Cheatsheet Docker

Completa - Autor: Sergi García Barea

Docker Run

```
docker run -it --name = CONT1 ubuntu / bin / bash
```

- Crea un contenidor amb la imatge "ubuntu" (al no especificar, pren versió "latest"), li estableix un nom "CONT1" i llança en mode interactiu 01:00 shell "bash".

```
docker run -d -p 1200: 80 nginx
```

- Crea un contenidor amb la versió "latest" de la imatge "nginx" i el llança a "background", exposant el port 80 del contenidor al port 1200 de la màquina amfitrió.

```
docker run -it -i MISSATGE = HOLA ubuntu: 14.04 bash
```

- Crea un contenidor amb la imatge "ubuntu", versió "14.04" i estableix la variable d'entorn "MISSATGE".

Docker ps

```
docker ps
```

- Mostra informació dels contenidors en execució.

```
docker ps -a
```

- Mostra informació de tots els contenidors, tant aturats com en execució.

Docker Start / Stop / Restart

```
docker start micontenedor
```

- Arrenca el contenidor amb nom "el meu contenidor".

```
docker start -ai micontenedor
```

- Arrenca el contenidor amb nom "el meu contenidor", enllaçant el comandament executat a l'arrencada a l'entrada i eixida del terminal de l'amfitrió.

Docker Exec

```
docker exec -it -i FITXER = prova cont bash
```

- Llança al contenidor "cont" (que ha d'estar arrencat) el comandament "bash", establint la variable d'entorn "FITXER" i enllaçant l'execució de forma interactiva a l'entrada i sortida estàndard de l'amfitrió.

```
docker exec -d cont touch / tmp / prova
```

- Llança al contenidor "cont" (que ha d'estar arrencat) el comandament "touch / tmp / prova". Aquesta comanda s'executa en segon pla, generant el fitxer "/ tmp / prova".

Cheatsheet Docker

Completa - Autor: Sergi García Barea

Docker attach

```
docker attach idcontainer
```

- Enllaça la nostra terminal l'entrada / sortida de la nostra al procés en segon pla del contenidor "idcontainer".

Docker logs

```
docker logs -n 10 idcontainer
```

- Mostra les 10 últimes línies de la sortida estàndard produïda pel procés en execució en el contenidor.

Docker cp

```
docker cp idcontainer: / tmp / prova ./
```

- Copia el fitxer "/ tmp / prova" del contenidor "idcontainer" a directori actual de l'amfitrió.

```
docker cp ./miFichero idcontainer: / tmp
```

- Copia el fitxer "miFichero" de directori actual de l'amfitrió a la carpeta "/ tmp" del contenidor.

Gestió d'imatges

```
docker images
```

- Informació d'imatges locals disponibles.

```
docker search ubuntu
```

- Cerca la imatge "ubuntu" al repositori remot (per defecte Docker Hub).

```
docker pull alpine
```

- Descàrrega localment imatge "alpine".

```
docker history alpine
```

- Mostra la història de creació de la imatge "alpine".

```
docker rmi ubuntu: 14.04
```

- Elimina localment la imatge "ubuntu" amb tag "14.04".

```
docker rmi $ (docker images -q)
```

- Esborra tota imatge local que no estigui sent usada per un contenidor.

```
docker rm IDCONTENEDOR
```

- Esborra un contenidor amb IDCONTENEDOR.

```
docker stop $ (docker ps -a -q)
```

Cheatsheet Docker

Completa - Autor: Sergi García Barea

- Per a tots els contenidors del sistema.

```
docker rm $ docker ps -a -q)
```

- Esborra tots els contenidors aturats del sistema.

```
docker system prune -a
```

- Esborra totes les imatges i contenidors aturats del sistema.

Creació d'imatges a partir de contenidors

```
docker commit -m "comentari" IDCONTENEDOR usuari / imatge: versió
```

- Fa commit d'un contenidor existent a una imatge local.

```
docker save -o copiaSeguridad.tar imagenA
```

- Guarda una còpia de seguretat d'una imatge en fitxer ".tar".

```
docker load -i copiaSeguridad.tar
```

- Restaura una còpia de seguretat d'una imatge en fitxer ".tar".

