

INSTALACIÓN DE DOCKER

Pasos para configuración e instalación de Docker

1- Instalamos las dependencias.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

2- Agregamos el repositorio de Docker a CentOS

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

3- Actualizamos e instalamos Docker en CentOS

```
yum update -y && yum install -y containerd.io-1.2.10 docker-ce-19.03.4 docker-ce-cli-19.03.4
```

4- Creamos el directorio donde irá el Demonio

```
mkdir /etc/docker
```

5- Agregar y configurar el Demonio de extensión .json

```
cat > /etc/docker/daemon.json <<EOF
```

```
{  
  "exec-opts": ["native.cgroupdriver=systemd"],  
  "log-driver": "json-file",  
  "log-opts": {  
    "max-size": "100m"  
  },  
  "storage-driver": "overlay2",  
  "storage-opts": [  
    "overlay2.override_kernel_check=true"  
  ]  
}  
EOF
```

6- Se crea una capeta más para Docker Service

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

7- Se carga nuevamente el Demonio

```
systemctl daemon-reload
```

8- Se reinicia y habilita Docker

```
systemctl restart docker
```

```
systemctl enable docker
```

Instalación de Kubernetes

Pasos para configuración e instalación de Kubernetes

9- Se agrega el repositorio de Kubernetes y se añade la llave GPG

```
cat <<EOF > /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
```

```
EOF
```

10- Se instala Kubeadm

```
yum install -y kubelet kubeadm kubectl
```

11- Luego se habilita e inicia Kubelet

```
systemctl enable kubelet && systemctl start kubelet
```

12- Se crean los dos maestros

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

13- Se deshabilita el intercambio de archivos (swapping)

```
swapoff -a
```

Instalación de NGINX

Instalación y configuración de nginx para balancear las cargas:

```
apt update && apt -y upgrade
```

```
/*Instalamos NGINX*/ apt install -y nginx
```

```
/*Habilitamos NGINX*/ systemctl enable nginx
```

```
/*Se revisa el estado de NGINX*/ systemctl status nginx
```

```
/*Se inserta servidor y cliente SSH*/ apt install -y openssh-server openssh-client
```

```
/*Se habilita de nuevo NGINX*/ systemctl enable nginx
```

```
/*y se revisa su estado*/ systemctl status nginx
```

```
/*Abrir el archivo de ssh_config*/ nano /etc/ssh/sshd_config
```

14- Añadimos la configuración de Kubernetes

```
cat <<EOF | sudo tee /etc/nginx/tcpconfig.l/kubernetes.conf
```

```
stream {
```

```
    upstream kubernetes {
```

```
        server <controller 0 private ip>:6443;
```

```
        server <controller 1 private ip>:6443;
```

```
    }
```

```
server {
```

```
    listen 6443;
```

```
listen 443;

proxy_pass kubernetes;

}

}

EOF
```

/*IMPORTANTE SABER EL NO. DE IP DE LOS MAESTROS, SE REMUEVEN LOS <>, UNICAMENTE IP Y PUERTO DE ESCUCHA DE KUBE-APISERVER*/

15- Se abre el archivo de configuración de Kubernetes

nano /etc/nginx/tcpconfig.l/kubernetes.conf

16- Se reinicia NGINX

nginx -s reload

YA UBICADOS EN EL PRIMER MAESTRO, SE REALIZA:

/*IMPORTANTE TENER SWAP DESHABILITADO, DE LO CONTRARIO PUEDE NO FUNCIONAR*/

17- Se levanta el maestro:

kubeadm init --control-plane-endpoint "192.168.0.8:6443" --upload-certs

[CON LA IP DEL LOAD_BALANCER Y EL PUERTO DE KUBE-APISERVER]

```
root@master2:~# kubeadm init --control-plane-endpoint "192.168.0.8:6443" --upload-certs
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 192.168.0.8:6443 --token w29kud.im36bvu69kwcem4o \
--discovery-token-ca-cert-hash sha256:1d0ee4ed797726575c9b743644cacb3be55cc0e265956eb02fc49a7f36ad82fd \
--control-plane --certificate-key abc2d917e914ab397659e345b5c0d75668e055d471d6aa526174f24397bed07

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 192.168.0.8:6443 --token w29kud.im36bvu69kwcem4o \
--discovery-token-ca-cert-hash sha256:1d0ee4ed797726575c9b743644cacb3be55cc0e265956eb02fc49a7f36ad82fd
[root@master2 ~]# mkdir -p $HOME/.kube
[root@master2 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master2 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@master2 ~]# kubeadm join 192.168.0.8:6443 --token w29kud.im36bvu69kwcem4o --discovery-token-ca-cert-hash sha256:1d0ee4ed797726575c9b743644cacb3be55cc0e265956eb02fc49a7f36ad82fd --control-plane --certificate-key abc2d917e914ab397659e345b5c0d75668e055d471d6aa526174f24397bed07
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR] DirAvailable--etc-kubernetes-manifests: /etc/kubernetes/manifests is not empty
```

[PARA COMPLEMENTAR LA CONFIGURACIÓN DEL KLUSTER, ES NECESARIO EJECTURAR ESTE CÓDIGO DIRECTAMENTE EN EL PRIMER MAESTRO]:

```
mkdir -p $HOME/.kube  
  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

[ESTO GENERARÁ UN TOKEN, ESTE SE DEBE EJECUTAR EN EL SEGUNDO DE NUESTROS MAESTROS]:

```
kubeadm join 192.168.0.8:6443 --token wz9kud.im36bvu69kwcem4o --discovery-token-ca-cert-hash sha256:1d0ee4ed797726575c9b743644cacb3be55cc0e265956eb02fc49a7f36ad82fd --control-plane --certificate-key abc2d917e914ab397659e345b5c0d75668e8055d471d6aa526174f24397bed07
```

[SE COMPLEMENTA EL SEGUNDO MAESTRO CON EL SIGUIENTE CÓDIGO]:

```
mkdir -p $HOME/.kube  
  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

[PARA LOS ESCLAVOS, SE EJECUTA EL MISMO TOKEN QUE SE OBTUVO Y SE EJECUTÓ EN EL SEGUNDO MAESTRO]:

```
kubeadm join 192.168.0.8:6443 --token wz9kud.im36bvu69kwcem4o --discovery-token-ca-cert-hash sha256:1d0ee4ed797726575c9b743644cacb3be55cc0e265956eb02fc49a7f36ad82fd
```

[LUEGO NOS TRASLADAMOS AL PRIMER MAESTRO, AQUÍ SE DEBE APLICAR EL CNI DE KUBERNETES]:

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

[SE EJECUTA EL SIGUIENTE SCRIPT PARA VERIFICAR EL ESTADO DE MAESTROS Y NODOS]:

```
kubectrl get nodes
```

[SI SE DESEA VER EL ESTADO DE NUESTROS CONTENEDORES EN TIEMPO REAL SE EJECUTA EL SIGUIENTE CÓDIGO]:

```
kubectl get pod -n kube-system -w
```

[LUEGO VERIFICAMOS QUE LOS PODS SE ESTÉN EJECUTANDO Y CORRIENDO CORRECTAMENTE]:

```
kubectl get pod -n kube-system
```

CONFIGURACIÓN DEL DASHBOARD

[IMPLEMENTAMOS EL SERVIDOR DE MÉTRICAS]:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/components.yaml
```

[POSTERIORMENTE, DEBEMOS COMPROBAR QUE LA IMPLEMENTACIÓN DE METRICS-SERVER ESTÁ EJECUTANDO LA CANTIDAD DESEADA DE PODS]:

```
kubectl get deployment metrics-server -n kube-system
```

[ADICIONAL, SE DEBE DE IMPLEMENTAR EL PANEL DE KUBERNETES, DE ESTA FORMA]:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta8/aio/deploy/recommended.yaml
```

[SE DEBE DE CREAR UNA CUENTA DE SERVICIO EKS-ADMIN Y UN ENLACE DE ROL DE CLÚSTER. TAMBIÉN CREAR UN ARCHIVO LLAMADO EKS-ADMIN-SERVICE-ACCOUNT.YAML. ESTE MANIFIESTO DEFINE UNA CUENTA DE SERVICIO Y UN ENLACE DE ROL DE CLÚSTER LLAMADO EKS-ADMIN.]

[AGREGAMOS LA CONFIGURACIÓN AL ARCHIVO DASHBOARD-ADMIN.YML APIVERSION: V1]:

```
kind: ServiceAccount
```

```
metadata:
```

```
  name: eks-admin
```

namespace: kube-system

[AGREGAMOS LA CONFIGURACIÓN AL ARCHIVO ADMIN-ROLE-BINDING.YML]:

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: eks-admin

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: eks-admin

namespace: kube-system

[NECESITAMOS APLICAR LA CUENTA DE SERVICIO Y EL ENLACE DE ROL DE CLÚSTER A NUESTRO CLÚSTER]:

kubectl apply -f eks-admin-service-account.yaml

[POSTERIORMENTE NOS CONECTAMOS AL PANEL DE KUBERNETES]:

kubectl -n kube-system describe secret \$(kubectl -n kube-system get secret | grep eks-admin | awk '{print \$1}')

[SE INICIA KUBECTL PROXY].

kubectl proxy --address="192.168.0.5" -p 8001 --accept-hosts='^*\$'

[LUEGO, SE INGRESA AL ENLACE EN DONDE SE VISUALIZARÁ NUESTRO DASHBOARD]:

<http://192.168.0.5:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>

LANZAMIENTO DE APLICACIÓN CON KUBERNETES

[PARA INICIAR, DEBEMOS CREAR NUESTRO DEPLOYMENT Y A LA VEZ BAJAMOS LA IMAGEN DE DOCKER HUB]:

```
kubectl create deployment application --image=leonel2708v03/nginxprueba:latest
```

