

- EBOOK

# Optimizing cloud infrastructure costs with load testing

A practical FinOps guide for engineering and platform teams

## Introduction

Cloud infrastructure has made scaling applications easier than ever, and overspending just as easy. Autoscaling, pay-as-you-go pricing, and managed services promise flexibility, yet many organizations still rely on rough estimates, historical traffic, or oversized “safety buffers” when managing environments.

The result is predictable: non-production clusters running 24/7, autoscaling that reacts too late or too early, and infrastructure bills that outpace user growth. Traditional FinOps often catches these inefficiencies after the fact—once the money is already spent.

**Load testing changes this dynamic.** By simulating realistic traffic patterns before production, engineering and platform teams can:

- Expose scaling gaps and hidden bottlenecks early.
- Right-size environments based on evidence, not guesswork.
- Quantify cost/performance trade-offs for architectural decisions.

This whitepaper provides a practical, engineering-driven guide to controlling cloud costs through load testing. It focuses on key capabilities that help teams identify inefficiencies, optimize scaling strategies, and institutionalize cost-aware practices, illustrated with examples from Gatling Enterprise Edition, a modern load testing platform.



- OPTIMIZING CLOUD INFRASTRUCTURE COSTS WITH LOAD TESTING

# The FinOps challenge in modern cloud environments

## Why autoscaling alone isn't enough

Autoscaling is a cornerstone of modern cloud architectures, but it's inherently reactive. Scaling rules typically rely on fixed utilization thresholds, such as "add instances when CPU > 70 %." These values are often chosen heuristically, not empirically, and the scaling process itself can take minutes.

By the time new capacity comes online, users may have already experienced degraded performance.

Conversely, conservative thresholds can lead to premature scale-ups, adding cost without delivering real user benefit. Both scenarios stem from the same root cause: lack of data about how the system behaves under actual load conditions.

## Common cost waste patterns

Several recurring patterns contribute to runaway cloud spend:

- **Always-on non-production environments:** Staging or pre-production clusters left running overnight or during weekends.
- **Safety buffers everywhere:** Over-sizing instances or clusters "just in case," with no clear justification.
- **Lagging autoscalers:** Thresholds that are too optimistic or too slow to react, causing either over-spend or performance degradation.
- **Invisible inefficiencies:** Cloud billing and monitoring tools rarely highlight underused resources or suboptimal scaling decisions.
- **One-time decisions:** Infrastructure sizing choices made early in a project often persist unchanged for years.

## Where load testing changes the equation

Load testing allows teams to close the loop between expected traffic and actual infrastructure behavior.

Instead of waiting for real users to reveal scaling inefficiencies, engineering teams can:

- Rehearse scaling events under controlled conditions.
- Measure capacity limits and failure modes before production.
- Correlate performance signals with infrastructure metrics to detect waste.
- Identify the cost-performance sweet spot through empirical evidence.

When used systematically, load testing transforms FinOps from an after-the-fact accounting exercise into a proactive engineering discipline.

## Key capabilities to optimize infrastructure costs with load testing

Modern load testing platforms offer more than just virtual user generation.

The following five capabilities enable engineering and platform teams to directly address cloud cost challenges. Examples from Gatling Enterprise Edition illustrate how these concepts can be applied in practice.

## Generate realistic load patterns to reveal hidden inefficiencies

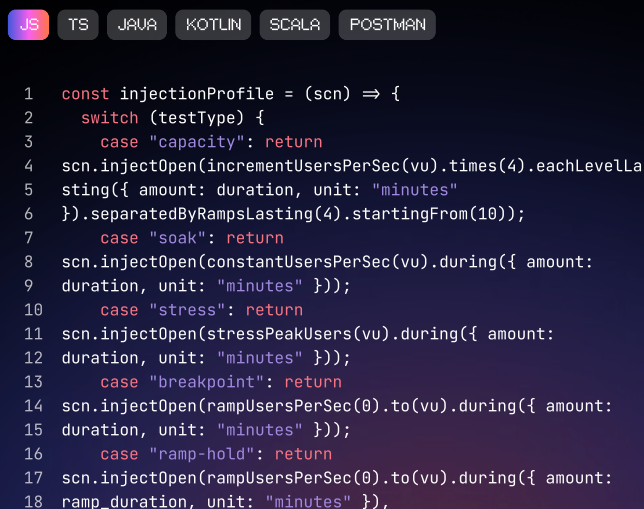
**Benefit:** Understand how your system behaves under realistic user conditions, not synthetic extremes.

