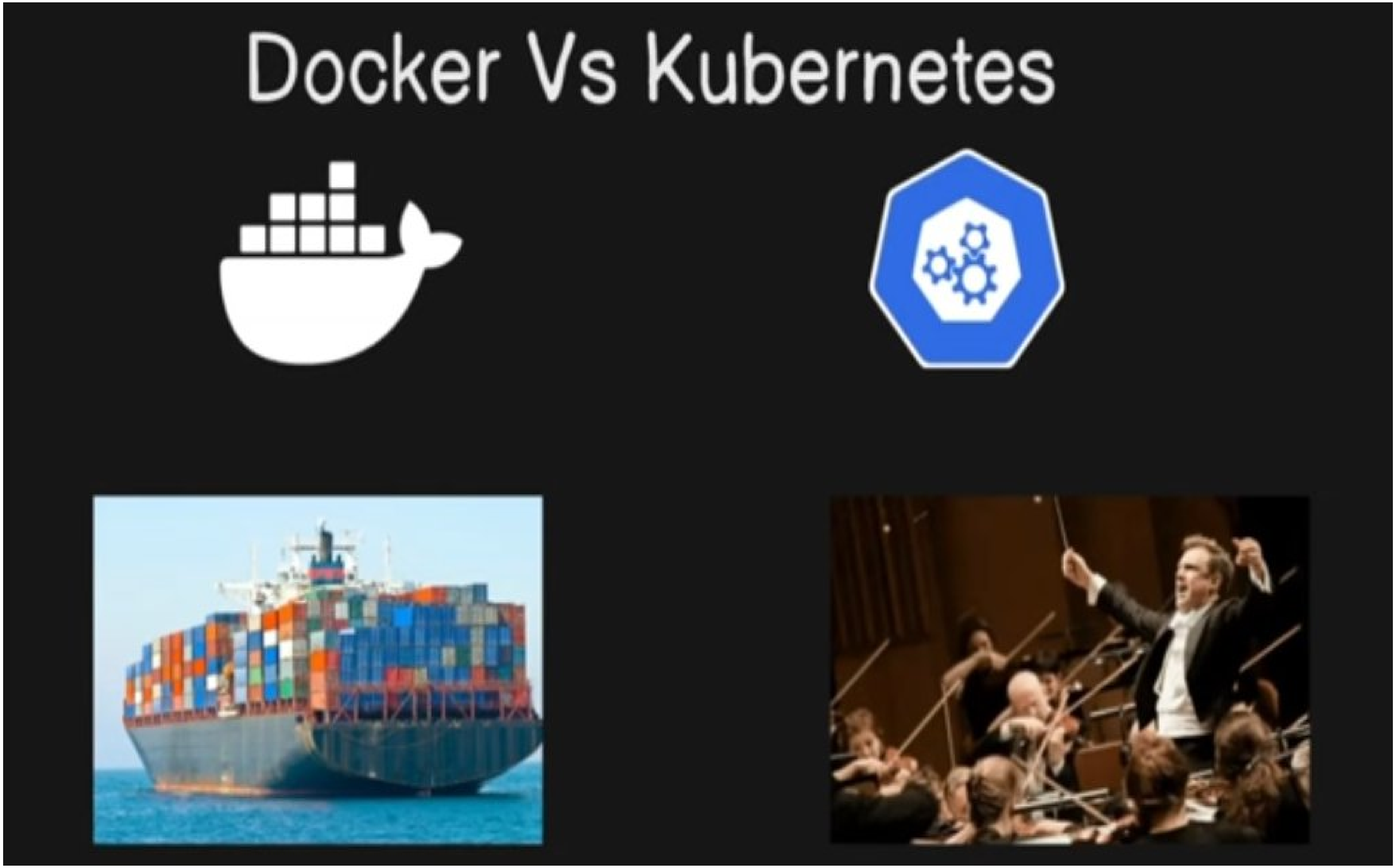
Kubernetes



Docker Vs Kubernetes

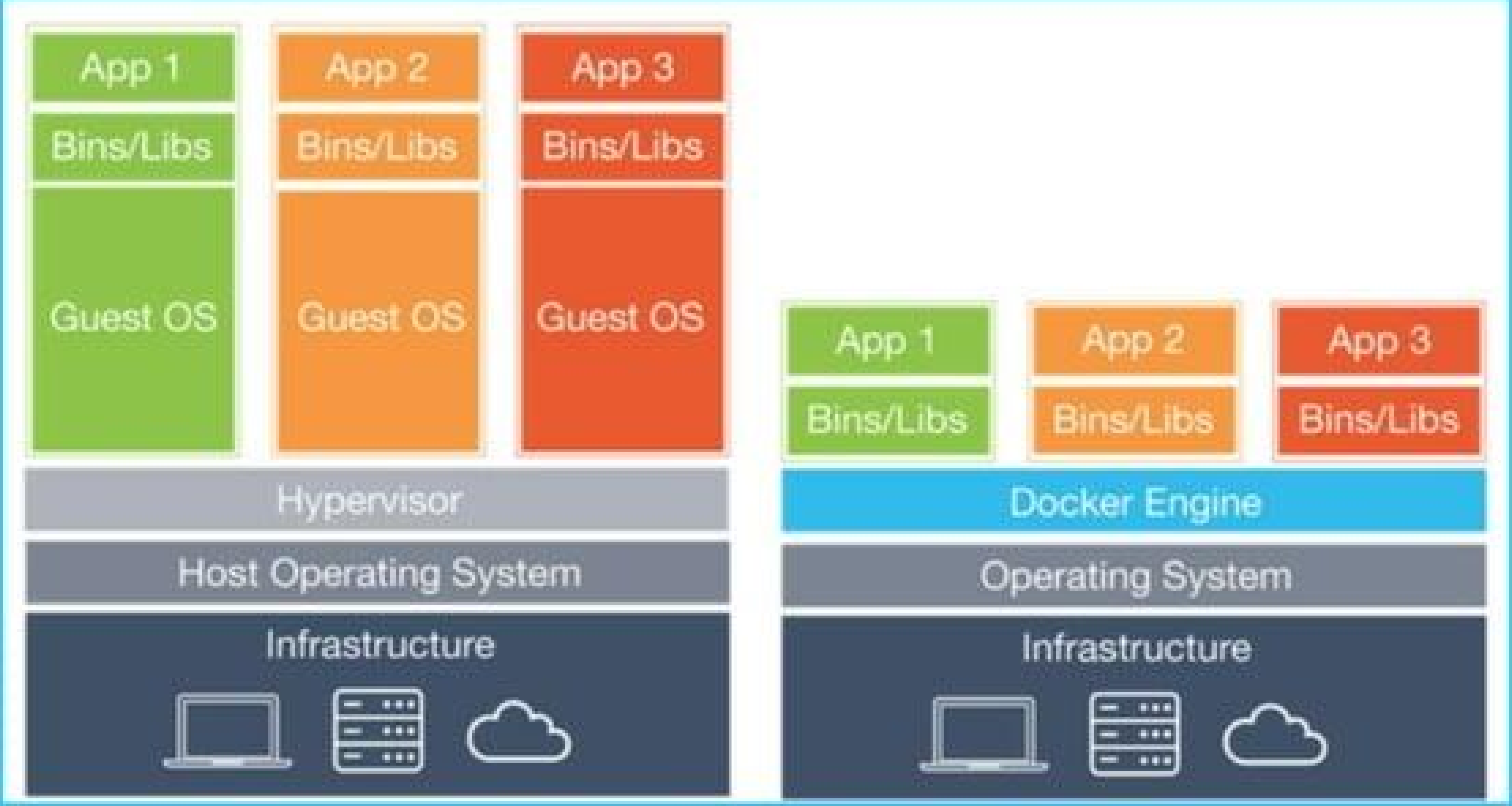
Virtualisation Technologies - VMs vs. Containers

Both are essentially structures for storing and running applications.

