11 – JMeter CLI & Azure Load Testing

🎯 Objective

Understand how to run JMeter **without the GUI** (CLI Mode) for maximum performance, and how to take that same script and scale it to thousands of users using **Azure Load Testing**.

1️⃣ Why CLI (Non-GUI) Mode?

We established earlier that the JMeter GUI consumes too much memory for heavy load.

- **GUI Mode:** For designing, debugging, and small tests (1-50 users).
- **CLI Mode:** For actual load testing (50 - 10,000+ users).

The Golden Rule:

> "Don't load test with the GUI open."

2️⃣ Running JMeter from Command Line

To run a test in CLI mode, you use the `jmeter` command (or `jmeter.sh` / `jmeter.bat`).

🔹 The Command Syntax

```
jmeter -n -t test_plan.jmx -l results.jtl -e -o ./report
```

🔹 Flag Breakdown (Memorize This)

- `-n` : **Non-GUI mode** (Run headless).
- `-t [file]` : Path to your **Test Plan (.jmx)**.
- `-l [file]` : Path to output the **Log file (.jtl)**. This contains raw data.
- `-e` : Generate the **HTML Dashboard Report** at the end.
- `-o [folder]` : Output folder for the HTML report.

🔹 Example Output

When you run this, you won't see graphs. You will see a text summary:

```
Creating summariser <summary>
Created the tree successfully using test_plan.jmx
Starting the test @ Mon Jan 01 12:00:00 UTC 2024 (1704110400000)
summary +     50 in 00:00:10 =    5.0/s Avg:   200 Min:   100 Max:   500
summary +    150 in 00:00:20 =    7.5/s Avg:   198 Min:    99 Max:   450
...
```

3️⃣ The HTML Dashboard Report

The `-e -o ./report` flags generate a beautiful HTML folder.

Open `index.html` in your browser.

You will see:

- **APDEX Score:** User satisfaction score.
- **Response Time Over Time:** Graphs showing latency trends.
- **Throughput:** Requests per second graphs.
- **Error Analysis:** Pie charts of error codes (404, 500, etc.).

👉 **This is what you send to your manager.**

4️⃣ Introduction to Azure Load Testing (ALT)

Your laptop has limits (CPU/RAM). You probably can't simulate 50,000 users.
Azure Load Testing is a fully managed service that runs JMeter scripts for you in the cloud.

🔹 How It Works

1. **Engine:** Azure spins up compute engines (internally uses JMeter).
2. **Script:** You upload your existing `.jmx` file.
3. **Scale:** You say "I want 50 engines". Azure splits the load automatically.
4. **Analysis:** It provides built-in dashboards (graphs, percentiles, errors) inside the Azure Portal.

🔹 Does it require code changes?

NO.
If your script works locally in JMeter, it works in Azure Load Testing.

- *Note: If you use CSV files, you must upload them alongside the JMX.*

5️⃣ Moving from Local to Azure (Workflow)

1. **Develop Locally:** Create `test.jmx` on your laptop. Validate with 10 users.
2. **Create ALT Resource:** Go to Azure Portal → Search "Azure Load Testing" → Create.
3. **Create Test:**

- Click "Create" → "Upload a script".
- Select `test.jmx` .
- Select `users.csv` (if using parameterization).

4. **Configure Load:**

- **Engine Instances:** 1 engine ≈ 250 threads (rough estimate).
- If you want 1,000 users, ask for 4 engines.

5. **Run:** Azure provisions the VMs, runs the `jmeter -n -t ...` command internally, and aggregates the results.

6️⃣ Azure-Specific Features

- **Server-Side Monitoring:** ALT can connect to your App Service / AKS / SQL Database and show you **CPU %** and **Memory %** alongside your JMeter response times.

- *Why is this cool?* You can see: "At 12:05, Response Time spiked to 5s **BECAUSE** SQL CPU hit 100%."

- **CI/CD Integration:** You can trigger ALT from Azure DevOps Pipelines or GitHub Actions (covered in Module 13).

7️⃣ Interview Question

"I have a JMeter script. How do I run it for 100,000 users?"

Answer:
"I cannot do that on a single machine due to port exhaustion and CPU limits. I would use a distributed

testing solution like **Azure Load Testing**. I would upload the JMX script, split the CSV data (if necessary), and configure enough Engine Instances to generate the required concurrency."

🎯 Mini Exercise

1. Open your terminal/cmd.
2. Navigate to your script folder.
3. Run the CLI command:
   `jmeter -n -t mock-working.jmx -l result.jtl`
4. Wait for it to finish.
5. Look at the `result.jtl` file. It's just a CSV!

- *Lesson: JMeter results are just text data.*