Continuous Integration 101: How to Run JMeter With Jenkins

Why Continuous Integration is Crucial

When a developer is working on a new feature or fixing a defect, he needs to ensure that these changes won’t have a negative impact on other software parts which were previously working perfectly. To make the process as efficient as possible, it should be unattended.

Continuous Integration (CI) systems facilitate all of this by orchestrating software build, testing, and deployment processes. In short, CI systems:

1. Monitor source code repositories
2. Run build whenever code changes are detected
3. Run tests against compiled software (unit, acceptance, automated, performance, integration etc.)
4. Generate artifacts (binaries, documentation, installation packages, etc.)

It’s no wonder that CI systems are being widely adopted by software development companies. In fact, Continuous Integration is now a vital part of the modern software development life cycle.

In fact, by implementing a Continuous Integration system in your workflow, you’ve already checked off the first three items of [Joel Spolsky’s 12 Steps to Better Code](http://www.joelonsoftware.com/articles/fog0000000043.html). For the uninitiated, these are:

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?

Why Performance Testing is Important in Your CI Process

Adding performance tests to your CI scope will help you reduce the risk of performance degradations whenever you add a new feature or fix a bug on your product.

For example: if your latest commit inadvertently causes response times to go above an acceptable threshold or SLA, your CI system will automatically raise a ‘red flag’ and mark the build as ‘failed’. This will make it very easy for you to identify the cause (and the responsible person! :) behind the problem.

What’s more; it’s much better to start running performance tests in the early software development stages. If you detect a major error in the pre-release phase, it will be much harder to identify and fix the issue.

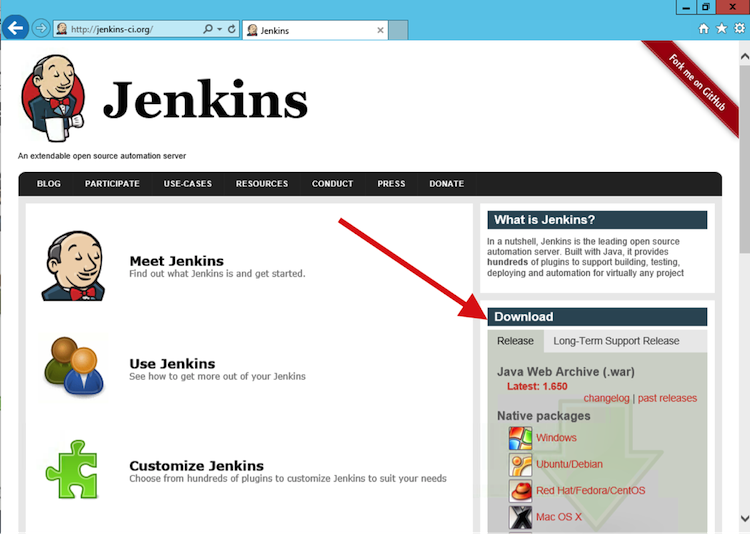
Today, I’ll be showing you how to add your performance testing with [JMeter](http://jmeter.apache.org/" \t "_blank)to a [Jenkins Continuous Integration](https://jenkins-ci.org/) build. I chose these tools for today’s demonstration because they are leading tools for performance testing and Continuous Integration. The fact that they are open source and extremely popular means that they have great documentation and large, active communities where you can ask questions.

