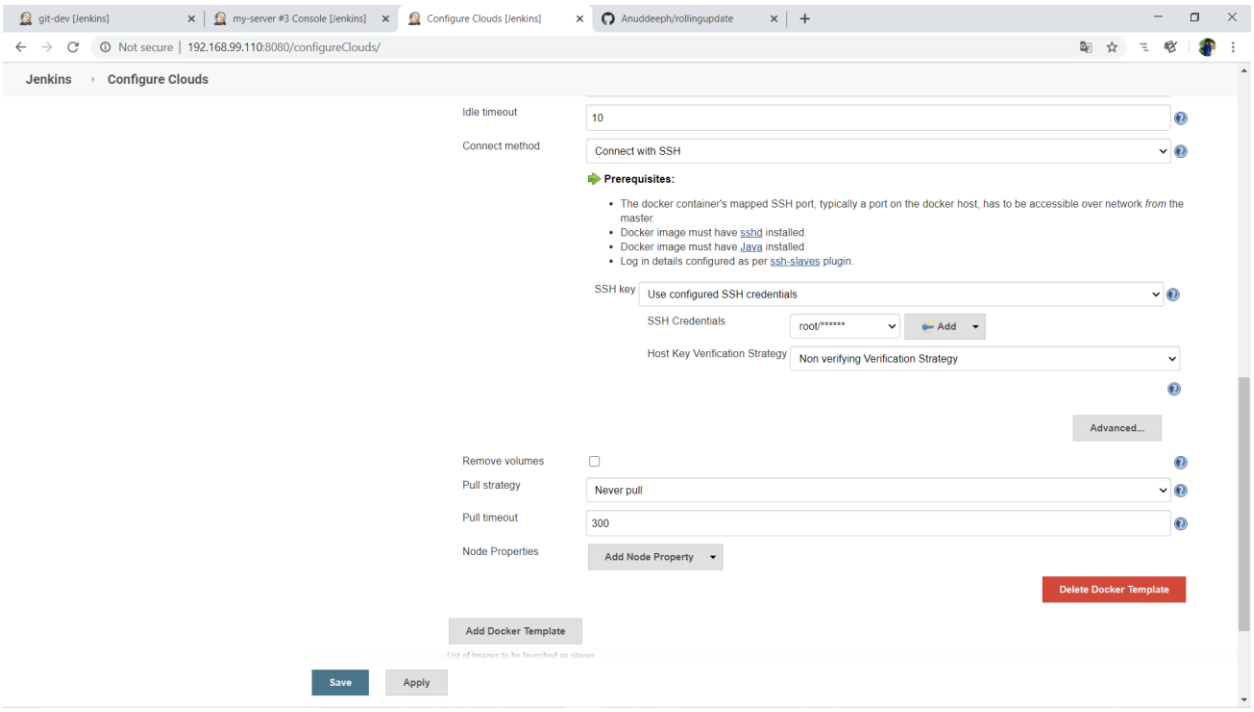
```
export DOCKER_HOST=IP_of_server:port_you_give_in_docker_service_file
```



- Now come to Jenkins, Here install Docker plugin first.
- Go to Configure -> Manage Node and Clouds -> configure cloud.





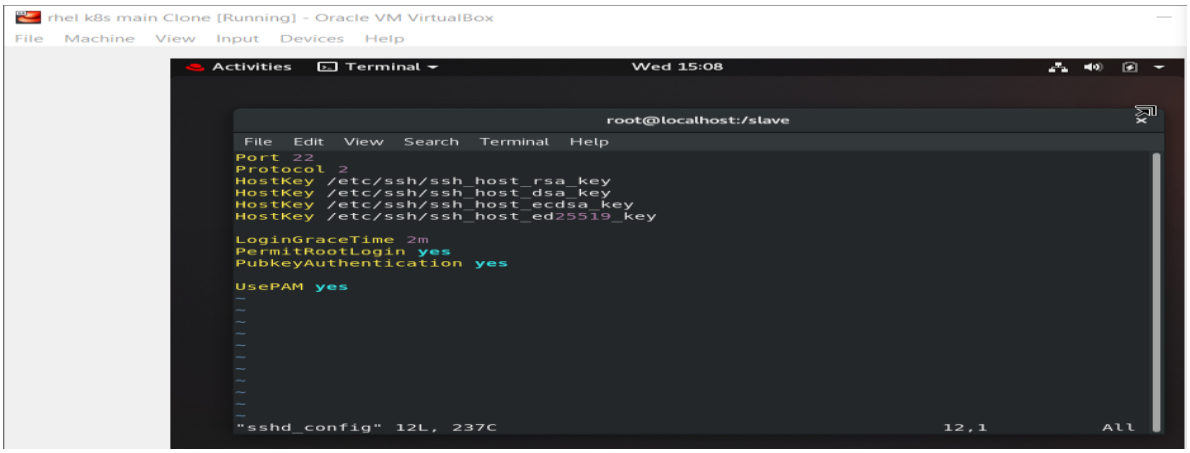


Now my docker cloud is configured. Now let us come to task...

