



**Siegfried Goeschl**

**Gatling**

**Tales From A Journey**



*Eine Innovation von*




**ERSTE**  **BANK**

**SPARKASSE** 






**Jetzt  
wechseln:  
mygeorge.at**

 **GIROKONTO Familienkonto**  
€ 3.280,<sup>25</sup>

-  Historie
-  **Neue Überweisung**
-  Daueraufträge
-  Favoriten 34
-  Rundungssparen
-  Kontoeinstellungen

## Überweisung

ABBRECHEN 

An	<input type="text" value="Name, IBAN oder Kontonummer des Empfängers"/>  
Betrag	<input type="text" value="0,00"/> € ▾
Verwendungszweck	<input type="text"/>
Zahlungsreferenz	<input type="text"/>
Auftraggeber-Referenz	<input type="text"/>
Empfang bis ▾	<input type="text" value="TT.MM.JJJJ"/> 

Speichern & neue Überweisung

Diese Überweisung freigeben



# George International

- Turning George Online Banking into a multi-tenant and group-wide platform
  - ▶ Single code base
  - ▶ Currently targeting four tenants
- Server-side George consists of two parts
  - ▶ George API Server
  - ▶ Core Banking Interface



# George International

- George API Server is a fancy transaction search engine based on Elastic Search
- Core banking interface consists of a set REST endpoints provided by each tenant
- Multiple geographical distributed development teams working together





# Continuous Delivery





Wrong End of  
Continuous  
Delivery?!



# Being At The Wrong End

- Automatic deployment in the early hours
- REST endpoints broken in the morning
- Front-end development tasks delayed
- Complaining became a form of art

Janos Horvath 9:38 AM  
uploaded an image: [Pasted image at 2016-10-06, 9:38 AM](#)



Attila 9:48 AM  
<http://sd.keepcalm-o-matic.co.uk/i/keep-calm-and-beam-me-up>





What about having a REST client which tests the various REST endpoints every morning to see if they work?!







# Functional versus Performance Test Tools



# Which Tool To Choose

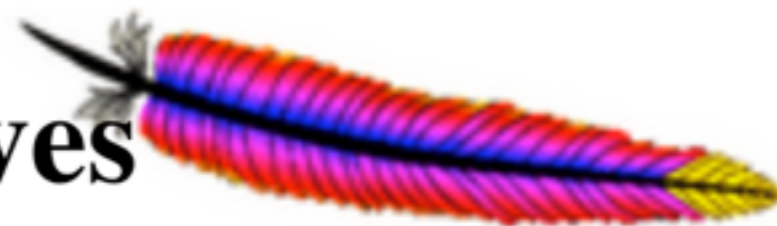
- Need to test various REST endpoints
- Various functional testing tools in use
  - ▶ SoapUI, Selenium, Tosca
- Performance testing got little love
  - ▶ Existing JMeter test suite outdated
  - ▶ Having a working performance test suite would be a huge bonus



<http://gatling.io>



# lucene-solr-user mailing list archives



[Site index](#) · [List index](#)

## Message view

« [Date](#) » · « [Thread](#) »

**From** Walter Underwood <wun...@wunderwood.org>

**Subject** Re: Measuring QPS

**Date** Mon, 06 Apr 2015 22:14:45 GMT

That sounds neat. Our QA people are moving to Gatling, so we probably won't change our JMeter approach now.

We use the JMeter Plugs CMDrunner, telling it to generate only CSV.

<http://jmeter-plugins.org/wiki/JMeterPluginsCMD/>

Walter Underwood

wunder@wunderwood.org

<http://observer.wunderwood.org/> (my blog)

On Apr 6, 2015, at 3:02 PM, Siegfried Goeschl <sgoeschl@gmx.at>

# The Case For Gatling

- Gatling scripts are written in Scala
- Scala DSL works nicely with your IDE
  - ▶ Auto-completion
  - ▶ Refactoring
  - ▶ Version control
  - ▶ Debugging



