THE
ENTERPRISERS
PROJECT

Kubernetes: Everything you need to know

Supported by
Red Hat

Kubernetes: Everything you need to know

What is Kubernetes?

What is Kubernetes and what does it have to do with containers? Where did this unusual word come from? The agreed-upon origin is from the Greek, meaning "helmsman" or "sailing master."

Here's how Red Hat technology evangelist Gordon Haff explains Kubernetes in his book, "From Pots and Vats to Programs and Apps," co-authored with Red Hat Senior Distinguished Engineer William Henry:

Kubernetes eliminates many of the manual processes involved in deploying and scaling containerized applications.

"Kubernetes, or k8s (k, 8 characters, s... get it?), or 'kube' if you're into brevity, is an open source platform that automates Linux container operations. It eliminates many of the manual processes involved in deploying and scaling containerized applications," Haff and Henry write. "In other words, you can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters."

Here's how Dan Kohn, executive director of the Cloud Native Computing Foundation (CNCF), in a podcast with Gordon Haff, explained it: "Containerization is this trend that's taking over the world to allow people to run all kinds of different applications in a variety of different environments. When they do that, they need an orchestration solution in order to keep track of all of those containers and schedule them and orchestrate them. Kubernetes is an increasingly popular way to do that."

The most recent version of Kubernetes, 1.19, was released in August 2020. (This frequently-updated project has had releases on about a quarterly basis recently.)

What is Kubernetes used for?

Containers appeal to organizations for a broad range of workloads. But provisioning and operationalizing containers at scale, often in concert with microservices, is not for weekend enthusiasts. Especially for stateful apps (such as databases), it requires planning, and most experts say an orchestration tool is a must. That's where Kubernetes comes in.

Containers, in concert with Kubernetes, are helping enterprises better manage workloads and reduce risks. In organizations using DevOps practices – including short development sprints, experimentation, and iteration – containers can be key to the evolution of processes, and to an organization's increasing usage of cloud infrastructure and microservices.

"Once organizations understand the benefits of containers and Kubernetes for DevOps, application development, and delivery, it opens up so many possibilities, from modernizing traditional applications, to hybrid- and multi-cloud implementations and the development of new, cloud-native applications with speed and agility," says Ashesh Badani, SVP and general manager for cloud platforms at Red Hat.

Want a plain English way to explain what this looks like? This one's pretty great: You can use a lunchbox analogy, notes Mike Kail, CTO and cofounder at CYBRIC: "Let's say an application environment is your old-school lunchbox. The contents of the lunchbox were all assembled well before putting them into the lunchbox [but] there was no isolation between any of those contents. The Kubernetes system provides a lunchbox that allows for just-in-time expansion of the contents (scaling) and full isolation between every unique item in the lunchbox and the ability to remove any item without affecting any of the other contents (immutability)."

Why use Kubernetes and containers?

Maybe you're trying to help people in your organization understand why Kubernetes – and orchestration tools in general – are necessary in the first place.

Kubernetes lets you schedule and run containers on clusters of physical or virtual machines, while automating many operational tasks. In other words, Kubernetes helps enterprises tap into the potential of containers in day-to-day work, in automated fashion. It also helps with load balancing and ensuring high-availability environments.

Many organizations find the Kubernetes platform becomes essential when you start deploying containers in significant numbers, especially in production environments.

Thus the many marketplace statistical signals indicating growing adoption. "As more and more organizations continue to expand on their usage of containerized software, Kubernetes will increasingly become the de facto deployment and orchestration target moving forward," says Josh Komoroske, senior DevOps engineer at StackRox.

Kubernetes has earned wide tech industry support and benefits from an active open source community.

When making a case for Kubernetes, you can pitch orchestration as a means of effectively managing containers (and, increasingly, containerized microservices) and Kubernetes as the right platform for doing so. It boils down to this: Using orchestration enables greater automation, repeatability, and definability within your environment, while reducing a potentially crushing burden of manual work throughout the continer lifecycle and application lifecycle, especially as your container adoption grows and as you work with multiple cloud providers.

"Kubernetes is one of those rare technologies that appeals to developers, operations teams, and lines of business. It's a win-win-win situation that offers benefits to all constituents in terms of productivity, collaboration, and meeting the needs of customers," says Red Hat's Badani.

