# Status as on 6/1/18 (Gurpreet and Sidharth)

Researched about the following topics:

* Container
* Docker
* Kubernetes

We went through the initial part of documentation of Kubernetes and Docker and set it up on our personal laptops. Installed and learnt the usage of basic Docker commands. We used minikube to set up Kubernetes on our local machine. We set up the containers using Docker and then ran a small nodejs app on it.

# Future Plan:

We’ll set up Kubernetes on the Linux server provided in the Innovation Lab.

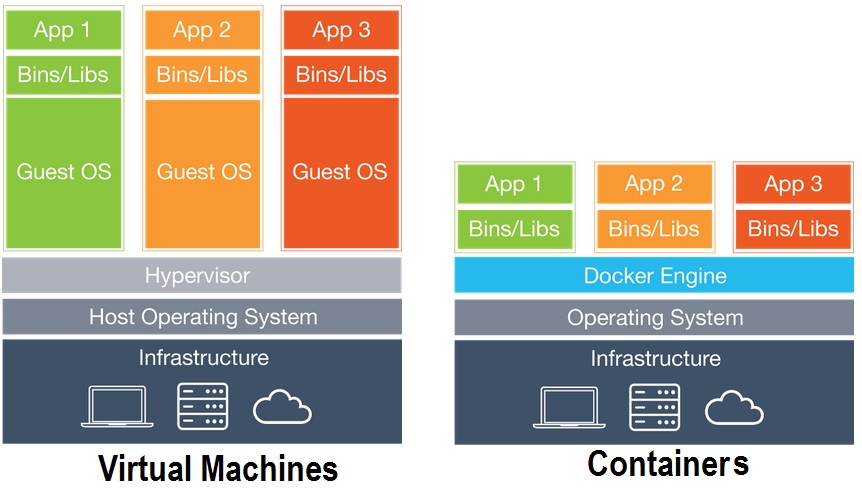
-------------------------------------------------------------------------------------------------------------

# Researched Content (in brief):

**What is a container?**

* A container consists of an entire **runtime environment**: an **application**, plus all its **dependencies**, **libraries** and other **binaries**, and **configuration files** needed to run it, **bundled into one package.**
* By containerizing the application platform and its dependencies, **differences in OS** distributions and underlying infrastructure are **abstracted away**.
* Difference from Virtual Machines:

1. They run a single operating system hence are **lightweight**.
2. A container is small in size, in MBs, whereas a virtual machine with its own entire operating system is in GBs. Because of this, a **single server can host far more containers than virtual machines.**
3. Virtual machines may take several minutes to boot up their operating systems and begin running the applications they host, while containerized applications can be **started almost instantly**.



**What is Docker?**

* Open source tool that **packages, provisions and runs containers** independent of the OS.
* Allows a high degree of portability so that users can register and share containers over various hosts in private and public environments.
* Traditionally VM’s were used. Docker made container technology popular and easy to use.

**What is Kubernetes?**

* An open source platform that automates [Linux container](https://www.redhat.com/en/topics/containers/whats-a-linux-container) operations.
* Real **production apps span multiple containers**. Those containers must be deployed across multiple server hosts. Kubernetes gives you the **orchestration and management capabilities required to deploy containers**, at scale, for these workloads.
* Essentially it provides an abstraction to make a cluster of machines behave like one big machine.

## Why do we need it?

When Kubernetes was not there, developers ran any application on a Virtual Machine to isolate the Binaries and Libraries, and also to eliminate the effect of other applications on the current application. But setting up a virtual machine requires some administrative effort and cost as well. And machines will be underutilized if you just dedicate it for just one task, which is how people typically use VMs. This underutilization led to the invention of containers. Think of a container as another form of virtualization. Virtual Machines (VM) allow a piece of hardware to be split up into different VMs – or virtualized — so that the hardware power can be shared among different users and appear as separate servers or machines. Containers virtualize the OS, splitting it up into virtualized compartments to run container applications.

We deploy containers 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 can’t see each others’ processes, and their computational resource usage can be bounded. They are easier to build than VMs, and because they are decoupled from the underlying infrastructure and from the host filesystem, they are portable across clouds and OS distributions.

With containers, immutable container images can be created at build/release time rather than deployment time, since each application doesn’t need to be composed with the rest of the application stack, nor married to the production infrastructure environment.

Generating container images at build/release time enables a consistent environment to be carried from development into production.

Now, there is an inherent problem with containers, just like there is with virtual machines. That is the need to keep track of them. When public cloud companies bill you for CPU time or storage then you need to make sure you do not have any orphaned machines spinning out there doing nothing. Plus there is the need to automatically spin up more when a machine needs more memory, CPU, or storage, as well as shut them down when the load lightens.

Kubernetes helps us manage these omnipotent Containers. It is a single interface to deploy all kinds Virtual Machines, containers on one or many machines.