

Open in app ↗

Sign up

Sign in

Medium

Search

Write



Load testing with Apache JMeter



Arie M. Prasetyo · Follow

7 min read · Jun 1, 2021



4

Photo by [Norbert Kundrak](#) on [Unsplash](#)

There are several ways to test the performance of a web application. In this article I want to discuss about how we can use Apache JMeter to do that. I'll start with a basic use. And in the end of the article, I'll discuss about how we can use JMeter to do a concurrent users stress test.

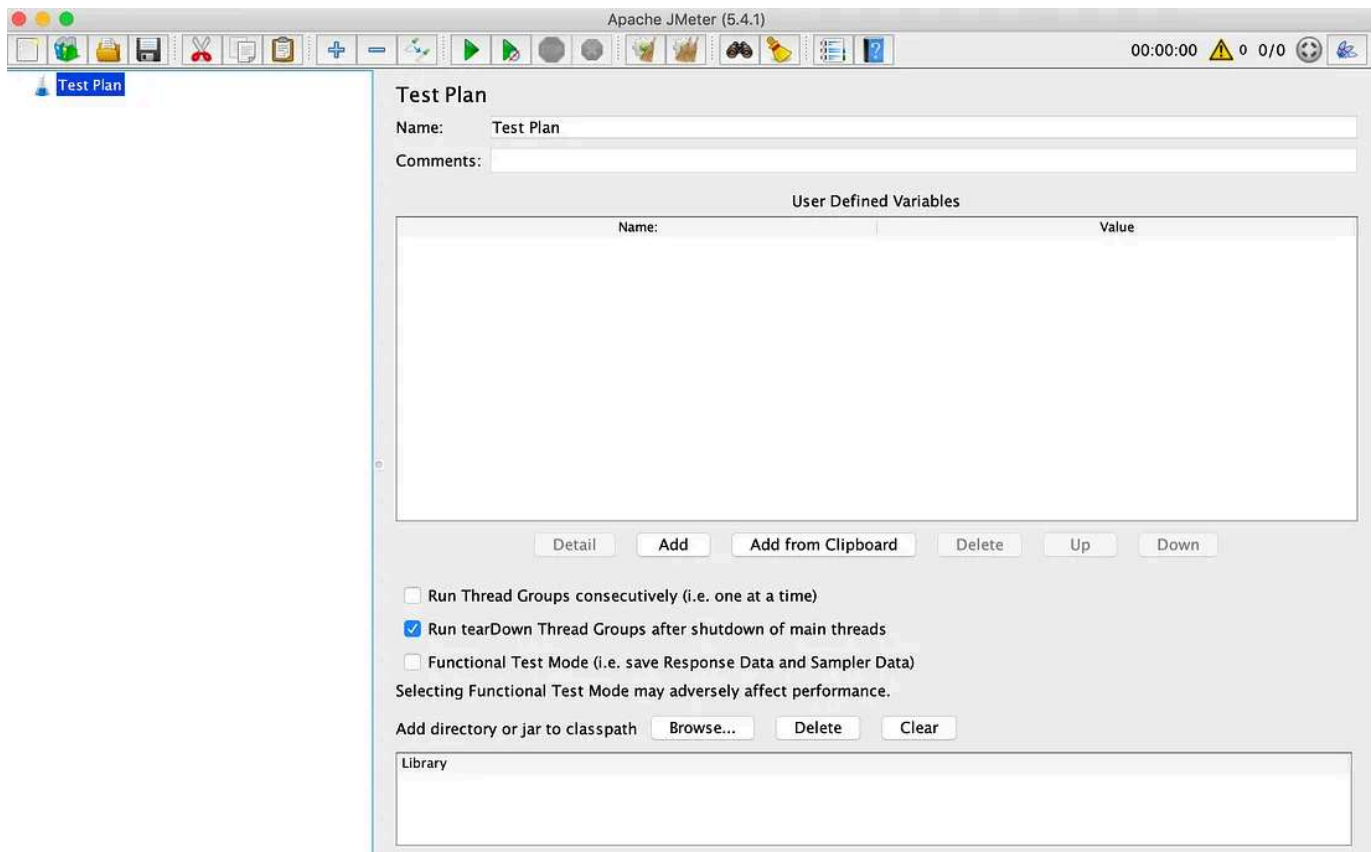
Write a basic test

There are two ways we can use JMeter:

1. Graphical user interface
2. Command line interface

It is recommended that we run actual test on CLI. But we can write a test & have a test-run on the GUI. Start the GUI by running this command:

```
$ apache-jmeter-5.4.1/bin/jmeter
```



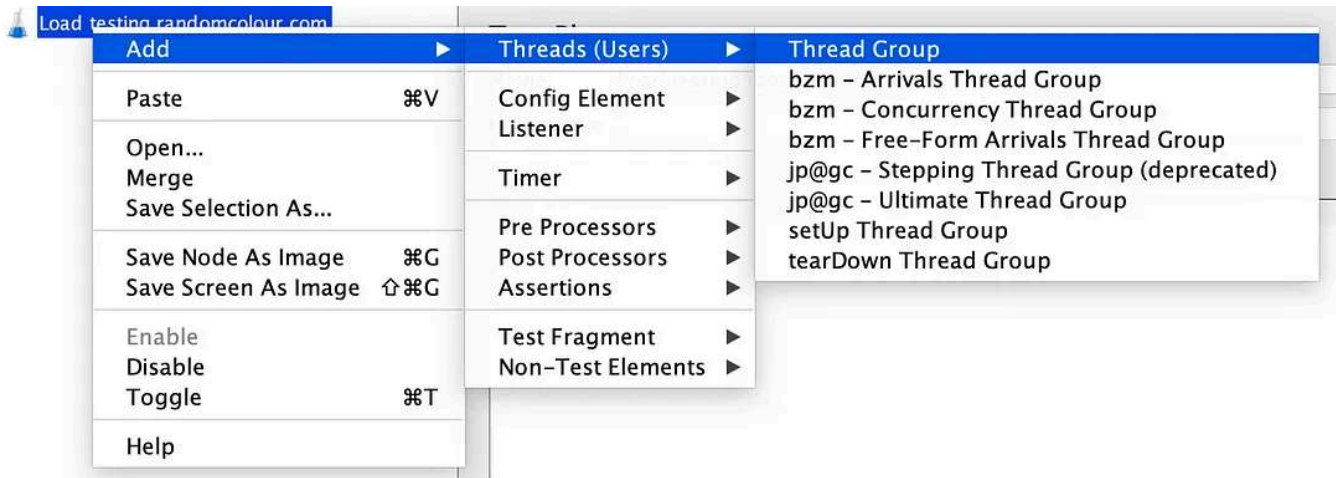
Apache JMeter GUI

In this scenario, we are going to write a basic performance test targeting randomcolour.com, a very simple website.

