

# μServices Architecture

# things about me

- SA at EPAM Systems
- primary skill is Java
- hands-on-coding with **Groovy**, **Ruby**
- trying to learn some **Scala** and **Erlang**
- passionate about **agile**, **clean code** and **devops**

Izzet Mustafayev@EPAM Systems



@webdizz



webdizz



izzetmustafaiev



<http://webdizz.name>

# Agenda

- What is this?
- Architecture
- Case study
- Existing tools
- Summary
- How to get there?
- Q&A

# WHAT IS THIS?

I DON'T EVEN...

# Microservices

*The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.*

---

25 March 2014



**James Lewis**

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.



**Martin Fowler**

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

# Overview

- Small (10-100 LOC)



# Overview

- **Small** (10-100 LOC)
- **Lightweight** (run several per box)

# Overview

- **Small** (10-100 LOC)
- **Lightweight** (run several per box)
- Takes **strength** of platform/language (**polyglot**)

# Overview

- **Small** (10-100 LOC)
- **Lightweight** (run several per box)
- Takes **strength** of platform/language (**polyglot**)
- **Independent** (development/deployment)

# Overview

- **Small** (10-100 LOC)
- **Lightweight** (run several per box)
- Takes **strength** of platform/language (**polyglot**)
- **Independent** (development/deployment)
- **Stateless** (everything persisted in DB)

# Overview

- **Small** (10-100 LOC)
- **Lightweight** (run several per box)
- Takes **strength** of platform/language (**polyglot**)
- **Independent** (development/deployment)
- **Stateless** (everything persisted in DB)
- **Monitored** (health and business value)

# Architecture

- Design
- Implementation
- Delivery
- Monitoring



# Design

- What ab SOA?
- Externalize Configuration
- Versioning
- Communication Contract
- Asynchronicity
- Reusability
- Scalability
- Availability



# What ab SOA?



- Sensible idea
- Combat challenges of large monolithic applications
- Lack of guidance

# Externalize Configuration



- Support for  $X > 1$  environments
- Run-time changes

# Versioning



- Leave multiple old microservice versions running
- Fast introduction vs. slow retirement asymmetry

# Communication Contract



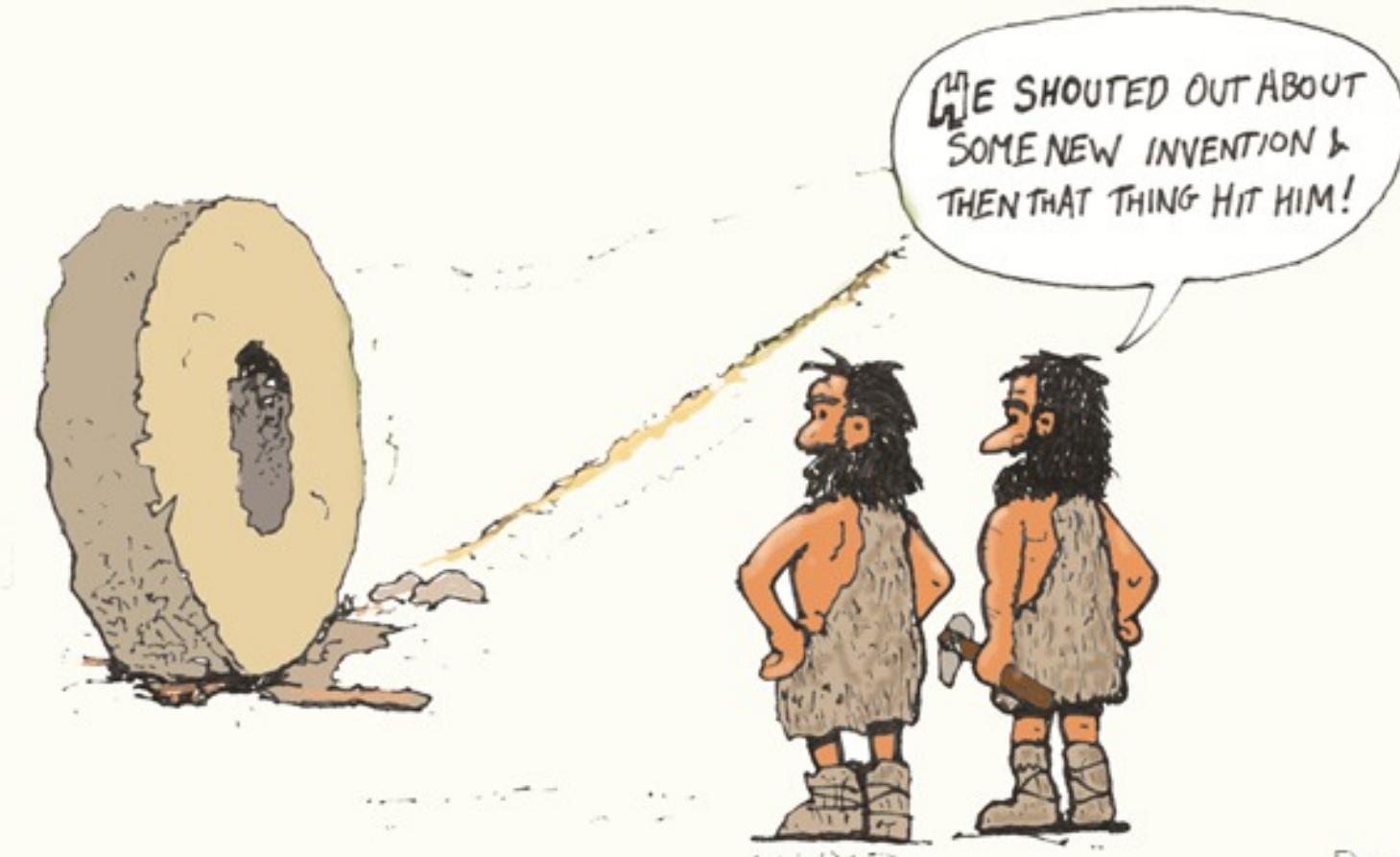
- Own service - own client
- Thin transport

# Asynchronicity



- Non-blocking
- Reactive model
- Messaging to send and forget

# Reusability



DeepFat '09

- Each service does one dedicated thing
- Utilize suitable components/frameworks

# Scalability



- Cohesive
- Stateless
- Independent

# Availability

- Fault-tolerance
- Introduce failures

# Implementation

A close-up photograph of a person's hands cupping a small, young plant with four leaves. The plant is growing out of a mound of dark, moist soil. The background is blurred, showing what appears to be a natural outdoor setting with trees.

