FIAP MBA+

MACHINE LEARNING ENGINEERING

MBA EM DATA SCIENCE & IA

APRESENTAÇÃO

CREDIT SCORING

A QuantumFinance realizou uma parceria com algumas empresas clientes para fornecer o score de crédito de seus clientes optantes para permitir melhores condições de pagamento.
Para que o modelo seja mais simples e diferente do modelo de score principal do banco, será necessário treinar esse modelo com os dados das transações mais recentes dos clientes.

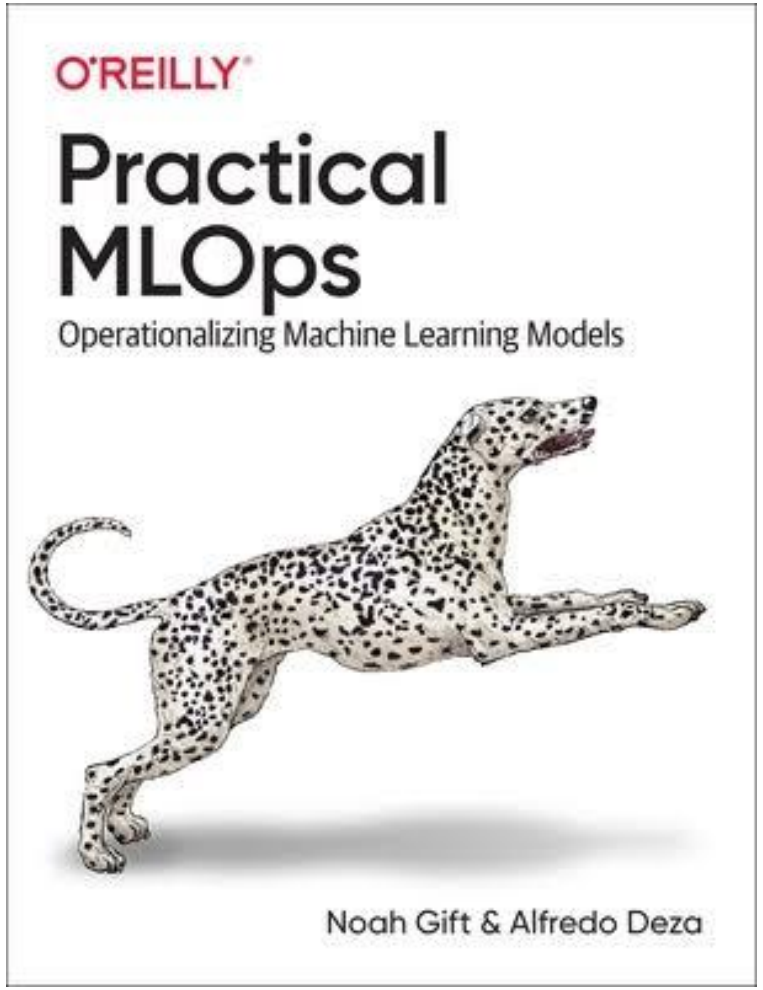
Para permitir governança interna e integração com outros sistemas, esta solução precisa incluir:

- Template de repositório para organização dos arquivos (dataset, notebook, modelo, etc.)
- Rastreamento dos experimentos do treinamento do modelo
- Versionamento do modelo
- Disponibilização de endpoint de API seguro com autenticação e throttling
- Documentação da API

Afim de validar e tornar como um exemplo de implementação para os parceiros integre uma aplicação modelo no Streamlit com a API disponibilizada.

