國立臺北科技大學 National Taipei University of Technology

Department of Computer Science and Information Engineering

Object Oriented Programming

Environment Setup

Shuo-Han Chen (陳碩漢),
shchen@ntut.edu.tw

The Sixth Teaching Building 327
M 15:10 - 16:00 & F 10:10 - 12:00

Steps of Environment Setup

- Following steps are required to do your future homework
  - Part I -> Submit report of the success screenshot

1. Install Visual Studio Code
2. Setup ubuntu WSL & Google Test Library on Windows
3. Do the HelloWorld
4. Try using Google Test for your function

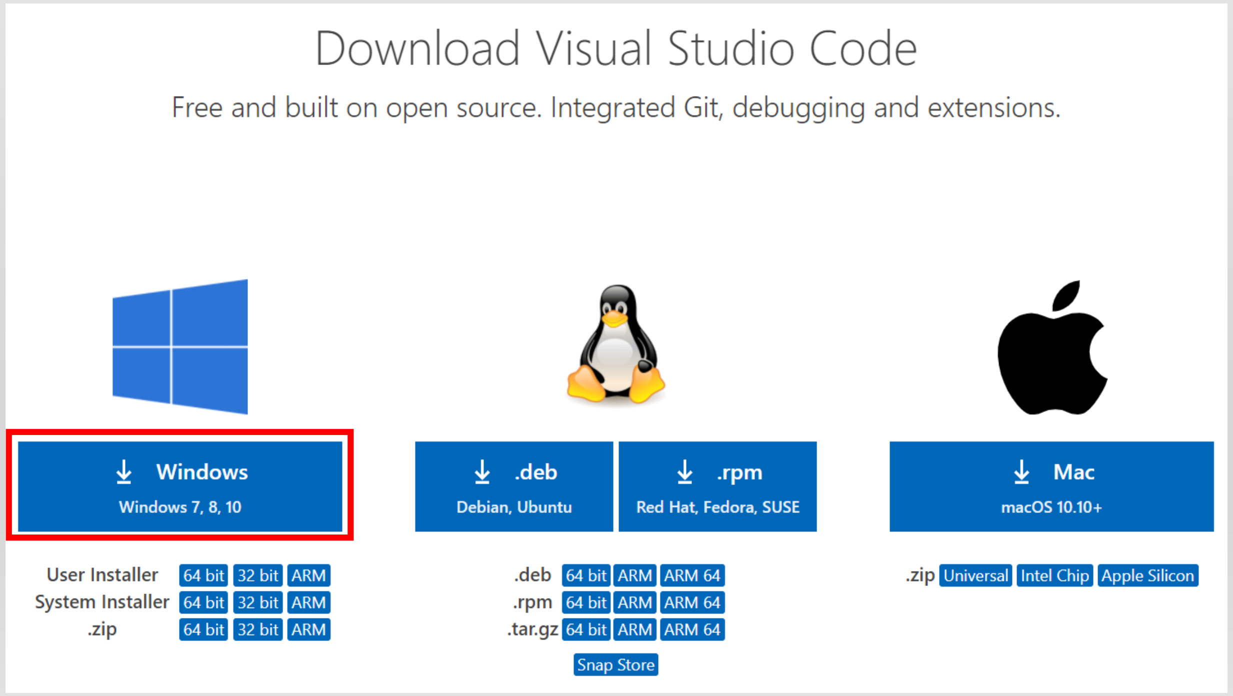
  - Part II -> Trigger Jenkins

1. Go check GitLab and Jenkins websites
2. Setup ssh key for git and Using git cmd
3. Git push the HelloWorld

Part I – Before 09/29 24:00

Install Visual Studio Code

- You will find that every teacher ask you to use different Integrated Development Environment (IDE)
  - It’s quire normal since every company use different ways for writing code
  - And, setup the environment is always the first thing for programmer
  - Go here and download https://code.visualstudio.com/download

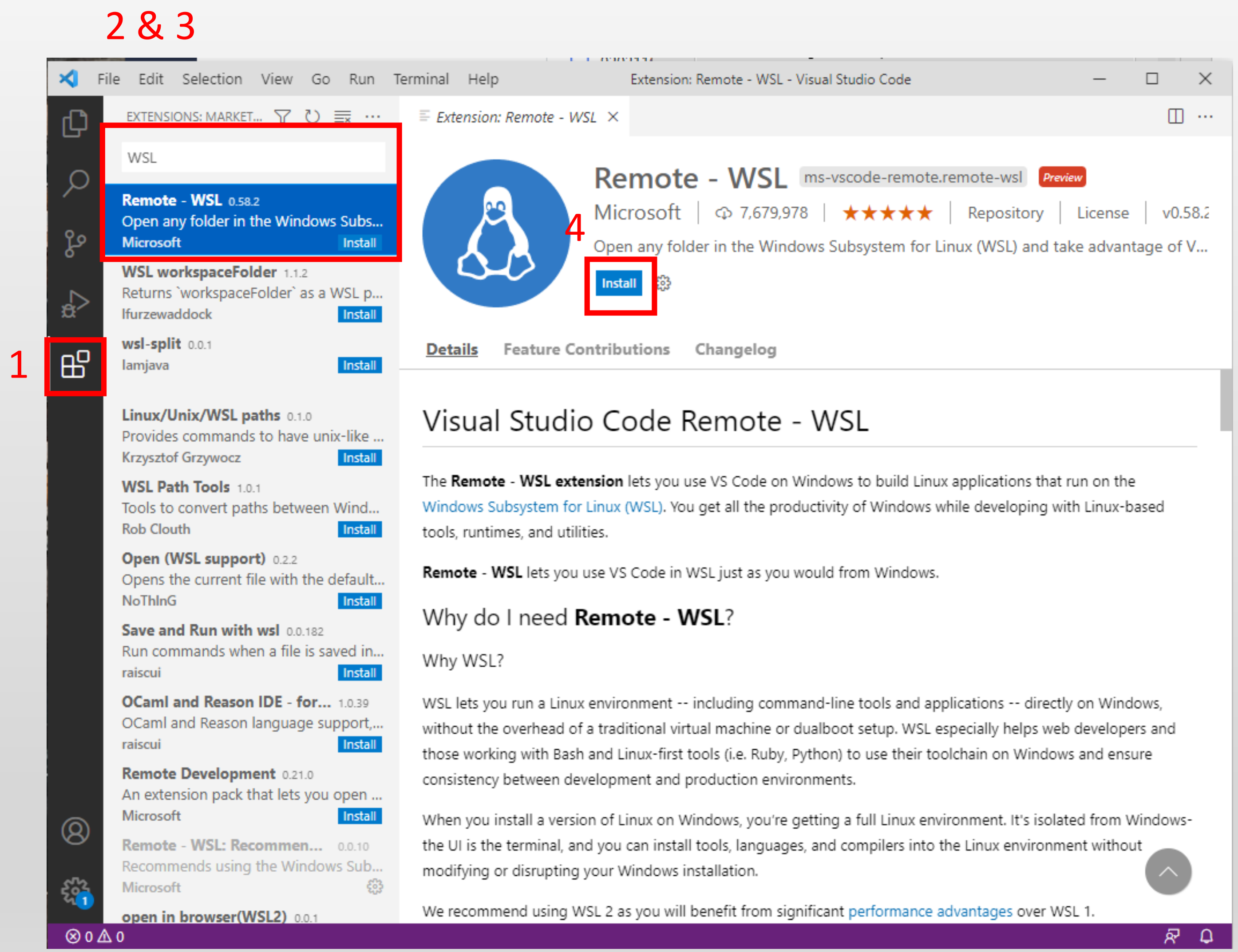


Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

Install Visual Studio Code (Cont'd)

- Install an extension in VS Code
  1. Select Extension
  2. Search “WSL“
  3. Find “Remote - WSL “
  4. Click Install
- What is WSL ?
  - WSL = Windows Subsystem for Linux
  - A virtual machine in Windows that have the functionality of Linux
- Why do we need this ?
  - We are going to compile your program with Linux commands
  - Key terms you can look into :
    - gcc, g++, make, makefile


2 & 3
1
4
File Edit Selection View Go Run Terminal Help
EXTENSIONS: MARKET...
Extension: Remote - WSL
Details Feature Contributions Changelog

Extension: Remote - WSL - Visual Studio Code

WSL

Remote - WSL 0.58.2
Open any folder in the Windows Subs...
Microsoft Install

WSL workspaceFolder 1.1.2
Returns `workspaceFolder` as a WSL p...
lfurzewaddock Install

wsl-split 0.0.1
lamjava Install

Linux/Unix/WSL paths 0.1.0
Provides commands to have unix-like ...
Krzysztof Grzywocz Install

WSL Path Tools 1.0.1
Tools to convert paths between Wind...
Rob Clouth Install

Open (WSL support) 0.2.2
Opens the current file with the default...
NoThInG Install

Save and Run with wsl 0.0.182
Run commands when a file is saved in...
raiscui Install

OCaml and Reason IDE - for... 1.0.39
OCaml and Reason language support,...
raiscui Install

Remote Development 0.21.0
An extension pack that lets you open ...
Microsoft Install

Remote - WSL: Recommen... 0.0.10
Recommends using the Windows Sub...
Microsoft

open in browser(WSL2) 0.0.1

Remote - WSL ms-vscode-remote.remote-wsl Preview
Microsoft | 7,679,978 | ★★★★★ | Repository | License | v0.58.2
Open any folder in the Windows Subsystem for Linux (WSL) and take advantage of V...
Install

Visual Studio Code Remote - WSL

The **Remote - WSL extension** lets you use VS Code on Windows to build Linux applications that run on the Windows Subsystem for Linux (WSL). You get all the productivity of Windows while developing with Linux-based tools, runtimes, and utilities.

Remote - WSL lets you use VS Code in WSL just as you would from Windows.

Why do I need **Remote - WSL**?

Why WSL?

WSL lets you run a Linux environment -- including command-line tools and applications -- directly on Windows, without the overhead of a traditional virtual machine or dualboot setup. WSL especially helps web developers and those working with Bash and Linux-first tools (i.e. Ruby, Python) to use their toolchain on Windows and ensure consistency between development and production environments.

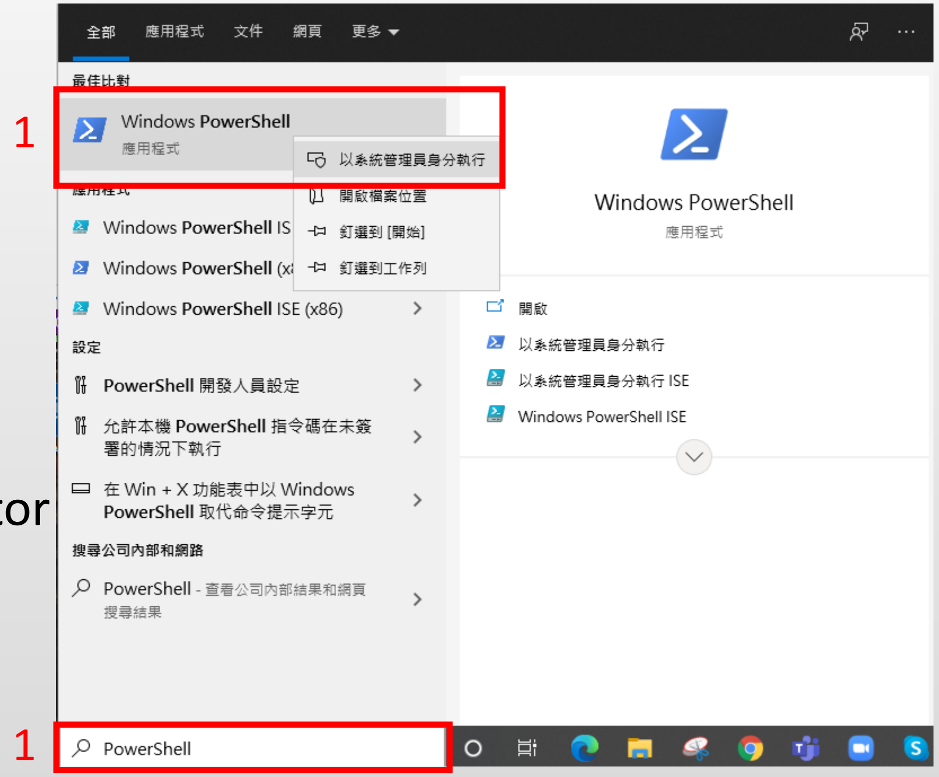
When you install a version of Linux on Windows, you're getting a full Linux environment. It's isolated from Windows- the UI is the terminal, and you can install tools, languages, and compilers into the Linux environment without modifying or disrupting your Windows installation.