[CON ESTE SCRIPT VEMOS EL ESTADO DE LOS PODS]:

```
kubectl get pods
```

[SI NECESITAMOS VER LOS DETALLES DE NUESTRO DESPLIEGUE, PODEMOS EJECUTAR]:

```
kubectl describe pod application
```

[PROCEDEMOS A CREAR EL ARCHIVO YML]:

```
nano nginx-apli.yml
```

[HABILITAMOS EL CONJUNTO DE CONTENEDORES DESCRITOS DESDE NUESTRO FICHERO]:

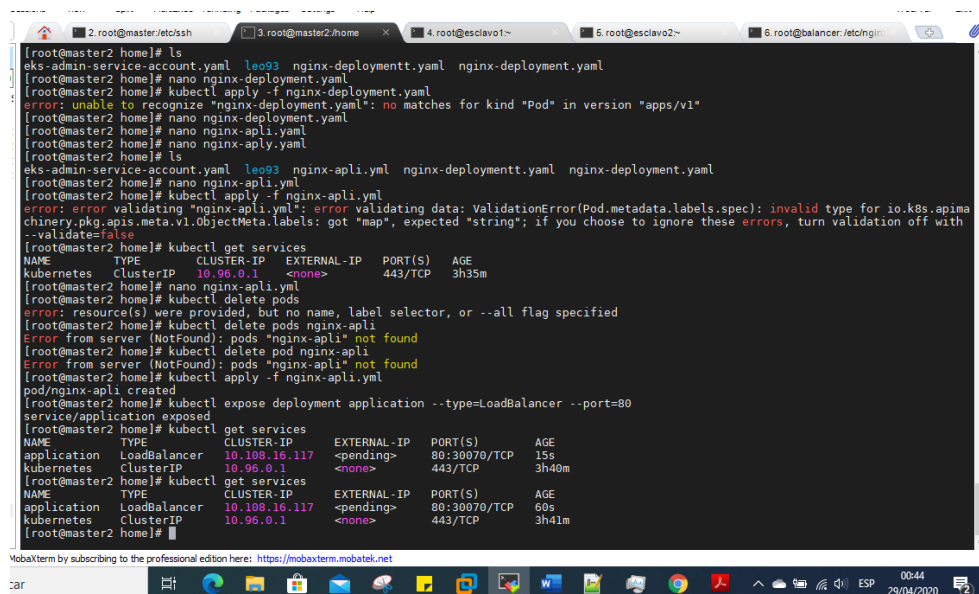
```
kubectl apply -f nginx-apli.yml
```

[HABILITAMOS EL SERVICIO EN EL PUERTO "80"]:

```
kubectl expose deployment application --type=LoadBalancer --port=80
```

[Y POSTERIORMENTE VEMOS LOS SERVICIOS QUE TENEMOS ACTIVOS]:

```
kubectl get services
```



```
[root@master2 home]# ls
eks-admin-service-account.yaml  leo93  nginx-deploymentt.yaml  nginx-deployment.yaml
[root@master2 home]# nano nginx-deployment.yaml
[root@master2 home]# kubectl apply -f nginx-deployment.yaml
error: unable to recognize "nginx-deployment.yaml": no matches for kind "Pod" in version "apps/v1"
[root@master2 home]# nano nginx-deployment.yaml
[root@master2 home]# nano nginx-apli.yml
[root@master2 home]# nano nginx-apli.yml
[root@master2 home]# ls
eks-admin-service-account.yaml  leo93  nginx-apli.yml  nginx-deploymentt.yaml  nginx-deployment.yaml
[root@master2 home]# nano nginx-apli.yml
[root@master2 home]# kubectl apply -f nginx-apli.yml
error: error validating "nginx-apli.yml": error validating data: ValidationError(Pod.metadata.labels.spec): invalid type for io.k8s.apimachinery.pkg.apis.meta.v1.ObjectMeta.labels: got "map", expected "string"; if you choose to ignore these errors, turn validation off with --validate=false
[root@master2 home]# kubectl get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1        <none>           443/TCP      3h35m
[root@master2 home]# nano nginx-apli.yml
[root@master2 home]# kubectl delete pods
error: resource(s) were provided, but no name, label selector, or --all flag specified
[root@master2 home]# kubectl delete pods nginx-apli
Error from server (NotFound): pods "nginx-apli" not found
[root@master2 home]# kubectl delete pod nginx-apli
Error from server (NotFound): pods "nginx-apli" not found
[root@master2 home]# kubectl apply -f nginx-apli.yml
pod/nginx-apli created
[root@master2 home]# kubectl expose deployment application --type=LoadBalancer --port=80
service/application exposed
[root@master2 home]# kubectl get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
application  LoadBalancer  10.108.16.117    <pending>        80:30070/TCP  15s
kubernetes  ClusterIP  10.96.0.1        <none>           443/TCP      3h40m
[root@master2 home]# kubectl get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
application  LoadBalancer  10.108.16.117    <pending>        80:30070/TCP  60s
kubernetes  ClusterIP  10.96.0.1        <none>           443/TCP      3h41m
[root@master2 home]#
```