- Virtual Machines (VMs) - Isolated at hardware level, higher resource usage.

- Containers - Slightly less isolated, leaner and faster.
  - Docker has ~95% market share in containerisation technology.

Virtualisation Technologies - VMs vs. Containers



Bytemark

You can have 1000's of containers on a server

- Because containers are so lean, you can have a huge number of them on a single server. They allow large monolithic applications to be split into many microservices.
- Each microservice has its own container & controls one feature of the application.
  - **Benefit:** This is an efficient use of resources & can save on hosting costs compared to VMs.
  - **Problem:** How do you manage, rollout, rollback, maintain and repair such a large number of containers?
- The solution: **Kubernetes**

Bytemark

Kubernetes: Definition

- Kubernetes (K8S) is an open source software tool for managing containerised workloads.

- It operates at the container (*not* hardware) level to automate the deployment, scaling and management of applications.

- K8s works alongside a containerisation tool, like Docker. So if containers are the 'ingredients' of an application, then K8S would be the 'chef'.

- As well as managing individual containers, K8s can also manage **clusters**.
  - A cluster is a series of servers connected to run containers.
  - K8s can scale up to 5,000 servers and 150,000 **pods** in a single cluster.
  - A pod is a group of containers that share resources, a network and can communicate with one another.

Bytemark

How Does Kubernetes Work?

Kubernetes has a master-slave architecture:

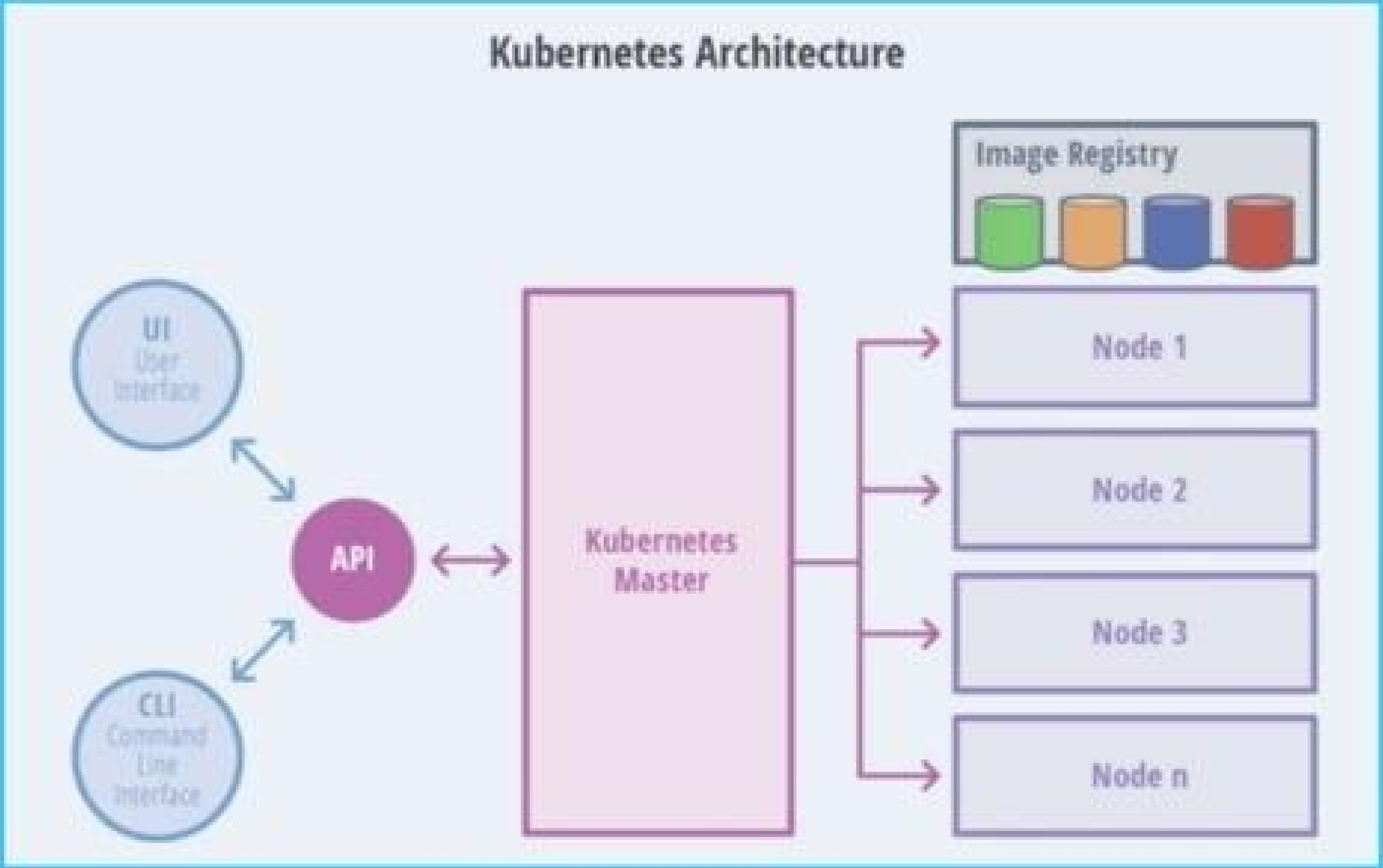
1. **Worker Nodes** (slave) - This is where containers are deployed. These nodes contain:
   a. Multiple pods
   b. Docker engine
   c. Any add-ons e.g. DNS
   d. Kubelet - this is an important component as it carries out the instructions from the master node.

