API-First Development
Workshop

apigee

Alex Muramoto

alexm@apigee.com
@alexmuramoto

Prabhat Jha

pjha@apigee.com
@prabhatjha

#Apigee

We work for Apigee

Walgreens ebay BEST BUY Chegg VIACOM redbox swisscom

DELL FT FINANCIAL TIMES cars.com NBC UNIVERSAL vodafone THOMSON REUTERS verizon

PEARSON

gracenote. Earth Networks CISCO NDS World Leader • World Vision mnBlox Adobe Telefónica kt

THE WORLD BANK MARKS & SPENCER FOX SPORTS dish NETWORK

MORNINGSTAR TRADEKING Digital River

INNOTAS On Time. On Plan. On Demand. Autodata EQUIFAX AT&T U-verse

CITRIX LIVE NATION HCSC Health Care Service Corporation

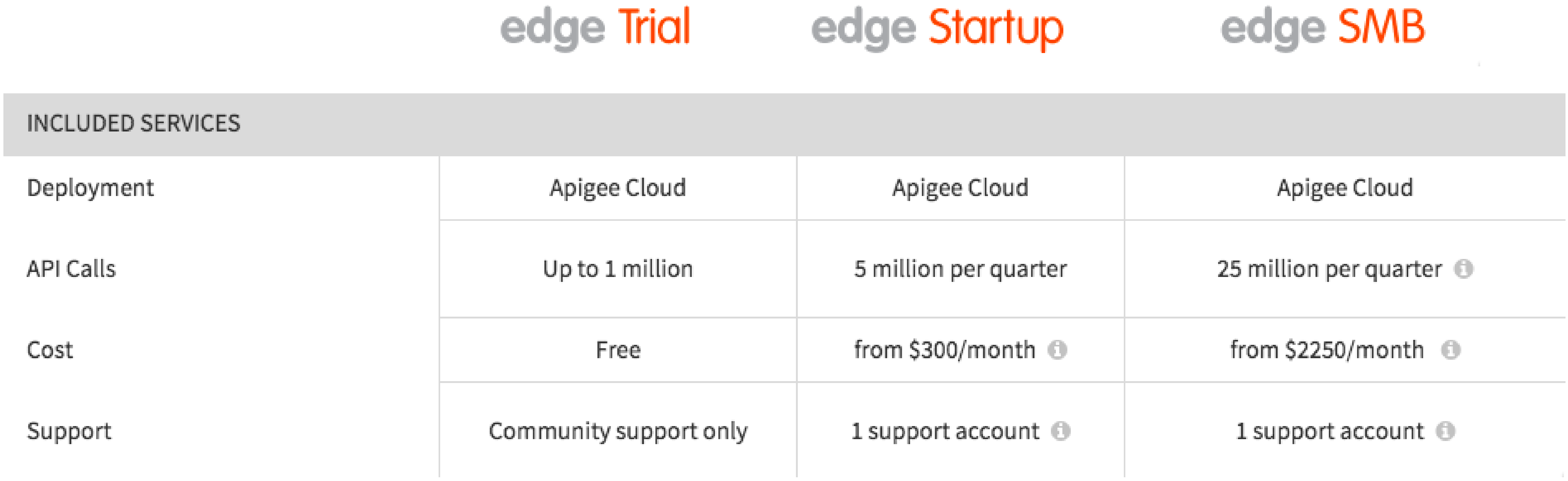
infogroup NONPROFIT SOLUTIONS

BECHTEL meredith mitchell IQT IN·Q·TEL The Motley Fool SAMTRAFIKEN SAMTRAFIKEN I SVERIGE AB European Patent Office

apigee

Why do we do this workshop?

Apigee Trial / Startup / Apigee SMB



	edge Trial	edge Startup	edge SMB
INCLUDED SERVICES			
Deployment	Apigee Cloud	Apigee Cloud	Apigee Cloud
API Calls	Up to 1 million	5 million per quarter	25 million per quarter ⓘ
Cost	Free	from $300/month ⓘ	from $2250/month ⓘ
Support	Community support only	1 support account ⓘ	1 support account ⓘ

apigee

Who are you?

