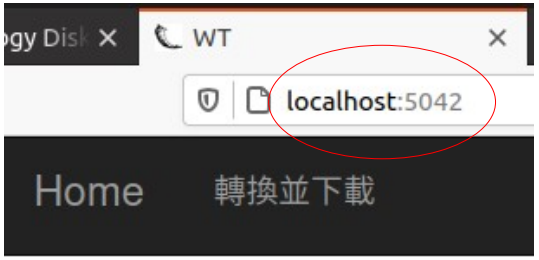
建立新的 docker image 來移植應用

- 為何需要建立新的映像檔？因為別人的image很難在不用改的情況下直接套用
  - 以smartlab/flask為例，實現一個youtube轉mp3網頁應用程式會缺少
    - 4個python第三方模組: flask_bootstrap, flask_moment, flask_wtf 和 youtube-dl
    - 2個ffmpeg轉檔程式: ffmpeg和ffprobe
- 建立image有3種方式(*Learn Docker – Fundamentals of Docker 18.x by Gabriel N. Schenker*)
  - 作法1: Create a custom image by interactively changing the container layer and committing it (實驗)
  - 作法2: Author a simple Dockerfile using keywords such as FROM, COPY, RUN, CMD and ENTRYPOINT to generate a custom image (正式規劃)
  - 作法3: Export an existing image using docker image save and import it into another Docker host using docker image load (分享)
- 透過docker來方便實現程式的移植：從個人私下用的Linux作業系統到公司的群暉作業系統

將應用程式從個人電腦移植到雲端



先在個人電腦上開發好網頁應用程式

後在公司的私有雲上跑一樣的網頁應用

遵守 docker 的標準化規範
(image 和 container)

https://registry.hub.docker.com/r/smartlab/flask/ 120%

Smartlab Flask

以別人現成的映像檔為基礎來加入自己的工具

Please, use this docker-composer to run the full stack `NGINX > uWSGI > Flask`

Run app with pure Flask

```
docker run -p 8080:5000 -d -v /path/app:/app smartlab/flask
```

← 主檔命名為 main.py

Run app with uWSGI

```
docker run -p 8080:5000 -d -v /path/app:/app smartlab/flask uwsgi
```

← 主檔命名為 uwsgi.py

Run app in debug mode

```
docker run -p 8080:5000 -d -v /path/app:/app smartlab/flask debug
```

← 主檔命名為 main.py

Run tools to test code quality

```
docker run -p 8080:5000 -d -v /path/app:/app smartlab/flask test
```

Run terminal to inspect container

```
docker run -p 8080:5000 -it -v /path/app:/app smartlab/flask terminal
```

Note: the main file must be named as `/app/main.py`

該映像檔的使用說明只有這一頁，而且還講得不夠精確。

Docker Pull Command

```
docker pull smartlab/flask
```

Owner

smartlab

Source Repository

Github
smartlab-br/flask

某網頁程式的架構和建映像檔的指令

```
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$ tree
.
├── Dockerfile
├── main.py
├── requirements.txt
├── static
│   └── favicon.ico
├── static_bin
│   ├── ffmpeg
│   └── ffprobe
├── templates
│   ├── 404.html
│   ├── 500.html
│   ├── base.html
│   └── index_41.html
├── tmp
└── uwsgi.py

4 directories, 11 files
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$
```

程式的架構 (包含要加到映像檔的部份工具)

```
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$ cat Dockerfile
FROM smartlab/flask

COPY ./requirements.txt /app/requirements.txt
COPY ./static_bin/ /usr/bin/

WORKDIR /app

RUN pip3 install -r requirements.txt
```

作法 2: 使用 Dockerfile 建映像檔

```
pydoc@pydoc: ~/python3_scripts/docker_flask_youtube2mp3
  GNU nano 4.8                          requirements.txt
#flask
flask_bootstrap
flask_moment
flask_wtf
youtube-dl
```

