**Performance/Load Testing**

# Introduction

In this tutorial, you will learn what performance test is, how to use a performance test tool, how to design a performance test, and execute performance tests.

In particular, you will load test the target system [OpenMRS](https://openmrs.org/) web system. By completing this tutorial, you will know how to:

(1) Configure the load driver

(2) Execute the load test

(3) Analyze the results of this load test

## Definition

* Performance testing

Performance testing is a type of testing for determining the speed of a computer, network or device. It checks the performance of the components of a system by passing different parameters in different load scenarios.

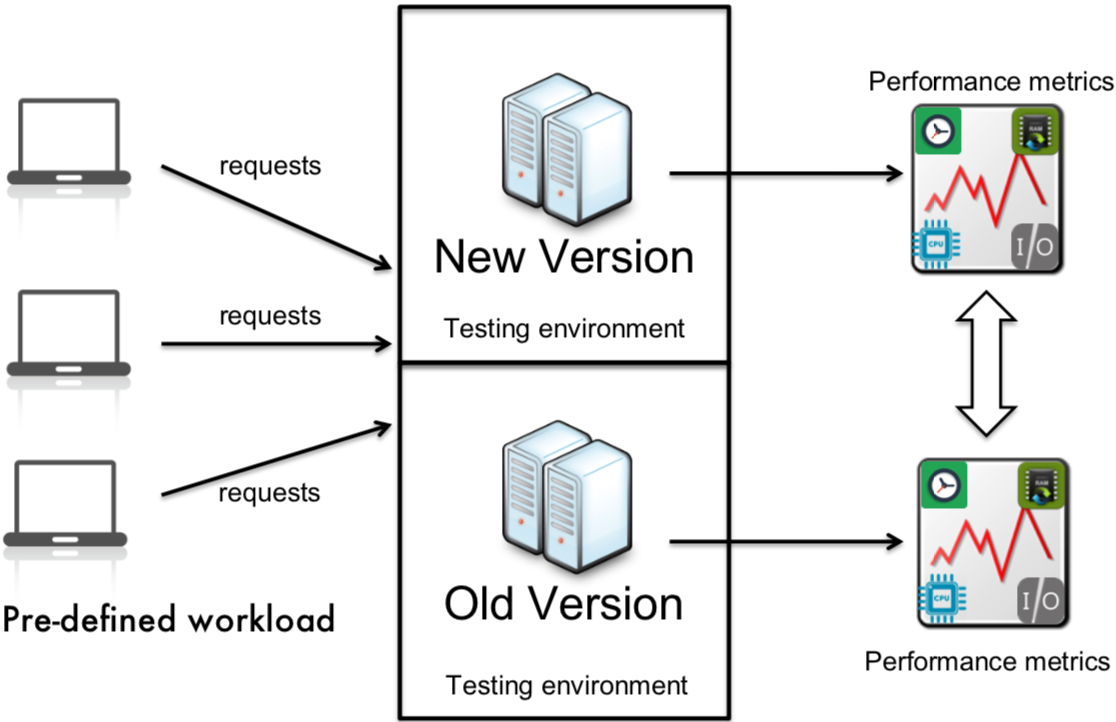
* Load testing

Load testing is the process that simulates actual user load on any application or website. It checks how the application behaves during normal and high loads. This type of testing is applied when a development project nears its completion.

## Mechanism

Performance/Load testing consists of three processes: 1) designing a proper load, 2) executing a load test, and 3) analyzing the results of a load test.

In practice, testers setup the testing environment and define the workload. Then the same requests are sent to exercise the system and the performance metrics are collected. Finally testers compare the two performance metrics and identify the performance regression. Below is a figure showing the mechanism of performance/load testing:



## Tools

There exist various tools that conduct a performance/load test:

* **JMeter**: JMeter is an open source tool that can be used for performance and load testing for analyzing and measuring the performance of a variety of services.
* **LoadRunner**: An enterprise performance testing version of Loadrunner and a platform enabled both global standardization.
* **ReadLine13**: is a load testing platform that brings the low cost power of the cloud to JMeter and other open source load testing tools.