**Software is like  
sex, it's better  
when it's free.**

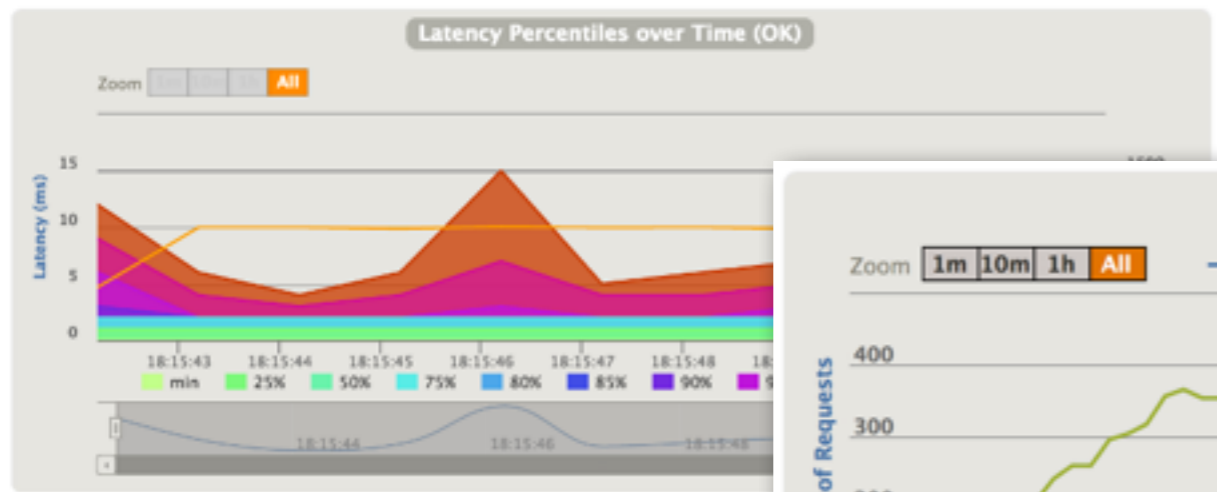
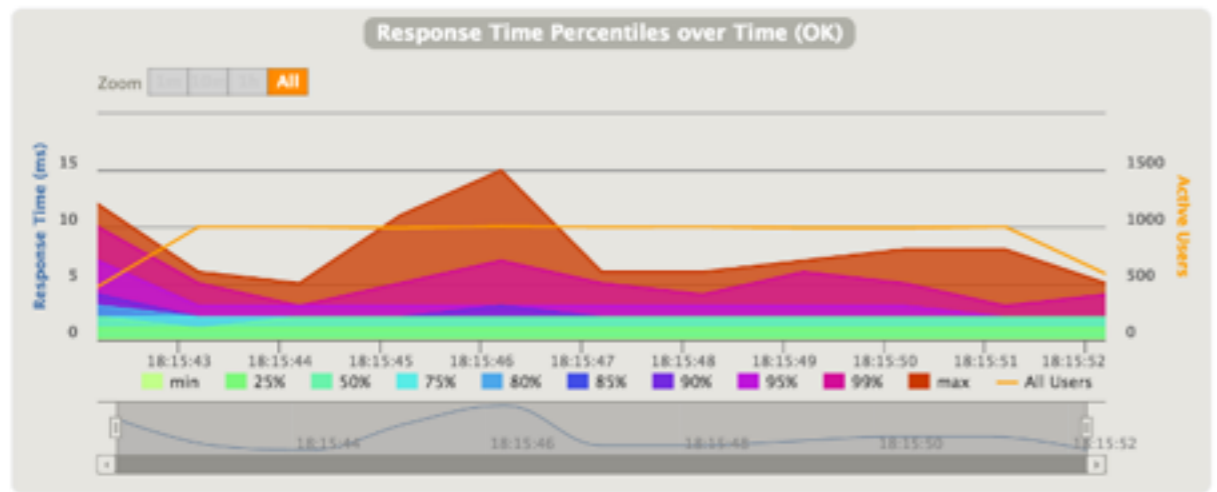
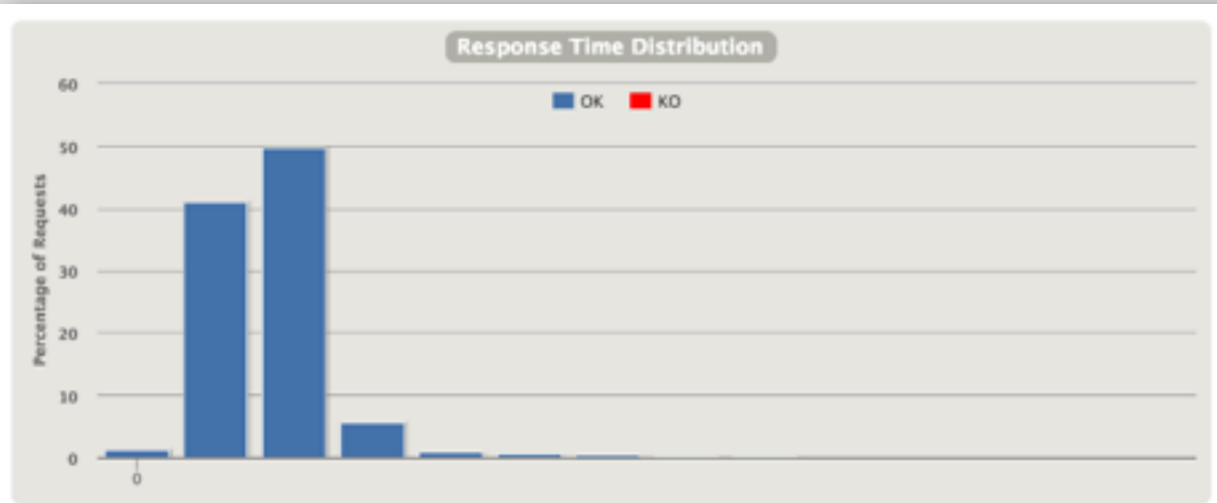
Linus Torvalds



