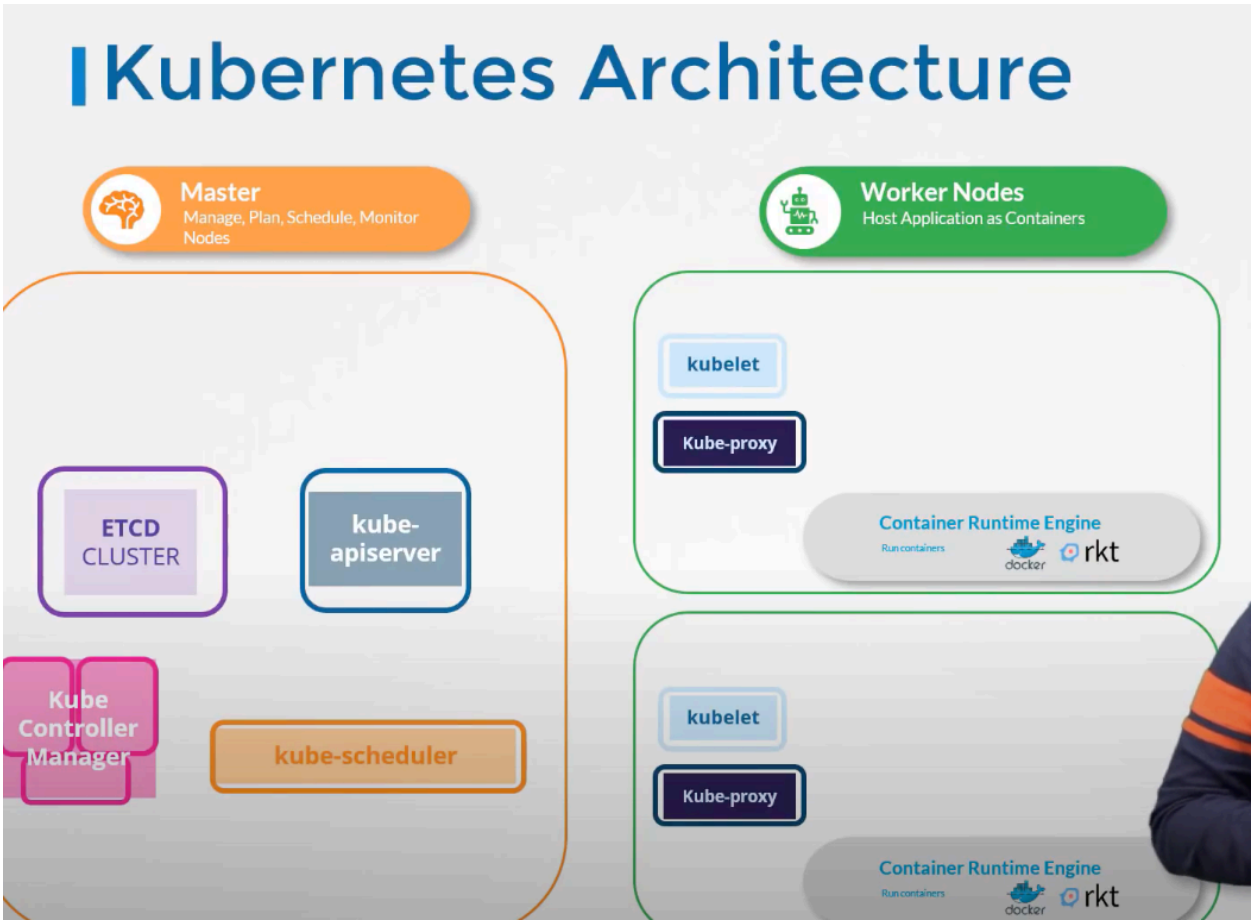
Thursday, October 1, 2020

Kubernetes Explained

Architecture

- Worker nodes : host application as containers
- Master : manage, plan, schedule, and monitor nodes
  - Using control plane components
- ETCD CLUSTER : key, value database to help manage the nodes on the cluster
- Kube-scheduler : identifies the right node based on the container's resources requirements, workload capacity, and other policies/constraints.
- Controllers:
  - Controller-manager
  - Node-controller
    - Adding new nodes to the cluster,
  - Replication-controller
    - Ensures desired number of containers are running in a desired group
- Kube-apiserver: responsible for orchestrating all operations in the cluster
- Docker : container runtime engine
- Kubelet : master of the ship, manages each node on the cluster, responds to instructions from the kube-apiserver, creates or destroys accordingly
  - Periodically gets status reports
- Kube-proxy: ensures rules are in place on the worker nodes so they nodes can properly reach each other