Realistic load modeling is the foundation of meaningful cost and performance insights. Before production traffic arrives, teams can run controlled tests that reproduce actual usage behaviors, such as:

- Baseline steady states (e.g., average daily traffic).
- Gradual ramps to mimic organic growth.
- Traffic spikes that stress autoscaling and queuing systems.
- Sustained plateaus to observe long-term resource consumption.

By progressively increasing virtual users and observing response times, error rates, and resource utilization, teams can pinpoint the exact load levels where performance starts degrading or costs accelerate sharply.

This allows you to refine autoscaling thresholds, right-size clusters, or optimize database connections before production users are impacted.



```
1  const injectionProfile = (scn) => {
2    switch (testType) {
3      case "capacity": return
4      scn.injectOpen(incrementUsersPerSec(vu).times(4).eachLevelLa
5      sting({ amount: duration, unit: "minutes"
6      }).separatedByRampsLasting(4).startingFrom(10));
7      case "soak": return
8      scn.injectOpen(constantUsersPerSec(vu).during({ amount:
9      duration, unit: "minutes" }));
10     case "stress": return
11     scn.injectOpen(stressPeakUsers(vu).during({ amount:
12     duration, unit: "minutes" }));
13     case "breakpoint": return
14     scn.injectOpen(rampUsersPerSec(0).to(vu).during({ amount:
15     duration, unit: "minutes" }));
16     case "ramp-hold": return
17     scn.injectOpen(rampUsersPerSec(0).to(vu).during({ amount:
18     ramp_duration, unit: "minutes" })),
```

Gatling Enterprise Edition supports sophisticated injection profiles (ramps, spikes, plateaus, and custom scenarios) enabling teams to reproduce realistic load shapes that match expected traffic patterns.

## Scale load generation across regions and services

**Benefit:** Test distributed architectures the way they will actually be used in production.

For global applications or microservices-based systems, generating load from a single location isn't enough. Network latencies, regional routing, and service boundaries can create bottlenecks that only appear under distributed, high-scale conditions.

By running load generators in multiple regions simultaneously, teams can:

- Identify geo-specific bottlenecks (e.g., slow routing through a specific region).
- Validate multi-region autoscaling behaviors.
- Reproduce complex traffic topologies, such as API calls originating from mobile users worldwide.
- Measure the cost impact of scaling distributed services under real conditions.

 **AP - Hong kong**

 **AP - Tokyo**

 **AP Pacific - Mumbai**

 **AP SouthEast - Sydney**

 **Europe - Dublin**

Gatling Enterprise Edition enables distributed load generation, letting teams orchestrate tests across multiple load generators and regions to mimic real-world, large-scale user traffic.

