**Introduction to Container Orchestration**

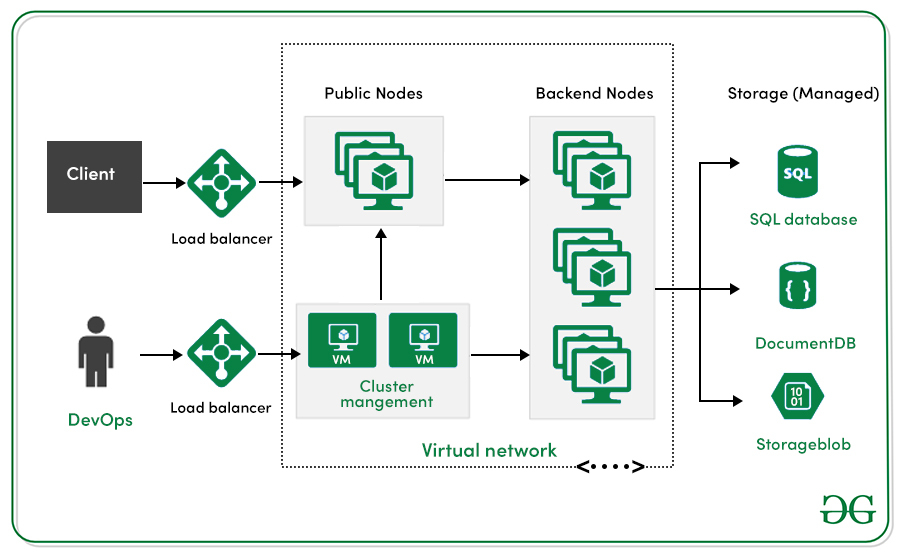
In this article, we will look into Container Orchestration in Kubernetes. But first, let’s explore the trends that gave rise to containers, the need for container orchestration, and how that it has created the space for Kubernetes to rise to dominance and growth.

The growth of technology into every aspect of our lives and days has created immense demand for software, the companies, and organizations that sell and deliver products based on software or just use software in their businesses. This pressure has spurred innovation and standardization. And the needs and stakes are so high it has fundamentally changed how softwares are developed and deployed into production. But this evolution isn’t without precedent. Could you imagine the world’s economy relying on methods like using ships for the trade of goods?! There would be no way you’d get your new iPhone in time. Now we take it for granted that shipping containers efficiently move between different modes of transportation different ship owners and shipping companies and have standardized ubiquitous and available anywhere in the world almost, always no matter what it’s in. Modern software is becoming quite similar. As software systems become more complicated this complication has driven software to be divided into smaller pieces such as **microservices.**Whether they’re called microservices or not these smaller pieces of software each need to be packaged, built, deployed, and accessed by other pieces of software around them to function as a total system.

These small pieces of software can be deployed into containers that allow them to run on the same virtual or real machine but making it appear to the software that is the only process running. Here’s where we depart from the shipping container metaphor a bit. While it should be containers sole job is to keep things contained within itself typically, useful software depends on other software around it to do its job. Here is where the need for**container orchestration** is born. There are definite benefits in keeping logically distinct software separate from each other, you can develop, deploy, scale and, maintain each small piece without too much fuss between the other parts of the system when it’s not all in the same module. However, at some point different parts of your application likely to communicate with each other to do perform a task. For example, an API may need to communicate with business logic and it needs in turn to access information or database.

**Containers**themselves keep logically distinct pieces of software separate so they can be built, deployed, maintained, managed and, scaled on their own without unduly affecting other parts of the system.

**Container orchestration** on the other hand defines how these containers interact as a system, the needs between each other and how they come together to your performant, manageable, reliable and, scalable system. It is currently safe to say that docker is the dominant leader in both technology and an adoption in containers while others exist in the space their dominance is so great when you mention containers it is almost synonymous with docker. Other players exist in niche applications and some movement around standardization is occurring although is nascent at the docker file standard is essentially today’s standard and docker repositories define the dominant way of hosting versions of a container between developers and between systems that need to use them, such as orchestration systems like kubernetes.



Since container orchestration is a much newer space only growing since the adoption of containers the technologies are newer and still evolving. However, kubernetes is quite advanced and with Amazon Web Services finally joining its competitors and embracing kubernetes it has the backing of nearly every major player and cloud computing through the **Cloud Native Computing Foundation** also known as the CNCF. This is the organization that now governs kubernetes. It has essentially come the de facto standard especially now that it’s Mearing alternative competing technology docker swarm, Dockers solution to container orchestration has now announced support for kubernetes style configuration and processing it is clear that kubernetes is now the de facto standard if not the acknowledged standard in the industry and container orchestration.

Containerization changed how software was developed and packaged, container orchestration is changing how containers are deployed into bigger useful systems.