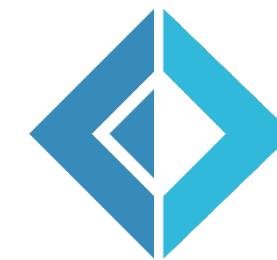




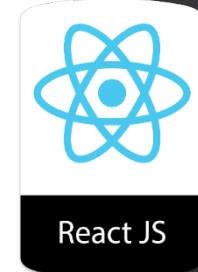
# Load testing your apps with **Gatling**

Denys Kholod @dkholod

Tech Lead @SBTech



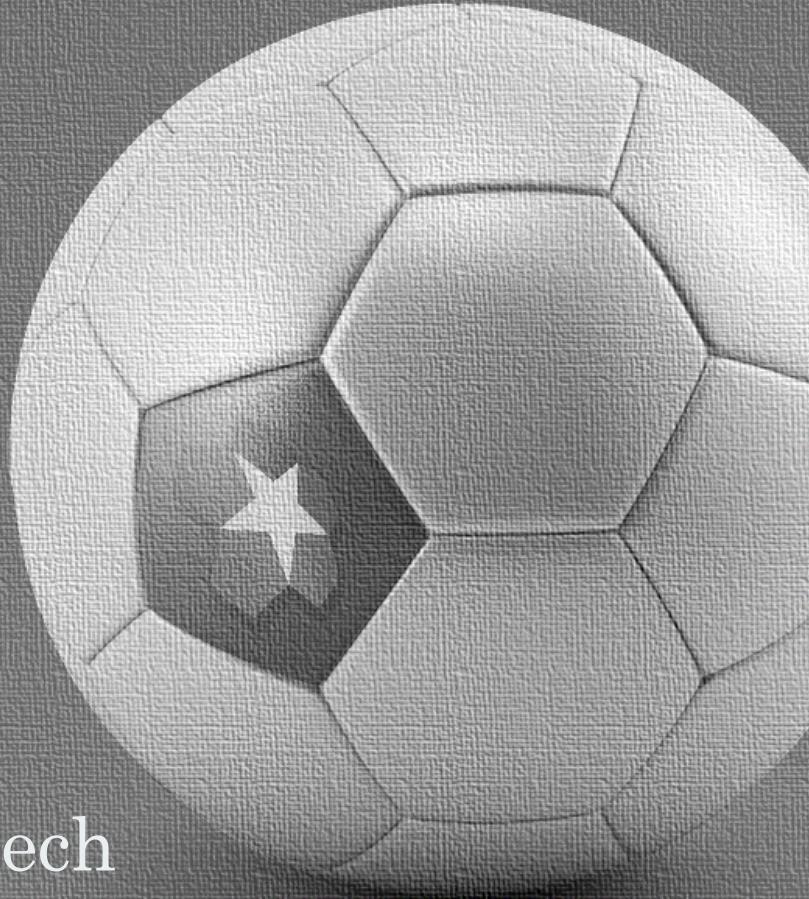
AEROSPIKE





**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

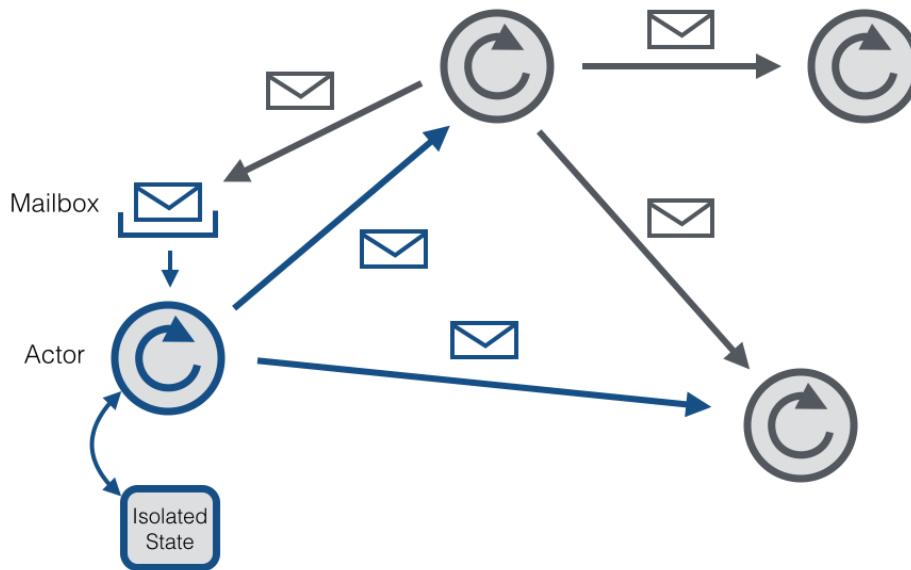