- Separate Concerns

# Separate Concerns

- Inverse Conway's law – teams own service groups



# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice



# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice
- One developer independently produces a microservice

# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice
- One developer independently produces a microservice
- Each microservice is it's own build

# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice
- One developer independently produces a microservice
- Each microservice is it's own build
- Stateful cached data access layer

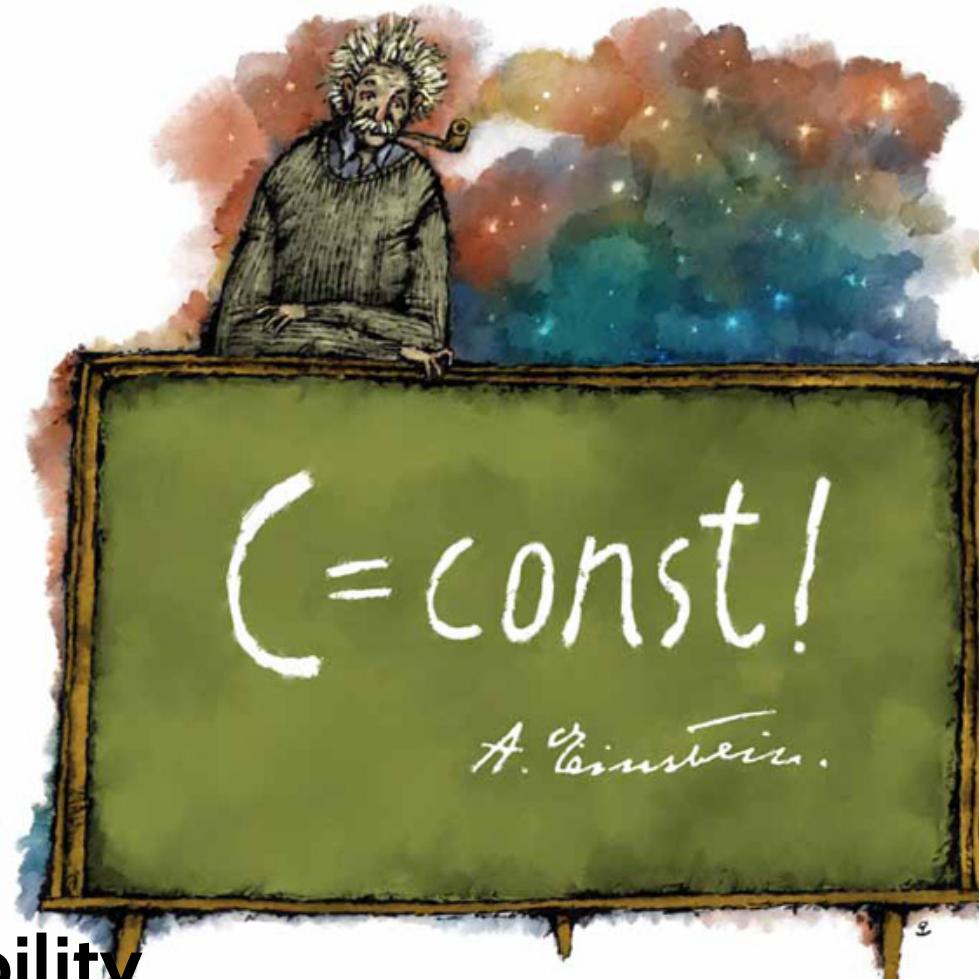
# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice
- One developer independently produces a microservice
- Each microservice is it's own build
- Stateful cached data access layer
- Lightweight mocked dependencies

# Separate Concerns

- Inverse Conway's law - teams own service groups
- One “verb” per single function microservice
- One developer independently produces a microservice
- Each microservice is it's own build
- Stateful cached data access layer
- Lightweight mocked dependencies
- Easy to run locally

# Delivery



- **Disposability**
- **Containerisation**

# Disposability



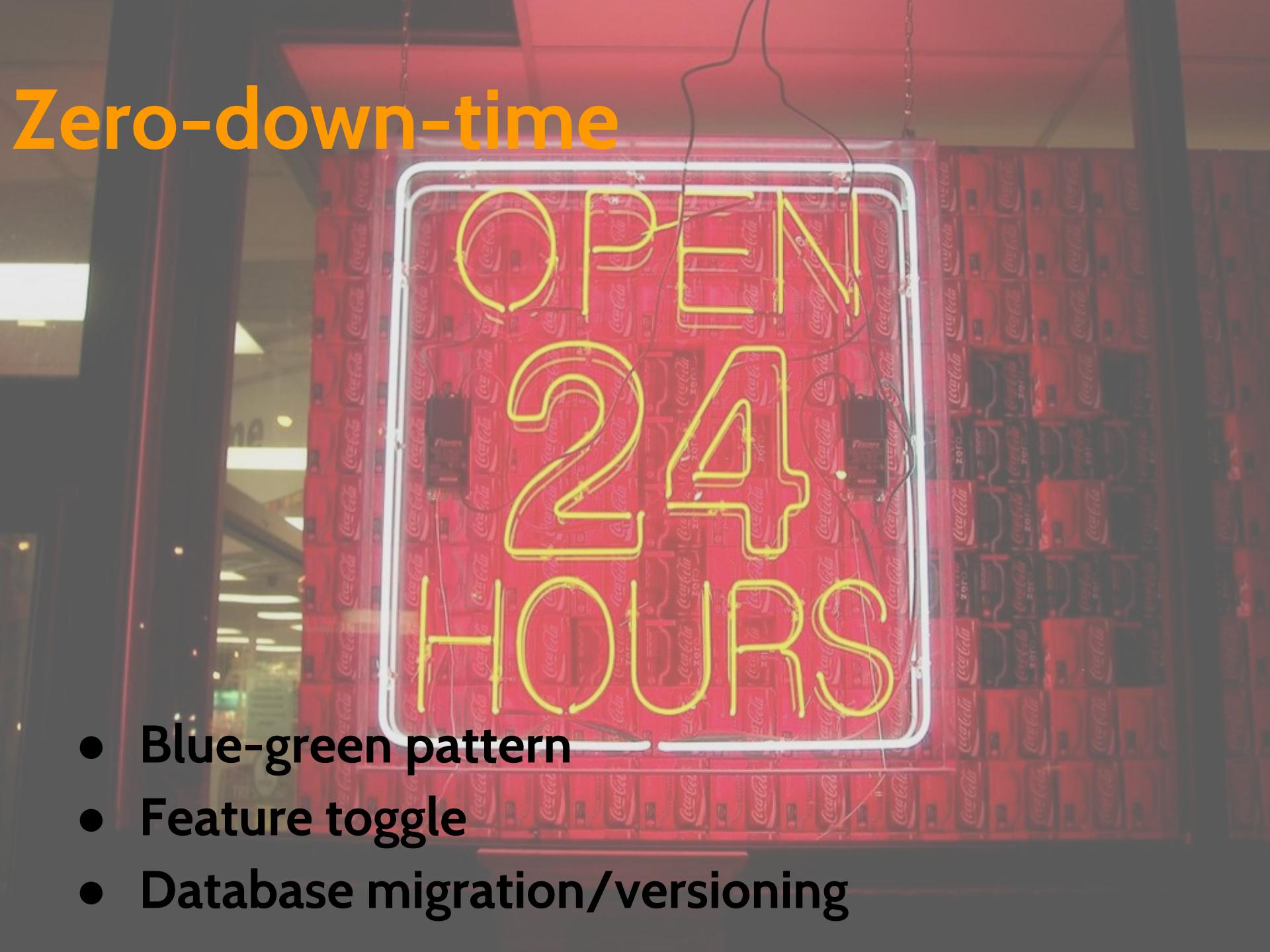
- **Build**
- **Run**
- **Destroy**

# Containerisation



- Container as deployment artifact
- Environment agnostic
- New version - new container
- All dependencies built in

