Gatling

2025

DATASHEET

Why is Gatling the best load generation engine?

Discover the high-performance engine behind Gatling's load generation architecture

Common blockers to achieving your load testing goals

What engineering teams consistently run into when trying to run reliable, high-performance load tests.

Test architectures are complex and time-consuming to build and maintain

Standing up a reliable load testing environment requires heavy engineering effort.

Common issues:

- Inability to simulate real concurrency due to blocking or thread-per-user models
- Fragile distributed setups that break under CPU, network, or configuration drift
- Metrics pipelines that drop samples or flatten peaks when throughput increases
- High operational load just to keep load generators, configurations, and environments aligned

Even existing load testing setups often fail to scale

Most engines saturate compute, memory, or sockets long before reaching realistic traffic levels.

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

Large-scale tests become too expensive to run

Inefficient engines require too many machines to achieve meaningful load.

Common issues:

- Large load generator fleets driving up cloud compute costs
- High CPU, memory, and bandwidth consumption for each virtual users
- Unpredictable test costs as traffic scales
- Long warm-up/cool-down periods that waste infrastructure time

Modern environments require flexible & secure deployment models

Today's systems span SaaS, Kubernetes, private VPCs, and on-prem environments. Load testing must adapt.

Common issues:

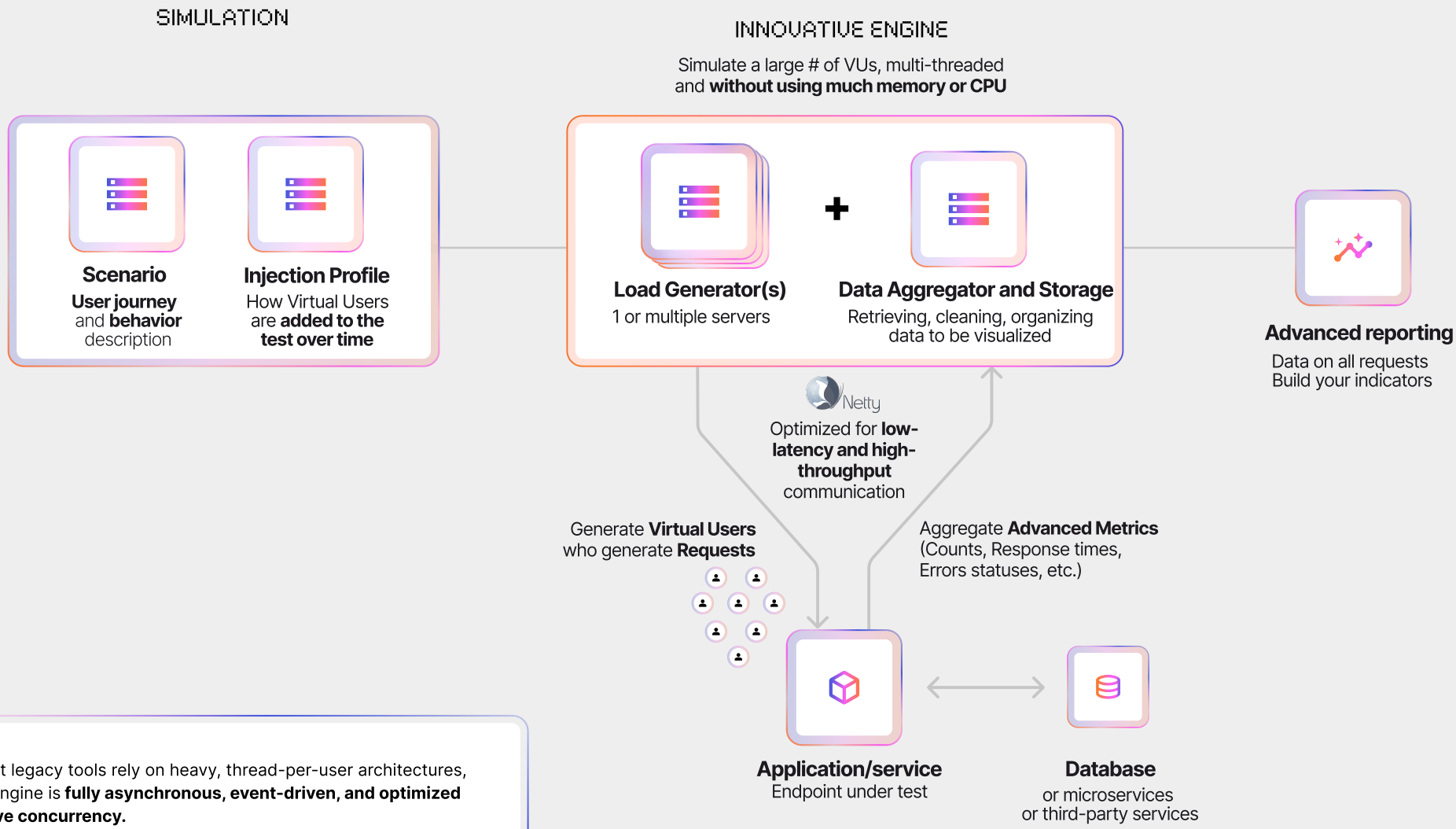
- Difficulty running tests behind firewalls or inside private networks
- Limited support for hybrid setups (SaaS-controlled, self-managed load generators)
- No ability to run load generators close to microservices for low-latency tests
- Lack of secure, compliance-friendly options for regulated workloads

