Gatling

2026

WHITEPAPER

Scaling performance engineering through distributed ownership

Why the central performance team can no longer be the gate, and how enterprises move performance and reliability to the teams that build, without losing standards, governance, or control.

The day the wall came down

In 2006, Amazon's CTO Werner Vogels described a quiet revolution inside the company. Amazon had torn down the wall between the engineers who wrote software and the operators who ran it. In its place was a single principle, four words long: "You build it, you run it."

The teams who shipped a service now carried the pager for it, owned its performance, and felt its failures directly. Vogels argued this brought developers into daily contact with their customers, and that contact sharpened the service.

Two decades later, that idea has become the defining pressure on how large enterprises organize performance engineering. The old arrangement put a small central team in the middle of the road as a specialist gate, testing and approving releases for dozens of applications.

In a world of cloud-native systems, elastic capacity, and microservices that multiply by the quarter, that gate now creates the very delays it was built to prevent. The pressure is to push performance and reliability toward the teams that build and run services, the way Amazon did.

Yet most enterprises that try to copy Amazon discover the hard part. Handing teams the pager without handing them the tools, standards, and support to succeed simply moves the problem around.

The goal is distributed ownership done right: product teams own the performance of their services, a central platform team paves the road they drive on, and governance lives in policy rather than in a review board. Distribution succeeds only when the tooling on the paved road was built for it, and that is the question this paper puts to the test.

$2.41T

Annual cost of poor software quality in the US, including $1.56T in operational failures.

CISQ, 2022

$400B

Annual cost of unplanned downtime to Global 2000 firms, roughly 9% of profits.

Splunk & Oxford Economics, 2024

4×

Observability leaders resolve downtime in minutes & see 33% fewer outages a year.

Splunk & ESG, 2023

> **Performance stopped being an engineering preference a long time ago. At enterprise scale it is a board-level risk, and the operating model that owns it decides how fast that risk gets found and fixed.**

From a central gate to a paved road

Vogels' principle did not stay inside Amazon. Gartner projects that 80% of large software organizations will run platform engineering teams by 2026, up from 45% in 2022. Platform engineering is the mechanism enterprises use to give teams ownership while keeping the estate coherent. It is how an organization distributes responsibility without distributing chaos.

