**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...

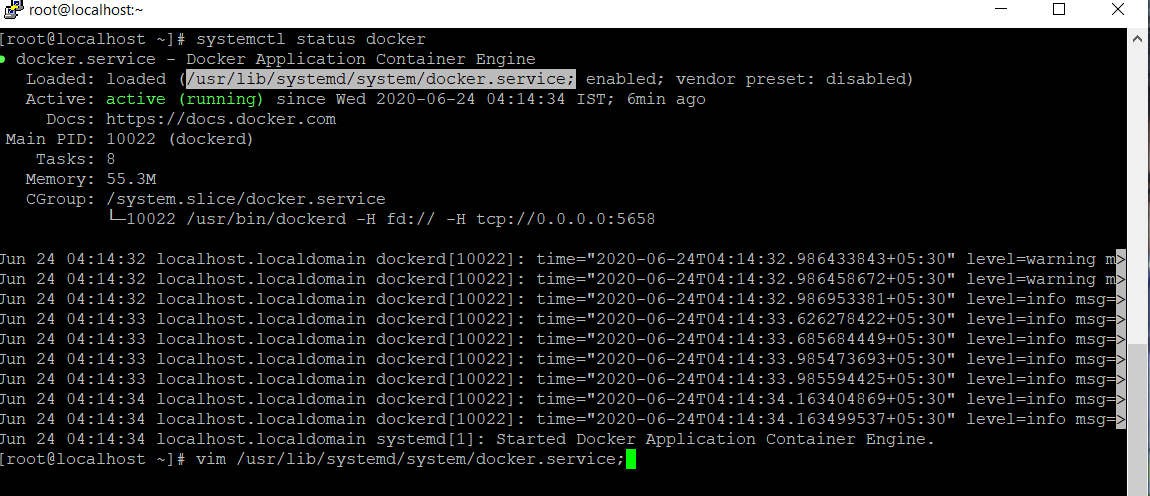
**Static Cluster:**For creating a static cluster, we need active permanent resources so that when you run your job in Jenkins, they run the job without any delay. But here is one drawback you can see that there is pure wastage of resources since machines are running continuously with no use until required.

**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.

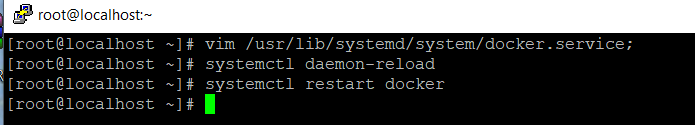
Here I implemented the dynamic cluster for completing the project...

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...



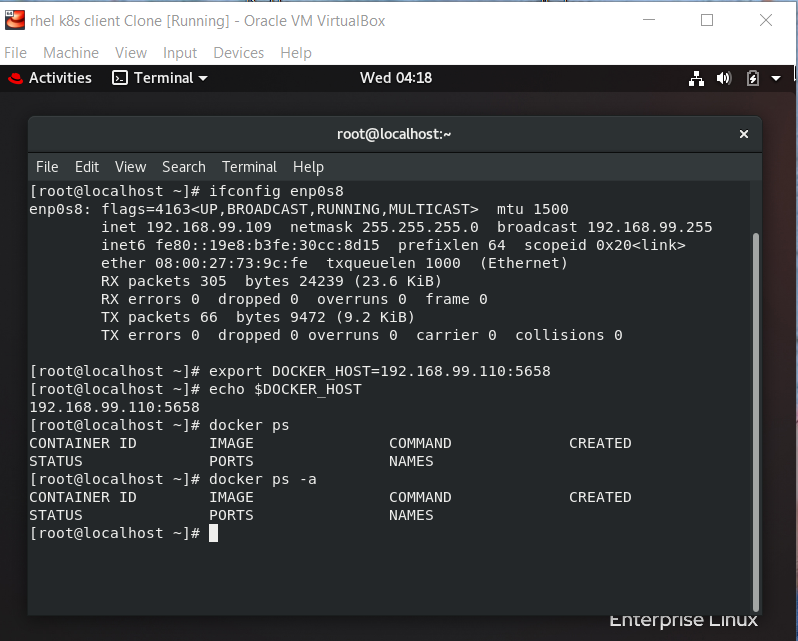




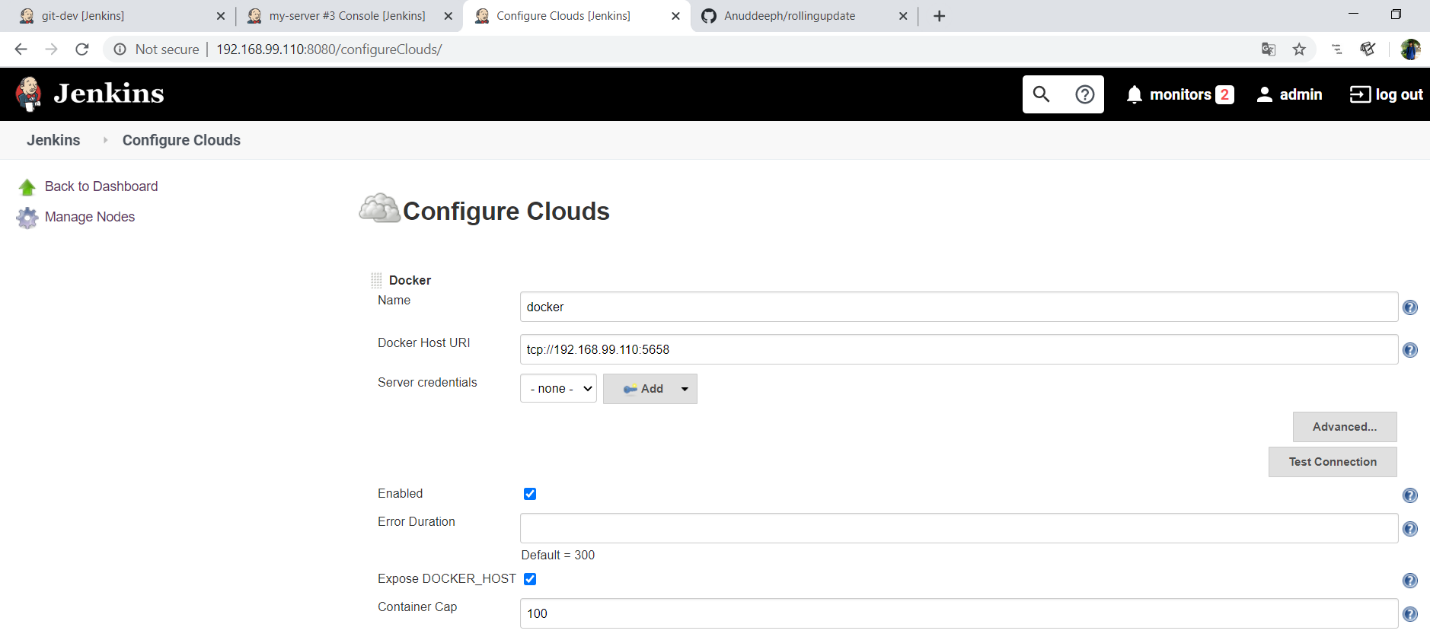
You must do these three steps and your docker service is ready for connect client from another VM.

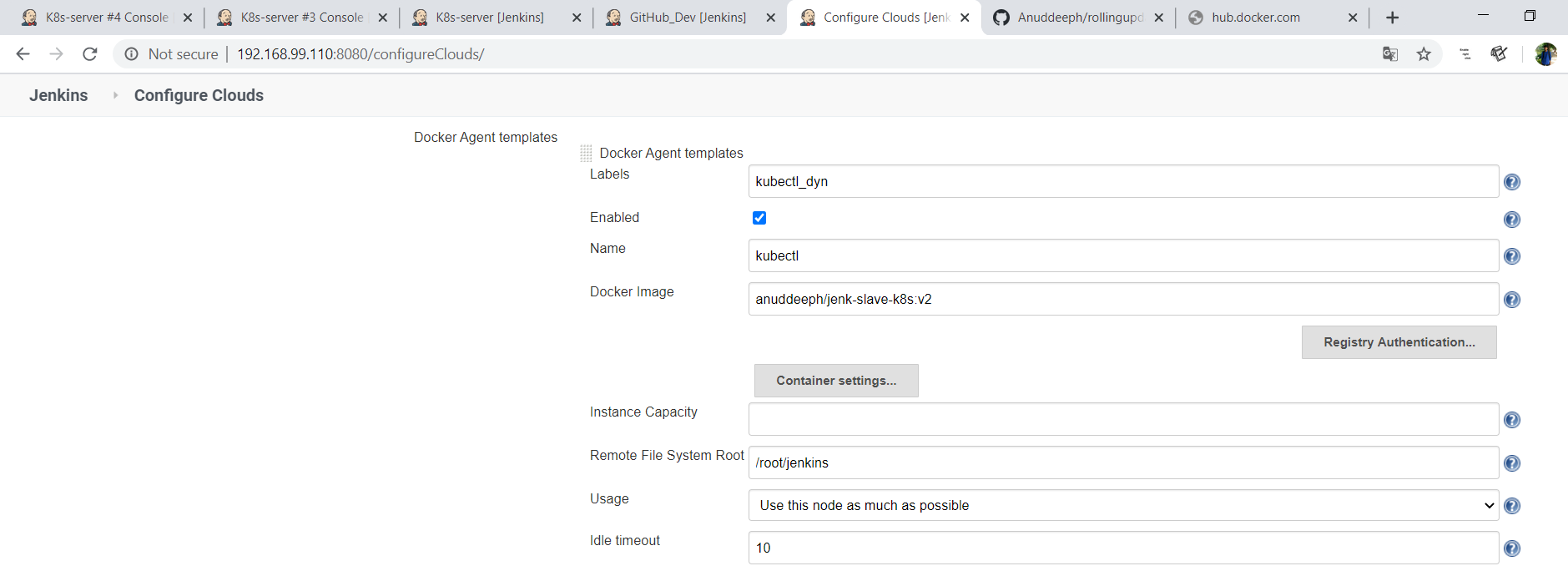
* Now come to another VM and stop the docker services using **systemctl stop docker**. After this you need to export the DOCKER\_HOST, so that you can use this VM as a docker client.

