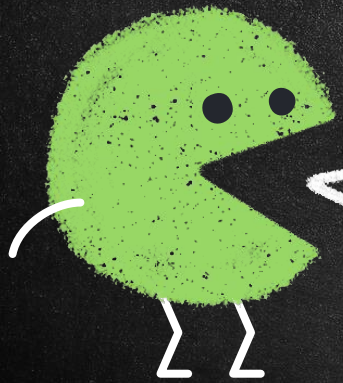
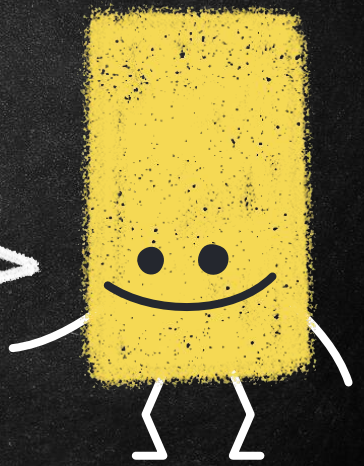
# Pipeline de APIs para o século XXI

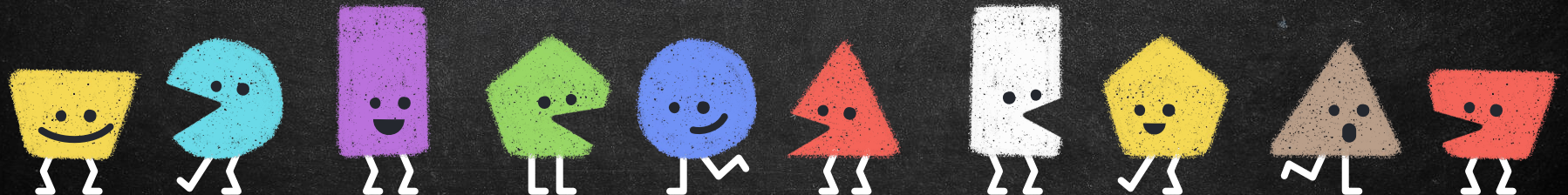





# Hello!

**I am Marcelo Marinho**

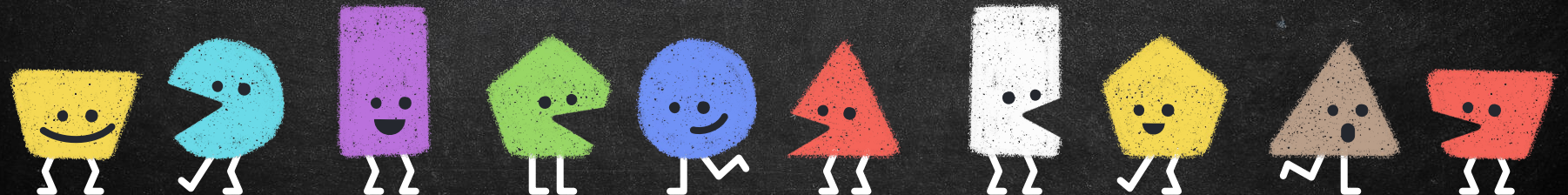
Development Specialist @luizalabs



# Hello!

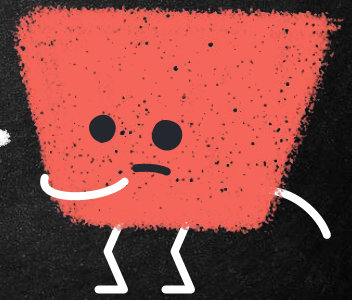
**I am Cláudio de Oliveira**

Tech Lead API Team @luizalabs



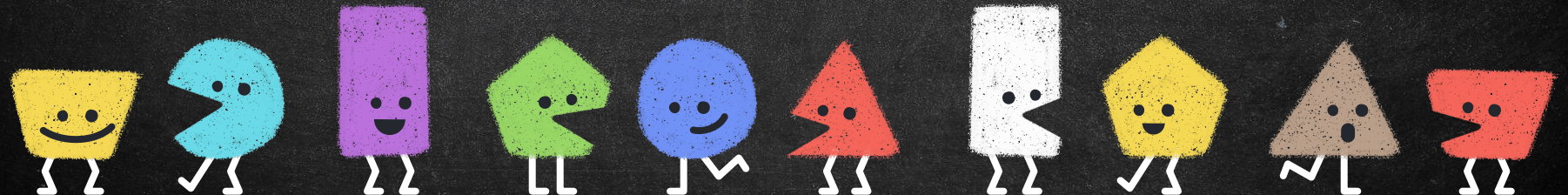
# Agenda

- Fundamentação
  - OpenAPI
  - git
  - GitOPS
- CI
  - Argo Workflow
  - Argo Events
- CD
  - Argo CD

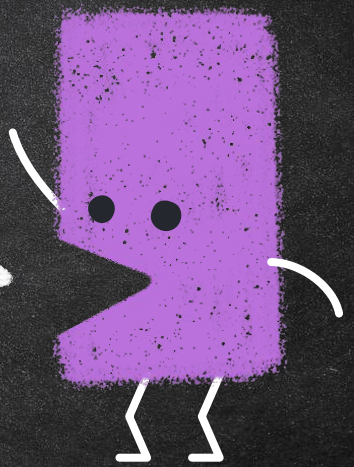


# OpenAPI

Tudo começa aqui!!!



> *The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service* ***without requiring access to source code****, additional documentation, or inspection of network traffic*

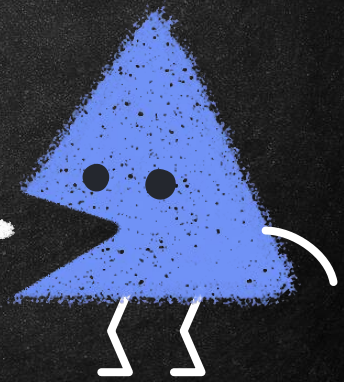


https://spec.openapis.org/oas/latest.html

A idéia principal é fazer com que as outras pessoas possam usar suas funções providas por um serviço sem a necessidade de entender nos detalhes a implementação

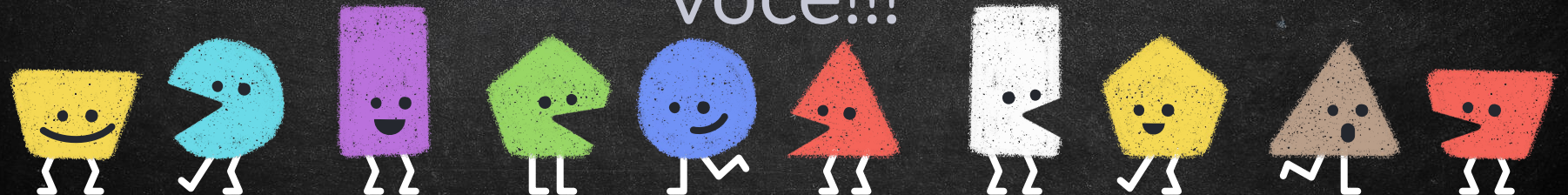
# Estabelecimento de contrato


Quando um artefato OpenAPI é produzido é estabelecido um contrato, clientes se “espelham” no contrato para implementar suas aplicações