# Zero-down-time

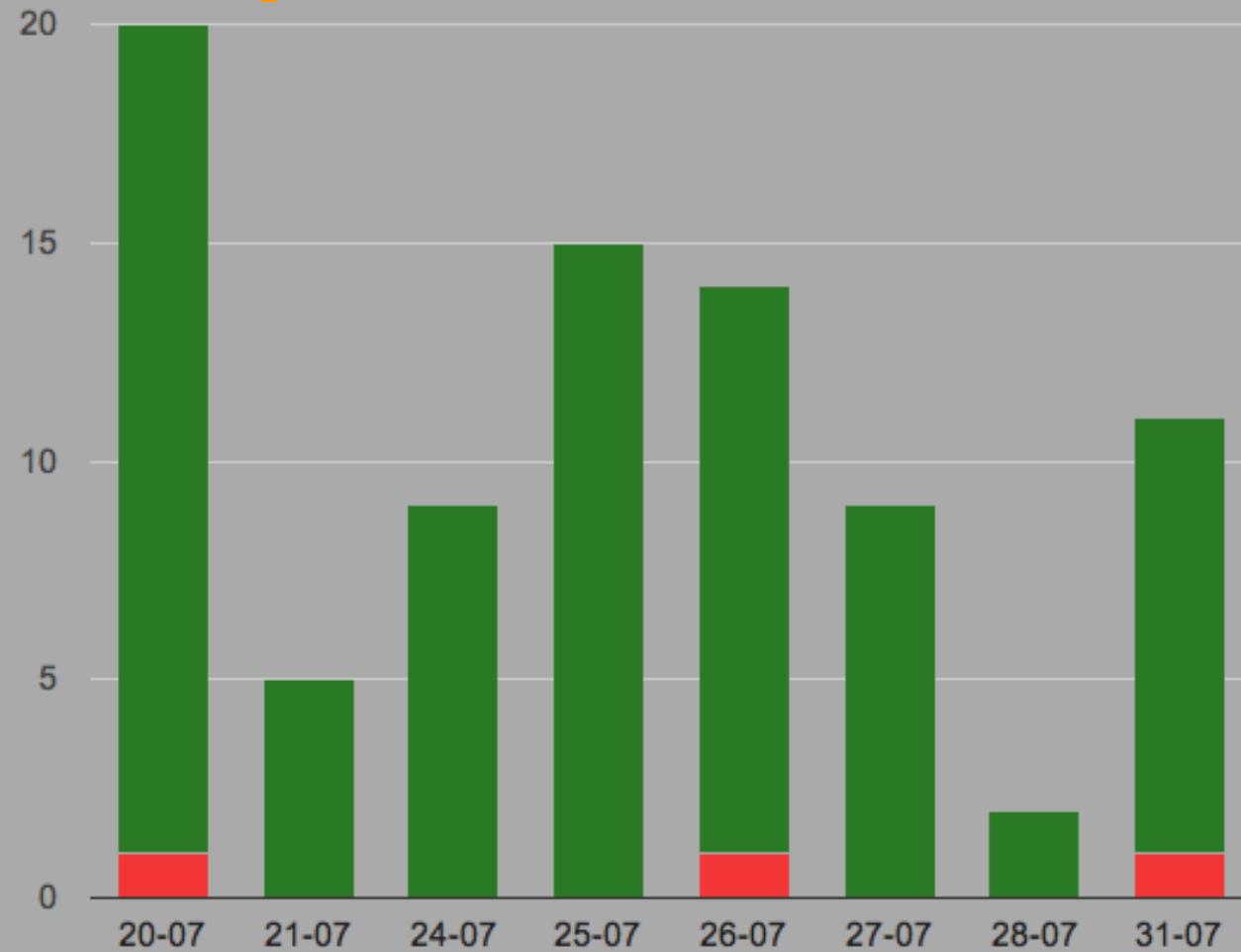
A red Coca-Cola delivery truck is shown from a three-quarter perspective. A large, illuminated yellow sign on the side of the truck reads "OPEN 24 HOURS". The background is dark, suggesting it might be nighttime or the truck is parked in a shaded area.

