**What is Kubernetes?**

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](https://www.redhat.com/en/topics/containers/whats-a-linux-container) operations. It eliminates many of the manual processes involved in deploying 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. These clusters can span hosts across [public](https://www.redhat.com/en/topics/cloud-computing/what-is-public-cloud), [private](https://www.redhat.com/en/topics/cloud-computing/what-is-private-cloud), or [hybrid clouds.](https://www.redhat.com/en/topics/cloud-computing/what-is-hybrid-cloud)

Kubernetes was originally developed and designed by engineers at Google. Google was one of the [early contributors to Linux container technology](https://en.wikipedia.org/wiki/Cgroups) and has talked publicly about how [everything at Google runs in containers](https://speakerdeck.com/jbeda/containers-at-scale). (This is the technology behind Google’s cloud services.) Google generates more than 2 billion container deployments a week—all powered by an internal platform: [Borg](http://blog.kubernetes.io/2015/04/borg-predecessor-to-kubernetes.html). Borg was the predecessor to Kubernetes and the lessons learned from developing Borg over the years became the primary influence behind much of the Kubernetes technology.

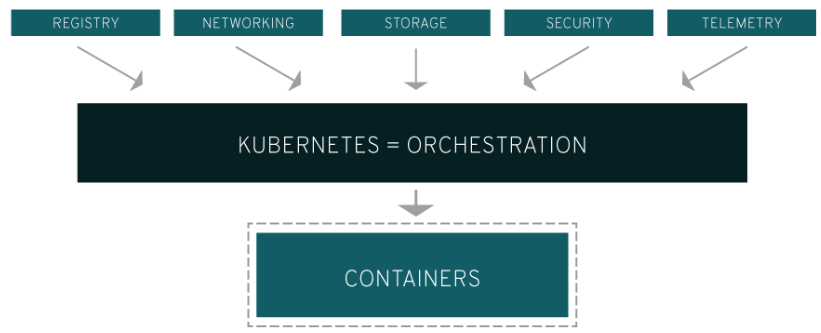
*Fun fact: The seven spokes in the Kubernetes logo refer to the project’s original name, “*[*Project Seven of Nine*](https://cloudplatform.googleblog.com/2016/07/from-Google-to-the-world-the-Kubernetes-origin-story.html)*.”*

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](http://stackalytics.com/?project_type=kubernetes-group&metric=commits) to Kubernetes upstream project. Google [donated](https://techcrunch.com/2015/07/21/as-kubernetes-hits-1-0-google-donates-technology-to-newly-formed-cloud-native-computing-foundation-with-ibm-intel-twitter-and-others/) the Kubernetes project to the newly formed [Cloud Native Computing Foundation](https://www.cncf.io/) in 2015.

**Why do you need Kubernetes?**

Real production apps span multiple containers. 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.

Kubernetes also needs to integrate with networking, [storage](https://www.redhat.com/en/topics/data-storage), security, telemetry and other services to provide a comprehensive container infrastructure.



Of course, this depends on how you’re using containers in your environment. A rudimentary application of Linux containers treats them as efficient, fast virtual machines. 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. 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—sorting containers together into a ”pod.” 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 load balance 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 [Atomic Registry](http://www.projectatomic.io/), [Open vSwitch](http://openvswitch.org/), [heapster](https://github.com/kubernetes/heapster), [OAuth](https://oauth.net/), and [SELinux](https://selinuxproject.org/page/Main_Page)— you can orchestrate all parts of your container infrastructure.

[**Learn what containers can do for the enterprise**](https://www.redhat.com/en/explore/enterprise-container-platform)

**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](https://www.redhat.com/en/topics/cloud-native-apps), is that it gives you the platform to schedule and run containers on clusters of physical or virtual machines. 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 that other application platforms or management systems let you do, but for your containers.

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 how you deployed them.
* Health-check and self-heal your apps with autoplacement, autorestart, autoreplication, and autoscaling.

