

DevOps 自動化測試

Jenkins帳號:

https://docs.google.com/spreadsheets/d/1KGnO8vX6RDfjLeLEBtUM5B0gA5rcii-cliCbU1_krAg/edit?usp=sharing

需求文件:

<https://docs.google.com/document/d/1cCkd94TrRITmbrUMRkeXPUBM7NYA8k2Do2dl9gvgetE/edit?usp=sharing>



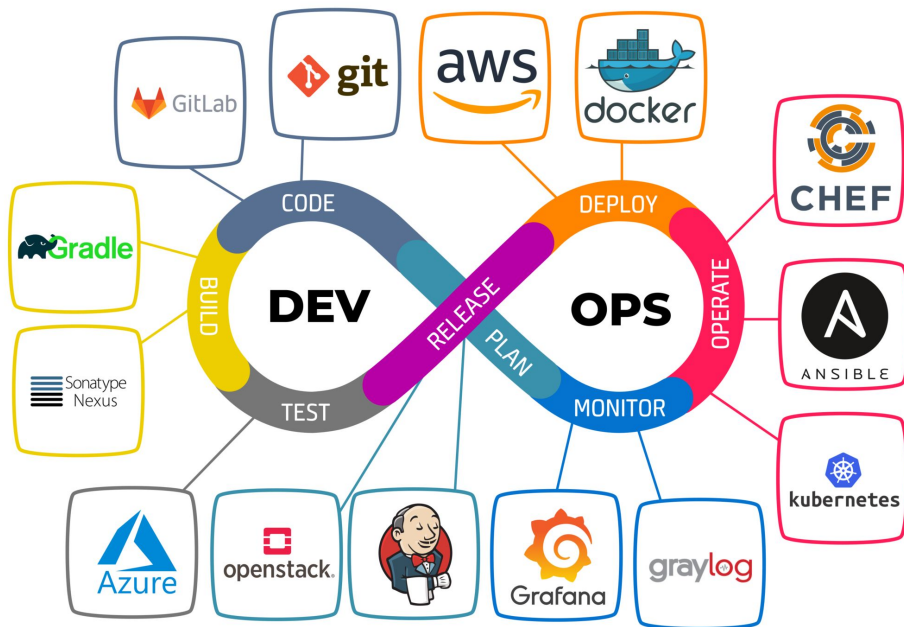
<https://reurl.cc/7r2EW9>

Outline

- DevOps 與 整合測試工具 (10 mins)
 - Gradle 專案架構 (5 mins)
 - 建立與設定自動化測試工作 (50 mins)
 - 撰寫測試與測試流程 (40 mins)
 - 觀察涵蓋率 (20 mins)
-
- 觀察程式碼品質 (30 mins)
 - 改善程式碼品質 (50 mins)

DevOps

DevOps, 是一種將開發 (Development) 和維運 (Operations) 融為一體的文化改革
且經常使用自動化工具來促進合作改革



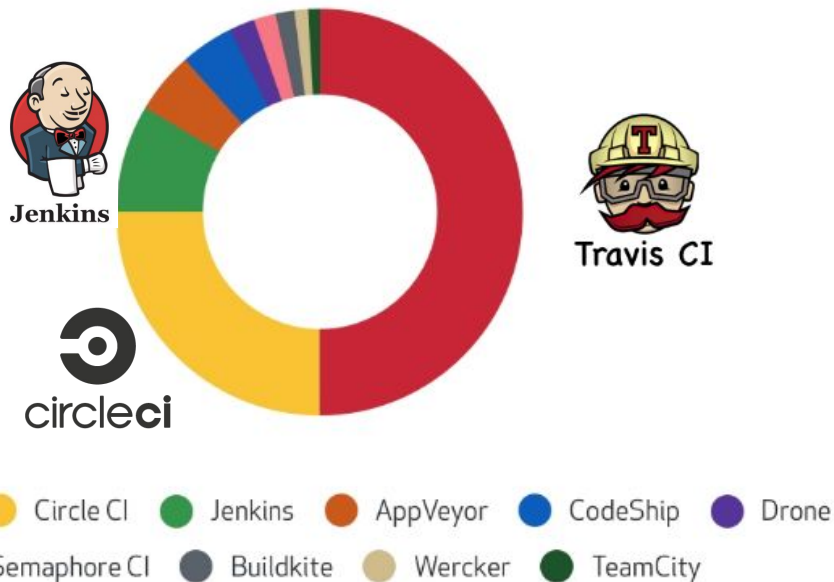
常用的持續整合工具

持續整合工具可以協助 **團隊** 流暢的進行整合測試, 通常工作在比單元測試更大的範圍上。

主要針對開發/修復的分支合併等狀況, 替不同開發者貢獻的整合提供一個保險準繩 (至少可以編譯通過, 不會破壞功能...)

