BASICS OF KUBERNETES

# Basics of Kubernetes

## Terminology

We have learned that Kubernetes is an orchestration system to deploy and manage containers. Containers are not managed individually; instead, they are part of a larger object called a **Pod**. A Pod consists of one or more containers which share an IP address, access to storage and namespace. Typically, one container in a Pod runs an application, while other containers support the primary application.

Kubernetes uses namespaces to keep objects distinct from each other, for resource control and multi-tenant considerations. Some objects are cluster-scoped, others are scoped to one namespace at a time. As the namespace is a segregation of resources, pods would need to leverage services to communicate.

Orchestration is managed through a series of watch-loops, also called controllers or **operators**. Each controller interrogates the **kube-apiserver** for a particular object state, then modifying the object until the declared state matches the current state. These controllers are compiled into the **kube-controller-manager**, but others can be added using custom resource definitions. The default and feature-filled operator for containers is a **Deployment**. A Deployment does not directly work with pods. Instead it manages **`ReplicaSets`**. The ReplicaSet is an operator which will create or terminate pods according to a podSpec. The podSpec is sent to the kubelet, which then interacts with the container engine to download and make available the required resources, then spawn or terminate containers until the status matches the spec.

The service operator requests existing IP addresses and information from the endpoint operator, and will manage the network connectivity based on labels. A service is used to communicate between pods, namespaces, and outside the cluster. There are also Jobs and CronJobs to handle single or recurring tasks, among other default operators.

To easily manage thousands of Pods across hundreds of nodes could be difficult. To make management easier, we can use **labels**, arbitrary strings which become part of the object metadata. These can then be used when checking or changing the state of objects without having to know individual names or UIDs. Nodes can have **taints** to discourage Pod assignments, unless the Pod has a **toleration** in its metadata.

There is also space in metadata for **annotations** which remain with the object but is not used as a selector. This information could be used by the containers, by third-party agents or other tools.