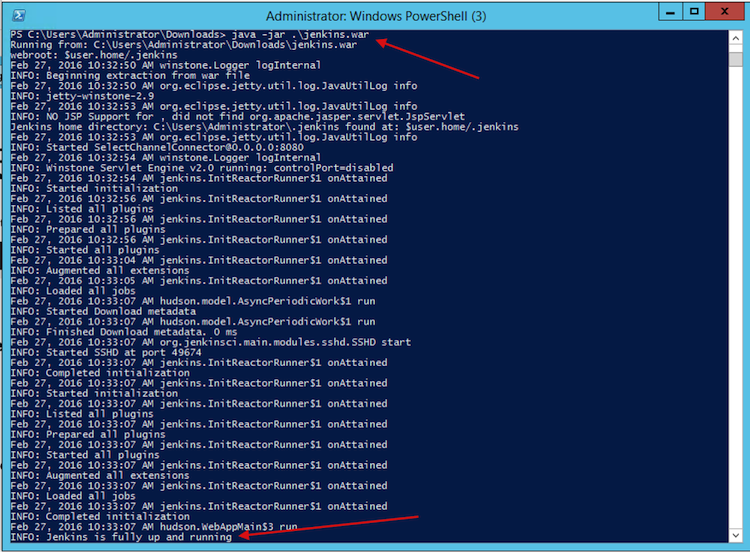
Want to learn more about combining performance testing and continuous integration? Watch our on-demand webcast on [How to Build Testing Into Your CI Pipeline](http://info.blazemeter.com/how-to-build-testing-into-continuous-integration-pipeline-webinar?utm_source=Blog&utm_medium=BM_Blog&utm_campaign=JenkinsJMeter), featuring a special guest DevOps Engineer from MIT, who reveals exactly how he added testing to the CI process and eliminated performance surprises.

How to Install the Jenkins and JMeter Plugin

In this article, I’m assuming that you already have [Jenkins](https://www.blazemeter.com/jenkins?utm_source=Blog&utm_medium=BM_Blog&utm_campaign=JenkinsJMeter) up and running somewhere. If you want to experiment, it’s pretty easy to get Jenkins. Here’s how:

1. Download the latest [Jenkins](https://www.blazemeter.com/jenkins?utm_source=Blog&utm_medium=BM_Blog&utm_campaign=JenkinsJMeter) release from the [https://jenkinsio/](https://jenkins-ci.org/) site (note: the .war file should be quite enough)

Increase image

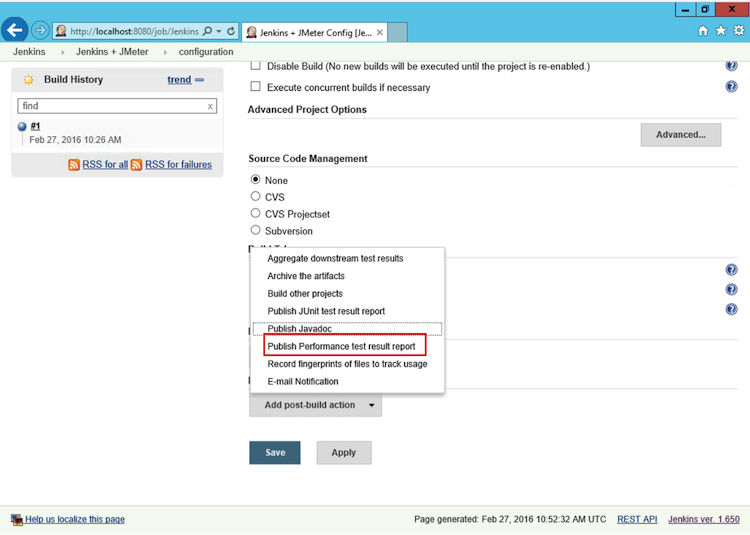
2. Run Jenkins as java -jar jenkins.war and wait for the “Jenkins is fully up and running" message.  
  
Increase image

3. Open the [http://localhost:8080](http://localhost:8080/) URL in your favourite browser

4. You should now be able to see the Jenkins dashboard

To get JMeter support on Jenkins, you need to use the [Performance Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Performance+Plugin). Once again, it’s really easy to install this plugin. Here’s how:

1. Download the latest version of the plugin from the [Performance Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Performance+Plugin)page
2. Copy the performance.hpi file to the plugins folder of your Jenkins’ installation. If you run Jenkins from the .war file - drop the plugin to the .jenkins/plugins path under your user home folder
3. Restart Jenkins to pick the plugin up
4. If everything is fine you should see the “Publish Performance Test Result Report” option under Jenkins -> Your Project -> Configure -> “Add post-build action” dropdown

