

# Docker Image 相關

## 鏡像的獲取

- pull from `registry` (online) 從registry拉取
  - public (公有)
  - private (私有)
- build from `Dockerfile` (online) 從Dockerfile構建
- load from `file` (offline) 文件導入 (離線)

官方image:

[https://hub.docker.com/search?type=image&image\\_filter=official](https://hub.docker.com/search?type=image&image_filter=official)

常用 public image server

- [Docker Hub Container Image Library | App Containerization](#)
- [Quay Container Registry · Quay](#)

### \*建立映像檔

```
# 於當前目錄，按Dockerfile.test中的指令，建立test:v1 ({name:tag})的映像檔
docker build . -f Dockerfile.test -t test:v1
```

### \*拉取與推送映像檔

```
# 使用docker hub 下載image ({name:tag})
docker pull node:alpine
# 使用quay.io 下載image
docker pull quay.io/bitnami/tomcat
# node:alpine 重新命名標籤 username/node:alpine
docker tag node:alpine username/node:alpine
# 推送至私人倉庫
docker push username/node:alpine
```

## 使用私有registry

```
# 修改鏡像registry
sudo vim /etc/docker/daemon.json
#####
# 可與http鏡像 "insecure-registries" : ["10.60.78.79"],
# 鏡像站 "registry-mirrors": ["http://hub-mirror.c.163.com"]

{
  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
#####
# 使文件生效
sudo systemctl daemon-reload
# 重啟docker
sudo service docker restart
```

## 使用proxy 抓取鏡像檔

\*ubuntu

```
# 編輯設定檔
vim /etc/default/docker

##檔案內容
export http_proxy="http://192.168.33.10:3128/"
export https_proxy="http://192.168.33.10:3128/"
##
```

```
# proxy 如果有帳密驗證
export http_proxy="https://username:password@192.168.33.10:3128/"

# 重啟docker
systemctl restart docker
```

\*centos

```
vim /etc/systemd/system/docker.service.d/http-proxy.conf

#####

[Service]
Environment="HTTP_PROXY=http://192.168.1.1:3128/"
Environment="HTTPS_PROXY=http://192.168.1.1:3128/"
#如果要排除不走
Environment="NO_PROXY=localhost,127.0.0.0/8,docker-registry.somecorporation.com"

#####

#重讀設定
systemctl daemon-reload

#重啟docker
systemctl restart docker
```

## 查看image

```
# 取得id
[root@localhost ~]# docker image ls

REPOSITORY TAG          IMAGE ID      CREATED      SIZE
tomcat      latest          9f35737a8466 30 hours ago 680MB
busybox     latest          beae173ccac6 3 weeks ago  1.24MB
openjdk     8-jre-alpine   f7a292bbb70c 2 years ago  84.9MB
```

```
[root@localhost ~]# docker image inspect f7a292bbb70c
[
  {
    "Id": "sha256:f7a292bbb70c4ce57f7704cc03eb09e299de9da19013b084f138154421918cb4",
    "RepoTags": [
      "openjdk:8-jre-alpine"
    ],
    "RepoDigests": [
      "openjdk@sha256:f362b165b870ef129cbe730
.....

# 重要訊息
"Id": "sha256:f7a292bbb70c4ce57f7704cc03eb09e299de9da19013b084f138154421918cb4",
"RepoTags": [
  "openjdk:8-jre-alpine"
],

"Architecture": "amd64",
"Os": "linux",
```

## 刪除 image

```
# 查詢要刪除image id
[root@localhost ~]# docker image ls
REPOSITORY TAG          IMAGE ID      CREATED      SIZE
tomcat      latest          9f35737a8466 30 hours ago 680MB
busybox     latest          beae173ccac6 3 weeks ago  1.24MB
```

```

openjdk      8-jre-alpine  f7a292bbb70c  2 years ago  84.9MB
# 刪除 image
[root@localhost ~]# docker image rm beae173ccac6
Untagged: busybox:latest
Untagged: busybox@sha256:5acba83a746c7608ed544dc1533b87c737a0b0fb730301639a0179f9344b1678
Deleted: sha256:beae173ccac6ad749f76713cf4440fe3d21d1043fe616dfbe30775815d1d0f6a
Deleted: sha256:01fd6df81c8ec7dd24bbbd72342671f41813f992999a3471b9d9cbc44ad88374

# 刪除名字跟nginx 有關的image (-f 強制)
[root@localhost ~]# docker images | grep nginx | awk '{print $3}' | xargs docker rmi
[root@localhost ~]# docker images | grep nginx | awk '{print $3}' | xargs docker rmi -f

# 刪除沒有名字的image (-f 強制)
[root@localhost ~]# docker images | grep "<none>" | awk '{print $3}' | xargs docker rmi
[root@localhost ~]# docker images | grep "<none>" | awk '{print $3}' | xargs docker rmi -f

```

## image 匯入匯出

```

# 查看 image
[root@localhost ~]# docker image ls
REPOSITORY TAG          IMAGE ID      CREATED      SIZE
tomcat     latest      9f35737a8466  30 hours ago 680MB
busybox    latest      beae173ccac6  3 weeks ago  1.24MB
openjdk    8-jre-alpine f7a292bbb70c  2 years ago  84.9MB

# 匯出 image
# docker image save {name}:{tag} -o {匯出檔名}
[root@localhost ~]# docker image save openjdk:8-jre-alpine -o openjdk:8-jre-alpine.image
# 查看是否匯出
[root@localhost ~]# ls openjdk:8-jre-alpine.image
openjdk:8-jre-alpine.image