"Without an orchestration framework of some sort, you've just got services running 'somewhere' – where you set them to run, manually – and if you lose a node or something crashes, it's manual [work] to fix it," adds Sean Suchter, co-founder and CTO of Pepperdata. "With an orchestration framework, you declare how you want your environment to look, and the framework makes it look like that."

Orchestration unlocks one of the big-picture promises of microservices: Enabling small teams to solve big problems.

Some other pluses of Kubernetes: It has earned wide tech industry support and benefits from an active open source community. For more, read also: How to make the case for Kubernetes.

What Kubernetes does

The power of the open source cloud-native ecosystem comes from the breadth of complementary projects that come together around Kubernetes.

Kubernetes is an important piece of the cloud-native puzzle: But it's important to understand that its broader ecosystem provides even more value to IT organizations. As Red

Hat's Haff notes, "The power of the open source cloud-native ecosystem comes only in part from individual projects such as Kubernetes. It derives, perhaps even more, from the breadth of complementary projects that come together to create a true cloud-native platform."

This includes service meshes like Istio, monitoring tools like Prometheus, command-line tools like Podman, distributed tracing from the likes of Jaeger and Kiali, enterprise registries like Quay, and inspection utilities like Skopeo, says Haff. And, of course, Linux, which is the foundation for the containers orchestrated by Kubernetes.

Choosing from and integrating a variety of tools yourself takes time of course, which is one place where enterprise open source platforms, such as Red Hat OpenShift come into play.

[Read also: OpenShift and Kubernetes: What's the difference?]

Kubernetes eases the burden of configuring, deploying, managing, and monitoring even the largest-scale containerized applications.

In many organizations, the first step toward Kubernetes adoption to date might be best described as *Oh, we can use Kubernetes for this!* That means, for example, that a team running a growing number of containers in production might quickly see the need for orchestration to manage it all.

StackRox's Komoroske expects another adoption trend to grow in the near future: *We can build this for Kubernetes!* It's the software equivalent of a cart-and-horse situation: Instead of having an after-the-fact revelation that Kubernetes would be a good fit for managing a particular service, more organizations will develop software specifically with Kubernetes in mind. Some people will call this "Kubernetes-native" software.

Key Kubernetes trends for 2020

1. Expect a rising tide of "Kubernetes-native" software: Think "not only containerized software that happens to be deployable in Kubernetes, but also software that is aware of and able to provide unique value when deployed in Kubernetes," Komoroske says.

"Software that is released and branded as 'Kubernetes-first' will be increasingly common, possibly manifesting as custom resources definitions or Kubernetes Operators," Komoroske says.

2. Security will continue to be a high-profile focus: "As the adoption of Kubernetes and deployment of container-based applications in production accelerate to much higher volumes than we've seen to date, we can expect more security incidents to occur," says Rani Osnat, VP

of strategy at Aqua Security. "Most of those will be caused by the knowledge gap around what constitutes secure configuration, and lack of proper security tooling."

It's not that Kubernetes has inherent security issues, per se. In fact, there's a visible commitment to security in the community. It simply comes with some new considerations and strategies for managing risks, as bad actors are getting better at spotting vulnerabilities.

Other key trends include:

- A move toward federation
- A scramble for talent
- Efforts to reduce Kubernetes' resource consumption.

For full detail on these trends, read 5 Kubernetes trends to watch in 2020.

Kubernetes terms defined: Operators, secrets, kubectl, minikube, and more

What is a Kubernetes operator?

The conventional wisdom of Kubernetes' earlier days was that it was very good at managing stateless apps. But for stateful applications such as databases, it wasn't such an open-and-shut case: These apps required more hand-holding, says Jeremy Thompson, CTO at Solodev.

"Adding or removing instances may require preparation and/or post-provisioning steps – for instance, changes to its internal configuration, communication with a clustering mechanism, interaction with external systems like DNS, and so forth," Thompson explains. "Historically, this often required manual intervention, increasing the DevOps burden and increasing the likelihood of error. Perhaps most importantly, it obviates one of Kubernetes' main selling points: automation."

That's a big problem. Fortunately, the solution emerged back in 2016, when coreOS introduced Operators to extend Kubernetes' capabilities to stateful applications. (Red Hat acquired coreOS in January 2018, expanding the capabilities of the OpenShift container platform.)

Operators became even more powerful with the launch of the Operator Framework for building and managing Kubernetes native applications (Operators by another name) in March 2018.

