apigee

ABCs of APIs with Node.js

wifi: Village Events
password: atlantatechvillage

Greg Rewis

grewis@apigee.com
@garazi
#apigee

I work for Apigee


Walgreens
ebay
BEST BUY
Chegg
VIACOM
redbox
swisscom
DELL
FT
FINANCIAL TIMES
cars.com
NBC UNIVERSAL
vodafone
THOMSON REUTERS
verizon
PEARSON
Earth Networks
mBlox
Adobe
Telefónica
kt
gracenote
CISCO
NDS
World Leader • World Vision
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
mitchell
IQT
IN·Q·TEL
The Motley Fool
SAMTRAFIKEN
European Patent Office
meredith

Why do we do these workshops?
Why free?

Apigee is always free for developers

Free Hosted Accounts

25GB storage limit,

10M push notifications/month

no API/bandwidth limit

Commercial use OK

Paid plans available for large companies if you need to deploy on your own servers or SLAs, 4 nines, multi-region, phone support, more storage, etc.

Who are you?

Agenda

Section 1

Introduction to APIs

Section 2

Node.js Crash Course

Section 3

Best Practices in API Design

Section 4

Intro to Apigee127

Agenda

Section 5

Building API Specs/Docs

Section 6

Creating Controllers

Section 7

API Management

Section 8

Deploying the APIs


QUESTION
EVERYTHING

API Fundamentals

What is an API?

Application Programming Interface (abbr.: API)

noun

a system of tools and resources in an operating system, enabling developers to create software applications

What is an API?

A service that:

- specifies how one application can talk to another
- exposes some of an app's internal functions or logic to the outside world
- makes it possible for applications to share data
- is comprised of a set of HTTP/ HTTPS endpoints

Applications use APIs…

…like humans use websites

Node.js Crash Course

Who is using Node.js?

Node.js success stories

- Groupon re-implemented their system in Node.js which resulted in page load times dropping by a whopping 50%.

- LinkedIn moved to Node.js from Rails for their mobile traffic, reducing the number of servers from 30 to 3 (90% reduction) and the new system was up to 20x faster.

Node.js success stories

- PayPal reported: double the number of requests per-second and reduced response time by 35% or 200 milliseconds.

- In addition, PayPal measured a 2x increase in developer productivity, where it took half the number of developers to deliver an application when compared to Java, and it was delivered in less time

Node.js success stories

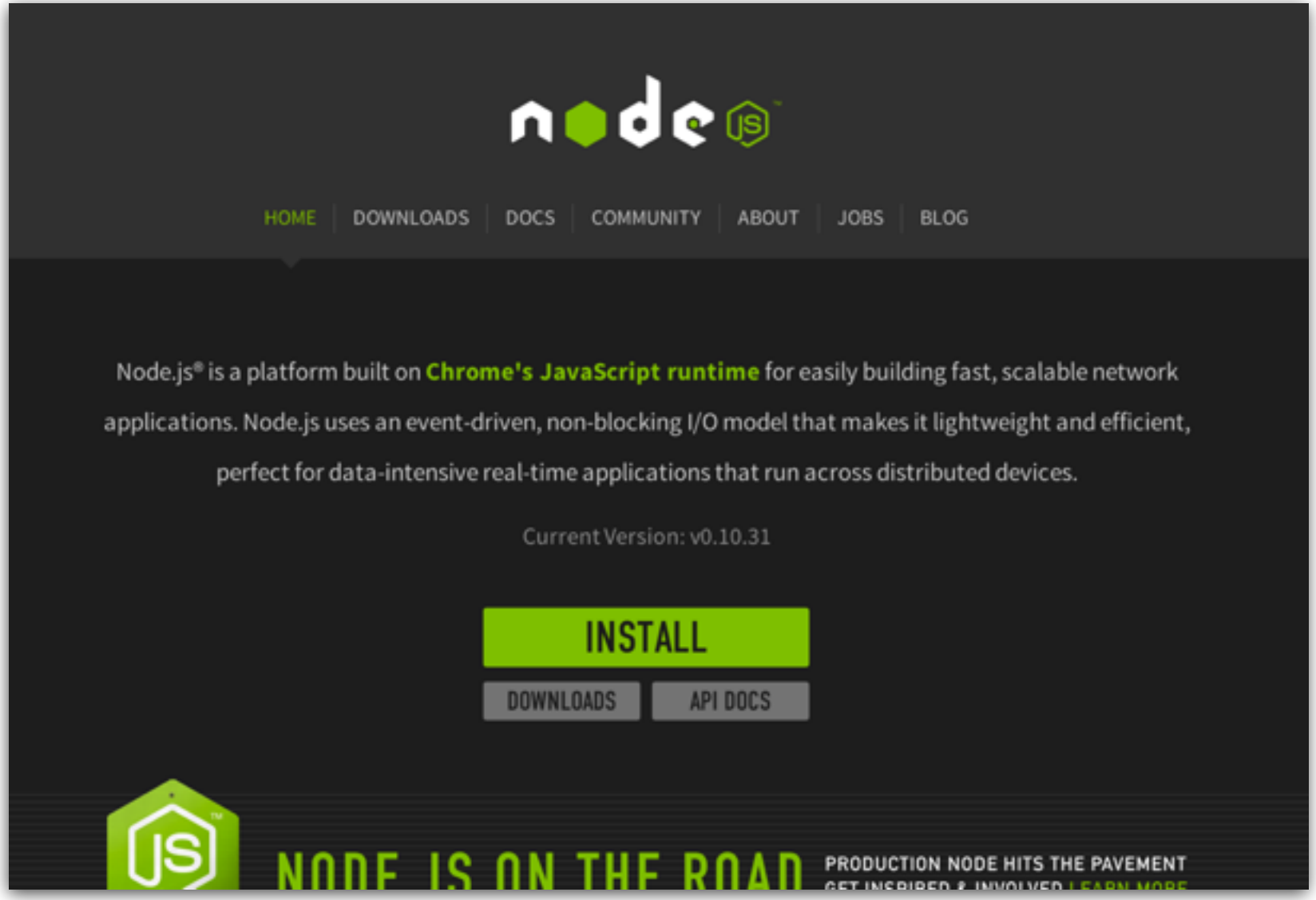
- WalMart Labs had a bumper launch with Node.js in 2013, where they put all of their Mobile traffic through Node.js on black-friday, the busiest shopping period of the year.