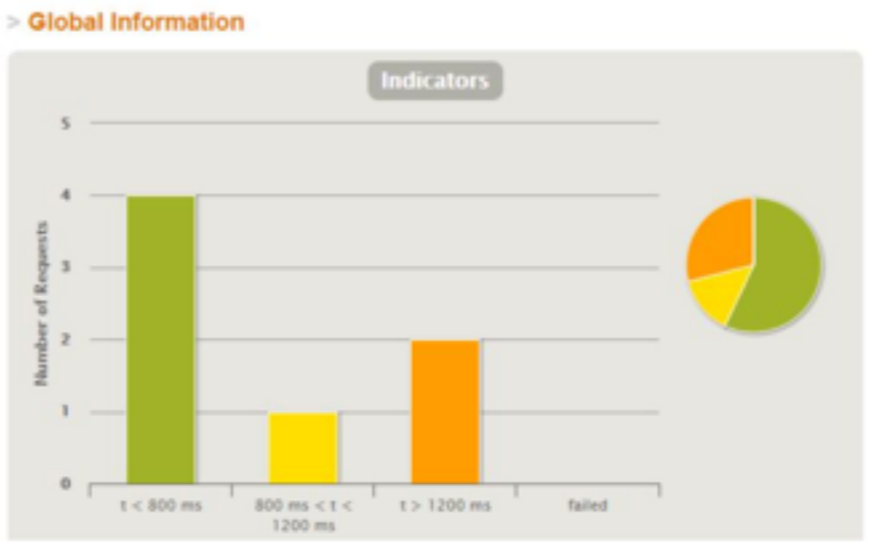
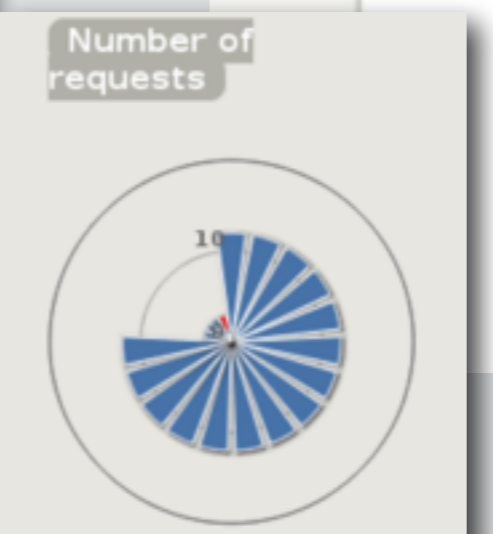
# Under The Hood

- Supports HTTP & JMS protocol
- Response validation
  - ▶ Regular expressions
  - ▶ XPath & JSONPath
  - ▶ CSS selectors
- Management-friendly HTML test reports





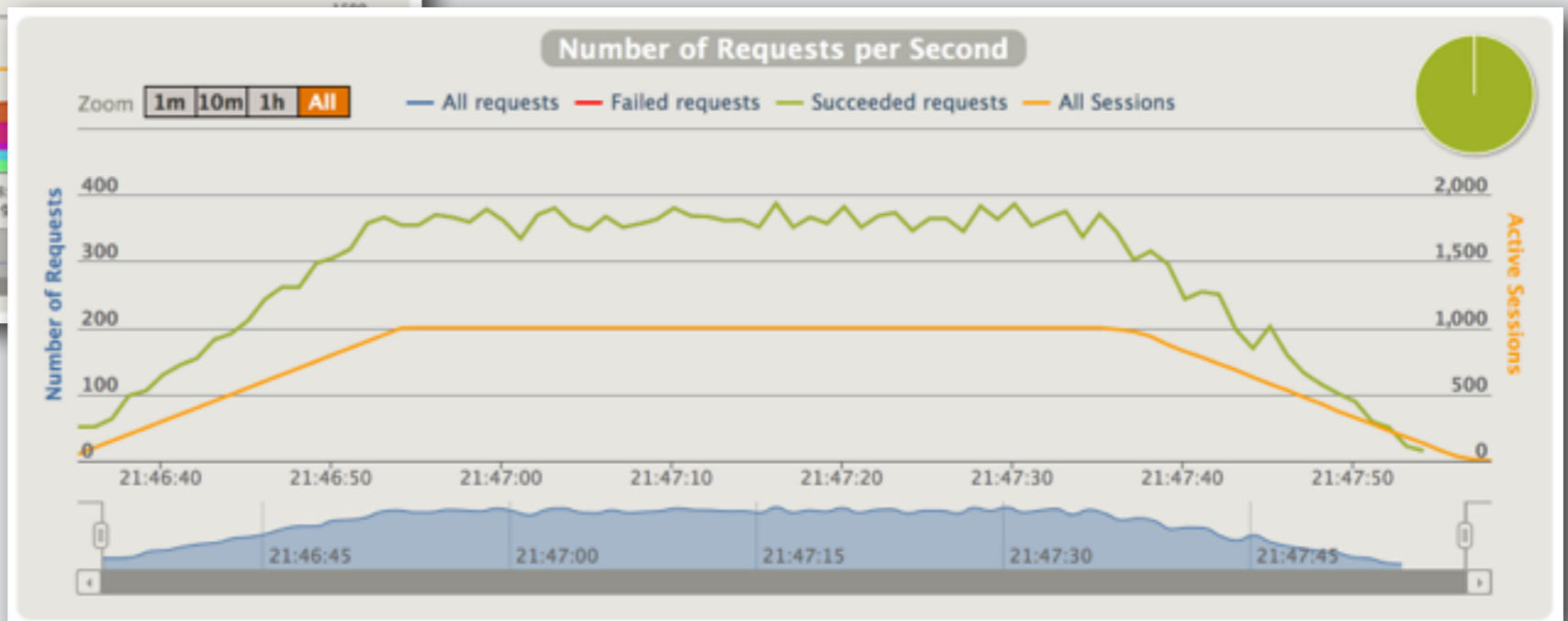
Active Users  
Requests / sec  
Responses / sec



