Tutorial - Jenkins

Working with Integration Tool: Jenkins

This tutorial is going to cover the following topics:

- Jenkins core concepts
- Jenkins installation
  - Installing Jenkins using Docker
  - Using a Virtual Machine for Jenkins
  - Not using Virtual Machine? Fresh Install Jenkins
- After installed Jenkins: Jenkins setup
- Accessing Jenkins
- Ngrok installation
- Integrate Jenkins with GitHub using web-hooks
- Continuous integration using Jenkins
- **Exercises (At the end of this tutorial)**

Jenkins Core Concepts

What is Jenkins? Jenkins is an automated server which helps in the software development process. For example, it can be used to trigger code builds when there is changes made to the source code repository.

Below find the steps for installing and configuring Jenkins. The installation can be difficult, so pay attention to the steps carefully while configuring Jenkins.

Installing Jenkins

Installing Jenkins using Docker

One of the easiest ways of setting up Jenkins is through Docker. We will cover docker next week, however for now, know that docker a software that allows delivering software in containers.

You must install docker first. Follow **this tutorial (https://docs.docker.com/get-docker/)** to install docker.

Once docker is installed, run the following command:

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 --name jenkins je
nkins/jenkins:lts
```

To get the unlock password as described in the **After installed Jenkins: Jenkins setup** section, run:

```
docker exec -it jenkins /bin/bash
jenkins@c537b805f333:/$ cat /var/jenkins_home/secrets/initialAdminPassword
```

And keep note of the password.

To stop the containers whenever needed, run:

```
docker stop jenkins
```

Visit **localhost:8080 (http://localhost:8080)** for the Jenkins portal.

Jump to **After installed Jenkins: Jenkins setup** after having done these steps.

Using a Virtual Machine for Jenkins

We have a Virtual Machine ready for you that has Jenkins and ngrok installed respectively. You may download this from **here (https://canvas.sydney.edu.au/courses/25875/pages/soft2412-virtual-machine?module_item_id=864405)**. You may be prompted to update the network settings to the wifi interface before the machine loads. The password for the user "soft2412" is "**soft2412**". Kindly note that gradle is not installed in the VM....figure out the workaround to this without installing gradle...?? (Think about wrapper).

After you login to the Ubuntu machine. visit **localhost:8080 (http://localhost:8080)** in firefox, you should see a jenkins login page. Again, the username and password is same as soft2412. After you have done this, jump to **Integrate Jenkins with GitHub** section of the tutorial**.**

Not using Virtual Machine? - Fresh Install Jenkins

If the virtual machine runs slow on your machine or you cannot use docker, you may choose to install and run jenkins as a software in your own PC.

For that you will have to install Jenkins in your machine if you do not have it.

Linux users 

For ubuntu/debian users, we will use the apt-get package. For other distributions, use yum/dnf/pacman.

```
sudo apt-get update
sudo apt update
```

Then, install java:

```
sudo apt install openjdk-8-jdk
```

Then, import the public GPG key of jenkins:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

For ubuntu/debian users, run the following command:

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/je
nkins.list'
```

Then you can run:

```
sudo apt update
sudo apt install jenkins
```

Then, after jenkins gets successfully installed, run the following commands:

```
sudo systemctl start jenkins
```

You can check the status of jenkins through:

```
sudo systemctl status jenkins
```

Output:

```
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Fri 2019-09-06 18:10:18 AEST; 2min 3s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 4658)
   CGroup: /system.slice/jenkins.service

Sep 06 18:10:17 soft2412-VirtualBox systemd[1]: Starting LSB: Start Jenkins at boot time...
Sep 06 18:10:17 soft2412-VirtualBox jenkins[7260]: Correct java version found
Sep 06 18:10:17 soft2412-VirtualBox jenkins[7260]:  * Starting Jenkins Automation Server jenk
ins
Sep 06 18:10:17 soft2412-VirtualBox su[7308]: Successful su for jenkins by root
```

```
Sep 06 18:10:17 soft2412-VirtualBox su[7308]: + ??? root:jenkins
Sep 06 18:10:17 soft2412-VirtualBox su[7308]: pam_unix(su:session): session opened for user j
enkins by (uid=0)
Sep 06 18:10:17 soft2412-VirtualBox su[7308]: pam_unix(su:session): session closed for user j
enkins
Sep 06 18:10:18 soft2412-VirtualBox jenkins[7260]:    ...done.
Sep 06 18:10:18 soft2412-VirtualBox systemd[1]: Started LSB: Start Jenkins at boot time.
```

Finally. enable jenkins to start on the system boot:

```
sudo systemctl enable jenkins
```

You may find this link helpful: https://linuxize.com/post/how-to-install-jenkins-on-centos-7/

Jump to **After installed Jenkins: Jenkins setup.**

MacOS users

For MacOSX users, you will need to install jenkins through brew. Brew is a package manager for Mac and Linux.

Type brew to check if you have it installed:

```
brew --version
```


If you have brew installed you should see something similar to:

```
Homebrew 2.1.11
Homebrew/homebrew-core (git revision a5bdae; last commit 2019-08-26)
Homebrew/homebrew-cask (git revision 8442f; last commit 2019-08-26)
```

Make sure java is installed.

```
java --version
```

Now, to install jenkins, run:

```
brew install jenkins
```

This will take some time. You should see the following after the installation:

```
...
==> Downloading http://mirrors.jenkins.io/war/2.193/jenkins.war
==> Downloading from http://ftp-nyc.osuosl.org/pub/jenkins/war/2.193/jenkins.war
######################################################################## 100.0%
==> jar xvf jenkins.war
```

```
==> Caveats
Note: When using launchctl the port will be 8080.

To have launchd start jenkins now and restart at login:
brew services start jenkins
Or, if you don't want/need a background service you can just run:
jenkins
==> Summary
🍺  /usr/local/Cellar/jenkins/2.193: 7 files, 78MB, built in 5 minutes 1 second
```

Then, you can start jenkins by running the following command: (or just run "jenkins")

```
brew services start jenkins
```

Output:

```
==> Successfully started `jenkins` (label: homebrew.mxcl.jenkins)
```


You can access **localhost:8080** **(http://localhost:8080)** to visit the jenkins homepage.

Jump to **After installed Jenkins: Jenkins setup** after having done these steps in Mac.

Windows Users

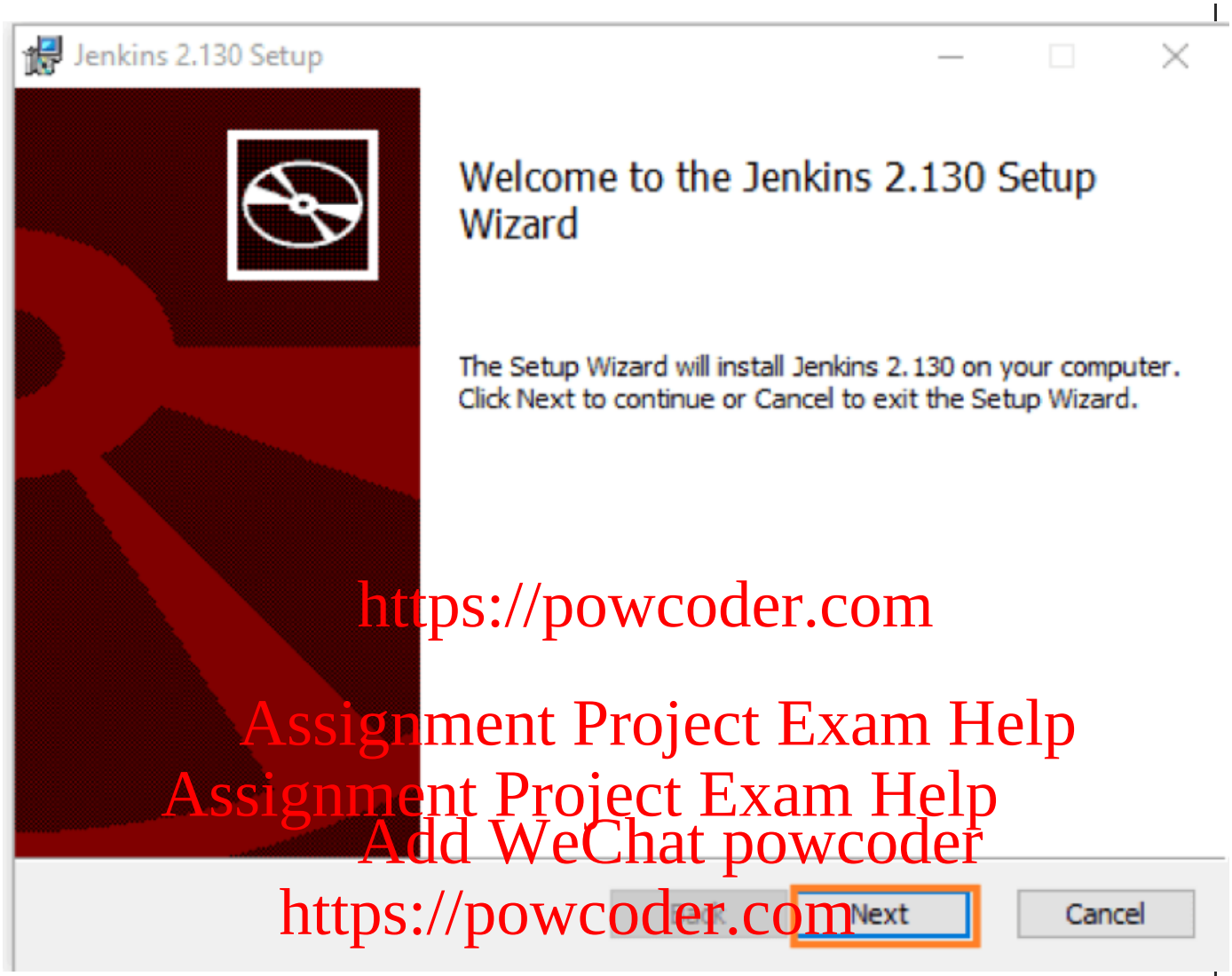
Before installing Jenkins, make sure that you have installed Java (JDK).

```
java -version
```

