OpenAPI Training

Pre-requisites

- Basic knowledge of REST

Why APIs and what simplifies their delivery?

- APIs have been proven to be the number one mechanism for electronic business
- APIs identify what they provide via their interface
- API Consumption has been proven to increase when interface is clean, simple and well documented.
- API eco-system has standardized on OpenAPI as the defacto format
    - API Design
    - API Testing
    - API Virts
    - API Implementation
    - API Management

What is OpenAPI?

- A YAML or JSON definition of a REST API
- The API Documentation standard
- Standardized language to articulate HTTP, URI, MIME semantics
- Eco-system of tooling to consume OpenAPI doc and provide additional value

History

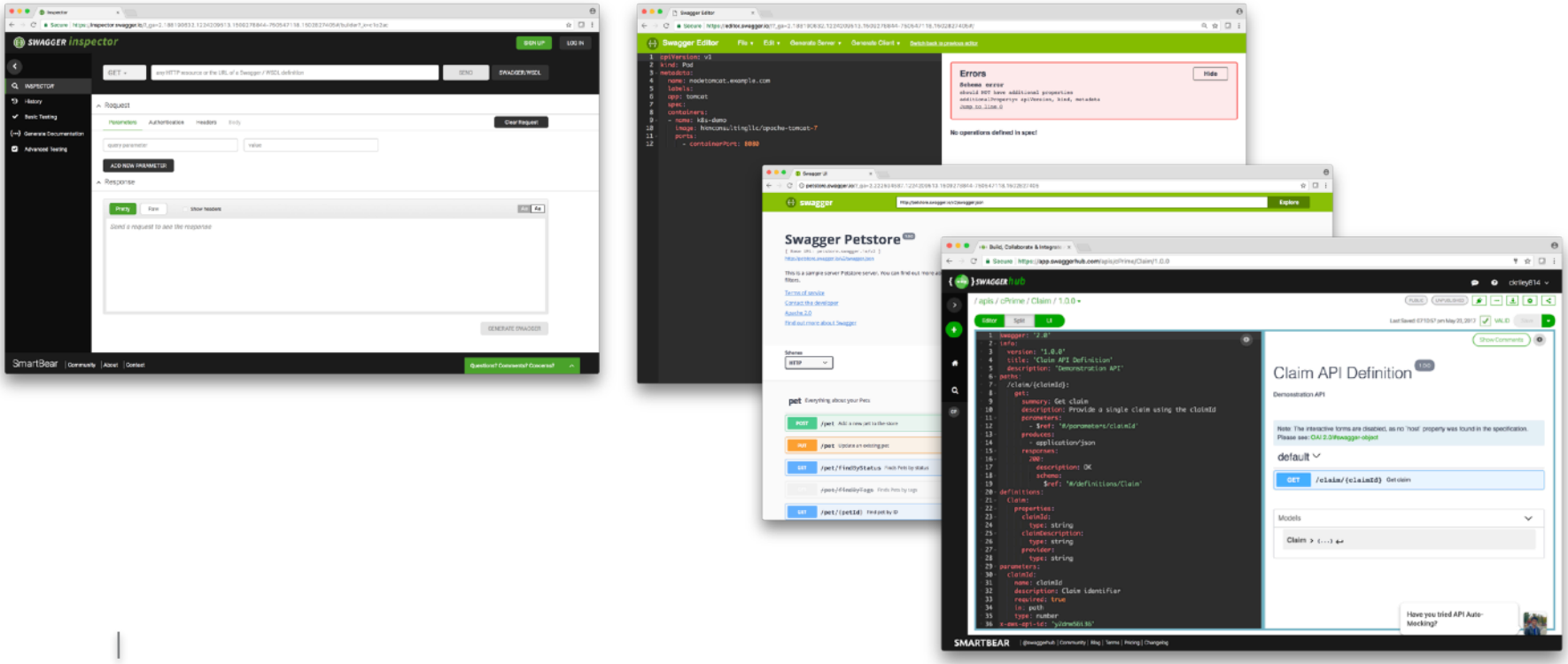
- 2010 - Wordnik creates Swagger to help describe APIs.
- 2015 – Smartbear acquires Swagger and assets from Reverb.
- 2015 – Smartbear, Google, 3Scale, Apigee and others create OpenAPI Initiative with Swagger 2.0 as basis and renames OpenAPI 2.0.
- 2017 – OpenAPI releases 3.0 of OpenAPI specification.