另外, 也可以針對不同的環境進行大量的重複測試, 例如在不同作業系統與依賴版本上的交叉測試

也提供直接部署 (Deploy) 到目標伺服器的能力, 團隊可以根據情況使用

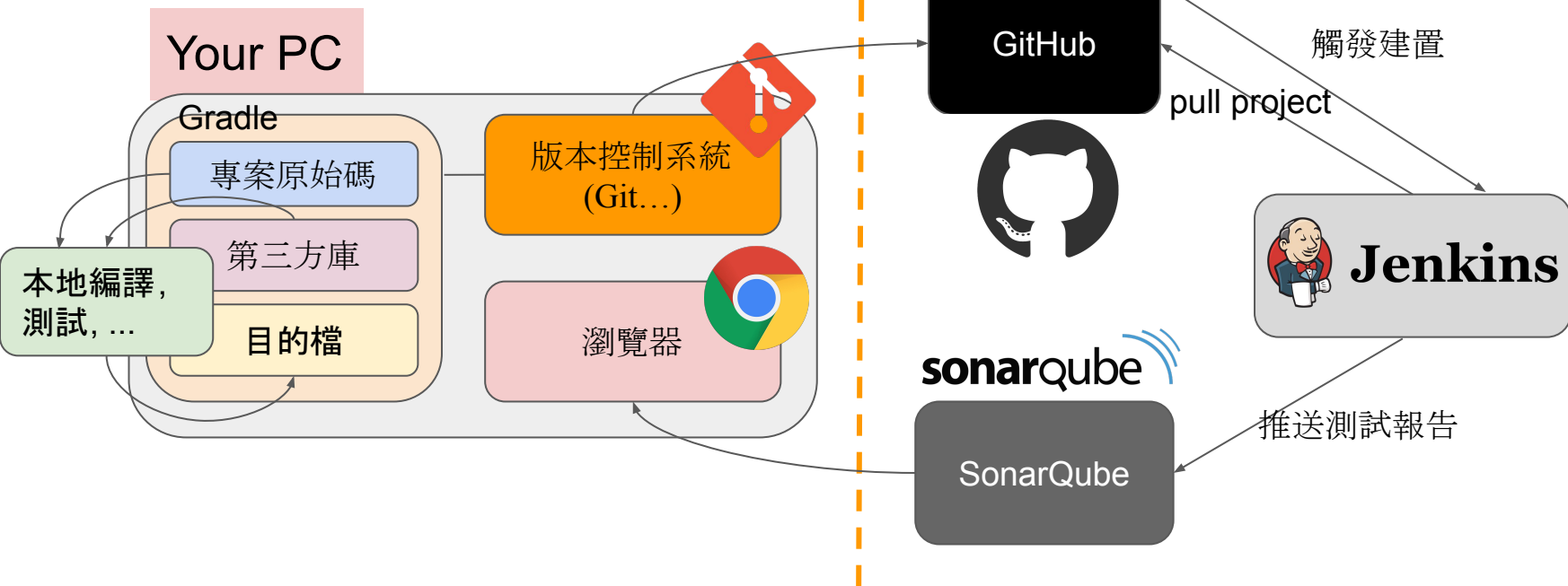


Github 統計在 2017 年 Top 10 的 CI 工具使用率

自動化測試 流程圖

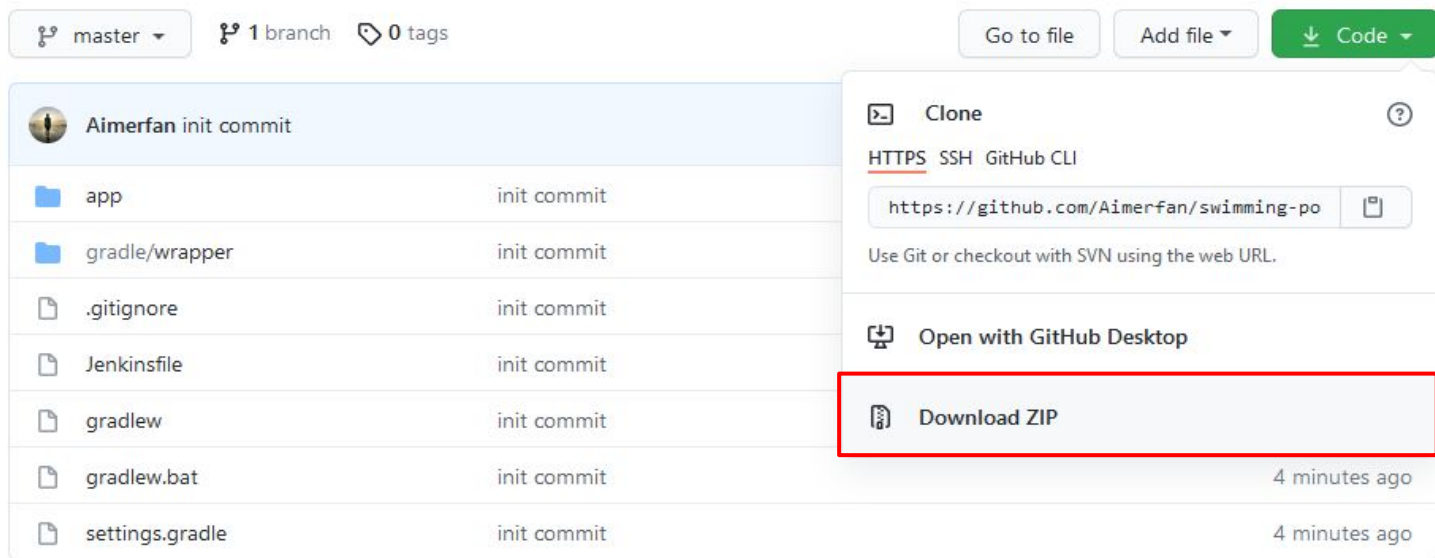
Local (Host)

Cloud (remote)



範例程式碼

- 先下載 [範例程式碼](#) 至本地資料夾，並解壓縮



建立 GitHub 專案並 push

使用 Git 的方法

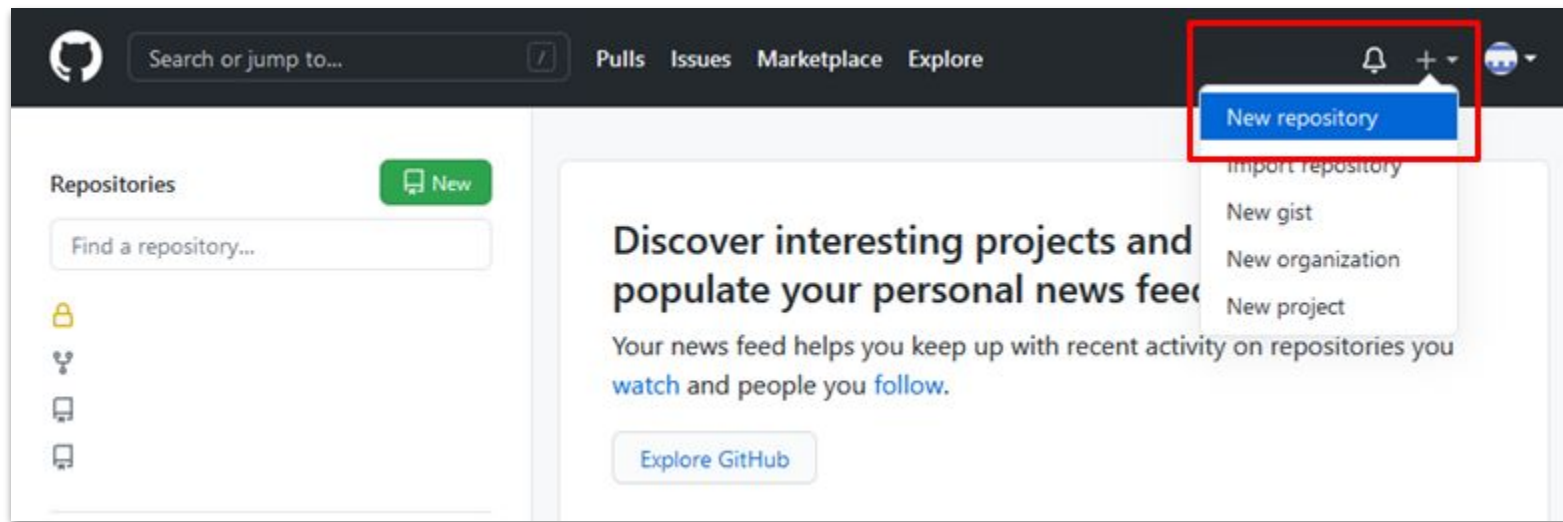
1. 直接使用命令操作
2. 使用圖形化介面 如:GitHub Desktop

方法一:使用命令操作:安裝 Git

- Windows >
 - 直接使用 command line 進行操作
 - 下載 [git](#)
- Mac >
 - 建議使用 Homebrew 安裝 Git
 - 首先, 先安裝 [Homebrew](#)
 - 安裝 Git (指令:brew install git)

在 GitHub 上建立遠端儲存庫

- 點選右上角 +
- 點選 New repository 新增儲存庫




建立遠端儲存庫

- 輸入 Repository name
(盡量與本地端專案一致)
ex: git_test
- 按下 Create repository


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *




Repository name *

git_test 


Great repository names are short and memorable. Need inspiration? How about [upgraded-doodle?](#)

Description (optional)

This is an example.

