Progress

Date :- 04-05-2022

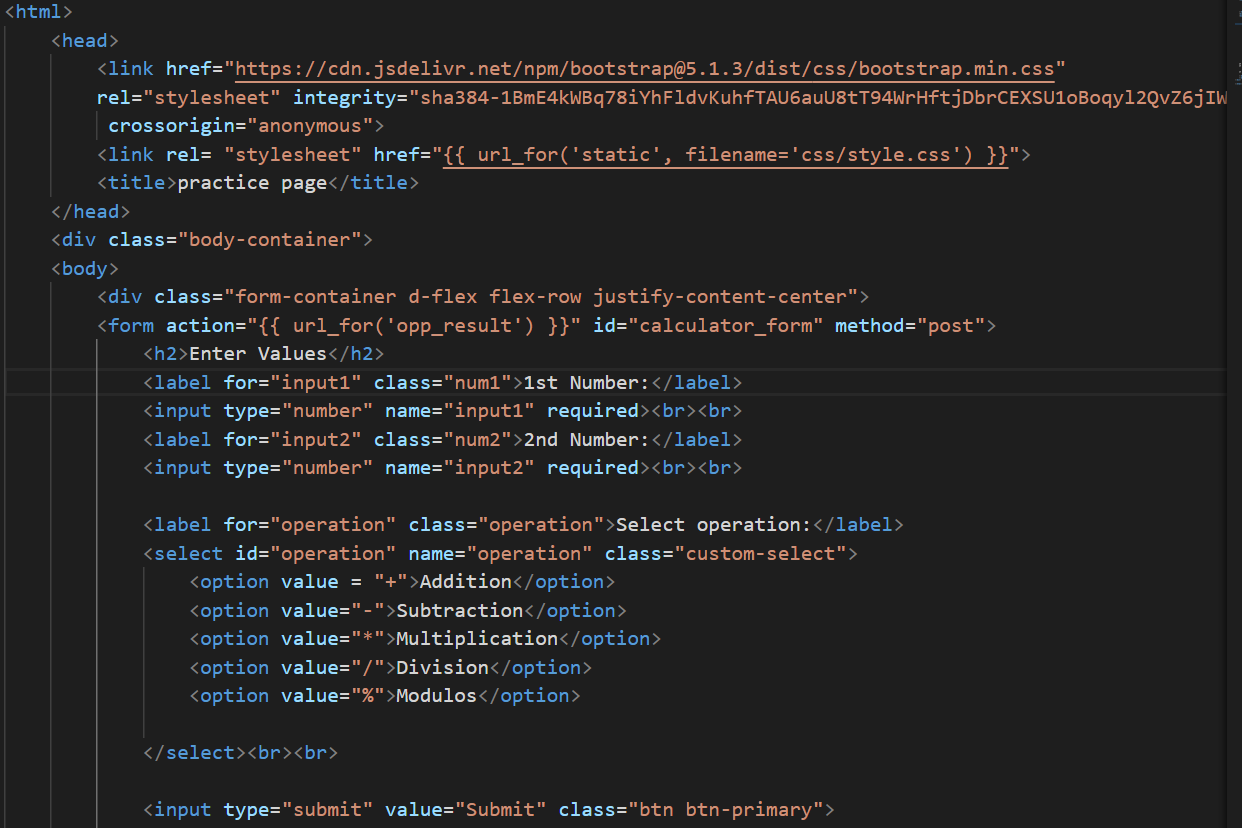
**Task1** :- Calculator using python and flask

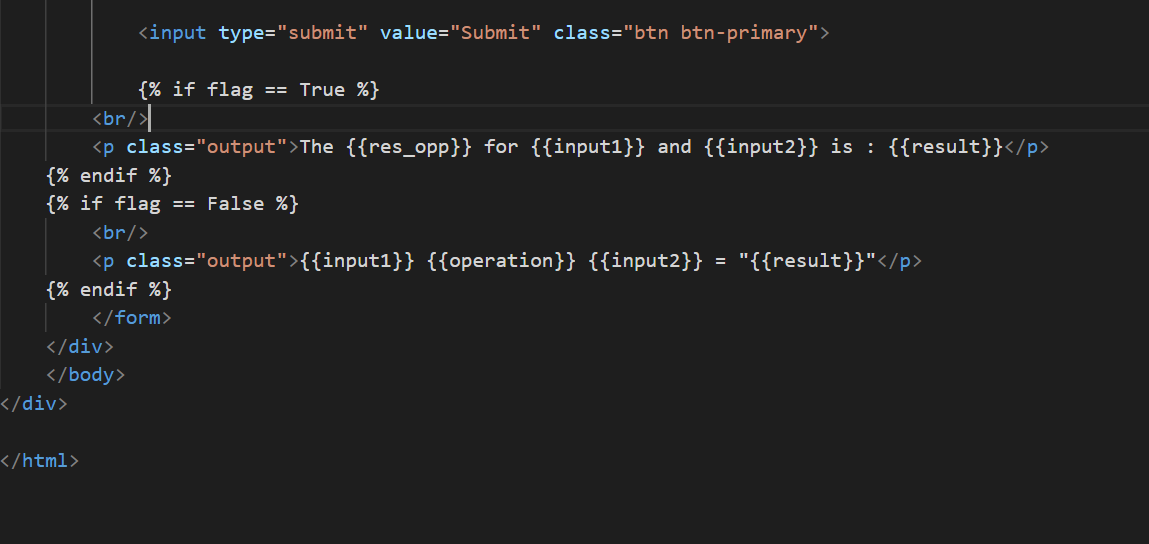
Python code for calculator:



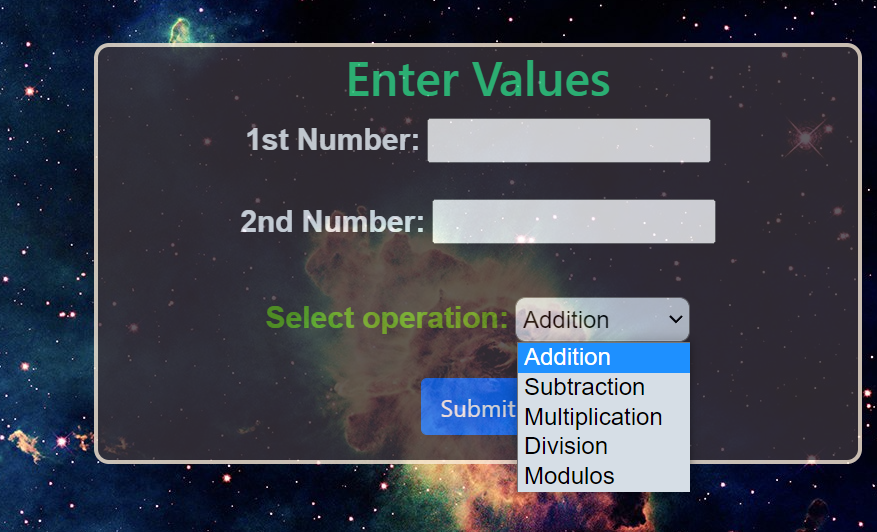


Html Code :-



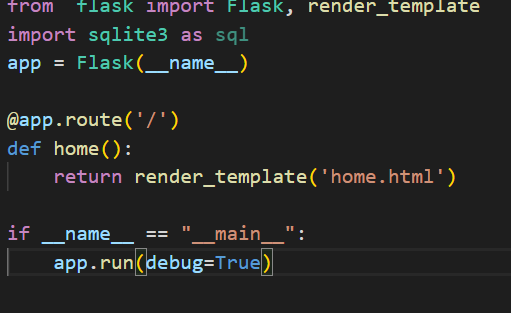


Output :-

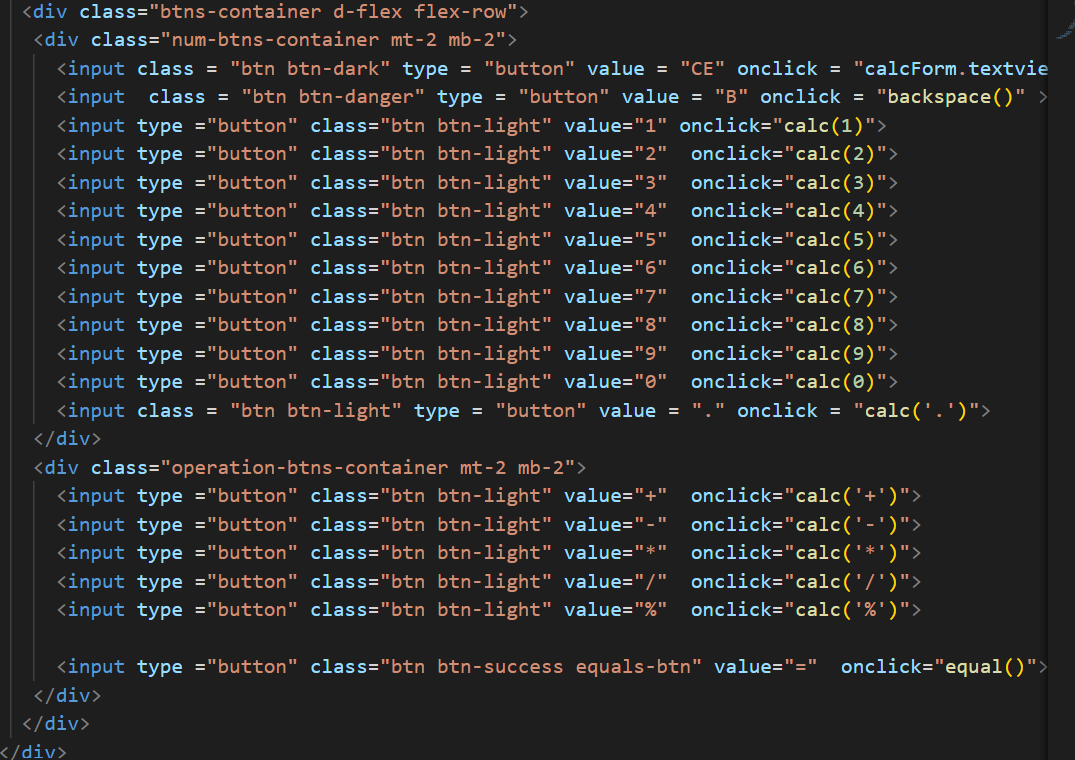


**Calculator using js :-**

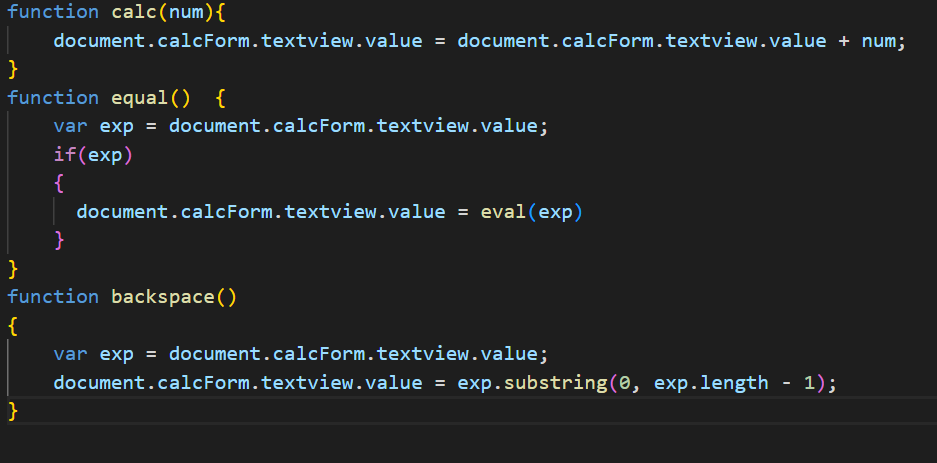
**Script:-**



**Html:-**



**Javascript:**

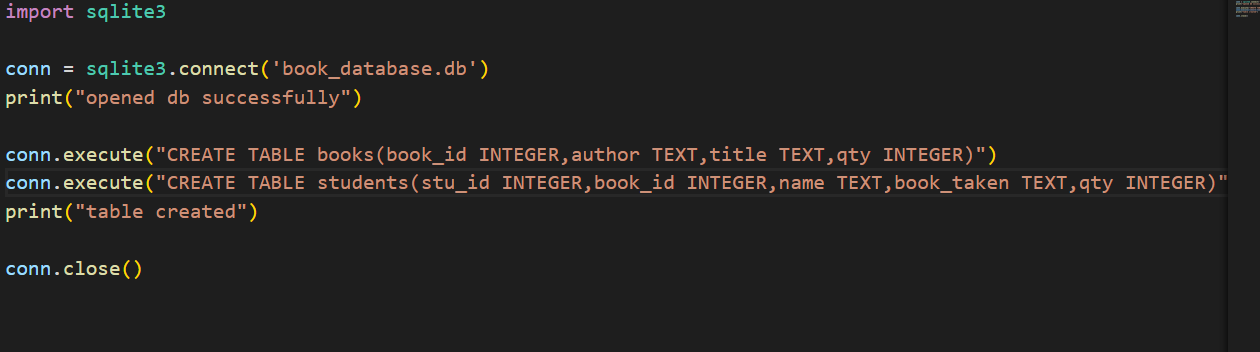


**Output:-**

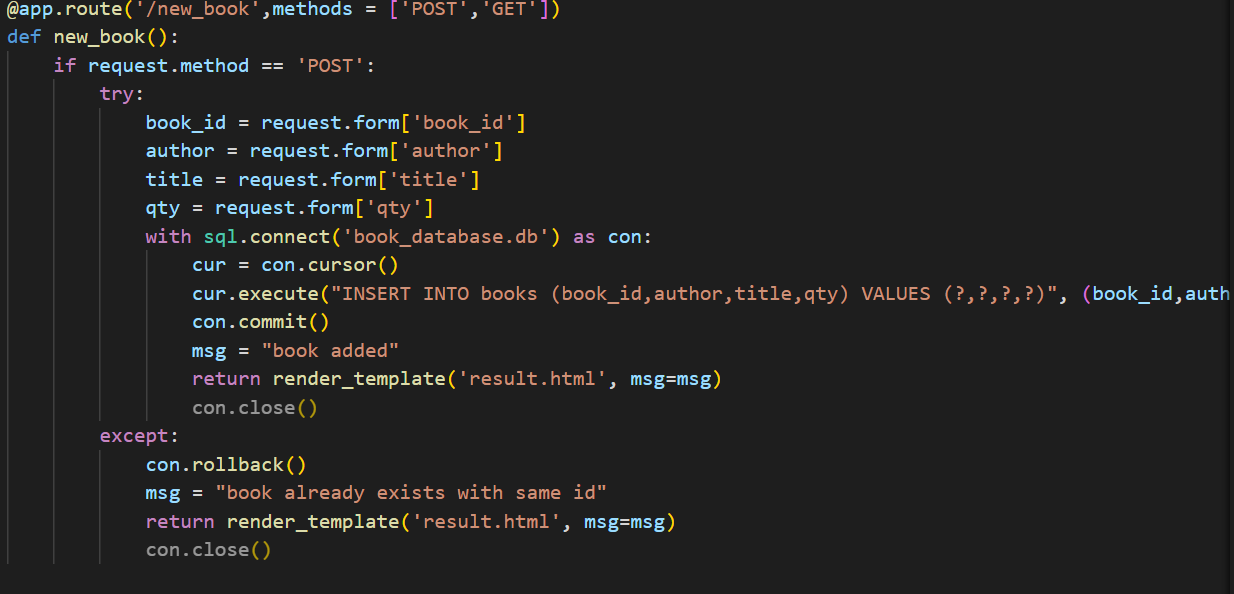


**Task2:- Book Inventory**

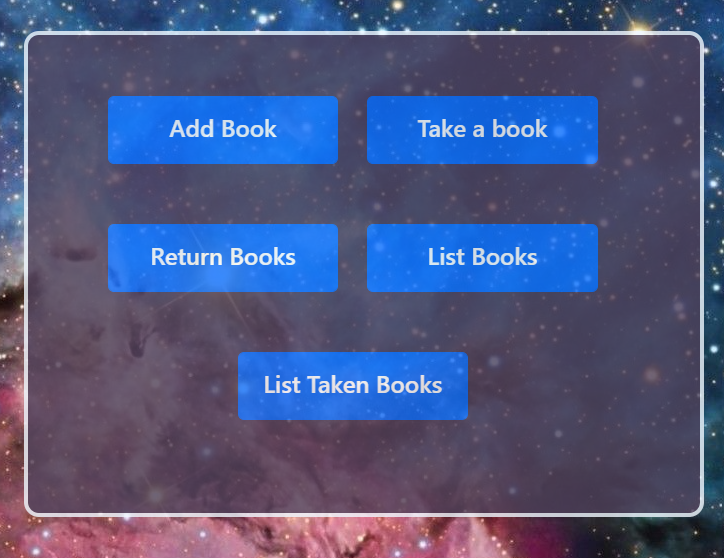
**Script for database creation and tables creation**



