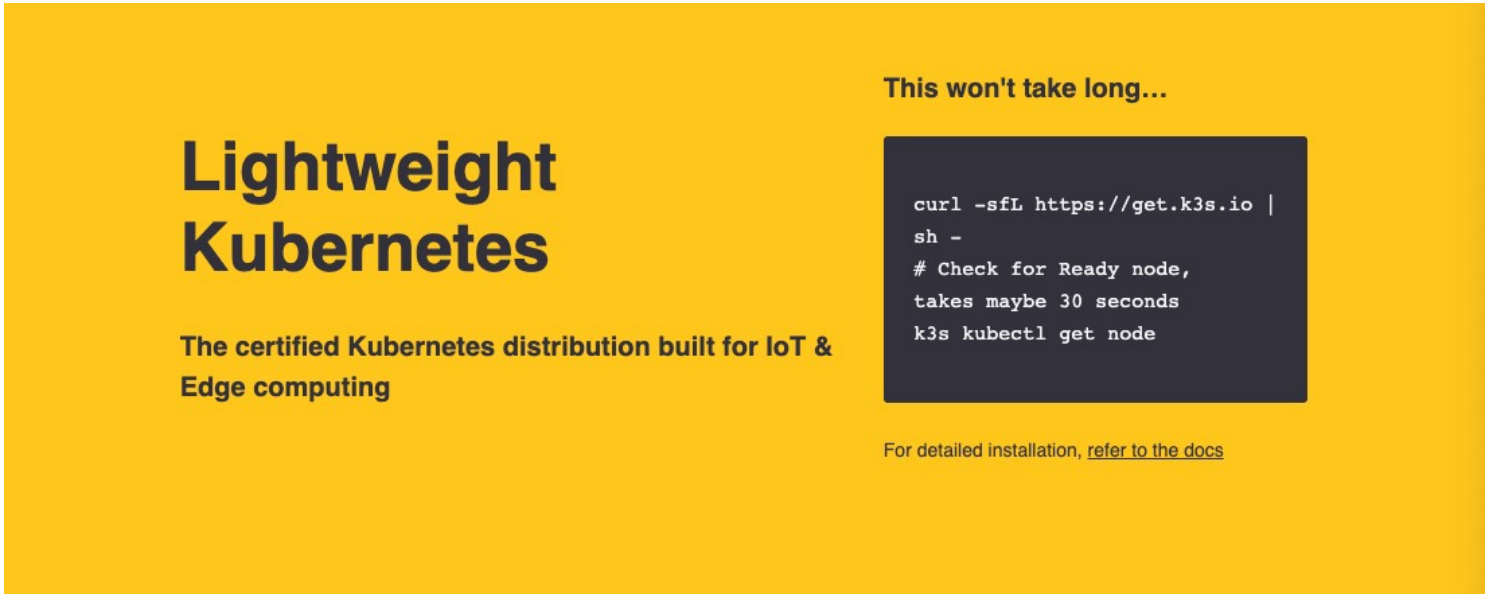
Kubernetes, also known as K8s, is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. It lets you run distributed systems resiliently with scaling and failover for your applications. But with great power comes great complexity, and K8s can be a really heavy and resource-intensive option, with a very steep learning curve.

Thankfully, there's a more lightweight solution out there: K3s. K3s is best described as a sort of distribution of Kubernetes. The same code that powers all the individual demons in Kubernetes is used in K3s, it just packages the software differently. It's a lightweight single binary that only takes up around 50 MB of space and has low resource usage of around 300 MB of RAM.

In this article, we'll go over the major differences between K8s and K3s, and why you would want to use one over the other.



Lightweight Kubernetes

Why Is Kubernetes Used to Build Applications?

Before diving into K3s, it's important to understand why you'd use Kubernetes in the first place. The most common way to build applications used to be as monoliths: single deployable artifacts. But this has many downfalls—it takes a long time for deployments because everything has to be rolled out altogether, and if different teams manage different parts of the monolith, there could be much complexity when preparing for the rollout.

Microservices combat this. Each piece of functionality is split apart into smaller individual artifacts. Kubernetes is all about managing these artifacts deployed in small containers on virtual machines or nodes. The nodes in the containers they run are all grouped together as a cluster, and each container has endpoints, DNS, storage, and scalability. Kubernetes is everything that modern applications need without the manual effort of doing it by hand.

What Is K3s?

Let's be clear: K3s is not a fork of K8s. A fork would imply diverging codebases from a common point, when in fact the opposite is true. K3s is a fully CNCF (Cloud Native Computing Foundation) certified Kubernetes distribution, secure by default and with best practices.

K3s was never actually intended as a standalone product. It came about when Rancher was building RIO (PAAS on Kubernetes) and wanted an easy way to bundle Kubernetes with it.

Rancher took all of the different processes that form Kubernetes and combined them into a single binary file made possible due to GO's simple goRoutines. Then they took that combined source code set and cross-compiled it to ARM. So now you can run K3s on ARM servers or even Raspberry Pi.

There are a lot of third-party non-CSI storage providers in the main K8s source code, so they didn't compile them; instead, they relied on supporting those that are CSI compliant.

There are also a lot of integrations with cloud providers, so they dropped them, too.

Security and communication through the Kubernetes control plane is a big part of K8s, and K3s automates all of that and manages rotating the certificates when required.

Finally, K3s doesn't need much on your base operating system because all the standard user-based tools are handled from BusyBox.

Major Differences Between K3s and K8s

If you put any YAML manifest files in a particular folder on your servers, they will be automatically installed when you install K3s.

You can use any of the common databases as backend storage because Rancher replaces `ETCD` with `KINE`. The default is SQLite, but if you use a proper database like MySQL or PostgreSQL, it can scale very well.

With the release of HELM3, this is less important, but before HELM3 was removed, the requirement for tiller K3s had custom resources for helm chart installations. This is still available, which means that you can easily install helm charts without needing the client installed on your local machine or the CI Instance.

They also took the opportunity to bundle a few components that are commonly installed in cluster core DNS, the metric server, and `traefik` as an opinionated choice of ingress controller. Installation is also really easy. You can either use their GET script:

```
curl -sfL https://get.K3s.io | sh
```

Or download it directly here. For more information, follow this official documentation.

When to Use K3s

K3s has multiple use cases, and knowing when to use it is most important. Let's highlight some scenarios where using K3s is more favorable than running K8s.

Benefits of K3s

Running a Lightweight Kubernetes Development Environment

Setting up and running Kubernetes becomes very easy and quick with K3s and also doesn't need developers to have deep knowledge of the inner workings of Kubernetes.

Running Kubernetes on Raspberry Pi Clusters

Lately, due to the rise in IoT devices and lightweight computing, clusters made with Raspberry Pi are coming into the limelight. Running Kubernetes on this lightweight device is made simple due to K3s. You can follow this article to know more about running K3s on Kubernetes.

Running Kubernetes on ARM Architecture

Devices, including IoT and mobile phones, are based primarily on ARM architecture and have a great future ahead of them in terms of computing. K3s is optimized to run such ARM-based architectures. Its small footprint and simplicity make it easier to run and manage smaller resource constraints. This article dives deeper into the edge computing patterns with K3s.

Conclusion

Kubernetes solved the distributed computing problem for everyone, but has since become quite complex. Rancher took all the major workflows of Kubernetes, tweaked it into a smaller version of Kubernetes, and named it K3s.

You now know about the significant differences between K3s and it's older cousin, K8s, and when you would prefer to use it, such as when running on a Raspberry Pi or ARM device, or if you simply want an easy to set up development environment. This has just been a basic introduction to K3s, and if you're still interested, you can learn more here.