Kubernetes, however, 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 Atomic Registry or Docker Registry.
* Networking, through projects like OpenvSwitch and intelligent edge routing.
* Telemetry, through projects such as heapster, kibana, hawkular, and elastic.
* Security, through projects like LDAP, SELinux, RBAC, and OAUTH with multi-tenancy layers.
* Automation, with the addition of Ansible playbooks for installation and cluster life-cycle management.
* Services, through a rich catalog of precreated content of popular app patterns.

[**Get all of this, prebuilt and ready to deploy, with Red Hat OpenShift**](https://www.redhat.com/en/technologies/cloud-computing/openshift)

**Learn to speak Kubernetes**

Like any technology, there are a lot of words specific to the technology that can be a barrier to entry. Let's break down some of the more common terms to help you understand Kubernetes.

**Master:** The machine that controls Kubernetes nodes. This is where all task assignments originate.

**Node:** These machines perform the requested, assigned tasks. The Kubernetes master controls them.

**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 away 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 to in the cluster or even if it’s been replaced.

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

**kubectl:** This is the command line configuration tool for Kubernetes.

[**Had enough? No? Check out the Kubernetes glossary.**](https://kubernetes.io/docs/reference/)

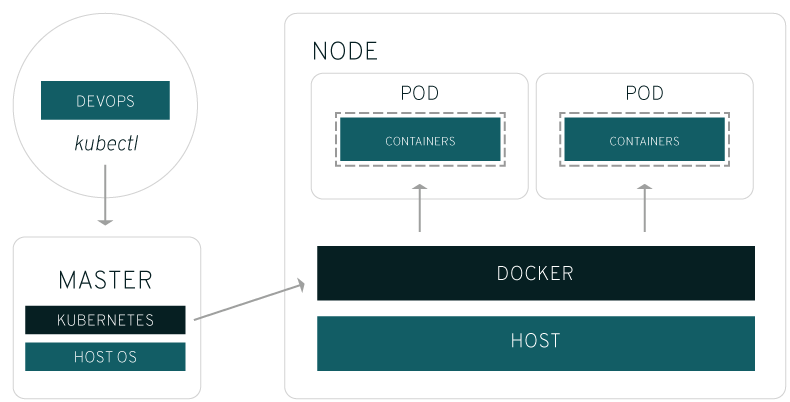
**Using Kubernetes in production**

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 on. If you had an issue with your implementation of Kubernetes, while running in production, you’re not going to be very happy. And your customers probably won’t, either.

That’s where [Red Hat OpenShift](https://www.redhat.com/en/technologies/cloud-computing/openshift) comes in. OpenShift is Kubernetes for the enterprise—and a lot more. OpenShift includes all of the extra pieces of technology that makes Kubernetes powerful and viable for the enterprise, including: registry, networking, telemetry, security, automation, and services. With OpenShift, 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.

Best of all, OpenShift is supported and developed by the #1 leader in open source, Red Hat.

**A look at how Kubernetes fits into your infrastructure**



Kubernetes runs on top of an operating system ([Red Hat Enterprise Linux Atomic Host](https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux), for example) and interacts with pods of containers running on the nodes. The Kubernetes master takes the commands from an administrator (or DevOps team) and relays those instructions to the subservient nodes. 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.

So, from an infrastructure point of view, there is little change to how you’ve been managing containers. Your control over those containers happens at a higher level, giving you better control without the need to micromanage each separate container or node. Some work is necessary, but it’s mostly a question of assigning a Kubernetes master, defining nodes, and defining pods.

**What about docker?**

The [docker](https://www.redhat.com/en/topics/containers/what-is-docker) technology still does what it's meant to do. 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 master. Docker pulls containers onto that node and starts and stops those containers as normal. The difference is that an automated system asks docker to do those things instead of the admin doing so by hand on all nodes for all containers.