Gatling

2026

WHITEPAPER

Case study: performance testing across banking, capital markets, and insurance

Performance failures are no longer a technical inconvenience; they have become a business crisis.

Most performance failures aren't visible. They show up as a reconciliation window missed at month-end. A payment authorization timing out during peak retail. A legacy migration that passes every pre-launch check, then degrades under real transaction volume.

In July 2024, one that was visible crashed 8.5 million systems globally. Banking absorbed $1.15 billion in direct losses; not from a sophisticated attack, but from a change nobody validated under production conditions.

The institutions most exposed are not the ones with the worst technology. They're the ones that rely on monitoring to catch failures after they happen, rather than testing to prevent them before they do.

This paper makes the case for performance testing as a risk management discipline rather than an engineering activity. It covers what's at stake across banking, capital markets, and insurance; how two European financial institutions operationalized it with Gatling Enterprise Edition; and a maturity framework leaders can use to evaluate where their organization stands.

The core question is simple: are you managing performance risk, or accepting it?

USE CASE 1 | BANKING AND PAYMENTS

When the system handling money fails

In financial services, the system handling money can't afford to fail. But most performance risks are invisible until peak.

95% API-driven architecture. Sub-100ms industry benchmark. Foreseeable spikes that aren't being tested for.

THE RISK JOURNEY OF A SINGLE TRANSACTION

One payment authorization. Four external dependencies. A 100ms window.

1. **Customer request**
   Auth & session validation **Risk:** Session timeout under concurrent load
2. **Your backend**
   Payment registration service **Risk:** Passes unit tests in isolation, chokes when the reconciliation pipeline competes for DB connections
3. **Fraud detection**
   External API call risk: Third-party latency spikes cascade inward. Your SLA, not theirs
4. **FX exchange**
   Currency conversion **Risk:** Rate-fetch failures cause silent transaction drops at peak volume
5. **Card network**
   Authorization & settlement **Risk:** Timeout cascades from upstream failures compound here
6. **Reconciliation**
   Settlement & ledger close **Risk:** Month-end window missed → regulatory reporting failure

WHAT CAN GO WRONG

60% Surge in API downtime incidents in financial services — Q1 2024 to Q1 2025

145-210% Latency increase in payment APIs during month-end processing

84% Of anomalies originate from inter-service interactions, not isolated components

The risk: Delayed settlements. Failed authorizations at peak. Reconciliation windows missed, cascading into regulatory reporting failures. Customer trust that takes years to rebuild in an industry where switching providers takes minutes.

Test beyond payment authorization

Settlement & reconciliation windows
Simulate month-end and end-of-day volumes on settlement services. Confirm the window closes on time before regulators are involved.

Third-party dependency chains
Model realistic failure modes for external APIs (FX rates, card networks, KYC providers) and test how your system degrades when they slow down or fail.

Core banking API stress
Test account creation, deposit processing, and ledger calls under realistic concurrency. Isolate which microservice degrades first.

Reduce risk with Gatling.
Welcome to Continuous Performance Intelligence.

AI capabilities
Automated run summaries surface what went wrong without manual analysis. IDE integration and MCP server support bring load testing into the workflows your engineers already use.

SLOs and compliance scoring
Define response time and error rate targets directly in Gatling Enterprise Edition. Every run returns a compliance score, not a pass/fail, a precise percentage. Know exactly how long your system held the line.

Test as code
Write load test scenarios in Java, JavaScript or Scala. Versioned, reviewed, and living in the same repository as your application. Plug directly into your CI/CD pipeline.

Private locations
Load generation runs inside your network perimeter. No traffic leaves, no firewall exceptions. Test what's actually in production.

USE CASE 2 | CAPITAL MARKETS

Where milliseconds are worth millions

In high-frequency trading, a 1 millisecond latency advantage can be worth $100 million annually to a major brokerage.

Uptime requirements push to five nines (99.999%).
Every second of downtime during market hours can cost dearly.

THE RISK JOURNEY OF A SINGLE TRADE

One click to buy. Multiple systems in motion.
A few milliseconds to get it right.

1. **Trader request**
   Order submission from web, mobile, or internal desk **Risk:** Session validation, stale state, or request bursts delay the trade
2. **Order management system**
   Order creation and routing logic **Risk:** Looks stable in isolation, then slows down when market data updates, position checks, and downstream routing all hit at once
3. **Risk and compliance checks**
   Pre-trade controls and policy validation **Risk:** Latency here holds up the trade. Under load, every slowdown can push orders outside acceptable windows
4. **Market data feed**
   Price lookup and quote validation **Risk:** Delayed or missing market data leads to stale pricing, rejected orders, or trades executed on outdated assumptions
5. **Execution venue or broker**
   Exchange, ECN, or external liquidity provider
   Risk: Third-party slowdowns and timeout cascades show up as your failure, not theirs
6. **Post-trade processing**
   Trade confirmation, booking, reconciliation, and ledger updates **Risk:** The trade executes, but downstream systems fall behind.

WHAT CAN GO WRONG

$100M Annual value of a 1ms latency advantage for a major brokerage

99.999% uptime required in trading environments

LOSS OF CONFIDENCE in the financial system

The risk: A performance Gap in capital markets is not simply downtime. It means competitive position lost in milliseconds, reporting windows missed, and regulatory exposure that compounds with every hour the system isn't fully operational.

Test beyond the trading day

Trade under real market pressure
Simulate market open, volatility spikes, and event-driven surges, and order routing systems. Make sure orders are accepted, priced, and routed fast enough when volume rises all at once.

Protect settlement windows
Test end-of-day, month-end, and settlement workloads under realistic conditions. Confirm critical post-trade services complete on time before delays or regulatory problems.

Validate systems
Model slowdowns and failures across market data feeds, exchanges, clearing partners, payment rails, and KYC providers.

Reduce risk with Gatling.
Welcome to Continuous Performance Intelligence.

AI capabilities
Automated run summaries surface what went wrong without manual analysis. IDE integration and MCP server support bring load testing into the workflows your engineers already use.

SLOs and compliance scoring
Define response time and error rate targets directly in Gatling Enterprise Edition. Every run returns a compliance score, not a pass/fail, a precise percentage. Know exactly how long your system held the line.

Test as code
Write load test scenarios in Java, JavaScript or Scala. Versioned, reviewed, and living in the same repository as your application. Plug directly into your CI/CD pipeline.

Private locations
Load generation runs inside your network perimeter. No traffic leaves, no firewall exceptions. Test what's actually in production.

USE CASE 3 | INSURANCE

Legacy systems as a ticking clock

Each integration milestone is a point where untested behavior under load can silently undermine the migration's business case.

A new API wrapper, a cloud-native component running alongside legacy infrastructure, a newly supported payment method, they can all disrupt things.

THE RISK JOURNEY OF INSURANCE MODERNIZATION

Legacy systems, cloud migration, and untested load.
Silent performance risk at every integration point.

1. **Legacy core**
   25 to 30 year old policy, billing, or claims platform
   Risk: Stable under known conditions, but brittle under new concurrency patterns.
2. **API wrapper**
   New integration layer around the legacy core **Risk:** Adds a new load surface. What worked as batch traffic now becomes real-time demand.
3. **Cloud-native services**
   Modern services running alongside legacy **Risk**: They scale differently, fail differently in ways teams often don't test until production
4. **New payment methods**
   Modern billing, payouts, and partner payment flows
   Risk: Every new provider, payment rail, or workflow adds another dependency that can fail silently
5. **Cross-system orchestration**
   Data and transactions moving between old and new stacks **Risk:** The handoff points become the weak points
6. **Operational milestone**
   Renewals, claims spikes, billing runs, or reporting windows **Risks:** Untested behavior under load can fail turning migration progress into operational risk

WHAT CAN GO WRONG

6% of insurance companies still run mainframe architecture averaging 25-30 years old

39% of insurers say legacy technology is actively blocking innovation

120K+ records harvested from GEICO and Travelers via credential stuffing

The risk: You've invested in dashboards that tell you when things break. Without performance testing embedded in your delivery process, you're missing the window to find out before they do, when you can still do something about it.

