# What has SoundCloud learnt about microservices?

Phil Calçado
SoundCloud

Please ask questions via the mobile app!

# Engage

SOUNDCLOUD

# >11 hours of audio uploaded every minute

# ~300 million people every month

The White House

ladygaga

nineinchnails

Pennywise

Snoop Dogg

Skrillex

GreenDay

Madonna

TheNational

Two Door Cinema Club

# What we did in 2012-13

# SoundCloud.com

Sounds & Sets

Social Graph

Activity Stream

Premium Features

Search

API

Social Graph

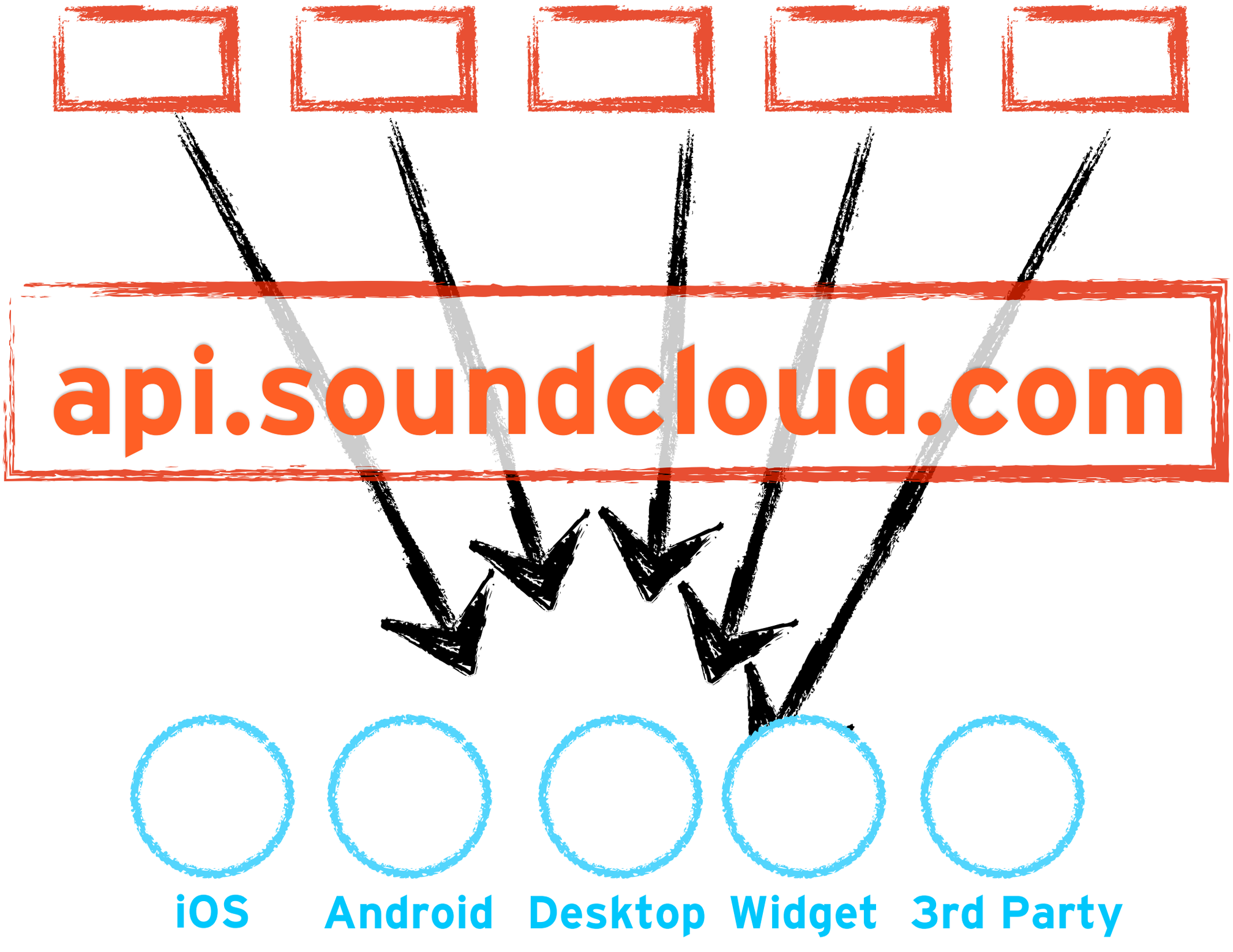
API

