Gatling

2025

WHITEPAPER

Modern performance testing workflow: From versioned scripts to continuous feedback

Automate test environments, enforce performance gates, and deliver reliable applications at scale with Gatling Enterprise Edition.

MODERN PERFORMANCE TESTING WORKFLOW

Performance testing has evolved.

It's no longer a last-minute check before release: it's a continuous signal embedded in the way modern software teams build, ship, and learn.

In the past, performance tests happened at the end of a cycle: a single, large-scale run that produced a static report.

By the time issues appeared, it was too late to fix them without slowing delivery. This made performance testing reactive, expensive, and disconnected from everyday development.

Today, the pace of software delivery doesn't allow for that.

Applications change daily, infrastructure scales dynamically, and user expectations keep rising.

To keep up, teams have redefined how they test for performance: not as a milestone, but as an ongoing discipline.

Modern engineering organizations now treat performance as a workflow, not an event. It's built into the same system that governs everything else—code reviews, pipelines, monitoring, and incident response.

Instead of one-off tests, teams rely on versioned scripts, automated executions, and shared insights that create a living feedback loop.

Versioning load tests → Automating them → Sharing results → Building a feedback loop.

Each step reinforces the next, turning performance testing from a single team's task into a shared engineering practice.

> This whitepaper explores how to modernize your performance testing workflow so you go from static reports to continuous confidence in every release.

Version your load tests like code

Performance testing starts with visibility and ownership.

If your load tests live in a folder that no one opens—or worse, in a tool only one person knows how to use—they're already outdated.

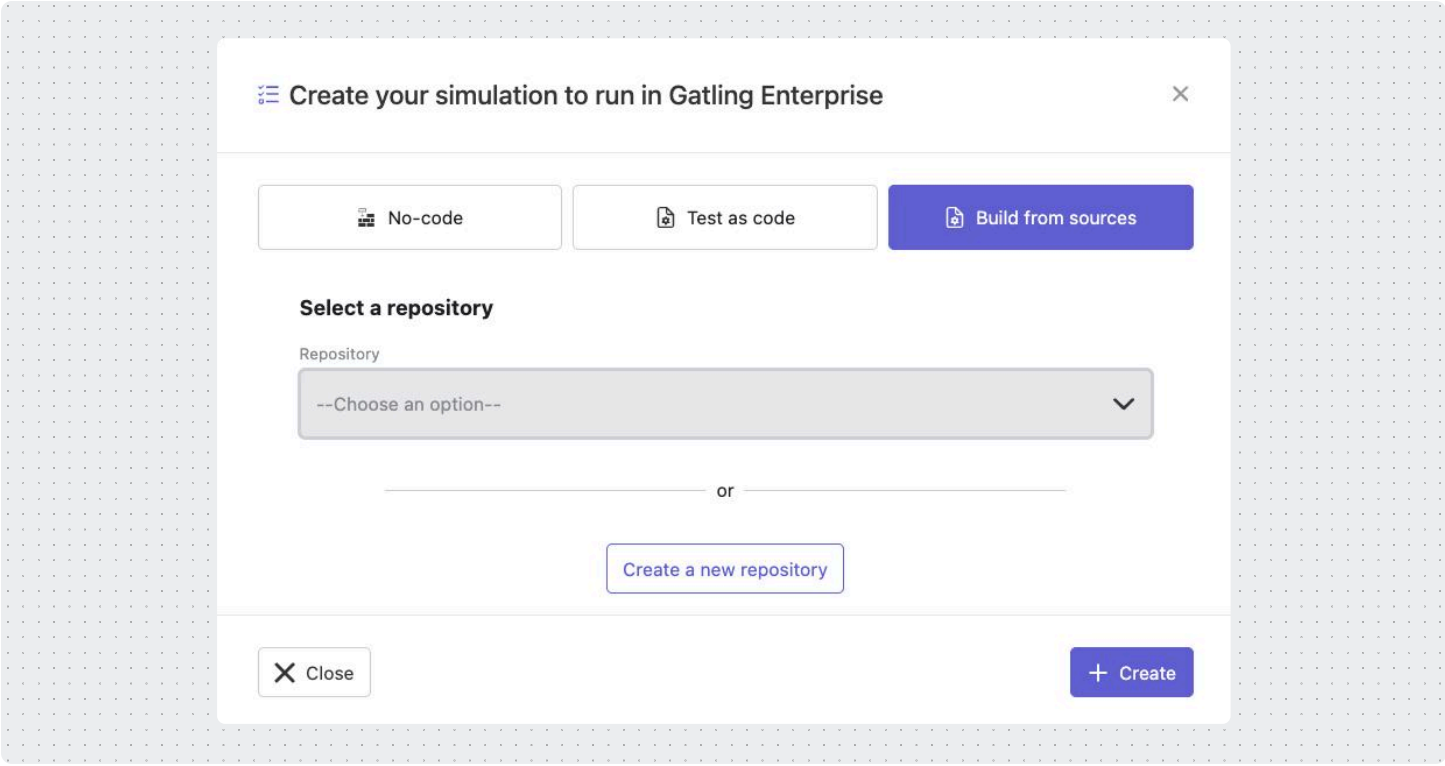
Treating tests as code changes that dynamic completely. When tests are versioned alongside your application, they evolve with it.

Every new feature, dependency, or infrastructure change has a corresponding update in test coverage. That means no surprises at release time and no forgotten scenarios that only fail in production.

Versioning also creates a shared source of truth. Developers can review test logic in pull requests, suggest improvements, and trace when and why performance conditions changed.

QA and SRE teams gain transparency into how load profiles are defined and can reproduce tests easily in different environments. What used to be a static artifact becomes a living component of your system.

> **Pro tip:** With Gatling Enterprise Edition, your tests aren't just stored in Git: they're built from it. Each simulation and configuration is fetched directly from your repository, executed automatically, and tied to a specific commit for full traceability between source code and performance results.



With build from sources you can create your own simulaton by connecting your own repository

Automate tests in CI/CD or through APIs

Once tests are versioned, automation connects them to your delivery cycle.

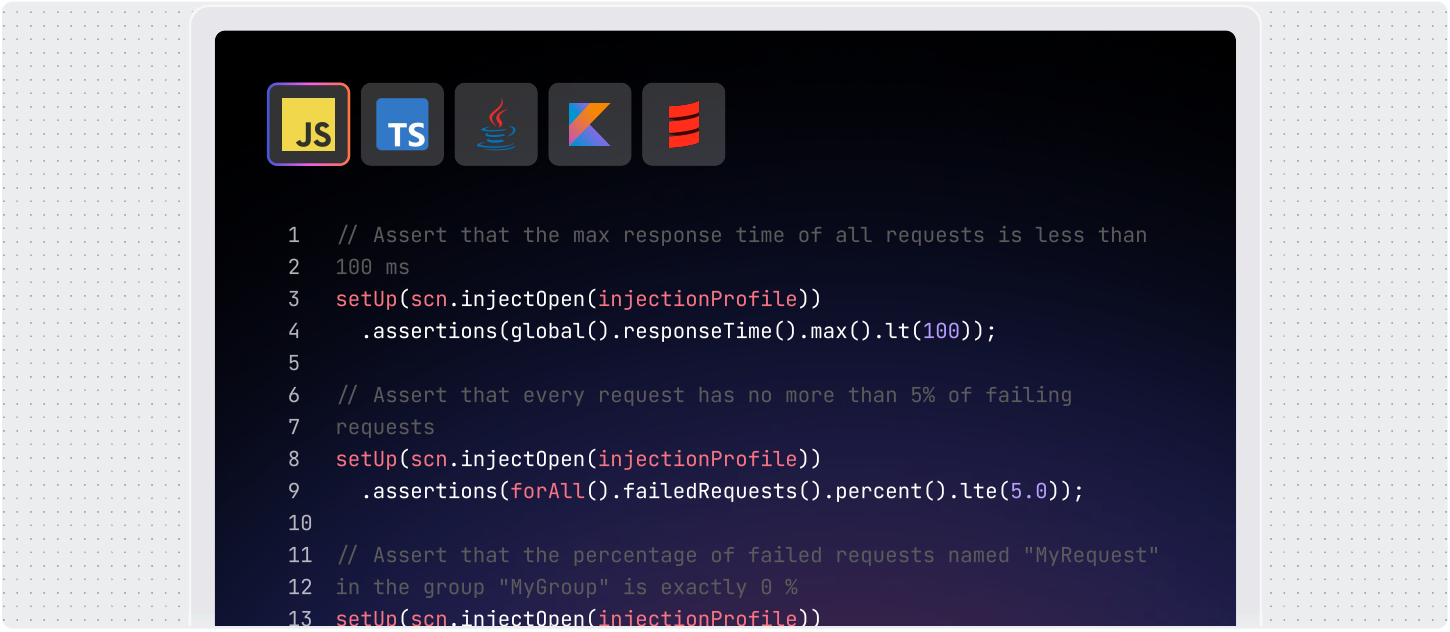
Performance shouldn't depend on manual runs—it should trigger automatically, just like unit or integration tests.