Dataset https://www.kaggle.com/datasets/parisrohan/credit-score-classification

BIBLIOGRAFIA **BÁSICA**

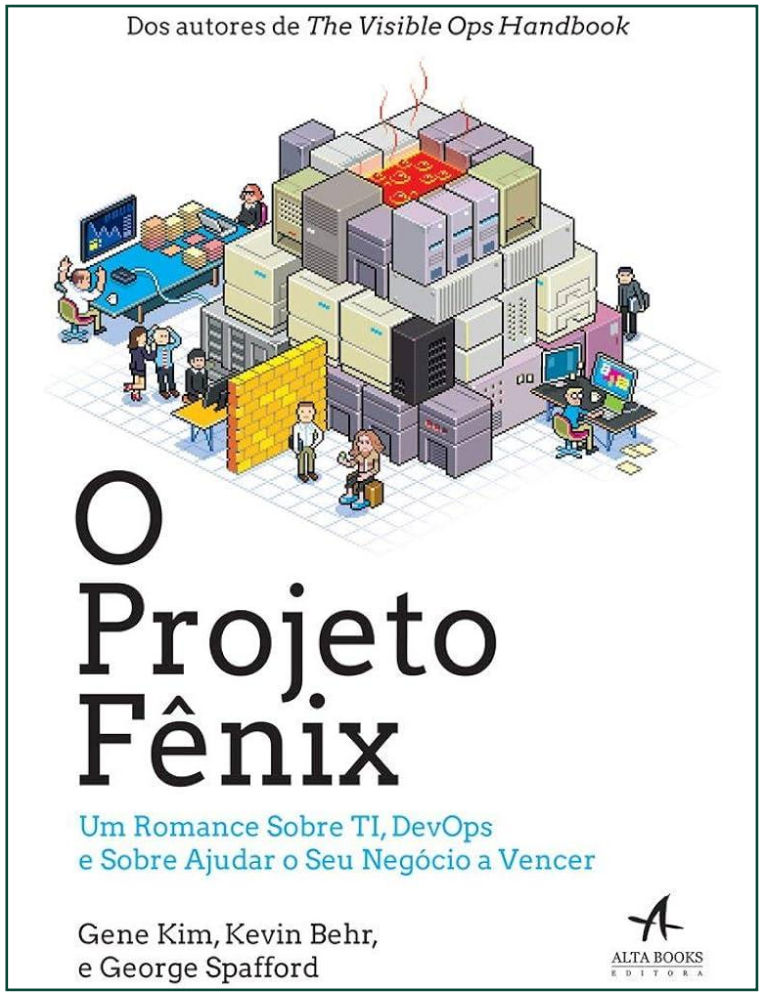


Practical MLOps

Noah Gift & Alfredo Deza

https://www.amazon.com/Practical-MLOps-Operationalizing-Machine-Learning/dp/1098103017

BIBLIOGRAFIA **COMPLEMENTAR**



O Projefo Fênix

Gene Kim, Kevin Behr e George Spafford

https://www.amazon.com.br/projeto-f%C3%AAnix-Gene-Kim/dp/8550801895

PLANO DE **AULA**

FIAP MBA+

AULA 1: INTRODUÇÃO

Introdução a MLOps

DevOps & CD4ML

Práticas de engenharia de Software

Desafios de projeto de machine learning

Testes de unidade e análise de código estático (linting)

AULA 2: VERSIONAMENTO E CONTROLE

Gerenciamento de versão e controle de código-fonte

Git e Gitflow para projetos de machine learning

Scaffolding de projeto de machine learning

AULA 3: DEPLOYMENT DE MODELOS

Estratégias de implantação de modelo em produção

Ciclo-de-vida de modelos e experimentos

Fluxos de trabalho de machine learning

Contêinerização de aplicações

Desenvolvimento de APIs em cloud

AULA 4: AUTOMAÇÃO DE PIPELINES

Automação de fluxos de trabalho em machine learning

Automação de treinamento e implantação de modelos

Pipelines de entrega contínua

Monitoramento de modelos

Desvios de modelos de machine learning

PROJETO DA DISCIPLINA

FIAP MBA+

Share

Machine Learning Engineering

Predição de Preço de Laptop

Este modelo é capaz de prever o preço de um laptop dada algumas características.

A aplicação é para ser utilizada em uma loja eletrônica que avalia laptops usados como parte do pagamento de um novo, por tal razão a avaliação não é tão exaustiva e se baseia em caracteríticas comuns, como marca, processador, memória etc. sem nenhuma outra avaliação visual, pelo menos por enquanto.

Características do laptop

Qual é a marca?

Asus

Possui touchscreen (tela sensível ao toque)?

- Não
- Sim

Qual a marca do processor?

- AMD
- Intel
- M1

Qual é o nome do processador?

Core i3

Qual a arquitetura do sistema operacional?

Manage app

Laptop Pricing App

Solução de avaliação de notebooks baseado em características de configuração como memória, processador em marca.

Baseado no dataset Laptop Prices (https://www.kaggle.com/datasets/anubhavgoyal10/laptop-prices-dataset)

LAB

O programa do curso utilizará as últimas versões do Python disponível no Google Colab.

Paralelamente construiremos pequenas aplicações utilizando o Streamlit para dar mais interatividade com as plataformas cloud e que poderão servir de exemplo para aplicações reais.

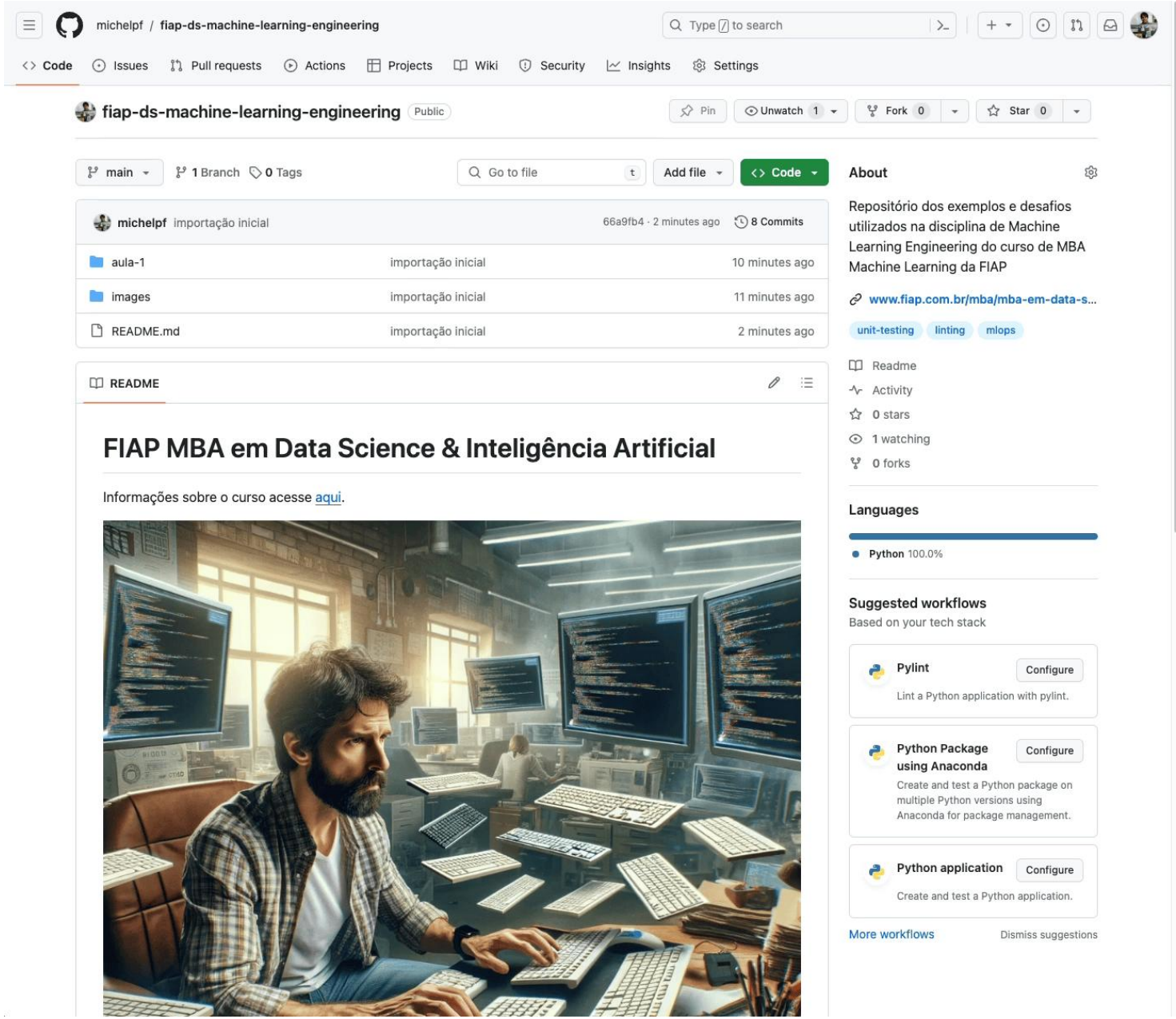
Consulte as documentações do Google Colab e Streamlit para se familiarizar com as plataformas.

