**Name: Bhargava Varanasi**

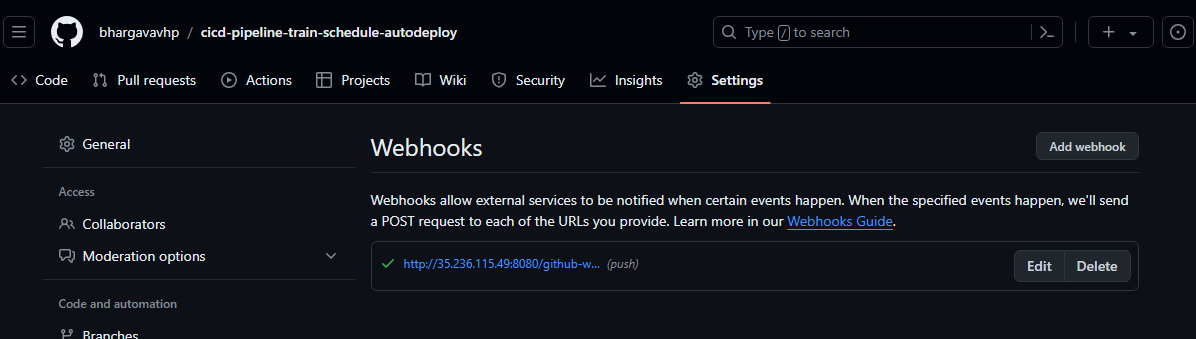
**Certification Project – 2:**

Code: [bhargavavhp/cicd-pipeline-train-schedule-autodeploy: Train Schedule sample app for Jenkins Pipelines CD with Kubernetes Fully Automated Deployment exercises (github.com)](https://github.com/bhargavavhp/cicd-pipeline-train-schedule-autodeploy)

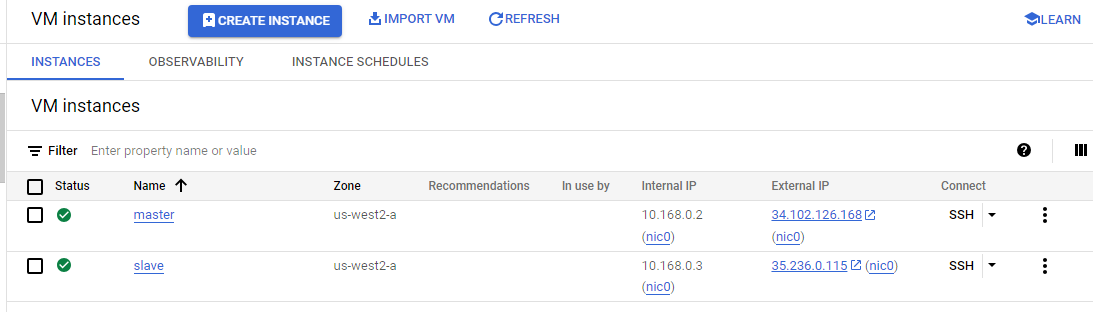
* The team of developers working on new features will merge their code to a GitHub repo.
* As soon as the code reaches GitHub, using a CI (Continuous Integration) pipeline, setup in Jenkins, automated builds will be triggered.
* The automated builds will frequently deploy new features to the production website.
* Every build will prepare a Dockerfile and push docker images to docker-hub.
* Every docker image will be deployed (Continuous Deployment) to a kubernetes-cluster.

**Solution:**

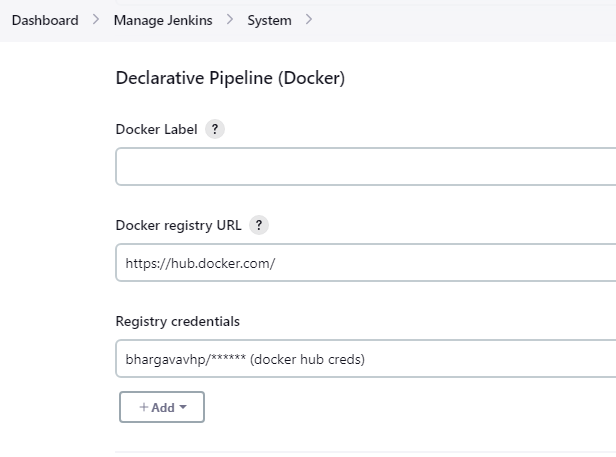
Configured github-webhook for Jenkins server



