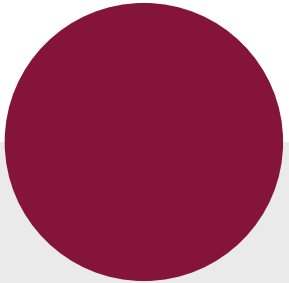
.msg .consulting .solutions .partnership



Start your Engines
White box Monitoring for your Load Tests

Alexander Schwartz, Principal IT Consultant

PromCon Munich 17 August 2017

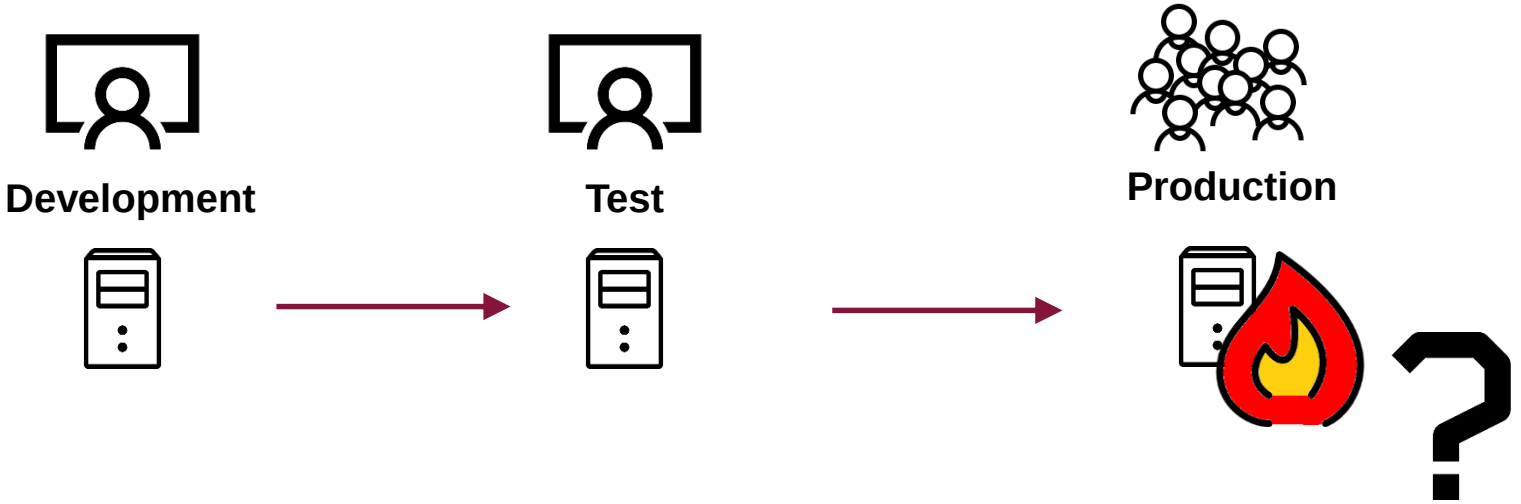
Start your Engines – Whitebox Monitoring your Load Tests