To overcome these structural limitations, teams need a load testing engine that is purpose-built for real concurrency, predictable performance, and seamless operation at scale. **This is exactly where Gatling stands apart.**

Gatling

Inside the Gatling load generation architecture

Gatling Enterprise Edition is powered by one of the **most scalable, resource-efficient, and battle-tested load generation architecture in the industry.**



While most legacy tools rely on heavy, thread-per-user architectures, Gatling's engine is **fully asynchronous, event-driven, and optimized for massive concurrency.**

This design allows organizations to **simulate millions of virtual users, sustain extreme throughput, and execute hundreds of parallel tests**, all with **far fewer machines than traditional solutions.**

Operating at the physical limits of a machine

Why Gatling's engine delivers real concurrency, predictable performance, and extreme throughput with fewer machines.

CONSTRAINT	DESCRIPTION	HOW GATLING REACHES MAXIMUM CAPACITY
~64,000 concurrent sockets per target server	A TCP connection corresponds to a quadruplet (local IP, local port, target IP, target port). With one local IP, a machine can open about 64,000 concurrent connections to the same server.	Gatling's Netty-based engine uses non-blocking I/O to fully leverage available sockets and sustain maximum concurrency.
60-second port reuse delay	The Linux kernel enforces a ~60s TIME_WAIT before reusing closed TCP ports, a hard-coded safeguard to prevent duplicate packet collisions.	Gatling reuses persistent connections within virtual user sessions to minimize reconnections and keep throughput steady.
TLS handshake CPU cost	Establishing HTTPS connections requires CPU-intensive encryption key exchanges and validation, which grow heavier with stronger ciphers.	Gatling integrates BoringSSL for efficient, secure TLS operations and optimized session reuse.
Variable resource cost per request	CPU, memory, and bandwidth consumption vary depending on payload size, response complexity, and scenario design.	Gatling's asynchronous scheduler balances I/O and computation to maintain high concurrency under real-world loads.

> A single load generator can realistically sustain up to **60,000 concurrent virtual users or 300,000 requests per second**, depending on protocol complexity and service architecture.

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
TV5 MONDE

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

Deployment models for load generation

Gatling Enterprise Edition adapts to any network, security, or infrastructure context.

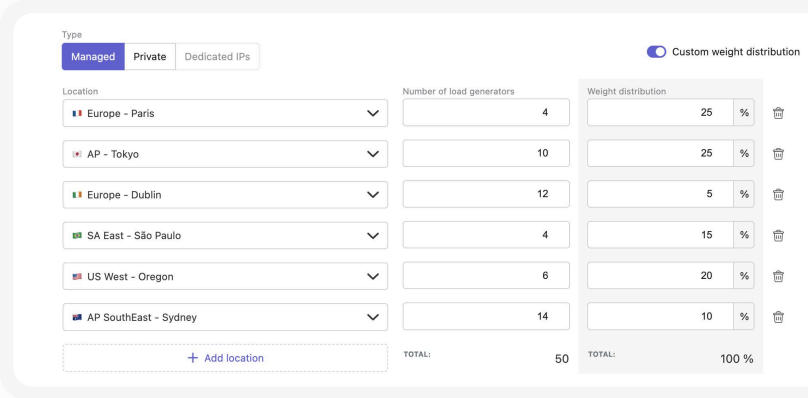
1. Gatling-managed load generators

What it is

Load generators hosted and operated by Gatling on AWS, pre-configured for optimal performance. Default instance type: c6i.xlarge (4 vCPU).