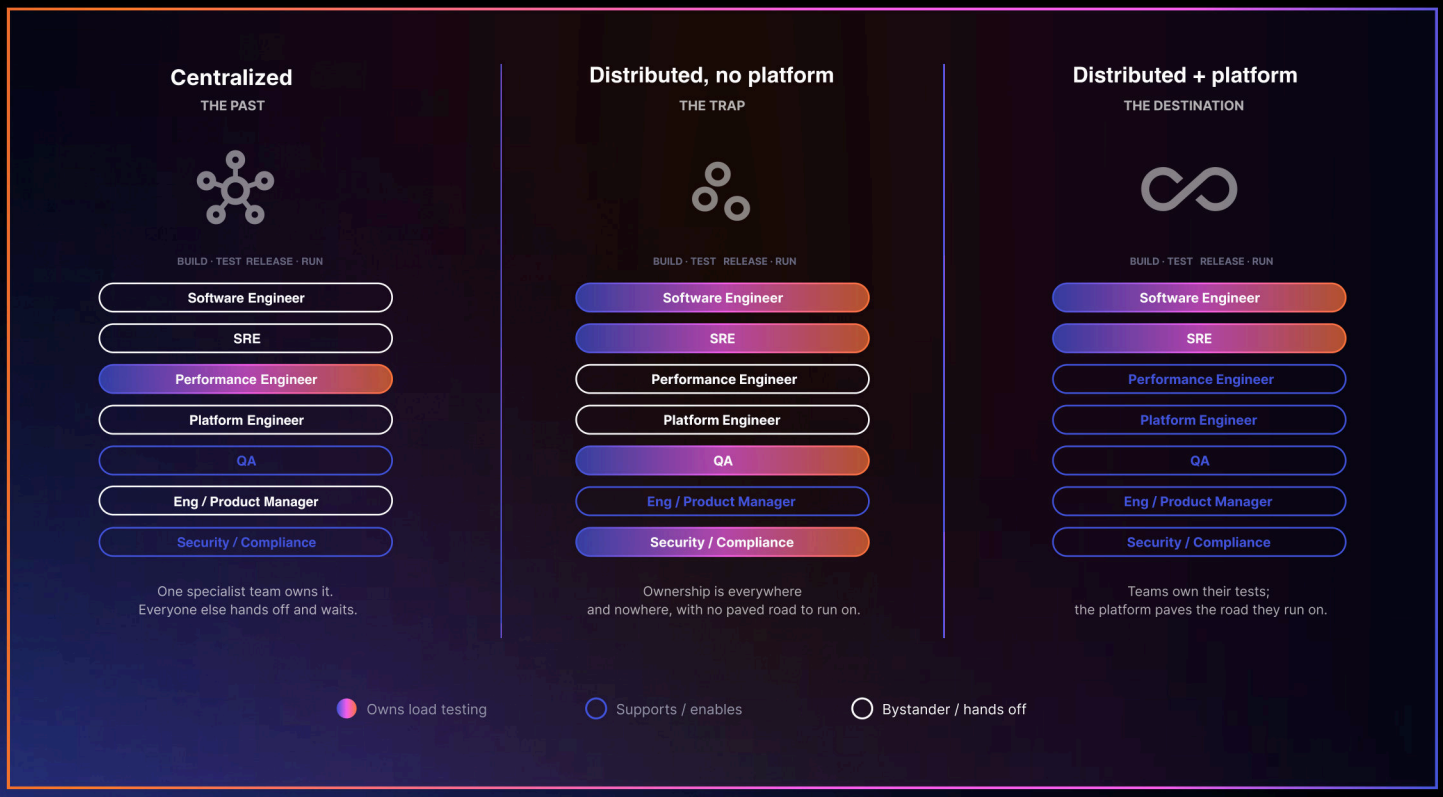
Three models describe the spectrum. The distinction matters, because plenty of organizations claim distributed ownership when all they have done is shift cost from a central team onto delivery teams with no platform underneath them.

MODEL	DESCRIPTION	STRENGTH	RISK	BEST FIT
CENTRALIZED	A specialist team designs tests, approves releases, and owns tooling for dozens of apps.	Deep expertise; strong consistency; easier control in regulated or legacy estates.	Queueing delays, late discovery, weak product context, poor scaling.	Legacy and mainframe-heavy domains; early transformation.
DISTRIBUTED, NO PLATFORM	"Everyone is responsible." Ownership is pushed to teams with no shared road underneath.	Best balance of speed, consistency and scale; cuts cognitive load while keeping local context.	Needs sustained platform investment; fails if the platform becomes a new gatekeeper.	Most large enterprises modernizing into cloud and microservices.
DISTRIBUTED, WITH A PLATFORM	Teams own performance end to end; a central platform paves the road with standards, templates, and governance.	Maximum autonomy; fast feedback; tight code-to-customer alignment.	Wide capability variance, duplicated tooling, governance drift, unstable delivery if platform support is weak.	Mature product organizations with strong internal platforms.

Ownership across the release cycle

The clearest way to tell the three models apart is to ask a single question: across the build-test-release-run cycle, who actually owns load testing? Map the roles onto the CI/CD loop and the difference stops being abstract.

In the centralized model one specialist team carries it while everyone else hands off and waits. Strip away the platform and ownership scatters across every role at once, with no road to run on. Add the platform and the picture resolves: the teams that build a service own its tests, while specialists and governance shift into enabling roles that keep the road paved.



Roles shown are illustrative; titles and boundaries vary by organization. The pattern is what matters: ownership moves to the teams closest to the code, supported rather than gated by the center.

Be careful about these three antipatterns

ANTI-PATTERN	WHY IT FAILS	BETTER PATTERN
"EVERYONE IS RESPONSIBLE" WITH NO PLATFORM	Work fragments; tools diverge; teams reinvent basics	Build the platform and enabling functions before pushing ownership out broadly.
SLOS AS DASHBOARDS ONLY	No effect on planning or release behavior	Attach error-budget policy, burn-rate alerts, and a review cadence.
AVERAGES INSTEAD OF PERCENTILES	Hides tail latency and real customer pain	Use p95/p99 and journey-based SLOs.

