**Continuous integration with Jenkins**

**Setup a continuous integration process to build and test a Java web app with Jenkins and JUnit**

TABLE OF CONTENTS

* [Introduction to CI/CD](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=1#toc-1)
* [Download and install Jenkins](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=1#toc-2)
* [Setup Jenkins CI for a Java web app](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=1#toc-3)
* [CI reports with Jenkins and JUnit](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=2#toc-4)
* [Using static code analysis tools with Jenkins](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=2#toc-5)

[PREVIOUS](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=2) [1](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html) [2](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=2) [3](https://www.javaworld.com/article/3123117/open-source-java-projects-jenkins-with-docker-part-1.html?page=3)

From the Jenkins homepage, click on **Manage Jenkins**, then **Manage Plugins**. Choose the Available tab and search for the following plugins, checking each one as you find it:

[[jw jenkins fig11 copy](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig11-copy-100820974-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig11-copy-100820974-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 11.&nbsp;Installing the Static Analysis Collector plugin</p> )

Steven Haines

Figure 11. Installing the Static Analysis Collector plugin

[[](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig12-copy-100820975-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig12-copy-100820975-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 12.&nbsp;Installing the Checkstyle plugin</p> )

Steven Haines

Figure 12. Installing the Checkstyle plugin

[[](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig13-copy-100820976-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig13-copy-100820976-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 13.&nbsp;Installing the FindBugs plugin</p> )

Steven Haines

Figure 13. Installing the FindBugs plugin

[[](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig14-copy-100820977-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig14-copy-100820977-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 14.&nbsp;Installing the PMD plugin</p> )

Steven Haines

Figure 14. Installing the PMD plugin

[[](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig15-copy-100820978-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig15-copy-100820978-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 15.&nbsp;Installing the Cobertura plugin</p> )

Steven Haines

Figure 15. Installing the Cobertura plugin

**Step 3. Update the project build config**

Next you'll update your project build configuration. Navigate to your project page and click the **Configure** button. Scroll down to the Build section and update your Maven goals to the following:

clean install checkstyle:checkstyle findbugs:findbugs pmd:pmd pmd:cpd cobertura:cobertura -**Dcobertura**.report.format=xml

**Step 4. Add your post-build actions**

Finally, scroll to the post-build actions section and do the following:

* Publish Checkstyle analysis results: Enter the following into the Checkstyle results text field, so that the Checkstyle plugin can find your test results: \*\*/checkstyle-result.xml
* Publish FindBugs analysis results: Enter the following into the FindBugs results text field, so that the FindBugs plugin can find your test results: \*\*/findbugsXml.xml
* Publish PMD analysis results: Enter the following into the PMD results text field, so that the PMD plugin can find your test results: \*\*/pmd.xml
* Publish Cobertura Coverage Report: Enter the following into the Cobertura.xml report pattern text field, so that the Cobertura plugin can find your test results: \*\*/target/site/cobertura/coverage.xml
* Publish the combined analysis results.

Save your project and click **Build Now**. As shown in Figure 16, you should now see that reports are being collected and trends aggregated for Checkstyle, FindBugs, PMD, and Cobertura.

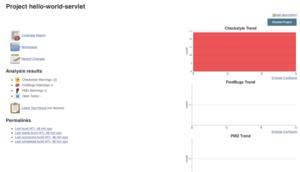
[[](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig16-copy-100820979-orig.jpg)](https://images.idgesg.net/images/article/2019/12/jw-jenkins-fig16-copy-100820979-orig.jpg" \o "<div class='credit'>Steven Haines</div> <p>Figure 16.&nbsp;New project homepage</p> )

Figure 16. New project homepage

You can click through the various reports and review the information to learn more about the health of your application.

**Conclusion**

In this tutorial you've taken your first steps with Jenkins CI. To start, you integrated Jenkins with GitHub and learned how to poll for source code changes, checkout a project, build the project from a Maven command, and publish the project's unit test results to the console. You then added a set of static code analysis tools to your project's Maven pom.xml file, installed the five Jenkins plugins, and published them to the Jenkins build. Finally, you updated your Maven build goals, and added post-build actions to generate a set of static code analysis reports.

Setting up and testing an example project in Jenkins is a good way to familiarize yourself with how the Jenkins directory structure is organized. The more you understand how Jenkins works internally, the better, especially as you start to do more complex operations.