Who can use OpenAPI?

- Anyone who participates in the evolution of API based products / services
  - Product Manager
  - Business Analyst
  - Software Architect
  - QA Engineer
  - Lead Developer
  - Third-party Developers

Smartbear Swagger Tooling

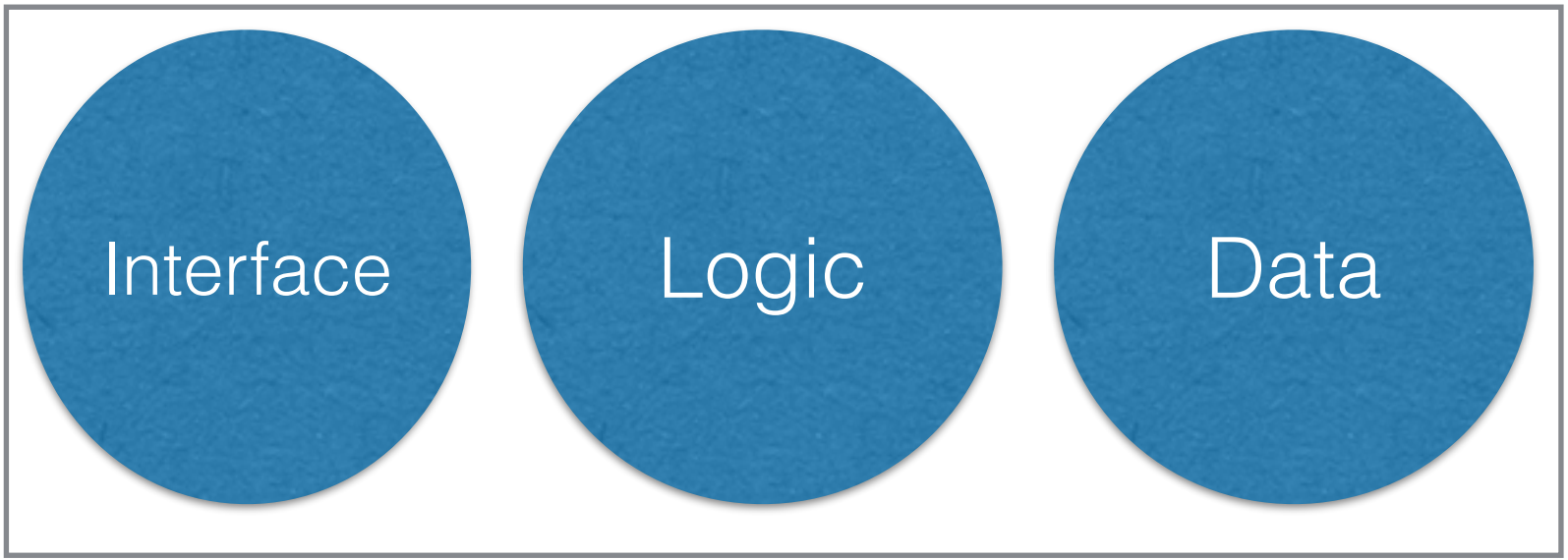
Inspector, Editor, Codegen, UI, SwaggerHub



What is a REST API?

- Representational State Transfer, Roy Fielding Dissertation 2000
- Uses HTTP, URI and HTML to share information
  - HTTP = How do I communicate?
  - URI = Where is it available to communicate?
  - HTML = What data am I sharing?

What is a Service?



