contents

[Introduction to Kubernetes Architecture](https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture?source=sl_frs_nav_playlist_video_clicked#introduction_to_kubernetes_architecture)

[Need for Containers](https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture?source=sl_frs_nav_playlist_video_clicked#need_for_containers)

[Docker Swarm vs. Kubernetes](https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture?source=sl_frs_nav_playlist_video_clicked#docker_swarm_vs_kubernetes)

[Hardware Components](https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture?source=sl_frs_nav_playlist_video_clicked#hardware_components)

[Software Components](https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture?source=sl_frs_nav_playlist_video_clicked#software_components)

Since 2014, Kubernetes has grown immensely in popularity. The adoption of this container deployment tool is still growing among IT professionals, partly because it is highly secure and easy to learn. But with every tool, knowing its architecture makes it easier to understand.

Let’s go over the fundamentals of Kubernetes architecture from what it is and why it is important, to a deep dive into what it’s made of.

[Kubernetes](https://www.simplilearn.com/tutorials/kubernetes-tutorial/what-is-kubernetes) is a flexible container management system developed by Google that’s used to manage containerized applications in multiple environments. Initially introduced as a project at Google (as a successor to Google Borg), Kubernetes was released in 2014 to manage applications running in the cloud. The Cloud Native Computing Foundation currently maintains Kubernetes.

Kubernetes is often chosen for the following reasons:

* Kubernetes has a better infrastructure than many of the DevOps tools
* Kubernetes breaks down containers into smaller modules to enable more granular management
* Kubernetes deploys software updates often and seamlessly
* Kubernetes lays the foundation for cloud-native apps

Let us now begin with the introduction to the Kubernetes architecture.

Introduction to Kubernetes Architecture

Kubernetes architecture comprises the following components.

Cluster

* A collection of servers that combines available resources
* Includes RAM, CPU, disk, and devices

Master

* A collection of components that make up the control panel of Kubernetes
* Consists of both scheduling and cluster events

Node

* A single host capable of running on a virtual machine
* Runs both Kube-proxy and Kubelet, which are a part of the cluster

After going through the introduction to Kubernetes architecture, let us next understand the need for the containers.

Need for Containers

With the ever-expanding presence of technology in our lives, downtime on the internet is becoming unacceptable. Hence, developers need to find solutions to maintain and update the infrastructure of the applications we depend on without interrupting other services people depend on.

The solution is container deployment. Containers work in isolated environments, making it easy for developers to build and deploy apps.

Docker Swarm vs. Kubernetes

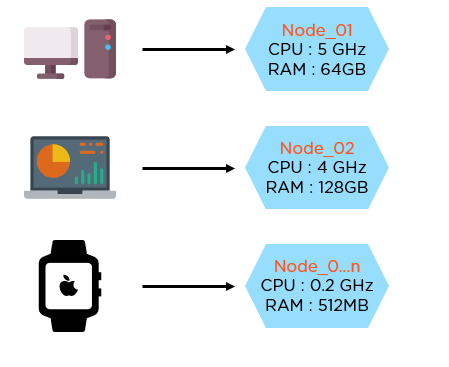
|  |  |  |
| --- | --- | --- |
| Category | Docker Swarm | Kubernetes |
| Scaling | No Auto Scaling | Auto Scaling |
| Load Balancing | Autoload Balancing | Manually configures load balancing |
| Installation | Easy and fast | Long and time-consuming |
| Scalability | Cluster strength is weak when compared to Kubernetes | Cluster strength is strong |
| Storage Volume Sharing | Shares storage volumes with any other container | Shares storage volumes between multiple containers inside the same pod |
| GUI | Not available | Available |

Hardware Components

Nodes

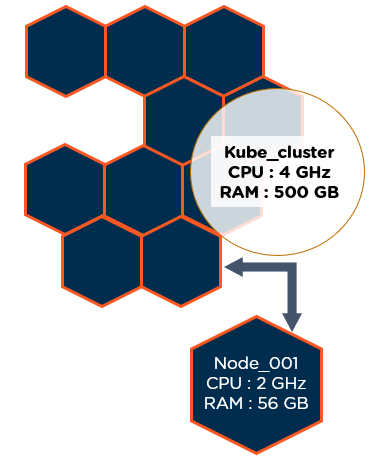
A node is a worker machine on Kubernetes. It is a Virtual Machine or a physical machine based on the cluster. The master maintains the code, and each node contains the necessary components required to run the Kubernetes cluster.