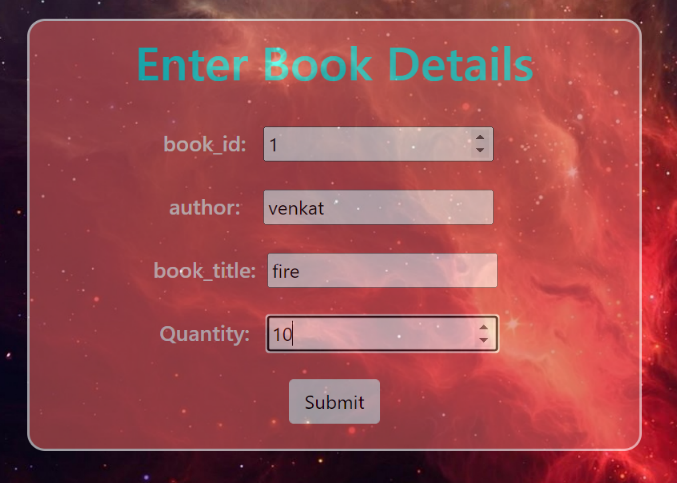
**Script sample:-**



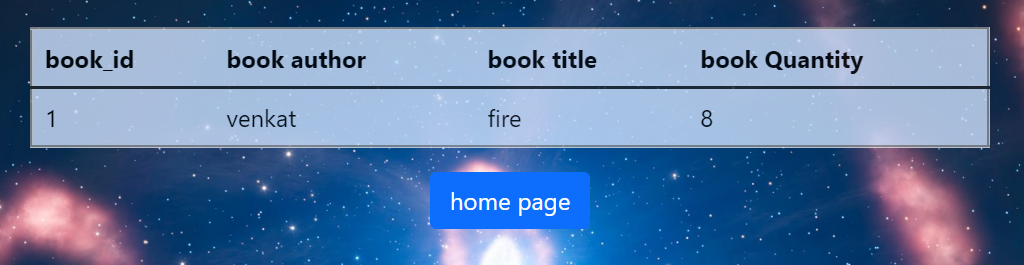
**Output:-**



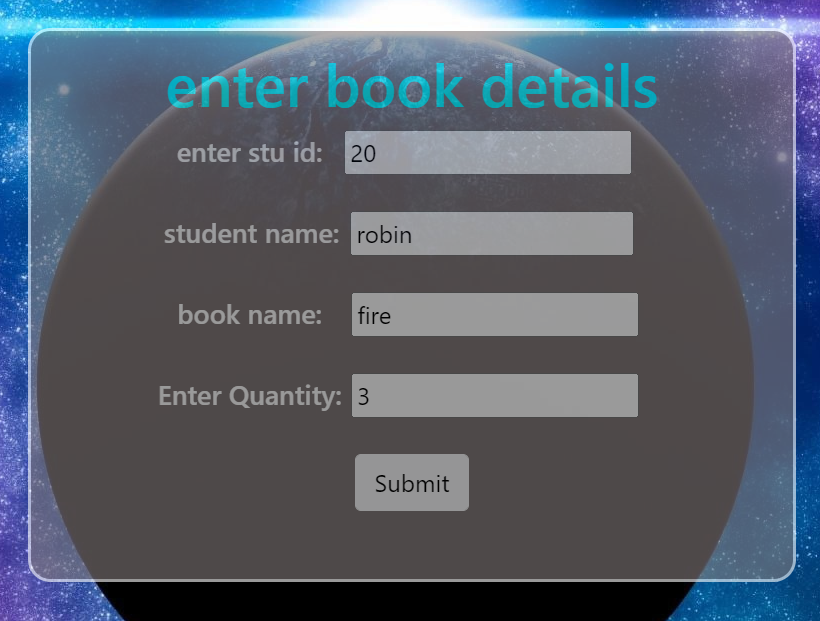
**Taking books:**



**List books:**



**take books:**



**From Date :- 16-05-2022**

**Task1:-**

**Docker & Containerizing our app using docker:**

Existing Problem :

It works on my machine.

Dockers are used to build containers which contains everything like (code,dependencies,configuration,process,os(chunks),networking etc)

We can bulid our application in the guest os with all necessary dependencies and all…

Old solution:



But every new application we need to install new Guest os and build our app in them so it takes burden to vms.



So we have to replacing the hypervisor…we can just install docker on top of os and build aour applications…



Docker\_basic\_cmds:

**1.Docker run hello-world**

It will run the hello-world image from docker hub…and prints hello world.

**2.docker images**

Shows images

**3.docker pull ubuntu**

It will pull docker image from hub

**4.docker run -it -d ubuntu**

It will run the ubuntu image as a container

**5.docker ps -a**

It will shows running containers

**6.docker exec -it container\_id bash**

It will access a running container…we will go within the container

**7.docker stop cont\_id**

For stopping docker container

**8.docker commit container\_id account\_name/ image\_name**

It will commit the new image to the local repository

**9.docker login**

For logging in to docker hub

**10.docker push dockerhub\_username/image\_name**

Pushes image to local docker repository

**11.docker rm container\_id**

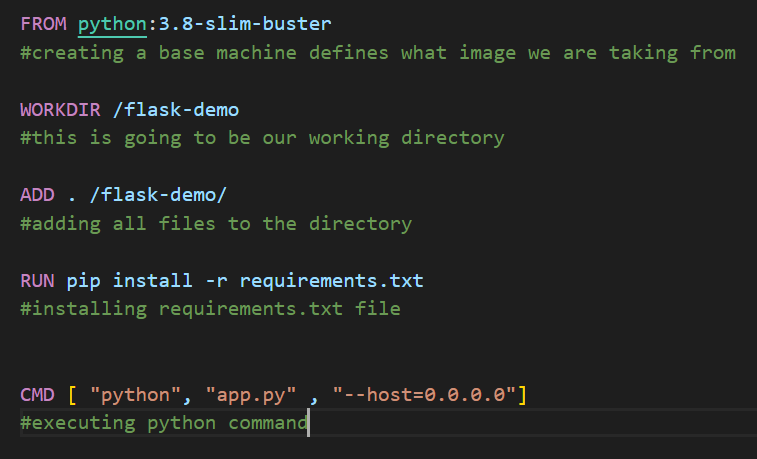
For deleting container

**12.docker rmi image\_id**

Deleting docker images

**Docker file is blue print for building images, image is a template for running container and a container is actual process.**

**calc\_app Dockerfile:**



Building image with tag:

**docker build --tag flask-demo:1.0 .**

running our image:

**🡪docker run -d -p 8080:5000 -t flask-demo:v1.0(without specifying name for container)**

**🡪docker run -p 8080:5000 -d --name v2\_container flask-demo:1.0(with container name)**

**d- detached mode of running container**

**p -port for host and container**

**dot(.) – similar to $pwd**

**🡪docker push venkatrobin/flask-demo:1.0**

**Pushing image to venkatrobin public repo with tag 1.0**

**Date :- 18-05-2022**

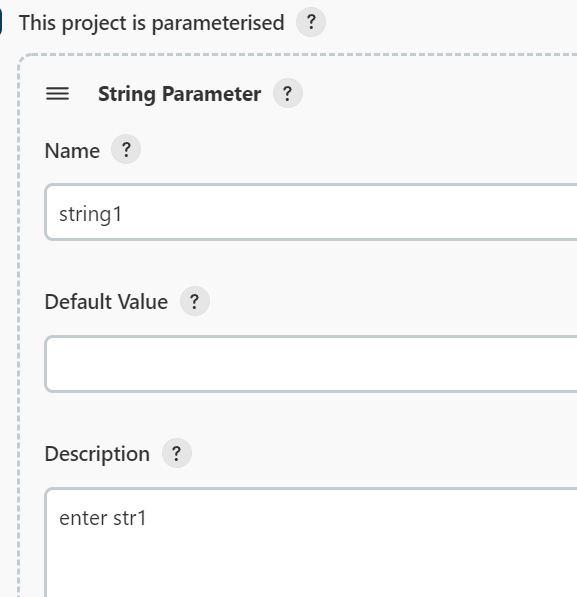
**Task 2:- install jenkins in docker desktop container**

**🡪docker run -p 8080:8080 -p 50000:50000 -d -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts**

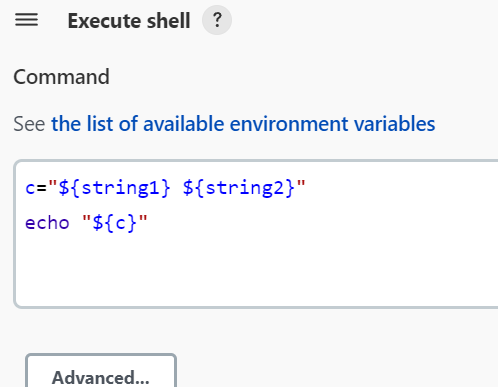
pulled Jenkins image and running on 8080 port without mounting docker.soock file

**Basic Jenkins Jobs:-**

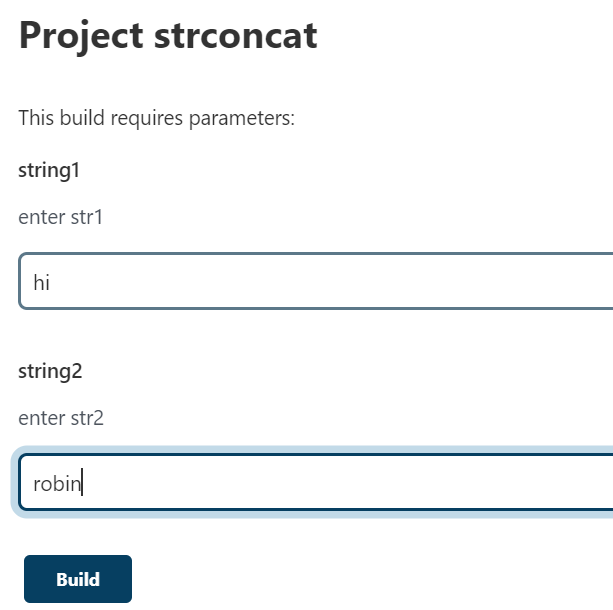
**Free style job for string concat(parameterized job):**



