CONTAINERS

What is Kubernetes?

Kubernetes (also known as k8s or "kube") is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications.

In other words, you can cluster together groups of hosts running Linux® containers, and Kubernetes helps you easily and efficiently manage those clusters.

Kubernetes clusters can span hosts across on-premise, public, private, or hybrid clouds. For this reason, Kubernetes is an ideal platform for hosting cloud-native applications that require rapid scaling, like real-time data streaming through Apache Kafka.

Kubernetes was originally developed and designed by engineers at Google. Google was one of the early contributors to Linux container technology and has talked publicly about how everything at Google runs in containers. (This is the technology behind Google's cloud services.)

Google generates more than 2 billion container deployments a week, all powered by its internal platform, Borg. Borg was the predecessor to Kubernetes, and the lessons learned from developing Borg over the years became the primary influence behind much of Kubernetes technology.

Fun fact: The 7 spokes in the Kubernetes logo refer to the project's original name, "Project Seven of Nine."

Red Hat was one of the first companies to work with Google on Kubernetes, even prior to launch, and has become the 2nd leading contributor to the Kubernetes upstream project. Google donatedthe Kubernetes project to the newly formed Cloud Native Computing Foundation (CNCF) in 2015.

Get an introduction to enterprise Kubernetes

What can you do with Kubernetes?

The primary advantage of using Kubernetes in your environment, especially if you are optimizing app dev for the cloud, is that it gives you the platform to schedule and run containers on clusters of physical or virtual machines(VMs).

More broadly, it helps you fully implement and rely on a container-based infrastructure in production environments. And because Kubernetes is all about automation of operational tasks, you can do many of the same things other application platforms or management systems let you do—but for your containers.

Developers can also create cloud-native apps with Kubernetes as a runtime platform by using Kubernetes patterns. Patterns are the tools a Kubernetes developer needs to build container-based applications and services.

With Kubernetes you can:

- Orchestrate containers across multiple hosts.
- Make better use of hardware to maximize resources needed to run your enterprise apps.
- Control and automate application deployments and updates.
- Mount and add storage to run stateful apps.
- Scale containerized applications and their resources on the fly.
- Declaratively manage services, which guarantees the deployed applications are always running the way you intended them to run.
- Health-check and self-heal your apps with autoplacement, autorestart, autoreplication, and autoscaling.

However, Kubernetes relies on other projects to fully provide these orchestrated services. With the addition of other open source projects, you can fully realize the power of Kubernetes. These necessary pieces include (among others):

- Registry, through projects like Docker Registry.
- Networking, through projects like OpenvSwitch and intelligent edge routing.
- Telemetry, through projects such as Kibana, Hawkular, and Elastic.
- Security, through projects like LDAP, SELinux, RBAC, and OAUTH with multitenancy layers.
- Automation, with the addition of Ansible playbooks for installation and cluster life cycle management.
- Services, through a rich catalog of popular app patterns.

Get an introduction to Linux containers and container orchestration technology. In this on-demand course, you'll learn about containerizing applications and services, testing them using Docker, and deploying them on a Kubernetes cluster using Red Hat® OpenShift®.

Start the free training course

Learn to speak Kubernetes

As is the case with most technologies, language specific to Kubernetes can act as a barrier to entry. Let's break down some of the more common terms to help you better understand Kubernetes.

Control plane: The collection of processes that control Kubernetes nodes. This is where all task assignments originate.

Nodes: These machines perform the requested tasks assigned by the control plane.

Pod: A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources. Pods abstract network and storage from the underlying container. This lets you move containers around the cluster more easily.