需要新增的 4 個 python 模組

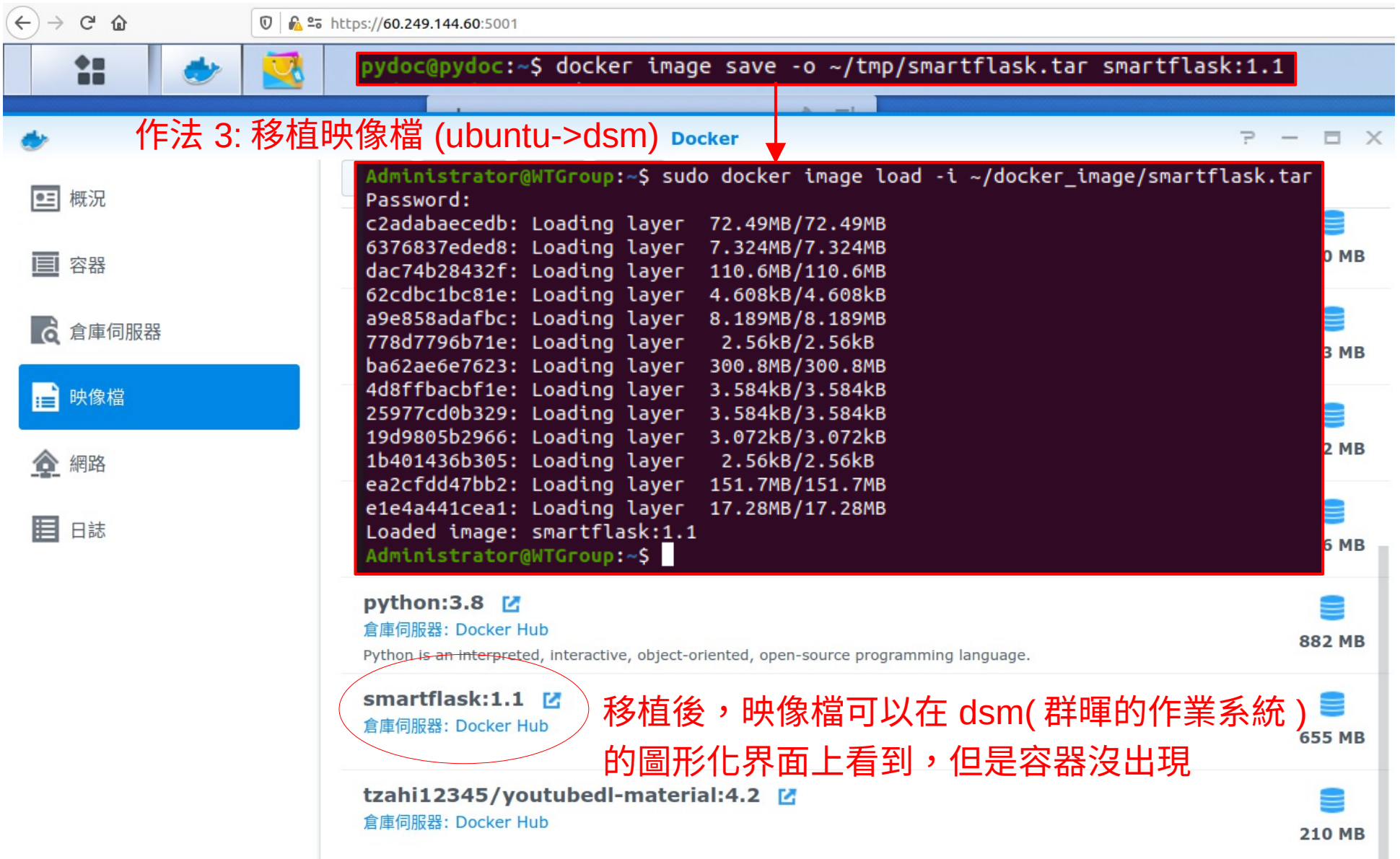
```
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$ tail uwsgi.py
        ydl.download([session.get('url')])

    #return('<h1>Done!</h1>')
    #return render_template('index_41_b.html') # 14pm
    #return send_file(url_file, as_attachment=True)
    return send_file(session.get('url_file'), as_attachment=True)

application = app
#if __name__ == "__main__":
#    app.run(host='0.0.0.0')
```

