Performance Testing

Speed, Stability and Scalability

Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device.

Performance testing is the process by which software is
tested to determine the current system performance.

Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and is also known as ***"Perf Testing"***.

What is Performance Testing?

- Performance Testing is defined as a type of Non-Functional software testing to ensure software applications will perform well under expected workload.
- The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks.
- The focus of Performance Testing is checking a software program's
  - Speed
  - Scalability
  - Stability
- Performance Testing is popularly called "Perf Testing".

- Load testing
- Stress testing
- Endurance testing
- Spike testing
- Volume testing

- **Load testing –** checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing –** tests the software's reaction to sudden large spikes in the load generated by users.
- **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
- **Scalability testing** – The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.


Load Testing
Stress Testing
Endurance Testing
Performance Testing
Spike Testing
Volume Testing
Scalability Testing

Load Test

To test the performance and behavior at peak load (or speed or configuration) ex. 100 users is the limit and testing the system by applying 100 user is called as Load Testing.

- Am I ok or not?
- Simulate expected conditions.
- Important/Critical transactions.
- Overall performance.
- SLA defined by owner.

Stress Test

It tests stress limits of a system (maximum number of users, peak demands, etc) ex. beyond 100 users and towards the system crash is called as Stress testing.

- Where am I weak?
- Unexpected conditions.
- Extreme load.
- Modified scripts.
- Find break points.
- Vital for some scenarios.

Scalability Test

- How much can I grow?
- Same app, same env.
- Future expectations.
- Max acceptable level.
- SLA could be changed.

VOLUME TESTING

Volume testing means testing the application for large volume for data is called volume testing. This is mainly conducted to check the memory leaks and capacity of the server handling huge volume of data.

SECURITY TESTING

Security testing is a process to determine that an information system protects data and maintains functionality as intended.

RECOVERY TESTING

Testing how well a system recovers from crashes, hardware failures. It tests system's response to presence of errors or loss of data.

Virtual Users (VUs):

Start: 5

Time <= 20 Sec

Incremented by: 5

Maximum: 200

Think Time: 5 sec

Test Goals

Max Response

Test Script:

One typical user from login through completion.

Performance Testing Metrics

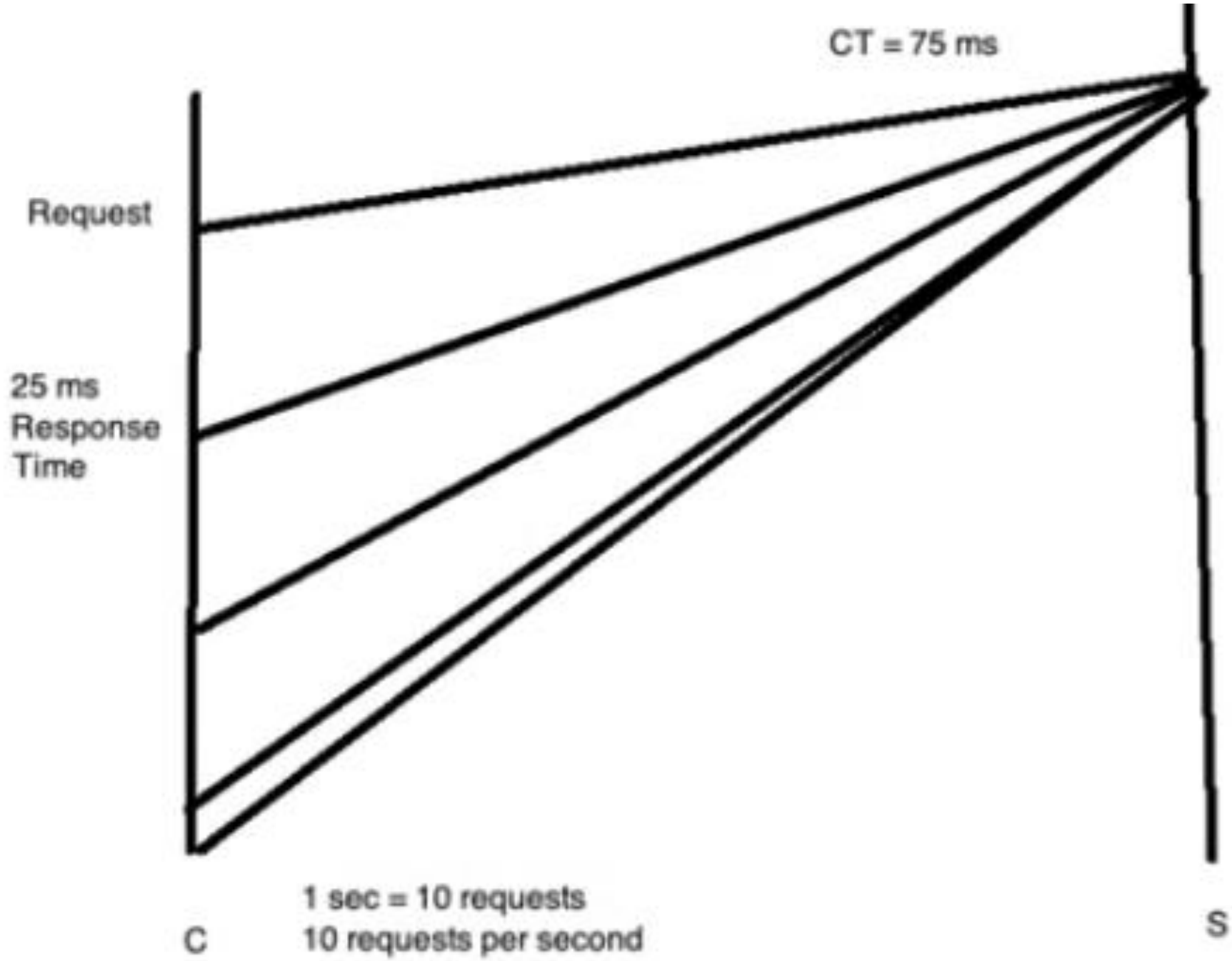
Committed memory	The amount of virtual memory used
Response time	Time between user's request and the first response character
Throughput	Rate of requests received per second by a network
Hit ratios	Number of SQL statements that are handled by cached data

