Gatling

2026

DATASHEET

Top load testing metrics finance companies need to follow

For performance engineering, platform, and QA teams in financial services

TOP LOAD TESTING METRICS FINANCE COMPANIES NEED TO FOLLOW

Between strict SLAs, regulatory scrutiny, non-internet-facing infrastructure, and the encryption overhead that comes with every payment flow, finance teams need a different lens on performance data, not just pass/fail results, but rich, contextualized metrics that reflect the realities of banking, trading, and fintech systems.

Gatling Enterprise Edition gives you that full picture: from transaction-level behavior to network and infrastructure signals, all tuned for the demands of financial services environments.

1. Transactions per Second (TPS) & Response Throughput

What it is

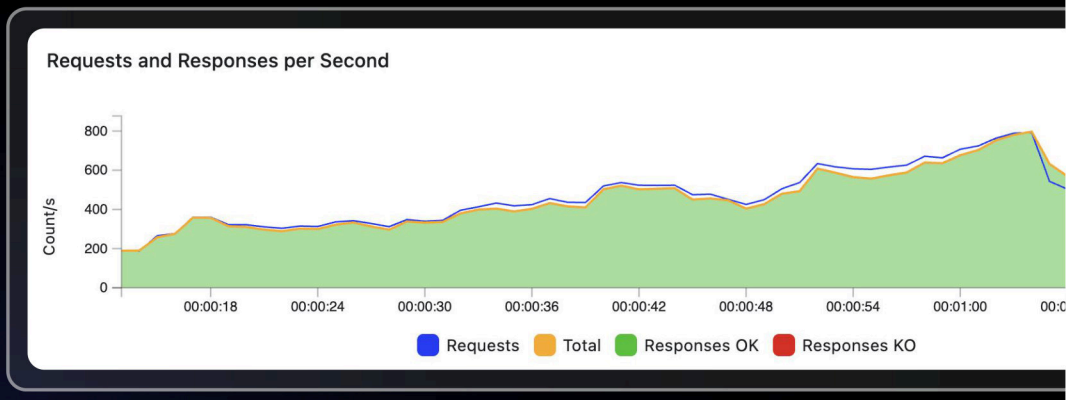
Measures the volume of financial transactions processed and responses returned every second during a load test.

What it reveals

TPS trends expose where the system hits its ceiling under sustained load. If the transaction rate continues rising while the response curve plateaus or drops, it signals thread pool exhaustion, backend saturation, or queue overflow.

Pro tips

- Target sustained TPS over 30–60 minute endurance runs, not just spike tests
- Overlay TPS with concurrent session count and error rate to pinpoint the exact inflection point before collapse
- Validate your target TPS against your SLA ceiling and document both numbers before each test run



2. Response Time Percentiles (P95 / P99)

What it is

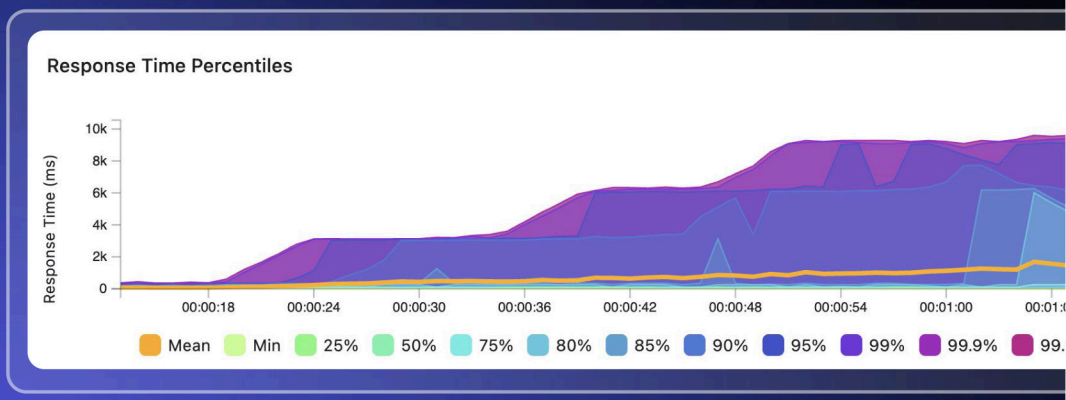
Measures latency for the slowest users — the 95th and 99th percentile — rather than averages. In financial services, these percentiles are typically what SLAs and regulatory commitments are written against.