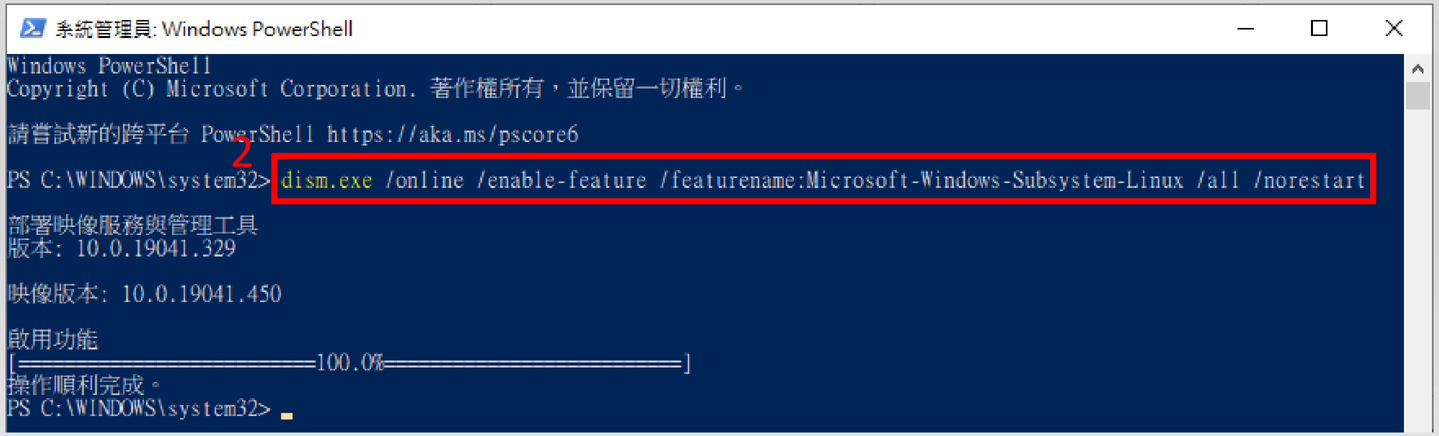
We recommend using WSL 2 as you will benefit from significant performance advantages over WSL 1.

⊗ 0 ⚠ 0

Setup Ubuntu Bash Shell on Windows

- We need ubuntu for
  - Compiler: g++
  - Builder: make and makefile
- We need to enable the “Windows Subsystem for Linux 1” on Windows 10
  1. Search & Right Click on PowerShell -> Run as Administrator
  2. Entry command: dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
     - Blank space could be missing
     - Compare the your copied command with the one in the figure

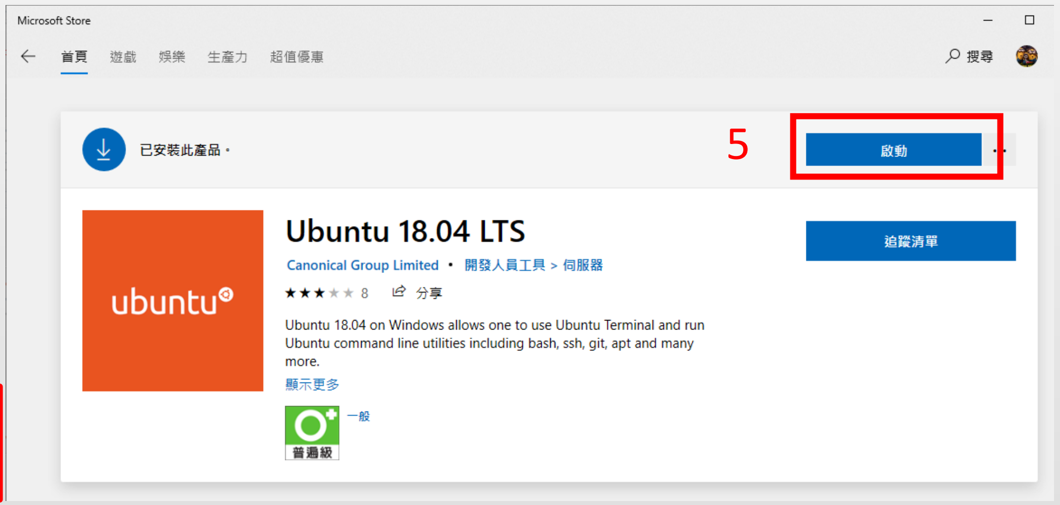




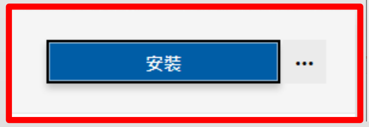
```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

部署映像服務與管理工具
版本: 10.0.19041.329

映像版本: 10.0.19041.450

啟用功能
[==========================100.0%==========================]
操作順利完成。
PS C:\WINDOWS\system32> _
```

Setup Ubuntu Bash Shell on Windows (Cont'd)

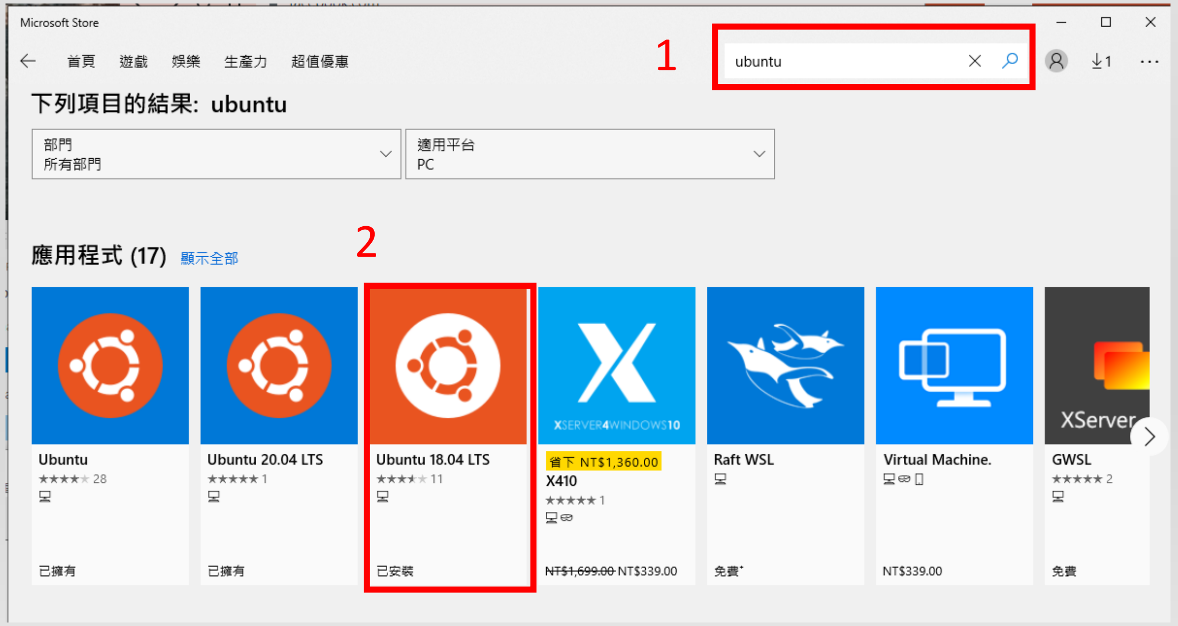
- Go to Microsoft Store and Install Ubuntu 18.04 LTS
  1. Search “ubuntu”
  2. Select Ubuntu 18.04 LTS
  3. Click “Install”
  4. Wait until it’s installed
  5. Restart your computer
  6. Run ubuntu & Complete setup

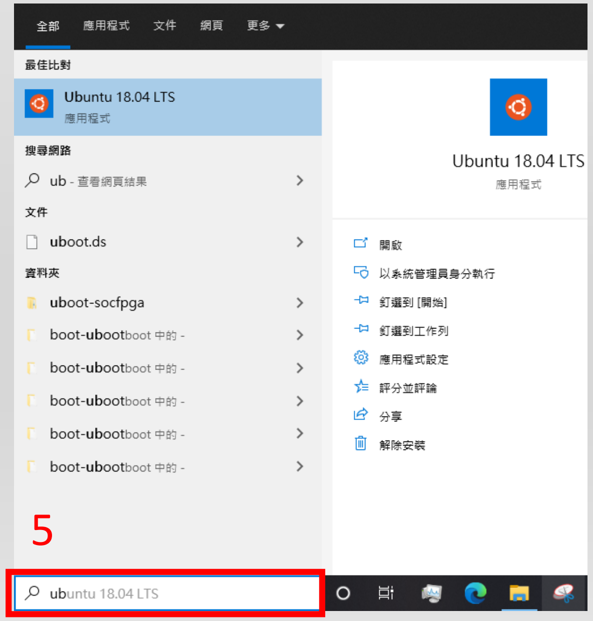


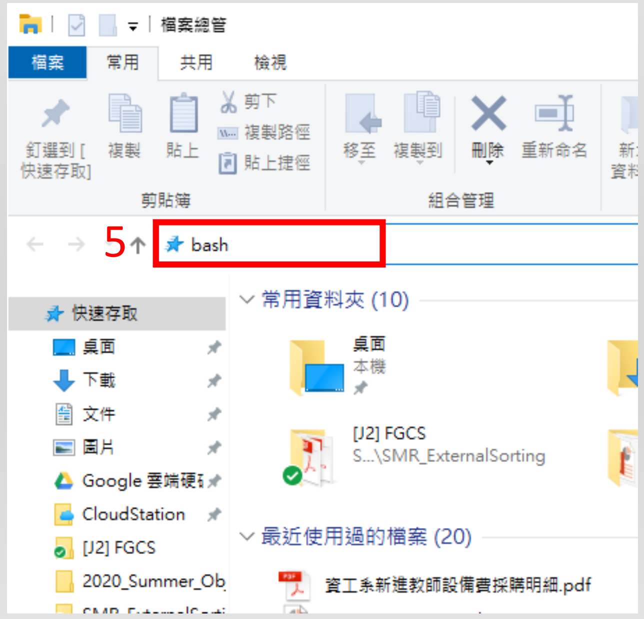
Ubuntu 18.04 on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.



3

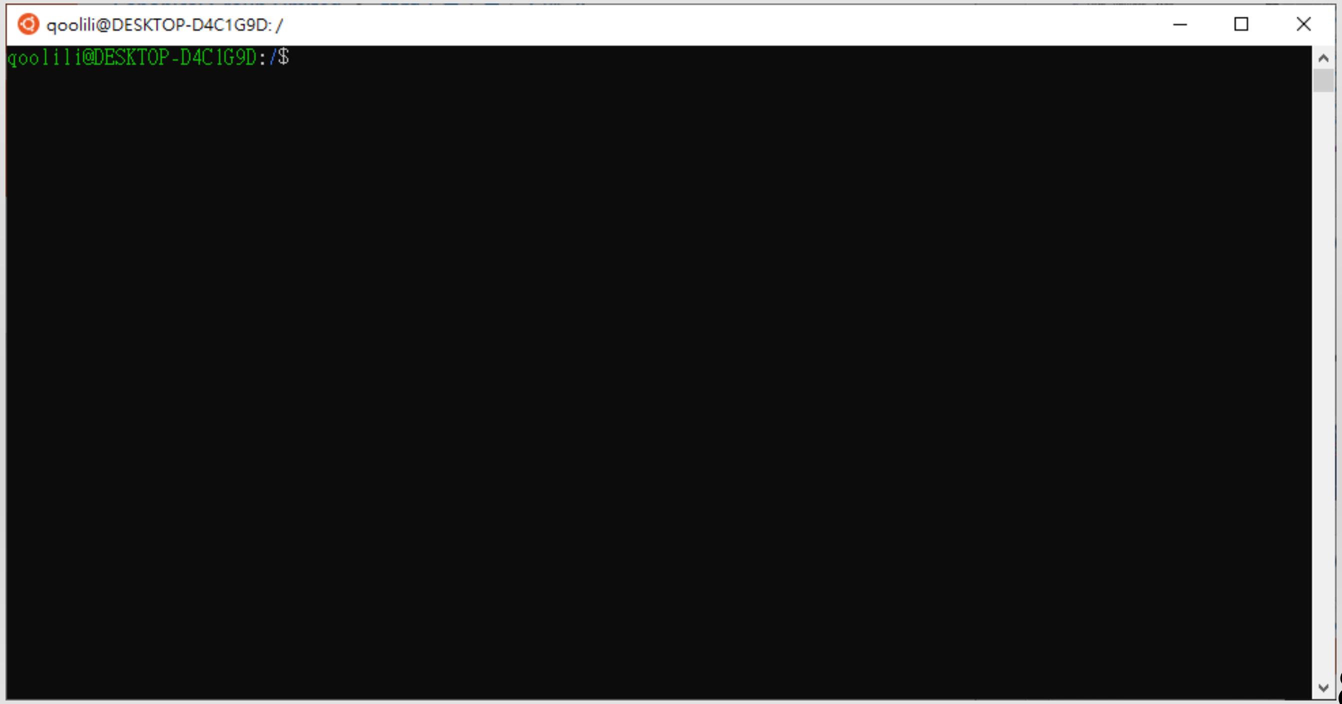






Setup Ubuntu Bash Shell on Windows (Cont'd)