## Track cost and performance regressions over time

**Benefit:** Make infrastructure tuning a continuous, data-driven process.

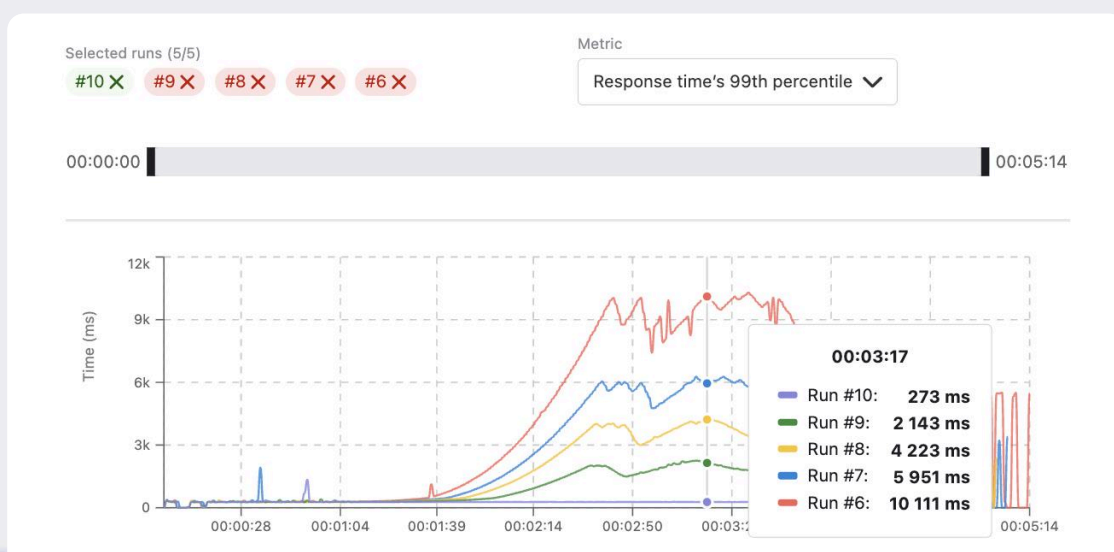
Cost inefficiencies often creep in gradually. A new feature might increase API call volume, or a database query might grow more expensive over time. These regressions are easy to miss without longitudinal visibility.

Regularly scheduled load tests (daily, weekly, or aligned with release cycles) create a performance and cost baseline. By comparing successive runs, teams can detect:

By running load generators in multiple regions simultaneously, teams can:

- Slower response times under the same load → potential performance regressions.
- Higher resource consumption for similar traffic → potential cost regressions.
- Changes in scaling behavior → drift in autoscaling configurations or architecture.

This requires consistent test naming, controlled environments, and clear comparison methodologies. Run-to-run comparison tools are essential to separate real regressions from expected variability.



Gatling Enterprise Edition provides Run Trends and Run Comparison features. Teams can overlay multiple test results, visualize percentile response times and throughput over time, and detect regressions quickly.

## • OPTIMIZING CLOUD INFRASTRUCTURE COSTS WITH LOAD TESTING

# Connect performance metrics with infrastructure behavior

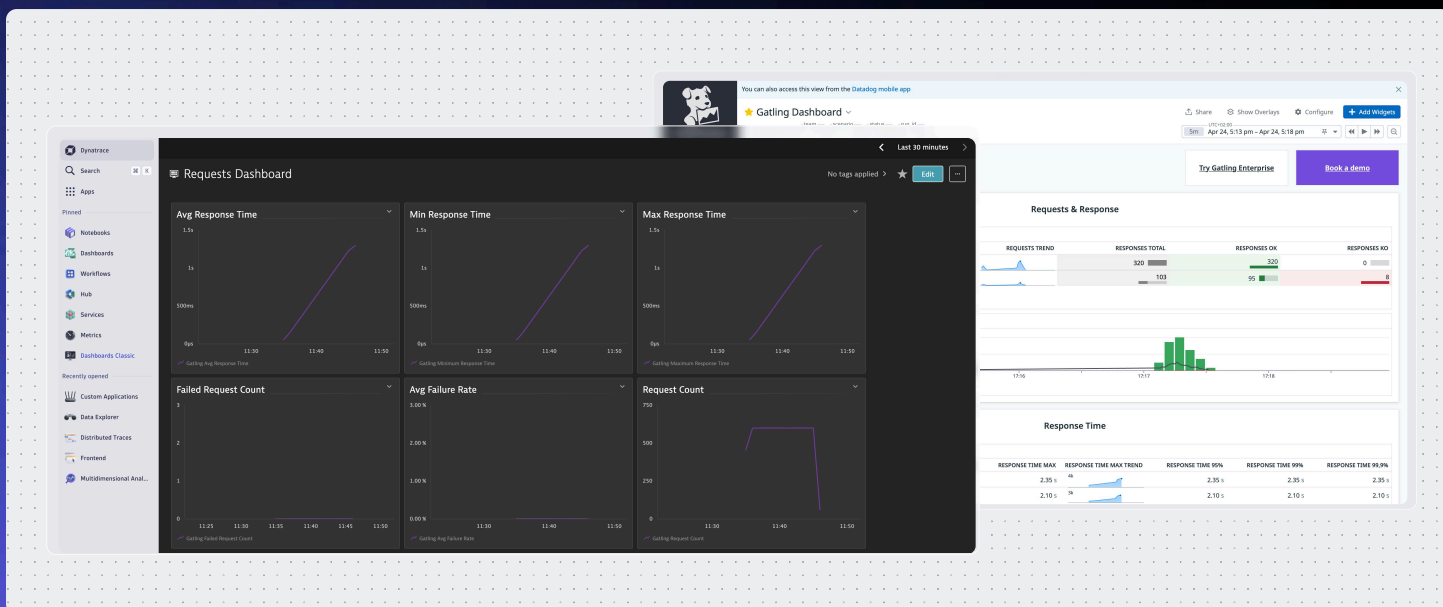
**Benefit:** Correlate application performance under load with underlying infrastructure signals to uncover hidden waste.

