# 🚀 Performance Testing Process

A **well-defined process** ensures accuracy, repeatability, and actionable insights. The eight-phase approach below is **tool-agnostic** and applicable across Agile, Waterfall, or DevOps methodologies.

**📌 Phase 1: Planning and Strategy**

**🎯 Objective:** Define scope, goals, and success metrics before writing a single script.

**🔹 Key Activities:**

* **Define Performance Requirements/SLAs**  
  *Example:* "Login must complete in < 2 seconds for 1,000 concurrent users."
* **Identify Critical Business Scenarios**  
  *Example:* For a food delivery app: login, browse restaurants, apply filters, add items, place order.
* **Understand System Architecture**  
  *Example:* Architecture includes load balancer → app tier → Redis cache → PostgreSQL DB → payment gateway API.
* **Resource Identification**  
  *Example:* Assign 1 QA Engineer, 1 DevOps Engineer, and select AWS staging environment.
* **Risk Assessment**  
  *Example:* Checkout and payment flows are considered high risk due to external dependencies.
* **Tool Selection**  
  *Example:* Choose JMeter for load testing, Grafana + Prometheus for monitoring.

**📌 Phase 2: Workload Modeling**

**🎯 Objective:** Simulate user activity patterns based on real usage.

**🔹 Key Activities:**

* **Gather Usage Stats**  
  *Example:* Google Analytics shows peak usage at 7 PM with 3,000 concurrent users.
* **Define User Personas & Paths**  
  *Example:* Guest user flow vs. registered user flow with saved address and card.
* **Transaction Mix**  
  *Example:* 60% browse → 30% add to cart → 10% checkout.
* **Think Time & Pacing**  
  *Example:* Think time = 3 seconds; Pacing = 10 seconds per transaction loop.
* **Data Volume Estimation**  
  *Example:* 10,000 unique products and 5,000 test users seeded into test DB.

**📌 Phase 3: Test Script Development**

**🎯 Objective:** Create parameterized, scalable, and accurate scripts.

**🔹 Key Activities:**

* **Script Recording/Coding**  
  *Example:* Record product search and checkout flow in JMeter or k6.
* **Enhancements**
  + *Parameterization:* Use CSV for dynamic product IDs.
  + *Correlation:* Extract session token and reuse in headers.
  + *Assertions:* Validate 200 OK and “Order Confirmed” in response.
  + *Error Handling:* Retry once if a network timeout occurs.
* **Dry Runs & Debugging**  
  *Example:* Run the script for 1 user to verify flow, assertions, and cookies.

**📌 Phase 4: Test Environment Setup**

**🎯 Objective:** Ensure the environment mirrors production as closely as possible.

**🔹 Key Activities:**

* **Provision Infrastructure**  
  *Example:* 3 app servers, 1 DB replica, 2 load generators on separate VMs.
* **Test Data Preparation**  
  *Example:* Populate with 50,000 customers, 100,000 SKUs, and order history.
* **Tool Configuration**  
  *Example:* JMeter set to 10 threads per engine across 5 engines.
* **Environment Isolation**  
  *Example:* Dedicated staging URL staging.company.com with VPN access.
* **Monitoring Setup**  
  *Example:* Prometheus to collect CPU/Memory; Dynatrace for APM insights.

**📌 Phase 5: Test Execution**

**🎯 Objective:** Run tests based on types and gradually increase load.

**🔹 Key Activities:**

* **Baseline Test**  
  *Example:* Run with 10 users to confirm system and script stability.
* **Gradual Load Increase**  
  *Example:* 500 → 1,000 → 2,500 → 5,000 VUs across Load, Stress, Spike scenarios.
* **Monitor in Real Time**  
  *Example:* CPU spikes to 85% when TPS reaches 300.
* **Capture Metrics**  
  *Example:* Collect average response time, 95th percentile, error %, and resource usage.
* **Troubleshoot During Test**  
  *Example:* Stop and restart DB pod due to connection pool saturation.

**📌 Phase 6: Monitoring and Data Collection**

**🎯 Objective:** Gather complete system-level and application-level telemetry.

**🔹 Key Activities:**

* **Server Monitoring**  
  *Example:* CPU = 80%, Memory = 60%, Disk I/O = 300MB/s on app server.
* **Database Monitoring**  
  *Example:* Slow query logs show cart API has a join that takes 1.2 seconds.
* **Application Monitoring (APM)**  
  *Example:* Dynatrace highlights a memory leak in the product page API.
* **Network Monitoring**  
  *Example:* Bandwidth saturation observed on ingress at 9 PM test window.
* **Log Analysis**  
  *Example:* Server logs reveal “502 Gateway Timeout” under load.

**📌 Phase 7: Analysis and Reporting**

**🎯 Objective:** Turn raw data into insights and communicate them clearly.

**🔹 Key Activities:**

* **Data Correlation**  
  *Example:* Response time spikes when GC time exceeds 800ms → JVM tuning required.
* **Identify Bottlenecks**  
  *Example:* Cart service fails under 3,000 users due to missing DB index.
* **Compare to SLAs**  
  *Example:* SLA requires 90% checkout < 3s → test shows only 75% meet this.
* **Generate Reports**
  + **Executive Summary:**  
    *“Checkout SLA not met. Bottleneck in cart DB query. Recommend indexing + connection pool tuning.”*
  + **Detailed Metrics:**
    - Avg Response Time: 1.8s
    - TPS: 250
    - Error Rate: 2.3%
    - CPU: 85%, Memory: 78%
* **Present to Stakeholders**  
  *Example:* Use slide deck with charts + written summary + prioritized fixes.

**📌 Phase 8: Tuning, Optimization, and Re-testing**

**🎯 Objective:** Fix issues and confirm improvements.

**🔹 Key Activities:**

* **Implement Fixes**  
  *Example:* Added index on cart\_items.product\_id; increased thread pool from 100 to 200.
* **Re-run Performance Tests**  
  *Example:* Response time dropped from 3.2s → 1.6s after DB fix.
* **Regression Check**  
  *Example:* Confirm payment flow still passes with original scripts post tuning.

**🔁 Summary**

| **Phase** | **Purpose** | **Example** |
| --- | --- | --- |
| **1. Planning** | Set goals | SLA = 2s for login |
| **2. Modeling** | Simulate user load | 60% browse, 30% cart, 10% checkout |
| **3. Scripting** | Automate actions | Parameterize product IDs |
| **4. Environment** | Mirror production | 2 app servers, 1 DB |
| **5. Execution** | Apply load | Spike test to 10,000 users |
| **6. Monitoring** | Collect metrics | CPU, Memory, Logs |
| **7. Analysis** | Identify bottlenecks | DB joins = slow |
| **8. Tuning** | Optimize and validate | Add index, rerun test |