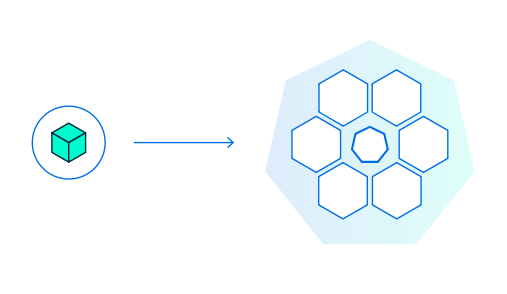
Deploying an app from crow-eye view means this:

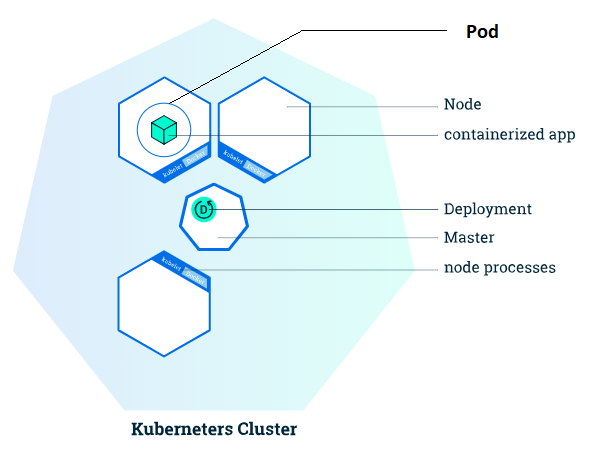


Or rather this: When we deploy our containerized app on Kubernetes, which node (computer, either VM or Physical Machine) will process it is unknown. Master computer will just manage things and entire cluster, but is not used for application deployment purpose.

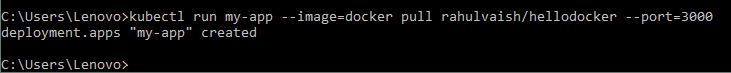
The master starts our containerized app as container (running instances of the containerized app is called container) within a created Pod, which is inside the node. You can also consider Node as RAM, POD as Heap memory and Container as Object.NOTE:Kubernetes does not create containers directly. It will create Pods, with containers inside them

NOTE: Pods are tied to the Node where they are deployed and remain there until termination (according to restart policy) or deletion. In case of a Node failure, new identical Pods will be deployed on other available Nodes. The Pod is the atomic deployment unit on the Kubernetes platform. When we trigger a Deployment on Kubernetes, it will create Pods with containers inside them, not containers directly.

**Kubernetes role is to automate the distribution (scheduling) of application containers across a cluster in an efficient way.**

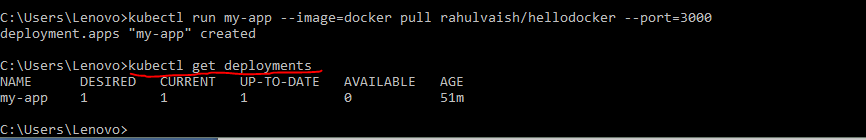


**PRACTICAL:**

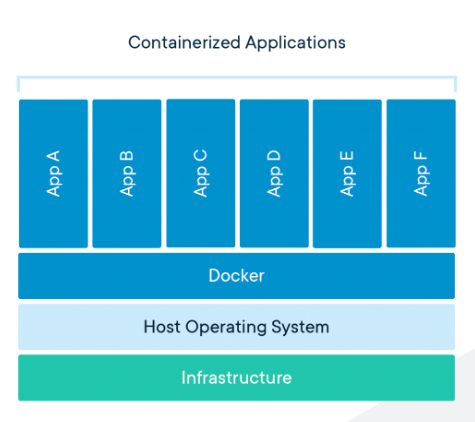
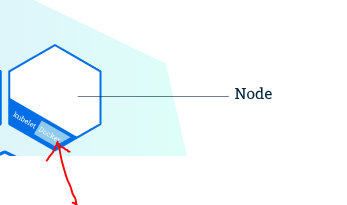
****

**What Kubernetes did, when you executed the above command?**

* This activity searched for a suitable node where an instance of the application could be run (we have only 1 available node)
* Scheduled the application to run on that Node.
* Configured the cluster to reschedule the instance on a new Node when needed

****

We see that there is 1 deployment running a single instance of your app. The instance is running inside **as** a Docker container on Docker Engine of your node.

