Load testing your apps with Gatling

Denys Kholod @dkholod

Tech Lead @SBTech


APACHE
kafka
A distributed streaming platform


SBTech
We Know Sports


RabbitMQ
Messaging that just works


ASP.NET Core


SignalR


CONSUL


TERRAFORM


ZIPKIN


JFrog Artifactory


Orleans


Gatling
LOAD TESTING


TypeScript


mongoDB

Google Cloud Platform


PostgreSQL


AEROSPIKE


elastic


kibana


STATSD


Grafana


React JS

SBTech

we know sports

- Biggest provider in sport offering
- Supports a lot of regulated markets
- About 1000 microservices (200 distinct types)
- About 5 datacenters maintained fully by SBTech
- About 500 concurrent live events at pick time
- On average we handle about 100K+ RPS

Load testing: why?

- Challenge your quality attributes
- Anticipate problems on production system
- Reproduce issues with “no repro” in dev
- Train to run CD/Monitoring

Load testing: what do you need?

- People
  - QA, BA, Devs, DBA, Ops

- Tools
  - Monitoring
  - Logging
  - Load Testing Tool

Gatling?

- JMeter
- VS/VSTS
- Load Runner
- …
- ONE MORE Load Test tool

Gatling!

- Simulate bunch of users, events, etc.
- Protocol level (http, websockets, etc.)
- Complex behavior, not just a recorder (ab, wrk)

