

API billioner

How to develop successful cloud native software

Piotr Bochyński



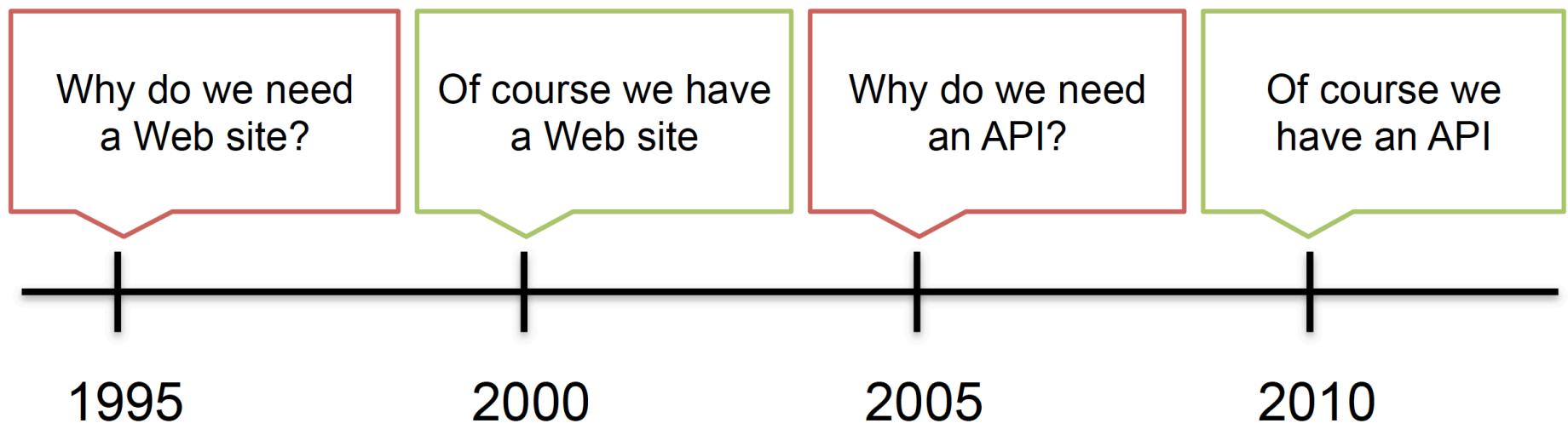
Agenda

- **Basics of micro-services architecture**
- **API and its economy**
- **API first approach to software design**
- **Examples of API in our daily lives**
- **Practical exercise: how to use existing API**

Drive innovation	Marketing channel	BizDev / LeadGen
User acquisition		New line of business
Upsell opportunity	API as Product	Increase footprint
Device and mobile support		Distribution channel
Content acquisition		Partner opportunities
Accelerate internal projects	Drive traffic	Increase stickiness
		Extend product

“APIs have become the currency for success in the software as a service (SaaS) and broader cloud computing marketplace.”