1. Why to use load tests
2. Setup of the test environment
3. Demo
4. What to expect

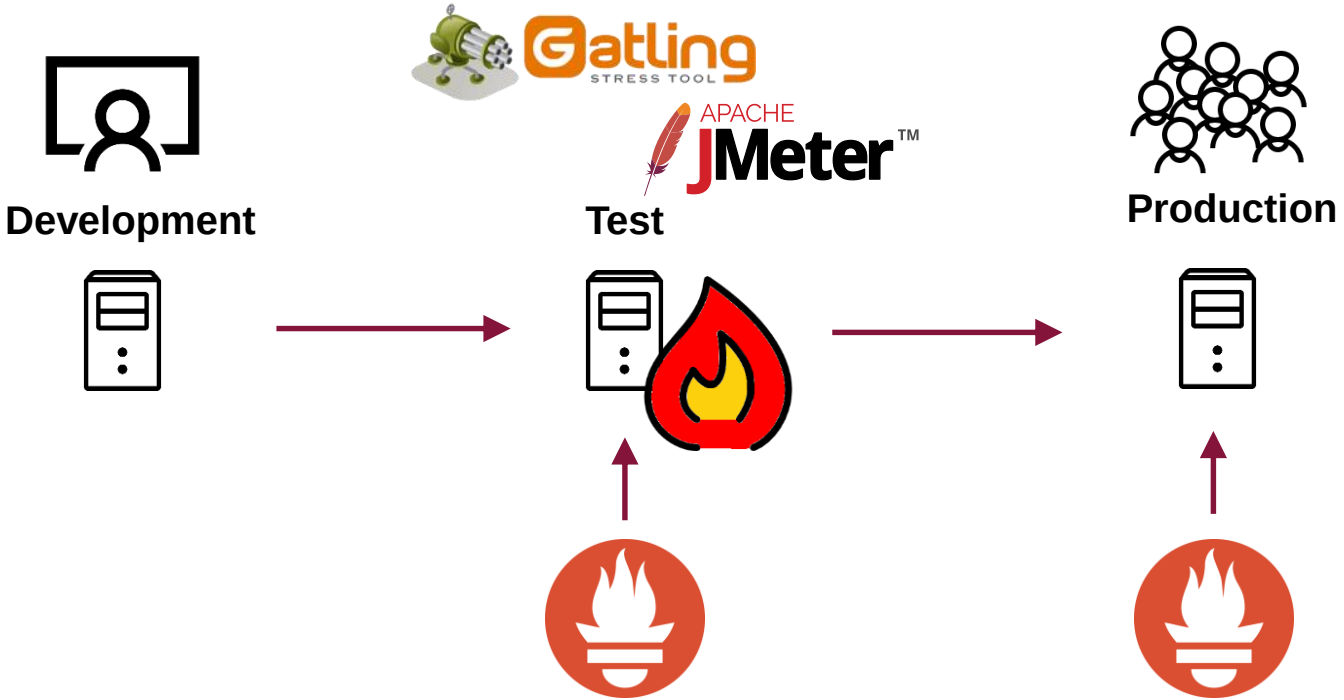
Start your Engines – Whitebox Monitoring your Load Tests

1. **Why to use load tests**
2. Setup of the test environment
3. Demo
4. What to expect

Naive Approach



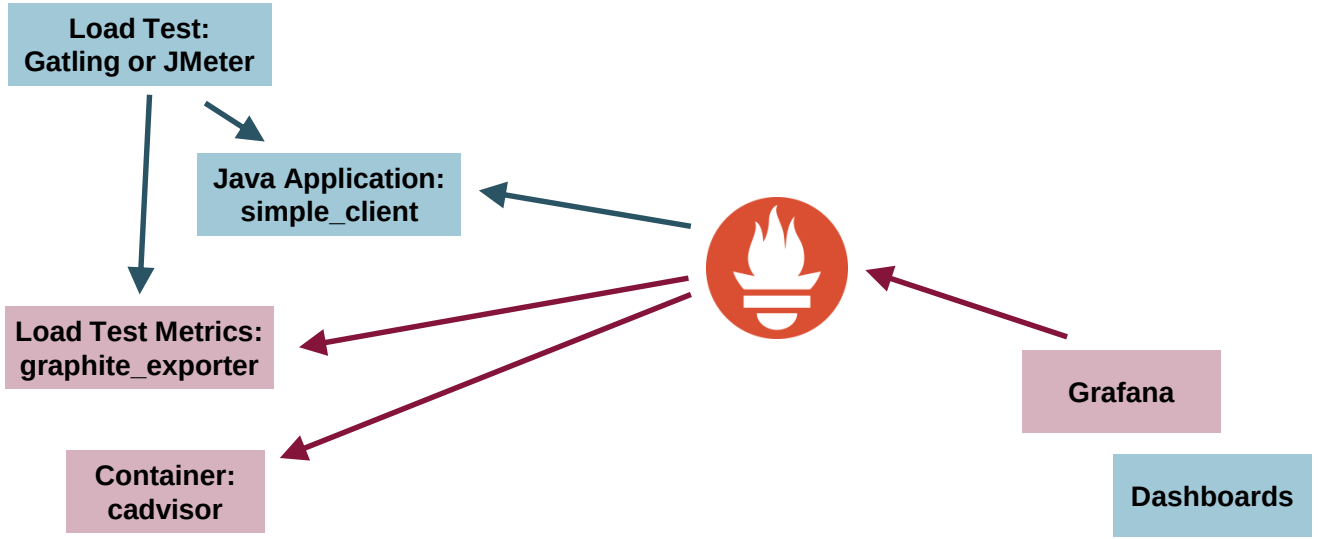
Alternative Approach with Load Testing and Monitoring