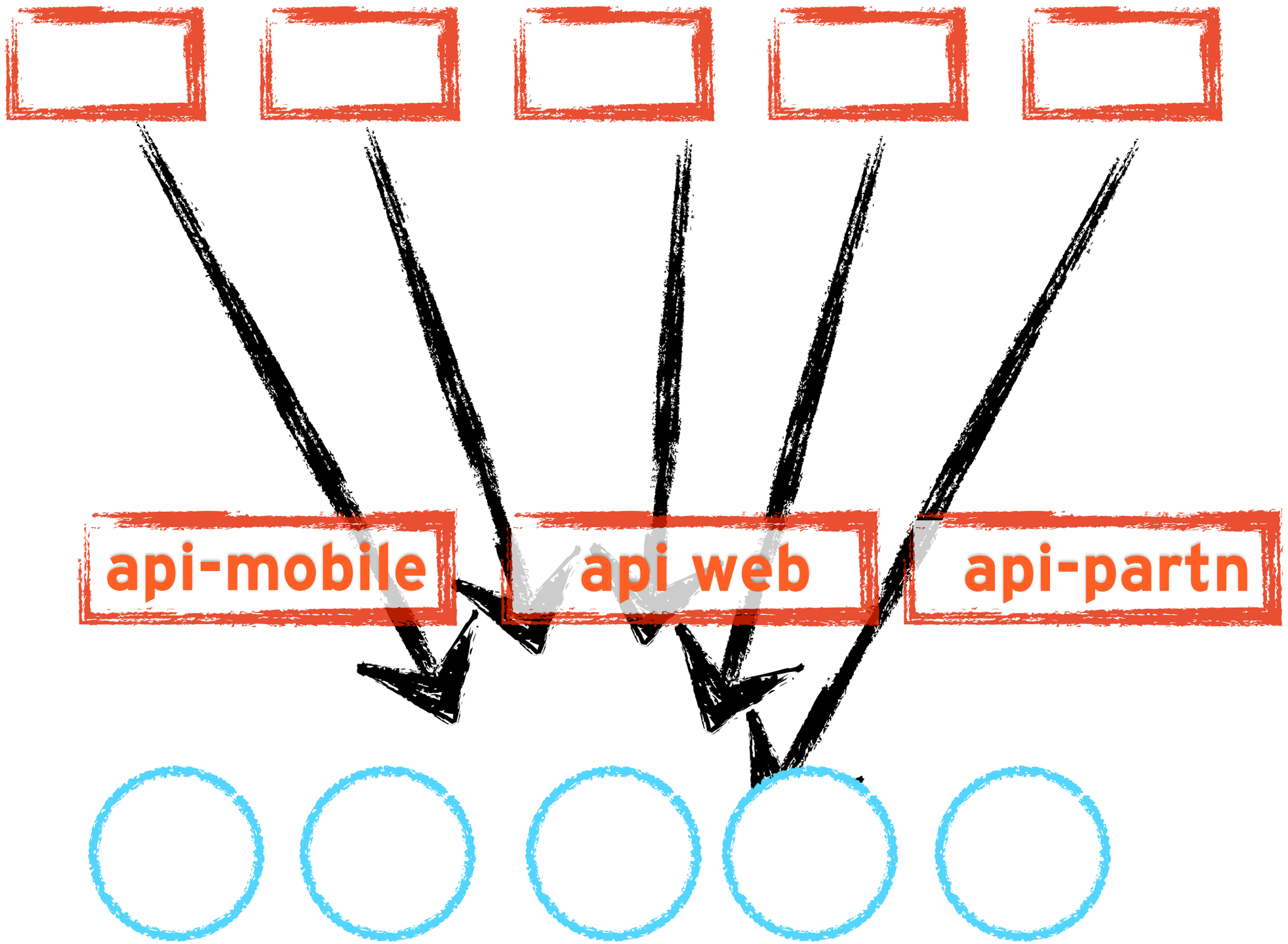
Sounds & Sets

Activity Stream

Premium Features

Search


api.soundcloud.com
iOS
Android
Desktop
Widget
3rd Party


api-mobile
api web
api-partn
iOS
Android
Desktop
Widget
3rd Party

## More details

http://bit.ly/dealing-with-the-monolith

# minimise your fixed cost per app

#1

every new app answers

# What language?

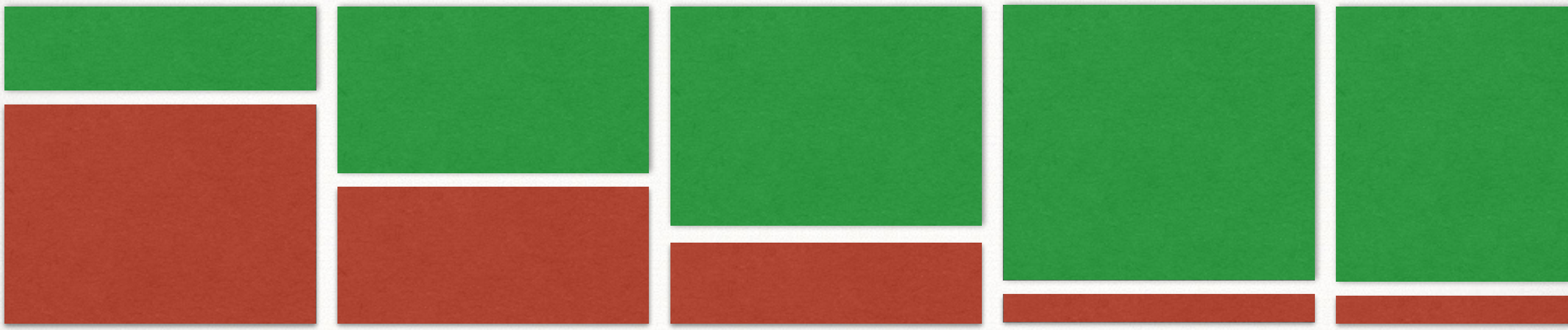
# What build system?

# What I/O lib?

# How do we deploy?

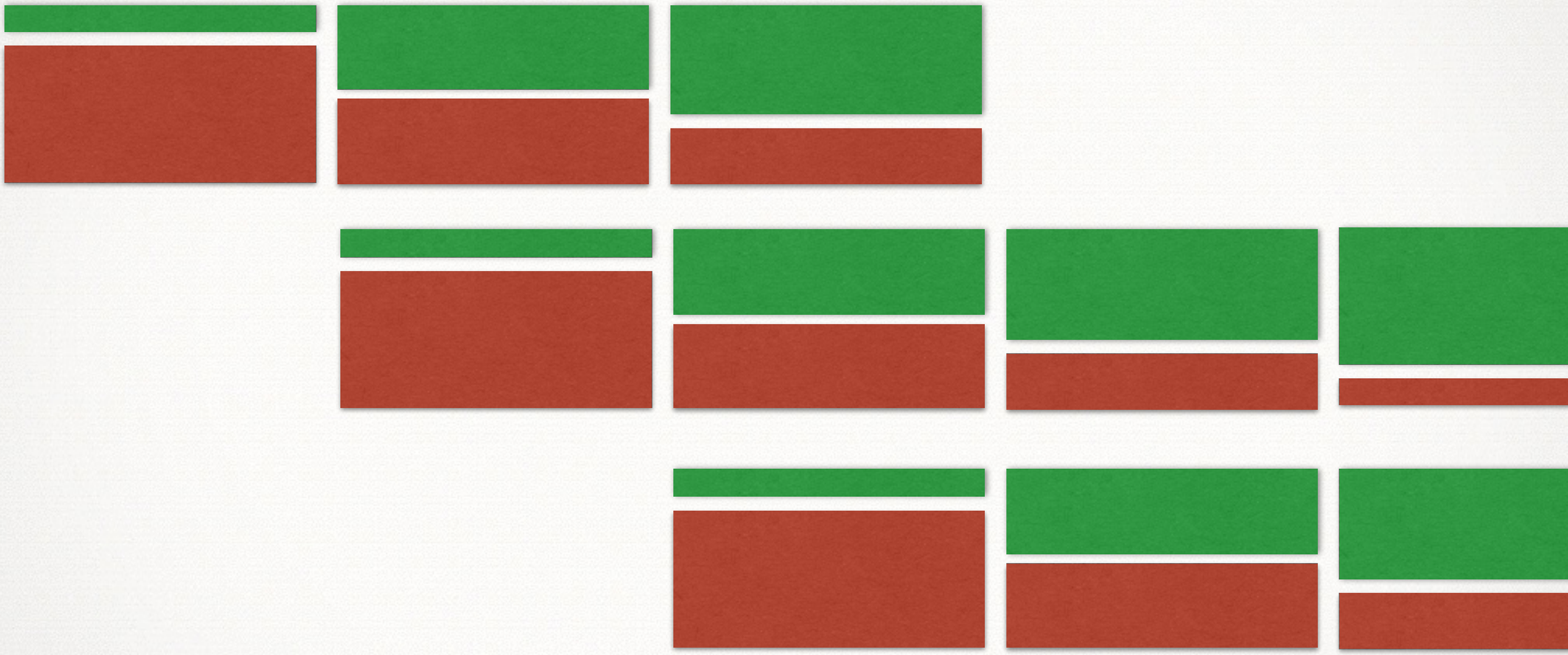
# What do we monitor?

