成為初級資料分析師 | Python 與資料科學應用

網頁資料擷取排程與分享資料

郭耀仁

大綱

- 排程網頁資料擷取
- 創建 Web API 分享資料

排程網頁資料擷取

如何排程網頁資料擷取

- 創建一個 AWS EC2 元件
- 在 EC2 上執行網頁資料擷取程式
- 以 crontab 排程

創建一個 AWS EC2 元件：關於 EC2

EC2 （Elastic Compute Cloud）是一種網站服務，可在雲端提供安全、可調整大小的運算容量，旨在降低開發人員進行網站規模化的難度。

創建一個 AWS EC2 元件：前往 AWS 創建

- 前往 https://aws.amazon.com (https://aws.amazon.com) 註冊帳號
- 選擇 EC2
- 點選 Launch Instance
- 挑選 Free-tier eligible 標籤的映像檔
- 選擇 Ubuntu Server
- 在 Configure Security Group 設定 Custom port range
- 選擇 Create a new key pair
- 點選 Download Key Pair 下載金鑰
- 點選 Launch Instance，EC2 創建完成

創建一個 AWS EC2 元件：設定 Custom port range

創建一個 AWS EC2 元件： Windows 以 PUTTY 連線

https://www.putty.org/ (https://www.putty.org/)

創建一個 AWS EC2 元件：Unix 以 ssh 連線

- `chmod 400 YOUR-KEYPAIR-NAME.pem`
- `ssh -i "YOUR-KEYPAIR-NAME.pem" ubuntu@YOUR-EC2-PUBLIC-DNS`

在 EC2 上執行網頁資料擷取程式：安裝 miniconda

- 更新 apt `sudo apt update`
- 取得 Miniconda `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh`
- 安裝 Miniconda `bash Miniconda3-latest-Linux-x86_64.sh`

在 EC2 上執行網頁資料擷取程式：建立虛擬環境

```
conda create -n flask python=3
conda activate flask
```

在 EC2 上執行網頁資料擷取程式：安裝所需套件

- 安裝所需套件 `pip install requests beautifulsoup4 pandas flask`
- 執行程式 `python YOUR-SCRIPT.py`

```
# YOUR-SCRIPT.py
import requests
import pandas as pd
from bs4 import BeautifulSoup
import sqlite3
import datetime

def insert_data():
    conn = sqlite3.connect('/home/ubuntu/demo.db')
    current_dt = datetime.datetime.now().strftime("%Y-%m-%d %X")
    repo_social_counts = []
    for page in range(2):
        r = requests.get("https://github.com/search?p={}&q=stars%3A%3E0&s=stars&t
ype=Repositories".format(page+1))
        soup = BeautifulSoup(r.text, 'html.parser')
        repos = [i.get("href") for i in soup.select(".pr-md-3 .v-align-middle")]
        repo_links = ["https://github.com" + repo for repo in repos]
        for repo_link, repo in zip(repo_links, repos):
            repo_dict = {}
            r = requests.get(repo_link)
            soup = BeautifulSoup(r.text, 'html.parser')
            watches, stars, forks = [int(i.text.strip().replace(",", "")) for i i
n soup.select(".social-count")]
            repo_dict["scrapingTime"] = current_dt
            repo_dict["repoOrg"] = repo.split("/")[1]
            repo_dict["repoName"] = repo.split("/")[2]
            repo_dict["watch"] = watches
            repo_dict["star"] = stars
            repo_dict["fork"] = forks
            repo_social_counts.append(repo_dict)
            print("Scraping {}...".format(repo))
    df = pd.DataFrame(repo_social_counts)
    df.to_sql("top_github_repos", con=conn, if_exists="replace", index=False)
try:
    insert_data()
```

```
except:
    pass
```

以 crontab 排程：先將 EC2 調整為台灣台北時間

- `sudo apt-get install tzdata`
- `sudo dpkg-reconfigure tzdata`
- 以 `timedatectl` 檢查

