

APIOps

Daniel Kocot, Senior Solution Architect / Head of API Experience & Operations

Name: Daniel Kocot

Role: Senior Solution Architect / Head of API
Experience & Operations

Email: daniel.kocot@codecentric.de

Twitter: @dk_1977

LinkedIn: <https://www.linkedin.com/in/danielkocot/>



Continuous
Deployment
Integration
Code
Lead
Fast
market
GitOps
Agile
Everything
Delivery
DevOps
Failure
Mean
Collaboration
DevSecOps
BizOps

No Buzzword Bingo!

CALMS Model

Collaboration

Automation

Lean Principles and Processes

Measurement

Sharing

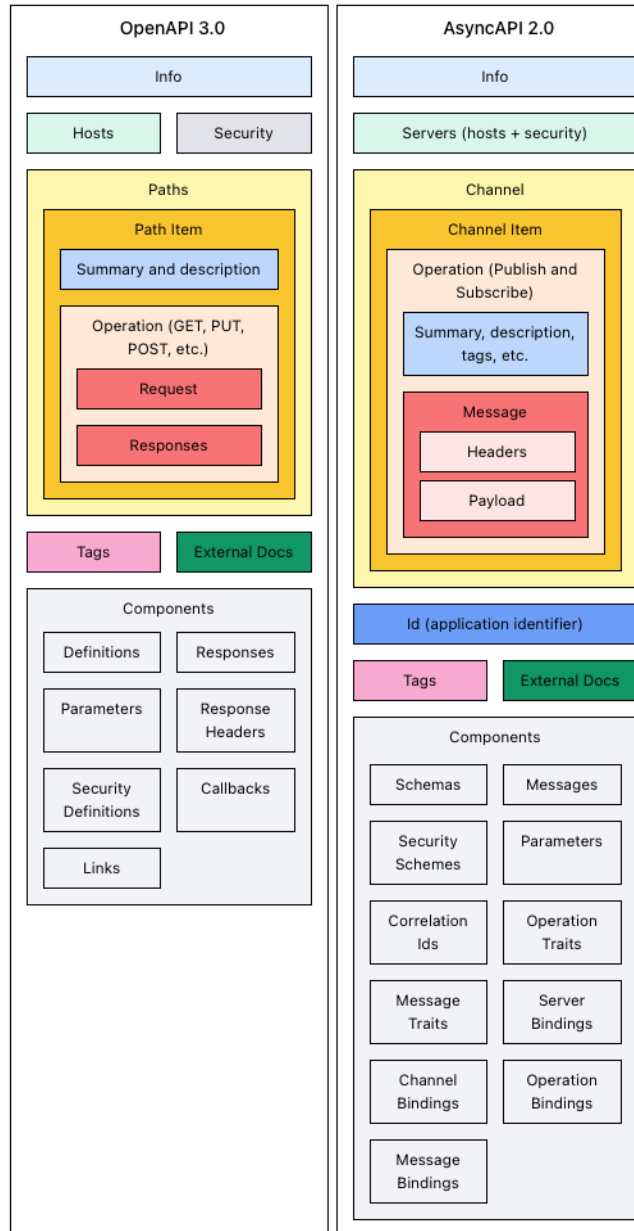
API first

- An API is the first (and often only) interface to users of an application
- An API comes first — before the implementation
- An API is described (documented) or self-descriptive

A close-up photograph of a person's hands, which are heavily smeared with various colors of paint (red, yellow, green, blue). The person is wearing a light blue button-down shirt. They are holding two paintbrushes; the one in the right hand is thick with yellow and green paint. In the foreground, there are several small containers of paint in green, yellow, and red, and a white surface with paint splatters. A semi-transparent white banner with the text 'API Design (first)' is centered over the image.

