**Prerequisites All Nodes**

* Static IP on servers
* /etc/hosts file is filled up

192.168.1.10 master01.yaydog.local master01

192.168.1.11 master02.yaydog.local master02

192.168.1.12 master03.yaydog.local master03

192.168.1.13 worker01.yaydog.local worker01

192.168.1.14 worker02.yaydog.local worker02

192.168.1.15 lb.yaydog.local

192.168.1.16 nfs-server.yaydog.local

192.168.1.10 k8s.project.io

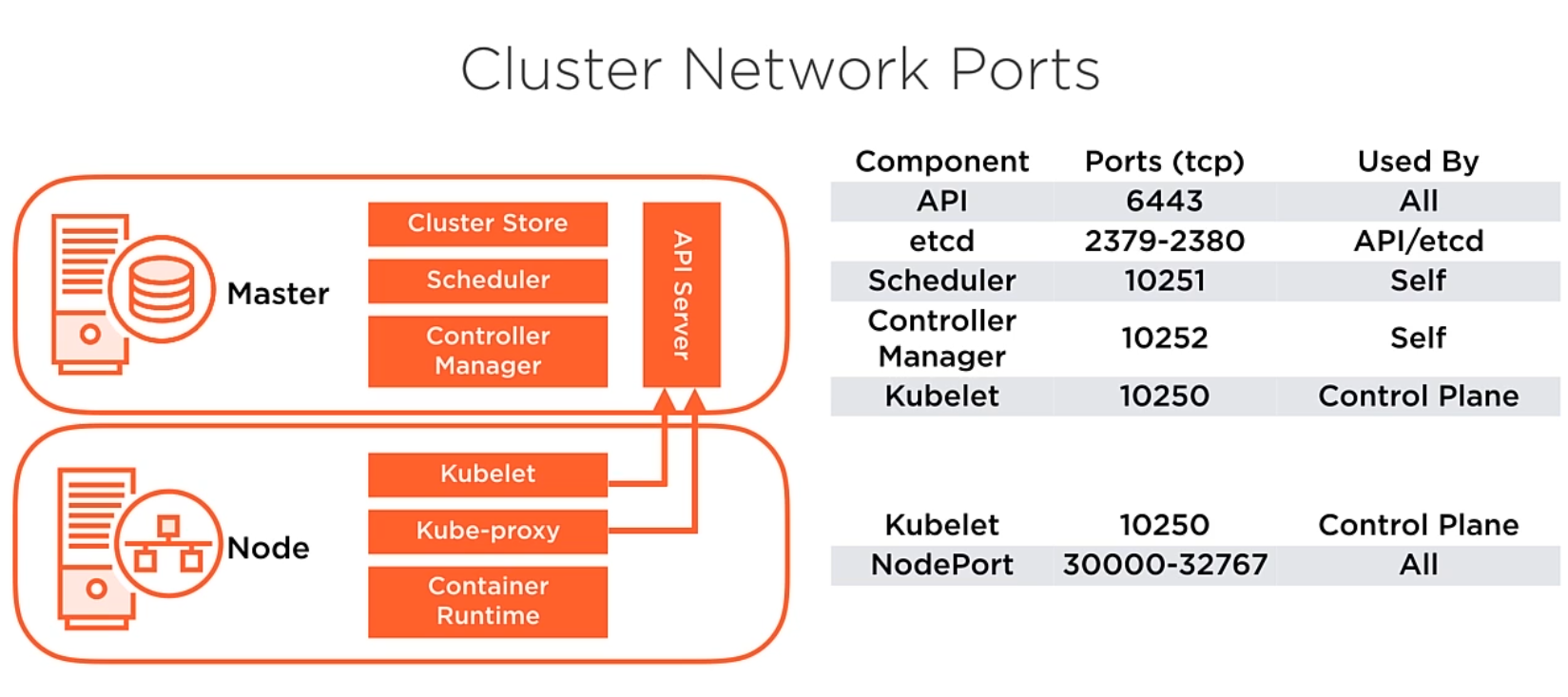
* Firewall disable

systemctl stop firewalld

systemctl disable firewalld

(master node: ports: 6443/tcp, 2379-2380/tcp, 10250/tcp, 10251/tcp, 10252/tcp, 10255/tcp)

(worker node: ports: 10251/tcp, 10255/tcp)



* Disable selinux
* Gerekli yardımcı paketlerin kurulması

yum install epel-release -y

yum install wget ksh rpcbind traceroute gzip unzip bzip2 gedit xterm unzip vim lsof lvm2 tree gnome-system-log nmap at ntp ntpdate rsync grsync audit audit-libs yum-utils xdpyinfo xorg-x11-apps xorg-x11-xauth xorg-x11-fonts-\* xorg-x11-font-utils xorg-x11-fonts-Type1 finger system-config-keyboard system-config-language curl rsync gawk sysstat htop gnome-system-monitor NetworkManager net-tools glogg bind-utils bash-completion tcpdump git yum-plugin-versionlock yum-utils device-mapper-persistent-data lvm2 telnet nfs-utils nfs-utils-lib openssl -y

* NTP Time Synch

systemctl enable ntpd

vi /etc/ntp.conf

server server 0.tr.pool.ntp.org

server server 1.tr.pool.ntp.org

server server 2.tr.pool.ntp.org

server server 3.tr.pool.ntp.org

systemctl start ntpd

* yum update

* useradd k8s -d /home/k8s -m -G wheel
* disable swap memory

sed -i '/swap/d' /etc/fstab

swapoff -a

reboot

* vi /etc/sysctl.conf

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

(check example)

sysctl -a | grep net.bridge.bridge-nf-call-ip6tables

* vi /etc/sysconfig/modules/**overlay**.modules

#!/bin/sh

exec /sbin/modprobe nfs >/dev/null 2>&1

chmod +x /etc/sysconfig/modules/overlay.modules

vi /etc/sysconfig/modules/**br\_netfilter**.modules

#!/bin/sh

exec /sbin/modprobe nfs >/dev/null 2>&1

chmod +x /etc/sysconfig/modules/br\_netfilter.modules

(check)

lsmod | grep overlay

lsmod | grep br\_netfilter

* sysctl --system

(installing kubernetes)

* vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

yum install -y kubelet kubeadm kubectl

yum versionlock kubelet kubeadm kubectl

systemctl enable kubelet

systemctl daemon-reload

(check versions)

kubectl version --client

kubeadm version

(installing docker)

* yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>

yum install containerd.io-1.2.13 docker-ce-19.03.11 docker-ce-cli-19.03.11

mkdir /etc/docker

vi /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2",

"storage-opts": [

"overlay2.override\_kernel\_check=true"

]

}

mkdir -p /etc/systemd/system/docker.service.d

systemctl daemon-reload

systemctl enable docker

* systemctl start kubelet docker

(bash autocomplete for kubectl commands)

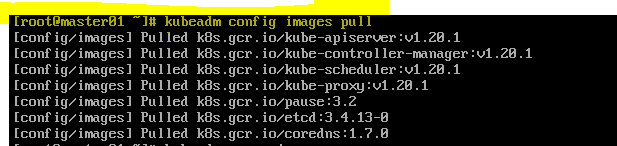
yum install bash-completion bash-completion-extras

echo "source <(kubectl completion bash)" >> /root/.bashrc

source /root/.bashrc

**ON MASTER01**

kubeadm config images pull



kubeadm init --pod-network-cidr=10.1.0.0/16 --control-plane-endpoint=k8s.project.io

mkdir -p $HOME/.kube

cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config

mkdir -p /home/k8s/.kube

cp -i /etc/kubernetes/admin.conf /home/k8s/.kube/config

chown -R k8s:k8s /home/k8s/.kube/config

kubeadm init phase upload-certs --upload-certs (for other masters)

**1cff0773477f20423628adf30213c8e7bead84aa1293b17b901ad791b8b9077e**

kubeadm token create --print-join-command (for nodes)



Certificate location: /etc/kubernetes/pki

**ON MASTER02,WORKER03**

kubeadm join k8s.project.io:6443 --token 5i2h5q.xmzljftl5mmiq8mb --discovery-token-ca-cert-hash **sha256:19295551f28d9580351dba027ed2c7f8d1b3ea979dd385ba403004aa57dd1615** --control-plane --certificate-key **1cff0773477f20423628adf30213c8e7bead84aa1293b17b901ad791b8b9077e**