"Operators are clients of the Kubernetes API that control custom resources," says Matthew Dresden, director of DevOps at Nexient. "This capability enables automation of tasks like deployments, backups, and upgrades by watching events without editing Kubernetes code."

As Red Hat product manager Rob Szumski notes in a blog, "The key attribute of an Operator is the active, ongoing management of the application, including failover, backups, upgrades, and autoscaling, just like a cloud service. Of course, if your app doesn't store stateful data, a backup might not be applicable to you, but log processing or alerting might be important. The important user experience that the Operator model aims for is getting that cloud-like, self-managing experience with knowledge baked in from the experts."

If you can't fully automate, you're undermining the potential of containers and other cloud-native technologies.

Want to find or share operators? Meet OperatorHub.io

There's been a noticeable bump in the interest in and implementation of Operators of late, according to Liz Rice, VP of open source engineering at Aqua Security. Rice also chairs the Cloud Native Computing Foundation's technical oversight committee.

"At the CNCF, we're seeing interest in projects related to managing and discovering Kubernetes Operators, as well as observing an explosion in the number of Operators being implemented," Rice says. "Project maintainers and vendors are building Operators to make it easier for people to use their projects or products within a Kubernetes deployment."

This growing menu of Operators means there's a need for a, well, menu. "This proliferation of Operators has created a gap for directories or discovery mechanisms to help people find and easily install what's available," Rice says.

The relatively new OperatorHub.io is one place where Kubernetes community members can find existing Operators or share their own. (Red Hat launched Operator Hub in conjunction with Amazon, Microsoft, and Google.)

What is a Kubernetes secret?

A Kubernetes secret is a cleverly named Kubernetes object that is one of the container orchestration platform's built-in security capabilities. A "secret" in Kubernetes is a means of storing sensitive information, like an OAuth token or SSH key, so that it's accessible when necessary to pods in your cluster but protected from unnecessary visibility that could create security risks.

As the Kubernetes documentation notes, "Putting this information in a Secret is safer and more flexible than putting it verbatim in a Pod definition or in a container image."

Secrets could be thought of as a relative of the least privilege principle, except instead of focusing on limiting the access of individual users to that which they actually do to get their work done, they focus on giving your applications the data they need to properly function without giving them (and the people that manage them) unfettered access to that data.

Put another way, Secrets help fulfill a technical requirement while solving a problem that rises out of that requirement: Your containerized applications need certain data or credentials to run properly, but how you store that data and make it available is the kind of thing that keeps security analysts up at night.

What is a Kubernetes cluster?

You can begin to understand this major piece literally: A cluster is a group or bunch of nodes that run your containerized applications. You manage the cluster and everything it includes – in other words, you manage your application(s) – with Kubernetes.

What is a Kubernetes pod?

This is essentially the smallest deployable unit of the Kubernetes ecosystem; more accurately, it's the smallest object. A pod specifically represents a group of one or more containers running together on your cluster.

What is a Kubernetes node?

Nodes are comprised of physical or virtual machines on your cluster; these "worker" machines have everything necessary to run your application containers, including the container runtime and other critical services. (The Kubernetes Github repository has a good, detailed breakdown of the Kubernetes node.)

What is kubectl?

Simply put, kubectl is a command line interface (CLI) for managing operations on your Kubernetes clusters. It does so by communicating with the Kubernetes API. (It's not a typo, either: The official Kubernetes style, as it were, is to lowercase the k in kubectl.) It follows a standard syntax for running commands: kubectl [command] [TYPE] [NAME] [flags]. You can find an in-depth explanation of kubectl here, as well as examples of common operations, but

here's a basic example of an operation: "run." This command runs a particular container image on your cluster.

What is a Kubernetes service?

A Kubernetes service is "an abstract way to expose an application running on a set of pods as a network service," as the Kubernetes documentation puts it. "Kubernetes gives pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them."

But pods sometimes have a short lifespan. As pods come and go, services help the other pods "find out and keep track of which IP address to connect to."

What is Minikube?

Minikube is an open source tool that enables you to run Kubernetes on your laptop or other local machine. It can work with Linux, Mac, and Windows operating systems. It runs a single-node cluster inside a virtual machine on your local machine.

