SoftLeader

Softleader Technology Corp.

www.softleader.com.tw

Jenkins

01 Jenkins 本機安裝

02 Jenkins 操作教學

03 Jenkins pipline撰寫

Jenkins

Jenkins介紹

Jenkins 是一套的可擴充套件的用於自動化部署的開源 CI/CD 工具。

Jenkins 是用 Java 編寫的，是在 MIT 許可下發布的。它有一組強大的功能，可以將軟體的構建、測試、部署、整合和釋出等相關任務自動化。

這款用於測試的自動化 CI/CD 工具可以在 macOS、Windows 和各種 UNIX 版本（例如 OpenSUSE、Ubuntu、Red Hat 等）系統上使用。

除了通過本地安裝包安裝，它還可以在任何安裝過 Java 執行時環境（Java Runtime Environment，JRE）的機器上單獨安裝或者作為一個 Docker 安裝

Jenkins

本機安裝教學

以docker 安裝 Jenkins

1. 安裝 Docker

Switch to Linux Containers

https://docs.docker.com/desktop/windows/install/

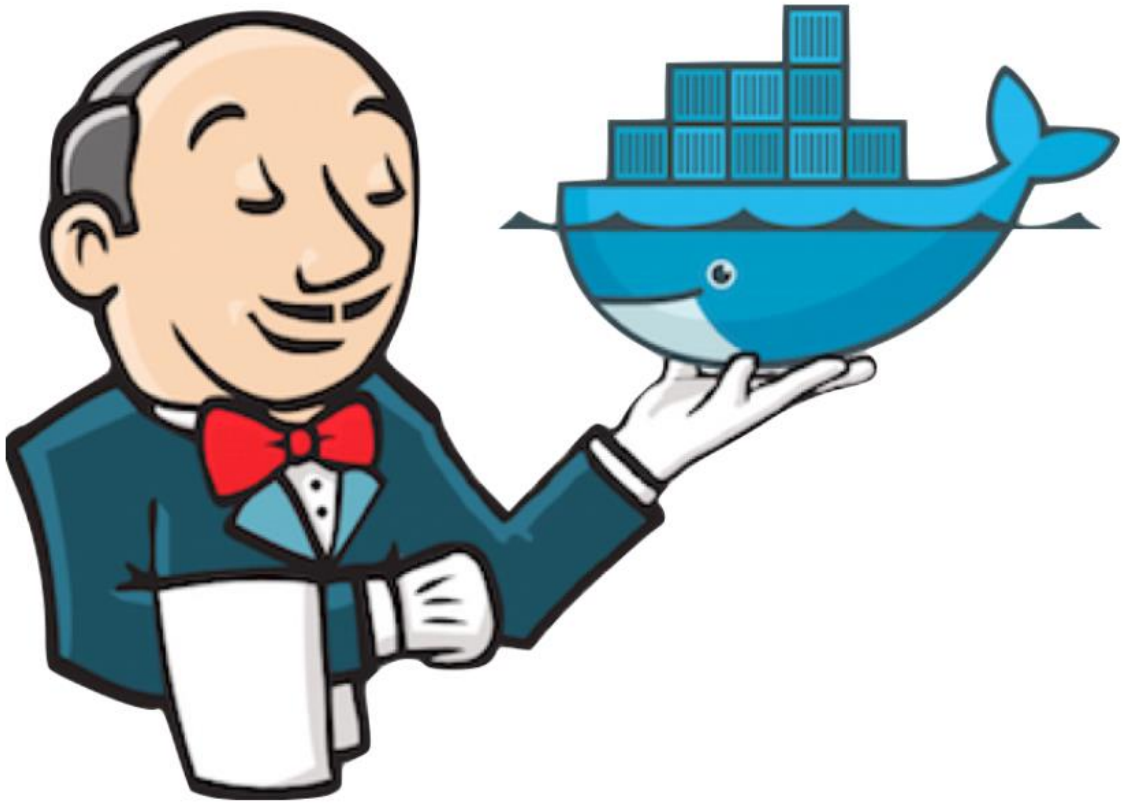
2. 選擇Jenkins image，並下載

docker pull jenkins/jenkins: 2.325-jdk8

3. 啟動Jenkins

- 以docker指令直接啟動

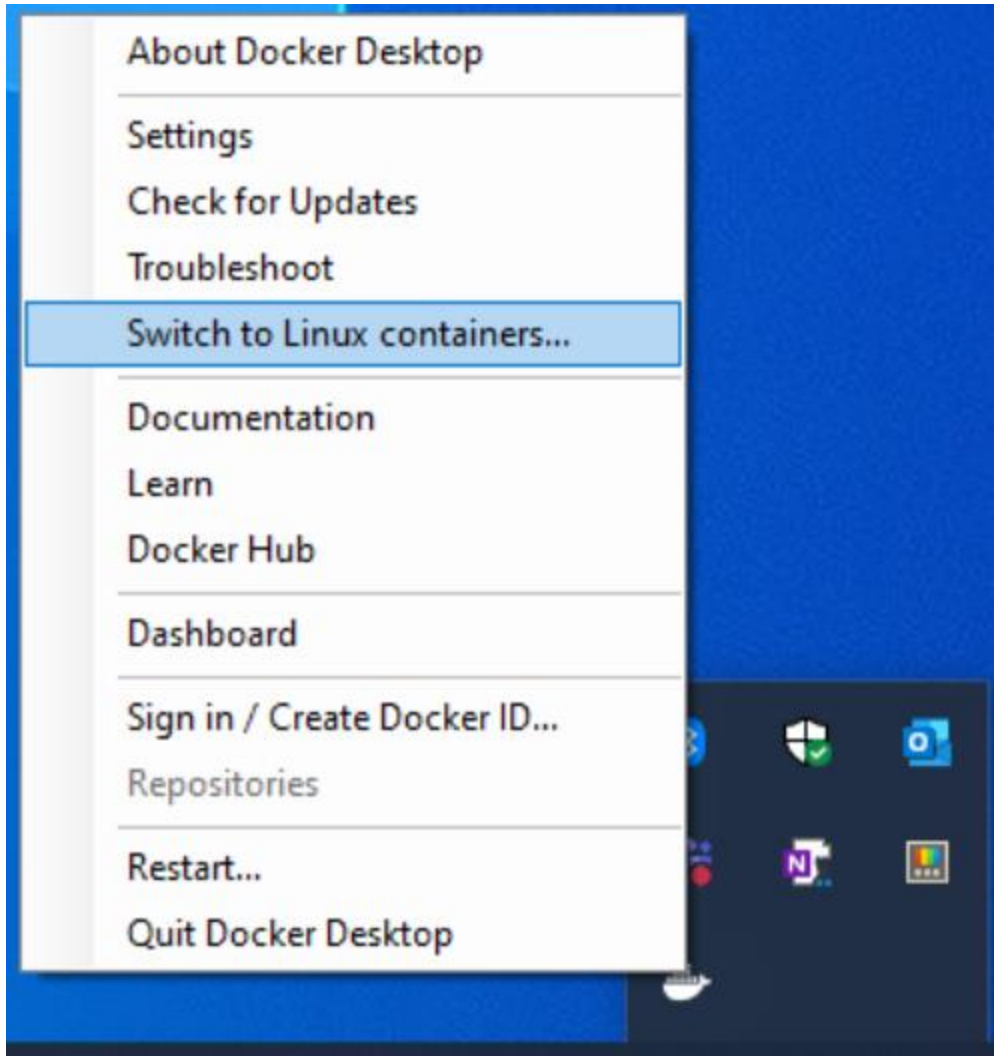
- 以docker-compose 啟動



https://www.jenkins.io/doc/book/installing/

https://hub.docker.com/_/jenkins?tab=tags

切換linux containers



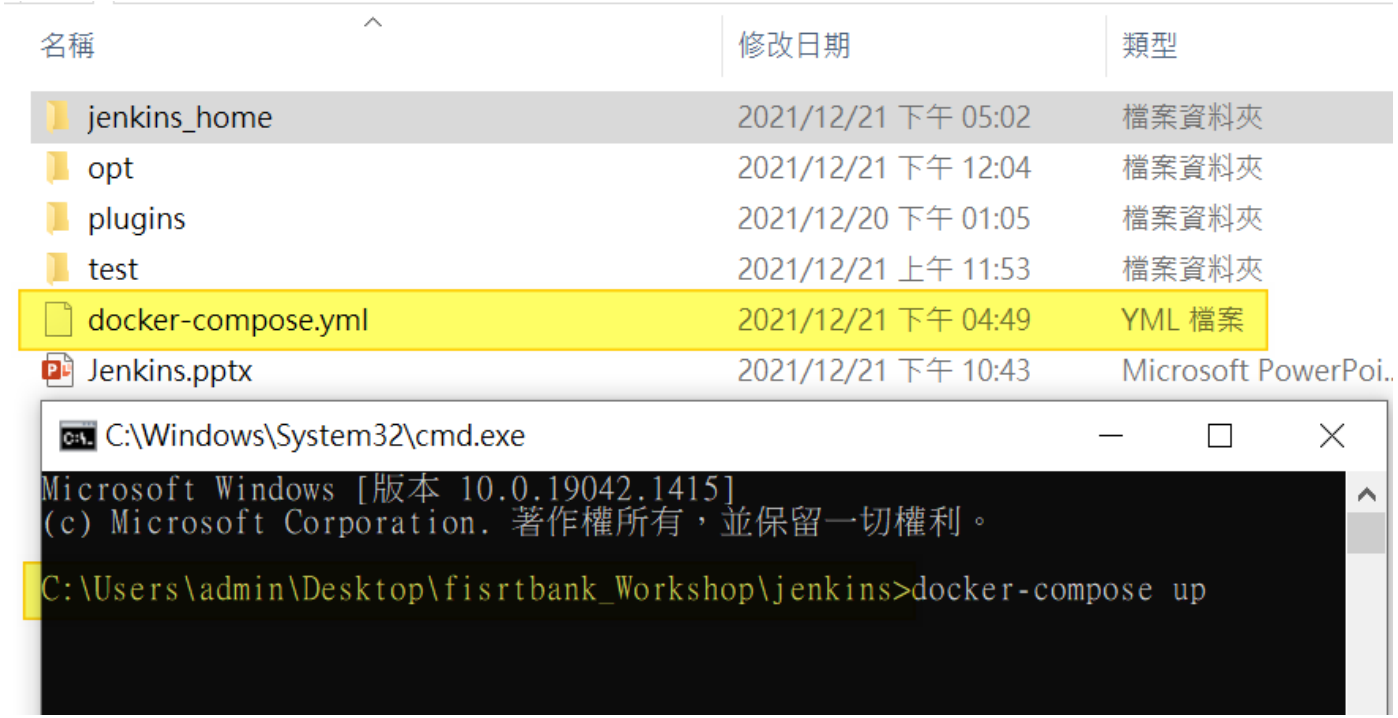
docker-compose.yml

```
version: '3.8'
services:
  jenkins:
    image:  jenkins/jenkins:2.325-jdk8
    restart: always
    ports:
      - 8080:8080
      - 50000:50000
volumes:
     - C:\Users\admin\Desktop\fisrtbank_Workshop\Jenkins\Jenkins_home:/var/jenkins_home
```

