

# 網頁自動化測試

## 從內部工具開始

# Now

S	W	Name ↑	Last Success	Last Failure	Last Duration	Fav
		<a href="#">Upload Test</a>	6 days 23 hr - <a href="#">#12</a>	10 days - <a href="#">#10</a>	13 min	 
		<a href="#">Smart App Test</a>	10 hr - <a href="#">#37</a>	1 mo 1 day - <a href="#">#5</a>	46 sec	 
		<a href="#">Automation Daily Test</a>	14 hr - <a href="#">#94</a>	12 days - <a href="#">#81</a>	1 hr 43 min	 

Daily Upload: 2020/08/08 00:05 ~ 2020/08/08 00:08 - standard files

Upload File Processing (100)

Import Orderlist Processing (100)

Orderlist Count Check (100)

Pass File (76)

# 自我介紹

- Set Mao / 太一
- ORDERLY 奧特圖 Full-Stack Engineer



**setmao**

setmao

Life is short, I use Python.

# Agenda

- 怎麼開始的？
- 人是貪心的，能不能更好用一點？
- 一個你就滿足了嗎？有沒有想過提高測試範圍？
- 這個東西很讚，可以讓它每天自己跑嗎？
- 自動化測試該怎麼跟手動測試合作、溝通？

# 怎麼開始的？

## 一鍵匯入 i

匯入訂單資料，智能辨識平台來源，資料比對與校正，協助您建立訂單、客戶與產品數據中心

🏠 儀表板 / 一鍵匯入

匯入各平台之完整欄位檔案，限 `.xlsx` `.csv` 或 `.xls` [查看支援平台](#)

未選取

 選擇檔案

Mac 系統鍵盤長按 `command` , Windows 系統鍵盤長按 `Ctrl` ，可最多同時選擇 20 個檔案進行批次上傳。

訂單資料 i

# 怎麼確保功能運作正常

1. 選擇並上傳檔案
2. 上傳中
3. 確認並匯入資料
4. 匯入完成
5. 檢視完整資料

# 這樣有什麼問題？

- 無聊、浪費人力（錢）、~~沒有我就只是想玩看看 selenium~~
- 而且隨著支援平台的增加、測試檔案的增加、測試所花的人力成本也會跟著越來越高

**所以，怎麼開始的？**



# How?

- Python + Selenium

```

17 # start a browser
18 test_start_time = time.time()
19 browser = webdriver.Chrome()
20 browser.get(BASE_URL + "/accounts/login/")
21 # browser.implicitly_wait(30)
22
23 # login
24 browser.get(BASE_URL + "/accounts/login/")
25 browser.find_element_by_id("id_login").send_keys(ACCOUNT)
26 browser.find_element_by_id("id_password").send_keys(PASSWORD)
27 browser.find_element_by_id("form-login-btn").click()
28
29 # upload file
30 browser.get(BASE_URL + "/importly/uploadform/")
31 index = 0
32 for files in all_file:
33     file_format = files.split(".")[1]
34     if file_format not in ["xlsx", "xls", "csv"]:
35         continue
36     if files.startswith("~$"):
37         continue
38
39     browser.find_element_by_id("files_field").send_keys(join(FILE_DIRECTORY_PATH, files))
40     upload_button = browser.find_element_by_class_name("fileinput-upload-button")
41     browser.execute_script("arguments[0].click();", upload_button)
42     # browser.find_element_by_class_name("fileinput-upload-button").click()
43
44     start_time = time.time()
45     WebDriverWait(browser, 60, 0.5).until(EC.presence_of_element_located((By.CLASS_NAME, "ui-pnotify-title")))
46     return_text = browser.find_elements_by_class_name("ui-pnotify-title")
47     if return_text:
48         print("index: {} files: {} result: {}".format(index, files, return_text[0].text))
49         print("--- %s seconds ---" % (time.time() - start_time))
50         time.sleep(5.5)
51         index += 1
52
53     while True:
54         import_button = browser.find_elements_by_class_name("button-import")
55         if import_button:
56             import_button[0].click()
57             next_step_button = browser.find_elements_by_class_name("next-step")
58             if next_step_button:
59                 next_step_button[0].click()
60         else:
61             break

