


Creating a Local Kubernetes Cluster with Minikube

In this lesson, we will run a Kubernetes cluster locally using Minikube.

WE'LL COVER THE FOLLOWING ^

- Starting Minikube to Create a Cluster
 -  A Note to Windows Users
- Understanding the Process
- Verification
- Exploring the Kubernetes Dashboard

Starting Minikube to Create a Cluster

The folks behind Minikube made creating a cluster as easy as it can get. All we need to do is to execute a single command. Minikube will start a virtual machine locally and deploy the necessary Kubernetes components into it. The VM will get configured with Docker and Kubernetes via a single binary called **localkube**.

```
minikube start --vm-driver=virtualbox
```



A Note to Windows Users

You might experience problems with *VirtualBox*. If that's the case, you might want to use *Hyper-V* instead.

- Open a Powershell Admin Window and execute the `Get-NetAdapter` command, noting the name of your network connection.
- Create a `hyperv` virtual switch `New-VMSwitch -name NonDockerSwitch -NetAdapterName Ethernet -AllowManagementOS $true` replacing `Ethernet` with your network connection name.

- Then, create the Minikube vm: `minikube start --vm-driver=hyperv --hyperv-virtual-switch "NonDockerSwitch" --memory=4096`.

Other minikube commands such as `minikube start`, `minikube stop` and `minikube delete` all work the same whether you're using *VirtualBox* or *Hyper-V*.

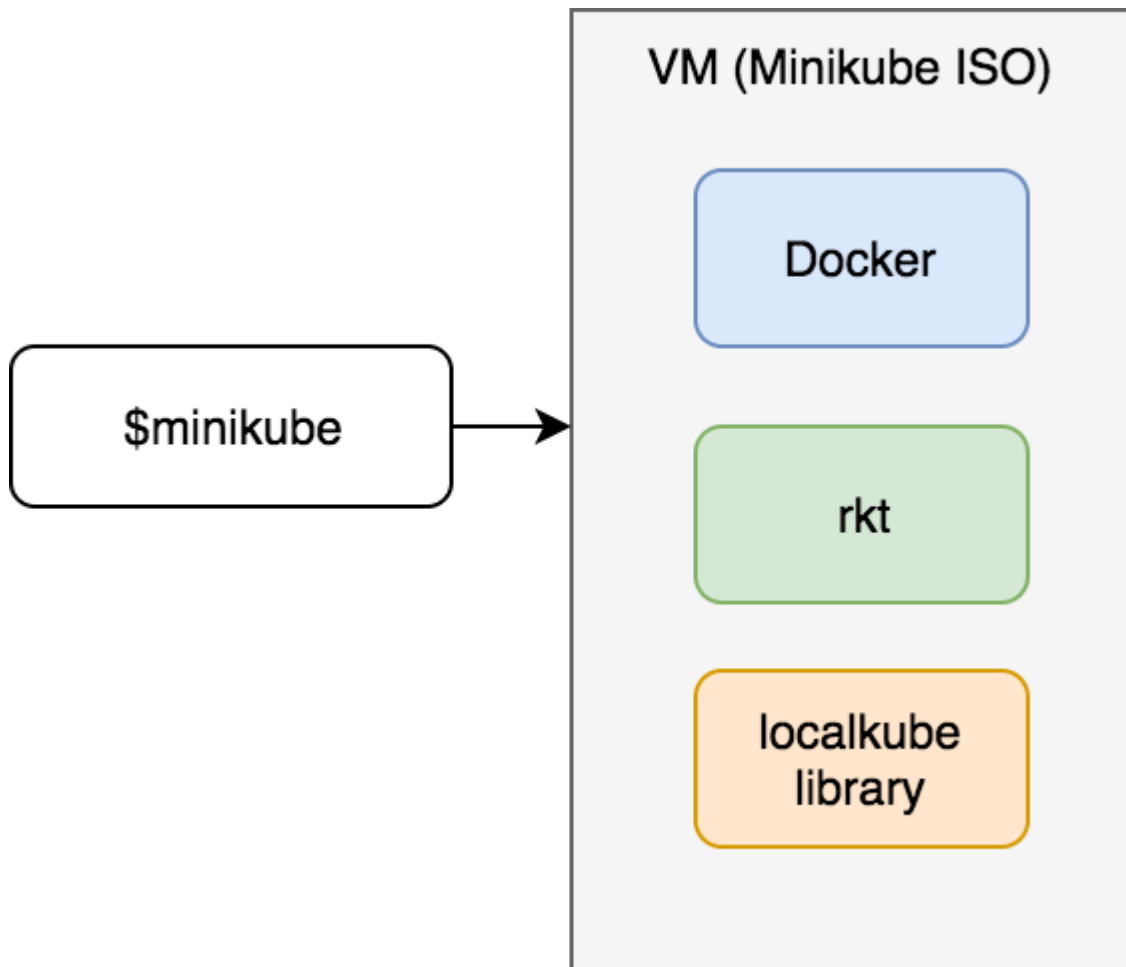
A few moments later, a new Minikube VM will be created and set up, and a cluster will be ready for use.