****

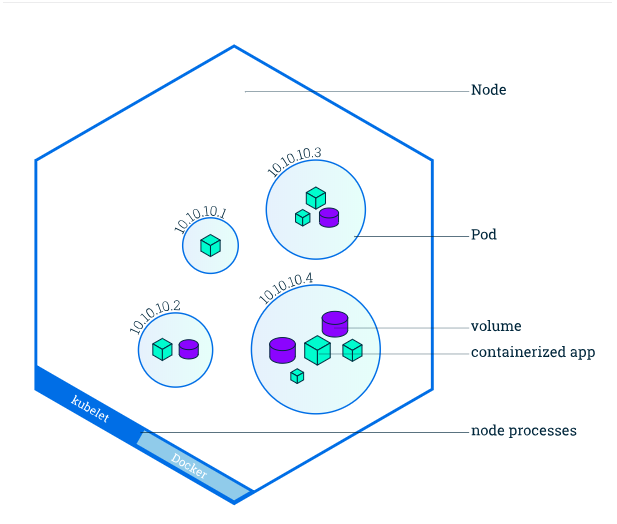
**Fig1-Docker Engine on Node.**

**Fig2-Containerized Applications running as Containers on Docker Engine. Note: We didn’t showcase anything related to POD over here, because that is a part/feature of Kubernetes Project/Framework (If AppA of Fig1 was supposed to execute on Docker Engine of Fig2, then there might be POD showcased) Fig2 is simply a Docker Engine, used to facilitating Docker Containerized Images.**

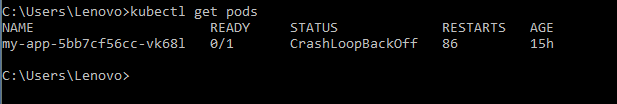
Note From <https://www.docker.com/resources/what-container-> A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A **Container image, or containerized Application, or image** is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

**Container images** become **containers** at runtime and in the case of Docker containers - **images** become **containers** when they run on [**Docker Engine**](https://www.docker.com/products/docker-engine). Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

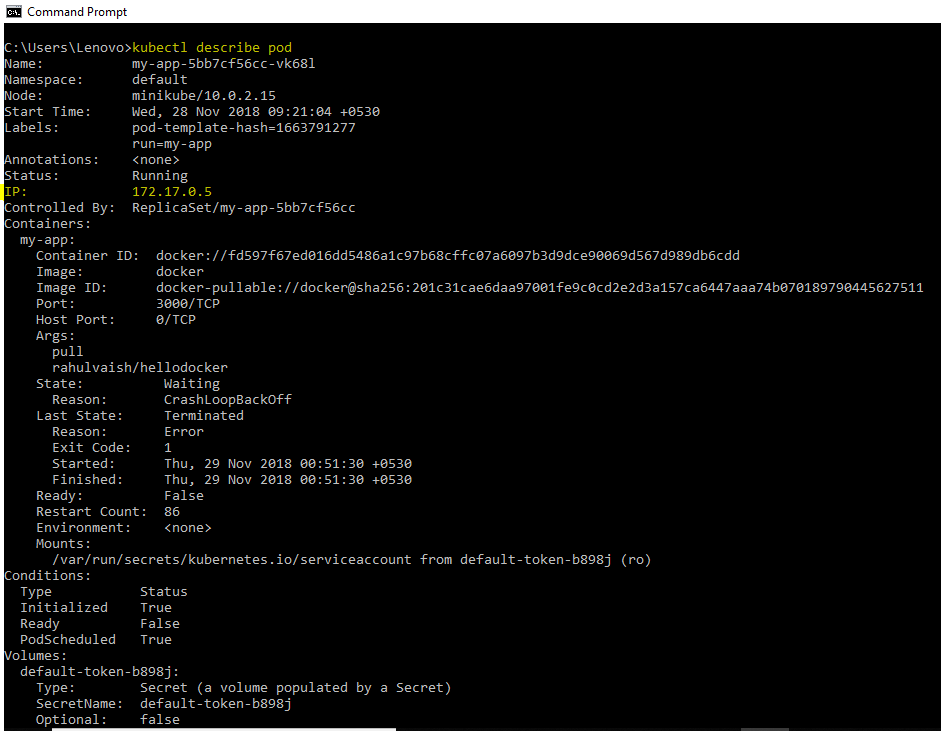
Now after deployment is done, the containerized app (turns into containers) and starts living in a POD within a Node.



POD LEVEL INVESTIGATION: kubectl get pods



INSIDE POD LEVEL INVESTIGATION: describe pods

****