PERFORMANCE TESTING

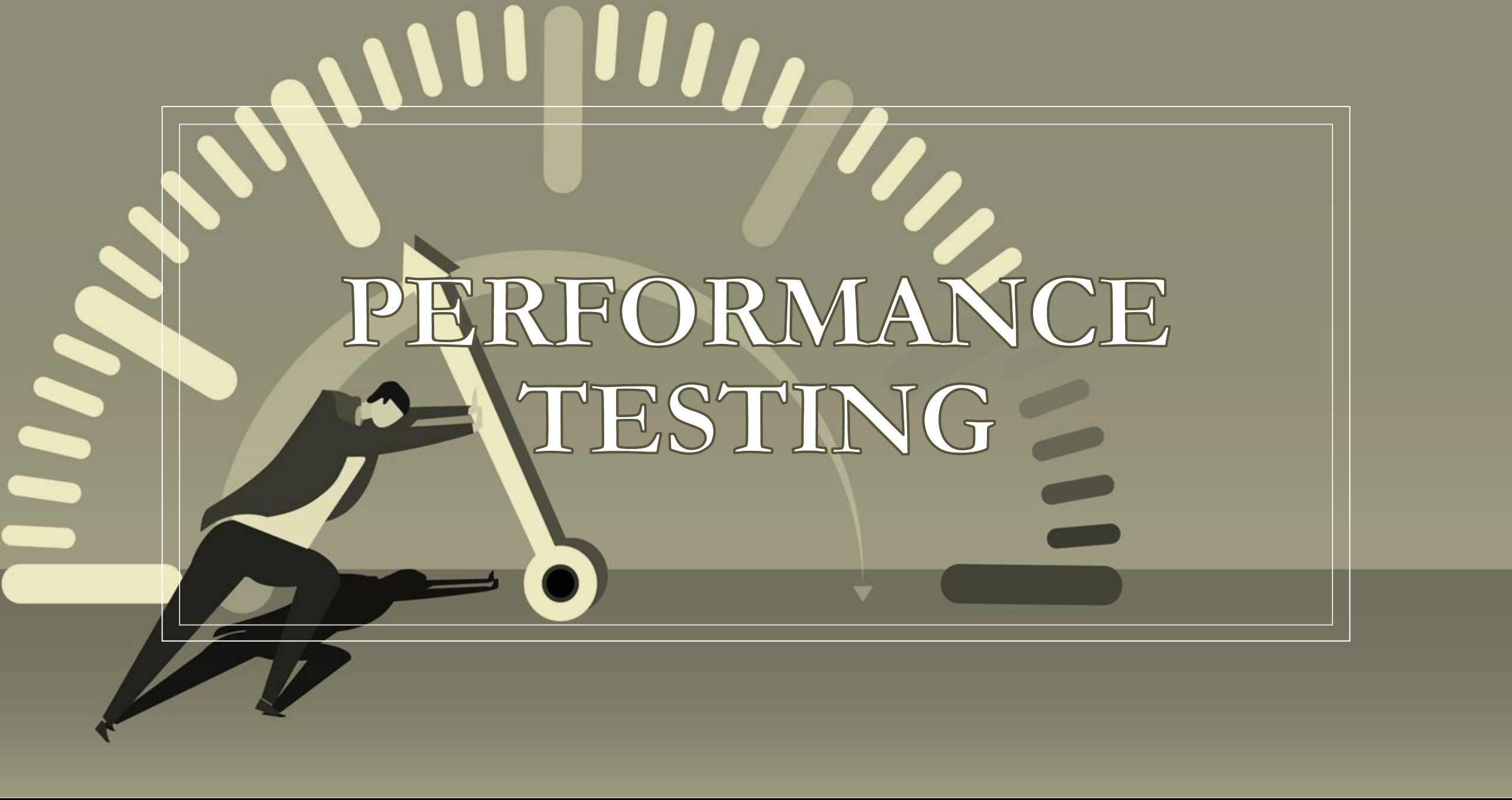


Table of Content

Performance Testing

Time Behavior

The most common performance testing objective. It examines the ability of the system to respond to user inputs within a specified time and under specified conditions.

Resource Utilization

If the availability of system resources is identified as a risk, the utilization of those resources (e.g., the allocation of limited RAM) may be investigated by conducting specific performance tests

Capacity

If issues of system behavior at the required capacity limits of the system (e.g., numbers of users or volumes of data) is identified as a risk, performance tests may be conducted to evaluate the suitability of the system architecture.

Performance Testing

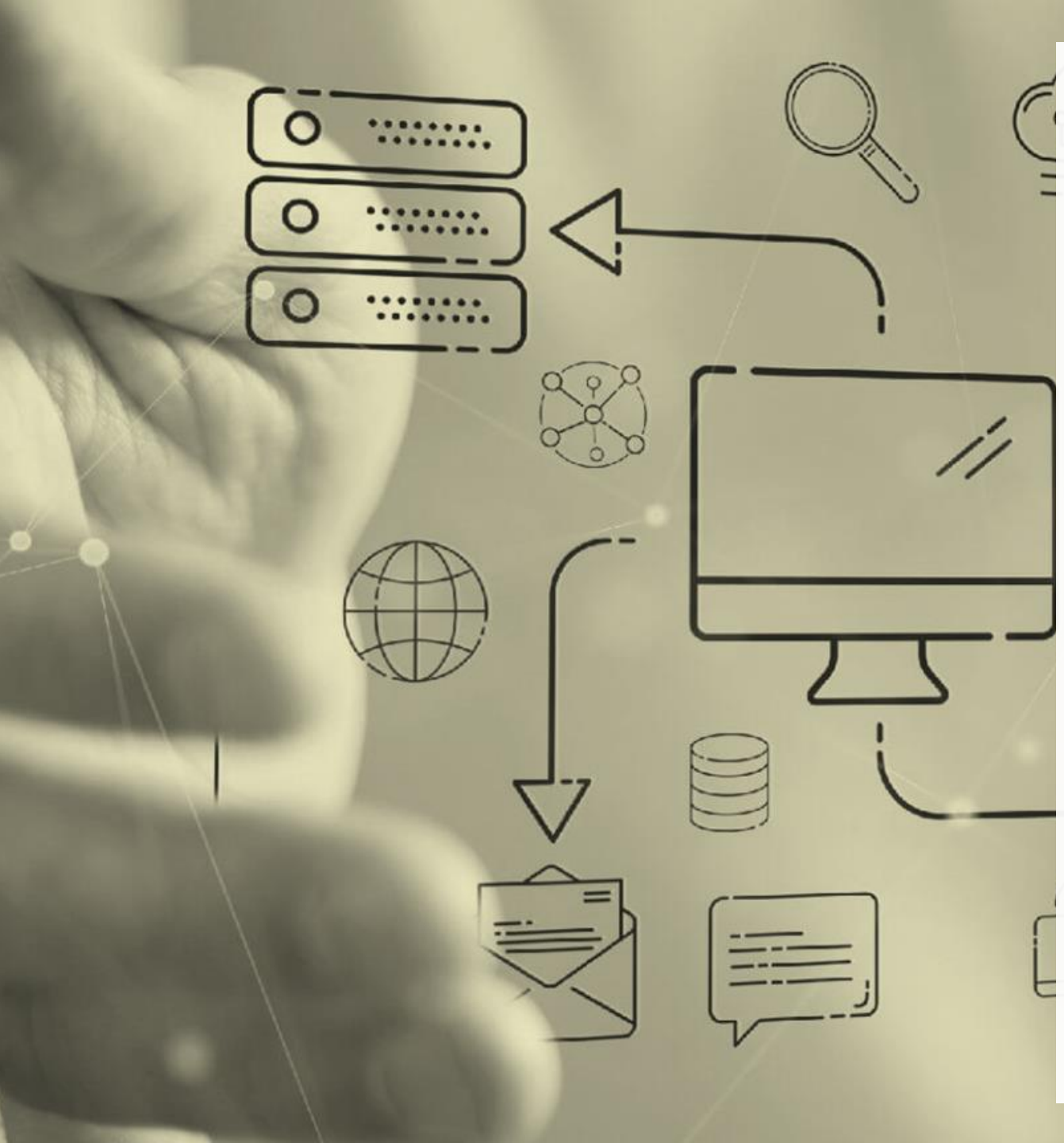
Performance testing is an umbrella term including any kind of testing focused on performance (responsiveness) of the system or component under different volumes of load.

Load Testing

- Load testing focuses on the ability of a system to handle increasing levels of anticipated realistic loads resulting from transaction requests generated by controlled numbers of concurrent users or processes.

Stress Testing

- Stress testing focuses on the ability of a system or component to handle peak loads that are at or beyond the limits of its anticipated or specified workloads.



Scalability Testing

- Scalability testing focuses on the ability of a system to meet future efficiency requirements which may be beyond those currently required.