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

建立設定與第一次 push

- 在本機專案的資料夾根目錄下

- 輸入 `git init`

- 設定使用者資訊

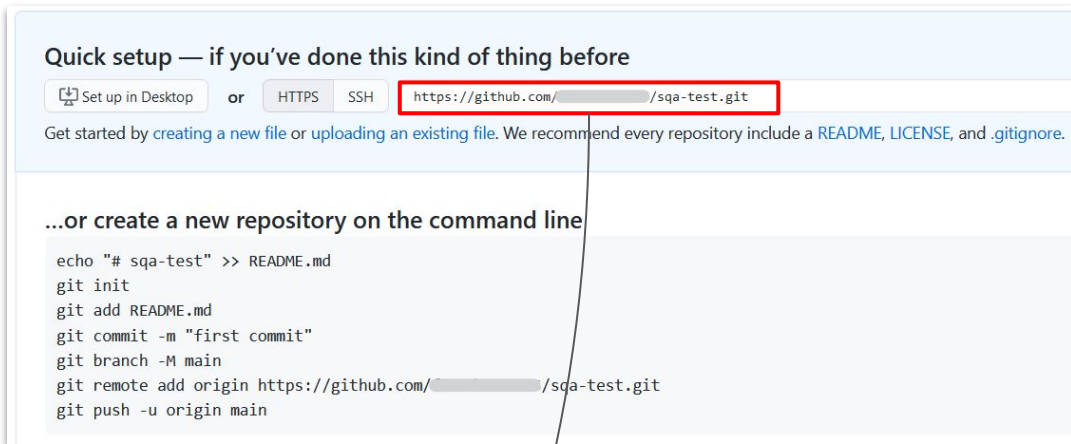
`git config --global user.name "<Your Name>"`

