# Chapter 2

## Lab 2.3

### Objective: Install Jenkins

Jenkins is an open source tool that can be used to automate the building and testing of the applications created in the previous exercises.

1. Before Jenkins can be installed, you need to ensure that you have the correct version of Java:

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk
```

Make sure you are using Java 1.8.0 by running **`java -version`**, and, if necessary, change it with the following command:

```
$ sudo update-alternatives --config java
```

While at the command line, modify the RVM script from Lab 2.2, so that it can be accessed by Jenkins. Ensure that it is replaced with the correct path for the RVM installation:

```
$ sudo chmod +x /home/username/.rvm/scripts/rvm
```

2. Jenkins can be added to a source list by using **`wget`**:

```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key |
$ sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins
```

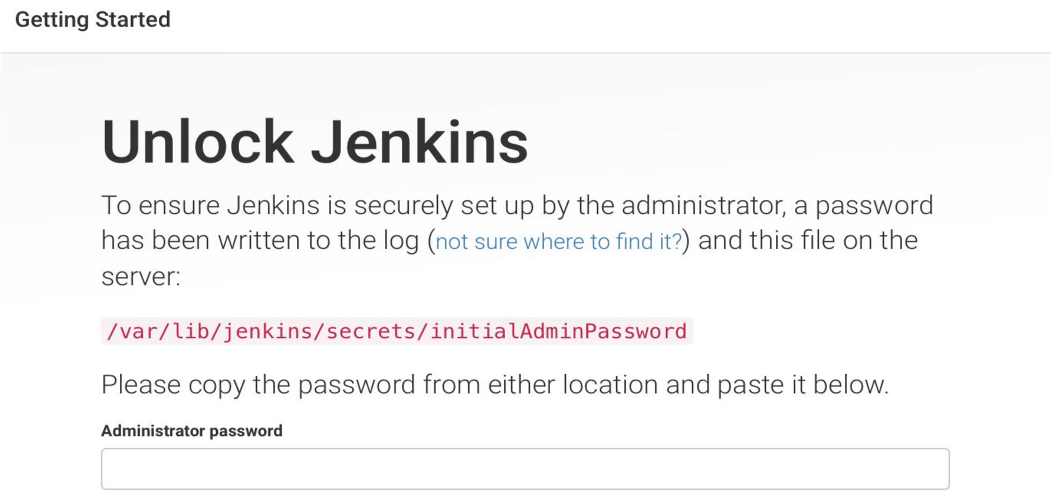
Jenkins is accessible on port 8080 of the machine: **`http://ip.of.the.machine:8080`**. After navigating here, a random password generated by the installer will be prompted. It is accessible from the command line by entering:

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

