

# OpenAPI development with Python



EuroPython2017@Rimini, 11 July 2017

Takuro Wada

# Hi



## Takuro Wada

Kabuku Inc.

**Software Engineer**

- Speaker of EuroPython 2016, PyConJP 2015
- Member of swagger codegen technical committee
  - Python, TypeScript



**@taxpon**



**taxpon**



**<http://takuro.ws>**

# Agenda

## 1. What is OpenAPI?

Introduction and basics



## 2. OpenAPI tools

Introduce some tools to increase your productivity



## 3. Actual case study

Introduce our company's project

# What is OpenAPI?

# What is OpenAPI?

OpenAPI is **API description language** which is focusing on creating, evolving and promoting vendor neutral description format

(Partially cited from <https://www.openapis.org/about>)

You can **write your API spec with OpenAPI**

# What is OpenAPI?

- ▶ Supported spec format
  - YAML and JSON
- ▶ Based on JSON schema
  - Vocabulary to annotate and validate JSON
- ▶ Originally known as Swagger

```
paths:
  /books:
    get:
      tags: [Books]
      operationId: getBooksList
      responses:
        '200':
          description: success
          schema:
            title: books
            type: array
            items:
              $ref: '#/definitions/Book'
    post:
      tags: [Books]
      operationId: createBook
      parameters:
        - name: book
          in: body
          description: book parameter
          required: true
          schema:
            $ref: '#/definitions/Book'
```

Actual example spec in YAML format

# How to use OpenAPI?

## ▶ As API documents

- Generate good looking documents
  - Share API spec in team
    - Frontend and Backend
    - Developers and non-developers
- Public API Document for Any developers

# How to use OpenAPI?

## ▶ As API tools

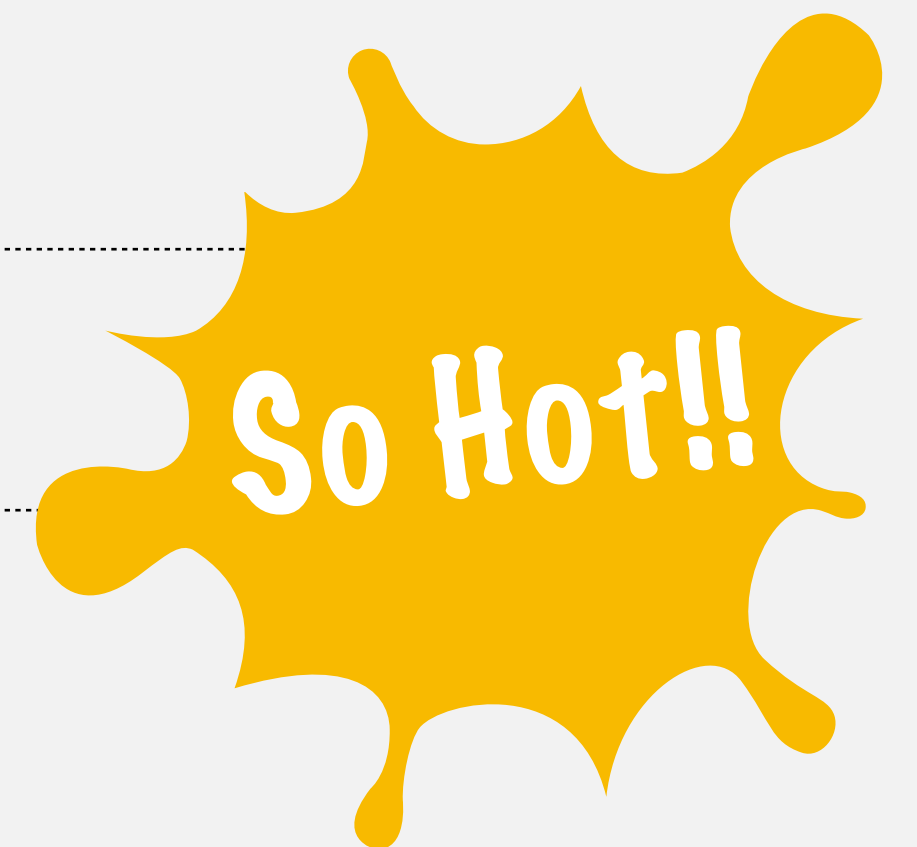
- Code generation
  - Codes for request data validation in server
  - Code for API calling in clients