```

# Result

1. ~~選擇並上傳檔案~~
2. 上傳中
3. 確認並匯入資料
4. 匯入完成
5. 檢視完整資料

# Result

- 減少了 10~15 分鐘的手動上傳操作時間，但是仍然需要人力比對答案，而且要從 CLI 介面啟動，對沒有程式背景的人來說不那麼友善

**（人是貪心的，） 能不能更好用一點？**

**好用？現在難用在哪？**

# 難用的點

- 只有上傳功能，沒有自動按匯入的功能
- 還是需要人力檢查答案
- 要從 CLI 啟動
- 測試檔案要自己下載

# 難用的點

- 只有上傳功能，沒有自動按匯入的功能 -> 寫
- 還是需要人力檢查答案 -> 只判斷資料筆數就好
- 要從 CLI 啟動 -> 包成 exe
- 測試檔案要自己下載 -> 串 google drive api 直接從雲端下載檔案



# Result

- 測試流程全部自動化了（大約減少 1 時/天）

# Result

- 測試流程全部自動化了（大約減少 1 時/天）
- ~~工讀生也拚拚了~~

**一切都看起來很完美**

**But**

人是貪心的

**一個測試你就滿足了嗎？**

**你有沒有想過提高測試範圍？**

**一切都看起來很完美**



**一切都看起來很完美  
(以一個內部工具來說)**

**那以一個自動化測試來說**

# 在寫扣之外

- 測試文件
- 測試檔案
- 測試流程
- 測試環境

# 在寫扣之前

- 現有的扣有什麼問題？

```

17 # start a browser
18 test_start_time = time.time()
19 browser = webdriver.Chrome()
20 browser.get(BASE_URL + "/accounts/login/")
21 # browser.implicitly_wait(30)
22
23 # login
24 browser.get(BASE_URL + "/accounts/login/")
25 browser.find_element_by_id("id_login").send_keys(ACCOUNT)
26 browser.find_element_by_id("id_password").send_keys(PASSWORD)
27 browser.find_element_by_id("form-login-btn").click()
28
29 # upload file
30 browser.get(BASE_URL + "/importly/uploadform/")
31 index = 0
32 for files in all_file:
33     file_format = files.split(".")[1]
34     if file_format not in ["xlsx", "xls", "csv"]:
35         continue
36     if files.startswith("~$"):
37         continue
38
39     browser.find_element_by_id("files_field").send_keys(join(FILE_DIRECTORY_PATH, files))
40     upload_button = browser.find_element_by_class_name("fileinput-upload-button")
41     browser.execute_script("arguments[0].click();", upload_button)
42     # browser.find_element_by_class_name("fileinput-upload-button").click()
43
44     start_time = time.time()
45     WebDriverWait(browser, 60, 0.5).until(EC.presence_of_element_located((By.CLASS_NAME, "ui-pnotify-title")))
46     return_text = browser.find_elements_by_class_name("ui-pnotify-title")
47     if return_text:
48         print("index: {} files: {} result: {}".format(index, files, return_text[0].text))
49         print("--- %s seconds ---" % (time.time() - start_time))
50         time.sleep(5.5)
51         index += 1
52
53     while True:
54         import_button = browser.find_elements_by_class_name("button-import")
55         if import_button:
56             import_button[0].click()
57             next_step_button = browser.find_elements_by_class_name("next-step")
58             if next_step_button:
59                 next_step_button[0].click()
60         else:
61             break

```

**你有聽過 Page Objects 嗎？**

# Benefits of using page object pattern:

- Creating **reusable** code that can be shared across multiple test cases
- **Reducing** the amount of **duplicated code**
- If the user interface changes, the fix needs **changes in only one place**