Test beyond peak policy events

Handle quote and claims surges
Simulate quote spikes, catastrophe-driven claims volumes, renewal peaks, and partner-driven traffic across policy, billing, and claims systems.

Protect renewal windows
Test end-of-day, month-end, and renewal-cycle workloads under realistic conditions. Confirm invoicing, policy updates, claims processing, and reporting workflows complete on time

Validate critical dependencies
Model slowdowns and failures across payment providers, identity verification services, data sources, and document generation platforms.

Reduce risk with Gatling.
Welcome to Continuous Performance Intelligence.

AI capabilities
Automated run summaries surface what went wrong without manual analysis. IDE integration and MCP server support bring load testing into the workflows your engineers already use.

SLOs and Compliance scoring
Define response time and error rate targets directly in Gatling Enterprise Edition. Every run returns a compliance score, not a pass/fail, a precise percentage. Know exactly how long your system held the line.

Test as code
Write load test scenarios in Java, JavaScript or Scala. Versioned, reviewed, and living in the same repository as your application. Plug directly into your CI/CD pipeline.

Private locations
Load generation runs inside your network perimeter. No traffic leaves, no firewall exceptions. Test what's actually in production.

USE CASE 4 | FINTECH

Modernization adds speed and risk

Real-time finance depends on systems that fail asymmetrically.
In fintech, not every component slows down the same way under pressure.

The result is not always an outage. More often, it is rising latency, retry storms, queue buildup, and failed transactions when customers expect instant execution.

THE RISK JOURNEY OF A SINGLE FINTECH TRANSACTION

One payment. Multiple systems in motion.
Milliseconds between approval and failure.

1. **Customer request**
   Card payment, bank transfer, account funding, or wallet action **Risk:** Session delays, authentication retries, or traffic bursts slow the request.
2. **Transaction service**
   Payment initiation, orchestration, and routing **Risk:** Works cleanly in isolation, then degrades when ledger writes, balance checks, and downstream calls
3. **Fraud and identity checks**
   Risk scoring, verification, and policy enforcement
   Risk: Latency blocks the whole transaction. Under load, every call increases abandonment rate
4. **External dependency**
   Banking API, card network, FX provider, or KYC service
   Risk: Third-party slowdowns show up as your outage, not theirs.
5. **Authorization and ledger update**
   Approval, balance update, and transaction recording
   Risk: The payment is approved, but ledger or balance services fall behind, creating inconsistencies
6. **Settlement and reconciliation**
   Post-transaction matching, clearing, and reporting
   Risks: Peak periods bottlenecks. Miss the processing window, and compliance risk follow

WHAT CAN GO WRONG

15-25% reduction in revenue due to transaction drop-offs

85% of customers stay loyal to apps they trust

99% of the most significant errors in fintech occur in real-time operations management

The risk: You've invested in observability that tells you when users are already feeling the problem. Without performance testing embedded in delivery, you're still finding out too late.

Test beyond everyday transaction volume

Keep operations on schedule
Simulate the processing peaks that happen after the transaction itself: posting, ledger synchronization, reconciliation, exception handling, and reporting.

Handle payment surges
Simulate spikes in account creation, deposits, transfers, card authorizations, and partner-driven traffic across payment, fraud, and ledger systems.

Validate third-party systems
Model slowdowns and failures across banking APIs, card networks, FX providers, fraud engines, and identity services.

Reduce risk with Gatling.
Welcome to Continuous Performance Intelligence.

AI capabilities
Automated run summaries surface what went wrong without manual analysis. IDE integration and MCP server support bring load testing into the workflows your engineers already use.

SLOs and compliance scoring
Define response time and error rate targets directly in Gatling Enterprise Edition. Every run returns a compliance score, not a pass/fail, a precise percentage. Know exactly how long your system held the line.

Test as code
Write load test scenarios in Java, JavaScript or Scala. Versioned, reviewed, and living in the same repository as your application. Plug directly into your CI/CD pipeline.