$$\frac{\text{Number of cache hits}}{(\text{Number of cache hits} + \text{Number of cache misses})} = \text{Cache hit ratio}$$

Common Performance Problems

- Long Load time
- Poor response time
- Poor scalability
- Bottlenecking
- **Some common performance bottlenecks** are
  - CPU utilization
  - Memory utilization
  - Network utilization
  - Operating System limitations
  - Disk usage

- **Processor Usage -** an amount of time processor spends executing non-idle threads.
- **Memory use -** amount of physical memory available to processes on a computer.
- **Disk time -** amount of time disk is busy executing a read or write request.
- **Bandwidth -** shows the bits per second used by a network interface.
- **Private bytes -** number of bytes a process has allocated that can't be shared amongst other processes. These are used to measure memory leaks and usage.
- **Committed memory -** amount of virtual memory used.
- **Response time -** time from when a user enters a request until the first character of the response is received.
- **Throughput -** rate a computer or network receives requests per second.
- **Amount of connection pooling -** the number of user requests that are met by pooled connections. The more requests met by connections in the pool, the better the performance will be.
- **Maximum active sessions -** the maximum number of sessions that can be active at once.
- **Hit ratios -** This has to do with the number of SQL statements that are handled by cached data instead of expensive I/O operations. This is a good place to start for solving bottlenecking issues.
- **Hits per second -** the no. of hits on a web server during each second of a load test.
- **Thread counts -** An applications health can be measured by the no. of threads that are running and currently active.


CT = 75 ms
Request
25 ms
Response
Time
1 sec = 10 requests
10 requests per second
C
S

- **Bottleneck -** *Used to describe a single part of a system that prevents further processing or significantly degrades the performance of the system as a whole*
- **Capacity -** *The degree to which a system can perform data processing until performance degrades. For example, the number of new customers being added to a database*
- **Concurrency -** *Normally this means the number of simultaneous virtual users driving transactions across the User Journeys in a given performance test scenario, but can also mean the number of transactions synchronised to happen at exactly the same point.*
- **Key Performance Indicators (KPI) -** *The set of targets which set the expected performance targets within the Production system. These may include page response times (e.g. 99% of pages loaded <= 2 seconds), user concurrency, batch processing times, data throughput volumes, transaction failure rates, and underlying infrastructure behaviour (e.g. Maximum Average CPU used, Minimum Free Memory Available, thresholds for remaining physical storage / disk usage, logging space etc)*