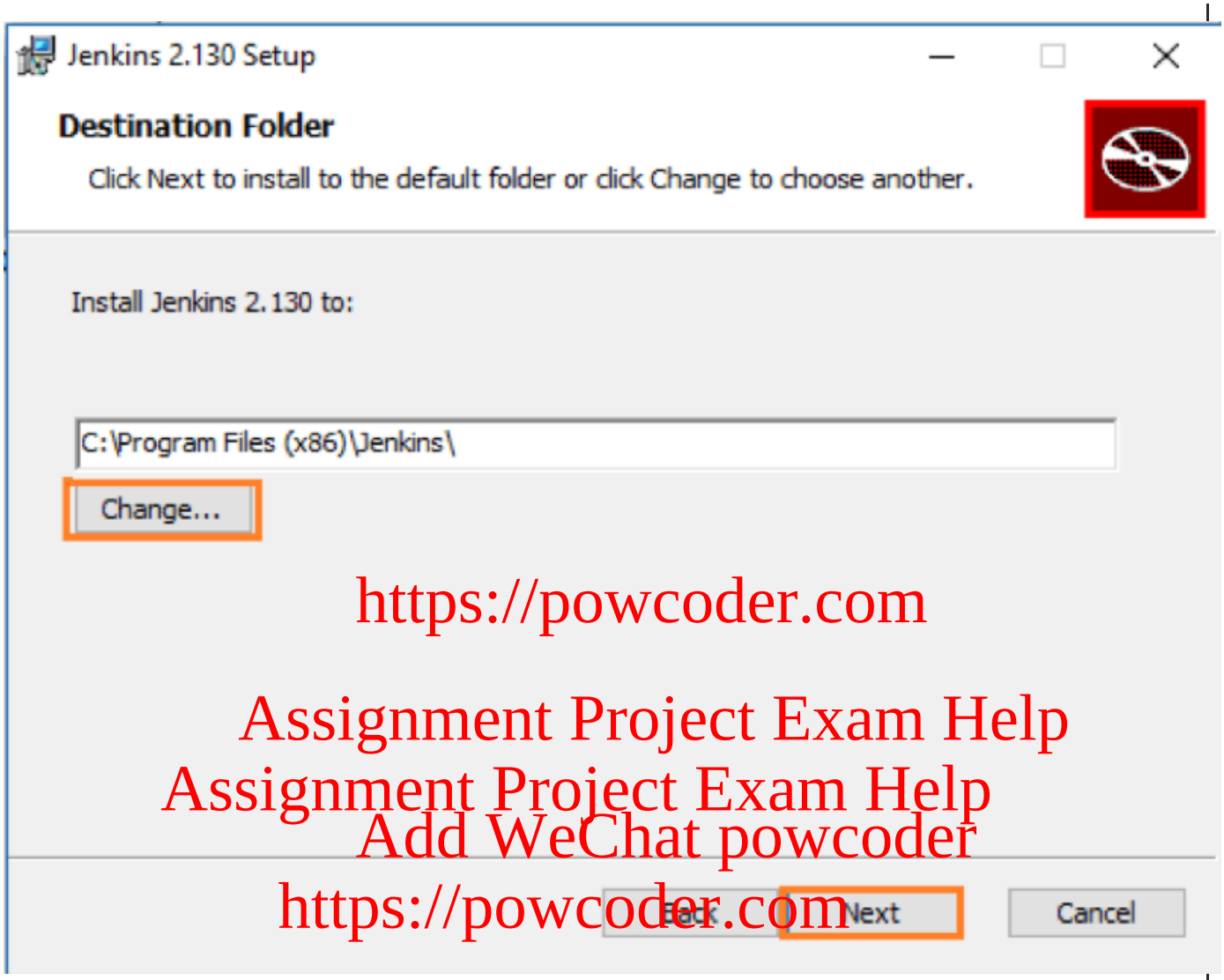
You can download Jenkins installer from this link: **https://jenkins.io/download/** **(https://jenkins.io/download/)**

After you select ‘windows’, your browser will automatically download the installer. Unzip the downloaded file to get “jenkins.exe” ready. These are the step to install Jenkins:

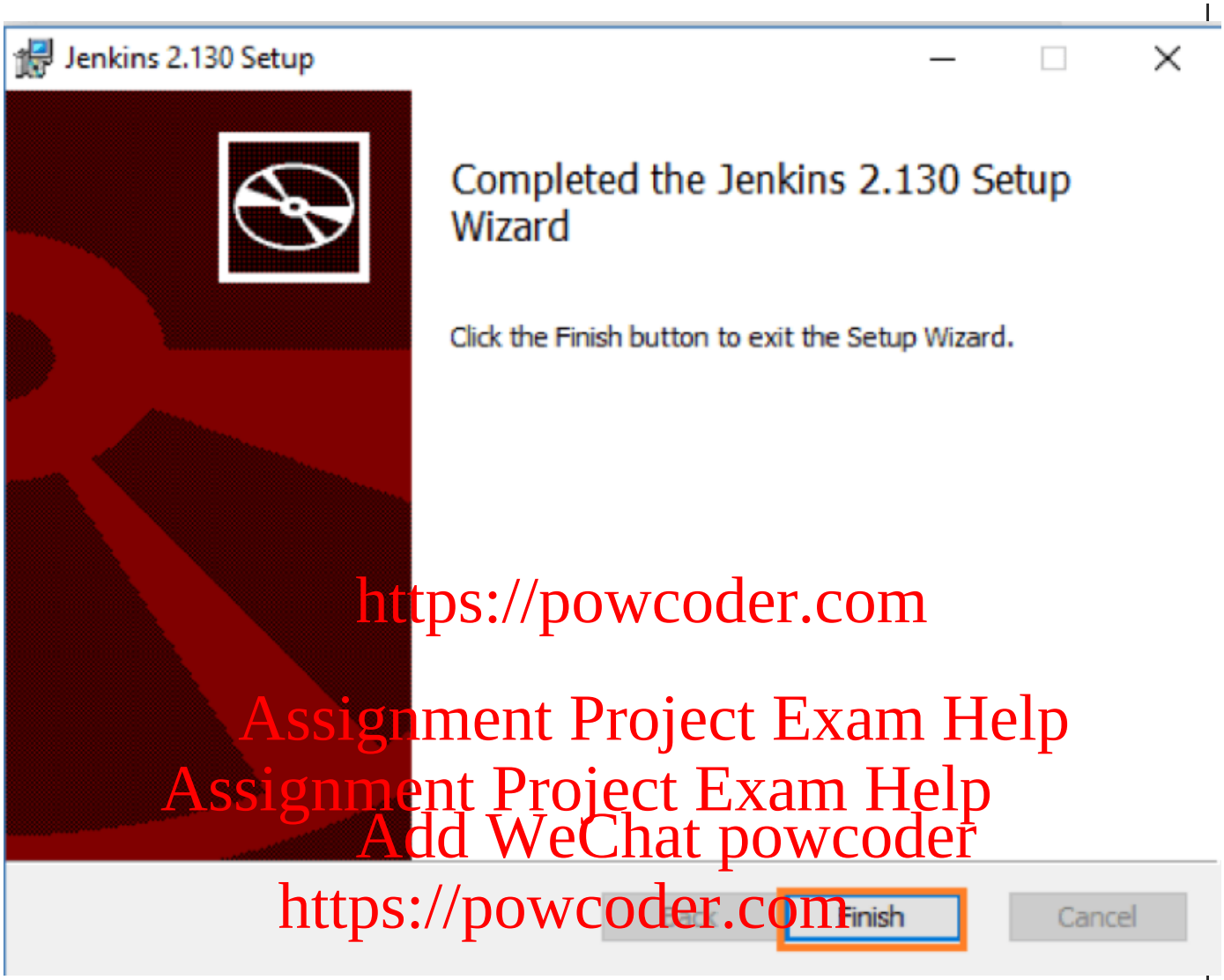
Click “Next” button to start the setup.



Click “Change” if you want to install Jenkins in another folder. Do remember this installation path as we will need the path later. Press “next” to continue the installation.



Click the “Install” button to start the installation and wait until it is done. Afterwards, click “Finish” button after the installation is complete.

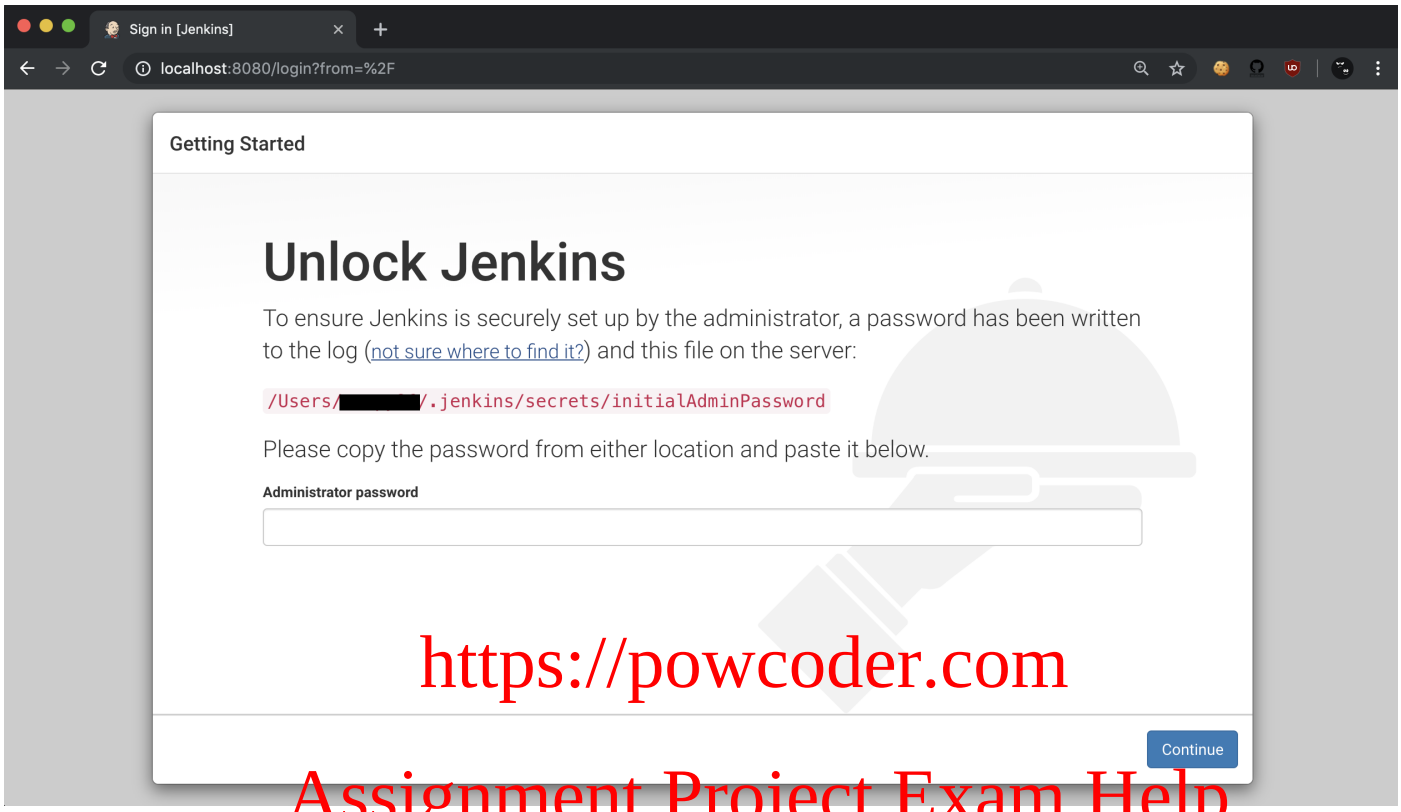


You will have to wait for at least 3-5 minutes as the Jenkins service is starting in the background (as a Windows's service). Afterwards, open your browser and navigate to page: **http://localhost:8080** (http://localhost:8080) . (http://localhost:8080/)