A 24-month path to distributed ownership

The solid target is to move from centralized execution to platform-backed team ownership inside the first year, then scale coverage steadily as platform maturity and governance evidence grow.

The goal is broad coverage that is also deep: every critical service with real SLOs, trace coverage, automated checks, and policy behind it, not dashboards that light up green while nobody acts on them.

ANTI-PATTERN	TIMING	KEY MILESTONE	ILLUSTRATIVE KPIS
FOUNDATION	Months 0–3	Inventory tier-one services and critical journeys; name owners; agree SLO standards	100% tier-one services have a named owner; baseline DORA metrics captured
PILOT	Months 3–6	Instrument 5–10 services; stand up dashboards and burn-rate alerts; add smoke tests to CI/CD	70%+ trace coverage on pilot flows; smoke tests in every pilot repo
FEDERATE	Months 6–12	Launch champion network & CoP; publish self-service test templates; automate evidence capture	25–40% of tier-one/two services have SLOs; a champion in every product area
SCALE	Months 12–18	Extend standards across critical domain; tie release decisions to SLO status	60–80% SLO coverage; trace coverage >80% on critical flows
OPTIMIZE	Months 18–24	One-click observability and test capability prune low-value tooling	80%+ SLO coverage; platform adoption is pulled, not mandated

Good news: distributed ownership is a tooling problem

Distributed ownership only scales if the tools on the paved road are built for it. DORA's 2024 research found that internal platforms improve productivity and organizational performance, yet can reduce stability and throughput when built poorly. Read carelessly, that sounds like a reason to hold back from distribution. Read correctly, it says the opposite: distribution succeeds or fails on the strength of the platform beneath it.

The job is to build a platform-and-enablement spine that makes team ownership safe: shared telemetry standards, reusable test frameworks, coaching and communities of practice, service owners who hold their own SLOs and load models, and policy encoded into pipelines so governance scales without a review board. Get that spine right and distribution stops being a risk. It becomes the fastest, most resilient way an enterprise can run

For load testing specifically, distribution sets four hard requirements. The tool must be developer-native (tests written as code engineers already use), CI/CD-friendly (fast enough to gate every change), self-service (templates a product team can adopt without a specialist), and increasingly AI-assisted (so authoring, migration, and result interpretation don't bottleneck on scarce expertise).

Where Gatling fits the model

Distributed ownership only scales if the tools on the paved road are built for it. DORA's 2024 research found that internal platforms improve productivity and organizational performance, yet can reduce stability and throughput when built poorly. Read carelessly, that sounds like a reason to hold back from distribution. Read correctly, it says the opposite: distribution succeeds or fails on the strength of the platform beneath it.

The job is to build a platform-and-enablement spine that makes team ownership safe: shared telemetry standards, reusable test frameworks, coaching and communities of practice, service owners who hold their own SLOs and load models, and policy encoded into pipelines so governance scales without a review board. Get that spine right and distribution stops being a risk. It becomes the fastest, most resilient way an enterprise can run

For load testing specifically, distribution sets four hard requirements. The tool must be developer-native (tests written as code engineers already use), CI/CD-friendly (fast enough to gate every change), self-service (templates a product team can adopt without a specialist), and increasingly AI-assisted (so authoring, migration, and result interpretation don't bottleneck on scarce expertise).

Own your critical journeys
Each team defines and owns the SLOs and load models for the user journeys its services power, rather than waiting in a central queue for someone else to model their traffic.

Test on every change
Performance checks run in CI/CD on every pull request, gating releases automatically. No Jira ticket, no specialist hand-off, no late-stage surprise before a launch.

Run on the paved road
Teams use shared templates, self-service load generators, and central standards, so they get autonomy without reinventing infrastructure or diverging into one-off tooling.

Answer to a shared scoreboard
Ownership comes with common, visible metrics, p95/p99, error budgets, and coverage, so the platform team andleadership can see reliability across every team at a glance.

What it looks like with Gatling underneath

Data proves the pattern is real. The teams below prove it is buildable on Gatling. Each runs distributed ownership at scale, with Gatling Enterprise as the platform that makes team autonomy safe.

