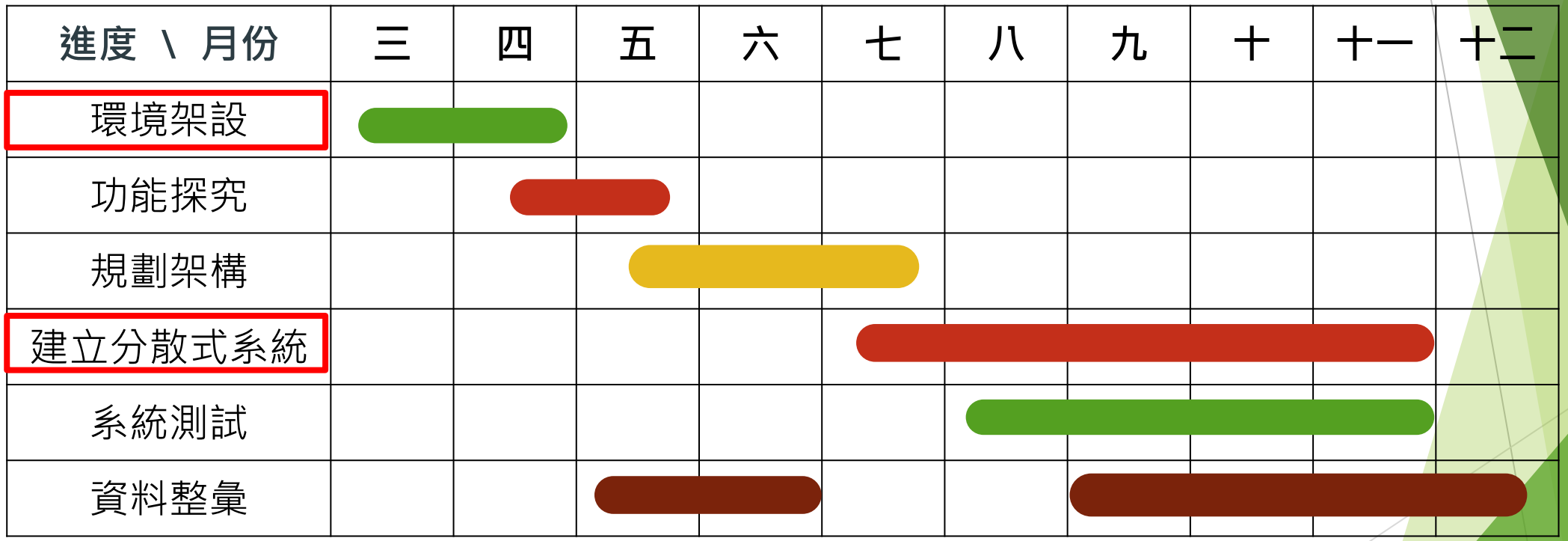
Reverse Proxy for Jobe

前言

- 這份 PPT 主要紀錄的是從暑假到年尾的 meeting。
- 暑假之前都在建構環境與規劃系統架構。

專題進度總覽(一)

