Kubernetes - K8s

What is Kubernetes

- is an open source container orchestration tool, developed by google.
- on the foundation it manages container be it dockers to other technologies.
- Helps you manage containerized applications in different deployment environments like physical , virtual machines or cloud hybrid deployment environment, etc.

What problems dos Kubernetes solve?
What are the tasks of an orchestration tool?

- Trend from Monolith to Microservices.
- Increased usage of containers.
- So this resulted in the applications now comprised of 100 or 1000 of containers, so managing those lots of containers across multiple environments using scripts and self made tools is complex and sometimes even impossible.
- So this actually caused the need for having container orchestration technologies.

What feature do orchestration tools offer?

- High availability or no downtime, its always accessible by user
- Scalability or high performance (high response rates from the app)
- Disaster Recovery - backup and restore , so it actually doest loose the data and the containerised application can run from the latest state after the recovery

K8s Components

- Most basic fundamental components of Kubernetes
- It has tons of components but most of the time we will be using only handful of them

- Node (Worker node) and Pod
    - Lets start with a basic setup of a worker node, which is a simple server, a physical or a virtual machine and the basic component or a smallest unit of Kubernetes is a pod
    - Pod
        - Smallest unit of K8s
        - abstraction over container
            - basically an abstraction over container, if you are familiar with docker containers or images, what pod does is it creates this running environment on or a layer on top of the container and reason is that because k8s wants to abstract away the container runtime or

container technologies so that you can replace them if you want to and also because you don't have to directly work with docker or whatever container technology you use in k8s
    - So you only interact with the k8s layer
    - So we have an application pod which is our own application and that will maybe use db pod with its own container and this also an important concept here as pod is usually meant to tun inside of it,
        - Usually 1 application per pod
    - you can run multiple containers inside one pod but usually its only the case if you have one main application container and a helper container or some side service that has to run inside of that pod
    - So as you see, this is nothing special, you just have one server and two containers running on it with a abstraction layer on top of it (pods)
- Now lets see how they communicate with each other in k8s world
    - K8s offers out of the box a virtual network which means that each pod gets its own ip addrs not the containers
    - And each pod can communicate with each other using that ip address which is an internal ip address obviously not the public one
    - So my app can communicate with database using the ip address however pod components in k8s are ephemeral which mean that they can die very easily and when that happens for ex if I lose the db container because of some crash or anything like that the new one will get crated in its place and when that happens it will get assigned new ip address this might be inconvenient if you are communicating with the database using the ip address because now you have adjust it every time pod restarts,
    - so because of that another component if k8s called service is used. Service and ingress
- Service and ingress
    - Service
        - It has 2 functionalities
            - Permanent IP
            - Load balancer
        - Service is basically a static ip address or permanent ip address that can be attached to each pod, so my app pod will have its own service and db pod will have its own service and the good thing is
        - Lifecycle of pod and service aren't connected so even

if the pod pod dies the service and its ip address Weill retain , so don't have change that the endpoint all the time when something happens.
        - So at some point it is obvious that you want you are app to be accessed from the browser right? So for this you would have to create an external service
        - External services is a service that opens the communication from the external sources
        - But obviously you don't want your db to be accessed from the outside like open to the public, so for that you would create internal service
        - This are the types of the service that you will specify while creating the service.

    - Ingress
        - Used to route traffic into the cluster
        - If you see the url, it will have an IP address of the node (not the service) and port no. of the service - http://123.78.101.2:8080, this is good for testing and not for the end product , for the end product you will something like https://my-app.com, so for that k8s has one component called ingress
        - Instead of service the request first goes to ingress and it does the forwarding to the service.
- so this are all the most basic component of k8s not that are special where k8s advantages, you will see the actual cool features while learning ahead

- ConfigMap and Secret
    - ConfigMap
        - External configuration of you application
        - Centralised
        - Pods communicate with each other using a serve so my application will have a db endpoint lets called mongo-db-service but where do you configure these configs, usually you do it inside the application itself like ex: env variables.
        - Incase if the name of the service changed then you would have to change in the codebase and re build push restart and do everything that you usually do what you do while deploying, so for that purpose k8s has something called "ConfigMap"
        - ConfigMap
            - External configuration of you application
            - Centralised
            - It usually contain configuration data like Urls of db and other services that use and in k8s you just connect it to the

pos so that pod actually gets the data that configMap contains, so now you can update/modify just the configMap.
        - Can also username and pwd be out in there right? But don't put credentials into ConfigMap, it is insecure to put there.
    - Secret
        - So for this purpose k8s has another component called secret, its just like ConfigMap but it is used to store secret data.
        - Will be stored in base64 encoded format
    - Built in security mechanism is not enabled by default!
    - You can use these data from ConfigMap and Secret inside of your app pod using for ex as environment variables to as properties file

- Volumes
    - K8s explicitly doesn't manage data persistence
    - Another very important component of k8s
    - By some means if pods get restarted then the db pod might loose the data as it will not be persisted.
    - The way in k8s to achieve or persist the data is by the component called Volumes
    - And how it works is that is basically attaches a physical or virtual storage on a hard drive to your pod and that storage could be either on a local machine meaning same server node where the pos is running or it could be on the storage which is outside of k8s cluster like remote/cloud storage

- Deployment and StatefulSet
    - Assume if the app pod gets destroyed or deleted or crashed or restored while updating the new container, we could have down time at those points which is bad thing in prod, this exactly what distributed systems and containers solves or the advantage of.
    - Instead of relying on just one pod or one db pod etc, we are replicating everything (like another node of all pods), which will also connected to the service which will have load balancer as well, which usually will catch the request and forward it to the least busy pod
    - In order to create the second replica or to replicate n numbers, we will not create the second part but instead would define a blueprint, for which k8s has the component called deployment
    - Deployment
        - Define blueprints for/of pods with specifying the number of replica
        - Can be scaled up or down in terms of replica
        - It is another abstraction on top of pods (like pods is

abstraction of containers)

- StatefulSet
  - Will make sure the database read and writes are synchronised so no database inconsistencies are offered.
  - DB (pods) can't be replicated via deployment and the reason for that is because db has a state which is its data
    - If we have clone or replicas of the database they would all need to access the same data storage and would need some kind of mechanism that manages which pods are currently written to that storage or which pods are reading from the storage in order to avoid data inconsistencies and that mechanism is offered by another k8s component called "StatefulSet"
  - This is specifically meant for applications like databases (mysql, mongoDb, elastic search)
    - Deploying StatefulSet is more difficult and tedious so its a common practice to host a db outside of k8s cluster and just have have the deployments (above one)/ stateless app deployment inside the k8s cluster