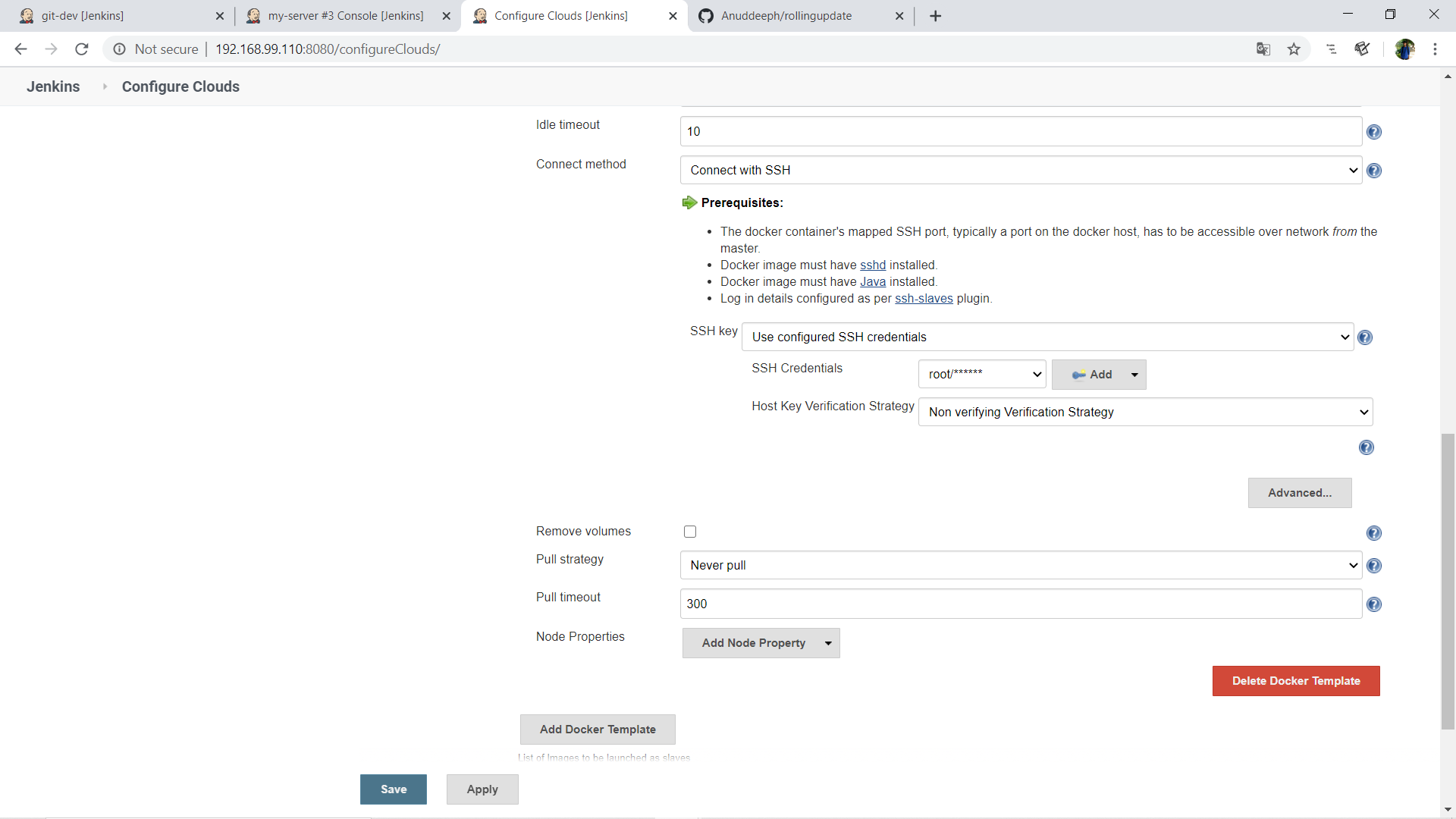
export DOCKER\_HOST=IP\_of\_server:port\_you\_give\_in\_docker\_service\_file



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





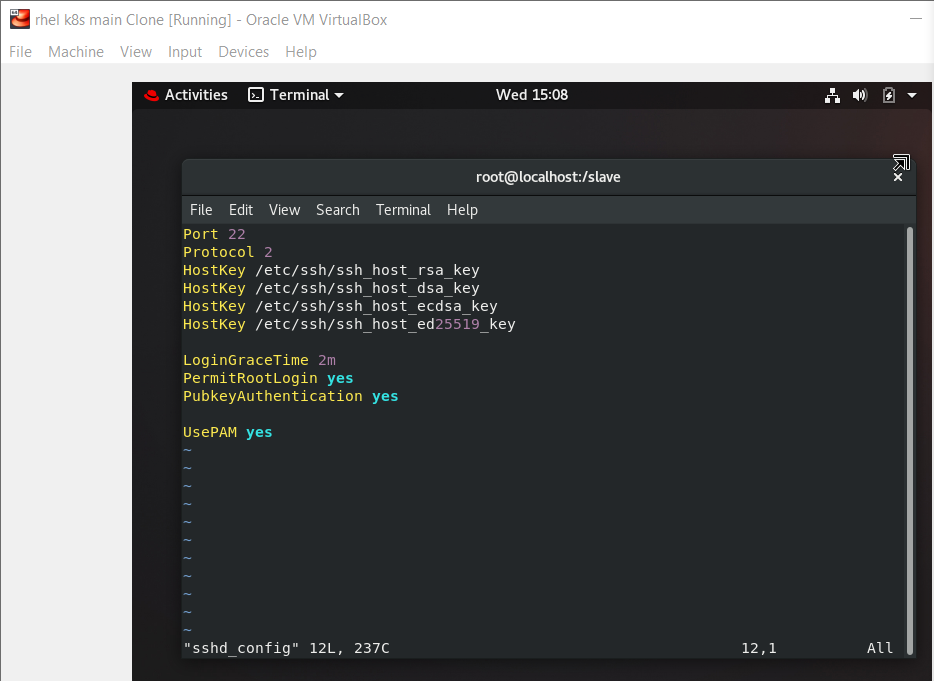


Now my docker cloud is configured. Now let us come to task...

## 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

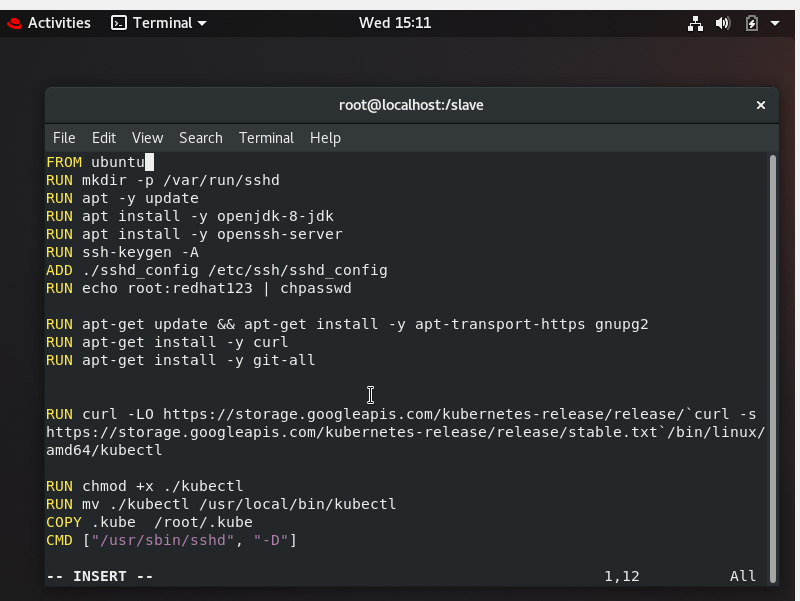
LoginGraceTime 2m

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

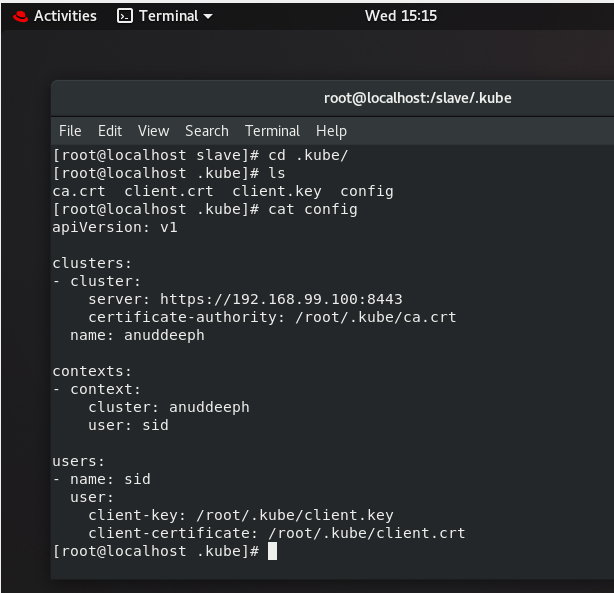
RUN chmod +x ./kubectl

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.



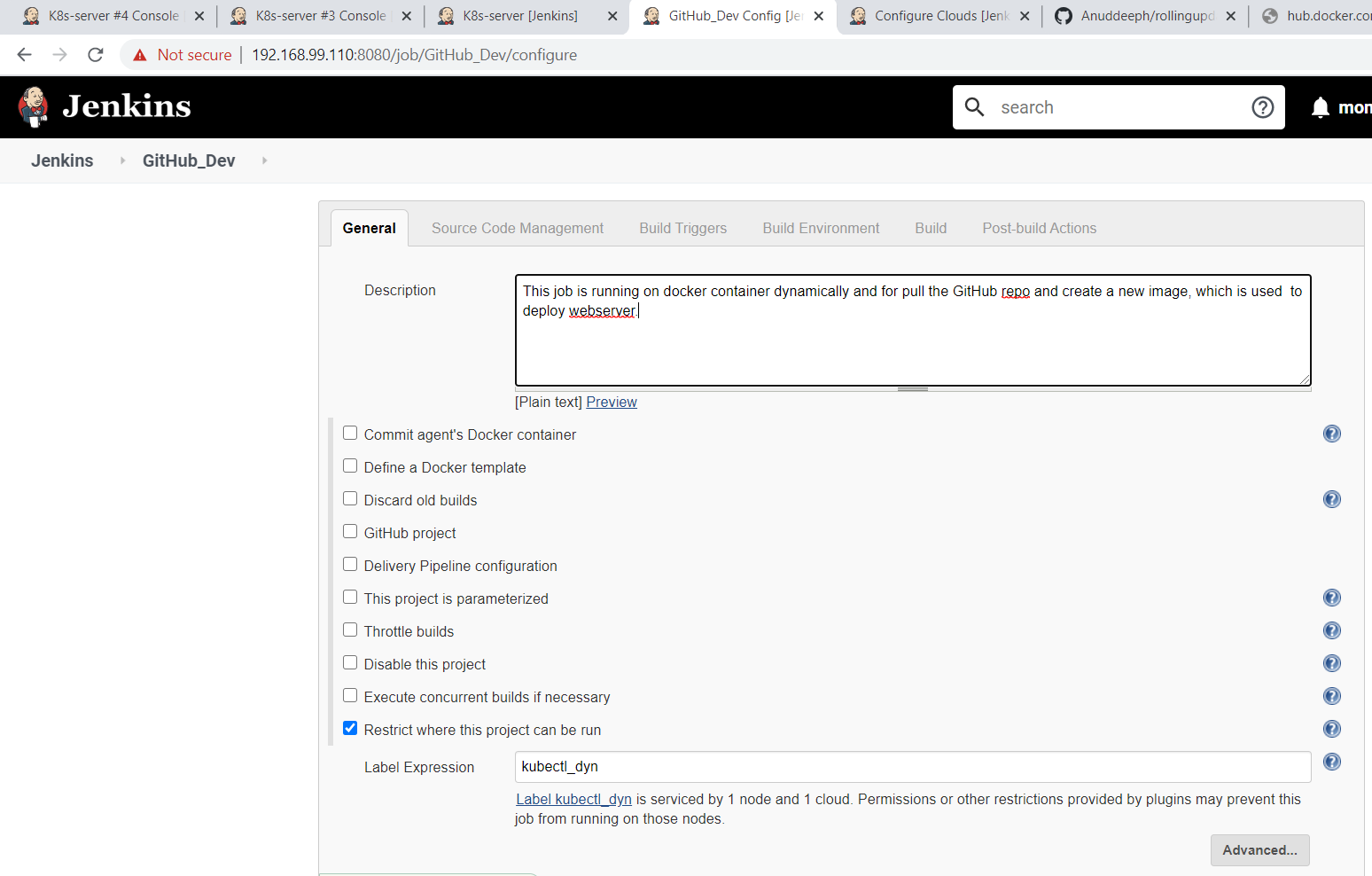
This is my Dockerfile for creating an image that runs on Jenkins as cloud node. Here I also copy kube\_config file to image so it run as a slave for my base K8s cluster.

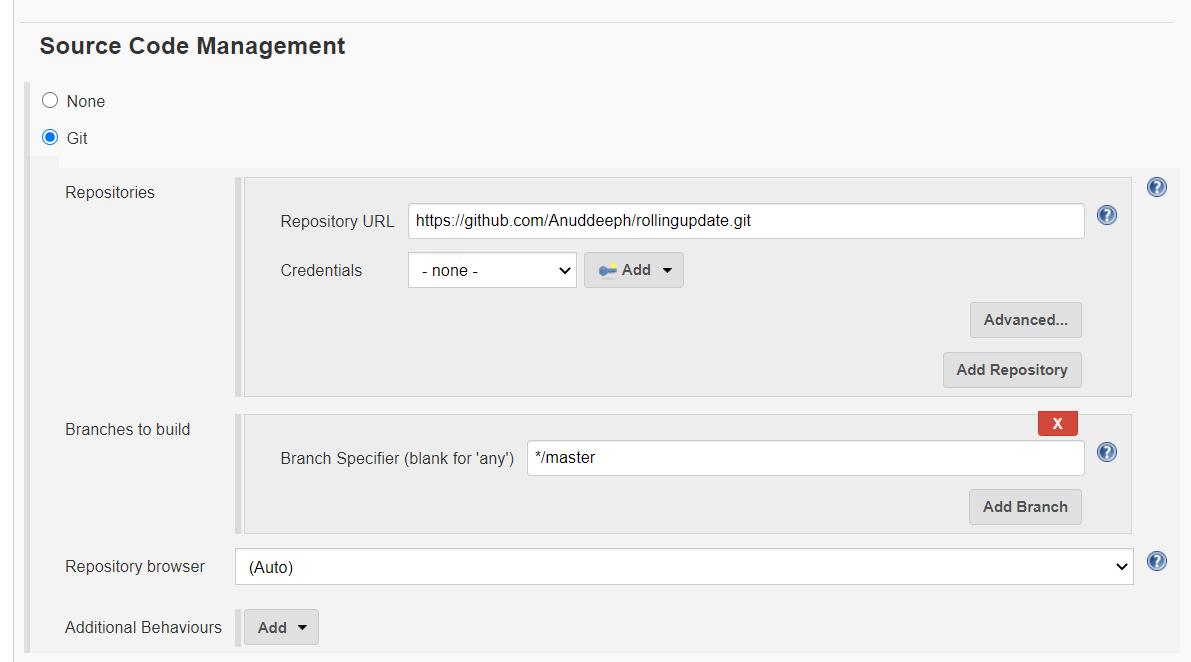
### 2. Jobs in Jenkins:

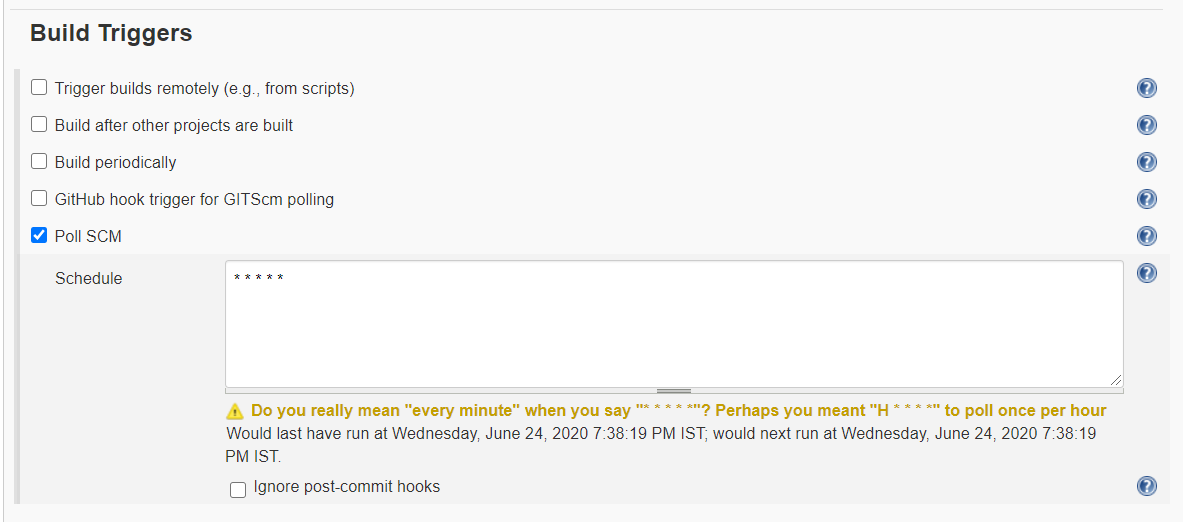
* **Job1: (Copy code and Dockerfile from GitHub and create an image using Dockerfile and push it in to docker hub.)**

