Software Testing
Project1
Continuous Integration

Tutorial

Junit + Github + Jenkins+JaCoCo

Instructor: 王凡教授

TA: 鄭大容

Email: R06943080@ntu.edu.tw

Unit Test

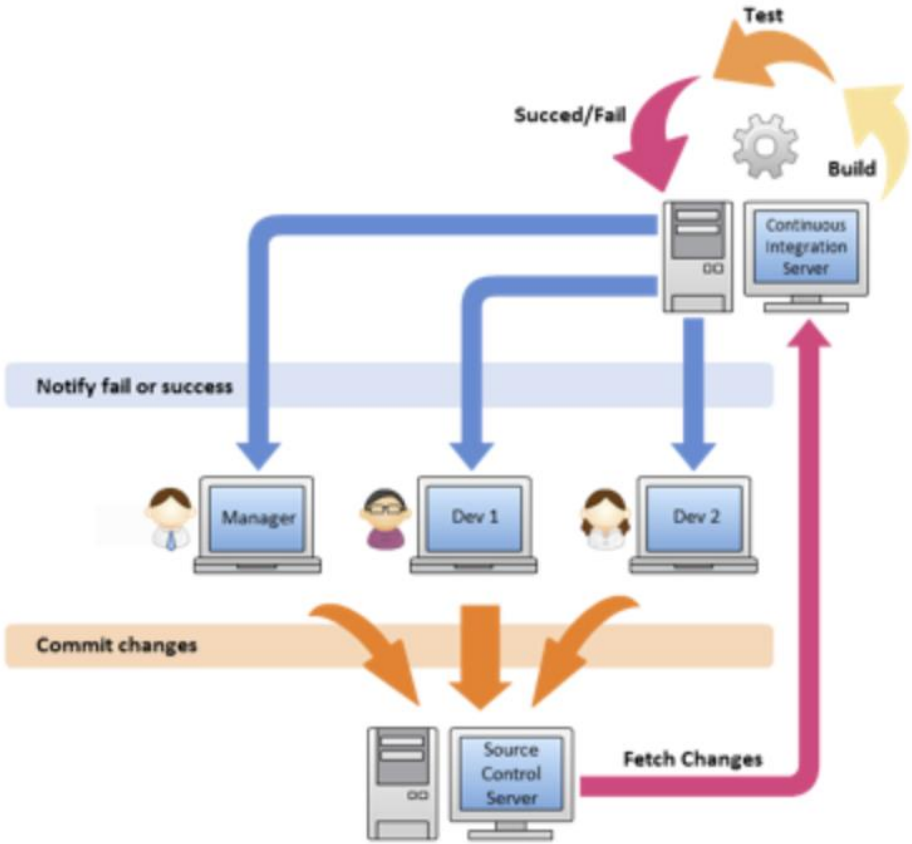
- 對於程式碼中的單元如class, method進行測試的方法。
- 對於大型專案的單元測試，已有現成的IDE工具可供使用。

  Ex. Eclipse

What is CI?

- Continuous integration is the method used to determine that the entire system is in the running state and the changes made have not lead to errors in some parts of the system.

How CI works?


Test
Succed/Fail
Build
Continuous Integration Server
Notify fail or success
Manager
Dev 1
Dev 2
Commit changes
Source Control Server
Fetch Changes

What is Jenkins?

- An open source continuous integration and build automation tool
- Helps automate tasks:
  - Building
  - Testing
  - Deploying

Goals

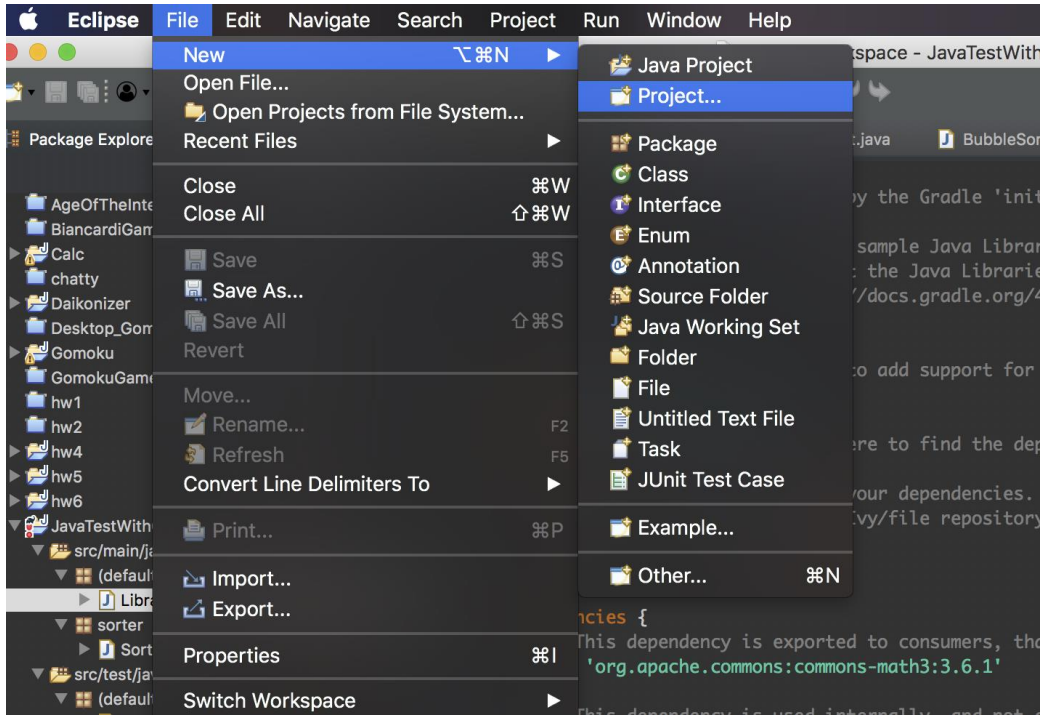
1. Create a Java project
2. Use an external library. (ex. Sort)
3. Writing testcases for this project
4. Create a public repo and push your work to Github
5. Setup Jenkins server
6. Automatic fetch and build your java project from Github on Jenkins
7. Coverage report with JaCoCo plugin
8. Find an external library and design your testcases

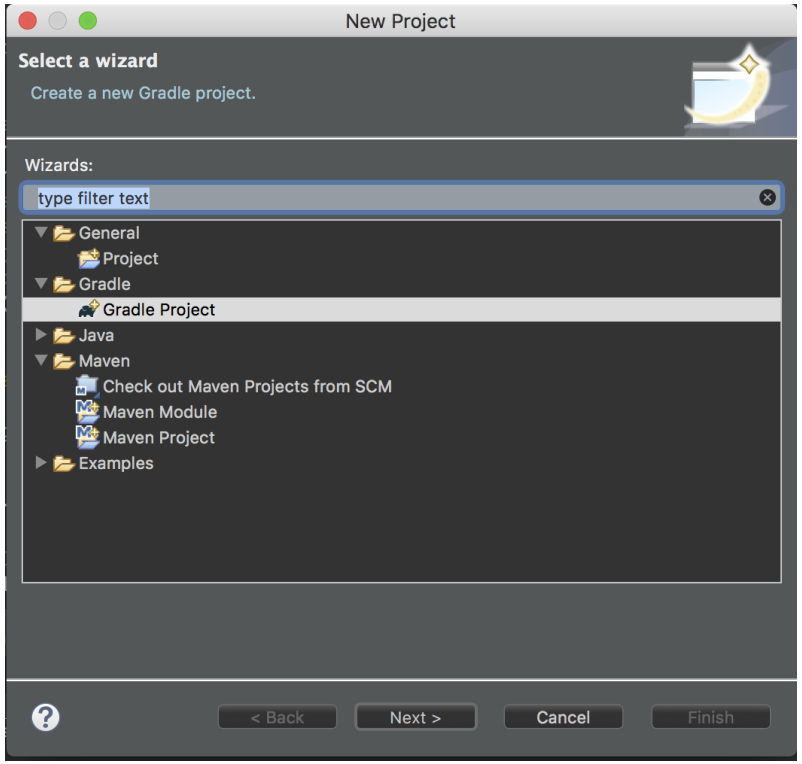
Prerequisite

- Ubuntu 16.04
  - This tutorial was written and tested on Ubuntu 16.04
  - There might be some difference if you use other OS.
- Install Java SE 8 first
  - For Ubuntu
    - `sudo add-apt-repository ppa:webupd8team/java`
    - `sudo apt-get update`
    - `sudo apt-get install oracle-java8-installer`
- Eclipse IDE

Goal 1: Create a Java project

- We use eclipse for demo:
  - Download here: https://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-ide-java-developers
- We'll be using Gradle to manage our Java project
- File >> New >> Project >> Select Gradle Project



