Kubernetes

How Everything Works Together, etcd, API Server, Scheduler, Controller, Kubelet, CRI, CNI, CSI, Project

Once you go beyond running a few containers on a single machine, you need to think about how to manage multiple nodes and make them work together at scale and that is where Kubernetes comes in. Kubernetes is the leading orchestration system for container clusters.

At first glance Kubernetes is confusing, because it involves multiple components running demanding / varying workloads on different nodes in a network. It needs to offer resilience in the face of all kinds of failure, very tight security, efficiency, timely delivery of application updates and many more features.

However, after we first review the big picture for Kubernetes, when we examine each of its components in turn, we see they each performs a focused task (storage config, change data, schedule a container, start a container). Each component on its own is relatively simple, and so we gradually build up a deeper understanding of how Kubernetes actually works.

This intensive course brings engineering professionals who already know about containers and microservices up to speed with the world's leading open source container orchestration platform.

	Contents of One-Day Training Course
Target Audience Developers, devops personnel and system administrators wishing to provision large clusters of containers using Kubernetes. **Prerequisites** Attendees to this course must have already attended our *Microservices and Containers* course or have similar experience.	**Big Picture Concepts** What is a container cluster, a master node, a worker node, schedulers, controllers, a container, a pod, kubelet, etc. How everything works together **Setting up a Container Cluster** Set up Kubernetes so that master talks to kubelet on each node, which manages a set of pods on that node, and each pod contains one or more containers **etcd** etcd is a highly reliable distributed database engine that optimally stores and allows observing of config data Like what you have in /etc config files but managed by daemons (hence name) RAFT Protocol **API Server** Offers a REST API to remote clients to configure and monitor a cluster Clients could be command-line interface (CLI) tools or admin consoles or scripts Writes settings into etcd Let's examine APIs on offer **Kubectl** A CLI that talks to API Server Main route for devops people to directly interact with API Server **Scheduler** Watches etcd for changes to configuration (e.g. via API Server) and makes scheduling decisions, writes these to etcd Scheduling algorithms and settings **Controllers** Multiple controller types are supplied to manage node lifetimes, replication, endpoints and account details Cloud controllers **Kubelet** Makes a worker node part of the cluster Watches settings in etcd for this node When change detected in etcd config, makes it so in what runs on worker node **Container Runtime Interface** CRI is an interface between kubelet and actual container runtime Popular options: CRI-o or cri-containerd Containers are usually run via OCI's runc **CNI** Container Network Interface (CNI) is a spec and implementation for networking functionality in the world of containers **CSI** Container Storage Interface (CSI) is a plugin architecture for storage volumes **Labels and selectors** Attaching key/value pairs to objects Selecting many objects based on labels **Kubernetes Source Tree** To build up deeper knowledge, we explore the source trees for Kubernetes and related projects, such as etcd (all written in Go) **Project** Deploying a real world application to a Kubernetes cluster