### STATISTICS

Expand all groups | Collapse all groups

Requests ^	Executions			Response Time (ms)						
	Total †	OK †	KO †	Min †	Max †	Mean †	Std Dev †	95th pct †	99th pct †	Req/s †
Global Information	22	22	0	0	30	6	9	30	30	4
request_1	1	1	0	30	30	30	0	30	30	0
request_2	1	1	0	30	30	30	0	30	30	0
request_3	1	1	0	0	0	0	0	0	0	0
request_4	1	1	0	0	0	0	0	0	0	0
request_5	1	1	0	0	0	0	0	0	0	0



# Under The Hood

- Gatling Recorder to capture user requests
- Based on Akka & Netty libraries
  - ▶ Non-blocking, asynchronous requests
  - ▶ One thread for many virtual users



Plans are only good intentions  
unless they immediately  
degenerate into hard work.

*Peter Drucker*

# The First Steps

- Proof of concept done as side project
- Gatling tests executed every morning
- When the Gatling test failed
  - ▶ Contact backend developers
  - ▶ Inform George FE team



# The First Steps

```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
      .exec(  
        Token.get,  
        GeorgeConfiguration.get,  
        GeorgePreferences.get,  
        GeorgeAccounts.get,  
        GeorgeTransactions.get  
      )  
    }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```

```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
        .exec(  
          Token.get,  
          GeorgeConfiguration.get,  
          GeorgePreferences.get,  
          GeorgeAccounts.get,  
          GeorgeTransactions.get  
        )  
    }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```



Iterate over CSV users



```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
        .exec(  
          Token.get,  
          GeorgeConfiguration.get,  
          GeorgePreferences.get,  
          GeorgeAccounts.get,  
          GeorgeTransactions.get  
        )  
    }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```



Steps to execute

```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
        .exec(  
          Token.get,  
          GeorgeConfiguration.get,  
          GeorgePreferences.get,  
          GeorgeAccounts.get,  
          GeorgeTransactions.get  
        )  
    }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```



Run with one user

```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
        .exec(  
          Token.get,  
          GeorgeConfiguration.get,  
          GeorgePreferences.get,  
          GeorgeAccounts.get,  
          GeorgeTransactions.get  
        )  
    }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```



Fail on errors



```
class Test extends ConfigurableSimulation {  
  
  val users = scenario("Users")  
    .repeat(userCsvFeeder.records.length) {  
      feed(userCsvFeeder)  
        .exec(  
          Token.get,  
          GeorgeConfiguration.get,  
          GeorgePreferences.get,  
          GeorgeAccounts.get,  
          GeorgeTransactions.get  
        )  
      }  
  
  setUp(users.inject(atOnceUsers(1)))  
    .protocols(httpConfServer)  
    .assertions(global.failedRequests.count.is(0))  
}
```



Fetch user accounts

# The First Steps

```
object GeorgeAccounts {  
  
  val get = exec(http("my/accounts")  
    .get(ConfigurationTool.getUrl("georgeapi", "/my/accounts"))  
    .header("Authorization", "bearer ${token}")  
    .check(  
      jsonPath("$.collection[*].id")  
        .ofType[String].findAll.optional.saveAs("accountIds")))  
}
```

## Determine URL

```
object GeorgeAccounts {  
  
  val get = exec(http("my/accounts")  
    .get(ConfigurationTool.getUrl("georgeapi", "/my/accounts"))  
    .header("Authorization", "bearer ${token}")  
    .check(  
      jsonPath("$.collection[*].id")  
        .ofType[String].findAll.optional.saveAs("accountIds"))  
  )  
}
```



## Set OAuth Token

```
object GeorgeAccounts {  
  
    val get = exec(http("my/accounts")  
        .get(ConfigurationTool.getUrl("georgeapi", "/my/accounts"))  
        .header("Authorization", "bearer ${token}")  
        .check(  
            jsonPath("$.collection[*].id")  
                .ofType[String].findAll.optional.saveAs("accountIds"))  
    )  
}
```

