Docker vs. Kubernetes vs. Kubernetes Platform. What's the difference and do you need it all?

Containers vs. container orchestrators. It's not an either or

Digital transformation, powered by cloud-native technology, is taking off and container adoption, particularly **Docker containers**—the container runtime—is rapidly increasing. Containers, much more lightweight than VMs and containing all dependencies, are much more efficient than virtual machines. Kubernetes, on the other hand, is a container orchestration framework. It allows users to schedule, scale, and manage containers. So it isn't really a Docker vs.. Kubernetes; it's more of a Docker + Kubernetes.

As many cloud-native technologies, Docker containers were created for the cloud, but aren't necessarily cloud-bound. Portability, one of its revolutionary benefits, enables organizations to move applications and all its dependencies across clouds and on-prem. Containers are much less resource intensive as they have one kernel running in isolated environments instead of having many virtual machines. The reason is that each virtual machine required a separate kernel and that would waste ram, CPU and hard drive storage.

Linux Kernel Gets Containers

Around 2008 LXC came to be. LXC or Linux containers is kernel-level virtualization, allowing you to have multiple server applications in isolated environments. LXC is similar to BSD Jails. These isolated environments are secure and powerful because you don't need to allocate the resources as you would a virtualization solution such as KVM or XEN. Before this, there was only a solution like OpenVZ. This had severe drawbacks and was mostly used by low-end hosting companies who would oversell capacity on a single server. Naturally, this led to problems for customers and a bad reputation for OpenVZ. Early on these LXC were very hard to use. It was not like a simple LAMP setup. Then Docker was born. Docker containers have all the benefits of LXC but are much more user-friendly. This was when containers started to take off. As early adopters started using them in production, orchestration engines such as Docker Swarm, Mesos, and ultimately Kubernetes, the de facto market leader, started to appear.

The Role of Containers in The Cloud-Native Ecosystem

Cloud-native technologies treat servers and the underlying infrastructure as commodities. They are merely the roads you drive on. You can drive any type of car on the road without worrying about the underlying road. Virtualized servers (VPS) are more rigid when deployed. If you wanted to move your

virtualized instance to another physical server, then you would have to make a physical copy and then create a virtualized server on the new physical server. This meant it was hard for hosts and clients to be nimble in how they worked with infrastructure. There is also the fact that once your instance was destroyed, the data would be gone forever unless you had a backup solution.

Container technology enables flexibility and efficient resource utilization. A container is an isolated environment providing greater flexibility with your servers. They recover from failure easier, and when a container or the underlying server crashes, you just move it to another server. When deploying containers, often used with microservices which splits an application in different services each of which has its own container, you are likely dealing with much more containers than VMs. Managing them manually becomes an endeavor and automation become a necessity. This is where Kubernetes comes in.

Kubernetes. Scheduling and scaling Docker containers

Kubernetes is the Docker runtime. It's a framework to schedule and scale Docker containers. With Kubernetes, users can start, stop, and manage containers. Docker and Kubernetes complement one another; it's not an either or. If you want to compare apples to apples, you'd have to compare Docker Swarm vs. Kubernetes.

If you are using Docker containers in development only, then there may not be a need for Kubernetes. Dev environments are much less complex, and you're likely not dealing with a vast number of containers. However, once you start to scale, there is no way around a container orchestrator.

Kubernetes, open sourced by Google in XX, is a framework that helps automate deployment, scaling, and management for containerized applications. Its scalability allows companies to manage an almost infinite number of containers. You can even orchestrate storage with Kubernetes which is vital for organizations that have a massive amount of data that is spread across multiple servers. If you are moving to containers, then Kubernetes is likely your best options.

Kubernetes vs. Kubernetes Platform

You may now be asking yourself if Kubernetes takes care of all the scheduling and scaling; why would you need a Kubernetes platform. Well, Kubernetes is a framework, not a fully fletched production ready solution. There is also no support—ask your ops team if they feel comfortable running applications

without support. There are 1000 ways to configure Kubernetes and to do it properly you need deep in-house Kubernetes expertise. Additionally, Kubernetes is upgraded every quarter, and keeping up to date is very time-consuming. Some companies may have the in-house expertise and resources and want to go this route. If you don't, you'll likely have to choose a vendor supported platform.

How does Kublr fit in?

When you are working at scale, it can be complicated to manage and orchestrate your clusters. There is a lot of flexibility that Kubernetes provides and that leads to the complexity that can make even the most seasoned IT Pro lose their hair. Kublr is a tool that makes this process a lot simpler. This tool was built to help organizations create a secure and highly available cluster without the drawback of being complicated and hard to operate. It features an innovative graphical user interface that allows you to set up the various master and worker nodes and it can even auto-scale if your environment permits it.

Kublr helps maintain security for your critical Kubernetes Nodes. The best part is that this tool is enterprise grade. This means that if you are a large organization who want to simplify developer operations, it can be quite helpful to you.

These are just some of the tools that you need to work with containers. Kubernetes is on the way to being the most used orchestration tool for containers, and you can extend this with the power of Kublr.