Please ask questions via the mobile app!
Let’s say I wanted to build a large application
Some requirements
Some requirements

- Must scale to millions of users
Some requirements

- Must scale to millions of users
- Support multiple platforms
Some requirements

- Must scale to millions of users
- Support multiple platforms
- Handle complex business rules
Some requirements

- Must scale to millions of users
- Support multiple platforms
- Handle complex business rules
- Competitive in a fast moving market
Some requirements

- Must scale to millions of users
- Support multiple platforms
- Handle complex business rules
- Competitive in a fast moving market
  - React Quickly
Some requirements

- Must scale to millions of users
- Support multiple platforms
- Handle complex business rules
- Competitive in a fast moving market
  - React Quickly
  - Out-innovate
A product like this
A product like this
A product like this

› 75+ Million Monthly Active Users
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- > 20,000 songs added per day
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- > 20,000 songs added per day
- > 2B playlists
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- >20,000 songs added per day
- >2B playlists
- Incredibly complex business rules
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- >20,000 songs added per day
- >2B playlists
- Incredibly complex business rules
- Lots of competition
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- >20,000 songs added per day
- >2B playlists
- Incredibly complex business rules
- Lots of competition
A product like this

- 75+ Million Monthly Active Users
- 58 Countries
- >20,000 songs added per day
- >2B playlists
- Incredibly complex business rules
- Lots of competition
How do you support these requirements while moving fast and innovating?
Solution

Autonomous full-stack teams
Autonomous

*adjective*

au·ton·o·mous - \\ə-'tä-nə-məs\\

(of a country or region) having the freedom to govern itself or control its own affairs.
"the federation included sixteen autonomous republics"

having the freedom to act independently.
"school governors are legally autonomous"

synonyms: *self-governing, independent, sovereign, free, self-ruling, self-determining, autarchic; self-sufficient*

"an autonomous republic"
Autonomous

*adjective*

au·ton·o·mous - ˈō-tä-nə-məs

(of a country or region) having the freedom to govern itself or control its own affairs.
"the federation included sixteen autonomous republics"

**having the freedom to act independently.**
"school governors are legally autonomous"

synonyms: *self-governing, independent, sovereign, free, self-ruling, self-determining, autarchic; self-sufficient*

"an autonomous republic"
Server

Core Library

Platform
Platform
Platform
Platform

Server

Infrastructure
Challenges
Challenges

Synchronization
Challenges

Synchronization

Client UX implementation
Challenges

Synchronization

Client UX implementation depends on
Challenges

Synchronization

Client UX implementation depends on Core Library Implementation
Challenges

Synchronization

Client UX implementation depends on
Core Library Implementation depends on
Challenges

Synchronization

- Client UX implementation
- Core Library Implementation
- Server Implementation

depends on
depends on
Challenges

Synchronization

Client UX implementation
Core Library Implementation
Server Implementation
Challenges

Synchronization

Client UX implementation depends on
Core Library Implementation depends on
Server Implementation depends on
Infrastructure Implementation
What this looks like at Spotify
What this looks like at Spotify

- 90+ teams
What this looks like at Spotify

- 90+ teams
- 600+ Developers
What this looks like at Spotify

- 90+ teams
- 600+ Developers
- 5 Development offices
What this looks like at Spotify

‣ 90+ teams
‣ 600+ Developers
‣ 5 Development offices
‣ 1 Product
Full-stack autonomous teams

Requires you to structure your application in loosely coupled parts
Services
Microservices yay!

Easier to scale based on real-world bottlenecks
Microservices yay!

Easier to test
Microservices yay!

Easier to deploy
Microservices yay!

Easier to monitor
Microservices yay!

Can be versioned independently
Microservices yay!

Are less susceptible to large failures
Microservices boo!

are harder to monitor
Microservices boo!

need good documentation / discovery tools
Microservices boo!

create increased latency
I see you have a poorly structured monolith. Would you like me to convert it into a poorly structured set of microservices?
What does this look like at Spotify?
What does this look like at Spotify?

- 810 active services
What does this look like at Spotify?

- 810 active services
- ~10 Systems per squad
What does this look like at Spotify?

- 810 active services
- ~10 Systems per squad
- ~1.7 Systems per person with access to production servers
What does this look like at Spotify?

- 810 active services
- ~10 Systems per squad
- ~1.7 Systems per person with access to production servers
- ~1.15 Systems per member of Technology
http://spotify.github.io/apollo/

SPOTIFY APOLLO OPERATIONS MANUAL

A LIBRARY FOR WRITING SERVICES THAT FOCUSES ON COMPOSABILITY AND SIMPLICITY, WITH HIGH PERFORMANCE USING MODERN JAVA IDIOMS AND FEATURES.

SECTION I GENERAL DESCRIPTION

APOLLO
Apollo is a set of Java libraries that we use at Spotify when writing micro-services. Apollo includes features such as an HTTP server and a URI routing system, making it trivial to implement RESTful services.

WARNING
Open-source Apollo is still in development and only release candidates are available. The API and documentation might change prior to the stable 1.0.0 open source release.

Apollo has three main parts:

APOLLO API
The apollo-api library defines the interfaces for your request routing and request/reply handlers.

APOLLO CORE
The apollo-core library manages the lifecycle (loading, starting, and stopping) of your service and defines a powerful module system for adding functionality to an Apollo assembly. You do not usually need to interact directly with apollo-core; think of it merely as "plumbing."

APOLLO HTTP SERVICE
The apollo-http-service library is a standardized assembly of Apollo modules. It incorporates both apollo-api and apollo-core and ties them together with several other useful modules to get a complete microservice running.
Questions

twitter: @kevingoldsmith

(yes, we’re hiring)
Please evaluate this talk via the mobile app!

Engage