# OpenAPI versions

- ▶ OpenAPI is originally known as Swagger
  - Swagger spec was renamed OpenAPI spec in 2016

| Version | Release Date  |
|---------|---|
| 1.2     | 14 Mar 2014   |
| 2.0     | 8 Sep 2014  |
| 3.0     | Will be released in <b>July 2017</b><br><a href="https://www.openapis.org/specification/repo">https://www.openapis.org/specification/repo</a> |



# OpenAPI's competitors

## ► RESTful API DL (Description Language)

- RAML (RESTful API Modeling Language)

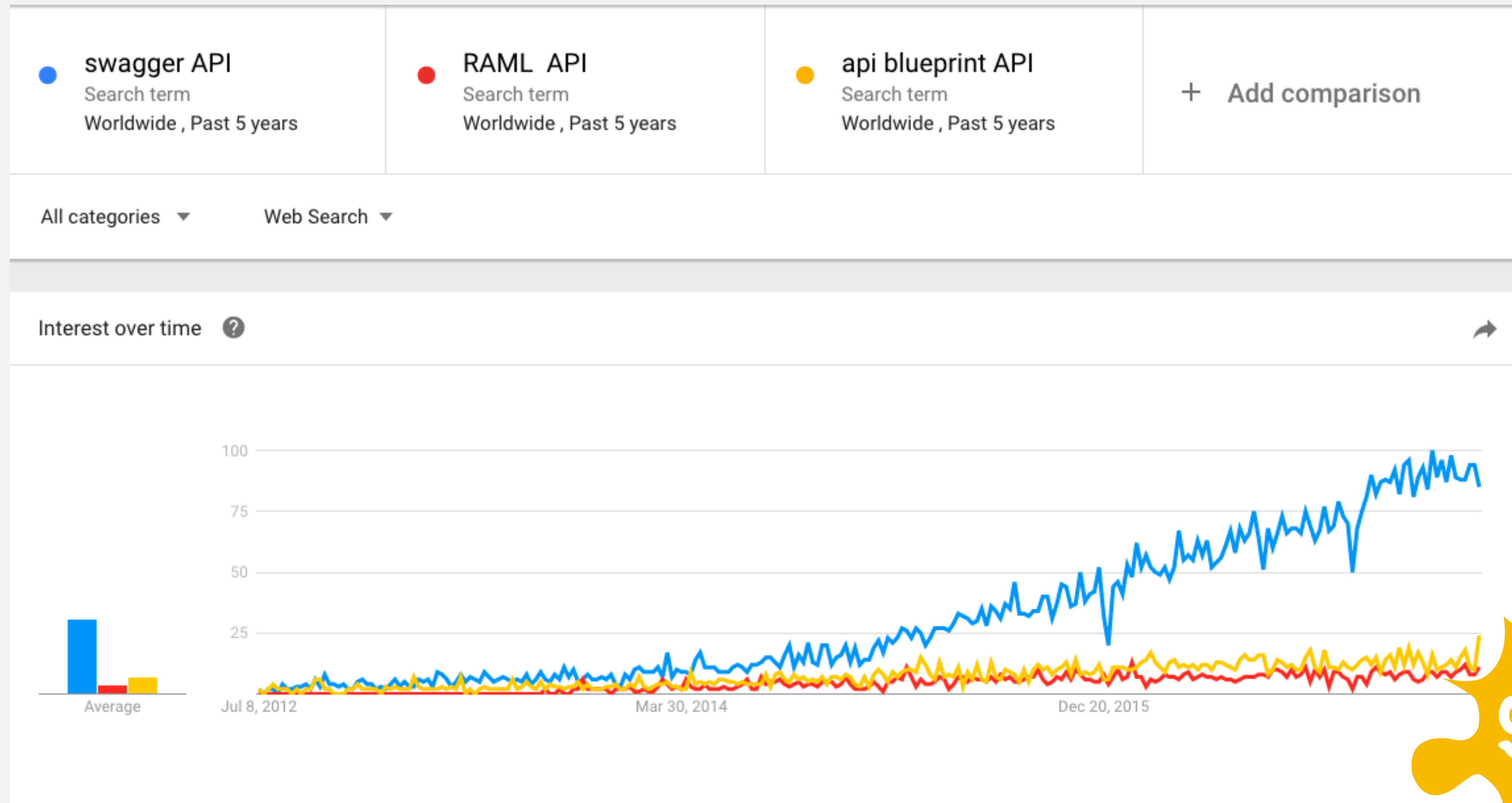
  - YAML base

- API Blueprint

  - Markdown base

  - Oracle acquired Apiary in Jan 2017

# Google Trends



**OpenAPI (Swagger) is gathering more attention than others!**

**So Hot!!**

# OpenAPI tools

# OpenAPI tools

## ▶ Core tools

- Developed by OpenAPI (Swagger) team

## ▶ Community tools

- Developed by Community
- Introduce Python tools in this session

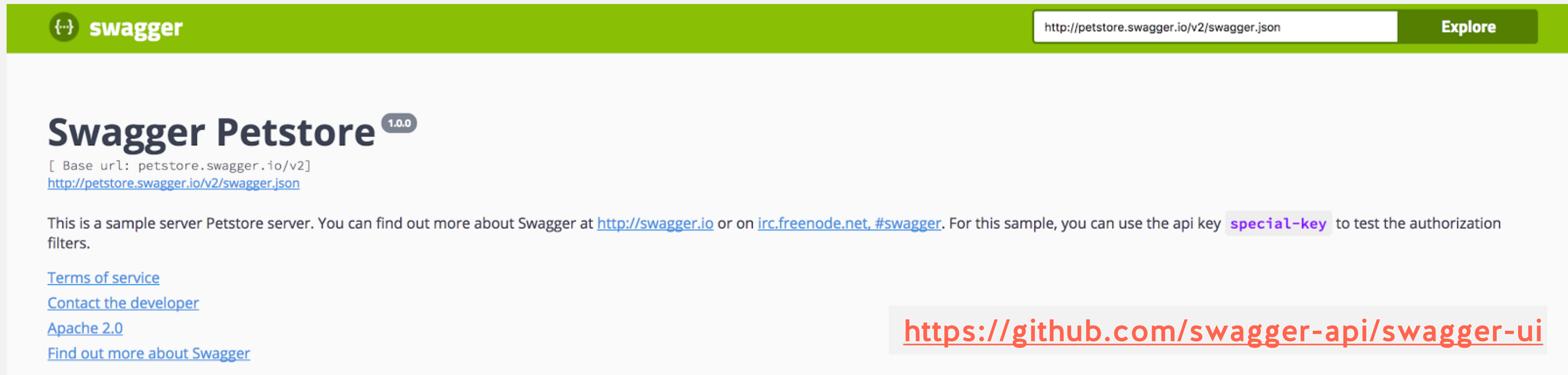
# Core tools

- ▶ Swagger UI
- ▶ Swagger Editor
- ▶ Swagger Codegen

# Core tools (1/3)

## ► Swagger UI

- Show spec with beautiful format
- Directly API calling from browser
- <http://petstore.swagger.io/>



The screenshot shows the Swagger UI interface. At the top, there is a green header bar with the Swagger logo on the left and a search bar on the right containing the URL `http://petstore.swagger.io/v2/swagger.json` and an **Explore** button. Below the header, the main content area displays the title **Swagger Petstore** with a **1.0.0** version tag. Underneath the title, it shows the base URL `[ Base url: petstore.swagger.io/v2 ]` and a link to the Swagger JSON file. A paragraph of text explains that this is a sample server and provides links to the Swagger website and IRC channel, along with an API key `special-key` for testing. At the bottom left, there are links for [Terms of service](#), [Contact the developer](#), [Apache 2.0](#), and [Find out more about Swagger](#). At the bottom right, there is a red link to the GitHub repository: <https://github.com/swagger-api/swagger-ui>.

