Absolutely! Here's your complete step-by-step guide from scratch — perfect for documentation, demos, or handoff to teammates. This will walk through JMeter setup, login simulation, performance testing, and HTML report generation on your Linux machine.

# 📘 **Performance Testing Documentation with Apache JMeter (Linux)**

Target URL: [https://dashboard.sm.test.cwdin.com](https://dashboard.sm.test.cwdin.com/)  
Test Goal: Load test login + device page under high user traffic and generate a clean HTML performance report.

## ✅ Prerequisites

### 1. **Install Java**

sudo apt update

sudo apt install openjdk-21-jdk -y

Check version:

java -version

### 2. **Download and Setup Apache JMeter**

cd /opt

sudo wget https://downloads.apache.org//jmeter/binaries/apache-jmeter-5.6.3.tgz

sudo tar -xvzf apache-jmeter-5.6.3.tgz

sudo mv apache-jmeter-5.6.3 jmeter

Add to PATH (optional):

echo 'export PATH=$PATH:/opt/jmeter/bin' >> ~/.bashrc

source ~/.bashrc

Test:

jmeter -v

## ✅ Create and Save Test Plan (performance\_test\_plan.jmx)

### 1. Open JMeter GUI:

/opt/jmeter/bin/jmeter

### 2. Build the Test Plan:

#### a. Add Thread Group:

* Threads: 500
* Ramp-up: 60
* Loop Count: 1

#### b. Add HTTP Request → for **Login**:

* Method: POST
* Protocol: https
* Server: web.sm.test.cwdin.com
* Path: /api/v1/user/auth/login
* Body Data:

{

"username": "your\_username",

"password": "your\_password"

}

#### c. Add HTTP Header Manager (under Login):

* Content-Type: application/json
* Accept: application/json

#### d. Add Cookie Manager (under Thread Group)

#### e. Add HTTP Request → for **Open Devices Page**:

* Method: GET
* Protocol: https
* Server: dashboard.sm.test.cwdin.com
* Path: /home/entity-management/devices

#### f. (Optional) Add Assertions:

* Response Assertion: contains Devices or check status = 200

#### g. Add Listeners (optional for GUI testing):

* Aggregate Report
* Graph Results
* View Results Tree (for debugging only)

#### h. Save your plan:

/home/cwd/Downloads/Performancss/performance\_test\_plan.jmx

## ✅ Run Test and Generate HTML Report

### Step 1: Create results folder

mkdir -p /home/cwd/jmeter\_results/html\_report

### Step 2: Run test in **Non-GUI (headless) mode**

/opt/jmeter/bin/jmeter \

-n \

-t /home/cwd/Downloads/Performancss/performance\_test\_plan.jmx \

-l /home/cwd/jmeter\_results/result.jtl \

-e \

-o /home/cwd/jmeter\_results/html\_report

### Step 3: Open the HTML Report

xdg-open /home/cwd/jmeter\_results/html\_report/index.html

## 📊 What You’ll See in Report

* Summary Dashboard: Test duration, errors, users
* Throughput / Response Time Graphs
* Percentiles: 90%, 95%, 99%
* Failures: With cause and response times

## 💡 Optional Automation Script

Create a run\_performance\_test.sh:

#!/bin/bash

echo "Running JMeter performance test..."

mkdir -p ~/jmeter\_results/html\_report\_$(date +%F\_%H-%M)

jmeter -n \

-t ~/Downloads/Performancss/performance\_test\_plan.jmx \

-l ~/jmeter\_results/result\_$(date +%F\_%H-%M).jtl \

-e \

-o ~/jmeter\_results/html\_report\_$(date +%F\_%H-%M)

echo "Done! Report saved in ~/jmeter\_results/"

Make it executable:

chmod +x run\_performance\_test.sh

Run:

./run\_performance\_test.sh

## ✅ Summary

| Step | Description |
| --- | --- |
| 1 | Install Java & JMeter |
| 2 | Create Login + Devices page test plan |
| 3 | Add Headers, Cookies, Assertions |
| 4 | Save .jmx plan |
| 5 | Run via CLI and generate .jtl + HTML report |
| 6 | Open report for analysis |

Would you like this in a downloadable PDF or Word format for sharing with your team?

Excellent question! You’re now moving from basic performance testing → into realistic load simulation and advanced test scenarios.

## 💥 How to Simulate Multiple Users in JMeter

You already have a basic setup. Now you’ll simulate real-world multiple user credentials for login and track how the system performs.

### ✅ Method 1: Using CSV Data Set Config (for multiple usernames/passwords)

### 📁 Step 1: Create a CSV file

Example: users.csv  
Place it in /home/cwd/Downloads/Performancss/

username,password

user1@example.com,pass1

user2@example.com,pass2

user3@example.com,pass3

...

### 🔧 Step 2: In JMeter, add **CSV Data Set Config**

1. Right-click on Thread Group → Add → Config Element → CSV Data Set Config
2. Set:

| Field | Value |
| --- | --- |
| Filename | /home/cwd/Downloads/Performancss/users.csv |
| Variable Names | username,password |
| Delimiter | , |
| Recycle on EOF | True |
| Stop thread on EOF | False |
| Sharing Mode | All threads |

### 🔄 Step 3: Use CSV variables in Login request

Update your Login POST body like this:

{

"username": "${username}",

"password": "${password}"

}

Now, every thread/user will pick a unique login from the CSV and hit the API.

