Gatling

2025

DATASHEET

Shift-left performance testing with Gatling

A practical guide for engineering leaders building faster, more resilient systems.

What is shift-left in performance testing?

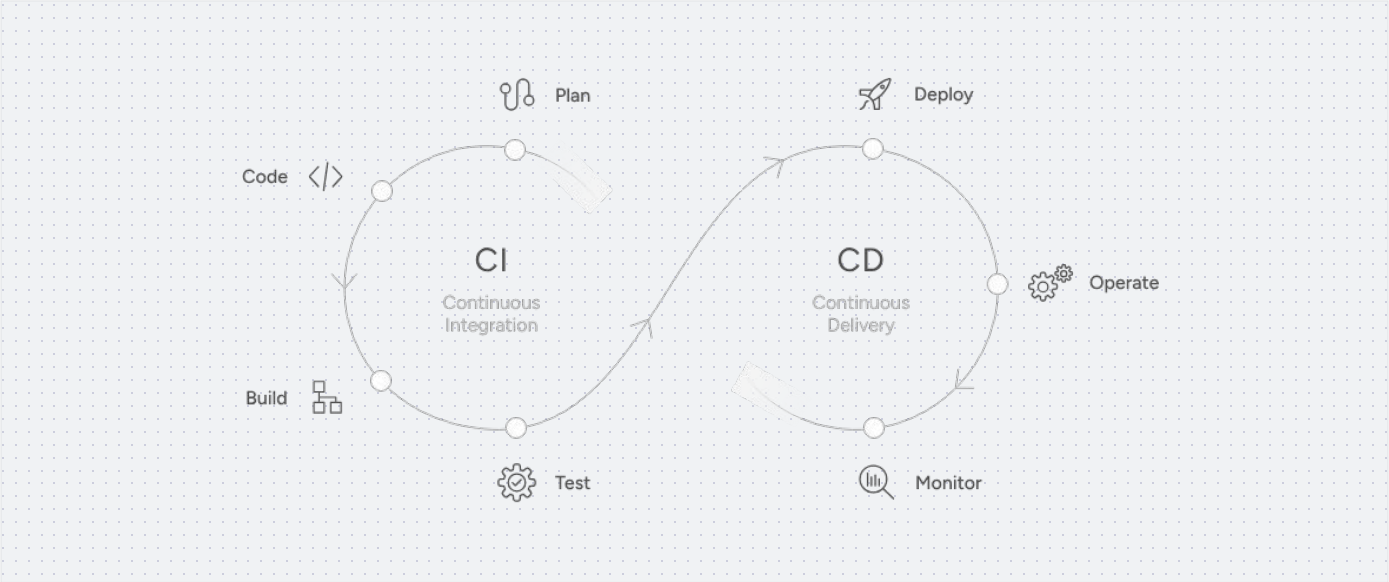
Modern software systems evolve fast, and break even faster.

Performance issues that used to appear only during peak traffic now show up in everyday interactions across distributed microservices, AI-powered workloads, and globally scaled architectures.

Shift-left performance testing means moving performance and load validation earlier in the software delivery lifecycle:

- **from late-stage QA to development and integration,**
- **from "big-bang" pre-release campaigns to continuous, automated checks,**
- **from a specialized performance team to shared ownership across engineering.**

> Instead of treating performance testing as a final gate, shift-left makes it a built-in development practice: developers, QA, and SRE continuously validate response times, error rates, and scalability as the system evolves.



Core principles of shift-left performance testing

Early and continuous testing
Integrate lightweight Gatling tests in CI/CD to hvalidate critical functions on every build.

Realistic load and data
Model true user behavior and peak usage to make early test results meaningful.

Service-level performance testing
In microservice architectures, you must test components independently before integrating them together.

Clear SLAs and thresholds
Teams need unambiguous thresholds like p95 < 350 ms; error rate < 0.1%; CPU stays < 80%; CI/CD enforces these automatically.

Shared visibility and collaboration
Performance is not a department. It's a culture. Shift-left works only when developers, QA, and SRE see the same performance signals.

What happens when teams test, but don't shift left?

Even teams that "do performance testing" face recurring problems when tests occur too late:

Late discovery of bottlenecks

Issues emerge when architecture is fixed, leaving little room to correct design flaws.