Increase image

How to Run a JMeter Test in Jenkins

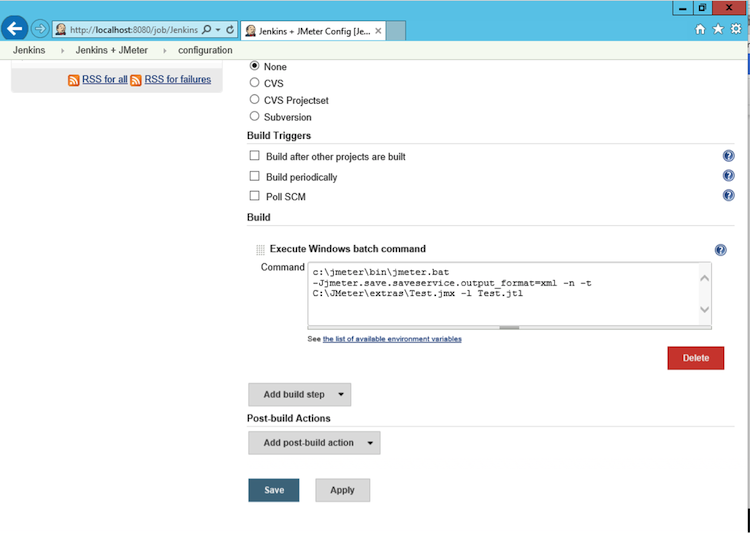
You’ll be happy to hear that running a [JMeter](https://www.blazemeter.com/jmeter-load-testing?utm_source=Blog&utm_medium=BM_Blog&utm_campaign=JenkinsJMeter) test in Jenkins isn’t very complex. Just remember that the test has to be run in non-GUI mode ([check out this article to learn more about running a JMeter test in non-GUI mode](https://blazemeter.com/blog/5-ways-launch-jmeter-test-without-using-jmeter-gui)).

For your first couple of runs, I recommend “telling” JMeter to store its results in an XML format using one of these approaches:

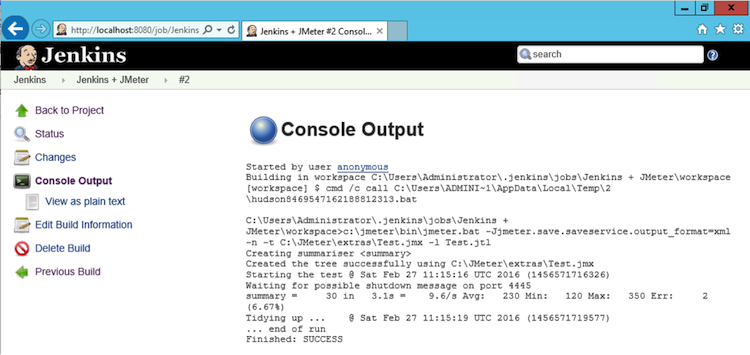
1. Add the jmeter.save.saveservice.output\_format=xml line to the user.properties file (this is located in /bin folder of your JMeter installation)
2. Pass the property via the -J command-line argument, such as: C:\jmeter\bin\jmeter.bat -Jjmeter.save.saveservice.output\_format=xml -n -t C:\jmeter\extras\Test.jmx -l Test.jtl

You can also use CSV files with the Performance Plugin. However, this requires extra  configuration whereas XML files work fine out of the box.

Here’s how it looks when you configure the Windows batch command to run a JMeter test:

Increase image

To verify the execution was successful (or to get extra information regarding any problems), check the “Console Output” tab of the project in question.

Increase image

In the JMeter log file and the .jtl results file, you can find it under the Project’s “Workspace”

