Gatling

2026

WHITEPAPER

Case study: performance testing for broadcasting and streaming

When every second of buffering is a business decision

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

Most broadcast and streaming organizations don't operate at Super Bowl scale.
But every one of them has their version of a tentpole event: a match day, a season premiere, a live concert, a product launch stream. And in every case, the moment of peak demand is known in advance.

The traffic spike is foreseeable. The failure, if it happens, is preventable.

The broadcasting industry is undergoing a structural transformation. Traditional linear television is migrating to IP delivery. OTT platforms are competing for subscriber attention measured in seconds of startup delay. Ad-supported streaming models make every failed ad insertion a direct revenue event. And live sports — the last content category driving appointment viewing — now runs on the same CDN infrastructure as everything else, at 10 to 100 times normal operating load.

Performance failures in this environment are not technical inconveniences. They are viewer defection events. According to research across 150 video services, when more than 2% of viewing sessions are subpar, new user churn can reach 50% — more than double the average OTT churn rate in the United States. A buffering ratio that crosses 2% of total viewing time is not a metric to watch. It is the boundary between a platform viewers trust and one they leave.

This paper makes the case for performance testing as a core operational discipline in broadcasting and streaming — not a pre-launch checklist, not a quarterly exercise.

It covers what's at stake across four critical delivery scenarios, how broadcasting organizations can operationalize it with Gatling Enterprise Edition, and a maturity framework teams can use to evaluate where they stand.

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

USE CASE 1 | LIVE SPORTS & TENTPOLE EVENTS

When the world watches at once

Live events are the most demanding infrastructure problem in streaming. Traffic does not ramp gradually. It arrives in surges.

CDN providers sustain request rates exceeding 12 million requests per second during peak live events.

THE RISK JOURNEY OF A LIVE BROADCAST EVENT

One tentpole event. Six systems under simultaneous pressure. A 60-second window where failure becomes defection.

1. **Viewer request**
   Authentication and session validation
   Risk: Session timeout under concurrent peak load
2. **CDN edge**
   Content delivery from nearest point of presence
   Risk: Cache miss rate spikes when content is new; origin pulls flood the ingest layer
3. **Origin servers**
   Master playlist and segment delivery
   Risk: At 5% miss rate and 10M viewers, that's 500,000 simultaneous origin requests
4. **Video encoding pipeline**
   Live transcoding and packaging
   Risk: Encoding latency adds to end-to-end delivery delay; segment generation falls behind under load
5. **Ad insertion**
   Dynamic ad insertion at scale
   Risk: Ad decision server latency causes buffering; ad timeout causes content gap or viewer exit
6. **Adaptive bitrate delivery**
   ABR ladder management per viewer
   Risk: Bitrate degradation at scale; viewers drop to lowest tier or abandon during quality adaptation

STATS AT A GLANCE

10–100x Traffic surge required during major live broadcast events

25–35% Traffic spike within seconds of a scoring play as viewers simultaneously join or replay content

$14,056 Average cost per minute of IT downtime in 2024

The risk: A CDN that passes capacity planning assumptions fails when the top 1% of channels — which account for more than 70% of total platform traffic — all spike at once. Startup failures at kickoff. Buffering during decisive moments. Ad insertion timeouts that turn a 30-second break into a viewer's exit point.

Test beyond game day

Simulate the spike, not the average
Test injection profiles must simulate simultaneous connection establishment: thousands of sessions starting within seconds. Validate your CDN cache warm-up strategy before the event,

Validate capacity under load
AWS recommends a 95% CDN cache hit rate as the planning baseline. That means 5% of peak concurrent viewers hit your origin simultaneously.

Test ad insertion under load
Test your SSAI/CSAI infrastructure at full event concurrency. A 200ms ad insertion timeout that passes at 100,000 concurrent viewers may fail at 2 million.

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.

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

USE CASE 2 | OTT PLATFORM RELIABILITY & SUBSCRIBER SCALE

Where startup delay is a subscriber decision

OTT platforms compete on experience, not just content. In a market with 5.27B active users, platform performance is a retention factor.

Delays past the 2s threshold, buffering ratios above 2% of viewing time, and bitrate drops are issues viewers notice even when they can't name them.

THE RISK JOURNEY OF A SUBSCRIBER SESSION

One user opens the app. Six systems must respond correctly, quickly, and in sequence.