```
object GeorgeAccounts {
```

```
  val get = exec(http("my/accounts")  
    .get(ConfigurationTool.getUrl("georgeapi", "/my/accounts"))  
    .header("Authorization", "bearer ${token}")  
    .check(  
      jsonPath("$.collection[*].id")  
        .ofType[String].findAll.optional.saveAs("accountIds"))  
  )  
}
```

Save account ids

[Back to Dashboard](#)[Status](#)[Changes](#)[Workspace](#)[Build Now](#)[Delete Project](#)[Configure](#)

## Project gatling-george-at-dev-functional

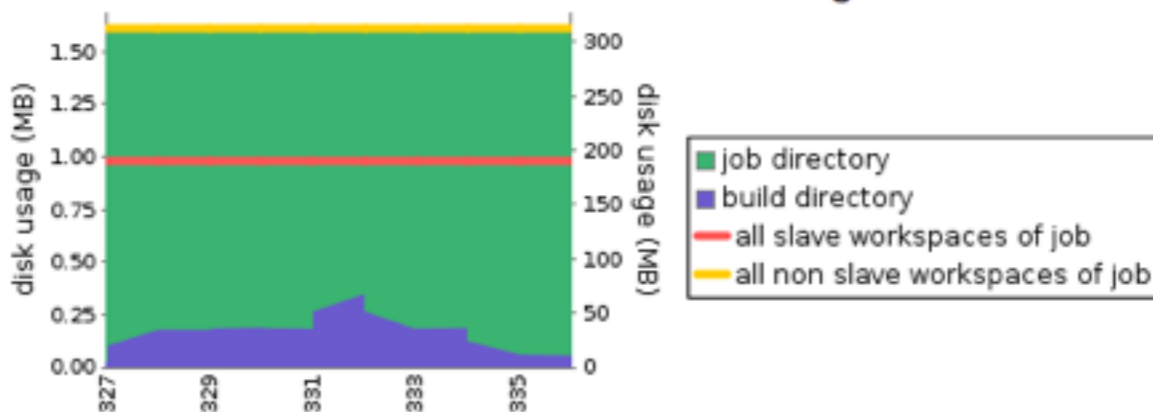
Functional Gatling test for the George Deployment on george.beeone.lan

[edit description](#)[Disable Project](#)

Project disk usage information + trend graph

**Disk Usage:** Workspace 501 MB (On slaves 189 MB, Non slave workspaces 312 MB), Builds 2 MB (Locked -), Job directory 2 MB

### Disk Usage Trend

[Workspace](#)[Recent Changes](#)

### Permalinks

- [Last build \(#336\), 14 min ago](#)
- [Last stable build \(#334\), 1 day 2 hr ago](#)
- [Last successful build \(#334\), 1 day 2 hr ago](#)
- [Last failed build \(#336\), 14 min ago](#)
- [Last unsuccessful build \(#336\), 14 min ago](#)
- [Last completed build \(#336\), 14 min ago](#)

Build History		
Build	Time	Size
<a href="#">#336</a>	May 31, 2016 9:44 AM	57 KB
<a href="#">#335</a>	May 31, 2016 7:30 AM	
<a href="#">#334</a>	May 30, 2016 7:30 AM	191 KB
<a href="#">#333</a>	May 29, 2016 7:30 AM	186 KB
<a href="#">#332</a>	May 28, 2016 7:30 AM	356 KB
<a href="#">#331</a>	May 27, 2016 7:30 AM	187 KB
<a href="#">#330</a>	May 26, 2016 7:30 AM	194 KB
<a href="#">#329</a>	May 25, 2016 7:30 AM	182 KB
<a href="#">#328</a>	May 24, 2016 3:21 PM	184 KB
<a href="#">#327</a>	May 24, 2016 2:53 PM	25 KB

[RSS for all](#) [RSS for failures](#)



# Ant Or Not To Ant

- Apache Ant to start Gatling
  - ▶ Simplify command line invocation
  - ▶ Providing standard Maven targets
- Add convenience function
  - ▶ Configuration information
  - ▶ Delete temporary data
  - ▶ Archiving test reports

# Ant Or Not To Ant

```
gatling.sh --simulation  
at.beeone.george.test.gatling.  
simulation.george.at.functional.Test
```