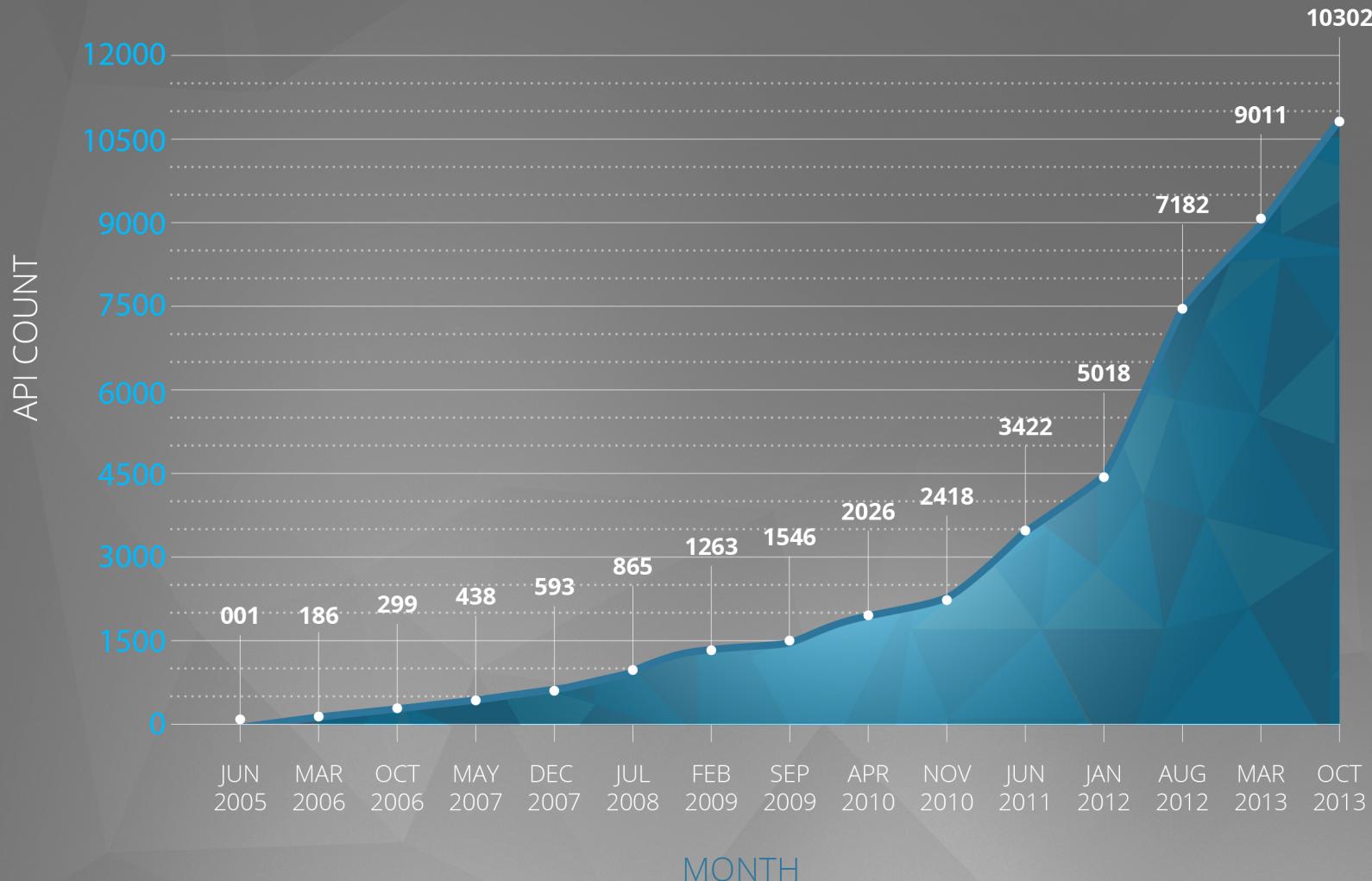
Jeff Kaplan





ProgrammableWeb

Growth In Web APIs Since 2005





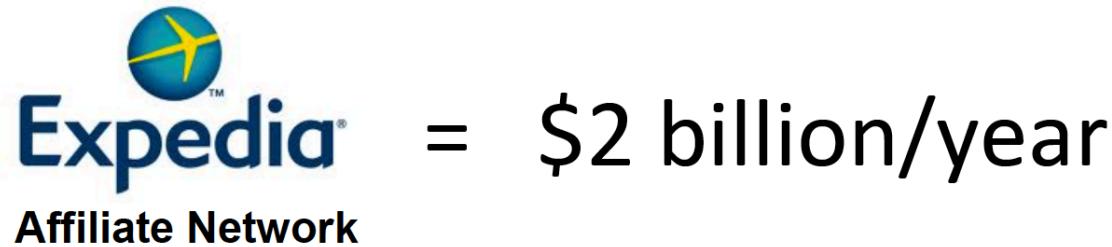
<http://www.programmableweb.com/apis/directory>