Load testing generates rich performance data: response times, error rates, throughput, connection metrics, and more. However, without correlating these with infrastructure and APM data, the picture remains incomplete.

By integrating load testing with APM platforms (e.g., Datadog, Dynatrace), teams can correlate:

- Spikes in latency with CPU or memory saturation on specific services.
- Scaling delays with slow pod scheduling or delayed capacity provisioning.
- Connection failures with network or TLS bottlenecks.

These correlations reveal **inefficient scaling behaviors** (for example, autoscaling rules that spin up compute after latency has already spiked) or components that are over-provisioned relative to their actual load.



Gatling Enterprise Edition integrates with leading APM tools, enabling teams to view load test results alongside infrastructure telemetry in real time.



## Validate cost thresholds automatically

**Benefit:** Prevent cost and performance regressions from reaching production through automated gates.

Manual review of load test results doesn't scale. For FinOps practices to be sustainable, cost and performance validations must be automated. This is where assertions become powerful.

Assertions allow teams to define quantitative thresholds, such as:

- "P95 latency must remain below 500 ms at 10,000 concurrent users."
- "The error rate must stay under 1 % for all endpoints."
- "Throughput for key transactions must stay within an expected range."

If a load test exceeds these thresholds, the pipeline can automatically fail or flag the issue. Over time, these thresholds can be tied not only to performance but also to cost projections: for example, rejecting configurations that trigger unnecessary scale-ups at low loads.



```
1 // Assert that the max response time of all requests is less than
2 100 ms
3 setUp(scen.injectOpen(injectionProfile))
4   .assertions(global().responseTime().max().lt(100));
5
6 // Assert that every request has no more than 5% of failing
7 requests
8 setUp(scen.injectOpen(injectionProfile))
9   .assertions(forAll().failedRequests().percent().lte(5.0));
10
11 // Assert that the percentage of failed requests named "MyRequest"
12 in the group "MyGroup" is exactly 0 %
13 setUp(scen.injectOpen(injectionProfile))
14   .assertions(details("MyGroup",
15 "MyRequest").failedRequests().percent().is(0.0));
```

Gatling Enterprise Edition supports detailed assertions that can be used to gate CI/CD pipelines, ensuring that only configurations meeting defined SLOs and cost targets proceed.