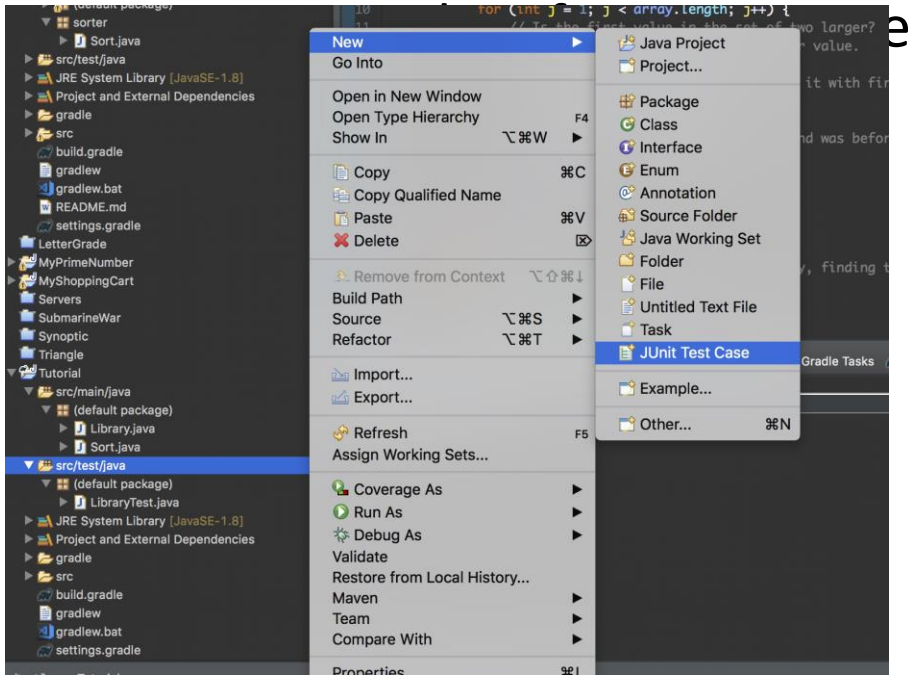


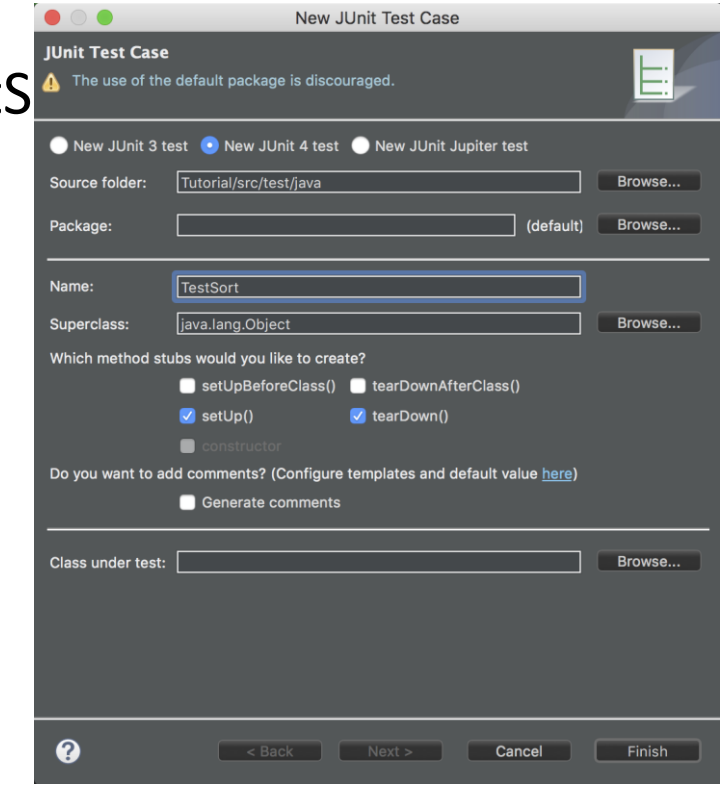
Goal 2: Use an external library

- In our demo:
  - We use a sort algorithm library: LINK
- Drag & Drop the file into `src/main/java`

Goal 3: Writing testcases

- In src/test/java >> Right click >> New >> Junit Testcase:





Goal 3: Writing testcases

- Let’s say, we’re testing on the Bubble sort algorithm
  - Method with decorator `@Test` will be executed while testing
  - You can import other decorator such as @BeforeClass, @AfterClass, @Ignore
  - See https://netbeans.org/kb/docs/java/junit-intro.html#Exercise_10
  - Using the external class Sort

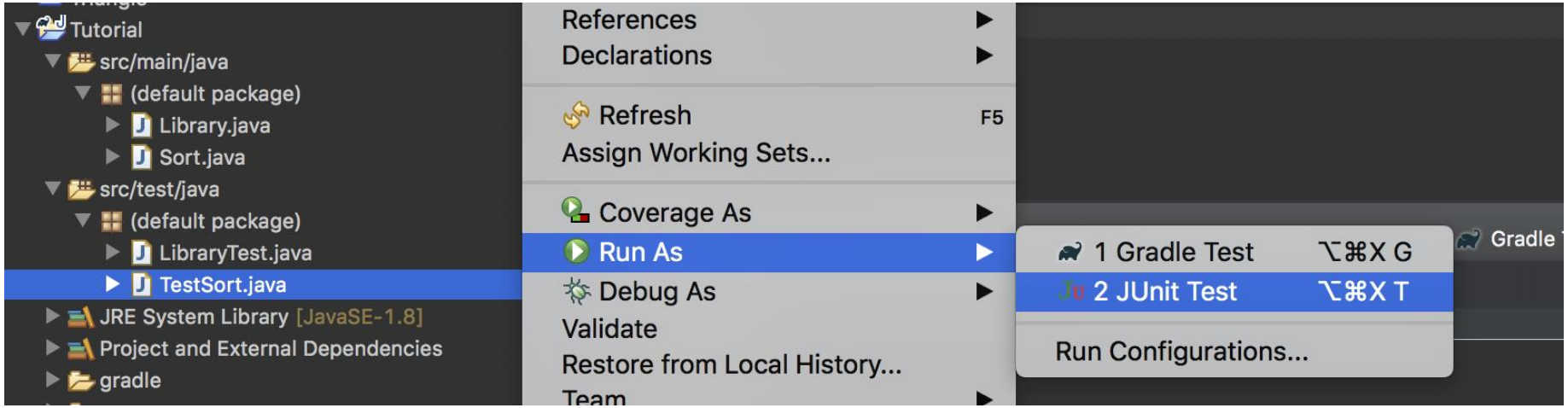
Goal 3: Writing testcases

- Example

```
1  package sorter;
2
3  import static org.junit.Assert.*;
4
5  import org.junit.After;
6  import org.junit.Before;
7  import org.junit.Test;
8
9  public class SortTest {
10
11     @Before
12     public void setUp() throws Exception {
13     }
14
15     @After
16     public void tearDown() throws Exception {
17     }
18
19     @Test
20     public void testBubble() {
21         System.out.println("bubble");
22         Sort sort = new Sort();
23         assertArrayEquals(new int[] {1,2,3}, sort.bubble(new int[] {3,1,2}));
24         assertArrayEquals(new int[] {1,2,3,4}, sort.bubble(new int[] {3,1,4,2}));
25         assertArrayEquals(new int[] {1,2,3,4,5}, sort.bubble(new int[] {5,3,1,4,2}));
26         assertArrayEquals(new int[] {1,2,3,4,5,6}, sort.bubble(new int[] {6,3,1,5,4,2}));
27         assertArrayEquals(new int[] {1,2,3,4,5,6,7}, sort.bubble(new int[] {3,7,1,6,4,2,5}));
28         assertArrayEquals(new int[] {1,2,3,4,5,6,7,8}, sort.bubble(new int[] {3,7,1,8,6,4,2,5}));
29         assertArrayEquals(new int[] {1,2,3,4,5,6,7,8,9}, sort.bubble(new int[] {3,7,9,1,8,6,4,2,5}));
30         assertArrayEquals(new int[] {1,2,3,4,5,6,7,8,9,10}, sort.bubble(new int[] {3,7,9,1,8,6,4,2,10,5}));
31     }
32
33     @Test
34     public void testSelection() {
35         System.out.println("selection");
36         Sort sort = new Sort();
37         assertArrayEquals(new int[] {1,2,3}, sort.selection(new int[] {3,1,2}));
38         assertArrayEquals(new int[] {1,2,3,4}, sort.selection(new int[] {3,1,4,2}));
39         assertArrayEquals(new int[] {1,2,3,4,5}, sort.selection(new int[] {5,3,1,4,2}));
40         assertArrayEquals(new int[] {1,2,3,4,5,6}, sort.selection(new int[] {6,3,1,5,4,2}));
41         assertArrayEquals(new int[] {1,2,3,4,5,6,7}, sort.selection(new int[] {3,7,1,6,4,2,5}));
```