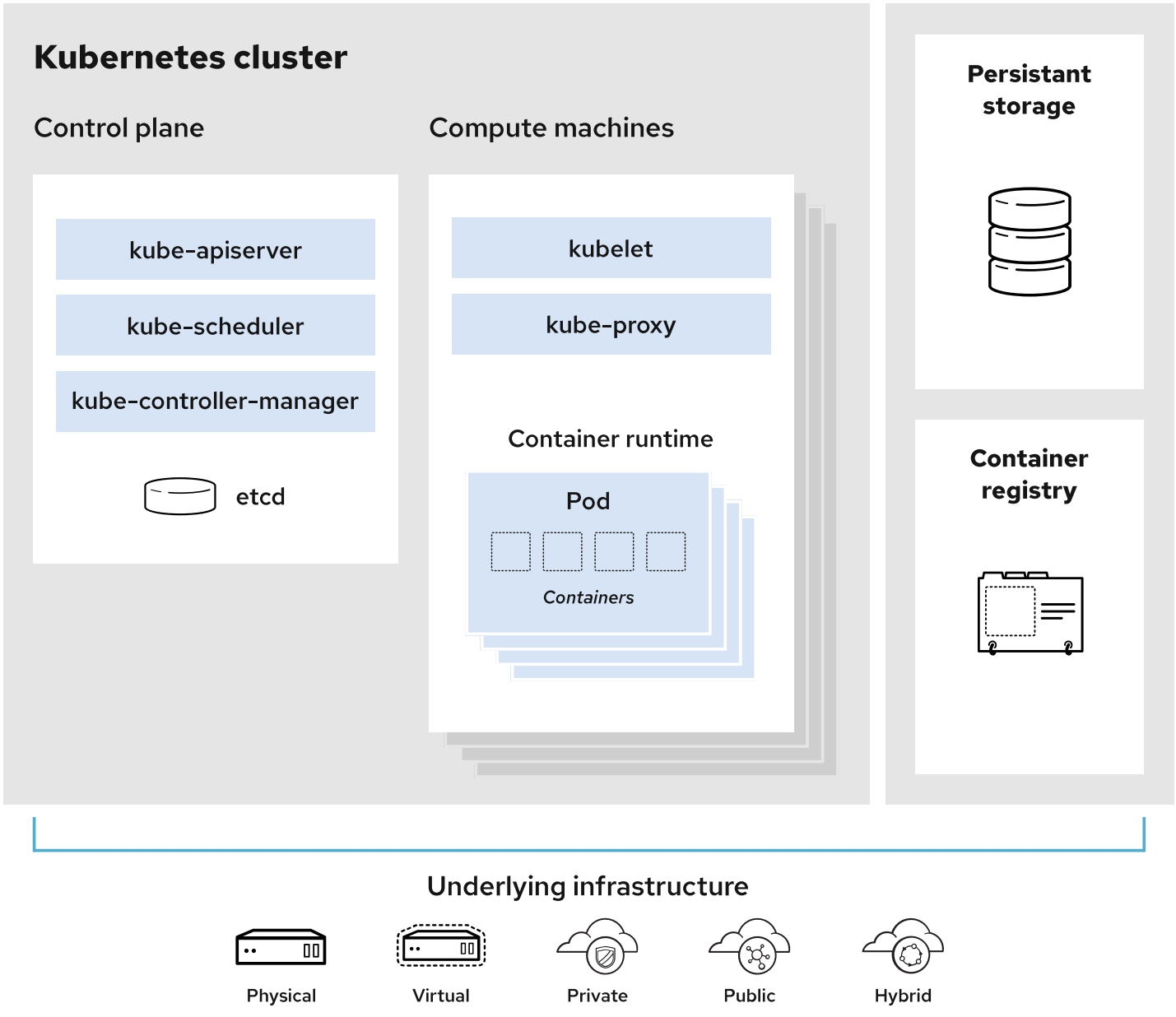
Replication controller: This controls how many identical copies of a pod should be running somewhere on the cluster.

Service: This decouples work definitions from the pods. Kubernetes service proxies automatically get service requests to the right pod—no matter where it moves in the cluster or even if it's been replaced.

Kubelet: This service runs on nodes, reads the container manifests, and ensures the defined containers are started and running.

kubectl: The command line configuration tool for Kubernetes.

How does Kubernetes work?



A working Kubernetes deployment is called a cluster. You can visualize a Kubernetes cluster as two parts: the control plane and the compute machines, or nodes.

Each node is its own Linux® environment, and could be either a physical or virtual machine. Each node runs pods, which are made up of containers.

The control plane is responsible for maintaining the desired state of the cluster, such as which applications are running and which container images they use. Compute machines actually run the applications and workloads.

Kubernetes runs on top of an operating system (Red Hat® Enterprise Linux®, for example) and interacts with pods of containers running on the nodes.