I'll rename the test plan as "*Load testing randomcolour.com*".

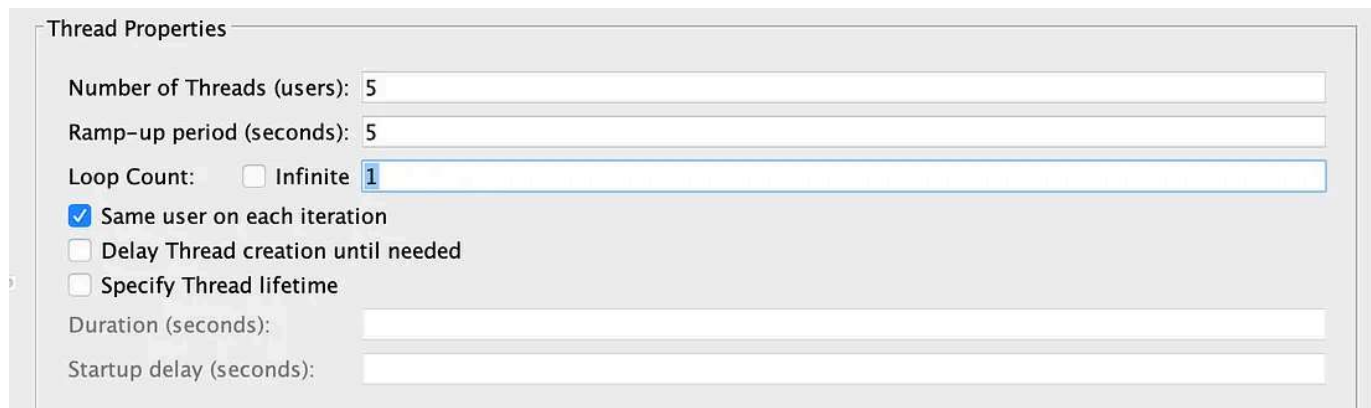
Thread group

Next, let's add a thread group on our test plan. Thread group determines user flow and how a user would behave. Each thread represent a user.



Add a thread group

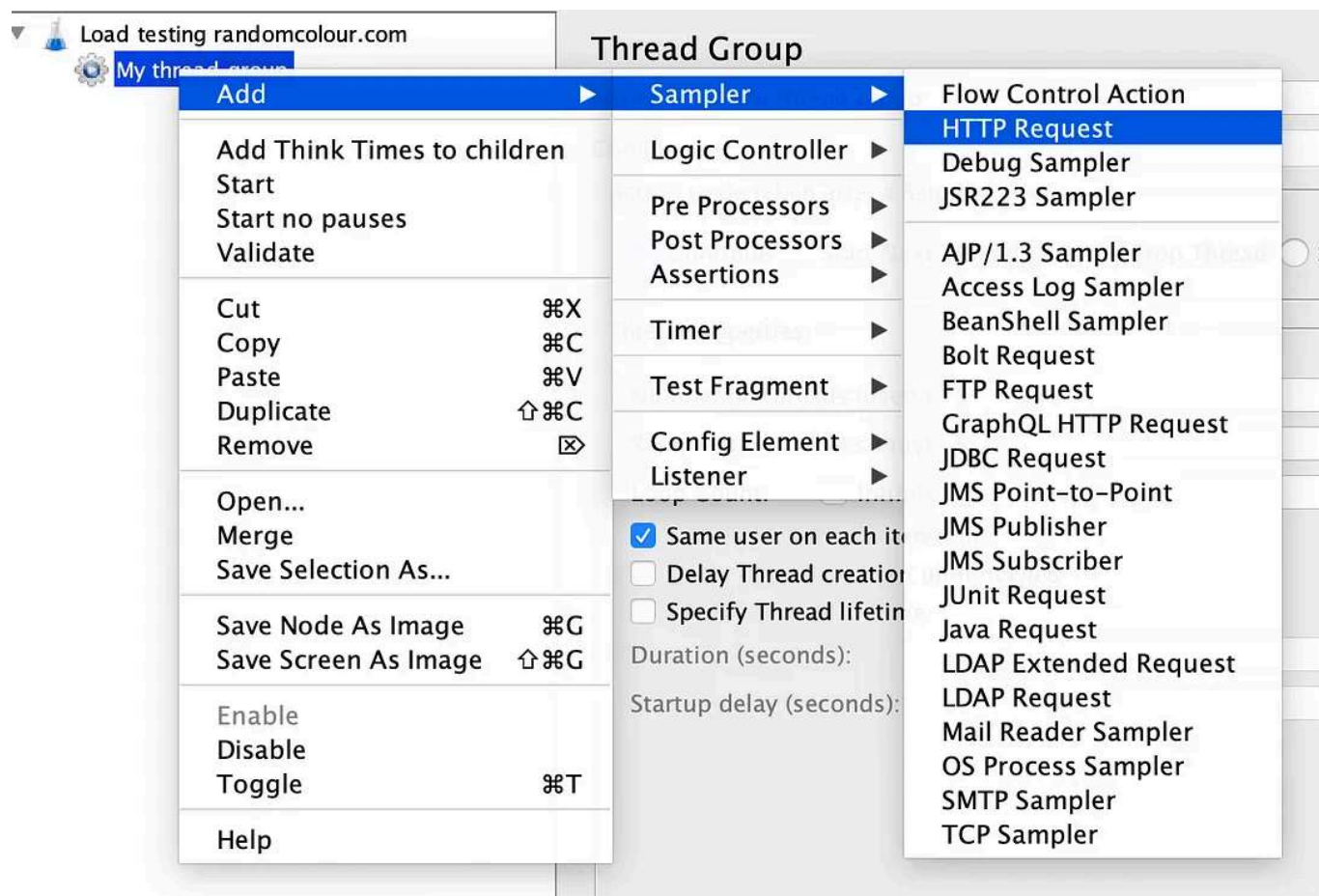
In this thread group, I am adding *5 users* (threads), ramp all users in *5 seconds* (thus 1 second after each other). And I'll only be running the test once (*loop count = 1*).