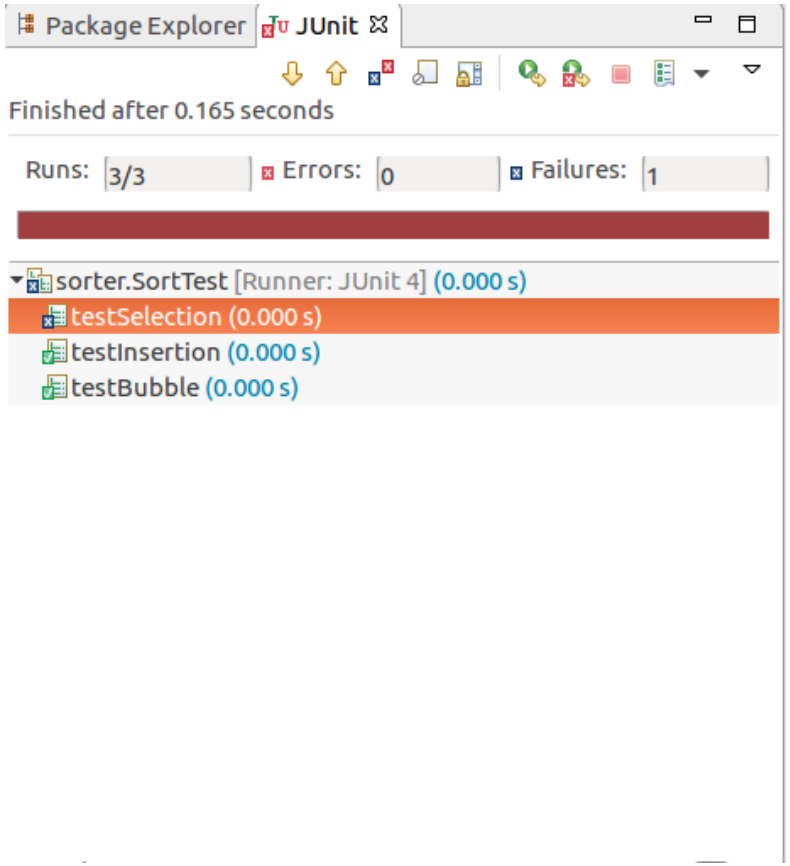
Goal 3: Writing testcases

- We can run the test locally
  - Right click on the testing class >> Run as >> Junit Test



Goal 3: Writing testcases

- Test Result:



Goal 3: Writing testcases

- Buggy method:

```
public int[] selection(int[] array) {
    // Go through the array once, from last place to first
    for (int i = array.length - 1; i > 0; i--) {
        // The highest number in the starting is always the first.
        int highest = array[0];
        int index = 0;
        // Cycle through all the unsorted numbers, to find the highest
        for (int j = 0; j < i; j++) {
            // Is this number higher than the largest number so far?
            if (array[j] > highest) {
                // Then replace the current record with this number.
                highest = array[j];
                // ...and store which index it's in for later.
                index = j;
            }
        }
        // After the highest unsorted number has been found, switch the highest number with the last unsorted number.
        int temp = array[i - 1] = highest;
        array[index] = temp;
    }
    return array;
}
```

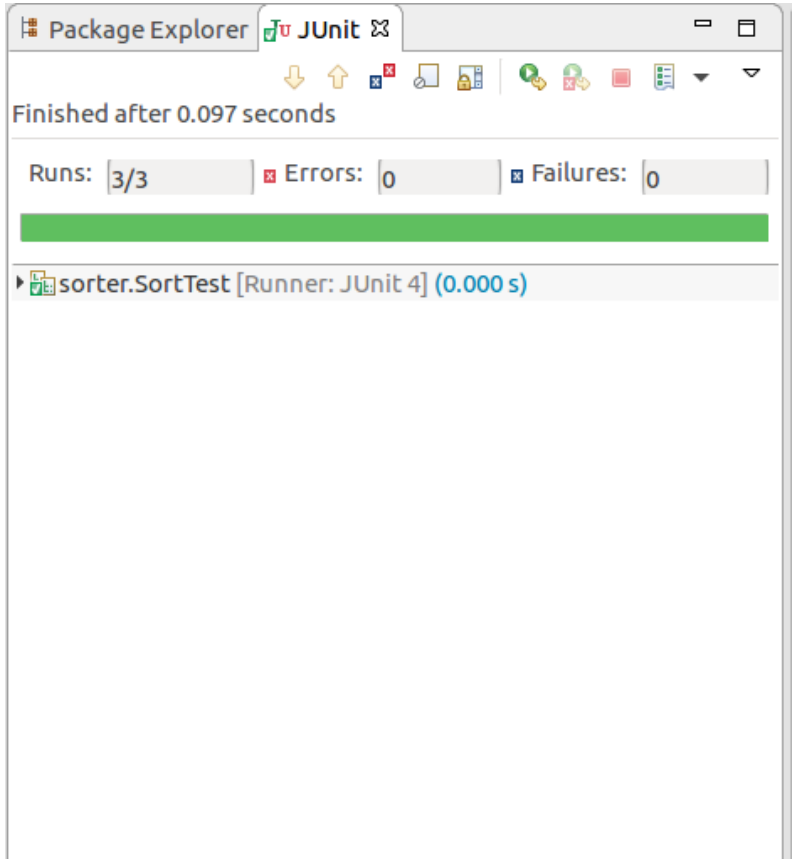
Goal 3: Writing testcases

- Replaced with correct codes:

```
26    public int[] selection(int[] array) {
27        // Go through the array once, from last place to first
28        for (int i = array.length - 1; i > 0; i--) {
29            // The highest number in the starting is always the first.
30            int highest = array[i];
31            int index = i;
32            // Cycle through all the unsorted numbers, to find the highest
33            for (int j = 0; j < i; j++) {
34                // Is this number higher than the largest number so far?
35                if (array[j] > highest) {
36                    // Then replace the current record with this number.
37                    highest = array[j];
38                    // ...and store which index it's in for later.
39                    index = j;
40                }
41            }
42            // After the highest unsorted number has been found, switch the highest number with the last unsorted number.
43            int temp = array[i];
44                       array[i] = array[index];
45            array[index] = temp;
46        }
47        return array;
48    }
```

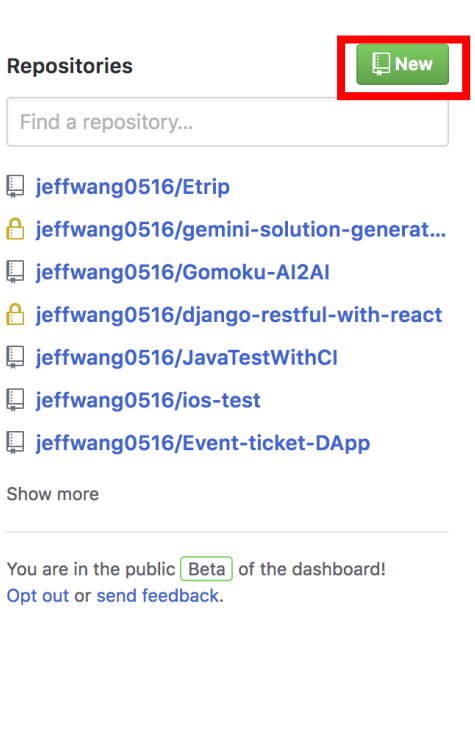
Goal 3: Writing testcases

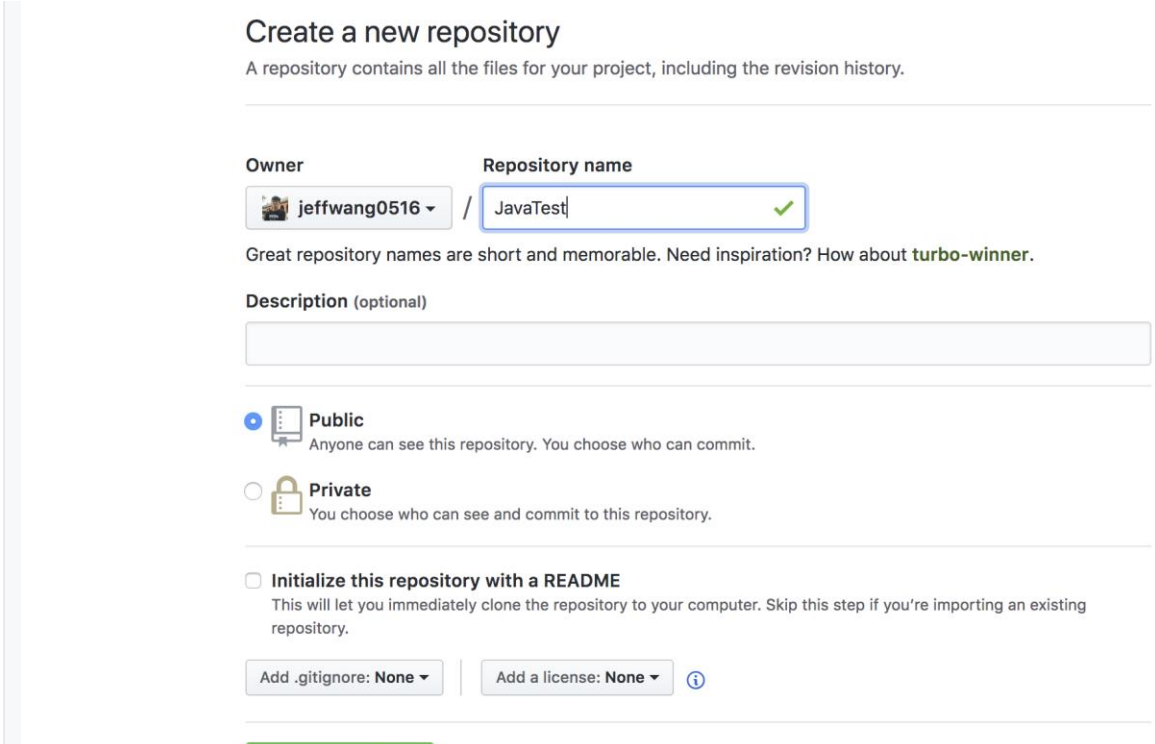
- Test result:



Goal 4: Create a public repo and push your work

- If you do not have a Github account, ***register first on https://github.com/***
- Create a new public repo



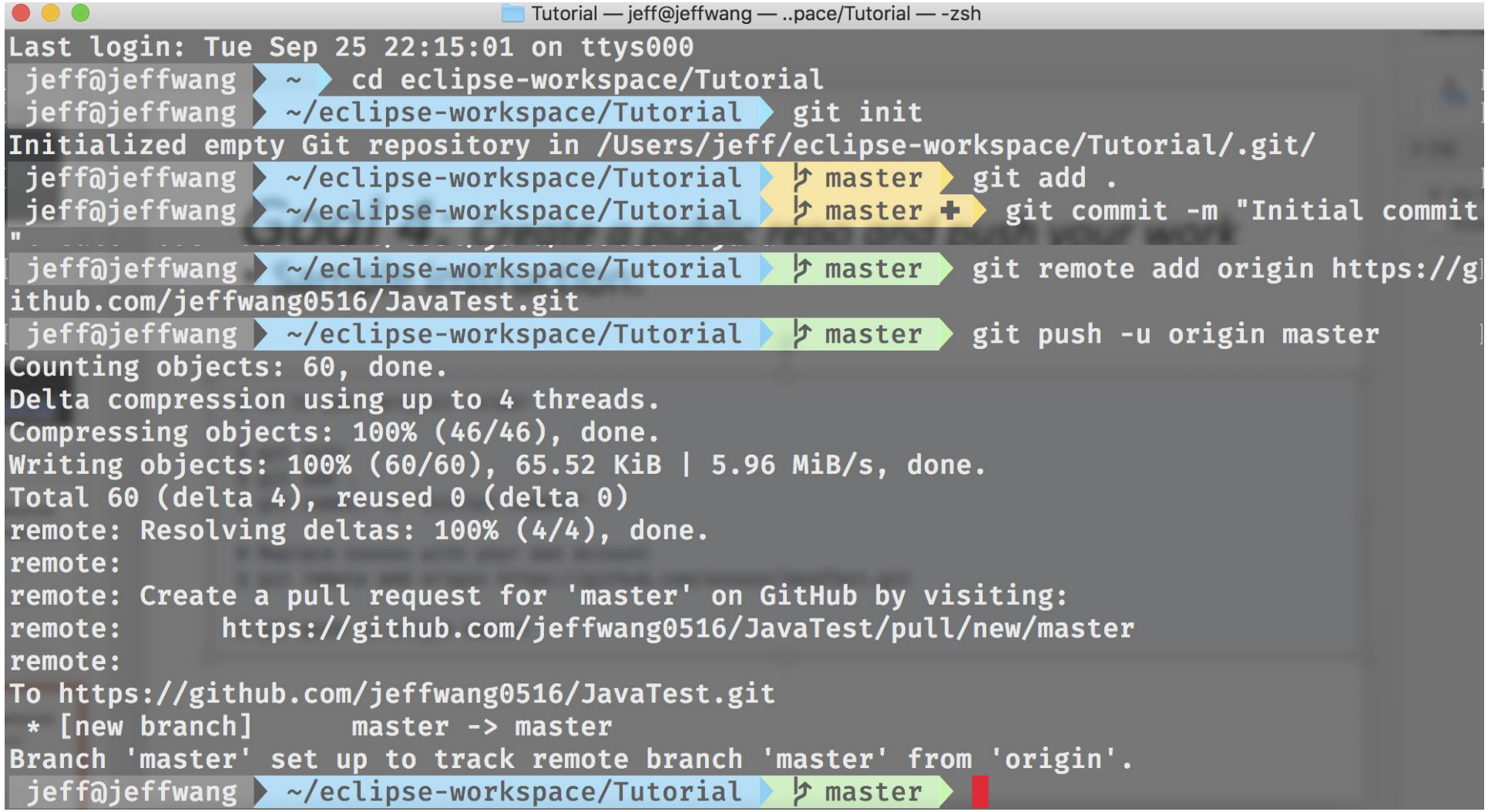


Goal 4: Create a public repo and push your work

- On your terminal, cd to your project folder
- If your not familiar with git: Tutorial
- Sample instruction:

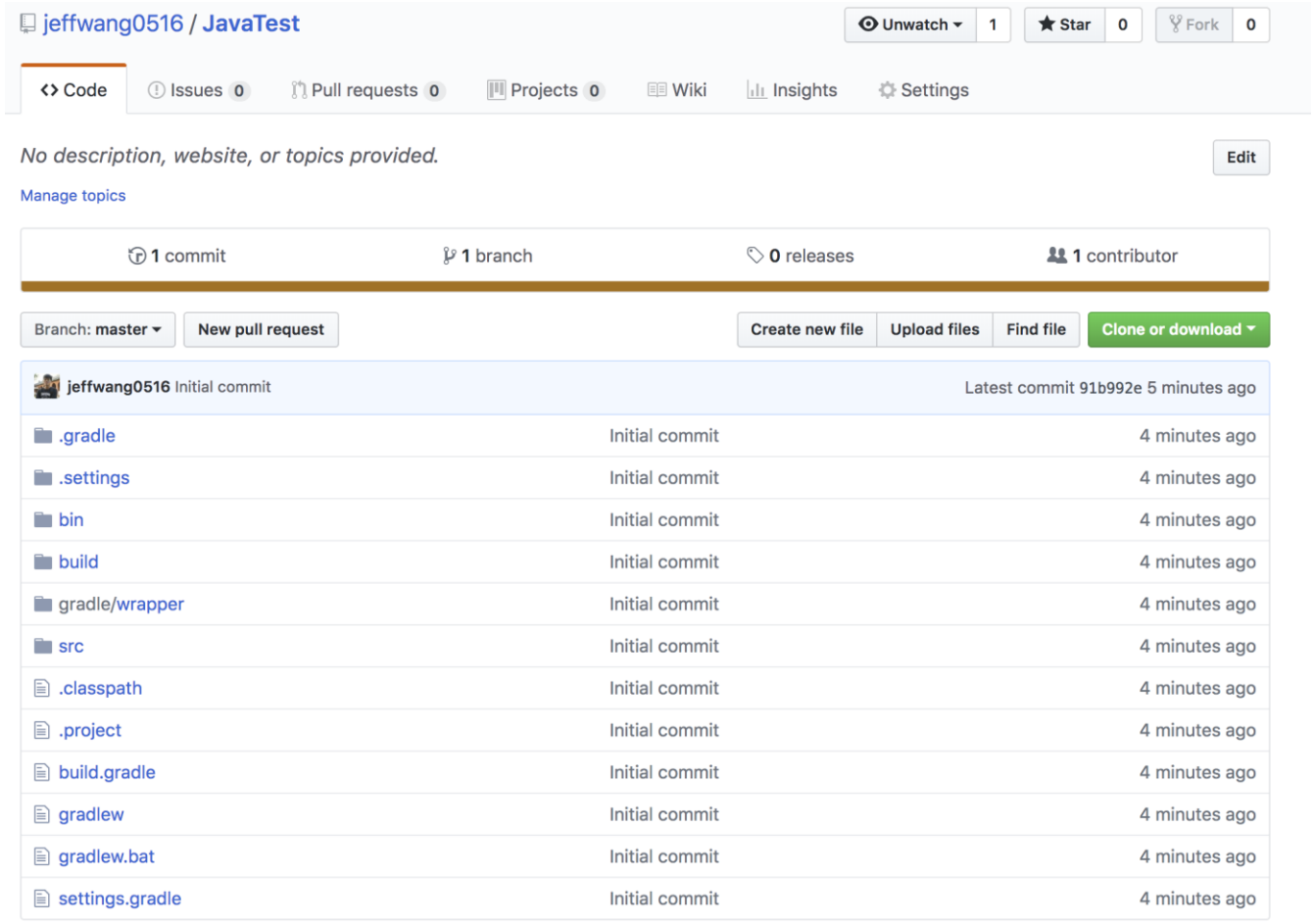
```
$ cd to your project folder

$ git init
$ git add .
$ git commit -m "Initial commit"

# Replace xxxxxx with your own account
$ git remote add origin https://github.com/xxxxxx/JavaTest.git

$ git push -u origin master
```