Components

Node and Pod

Node - any physical computer

Pod - smallest unit of K8, the smallest deployable unit

- abstraction over container
- Usually 1 application per pod
- Each pod gets its own ip address (internal)
- Pods can communicate with each using ip add

- New ip add on re-creation of pods

Services and Ingress

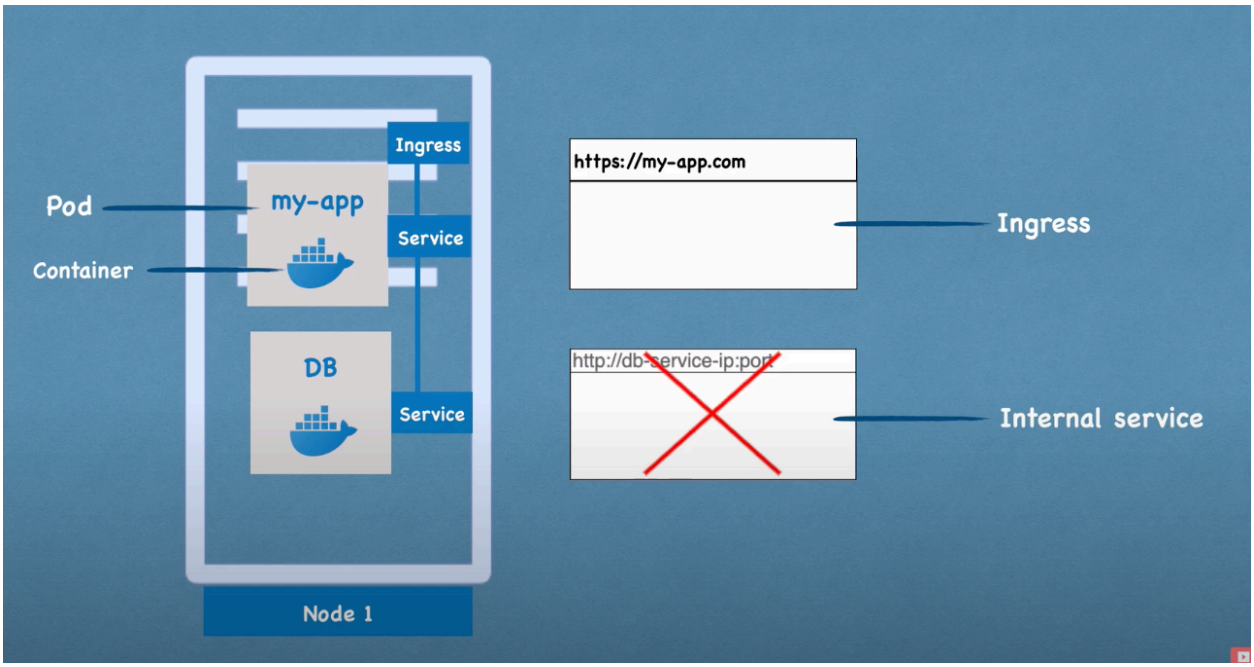
Service: persistence, static ip add

- life cycle not connected to pod, so if pod dies, service is still active

External service: allows application to be accessible by the public

Internal service: allows applications to be accessed in private

Ingress: forwards request from public to internal services



ConfigMap and Secret

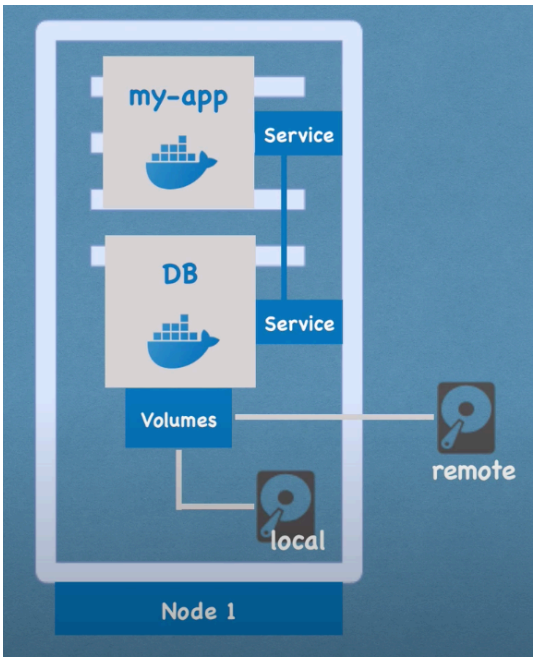
ConfigMap: configuration file that is referenced for applications