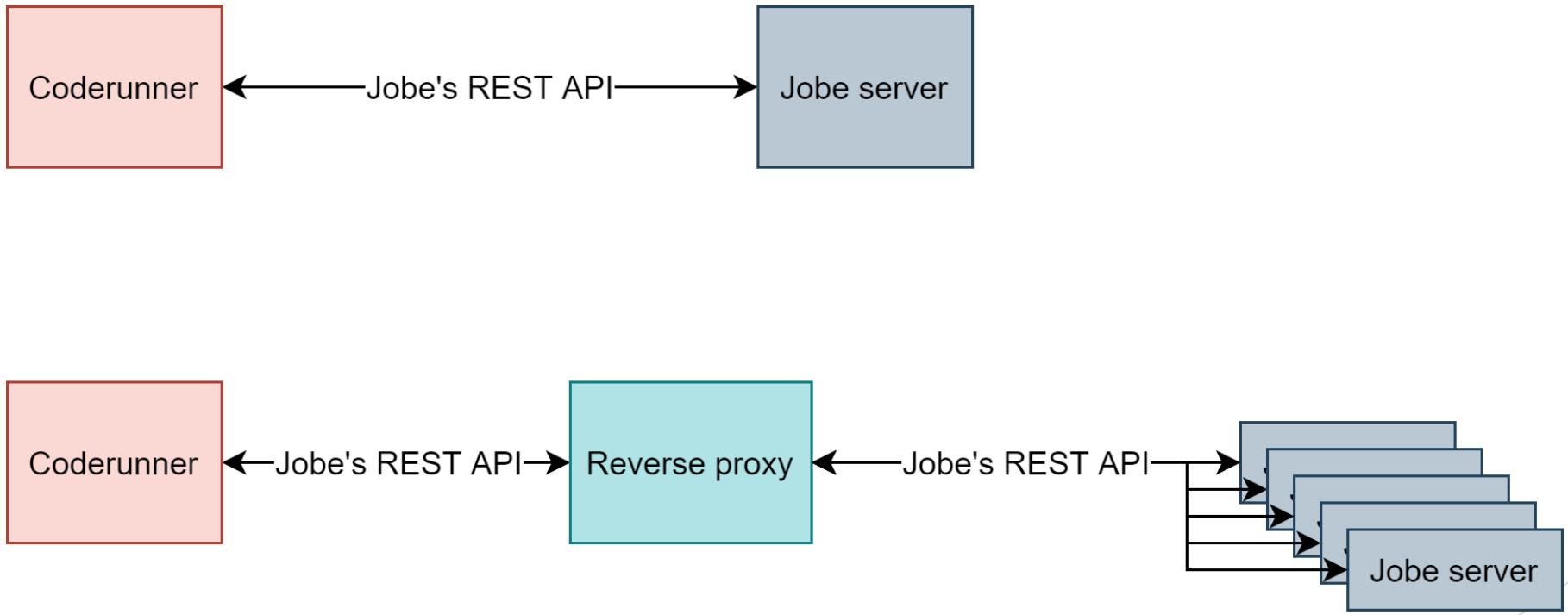


建立環境是一個很長的時程，有時候嘗試使用某個開源軟體不一定會成功。

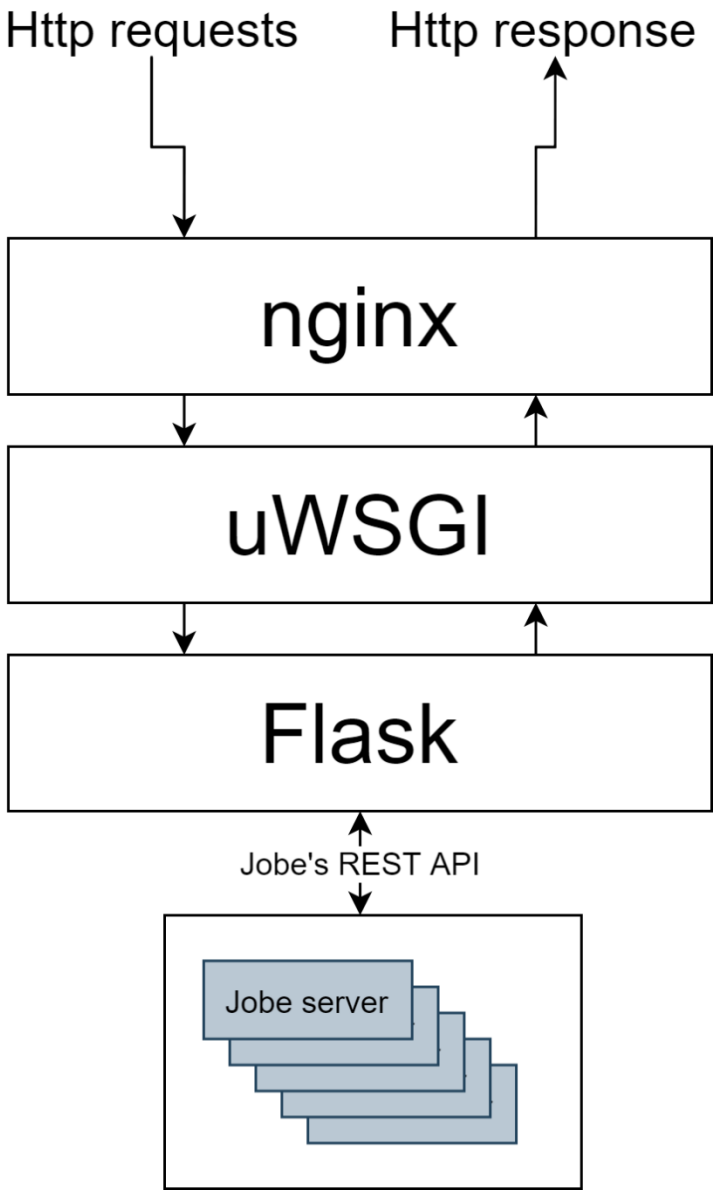
專題進度總攬(二)

- 進度 0720-0726
- 進度 0727-0802
- 進度 0803-0809
- 進度 0810-0922
- 進度 0923-0928
- 進度 1006-1013
- 進度 1021-1104

目標


Coderunner
Jobe's REST API
Jobe server
Coderunner
Jobe's REST API
Reverse proxy
Jobe's REST API
Jobe server

實作環境


Http requests
Http response
nginx
uWSGI
Flask
Jobe's REST API
Jobe server

進度 0720-0726

- 架設VS Code的遠端開發環境
- 然後腦殘把前一周做進度的給刪掉了
- 重寫進度，並改善例外以及錯誤處理
- 更新了jobe_list.json的架構
- 計畫各個API的實現方式

Jobe有實現的API call

- `get_languages`
- `put_file`
- `check_file`
- `submit_run`

Jobe沒有實現的API call

- get_run_status
  - 只回覆404

application > controllers > Restapi.php > Restapi

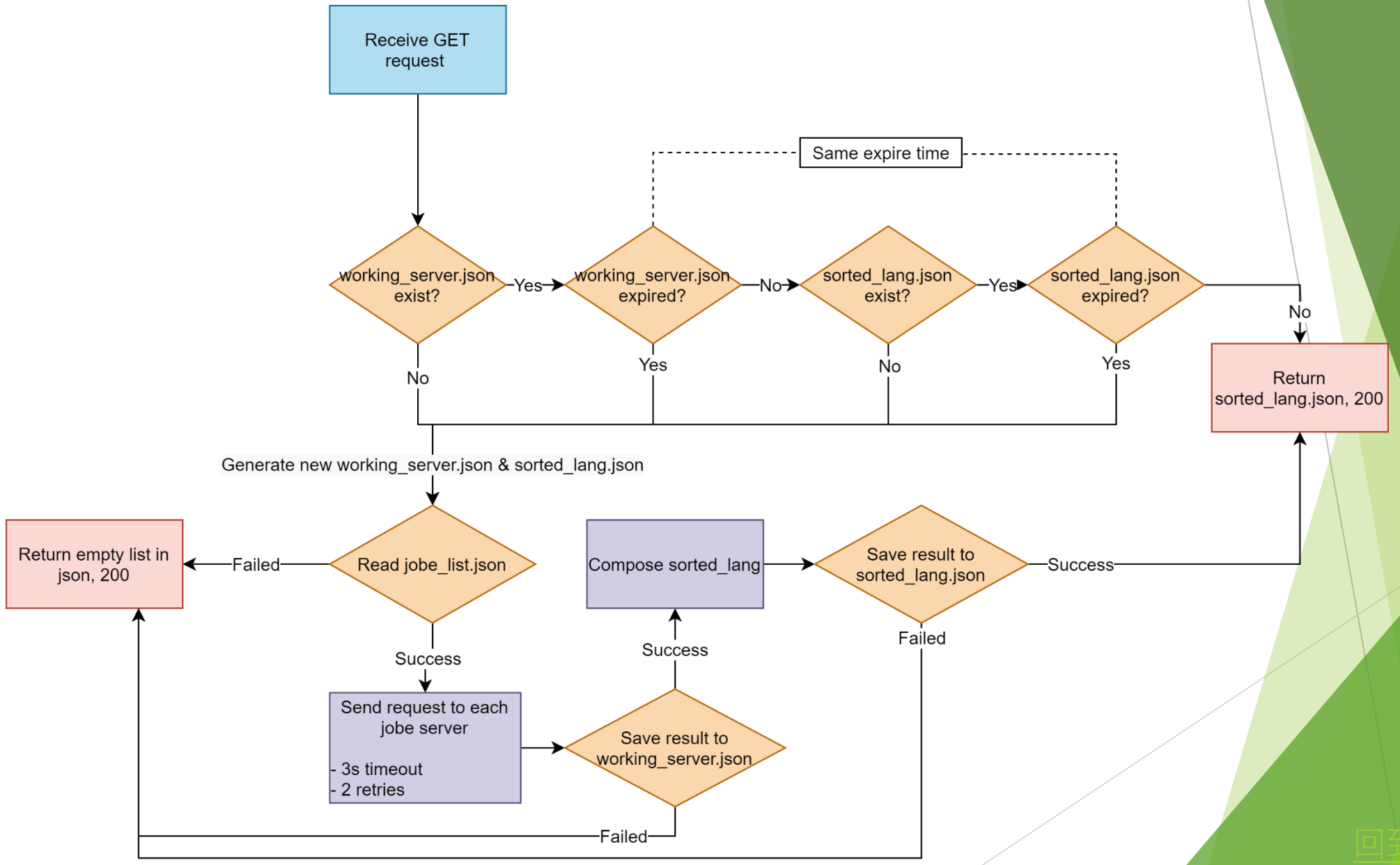
```
249
250     // **********************
251     //      RUN_RESULTS
252     // **********************
253     public function runresults_get()
254     {
255         $this->error('runresults_get: unimplemented, as all submissions run immediately.', 404);
256     }
257
```

