D2
IQ

Kubernetes FAQ

CONTENTS

What is Kubernetes?

Kubernetes (also known as K8s) is an open-source container orchestration platform used to deploy, manage, and scale containerized applications from anywhere. It provides a highly resilient infrastructure with built-in commands for rolling out changes to applications (automatic rollback), monitoring applications (health checks), restarting applications that fail (self healing), and scaling applications up and down, and more. In short, it takes care of a lot of the heavy lifting that goes into deploying and managing your applications, reducing the time, resources, and costs associated with your day-to-day operations.

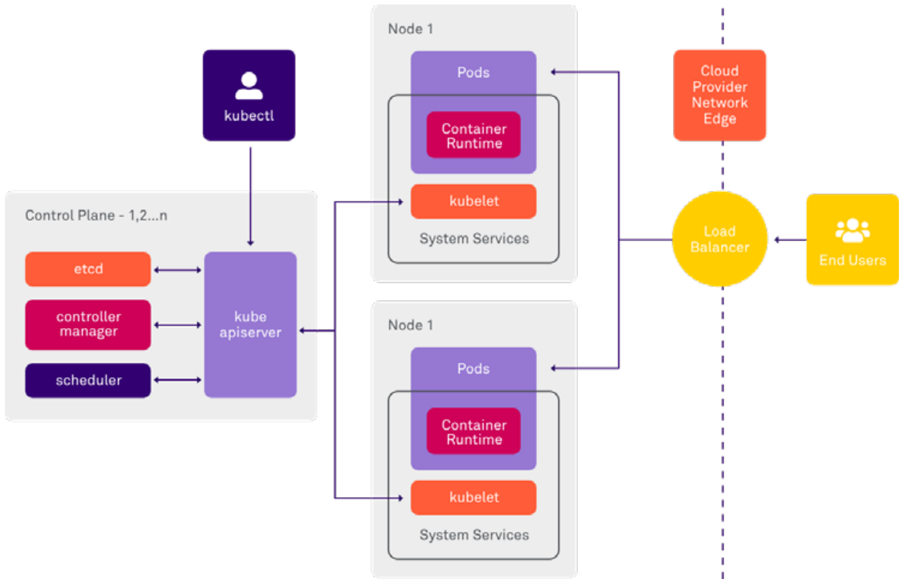
The name Kubernetes originates from the ancient Greek word, “helmsman” or “sailing master.” The seven spokes on the wheel of the Kubernetes logo refer to its original codename at Google, Project Seven of Nine.·

What does Kubernetes do?

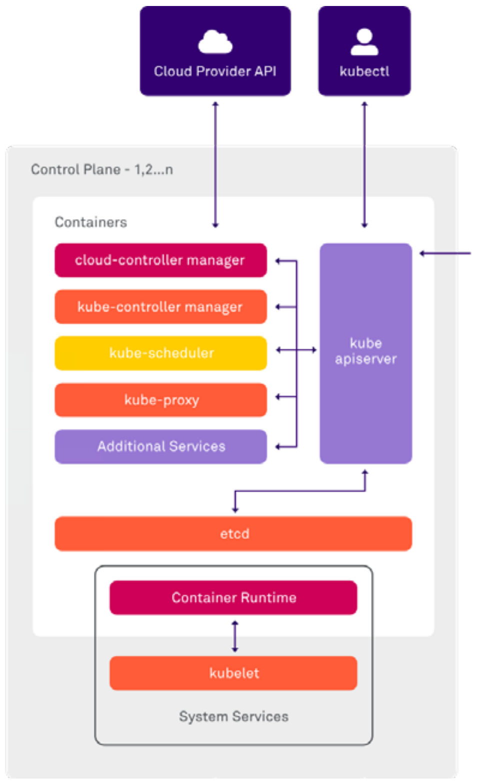
Containers are a way to package and ship an application’s code and dependencies from one computing environment to another. Using containers simplifies the development and deployment of an application. As the number or complexity of applications grows to span multiple containers that are deployed across multiple infrastructures, it becomes increasingly difficult to manage them.

To manage this complexity, Kubernetes is an open-source container orchestration platform that automates many of the manual operational tasks involved in deploying, updating, scaling, and monitoring multiple containerized applications. It does so by creating an abstraction layer on top of the underlying physical infrastructure, making it easier to run and operate applications more resiliently.

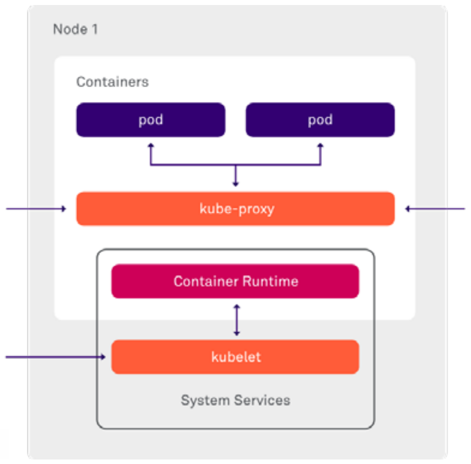
What does Kubernetes look like?



