**Performance Testing:**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload.

* Speed
* Scalability
* Stability
* Reliability

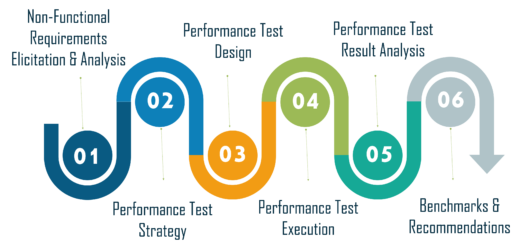
**Speed:**   
It determines whether the software product responds rapidly.

**Scalability:**   
It determines amount of load the software product can handle at a time.

**Stability:**   
It determines whether the software product is stable in case of varying workloads.

**Reliability:**   
It determines whether the software product is secure or not.

**Performance testing life cycle**



**Non-Functional Requirements Elicitation and Analysis**

It is one of the most important and critical steps to understand the non-functional requirements in PTLC. It helps to evaluate the degree of compliance with non-functional needs.

**Performance Test Strategy**

The second defines how to approach Performance Testing for the identified critical scenarios. You need to address the kind of performance testing and the tools required

**Performance Test Design**

This phase involves the script generation using the identified testing tool in a dedicated environment. The script enhancements are needed to be done and unit tested

**Performance Test Execution**

The next phase is dedicated to the test engineers who design scenarios based on identified workload and load the system with concurrent virtual users.

**Performance Test Result Analysis**

In this phase, the collected log files are analyzed and reviewed by the experienced test engineers. Tuning recommendation will be given if any conflicts are identified.

**Benchmark & Recommendations**

This is the last phase in PTLC which involves benchmarking and providing a recommendation to the client.

**Performance testing steps:**

Step 1: Identify the Testing Environment

The testing environment or the test best is where all the magic happens. Identify the testing environment and know [what testing tools are available](https://www.simplilearn.com/best-automation-testing-tools-for-software-development-article) at your disposal. Understand the details of all the hardware, software and different network configurations ahead of time.

Step 2: Identify the Performance Metrics

In addition to the general performance metrics such as response time, throughput and constraints, it is also important to identify the performance success criteria. Oftentimes, there may not be a wide enough variety of performance benchmarks that you can identify. You can find similar applications which are already successful in order to set performance goals.

Step 3: Plan and Design Performance Tests

Identify a number of key scenarios by taking into account user variability, test data, and plan performance. This is required to simulate a variety of use cases and outline what metrics will be gathered.

Step 4: Configure the Test Environment

Arrange all the necessary testing tools and monitoring resources to prepare the testing environment before execution.

Step 5: Implement the Test Design

Design all the performance tests according to your performance criteria and metrics.

Step 6: Run the Tests

Execute and run the performance tests. Also, capture and monitor all the test data that is generated.

Step 7: Analyze, Tune and Retest

After every performance test, analyze the finding and fine tune the test again to see an increase or decrease in performance. Run the tests again using the same or different parameters

**Performance testing work flow**

1. **Requirement Analysis/Gathering:** The performance team interacts with the client for identification and gathering of requirements.
2. **POC/Tool selection:** Once the key functionality is identified, POC (Proof Of Concept – which is a sort of demonstration of the real-time activity but in a limited sense) is done with the available tools.
3. **Performance Test Plan & Design:** Test Planning involves information on how the performance test is going to take place – test environment, workload, hardware, etc.
4. **Performance Test Development:**

Use cases are created for the functionality identified in the test plan as the scope of PT.

These use cases are shared with the client for their approval. This is to make sure the script will be recorded with the correct steps.

1. **Performance Test Modeling:** Performance Load Model is created for the test execution. The main aim of this step is to validate whether the given Performance metrics (provided by clients) are achieved during the test or not.
2. **Test Execution:** The scenario is designed according to the Load Model in Controller or Performance Center but the initial tests are not executed with maximum users that are in the Load model.
3. **Test Results Analysis:** Test results are the most important deliverable for the performance tester. This is where we can prove the ROI (Return on Investment) and productivity that a performance testing effort can provide.
4. **Report:** Test results should be simplified so the conclusion is clearer and should not need any derivation. Development Team needs more information on analysis, comparison of results, and details of how the results were obtained.

