Microservices: lessons from the trenches

- Andy Ben-Dyke

1. 2 key takeaways
2. Microservices 101
3. How did I get here?
4. 6 Pros and 3 Cons
5. Summary

TERADATA.

2 Key Takeaways

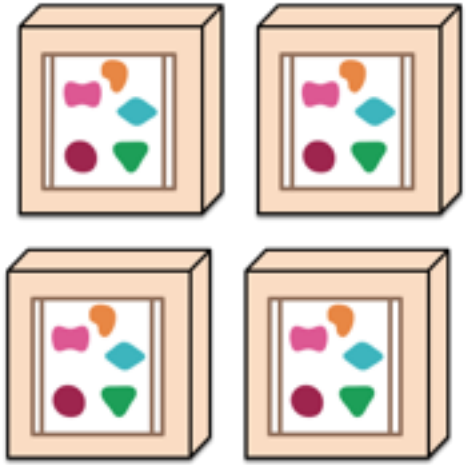
- Watch Martin Fowler's 2014 video on Microservices (last of the 3 videos)
  - Excellent pragmatic overview of the basic concepts
  - Sensible set of guidelines on when to use microservices versus monoliths

- Watch Coda Hale's 2011 talk on Metrics, Metrics Everywhere
  - The first 15 minutes are just excellent
  - All developers should "Mind the Gap" and deliver "Business Value"

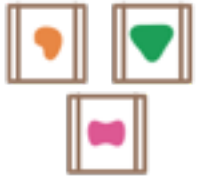
Microservices 101

A monolithic application puts all its functionality into a single process...

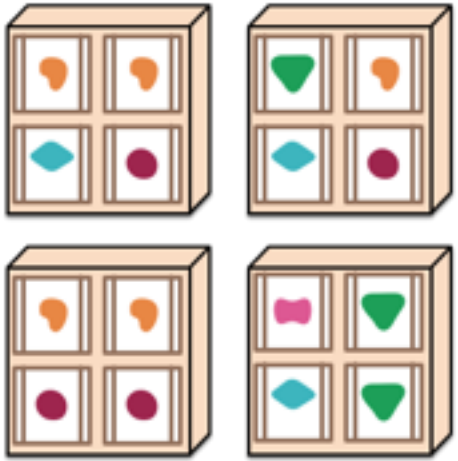
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



Microservices Summary

- What are Microservices
  - Small, and focused on doing one thing well
  - Autonomous
- Key benefits
  - Technology Heterogeneity
  - Resilience
  - Scaling
  - Ease of Deployment
  - Organizational Alignment (Conway's Law, Scrum teams, DevOps)
  - Composability
  - Optimizing for Replace-ability
- Not a "Sliver Bullet"
- **As much about "culture" as "technology"**

TERADATA.

Characteristics of a Microservice Architecture (Martin Fowler)

1. Componentization via services
2. Organized around business capabilities (DevOps)
3. Products not projects
4. Smart endpoints and dumb pipes (not EMBs!!!)
5. Decentralized governance (technology heterogeneity, teams)
6. Decentralized data management
7. Infrastructure automation (DevOps)
8. Design for failure (the Chaos Monkey!!!)
9. Evolutionary design

How did I get here?

- CTO of RainStor, a UK-based **startup**
- Working with commercial **Big Data** since 2008
- R&D using **Scrum** since 2008
- OEM business to $2M revenue by 2012: HP, Informatica, Anritsu, …
- First commercial SQL product on **Hadoop** (disputed)
- Direct sales since 2012: **AT&T, T-Mobile, Credit Suisse, and Barclays**
- 35 employees, majority in R&D
- $8M revenue at end of 2013
- **Acquired by Teradata** end of 2014
- But…

My Introduction to Microservices

- Teradata put the RainStor product into maintenance mode start of 2016
  - "Business value" decision by Teradata
  - …but a number of technical and business **mistakes were made**
  - ...our recent experience would have helped avoid some of them
- Existing R&D team formed the Teradata **IoT** Analytics team
- **4 Data Scientists** added, plus 2 off-shore developers
- Team is now working on two new products:
  - One is based upon Kafka and using **Microservices**
  - Other has **0 lines of code**, and will use **Microservices**
- Support team and PS team re-tooled as **DevOps**
- We're now into our third Sprint…

TERADATA.