QUESTION
EVERYTHING

Agenda

Section 1

RESTful API Best Practices

Section 2

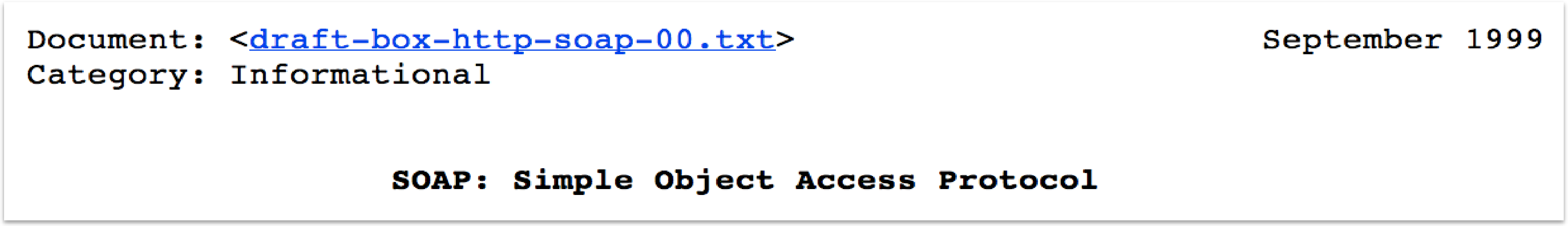
API-First Development

Section 3

Let’s get our hands dirty -- workshop

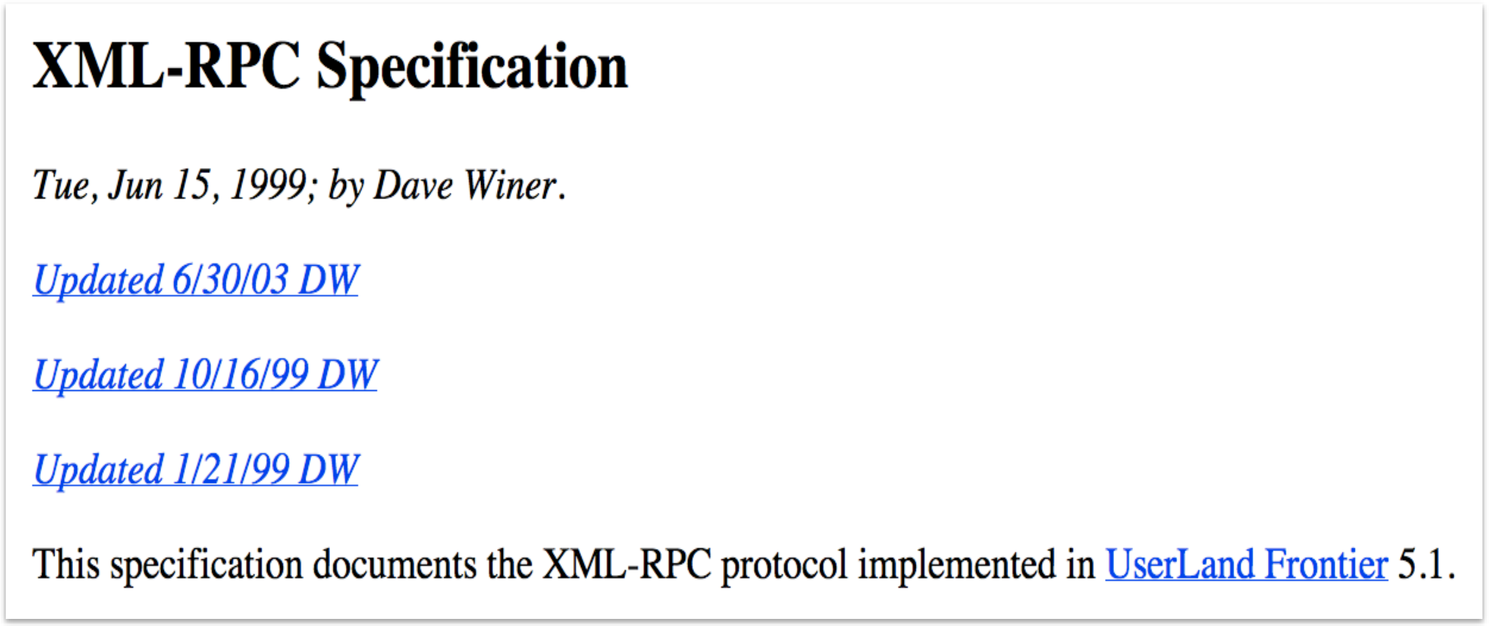
RESTful API Design Best Practices

In the beginning…



```
Document: <draft-box-http-soap-00.txt>                  September 1999
Category: Informational


                  SOAP: Simple Object Access Protocol
```



XML-RPC Specification

Tue, Jun 15, 1999; by Dave Winer.

Updated 6/30/03 DW

Updated 10/16/99 DW

Updated 1/21/99 DW

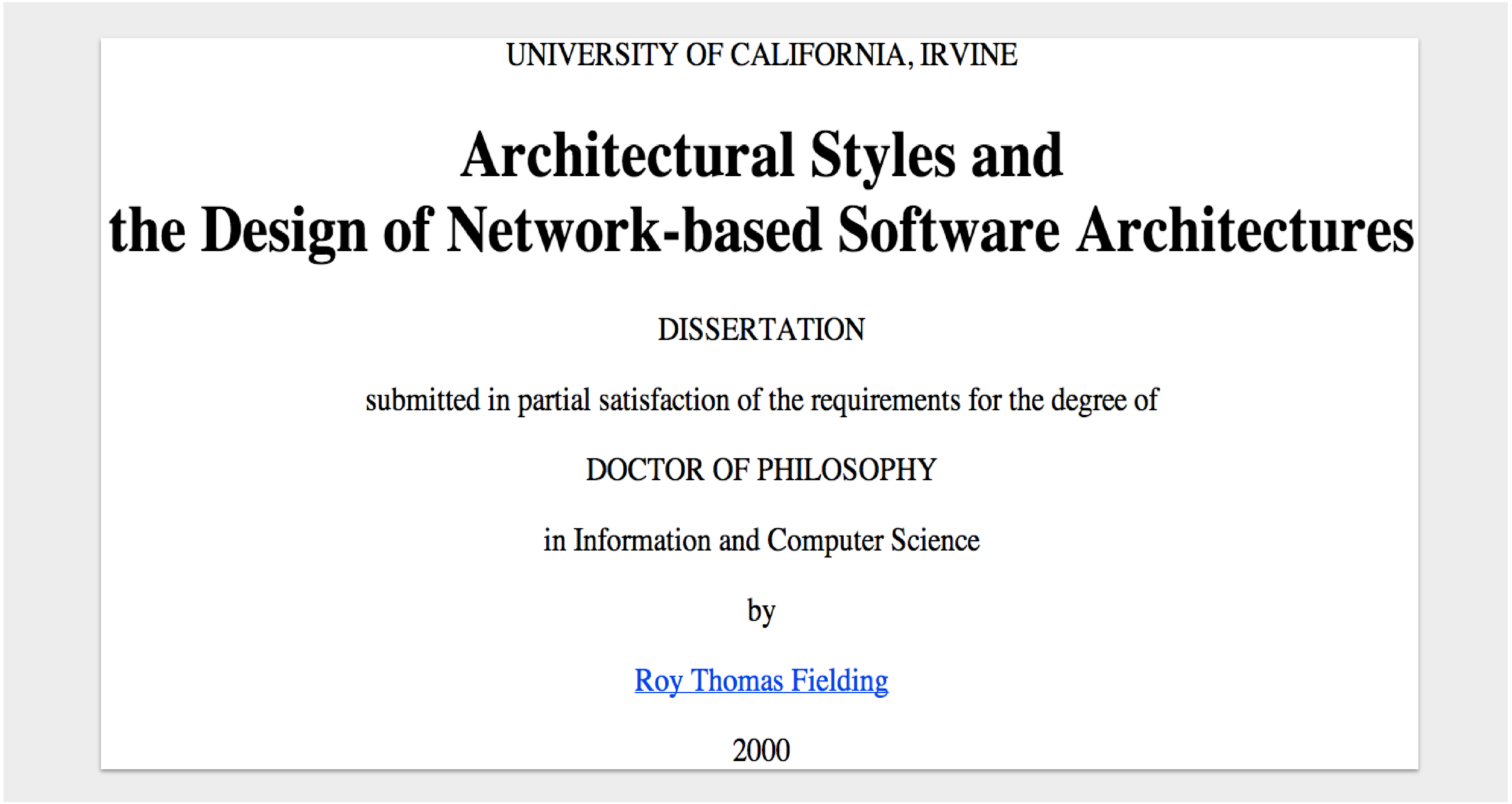
This specification documents the XML-RPC protocol implemented in UserLand Frontier 5.1.