- The WalMart servers didn't go over 1% CPU utilization and the team did a deploy with 200,000,000 users online.

Why node.js?

- Javascript running on the server
- Asynchronous, event-driven
- HTTP is a first-class citizen of the Node world
- Exploding in popularity

Node.js installation



Blocking vs. non-blocking

```
<html>
<head>
    <script>
    var test = document.getElementsByTagName('li');
    alert("There are " + test.length + " items");
    </script>
</head>
<body>
    <ul>
        <li>A list item</li>
        <li>Another list item</li>
        <li>A third item in the list</li>
    </ul>
</body>
</html>
```

Blocking vs. non-blocking

```
<html>
<head>
    <script>
        function countItems(el, callback) {
            setTimeout(function() {
                var test = document.getElementsByTagName(el);
                callback(test)
            }, 2000);
        }
        countItems('li', function(test) {
            alert("There are " + test.length + " items");
        });
        alert('Hello World');
    </script>
</head>
<body>
    <ul>
        <li>A list item</li>
        <li>Another list item</li>
        <li>A third item in the list</li>
    </ul>
</body>
</html>
```

The world's simplest web server

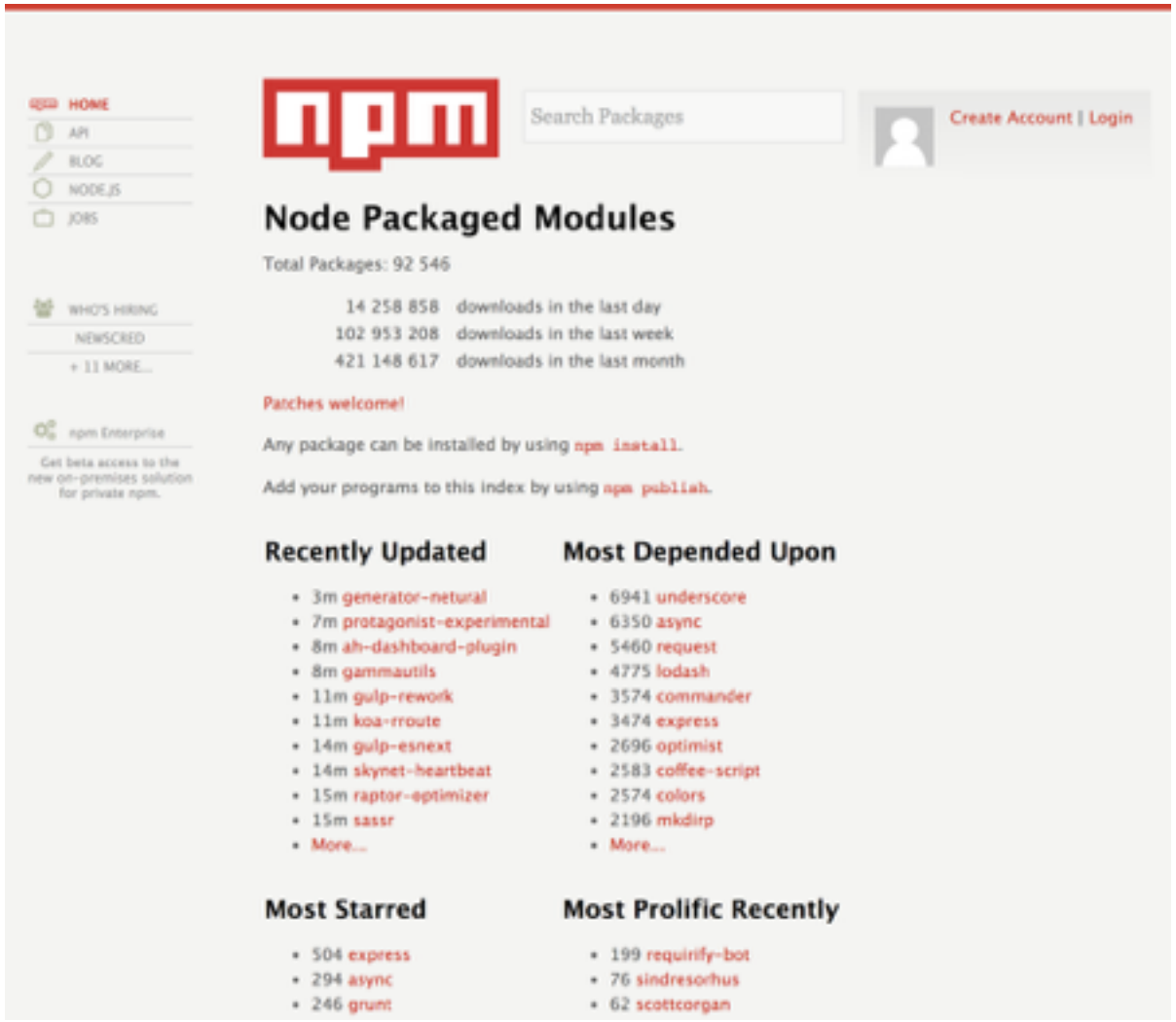
```
var http = require('http');

http.createServer(function(req, res) {
    res.end('Hello World');
}).listen(8888, '127.0.0.1');

console.log('Server running at http://127.0.0.1:8888/');
```

```
LapGARbage:api-workshop greg$ node server.js
Server running at http://127.0.0.1:8888/
```

