Describing APIs

- Describe RESTful HTTP APIs in a machine-readable way
- API description is independent of outputs such as documentation
- Enable things that are not "just" documentation

Spec-First API Design

About OpenAPI Spec

API description language formerly known as "Swagger".

Became "OpenAPI Spec" -> v3 released

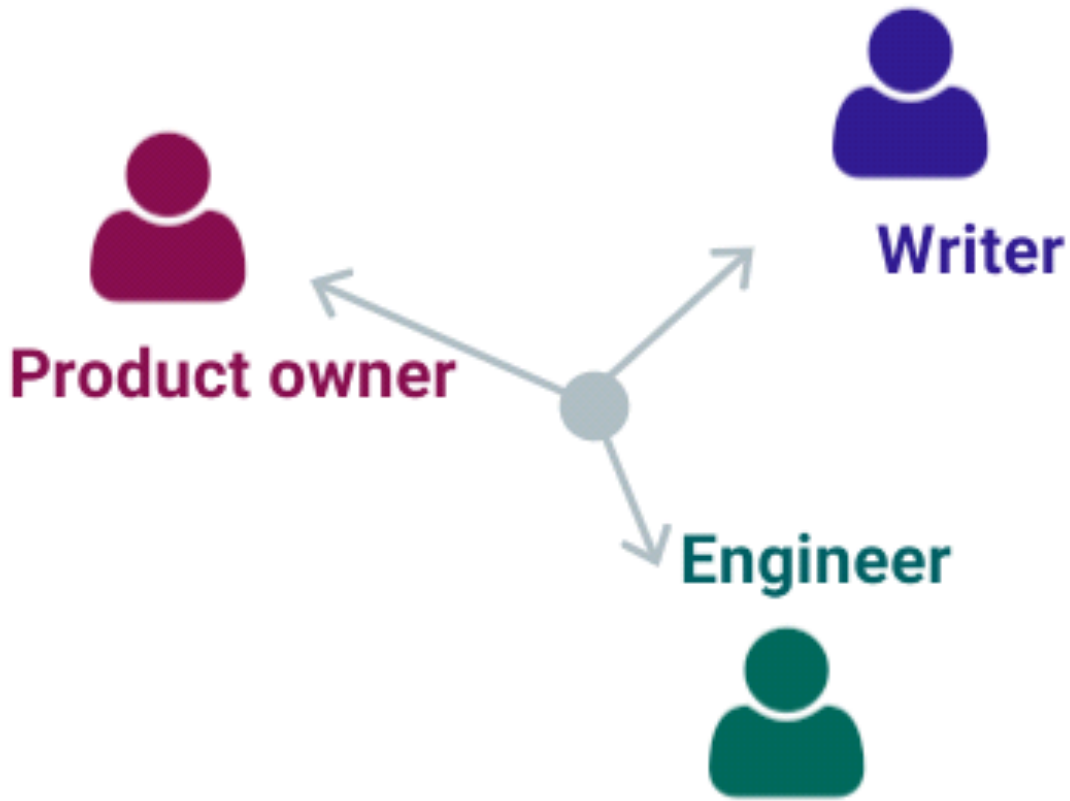
(some tools are still catching up on v3)

New APIs or Existing Ones?

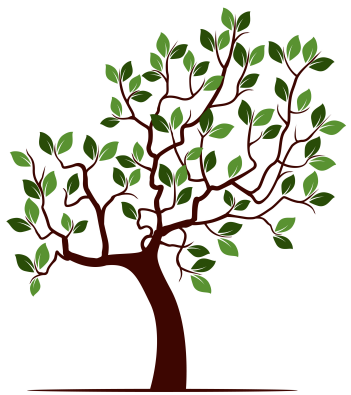
New APIs or Existing Ones?

Yes!

Who Writes OpenAPI Specs?



