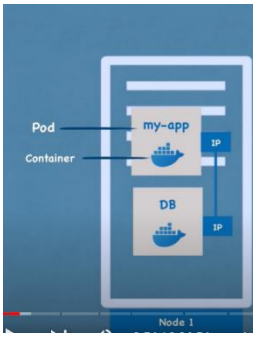
Kubernetes

MAIN COMPONENTS (17.02.22)

Worker node



simple server, physical or virtual machine

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.

Pod – abstraction of container

The smallest unit of Kubernetes (usually 1 container per pod), creates a running environment (layer on top of container) for the purpose of abstraction away from container runtime/technology (elevate users away from deeper mechanics) so that users can replace them if required. Users only interact with the Kubernetes layer.

>Can be multiple containers within a pod in cases where there is a main application and helper container/side services which must run within the pod

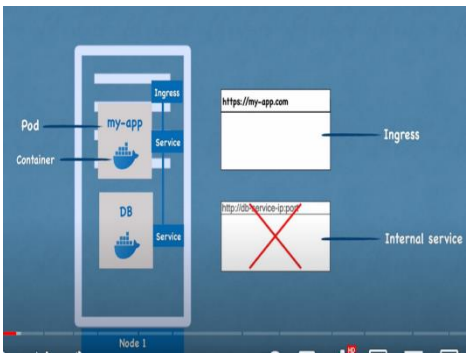
Pod IP

Each pod gets its own IP address meaning they can communicate to each other (private) E.g application pod can communicate to db pod using IP

>Pods are ephemeral so they can die easily – when this occurs a new pod is automatically created and is assigned a new IP – every time a pod dies we have to adjust the Ip for communication

Service - communication

Service is a permanent IP address that can be attached to each pod – lifecycle of service and pod are not connected.



It also acts as a load balancer (sends to least busy pod)

External Service

Allows container to be accessed from external source using an IP

Internal Service

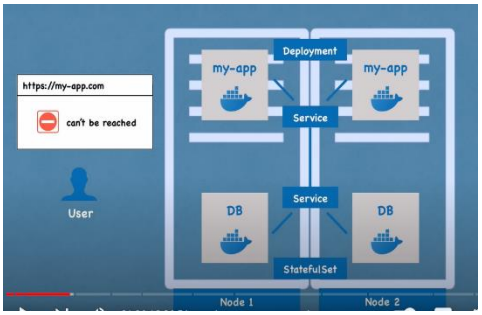
Does not allow container to be accessed from an external source using an IP

Ingress – route traffic into cluster

Service Ip often are unattractive consisting of service IP with the port number, Ingress offers a customable IP which users access and it forwards them to the service IP.

ConfigMap – external configuration (plain text)

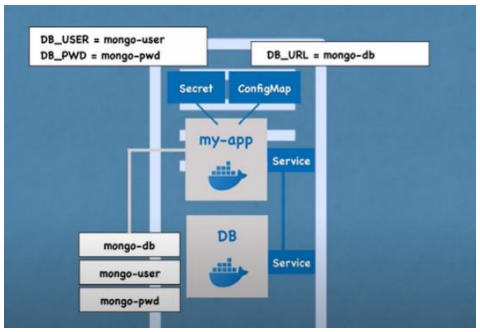
The database URL is often in the built application (properties file) - if the URL changes you would have to adjust the URL in the application (rebuild, push to repo, pull to pod) – which is a lot of work.



ConfigMaps are used to manage the external configuration of your application (URL, External services etc). This is then connected to the pod – so when adjusting the name of a service we only have to update config map

Secret – external configuration (encoded)

Secret operates like a config map but its used to store secret information such as user names and passwords (credentials) e.g user names, passwords etc. This is stored in base 64 encoded



>Can use data from config map or secrets using environment variables or as a properties file

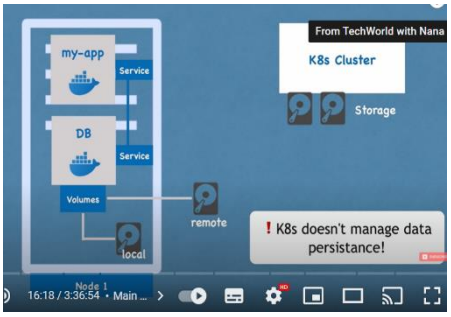
Volumes – data persistence

If the database or container gets restarted the data would be gone. Volumes attach a storage to your pod. This storage could be on a local machine (same server node that the pod is running) or remotely outside k8s cluster (e.g cloud or different server). The data is persisted and remains available after restarts.

>Have to recognise separation of k8s cluster and storage as k8s doesn't manage data persistence. You as a user or admin are responsible for backing up, replicating and ensuring the data is stored on appropriate hardware.

Deployment – replicating mechanism (stateless)

If I restart my application or the pod dies, there's downtime where users cannot reach my application – this is bad In prod. Instead of relying on one node we replicate everything in a separate node and connect to the same service – as services act as a load balancer users a directed to the container which is least busy.



When creating the replica pod of our application we don't create a second pod and instead define a blueprint (deployment) and specify how many duplicates we require. In application we would not be creating pods we would instead be creating deployments – as we can specify how many instances we need and scale up and down based on requirements

>deployments are an abstraction of pods

>deployment for stateless Apps

Statefulset– replicating mechanism (stateful)

Databases cannot be duplicated via deployment as they have states – if we have clones of DB they need to access shared data storage this means we need a mechanism which manages which pod can

write/read to/from the storage to avoid data inconsistencies. Stateful set is directed towards stateful apps – like deployment statefulset is responsible for replicating the pods and scaling them up or down

>statefulset for stateful apps or databases