Goal 4: Terminal Snapshot



Goal 4: Create a public repo and push your work

- You're now able to see files on Github

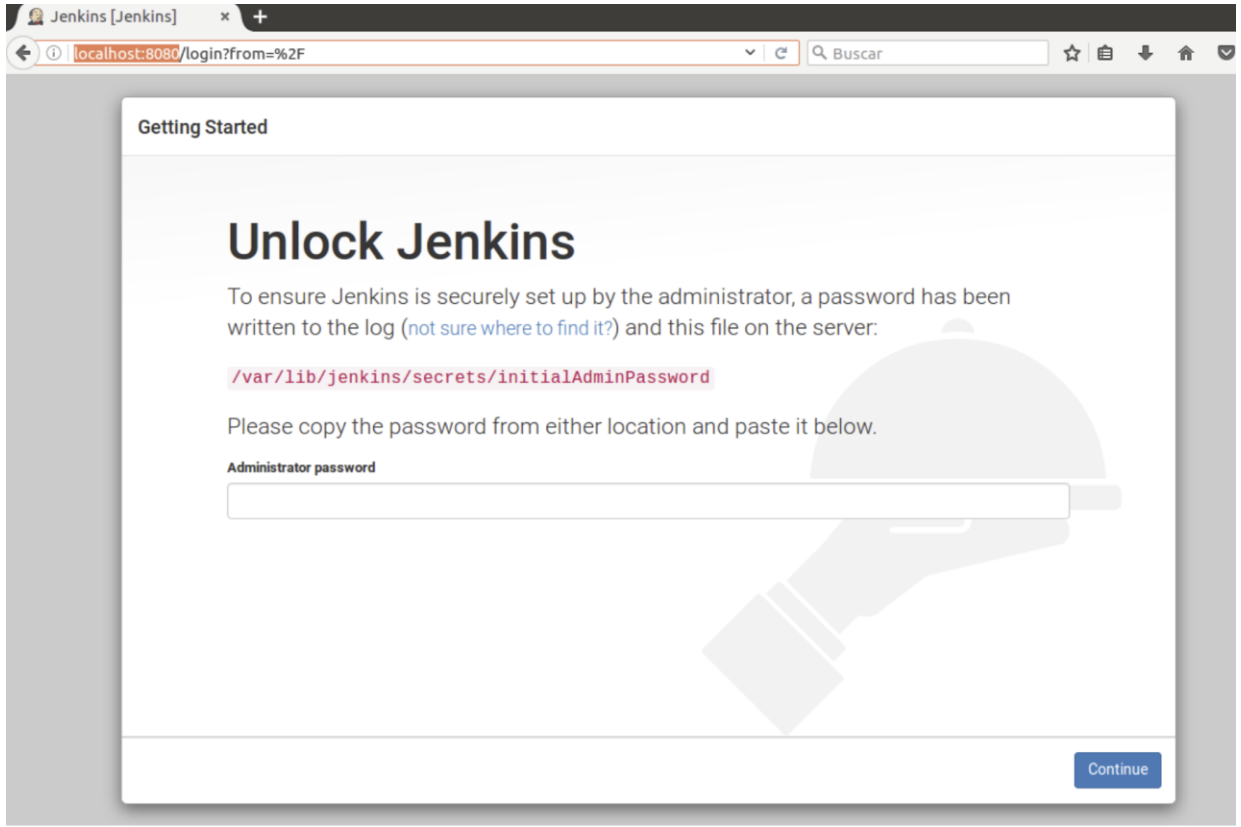


Goal 5: Setup Jenkins server

- Install Jenkins https://jenkins.io/download/
  - For Ubuntu, follow the steps here.
    - https://pkg.jenkins.io/debian-stable/
- After finished, Jenkins service will always start at boot time and will be available at http://localhost:8080

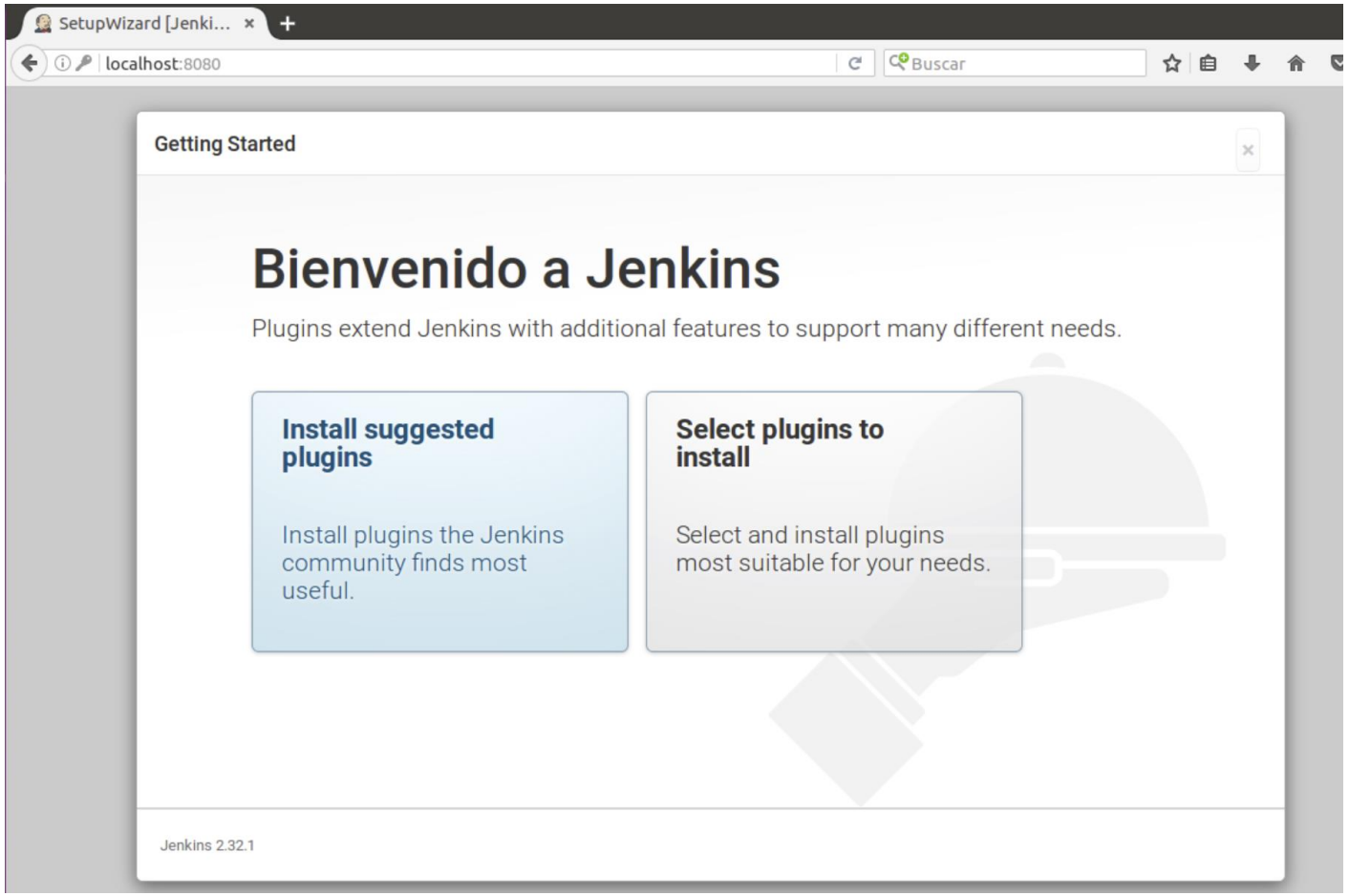
Goal 5: Setup Jenkins server

- You'll see this page for the first time
- `$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
- Paste the code shown