In Kubernetes, there are two types of nodes, Master Node and Slave Node.



Cluster

Kubernetes does not work with individual nodes; it works with the cluster as a whole. Kubernetes clusters make up the master and slave node and manage it as a whole. There can be more than one cluster in Kubernetes.

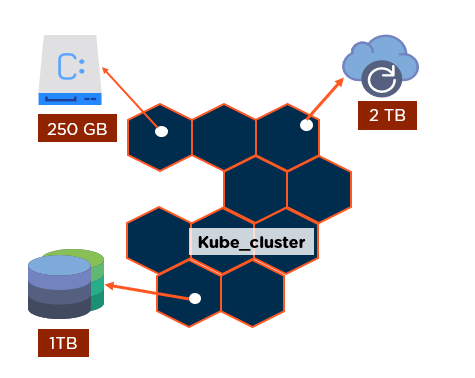


Persistent Volumes

Kubernetes persistent volumes are administrator provisioned volumes with the following characteristics.

* Allocated either dynamically or by an administrator
* Created with a particular file system
* It has a specific size.
* Has identifying characteristics such as volume IDs and a name.

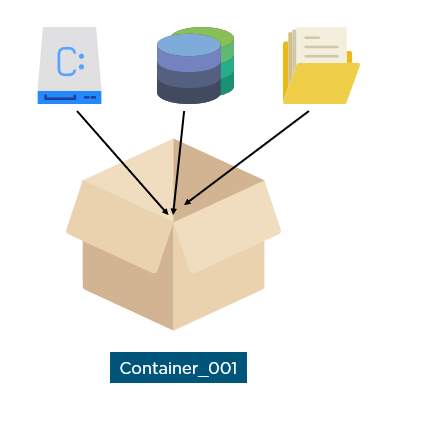
Kubernetes Persistent Volumes remain on a pod even after the pod is deleted. It’s used for the temporary storage of data.



Software Components

Containers

Containers are used everywhere, as they create self-contained environments where applications are executed. The programs are bundled up into single files (known as containers) and then shared on the internet. Multiple programs can be added to a single container; be sure to limit one process per container. Programs run on the Linux package as containers.

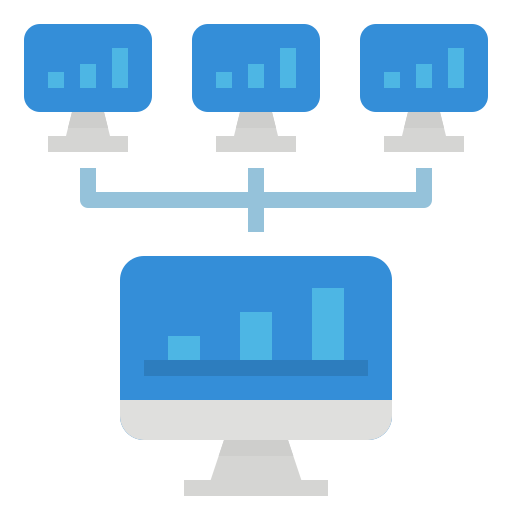


Pods

A Kubernetes pod is a group of containers deployed together on the same host. Pods operate one level higher than individual containers, and these groups of containers work together to operate for a single process. Pods provide two different types of shared resources: networking and storage, and are the units of replication in Kubernetes.

Deployment

A deployment is a set of identical pods. It runs multiple replicas of the application, and if in case an instance fails, deployment replaces those instances. Pods cannot be launched on a cluster directly; instead, they are managed by one more layer of abstraction. The manual management of pods is eradicated when deployment is used.



