

由使用者更新檔案 `commit`，就會自動建置更新，及相關動作。
(觸發/建置/建置後)

- > 「持續整合」
- > 「持續部屬」

階段功能：

1. GitHub 同步建置
2. 使用 token 建置
3. Jenkins 自動 build
4. 自動 Send Email
5. 自動發 Slack 通知

所需軟體工具：

1. Docker
2. Jenkins
3. ngrok
4. GitHub
5. Github desktop
6. Slack

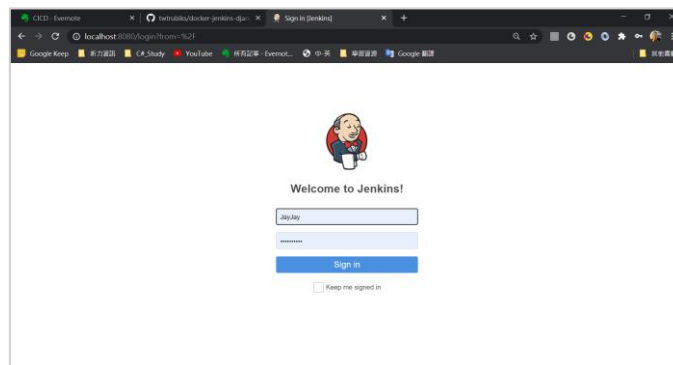
整合 Jenkins + GitHub

前提：

> Jenkins 需先安裝完畢。(可由 Docker 安裝)

> GitHub 需先建立帳號。

Jenkins 建立成功，可直接瀏覽 <http://localhost:8080/>，如下圖：



步驟：

1. 新增外掛程式：GitHub Integration Plugin：

> 點擊「管理 Jenkins」選「管理外掛程式」。



> 安裝 GitHub Integration Plugin。

過濾條件:

更新 可用的 已安裝 進階

安裝 ↓	名稱	版本
<input checked="" type="checkbox"/>	GitHub Integration Plugin GitHub Integration Plugin for Jenkins	0.1.0-rc27

直接安裝 下載並於重新啟動後安裝 Update information obtained: 1 天 0 時 ago 馬上檢查

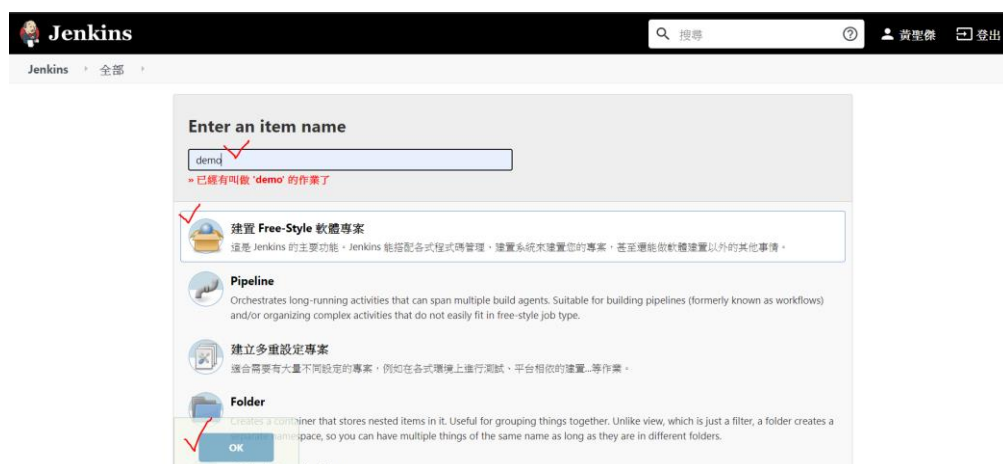
2. 建立一個專案：

> 於左側點選「新增作業」。



The screenshot shows the Jenkins dashboard. At the top is the Jenkins logo and a dropdown menu. Below it is a list of links: '新增作業' (Add Job) with a red checkmark, '使用者' (Users), '建置歷程' (Build History), '管理 Jenkins' (Manage Jenkins), '我的視景' (My Views), 'Lockable Resources', and 'New View'. Below these links are two sections: '建置佇列' (Build Queue) showing '佇列中沒有建置作業。' (No build jobs in queue), and '建置執行程式狀態' (Build Execution Status) showing a list of two items, both labeled '閒置' (Idle).

