**The Rise of Containerization and Docker**

Containers and container platforms provide a lot more advantages over traditional virtualization. Isolation is done on the kernel level without the need for a guest operating system, so containers are much more efficient, fast, and lightweight. Allowing for applications to become encapsulated in self-contained environments comes with a slew of advantages, such as quicker deployments, scalability, and closer parity between development environments.

Docker is currently the most popular container platform. Although the idea of isolating environments dates quite far back, and there has been other container software in the past, Docker appeared on the market at the right time, and was open source from the beginning, which likely led to its current market domination.

Docker features the Docker Engine, which is a runtime and allows you to build and run containers, and includes Docker Hub, a service for storing and sharing images.

**Docker** is a platform and tool for building, distributing, and running **Docker** containers. ...**Kubernetes** is a container orchestration system for **Docker** containers that is more extensive than **Docker Swarm** and is meant to coordinate clusters of nodes at scale in production in an efficient manner.

**Kubernetes and Docker** Swarm are probably two most commonly used tools to deploy containers inside a cluster. Both are created as helper tools that can be used to manage a cluster of containers and treat all servers as a single unit.

**How Does Kubernetes Relate to Docker?**

Kubernetes and Docker are both comprehensive de-facto solutions to intelligently manage containerized applications and provide powerful capabilities, and from this some confusion has emerged. “Kubernetes” is now sometimes used as a shorthand for an entire container environment based on Kubernetes. In reality, they are not directly comparable, have different roots, and solve for different things.

**Docker** is a platform and tool for building, distributing, and running Docker containers. It offers its own native clustering tool that can be used to orchestrate and schedule containers on machine clusters.

**Kubernetes** is a container orchestration system for Docker containers that is more extensive than Docker Swarm and is meant to coordinate clusters of nodes at scale in production in an efficient manner. It works around the concept of pods, which are scheduling units (and can contain one or more containers) in the Kubernetes ecosystem, and they are distributed among nodes to provide high availability. One can easily run a Docker build on a Kubernetes cluster, but Kubernetes itself is not a complete solution and is meant to include custom plugins.

Kubernetes and Docker are both fundamentally different technologies but they work very well together, and both facilitate the management and deployment of containers in a distributed architecture.

**The Need for Orchestration Systems**

While Docker provided an open standard for packaging and distributing containerized applications, there arose a new problem. How would all of these containers be coordinated and scheduled? How do all the different containers in your application communicate with each other? How can container instances be scaled?

Solutions for orchestrating containers soon emerged. Kubernetes, Mesos, and Docker Swarm are some of the more popular options for providing an abstraction to make a cluster of machines behave like one big machine, which is vital in a large-scale environment.

When most people talk about “Kubernetes vs. Docker,” what they really mean is “Kubernetes vs. Docker Swarm.” The latter is Docker’s own native clustering solution for Docker containers, which has the advantage of being tightly integrated into the ecosystem of Docker, and uses its own API. Like most schedulers, Docker Swarm provides a way to administer a large number of containers spread across clusters of servers. Its filtering and scheduling system enables the selection of optimal nodes in a cluster to deploy containers.

Kubernetes is the container orchestrator that was developed at Google which has been donated to the CNCF and is now open source. It has the advantage of leveraging Google’s years of expertise in container management. It is a comprehensive system for automating deployment, scheduling and scaling of containerized applications, and supports many containerization tools such as Docker.

For now, Kubernetes is the market leader and the standardized means of orchestrating containers and deploying distributed applications. Kubernetes can be run on a public cloud service or on-premises, is highly modular, open source, and has a vibrant community. Companies of all sizes are investing into it, and many cloud providers offer Kubernetes as a service. Sumo Logic provides support for all orchestration technologies, including Kubernetes-powered applications.

Ref: <https://www.sumologic.com/blog/devops/kubernetes-vs-docker/>

Can you use Kubernetes without Docker?

Kubernetes is a tool that can help you deploy Docker containers. It's only one of several such tools. You don't have to use it to use Docker—but you can't use Kubernetes without also using software from Docker.

What is Docker Swarm vs Kubernetes?

Docker Swarm or simply Swarm is an open-source container orchestration platform and is the native clustering engine for and by Docker. Any software, services, or tools that run with Docker containers run equally well in Swarm. Also, Swarm utilizes the same command line from Docker.

How does Kubernetes work with Docker?

It offers its own native clustering tool that can be used to orchestrate and schedule containers on machine clusters. **Kubernetes** is a container orchestration system for **Docker** containers that is more extensive than **Docker** Swarm and is meant to coordinate clusters of nodes at scale in production in an efficient manner.

Does Kubernetes use Docker?

You can **run docker** container on a single server and **use** the container as if an virtual operation system to **run** applications. **Kubernetes** is an orchestration tool for**Docker** container. You can **use** it to **run** and manage multiple **docker** containers.

Why is Docker needed?

From a certain point of view a container is not so much different from a virtual machine. But, instead of creating a full operating system, a **Docker** Container has just the minimum set of operating system software **needed** for the application to run and rely on the host Linux Kernel itself

How does Docker swarm work?

**Docker works** to maintain that desired state. For instance, if a worker node becomes unavailable, **Docker** schedules that node's tasks on other nodes. A task is a running container which is part of a **swarm** service and managed by a **swarm**manager, as opposed to a standalone container.

Is PaaS a docker?

**Docker** or generally container technologies are an enabler for **PaaS** services. ... I'd go as far as saying **PaaS** is a layer of automation and orchestration across many IaaS layers, the most important of which (the compute) has mostly transitioned from a virtual-machine to a container.

Is Kubernetes a PaaS?

**Kubernetes** is not a traditional, all-inclusive **PaaS** (Platform as a Service) system. Since **Kubernetes** operates at the container level rather than at the hardware level, it provides some generally applicable features common to **PaaS** offerings, such as deployment, scaling, load balancing, logging, and monitoring.

**What is chef puppet?**

Puppet is a powerful enterprise-grade configuration management tool. Both Chef and Puppet help development and operations teams manage applications and infrastructure.

What is the difference between Chef and Puppet?

The definitions are usually written in Ruby. Puppet is a user application, so Chef is also a user application but can also become part of the application. In Puppet, the code can be executed on both the machines that are master and slave, while in case of Chef the code can only be executed on node machines.

**Does Docker replace chef?**

Docker does NOT replace configuration management tools like Chef, Puppet and Ansible. Those are still necessary for preparing the host machine which Docker containers will run on. ... Instead of configuring the host for Ruby/Python/etc. you would move that configuration to your Dockerfile.

Chef, Puppet, Ansible, and SaltStack are industry-wide used DevOps tools, included in DevOps Certification. They are all “configuration management” tools, which means they are designed to deploy, configure and manage servers. But do you know which among Chef vs Puppet vs Ansible vs Saltstack is the best tool for IT automation?

Below tools shows the pros and cons of each of these tools, after which you will be able to decide the most appropriate tool for your organization’s need and environment. These tools are very simple to use yet powerful enough to automate complex multi-tier IT application environments.

<https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/>

<https://azure.microsoft.com/en-in/blog/azure-virtual-machines-using-chef-puppet-and-docker-for-managing-linux-vms/>

<https://info.microsoft.com/rs/157-GQE-382/images/Infrastructure-as-Code-guide-EN-v6_299129.pdf>