Jump to **After installed Jenkins: Jenkins setup**

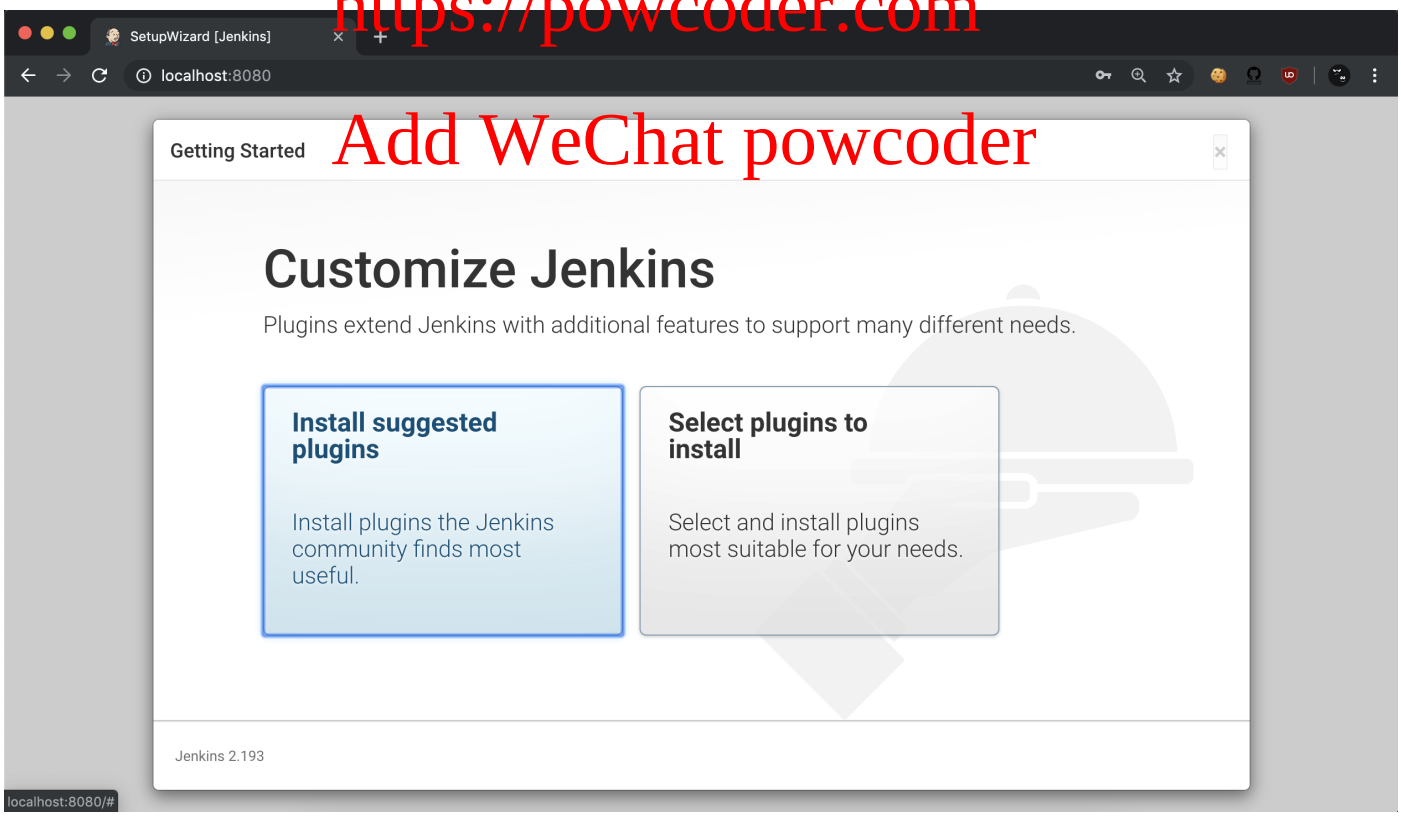
After installed Jenkins: Jenkins setup

Then visit **http://localhost:8080** (http://localhost:8080) to view the homepage of jenkins. You will be promted to enter a password

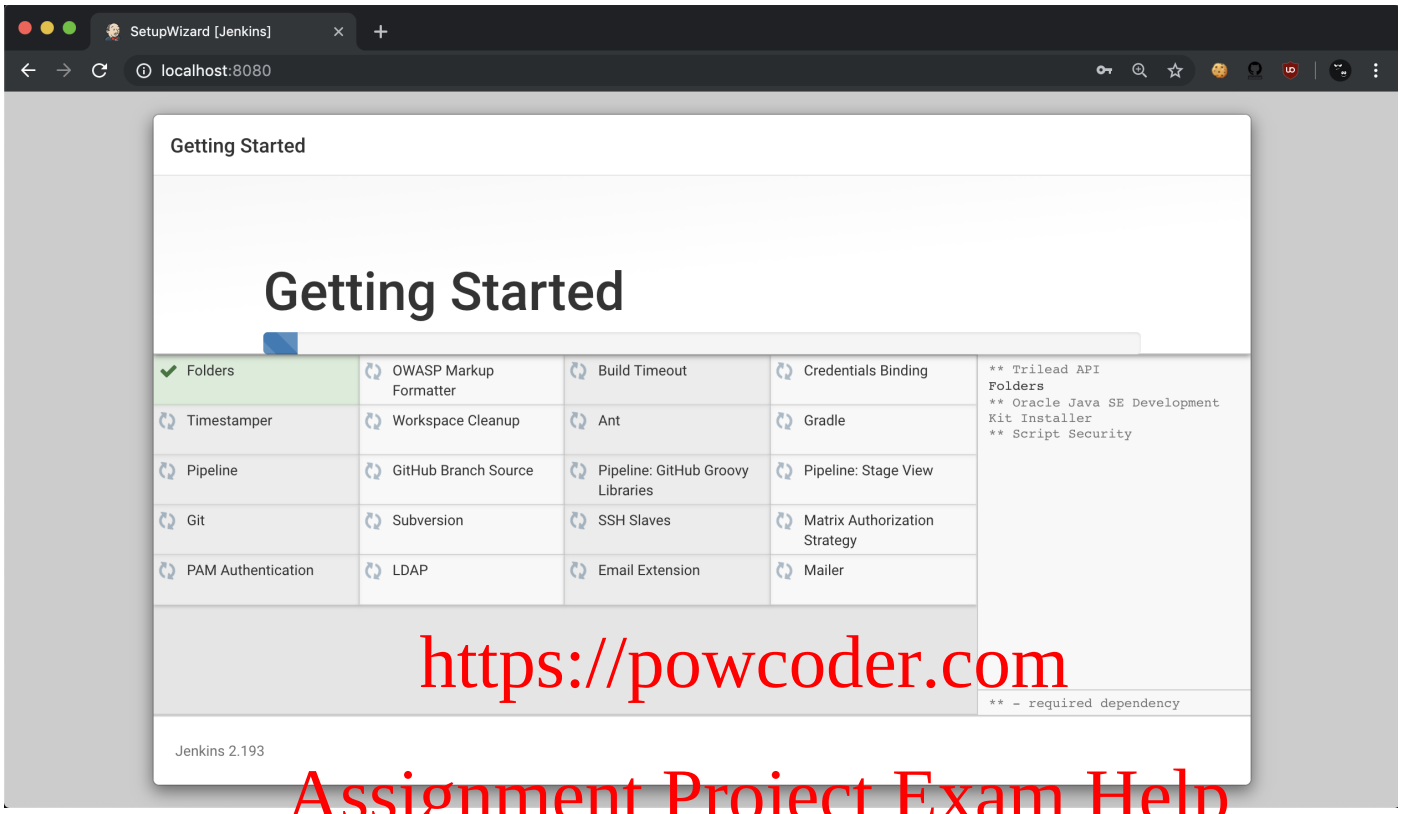


Copy paste the the password from the file specified in the above image.

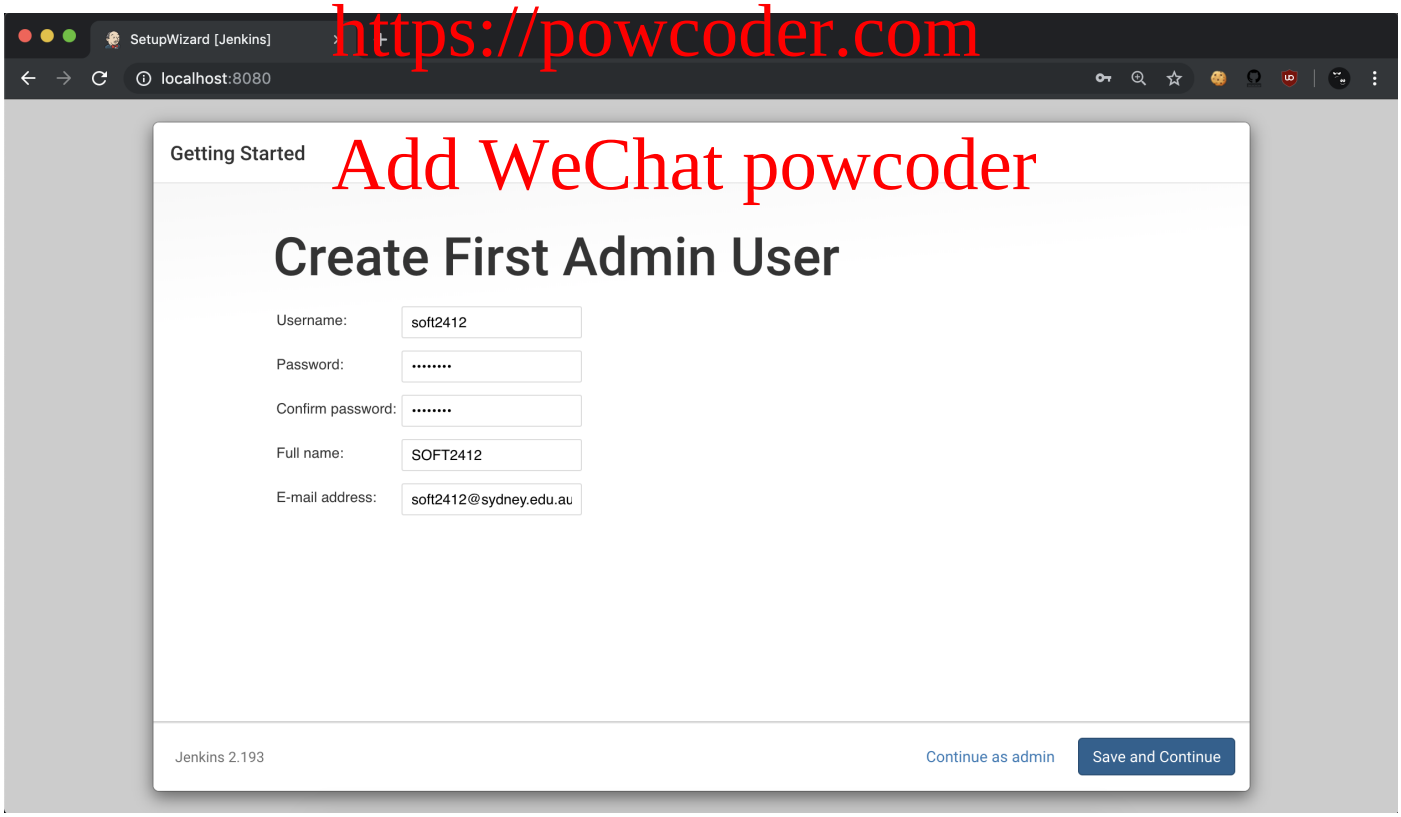
When you do so, you should be redirected here:



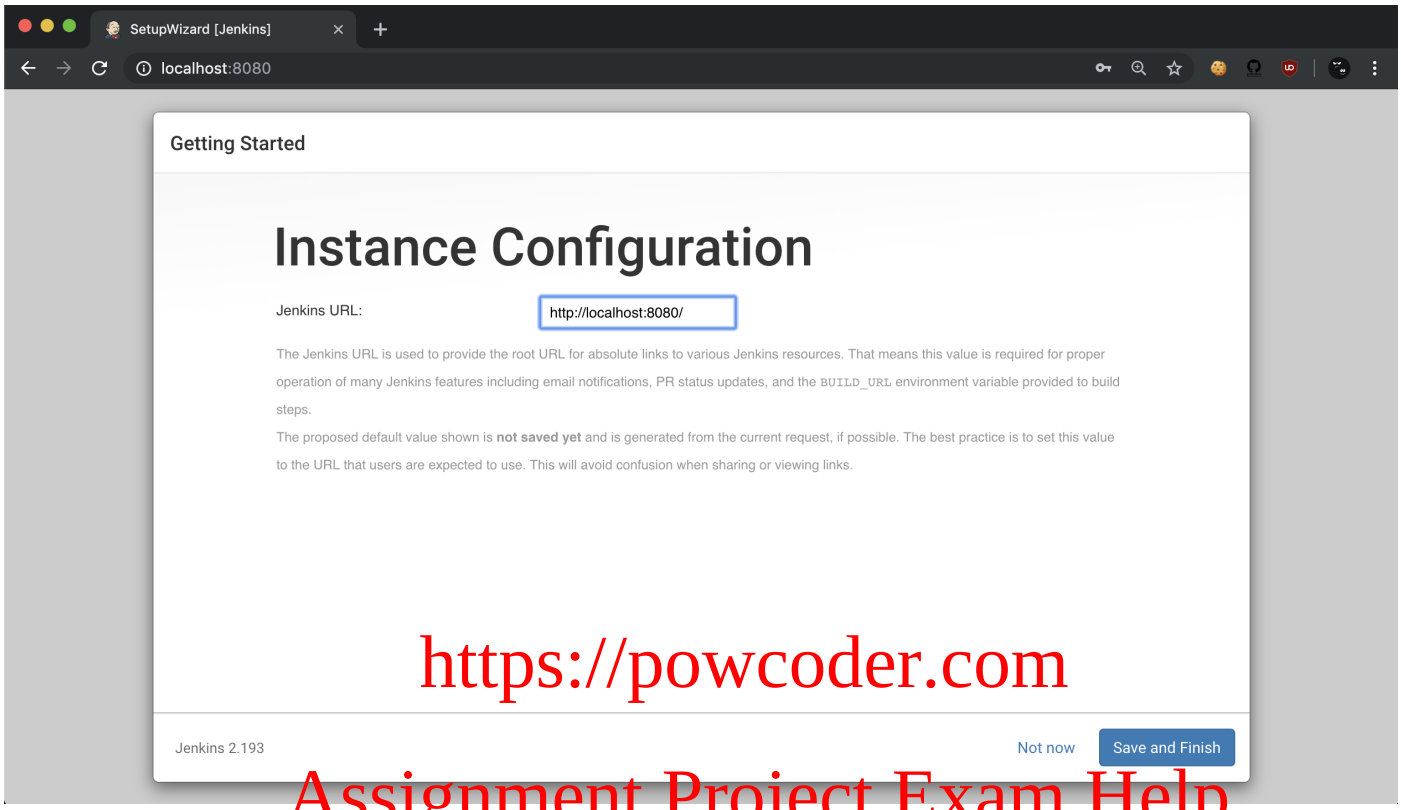
This will install packages. It may take some time



After the installation is completed, you will be asked to create a admin user. Fill in the admin credentials and click "Save and continue"



Then you will be asked to set the url. Let it remain default:



Click on "Save and Finish" and then click on "Start using Jenkins". You may have to restart jenkins. You can now start using it.

Accessing Jenkins

If you have installed jenkins in your own machine: You should visit [localhost:8080 (http://localhost:8080)](http://localhost:8080)
If you are using the virtual machine: You can access jenkins in firefox in the VM.

Installing Ngrok

Since Jenkins is running on localhost at port 8080, we need to expose it on the internet in order to allow collaboration. We will use a free software called ngrok that provides this.

You can download ngrok from here: [https://ngrok.com/download (https://ngrok.com/download)](https://ngrok.com/download). Follow the steps there to unzip ngrok. Then, cd to the directory containing the ngrok program and run the following command:

```
ngrok http 8080
```

Output:

```
@inconshreveable                                (Ctrl+C to quit)

Session Status online
Account <Name>
Version 2.3.34
Region United States (us)
```

```
Web Interface http://127.0.0.1:4040
Forwarding http://f9e633c8.ngrok.io -> http://localhost:8080
Forwarding https://f9e633c8.ngrok.io -> http://localhost:8080

Connections ttl opn rt1 rt5 p50 p90
0 0 0.00 0.00 0.00 0.00
```

You should get a public IP address that expires in 8 hours. **You could create an account in ngrok's website to extend that time, however you can obtain a new domain by typing the above command again after the current session expires.**

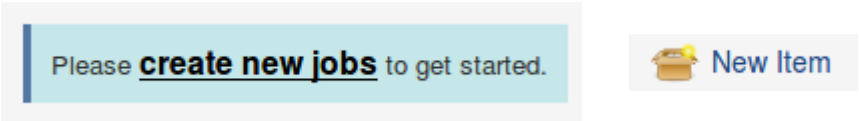
To create an ngrok account for free sign up here (https://dashboard.ngrok.com/signup) and then follow the instructions in the ngrok dashboard.

The image here shows the obtained the URL **https://f9e633c8.ngrok.io (https://f9e633c8.ngrok.io)** (https://9387d23f.ngrok.io/) . You will obtain a different one. Keep a note of this URL as you will require it later. Make sure you take a note of the "https" one and not the "http" one for obvious reasons.

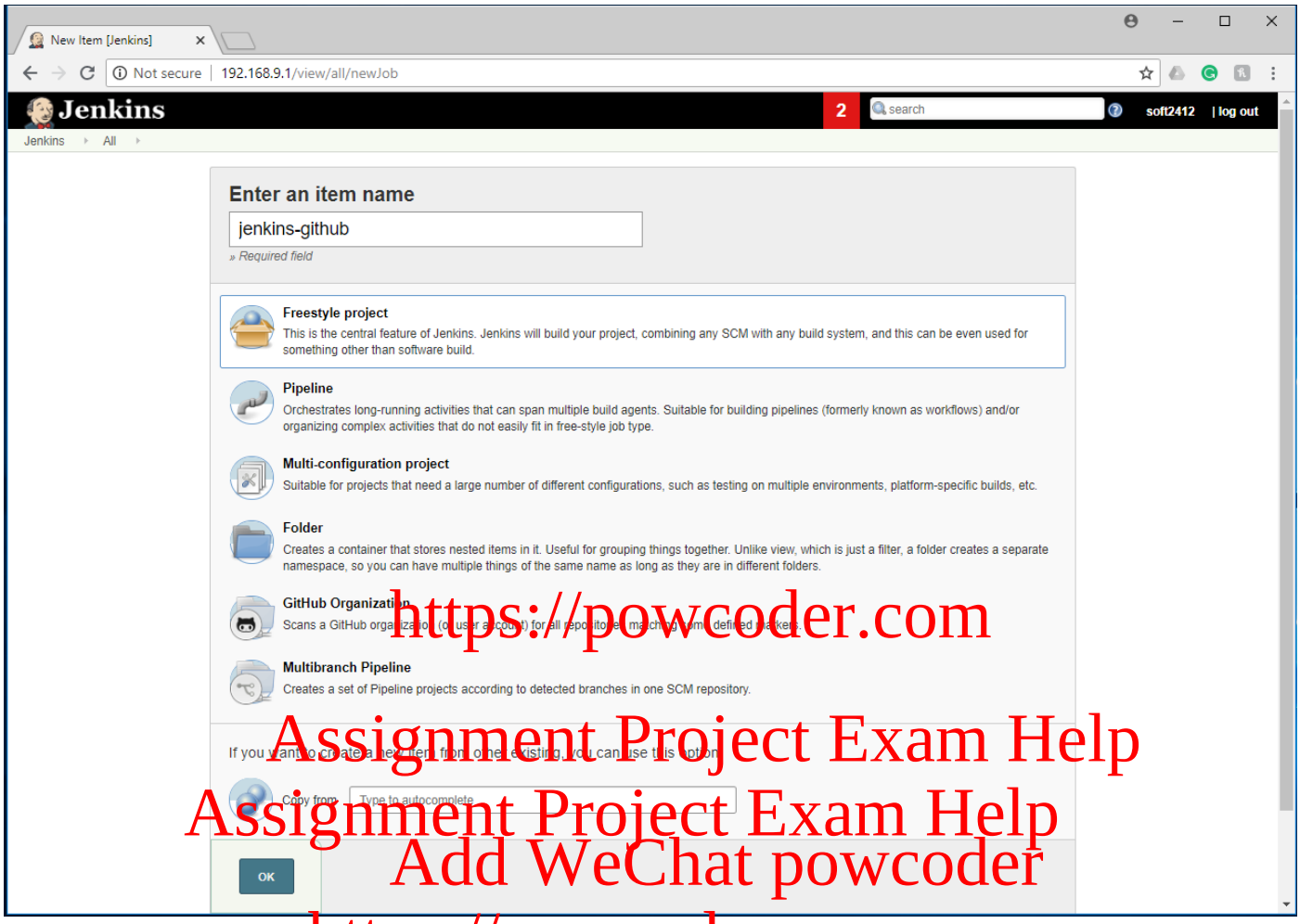
Integrate Jenkins with GitHub using web-hooks

Integrating Jenkins with Github provides developers with an automated checking of new code builds. It helps to determine if new code builds are a success or a failure. This ensures that erroneous codes is identified when committed to the final or master repository where it will be used for eventual product or application development.

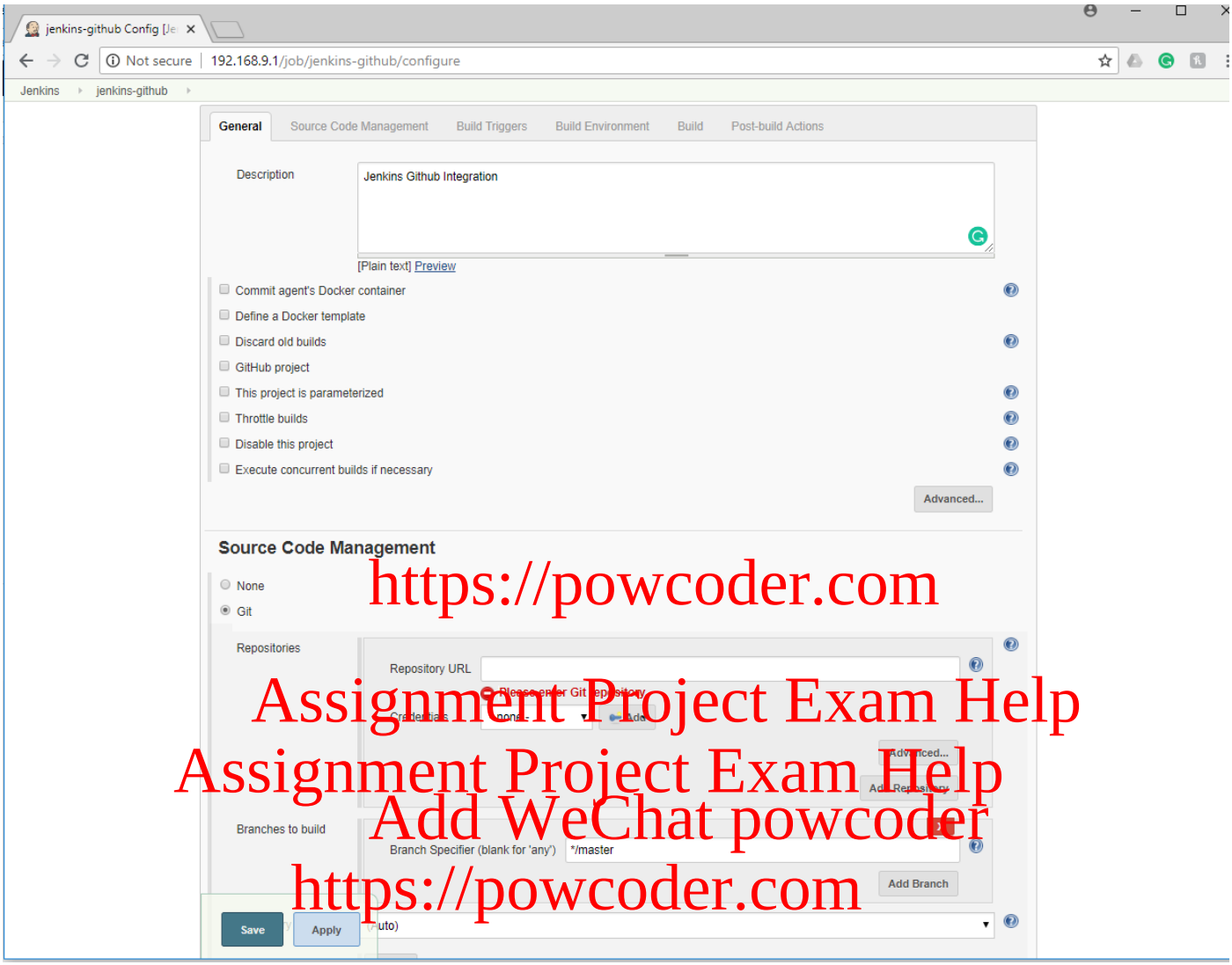
To start using Jenkins, we need to create a new job or item. This can be done by clicking on one of the following links shown below.