OPEN  
24  
HOURS

- Blue-green pattern
- Feature toggle
- Database migration/versioning

# Frequency

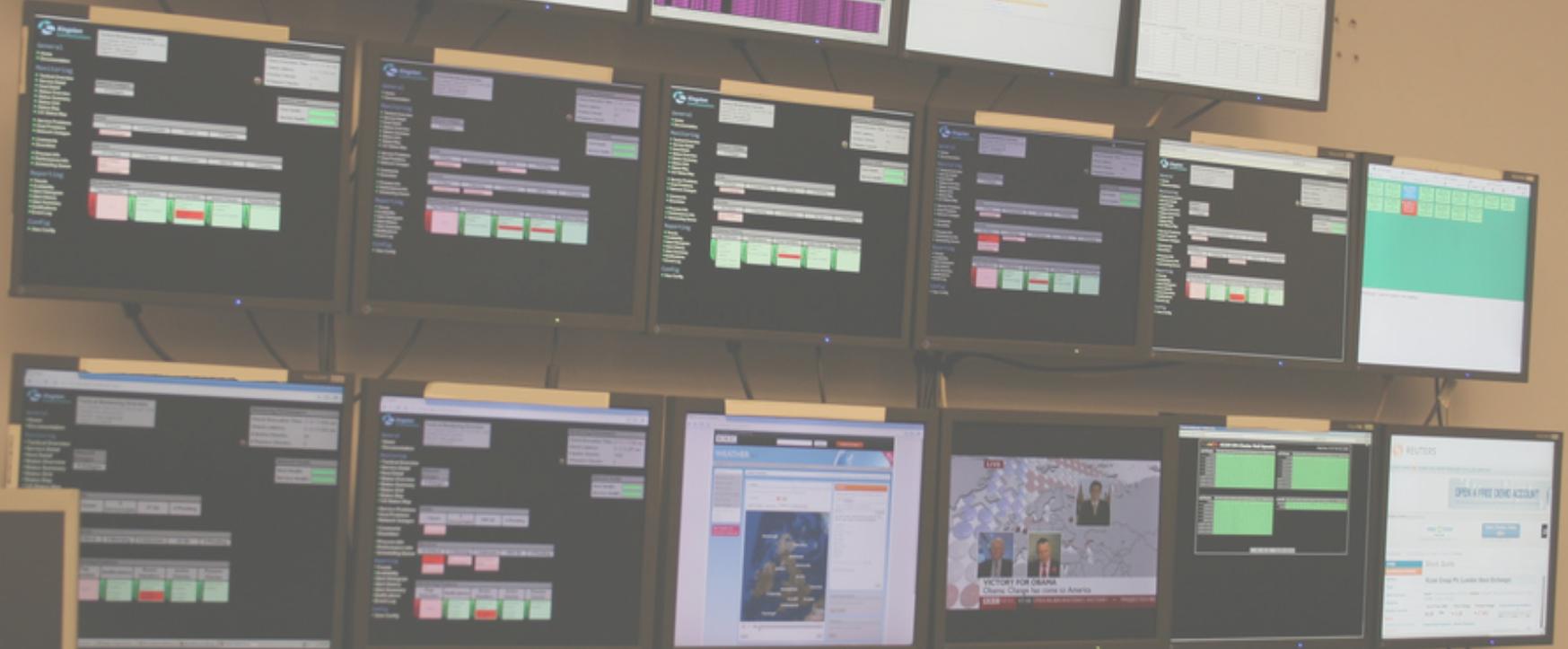
Deployments by date



- Fast rollouts
- Fast rollbacks

*Date*

# Monitoring



- Technical
- Business
- Stats

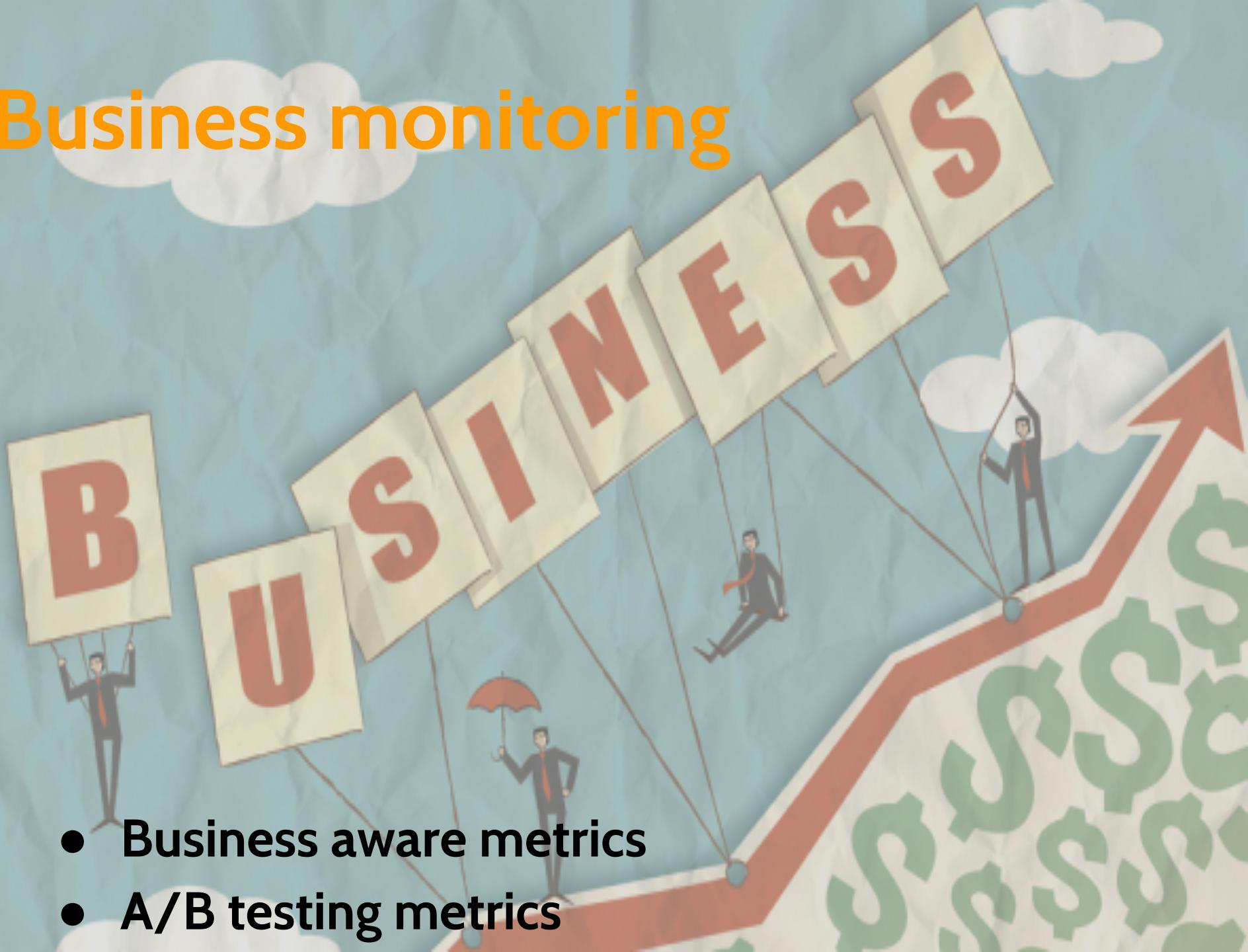
# Technical monitoring

```
0.0%] Hostname: frylock
0.5%] Uptime: 6 days, 12:26:44
0.7%] Tasks: 217; 1 running
0.5%] Load average: 0.06 0.19 0.25
0.0%]
```