# typical agile project




technical tasks
user stories

# typical microservices project



iteration 0 iteration 1 iteration 2 iteration 3 iteratio

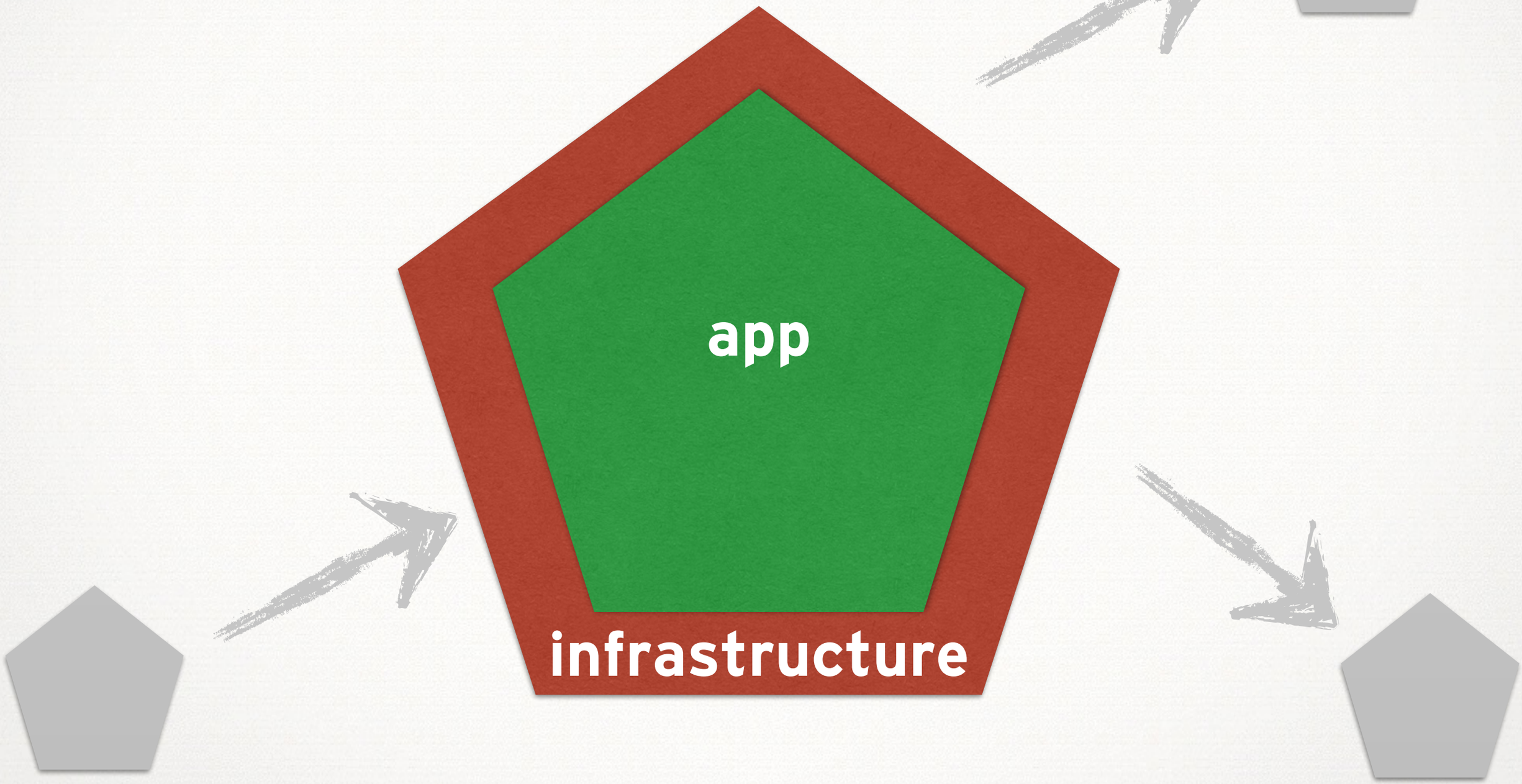
# You will have lots of microservices

# Consider having a “standard”, well-supported and documented stack

# outsource as much as possible

#2