`git config --global user.email "<your@gmail.com>"`

- 輸入 `git commit -m "<提交資訊>"`

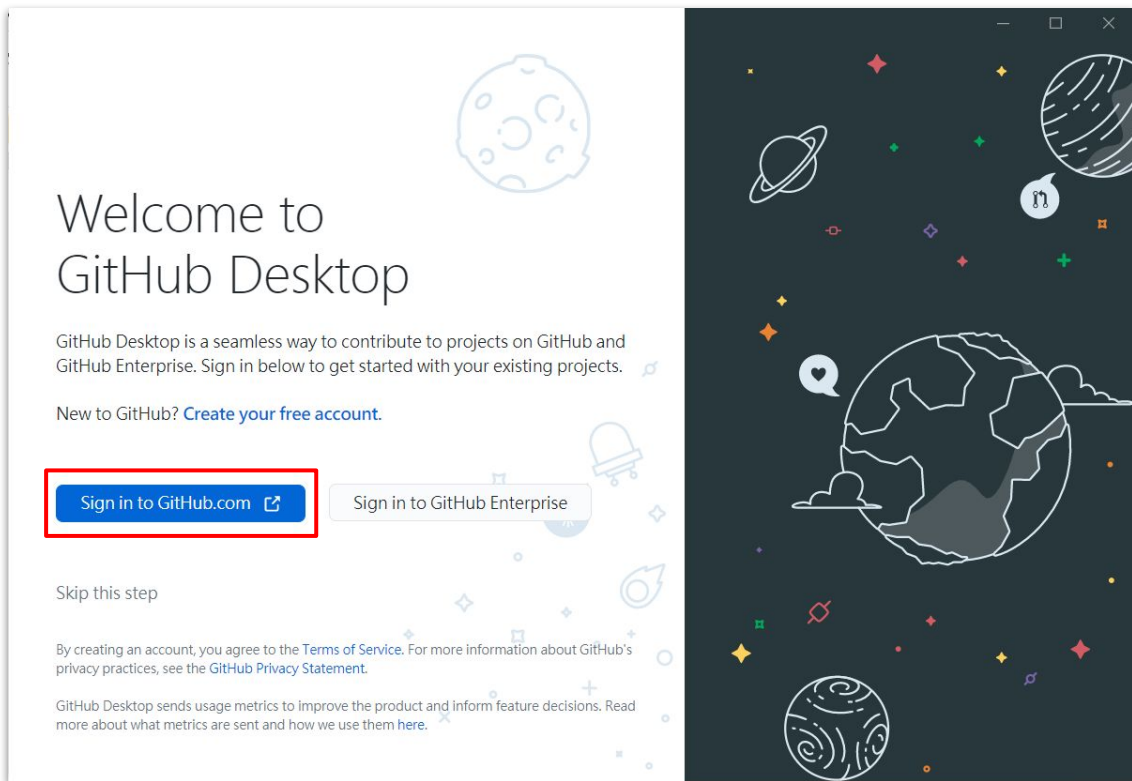
- 輸入 `git remote add origin <遠端網址>`

- 輸入 `git push -u origin <分支名稱>`



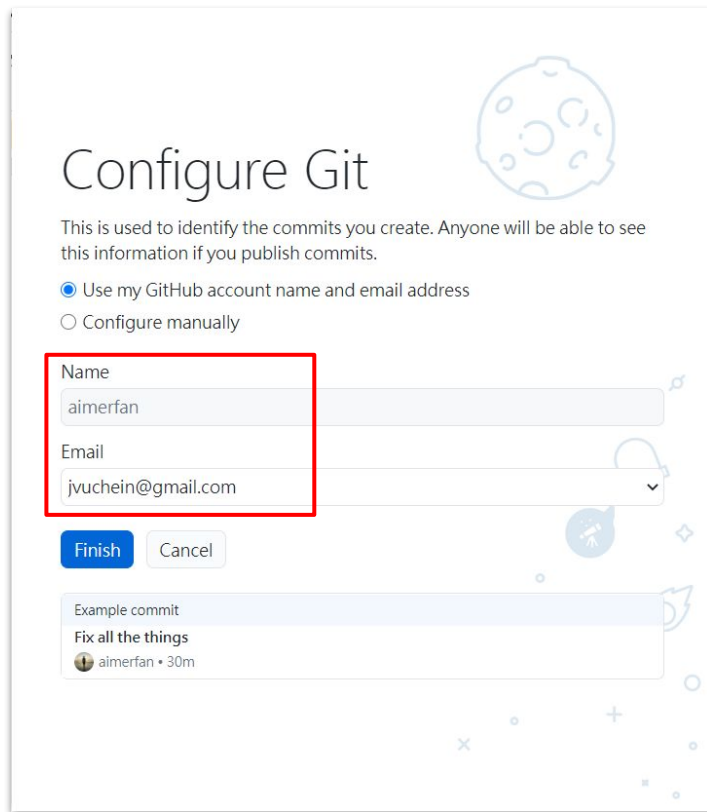
方法二:使用圖形化介面:安裝 GitHub Desktop

- Windows / Mac>
 - 安裝 [GitHub Desktop](#)
- 點選左邊藍色按鈕
登入 GitHub



Configure Git

- 輸入 GitHub 上的 username
- 與 GitHub 綁定的 email
- 按下 Finish



Configure Git

This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

☒ Use my GitHub account name and email address
☐ Configure manually

Name
aimerfan

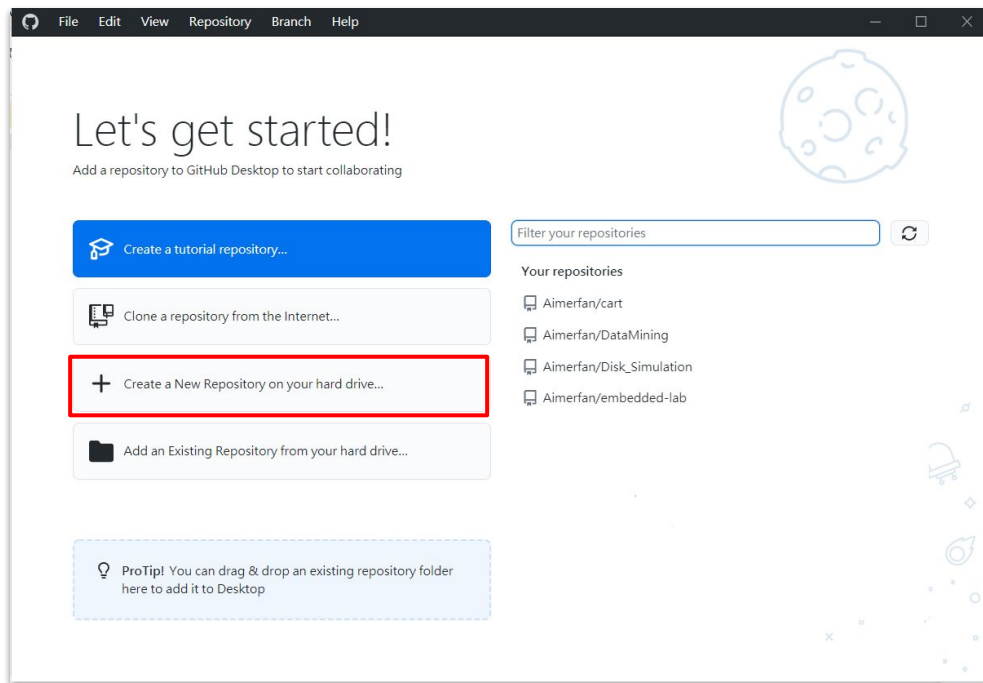
Email
jvuchein@gmail.com

Finish **Cancel**

Example commit
Fix all the things
aimerfan • 30m

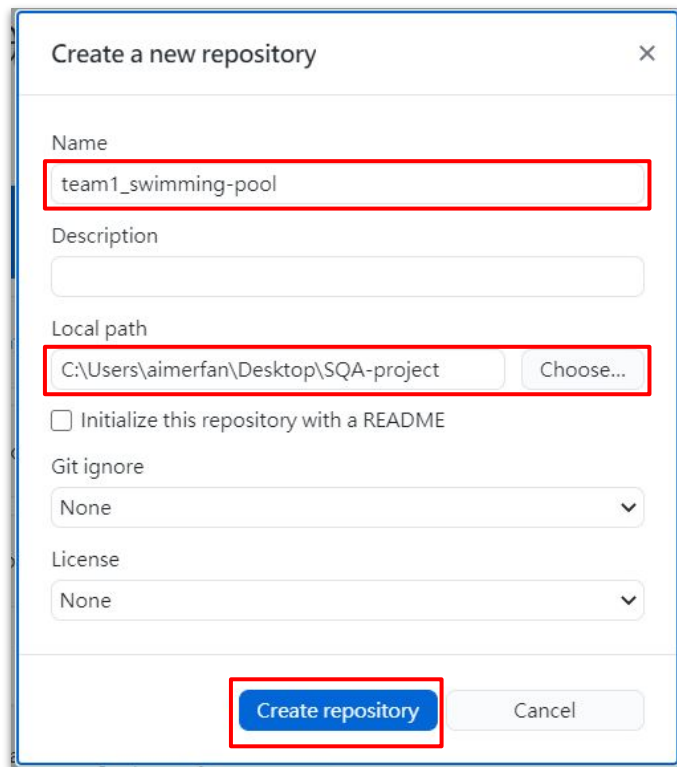
建立一個本地儲存庫 (repository)

- 點選 Create a New Repository on your hard drive...



設定儲存庫的資訊

- Name 輸入專案名稱
- Local path 選擇想儲存的本機位置
- 按下 Create repository

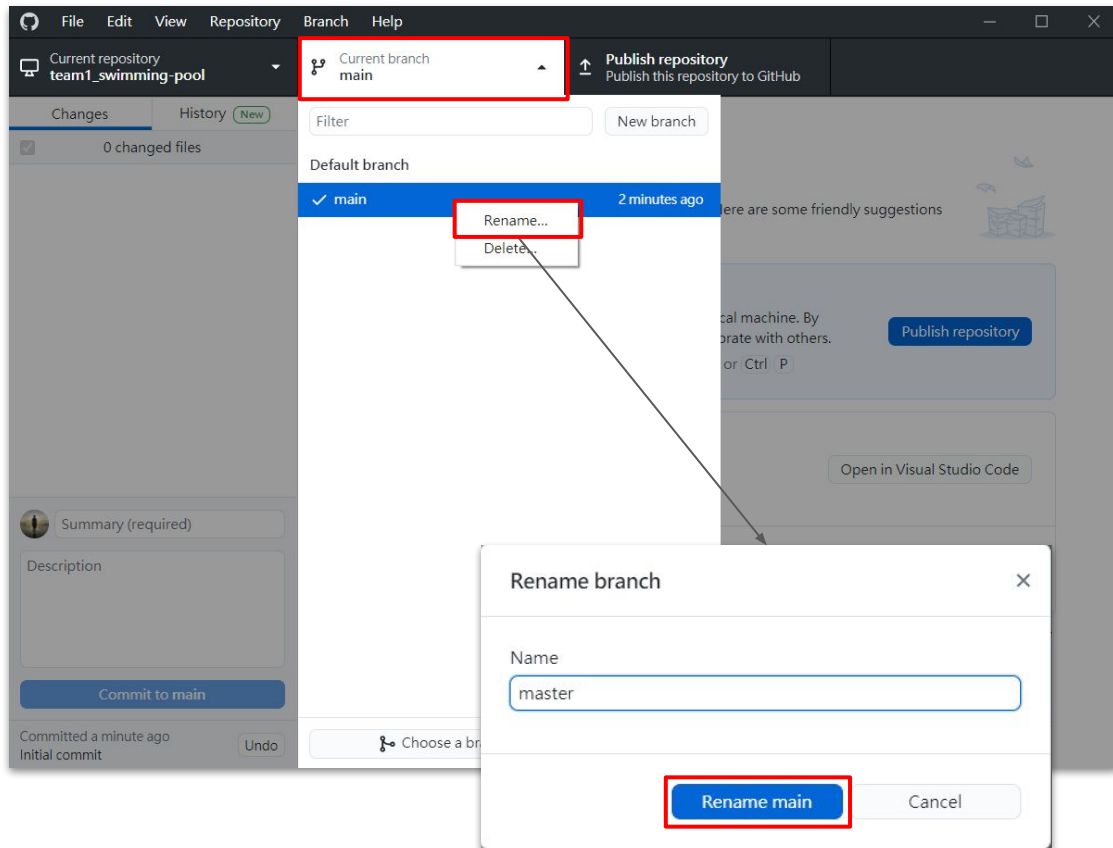


The screenshot shows a 'Create a new repository' dialog box. It has a title bar with a close button (X). The form contains the following fields and controls:

- Name:** A text input field containing 'team1_swimming-pool', highlighted with a red box.
- Description:** An empty text input field.
- Local path:** A text input field containing 'C:\Users\aimerfan\Desktop\SQA-project', highlighted with a red box. To its right is a 'Choose...' button, also highlighted with a red box.
- Initialize this repository with a README:** An unchecked checkbox.
- Git ignore:** A dropdown menu with 'None' selected.
- License:** A dropdown menu with 'None' selected.
- Buttons:** At the bottom right, there is a blue 'Create repository' button (highlighted with a red box) and a grey 'Cancel' button.