```
ant test
```

**What about running  
Gatling for other tenants  
and/or environments?!**







# Welcome To The Matrix



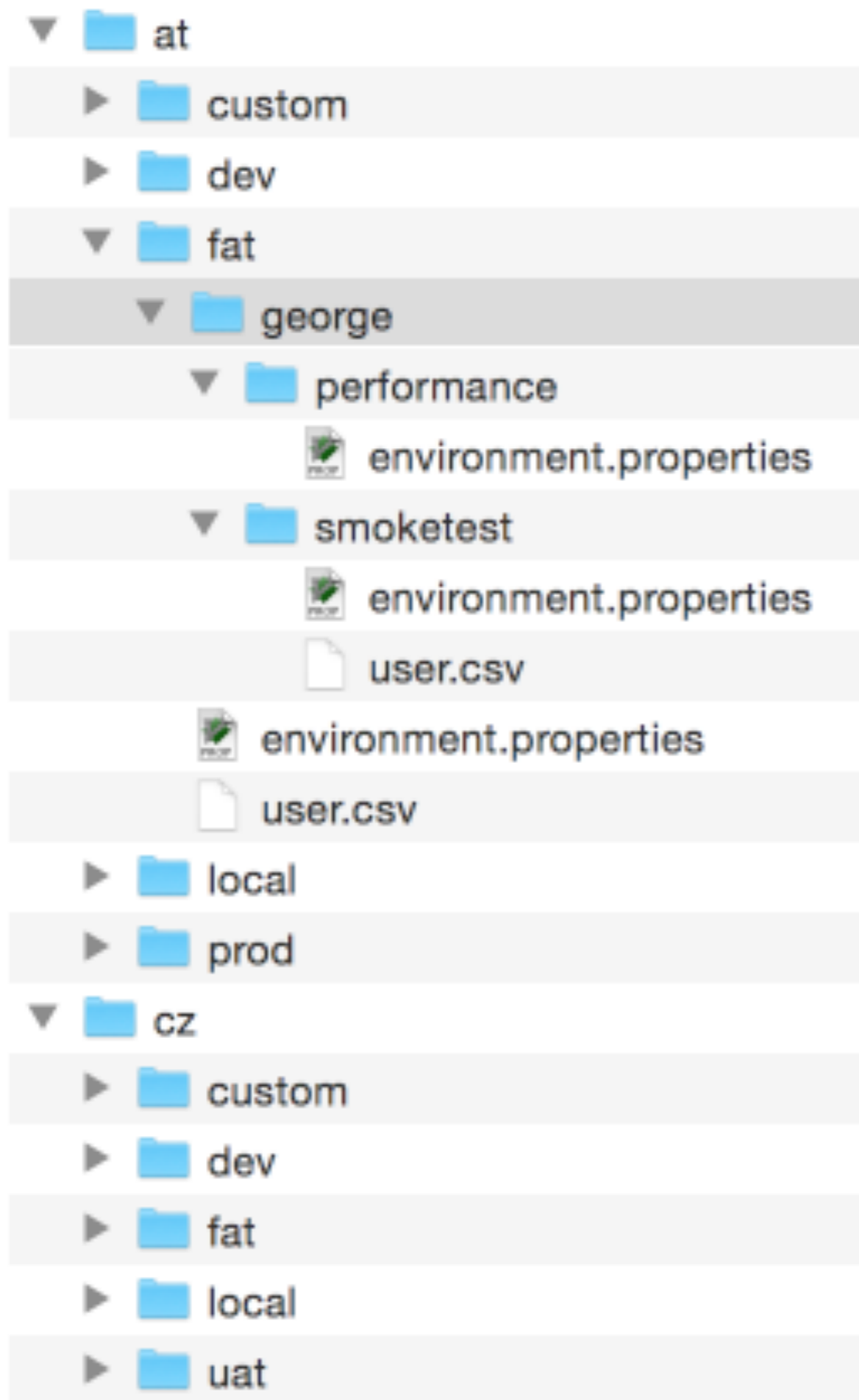
# Configuration Dimensions

<b>Dimension</b>	<b>Description</b>
tenant	The tenants (aka countries) provide additional features you need to test
application	The application to simulate, e.g. the web client might provide a different functionality than an iOS app
site	Different sites (staging environments) require different REST endpoint URLs
scope	Scope of tests, e.g. smoke test versus functional test

# Configuration Items

environment.properties	Base URLs for REST endpoints and other data
user.csv	User credentials consisting of username and password





- ‘at/fat/george/smoketest’
- ‘environment.properties’ are merged upwards where first value wins
- ‘user.csv’ are searched upwards where first file wins

***ant***

***-Dtenant=at***

***-Dsite=fat***

***-Dapplication=george***

***-Dscope=smoketest***

***clean info test***

# Server-side Refactoring





# Server Side Refactoring

- Large refactoring to expose DTOs only
- Mostly a mechanical task aided by the IDE
- There is still room for errors
  - ▶ Web UI can be quickly fixed
  - ▶ Breaking mobile apps?!

What about having a REST client  
which tests the REST endpoints  
every morning to see if the  
JSON response has changed?!



# Gatling Going Functional

- Pretty-print the server's JSON response
- Skip moving parts, e.g. "lastLoginDate"
- Save final JSON with a meaningful name
- Ant script compares current and expected JSON responses



```
object GeorgeTransactions {

    val RELATIVE_URL = "my/transactions"

    val getAll = exec(http("my/trx")
        .get(ConfigurationTool.getUrl("georgeapi", RELATIVE_URL))
        .header("Authorization", "bearer ${token}")
        .queryParams("pageSize", 50)
        .check(
            jsonPath("$.ofType [Any] .find .saveAs ("lastResponse")
        ))
        .exec(session => {
            val userId: String = session.get("user").as[String]
            JsonResponseTool.saveToFile(session,
                "lastResponse",
                "george",
                RELATIVE_URL,
                userId)
            session
        })
    })
}
```

# Gatling Going Functional

- JSON path expression to extract response
- Stored as “lastResponse” in user session
- Extract current user id from user session
- The “JSONResponseTool” pretty-print and stores JSON content in file system
- Creates meaningful name “george-my-transactions- $\{userId\}$ .json”

**ant clean info record**

**ant clean info verify**



# Did Not Work For Us

- Test users shared with QA
- Regular import of new transactions
- Useful for comparing JSON responses before/after a server deployment

**Can we use this  
Gatling tool for  
performance tests?**



**Anonymous Manager**

**YES,  
WE CAN.**





**IT'S A BANK,  
STUPID.**





# The Challenges

- Working with highly sensitive data
- Running within a secure data center
- No access to load injector boxes
- Minimal effort for rollout
- Time window from 22:00 to 03:00
- Pre-configured test scenarios

# Disaster Recovery Tests

- Test failover to backup data center
- Nightly session with multiple teams
- Using Gatling seriously for the first time
  - ▶ VMs with 4 CPUs and 8 GB RAM
  - ▶ Simulation of 1.000 concurrent users
- Gatling is fast and scales well







[New Item](#)[People](#)[Build History](#)[Manage Jenkins](#)[Credentials](#)[My Views](#)[add description](#)

All

Gatling

+

S	W	Name ↓	Last Success	Last Failure	Last Duration	
			23 hr - <a href="#">#3</a>	N/A	7 min 6 sec	
			23 hr - <a href="#">#4</a>	N/A	5 min 34 sec	
			23 hr - <a href="#">#3</a>	N/A	7 min 6 sec	
			N/A	N/A	N/A	
			12 days - <a href="#">#7</a>	12 days - <a href="#">#4</a>	7 min 17 sec	
			2 days 12 hr - <a href="#">#11</a>	1 day 0 hr - <a href="#">#14</a>	2 min 10 sec	
			12 days - <a href="#">#5</a>	N/A	0.54 sec	

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

Icon:  
[S](#) [M](#) [L](#)[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

**READ THE**

**SOURCE**

**LUKE**