Private locations
Load generation runs inside your network perimeter. No traffic leaves, no firewall exceptions. Test what's actually in production.

Going beyond load testing and monitoring

Your observability stack already defines what good looks like. Datadog, Dynatrace, New Relic, InfluxDB — these APM tools capture SLOs for latency, error rates, and throughput in production. **Performance testing lets you apply those same thresholds before production, catching regressions at the point where they're still cheap to fix.**

The question isn't monitoring or testing. It's whether your load tests are using the same success criteria your dashboards already enforce.

With Gatling Enterprise Edition, the same SLOs you track in your observability platform become automated pass/fail gates in your CI/CD pipeline. A payment flow that exceeds your p95 latency threshold fails the build before it reaches the environment your dashboards are watching.

Closing this gap means moving performance testing from a periodic activity to a continuous discipline, embedded in delivery pipelines, automated where possible, and governed as part of the release process. Not replacing monitoring, but complementing it.

Monitoring catches what slipped through; performance testing makes sure less slips through in the first place.

Generating high-scale load in a secure environment

Across every financial services company in Gatling's portfolio, one pattern holds without exception: load generators must run inside the company's own infrastructure.

Gatling is built for API-heavy architectures — the kind financial services runs on. Authentication flows, payment endpoints, settlement APIs, fraud detection calls: Gatling handles the full complexity of modern API testing, including JWT, mTLS, HMAC, dynamic tokens, and chained requests across microservices.

More than a preference, this is a non-negotiable and a hard constraint. Banking APIs sit behind VPNs. Payment systems are VPC-bound. Services aren't exposed to the internet. A SaaS-only testing tool that sends traffic from outside your perimeter simply cannot reach the systems that matter most.

This is the single most common reason financial companies displace legacy tools. LoadRunner has been replaced at multiple institutions specifically because it cannot test non-internet-facing services. JMeter struggles with distributed high-load scenarios. NeoLoad lacks CI/CD integration. In finance, the tooling choice is shaped as much by security and infrastructure constraints as by performance capabilities.

There's also a technical dimension unique to financial services: TLS and encryption overhead from security-heavy flows is often the CPU bottleneck on load generators, not virtual user count. Custom security libraries, certificate validation, and encrypted API calls all drive compute costs that generic testing tools don't account for.

Gatling Enterprise Edition is built for financial services

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.

DISTRIBUTED LOAD GENERATION

Spread injection across multiple private nodes so no single machine becomes a CPU bottleneck on high-encryption payment workloads.

What financial services companies actually test with Gatling?

EPI Company, backed by 16 major European banks, is building Wero, a European payment scheme operating across five countries.

From early development, EPI made performance testing a core engineering practice. Using Gatling Enterprise Edition, their teams test security-heavy flows including API authentication (JWT, mTLS, HMAC), client certificate validation, mobile performance under real-world latency, and high-volume transactional traffic.

15+ engineers across multiple teams collaborate on performance tests. Private locations on AWS keep test traffic inside their infrastructure. SSO and Datadog integrations bring performance testing in line with their security and observability stack.

✓ **THE RESULT?** Performance testing is embedded in their release process, not a bottleneck in it.

Nickel, a BNP Paribas subsidiary, serves 4.5 million accounts through 8,200 retail partners across France. During social benefit disbursements, traffic spikes 20 to 30 times baseline for one to two days. Every transaction must still complete in under three seconds.

Performance testing at Nickel isn't optional. It's mandatory before every release, part of their Change Advisory Board review. Each application must demonstrate no degradation compared to the previous version.

Using Gatling Enterprise Edition's private locations, load generators run inside Nickel's own on-premises environment, a non-negotiable for a banking application.

✓ **THE RESULT?** They're now expanding to automated testing across 20 critical applications and planning denial-of-service simulations.

Both followed the same journey: start with Gatling Community Edition, hit an operational ceiling around visibility, governance, and infrastructure control, then move to Enterprise to scale across teams. Both deploy load generators inside their own perimeter, integrate testing into release governance, and treat performance results as decision inputs, not just dashboards to check.

