Microservices + DevOps

Charlie Garrod

Michael Hilton
Administrivia

- Homework 6 checkpoint – Monday Dec 4\textsuperscript{th}
- Final Exam Review: Dec 13\textsuperscript{th}, 2-4pm Wean 5409
- Final Exam: Dec 15\textsuperscript{th}, 5:30-8:30pm Wean 7500
Last Time:

- Architectural styles
Simple Layers App
More functionality
Even more functionality
Organize our backend

Our Cool App

Frontend

Backend Server

Mod 1
Mod 3
Mod 5

Mod 2
Mod 4
Mod 6

Database
How to scale?
Monolith

- What happens when we need 100 servers?
- What if we don’t use all modules equally?
- How can we update individual models?
- Do all modules need to use the same DB, language, runtime, etc?
Microservices

Our Cool App
- Frontend
- "Dumb" Backend

NodeJS, Firebase
- Todos
  - REST service
  - Database

Google Service
- Accounts
  - REST service
  - Database

Java, MySQL
- Mailer
  - REST service
  - Database

AJAX

Search Engine
- REST service
- Database

Java, Neo4J

Analytics
- REST service
- Database

C#, SQLServer

Facebook Crawler
- REST service
- Database

Python, Firebase

15-214

Institute for Software Research
Microservices should be:

- Modelled around business domain
- Culture of automation
- Hide implementation details
- Decentralized governance
- Deploy independently
- Design for failure
- Highly observable
Microservice prerequisites

- Rapid Provisioning
- Basic Monitoring
- Rapid Application Deployment
- DevOps Culture

You must be this tall to use microservices
MICROSERVICES...

...SO HOT RIGHT NOW
Why are microservices such a big deal?
Impact on development practices

• Amazon transitioned to “two-pizza” teams
• “Full Stack” developers
• ”Devops” as a prereq
• Live testing and rollback
• Migrating from “monolith to microservices” is popular, but comes at a cost
Microservices benefits

- Strong Module Boundaries
- Independent Deployment
- Technology Diversity
Microservices overhead

For less-complex systems, the extra baggage required to manage microservices reduces productivity.

As complexity kicks in, productivity starts falling rapidly.

The decreased coupling of microservices reduces the attenuation of productivity.

But remember the skill of the team will outweigh any monolith/microservice choice.
Microservice costs

- Distribution
- Eventual Consistency
- Operational complexity
Discussion of Microservices

- Are they really “new”?
- Do microservices solve problems, or push them down the line?
- What are the impacts of the added flexibility?
- Good Architecture doesn’t fix poor low level design problems
- Beware “cargo cult”
- “If you can’t build a well-structured monolith, what makes you think microservices is the answer?” – Simon Brown
- Leads to more API design decisions
Microservice prerequisites

- Rapid Provisioning
- Basic Monitoring
- Rapid Application Deployment
- DevOps Culture

You must be this tall to use microservices
DEVOPS
Why DevOps?

• Developers and Operations don’t have the same goals
  – Devs want to push new features
  – Ops wants to keep the system available (stable, tested, etc.)
• Poor communication between Dev and Ops
• Limited capacity of operations staff
• Want to reduce time to market for new features
• Reduce “Throw it over the fence” syndrome
DevOps Definition

• “DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.”
What are implications of DevOps?

• Quality of the code must be high
  – Testing

• Quality of the build & delivery mechanism must be high
  – Automation & more testing

• Time is split:
  – From commit to deployment to production
  – From deployment to acceptance into normal production

• Goal-oriented definition
  – May use agile methods, continuous deployment (CD), etc.
  – Likely to use tools

• Achieving it starts before committing
Microservices rely on DevOps
DevOps Toolchain
DevOps Toolchain

• Code — code development and review, source code management tools, code merging
• Build — continuous integration tools, build status
• Test — continuous testing tools that provide feedback on business risks
• Package — artifact repository, application pre-deployment staging
DevOps Toolchain continued

- Release — change management, release approvals, release automation
- Configure — infrastructure configuration and management, Infrastructure as Code tools
- Monitor — applications performance monitoring, end-user experience
DevOps Toolchain - Code

- **Code development** and review
- Source code management tools
- Code merging
DevOps Toolchain - Code

- Code development and review
- Source code management tools
- Code merging

More on Code Review in 17-313
DevOps Toolchain - Code

- Code development and review
- Source code management tools
- Code merging

![git+github](image-url)
DevOps Toolchain - Code

- Code development and review
- Source code management tools
- Code merging
DevOps Toolchain - Build

- Continuous integration tools
- Build status
DevOps Toolchain - Test

• Continuous testing tools that provide feedback on business risks
DevOps Toolchain - Test

- Continuous testing tools that provide feedback on business risks

Testing tools must have tests to be valuable!!