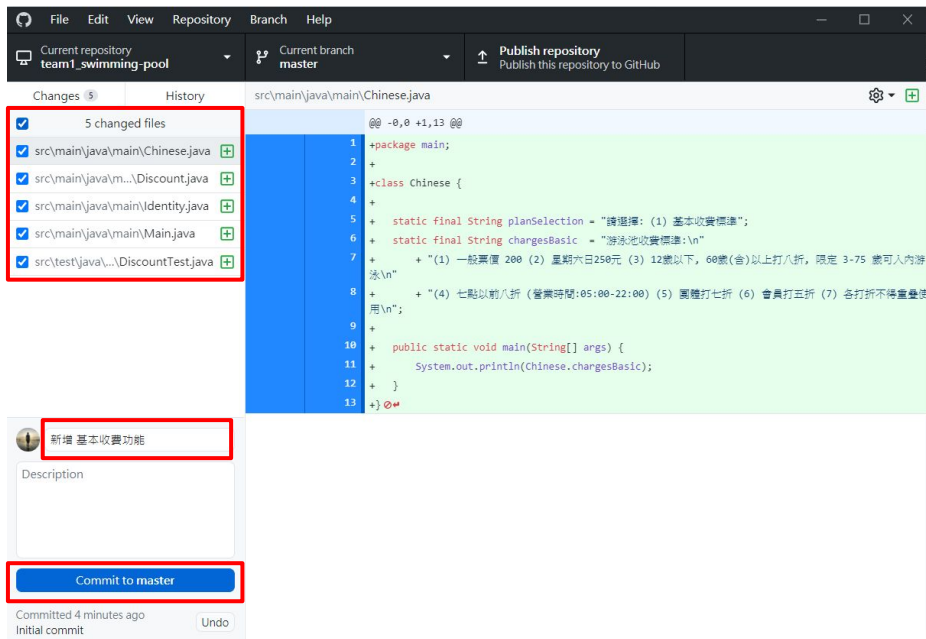
修改分支名稱

- 點選 Current branch
- 對 main 按右鍵
- 點選 Rename
- 輸入 master
- 點選 Rename main



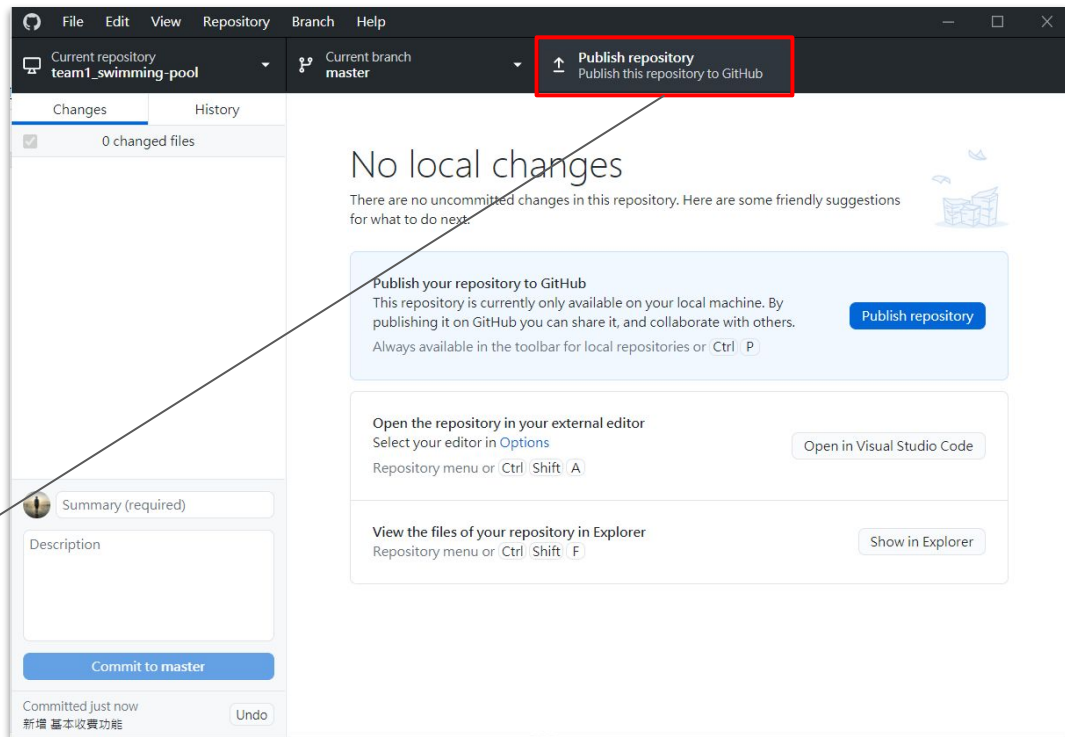
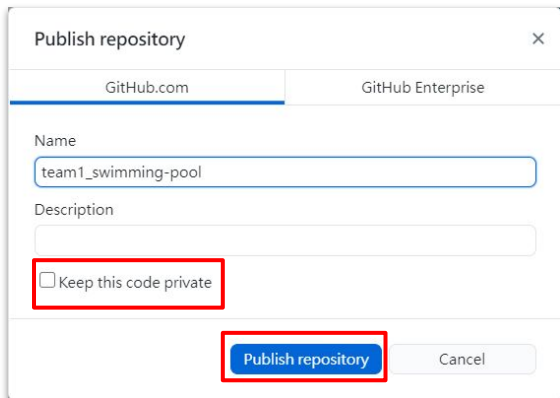
進行專案的 commit

- 將下載的專案放到本地儲存庫中
Ex: 放在 team1_swimming-pool 根目錄下
- 回到 GitHub Desktop
- 選擇想要 commit 的檔案
- 在 Summary 中
填寫此次修改資訊
- 按下 Commit to master



將專案 push 到遠端儲存庫 (GitHub)

- 點選 Publish repository
- 取消勾選
Keep this code private
- 按下 Publish repository



查看 GitHub

- 完成 push

The screenshot displays the GitHub web interface. At the top, there's a dark navigation bar with links for Issues, Marketplace, and Explore. Below this, a secondary navigation bar shows tabs for Overview, Repositories (selected), Projects, and Packages. A search bar labeled 'Find a repository...' is present, along with filters for Type, Language, and Sort, and a 'New' button. The main content area shows the repository 'team1_swimming-pool' by user 'Aimerfan', with a Java icon and 'Updated now' status. A 'Star' button is visible. Below the repository name, the navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Code' tab is active, showing a commit history table. The table has columns for the commit author and message, the commit hash and time, and the number of commits. The first commit is by 'Aimerfan' with the message '新增 基本收費功能', hash '46c692f', and '7 minutes ago', with '2 commits' indicated. Below this, a file tree shows 'src' (added 7 minutes ago) and '.gitattributes' (initial commit 11 minutes ago).

Issues Marketplace Explore

Overview Repositories 26 Projects Packages

Find a repository... Type Language Sort New

team1_swimming-pool Star

Java Updated now

Aimerfan / team1_swimming-pool

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

Aimerfan	新增 基本收費功能	46c692f 7 minutes ago	2 commits
src	新增 基本收費功能		7 minutes ago
.gitattributes	Initial commit		11 minutes ago