Thread configuration

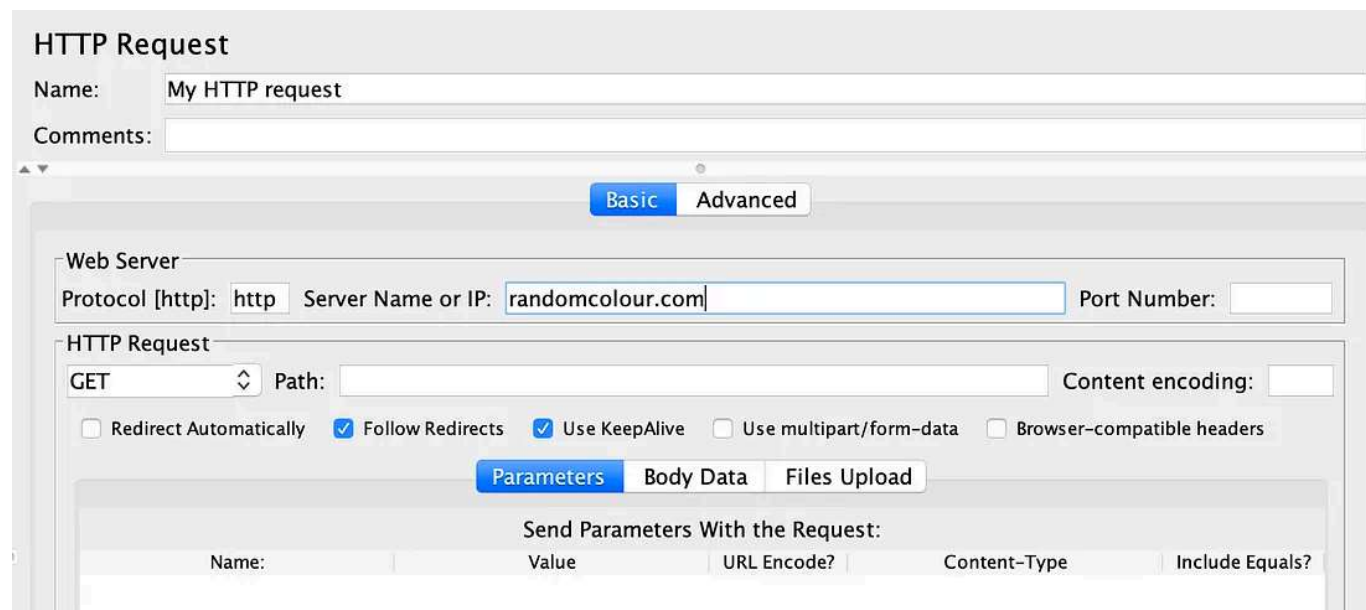
Sampler

We want the thread to send an HTTP request. To do this we can add a “Sampler” to the thread group.



Add a sampler

In this sampler, I'm sending a request to `randomcolour.com`, using an `http` protocol.

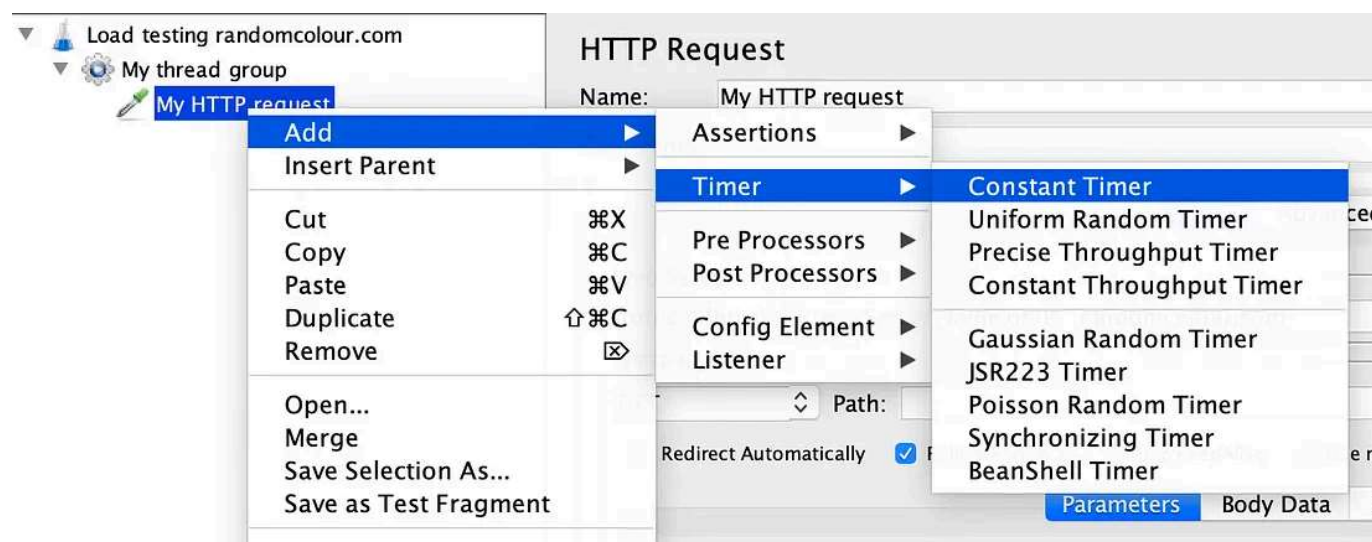


Sampler configuration

As you can see, we can configure many parameters on this screen. We can simulate a POST request, simulate form filling action, file uploads, etc. But in this example I'm just going to request a simple HTTP request.

Timer

We can simulate the pause/delay when users click a website by adding a "Constant Timer". Let's add a timer to the HTTP request we've just created.

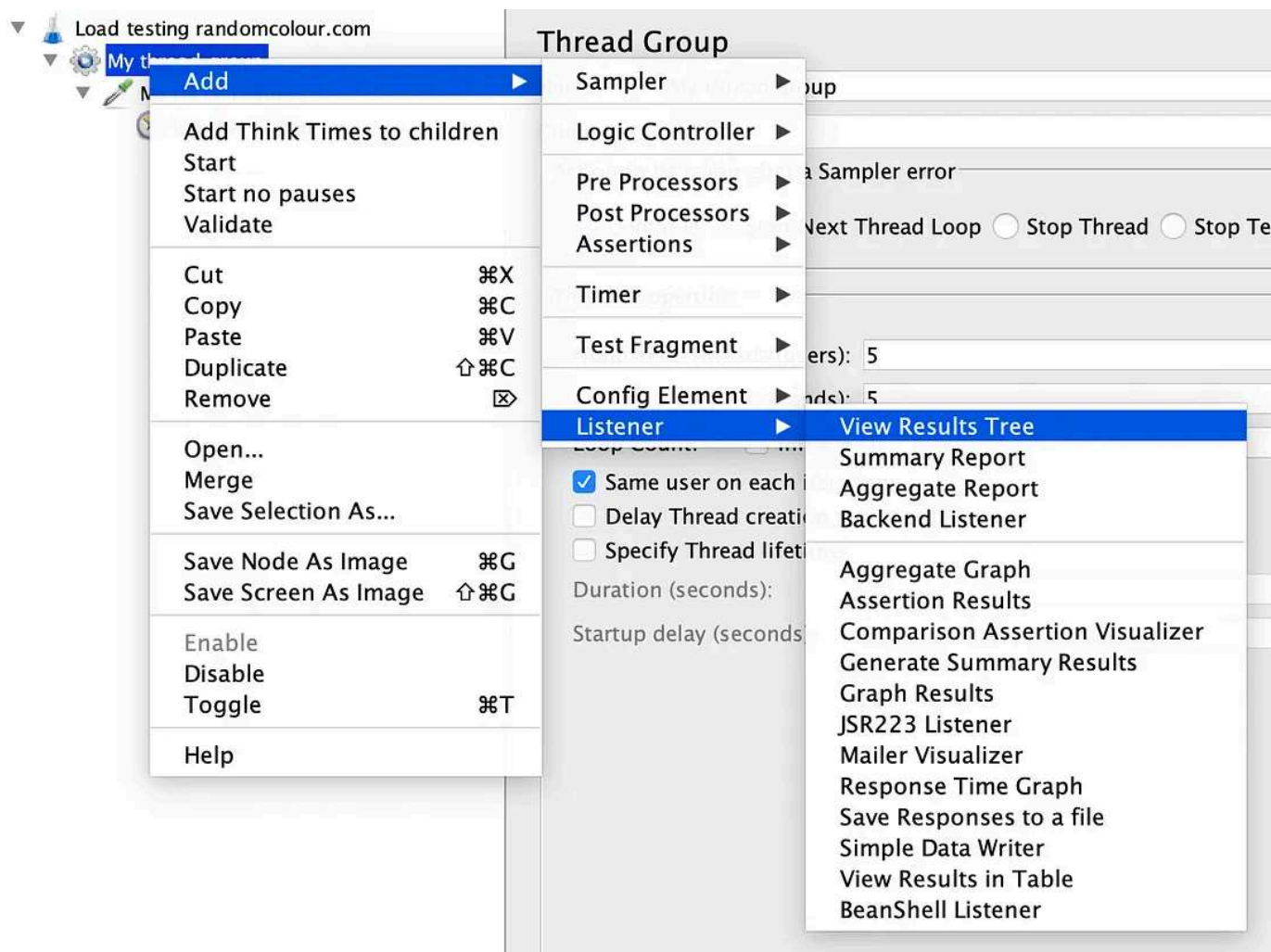


Add a timer

I'm going to use the default value of 300ms.

Listener

To see the result of our test on the GUI, we can add a "Listener". In this example, I'm adding a "View Results Tree" listener to the thread group.



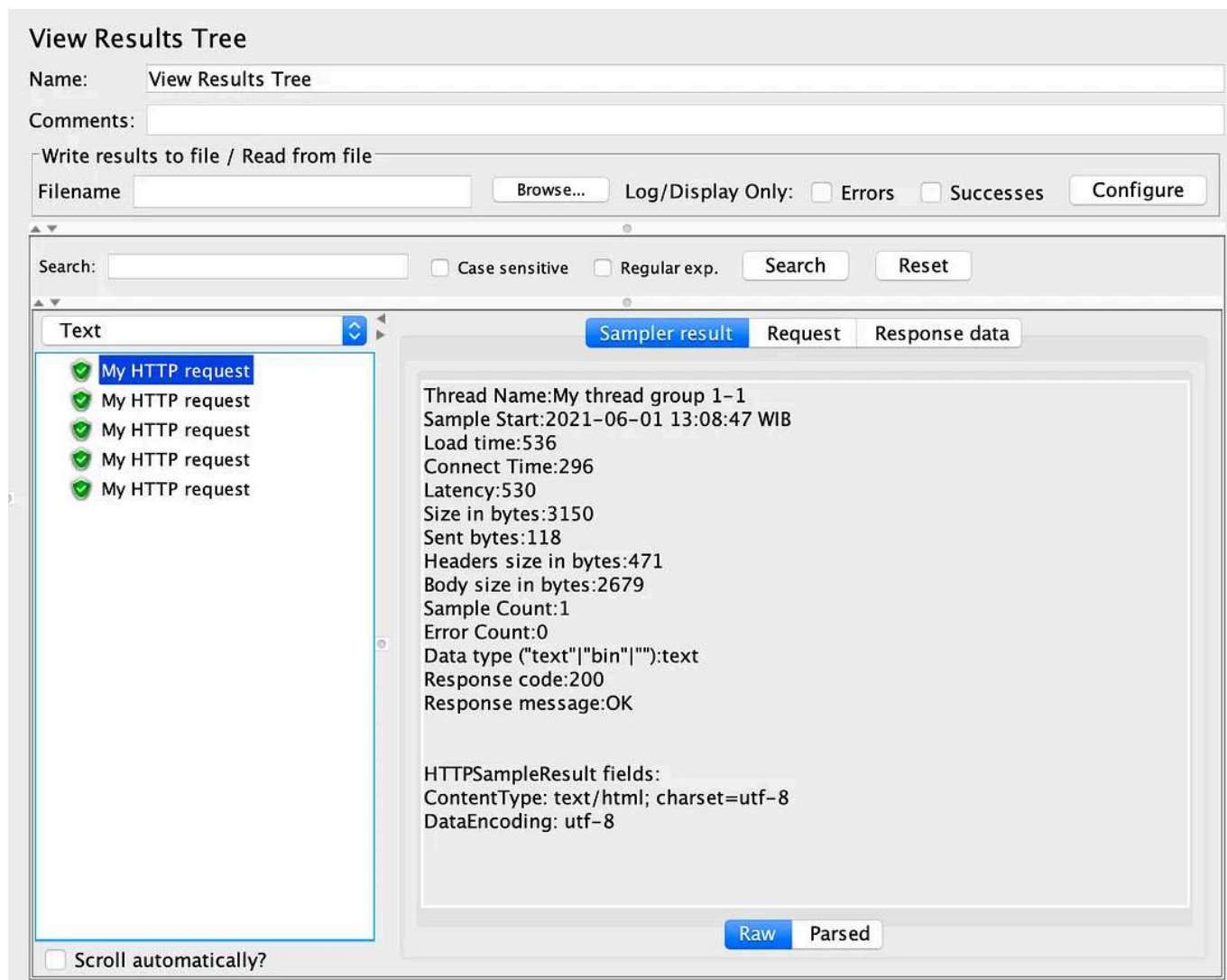
Add a listener

Run test on GUI

Remember, you should always run the test on JMeter CLI interface. But we can do a test-run on the plan we've just created. First we need to save it as a `.jmx` file. In this example, I name the file `test-randomcolour.jmx`.