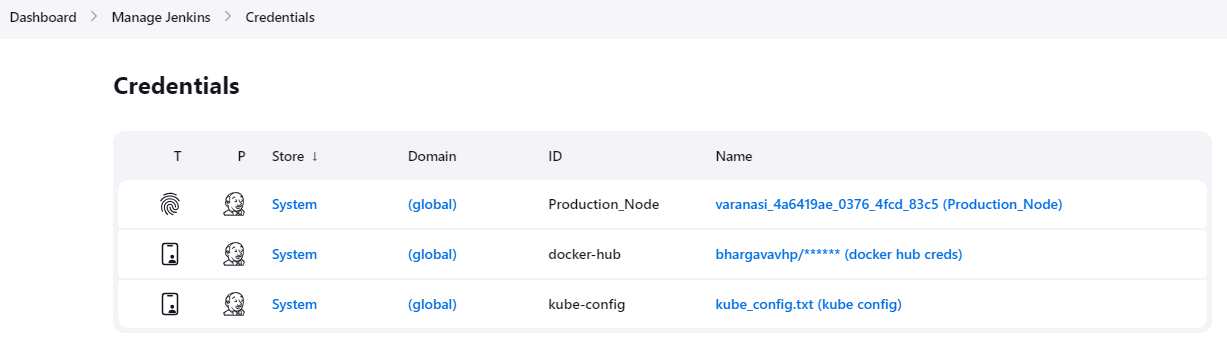
Created Kubernetes master and slave node in GCP



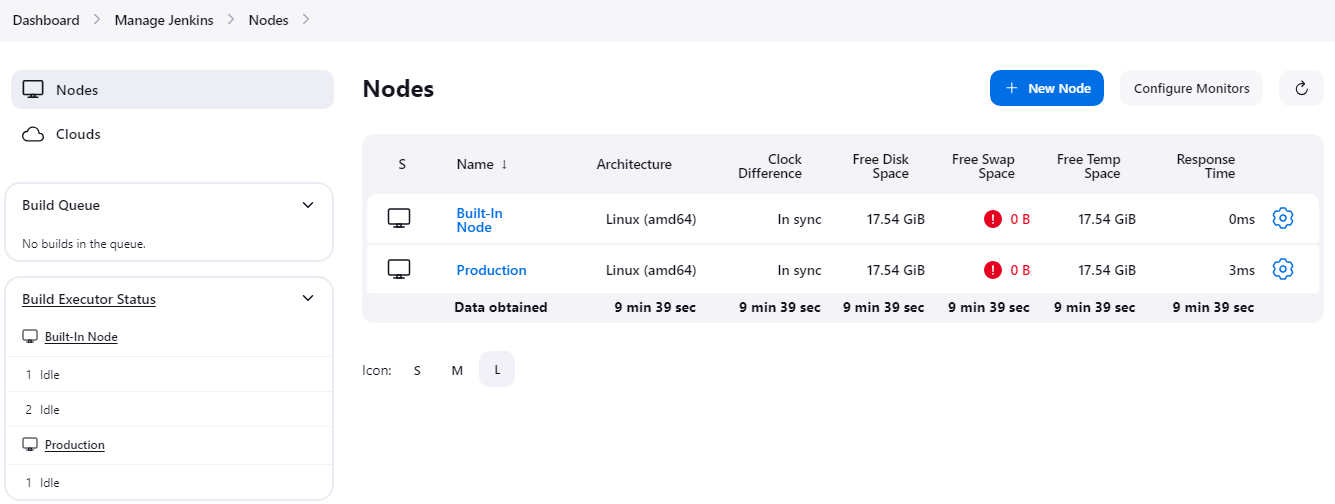
Credentials used for Docker Hub in Jenkins.