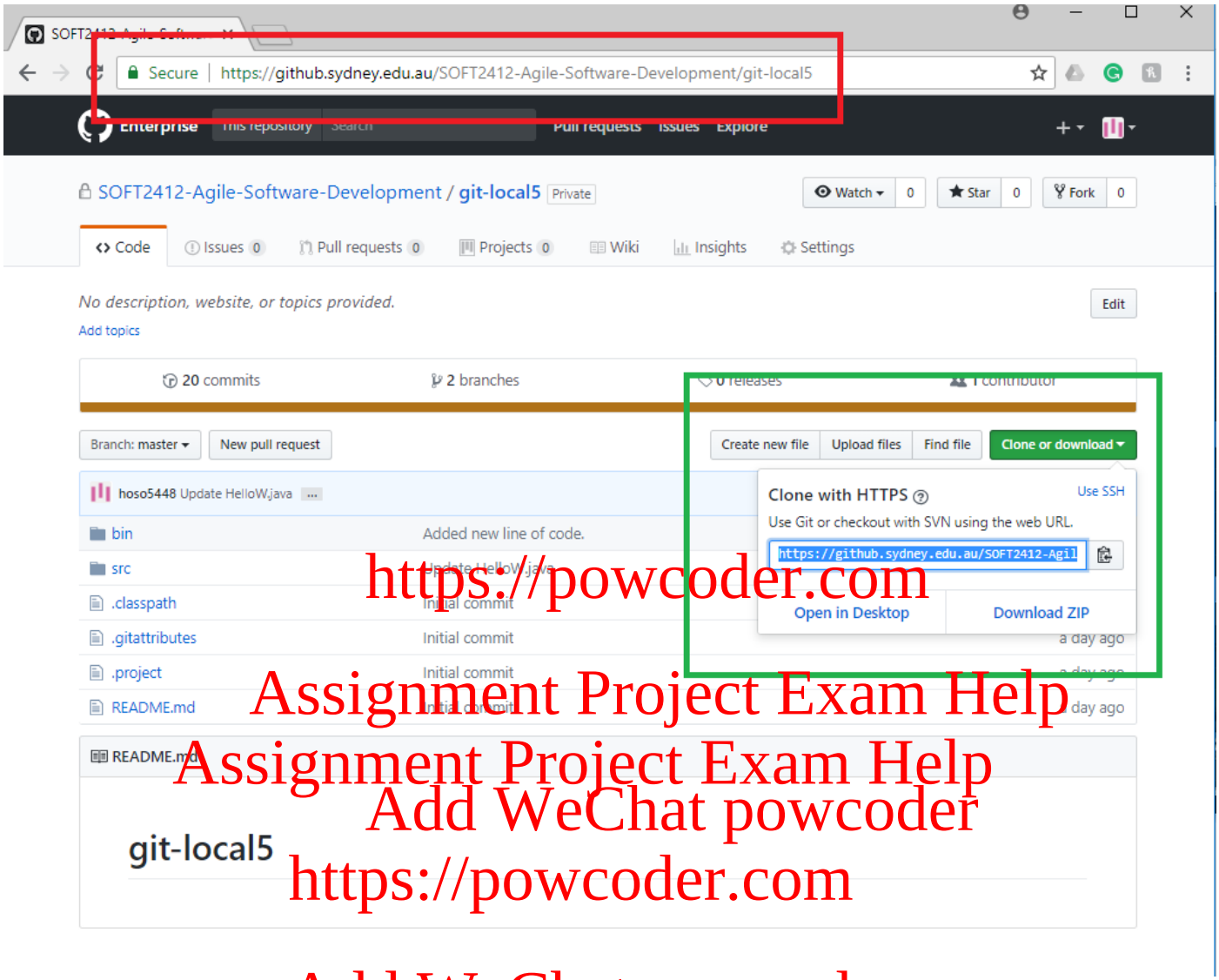
Fill in the item name as *jenkins-github* and select *Freestyle project*. Click *Ok* to proceed.



Under General, you may want to fill in the description to provide more detail about the item that has been created. Although this is optional. What is required is to fill in the Source Code Management. Select Git.



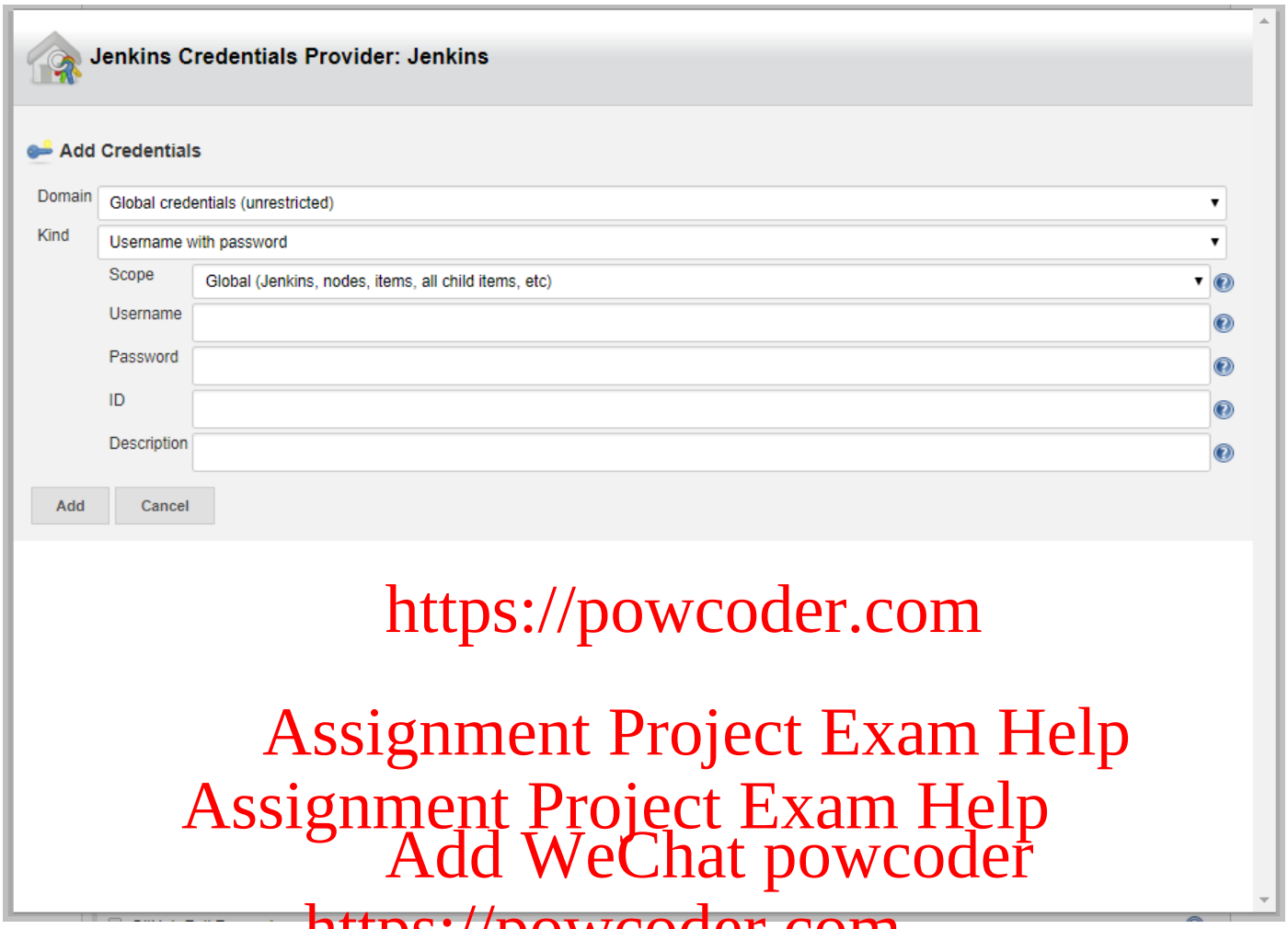
Indicate your repository URL. To get your working git repository URL, you need to switch to github.sydney.edu.au and select the repository that you will work on.



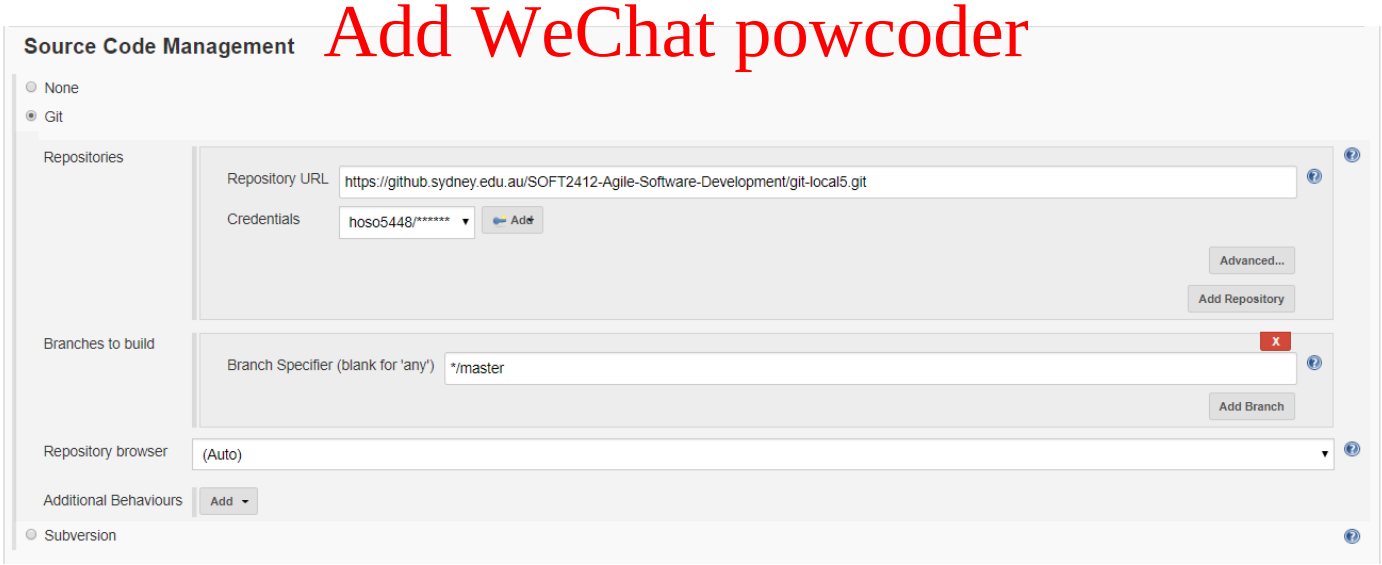
The URL is not what is displayed on the browser (red box). Instead, search for the *Clone or download* button and click on it. Copy the URL provided. This is the URL which references the repository URL (green box). Paste this URL back to the *Jenkins Source Code Management*.

Since we are using github.sydney.edu.au, you need to supply your Unikey credentials to allow Jenkins access to the repository.