Interface Examples

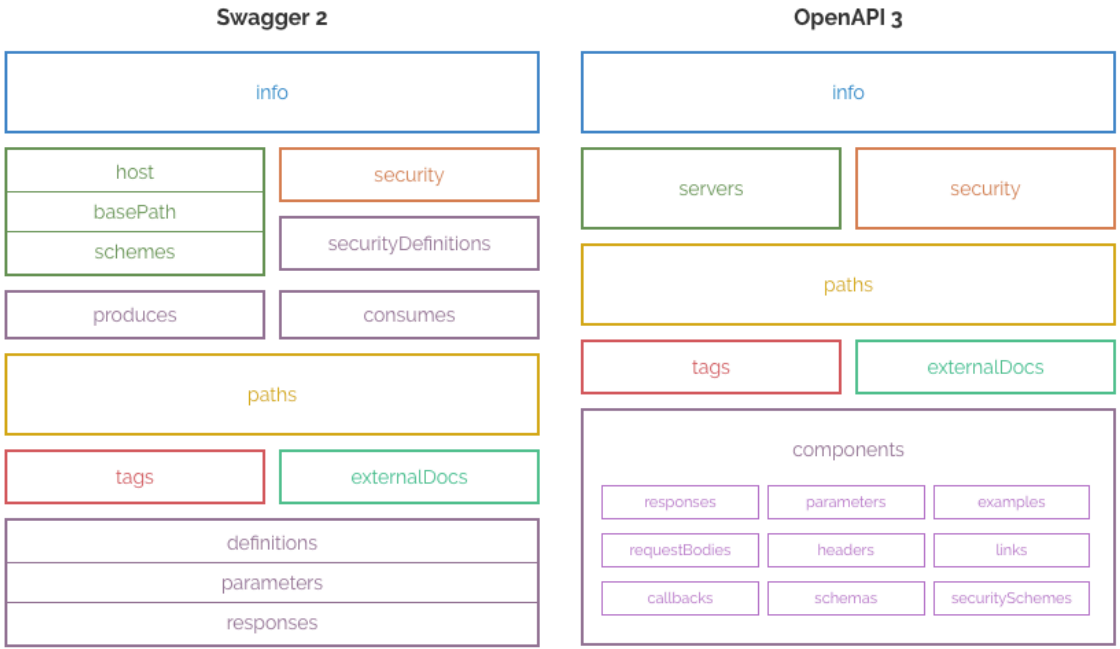
- IDL - CORBA
- WSDL / XML Schema – SOAP Based Services
- WADL – Yahoo Web Application Description Language
- RAML / Blueprint / Swagger
- OpenAPI / JSON Schema – Linux Foundation