```
package at.beeone.george.test.gatling.simulation.george.at.rampupload
```

```
import at.beeone.george.test.gatling.common.ConfigurableSimulation
```

```
import at.beeone.george.test.gatling.simulation.george.at.TenantTestBuilder
```

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

```
class Test extends ConfigurableSimulation {
```

```
  val users = scenario("Rampup Load") {
```

```
    repeat(getSimulationLoops) {
```

```
      feed(userCsv.circular.random)
```

```
        .exec(
```

```
          TenantTestBuilder.create("performance")
```

```
        )
```

```
    }
```

```
  }
```

```
  setUp(users.inject(rampUsers(getSimulationUsers) over getSimulationUsersRampup))
```

```
    .maxDuration(getSimulationDuration)
```

```
    .protocols(httpConfServer)
```

```
}
```



```
/**
 * Convenience class to have the test steps in one single place.
 */
object TenantTestBuilder {

  val GEORGE_STATE_POLLING_DURATION = 120

  def create(scope: String): List[ChainBuilder] = {

    val result = scope match {
      case "functional" =>
        List(
          User.init,
          Token.get,
          GeorgeConfiguration.get,
          GeorgePersonalFinanceManager.post,
          GeorgeAccounts.get,
          GeorgeAccounts.stats,
          GeorgeTransactions.get,
          GeorgeTransactions.queryLastYear,
          GeorgeTransactions.queryAmount,
          GeorgePreferences.get,
          GeorgeCategorization.categorize,
          GeorgeImages.userImage,
```



**Building  
Blocks**

# Five Years Ago

## Hudson

search ?

[ENABLE AUTO REFRESH](#)

[add description](#)

[New Job](#)

[Manage Hudson](#)

[People](#)

[Build History](#)

[Edit View](#)

**Build Queue**

No builds in the queue.

**Build Executor Status**

#	Status
1	Idle
2	Idle

Icon: [S](#) [M](#) [L](#)

[Legend](#)  for all  for failures  for just latest builds

Page generated: 24 Oct 2011 8:21:00 PM [Hudson ver. 1.395](#)

All	ER-ORT	ER-TCH	Maintenance	Monitoring	ORT-BO	Smoketests	TCH-CRM	VPC	+
S	W	Job ↓	Last Success	Last Failure	Last Duration				
		<a href="#">er-ort-oefc-baseline</a>	28 days ( <a href="#">#10003</a> )	N/A	1 hr 0 min				
		<a href="#">er-ort-oefc-endurance</a>	6 days 0 hr ( <a href="#">#10004</a> )	16 days ( <a href="#">#10003</a> )	37 sec				
		<a href="#">er-ort-oefc-jmeter-shutdown</a>	5 days 22 hr ( <a href="#">#28</a> )	N/A	0.44 sec				
