Docker

Main Conception

Image 镜像，可看作一个未运行的程序

Container 容器，一个容器可看作一个正在运行的程序

Repository 仓库，保存镜像的地方（官方：https://hub.docker.com/）

Image

Consist of **repository name** and **tag**, split by "**:**"

`docker images` 查看镜像

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
obsismc/ubuntu	v0	15cf04d8635b	5 months ago	1G
ubuntu	latest	e4046c038836	About an hour ago	3G

> The fist image is `obsismc/ubuntu:v0`

💡 **Repository name** is made of **user name** and the **image name** !

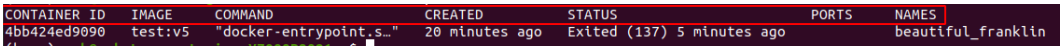
Container

进入容器终端后，用 **exit** 或 **Ctrl + D** 退出终端

Common Operation

Query

- `docker images`：查看镜像
- `docker ps` ：查看正在运行的容器
  - `-a` ：查看所有容器，包括停止的



💡 容器状态(status)有:
• created（已创建）
• exited（停止）
• removing（迁移中）

RUN

- `docker run [args] [Image ID or name] [command]`: **Start a image**
  - `Image ID or name`: local or remote image to use (if not in local, fetch from remote)
  - `command`: command used after opening a image

Args	Meaning	Example
`-it`	交互式终端运行，一般和 `bash` 命令一起用	`docker run -it ubuntu:lastest bash`
`-d`	容器启动后后台运行	
`-p [Host Port]:[Container Port]`	将容器的某个端口映射到主机的某个端口	
`-v [Host Path]:[Container Path]`	目录挂载，详细见后面	
`--name [container name]`	给容器命名，否则容器名字是随机的	

- `docker start [exited container]` : let exited container be running
- `docker restart [container]` : restart container

Delete

Remove

- `docker rm [exited container ID or name]`: delete **exited** container(s)
- `docker rmi [Image ID or name]`: delete images(s)

Stop

- `docker stop [container ID or name]`: let a container be exited

Attach

进入一个运行的容器

- `docker attach/exec [args] [container ID or name] [command]`

- attach：进入后退出会导致该容器exit
- exec：进入后退出不会导致容器退出

> 💡 目前一般直接 `docker exec -it [container name] bash`

Mount

数据管理的方法，作用

1. 可持久化，容器退出后和数据还存在
2. 同步容器和本地的文件，主机修改后容器可以立即同步

可用 `-v` 或 `--mount` 挂载

- bind mount: 直接把宿主机目录映射到容器内，适合挂代码目录和配置文件
- volume: 在宿主机上的docker管理的目录，适合数据库数据

-v

bind mount 用绝对路径：`-v /home/:/app`

volume 没试过

> 💡 路径之间用 ":" 隔开

Dockerfile

创建自己的docker镜像：

- 写 Dockerfile 文件
- build

```
# FROM：以什么镜像为基础
# MAINTAINER：作者
FROM node:11
MAINTAINER name

# ADD：复制主机目录到镜像目录 ( ADD [host path] [image path] )
ADD . /app

# 在镜像中运行脚本，可以用来安装依赖
# 最好不要太多RUN，因为每次RUN会新建一层（不知道是啥反正降低性能）
# 用 && 执行多个指令
RUN npm install && npm run build:stage

# CMD只能有一个，但可以用 && 执行多个脚本（shell等）
# 容器启动后马上执行的命令
CMD npm run dev
```

Build

```
# 用directory里的Dockerfile生成名为image name的镜像
docker build -t [image name] [directory]
```

Docker-compose

Credential

Login

```
docker login
```

Images

`docker images`

REPOSITORY: [user]/[repo_name]

push的时候必须加上自己的账号名，否则出现无权限被拒绝的问题

> `docker tag [REPOSITORY]:[TAG] [new REPOSITORY]:[new TAG]` ： 换名字

Exception

> `denied: requested access to the resource is denied`

未登录

- `docker images`：查看镜像