The Kubernetes control plane takes the commands from an administrator (or DevOps team) and relays those instructions to the compute machines.

This handoff works with a multitude of services to automatically decide which node is best suited for the task. It then allocates resources and assigns the pods in that node to fulfill the requested work.

The desired state of a Kubernetes cluster defines which applications or other workloads should be running, along with which images they use, which resources should be made available to them, and other such configuration details.

From an infrastructure point of view, there is little change to how you manage containers. Your control over containers just happens at a higher level, giving you better control without the need to micromanage each separate container or node.

Your work involves configuring Kubernetes and defining nodes, pods, and the containers within them. Kubernetes handles orchestrating the containers.

Where you run Kubernetes is up to you. This can be on bare metal servers, virtual machines, public cloud providers, private clouds, and hybrid cloud environments. One of Kubernetes' key advantages is it works on many different kinds of infrastructure.

Learn about the other components of a Kubernetes architecture

What about Docker?

Docker can be used as a container runtime that Kubernetes orchestrates. When Kubernetes schedules a pod to a node, the kubelet on that node will instruct Docker to launch the specified containers.

The kubelet then continuously collects the status of those containers from Docker and aggregates that information in the control plane. Docker pulls containers onto that node and starts and stops those containers.

The difference when using Kubernetes with Docker is that an automated system asks Docker to do those things instead of the admin doing so manually on all nodes for all containers.

Why do you need Kubernetes?

Kubernetes can help you deliver and manage containerized, legacy, and cloud-native apps, as well as those being refactored into microservices.

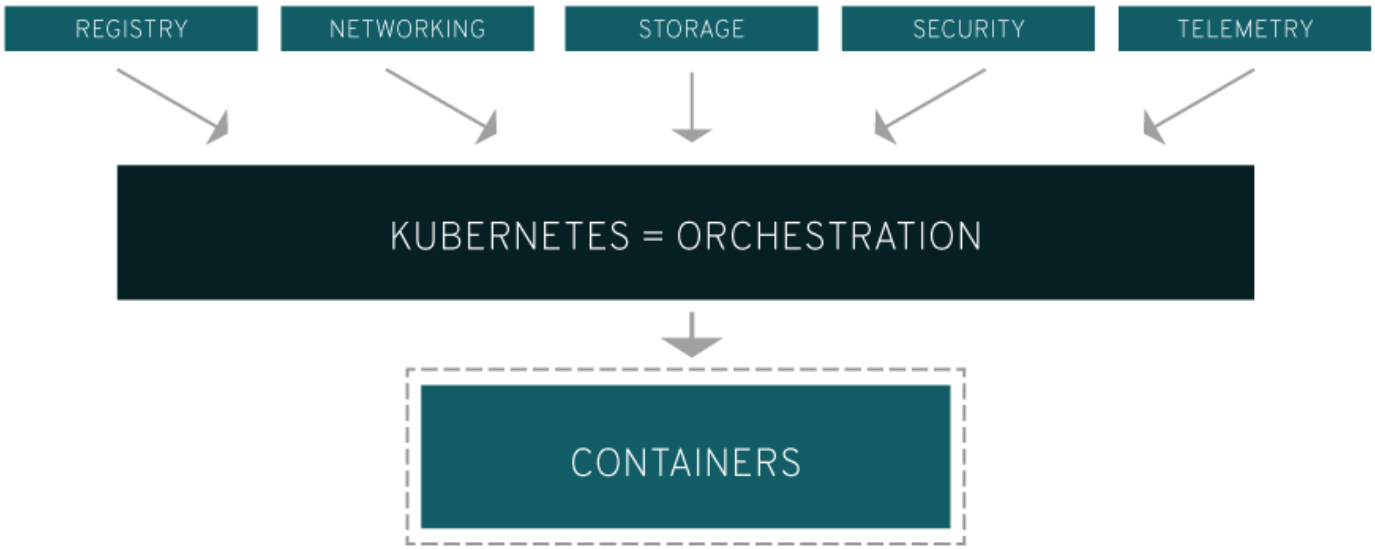
In order to meet changing business needs, your development team needs to be able to rapidly build new applications and services. Cloud-native development starts with microservices in containers, which enables faster development and makes it easier to transform and optimize existing applications.