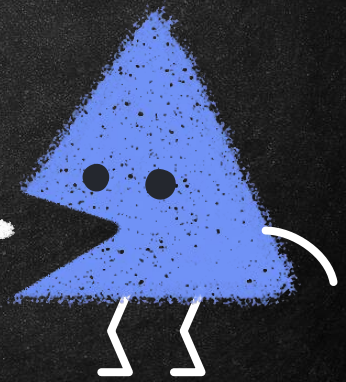
# Automação

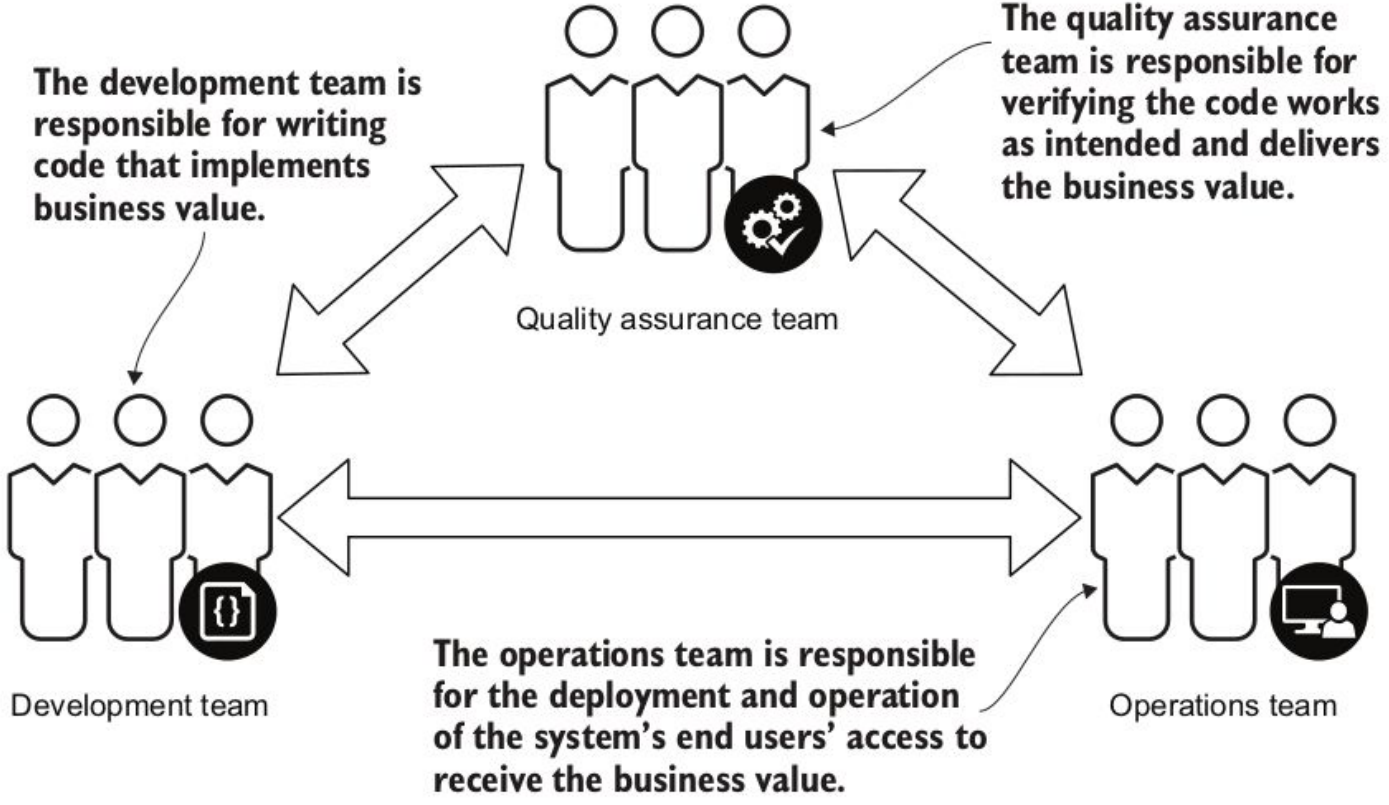
Alguém faz isso pra você!!!



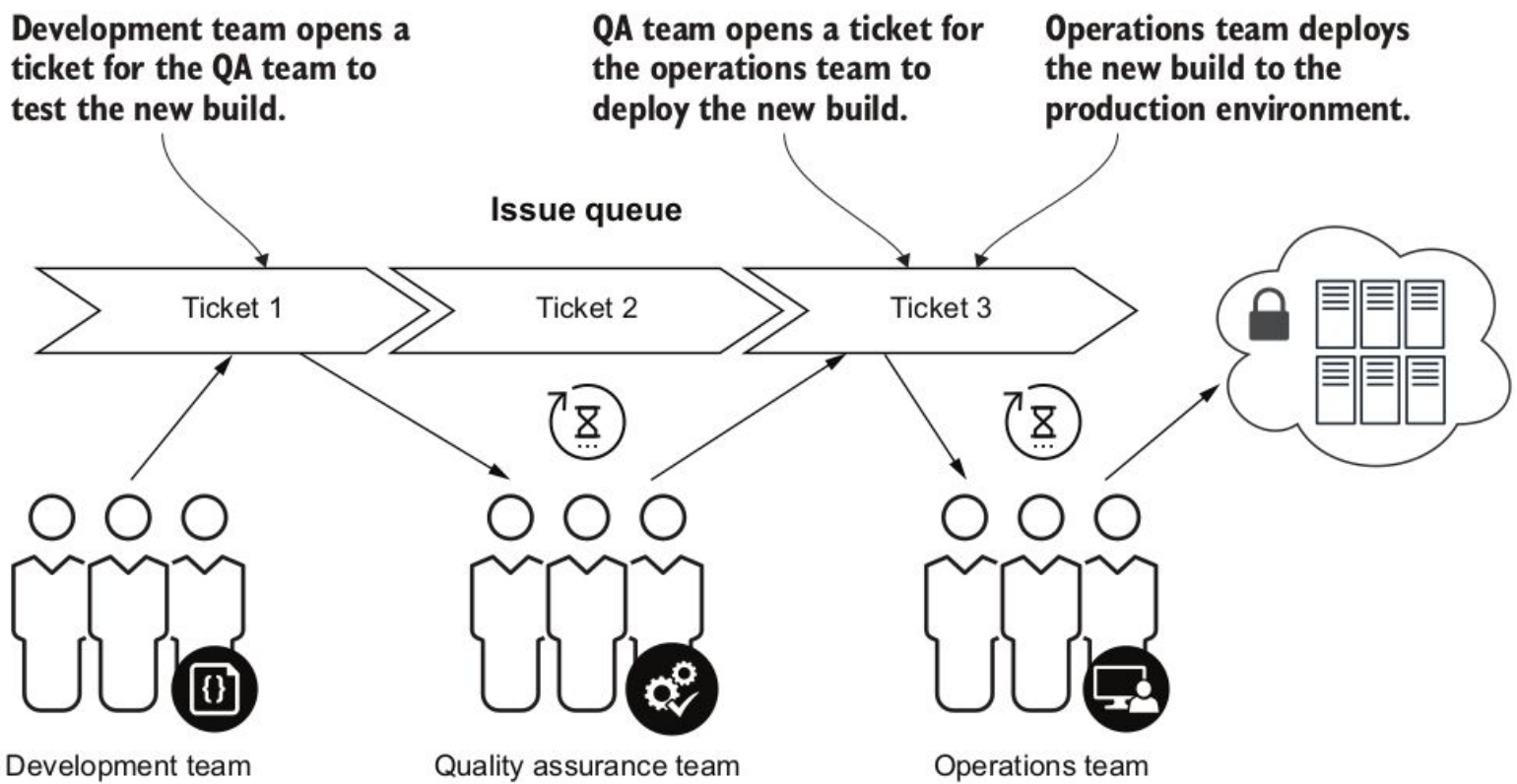
# Automação

Retirar processos manuais e prover
retirar processos manuais e prover
velocidade e consistência.