- Blocking IO
- Simulating user delays



# Gatling Performance



- Message oriented
- Netty



# Usage

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



# Usage



**James Shore**  
@jamesshore

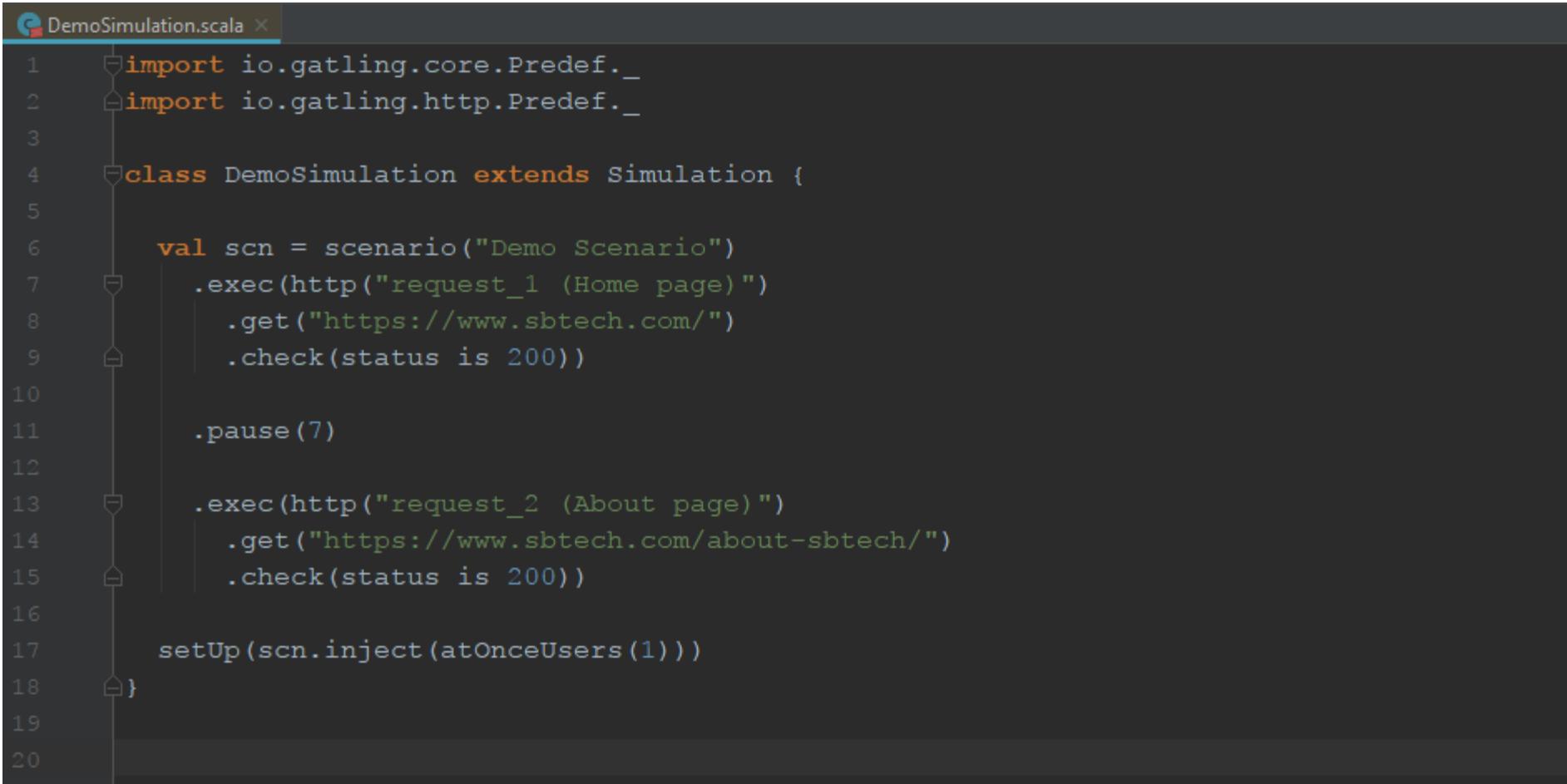
[Follow](#)