在docker-compose.yml 相同路徑中，下指令

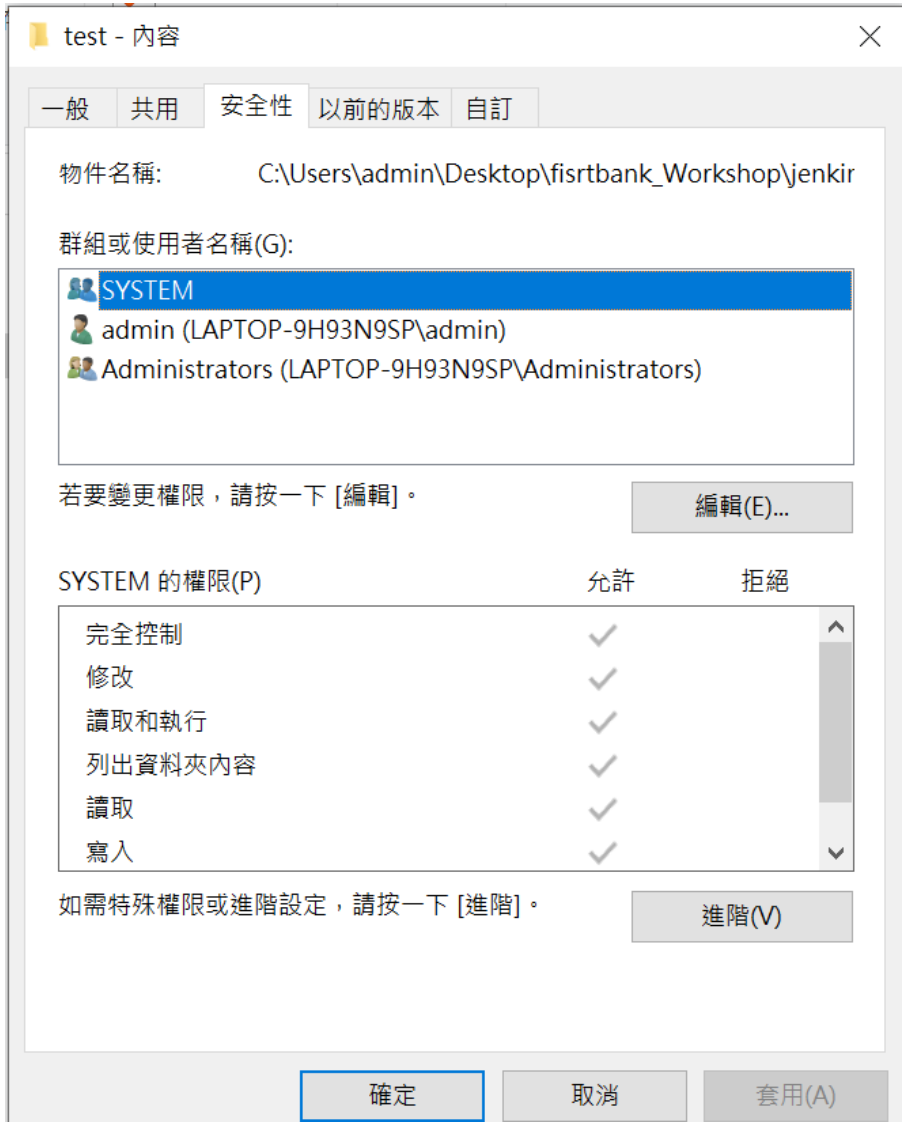
開啟container
docker-compose up –d

移除container
docker-compose down

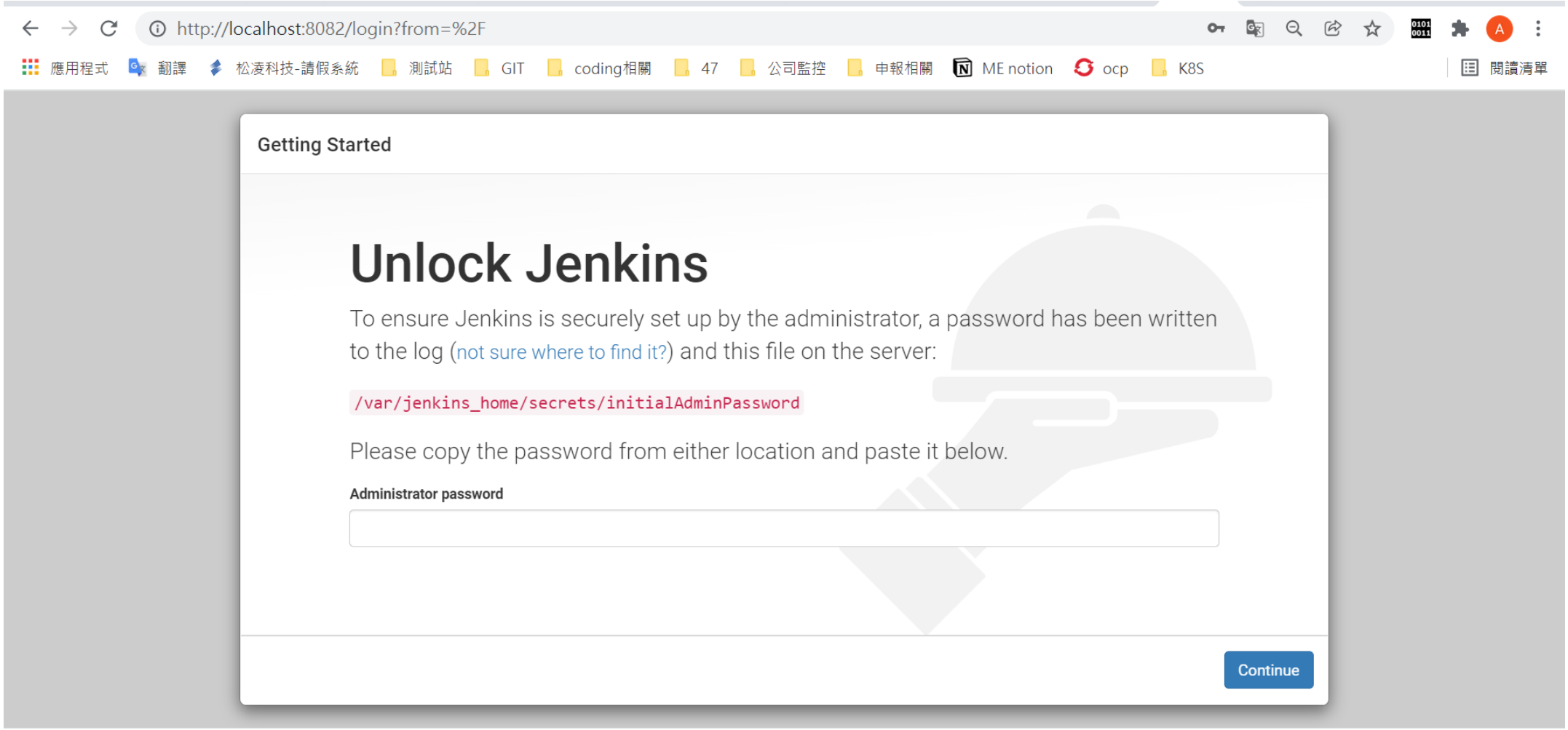


資料夾權限問題


Music Info
Music
62 KB
Modified: July 23, 2019 at 1:34 PM
Add Tags...
General:
More Info:
Name & Extension:
Comments:
Preview:
Sharing & Permissions:
You can read and write
Name
Privilege
michaelcavann...
Read & Write
staff
Read only
everyone
Read only


test - 內容
一般
共用
安全性
以前的版本
自訂
物件名稱:
C:\Users\admin\Desktop\fisrtbank_Workshop\jenkir
群組或使用者名稱(G):
SYSTEM
admin (LAPTOP-9H93N9SP\admin)
Administrators (LAPTOP-9H93N9SP\Administrators)
若要變更權限，請按一下 [編輯]。
編輯(E)...
SYSTEM 的權限(P)
允許
拒絕
完全控制
修改
讀取和執行
列出資料夾內容
讀取
寫入
如需特殊權限或進階設定，請按一下 [進階]。
進階(V)
確定
取消
套用(A)

密碼可在log或者/var/jenkins_home/secrets/initialAdminPassword找到



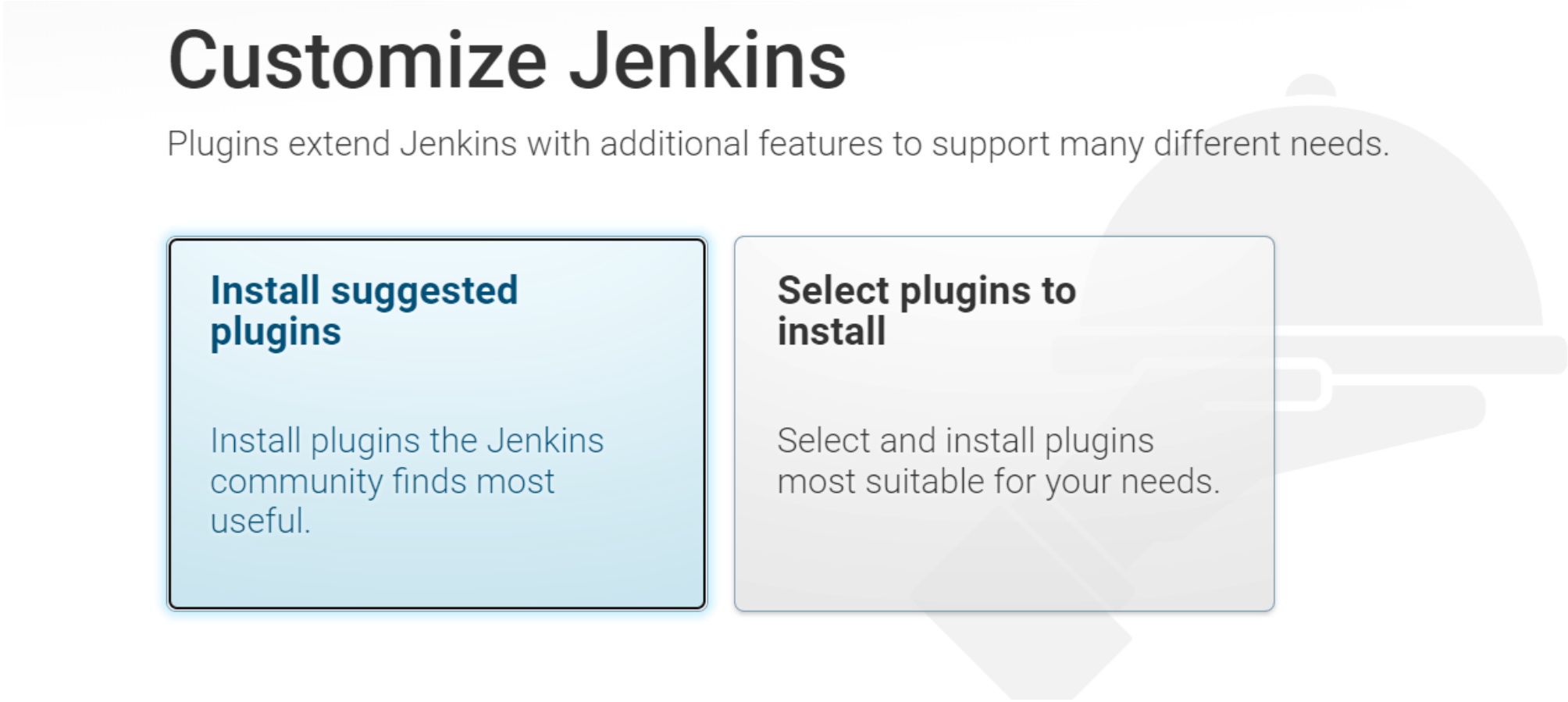
查看containe的log

docker logs –f ${container ID}

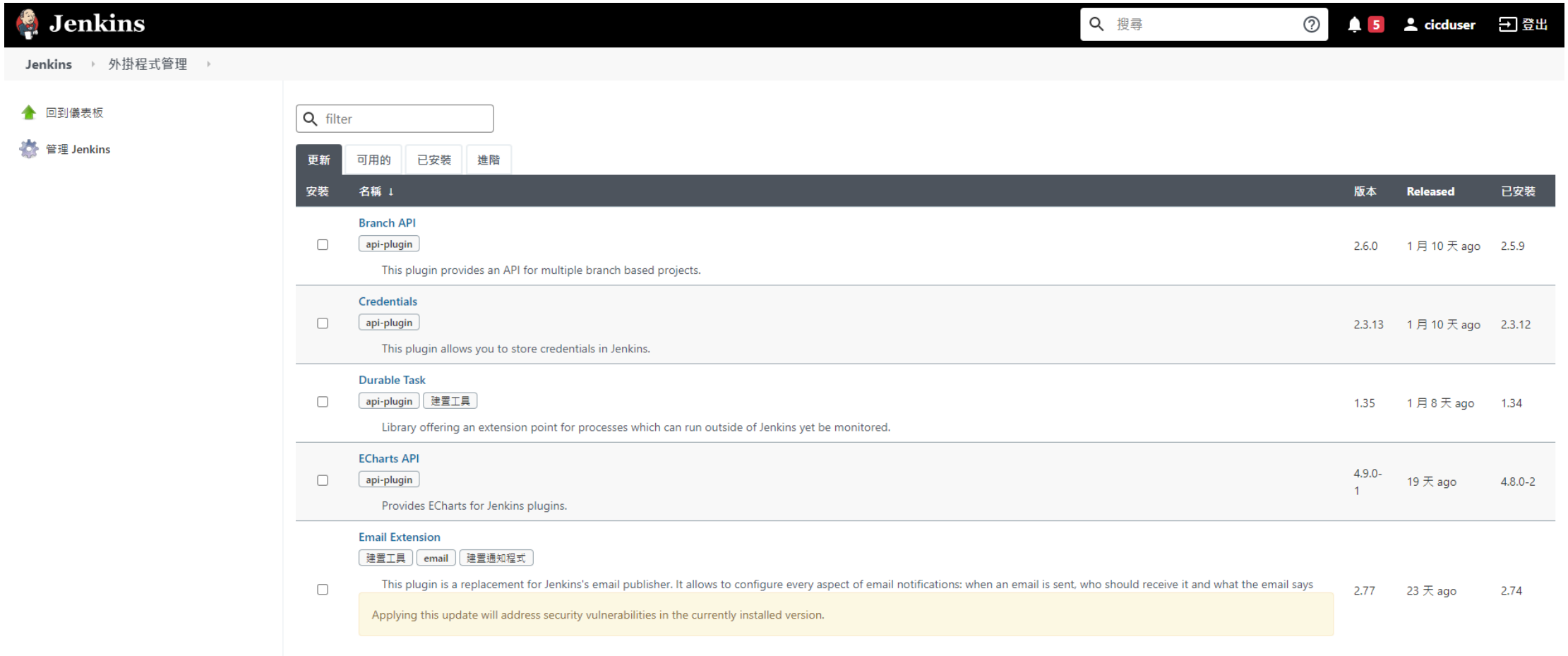
```
C:\Users\admin\Desktop\fisrtbank_Workshop\jenkins>docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS
                  NAMES
8ec225875b61   jenkins/jenkins:2.325-jdk8   "/sbin/tini -- /usr/…"   37 seconds ago   Up 34 seconds   0.0.0.0:8082->8080/tcp, :::8082->8080/tcp, 0.0.0.0:50002->50000/tc
:::50002->50000/tcp   jenkins_jenkins_1

C:\Users\admin\Desktop\fisrtbank_Workshop\jenkins>docker logs -f 8ec225875b61
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
2021-12-21 08:58:13.448+0000 [id=1]     INFO    org.eclipse.jetty.util.log.Log#initialized: Logging initialized @1641ms to org.eclipse.jetty.util.log.JavaUtilLog
2021-12-21 08:58:13.977+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2021-12-21 08:58:14.062+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2021-12-21 08:58:14.222+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4.43.v20210629; built: 2021-06-30T11:07:22.254Z; git: 526006ecfa3af7f1a2
3a288e2bef7ea9dd7e8; jvm 1.8.0_312-b07
2021-12-21 08:58:19.019+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServle
2021-12-21 08:58:19.177+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=node0
2021-12-21 08:58:19.177+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2021-12-21 08:58:19.179+0000 [id=1]     INFO    o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 660000ms
2021-12-21 08:58:21.144+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENK
 HOME")
```

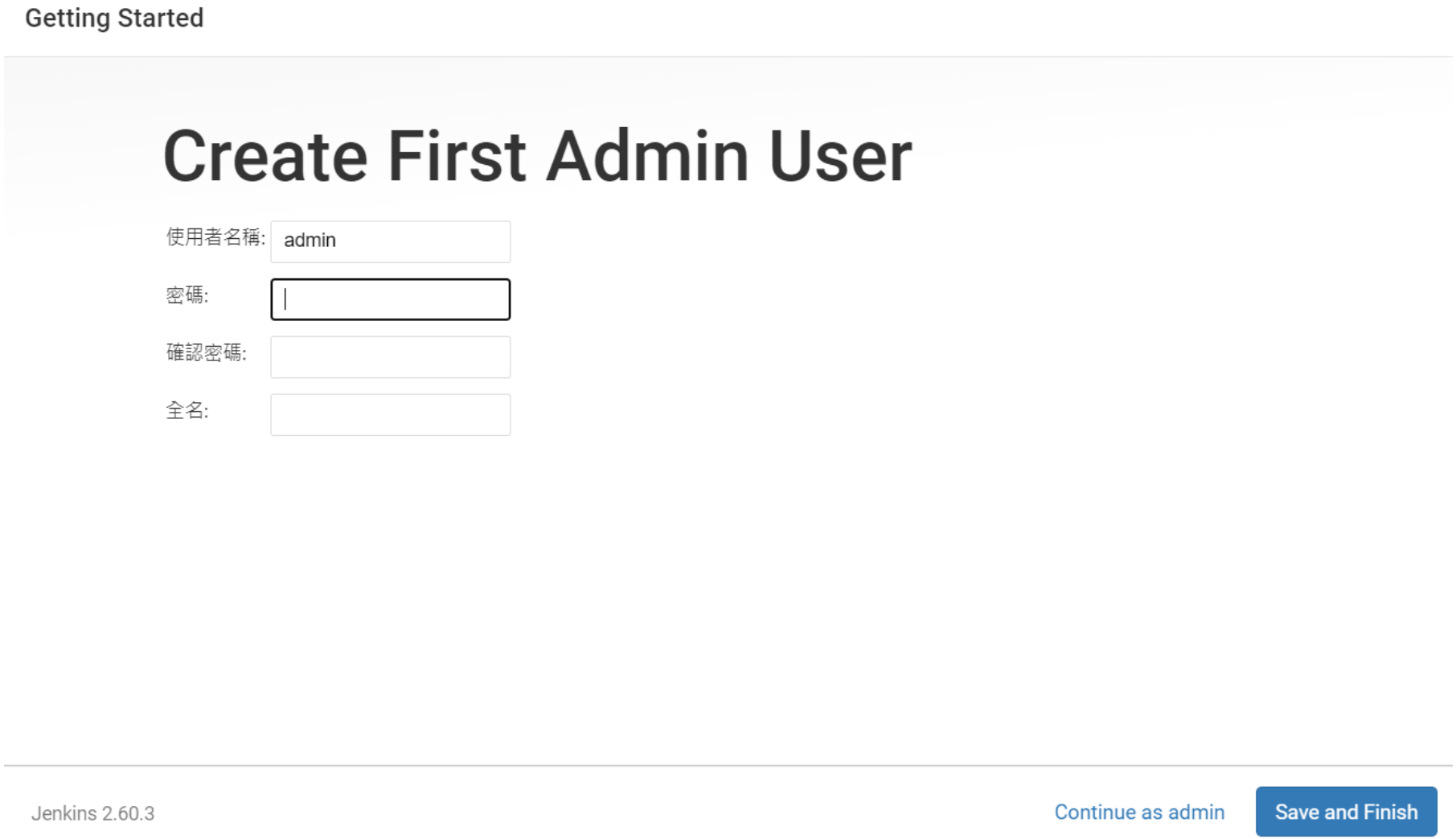
plugin 安裝


Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.