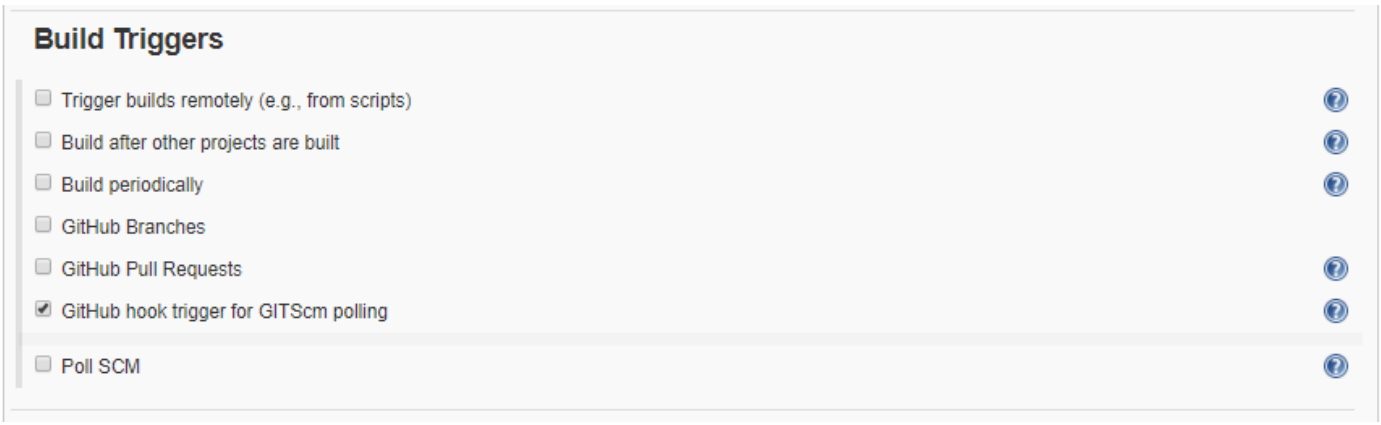
Click on *Add* to add new credentials. Select *Domain* as *Global credentials*. Select *Kind* to be *Username with password*. Select *Scope* to be *Global*. Then indicate your Unikey under *Username* and your Unikey password under **Password**. Next click on *Add*.



You should not be getting any more error messages regarding access to the git repository.

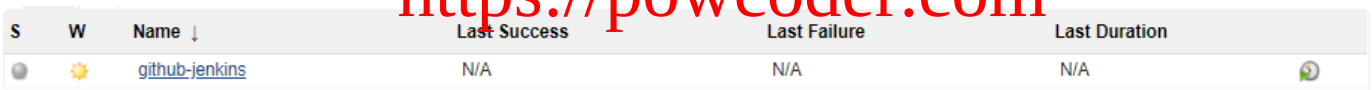


Since we are utilising Jenkins to check for build errors, we select *GitHub hook trigger for GITScm polling*. This feature triggers build in Jenkins when a git event is performed, for example, *push*.



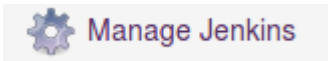
Next click on Save.

If you return to your dashboard, you should be able to see the github-jenkins item added as shown below.

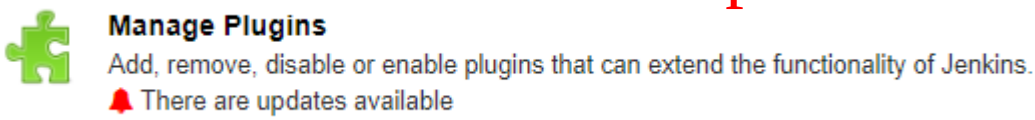


To enable Jenkins to receive triggers from GitHub, you will have to set some plugins in Jenkins.

Click on *Manage Jenkins*.



Click on *Manage Plugins*.



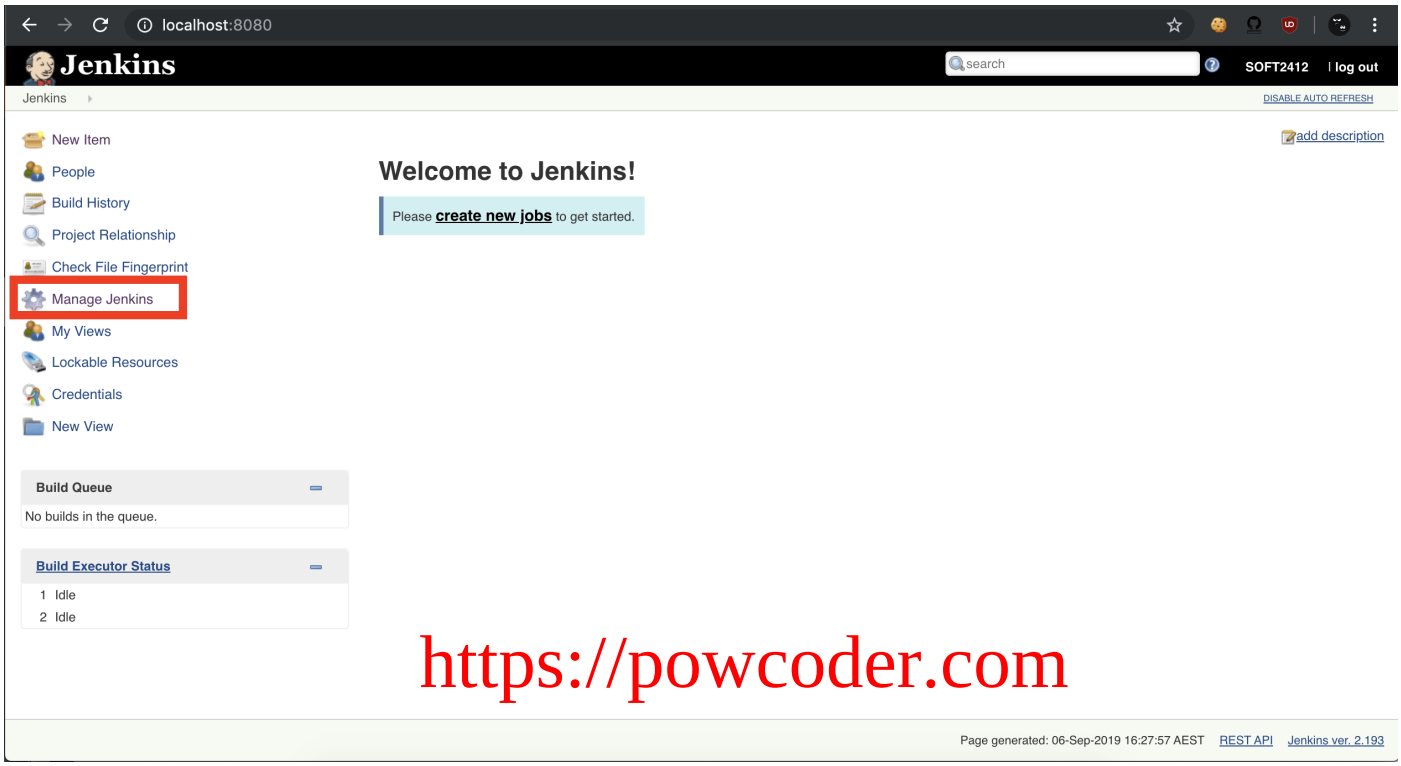
Click on the *Available* tab.

Using the Filter option, key in 'GitHub'. This will display a list of plugins but we are only interested on the *GitHub Integration Plugin.*

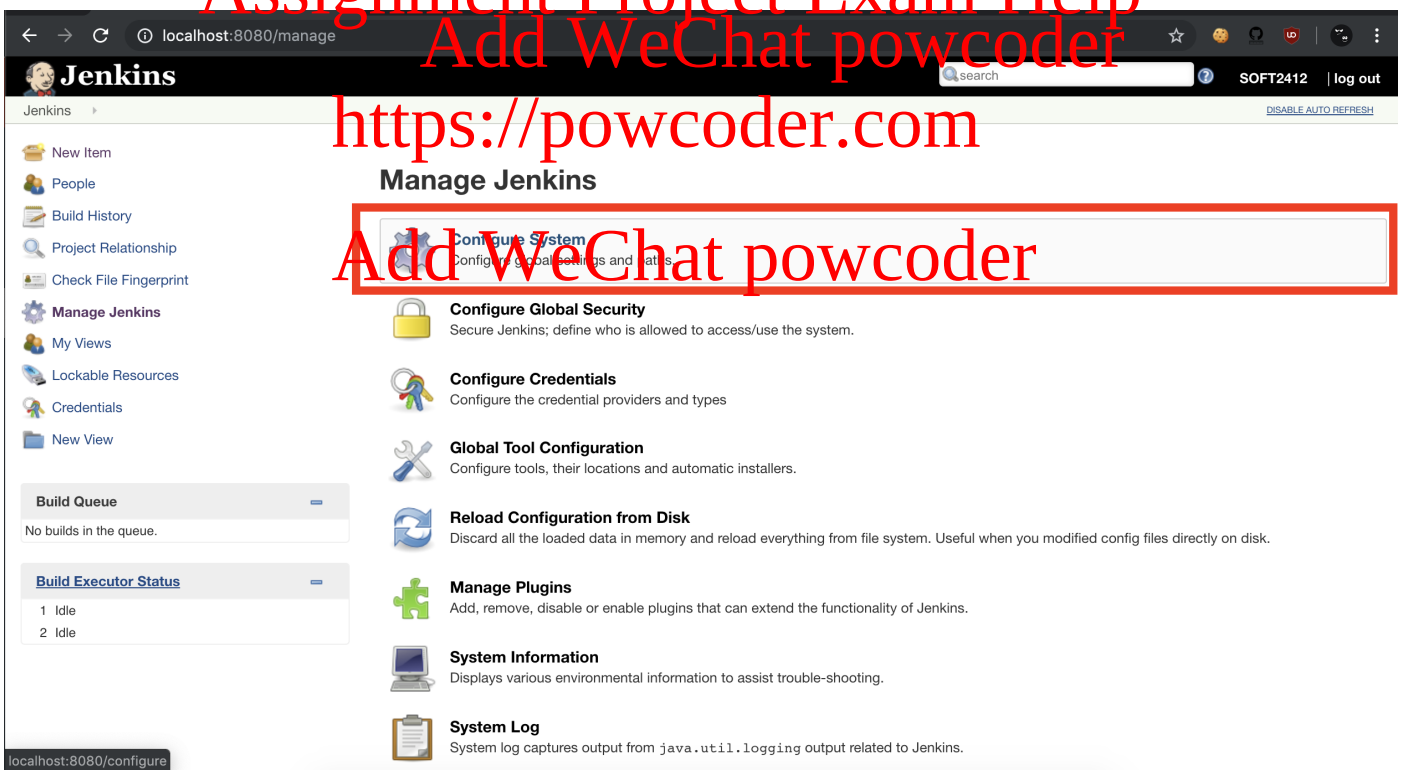
Check this plugin and click on *Download now and install after the restart*. Jenkins will install the plugin and restarts the server. You may be required to log in again.