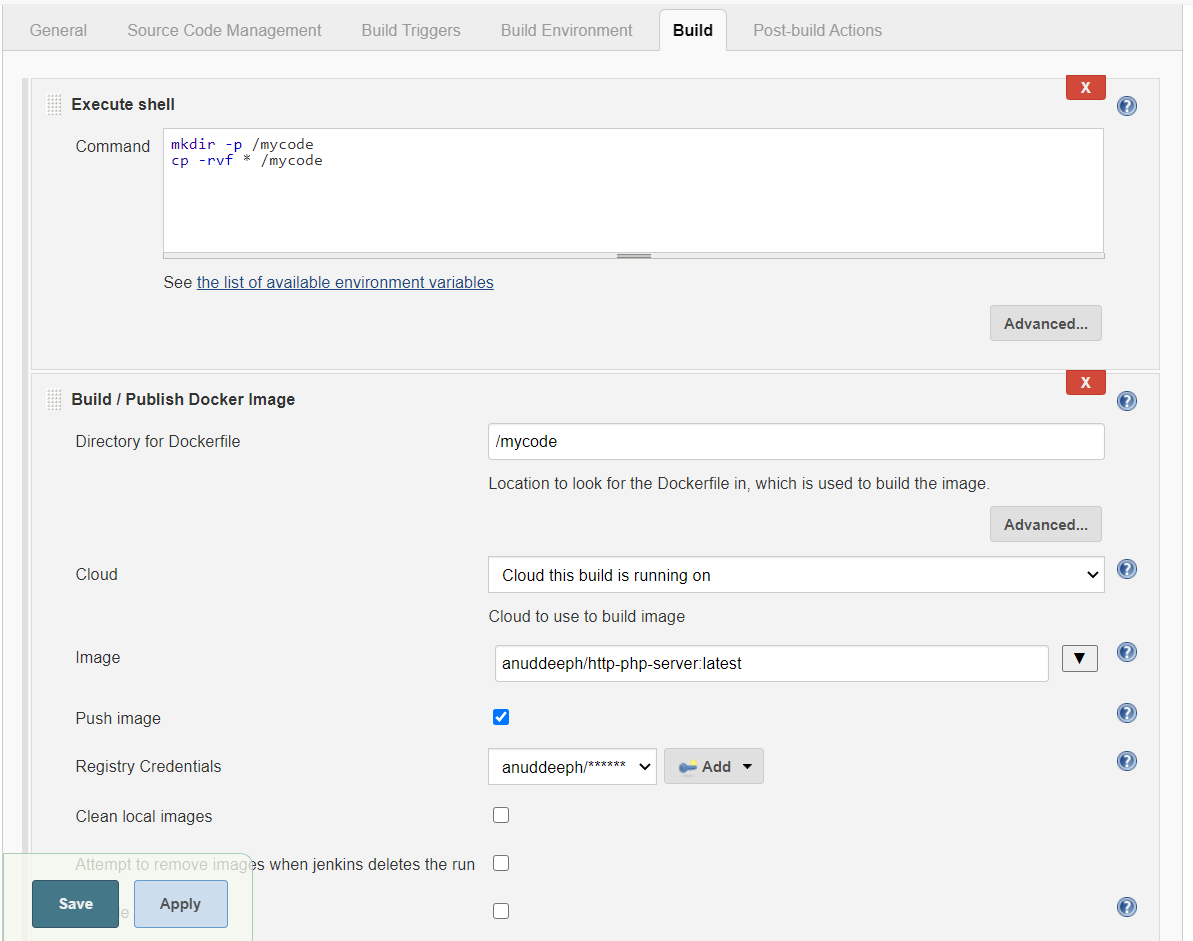
This is the link of my GitHub Repo:

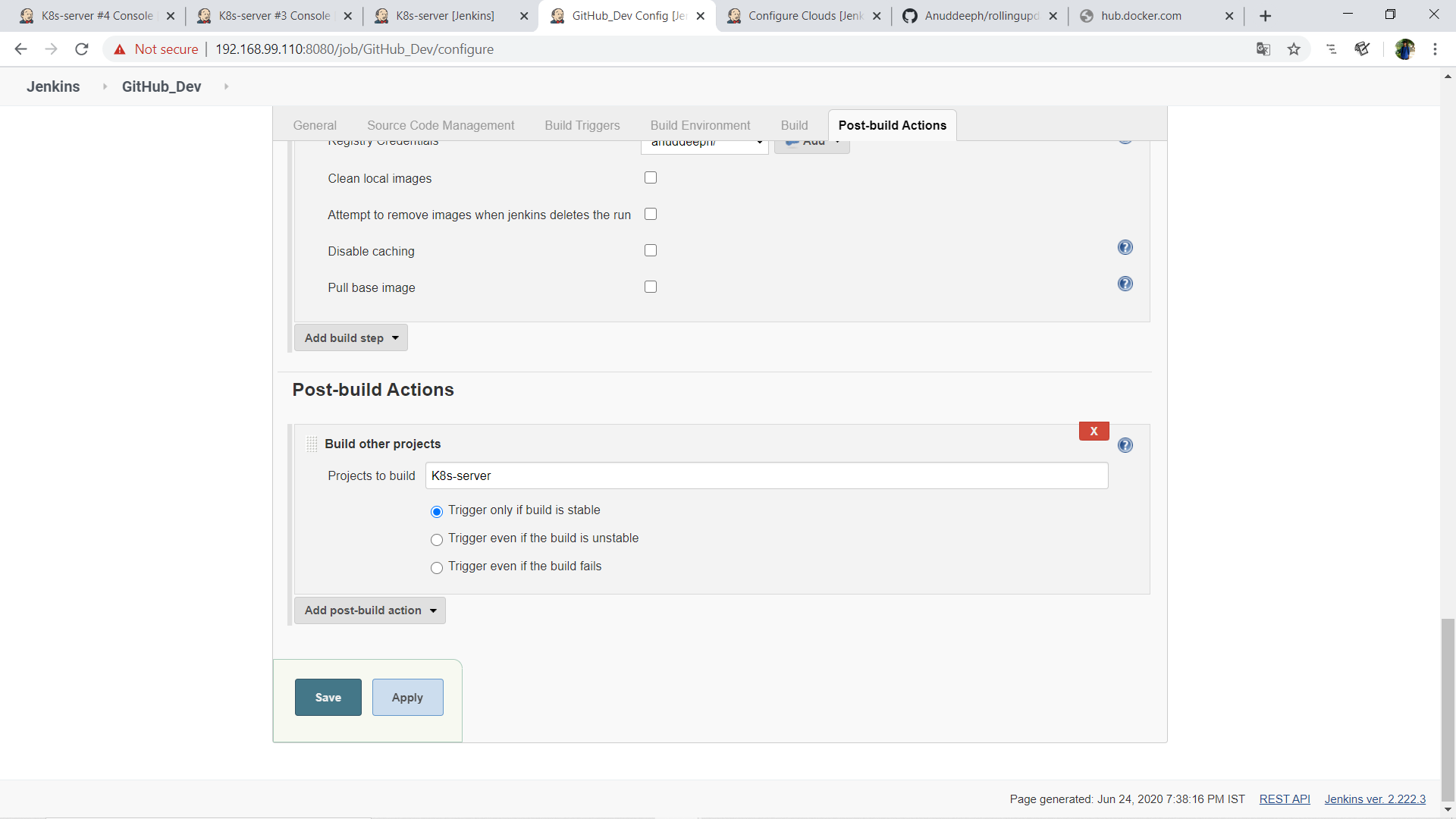
<https://github.com/Anuddeeph/rollingupdate.git>