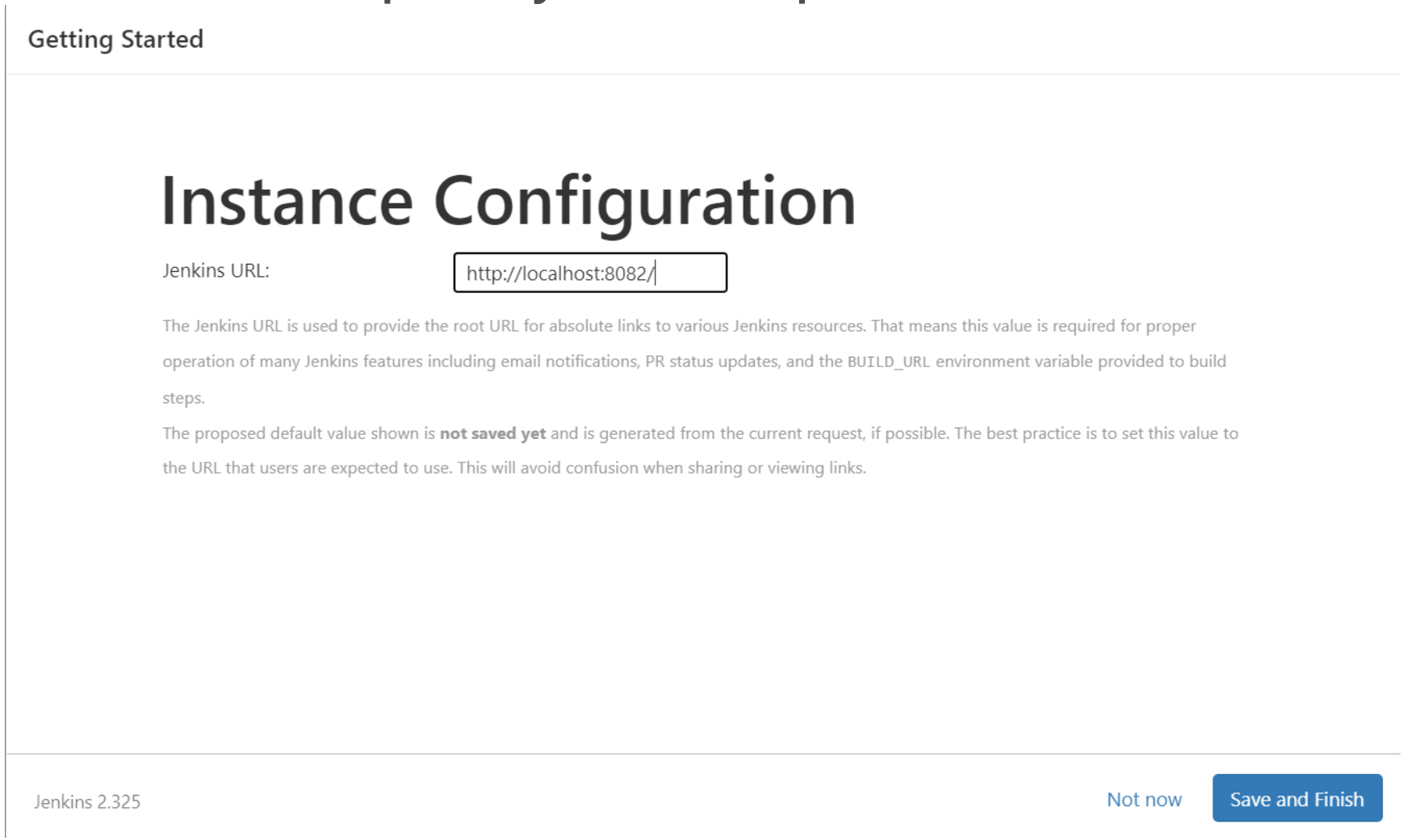
若離線安裝 可以將plugin 放在: /jenkins_home/plugins/內，完成插鍵離線安裝。



設定Admin User



設定連線jenkins的介面的URL
要跟剛剛docker-compose.yml 對外port號相同



安裝成功後，大家可以先玩玩看介面

Jenkins

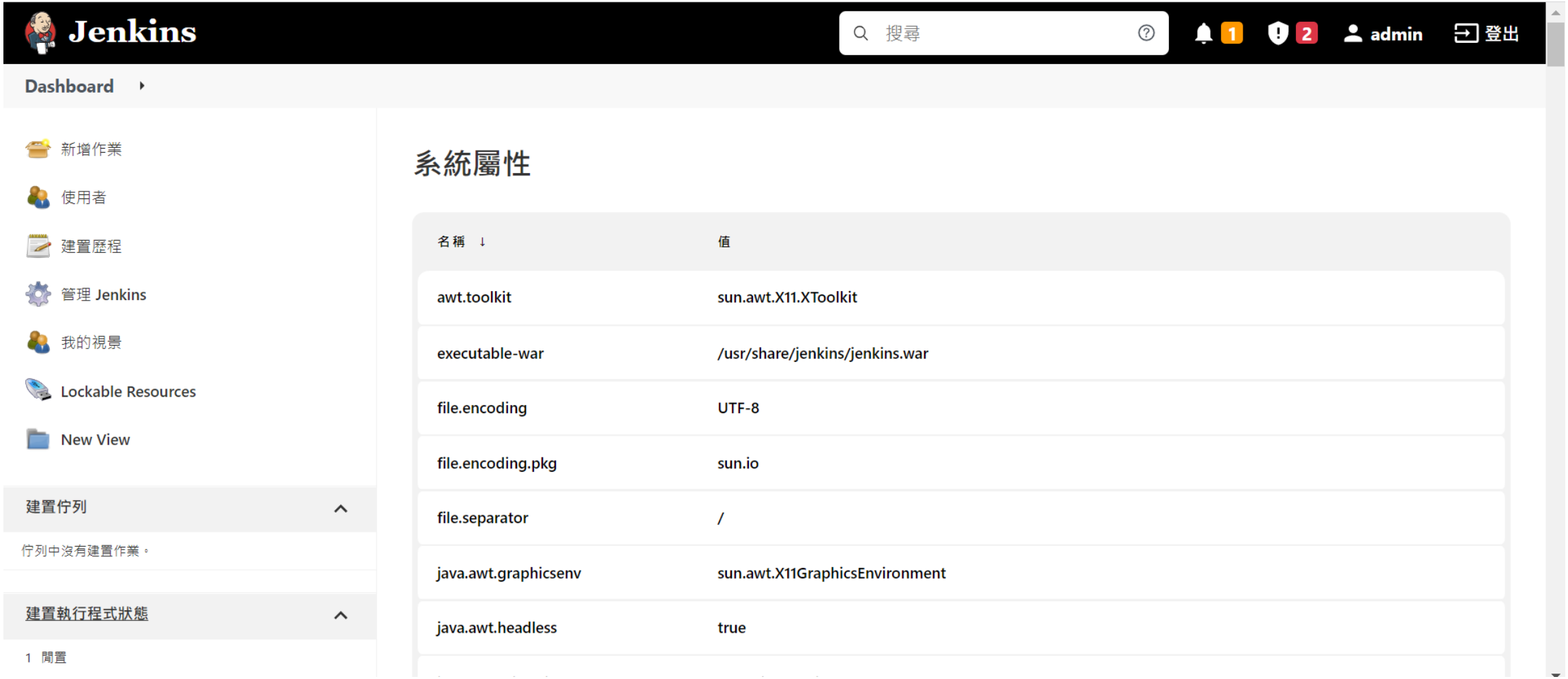
設定配置

Jenkins 設定配置

- 預覽基礎設定
- Plugin 安裝
- 設定使用者認證- Jenkins 內部使用者
- 設定專案權限與角色權限

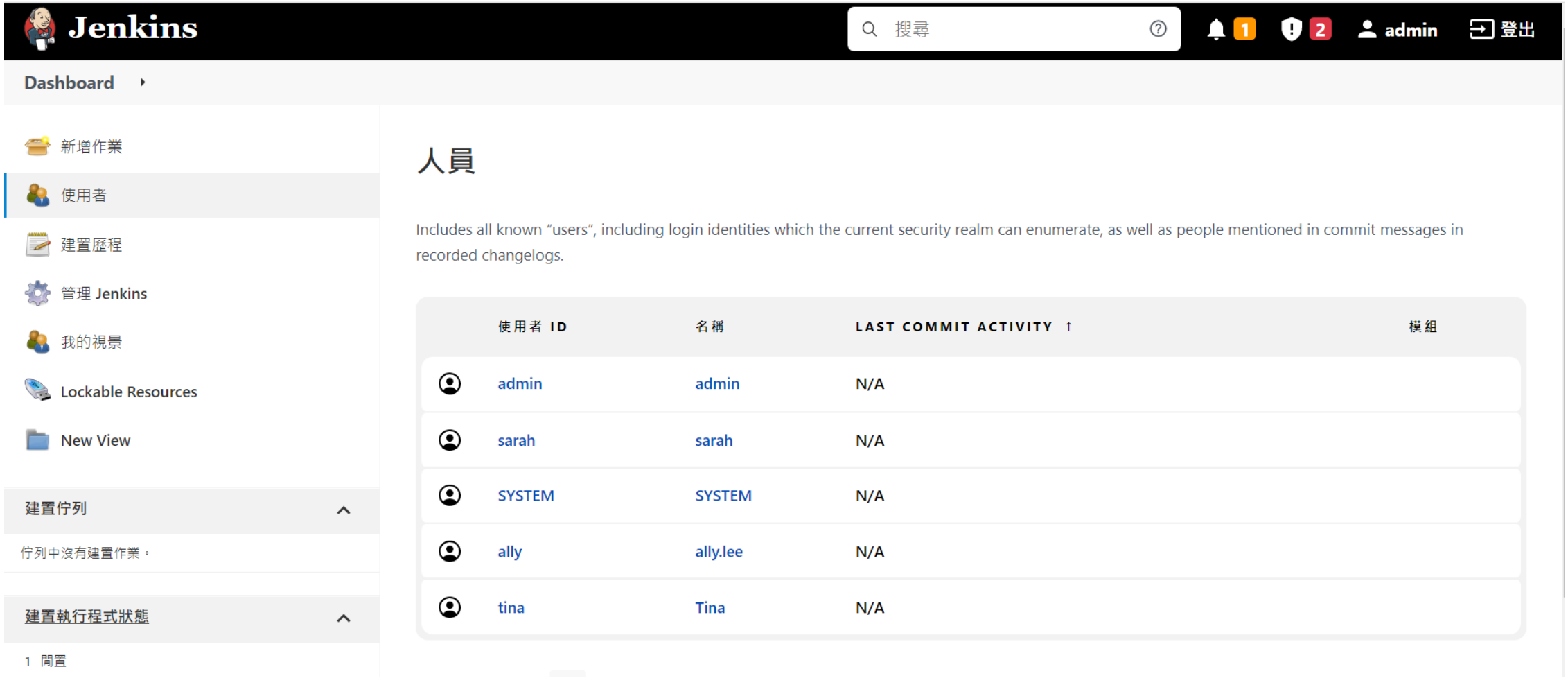
基礎設定

Jenkins 基礎瀏覽



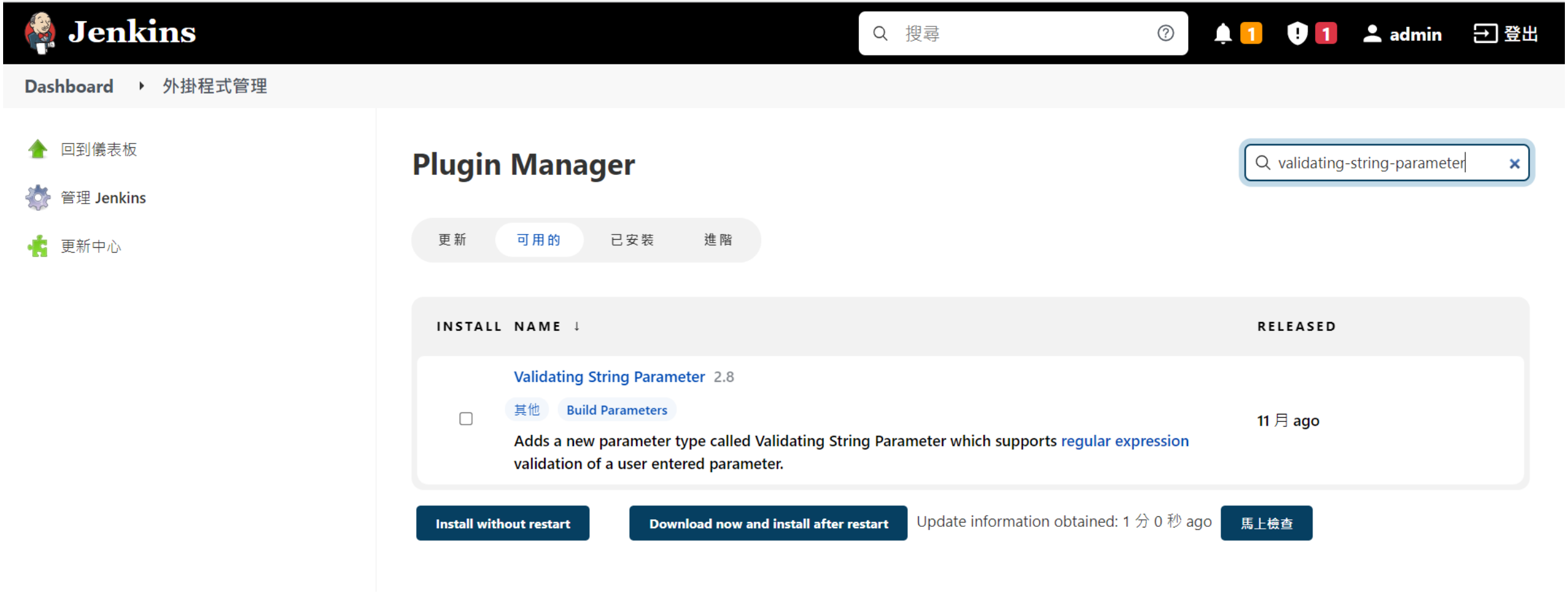
Jenkins 建置歷程

查看使用者

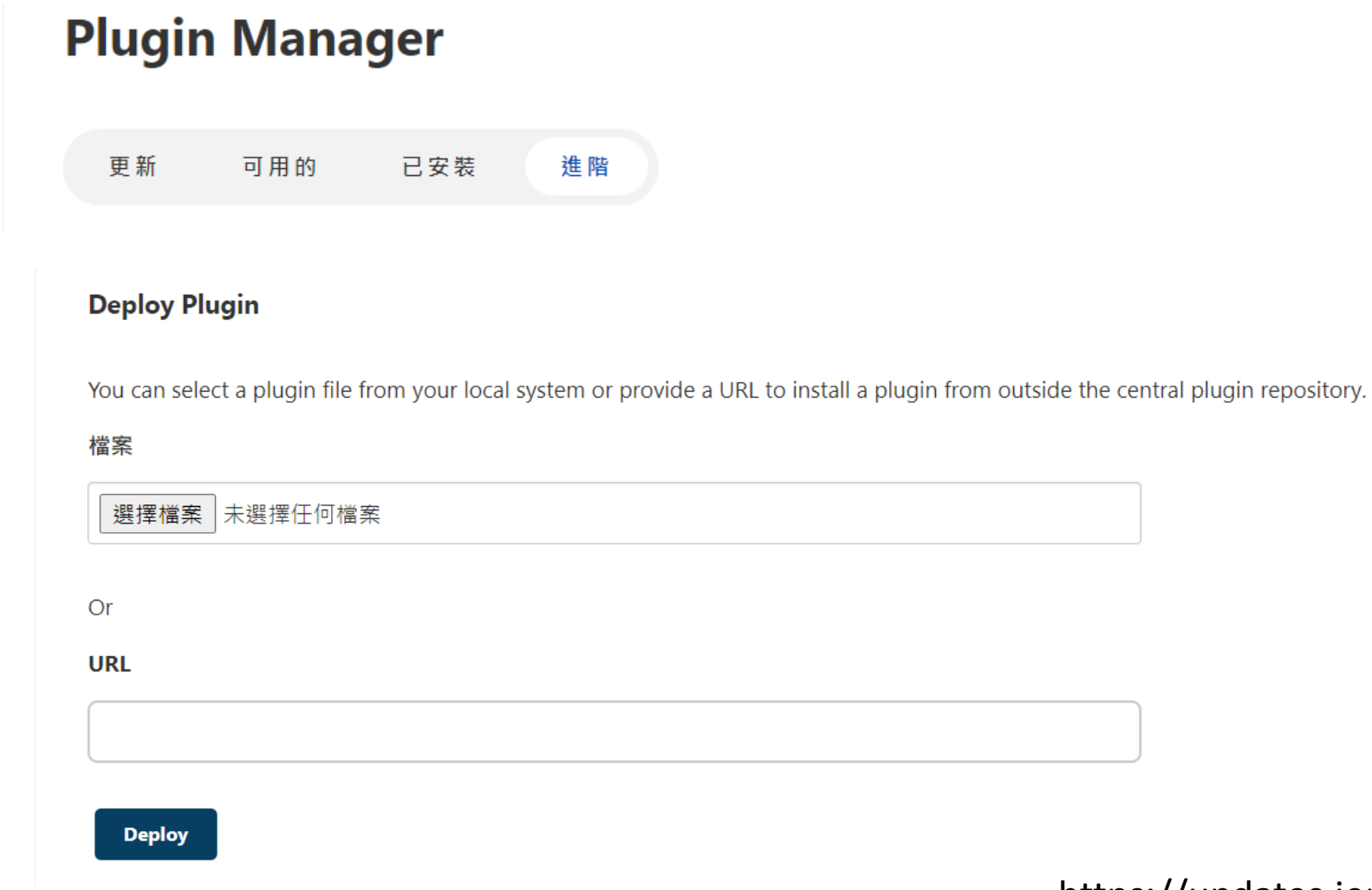

Jenkins
搜尋
1
2
admin
登出
Dashboard
新增作業
使用者
建置歷程
管理 Jenkins
我的視景
Lockable Resources
New View
建置佇列
佇列中沒有建置作業。
建置執行程式狀態
1 閒置
人員
Includes all known "users", including login identities which the current security realm can enumerate, as well as people mentioned in commit messages in recorded changelogs.
使用者 ID
名稱
LAST COMMIT ACTIVITY
模組
admin admin N/A
sarah sarah N/A
SYSTEM SYSTEM N/A
ally ally.lee N/A
tina Tina N/A

