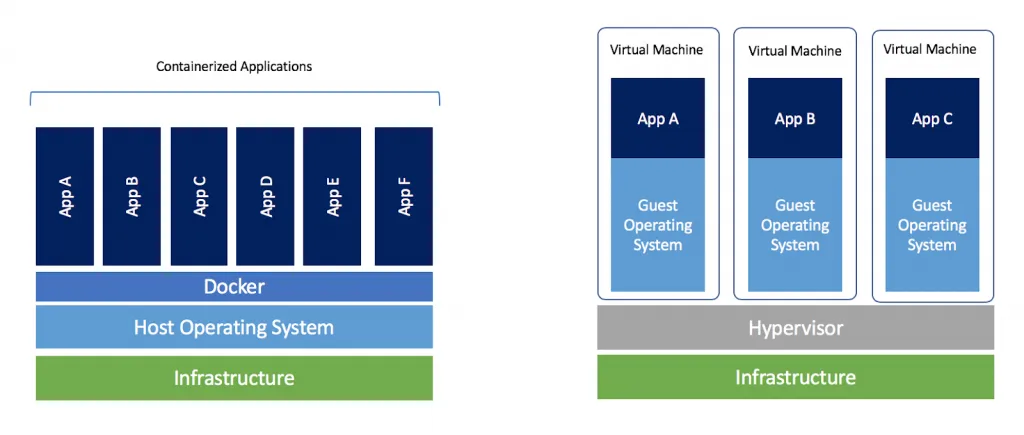
**Container and Container Orchestration**

You can think of containers as lightweight, scale-able, and isolated VMs (not an actual VM) in which you run your applications. A container is a standalone package that encapsulates application code and all of its dependencies so the application can run in any environment. Containers give you portability, isolation, and packaging.

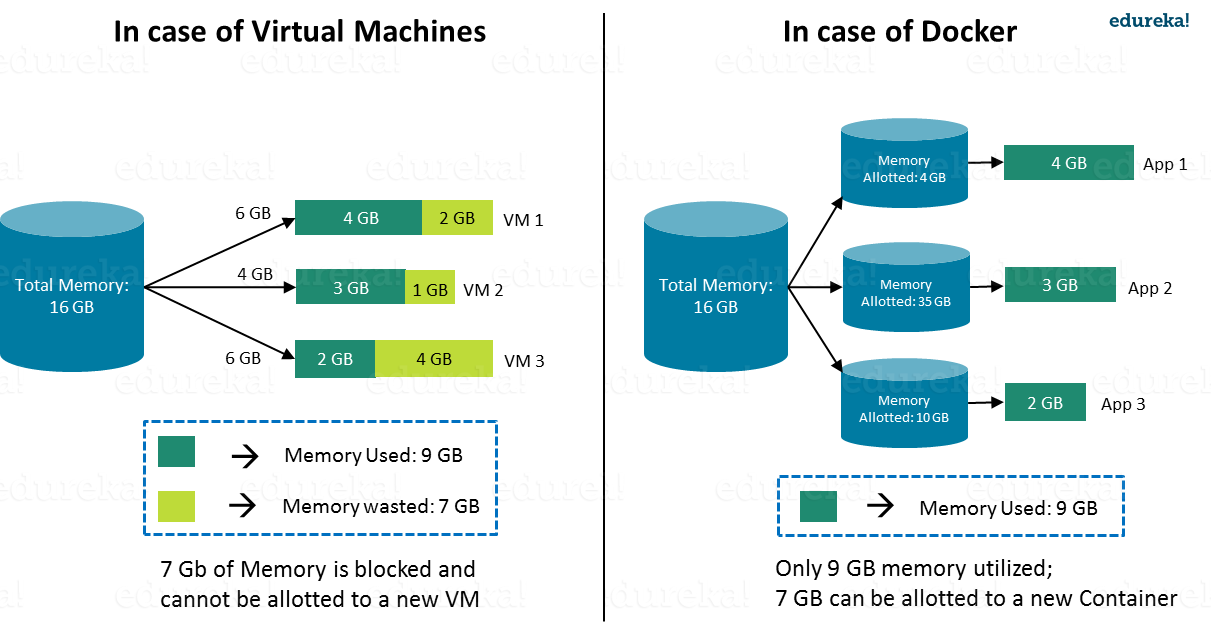
To understand the concept of containers, suppose one works on a project and it works fine in one’s machine and when the same project is run on any other machine it doesn’t work, it can be because of any dependency or because of a different version of a library. So, containers can help, just wrap everything inside an image ( containers are running instances of images ) and other users can directly run this image.