# Core tools (2/3)

## ► Swagger Editor

- WYSIWYG Spec editor in web browser
  - Syntax highlighting, Autocompletion, Real time spec validation
  - <http://editor.swagger.io/#/>

The screenshot displays the Swagger Editor web interface. On the left, a code editor shows a Swagger 2.0 specification for the 'Swagger Petstore (Simple)' API. The specification includes details such as version (1.0.0), title, description, terms of service, contact information, license (MIT), host (petstore.swagger.io), base path (/api), and a list of paths. The right panel provides a visual representation of the specification, including the title 'Swagger Petstore (Simple)', a description, version, contact information, terms of service, license, and a list of paths. A red box highlights the URL <https://github.com/swagger-api/swagger-editor> in the 'License' section.

```
1 swagger: '2.0'
2 info:
3   version: '1.0.0'
4   title: Swagger Petstore (Simple)
5   description: A sample API that uses a petstore as an example to demonstrate features in the swagger-2.0
6     specification
7   termsOfService: http://helloverb.com/terms/
8   contact:
9     name: Swagger API team
10    email: foo@example.com
11    url: http://swagger.io
12  license:
13    name: MIT
14    url: http://opensource.org/licenses/MIT
15  host: petstore.swagger.io
16  basePath: /api
17  schemes:
18    - http
19  consumes:
20    - application/json
21  produces:
22    - application/json
23  paths:
24    /pets:
25      get:
26        description: Returns all pets from the system that the user has access to
```

Swagger Petstore (Simple)

A sample API that uses a petstore as an example to demonstrate features in the swagger-2.0 specification

Version 1.0.0

Contact information  
Swagger API team  
foo@example.com  
http://swagger.io

Terms of service  
http://helloverb.com/terms/

License  
MIT

<https://github.com/swagger-api/swagger-editor>

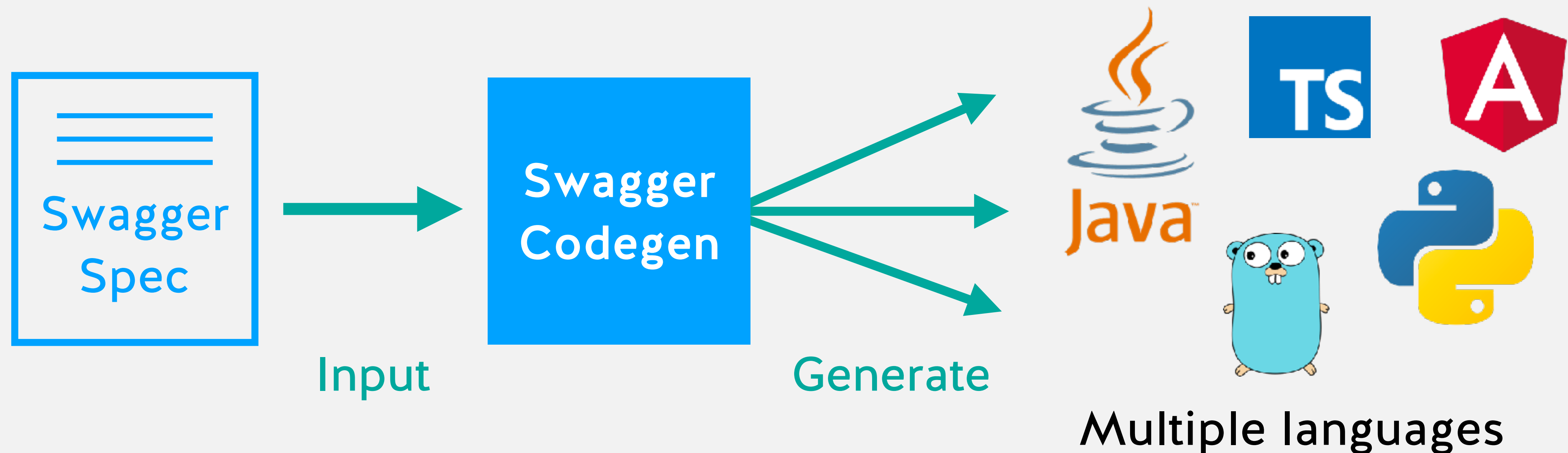
Paths  
/pets



# Core tools (3/3)

## ► Swagger Codegen

- Generate server's and client's code from spec



# Community tools

## ► There are many Python tools for OpenAPI

- Validator
- Code generator
- Spec parser
- Some tools are for the specific framework