> 點選自行需要的建置。



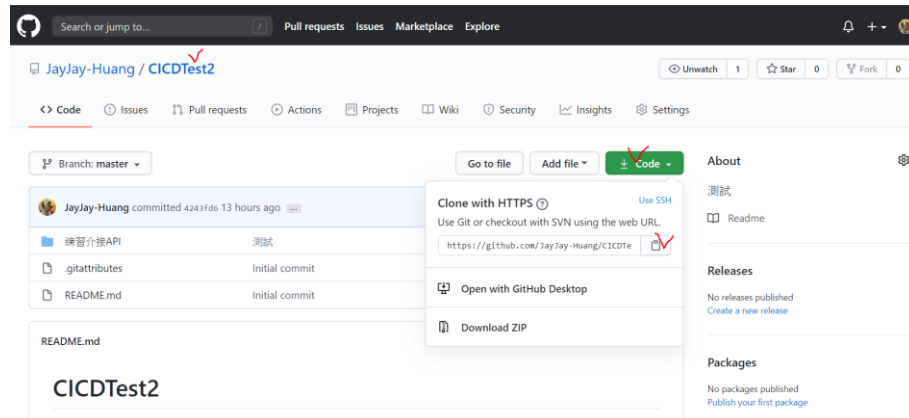
The screenshot shows the 'Enter an item name' dialog in Jenkins. The input field contains 'demo' with a red checkmark. Below the input field is a red error message: '已有叫做 'demo' 的作業了' (A job named 'demo' already exists). Below the error message are four options: '建置 Free-Style 軟體專案' (Build Free-Style Software Project) with a red checkmark, 'Pipeline', '建立多重設定專案' (Build Multi-Configuration Project), and 'Folder'. The 'Folder' option is also checked. At the bottom are 'OK' and 'Cancel' buttons.

> 由此便可對新增的專案進行編制。

3. 新增 Git

> 於 GitHub 先新增好 repository 。(這裡名為 CICDTest2)