Anatomy of OpenAPI Spec



Anatomy of OpenAPI Spec

Top-level elements:

- `openapi`
- `info`
- `servers`
- `paths`
- `components`
- `security`
- `tags`

OpenAPI Spec Examples

A JSON or YAML file holds the description (this is YAML)

```yaml
openapi: 3.0.0
servers:
    - url: 'https://api.nexmo.com/ni'
info:
    title: Number Insight API
    version: 1.1.0
    description: >-
        Nexmo's Number Insight API delivers real-time intelligence about the val

... a few hundred more lines here
```

Documenting an Endpoint

```yaml
paths:
  '/basic/{format}':
    parameters:
      - $ref: '#/components/parameters/format'
    get:
      operationId: getNumberInsightBasic
      parameters:
        - $ref: '#/components/parameters/number'
        - $ref: '#/components/parameters/country'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/niResponseJsonBasic'
```

Example Parameter

```yaml
number:
  name: number
  in: query
  description: 'A single phone number that you need insight about in national or
  example: '447700900000'
  required: true
  schema:
    type: string
    pattern: '^[0-9-+\(\)\s]*$'
```

Example Response

```
niResponseJsonBasic:
  type: object
  properties:
    status:
      $ref: '#/components/schemas/niBasicStatus'
    status_message:
      type: string
      description: 'The status description of your request.'
      example: 'Success'
    request_id:
      type: string
      description: 'The unique identifier for your request. This is a alphanumer
      example: 'aaaaaaaa-bbbb-cccc-dddd-0123456789ab'
      maxLength: 40

    ...
```

That looks complicated!

Rendered Example: ReDoc

Retrieve API Secrets

AUTHORIZATIONS: basicAuth

PATH PARAMETERS

account_id
required

string
Example: "abcd1234"
ID of the account

Responses

200 API secret response

RESPONSE SCHEMA: application/json

secrets >

object
A list of secrets under the "secrets" key.

401 Credential is missing or invalid

404 Resource could not be found

GET /accounts/{account_id}/secrets

Response samples

200 401 404

application/json

Copy Expand all Collapse all


```
{
  - "secrets": {
      - "secrets": [
          - {
              "id": "ad6dc56f-07b5-46e1-a527-
              85530e625800",
              "created_at": "2017-03-02T16:34:49Z"
            }
        ]
    }
}
```

Rendered Example: Nexmo

Retrieve API Secrets

`GET` `https://api.nexmo.com/accounts/:account_id/secrets`

Authentication

Key	Description	Example	Default
Authorization	**Base64 encoded API key and secret joined by a colon.** Read more	`Basic <base64>`	None

Path Parameters

Key	Description	Example	Default
account_id REQUIRED | string	ID of the account	`abcd1234`	None

