Network Programming

- 網路程式設計-

部署網站到Heroku

授課教師：張珀銀 老師

13

部署網站到Heroku

0

前一單元內容回顧

1

部署網站環境建置

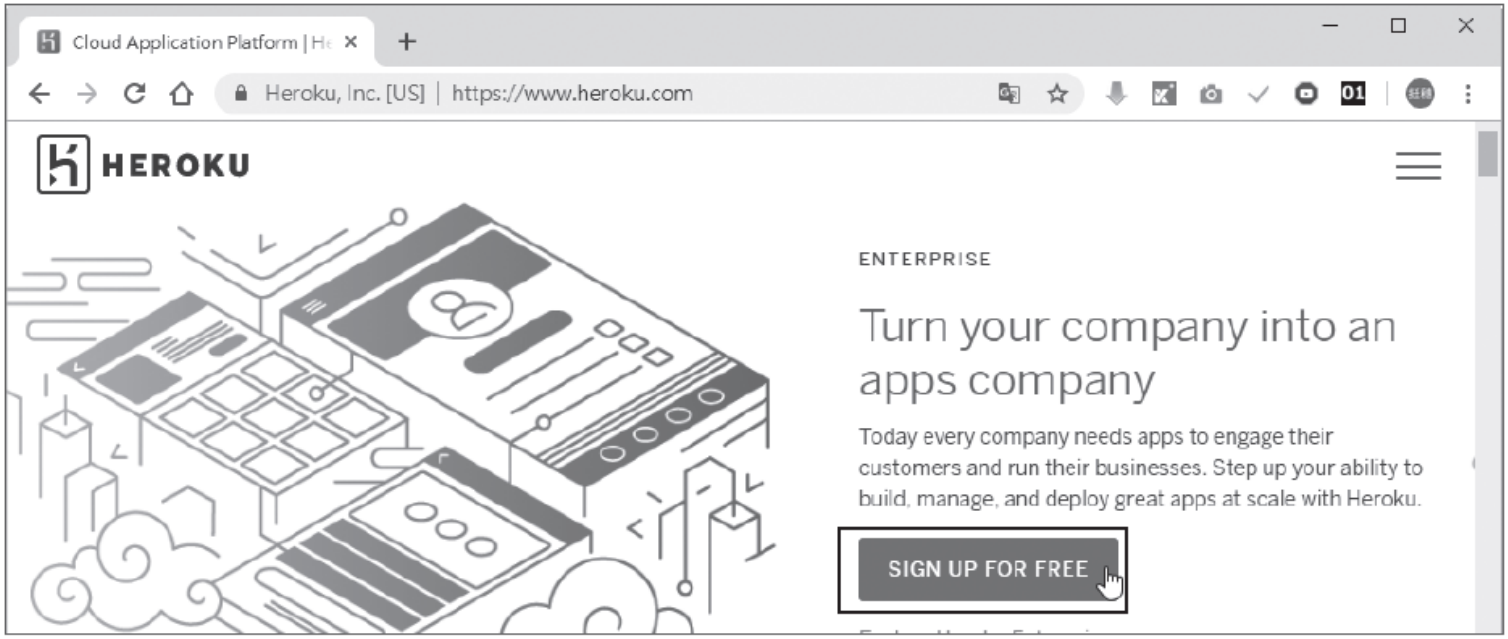
13.1 部署網站環境建置

在本機執行網站雖然方便，但無法讓所有人瀏覽自己開發的成果。而自行架設網頁伺服器不僅需耗費大量的時間及金錢，後續管理更要花費不少精力。將網站置於PaaS(Platform as a Service) 網路服務平台是目前大多數網站開發者的選擇，PaaS 將網站視為一個應用程式，只要調整網站的結構符合PaaS 的規則，系統就可正常運行。

PaaS 的優點是開發者只需專注於網站功能，其餘主機相關事宜都由 PaaS 去操心。Heroku (https://www.heroku.com/) 是一個支援多種程式語言的雲端即時服務平台，目前支援Ruby、Java、Node.js、Scala、Clojure、Python、PHP 和Perl。雖然目前Google、Microsoft Azure、Amazon 都有提供類似的服務，但是Heroku 除了簡單易用，並且有提供免費的方案，非常適合一般開發者使用。