To illustrate the concept of container further, suppose a developer is working on a web application that requires PHP 7.2, so he install PHP 7.2 on his local machine, later he updates the app to use PHP 7.2 to 8.0 then all the developers who are working on the same web app also need to update PHP, and if there are more such libraries or dependencies which are updated then all the developers need to update everything. But while using containers, only one developer has to update everything and create a new image and other developers can use that image. Since the new image contains all the updated libraries and dependencies inside it, therefore, other developers need not update everything, they can just use the new image.



(image link: <https://i1.wp.com/www.docker.com/blog/wp-content/uploads/Blog.-Are-containers-..VM-Image-1-1024x435.png?ssl=1>)

In the above right figure, each application is running its own copy of the OS (Guest OS). Whereas containers share host OS and therefore are much more efficient, lightweight, and also take less time to start. There are varieties of container runtime available like docker, rocket, containerd, etc. Docker is the most popular container runtime. Maybe now you can reread para one to understand it better.



(Image link: <https://qph.fs.quoracdn.net/main-qimg-6b6ea3dadd8b83eefc00d69b2694ad72>)

**Container Orchestration:**

There are multiple container orchestrations available like Kubernetes, docker swarm, etc. Kubernetes is the most popular container orchestrator. Kubernetes was first developed by a team at Google, and later donated to the Cloud Native Computing Foundation (CNCF) and became open source. It automates container operations. It eliminates many of the manual processes involved in deploying and scaling containerized applications. You can cluster together groups of hosts running containers, and Kubernetes helps you easily and efficiently manage those clusters.

To understand the concept of Kubernetes and what we can do with Kubernetes. Suppose one owns a shopping website.