Integrating load tests into CI/CD pipelines allows you to validate every change under realistic conditions, long before production.

This continuous approach turns performance validation into part of your quality gate: Builds fail when latency spikes, throughput drops, or error rates rise. Automation transforms testing from a scheduled task into a safety net that works around the clock.

Pro tips: With Gatling Enterprise Edition, automation isn't limited to triggering tests; it's about making performance testing a governed, intelligent part of your CI/CD pipeline.

- **CI/CD integrations and APIs:** Automate your performance tests programmatically through native integrations with GitHub Actions, GitLab CI/CD, Jenkins, or Azure DevOp, or use Gatling's APIs to embed load testing directly into your internal platforms and delivery pipelines.
- **Assertions as quality gates:** Define pass/fail performance thresholds (for response times, error rates, throughput, etc.) so that any build exceeding your SLAs automatically fails, preventing regressions before they reach production.
- **Run Stop Criteria for smarter testing:** Set conditions that automatically stop a test when it no longer provides valid data — for example, when error rates spike or when performance thresholds are consistently breached, to avoid wasted compute time and skewed results.



```
// Assert that the max response time of all requests is less than
100 ms
setUp(scn.injectOpen(injectionProfile))
  .assertions(global().responseTime().max().lt(100));

// Assert that every request has no more than 5% of failing
requests
setUp(scn.injectOpen(injectionProfile))
  .assertions(forAll().failedRequests().percent().lte(5.0));

// Assert that the percentage of failed requests named "MyRequest"
in the group "MyGroup" is exactly 0 %
setUp(scn.injectOpen(injectionProfile))
```

Assertions are used to verify that global statistics, like response time or number of failed requests, match expectations for a whole simulation.

Share and correlate results across teams

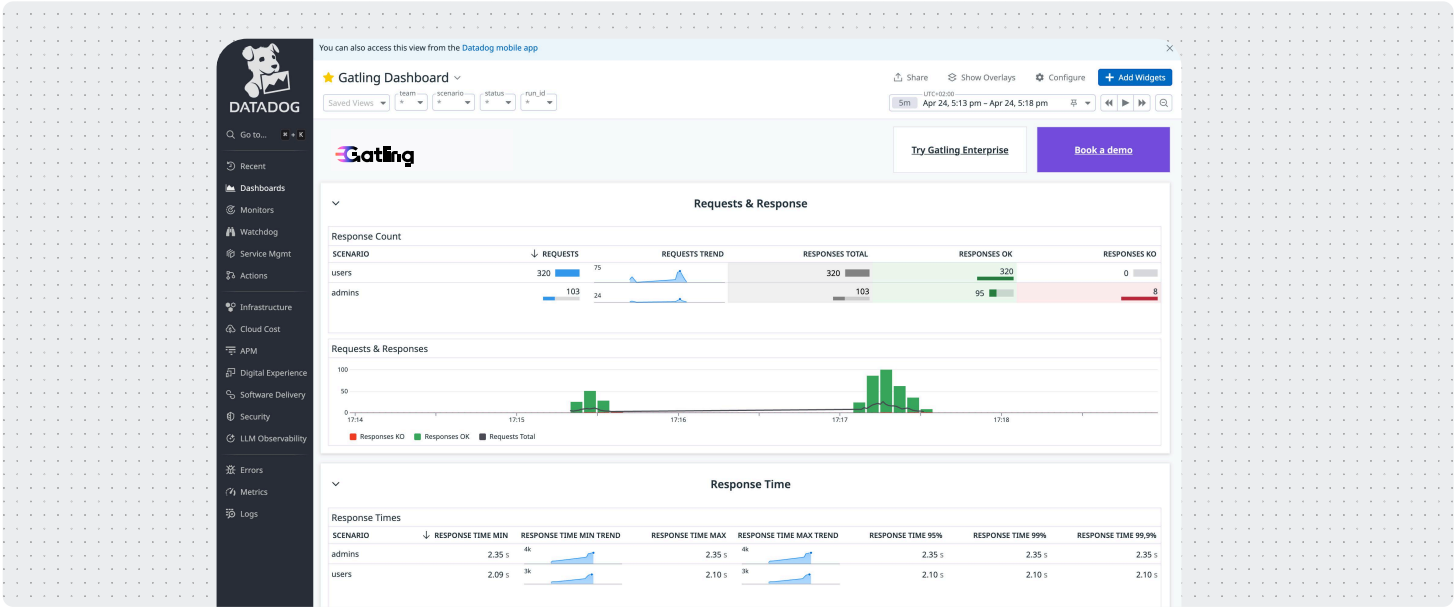
The real value of performance testing isn't just in collecting numbers; it's in sharing insights.

