

**API Design First  
and Always**

# HELLO!

I'm Phil Sturgeon

I love to talk about APIs,

fixing bikes, and

saving the planet.

You can find me at

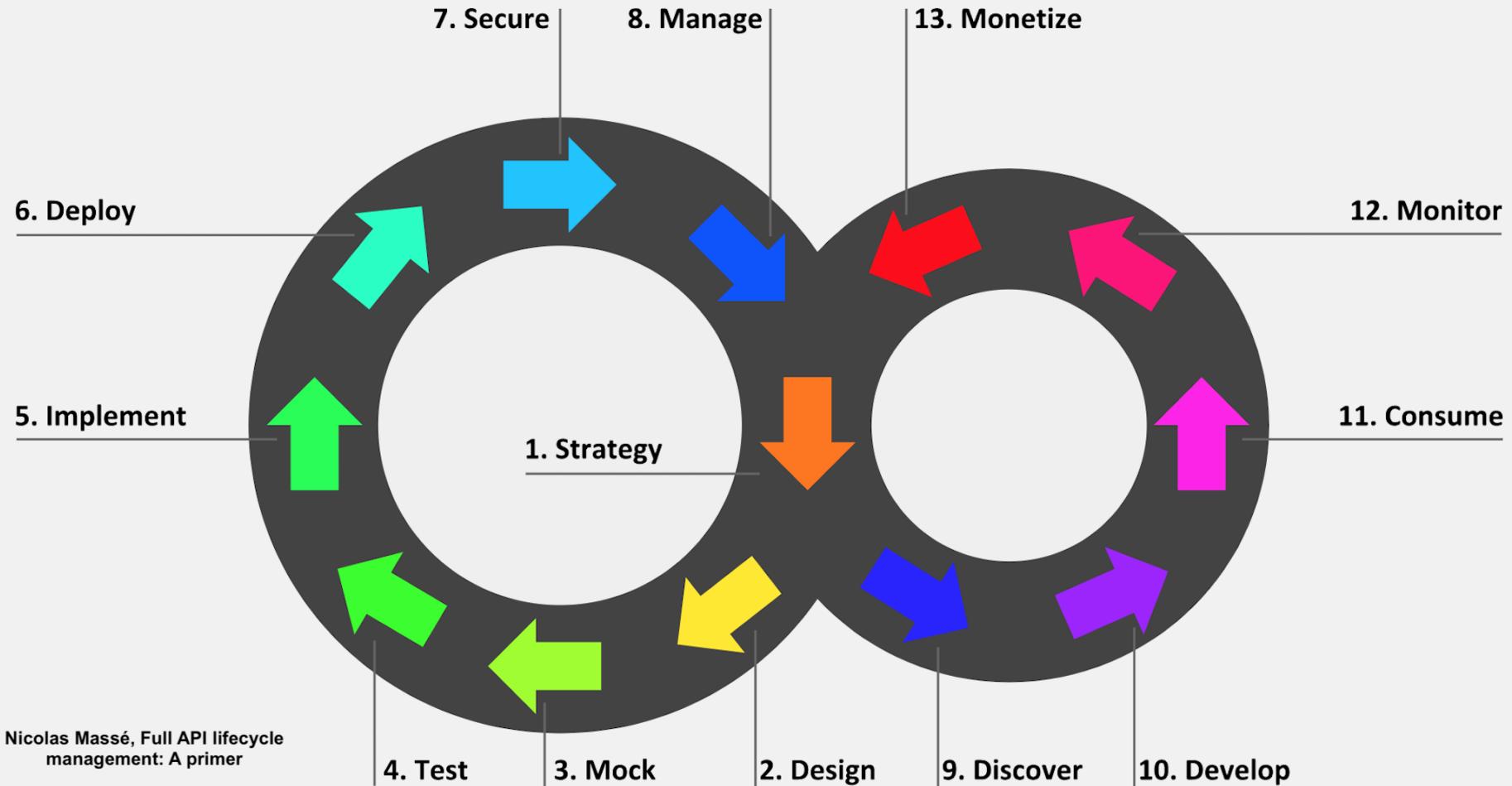
[philsturgeon](https://philsturgeon.ca)