以 crontab 排程：關於 cron 與 crontab

> *cron 和 crontab 可以在指定時間運行指定命令，藉由編輯 crontab 能夠安排任務或工作在特定日期、時間定期執行。*

Source: Linux Basics for Hackers (https://www.amazon.com/Linux-Basics-Hackers-Networking-Scripting/dp/1593278551)

以 crontab 排程：crontab 的格式

分鐘 小時 日期 月份 星期 執行指令

- 分鐘：0-59
- 小時：0-23
- 日期：1-31
- 月份：1-12
- 星期：0-6（0 為 Sunday）

以 crontab 排程：每分鐘執行一次 YOUR-SCRIPT.py

```
0-59 * * * * /home/ubuntu/miniconda3/envs/flask/bin/python3 /home/ubuntu/YOUR-SCR
IPT.py
```

創建 Web API 分享資料

Web API 是透過網站應用程式達成讓資料在不同工作環境中共同分享的工具

輕量的 Flask 框架就足夠勝任 Web API 的需求：管理 HTTP 請求和顯示資料內容

> *Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.*

Source: https://palletsprojects.com/p/flask/ (https://palletsprojects.com/p/flask/)

測試 Flask 的 Hello World

```python
# api.py
import flask

app = flask.Flask(__name__)
app.config["DEBUG"] = True

@app.route('/', methods=['GET'])
def home():
    return "<h1>Hello world!</h1>"

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

前往 Public DNS (IPv4):5000 觀看是否有 Hello world!

修改 api.py 顯示表格所有內容

```
# api.py
import flask
from flask import jsonify
import numpy as np
import pandas as pd
import sqlite3

app = flask.Flask(__name__)
conn = sqlite3.connect('/home/ubuntu/demo.db')
query_str = """SELECT * FROM top_github_repos;"""
cur = conn.cursor()
cur.execute(query_str)
results = cur.fetchall()
cur.close()
@app.route('/', methods=['GET'])
def home():
    return jsonify(results)

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

前往 Public DNS (IPv4):5000 觀看是否有表格所有內容

中斷 EC2 連線之後再前往 Public DNS (IPv4):5000 觀看是否有表格所有內容

使用 systemd 將 Flask API 轉換成服務，避免中斷 EC2 連線後 Flask 也停止服務

```
cd /etc/systemd/system
sudo vim YOUR-SERVICE.service
```

在 YOUR-SERVICE.service 檔案中輸入下列內容

```
# YOUR-SERVICE.service
[Unit]
Description=Web API
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/home/ubuntu
ExecStart=/home/ubuntu/miniconda3/envs/flask/bin/python3 /home/ubuntu/api.py
WatchdogSec=60
Restart=always

[Install]
WantedBy=multi-user.target
```

執行 systemd

```
sudo systemctl daemon-reload
sudo systemctl start YOUR-SERVICE
sudo systemctl status YOUR-SERVICE
```

前往 Public DNS (IPv4):5000 觀看是否有表格所有內容

- 中斷 EC2 連線
- 過一段時間觀察資料是否有更新

最後處理啟動 Flask 服務時會看到的警告訊息

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

這個訊息建議在 Flask API 與 HTTP 請求中間置放一個 WSGI

1. HTTP 請求
2. WSGI
3. Flask API

WSGI, Web Server Gateway Interface 僅適用 Python，負責介接網頁服務與網頁框架

在專案資料夾中新增 wsgi.py

```
from api import app

if __name__ == "__main__":
    app.run()
```

安裝 Python 網頁服務：gunicorn

```
conda activate flask
pip install gunicorn
```

修改 YOUR-SERVICE.service

```
# YOUR-SERVICE.service
[Unit]
Description=Web API
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/home/ubuntu
ExecStart=/home/ubuntu/miniconda3/envs/flask/bin/gunicorn --bind 0.0.0.0:5000 -w
4 wsgi:app
WatchdogSec=60
Restart=always

[Install]
WantedBy=multi-user.target
```

執行 systemd

```
sudo systemctl daemon-reload
sudo systemctl start YOUR-SERVICE
sudo systemctl status YOUR-SERVICE
```

前往 Public DNS (IPv4):5000 觀看是否有表格所有內容

- 中斷 EC2 連線
- 過一段時間觀察資料是否有更新