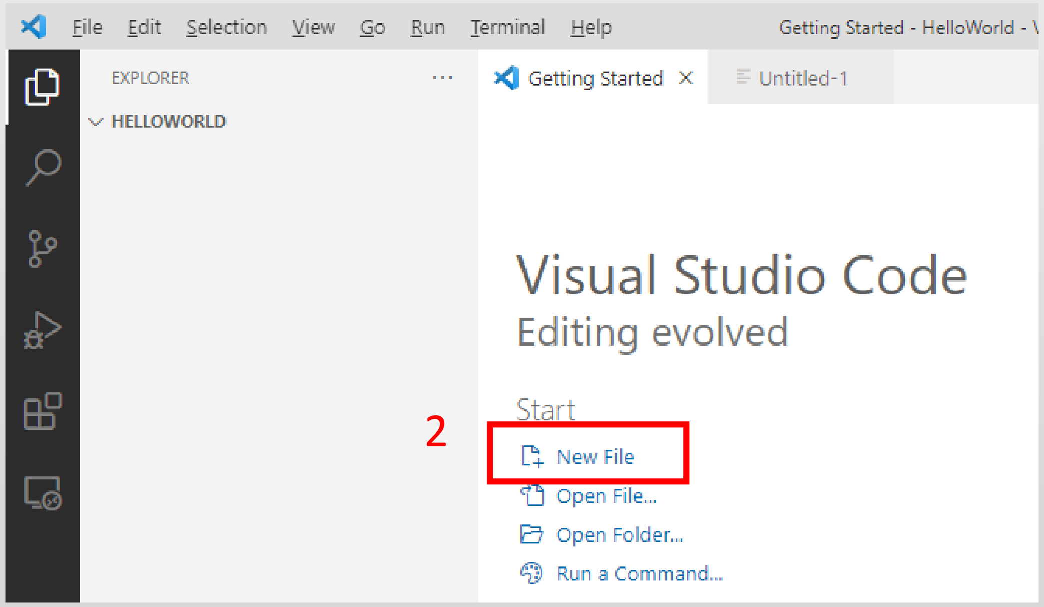
- Setting up Ubuntu 18.04 LTS may require to set username & password
- After Ubuntu 18.04 LTS is setup, you should see a black terminal
- Let's install the tools we need by entering following command
  1. sudo apt-get update
  2. sudo apt-get install g++ make libgtest-dev cmake
  3. cd /usr/src/gtest
  4. sudo cmake CMakeLists.txt
  5. sudo make
  6. sudo cp *.a /usr/lib
- libgtest-dev is the google test library

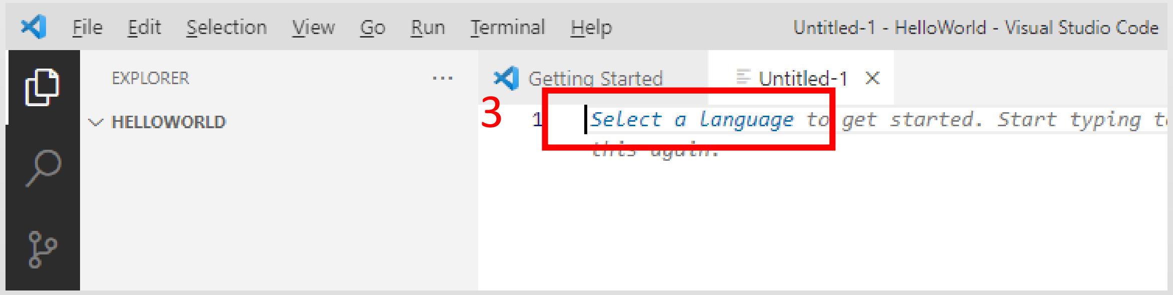


```
qoolili@DESKTOP-D4C1G9D:/$
```

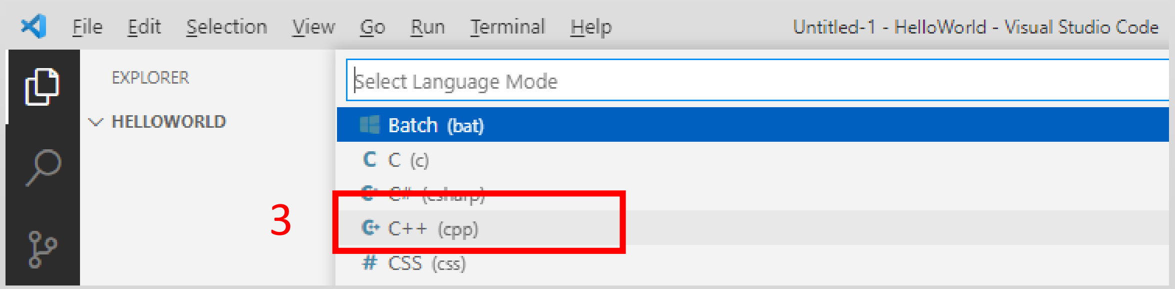
Do the HelloWorld

1. Create a folder with the name of HelloWorld
2. Go back to VS code -> Open a folder -> Find the HelloWorld folder
3. Select New File -> Select Language "C++"



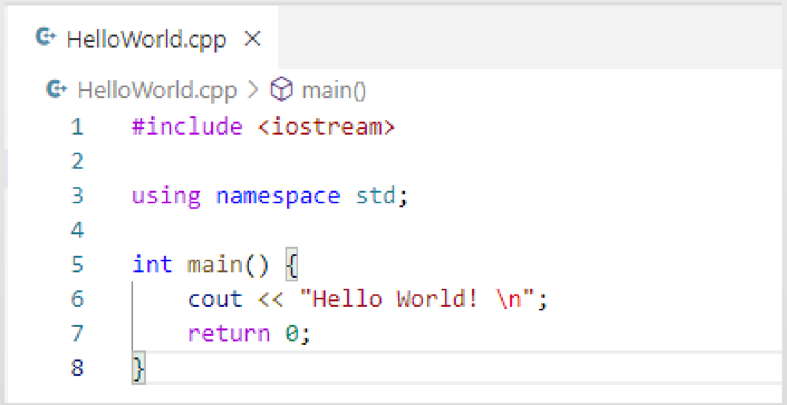

File Edit Selection View Go Run Terminal Help
Untitled-1 - HelloWorld - Visual Studio Code
EXPLORER
HELLOWORLD
Getting Started
Untitled-1
3

Select a language to get started. Start typing t
this again.


File Edit Selection View Go Run Terminal Help
Untitled-1 - HelloWorld - Visual Studio Code
EXPLORER
HELLOWORLD
Select Language Mode
Batch (bat)
C (c)
C# (csharp)
C++ (cpp)
CSS (css)
3

Do the HelloWorld (Cont'd)

4. Start your HelloWorld coding
5. Save file by pressing Ctrl and s on your keyboard
6. Enter the file name as “HelloWorld.cpp”



```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello World! \n";
    return 0;
}
```

- How do we run the program ? Where is the compile and run button ?
  - There is no such thing in large scale programming
  - Image you’re now working at Google, there will be no compile and run button at all
  - Your code are integrated to the beta/release code through continuous integration

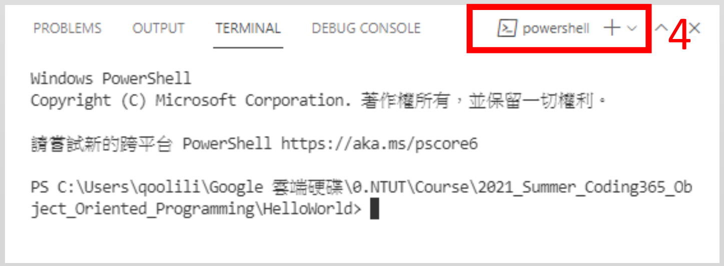
Do the HelloWorld (Cont'd)

- Before we jump into continuous integration, let's try running the code locally

1. Create another new file -> Don't Select language
2. Copy and Paste the following
   - Add a TAB before g++ and rm. Three lines that need tabs.
3. Save the file with filename "makefile" without extension
4. Click Terminal -> New Terminal
   - You should see the terminal
   - If you didn't see this, click "+" & "power shell"

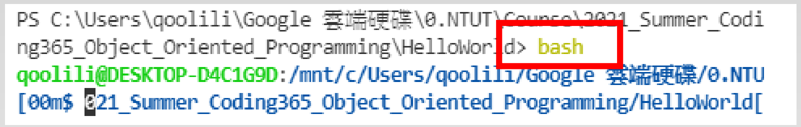
2

```
# This is the default target, which will be built when you invoke make
.PHONY: all

# This rule tells make how to build HelloWorld from HelloWorld.cpp
all: HelloWorld.cpp
	g++ HelloWorld.cpp -o HelloWorld

# This rule tells make to delete hello and hello.o
.PHONY: clean
clean:
	rm -f HelloWorld
	rm -f ut_all
```

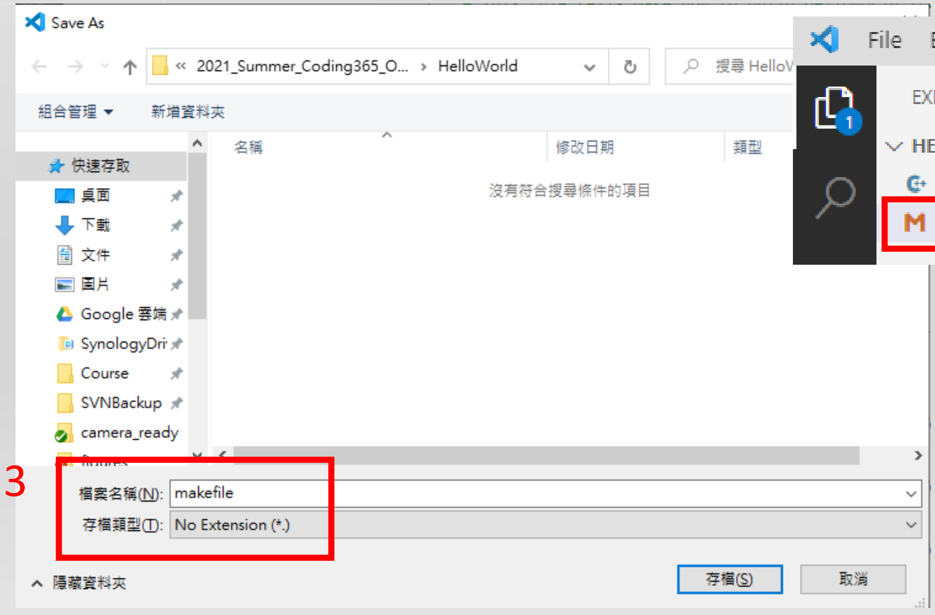


```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\qoolili\Google 雲端硬碟\0.NTUT\Course\2021_Summer_Coding365_Ob
ject_Oriented_Programming\HelloWorld>
```

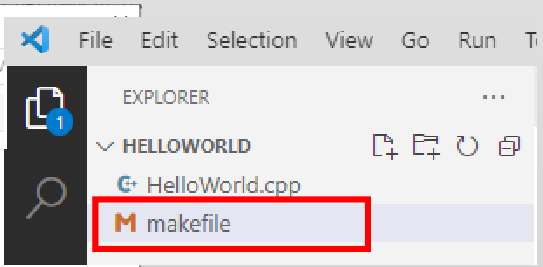
5. Enter "bash" to enter WSL



```
PS C:\Users\qoolili\Google 雲端硬碟\0.NTUT\Course\2021_Summer_Codi
ng365_Object_Oriented_Programming\HelloWorld> bash
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTU
[00m$ 021_Summer_Coding365_Object_Oriented_Programming/HelloWorld[
```



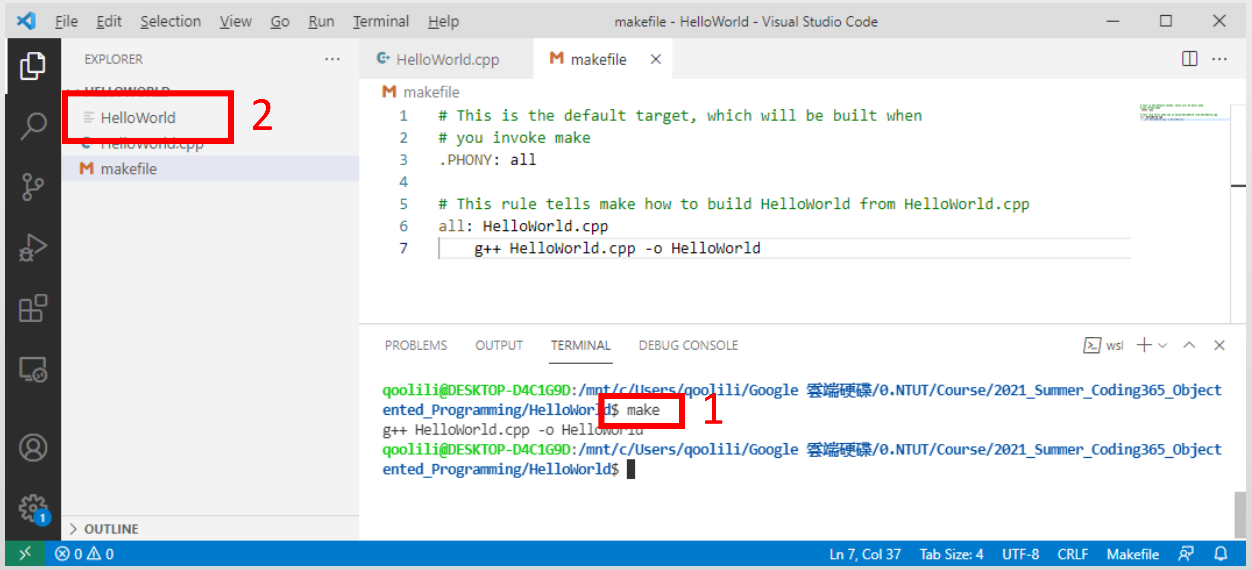
檔案名稱(N): makefile
存檔類型(T): No Extension (*.)



No Extension !