# micro service x-ray



# Avoid building as much " infrastructure as possible

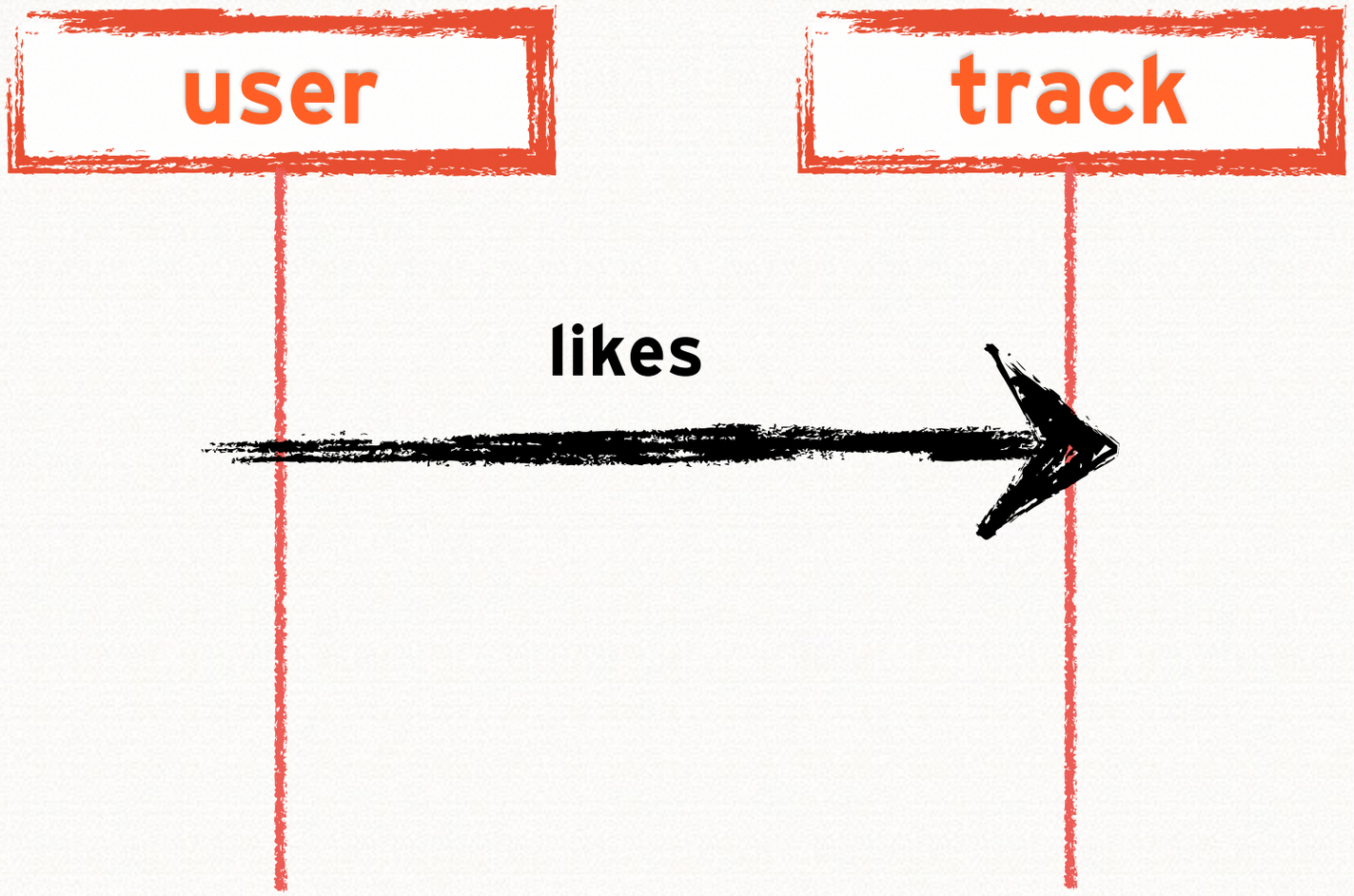
# some good options

Only infrastructure we need to build is the glue between the lib and our quirks

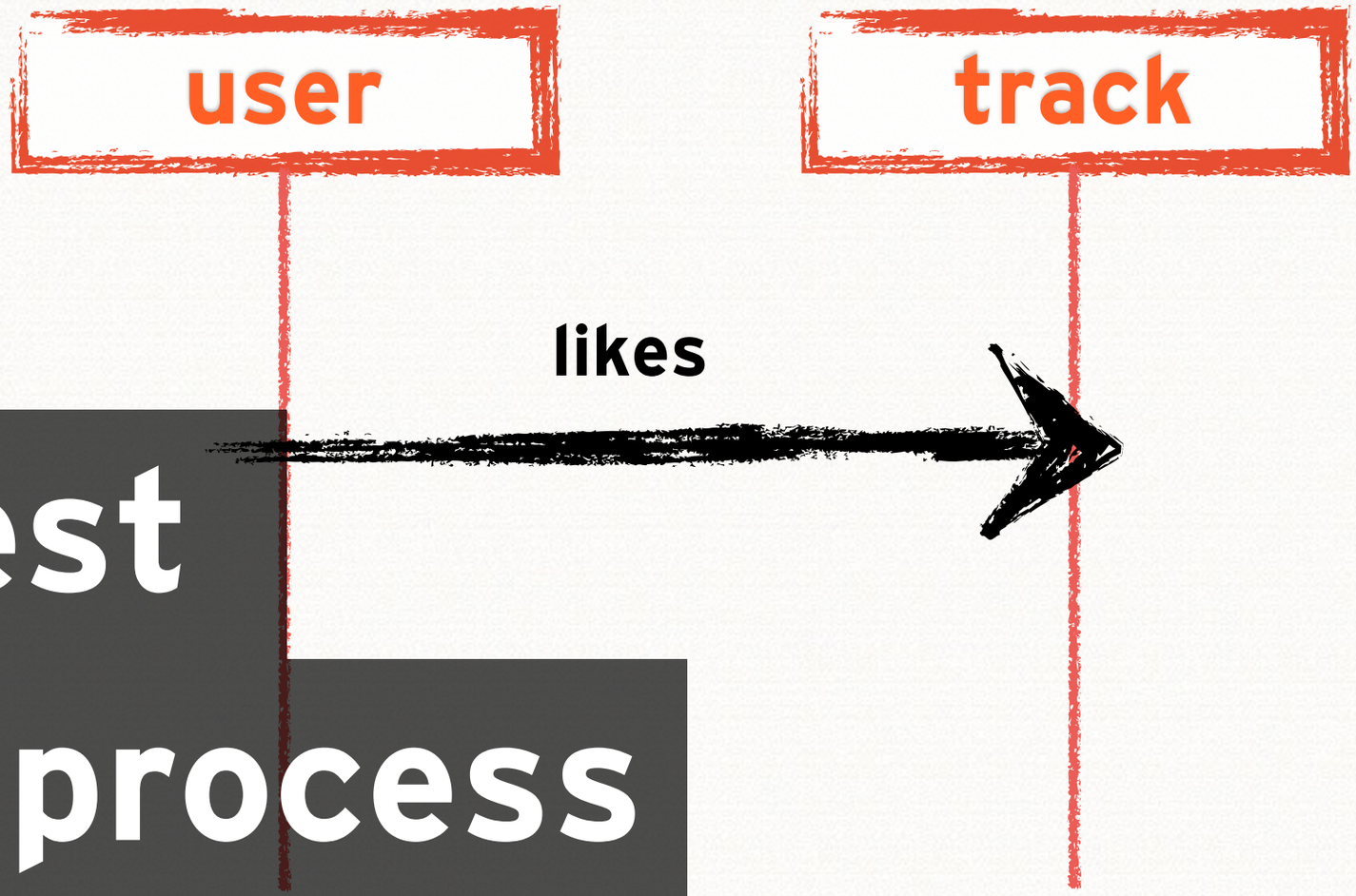
# Avoid having quirks

# acknowledge the new complexity

#3


user
track
likes

# typical app

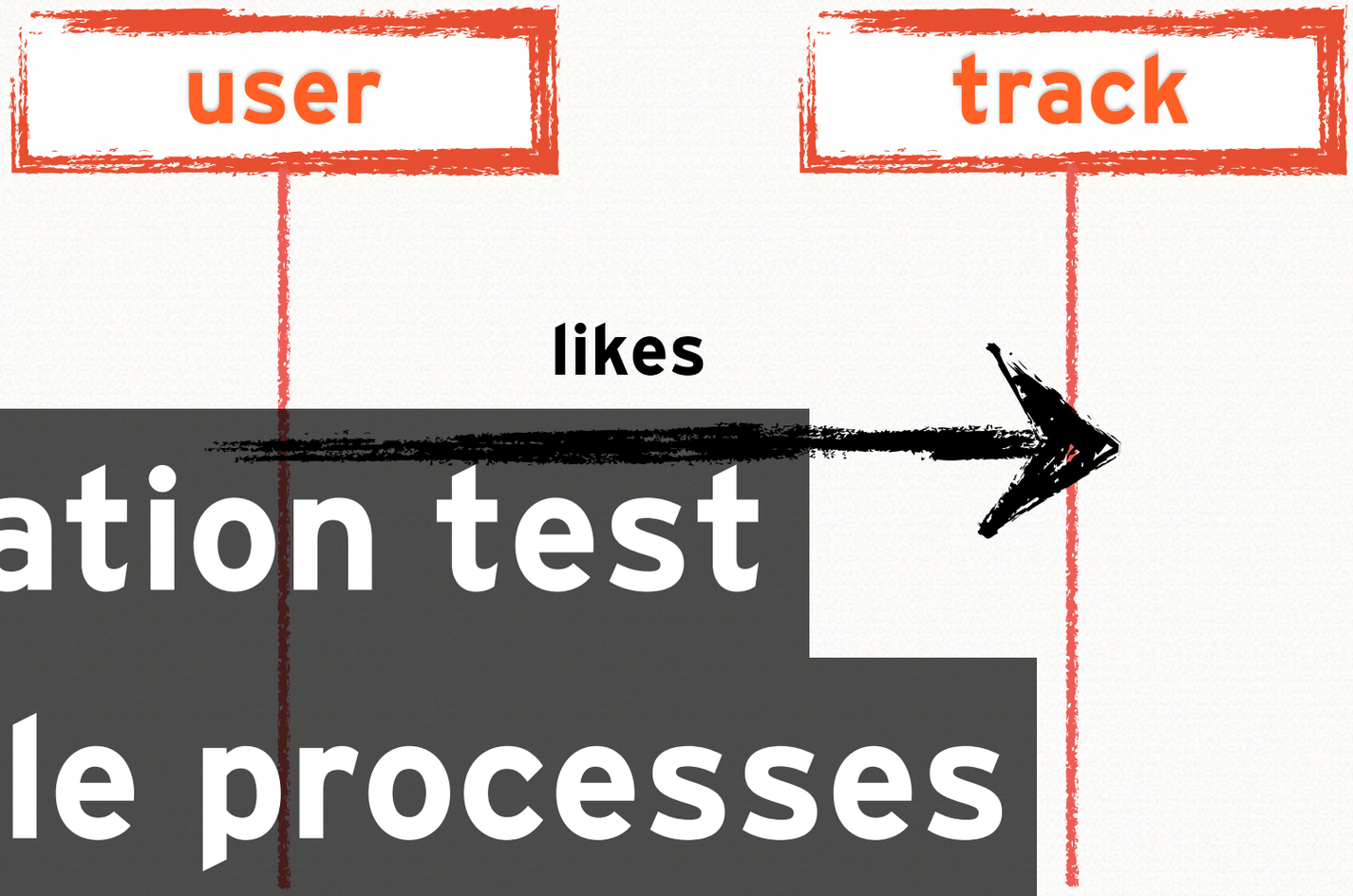