The world of npm



Modules must be installed

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
    res.send('Hello World');
}).listen(8888);

console.log('Server running at http://127.0.0.1:8888/');
```

```
LapGARbage:api-workshop greg$ node server.js

module.js:340
    throw err;
          ^
Error: Cannot find module 'express'
    at Function.Module._resolveFilename (module.js:338:15)
    at Function.Module._load (module.js:280:25)
    at Module.require (module.js:364:17)
    at require (module.js:380:17)
```

Installing modules

- Individual modules must be installed using the NPM command line

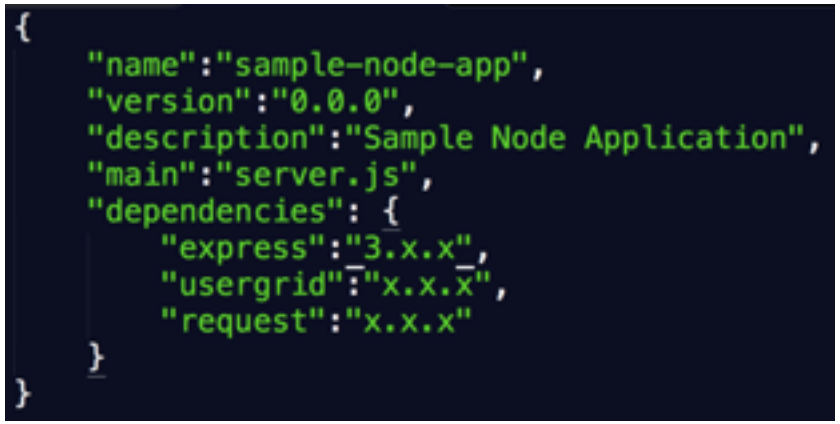
```
LapGARbage:api-workshop greg$ npm install express
npm http GET https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/express/-/express-4.8.7.tgz
npm http 200 https://registry.npmjs.org/express/-/express-4.8.7.tgz
npm http GET https://registry.npmjs.org/debug/1.0.4
```

- Once installed (node_modules folder), they are added to the app using require('module name')

```
var express = require('express');
```

Installing multiple modules

- When a Node.js app launches, it looks for a file, package.json, in the same folder as the main JS file to determine if all of its 'pieces' are present



- Using npm install will install all of the listed dependencies at one time.

Express routes

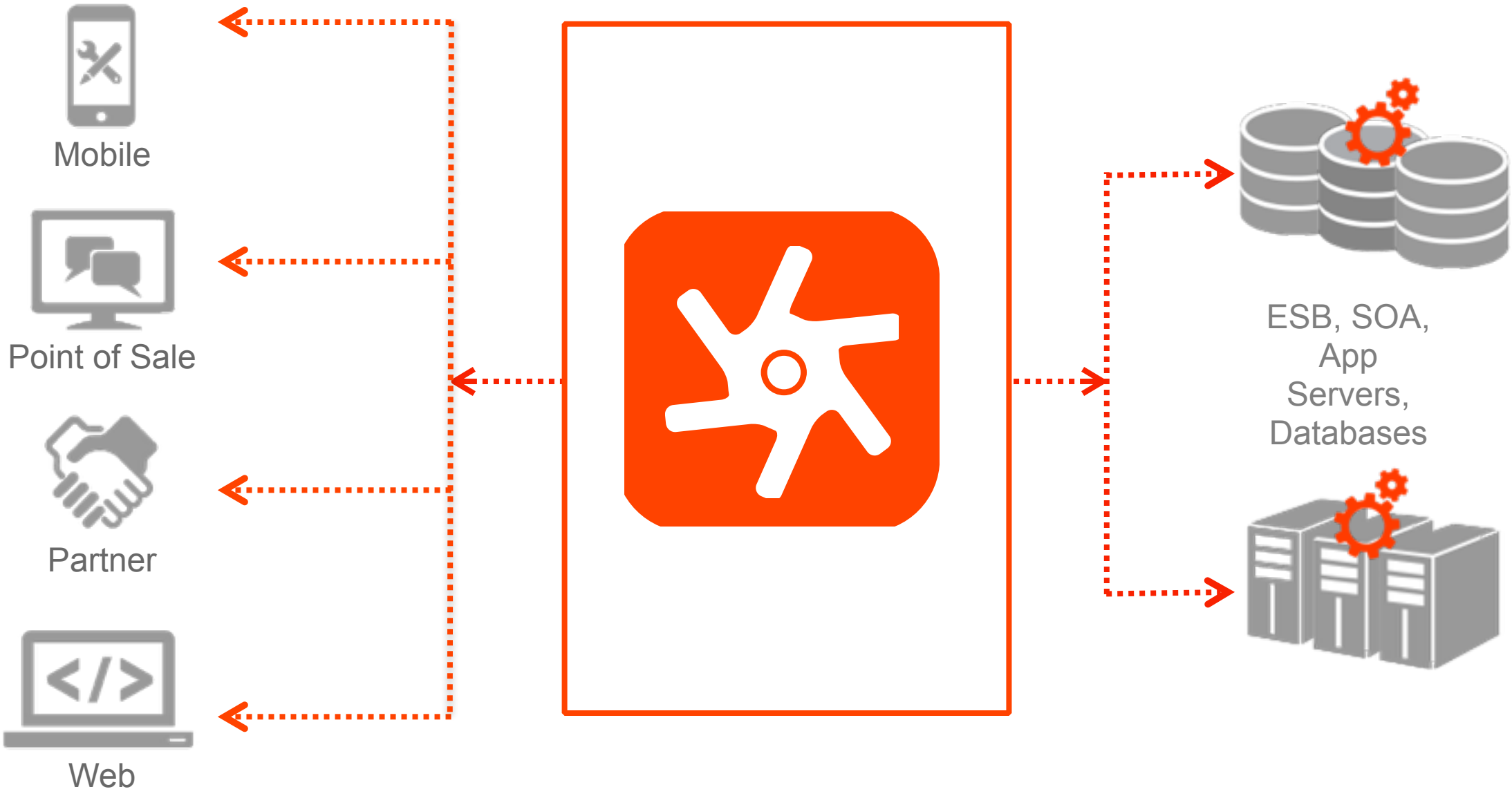
```
var express = require('express');
var app = express();

