Chapter 2.

Kubernetes

Learning Objectives

- By the end of this chapter, you should be able to:
    - Define Kubernetes.
    - Explain the reasons for using Kubernetes.
    - Discuss the features of Kubernetes.
    - Discuss the evolution of Kubernetes from Borg.
    - Explain what the Cloud Native Computing Foundation does.

What Is Kubernetes?

According to the Kubernetes website,
"Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications."

Kubernetes comes from the Greek word **κυβερνήτης:**, which means ***helmsman or ship pilot***. With this analogy in mind, we can think of Kubernetes as the manager for shipping containers.

Kubernetes is also referred to as **k8s**, as there are 8 characters between *k* and *s*.

Kubernetes is highly inspired by the Google Borg system, which we will explore in this chapter. It is an open source project written in the **Go language**, and licensed under the Apache License Version 2.0.

Kubernetes was started by Google and, with its v1.0 release in July 2015, Google donated it to the Cloud Native Computing Foundation (CNCF)

From Borg to Kubernetes

According to the abstract of Google's Borg paper, published in 2015,

> *"Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines."*

For more than a decade, Borg was Google's secret to run containerized workloads in production. Whatever services we use from **Google, like Gmail, Drive**, etc., they are all serviced using Borg.

Some of the initial authors of Kubernetes were Google employees who have used Borg and developed it in the past. They poured in their valuable knowledge and experience while designing Kubernetes. Some of the features/objects of Kubernetes that can be traced back to Borg, or to lessons learnt from it, are:

- API servers
- Pods
- IP-per-Pod
- Services
- Labels.

We will explore all of them, and more, in this course.

Kubernetes Features I

Kubernetes offers a very rich set of features for container orchestration. Some of its fully supported features are:

- **Automatic binpacking**
  Kubernetes automatically schedules the containers based on resource usage and constraints, without sacrificing the availability.

- **Self-healing**
  Kubernetes automatically replaces and reschedules the containers from failed nodes. It also kills and restarts the containers which do not respond to health checks, based on existing rules/policy.

- **Horizontal scaling**
  Kubernetes can automatically scale applications based on resource usage like CPU and memory. In some cases, it also supports dynamic scaling based on customer metrics.

- **Service discovery and Load balancing**
  Kubernetes groups sets of containers and refers to them via a Domain Name System (DNS). This DNS is also called a Kubernetes **service**. Kubernetes can discover these services automatically, and load-balance requests between containers of a given service.

Kubernetes Features II

Some other fully supported Kubernetes features are:

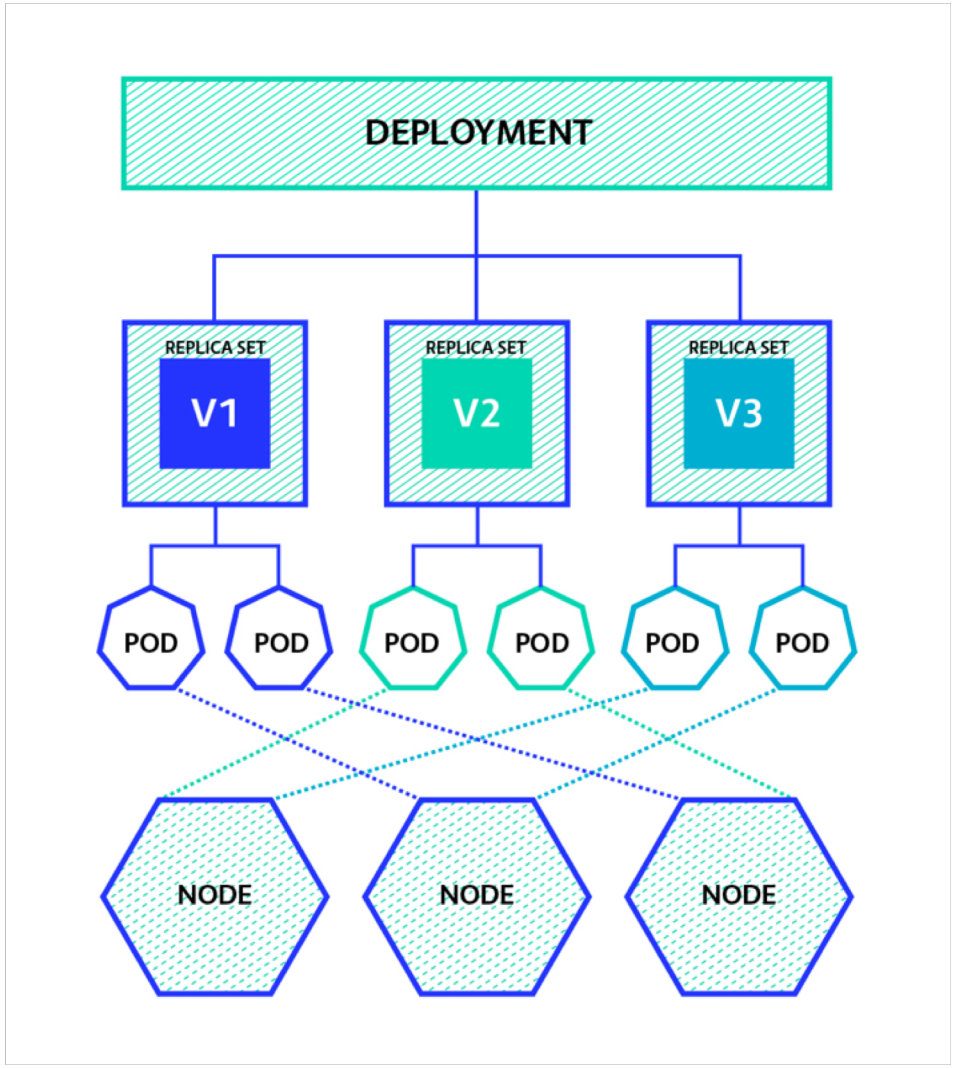
- **Automated rollouts and rollbacks**
  Kubernetes can roll out and roll back new versions/configurations of an application, without introducing any downtime.

- **Secrets and configuration management**
  Kubernetes can manage secrets and configuration details for an application without re-building the respective images. With secrets, we can share confidential information to our application without exposing it to the stack configuration, like on GitHub.

- **Storage orchestration**
  With Kubernetes and its plugins, we can automatically mount local, external, and storage solutions to the containers in a seamless manner, based on **software-defined storage (SDS).**

- **Batch execution**
  Besides long running jobs, Kubernetes also supports batch execution.

