<https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16>

<https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-deployment-tutorial-example-yaml.html>

<https://www.edureka.co/blog/kubernetes-architecture/>

**Why Kubernetes**

An open-source system for automating deployment, scaling, and management of containerized applications

If you want use multiple services like more than 20 services, going with .net we need to create 20 projects and difficult to link those services and going with VM’s we need to create multiple VM’s and same difficult to link.

In AKS, we have options to use more than 20 services as micro services (A micro service is an application with a single function, such as routing network traffic, making an online payment or analyzing a medical result). In POD able to deploy multiple services as container and single POD has multiple container

**Docker File**

The Docker file is a text file that contains the instructions needed to create a new container image

**DOCKER**

Example: sometimes developer system your code will work at same in time tester system code will not work. Issue is application is working in one platform and not working in anther platform

Docker will solve the platform related issues

Docker comes in deployment stage of application

Software application consists of frontend, Backend, DB, libraries, server. Therefore, we have to make sure that support and run all the platforms. It is very difficult to run all the s/w components in all possible platform because some server does not support windows or Linux OS, some DB does not work in windows OS like that

Developer will package all the s/w components like frontend, Backend, DB, Server, Libraries into container and Docker will distribute the all the platforms where s/w needs to run



**Containers**

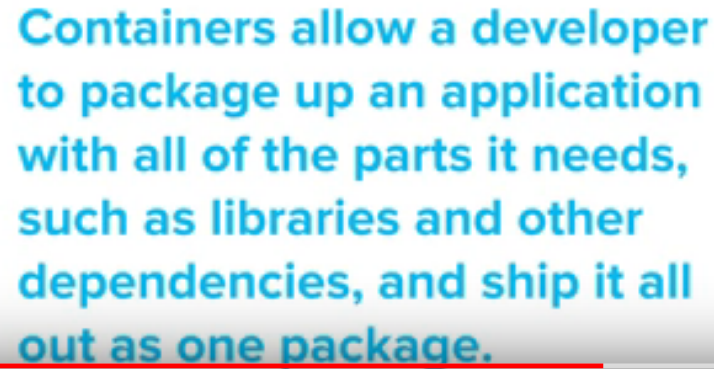
A **container** is a mini-virtual machine

It is lightweight

If you are creating VM, it has all the default libraries and if you want to host .net application or java application except their related libraries, others are unnecessary and did not use at all. 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 and wastage of Resources (RAM, memory, disk space) not utilized properly

If you are container, it has basic OS settings and we can pick the libraries what we want while creating container. To select the required libraries for container is “**DOCKER**”

So container take the RAM, memory, space everything based on application requirements



Using container, if you want to host .net application, it has only .net related libraries so quickly we can access

**Docker Hub**

There are thousands of preconfigured Docker images at the [Dockerhub](https://hub.docker.com/) public repository.

People benefit from that because they can install nginx or even far more complicated items simply by downloading them from there.

**Kubernetes**

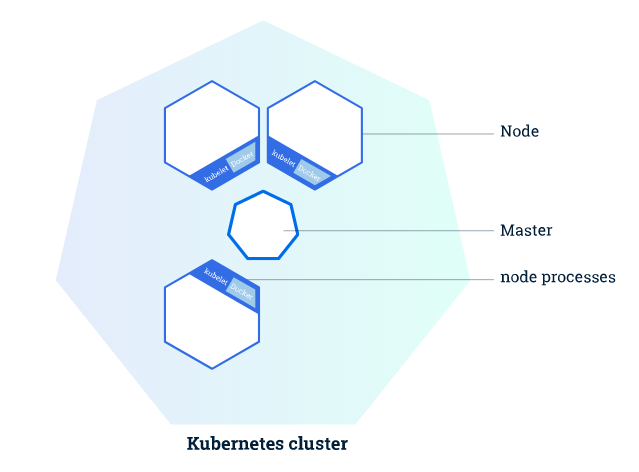
Google pushed Kubernetes into open source

Kubernetes is a cluster and container management tool

It lets you deploy containers to clusters, meaning a network of virtual machines.

 It works with different containers, not just Docker

Kubernetes allow you to deploy containerized applications to a cluster without tying them specifically to individual machines



**The Master is responsible for managing the cluster.** The master coordinates all activities in your cluster, such as scheduling applications, maintaining applications' desired state, scaling applications, and rolling out new updates.

**A node is a VM or a physical computer that serves as a worker machine in a Kubernetes cluster.** Each node has a Kubelet, which is an agent for managing the node and communicating with the Kubernetes master. The node should also have tools for handling container operations, such as Docker or rkt. 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 you deploy applications on Kubernetes, you tell the master to start the application containers. The master schedules the containers to run on the cluster's nodes. **The nodes communicate with the master using the Kubernetes API**, which the master exposes. End users can also use the Kubernetes API directly to interact with the cluster.

**POD**

We are specifying that we want to create a Pod; we might specify instead a Deployment, Job, Service, and so on, depending on what we are trying to achieve

A Pod is a group of one or more [containers](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-containers) (such as Docker containers), with shared storage/network, and a specification for how to run the containers.

A single POD can have multiple containers and these containers inside a single POD shares the same IP and can talk to each other using localhost address.

**Deployment**

A deployment is an object in Kubernetes that lets you manage a set of identical pods.

Without a deployment, you’d need to create, update, and delete a bunch of pods manually

With a deployment, you declare a single object in a YAML file. This object is responsible for creating the pods, making sure they stay up to date, and ensuring there are enough of them running

You can also easily autoscale your applications using a Kubernetes deployment. Suppose 2 applications are running if any one of them not working in this case deployment create new app and replace as well

**Service Endpoint**

To get access to a Deployment with one or many PODs, you need a Kubernetes Service endpoint mapped to the deployment using labels and selectors.