app.get('/hello', function (req, res) {
    res.send('Hello World');
});

app.use("/", express.static(__dirname));

app.listen(8888, function (req, res) {
    console.log('Server running at http://127.0.0.1:8888/');
});
```

Best Practices in API Design


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

Source: http://stylebizz.com/design-quotes/

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

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
GetTransactionDetails	NVP	SOAP
ManageRecurringPaymentsProfileStatus	NVP	SOAP
ManagePendingTransactionStatus	NVP	SOAP
MassPayment	NVP	SOAP
RefundTransaction	NVP	SOAP
SetCustomerBillingAgreement	NVP	SOAP
SetExpressCheckout	NVP	SOAP
TransactionSearch	NVP	SOAP
UpdateRecurringPaymentsProfile	NVP	SOAP

Design for adoption

A resource is something that lives somewhere

IP: 173.194.70.102 ➡ google.com

Twitter: 14352786 ➡ @garazi

https://github.com/apigee

Verbs are an action

Operation	SQL	HTTP
Create	INSERT	POST
Read (Retrieve)	SELECT	GET
Update (Modify)	UPDATE	PUT / PATCH
Delete (Destroy)	DELETE	DELETE

Let's start by applying some of the best-practice principles from REST.

Resource modeling

We only need two base URLs for a resource

A collection

```
/dogs
```

A resource

```
/dogs/1234
```

Collection names – singular or plural?

Foursquare

```
/checkins
```

Groupon

```
/deals
```

Zappos

```
/Product
```

Collection names – abstract or concrete?

Super High

`/things`

High

`/animals`

Medium

`/dogs`

Low

`/beagles`

too few things?
go lower

too many things?
go higher

Three's company

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

Sweep complexity behind the “?”

Sweep complexity behind the “?”

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

Give developers what they want

```
/dogs?fields=name,color,location.postalCode
```

Putting it all together

Resource	**POST** create	**GET** read	**PUT** update	**DELETE** delete
/dogs	create a new dog	list dogs	replace dogs with new dogs	delete all dogs
/dogs/1234	treat as a collection create new dog in it	show **1234**	if exists update **1234** if not create **1234**	delete **1234**

Actions are collections

(hypothetical)

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

DigitalOcean

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

Facebook

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

How many versions?

How many versions?

- Maintain at least one version back, to give developers at least one cycle to react before obsoleting a version.

- The amount of time that passes until an old version is deprecated can vary, but 6-12 months is a good rule-of-thumb.

Versioning schemes

Twilio

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

salesforce.com

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

Facebook

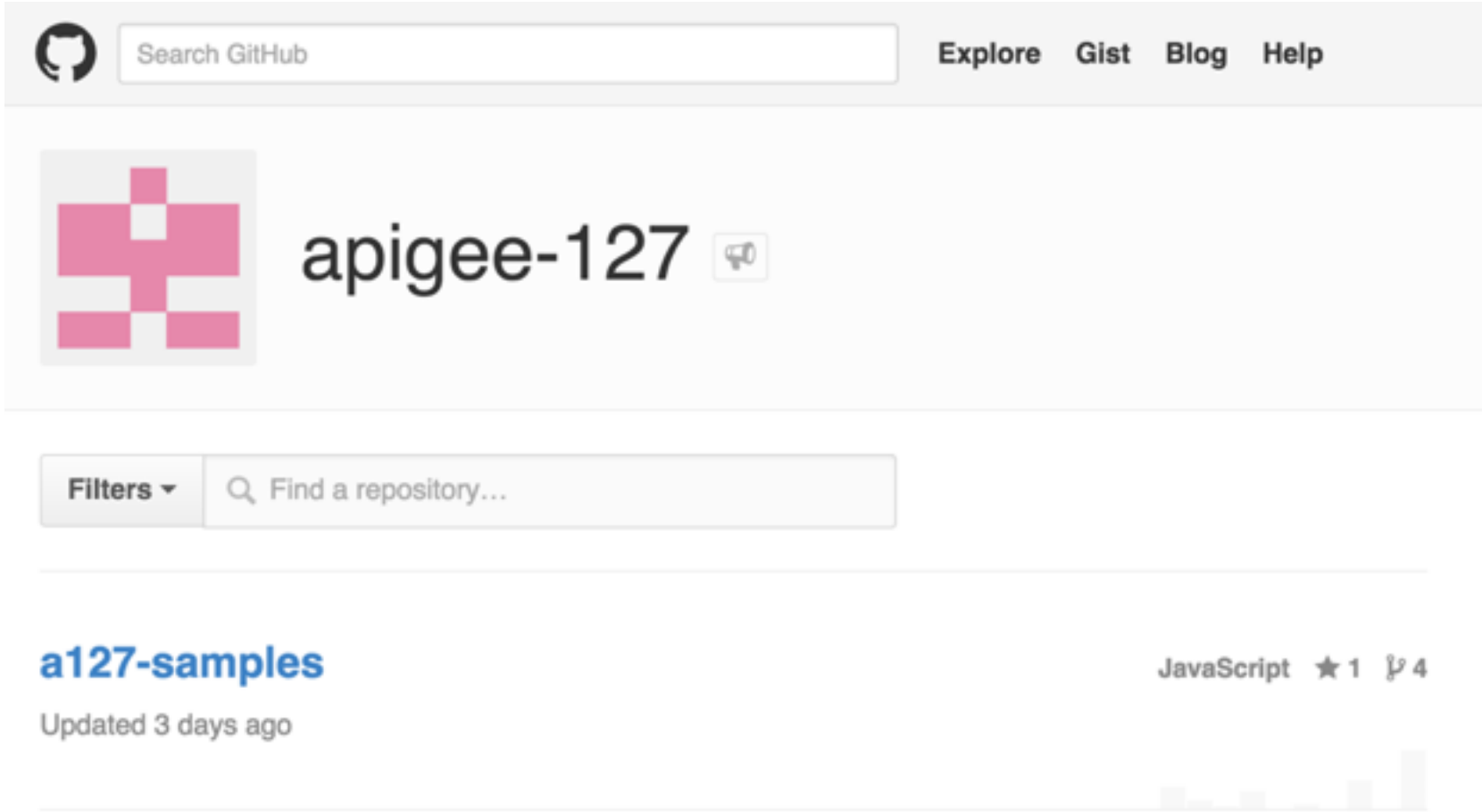
```
/v2.0/me
```

Foursquare

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

Intro to Apigee 127

Apigee 127 on the web



https://github.com/apigee-127

apigee

Installing Apigee 127

- Apigee 127 is installed using the command line – note that you may have to use sudo on a Mac