```

```

# image 匯入
# docker image load -i {image file}
[root@localhost ~]# docker image load -i ./openjdk:8-jre-alpine.image
f1b5933fe4b5: Loading layer [=====>] 5.796MB/5.796MB
9b9b7f3d56a0: Loading layer [=====>] 3.584kB/3.584kB
edd61588d126: Loading layer [=====>]
80.28MB/80.28MB
Loaded image: openjdk:8-jre-alpine

```

## push image to docker hub

```

# 建立上傳image
# docker image tag nginx {docker hub id}/{name}:{tag}
docker image tag nginx treemanou/my_nginx:v2
# docker image push {docker hub id}/{name}:{tag}
docker image push treemanou/my_nginx:v2

```

## commit 建立 image

```

# 找一個container
$ docker ps -a
CONTAINER ID  IMAGE    COMMAND    CREATED      STATUS      PORTS      NAMES
e09bb406dcef  busybox  "sh"       53 seconds ago Exited (0) 18 seconds ago      hardcore_bartik
# 利用此container建立 image
# docker commit {container id} {name}:{tag}
$ docker commit e09 treemanou/my_busybox:v3
sha256:42bc77669110af56336ec10afdbd7f92fa371f02eccbda1e0bbb1bbc00f3cf6

```

## Docker File

# Docker File 官方說明

[Dockerfile reference](#) | [Docker Documentation](#)

## \*ENV 環境變數

```
ENV ANT_HOME /opt/ant
```

## \*ARG傳入參數

```
FROM alpine
ARG NODE_ARG
ENV NODE_ENV="${NODE_ARG:-development}"
RUN echo "ARG=${NODE_ARG}, ENV=${NODE_ENV}"
CMD echo "ARG=${NODE_ARG}, ENV=${NODE_ENV}"
```

```
# 傳入NODE_ARG
# DOCKER_BUILDKIT=0
DOCKER_BUILDKIT=0 docker build --build-arg NODE_ARG=staging .
```

## \*工作目錄 WORKDIR

```
# / 下
RUN echo "say hi" > hi.txt
WORKDIR /app
# /app 下
RUN echo "hello" > world.txt
```

## \*指定用戶USER (不指定舊識)

```
# 新增 group與user
RUN groupadd -r redis && useradd -r -g redis redis
# 指定當前user為redis
USER redis

# 以redis的身份執行redis-server
RUN ["redis-server"]
```

## \*Health Check

```
FROM nginx
RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/*
HEALTHCHECK --interval=5s --timeout=3s \
CMD curl -fs http://localhost/ || exit 1
```

```
FROM registry.test.com.tw/test/centos:openjdk8_tomcat8.5

ENV CATALINA_HOME /opt/tomcat
ENV SERVICE_HOME /usr/AP/fubon
ENV ANT_HOME /opt/ant
ENV JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk
ENV PATH="${PATH}:${ANT_HOME}/bin:${CATALINA_HOME}/bin"
ENV AWS_HOME="/root/aws"

RUN wget https://archive.apache.org/dist/ant/binaries/apache-ant-1.10.7-bin.tar.gz \
&& tar -zxvf apache-ant-1.10.7-bin.tar.gz \
&& rm apache-ant-1.10.7-bin.tar.gz \
&& mv apache-ant* ${ANT_HOME} \
&& yum install -y python3 \
&& pip3 install awscli --upgrade --user \
&& ln -s /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7_8.x86_64/ /usr/lib/jvm/java-1.8.0-openjdk.x86_64 \
&& yum install -y httpd php php-gd
```

```
COPY server.xml ${CATALINA_HOME}/conf/
```

```
COPY my_entrpoint.sh /my_entrpoint.sh  
RUN chmod -v +x /my_entrpoint.sh
```

```
COPY --chown=user:group file* /app/
```

```
COPY server_builder.sh ${SERVICE_HOME}/
```

```
#Set Alias
```

```
RUN echo "alias cdc='cd /usr/AP/test/cm'" >> ~/.bashrc \
```

```
&& echo "alias restartTomcat='cd /opt/tomcat/bin; ./shutdown.sh; /bin/sleep 3; ps -ef | grep \"org.apache.catalina.startup.Bootstrap  
start\" | grep -v grep | awk '{print $2}' | xargs kill; ./startup.sh'" >> ~/.bashrc \
```

```
&& echo "alias cmant='cd /usr/AP/test/cm/src; ant; restartTomcat'" >> ~/.bashrc \
```

```
EXPOSE 8080
```

```
CMD ["-D","FOREGROUND"]
```

```
# 預設目錄
```

```
WORKDIR /usr/AP/test/
```

```
# 一定會執行
```

```
ENTRYPOINT ["/my_entrpoint.sh"]
```

\* 複寫CMD

```
docker run --rm apache:v1 -v
```

## 多階段建置

```
# 建置環境
```

```
FROM alpine as build
```

```
RUN echo "hello" > mytest
```

```
# 執行環境
```

```
FROM alpine
```

```
COPY --from=build /mytest .
```

```
RUN cat /mytest
```

**Distroless Dock image** => 精簡image

# 常用image

## busybox

- 提供輕量簡單的shell 的 linux 容器

## containous/whoami

- 提供返回ip hostname 服務

```
vagrant@swarm-manager:~$ curl 192.168.200.10:8080  
Hostname: fdf7c1354507  
IP: 127.0.0.1  
IP: 10.0.0.7  
IP: 172.18.0.3  
IP: 10.0.1.14  
RemoteAddr: 10.0.0.2:36828  
GET / HTTP/1.1  
Host: 192.168.200.10:8080  
User-Agent: curl/7.68.0  
Accept: */*
```

⌚修訂版本 #25

★由 treeman 建立於 14 @ 2022 22:41:57

✎由 treeman 更新於 9 | 2024 15:05:20