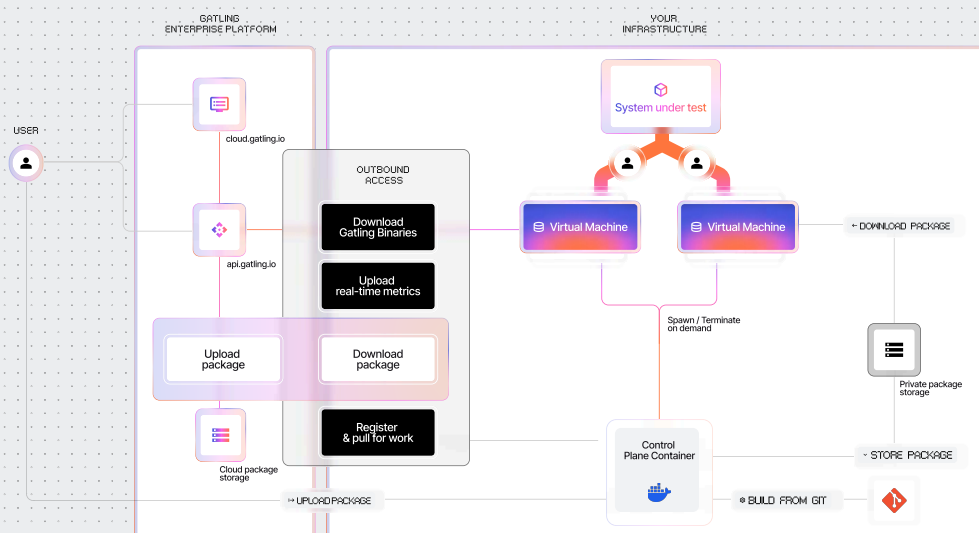
## Institutionalize cost-efficient testing practices

**Benefit:** Make cost optimization part of engineering culture, not one-off audits.

Even the best tools are ineffective without disciplined practices. Mature organizations implement several practices to ensure load testing informs cost decisions systematically:

- **Source-controlled test definitions:** Storing simulations in Git ensures reproducibility and version control.
- **Configuration-as-Code:** Defining packages and simulations as code allows automated deployment, auditing, and rollback of test configurations.
- **Clear run naming conventions:** Meaningful test names (e.g., api-checkout-baseline-v3) make historical analysis and run comparisons easier.
- **Scheduled baseline tests:** Weekly or pre-release runs to detect regressions early.
- **Centralized result sharing:** Dashboards or shared links so leadership and finance teams can see trends without deep technical tooling.

These practices make load testing an integral part of platform engineering and FinOps, not an isolated QA activity.



Gatling Enterprise Edition supports Build-from-Git pipelines, Configuration-as-Code for packages and simulations, and easy run labeling. These capabilities help teams industrialize their testing workflows.



## Practical test scenarios to optimize cloud costs

Load testing isn't just about resilience, it's also a powerful lever to uncover infrastructure inefficiencies and reduce cloud waste. Below are six Gatling Enterprise Edition test scenarios that platform and engineering teams can easily launch to support FinOps goals.



### Autoscaling spike test

**Goal:** Validate how fast and efficiently autoscaling responds to sudden surges

**Scenario:** Use spike injection profiles from multiple regions to simulate flash traffics

**FinOps impact:** Reveals scaling delays and overreactions, helping teams fine-tune rules to minimize both wasted capacity and degraded performance.



### Peak capacity ramp-up Test

**Goal:** Identify the precise load at which your system starts to degrade.

**Scenario:** Gradually ramp virtual users from baseline to peak and beyond, monitoring latency, error rates, and resource utilization.

**FinOps impact:** Enables evidence-based sizing for peak traffic periods, avoiding expensive "just-in-case" provisioning.



### Pre-release cost regression test

**Goal:** Catch cost-impacting performance changes before deployments go live.

**Scenario:** Launch distributed load tests from various geographic regions to simulate global user behavior.

**FinOps impact:** Prevents silent cost regressions (e.g., new features that increase backend load) from reaching production.



### Baseline steady-state test

**Goal:** Measure real infrastructure usage under normal conditions.

**Scenario:** Run a constant user load that mirrors average daily traffic in a staging or pre-production environment.

**FinOps impact:** Highlights oversized environments and idle capacity that can be reduced without affecting user experience.



### Off-hours idle load test

**Goal:** Detect resources that remain active unnecessarily during low-traffic periods.

**Scenario:** Run very light constant loads at night or on weekends to observe autoscaling down behavior.

**FinOps impact:** Uncovers opportunities to automate shutdowns or scale-down schedules for non-production environments.



### Multi-region load distribution test

**Goal:** Understand the cost and performance implications of serving traffic from multiple regions.

**Scenario:** Launch distributed load tests from various geographic regions to simulate global user behavior.

**FinOps impact:** Helps optimize regional placement, routing strategies, and autoscaling configurations to reduce latency and unnecessary cross-region traffic costs.