Do the HelloWorld (Cont'd)

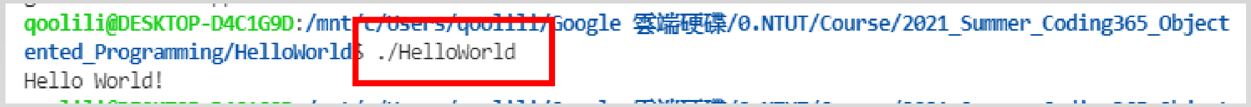
- Let’s compile and run your code locally -> At least you know your code can compile ☺
  1. Type in cmd “make”
  2. A runnable file will be generated


File Edit Selection View Go Run Terminal Help
makefile - HelloWorld - Visual Studio Code
EXPLORER
HELLOWORLD
HelloWorld
HelloWorld.cpp
makefile
2
HelloWorld.cpp
makefile
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
wsl
1
OUTLINE
Ln 7, Col 37 Tab Size: 4 UTF-8 CRLF Makefile

```
# This is the default target, which will be built when
# you invoke make
.PHONY: all

# This rule tells make how to build HelloWorld from HelloWorld.cpp
all: HelloWorld.cpp
	g++ HelloWorld.cpp -o HelloWorld
```

```
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object
ented_Programming/HelloWorld$ make
g++ HelloWorld.cpp -o HelloWorld
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object
ented_Programming/HelloWorld$
```

  3. Run the compiled file by typing in “./HelloWorld”

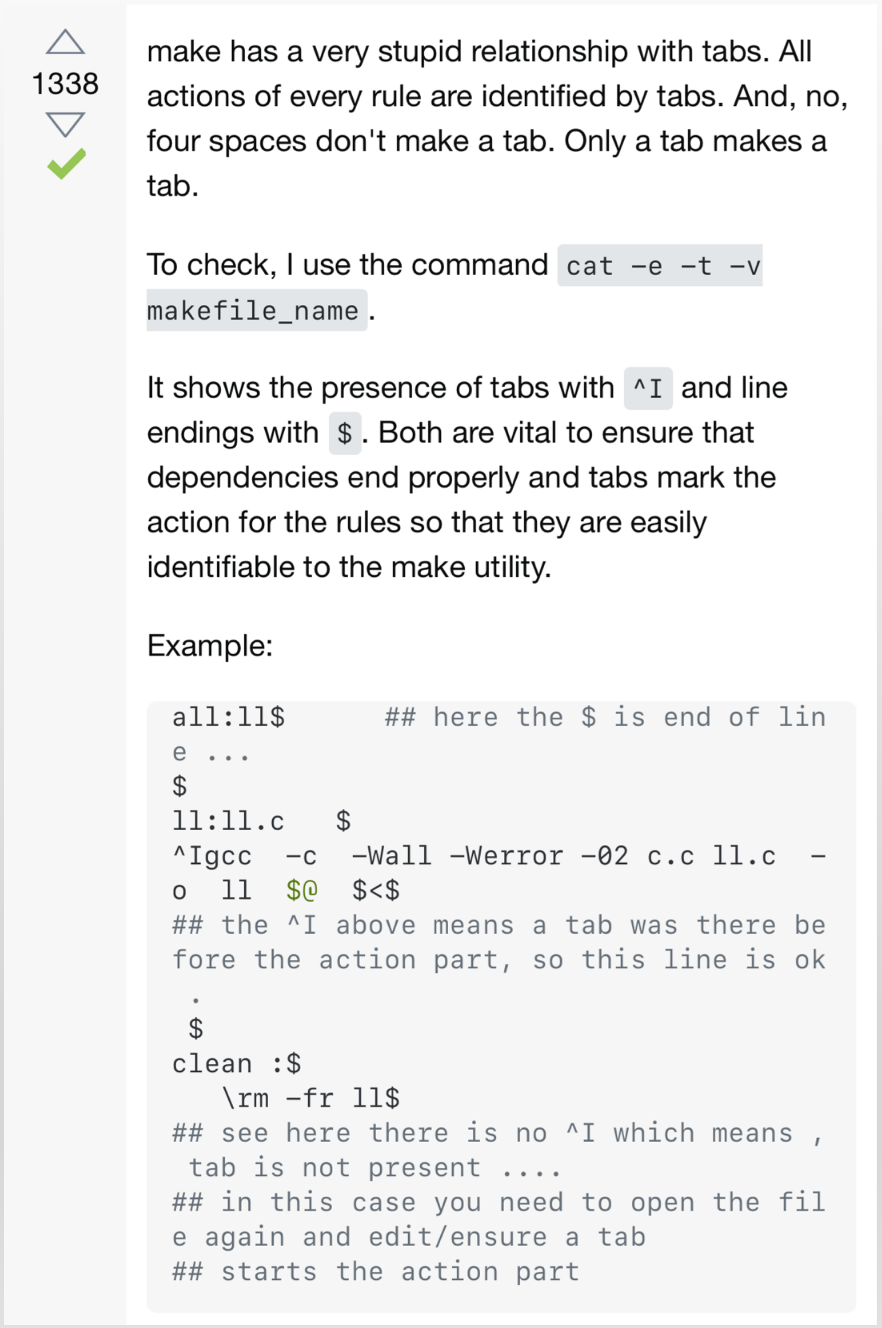


```
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object
ented_Programming/HelloWorld$ ./HelloWorld
Hello World!
```

  4. If you see the output, then Congrats ! We’re half way there.

Do the HelloWorld (Cont'd)

- Have problem with makefile?
  - It's usually the issue of missing a "tab" before gcc & rm
  - Do the following to double check
  - In cmd
    - cat -e -t -v makefile
  - Make sure there is a "^I" before gcc & rm
  - Try entering "tab" a few time, it might not show up at first time
  - Ref: https://stackoverflow.com/questions/16931770/makefile4-missing-separator-stop



make has a very stupid relationship with tabs. All actions of every rule are identified by tabs. And, no, four spaces don't make a tab. Only a tab makes a tab.

To check, I use the command `cat -e -t -v makefile_name`.

It shows the presence of tabs with `^I` and line endings with `$`. Both are vital to ensure that dependencies end properly and tabs mark the action for the rules so that they are easily identifiable to the make utility.

Example:

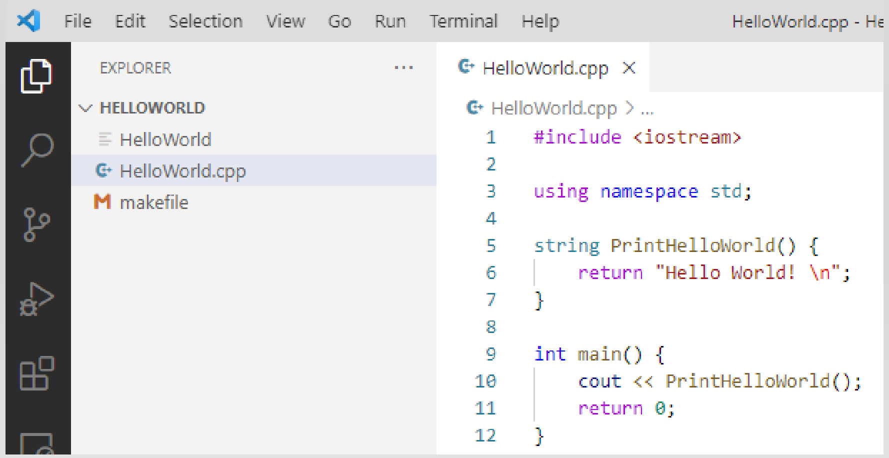
```
all:ll$      ## here the $ is end of lin
e ...
$
ll:ll.c   $
^Igcc  -c  -Wall -Werror -02 c.c ll.c  -
o  ll  $@  $<$
## the ^I above means a tab was there be
fore the action part, so this line is ok
 .
 $
clean :$
   \rm -fr ll$
## see here there is no ^I which means ,
 tab is not present ....
## in this case you need to open the fil
e again and edit/ensure a tab
## starts the action part
```

Try Google Test Your Code

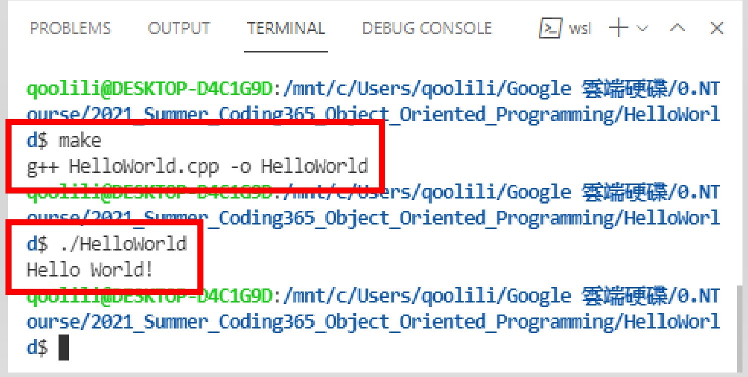
- As we already discussed, big company usually intergrade your code to the beta/release code through continuous integration
- Before integration, testing need to be carried out
  - Make sure your code will do what it aims to
  - Make sure your code will not mess up existing code
  - This is very common, even for senior engineers ! So, always do testing !
- In this course, we will use google test library for testing your functions.

A. Let's rewrite your HelloWorld to function-based from

1. Type "make" to recompile
2. Run the compiled file again -> "./HelloWorld"
3. Make sure the result is the same



```cpp
#include <iostream>

using namespace std;

string PrintHelloWorld() {
    return "Hello World! \n";
}

int main() {
    cout << PrintHelloWorld();
    return 0;
}
```

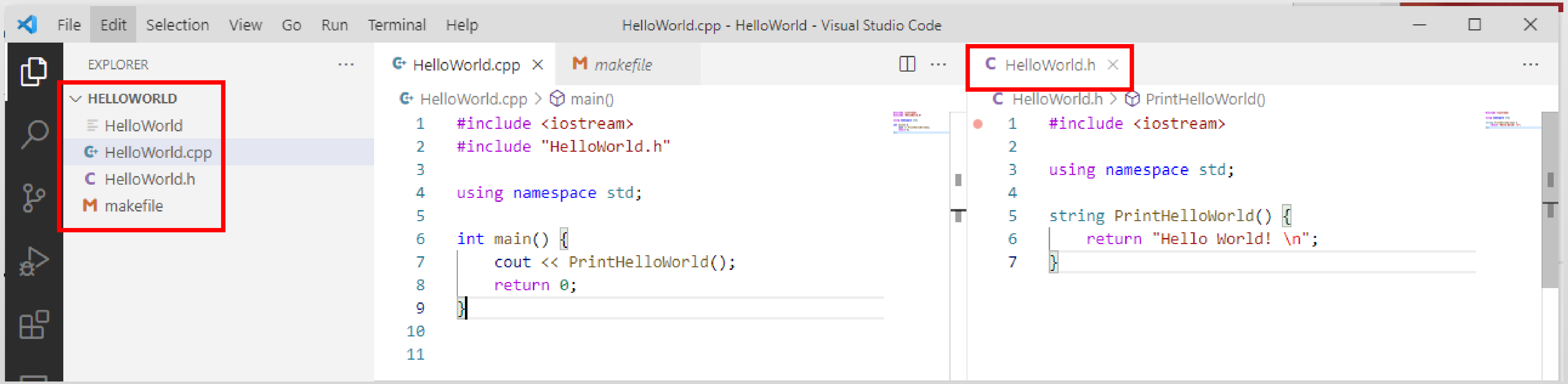


```
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NT
ourse/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorl
d$ make
g++ HelloWorld.cpp -o HelloWorld
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NT
ourse/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorl
d$ ./HelloWorld
Hello World!
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NT
ourse/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorl
d$ 
```

Try Google Test Your Code (Cont'd)

B. Move the function you just write to “HelloWorld.h” and include “HelloWorld.h” in “HelloWorld.c”

- Recompile & run again to make sure everything works fine


File Edit Selection View Go Run Terminal Help
HelloWorld.cpp - HelloWorld - Visual Studio Code
EXPLORER
HELLOWORLD
HelloWorld
HelloWorld.cpp
HelloWorld.h
makefile
HelloWorld.cpp
makefile
HelloWorld.h
HelloWorld.cpp > main()
HelloWorld.h > PrintHelloWorld()