		<a href="#">er-ort-oefc-smoketest</a>	1 day 2 hr ( <a href="#">#10025</a> )	11 days ( <a href="#">#10017</a> )	23 sec				
		<a href="#">er-ort-oefc-stress</a>	11 days ( <a href="#">#10020</a> )	6 days 0 hr ( <a href="#">#10031</a> )	46 min				



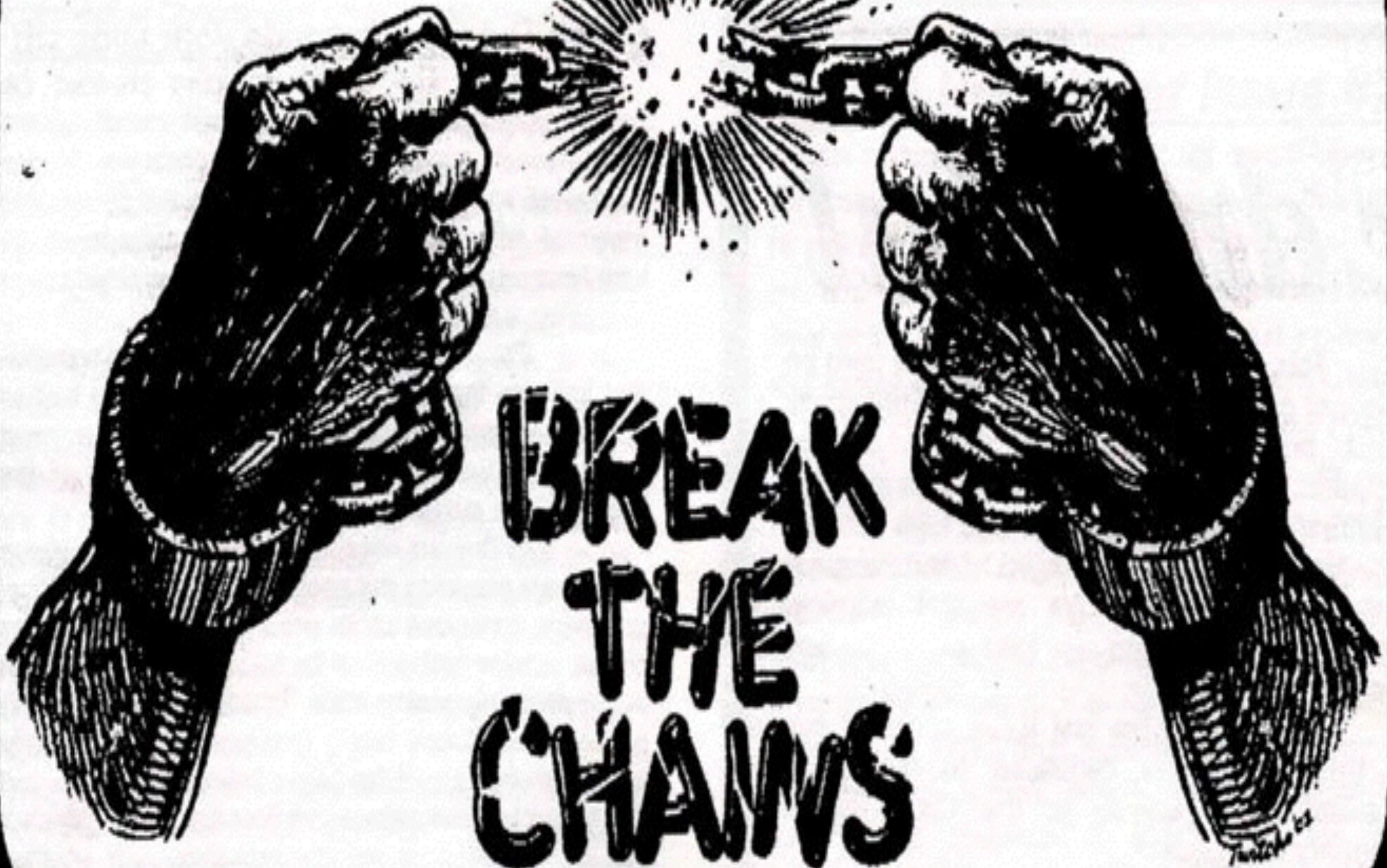
# That Was Five Years Ago

- JMeter tests run on the command line
- Ant scripts as cross-platform glue code
  - ▶ Multi-dimensional configuration
  - ▶ Archiving of test reports
- Hudson job to trigger test execution
  - ▶ Scheduler, user management, notifications

# The Building Blocks

- Test execution over command line
  - ▶ Cater for SSH only access
- Ant script as cross-platform glue code
  - ▶ Human-friendly command line
- CI server to trigger test execution
  - ▶ Pre-configured test scenarios





**BREAK  
THE  
CHAINS**

1924



# The Case for CI Servers

- Everyone can start a performance test
- Test results are visible and archived
- CI servers have powerful features
  - ▶ User management, scheduler, emails
- Wired with version control system
  - ▶ Always using the latest version

# Gatling @ Erste Bank



# The Current State

- Gatling is well established for AT & CZ
  - ▶ Using a Gatling, Ant, Jenkins setup
  - ▶ Daily tests for DEV & FAT
- More things for CZ deployment
  - ▶ Performance test setup for CZ PROD
  - ▶ Competing with HP LoadRunner





# Is Gatling For You?

- Gatling's DSL is elegant & powerful
  - ▶ Scala & DSL learning curve
  - ▶ Requires solid development skills
- Developer-friendly tool
  - ▶ Code only
  - ▶ IDE support & refactoring
  - ▶ Works on Windows, Linux & OS X





Things To Take Home



# Things To Take Home

- Gatling is powerful but not for everyone
- Move performance tests to a CI server
  - ▶ Ease of use & visibility
- Embrace the “configuration matrix”
  - ▶ You will have DEV & PROD anyway

# Things To Take Home

- Performance test tools should be agile
  - ▶ Move freely from your laptop to a newly provisioned VM in a data center
- Performance tests are an asset
  - ▶ Smoke tests for free
  - ▶ Re-used in unplanned & creative ways





Unplanned & Creative Ways



# Questions & Answers



# Resources

- <http://gatling.io>
- <https://huddle.eurostarsoftwaretesting.com/gatling-tales-from-a-journey/>
- <http://automationrhapsody.com/performance-testing-with-gatling/>
- <https://github.com/sgoeschl>
- <http://people.apache.org/~sgoeschl>