Capabilities

Up to 3 million virtual users with 50 concurrent load generators. Configurable up to 100–200 with validation to avoid triggering cloud provider security limits.



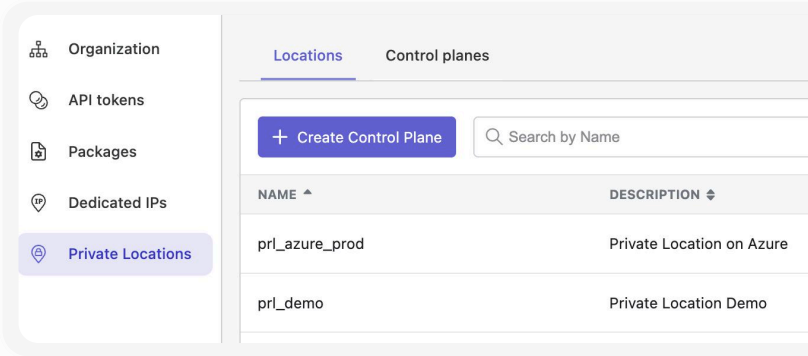
2. Self-managed load generators

What it is

Deploy and manage your own injectors in AWS, Azure, GCP, Kubernetes, OpenShift, or on-premises. Full control over sizing, scaling, and security.

Capabilities

Unlimited scaling, constrained only by your own infrastructure quotas and compute capacity.



> Whichever model you choose, test orchestration, reporting, data aggregation, and scenario execution remain **fully centralized within Gatling Enterprise Edition.**

Premium support for high-scale testing

When you choose Gatling Enterprise Edition, you also get enterprise-grade support designed for mission-critical load testing.

- **Architecture & topology recommendations**
- **Scenario design & performance strategy**
- **Debugging system bottlenecks**
- **High-scale configuration guidance**
- **Injector health monitoring & troubleshooting**
- **Best practices for JVM warmup, TLS optimization, scaling patterns**

> **You are never alone in your high-scale testing journey.**

How to choose your load generation configuration

Gatling Enterprise Edition adapts to any network and security context. Whether your services are publicly exposed, protected by firewalls, or fully isolated on-premises, **Gatling provides the right deployment model for your load generators.**

The following table summarizes which configuration fits best depending on your endpoint environment and testing requirements.

ENDPOINT INFRASTRUCTURE	TESTING REQUIREMENTS	RECOMMENDED SOLUTION	DESCRIPTION
Public Services (Internet-exposed)	Public endpoints without strict firewall rules or rate limits	Gatling-Managed Load Generators	Fully managed by Gatling in AWS
Public Services with Firewall / DDoS Protection	Client must allow Gatling traffic through security layers	Gatling-Managed Load Generators with Dedicated IP	Fully managed by Gatling in AWS, using fixed IPs for whitelisting
Cloud-Native or Microservices Environments	Need to simulate load close to the application to minimize latency	Private Locations (Cloud-Managed)	You can deploy in AWS, Azure, GCP, Kubernetes, or OpenShift, with full control over the entire environment.
Secure or Regulated Environments	Testing requires sensitive data, secret keys, or regulated handling of credentials	Private Locations (Cloud-Managed)	You can deploy in AWS, Azure, GCP, Kubernetes, or OpenShift, with full control over the entire environment.
Private Services (Behind Firewalls, No Direct Internet Access)	Application or API cannot be reached from public networks	Private Locations (Cloud-managed)	You can deploy in AWS, Azure, GCP, Kubernetes, or OpenShift, with full control over the entire environment.
No Access to Public Cloud Providers	On-premises environments, managed datacenters	Private Locations (Dedicated Machines)	You can deploy in AWS, Azure, GCP, Kubernetes, or OpenShift, with full control over the entire environment.

Gatling

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 ›