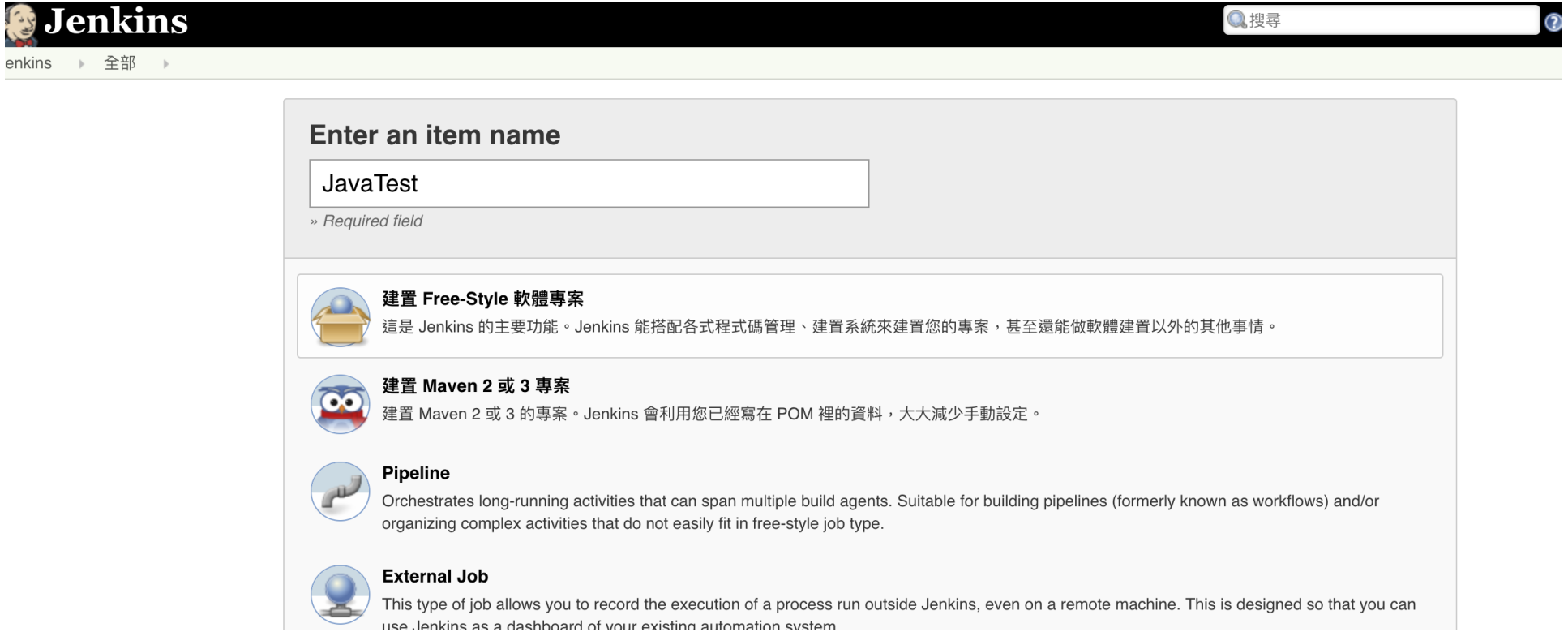
Goal 5: Setup Jenkins server

- Install suggested plugins



Goal 5: Setup Jenkins server

- Create a Jenkins job
  - Select Free-style project

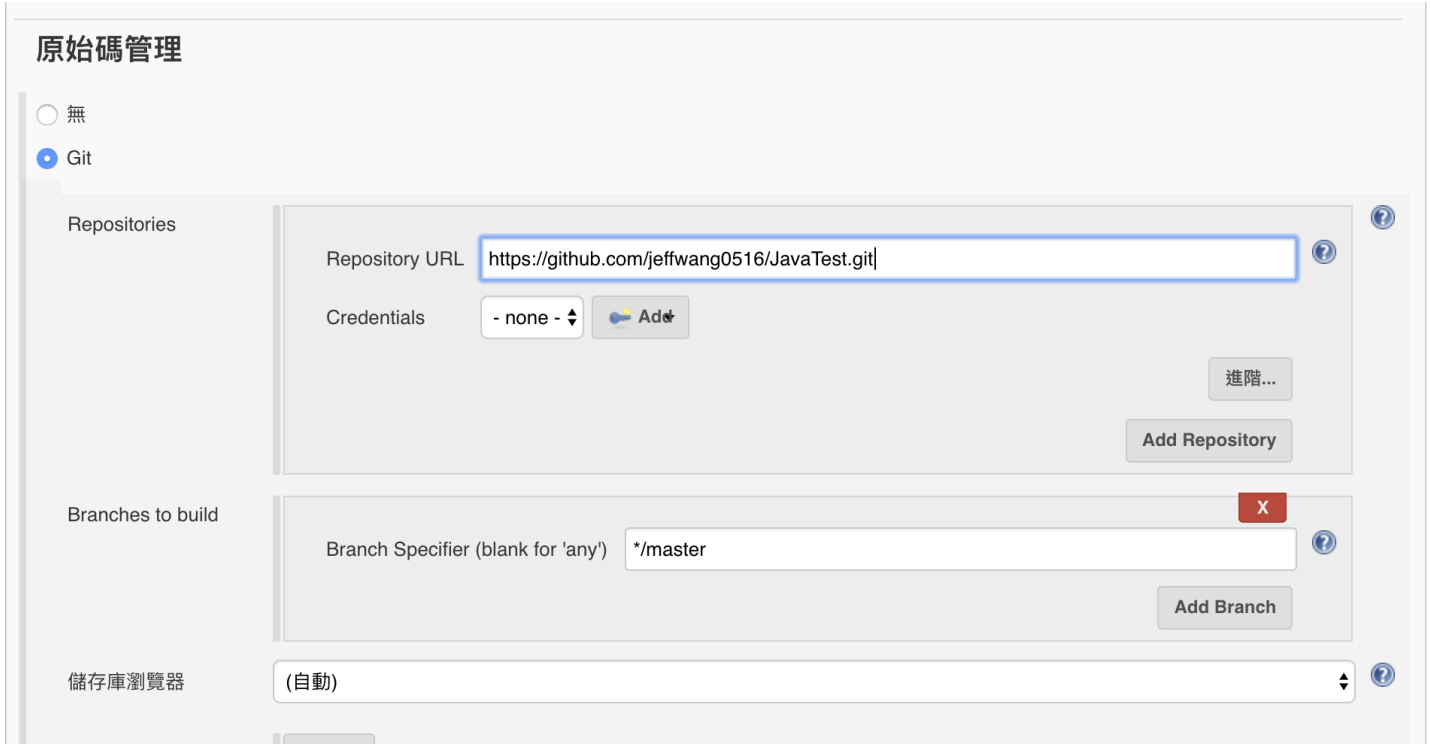


Goal 5: Setup Jenkins server

- Set project url (without .git)

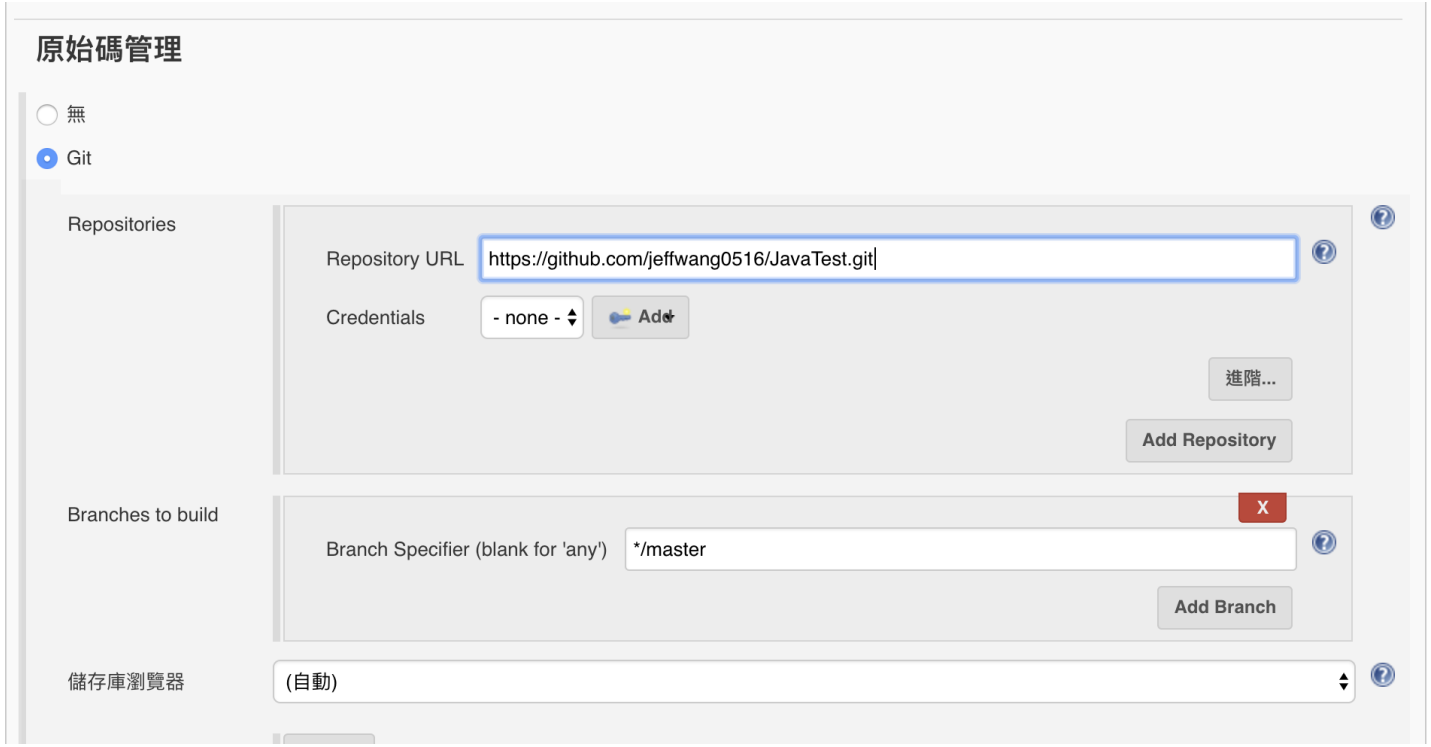
Goal 5: Setup Jenkins server

- Set git repository (with .git)