# Page Objects 是什麼？

Selenium (Python) 的官方文件中 Page Objects 會包含以下幾種東西

1. Test case
2. Page object classes
3. Page Element
4. Locators



**第一步，寫好測試流程**

# 寫好測試流程

1. 進到儀表板頁面
2. 定位到想要看的圖表
3. 確認圖表中的數字是否正確

**第二步，把網頁切成一塊一塊的**

## 儀表板：總覽

儀表板

總覽

銷售分析

訂單分析

顧客分析

進階

回購分析

顧客活躍分析

儀表板更新! 為提供更佳分析體驗，圖表已被重新排序，若顯示出現問題，請重整頁面。

3,137,150

0 %

累計銷售額

每日平均銷售額：0.0 元

2,244

0 %

累計訂單數

每日平均：0.0 張

1,417

+87.8 %

累計不重複顧客數

每日新增：10.5 人

## 銷售趨勢

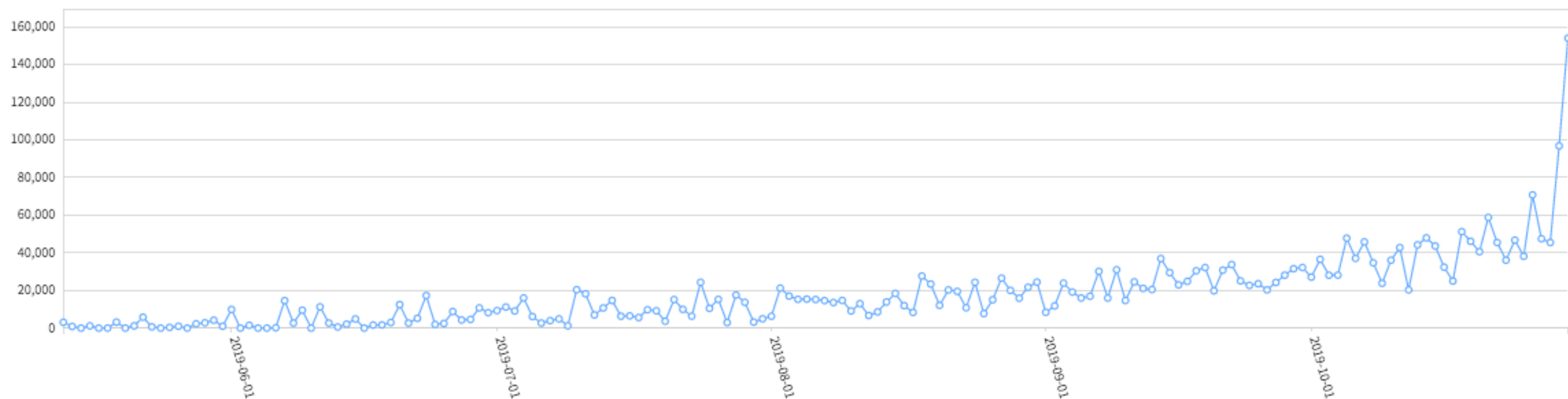
銷售是否活躍反映您的經營能力，趨勢則說明日常經營進展，當圖表出現異常起伏時，要仔細判斷形成的原因