**Pro Tip:** Start by implementing 1–2 of these scenarios for your most critical services. Document baseline results and use Gatling Enterprise Edition's trend and comparison features to track improvements over time. This structured approach turns load testing into a FinOps feedback loop, not just a one-time exercise.

- OPTIMIZING CLOUD INFRASTRUCTURE COSTS WITH LOAD TESTING

# Choosing the right platform for modern load testing

When performance matters, your load testing platform becomes mission-critical. The complexity of today's distributed systems, global user bases, and rapid release cycles requires more than just generating virtual users, it demands precision, scale, and deep integration with engineering workflows.

Gatling Enterprise Edition is designed to meet these demands. It provides a complete, production-grade platform built to help teams analyze faster, collaborate better, automate with confidence, and test at any scale. Its capabilities are structured around five core pillars that support the entire performance engineering lifecycle:



## Analyze smarter & 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 automations

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



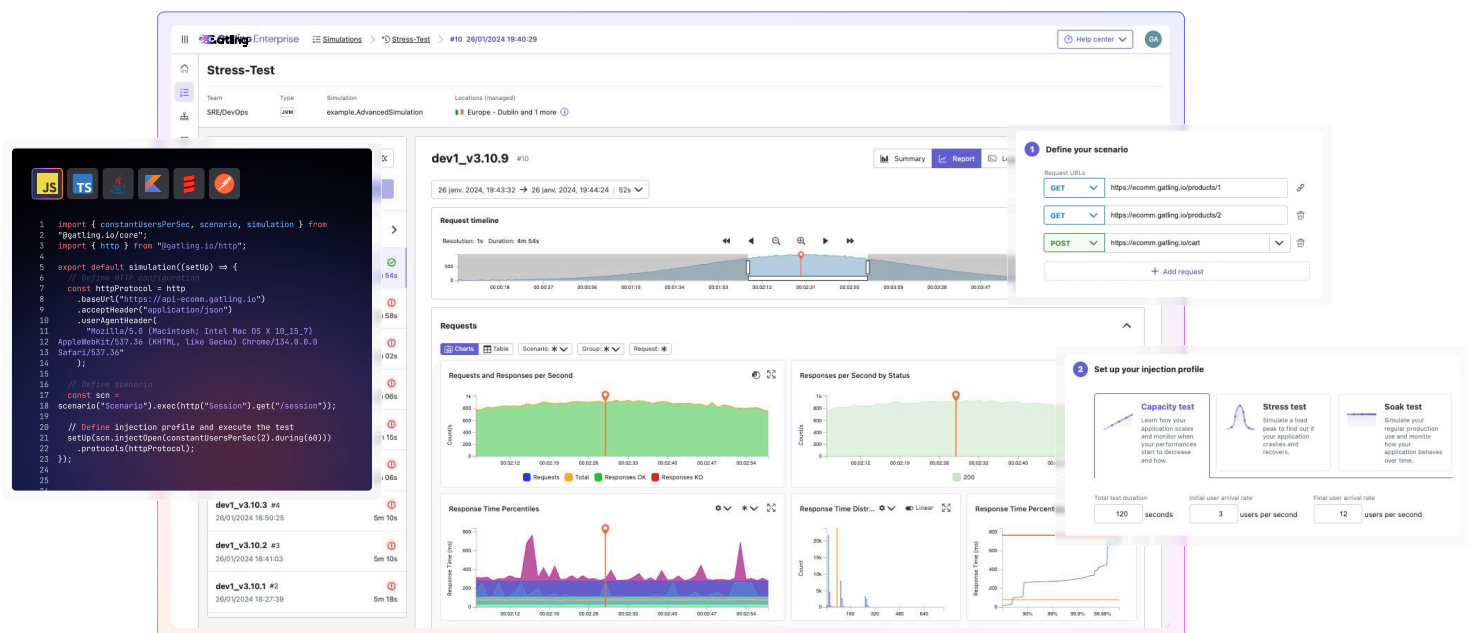
## Collaborate & share results easily

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 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](#) >