# unit test

# single process

# run inside an IDE

# microservice

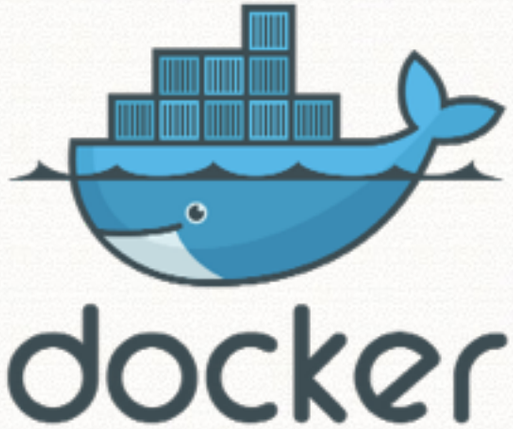



# integration test

# multiple processes

# stubs and fakes

# must be easy to provision



# make everything visible

#4

**adrian cockroft**
@adrianco

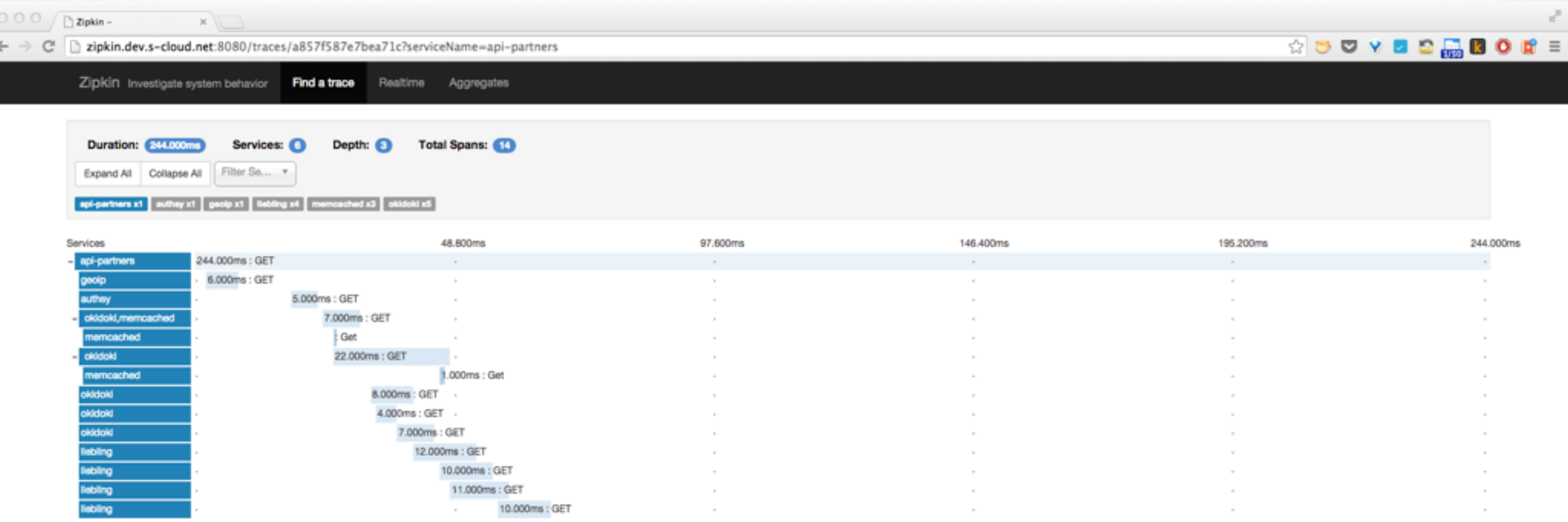
Following

Twitter microservices map looks just like the Netflix one. We called this the "Death Star" diagram. #qconlondon