So How's it Going?

- From my perspective as an experienced builder of monoliths
  - 6 Pros versus 3 Cons
  - roughly in priority order
  - …but the Cons are really, really big!

Pro: “Business Value” and “Good Enough”

- Watch Coda Hale’s video on Metrics, Metrics, Everywhere!!!
- Everyone should be focused on delivering business value
  - It should be the core part of any interaction/discussion
  - Testers and Engineers: shout if the business value is unclear!
- But only ever aim for good enough
  - Identify any parts of the product which aren’t good enough!
- Using a combination of these two concepts has significantly improved all aspects of our team’s work
  - Great acid test for story creation
  - Easy to determine when a story is done
  - Most long rambling discussions can be cut short

Pro: Clean Slate

- 2 week Sprint process delivering Docker images
- Development:
  - Java 8
  - Gradle: style checking, PMD, FindBugs, code coverage
  - GIT
  - Jenkins and Artifactory
  - Libraries: Junit, LogBack, Metrics, Vert.x (REST)
  - Code reviews every week
- Test:
  - End-to-end testing via Cucumber
  - REST Assured <- main focus for the test team
  - Performance testing TBD
  - Test reviews with developers every week

Pro: Cucumber (Behaviour-Driven Development)

Result for *ExecuteForDate* in build: 14

	Scenarios			Steps								
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Undefined	Missing	Duration	Status
ExecuteForDate	1	1	0	5	5	0	0	0	0	0	2m 23s 912ms	Passed

Feature: ExecuteForDate
Scenario: Basic execute for date
Given I have a configured system18s 414ms
And I run the segmentation engine "install" command22s 708ms
And I run the segmentation engine "loadSegments" command14s 702ms
When I run the segmentation engine "executeForDate" command with args 201602101m 19s 011ms
Then all tables and views contain records09s 075ms

TERADATA.

Con: Depth of our Technical Stack

- The basic OS:
  1. OpenStack: manage VMs
  2. Vagrant: provision VMs
  3. Ansible: configure VMs
  4. Docker: provides lightweight Ubuntu containers
  5. Mesos: distributed CPU/Memory/Disk provisioning
  6. Marathon: manages applications
- Other services
  1. Kafka: distributed messaging system
  2. ELK: logging and monitoring

Con: OpenStack on Bare Metal

- We hade lots of in-house experience with Cloud and VMs
  - how hard could it be?
- It was extremely painful to get OpenStack!
  - 6+ weeks to get happyish (1 DevOps out!)
- Should have used AWS (but it is expensive)
- Should have used Mirantis
- However, working through the pain is beginning to pay off

Con: What is going on here? How many cores?