apigee

SOAP and XML-RPC

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:RequestHeader
         soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
         soapenv:mustUnderstand="0"
         xmlns:ns1="https://www.google.com/apis/ads/publisher/v201403">
      <ns1:networkCode>123456</ns1:networkCode>
      <ns1:applicationName>DfpApi-Java-2.1.0-dfp_test</ns1:applicationName>
    </ns1:RequestHeader>
  </soapenv:Header>
  <soapenv:Body>
    <getAdUnitsByStatement xmlns="https://www.google.com/apis/ads/publisher/v201403">
      <filterStatement>
        <query>WHERE parentId IS NULL LIMIT 500</query>
      </filterStatement>
    </getAdUnitsByStatement>
  </soapenv:Body>
</soapenv:Envelope>
```

The dawn of REST



UNIVERSITY OF CALIFORNIA, IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

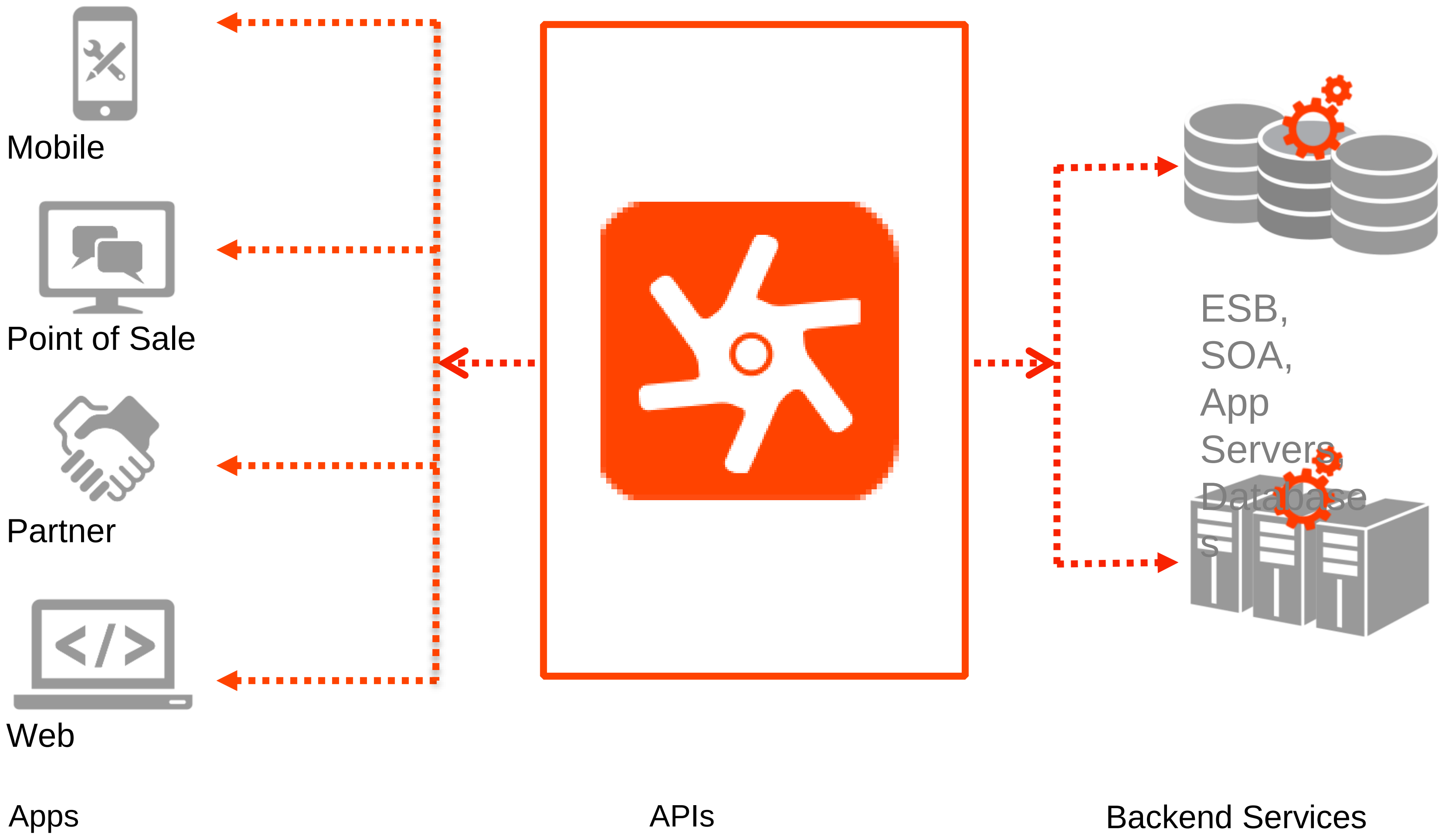
2000

Representational State Transfer

- Resources and resource identifiers
- Transport and semantic independence
- Statelessness
- Interface/contract uniformity
- Metadata and shared understanding of data types
- Self-descriptive messages
- Hypertext as the Engine of Application State (HATEOAS)

What could be easier?