RETWEETS 38 FAVORITES 31

11:30 AM - 7 Mar 2014

# make interactions visible

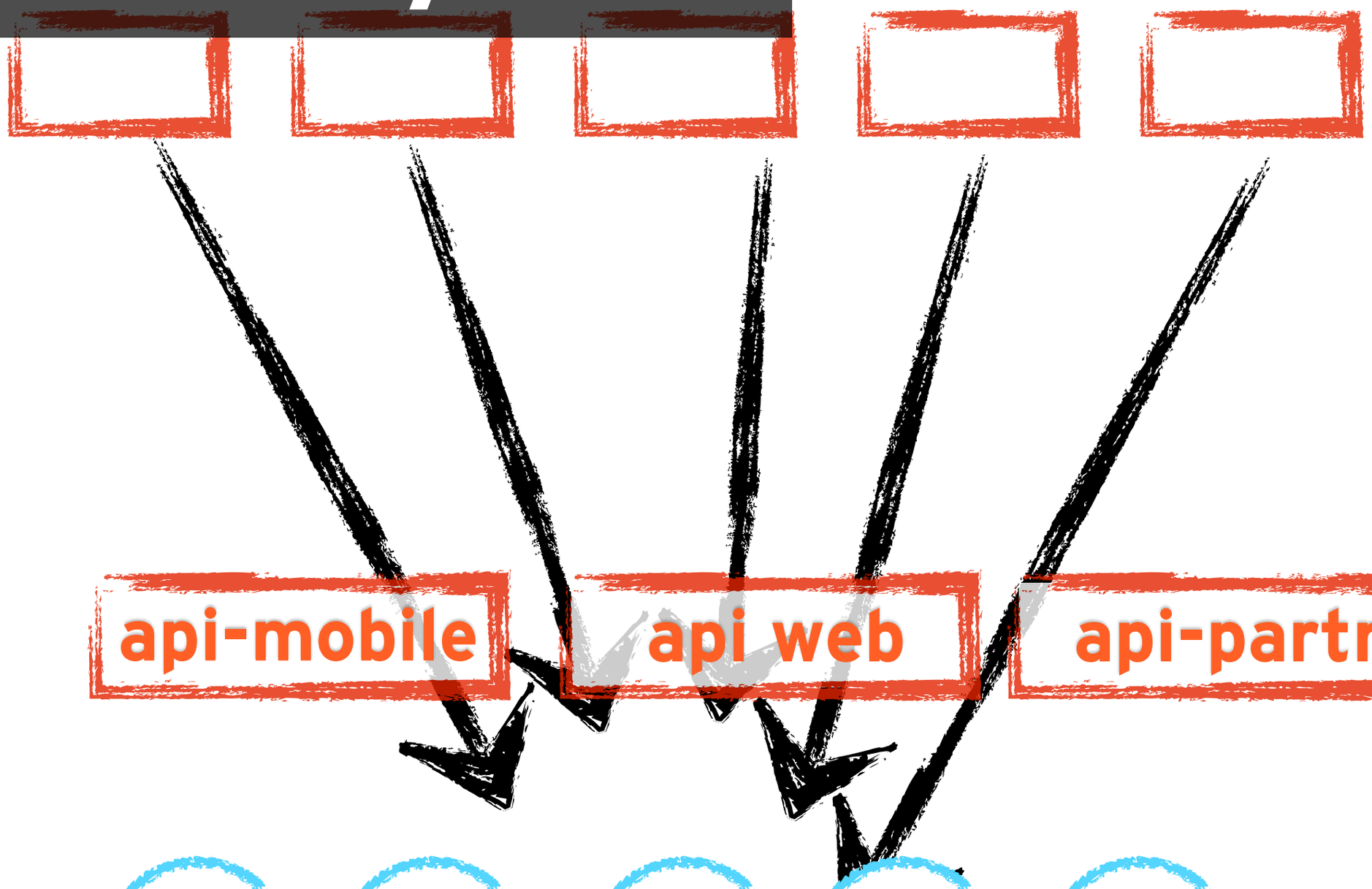


# standardised dashboards

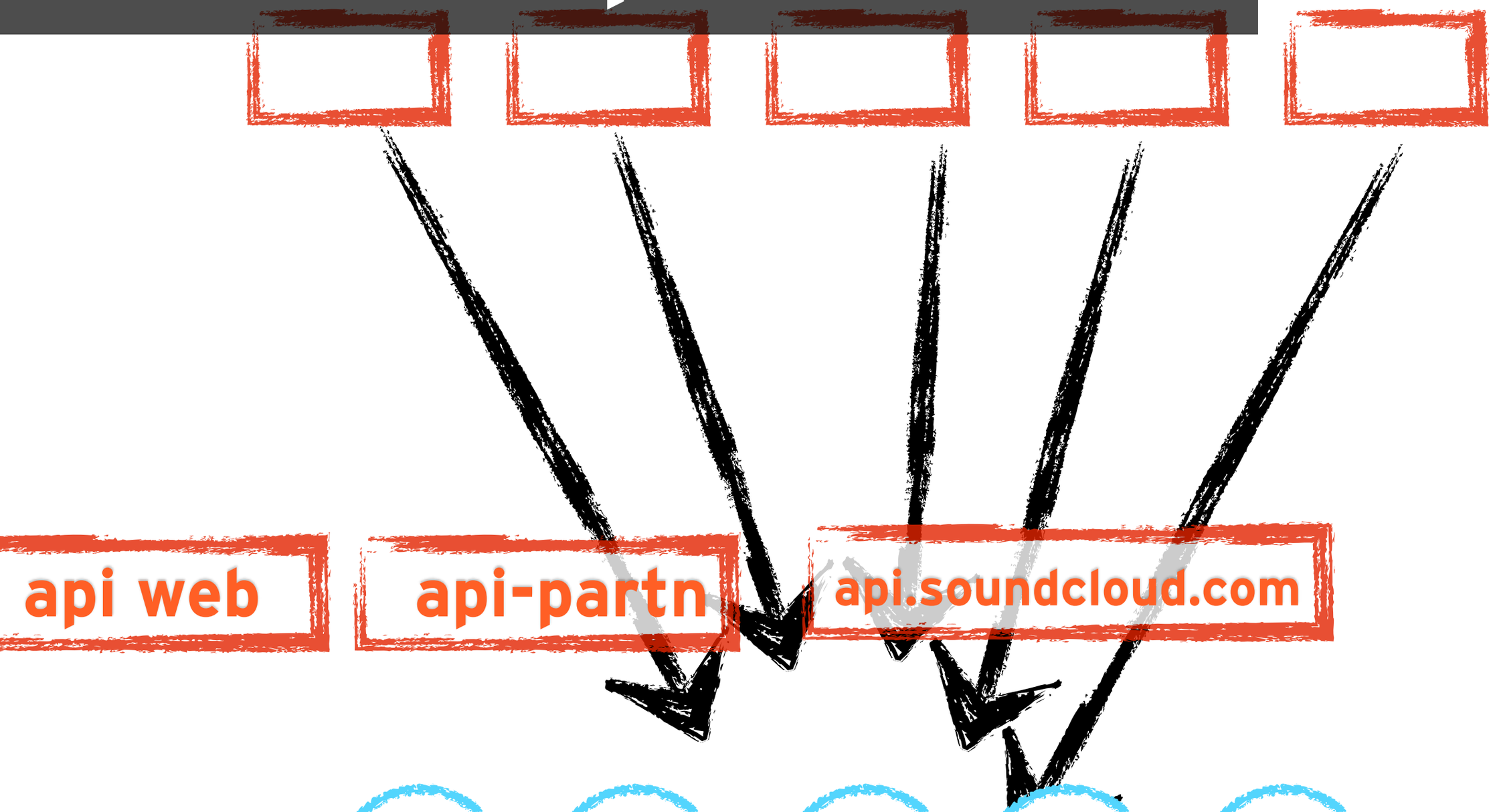
# wrap your legacy with stranglers

#5

# not exactly true



# mothership still alive

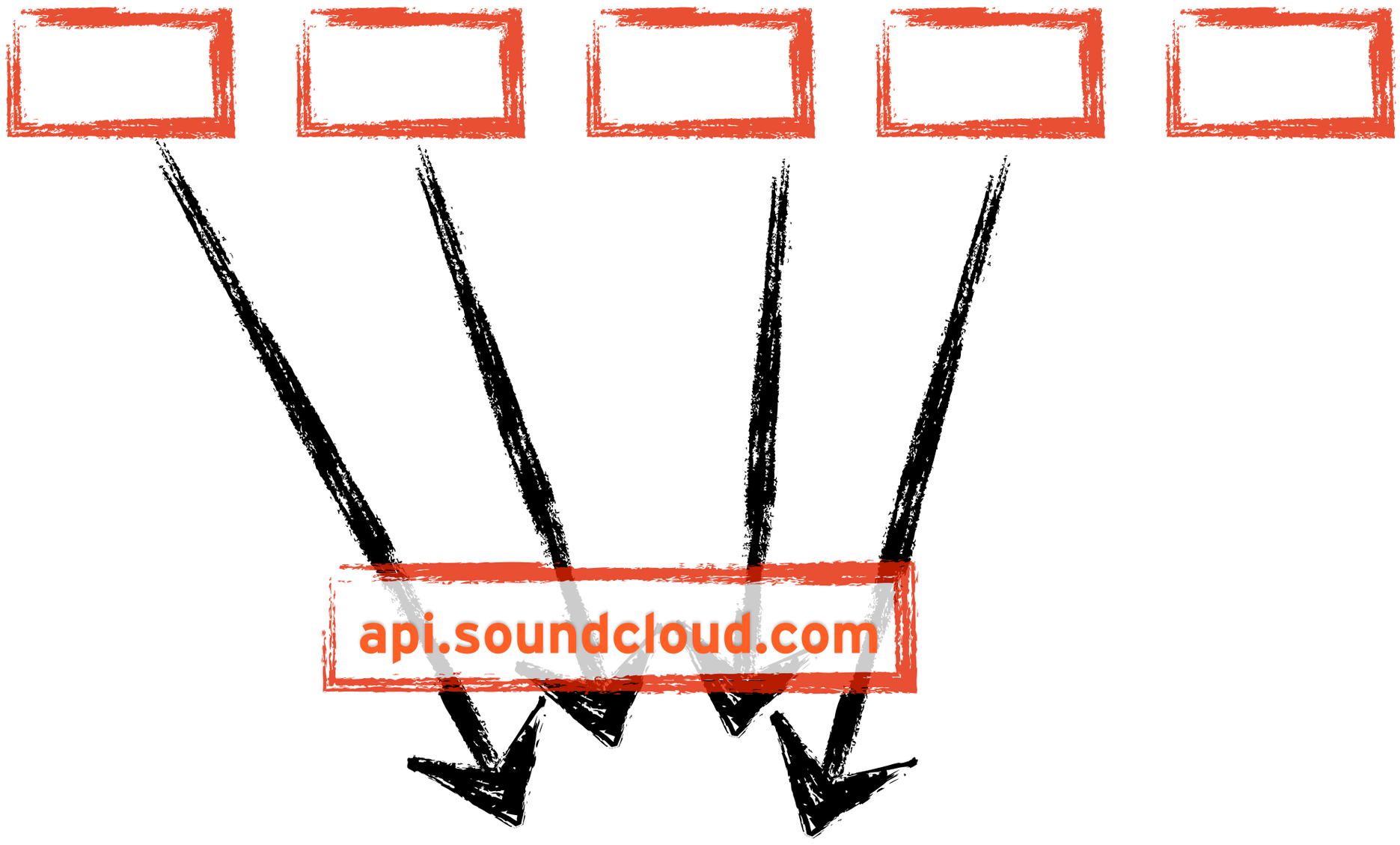


# StranglerApplication

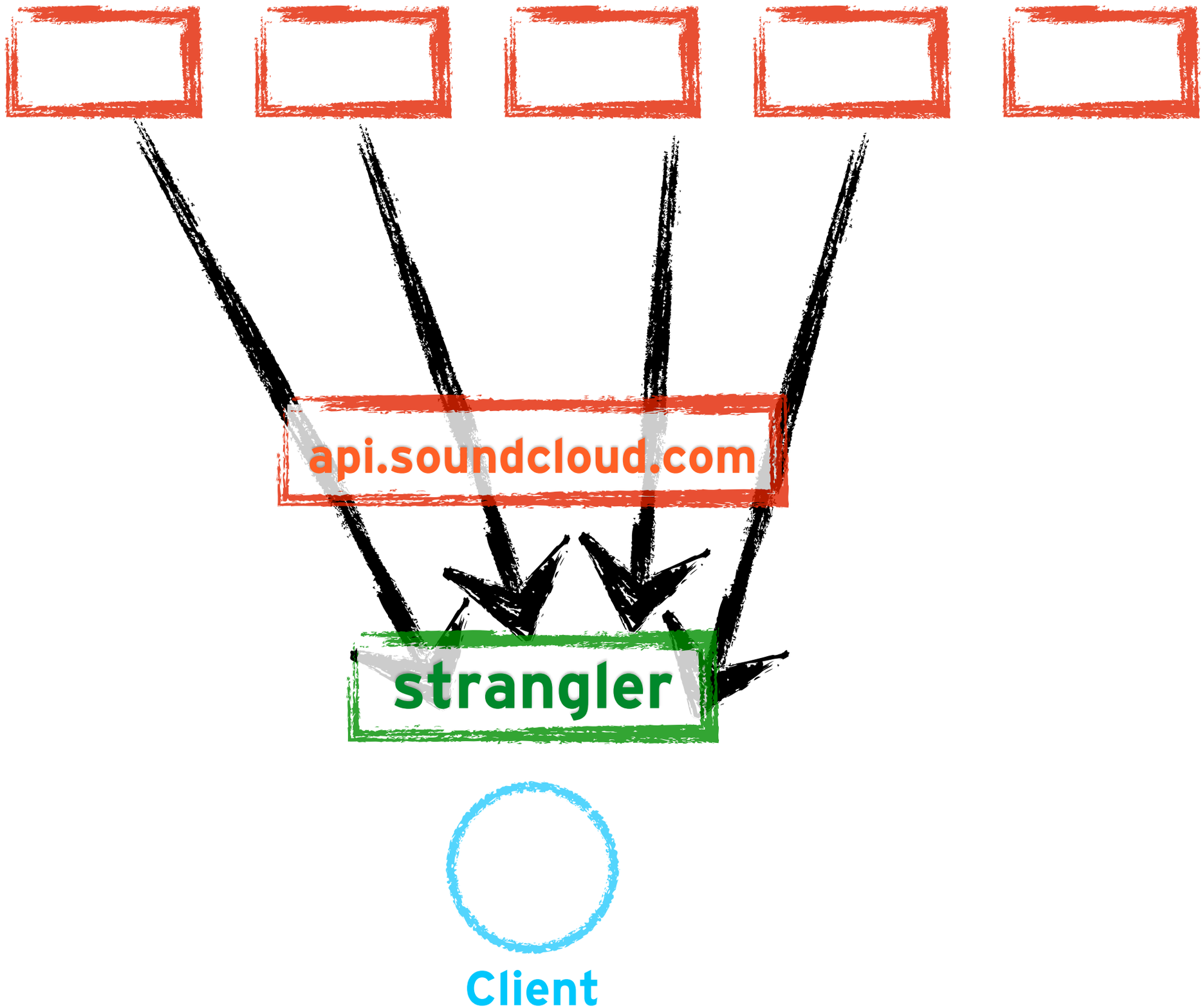
*Martin Fowler*
*29 June 2004*

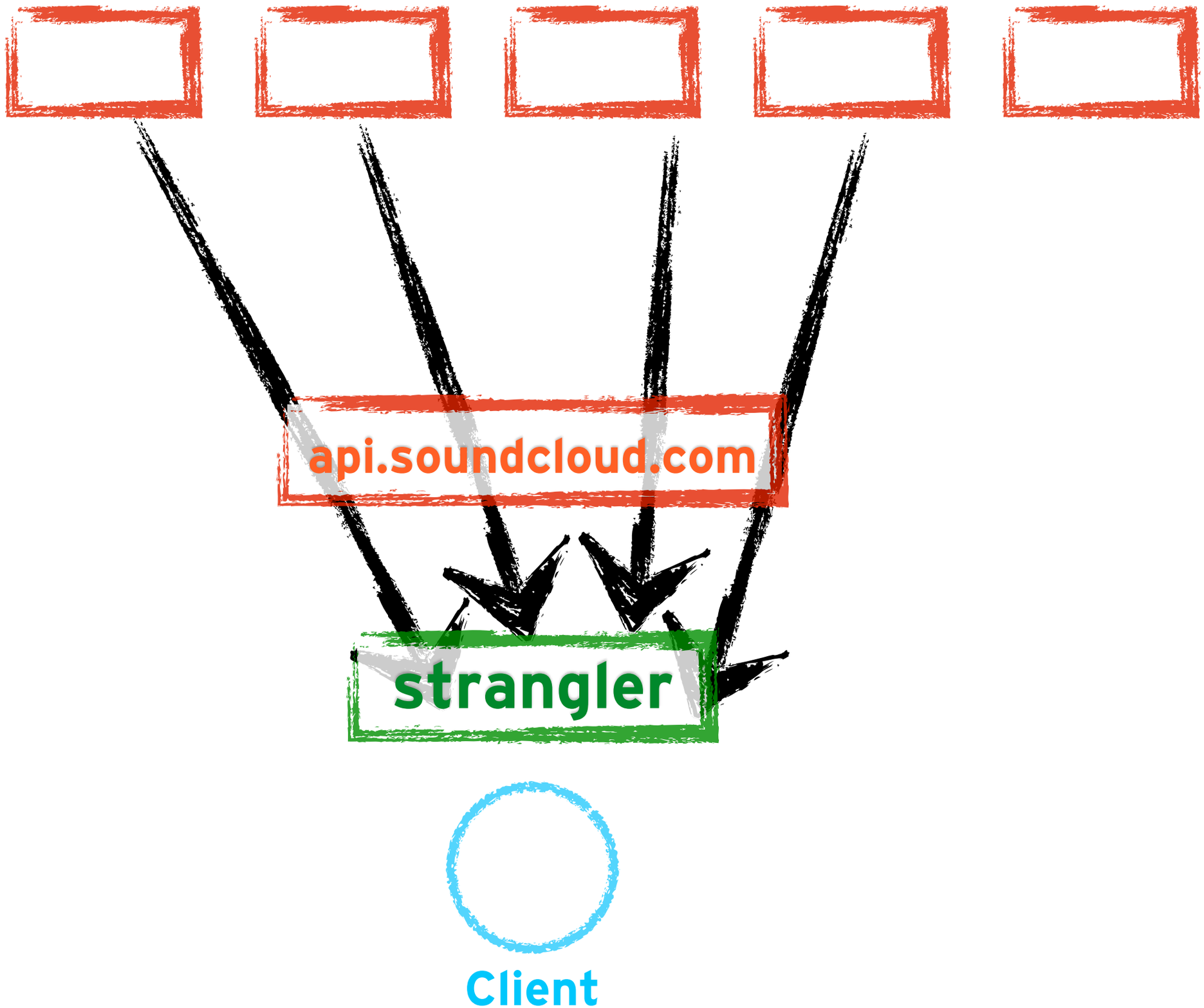
When Cindy and I went to Australia, we spent some time in the rain forests on the Queensland coast. One of the natural wonders of this area are the huge strangler vines. They seed in the upper branches of a fig tree and gradually work their way down the tree until they root in the soil. Over many years they grow into fantastic and beautiful shapes, meanwhile strangling and killing the tree that was their host.

This metaphor struck me as a way of describing a way of doing a rewrite of an important system. Much of my career has involved rewrites of critical systems. You would think such a thing as easy - just make