Rollout and Roll-Back

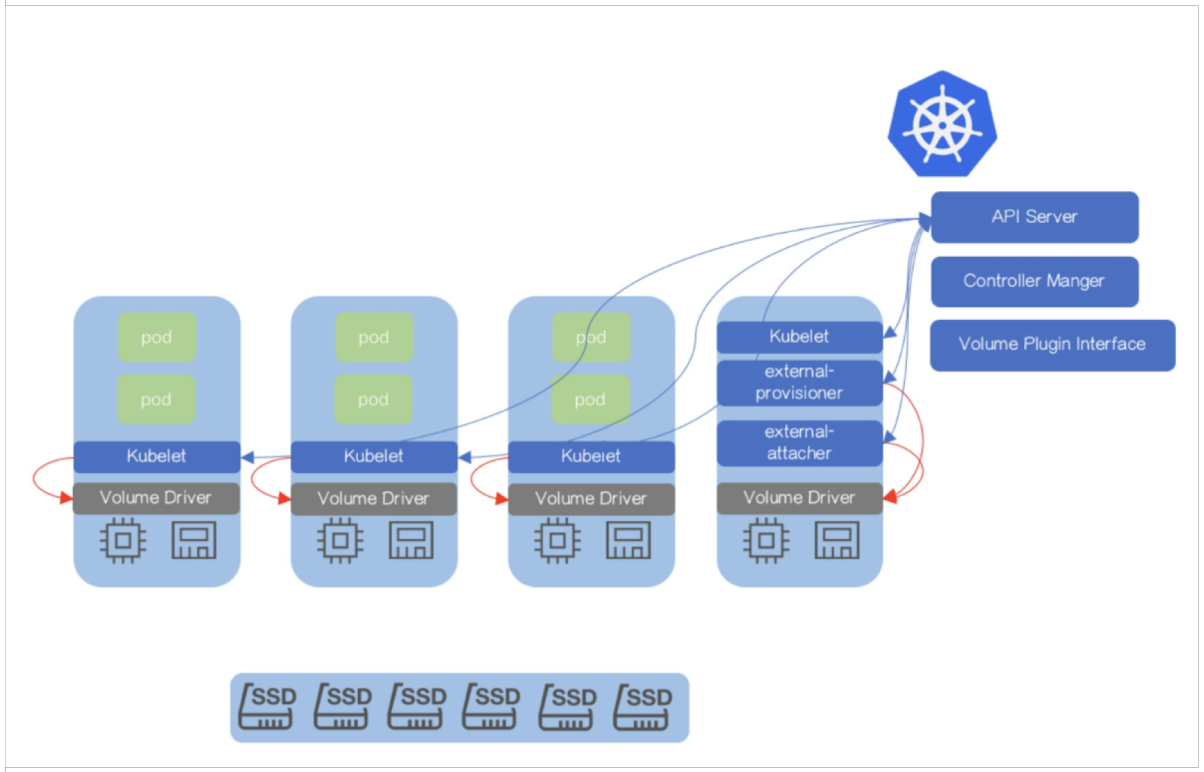


Volume

Pod-scoped storage

Support many types of volume plugins

- Empty dir (and tmpfs)
- Host path
- Git repository
- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- iSCSI
- Flocker
- NFS
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- FibreChannel
- Secret, ConfigMap, DownwardAPI
- Flex (exec a binary)
- ...



Why Use Kubernetes?

We just looked at some of the fully-supported Kubernetes features. We should also mention that Kubernetes is very portable and extensible. Kubernetes can be deployed on the environment of our choice, be it VMs, bare metal, or public/private/hybrid/multi-cloud setups. Also, Kubernetes has a very modular and pluggable architecture. We can write custom APIs or plugins to extend its functionalities.

For a successful open source project, the community is as important as having great code. Kubernetes has a very thriving community across the world. It has more than 1600 contributors, who, over time, have done over 62,000 commits. There are meet-up groups in different cities which meet regularly to discuss Kubernetes and its ecosystem. There are *Special Interest Groups* (SIGs), which focus on special interests, such as scaling, bare metal, networking, etc. We will talk more about them in our last chapter, *Kubernetes Communities*.

Kubernetes Users

With just a few years since its debut, many companies are running workloads using Kubernetes. We can find numerous user case studies on the Kubernetes website:

- Pearson
- Box
- eBay
- Wikimedia
- Huawei
- Haufe Group
- BlackRock
- BlaBlaCar
- And many more.

Cloud Native Computing Foundation (CNCF)

The Cloud Native Computing Foundation (CNCF) is one of the projects hosted by The Linux Foundation. CNCF aims to accelerate the adoption of containers, microservices, and cloud-native applications.

CNCF hosts a set of projects, with more to be added in the future. CNCF provides resources to each of the projects, but, at the same time, each project continues to operate independently under its pre-existing governance structure and with its existing maintainers. At the time this course was created, the following projects were part of CNCF:

- containerd for container runtime
- rkt for container runtime
- Kubernetes for container orchestration
- Linkerd for service mesh
- Envoy for service mesh
- gRPC for remote procedure call (RPC)
- Container Network Interface (CNI) for networking API
- CoreDNS for service discovery
- Rook for cloud-native storage
- Notary for security
- The Update Framework (TUF) for software updates
- Prometheus for monitoring
- OpenTracing for tracing
- Jaeger for distributed tracing
- Fluentd for logging
- Vitess for storage.

As we can see, this set of CNCF projects can cover the entire lifecycle of an application, from its execution using container runtimes, to its monitoring and logging. This is very important to meet the CNCF goal.

CNCF and Kubernetes

For Kubernetes, the Cloud Native Computing Foundation:

- Provides a neutral home for the Kubernetes trademark and enforces proper usage
- Provides license scanning of core and vendored code
- Offers legal guidance on patent and copyright issues
- Creates open source curriculum, training, and certification
- Manages a software conformance working group
- Actively markets Kubernetes
- Hosts and funds developer marketing activities like K8Sport
- Supports ad hoc activities
- Funds conferences and meetup events.

Learning Objectives (Review)

You should now be able to:

- Define Kubernetes.
- Explain the reasons for using Kubernetes.
- Discuss the features of Kubernetes.
- Discuss the evolution of Kubernetes from Borg.
- Explain what the Cloud Native Computing Foundation does.