Configuring Jenkins

# Download and Run Jenkins

1. Download Jenkins Java Web Archive from jenkins-ci.org.
   1. For Linux users, wget <http://mirrors.jenkins-ci.org/war/latest/jenkins.war>
2. To start Jenkins manually, execute the WAR file.
   1. Jenkins defaults to port 8080, so override the port number to prevent collisions with other applications.

java -jar jenkins.war --httpPort=abcd

* 1. Jenkins will be accessible from http://localhost:abcd/

1. To deploy Jenkins to Tomcat, put the .WAR file into the /webapps directory.
   1. Jenkins will be accessible from http://localhost:tomcatport/jenkins

# Configure Git Plugin

1. On the Jenkins homepage, select Manage Jenkins.



1. Select Manage Plugins.



1. Click the Available tab, and search for Git plugin.



1. Click Download Now and Install After Restart



1. Restart Jenkins. Force shutdown in the command prompt and restart.
   1. If Jenkins is deployed on Tomcat, enter <http://localhost:abcd/jenkins/restart>

# Configure Jenkins

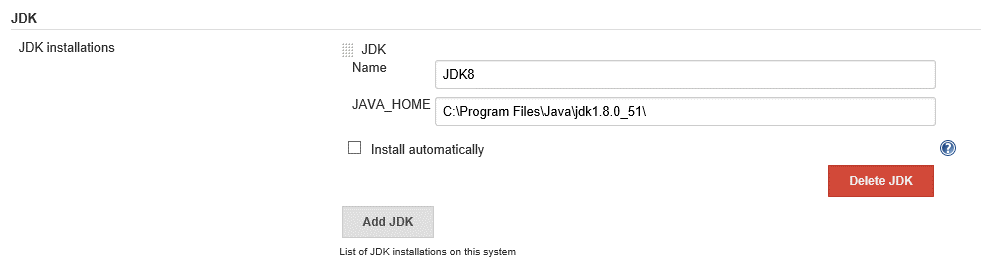
1. On the Jenkins homepage, select Manage Jenkins.



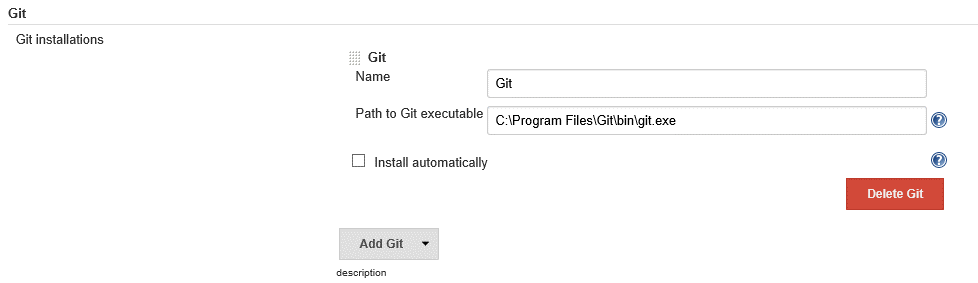
1. Select Global Tool Configuration



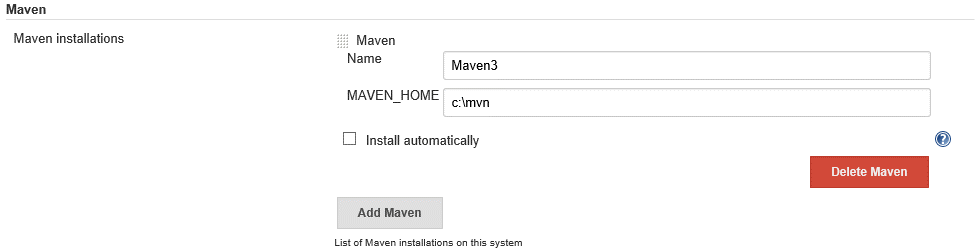
1. Under JDK, enter the path to your JAVA\_HOME.



1. Under Git, show Jenkins how to execute Git commands.



1. Under Maven, enter in the path to your MAVEN\_HOME.



1. Under Git Plugin, -in Configure System- configure your user and email information to sign your commits.



1. Save your changes.

# Secure Jenkins

1. On the Jenkins homepage, select Manage Jenkins.



1. Select Configure Global Security.

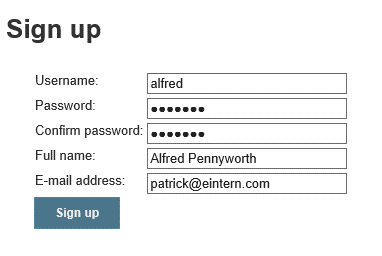


1. Check Enable Security.
2. Under Access Control:
   1. Select Jenkins’ own user database under Security Realm.
   2. Under Authorization, select Matrix-based security
   3. Enter a username in the “User/group to add” input box. Click “Add” button.
   4. Give your user access to all of the privileges in the access control list (ACL).