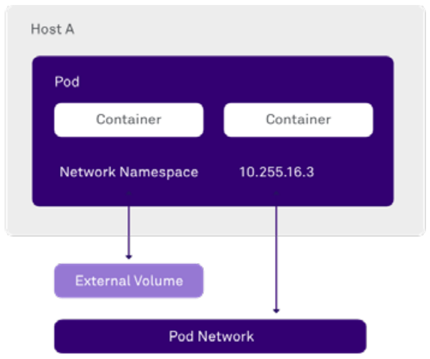
When you deploy Kubernetes, you get a Kubernetes cluster. The cluster is made up of two parts: the control plane (Kubernetes master) and the nodes in the cluster (also known as Kubelets, cluster nodes, worker nodes, or minions).



The Kubernetes control plane is responsible for maintaining the desired state of the cluster (e.g. scaling) and identifies and responds to cluster changes (e.g. auto-restarting a new pod when deployment is unsatisfied). The control plane components include a kube-api server, kube scheduler, etcd, kube-controller-manager, and cloud-controller-manager.



Nodes are machines that run containers and workloads. Each node is its own Linux environment, and could be either a physical or virtual machine. Each node also runs a container runtime engine (e.g. Containerd), as well as additional components for monitoring, logging, service discovery, and more.



Every node also runs pods, which package up a single application and consist of multiple containers that go together, along with rules that control how the containers run. Pods are capable of horizontal autoscaling, automatic rollouts, and canary deployments.

How is the Kubernetes-based container service managed?

Within an organization, a system administrator or member of the DevOps team assigns tasks through the Kubernetes control plane, which relays those instructions to the nodes.

Kubernetes services determine which nodes are available and best suited for the task. It then allocates resources and assigns the pods in that node to complete the task. Because Kubernetes takes care of various operational tasks and orchestration of the containers, the administrator does not have to manage each and every container or node. Instead, they can focus on configuring Kubernetes and the nodes, pods, and containers within them.

Kubernetes is portable by nature, meaning you can run Kubernetes on many different infrastructures, whether it's on-premise, in the cloud, at the edge, or any combination of them.

Why do I need Kubernetes?

Operating containerized applications can be very complex because they often involve multiple containers deployed across different infrastructures. It's not far-fetched to say you could end up with dozens, hundreds, or even thousands of containers over time. The efforts to deploy, manage, connect, update, and monitor these containers can delay value realization. Kubernetes includes built-in commands to help you build, schedule, and deploy multiple applications, scale them up and down to fit your changing needs, and manage their lifecycle.

What problems does Kubernetes solve?

- **Autoscaling** - Having to scale up and down workloads based on the demand of your application is no easy feat. And having to accurately estimate the compute and resource consumption of your application is just as challenging. Kubernetes relieves you from these concerns. With Kubernetes, you can not only automatically adjust the servers or machines required to run a service (horizontal auto scaling), but adjust the compute power required to run a service (vertical auto scaling). This means that you can scale down workloads when it's not needed, and scale up workloads when there is a greater demand.

- **Automated rollouts and rollbacks** - Kubernetes takes care of application deployment, making day-to-day operations easier. Rollout new versions or changes of your application or configuration, while monitoring the application's health. If something goes wrong, Kubernetes will automatically rollback the changes for you.

- **Health checks and self healing** - When a container goes down, you need to troubleshoot the problem without losing valuable time. Kubernetes continuously monitors the health of your containers, replaces containers that fail or stall, and doesn't make services available to users until they are successfully running.

- **Traffic routing and load balancing** - When you have an application comprising hundreds of containers across multiple deployment environments, they need to efficiently communicate with each other. Kubernetes sends requests to the appropriate containers and optimizes resources to respond to outages or periods of downtime.

- **Storage orchestration** - Kubernetes provides effective management of storage required by an application. Mount and add a storage system of your choice to run stateful applications, such as local storage, public cloud providers, and more.

- **Secret and configuration management** - Kubernetes provides built-in mechanisms to effectively store and manage sensitive information (such as passwords, OAuth tokens, and SSH keys) and configuration data across different environments without having to rebuild your container images or expose secrets in your stack configuration.

- **Service mesh** - When there is a large number of services being used across an organization, it can lead to significant inefficiencies and security concerns, if not effectively governed. Service mesh solves the challenge caused by container and service sprawl by controlling how different parts of an application share data with one another, enabling you to standardize communication between services.

- **Service discovery** - When you have a large number of services within an organization, having to manually configure the list of services is time-consuming and error prone. Service discovery solves this problem by automatically managing the list of service instances to be accessed. When you add a service or close a service, the registration service automatically updates the service list, dramatically reducing the configuration process.