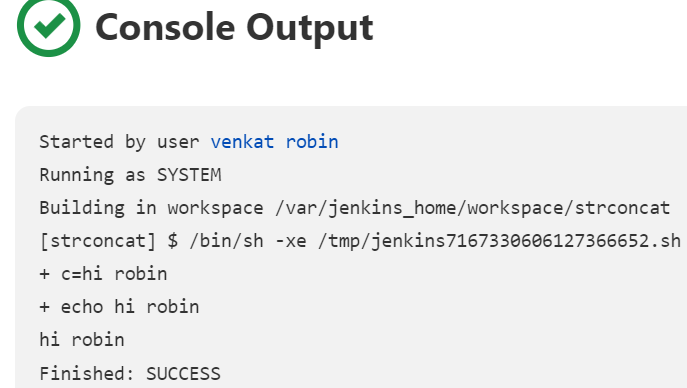
**Str concat shell cmd:**



**Building with parameters:-**



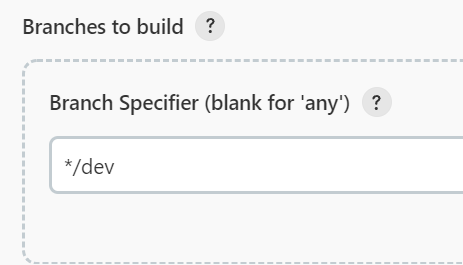
**Output:-**



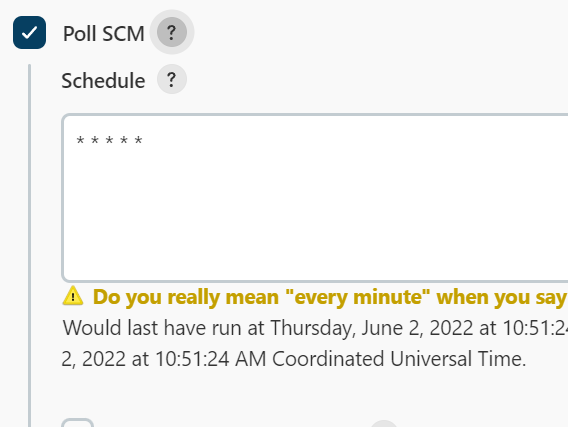
**Another job for printing Dockerfile wordcount taking from github whenever a commit occurred checks each minute..**



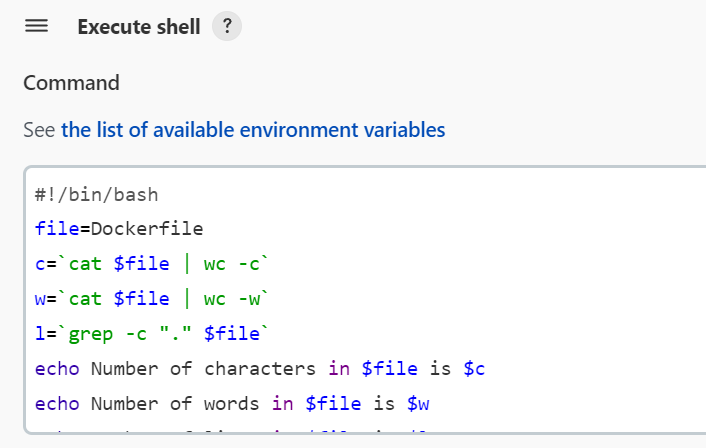
**Adding our git ripo…authenticating ssh key**



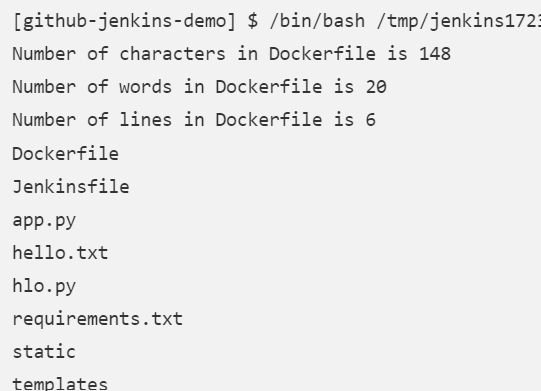
**Our branch…**



**Triggers for every commit checks every minute…**



**This will prints wordcount , char count and lines in dockerfile…**



**Date:- 20-05-2022**

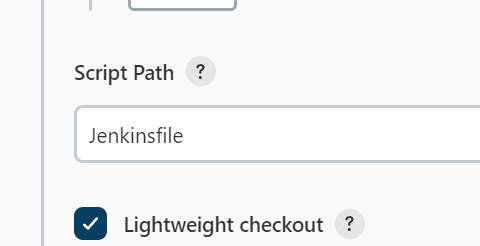
**Task 3:- installed docker plugins and pipeline jobs for Jenkins and tried to run docker commands different pipeline jobs,post builds…**

**🡪but couldn’t run docker commands because I didn’t mounted docker.sock file while creating Jenkins container tried other ways but couldn’t complete so mounted Jenkins again with docker\_file…and my jobs also not removed because I have mounted with same path as my old Jenkins volume….**

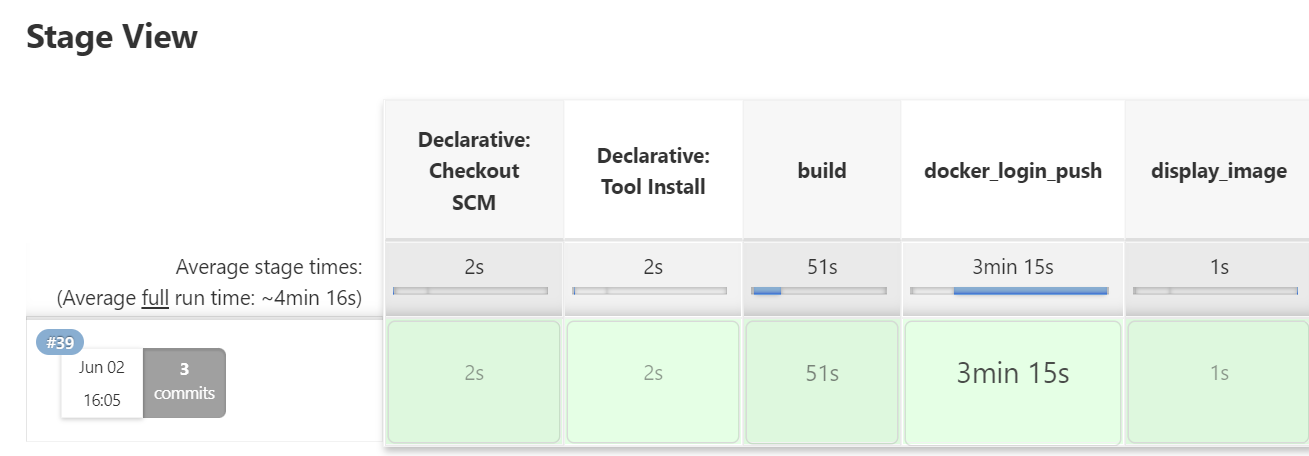
**In this task tried pipelines jobs as well…after mounting docker.sock file was able to use docker in Jenkins…for that we need to create an installation tool for docker as well…**

**In the same task**

**Building a pipeline job using jenkinsfile from github and building an image then pushing that image to local docker repo then executing “docker images” cmd…**



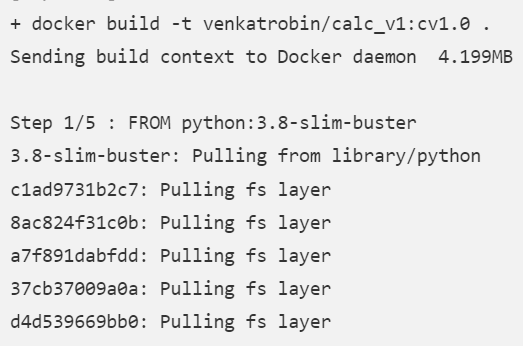
**This is jenkinsfile with pipeline code present in github..**