LAB

Como alternativa, podemos utilizar o Codespaces (clique nos links do Readme).

Por ser limitado na conta gratuita, utilize com cautela para não perder os créditos com o uso mais intenso.

Considere o uso do codespaces como pontual, como por exemplo para verificar os desafios.

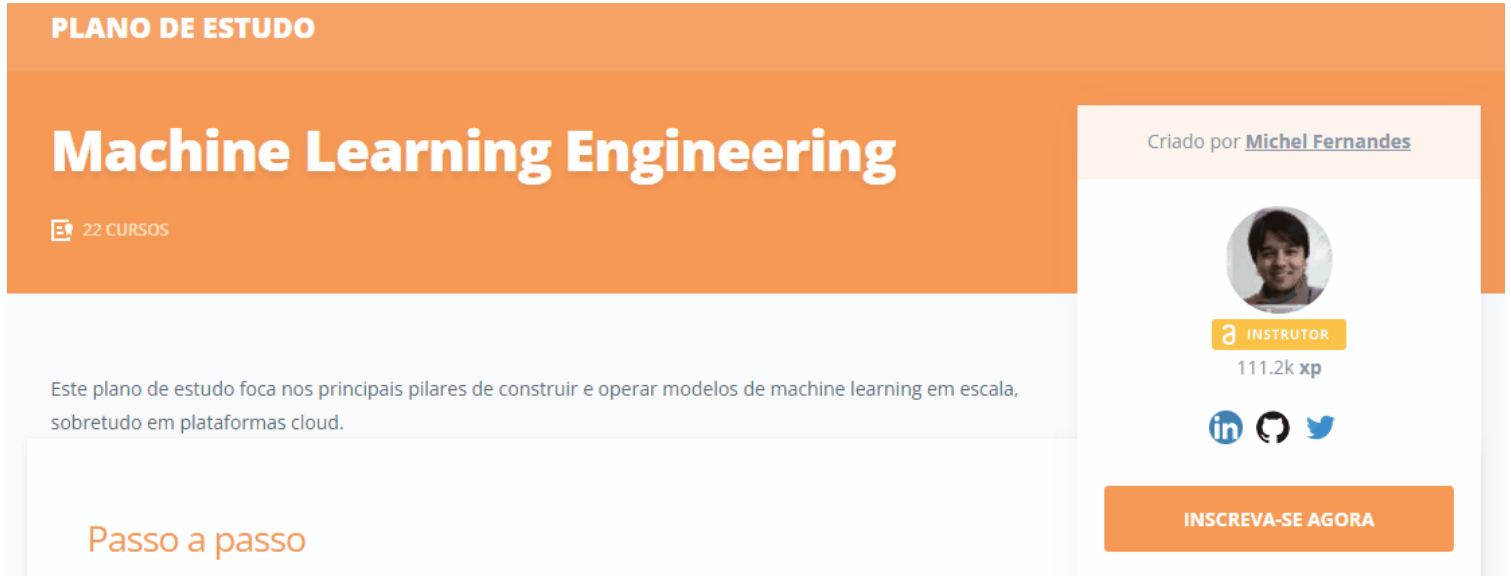


RECURSOS **ÚTEIS**

Os materiais de aula se encontram em:

- **PORTAL DO ALUNO**: apresentações, artigos e documentos de uso geral
- **GITHUB** dos projetos utilizados pelo professor:
  https://github.com/michelpf/fiap-ds-machine-learning-engineering