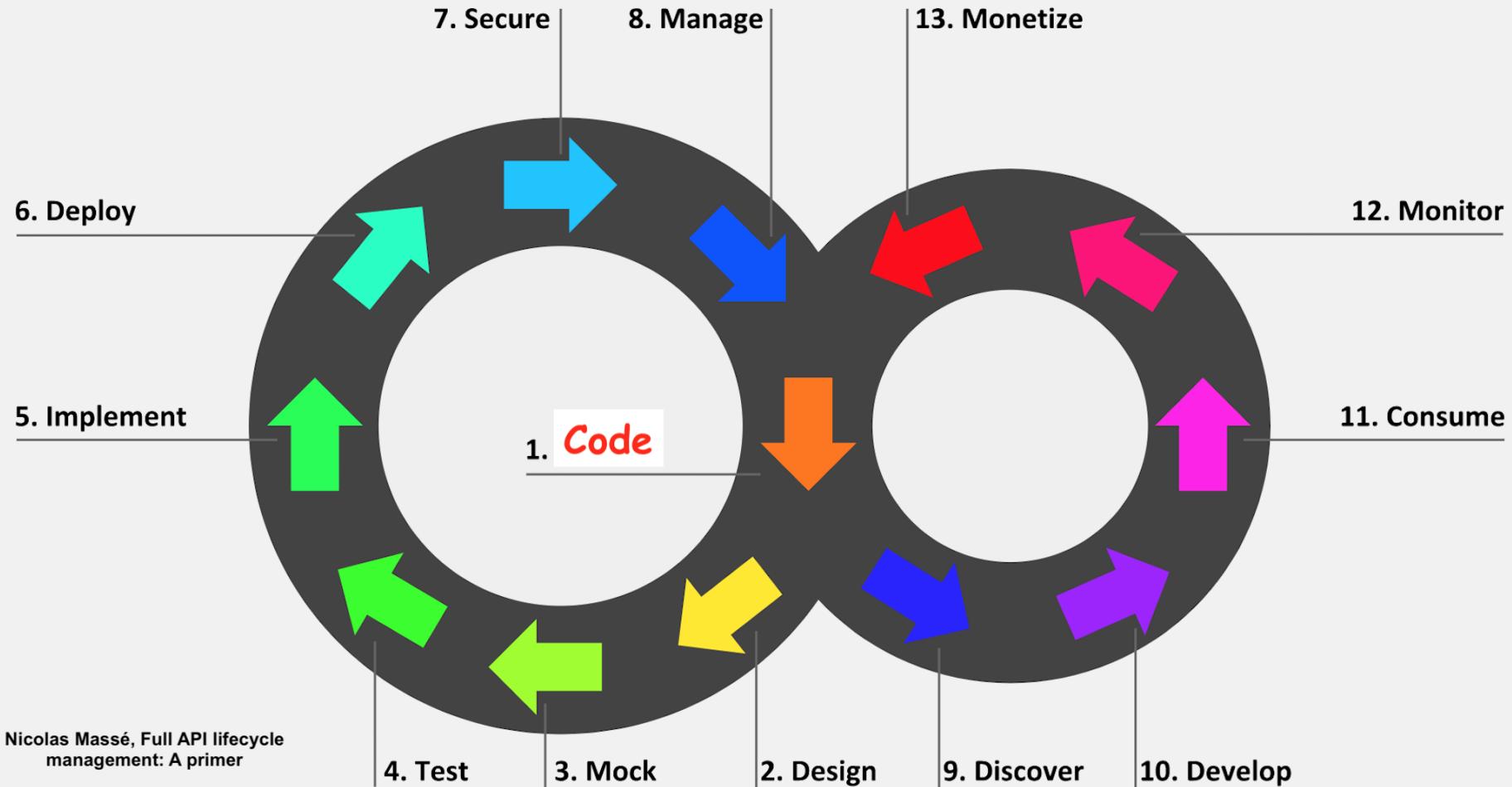


**wework**

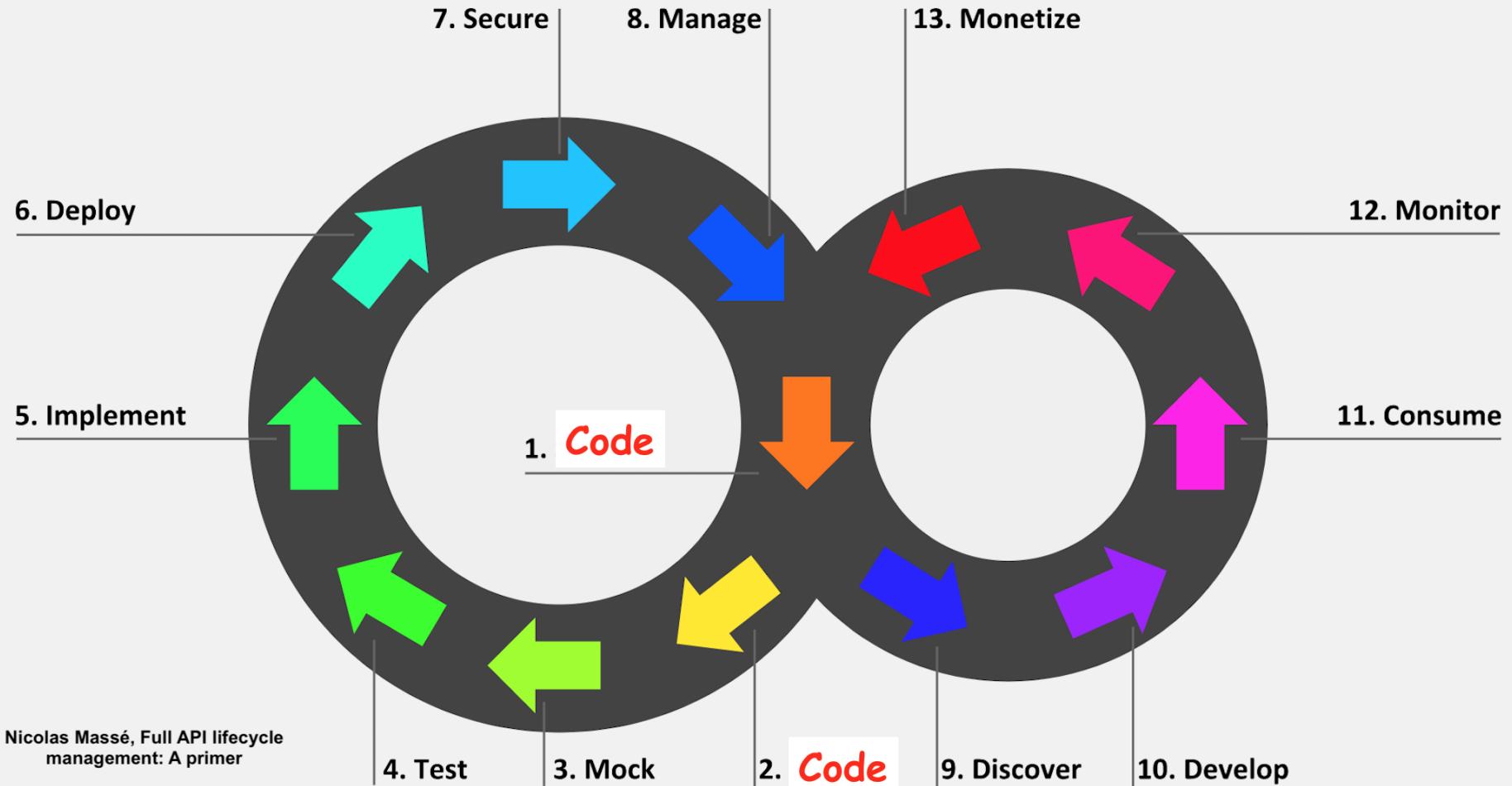


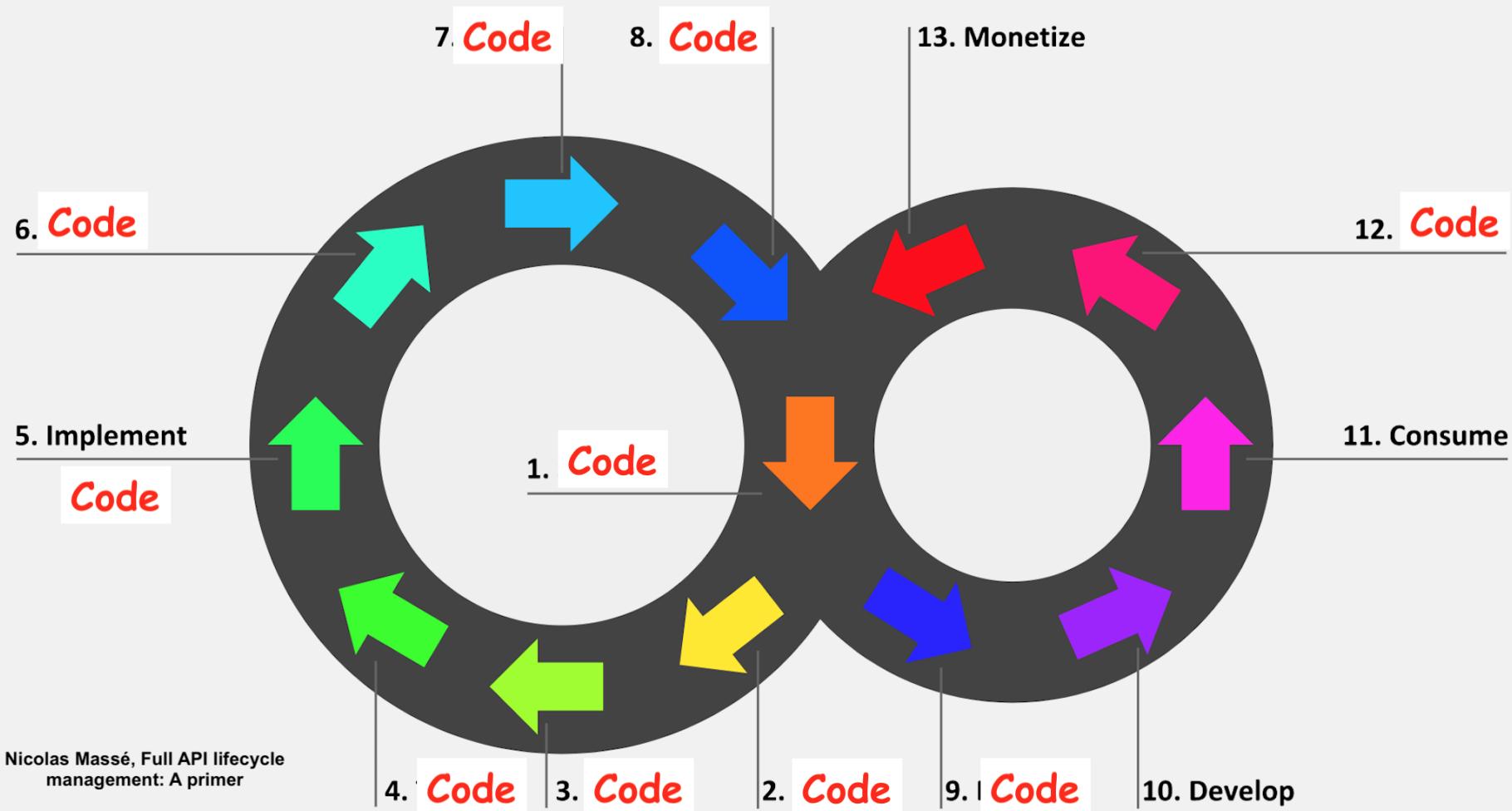


Nicolas Massé, Full API lifecycle  
management: A primer



Nicolas Massé, Full API lifecycle management: A primer





# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?
3. Design First or Code First?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?
3. Design First or Code First?
4. Where are the visual editors?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?
3. Design First or Code First?
4. Where are the visual editors?
5. How/when do we create documentation?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?
3. Design First or Code First?
4. Where are the visual editors?
5. How/when do we create documentation?
6. How/when do we create mocks?

# API WORKFLOW QUESTIONS

1. Swagger, OpenAPI v3, API Blueprint, RAML?
2. Annotations or YAML?
3. Design First or Code First?
4. Where are the visual editors?
5. How/when do we create documentation?
6. How/when do we create mocks?
7. How do we keep code and docs in sync?

# THE API DESCRIPTION FORMAT



OpenAPI is an API Description Format for creating  
HTTP API Descriptions.

You can describe your API with a Description Document.

e.g.: openapi.yaml

```
openapi: 3.0.3

info:
  version: v1.0
  title: RemoteOk Jobs API
  contact:
    email: support@remoteok.io
  description: >
    Remotely interested in an a job? This is a JSON feed from Rem
    all the latest jobs. You can't search by category or tag as f
    tell, but... you can sift through!

servers:
  - url: 'https://remoteok.io/api'
    description: Production
  - url: 'https://sandbox.remoteok.io/api'
```

```
type: object
properties:
  name:
    title: Name
    type: string
    description: Users full name supporting unicode but no emojis
    maxLength: 20
  email:
    title: Email
    description: Like a postal address but for computers.
    type: string
    format: email
  date_of_birth:
    title: Date Of Birth
    type: string
    description: 'Date of users birth in the one and only date st
```



Assertible



ORACLE + apairy



# ANNOTATIONS OR YAML?

```
class UserController {
    @OpenApi(
        path = "/users",
        method = HttpMethod.POST,
        // ...
    )
    public static void createUser(Context ctx) {
        // ...
    }
}
```

```
/**
 * @OA\Get(path="/2.0/users/{username}",
 *     operationId="getUserByName",
 *     @OA\Parameter(name="username",
 *         in="path",
 *         required=true,
 *         description=Explaining all about the username parameter
 *         @OA\Schema(type="string")
 *     ),
 *     @OA\Response(response="200",
 *         description="The User",
 *         @OA\JsonContent(ref="#/components/schemas/user"),
 *         @OA\Link(link="userRepositories", ref="#/components/links
 *     )
 * )
 */
```

```
/**
 *  @swagger
 *  /users:
 *    get:
 *      description: Returns users
 *      produces:
 *        - application/json
 *      responses:
 *        200:
 *          description: users
 *          schema:
 *            type: array
 *            items:
 *              $ref: '#/definitions/User'
 */
app.get('/users', (req, res) => {
```

*Code comments are just facts waiting to  
become lies.*

*Vinai Kopp, A talk at #MM18IT*

# MILK



38

88c

# CODE FIRST VS DESIGN FIRST



Crashy McCiderface 🚲🌐🔥  
@philsturgeon

API Devs: Would you summarize the API development culture at your organization as:

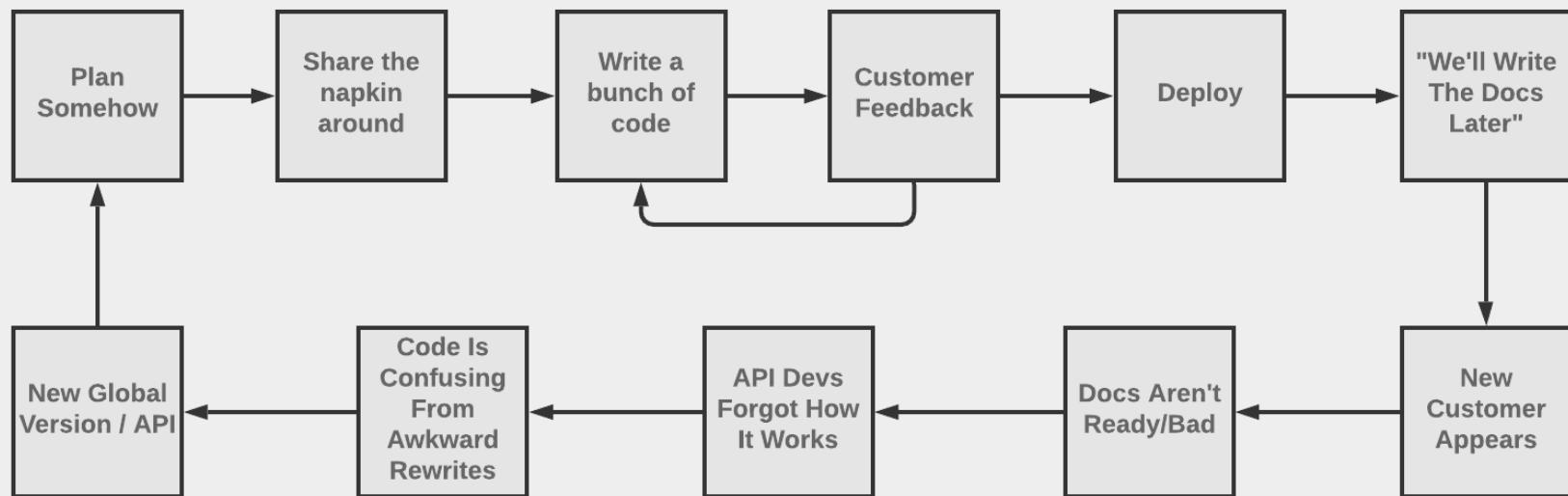
- A) entirely Design First
- B) entirely Code First
- C) used to be Code First, adopting Design First for new APIs
- D) Some other awkward combination