| Python                                 |  |
|--|--|
| Name                                   | Description  |
| <a href="#">falsy</a>                  | with FAL.S.Y, you can use falcon, swagger-ui, yml together, which makes writing api easy!      |
| <a href="#">pyramid-swagger</a>        | Convenient tools for using Swagger to define and validate your interfaces in a Pyramid webapp. |
| <a href="#">flask-restplus</a>         | Helpers, syntactic sugar and Swagger documentation for Flask-Restful                           |
| <a href="#">pyswagger</a>              | A type-safe, dynamic, spec-compliant Swagger client.   |
| <a href="#">flex</a>                   | Swagger 2.0 schema validation, and tooling for validating arbitrary request/response objects.  |
| <a href="#">Flasgger</a>               | Flask Extension to provide Swagger 2.0 to any view using docstrings (embeds swagger UI)        |
| <a href="#">flask-swagger</a>          | A Swagger 2.0 extractor for Flask via YAML in docstrings                                       |
| <a href="#">bravado</a>                | Swagger 2.0 client with support for both synchronous and asynchronous http.                    |
| <a href="#">bravado-core</a>           | Library for Swagger 2.0 schema ingestion, validation, request/response validation, etc.        |
| <a href="#">swagger-spec-validator</a> | Library for validating Swagger 1.2 and 2.0 schemas.  |
| <a href="#">swagger-py-codegen</a>     | Generate Flask-RESTful application code from a Swagger Specification doc.                      |
| <a href="#">Connexion</a>              | Swagger-first REST framework on top of Flask with validation and OAuth 2 support.              |
| <a href="#">pecan-swagger</a>          | Partial swagger extractor for pecan.   |
| <a href="#">swagger-parser</a>         | Give useful informations about your swagger files.   |
| <a href="#">swagger-tester</a>         | Automatic swagger API tester.  |
| <a href="#">swagger-aggregator</a>     | Aggregate several swagger APIs in one.   |
| <a href="#">swagger-stub</a>           | Generate a stub from a swagger file.   |
| <a href="#">bottle-swagger</a>         | Swagger integration for the Bottle web framework   |
| <a href="#">prance</a>                 | Swagger parser that resolves JSON references.  |

<https://swagger.io/open-source-integrations/#python-19>

# bravado-core

- ▶ Python library that adds client-side and server-side support for OpenAPI (2.7, 3.4+, developed by Yelp, <https://github.com/Yelp/bravado-core>)
- ▶ Not dedicated to any specific framework (You can use it in you own project today)
- ▶ Very simple to use
- ▶ Features
  - Validation
  - Marshaling and Unmarshaling
    - Custom formats for type conversion

# bravado-core

## ► Spec example

```
Book:  
  type: object  
  required: [id]  
  properties:  
    id:  
      type: integer  
    title:  
      type: string  
    author:  
      type: string
```

# bravado-core: (1)Validate

## ► Validation execution

```
import yaml
from bravado_core.spec import Spec

# 1
with open('openapi.yaml', 'r') as f:
    raw_spec = yaml.load(f)
# 2
spec = Spec.from_dict(raw_spec)
# 3
book = raw_spec['definitions']['Book']
# 4
validate_schema_object(spec, book, target)
```

1. Load YAML file with OpenAPI spec (JSON is also OK)
2. Create Spec object
3. Retrieve “Book” definition
4. Validate (target is dict object which is dumped from client's request)

# bravado-core: (1)Validate

► if required property “id” is not defined in dict:

```
validate_schema_object(spec, book, {})
```

Code

```
jsonschema.exceptions.ValidationError: 'id' is a required property
```

Result

```
Failed validating 'required' in schema:
  {'properties': {'author': {'type': 'string'},
                  'id': {'type': 'integer'},
                  'release_date': {'format': 'date', 'type': 'string'},
                  'title': {'type': 'string'}},
   'required': ['id'],
   'type': 'object',
   'x-model': 'Book'}
```

```
On instance:
  {}
```

# bravado-core: (1)Validation

► If a property has invalid type value:

```
validate_schema_object(spec, book, {"id": 1, "title": 1})
```

Code

```
jsonschema.exceptions.ValidationError: 1 is not of type 'string'

Failed validating 'type' in schema['properties']['title']:
    {'type': 'string'}

On instance['title']:
    1
```

Result



# bravado-core: (2)Unmarshal

## ► Unmarshal (dict to object)

```
from bravado_core.unmarshal import unmarshal_schema_object
```

Code

```
book_obj = unmarshal_schema_object(
    spec, book,
    {"id": 1,
     "title": "Merchant of Venice",
     "author": "William Shakespeare"})
print(book_obj)
```

Dict need to be converted

```
Book(author='William Shakespeare', id=1, title='Merchant of Venice')
```

Result

Model object is created !!