API Design (first)

OpenAPI / AsyncAPI



OpenAPI Spec Example

News API

Overview

ENDPOINTS

news

getNews

SCHEMAS

ArticleList

Article

Error

powered by Stoplight

getNews

gets latest news

Request

GET /news

Responses 200 404

Expected response to a valid request

Body

application/json

array of:

id integer required

title string required

date string<date> required

description string required

imageUrl string required

GET /news

Server 1

Send Request

Request Sample: Shell / cURL

```
curl --request GET \
--url http://localhost:8080/api/news \
--header 'Content-Type: application/json'
```

Response Example

```
1 [
2   {
3     "id": 0,
4     "title": "string",
5     "date": "2019-08-24",
6     "description": "string",
7     "imageUrl": "string"
8   }
9 ]
```

Specification Version


main ▾


OpenAPI-Specification / versions /

Go to file








Add file ▾

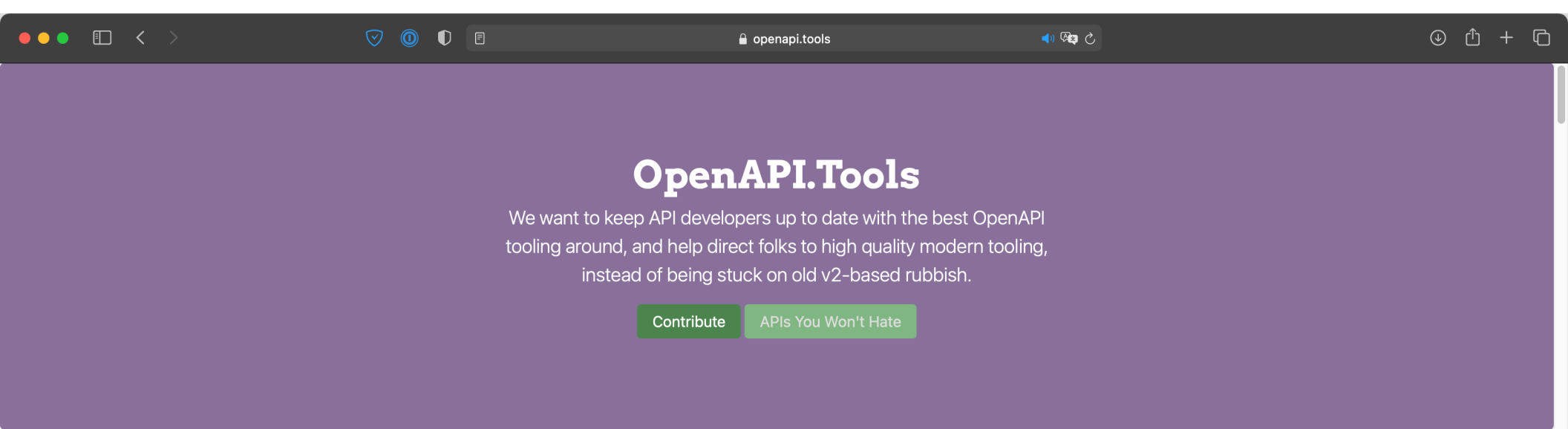
...

 30 authors 3.1.0 Release (#2462) ...

✖ 42a9e3d on 16 Feb  History

..

 1.2.md	License and link cleanup	6 years ago
 2.0.md	Replace <code>#124;</code> with <code>\\</code>	5 years ago
 3.0.0.md	link change	4 years ago
 3.0.1.md	Merge pull request #1430 from OAI/release-prep	4 years ago
 3.0.2.md	Update release date	3 years ago
 3.0.3.md	OAS v3.0.3 Release (#2148)	2 years ago
 3.1.0.md	3.1.0 Release (#2462)	7 months ago



Tool Types

We've organised everything into categories so you can jump to the section you're interested in.