Project Description:

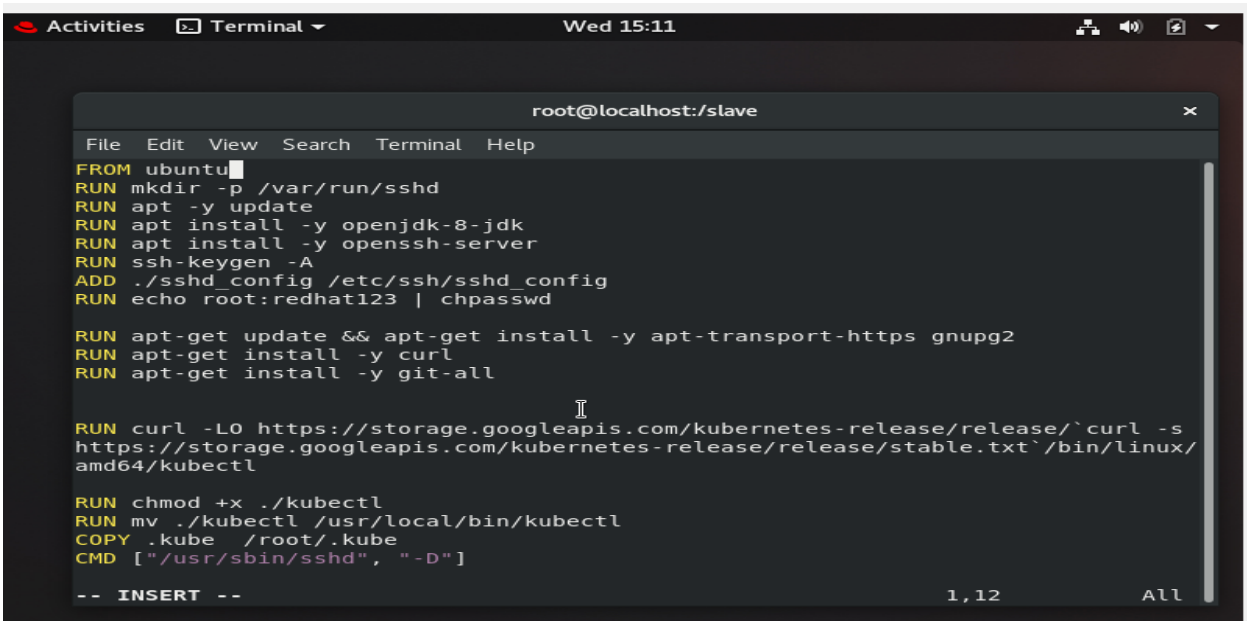
1. Create a container image that has Linux and another basic configuration required to run Slave for Jenkins:

I use ubuntu for creating the Dockerfile which also has kubectl configured...



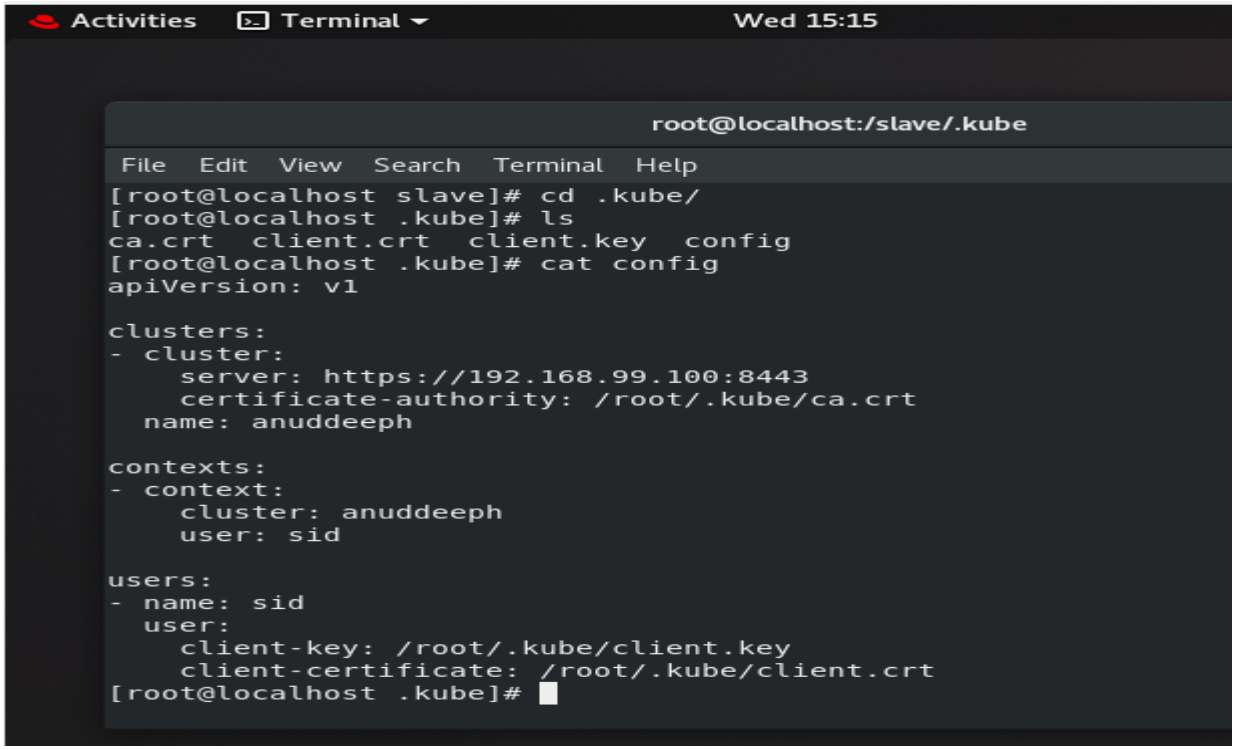

```
Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

LoginGraceTime 2m
PermitRootLogin yes
PubkeyAuthentication yes

UsePAM yes
```