Spike Testing

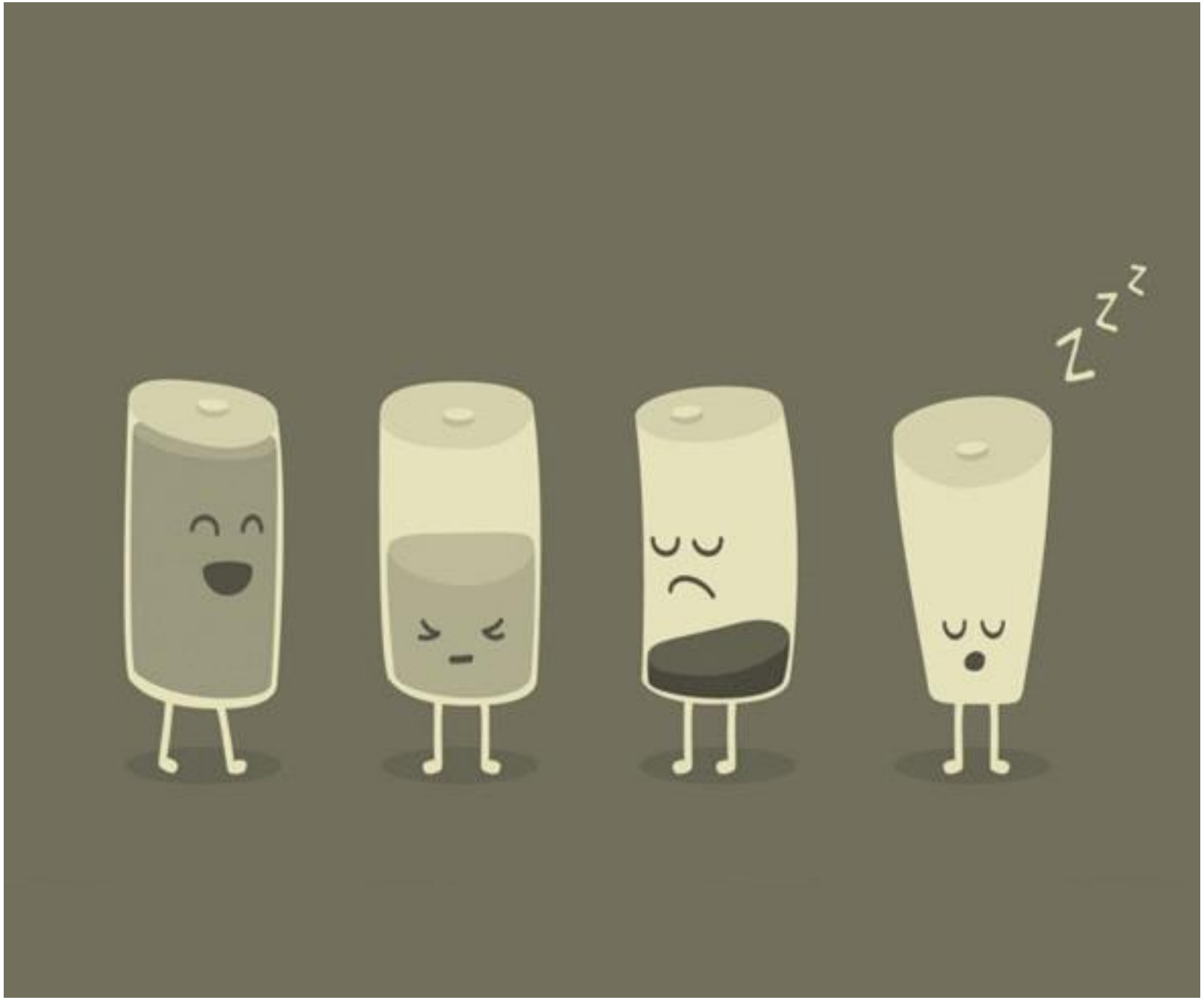
- Spike testing focuses on the ability of a system to respond correctly to sudden bursts of peak loads and return afterwards to a steady state.

Endurance Testing

- Endurance testing focuses on the ability of the system over a time frame specific to the system's operational context.

Concurrency Testing

- Concurrency testing focuses on the impact of situations where specific actions occur simultaneously (e.g., when large numbers of users log in at the same time)



Capacity Testing

- Capacity testing determines how many users and/or transactions a given system will support and still meet the stated performance objectives.

Concept of Load Generation

- **User Interface** — Maybe an adequate approach if only a small number of users are to be represented
- **Crowds** — Depends on the availability of a large number of testers who will represent real users
- **APIs** — This approach is less sensitive to changes in the UI.
- **Communication Protocols** — This tool-based approach simulates very large numbers of users in a repeatable and reliable manner.

Creating Load Profiles

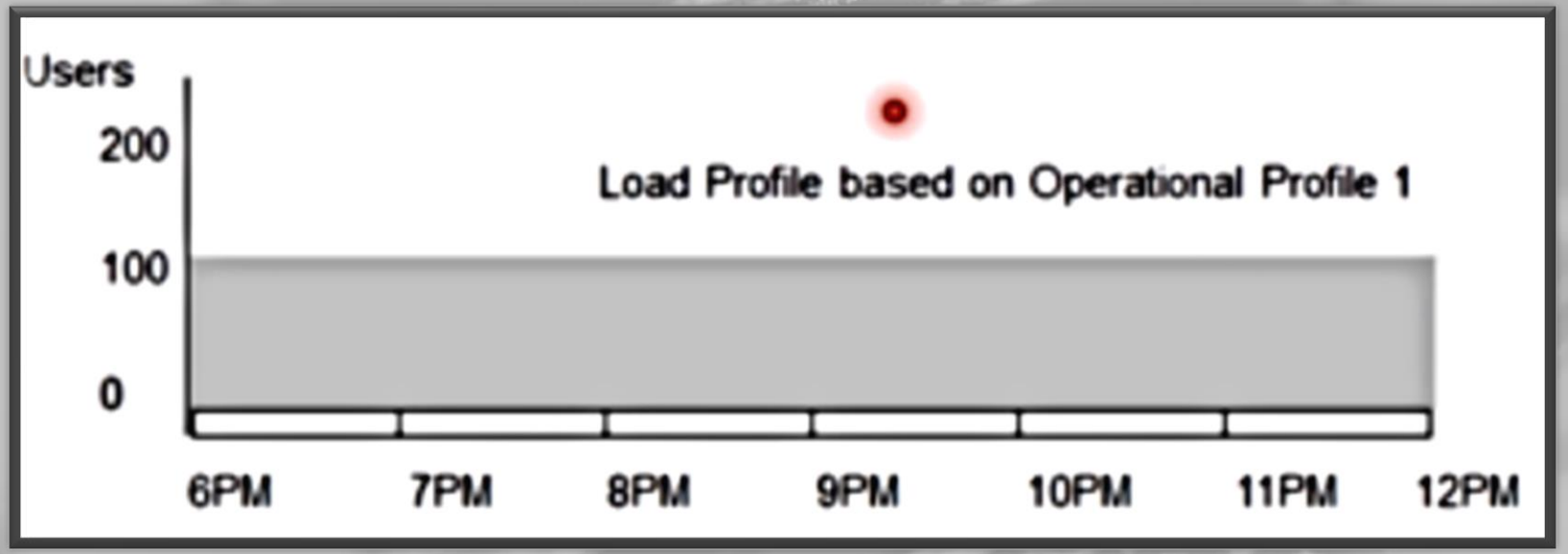
- A load profile specifies the activity which a component or system being tested may experience in production. It consists of a designated number of instances that will perform the actions of predefined operational profiles over a specified time period.

Creating Load Profiles

To create a correct profile, you need to understand the following:

- **Performance Testing Objectives**
- **Operational Profiles which accurately represent individual usage patterns**
- **The quantity and time distribution with which the operational profiles are to be executed such that the SUT experiences the desired load. Examples are:**
  - **Ramp-ups:** Steadily increasing load (e.g., add one virtual user per minute)
  - **Ramp-downs:** Steadily decreasing load
  - **Steps:** instantaneous changes in load (e.g., add 100 virtual users every five minutes) .
  - **Predefined distributions:** (e.g., volume mimics daily or seasonal business cycles)

