WHITE PAPER – **SEPTEMBER 2017**

DEMYSTIFYING KUBERNETES

Overcoming Misconceptions About Container Orchestration

vmware®

Table of Contents

DOCKER CONTAINER DEFINED

With containers, Docker has defined a standard format for packaging and porting software, much like ISO containers define a standard for shipping freight. As a runtime instance of a Docker image, a container consists of three parts:

- A Docker image
- An environment in which the image is executed
- A set of instructions for running the image

—ADAPTED FROM THE DOCKER GLOSSARY

Introduction

As an emerging technology with a Greek name, Kubernetes carries mythical connotations. Some of its features only add to the suspicion of magic—the meaning of capabilities like automatic binpacking, horizontal scaling, self-healing, and secret management might not be readily apparent. The sense of power that these terms engender, however, seems palpable: The potential to automatically place, pilot, scale, and heal an application in secret would turn the head of anyone working in IT.

This paper aims to demystify Kubernetes by addressing some of the common misconceptions surrounding the platform. Here you'll find brief explanations of what it is, what it isn't, what it does, and why you should care.

Myth: It's a Runtime Environment for Containers

It's not a runtime environment. Kubernetes is a platform for managing, or orchestrating, application containers. The platform deploys, scales, and operates containers.

As an application and its services run in containers on a distributed cluster of virtual or physical machines, Kubernetes choreographs all the moving pieces so they operate in a synchronized way to optimize the use of computing resources and to maintain the correct state.

Maintaining the desired state of a distributed application running in containers is one of the key value propositions of Kubernetes—you specify the state you want the application to be in, and Kubernetes manages all the application's services and resources to establish and maintain that desired state.

In Kubernetes, the container runtime itself is typically provided by Docker, but you can optionally use other container runtimes, such as rkt (pronounced the same as the word rocket). In other words, containers have their own runtime.

Although you don't need Kubernetes to use containers, you will likely need Kubernetes if you want to robustly and repeatedly deploy and automate an application container in a production environment.

Myth: It's Always Been Open Source

Kubernetes started out as a closed-source project at Google based on an orchestration system called Borg. Google uses Borg to initiate, schedule, restart, and monitor public-facing applications, such as Gmail and Google Docs, as well as internal frameworks, such as MapReduce.[1] Kubernetes was heavily influenced by Borg and the lessons learned from running Borg on a massive scale in a production environment. In 2015, Google open-sourced Kubernetes. Shortly afterward, Google donated it as seed technology to the Cloud Native Computing Foundation, a newly formed open-source project hosted by the Linux Foundation. (VMware is a member of the Linux Foundation and the Cloud Native Computing Foundation.)

A burgeoning open-source ecosystem around Kubernetes is rapidly evolving. A project called Prometheus adds monitoring; containerd and rkt provide alternative container runtimes; linkerd establishes a service mesh; and a number of other projects cover additional requirements, such as logging and service discovery. A project called Kubo brings the industrial-strength release engineering, deployment, and lifecycle management capabilities of BOSH to Kubernetes.

[1] For more on Borg, see Research at Google, "Large-Scale Cluster Management at Google with Borg," 2015.

Myth: Abstract Terminology Clouds the System

Terminology is partly responsible for enshrouding Kubernetes in myth. Even the name itself sounds somewhat mythical—it's the Greek word for helmsman or pilot. But there's an assortment of other terms that help push the system's intelligibility into the shadows: pod, kubelet, replica set, NodePort, horizontal autoscaler, and stateful set.

Other terms, abbreviations, and acronyms taint the fringes of the Kubernetes platform as it bumps up against containers on the one hand and the accompanying infrastructure on the other: runC, OCI, YAML, JSON, IaaS, PaaS, and KaaS. There's even the odd abbreviation of Kubernetes itself: K8s.

Yet once you become familiar with the system, its relationship to containers, and the infrastructure at its edges, the meaning of the terms comes into focus.

On Kubernetes, a pod is the smallest deployable unit in which one or more containers can be managed—in other words, you run a container image in a pod. A set of pods typically wraps a container, its storage resources, IP address, and other options up into an instance of an application that will run on Kubernetes. Docker is usually the container runtime used in a pod. As a Kubernetes administrator, you specify a pod by using a YAML file.

Another fundamental term in Kubernetes is kubelet. It manages pods. The lifecycle of pods is in turn managed by a replica set. And when a pod provides a service, such as a web server, a NodePort presents the service on a port on the nodes in the cluster for external access. When requests of that service exceed a threshold, the horizontal pod autoscaler adds resources to handle the increase in demand. If the service happens to be a stateful application running in a set of pods, the stateful set allocates and manages resources for the stateful pods, such as persistent storage.