13.1.1 建立 Heroku 應用程式

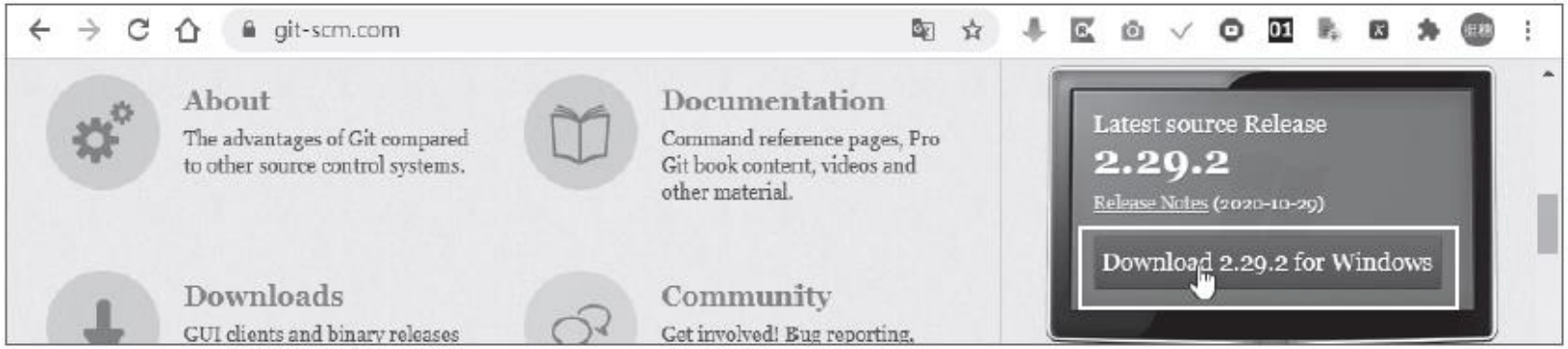
如果沒有帳號必須先註冊： 開啟「https://www.heroku.com/」網頁，按 **SIGN UP FOR FREE** 鈕進入建立免費帳號頁面。填寫所有欄位資料，最後按**CREATE FREE ACCOUNT** 鈕建立免費帳號，建立後按**LOG IN** 鈕登入。



13.1.2 安裝 Git 版本管理軟體

Heroku 使用Git 版本管理軟體進行網站部署，因此必須安裝Git 版本管理軟體。

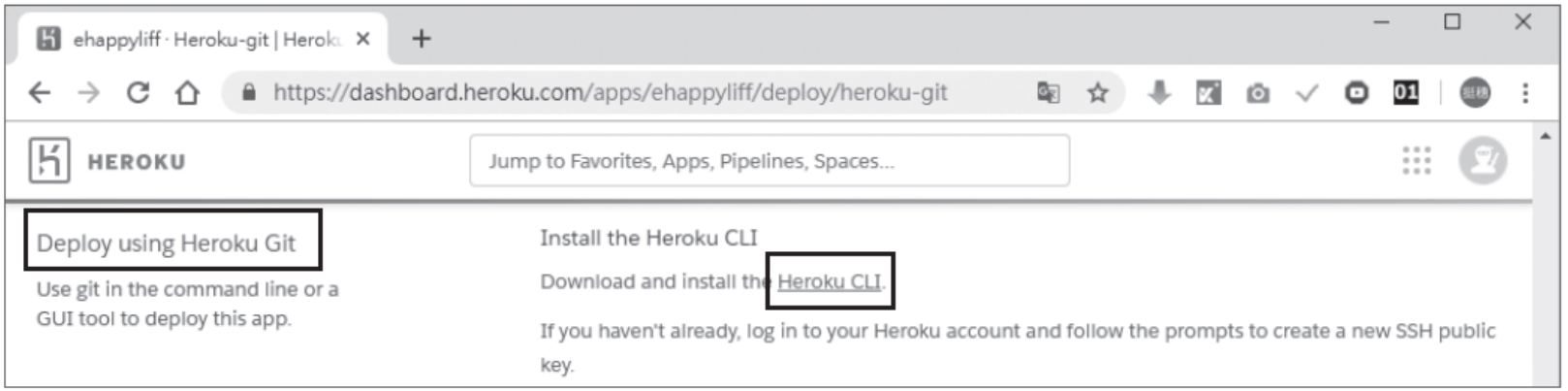
開啟Git 官網「https://git-scm.com/」，點選**Download x.x.x for Windows** 鈕下載 Git 安裝程式。



13.1.3 安裝Heroku CLI

Heroku 使用Git 版本管理軟體進行網站部署，Heroku 官方撰寫了Heroku CLI 套裝軟體，方便使用者利用Git 將檔案與Heroku 伺服器同步。

應用程式建立完成後，會切換到應用程式管理頁面，將網頁向下捲到**Deploy using Heroku Git** 處，點選**Heroku CLI** 連結。



13.1.4 建置空白虛擬環境

Python 環境使用一段時間後會安裝許多模組，如果部署專題時將這些模組一併部署到伺服器的話，不但會佔據大量伺服器空間，也可能影響伺服器執行效率，因此部署專題時，最好先新增一個空白虛擬環境，將要部署的專題置於此空白虛擬環境，再安裝專題所需的模組，就可達到最佳部署狀態。

1. Python 安裝時並未加入建立虛擬環境的模組，在命令提示字元視窗執行下面命令安裝建立虛擬環境的模組：

```
pip install virtualenv
```

2. 切換到C 磁碟機根目錄，以virtualenv 指令建立herokuenv 虛擬環境：

```
cd c:\
virtualenv herokuenv
```

2

部署網站

13-2 部署網站

部署網站的環境建置完成後，還需要調整網站的檔案結構，才能將網站檔案上傳到伺服器讓所有使用者瀏覽。

13.2.1 使用現有資料庫的網站結構

安裝模組

首先安裝在Heroku 伺服器執行Django 網站要用到的模組。在 **命令提示字元** 視窗以下列命令安裝模組：

```
pip install dj-database-url dj-static gunicorn
```