In this tutorial, we will use **JMeter** for performance/load testing, and through the following sections, you will gain more hands-on knowledge and experience about performance/load testing.

## JMeter - Features

JMeter is a free open source load driver from Apache foundation. JMeter is a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.

<https://jmeter.apache.org/usermanual>

Apache JMeter features include load and performance testing many different applications/server/protocol types:

* Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
* SOAP / REST Web Services
* FTP
* Database via JDBC
* LDAP
* Message-oriented middleware (MOM) via JMS
* Mail - SMTP(S), POP3(S) and IMAP(S)
* Native commands or shell scripts
* TCP
* Java Objects

Look at its user manual for more details <https://jmeter.apache.org/usermanual/index.html>

# Requirements

Please make sure that you have JDK, Apache JMeter installed on your computer (try to install the same version).

* OS: macOS/Windows 10

Note: macOS 10.15 (Catalina) disables 32bit applications. Therefore, if your system is macOS 10.15, please install docker.

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>

* Java: version 1.8
* **Apache JMeter**: version 5.2.1
* **Psutil**: A system performance monitoring tool

<https://psutil.readthedocs.io/>

*Note: Only the above environment is tested by us, not sure if other versions work well.*

# Experiments

**Part A**

(Deploy OpenMRS Standalone system, and use Jmeter to conduct a steady workload)

Note: Only for macOS 10.15, please ignore the first to third steps.

If your system is macOS 10.15, please download the docker project [here](https://drive.google.com/file/d/1M5jSbEH5sFBeqtPJLDLTL5wLZRB0evfz/view?usp=sharing), and then unzip the demo.zip. Please go to the demo folder, and use the below commands to deploy OpenMRS:

$ docker-compose down -v

$ OPENMRS\_VERSION=2.5 docker-compose up

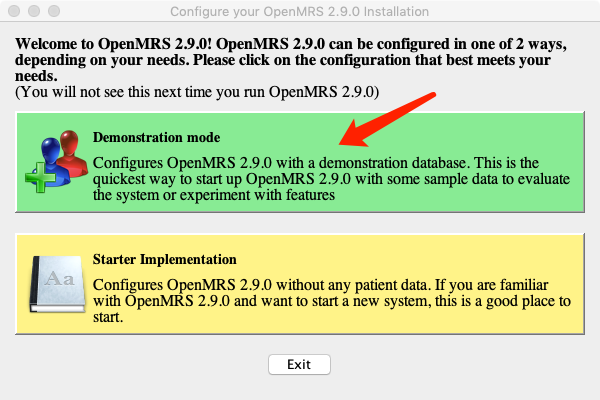
**Note: If your system is windows, please start *OpenMRS* in a terminal with Administrator role**

1. Download the project from [Google Drive](https://drive.google.com/file/d/1omyNuAwlA3eo3B2oa3iwhxrLNJsUAyOM/view?usp=sharing).
2. Unzip the downloaded file loadTests.zip, and unzip the two subfiles apache-jmeter-5.2.1.zip, and OpenMRS-2.9.0.zip.
3. Start OpenMRS Standalone system (*This may take a few minutes*)

* Navigate to the decompressed folder OpenMRS-2.9.0 and run the following command:

java -jar openmrs-standalone.jar (mac os)

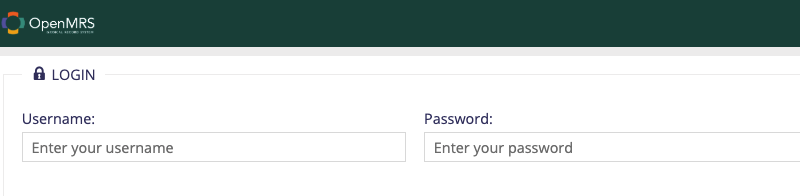
java -jar openmrs-standalone.jar (windows, running with Administrator role)



1. Verify OpenMRS server

Go to http://localhost:8081/openmrs-standalone in your browser, and login as “admin” and “Admin123” for the user ID and password, respectively.