PID	USER	PRI	RES	VIRT	S	10 CPU%	MEH%	TIME+	Command
28679	www-data	20	7452	97352	S	0	0.0	0.0	0:00:03
28678	www-data	20	9668	98112	S	2.59	0.0	0.1	0:00:03
28108	root	20	11484	683M	S	0	0.0	0.1	0:11:28
28544	root	20	4	168	S	0	0.0	0.0	0:00:01
28568	root	20	1544	21188	S	0	0.0	0.0	0:00:00
28569	root	20	32	188	S	0	0.0	0.0	0:00:00
28573	root	20	4	168	S	0	0.0	0.0	0:00:00
28575	root	20	3029	61364	S	0	0.0	0.0	0:00:00
28572	root	20	4	168	S	0	0.0	0.0	0:00:00
28576	root	20	1844	58228	S	0	0.0	0.0	0:00:00
28581	lee	20	1892	29772	S	0	0.0	0.0	0:00:35
28591	lee	20	158M	374M	S	0	0.0	1.0	0:05:33
28606	lee	20	162M	398M	S	0	7.0	1.0	0:05:18
28669	lee	20	174M	963M	S	0	0.0	1.1	0:05:52
28651	lee	20	174M	963M	S	0	0.0	1.1	0:06:25
28622	lee	20	169M	532M	S	0	0.0	1.1	0:01:76
28571	root	20	4	168	S	0	0.0	0.0	0:00:00
28574	root	20	4344	98268	S	0	0.0	0.0	0:00:00
28586	www-data	20	2384	98696	S	0	0.0	0.0	0:00:05
28585	www-data	20	1888	98696	S	0	0.0	0.0	0:00:02
28584	www-data	20	1888	98696	S	0	0.0	0.0	0:00:03
28583	www-data	20	2649	98696	S	0	0.0	0.0	0:00:32
28578	root	20	4	168	S	0	0.0	0.0	0:00:00
28577	root	20	615	7436	S	0	0.0	0.0	0:00:00
13152	dillinger	20	1312	37964	S	0	0.0	0.0	0:00:00
13153	dillinger	20	1616	28764	S	0	0.0	0.0	0:00:00
13154	dillinger	20	87724	717M	S	0	0.0	0.5	0:01:89
13526	root	20	1584	54468	S	0	0.0	0.0	0:00:02
11578	lee	20	8952	27672	S	0	0.0	0.1	0:00:28
11784	root	20	1840	39180	S	0	0.0	0.0	0:00:01
11715	root	20	10888	28864	S	0	0.0	0.1	0:00:17
7958	root	20	4	168	S	0	0.0	0.0	0:02:35
38845	lee	20	171M	398M	S	0	0.0	1.1	1:07:56
38146	lee	20	174M	418M	S	0	0.0	1.1	0:43:42
38142	lee	20	183M	975M	S	0	0.0	1.1	0:44:42
38127	lee	20	188M	424M	S	0	0.0	1.2	0:49:42
18977	lee	20	163M	398M	S	0	0.0	1.0	0:00:10
18991	lee	20	161	398M	S	0	0.0	1.0	0:00:17
11830	lee	20	163M	414M	S	0	0.0	1.0	0:00:09
11833	lee	20	163M	414M	S	0	0.0	1.0	0:00:09
11848	lee	20	163M	418M	S	0	0.0	1.0	0:00:10
11862	lee	20	163M	420M	S	0	0.0	1.0	0:00:10
11866	lee	20	163M	420M	S	0	0.0	1.0	0:00:10
11884	lee	20	163M	438M	S	0	0.0	1.0	0:00:09
11126	lee	20	164M	446M	S	0	0.0	1.0	0:00:10
11134	lee	20	161M	459M	S	0	0.0	1.0	0:00:10
11135	lee	20	161M	459M	S	0	0.0	1.0	0:00:10

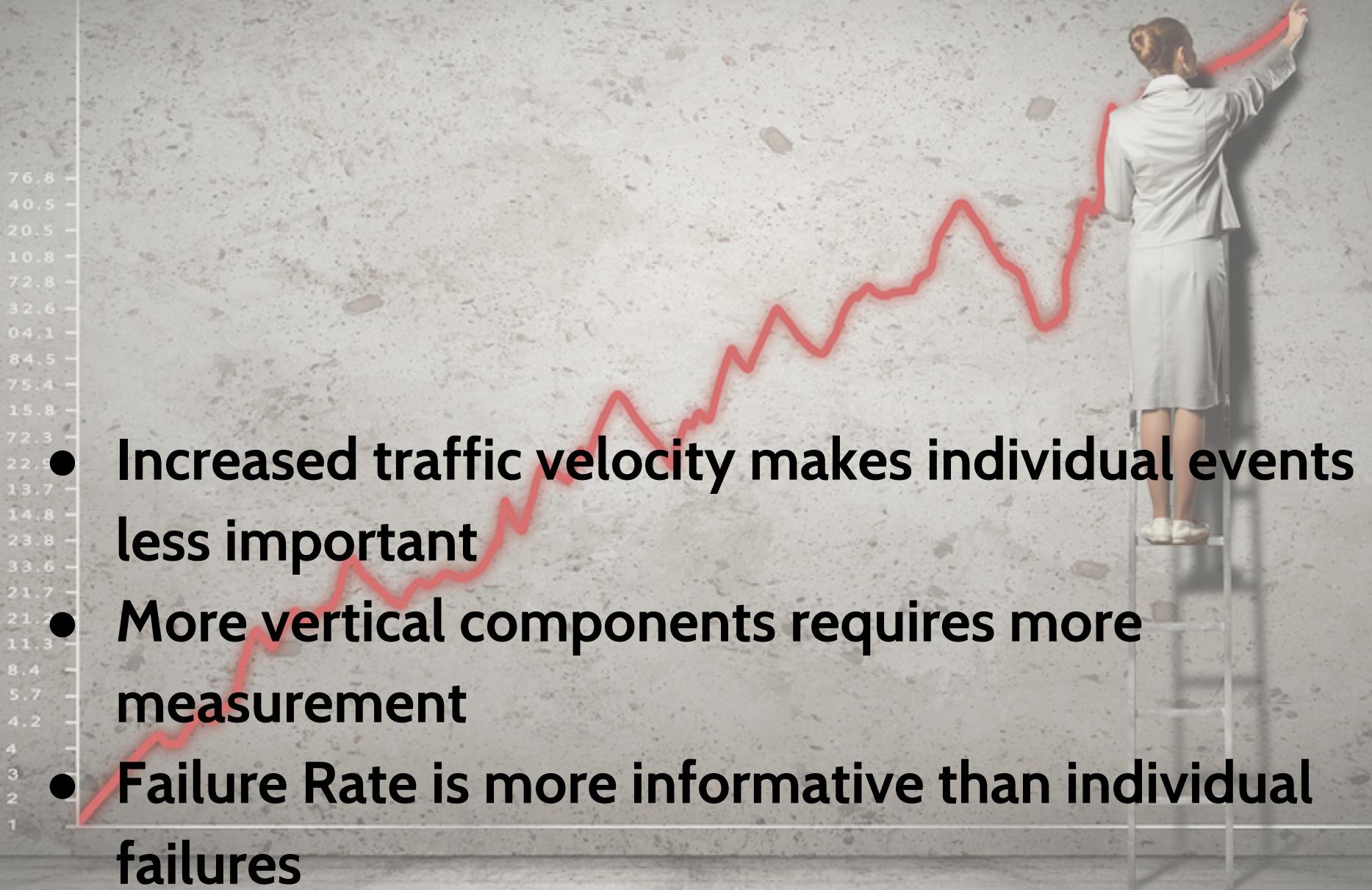
- Hardware monitoring
- Resources utilisation
- Health checking

# Business monitoring



- Business aware metrics
- A/B testing metrics

# Stats matter on scale

- 
- Increased traffic velocity makes individual events less important
  - More vertical components requires more measurement
  - Failure Rate is more informative than individual failures

# Benefits



# Toolset unchained



- Polyglot technology stack
- Polyglot persistence
- Frameworks
- Thin transport

# Scalability

- HTTP stack
- Independent provisioning
- Fine tuning
- Elasticity

# Independence



- Development
- Testing
- Deployment
- Reliability

# Shortcomings



# Shortcomings

- More responsibility from Devs to support Ops

# Shortcomings

- More responsibility from Devs to support Ops
- Polyglot infrastructure (if any)