What it reveals

Percentiles expose tail latency, which is what real users experience in worst-case scenarios. A rising P99 with a stable mean is a classic early warning sign of a regression. In payment systems, that P99 spike can mean timeouts, failed transactions, and contractual breaches.

Pro tips

- Always track P95 and P99 during ramp-up and peak phases.
- Run tests outside market hours, but design scenarios to simulate market-hours load profiles. The system needs to prove it can sustain SLAs under realistic conditions
- A stable mean with a climbing P99 is your earliest signal of a problem. Don't wait for averages to move.



3. Error Rate & HTTP Status Code Distribution

What it is

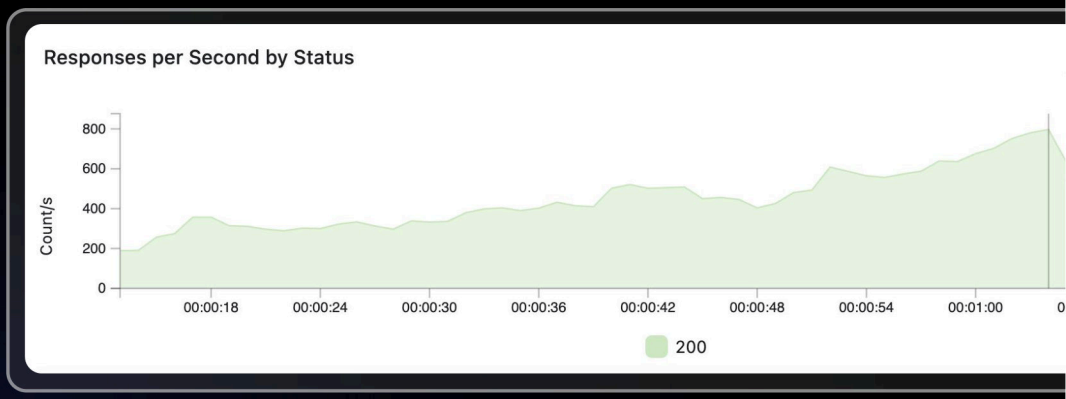
Monitors the volume of errors per second and breaks down responses by HTTP status codes across financial transaction flows.

What it reveals

A 5xx surge on a payment endpoint means money movement has failed. A 4xx spike often reveals authentication or session management issues, which are particularly common in banking apps with complex state management.

Pro tips

- Slice errors by business flow like payment submission, account creation, login, balance retrieval
- A sudden 5xx surge combined with flattening TPS is the signature of backend collapse
- Track 4xx errors separately in session-heavy banking applications. Rising 4xx often signals that your load tool is failing to maintain session state correctly.



