THE TRUTH ABOUT MICROSERVICES

John Frizelle
Mobile Platform Architect
Wednesday, May 3rd 2017
Agenda

- What are Microservices
- Challenges with Microservices
- When to consider Microservices
- Summary
“...an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are [...] independently deployable by fully automated deployment machinery.”

- Martin Fowler
"...an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are [...] independently deployable by fully automated deployment machinery."

- Martin Fowler
"The Microservices buzz is much like teen dating. You do it because it's exciting and everyone is doing it. You don't really know what you are getting into, but want to do it anyways. By the time you realize what it's all about, you're already in a relationship."

- Zohaib Khan, Red Hat
Microservice Vs Microservices Architecture

- Building a single Microservice is **EASY**
- Building a Microservices Architecture is **HARD**
CHALLENGES WITH MICROSERVICES
Challenges with Microservices Architectures

- Building
- Testing
- Versioning
- Deploying
- Logging
- Monitoring
- Debugging
- Connectivity
Microservices Challenge 1 / 8 - Building

- How to identify dependencies between services
  - One build completing may need to trigger several other builds

- Building individual services vs building a product release
  - What versions of which services constitute a “release”
Microservices Challenge 2 / 8 - Testing

● Integration testing
  ○ Mocks vs full system stand up

● End to end testing
  ○ Identifying root cause of failures
Microservices Challenge 3 / 8 - Versioning

- Public API vs versioning your microservices
  - How do the two relate

- Updating to new versions
  - Backward compatible APIs - code smell
  - Multiple versions live for different clients
Microservices Challenge 4 / 8 - Deploying

• Huge amount of Automation
  ○ Too complex for manual human deployment

• Blue / Green Deployments
  ○ Deciding when to roll back and knowing it’s worked
Microservices Challenge 5 / 8 - Logging

- Distributed Systems => Centralised Logs
  - Impossible to manage without centralization

- Tracing requests across services
  - Global request Ids are essential
Microservices Challenge 6 / 8 - Monitoring

- Critical to have centralised view of system
  - Pinpoint source of problems

- Distributed Request Tracing
  - Need to be able to track a request across services
Microservices Challenge 7 / 8 - Debugging

● Root cause analysis
  ○ How to determine where the problem is

● Remote Debugging
  ○ Not feasible across dozens or hundreds of services
Microservices Challenge 8 / 8 - Connectivity

- **Service Discovery**
  - Centralised (etcd) Vs integrated (properties)

- **Network**
  - HTTP - Circuit Breaker, dropped requests
WHEN TO CONSIDER MICROSERVICES
Challenges with Monolithic Architectures

- Size
- Stack
- Failure
- Scaling
- Productivity
Monolith Challenge 1 / 5 - Size

● Code Base Size
  ○ Can overload IDE

● Local Development
  ○ Requires huge amount of resources
Monolith Challenge 2 / 5 - Stack

- Hard to use any new tech stack
  - How / where does it fit in

- Initial jump is the hardest
  - Moving from 1 to 2 tech stacks doubles the overhead
Monolith Challenge 3 / 5 - Failure

- If anything fails, everything fails
  - It's all one system

- Much larger surface area
  - Higher likelihood of failure
Monolith Challenge 4 / 5 - Scaling

- Scale everything
  - For any contention in any part of the system

- Excessive cost and resource consumption
  - Larger footprint for CPU / Memory / Disk
Monolith Challenge 5 / 5 - Productivity

- Developers can not work independently
  - Single codebase, single deployment

- Develop test debug cycle increases
  - Compile time, run time, test execution time
Benefits of Microservices

- Agility and flexibility to change rapidly
- Smaller codebases => less context for developers
- Smaller teams => clearer focus and responsibility
- Easier to scale => only scale the parts that need it
- Right tool for the right job => pick the technology that works best
Understand Why

- Don’t get caught up in the hype
- If you do Microservices, make sure you know why you are doing them
Complex systems require automation

- Microservices don’t make complexity disappear - they just move it

- Automation is key - Microservices Architectures are too complex for humans to manage
Transformation is hard

- Moving from Monolith to Microservices is challenging and will take time
- Remember Conway’s law - moving to microservices will likely require an “Agile Transformation”
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews