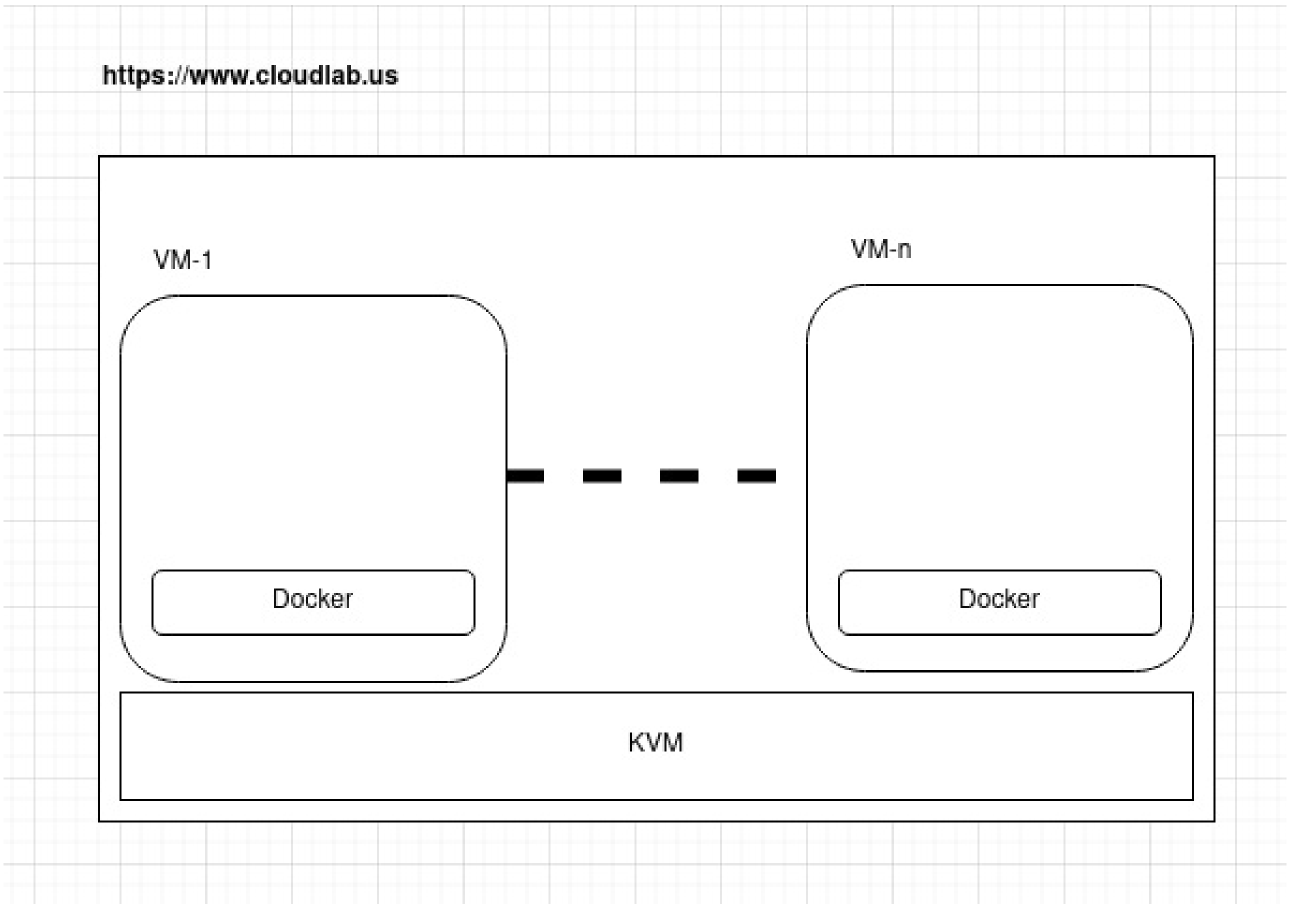
TEST IN 8 STEPS

Step 1: Define the Goals

- What do you want to test? Examples:
  - How many concurrent users the app can handle.
  - How fast the app responds under load.
  - How the database performs during read/write-heavy workloads.
- Identify key metrics:
  - **Latency** (response time).
  - **Throughput** (data processed per second).
  - **Requests per second** (RPS).
  - **Error rate**.

Step 2: Prepare the Environment

- **Use a Testing Environment**: Avoid running benchmarks on production systems.
- **Preload Data**:
  - Populate the database with realistic data.
  - Include a mix of scenarios (e.g., small and large records).
- **Mock External Services**:
  - If your app calls APIs or services, mock them to isolate your tests.



