



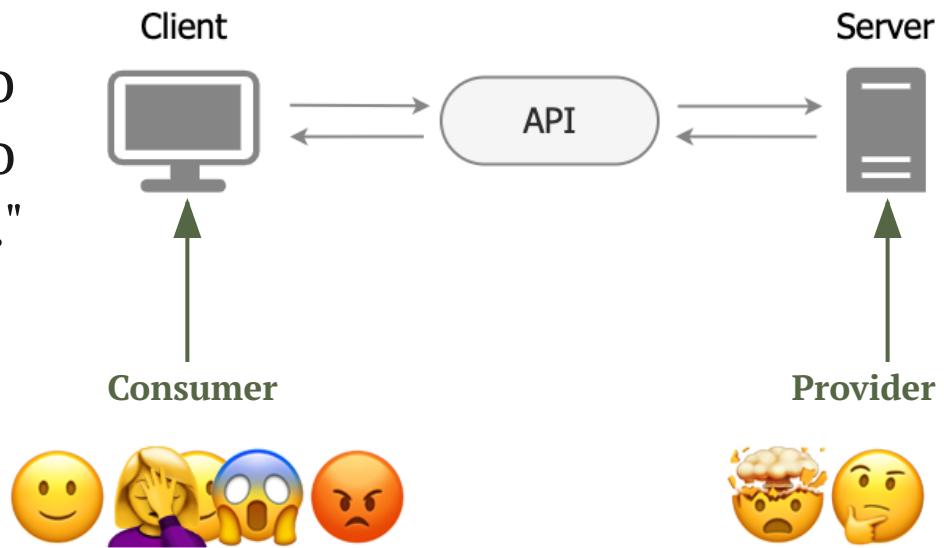
Communicating Our APIs: Enhance Provider and Consumer Interaction

Enis Spahi

APIs

"An application programming interface (API) is a way for two or more computer programs to communicate with each other."

Source: Wikipedia, "API"



About me

Consultant Architect at OpenValue

✉ enis@openvalue.de

⌚ enisspahi

㏌ enisspahi



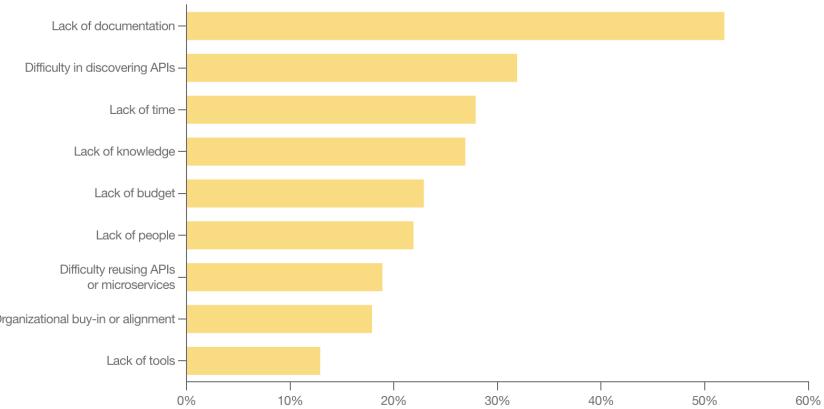
Boundaries

- Private APIs
 - Provider and consumer are developers in the same team or same organisation
- Partner Facing APIs
 - Serving partners (i.e.: Payment Service Providers)
 - Provider and consumer might not communicate directly
- Public APIs
 - Publicly available (i.e... Geo-Location services)
 - Communication at scale: Many Consumers ↔ 1 Provider

Statistics

Top 3 Obstacles to consuming APIs

- Lack of documentation
- Difficulty in discovering APIs
- Lack of time



Source: Postman, "2023 State of the API Report"

Enhancing API Discoverability

Her: I bet he's
thinking about other women

Him: I don't understand this API



Find a common language



OPENVALUE

OpenAPI Specification

- Technology agnostic standard to describe Rest APIs
- Formerly Swagger, OpenAPI as of version 3
- Written as JSON or YAML
- Great tooling for code and documentation generation
- <https://openapi.tools/>