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



A screenshot of a code editor window titled "DemoSimulation.scala". The code is written in Scala and defines a simulation scenario for testing a web application. The code uses the Gatling library to define two requests: "request\_1" (Home page) and "request\_2" (About page). It includes setup instructions for injecting users.

```
1 import io.gatling.core.Predef._  
2 import io.gatling.http.Predef._  
3  
4 class DemoSimulation extends Simulation {  
5  
6   val scn = scenario("Demo Scenario")  
7     .exec(http("request_1 (Home page)")  
8       .get("https://www.sbtech.com/")  
9       .check(status is 200))  
10  
11   .pause(7)  
12  
13   .exec(http("request_2 (About page)")  
14     .get("https://www.sbtech.com/about-sbtech/")  
15     .check(status is 200))  
16  
17   setUp(scn.inject(atOnceUsers(1)))  
18 }  
19  
20
```

# 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

The screenshot shows the FrontLine application interface. On the left, there is a sidebar with a "Login" form containing fields for "Login" and "Password". The main area has a header with a "Create" button and a search bar labeled "Search by username...". Below this is a table titled "Username" with three rows: "SomeoneElse", "Someone", and "JohnDoe". Each row has a checkbox column and a delete icon. At the bottom of the table are buttons for "10", "25", "50", and "100". In the center, there is a "Create Team" section. It includes a "Team name:" label followed by a text input field containing "Great Team", which is highlighted with a green border. At the bottom right are "Save" and "Cancel" buttons.

Username		
SomeoneElse	<input type="checkbox"/>	<span>✗</span>
Someone	<input type="checkbox"/>	<span>✗</span>
JohnDoe	<input type="checkbox"/>	<span>✗</span>

10 25 50 100

Team name: Great Team

Save Cancel

# FrontLine



- Simulations

Logs

Simulations

Global settings

10 25 50 100

Team	Name	Start	Runs	Last Run	Duration	Status		
Great Team	FastWorkload	▶	1	10/27/2016 4:42:36 PM	10/27/2016 4:42:47 PM	10/27/2016 4:43:07 PM	20s	Successful

[14:47:43,423] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-codec-dns/pom.properties' with strategy 'discard'  
[14:47:43,423] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-codec-dns/pom.xml' with strategy 'discard'  
[14:47:43,423] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-resolver-dns/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-resolver-dns/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-resolver/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.asynchttpclient/netty-resolver/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.hdrhistogram/HdrHistogram/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.hdrhistogram/HdrHistogram/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.javassist/javassist/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.javassist/javassist/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.jctools/jctools-core/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.jctools/jctools-core/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/jul-to-slf4j/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/jul-to-slf4j/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/log4j-over-slf4j/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/log4j-over-slf4j/pom.xml' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/slf4j-api/pom.properties' with strategy 'discard'  
[14:47:43,424] [warn] Merging 'META-INF/maven/org.slf4j/slf4j-api/pom.xml' with strategy 'discard'  
[14:47:43,615] [warn] Strategy 'discard' was applied to 63 files  
[14:47:43,615] [warn] Strategy 'filterDistinctLines' was applied to a file  
[14:47:43,615] [warn] Strategy 'rename' was applied to 8 files  
[14:47:43,749] [info] SHA-1: d8449bc41a0ffba441d0d8d69167681531866cd  
[14:47:43,751] [info] Packaging /tmp/8731396520718514571/target/scala-2.11/root-8731396520718514571-test-0.1-SNAPSHOT.jar ...  
[14:47:47,910] [info] Done packaging.  
[14:47:47,924] [success] Total time: 15 s, completed 12 juil. 2016 14:47:47  
[14:47:49,283] Simulation deployment completed successfully

Close

# FrontLine



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

Screenshot of the Gatling FRONTLINE web interface showing the 'Pools > On-premises' page.