A) Design-First	20.9%
B) Code First	23.3%
C) Adopting Design First	20.3%
<b>D) Awkward Combination</b>	<b>35.5%</b>

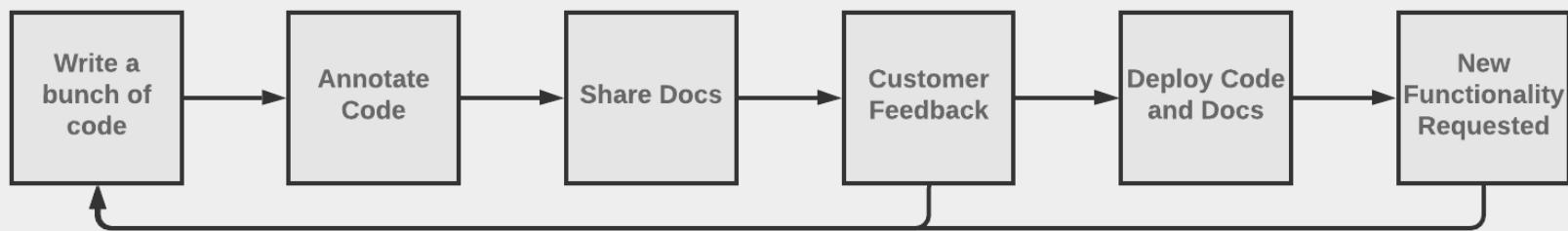
296 votes · Final results

1:28 PM · Mar 2, 2020 · [Twitter Web App](#)