Note: macOS 10.15, please go to <http://localhost:8081/openmrs>, the same for the latter steps.



1. Launch JMeter:

Open another terminal, and go to the JMeter folder and launch jmeter

$./bin/jmeter (mac os)

Double click “ApacheJMeter.jar” (windows)

**Note**: Download the jmeter script steady-workload from [here](https://drive.google.com/file/d/10xiJ9K0In_MwYgBuOkPrWfuzrOXGfeI3/view?usp=sharing). The script is generated by the steps from step 6 to step 13.

1. Create thread group “Steady-Workload”, and setup the default HTTP request, HTTP header, and HTTP request authorization

* Right click Test Plan -> Add -> Threads (Users) -> Thread Group -> Name this thread group as “Steady-Workload”
* Right click Steady-Workload -> Add -> Config Element -> HTTP Request Defaults

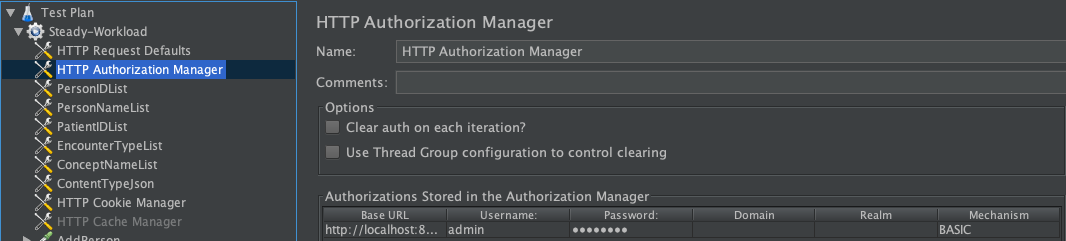
Enter “localhost” as the server name, port number as “8081”

* Right click Steady-Workload -> Add -> Config Element -> HTTP Header Manager

Config header by adding name “Content-Type”, and value “application/json”

* Right click Steady-Workload -> Add -> Config Element -> HTTP Authorization Manager

Config Base URL, Username, and Password.

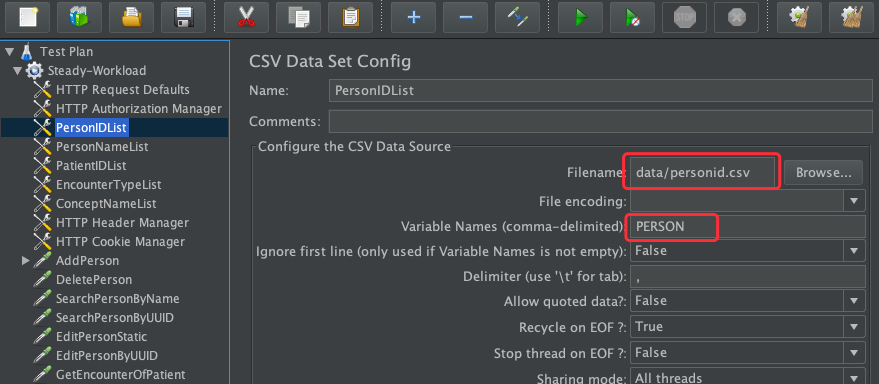


1. Import CSV data under the Steady-Workload:

* Right click Steady-Workload -> Add -> Config Element -> CSV Data Set Config

Enter the proper path for the csv file and variable names as shown below

* Import the remaining CSV data from the folder “apache-jmeter-5.2.1/data”.