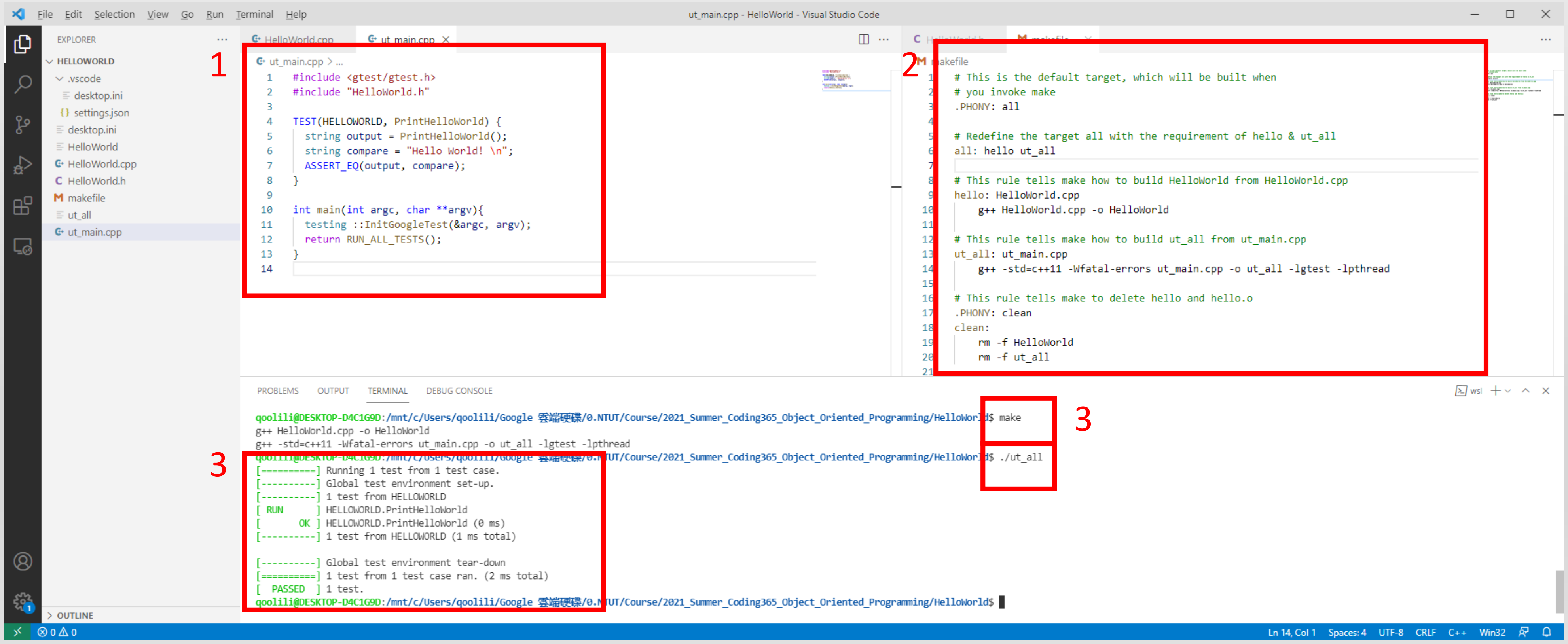
```cpp
#include <iostream>
#include "HelloWorld.h"

using namespace std;

int main() {
    cout << PrintHelloWorld();
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

string PrintHelloWorld() {
    return "Hello World! \n";
}
```

Try Google Test Your Code (Cont'd)

c. Prepare Google Test related code

1. New a file -> “ut_main.cpp” and Prepare the content as follows
2. Update the makefile as follows
3. Try make and run by typing in “./ut_all”



```cpp
#include <gtest/gtest.h>
#include "HelloWorld.h"

TEST(HELLOWORLD, PrintHelloWorld) {
    string output = PrintHelloWorld();
    string compare = "Hello World! \n";
    ASSERT_EQ(output, compare);
}

int main(int argc, char **argv){
    testing ::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

```makefile
# This is the default target, which will be built when
# you invoke make
.PHONY: all

# Redefine the target all with the requirement of hello & ut_all
all: hello ut_all

# This rule tells make how to build HelloWorld from HelloWorld.cpp
hello: HelloWorld.cpp
    g++ HelloWorld.cpp -o HelloWorld

# This rule tells make how to build ut_all from ut_main.cpp
ut_all: ut_main.cpp
    g++ -std=c++11 -Wfatal-errors ut_main.cpp -o ut_all -lgtest -lpthread

# This rule tells make to delete hello and hello.o
.PHONY: clean
clean:
    rm -f HelloWorld
    rm -f ut_all
```

```
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorld$ make
g++ HelloWorld.cpp -o HelloWorld
g++ -std=c++11 -Wfatal-errors ut_main.cpp -o ut_all -lgtest -lpthread
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorld$ ./ut_all
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from HELLOWORLD
[ RUN      ] HELLOWORLD.PrintHelloWorld
[       OK ] HELLOWORLD.PrintHelloWorld (0 ms)
[----------] 1 test from HELLOWORLD (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (2 ms total)
[  PASSED  ] 1 test.
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object_Oriented_Programming/HelloWorld$
```

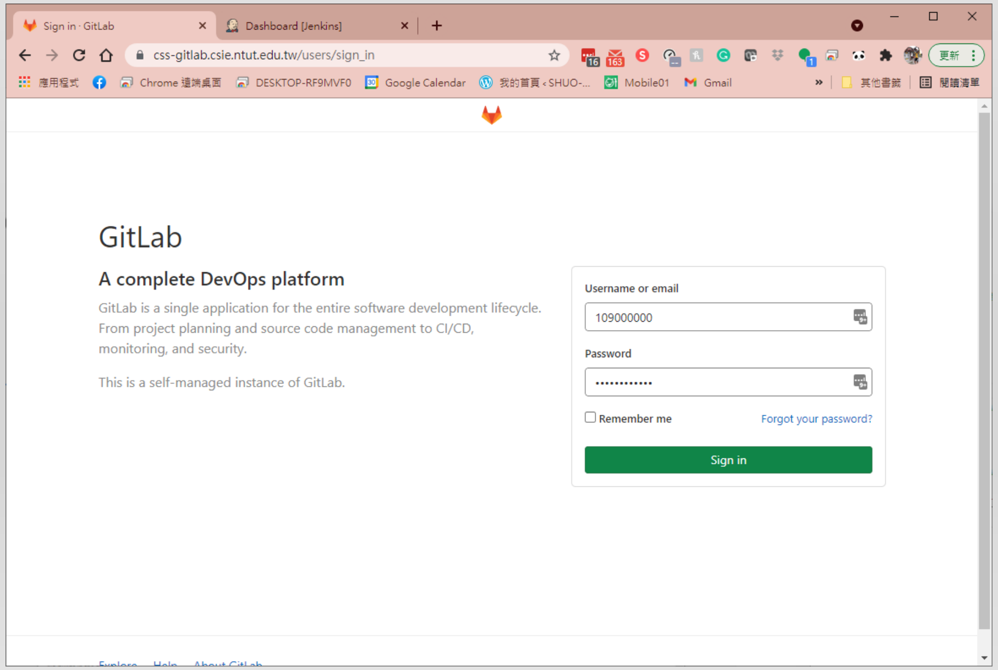
Submit Report

- Please see the word file for report template
- Strict format is enforced
  - Content format: should be set with 16pt row height, align to the left, font size 12pt.
  - Caption format: 18pt and Bold font.
  - Font format: Times New Roman.
  - Figure: center with single line row height.
  - Change the title to your student ID and name in Chinese. If you don‘t has ID, just leave it blank.
  - Upload pdf file with the file name format : **OOP_HW00_109000000.pdf** (change to your student ID) If you don‘t has ID, fill your name instead.
  - Remove the line starting with //.
  - Remove format guide part before uploading.

Part II – Before 10/3 24:00

Go check GitLab and Jenkins websites

- After testing our code locally, we now need to upload our code to GitLab server for triggering continuous integration (default password: 12345678. Change it after login, loss points if you did not change it TODAY.)
- Try login at following two websites
  - https://css-gitlab.csie.ntut.edu.tw/users/sign_in
  - https://css-lab.csie.ntut.edu.tw/jenkins/



GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

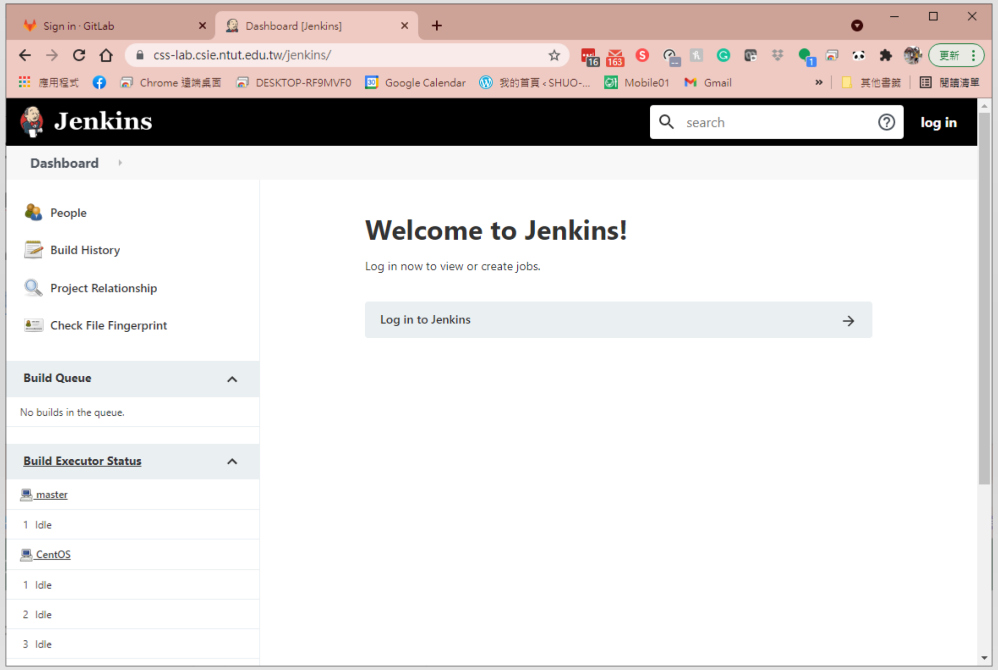
This is a self-managed instance of GitLab.

Username or email: 109000000

Password: ••••••••••••

☐ Remember me — Forgot your password?

Sign in



Welcome to Jenkins!

Log in now to view or create jobs.

Log in to Jenkins

Build Queue

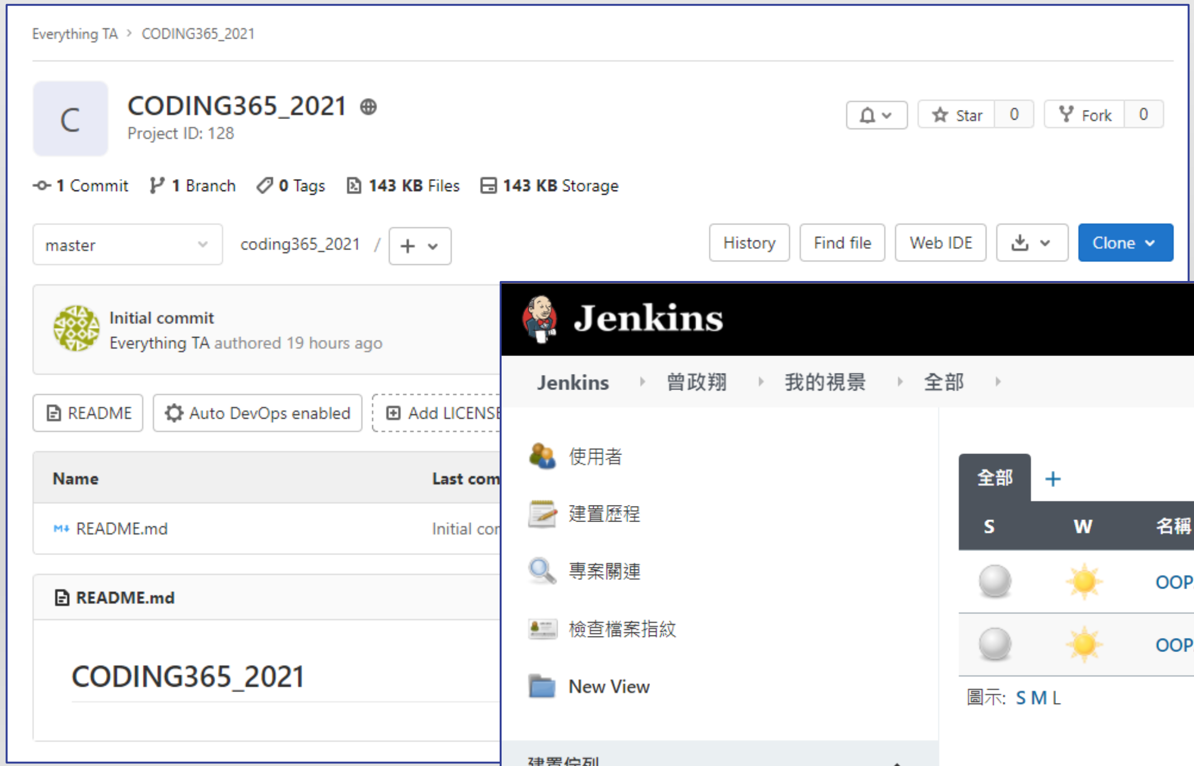
No builds in the queue.

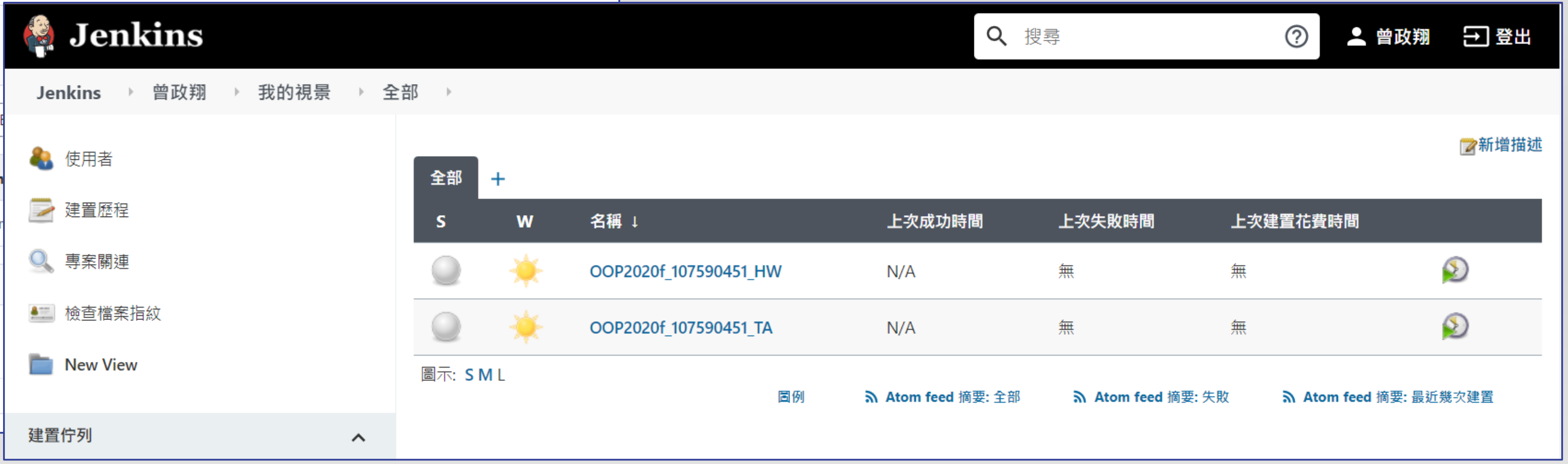
Build Executor Status

master
1 Idle
CentOS
1 Idle
2 Idle
3 Idle

Go check GitLab and Jenkins websites (Cont'd)

- All Setting has been set
  - On Gitlab, there is one repository. -> It's a remote repository for uploading your code
    - Git push to this repository will trigger jobs on Jenkins
  - On Jenkins, there is two jobs. -> Used for running testing on your code
    - HW will run the test cases you wrote. TA will run the test cases provided by us.

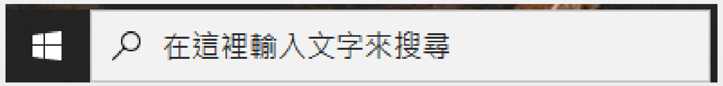




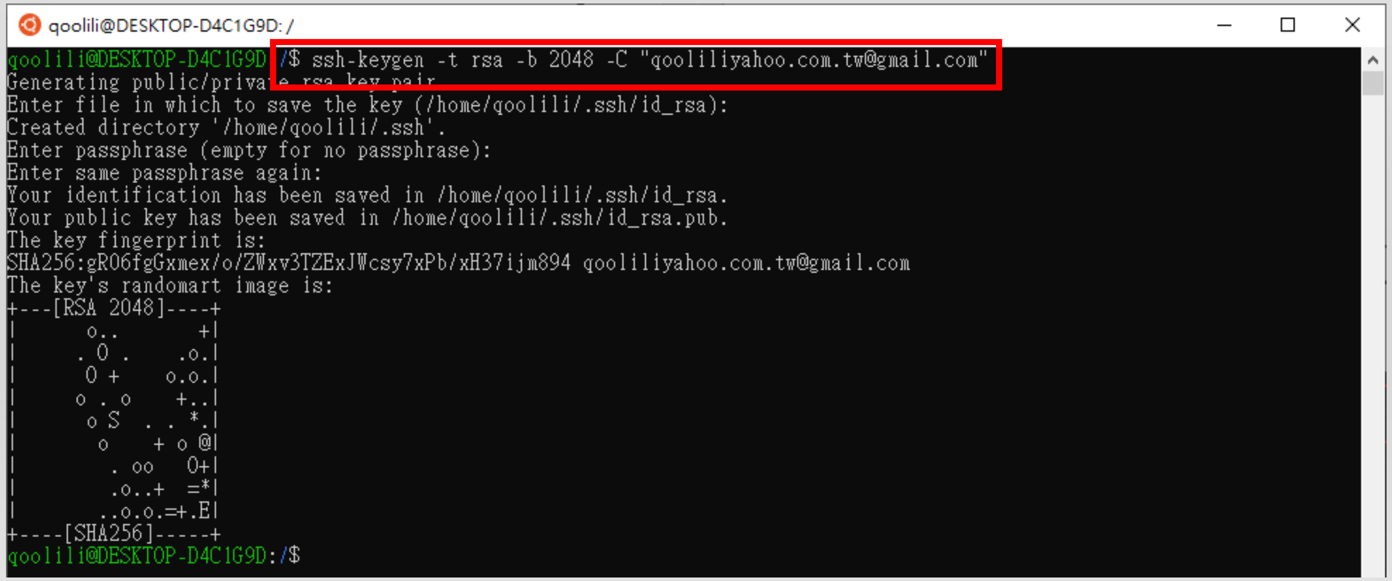
S	W	名稱 ↓	上次成功時間	上次失敗時間	上次建置花費時間
●	☀	OOP2020f_107590451_HW	N/A	無	無
●	☀	OOP2020f_107590451_TA	N/A	無	無

Setup ssh key for git and Using git cmd

1. Generate a ssh key first locally in your bash terminal
   - If forget how to open this windows by searching, just search bash in



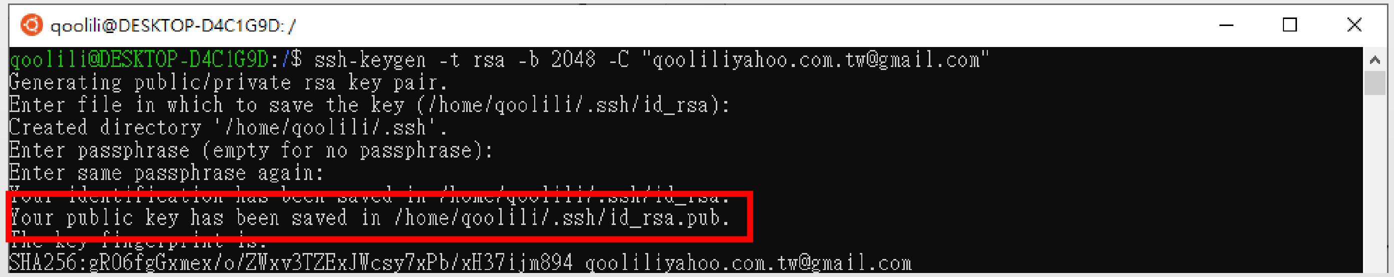
2. Enter ssh-keygen -t rsa -b 2048 -C "email@example.com"
   - Change email@example.com to your email
   - Press enter for using default location
   - Leave the your passphrase (secret token) BLANK, then enter
   - Then, you should see the following