Make a folder and Save this file as ***sshd_config*** to configure the ssh in Container.



```
FROM ubuntu

RUN mkdir -p /var/run/sshd
RUN apt -y update
RUN apt install -y openjdk-8-jdk
RUN apt install -y openssh-server
RUN ssh-keygen -A
ADD ./sshd_config /etc/ssh/sshd_config
RUN echo root:redhat123 | chpasswd
RUN apt-get update && apt-get install -y apt-transport-https gnupg2
RUN apt-get install -y curl
RUN apt-get install -y git-all
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl
-s https://storage.googleapis.com/kubernetes-
release/release/stable.txt`/bin/linux/amd64/kubectl
RUN chmod +x ./kubectl
RUN mv ./kubectl /usr/local/bin/kubectl
COPY .kube  /root/.kube
CMD ["/usr/sbin/sshd", "-D"]
```

RUN command ***docker build -t image:tag .*** for creating own image. Put both Dockerfile and sshd_config in same folder. This is the content of my .kube folder.



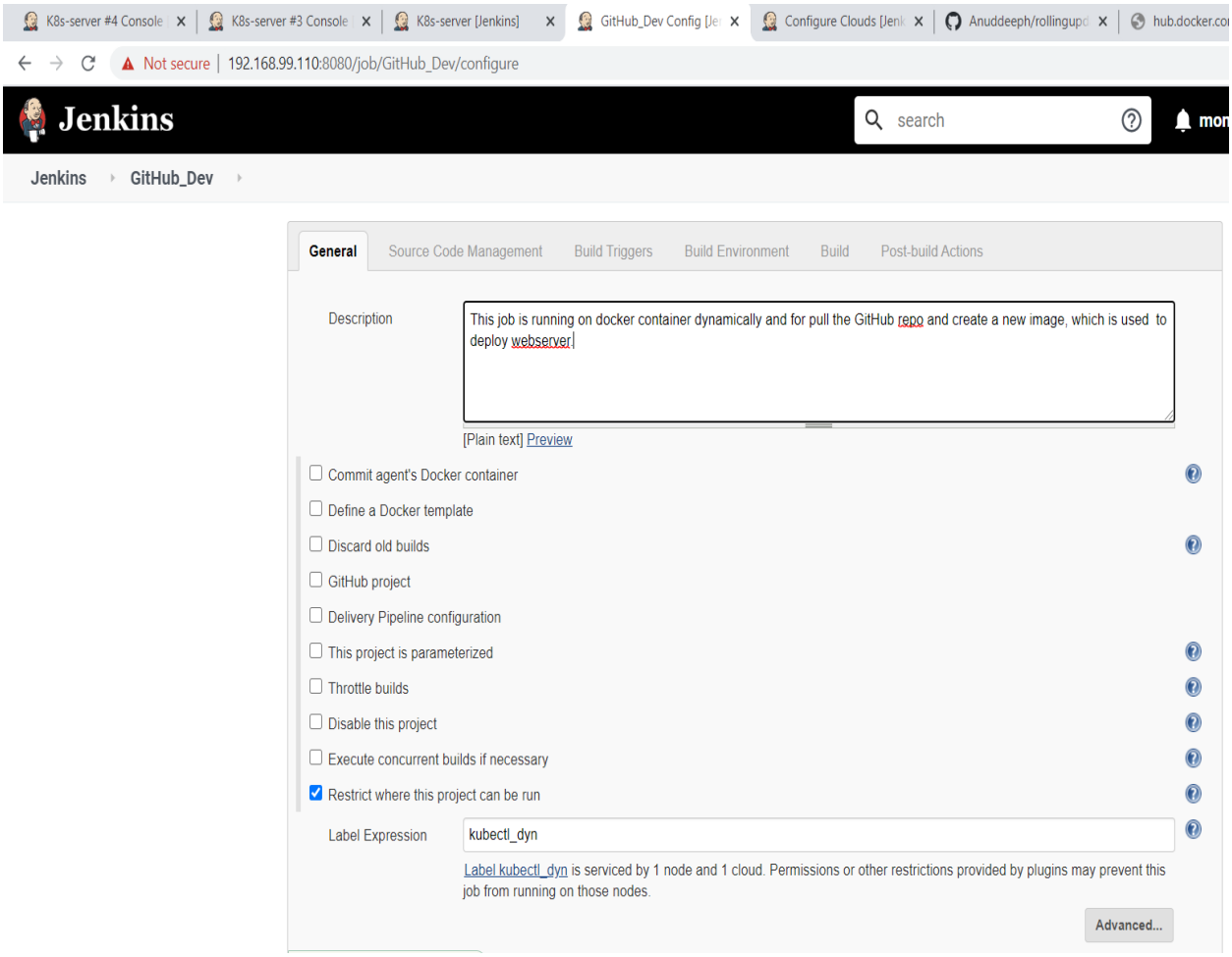
This is my Dockerfile for creating an image that runs on Jenkins as cloud node. Here I also copy kube_config file to image so it run as a slave for my base K8s cluster.

2. Jobs in Jenkins:

- **Job1: (Copy code and Dockerfile from GitHub and create an image using Dockerfile and push it in to docker hub.)**

This is the link of my GitHub Repo:

https://github.com/Anuddeeph/rollingupdate.git



Source Code Management

- ○ None
- ◉ Git

Repositories

Repository URL: https://github.com/Anuddeeph/rollingupdate.git

Credentials: - none - | Add

Advanced...