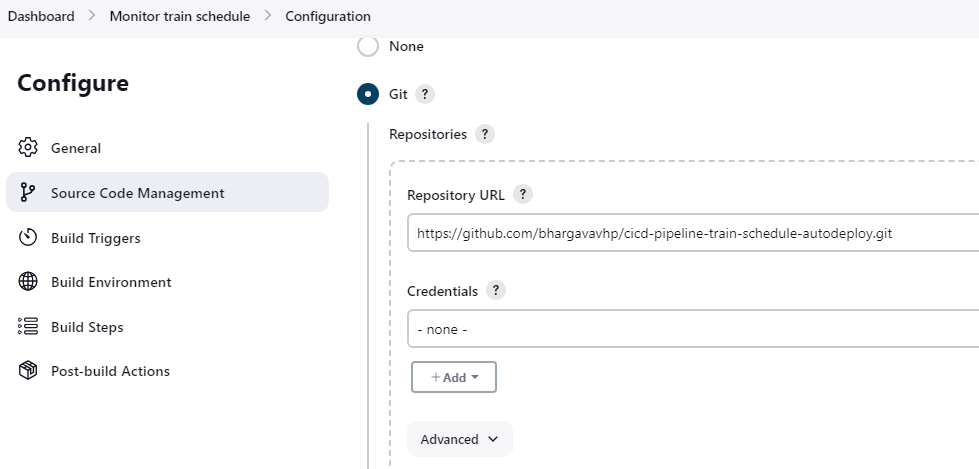
Kube-config secret file added in credentials

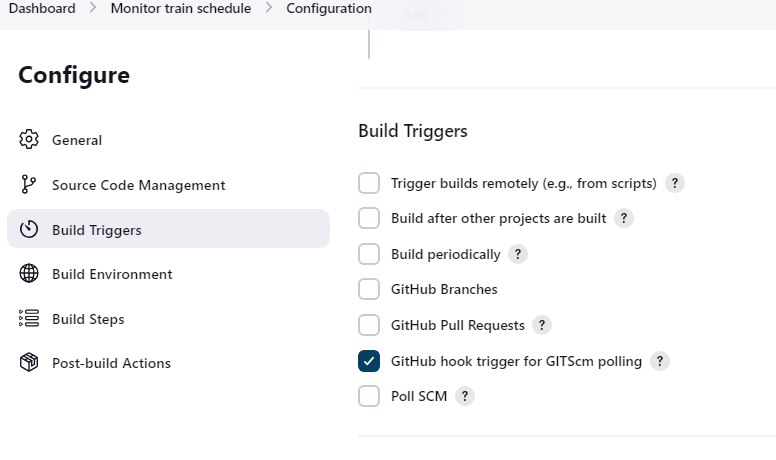


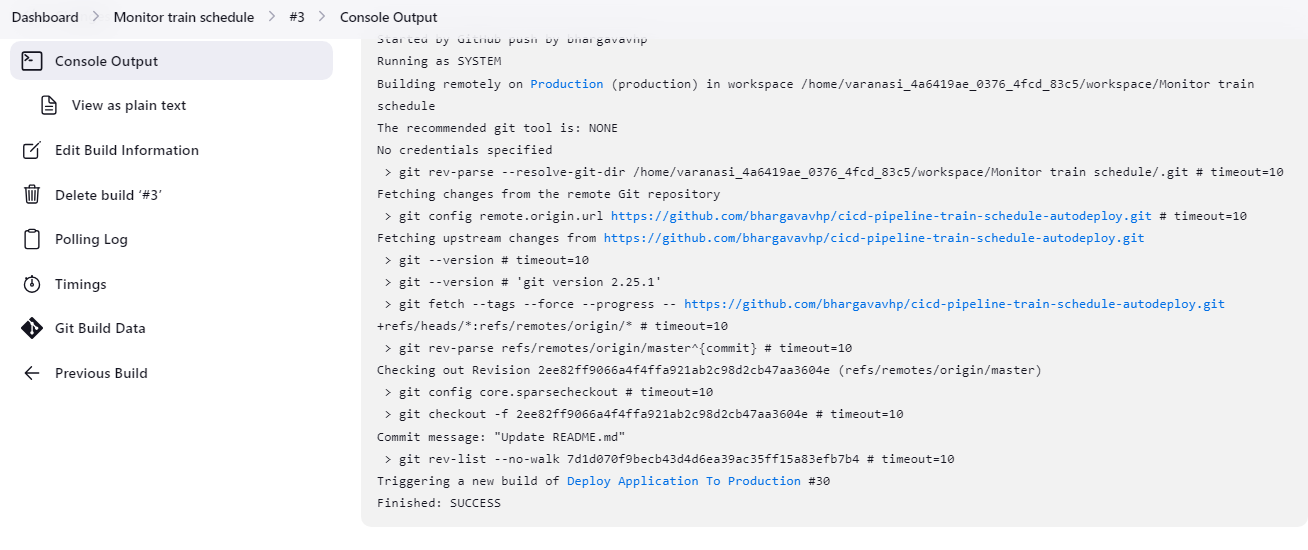
Production node added in Jenkins



Created job “Monitor train schedule” which will be triggered when developer pushes code to git repository.