```
qoolili@DESKTOP-D4C1G9D:/$ ssh-keygen -t rsa -b 2048 -C "qooliliyahoo.com.tw@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/qoolili/.ssh/id_rsa):
Created directory '/home/qoolili/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/qoolili/.ssh/id_rsa.
Your public key has been saved in /home/qoolili/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:gR06fgGxmex/o/ZWxv3TZExJWcsy7xPb/xH37ijm894 qooliliyahoo.com.tw@gmail.com
The key's randomart image is:
+---[RSA 2048]----+
|      o..       +|
|     . O .    .o.|
|      O +    o.o.|
|     o . o    +..|
|      o S  . . *.|
|       o    + o @|
|        . oo   O+|
|        .o..+  =*|
|       ..o.o.=+.E|
+----[SHA256]-----+
qoolili@DESKTOP-D4C1G9D:/$
```

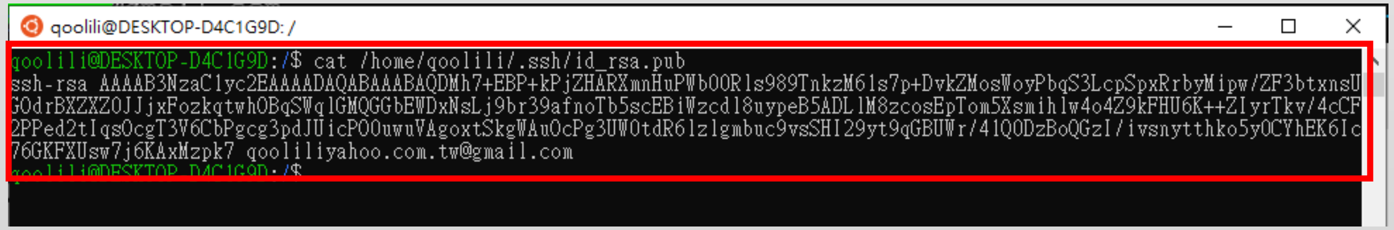
Setup ssh key for git and Using git cmd (Cont'd)

3. Print the public key through cat command
   - The path of your public key can be found as follows



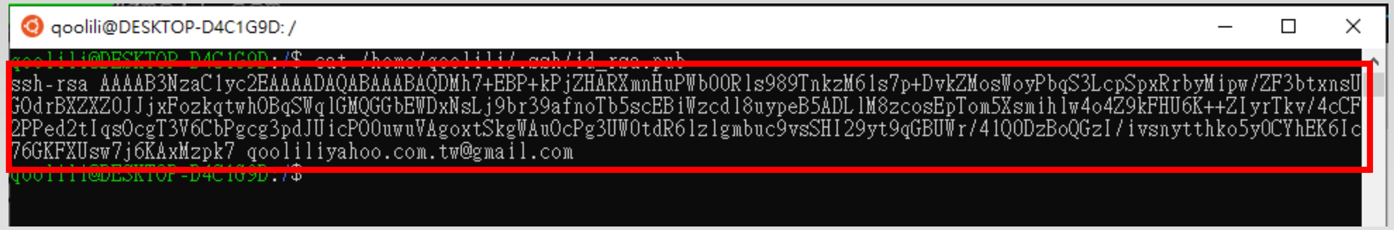
```
qoolili@DESKTOP-D4C1G9D:/$ ssh-keygen -t rsa -b 2048 -C "qooliliyahoo.com.tw@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/qoolili/.ssh/id_rsa):
Created directory '/home/qoolili/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/qoolili/.ssh/id_rsa.
Your public key has been saved in /home/qoolili/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:gR06fgGxmex/o/ZWxv3TZExJWcsy7xPb/xH37ijm894 qooliliyahoo.com.tw@gmail.com
```

   - Then, type cat /home/{see your path above}/.ssh/id_rsa.pub



```
qoolili@DESKTOP-D4C1G9D:/$ cat /home/qoolili/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMh7+EBP+kPjZHARXmnHuPWb0ORls989TnkzM61s7p+DvkZMosWoyPbqS3LcpSpxRrbyMipw/ZF3btxnsU
GOdrBXZXZ0JJjxFozkqtwhOBqSWqlGMQGGbEWDxNsLj9br39afnoTb5scEBiWzcdl8uypeB5ADL1M8zcosEpTom5Xsmihlw4o4Z9kFHU6K++ZIyrTkv/4cCF
2PPed2tIqsOcgT3V6CbPgcg3pdJUicPOOuwuVAgoxtSkgWAuOcPg3UW0tdR6lzlgmbuc9vsSHI29yt9qGBUWr/41Q0DzBoQGzI/ivsnytthko5y0CYhEK6Ic
76GKFXUsw7j6KAxMzpk7 qooliliyahoo.com.tw@gmail.com
qoolili@DESKTOP-D4C1G9D:/$
```

   - Copy the text starting with ssh-rsa and ending with your email (Be careful of extra blank at the end)

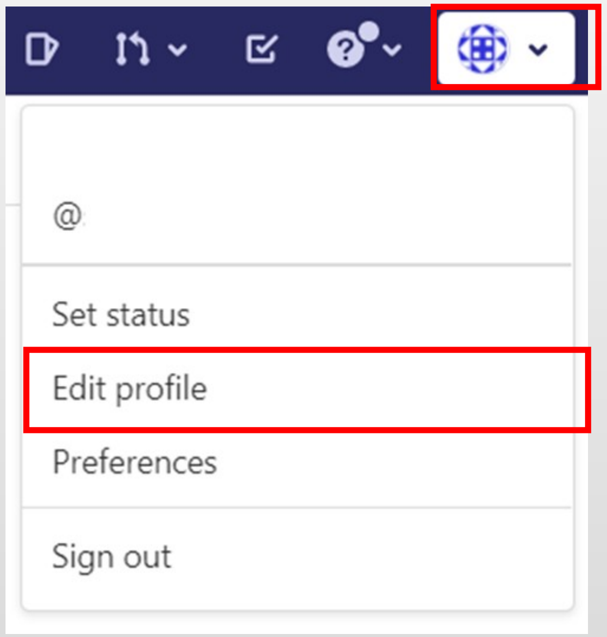