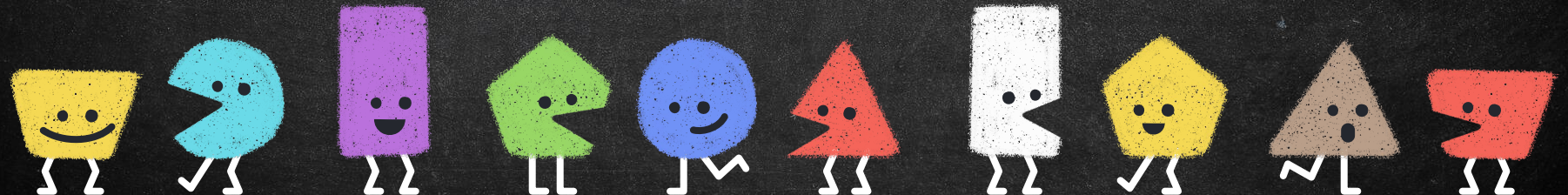

https://www.manning.com/books/gitops-and-kubernetes

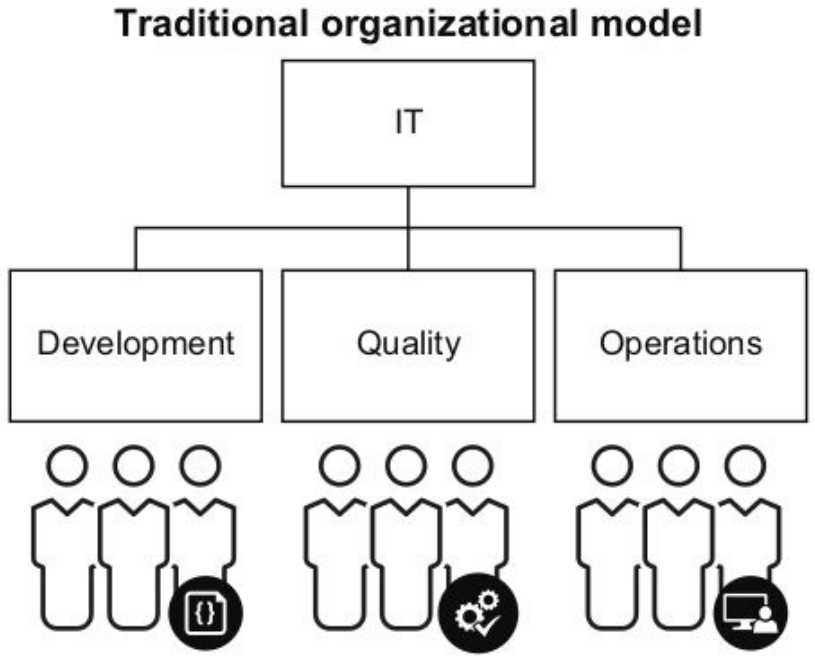


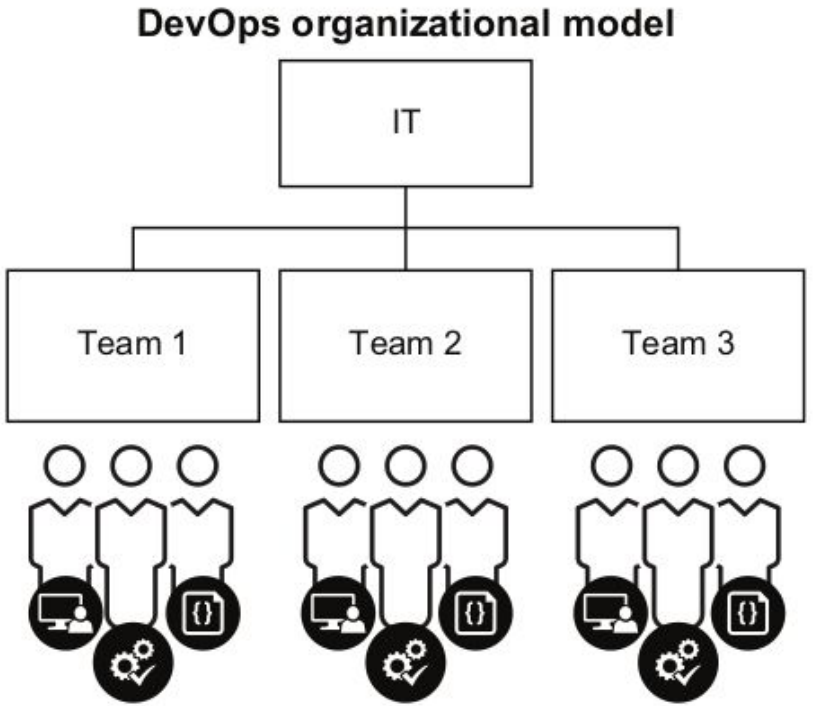

https://www.manning.com/books/gitops-and-kubernetes

# GitOps

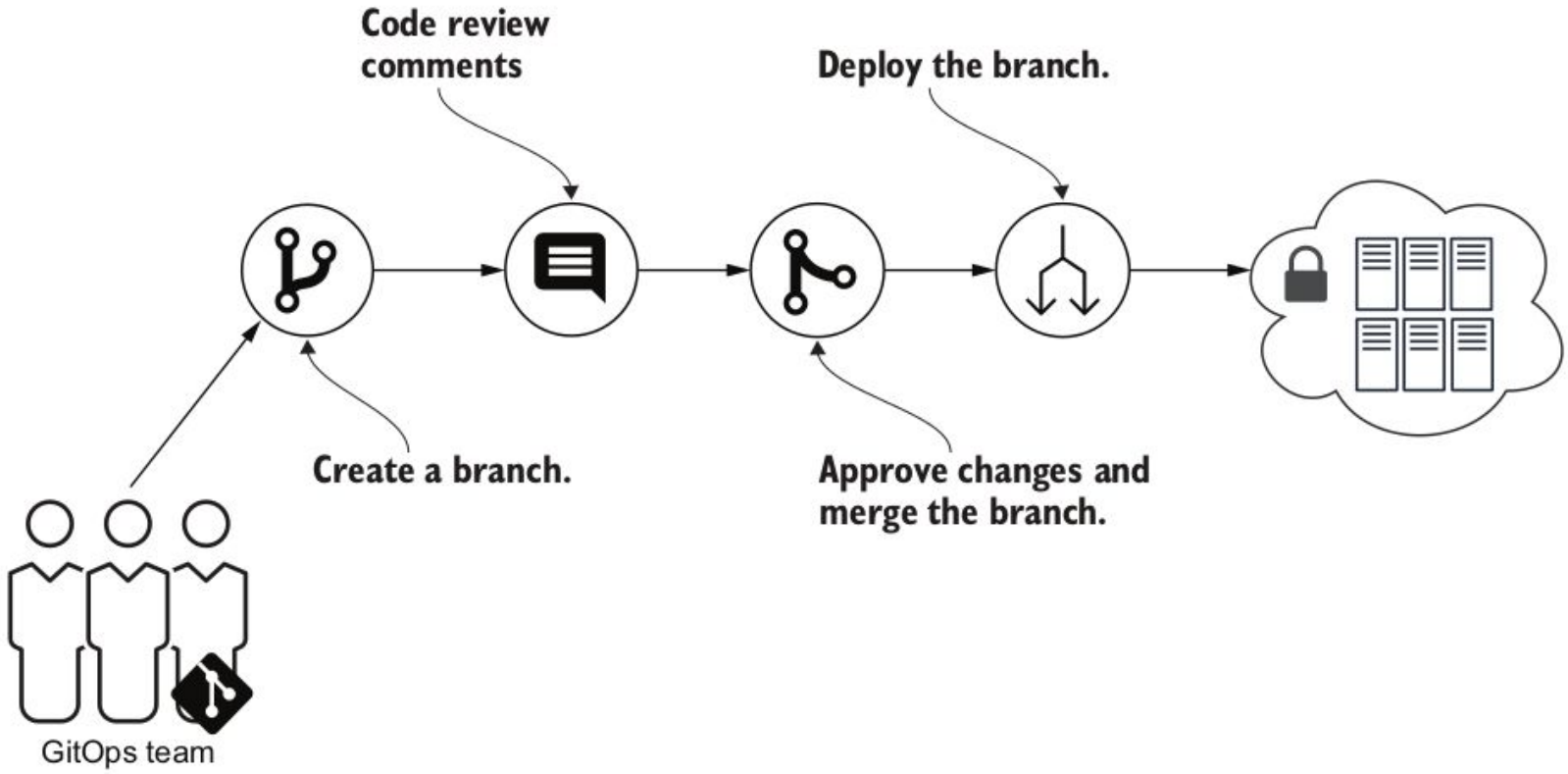
Guarde suas coisas!!!



