Docker + Kubernetes 基礎入門

Docker Hands on

Docker installation

Windows and MacOS

https://docs.docker.com/engine/install/

Install Docker Engine • docs.docker.com

Ubuntu

https://docs.docker.com/engine/install/ubuntu/

Install Docker Engine on Ubuntu • docs.docker.com

```
su
apt-get update
apt-get -y install apt-transport-https ca-certificates curl gnupg lsb-release
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
 echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update && apt-get install docker-ce docker-ce-cli containerd.io
exit

#Add current user to docker group (using docker without sudo)
```

```
sudo usermod -aG docker $USER
```

https://labs.play-with-docker.com/

Run Hello world Container

```
docker run --rm hello-world
```

Search Images

```
docker search httpd
```

Pull Image

```
docker pull httpd:alpine
```

List images

```
docker images
```

Web Server + Mapping Port

```
docker run -d -p 80:80 --name httpd1 httpd:alpine
```

List containers

```
docker ps
```

Web Server + Mount Volume

```
echo "Hello Docker!" > index.html
docker run -d --name httpd2 -v $PWD/index.html:/usr/local/apac
he2/htdocs/index.html -p 81:80 httpd:alpine
docker ps
```

Fetch the logs of a container

```
docker logs httpd1
docker logs -f httpd2
```

Run a command in a running container

```
docker exec -it httpd2 sh
# 進入sh後再執行
cat /usr/local/apache2/htdocs/index.html
exit
```

Return low-level information on Docker objects

```
docker inspect httpd2
```

Stop one or more running containers

```
docker stop httpd2
docker ps -a
```

Start containers

```
docker start httpd2
docker ps -a
```

Kill one or more running containers

```
docker kill httpd1
docker ps -a
```

Remove one or more containers

```
docker run hello-world
docker ps -a
docker rm $( docker ps -a | grep hello-world | awk {'print
$1'})
docker ps -a
```

Remove images

```
docker rmi httpd:alpine
docker ps -a
docker rm -f httpd2
docker rmi -f httpd:alpine
docker images
docker rm -f httpd1
docker rmi hello-world
docker images
```

Build image from Dockerfile

Dockerfile

```
cat << EOF > Dockerfile
FROM httpd:alpine
LABEL maintainer="Docker Maintainers <test@test.com>"

ENV IMAGE_VERSION   1.0.0
ENV RELEASE   1
RUN set -x \
    apk add vim
COPY index.html /usr/local/apache2/htdocs/index.html
EOF
```

```
docker build . -t myhttpd:v1
```

Check my image

```
docker run -d --name myhttpd -p 80:80 myhttpd:v1
docker exec -it myhttpd env
```

Kubernetes Installation

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Installing kubeadm • kubernetes.io

Installing kubelet and kubeadm on your hosts

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Check kubeadm version

```
kubeadm version
```

Turn of swap

Turn of swap checking

```
swapoff -a
sed -e '/swap/ s/^#*/#/' -i /etc/fstab
free -m
```

> *修改 /etc/fstab 檔，注釋掉 SWAP 的自動掛載，使用free -m確認swap已經關閉。*

https://www.katacoda.com/courses/kubernetes/getting-started-with-kubeadm

Getting Started With Kubeadm | Kubernetes | Katacoda • www.katacoda.com

Initializing your master

```
 kubeadm init --pod-network-cidr 10.5.0.0/16
```

Config kubectl

(please check your current account)

```
mkdir -p $HOME/.kube/

sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Enabling shell autocompletion (skipped in playground)

(please check your current account)

```
sudo apt-get install -y bash-completion

echo "source /etc/bash_completion" >> ~/.bashrc

echo "source <(kubectl completion bash)" >> ~/.bashrc

source <(kubectl completion bash)
```

Enable scheduling pods on the master (optional)

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

Installing a pod network

kuberouter

```
kubectl apply -f https://raw.githubusercontent.com/cloudnative
labs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
```

flannel

```
kubectl apply --namespace kube-system -f https://raw.githubuse
rcontent.com/coreos/flannel/master/Documentation/kube-flannel.
yml
```

weave

```
kubectl apply -n kube-system -f "https://cloud.weave.works/k8
s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49

Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network (Updated: August 2020) • it...