View response field descriptions

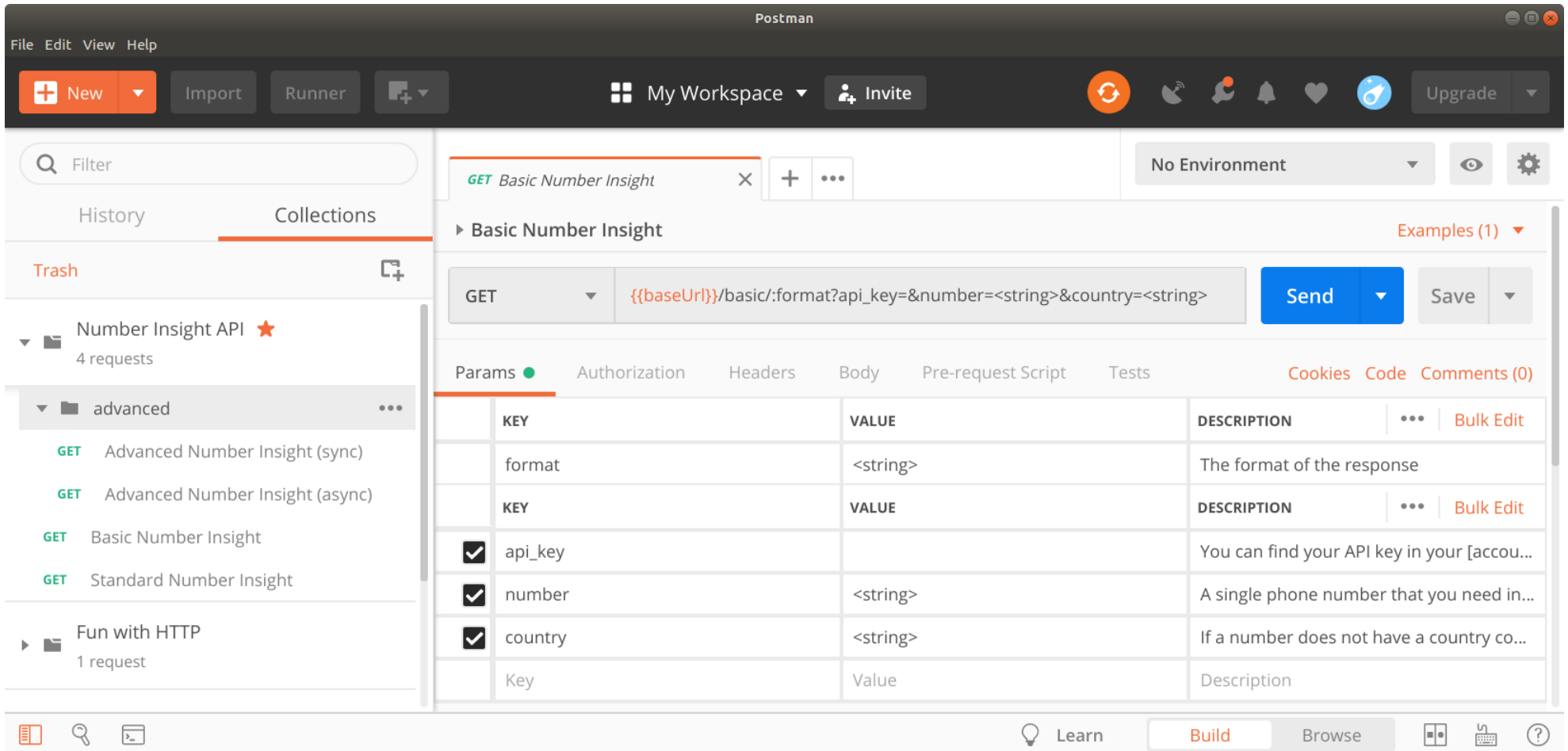
HTTP response 200


```
{
  "secrets": {
    "secrets": [
      {
        "id": "ad6dc56f-07b5-46e1-a527-8553
        "created_at": "2017-03-02T16:34:49Z
      }
    ]
  }
}
```