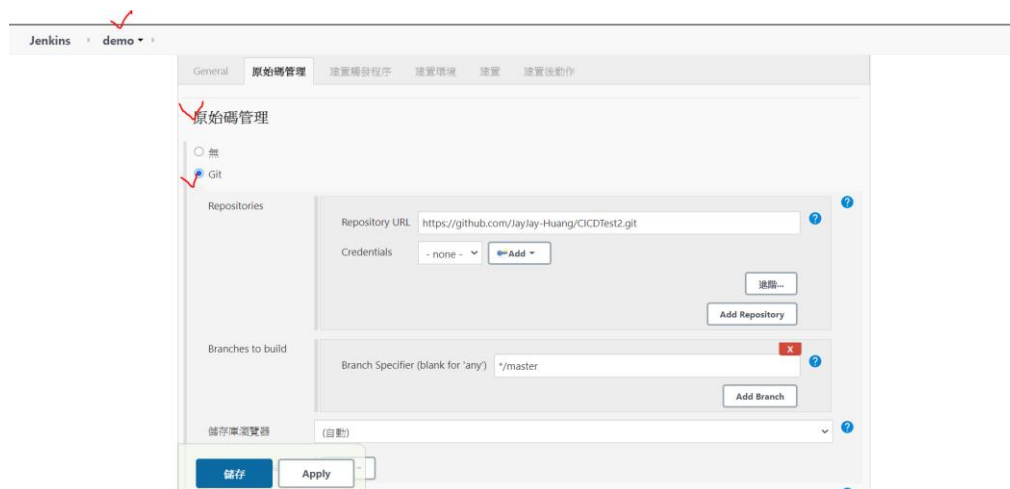
> 然後 Clone with HTTPS (或複製瀏覽器網址) 。



> 於 Jenkins 的 demo 專案中找尋「原始碼管理」，點「Git」。

> 貼上 GitHub 複製的網址。

> 儲存專案。



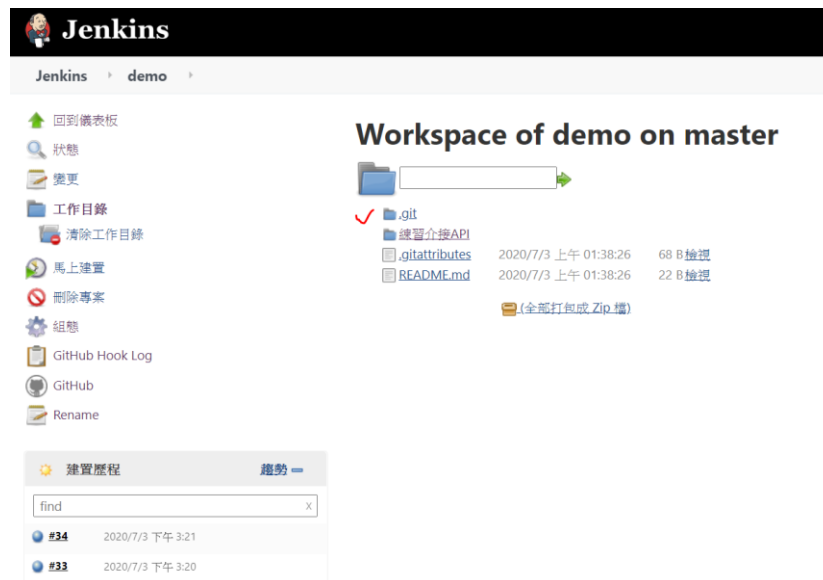
4. 驗證有無成功：

> 至 demo 專案點擊「馬上建置」。

> 可查看「建置歷程」：藍色表示成功，紅色表示失敗。(時間即當下時間)



>或至工作區域查看：可以看見跟 GitHub 一樣的資料。



使用 token 建置

1. 新增 token：

>Jenkins 選 demo 專案，點「組態」。(便可進入編輯)



> 找尋「建置觸發程序」：勾選「遠端觸發建置」。

說明：

這裡可自訂「token」名稱；所定義的名稱要放在執行網址的後方。Ex:

驗證 token: JayJayTestToken

執行網址 <http://localhost:8080/job/demo/build?token=JayJayTestToken>

> 按下專案儲存。



> 下方的「GitHub hook trigger for GITScm polling」、「定期建置」不用勾選。

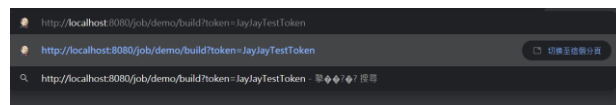
(非本次功能)

> GitHub hook trigger for GITScm polling：為自動建置設定需要。

> 定期建置：為自動定時排程設定需要。

2. 驗證有無成功：

> 至瀏覽器貼上 token。



> 至 demo 專案查看「建置歷程」，會發現已經建置。



使用 GitHub Webhooks 觸發

前提：

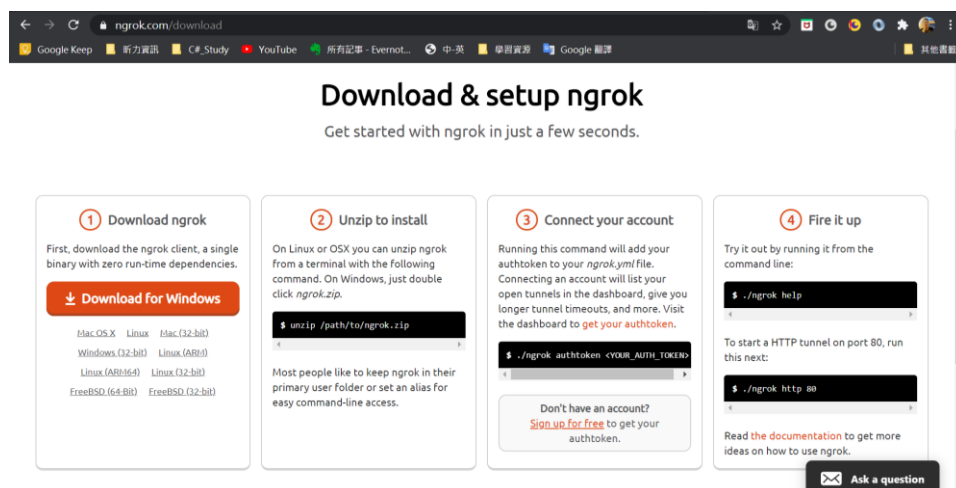
> 需要 public URL。(Jenkins 預設的 <http://localhost:8080/> 為本地端，所以不行。)