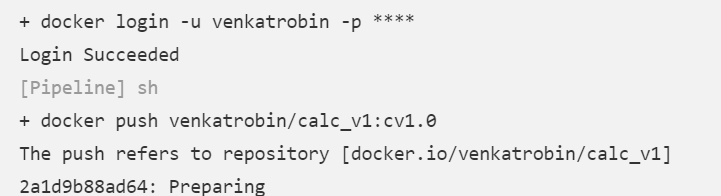
**Build steps in stage view….**



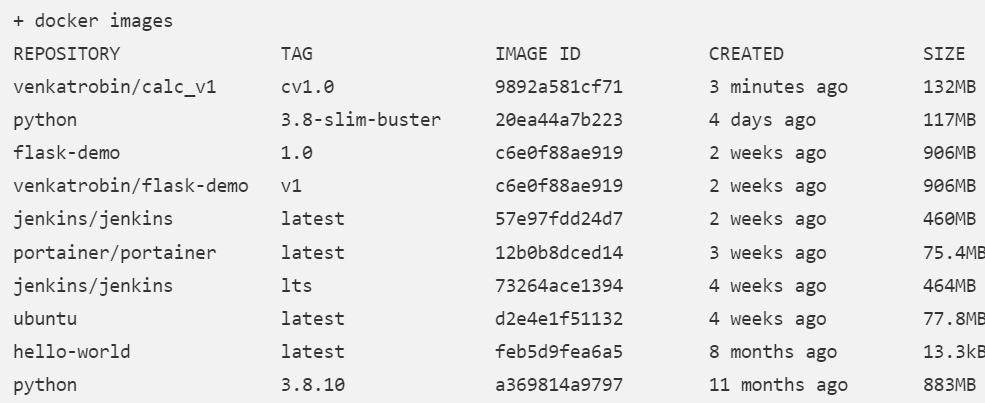
**Github and docker login**



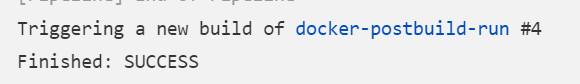
**Building docker image…**



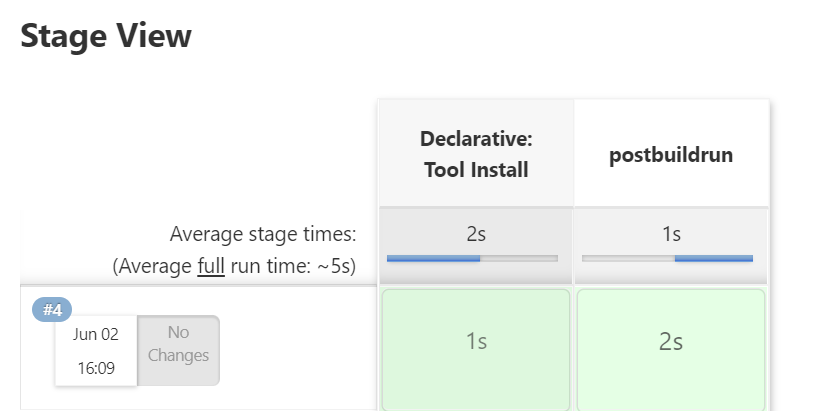
**Login to docker and Pushing image**

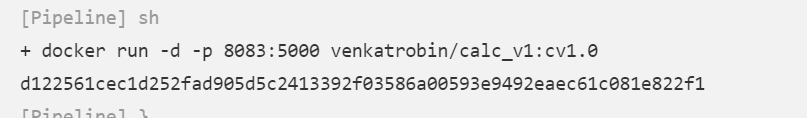


**Displaying docker images**



**Triggered new job**

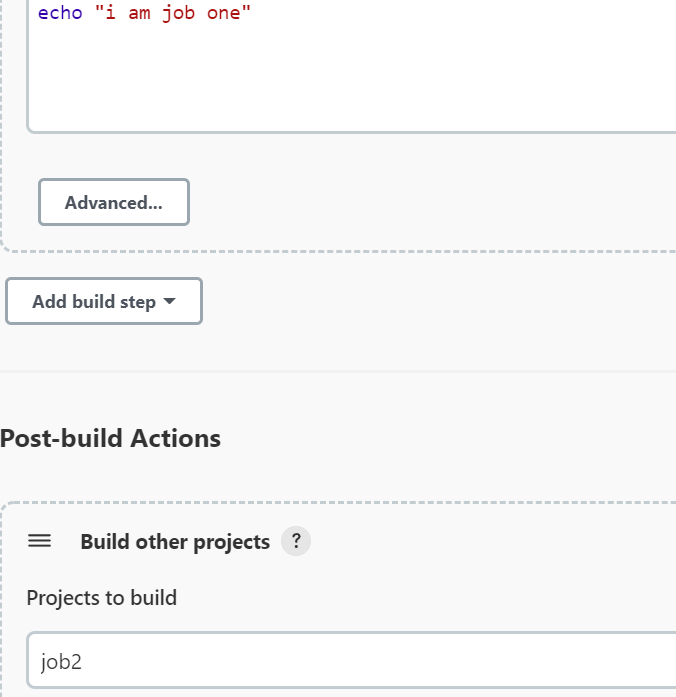


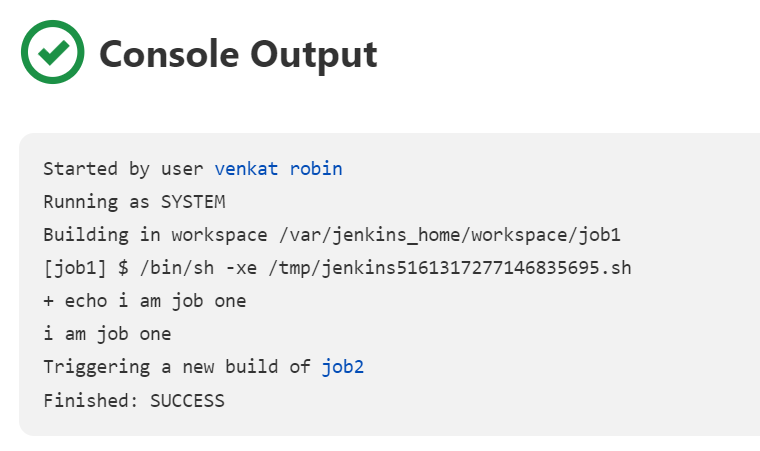


**Running new container…**

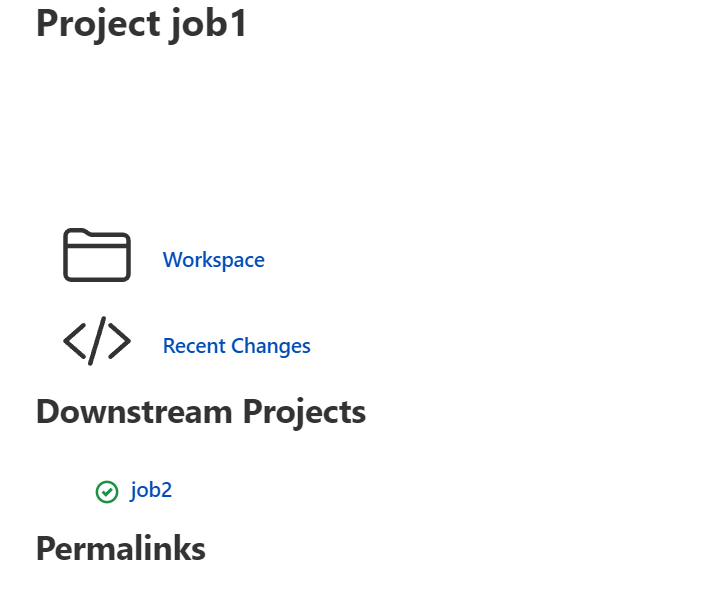
**UPSTREAM and DOWNSTREAM free style:-**