Features

- Performance
- Usage
- Metrics
- Plugins
- Open Source and for Enterprise

Performance

1 virtual user = 1 thread

50 threads on JVM or .NET

Performance

1000 threads on JVM or .NET

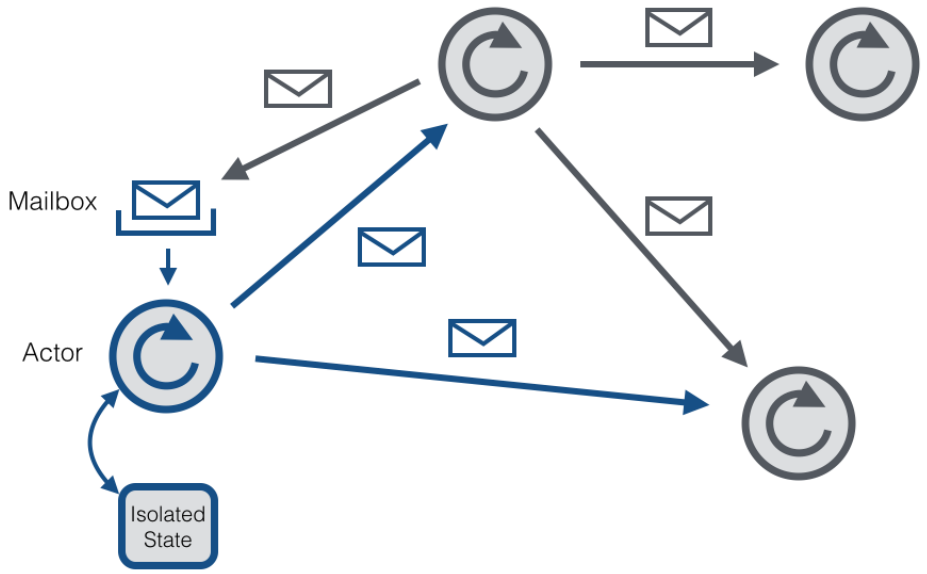
Performance

- Blocking IO

- Simulating user delays

Gatling Performance

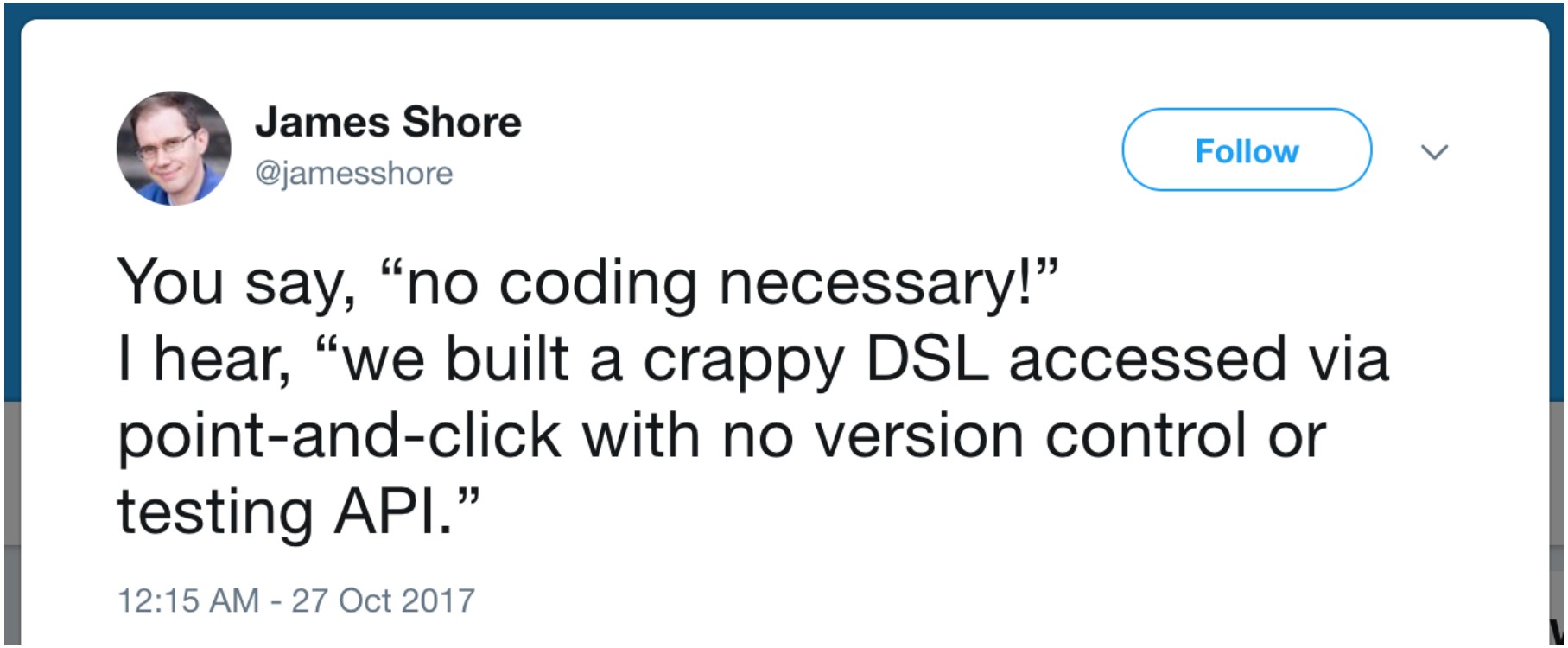
- Message oriented
- Netty



Usage

- Versioning
- Maintaining
- Refactoring
- Peer review
- Implementing scenarios

Usage


James Shore
@jamesshore
Follow
You say, “no coding necessary!”
I hear, “we built a crappy DSL accessed via point-and-click with no version control or testing API.”
12:15 AM - 27 Oct 2017

Usage: maintainability

DemoSimulation.scala

```scala
import io.gatling.core.Predef._
import io.gatling.http.Predef._

class DemoSimulation extends Simulation {

  val scn = scenario("Demo Scenario")
    .exec(http("request_1 (Home page)")
      .get("https://www.sbtech.com/")
      .check(status is 200))

    .pause(7)

    .exec(http("request_2 (About page)")
      .get("https://www.sbtech.com/about-sbtech/")
      .check(status is 200))

  setUp(scn.inject(atOnceUsers(1)))
}
```

Usage: simple DSL

- Describe your users behavior
- Tune your simulation
- Inject data in your scenario
- Control how users are injected in your scenario
- Check that your results match your expectations
- Define the HTTP requests sent in your scenario
- Verifying server responses

Metrics

- Bad (reverse compute):
  - Derived from Mean + StdDev
  - Sliding Window Sampling
  - Averaging

- Good
  - HdrHistogram/HdrHistogram.NET
  - Tdigest/T-Digest.NET

Plugins

- Official
  - SBT plugin/Maven/Gradle
  - Jenkins/TeamCity/Bamboo
  - WebSockets, SSE, polling