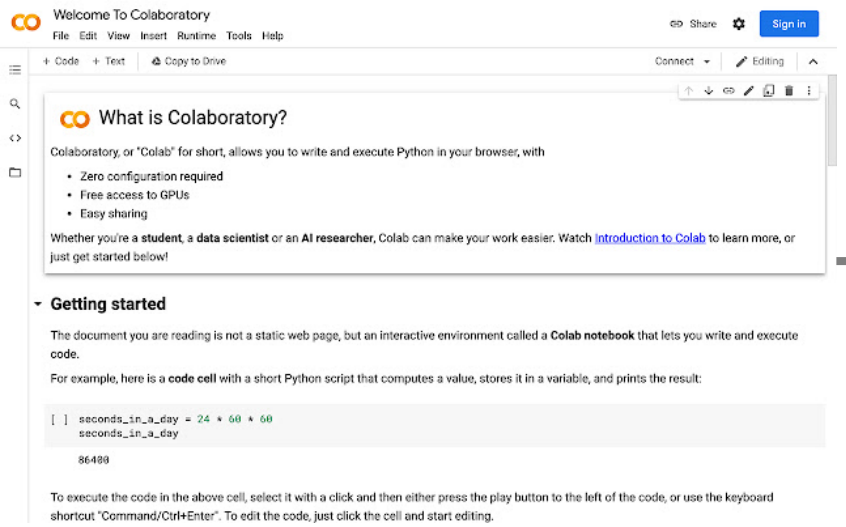
Plano de estudo na Alura para aperfeiçoamento: https://cursos.alura.com.br/machine-learning-engineering-michelpf-1705269316633-p710416