```
npm install -g apigee-127
```

```
LapGARbage:api-workshop greg$ sudo npm install -g apigee-127
npm http GET https://registry.npmjs.org/apigee-127
npm http 200 https://registry.npmjs.org/apigee-127
npm http GET https://registry.npmjs.org/apigee-127/-/apigee-127-0.1.3.tgz
npm http 200 https://registry.npmjs.org/apigee-127/-/apigee-127-0.1.3.tgz
npm http GET https://registry.npmjs.org/lodash
npm http GET https://registry.npmjs.org/async
npm http GET https://registry.npmjs.org/commander
npm http GET https://registry.npmjs.org/apigee-access
```

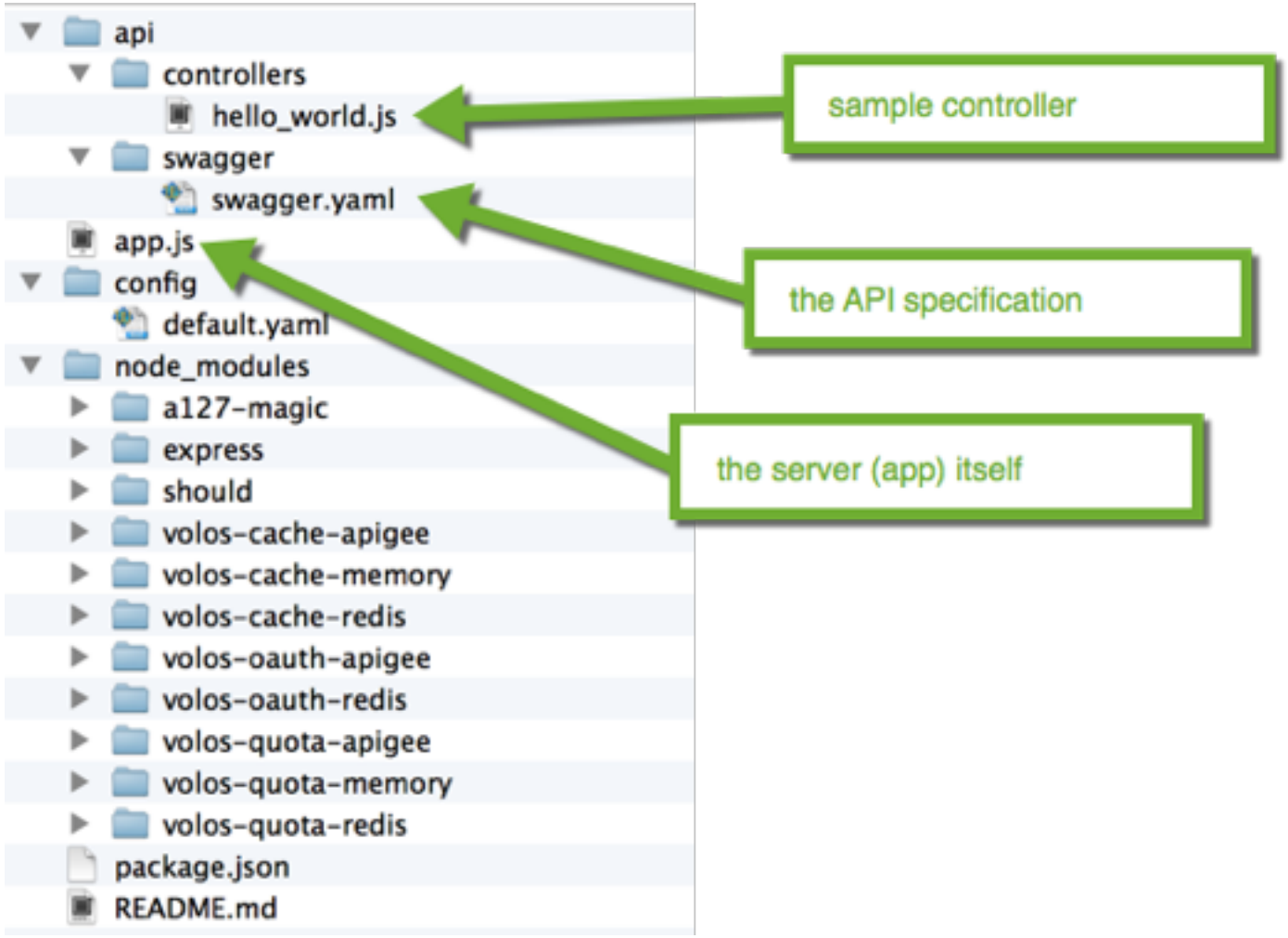
- Using the -g option installs Apigee 127 as a globally available app – this is recommended

Creating an Apigee 127 Project

- Using the command line, navigate to the place where you want to create a new project – note, this will create a folder where you will be working throughout the workshop