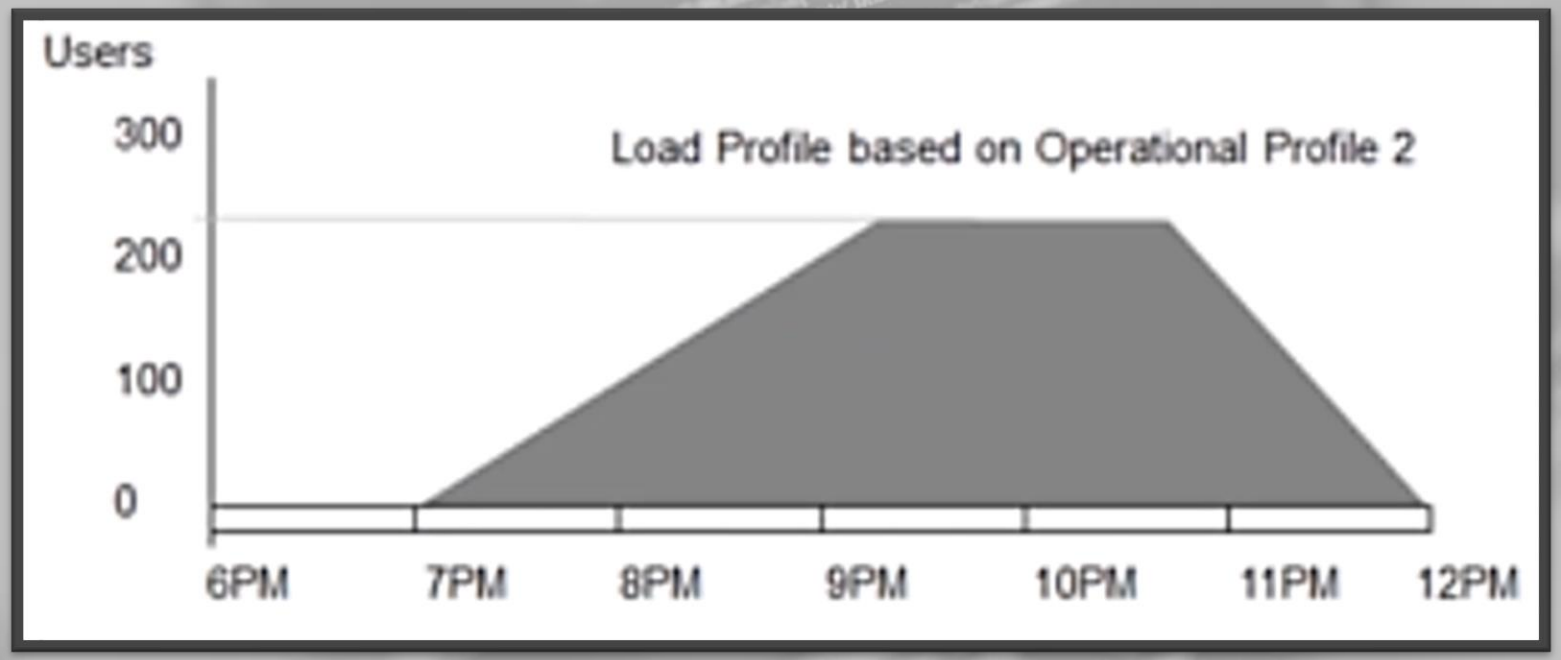
Creating Load Profiles



Load Profile 1

- A load profile is shown which consists of a step input of 100 virtual users.
- These users perform the activities defined by Operation Profile 1 over the entire duration of the test.
- This is typical of many performance load profiles that represent a background load.

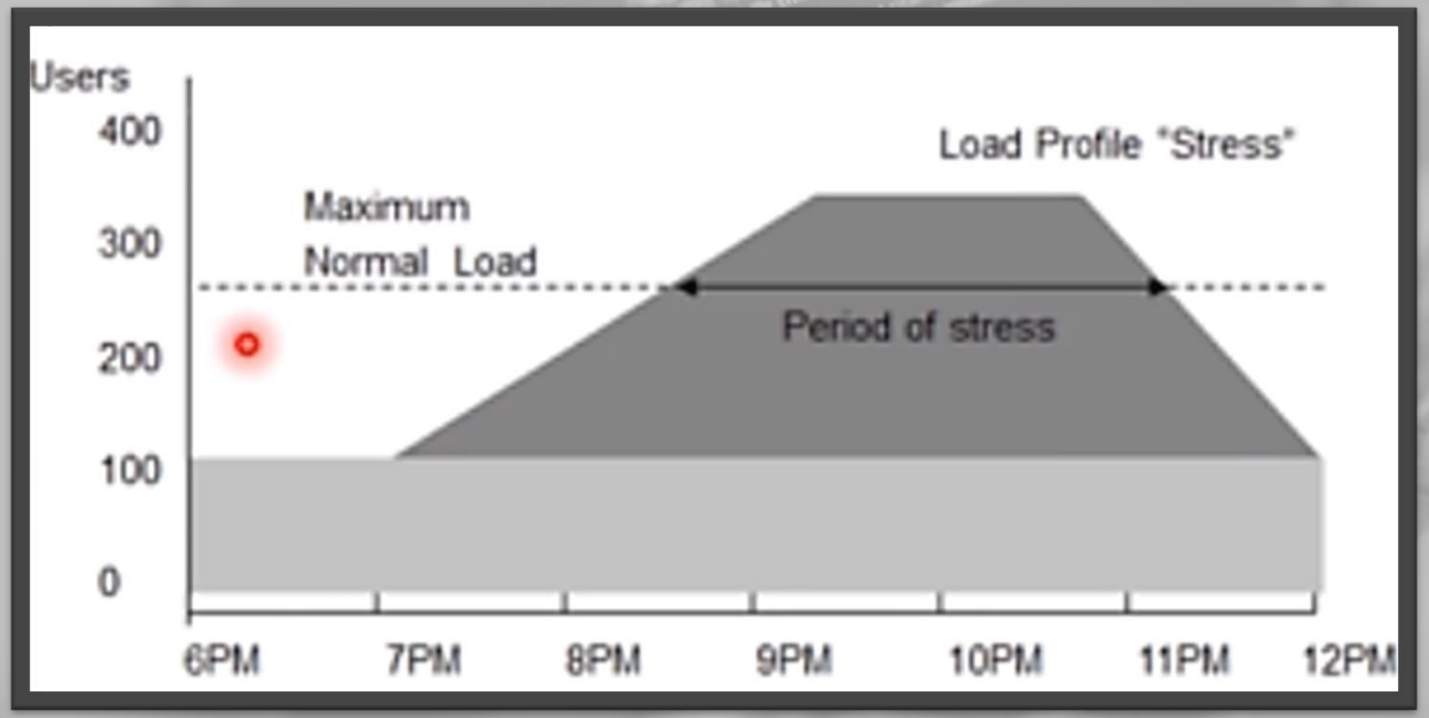
Creating Load Profiles



Load Profile 2

- The diagram shows a load profile that consists of a ramp-up to 220 virtual users that is maintained for 2 hours before ramping down.
- Each virtual user performs activities defined in Operational profile 2

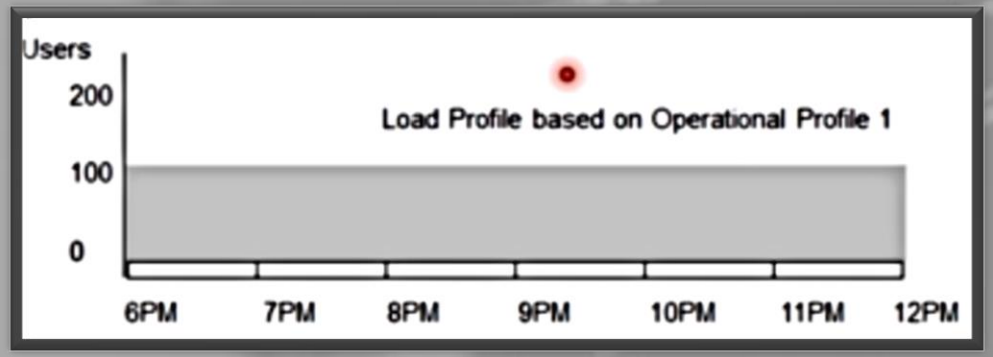
Creating Load Profiles



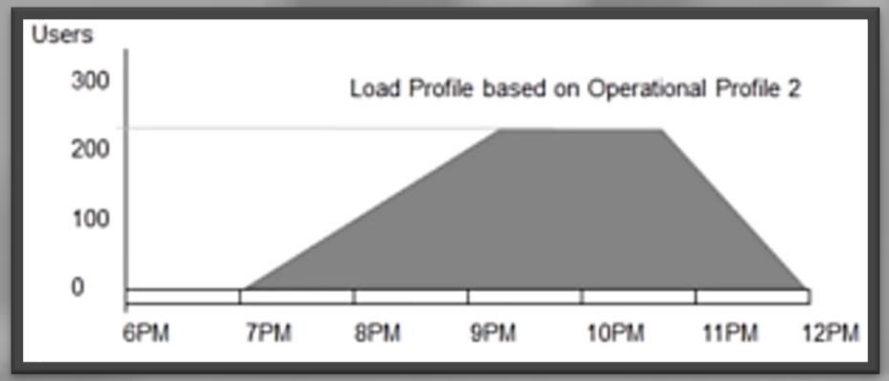
Load Profile (1+2)

- The diagram shows the load profile that results from the combination of the 2 described above.
- The system under test is subjected to the three-hour period of stress.