4. Session & Concurrency Metrics (Arrival, Termination & Active Users)

What it is

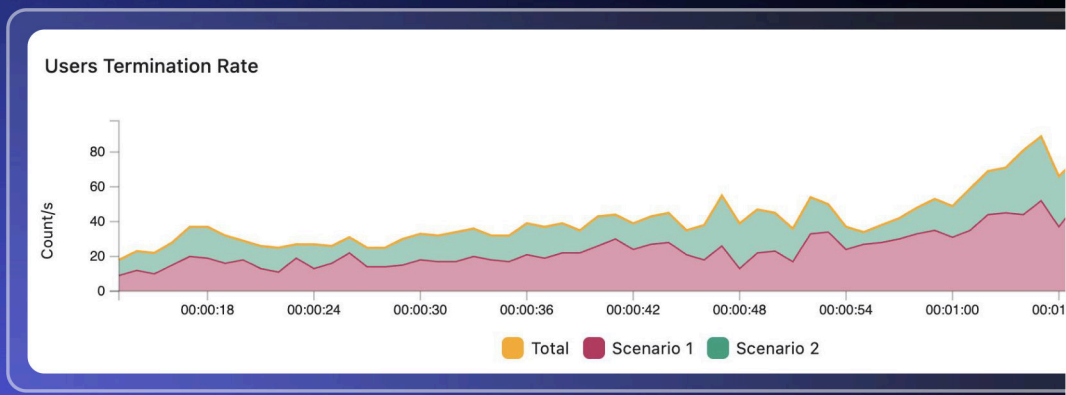
Tracks how many virtual users start (arrival), finish (termination), and remain active simultaneously (concurrency) across the test duration.

What it reveals

Diverging arrival and termination curves signal that users are getting stuck: slow transactions, session timeouts, or flows that can't complete cleanly under load. This is exactly the failure pattern that generic load tools miss or misattribute.

Pro tips

- Use concurrency alongside P99 response times to pinpoint your capacity ceiling. An unexpected plateau in termination rate usually means sessions are being held open, often a sign of a backend thread or database lock.
- Banking apps with server-side state management are particularly vulnerable to session miscounting. If your tool can't persist session values correctly across page-refresh-triggered flows, your concurrency data will be wrong.



5. Business Transaction Duration (Group Duration Percentiles)

What it is

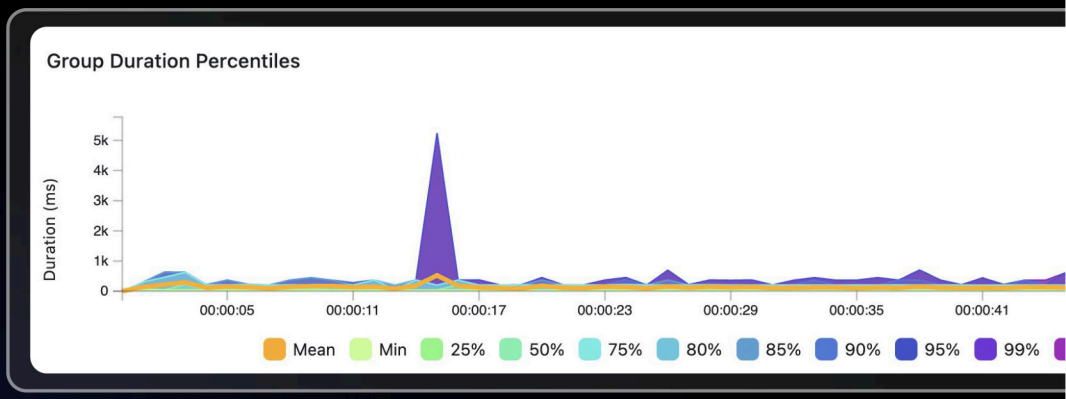
Measures end-to-end duration across defined groups of requests mapped to financial user journeys like payment submission, account opening, balance check, KYC flow, fund transfer.

What it reveals

Group metrics expose compounded latency across multiple services and steps that single-request timings miss entirely. A payment submission might touch authentication, fraud screening, core banking, and a notification service in sequence.

Pro tips

- Define groups around your regulatory-critical and revenue-critical flows first. Optimize for what matters to the business, not just what's easiest to instrument.
- Use group percentiles to benchmark before every release.
- Use Gatling's SLOs to gate releases on global P99 response time and error ratio thresholds.