Increase image

How to Use the Performance Plugin

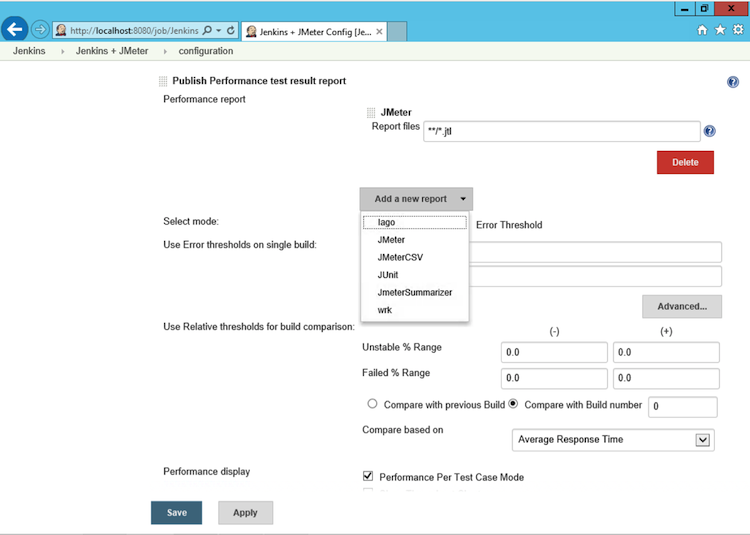
The Performance Plugin displays statistics, trends and can also be used to mark builds as failed based on results.

Here’s how you add it:

1. Open the Jenkins project
2. Click the “Configure” button
3. Click the “Add post-build action” button
4. Select:

-  “JMeter” if your results file is in an XML format

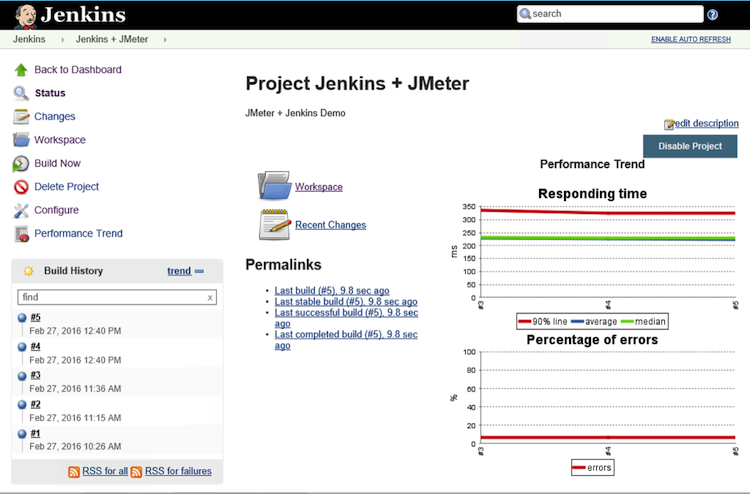
- “JMeterCSV” if your results file is in a CSV format

Increase image

“Report Files” is the main configuration option here. This is the path where the plugin will scan for JMeter results’ files. The format matches the [Ant Fileset](https://ant.apache.org/manual/Types/fileset.html)include/exclude format, and it’s relative to the Jenkins Workspace for the project.

All the other names of the settings here are quite self-explanatory.

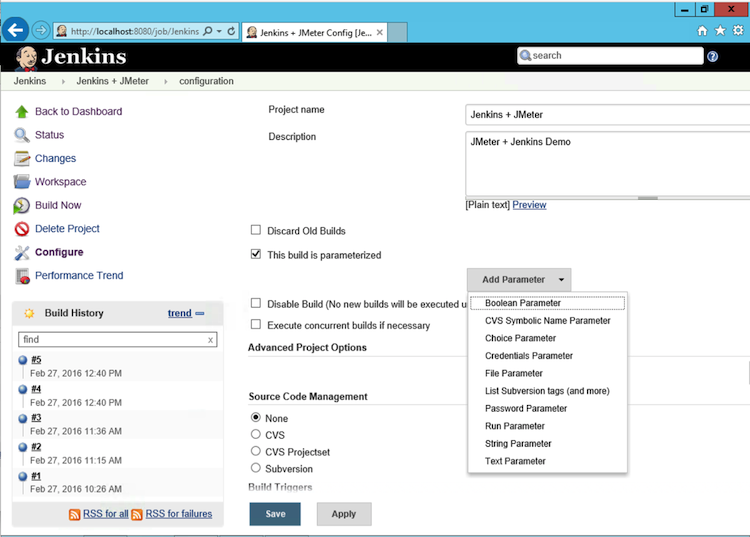
On the first build with the plugin enabled, you’ll see empty charts for the “Performance Trend”.