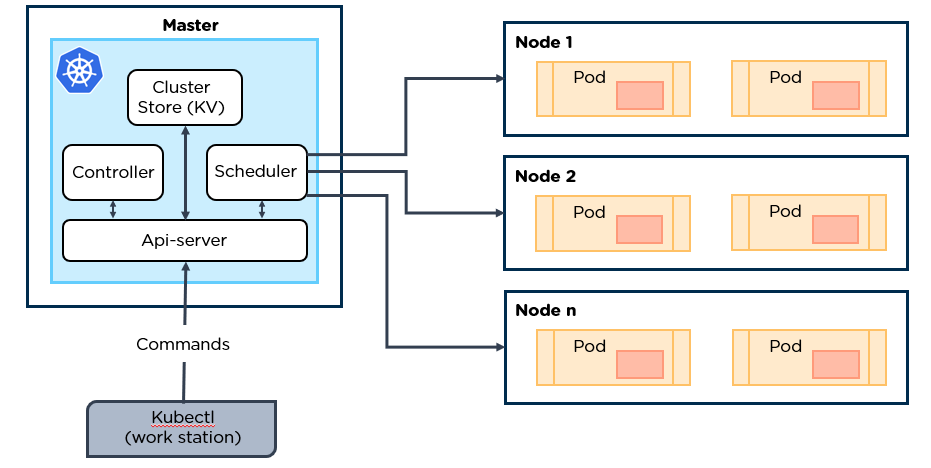
Ingress

Ingress is a collection of routing rules that decide how the external services access the services running inside a Kubernetes cluster. Ingress provides load balancing, SSL termination, and name-based virtual hosting.



Kubernetes Architecture

Kubernetes has two nodes—Master Node and Server Node.



Master

The master node is the most vital component of Kubernetes architecture. It is the entry point of all administrative tasks. There is always one node to check for fault tolerance.

The master node has various components, such as:

* ETCD
* Controller Manager
* Scheduler
* API Server
* Kubectl

1. ETCD

* This component stores the configuration details and essential values
* It communicates with all other components to receive the commands to perform an action.
* Manages network rules and post-forwarding activity

2. Controller Manager

* A daemon (server) that runs in a continuous loop and is responsible for gathering information and sending it to the API Server
* Works to get the shared set of clusters and change them to the desired state of the server
* The key controllers are the replication controllers, endpoint controller, namespace controllers, and service account controllers
* The controller manager runs controllers to administer nodes and endpoints

3. Scheduler

* The scheduler assigns the tasks to the slave nodes
* It is responsible for distributing the workload and stores resource usage information on every node
* Tracks how the working load is used on clusters and places the workload on available resources.

4. API Server

* Kubernetes uses the API server to perform all operations on the cluster
* It is a central management entity that receives all REST requests for modifications, serving as a frontend to the cluster
* Implements an interface, which enables different tools and libraries to communicate effectively

5. Kubectl

* Kubectl controls the Kubernetes cluster manager

        Syntax - kubectl [flags]

Slave

The slave node has the following components:

1. Pod

* A pod is one or more containers controlled as a single application
* It encapsulates application containers, storage resources, and is tagged by a unique network ID and other configurations that regulate the operation of containers

2. Docker

* One of the basic requirements of nodes is Docker
* It helps run the applications in an isolated, but lightweight operating environment. It runs the configured pods
* It is responsible for pulling down and running containers from Docker images

3. Kubelet

* Service responsible for conveying information to and from to the control plane service
* It gets the configuration of a pod from the API server and ensures that the containers are working efficiently
* The kubelet process is responsible for maintaining the work status and the node server

4. Kubernetes Proxy

* Acts as a load balancer and network proxy to perform service on a single worker node
* Manages pods on nodes, volumes, secrets, the creation of new containers, health check-ups, etc.
* A proxy service that runs on every node that makes services available to the external host

After going through the Kubernetes architecture, let us next understand its uses in the enterprise.

How is Kubernetes Being Used in the Enterprise?

Some companies merge Kubernetes with their existing systems for better performance. For example, let's take a look at the company Black Rock. Black Rock needed better dynamic access to their resources because managing complex Python installations on users' desktops were extremely difficult. Their existing systems worked, but they wanted to make it work better and scale seamlessly. The core components of Kubernetes were hooked into their existing systems, which gave the support team better, more granular control of clusters.