1. **Authentication**
   Login, token validation, entitlement check
   Risk: Auth service degradation under concurrent login surge on season premiere or new release day
2. **Content discovery**
   Homepage personalization, search, recommendations
   Risk: Recommendation engine latency adds to perceived startup time before the user even hits play
3. **Playback initiation**
   License acquisition, DRM validation, manifest fetch
   Risk: DRM service timeout causes black screen; manifest fetch delay is the first buffering event
4. **CDN delivery**
   Segment delivery from edge cache
   Risk: Cache miss on new content forces origin pull at scale; miss rate spikes sharply on premiere day
5. **Adaptive bitrate management**
   ABR algorithm selects quality tier per session
   Risk: Rebuffering ratio and average bitrate are the top causal QoE drivers.
6. **Background telemetry**
   Playback metrics, heartbeats, quality reporting
   Risk: Telemetry backpressure under scale degrades real-time monitoring.

STATS AT A GLANCE

$5.27B Global OTT users in 2024

5.8% Viewer abandonment rate per additional second of delay beyond 2 seconds

50% New user churn rate when more than 2% of viewing sessions are subpar; more than double the average OTT churn rate

The risk: You have invested in dashboards that tell you when users are already having a bad experience. Without performance testing embedded in your delivery pipeline, you are finding out after the damage is done.

Test beyond average viewing sessions

Make startup latency a metric
Two seconds of video start delay produces 5.8% abandonment per additional second. Test the full startup journey — auth, DRM, manifest, first segment — not just individual endpoints.

Simulate premiere day traffic
New content releases concentrate traffic in a way that normal load profiles don't capture. Simulate the simultaneous login surge, content discovery spike, and cold CDN cache conditions of a premiere.

Run soak tests
Subscription platforms run 24/7. Streaming quality degradation often compounds over time. Soak tests are the only way to find these before subscribers do.

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.

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

USE CASE 3 | AD DELIVERY & MONETIZATION INFRASTRUCTURE

Where performance failures are revenue

Every failed ad insertion is not a user experience issue. It is a direct revenue loss event.

Every 100ms of ad decision server latency at 10 million concurrent viewers is a compounding financial exposure.

THE RISK JOURNEY OF A VIDEO AD IMPRESSION

One viewer reaches an ad break.
Four systems must respond in under 200 milliseconds.

1. **Ad trigger**
   Player signals ad break to ad manager
   Risk: Simultaneous ad break signals across millions of concurrent sessions create a request spike
2. **Ad decision server**
   Programmatic bid request and creative selection
   Risk: Third-party ad server latency cascades inward; your SLA, not theirs, even when the failure originates externally
3. **Creative delivery**
   VAST tag fetch and media asset retrieval
   Risk: Creative CDN miss on new campaigns; ad asset delivery latency causes timeout before the ad starts
4. **SSAI stitching**
   Server-side ad insertion into the video stream
   Risk: SSAI pipeline latency under load adds to the ad break gap; stitching errors cause black to the viewer
5. **Measurement and attribution**
   Data and transactions moving between old and new stacks **Risk:** The handoff points become the weak points
6. **Operational milestone**
   Impression tracking, quartile events, completion
   Risks: Tracking beacon failures invalidate impressions for billing; revenue recognized but not delivered

STATS AT A GLANCE

16% of streamed video ads that failed to play globally in Q2 2021 — down from 47% in 2018

$1.9B annual revenue loss from live-streamed video latency over CDNs in 2021

$9B projected accumulated loss through 2026 under HTTP streaming conditions

The risk: Ad delivery infrastructure sits at the intersection of real-time video delivery and programmatic advertising systems. When they don't, the failure is invisible to the viewer (who sees a black screen or a content gap) but immediately visible in fill rate and revenue reporting.

Test beyond normal ad load

Simulate ad break events

Unlike on-demand, where ad calls are staggered naturally across sessions, live broadcast concentrates ad request load into a single instant.

Test ad-to-content transitions

Buffering that occurs during the transition from ad to content is disproportionately damaging to viewer experience.

Model third-party latency failures

Your ad delivery chain includes external DSPs, SSPs, and ad servers outside your control. Model their latency degradation and timeout behavior.

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.

Distributed load generation across regions

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

USE CASE 4 | LEGACY BROADCAST MODERNIZATION

The hidden risk in every migration milestone

Traditional broadcasters are migrating from linear delivery and on-premises infrastructure to IP-based, cloud-native architectures.

