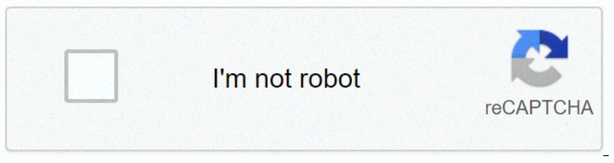

Continue

Jenkins checkstyle report

From the Jenkins home page, click On Manage Jenkins, and then manage plugins. Select the Available tab and search for the following plugins, check each one as you find it: Steven Haines Figure 11. Installing the Static Analysis Collector plugin Steven Haines Figure 12. Installing the Checkstyle plugin Steven Haines Figure 13. Installing the FindBugs plugin Steven Haines Figure 14. Installing the PMD plugin Provided by author for JavaWorld. Figure 15. Installing the Cobertura plugin Step 3. Update the project to build configNext you will 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 after the following: clean install checkstyle: checkstyle findbugs: findbugs pmd:pmd pmd:cpd cobertura:cobertura -Dcobertura.report.format=xml Step 4. Add your post-build actionsFinally, browse to the post-build actions section and do the following: Publish Checkstyle analysis results: Enter the following in 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 in the FindBugs results text field, so the FindBugs plugin can find your test results: **/findbugsXml.xml Publish PMD analysis results: Enter the following in the PMD results so that the PMD plugin can find your test results: **/pmd.xml Publish Cobertura Coverage Report: Enter the following in the Cobertura.xml report pattern text field, so the Cobertura plugin can find your test results: **/target/website/cobertura/coverage.xml Publish the combined analysis results. Save your project and click Build now. As shown in Figure 16, you should now see reports being collected and trends merged for Checkstyle, FindBugs, PMD, and Cobertura. Steven Haines 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. ConclusionIn this tutorial has taken you your first steps with Jenkins CI. For starters, you integrated Jenkins with GitHub and learned how to vote for source code changes, check a project, build the project from a Maven assignment, and publish the project's unit test results to the console. You then installed a set of static code analysis tools with your project's Maven pom.xml file, the five Jenkins plugins, and published them to the Jenkins build. Finally, you've updated your Maven build goals, adding post-build actions to generate a set of static code analysis reports. Setting up and testing a sample project in Jenkins is a great way to requander yourself with how the Jenkins directory structure is organized. The more you understand how Jenkins works internally, the better, especially if you start doing more complex operations. This story, Continuous integration with Jenkins is published by JavaWorld. Copyright © 2019 IDG Communication, Communications, v1.77 v1.76 v1.75 v1.74 v1.73 v1.72 v1.71 v1.70 v1.69 v1.68 v1.6 v1.66 v1.65 v1.64 v1.63 v1.62 v1.61 v1.60 v1.59 v1.58 v1.57 v1 .56 v1.55 v1.54 v1.53 v1.52 v1.51 v1.50 v1.49 v1.48 v1.47 v1.46 v1.45 v1.44 v1.43 v1.42 v1.41 v1.40 v1.39 v1.38 Last time we set up a work-check style check locally. But for this test to be truly significant and to benefit the team, it is better to integrate this action into the CI/CD of your development process. In this article we will look at setting up check style control using the example of the popular Jenkins automation server. I assume you've already configured Jenkins and you already know how to run the check-style check locally. Let's create a base project with a project that builds in maven. Add the script of this article to the jenkins pipeline. Plugin installation In order to display the check results, we must install the Alerts Next Generation Plugin Management Jenkins > Plugin Manager > Available Enter name of plugin - Alerts Next Generation and install it Configure Jenkins to store checkstyle results Let's configure JDK and Maven to build the project. Go to Manage Jenkins > Global Tool Configuration In the Maven section, click Add Maven. Remember text you entered in the name field. Repeat the same actions for jdk section. Now time to write pipeline script pipeline { agent any tools { maven 'mvn3' jdk 'jdk8202' } stages { stage('Checkout') { steps { git credentialId: 'github-ssh', url: 'git@github.com:username/checkstyle.git' } stage('Build') { parallel stage ('Build') { parallel stage('Maven build') { steps { sh mvn clean install } stage('Checkstyle') { steps { sh mvn checkstyle:check recordIssues(tools: [checkStyle(reportEncoding: 'UTF-8')]) } } git credentialId: 'github-ssh', url: 'git@github.com:username/checkstyle.git' sh mvn checkstyle:check recordIssues(tools: [checkStyle(reportEncoding: 'UTF-8')]) parallel query allows step to be run in parallel recordIs s (tools: [checkStyle(reportEncoding: 'UTF-8')]) recordIssues(tools: [checkStyle(reportEncoding: 'UTF-8')]) login our chesckstyle issues to sign in. Now perform a job and see result. Clicking on Checkstyle Warning link in work details, we see detailed information about checkstyle checks, their level as well as the history of the last multiple builds. I have the following problem – I run a check style ant task when building with Jenkins and the respective report is produced, but the thing is that when the build finishes I get a message that there is an error during analysis because the error report is not found. I verified that I set the path to the report to be published after the real thing (when I change it to something slightly different, I get a message that xxx doesn't exist, but the path the previous version exists). Any idea what could be wrong? What format does Jenkins expect to publish the check-style report? I use the following build.xml<taskdef classpath=libs/checkstyle-5.6-all.jar checkstyle-5.6-all.jar /> <target name=checkstyle> <checkstyle config=checkstyle.xml failonviolation=false> <fileset dir=src includes=**/*.java></fileset> <formatter type=plain></formatter> <formatter type=xml></formatter> <formatter tofile=checkstyle-result.xml type=xml></formatter> </checkstyle> <style style=checkstyle-noframes.xsl in=checkstyle-result.xml out=checkstyle-result.html></style></target>