```
GET  https://www.googleapis.com/adexchangebuyer/v1.3/accounts
```


Mobile
Point of Sale
Partner
Web
ESB,
SOA,
App
Servers,
Database
s
Apps
APIs
Backend Services

Teaching a dog to REST

Reluctant API Design

```
/getDog
/getAllDogs
/petDog
/feedDog
/createRecurringDogWalk
/giveCommand
/healthCheck
/getRecurringDogWalkSchedule
/getLocation
/teachTrick
```

Reluctant API Design

/getAllDogs

/petDog

/feedDog

/createRecurringWakeUp

/giveCommand

/checkHealth

/getRecurringWakeUpSchedule

/getLocation

/getDog

/newDog

/getNewDogsSince

/getSittingDogs

/setDogStateTo/saveDog

/getAll**Leashed**Dogs

/verify**Veterinarian**Location

/createRecurring**Medication**

/doDirect**Owner**Discipline

/doCheckupWith**Veterinarian**

/getRecurring**Feeding**Schedule

/get**Hunger**Level

/get**Squirrels**ChasingPuppies

/newDogFor**Owner**

/getNewDogsAt**Kennel**Since

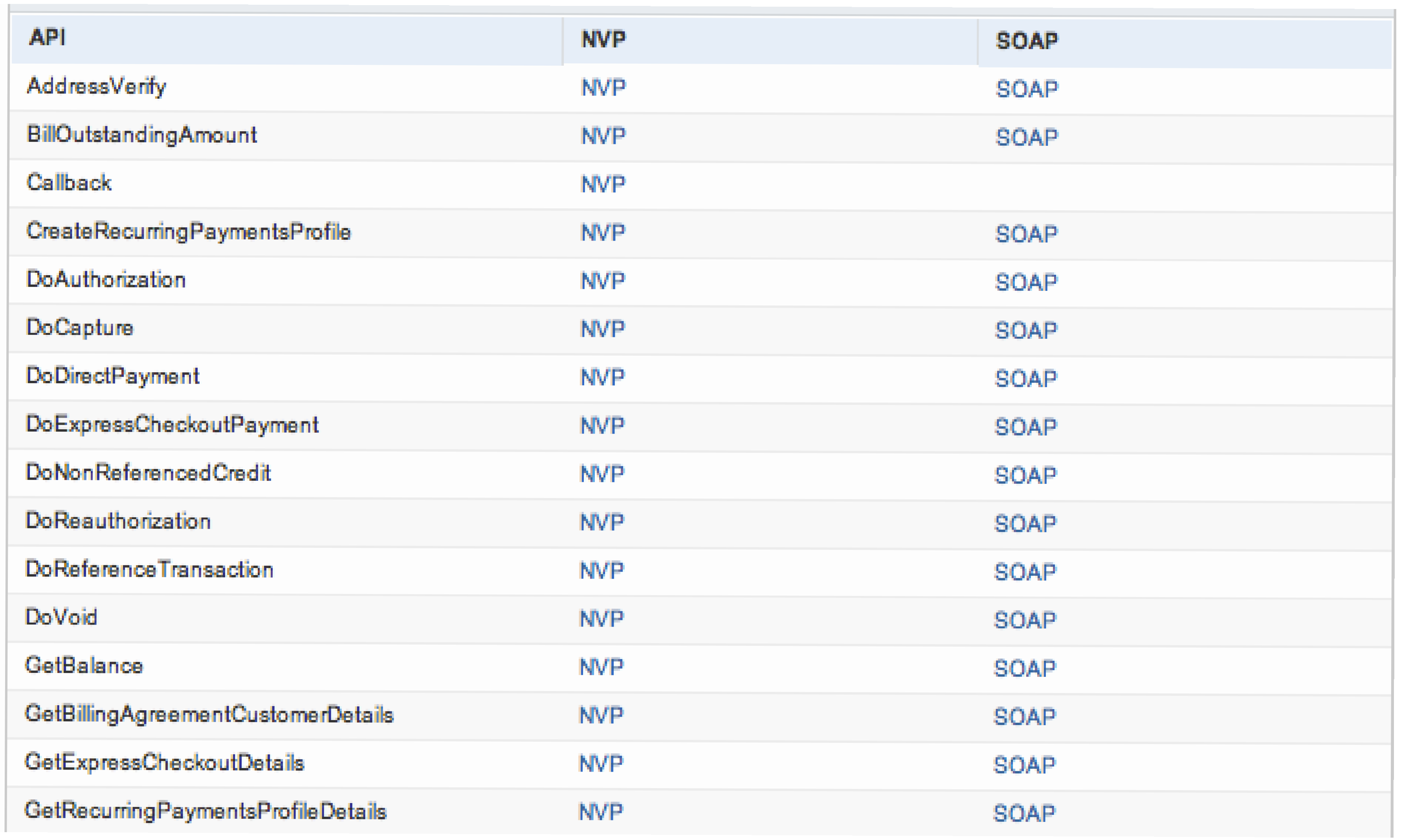
/getSittingDogsAt**Park**

/set**Leashed**DogStateTo

/save**MommaDogs**Puppies

apigee

It happens in the real world



API	NVP	SOAP
AddressVerify	NVP	SOAP
BillOutstandingAmount	NVP	SOAP
Callback	NVP	
CreateRecurringPaymentsProfile	NVP	SOAP
DoAuthorization	NVP	SOAP
DoCapture	NVP	SOAP
DoDirectPayment	NVP	SOAP
DoExpressCheckoutPayment	NVP	SOAP
DoNonReferencedCredit	NVP	SOAP
DoReauthorization	NVP	SOAP
DoReferenceTransaction	NVP	SOAP
DoVoid	NVP	SOAP
GetBalance	NVP	SOAP
GetBillingAgreementCustomerDetails	NVP	SOAP
GetExpressCheckoutDetails	NVP	SOAP
GetRecurringPaymentsProfileDetails	NVP	SOAP

Design for adoption

RESOURCEVERB

Resource modeling

We only need two base URLs for a resource

A collection	`/dogs`
A resource	`/dogs/1234`

Three's company

```
/owners/5678/dogs
```

Sweep complexity behind the “?”