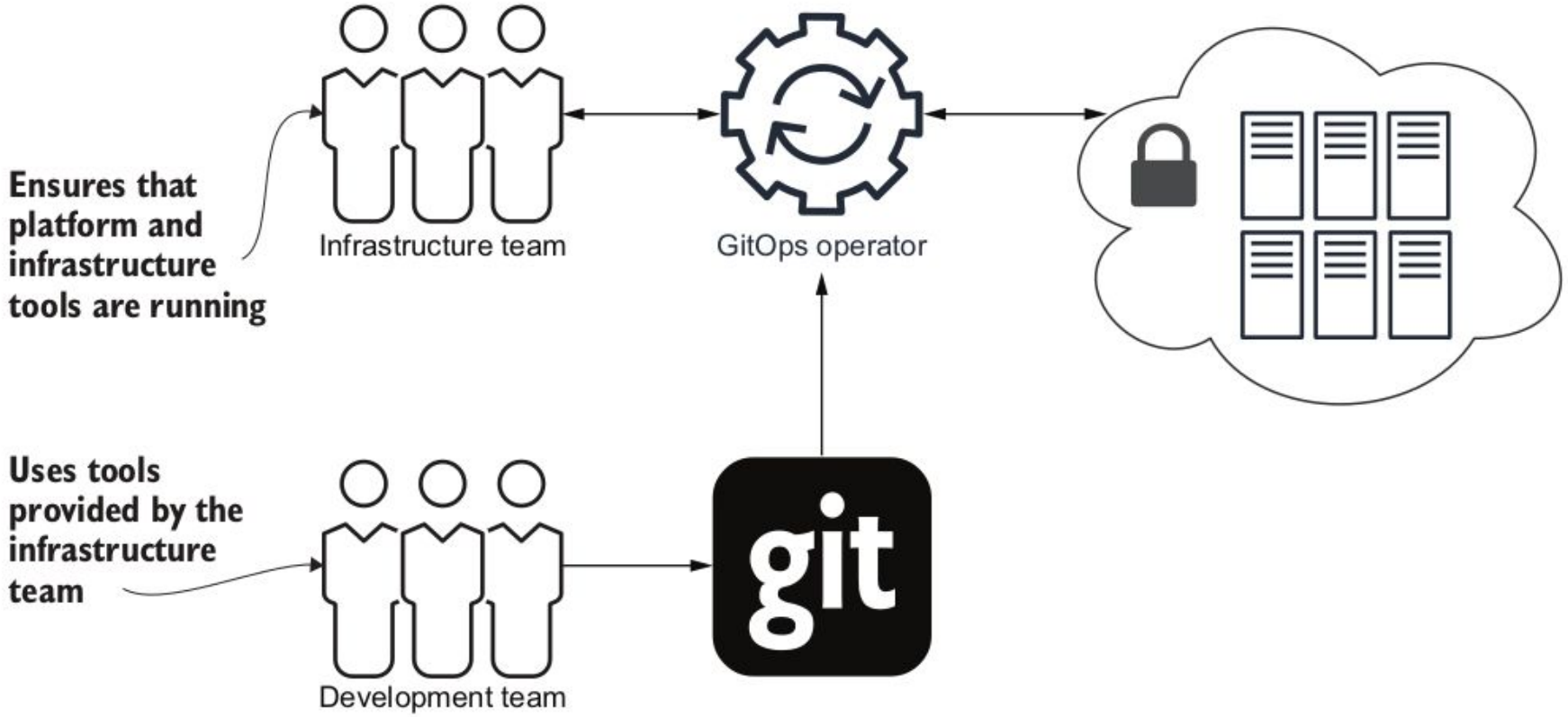



https://www.manning.com/books/gitops-and-kubernetes



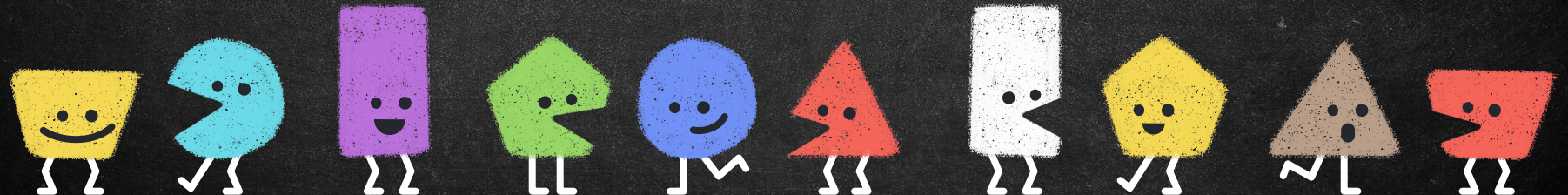

https://www.manning.com/books/gitops-and-kubernetes




https://www.manning.com/books/gitops-and-kubernetes

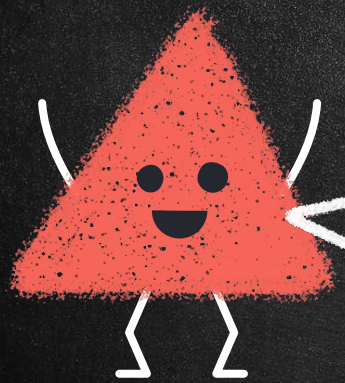
# Ferramentas

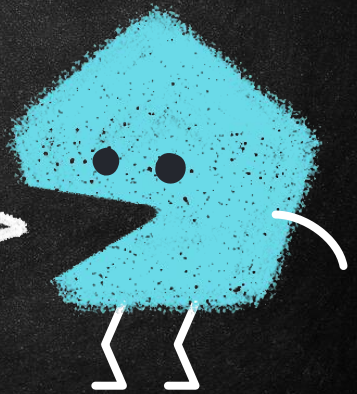
Let’s do it!!



# 1.

# Fluxo de aprovação de contratos





# Fluxo de Aprovação

## Repósitorio de APIs

Criação de um repositório central com artefatos de OpenAPIs

## Git

Ferramenta conhecida no ecossistema de desenvolvedores, é possível armazenar um determinado estado em uma fatia de tempo

## Pull Requests

Conhecido padrão no ambiente de desenvolvedores, intenção de tornar um código parte da companhia, no nosso caso APIs

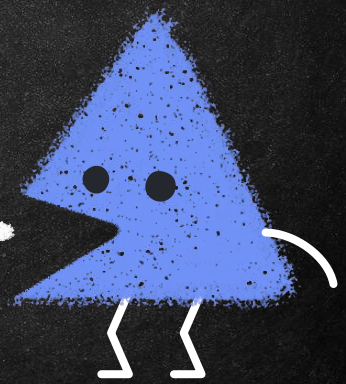
## Validações automáticas ***

Algumas ferramentas podem auxiliar o processo de validação de um artefato OpenAPI

# Spectral

## Spectral, an Open Source JSON/YAML Linter

Improve the quality of your API descriptions, Kubernetes config, GitHub Actions, or any other JSON/YAML data. Get automatic validation & linting warnings, powered by Spectral, when you use Stoplight.



# Spectral

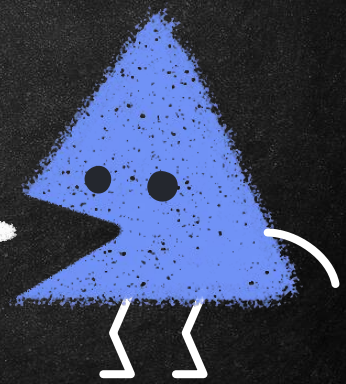
| Type | Line | Message |
|---|---|---|
| ⚠ | 1 | OpenAPI object info `description` must be present and non-… |
| ⚠ | 1 | Info object should contain `contact` object. |
| ⚠ | 10 | Operation `description` must be present and non-empty string. |
| ⚠ | 41 | Operation `description` must be present and non-empty string. |
| ⚠ | 56 | Operation `description` must be present and non-empty string. |
| ⚠ | 83 | Model `description` must be present and non-empty string. |
| ⚠ | 95 | Model `description` must be present and non-empty string. |
| ⚠ | 99 | Model `description` must be present and non-empty string. |

**consistent API descriptions**

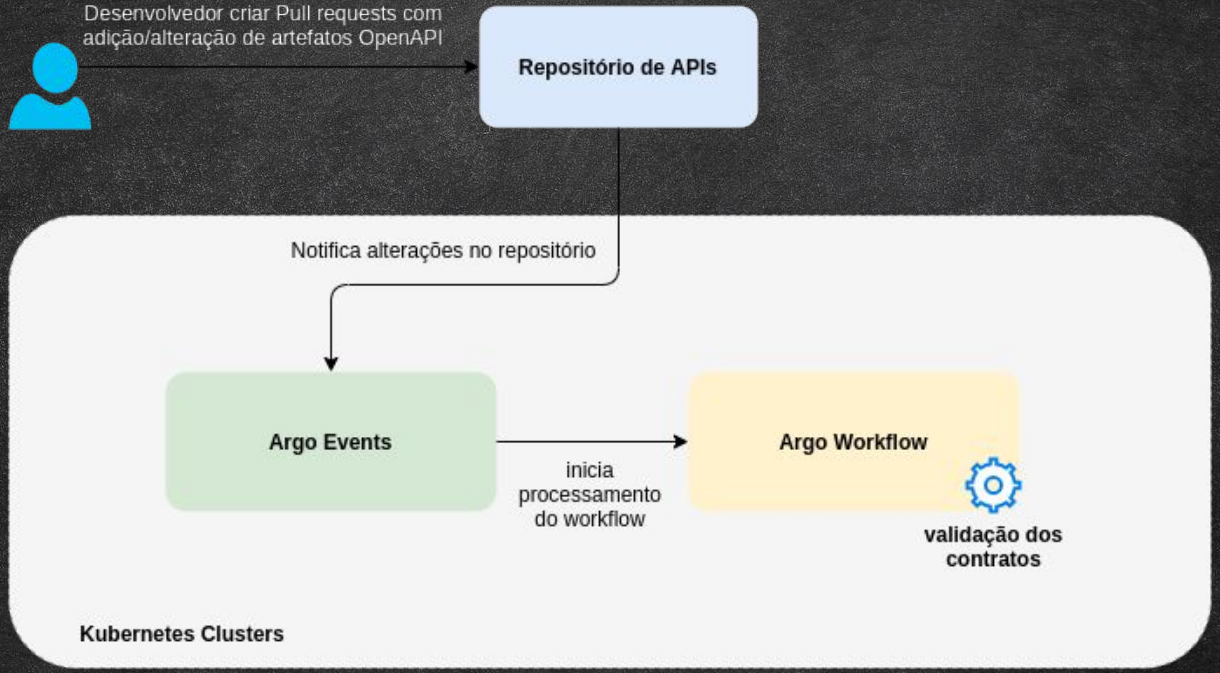
Ask 100 API designers what makes a good API design and you'll get 101 answers, but all most developers really want is consistency. Using a style guide can reduce decision making and improve consistency for all your teams.

You can use the default style guide, extend it, or create one to match your organization's style guide.

Learn More →



# Fluxo de Aprovação



# Prática de Pull Requests

## Descentralização de responsabilidades

Times com responsabilidades no contrato que eles trabalham em geral aumenta a qualidade do artefato produzido

## Compartilhamento da informação

Em geral outras pessoas revisando aumenta a distribuição do conhecimento das APIs da companhia

## Validações automáticas

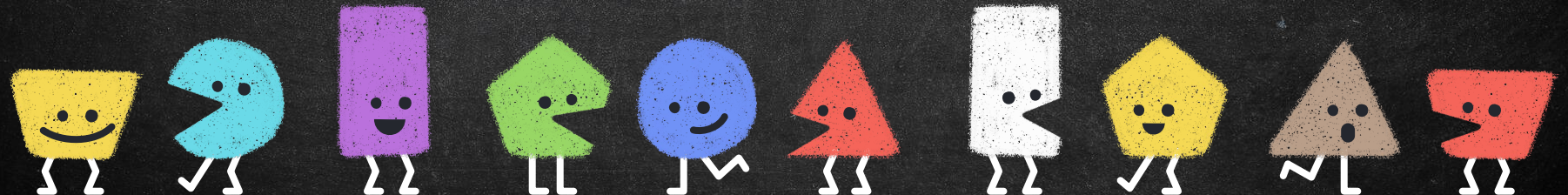
**Procure** utilizar processos automatizados de validação de contrato, em geral a consistência é maior e elimina a possibilidade de erros humanos

## Prática recorrente

Essa prática já é comum aos desenvolvedores não é necessário a adição de uma prática/ferramenta desconhecida