## Code First, Docs "When We Have Time"

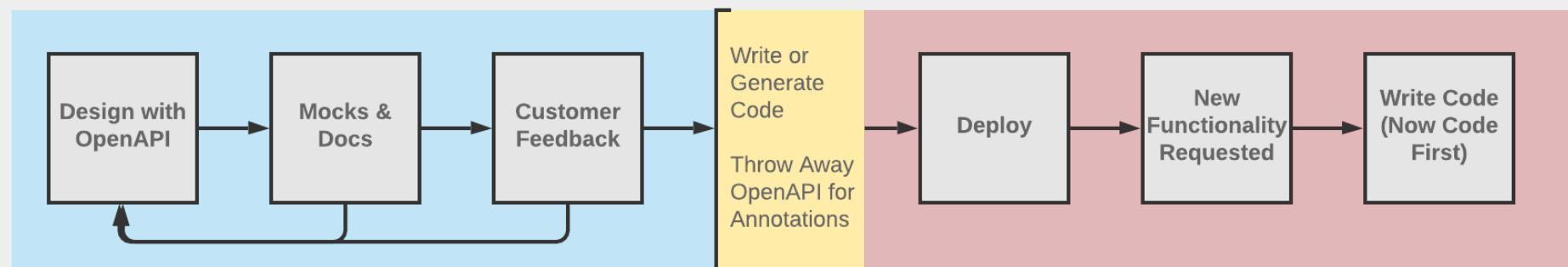


## Code First, Generate Docs

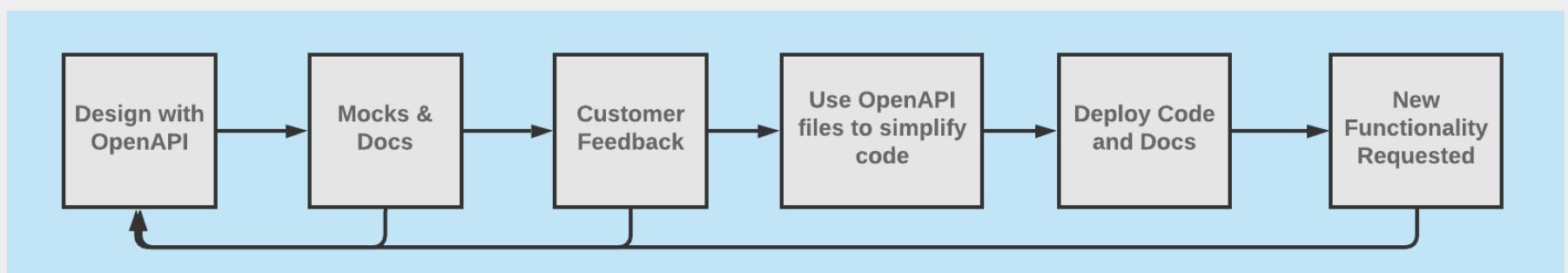


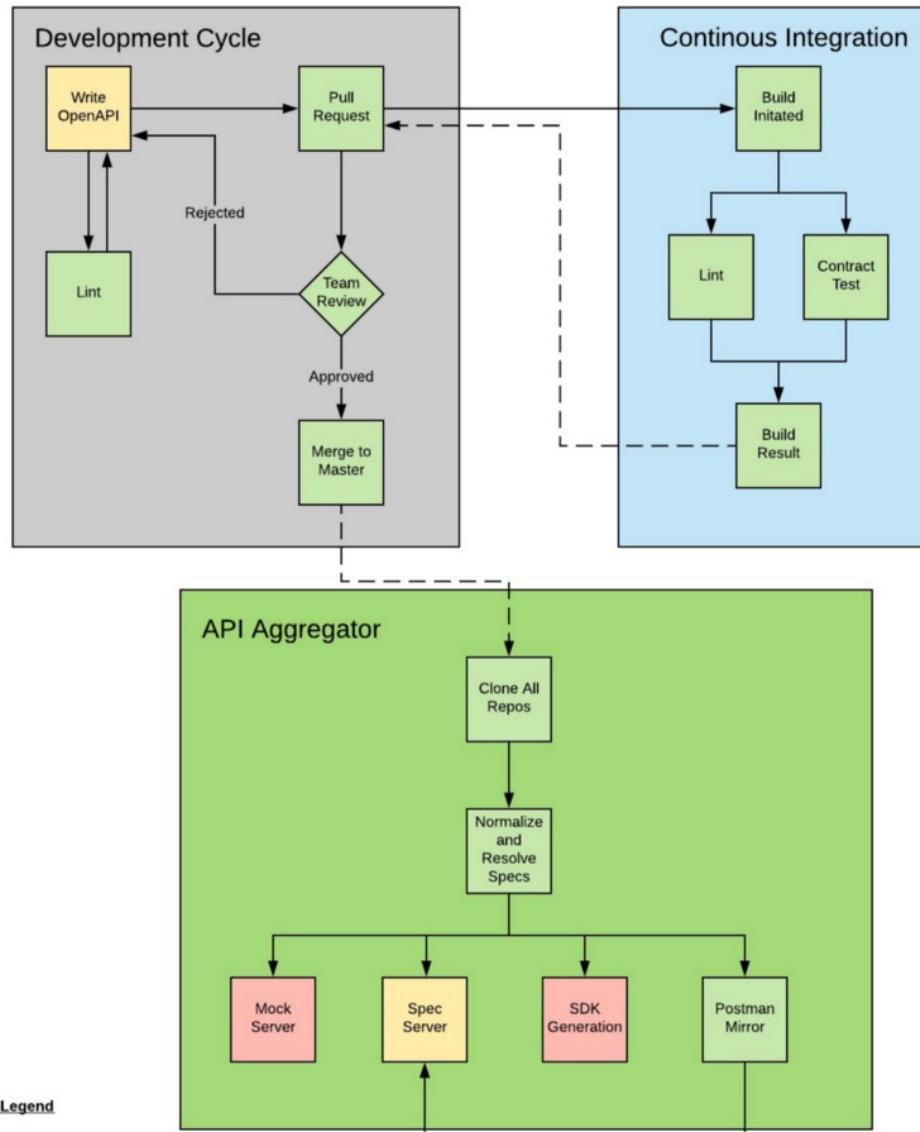


## Design First, Ditch For Code First



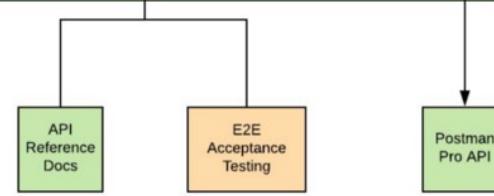
## Design First, Evolve With Code

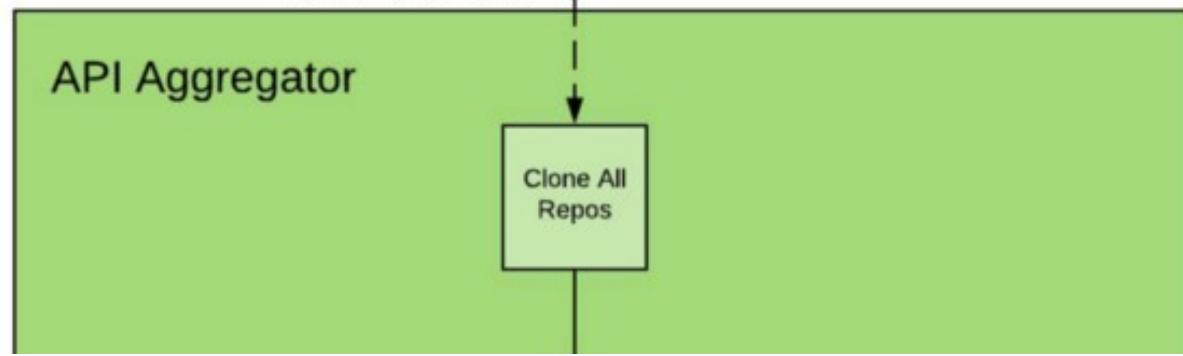
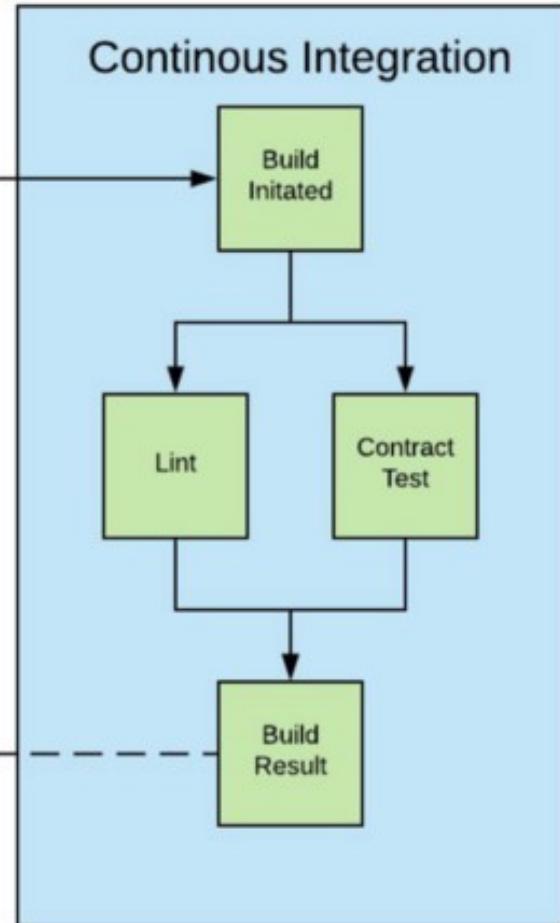
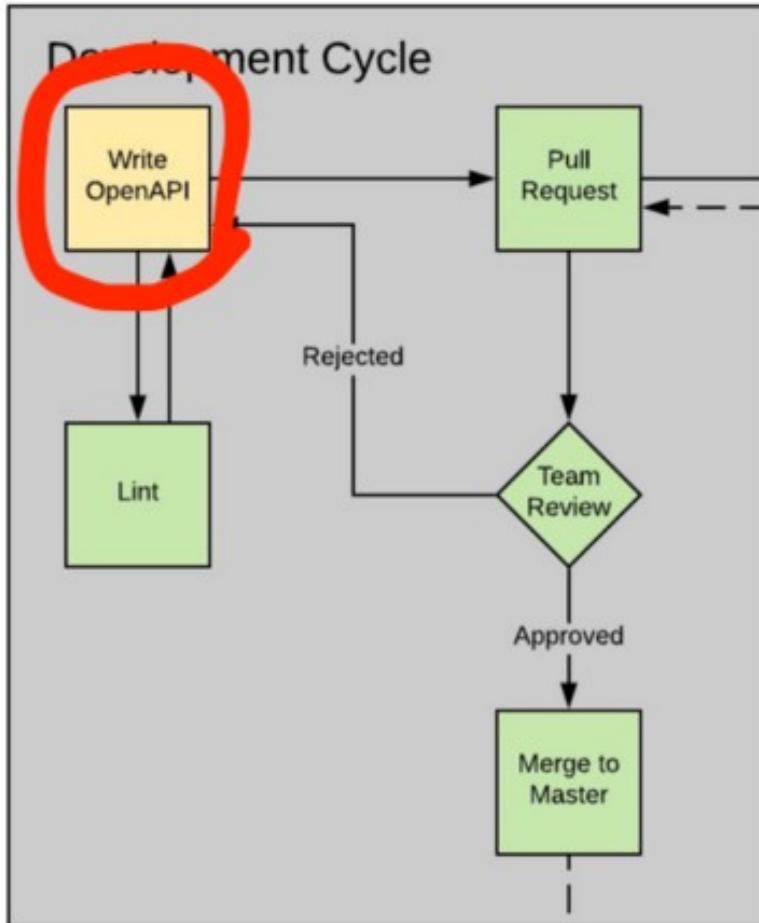




#### Legend

- [Green Box] Done
- [Yellow Box] Needs Improvement
- [Orange Box] In Progress
- [Red Box] TODO





*While OpenAPI is great for describing APIs, ... it's definitely not a great experience to write OpenAPI documents from scratch.*

*Sebastien Armand, "Making OpenAPI Bearable With Your Own DSL"*

```
;;; ENTITIES
(define pet-entity
  (entity "Pet"
    'race (string "What kind of dog / cat this is (labrador, gold
'origin (string "Country of origin" "Egypt")
'birthday (datetime "Birth date of the pet" "2017-10-20T00:14
'species (string "What kind of animal is this" "dog" #:enum '
(define $pet (schema-reference 'Pet pet-entity))

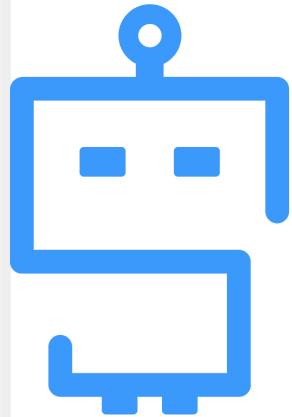
;;; RESPONSES
(define list-pets-response (jsonapi-paginated-response "List of p

;;; REQUESTS
(define pet-request (json-request "Pet Request Body" ($pet)))

... MATH DOC
```

Where are the visual editors?

<https://openapi.tools/#gui-editors>

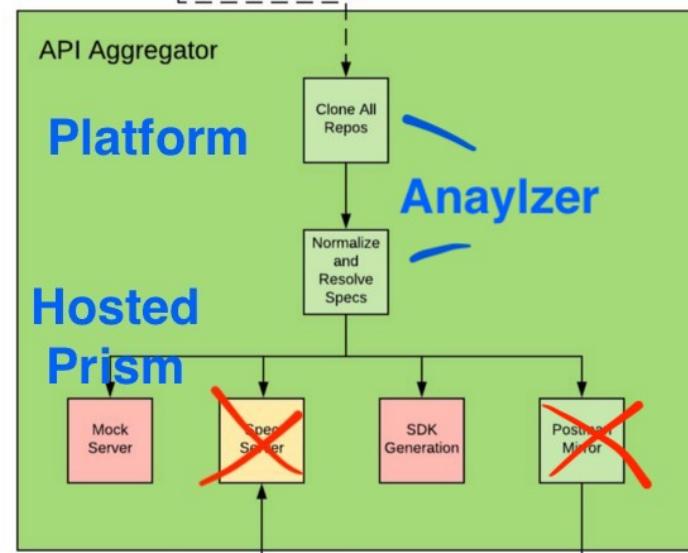
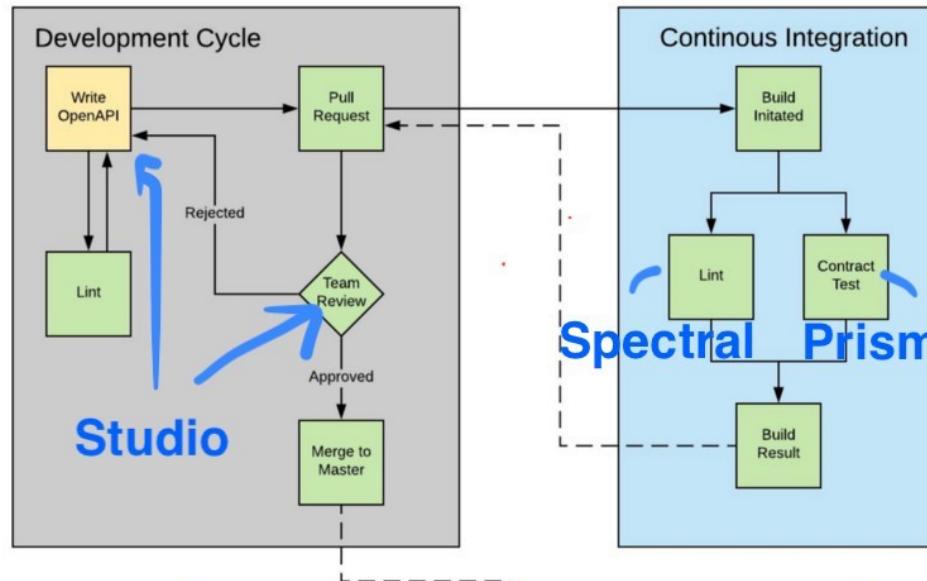


# Stoplight

# Build Quality APIs Faster

Supercharge your API lifecycle with best-in-class tooling that integrates seamlessly into your Git workflows.

[Get Started For Free](#)



**Legend**

- [Green Box] Done
- [Yellow Box] Needs Improvement
- [Orange Box] In Progress
- [Red Box] TODO

**Docs**

**Prism**

API Reference Docs

E2E Acceptance Testing

Postman Pro API

- Add**
- API
- Endpoint
- Model
- Article
- Style Guide
- Stoplight Config
- Image
- File
- Folder
- Import File

pets

## Create a pet

http://petstore.swagger.io/v1 /pets Path Params

Path Param +

OPERATION ID  
createPets

DESCRIPTION  
Endpoint description...

+ Security + Header + Query Param

Query Params  
Name...

+ Add Body

+ Response 201 default

Null response

Headers +

**PARAMETER PROPERTIES**

style none deprecated

**SCHEMA PROPERTIES**

format none default

enum type an option and press enter

example example

**STRING PROPERTIES**

pattern ^[A-Za-z0-9 -]+

minLength >= 0 maxLength >= 0

## **Collaborate Flexibly with Git Workflows**



Collaborate on API designs easily using your existing Git repositories.  
Stoplight fits seamlessly into your current development workflows.

[APIs](#)[Docs](#)[Files](#) [reference](#) [checkout](#) [openapi.yaml](#) [payment](#) [openapi.yaml](#) [recurring](#) [openapi.yaml](#) [specs](#) [.DS\\_Store](#) [.spectral.yml](#) [LICENSE](#) [README.md](#)

## Returns ava

No server URL de

[GET](#)[POST](#)

Queries the ava  
transaction bas  
(like amount, c  
giving back a l  
methods, the re  
input details y  
shopper (to be

Although we hig  
endpoint to ens  
the most up-to-  
its usage is op

+ Response

● 200

● default

200



161

162

NewPet:

pet

TYPE

allOf    oneOf    anyOf  
object    array    string    number    integer    boolean    null    \$ref

Head

SUBTYPE

none    object    allOf    oneOf    anyOf    string    number    integer    boolean    null    \$ref



\$REF TARGET

This File



#/components/schemas/Pet

Sch

array[\$ref( #/components/schemas/Pet )] (go to ref)



175

176

177

178

type

ui

re

name

prop

er

ame:

type

tag:

type

of:

\$ref

req

u-

prop

ic



## FAQs

Add a section for FAQs and Support.