- post_file
  - 只回覆403

application > controllers > Restapi.php > Restapi

```
126     // Post file
127     public function files_post() {
128         $this->error('file_post: not implemented on this server', 403);
129     }
```

get_languages架構



回到force_update

get_languages目的

- 回覆目前所有活著的jobe支援的語言
- 產生working_server.json紀錄活著的jobe和其支援的語言
- 產生sorted_lang.json做快取用

進度 0727-0802

- `put_file` 和 `check_file` 完成
- `submit_run` 大致完成，可運行:
  - 目前可以用隨機分配
- working_server.json 架構更新

Working_server.json 架構

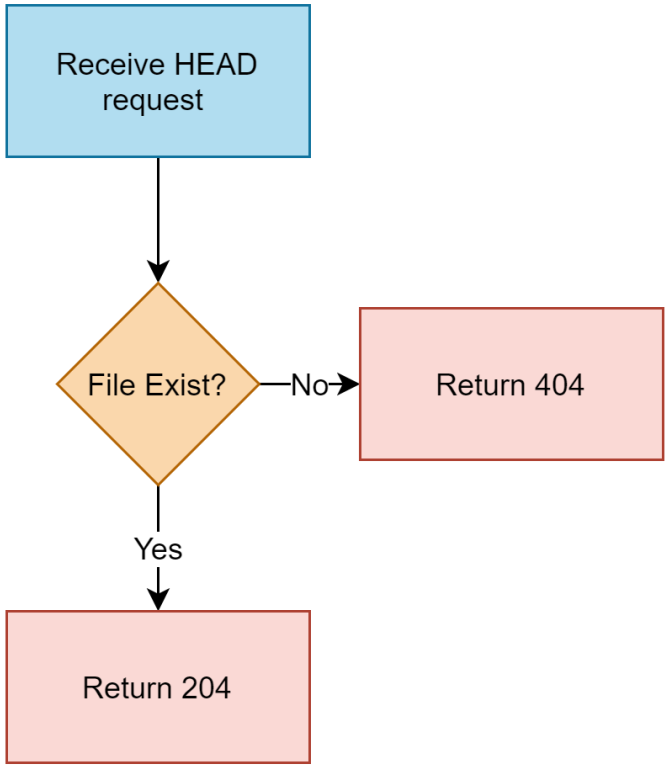
working_server.json

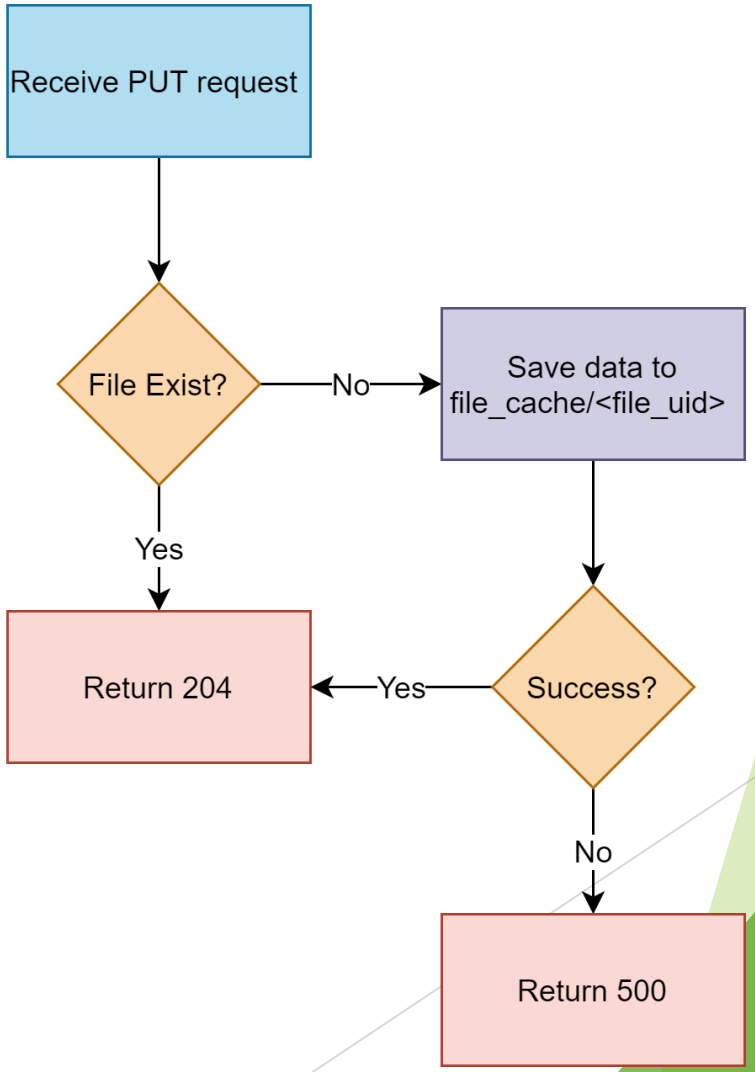
app > working_server.json > ...

```json
{
    "http://10.16.173.225:4000": {
        "c": "7.3.0",
        "cpp": "7.3.0",
        "java": "10.0.2",
        "nodejs": "8.10.0",
        "octave": "4.2.2",
        "pascal": "3.0.4",
        "php": "7.2.7",
        "python3": "3.6.5"
    },
    "http://10.16.173.225:4100": {
        "c": "7.3.0",
        "cpp": "7.3.0",
        "java": "10.0.2",
        "nodejs": "8.10.0",
        "octave": "4.2.2",
        "pascal": "3.0.4",
        "php": "7.2.7",
        "python38": "3.8.2",
        "python3": "3.6.5"
    },
    "http://10.16.173.226:4000": {
        "c": "7.3.0",
        "cpp": "7.3.0",
        "java": "10.0.2",
        "nodejs": "8.10.0",
        "octave": "4.2.2",
        "pascal": "3.0.4",
        "php": "7.2.7",
        "python3": "3.6.5"
    }
}
```

