docker container run nginx

docker container run --name machine -ti ubuntu

docker container ps -a

docker container exec -it machine /bin/bash

docker start machine

docker container stop $(docker container ps -aq)

docker container start $(docker container ps -aq)

docker container rm $(docker ps -aq)

etape de creation site web mdify page index:

docker container run -d -p 8080:80 --name web1 nginx

docker container exec -it web1 /bin/bash

echo "Bonjour" > /usr/share/nginx/html/index.html

--------------------------volume-----------------------------:

docker create volume wasse

docker run -v wasse:/usr/share/nginx/html -d -p 8080:80 nginx

docker volume inspect wasse

echo 'wasse' > /var/lib/docker/volumes/wasse/\_data/index.html

2.Bind Mounts :

docker run -v /chemin/sur/hote:/chemin/dans/conteneur image\_docker

--------------------------docker-file-----------------------------:

Dockerfile

ROM ubuntu

RUN apt update -y && apt install nginx -y

WORKDIR /var/www/html

COPY index.html .

EXPOSE 80/tcp

CMD ["/usr/sbin/nginx", "-g", "daemon off;"]

docker image build -t wasse .

docker run -d -p 8077:80 --name IMAGE wasse

----------------------------compose-------------------------------:

version: '3.3'

services:

wordpress:

image: wordpress

depends\_on:

- mysql

ports:

- 8080:80

environment:

WORDPRESS\_DB\_HOST: mysql

WORDPRESS\_DB\_NAME: wordpress

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

volumes:

- ./wordpress-data:/var/www/html

networks:

- my\_net

mysql:

image: mariadb

environment:

MYSQL\_ROOT\_PASSWORD: wordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

volumes:

- mysql-data:/var/lib/mysql

networks:

- my\_net

volumes:

mysql-data:

networks:

my\_net:

----commandes compose---

sudo apt install docker-compose

docker-compose up -d

docker-compose down

----------------swarm--------

docker swarm init --advertise-addr @ip

docker node ls

docker network ls

docker service create --replicas 5 -p 8099:80 --name web2 nginx

docker service ls

docker service ps web2

docker service scale web=10

docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer

----------------stack----------

compose.yaml:

version: '3.3'

services:

wordpress:

image: wordpress

depends\_on:

- mysql

ports:

- "80:80"

deploy:

replicas: 2

placement:

constraints:

- node.role == manager

environment:

WORDPRESS\_DB\_HOST: mysql

WORDPRESS\_DB\_NAME: wordpress

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

volumes:

- wordpress-data:/var/www/html

networks:

- my\_net

mysql:

image: mariadb

deploy:

replicas: 1

placement:

constraints:

- node.role == worker

environment:

MYSQL\_ROOT\_PASSWORD: wordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

volumes:

- mysql-data:/var/lib/mysql

networks:

- my\_net

networks:

my\_net:s

volumes:

wordpress-data:

mysql-data:

docker stack deploy -c compose.yml ltest

docker stack ps ltest

**kubernetes**

Minikube: lien telechargement:https://www.learnitguide.net/2023/04/how-to-install-minikube-on-ubuntu-2004.html#google\_vignette

**--------------Nodes-----------------:**

minikube start --nodes=3

minikube node add

**-----------------PODS----------------:**

apiVersion: v1

kind: Pod

metadata:

name: nginx2

labels:

owner: ayoub

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

kubectl describe pod nom

kubectl exec -it nginx -- /bin/bash

echo "hello" > /usr/share/nginx/html/index.html

curl https://localhost

minikube dashboard --url

kubectl create -f name

kubectl apply -f name

kubectl delete pod name

**---------labels&selectors---------------------:**

kubectl label pod nginx-pod app=dev

kubectl get pods --show-labels

kubectl get pods --selector app=dev

kubectl get pods -o wide

minikube ssh

curl @IP

**---------replicaset---------------------:**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-replicaset

labels:

app: webapp

type: front

spec:

replicas: 3

selector:

matchLabels:

app: webapp

template:

metadata:

name: nginx-pod

labels:

app: webapp

spec:

containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80

**------DaemonSet------:**

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: fluentd

spec:

selector:

matchLabels:

name: fluentd

template:

metadata:

labels:

name: fluentd

spec:

containers:

- name: fluentd

image: fluent/fluentd

**------Deployments------:**

kubectl create deployment test1 --image=nginx

kubectl scale deployment test --replicas=3

kubectl api-resources

--yaml--

apiVersion: apps/v1

kind: Deployment

metadata:

name: dep1

spec:

replicas: 7

selector:

matchLabels:

app: web

template:

metadata:

labels:

app: web

spec:

containers:

- name: web

image: nginx

ports:

- containerPort: 80

kubectl apply -f file.yaml

kubectl delete deployment test

**---------Scheduling---------------------:**

apiVersion: v1

kind: Pod

metadata:

name: nginx

labels:

env: test

spec:

containers:

- name: nginx

image: nginx

nodeSelector:

environment: dev

**2-----**

apiVersion: v1

kind: Pod

metadata:

name: with-node-affinity

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: environment

operator: In

values:

- prod

containers:

- name: nginx-container

image: nginx

**---------SERVICE---------------------:**

kubectl expose deployment dep1 --name=service-test --type=ClusterIP --port=8080

forward pod:kubectl port-forward pod/nginx2 8080:80

kubectl port-forward deployment/dep1 8090:80

kubectl port-forward service/service-test 8010:8080

minikube service service-test

-----service clusterIP----------------:

apiVersion: v1

kind: Service

metadata:

name: service-python

spec:

selector:

run: pod-python

type: ClusterIP

ports:

- port: 3000

protocol: TCP

targetPort: 443

-----service clusterIP----------------:

apiVersion: v1

kind: Service

metadata:

name: service-python

spec:

selector:

run: pod-python

type: ClusterIP

ports:

- port: 3000

protocol: TCP

targetPort: 443

-----service loadbalancer----------------:

apiVersion: v1

kind: Service

metadata:

name: service-python

spec:

selector:

run: pod-python

type: LoadBalancer

ports:

- port: 3000

protocol: TCP

TargetPort: 443

nodePort : 30080

kubectl get svc

**----------------NOT WORK-----------------**

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress

spec:

replicas: 2

selector:

matchLabels:

app: wordpress

template:

metadata:

labels:

app: wordpress

spec:

containers:

- name: wordpress

image: wordpress

ports:

- containerPort: 80

env:

- name: WORDPRESS\_DB\_HOST

value: mysql

- name: WORDPRESS\_DB\_NAME

value: wordpress

- name: WORDPRESS\_DB\_USER

value: wordpress

- name: WORDPRESS\_DB\_PASSWORD

value: wordpress

volumeMounts:

- name: wordpress-data

mountPath: /var/www/html

volumes:

- name: wordpress-data

emptyDir: {}

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mariadb

env:

- name: MYSQL\_ROOT\_PASSWORD

value: wordpress

- name: MYSQL\_DATABASE

value: wordpress

- name: MYSQL\_USER

value: wordpress

- name: MYSQL\_PASSWORD

value: wordpress

volumeMounts:

- name: mysql-data

mountPath: /var/lib/mysql

volumes:

- name: mysql-data

emptyDir: {}

---

apiVersion: v1

kind: Service

metadata:

name: wordpress

spec:

selector:

app: wordpress

ports:

- protocol: TCP

port: 80

targetPort: 80

type: NodePort

---

apiVersion: v1

kind: Service

metadata:

name: mysql

spec:

selector:

app: mysql

ports:

- protocol: TCP

port: 3306

targetPort: 3306

type: ClusterIP

--------2--------

-----------------

kubectl expose deployment web-dep --name=web-service --type=ClusterIP --port=8080 --target-port=8080

# kubectl get all == 5tar smit pod == pod name

kubectl port-forward pod\_name 8080:80

------------------------

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx-app

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

minikube service nginx-service

------------------------

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-app-deploy

spec:

replicas: 3

selector:

matchLabels:

app: nginx-app

template:

metadata:

labels:

app: nginx-app

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

------------

NOT WORK (BD ERROR)

------------

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress-mysql

spec:

replicas: 3

selector:

matchLabels:

app: wordpress-mysql

template:

metadata:

labels:

app: wordpress-mysql

spec:

containers:

- name: wordpress

image: wordpress:latest

ports:

- containerPort: 80

env:

- name: WORDPRESS\_DB\_HOST

value: mysql

- name: WORDPRESS\_DB\_NAME

value: wordpress

- name: WORDPRESS\_DB\_USER

value: wordpress

- name: WORDPRESS\_DB\_PASSWORD

value: wordpress

- name: mysql

image: mariadb

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

value: wordpress

- name: MYSQL\_USER

value: wordpress

- name: MYSQL\_DATABASE

value: wordpress

- name: MYSQL\_PASSWORD

value: wordpress