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

THE RISK JOURNEY OF A BROADCAST MODERNIZATION

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

1. **Legacy ingest**
   Broadcast ingest and media asset management
   Risk: Stable under known conditions, but brittle under new concurrency patterns introduced by IP delivery
2. **Transcoding pipeline**
   Encoding and packaging for multi-format delivery
   Risk: What worked as scheduled batch processing now becomes real-time demand under concurrent requests
3. **API integration layer**
   APIs connecting legacy systems to modern delivery
   Risk: Each new API wrapper adds latency and a new failure surface; integration points become the weak points under load
4. **Origin infrastructure**
   Content origin for CDN delivery
   Risk: Capacity planning that worked for linear audiences fails under the demands of simultaneous sessions
5. **Multi-platform delivery**
   Simultaneous delivery across web, mobile, and CTV
   Risk: Peak platform load doesn't align with what the testing team modeled
6. **Operational milestones**
   Live events, content launches, seasonal peaks
   Risks: Untested behavior surfaces exactly when the business needs the infrastructure to hold

STATS AT A GLANCE

15-25% annual growth in digital viewership for major live sporting events

13x of customers stay loyal to apps they trust

Uptick in outages driven by rising demand, legacy strain, and added complexity from 5G and software-defined networking

The risk: You have invested in monitoring that tells you when things break. Without performance testing embedded in your migration process, you are missing the window to find out before they do, when you can still do something about it.

Test beyond the migration checkpoint

Validate hybrid architecture

During migration, legacy and cloud-native systems run in parallel. The performance risk is highest at the handoff points — where requests cross from the old stack to the new one.

Simulate your future peak

At 15–18% annual viewership growth, infrastructure adequate for today's peak events will be under-provisioned within three years.

Add regression detection

Embed load tests in the CI/CD pipeline so every release is validated against performance thresholds before it reaches production.

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.

Distributed load generation across regions

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

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 broadcasting and streaming company in Gatling's portfolio, one constraint holds consistently: load generators must be capable of simulating the actual concurrency patterns of live events — not just average traffic.

Gatling is built for the API-heavy, microservices-based architectures that modern streaming platforms run on. Authentication flows, content delivery APIs, ad decision calls, playback reporting endpoints, license acquisition services — Gatling handles the full complexity of modern streaming API testing, including JWT, OAuth, dynamic tokens, and chained requests across microservices.

There is a technical dimension unique to live broadcasting: the simultaneous connection establishment pattern that occurs at event start. Unlike e-commerce, where load ramps gradually, live broadcast concentrates millions of connection attempts into a window of seconds. Load generators that ramp linearly cannot reproduce this pattern. Gatling's injection profiles are designed to simulate the instantaneous surge that broadcast infrastructure must actually absorb.

For broadcast platforms testing origin infrastructure that cannot be exposed to the internet — origin servers, internal ad systems, legacy ingest pipelines — load generators running inside the network perimeter are not a preference. They are a requirement. Private locations make this possible without firewall exceptions or architectural compromises.

Gatling Enterprise Edition is built for broadcasting and streaming 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 you can achieve with Gatling

Gatling Enterprise Edition enables teams to simulate millions of virtual users and millions of requests per second, across any architecture: monoliths, APIs, microservices, Kubernetes clusters, and globally distributed streaming platforms.

Up to 60,000
concurrent virtual users
per load generator

Up to 300,000
requests per seconds
on each load generator

They generate millions of requests per second for any kind of architecture

Adobe · CIRCLE K · Desjardins · HMH · InPost · TUI

Streaming leaders generate massive traffic to prepare for record-breaking peaks

BBC · CANAL+ · france•tv · M6+ · MultiChoice · TV5MONDE

What customers leveraging Gatling Enterprise Edition can reach

200+
Automated tests running in parallel, 24/7

100+
Load generators running simultaneously

5 million+
Concurrent virtual users with 20 load generators

Gatling

What broadcast companies actually test with Gatling

JioStar: engineering for the spike inside the spike

ioStar is India's leading media and entertainment company, operating JioCinema, a platform that holds exclusive streaming rights to the IPL, MotoGP, and the Olympics. During major events, it routinely sustains more than 30 million concurrent users.