Common issues:

- Performance fixes require redesign or deep refactoring, not simple code changes
- Incidents only surface under full system integration, making root-cause analysis slow
- Test data, dependency mocks, or environments no longer reflect actual traffic or usage patterns
- Performance regressions accumulate unnoticed across multiple services

Reliance on specialized teams

Performance testing often sits with a small expert group, creating bottlenecks and fragmented visibility.

Common issues:

- Developers wait days or weeks for results from a performance team
- Scenarios become outdated because experts can't keep up with product pace
- Knowledge silos lead to inconsistent testing patterns across services
- Manual coordination required between dev, QA, SRE, and perf engineering

Limited coverage

Only a fraction of services or endpoints receive meaningful performance validation.

Common issues:

- Teams test only "flagship" flows; long-tail APIs go unvalidated
- Metrics pipelines drop samples or smooth peaks when throughput increases
- Mocked dependencies behave too differently from real services
- High operational overhead prevents expanding coverage to all microservices

High cost of defects

A slow endpoint found in development takes hours to fix; found in production, it triggers incidents, rollbacks, and customer impact.

Common issues:

- Limited scalability caused by thread-per-user architectures
- CPU exhaustion before real concurrency is reached
- Unpredictable throughput under bursts or TLS-heavy traffic
- Inability to reach peak-load scenarios without load generator failures

Infrequent, high-stakes test cycles

"Big bang" tests before release are hard to coordinate and rarely reflect the latest changes.

Common issues:

- Difficulty simulating real concurrency due to blocking or thread-per-user models
- Distributed setups that break under CPU pressure, network constraints, or config drift
- Load generators require constant tuning and manual scaling
- Tests run only before releases, leaving long intervals where regressions accumulate

> Shift-left solves these issues by embedding performance into everyday development and automation workflows, ensuring teams identify, validate, and resolve bottlenecks long before they reach production.

Why Gatling enables shift-left?

Shift-left requires tools built for developers, automation, and modern architectures. Gatling is designed around these needs.

Developer-First Test-as-Code

Developers write and maintain performance tests directly in code, stored in Git, reviewed in pull requests, and easily refactored as services evolve.

CI/CD-native execution

Gatling integrates seamlessly with GitHub Actions, GitLab CI, Jenkins, Azure DevOps, and more.
Performance thresholds can break the build when SLAs are not met—just like unit tests.

Configuration-as-Code

Environments, load profiles, regions, and SLAs are all declared as code for reproducibility, auditability, and reliable automation.

From local tests to global scale

Run lightweight tests locally for rapid feedback, automatically run tests on feature branches through CI/CD, and scale to distributed, multi-region load testing using Gatling Enterprise Edition, without rewriting scenarios.

Aligned with modern architecture complexity

Gatling supports real-world protocols and patterns: HTTP, APIs, microservices, SSE, WebSockets, gRPC, and more.
It suits hybrid, cloud-native, and containerized environments and adapts to any security or network context.

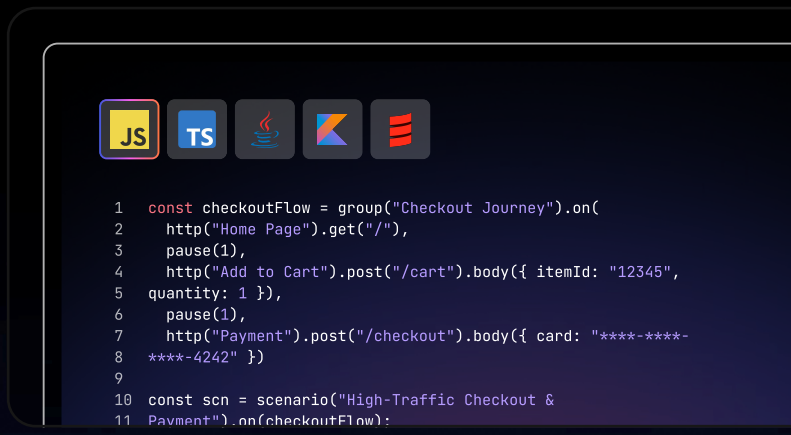
> Gatling brings performance into the **same workflows developers use every day** —making shift-left not just possible, but practical.