```
qoolili@DESKTOP-D4C1G9D:/$ cat /home/qoolili/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMh7+EBP+kPjZHARXmnHuPWb0ORls989TnkzM61s7p+DvkZMosWoyPbqS3LcpSpxRrbyMipw/ZF3btxnsU
GOdrBXZXZ0JJjxFozkqtwhOBqSWqlGMQGGbEWDxNsLj9br39afnoTb5scEBiWzcdl8uypeB5ADL1M8zcosEpTom5Xsmihlw4o4Z9kFHU6K++ZIyrTkv/4cCF
2PPed2tIqsOcgT3V6CbPgcg3pdJUicPOOuwuVAgoxtSkgWAuOcPg3UW0tdR6lzlgmbuc9vsSHI29yt9qGBUWr/41Q0DzBoQGzI/ivsnytthko5y0CYhEK6Ic
76GKFXUsw7j6KAxMzpk7 qooliliyahoo.com.tw@gmail.com
qoolili@DESKTOP-D4C1G9D:/$
```

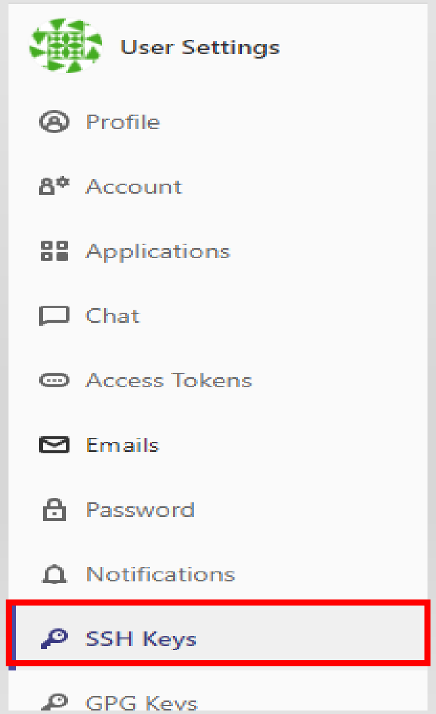
Setup ssh key for git and Using git cmd (Cont'd)

1. On the right-upper corner, click your icon -> Settings/Edit Profile
2. Click "SSH Keys" on the left
3. Paste the text you copied to the text box, then click add key
4. You're all set

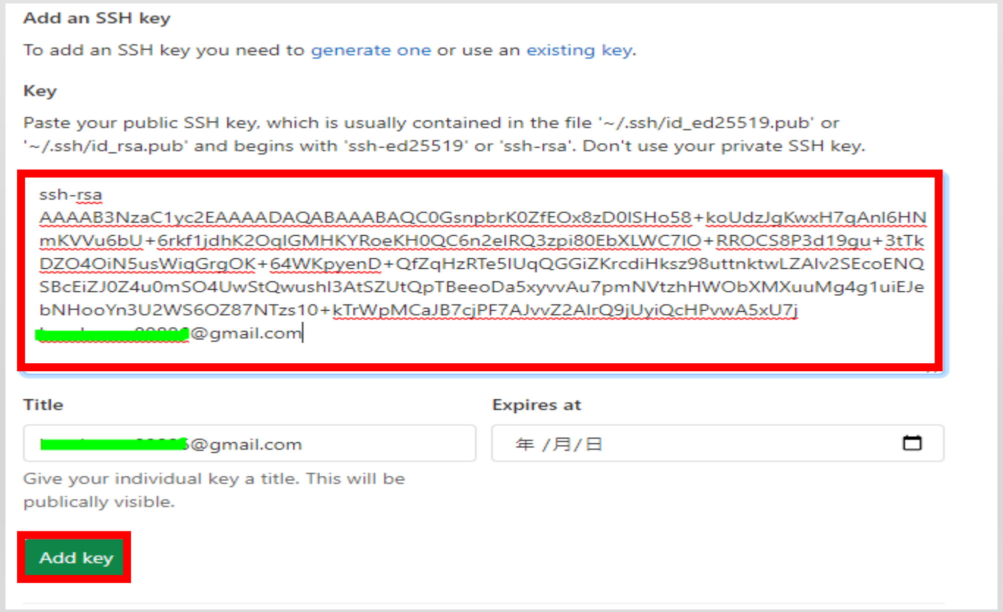
1



2



3



Add an SSH key

To add an SSH key you need to generate one or use an existing key.

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC0GsnpbrK0ZfEOx8zD0ISHo58+koUdzJgKwxH7qAnl6HN
mKVVu6bU+6rkf1jdhK2OqlGMHKYRoeKH0QC6n2elRQ3zpi80EbXLWC7IO+RROCS8P3d19gu+3tTk
DZO4OiN5usWiqGrgOK+64WKpyenD+QfZqHzRTe5IUqQGGiZKrcdiHksz98uttnktwLZAlv2SEcoENQ
SBcEiZJ0Z4u0mSO4UwStQwushI3AtSZUtQpTBeeoDa5xyvvAu7pmNVtzhHWObXMXuuMg4g1uiEJe
bNHooYn3U2WS6OZ87NTzs10+kTrWpMCaJB7cjPF7AJvvZ2AlrQ9jUyiQcHPvwA5xU7j
[redacted]@gmail.com
```

Title: [redacted]@gmail.com

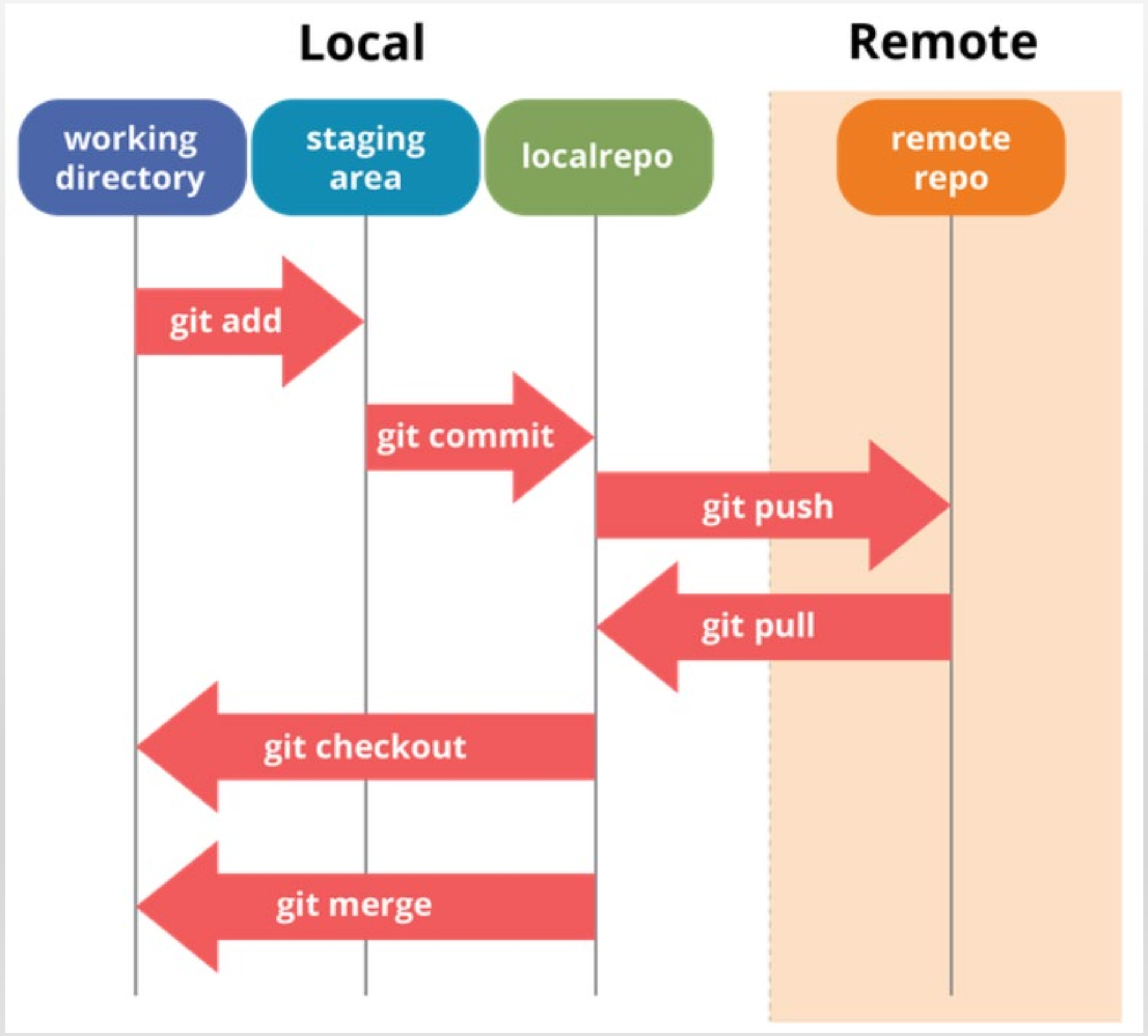
Expires at: 年 /月/日

Give your individual key a title. This will be publically visible.

Add key

Setup ssh key for git and Using git cmd (Cont'd)

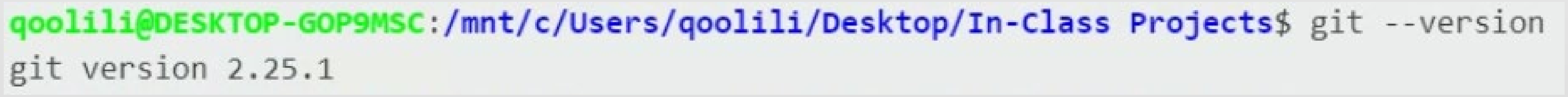
- Using repository to track the revised history of files and folders
  - Local repository (本地)
  - Remote repository (遠端)



https://blog.techbridge.cc/2018/01/17/learning-programming-and-coding-with-python-git-and-github-tutorial/

Git push the HelloWorld

- If you haven’t installed git, please follow …
  1. Open bash
  2. Install git with -> sudo apt-get install git
  3. Check you installation -> git --version

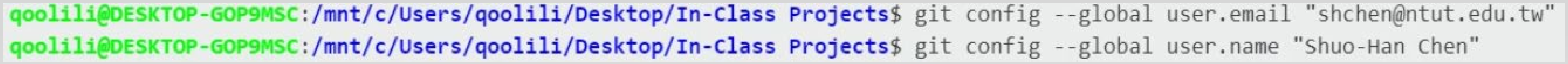


```
qoolili@DESKTOP-GOP9MSC:/mnt/c/Users/qoolili/Desktop/In-Class Projects$ git --version
git version 2.25.1
```

  4. Set information (Change the email and name)

     -> git config --global user.email “t109000000@ntut.edu.tw”

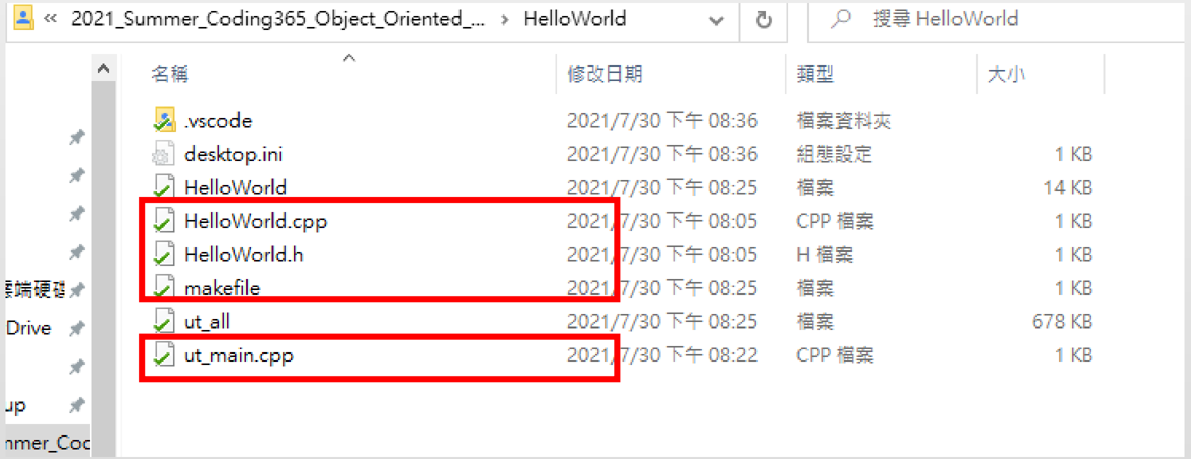
     -> git config --global user.name “Shuo-Han Chen”



```
qoolili@DESKTOP-GOP9MSC:/mnt/c/Users/qoolili/Desktop/In-Class Projects$ git config --global user.email "shchen@ntut.edu.tw"
qoolili@DESKTOP-GOP9MSC:/mnt/c/Users/qoolili/Desktop/In-Class Projects$ git config --global user.name "Shuo-Han Chen"
```

Git push the HelloWorld (Cont'd)

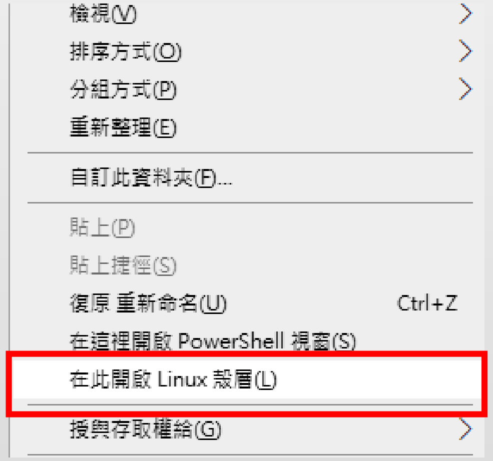
1. In your HelloWorld folder, press shift and right click at anywhere
2. Then, select Open Linux bash



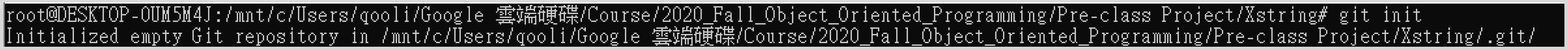
« 2021_Summer_Coding365_Object_Oriented_... › HelloWorld

