Este contenedor en esta máquina etc, porque hay diferentes servidores y apps en una empresa.

Kubernetes es un orquestador de contenedores.

Minikube: herram que levanta una maquina virtual de un cluster de un nodo de k y permite interactual con el. Interactua con maq virtual. ES UN CLUSTER DE K DE UN NODO

Kubectl herram que maneja el cluster

INSTALACIÓN:

[Instalar kubectl ⚓️ #10 (youtube.com)](https://www.youtube.com/watch?v=xNMRtEEbCBc)

[Instalacion virtualbox ⚓️ #11 (youtube.com)](https://www.youtube.com/watch?v=uquQ9xT7oQM)

[Instalar minikube ⚓️ #12 (youtube.com)](https://www.youtube.com/watch?v=okV7a0s_gcE)

CODIGO

[GitHub - inigoserrano/cursoKubernetes: Contenido del curso Kubernetes sencillo para desarrolladores](https://github.com/inigoserrano/cursoKubernetes/tree/master)

Rm –rf .kube para borrar directorios si solo tengo un pod

Minikube config set memory 4096

Minikube config view

Minikube profile cliente2

Cluster virtual es namespace

Kubectl get namespace: listado de namespace

Kubectl create namespace curso1

Con una plantilla se crea asi: kubectl apply –f <nombreficheroyml>

apiVersion: v1

kind: Namespace

metadata:

name: curso-namespace

Kubectl delete namespace <nombrenamespace>

Un pod normalmente tiene un contenedor cno su IP, memoria...

apiVersion: v1

kind: Pod

metadata:

name: wildfly

spec:

containers:

- name: wildfly

image: jboss/wildfly

Kubectl get pods

Kubectl describe <nombrepod>: ver config

Kubectl exec –it <nombrepod> /bin/bash: para entrar en un pod

Kubectl logs –f <nombrepod>

Escalada horizontal vs una instancia: aumentar instancias de un servicio, replicar

Deployment: creacion de pods y replicaset

apiVersion: apps/v1

kind: Deployment

metadata:

name: wildfly-deployment

spec:

selector:

matchLabels:

app: wildfly

replicas: 1

template:

metadata:

labels:

app: wildfly

spec:

containers:

- name: wildfly

image: jboss/wildfly:15.0.0.Final

ports:

- containerPort: 8080

Servicio: hasta ahora sabíamos cuantas instancias queremos de un pod, pero estos son mortales.

Forma de exponer diferentes pods, el sabe cuantos tiene y redirige el trafico o la carga

Kubectl get deployments.apps

cat /etc/hosts

Para comunicarnos entre servicios necesitamos una bbdd

apiVersion: apps/v1

kind: Deployment

metadata:

name: mariadb-deployment

spec:

selector:

matchLabels:

app: mariadb

replicas: 1

template:

metadata:

labels:

app: mariadb

spec:

containers:

- name: mariadb

image: mariadb

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "123"

apiVersion: apps/v1

kind: Deployment

metadata:

name: wildfly-deployment

spec:

selector:

matchLabels:

app: wildfly

replicas: 1

template:

metadata:

labels:

app: wildfly

spec:

containers:

- name: wildfly

image: jboss/wildfly:15.0.0.Final

ports:

- containerPort: 8080

kind: Service

apiVersion: v1

metadata:

name: mariadb-service

spec:

selector:

app: mariadb

ports:

- protocol: TCP

port: 3306

targetPort: 3306

Creamos la plantilla de un servico y los deployments de ambos

Targetport: puerto interno del container

Port es el externo

Kubectl get service

Servicio con quien engancha y el despliegue de que es (selectores)

INGRESS(routes): tenemos servicios que solo son accesibles dentro del cluster, y con el ingress podemos exponerlos para ser accesibles por fuera. Configuraciones de rutas, exponer el puerto...

Una vez creado el servicio

kind: Service

apiVersion: v1

metadata:

name: wildfly-service

spec:

selector:

app: wildfly

ports:

- protocol: TCP

port: 8080

targetPort: 8080

Creamos el ingress

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: wildfly-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /

backend:

serviceName: wildfly-service

servicePort: 8080

Minikube addons enable ingress

Minikube ip para saber la direccion IP del servicio expuesto, en el navegador se puede probar

COMO COMPILAR APP:

En w no se puede tener una maq virtual corriendo por un lado y por otro kubernetes asi que o bien docker desktop con v nativa o bien usando el motor de docker que viene dentro de minikube

minikube docker-env

Eval $(minikube docker-env)

Docker run –it —rm —nam.....

Docker ps: los containers ejecutandose dentro del runtime de docker

APP:

[cursoKubernetes/Bloque11Aplicacion at master · inigoserrano/cursoKubernetes · GitHub](https://github.com/inigoserrano/cursoKubernetes/tree/master/Bloque11Aplicacion)

Para usar una img de maven en vez de instalar:

[Course: Kubernetes sencillo para desarrolladores | Udemy Business](https://indra.udemy.com/course/kubernetes-sencillo-para-desarrolladores/learn/lecture/14899414#notes)

METER LA APP EN EL SERVIDOR:img del servidor de apps y de mariadb

Kubens default

Kubectl delete –f .

Kubectl get events –w

Kubectl logs –f wildfly-deplyment

CONFIGMAP Y SECRETOS:

Primero tenemos que preparar la app para que no esté hardcodeada, copiamos la carpeta app y modificamos aplicación.java (previa creacion de clase). En la carpeta k8s dentro del bloque 14 tenemos tanto los configmaps como los secrets

Dentro de la carpeta de app hacemos eval y lanzamos unstruccion para docker run bajarse las dependencias y generar la imagen. Copiamos el fichero en el wildfly en la app:

Kubectl delete namespace <nombre> y dentro de k8s apply namespace.yml

Kubectl create confirgmap <nombre> —from-file=config-properties –n <nombrenamespace>

Kubectl create secret generic <nombre> —from-file=secretos-properties –n <nombrenamespace>

Para poder usarlos hay que modificar el deployment del wildfly poniendo en vez de var volumenes que se corresponden con configmap y secreto y después se montan vol en contenedor

Que nombre queremos darle al mount volumen y la ruta