### Can i cancel my Payment?

lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi tellus. Phasellus iaculis sed sapien sed interdum. Pellentesque nunc, vitae laoreet ante.

### How do i recieve money through Acme Payments?

lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi tellus. Phasellus iaculis sed sapien sed interdum. Pellentesque nunc, vitae laoreet ante.



### We're here to Help

Get in touch and let us know how we can help.

Community Support

Acme Payments Technical Support

Phone Support

Ask a peer in the community how they integrate Acme Payments.

[Go to Community](#)

If you're a registered user, please login [here](#).

```
40 |           description: OK – the request has succeeded.  
41 |           schema:  
42 |             $ref: '#/definitions/PaymentMethodsResponse'
```

Type	Line	Message	powered by Spectral
⚠	84	Operation should have non-empty `tags` array.	 
⚠	86	Operation summary should be succinct, no full stops, and less than ...	 
⚠	86	Operation summary should be succinct, no full stops, and less than ...	 
⚠	96	Parameters must have a description.	 
⚠	96	Parameters must have a description.	 
⚠	96	Parameters must have a description.	 
⚠	96	Parameters must have a description.	 
⚠	96	Parameters must have a description.	 

Add

- API
- Endpoint
- Model
- Article
- Style Guide**
- Stoplight Config
- Image
- File
- Folder
- Import File

## Custom rules



**info-matches-stoplight**

This is an example rule

[oas3, oas2]



## Inherited rules



**contact-properties**

Contact object should have `name`, `url` and `email`.

[oas2, oas3]



**info-contact**

Info object should contain `contact` object.

[oas2, oas3]



**info-description**

OpenAPI object info `descri`

Form </> Code

Preview Mocks

17 Problems



**info-license**

OpenAPI object `info` shou

Type	Line	Message	powered by Spectral 5.5.0-beta1	Filter results...
⚠	1	OpenAPI object should have non-empty 'tags' array.		
⚠	2	Info object should contain 'contact' object.		
⚠	2	OpenAPI object info 'description' must be present and non-empty string.		
⚠	4	Info must contain Stoplight		
⚠	11	Operation 'description' must be present and non-empty string.		
⚠	15	Operation tags should be defined in global tags.		
⚠	74	Operation 'description' must be present and non-empty string.		
⚠	74	Operation parameters are unique and non-repeating.		



**no-eval-in-markdown**

Markdown descriptions shou



**no-script-tags-in-markdown**

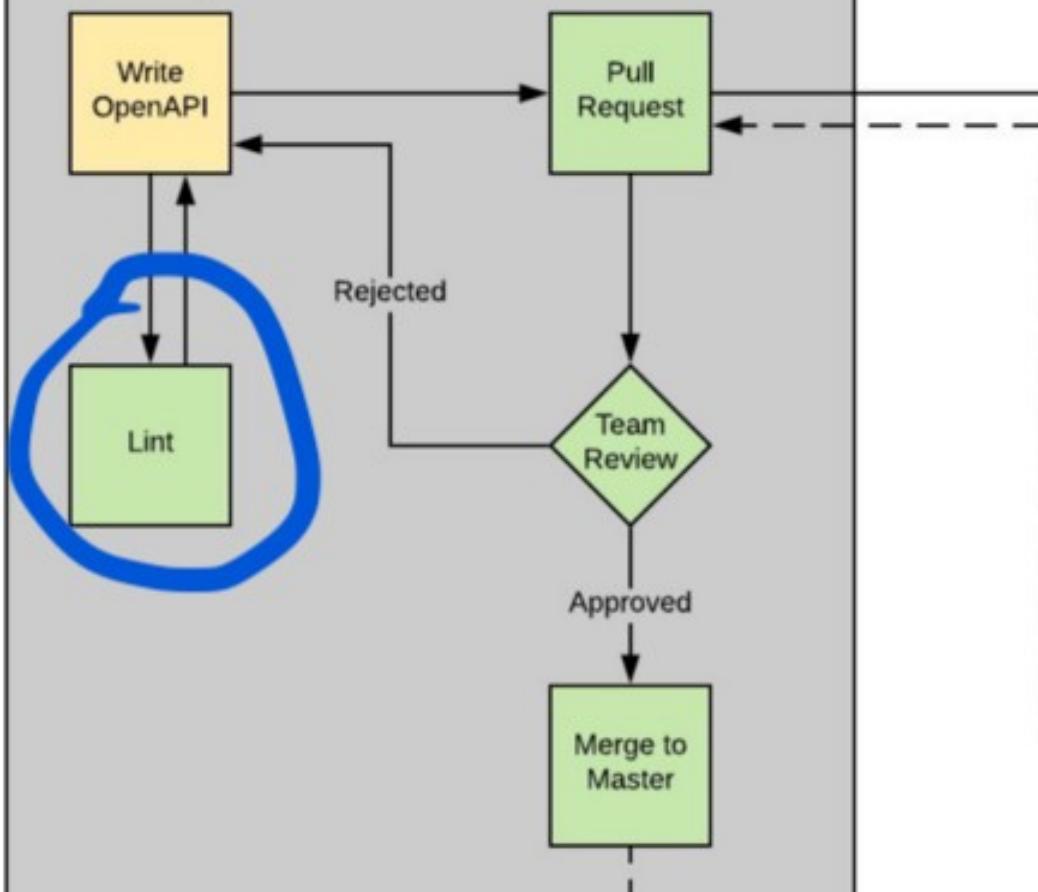
Markdown descriptions should not contain `<script>` tags.

[oas2, oas3]

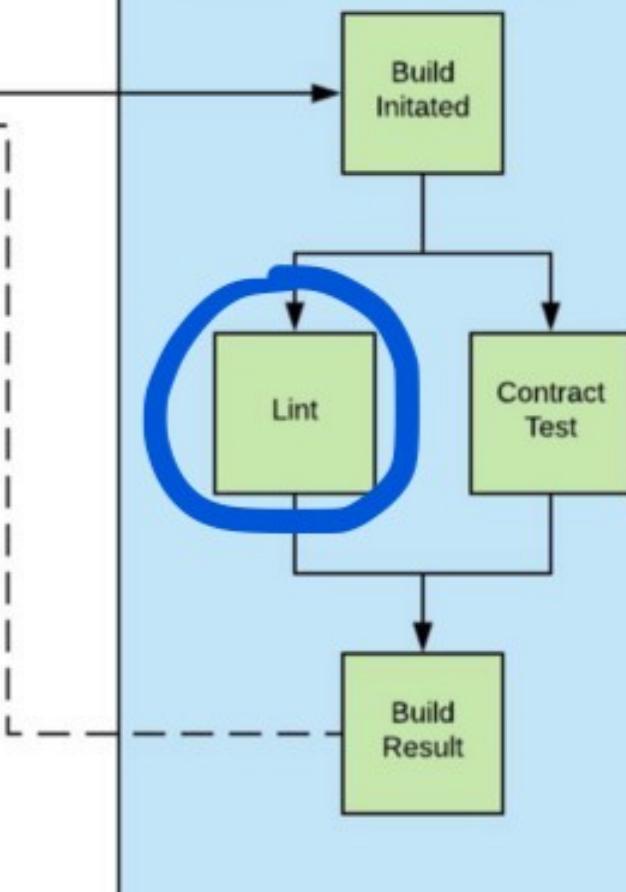
# API DESCRIPTION LINTING

a.k.a "API Style Guides"

## Development Cycle



## Continuous Integration



## Spectral CLI

```
▶ spectral lint -r apisyouwonthate.yml example.yaml  
OpenAPI 3.x detected
```

```
/Users/phillip/src/openapi-contrib/style-guides/example.yaml  
6:22 warning paths-kebab-case /why_underscores should be kebab-case (lower  
and separated with hyphens)
```

```
* 1 problem (0 errors, 1 warning, 0 infos, 0 hints)
```



```
5  paths:  
6    "/why_underscores":  
7      get:  
8        responses:  
9          why_underscores:  
10         /why_underscores MUST be kebab-case (lower case and  
11           separated with hyphens) spectral(paths-kebab-case)  
12         Peek Problem (⌞F8) No quick fixes available  
13         | type: string  
14         example: hello!  
15
```

## VS Code Spectral

```
5 paths:
6   "/hello":
7     get:
8       responses:
9         "400":
10        content:
11          nonsense/json: ~
12          schema: ~
13          type: string
14          example: hello!
15
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1 Filter: E.g.: text, \*\*/\*.ts, !\*\*/node\_modules/\*\* ⌂ ⌄ ⌈ ×

! example.yaml 1

⚠ Every error response SHOULD support either RFC 7807 or the JSON:API Error format. spectral(unknown-error-format) [11, 27]

## How/when do we create mocks?

1. Local Mock Servers
2. Hosted Mock Servers

<http://openapi.tools/#mock>

# Local Mock Server

```
→ Stoplight prism mock petstore.yml
[CLI] ... awaiting Starting Prism...
[HTTP SERVER] i info      Server listening at http://127.0.0.1:4010
[CLI] • note      GET      http://127.0.0.1:4010/pets
[CLI] • note      POST     http://127.0.0.1:4010/pets
[CLI] • note      GET      http://127.0.0.1:4010/pets/{petId}
[CLI] ▶ start    Prism is listening on http://127.0.0.1:4010
[HTTP SERVER] get /pets i info      Request received
[NEGOTIATOR] i info      Request contains an accept header: /**
[VALIDATOR] △ warning   Request did not pass the validation rules
[NEGOTIATOR] • note      Unable to find a 422 response definition
[NEGOTIATOR] • note      Unable to find a 400 response definition.
[NEGOTIATOR] ✓ success   Created a 422 from a default response
[NEGOTIATOR] • note      Unable to find a content with an example defined fo:
[NEGOTIATOR] ✓ success   The response 422 has a schema. I'll keep going with
[NEGOTIATOR] ✓ success   Responding with the requested status code 422
```

# Hosted Mock Server

<https://acme.stoplight.io/mocks/widgets>

Enable Mocking

[Response Body](#)    [Response Headers \[1\]](#)    [Original Request](#)    [Mocked Response](#)

raw     pretty     rendered