Step 3: Choose Your Tools

- **General Load Testing**:
    - Locust (https://locust.io/): Simulate user behaviors in Python.
    - Apache JMeter (https://jmeter.apache.org/): GUI-based, multi-protocol, versatile load testing.
- **Throughput-Oriented Tools**:
    - wrk2 (https://github.com/giltene/wrk2): For sustained RPS testing.
    - Vegeta: A simple HTTP load testing tool.
- **Scalability Testing**:
    - k6 (https://k6.io/open-source/): For api testing,...
- **Monitoring Tools**:
    - Grafana/Prometheus for live monitoring.
    - Sar for system monitoring (cpu, disk, memory, I/0)
    - ...etc.

Step 4: Design the Test Scenarios

1. **Read-Heavy**: Simulate mostly data reads from the database.

    - Example: 90% reads, 10% writes.

2. **Write-Heavy**: Focus on inserting/updating data.
    - Example: 70% writes, 30% reads.
3. **Balanced Load**: Equal distribution of reads and writes.
4. **Concurrent Users**: Simulate different user loads:
    - Low: 10 users.
    - Medium: 100 users.
    - High: 1,000+ users.
5. **Sustained Load**:
    - Test for hours to identify long-term issues (e.g., memory leaks).

Step 5: Run the Benchmark

1. **Start Small**:
    - Begin with low traffic to establish a baseline.
2. **Gradually Increase Load**:
    - Add more users or requests per second to observe system limits.
3. **Monitor Key Metrics**:
    - Latency, throughput, error rates, and system resource usage (CPU, RAM, Disk I/O, Network).

Step 6: Analyze Results

- **Look for Bottlenecks**:
    - High database query times.
    - Slow API endpoints.
    - Resource constraints (e.g., 100% CPU usage).
- **Interpret Metrics**:
    - High latency: Indicates delays in processing.
    - Low throughput: The system might be overwhelmed.
    - High error rates: The system is failing under load.

Step 7: Optimize and Retest

- Common Optimizations:
    - **Caching**: Store frequent queries in memory.
    - **Database Tuning**: Add indexes or optimize queries.
    - **Scaling**: Add more servers (horizontal scaling) or resources (vertical scaling).
    - **Connection Pooling**: Efficiently manage database connections.

- **Retest After Changes**: Ensure optimizations work.

Step 8: Document and Share Results

- Include:
  - Test configurations (tools, user scenarios).
  - Key metrics and graphs (e.g., latency, RPS).
  - Observed bottlenecks and applied optimizations.
- Share your findings with the team.

Best Practices

- Use **realistic workloads** to simulate real-world usage.
- Always **warm up the application** before testing.
- Run tests multiple times to ensure **consistent results**.
- Avoid benchmarking on **shared environments** (e.g., your laptop during normal usage).

Quelques liens

- https://jmeter.apache.org/usermanual/get-started.html (https://jmeter.apache.org/usermanual/get-started.html)
- https://alimco.in/WriteReadData/CMS/jmeter_quick_guide.pdf (https://alimco.in/WriteReadData/CMS/jmeter_quick_guide.pdf)
- https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf (https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf)
- https://sqa.jdn.gov.my/images/bootcamp_pt/silibus_SUKN9/IntroductionJMeter_Siri12021.pdf (https://sqa.jdn.gov.my/images/bootcamp_pt/silibus_SUKN9/IntroductionJMeter_Siri12021.pdf
- https://docs.locust.io/en/stable/ (https://docs.locust.io/en/stable/)
- http://alecoledelavie.com/accueil/vie_uploads/Portfolio_Programs_Projects_and%20BAU/PortF (http://alecoledelavie.com/accueil/vie_uploads/Portfolio_Programs_Projects_and%20BAU/PortI
- https://medium.com/@ravipatel.it/step-by-step-guide-to-load-testing-with-k6-5afb625e231a (https://medium.com/@ravipatel.it/step-by-step-guide-to-load-testing-with-k6-5afb625e231a)
- https://k6.io/open-source/ (https://k6.io/open-source/)
- https://www.datadoghq.com/blog/collecting-mysql-statistics-and-metrics/ (https://www.datadoghq.com/blog/collecting-mysql-statistics-and-metrics/)
- https://dev.mysql.com/downloads/benchmarks.html (https://dev.mysql.com/downloads/benchmarks.html)
- https://scalegrid.io/blog/how-to-benchmark-mongodb-with-ycsb/ (https://scalegrid.io/blog/how-to-benchmark-mongodb-with-ycsb/)