安装plugin

Jenkins 安裝plugin



若離線安裝 plugin :



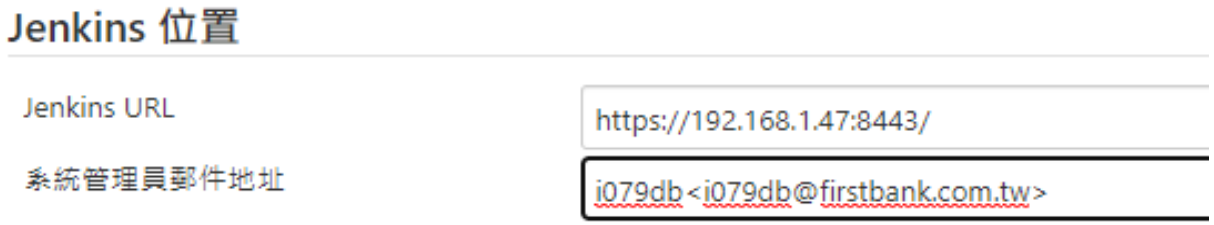
https://updates.jenkins-ci.org/download/plugins/

離線安裝，需要注意Jenkins版本與plugin版本的相容性，或與其他plugin的依賴性

SMTP設定

SMTP設定: 登入Jenkins --> 系統設定 --> jenkins位置、電子郵件通知

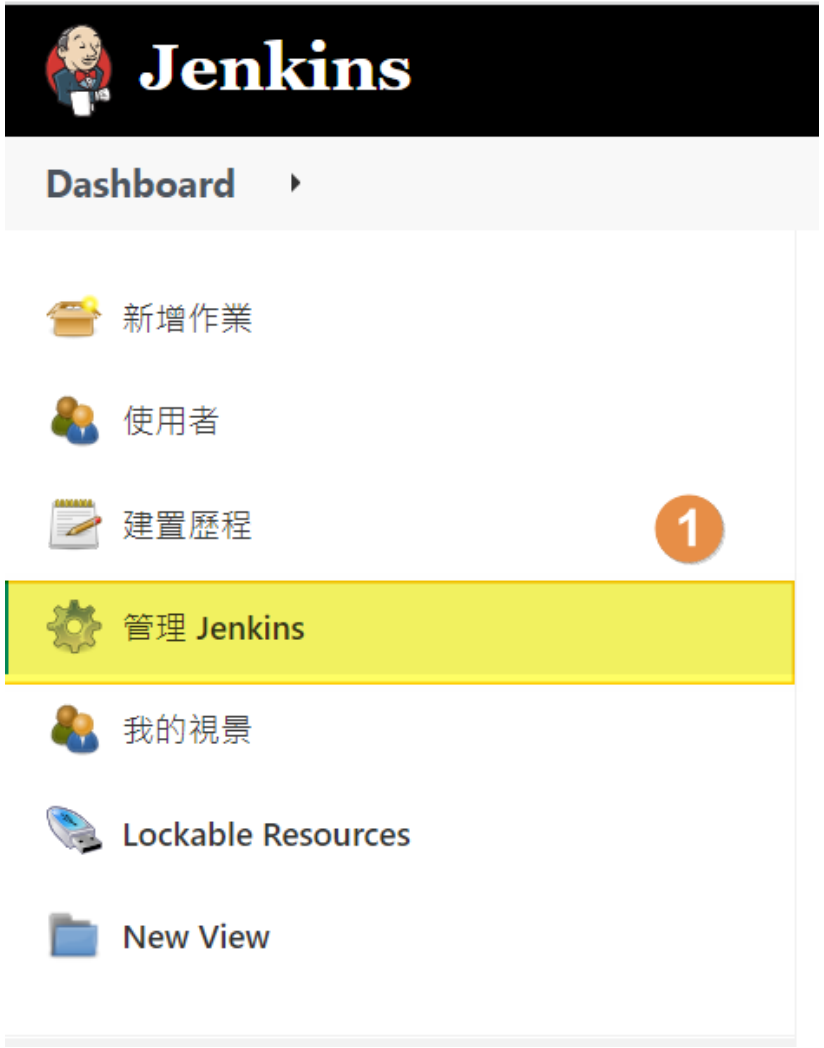
jenkins位置、電子郵件通知

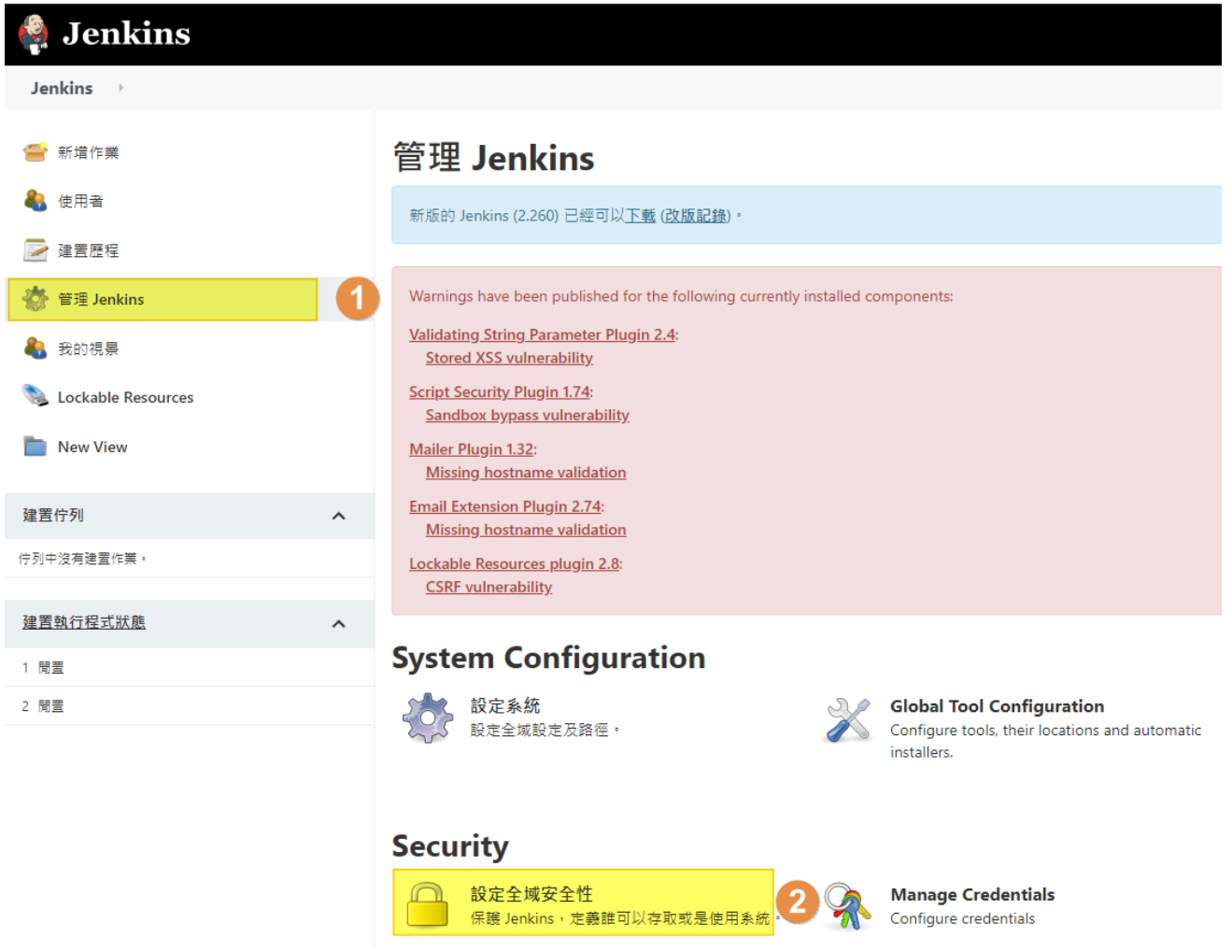


1.SMTP伺服器
2.允許使用SMTP認證
3.使用者名稱:
4.密碼:
5.是否允許使用SSL
6.SMTP連接埠:
7.預設內容類型：請選Html
8.預設主旨：請依照自身需求設定，這裡我保留預設
預設內容：這邊請設定 `${JELLY_SCRIPT,template="email-template.jelly"}`,
也就是上一個步驟存放的email template檔名，要用這個檔來產生電子郵件內容

設定使用者認證

權限設定: 登入Jenkins --> 管理jenkins --> 設定全域安全性





設定全域安全性

Authentication

☐ Disable remember me

安全性領域

◉ Jenkins 內建使用者資料庫 ?

☑ 允許使用者註冊 ?

⚠ **With signup enabled, anyone on your network can become an authenticated user. It is recommended in this case to minimize the permissions granted to any authenticated user.**

○ LDAP ?

○ Unix user/group database ?

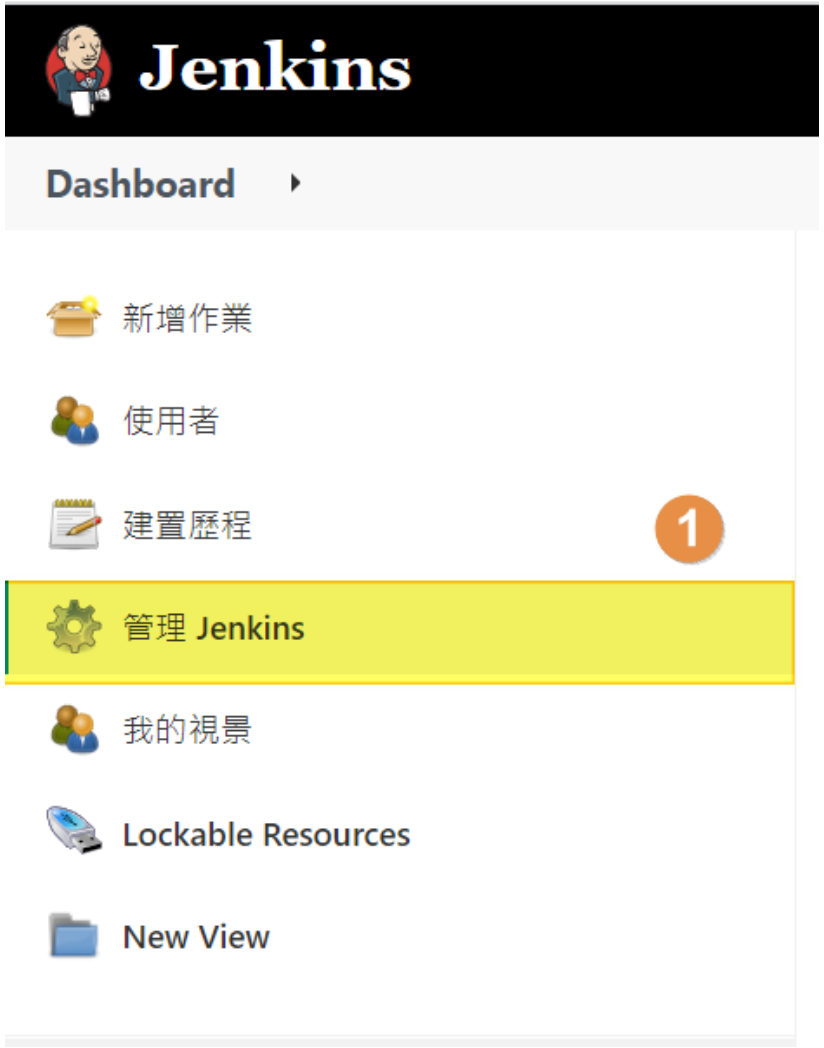
○ 委派給 Servlet Container ?

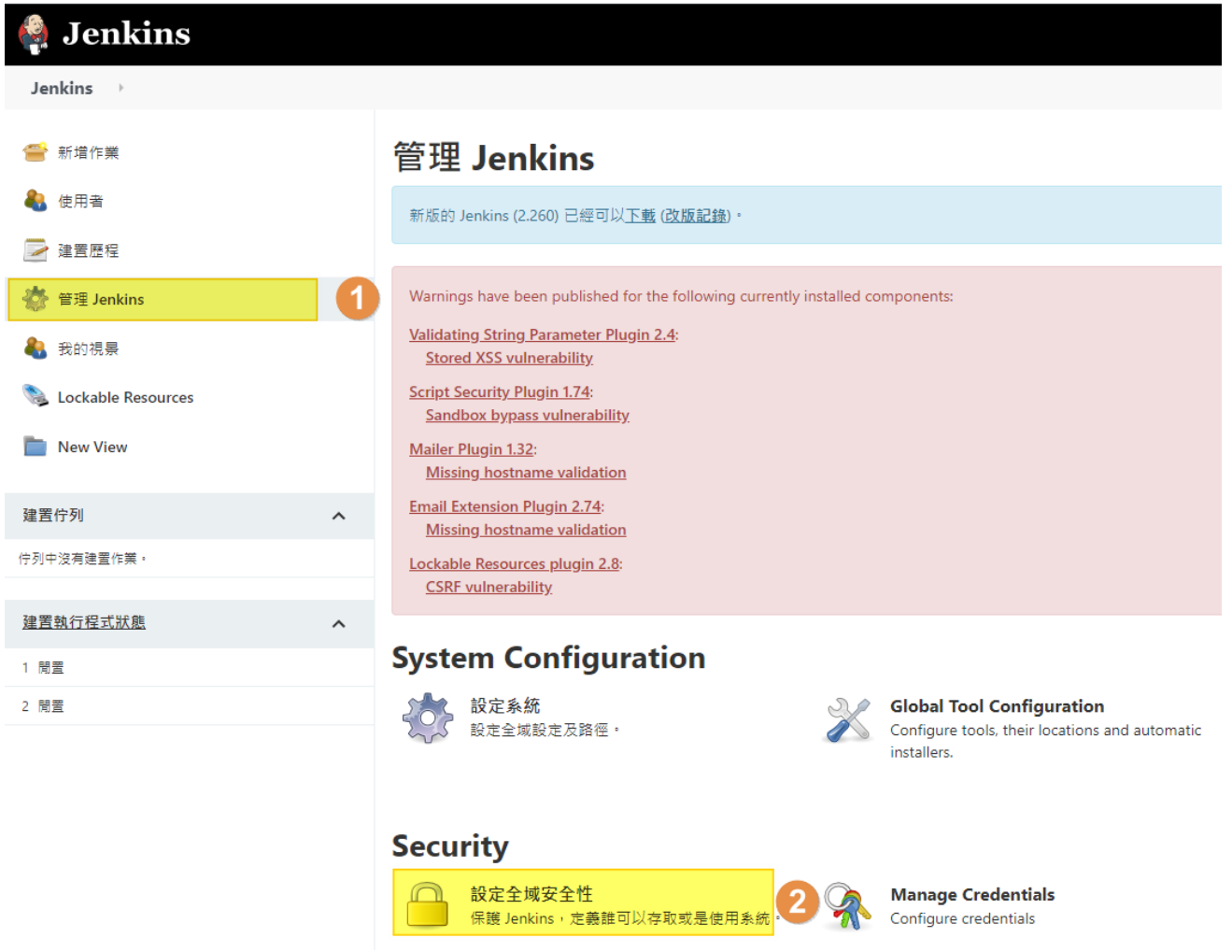
○ None

設定完後可以以其他使用者登入看看

權限設定

權限設定: 登入Jenkins --> 管理jenkins --> 設定全域安全性





SoftLeader
Softleader Technology Corp.