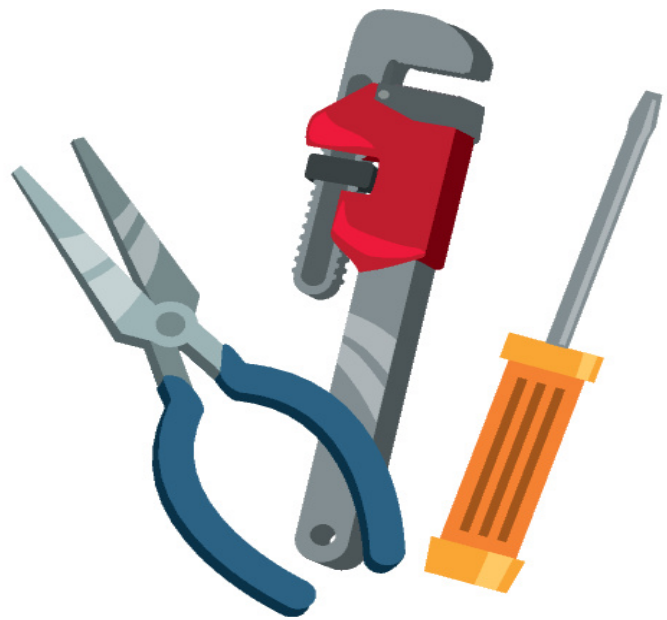

HTTP response 401

HTTP response 404

Imported into Postman



Tools To Get Things Done



Please use Source Control

See also: https://gitworkbook.com

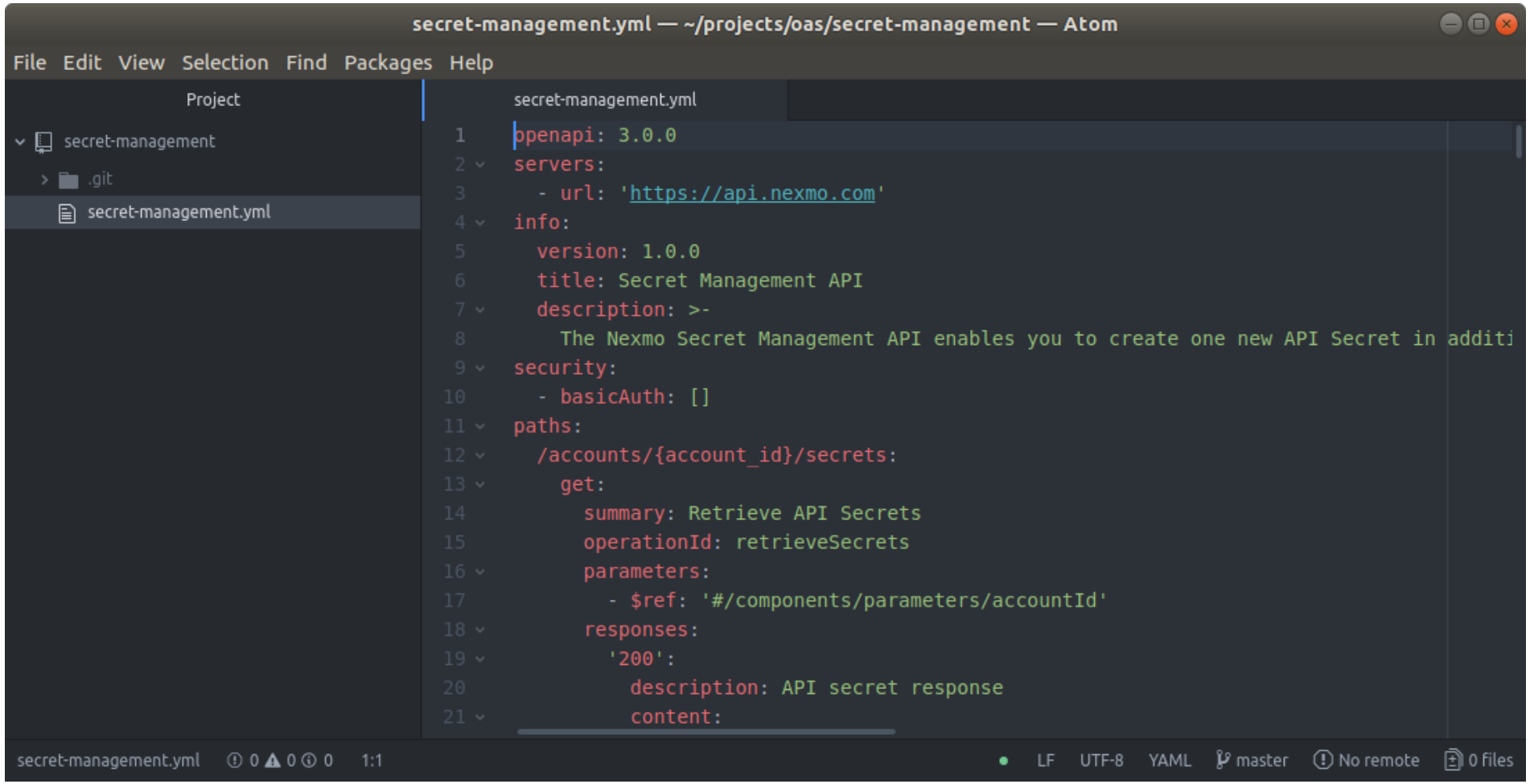
Editing Tools

There are editors with helpful tools

- I like Atom with `linter-swagger` https://atom.io
- Try SwaggerUI, SwaggerHub, etc https://swagger.io/tools/
- APICurio Studio gets good reviews https://www.apicur.io/
- Stoplight looks interesting https://stoplight.io

(feel free to tweet your best tools at me, I'll share them all later)