6. TCP Connect Duration Percentiles

What it is

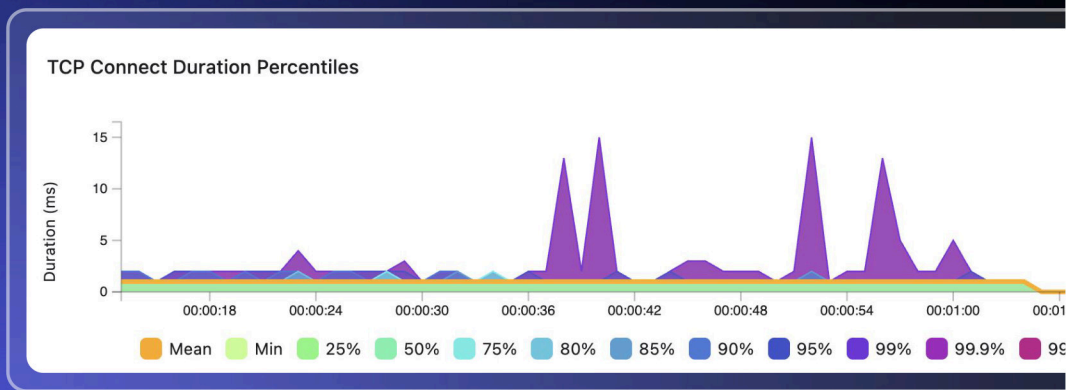
Measures the time taken to establish TCP connections from the load generator to the system under test.

What it reveals

TCP connect time is a direct signal of network layer health non-internet environments. Rising TCP connect times before response times climb is almost always a network or infrastructure problem: connection pool exhaustion, firewall limits, or generator saturation.

Pro tips

- When running with private locations inside your network, watch TCP connect percentiles during ramp-up before anything else. If they move first, investigate infrastructure before blaming the app.
- Elevated TCP connect times across all endpoints simultaneously usually means the load generator is the bottleneck. This is worth ruling out before escalating to the platform team.



7. TLS Handshake Duration Percentiles

What it is

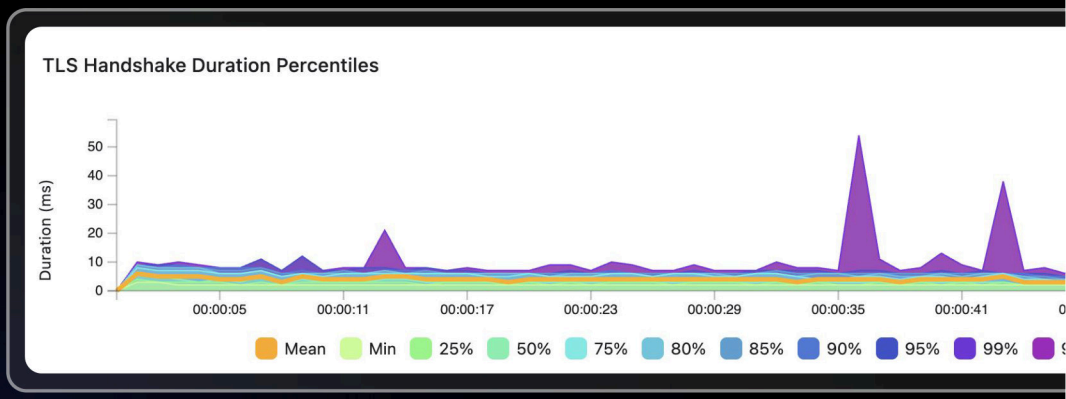
Tracks the duration of SSL/TLS handshakes during secure connection establishment.

What it reveals

Custom security libraries, heavy cipher suites, mutual TLS, and certificate chain complexity are standard in banking infrastructure. At scale, TLS overhead can dominate load generator CPU before the application under test shows any sign of stress.

Pro tips

- Baseline TLS handshake times early in your test program and watch them closely during high-concurrency runs. These are often invisible in APM tools but clearly visible in Gatling's client-side view.
- In environments with custom encryption libraries, allocate generators by CPU capacity, not virtual user count. Finance teams frequently need 32-core equivalent because of TLS overhead.