<https://www.mashape.com/explore>

<https://www.publicapis.com/>

<http://developer.mashery.com/apis>

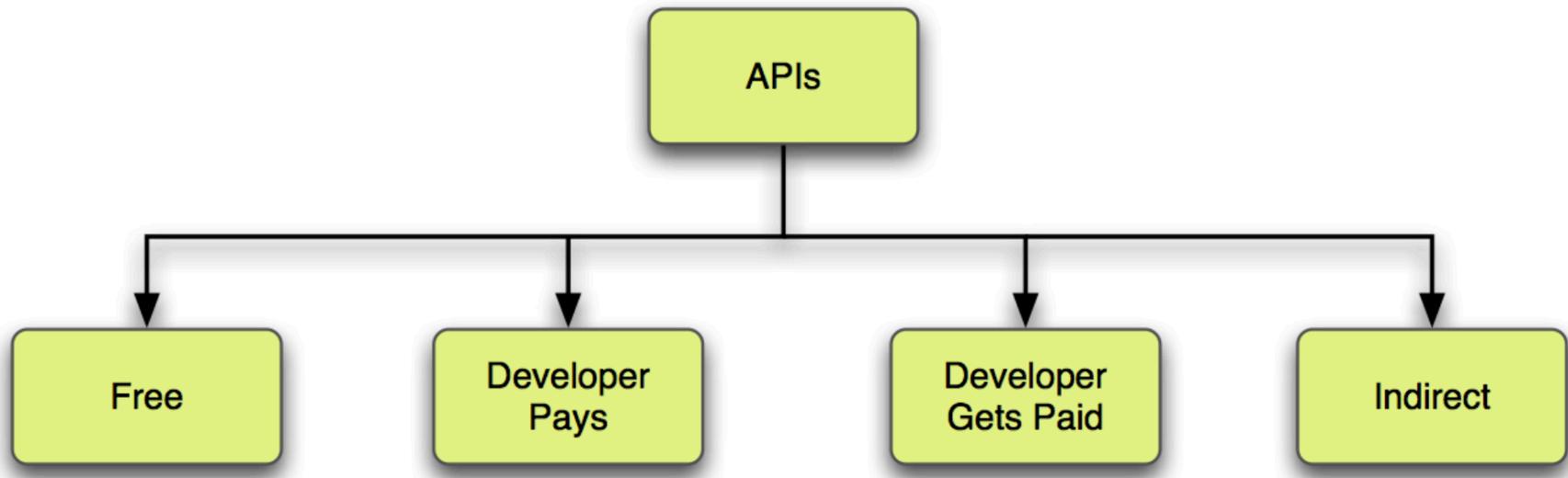
APIs, now a billion \$ business



“90% of what we do is
business through APIs”

John Watton, Expedia Affiliate Network, Travolution.co.uk, April 2012

API Buisness Models



API economy

- **your API is your software frontend**
- **API usage is important feedback**
- **KPIs mainly based on API usage and performance**
- **secured API = less problems with frontends**

- **Traffic: total API calls, top methods, call chains, quota faults**
- **Developers: total, active, top developers, trending apps**
- **Service: performance, availability, error rates**
- **Marketing: registrations, traffic sources**
- **Business: direct revenue, market share**
- **engagement**

API Billionaire Club

twitter

13 billion API calls / day (May 2011)

Google™

5 billion API calls / day (April 2010)

facebook

5 billion API calls / day (October 2009)

ACCU WEATHER

1.1 billion API calls / day (April 2011)

KLOUT

1 billion API calls / day (May 2012)

ebay®

1 billion API calls / day (Q1 2012)

Sabre

1 billion API calls / day (January 2012)

TRY IT !!!