```
/dogs?color=red&state=running&location=park
```

Actions are resources

(hypothetical)

```
/convert?from=EUR&to=CNY&amount=100
```

DigitalOcean

```
/droplets/{droplet_id}/reboot
```

Facebook

```
/search?q=watermelon&type=post
```

Never EVER

Never Break
The Client.

EVER.

unless…

Versioning schemes

```
/2010-04-01/Accounts/
```

```
/services/data/v30.0/limits
```

```
/v2.0/me
```

```
/v2/users/{userid}/checkins
```

API-First Development

Zen of API Development

The code defines the API

The API generates the code

The code is the API

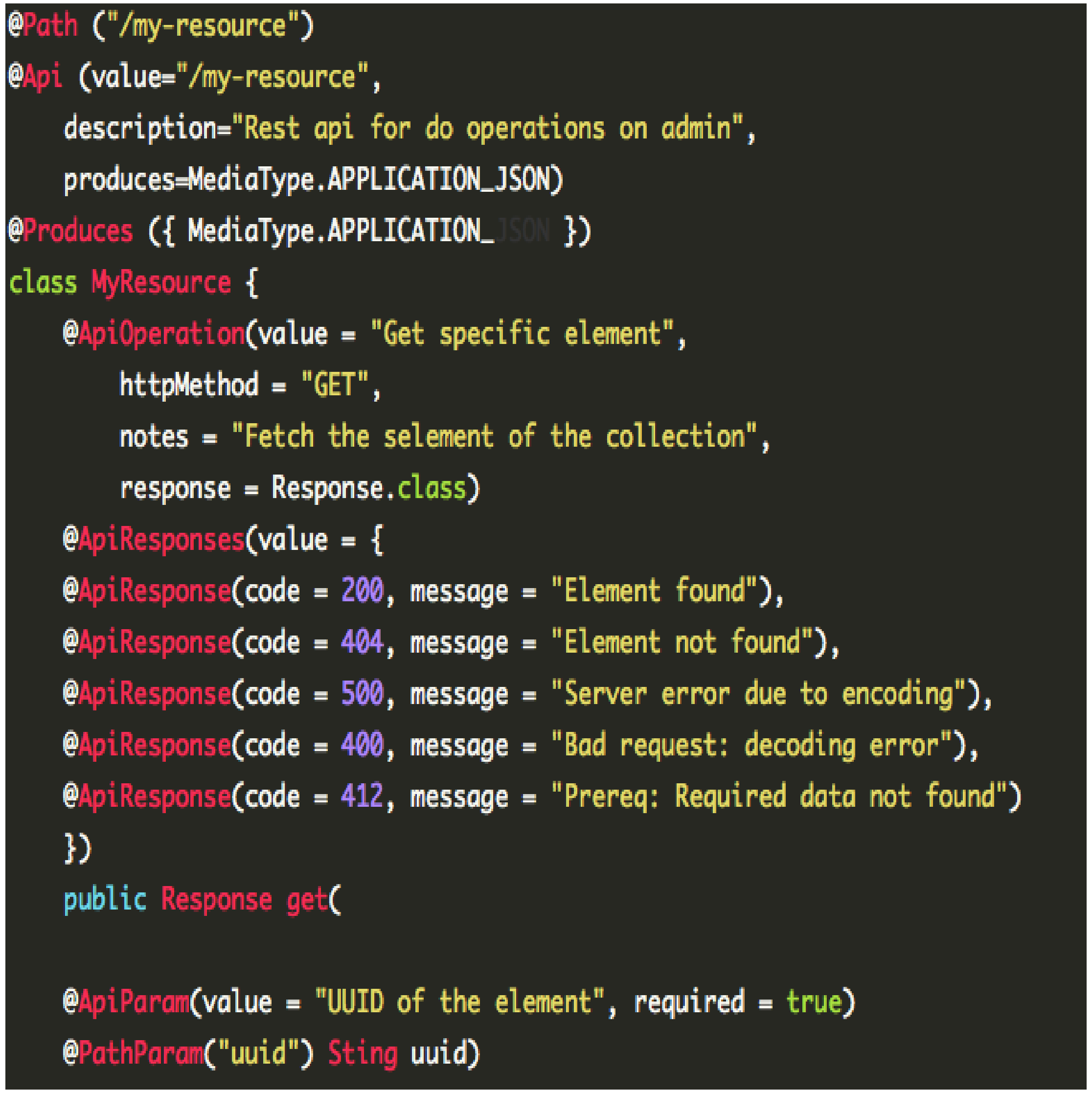
API-driven code

The code defines the API

Annotation-driven

Maintained in code

API is generated



```
@Path ("/my-resource")
@Api (value="/my-resource",
    description="Rest api for do operations on admin",
    produces=MediaType.APPLICATION_JSON)
@Produces ({ MediaType.APPLICATION_JSON })
class MyResource {
    @ApiOperation(value = "Get specific element",
        httpMethod = "GET",
        notes = "Fetch the selement of the collection",
        response = Response.class)
    @ApiResponses(value = {
    @ApiResponse(code = 200, message = "Element found"),
    @ApiResponse(code = 404, message = "Element not found"),
    @ApiResponse(code = 500, message = "Server error due to encoding"),
    @ApiResponse(code = 400, message = "Bad request: decoding error"),
    @ApiResponse(code = 412, message = "Prereq: Required data not found")
    })
    public Response get(

