Published in SysopsMicro

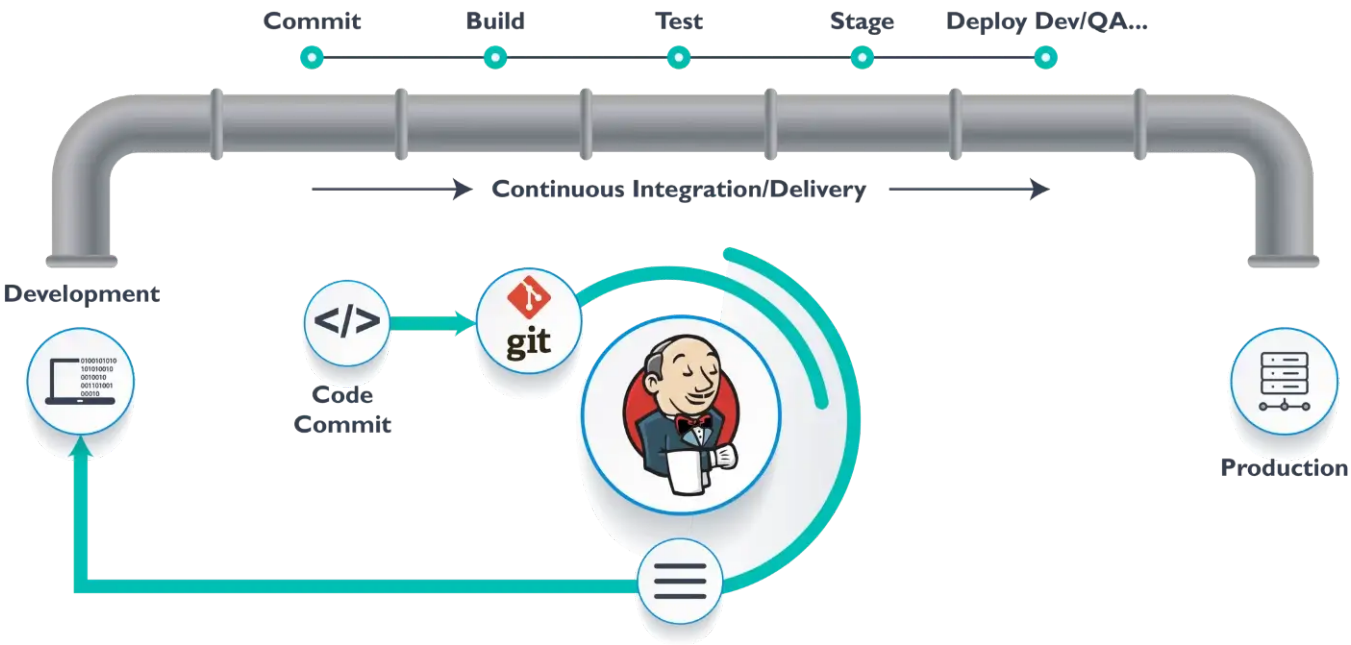
@pramodAIML Follow

May 23, 2021 · 7 min read · Listen

Save

DevOps CI/CD Pipeline with Jenkins, Kubernetes & GitHub: Part 1

How to set up Jenkins and CI/CD pipelines using GitHub?



source

I feel:

> “DevOps is a must for all kinds of tech startups, small or big. It may be complex to start with but will make your life hell easy going forward and can save loads

307 5

> of time and energy for the organization to scale, scale and scale..”

Today we will look into how we can set up a CI/CD pipeline to automate the code deployment using the power of Jenkins.

In this part, we will cover :

- How to build a Jenkins Docker image?
- How to deploy the Jenkins image onto our local Minikube cluster?
- How to run the deployed Jenkins image on our web browser?

Prerequisite Steps :

- Install minikube

minikube start

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes. All you need is Docker...

minikube.sigs.k8s.io

- Install docker desktop

Install Docker Desktop on Mac

Estimated reading time: 5 minutes Welcome to Docker Desktop for Mac. This page contains information about Docker...

docs.docker.com

- Create GitHub organization, give it a name, and fork all of the sample repositories from the given URL :

eazyfin-pramod

Java MIT Updated 495 0 0 0 May 22, 2021 Java MIT Updated 489 0 0 0 Nov 11, 2020 Java MIT Updated 485 0 0 0 Apr 13, 2020...

github.com

Once you are done with the above-mentioned steps, its time that we learn how to setup Jenkins in our local Minikube cluster

How To Setup Jenkins on Our Minikube K8s cluster?