github.com/pbochynski/api-workshop

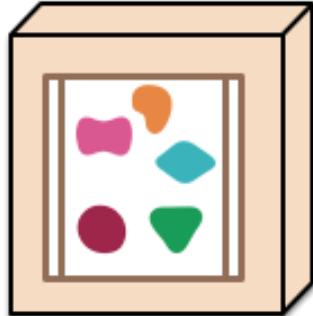
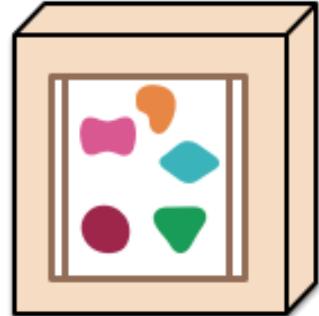
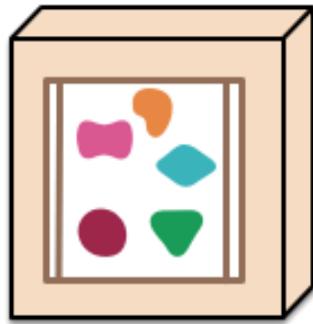
Monolith and microservices (martinfowler.com)



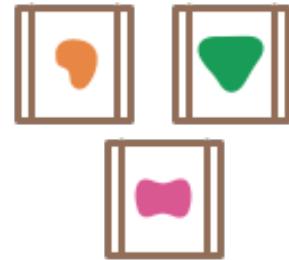
A monolithic application puts all its functionality into a single process...



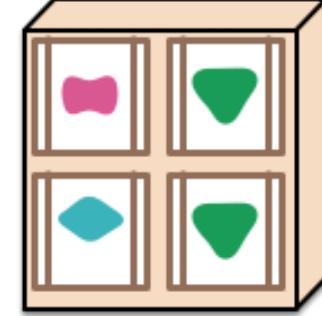
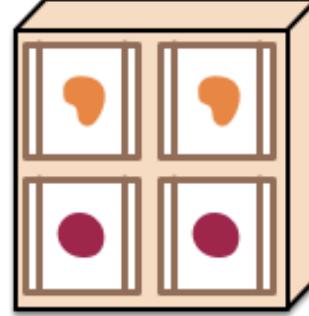
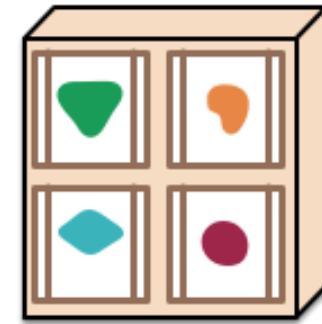
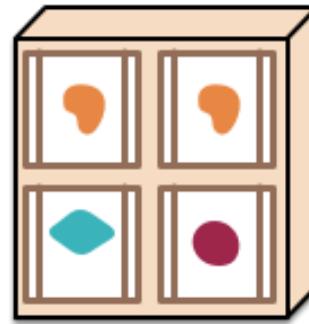
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