Swagger 2 to OpenAPI 3

Restructuring / Simplification

OpenAPI 2.0 to OpenAPI 3.X

Image from: https://blog.readme.io/an-example-filled-guide-to-swagger-3-2/

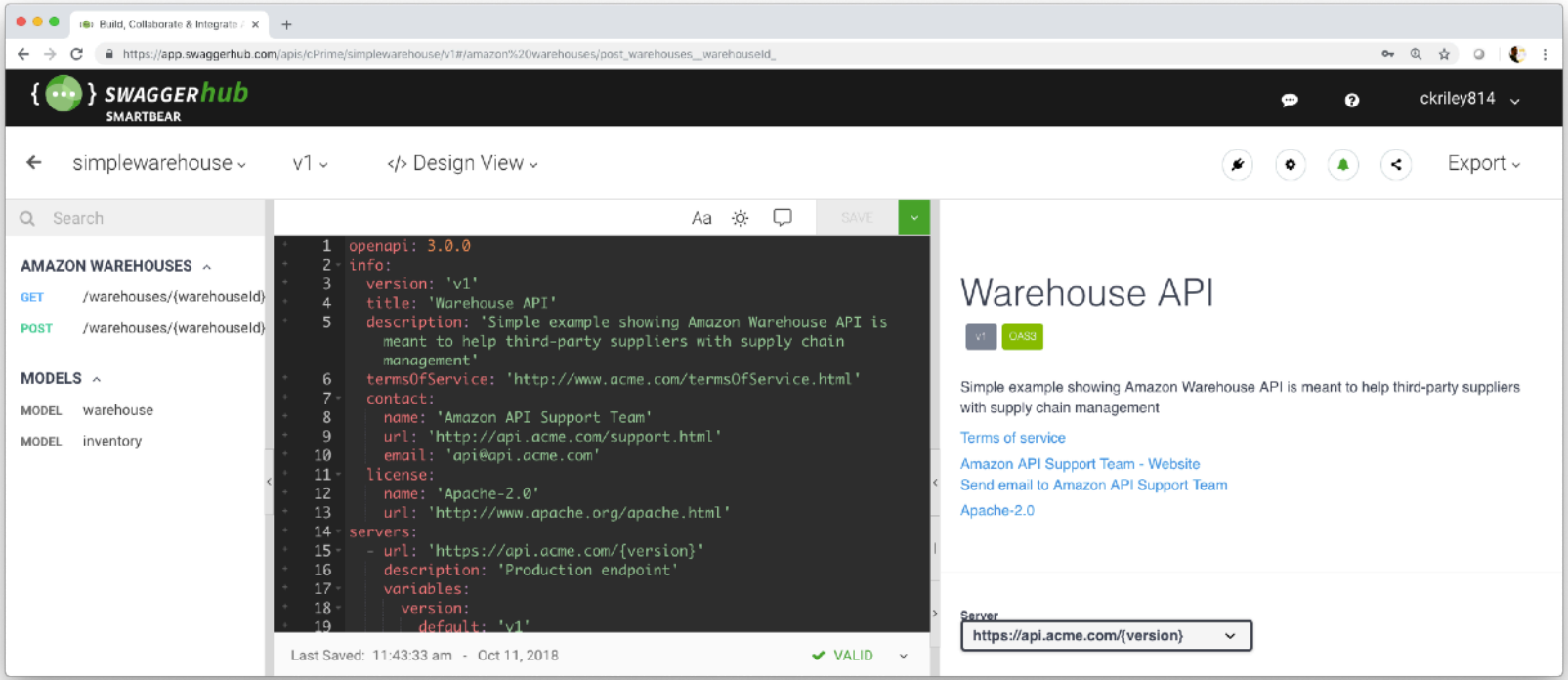


Class Prep

Login to SwaggerHub

SwaggerHub

https://swagger.io/tools/swaggerhub/



OpenAPI / Example Overview

API Documentation Standard

Training Exercise

Amazon Warehouse: Tech Insider https://www.youtube.com/watch?v=5TL80_8ACPc

Characteristics

- Chaotic Storage, fill up available space quickly / evenly
- Every item that is stocked is scanned and where it is located is scanned
- Database stores these details and generates manifest for pickers to create a shipment
- Workers use hand-held device to read manifest of product and locations.

Data

- Warehouse Identifier
- Stock Keeping Unit (SKU) – Barcode based
- Location in Warehouse – Barcode based
- Quantity
- https://app.swaggerhub.com/apis/cPrime/simplewarehouse/v1

Info Object

Metadata about the API

Info Object

- Metadata about the API
- Who, what, where (contact, license, hostname information)
- API Custodian

Info Object Example

```yaml
openapi: 3.0.0
info:
  version: 'v1'
  title: 'Warehouse API'
  description: 'Simple example showing Amazon Warehouse API is meant to help third-party suppliers with supply chain management'
  termsOfService: 'http://www.acme.com/termsOfService.html'
  contact:
    name: 'Amazon API Support Team'
    url: 'http://api.acme.com/support.html'
    email: 'api@api.acme.com'
  license:
    name: 'Apache-2.0'
    url: 'http://www.apache.org/apache.html'
```

Info Object Exercise

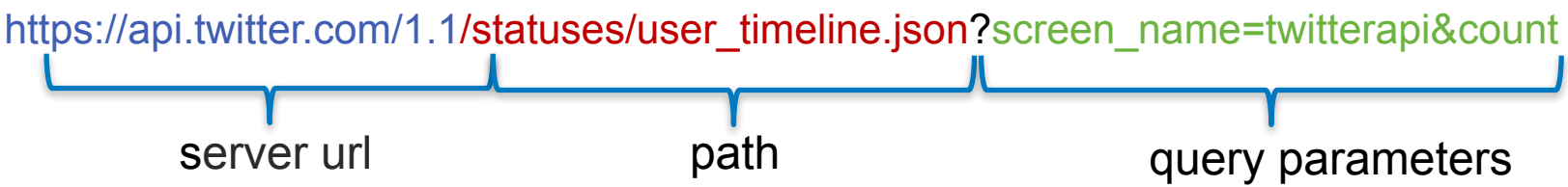
- Create *openapi = 3.0* entry
- Create an Info Object

