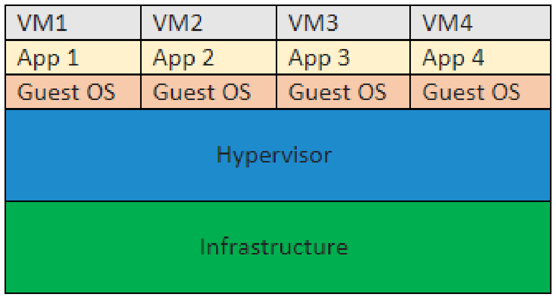
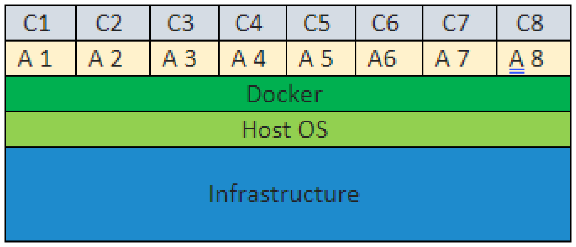
Cloud-Based system - Kubernetes, Persistence Comparison

We live in an era where a massive amount of data is generated every second, and several queries are handled each second. But how is this managed and stored??

Previously, we used to store all applications in a single system. Everything was working fine until our applications were small and easily managed. As the application size grew, the problem begins. The applications which need more resources hamper other applications' performance. To handle the resource requirements of each application, we need new systems. Maintaining several systems was costly for organizations.

To handle this problem, organizations started using Virtual Machines. Now on a single system, different VMs can be created, and each application can run on it as if it is running on a separate network. Each VM has its own set of OS, so they use more memory.





For more efficient memory management containers, they are similar to VMs, but they don't have their OS like VMs. Containers have their filesystem, CPU, memory, etc. same as VMs. It also provides numerous benefits like customized image, CI/CD (Continuous Integration and Continuous Deployment), separability of applications, efficient resource utilization, portability.

Now, manual monitoring, load balancing, version update, and auto-scaling are tedious tasks. All these problems are handled efficiently by Kubernetes. It runs smoothly on public, private, and hybrid clouds.

Kubernetes is an orchestration tool developed by Google to manage containerized applications in different environments like a physical machine, virtual machine, cloud, or hybrid environment. It is a highly available, scalable, and reliable system.

It means Kubernetes helps us manage hundreds and thousands of containers in different environments. It is highly available means have zero downtime. Scalable means we can increase or decrease resources as per need. Portable means container, once created, can run on different systems. Also, several replicas can be made as per requirement.

Suppose an organization has 100 microservices running on the separate containers on the Docker engine. In that case, if anyone fails, the Kubernetes will create another Pod containing the same image and destroy an unhealthy one. It will keep all replicas of the Container up and running.

Kubernetes handles these functionalities by the master and worker nodes. There can be at least one master node and several worker nodes. Each worker node has a Kubelet process running on it. Kubelet is a Kubernetes process that makes it possible for the cluster to communicate with each other and executes some tasks on nodes like the running application process. Each worker node has a container of several applications deployed on it. Depending on the workload requirement, we have several Pod running in the node. The actual application runs on the worker node.

There is various Cloud-Based system in the market like Kubernetes we have to choose a tool based on multiple parameters. The end selection depends on the needs and requirements of the organization. Some parameters are as below:

- Container Network Interface Networking: A promising tool allows proper network connectivity between services in a cluster. So that developers can spend time wisely on code instead of being stuck in fixing connectivity issues, they can work on other essential parts of the application.
- Simplicity: The selected tool should be simple to implement.
- Active Development: The tool should have a development team that provides regular updates to the user because container orchestration is evolving each day.
- Cloud Vendor: We should not rely on any single cloud provider. There should be options to migrate to different vendors as per requirements.

Kubernetes vs Apache Mesos

Kubernetes and Mesos follow different approaches to the same problem. Kubernetes acts as a container orchestrator, whereas Apache Mesos works like a cloud operating system. Therefore, there are various fundamental differences between the two, which are highlighted in the table below:

Points of Difference	Kubernetes	Apache Mesos
Application Definition	Kubernetes is an orchestration tool developed by Google to manage containerized applications in different environments like a	The Mesos' Application Group is an n-ary tree, with branches and applications as leaves. It partitions applications into multiple

	physical machine, virtual machine, cloud, or hybrid environment. It is a highly available, scalable, and reliable system.	manageable sets, where components are deployed in order of dependency.
Availability	The task is performed by the worker node with the applications deployed in the container in Pods.	Applications are divided among Slave nodes.
Load Balancing	Service exposes Pods to external URL and acts as load balancer.	Mesos-DNS helps applications to expose URL to browser and act as load balancer.
Storage	In Kubernetes, data inside Pod is not persistent, so to provide permanent storage on NFS, AWS, EBS, etc. Kubernetes Persistent Volume provides global storage available to the whole cluster.	Persistent volumes storage is provided by a Marathon Container, which is local to the node where they are created, but it has to be run on the said node. The experimental Flocker integration supports persistent volumes that are available to the whole cluster.
Networking Model	In Kubernetes' any Pod can communicate with other Pods inside the cluster by Services having internal IP addresses. Whenever a Pod is created a dynamic IP is assigned to the Service of Pod.	There is no dynamic IP creation for each container, it is done by integrating with Calico.
Purpose of Use	It provides a quick, easy, and light way for newcomers to understand orchestration and implement it in an organization. It is portable, a	It is good for large systems as it is designed for maximum redundancy. For existing workloads like Hadoop or Kafka, Mesos provides a