Add Repository

Branches to build

X

Branch Specifier (blank for 'any'): */master

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☑ Poll SCM

Schedule

```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Wednesday, June 24, 2020 7:38:19 PM IST; would next run at Wednesday, June 24, 2020 7:38:19 PM IST.

☐ Ignore post-commit hooks

General | Source Code Management | Build Triggers | Build Environment | **Build** | Post-build Actions

Execute shell

Command

```
mkdir -p /mycode
cp -rvf * /mycode
```

See the list of available environment variables

Advanced...

Build / Publish Docker Image

Directory for Dockerfile: /mycode

Location to look for the Dockerfile in, which is used to build the image.

Advanced...

Cloud: Cloud this build is running on

Cloud to use to build image

Image: anuddeeph/http-php-server:latest

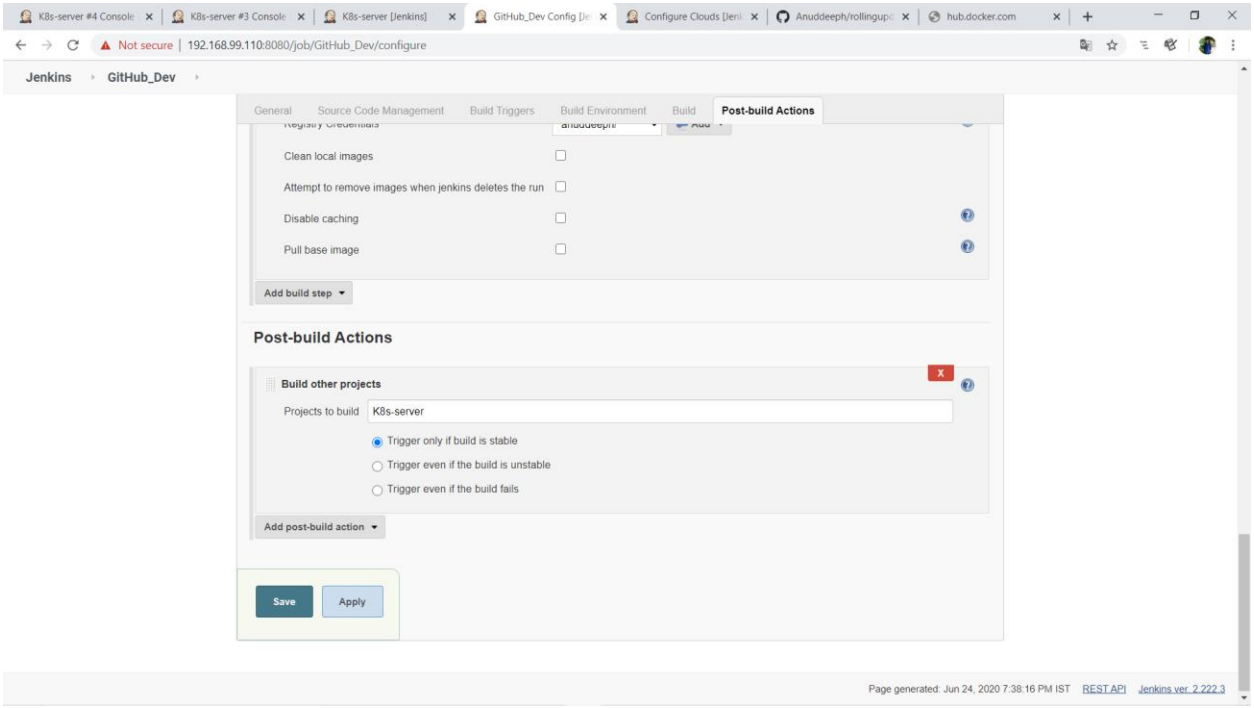
Push image: ☑

Registry Credentials: anuddeeph/****** Add

Clean local images: ☐

Attempt to remove images when jenkins deletes the run: ☐

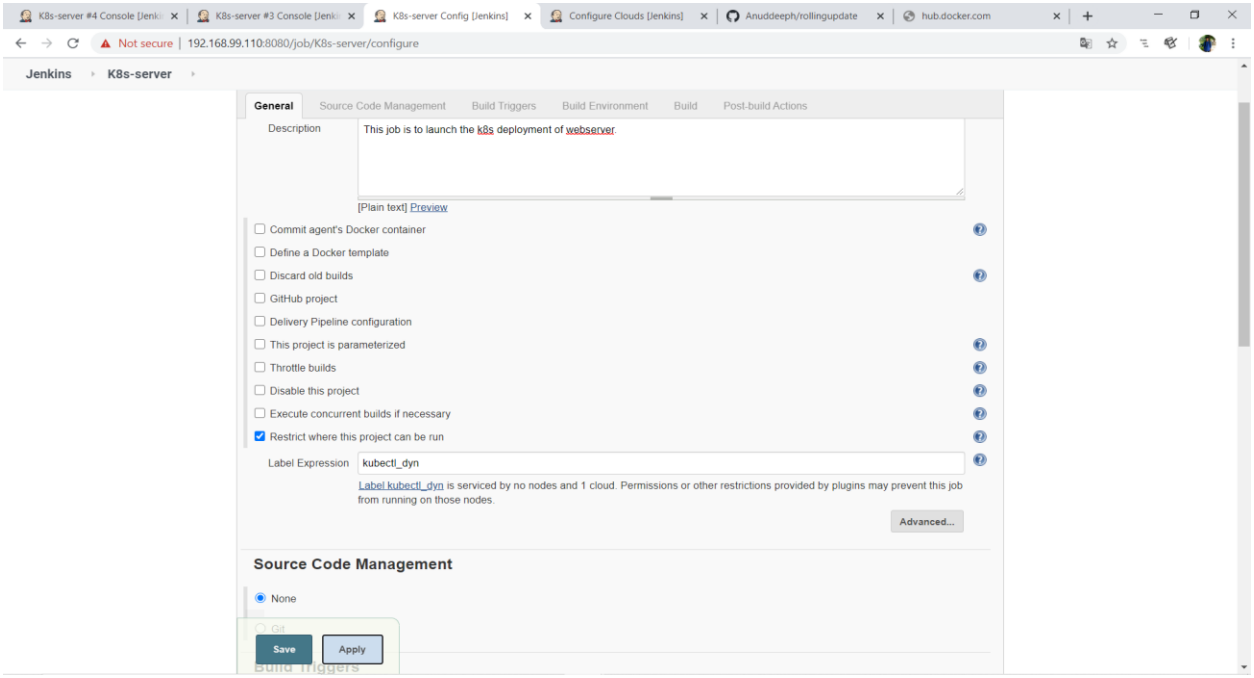
Save | Apply

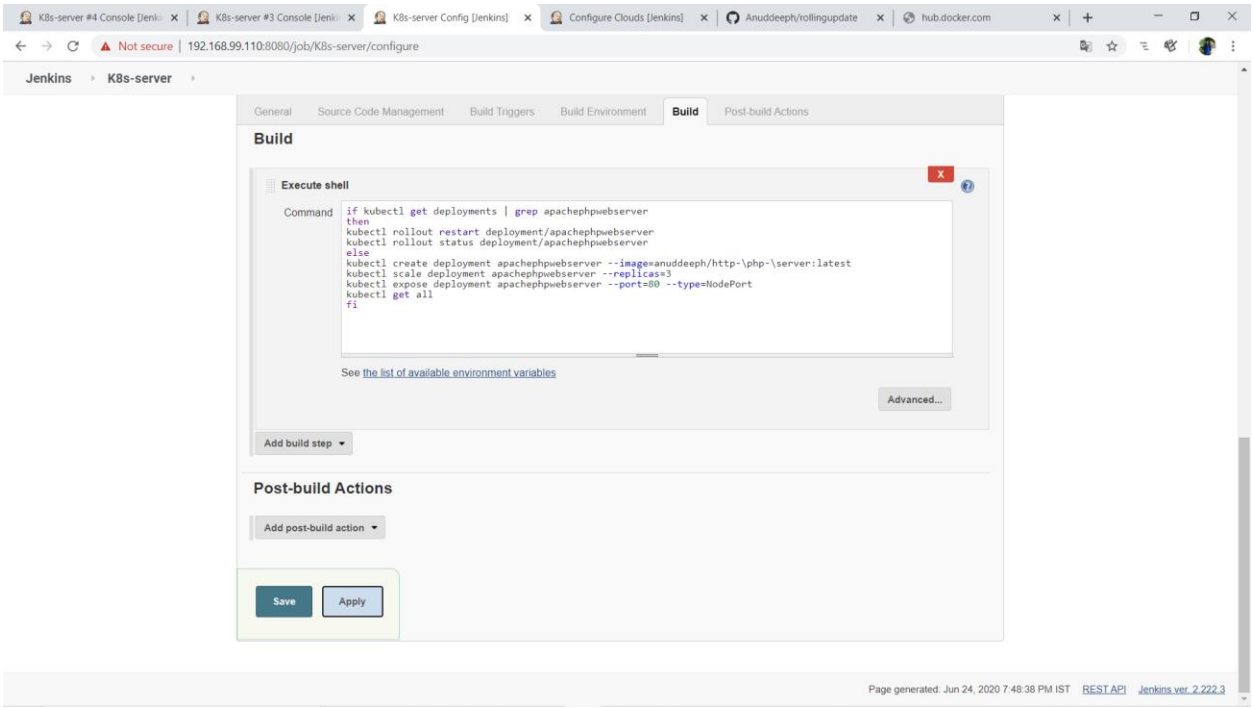