銷售額

銷售量

訂單數

顧客數



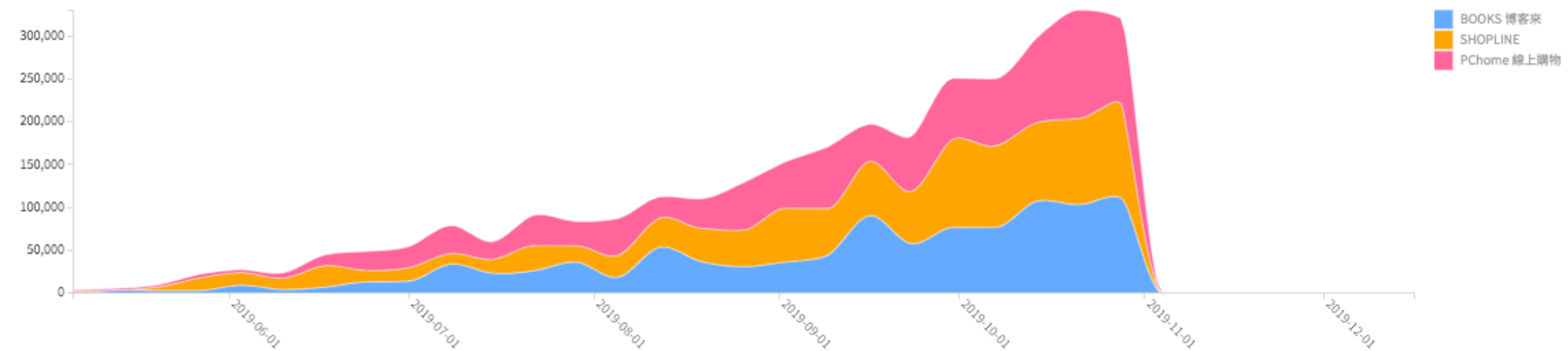
## 周 銷售趨勢（依通路）

銷售額

銷售量

訂單數

顧客數



**第三步，抓到每個要用的元素的位置**

```
▼ <div class="col-md-4">
  ▼ <div class="panel bg-secondary">
    ▼ <div id="team-current-statistics" class="panel-body">
      ::before
      ▶ <div class="heading-elements"> ... </div>
      ▼ <div class="h1 no-margin">
        <span class="text-size-small">$</span>
        <span id="count_sales_all">8,312,863</span>
      </div>
      <div class="h3 no-margin">累計銷售額</div>
      ▶ <div class="text-muted text-size-base"> ... </div>
      ::after
    </div>
    ▶ <div class="container-fluid"> ... </div>
  </div>
</div>
.. - .. - . ... — ...
```