```
LapGARbage:Development greg$ a127 project create api-workshop
creating: .npmignore
creating: README.md
creating: api
creating: app.js
creating: config
creating: package.json
creating: api/controllers
creating: api/swagger
creating: config/default.yaml
creating: api/controllers/hello_world.js
```

Apigee 127 App Structure



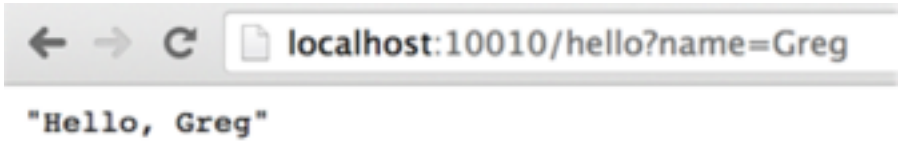
apigee

Try it out

- Navigate inside the project folder, then execute the command **`a127 project start`**

```
LapGARbage:api-workshop greg$ a127 project start
starting with account settings: workshop
deployment files will remain after exiting: /Users/greg/Development/api-workshop
/config/.a127_env,/Users/greg/Development/api-workshop/config/.a127_secrets
Starting: /Users/greg/Development/api-workshop/app.js...
  project started here: http://localhost:10010
  project will restart on changes.
  to restart at any time, enter `rs`
connect: multipart: use parser (multiparty, busboy, formidable) directly
connect: limit: Restrict request size at location of read
usergrid
try this:
curl http://localhost:10010/hello?name=Scott
```

- Open a browser window and navigate to: **`http://localhost:10010/hello?name=YOUR-NAME-HERE`**

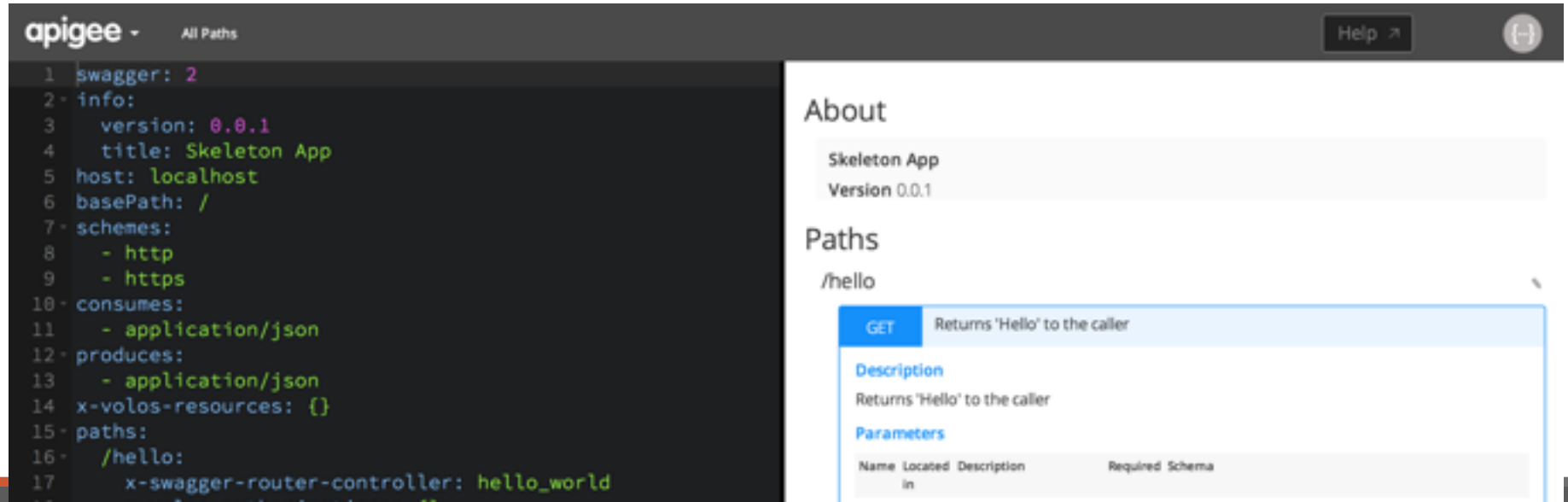


apigee

Launching the Editor

- Launch the local Swagger editor by running **`a127 project edit`**

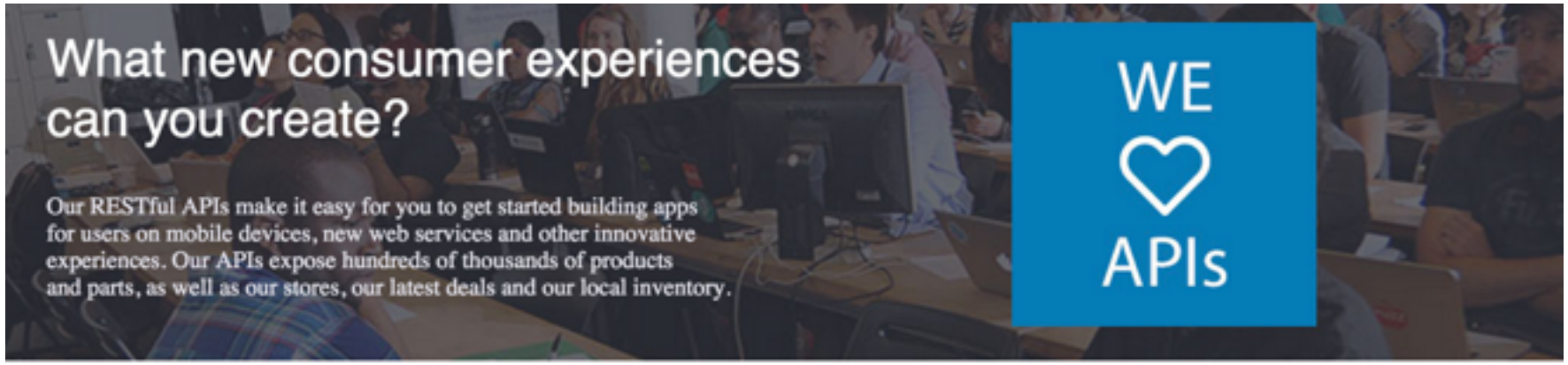
```
LapGARbage:api-workshop greg$ a127 project edit
Swagger editor running: http://localhost:64268/
opening browser to: http://localhost:64268/
```



Building an API Proxy

Sears Developer Portal

- For this example, we'll use the Product Search API



Home

Sears APIs

Product Search API

Use the Product Search API to find products based on keyword and filtered by other parameters, and to fetch product category information.

Learn More

Product Details API

Use the Product Details API to retrieve detailed information on products from colors and brands, to collections and lines.

Learn More

Top Sellers API

Use the Top Sellers API to retrieve product information on the top sellers at Sears and Kmart Stores.

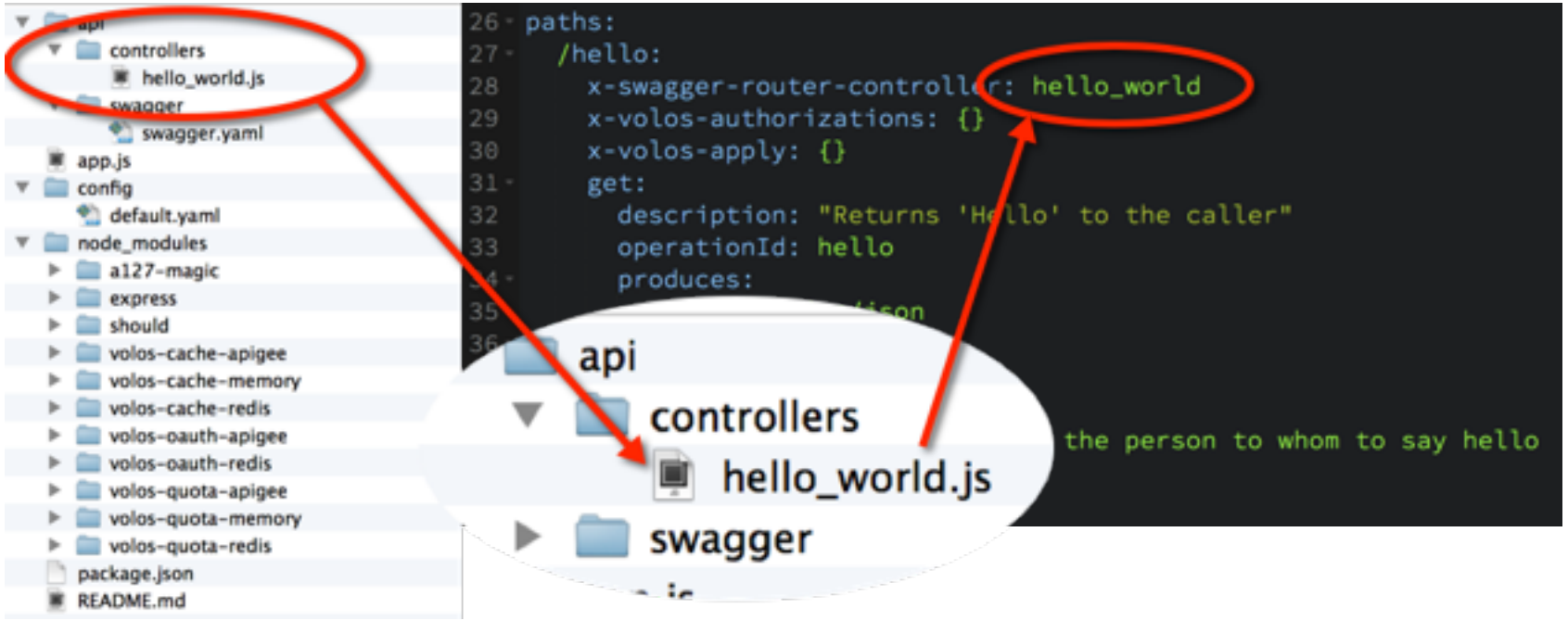
Learn More

Create a definition “/products” in the Swagger doc

- Copy/paste the /hello definition
- Change the route, x-swagger-router-controller and operationId to something that makes sense for you

http://j.mp/products-definition-v1

How Do Controllers Work?



- All controllers are stored in the api/controllers folder
- The name of the controller (.js file) must match the **`x-swagger-router-controller:`** value in the swagger doc

How Do Controllers Work?

- Each controller exports an array of name/value pairings where the name corresponds to the **operationId:** in the swagger document and the value is the function within the controller