Performance testing tools

Apache JMeter

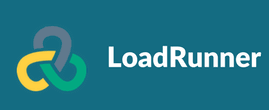
It is used to test the performance of both static and dynamic resources and dynamic web applications. This tool is completely designed on the JAVA application to load the functional test behavior and measure the performance of the application. It is an open-source tool that facilitates users or developers to use the source code for the development of other applications.



LoadRunner

It is one of the most powerful tools of performance testing, which is used to support the performance testing for the extensive range of protocols, number of technologies, and application environments.

It quickly identifies the most common causes of performance issues. And also accurately predict the application scalability and capacity.



LoadNinja

LoadNinja is powered by SmartBear. With the help of this tool, product teams and the test engineer will construct the application with more concentration rather than writing the load testing scripts. We can keep track of user interactions, find the performance issues directly, and debug them in real-time. It will change the load emulators with the real browsers.



WebLOAD

WebLOAD testing tool is used to test the test application with the help of load testing, performance testing, and stress testing. For the authentication of web and mobile applications, the WebLOAD tool combines the performance, scalability, and integrity as a single process. It will support the multi-protocols such as HTTPS, XML, HTTP, and so on, which helps us to control the load of the large number of users.



NeoLoad

Neotys develop a testing tool which is called NeoLoad. The NeoLoad is used to test the performance test scenarios. With the help of NeoLoad, we can find the bottleneck areas in the web and the mobile app development process.

Difference between jmeter and loadrunner

|  |  |  |
| --- | --- | --- |
| **Basis of Comparison** | **JMeter** | **Loadrunner** |
| **Owner** | Apache owns the JMeter software tool. | The Loadrunner software tool is owned by Hewlett Packard (HP). |
| **Price** | The JMeter software testing tool is open-source software that can be easily downloaded and can be used for functional performance testing of web applications. | The Loadrunner software testing tool is not open-source software, and the user has to buy to do functional performance testing of web applications. There are different editions available for the software that includes community edition, enterprise edition. |
| **Integrations** | The JMeter software can integrate with other applications and business systems that include Jenkins, JSUnit, BlazeMeter, Maven, CloudGen, Visual Studio, Dynatrace, Meliora TestLab, and many more other applications. | The Loadrunner software can integrate with other applications and business systems that include Jenkins, DynaTrace, UFT, SiteScope, AppDynamics, Selenium, HP Diagnostics, and Visual studio. The Loadrunner software tool can be easily integrated with the above applications. |
| **Devices support** | The JMeter software supports Linux, Windows, Web-based and Mac and can be used to test the software performance. | The LoadRunner software supports Windows, Web-based and Mac and can be used to test the software performance. |
| **Features** | Several features are offered by JMeter software. The features include compatible with REST/SOAP, FTP, HTTP, HTTPS, LDAP, TCP, Mail (SMTP), database using JDBC, shell scripts. The software is compatible with all the above. And the software uses javascript as the programming language. | There are several features offered by LoadRunner software. Some of them are the support of cloud testing, continuous rigorous testing, mobile testing, support to a variety of applications, support for root cause analytics, interactive user transaction simulation and many more. The software comes up with all the above-mentioned features. |
| **GUI** | The user interface of the JMeter software is less efficient when it is compared to LoadRunner software. | The LoadRunner software comes up with an efficient graphical user interface when it is compared to JMeter software. |
| **Performance** | The performance of JMeter is less compared to the LoadRunner. Small bugs and threats can be identified by this tool, but it is inefficient to detect threats in large scale applications. | The LoadRunner is more efficient and faster to detect the bus present in the application compared to JMeter software. All the system-level performance information can be collected using the LoadRunner software. |

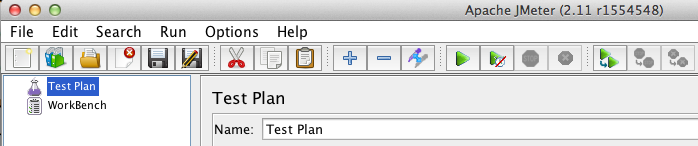
Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications.  
It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.