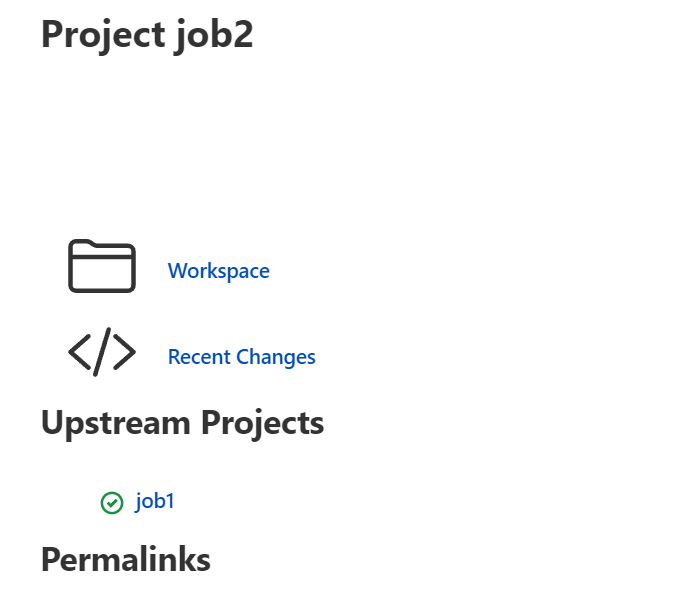
Freestyle job1

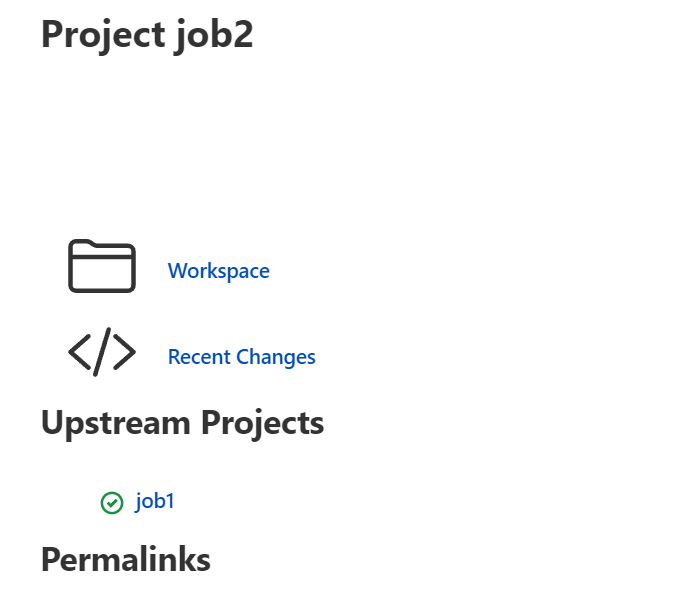




After building it will trigger job2





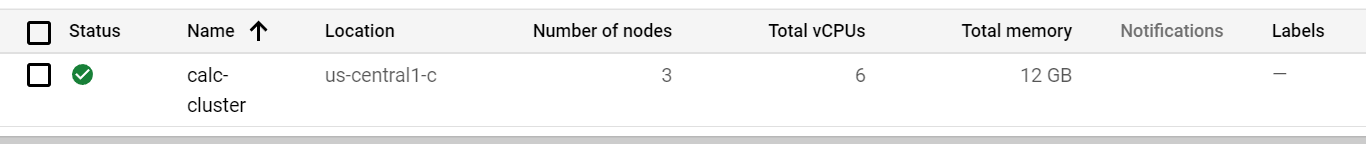




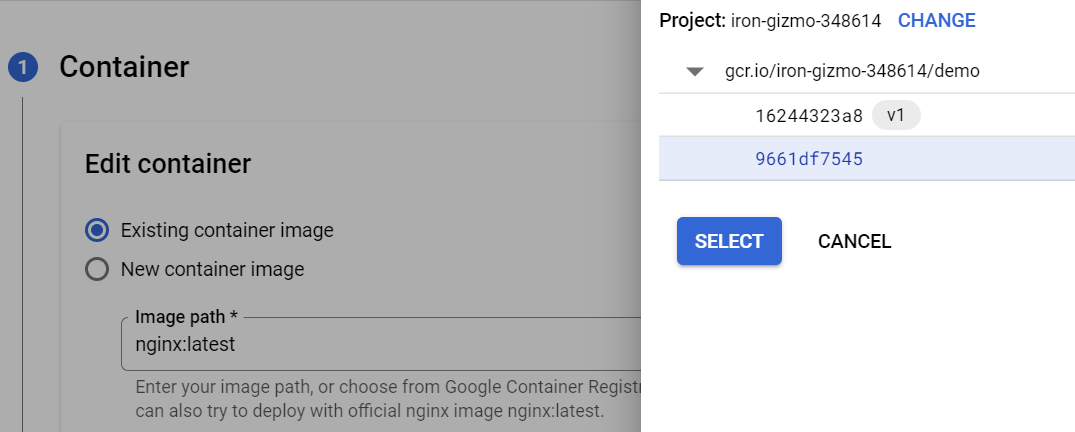
**Date :- 27-05-2022**

**Task 4:- Kubernetes**

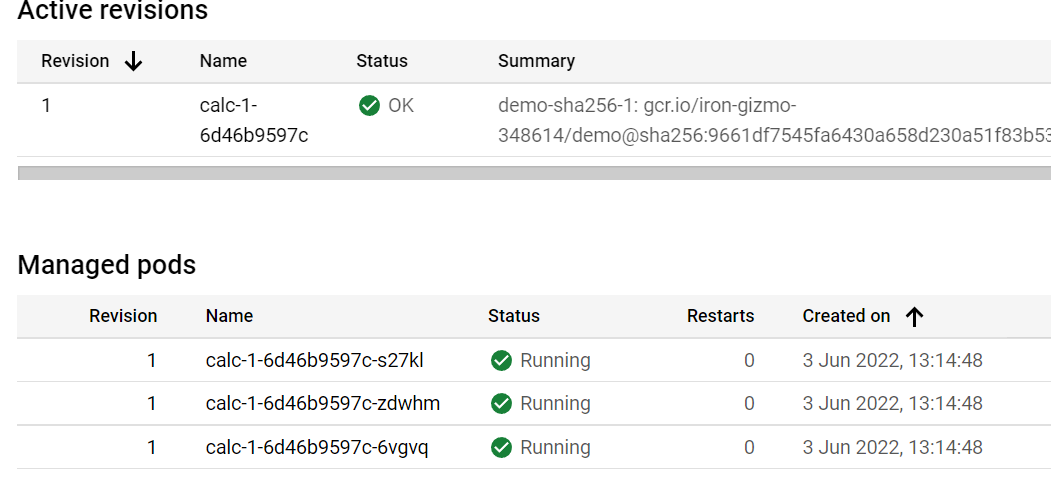
**Creating cluster ,deploying and exposing pods**



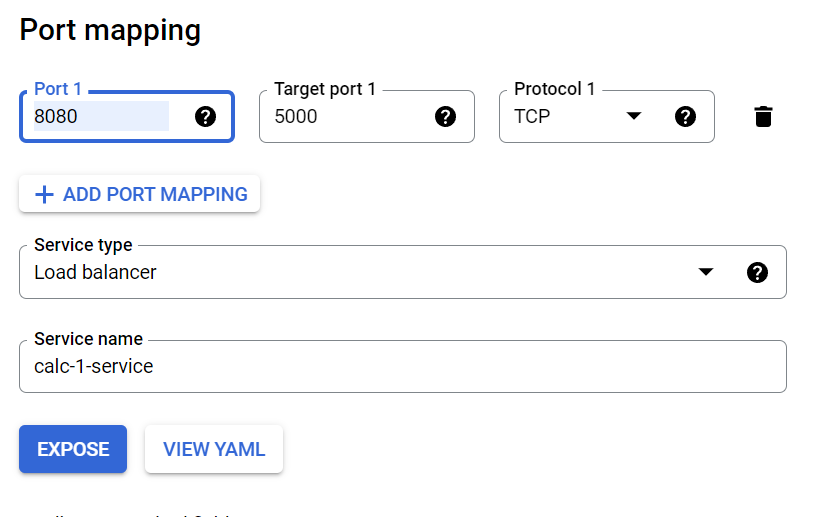
**Created cluster calc-cluster with 3 nodes…**



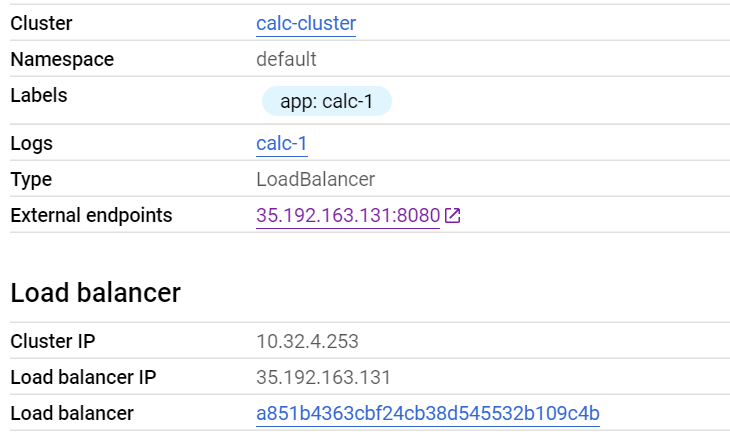
**Giving container from container registry…already uploaded from cloud shell…**



**Our pods:**



**Creating service for our pods;**



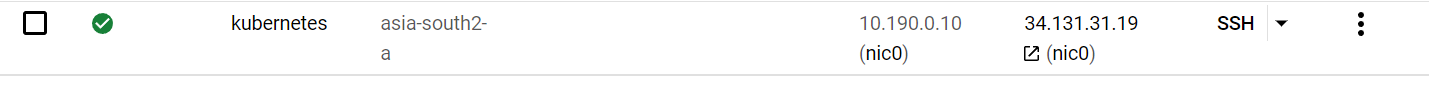
**There we have our endpoints…app running on 8080 port**



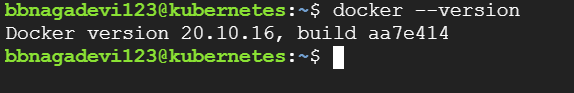
**Final output…**

**Kubernetes using minikube and nginx:-**

**🡪create instance**

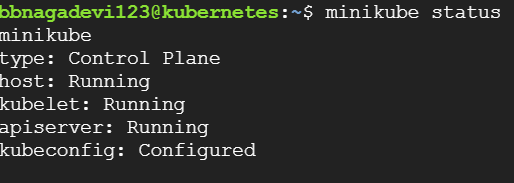


**Install Docker:**



**Install minikube and start**

[**https://minikube.sigs.k8s.io/docs/start/**](https://minikube.sigs.k8s.io/docs/start/)



**🡪Installation link for minikube**

**sudo usermod -aG docker $USER && newgrp docker 🡪add docker user**

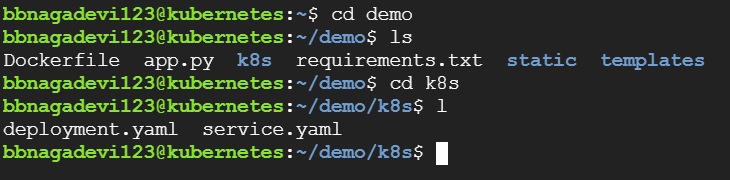
**minikube: a ONE Node cluster, where the master and worker processes are on the same machine**

**Install kubectl CLI using below cmd**

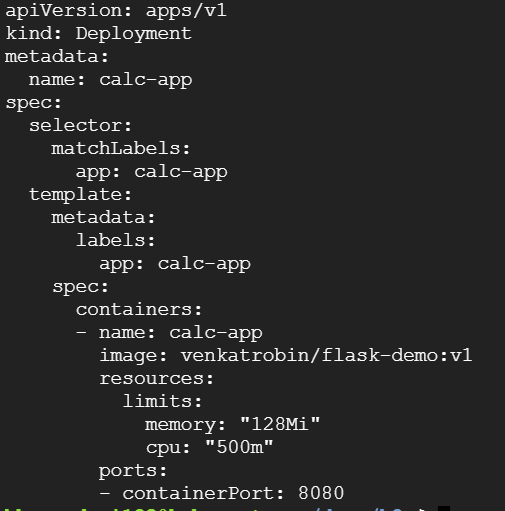
**sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl**

**Clone our git repository**

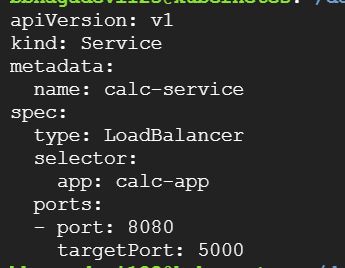
**git clone -b master https://github.com/venkat-0007/demo.git**