```
'use strict';

var util = require('util');

module.exports = {
  hello: hello
};

function hello(req, res) {
  var name = req.swagger.params.name.value;
  var person = name ? util.format('Hello, %s', name) : 'Hello, stranger!';
  res.json(person);
}
```

```
paths:
  /hello:
    x-swagger-router-controller: hello_world
    x-volos-authorizations: {}
    x-volos-apply: {}
    get:
      description: Returns 'Hello' to the caller"
      operationId: hello
      produces:
        - application/json
      parameters:
        - name: name
          in: query
          description: The name of the person to whom to say hello
          required: false
          type: string
```

The products.js controller

- Create a file **`products.js`** (or whatever you called the x-swagger-router-controller) and save it in the **`api > controllers`** folder
- Require the request module
- Add a module.exports JSON object to map the operationID from the swagger file to an internal function in the controller

The products.js controller

- Install the request module and add it to the package.json file
  `npm install request --save`
- Use the request module to make a call to the Sears Product Search API to search for a washer

  http://j.mp/sears-product-search-v1

- Test your API by going to
  `http://localhost:10010/products`

Make the API flexible

Currently, your API only returns washers. Let's make it a little more flexible by adding variables — or parameters as we call them

- Add a path parameter to /products
- Add a limit parameter to the GET request on /products
- Use `req.swagger.params.{Parameter}.value` to grab the variable in the controller

http://j.mp/products-definition-v2

http://j.mp/sears-product-search-v2

Test your API with different combinations