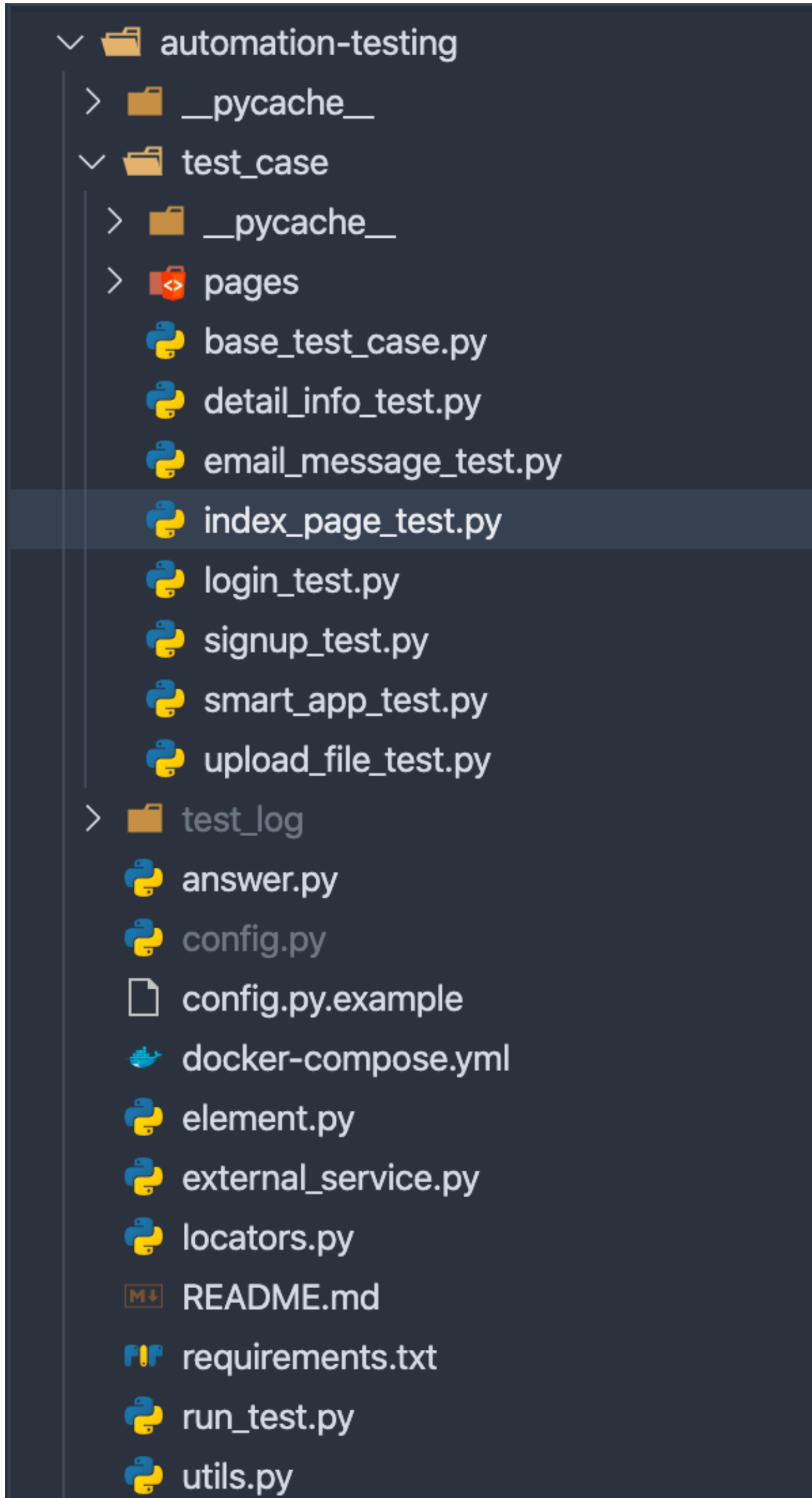
# 替代方案

- Selenium IDE

**第四步，開始寫每個 page 的 func**



# Result



**用 Page Object 改寫完了  
看起來一切都很完美**

**所以我們可以開始寫下一個測試了（嗎？）**

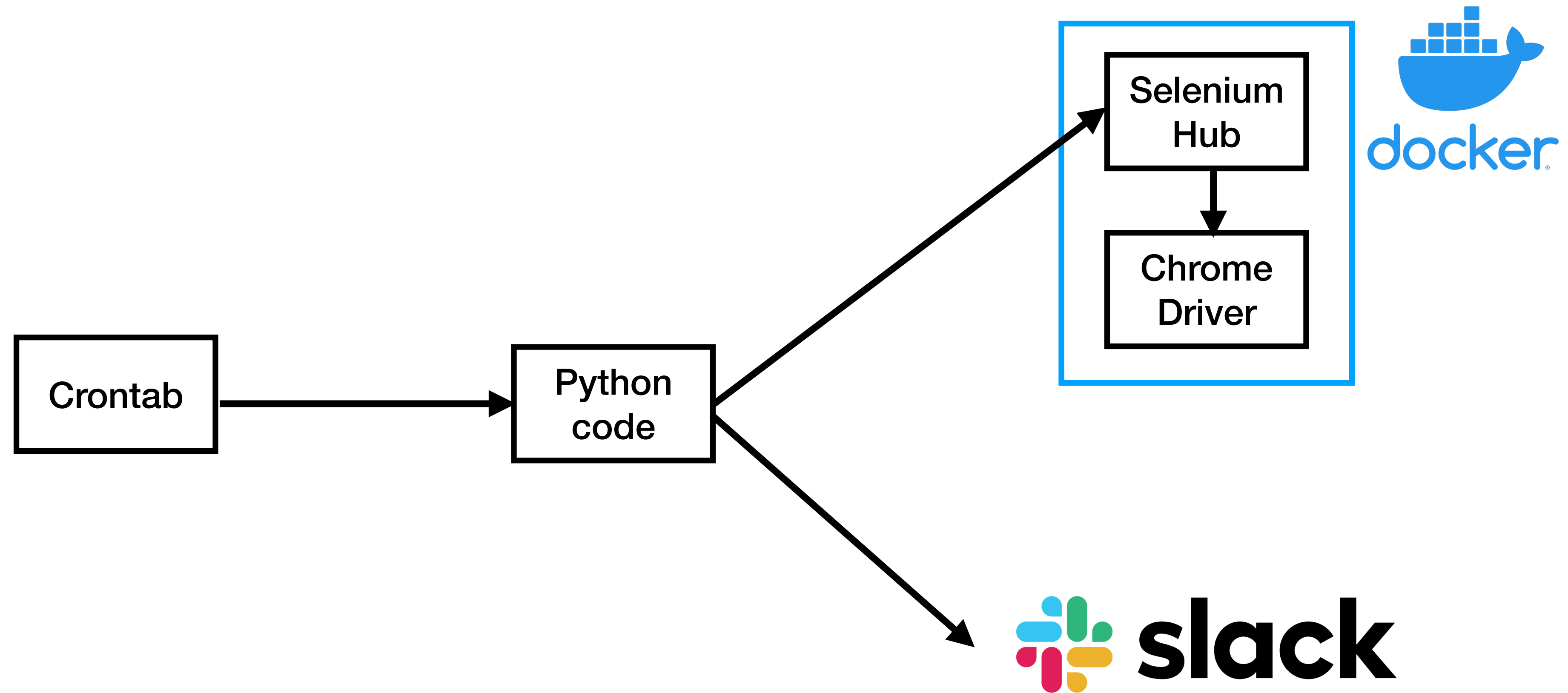
**這個東西看起來很讚，  
但是可以讓它每天自己跑嗎？**

**啊，跑完記得要回傳測試結果喔**

**這個東西看起來很讚，但是可以讓他在 test server 上每天自動跑嗎？**

- crontab
- Selenium hub
- Docker
- Slack

# Result



**手動測試：你們弄了那麼久在弄什麼啊？**



■ ■ ■ ■

**自動化測試該怎麼跟手動測試合作、溝通？**

# 來自手動測試心中的疑惑

- 你們在幹嘛？
- 這個怎麼用？

# 來自手動測試心中的疑惑

- ~~你們在幹嘛？~~
  1. 你們做了哪些 Test case ？
  2. 測試覆蓋度多大？
  3. 可以教我怎麼操作自動化測試嗎？

# 來自手動測試心中的疑惑

- ~~這個怎麼用？~~
  1. 我要怎麼使用自動化測試？
  2. 我要怎麼知道執行結果是成功還是失敗？
  3. 我要怎麼知道是哪個測試失敗了？

# 你們在幹嘛？

- 教他看 Code ？

# 你們在幹嘛？

- ~~教他看 Code？~~
- 寫文件？

# 你們在幹嘛？

- ~~教他看 Code？~~
- ~~寫文件？~~
- Gherkin (Python: Behave)



# Gherkin

- Given I have a correct account and password
- When I login
- Then I should login success

# Behave

```
@browser
Feature: Login # features/login.feature:2
  Can Login success
  Scenario: Login # features/login.feature:6
    Given I have an account and password # features/steps/login.py:6 0.000s
    When I login # features/steps/login.py:18 2.939s
    Then I should redirect to team page # features/steps/login.py:30 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m2.940s
```

# Result













- Before: Test Case -> Page -> Locator/Element
- After: Feature -> Step -> Page -> Locator/Element