A performance risk framework for financial services

Use these questions to assess where your organization stands; bring them to your next leadership meeting as a starting point for the conversation.

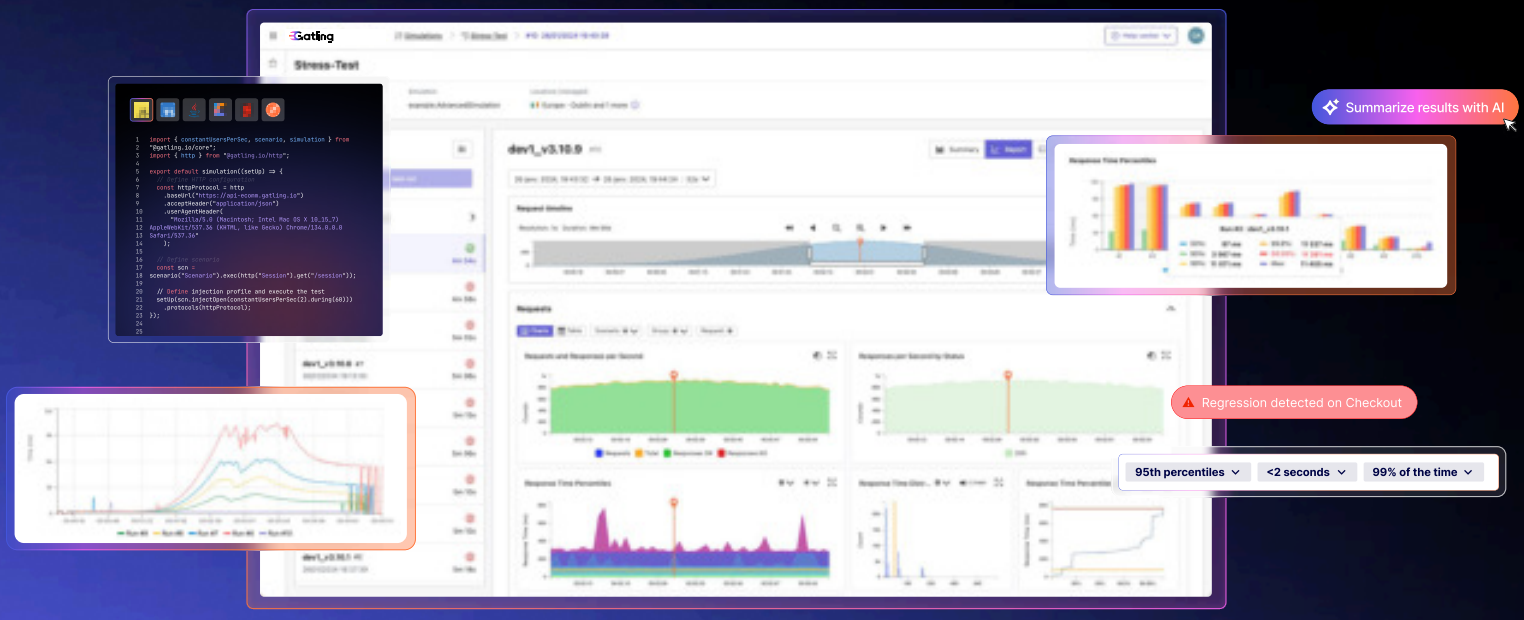
Building performance into the business

Financial institutions that treat performance testing as an engineering checkpoint will keep getting surprised by predictable failures.

The ones that treat it as an operational discipline, with clear ownership, realistic conditions, and decision-driven results, will ship faster, fail less, and operate with the confidence that earns and keeps trust in an industry where trust is everything.

The pattern is the same across every financial services company in this whitepaper: test private, business-critical systems inside your own infrastructure. Scale from open source to enterprise when governance and visibility matter. Embed testing into your release process. Use results to make decisions, not just produce reports.

Your next peak event, month-end close, or major release is already scheduled. The question is whether you'll know your systems are ready before it arrives, or after.



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 ›