ABOUT ME

▸ Working with clients to migrate legacy monoliths -> microservices

▸ Worked on the Spring Cloud Services team

▸ Trying not to be a Chief Principal Senior Consultant Scientist
ABOUT YOU

- Have a monolith that needs migrating?
- Actively migrating a monolith?
- Building microservices net new?
- This talk has a little bit for all of you...
AGENDA

- Where did these ideas come from?
- Which comes first?
- What is the path?
BACKGROUND
Barinek
@barinek

📍 Boulder, CO
📅 Joined October 2008

https://twitter.com/barinek
HTTP://WWW.APPCONTINUUM.IO/
Simon Brown
@simonbrown

Author (leanpub.com/b/software-arc...), award-winning speaker, trainer, coder, creator of C4 software architecture model, founder of @structurizr.

TWEETS 17.3K
FOLLOWING 1,893
FOLLOWERS 8,235
LIKES 2,563

If you’re looking for information about my #c4model ... this is a good place to start ->

https://twitter.com/simonbrown
Well-defined, in-process components is a stepping stone to out-of-process components (i.e. microservices)

- High cohesion
- Low coupling
- Focussed on a business capability
- Bounded context or aggregate
- Encapsulated data
- Substitutable
- Composable

-> From components to microservices

<- All of that plus
- Individually deployable
- Individually upgradeable
- Individually replaceable
- Individually scalable
- Heterogeneous technology stacks
NOTICE ANY SIMILARITIES?
THIS SHOULD NOT BE REVOLUTIONARY
WHICH COMES FIRST?
any decent answer to an interesting question begins, "it depends..."
MONOLITH OR MICROSERVICES
IT DEPENDS...
Modular Monolith

Well-defined, in-process components is a stepping stone to out-of-process components (i.e. microservices)

- High cohesion
- Low coupling
- Focused on a business capability
- Bounded context or aggregate
- Encapsulated data
- Substitutable
- Composable

From components to microservices

<- All of that plus

- Individually deployable
- Individually upgradeable
- Individually replaceable
- Individually scalable
- Heterogeneous technology stacks
IF YOU CAN’T BUILD A WELL STRUCTURED MONOLITH, WHAT MAKES YOU THINK YOU CAN BUILD A WELL STRUCTURED SET OF MICROSERVICES?
WINNER?

WELL STRUCTURED MONOLITH
BOUNDARIES?
BOUNDDED CONTEXTS
Domain-Driven Design
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Implementing Domain-Driven Design
Vaughn Vernon
Foreword by Eric Evans

2003 2013
SOUND FAMILAR?
HOW DO I GET THERE?
GOALS

- Migrate a monolith that was in production, making money, to a more sustainable solution for the future.
- Architect for scalability of app in the future.
- Allow multiple teams to deliver business value quickly.
- Allow for addition of resources without overcrowding.
CURRENT STATE

- API servers
- Multiple clients, including JavaScript front end
- Some dead code
- Lack of comprehensive, current API tests
WHERE DO WE START?
STEPS 1: API LEVEL TESTS

GOAL: Validate you don't break any client facing behavior while migrating
CURRENT STATE

CLIENT -> API

API <- CLIENT

CLIENT <- API

CLIENT <- API
DESIRED END STATE
WHAT STAYED THE SAME?
TESTS

API

TEST

TEST

TEST

TEST
Enables consumer driven contract testing, providing a mock service and DSL for the consumer project, and interaction playback and verification for the service provider project.
RESULT: YOU NOW KNOW WHEN YOU HAVE BROKEN USER FACING BEHAVIOR BEFORE YOU GO INTO PRODUCTION
GOAL: BEGIN TO UNDERSTAND WHAT BOUNDED CONTEXTS EXIST

STEP 2: ARRANGE APPLICATION SO YOU CAN SEE YOUR DOMAIN
tree app

app
|-- controllers
   |-- application_controller.rb
   |-- orders_controller.rb
   |-- users_controller.rb
|-- helpers
   |-- application_helper.rb
   |-- order_helper.rb
   |-- user_helper.rb
|-- models
   |-- order.rb
   |-- user.rb

3 directories, 8 files
tree src/main/java/

com
  example
    myApplication
      WebApplication.java
      order
        Order.java
        OrderController.java
  user
    User.java
    UserController.java

5 directories, 5 files
STEP 2: ARRANGE APPLICATION SO YOU CAN SEE YOUR DOMAIN

- Minimizes the number of directories where changes happen.
- Less costly to experiment with/evolve bounded contexts.
- Delaying architecture decisions.
STEP 3: BREAK OUT COMPONENTS

GOAL: GIVE MORE SPACE IN THE CODEBASE AND SOLIDIFY DEPENDENCIES/BOUNDARIES.
WHAT ABOUT DATABASES?

- Application manages?
- Component manages?
- “It depends...”
RULE: MIGRATIONS ONLY TOUCH ONE TABLE.
WHY?
STEP 3: BREAK OUT COMPONENTS

- More room in the codebase for multiple teams.
- Strict boundaries between framework code and domain code.
- Moving closer to microservices without the overhead of multiple processes.
- Stopping here gives you a lot of benefits without overhead of microservices.
STEP 4: PROMOTE YOUR FIRST MICROSERVICE

GOAL: IT DEPENDS . . .
WHY EXTRACT A MICROSERVICE?

- Scaling a certain bounded context becomes an issue.
- You want to deploy one bounded context more frequently.
- Many, many other reasons that I won’t cover here.
STEP 4: EXTRACT YOUR FIRST MICRO SERVICE

- Congratulations you now have a distributed system!!!
- And with that distributed system come new costs:
  - Billing now needs to know how to contact the email service.
  - Network communication between billing and email will fail at some point.
SERVICE DISCOVERY

GOAL: ABSTRACT LOCATION AND QUANTITY OF SERVICE INSTANCES
SERVICE DISCOVERY

- Service calls are most loosely coupled to location and number of instances.
  - Client side load balancing can reduce number of network calls.
- But there is an additional applications to run and monitor.
CIRCUIT BREAKER

GOAL: PROTECT AGAINST CASCADING FAILURES THAT CAUSE SYSTEM DOWNTIME
CIRCUIT BREAKERS

- Increased resiliency of system when the network or a service fails...because they will.
- Increased visibility of health of system as a whole.
- But there is an additional applications to run and monitor.
CONGRATULATIONS YOU ARE THE PROUD OWNER OF A SET OF MICROSERVICES!!
NEXT STEPS

- Break out more microservices as necessary.
- Change up the communication interface...maybe use a message queue?
- Don't do anything and continue to make money.
- The choice is up to you!
SOURCE CODE

- https://github.com/mikegehard/journeyFromMonolithToMicroservices (Java)
- https://github.com/mikegehard/user-management-evolution-kotlin (Kotlin)
THANK YOU!
QUESTIONS?

@MIKEGEHARD, @PIVOTAL