8. Load Generator Infrastructure Metrics (CPU & Heap)

What it is

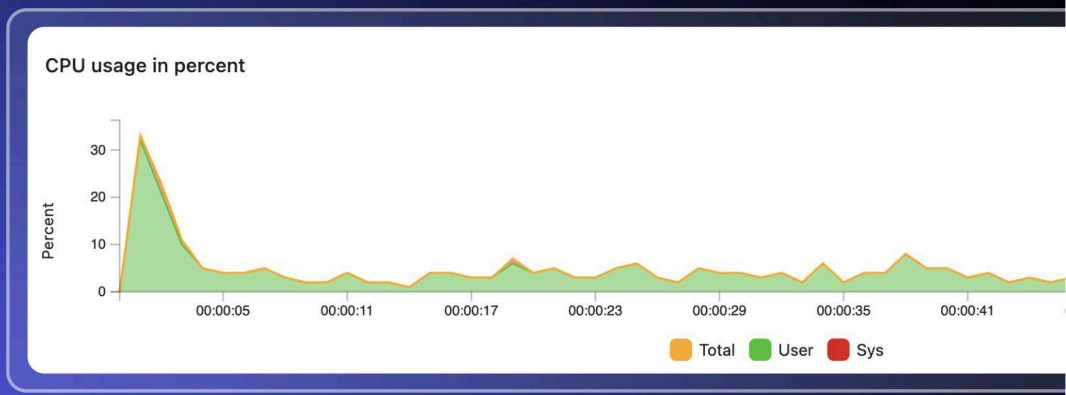
Monitors CPU usage and memory consumption on the load generators themselves during test execution.

What it reveals

The encryption overhead that defines finance environments hits the generator hard. A saturated generator introduces artificial latency and fails to maintain target injection rates, producing results that look like application failures but are actually tooling problems.

Pro tips

- Always check generator health before attributing results to the system under test. Sustained CPU above 80% or heap above 85% means your metrics are unreliable.
- Finance environments typically require fewer, larger generator instances rather than many small ones. Size for encryption load, not just virtual user count.
- Monitor generator metrics continuously across endurance runs, not just at peak. Heap exhaustion often creeps in after 20–30 minutes of sustained load



How to go further

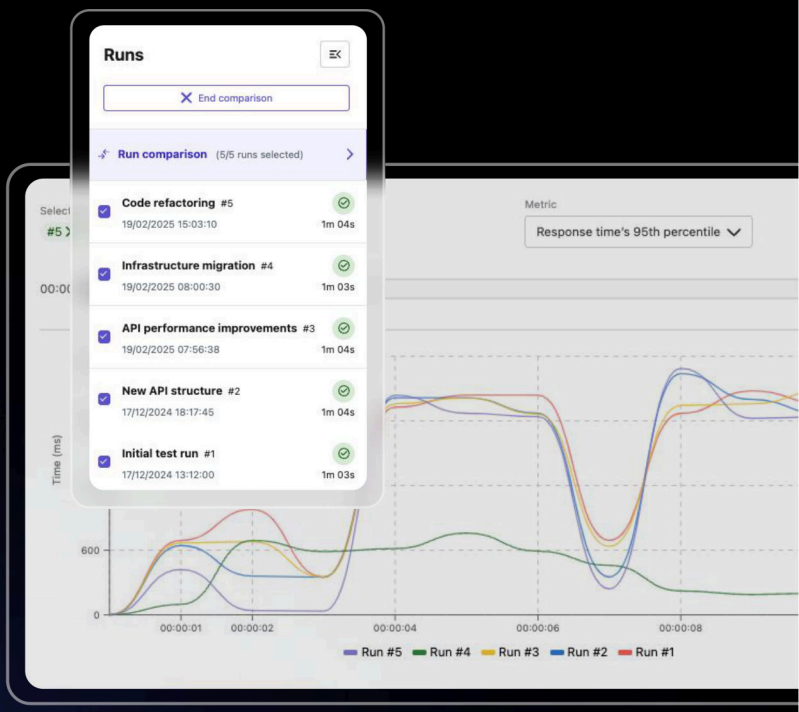
Test in your environment, not around it

Most financial systems are never exposed to the public internet. Your load testing infrastructure needs to match: private locations that run inside your network, behind your firewall, against the actual systems your users hit. Testing through a public proxy or against a stripped-down staging environment produces metrics that don't transfer to production.