Example Performance Test Cases

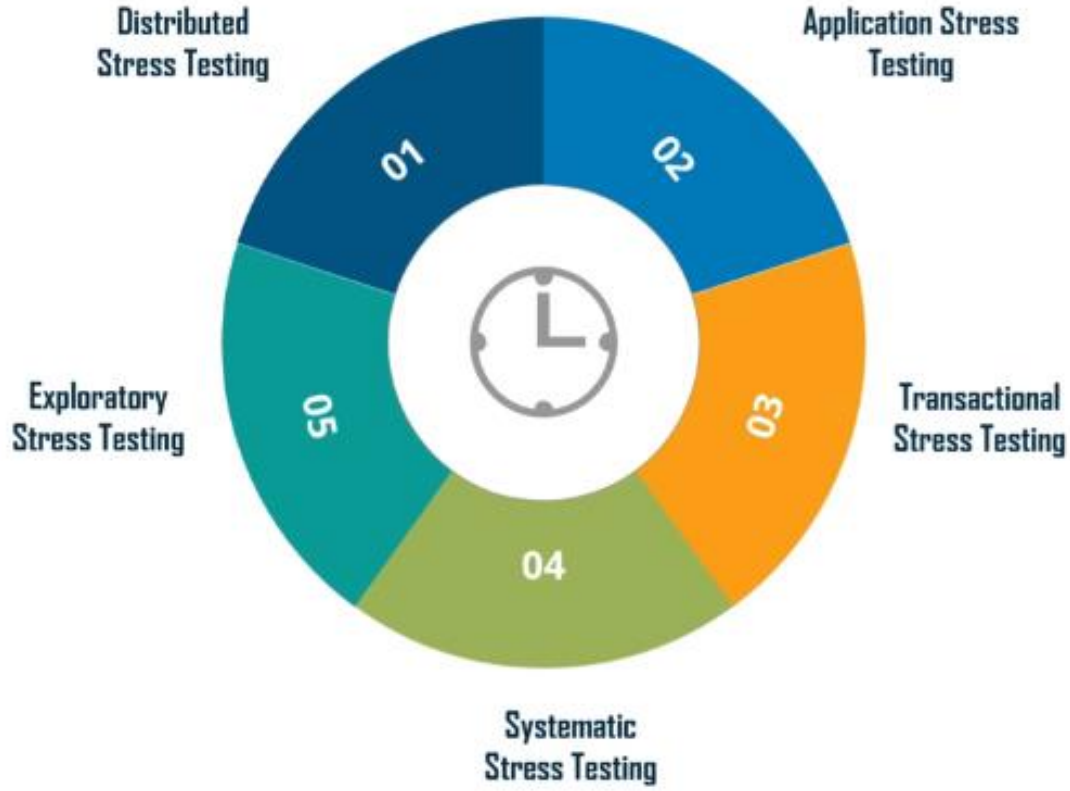
- Verify response time is not more than 4 secs when 1000 users access the website simultaneously.
- Verify response time of the Application Under Load is within an acceptable range when the network connectivity is slow
- Check the maximum number of users that the application can handle before it crashes.
- Check database execution time when 500 records are read/written simultaneously.
- Check CPU and memory usage of the application and the database server under peak load conditions
- Verify response time of the application under low, normal, moderate and heavy load conditions.