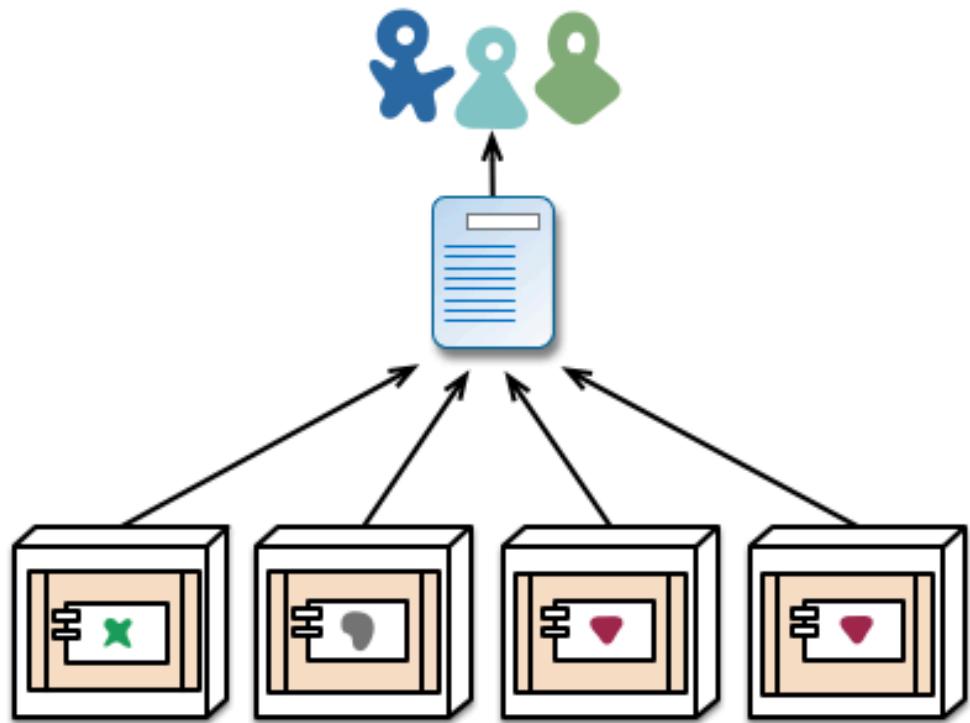


... and scales by distributing these services across servers, replicating as needed.

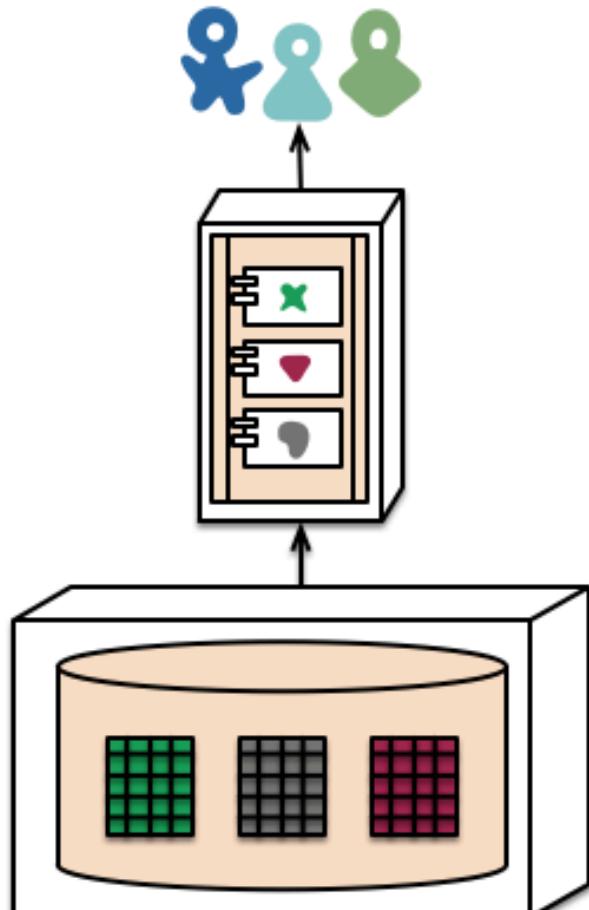


- **Componentization via services**
- **Organized around business capabilities**
- **Products not projects**
- **Smart endpoints and dumb pipes**
- **Decentralized governance**
- **Decentralized data management**
- **Infrastructure automation**
- **Design for failure**
- **Evolutionary design**