2. **Master Node** (master) - This controls the deployment. This node contains:
   a. API server - this receives inputs from the User Interface (UI) or Command Line Interface (CLI).
   b. Controller - uses information from the API to drive the application from its current state towards the desired state provided.
   c. Scheduler
   d. Etcd - handles configuration management, service discovery etc.

Bytemark

Kubernetes Architecture

To make the last slide easier to visualise, here is a diagram of the architecture:



Nodes 1-n (in purple) are the worker nodes.

Bytemark

Kubernetes Architecture

- Remembering the specific components and their roles isn't important here.
- The architecture has just been outlined to give some background on how Kubernetes is able to orchestrate containers at scale.

- Whilst initially the idea of Kubernetes sounds quite simple, seeing the architecture shows that is is actually fairly complex to work with.
- This is *why* Bytemark want to offer a managed Kubernetes service - so our customers can get the benefits of Kubernetes without worrying about managing and maintaining everything themselves.

Benefits of Kubernetes

- **Self-healing**
  - Clusters can auto-restore from errors by rolling back to the last working version of software. This allows teams to ship quickly without the risk of breaking anything.
- **High Availability**
  - Clusters can be recreated on a working node to avoid downtime during server failure.
- **Simplifys Maintenance**
  - If a server needs to be rebooted, or the host OS needs updating, containers can be moved to another node whilst maintenance is carried out.
- **Automatic Scaling**
  - Uses information from user requests and CPU usage to increase or decrease the number of nodes running to match demand.
- **Efficient**
  - Automatically spins up any new containers on under-utilized nodes.

Bytemark

Things Kubernetes does *not* do...

There are sometimes misconceptions about what Kubernetes can do. Kubernetes does not…

- Provide any comprehensive machine configuration.
- Provide a configuration language/system
- Dictate logging, monitoring or alerting solution
- Build your application
- Provide middleware, data-processing frameworks, databases, caches etc. **BUT** these components can run on Kubernetes