Apache JMeter features include:

* Ability to load and performance test many different applications/server/protocol types:
  + Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
  + SOAP / REST Webservices
  + 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

**Building a Basic Test Plan**

After starting JMeter, you should see the graphical user interface with an empty *Test Plan*:



A test plan is composed of a sequence of test components that determine how the load test will be simulated. We will explain the how some of these components can be used as we add them into our test plan.

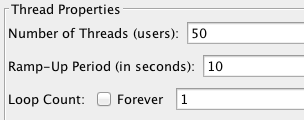
**Add a Thread Group**

First, add a *Thread Group* to *Test Plan*:

1. Right-click on *Test Plan*
2. Mouse over *Add >*
3. Mouse over *Threads (Users) >*
4. Click on *Thread Group*

The *Thread Group* has three particularly important properties influence the load test:

* **Number of Threads (users)**: The number of users that JMeter will attempt to simulate. Set this to **50**
* **Ramp-Up Period (in seconds)**: The duration of time that JMeter will distribute the start of the threads over. Set this to **10**.
* **Loop Count**: The number of times to execute the test. Leave this set to **1**.

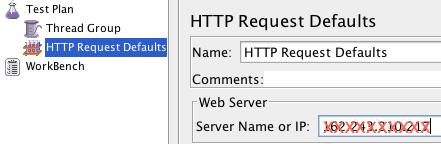


**Add an HTTP Request Defaults**

The *HTTP Request Defaults* Config Element is used to set default values for HTTP Requests in our test plan. This is particularly useful if we want to send multiple HTTP requests to the same server as part of our test. Now let’s add *HTTP Request Defaults* to *Thread Group*:

1. Select *Thread Group*, then Right-click it
2. Mouse over *Add >*
3. Mouse over *Config Element >*
4. Click on *HTTP Request Defaults*

In HTTP Request Defaults, under the Web Server section, fill in the *Server Name or IP* field with the name or IP address of the web server you want to test. Setting the server here makes it the default server for the rest of the items in this thread group.



**Add an HTTP Cookie Manager**

If your web server uses cookies, you can add support for cookies by adding an HTTP Cookie Manager to the Thread Group:

1. Select *Thread Group*, then Right-click it
2. Mouse over *Add >*
3. Mouse over *Config Element >*
4. Click on *HTTP Cookie Manager*

**Add an HTTP Request Sampler**

Now you will want to add an *HTTP Request* sampler to *Thread Group*, which represents a page request that each thread (user) will access:

1. Select *Thread Group*, then Right-click it
2. Mouse over *Add >*
3. Mouse over *Sampler >*
4. Click on *HTTP Request*

In HTTP Request, under the HTTP Request section, fill in the *Path* with the item that you want each thread (user) to request. We will set this to /, so each thread will access the homepage of our server. Note that you do not need to specify the server in this item because it was already specified in the HTTP Request Defaults item.

**Jmeter**

**1 – Verify System Requirements**

Before using JMeter, you must ensure that a current version of Java is installed on your system. The latest version requires Java 8 or higher. It is best practice to always use the most current version of JMeter as the software is constantly being improved and updated.

**1-1 Download & Install JMeter**

1. Visit the Apache JMeter website.
2. Select *Download Releases* from the left-hand side of the page, under *Download*.
3. Select from JMeter **Binaries** or JMeter **Source** files and it will begin downloading.
4. Unzip and extract the file into the location/folder on your machine once the file is downloaded.

Note: JMeter can be started in three different modes. For the purposes of this **JMeter load testing** tutorial, we will focus on starting in GUI mode.

* GUI mode. GUI mode is used to build test scripts and debugging.
* Command Line Interface (CLI) Mode. CLI mode is used for load testing.
* Server Mode. Server mode is used for distributed load testing on remote nodes.

**1-2 Launch JMeter**

1. Navigate to the location where you saved the JMeter directory.
2. Select the */bin* folder
3. Locate and select the *jmeter.bat* file
4. Wait for the interface to begin loading (GUI mode). Once it has finished loading, we can begin creating our test plan.