**Deployment manifest file**



**Service yaml file**

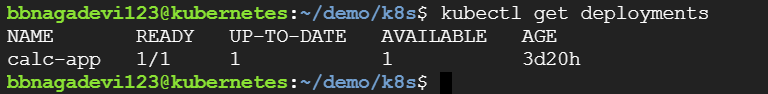


**kubectl apply -f deployment.yaml**

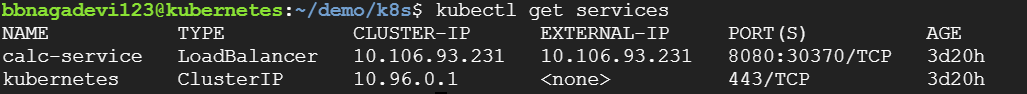
**kubectl apply -f service.yaml**

**apply both files and the deployment and services will be created…with above configuration**

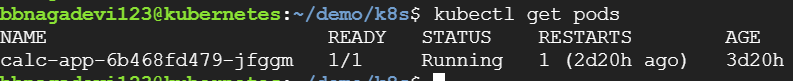
**get our deployments:-**



**Get our services:**

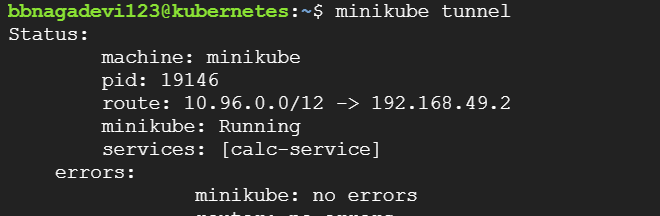


**Get pods:**



**Use minikube tunnel for creating network route to the service CIDR of the cluster using the cluster's IP address ..**

**Use this in other ssh session**



**Then try curl to our external ip**



**Minikube is like a vm so basically we are creating a vm in another vm so for accessing the app through browser we use nginx…**

NGINX is open source software for **web serving, reverse proxying, caching, load balancing, media streaming…**

**Install Nginx:-**

[**https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04**](https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04)

**sudo apt install nginx**

**🡪cmd to install nginx**

[**https://stackoverflow.com/questions/61044914/how-to-expose-minikube-in-gcp-vm**](https://stackoverflow.com/questions/61044914/how-to-expose-minikube-in-gcp-vm)

**this is the link I have used to make connect with minikube…**

**nginx will run with gcp vm external ip and from that nginx will redirect request to minikube and give response as well..**

**sudo vim /etc/nginx/conf.d/upstream.conf**

**use this command to config upstream file**

**upstream app\_server\_32108 {**

**server <minikube ip>:<port>;**

**}**

**server {**

**listen 80;**

**location /proxy {**

**proxy\_pass http://app\_server\_32108/;**

**}**

**}**

**This is how it will looks before configuring…**

**upstream calc\_app {**

**server 10.106.93.231:8080;**

**}**

**server {**

**listen 80;**

**location / {**

**proxy\_pass http://calc\_app/;**

**}**

**}**

**We need to give our external ip and port…then remove proxy at path if we don’t remove proxy then we will get only html output it cant access static and templates files..**

**And finally we need to comment out a line in nginx cofig file**

**sudo vim /etc/nginx/nginx.conf (path)**



**Finally reload nginx..**

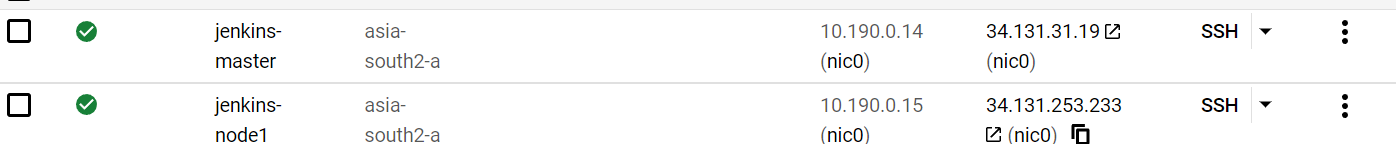
sudo nginx -t

sudo systemctl reload nginx

Now we can access our application just by clecking vms external ip..

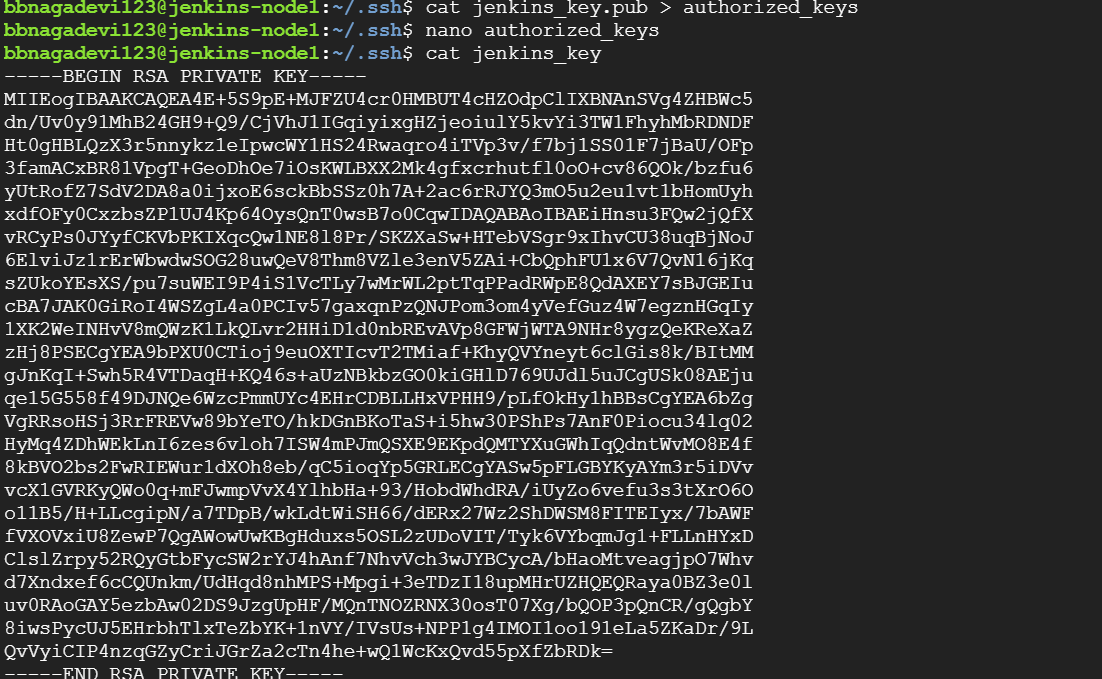
**Jenkins Master and Slave:-**

**Create master and slave instances**

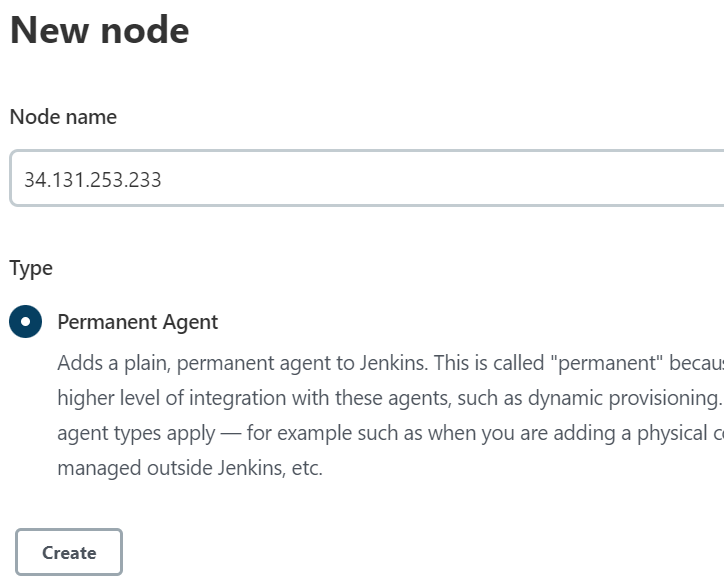


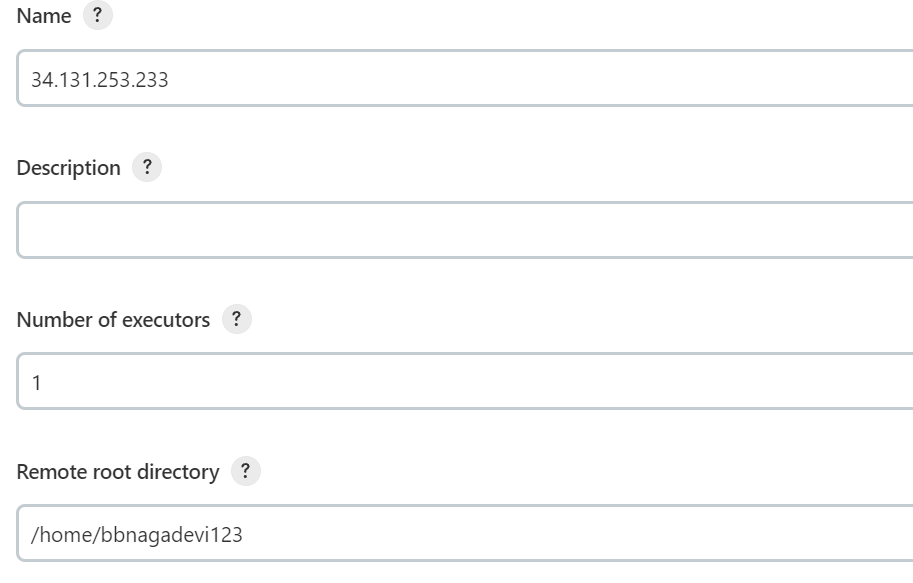
**Install Jenkins and java on master…install java on slave instance**