Performance Test Tools

Micro Focus
LoadRunner

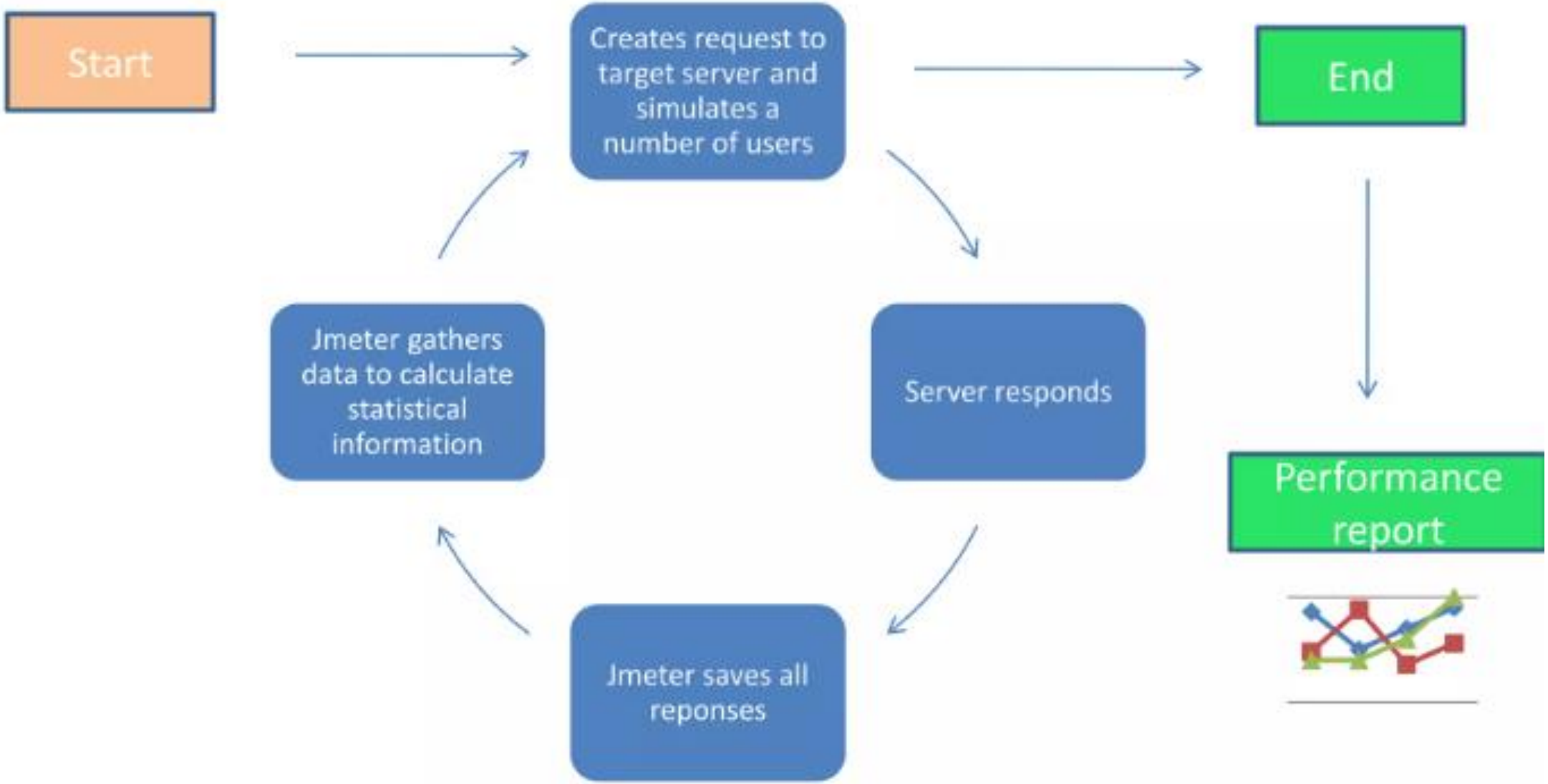
What is JMeter?

- The Apache JMeter is an open-source purely Java based software
- It is used to test load testing functional behavior and measuring performance
- Initially, JMeter was developed to test applications, but now has expanded to other test functions


Distributed
Stress Testing
Application Stress
Testing
01
02
Exploratory
Stress Testing
05
03
Transactional
Stress Testing
04
Systematic
Stress Testing


Planning
Create Automation
Script Execution
Result Analysis
Optimization

Jmeter simulates a group of users sending requests to a target server , and returns statistics that show the performance of the target server/application through graphical diagrams. This is a basic description of how jmeter works.



Open source license	Friendly GUI	Platform independent
Full multi-threading framework	Visualize Test Result	Easy installation
Highly extensible	Unlimited testing capabilities	Support multi protocol