建立 自動化整合測試專案

Gradle 專案結構

調整專案結構如右圖

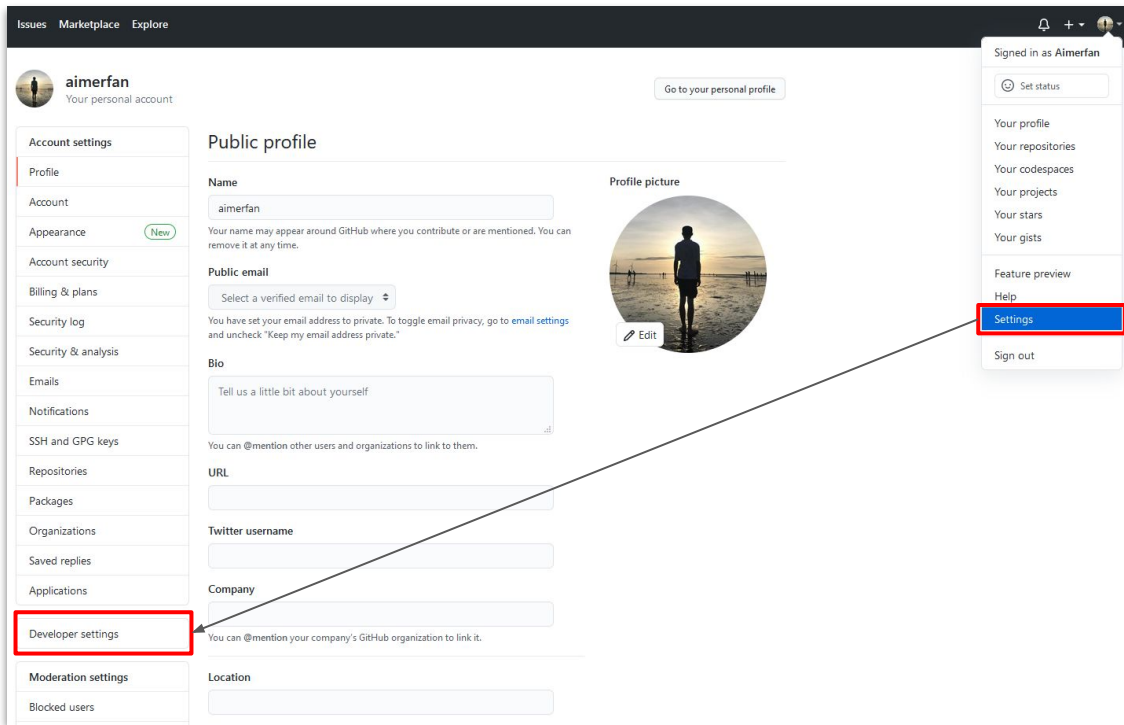
1. 原始碼路徑: `src/main/java/*`
2. 單元測試路徑: `src/test/java/*`
3. 其他 **gradle** 設定: 專案根目錄, 與 `src` 資料夾平行

這是 Gradle 預設的專案結構, 使用此結構後續就不用再指定路徑
可以省下很多時間



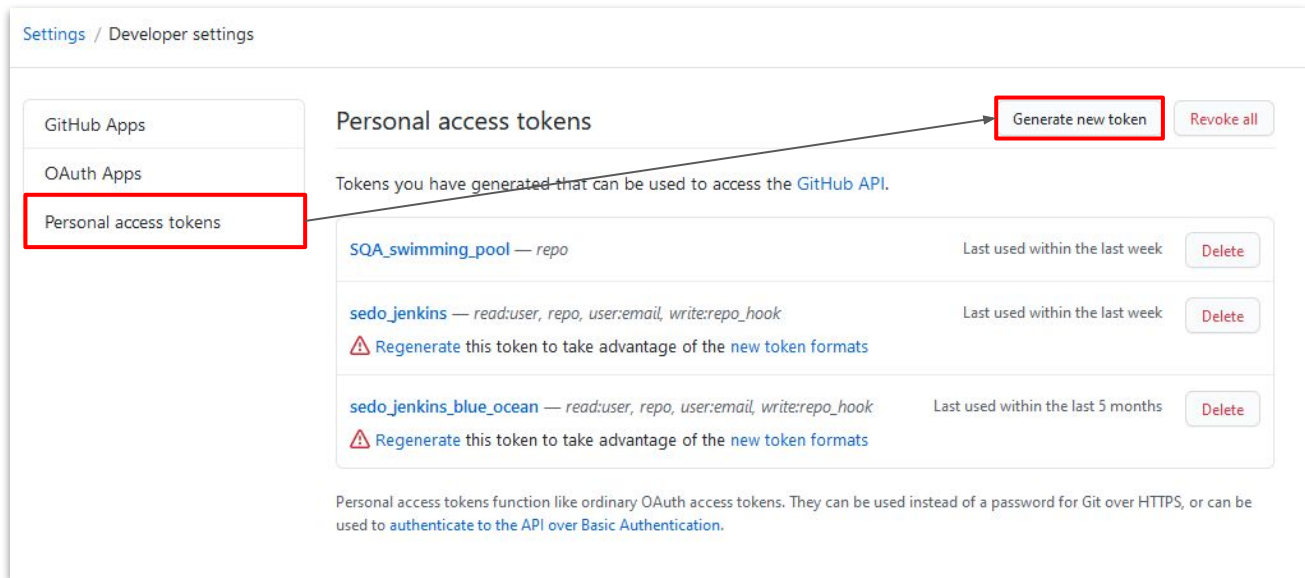
建立 GitHub 上的 Personal access token

- 點選頭貼下的 Settings
- 點選左側的 Developer settings



建立 GitHub 上的 Personal access token

- 點選左側的 Personal access tokens
- 建立一個 token (Generate new token)



設定 Personal access token

- 輸入 Note
方便辨識即可
- 將 repo 全選
- 點選下方的按鈕
Generate token

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

SQA_Jenkins

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

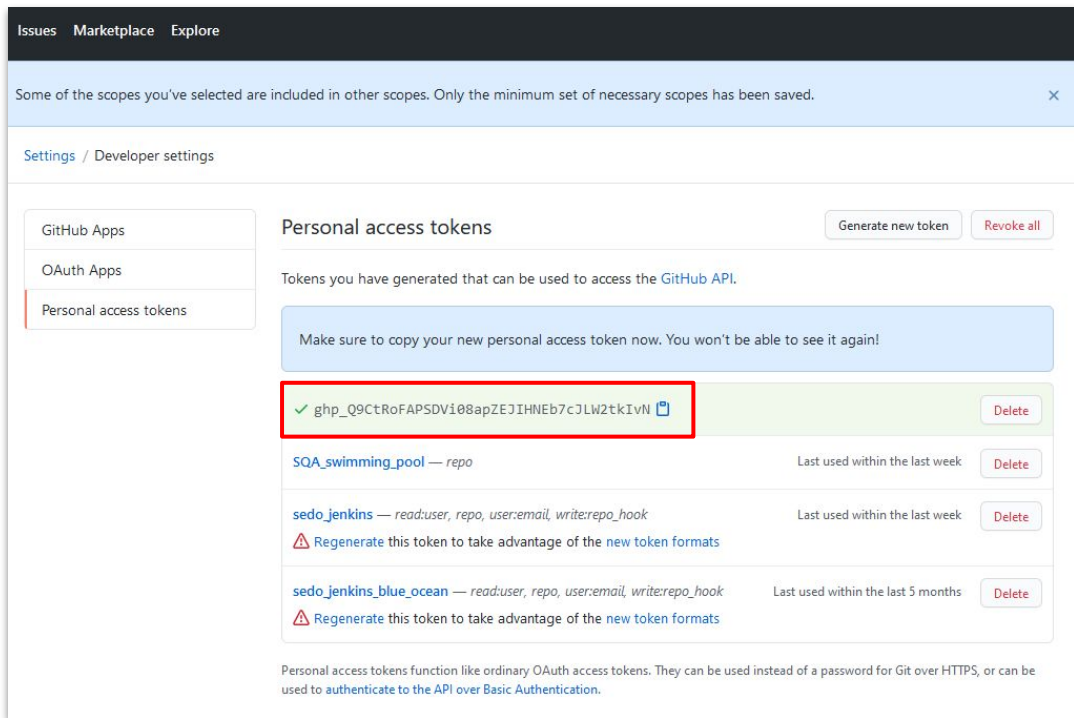
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry