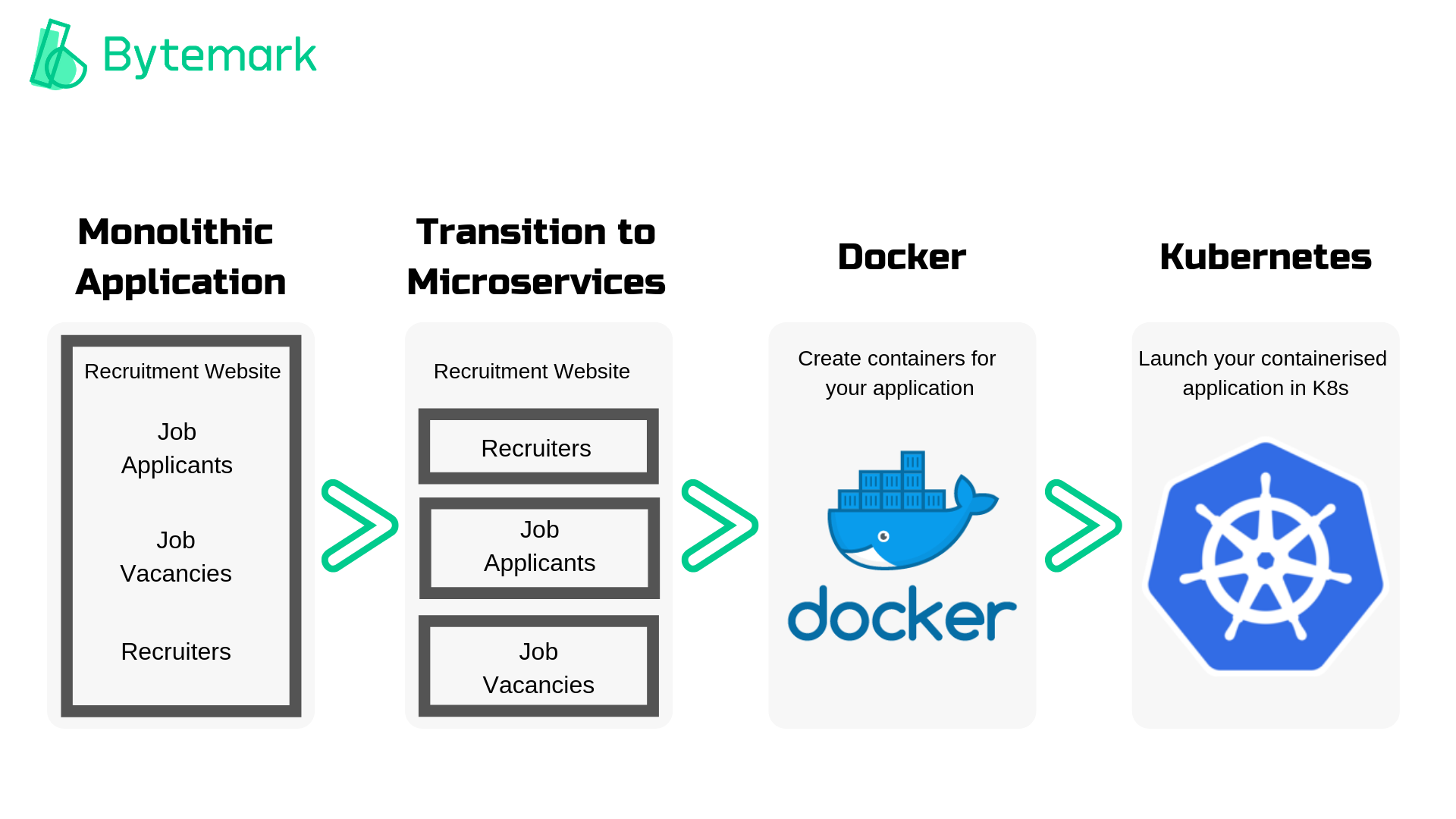
Case 1: Suppose he hosts a server only in the Delhi region to handle all requests from India. But suppose due to any reason (like hardware failure) the Delhi server gets down, then his site goes down. Therefore he should host his website in multiple regions in India like Delhi, Mumbai, and Kolkata. This is called **high availability**. Any problem like hardware failures does not bring his application down because he has multiple instances of his application.

Case 2: Suppose there is too much load in the Delhi region then how he can handle that? By sending some traffic to other regions. So he needs a **load balancing** way to balance the traffic in all the regions.

Case 3: Suppose there is a sale and server traffic increases in all the regions too much. So he needs to scale up the server to handle all the traffic. And when the sale is over then he is wasting his scaled resources so he needs to scale down his server. Therefore he needs something so that he can easily **scale up and down** according to his needs.

Case 4: Suppose during the sale he has to update his website and that needs to be done without any downtime. So he wants **zero downtime deployment**.

Case 5: He doesn't want any of his services (in a microservice architecture, see image below) as the payment service goes down. So he needs a **Health check and Self-healing** of all his services. If anything goes down, it automatically heals, which means automatic restart of the service.



(Image link: <https://docs.bytemark.co.uk/wp-content/uploads/2019/02/Monolithic-Application.png>)

Using Kubernetes he can solve all the above problems. There are lots of other features available that anyone can explore. As it is open source and used by lots of different organizations there are lots of new features added in every version. Many companies are getting a head start on Kubernetes and joining the revolution. By using containers and container orchestration, organizations can build, deploy, and horizontally scalable lightweight applications across multiple types of server hosts, cloud environments, and other infrastructure more efficiently.