The infrastructure challenge at JioStar is the spike inside the peak. When a cricket wicket falls, or a legendary batsman walks to the crease, concurrent users can surge by 7 to 8 million within 90 seconds. That is not a traffic ramp. It is a wall. And unlike a Super Bowl kickoff, it is unpredictable.

With Gatling Enterprise Edition, a team of 8 developers and automation engineers now validates JioCinema's infrastructure at 30 million concurrent users, with spike simulation of 7 to 8 million users within 90 seconds.

✓ **THE RESULT?** Benchmarking against realistic load profiles has allowed JioStar to avoid over-scaling while maintaining the performance thresholds their viewers expect.

Canal+ Group operates in 52 countries with 26.4 million subscribers and more than 400 million monthly active users worldwide. Its myCanal platform delivers live TV, on-demand video, and premium sports coverage across web, mobile, smart TVs, and set-top boxes.

The myCanal platform runs on a complex microservices architecture spanning authentication, licensing, catalog, DRM, and CDN distribution. The complicating factor was that Canal+ tests had to be executed against production itself, with all the risk that entails.

Using Gatling Enterprise Edition, Canal+'s quality engineering team built a progressive, iterative testing strategy. Each campaign combined capacity, soak, and stress test types, with load increasing across iterations based on what the previous session revealed.

✓ **THE RESULT?** Zero incidents during a major live football broadcast serving millions of concurrent viewers. Increased load thresholds across all critical services.

JioStar and Canal+ operate in different markets and at different scales, but their performance testing journeys follow the same arc. Both moved to Gatling Enterprise when governance, collaboration, and infrastructure control became non-negotiable. Both treat performance testing as a continuous discipline, not a pre-event ritual. And in both cases, the shift was engineer-driven: the teams running the tests are also the teams improving the platform.

BETTER TESTING STARTS WITH BETTER QUESTIONS

A performance risk framework for broadcasting and streaming

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

1 COVERAGE: ARE WE TESTING WHAT ACTUALLY MATTERS?

Live event delivery infrastructure, CDN origin capacity, ad insertion pipelines, and authentication flows deserve different testing cadences than internal dashboards and admin tools. Most organizations test what's easy to reach, not what's critical to protect.

- [ ] **Can you name your five most critical delivery paths in your platform?**
- [ ] **Were each of them tested under realistic peak conditions in the last 90 days?**
- [ ] **Do you test your CDN origin, ad insertion pipeline, and authentication flows separately?**
- [ ] **Are test cadences tied to event calendars and release schedules, not just sprint cycles?**

2 REALITY: DO OUR TESTS REFLECT WHAT ACTUALLY HAPPENS?

Smooth ramps and clean test environments produce clean results and false confidence. Live events spike. Premiere day CDN cache hit rates are fundamentally different from day fourteen. Ad break timing creates simultaneous request concentration that staged testing environments routinely fail to simulate.

- [ ] **Does your test environment include the same CDN configuration and origin setup as production?**
- [ ] **Do your load scenarios account for simultaneous connection establishment at event start?**
- [ ] **Do you simulate foreseeable spike events — game day, premiere night, live concert?**
- [ ] **Do load generators run inside your own infrastructure, behind your VPN or VPC?**

3 INTERPRETATION: DO WE KNOW WHAT TO DO WITH THE RESULTS?

A test without a decision path is just a report. Every performance test should answer a specific question, with a named owner and a clear path for what happens if it passes or fails.

- [ ] **Does each recurring test have a named owner or team accountable for the results?**
- [ ] **Is there a specific, documented question each test is designed to answer?**
- [ ] **Is there a predefined decision path for both pass and fail outcomes?**
- [ ] **Did your last performance test produce a ship/hold decision, not just a dashboard?**

4 OWNERSHIP: WHO IS ACCOUNTABLE?

Performance can't live with one team. Platform engineers, CDN operations, ad tech teams, and engineering leaders each need a defined role. Business leadership needs visibility into trends over time.

