Diagram

Description automatically generated

Kubernetes step by step setup guide for beginners.

If you are new to container orchestration and Kubernetes do not worry. We will have a quick glance to what container is and what Kubernetes do for us. Finally, we will set up your first Kubernetes cluster together. Let’s start, no time to lose.

1. **What is a Container?**

A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Docker is a containerization software that performs operating-system-level-virtualization. The developer of this software is Docker, Inc. The initial release of this software happened in the year 2013. It is written in ‘Go’ programming language.

Container images: Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Docker Engine: Docker container technology was launched in 2013 as an open-source Docker Engine. Docker's technology is unique because it focuses on the requirements of developers and systems operators to separate application dependencies from infrastructure. Technology available from Docker and it is open source. Integrated into cloud technologies by all major data centre vendors and cloud providers. Many of these providers are leveraging Docker for their container-native IaaS offerings. Additionally, the leading open source serverless frameworks utilize Docker container technology.

1. **What Kubernetes do?**

With enterprises containerizing their applications and moving them to the cloud, there is a growing demand for container orchestration solutions. While there are many solutions available, some are mere re-distributions of well-established container orchestration tools, enriched with features and, sometimes, with certain limitations in flexibility. There are number of paid or free to use container orchestration tools and services available and currently most popular of them is Kubernetes.

Kubernetes is an open-source platform supports automation of deployment, scaling and management of containerised services. Kubernetes originally developed by google and maintained by the Cloud Native Computing Foundation. Kubernetes pronunciation of a Greek word, meaning “helmsmen” or “ship pilot”.

When you deploy Kubernetes, you get a cluster. A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node. A node may be a virtual or physical machine. Each node is managed by the control pane and contains the services necessary to run pods. Each pod is a logical host for a container. The worker node(s) host the pods that are the components of the application workload. The control node manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

1. **Kubernetes setup**

Welcome to magical world of Kubernetes container orchestration. This part provides a beginners’ hands-on guide for setting up a Kubernetes cluster on Ubuntu (20.x) servers.

Please note following pre-setup information before you begin your Kubernetes journey, taken from; <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/> on 17th February 2021

* One or more machines running one of:
  + Ubuntu 16.04+
  + Debian 9+
  + CentOS 7+
  + Red Hat Enterprise Linux (RHEL) 7+
  + Fedora 25+
  + HypriotOS v1.0.1+
  + Flatcar Container Linux (tested with 2512.3.0)
* 2 GB or more of RAM per machine (any less will leave little room for your apps).
* 2 CPUs or more.
* Full network connectivity between all machines in the cluster (public or private network is fine).
* Unique hostname, MAC address, and product uuid for every node.
* Certain ports are open on your machines.
* Swap disabled. You **MUST** disable swap in order for the kubelet to work properly. If you are using AWS instances you can ignore this as AWS do not allow swap as default.

Once you made sure that your equipment complies with above listed requirements you can go ahead and start the process. You have to have two or more instances and can be physical or virtual machines.

Please establish ssh remote connection to each machine separately. If you have not done this before, please refer to; <https://code.visualstudio.com/docs/remote/ssh-tutorial> for guidance.

Please note I have used two ubuntu machines. One for master / control plane and one for worker node. You may choose to use more than one worker node. I name them as follows.

**On master:**

sudo hostnamectl set-hostname kubemaster

bash

**On worker node:**

sudo hostnamectl set-hostname kubeworker

bash

You now can easily distinguish which one is which now and you can go ahead and start installing packages for Kubernetes as follows

**On both machines.**

Installation of Kubernetes helper packages:

sudo apt-get update && sudo apt-get install -y apt-transport-https gnupg2

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

Update app repository:

sudo apt-get update

Install Kubernetes and packages and Docker on each machine:

sudo apt-get install -y kubectl kubeadm kubelet kubernetes-cni docker.io

Start Docker service:

sudo systemctl start docker

Enable Docker service so that the service will automatically resume whenever rebooted:

sudo systemctl enable docker

Start kubelet service:

sudo systemctl start kubelet

Enable kubelet service so that the service will automatically resume whenever rebooted:

sudo systemctl enable kubelet

Add the current user to the Docker group so that the Docker commands can be run with needing root privileges:

sudo usermod -aG docker $USER

newgrp docker

In order to enable the iptables of Linux Nodes to see bridged traffic correctly please set ‘net.bridge.dridge-nf-call-iptables’ to 1 in ‘sysctl’ config and activate ‘iptables’ as follows:

cat << EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl --system

**On kubemaster**

Pull Kubernetes packages:

sudo kubeadm config images pull

Let ‘kubeadm’ to prepare the environment:

sudo kubeadm init --apiserver-advertise-address=<private ip addres of kubemaster> --pod-network-cidr=172.16.0.0/16

Please ensure to check your Master instance’s IP address and replace the same with ‘<private ip addres of kubemaster>’

You should now see the result along with following lines:

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

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/

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

kubeadm join 172.33.5.107:6443 --token 1aiej0.kf0t4on7c7bm2hlu \

--discovery-token-ca-cert-hash sha256:0e2abfb56733665c0e620423337f34be2a4f3c4b8d1ea44dff85666ddf722c02

Activate ‘Calico’ pod networking:

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

Setting up of control pane is now complete. You can now add slave / worker nodes to cluster.

**On kubeworker**

Run following command to have each node to join the cluster:

kubeadm join 172.33.5.107:6443 --token 1aiej0.kf0t4on7c7bm2hlu \

--discovery-token-ca-cert-hash sha256:0e2abfb56733665c0e620423337f34be2a4f3c4b8d1ea44dff85666ddf722c02

**On kubemaster**

Now you should be able to see the new workers in the list:

kubectl get nodes

Now you should get similar to following on the screen:

NAME STATUS ROLES AGE VERSION

kubemaster Ready control-plane,master 6h47m v1.20.2

kubeworker Ready <none> 6h38m v1.20.2

You can obtain more info about cluster using following command:

kubectl get nodes -o wide

Now setting up the Kubernetes environment is completed. You can run the cluster from control plane’s command line. Depending in the tasks you need to accomplish with Kubernetes cluster you may need to import container images, effect deployments, create replica sets, increase / decrease number of replicas in each set, set up horizontal auto scaling, set up name spaces and set up services.

You may refer to Kubernetes documentation on <https://kubernetes.io/docs/home/> for more detailed information.

Author:

Mehmet Altun

17Feb2021, London

DevOps Engineer @ Finspire Technology