Guide for Kubernetes Cluster on CloudLab

Shangyu Xie

April 1, 2017

1 Prepration

1.1 Background

This document is based on my installation process and the official guide. In fact, there are two guides for installing kubernetes cluster. However the first maintained by Zhejiang University does not work for Ubuntu 16.04 and the test version for kubernetes is old. See here.

Via checking two methods myself, I take the second method, that is, use kubeadm (the command to bootstrap the cluster, refer here. In this guide, I mainly stress the possible error when you install the cluster or configure. The specific install process is direct and easy on the official guide.

1.2 Note

a. There are four packages you will use, docker, kubelet, kubectl and kubeadm.

- Docker is the dependency for the kubernetes, as is known to all.
- Kubelet is the core component of the kubernetes, take responsibility for staring containers and pods on machines.
- Kubectl is the command to control the cluster. More help refer here.
- kubeadm is the bootstrap command for installing cluster.

b. The kubeadm is still in beta version, maybe not stable for production of clusters.

c. Make sure you are in the root to execute the command.

2 Implementation

2.1 Initialize the machine

We use the cloudlab to implement the cluster. I take the system Ubuntu 16.04 image, and install the four packages, snapshot this image. Then use the edit source file to create the three-nodes cluster. Here are some points your should notice:

- Remember to change the owner of /local directory to your user, otherwise you cannot make any operation without sudo.
- Edit the setup.sh to make sure that ssh log on without password, and ensure the setup.sh be executed in /etc/rc.local.
- when you install the docker, do NOT forget to test the docker to see if it works, that is, make sure every step is correct and reasonable besides docker.

2.2 Initialize the master node

The initialization of master node is easy because kubeadm helps to finish it. Just run
$ kubeadm init
And this command will run prechecks to ensure that machine is ready for the kubernetes. So you can always find the error by the output message after running this command. The common error includes some files are not empty, etc. , you need to use
$ kubeadm reset
or just delete the corresponding files via message.

Note here is a bug for my test even though you pass the precheck tests. There is a open issue for this error on Github, that is, you will get stuck at

$ **[apiclient] First node has registered, but is not ready yet**

(Refer here).

And I summarized the comments below, the steps which do work for my test show below:

a. Remove the environment variable $ KUBELET_ NETWORK_ARGS in file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf;

b. Run $ systemctl daemon-reload to reload the configuration.

c. Run $ kubeadm reset(an insurance to run);

d. Finally, run the $ kubeadm init again.

Note again, all commands run as root. If you follow the steps, you will something like:
....
Kubernetes master has initialized successfully!
....

2.3 Another possible error

But it is not finished yet. You probably get another error
The connection to the server localhost:8080 was refused
if you do not notice the outputs.
....
To start using your cluster, you need to run (as a regular user):
....
You should run the following command to let the cluster get configured, otherwise you will get the error.
$ sudo cp /etc/kubernetes/admin.conf $HOME/
$ sudo chown $(id -u):$(id -g) $HOME/admin.conf
$ export KUBECONFIG= $HOME/admin.conf

The reason you get this error because the kubectl ask the wrong server(localhost:8080) if you do not configure the cluster. You can run
$ kubectl config view
to check the configuration of the whole cluster.

2.4 Install the pod network

You need to install the pod network add-on before you add the nodes to the cluster. The network configuration file is in .yaml format, and you can choose to download the network configuration you like from here. And run
$ kubectl apply -f ADD_ ON.yml

2.5 Add work nodes to the cluster

Just make sure the work nodes you add can communicate with your master node in a private or public network. By the way, you have installed docker, kuber essential already. Run
$ kubeadm join –token $YOUR_ TOKEN YOUR_ master-ip:port

Someone on Github mentioned that the token need to be remember because you need to add nodes via this. But you can always find the token in the admin.conf. Just keep it safe, because anyone can add nodes to your cluster if they have the token.(But I do not think someone would be funny enough to do this.)

After you add the nodes, you can always check the nodes via $ kubectl get nodes

3 Test

Just follow the guide to test the cluster, and try to learn more command of kubectl, anything else. And there are also some interesting applications, you can check at the official guide.