# Catalogo

## Torne suas APIs “acháveis”



A idéia principal aqui é que sua API seja disponibilizada em um catálogo da companhia para que outros times consigam consultar e encontrar serviços já disponibilizados

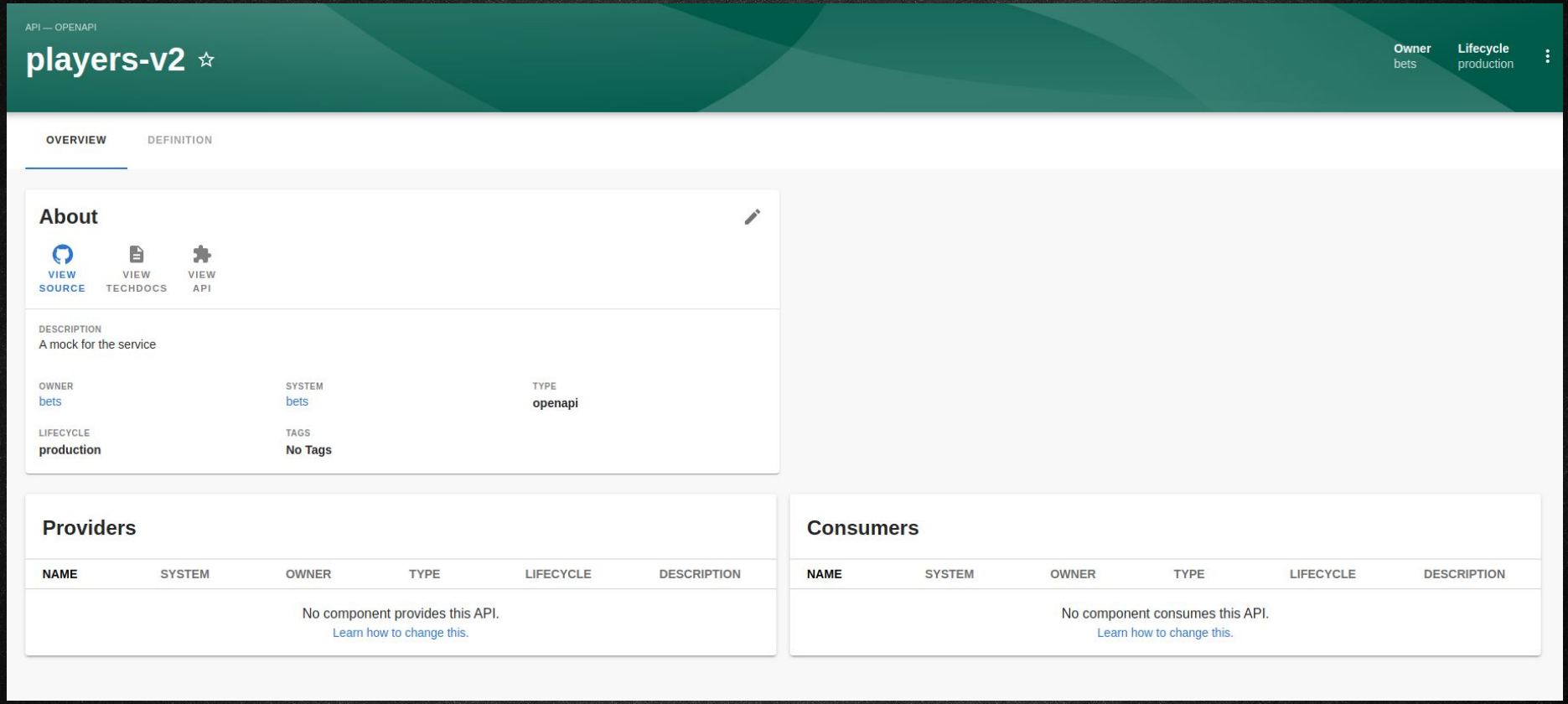
É importante que o serviço tenha algumas informações como estágio de desenvolvimento, time responsável e definição, no nosso caso OpenAPI

# Dica

Sempre que possível "incorpore" o fluxo de atualização de documentação no seu fluxo de pipeline de CI.

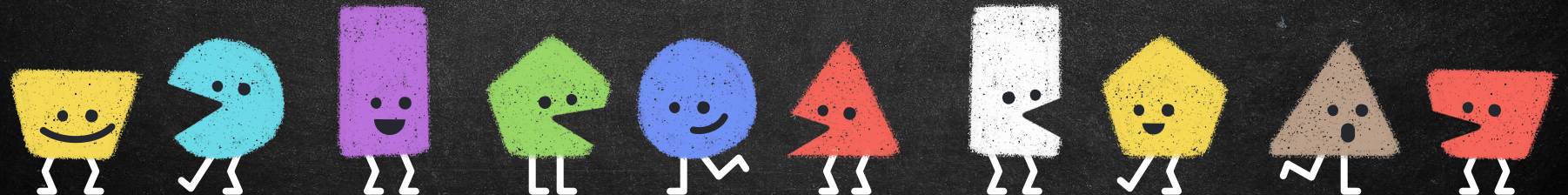
https://www.youtube.com/watch?v=aIURa0cR5D8

# Backstage


API — OPENAPI
players-v2
Owner
bets
Lifecycle
production
OVERVIEW
DEFINITION
About
VIEW SOURCE
VIEW TECHDOCS
VIEW API
DESCRIPTION
A mock for the service
OWNER
bets
SYSTEM
bets
TYPE
openapi
LIFECYCLE
production
TAGS
No Tags
Providers
NAME
SYSTEM
OWNER
TYPE
LIFECYCLE
DESCRIPTION
No component provides this API.
Learn how to change this.
Consumers
NAME
SYSTEM
OWNER
TYPE
LIFECYCLE
DESCRIPTION
No component consumes this API.
Learn how to change this.

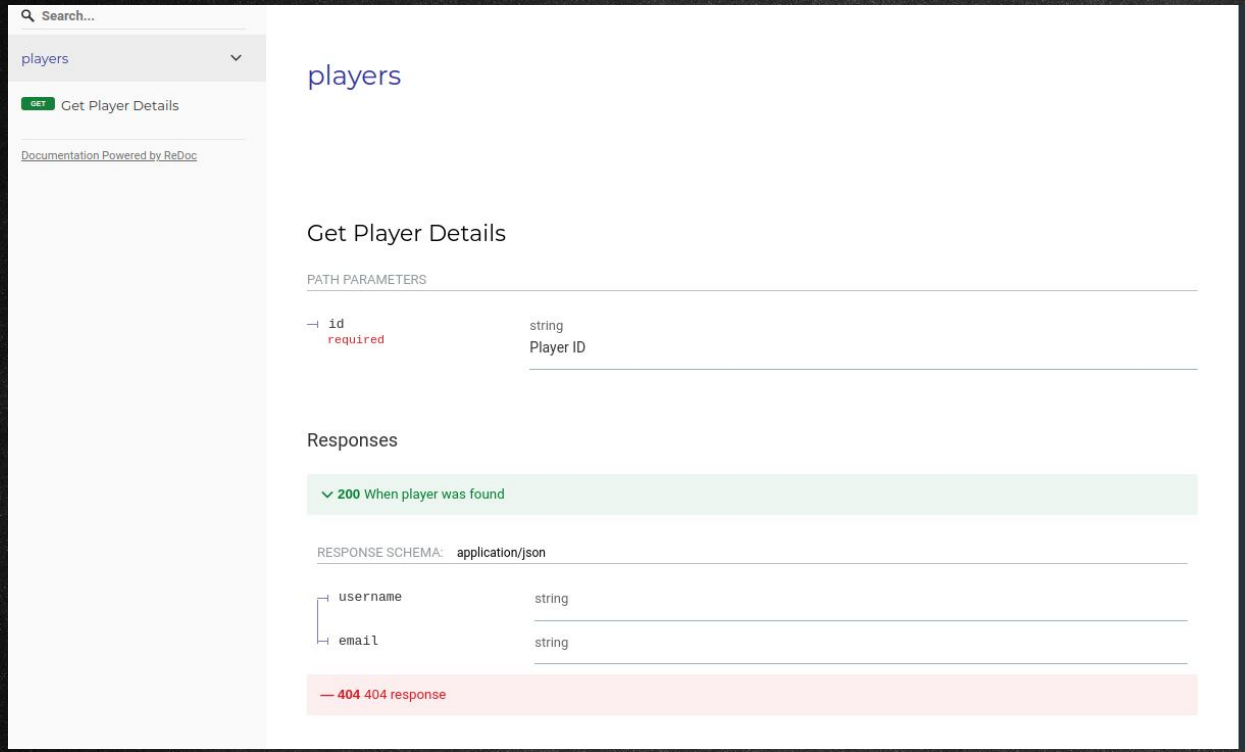
# Documentação

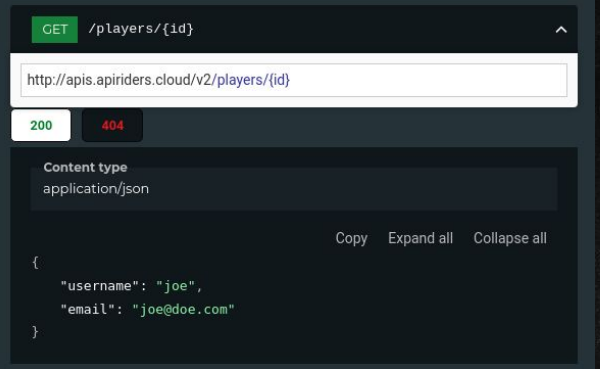
## OpenAPI com “esteroídes”