In other words, Minikube takes the vast cloud-scale of Kubernetes and shrinks it down so that it fits on even your laptop. Don't mistake that for a lack of power or functionality, though: You can do plenty with Minikube. And while developers, DevOps engineers, and the like might be the most likely to run it on a regular basis, IT leaders and the C-suite can use it, too. That's part of the beauty.

"With just a few installation commands, anyone can have a fully functioning Kubernetes cluster, ready for learning or supporting development efforts," says Chris Ciborowski, CEO and cofounder at Nebulaworks.

The official Kubernetes documentation includes instructions for installing Minikube – note that you'll also need to install kubectl, the native command-line interface for Kubernetes. It also offers a quickstart guide for getting up and running.

Pro tip if you're using a RHEL/Fedora/CentOS workstation: Over at Opensource.com, Bryant Son wrote a great guide on getting started with Minikube tailored specifically for you.

Here are four ways you can use Minikube:

- Fast route to experimentation and learning - for developers and IT leaders
- Evaluate important Kubernetes features
- Play with Kubernetes' extensibility in a sandbox
- Do a Kubernetes proof of concept project

For more, see Minikube: 5 ways IT teams can use it.

Now, let's dig into Kubernetes tutorials; classes and books; security essentials; and best practices for building and migrating apps.

Kubernetes tutorials

"Minikube allows teams to experiment quickly and easily."

Minikube makes a great do-it-yourself learning opportunity. “To get hands-on with Kubernetes or to run a trial of it, the easiest way to get started is to use Minikube on one of the Linux OS flavors,” says Raghu Kishore Vempati, principal systems engineer for innovation at Altran. “Minikube allows teams to experiment quickly and easily. The setup is simple.”

Here are four other tutorials to consider:

Learn Kubernetes Basics tutorial – Kubernetes official site

When you’ve got the lingo, concepts, and emerging trends down pat, consider digging into this tutorial on the basics of Kubernetes orchestration via the official project site.

Learn Kubernetes Using Interactive Browser-Based Scenarios – Katacoda

This browser-based tool offers 17 hands-dirty scenarios for learning Kubernetes, including fundamentals such as deploying a container using the kubectl command-line interface and more advanced tasks such as Ingress routing. It also includes a “playground” environment for unstructured tinkering and learning.

Kubernetes by Example – Red Hat OpenShift

These step-by-step walk-throughs of Kubernetes concepts and capabilities, created by the Red Hat OpenShift team, includes commands for the kubectl command-line interface for various tasks and operations. These can then be replicated in DIY fashion, either in a local environment or in an online environment on openshift.com.

Getting started with Kubernetes - Opensource.com

Our sister site, Opensource.com, also offers a tutorial that will walk you through the basics of creating a cluster, deploying an app, and creating a proxy: See Getting started with Kubernetes.

Kubernetes classes and books

Kubernetes classes

Introduction to Kubernetes – edX

This edX course developed by The Linux Foundation functions like a "101" course for people and teams new to the tool. The edX course page includes a bullet-point syllabus for the topics covered, including (near the end of the class) the value of the Kubernetes community and how to get involved. The course is free, with a paid option ($99) for those who want a verified certificate of successful completion.

Scalable Microservices with Kubernetes – Udacity

This free course introduces the ins and outs of managing containerized applications with Kubernetes, especially in the context of today's 24-7 expectations for applications and services and the demands those expectations place on infrastructure.

Deploying Containerized Applications Technical Overview - Red Hat

This free course is a series of on-demand, online videos that introduces you to Linux containers and container orchestration technology. In these short lectures and in-depth demonstrations, you will learn about containerizing applications and services, testing them, and deploying them on a Kubernetes cluster using Red Hat OpenShift. You will also learn how to build and deploy an application from source code using the source-to-image facility of OpenShift.

Kubernetes books

O'Reilly: Kubernetes Operators: Automating the Container Orchestration Platform

Want to learn more about building and deploying Operators? Get this free eBook.

O'Reilly: Kubernetes patterns for designing cloud-native apps

In this free eBook aimed at developers, get detailed, reusable Kubernetes patterns for container deployment and orchestration. Learn everything Kubernetes offers for each particular pattern, with tested conclusions for each concept and full code examples.

Kubernetes: Up and Running: Dive into the Future of Infrastructure

This book (available in both electronic and physical editions) is considered one of the better introductions to Kubernetes fundamentals, especially for beginning audiences. It's written by

noted K8s expert Kelsey Hightower along with two of the orchestrator's original creators at Google: Brendan Burns and Joe Beda.