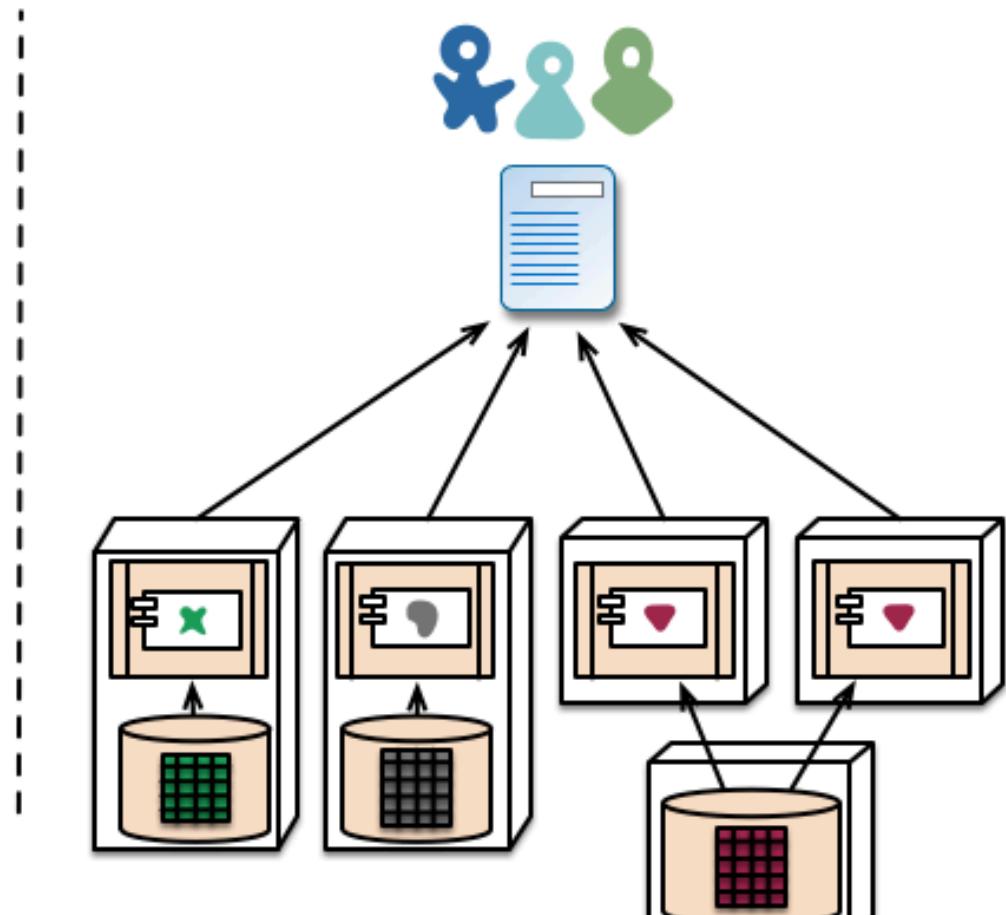
- procedures (libraries)
- CORBA
- EJB
- SOA + ESB
- microservices



Decentralized data management



monolith - single database



microservices - application databases

CONTINUOUS DELIVERY



CONTINUOUS DEPLOYMENT



- **strong module boundaries**
- **independent deployment**
- **technology diversity**

TIME TO MARKET



Service-oriented architecture (SOA):
an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.

Microservices:
a software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs



BIG, EXPENSIVE, PROPRIETARY SOFTWARE

COMPLEX BIG SERVICES

CENTRAL INFRASTRUCTURE – ESB

HEAVY WEIGHT PROTOCOLS (SOAP)

SOAP vs REST

SOAP

```
POST  
/GetStock  
HTTP/1.1  
Host: www.example.org  
Content-Type: application/soap+xml  
<?xml version="1.0"?>  
<soap:Envelope  
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"  
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">  
<soap:Body xmlns:m="http://www.example.org/stock">  
    <m:GetStockPrice>  
        <m:StockName>SAP</m:StockName>  
    </m:GetStockPrice>  
</soap:Body>  
</soap:Envelope>
```

REST

```
GET http://example.org/stock/SAP
```

XML

```
<?xml  
version="1.0"?>  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"  
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">  
<soap:Body  
xmlns:m="http://www.example.org/  
stock">  
<m:GetStockPriceResponse>  
<m:Price>68.72</m:Price>  
</m:GetStockPriceResponse>  
</soap:Body>  
</soap:Envelope>
```