> 故需要借用 ngrok。

1. 下載 ngrok.exe：

> 可至 google 搜尋「ngrok download」便可找到網站，上面也有相關教學。

> 下載自行電腦版本。



> 點擊下載好的 ngrok.exe，並輸入「ngrok http 8080」，會出現所需的 url。

(補充：輸入 ngrok http 8080，是因最初是設 8080，最初設定什麼這邊就輸入相對應的)

> 複製產出的網址。

```
C:\ngrok.exe - ngrok http 8080
ngrok by @inconshreveable

Session Status      online
Account             JayJay (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://ec80dae5f749.ngrok.io -> http://localhost:8080
                    ✓ https://ec80dae5f749.ngrok.io -> http://localhost:8080

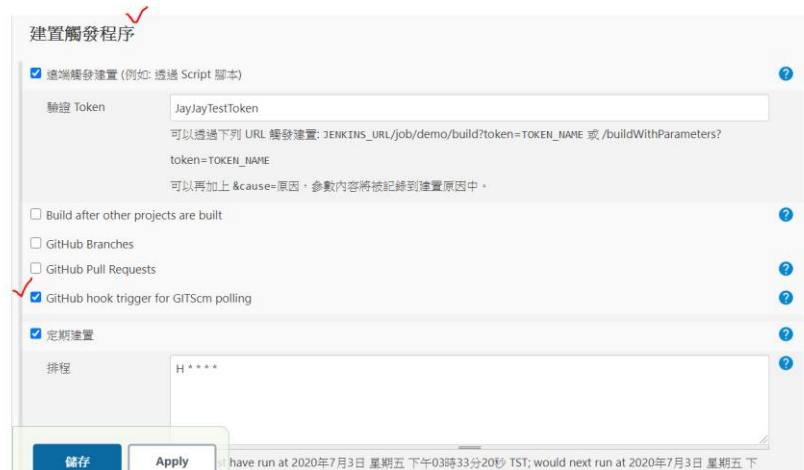
Connections          ttl    opn    rt1    rt5    p50    p90
                   6      0      0.00   0.00   5.02   5.99

HTTP Requests
-----
POST /github-webhook/ 200 OK
POST /github-webhook/ 200 OK
POST /github-webhook/ 200 OK
POST /github-webhook/ 200 OK
POST /github-webhook/ 200 OK
```

2. Jenkins 新增

>至 Jenkins 的 demo 專案，選「組態」，找尋「建置觸發程序」。

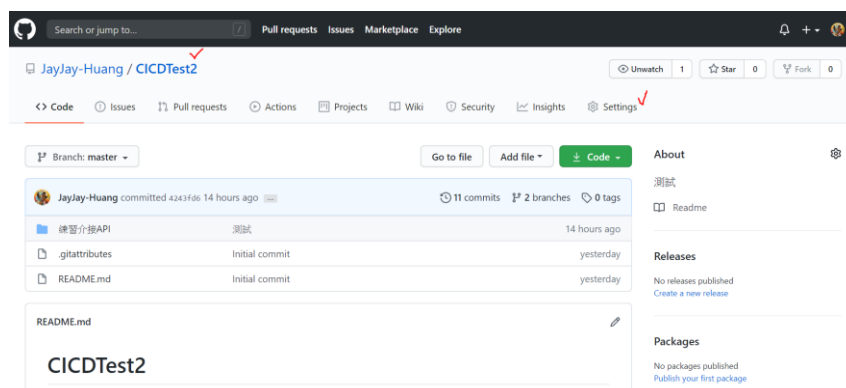
>勾選「GitHub hook trigger for GITScm polling」，並儲存專案。