```
http://localhost:10010/products/dryer?limit=5
http://localhost:10010/products/washer?limit=20
```

Deploying the APIs

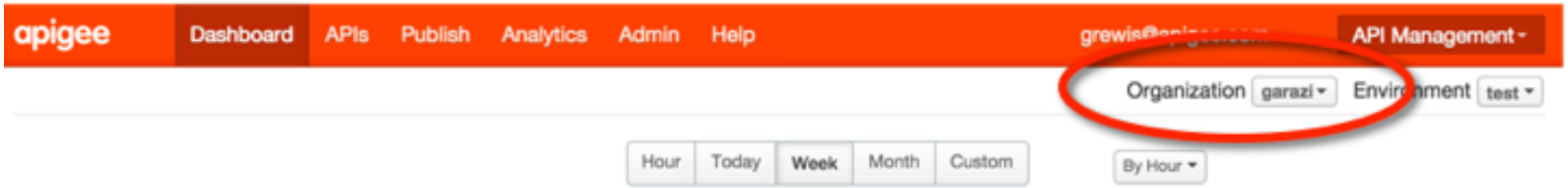
Deploying

An a127 account is needed to deploy to the Apigee Edge platform. You only need to create the account once, and it will be valid for all of your a127 projects.

- Create an a127 account
  `a127 account create {NAMEYOURACCOUNT}`
- Select apigee as the provider and follow the prompts
- For the BaseURL and Virtual hosts options, simply hit Return

Deploying

- For organization, log into your Apigee account, click on the API Management panel. Your organization is in the upper righthand corner.
- For userID, use the email address that you used when you signed up for your Apigee account



Deploying

- To deploy to Apigee Edge
  `a127 project deploy`
- After the upload is complete copy the URI of your APIs

```
LapGARbage:api-workshop greg$ a127 project deploy
Deploying project api-workshop to apigee...
name: api-workshop
environment: test
revision: 1
state: deployed
basePath: /
uris:
  - 'http://grewis-test.apigee.net/api-workshop'
```

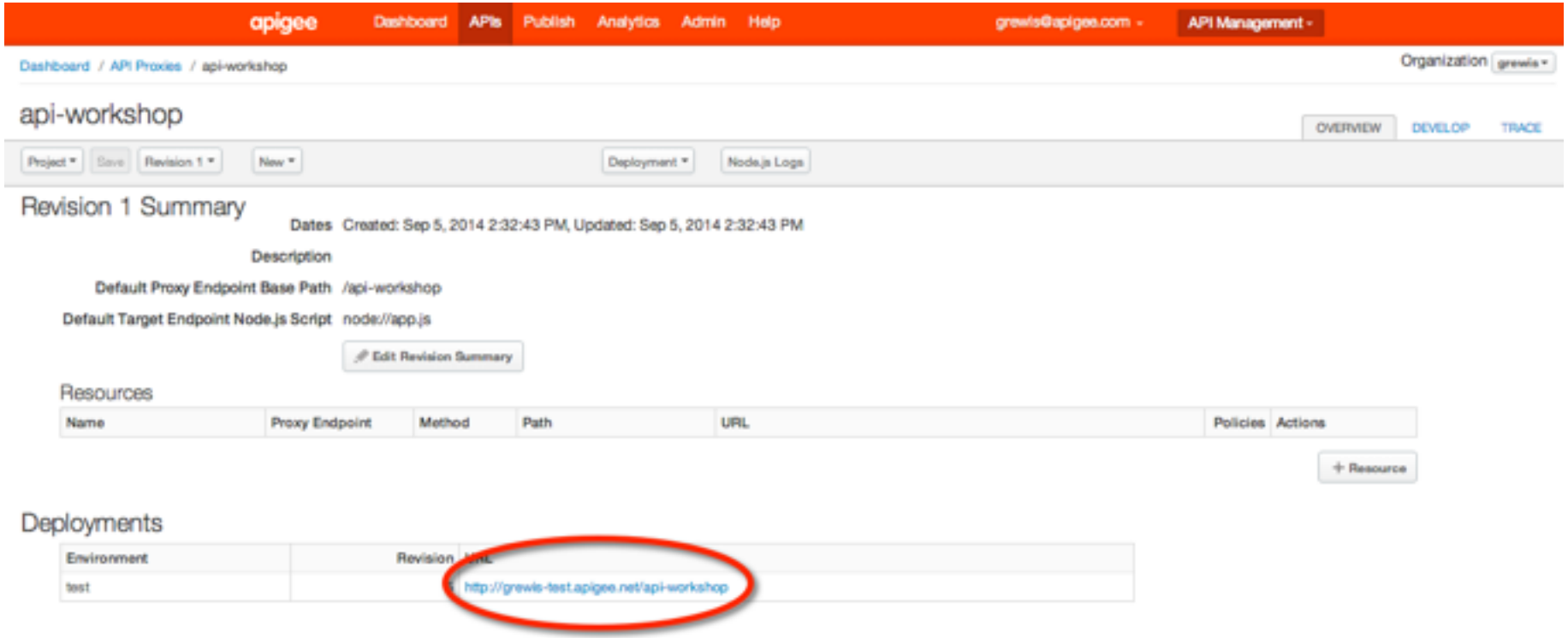
Deploying

- If the upload times out — which can happen due to the server needing to run npm install — you can chose to have it upload your node_modules using the -u modifier

```
a127 project deploy -u
```

Congratulations!

- Open the API Management console and choose API Proxies from the API menu
- Click on your API to view it



apigee

Get more training

Advanced class?

Training for your company? Still free!

grewis@apigee.com

Congratulations!

I don’t accept tips but I do accept tweets!

@garazi #apigee

grewis@apigee.com

apigee

Thank you