If you have forked all the repositories from my Github organization repository account, you will find one repo named:

eazyfin-pramod/fleetman-api-gateway

This branch is 2 commits ahead of fleetman-ci-cd-demo:master. This is not intended to be a full production strength API...

github.com

So now you can simply clone the same using the following K8s command, let's see step by step how to do so

I will be using my mac terminal to run all my bash commands

Step1: Start Minikube cluster :

Here, I am specifying a memory of 4 GB, because Jenkins is quite resourced intensive so please ensure you are starting the Minikube cluster with these extra flags.

```
$ minikube start — memory 4028
```

```
pramodchandrayan@Pramods-MacBook-Pro ~ % minikube start --memory 3900
😄  minikube v1.17.1 on Darwin 11.0.1 (arm64)
✨  Using the docker driver based on existing profile
❗  You cannot change the memory size for an exiting minikube cluster. Please first delete the cluster.
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.20.2 on Docker 20.10.2 ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass

❗  /usr/local/bin/kubectl is version 1.18.9-eks-d1db3c, which may have incompatibilites with Kubernetes 1.20.2.
    ▪ Want kubectl v1.20.2? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Fig 1.0: Start minkuibe

If you managed to successfully start the minukube you will see the output as shown in Fig 1.0

Then test if everything is fine by coining the below command

```
$ minikube status
```

```
pramodchandrayan@Pramods-MacBook-Pro jenkins % minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent

pramodchandrayan@Pramods-MacBook-Pro jenkins %
```

minikube status output

You should be able to see the output as shown above.

2. Clone the Sample repo:

Given below

eazyfin-pramod/fleetman-api-gateway

This branch is 2 commits ahead of fleetman-ci-cd-demo:master. This is not intended to be a full production strength API...

github.com

In your local machine and keep it inside a folder named CICDdemo(you can keep whatever name to your liking)

3. Enter into the CICDdemo folder in your terminal using the below-given, “change directory” command

```
$ cd /Applications/K8s/CICDDemo
```

you will get to see jenkins folder which will have two files. If you further go inside the jenkins folder by changing the directory from your terminal, you will be seeing the below-given files.

- *jenkins.yaml*
- *Dockerfile*

As shown below,

```
Last login: Sun May 23 01:45:11 on ttys001
pramodchandrayan@Pramods-MacBook-Pro ~ % cd /Applications/K8s/CICDDemo
pramodchandrayan@Pramods-MacBook-Pro CICDDemo % ls
jenkins
pramodchandrayan@Pramods-MacBook-Pro CICDDemo % cd jenkins
pramodchandrayan@Pramods-MacBook-Pro jenkins % ls
Dockerfile      jenkins.yaml
pramodchandrayan@Pramods-MacBook-Pro jenkins %
```

fig 2.0

Now that you have cloned and are in the jenkins directory, it's time to build the docker image of Jenkins.

3. Build Jenkins Docker Image:

We will be building a jenkins docker image using Dockerfile present inside the jenkins folder, by making use of the **docker build image** command, but before that, it is important to change the environment variable so that we can make use of the local docker-daemon service running inside the minikube cluster,

Assuming that you are into your local jenkins directory, type the following commands

```
$ minikube docker-env
$ eval $(minikube -p minikube docker-env)
```

This will point your shell to dicker-daemon, run, as shown below

```
pramodchandrayan@Pramods-MacBook-Pro ~ % minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://127.0.0.1:53691"
export DOCKER_CERT_PATH="/Users/pramodchandrayan/.minikube/certs"
export MINIKUBE_ACTIVE_DOCKERD="minikube"

# To point your shell to minikube's docker-daemon, run:
# eval $(minikube -p minikube docker-env)
pramodchandrayan@Pramods-MacBook-Pro ~ %
```