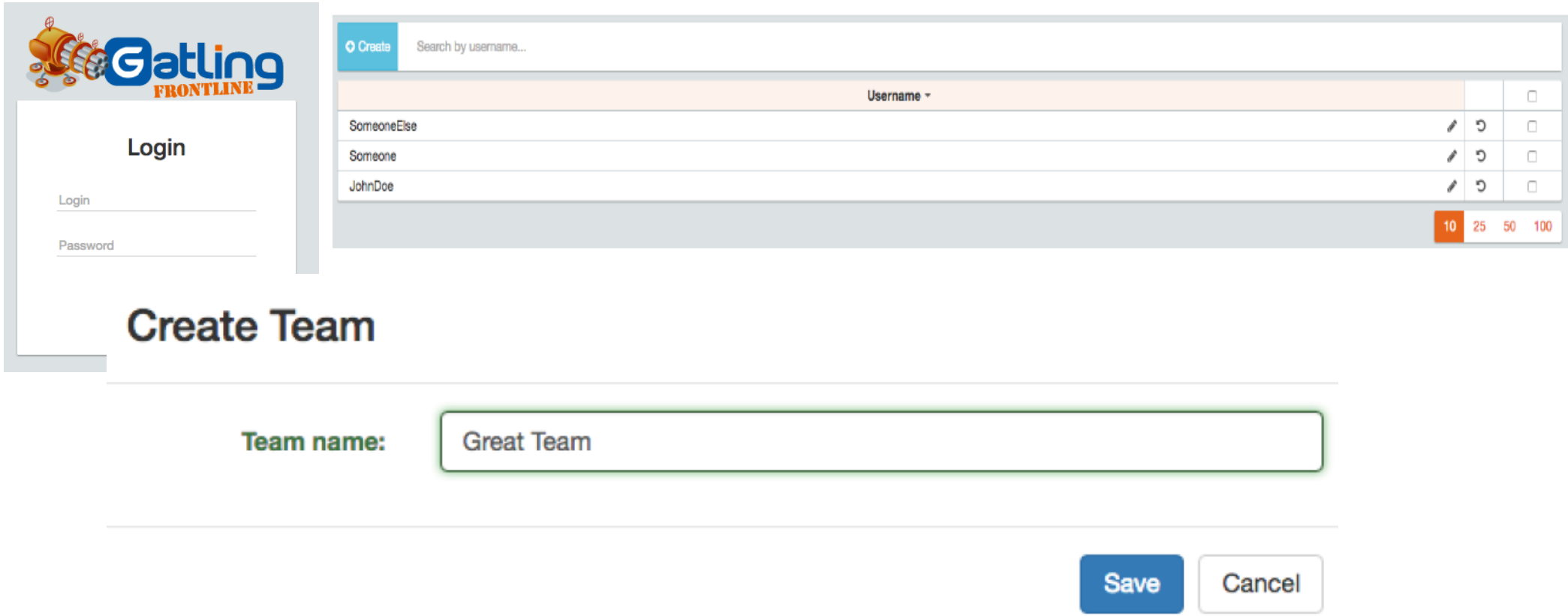
- Third parties
  - Cassandra
  - Kafka
  - AMQP
  - SQL
  - …

Open Source

- Gatling is Open Source since 2011
- Work with single scenarios in Sublime, VS Code, etc.
- Create SBT project in IntelliJ IDEA, Eclipse, etc.
- Inject remote nodes and collect results via script