Understanding the Process

When we executed the `minikube start` command, it created a new VM based on the Minikube image. That image contains a few binaries. It has both *Docker* and *rkt* container engines as well as *localkube* library.

rkt is an application container engine developed for modern production cloud-native environments.

The *localkube* library includes all the components necessary for running Kubernetes. We'll go into details of all those components later. For now, the important thing is that *localkube* provides everything we need to run a Kubernetes cluster locally.



Minikube simplified architecture

Remember that this is a single-node cluster running locally on our machine only. While that is unfortunate, it is still the easiest way to “play” with Kubernetes locally. It should do, for now. Later on, we’ll explore ways to create a multi-node cluster that will be much closer to a production setup.

Verification

Let’s take a look at the status of the cluster.

```
minikube status
```



The **output** is as follows.

```
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.100
```



Minikube is running, and it initialized a Kubernetes cluster. It even configured `kubectl` so that it points to the newly created VM.

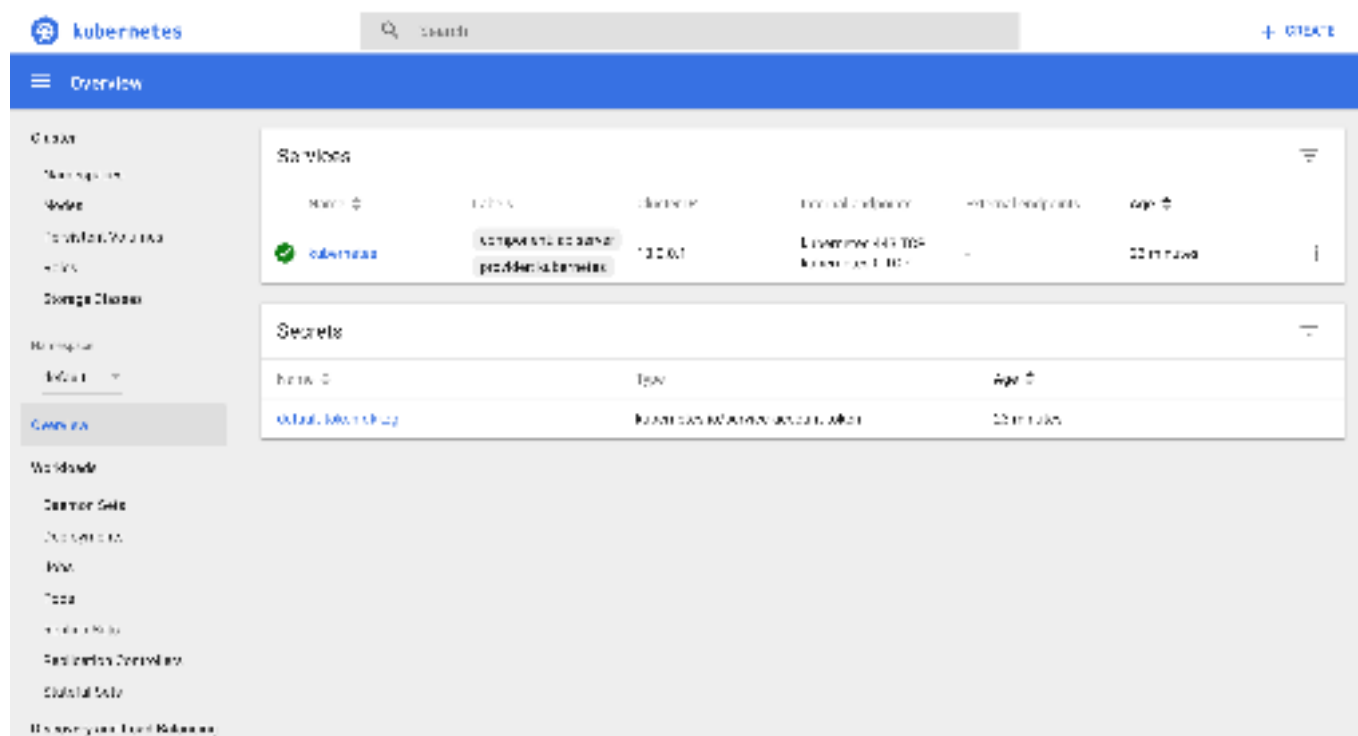
Exploring the Kubernetes Dashboard

You won't see much UI in this course. We believe that a terminal is the best way to operate a cluster. More importantly, we are convinced that one should master a tool through its commands first. Later on, once we feel comfortable and understand how the tool works, we can choose to use a UI on top of it. We'll explore the Kubernetes UI in one of the later chapters. For now, you can have a quick glimpse of it.

```
minikube dashboard
```



The above command will redirect you to the following page on your browser.



Feel free to explore the UI but don't take too long. You'll only get confused with concepts that we did not yet study. Once we learn about pods, replica-sets, services, and a myriad of other Kubernetes components, the UI will start making much more sense.

In the next lesson, we'll explore some other useful Minikube commands.