# bravado-core: (2)Unmarshal

## ► Formatting in Unmarshal

```
Book:
  type: object
  required: [id]
  properties:
    id:
      type: integer
    title:
      type: string
    author:
      type: string
    release_date:
      type: string
      format: date
```

This property is added and  
expected to be string with "date" format

# bravado-core: (2)Unmarshal

## ► Formatting in Unmarshal

```
book_obj = unmarshal_schema_object(
    spec, book,
    {"id": 1,
     "title": "Merchant of Venice",
     "author": "William Shakespeare",
     "release_date": "2017-07-11"})
print(book_obj)
```

Code

Dict need to be converted

```
Book(author='William Shakespeare', id=1,
      release_date=datetime.date(2017, 7, 11), title='Merchant of Venice')
```

Result

String with date format is successfully converted to a date object!!

# bravado-core: (2)Unmarshal

## ► Defined formatter

- Default defined formatter:
  - byte, date, double, date-time, float, int32, int64
  - formatter.py ([https://github.com/Yelp/bravado-core/blob/master/bravado\\_core/formatter.py](https://github.com/Yelp/bravado-core/blob/master/bravado_core/formatter.py))
- Custom formatter by yourself
  - <https://github.com/Yelp/bravado-core/blob/master/docs/source/formats.rst>

# bravado-core: (3) Marshal

## ► Marshal (object to dict)

```
Book = spec.definitions['Book']
book_obj = Book(id=1, title="Merchant of Venice",
                author="William Shakespeare",
                release_date=date(2017, 7, 11))
book_dict = marshal_schema_object(spec, book, book_obj)
print(book_dict)
```

Code

“Book” object

```
{'release_date': '2017-07-11', 'title': 'Merchant of Venice', 'id': 1,
 'author': 'William Shakespeare'}
```

Result

Date object is successfully converted to string with date format!!

# bravado-core

## ► And many useful features

- Document

- <http://bravado-core.readthedocs.io/en/latest/>

- Examples

- <https://github.com/taxpon/bravado-core-example>

**Actual case study**

# Project overview

## ▶ Kabuku Connect

- Manufacturing cloud platform
- Connect people who want to make something and the factories selected by AI



