**Basic Command to Run JMeter in non-GUI Mode**

Command:

*jmeter -n -t <path-to-your-script>.jmx -l <result-file>.jtl*

-n → non-GUI mode

-t → Test plan file (JMX file)

-l → Results file (.jtl for storing logs/results)

**Complete List of JMeter Command-Line Arguments with Descriptions & Examples**

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Example** |
| -n | Run in Non-GUI (headless) mode | -n |
| -t <file> | Path to your JMX test plan | -t test-plan.jmx |
| -l <file> | Log results to a file (.jtl) | -l result.jtl |
| -j <file> | Write JMeter log to custom file | -j jmeter-log.txt |
| -r | Run test on remote servers (defined in jmeter.properties) | -r |
| -R <list> | Run test on specified remote servers only | -R server1,server2 |
| -Gprop=value | Set a property to be sent to all remote servers | -Gusers=100 |
| -X | Print debugging info (for developers) | -X |
| -q <file> | Additional property file to load (can be repeated) | -q custom.properties |
| -p <file> | JMeter properties file to use | -p my-jmeter.properties |
| -J<prop>=<value> | Set a JMeter user property (can be overridden in script) | -Jthreads=50 |
| -D<prop>=<value> | Set a system property | -Dlog\_level.jmeter=DEBUG |
| -L<component>=<LEVEL> | Set logging level for a component | -Lorg.apache=DEBUG |
| -H <proxy> | Set proxy server hostname | -H proxy.example.com |
| -P <port> | Set proxy server port | -P 8080 |
| -u <username> | Proxy username | -u user123 |
| -a <password> | Proxy password | -a pass123 |
| -S <host> | Non-proxy hosts list | -S localhost,127.0.0.1 |
| -e | Generate dashboard report after test | -e |
| -o <folder> | Output folder for dashboard report (used with -e) | -o report-folder |
| --addprop <file> | Load properties file (same as -q) | --addprop props.properties |
| --loglevel <level> | Set global log level | --loglevel DEBUG |

**Practical Example**

**Command:**

*jmeter -n -t loadtest.jmx -l results.jtl -j jmeter.log -e -o /output/report -Jthreads=100 -Jrampup=10 -Jduration=60*

If your **JMeter script (.jmx)** is located in a different path like:

Example:

C:\Users\XXXXXX\XXXXXX\Documents\XXX\XX\XX\script

You can **safely pass the full path** in the -t (test plan) option by **wrapping it in double quotes**

**Command:**

*jmeter -n -t "C:\Users\XXXX\XXX\Documents\XX\XX\XX\script\test-plan.jmx" -l "results.jtl" -j "jmeter.log" -e -o "report" -Jthreads=100 -Jrampup=10 -Jduration=60*

**Explanation**

* -t "full\path\to\your.jmx" — Always wrap in quotes if the path has spaces.
* The output files (.jtl, .log, report folder) will be created in the current directory unless you specify full paths for them too.

**Optional: Set full path for logs & report**

**Command:**

*jmeter -n -t "C:\...\test-plan.jmx" ^*

*-l "C:\Results\results.jtl" ^*

*-j "C:\Results\jmeter.log" ^*

*-e -o "C:\Results\report" ^*

*-Jthreads=100 -Jrampup=10 -Jduration=60*

Use ^ (line continuation) to keep the command readable in CMD or batch scripts.

**Explanation:**

* Runs loadtest.jmx in non-GUI
* Logs results to results.jtl
* Uses jmeter.log for internal logs
* Generates HTML report in /output/report
* Passes custom parameters: threads=100, rampup=10, duration=60

**How to Use Parameters Inside JMX**

To use parameters like -Jthreads=100, your test plan should use **${\_\_P(threads, defaultValue)}** in the thread group config.

**Tips**

* Use .jtl in CSV format for better parsing in external tools.
* Keep a separate .properties file and use -q for better config control.
* For CI/CD integration, combine with bash/batch + Jenkins/GitLab runners.

Here’s a **template shell script** to run your JMeter .jmx script in **Non-GUI mode** with customizable parameters and full logging/reporting support:

JMeter Non-GUI Execution Script (run\_jmeter.sh)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#!/bin/bash*

*# -----------------------*

*# JMeter Script Runner*

*# -----------------------*

*# === Configuration ===*

*JMETER\_PATH="/path/to/apache-jmeter-5.6.3/bin/jmeter" # Change this to your JMeter bin path*

*JMX\_FILE="test-plan.jmx" # Your JMX file name*

*RESULTS\_FILE="results.jtl" # JTL result file*

*LOG\_FILE="jmeter.log" # JMeter execution log*

*REPORT\_DIR="report" # Dashboard report folder*

*PROPERTIES\_FILE="custom.properties" # (Optional) Additional properties file*

*# === Dynamic Parameters ===*

*THREADS=100*

*RAMPUP=10*

*DURATION=60*

*# === Run Command ===*

*$JMETER\_PATH -n \*

*-t "$JMX\_FILE" \*

*-l "$RESULTS\_FILE" \*

*-j "$LOG\_FILE" \*

*-e \*

*-o "$REPORT\_DIR" \*

*-Jthreads="$THREADS" \*

*-Jrampup="$RAMPUP" \*

*-Jduration="$DURATION" \*

*-q "$PROPERTIES\_FILE"*

*# === Post-Run Message ===*

*echo "JMeter test completed. Report generated in: $REPORT\_DIR"*

**How to Use**

1. Save the above as run\_jmeter.sh
2. Make it executable:

Command: *chmod +x run\_jmeter.sh*

**Run the script**:

Command: *./run\_jmeter.sh*

**Important Notes**

* Make sure your JMX file uses ${\_\_P(threads)}, ${\_\_P(rampup)}, etc., for this script to override values.
* Adjust the path to jmeter binary and JMX file as needed.
* Add more -J parameters if your script uses more dynamic inputs.
* This script can be integrated into Jenkins, GitLab CI, or any other pipeline tool.

**Windows Batch Script Version (run\_jmeter.bat)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*@echo off*

*set JMETER\_BIN="C:\apache-jmeter-5.6.3\bin\jmeter.bat"*

*set TEST\_PLAN="C:\Users\ChandrasekarM\OneDrive - IBM\Documents\ICA\automation\automation\script\test-plan.jmx"*

*set RESULTS="C:\Results\results.jtl"*

*set LOG="C:\Results\jmeter.log"*

*set REPORT="C:\Results\report"*

*set THREADS=100*

*set RAMPUP=10*

*set DURATION=60*

*%JMETER\_BIN% -n -t %TEST\_PLAN% -l %RESULTS% -j %LOG% -e -o %REPORT% -Jthreads=%THREADS% -Jrampup=%RAMPUP% -Jduration=%DURATION%*

*echo Done. Report generated at %REPORT%*

*pause*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*