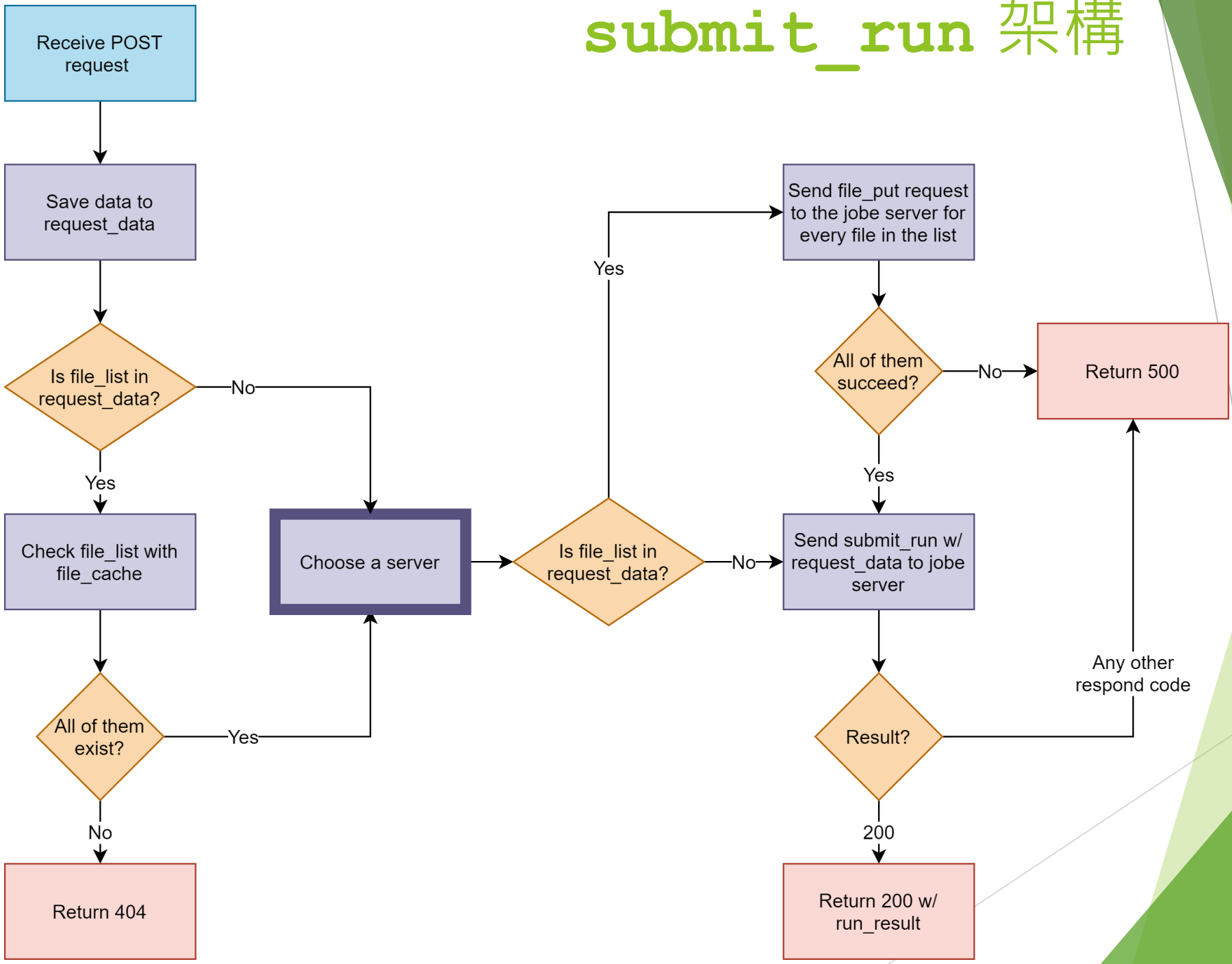
put_file 和 check_file 架構

Request URL:
.../jobe/index.php/restapi/files/<file_uid>





submit_run 架構



run_spec 物件
(存進request_data的內容)

run_spec.json

app > run_spec.json > ...

```
{
    "run_spec": {
        "language_id": "cpp",
        "sourcecode": "#include <iostream>\r\nusing namespace std ;\r\n\r\nint main(){\r\n    while(1){\r\n    cout << 5;\r\n    }\r\n}",
        "sourcefilename": "__tester__.cpp",
        "input": "1\n",
        "file_list": [
            [
                "8305fd623915109efc98803b093eab55",
                "code.cpp"
            ]
        ],
        "parameters": {
            "cputime": 2
        }
    }
}
```

進度 0803-0809

- 用Jmeter做簡單的測試，同時觀察jobe的行為

Jmeter 測試項目

- 單一jobe:
  - 同時30個要求
  - 同時100個要求
  - 同時200個要求
- Proxy:
  - 同時30個要求
  - 同時100個要求
  - 同時200個要求