Servers Object

Where is the API

Servers Object

- Answers the where question regarding the API
- One or more server / hostnames are provided along with additional meta-data such as ports, basePath or other URI variables



Servers Object Example

```yaml
servers:
  - url: 'https://api.acme.com/{version}'
    description: 'Production endpoint'
    variables:
      version:
        default: 'v1'
  - url: 'https://testapi.acme.com/{version}'
    description: 'Test endpoint'
    variables:
      version:
        default: 'v2'
        enum: ['v1','v2']
```

Servers Object Exercise

- Create a Servers Object

Components Object

Supporting data / security definitions for the API

Components Object

- Encompasses schemas, parameters, securitySchemes, requestBody, responses, headers, examples, links, callbacks
- Focuses on reusable elements and the uses reference to component in appropriate section

Schemas Object

API Data

Schemas Object

- Using JSON Schema, the Schemas Object identifies the JSON payloads that are received or sent by the API.
- Supports different data types including string, number, boolean and further classification such as int32, int64, dateTime.
- Optionality, sequencing, descriptions and other meta-data provide a means to clearly define the JSON payload.

Schemas Object Example

```yaml
schemas:
  warehouse:
    type: 'object'
    description: 'warehouse meta-data including location and inventory'
    properties:
      warehouseId:
        description: 'warehouse identifier that identifes its location'
        type: 'string'
      inventories:
        type: 'array'
        description: 'current inventory of items available in the warehouse'
        items:
          $ref: '#/components/schemas/inventory'
```

Data Types

Base Data Types

Data Type	Examples
string	“hello”, c:/file.txt, 2017-11-12
number	Any number
integer	1,15,22
boolean	True / False
object	JSON Object
array	JSON Array

Note: Mixed types are supported via oneOf, anyOf

Number with format

How to further articulate specific number / integer types

Data Type	Format	Description
number	--	Any number
number	float	Floating-point numbers
number	double	Floating-point numbers with double precision
integer	--	Integer numbers
integer	int32	Signed 32 bit integers
integer	int64	Signed 64 bit integers (long type)

String with format

How to further articulate string formatting

Data Type	Format	Description
string	date	Date notation e.g. 2017-11-12
string	date-time	Date time notation e.g. 2017-11-12T12:00:00Z
string	password	Notify UI to mask
string	byte	Base64 encoded characters
string	binary	Binary data used to describe files.

Minimum, Maximum, Multiples

- Minimum, Maximum value constraints can be specified
- Multiples can be used to constrain a value to a multiple
- ExclusiveMinimum and ExclusiveMaximum excludes the value

```
type: integer
minimum: 5
maximum: 35
multiple: 5
exclusiveMaximum: true
```

minLength / maxLength / pattern

- Constrain a string to have a minimum / maximum length
- Apply a pattern to constrain a string

```
zip:
  type: string
  pattern: '^[0-9]{5}(?:-[0-9]{4})?$'
  maxLength: 10
```

Schemas Object Exercise

- Create a Schemas Object

Paths Object

API Capabilities

Paths Object

- What the API Provides
- URL, MIME Types, Parameters Object, Responses Object

Determine Interaction

- What Verb, Request, Response needs to be identified?
- Parameters as input?
- RequestBody as input (e.g. JSON)?