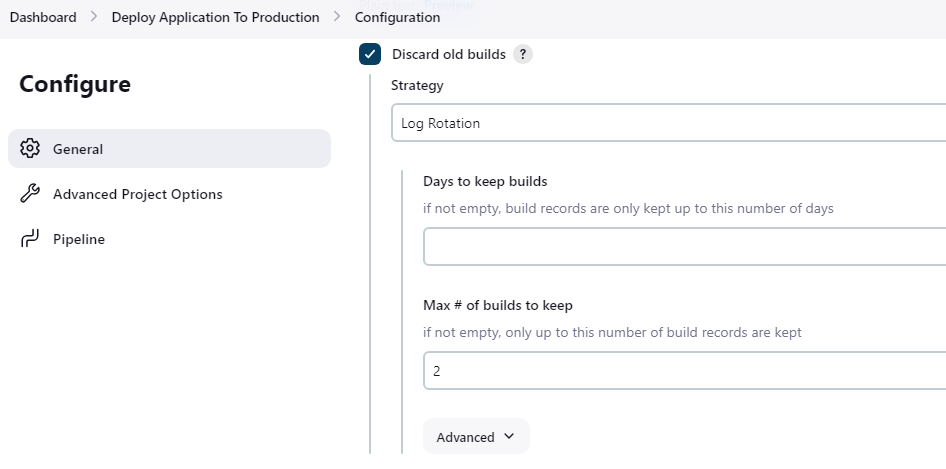




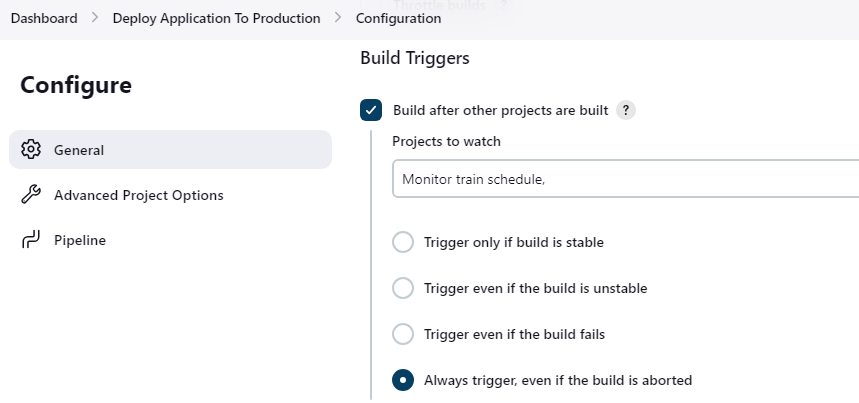


Created a pipeline job “Deploy to production” which will be triggered when “Monitor train schedule” job receives webhook events from git.

