Kubernetes

YAML

- Kind deployment or kind service
- ETCD holds the cluster status
- Deployment has configuration and then also pod configuration
- Key value pairs for components app: nginx
- Targetport should match the containerport, else they cant communicatie with each other
- kubectl get pod -o wide for more information

———

Deployment

- Replica’s specified the number of pods at any given time
- Selector identifies which pods belong to this deployment
- Labels are attached to the deployment itself
- The selector must match the labels applied to the deployment itself
- Starts at spec

Pod

- Starts at template

Containers

- Starts at containers

Service

- Selector has a name that references the pod so in the mongo db example it was mongodb
- To expose the service to the outside use LoadBalancer type and add nodeport 30000-32767

——

Deployment is the what, the service the how. They rarely work without each other

Namespace

- kubectl create namespace <bla>
- kubectl create ns

Why do I need a namespace?

- Separation
  - Database

    - Monitoring
    - Elastic stack
    - Nginx ingress
- Multiple teams in a kubernetes cluster
    - Only access to their own cluster

—
Change the namespace instead of default
- kubectl config set-context —current namespace=<namespace>
-

Services

Stable IP adress
Loadbalancing
Loose coupling
Within & Outside cluster

ClusterIP service
- Gives an Ip range
- The third number in the IP specifies the node which is on
- Service registers endpoints in the deployment app: my-app to each other, so it knows they belong to each other.
- Service will find the pods based on labels and selector and send the request based on targetport
- Service target port needs to match the containerport van the pod
- If you have multiple ports, you have to name those ports.

Headless service
- Most commonly used with databases (Mysql, postgress or mongo) **statefulset**
- Pods are not identifical
- DNS lookup for a service
- clusterIP None

Nodeport service
- Predefined values between 30000 to 32000
- External traffic
- Testing service
- Not for production

Loadbalancer service
- Server is accessible from outside

StatefulSet

- Stateful applications
  - That track state all databases
  - Mysql main, and 2 replicas if you have 3 database pods. Data is else not interchangeable
  - They dont have access to the same storage
  - Use data persistence storage
  - Own DNS
- Stateless applications
  - Dont keep records of state

HELM

- chmod 400 k8s-test.yaml allows only your user to access the kube config
- export KUBECONFIG=
- Helm install <name we give> —values helm-mongodb.yaml bitnami/mongodb

CONFIG MAP & Secret volume

- Config map & Secret are used for external configuration
- Adding the volume files first into the pod, but that does not mean its in the container. The volumeMount needs to be in the container level
- VolumeMount name is the name of the volume that is attached to the POD.
- kubectl get configmap prometheus-config -n default -o yaml
- Left part of docker compose (host path is volume in Kubernetes
- Right part of docker compose is volumeMount and specifies the container

Volumes

- **Persistent volume**
  - Need physical storage
  - Available to the whole cluster
  - Local
    - Tied to one specific node
    - Surviving cluster crashes
  - Remote volume type
    - DB persistence should use remote storage
  - K8s admin
    - Sets up and maintains the cluster
  - K8s user
    - Deploys applications in cluster
    - Explicitly set the Persistent claim

    - Pod needs to use PV claim and claim name needs to be same as the Persistent Volume claim yaml
    - Left part of docker compose (host path is volume in Kubernetes
    - Right part of docker compose is volumeMount and specifies the container (right
    - subpath: Maps the file, if you would not specify the submap it would take the entire directory
- **Persistent volume claim**

———————————

- **Storage class**
    - Uses yaml as well
    - provisioner is needed in the yaml
    - Requested by PVC, storage classname needs to be defined in the yaml in order to specify

Kubernetes operators

- **Stateless applications**
    - k8s can manage this in automated way

- **Stateful applications**
    - Should be updated and destroyed in order
    - Need manual intervention
    - **Operator**
        - Deployment
        - service
        - configmap
        - Statefulset
        - **CRD**
- Operatorhub

RBAC

- Permissions for a cluster
- Role only defines resources and access permissions
- Role binding (Link role to a user or user group)
- Cluster role defines what resources have what permissions, clusterwide.
- External sources for user management
    - Admin will configure external sources
- Application only get permissions to work
- Service account component inside the cluster as a represnetation of that user
- Rules
    - API groups
    - Resources (pods)

    - verbs (you can add crud functionality here)
    - Resourcenames (here you can specify in example the app or the db and specific permissions for those items

Kubectl commands

- kubectl get pods (gets the pods)
- kubectl get services (gets services)
- kubectl create deployment nginx-depl --image=nginx
- kubectl get replicaset
- kubectl logs <podname> this will log
- kubectl describe pod mongo-deployment-555654868f-xn2q7
- kubectl exec -it <podname> — bin/bash
- Deployment > Pod > Container
- kubectl get all | grep mongodb
-

Ingress

- Ingress controller is needed to be able to use ingress and is separated from the release
- TLS certificate needs to be hosted in the cluster to give it https
- Only values no files
- Single entrypoint into the cluster