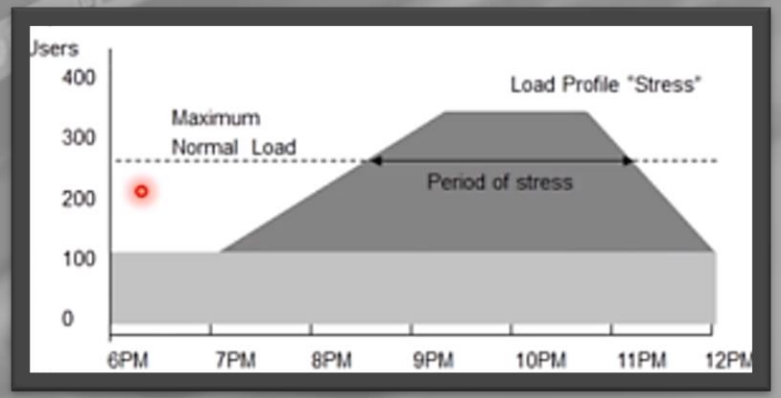
Creating Load Profiles



Load Profile 1



Load Profile 2



Total Load Profile

Try to solve this example

You are working on a project that tracks health history information for patients across a region. The number of records handled by the system is in the millions due to the large number of patients in the region. Patient information must be accessible to doctors in offices, hospitals and urgent care facilities. The information should be presented to the requestor within three seconds of request, particularly for patients with critical allergies and preconditions.

One of the operational profiles you have identified is an emergency room doctor. You have determined that this person will access the system 10 times per shift (a shift is 10 hours) and that they will normally update 6 patient records for each access by entering notes into the database regarding the patient's treatment. They will print those patient records to be retained in the patient's file at the hospital. For new patients, another user will enter the information into the system.

The shifts these doctors work are: 7am - 5pm (day shift), 2pm - midnight (evening shift), 9pm to 7am (night shift). There are 1000 doctors that work the day shift, 1000 that work the evening shift, and 500 that work the night shift. Assuming an even distribution of the system access across a shift, what is the proper load profile for these doctors? And what is the highest number of concurrent accesses on the system from these doctors?

Solution

- Total transactions should be 2500 doctors x 10 accesses x 6 record updates: 150,000 transactions per day.

- Hourly load:
  - 7 hours of 500 doctors = 21,000 transactions
  - 11 hours of 1000 doctors = 66,000 transactions
  - 3 hours of 1500 doctors = 27,000 transactions
  - 3 hours of 2000 doctors = 36,000 transactions

- Doctors per hour in 24 hour clock:

- 1:00 - 500
- 2:00 - 500
- 3:00 - 500
- 4:00-500
- 5:00 - 500
- 6:00 - 500
- 7:00 - 1000
- 8:00-1000
- 9:00-1000
- 10:00 - 1000
- 11:00 - 1000
- 12:00 - 1000
- 13:00 - 1000
- 14:00-2000
- 15:00 - 2000
- 16:00-2000
- 17:00-1000
- 18:00-1000
- 19:00-1000
- 20:00-1000
- 21:00-1500
- 22:00-1500
- 23:00-1500
- 24:00-500

Try to solve this example

You are working for a company that has developed software that will be used in the Olympic skiing events to track and post times. This software will take data from various sensors to determine start and mish times for each individual racer as they race one at a time. It will also allow judges to enter a disqualification status for an individual skier at any time during their run and within 5 minutes of the end their run. The information will be sent through an API to software that will control the display board at me event and to the interface used by the broadcast communities.

Your company has also developed a mobile application that will be available for free download to anyone in the world who wants to receive "real time" results from the events. Testing for the mobile application will be done in the cloud using device simulators.

You have identified that you expect to have a load of 100,000 users for a high interest race and 20,000 users for a low-interest race. There are 5,000 users who seem to always be connected, even when nothing is happening.

High-interest races are the semi-finals and finals for any particular contest. A race normally lasts for 5 minutes and there are usually 8 racers in the semi-finals and 4 racers in the finals. A low-interest race is any of the first 30 races that lead to determining who will be in the semi-finals. How would the proper load profile look like for one contest?

Solution

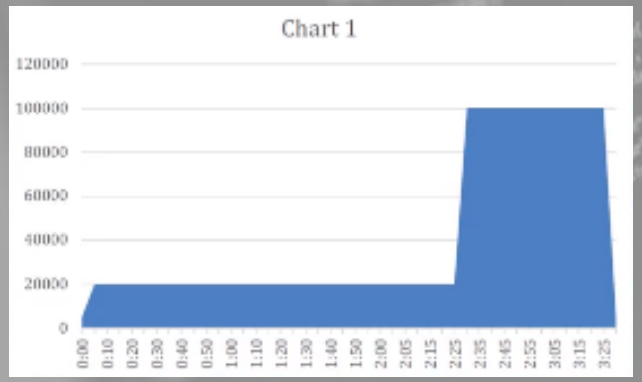


Chart 1
120000
100000
80000
60000
40000
20000
0
0:00 0:10 0:20 0:30 0:40 0:50 1:00 1:10 1:20 1:30 1:40 1:50 2:00 2:05 2:15 2:25 2:35 2:45 2:55 3:05 3:15 3:25

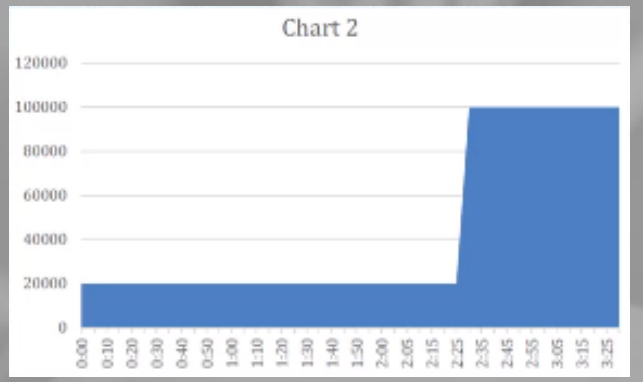


Chart 2
120000
100000
80000
60000
40000
20000
0
0:00 0:10 0:20 0:30 0:40 0:50 1:00 1:10 1:20 1:30 1:40 1:50 2:00 2:05 2:15 2:25 2:35 2:45 2:55 3:05 3:15 3:25

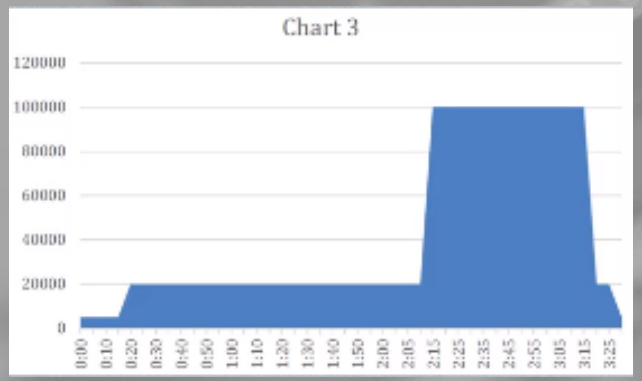


Chart 3
120000
100000
80000
60000
40000
20000
0
0:00 0:10 0:20 0:30 0:40 0:50 1:00 1:10 1:20 1:30 1:40 1:50 2:00 2:05 2:15 2:25 2:35 2:45 2:55 3:05 3:15 3:25

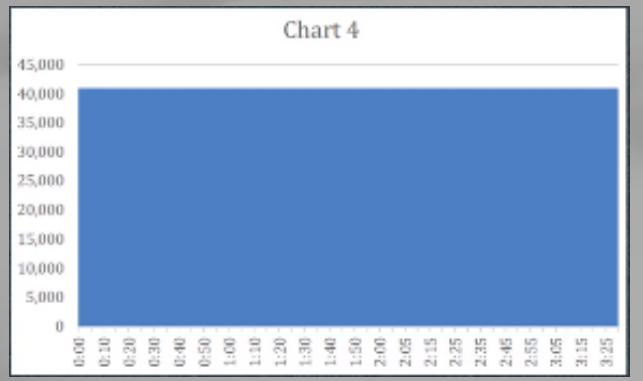


Chart 4
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
0:00 0:10 0:20 0:30 0:40 0:50 1:00 1:10 1:20 1:30 1:40 1:50 2:00 2:05 2:15 2:25 2:35 2:45 2:55 3:05 3:15 3:25


Advanced Performance Testing

RESPONSE TIME

Time taken by the application to respond to user's request

Absolute Response Time

- Total time from the instant a user clicks on a link until the response from the server is rendered completely

Perceived Response Time

- Response time as perceived by the user

Rendering Time

- Time taken by the browser to parse and render the response received from the server.

Network Latency

- Time taken by a data packet to be captured, transmitted, and processed through multiple devices, then received at its destination.

Network Latency

- Typically, anything at 100ms is acceptable for gaming. However, the 20ms to 40ms range is considered optimal.