# Shortcomings

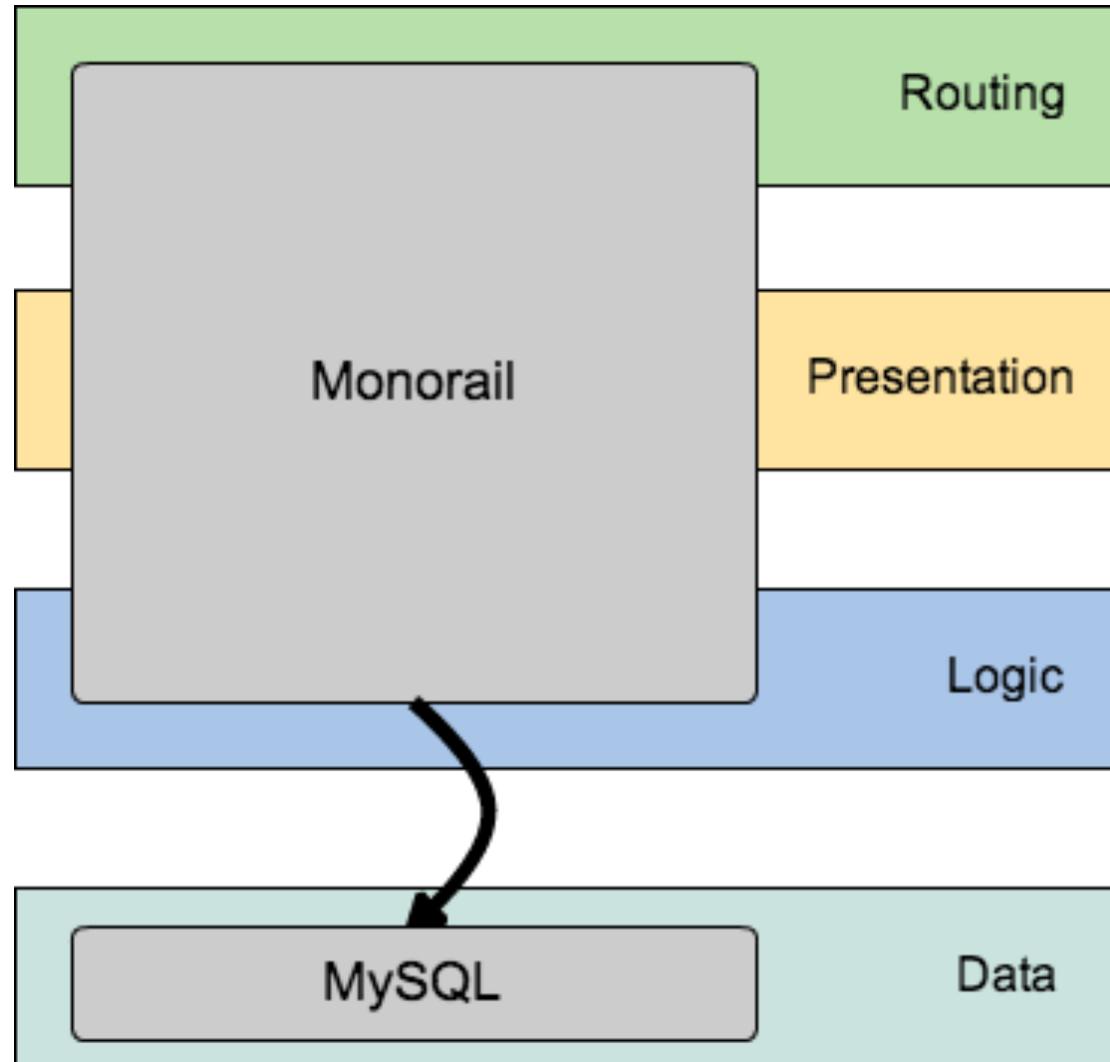
- More responsibility from Devs to support Ops
- Polyglot infrastructure (if any)
- Orchestration

# Shortcomings

- More responsibility from Devs to support Ops
- Polyglot infrastructure (if any)
- Orchestration
- Costs



