Gatling

2025

EBOOK

Scaling load testing across the enterprise

How to make performance testing everyone's job, and scale with confidence using Gatling

Performance testing used to be a one-off activity — something a specialized team did before a major release, often with brittle tooling and limited visibility. That model doesn't hold up anymore.

Today, systems are distributed, releases happen daily, and user expectations are unforgiving.
A performance regression in one service can ripple through your platform. A missed slowdown can cost conversions, support tickets, or brand trust.

That's why scaling load testing across the organization is no longer a nice-to-have. You need a strategy where every team contributes to performance, and where infrastructure, tooling, and culture support testing at scale.

The good news? It's absolutely possible — with the right workflows, governance, and tooling.

> This guide shows you how to get there, with a focus on test-as-code, automation, collaboration, and platform-level tooling. And it shows you how Gatling Enterprise Edition helps you implement that vision without adding complexity.

Common blockers for organizations that try to scale...

- **Siloed and late testing:** performance tests only happen right before a big release
- **Local limits:** with open-source solutions, you quickly hit CPU limits on a single machine.
- **CI/CD friction:** Your functional tests are automated, but performance tests aren't.
- **Inconsistent tools and practices:** Different teams use different tools, performance remains a black box.
- **No visibility or governance:** Stakeholders don't know if a system is ready for scale.

...And how to avoid them

- **Shift left:** Bring performance testing earlier in the dev cycle
- **Test as code:** Treat performance tests like real software — versionable, reviewable, reusable
- **Automate everything:** CI/CD integration isn't optional
- **Build a culture of ownership:** Developers, QA, and SREs must all play a role
- **Eliminate infra bottlenecks:** Don't let test setup or load generators block anyone
- **Standardize, but stay flexible:** Use shared platforms and practices, not one-size-fits-all mandates

Gatling Enterprise Edition supports all of this by design. It removes the infrastructure burden, integrates cleanly into developer workflows, and gives teams the visibility and control they need to test with confidence.

Let's break them down.

Tips to optimize your load tests with Gatling Enterprise Edition

Scale with test as code

Load tests are too important to live in fragile UIs or out-of-date XML configs. The best way to scale testing is to treat it like code.

With Gatling, your test scenarios are written in code (JavaScript, Java, Scala, or Kotlin), stored in Git, and versioned like everything else. That means:

- Developers can reuse libraries, share setup logic, and run tests locally
- Teams can code review load tests like any other change
- Changes to load profiles, SLAs, or endpoints are tracked automatically
- Test scripts can be templated and reused across projects

This ensures repeatability, transparency, and version control across teams.

With Gatling Enterprise Edition, you keep this test-as-code model — but gain team-level control, CI integration, and distributed execution at scale.

Use a shared testing platform

Large organizations succeed when they centralize infrastructure but decentralize usage.

Use a common execution platform, such as Gatling Enterprise Edition, that provides:

- Distributed and scalable load generators
- Quotas, RBAC, and audit logs
- Shared result storage and reporting

Let every team run tests independently, within governance boundaries.

This approach avoids tool sprawl while enabling self-service and consistent standards across the organization.

Integrate CI/CD to your pipeline

If performance tests aren't automated, they'll be skipped. Scaling means every team should be able to run lightweight tests on every build or merge.

Gatling Enterprise Edition supports this natively. You can:

- Trigger tests via CLI, GitHub Actions, GitLab, Jenkins, or REST API
- Define thresholds (e.g., "95th percentile response time < 2s")
- Stop builds or alert the team if thresholds are breached
- Schedule large tests off-hours or after merges
- Export results or compare them across builds and releases

With these integrations in place, teams build performance validation into their workflow, not as a phase, but as a habit.

Establish common metrics and reporting standards

Different teams can use different tools — but they must speak the same **performance language**.

Standardize your metrics and KPIs:

- Use percentiles (P95, P99) instead of averages
- Track error rate, throughput, and availability consistently
- Define clear SLAs and SLOs per service
- Use standard naming conventions and tagging across reports

Centralize dashboards and reporting for visibility.

Gatling Enterprise Edition offers customizable reports and run comparisons, making cross-team performance data **comparable, traceable, and actionable**.

Shift left to test early and often

The earlier you test, the cheaper it is to fix.

Make performance testing part of development by:

- Running lightweight load tests on pull requests or nightly builds
- Teaching developers to write and run small scenarios locally
- Including performance checks in sprint demos or “definition of done”

This embeds performance thinking into daily development and helps teams catch regressions before they reach staging or production.

Promote collaboration and shared ownership

Performance testing is cross-functional by nature:

- Developers handle scenario creation and early feedback
- QA/SDETs maintain coverage and assertions
- SREs manage infrastructure and monitoring
- Product and business teams consume insights

Everyone contributes; nobody is the bottleneck.

Gatling Enterprise Edition makes this easy with project spaces, usage quotas, and team permissions, enabling collaboration without conflict.

Distribute load your load across locations

Scaling from 1,000 to 100,000 users shouldn’t mean more infrastructure headaches.

Gatling Enterprise Edition removes the pain with:

- Distributed load generation across managed or private locations
- One-click test launches and automatic resource scaling
- Built-in queuing, quotas, and shutdown rules

You focus on writing and analyzing tests — not configuring virtual machines or managing regions.

Integrate with observability and APM

Performance testing shouldn’t exist in a vacuum.

Integrate your test results with your observability stack to understand end-to-end behavior.

- Correlate Gatling metrics with logs, traces, and APM data
- Detect root causes (CPU spikes, database latency, network timeouts) faster
- Feed insights into your incident postmortems and dashboards

This tight integration helps SRE and DevOps teams connect load results with production realities, turning performance testing into proactive reliability engineering.

On Gatling Enterprise Edition, you can add your load testing metrics to your observability dashboards, thanks to direct integration with Datadog and Dynatrace.

Ensure production-like environments

If your test environment doesn't reflect production, your results don't matter.

To scale performance testing across teams, you need:

- Environments that mirror production topology, including autoscaling and load balancers
- Datasets that match production volume and shape, via masked or synthetic data
- Network variability, to simulate real regional latency
- Scenarios based on real user behavior, not static loops

Gatling Enterprise Edition supports hybrid deployments — you can run load generators inside your VPC or private cloud while still using the hosted UI.

That gives you realistic conditions, security, and flexibility.

Build a culture of continuous improvement

Scaling load testing isn't a one-time project — it's an ongoing practice.

- Document lessons learned from each test
- Compare results over time and track regressions
- Celebrate improvements publicly to reinforce the culture

With Gatling Enterprise Edition, you can easily compare runs, measure trends, and share wins across teams.

Performance becomes a shared success, not just a technical metric.

Governance and consistency

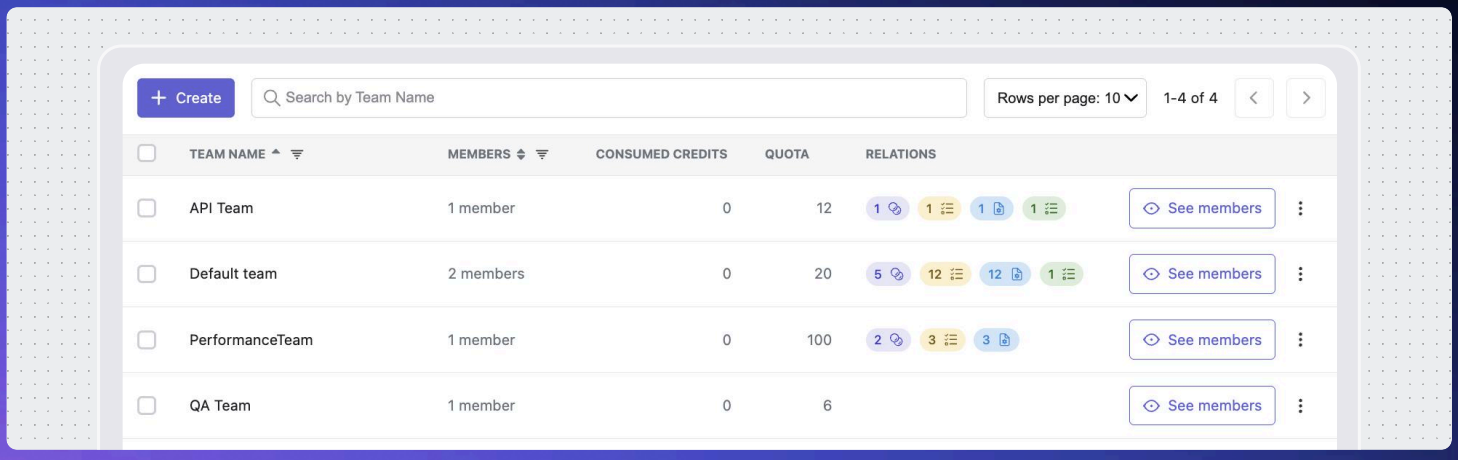
Autonomy doesn't mean chaos.

Set lightweight governance that balances freedom and control:

- Define clear roles, permissions, and quotas
- Standardize test naming conventions and tagging
- Run periodic performance reviews per release or sprint

Gatling Enterprise Edition simplifies this with built-in RBAC, audit logs, and team-level resource limits.

Consistency keeps testing safe and scalable without slowing teams down.



On Gatling Enterprise Edition, you can directly assign roles, create team, and allocate credits in the interface.

Do's and don'ts of load testing

Adding load tests to your CI/CD pipeline is one of the best ways to prevent performance regressions from slipping into production, but timing is everything and not all tests should be run in CI.

Do's

Run smoke tests on every build

Short, lightweight tests that validate basic performance (e.g., response times under small load). These catch regressions early without slowing the pipeline.