這個怎麼用？

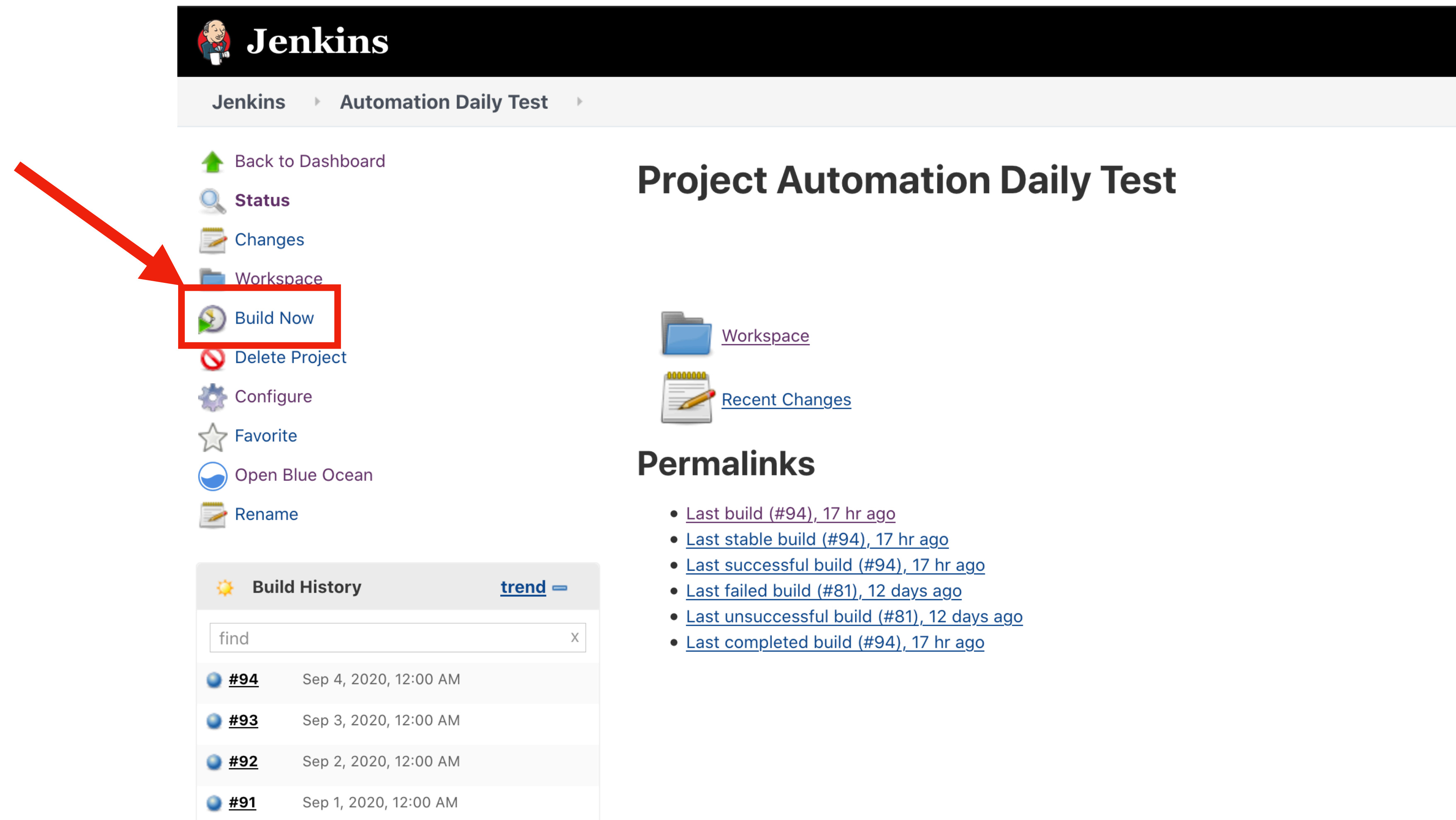


# Jenkins

# 這個怎麼用？

S	W	Name ↑	Last Success	Last Failure	Last Duration	Fav
		<a href="#">Upload Test</a>	6 days 23 hr - <a href="#">#12</a>	10 days - <a href="#">#10</a>	13 min	 
		<a href="#">Smart App Test</a>	10 hr - <a href="#">#37</a>	1 mo 1 day - <a href="#">#5</a>	46 sec	 
		<a href="#">Automation Daily Test</a>	14 hr - <a href="#">#94</a>	12 days - <a href="#">#81</a>	1 hr 43 min	 

# 這個怎麼用？



The screenshot shows the Jenkins interface for the 'Automation Daily Test' project. The left sidebar contains several action buttons: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now' (highlighted with a red box and a red arrow), 'Delete Project', 'Configure', 'Favorite', 'Open Blue Ocean', and 'Rename'. The main content area displays the project name 'Project Automation Daily Test' and links to 'Workspace' and 'Recent Changes'. Below this is a 'Permalinks' section with a list of build links. At the bottom, there is a 'Build History' table showing the last four builds.