Intuit

Serving 100 million customers and processing $2T+ in invoices, Intuit could not run performance testing as a specialist function. Feedback arrived late, regressions reached production, and every team that needed a load test joined a queue.

Four decisions made distribution work: performance test scaffolding ships automatically in every new service repo; CI/CD pipelines onboard services and run tests on every pull request with no ticket and no specialist; templates and libraries are pre-vended and centrally maintained; and one platform gives real-time visibility into who is testing, what is failing, and where coverage gaps sit, across every team.

"Reliability is not an option. It's a feature. Any developer getting started can drive hundreds of thousands of transactions per second with ease. You don't have to tune it. You don't have to call in an expert."

Chaitanya Bhatt, Principal Engineer, Intuit

3,000+	80,000+	97%	30 → 1
Developers onboarded, from under 100 in 2018	Load tests run annually	Perf coverage on critical services	Tools consolidated 100% CI/CD

InPost

Europe's leading out-of-home parcel network coordinates billions of inter-service calls under strict latency targets. Performance tests are a must-pass release gate, with teams codifying guardrails as assertions.

Gatling Enterprise replaced fragmented tooling with autoscaled Kubernetes load generators and Private Locations, GitLab CI, and Dynatrace correlation. A five-day logistics simulation surfaced bottlenecks from code to cables, helping the network scale its daily volume more than six-fold.

"With autoscaled generators we stopped maintaining boxes. We likely saved FTEs, now we pay a subscription instead of paying with people's time."

Mateusz Piasta, Site Reliability Engineer, InPost

10M+	99%	1–5,000	60+
Parcels per day, up from 1.6M	Success rate gate	RPS service targets	Users across 30 teams

5. RECOMMENDATIONS

Ten moves for the next twelve months

1. Make distributed ownership the goal, and the platform the way you reach it.
   Centralized execution rarely scales; distribution without a platform spine is just cost-shifting
2. Define service owners and critical journeys before buying tooling.
   Tooling without ownership produces dashboards without decisions.
3. Standardize instrumentation with OpenTelemetry.
   Keep the backend flexible; make telemetry shape, naming, and collection consistent.
4. Make distributed ownership the goal, and the platform the way you reach it.
   Centralized execution rarely scales; distribution without a platform spine is just cost-shifting
5. Define service owners and critical journeys before buying tooling.
   Tooling without ownership produces dashboards without decisions.
6. Standardize instrumentation with OpenTelemetry.
   Keep the backend flexible; make telemetry shape, naming, and collection consistent.
7. Make distributed ownership the goal, and the platform the way you reach it.
   Centralized execution rarely scales; distribution without a platform spine is just cost-shifting
8. Define service owners and critical journeys before buying tooling.
   Tooling without ownership produces dashboards without decisions.
9. Standardize instrumentation with OpenTelemetry.
   Keep the backend flexible; make telemetry shape, naming, and collection consistent.
10. Standardize instrumentation with OpenTelemetry.
    Keep the backend flexible; make telemetry shape, naming, and collection consistent.

Build the paved road on tooling that was made for it

Gatling Enterprise Edition is built for distributed ownership

PRIVATE LOCATIONS

Deploy load generators inside your VPC, your Kubernetes cluster, or your on-premises data center. Test traffic never leaves your network.

SLOs

Define p95 and error rate thresholds in Gatling. Breach them and the pipeline stops automatically.

ENTERPRISE SECURITY

Plugs into the governance frameworks your organization already enforces.via SSO (OIDC/SAML), role-based access controls, and audit logs.

AI ASSISTANT

Accelerates how developers create, explain, and optimize, and understand performance tests and Gatling simulations

CI/CD INTEGRATION

Trigger runs, enforce thresholds, and gate promotions from GitLab CI, GitHub Actions, or any pipeline.

TEST AS CODE

Write scenarios in Java, Kotlin, JS/TS, or Scala using the same toolchain your backend and version control engineers already use.

SUPPORTED PROTOCOLS

HTTP, WebSocket, SSE, gRPC, JMS, and MQTT support APIs, streams, messaging, microservices.

COMPARISON AND TRENDS

Run comparison and trends help teams spot regressions, track performance over time, and see whether each release improves, holds, or degrades under load.