Observe trends across release cycles, not just individual runs

Load testing isn't just about validating a single release, it's about understanding how your system evolves under pressure. By analyzing performance trends across multiple runs, teams can detect subtle regressions, capacity drifts, or architectural bottlenecks that might otherwise go unnoticed.

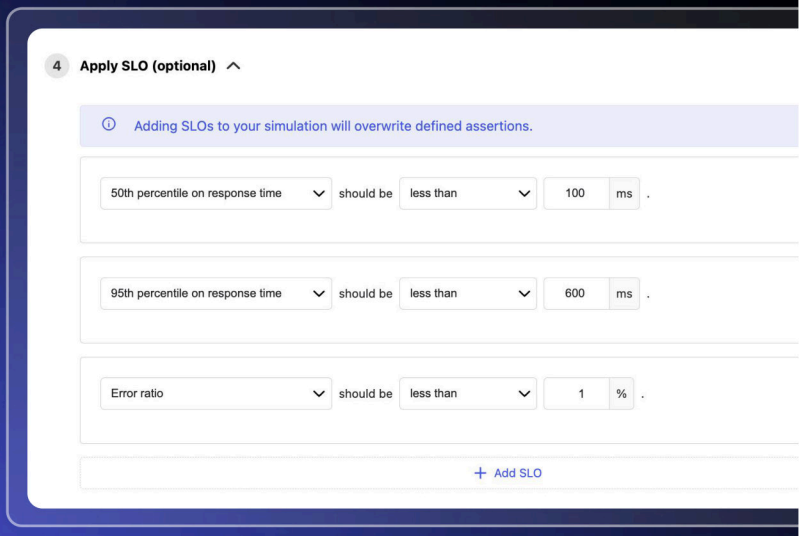
With Gatling Enterprise Edition's built-in dashboards, you can **compare test runs side by side**, track key metrics over weeks or months, and spot performance degradation before it turns into production incidents. This historical view gives engineering and QA teams the data they need to make informed, proactive decisions instead of reacting after failures occur.



Define SLOs and enforce them automatically

In financial services, SLAs are contractual. Gatling Enterprise Edition lets you define Service Level Objectives directly on your load tests: set a target percentile (P50, P95, P99, up to P99.9999) and a response time threshold in milliseconds, or cap your error ratio at a maximum percentage.

For finance teams, this turns every test run into a verifiable compliance check: did the payment flow meet its P99 target for the required duration? Did error ratio stay below the threshold that would trigger a regulatory flag? The answer is in the report, not in someone's interpretation of a chart.



Choosing the right load testing platform for financial services

Embedding continuous performance testing into financial services delivery requires more than the right intent; on the contrary, it requires a platform built for the constraints and expectations of regulated, private-infrastructure environments.

Gatling Enterprise Edition enables financial companies to:

Analyze smarter and act faster

Real-time dashboards, trend comparisons across releases, and SLO compliance scores that give leadership the visibility they need, without requiring them to interpret raw engineering data.

Create tests your way

Build tests via code, low-code, or no-code. Script in JS/TS or Java, import existing collections, or design visually. Even people who've never written a load test can use Gatling

Deploy load tests from your coding agent

Gatling's MCP Server & Skills lets engineering teams trigger and manage load tests directly from their AI coding environment, embedding performance validation into the workflow.