```
pramodchandrayan@Pramods-MacBook-Pro ~ % eval $(minikube -p minikube docker-env)
pramodchandrayan@Pramods-MacBook-Pro ~ % docker image ls
REPOSITORY                                          TAG                 IMAGE ID       CREATED          SIZE
eazyfinpramod/eazyfin-pramod-fleetman-api-gateway   2                   e8b6c2f920f6   34 minutes ago   130MB
eazyfinpramod/eazyfin-pramod-fleetman-api-gateway   1                   b465775fac5e   10 hours ago     130MB
myjenkins                                           latest              e31fcd2e3f67   12 hours ago     1.01GB
jenkins/jenkins                                     lts-alpine          d0b73a070e4f   2 weeks ago      365MB
k8s.gcr.io/kube-proxy                               v1.20.2             788e63d07298   4 months ago     116MB
k8s.gcr.io/kube-apiserver                           v1.20.2             95d99817fc33   4 months ago     113MB
k8s.gcr.io/kube-controller-manager                  v1.20.2             3a1a2b528610   4 months ago     107MB
k8s.gcr.io/kube-scheduler                           v1.20.2             60d957e44ec8   4 months ago     43.3MB
kubernetesui/dashboard                              v2.1.0              85e6c0cff043   5 months ago     223MB
gcr.io/k8s-minikube/storage-provisioner             v4                  84bee7cc4870   5 months ago     27.5MB
k8s.gcr.io/etcd                                     3.4.13-0            05b738aa1bc6   8 months ago     312MB
k8s.gcr.io/coredns                                  1.7.0               db91994f4ee8   11 months ago    42.8MB
kubernetesui/metrics-scraper                        v1.0.4              a262dd7495d9   14 months ago    35.2MB
k8s.gcr.io/pause                                    3.2                 2a060e2e7101   15 months ago    484kB
openjdk                                             8u131-jdk-alpine    9c0b8e044fa7   3 years ago      96.9MB
pramodchandrayan@Pramods-MacBook-Pro ~ %
```

fig 3.0

Check that you are able to get all the pre-existing docker images in that docker-daemon env:(refer the fig 3.0, for command and output)

```
$ docker image ls
```

Now that you have the docker -daemon running we can now go ahead and build the docker image by using the command shown below :

```
$ docker image build -t myjenkins .
```

Note!, the tag for the Jenkins image is myjenkins , which should match with the image tag mentioned in jenkins.yaml file (existing in your local directory), as shown below, and don't forget to apply the dot at the end with a single space while typing in your terminal, as this instructs the system to build the image from the local Docker file.

```
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
        - name: jenkins
          image: myjenkins:latest
          env:
            - name: JAVA_OPTS
              value: -Djenkins.install.runSetupWizard=false
          ports:
            - name: http-port
              containerPort: 8080
            - name: jnlp-port
              containerPort: 50000
          volumeMounts:
```

4. Wait for the Jenkins Docker Image to build :

Mind it this process is quite heavy and may take some time, so be patient until you get the following message

- *"Successfully built "imageid(this will vary for you)" "*
- *"Successfully tagged myjenkins: latest"*

As can be seen below in Fig 4.0

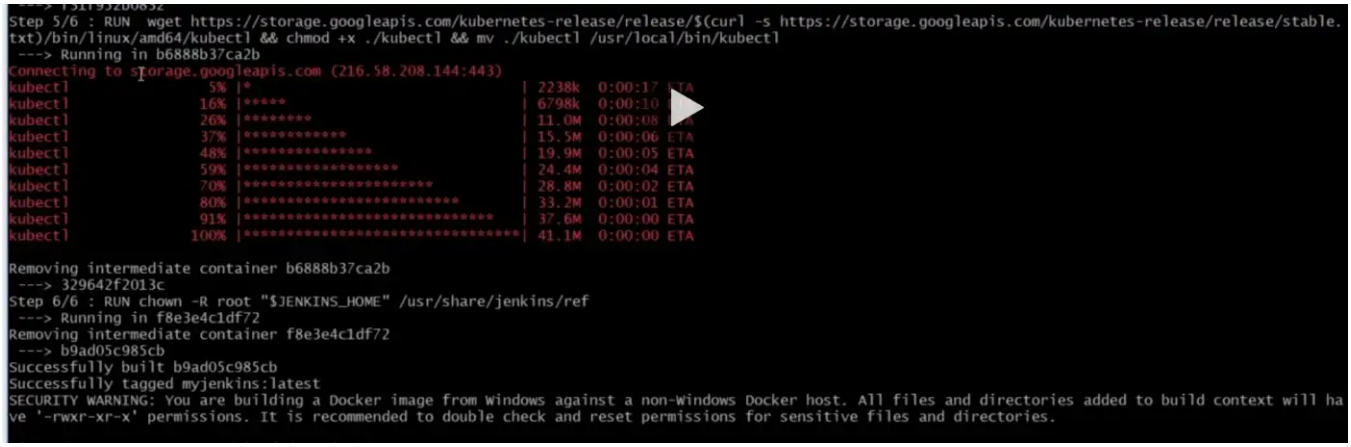


Fig 4.0

5. Deploy Jenkins Image :

Assuming that you are still in your local folder where jenkins.yaml file exists, if yes go ahead and type this k8s CLI command, to create a Jenkins deployment file using the jenkins docker image, we just created in the previous process

