**Kubernetes**

Is a portable, extensible open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. Kubernetes provides a container-centric management environment. Google open-sourced Kubernetes project in June 2014. Kubernetes can be thought of as:

1. A container platform
2. A Micro-services platform
3. A portable cloud platform and lot more

**Container concepts**

At the core of container technology are control groups (cgroups) and namespaces. Additionally, Docker uses union filesystems for added benefits to the container development process.

Cgroups work by allowing the host to share and also limit the resources each process or container can consume. This allows several containers to share CPU and memory while staying within the predefined constraints.

Namespaces offer another form of isolation for process interaction within operating systems. Container processes are limited to see only what is in the same namespace.

Union filesystems are also a key advantage of using Docker containers. Containers run from an image. Much like an image in the VM or cloud world, it represents state at a particular point of time. Container images snapshot the filesystem but tend to be much smaller than a VM.

Deployment of containers are based on operating-system-level virtualization rather than hardware virtualization. These containers are isolated from each other and from the host. They have their own filesystems, they cannot see each other’s processes and their computational resource usage can be bounded. They are portable across clouds and OS distributions.

**Points to remember**

* Continuous integration is a development practice that requires developers to integrate code to a shared repository several times a day. The result is that updates and bug fixes happen much faster and the overall quality improves
* Using Docker ensures that OS, packages and application versions are the same across development, staging and production environments
* Micro-service is an architectural style – an approach to develop a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms. These services are built around business capabilities and independently deployable by fully automated deployment machinery.
* Kubernetes (k8s) is a management and orchestration tool for containerized application deployment.
* K8s operates at container level rather than at hardware level. If an application can run in a container, it should run great on k8s
* K8s control script is known as ‘kubectl’. Most administrative interactions are done via this script
* The ideas of actual state and desired state is the key of how k8s manages the cluster and its workloads. All the pieces of k8s are constantly working to monitor the current actual state and synchronize it with the desired state defined by the administrators via the API server or kubectl script. There will be times when these states do not match up, but the system is always working to reconcile the two
* By default, Kubernetes uses the ‘gce’ provider for Google cloud. We can override this default by setting the KUBERNETES\_PROVIDER environment variable. Some permissible values are listed below:
  + gce (Google Compute Engine – public cloud)
  + gke (Google Conatiner Engine – public cloud)
  + aws (Amazon Web Services – public cloud)
  + azure (Microsoft Azure – public cloud)

**Master and slave**

* Master is the brain of k8s cluster. Here we have a core API server that maintaines RESTful web services for querying and defining our desired cluster and workload state.
* Master includes a scheduler that works with the API server
* Public cloud (Anyone can use it)