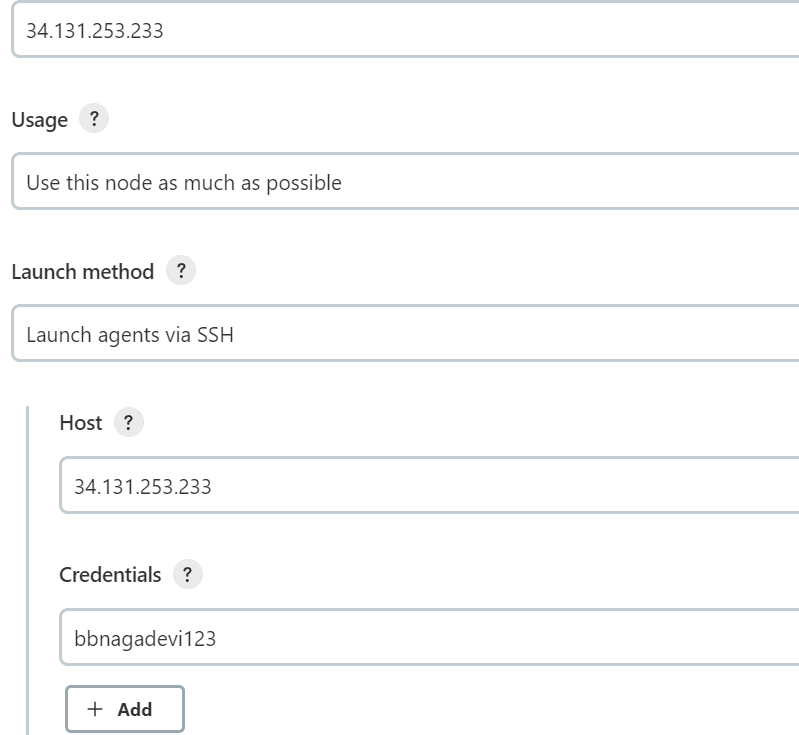
**In slave instance create ssh keys public and private…another key file authorized\_keys as well and copy public key to authorized keys…**



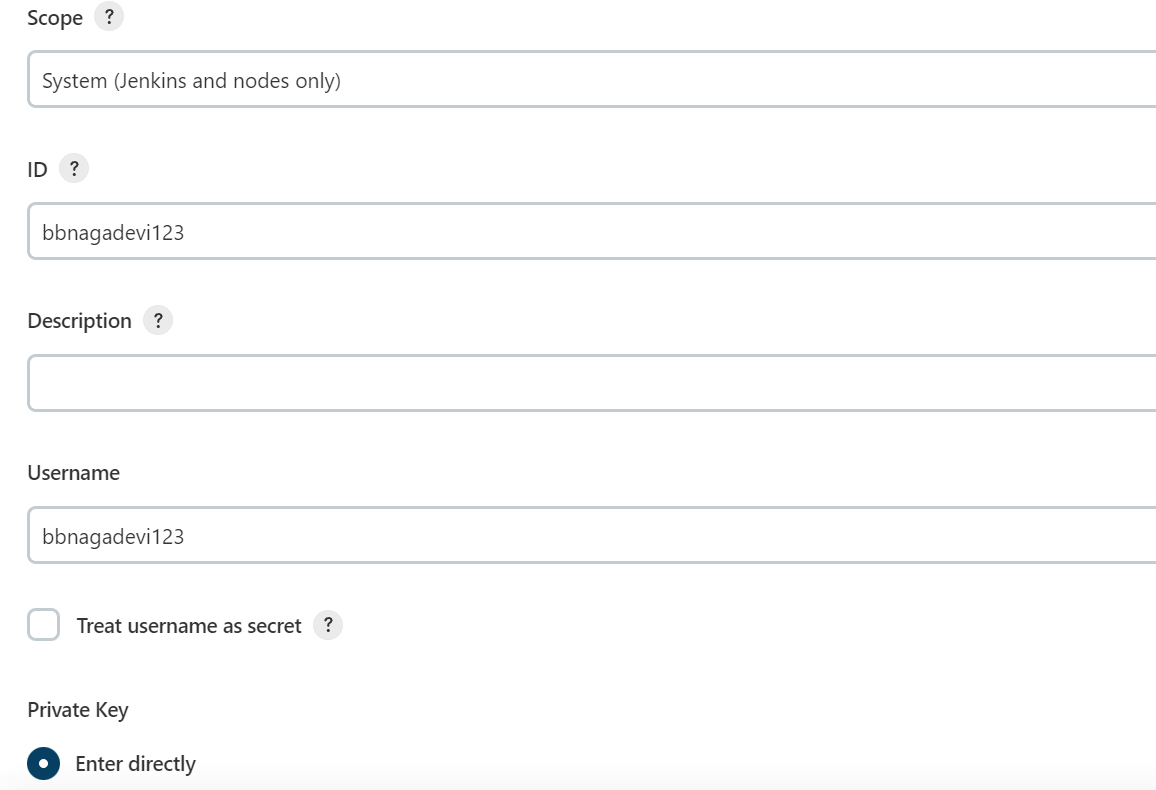
**Open Jenkins and create new node**



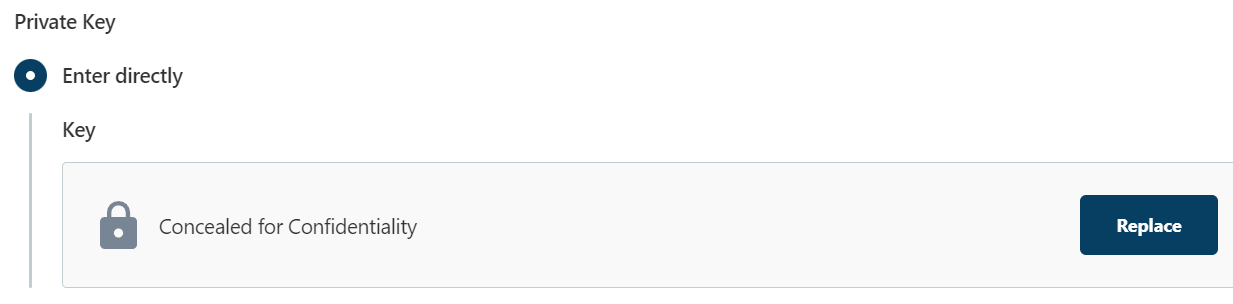


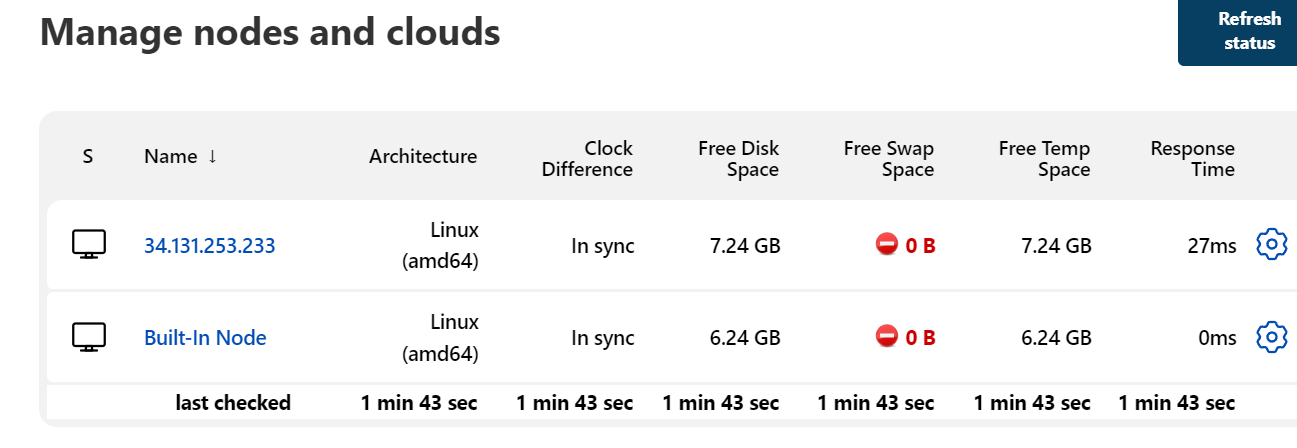


**Add credentials**

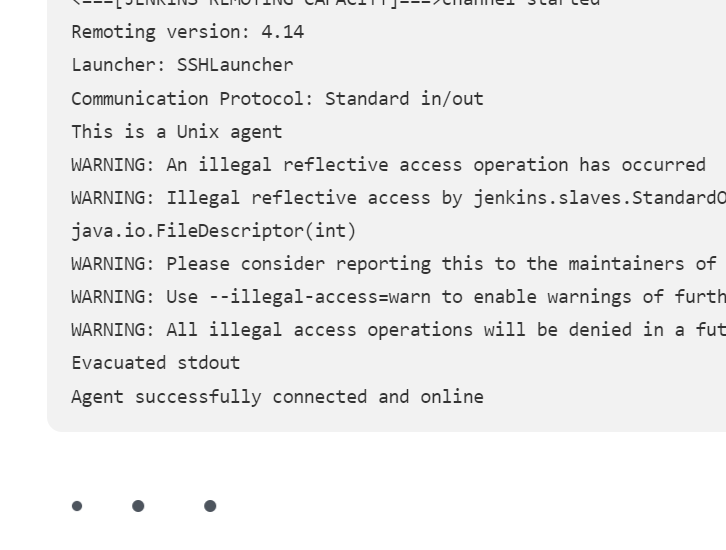


**Add private key from slave node**





**Launch new node**



**Now its online…**

**https://www.code4projects.net/kubernetes-services-cluster-ip-vs-nodeport-vs-loadbalancer-vs-ingress/**