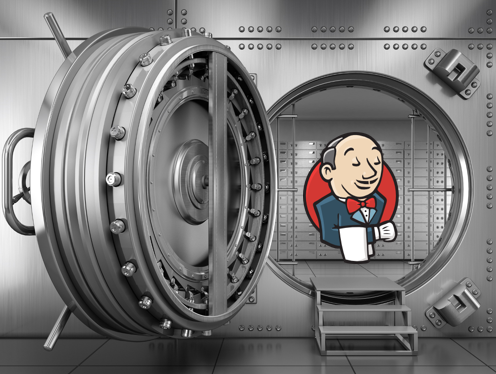
Jenkins security example:



1. Click “Sign up” in the top right corner. Sign up using the username you created in the ACL.

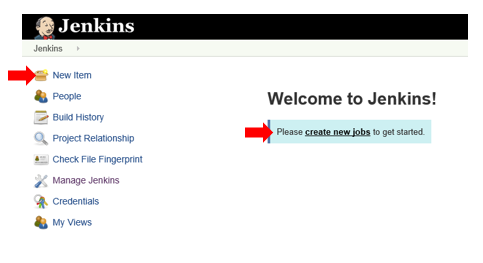


1. You can now require a login to protect your Jenkins server from unwanted access.

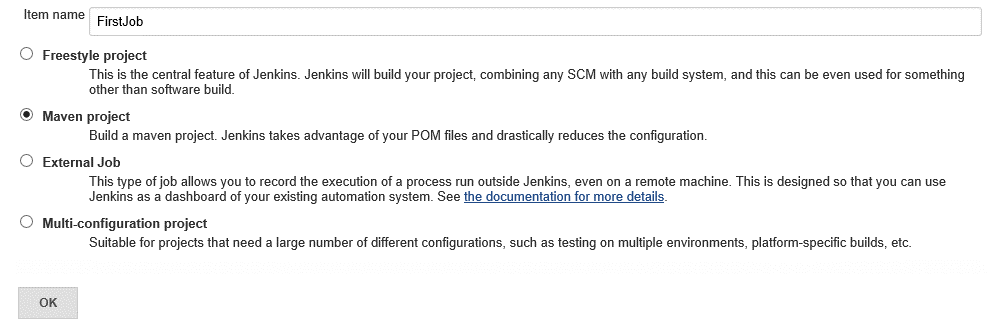


# Create a Jenkins Job

1. On the Jenkins homepage, select New Item. For your first job, you can also click create new jobs.



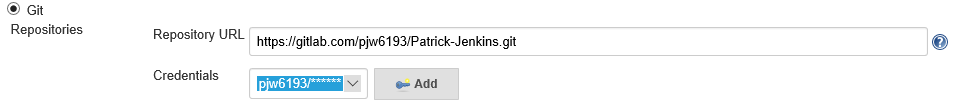
1. Enter a unique name for your job. Select Maven project. Click OK.



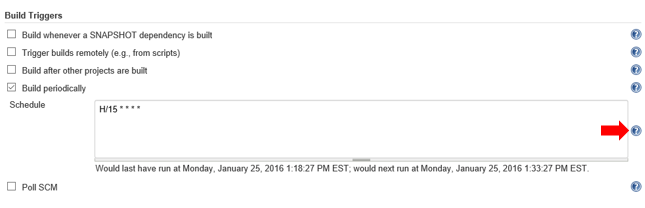
1. Under Source Code Management, select Git. Enter in your repository URL.



1. By Credentials, click Add. Register your username and password for your Gitlab account.
2. Select your credentials.



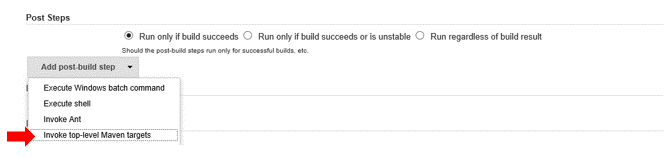
1. Under Build Triggers, select Build Periodically. Enter H/15 \* \* \* \* in the Schedule box. Click the Information icon to see more about cron syntax.



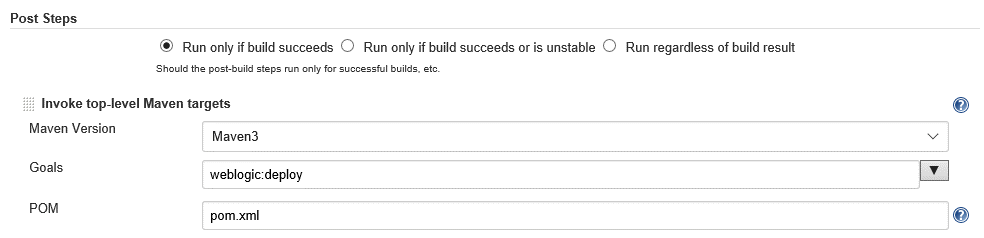
1. Under Build, select the Maven goal for your build. For example, install.



1. OPTIONAL. Add a Post-Build Step to execute the weblogic-maven-plugin or tomcat-maven-plugin deploy command. This will deploy your application to your server.
   1. Click Add post-build step. Select Invoke top-level Maven targets.



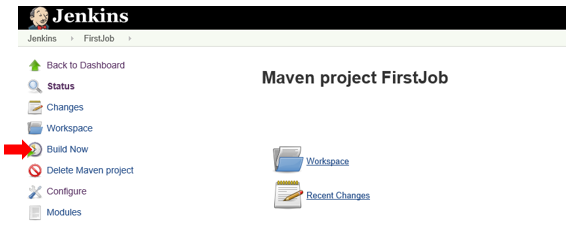
* 1. Select your configured Maven installation. Enter the goal you wish to use. (Note: click Advanced and enter “pom.xml” to tell Jenkins to execute the goal on this POM)



1. Click Save.

# Running a Job

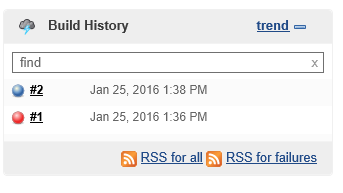
1. Test your job by clicking Build Now.



1. Under Build History, click your build number then click Console Output to view the log during the build. You can debug build issues using this output.



1. Under Build History, failed builds will be colored red; successful builds are blue.



1. On the Jenkins homepage, you can see the status of the last build for each of your jobs. You can also get the “Weather report,” which shows an image depicting the status of the last 5 builds.



1. Jenkins will also continue to run jobs automatically based on your build triggers.