Watch this webinar series to get expert perspectives to help you establish the data platform on enterprise Kubernetes you need to build, run, deploy, and modernize applications.

Production apps span multiple containers, and those containers must be deployed across multiple server hosts. Kubernetes gives you the orchestration and management capabilities required to deploy containers, at scale, for these workloads.

Kubernetes orchestration allows you to build application services that span multiple containers, schedule those containers across a cluster, scale those containers, and manage the health of those containers over time. With Kubernetes you can take effective steps toward better IT security.

Kubernetes also needs to integrate with networking, storage, security, telemetry, and other services to provide a comprehensive container infrastructure.



Once you scale this to a production environment and multiple applications, it's clear that you need multiple, colocated containers working together to deliver the individual services.

Linux containers give your microservice-based apps an ideal application deployment unit and self-contained execution environment. And microservices in containers make it easier to orchestrate services, including storage, networking, and security.

This significantly multiplies the number of containers in your environment, and as those containers accumulate, the complexity also grows.

Kubernetes fixes a lot of common problems with container proliferation by sorting containers together into "pods." Pods add a layer of abstraction to grouped containers, which helps you schedule workloads and provide necessary services—like networking and storage—to those containers.

Other parts of Kubernetes help you balance loads across these pods and ensure you have the right number of containers running to support your workloads.

With the right implementation of Kubernetes—and with the help of other open source projects like Open vSwitch, OAuth, and SELinux— you can orchestrate all parts of your container infrastructure.

Use case: Building a cloud platform to offer innovative banking services

Emirates NBD, one of the largest banks in the United Arab Emirates (UAE), needed a scalable, resilient foundation for digital innovation. The bank struggled with slow provisioning and a complex IT environment. Setting up a server could take 2 months, while making changes to large, monolithic applications took more than 6 months.

Using Red Hat OpenShift Container Platform for container orchestration, integration, and management, the bank created Sahab, the first private cloud run at scale by a bank in the Middle East. Sahab provides applications, systems, and other resources for end-to-end development—from provisioning to production—through an as-a-Service model.

With its new platform, Emirates NBD improved collaboration between internal teams and with partners using application programming interfaces (APIs) and microservices. And by adopting agileand DevOps development practices, the bank reduced app launch and update cycles.

Read the full case study

Support a DevOps approach with Kubernetes

Developing modern applications requires different processes than the approaches of the past. DevOps speeds up how an idea goes from development to deployment.

At its core, DevOps relies on automating routine operational tasks and standardizing environments across an app's lifecycle. Containers support a unified environment for development, delivery, and automation, and make it easier to move apps between development, testing, and production environments.

A major outcome of implementing DevOps is a continuous integration and continuous deployment pipeline (CI/CD). CI/CD helps you deliver apps to customers frequently and validate software quality with minimal human intervention.

Managing the lifecycle of containers with Kubernetes alongside a DevOps approach helps to align software development and IT operations to support a CI/CD pipeline.

With the right platforms, both inside and outside the container, you can best take advantage of the culture and process changes you've implemented.

Learn more about how to implement a DevOps approach

Using Kubernetes in production

Kubernetes is open source and as such, there's not a formalized support structure around that technology—at least not one you'd trust your business to run on.

If you had an issue with your implementation of Kubernetes while running in production, you'd likely be frustrated. And your customers would be, too.

Think of Kubernetes like a car engine. An engine can run on its own, but it becomes part of a functional car when it's connected with a transmission, axles, and wheels. Just installing Kubernetes is not enough to have a production-grade platform.

Kubernetes needs additional components to become fully functional. You'll need to add authentication, networking, security, monitoring, logs management, and other tools.

That's where Red Hat OpenShift comes in—it's the complete car.

Red Hat OpenShift is Kubernetes for the enterprise. It includes all the extra pieces of technology that make Kubernetes powerful and viable for the enterprise, including registry, networking, telemetry, security, automation, and services.

Red Hat OpenShift includes Kubernetes as a central component of the platform and is a certified Kubernetes offering by the CNCF.

With Red Hat OpenShift Container Platform, your developers can make new containerized apps, host them, and deploy them in the cloud with the scalability, control, and orchestration that can turn a good idea into new business quickly and easily.

You can try using Red Hat OpenShift to automate your container operations with a free 60-day trial.