OpenAPI Specification

```
openapi: 3.0.3
info:
  title: Recipes API
  ...
servers:
- url: http://localhost:8080
paths:
  /recipes:
    get:
      summary: List all recipes
      ...
      responses:
        ...
        "200":
          description: OK
          content:
            'application/json':
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Recipe'
```

- **Openapi:** Spec Version
- **Info:** General API information as metadata
- **Servers:** Connectivity information about target servers
- **Paths:** Paths to the endpoints with their expected request, response and errors.
- **Components:** Holds the schemas for the request, response and errors for referencing

AsyncAPI Specification

- Technology agnostic standard to describe message-driven APIs
- An adaptation of the OpenAPI specification
- Written as JSON or YAML
- Protocols: AMQP, HTTP, JMS, Kafka, but not only
- <https://www.asyncapi.com/tools>

AsyncAPI Specification

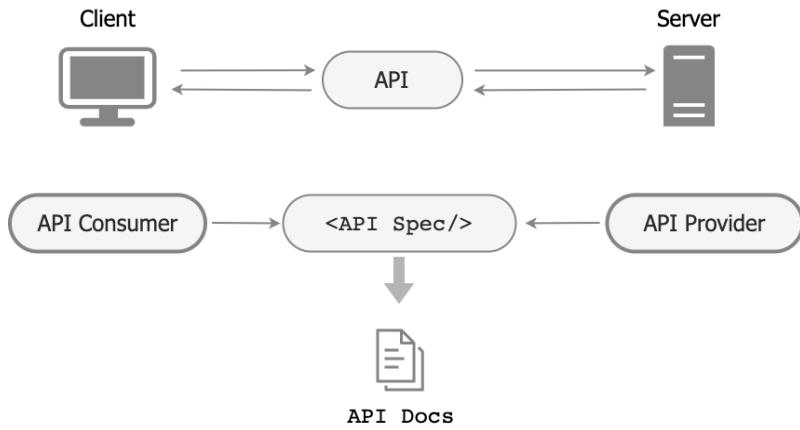
```
asyncapi: 2.0.0
info:
  title: Ping Service
  ...
servers:
  localhost:
    url: localhost:9092
    protocol: kafka
    protocolVersion: '1.0.0'

channels:
  ping.topic:
    description: Kafka topic for ping messages
    publish:
      operationId: pingSent
      message:
        $ref : '#/components/messages/Ping'
  pong.topic:
    description: Kafka topic for pong messages
    subscribe:
```

- **Asyncapi:** Spec Version
- **Info:** Metadata information about the API
- **Servers:** Connectivity information about servers (i.e. Kafka brokers)
- **Channels:** Messages exchange between provider and consumer
- **Components:** Defines the reusable objects such as schemas or messages which could be referenced.

generated by AsyncAPI Generator

Standardized API Communication



- Common Language for API discovery
 - OpenAPI/Swagger for REST APIs
 - AsyncAPI for message-driven APIs
- Foundation for tooling
 - Code generation
 - Documentation
- Community

Enhancing API development



Demo Time

Let's build a Recipes API

Client

What should I eat?

Title:

Ingredients:

Nutrition Facts:

- High Protein
- Low Calorie
- Carbs
- High Calorie

Recipes

1. Pumpkin Soup

Ingredients

Pumpkin - 1000.0 grams

Server

```
curl 'http://localhost:8080/recipes?title=Pumpkin&nutrition=Low%20Calorie'
```

```
[  
  {  
    "title": "Pumpkin Soup",  
    "ingredients": [  
      {  
        "name": "Pumpkin",  
        "quantity": 1000.0,  
        "unit": "grams"  
      },  
      {  
        "name": "Onion",  
        "quantity": 1.0,  
        "unit": "unit"  
      },  
      {  
        "name": "Vegetable broth",  
        "quantity": 100.0,  
        "unit": "milliliters"  
      }  
    ]  
  }]
```