    @ApiParam(value = "UUID of the element", required = true)
    @PathParam("uuid") Sting uuid)
```

API-driven code philosophy

The API must be designed first.

The artifact that represents the API design must drive the API runtime.

The API design will change, and the framework must make it possible to adapt quickly without letting the code, design, and documentation fall out of sync.

The "DRY" Principle

©2015 Apigee. All Rights Reserved.

The API generates the code

Interface Definition Language (IDL) defines the API

Client and server-side stubs code stubs are generated

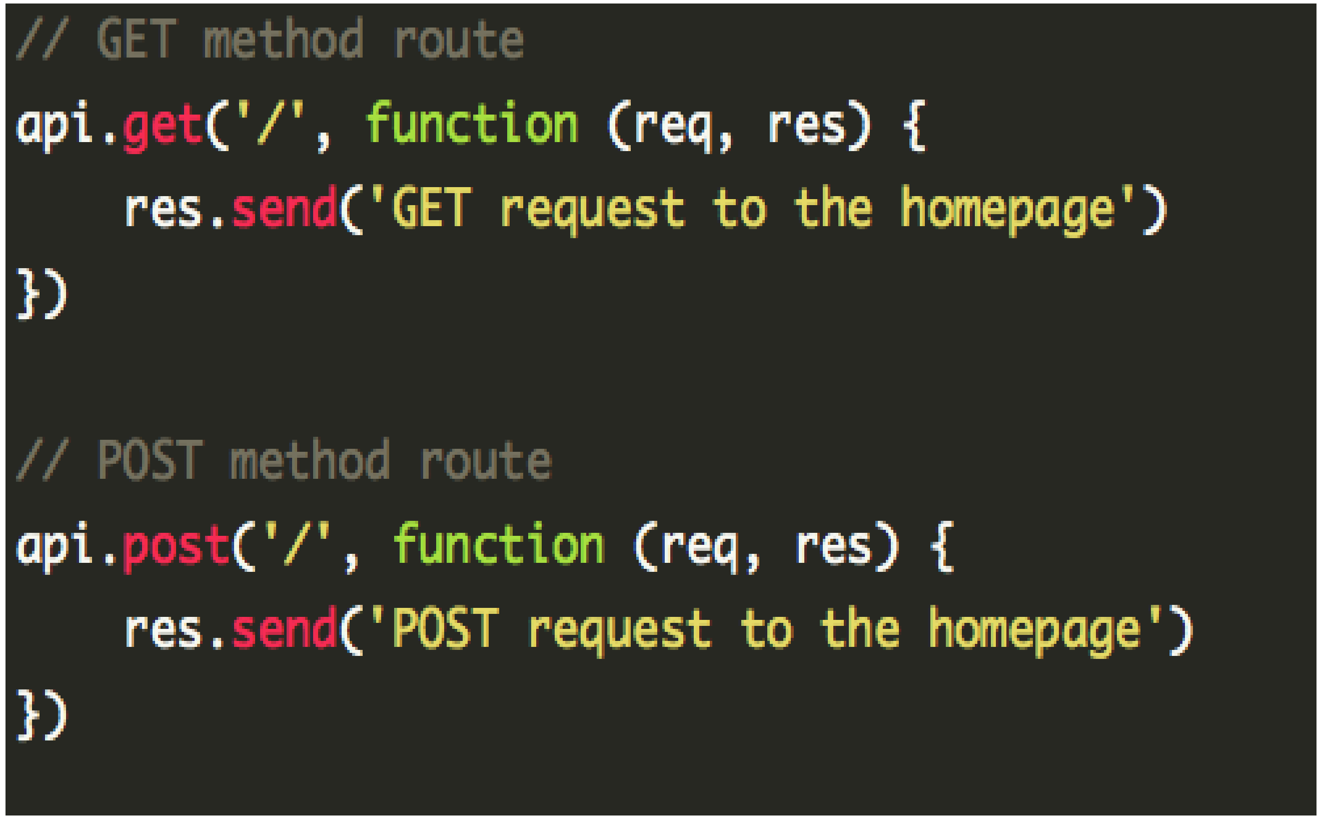
Examples: SOAP, CORBA, and similar RPC systems

©2015 Apigee. All Rights Reserved.

The code is the API

Interpreted

No formal specification



```
// GET method route
api.get('/', function (req, res) {
    res.send('GET request to the homepage')
})

