Learn about Kubernetes

By

Stanley Stephen

Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/
Email: s.stanley.mca@gmail.com

What is Kubernetes?

A DevOps platform that makes it easy to deploy and manage distributed systems.

Kubernetes is an open-source platform that orchestrates container runtime systems across a cluster of networked hardware resources. It was originally developed by Google, who needed a new way to run billions of containers a week at scale. Google developed three different container-management systems to do so over the years, with the most recent being the open-source system Kubernetes.

The word Kubernetes originates from Greek and means helmsman or pilot. Kubernetes is now the market leader and industry standard orchestration tool for containers and distributed application deployment. Kubernetes is an evolution apex of earlier tools and systems that developers have been combining together to orchestrate distributed systems.

DevOps evolution started out managing physical machines in server rack rooms. These physical machines still exist but are less hands-on. Virtual machines and virtual hardware were the next layer of abstraction that get run through hypervisors on the physical machines. Developers were still spending excessive effort to manage infrastructure and virtual machines across infrastructure. Containers were developed to allow developers to focus more on high level application development and less on infrastructure. Kubernetes was then introduced to help automate and manage containers across cloud infrastructure.

Kubernetes helps developers writing applications that run in a cluster since it's used for orchestration of container runtime systems across a cluster of networked hardware resources. As Google notes, Kubernetes' "main design goal is to make it easy to deploy and manage complex distributed systems, while still benefiting from the improved utilization that containers enable."
Kubernetes is sometimes referred to as K8s or kube and is now maintained by the Cloud Native Computing Foundation.

What is Kubernetes used for?

Kubernetes is fundamentally used to deploy and manage a cluster of containerized applications and can be used to establish your own CaaS platform. It helps manage microservice application architectures and is now a critical tool for building robust modern devops CI/CD pipelines.

To better understand the relation between Kubernetes and containers it's important to understand containers. Kubernetes bundles a set of containers. An example

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/Kubernetes_by_Stanley.pdf
Email: s.stanley.mca@gmail.com

container set could be an app server, redis cache, sql database. Docker containers are one process per container. Kubernetes puts these containers into a group that it manages on the same machine to reduce network overhead and increase resource usage efficiency.

Kubernetes will provision and manage underlying hardware resources designated in your configuration. It will then proliferate and distribute application containers across the hardware resources. This allows Kubernetes to orchestrate containers across machines or on the same machine.

Kubernetes also helps optimize application development for the cloud. Most modern non-trivial applications are composed of several different business case responsibilities, such as account management, payments, and content. These separate responsibilities are suited well for a microservice architecture. By using Kubernetes early in the development of a new project, the design and implementation of microservices will often yield a strong return on investment. As the project grows it will benefit from the auto scaling, monitoring, and ease of deployment features offered by Kubernetes.

Kubernetes can be used to schedule resource intensive computational jobs. Some projects may require intensive workloads that use up machine resources for an extended period of time. Imagine a project that runs a complex simulation like protein folding, genetic analysis, or 3D graphic rendering. These types of projects often have a dedicated cluster of computational resources to run the simulation asynchronously. Kubernetes is an ideal tool for managing these clusters.

Efficiently provisioning resources across a cluster can be a challenging task. How much CPU and memory containers require can fluctuate with usage. Kubernetes has both horizontal and vertical auto-scaling mechanisms. There are 2 flavours of horizontal scaling. Kubernetes can add and remove more pods on a node or Kubernetes will add and remove new machines to the cluster. Yet vertical scaling will increase or decrease the local machine resources that a container uses. Kubernetes has monitoring built in and a set of configurable rules that it reviews to determine if it should scale up or down.

Kubernetes can be used in conjunction with some modern devOps pipeline tools to build a CI/CD pipeline since it controls and automates application deployments and updates. This deployment automation coupled with a robust automated test suite encompasses a full CI/CD pipeline.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/Kubernetes_by_Stanley.pdf
Email: s.stanley.mca@gmail.com

Who should use Kubernetes?

Kubernetes is great for modern high-performance software projects. However, there is an expensive and time consuming switching cost when moving an existing project to Kubernetes. With that in mind it's often better suited for early stage projects. Yet the return on investment for migrating an older, established system to Kubernetes may be worth the cost. This is an engineering business cost decision that should be carefully reviewed.

