RED HAT® OPENSHIFT

redhat.

TECHNOLOGY DETAIL

# AN INTRODUCTION TO ENTERPRISE KUBERNETES

## INTRODUCTION

Containers are transforming the way we think about application architecture and the speed at which teams can deliver on business requirements. They promise application portability across hybrid cloud environments and allow developers to focus on building a great product, without the distraction of underlying infrastructure or execution details.

While containers remove complexity through minimalism, managing container deployments presents new challenges for IT operations teams. Containers are deployed for much shorter periods of time than virtual machines (VMs), with greater utilization of underlying resources. Related container technologies must manage far more objects with greater turnover, introducing the need for more automated, policy-driven management.

Many teams are turning to Kubernetes and its rich set of complex features to help them orchestrate and manage containers in production, development, and test environments. Kubernetes has emerged as the de facto standard for container orchestration and management, becoming a critical platform for organizations to understand.

## WHAT IS KUBERNETES?

Kubernetes is an open source container orchestration platform that helps manage distributed, containerized applications at massive scale. You tell Kubernetes where you want your software to run, and the platform takes care of virtually everything else.

Kubernetes provides a unified application programming interface (API) to deploy web applications, batch jobs, and databases. Applications in Kubernetes are packaged in containers and are cleanly decoupled from their environment. Kubernetes automates the configuration of your applications and maintains and tracks resource allocation.

## BORG, OMEGA, AND THE ORIGIN OF KUBERNETES

While widespread interest in and adoption of containers is relatively new, containers have been around for decades (albeit in a much smaller capacity). Google was one of the first organizations to run containers at massive scale, starting well before it made Kubernetes open source in 2014.

Kubernetes is the product of more than 15 years of Google's experience in one of the most demanding production environments in the history of compute. It began with Borg, an internal Google project built to manage long-running services and batch jobs, replacing two separate systems. As Google developed more applications to run on top of Borg, the company created a broad and intelligent ecosystem of tools and services for:

- Autoscaling.
- Self-healing infrastructure.
- Configuration and updating of batch jobs.
- Service discovery and load balancing.
- Application life-cycle management.
- Quota management.

Kubernetes is the third generation of the Borg project. The second generation is called Omega, which is the baseline of Apache Mesos.

Omega was developed to improve upon the engineering of the Borg ecosystem. It focused on a more intelligent scheduler that was needed to handle an increasing number of diverse jobs. Google needed a system to schedule workloads, an increasingly complex task given the volume of applications and long-running services. With Omega, the scheduler was divided into two separate schedulers with a central shared-state repository to mitigate conflicts. This solution worked but was complex; a better system was needed.

Kubernetes is Google's third container management system. It is a combination of the approaches used in the monolithic Borg controller and the more flexible Omega controller. Kubernetes was designed to remove complexity and simplify the management of massive infrastructures. Since its launch, the Kubernetes project has fostered a highly dedicated developer community that continues to support and contribute to its growth.

## CONTAINERS AND MODERN APPLICATION DEVELOPMENT

Containers are popular because they allow an application to run as an isolated process with its own resources. Containers are unlike VMs because they allow the Linux® kernel to manage compute resources rather than performing the costly operation of hardware virtualization. This lightweight nature means each application gets its own container, preventing dependency conflicts.

More simply put, a container is a tarball, which is a type of zip file. With this in mind, containers are a way to bundle the code, configuration, and dependencies of an application. This bundling eliminates the problem of "It worked in my environment; why doesn't it work in yours?" Ideally, applications produce the same output regardless of environment, and containerization makes that ideal easier to reach. A containerized application will start, stop, make requests, and log the same way in different environments.

For businesses, containers help produce:

- Faster time to market as developers spend less time debugging environments and more time writing code.
- Lower server costs because more containers drive greater resource utilization.
- Increased deployment infrastructure options because containers can run anywhere.

For complex applications consisting of multiple components, containers vastly simplify updates. Placing each component in a container makes it simple to make changes without having to worry about ripple effects. This simplicity has led to containers becoming a great way to develop and deploy microservices. Microservices, applications with a single function, naturally benefit from containers because they provide a clean separation between components and services.