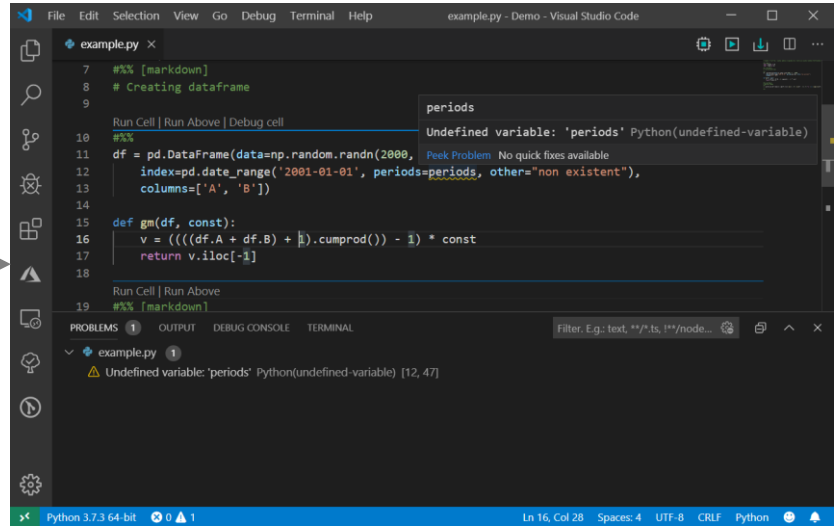


INTRODUÇÃO A MLOPS

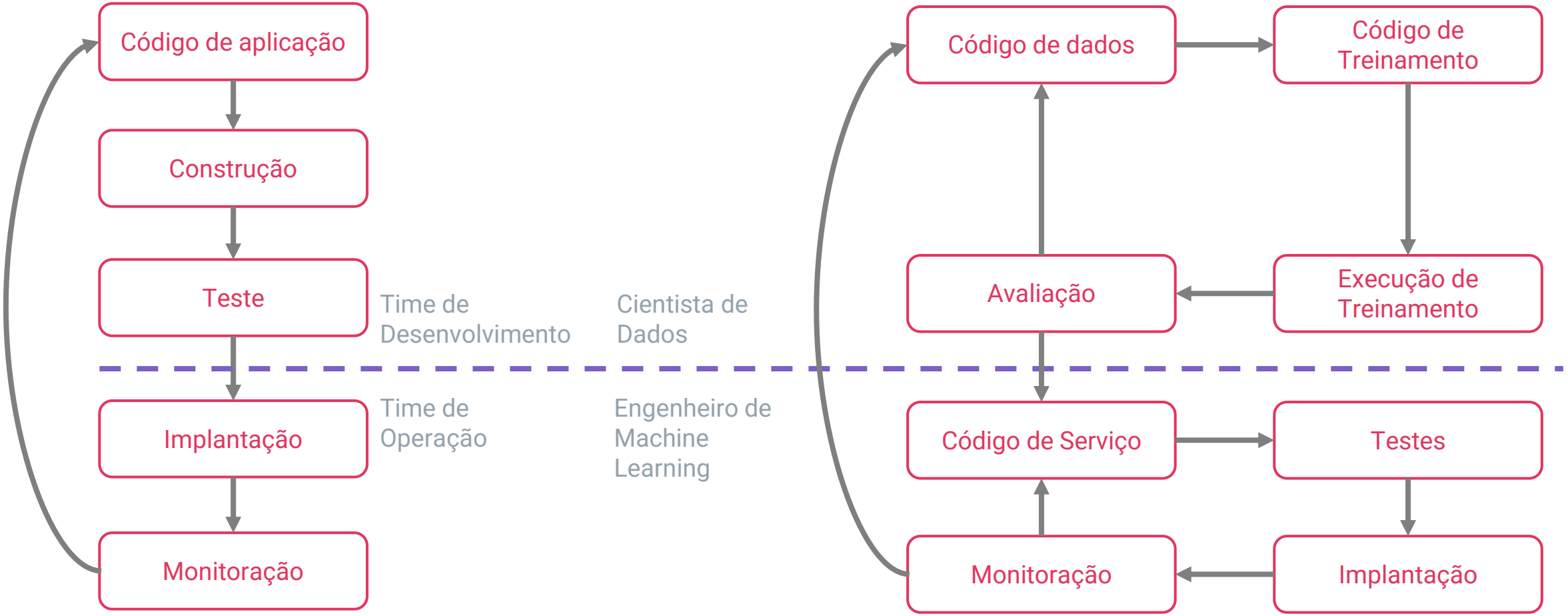
MACHINE LEARNING OPS

Colocar modelos de aprendizado de máquina em produção exige uma mudança de paradigma que começa no próprio ambiente de desenvolvimento. Apesar dos experimentos continuarem sendo uma etapa essencial, passaremos a utilizar o VS Code como ambiente principal, já que ele oferece um conjunto mais abrangente de recursos em comparação ao uso exclusivo de Notebooks. Entre esses recursos, destacam-se ferramentas que facilitam a construção de pipelines e a definição de processos mais estruturados, repetíveis e versionáveis, que podem ser facilmente integrados e implantados como serviços.





SOFTWARE ENGINEERING E MACHINE LEARNING OPS



Engenharia de Software há 20 anos

Machine Learning Atualmente

https://www.youtube.com/watch?v=N96yiqUFwEo&t=1109s

A AURORA DO **DEVOPS**

DevOps representa a união de pessoas, processos e produtos com o objetivo de habilitar a entrega contínua de valor aos clientes finais. Com os times atuando de forma colaborativa e adotando um novo mindset, surgiram práticas que aproveitam o melhor de cada área, como testes automatizados, implantações contínuas e infraestrutura como código, entre outras. Essas práticas viabilizam ciclos de entrega mais ágeis, confiáveis e sustentáveis.

https://www.youtube.com/watch?v=N96yiqUFwEo&t=1109s

YOU BUILD IT, YOU RUN IT

Werner Vogels, CTO AWS

https://aws.amazon.com/pt/blogs/aws/acm_queue_inter

ANTES DO **DEVOPS**

FIAP MBA+

Os times apresentavam diversas dificuldades, especialmente nos seguintes domínios:

COMUNICAÇÃO E COLABORAÇÃO

Divisão significativa entre as equipes de desenvolvimento e infraestrutura (operações).

ENTREGA E IMPLANTAÇÃO CONTÍNUA

Ciclos de desenvolvimento e liberação lentos, com pouca automação nos processos de entrega.

AGILIDADE E RESILIÊNCIA

Ambientes que não conseguem se adaptar rapidamente as mudanças, resultando em sistemas menos flexíveis e resistente a falhas.

GESTÃO DE INFRAESTRUTURA

Configuração manual e propensa a erros de ambientes de infraestrutura.

MONITORAMENTO E FEEDBACK RÁPIDO

Dificuldades em identificar e corrigir problemas de desempenho em tempo real.

CULTURA DE MELHORIA CONTÍNUA

Falta de cultura que promova a aprendizagem contínua e aprimoramento constante.

COM O **DEVOPS**

FIAP MBA+

DevOps, o começo da unificação dos times de desenvolvimento e operações.

COMUNICAÇÃO E COLABORAÇÃO

Comunicação efetiva, colaboração e quebra de silos entre as equipes.

ENTREGA E IMPLANTAÇÃO CONTÍNUA

CI/CD é uma prática central do DevOps, permitindo automação de build, testes e implantação, resultando em ciclos de desenvolvimento mais curtos.

AGILIDADE E RESILIÊNCIA

Automação e a abordagem orientada por práticas ágeis ajudam a tornar os processos mais ágeis, permitindo adaptações rápidas às mudanças.

GESTÃO DE INFRAESTRUTURA

Infraestrutura como código permitindo automação para criar e modificar ambiente, diminuindo erros humanos.