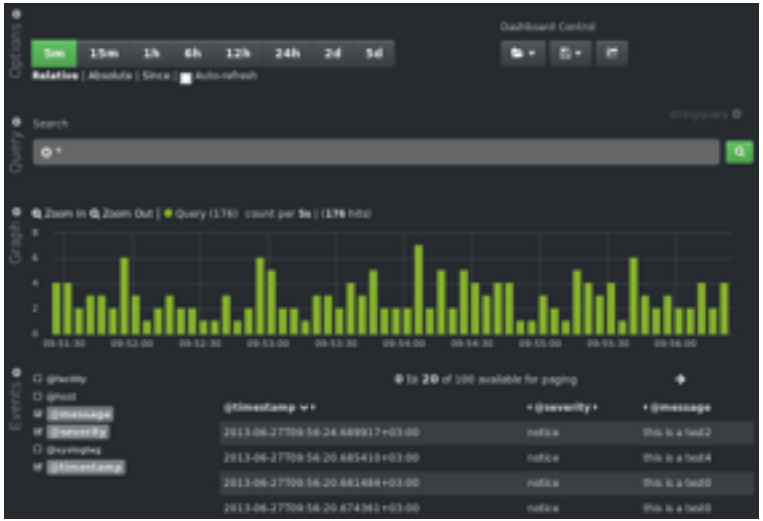
MARATHON Apps Deployments About Docs

Filter list + New App

ID	Memory (MB)	CPUs	Tasks / Instances	Health	Status
/cloud-platform/alertmanager	1024	0.5	1 / 1		Running
/cloud-platform/data-services	3072	3	3 / 3		Running
/cloud-platform/grafana	128	1	1 / 1		Running
/cloud-platform/installer-services	384	0.5	1 / 1		Running
/cloud-platform/mesos-dns	768	0.6	3 / 3		Running
/cloud-platform/pushgateway	128	1	1 / 1		Running
/cloud-platform/registry	2048	2	2 / 2		Running
/listener/app-services	768	2	2 / 2		Running
/listener/ingest-services	3600	6	3 / 3		Running
/listener/log-services	384	0.3	1 / 1		Running
/listener/router	4500	6	3 / 3		Running
/listener/system-monitor	384	0.5	1 / 1		Running
/listener/system-services	1024	2	1 / 1		Running
/listener/web	768	0.9	3 / 3		Running
/writer-13343191-300c-4e8d-820c-e0ccc7718c5f	4096	4	2 / 2		Running
/writer-d73a5899-a4d6-4a39-aff2-dfad3c1d76be	4096	4	2 / 2		Running
/writer-fe161916-4a45-49d6-94e7-047a61e4a86c	4096	4	2 / 2		Running

TERADATA

Pro: Mind the Gap!



- From Coda Hale!!!
- Design for failure
  - not just logging!
  - monitoring and alerting
  - use ELK as a minimum
- DevOps culture
  - design for supportability
  - always be thinking about install/upgrade/reconfiguration
  - always have a DevOps on each Story
- Capture and trend all metrics that relate to business value

TERADATA.

Pro: Sociology

- Parkinson's Law
  - If you give someone 8 hours to fill a bath it will take 8 hours…
  - 2 week Sprints turn this into a very real problem
  - We now combat using "Good Enough" to close a story
  - …but we now have a problem starting stories midway through a Sprint
  - => out estimating via story points is not good enough (Mind the Gap!)
- Conway's Law:
  - System architecture will always mirror the organization's structure
  - If you have 4 teams building a compiler, you will end up with a 4-pass compiler
  - Not sure if we've got this right yet…
  - Avoided splitting into functional teams – e.g. query team and an apps team
- Shared Vision (Coda Hale, yet again!)

Pro: Security Model defined for us

- HTTPS with signed certificates for REST
- LDAP for user authentication and authorization
- API tokens for collectors and end points
- (not sure about delegation)

Summary

- How do you start a Microservices project?
  - Mandate it – massively expensive
  - Already be doing it somewhere else
  - Start small: refactor a monolith into 2-3 pieces
- Always Mind the Gap
  - Keep checking that your understanding is correct
  - Most of my mistakes can be attributed to not doing this
- Culture is really, really important

2 Key Takeaways

- Watch Martin Fowler's 2014 video on Microservices (last of the 3 videos)
  - Excellent pragmatic overview of the basic concepts
  - Sensible set of guidelines on when to use microservices versus monoliths

- Watch Coda Hale's 2011 talk on Metrics, Metrics Everywhere
  - The first 15 minutes are just excellent
  - All engineers should "Mind the Gap" and deliver "Business Value"

Microservices Links

- Martin Fowler
  - 2014 video on Microservices (last of the 3 videos)
  - A definition of Microservices (9 characteristics)
- Sam Newman
  - Building Microservices, O'Reilly, February 2015

TERADATA