Start your Engines – Whitebox Monitoring your Load Tests

1. Why to use load tests
2. **Setup of the test environment**
3. Demo
4. What to expect

Technical Building Blocks



Purple (including Prometheus): Provided as infrastructure in a testing environment
Blue: Setup and maintained by product team (developers/testers)

Gatling vs. JMeter

- JMeter support GUI-based point-and-click configuration of load tests
  (but at some point you need to start scripting)
- Gatling requires you to script your tests in a Scala DSL
  (that is powerful but is sometimes mind twisting)
- JMeter stores the tests as XML blobs, where Gatling stores them as Scala code
- JMeter uses threads to drive the load, while Gatling uses a non-blocking event loop based on Akka

Why Graphite and Graphite Exporter?

Pro:

- JMeter and Gatling support Graphite out of the box
- For exploratory tests you can run JMeter and Gatling a local machine
  (and it will send the metrics to the Graphite Exporter)
- Graphite Exporter forgets metrics after five minutes
  (no need to clear metrics)
- Metrics exported from Gatling per step and overall:
  users waiting/active/done, aggregated request timings with percentiles, ok/failure counts

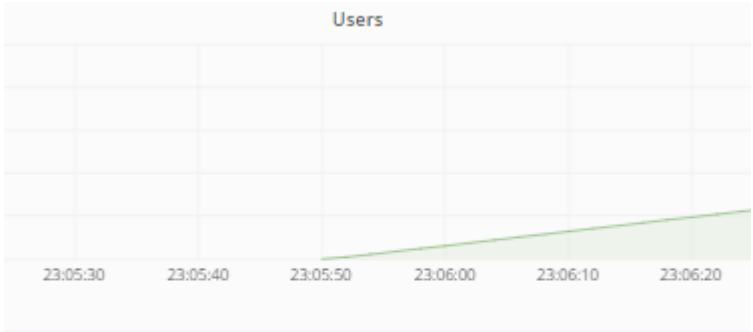
Contra:

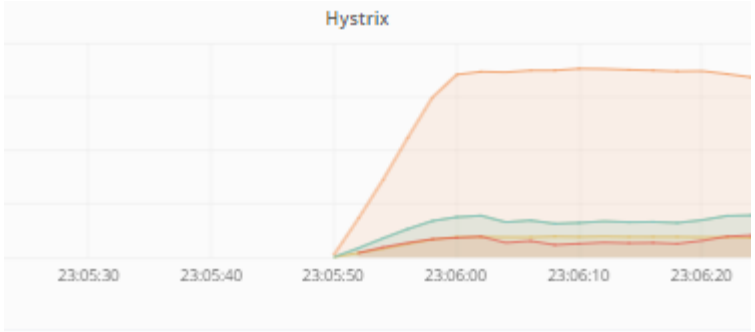
- Graphite metrics don’t use labels unless you configure a conversion in the Graphite Exporter
  (example: “gatling_mysimulation_users_allusers_done”)