Secret: like ConfigMap, but used to store secret data, encoded in base64

Volumes

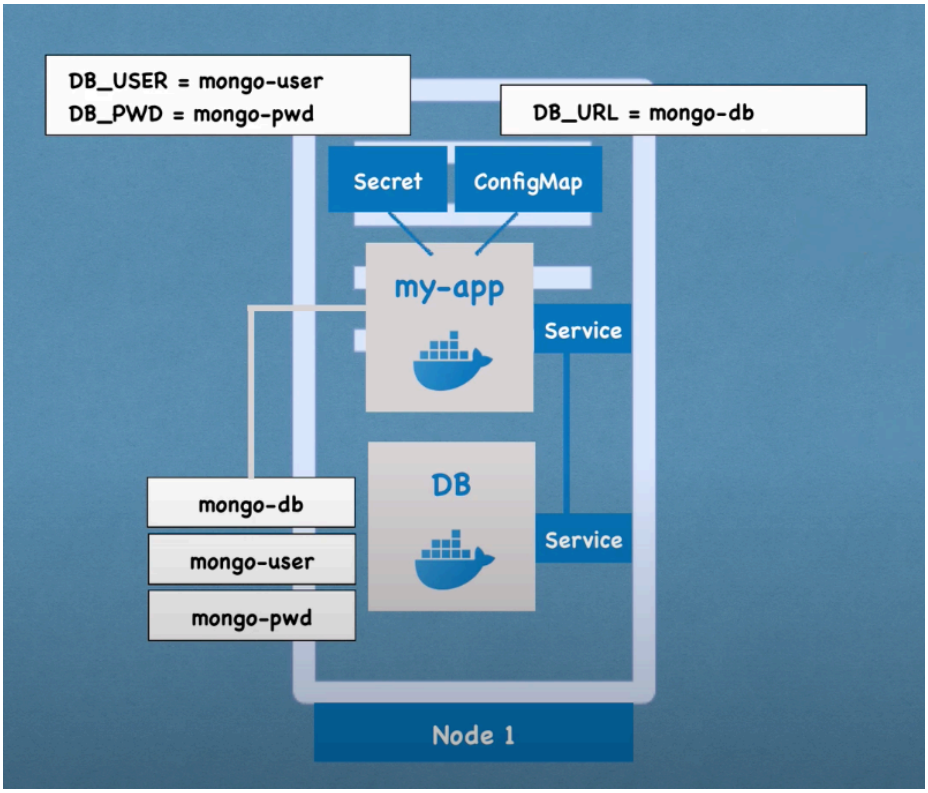
Volume: data storage (either local or remote) to back up a database for persistence

Thursday, October 1, 2020



K8s doesn't manage data persistence, so storage is separate from the data management itself.

Deployment and Stateful Set



Service is a load balancer: it'll catch a request and send it to least busy pod.

To create a second replica (node 2) you wouldn't literally create a second pod, you'd create blueprint and specify how many replicas of that pod you'd like to run. That blueprint is called a deployment.

Databases can't be replicated using deployment, because the data can't be replicated.

Stateful set solves that -> it'll replicate pods and scale them, as well as synchronize them so no data is lost