**Note**: In Vagrant, ports 8080/8081 have been mapped to 18080/18081.

3. Configuring Jenkins can be accomplished by following the onscreen instructions in the Jenkins GUI.

   After entering the password, select “Install suggested plugins” to install the default recommended Jenkins plugins. Additional plugins can be installed at a later date from the Jenkins console, if necessary. The plugins will then install. This may take a few minutes depending on the system and the internet connection.



Getting Started

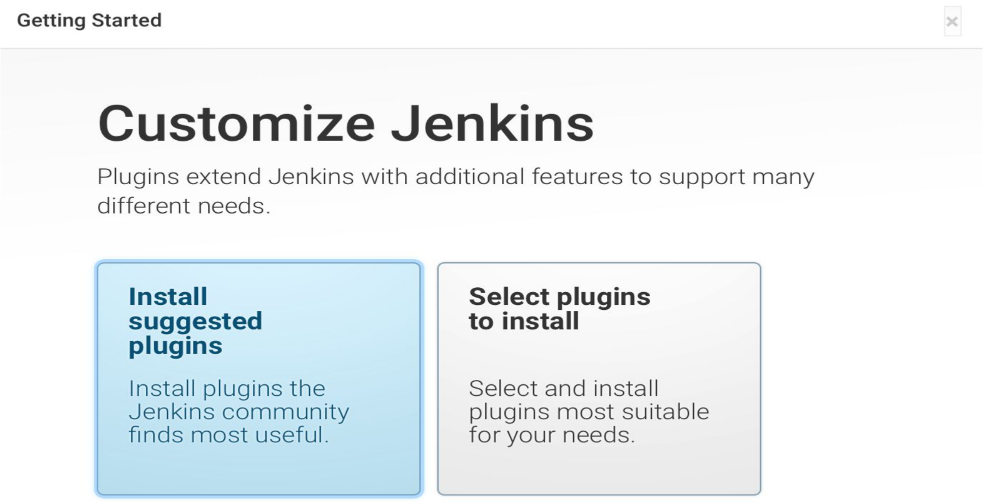
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



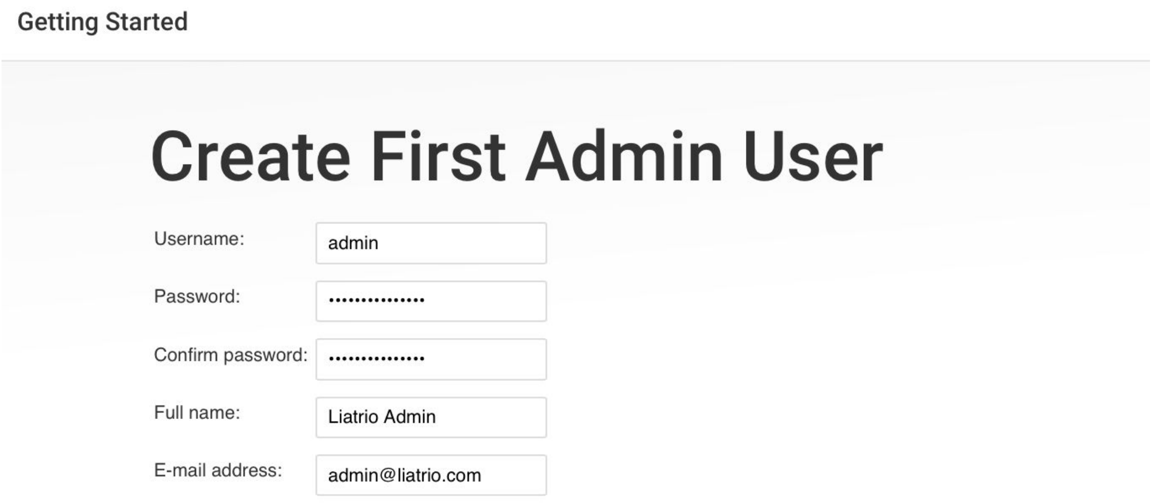
Getting Started

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.

Next, an admin needs to create an Admin User to log into Jenkins. This can be named anything. FIll out the form completely and Jenkins will be ready to use.



Getting Started

Create First Admin User

Username: admin

Password: ···············

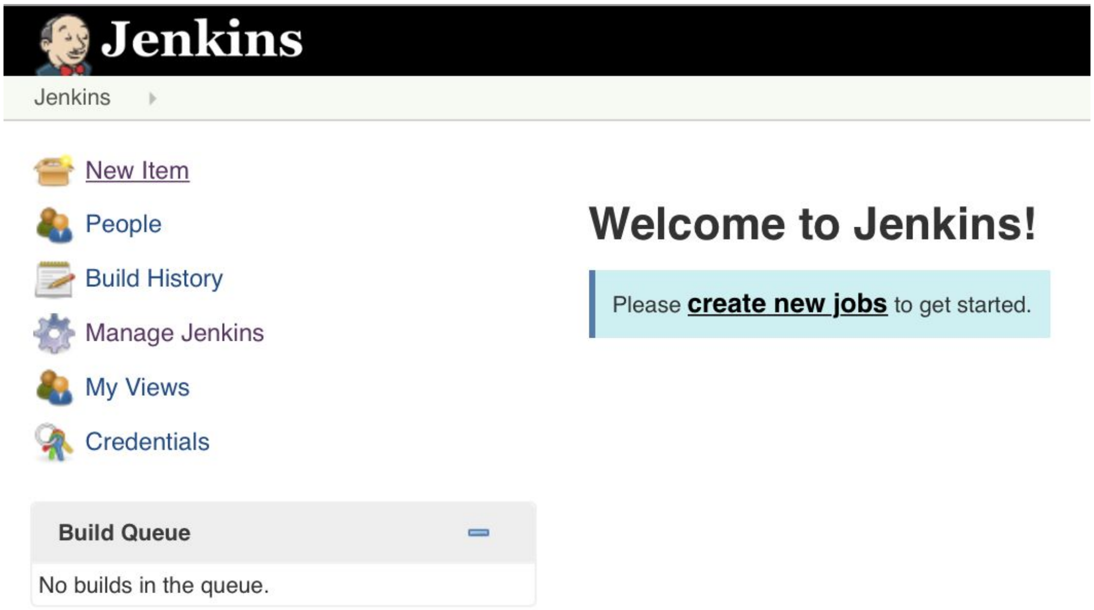
Confirm password: ···············

Full name: Liatrio Admin

E-mail address: admin@liatrio.com

4. To start using Jenkins, you will make a simple job that runs your previously created tests from Lab 2.2.

   Click on the “New Item” or the “create new jobs” link to start configuring your new job.

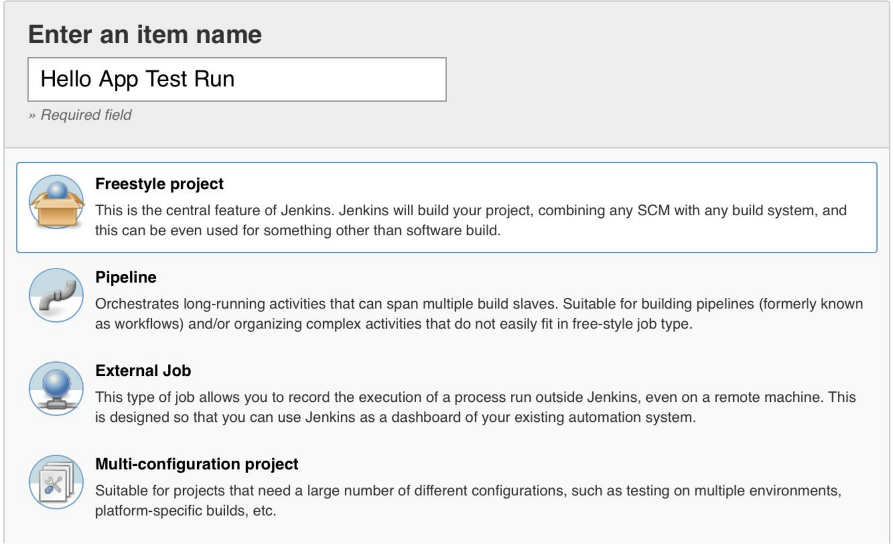