// POST method route
api.post('/', function (req, res) {
    res.send('POST request to the homepage')
})
```

©2015 Apigee. All Rights Reserved.

API-First Philosophy

The API must be designed first.

©2015 Apigee. All Rights Reserved.

API-First Philosophy

API design, documentation, and code must remain in sync.

©2015 Apigee. All Rights Reserved.

API-First Philosophy

The system must adhere to the "DRY Principle"

©2015 Apigee. All Rights Reserved.

API-driven philosophy

The API must directly drive runtime and documentation.

©2015 Apigee. All Rights Reserved.

Swagger-Node

Swagger-Node

The API is written in Swagger, optionally using Swagger-Editor

The Swagger API document is parsed when server starts

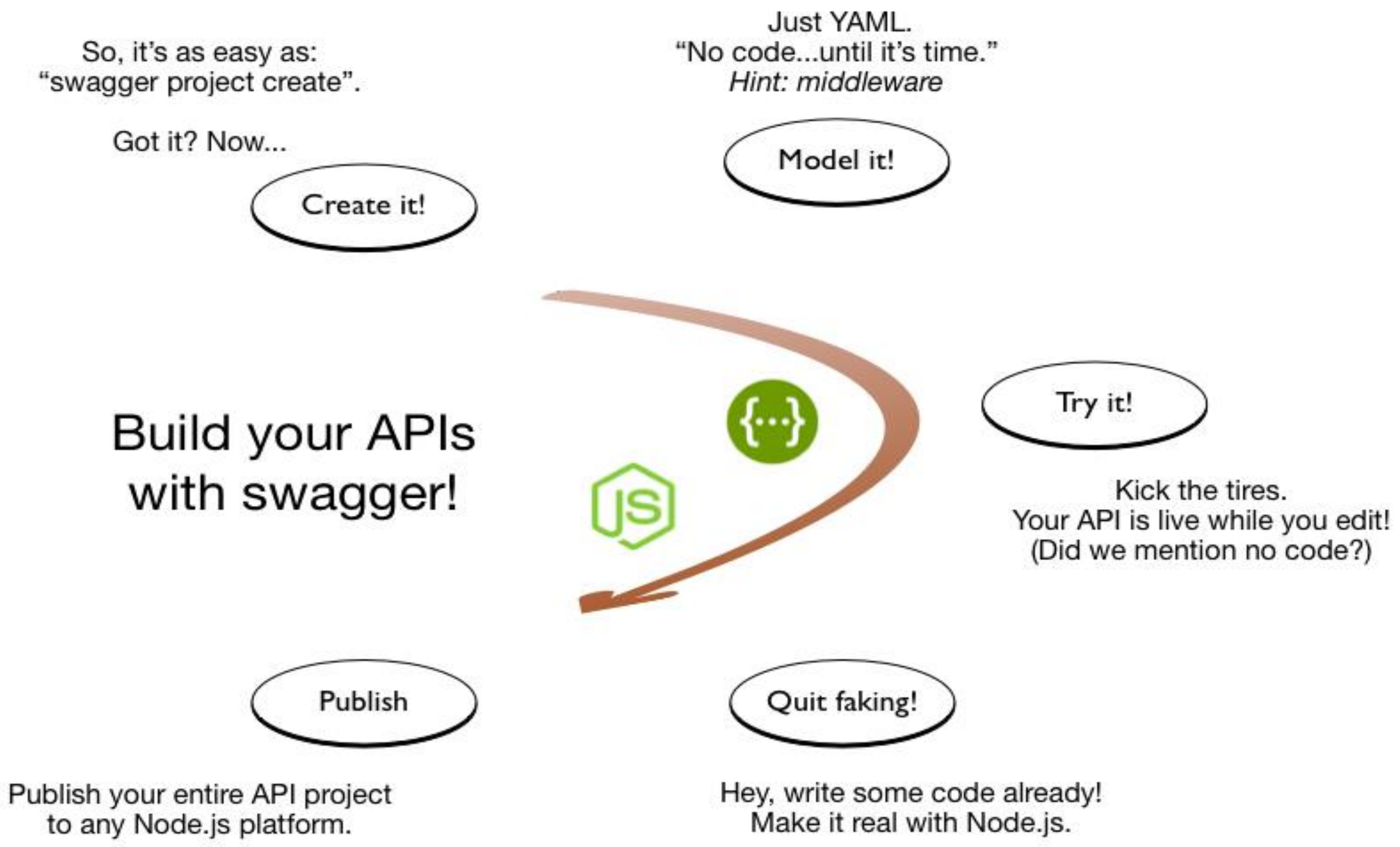
Incoming calls are classified, validated, and routed in real time

Integrates with Connect, Express, Hapi, Restify, Sails...

Incorporates a plugin model for Swagger (or non-Swagger) extensions

©2015 Apigee. All Rights Reserved.

Swagger-Node: Development Lifecycle



So, it's as easy as:
"swagger project create".

Got it? Now...

Just YAML.
"No code...until it's time."
Hint: middleware

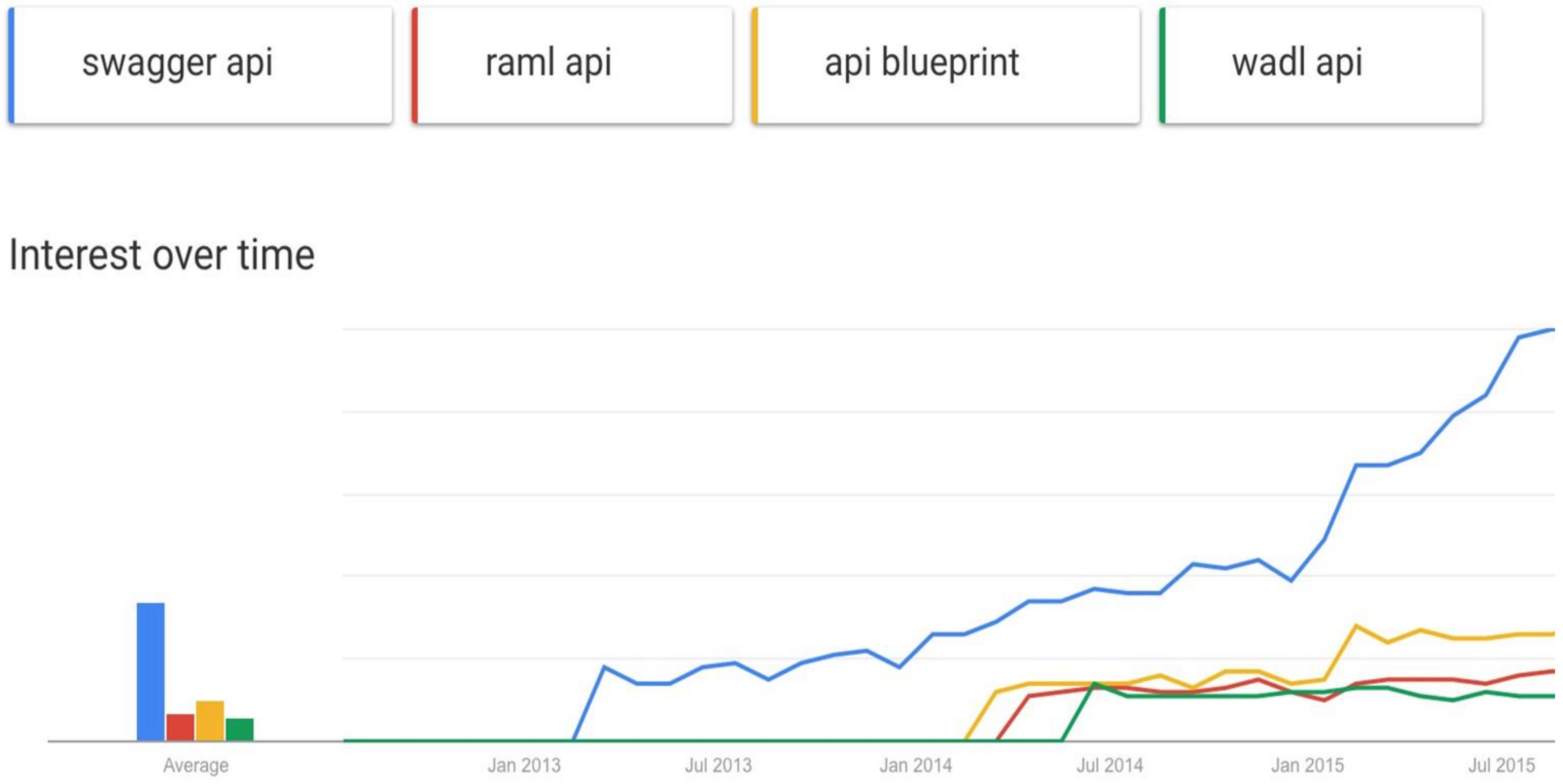
Kick the tires.
Your API is live while you edit!
(Did we mention no code?)

Hey, write some code already!
Make it real with Node.js.

Publish your entire API project
to any Node.js platform.

©2015 Apigee. All Rights Reserved.

But why Swagger ?



©2015 Apigee All Rights Reserved.

But why Node.js ?

Installation

Installation – Step 1

Make sure you have node.js installed. `V4.2.x` preferred.

https://nodejs.org/en/