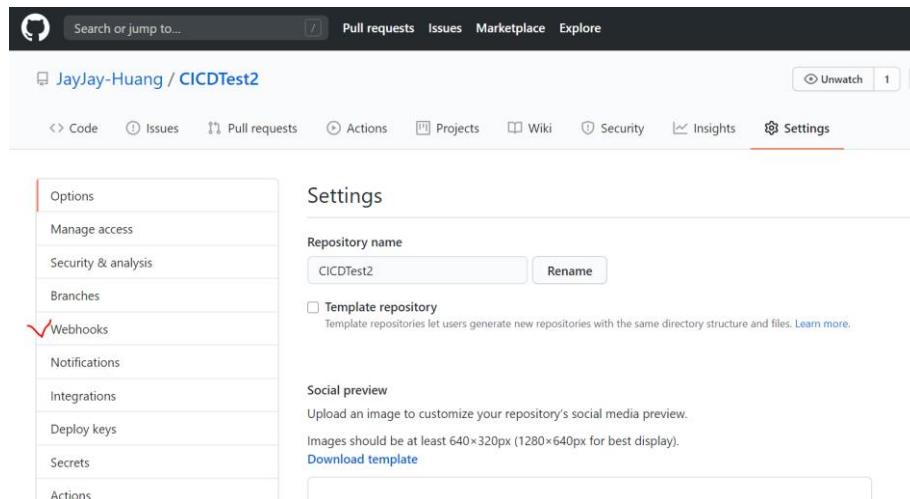
>其它的「遠端觸發建置」、「定期建置」不用勾選。(非本次功能)

3. 設定 Webhooks

>打開 GitHub，進入相對的 repository，點選「Setting」。

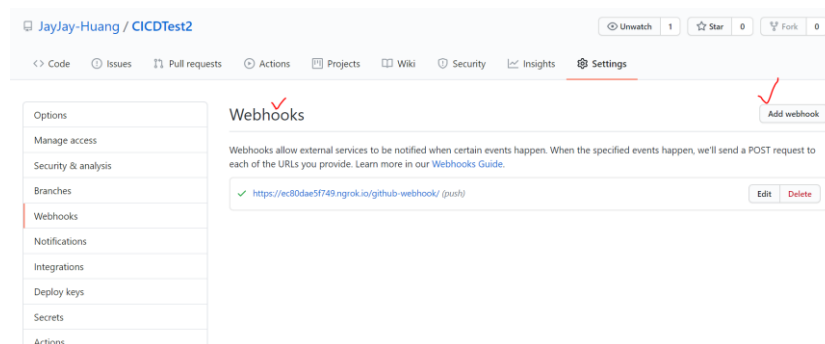


> 點選「Webhooks」



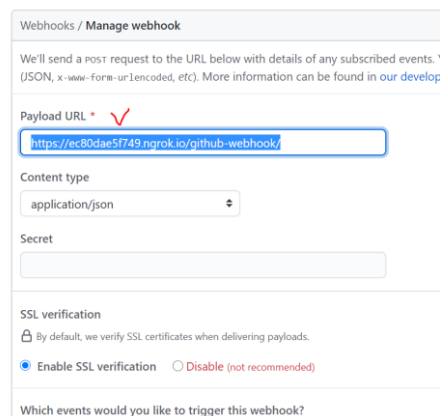
> 進入 Webhooks 後，點「Add Webhook」。

(ps：此處底下為之前新增的 Webhook，一般初次進來應為空的。)

