Control plane

Our servers, just for reference:

```
ubuntu@control01:~$ cat /etc/hosts
127.0.0.1 localhost

192.168.0.101 control01 control01.local
192.168.0.109 control02 control02.local
192.168.0.108 control03 control03.local

192.168.0.102 cube01 cube01.local
192.168.0.103 cube02 cube02.local
192.168.0.104 cube03 cube03.local
192.168.0.105 cube04 cube04.local
192.168.0.106 cube05 cube05.local
192.168.0.107 cube06 cube06.local
```

Master 1

In our case: control01

This is our primary node, one of 3 control nodes.

We are going to install the K3s version after the K8s disaster 🙂. Use the following command to download and initialize K3s' master node. We pasted the --server into the command to tell it that we will be adding additional master nodes:

```
curl -sfL https://get.k3s.io | K3S_TOKEN="some_random_password" sh -s - server --cluster-init --disable servicelb
```

Master 2 and 3

Again, for us control02 and control03.

By using the following command on both nodes, we will add these to the cluster as master nodes containing the etcd database:

```
curl -sfL https://get.k3s.io | K3S_TOKEN="some_random_password" sh -s - server --server https://192.168.0.101:6443 --no-deploy servicelb
```

You can do that with Ansible as well:

```
ansible control02,control03 -b -m shell -a "curl -sfL https://get.k3s.io | K3S_TOKEN="some_random_password" sh -s - server --server https://192.168.0.101:6443 --no-
deploy servicelb"
```

For the `--server` parameter, we are using the IP of our primary master node. This will create a control plane for our cluster.

The control plane should be like this:

```
root@control01:/home/ubuntu# kubectl get nodes
NAME        STATUS   ROLES         AGE    VERSION
control01   Ready    etcd,master   5d3h   v1.19.4+k3s1
control02   Ready    etcd,master   5d3h   v1.19.4+k3s1
control03   Ready    etcd,master   5d3h   v1.19.4+k3s1
```

Workers

We need to join some workers now; in our case cube01 to 06.

On every worker node do the following:

```
curl -sfL https://get.k3s.io | K3S_URL="https://192.168.0.101:6443" K3S_TOKEN="some_random_password" sh -
```

You can do that with ansible as well:

```
ansible workers -b -m shell -a "curl -sfL https://get.k3s.io | K3S_URL="https://192.168.0.101:6443" K3S_TOKEN="some_random_password" sh -"
```

Setting role/labels

We can `tag` our cluster nodes to give them labels.

> **Important**
>
> k3s by default allow pods to run on the control plane, which can be OK, but in production it would not. However, in our case, we want to use disks on control nodes for storage, and that does require pods to run on them from Longhorn. So, I'll be using labels to tell pods/deployment where to run.

Lets add this tag key:value: `kubernetes.io/role=worker` to worker nodes. This is more cosmetic, to have nice output from `kubectl get nodes`.

```
kubectl label nodes cube01 kubernetes.io/role=worker
kubectl label nodes cube02 kubernetes.io/role=worker
kubectl label nodes cube03 kubernetes.io/role=worker
kubectl label nodes cube04 kubernetes.io/role=worker
kubectl label nodes cube05 kubernetes.io/role=worker
kubectl label nodes cube06 kubernetes.io/role=worker
```

Another label/tag. I will use this one to tell deployments to prefer nodes where `node-type` equals `workers`. The node-type is our chosen name for key, you can call it whatever.

```
kubectl label nodes cube01 node-type=worker
kubectl label nodes cube02 node-type=worker
kubectl label nodes cube03 node-type=worker
kubectl label nodes cube04 node-type=worker
kubectl label nodes cube05 node-type=worker
kubectl label nodes cube06 node-type=worker
```

Whole Kubernetes cluster:

```
root@control01:/home/ubuntu# kubectl get nodes
NAME        STATUS   ROLES         AGE    VERSION
control01   Ready    etcd,master   5d3h   v1.19.4+k3s1
control02   Ready    etcd,master   5d3h   v1.19.4+k3s1
control03   Ready    etcd,master   5d3h   v1.19.4+k3s1
cube01      Ready    worker        5d3h   v1.19.4+k3s1
cube02      Ready    worker        5d3h   v1.19.4+k3s1
cube03      Ready    worker        5d3h   v1.19.4+k3s1
cube04      Ready    worker        5d3h   v1.19.4+k3s1
cube05      Ready    worker        5d3h   v1.19.4+k3s1
cube06      Ready    worker        5d3h   v1.19.4+k3s1
```

You can also use `kubectl get nodes --show-labels` to show all labels for nodes.