To start a test, click the "Start" button on the top toolbar.

Here is the result of our test as shown in the "View Results Tree" screen.



The test result on JMeter GUI

Run test on CLI

To run the test on CLI, we can just use the `.jmx` file created using the GUI. Use the parameter `-n` to tell JMeter to run in non-GUI mode. Use the parameter `-t` to specifies the file to use for the test*.

**in this example, I store my `.jmx` files in the `configs/` directory*


```
$ apache-jmeter-5.4.1/bin/jmeter -n -t configs/test-randomcolour.jmx
```

Below is a screenshot of the test result. As you can see, JMeter returns the summary of the test which it runs on port 4445.

```
→ load_testing apache-jmeter-5.4.1/bin/jmeter -n -t configs/test-randomcolour.jmx
WARNING: package sun.awt.X11 not in java.desktop
Creating summariser <summary>
Created the tree successfully using configs/test-randomcolour.jmx
Starting standalone test @ Tue Jun 01 13:16:31 WIB 2021 (1622528191790)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:01 =    1.1/s Avg:   500 Min:   500 Max:   500 Err:    0 (0.00%) Active: 1 Started: 1 Finished: 0
summary +      4 in 00:00:05 =    0.8/s Avg:   729 Min:   478 Max:  1482 Err:    0 (0.00%) Active: 0 Started: 5 Finished: 5
summary =      5 in 00:00:06 =    0.9/s Avg:   683 Min:   478 Max:  1482 Err:    0 (0.00%)
Tidying up ...    @ Tue Jun 01 13:16:37 WIB 2021 (1622528197662)
... end of run
```

The test results on JMeter CLI



Concurrency load test

We can also do concurrency load test where we simulate multiple users (threads) accessing a website simultaneously. JMeter doesn't have a default module for this, but luckily it supports plugins.

Installing JMeter plugin manager

You can find the JMeter Plugins Manager [here](#). Simply download the JAR file, then copy it to JMeter's `lib/ext/` directory.

Open or restart JMeter GUI.

You can find the "Plugins Manager" on the "Options" menu and on the right of the top menubar.

Installing Concurrency Thread Group plugin

1. Open the "Plugins Manager" on the GUI
2. Click on the "Available Plugins" tab
3. On the search bar, type in "Custom Thread Groups"
4. Select it and click the "Apply Changes and Restart JMeter" button

Custom Thread Groups

Vendor: *JMeter-Plugins.org*

Adds new Thread Groups:

- [Stepping Thread Group](#)
- [Ultimate Thread Group](#)
- [Concurrency Thread Group](#)
- [Arrivals Thread Group](#)
- [Free-Form Arrivals Thread Group](#)

Documentation: <https://jmeter-plugins.org/wiki/ConcurrencyThreadGroup/>

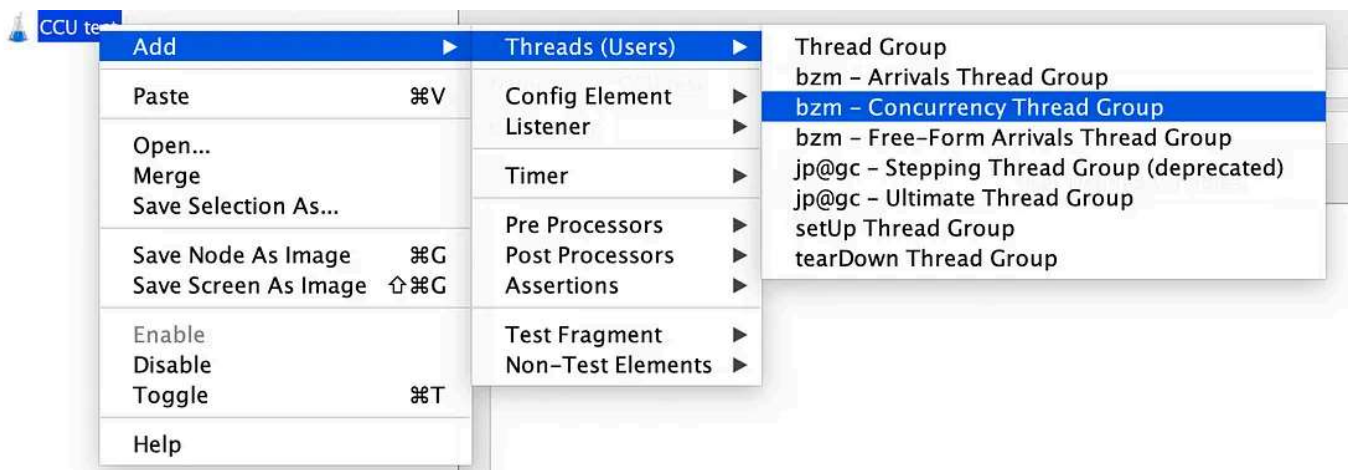
What's new in version 2.9: Support JMeter's way of prefixing threads for distributed testing

Maven groupId: *kg.apc*, artifactId: *jmeter-plugins-casutg*, version: *2.9*

Custom Thread Groups plugin

Write a concurrent load test

Open the JMeter GUI to create a new test. Then, add a concurrency thread group.