O OpenAPI define as operações de uma API, porém precisamos de uma ferramenta que nos ajude a deixar a documentação mais fácil de ser lida

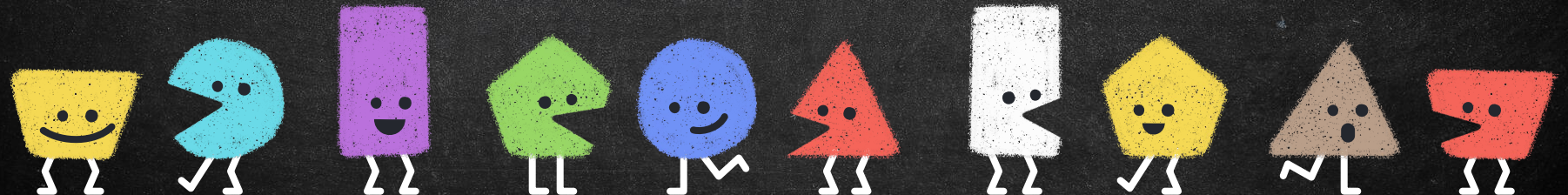

ReDoc


Search...
players
GET Get Player Details
Documentation Powered by ReDoc
players
Get Player Details
PATH PARAMETERS
id
required
string
Player ID
Responses
200 When player was found
RESPONSE SCHEMA: application/json
username
string
email
string
404 404 response


GET /players/{id}
http://apis.apiriders.cloud/v2/players/{id}
200
404
Content type
application/json
Copy
Expand all
Collapse all
{
"username": "joe",
"email": "joe@doe.com"
}

# Mocking

Comece a “enxergar” a implementação da sua API



Durante a fase de prototipação da sua API a criação de mocks pode permitir que você disponibilize suas APIs para seus consumidores de maneira fictícia e criar um design colaborativo permitindo que outras pessoas possam dar "feedbacks" na construção

O OpenAPI tem uma seção de examples que pode ser utilizado para criação/geração de mocks automatizados, além de facilitar o consumo da API deixando ela mais amigável


ap:rator

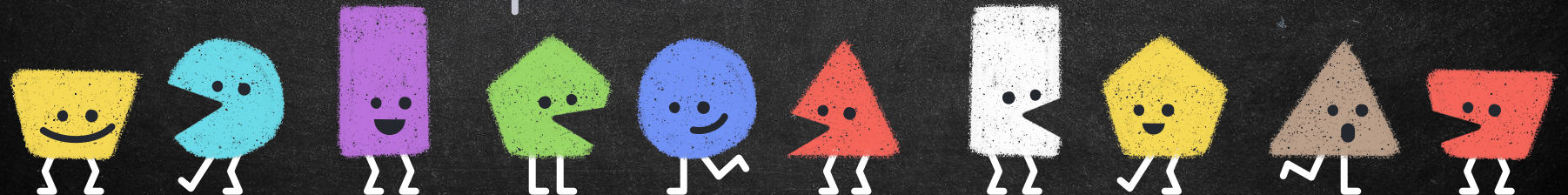
77 lines (77 sloc) | 1.97 KB

Raw Blame

```yaml
apiVersion: apirator.io/v1alpha1
kind: APIMock
metadata:
  creationTimestamp: null
  name: players-v3
  namespace: mocks
spec:
  definition: |
    openapi: 3.0.0
    x-kong-name: players
    info:
      x-kubernetes-ingress-metadata:
        name: players-v3
      title: Bets - Players API
      version: 2.0.0
      contact:
        name: Apirator Dev Team
        email: apirator@apirator.io
      license:
        name: MIT License
        url: 'https://opensource.org/licenses/MIT'
    servers:
      - url: http://apis.apiriders.cloud/v3
    paths:
      '/players/{id}':
        description: Find players data
        get:
          tags:
            - players
          parameters:
            -   name: id
                description: Player ID
                schema:
                  type: string
                in: path
                required: true
          responses:
            '200':
              content:
                application/json:
                  schema:
```

# Testing

## Validando o comportamento da API

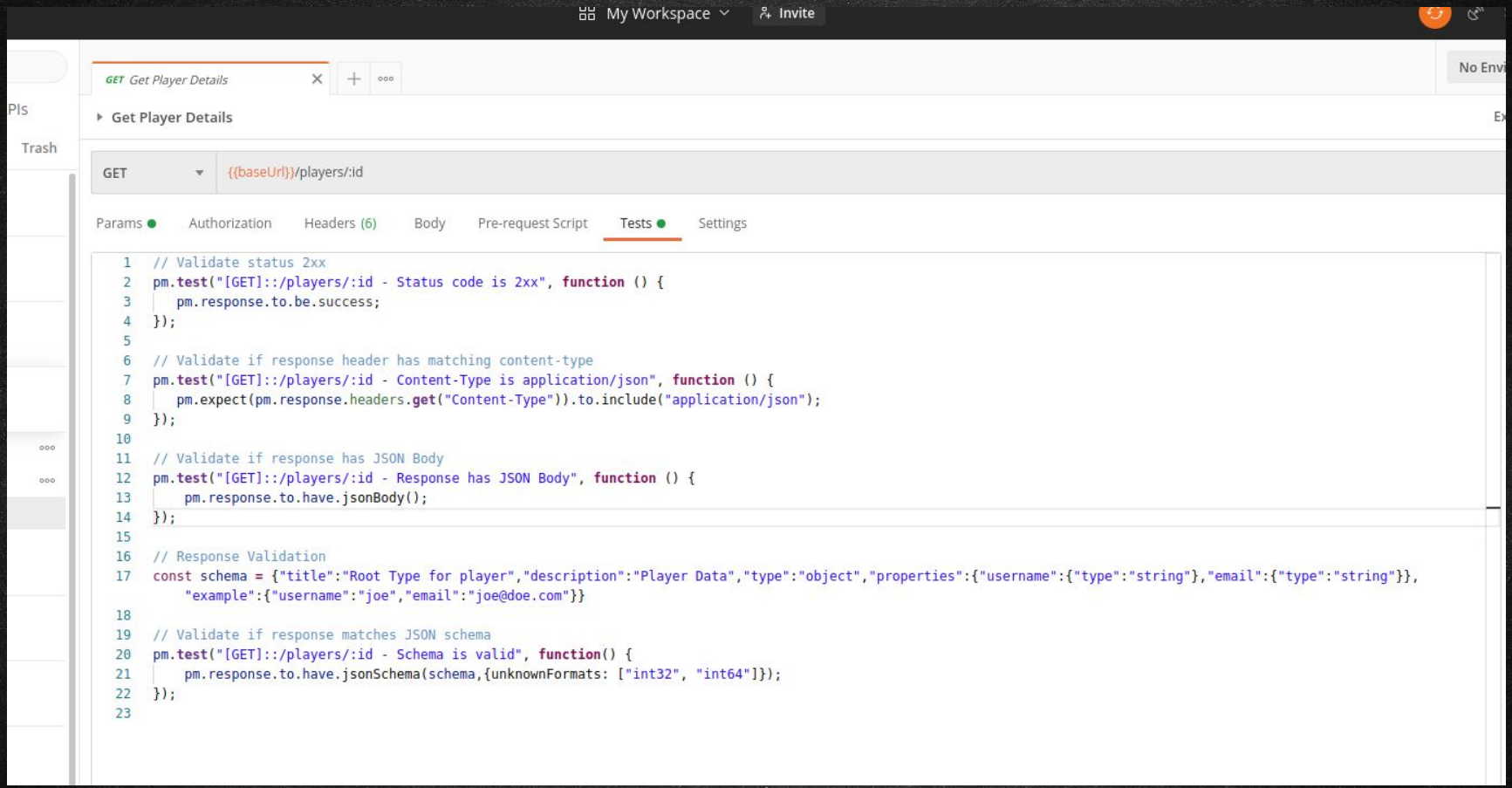


A partir de um OpenAPI bem documentado, com a seção de examples implementada é possível gerar coleções de postman validando o comportamento da sua API, isso pode ser um bom início para testes de contrato