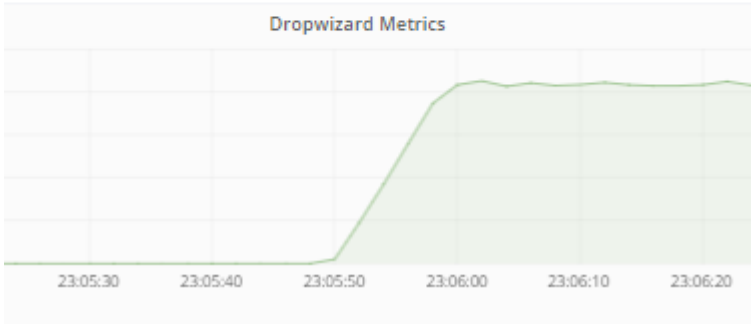
Start your Engines – Whitebox Monitoring your Load Tests

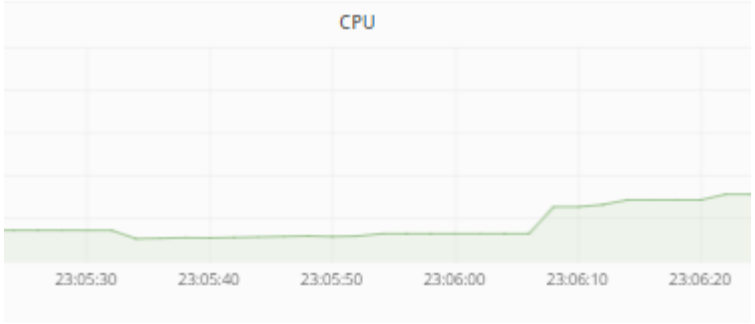
1. Why to use load tests
2. Setup of the test environment
3. **Demo**
4. What to expect

Demo


Users
23:05:30
23:05:40
23:05:50
23:06:00
23:06:10
23:06:20


Hystrix
23:05:30
23:05:40
23:05:50
23:06:00
23:06:10
23:06:20


Dropwizard Metrics
23:05:30
23:05:40
23:05:50
23:06:00
23:06:10
23:06:20


CPU
23:05:30
23:05:40
23:05:50
23:06:00
23:06:10
23:06:20

Start your Engines – Whitebox Monitoring your Load Tests

1. Why to use load tests
2. Setup of the test environment
3. Demo
4. **What to expect**

Metrics the teams used in their Dashboards to monitor tests

From infrastructure:

- CPU usage per container
- RAM usage per container

From application:

- Standard runtime metrics (Java in our case)
- Counters
- Gauges [gājes] for Queues and Pools
- Collector for Dropwizard Metrics library for timing of critical business functionality
- Timings for dependent services from Neflix' circuit breaker library Hystrix

Lessons learned

The approach worked well for us to pass the load tests:

- Load Tool metrics correlated with application and infrastructure metrics
- Inter-application communication captured by Hystrix
- Self-service functionality for product teams to add applications and metrics

… but to use the instrumentation also in production create awareness:

- Exported metrics should following Prometheus naming conventions
- Collector for Dropwizard Metrics can’t fill HELP text of metrics
- Counters and averages vs. histograms, summaries and percentiles
- Consistent use of USE Method (utilization – saturation – errors)
  or RED Method (rate – errors – duration) for metrics

1. http://www.brendangregg.com/usemethod.html
2. https://www.weave.works/blog/prometheus-and-kubernetes-monitoring-your-applications/

Links

Prometheus:
https://prometheus.io

Java Simple Client
https://github.com/prometheus/client_java

Dropwizard Metrics
http://metrics.dropwizard.io

Prometheus Hystrix Metrics Publisher
https://github.com/ahus1/prometheus-hystrix

Apache JMeter
http://jmeter.apache.org/

Gatling
http://gatling.io/

CAdvisor
https://github.com/google/cadvisor

Graphite Exporter
https://github.com/prometheus/graphite_exporter

@ahus1de

Alexander Schwartz
Principal IT Consultant

+49 171 5625767
alexander.schwartz@msg-systems.com

@ahus1de

msg systems ag (Headquarters)
Robert-Buerkle-Str. 1, 85737 Ismaning
Germany

www.msg-systems.com

.msg .consulting .solutions .partnership