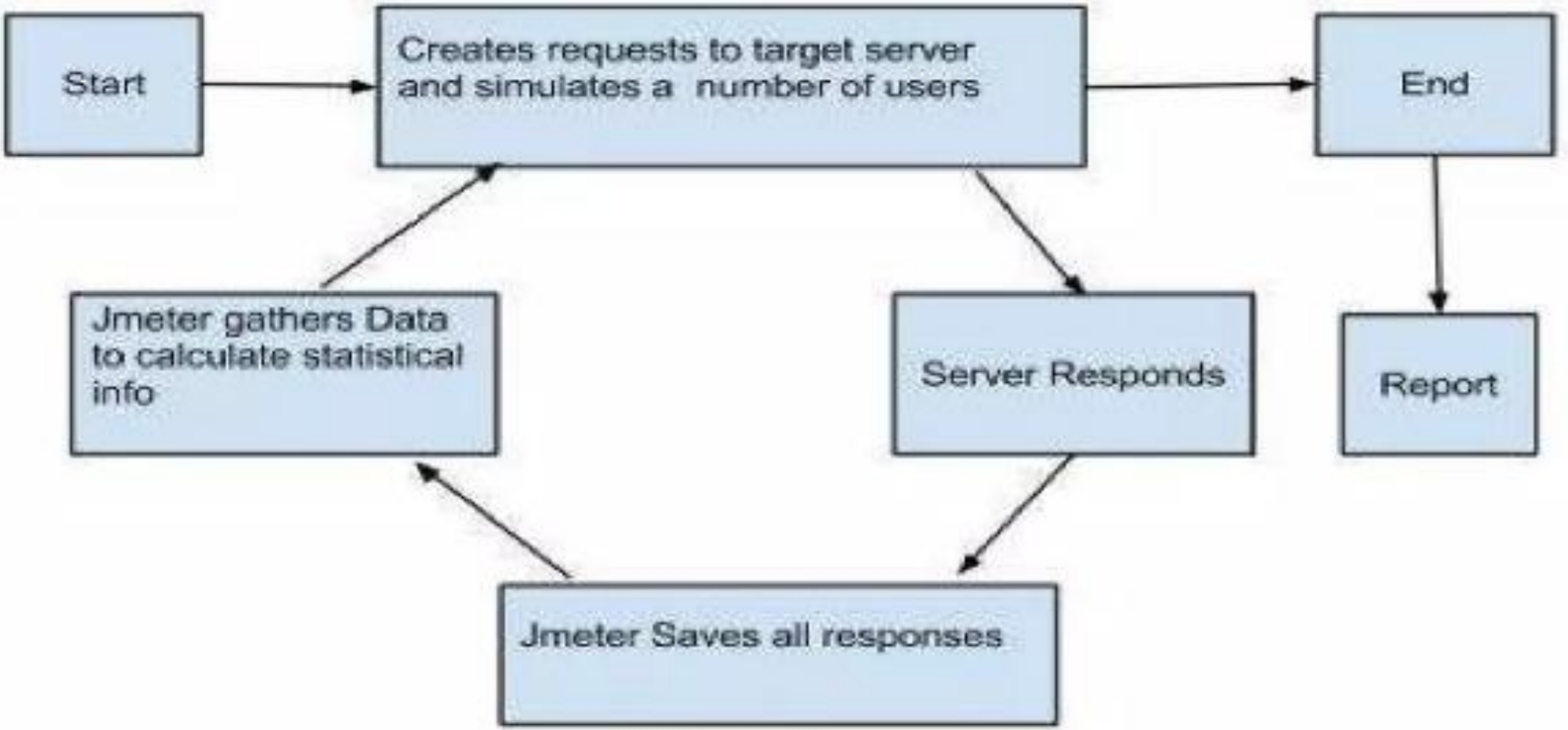
Demo: JMeter Load Testing

jMeter is an Open Source testing software. It is 100% pure Java application for load and performance testing.

jMeter is designed to cover categories of tests like load, functional, performance, regression, etc.

Protocol Support

- Web - HTTP, HTTPS
- SOAP / REST
- 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



How jMeter works?

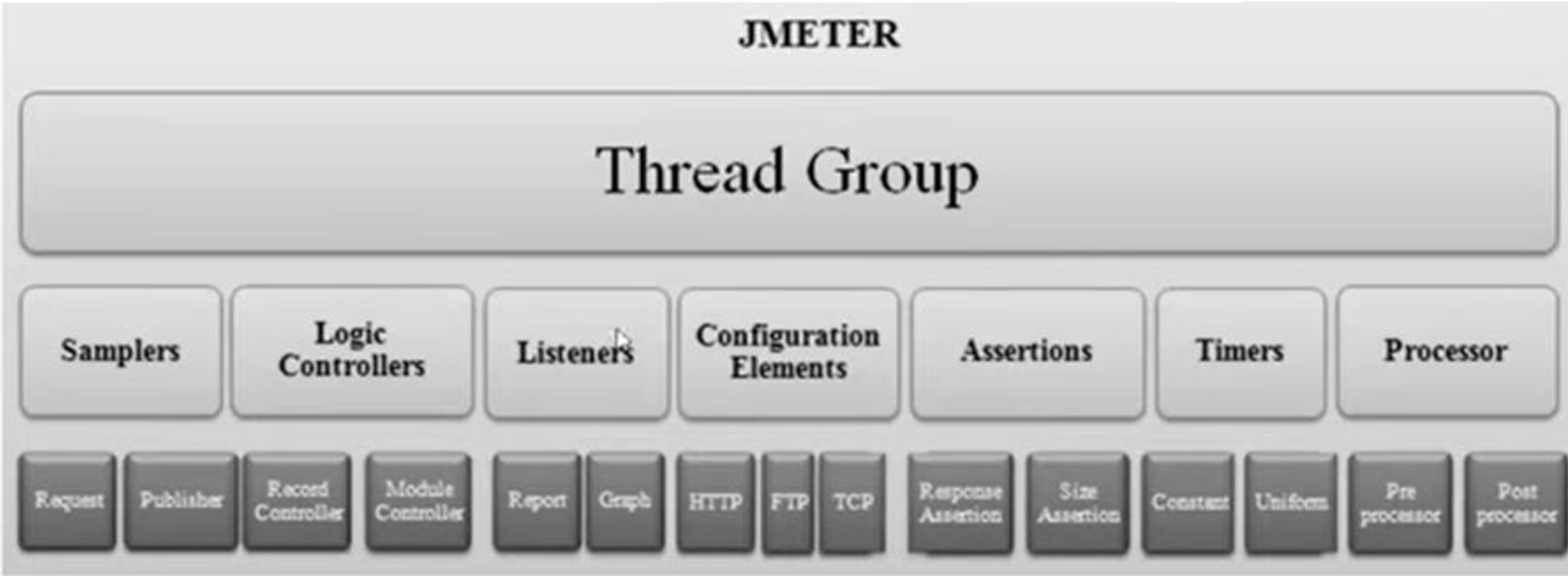
- Set JAVA_HOME
- Download jMeter (http://jmeter.apache.org/download_jmeter.cgi)
- Set JMETER_HOME (export PATH=$PATH:$JMETER_HOME/bin)
- Run jMeter via Command **jmeter**