Key capabilities for shift-left

Test-as-Code for developer ownership

Gatling scenarios are written directly in code—JavaScript, TypeScript, Scala, or Java—and stored in Git like any other part of the application.

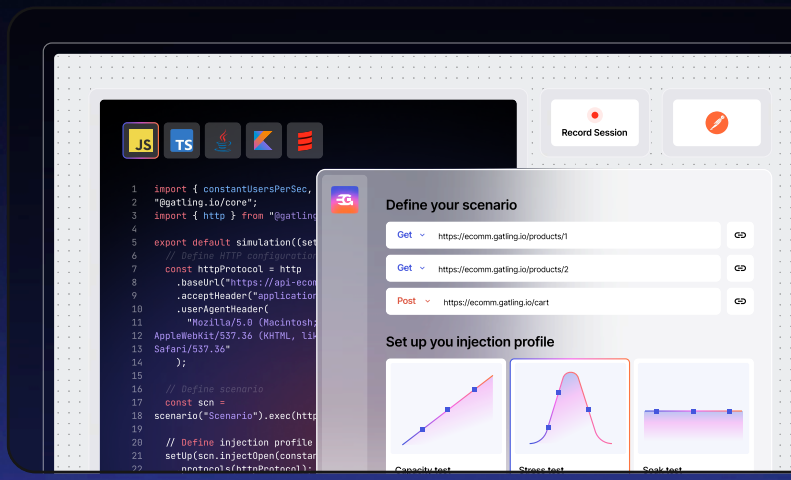
This model lets developers review, refactor, and evolve performance tests alongside the services they validate; the same way as application code, teams gain repeatability, traceability, and shared ownership from the very first commit.



Flexible test design options

Gatling provides several developer-friendly ways to create performance and API tests early in the lifecycle, so teams can validate scalability as soon as an endpoint or user flow exists:

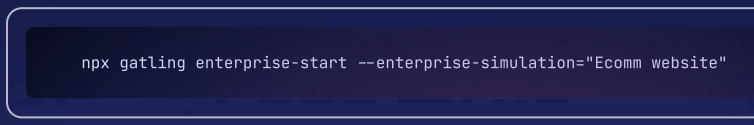
- **Test-as-Code in JS, TS, Java, or Scala**
- **Postman → Gatling import** bootstraps API performance tests from existing collections.
- **No-code test builder** with one-click export-as-code for further editing in the IDE.
- **Gatling Studio** records browser journeys into clean, maintainable Gatling scripts.



Fast execution from IDE

Developers can run Gatling tests directly from their IDE for rapid iteration, or trigger them from the command line to execute locally or on Gatling Enterprise Edition.

This provides fast, actionable feedback during development and makes performance testing a natural part of daily coding routines.



CI/CD-Native Automation and Performance Gates

Gatling integrates seamlessly with GitHub Actions, GitLab CI, Jenkins, Azure DevOps, and more:

- Automatic execution on commits, merges, and nightly builds
- Performance thresholds and assertions that fail the build when SLAs are not met

This turns performance testing into a continuous, automated aspect of your delivery pipeline.

Gatling

Key capabilities for shift-left

Service-level performance testing for microservices

Gatling is perfect for API-level testing, making it ideal for validating microservices early:

- Test isolated services directly during development
- Catch latency spikes, timeout risks, or dependency issues before integration
- Prevent bottlenecks from propagating through distributed architectures

Teams validate each service's SLAs upstream, not during late-stage system testing.

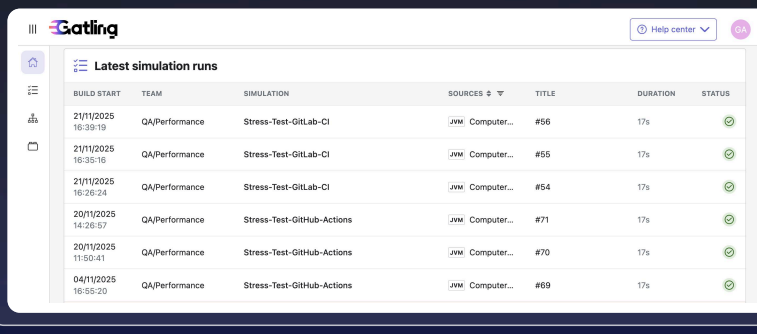
```
import { simulation, scenario, atOnceUsers }
from "@gatling.io/core";

import { http, ws } from "@gatling.io/http";

import { mqtt } from "@gatling.io/mqtt";

import { grpc } from "@gatling.io/grpc";
```

Reusable scenarios across local → CI/CD → Enterprise

A single Gatling simulation works everywhere:

- On developers' machines,
- Inside CI pipelines,
- And at scale in Gatling Enterprise Edition.