the new one do what the old one did. Yet they are always much more complex than they seem, and overflowing with risk. The big cut-over date looms, the pressure is on. While new features (there are always new features) are liked, old stuff has to remain. Even old bugs often need to be added to the rewritten system.




Client


api.soundcloud.com
strangler
Client


api.soundcloud.com
strangler
Client

api.soundcloud.com

strangler

Client

# what changed changed since 2013?

BERLIN
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2013
goto; conference
Training:Oct 15-16 / Conference:Oct 17-18

**Presentation: APIs: The Problems with Eating your Own Dog food**

Tweet

Track: Modern Web Architecture / Time: Thursday 13:20 - 14:10 / Location: Estrelsaal C5/6

SoundCloud's web and mobile properties are all built on top of our Public API. While building the user-interface atop a RESTful layer has proven itself a sound decision, the one-size-fits-all nature of a Public API is not ideal.

When it comes to data transfer and HTTP resource modelling, each client has their own needs, and in the end hacks and workarounds have to be implemented in both clients and servers. Feature development also becomes complicated, with coordination between multiple teams required for every single little feature.

SoundCloud is now moving to a different model, where clients have their own façade APIs, modelled after their core characteristics and needs. We are also using the architecture to break away from Conway's Law and building more cross-functional, end-to-end teams.

To implement this, a lot of change in our architecture, tech stack and development processes were required. In this talk we will explore the challenges we had, the options we investigated and how ultimately SoundCloud decided to move forward.

# Stop babysitting Rails

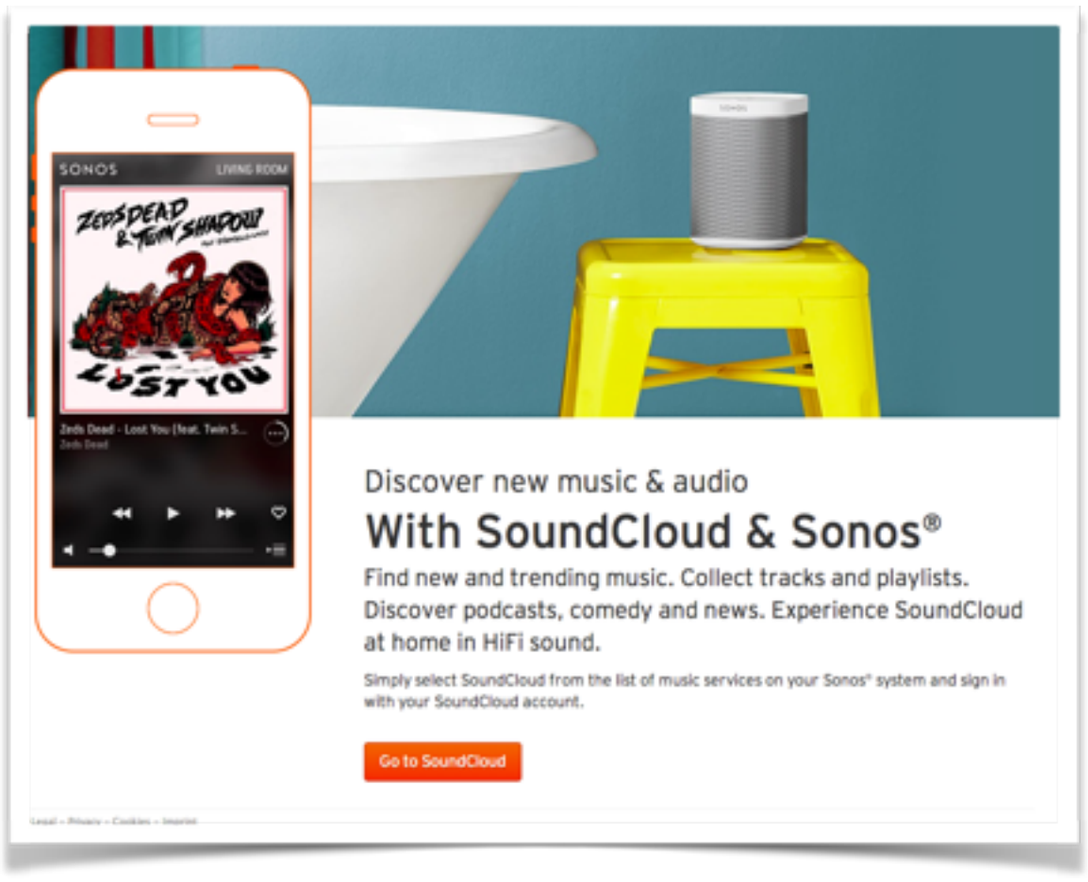
# Enable & empower

## app devs

# "Experience based" api?

# What comes next?

# more devices



# IDL based RPC?

# ephemeral test environments

Please evaluate this talk via the mobile app!

Engage

phil calçado

http://philcalcado.com″
@pcalcado″

www.soundcloud.com″