搜尋 HelloWorld

名稱	修改日期	類型	大小
.vscode	2021/7/30 下午 08:36	檔案資料夾	
desktop.ini	2021/7/30 下午 08:36	組態設定	1 KB
HelloWorld	2021/7/30 下午 08:25	檔案	14 KB
HelloWorld.cpp	2021/7/30 下午 08:05	CPP 檔案	1 KB
HelloWorld.h	2021/7/30 下午 08:05	H 檔案	1 KB
makefile	2021/7/30 下午 08:25	檔案	1 KB
ut_all	2021/7/30 下午 08:25	檔案	678 KB
ut_main.cpp	2021/7/30 下午 08:22	CPP 檔案	1 KB

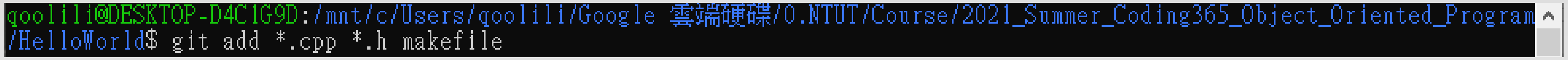


2. Initialize local repository -> sudo git init



```
root@DESKTOP-OUM5M4J:/mnt/c/Users/qooli/Google 雲端硬碟/Course/2020_Fall_Object_Oriented_Programming/Pre-class Project/Xstring# git init
Initialized empty Git repository in /mnt/c/Users/qooli/Google 雲端硬碟/Course/2020_Fall_Object_Oriented_Programming/Pre-class Project/Xstring/.git/
```

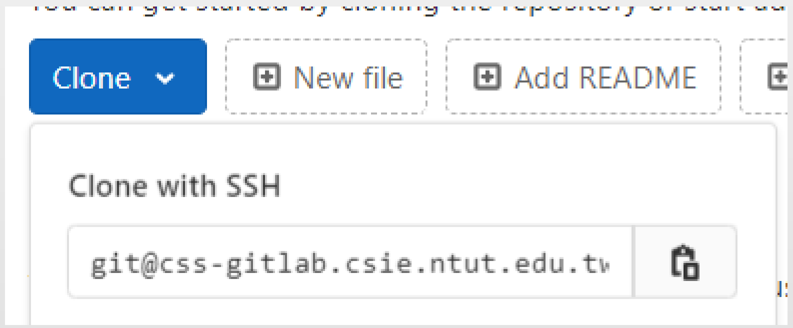
3. Add files to be uploaded into local repository -> git add
   - Add only *.cpp, *.h, makefile



```
qoolili@DESKTOP-D4C1G9D:/mnt/c/Users/qoolili/Google 雲端硬碟/0.NTUT/Course/2021_Summer_Coding365_Object_Oriented_Program
/HelloWorld$ git add *.cpp *.h makefile
```

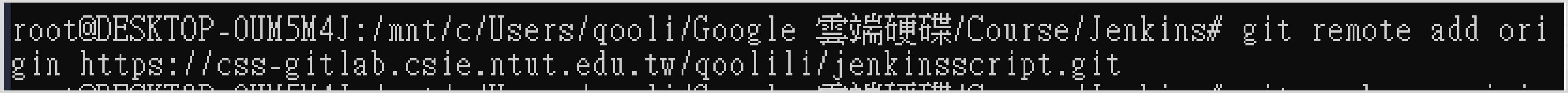
Set Remote Repository

4. Copy your git link from our gitlab



```
git@css-gitlab.csie.ntut.edu.tw
```

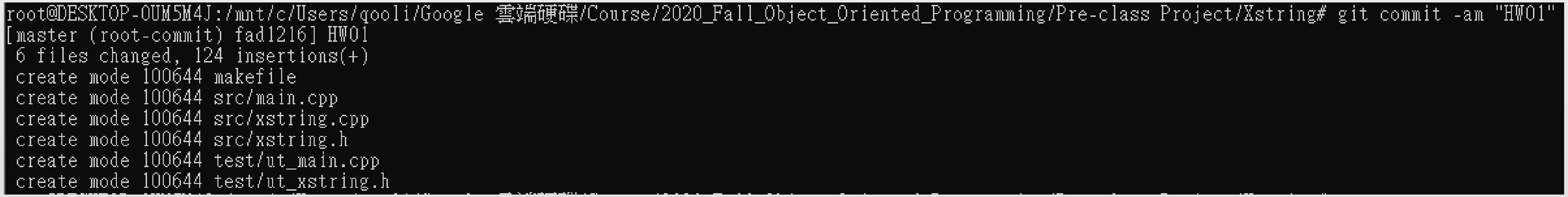
5. Add remove -> sudo git remote add origin git@....



```
root@DESKTOP-OUM5M4J:/mnt/c/Users/qooli/Google 雲端硬碟/Course/Jenkins# git remote add ori
gin https://css-gitlab.csie.ntut.edu.tw/qoolili/jenkinsscript.git
```

Git Commit & Push

6. Commit files to local repository -> git commit -am “HW01”
   - -a : commit all changed files , -m “提交訊息” : specify commit message
   - This command will make changes to your local repository

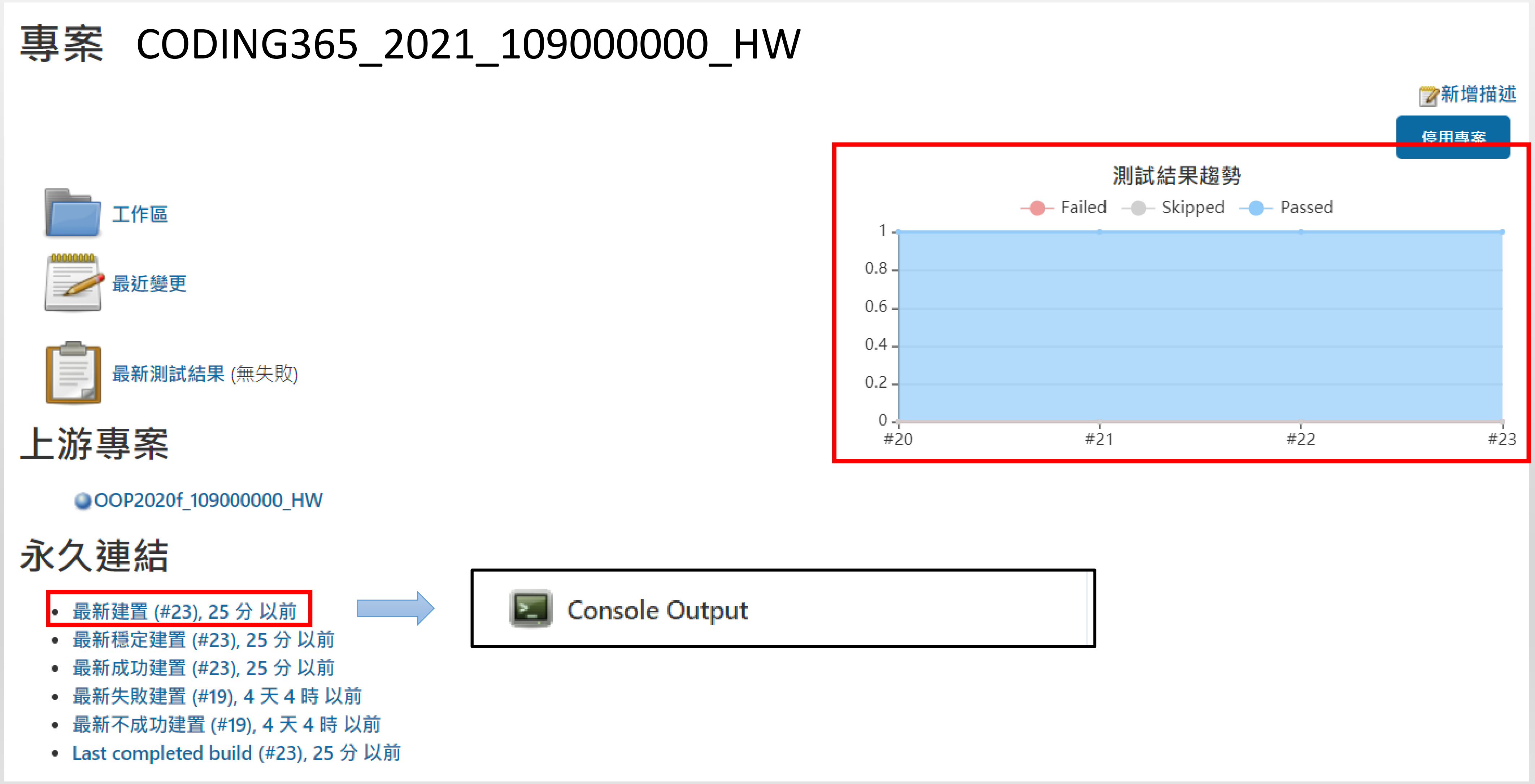


```
root@DESKTOP-0UM5M4J:/mnt/c/Users/qooli/Google 雲端硬碟/Course/2020_Fall_Object_Oriented_Programming/Pre-class Project/Xstring# git commit -am "HW01"
[master (root-commit) fad1216] HW01
 6 files changed, 124 insertions(+)
 create mode 100644 makefile
 create mode 100644 src/main.cpp
 create mode 100644 src/xstring.cpp
 create mode 100644 src/xstring.h
 create mode 100644 test/ut_main.cpp
 create mode 100644 test/ut_xstring.h
```

7. Push Files onto Gitlab Project -> git push -u origin master
   - This command only used for the first-time push
   - Next time, you only need git push
   - Go to your project on our Gitlab, you should see files on your git
   - And it will automatically trigger Jenkins

Git push the HelloWorld (Cont'd)

- Go check your result on Jenkins



專案 CODING365_2021_109000000_HW

新增描述

停用專案

工作區

最近變更

最新測試結果 (無失敗)

上游專案

OOP2020f_109000000_HW

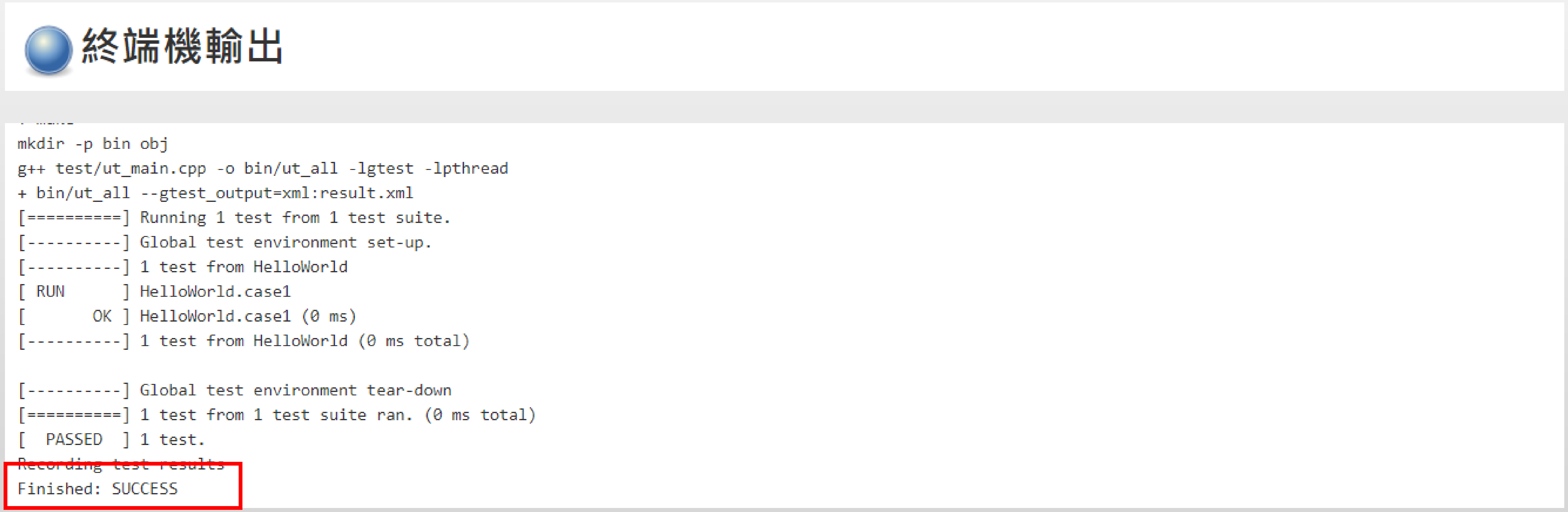
永久連結

- 最新建置 (#23), 25 分 以前
- 最新穩定建置 (#23), 25 分 以前
- 最新成功建置 (#23), 25 分 以前
- 最新失敗建置 (#19), 4 天 4 時 以前
- 最新不成功建置 (#19), 4 天 4 時 以前
- Last completed build (#23), 25 分 以前

Console Output

Git push the HelloWorld (Cont'd)

- For now, only the job with HW is triggered. The job with TA will be used in the future.



終端機輸出

```
mkdir -p bin obj
g++ test/ut_main.cpp -o bin/ut_all -lgtest -lpthread
+ bin/ut_all --gtest_output=xml:result.xml
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from HelloWorld
[ RUN      ] HelloWorld.case1
[       OK ] HelloWorld.case1 (0 ms)
[----------] 1 test from HelloWorld (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (0 ms total)
[  PASSED  ] 1 test.
Recording test results
Finished: SUCCESS
```

Please contact TA if you has any trouble.