No rewrites or alternate formats — one test evolves with the codebase and scales with team maturity.



Real-time feedback and observability-ready results

With Gatling Enterprise Edition, developers get instant feedback on what they are building, validating performance as soon as new code is written:

- **Real-time dashboards** that show how new code behaves under load
- **Trend and comparison views** that highlight regressions across builds and releases
- **Correlated insights through integrations with Datadog, Dynatrace** and other observability platforms

Benefits of shift-left testing

Technical benefits

Earlier detection of performance regressions

Running tests during development surfaces latency issues and bottlenecks before they spread across services. Teams fix problems when context is fresh and changes are still cheap.

Continuous, automated validation in CI/CD

Performance tests run automatically on each build, ensuring every commit meets defined SLAs. This prevents late-stage surprises and keeps delivery predictable.

Higher coverage across services and APIs

Teams can test microservices, APIs, and full user flows from the moment endpoints exist. This expands performance visibility across the entire architecture.

Improved customer experience through reliable performance

Stable response times and resilient services lead to smoother user journeys. Better performance directly supports engagement and retention.

Business benefits

Reduced release risk and fewer production incidents

Early validation catches problems long before they impact users. Releases become safer, smoother, and more predictable.

Lower cost of fixing performance issues

Issues resolved in development cost a fraction of post-release fixes. Teams avoid expensive firefighting during or after deployment.

Faster time-to-market with predictable delivery cycles

Automated performance gates keep teams moving without sacrificing quality. This reduces delays and accelerates feature delivery.

Optimized infrastructure and cloud spend

By identifying inefficient endpoints early, teams avoid over-provisioning and reduce waste. Performance testing becomes a lever for cost efficiency.

They shifted-left with Gatling

TRANSPORTATION

How an airline implemented shift-left testing with Gatling Enterprise Edition

5M
DAILY VISITS

601ms
TIME TO FIRST BYTE

By integrating performance testing early in the SDLC, the airline's team identified and addressed performance issues

With the airline's continuous implementation of a comprehensive approach with Gatling Enterprise, it became the fastest-loading website among airlines.

Currently, the airline's home page peaks with over 5 million daily visits.

> Gatling's simplicity, scalability, seamless integration, extensibility, value out-of-the-box along with its code-driven approach, low resource footprint, and ability to handle high loads, made it the ideal choice for us.

Principal Architect
Software Performance Engineer

SOFTWARE

How Sophos scaled backend API performance testing across teams with Gatling Enterprise Edition

75-80%
BACKEND APIS COVERED

100+
SIMULATIONS

Sophos adopted Gatling to democratize performance testing across decentralized teams

To meet these demands, Sophos adopted Gatling Enterprise as their performance testing platform. Load testing became a proactive part of development, helping teams detect resource-heavy behavior before it reached production.

Tests were designed not only to verify SLAs but also to optimize cloud resource consumption, ensuring services scaled effectively without over-provisioning.

> One reason we chose Gatling was how easy it is to re-run tests. I also really appreciate having the results directly available in the UI; it makes it easy to share insights.

Hemali Parekh
Senior Manager, Software Development

Shift-left is not a single action.
It's a transformation.

With Gatling Enterprise Edition, organizations gain:

- A developer-first testing model that integrates seamlessly into existing workflows
- Fully CI/CD-native automation to validate performance on every commit and build
- A unified workflow that scales smoothly from local tests to global, multi-region campaigns
- Centralized dashboards and reporting that give all teams shared performance visibility
- One consistent performance process across developers, QA, SRE, platform teams, and leadership

---

> Gatling turns performance testing from a last-minute checkpoint into a continuous, collaborative, engineering discipline.

Ready to shift-left with confidence?

Gatling Enterprise Edition brings the performance culture your teams need—long before users ever feel the difference.

Talk to an expert >