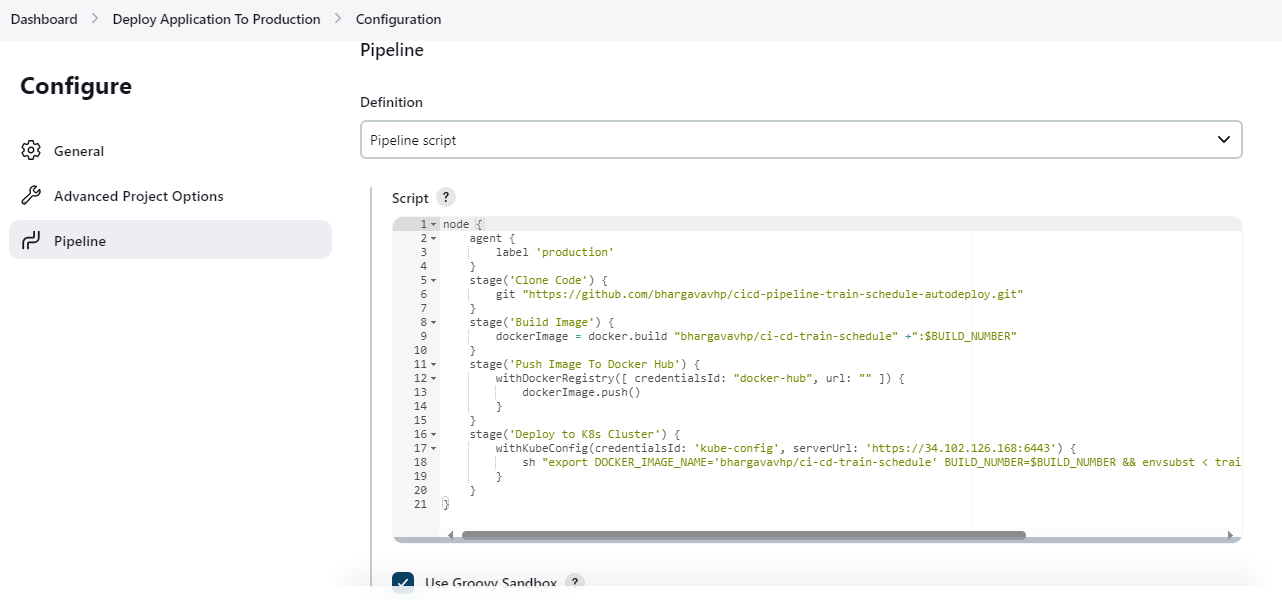
Only last two builds in the pipeline will be kept and old builds are discarded.



Configured to trigger this pipeline when “Monitor train schedule” receives webhooks from git.



Created a pipeline script to clone code, build image, push image to docker-hub and deploy to k8s cluster.



Steps Performed for this pipeline:

1. Configured docker-hub credentials in Manage Jenkins >> System
2. Installed required plugins for Docker and Kubernetes configuration
3. Deployed on the agent node labelled with “production”
4. Cloned the code from <https://github.com/bhargavavhp/cicd-pipeline-train-schedule-autodeploy.git>
5. Built the docker image with tag “bhargavavhp/ci-cd-train-schedule” and the build number of Jenkins build
6. Pushed the image with custom tag to <https://hub.docker.com/> using docker-hub credentials defined previously
7. After image is pushed, deployed the application on to Kubernetes cluster using “withKubeConfig” attribute provided by “Kubernetes CLI” plugin, authenticated to k8s cluster using kube config secret file with name “kube-config” created in Jenkins.
8. Verified that the application is deployed and accessible on NodePort - 31234

**Pipeline code:**

node {

agent {

label 'production'

}

stage('Clone Code') {

git "https://github.com/bhargavavhp/cicd-pipeline-train-schedule-autodeploy.git"

}

stage('Build Image') {

dockerImage = docker.build "bhargavavhp/ci-cd-train-schedule" +":$BUILD\_NUMBER"

}

stage('Push Image To Docker Hub') {

withDockerRegistry([ credentialsId: "docker-hub", url: "" ]) {

dockerImage.push()

}

}

stage('Deploy to K8s Cluster') {

withKubeConfig(credentialsId: 'kube-config', serverUrl: 'https://34.102.126.168:6443') {