Thread Group elements are the beginning points of your test plan.

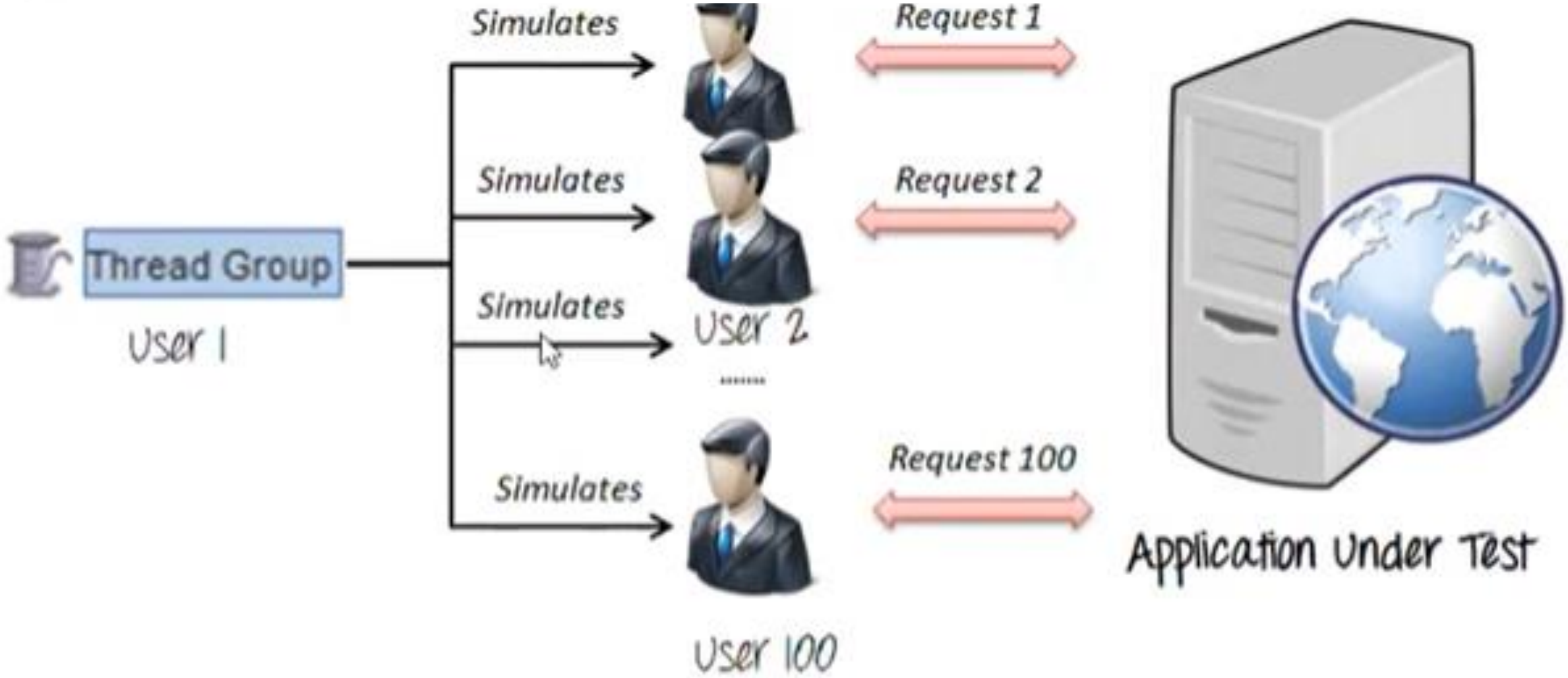
- Set the number of threads
- Set the ramp-up period
- Set the number of times to execute the test