- 測試跑的程式碼:
  - C++
  - 無限迴圈
  - CPU現時2秒

觀察結果:單一Jobe

- 伺服器滿載(10個使用中)時，jobe會將要求佇列起來
- 未滿載時，每一個要求回覆時間約為3秒

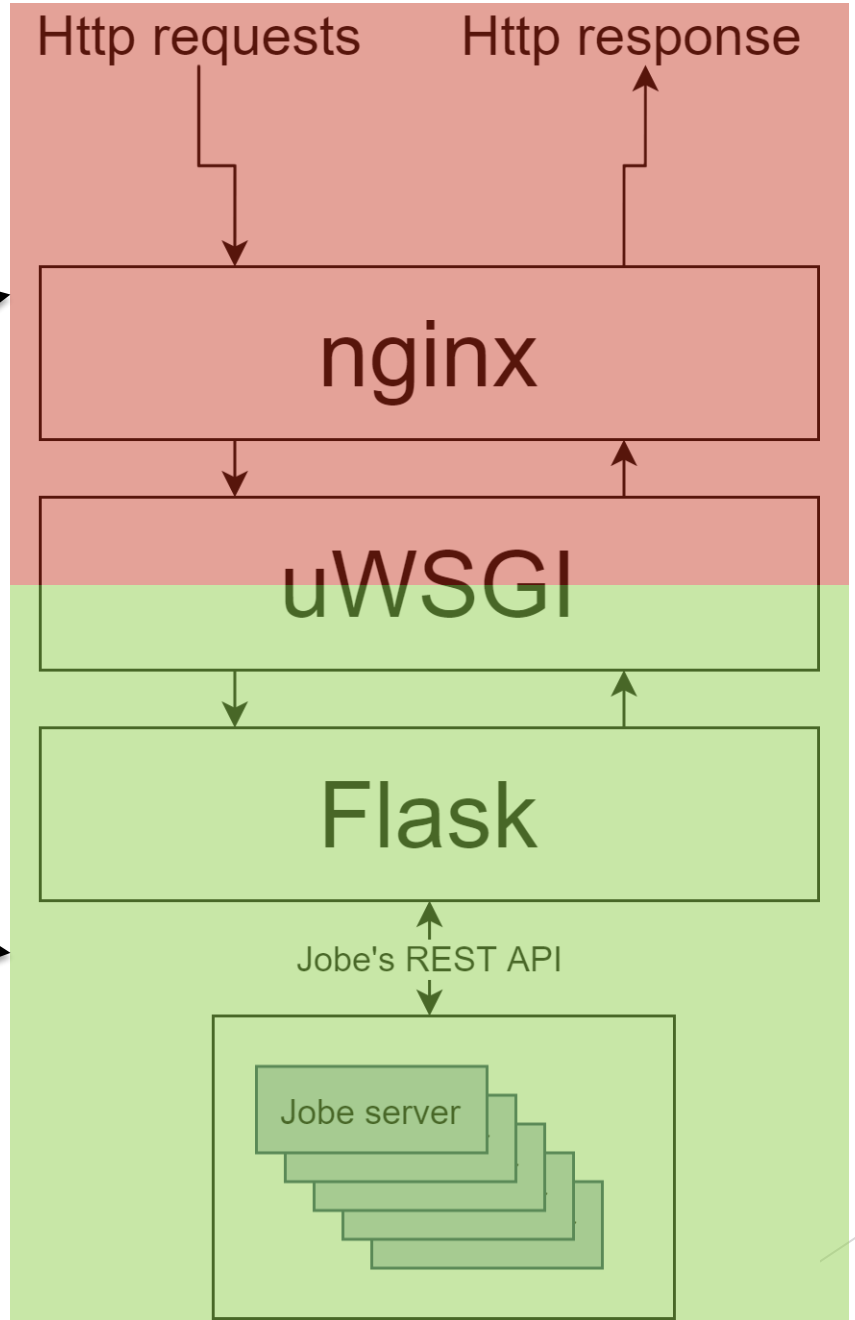
觀察結果:Proxy

- 30人同時要求效率不如單一Jobe
- 100人同時開始出現錯誤。
  - 從log中得知是目前nginx和uwsgi的設置沒辦法處裡這樣的流量

觀察結果:Proxy

主要問題出在這段

uWSGI完成回覆的
時間約為3秒



進度 0810-0922

- 調整uWSGI設定
- 搞得毫無頭緒
- uWSGI換成gunicorn
- 調整gunicorn設定

uWSGI的狀況

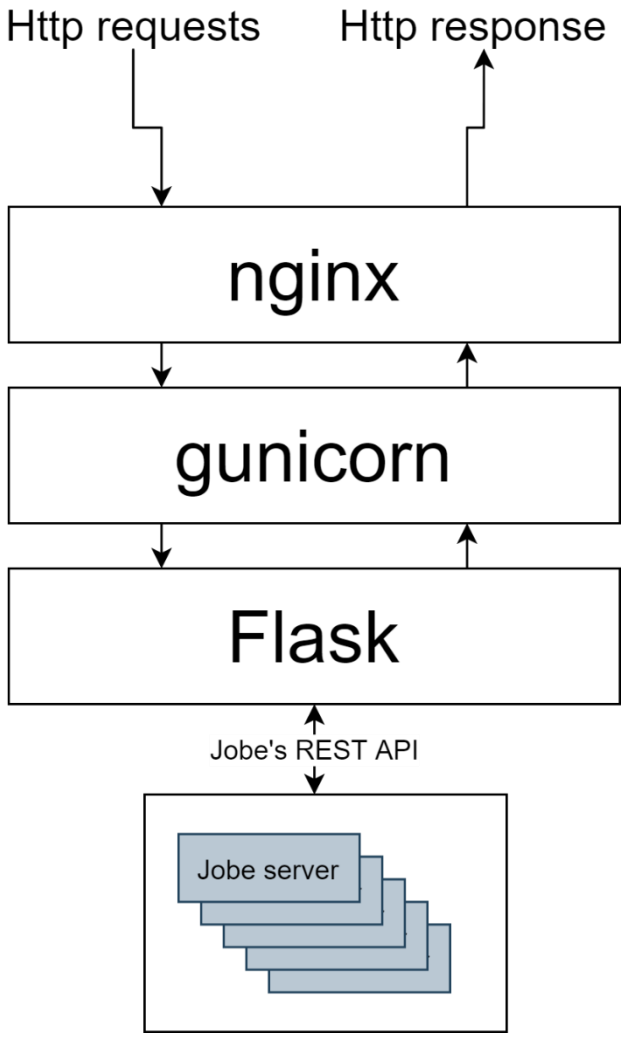
- 功能太多，可以設定的選項太多、太複雜。
- 官方文件不好讀。
- 最後決定換成gunicorn