為搭配該映像檔而修改

```
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$ docker build -t smartflask:1.1 .
Sending build context to Docker daemon  151.9MB
Step 1/5 : FROM smartlab/flask
 ---> 8845242c47e6
Step 2/5 : COPY ./requirements.txt /app/requirements.txt
 ---> Using cache
 ---> 68b02afb8908
Step 3/5 : COPY ./static_bin/ /usr/bin/
 ---> 5f3bab6e4966
Step 4/5 : WORKDIR /app
 ---> Running in 02293679fab2
Removing intermediate container 02293679fab2
 ---> 572890c44935
Step 5/5 : RUN pip3 install -r requirements.txt
 ---> Running in d7b2754acbc3
Collecting flask_bootstrap
  Downloading Flask-Bootstrap-3.3.7.1.tar.gz (456 kB)
Collecting flask_moment
  Downloading Flask_Moment-0.11.0-py2.py3-none-any.whl (4.3 kB)
Collecting flask_wtf
  Downloading Flask_WTF-0.14.3-py2.py3-none-any.whl (13 kB)
Collecting youtube-dl
  Downloading youtube_dl-2021.3.2-py2.py3-none-any.whl (1.9 MB)
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.8/site-packages (from flask_bootstrap->-r requirements.txt (line 2)) (1.0.2)
Collecting dominate
  Downloading dominate-2.6.0-py2.py3-none-any.whl (29 kB)
Collecting visitor
  Downloading visitor-0.1.3.tar.gz (3.3 kB)
Requirement already satisfied: itsdangerous in /usr/local/lib/python3.8/site-packages (from flask_wtf->-r requirements.txt (line 4)) (1.1.0)
Collecting WTForms
  Downloading WTForms-2.3.3-py2.py3-none-any.whl (169 kB)
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.8/site-packages (from Flask>=0.8->flask_bootstrap->-r requirements.txt (line 2)) (7.1.2)
Requirement already satisfied: Jinja2>=2.10 in /usr/local/lib/python3.8/site-packages (from Flask>=0.8->flask_bootstrap->-r requirements.txt (line 2)) (2.11.2)
Requirement already satisfied: Werkzeug>=0.14 in /usr/local/lib/python3.8/site-packages (from Flask>=0.8->flask_bootstrap->-r requirements.txt (line 2)) (0.14.1)
Requirement already satisfied: MarkupSafe in /usr/local/lib/python3.8/site-packages (from WTForms->flask_wtf->-r requirements.txt (line 4)) (1.1.1)
Building wheels for collected packages: flask-bootstrap, visitor
  Building wheel for flask-bootstrap (setup.py): started
  Building wheel for flask-bootstrap (setup.py): finished with status 'done'
  Created wheel for flask-bootstrap: filename=Flask_Bootstrap-3.3.7.1-py3-none-any.whl size=460123 sha256=49921fcc5037c9976d1c97b390112a6a2e498aab084da48867b8bd35d3f58b9b
  Stored in directory: /root/.cache/pip/wheels/f2/a3/85/fe8b65a65a447c9906e3b7edb7d9e6c74dfa9c8425c3dd3007
  Building wheel for visitor (setup.py): started
  Building wheel for visitor (setup.py): finished with status 'done'
  Created wheel for visitor: filename=visitor-0.1.3-py3-none-any.whl size=3931 sha256=d2262b87a54acdd4589a9393cef4cb66253aa4b542128a17ab0601184068bcbb
  Stored in directory: /root/.cache/pip/wheels/d3/40/52/5dae7760434a82caf8b8f88323029188b2d4ea3ac1235e550a
Successfully built flask-bootstrap visitor
Installing collected packages: dominate, visitor, flask-bootstrap, flask-moment, WTForms, flask-wtf, youtube-dl
Successfully installed WTForms-2.3.3 dominate-2.6.0 flask-bootstrap-3.3.7.1 flask-moment-0.11.0 flask-wtf-0.14.3 visitor-0.1.3 youtube-dl-2021.3.2
WARNING: You are using pip version 20.1; however, version 21.0.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container d7b2754acbc3
 ---> 24ada3febae5
Successfully built 24ada3febae5
Successfully tagged smartflask:1.1
pydoc@pydoc:~/python3_scripts/docker_flask_youtube2mp3$ docker run --name my-flask-uwsgi-2 -p 5042:5000 -d -v ~/python3_scripts/docker_flask_youtube2mp3:/app smartflask:1.1 uwsgi
03de114a5955f794c6bcee4d1743a8992abebae5d6198c6554c85b87c36e30e4
```