The left sidebar shows navigation links: My profile, Admin, Simulations, Pools (selected), On-Premises (selected), Amazon Web Services, Google Cloud Platform, openstack, DigitalOcean, and Microsoft Azure.

The main content area displays two tables:

Pool	Hostnames	Pools
Unassigned Hosts		
MyPool		

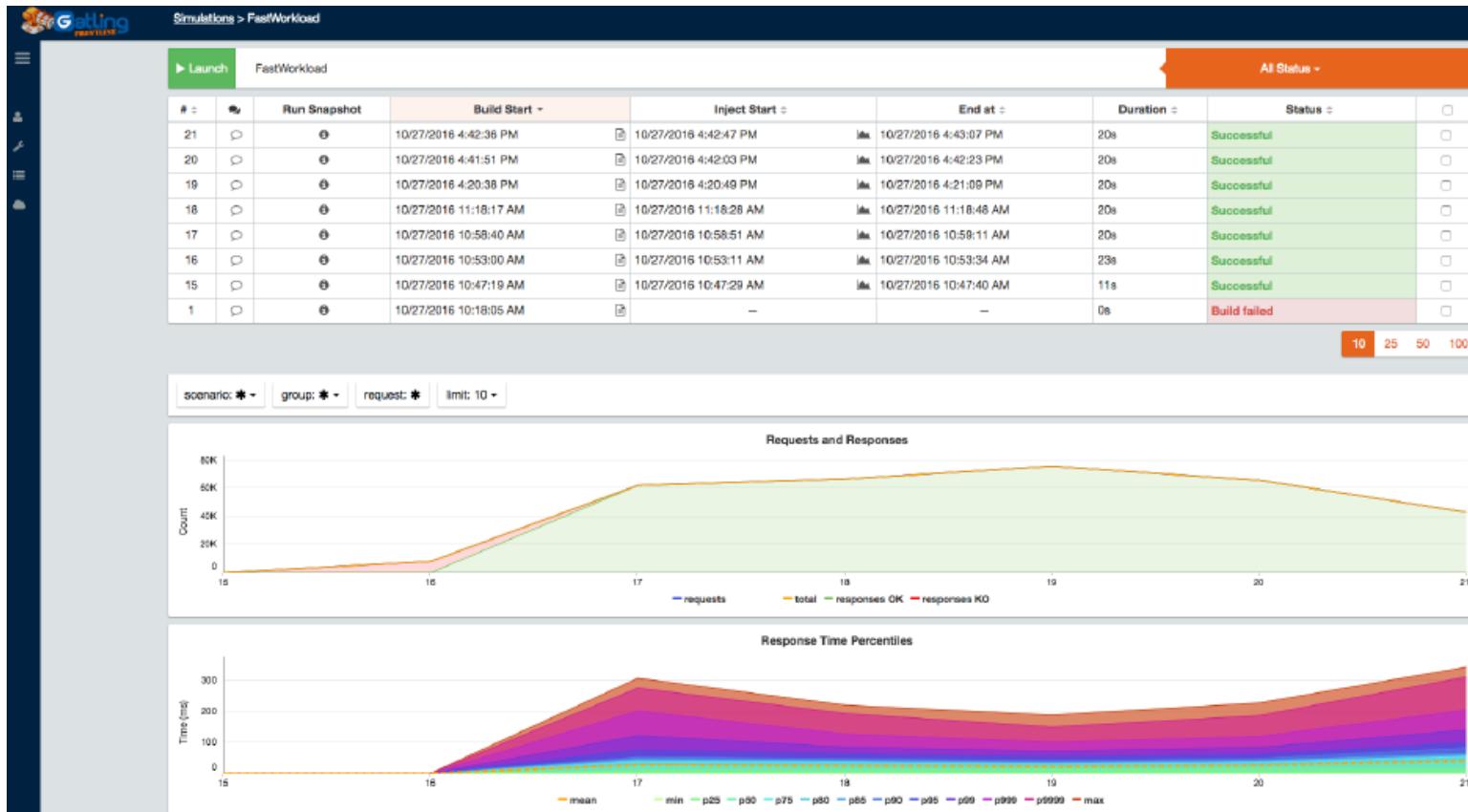
Hostname	Pool
gamma.frontline	MyPool

Below the tables are pagination controls: 10, 25, 50, 100.