設定全域安全性 -->授權

授權

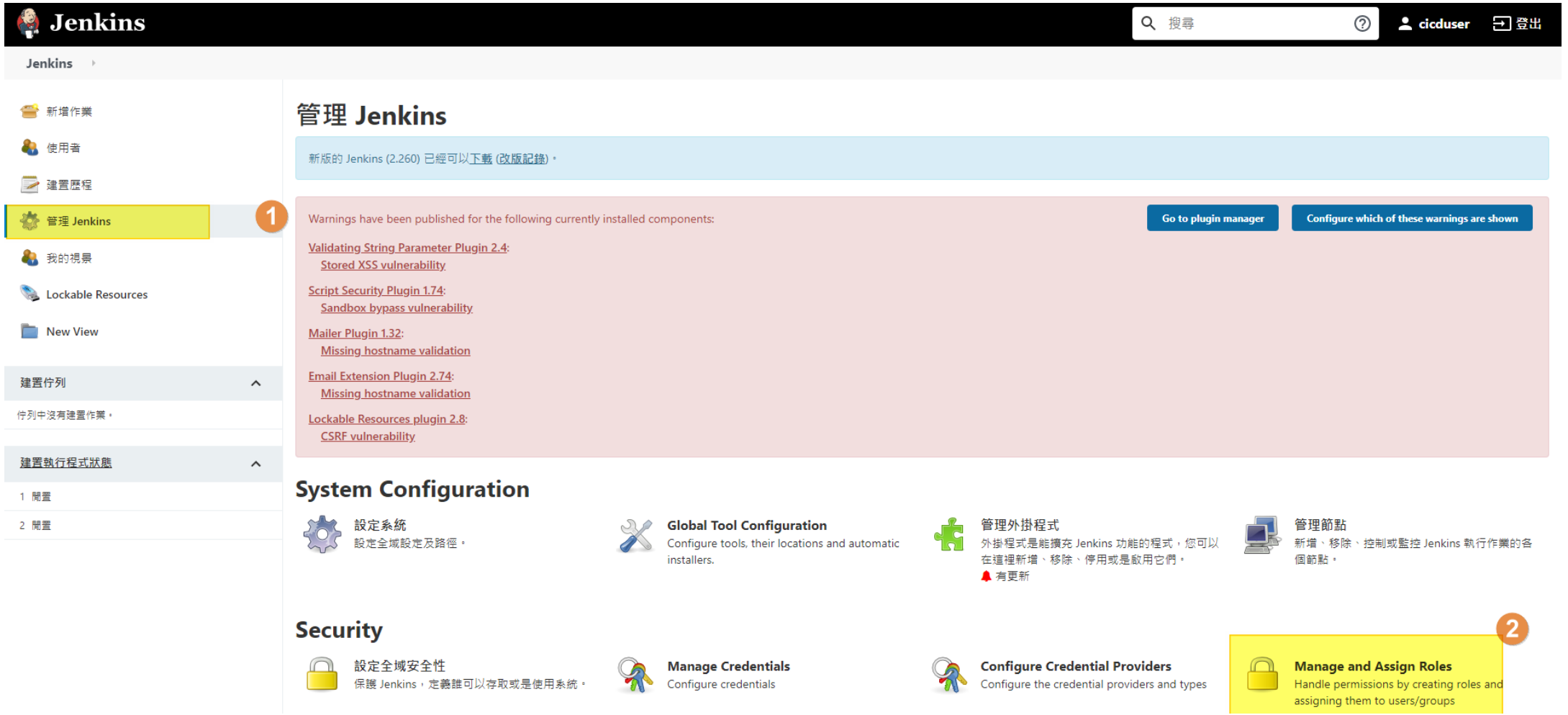
- (●) Role-Based Strategy ?
- ( ) 大家都可以做任何事 ?
- ( ) 專案型矩陣授權策略 ?
- ( ) 登入成功的使用者可以做任何事 ?
- ( ) 矩陣型安全性 ?

在這種配置下，您可以透過一張大表格，設定每個人可以做的每件事。
每一欄都表示一項權限。將滑鼠游標移到權限名稱上，可以看到權限代表的意義說明。

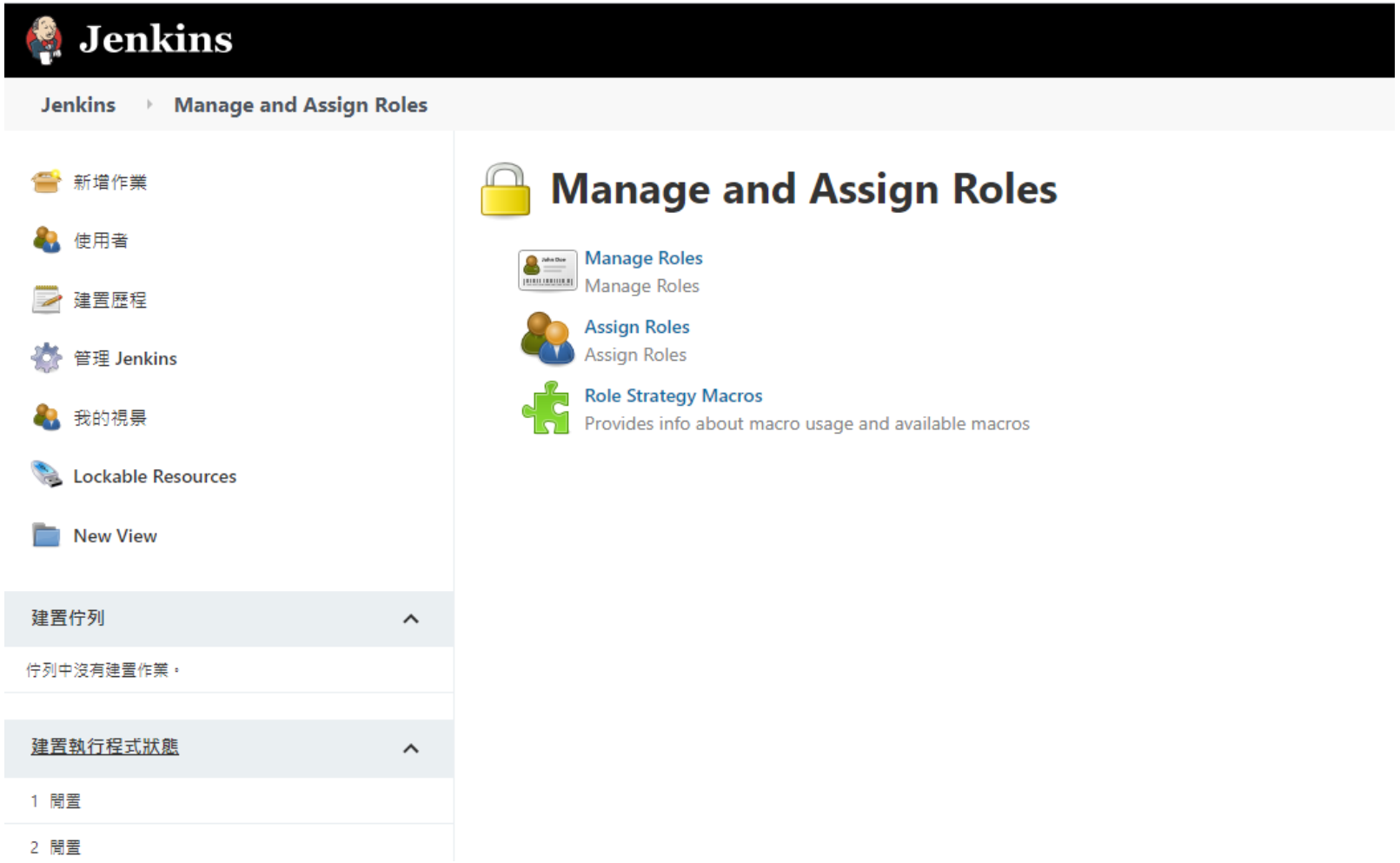
一行就是一個使用者或群組 (依據安全性領域不同，一般也叫做「角色」)。 包括 "anonymous" 特殊使用者，代表沒有通過驗證的人; 另外也有 "authenticated"，代表所有驗證通過的人 (換句話說，就是除了匿名使用者以外的所有人)。 透過表格下方的文字方塊可以新增使用者、群組、角色進來，按一下 [x] 圖示可以把它由表格中移掉。

儲存 Apply

權限設定: 管理jenkins --> Manage and Assign Roles

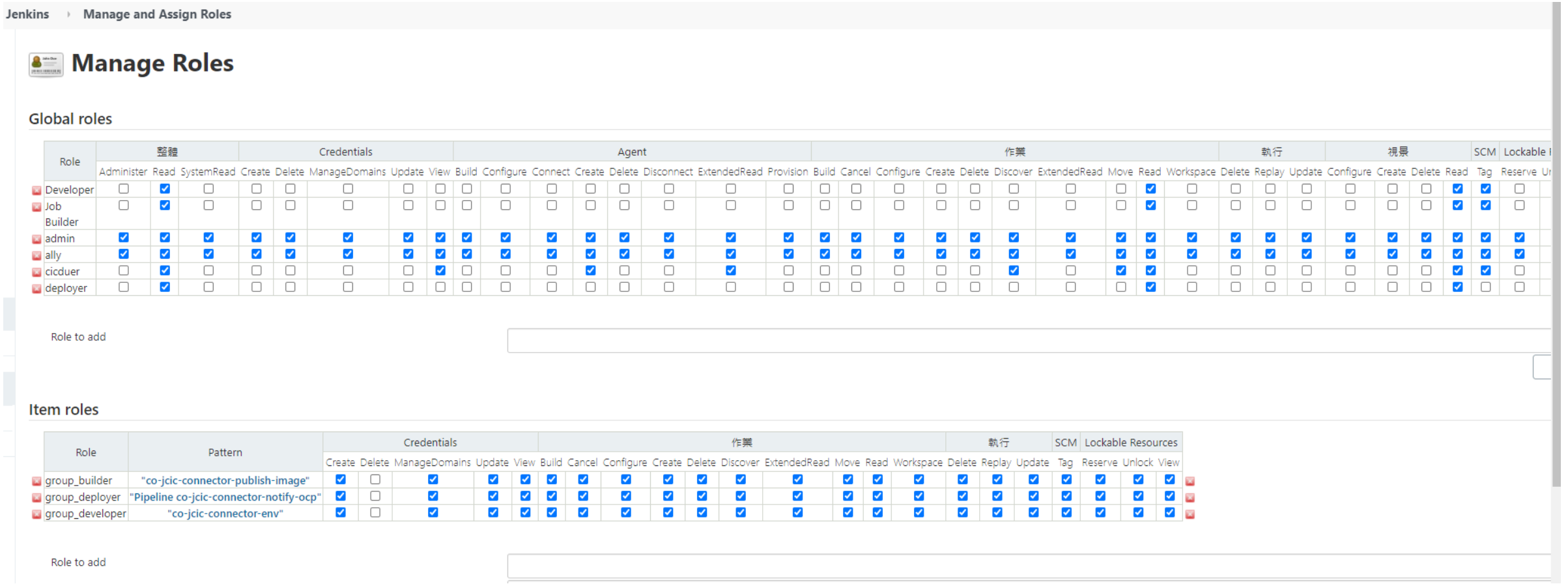


Manage and Assign Roles



Manage Roles

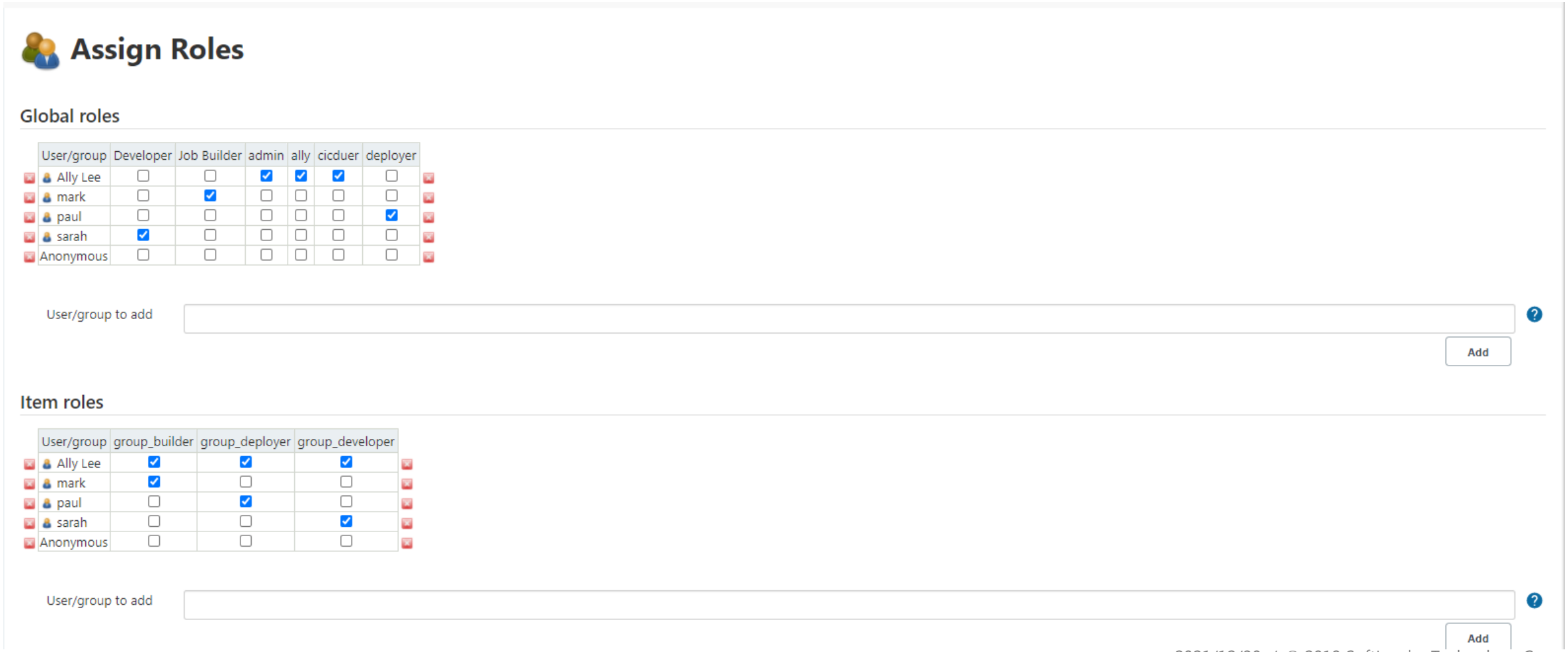
roles綁定它可以做的事情，或者item roles 綁定他可以用的專案 ，在這個專案可以做甚麼(patten那邊可以用正規表示法)



SoftLeader
Softleader Technology Corp.