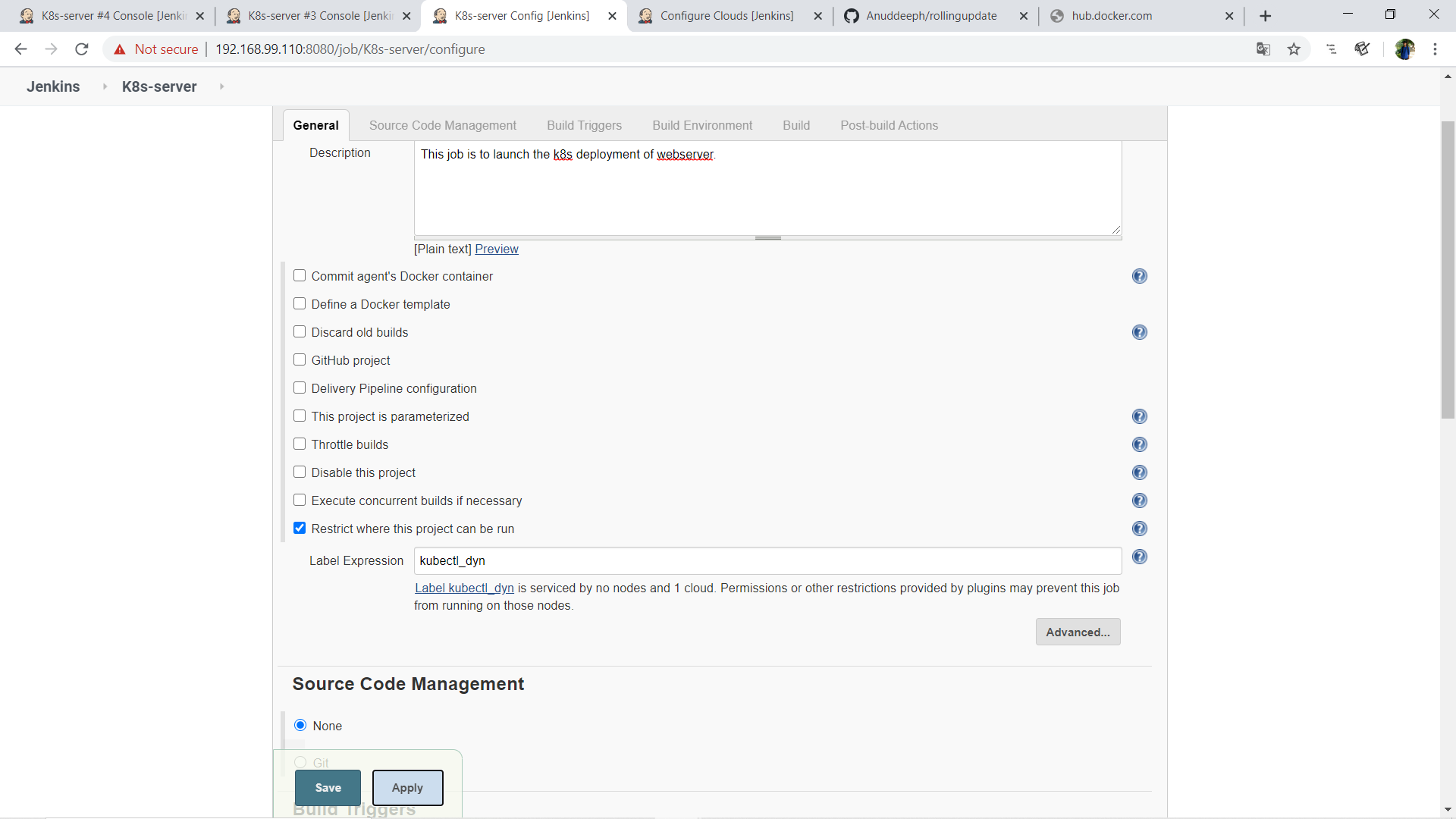


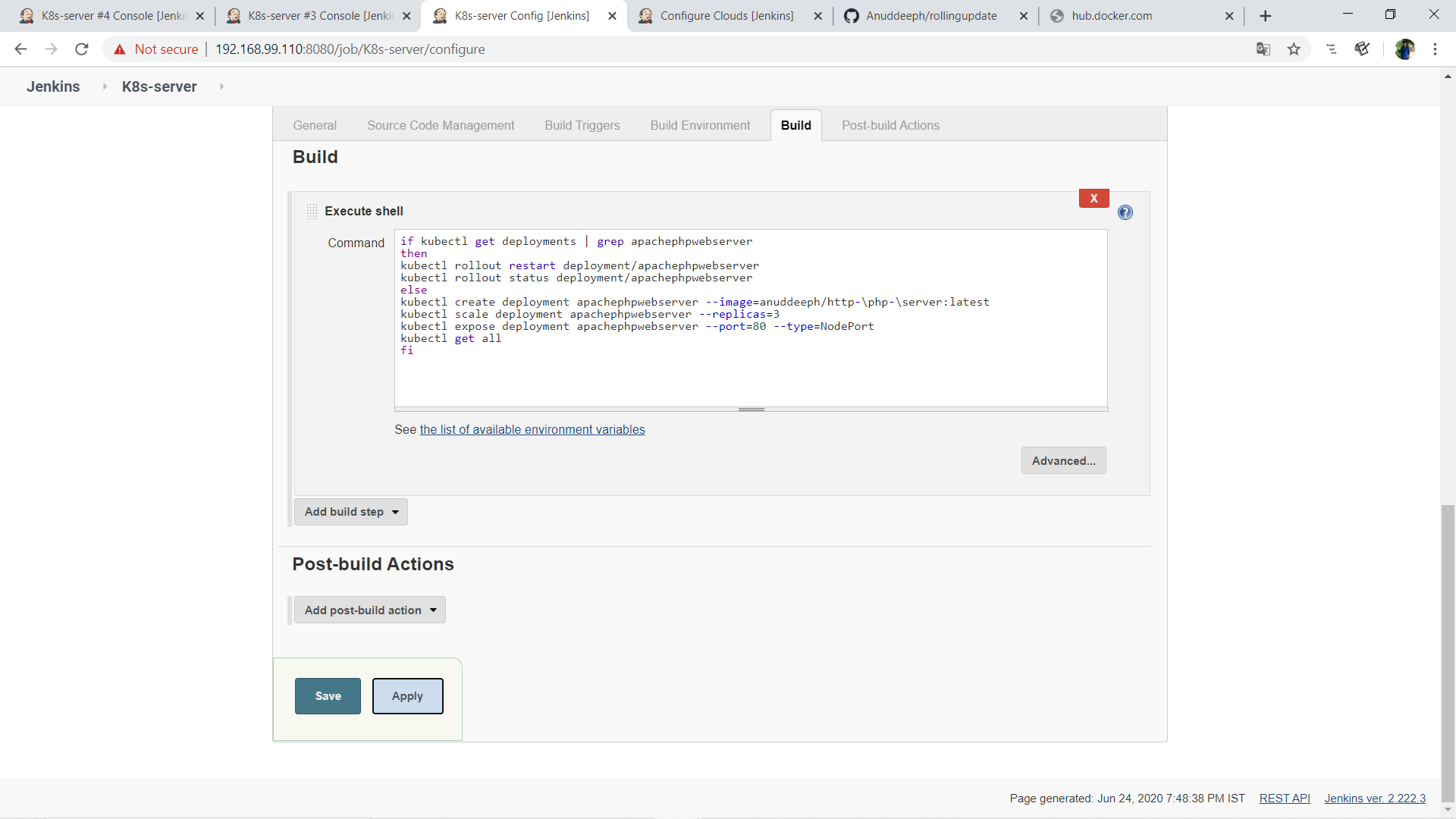




Now come to next job...

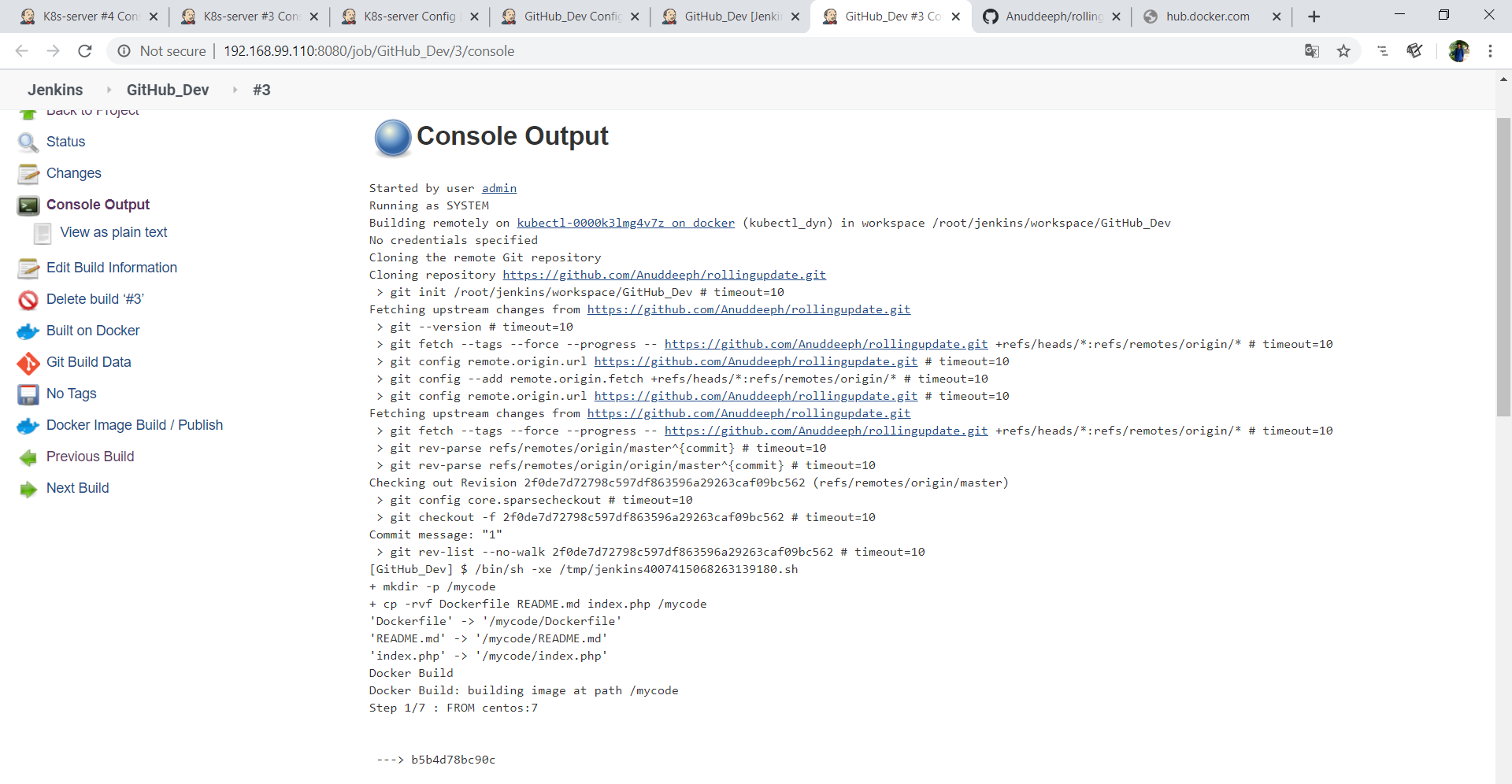
* **Job2: (Launch the application deployed on top of Kubernetes cluster...)**
* Before building this job start minikube using command minikube start.