```
1  {
2    "id": 1,
3    "name": "get food",
4    "completed": false,
5    "completed_at": "1955-04-23T11:23:59Z",
6    "created_at": "1994-11-05T03:14:37Z",
7    "updated_at": "1989-07-29T11:55:35Z"
8 }
```

 Nauman 4:30 PM  
I need a description field in this endpoint to fulfil my use case.   
BTW the mock was really helpful 

## How/when do we create documentation?

1. CI deploys HTML to S3
2. Read from Git repo on push

# Automatic generated API Reference docs

**POST** /todos

This creates a Todo object.

*I'm a callout. I can be a description, info, error or warning.*

Just some inline code .

Authorization

apikey      Use `?apikey=123` to authenticate requests. It's su  
apiKey      Query parameter name: apikey

Request Body

**Todo Partial {2}** optional

name string required

completed boolean or null required

Responses

201 Developers love examples!

401

500

Schema application/json

**Todo Full {7}** optional

name string required

completed boolean or null required

# Automatic generated Code Samples

The screenshot shows a user interface for generating code samples. At the top, there are tabs: Query, Headers [1], Body, Path [3], Code Generation (which is underlined, indicating it's active), and Mocking. Below the tabs, there's a sidebar with a list of programming languages: Shell, Stoplight Markdown, Java, JavaScript, Node, PHP, Python, Powershell, R, Ruby, Go, C, C#, Obj-C, Swift, OCaml, and HAR. The 'Ruby' item is selected, highlighted with a blue background. To the right of the sidebar, there's a dropdown menu set to 'Ruby' and a 'Copy to Clipboard' button. The main area displays Ruby code:

```
1 require 'uri'
2 require 'net/http'
3 require 'openssl'
4
5 url = URI("https://api.bitbucket.org/2.0/repositories/repo_slug/wc")
6
7 http = Net::HTTP.new(url.host, url.port)
8 http.use_ssl = true
9 http.verify_mode = OpenSSL::SSL::VERIFY_NONE
10
11 request = Net::HTTP::Get.new(url)
12 request["authorization"] = ''
13
14 response = http.request(request)
15 puts response.read_body
```

- How do we keep code and API docs in sync?

API

**name**

string

max length: 20

**email**

format: email

**age**

type: integer

minimum: 18

Ref Docs

**name**

string

max length: 20

**email**

format: email

**age**

type: integer

minimum: 18

 **How to re-use API descriptions as code!**

# SERVER-SIDE VALIDATION

- Model?

# SERVER-SIDE VALIDATION

- Model?
- Controller?

# SERVER-SIDE VALIDATION

- Model?
- Controller?
- View? 

# SERVER-SIDE VALIDATION

- Model?
- Controller?
- View? 
- Service?

# SERVER-SIDE VALIDATION

- Model?
- Controller?
- View? 
- Service?
- Contract?

# SERVER-SIDE RUBY VALIDATION

```
class NewUserContract < Dry::Validation::Contract
  params do
    required(:email).filled(:string)
    required(:age).value(:integer)
  end

  rule(:email) do
    unless /\A[\w+\.-]+\@[a-z\d\-.]+\(\.[a-z\d\-.]+\)*\.\[a-z]+\z/i.match?
      key.failure('has invalid format')
    end
  end

  rule(:age) do
    key.failure('must be greater than 18') if value < 18
  end
end
```

# SERVER-SIDE JAVASCRIPT VALIDATION

```
import Joi from 'joi';

export default {
  createUser: {
    body: {
      username: Joi.string().regex(/^[0-9a-fA-F]{24}$/).required(),
      email: Joi.string().required(),
      age: Joi.number()
    }
  }
};
```

# LOOKS LIKE AN API DESCRIPTION

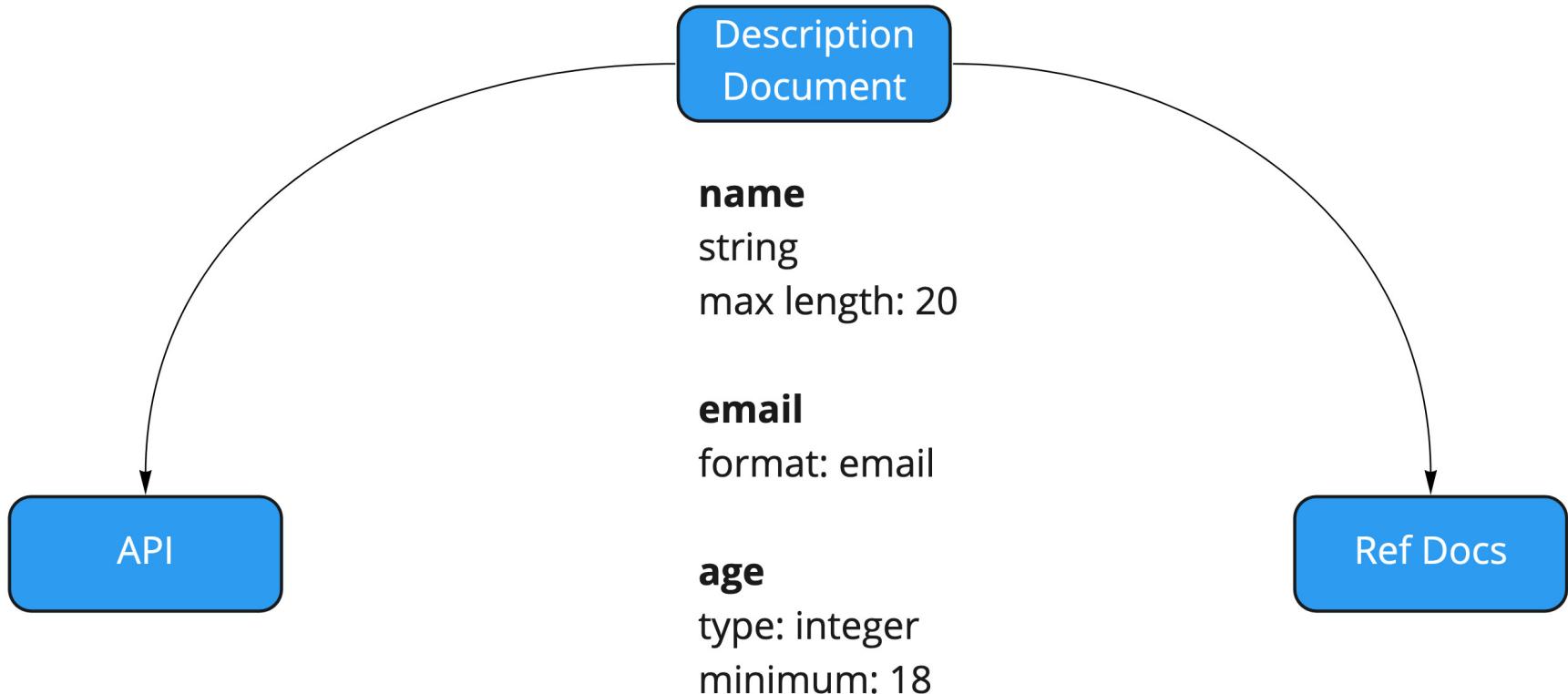
```
type: object
properties:
  username:
    type: string
  email:
    type: string
    format: email
  age:
    type: integer
    minimum: 18
required:
- email
- age
```

Multiple Sources Of Truth = Checking For LIES

Dredd, takes a lot of work and breaks so much teams  
just turn it off

<https://dredd.org/>

# REUSE API DESCRIPTIONS FOR VALIDATION



OpenAPI Validation Middlewares can reject invalid requests with "no code".

```
use League\OpenAPIValidation\PSR15\ValidationMiddlewareBuilder;
use League\OpenAPIValidation\PSR15\SlimAdapter;

$psr15Middleware = (new ValidationMiddlewareBuilder)
    ->fromYamlFile('openapi.yaml')
    ->getValidationMiddleware();

/** @var \Slim\App $app */
$app->add(new SlimAdapter($slimMiddleware));
```

```
# config/application.rb

config.middleware.use Committee::Middleware::RequestValidation,
  schema_path: 'openapi.yaml'
```

```
{  
  "errors": [  
    {  
      "status": "422",  
      "detail": "The property '#/widget/price' of type string did  
      "source": {  
        "pointer": "#/widget/price"  
      }  
    }  
  ]  
}
```

- Ruby: `committee`
- PHP: `league/openapi-psr7-validator`
- NodeJS: `fastify` or `express-ajv-swagger-validation`
- Java/Kotlin: `openapi-spring-webflux-validator`
- Python: `connexion`
- Mojolicious: `Mojolicious`

If you have validation code, you can delete it.

If this is a new application, you don't need to write it.

"Data-store Validations" like "is email unique" still need  
to happen 

Your integration/e2e test suite proves the **requests** work as expected.

Documented requests are now *proven* to be correct.

What about responses?

Middlewares *can* validate responses, but why waste  
time in prod?

Your test suite can contract test **responses** with  
OpenAPI.

```
JsonMatchers.schema_root = "api/models/"