API Development - Code First

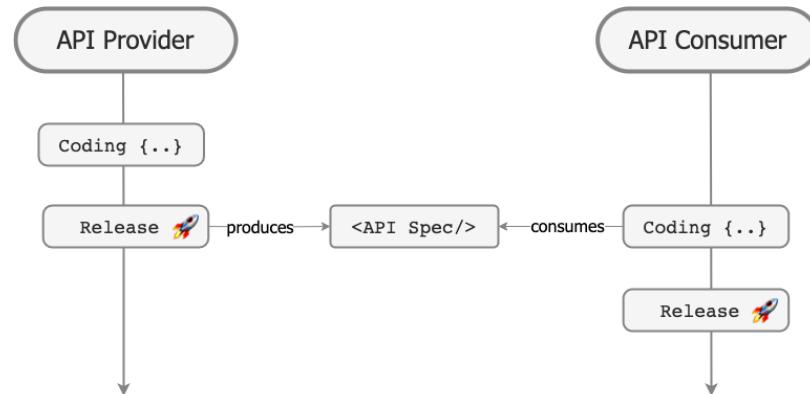
Communicate API specification once coding has been done

- **Advantages:**

- Focus on coding
- Flexibility to change the API design

- **Disadvantages:**

- Late communication with the consumer
- Does not enable development in parallel
- Annotations



API Development - API First

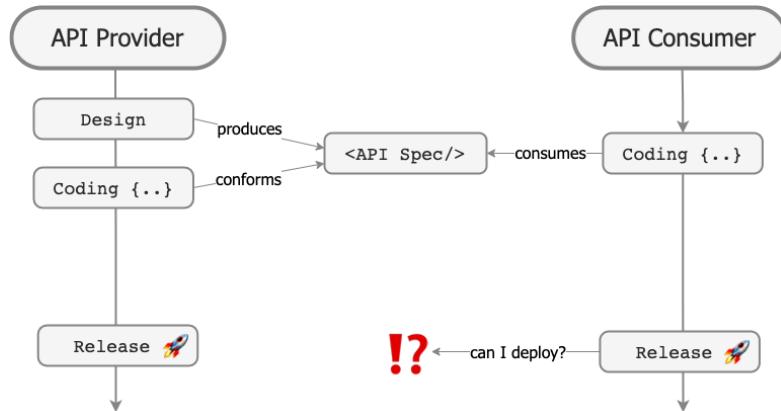
Communicate API specification before coding. Prioritizes API design over implementation.

- **Advantages:**

- Early communication with the consumer
- Documentation thought ahead
- Enables development in parallel

- **Disadvantages:**

- Less flexibility to change the API design
- Sometimes bureaucratic for providers



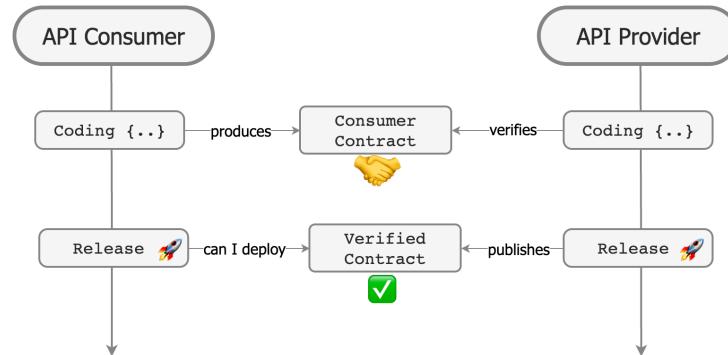
API Development - Consumer First

Consumer dictates the expected API behavior to the provider

Pact: A Code-first consumer contract testing tool that enables consumer driven API development.

Process:

- Consumer produces a pact
- Provider verifies it's API implementation
- Server / Client deployments synced



The right methodology for me?

When to use Code first?

Provider initially focuses on coding speed
Flexibility to change the design

When to use API first?

API design over implementation
Early communication with the consumer → Documentation
Utilize code generation
Large number of consumers

When to use Consumer first?