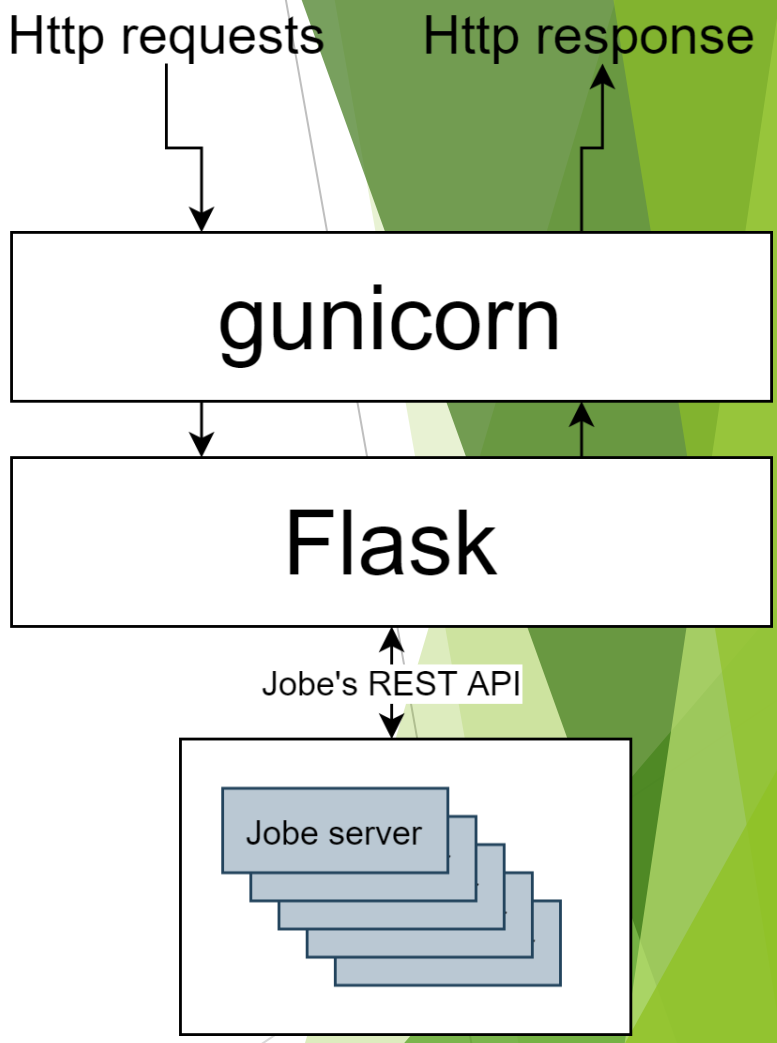
Gunicorn

- Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX
- pre-fork worker model
  - master process 在request來之前就fork子程序來接收request
- 比較容易設置，但可設置的程度較不及uWSGI
- 官方文件寫得比較明瞭

Gunicorn目前設置

- 以systemd服務執行
- 目前直接由gunicorn接收http request
- 官方建議用nginx轉發給gunicorn
- Timeout 300秒
- 150個worker
- 已經可以達到單一jobe的速度了
  - (甚至更快)





目前代辦事項

- Gunicorn和Flask的logging設置。
- 更多的測試。

補充：暑假研究 K8S 的過程

- 一開始會想用 K8S 的原因：
- 我們使用 docker 建立我們的 jobe server

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

Load balance

- 後來經過詢問，決定不採用 K8S。原因是「殺雞焉用牛刀」。
- 我們的專案算小，學生就那些人，手動部署即可，用了 K8S 反而會拖慢我們的速度或開發。

進度 0923-0928

- 找出賴媽課程中出現的問題的原因
- 設定好新的2台Jobe Servers

賴媽問題(未使用變數)

- https://hackmd.io/@kevinhsu/HylO1TFBP

設定新的Jobe

- 0928才拿到固定IP所以只有設定好而已。

進度 1006-1013

- 權重隨機分配
- Logging Handling
- force_update
- submit_runs 小修理

權重隨機分配

- jobe_list.json 多一個欄位儲存權重
- sorted_list.json 和 working_server.json 新增語言id “__weight”
- 用python的random.choices()做權重隨機選擇
- 順便縮短了該部分的程式碼

Logging Handling

- flask的log現在是讓gunicorn管理
- gunicorn的log現在還是在systemlog中

force_update

- 直接更新現在可用的伺服器列表(sorte_lang.json & working_server.json)
- 連帶將原get_languages()部分拉出成為獨立的函式

submit_runs 小修理

- 加入Timeout
- 加入接Timeout和ConnectionError兩個exception

進度 1014-1021

- Log檔終於存到獨立的檔案
- 評估自動重送
- File concurrency 問題浮現

File concurrency 問題浮現

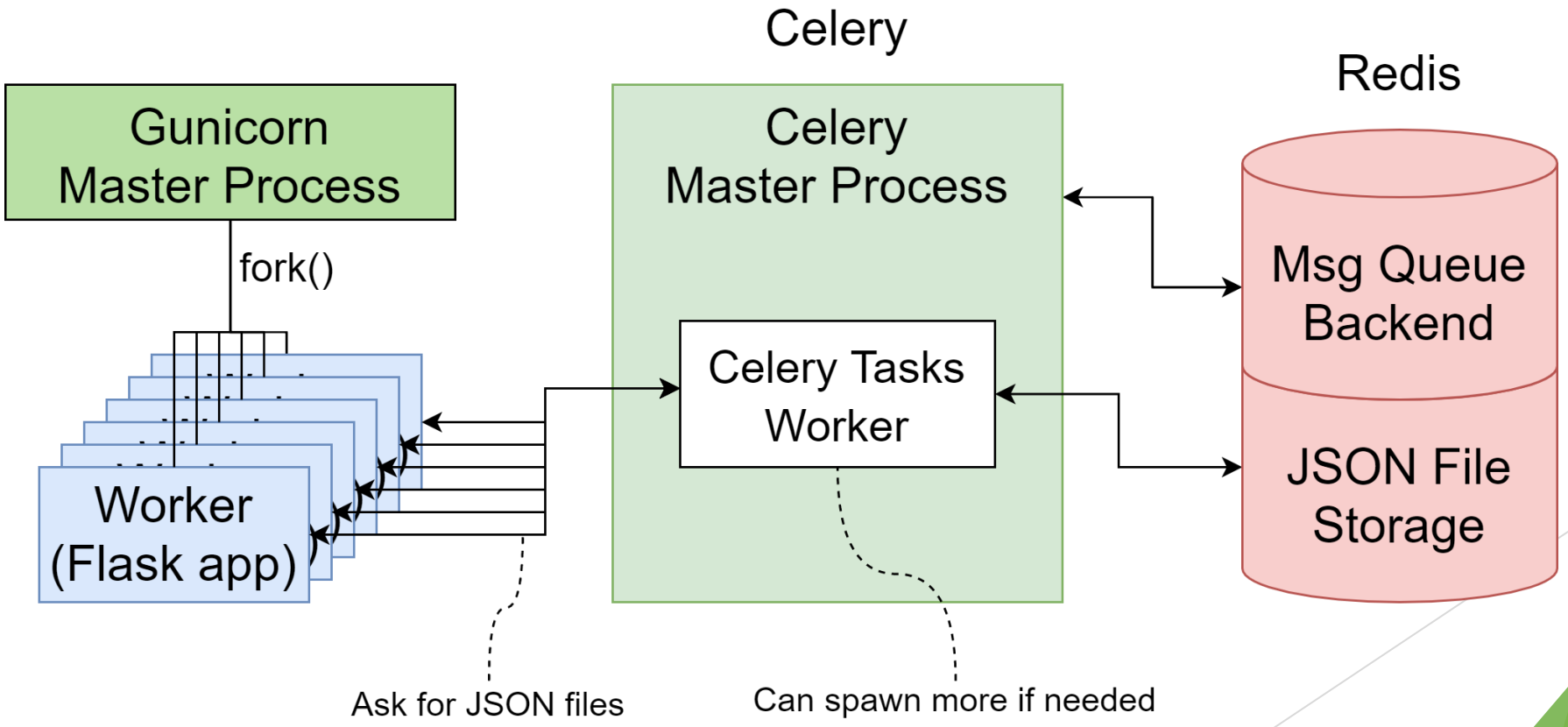
- 在使用Jmeter測試的時候出現的問題
- 在讀取各json檔時會出現讀入內容是空白的
- 猜測:讀入檔案時，其它worker(thread)正要寫入
- 預計用Threading.Lock()的方式解決