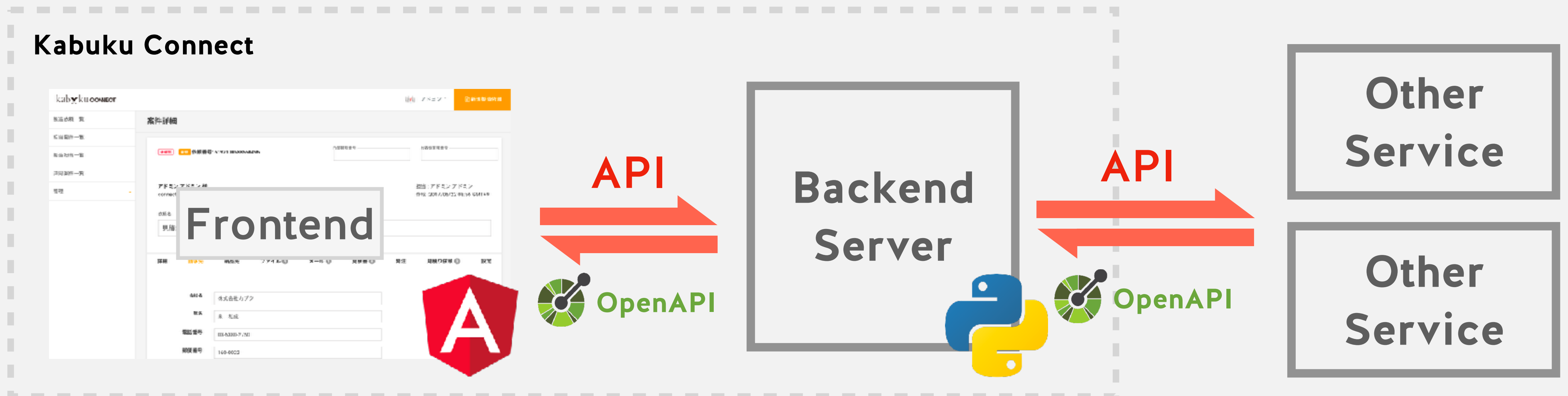
# Architecture

## ▶ Kabuku Connect

- Frontend: Angular (TypeScript)
- Backend: Python

## ▶ Other services

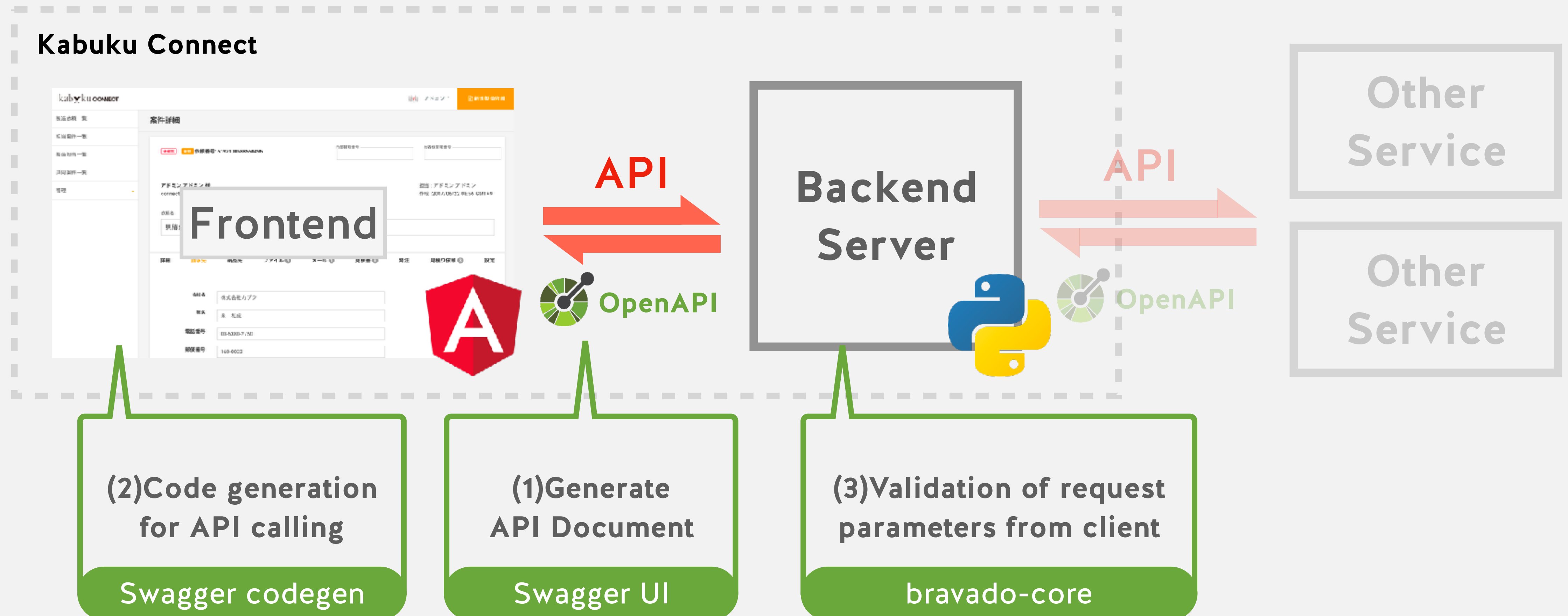
- Manufacturing management service
- Data analyzing service