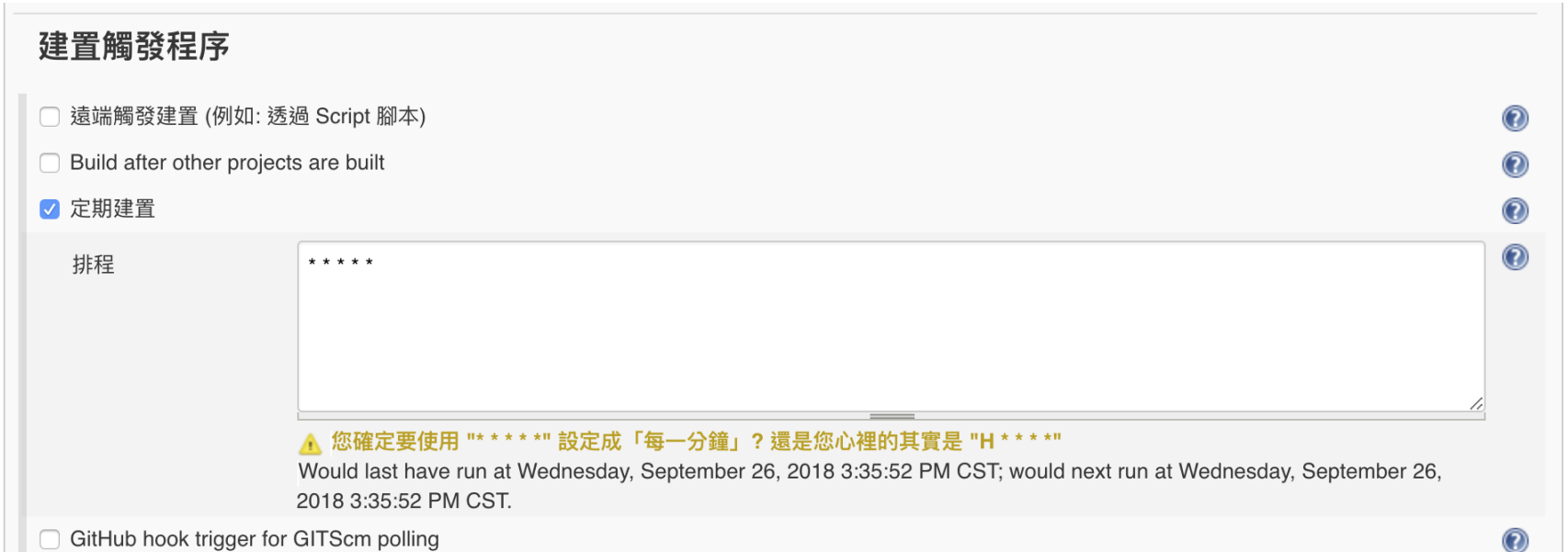
Goal 5: Setup Jenkins server

- Set git repository (with .git)



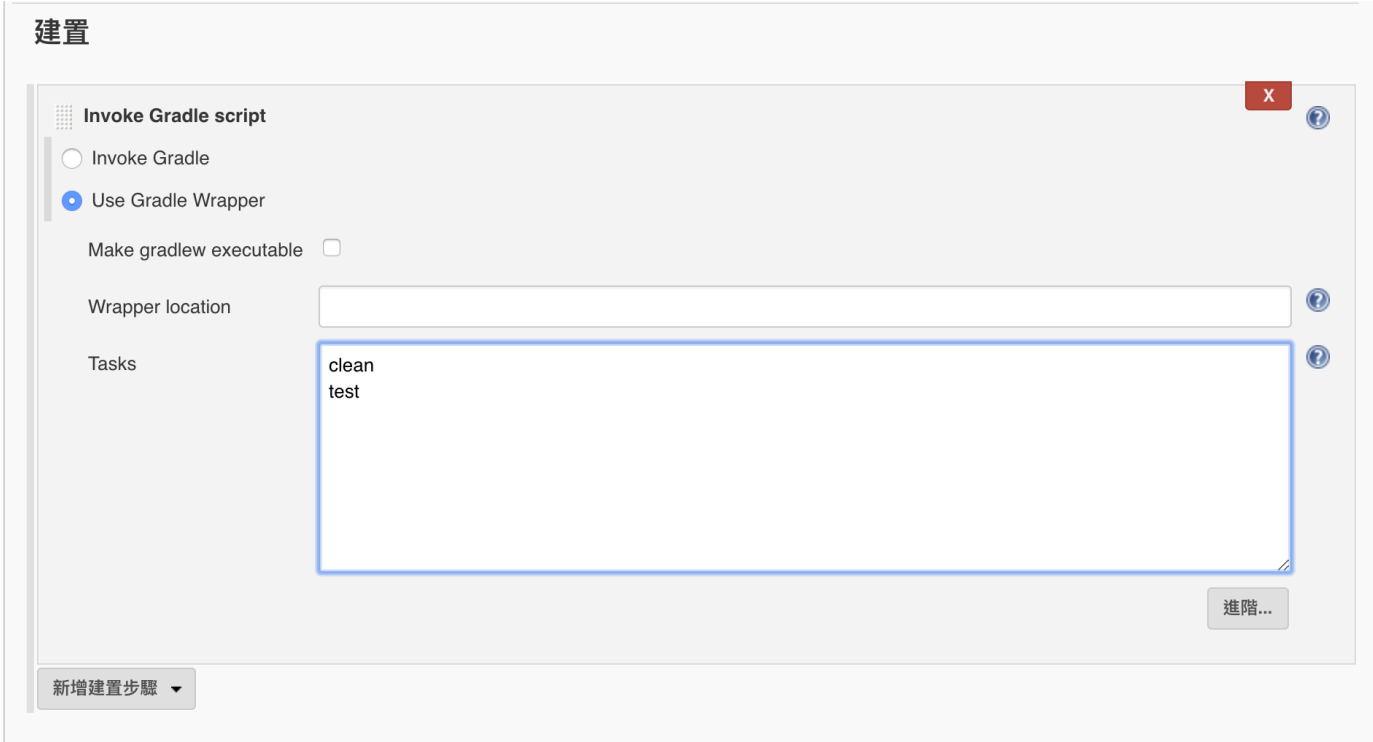
Goal 5: Setup Jenkins server

- Go to “Build Triggers” section
  - Check Poll Scm and enter five stars with spaces (* * * * *)
  - It means “every minute” for cron time process. So our jenkins server will check github every minute, if there is any change in the repo it will trigger the job and tests will start. You can set trigger time however you want using cron time.



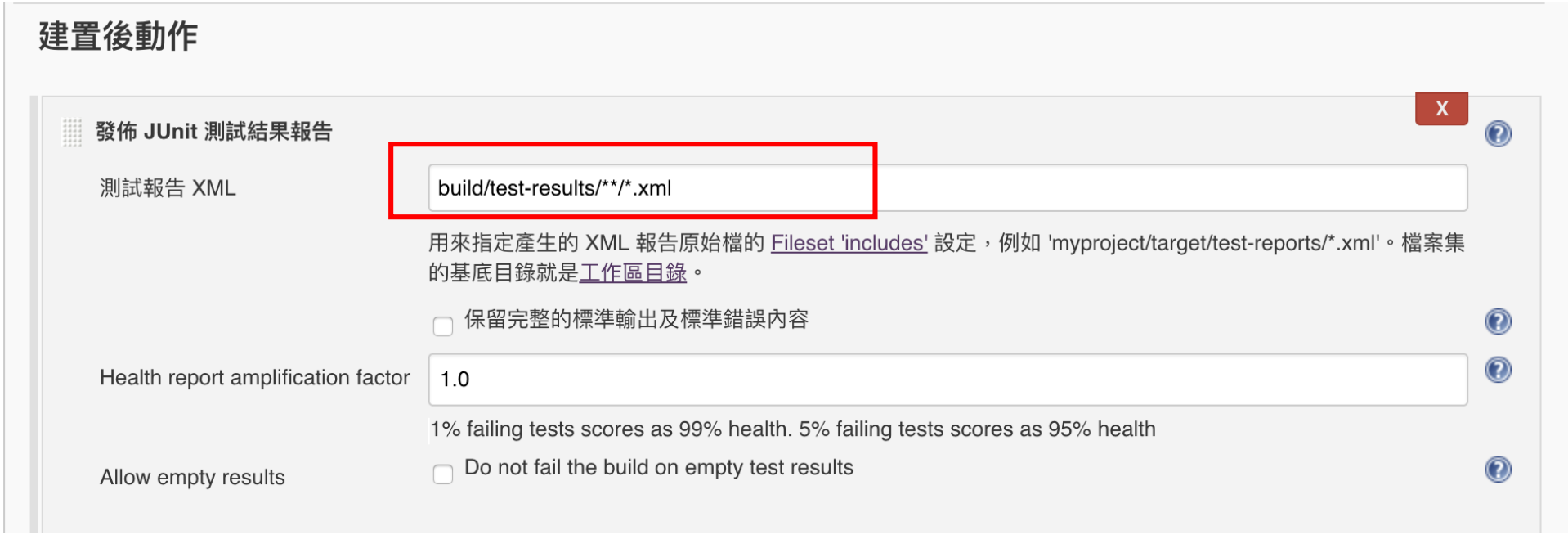
Goal 5: Setup Jenkins server

- Go to “Build Options” section
  - >> Add build step >> Invoke Gradle script
  - Select “Use Gradle Wrapper”
  - Input the script into “Tasks”



Goal 5: Setup Jenkins server

- Go to “Post-Build Actions” section
  - Add “Publish JUnit test result report ”



Goal 6:

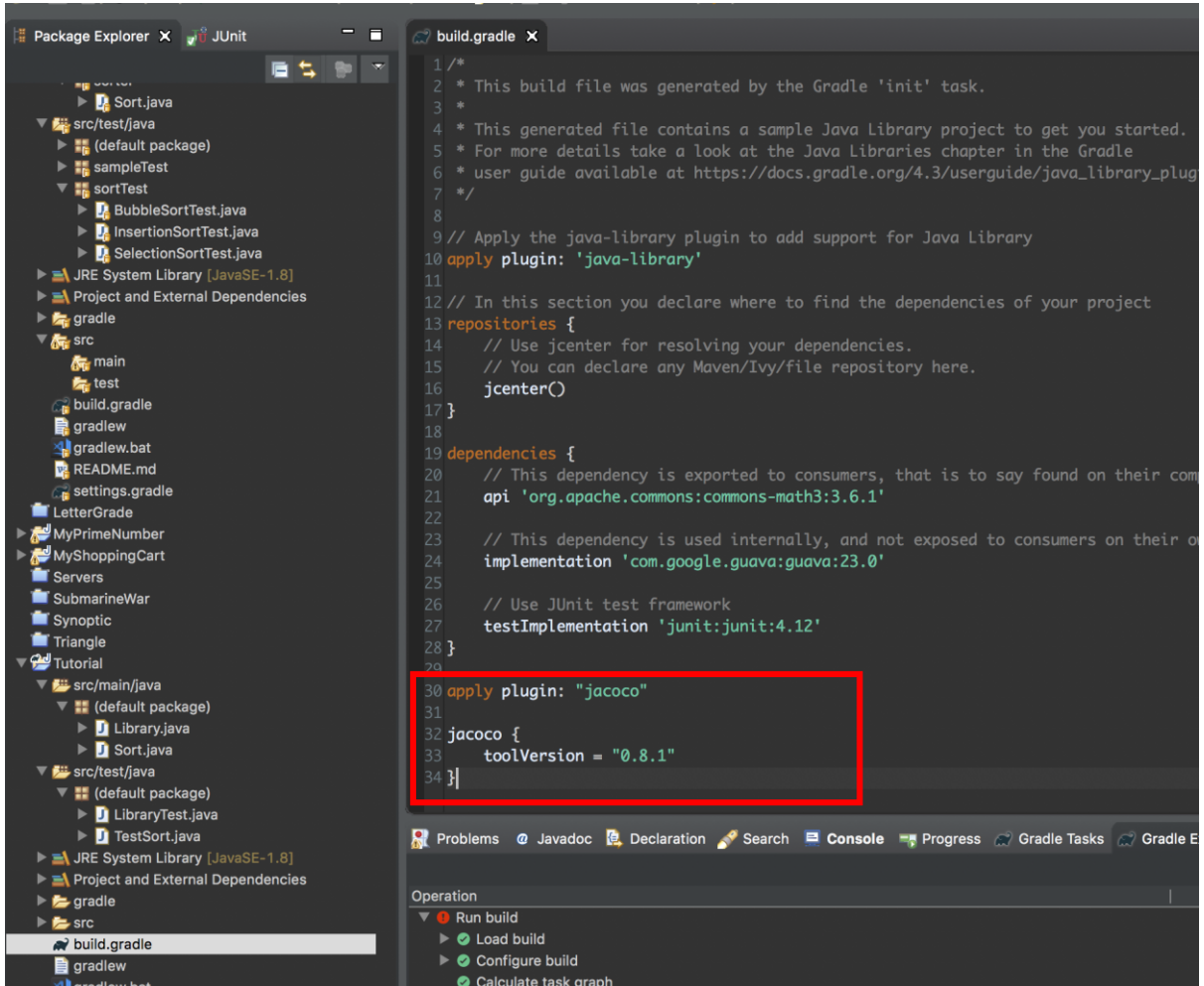
Automatic fetch and build your java project from Github on Jenkins

- Apply the settings and your done!!!
- See the build history and results

>> Click “#6” to see details

Goal 7: Coverage report with Jacoco plugin

- In your project, Edit build.gradle file (Add line 30~34)



Remember to push to github after you modified your project

Goal 7: Coverage report with Jacoco plugin

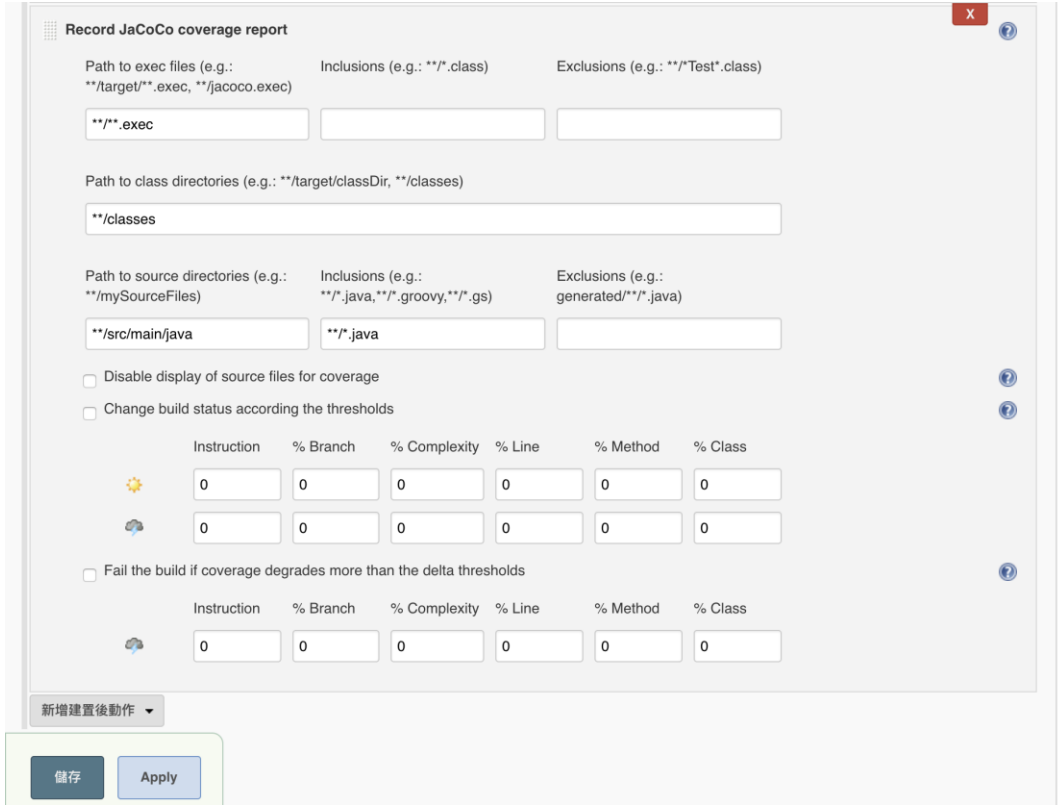
- In Jenkins, add the extension JaCoCo

Goal 7: Coverage report with Jacoco plugin

- In Jenkins, add the extension JaCoCo

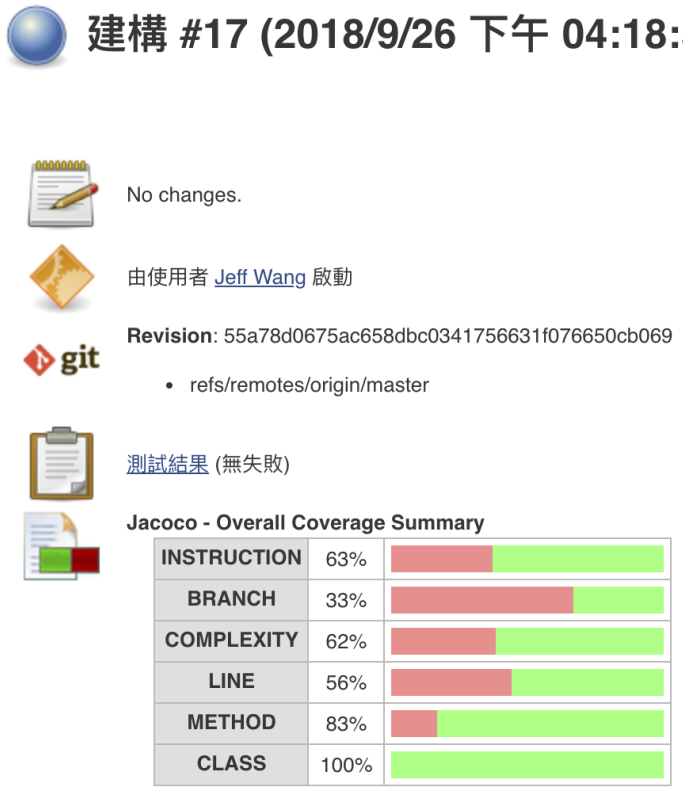
Goal 7: Coverage report with Jacoco plugin

- In Jenkins project
  - Go to “Post-Build” section
  - Add “Record JaCoCo coverage report”
  - Apply



Goal 7: Coverage report with Jacoco plugin

- In the next build
  - Extra coverage report is shown!



Final Goal :

Find an external library and design your testcases

1. Find a library (ex. the Sort we use in the demo)
2. Create a Gradle project and write testcases
3. Make sure to put it on github
4. Automatically build and test on Jenkins!

Git tutorial: https://lee-w.github.io/git-tutorial/

Score

- rawScore(you) = #bugs-found + line-coverage * log2(#total lines of the library SUT)
- *projectScore(you)* = 15*rawScore(you)/(max $_{\text{anyStudent s}}$ rawScore(s))

Submission

- Put everything in one folder named by your student id
- The folder should include:

1. Your Junit project Github repository URL
2. Screenshot of your JaCoCo coverage report
3. rawScore calculate yourself

- Compress to .zip format
- Upload to Ceiba

End