進度 1021-1104

- File concurrency 已解決

File concurrency 解決方法

- 加入Celery和Redis進系統中

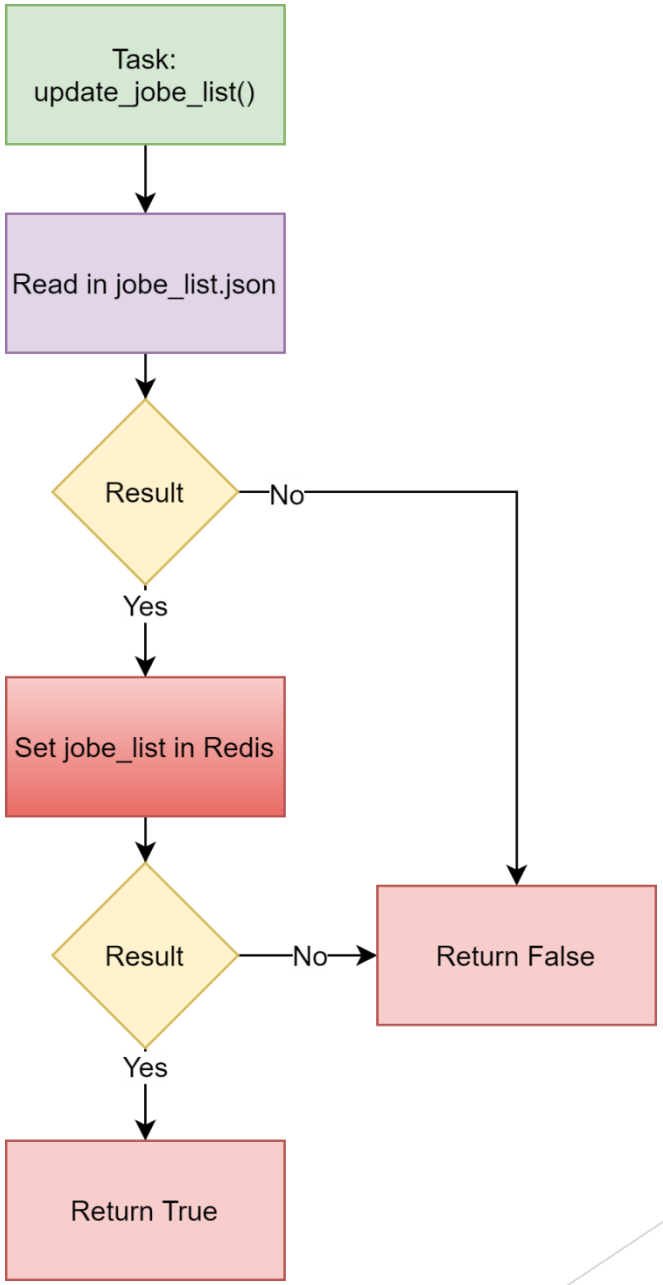


Celery Tasks

- 目前寫了三個
  - update_jobe_list()
  - update_working_server()
  - get_data()