When results stay trapped in reports or dashboards no one opens, teams miss the opportunity to learn from them.

Publishing results and correlating them with metrics from observability tools creates a shared language across Dev, QA, and SRE teams.

This transparency turns performance data into a decision-making tool: Developers see how their code behaves under load, operations see how infrastructure responds, and leaders see the long-term impact on reliability.

> **Pro tip:** Use Gatling's native integrations with Datadog and Dynatrace to bring load-testing insights into the same observability dashboards your SRE and Ops teams already rely on. Test metrics like latency, throughput, and error rates appear alongside infrastructure telemetry, making it effortless to correlate performance trends with CPU usage, memory pressure, or network saturation.
>
> Then, close the loop with webhooks and notification integrations: automatically push test summaries or alerts to Microsoft Teams, Slack, or your internal systems, so developers and stakeholders get real-time visibility without leaving their workspace.
>
> With these integrations, Gatling Enterprise Edition becomes a central node in your performance ecosystem—connecting test data, observability, and communication in one continuous feedback flow.



The Datadog integration allows Gatling Enterprise Edition to send load-test metrics such as response times, throughput, and error rates directly into Datadog's observability platform.

Build a continuous feedback loop

Modern performance testing isn't a one-time validation step—it's an ongoing signal in your delivery pipeline. By turning performance data into a continuous feedback loop, teams can connect how code changes impact real-world behavior over time.

Every test contributes to a growing performance history, helping you understand not just whether your system scales, but how it evolves.

Instead of isolated snapshots, modern teams rely on trend tracking and regression detection to surface subtle drifts before they become issues.

Automated alerts highlight anomalies early, while data from previous runs guides capacity planning and architectural improvements.

This approach shifts performance testing from reactive firefighting to proactive engineering—where teams continuously learn, adjust, and optimize as part of the release cycle.

> **Pro tip:** Gatling Enterprise Edition's Run Trends and Multi-Run Comparison features let you overlay new test results against your historical baselines — instantly revealing drifts, regressions, or performance improvements over time.
>
> These insights make it easier to monitor long-term trends, validate optimizations, and quantify progress across versions or releases. Combined with observability integrations, they give you a unified timeline of performance health — from commit to production.
>
> You can even take it one step further: use webhooks to automatically create Jira tickets or optimization tasks when thresholds are breached, turning test results directly into actionable engineering work.

Your recent runs are displayed in 3 bar graphs, with up to 10 runs in total.
Only successfully executed runs (noted by green check marks in the left-side menu) are included in the graphs.

Modern load testing as a discipline

Making performance testing feel like code

For developers, modern load testing should feel as natural as pushing a commit or running a build.
Tests evolve through pull requests, execute automatically, and surface insights right where teams work.

When performance testing follows the same principles as software engineering—version control, automation, and collaboration—it scales with your organization instead of slowing it down.

For teams, it becomes the invisible safety net that keeps releases smooth and users happy.