Collaborate and share results easily

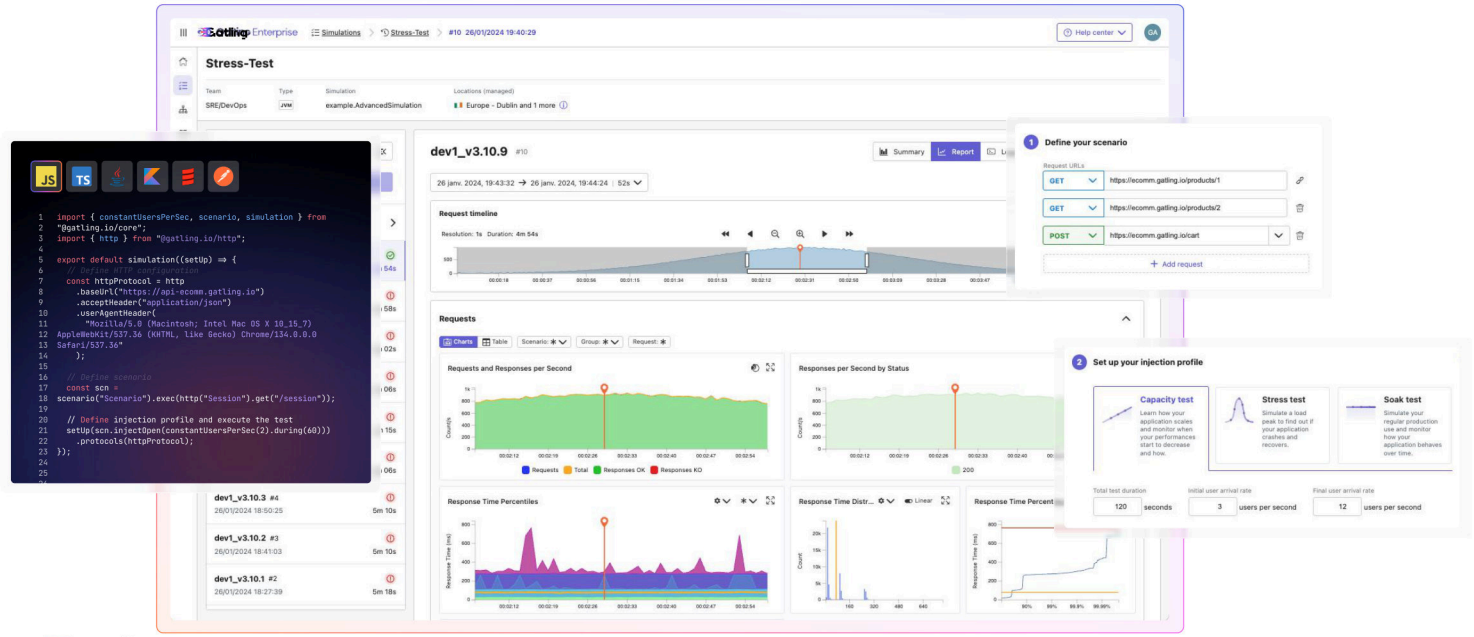
RBAC, SSO, quotas, integrations with Slack, Teams, and Jira mean performance signals surface where teams already work and shared reports ensure results reach the right people.

Deploy load generators anywhere

Financial services infrastructure requires testing that reflects real production conditions, not a public proxy. Run tests from inside your network or from Gatling managed regions.

Track performance compliance on every run

Define SLOs against response time and error rate thresholds. Gatling evaluates them continuously across each test run and produces a compliance score: objective, repeatable, and auditable.



Gatling

Gatling Enterprise is the platform purpose-built to deliver Continuous Performance Intelligence: transforming load testing from a technical activity into an organizational capability that business leaders can govern, product teams can engage with, and engineering teams can scale.

With its powerful open-source and enterprise platforms, Gatling empowers teams to test APIs, microservices, and web apps in real-world conditions.

Trusted by thousands of companies worldwide, Gatling is the performance backbone for development, QA, and DevOps teams building the next generation of software.

Whether you're scaling APIs, migrating to the cloud, or handling flash traffic spikes, Gatling helps you deliver fast, reliable performance.

Ready to impment continous performance intelligence?

See how AI-assisted performance testing can take your tests to the next level

Talk to an expert ›