Some terms repeatedly come up in relation to containers or infrastructure. runC refers to the code module that launches containers; it is part of containerd and managed by OCI, which stands for Open Container Initiative, an organization dedicated to setting industry-wide container standards. IaaS stands for infrastructure as a service; PaaS stands for platform as a service; and KaaS stands for Kubernetes as a service.

An example of a platform as a service is Pivotal Cloud Foundry, which in turn requires elastic infrastructure as a service—such as VMware vSphere® or a VMware software-defined data center—to meet its resource demands.

Myth: It's Just for Managing Cloud-Native Apps Built with Microservices

Although it's true that Kubernetes is the optimal system for orchestrating containerized applications built with microservices, Kubernetes can serve other use cases, most notably 12-factor apps. The 12-factor app is a methodology for developing a software-as-a-service (SaaS) application—that is, a web app—and deploying it on a platform as a service (PaaS), such as Pivotal Cloud Foundry.

Transitioning to cloud-native architectures is also a key Kubernetes use case. Even though you might not plan on using microservices in the near future, implementing Kubernetes and the right underlying infrastructure will ease the transition to a microservices architecture when you are ready to take that step. Implementing

THE BENEFITS OF MICROSERVICES

Coupled with containers, microservices are increasingly becoming the architectural pattern of choice for developing a new application. The architecture breaks up the functions of an application into a set of small, discrete, decentralized, goal-oriented processes, each of which can be independently developed, tested, deployed, replaced, and scaled.

- Increase modularity
- Make apps easier to develop and test
- Parallelize development: A team can develop and deploy a service independently of other teams working on other services
- Support continuous code refactoring to heighten the benefits of microservices over time
- Drive a model of continuous integration and continuous deployment
- Improve scalability
- Simplify component upgrades

Kubernetes along with developer-ready infrastructure addresses a lingering problem that undermines many organizations: the monolithic application. It is difficult to modify, scale, and redeploy. Lifting and shifting a monolithic application to containers and Kubernetes opens the door to begin breaking it up into easily modifiable, scalable parts later. Its new packaging in a container also increases its agility and portability now.

Another compelling use case is portability. Kubernetes works across different types of clouds. In other words, the portability of containers combined with the power of Kubernetes gives you cloud independence: You can move the same containerized application among a private cloud, a public cloud, or a hybrid cloud with minimal effort.

Flexibility is an intriguing characteristic of Kubernetes. Although it's not a use case per se, flexibility helps you adapt to unknown use cases in the future. For example, you might think of the current mantra of "delivering applications early and often" as a use case. But as your application matures, you might find that other use cases, such as service discovery, become more important. In other words, once you can successfully fulfill one use case, you might aim for another one. The flexibility and evolving power of Kubernetes can help improve your application development and deployment practices over time.

The engineers working on Kubernetes recognize that the platform's flexibility can address new use cases as they emerge. "In our experience, any system that is successful needs to grow and change as new use cases emerge or existing ones change. Therefore, we expect the Kubernetes API to continuously change and grow," the Kubernetes website says.[2]

Myth: Containers and Kubernetes Are Just Another Fad in the Hype Cycle

In July, Kubernetes celebrated its second anniversary. Kubernetes is among the highest velocity cloud-related open-source development projects in the world; for a listing of facts and figures detailing the project's popularity and adoption, see the Kubernetes retrospective.

In addition, the membership of the Cloud Native Computing Foundation, which is the open-source group managing Kubernetes, has attracted major players in the cloud-computing space, including Dell Technologies, IBM, Amazon, Microsoft, Google, Intel, AT&T, and many more. End-user members include Twitter, Capital One, eBay, and Goldman Sachs. For a list of members, see the Cloud Native Computing Foundation.

As for containers, a recent survey by 451 Research revealed a profile of high-growth implementation for an emerging ecosystem in the cloud-enabling technology market.[3]

Myth: Kubernetes Isn't Ready for Production Environments

Numerous organizations have deployed Kubernetes in production environments. Although it is an emerging technology with a burgeoning ecosystem, the feature set and API of Kubernetes are robust for a two-year-old open-source project. Keep in mind, though, that the predecessor of and the principles behind Kubernetes have been running in production at Google since 2005, orchestrating applications such as Gmail in Google's cloud-scale production environment. The Kubernetes website contains several case studies that detail how different organizations have adopted Kubernetes.

[2] Kubernetes, The Kubernetes API.

[3] 451 Research, "Application Containers Will Be a $2.7bn Market by 2020, Representing a Small but High-Growth Segment of the Cloud-Enabling Technologies Market," January 10, 2017.

BENEFITS FOR DEVELOPERS

The business value of containers and Kubernetes isn't limited to the business as a whole or the office of the CIO. Developers like containers because they make life easier, development more engaging, and work more productive.

