Installing K8s locally with kubeadm

TL;DR Use Script (https://gitlab.f-in.io/finot/kubernetes/-/tree/master/1-easy-scripts), make one-node, done!

In order to start some prerequisites have to be in order. Please make sure you:

- Have access to our internal GitLab instance
- Have created a VM with either Debian 10/11 (Buster/Bullseye) or Ubuntu 20.04 (Focal) and 8+GB of RAM
- Preferably you know and work on Linux
- Have an active and relatively fast internet connection

First Steps: Getting the necessary tools

For easier development and debugging it is recommended that you install at your computer the LENS IDE for Kubernetes (https://k8slens.dev/). It will simplify your debugging procedures by a lot. If you prefer a no-frills terminal approach (as Arch Linux bros have preached) then please install kubectl (https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/). And yes, working on Linux for these kinds of procedures is a must. It is easier, faster and more secure than creating a whole K8s development environment on Windows (DOS included), OS X, FreeBSD (and OpenBSD) and Amiga OS.

Installing kubeadm with our script

Installing kubeadm is a hassle and we know that. This is why some Easy Install scripts have been created for Ubuntu (20.04) and Debian (10/11). If being a masochist is more your cup of tee, you can certainly install kubeadm by hand. You can find more info bellow. These tools were created for easy and uniform installation of the tool to multiple hosts/clusters.

⚠ **WARNING**: Make sure that you haven’t installed Docker or Kubeadm into the deployment system.
If you want to run the script with docker and/or kubeadm install, please comment out these sections from the script before running it.

ℹ **Link for the scripts**: https://gitlab.f-in.io/finot/kubernetes/-/tree/master/1-easy-scripts

Installing kubeadm by hand

First of all, we have to disable swap because of problems of Kubernetes with swap

```
swapoff -a
```

Then we have to enable netfilter so the interface networking can pass through to our Kubernetes environment.

```
# enable bridge netfilter
modprobe br_netfilter;
echo 'net.bridge.bridge-nf-call-iptables = 1' > /etc/sysctl.d/20-bridge-
nf.conf;
sysctl --system;
```

After that, some prerequisites have to be installed

```
# install tools for adding apt sources
apt-get update;
apt-get install -y \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg2;
```

Before installing docker, we have to edit the docker daemon in order to declare a k8s compatible storage driver, and to pass the systemd cgroup driver of our host system.

```
mkdir /etc/docker;
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": { "max-size": "100m" },
  "storage-driver": "overlay2"
}
EOF
```

After that, we are continuing on installing docker. Please make sure to change the disto and version tags in the repository link

```
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -;
echo 'deb [arch=amd64] https://download.docker.com/linux/[distro]
[version] stable' > /etc/apt/sources.list.d/docker.list;
apt-get update;
apt-get install -y --no-install-recommends docker-ce;
```

We are ready to install KubeAdmin in our system! We start by declaring the repository and installing the 3 tools that come together with kubeadm: kublet, kubeadm and kubectl.

⚠️ Please do NOT change the xenial tag in here, it is correct (source: `https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-using-native-package-management`)

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key
add -;
echo 'deb https://apt.kubernetes.io/ kubernetes-xenial main' > /etc/apt
/sources.list.d/kubernetes.list;
apt-get update;
apt-get install -y kubelet kubeadm kubectl;
```

Let’s initialize our k8s cluster. We have now to declare our internal k8s network range

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

If everything goes to plan, your first k8s cluster should be ready 🎉

In order to use and manage the k8s cluster, we have to use **kubectl.** In order to use that tool, we have to give the k8s admin config file (that is like our password for the k8s cluster).

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config;
```

After that you can check the kubectl tool by giving the command
`kubectl get pods`

🎉 **If it works, you are ready!** 🎉

You have just deployed a working K8s cluster! Your work though is not over yet. You have to install a Network Layer Composition tool and, optionally, the Kubernetes Dashboard, so that you can have a better overview in your K8s Cluster

Making our Instance a One-Node instance with no Control-Pane

From its inception, Kubernetes has in mind a cluster topology of multiple slaves that run the pods and 1+ masters that run the important infrastructure only and the orchestration. Without explicitly stating, a Master node cannot run pods or deployments. Most of Development servers however are One-Node installs, that means that they only have a master that is changed in order to run pods.

ℹ To convert an install to One-Node configuration please run the commands bellow

```
kubectl taint nodes --all node-role.kubernetes.io/master-
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

When they are both complete and show a result similar to

```
node/<your-hostname> untainted
```

then your K8s install is successfully transformed into One-Node config.

Congrats

You can now continue your journey into K8s. This would be the perfect time to deploy a StorageClass for your cluster (Local Storage (any why you are getting the error 0/1 deployed))