ToiletPaper #104

jambit
WHERE INNOVATION WORKS

Minikube vs. kind vs. k3s – Which local Kubernetes cluster should I use?

Author: Maximilian Brenner / DevOps Engineer / Office Leipzig

➔ Explanation

These days there are a few tools that claim to (partially) replace a fully fledged Kubernetes cluster. Using them allows e.g. every developer to have their own local cluster instance running to play around with it, deploy their application or execute tests against applications running in K8s during CI/CD. In this post we'll have a look at three of them, compare their pros and cons and identify use cases for each of them.

✔ minikube

minikube is a Kubernetes SIGs project and has been started more than three years ago. It takes the approach of spawning a VM that is essentially a single node K8s cluster. Due to the support for a bunch of hypervisors it can be used on all of the major operating systems. This also allows you to create multiple instances in parallel.
From a user perspective minikube is a very beginner friendly tool. You start the cluster using `minikube start`, wait a few minutes and your `kubectl` is ready to go. To specify a Kubernetes version you can use the `--kubernetes-version` flag. A list of supported versions can be found here.
If you are new to Kubernetes the first class support for its dashboard that minikube offers may help you. With a simple `minikube dashboard` the application will open up giving you a nice overview of everything that is going on in your cluster. This is being achieved by minikube's addon system that helps you integrating things like, Helm, Nvidia GPUs and an image registry with your cluster.

✔ kind

Kind is another Kubernetes SIGs project but is quite different compared to minikube. As the name suggests it moves the cluster into Docker containers. This leads to a significantly faster startup speed compared to spawning VM.
Creating a cluster is very similar to minikube's approach. Executing `kind create cluster`, playing the waiting game and afterwards you are good to go. By using different names (`--name`) kind allows you to create multiple instances in parallel.
One feature that I personally enjoy is the ability to load my local images directly into the cluster. This saves me a few extra steps of setting up a registry and pushing my image each and every time I want to try out my changes. With a simple `kind load docker-image my-app:latest` the image is available for use in my cluster. Very nice!
If you are looking for a way to programmatically create a Kubernetes cluster, kind kindly (you have been waiting for this, don't you :P) publishes its Go packages that are used under the hood. If you want to get to know more have a look at the GoDocs and check out how KUDO uses kind for their integration tests.

✔ k3s

K3s is a minified version of Kubernetes developed by Rancher Labs. By removing dispensable features (legacy, alpha, non-default, in-tree plugins) and using lightweight components (e.g. sqlite3 instead of etcd3) they achieved a significant downsizing. This results in a single binary with a size of around 60 MB.
The application is split into the K3s server and the agent. The former acts as a manager while the latter is responsible for handling the actual workload. I discourage you from running them on your workstation as this leads to some clutter in your local filesystem. Instead put k3s in a container (e.g. by using rancher/k3s) which also allows you to easily run several independent instances.
One feature that stands out is called auto deployment. It allows you to deploy your Kubernetes manifests and Helm charts by putting them in a specific directory. K3s watches for changes and takes care of applying them without any further interaction. This is especially useful for CI pipelines and IoT devices (both target use cases of K3s). Just create/update your configuration and K3s makes sure to keep your deployments up to date.

➔ Conclusion

I was a long time minikube user as there where simply no alternatives (at least I never heard of one) and to be honest…it does a pretty good job at being a local Kubernetes development environment. You create the cluster, wait a few minutes and you are good to go. However for my use cases (mostly playing around with tools that run on K8s) I could fully replace it with kind due to the quicker setup time. If you are working in an environment with a tight resource pool or need an even quicker startup time, K3s is definitely a tool you should consider.
All in all these three tools are doing the job while using different approaches and focusing on different use cases. I hope you got a better understanding on how they work and which is the best candidate for solving your upcoming issue. Feel free to share your experience and let me know about use cases you are realizing with minikube, kind or k3s at @ brennerm.

Below you can find a table that lists a few key facts of each tool.

	minikube	**kind**	**k3s**
runtime	VM	container	native
supported architectures	AMD64	AMD64	AMD64, ARMv7, ARM64
supported container runtimes	Docker,CRI-O,containerd,gvisor	Docker	Docker, containerd
startup time initial/following	5:19 / 3:15	2:48 / 1:06	0:15 / 0:15
memory requirements	2GB	8GB (Windows, MacOS)	512 MB
requires root?	no	no	yes (rootless is experimental)
multi-cluster support	yes	yes	no (can be achieved using containers)
multi-node support	no	yes	yes
project page	minikube	kind	