- **Auto Generators:** Tools that will take your code and turn it into an OpenAPI Specification document
- **Converters:** Various tools to convert to and from OpenAPI and other API description formats.
- **Data Validators:** Check to see if API requests and responses are lining up with the API description.
- **Description Validators:** Check your API description to see if it is valid OpenAPI.
- **Documentation:** Render API Description as HTML (or maybe a PDF) so slightly less technical people can figure out how to work with the API.
- **DSL:** Writing YAML by hand is no fun, and maybe you don't want a GUI, so use a Domain Specific Language to write OpenAPI in your language of choice.
- **GUI Editors:** Visual editors help you design APIs without needing to memorize the entire OpenAPI specification.
- **Learning:** Whether you're trying to get documentation for a third party API based on traffic, or are trying to switch to design-first at an organization with no OpenAPI at all, learning can help you move your API spec forward and keep it up to date.
- **Miscellaneous:** Anything else that does stuff with OpenAPI but hasn't quite got enough to warrant its own category.
- **Mock Servers:** Fake servers that take description document as input, then route incoming HTTP requests to example responses or dynamically generates examples.
- **Parsers:** Loads and read OpenAPI descriptions, so you can work with them programmatically.
- **SDK Generators:** Generate code to give to consumers, to help them avoid interacting at a HTTP level.
- **Security:** By poking around your OpenAPI description, some tools can look out for attack vectors you might not have noticed.
- **Server Implementations:** Easily create and implement resources and routes for your APIs.
- **Testing:** Quickly execute API requests and validate responses on the fly through command line or GUI interfaces.
- **Text Editors:** Text editors give you visual feedback whilst you write OpenAPI, so you can see what docs might look like.

Auto Generators

Tools that will take your code and turn it into an OpenAPI Specification document

Name	Language	v3.1	v3.0	v2.0	GitHub
------	----------	------	------	------	--------

Tooling



GIT

- some kind of GitFlow
- no direct commits to main Branch
- every change to specification and the pipeline has to be a pull request



Developer Setup

IDE or Editor

- Eclipse
- JetBrains Products
- Visual Studio Code
- Stoplight Studio
- Apicurio Studio
- Insomnia

local validation

- Redocly-CLI
- Spectral

Spectral

```
> npm install -D @stoplight/spectral  
> npx spectral lint news.yaml  
OpenAPI 3.x detected  
No results with a severity of 'error' or higher found!
```

Ruleset

```
formats:
  - oas3.0
extends:
  - 'spectral:oas'
rules:
  tags-have-description:
    description: Tags must have a description.
    message: Description of Tag is missing
    given: $.tags[*]
    recommended: true
    type: style
    then:
      field: description
      function: truthy
```

local mocking

Prism

```
> npm install -D @stoplight/prism-cli
> npx prism mock news.yaml -p 8080
[17:13:00] › [CLI] ... awaiting Starting Prism...
[17:13:01] › [CLI] i info GET http://127.0.0.1:8080/news
[17:13:01] › [CLI] ► start Prism is listening on http://127.0.0.1:8080
```

local testing

Contract testing

- Create a test suite based directly on the spec
 - Using a BDD framework
- Create a test suite based on a postman collection

Portman

```
> npm install -D @apideck/portman  
> prism mock specs/news.yaml -p 8080 | portman -l specs/news.yaml -n
```

Portman

```
=====
Local Path:          specs/news.yaml
Portman Config:      portman-config.default.json
Postman Config:      postman-config.default.json
Environment:         .env
Inject Tests:        true
Run Newman:          true
Newman Iteration Data: false
Upload to Postman:    false
=====

  ✓ Conversion successful
=====

Run Newman against:
=====

newman

News API

❑ news
  ↳ get News
    GET http://localhost:8080/news [200 OK, 384B, 85ms]
    ✓ [GET]::/news - Status code is 2xx
    ✓ [GET]::/news - Content-Type is application/json
    ✓ [GET]::/news - Response has JSON Body
```

Load testing



- Smoke
- Load
- Stress
- Soak

postman-to-k6

```
> npm install -D postman-to-k6  
> mkdir k6  
> npx postman-to-k6 post-collections/news-postman-collection.json -o k6/news-k6-script.js
```