Lastly, add following into /etc/environment (this is so the Helm and other programs know where the Kubernetes config is.)

On every node:

```
echo "KUBECONFIG=/etc/rancher/k3s/k3s.yaml" >> /etc/environment
```

Or use Ansible:

```
ansible cube -b -m lineinfile -a "path='/etc/environment' line='KUBECONFIG=/etc/rancher/k3s/k3s.yaml'"
```

Done

> **Note**
>
> There are other options to deploy k3s. For example, Ansible can deploy everything (that might not end up the same as mine); for inspiration check out thing git repo: https://github.com/k3s-io/k3s-ansible
>
> Another solution could be to do GitOps, and make the infrastructure as a code using Flux 2 (or alternative). I might do a separate article on how to set this up, but you can have look here in mean time: https://github.com/k8s-at-home/awesome-home-kubernetes for some more inspiration 🙂

Liked it ? Buy me a drink :)

Last update: October 20, 2021

Comments

What do you think?

3 Responses

Upvote Funny Love Surprised Angry Sad

9 Comments https://rpi4cluster.com Disqus' Privacy Policy Login

Favorite Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Jose de Leon • 8 months ago

I'm really loving this guide! However, I don't think k3s is fully baked, it gives me some weird issues with the k3s database or the apiserver going sporadically unavailable, but once I switched to k8s, no issues anymore.

Reply • Share ›

Professor Chaos • 9 months ago

Is your /etc/hosts correct? I'm seeing all of them as cube1.local

```
ubuntu@control01:~$ cat /etc/hosts
127.0.0.1 localhost


192.168.0.101 control01 control01.local
192.168.0.109 control02 control02.local
192.168.0.108 control03 control03.local


192.168.0.102 cube01 cube01.local
192.168.0.103 cube02 cube01.local
192.168.0.104 cube03 cube01.local
192.168.0.105 cube04 cube01.local
192.168.0.106 cube05 cube01.local
192.168.0.107 cube06 cube01.local
```

Reply • Share ›

vladoportos Mod ↱ Professor Chaos • 9 months ago

Oh yes, thats wrong, should be cube02 cube02.local etc.. I'll fix that

Reply • Share ›

cannfoddr • 10 months ago

I have been following this and a few other guides to get k3s running on a Pi3 cluster with Ubuntu 20.04.2 with the latest updates. As soon as I try to bring up k3s things go to hell. The scared light is on almost continuously. I can see tons of k3s activity in syslog but nothing seems to start.

I got a k3s system up on 32bit Hypriot using the same hardware but 64 bit ubuntu seems to be suffering on the same hardware.

I can see failed leader elections, timeouts, its a real mess

I am tempted to give up entirely on k3s and try again with kubeadm - I have a kubeadm cluster running on a Pi4 setup.

Any suggestions on what I might be doing wrong? This is server only on a single node at the moment.

Reply • Share ›

DfUSE DfU • 10 months ago

And also confirming that this ansible command contains a typo:

ansible control02,control03 -b -m shell -a "curl -sfL https://get.k3s.io | K3S_TOKEN="some_random_password" sh -s - server --server https://192.168.0.101:6443" --no-deploy servicelb

and should be like:

ansible control02,control03 -b -m shell -a "curl -sfL https://get.k3s.io | K3S_TOKEN="some_random_password" sh -s - server --server https://192.168.0.101:6443 --no-deploy servicelb"

Reply • Share ›

vladoportos Mod ↱ DfUSE DfU • 10 months ago

Thanks I will fix that.

Reply • Share ›

DfUSE DfU • 10 months ago

Just confirming... k3s install must be done as root, right?

Reply • Share ›