```
$ kubectl apply -f jenkins.yaml
```

```
Dockerfile      jenkins.yaml
pramodchandrayan@Pramods-MacBook-Pro jenkins % kubectl apply -f jenkins.yaml
serviceaccount/jenkins unchanged
role.rbac.authorization.k8s.io/jenkins unchanged
rolebinding.rbac.authorization.k8s.io/jenkins unchanged
clusterrolebinding.rbac.authorization.k8s.io/jenkins-crb unchanged
clusterrole.rbac.authorization.k8s.io/jenkinsclusterrole unchanged
deployment.apps/jenkins unchanged
service/jenkins unchanged
pramodchandrayan@Pramods-MacBook-Pro jenkins %
```

6. Check If Jenkins image is deployed :

```
$ kubectl get all
```

You can see that,

- *pod:* ***jenkins-79966d8db-x4kn7*** *is deployed and the* ***jenkins***
- *service(****service/jenkins****) is also up and running on the NodePort 30020*
- *Deployment:* ***deployment.apps/jenkins*** *is also up and running on the NodePort: 31000*

```
service/jenkins unchanged
pramodchandrayan@Pramods-MacBook-Pro jenkins % kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/api-gateway-766545bc6-rx9kf     1/1     Running   0          125m
pod/jenkins-799666d8db-x4kn7        1/1     Running   2          13h

NAME                              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                          AGE
service/fleetman-api-gateway      NodePort    10.105.58.32    <none>        8080:30020/TCP                   11h
service/jenkins                   NodePort    10.108.87.66    <none>        8080:31000/TCP,50000:32251/TCP   13h
service/kubernetes                ClusterIP   10.96.0.1       <none>        443/TCP                          17h

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/api-gateway       1/1     1            1           11h
deployment.apps/jenkins           1/1     1            1           13h

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/api-gateway-58677b968c       0         0         0       11h
replicaset.apps/api-gateway-766545bc6        1         1         1       125m
replicaset.apps/jenkins-799666d8db           1         1         1       13h
pramodchandrayan@Pramods-MacBook-Pro jenkins %
```

7. Running the Jenkins In Your Web :

If everything is set up. nicely without any error, you need to type the following command :