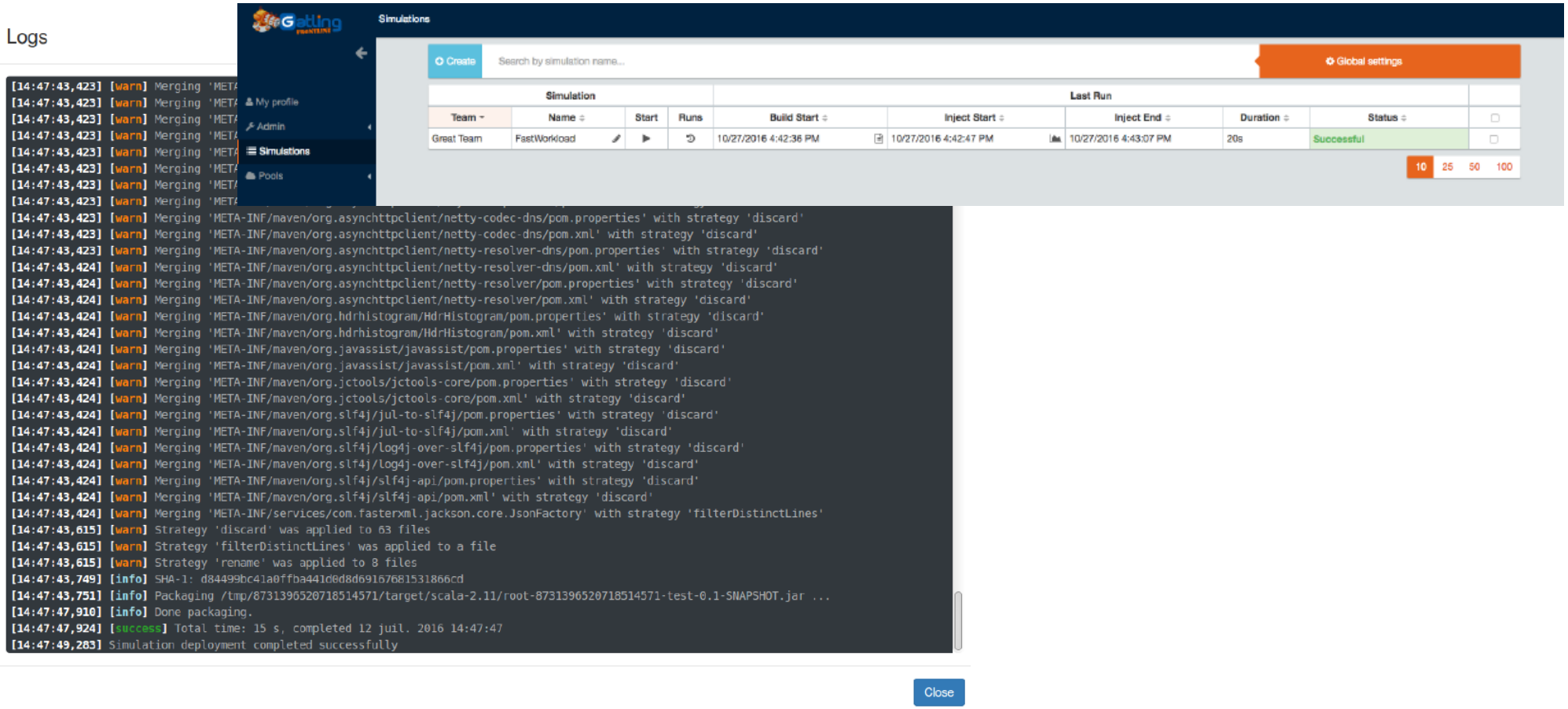
FrontLine

- Users, permissions management



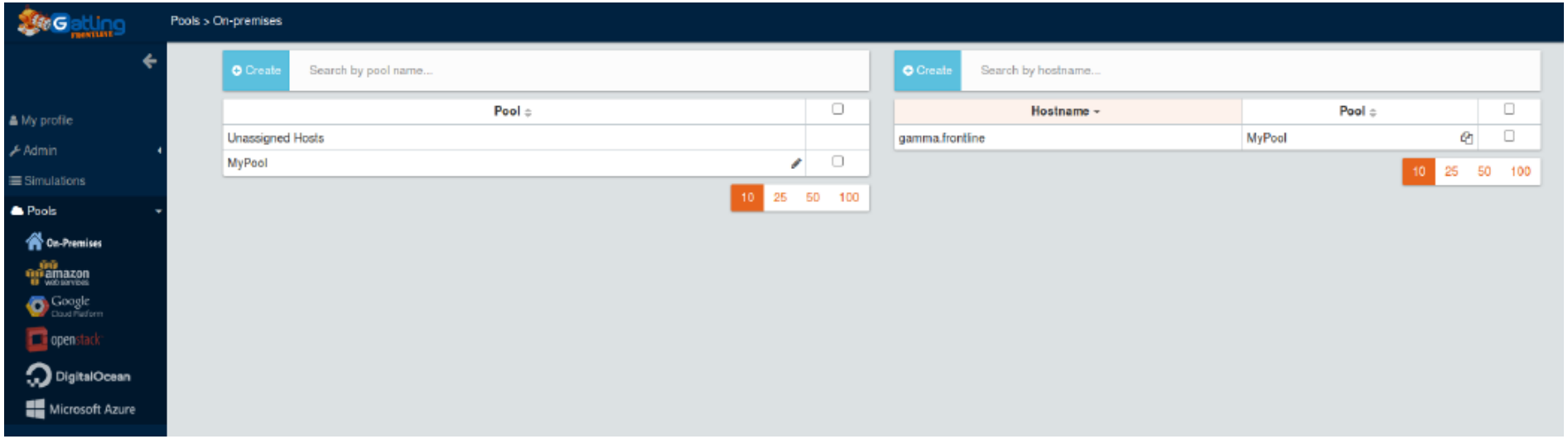
FrontLine

- Simulations



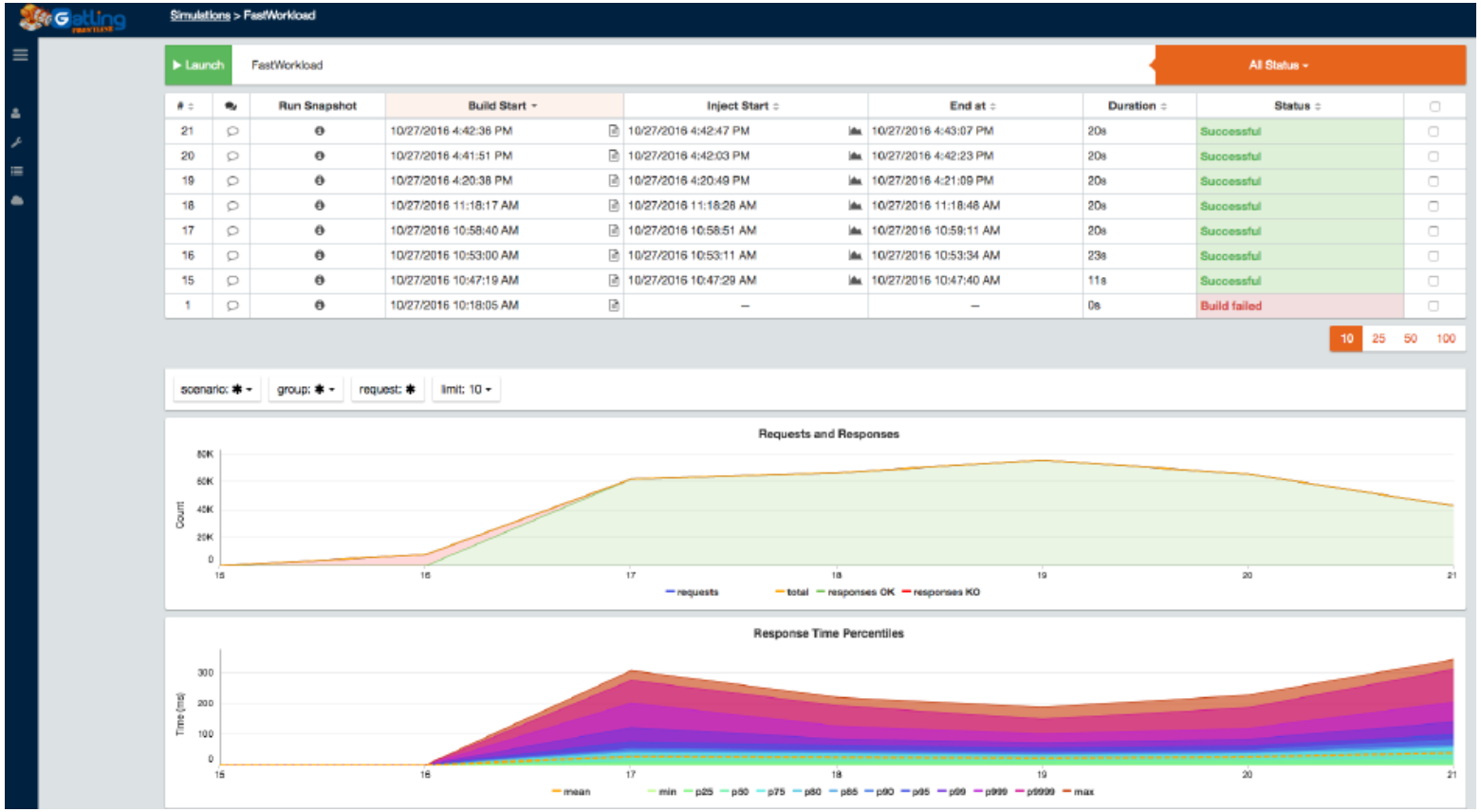
FrontLine

- Pools
  - AWS
  - GCE
  - OpenStack
  - digitalOcean
  - Microsoft Azure



FrontLine

- Runs and Trends



FrontLine

- Live metrics Graphite-InfluxDB-Grafana

Demo

Q&A

@dkholod