Kubernetes security: What you need to know

Red Hat security strategist Kirsten Newcomer encourages people to think of container security as having ten layers – including both the container stack layers (such as the container host and registries) and container lifecycle issues (such as API management). For complete details on the ten layers and how orchestration tools such as Kubernetes fit in, check out this podcast with Newcomer, or this whitepaper: Ten Layers of Container Security.

Here are 3 Kubernetes security areas for your teams to focus on:

- Application and environment misconfigurations
- Poor container security hygiene
- Production deployments expose misconfigurations and other vulnerabilities

Taking a DevSecOps approach – which bakes security into dev processes from the start - helps, as does active participation in the Kubernetes community. For more security tips, read also:

Kubernetes security: 4 tips to manage risks.

Kubernetes security: 5 mistakes to avoid

6 Kubernetes security questions, answered

7 best practices: Building applications for containers and Kubernetes

Don't let the growing popularity of containers and Kubernetes dupe you into thinking that you should use them to run any and every type of application. You need to distinguish between "can" and "should."

One basic example of this distinction is the difference between building an app specifically to be run in containers and operated with Kubernetes (some would refer to this as cloud-native development) and using these containers and orchestration for existing monolithic apps.

Building new applications specifically for containers and Kubernetes might be the better starting point for teams just beginning with containers and orchestration.

Here are seven best practices to keep in mind:

1. Think and build modern: Think microservices, for example. Define container images as logical units that can scale independently. Consider cloud-native APIs.

2. CI/CD and automation are your friends: A well-conceived CI/CD pipeline is an increasingly popular approach to baking automation into as many phases of your development and deployment processes as possible. Check out our recent primer for IT leaders: How to build a CI/CD pipeline.

3. Keep container images as light as possible: Keep your container images as small as possible for performance, security, and other reasons. Only include what you absolutely need. Remove all other packages – including shell utilities – that are not required by the containerized application.

4. Don't blindly trust images: If you're going to grab a container image rather than build it from scratch, don't have blind faith in its security. Any images you use, even ones in your own repositories, should be scanned for vulnerabilities and compliance, experts advise.

5. Plan for observability, telemetry, and monitoring from the start: Kubernetes' self-healing capabilities are a piece of the platform's appeal, but they also underscore the need for proper visibility into your applications and environments. This is where observability, telemetry, and monitoring become key.

6. Consider starting with stateless applications: One early line of thinking about containers and Kubernetes has been that running stateless apps is a lot easier than running stateful apps (such as databases). That's changing with the growth of Kubernetes Operators, but teams new to Kubernetes might still be better served by beginning with stateless applications.

7. Remember, this is hard: "None of the abstractions that exist in Kubernetes today make the underlying systems any easier to understand. They only make them easier to use," says Chris Short, Red Hat OpenShift principal technical marketing manager. Your teams should be ready to learn from mistakes, Short notes.

For full detail on each of these seven best practices, read 7 best practices: Building applications for containers and Kubernetes. Want to migrate existing apps rather than build from scratch? Read Migrating applications to containers and Kubernetes: 5 best practices.

Kubernetes resources: Learn more

Check out these eBooks and articles, for even more learning on Kubernetes, and share with your team:

Try out Kubernetes: Try Kubernetes for two ways to set up and run.

eBook: Getting Started with Kubernetes

eBook: O'Reilly: Kubernetes Operators: Automating the Container Orchestration Platform

eBook: O'Reilly: Kubernetes patterns for designing cloud-native apps

Kubernetes glossary cheat sheet: 10 key concepts in plain English

Containers primer: Learn the lingo of Linux containers

Articles:

What is Kubernetes?

Kubernetes by the numbers: 13 compelling stats

Kubernetes: 6 secrets of successful teams

Minikube, Kubernetes' best friend: 6 facts to know

How to explain Kubernetes Operators in plain English

Kubernetes Operators: 4 facts to know

Kubernetes: 3 ways to get started

How to make the case for Kubernetes

Kubernetes jobs hunt: How to land that role

14 Kubernetes interview questions: For hiring managers and job seekers

5 interview questions every Kubernetes job candidate should know

Running Kubernetes on your Raspberry Pi homelab

Developing applications on Kubernetes

How to run a Kubernetes cluster on your laptop

Deep dive: Understanding Kubernetes for enterprises