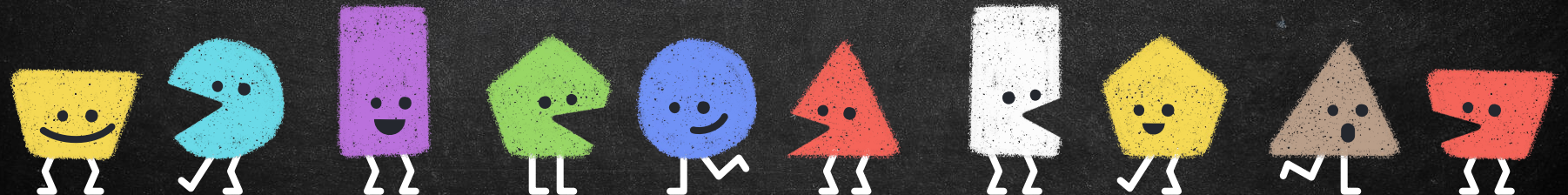
# Portman CLI 1.0 is here

npm @apideck/portman

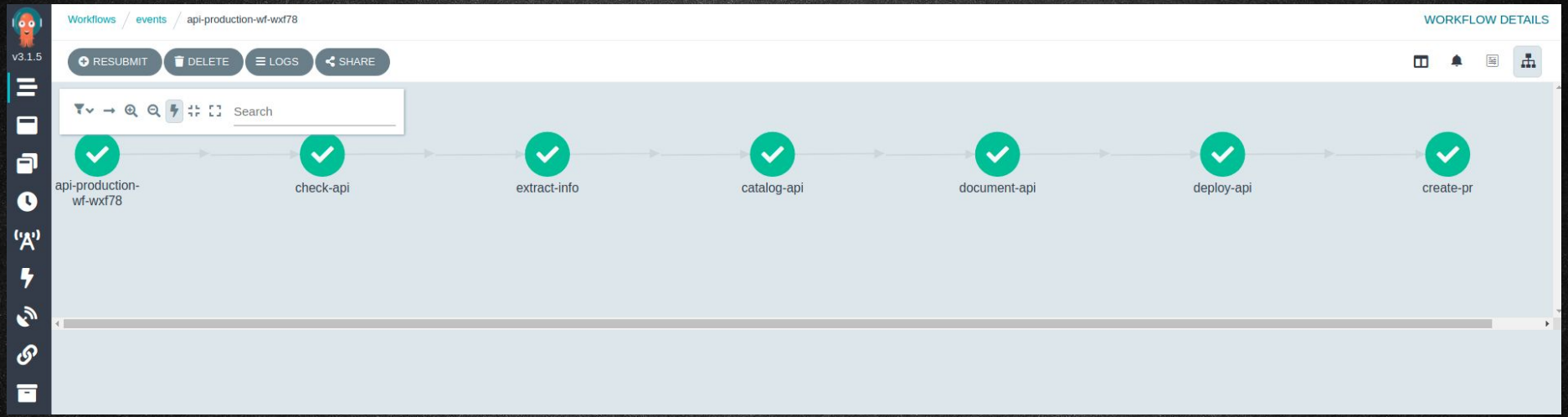
```
npx @apideck/portman -l your-openapi-file.yaml

Usage: -u <url> -l <local> -b <baseUrl> -t <includeTests>

Options:
      --help                  Show help
      --version               Show version number
  -u, --url                   URL of OAS to port to Postma
  -l, --local                 Use local OAS to port to Pos
  -b, --baseUrl               Override spec baseUrl to use
  -o, --output                Write the Postman collection
  -n, --runNewman             Run Newman on newly created
  -d, --newmanIterationData   Iteration data to run Newman
  --localPostman              Use local Postman collection
  --syncPostman               Upload generated collection
  -p, --postmanUid            Collection UID to upload wit
  -t, --includeTests          Inject Portman test suite (d
  -c, --portmanConfigFile     Path to Portman settings con
  -s, --postmanConfigFile     Path to openapi-to-postman c
  -s, --filterFile            Path to openapi-format confi
  --envFile                   Path to the .env file to inj
  --cliOptionsFile            Path to Portman CLI options
  --init                      Configure Portman CLI option
```


My Workspace
Invite
GET Get Player Details
Get Player Details
Trash
GET
{{baseUrl}}/players/:id
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
// Validate status 2xx
pm.test("[GET]::/players/:id - Status code is 2xx", function () {
pm.response.to.be.success;
});
// Validate if response header has matching content-type
pm.test("[GET]::/players/:id - Content-Type is application/json", function () {
pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
// Validate if response has JSON Body
pm.test("[GET]::/players/:id - Response has JSON Body", function () {
pm.response.to.have.jsonBody();
});
// Response Validation
const schema = {"title":"Root Type for player","description":"Player Data","type":"object","properties":{"username":{"type":"string"},"email":{"type":"string"}},
"example":{"username":"joe","email":"joe@doe.com"}}
// Validate if response matches JSON schema
pm.test("[GET]::/players/:id - Schema is valid", function() {
pm.response.to.have.jsonSchema(schema,{unknownFormats: ["int32", "int64"]});
});

# Pipelines

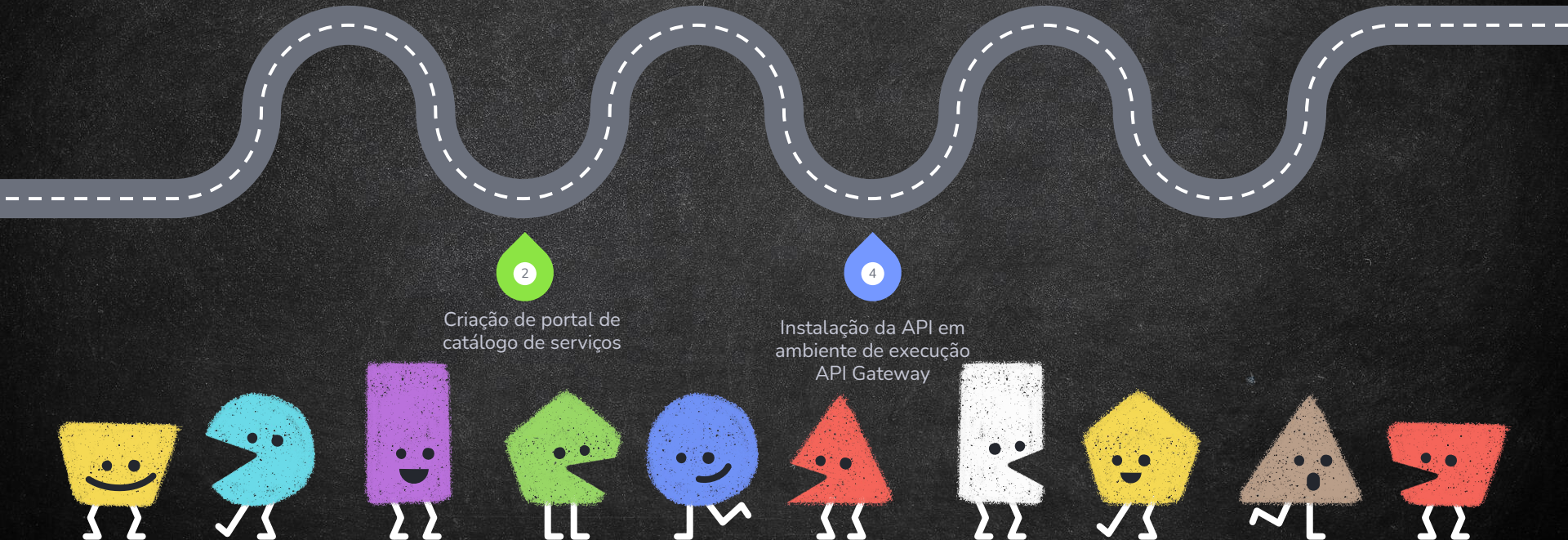



EVENTS


v3.1.5
Workflows
events
api-production-wf-wxf78
WORKFLOW DETAILS
RESUBMIT
DELETE
LOGS
SHARE
Search
api-production-wf-wxf78
check-api
extract-info
catalog-api
document-api
deploy-api
create-pr

# Production Pipeline

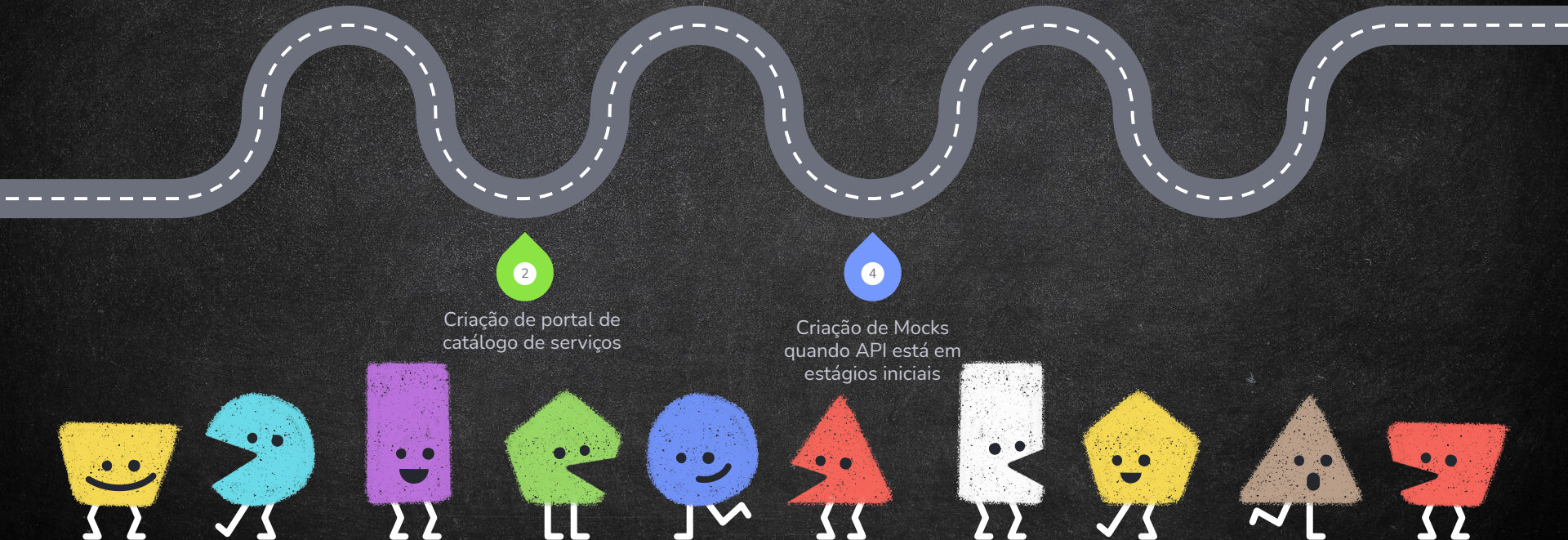
1. Fluxo de aprovação de contratos
2. Criação de portal de catálogo de serviços
3. Criação de documentação da API em formato intuitivo
4. Instalação da API em ambiente de execução API Gateway
5. Criaçao de Pull Requests