## CONTAINER MANAGEMENT AND ORCHESTRATION

Applications are increasingly built as discrete functional parts, each of which can be delivered in a container. That means for every application, there are more parts to manage. Furthermore, containers have shorter life spans than traditional, VM-only deployments. The complexity of managing applications with more objects and greater churn introduces new challenges: configuration, service discovery, load balancing, resource scaling, and discovering and fixing failures. Managing this complexity manually is impossible. Clusters commonly run more than 1,000 containers; updating these large clusters is infeasible without automation.

Kubernetes delivers production-grade container orchestration by automating container configuration, simplifying scaling, and managing resource allocation. Kubernetes can run anywhere. Whether you want your infrastructure to run on-site, on a public cloud, or on a hybrid configuration of both, Kubernetes delivers at massive scale.

## KUBERNETES AS CONTAINER ORCHESTRATION

Kubernetes is container orchestration. This means Kubernetes figures out where and how to run your containers. More explicitly, it provides three primary functions.

### Schedulers and scheduling

Schedulers intelligently compare the needs of a container with the health of your cluster, and they suggest where new containers might fit. Think of a scheduler as an assistant who manages a very large calendar. A controller consults the scheduler and then assigns the work and monitors the container.

Your household thermostat is a representative example of a scheduler. When you set it to 72 degrees, you expect it to maintain that temperature. The thermostat constantly checks the temperature and resolves it against a setting—72 degrees—on a regular basis. You set a temperature, and the thermostat seeks to maintain that temperature on your behalf.

In the same way, Kubernetes seeks to maintain cluster health on your behalf. Freed from minding the proverbial temperature, your developers can focus on work that provides value for your business.

### Service discovery and load balancing

In any system, service discovery can be a challenge. Kubernetes is no exception. The more services that make up your application, the more difficult your application is to track and manage. Thankfully, Kubernetes automatically manages service discovery. We might ask Kubernetes to run a service like a database or a RESTful API. Kubernetes takes notes about these services and can return a list if we ask about them later.

Kubernetes also checks the health of individual services. If Kubernetes detects a crash in your service, it will automatically attempt to restart it. In addition to these basic checks, Kubernetes allows you to add more subtle health checks. For example, perhaps your database has not crashed. But what if it is very slow? Kubernetes can track this and direct traffic to a backup if it detects slowness.

Kubernetes also incorporates load balancing. Modern services scale horizontally—by running duplicates of the service. The load balancer is the key piece that distributes and coordinates traffic across these duplicates. Kubernetes makes it easy to incorporate a custom load balancing solution like HAProxy or a cloud-provided load balancer from Amazon Web Services, Microsoft Azure, or Google Cloud Platform, as well as for OpenStack®.

### Resource management

Every computer has a limited amount of central processing unit (CPU) power and memory. An application that transcodes video might be CPU bound, and an application that parses text might be memory bound. This means the video application will run out of CPU power first, whereas the text parsing application will run out of memory first. Completely running out of either resource leads to system instability and crashes.

Proper resource management is the result of intelligent scheduling. Kubernetes schedules applications to appropriately use resources like CPU power and memory while staying cautious about over-utilization that leads to system instability.

## THE BENEFITS OF KUBERNETES

Containers are changing the way modern applications are built. Shifting from VMs to containers can seem daunting, but the benefits are real. Here is why now is the perfect time to begin your containerized future.

### Scalability

Kubernetes automatically scales your cluster based on your needs, saving you resources and money. Imagine you run an online event ticketing service where customers can purchase tickets 24 hours a day, 7 days a week with a load that is variable in time. You have a rough idea of what load (traffic) looks like when tickets go on sale during the day, and you know that evenings are rarely a strain on your system. However, there are a few popular events that cause exponential spikes in traffic. In this scenario, you would want your cluster to dynamically scale to meet the demand. With Kubernetes' Cluster Autoscaler and Horizontal Pod Autoscaler features, those adjustments happen automatically.

### Portability

Kubernetes can run anywhere. It runs on-site in your own datacenter, or in a public cloud. It also runs in a hybrid configuration of both public and private instances. With Kubernetes, the same commands can be used anywhere.

### Consistent deployments

Kubernetes deployments are consistent across infrastructure. Containers embody the concept of immutable infrastructure, and all the dependencies and setup instructions required to run an application are bundled with the container.

This is sometimes also called the "Pets vs. Cattle" argument. Pets have unique personalities, and we tend to give them names. For instance, you might recognize that a server behaves strangely, and start remembering its name. You spend time patching it, creating a customized environment to run your production. You might be scared to kill this server and replace it with another one. This fear often occurs because it is difficult to remember how the server was set up, or what changed after the initial setup.