Docker Hub

```
docker login
```

- Permet introduir credencials del registre (per defecte "Docker Hub").

```
docker push usuari / imatge: versió
```

- Permet pujar al repositori una imatge mitjançant "push".

Exemple de Dockerfile

```
FROM alpine
LABEL maintainer = "email@gmail.com"
#Actualizamos i instal·lem paquets amb APK per Alpine
RUN Apk update && Apk add apache2 php php-apache2 openrc tar
#Copiamos script per llançar Apache 2
ADD ./start.sh /start.sh
#Descargamos un exemple de <? php phpinfo (); ?> Per ensenyar com baixar una mica d'Internet
# Podria haver estat simplement
#RUN echo "<? Php phpinfo ();?>"> /Var/www/localhost/htdocs/index.php
ADD https: //gist.githubusercontent. com / SyntaxC4 / 5.648.247 / raw / 94277156638f9c309f2e36e19bff378ba7364907 /
info.php /var/www/localhost/htdocs/index.php
# Si volguéssim alguna cosa com Wordpress faríem
#ADD http://wordpress.org/latest.tar.gz / var / www / localhost / htdocs / wordpress.tar.gz
#RUN tar xvzf /var/www/localhost/htdocs/wordpress.tar.gz && rm -rf /var/www/localhost/htdocs/wordpress.tar.gz

# Fem servir usuari i grup www-data. El grup el crea Apache, però si volguéssim crear grup
# Grup www-data RUN setembre -x && addgroup -g 82-S www-data
# Creem usuari www-data i l'afegim a aquest grup
```

Cheatsheet Docker

Completa - Autor: Sergi García Barea

```
RUN adduser -o 82 -D - s -G www-data www-data
# Fem tots els fitxers de / var / www propietat de www-data
# I donem permisos s aquests fitxers ja start.sh
RUN chown -R www-data: www-data / var / www / && chmod -R 775 / var / www / && chmod 755 /start.sh
#Indicamos port a exposar (per altres contenidors) 80
Exposé 80
#Comando llançat per defecte a l'instal·lar el contenidor
CMD /start.sh
```

- Exemple de fitxer "Dockerfile".

Gestió de xarxes

```
docker network create redtest
```

- Creem la xarxa "redtest"

```
docker network ls
```

- Ens permet veure el llistat de xarxes existents.

```
docker network rm redtest
```

- Esborrem la xarxa "redtest".

```
docker run -it --network redtest ubuntu / bin / bash
```

- Connectem el contenidor que vam crear a la xarxa "redtest".

```
docker network connect IDRED IDCONTENEDOR
```

- Connectem un contenidor a una xarxa.

```
docker network disconnect IDRED IDCONTENEDOR
```

- Desconnectem un contenidor d'una xarxa

Volums

```
docker run -d -it --name appcontainer -v / home / sergi / target: / app nginx: latest
```

- Creem un contenidor i assignem un volum amb "binding mount".

```
docker run -d -it --name appcontainer -v micontenedor: / app nginx: latest
```

- Creem un contenidor i assignem un volum Docker anomenat"micontenedor".

```
docker volume create / ls / rm mivolumen
```

- Permet crear, llistar o eliminar volums Docker.

```
docker run -d -it --tmpfs / app nginx
```

- Permet crear un contenidor i associar un volum "tmpfs".

```
docker run --rm --volumes-from contenedor1 -v / home / sergi / backup: / backup ubuntu bash -c
```

Cheatsheet Docker

Completa - Autor: Sergi García Barea

```
"cd / dades && tar cvf /backup/copiaseguridad.tar."
```

- Permet fer una còpia de seguretat d'un volum associat a"contenedor1" i que es munta a"/dades". Aquesta còpia finalment acabarà a "/home/sergi/backup"de la màquina amfitrió.

