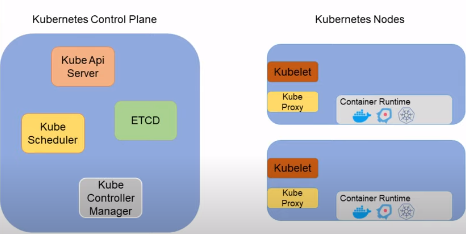
**KUBERNETES**

É formado por cluster, um conjunto de maquinas. Cada uma delas exerce um papel possível:

**Control plane**: Orquestra e gerencia. Precisa de redundância para alta disponibilidade.

**Node**: Quem executa os containers das aplicações



API SERVER: Comunicação com o cluster

Scheduler: Onde cada processo vai ser executado

ETCD: banco chave valor q armazena dados do kubernetes

Controller manager: executa e gerencia os controladores. Gerencia tomada de decisões, autenticações, admissoes etc

Kubelet: agente de inspeção e execução. Ele monitora e executa os containers e interage com o api server

Kube proxy: conexões de rede

Container Runtime Interface : especificações necessárias pro runtime executar dentro do kubernetes para executar os containers. O Docker não implementa um runtime interface, mas sim o DOCKERSHIM como adaptador. Containerd e cri-o são substitutos para o Docker e implementam o RI.

On premise: Kube-adm e kubes-play

K3d: baseado em K3s, bem simples. Roda o K3S em um container.

Kubectl : linha de comando pro Kubernet

Choco install K3D

choco install kubernetes-cli

Criar cluster padrão com 1 nó apenas (possui load balance por default):

k3d cluster create clusterchel

Para listar os nodes:

Kubectl get nodes

Criar cluster com 1 nó e sem o load balance:

K3d cluster create clusterchelNoLb --no-lb

Listar os clusters:

K3d cluster list

Deletar cluster:

K3d cluster delete NOME

Criar cluster com 3 servers e 3 nodes:

K3d cluster create chelcluster3s3a --servers 3 --agents 3

Listar recursos para api:

Kubectl api-resources

3 objetos do cluster kubernete:

**Pod**: Menor objeto do cluster. Nele q vai executar o container. Pode ter mais de um container dentro de um mesmo pod. Mesmo ip e pode dividir um sistema de arquivos. Não é bom pra replicar. Sidecar: container principal (aplicação principal) + containers auxiliares (exemplo logs, métricas ...). Ex service mesh

**ReplicaSET:** Controlador de resiliência e escalabilidade, controlando o numero de replicas . Quando atualiza a imagem, os pods não são atualizados automaticamente. Apenas quando são removidos e recriados.

**Deployment:** Facilidade de replicar e fazer o deploy e atualização dos pods. Quando atualiza uma imagem, ele guarda o replicaset antigo como backup.

**POD**:

Primeiro precisa de um manifesto yaml , especificando o q quer no objeto.

Para criar um pod a partir de um manifesto (se apontar para a pasta ele cria todos dentro da pasta):

Kubectl create -f .\chelpod.yaml

Listar pod:

Kubectl get pods

Kubectl get pods -l LABEL

Kubectl get pods -o wide

Deleta pod:

Kubectl delete pod chelpod

Decribe pod:

Kubectl describe pod chelpod

Para fazer um port bind:

Kubectl port-forward pod/chelpod 8088:8088

Label: chave e valor para marcar objetos

Selector: seleciona a chave/valor

**REPLICASET**:

É necessário o selector e o label no arquivo yaml e Utilizar apiVersion: apps/v1 no yaml.

Criar ou Atualiza replicaset:

Kubectl apply -f .\chelreplica7.yaml

Listar replicaset:

Kubectl get replicaset

Mudar scale por linha de comando:

Kubectl scale replicaset chelreplica7 --replicas 20

**DEPLOYMENT**:

Igual ao replicaset, apenas tem que mudar o nome do kind para Deployment no arquivo yaml.

Listar deployment:

Kubectl get deployment

Describe:

Kubectl describe deployment cheldeploy

Rollback da versão anterior:

Kubectl rollout undo deployment cheldeploy

**SERVICE**:

Service Discovery, pega o endereçamento e ele decide qual pod.

ClusterIP: Expor um pod de forma interna ( pod A acessando pod B por exemplo)

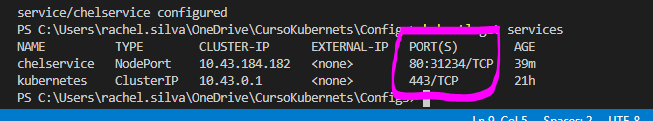
NodePort: Expor externamente. Acessa ele através de qualquer IP das maquinas. Porta é mapeada em todas maquinas. Porta gerada de forma aleatória (30000 a 32767, mas pode mudar no setup). Usa muito em On Premise.

LoadBalancer: Expoe externamente e é Integrado com nuvem.

Utilizar apiVersion: v1 no yaml

Listar services:

Kubectl get services



O service gera uma porta aleatória, mas ainda não é acessível. É necessário fazer o bind de portas. Para isso é necessário especificar a porta no arquivo yaml a partir do atributo

nodePort: XXX abaixo do ports.

Necessario recriar o cluster para fazer o bind usando o comando:

K3d cluster create clusterchel --agents 3 --servers 3 -p “8080:31234@loadbalancer”

Listar todos os elementos:

Kubectl get all