What is Element in JMeter?

- The different components of JMeter are called Elements. Each Element is designed for a specific purpose.

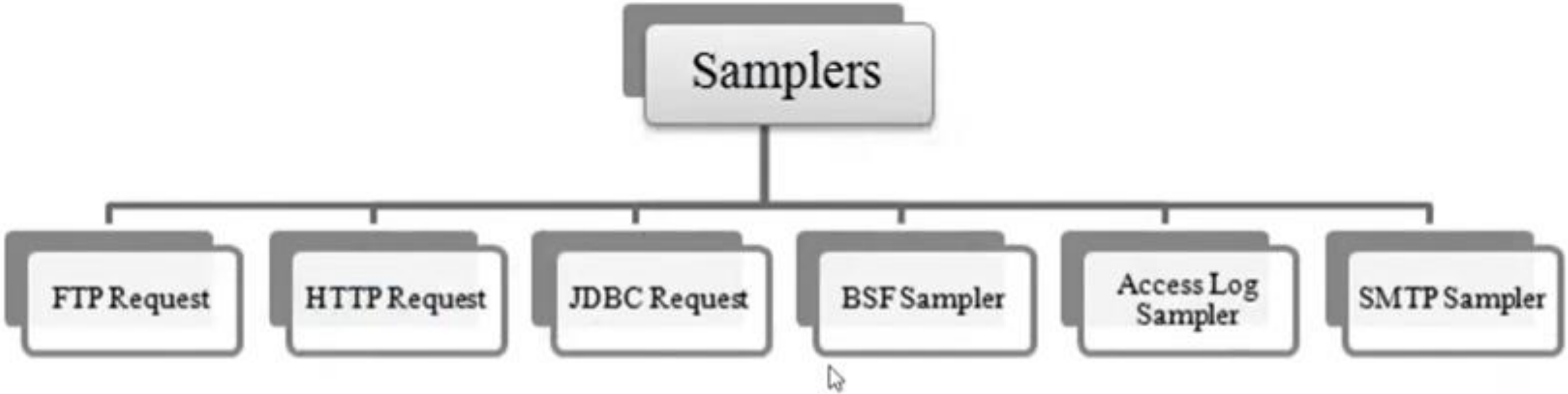


- Thread Groups is a collection of Threads. Each thread represents one user using the application under test. Basically, each Thread simulates one real user request to the server.
- The controls for a thread group allow you to Set the number of threads for each group.
- For example, if you set the number of threads as 100; JMeter will create and simulate 100 user requests to the server under test



Samplers

- Samplers are different type of requests send by Thread group.
- The user request could be FTP Request, HTTP Request, JDBC Request...Etc.