k6

```
> prism mock specs/news.yaml -p 8080 | k6 run k6/news-k6-script.js
```


k6



```
execution: local
  script: k6/news-k6-script.js
  output: -
```

```
scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
  * default: 1 iterations for each of 1 VUs (maxDuration: 10m0s, gracefulStop: 30s)
```

```
running (00m00.0s), 0/1 VUs, 1 complete and 0 interrupted iterations
```

```
default ✓ [=====] 1 VUs  00m00.0s/10m0s  1/1 iters, 1 per VU
```

```
data_received.....: 502 B    26 kB/s
data_sent.....: 134 B    7.0 kB/s
http_req_blocked.....: avg=1.31ms  min=1.31ms  med=1.31ms  max=1.31ms  p(90)=1.31ms  p(95)
http_req_connecting.....: avg=237µs  min=237µs  med=237µs  max=237µs  p(90)=237µs  p(95)
http_req_duration.....: avg=12.92ms min=12.92ms med=12.92ms max=12.92ms p(90)=12.92ms p(95)
http_req_failed.....: 100.00% ✓ 1      ✗ 0
http_req_receiving.....: avg=110µs  min=110µs  med=110µs  max=110µs  p(90)=110µs  p(95)
```

OpenAPI

News API

Overview

ENDPOINTS

news

getNews GET

SCHEMAS

ArticleList

Article

Error

powered by Stoplight

getNews

gets latest news

Request

GET /news

Responses 200 404

Expected response to a valid request

Body

application/json

array of:

id	integer	required
title	string	required
date	string<date>	required
description	string	required
imageUrl	string	required

GET /news Server 1

Send Request

Request Sample: Shell / cURL

```
curl --request GET \
--url http://localhost:8080/api/news \
--header 'Content-Type: application/json'
```

Response Example

```
1 [
2   {
3     "id": 0,
4     "title": "string",
5     "date": "2019-08-24",
6     "description": "string",
7     "imageUrl": "string"
8   }
9 ]
```

YAML / JSON

- YAML is *more* human-readable
- JSON is *more* machine-readable
- Parsing JSON is faster ;)

Converting YAML to JSON

```
> npm install -g yaml2json  
> yaml2json specs/news.yaml
```

Structure

<https://openapi-map.apihandyman.io>

**Splitting the structure for reuse and better
overview == Design Library**

Hard splitting

- one file per object

Soft splitting

- Depending on the size of the whole document or the objects

Use of references with \$ref

local

```
'#/components/schemas/myElement'
```

remote

```
'myElement.yaml'
```

url

```
'http://path/to/your/myElement.yaml'
```

Something is needed to rebundle the files to one

```
> npx @redocly/redocly-cli  
> redocly bundle specs/news.yaml --output output/news.yaml
```

Use of OpenAPI Extensions/X-Objects to handle own or vendor needs

- x-vendor-...
- X-...

Supported by:

- root level
- info
- paths
- operation parameters
- responses
- tags
- security schemes

From API description to configuration as code

- OpenAPI with Extensions
- AWS Cloudformation
- AWS CDK
- Azure ARM Templates
- Azure Bicep
- Pulumi

Example AWS Cloudformation - API Spec

```
openapi: 3.0.0
info:
  title: API Gateway OpenAPI Example
  version: 1.0.0

paths:
  /api/posts:
    get:
      summary: List Posts
      operationId: listPosts
      requestBody:
        required: true
        content:
          application/json:
            schema:
              '$ref': '#/components/schemas/CreatePostRequestBody'
      responses:
        '200':
          description: Retrieve the list of Posts
          content:
            application/json:
              schema:
                '$ref': '#/components/schemas/ListPostsResponseBody'
  x-amazon-apigateway-integration:
```

Example AWS Cloudformation - S3 Bucket Stack

```
AWSTemplateFormatVersion: 2010-09-09

Resources:
  ArtifactBucket:
    Type: AWS::S3::Bucket

Outputs:
  ArtifactBucket:
    Description: The name of the artifact bucket
    Value: !Ref ArtifactBucket
    Export:
      Name: !Sub ${AWS::StackName}-artifact-bucket
```

Example AWS Cloudformation - AWS API-Gateway Stack

```
AWSTemplateFormatVersion: '2010-09-09'

Parameters:

  ProjectId:
    Type: String
    Default: experiment

  Bucket:
    Type: String
    Default: api-gateway-openapi-artifact-bucket-1wmq2pswrxwjw

  OpenAPIS3Key:
    Type: String
    Default: openapi.yaml

Resources:

  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: !Ref AWS::StackName
      Description: 'An experimental API'
      FailOnWarnings: true
```