1. Add multiple HTTP Requests to represent different user behavior:

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “AddPerson”, and input “openmrs-standalone/ws/rest/v1/person” into Path, and select *POST* method, and Config Body Data:

{"names":[{"givenName": "inputgivenname",

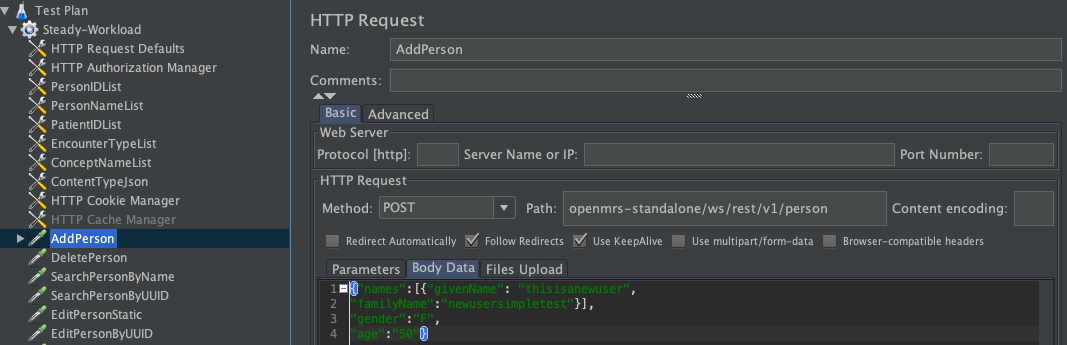
"familyName":"inputfamilyname"}],

"gender":"F",

"age":"50"}

Note: you can find the path and configuration from this link:

<https://wiki.openmrs.org/display/docs/REST+Web+Service+Resources+in+OpenMRS+1.9>



* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “DeletePerson”, and select *DELETE* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “SearchPersonByName”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “SearchPersonByUUID”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “EditPersonStatic”, and select *POST* method, and input the path, and config Body data

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “EditPersonByUUID”, and select *POST* method, and input the path, and config Body data

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “GetEncounterOfPatient”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “GetEncounterByType”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

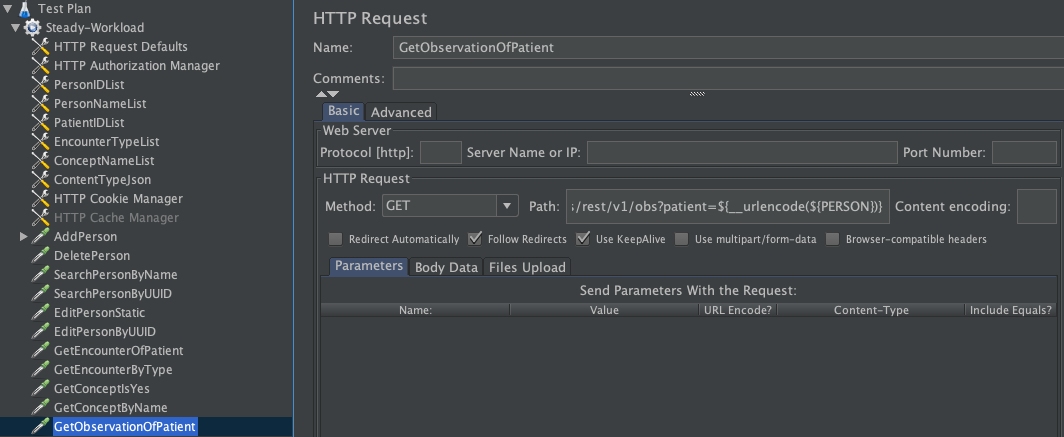
Name it as “GetConceptIsYes”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “GetConceptByName”, and select *GET* method, and input the path

* Right click Steady-Workload -> Add -> Sampler -> HTTP Request

Name it as “GetObservationOfPatient”, and select *GET* method, and input the path



1. Add HTTP Cookie Manager to manage cookies for each session:

* Right click Steady-Workload -> Add -> Config Elements -> HTTP Cookie Manager

Check “Clear cookies each iteration?”