Now come to next job...

- **Job2: (Launch the application deployed on top of Kubernetes cluster...)**

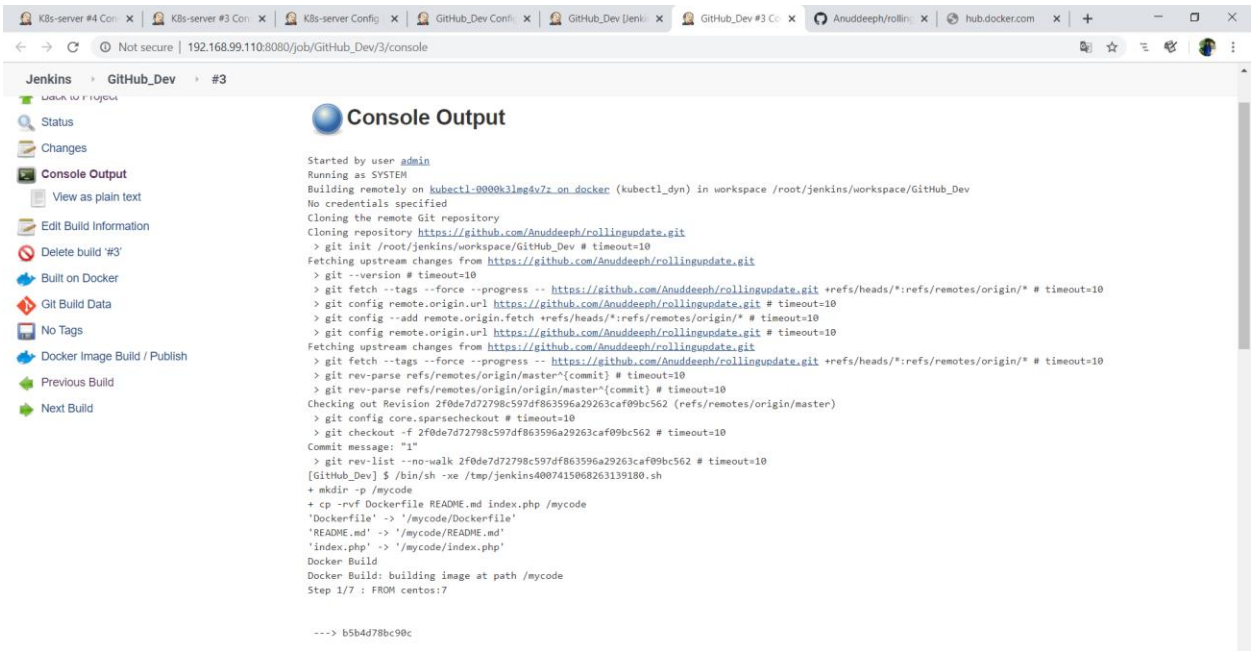
➔ Before building this job start minikube using command minikube start.

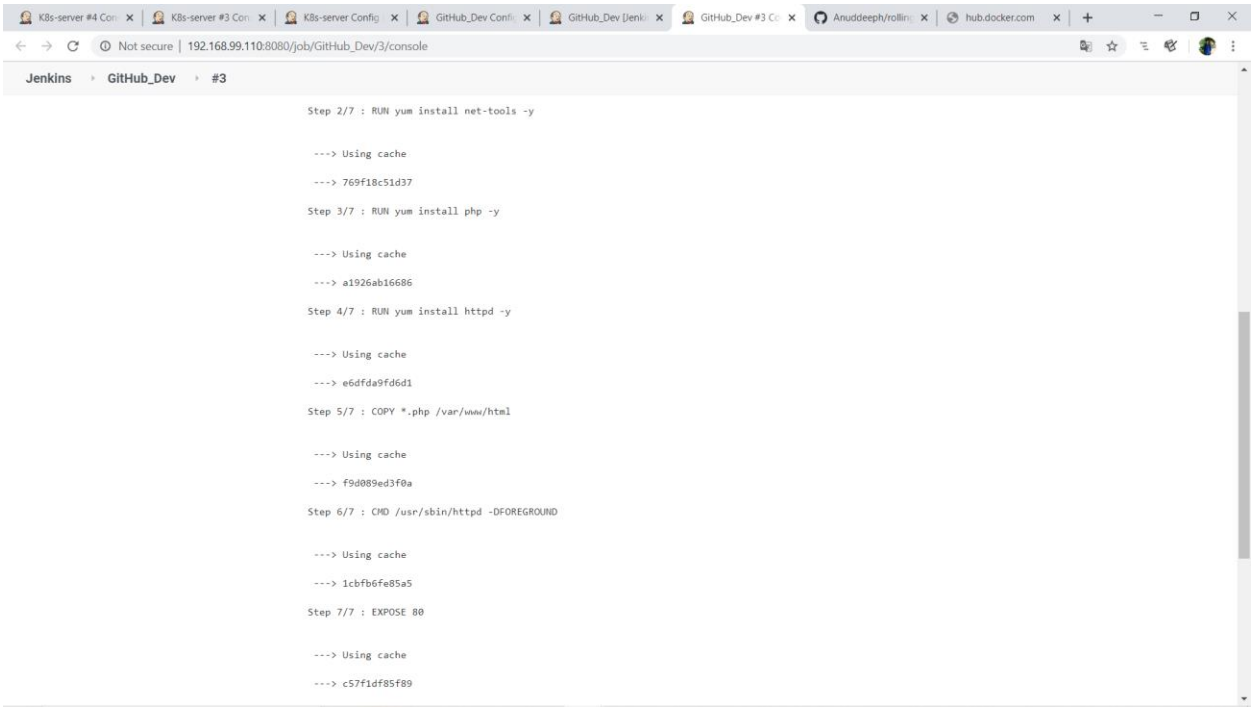




This code is deployed our web-services and when developer push any code again then its rollout the update without any downtime. This is my [docker-hub](). where job1 automatically uploaded the docker-image build by Dockerfile.

Now let us run our job1 and see the output:





```
Step 7/7 : EXPOSE 80

 ---> Using cache

 ---> c57f1df85f89

Successfully built c57f1df85f89

Tagging built image with anuddeeph/http-php-server:latest
Docker Build Response : c57f1df85f89
Pushing [anuddeeph/http-php-server:latest]
The push refers to repository [docker.io/anuddeeph/http-php-server]
4cc78314b6a8: Preparing
099d48031ada: Preparing
d58714fd8413: Preparing
e2e59c9608a5: Preparing
edf3aa290fb3: Preparing
4cc78314b6a8: Layer already exists
099d48031ada: Layer already exists
e2e59c9608a5: Layer already exists
d58714fd8413: Layer already exists
edf3aa290fb3: Layer already exists
latest: digest: sha256:6fb21f6f8dd65b8bf990f8f5018952dfca478c11df66e584385e41c61bc54f42 size: 1372
Docker Build Done
Triggering a new build of K8s-server
Finished: SUCCESS
```

Page generated: Jun 24, 2020 7:54:50 PM IST REST API Jenkins ve

Now my Docker image is created and push to docker-hub.

This trigger the job2... When job2 is running first time, it creates deployment.

K8s-server #4 Con | K8s-server #3 Con | K8s-server Config | GitHub_Dev Confi | GitHub_Dev [Jenki | GitHub_Dev #3 Co | Anuddeeph/rolling | hub.docker.com

Not secure | 192.168.99.110:8080/job/K8s-server/3/console

Jenkins search monitors 2 admin log out

Jenkins › K8s-server › #3

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete build '#3'
- Built on Docker
- Previous Build
- Next Build

Console Output

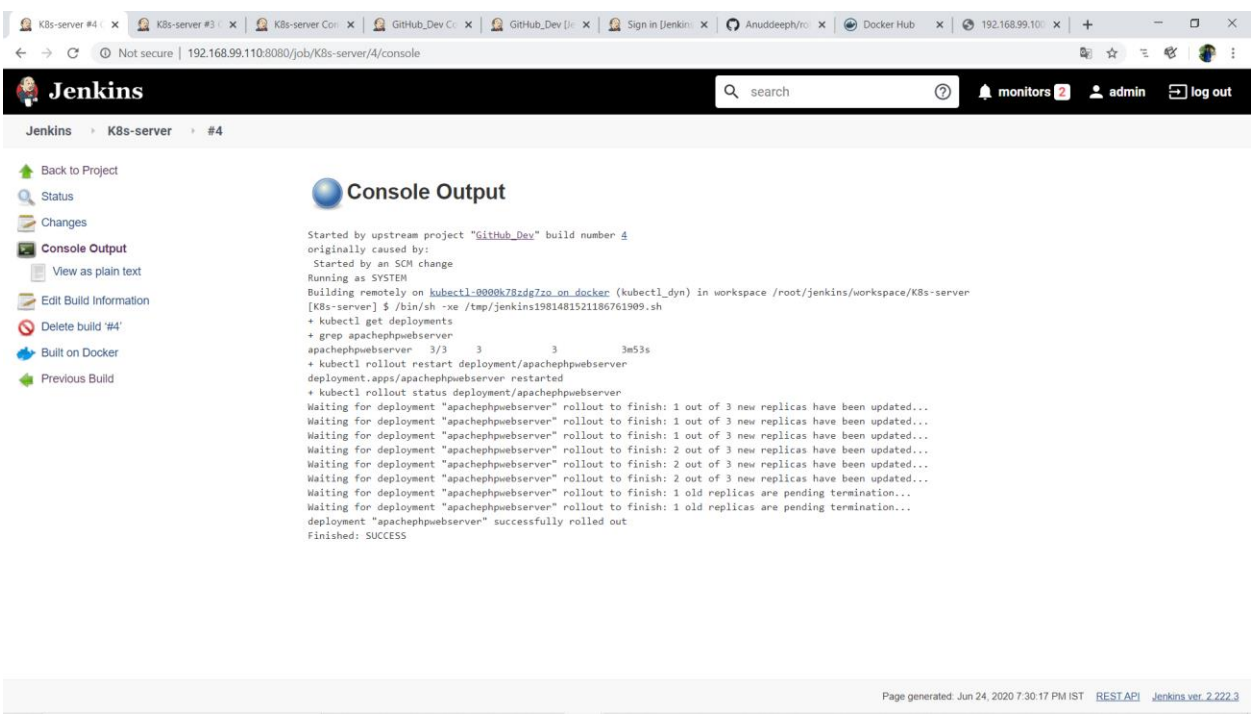
```
Started by upstream project "GitHub_Dev" build number 3
originally caused by:
 Started by user admin
Running as SYSTEM
Building remotely on kubectl-0000k4b7ogzs1 on docker (kubectl_dyn) in workspace /root/jenkins/workspace/K8s-server
[K8s-server] $ /bin/sh -xe /tmp/jenkins1127585836673330364.sh
+ grep apachephpwebserver
+ kubectl get deployments
+ kubectl create deployment apachephpwebserver --image=anuddeeph/http-php-server:latest
deployment.apps/apachephpwebserver created
+ kubectl scale deployment apachephpwebserver --replicas=3
deployment.apps/apachephpwebserver scaled
+ kubectl expose deployment apachephpwebserver --port=80 --type=NodePort
service/apachephpwebserver exposed
+ kubectl get all

+ kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/apachephpwebserver-f9f767658-5znbx  0/1     ContainerCreating   0          1s
pod/apachephpwebserver-f9f767658-8plc9  0/1     ContainerCreating   0          0s
pod/apachephpwebserver-f9f767658-j9kxz  0/1     ContainerCreating   0          0s
pod/mygra-7c5bc5fb87-5dczc              1/1     Running             5          26d
pod/myprom-69684ff8c5-qcjbt             1/1     Running             5          26d
pod/phpwebserver-754db45644-8bptt       1/1     Running             0          14m
pod/phpwebserver-754db45644-94fst       1/1     Running             0          14m
pod/phpwebserver-754db45644-zphzz       1/1     Running             0          14m

NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/apachephpwebserver   NodePort    10.103.255.147   <none>        80:31025/TCP     0s
service/kubernetes        ClusterIP   10.96.0.1        <none>        443/TCP          26d
service/mygra             NodePort    10.108.141.10    <none>        3000:30341/TCP   26d
service/myprom            NodePort    10.96.157.22     <none>        9090:30570/TCP   26d
service/phpwebserver      NodePort    10.101.19.135    <none>        80:31135/TCP     28m