Increase image

The charts will change as you create more builds. They will display average, median and 90 percentile values along with the errors count. The charts are clickable, so you can get more detailed information on each executed sampler.

Passing Jenkins’ Parameters to a JMeter Test

Jenkins allows you to specify [build parameters](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Build). For example: if you tick the “This build is parameterized” box on the “Configure” page, you’ll be able to set some parameter names along with the default values:

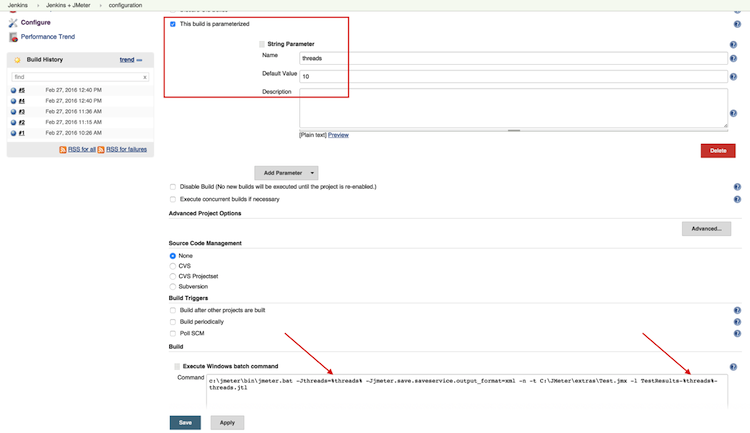
Increase image

This means that you can perform the configuration of your JMeter test without having to change anything in your .JMX script or Ant or Maven configuration.

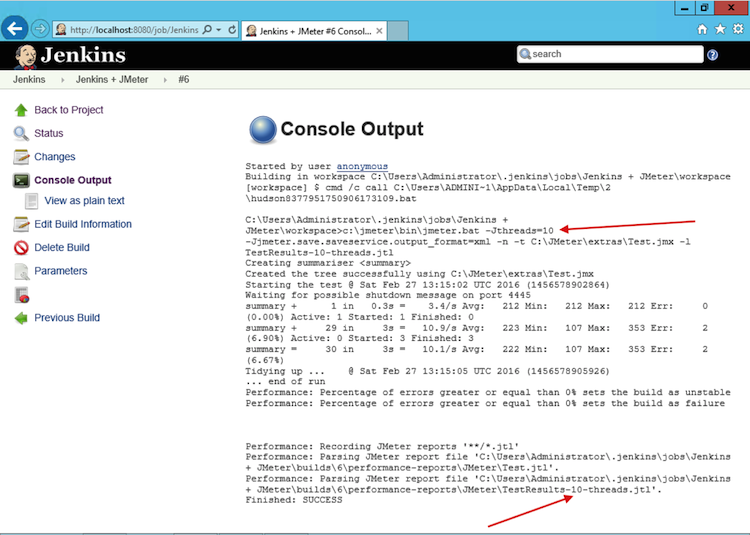
For instance, if you define the parameter with the name “foo” and the value “bar”,  you can refer to it as:

* %foo% - if the Jenkins build is executed on a Windows machine
* $foo  - if the Jenkins build is executed on Linux, Unix, or MacOSX machine

See below for an example of setting a JMeter thread count and accessing it from a JMeter Windows batch command text:

Increase image

If you run the build once more and take a look into the “Console Output”, you’ll see that the %threads% variable has been changed with the relevant parameter value:

Increase image

Congratulations! You’re now ready to start using JMeter with Jenkins. If you’re a BlazeMeter user, you can use [BlazeMeter's Jenkins Plugin](https://guide.blazemeter.com/hc/en-us/articles/206732669-BlazeMeter-s-Jenkins-Plugin?utm_source=Blog&utm_medium=BM_Blog&utm_campaign=JenkinsJMeter) to kick off a test in the BlazeMeter cloud directly from Jenkins.