	high degree of versatility, and is supported by Microsoft and IBM.	framework that allows the user to interleave those workloads. It is relatively complex to use but a more stable platform.
Vendors and Developers	Several companies and developers use Kubernetes and are supported by platforms like Red Hat, OpenShift and Microsoft Azure.	Organizations like Twitter, Apple, and Yelp support Mesos as its core focus is one Big Data and analytics, so the learning curve is steep and quite tricky.

Kubernetes vs OpenStack

Kubernetes and OpenStack have been viewed as competitors, but in reality, both are open-source technologies that can be combined and complementary. OpenStack lets businesses in running their Infrastructure-as-a-Service (IaaS) and is a powerful software application. They both give solutions to relatively alike problems but do so on different layers of the stack. If we combine Kubernetes and OpenStack, it can provide noticeably enhanced scalability and automation.

It consists of sixty-plus components called 'services.' Core services are for networking, access management, compute, identity, storage management. OpenStack comprises a series of commands known as scripts bundled together into packages called projects. The projects are responsible for creating cloud environments.

Both Kubernetes and OpenStack and provide solutions for networking and cloud computing but with different approaches. Some of the differences are explained in the table below.

Points of Difference	Kubernetes	OpenStack
Classification	Described as a Container tool	Described as an Open Source Cloud
User Base	It has a large GitHub community.	Not supported by much-organized community

Companies that Use them	Google, Slack, Shopify, Digital Ocean, 9GAG, Asana, etc.	PayPal, HubSpot, Wikipedia, Hazeorid, Survey Monkey, etc.
Main Functions	Efficient management of container.	A versatile and flexible tool for handling Public and Private Clouds
Tools that can be Integrated	Microsoft Azure, Docker, Google Compute Engine, Kong, etc.	Spinnaker, Distelli, Fastly, Stack Storm, Morpheus etc.

Kubernetes vs Docker Swarm

Docker swarm is Docker's container's orchestration tool. It uses the standard Docker API and networking, making it easy to integrate into the Docker containers' environment. Dock Swarm designed considering these points:

- Provide user simple and powerful "just works" user experience
- Flexible zero single-point-of-failure design
- Guarded by default with automatically created certificates
- Backward compatibility among existing components

Both orchestration tools offer the same functionalities, but there are fundamental differences in between how the two operate. Some are listed below:

Points of Difference	Kubernetes	Docker swarm
plication definition	Applications and database are deployed on Pods by deployment and Stateful Set	Applications are used only as microservices in a swarm cluster. Containers are created by YAML files. Docker Compose helps to install application.
alability	Kubernetes cluster provide scalability as per need of the	Docker Swarm can deploy containers faster than Kubernetes, which provides

	application by creating and deleting the replicas of Pods.	faster reaction times for scaling.
ntainer Setup	By utilizing its API, YAML, and client definitions, Kubernetes differs from other standard docker equivalents. YAML commands and definitions must be rewritten when shifting platforms.	The Docker Swarm API offers much of Docker's familiar functionality, supporting most of the tools that run with Docker. However, Swarm cannot be used if the Docker API lacks a distinct operation.
Networking	It has a flat network model that allows the pods to communicate with each other. Service define how pods should interact with one another.	When a node joins a swarm cluster, an overlay network for each host's services is created in the docker swarm. It also creates a host-only docker bridge network. It gives users a choice while encrypting the container data traffic to create its overlay network.
Availability	Kubernetes offers high availability as it distributes all the pods among the worker nodes. Unhealthy pods are detected by load balancing services and are replaced by healthy Pods.	Docker also offers high availability since all the services are cloned in Swarm nodes. The Swarm manager Nodes manage the whole cluster and the worker's node resources.
Load Balancing	Pods are exposed via Ingress and Services in the Kubernetes cluster. Also, load balancing is done by both.	Swarm Mode comes with a DNS element that can distribute incoming requests to a service name. Thus, services are assigned automatically or function on ports that are pre-specified by the user.

Logging and Monitoring	Built-in tools are used for managing logging and monitoring.	No tool required for logging and monitoring in Docker swarm.

Kubernetes will be the best containerization platform to use if the application we are developing is complex and employs hundreds of thousands of containers in the process. It has auto-scaling capabilities and high availability policies. Kubernetes is suitable for those users who are comfortable with customizing their options and need extensive functionalities.