**2 – Create JMeter Test Plan**

Once JMeter has completely loaded, you can begin the process of creating the test plan. The test plan includes all of the elements that go into building the test plan. We will talk more about these elements in the coming sections.

**2-1 Set Up JMeter Test Plan**

1. Select *File* from the top-left corner to create a test plan from scratch.
2. Select *New*.
3. Select the *New* icon from the toolbar (alternative method). A new test plan will be displayed on the screen.
4. Configure the test plan elements.

JMeter also allows users to build test plans from templates that contain the necessary test elements. There are numerous options available depending on your **JMeter load testing** requirements. Listed below are just a few of the options that are available to users.

* BeanShell Sampler
* Building a SOAP WebService Test Plan
* Building an Advanced Web Test Plan
* Building an FTP Test Plan
* Functional Testing Plan
* Building Webservice Test Plan

**2-2 Create a Thread Group (Users)**

1. Right-click on *Test Plan* from the Test Plan Tree View window.
2. Select *Add*.
3. Hover over the *Threads (Users)* option. Another window will appear to the right.
4. Select *Thread Group*. The **Thread Group** properties will appear in the Editor window.
5. Configure *Thread Properties*: **Number of Threads (Users), Ramp-Up Period,**and**Loop Count.**
6. Add a Thread Group Name and specific test plan notes in the *Comments* field, if necessary.

**3 – Configure Samplers**

Samplers are the actual requests to the server. For example, HTTP samplers are used for a website, application, or API load test. Other Samplers include FTP, SMTP, Java, JDBC, and SOAP requests. To create a request Sampler, follow the steps below. For the purposes of this **JMeter load testing** tutorial, we will add an HTTP sampler.

**3-1 Adding a Sampler**

* 1. Right-click on *Thread Group* under your Test Plan window
  2. Hover your cursor over *Add*. A window will open to the right of the *Add* selection.
  3. Hover your cursor over *Sampler*. A drop-down menu will appear will all the Sampler options.
  4. Select the *HTTP Request Sampler*. The Editor window will be displayed.

From here, you can configure and customize your HTTP Sampler.

* 1. Add a name for your HTTP Request sampler in the *Name* field. Optionally, you can add comment for this specific to this Sampler in the *Comments* field.
  2. Add the specific protocol (HTTP or HTTPS).
  3. Enter the *Server Name* or *IP address*. If necessary, you can also enter the *Port Number*.
  4. The default request type is set to *GET*, but you can also change the request type (*POST, HEAD, PUT, PATCH,* etc., if necessary.)
  5. Lastly, if you are testing a specific URL path, add that path in the *Path* field.

There are also additional configuration options listed under the *Advanced* tab. Once you have configured the Sampler, you can select *Save*.

**3-2 Add Listeners**

1. Right-click*Thread Group*.
2. Place your cursor over Add. Hover your cursor over *Add*. A window will open to the right of the *Add* selection.
3. Select *Listener*. Here you will see a list of Listeners. Listeners are what allows you to review the test results and reports. For the purposes of this tutorial, we will use the *View Results Tree* Listener.
4. Select *View Results Tree*Listener. Note: Multiple Listeners can be added to a load test.

**4 – Run Load Test and Analyze Results**

Once you have finished configuring your Thread Group, Samplers, and Listeners, the test is ready to run. There are a couple of different ways to run, or execute load test

**4-1 Run JMeter Test**

* 1. Save your test plan by selecting *File > Save* from the top-left corner.
  2. Select *Start* from the *Run* menu.
  3. Click the green arrow Run icon from the toolbar. This is an alternative method to begin the test.
  4. Wait for the test to run and finish.
  5. View Results.

You will see the test results appear below in the Test Plan window, along with various metrics.

* 1. The **Sample #** represent the number of runs, or users.
  2. **Start Time** for each Thread.
  3. **Thread Name** for each run and user.
  4. **Sample Time** is the time taken for each request.
  5. **Status**. Green checkmark indicates it has passed. A red checkmark indicates a failure.
  6. Additional metrics, such as**Bytes, Sent Bytes,**and**Latency** are also included.