1. Add listeners under the thread group to view the result:

* Right click Steady-Workload -> Add -> Listener -> View Results Tree
* Right click Steady-Workload -> Add -> Listener -> Summary Report

1. Configure the number of threads and iterations/test duration:

Click Steady-Workload, and set “Number of Threads” to be 2, and ramp up period to be 5 seconds, and set “Loop Count” to be 5

1. Now we are ready to record our test scenario for “Steady-Workload”:

* Right click Steady-Workload -> Start

1. Save the test plan (File -> Save) to a location of your choice.

**Part B**

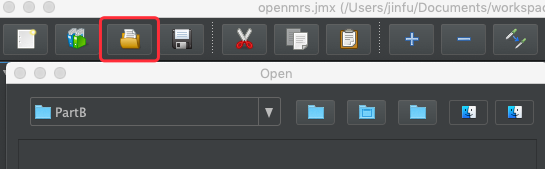
Use Jmeter to conduct a random workload using logic controllors

1. Launch JMeter

Go to the JMeter folder and run ./bin/jmeter

$./bin/jmeter

1. Import Jmeter script from Part A into Jmeter.



Name the thread group as “Random-Workload”

<https://jmeter.apache.org/usermanual/test_plan.html#logic_controller>

**Random Order Controller** provides functionality to run all user request in random

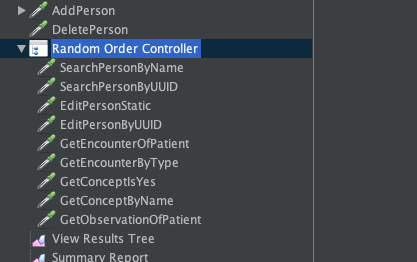
order for each loop.

**Random Controller** provides functionality to run user request in random order for each loop i.e. One random user request in each loop.

1. Add Random Order Controller under the Random-Workload:

* Right click Random-Workload -> Add -> Logic Controller -> Random Order Controller

Put HTTP Requests under the Random Order Controller.



1. Add Random controller under the Random-Workload
   1. Right click Random-Workload -> Add -> Logic Controller -> Random Controller

Put HTTP Requests under the Random Order Controller.

1. Now we are ready to record our test scenario for “Random-Workload”:

* Right click Random-Workload -> Start

1. Save the test plan (File -> Save) to a location of your choice.

Note: You can download the jmeter script random-workload from . The script is generated based on the above steps.

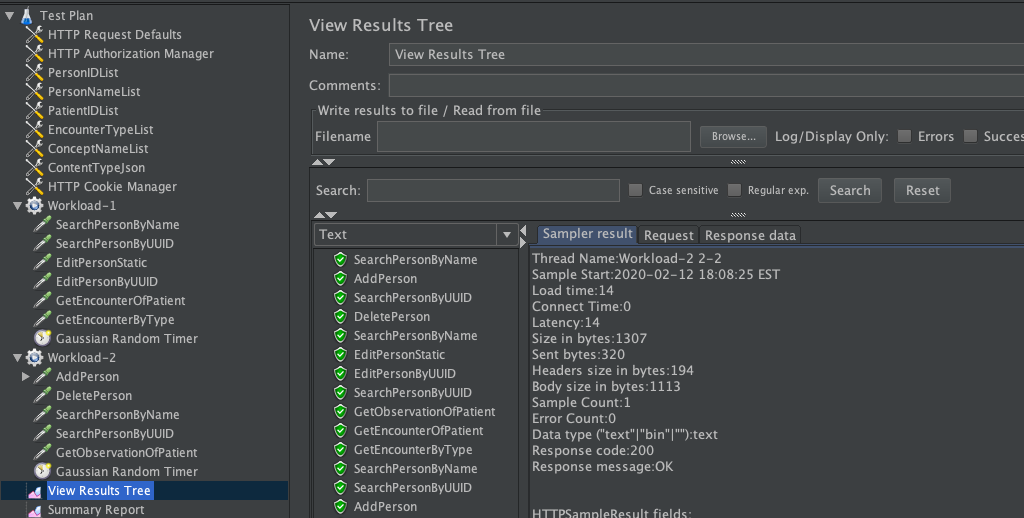
**Part C**

Use JMeter to load test OpenMRS by the below mix two workloads:

Workload-1: SearchPerson->EditPerson->GetEncounter…

Workload-2: AddPerson->DeletePerson->SearchPerson->GetObservationOfPatient..