## Project Automation Daily Test

[Workspace](#)

[Recent Changes](#)

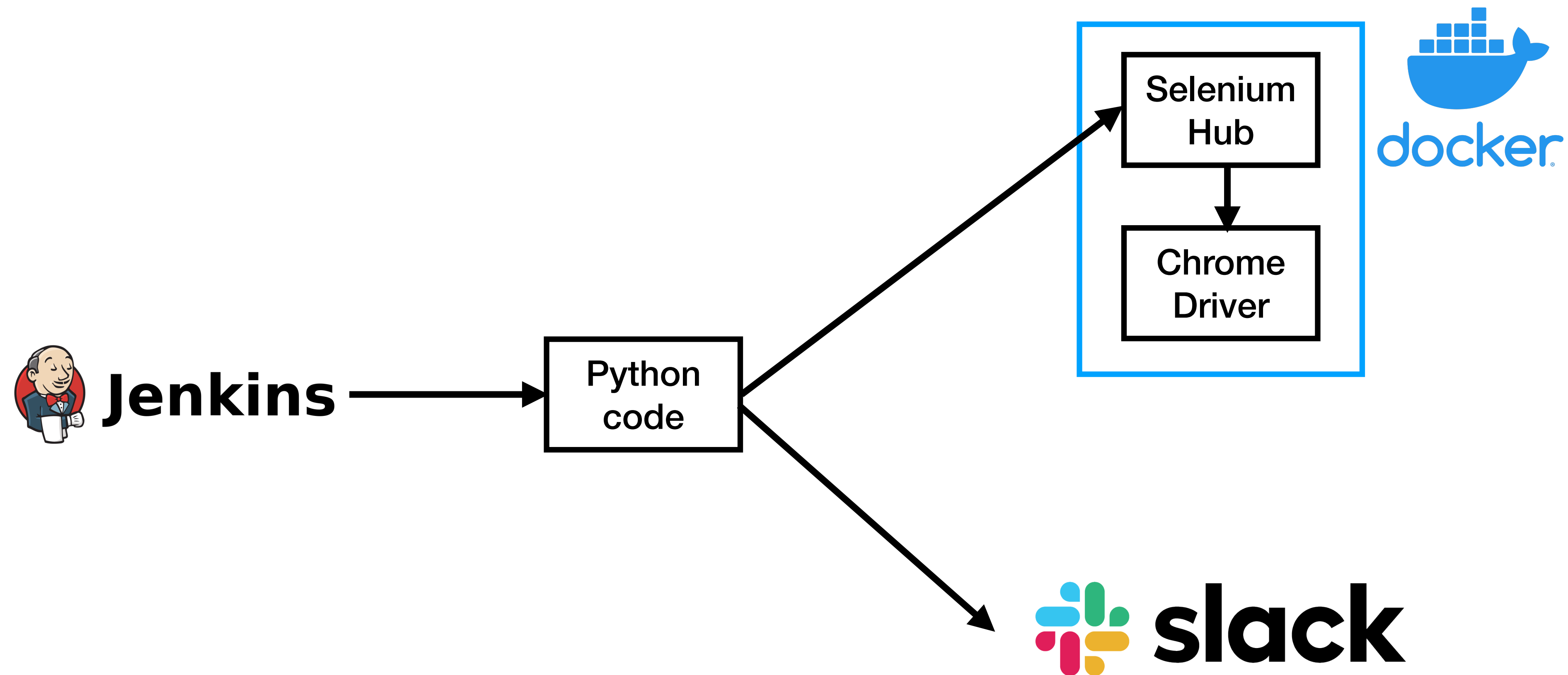
### Permalinks

- [Last build \(#94\), 17 hr ago](#)
- [Last stable build \(#94\), 17 hr ago](#)
- [Last successful build \(#94\), 17 hr ago](#)
- [Last failed build \(#81\), 12 days ago](#)
- [Last unsuccessful build \(#81\), 12 days ago](#)
- [Last completed build \(#94\), 17 hr ago](#)

#### Build History

<a href="#">#94</a>	Sep 4, 2020, 12:00 AM
<a href="#">#93</a>	Sep 3, 2020, 12:00 AM
<a href="#">#92</a>	Sep 2, 2020, 12:00 AM
<a href="#">#91</a>	Sep 1, 2020, 12:00 AM

# Result



**Thanks**



**Q & A**