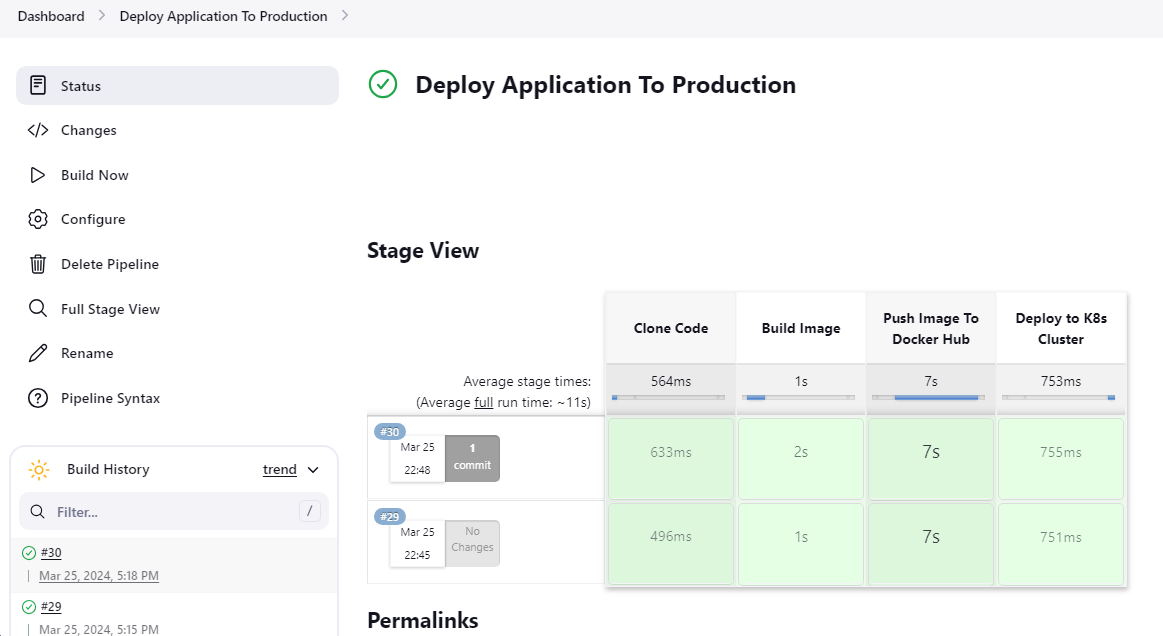
sh "export DOCKER\_IMAGE\_NAME='bhargavavhp/ci-cd-train-schedule' BUILD\_NUMBER=$BUILD\_NUMBER && envsubst < train-schedule-kube.yml | kubectl apply -f -"

}

}

}

Pipeline executed successfully after developer pushes changes to git using webhook



Build Pipeline View of project after github webhooks are triggered

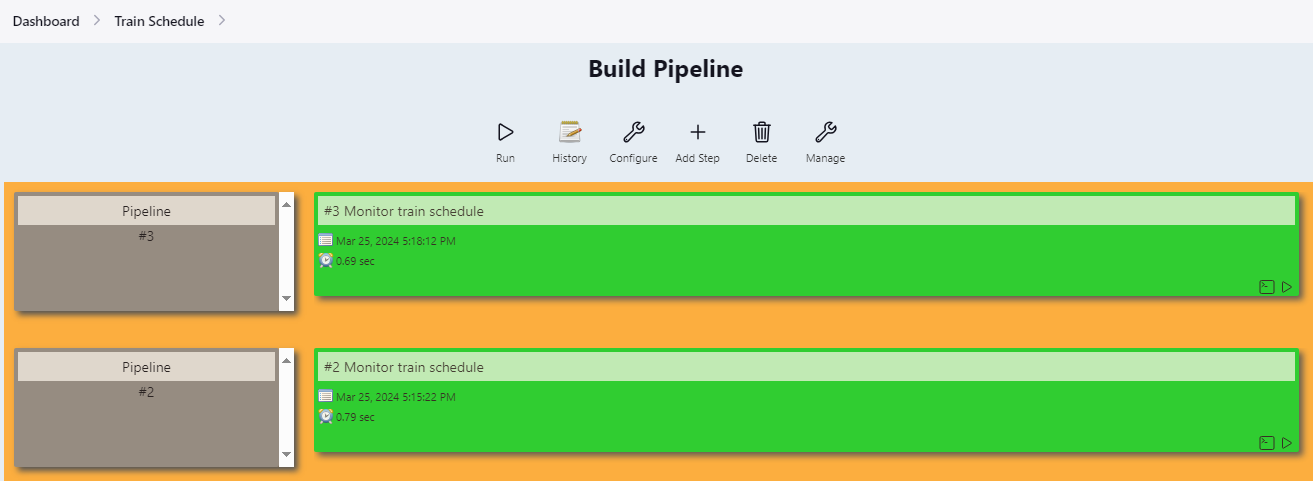
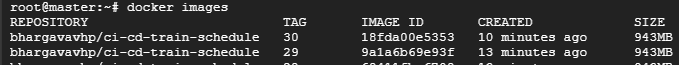
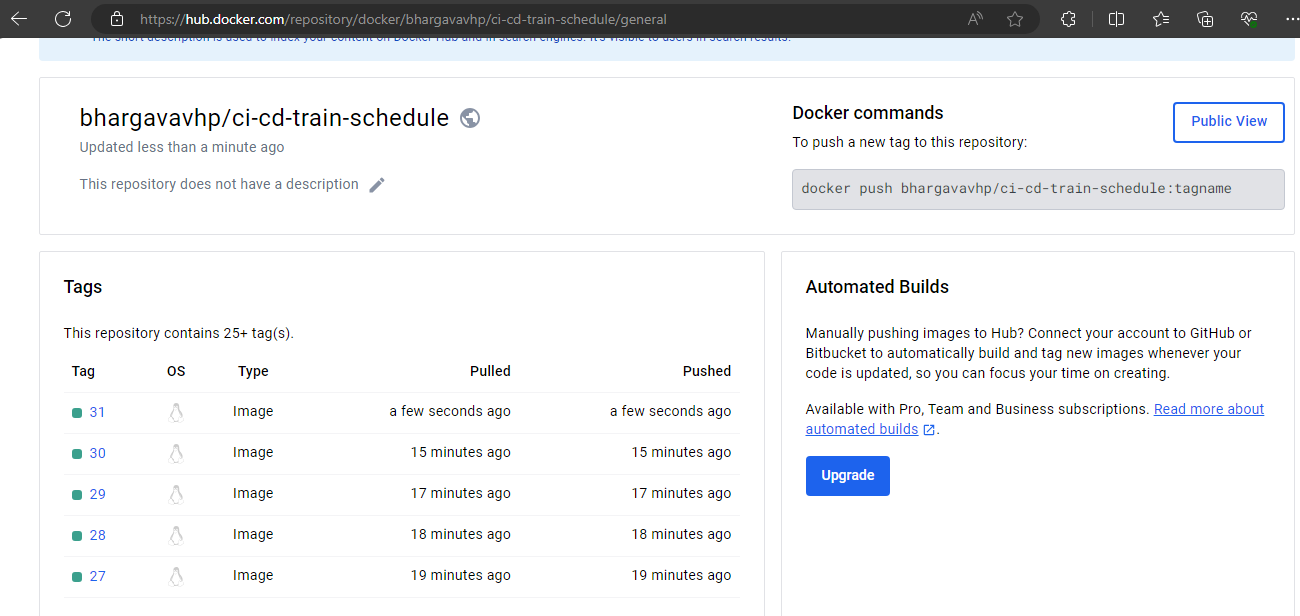