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apachephpwebserver      0/3     3            0           1s
deployment.apps/mygra                   1/1     1            1           26d
deployment.apps/myprom                  1/1     1            1           26d
deployment.apps/phpwebserver            3/3     3            3           28m

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/apachephpwebserver-f9f767658      3         3         0       1s
replicaset.apps/mygra-7c5bc5fb87                  1         1         1       26d
replicaset.apps/myprom-69684ff8c5                 1         1         1       26d
replicaset.apps/phpwebserver-754db45644           3         3         3       14m
replicaset.apps/phpwebserver-84649cb7fb           0         0         0       28m
Finished: SUCCESS
```

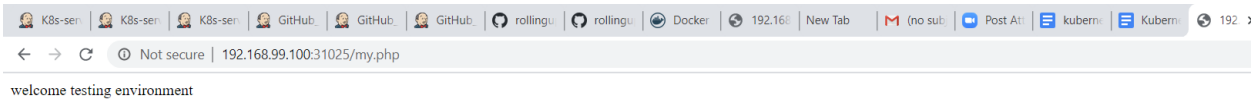
Page generated: Jun 24, 2020 7:54:29 PM IST REST API Jenkins ver. 2.222.3

```
C:\Users\Anuddeeph Nalla>curl 192.168.99.100:31025

welcome to anuddeeph testing webserver
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.10  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:0a  txqueuelen 0  (Ethernet)
        RX packets 14  bytes 1653 (1.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 2144 (2.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

              total        used        free      shared  buff/cache   available
Mem:           2136         839         103         510        1194         682
Swap:             0           0           0

C:\Users\Anuddeeph Nalla>curl 192.168.99.100:31025

welcome to anuddeeph testing webserver
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.13  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:0d  txqueuelen 0  (Ethernet)
        RX packets 16  bytes 1419 (1.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 1776 (1.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

              total        used        free      shared  buff/cache   available
Mem:           2136         839         102         510        1194         682
Swap:             0           0           0
```

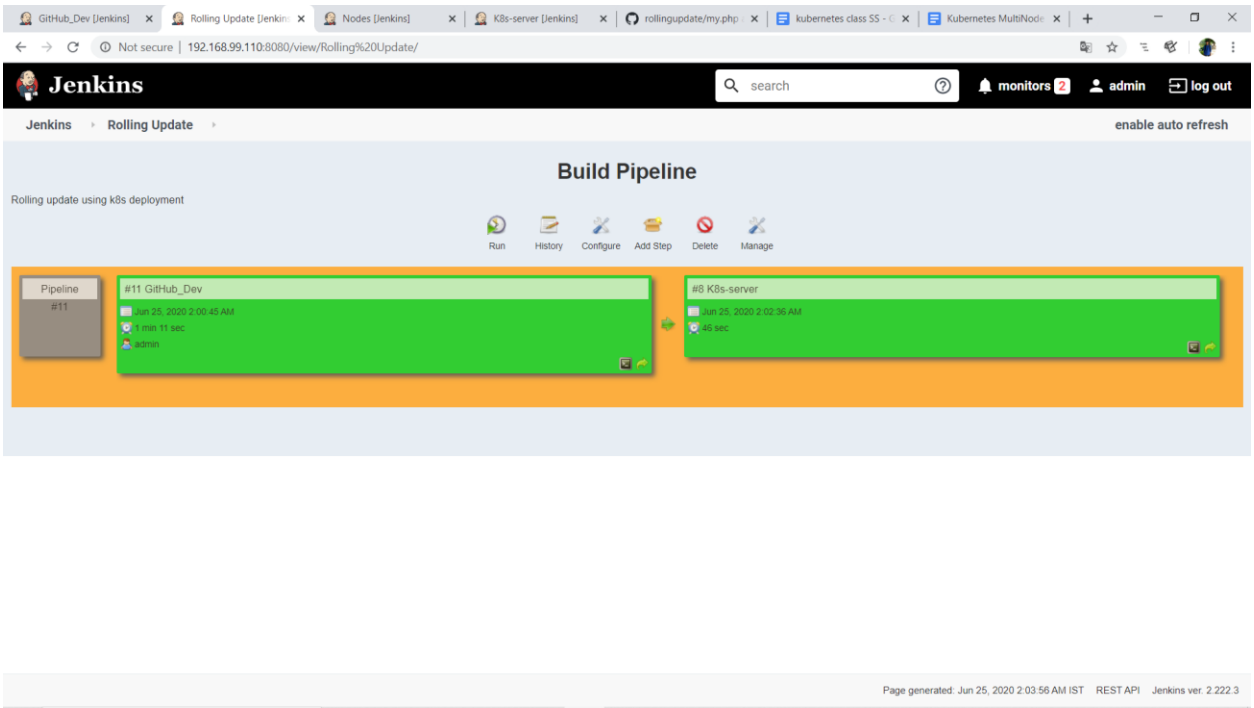
```
C:\Users\Anuddeeph Nalla>curl 192.168.99.100:31025

welcome to anuddeeph testing webserver
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.8  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:08  txqueuelen 0  (Ethernet)
        RX packets 11  bytes 654 (654.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 378 (378.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

              total        used        free      shared  buff/cache   available
Mem:           2136         839         102         510        1194         681
Swap:             0           0           0
```

Now let us see what it does when developer push any code to GitHub again.





Finally, my Build Pipeline



So here its rollout the update without any downtime. 😍

Feel free to give any suggestions...

Give a thumbs up if you find it useful...

You can directly message me if you have any problem in code...🙏

Thankyou...