(THE BELOW IS NOT NEEDED FOR NEW JENKINS VERSIONS, SKIP UNTIL YOU SEE "-------")

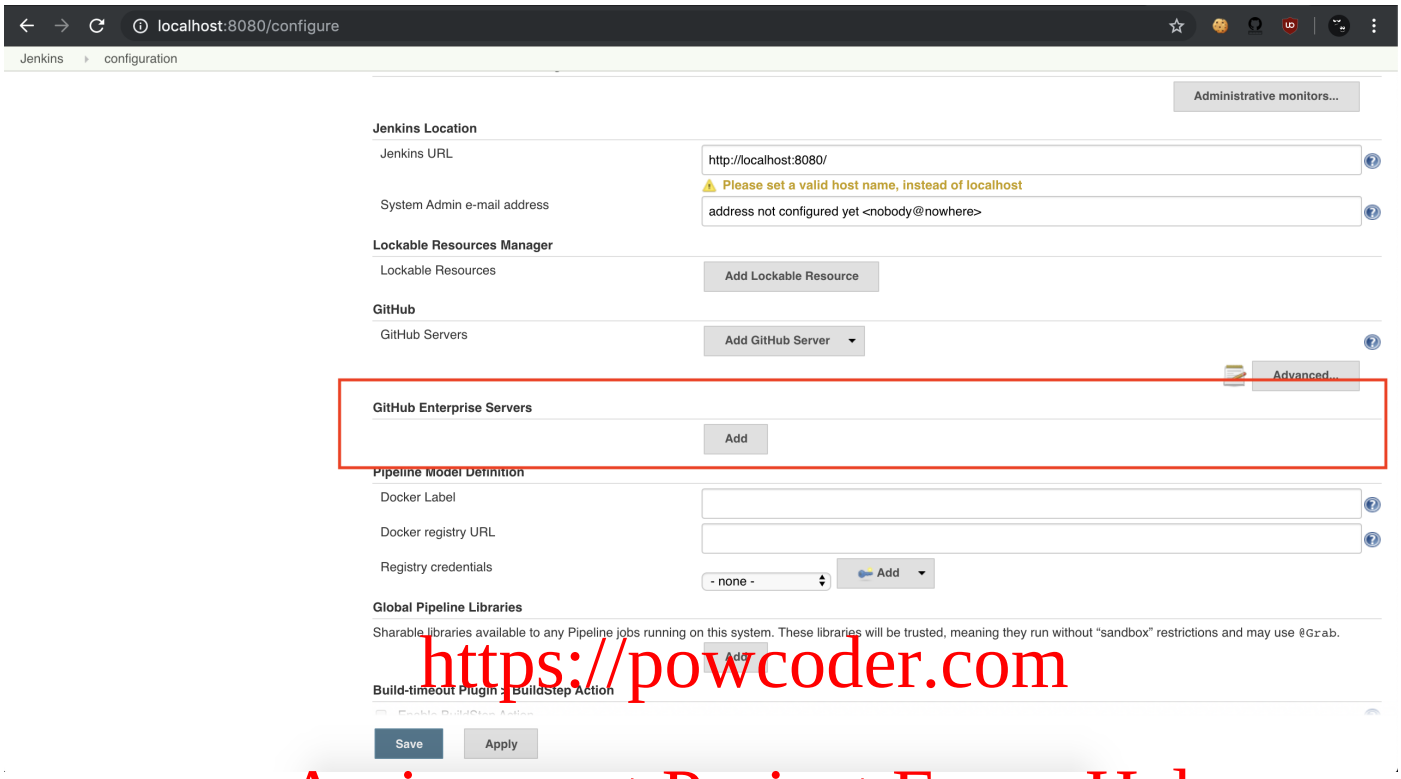
If you are installing jenkins in your own machines: Make sure you add the Github enterprise in jenkins. We need to tell jenkins that a github server that we will be pushing code into is the enterprise. Goto manage jenkins in the homepage:



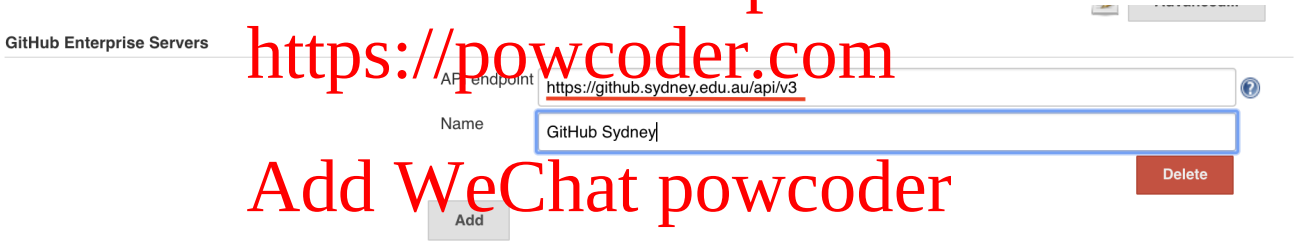
Then click on "Configure jenkins".



Scroll down to "GitHub enterprise servers" and click "Add"



Then add the GitHub Sydney url.



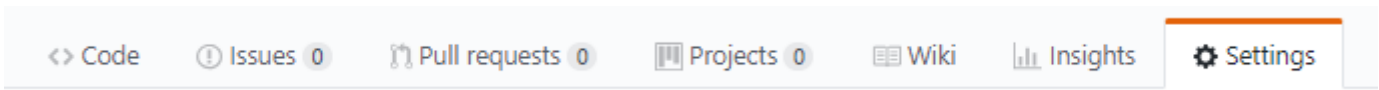
Apply and Save the changes.

(-------)

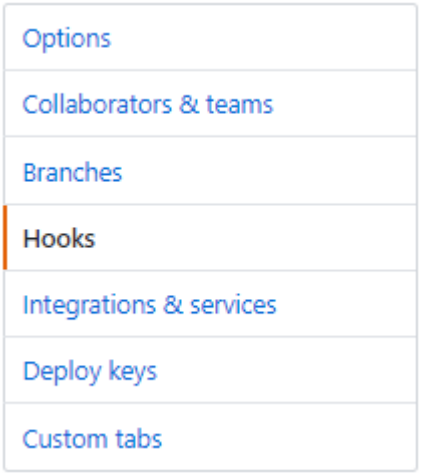
Next in GitHub

While waiting for Jenkins to restart, you will also need to configure your GitHub settings so that it can work with Jenkins.

Use ngrok ip address that you kept note of from above, return to your working GitHub repository.



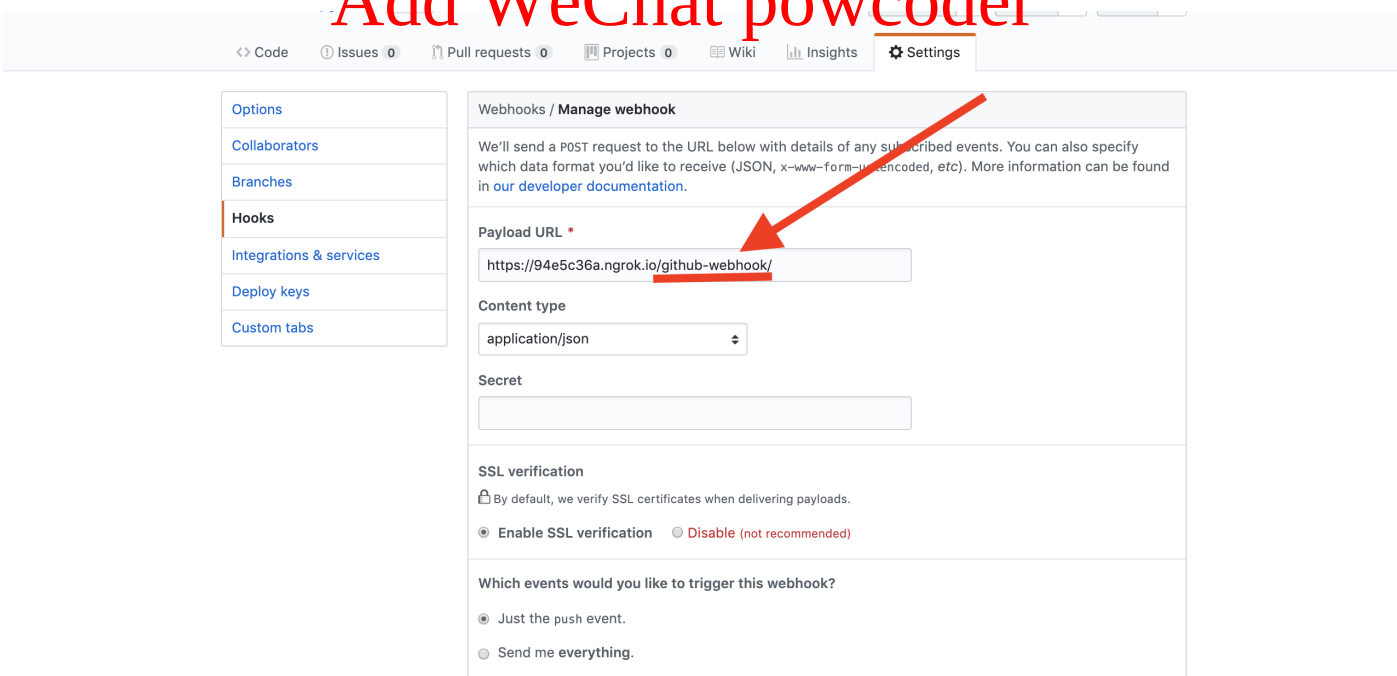
Click on the Settings tab from your working repository.



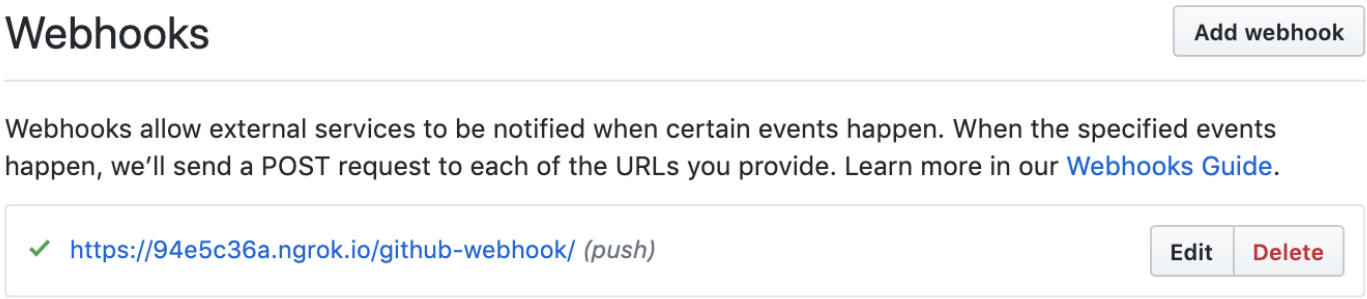
On the left panel, click on Hooks. This will create a service which notifies Jenkins when certain git events happen. Next, click on Add webhook.

You will be presented with a form. With the machine's URL, paste it on the Payload URL. This is the URL which will be triggered by GitHub after a push event is created to the repository.

Update your noted URL to: https://<doman-url>/github-webhook/. In this case, it would look like: https://9387d23f.ngrok.io/github-webhook/ (https://soft2412:soft2412@9387d23f.ngrok.io/) . Add this URL into the “payload URL” input field.



If the URL is correctly configured, there will be a green tick beside it. If there is no green tick, refresh the page several times or restart your virtual machine to restart Jenkins.

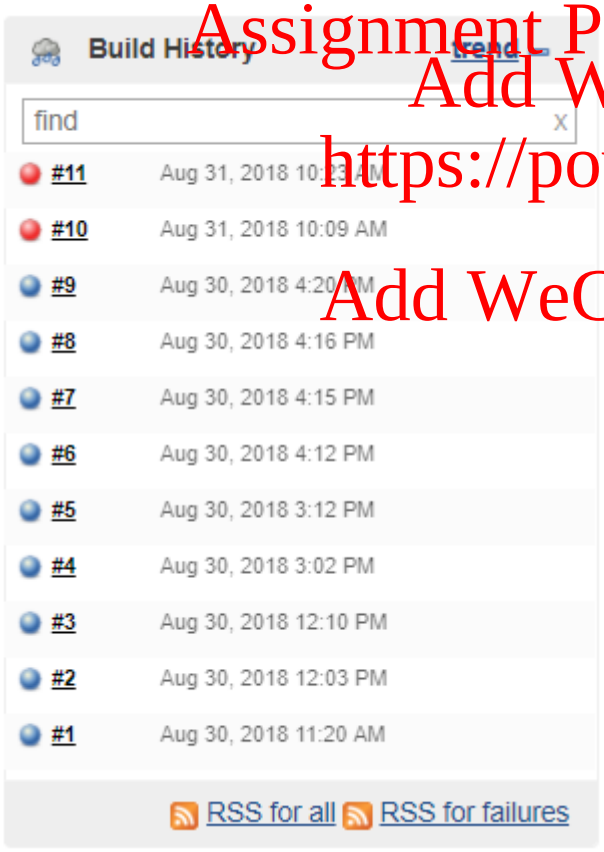


If a green tick is present, it means that the URL is accessible and we can start using GitHub and Jenkins integration.