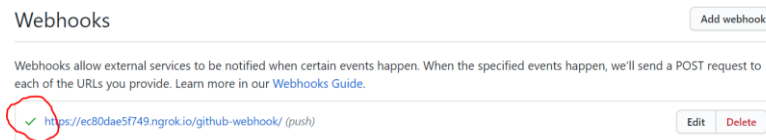


> Payload URL：貼上 ngrok 產生的網址，並於後面補上「/github-webhook」。

> Content 格式選「json」。

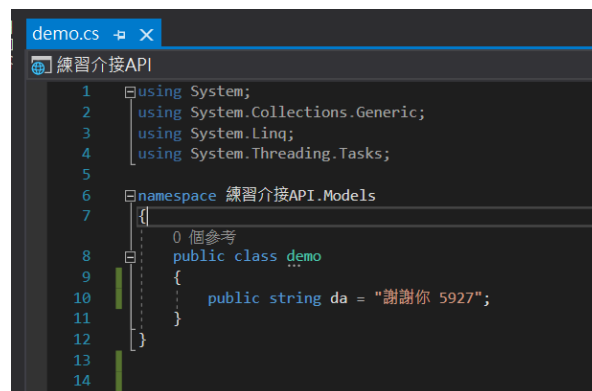


> 儲存設定，回到 Webhooks。出現綠勾代表新增成功。

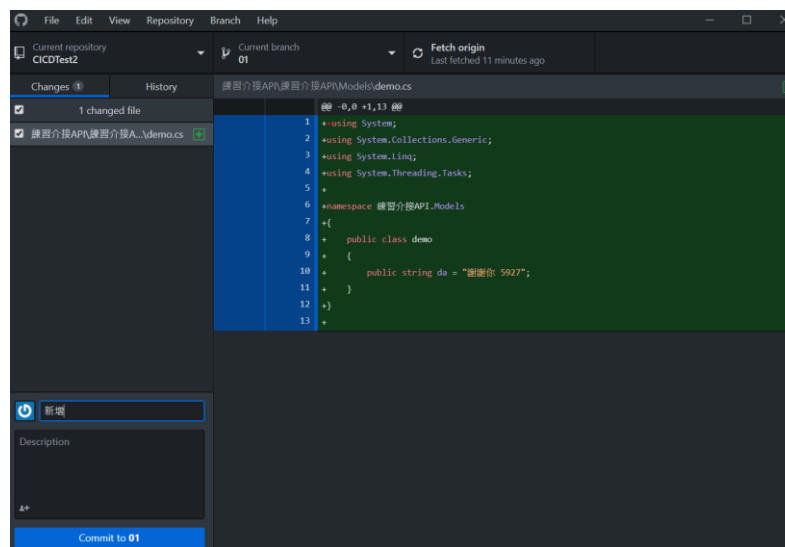


4. 驗證有無成功：

> 於程式中新增修改。



> Github desktop 「commit+push」上 Github。



> 完成後 Github 上會出現新增的資料。

Branch: master CI/CDTest2 / 練習介接API / 練習介接API / Models / demo.cs

黃聖傑 Jay Huang 新增

0 contributors

13 lines (11 sloc) | 218 Bytes

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace 練習介接API.Models
7 {
8     public class demo
9     {
10         public string da = "謝謝你 5927";
11     }
12 }
13
```

>恭喜你!!藉此同時，Jenkins 也已同時 build 好了。

Jenkins

Jenkins demo

回和儀表板

狀態

變更

工作目錄

概況摘要

系統專案

組態

GitHub Hook Log

GitHub

Rename

建置歷程

搜尋

find

✓ #38 2020/7/3 下午 5:15

✓ #38 2020/7/3 下午 5:00

✓ #37 2020/7/3 下午 4:33

Workspace of demo on master

練習介接API / 練習介接API / Models /

BackInfo.cs

demo.cs

JobInfo.cs

MaskInfo.cs

2020/7/3 上午 01:38:26

2020/7/3 下午 05:15:54

2020/7/3 上午 01:38:26

2020/7/3 上午 01:38:26

316 B 檢視

231 B 檢視

1.26 KB 檢視

585 B 檢視

全部訂製成 Zip 檔

localhost:8080/demo/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace 練習介接API.Models
{
 public class demo
 {
 public string da = "謝謝你 5927";
 }
}

自動發信 Send Email

1. 新增系統設定

> 於 Jenkins 「管理 Jenkins」進入「設定系統」，選「電子郵件通知」點「進階」。