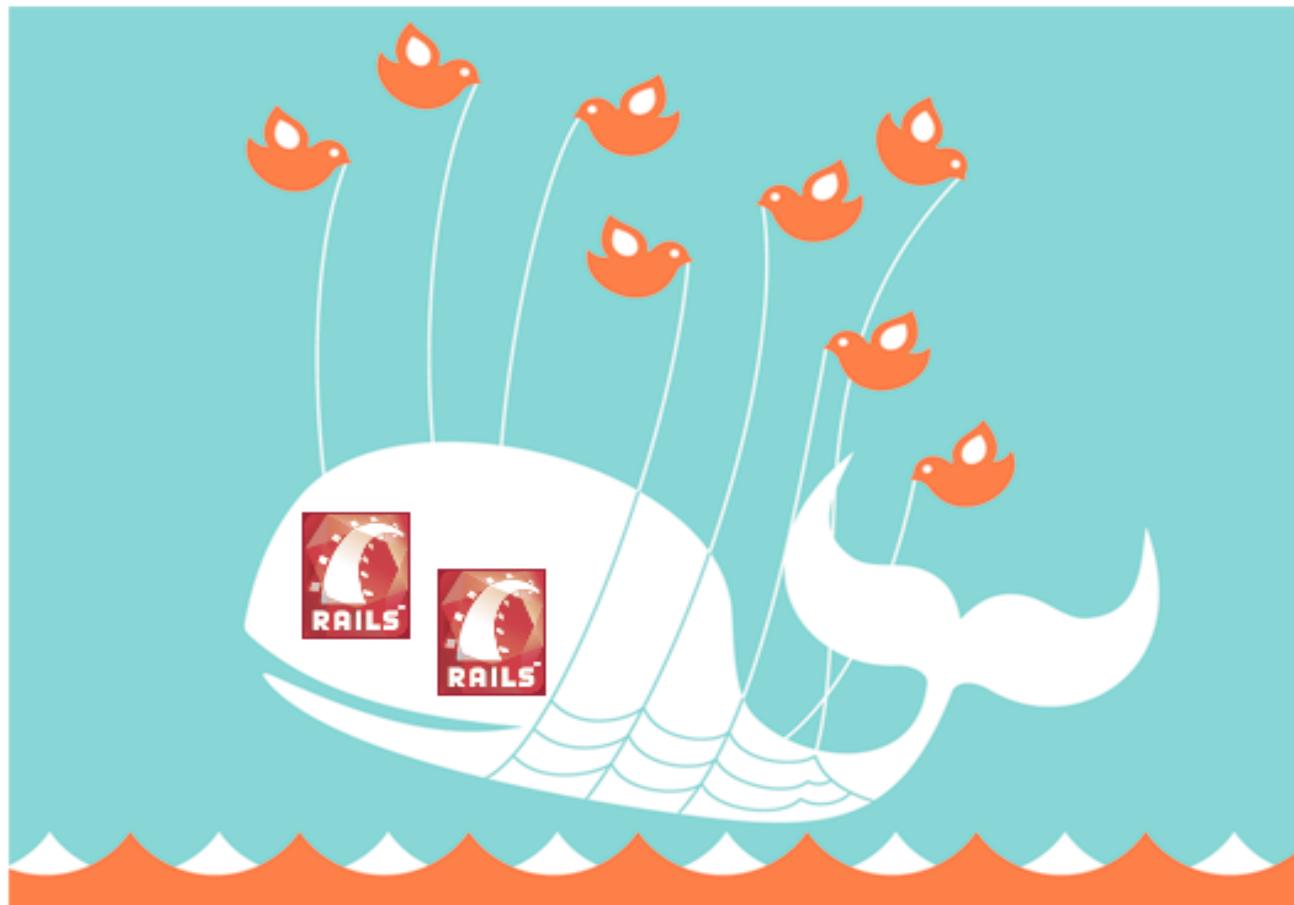
# Twitter before



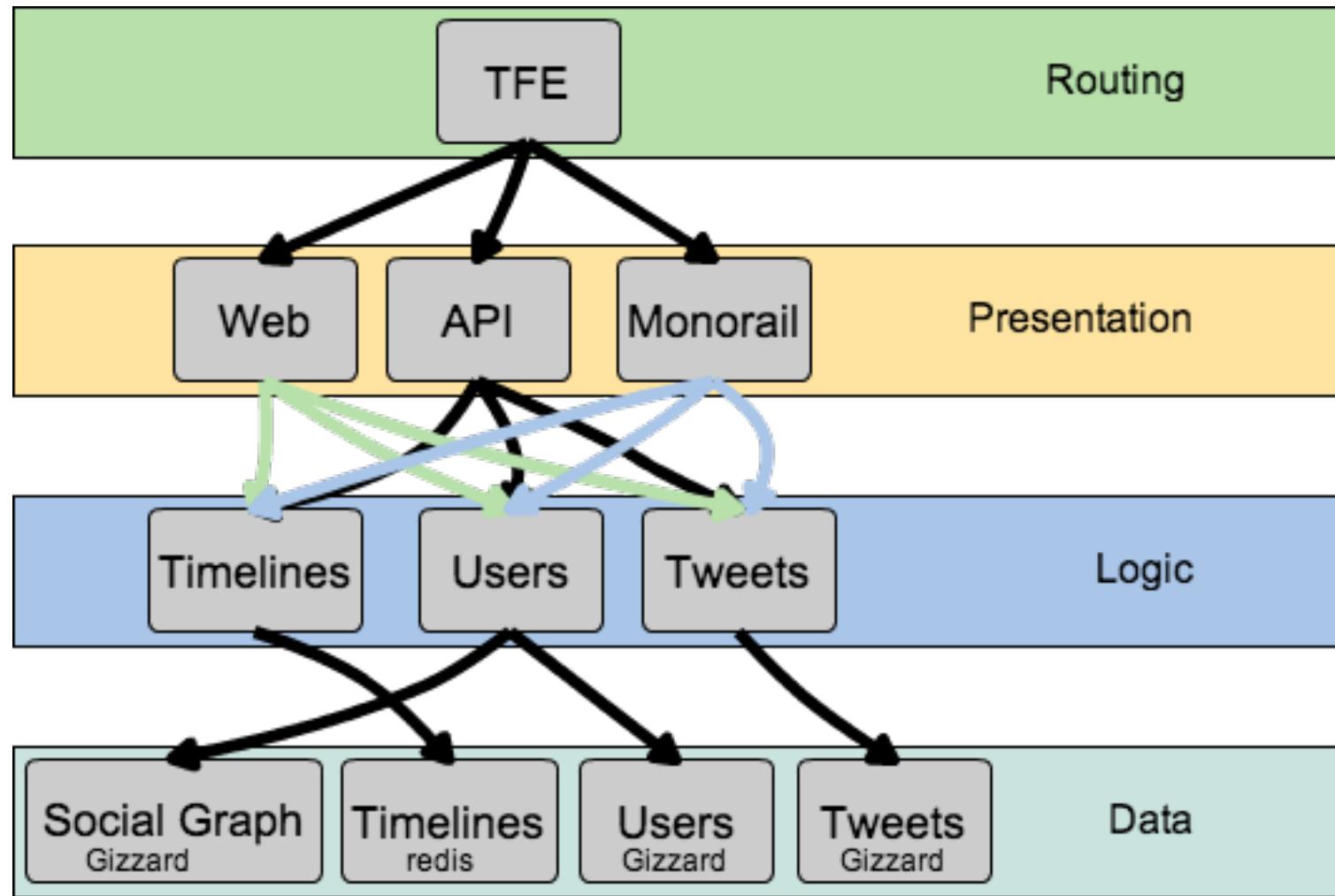


**Twitter is currently down for temporary maintenance.**

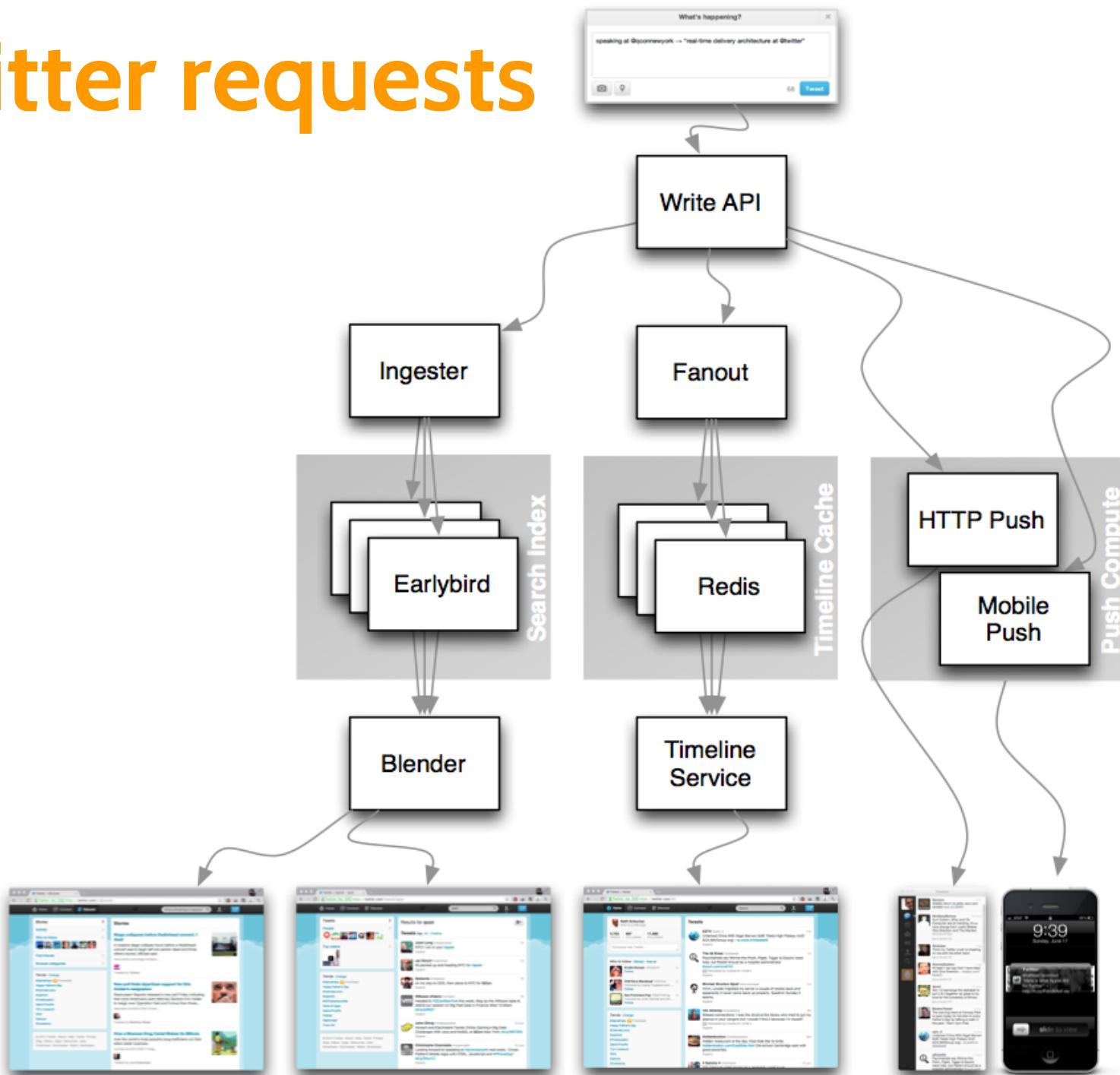
We expect to be back in 30 minutes. Thanks for your patience.



# Twitter now\*



# Twitter requests



# Lessons learned

- Speed wins in the marketplace



# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development

# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process

# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture

# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Simple patterns automated by tooling

# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Simple patterns automated by tooling
- Microservices for speed and availability

# Lessons learned

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Simple patterns automated by tooling
- Microservices for speed and availability
- Use statistics to monitor behavior

# Toolset

- Dropwizard
- Spring Boot
- Vert.X
- More...

# Dropwizard

<http://dropwizard.github.io/dropwizard/>

**Java framework for developing ops-friendly, high-performance, RESTful web services.**

- pulls together **stable**, mature **libraries** from the Java ecosystem into a simple, **light-weight** package
- has out-of-the-box support for sophisticated **configuration**, application **metrics**, **logging**, **operational tools**, and much more



# Spring Boot <http://projects.spring.io/spring-boot/>

**Takes an opinionated view of  
building production-ready Spring  
applications.**

- favors convention over configuration and is designed to get you up and running as quickly as possible.
- production-ready features such as metrics, health checks and externalized configuration



# Vert.X <http://vertx.io/>

- Lightweight, reactive, application platform
- Superficially similar to Node.js - but not a clone!
