**What is Docker, and why should I use it?**





A Real Life Scenario: Docker makes it really easy and simple for different people working on a project to run their application in the same environment without any dependencies or OS issue involved as Docker provides its own OS. Consider this before and after scenario:

**Before Docker:** a developer sends code to a tester but it doesn’t run on the tester’s system due to various dependency issues, however it works fine on the developer’s end.

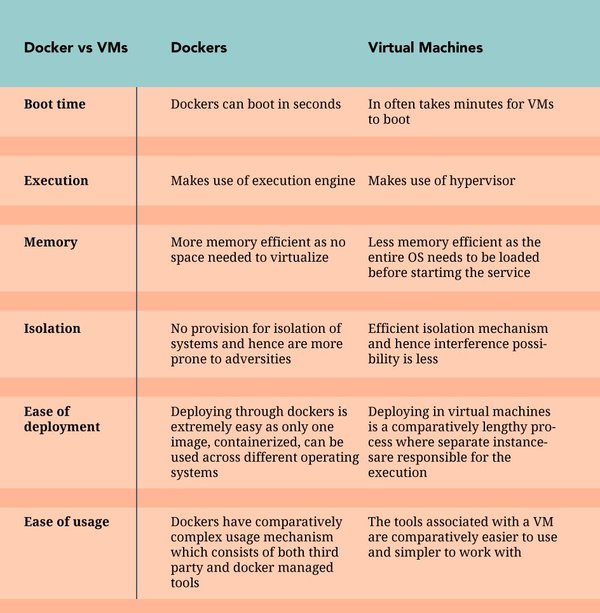
**After Docker:** As the tester and developer now have the same system running on Docker container, they both are able to run the application in the Docker environment without having to face differences in dependencies issue as before.

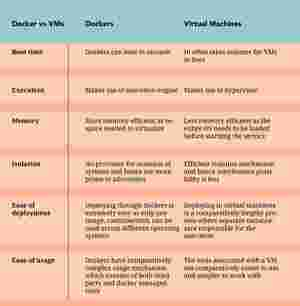
**A little more about Docker and containers:**

Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container. It’s a set of platform-as-a-service products designed to solve the many challenges created by the growing DevOps trend. Docker makes it easier to create, deploy, and run applications with the use of containers.

Containers are what make Docker so appealing to the modern developer. They create an abstraction at the app layer that packages your application and dependencies with everything it needs to run including: the operating system, application code, runtime, system tools, system libraries, etc.

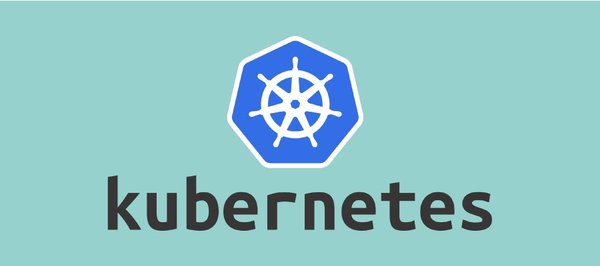
Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.





Credit via https://www.educba.com/docker-vs-vms/

**What is Kubernetes, and why should I use it?**





Kubernetes is a powerful container management tool that automates the deployment and management of containers. Kubernetes (k8’s) is the next big wave in cloud computing.

When it comes to running containers in production, you can end up with dozens, even thousands of containers over time. These containers need to be deployed, managed, and connected and updated; if you were to do this manually, you’d need an entire team dedicated to this.

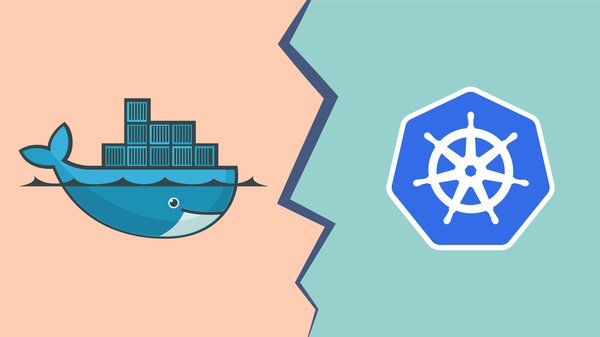
It’s not enough to run containers; you need to be able to:

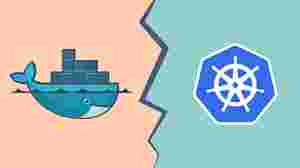
* Integrate and orchestrate these modular parts
* Scale up and scale down based on the demand
* Make them fault tolerant
* Provide communication across a cluster

You might ask: aren’t containers supposed to do all that? The answer is that containers are only a low-level piece of the puzzle. The real benefits are obtained with tools that sit on top of containers — like Kubernetes. These tools are today known as container schedulers.

*For more on this, see our guide:*[*Why and when it’s a good idea to use Kubernetes*](https://www.educative.io/blog/why-and-when-you-should-use-kubernetes)*.*

**The difference between Docker and Kubernetes**





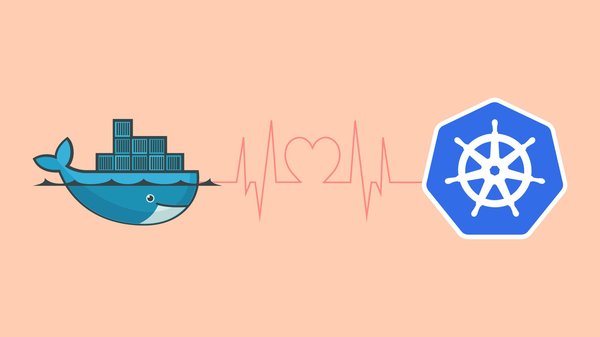
Let’s preface this by saying that they are two different technologies that are designed to work together. They are not competing tools, so there shouldn’t be any confusion around, “should I use one over the other?” or “which one is better?”. They both have their own roles in DevOps and are frequently used together.

With that said, here’s the difference:

**Docker** is used to isolate your application into containers. It is used to pack and ship your application.

**Kubernetes** on the other hand is a container scheduler. It is used to deploy and scale your application.

**Docker and Kubernetes: better together**





The two technologies are designed to work together, and when they do it’s a DevOps dream. As mentioned earlier in this post, it’s not enough to merely run containers in production, they need to be regulated and Kubernetes offers some great features that make working with containers even easier. Kubernetes offers things like auto-scaling, health checks, and load balancing which are crucial to managing the container lifecycle.

It’s important to get your application containerized, but don’t forget the next step in the process; how will you run your containers at scale in production? If you answered Kubernetes, you’re correct.