The screenshot shows the 'Configure System' page in Jenkins, specifically the 'Email Notification' section under the 'Advanced' tab. The settings are as follows:

- SMTP 伺服器: smtp.gmail.com
- 預設使用者信箱後綴字串: (empty)
- ☒ Use SMTP Authentication
- 使用者名稱: hankjaymail@gmail.com
- 密碼: Concealed (with a 'Change Password' button)
- 使用 SSL: ☒
- Use TLS: ☐
- SMTP 連接埠: 465
- 回信信箱: (empty)
- 字元集: UTF-8

At the bottom, there is a checkbox for '寄測試信，看看設定正不正確' which is currently unchecked.

>

SMTP 伺服器：smtp.gmail.com。

勾「Use SMTP Authentication」使用 SMTP 驗證。

使用者名稱：個人 Gmail 信箱。

密碼：登入 Gmail 密碼。(此處因 Google 限制，必須用「應用程式專用密碼」。)

勾「使用 SSL」。

SMTP 連接埠：465。

字元集：UTF-8。

補充：關於「應用程式專用密碼」

(1).Google 設定

<https://myaccount.google.com/people-and-sharing>

(2).產生「應用程式專用密碼」

(3).授權存取 Google 帳

戶:<https://www.google.com/accounts/DisplayUnlockCaptcha>

(通常會卡在(3)，(1)跟(2)大家比較沒問題，(3)由網址點進去勾選即可)

//我的參考資源

<https://youtu.be/D2F4rFC7tMk>

<https://reurl.cc/exdjLK>

2. 指定收件信箱及內容。

> 於 Jenkins 專案，點選「組態」，找尋「建置後動作」。

> 於建置後動作中，點選「電子郵件通知」。

(另有「可編式電子郵件」，大同小異這邊先略過)

> 填入收件信箱，及勾選個人所需設定。並儲存專案。

3. 驗證有無成功：

> 重複一次程式的新增修改，並 commit+push。

> 至收件信箱查看：

Icon	Subject	Content Preview	Time
📧	還沒設定地址	Jenkins 建置回復成正常: demo #25 - 查看 <http://localhost:8080/job/demo/25/display/redirect>	上午8:43
📧	還沒設定地址	Jenkins 建置回復成正常: demo #13 - 查看 <http://localhost:8080/job/demo/13/display/redirect>	上午5:32
📧	還沒設定地址	Jenkins 建置失敗: demo #12 - 查看 <http://localhost:8080/job/demo/12/display/redirect> 變更: -----...	上午5:32