THROUGHPUT
Number of transactions per unit time
Measured in transactions/second or bandwidth (bytes/second)

UTILIZATION

Ratio of throughput to maximum capacity

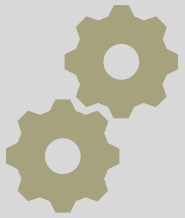
It is not desirable to operate above 80% utilization

ROBUSTNESS

How well the application detects and handles various errors and exceptions

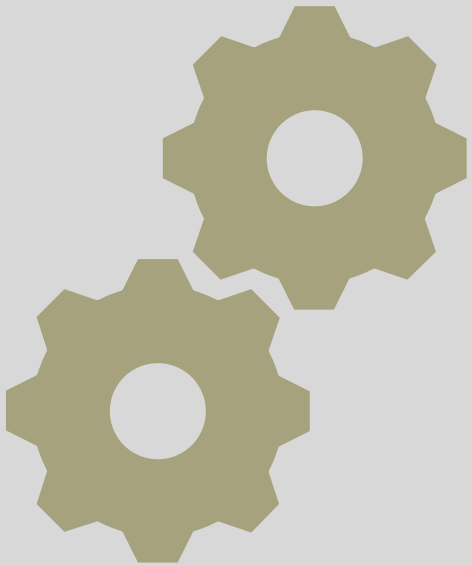
We use MTBF to measure robustness

Performance Test Environment



- Hardware & software used to conduct performance testing
- Not advisable to be done in the production environment

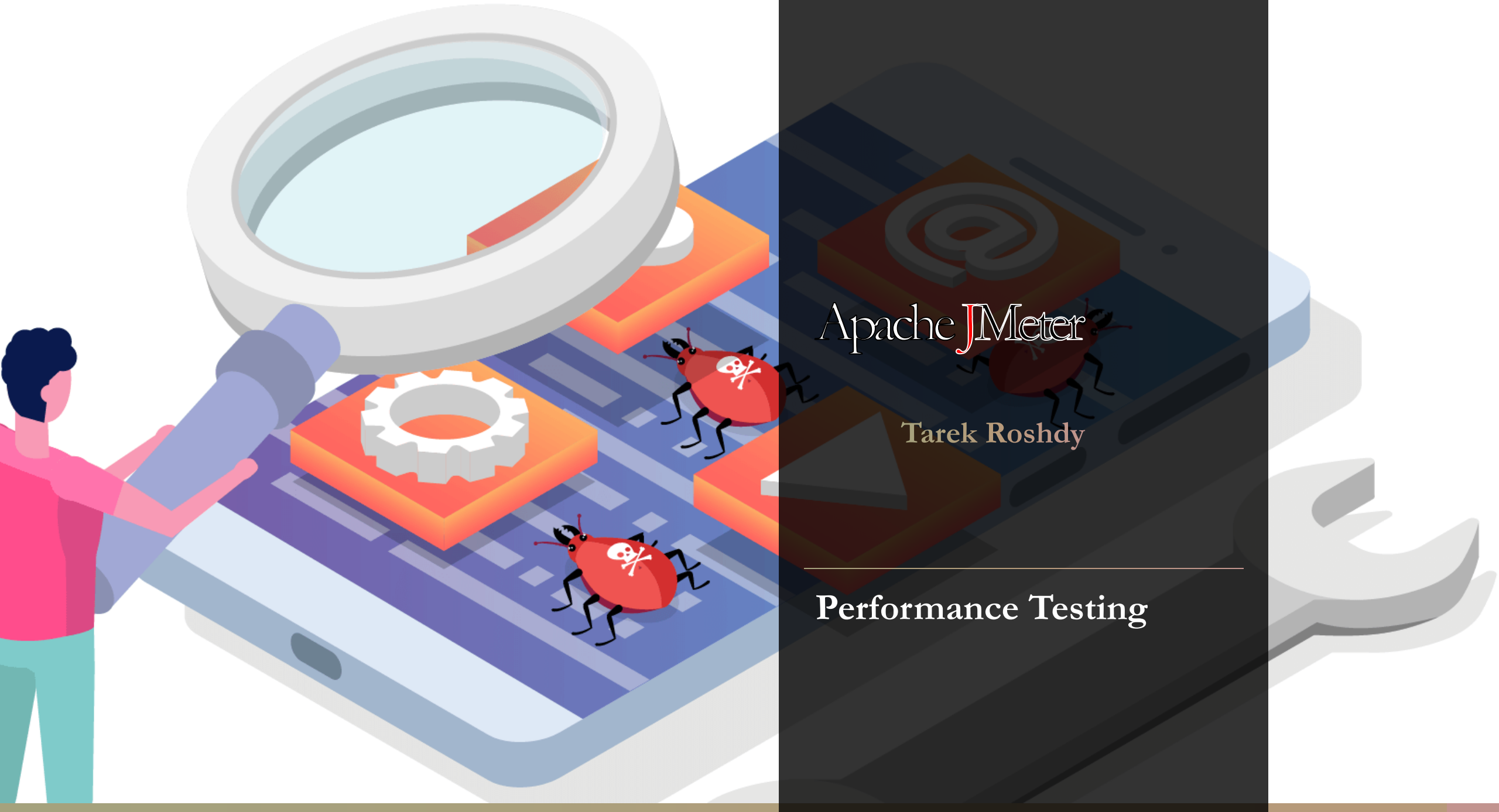
Why do we need a separate test environment?



- The system might crash
- Application response time is degraded
- Security holes might be created because of the use of test accounts
- Production database will include performance test input and output data
- Application log files and system log files may be filled up
- Analytics will be influenced