Exemple bàsic de fitxer "docker-compose.yml"

```
versió: "3.9"
services:
  db:
        image: mysql:5.7
        volumes:
        - db_data: / var / lib / mysql
        environment:
            MYSQL_ROOT_PASSWORD: somewordpress
            MYSQL_DATABASE: wordpress
            MYSQL_USER: wordpress
            MYSQL_PASSWORD: wordpress
  wordpress:
        image: wordpress: latest
        ports:
        - "8000: 80"
        environment:
            WORDPRESS_DB_HOST: db:3306
            WORDPRESS_DB_USER: wordpress
            WORDPRESS_DB_PASSWORD: wordpress
            WORDPRESS_DB_NAME: wordpress
volumes:
  db_data:
```

Principals ordres de "Docker Compose"

```
docker-compose up -d
```

- Inicia el sistema definit en***"docker-compose.yml"*** en segon pla. Genera i descàrrega imatges requerides.

```
docker-compose down
```

- Atura i elimina els contenidors segons la configuració de"docker-compose.yml".

```
docker-compose build / pull
```

- Construeix / descarrega les imatges de contenidors segons la configuració de"docker-compose.yml".

```
docker-compose ps
```

- Mostra informació dels contenidors segons la configuració de"docker-compose.yml".

```
docker-compose up -d --scale web = 3
```

- Similar a***"docker-compose up*** -d" només que a més, el servei definit com a "web" en el fitxer***"docker-compose.yml"*** l'escala creant 3 còpies i realitzant balanceig automàtic si es realitza una petició al host anomenat com el servei "web".

Cheatsheet Docker

Completa - Autor: Sergi García Barea

Principals ordres de "Kubernetes"

```
kubectl apply -f "fichero.yaml"
```

- Aplica en Kubernetes la configuració especificada a "fichero.yaml".

```
kubectl create deployment midespliegue --image = sergarb1 / flaskparakubernetes --port = 5000
```

- Crea un desplegament basat en una imatge donada i en el port 5000.

```
kubectl Expose deployment midespliegue --type = LoadBalancer --name = midespliegue-http
```

- Crea un servei de tipus "LoadBalancer" exponeniendo "midespliegue".

```
kubectl get pods; kubectl get services; kubectl get deployments
```

- Mostra informació de pods, serveis o desplegaments.

```
kubectl scale deployment midespliegue --replicas = 3
```

- Escala horitzontalment un desplegament a 3 rèpliques.

```
kubectl autoscale deployment midespliegue --min = 5 --max = 10
```

- Configura autoescalat horitzontal, acceptant entre 5 i 10 rèpliques.

```
kubectl delete pod / deployment / service / autoscale nom
```

- Permet eliminar un pod, desplegament, servei o autoescalat.

Principals ordres de "MniKube"

```
minikube start
```

- Inicia la màquina virtual que conté MiniKube i posa el clúster Kubernetes en marxa

```
minikube service miservicio
```

- Ens permet accedir a un servei dins de MiniKube des de la màquina local.

```
minikube tunnel
```

- Mentre estigui en execució, exposa un servei dins de MiniKube a la màquina local

Exemple de fitxer YAML desplegament / servei / persistència amb Kubernetes

```yaml
#Definimos la informació de l'servei
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
        app: wordpress
spec:
  ports:
```

Completa - Autor: Sergi García Barea

```
        #El servei s'exposa al port 80
        - port: 80
  selector:
        app: wordpress
        tier: frontend
  #Aplicamos balanceig de càrrega per facilitar la seva escalat horitzontal
  type: LoadBalancer
---
#Definimos un volum persistent
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
        app: wordpress
spec:
  #Indica que només pot ser muntat per a lectura / escriptura per un node. Per a la resta lectura.
  #En aquest cas, s'usa per modificar un fitxer de configuració.
  accessModes:
        - ReadWriteOnce
  resources:
        requests:
        storage: 20Gi
---
#definimos el desplegament
apiVersion: apps / v1
kind: Deployment
metadata:
  name: wordpress
  labels:
        app: wordpress
spec:
  selector:
        matchLabels:
        app: wordpress
        tier: frontend
  strategy:
        type: Recreate
  template:
        metadata:
        labels:
        app: wordpress
        tier: frontend
        spec:
        #Imagen
        contenidors:
        - image: wordpress: 4.8-apatxe
        name: wordpress
        #Indicamos variables d'entorn
        env:
        - name: WORDPRESS_DB_HOST
        value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
        value: CEFIREdocker
        ports:
        - containerPort: 80
        name: wordpress
```

Completa - Autor: Sergi García Barea

```
        volumeMounts:
        - name: wordpress-persistent-storage
        mountPath: / var / www / html
        volumes:
        - name: wordpress-persistent-storage
        persistentVolumeClaim:
        claimName: wp-pv-claim
```