📧	還沒設定地址	Jenkins 建置失敗: demo #11 - 查看 <http://localhost:8080/job/demo/11/display/redirect> 變更: -----...	上午5:31
📧	還沒設定地址	測試信 #6 - 這是中 6 寄出的測試信 #6	上午5:23
📧	還沒設定地址	測試信 #5 - 這是中 5 寄出的測試信 #5	上午5:23

> 有收到信件，即完成。

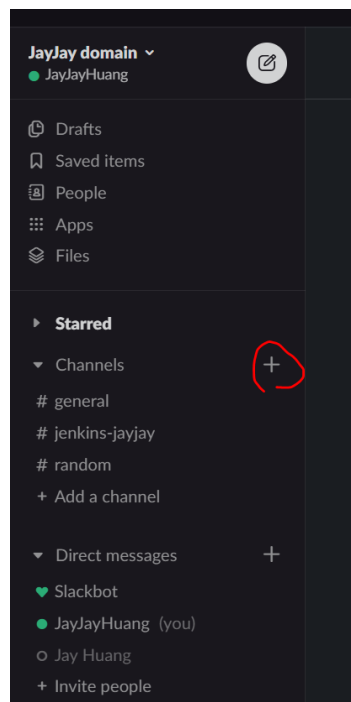
自動通知 Slack

前提：

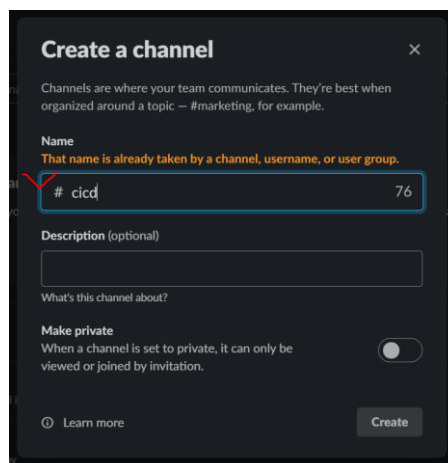
> 要有 Slack 帳號。

1. Slack 建立一個 channel

> 於 Slack 左側點選新增「Channels」。

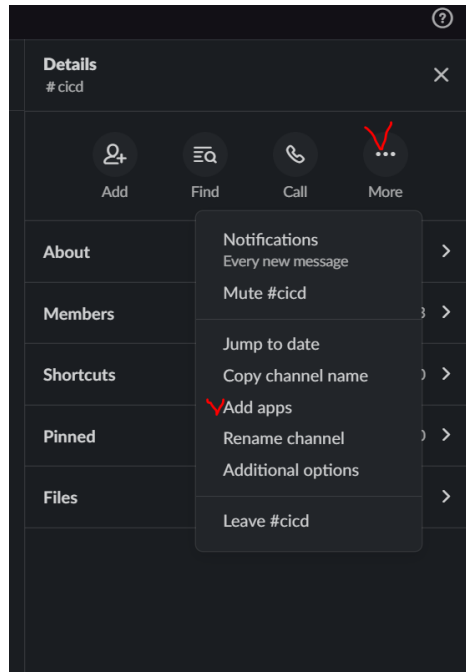


> 自定義一個名稱 (這裡用 cicc)。

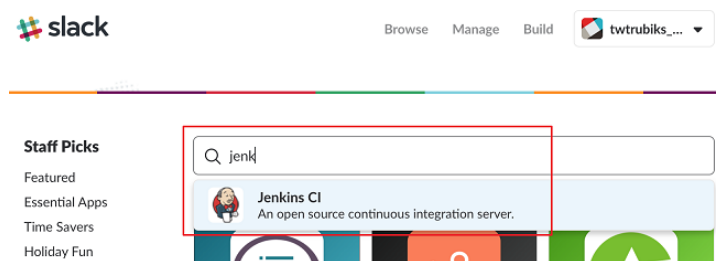


2. Slack 新增 jenkins

> 於所建立的 Channels，點選左邊「Details」。點選「More」在選「Add apps」。



> 搜尋並新增 jenkins。



3. 於 Jenkins 新增 token。

> 於 Jenkins 專案，點選「組態」，找尋「建置後動作」。

> 於建置後動作中，點選「Slack」。

輸入，並貼上 token。


Slack

Workspace ?

Credential ?

Default channel / member id ?

Custom slack app bot user ☐ ?

 **Jenkins Credentials Provider: Jenkins**

Add Credentials

Domain

Kind

Scope

Secret

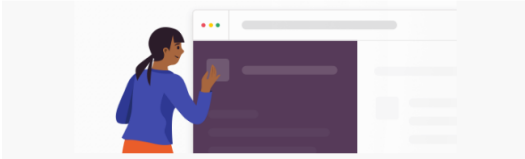
ID

Description

> 備註

Jenkins 的 token 輸入，要參考 Slack 新增時的輸入。

Review your team's details



Slack workspace name

Most teams use the name of their company or organization.

Slack URL

You'll use this URL to sign in to Slack. Letters, numbers, and hyphens only.

第1步，共2步

4. 設定建置後發 Slack：

> 回到 jenkins 專案，點選「組態」。

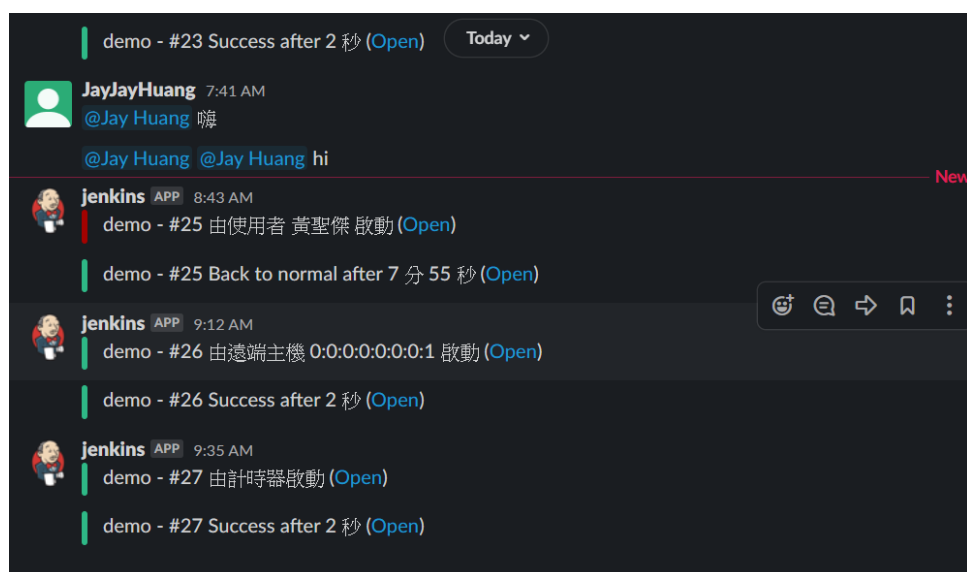
> 於「建置後動作」勾選「Slack Notifications」，並儲存專案。



5. 驗證有無成功：

> 重複一次程式的新增修改，並 commit+push。

> 至 Slack 查看：



> 有收到通知即成功。