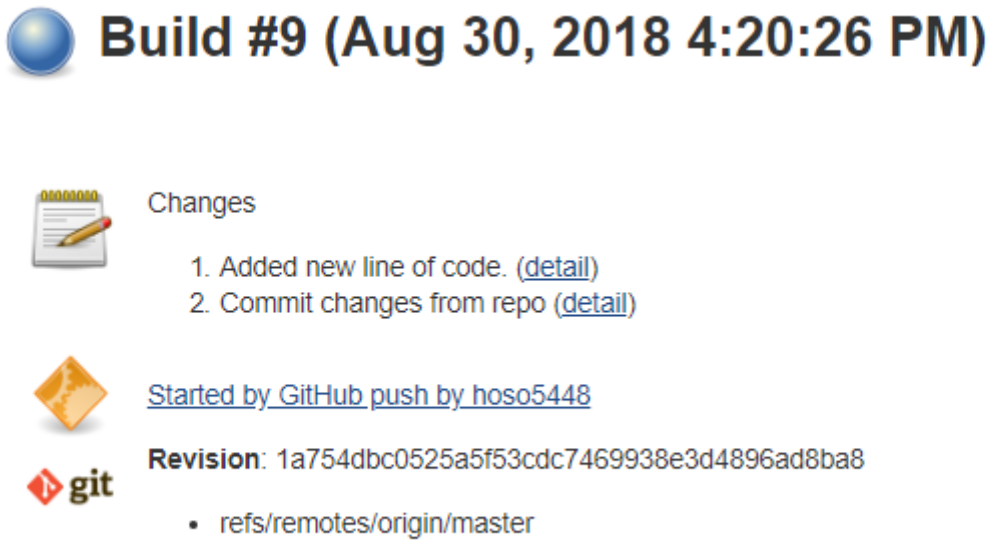
To test if this integration works, we can make a few code changes on git repository and trigger a webhook to Jenkins.

Return to your own code repository. If you do not have any files in your repository, add a new text file. In that file, add the following comment, 'Sending trigger to Jenkins from GitHub'.

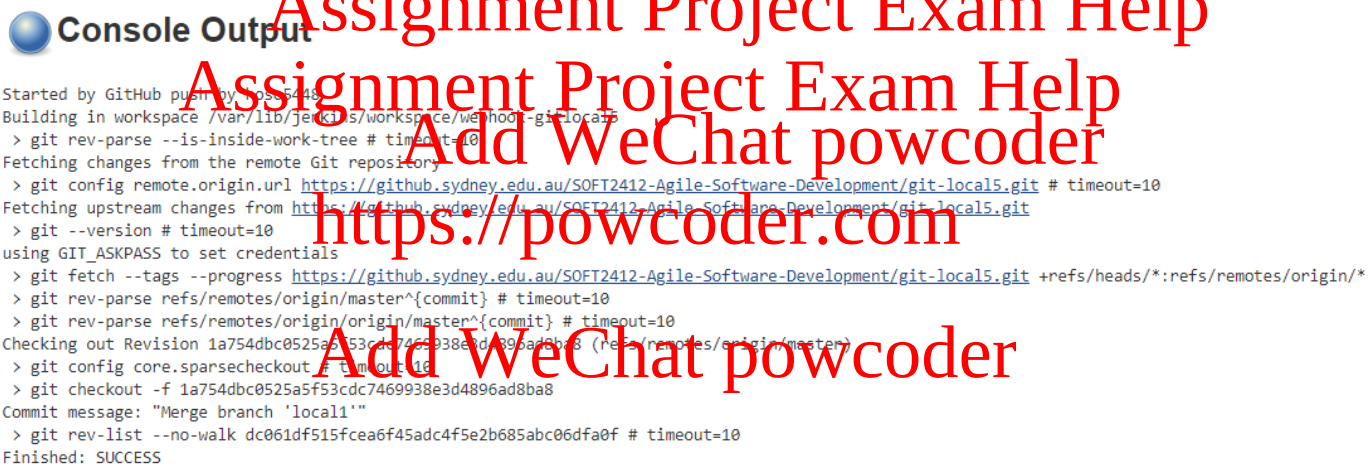
Return to your Jenkins Dashboard and monitor the build history for the newly created Jenkins item. If your build is successful, the circle icon will turn to blue.



Click on the build number to see the build information, like the one shown below.



You can also see the build information from Console Output as shown below.

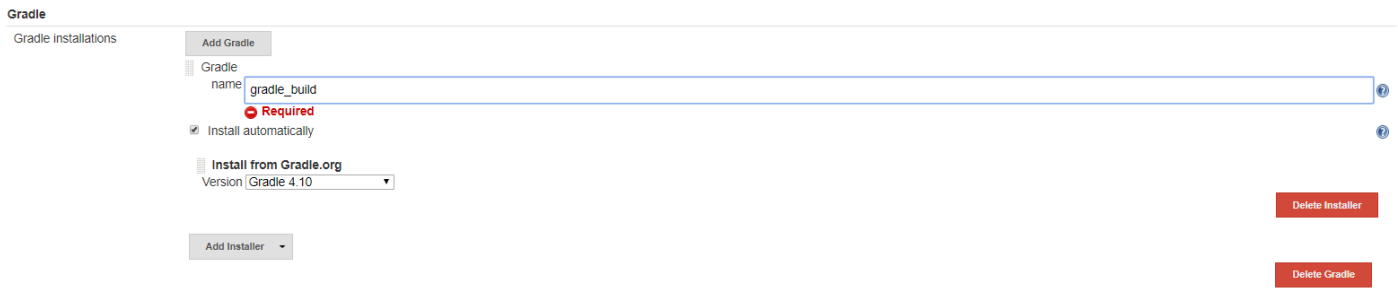


Continuous Integration using Jenkins

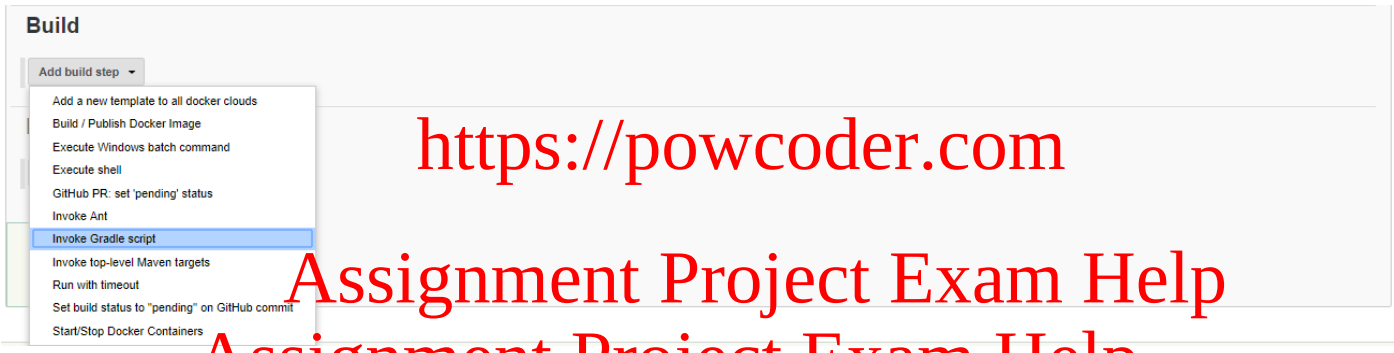
Integrating GitHub -> Jenkins -> Gradle

You can also automate your code build by integrating Gradle to Jenkins after every new code push to GitHub. To enable this, click on *Manage Jenkins* followed by *Global Tool Configuration*.

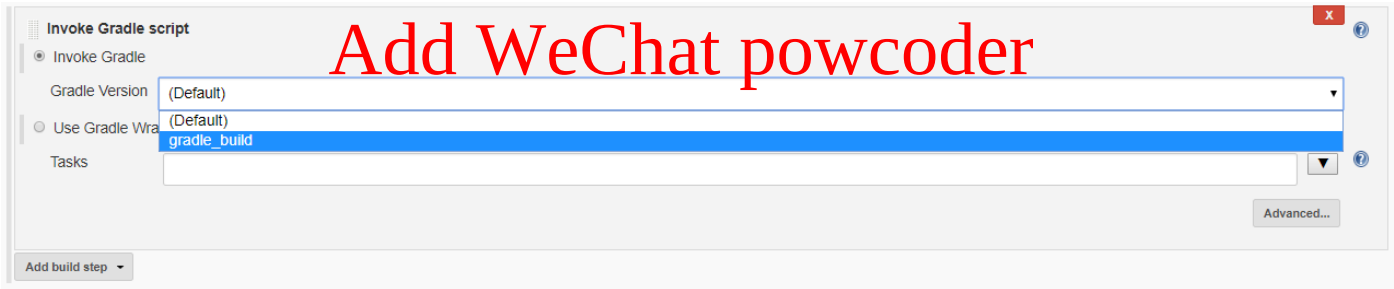
Next, under *Gradle Installations*, click *Add Gradle*. Input Gradle name as *gradle_build* and check *Install automatically*. Click *Save*.



Next, you will need to invoke this gradle_build. To do that, click on *github-jenkins* from the dashboard. Then click on Configure. This will bring up the settings for the *github-jenkins* item. Click on the Build tab. Click on Add build step and select Invoke Gradle script as shown below.



Under Invoke Gradle script, check Invoke Gradle. Select the Gradle version name as the name you have created previously in Gradle installations. In this case, we indicated the name as *gradle_build*. Then click Save.



Add the appropriate Gradle tasks i.e. clean build and test in the input field in the above image.

You can also add Post-build Actions into this process by selecting the appropriate post-build actions

Exercises

In this exercise, you will be tasked with creating a new repository in GitHub and a webhook that will trigger Jenkins and Gradle build. To assist you with this task, a breakdown of the items to do is provided below.

Task 1: (Important to stay on track with the assignment)

- Choose any existing repository you may have in your own github sydney account. Add your team members as collaborators in this repository.
- Push your JUnit test codes from last week's tutorial (the calculator program) to this new repository. The solutions to this must be available now.
- Integrate your GitHub repository with Jenkins and Gradle.
- Create a webhook to trigger Jenkins and Gradle.
- Test that this webhook is trigger correctly.
- Next, each member should include a new function in the code (one example below, find more examples like factorial, modulus, round etc.):

```
public int pow (int a, int b) {

    return Math.pow(a,b);

}
```


- One team member must have jenkins setup, with the webhook added in github.
- Each member should build and commit this new source code to your repository.
- The jenkins member setting up jenkins can see and provide a console report
- Next, each member must create a new test case to test their newly added source code.
- Each commit and build this new source code to your repository, jenkins must be triggered again.
- The same member that setup jenkins, can see all the test cases and builds the program through Jenkins and Gradle and can provide a console report.
- To see changes in the report, one member can try to fail a JUnit test and push the code, simultaneously triggering the webhook. There should be a red dot instead of a blue one in the jenkins build list.

Task 2:

- Try and integrate Jacoco with jenkins, such that every successful build will generate a Jacoco report.
- You may find this helpful: **https://dzone.com/articles/jacoco-jenkins-plugin (https://dzone.com/articles/jacoco-jenkins-plugin)**

Extensions:

Export a .jar file (artefact) from jenkins:
https://wiki.jenkins.io/display/JENKINS/Copy+Artifact+Plugin
(https://wiki.jenkins.io/display/JENKINS/Copy+Artifact+Plugin)

Additional Resources

- https://jenkins.io/solutions/github/