Assign Roles

user 綁 roles (Global roles) (因為只有role可以綁可以做的事情)
user綁item role(類似group感覺，給專案做設定)



實作:

新增4個使用者 User_A，User_B , User_C ,User_D

有個 test floder，裡面有三個pipeline

test1,test2,test3

User_A ,是test專案的領導,可看到專案底下所有pipeline

User_B ,可以看到test專案, 並且可以使用test1 pipeline,

User_C, 可以看到test專案, 並且可以使用test2 pipeline,

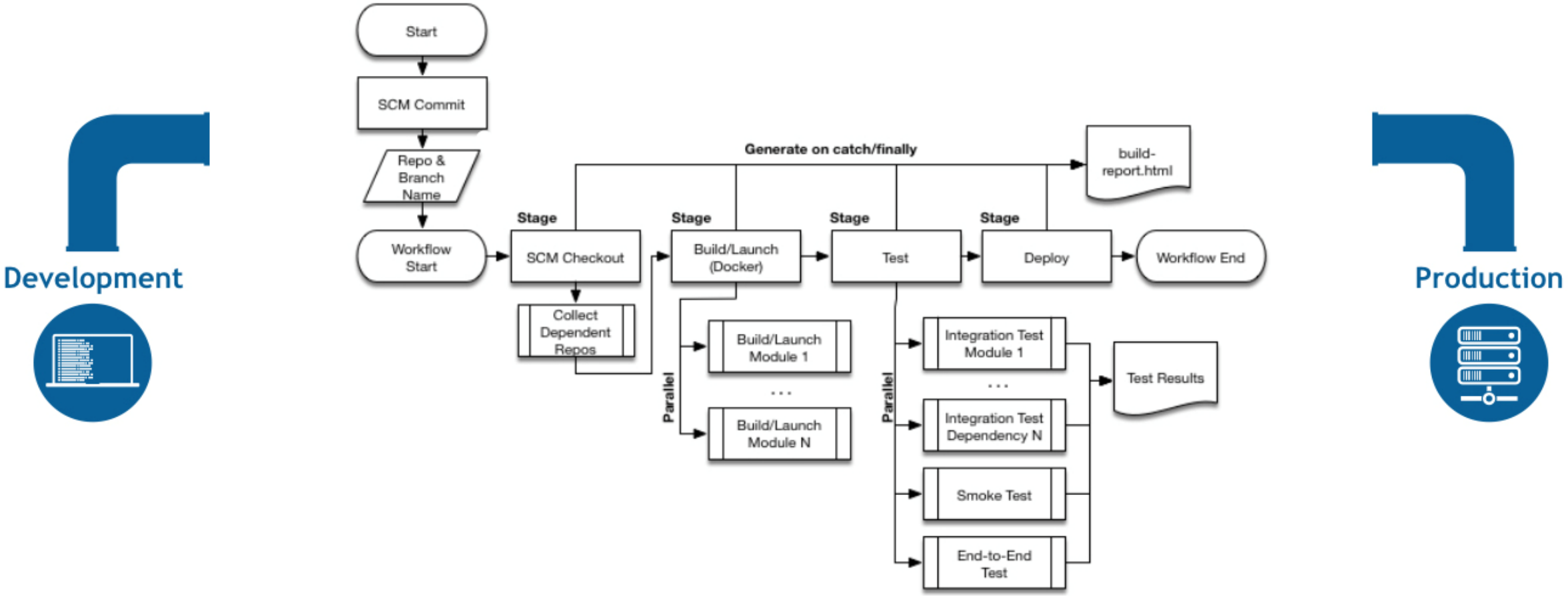
User_D,是訪客只能看見

Jenkins

Pipeline實作教學

Jenkins pipeline 介紹

Jenkins Pipeline是一套插件，支持將持續交付管道實現和集成到 Jenkins 中。



https://www.jenkins.io/doc/book/pipeline/

準備pipeline 所需環境

1. 安裝所需的plugin

準備 workspace環境 **Global Tool**

1. 找到包含的image

2. Dockerfile自製image

3. 嘗試是否可以把 mnt進去，或進入container安裝

pipeline 常用語法

```
pipeline {
    /* insert Declarative Pipeline here */
}
```

在聲明式流水線中有效的基本語句和表達式遵循與

- 流水線頂層必須是一個 `pipeline { }`
- 沒有分號作為語句分隔符，，每條語句都必須在自己的行上。
- 塊只能由 屬性引用語句被視為無參方法調用。例如, input被視為input()

agent

指定你pipeline將會在Jenkins環境中執行的位置，這取決於 該部分必須在 agent 的設定

```
pipeline {
    /* insert Declarative Pipeline here */
   agent any
}
```

any

在任何可用的代理上執行流水線或階段。例如: `agent any`

none

當在 比如: `pipeline` `stage` `agent` `agent none`

environment

設定這個agent內部的環境變數，或者特定步驟中的環境變數

```
pipeline {

    agent any

    environment {
        app_env = "uat"
        }
    stages {
        stage('準備環境') {
            steps {
                sh 'printenv'

            }
        }
    }
}
```

stages

在pipeline 用於設立多個pipeline階段關卡，包著多個的stage(關卡)

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

stage('Example')

設定階段性的關卡與關卡名稱

steps

在關卡內要執行的步驟，包含一個或多個步驟

JavaScript

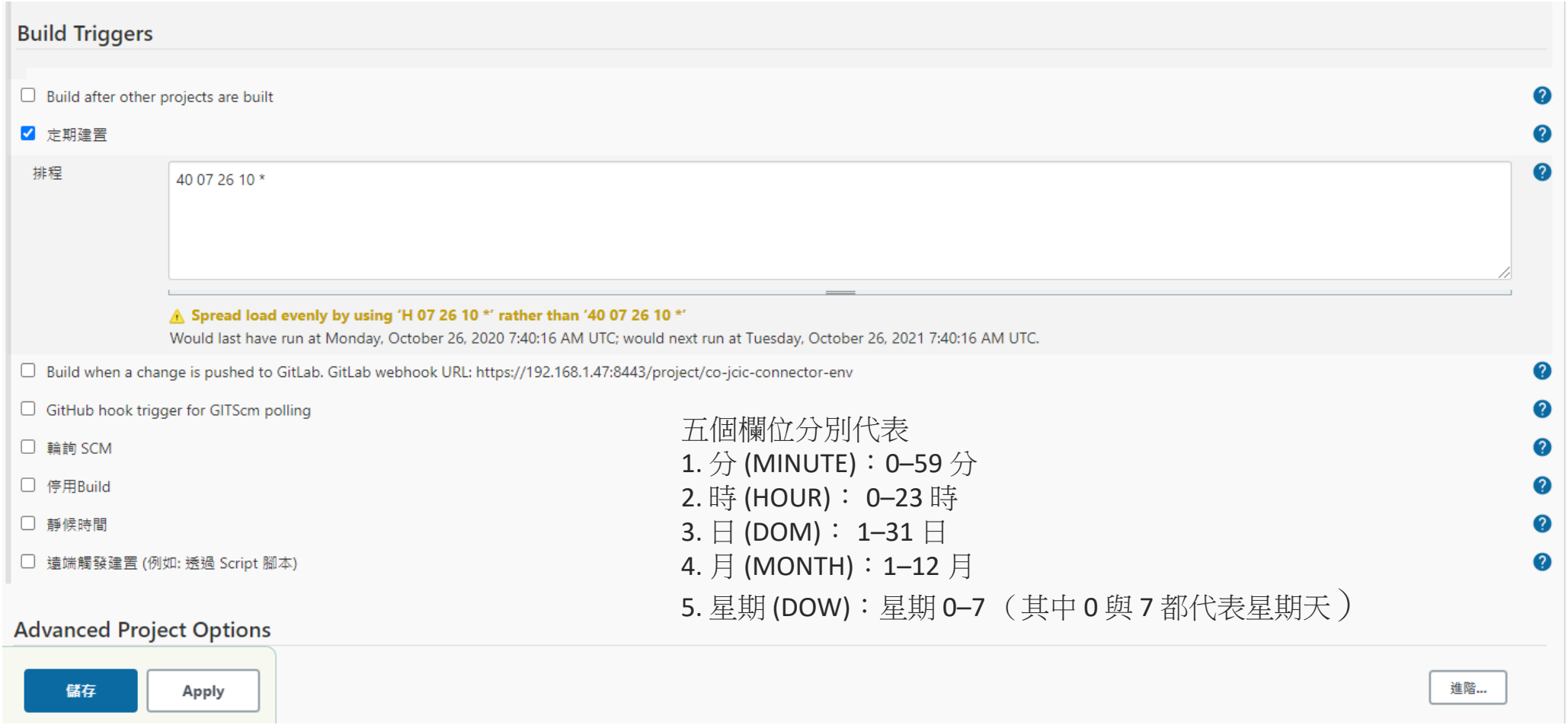
```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
                sh 'printenv'
            }
        }
    }
}
```

建立第一個pipeline

```
pipeline {
  agent any stages {
     stage('Example') {
        steps {
           echo 'Hello World'
        }
     }
  }
}
```

Pipeline的排程設定

使用cron表達時間，如: 00 10 26 10 * ，此設定為10月26日早上10:00會進行建置 (以jenkins內部時間為依據)



五個欄位分別代表

1. 分 (MINUTE)：0–59 分
2. 時 (HOUR)： 0–23 時
3. 日 (DOM)： 1–31 日
4. 月 (MONTH)：1–12 月
5. 星期 (DOW)：星期 0–7（其中 0 與 7 都代表星期天）

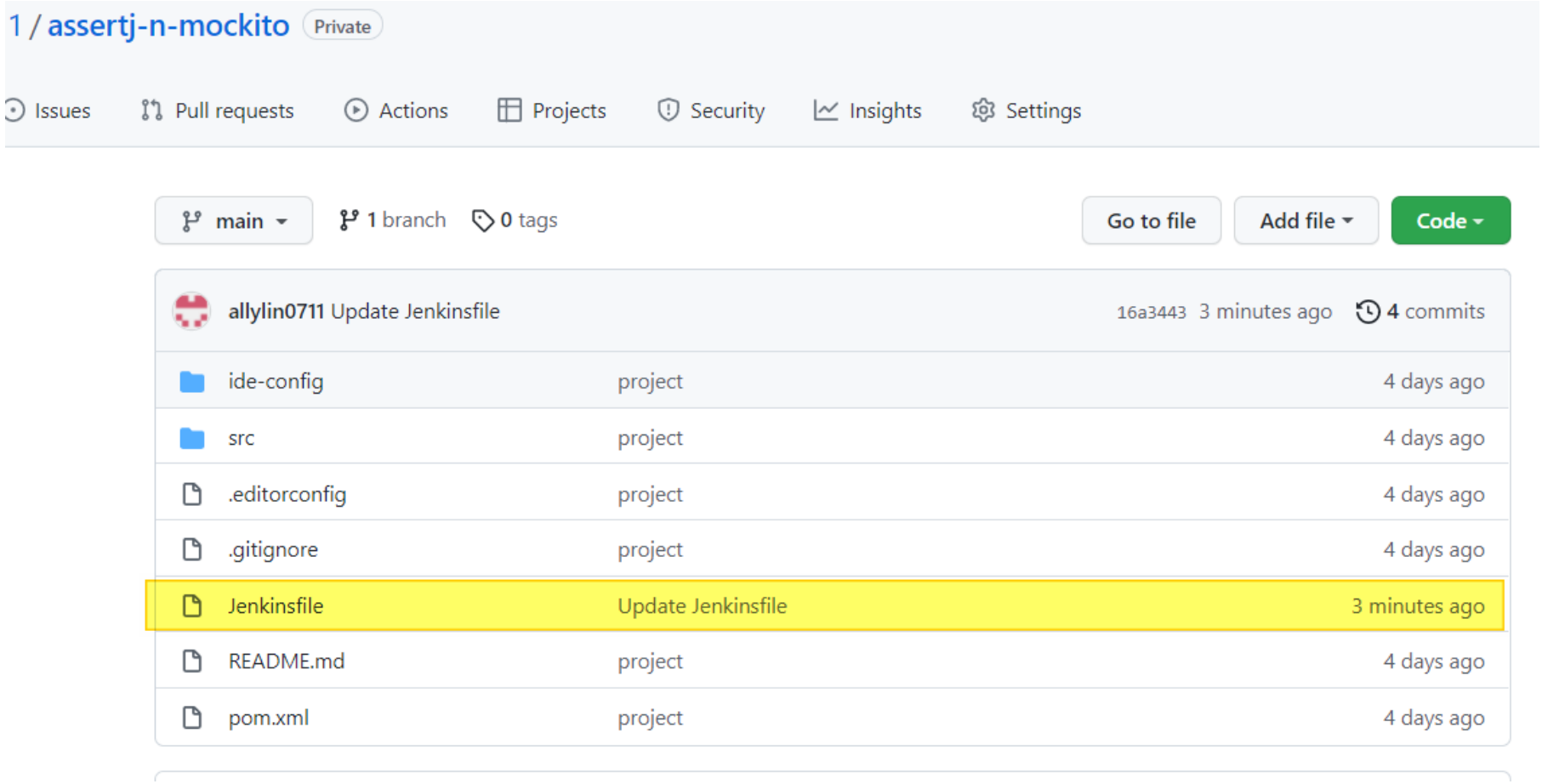
Jenkinsfile

Jenkinsfile 是Pipeline 的腳本，由Groovy語言構成。

- 方便code review 和迭代pipeline
- 方便追蹤pipeline的修改和紀錄每次變更
- 確保pipeline的來源正確性，方便專案成員的多人的追蹤和編輯