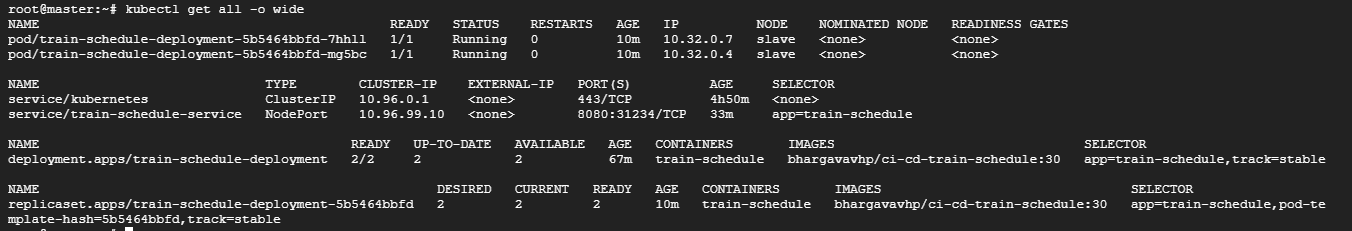
Image created on server



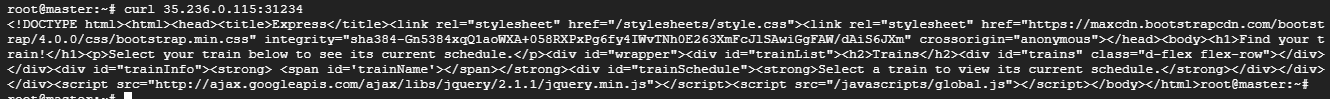
After pushing the image to docker-hub, can able to see images with custom tags under “bhargavavhp/ci-cd-train/schedule” repo.



Status of Kubernetes cluster after deployment, deployment with 2 replicas is created and exposed the deployment using NodePort.



Could able to curl on the application using public IP and NodePort



Can able to access application in browser.