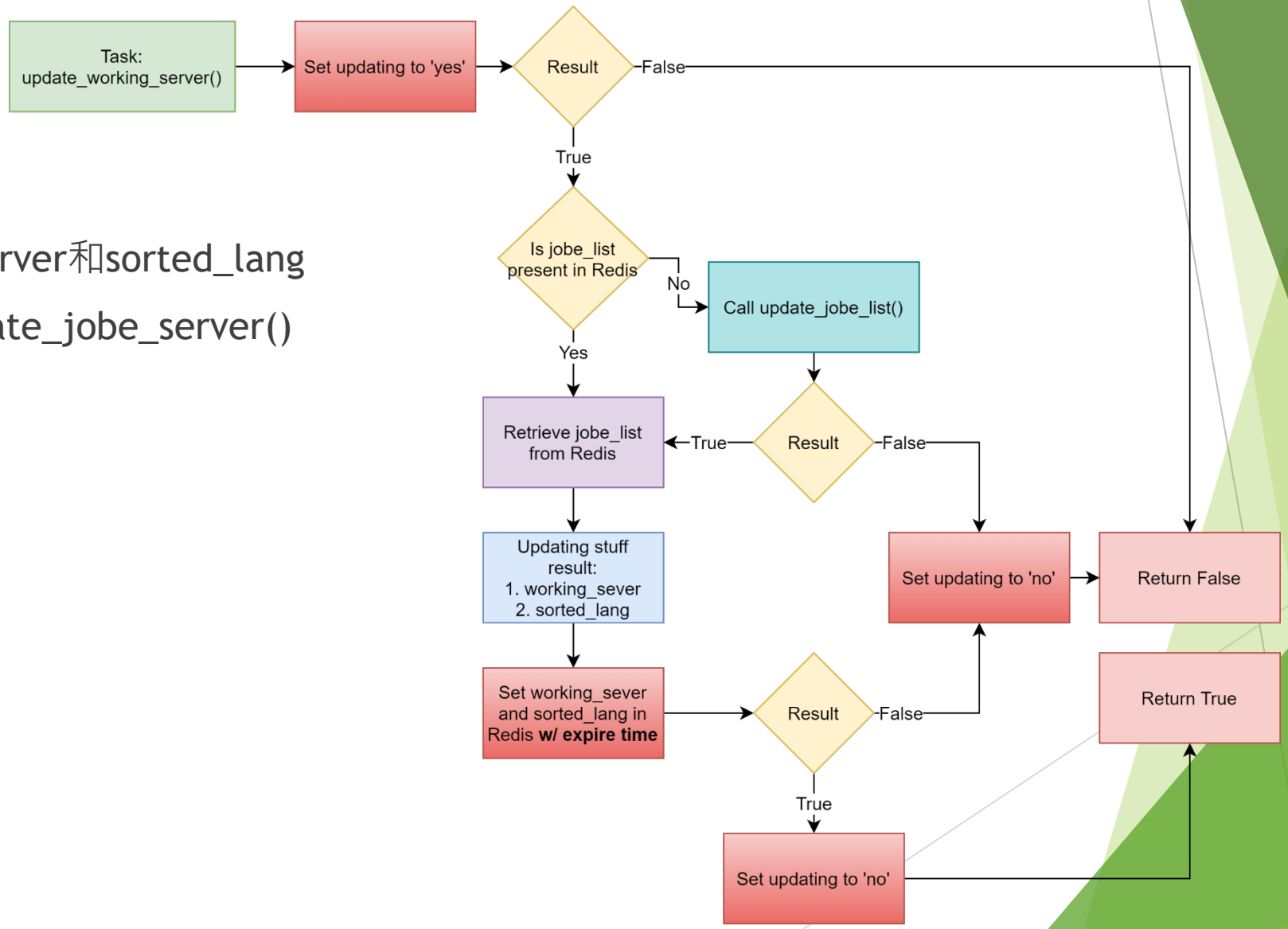
update_jobe_list()

- 更新Redis內儲存的jobe_list



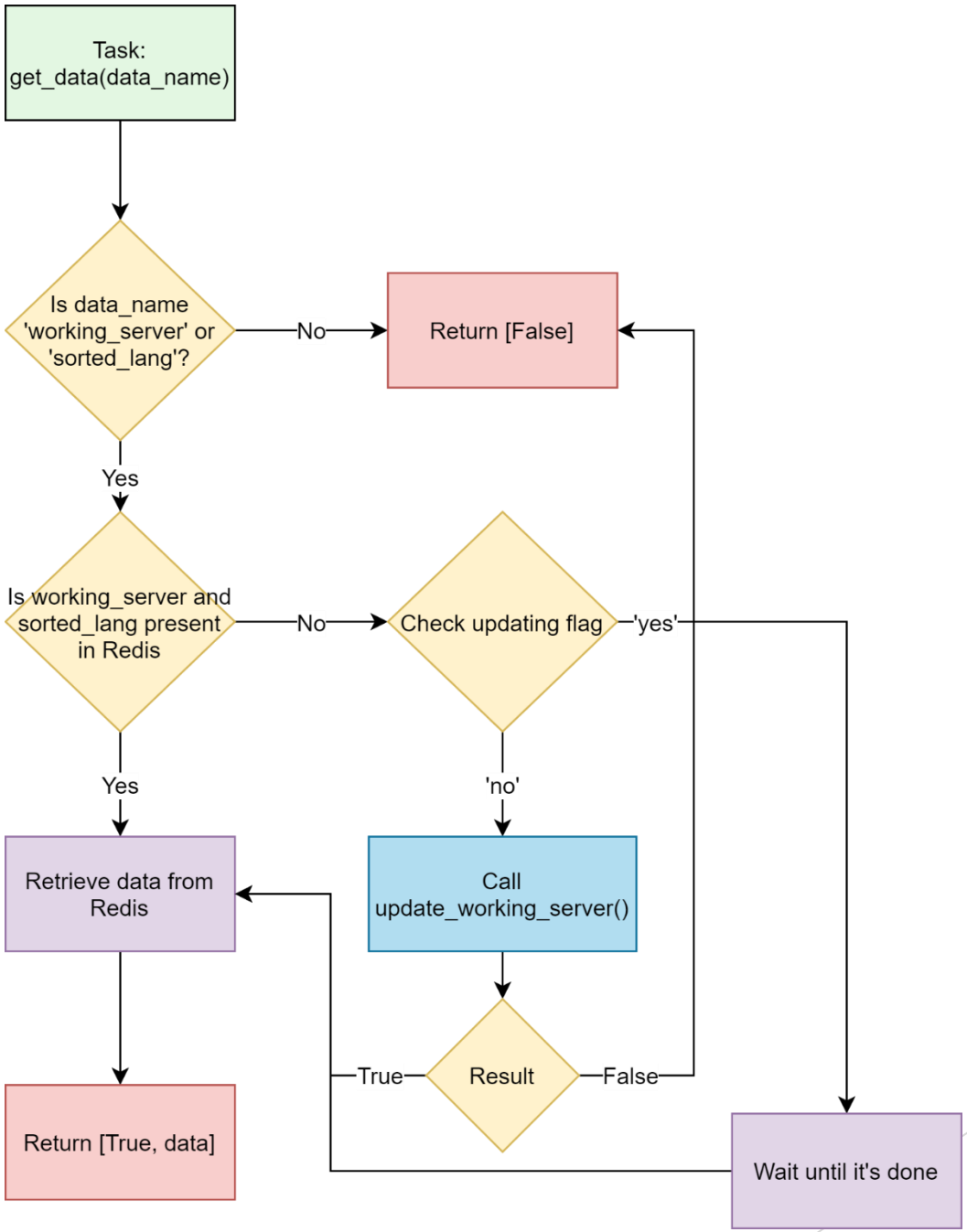
update_working_server()

- 更新working_server和sorted_lang
- 需要時呼叫update_jobe_server()



get_data()

- 要求資料



Gunicorn/Flask端的變動

- 移除generate_working_server()
- force_update()改為呼叫兩個update_ tasks
- 所有開檔案都改成呼叫get_data()