- **Authorization and Role-Based Access Control (RBAC)** - When you have multiple users who need access to resources, it can be incredibly challenging to track all of the individual logins and permissions, especially when there are multiple accounts and access levels to manage. With Kubernetes authorization, users can access the resources they need and admins can manage access with consistent identities. And with RBAC, admins can flexibly configure user roles, responsibilities, and account privileges to create consistency across Kubernetes usage within the environment.

Who should use Kubernetes?

Kubernetes is a powerful container orchestration platform, but if you aren't facing the problems that Kubernetes aims to solve, it may not be the right choice or time for your team or organization to use.

For Kubernetes to be useful, consider the following requirements:

- **You have more than one service to manage**
- **You want to deploy, update, and manage software at scale**
- **You want to simplify and automate a number of complex management tasks**
- **You want to build cloud-native applications that can run anywhere**

Read on for more details in the next section.

When do I need to migrate to Kubernetes?

- **You are building a solution based on containers at scale** - If you need to deploy and scale multiple instances of containers, operate multiple services, and scale them up and down, Kubernetes provides an easy way to run containers at scale.

- **Your application consists of multiple services and need to be able to scale**- If your application consists of multiple, independent services and you need to host them at scale, managing it with Kubernetes is much easier than with a traditional infrastructure because you can automate a lot of the manual provisioning and operational tasks.

- **You are unable to meet customer demands due to slow development or deployment**- Because Kubernetes automates various manual provisioning and operational tasks, DevOps teams can focus on managing the deployed workloads instead of the underlying infrastructure, which can improve the speed of application delivery, as well as the quality, uptime, and scalability of those services.

- **You want to prepare your workloads to move to the cloud**- Building on top of containers and Kubernetes makes it easier to migrate applications and workloads to the cloud, compared to building on direct infrastructure or VMs and then moving to the cloud.

- **You have the need to deploy services across multiple environments**- Open-source Kubernetes enables you to deploy some services on-premises, on restricted networks, on multiple clouds, and at the edge, so you're closer to your customers.

Who owns Kubernetes?

Kubernetes was originally developed and designed at Google and is now maintained by the Cloud Native Computing Foundation (CNCF).

Who has the best Kubernetes solution?

The best Kubernetes solutions not only provide comprehensive container infrastructure lifecycle operations from the data center to the cloud to the edge, but also helps developers modernize applications with integrated service catalogs and microservices, service mesh, and serverless features.

That said, enterprises should look for a balanced blend of development and operations features that:

1. Simplify the development of cloud-native applications. Leading Kubernetes solutions include features that simplify application development, such as serverless support, continuous integration and delivery (CI/CD) integrations, dependency management, microservices frameworks, service mesh, and application lifecycle management features, like code quality checks and vulnerability scanning.

2. Enable distributed infrastructure operations across different infrastructures. Because enterprise workloads are increasingly distributed and hybrid, the top Kubernetes solutions offer model-driven configuration, monitoring, security, external support, and cluster lifecycle features for unified multi-cloud and multi-cluster operations, as well as an enhanced control plane with cost management and distributed tracing, dashboarding, and auditing features to improve observability for operations teams.

3. Expand enterprise value with rich app and service partner ecosystems. Many enterprises adopt cloud-native technologies to support a wide range of use cases and developer needs. The best Kubernetes solutions offer broad partner service catalogs, integrations with public cloud and edge services, multiple pricing tiers, and a range of deployment options to meet enterprises where they are in their Kubernetes journey.

How to compare Kubernetes solutions?

Key capabilities that you should be looking for when adopting a Kubernetes solution, include:

- **Avoiding vendor lock-in and proprietary solutions** - Make sure your Kubernetes platform enables your long-term independence and does not include proprietary updates that lock you into versions or customized builds of components (forks) that prevent you from integrating with other upstream projects from the CNCF ecosystem. The freedom to move between public cloud providers, hybrid, or on-premise and even air-gapped infrastructures may be a critical requirement for your projects in the future even though it may not be obvious at the beginning.
- **Total Cost of Ownership (TCO)** - Your Kubernetes provider should not mandate seemingly never-ending consulting services and multi-year service agreements. Choose a platform that prioritizes the needs of your business first, while always keeping TCO as a top priority.
- **Cloud native expertise** - You need the right technical expertise to ensure successful outcomes when adopting Kubernetes. You need a team with deep experience in standing up the most complicated deployments that can help guide the strategy, identify and design the platform for a broad set of requirements, and to sit side by side with expert technical guidance when you need it the most.
- **Certified Kubernetes training** - It is critical that your internal team has the necessary skill sets to achieve a successful Day 2 outcome. That requires a partner that is certified by the CNCF to train and up-level your organization to meet the transformational needs of your company as your projects and requirements evolve.
- **Broad workload coverage** - Your Kubernetes platform should allow you to run complex, mission critical business services, data-rich applications, streaming analytics, and machine learning use cases with the enterprise-grade requirements of scale, security, and resilience from the beginning.

For more information on how
we can help, please visit:

D2iQ.com