所需東西準備好後，開始下指令建立映像檔

以新產生的映像檔為基礎，新建一個容器來測試應用程式


https://60.249.144.60:5001
pydoc@pydoc:~$ docker image save -o ~/tmp/smartflask.tar smartflask:1.1
作法 3: 移植映像檔 (ubuntu->dsm)
Docker
概況
容器
倉庫伺服器
映像檔
網路
日誌
Administrator@WTGroup:~$ sudo docker image load -i ~/docker_image/smartflask.tar
Password:
c2adabaecedb: Loading layer 72.49MB/72.49MB
6376837eded8: Loading layer 7.324MB/7.324MB
dac74b28432f: Loading layer 110.6MB/110.6MB
62cdbc1bc81e: Loading layer 4.608kB/4.608kB
a9e858adafbc: Loading layer 8.189MB/8.189MB
778d7796b71e: Loading layer 2.56kB/2.56kB
ba62ae6e7623: Loading layer 300.8MB/300.8MB
4d8ffbacbf1e: Loading layer 3.584kB/3.584kB
25977cd0b329: Loading layer 3.584kB/3.584kB
19d9805b2966: Loading layer 3.072kB/3.072kB
1b401436b305: Loading layer 2.56kB/2.56kB
ea2cfdd47bb2: Loading layer 151.7MB/151.7MB
e1e4a441cea1: Loading layer 17.28MB/17.28MB
Loaded image: smartflask:1.1
Administrator@WTGroup:~$
python:3.8
倉庫伺服器: Docker Hub
Python is an interpreted, interactive, object-oriented, open-source programming language.
882 MB
smartflask:1.1
倉庫伺服器: Docker Hub
655 MB
移植後，映像檔可以在 dsm(群暉的作業系統) 的圖形化界面上看到，但是容器沒出現
tzahi12345/youtubedl-material:4.2
倉庫伺服器: Docker Hub
210 MB

作業系統的圖形化界面遺失容器資訊

執行中的容器缺少
youtube2mp3-uwsgi

```
Administrator@WTGroup:~$ sudo docker container ls
CONTAINER ID        IMAGE                                   COMMAND                  CREATED             STATUS              PORTS                                                  NAMES
11ae2c698154        smartflask:1.1                          "sh /start.sh uwsgi"     12 hours ago        Up 6 seconds        0.0.0.0:5044->5000/tcp                                 youtube2mp3-uwsgi
bb091384d9ab        bbsdocker/imageptt:20210223             "sh -c 'sudo -iu bbs…"   5 days ago          Up 4 days           0.0.0.0:8888->8888/tcp, 0.0.0.0:48763->48763/tcp       ptt_bbs
303c437a6907        oddrationale/docker-shadowsocks:latest  "/usr/local/bin/ssse…"   22 months ago       Up 7 days                                                                  oddrationale-docker-shadow
socks1
Administrator@WTGroup:~$
```

https://hub.docker.com/r/tiangolo/uwsgi-nginx-flask/

uwsgi-nginx-flask

Docker image with **uWSGI** and **Nginx** for **Flask** web applications in **Python 3.6** and above, and **Python 2.7** running in a single container. Optionally using Alpine Linux.

Description

This **Docker** image allows you to create **Flask** web applications in **Python** that run with **uWSGI** and **Nginx** in a single container.

The combination of uWSGI with Nginx is a common way to deploy Python Flask web applications. It is widely used in the industry and would give you decent performance. (*)

There is also an Alpine version. If you want it, check the tags from above.

* Note on performance and features

If you are starting a new project, you might benefit from a newer and faster framework based on ASGI instead of WSGI (Flask and Django are WSGI-based).

You could use an ASGI framework like:

- **FastAPI** (which is based on Starlette) with this Docker image: **tiangolo/uvicorn-gunicorn-fastapi**.
- **Starlette** directly, with this Docker image: **tiangolo/uvicorn-gunicorn-starlette**.

FastAPI, or Starlette, would give you about 800% (8x) the performance achievable with Flask using this image (**tiangolo/uwsgi-nginx-flask**). You can see the third-party benchmarks here.

Also, if you want to use new technologies like WebSockets it would be easier (and *possible*) with a newer framework based on ASGI, like FastAPI or Starlette. As the standard ASGI was designed to be able to handle asynchronous code like the one needed for WebSockets.

If you need Flask

If you need to use Flask (instead of something based on ASGI) and you need to have the best performance possible, you can use the alternative image: **tiangolo/meinheld-gunicorn-flask**.

tiangolo/meinheld-gunicorn-flask will give you about 400% (4x) the performance of this image (**tiangolo/uwsgi-nginx-flask**).

It is very similar to **tiangolo/uwsgi-nginx-flask**, so you can still use many of the ideas described here.

Github
tiangolo/uwsgi-nginx-flask-docker

8 倍效能

4 倍效能

為何需要別人的映像檔？
有很多原因，例如
1 不會自己架設正式的 http server
2 安全性 / 穩定
3 程式容易和作業系統切割
4 效能有機會大提昇