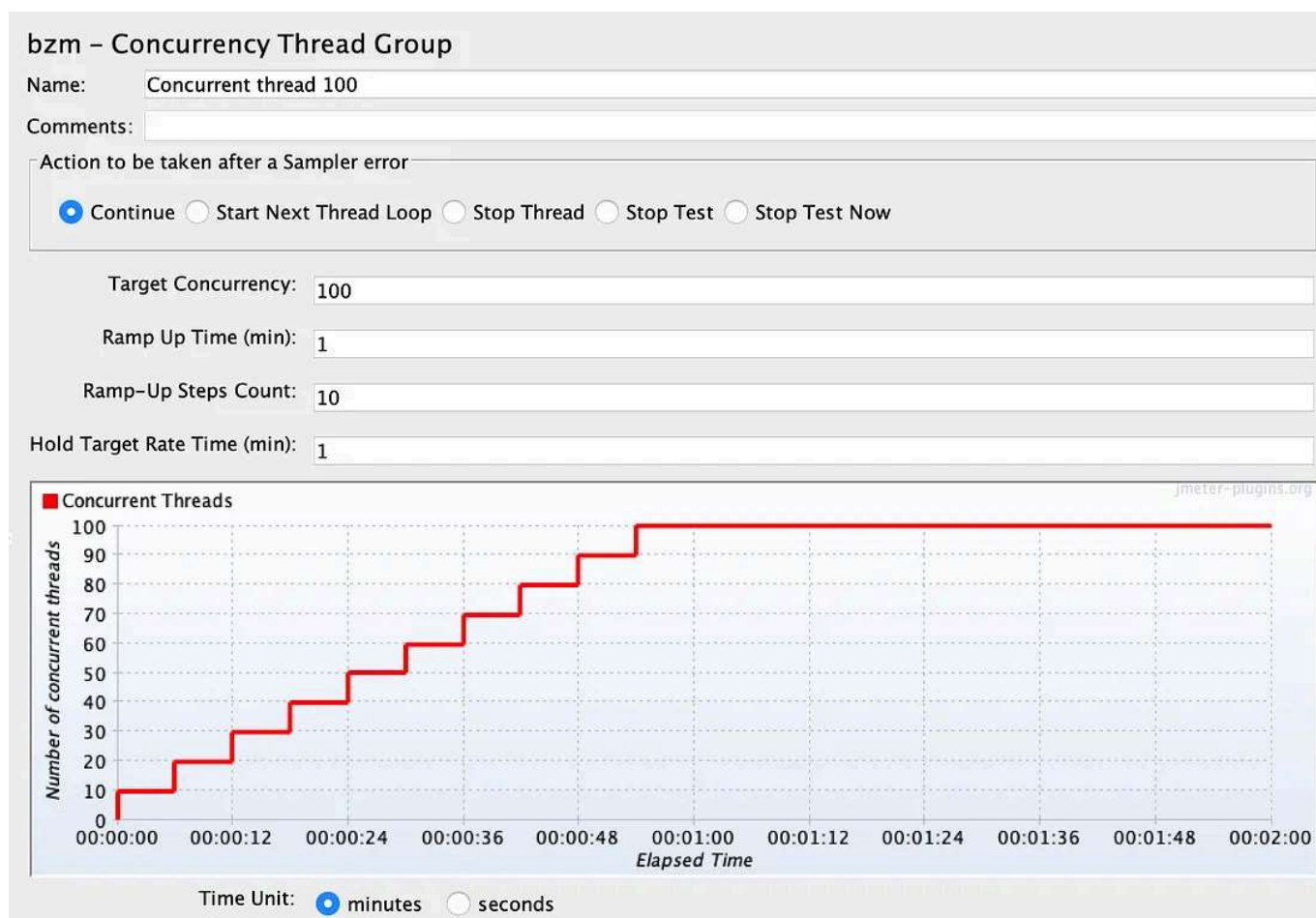


Add a Concurrency Thread Group

In this test I am going to simulate *100 concurrent users* (threads). We'll load all users within *1 minute in 10 increments* (thus 10 users per 6 seconds). Once all

100 threads are added, we *hold the connection for at least 1 minute**.

*we can manually stop the connection after 1 minute.



Concurrency thread group configuration

Next, we add the sampler, an HTTP request. We use a similar configuration as before: sending a request to `randomcolour.com`, using an `http` protocol.

For this test, I won't be adding a listener (more on that later). For the final step, we save the test as a `.jmx` file. I name my test file as: `test-randomcolour-ccu.jmx`.

Run the concurrent test

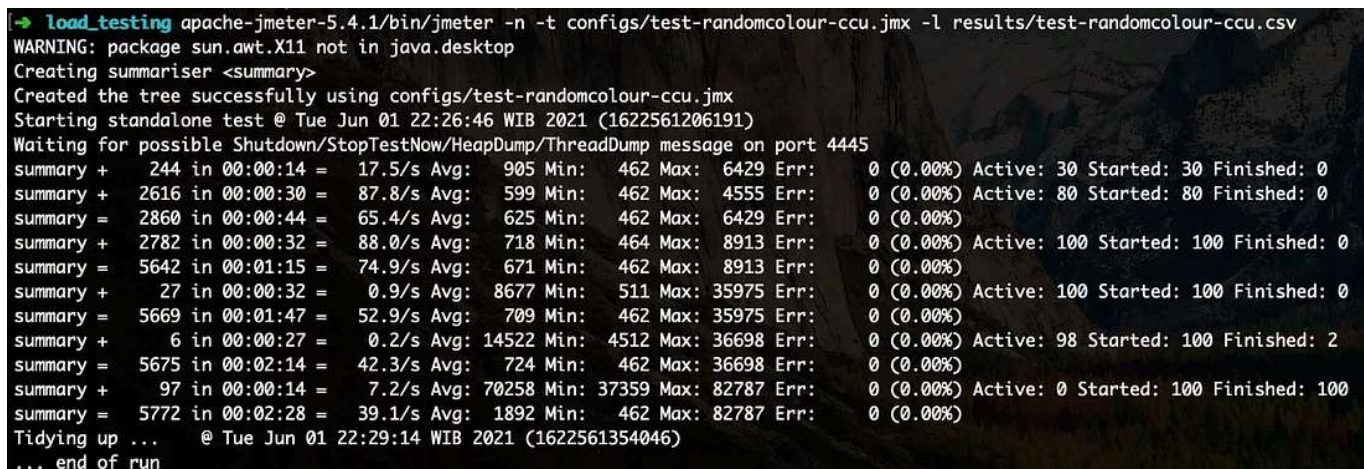
We run the test using CLI. But this time, we want to save the result as a CSV file. We can open the CSV file on a spreadsheet application to check the result. But more importantly, we can create a summary report using JMeter afterwards.

Run the CLI command to run the test and create a report, add the parameter `-l` to indicate where we want to save the report file*.