- Portability: Containers and Kubernetes enable containerized applications to be ported across different environments and different clouds with minimal effort.
- Speed: Containers and Kubernetes expedite workflows like testing, development iterations, and deployment.
- CI/CD pipeline: Kubernetes and containers support continuous integration and continuous deployment.

In addition, surveys in both 2016 and 2017 showed not only large growth in the adoption of containers but also significant increases in the number of organizations using Kubernetes.[4]

Implementing Kubernetes in production, however, is likely to require the addition of other projects and tools in the container ecosystem.

Myth: The Ecosystem Isn't Mature Enough for Production

The container ecosystem is rapidly maturing. A clear marker of that increasing maturity is the expansion of projects hosted by the Cloud Native Computing Foundation. To support Kubernetes deployments, the foundation hosts key open-source projects, including the following:

- Prometheus, a monitoring system for Kubernetes
- OpenTracing, a vendor-neutral standard for distributed tracing
- Fluentd, a data collector for unified logging
- linkerd, a service mesh that adds service discovery, routing, failure handling, and visibility to cloud-native applications

In addition, there are a variety of enterprise-grade, production-ready technologies for working with Kubernetes in a software-defined data center, such as VMware vRealize® Log Insight™, which can process a container's standard output as a data stream.

Myth: Kubernetes Will Solve All Your Problems

Kubernetes probably won't solve all your IT, application development, and deployment problems. But as your organization undergoes digital transformation, Kubernetes might solve some of the most pressing challenges in deploying and managing applications at scale.

Production deployments of Kubernetes show that it delivers substantial IT business benefits as well as benefits to a business's bottom line. Here are some of the benefits for IT, system administrators, and DevOps:

- Consolidate servers and reduce costs through efficient resource utilization.
- Elegantly handle machine failure through self-healing and high availability.
- Ease and expedite application deployment, logging, and monitoring.
- Automate scalability for containers and containerized applications.
- Decouple applications from machines for portability and flexibility.
- Easily modify, update, extend, or redeploy applications without affecting other workloads.

These technical benefits bubble up into significant business benefits that improve your competitive advantage, reduce costs, save time, and bolster the bottom line:

- Shorten software's time to market.
- Improve developer agility and productivity.
- Respond faster to change.

[4] See Portworx, "2017 Annual Container Adoption Survey: Huge Growth in Containers," April 12, 2017; ClusterHQ, "Container Market Adoption Survey 2016"; Sysdig, "The 2017 Docker Usage Report," Apurva Dave, April 12, 2017; and Forbes, "2017 State of Cloud Adoption and Security," Louis Columbus, April 23, 2017.

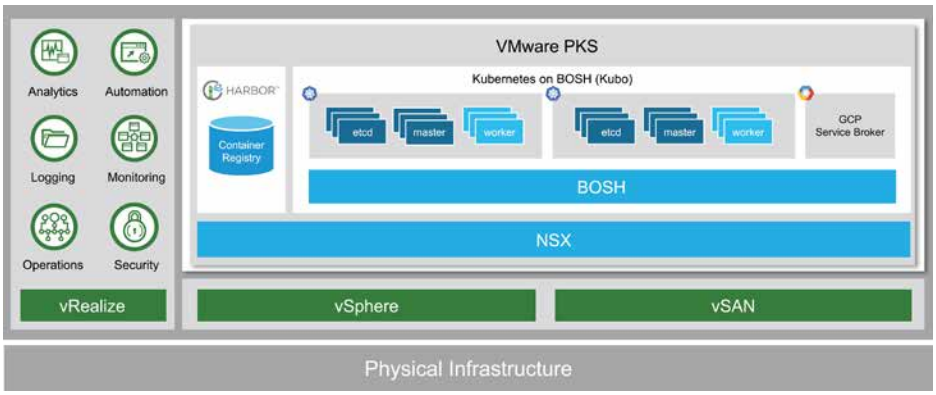
LEARN MORE ABOUT VMWARE PIVOTAL CONTAINER SERVICE

To find out more about VMware Pivotal Container Service, see https://cloud.vmware.com/pivotal-container-service.

On the other hand, Kubernetes might not be the right fit for your IT shop or your application development teams. The best way to find out, however, is to try Kubernetes out.

Myth: Kubernetes Is Hard to Deploy

An easy way to try out Kubernetes is with VMware® Pivotal Container Service (VMware PKS). PKS is a production-grade Kubernetes-based container service equipped with high availability, virtualized networking, security, and lifecycle management.



PKS uses Kubernetes on BOSH to simplify the deployment and operation of Kubernetes clusters. With Kubernetes on BOSH, PKS quickly and easily deploys Kubernetes clusters on VMware vSphere, giving you the opportunity to use Kubernetes.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com