it 'should conform to user schema' do
  get "/users/#{subject.id}"
  expect(response).to match_json_schema('user')
end
```

<https://apisyouwonthate.com/blog/writing-documentation-via-contract-testing>

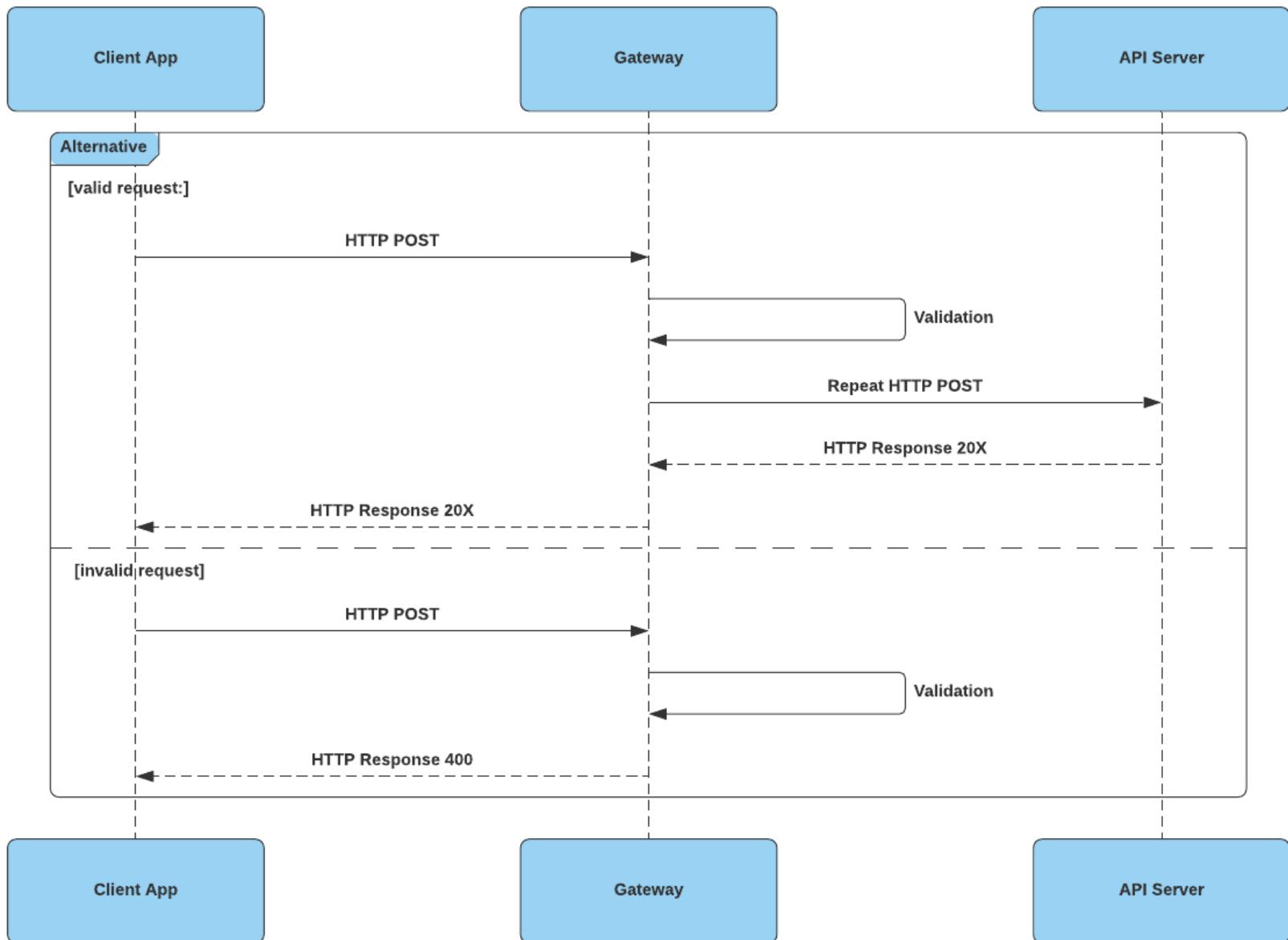
# REQUEST VALIDATION AT API GATEWAY

Middleware a bit slow in prod?

Can't find a middleware?

- AWS API Gateway
- Azure Gateway
- Express Gateway
- Kong
- Tyk

API Gateways can validate your payloads before your application server is called

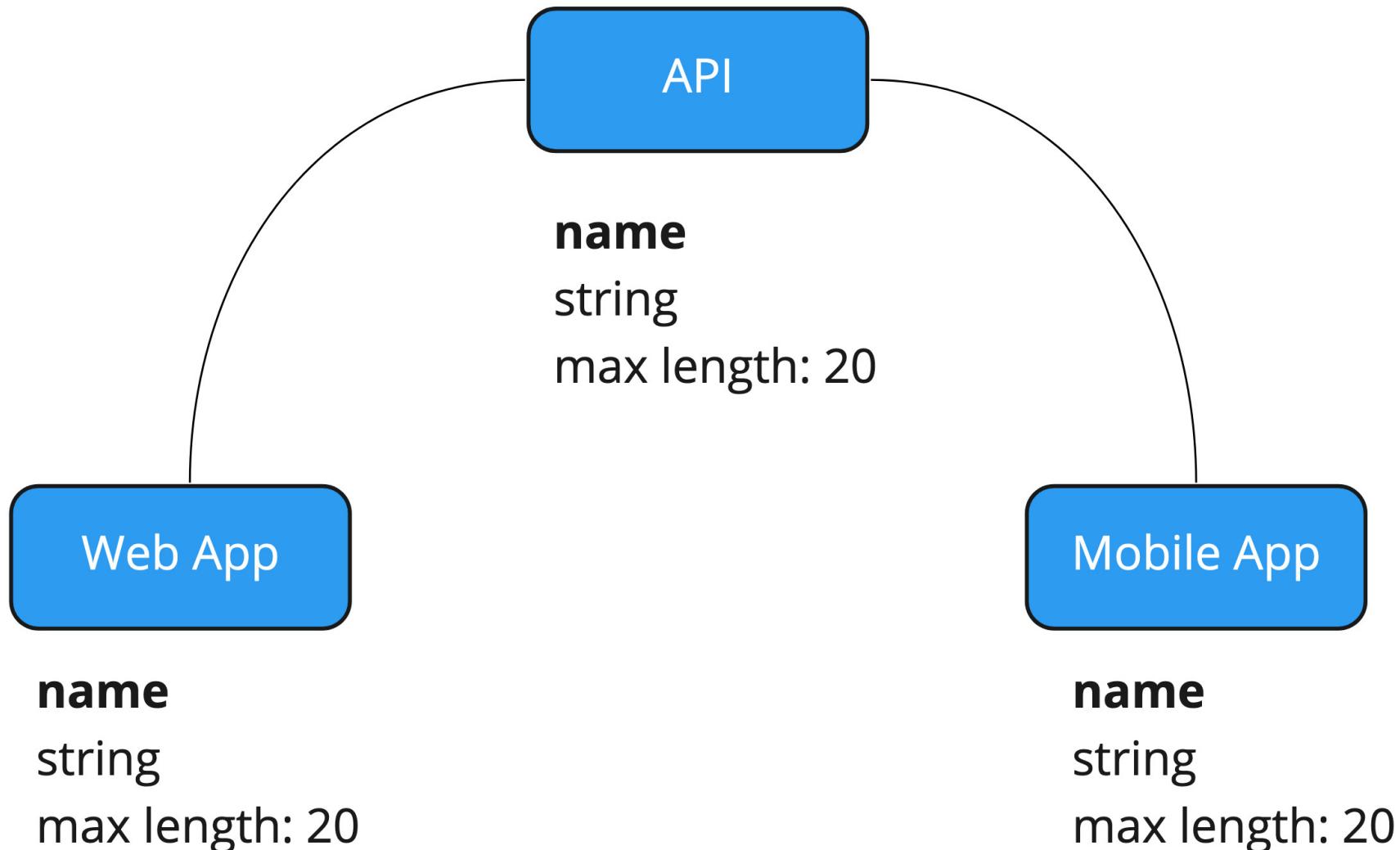


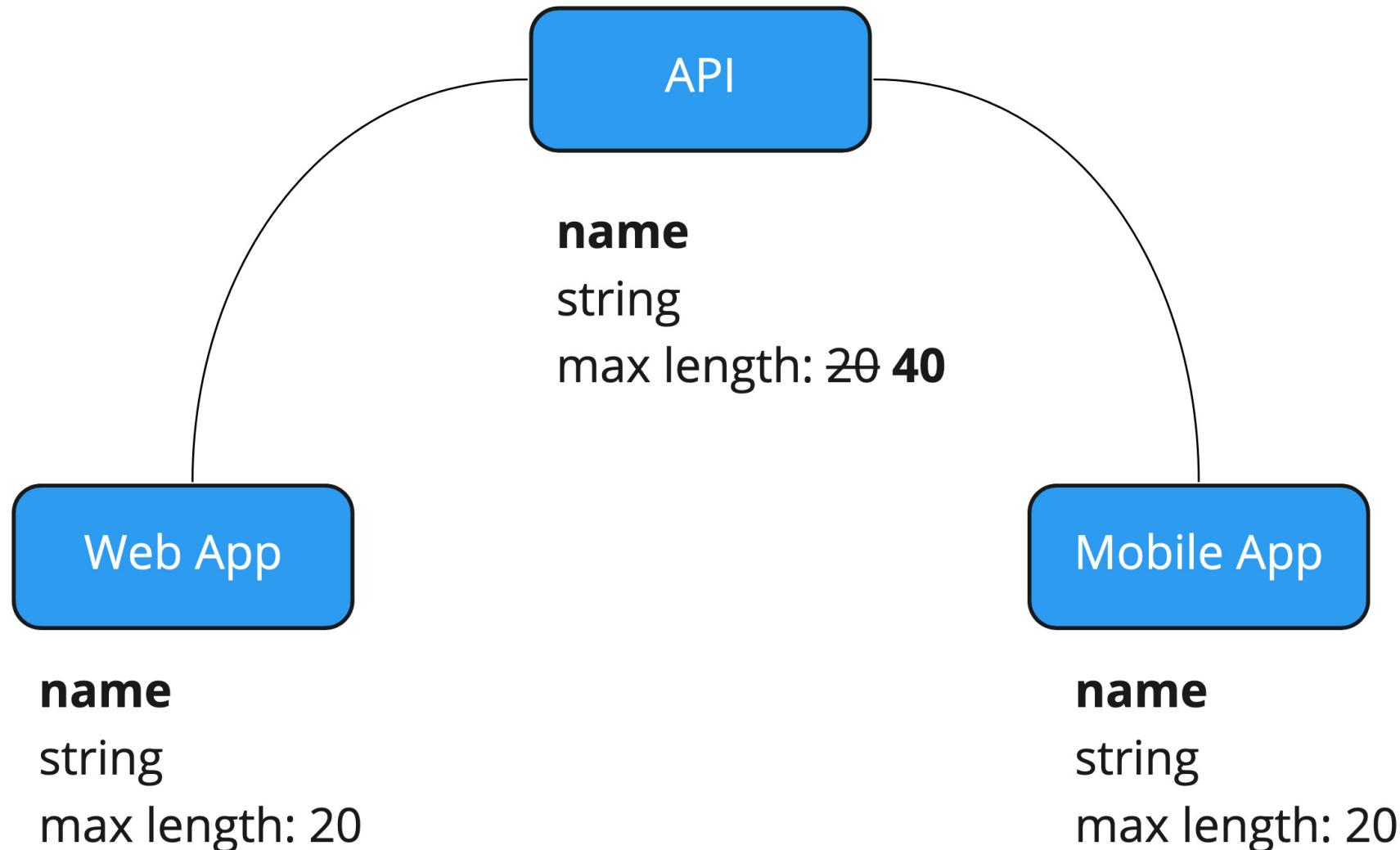
Register a middleware in dev?

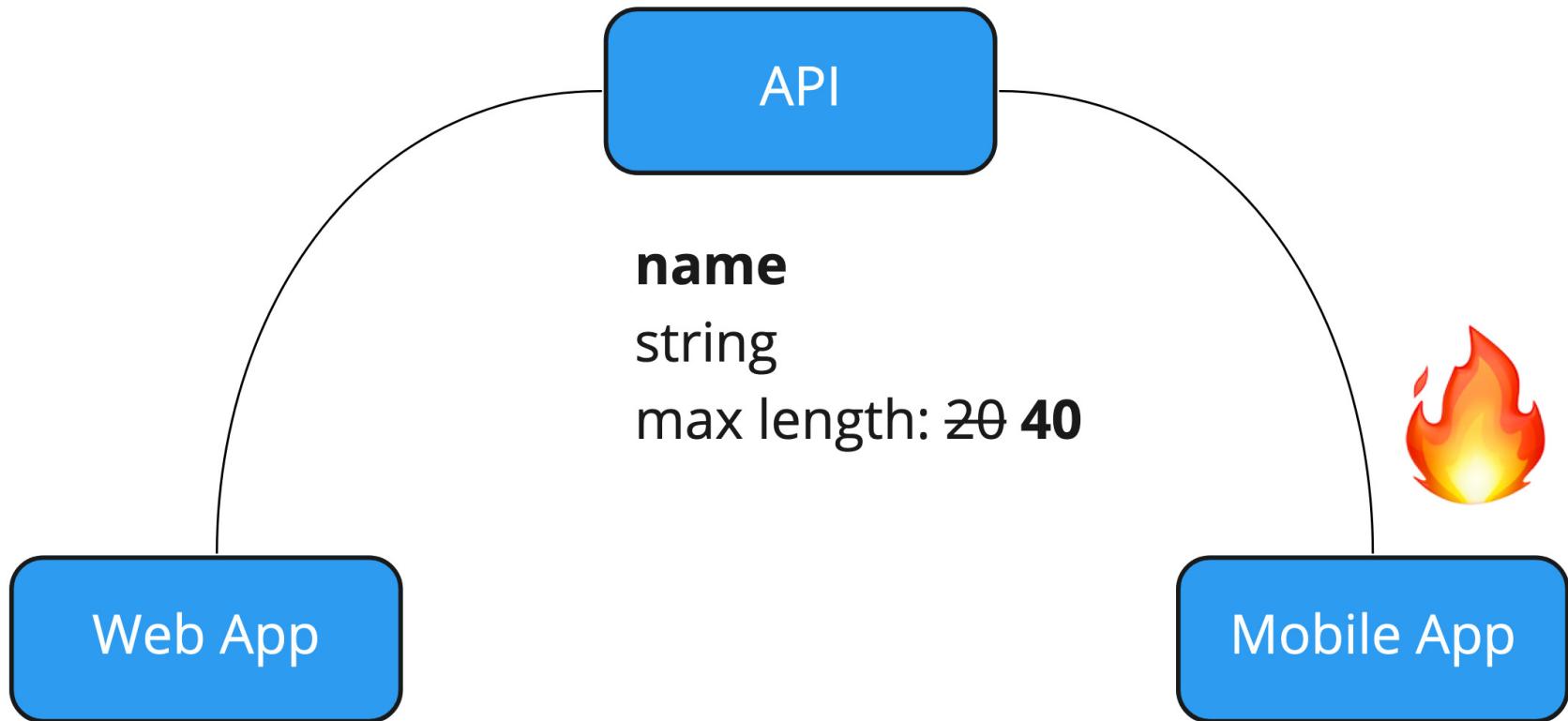
Register a gateway plugin in prod?

Standards compliance *should* mean dev/prod parity.\*

*Another (third) source of truth for validation... the client!*







**name**  
string  
max length: 20 **40**

**name**  
string  
max length: 20

# JSON Schema at runtime

Link: <<http://example.com/schemas/user.json#>>; rel="describedby"