INFRASTRUCTURE AS CODE

Spin injector fleets up and down as IaC. Nodes appear in your VPC, disappear after the run.

OBSERVABILITY

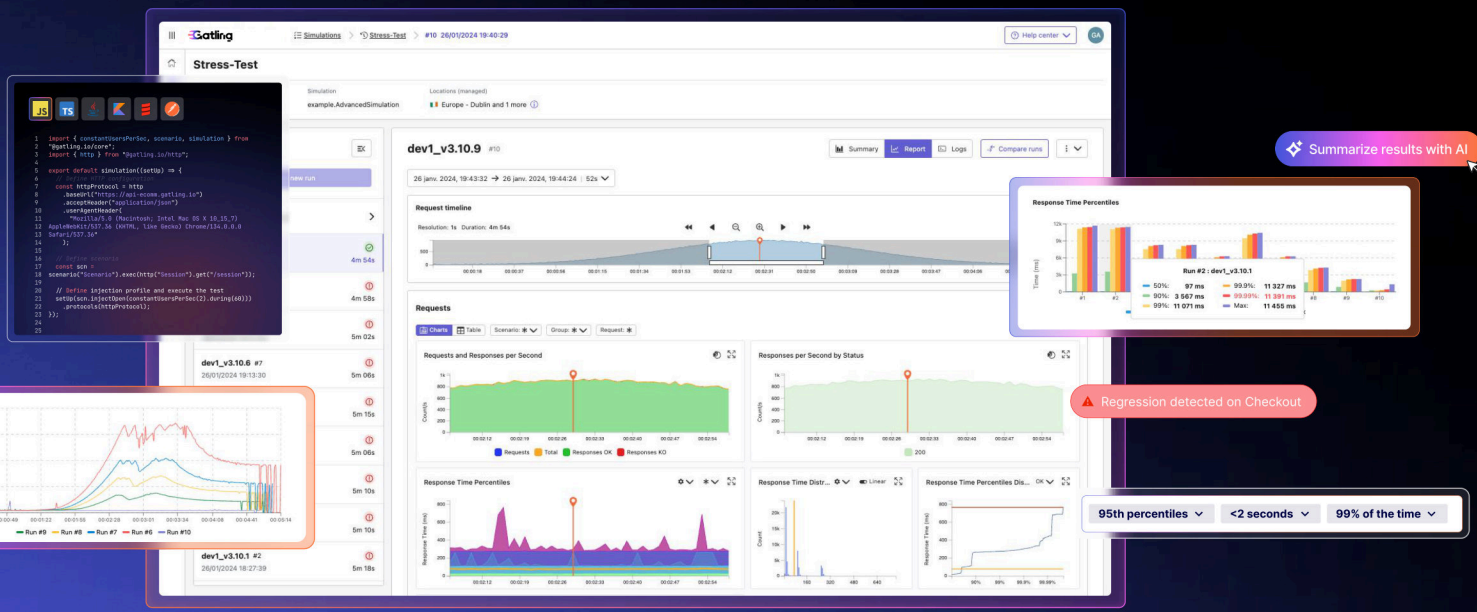
Integrations with APM tools enable you to correlate load test behavior with infrastructure metrics

Reliability is now a competitive advantage

In a market where alternatives are always one search away and subscription churn is the primary business risk, the organizations that win are the ones whose software keeps working when it matters most. Functionally correct, yes, but also performant, resilient, and reliable at the scale of real usage

The four delivery moments mapped here — product launches, CI/CD pipelines, API-first architectures, and legacy modernization — are where that reliability advantage is built or lost. They are the moments when assumptions about system behavior meet the reality of production load. And they are the moments when continuous performance intelligence is the difference between catching a problem in a pipeline and discovering it in a customer complaint.

Gatling Enterprise Edition is the platform built for this reality. Test as Code makes performance tests a first-class engineering artifact. CI/CD integration makes performance gates a first-class delivery control. Private locations, distributed load generation, and protocol coverage make it possible to test the architectures modern software actually uses — not the simplified approximations that fit inside a single load injector SLO compliance scoring, AI assistance, and observability integration turn test data into the business intelligence that drives decisions.



Gatling

Gatling Enterprise Edition 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 you to the next level

Talk to an expert ›