Performance Environment should be like the production environment


Performance Environment
should be isolated



Apache JMeter

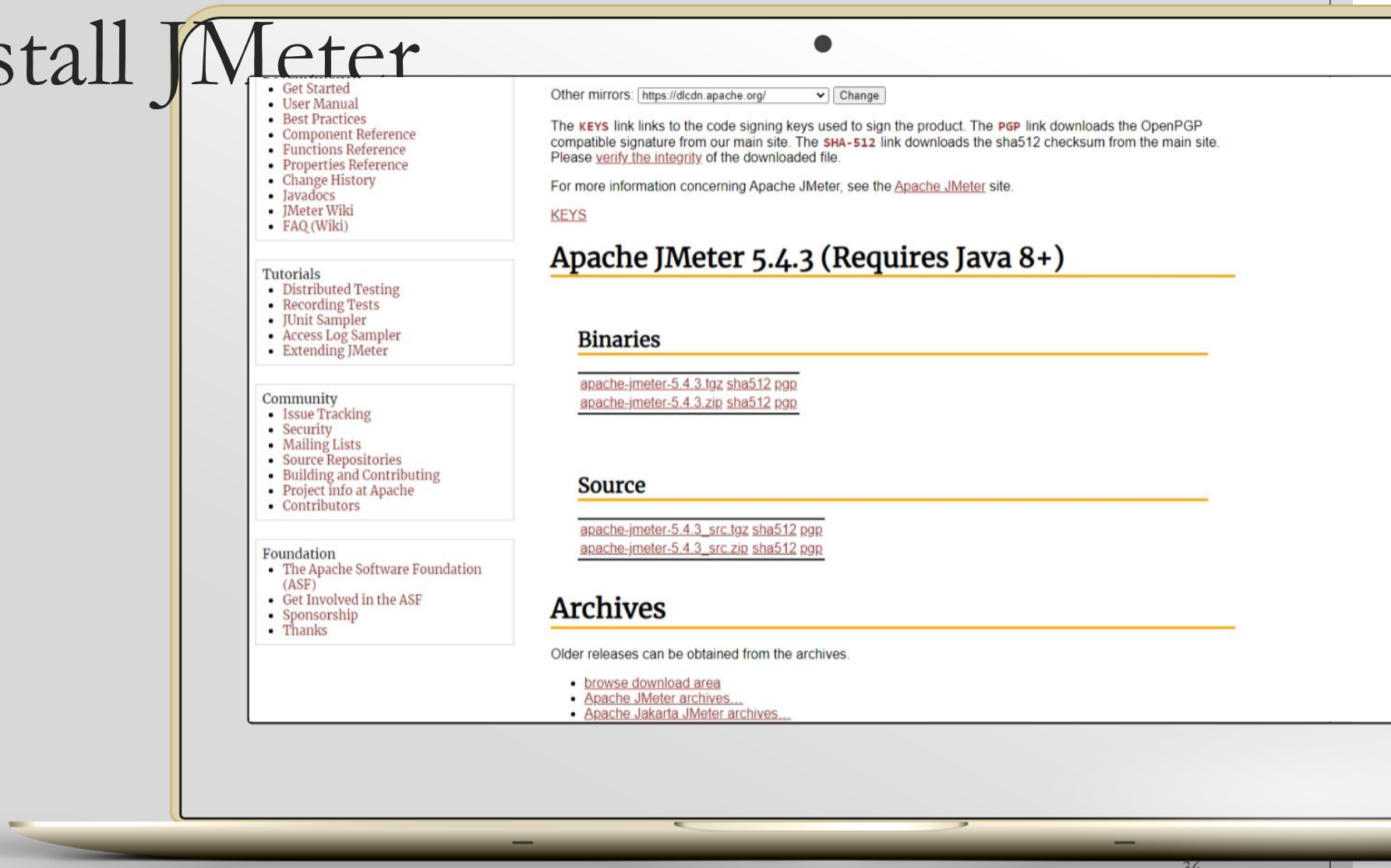
Tarek Roshdy

Performance Testing

How to install JMeter

jmeter.apache.org

1. Download Releases
2. Binaries
3. Check for Java version
4. jmeter.bat file

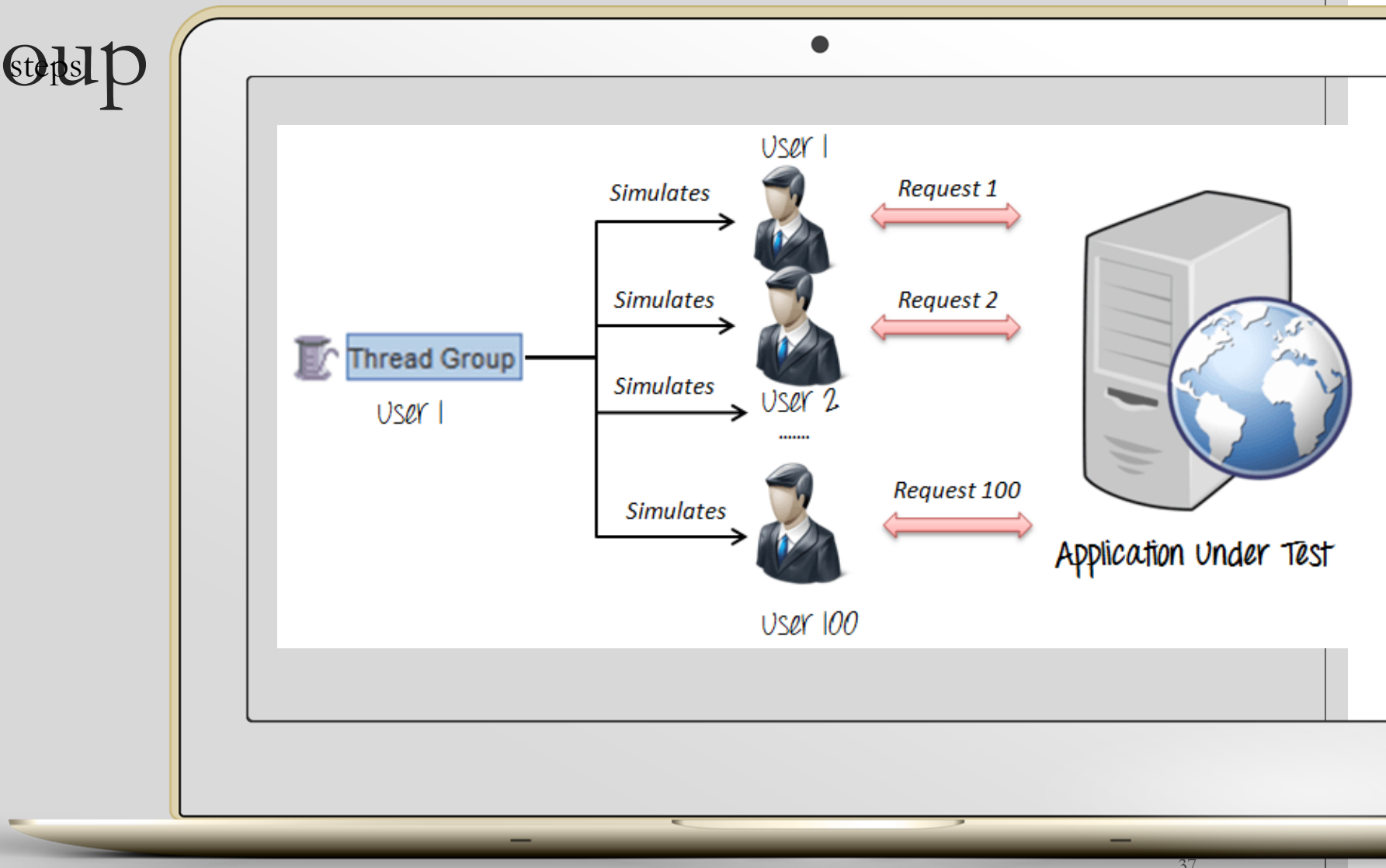


Thread Group

Thread Group elements are the initial steps of JMeter Test Plan.

Add a thread group

1. Test plan
2. Right click
3. Add
4. Threads (Users)
5. Thread group

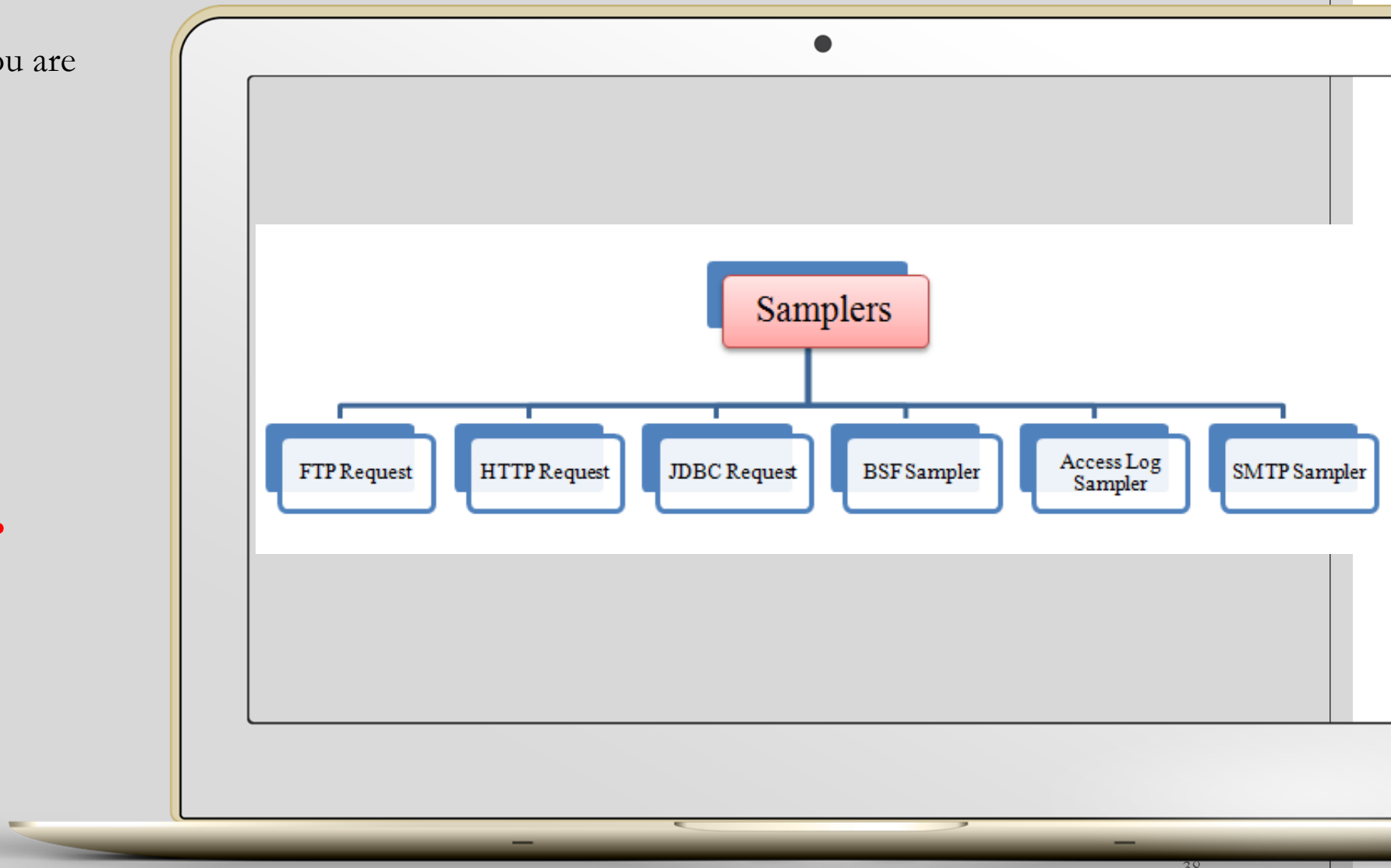


Samplers

It is used to identify the server that you are going to test.