- Inspired also from Erlang/OTP
- Polyglot
- High performance
- Simple but not simplistic

VERT.X

# More...

- **Ratpack** <http://www.ratpack.io/>
- **Sinatra** <http://www.sinatrarb.com/>
- **Webbit** <https://github.com/webbit/webbit>
- **Finagle** <http://twitter.github.io/finagle/>
- **Connect** <http://www.senchalabs.org/connect/>
- **XDropWizard** <https://github.com/timmolter/XDropWizard>

# Summary



# Summary

- Speed wins in the marketplace



# Summary

- Speed wins in the marketplace
- Separated concerns at app level



# Summary

- Speed wins in the marketplace
- Separated concerns at app level
- Focused and automated

# Summary

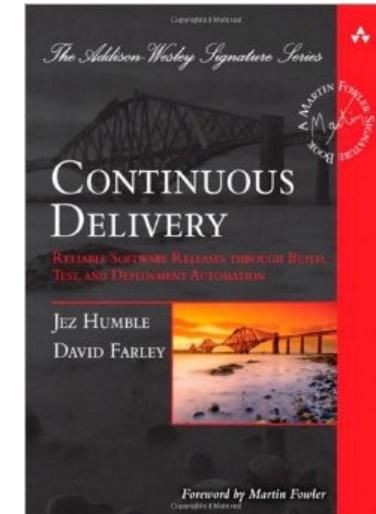
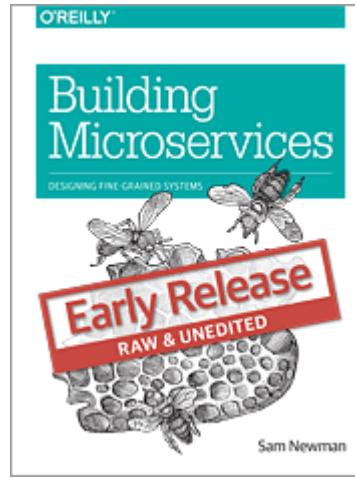
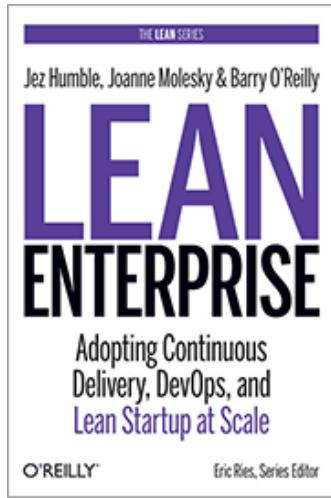
- Speed wins in the marketplace
- Separated concerns at app level
- Focused and automated
- Elastically scalable

# Summary

- Speed wins in the marketplace
- Separated concerns at app level
- Focused and automated
- Elastically scalable
- Polyglot architecture

# How to get there?

- Automate repeating activities
- “Run what you wrote” – root access and duty
- Freedom and responsibility for developers



# References

- <http://martinfowler.com/articles/microservices.html>
- <http://yobriefca.se/blog/2013/04/28/micro-service-architecture/>
- <http://goo.gl/4d0Vjj>
- <http://goo.gl/RGgnd3>
- <http://goo.gl/TmZ4Ee>
- <http://12factor.net/>
- <http://goo.gl/tnPxLP>

Q&A

(?)



**FEEDBACK**

A large, bold, green pixelated text "FEEDBACK" is centered within a thick, dark green circular arrow. The arrow is oriented clockwise, with its tail pointing towards the top-left and its head pointing towards the bottom-right. The entire graphic has a high-contrast, digital aesthetic.



Izzet Mustafayev@EPAM Systems

 @webdizz  webdizz  izzetmustafaiev  
 <http://webdizz.name>