Docker - Container 實務應用

連接資料庫

建立資料庫

```
docker run --rm -it --name 10.10.10.10 -e MYSQL_RANDOM_ROOT_PASSWORD=yes -e MYSQL_USER
=wunyu -e MYSQL_PASSWORD=1234 -e MYSQL_DATABASE=some percona
```

使用 container 連接資料庫

```
docker run --rm -it  percona mysql -h10.10.10.10 -uwunyu -p1234

#換成其他 fork的 client
docker run --rm -it mysql mysql -h10.10.10.10 -uwunyu -p1234
docekr run --rm -it mariadb mysql -h10.10.10.10 -uwunyu -p1234

#換成其他版本
docker run --rm -it percona:5.6 mysql -h10.10.10.10 -uwumyu -p1234
docker run --rm -it percona:5.7 mysql -h10.10.10.10 -uwumyu -p1234

#Redis
docker run --rm -it redis redis-cli -h10.10.10.10
docker run --rm -it redis:4 redis-cli -h10.10.10.10
docker run --rm -it redis:3 redis-cli -h10.10.10.10

#host安裝多個版本會有許多問題，使用具有"隔離特性"的 container 測試起來就容易許多了!
```

資料庫 Server 端

透過 pull 指令抓取 image，再藉由 container 來建立環境，等同於系統重灌，節省很多時間

可藉由 -p 指令將 port 開放出去，就跟在本機安裝 Server 是相同的道理! 方常好用!

```
docker run --rm -it --name 3306 -p 3306:3306 -e  MYSQL_ROOT_PASSWORD=pass mysql
docker run --rm -it --name 5432 -p 5432:5432 -e POSTGRES_PASSWORD=pass postgres
docker run --rm -it --name 6379 -p 6379:6379 redis
docker run --rm -it --name 11211 -p 11211:11211 memcached
```

透過docker run搭配一堆指令，達到"不安裝工具也能使用工具"的需求

```
docker run --rm -it -v $PWD:/source -w /source node:14-alpine npm -v
#--rm 執行完後刪除container
#-it 跟container互動需要的參數
#-v $PWD:/source 把host目錄綁定到container，$PWD代表執行指令的當下目錄，/source為container裡的絕對路徑
#node:14-alpine image名稱
#npm -v container內執行的指令，可以換成其他指令
```

上述步驟有點複雜，可以拆分成"進container"跟"container裡面執行指令"

將上面的合併指令拆解成下面code比較好運行

```
# 先查看目前檔案列表
ls -l

#使用 shell
docker run --rm -it -v $PWD:/source -w /source node:14-alpine sh

#進入container執行指令
npm -v

#進入做其他事情
npm init

#離開 container 看看，多了一個package.json檔案
ls -l

#上述有做bind mount，所以說container內所做的事情會回傳給host這邊，達到"不安裝工具也能使用工具"的需求!!
```

拿上面產生的package.json來進行範例

```
#確認內容
cat package.json
```

```
#安裝套件
docker run --rm -it -v $PWD:/source -w /source node:14-alpine npm install

#確認產生的檔案
```

精簡指令! 設定 alias

```
# 確認無法使用 npm
npm -v

# 設定 alias
alias npm="docker run -it --rm -v \$PWD:/source -w /source node:14-alpine npm"

# 確認可以透過docker run 使用 npm
npm -v

#alias 可以確認別名
alias
#unalias 名稱可以移除別名
unalias npm


#其他工具也可以使用，避免過多複雜指令
# 使用 Composer
alias composer="docker run -it --rm -v \$PWD:/source -w /source composer:1.10"

# 使用 npm
alias npm="docker run -it --rm -v \$PWD:/source -w /source node:14-alpine npm"

# 使用 Gradle
alias gradle="docker run -it --rm -v \$PWD:/source -w /source gradle:6.6 gradle"

# 使用 Maven
alias mvn="docker run -it --rm -v \$PWD:/source -w /source maven:3.6-alpine mvn"

# 使用 pip
alias pip="docker run -it --rm -v \$PWD:/source -w /source python:3.8-alpine pip"

# 使用 Go
alias go="docker run -it --rm -v \$PWD:/source -w /source golang:1.15-alpine go"

# 使用 Mix
alias mix="docker run -it --rm -v \$PWD:/source -w /source elixir:1.10-alpine mix"
```

測試不同版本跑單元測試

```
# 在 laravel-bridge/scratch
docker run -it --rm -v $PWD:/source -w /source php:7.1-alpine vendor/bin/phpunit
```

```
docker run -it --rm -v $PWD:/source -w /source php:7.2-alpine vendor/bin/phpunit
docker run -it --rm -v $PWD:/source -w /source php:7.3-alpine vendor/bin/phpunit
docker run -it --rm -v $PWD:/source -w /source php:7.4-alpine vendor/bin/phpunit
```