Add a sampler

1. Test plan
2. Samplers
3. Choose the type you want (e.g. **HTTP Request**)

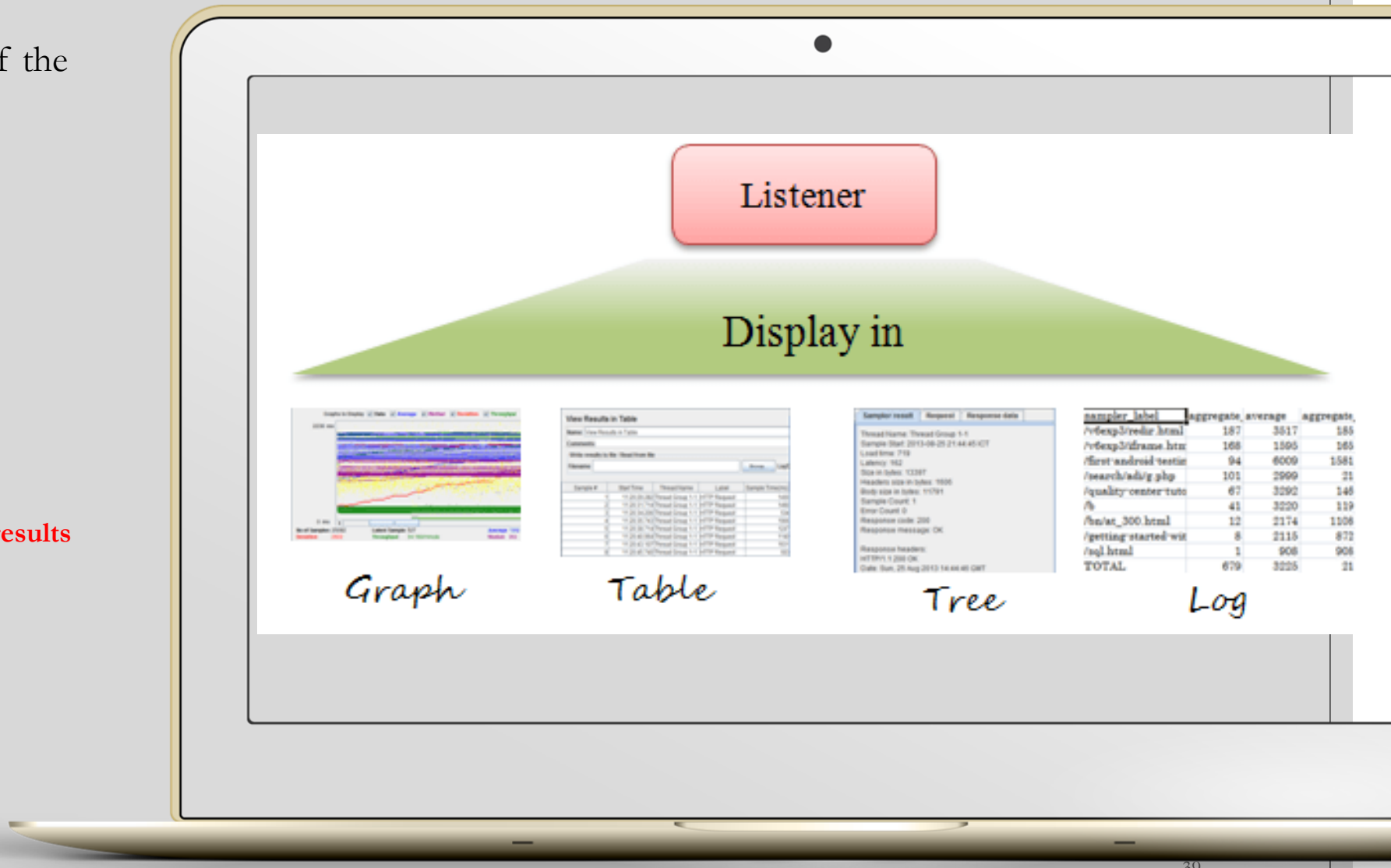


Listeners

It is used for monitoring the results of the testing.

Add a Listener

1. Test plan
2. Add > Listener
3. Choose the type you want (e.g. **View results tree – View results in table**)



Record/Playback or Capture/Replay Testing

Record and playback testing is a type of automated testing where the tool records the activity of the user and then imitates it.