Joining your nodes (on Worker)

Please follow Step 1, 2 and 3 then join to master.

```
kubeadm join --token <token> <master-ip>:<master-port>
```

List token (master node)

```
kubeadm token list
```

Check cluster information

```
kubectl cluster-info
```

Get nodes information

```
kubectl get nodes
```

Kubernetes Basic Operation

List Node

```
kubectl get nodes
```

Node description

```
kubectl describe node node01
```

Create Namespaces

```
kubectl create namespace my-namespace
```

List Namespaces

```
kubectl get namespaces
```

Delete Namespaces

```
kubectl delete namespace my-namespace
```

Create an nginx deployment

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/
website/master/content/en/examples/application/nginx-app.yaml
```

Pods list

```
kubectl get pods

kubectl get po

kubectl get pods --all-namespaces
```

Service list

```
kubectl get services

kubectl get svc

kubectl get services --all-namespaces
```

List all information in all namespaces

```
kubectl get all --all-namespaces
```

Guestbook Example

https://www.katacoda.com/courses/kubernetes/playground

Kubernetes Playground | Katacoda • www.katacoda.com

redis-master-deployment.yaml

```
apiVersion: apps/v1--> API 版本

kind: Deployment -->種類

metadata: -->識別名稱等

  name: redis-master -->Deployment的名稱

spec:--> 規格
```

```
  replicas: 1 --> 建立份數
  template: -->樣版描述
    metadata:
      labels: -->定義標籤
        app: redis -->標籤定義格式為key: value
        role: master
        tier: backend
    spec:-->樣版規格
      containers:-->容器定義
      - name: master-->名稱
        image: gcr.io/google_containers/redis:e2e  -->docker i
mage
        resources: -->運行環境定義
          requests: -->最低運行要求
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
```

redis-master-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: redis-master
  labels:
    app: redis
    role: master
    tier: backend
spec:
  ports:
    # the port that this service should serve on
  - port: 6379
    targetPort: 6379 -->對應Pod中的Container port，與port相同可省
略
  selector: -->將traffic導到那些Pod
    app: redis
    role: master
    tier: backend
```

Create a Service

```
kubectl create -f https://raw.githubusercontent.com/macchiang/
guestbook/master/redis-master-service.yaml
#Check Service
kubectl get services
```

Create a Deployment

```
kubectl create -f https://raw.githubusercontent.com/macchiang/
guestbook/master/redis-master-deployment.yaml
kubectl get deployments
kubectl get pods
```

Check pods information

```
POD=$(kubectl get pods | grep redis-master | awk {'print $1'})
kubectl describe pods $POD
```

View the container logs for a given pod

```
kubectl logs $POD
```

all-in-one/redis-slave.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: redis-slave
  labels:
    app: redis
    role: slave
    tier: backend
spec:
  ports:
    # the port that this service should serve on
  - port: 6379
  selector:
    app: redis
    role: slave
    tier: backend
--- #YAML分隔符號
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-slave
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: redis
        role: slave
        tier: backend
    spec:
      containers:
      - name: slave
        image: gcr.io/google_samples/gb-redisslave:v1
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:-->定義環境變數
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns serv
ice, then to
          # instead access an environment variable to find the
master
          # service's host, comment out the 'value: dns' line
above, and
          # uncomment the line below.
          # value: env
        ports:
        - containerPort: 6379
```

Start up the redis slave

```
kubectl create -f https://raw.githubusercontent.com/macchiang/
guestbook/master/all-in-one/redis-slave.yaml
kubectl get services
kubectl get deployments
kubectl get pods
```

all-in-one/frontend.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: NodePort
  ports:
    # the port that this service should serve on
  - port: 80
  selector:
    app: guestbook
    tier: frontend
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google-samples/gb-frontend:v4
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

Create frontend

```
kubectl create -f https://raw.githubusercontent.com/macchiang/
guestbook/master/all-in-one/frontend.yaml
kubectl get services
kubectl get deployments
kubectl get pods -L tier
```

Scaling out/in frontend

```
kubectl scale deployment frontend --replicas=5
kubectl get pod
kubectl scale deployment frontend --replicas=2
kubectl get pod
```

Open Browser and try it.

```
kubectl get svc
NAME           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AG
E
frontend       10.106.196.56   <nodes>       80:32075/TCP   4m
kubernetes     10.96.0.1       <none>        443/TCP        18
h
redis-master   10.106.42.126   <none>        6379/TCP       15
m
```

```
redis-slave    10.98.143.161   <none>        6379/TCP       11
m
```

Clean up

```
kubectl delete deployments,services -l "app in (redis, guestbo
ok)"
kubectl get all
```