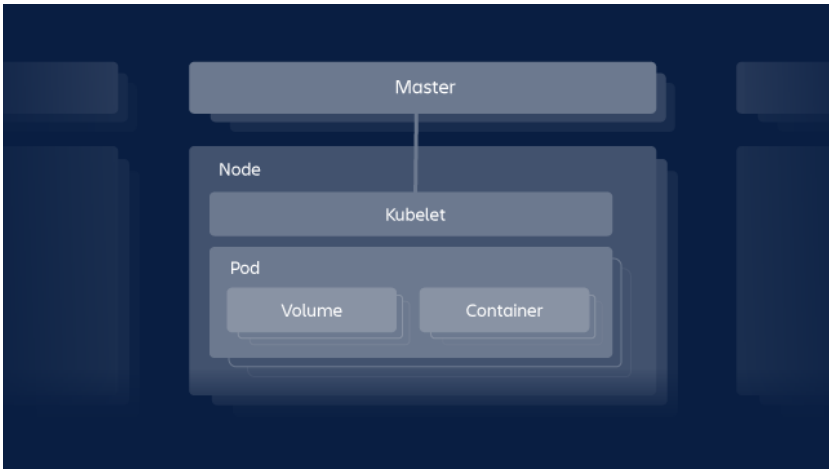
Kubernetes vs Docker

Docker is a container runtime while Kubernetes is a platform for running and managing containers from many container runtimes. Docker is one of many container runtimes that Kubernetes supports. You can almost think of Kubernetes as an "operating system" and Docker containers as "apps" that you install on the "operating system".

On its own, Docker is highly beneficial to modern application development. It solves the classic problem of "works on my machine" but then nowhere else. Standalone Docker is quite capable of handling a deployment of a few containers. When a system grows and needs to add many containers networked to each other, standalone Docker can face some growing pains that Kubernetes helps address.

Kubernetes basics

Kubernetes is made up of many components and has a few concepts to understand. An overall Kubernetes installation is known as a cluster. All Kubernetes clusters have a Master instance. The Master is responsible for managing the cluster. The Master is a daemon process that coordinates all activities in the cluster, like deployment roll outs, auto scaling, and health monitoring.



Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/Kubernetes_by_Stanley.pdf
Email: s.stanley.mca@gmail.com

The cluster is made up of Nodes. A Node is a machine entity like a VM or physical hardware that acts as a worker instance in the Kubernetes cluster. All Nodes contain a Kubelet which is a client process that communicates with the Master process using the Kubernetes AP. Nodes also house a container runtime system like Docker.

Each Node contains a Kubelet. The Kubelet is the primary process responsible for managing the node and communicating with the master. The Kubelet also digests a PodSpec that defines the life cycle and maintenance of containers in the pods that it is assigned.

Nodes contain pods, which are the smallest unit of Kubernetes. A pod is an abstraction over a container. A pod will usually include many containers per application. There is also a hidden container in every pod known as a "pause" container that holds the network namespace that other containers in the pod use to talk to localhost.

Services are static pointers to pods. Because pods are ephemeral, they have dynamic IPs that can be reassigned if the pod is restarted or recreated. This can be inconvenient if trying to network to a pod by IP address. Services help by guaranteeing a pod has a static IP address.

ReplicationController is a mechanism that ensures a given number of pods is running at any one time. The ReplicationController is a key piece of Kubernetes autoscaling functionality. If there are too many pods, the replication controller will cull them and if there are too few, the replication controller will create new pods. ReplicationSets are the successors to ReplicationController, they do basically the same thing but ReplicationSets have utilities that make it easier to target a set of pods.

Kubectl is the command line interface for executing commands against a Kubernetes cluster. Kubectl has an extensive list of available commands and options to manage Kubernetes.

Kubernetes can be deployed on both physical or virtual hardware. MiniKube is a lightweight version of Kubernetes that is used for development and will create a VM on your local machine that deploys a simple cluster with only one node.

How does Kubernetes work?

Kubernetes operates on a three phase loop of checks and balances. The three phases of a Kubernetes loop are as follows:

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/Kubernetes_by_Stanley.pdf
Email: s.stanley.mca@gmail.com

Observe

During the observe phase, Kubernetes aggregates a snapshot of the current state of the cluster. The Kubelets collect state information about their respective Nodes and feed this state back to the master process, giving the master a holistic view of the clusters current state.

Check differences

The state snapshot from the observe phase is then compared to the expected static cluster parameters specified in the Kubernetes configuration. Any discrepancies between the current state and the expected cluster state are identified and slated for action.

Take action

The master then issues commands to bring the cluster back up to the expected state. This can mean removing or creating pods, horizontal or vertical scaling across nodes, and more.

The above three phases were general descriptions to the Kubernetes loop. However, a Kubernetes cluster may actually have many different loops. These loops are executed by controllers. The ReplicationController follows the loop phases. During the take action phase, RC is solely responsible for culling or creating new Pods managed by the replica set.

Summary

Since Kubernetes is open source, it allows the freedom to take advantage of on-premises, hybrid, or public cloud infrastructures. And, it lets you effortlessly move workloads to where it matters to you. Kubernetes can be combined with modern pipeline tools like JenkinsX to build highly agile and lean CI/CD systems and should be utilized by any modern, high-performance DevOps teams.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca/Learn_from_Stanley/Kubernetes_by_Stanley.pdf
Email: s.stanley.mca@gmail.com