Jenkinsfile - github建立Jenkinsfile

在github建立Jenkinsfile



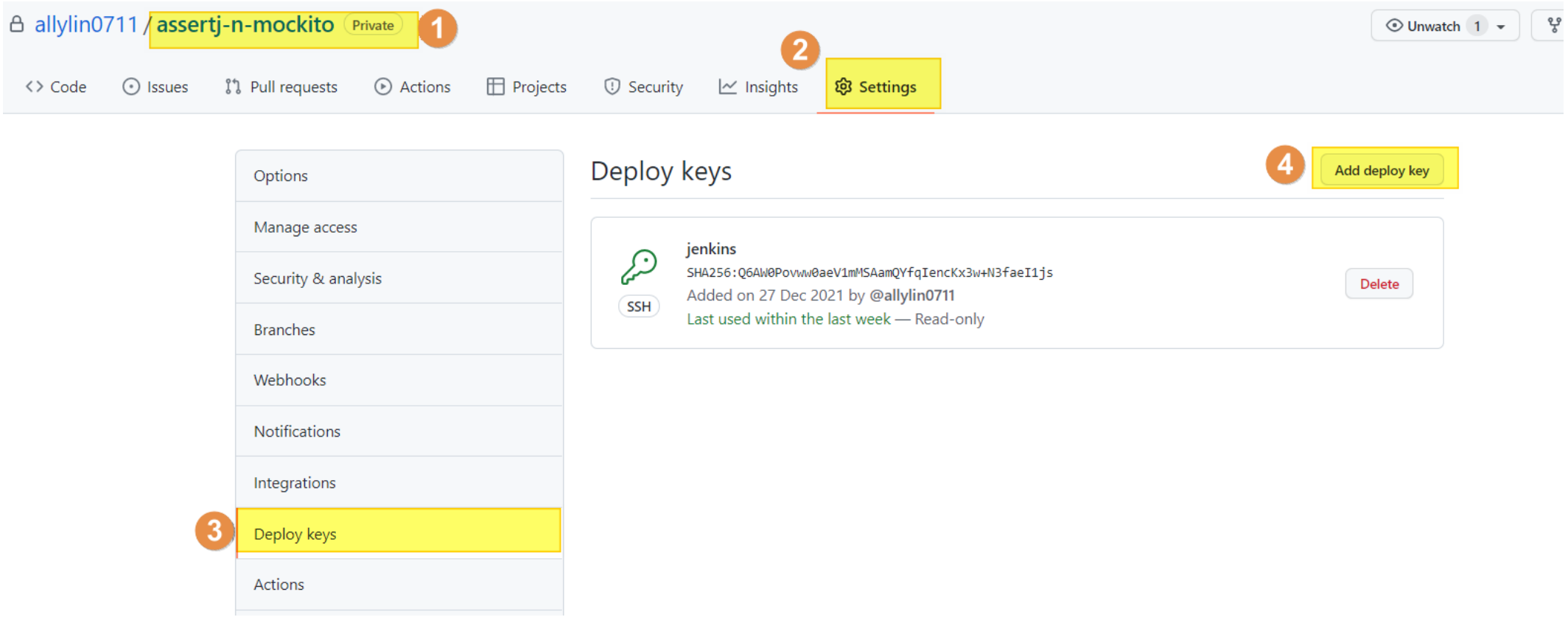
Jenkinsfile - 建立ssh key

建立ssh key ,記得要把key放在jenkins_home/.ssh內部且jenkins要有權限

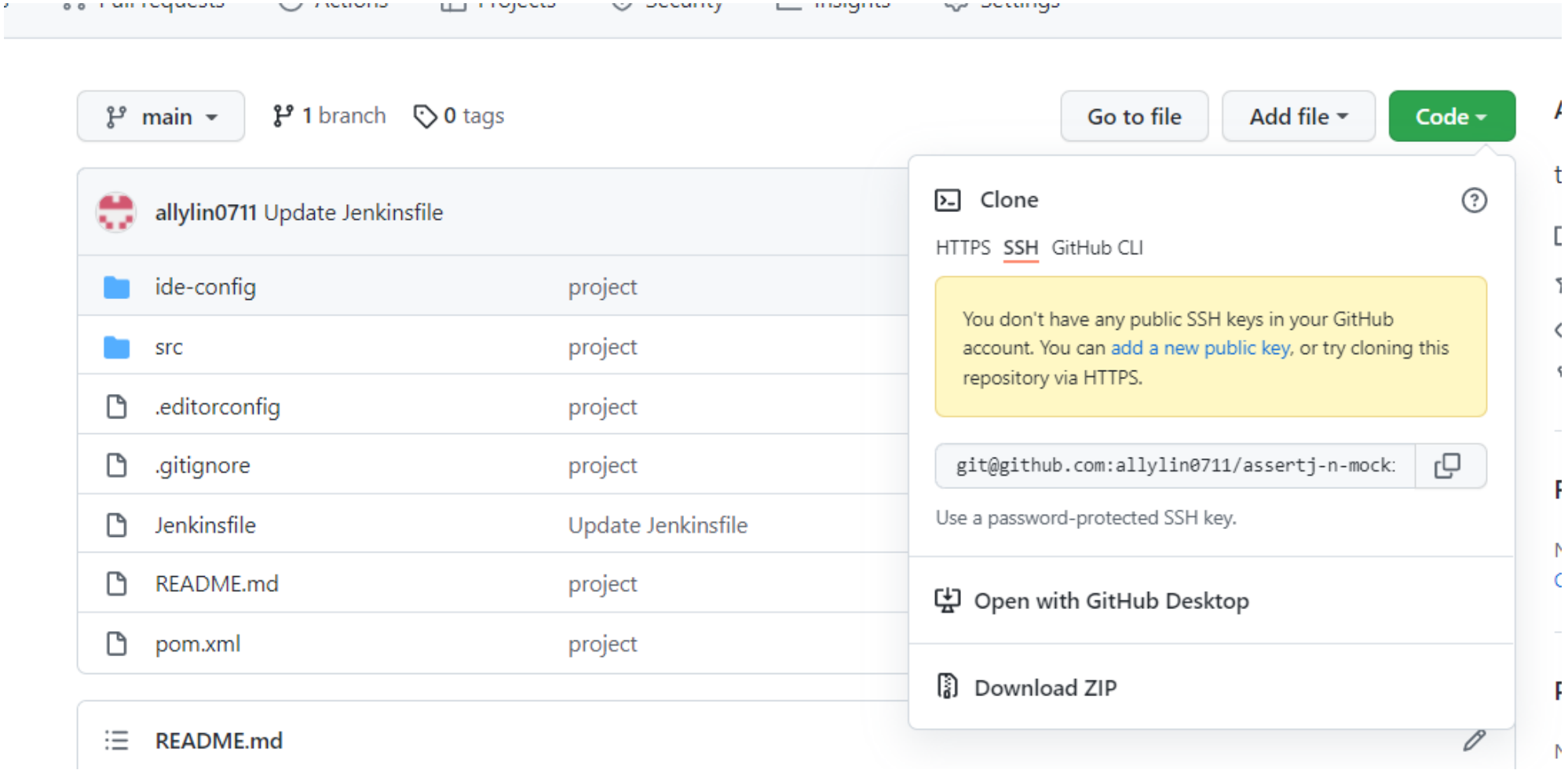
```
jenkins@9a4cf2dcfbce:/opt$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:Q6AWOPovwwOaeV1mMSAamQYfqIencKx3w+N3faeI1js jenkins@9a4cf2dcfbce
The key's randomart image is:
+---[RSA 3072]----+
|.o+=o o.         |
|..++.+ .o        |
|.+o.o   .o       |
|+ *.   .+        |
|.* + . +S        |
|o + O .  .       |
| . B *   o       |
|  . = + o.E.. .  |
|     + o. o+.o   |
+----[SHA256]-----+
jenkins@9a4cf2dcfbce:/opt$ ls
id_rsa  id_rsa.pub  java  jenkins-plugin-manager.jar  maven
```