- [ ] **Do developers own performance for the code they ship?**
- [ ] **Does business leadership have regular visibility into platform performance trends across releases?**
- [ ] **If your live event startup failure rate doubled today, is it clear who makes the launch/hold call?**
- [ ] **Do your load tests simulate realistic viewer journeys, not just isolated endpoint hits?**

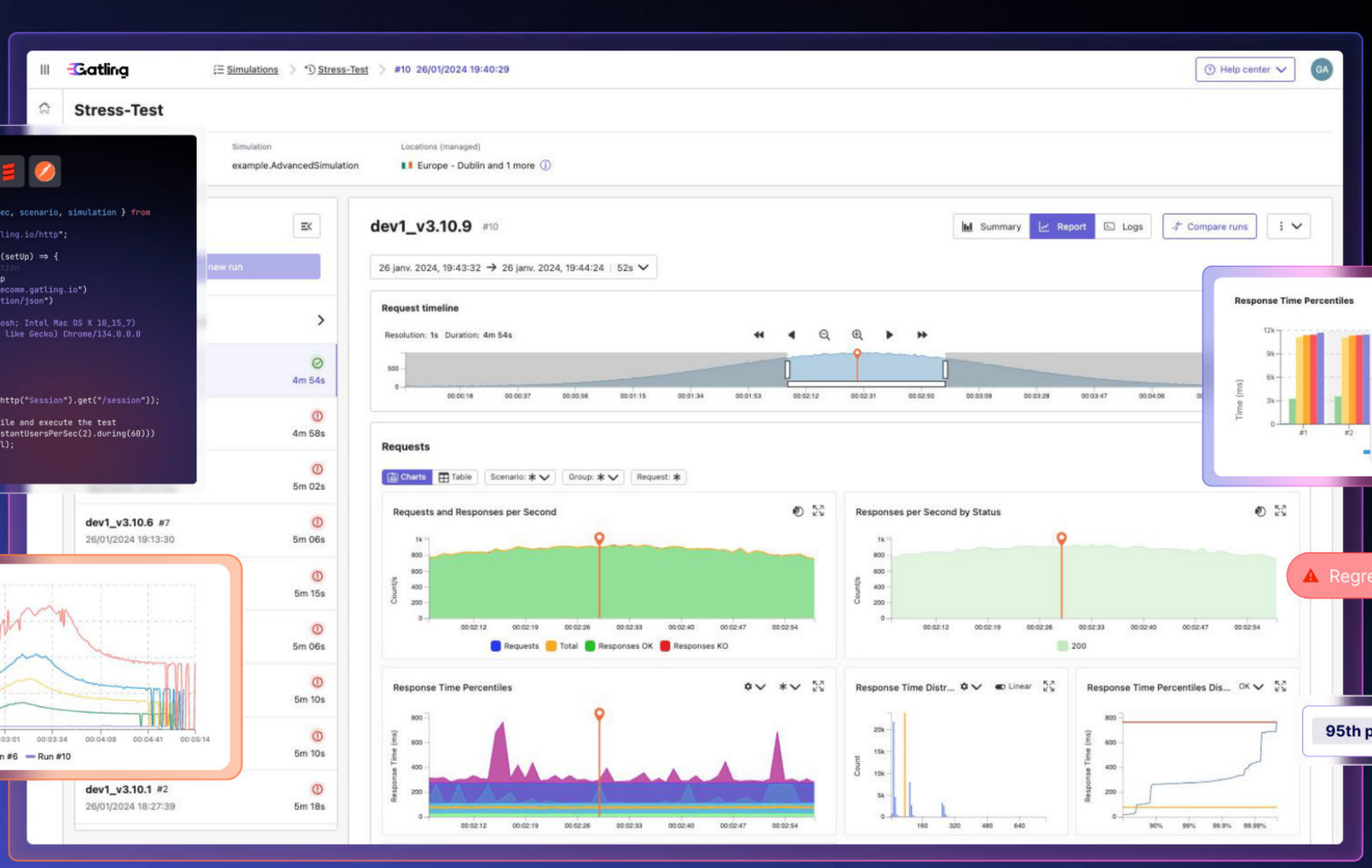
Building performance into the business

Broadcasting and streaming organizations that treat performance testing as a pre-event checklist will keep being surprised by predictable failures.

The ones that treat it as an operational discipline — with clear ownership, realistic conditions, and decision-driven results — will handle their next tentpole event with the infrastructure confidence that earns and keeps viewer trust in an industry where attention is the only currency that matters.

The pattern is the same across every organization in this whitepaper: test private, business-critical infrastructure inside your own network. Scale from point-in-time testing to continuous performance intelligence when governance and visibility matter. Embed testing into release pipelines. Use results to make decisions, not just produce reports.

Your next live event, season premiere, or major platform release is already on the calendar.
The question is whether you'll know your infrastructure is ready before viewers find out, 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 ›