(they will be control plane as well)

mkdir -p $HOME/.kube

cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config

mkdir -p /home/k8s/.kube

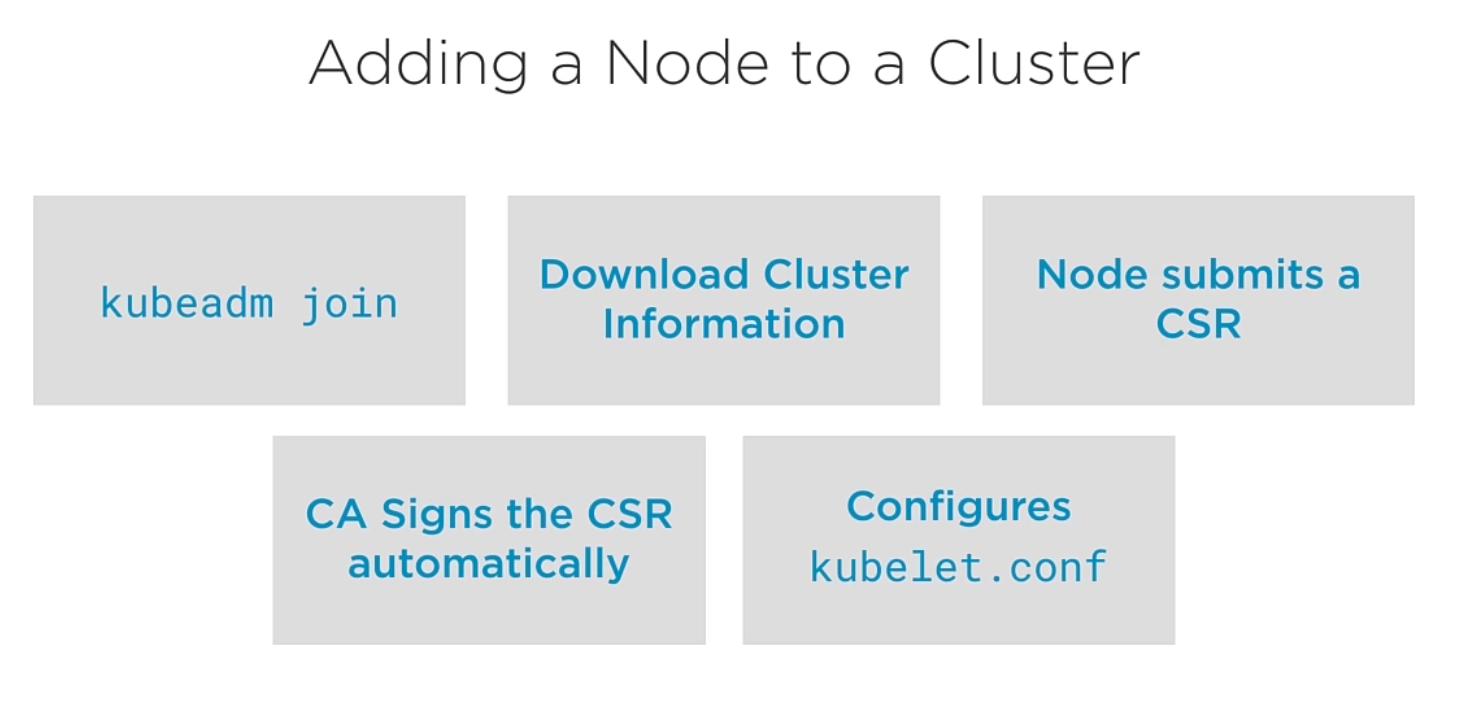
cp -i /etc/kubernetes/admin.conf /home/k8s/.kube/config

chown -R k8s:k8s /home/k8s/.kube/config

**ON WORKER NODES**

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join k8s.project.io:6443 --token 5i2h5q.xmzljftl5mmiq8mb --discovery-token-ca-cert-hash **sha256:19295551f28d9580351dba027ed2c7f8d1b3ea979dd385ba403004aa57dd1615**