JSON

```
{  
  "symbol": "SAP",  
  "price": 68.72,  
}
```

Pragmatic and agile approach to SOA

- single responsibility principle
- built for fault-tolerance
- infrastructure automation

SOA DONE RIGHT



Significant Operations Overhead

Substantial DevOps Skills Required

Duplication Of Effort

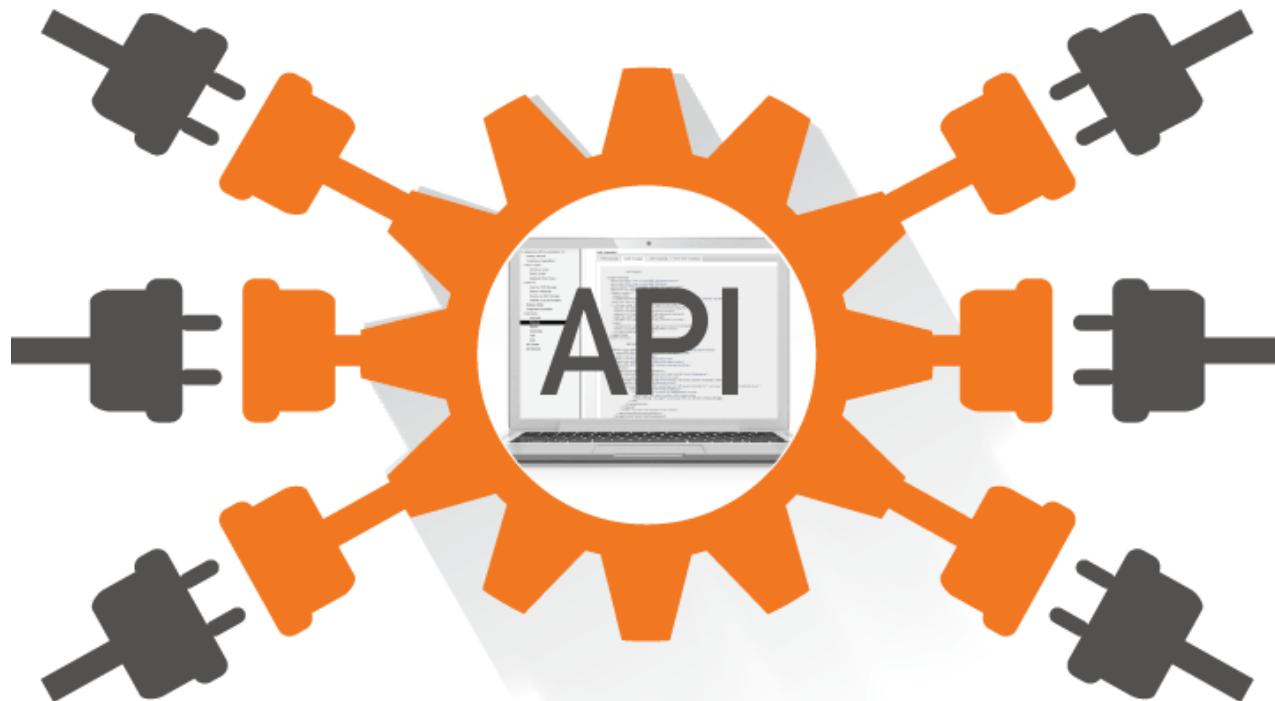
Distributed System Complexity

Asynchronicity Is Difficult!

Testability Challenges

Conway's Law

- **Standardize edges not nodes**
- **To make microservices interchangeable we need well defined, stable, technology independent interfaces**





- **API design: top-down vs. bottom-up**
- **RAML, Swagger, API Blueprint**
- **basics of REST**
- **security & OAuth 2.0 explained**