Gatling User Guide

1. Introduction

Gatling is an open-source load and performance testing tool designed for ease of use, high performance, and maintainability. It is commonly used to test APIs, web applications, and services by simulating virtual users and generating detailed reports.

This guide provides a structured approach for setting up, writing, and running Gatling simulations.

---

2. Prerequisites

- **Java:** Version 11 or higher (Gatling supports up to Java 21).
- **Build Tools:** Maven or Gradle (Maven examples are used here).
- **IDE:** IntelliJ IDEA or Eclipse (optional but recommended).
- **Network Access:** Ensure internet access for dependencies.

---

3. Project Structure

A typical Maven-based Gatling project has the following layout:

```
project-root/
├── src/test/java        # Java-based simulations
├── src/test/resources   # Data files (CSV, JSON, etc.)
├── pom.xml              # Maven configuration
```

---

4. Running Simulations

Quick Start

Run all simulations from the project root:

```
./mvnw gatling:test
```

Run a specific simulation:

```
./mvnw gatling:test -Dgatling.simulationClass=examples.BasicSimulation
```

Adding Descriptions

```
./mvnw gatling:test -Dgatling.simulationClass=examples.BasicSimulation -
Dgatling.runDescription="Smoke Test"
```

---

5. Workload Models

Open Model (Arrival Rate)

Injects users at a fixed rate, regardless of response times.

```
rampUsers(10).during(60) // 10 users per second for 1 minute
```

Use case: Modeling incoming traffic.

Closed Model (Concurrent Users)

Maintains a constant number of active users.

```
rampConcurrentUsers(1).to(50).during(5)
constantConcurrentUsers(50).during(10)
```

Use case: Capacity planning, stability testing.

---

6. Using CSV Data (Feeders)

Internal CSV

Place CSV file in `src/test/resources/`:

```
val csvFeeder = csv("data.csv").circular
feed(csvFeeder)
.exec(http("Get Case").get("/api/cases/#{caseId}"))
```

External CSV

Pass path at runtime:

```
./mvnw gatling:test -Dcsv.path=/path/to/data.csv
```

In code:

```
val csvFeeder = csv(System.getProperty("csv.path", "data.csv")).circular
```

---

7. Reports

Gatling automatically generates HTML reports.

- **Output directory:**

  ```
  target/gatling/<simulation-name>-<timestamp>/index.html
  ```

- **Regenerate report:**

  ```
  ./mvnw gatling:test -Dgatling.reportsOnly=target/gatling/<simulation-name>-
  <timestamp>
  ```

---

8. Assertions & SLAs

Add assertions to enforce SLAs:

```
assertions(
  global.responseTime.max.lt(10000),
  global.responseTime.mean.lt(5000)
)
```

---

9. Troubleshooting

- **401 Unauthorized:** Check authentication tokens or secrets.

- **Proxy Errors:** Update/remove proxy configuration.
- **Java Issues:** Ensure `java -version` is 11+.
- **Multiple Simulations:** Specify with `-Dgatling.simulationClass`.

---

10. Best Practices

- Externalize secrets (use environment variables or system properties).
- Use CI pipelines to generate and publish reports.
- Start with small loads before ramping up.
- Hold steady states for at least 3–5 minutes.
- Use feeders for realistic test data.

---

11. Next Steps

- Parameterize base URLs and endpoints.
- Integrate Gatling into CI/CD pipelines.
- Explore advanced injection models (stress, soak, spike testing).

End of Document