**ON MASTER01**

kubectl label node worker01.yaydog.local kubernetes.io/role=worker --overwrite

kubectl label node worker02.yaydog.local kubernetes.io/role=worker --overwrite

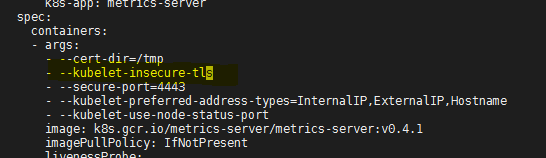
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

kubectl get nodes (check all nodes appear and ready)

kubectl get pods --all-namespaces -o wide

**(metric server installation)**

wget <https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.4.1/components.yaml>



kubectl apply -f components.yaml

kubectl get pods --all-namespaces (check metric server is running)

kubectl top nodes

kubectl top pods -A

kubectl get --raw /apis/metrics.k8s.io

(checking)

**(dashboard installation)**

wget <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.4/aio/deploy/recommended.yaml>

kubectl apply -f recommended.yaml

cat <<EOF | kubectl apply -f -

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

EOF

cat <<EOF | kubectl apply -f -

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard

EOF

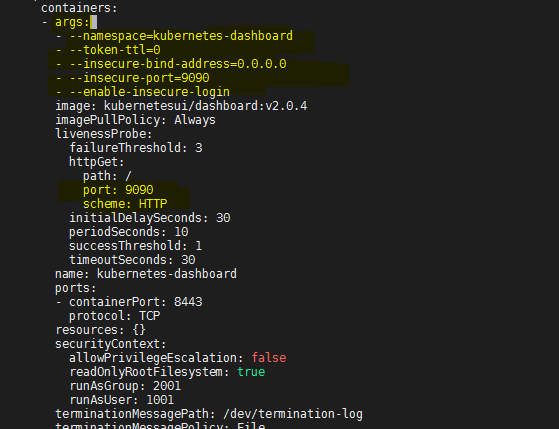
echo $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')

**admin-user-token-tv5gc**

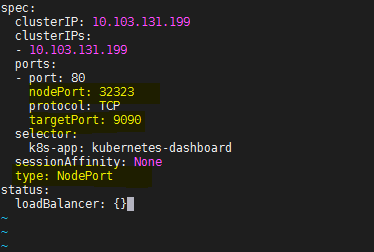
kubectl -n kubernetes-dashboard describe secret **admin-user-token-tv5gc**

(dashboard a giriş yaparken kullanılır)

kubectl edit deployment kubernetes-dashboard -n kubernetes-dashboard



kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard



kubectl get svc -n kubernetes-dashboard

<http://192.168.1.10:32323>

**(creating read-only user for dashboard)**

vi dashboard-read-only-user.yaml

*apiVersion: rbac.authorization.k8s.io/v1*

*kind: ClusterRole*

*metadata:*

*annotations:*

*rbac.authorization.kubernetes.io/autoupdate: "true"*

*labels:*

*name: read-only-clusterrole*

*namespace: default*

*rules:*

*- apiGroups:*

*- ""*

*resources: ["\*"]*

*verbs:*

*- get*

*- list*

*- watch*

*- apiGroups:*

*- extensions*

*resources: ["\*"]*

*verbs:*

*- get*

*- list*

*- watch*

*- apiGroups:*

*- apps*

*resources: ["\*"]*

*verbs:*

*- get*

*- list*

*- watch*

*---*

*apiVersion: rbac.authorization.k8s.io/v1*

*kind: ClusterRoleBinding*

*metadata:*

*name: read-only-binding*

*roleRef:*

*kind: ClusterRole*

*name: read-only-clusterrole*

*apiGroup: rbac.authorization.k8s.io*

*subjects:*

*- kind: ServiceAccount*

*name: read-only-user*

*namespace: kubernetes-dashboard*

kubectl apply -f dashboard-read-only-user.yaml

echo $(kubectl -n kubernetes-dashboard get secret | grep read-only-user | awk '{print $1}')

kubectl -n kubernetes-dashboard describe secret **read-only-user-token-kqwkm**

**PERSISTENT STORAGE**

**MASTER01**

systemctl enable rpcbind

systemctl start rpcbind

(creating persistent volume)

vim nfs-pv-app1.yml

*apiVersion: v1*

*kind: PersistentVolume*

*metadata:*

*name: nfs-pv-app1*

*spec:*

*capacity:*

*storage: 10Gi*

*volumeMode: Filesystem*

*accessModes:*

*- ReadWriteMany*

*persistentVolumeReclaimPolicy: Recycle*

*storageClassName: nfs*

*mountOptions:*

*- hard*

*- nfsvers=4.1*

*nfs:*

*path: /k8sdata/app1*

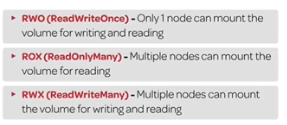
*server: nfs-server.yaydog.local*

kubectl create -f nfs-pv-app1.yml

kubectl get pv nfs-pv-app1

kubectl describe pv nfs-pv-app1

(persistent volume Access modes)



persistentVolumeReclaimPolicy:

**Recycle**:

**Retain**: if a user deletes a PersistentVolumeClaim, the corresponding PersistentVolume is not be deleted

**Delete**: dynamically provisioned volume is automatically deleted when a user deletes the corresponding PersistentVolumeClaim

(creating persistent volume claim)

vi nfs-pvc-app1.yml

*apiVersion: v1*

*kind: PersistentVolumeClaim*

*metadata:*

*name: nfs-pvc-app1*

*spec:*

*storageClassName: nfs*

*accessModes:*

*- ReadWriteMany*

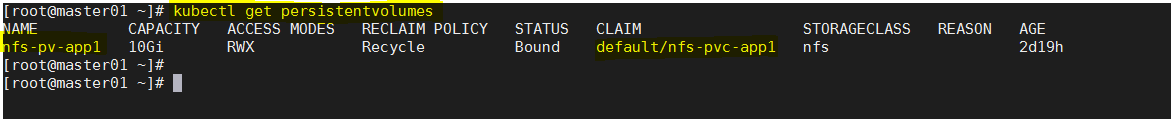
*resources:*

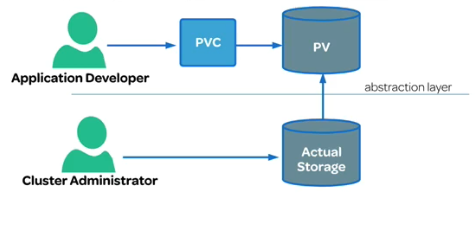
*requests:*

*storage: 10Gi*

kubectl create -f nfs-pvc-app1.yml

kubectl get pvc nfs-pvc-app1





(first delete pod and then pvc)

**ALTERNATIVE DASHBOARD (K8DASH)**

wget <https://raw.githubusercontent.com/indeedeng/k8dash/master/kubernetes-k8dash.yaml>

*kind: Deployment*

*apiVersion: apps/v1*

*metadata:*

*name: k8dash*

*namespace: kube-system*

*spec:*

*replicas: 1*

*selector:*

*matchLabels:*

*k8s-app: k8dash*

*template:*

*metadata:*

*labels:*

*k8s-app: k8dash*

*spec:*

*containers:*

*- name: k8dash*

*image: herbrandson/k8dash:latest*

*ports:*

*- containerPort: 4654*

*livenessProbe:*

*httpGet:*

*scheme: HTTP*

*path: /*

*port: 4654*

*initialDelaySeconds: 30*

*timeoutSeconds: 30*

*nodeSelector:*

*'beta.kubernetes.io/os': linux*

*---*

*kind: Service*

*apiVersion: v1*

*metadata:*

*name: k8dash*

*namespace: kube-system*

*spec:*

*type: NodePort*

*ports:*

*- port: 4654*

*targetPort: 4654*

*nodePort: 31971*

*selector:*

*k8s-app: k8dash*

kubectl create serviceaccount k8dash-sa

kubectl create clusterrolebinding k8dash-sa --clusterrole=cluster-admin --serviceaccount=default:k8dash-sa

kubectl get secrets

kubectl describe secret k8dash-sa-token-xxxxx

**ALTERNATIVE DASHBOARD (OCTANT)**

wget <https://github.com/vmware-tanzu/octant/releases/download/v0.16.3/octant_0.16.3_Linux-64bit.tar.gz>

tar -xzvf octant\_0.16.3\_Linux-64bit.tar.gz

cd octant\_0.16.3\_Linux-64bit/

/root/octant\_0.16.3\_Linux-64bit/octant --listener-addr 192.168.1.10:8080 --accepted-hosts=192.168.1.9 --disable-open-browser=true >> /dev/null 2>&1 &