MONITORAMENTO E FEEDBACK RÁPIDO

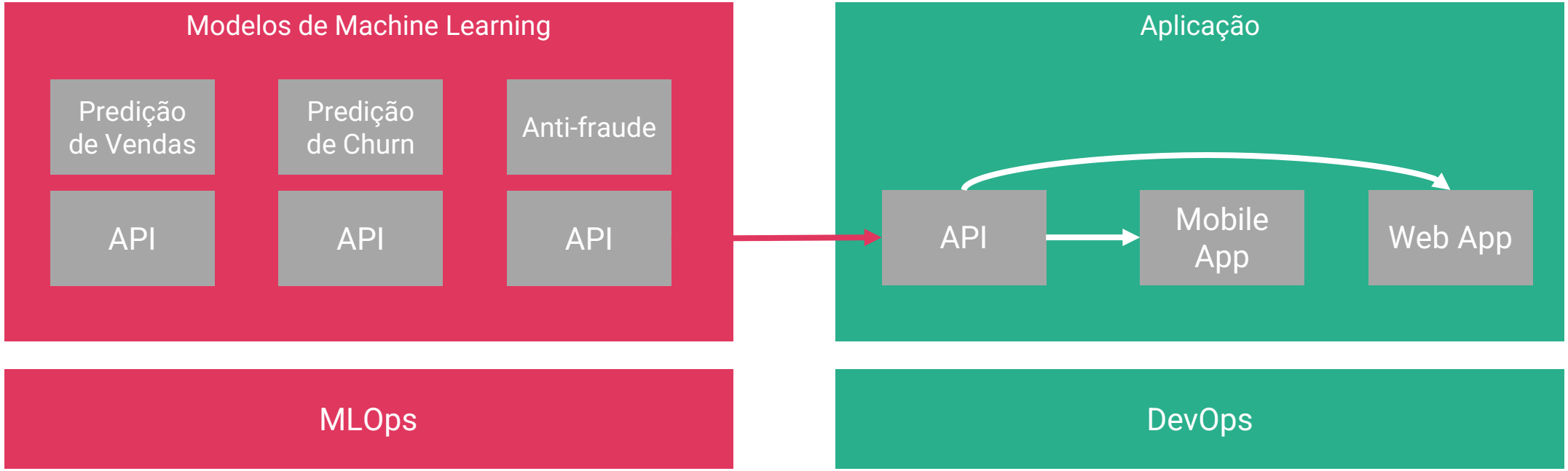
Observabilidade contínuo e feedback rápido permite a detecção precoce de problemas, facilitando a correção proativa.

CULTURA DE MELHORIA CONTÍNUA

Incentivo a experimentação e análise pós-implantação para impulsionar melhorias constantes.

ANTES DO **MACHINE LEARNING OPS**

Apesar de MLOps significar “Machine Learning Operations” e haver uma divisão bem definida entre a construção de modelos e sua disponibilização em produção, ainda existe uma parte, embora menor, relacionada ao desenvolvimento de aplicações. Essa parte diz respeito à forma como os modelos de aprendizado de máquina são expostos e utilizados, geralmente por meio de APIs, o que requer práticas específicas de engenharia de software para garantir eficiência, escalabilidade e segurança.

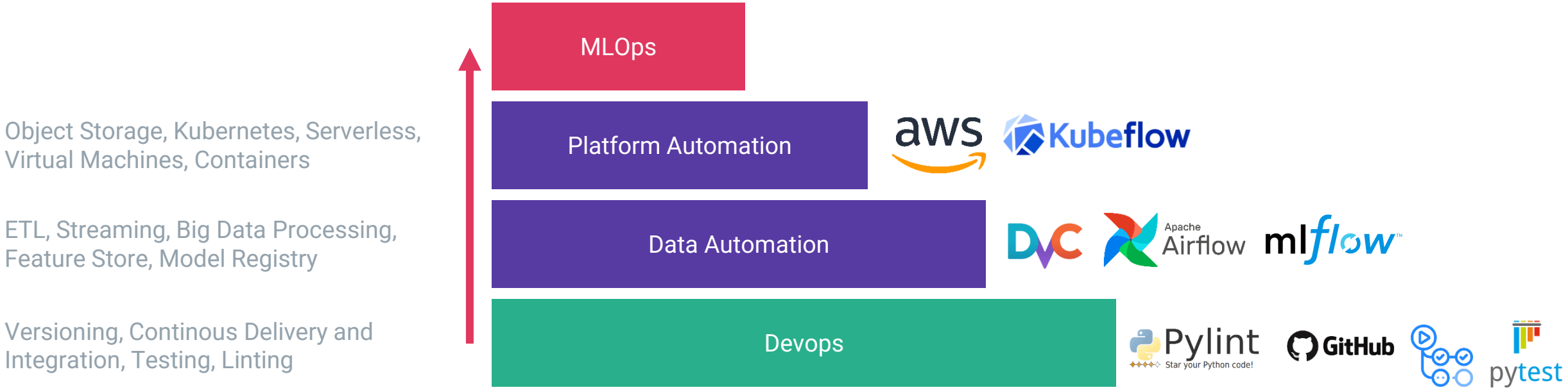


ANTES DO **MACHINE LEARNING OPS**

ML Engineer é um profissional, de acordo com a AWS,

"...possui a habilidade de projetar, implementar, implantar e manter soluções de machine learning para problemas de negócio."

Não compete com DevOps, na verdade MLOps utiliza esta importante fundação para trazer mais valor ao viabilidade ML as a Product.



