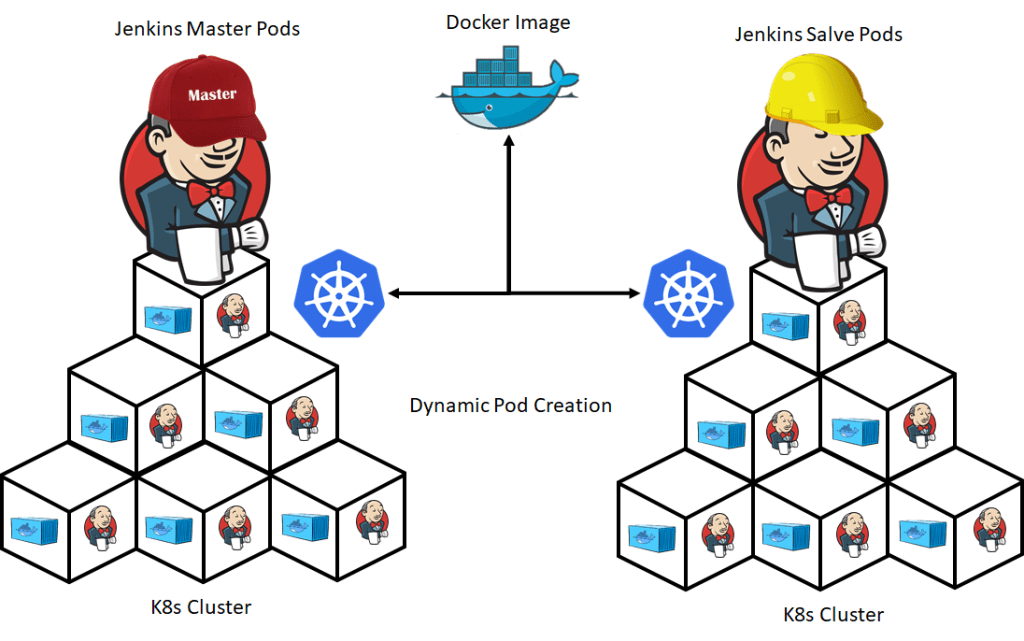
To Create Dynamically Scalable Jenkins Slaves with Docker and Kubernetes:

Create dynamically Scalable Jenkins slaves with Docker and Kubernetes

So, first, we are going to create Jenkins master from Kubernetes itself. Then we are going to create the Jenkins template which can be created from Jenkins master’s UI.

To implement the above idea, we are going to create the following steps to implement the dynamically Scalable Jenkins slaves.

1. Jenkins Instance service
2. Jenkins Service Discovery Service
3. Creating Docker Image with required plugins
4. Jenkins Deployment
5. Configuring Jenkins Slaves
6. Test The configuration – Create Jenkins jobs and run all together

As discussed, we are going to use Kubernetes to create Jenkins instance dynamically and provision it to The Jenkins server. In order to provision Jenkins instance, we need Jenkins Kubernetes service to create the Jenkins instance in Kubernetes pods.

kubectl create -f jenkins-instance-service.yml

Now, we are going to create another service for service discovery which will help us to route the external traffic to Jenkins-instance clusters. Just like Jenkins-instance service, we must create .yml file for Kubernetes to create the service discovery service. Copy the code and create jenkins-service-discovery-service.yml

Just like Jenkins-instance service, pass the kubectl command to create the service discovery service.

kubectl create -f jenkins-service-discovery-service.yml

To create Kubernetes clusters, Docker image is very important. So, we are going to create the docker image and push it to your docker registry so that we can use it when we deploy the application in Kubernetes. Use the below Dockerfile to create the docker image.

from jenkins/jenkins:lts

COPY install-plugin.sh /usr/local/bin/install-plugin.sh

RUN /usr/local/bin/install-plugin.sh ssh-slaves

RUN /usr/local/bin/install-plugins.sh kubernetes

Then, from the directory of this Dockerfile, run below docker build command to create the docker image

docker build -t digitalvarys/Jenkins

docker push <Image ID> digitalvarys/Jenkins

## **4. Jenkins-instance Deployment**

Now we need to deploy Jenkins-instance so that it will be available as pods. Then, use the below .yml code to create Kubernetes deployment script (jenkins-instance-deploy.yml).

To deploy the jenkins-instance, run the below kubectl command.

kubectl create -f jenkins-instance-deploy.yml

Then, this will create the pods and check whether all the pods are up and running by passing following command

kubectl get pods –namespace jenkins-instance

## **5. Configuring Jenkins Slaves**

Now, We need to configure the Jenkins slaves by configuring in the Jenkins UI itself. Follow the below steps to configure.

Go to **Configure System** from **Manage Jenkins** option and select **Cloud**section

From here, “**Add New Cloud**” and select “**Kubernetes**” the enter the required details of Kubernetes.

The, to get the details that need to be filled in Kubernetes form, get it by passing following command

kubectl describe pod –namespace jenkins-slave

## **6. Test The configuration – Create Jenkins jobs and run all together**

To test the above configuration, create two jobs and keep it running for a while. Which mean, create Jenkins jobs such a way that both jobs should run at the same time. Then, once the jobs are started running, check the pod status by running the following command.

kubectl get pods --namespace jenkins-instance