# FrontLine



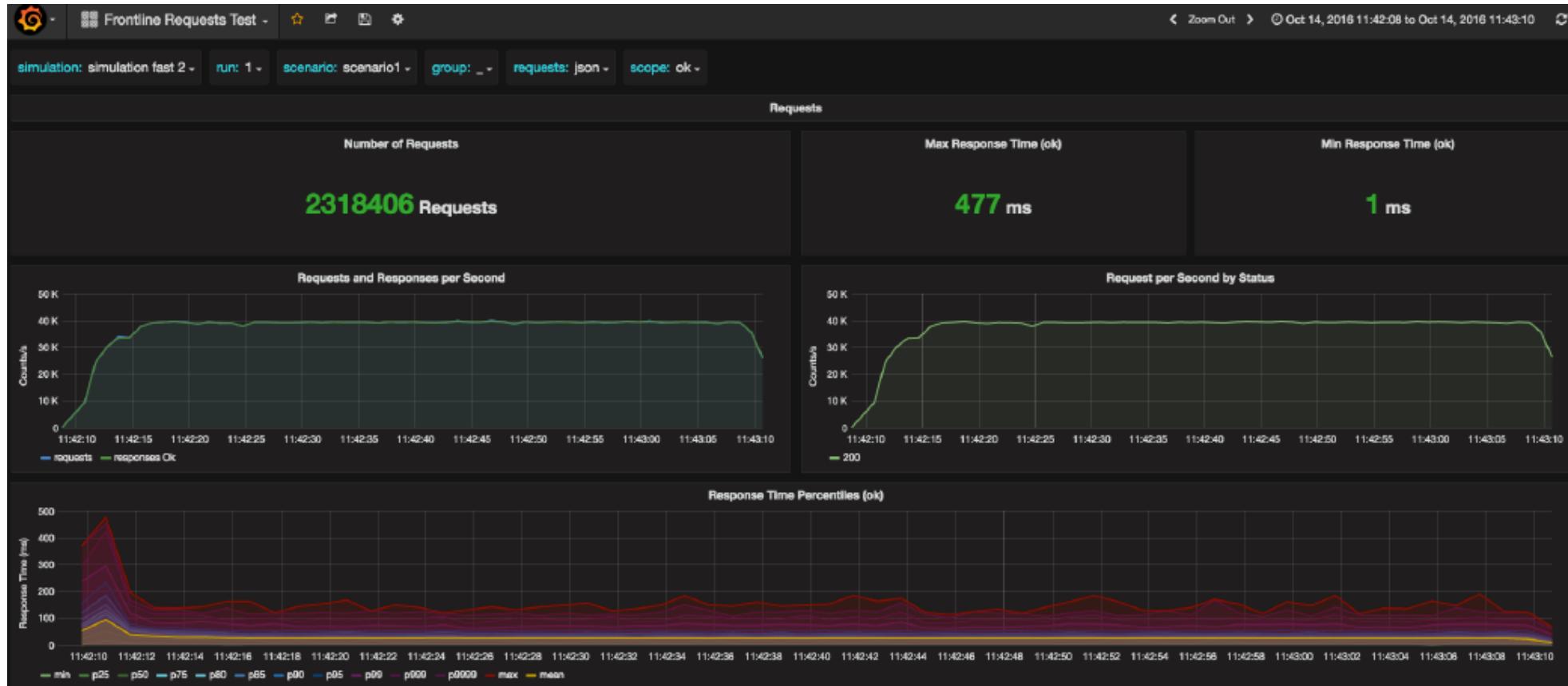
- Runs and Trends



# FrontLine



- Live metrics Graphite-InfluxDB-Grafana



# Demo



# Q&A



@dkholod