建立 <requirements.txt>

部署網站時，Heroku 如何知道該安裝哪些模組呢？ Heroku 是根據網站根目錄中<requirements.txt> 檔中所列的模組名稱及版本安裝模組，我們只要將目前虛擬環境中所有已安裝的模組名稱匯出即可：在 **命令提示字元** 視窗執行下列命令：

```
cd c:\herokuenv\boardhero
pip freeze > requirements.txt
```

建立 <Procfile>

網站根目錄中<Procfile> 檔是告訴Heroku 啟動網站的方式。在<boardhero> 資料夾新增一個<Procfile> 檔案，輸入下列文字：

```
web: gunicorn --pythonpath board board.wsgi
```

建立<runtime.txt>

網站根目錄中<runtime.txt> 檔是告訴Heroku 使用的Python 版本。目前支援的Python 版本有3.9.5、3.8.10、3.7.10 及3.6.13。這些版本可能會隨時間而改變，最新支援的Python 版本請查看「https://devcenter.heroku.com/articles/pythonsupport#supported-runtimes」網頁。

在 <boardhero> 資料夾新增一個 <runtime.txt> 檔案，輸入下列文字：

```
python-3.7.10
```

「3.7.10」為 Python 版本。

建立 <prod_settings.py>

在Heroku 中使用的設定有些會與本機執行的設定不相同，最好的方式是另外建立一個Heroku 使用的設定檔，此設定檔先匯入本機執行的所有設定，再撰寫Heroku 中特有的設定。為了方便匯入原有設定，新增的設定檔最好與<settings.py> 在同一個資料夾。

例如新的設定檔名稱為<prod_settings.py>： 在<board> 資料夾建立<prod_settings.py> 檔，其內容為：

```
from .settings import *
STATIC_ROOT = 'staticfiles'
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
ALLOWED_HOSTS = ['*']
DEBUG = False
```

建立 <.gitignore>

為了避免浪費Heroku 儲存空間，一些不必要的檔案可以不必上傳到Heroku 伺器。在網站根目錄的<.gitignore> 檔內列出的檔案及資料夾，部署時將不會上傳到Heroku。

在<boardhero> 資料夾新增一個<.gitignore> 檔案，輸入下列文字：

```
*.pyc
__pycache__
staticfiles
```

修改<wsgi.py>

Heroku 處理靜態檔案的方式與本機不相同，因此要修改<wsgi.py> 檔內容，使其符合Heroku 靜態檔案處理方式：(粗體字為修改部分)

```
略……
import os
from django.core.wsgi import get_wsgi_application
from dj_static import Cling

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "board.settings")
application = Cling(get_wsgi_application())
```

13.2.2 部署網站到 Heroku

切換到專案檔案資料夾(<C:\herokuenv\boardhero)，輸入下列指令：

```
heroku login
```

按任意鍵在Heroku 登入網頁完成登入。

接著在本機新建一個Git 倉庫(repository) 來存放專案檔案：

```
git init
```

再將此Git 倉庫與Heroku 伺服器的ehappyboard1 應用程式建立連結：

```
heroku git:remote -a ehappyboard1
```

設定Heroku 使用<board> 資料夾的<prod_settings.py> 檔案內容做為網站設定值：

```
heroku config:set DJANGO_SETTINGS_MODULE=board.prod_settings
```

將專案所有檔案加入Git 追縱：

```
git add .
```

將所有追縱的檔案加入Git 倉庫，並將此次執行動作命名為「init commit」：

```
git commit -am "init commit"
```

如此就可以將檔索上傳到Heroku 了：

```
git push heroku master
```

Heroku 的網址為「https:// 應用程式名稱.herokuapp.com/」：在瀏覽器網址列輸入「https://ehappyboard1.herokuapp.com/」就可看到網路留言版首頁。

13.2.3 部署後修改網站內容

網站部署一段時間後，也許發現需要修正或新增一些功能，要如何修改Heroku伺服器的網站內容呢？ Heroku 使用Git 做部署工具，因此只要在本機修改網站內容後更新Git 倉庫，重新上傳檔案就可更新Heroku 伺服器網站內容，同時Git會記錄每次更新所修改的內容。

於 **命令提示字元** 視窗切換到網站檔案資料夾 (<C:\herokuenv\boardhero>)，登入Heroku 伺服器：

```
heroku login
```

進行所有檔案追縱：

```
git add .
```

將檔案加入Git 倉庫，並將此次變更命名為「modify index.html」：

```
git commit -am "modify index.html"
```

最後將檔案上傳到Heroku 就完成網站內容更新：

```
git push heroku master
```

Exercise 13-1:
部署網站

Final-term

Final-term

1. HTML:
依循網址教學完成範例 URL
並嘗試建置於HEROKU。

2. FLASK:
依循網址教學完成範例URL，建置於PythonAnywhere