Getting Started with Production-Ready Clusters

In this lesson, we will discuss the first steps and choices we need to make for setting up a cluster for a production-ready environment.

WE'LL COVER THE FOLLOWING

Creating a Kubernetes cluster is not trivial. We have to make many choices, and we can easily get lost in the myriad of options. The number of permutations is getting close to infinite and, yet, our clusters need to be configured consistently.

Cluster Setup

Let's compare Docker Swarm and Kubernetes for setting up a cluster.

Docker Swarm

Unlike Docker Swarm that packs almost everything into a single binary, Kubernetes clusters require quite a few separate components running across the nodes. Setting them up can be very easy, or it can become a challenge. It all depends on the choices we make initially. One of the first things we need to do is choose a tool that we'll use to create a Kubernetes cluster.

If we'd decide to install a Docker Swarm cluster, all we'd need to do is to install Docker engine on all the servers, and execute `docker swarm init` or `docker swarm join` command on each of the nodes. That's it. Docker packs everything into a single binary. Docker Swarm setup process is as simple as it can get.

can get.

Kubernetes

The same cannot be said for Kubernetes. Unlike Swarm that is highly opinionated, Kubernetes provides much higher freedom of choice. It is designed around extensibility. We need to choose among many different components. Some of them are maintained by the core Kubernetes project, while others are provided by third-parties. Extensibility is probably one of the main reasons behind Kubernetes' rapid growth. Almost every software vendor today is either building components for Kubernetes or providing a service that sits on top of it.

Exploring the Options

Besides the intelligent design and the fact that it solves problems related to distributed, scalable, fault-tolerant, and highly available systems, Kubernetes' power comes from adoption and support from a myriad of individuals and companies. You can use that power, as long as you understand that it comes with responsibilities.

It's up to you to choose how will your Kubernetes cluster look like, and which components it'll host. You can decide to build it from scratch, or you can use one of the hosted solutions like Google Cloud Platform (GCE) Kubernetes Engine. There is a third option though. We can choose to use one of the installation tools. Most of them are highly opinionated with a limited amount of arguments we can use to tweak the outcome.

You might be thinking that creating a cluster from scratch using `kubeadm` cannot be that hard. You'd be right if running Kubernetes is all we need. But, it isn't. We need to make it fault tolerant and highly available. It needs to stand the test of time. Constructing a robust solution would require a combination of Kubernetes core and third-party components, AWS know-how, and quite a lot of custom scripts that would tie the two together. We won't go down that road. At least, not now.

Our Choice

We'll use Kubernetes Operations (kops) to create a cluster. It is somewhere in the middle between do-it-yourself-from-scratch and hosted solutions (e.g., GCE). It's an excellent fit for both newbies and veterans. You'll learn which components are required for running a Kubernetes cluster. You'll be able to

make some choices.

Typically, this would be a great place to explain the most significant components of a Kubernetes cluster. You were probably wondering why we didn't do that early on when we began the journey. Still, we'll postpone the discussion for a while longer. It'll be better to create a cluster first and discuss the components through live examples.

All in all, we'll create a cluster first, and discuss its components later.

Since we have already mentioned that we'll use kops to create a cluster, in the next lesson, we'll start with a very brief introduction to the project behind it.