That's the future of performance testing—built in, not bolted on.

> **Pro tip:** Gatling Enterprise Edition turns "performance-as-code" from an idea into a practice.
> Tests live in Git, automation runs them in CI, and analytics connect them to production telemetry.
>
> Everything—from test creation to reporting—follows the same engineering workflows your teams already use, so performance becomes a built-in part of delivery, not a separate phase.

Test as Code extends the same engineering principles that drive modern development:
Repeatability, traceability, and collaboration.

Choosing the right platform for modern load testing

Embedding load testing into the GitOps lifecycle transforms performance validation from a manual activity into an automated, measurable discipline. With Gatling Enterprise Edition, simulations, configurations, and infrastructure definitions are all expressed as code, versioned, and governed through the same workflows that manage application delivery.

Results are observable and auditable, providing end-to-end traceability from commit to performance impact. The outcome is a delivery pipeline where reliability grows in parallel with speed. Each release is deployed with confidence, because performance verification is no longer an afterthought but a built-in step of your continuous delivery process.

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

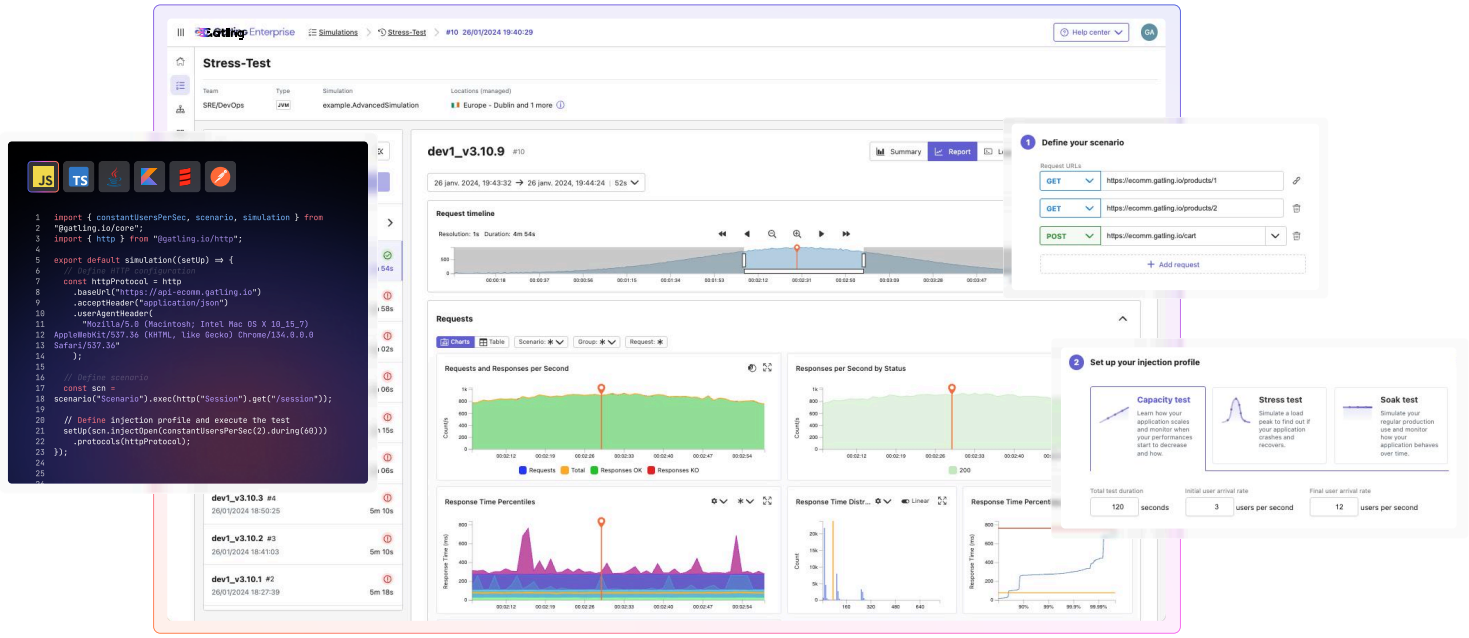
Trigger simulations via CI/CD or API, apply stop criteria, and gate releases with performance thresholds.

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 ›