1. Launch JMeter
2. Create two thread groups “Workload-1” and “Workload-2”
3. For each Workload, repeat the steps of Part A. The difference is that you need to create different HTTP requests based on the Workload-1 or Workload-2
4. Now we are ready to record our Test Plan
5. Save the test plan (File -> Save) to a location of your choice.

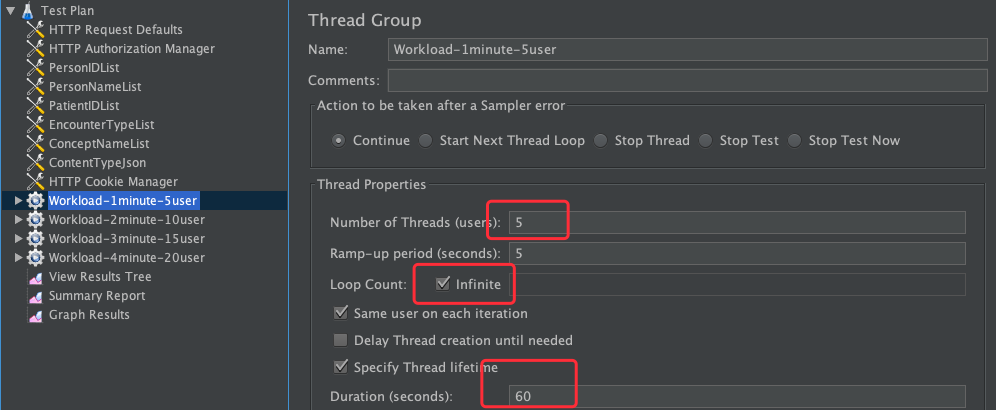


**Part D**

(Use JMeter to load test OpenMRS by **varied** workloads using **CLI** mode, and monitor the performance of the *openmrs-standalone.jar* process using tool ***psutil***:

Varied workloads: The first minute, 5 users access OpenMRS. The second minute 10 users access OpenMRS. The third minute 15 users access OpenMRS. The fourth minute 20 users access OpenMRS.

1. This part can be done creating four thread groups (one thread group represents one type of workload) under Test Plan.

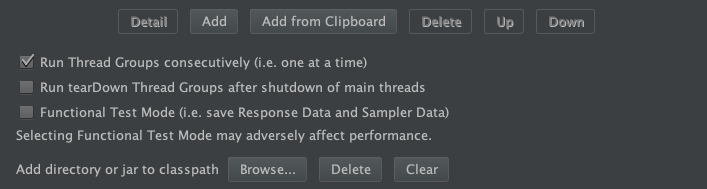


For each workload, specify the number of users.

1. Config the order of execution of such four workloads:

Click Test Plan

Check “Run Thread Groups consecutively”



1. Save the test plan (File -> Save, vary-workload.jmx) to a location of your choice.
2. Conduct varied workload using JMeter CLI mode:

Open a terminal, and go to JMeter folder

$ jmeter -n -t vary-workload.jmx -l testresults.jtl

(-n: It specifies JMeter is to run in non-gui mode

-t: Name of JMX file that contains the Test Plan

-l: Name of JTL(JMeter text logs) file to log results

-j: Name of JMeter run log file )

1. Capture CPU performance data

Get the process id of process openmrs-standalone.jar, and then use **psutil** to capture CPU performance data. (Example [here](https://github.com/shaoweiwang2010/SEII-load-testing-tutorial/blob/main/perfMonitor.py))