BlazeMeter

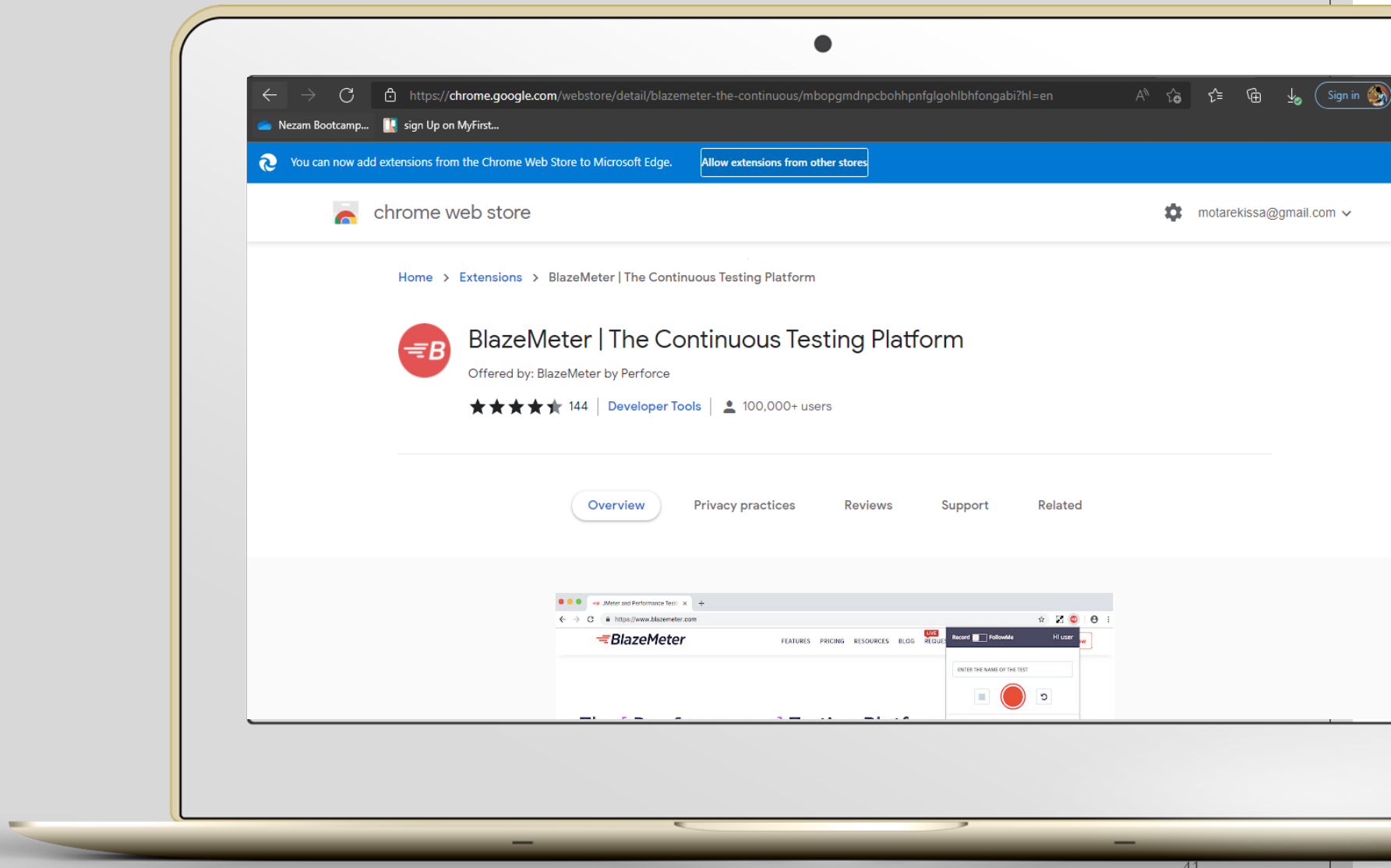
Using blazemeter plugin with Jmeter

Adding Blazemeter Plugin

Using Google Chrome

Steps

1. Open web store
2. Click on **Add to Chrome**
3. Create an account

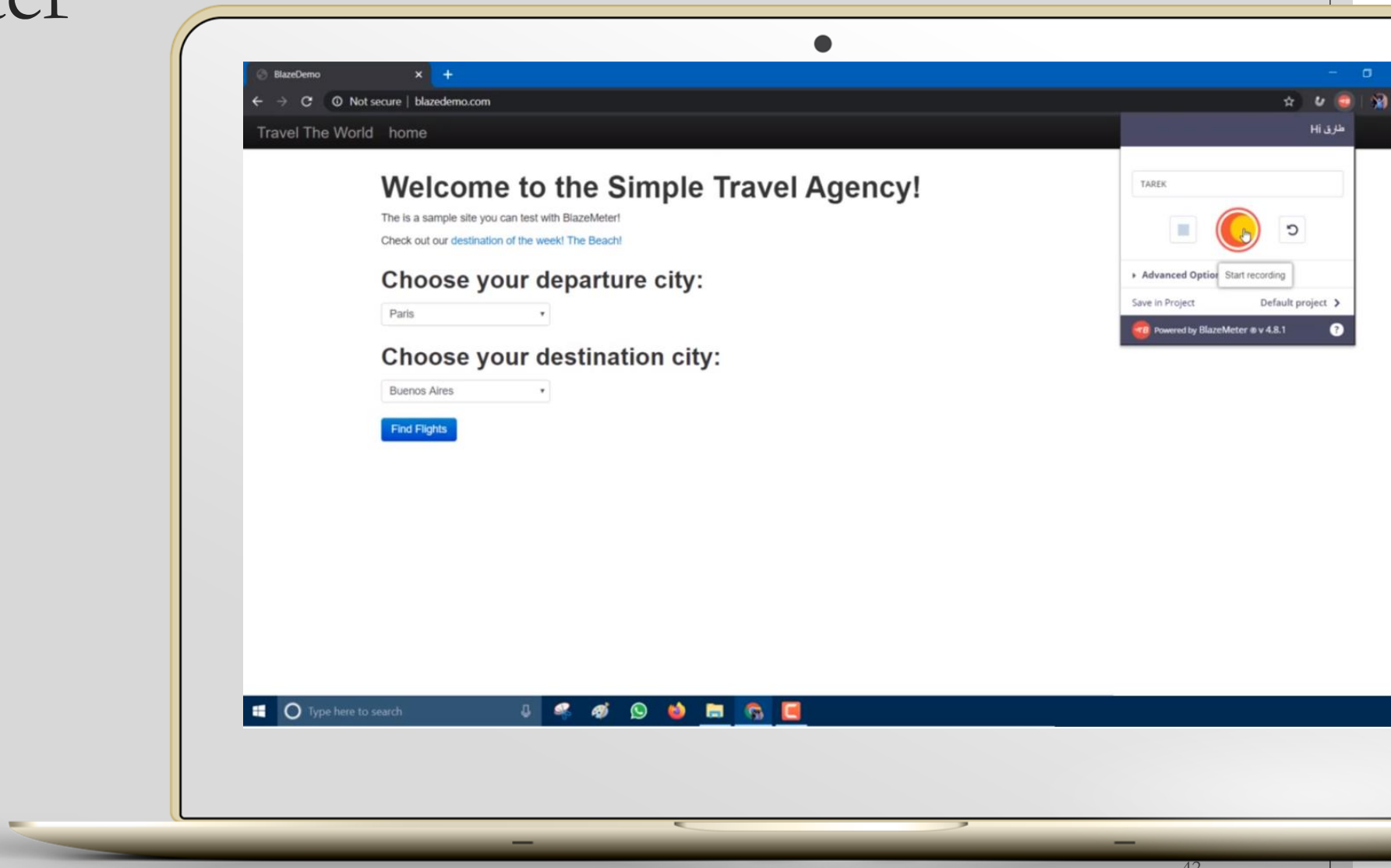


Recording scripts using Blazemeter

Using Google Chrome

Steps

1. Click on **Start Recording**
2. Open the website
3. Continue recording the script
4. Save as Jmeter script

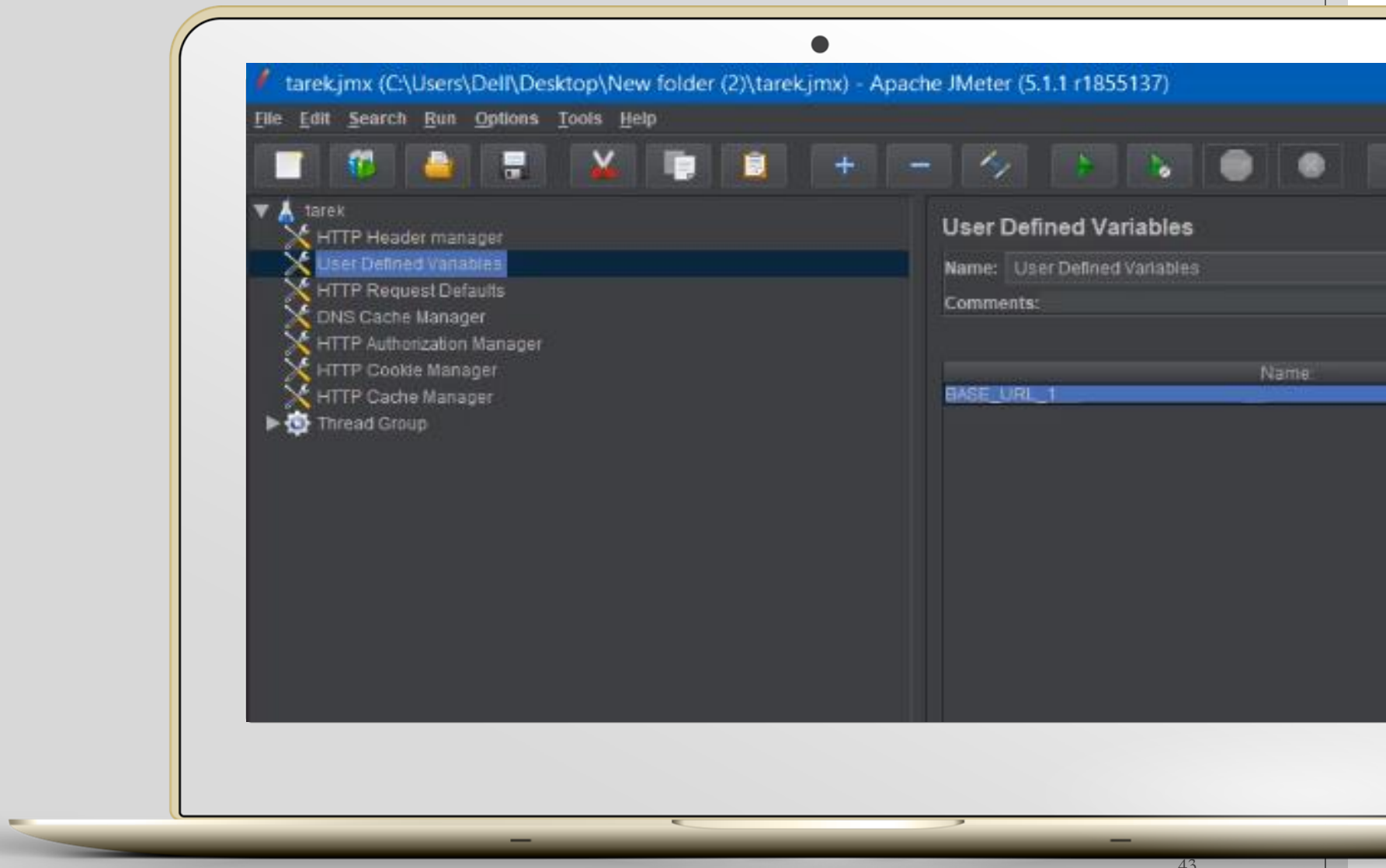


Replaying recorded scripts

Using JMeter

Steps

1. Click on **Open**
2. Choose script file



Difference between Average & Median

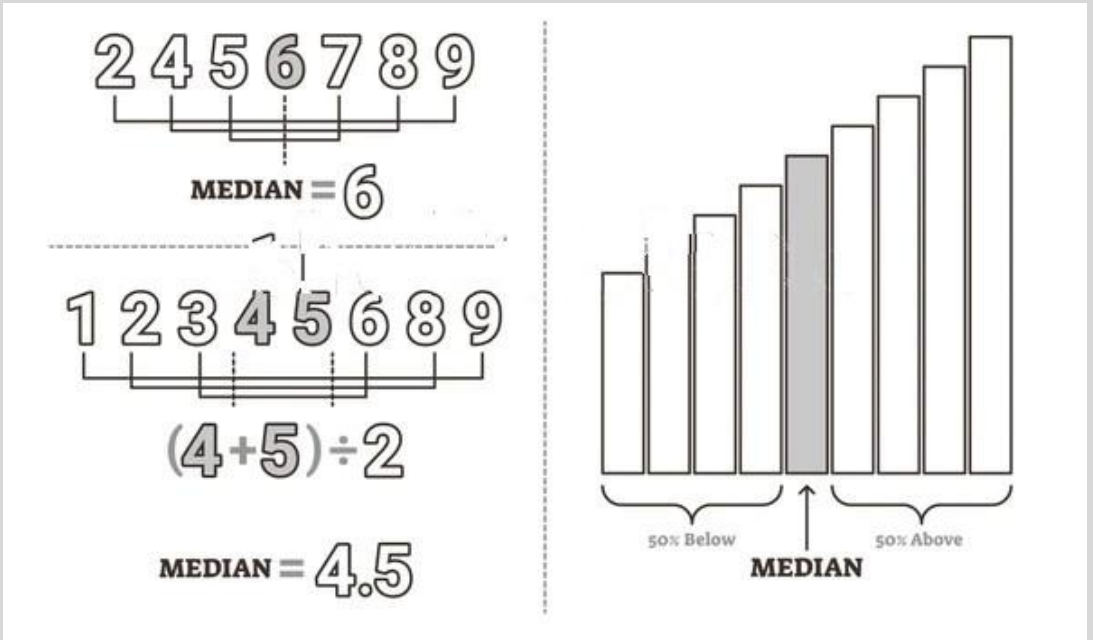
Average

- **The sum of the numbers divided by the amount of numbers.**

$$5+5+5+6+7+7+14$$
$$49/7 = 7$$

Median

- **The number in the middle**



Difference between Average & Median
With examples

Average **VS** Median

100,1000,2000,3000,100000

$$Average = \frac{100+1000+2000+3000+100000}{5}$$

$$= 21{,}220$$

$$Median = 2000$$

10,10,10,10,7000,9000,20000

$$Average = \frac{10+10+10+10+7000+9000+20000}{7}$$

$$= 5148$$

$$Median = 10$$