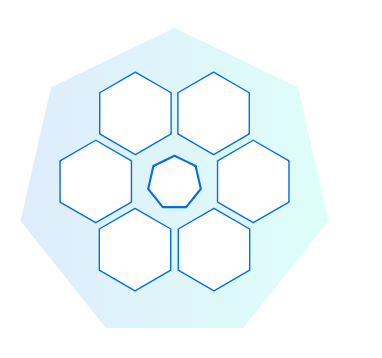
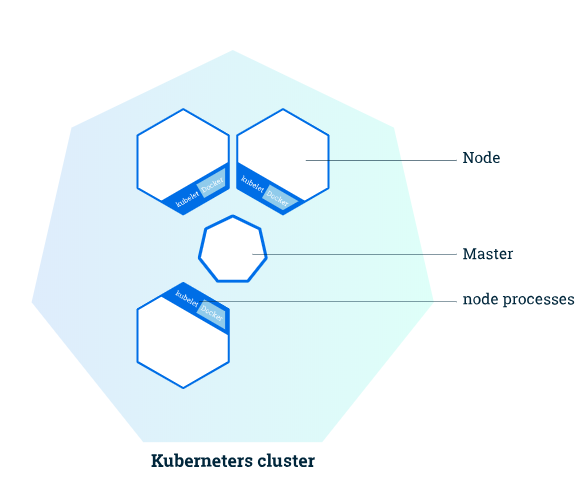
Let’s see and understand everything about Kubernetes.

Kubernetes Platform looks like this from crow-eye view:



Meaning of the above diagram: Kubernetes is a highly available cluster of computers that are connected to work as a single unit. So, all the hexagons in the diagram are interconnected computers. In the middle we have a master computer which is controlling and managing the cluster and all other node computers (where the actual supplication deployment happens).



**DESCRIPTIVE THEORY:**

As we have read that Kubernetes is designed as a highly available cluster of computers that are connected to **work as a single unit.** This abstraction allows us to deploy applications without thinking about which specific machines they need to run on. This means that when we deploy our application which node computer will process it is unknown. Master computer will just manage things and entire cluster, but not used for application deployment purpose.

To make use of this new model of deployment, applications need to be packaged in a way that decouples them from individual hosts: they need to be containerized. This is different compared to how applications were deployed in the past, when they were directly installed on specific machines as packages deeply integrated into the host. **Kubernetes role is to automate the distribution (scheduling) of application containers across a cluster in an efficient way.** Kubernetes is an [open-source](https://github.com/kubernetes/kubernetes) platform and is production-ready.

Kubernetes cluster is formed out of 2 types of resources:

* **Master** is coordinating the cluster
* **Nodes** are where we run applications

**The Master is responsible for managing the cluster.** The master nodes will coordinate all the activity happening in your cluster like scheduling applications, maintaining their desired state, scaling applications and rolling new updates.

**A node is a VM or a physical computer that is used as a worker machine in a Kubernetes cluster.** Every node from the cluster is managed by the master. On a typical node you will have tools for handling container operations (like Docker, rkt) and Kubelet, an agent for managing the node. A Kubernetes cluster that handles production traffic should have a minimum of three nodes.

*Masters manage the cluster and the nodes are used to host the running applications.*

When we deploy applications on Kubernetes we tell the master to start our containers and it will schedule them to run on some node agents. **Communications between the master and the nodes is done via an API exposed by the master.**The same API is exposed towards the users in order to facilitate interaction with the cluster.

A Kubernetes cluster can be deployed on either physical or virtual machines. The recommended way to start a Kubernetes cluster for development purposes is by using [minikube](https://github.com/kubernetes/minikube). **Minikube creates a VM on your local machine and deploys a simple cluster containing only one node**. We have seen installation of minikube in Chapter 0.

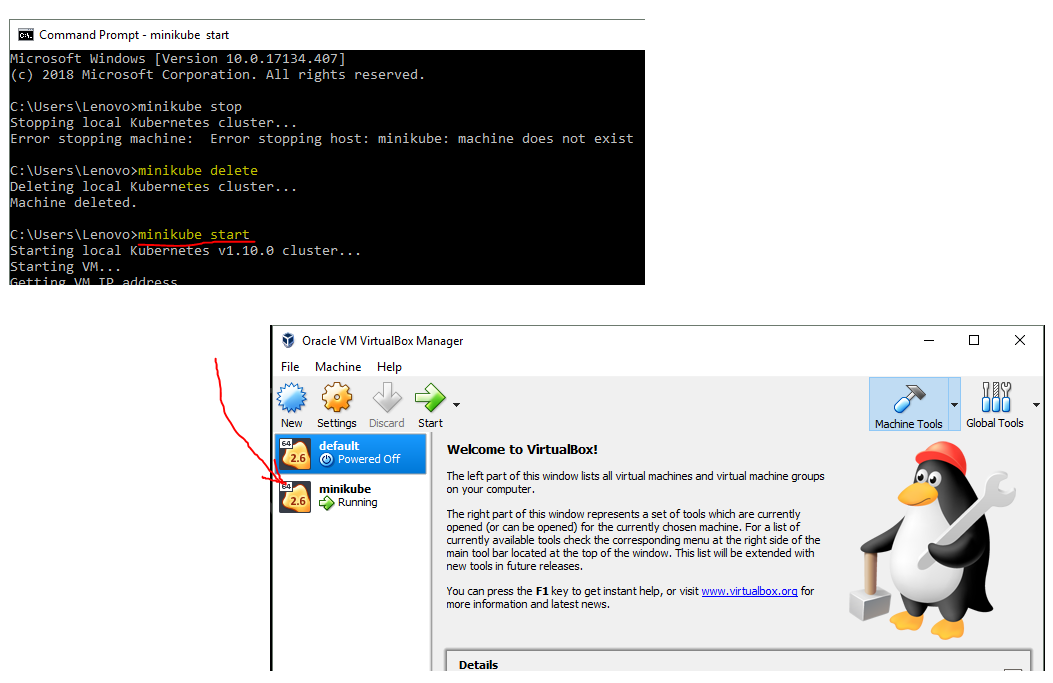
**EXERCCISE:**

Start the cluster, by running the minikube start command:

minikube start

Great! You now have a running Kubernetes cluster in your terminal. Minikube started a virtual machine for you,**(you can see it on Oracle VM)** and a Kubernetes cluster is now running in that VM.

To interact with Kubernetes we’ll use the command line interface, kubectl.



minikube stop –To stop the Kubernetes

minikube delete- To delete the Kubernetes cluster (Runnning on VM)

kubectl cluster-info

We have a running master and a dashboard. The Kubernetes dashboard allows you to view your applications in a UI

kubectl get nodes

This command shows all nodes that can be used to host our applications. Now we have only one node (minikube- a cluster of 1 node), and we can see that it’s status is ready (it is ready to accept applications for deployment).