Paths Object Detailed Example

```yaml
paths:
  /warehouses/{warehouseId}:
    get:
      description: 'Get all inventory in warehouses'
      tags:
        - 'amazon warehouses'
      parameters:
        - $ref: '#/components/parameters/warehouseId'
      responses:
        200:
          description: 'OK'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/warehouse'
```

Paths Object Exercise

- Create a Paths Object

$ref

Reference capabilities in OpenAPI

$ref overview

- OpenAPI uses $ref for referencing information within an OAS spec, outside on the file system or at a URL.
- $ref: '#/components/schemas/elementName'
- $ref: '../person.json#elementName'
- $ref: 'http://path to resource/#elementName'

Parameters Object

Request Data for API

Parameters Object

- Identifies where in the request the inbound data will reside
- Path, Query, HTTP Header, Cookies
- Data can be stored in a several locations (e.g. Path, Query + Header)
- Can be defined under Path or at the root level for reuse

Parameter Object Example

```yaml
parameters:
  warehouseId:
    name: warehouseId
    description: 'identifier for warehouse'
    in: path
    required: true
    schema:
      type: integer
      enum: [10001,10002,10003]
```

Parameters Object Exercise

- Create a Parameters Object

Media Types

Describe the API Content

Media Type for Requests/Responses

- MIME Type definition for a request or response
- Accept: application/json
- Content-Type: plain/text

Media Type Object Example

application/json

```yaml
content:
  application/json:
    schema:
      $ref: '#/components/schemas/warehouse'
```

Media Type Object Example

File Upload

```
content:
  video/mp4:
    schema:
      type: string
      format: binary
```

Media Type Object Example

Multipart

```yaml
requestBody:
  content:
    multipart/form-data:
      schema:
        type: object
        properties:
          geolocation:
            type: string
          video:
            type: string
            format: binary
```

Media Type Object Exercise

- Create a Media Type Object

Responses Object

Describe the API Responses

Responses Object

- Identifies how the data will be presented back to consumer
- MIME Type
- Associated Definition Object

Responses Object Example

```yaml
responses:
  200:
    description: OK
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/warehouse'
```

Response Exercise

- Create responses for both GET and POST
- For GET create 200, 4XX, 5XX
- For POST create 201 and 5XX

RequestBody Object

Describe the API Request Body

requestBody Object

- POST, PUT, PATCH
- Distinguishes Payload from Parameters

requestBody Example

```yaml
requestBody:
  description: Inventory object
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/inventory'
responses:
  201:
    description: Created
```

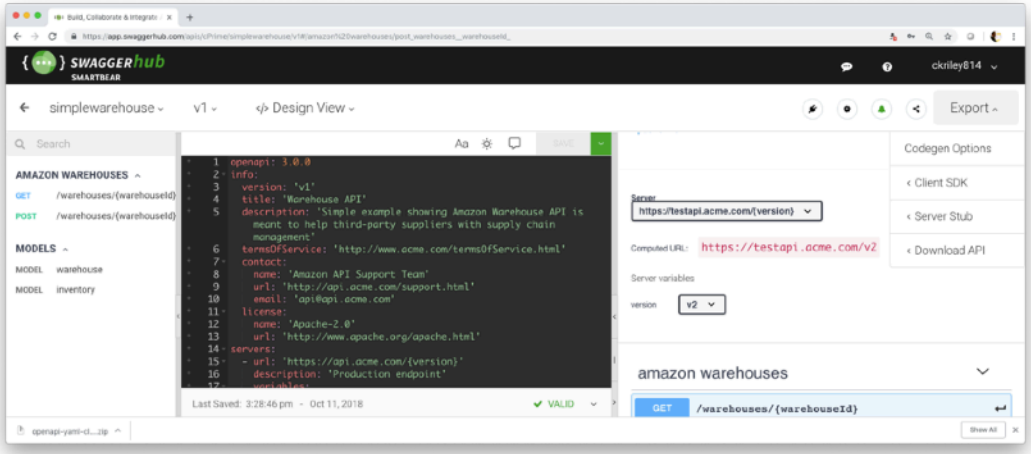
requestBody Object Exercise

- Create a requestBody Object

Completed Example

Version 0.1 of Warehouse API

SwaggerHub -> Spring Boot -> Docker -> GKE -> ReadyAPI



Summary

- API Economy continues to represent backbone of Enterprise
- Clear articulation of API is critical
- OpenAPI has been adopted as the way to communicate
- OpenAPI eco-system is increasing every day
- Participate, it is an open forum!

Thank you!