Kubernetes a.k.a k8s and Minikube

Content

1. What is Kubernetes?
2. Why Kubernetes?
3. Kubernetes glossary
4. Kubernetes vs. Docker Swarm
5. What is minikube?
6. How to setup minikube?
7. Running Kubernetes locally via Minikube

What is Kubernetes?

Kubernetes, or k8s (*k, 8 characters, s...get it?*), or “kube” if you’re into brevity, is an open source platform that automates Linux container operatio ns. It eliminates many of the manual processes involved in deploying and scaling containerized applications. In other words, you can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters. These clusters can span hosts across public, private, or hybrid clouds.

Why Kuberentes?

Real production apps span multiple containers. Those containers must be deployed across multiple server hosts. Kubernetes gives you the orchestration and management capabilities required to deploy containers, at scale, for these workloads. Kubernetes orchestration allows you to build application services that span multiple containers, schedule those containers across a cluster, scale those containers, and manage the health of those containers over time.
Kubernetes fixes a lot of common problems with container proliferation—sorting containers together into a ”pod.” Pods add a layer of abstraction to grouped containers, which helps you schedule workloads and provide necessary services—like networking and storage—to those containers. Other parts of Kubernetes help you load balance across these pods and ensure you have the right number of containers running to support your workloads.

Kubernetes Glossary

Like any technology, there are a lot of words specific to the technology. Below are some of the more common terms:

- **Master:** The machine that controls Kubernetes nodes. This is where all task assignments originate.
- **Node:** These machines perform the requested, assigned tasks. The Kubernetes master controls them.
- **Pod:** A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources. Pods abstract network and storage away from the underlying container. This lets you move containers around the cluster more easily.
- **Replication controller:** This controls how many identical copies of a pod should be running somewhere on the cluster.
- **Service:** This decouples work definitions from the pods. Kubernetes service proxies automatically get service requests to the right pod—no matter where it moves to in the cluster or even if it’s been replaced.
- **Kubelet:** This service runs on nodes and reads the container manifests and ensures the defined containers are started and running.
- **kubectl:** This is the command line configuration tool for Kubernetes.
- **Minikube:** Minikube is a tool that makes it easy to run Kubernetes locally

Above mentioned terms represent only the key terms of Kubernetes. If you are interested in more detailed glossary follow https://kubernetes.io/docs/reference/

Docker Swarm vs Kubernetes

Reading all that and assuming you are familiar with Docker in details you might be wondering why you should opt for Kubernetes while Docker comes with very similar native clustering system - the so called Swarm. Very good question - in short Swarm focuses on ease of use with integration with Docker core components while Kubernetes remains open and modular. Below is a summary of both systems pros and cons that should hopefully answer your dilemma:

Docker Swarm	Kubernetes

Pros	Pros
• Easy and fast setup • Works with other existing Docker tools • Lightweight installation • Open source	• Open source and modular • Runs well on any operating systems • Easy service organisation with pods • Backed by years of expert experience
Cons	**Cons**
• Limited in functionality by what is available in the Docker API • Limited fault tolerance	• Laborious to install and configure • Incompatible with existing Docker CLI and Compose tools

Docker provides a simple solution that is fast to get started with while Kubernetes aims to support higher demands with higher complexity. For much of the same reasons, Docker has been popular among users who prefer simplicity and fast deployments. At the same time, Kubernetes is used in production environments by many high profile Internet companies running popular services.

What is Minikube?

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a VM on your laptop/local machine for users looking to try out Kubernetes or develop with it day-to-day.
If you are interested in getting into more details of Minikube definition and features please visit https://kubernetes.io/docs/getting-started-guides/minikube/#minikube-features

How to setup Minikube?