**in this example, I save the report file as `test-randomcolour-ccu.csv` in the `results/` directory.*

```
$ apache-jmeter-5.4.1/bin/jmeter ↵
-n -t configs/test-randomcolour-ccu.jmx ↵
-l results/test-randomcolour-ccu.csv
```

Below is a screenshot of the test result and CSV file.



```
load_testing apache-jmeter-5.4.1/bin/jmeter -n -t configs/test-randomcolour-ccu.jmx -l results/test-randomcolour-ccu.csv
WARNING: package sun.awt.X11 not in java.desktop
Creating summariser <summary>
Created the tree successfully using configs/test-randomcolour-ccu.jmx
Starting standalone test @ Tue Jun 01 22:26:46 WIB 2021 (1622561206191)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 244 in 00:00:14 = 17.5/s Avg: 905 Min: 462 Max: 6429 Err: 0 (0.00%) Active: 30 Started: 30 Finished: 0
summary + 2616 in 00:00:30 = 87.8/s Avg: 599 Min: 462 Max: 4555 Err: 0 (0.00%) Active: 80 Started: 80 Finished: 0
summary = 2860 in 00:00:44 = 65.4/s Avg: 625 Min: 462 Max: 6429 Err: 0 (0.00%)
summary + 2782 in 00:00:32 = 88.0/s Avg: 718 Min: 464 Max: 8913 Err: 0 (0.00%) Active: 100 Started: 100 Finished: 0
summary = 5642 in 00:01:15 = 74.9/s Avg: 671 Min: 462 Max: 8913 Err: 0 (0.00%)
summary + 27 in 00:00:32 = 0.9/s Avg: 8677 Min: 511 Max: 35975 Err: 0 (0.00%) Active: 100 Started: 100 Finished: 0
summary = 5669 in 00:01:47 = 52.9/s Avg: 709 Min: 462 Max: 35975 Err: 0 (0.00%)
summary + 6 in 00:00:27 = 0.2/s Avg: 14522 Min: 4512 Max: 36698 Err: 0 (0.00%) Active: 98 Started: 100 Finished: 2
summary = 5675 in 00:02:14 = 42.3/s Avg: 724 Min: 462 Max: 36698 Err: 0 (0.00%)
summary + 97 in 00:00:14 = 7.2/s Avg: 70258 Min: 37359 Max: 82787 Err: 0 (0.00%) Active: 0 Started: 100 Finished: 100
summary = 5772 in 00:02:28 = 39.1/s Avg: 1892 Min: 462 Max: 82787 Err: 0 (0.00%)
Tidying up ... @ Tue Jun 01 22:29:14 WIB 2021 (1622561354046)
... end of run
```

Concurrency thread group test result

timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
1622561206412	6286	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-5	text	true		3130	118	20	20	http://randomcolour.com/	6283	0	6025
1622561206411	6287	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-3	text	true		3150	118	20	20	http://randomcolour.com/	6284	0	6039
1622561206411	6287	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-6	text	true		3152	118	20	20	http://randomcolour.com/	6284	0	6026
1622561206411	6289	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-2	text	true		3152	118	20	20	http://randomcolour.com/	6288	0	6041
1622561206412	6286	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-7	text	true		3131	118	20	20	http://randomcolour.com/	6283	0	6027
1622561206411	6287	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-1	text	true		3129	118	20	20	http://randomcolour.com/	6284	0	6026
1622561206411	6287	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-10	text	true		3150	118	20	20	http://randomcolour.com/	6284	0	6030
1622561206412	6286	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-8	text	true		3152	118	20	20	http://randomcolour.com/	6283	0	6038
1622561206412	6286	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-9	text	true		3150	118	20	20	http://randomcolour.com/	6283	0	6025
1622561212337	492	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-11	text	true		3150	118	20	20	http://randomcolour.com/	491	0	254
1622561212347	482	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-16	text	true		3151	118	20	20	http://randomcolour.com/	481	0	244
1622561212348	490	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-17	text	true		3128	118	20	20	http://randomcolour.com/	490	0	243
1622561212349	489	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-18	text	true		3150	118	20	20	http://randomcolour.com/	489	0	242
1622561212351	488	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-19	text	true		3149	118	20	20	http://randomcolour.com/	487	0	240
1622561212351	489	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-20	text	true		3151	118	20	20	http://randomcolour.com/	489	0	243
1622561212342	498	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-13	text	true		3129	118	20	20	http://randomcolour.com/	498	0	249
1622561206412	6429	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-4	text	true		3129	118	20	20	http://randomcolour.com/	6429	0	6050
1622561212346	499	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-15	text	true		3151	118	20	20	http://randomcolour.com/	499	0	248
1622561212344	502	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-14	text	true		3150	118	20	20	http://randomcolour.com/	502	0	247
1622561212338	517	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-12	text	true		3151	118	20	20	http://randomcolour.com/	515	0	256
1622561212715	472	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-6	text	true		3150	118	20	20	http://randomcolour.com/	472	0	232
1622561212714	473	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-8	text	true		3149	118	20	20	http://randomcolour.com/	473	0	235
1622561212714	479	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-9	text	true		3151	118	20	20	http://randomcolour.com/	478	0	237
1622561212715	484	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-2	text	true		3129	118	20	20	http://randomcolour.com/	484	0	234
1622561212714	493	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-3	text	true		3150	118	20	20	http://randomcolour.com/	493	0	241
1622561212714	494	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-1	text	true		3151	118	20	20	http://randomcolour.com/	493	0	244
1622561212714	513	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-5	text	true		3149	118	20	20	http://randomcolour.com/	493	0	244
1622561212714	514	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-10	text	true		3151	118	20	20	http://randomcolour.com/	514	0	249
1622561212714	514	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-7	text	true		3152	118	20	20	http://randomcolour.com/	514	0	254
1622561212839	479	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-17	text	true		3149	118	20	20	http://randomcolour.com/	479	0	237
1622561212847	471	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-14	text	true		3149	118	20	20	http://randomcolour.com/	471	0	230
1622561212841	477	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-13	text	true		3149	118	20	20	http://randomcolour.com/	477	0	236
1622561212841	494	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-20	text	true		3152	118	20	20	http://randomcolour.com/	493	0	237
1622561212846	489	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-15	text	true		3151	118	20	20	http://randomcolour.com/	488	0	243
1622561212839	496	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-18	text	true		3151	118	20	20	http://randomcolour.com/	496	0	250
1622561212829	506	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-16	text	true		3151	118	20	20	http://randomcolour.com/	506	0	249
1622561212839	497	HTTP Request	200	OK	Concurrent thread 100-ThreadStarter 1-19	text	true		3149	118	20	20	http://randomcolour.com/	497	0	250

Concurrency thread group test result CSV file

Note: we can manually stop the test from another terminal by running this command:

```
$ apache-jmeter-5.4.1/bin/stoptest.sh
```

Create a summary report

Using the CSV file created during the test we can make a summary report on JMeter GUI.

On the default test plan, add the listener “Summary Report”. We don’t have to create a thread group or a sampler, because we are going to read the CSV file to generate the report. Click the “Browse...” button and select `results/test-randomcolour-ccu.csv`.

Summary Report

Name:Summary Report

Comments:

Write results to file / Read from file

Filenamece/load_testing/results/test-randomcolour-ccu.csv

Browse...