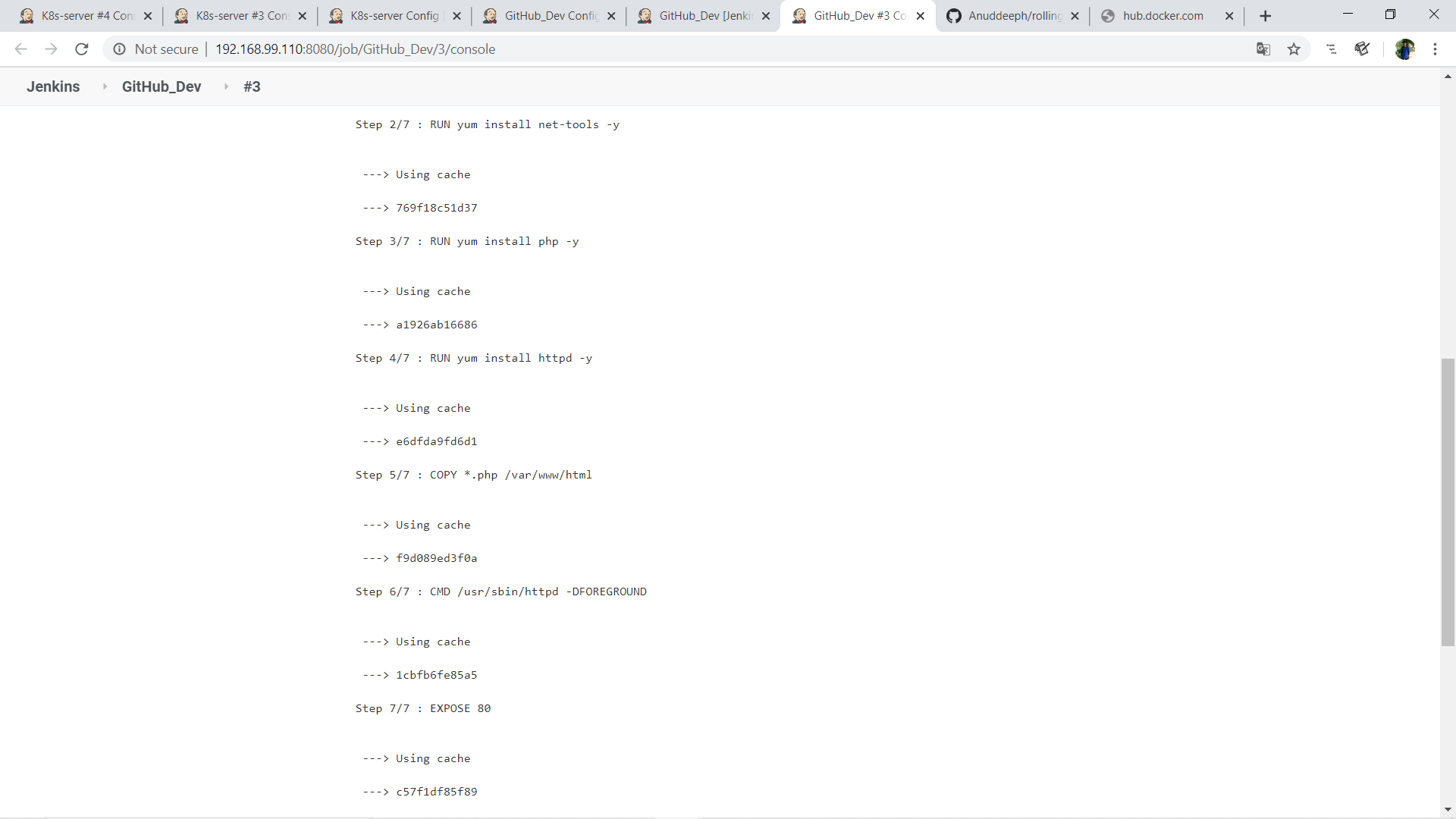


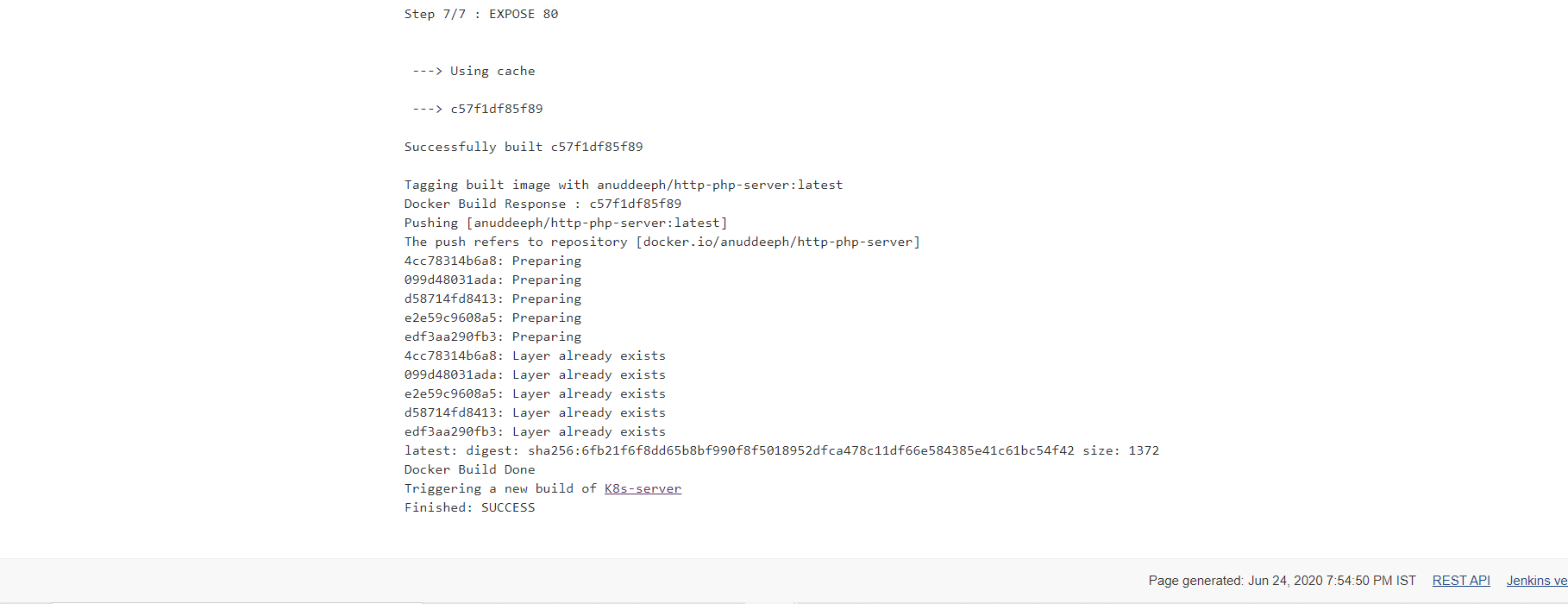


This code is deployed our web-services and when developer push any code again then its rollout the update without any downtime. This is my [docker-hub](https://hub.docker.com/r/anuddeeph/http-php-webserver). where job1 automatically uploaded the docker-image build by Dockerfile.