PRINCIPÍOS DE REFERÊNCIA

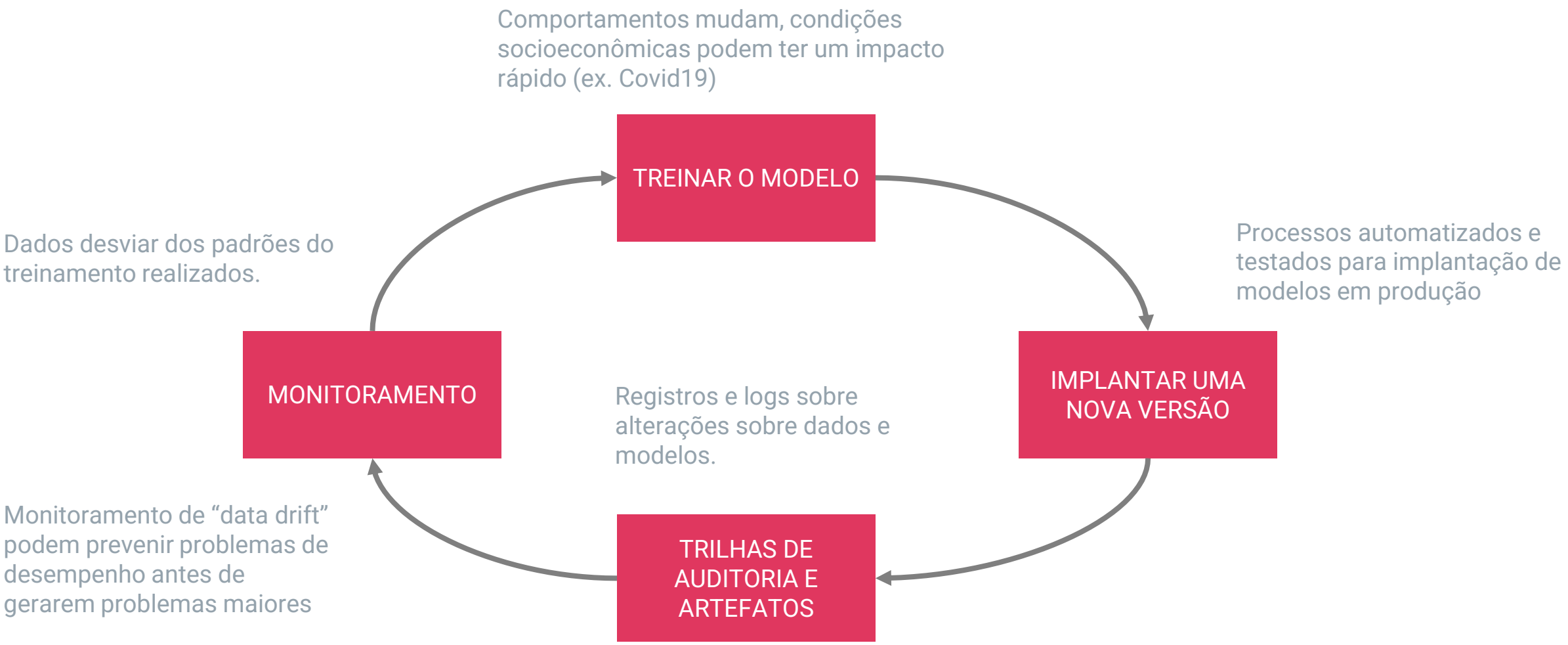
Diferentemente de uma aplicação tradicional, um modelo de aprendizado de máquina precisa seguir alguns princípios específicos. No Nubank (apresentado à época por Luam Catão), por exemplo, esses princípios servem como referência para garantir qualidade, robustez e alinhamento com os objetivos de negócio. Eles orientam desde a criação e validação do modelo até sua manutenção em produção, assegurando que o comportamento do sistema seja previsível e sustentável ao longo do tempo.

https://www.youtube.com/watch?v=N96yiqUFwEo

PRINCIPÍOS DE REFERÊCIA

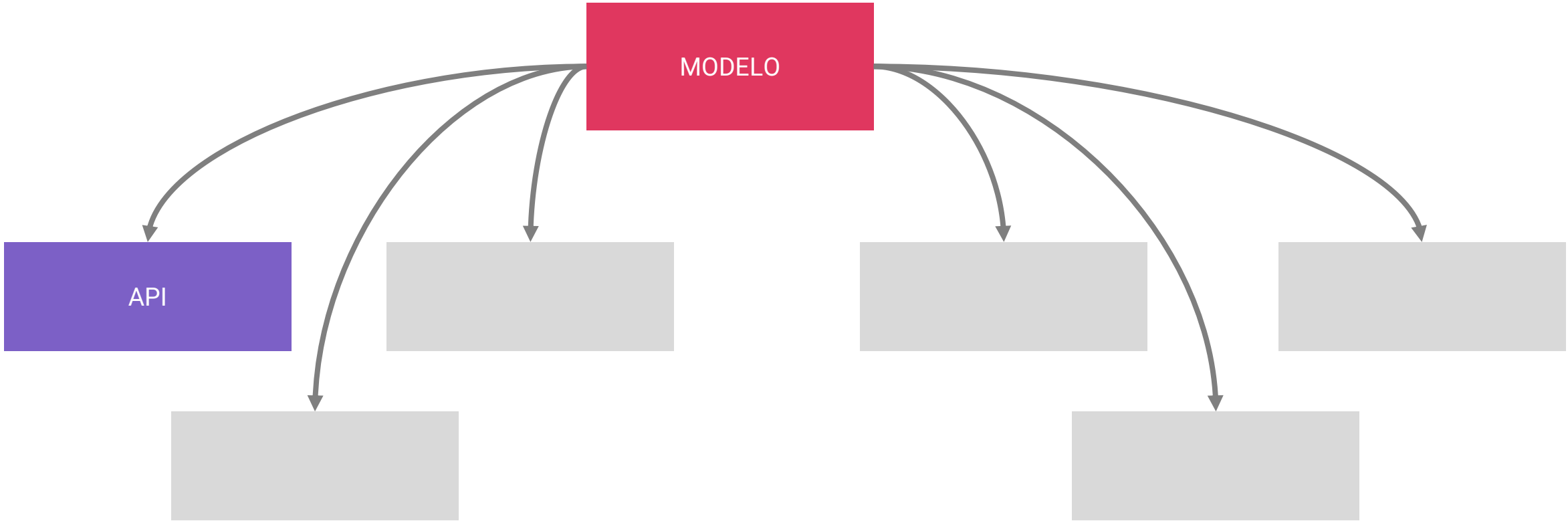
Reprodutibilidade	Modelos precisam ser reproduzíveis para que falhas possam ser verificadas, ou seja, um novo treinamento precisa ocorrer nas mesmas condições e com os mesmos valores, visando atender a requisitos de auditoria.
Consistência	O código-base deve incorporar métricas e monitoramento para promover reusabilidade e facilitar a manutenção.
Continuidade	Para lidar de maneira mais eficaz com a incerteza e possibilitar iterações contínuas nos modelos e decisões.
Responsabilidade	Todas as decisões precisam ser registradas, e qualquer adição ao modelo deve ser devidamente rastreada.