```
const ajv = new Ajv();
const userSchema = require('./cached-schema.json')

function validate(schema, data) {
  return ajv.validate(schema, data)
    ? true : ajv.errors;
}

const input = {
  name: "Lucrezia Nethersole",
  email: "l.nethersole@hotmail.com",
  age: 25
}

validate(userSchema, input); // true
```

```
validate(userSchema, { ...input, email: 123 });

[
  {
    keyword: 'type',
    dataPath: '.email',
    schemaPath: '#/properties/email/type',
    params: { type: 'string' },
    message: 'should be string'
  }
]
```

```
[ 'Name is a required field' ]  
[ 'Email should match format "email"' ]  
[ 'Date Of Birth is in an invalid format, e.g: 1990-12-28' ]  
[ 'Name should NOT be longer than 20 characters' ]
```

API

Web App

**name**  
string  
max length: 20

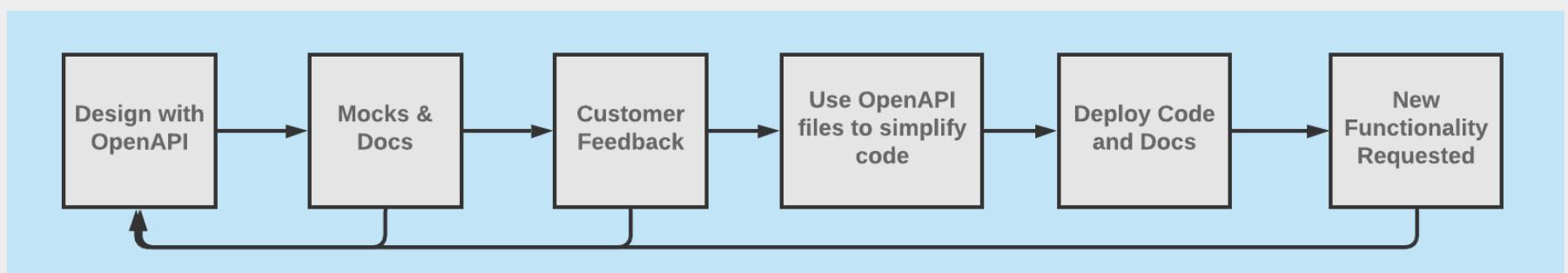
**email**  
format: email

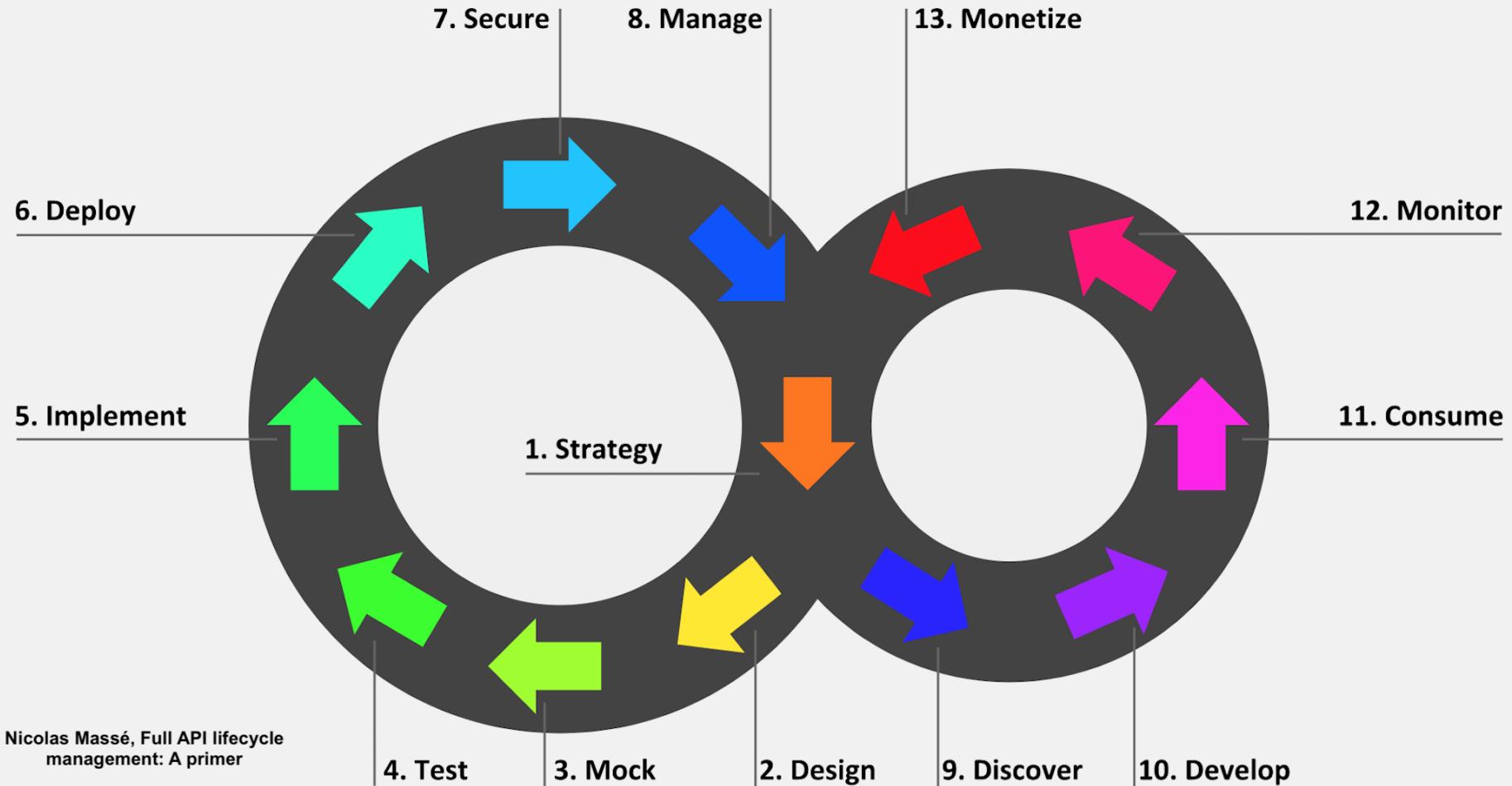
**age**  
type: integer  
minimum: 18

Ref Docs

Mobile App

## Design First, Evolve With Code





Nicolas Massé, Full API lifecycle  
management: A primer

# THANK YOU!

[ApisYouWontHate.com](http://ApisYouWontHate.com)

[Stoplight.io](http://Stoplight.io)