Schedule full-scale load tests regularly

Daily or nightly runs can simulate realistic traffic and catch issues before release, without blocking developer flow.

Test critical user journeys

Focus on checkout, login, API endpoints, or anything that's business-critical. Not everything needs to be load-tested every build.

Automate pass/fail criteria

Define thresholds for response time, error rates, and percentiles (e.g., 95th/99th). Fail builds when performance drops.

Don'ts

Don't run heavy tests on every PR

You'll slow down the pipeline and annoy developers.

Don't skip trend analysis

A single failed run isn't enough; look at patterns over time to catch gradual regressions.

Don't forget environment parity

CI tests should run on an environment that's close to production; otherwise results can mislead.

Choosing the right platform for modern load testing

When performance matters, your load testing platform becomes mission-critical. The complexity of today's distributed systems, global user bases, and rapid release cycles requires more than just generating virtual users, it demands precision, scale, and deep integration with engineering workflows.

Gatling Enterprise Edition is designed to meet these demands. It provides a complete, production-grade platform built to help teams analyze faster, collaborate better, automate with confidence, and test at any scale. Its capabilities are structured around five core pillars that support the entire performance engineering lifecycle:

Analyze smarter and act faster

Gain real-time visibility with dashboards, trend comparisons, and actionable insights.

Create tests your way

Build tests via code, low-code, or no-code, import Postman, script in JS/TS or Java, or design visually.

Unlock automation

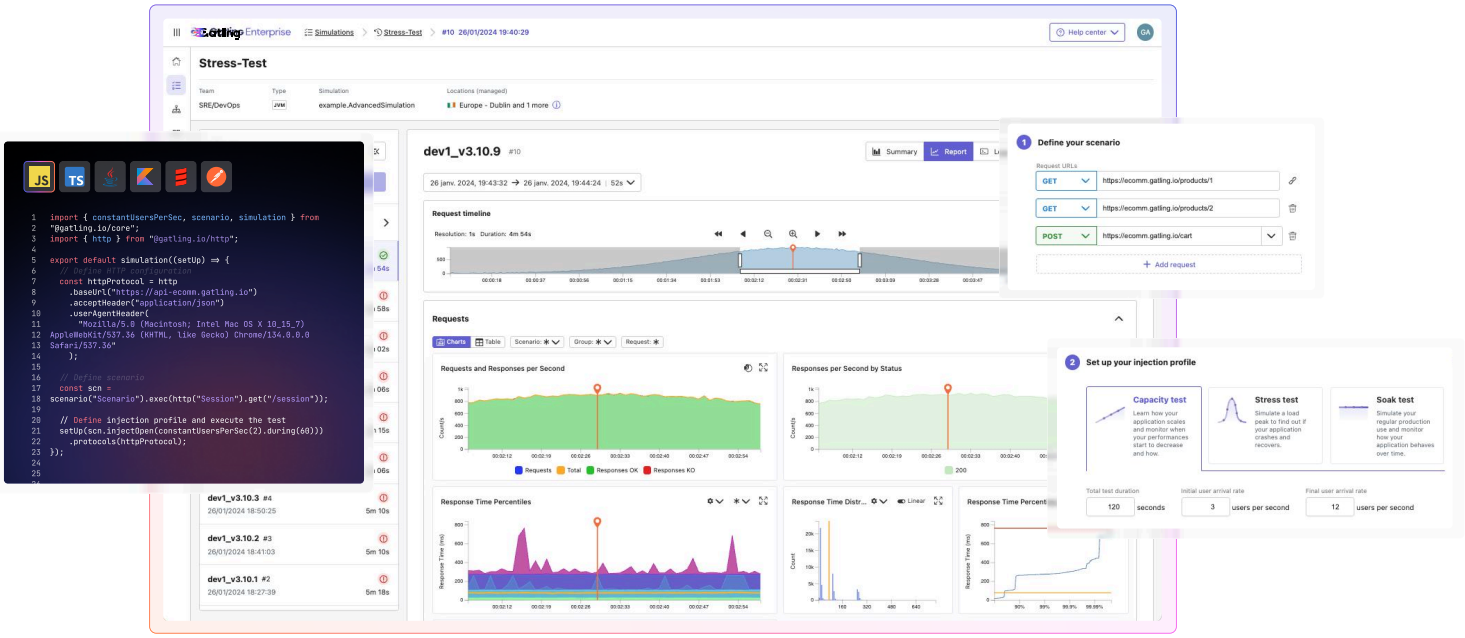
Trigger simulations via CI/CD or API, apply stop criteria, and gate releases with performance thresholds.

Collaborate and share results easily

Use RBAC, SSO, quotas, and shared reports. Share results via Slack, Teams, or Jira.

Deploy load generators anywhere

Run tests from Gatling managed regions, your cloud, or on-prem.



Gatling

Gatling is the leading solution for modern load testing, enabling developers and organizations to deliver fast, reliable applications at 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 evaluate Enterprise Edition?

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

Talk to an expert ›