Generate token

建立 Personal access token 成功

- 複製產生的 personal access token

(待會設定 Jenkins 時會用到)

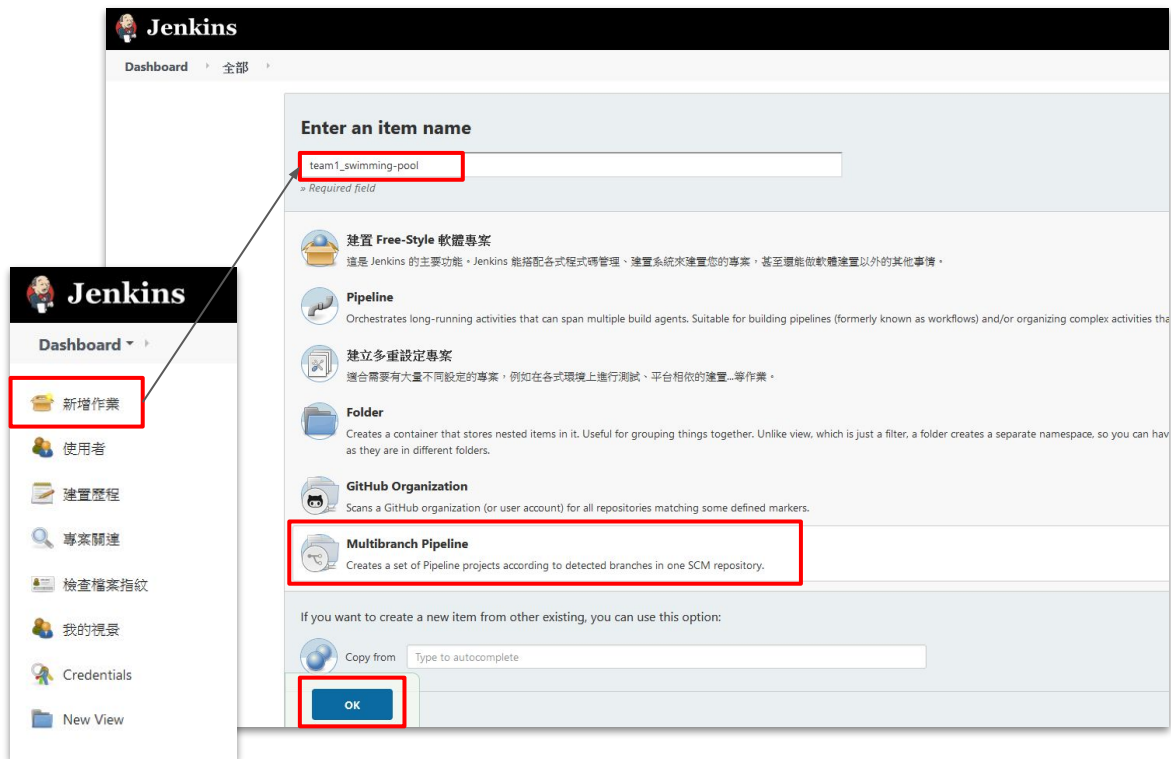


查看自己的 Jenkins 帳號

- Jenkins 帳號公布於 ilearn 上
- Jenkins 密碼為自己的學號 (開頭為小寫字母)

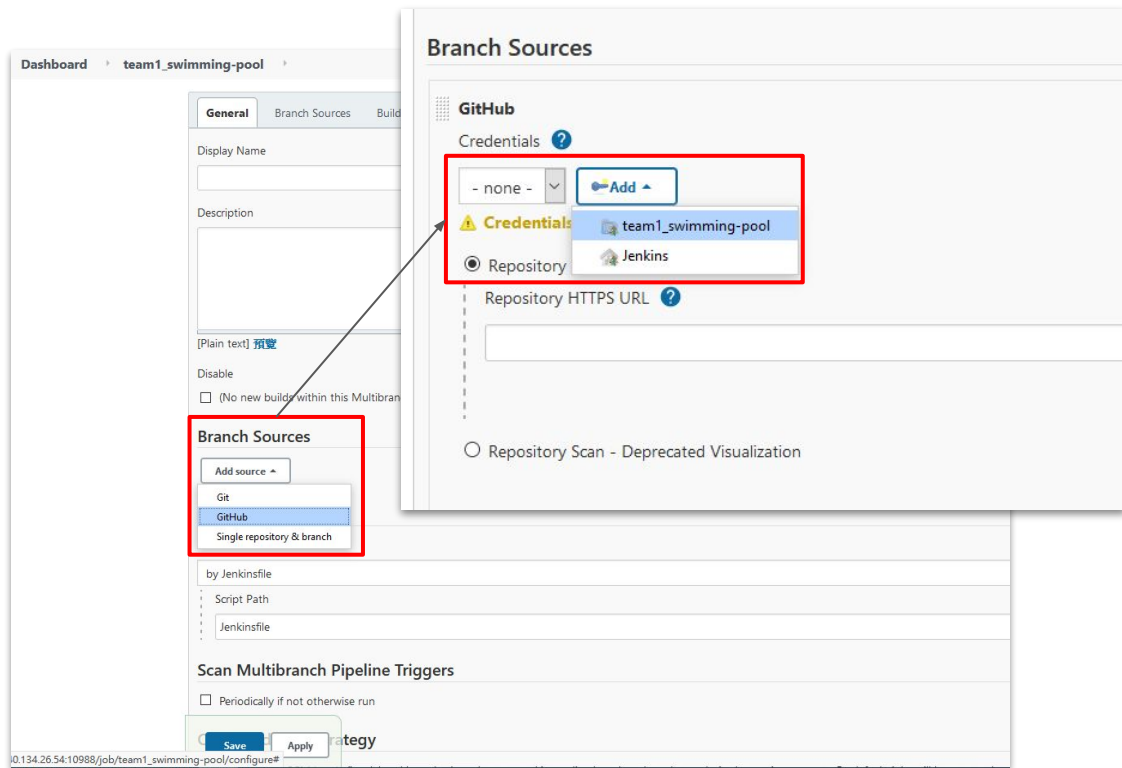
設定 Jenkins 自動化流程

- 進入 [Jenkins](#)
- 登入你的帳號
- 點選左側的新增作業
- 輸入 item name
格式為
team{數字}_{專案名稱}
- 選擇
Multibranch Pipeline
- 點選 OK



設定專案分支來源

- Branch Sources
選擇 GitHub
- 點選 Add →
team1_swimming-pool
(上一步建立的作業名稱)



設定 Credentials

- Kind 選擇 Username with password
- Username 輸入
自己在 GitHub 上的使用者名稱
- Password 輸入在 GitHub 上
建立的personal access token
- ID 自行輸入, 方便辨識即可
(全系統上 ID 不能重複)
- 按下 Add → 完成設定

General Branch Sources Build Configuration Scan Multibranch Pipeline Triggers Orphaned Item Strategy

Folder Credentials Provider: team1_swimming-pool

Add Credentials

Domain
Global credentials (unrestricted)

Kind
Username with password

Username
<your_github_username>

Password

ID


Description


Add Cancel


連接 GitHub Repository


- 在下拉式選單中選擇剛剛設定的 credentials
- Repository HTTPS URL 輸入 GitHub 上的專案連結

Branch Sources

 **GitHub**


Credentials 

aimerfan/***** 



User Aimerfan

☒ Repository HTTPS URL

Repository HTTPS URL 

https://github.com/Aimerfan/swimming-pool

☐ Repository Scan - Deprecated Visualization

設定自動化行為

- Behaviours 照右圖進行增減
- Filter by name (選填)
設定自動化測試時
排除或指定的分支

Behaviours

Within repository

Discover branches ?

