**Installing Kubernetes**

1. Check Host ip is listed before initiating kubernetes . The host ip will be used as adverstising ip address and would act as manager node.

**\*\* host ip should be static ip address**

hostname -I

*From <*[*https://askubuntu.com/questions/430853/how-do-i-find-my-internal-ip-address*](https://askubuntu.com/questions/430853/how-do-i-find-my-internal-ip-address)*>*

1. If the hostname is not listed , add the static IP as below

sudo nano /etc/netplan/50-cloud-init.yaml

*From <*[*https://websiteforstudents.com/configure-static-ip-addresses-on-ubuntu-18-04-beta/*](https://websiteforstudents.com/configure-static-ip-addresses-on-ubuntu-18-04-beta/)*>*

1. sudo netplan apply (for making change to static ip)

*From <*[*https://websiteforstudents.com/configure-static-ip-addresses-on-ubuntu-18-04-beta/*](https://websiteforstudents.com/configure-static-ip-addresses-on-ubuntu-18-04-beta/)*>*

1. Install Kubelet,kubeadm,kubectl in all nodes

<https://kubernetes.io/docs/setup/independent/install-kubeadm/>

5. On the Master node Init the kubeadm with the below command

kubeadm init pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=13.71.xx.xx(your ip adddress)

Pod-network-cdir is Pod network cluster : You can choose one ( Flannel ,Calico , Canal etc…)

<https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>

This is the range of IP address which will get assigned to pods internally

1. You will get a token as output which you can use to join other nodes .

Example :

"kubeadm join 13.71.82.75:6443 --token iih006.sdd1rmxndensm727 --discovery-token-ca-cert-hash sha256:e4a55cec61f3bf181c526328c33855ee73c101dd83dbb040f070cdffbfa0777a "

Use the above token to join other nodes.

1. Apply Pod cluster network command in the Master node . You can use any of the pod cluster network (Flannnel ,weavenet,calico etc…)

I have used Weavenet since at the time of writing flannel had issuse with ubuntu 18.0

kubectl apply -f "[https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')](https://cloud.weave.works/k8s/net?k8s-version=$(kubectl%20version%20|%20base64%20|%20tr%20-d%20'\n'))"

Verify the installation

kubectl get pods --all-namespaces

1. Assuming the master node is up and we have joined all the other minion nodes and we have applied the pod cluster network , we should be able to see that the master and minion are connected and are set up for deployment we can verify the same by using the below command

kubectl get nodes

13.71 .82.75 
kmast e r 
get 
srArus 
Ready 
Readv 
nodes 
ROLES 
master 
< none > 
AGE 
16m 
16m 
VERSION 
v1.13.3 
VI. 13.2 

**Kubernetes Basic terms :**

Pods : Smallest unit of deployment in Kuberenetes ,similar to containers in docker.

Deployment : A deployment controller where replicaset & pods are associated for deployment

Service : Connect / Expose pods to external applications.

**Configuring Kubernetes(K8) for Two - Org Fabric :**

Network Specification of Hyperledger Fabric : We have two organizations and an orderer service. i.e two peer of different organizations and 1 orderer service which uses Solo.

**What would we be creating with K8 ?**

We will leverage Kubernetes to create replicas of Orderer (3 replica service for orderer) , Peers (2 replicas of peer in each org )

**Note on NFS Service :**

I am using NFS to share files that I require for deployment in other minions nodes , this is necessary because pods run on different minions and we got to ship the necessary files for deployment in the respective nodes. NFS is similar to network sharing in Windows .

**Configuring NFS Service :**

1. Make sure you have mount-nfs installed in both master & minion (<https://www.tecmint.com/how-to-setup-nfs-server-in-linux/>)
2. Specify the path which you want to share in /etc/exports i.e sudo vim /etc/exports
3. Restart with exportfs -ra command**.**

(rw, sync, no subtree check) 
sync no subtree check) 

**Setting up Orderer service in K8 :**

**Creating Deployment for Orderer Service :**

In the below deployment configuration of Kubernetes , I have specified the replica set to 3 ,which means if one orderer fails the other pod would be brought up automatically by Kubernetes . Modify the configuration accordingly to your host path and save it as "Orderer-deployment.yml"

apiVersion: apps/v1

kind: Deployment

metadata:

name: orderertwoorg

spec:

replicas: 3

selector:

matchLabels:

app: orderertwoorg

template:

metadata:

labels:

app: orderertwoorg

spec:

hostname: orderer-twoorg-com

volumes:

- name: nfs-volume-twoorg

nfs:

# URL for the NFS server

server: 13.71.82.75 # Change this!

path: /home/u1048653/twoorgfabric/

containers:

- args:

- orderer

env:

- name: ORDERER\_GENERAL\_GENESISFILE

value: /home/u1048653/twoorgfabric/artifacts/channel/genesis.block

- name: ORDERER\_GENERAL\_GENESISMETHOD

value: file

- name: ORDERER\_GENERAL\_LEDGERTYPE

value: json

- name: ORDERER\_GENERAL\_LISTENADDRESS

value: 0.0.0.0

- name: ORDERER\_GENERAL\_LISTENPORT

value: "7050"

- name: ORDERER\_GENERAL\_LOCALMSPDIR

value: "/home/u1048653/twoorgfabric/artifacts/crypto/crypto-config/ordererOrganizations/twoorg.com/orderers/orderer.twoorg.com/msp"

- name: ORDERER\_GENERAL\_LOCALMSPID

value: OrdererMSP

- name: ORDERER\_GENERAL\_LOGLEVEL

value: debug

- name: ORDERER\_GENERAL\_TLS\_CERTIFICATE

value: /home/u1048653/twoorgfabric/artifacts/crypto/crypto-config/ordererOrganizations/twoorg.com/orderers/orderer.twoorg.com/tls/server.crt

- name: ORDERER\_GENERAL\_TLS\_ENABLED

value: "false"

- name: ORDERER\_GENERAL\_TLS\_PRIVATEKEY

value: /home/u1048653/twoorgfabric/artifacts/crypto/crypto-config/ordererOrganizations/twoorg.com/orderers/orderer.twoorg.com/tls/server.key

- name: ORDERER\_GENERAL\_TLS\_ROOTCAS

value: '[/home/u1048653/twoorgfabric/artifacts/crypto/crypto-config/ordererOrganizations/twoorg.com/orderers/orderer.twoorg.com/tls/ca.crt]'

image: hyperledger/fabric-orderer

name: orderertwoorg

ports:

- name: orderer-port

containerPort: 7050

resources: {}

volumeMounts:

- mountPath: "/home/u1048653/twoorgfabric/"

name: nfs-volume-twoorg

# - mountPath: "/home/u1048653/twoorgfabric/artifacts/channel"

# name: orderer-insurance-com-claim0

# - mountPath: "/home/u1048653/twoorgfabric/artifacts/crypto/crypto-config/ordererOrganizations/twoorg.com/orderers/orderer.twoorg.com"

# name: orderer-insurance-com-claim1

workingDir: /opt/gopath/src/github.com/hyperledger/fabric/orderer

restartPolicy: Always

**Running Deployment Orderer :**

**To run the deployment :**

**Kubectl create -f** orderer-deployment.yml

**To verify the deployment :**

**kubectl get pods**

**To delete the deployment :**

**kubectl delete -f** orderer-deployment.yml

— / twoorgfabric$ kubectl 
N AEF. 
b 4b 68 f —ctrsk 
-7647675ddB-fsbcj 
couch db O —peer ff6757767 —pvs6h 
986d4656f-45hrz 
orderertwoorg-6B 6dfc7B4B 
-rSB7g 
orderertwoorg— 68 6dfc7848 —wnwfb 
orde re rtwccrg-686dfc7848 
-z465v 
get 
pods 
READY 
1/1 
1/1 
1/1 
1/1 
STATUS 
Runni ng 
Running 
Runni ng 
Runni ng 
Running 
Runni ng 
Running 
RESTARTS 
AGE 
12d 
13d 
12d 
5d22h 
20h 
20h 
20b 

**Create Service for Orderer :**

The orderer port(7050) we defined cannot be accessed by other peers or pods unless we map the respective port with Nodeport using Kubernetes service. Ideally we will be mapping 7050 to different port for other containers to access it.

**Service Configuration for Orderer** :

apiVersion: v1

kind: Service

metadata:

name: orderertwoorg

spec:

selector:

app: orderertwoorg

ports:

- protocol: TCP

targetPort: 7050

port: 7050

nodePort: 32001

name: orderer-port

type: NodePort

Let us save this Service confiuration in orderer-service.yml

**Running Service for Orderer :**

**kubectl create -f orderer-service.yml**

get services 
10.102.181.91 
10.103.170.121 
10.106.132.185 
,rderertwoorg 
111.101.3.215 
• / twoorgt 
—servlce 
r Ic/artltacts 
TYPE 
Nodeport 
NodePort 
Cluster1P 
ClusterIP 
NodePort 
NodePort 
Port 
CLUSTER- IP 
10.99.219.160 
10.104.122.251 
96. O. 1 
EXTERNAL-IP 
<none> 
<none> 
< nano > 
PORT (S) 
5984 : 30005/TCP 
: 31005/TCP 
443/TCP 
2049/TCP, III/UDP 
705 
32001/TCP 
14d 
12d 
16d 
43h 
12d 
253 
12ci 

**Peer Configurations**