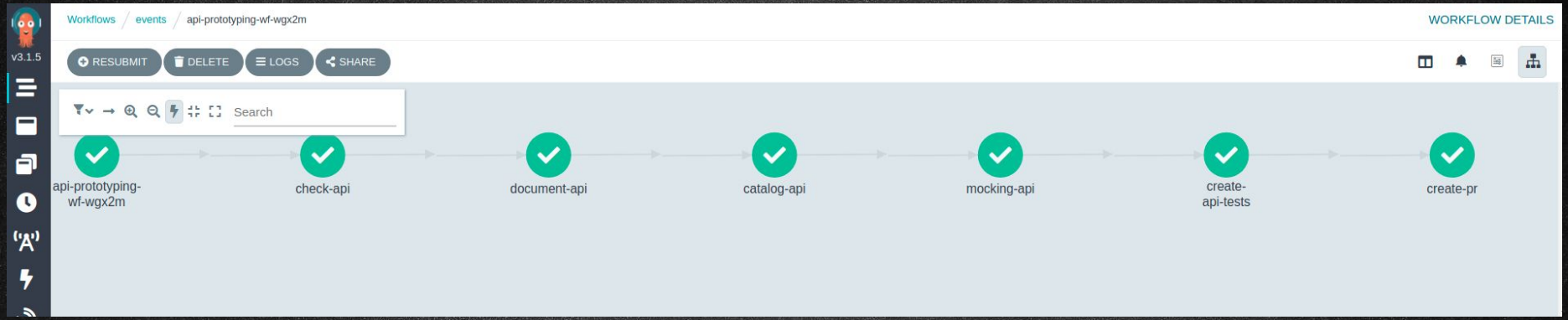



LIVE DEMO

# Prototyping Pipeline

1. Fluxo de aprovação de contratos
2. Criação de portal de catálogo de serviços
3. Criação de documentação da API em formato intuitivo
4. Criação de Mocks quando API está em estágios iniciais
5. Criação de artefatos de tests (postman)

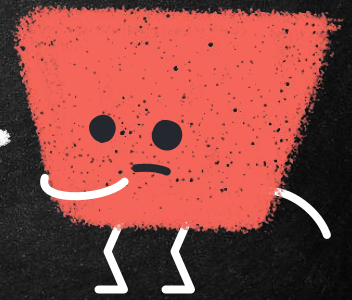



v3.1.5
Workflows
events
api-prototyping-wf-wgx2m
WORKFLOW DETAILS
RESUBMIT
DELETE
LOGS
SHARE
Search
api-prototyping-wf-wgx2m
check-api
document-api
catalog-api
mocking-api
create-api-tests
create-pr


LIVE DEMO

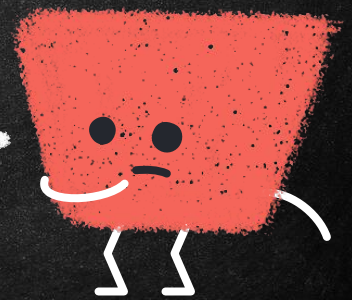
# Próximos passos

- Melhores legibilidade dos logs do WF
- Criar newman-operator para validar os testes criados no flow Prototyping (in progress)
- Adicionar trigger no pipeline de prototyping



# GitHub

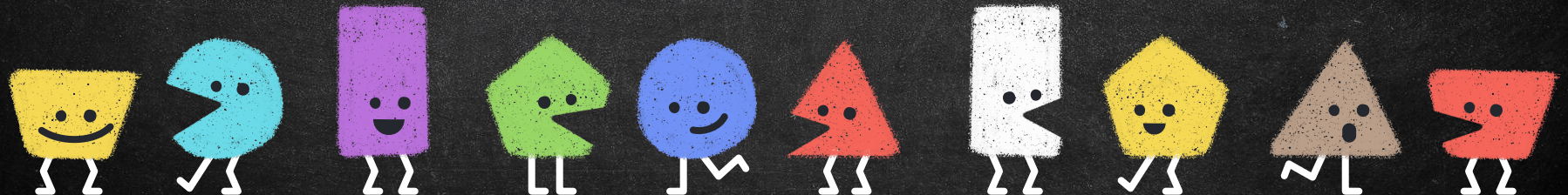
- https://github.com/apirator/workflow-api-docs
- https://github.com/apirator/workflow-manifests
- https://github.com/apirator/workflow-definitions
- https://github.com/apirator/workflow-apps

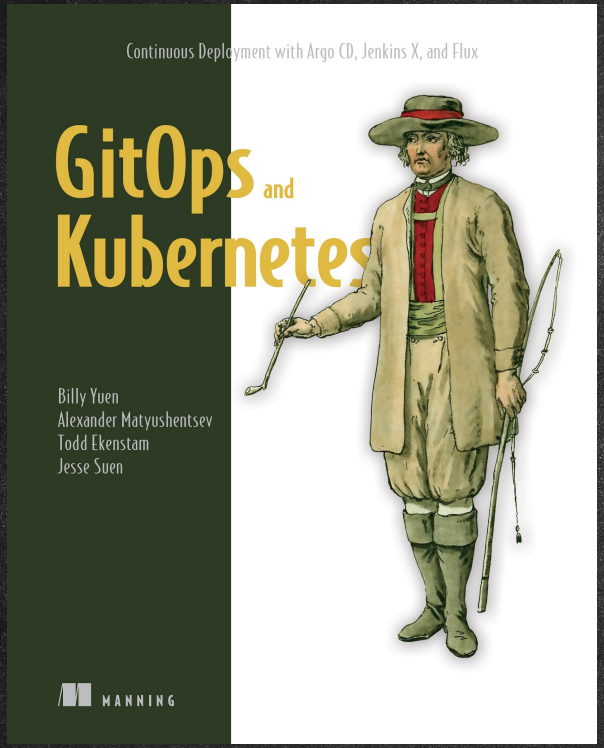


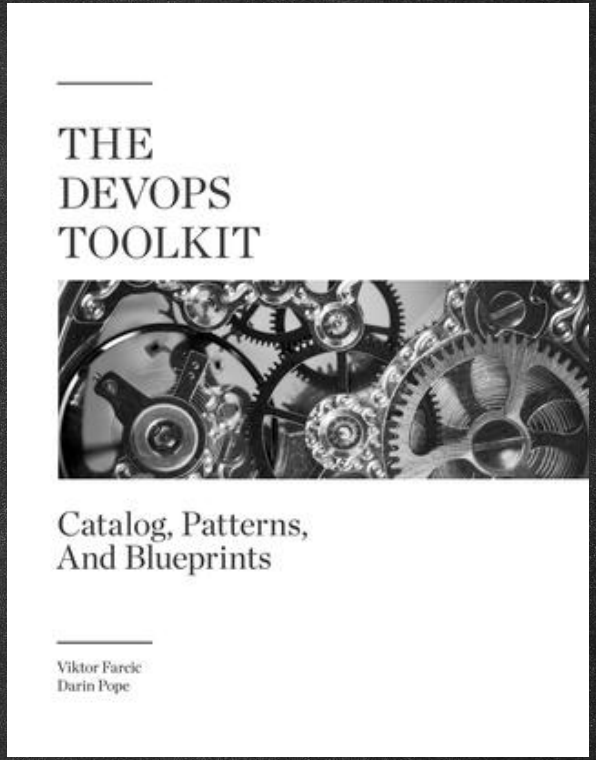
# Contatos

**Cláudio de Oliveira**
Tech Lead
@claudioed

**Marcelo Marinho**
Especialista em Desenvolvimento
https://www.linkedin.com/in/marcelo-marinho-4b728928/




Continuous Deployment with Argo CD, Jenkins X, and Flux
GitOps and Kubernetes
Billy Yuen
Alexander Matyushentsev
Todd Ekenstam
Jesse Suen
MANNING


THE DEVOPS TOOLKIT
Catalog, Patterns, And Blueprints
Viktor Farcic
Darin Pope