Containers, on the other hand, are designed like cattle: mass produced and identical. They set themselves up the same way every time. Because they are immutable, additional configuration changes are prohibited. There is no fear in killing a container and starting another. The consistent experience means developers can spend less time debugging and more time delivering business value.

### Separated and automated operations and development

It is common for operations and development teams to be in contention. Operations teams value stability and are more conservative about change. Development teams value innovation and prize a high change velocity.

Kubernetes resolves this conflict. Thanks to the intelligence and automation of Kubernetes, operations teams can feel confident in the stability of the system. Conversely, containers save developers time, empowering rapid iteration cycles.

## BASIC KUBERNETES ARCHITECTURE

Kubernetes is a mature platform consisting of hundreds of components. It is complex and delivers an amazing set of features. Understanding its inner workings is important for its operators, but this guide presents a simplified version of the architecture to establish a good model for reasoning about Kubernetes.

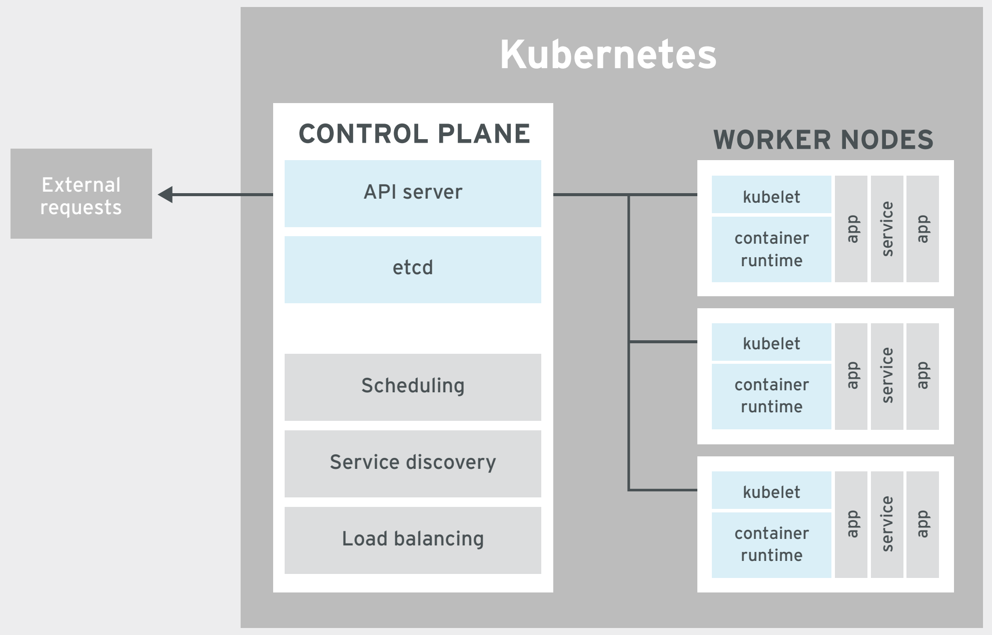

*Figure 1. A simplified Kubernetes control panel*

Kubernetes is software for managing containerized applications on a cluster of servers. These servers are either master or worker nodes. Together they run applications or services.

### Control plane

The control plane is roughly equivalent to the concept of a master node. It acts as the brain of any Kubernetes cluster. Scheduling, service discovery, load balancing, and resource management capabilities are all provided by the control plane. For this high-level architecture discussion, we will not get into the details of these functions. Rather, we will present them as parts of the control plane.

- **API server:** Kubernetes' API server is the point of contact for any application or service. Any internal or external request goes to the API server. The server determines if the request is valid and forwards the request if the requester has the right level of access.
- **etcd:** If the control plane is the brain, then etcd is where memories are stored. A Kubernetes server without etcd is like a brain that cannot make memories. As a fault-tolerant, inherently distributed key-value store, etcd is a critical component of Kubernetes. It acts as the ultimate source of truth for any cluster, storing cluster state, and configuration.

- **Worker nodes:** A worker node in Kubernetes runs an application or service. There are many worker nodes in a cluster, and adding new nodes is how you scale Kubernetes.
- **Kubelet:** A kubelet is a tiny application that lives on every worker node. The kubelet communicates with the control plane and then performs requested actions on the worker node. If the control plane is like a brain, a kubelet is like an arm. The control plane sends the command, and the kubelet executes the action.
- **Container runtime engine:** The container runtime, which complies with standards managed by Open Container Initiative (otherwise known as the OCI specification), runs containerized applications. It is the conduit between a portable container and the underlying Linux kernel.