No builds in the queue.

Welcome to Jenkins!

Please **create new jobs** to get started.

Name the new item “Hello App Test Run” and select “Freestyle project” as the project type.



Enter an item name

Hello App Test Run

» Required field

**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

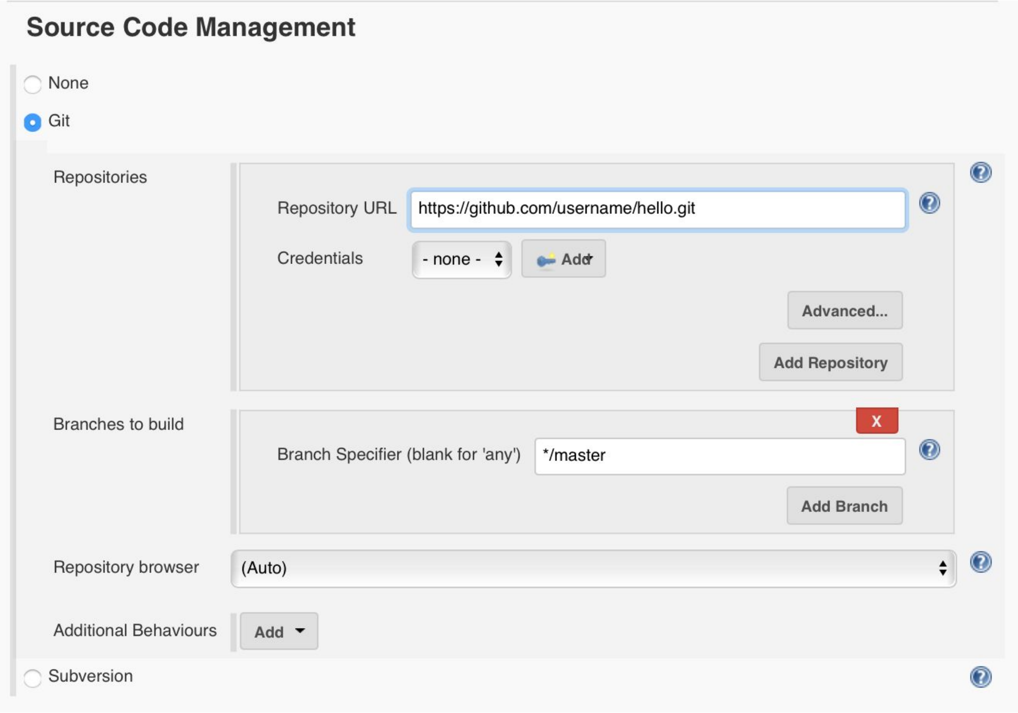
**Pipeline**
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

On the job configuration page, enter a brief description for the job in the "Description" field.

Under "Source Code Management" select "Git" (as the application you made is stored on GitHub), and enter its URL in the "Repository URL" field. If the Git repository is private, a username/password will be needed in the "Credentials" field. Leave the rest of this section in its default configuration.



Source Code Management

○ None
● Git

Repositories
Repository URL: https://github.com/username/hello.git
Credentials: - none -

Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

○ Subversion

To ensure that the test is running periodically, after each push, edit the “Build Triggers” section of the project configuration. Select “Poll SCM”; this setting will make Jenkins regularly check the Git repository for new commits, and trigger a new build when changes are made.

In the “Schedule” field enter:

```
$ H/5 * * * *
```

That will cause Jenkins to check SCM for changes every 5 minutes and build every time it detects a new commit.

Under “Build Environment”, check the box labeled “Delete workspace before build starts”.

This ensures the build executes in the same clean environment each time it is triggered.

Finally, under the “Build” section, add an “Execute shell” build step, and use the following **`bash`** script in the “Command” field. Make sure the commands have the correct paths for both the project’s directory, as well as the RVM installation:

```
$ #!/bin/bash

$ source /home/username/.rvm/scripts/rvm

$ cd /home/username/hello

$ ruby test.rb
```

Save the job and return to the Jenkins console home page.

5. To test this build, simply select “Build Now” from the left hand console menu.

   The build job will begin to execute shortly. Click on the job, and then select “Console Output” when on the job's page. The console output should match the test output from Lab 2.2.

6. To ensure that the build triggers correctly after a commit, and that it both succeeds and fails appropriately, change one of the assertions in **`test.rb`** to a value that will not pass. An easy example would be changing the assertion value to **`foo`**.

   Add, commit, and push the updated **`test.rb`** file to GitHub:

   ```
   $ git commit -am "changed test to fail"
   $ git push origin master
   ```

   After successfully pushing, navigate back to the Jenkins console, and either wait for the job to trigger automatically (up to 5 minutes), or trigger a new build manually.

The job should fail, indicating the push was successful and the test ran correctly.

**Jenkins Tutorial:**

In this chapter, we covered basics of installing and using Jenkins in the workflow example. It is recommended that you check this Jenkins Tutorial to better understand the Jenkins setup, configuration, and operations.

**Additional Exercises:**

The Jenkins project offers a very helpful tutorial on using the Pipeline plugin. It is strongly recommended to work on the examples provided in this tutorial (Pipeline Plugin Tutorial), before you move on to the next section.