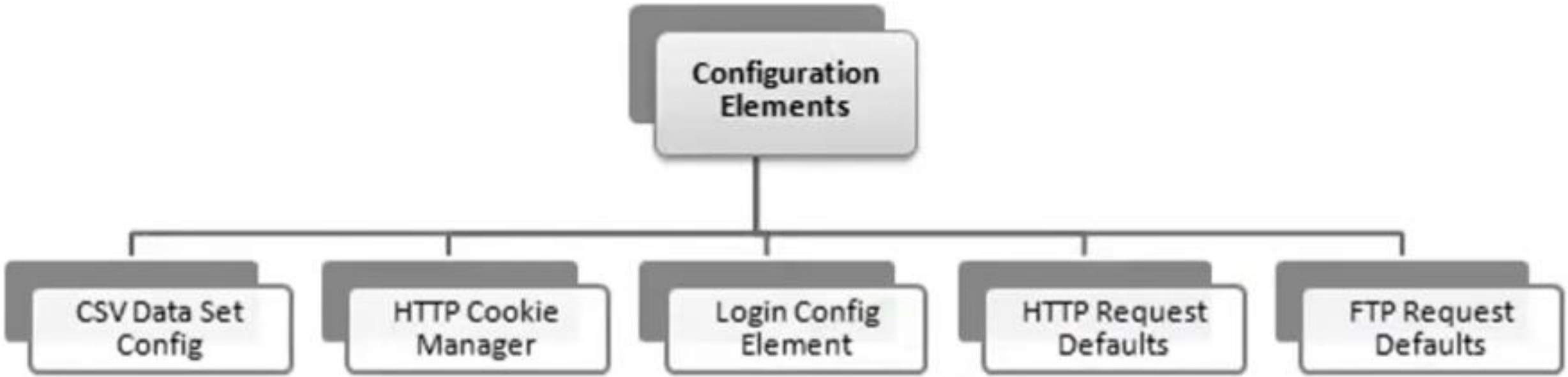


Listeners

- Listeners shows the results of the test execution.
- They can show results in a different format such as a tree, table, graph or log file
- Graph result listeners display the server response times on a Graph
- View Result Tree show results of the user request in basic HTML format
- Table Result show summary of a test result in table format
- Log show summary of a test results in the text file

Configuration Elements

- Set up defaults and variables for later use by samplers.
- **Commonly used configuration elements in Jmeter:**



Create First JMeter Test

- Step 1 - Start Jmeter
- Step 2 - Create a TestPlan
- Step 3 - Create a Thread Group (Users)
- Step 4 - Add a Sampler (Http)
- Step 5 - Add Listeners
- Step 6 – Run Test Plan
- Step7 – Save Test Plan

Assertions in JMeter

- **What is an Assertion?**
- Assertion help verifies that your server under test returns the **expected** results.
- **Types of Assertions**
  1. Response Assertion
  2. Duration Assertion
  3. Size Assertion
  4. HTML Assertion
  5. XML Assertion
  6. XML Schema Assertion
  7. XPATH Assertion
  8. JSON Assertion

- **Response Assertion**
- The response assertion used in test scripts to validate a pattern in the response body, header, code, message etc. There are different pattern matching rules to validate the response.
- **Size Assertion**
- The size assertion is used to validate the size of the response with a specified value in bytes.
- **Duration Assertion**
- The duration assertion is used to validate that the sampler request gets processed within a specified amount of time.
- **HTML Assertion**
- The HTML assertion is used to check the HTML syntax of the response.
- **XML Assertion**
- The XML assertion is used to validate that the response follows a valid XML syntax.
- **XML Schema Assertion**
- The XML Schema Assertion is used to validate the response against a specified XML schema.
- **XPath Assertion**
- The XPath assertion is used to validate the response using XPath expressions.
- **JSON Assertion**

  The JSON assertion is used to validate the response using JSON expressions.

Parameterization

We can pass parameters to the Request using Parameterization

1. JMeter internal parameters
2. Read Parameter values from CSV/Excel

Demo URL: https://openweathermap.org/current
API KEY: bd7f413abdadde68a8a41b65398d57b7

Current weather data for one location

By city name : You can call by city name or city name and country code. API responds with a list of results that match a searching word.

Example of API call:

```
api.openweathermap.org/data/2.5/weather?q=${city},${country}&APPID=${APIKEY}
```

By city ID : You can call by city ID. API responds with exact result.

Example of API call:

```
api.openweathermap.org/data/2.5/weather?id=${id}&APPID=${APIKEY}
```

Parameters:
APPID {APIKEY} is your unique API key

JMeter for API Testing?

- Fast API Testing
- Perform quick scope tests
- Enable load and stress testing
- Open-source tool
- Lots of plugins and extensions
- Cross-platform

- API or Application Programming Interface is a software that enables two applications to connect with each other

For example:

- Using an application on the mobile phone, sending a message or checking the weather on the phone

https://www.simplilearn.com/tutorials/jmeter-tutorial/jmeter-api-testing

1. Create a Simple Online Food Order and Delivery Web App using any of the technology you know. The app should have two modules at least
i. For login validation without Database
ii. Online Order of food
Test it with your browser.
Apply the Load Test to this developed webapp by setting up threads which will simulate concurrent users configuring 10 threads, with 2s ramp up and each thread will run 5 times.

Once the Test is completed Visulalize the Test Results with the following Listener
(a) Graph Results
(b) View Results Tree
(c) View Results Table

2. Develop the Following API for any typical use case with any of the platform and apply the Load test for that

a. GET API

b. POST API

c. PULL API

d. DELETE API

Also checks the response with Assertions Listener