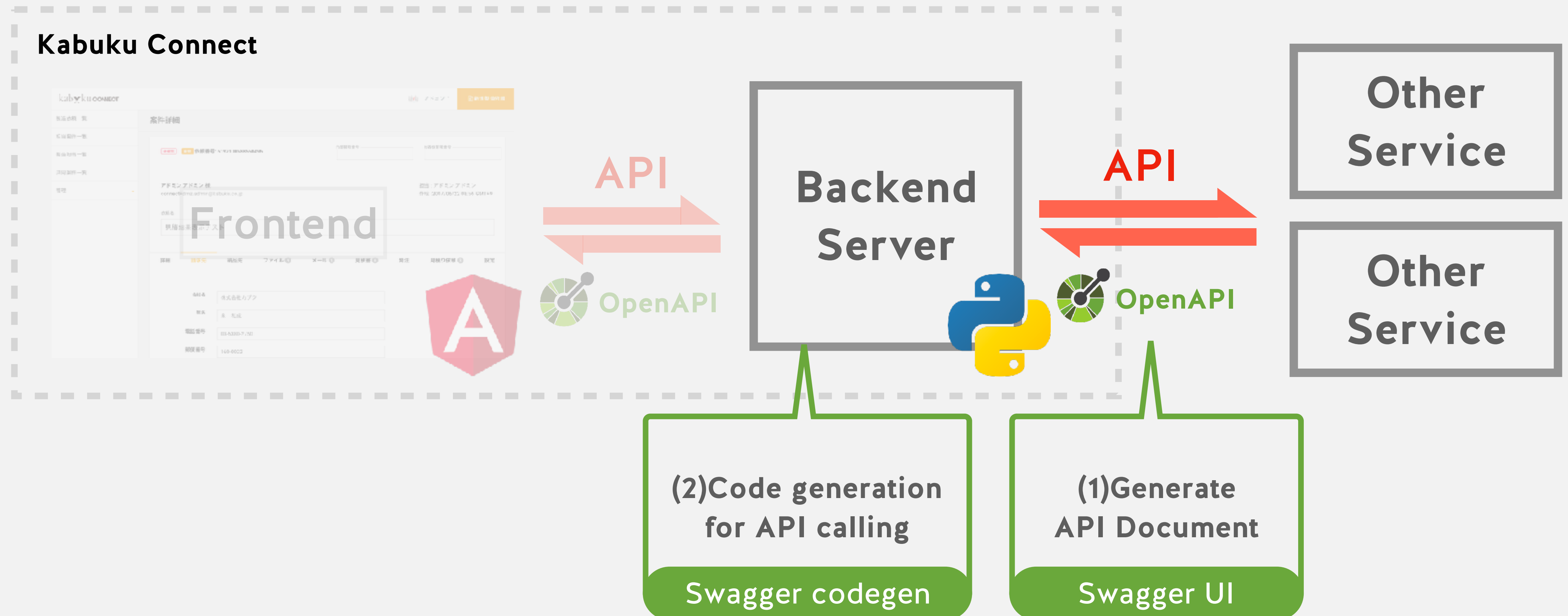
# How OpenAPI is used

## ► In Kabuku Connect



# How OpenAPI is used

## ► With other services



# Implementation workflow

## 1. Design

- ▶ API structure and write OpenAPI spec

## 2. Implementation

- ▶ Frontend (Angular, Typescript)
  - Using generated clients code and mock server
- ▶ Backend (Python)
- ▶ Frontend and Backend can be implemented in parallel

# Impression

► Using OpenAPI tools decrease your tasks so much

- Document generation
- Code generation
- Frontend and Backend, API provider and API consumer can be implemented in parallel

A yellow starburst graphic with a jagged, sun-like border, containing the text "Very Productive!".

Very Productive!

# Recap

# Recap

- ▶ OpenAPI is a hot technology to describe API specification
- ▶ There are many tools to increase your productivity with OpenAPI
- ▶ You learned actual case with OpenAPI

A yellow starburst graphic with a jagged, sun-like border. Inside the starburst, the text "So Hot!!" is written in a white, bold, sans-serif font.

So Hot!!

# Required more contributors!

- ▶ New Open API Spec (Version 3.0) will be released in July 2017
  - There are many added good features
  - Tools need to support OAS v3





# We are hiring!

- Python Developer
- C++ Developer
- Frontend Developer
  - Angular/Three.js
- You can use 3D printers all you want
- International members
- 3 Google Developer Experts



<https://www.kabuku.co.jp/en>



Some 3D printed members