**Now let us run our job1 and see the output:**

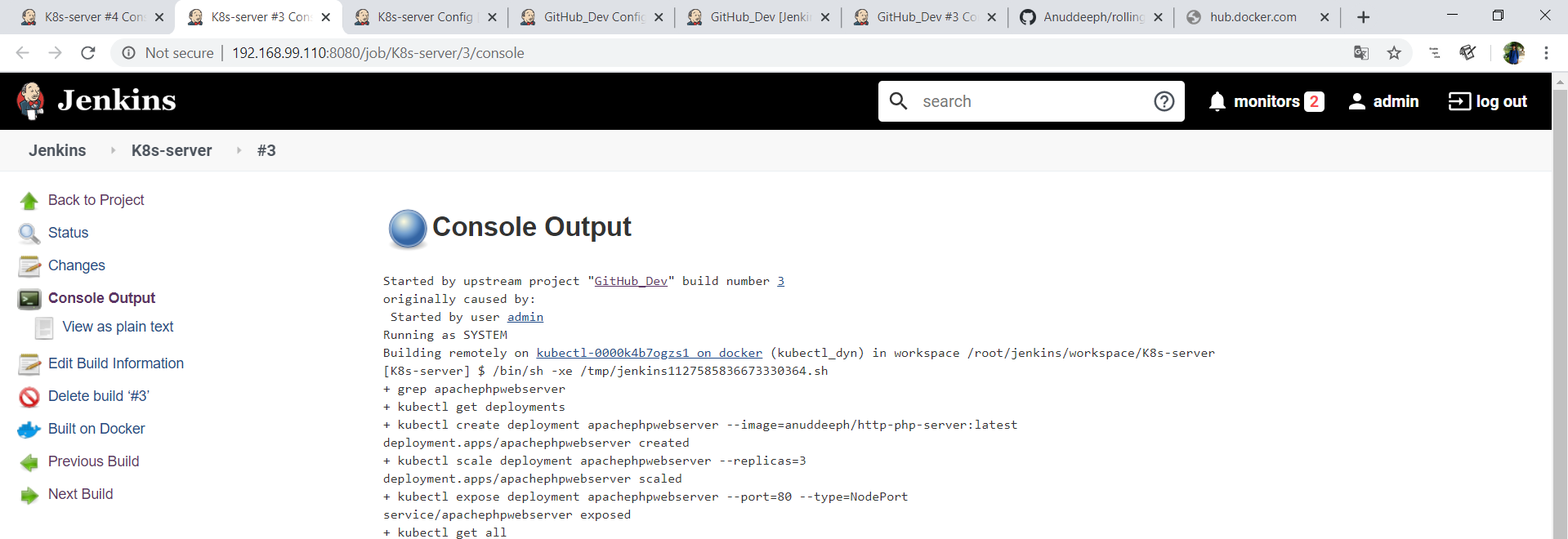


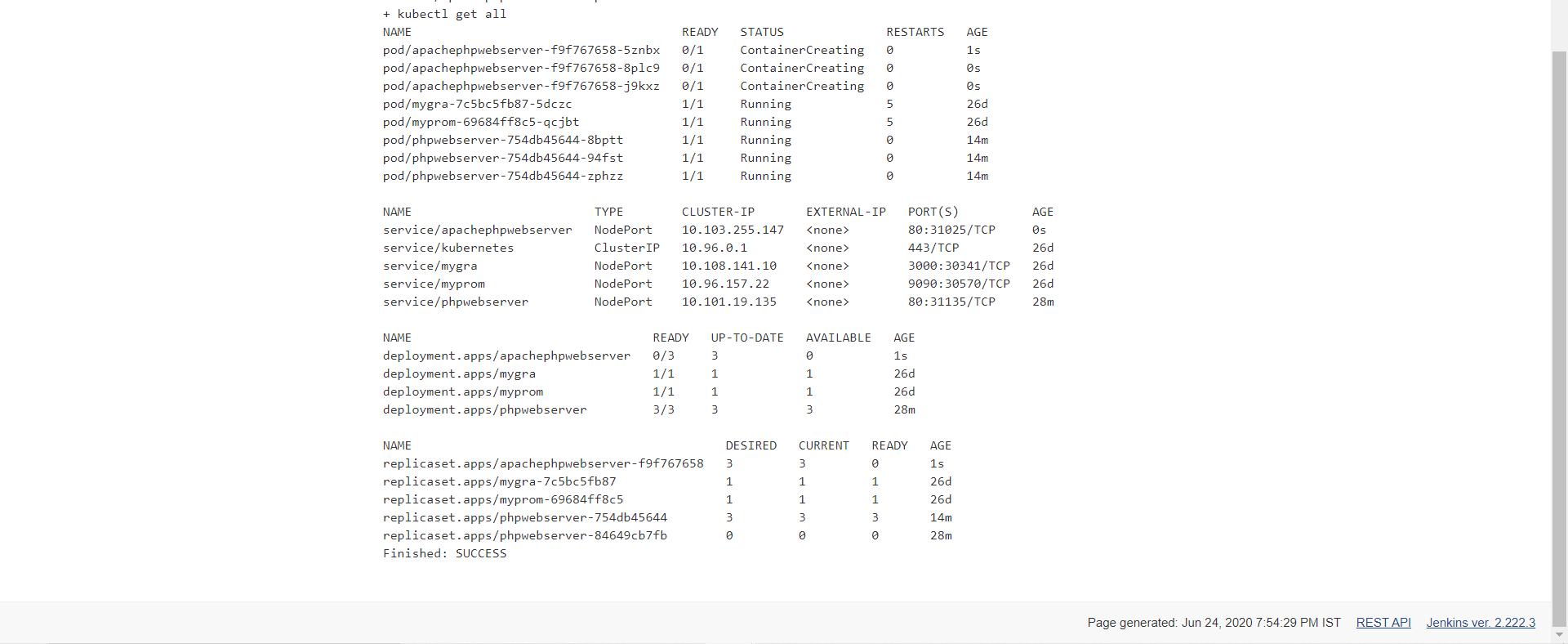


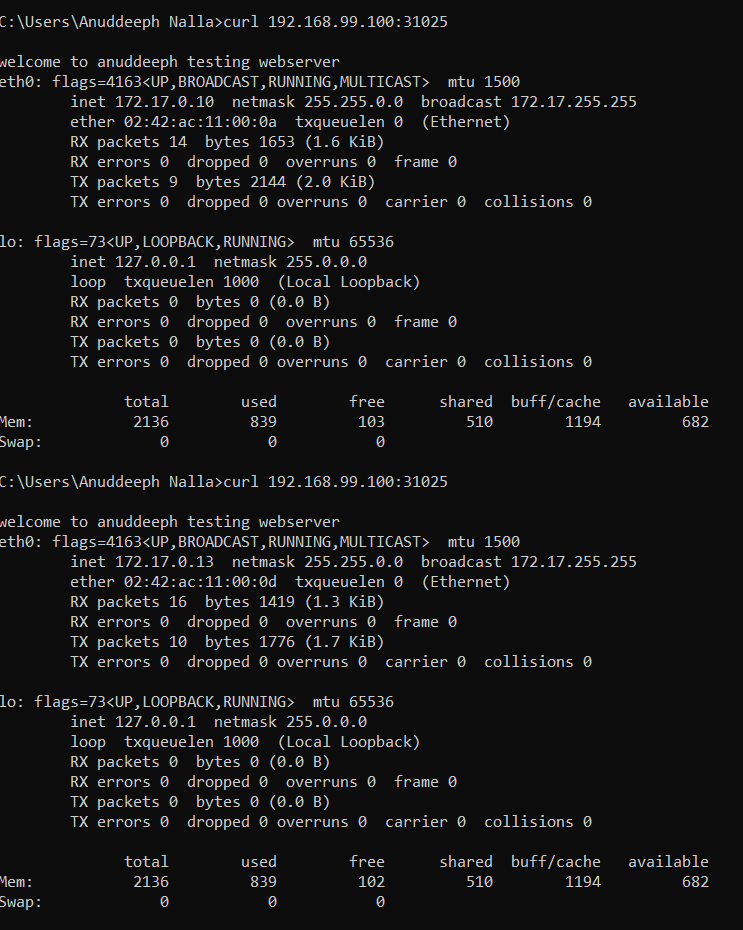


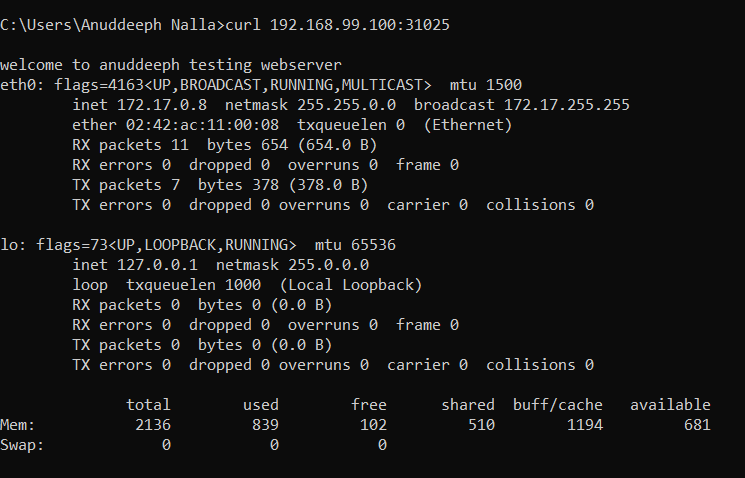
Now my Docker image is created and push to [docker-hub](https://hub.docker.com/r/anuddeeph/http-php-webserver).