Provider should conform to consumer needs
API consumer and provider test their applications independently
To sync provider and consumer deployments
Small number of consumers

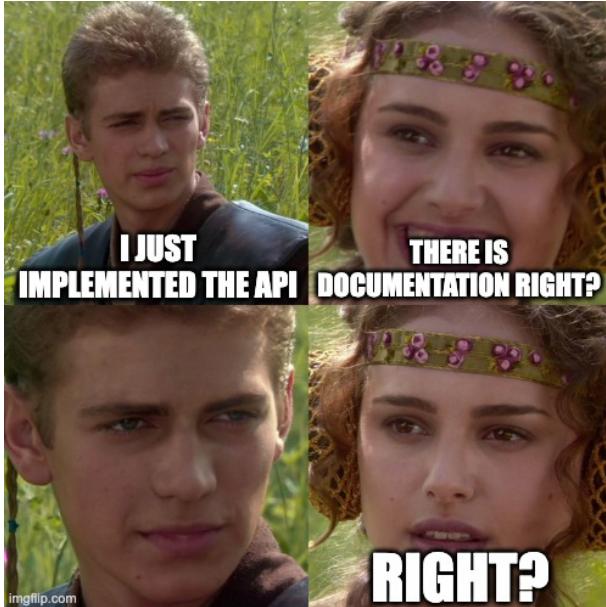
When to mix & match?

When API first alone is not sufficient to match consumer needs

Are same approaches applicable to AsyncAPI?

- Inspired by OpenAPI
- Code First and API First
- Pact Messaging support
- <https://www.asyncapi.com/tools>

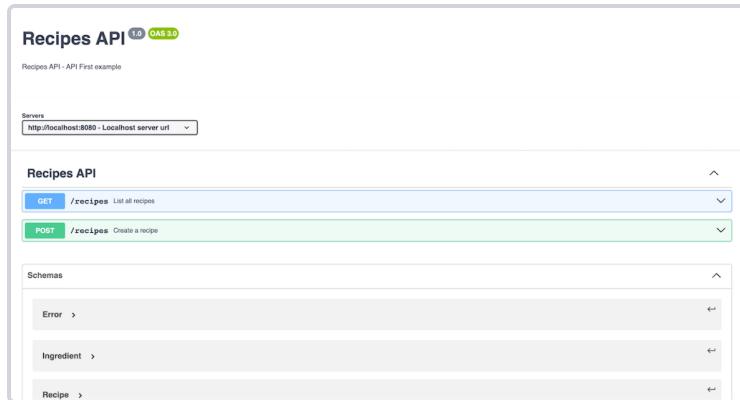
Enhancing API Documentation



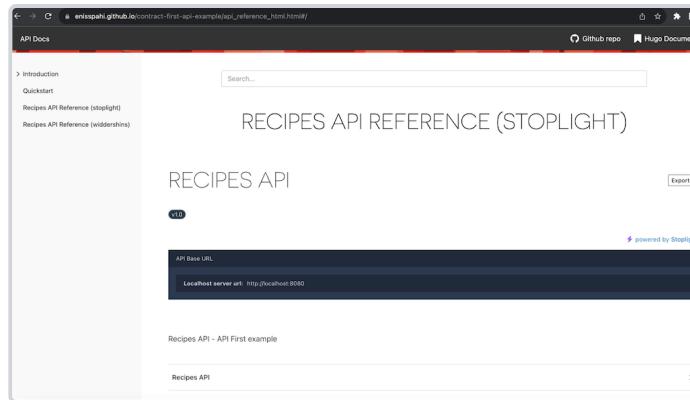
API Documentation

API Specifications can be leveraged to generate more human readable forms of documentation

API Reference



API Documentation



- APIs described as web pages
- Code samples, try-it-out
- Auto generated
- **Swagger-ui:** The most popular

- API Reference incorporated in a bigger ecosystem
- Conceptual technical documentation
- Docs as code: Markdown, Technical Writing
- Continuous documentation
- Demo

Recap

Enhancing API Discoverability

Speak common language

- OpenAPI, AsyncAPI, ...

Enhancing API development

Pick the right methodology

- Code first, API first, Consumer first, mix & match
- Speed up development
- OpenAPI Generator, Pact, Swagger-validator

Enhancing API Documentation

API Specification → API Reference → API Docs

Stay up-to-date with Continuous Documentation

Thank you for your attention!



Feel free to scan