OAS in Atom


secret-management.yml — ~/projects/oas/secret-management — Atom
File Edit View Selection Find Packages Help
Project
secret-management
.git
secret-management.yml
secret-management.yml
openapi: 3.0.0
servers:
- url: 'https://api.nexmo.com'
info:
version: 1.0.0
title: Secret Management API
description: >-
The Nexmo Secret Management API enables you to create one new API Secret in additi
security:
- basicAuth: []
paths:
/accounts/{account_id}/secrets:
get:
summary: Retrieve API Secrets
operationId: retrieveSecrets
parameters:
- $ref: '#/components/parameters/accountId'
responses:
'200':
description: API secret response
content:
secret-management.yml 0 0 0 1:1 LF UTF-8 YAML master No remote 0 files

Validation Tools

Tools that check or "lint" your file.

- Speccy is a CLI tool with configurable rules http://speccy.io/
- Open API Spec Validator https://github.com/p1c2u/openapi-spec-validator

Set up in your editor or use a `watch` command, e.g.:

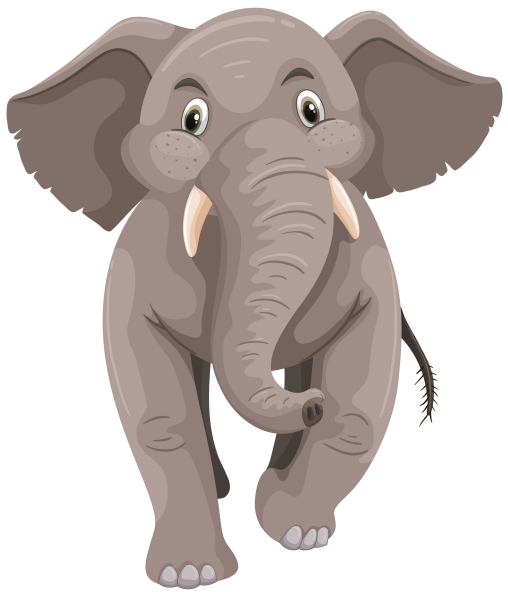
```
watch -n 1 speccy lint myspec.yml
```

Preview Tools

OAS is a standard! So any preview should do:

- ReDoc is great https://github.com/Rebilly/ReDoc
- Speccy also wraps ReDoc for its `serve` command
- You can run OpenApi-GUI locally https://github.com/mermade/openapi-gui

Creating OpenAPI Specs is like eating an elephant



Uses for OpenAPI Spec

Resources

- https://www.openapis.org
- https://apievangelist.com
- https://speccy.io
- https://github.com/Rebilly/ReDoc
- https://openapi.tools
- https://github.com/openapitools/openapi-generator

Bonus Extra Slides

Code Generators

Libraries can be generated as easily as docs.

For PHP (and so many other languages) try
https://github.com/openapitools/openapi-generator

Pull docker image, generate PHP code from your OpenAPI Spec

Code Generator Example

```php
$config->setUsername(NEXMO_API_KEY);
$config->setPassword(NEXMO_API_SECRET);

$client = new OpenAPI\Client\Api\DefaultApi(
    new GuzzleHttp\Client(), $config);
$obj = new \OpenAPI\Client\Model\InlineObject();

try {
    $result = $client->retrieveSecrets(NEXMO_API_KEY, $obj);
    print_r($result);
} catch (Exception $e) {
    echo 'Exception when calling DefaultApi->retrieveSecrets: ', $e->getMessag
}
```

Code Generator Example

```
 1 $config->setUsername(NEXMO_API_KEY);
 2 $config->setPassword(NEXMO_API_SECRET);
 3
 4 $client = new OpenAPI\Client\Api\DefaultApi(
 5     new GuzzleHttp\Client(), $config);
 6 $obj = new \OpenAPI\Client\Model\InlineObject();
 7
 8 try {
 9     $result = $client->retrieveSecrets(NEXMO_API_KEY, $obj);
10     print_r($result);
11 } catch (Exception $e) {
12     echo 'Exception when calling DefaultApi->retrieveSecrets: ', $e->getMessag
13 }
```