### Missing from Kubernetes

While Kubernetes offers portability, scalability, and automated, policy-driven management to its users, it is an incomplete solution. It does not include all of the components needed to build, run, and scale containers in production, such as the operating system, continuous integration/continuous delivery (CI/CD) tooling, application services, or storage. A large amount of work also needs to be done to set roles, access control, multitenancy, and secure default settings. Kubernetes does provide pluggable interfaces for many of these components and services, offering flexibility and choice for users.

## RUNNING AND MANAGING KUBERNETES APPLICATIONS

### So, you want to run an application on Kubernetes?

Create a manifest, and share it with Kubernetes. Manifests are simple JSON or YAML files that declare the type of application to run and how many duplicates are required to run a healthy system. Sharing the manifest with Kubernetes will cause it to pass through the API server and ultimately get stored on etcd. The control plane will read your manifest from etcd and place the application on a node.

### Are you ready to scale?

Scaling an application is one of the best demonstrations of Kubernetes' simplicity. Change a value in your manifest, and Kubernetes will begin deploying more containers until it has satisfied the request.

### What happens when a pod crashes?

Anything deployed on Kubernetes has a desired state. For example, if you deploy a web server, you can set a desired state of "three." In other words, you expect three copies of the web server to be running. If one of those containers crashes, Kubernetes will automatically reconcile the difference. It knows that you want three containers but can see that you have only two, so it will add one more to satisfy the desired state.

## WHAT TO CONSIDER WHEN ADOPTING KUBERNETES

Before deploying Kubernetes, consider these questions:

### How do you develop and manage applications today?

Adopting a container-centric view of application development and management is necessary to remain competitive in today's markets. Containers or modern application development might be new to your organization. The first step is to engage all stakeholders—including application developers, business owners, and IT operations staff—to understand the goals and challenges unique to your team in adopting containers and Kubernetes. If your team is already building and delivering with containers, you have an application that is ready to run on Kubernetes.

### What is your hybrid cloud strategy?

Many enterprises are pursuing a hybrid cloud strategy combining an on-site datacenter with public or private cloud solutions. The number of organizations planning to use multiple public clouds is growing as well.

Kubernetes is open source and designed to run on any kind of infrastructure. As a bonus, Kubernetes federation provides a unified interface for balancing workloads between multiple cloud providers. As conditions such as price, service level, or personal preference change, Kubernetes gives you the power to react accordingly.

### How will you manage the life cycle of your Kubernetes solution?

While many organizations choose to install and manage Kubernetes on their own, others prefer to work with vendors that provide commercial support for Kubernetes deployments. In addition to providing production support for Kubernetes, vendors can:

- Offer managed upgrades and critical patches.
- Increase security for Kubernetes and for applications running on Kubernetes.
- Validate a range of third-party integrations, such as container registries, storage, networking, cloud services, and monitoring, logging, and alerting solutions.

Commercially supported Kubernetes solutions vary widely, so take careful note of your team's needs when deciding whether to seek additional support and with whom to work.

## SECURE, SIMPLIFY, AND SCALE KUBERNETES APPLICATIONS WITH RED HAT OPENSHIFT

Ready to start your journey with containers and Kubernetes? Red Hat is a leader and active builder of open source container technology, including Kubernetes, and creates essential tools for securing, simplifying, and automatically updating your container infrastructure.

Red Hat® OpenShift® is an enterprise Kubernetes application platform. With Red Hat OpenShift, teams gain:

- An enterprise-grade Kubernetes distribution with hundreds of security, defect, and performance fixes in each release.
- A single, integrated platform for operations and development teams. Red Hat validates popular storage and networking plug-ins for Kubernetes and includes built-in monitoring, logging, and analytics solutions for IT teams. Red Hat OpenShift offers developers their choice of languages, frameworks, middleware, and databases, along with build and deploy automation through CI/CD to supercharge productivity.
- Automated operations from over-the-air updates for Red Hat CoreOS, a Linux operating system, and Kubernetes, with automated updates for Red Hat-certified third-party application services (coming soon).

Red Hat is ready to support your organization's journey to containers and the cloud. Visit openshift.com/trial to try Kubernetes with Red Hat OpenShift.

### ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#f13237_0718