Jenkinsfile - 在github建立deploy key

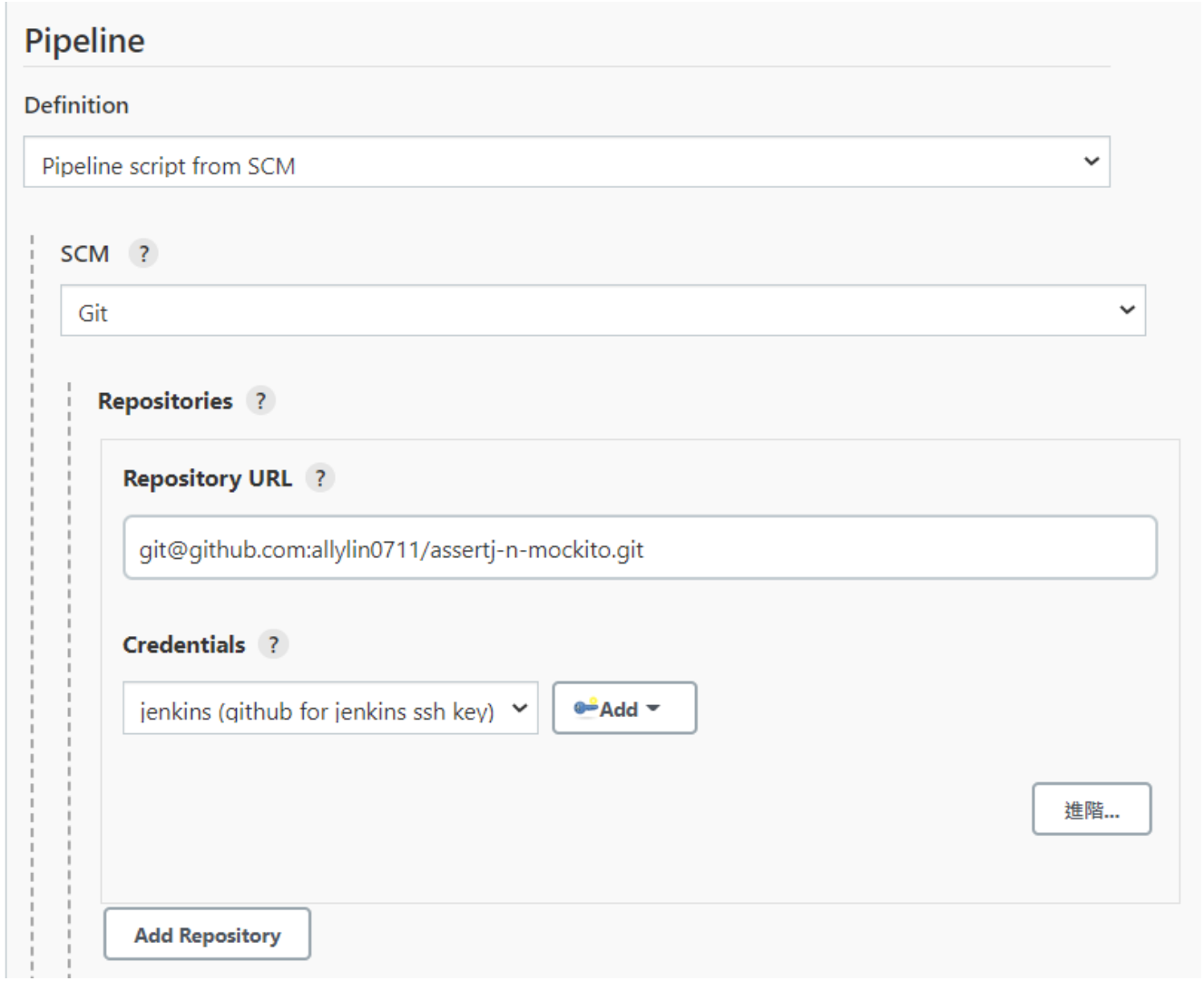
Id_rsa.pub是公鑰放在github deploy keys內

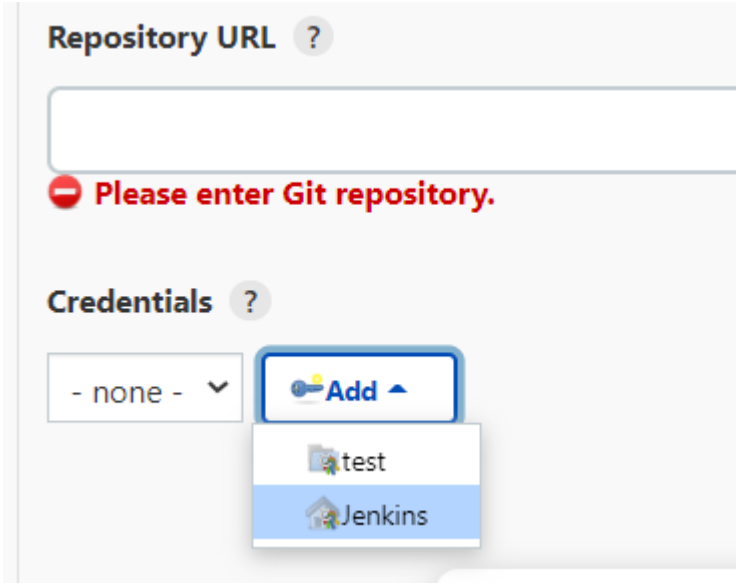


Jenkinsfile - 跟github的SSH連結

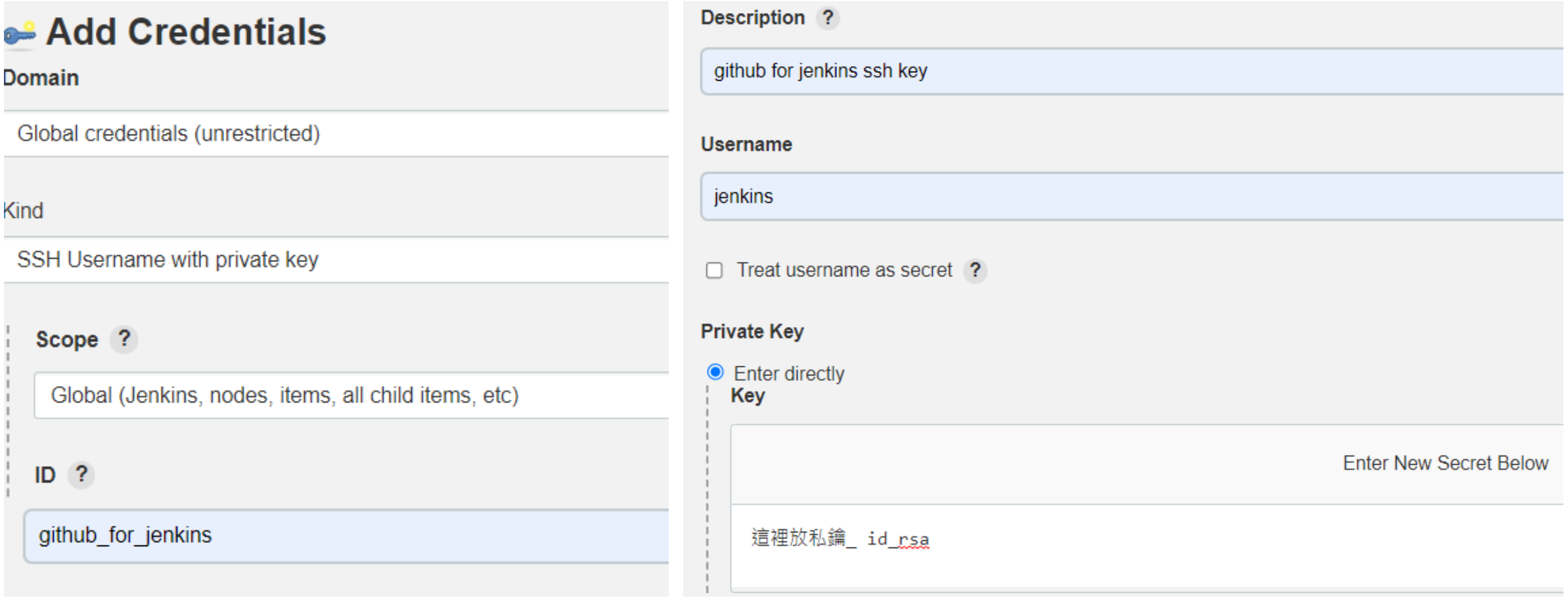


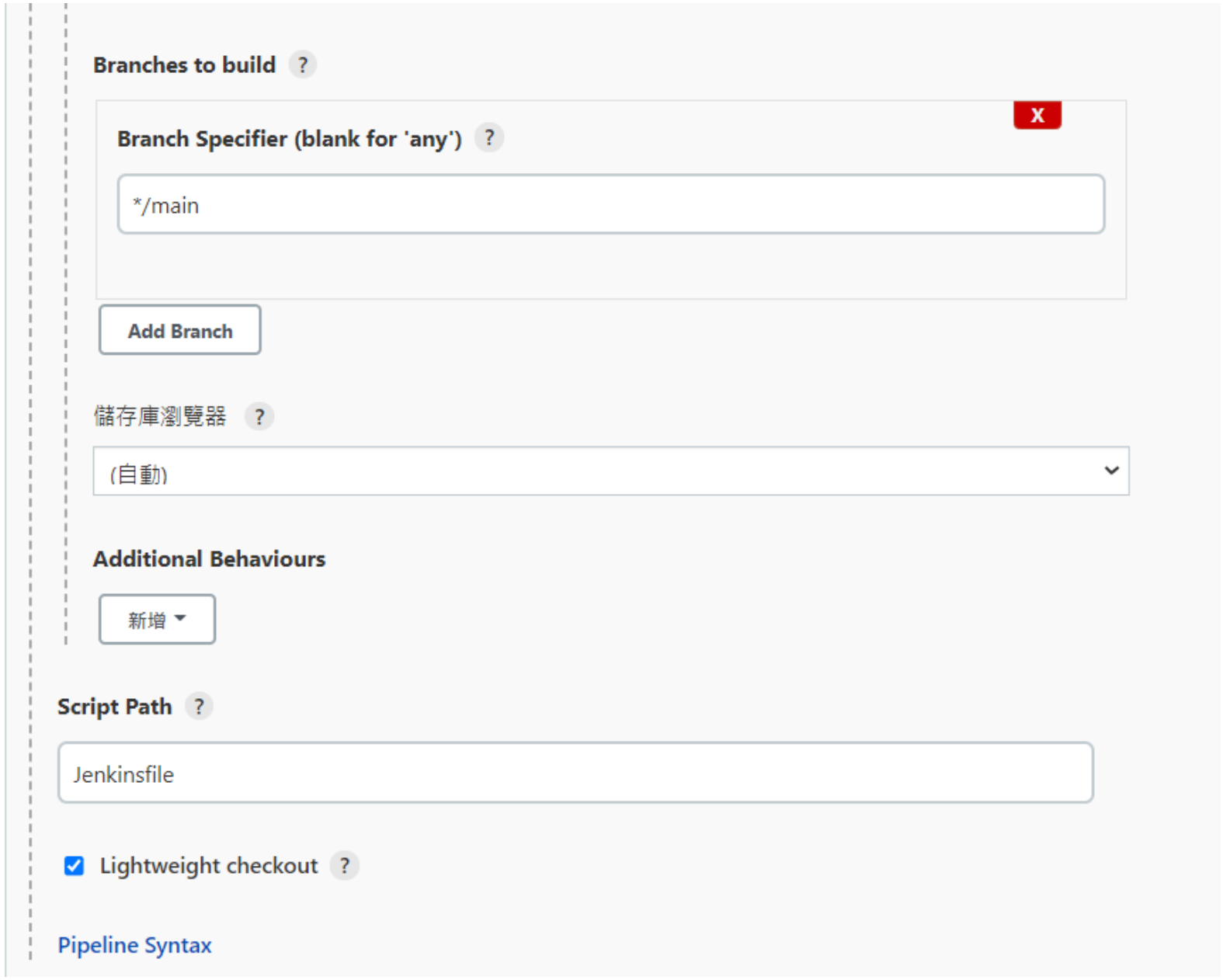
Jenkinsfile -建立pipeline



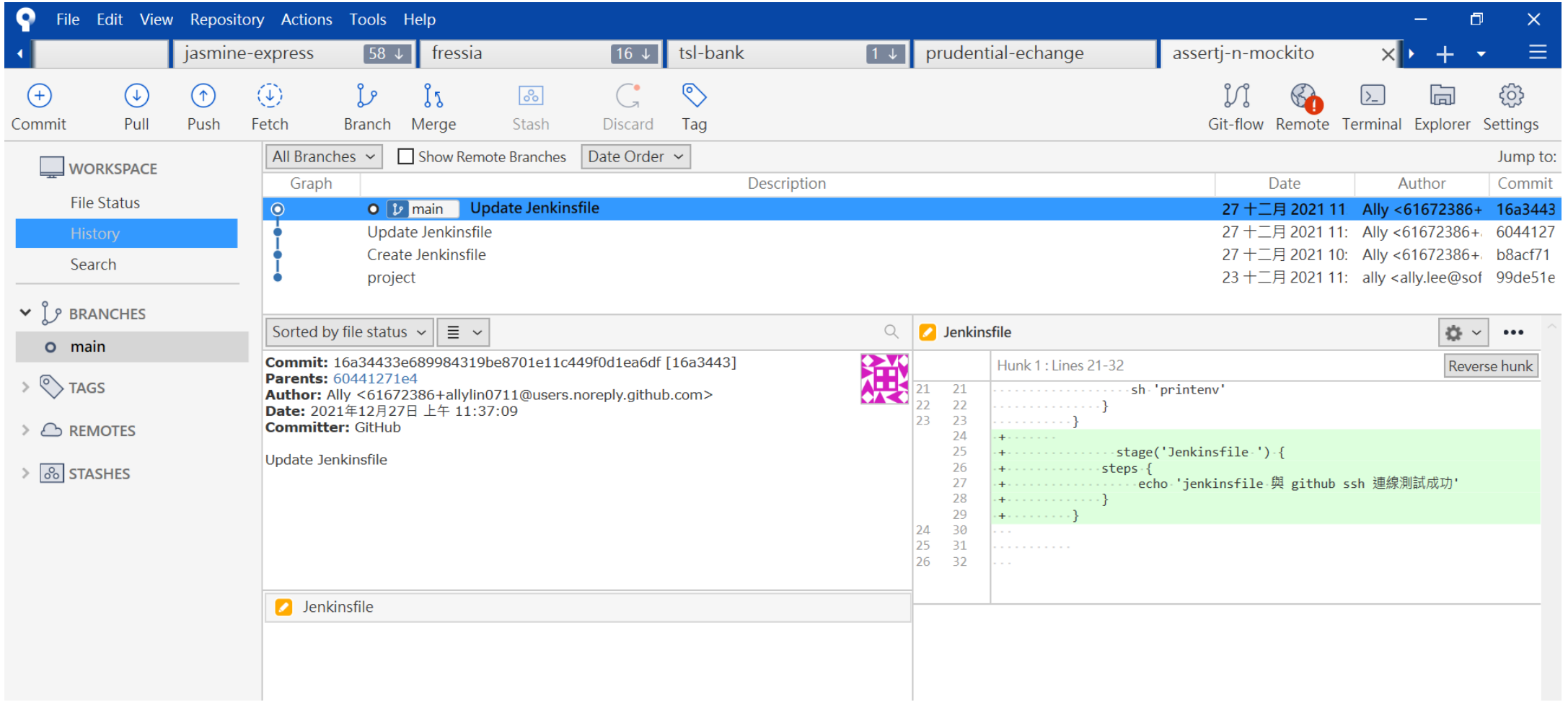


Jenkinsfile -新增add credentials




Branches to build ?
Branch Specifier (blank for 'any') ?
X
*/main
Add Branch
儲存庫瀏覽器 ?
(自動)
Additional Behaviours
新增
Script Path ?
Jenkinsfile
Lightweight checkout ?
Pipeline Syntax

Jenkinsfile –可以看到每次的commit紀錄和差異



環境準備 - 新增maven , java版本

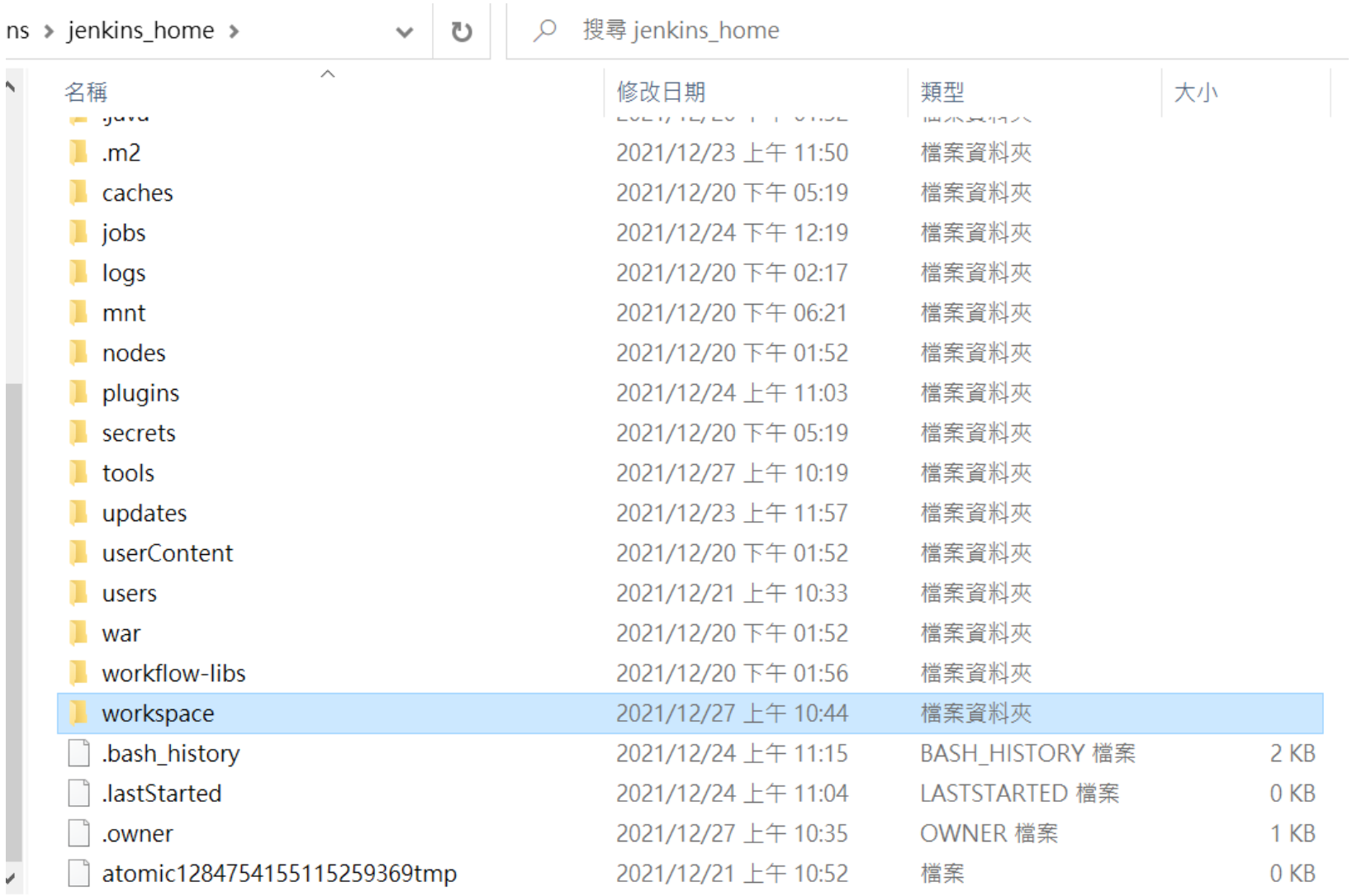
```yaml
version: '3.8'
services:
  jenkins:
    image: jenkins/jenkins:2.325-jdk8
    restart: always
    ports:
      - 8082:8080
      - 50002:50000
#      - 8443:8443
#    environment:
#      - "JENKINS_OPTS= --httpPort=8080 --httpsPort=8443 --httpsKeyStore=/var/ssl/serverl
    volumes:
#      - C:\Users\admin\Desktop\Dockerfile\47\jenkins\firtbank\compose\ssl:/var/ssl
      - C:\Users\admin\Desktop\fisrtbank_Workshop\jenkins\jenkins_home:/var/jenkins_home
      - C:\Users\admin\.m2:/var/jenkins_home/.m2
      - /var/run/docker.sock:/var/run/docker.sock
      - C:\Users\admin\Desktop\fisrtbank_Workshop\jenkins\opt:/opt
```

```
C:\Users\admin\Desktop\testpipeline\bak\mimosa-prudential-echange-master\console>docker ps
CONTAINER ID   IMAGE                        COMMAND                  CREATED      STATUS      PORTS
                                    NAMES
9a4cf2dcfbce   jenkins/jenkins:2.325-jdk8   "/sbin/tini -- /usr/…"   3 days ago   Up 3 days   0.0.0.0:8082->8080/tcp, :::8082->
8080/tcp, 0.0.0.0:50002->50000/tcp, :::50002->50000/tcp   jenkins_jenkins_1

C:\Users\admin\Desktop\testpipeline\bak\mimosa-prudential-echange-master\console>docker exec -it 9a /bin/bash
jenkins@9a4cf2dcfbce:/$ cd /opt
jenkins@9a4cf2dcfbce:/opt$ ls
java  jenkins-plugin-manager.jar  maven
jenkins@9a4cf2dcfbce:/opt$ ls -l
total 6048
drwxrwxrwx 1 root    root       4096 Dec 24 02:57 java
-rw-r--r-- 1 jenkins jenkins 6192861 Dec 20 10:21 jenkins-plugin-manager.jar
drwxrwxrwx 1 root    root       4096 Dec 23 04:39 maven
jenkins@9a4cf2dcfbce:/opt$
```

SoftLeader
Softleader Technology Corp.

Pipeline的 workspace位置



SoftLeader
Softleader Technology Corp.

.m2檔位置

```
drwxr-xr-x.  3 1000 mygroup     24  8月 26 12:19 .m2
-rw-r--r--.  1 1000 mygroup    216 10月 26  2020 njnm.plugins.aws_batch.AwsBatchBuilder.xml
-rw-r--r--.  1 1000 mygroup    907  8月 23 17:43 nodeMonitors.xml
drwxrwxrwx.  2 1000 cicduser     6  9月 22  2020 nodes
-rw-r--r--.  1 1000 mygroup    275 10月 26  2020 org.jenkinsci.plugins.github_branch_source.GitHubConfigu
-rw-r--r--.  1 1000 mygroup     46  8月 30 11:36 org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xm
-rw-r--r--.  1 1000 mygroup    153 10月 26  2020 org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDur
-rw-r--r--.  1 1000 mygroup    219 10月 26  2020 org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml
-rw-r--r--.  1 1000 mygroup    236 10月 26  2020 org.jenkins.plugins.lockableresources.LockableResourcesM
-rw-r--r--.  1 1000 mygroup     42  9月  3 09:32 .owner
drwxrwxrwx. 87 1000 cicduser 12288  8月 30 11:14 plugins
-rw-r--r--.  1 1000 mygroup    131  8月 30 11:36 queue.xml
-rw-r--r--.  1 1000 mygroup    131  8月 23 17:31 queue.xml.bak
-rw-r--r--.  1 1000 mygroup    369 10月 26  2020 scriptApproval.xml
-rwxrwxrwx.  1 1000 cicduser    64  9月 22  2020 secret.key
-rwxrwxrwx.  1 1000 cicduser     0  9月 22  2020 secret.key.not-so-secret
drwxrwxrwx.  4 1000 cicduser  4096 10月 26  2020 secrets
drwxrwxrwx.  2 1000 cicduser   182  9月  2 17:43 updates
drwxrwxrwx.  2 1000 cicduser    24  9月 22  2020 userContent
drwxrwxrwx.  7 1000 cicduser   184 10月  8  2020 users
drwxrwxrwx. 11 1000 cicduser  4096  9月 22  2020 war
drwxrwxrwx.  2 1000 cicduser     6  9月 22  2020 workflow-libs
drwxr-xr-x. 12 1000 mygroup    208  8月 30 11:35 workspace
[root@localhost jenkins]# pwd
/data/jenkins/test_test/jenkins
[root@localhost jenkins]#
```

若執行pipeline 的git指令中遇到有ssl錯誤

若在git clone錯誤,
可在pipeline中加入或者在Jenkins container 內加入:

```
git -c http.sslVerify=false clone
```

```
git config --global http.sslVerify false
```


Jenkinsfile (Declarative Pipeline)

```
pipeline {

  agent any stages {
       stage(' source code ' ) {
                steps {
                          echo ' git source code.. '
                      }
          }

       stage('Build') {
             steps {
                       echo 'Building.. '
                    }
       }

       stage('Test') {
            steps {
                       echo 'Testing.. '
                  }
       }
       stage('Deploy') {
             steps {
                       echo 'Deploying.... '
                   }
       }
   }
}
```

撰寫自己的pipeline ~

寫pipeline將source code從建置包版

SoftLeader
Softleader Technology Corp.

給個小提示~

```
stages {
    stage('準備環境') {
        steps {
            sh 'java -version'
            sh 'mvn -version'
            sh 'git --version'
            sh 'pwd'
            sh 'printenv'
        }
    }
    stage ('srouce code') {
      steps {
            sh 'pwd'
            sh 'rm -Rf  assertj-n-mockito'
            sh 'git clone git@github.com:allylin0711/assertj-n-mockito.git'

          }
      }

    stage ('目前目錄，進入目錄') {
      steps {
            sh 'pwd'
             dir("assertj-n-mockito") {

                sh 'pwd'

             }
          }
      }

     stage ('清理環境') {
      steps {
          dir("assertj-n-mockito") {
               sh "mvn clean -e  "

           }

          }
      }
```