```
$ node -v

//The latest LTS version 4.2.x
```

©2015 Apigee. All Rights Reserved.

Installation – Step 2

Install

```
$ sudo npm install -g swagger
```

Verify:

```
$ swagger --version

     0.7.4 or later
```

©2015 Apigee. All Rights Reserved.

What do you get?

- CLI
  - project scaffolding
  - project lifecycle
- Your own API Studio (sort of :-))
  - Write YAML
  - Immediate feedback loop
  - Try-it on the fly
- Runtime

©2015 Apigee. All Rights Reserved.

Worskshop Source

https://github.com/prabhatjha/iloveapi2015

Let’s take a tour of API Studio

- Code Completion
- Immediate feedback loop
- Simulated Response aka “Mock Mode”
- Collaboration
- Download YAML/JSON & Node.js project
- Generated doc
- Raw Spec endpoint

©2015 Apigee Corp. All Rights Reserved.

Let’s create a project

```
$ swagger project create
```

©2015 Apigee. All Rights Reserved.

Let’s run the project

```
$ cd $project-name

$ swagger project start
```

©2015 Apigee. All Rights Reserved.

Let’s make some API call

```
$ curl http://127.0.0.1:10010/hello?name=Scott
```

©2015 Apigee. All Rights Reserved.

Let’s not forget the tests

```
$ swagger project generate-test

$ swagger project test
```

©2015 Apigee. All Rights Reserved.

How about editing ?

```
$ swagger project edit
```

©2015 Apigee. All Rights Reserved.

Dissecting it

- Project Conventions
- swagger spec
- controllers
- helpers

©2015 Apigee. All Rights Reserved.

Let’s add something new

- A new path that uses POST operation

©2015 Apigee. All Rights Reserved.

..which requires

- A new controller

©2015 Apigee. All Rights Reserved.

Try it out

```
$ curl -X POST http://127.0.0.1:10010/conf/add -H
"content-type:application/json" -d '{"x":5,"y":6}'
```

How about some API Management

- Let’s add quota

©2015 Apigee. All Rights Reserved.

Quota : Get the bits

- Let’s add quota *[This is subject to change. See latest at https://github.com/apigee-127/volos-swagger-apply/blob/master/README.md]*

```
npm install --save volos-swagger-apply
npm install --save volos-quota-memory
```

©2015 Apigee. All Rights Reserved.

Quota : Annotate your Swagger

```
x-volos-resources:
 MyQuota:
   provider: volos-quota-memory
   options:
     timeUnit: minute
     interval: 1
     allow: 1


x-volos-apply:
       MyQuota: {}
```

apigee

©2015 Apigee Corp. All Rights Reserved.

Quota : Tell the framework about it.

Add fitting to config/default.yaml to `swagger_controllers` :

```
    - volos-swagger-apply
```

©2015 Apigee. All Rights Reserved.

Quota : Verify that it works

```
curl -X POST http://127.0.0.1:10010/conf/add -H
"content-type:application/json" -d '{"x":5,"y":6}'

{"message":"exceeded quota","status":403}

```

©2015 Apigee. All Rights Reserved.

Deploy to Apigee

```
$ sudo npm install -g apigeetool


$ apigeetool deploynodeapp -u sdoe@apigee.com -o sdoe
-e test -n 'Test Node App 2' -d . -m app.js -b /node2

OR

$ a127 project deploy
```

©2015 Apigee. All Rights Reserved.

Where can I get help?

Community: https://community.apigee.com

Github:

- github.com/apigee
- github.com/apigee-127
- github.com/swagger-api

©2015 Apigee. All Rights Reserved.

apigee

Thank you