First of all you'll need VT-x (for Intel processors) or AMD-v (for AMD processors) virtualization enabled in your computer's BIOS. In order to do that reboot your laptop/local machine and get into the BIOS menu. Locate Virtualization section and check if it's enabled.
Now that you are certain virtualization is supported you can proceed with Minikube setup by following the steps below:

- Install Hypervisor choosing one of the options below:
  - VirtualBox
  - KVM
- Install kubectl
- Install Minikube

Input
```
$ curl -Lo minikube
https://storage.googleapis.com/minikube/releases/v0.22.2/minikube-linux-amd64 && chmod +x
minikube && sudo mv minikube /usr/local/bin/
```

Running Kubernetes locally via Minikube

OK so you have Minikube installed - now what? Minikube binary is by far the easiest and quickest way to get Kubernetes for a spin. It will allow you to learn the Kubernetes API, resource objects as well as how to interact with it with the *kubectl* client. Below is brief description of most of the fundamental commands.

- Just typing *minikube* at your shell prompt will return the usage. The first command to use, however, is the *start* command. This will boot the virtual machine that will run Kubernetes.

Input
```
$ minikube start
```

Kubectl is now configured to use the cluster.
- You can check the status of your minikube VM with the *status* command.

Input
```
$ minikube status
```

- To check if *kubectl* is configured to talk to the minikube VM, try to list the *nodes* in your Kubernetes cluster. It should return a single node with the name *minikubevm*.

Input
```
$ kubectl get nodes
```

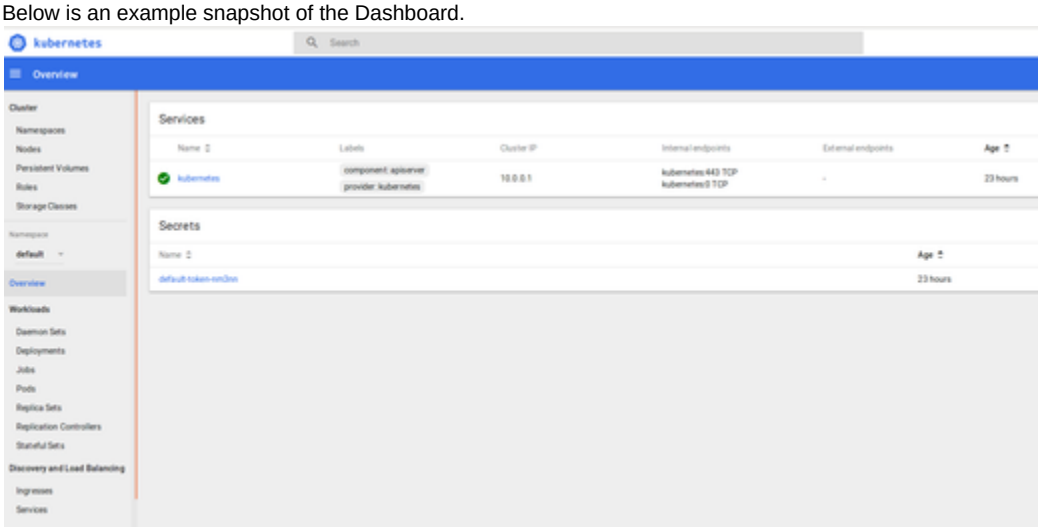
Output
```
NAME          STATUS    AGE

minikubevm    Ready     1h
```

- Kubernetes also has a Dashboard where you can see all resources within your Minikube. It's easily accessible by entering the below command in your Terminal:

Input
```
$ minikube dashboard
```

Below is an example snapshot of the Dashboard.



- If you want to learn what is actually happening inside the minikube VM, you can SSH into it with the *minikube ssh* command.
- Because Kubernetes uses the Docker engine to run the containers, you can also use minikube as a Docker host with the *minikube docker-env* command.
- You might also be interested in testing different versions of Kubernetes. Minikube allows you to do so. Check what versions are available to test with *minikube get-k8s-versions* and use the *--kubernetes-version=* flag in *minikube start* to set a specific version.
- Finally, you can stop and delete the minikube VM with intuitive commands like *minikube stop* and *minikube delete*.

- How to install VirtualBox
- How to install KVM