Strategy ?

Exclude branches that are also filed as PRs

Discover pull requests from origin ?

Strategy ?

Merging the pull request with the current target branch revision

Filter by name (with wildcards)

Include ?

Exclude ?

General

Clean before checkout ?

☒ Delete untracked nested repositories ?

Add ▾

將組員加入群組

- 點選 Add user or group...
- 輸入組員的 Jenkins user name
- 勾選 作業>Read
- 按下 Save 存檔

Properties

☒ Enable project-based security

Inheritance Strategy

Inherit permissions from parent ACL

This item will inherit its parent items permissions (in addition to any permissions granted here). If this item is at the top level in Jenkins, it will inherit the [global security security settings](#).

使用者或群組	Credentials				作業							執行		視覺			SCM					
	Create	Delete	Manage Domains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag
Anonymous Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
aimerfan	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
shan	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

設定 GitHub 專案與 Jenkins 連接

- 進入 GitHub 專案下
- 點選 Settings
- 點選左側的 Webhooks
- 輸入 Payload URL
- Content type
選擇 application/json
- 按下 Update webhook

The screenshot shows the GitHub 'Webhooks / Manage webhook' settings page. The left sidebar contains a list of settings: Options, Manage access, Security & analysis, Branches, Webhooks (highlighted with an orange bar), Notifications, Integrations, Deploy keys, Autolink references, Actions, Environments, Secrets, Pages, and Moderation settings. The main content area is titled 'Webhooks / Manage webhook' and includes a description: 'We'll send a POST request to the URL below with details of any subscribed events. You can find more information in our developer documentation (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).' Below this, the 'Payload URL' field is set to 'http://140.134.26.54:10988/github-webhook/' and is highlighted with a red box. The 'Content type' dropdown is set to 'application/json' and is also highlighted with a red box. A 'Secret' field is present but empty. Under 'Which events would you like to trigger this webhook?', the 'Just the push event.' option is selected. The 'Active' checkbox is checked, with a note: 'We will deliver event details when this hook is triggered.' At the bottom, the 'Update webhook' button is highlighted with a red box, and the 'Delete webhook' button is visible next to it.

在專案下建立一個 Jenkinsfile

- 建立一個檔名為 Jenkinsfile
- 將專案 push 至 GitHub 上
 - git add .
 - git commit -m "<提交訊息>"
 - git push

```
pipeline {
  agent any
  /* insert Declarative Pipeline here */
  stages {
    stage('run-test') {
      when {
        anyOf {
          branch 'master'
          branch 'dev'
        }
      }
      steps {
        sh 'chmod +x ./gradlew'
        sh './gradlew test'
        jacoco(
          changeBuildStatus: true,
          classPattern: 'build/classes',
          exclusionPattern: '**/*Test*.class',
          execPattern: 'build/jacoco/**/*.exec',
          inclusionPattern: '**/*.class'
        )
      }
    }
  }
}
```

push 專案是否能觸發 Jenkins 自動化測試

- 檢查 GitHub 上的 Webhooks

The screenshot displays the GitHub repository settings for 'Aimerfan / swimming-pool'. The 'Webhooks' section is active, showing a single configured webhook with the URL 'http://140.134.26.54:10988/github...' (push). A red box highlights this URL, with an arrow pointing to the 'Recent Deliveries' section below. The 'Recent Deliveries' section shows two successful deliveries, with the top one highlighted by a red box. The first delivery has a SHA hash 'c10bd314-bbc0-11eb-8f2d-88163b15d4bd' and a timestamp of '2021-05-23 20:16:52'. The second delivery has a SHA hash 'b28c94b0-bbbe-11eb-9698-336b0f226b86' and a timestamp of '2021-05-23 20:02:08'.

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Autolink references

Actions

Environments

Secrets

Pages

Moderation settings



Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. [Learn more in our Webhooks Guide.](#)

✓ <http://140.134.26.54:10988/github...> (push) Edit Delete

Recent Deliveries

✓	 c10bd314-bbc0-11eb-8f2d-88163b15d4bd	2021-05-23 20:16:52	...
✓	 b28c94b0-bbbe-11eb-9698-336b0f226b86	2021-05-23 20:02:08	...

檢查 Jenkins 上的自動化測試

- 點選專案 > 分支名稱

The screenshot displays the Jenkins web interface. On the left is a sidebar with navigation links: 新增作業, 使用者, 建置歷程, 專案關連, 檢查檔案指紋, 我的視景, Credentials, and New View. The main area shows the 'Dashboard' for a project named 'swimming-pool'. A table lists the project's builds, with the first build highlighted. A red box highlights the 'swimming-pool' project name, and another red box highlights the 'master' branch name in the expanded view below. An arrow points from the project name to the expanded view.

S	W	名稱 ↓	上次成功時間
		swimming-pool	1 時 6 分 - log

圖示: S M L

Branches (1)		Pull Requests (0)			
S	W	Name ↓	上次成功時間	上次失敗時間	上次建置花費時間
		master	1 時 3 分 - #1	無	11 秒

圖示: S M L

回例 | Atom feed 摘要: 全部 | Atom feed 摘要: 失敗 | Atom feed 摘要: 最近幾次建置

檢查 Jenkins 上的自動化測試

- 通過測試
- 點選左側的建置歷程可以查看細節

Up

Status

Changes

馬上建置

檢視設定

Full Stage View

GitHub

Pipeline Syntax

建置歷程 趨勢 ^

find X

#1 2021/5/23 下午 8:17

Atom feed 摘要: 全部 Atom feed 摘要: 失敗

Branch master

Full project name: swimming_pool_shan_gradle/master

Recent Changes

Stage View

Declarative: Checkout SCM	run-test
1s	7s

Average stage times:
(Average full run time: ~11s)

#1 May 23 20:17 No Changes

1s 7s

永久連結

- 最新建置 (#1), 1 時 5 分 以前
- 最新穩定建置 (#1), 1 時 5 分 以前
- 最新成功建置 (#1), 1 時 5 分 以前
- Last completed build (#1), 1 時 5 分 以前

查看覆蓋率狀況

- 從 Status 可以看見整體 coverage 的情況

The screenshot shows the Jenkins Build #2 interface. On the left is a sidebar with navigation links: Back to Project, Status (selected), Changes, Console Output, View Build Information, Git Build Data, Coverage Report, Pipeline Steps, Workspaces, and Previous Build. The main content area displays the build details for Build #2 (2021/5/23 下午 10:34:30). It includes a 'Changes' section with a commit message '1. chore: 新增 Jacoco 報告 (details / githubweb)', a 'Push event to branch master at 10:34:14 PM on May 23, 2021', and a 'Revision' section showing the commit hash '86f40c4f7ad078f05232f9a81a242bdfdd453bac' and the branch 'master'. Below this is the 'Jacoco - Overall Coverage Summary' section, which contains a table with coverage metrics and corresponding horizontal bar charts.

Metric	Coverage	Visual
INSTRUCTION	21%	[Bar chart showing 21% coverage]
BRANCH	25%	[Bar chart showing 25% coverage]
COMPLEXITY	23%	[Bar chart showing 23% coverage]
LINE	18%	[Bar chart showing 18% coverage]
METHOD	27%	[Bar chart showing 27% coverage]
CLASS	33%	[Bar chart showing 33% coverage]

查看 Coverage Report

- 從 Coverage Report 可以看到更細節的 coverage 資訊