## 📈 Advanced Performance Testing Ideas

Here are several realistic test cases you can simulate after login:

### 💡 Idea 1: **Login → Open Devices Page → View Device Detail**

Chain requests like:

1. POST /auth/login (gets auth token)
2. GET /home/entity-management/devices
3. GET /api/devices/:device\_id/overview

You can pass tokens using:

* JSON Extractor (for Bearer Token)
* HTTP Header Manager (to pass Authorization: Bearer <token>)

### 💡 Idea 2: **Test with Increasing Load (Ramp Test)**

Create a plan with:

| Threads | Ramp-up (s) | Purpose |
| --- | --- | --- |
| 50 | 30 | Light load |
| 100 | 60 | Moderate |
| 200 | 90 | Stress |
| 500 | 120 | Max capacity |

You can also use Stepping Thread Group (via plugin) for gradual user increase.

### 💡 Idea 3: **Test API Error Handling**

Provide:

* Invalid usernames/passwords
* Expired tokens
* Invalid URLs

Check how the system handles:

* 401 Unauthorized
* 403 Forbidden
* 500 Internal Server Errors

Use Assertions to track response codes and error %.

### 💡 Idea 4: **Test Under Network Delay (Think Time)**

Add Timers to simulate real user delay:

1. Right-click Thread Group → Add → Timer → Constant Timer
2. Add 1-3 seconds delay before each request

Helps model real-world pacing between user actions.

### 💡 Idea 5: **Compare Performance for Admin vs Regular User**

Create different CSVs for:

* Admin credentials
* Viewer credentials

Route different users through different thread groups or logic controllers.

## 🔁 Optional: Parameterize Dynamic Device IDs

If your devices are dynamic:

* Extract device IDs from /devices API using JSON Extractor
* Pass them to next request using ${deviceId}

## 📊 Optional: Export Results for Analysis

You can export .jtl to CSV and open in Excel for graphing or trends.

cat result.jtl > result.csv

Or use Python/Excel to:

* Filter slow requests
* Calculate 95th percentile
* Graph response times over time

## ✅ Next Steps You Can Try

| Task | Description |
| --- | --- |
| 🧪 Simulate 1000 users | Using Thread Group or Ultimate Thread Group |
| 📊 Monitor server CPU/memory | With tools like Grafana + InfluxDB |
| 🔁 Schedule test via CRON | Automate daily/nightly performance runs |
| 🧼 Auto archive old reports | Use shell scripts to manage old test runs |
| 🕵️ Detect memory leaks | Long-duration test (1–3 hours) with periodic load |
| 🚀 Test concurrent logins | Multiple users logging in at once using CSV |

Would you like me to:

1. Generate an advanced .jmx file with login + CSV + token + chained flow?
2. Give a bash script to automate test + HTML report + open?

Let me know your next goal — I’ll guide you fully!

This error log is very helpful, and the issue is clear:

## ❌ Problem: Response Code 400 – Bad Request

{"status":400,"message":"Bad Payload"}

This means your login request reached the server, but the format or structure of the JSON payload is invalid, even though it looks right.

## ✅ Most Likely Fixes

### ✅ 1. **Wrong Content-Type Header**

Even though you're sending JSON, the server might reject it if the header is not correct.

### 🔧 Check Your Header Manager:

You must have this exact header:

Content-Type: application/json

Also add:

Accept: application/json

If you missed adding it or typed content-type in lowercase, the server could treat it as malformed.

### ✅ 2. **Trailing Spaces / Bad Quotes in JSON**

Your POST data should exactly be like this:

{

"username": "${username}",

"password": "${password}"

}

If you hardcoded like:

{

"username": "rohan.kamble@cwdin.com",

"password": "Rohan@12345"

}

…make sure:

* No extra spaces or weird characters
* You're using straight double-quotes ", not curly quotes “ ”

Also check there’s no hidden newline/CRLF inside the payload box.

### ✅ 3. **Add** HTTP Header Manager **at the Right Level**

Add it under the Login HTTP Request like this:

Header Manager Config:

| Name | Value |
| --- | --- |
| Content-Type | application/json |
| Accept | application/json |

## ✅ How to Fix in JMeter GUI (Step-by-Step)

1. Go to Login HTTP Request
2. Make sure Method = POST
3. Set:
   * Protocol: https
   * Server: web.sm.test.cwdin.com
   * Path: /api/v1/user/auth/login
4. Switch to Body Data tab  
   Paste this exactly:

{

"username": "${username}",

"password": "${password}"

}

1. Right-click on this request → Add → Config Element → HTTP Header Manager  
   Add two headers:
   * Content-Type = application/json
   * Accept = application/json

## 🧪 Re-test

Now run the test again with:

/opt/jmeter/bin/jmeter \

-n \

-t /home/cwd/Downloads/Performancss/performance\_test\_plan.jmx \

-l /home/cwd/jmeter\_results/retest.jtl \

-e \

-o /home/cwd/jmeter\_results/html\_report\_retest

Then open:

xdg-open /home/cwd/jmeter\_results/html\_report\_retest/index.html

## ✅ If Still Failing:

Let me know and I’ll help you:

* Debug exact payload being sent
* Print actual request using Debug Sampler
* Use View Results Tree to inspect request/response headers

You’re very close — just a small config or formatting tweak left!

Want me to generate a sample .jmx file with working login + header + CSV config?