Some gateways vendors
have their own toolsets for
CaC which have to be
integrated in an existing
toolchain

For example:

- Kong
 - deck
 - Inso (Insomnia CLI)
- Tyk
 - Tyk Sync

Deployable Infrastructure based on the definition

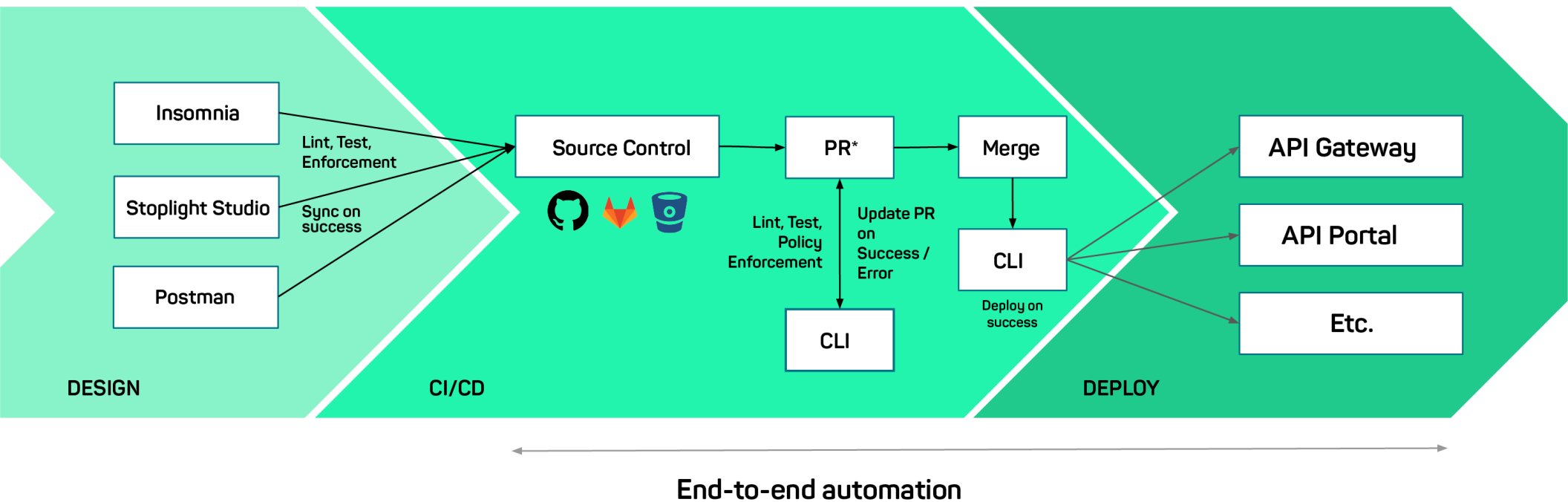
- Gateways
- Portals
- Hubs
- Registries

Transformation to automation within CI/CD

- API first
- GIT Process
- Well structured and formed API specification
- Automated Linting
- Automated Testing
- Automated Deployment of relevant infrastructure

APIOps in action

Run unit tests, lint specification, enforce policies, generate config, and deploy configuration / specification



*PR = Pull Request

Should we build a framework on our own for this?

Missing Parts

Building SDKs

Security

- OWASP API Top 10
- Security Best Practices

Policy (as Code)

- OPA
- Sentinel

Wrap Up

Posts on codecentric blog:

<https://blog.codecentric.de/en/author/daniel-kocot/>

Posts on my blog:

<https://danielkocot.github.io>

Posts on Medium:

<https://medium.com/@daniel.kocot>

Q&A



Thank you



References

- Photo on slide 7 by [Alice Dietrich](#) on [Unsplash](#)
- Photo on slide 13 by [Danial Igder](#) on [Unsplash](#)

@codecentric