```
$ minikube ip
```

The above command gives the local IP of the minikube cluster, we will need this IP, to run the Jenkins UI and see it running in our web

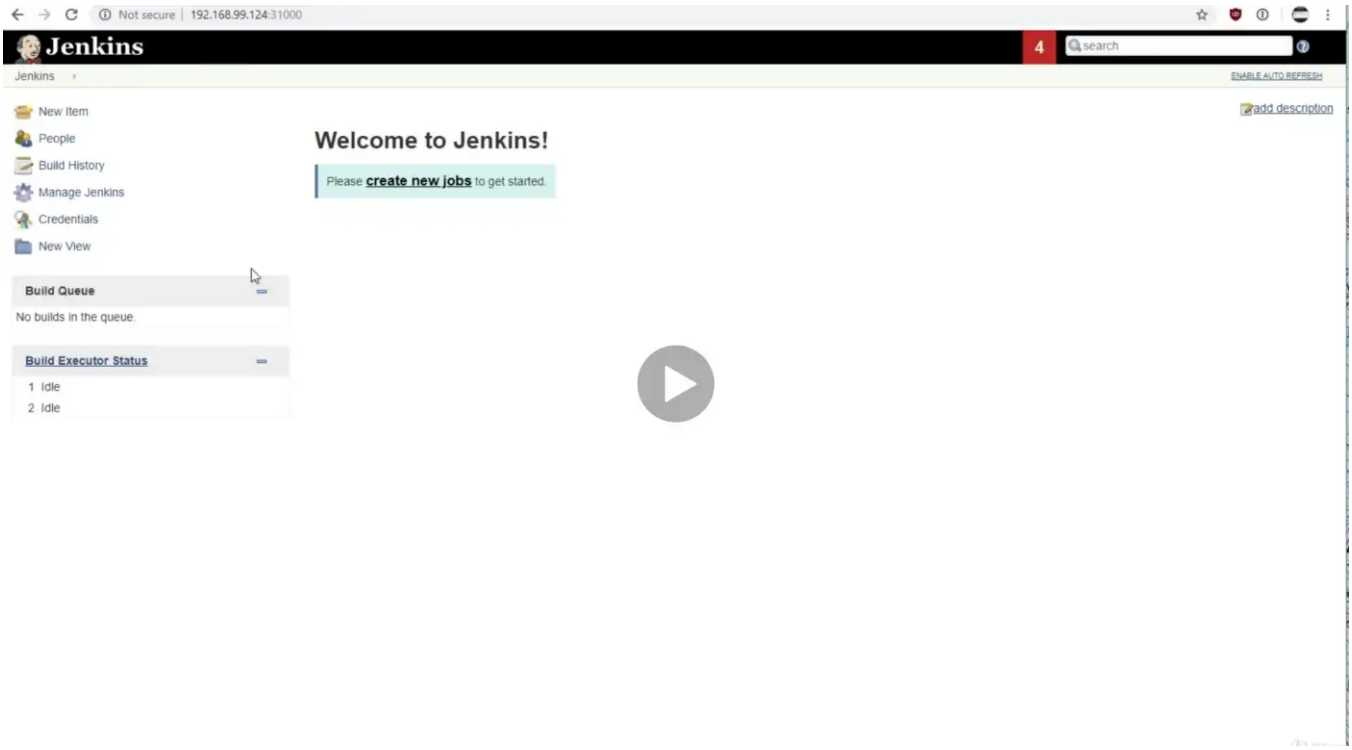
```
$ minikube ip
```

```
pramodchandrayan@Pramods-MacBook-Pro ~ % minikube ip
192.168.49.2
pramodchandrayan@Pramods-MacBook-Pro ~ %
```

As can be seen, for me my cluster IP is: **192.168.49.2**

Open the web and type this URL given below :

192.168.49.2:31000, and you will see the following screen with a welcome message



What If Your URL is not working?

It may happen if you are using the docker driver while starting the minikube , instead of the virtual box. In this situation minikube IP will not function and you will be required to start the local tunnel from your terminal using the following commands,

```
$ minikube service jenkins --url
```

```
pramodchandrayan@Pramods-MacBook-Pro ~ % minikube service jenkins --url
🏃  Starting tunnel for service jenkins.
|-----------|---------|-------------|------------------------|
| NAMESPACE |  NAME   | TARGET PORT |          URL           |
|-----------|---------|-------------|------------------------|
| default   | jenkins |             | http://127.0.0.1:53960 |
|           |         |             | http://127.0.0.1:53961 |
|-----------|---------|-------------|------------------------|
http://127.0.0.1:53960
http://127.0.0.1:53961
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
^[
```

The above commands start the tunnel for your jenkins service and give you the URL which needs to be used along with NodePort:53960 or 53961(**this may vary in your case**), to access the Jenkins UI on your web browser

For me the URL will look like this:

“127.0.0.1:53960”, type this and now your Jenkins UI should work in the browser.

Congratulations!!!

Congratulation you have successfully configured the Jenkins on your local minikube cluster and now you are all set to, configure the Jenkins interface and start automating the code repository deployment without much fuss

What’s Next?

As the length of this piece is becoming big, in part 2 of this series, we will cover how to Configure multibranch repository and GitHub organization, using Jenkins UI interface, and will see its magic to automate our CI/CD pipeline.

Part2 :

Building CI/CD Pipeline with Jenkins, Kubernetes & GitHub: Part 2

How To Configure Jenkins To Build Your CI CD Pipeline?

medium.com

Signing Off Message:

> “ When people within an organization, start collaborating with the intent to help each other the fabric for a stronger pipeline gets established which is bound to create success and happiness in the long run. DevOps is one such powerful tool to builds such a long-lasting culture ”

Thanks a ton for being with me and see you soon in the next part of CI/CD with Jenkins...

Dev Ops Continuous Integration Cloud Computing Kubernetes Microservices