Log/Display Only:

☐Errors

☐Successes

Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/ ...	Sent KB/sec	Avg. Bytes
HTTP Requ...	5772	1892	462	82787	9045.23	0.00%	39.1/sec	120.27	4.51	3149.9
TOTAL	5772	1892	462	82787	9045.23	0.00%	39.1/sec	120.27	4.51	3149.9

Summary generated from the report file

How to read the summary report

This is the description about the columns in the summary report:

1. **Samples:** total number of requests sent to the server during the duration of the test
2. **Average:** sum of all the sample times divided by total number of requests
3. **Min:** minimum time taken for the request to be completed (ms)
4. **Max:** maximum time taken for the request to be completed (ms)
5. **Std. Dev.:** Standard deviation of the response time average value
6. **Error %:** percentage of failed tests
7. **Throughput:** number of requests processed by the server per unit time
8. **Received KB/sec:** *self explanatory*
9. **Sent KB/sec:** *self explanatory*
10. **Avg. Bytes:** average response size

Conclusion

There are many other scenarios on how we can use Apache JMeter to test the performance of our website. We can simulate user behaviour, run several thread groups, etc. This article should help you getting a basic understanding on how to start using JMeter. It is a very useful tool to have in your development process, especially if you want to know the performance and scalability of your website.

If you have any question or suggestion, give me a nudge on [Twitter](#). Cheers!

Additional reading

1. [Getting Started with JMeter — A Basic Tutorial](#)
2. [Load Testing using Concurrency Thread Group in Apache JMeter](#)
3. [JMeter Result Analysis: The Ultimate Guide](#)

Test

Performance

Load Testing

Scalability

**Written by Arie M. Prasetyo**

527 Followers · 285 Following

Programmer

Follow



No responses yet



Write a response

What are your thoughts?

More from Arie M. Prasetyo



In ecomindo-dev by Arie M. Prasetyo

Comparison of gen-AI tools in software development

Comparing how different gen-AI tools work and how each can help us develop an...

Jan 11 53



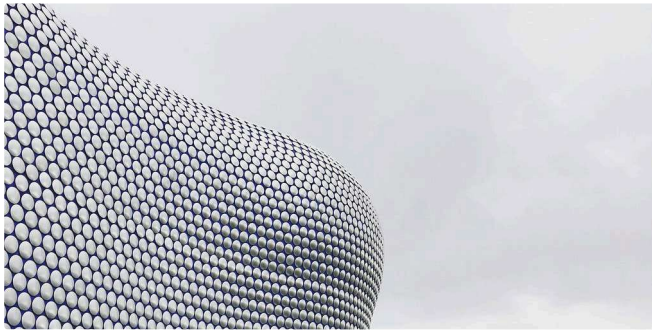
Arie M. Prasetyo

Using AI for software design & development

Artificial Intelligence is all the rage at the moment, from generating images to poweri...

Mar 24, 2024 27 1





Arie M. Prasetyo

Full stack web dev with Nest, GraphQL, and React

Getting my hands dirty on some of the latest technologies in web development

Nov 13, 2023



57



In ecomindo-dev by Arie M. Prasetyo

Integrasi dengan agen AI

Panduan singkat proses pembuatan & integrasi agen AI dari Pusaka.ai dengan...

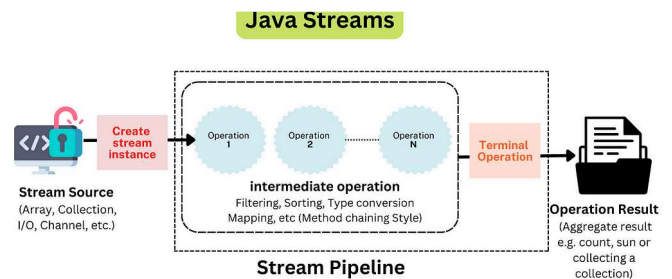
Feb 7



52

[See all from Arie M. Prasetyo](#)

Recommended from Medium





In Dev Genius by Rahul Ranjan

OpenTelemetry Collector : A Gateway to Modern Observability

What is Opentelemetry Collector?



Sep 25, 2024



12



Mohit Bajaj

How to use If/else Logic in Java Streams: 10 Practical Approaches

Remember when Java 8 introduced Streams and suddenly everyone was talking about thi...



4d ago



14



Lists



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 562 saves



A Guide to Choosing, Planning, and Achieving...

13 stories · 2362 saves



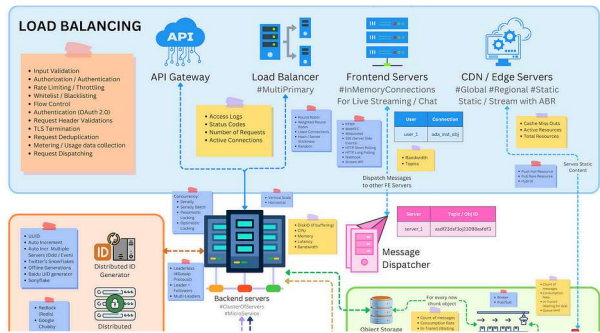
Generative AI Recommended Reading

52 stories · 1686 saves



Staff picks

822 stories · 1645 saves



 In ByteByteGo System Design Alli... by Love Shar...

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However,...

★ Sep 17, 2023 🖱️ 9.1K 💬 57 📌


 In System Design for Beginners: A Roa... by Yatins...

The Most Dangerous Linux Commands You Should NEVER R...

One wrong command can wipe out your entire system—don't make these mistakes!

★ 3d ago 🖱️ 3 💬 1 📌



 In DevOps.dev by Rafał Buczyński

Performance Testing with JMeter

What is JMeter?

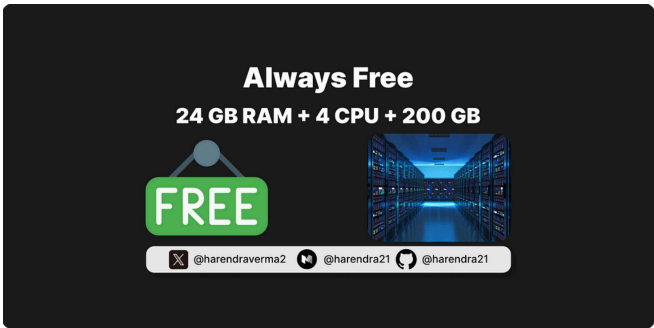
Feb 27 📌

 Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

★ Oct 26, 2024 🖱️ 9.4K 💬 173 📌



See more recommendations