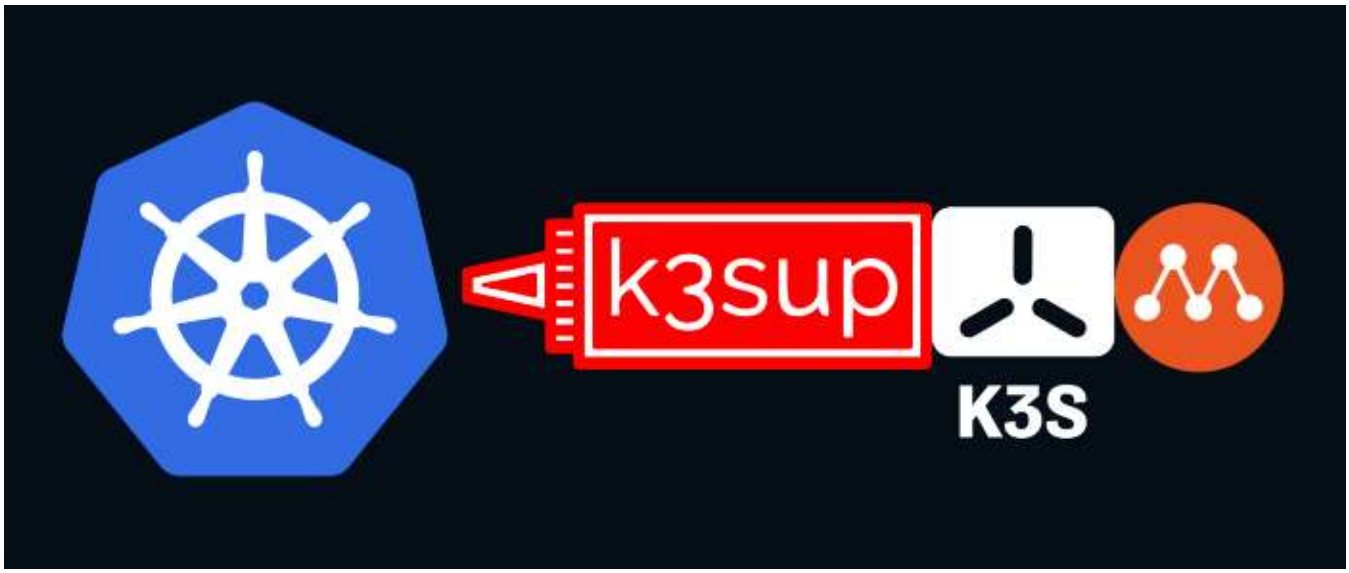
Yankee Maharjan

Follow 138 Followers About

Setting up multi-node Kubernetes cluster locally with K3s and Multipass

Yankee Maharjan Oct 24, 2020 · 4 min read



There are a lot of tools that allow you to setup a local Kubernetes cluster in no time. But with a full-blown K8s running on your local machine, you will soon hit a wall if you want to play with multi-node cluster.

For tackling this very issue, we will be looking into how we can setup a lightweight Kubernetes cluster using K3s and Multipass for our VMs.

> *If you are interested in what makes K3s so light, you can watch the talk on K3s under the hood.*

Prerequisites

machine. With the CLI, you can spawn VMs within minutes allocating optimal CPU, memory and disk space.

> **Download Multipass**

Installing K3sup (ketchup)

We talked about K3s but it can be daunting to set up if you just want a quick hands on with Kubernetes. To abstract all that we will be installing a handy CLI called k3sup. It will bootstrap Kubernetes cluster on our VMs using K3s within a minute! 🚀

k3sup uses SSH under the hood, so make sure you have SSH service on your machine.

> *Install k3sup*

Creating Virtual machines

Nodes are just virtual machines on Kubernetes working in sync. For our cluster we will be creating 3 virtual machines, each with 1CPU, 1GB RAM, 2GB Storage.

Generating keys

If you have SSH installed and already have `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub` , you can skip this part and move to **creating config file** section.

Create Private/Public key:

```
$ ssh-keygen
```

This will create the aforementioned files on your system. Copy the contents of `~/.ssh/id_rsa.pub`

```
$ cat ~/.ssh/id_rsa.pub
```

Creating config file

Create a file called `multipass.yaml` and place your public key in `ssh-rsa` .

```
  - ssh-rsa <add-your-public-key>
```

This config file makes sure the public key is stored on the virtual machine once it's created. Let's create our VMs with proper names based on their roles we will be assigning (master/worker).

```
$ multipass launch --cpus 1 --mem 1G --disk 2G --name master-node --
cloud-init multipass.yaml

$ multipass launch --cpus 1 --mem 1G --disk 2G --name agent-master -
-cloud-init multipass.yaml

$ multipass launch --cpus 1 --mem 1G --disk 2G --name agent-worker -
-cloud-init multipass.yaml

$ multipass ls
```

In K3s terms, a master node is called the server and the rest of the nodes are called agents. Agents are simply the nodes that gets added to the master node; they can be another master node or a worker node.

Adding K8s sauce with k3sup

Now that we have our VMs ready, let's install Kubernetes on them. First we will be creating a master node to setup a control plane.

Adding a master node

We will need the `IP` and `username` of our virtual machine to SSH into and install Kubernetes. Run `multipass ls` and take note of `IP` of `master-node` . All the usernames for VMs are `ubuntu` by default.

```
$ k3sup install --ip <IP> --user ubuntu --k3s-extra-args "--cluster-
init"
```

We are passing the `--k3s-extra-args "--cluster-init"` to make sure this node is prepared to connect with another master node, else it might cause errors.

downloaded `kubeconfig` file.

```
# mac / linux
$ export KUBECONFIG=<path-to-kubeconfig>

# windows
$ setx KUBECONFIG <path-to-kubeconfig>
```

Now you can `kubectl get nodes` to view the nodes on your cluster.

Adding a second master node (HA)

Multi master setup on local machine is an overkill for most use cases but for the purpose of this tutorial we are going to add it anyway. To join a new master node in the cluster we use the `join` command.

Here, all we have to do is introduce the server `IP` and `username` (previously created) that this node should connect to . And additionally pass the `--server` flag specifying this will be a server node (master).

```
$ k3sup install --ip <IP-of-agent-master> --user ubuntu --server-ip
<IP-of-master-node> --server-user ubuntu --server
```

And that's about it. Now you can `kubectl get nodes` again, and you'll see two master nodes that is Highly Available (HA).

Adding a worker node

Finally, we have to setup our worker node where we will be deploying our applications and services.

For this grab the `IP` of `agent-worker` and pass on the same command as before minus the `--server` flag.

```
$ k3sup join --ip <IP-of-agent-worker> --user ubuntu --server-ip
<IP-of-master-node> --server-user ubuntu
```

master or worker nodes depending on how you plan to utilize your cluster.

Conclusion

Thank you for reading. Hope this guide has been helpful for you to setup and play with multi-node Kubernetes cluster. If you have any corrections, suggestions, or feedback feel free to DM me on Twitter or comment below.

Happy hacking! 🚀

Kubernetes Cloud Computing DevOps Containers Technology