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

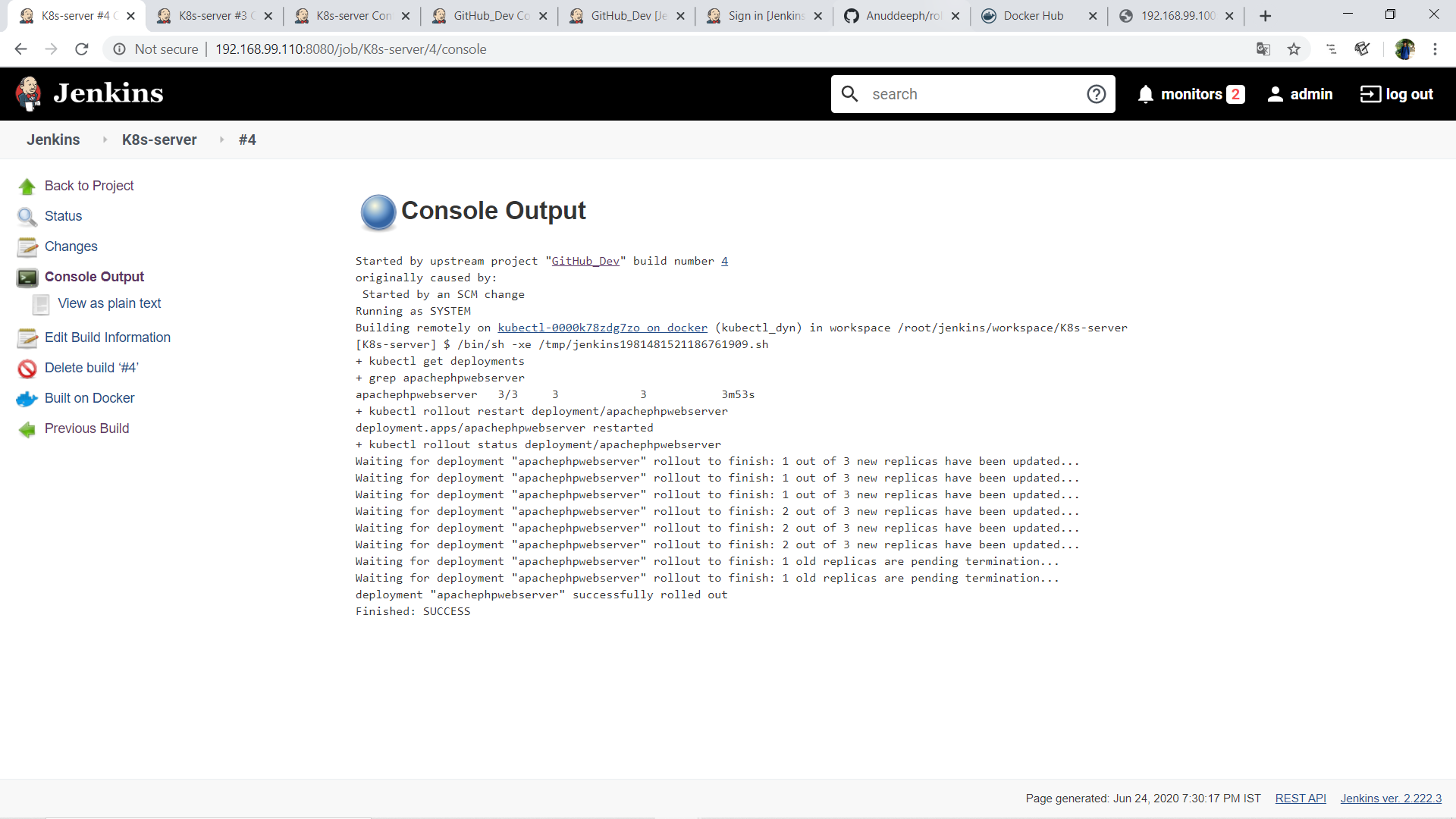


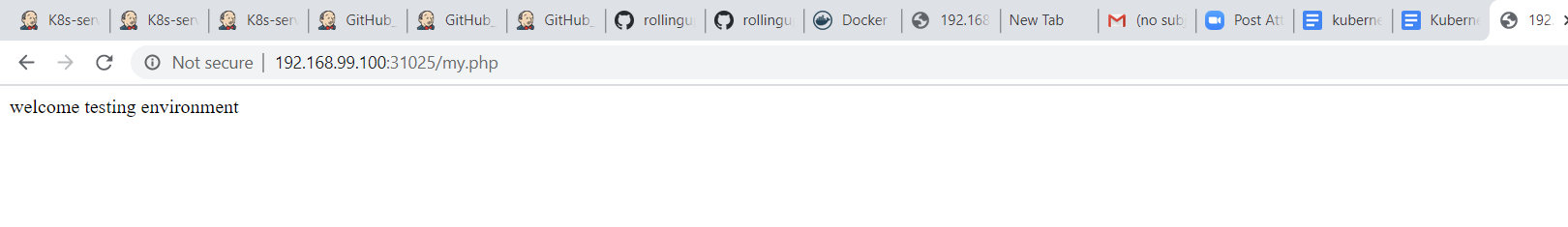




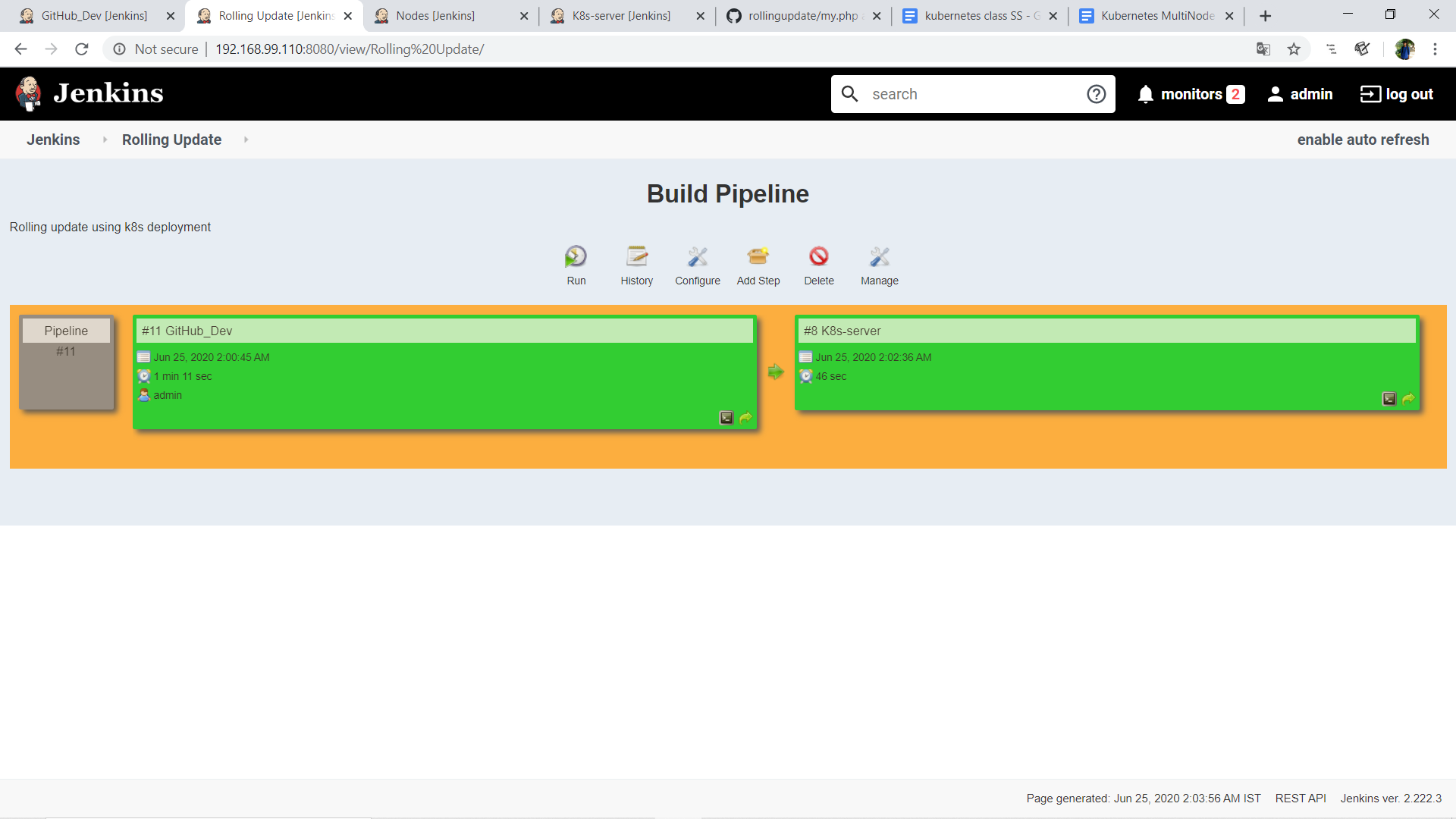


Now let us see what it does when developer push any code to GitHub again.





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...