MLOPS **FEEDBACK LOOP**



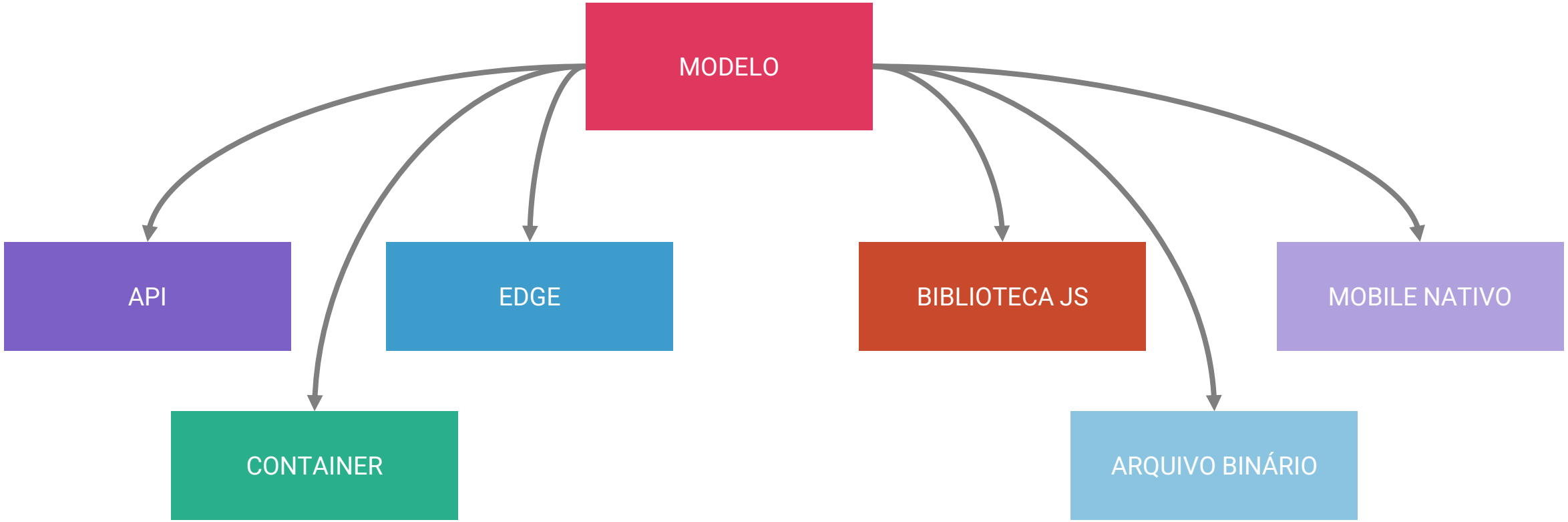
MODELOS EM OPERAÇÃO: NEM TUDO É API

O uso de modelos por meio de APIs é o caso mais comum na integração de soluções de machine learning. Outras plataformas também adotam essa abordagem, embora aplicadas a diferentes tipos de sistemas e contextos. Essa versatilidade permite que modelos sejam incorporados tanto em aplicações web e mobile quanto em serviços internos, mantendo consistência na entrega de previsões e facilitando a escalabilidade da solução.



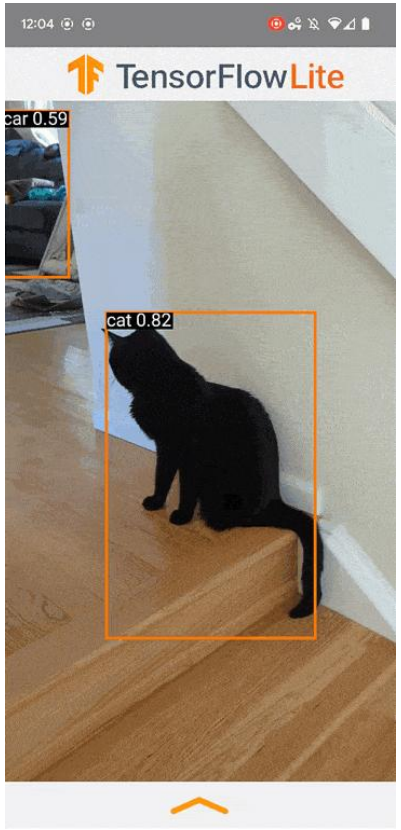
MODELOS EM OPERAÇÃO: NEM TUDO É API

O uso de modelos por meio de APIs é o caso mais comum na integração de soluções de machine learning. Outras plataformas também adotam essa abordagem, embora aplicadas a diferentes tipos de sistemas e contextos. Essa versatilidade permite que modelos sejam incorporados tanto em aplicações web e mobile quanto em serviços internos, mantendo consistência na entrega de previsões e facilitando a escalabilidade da solução.



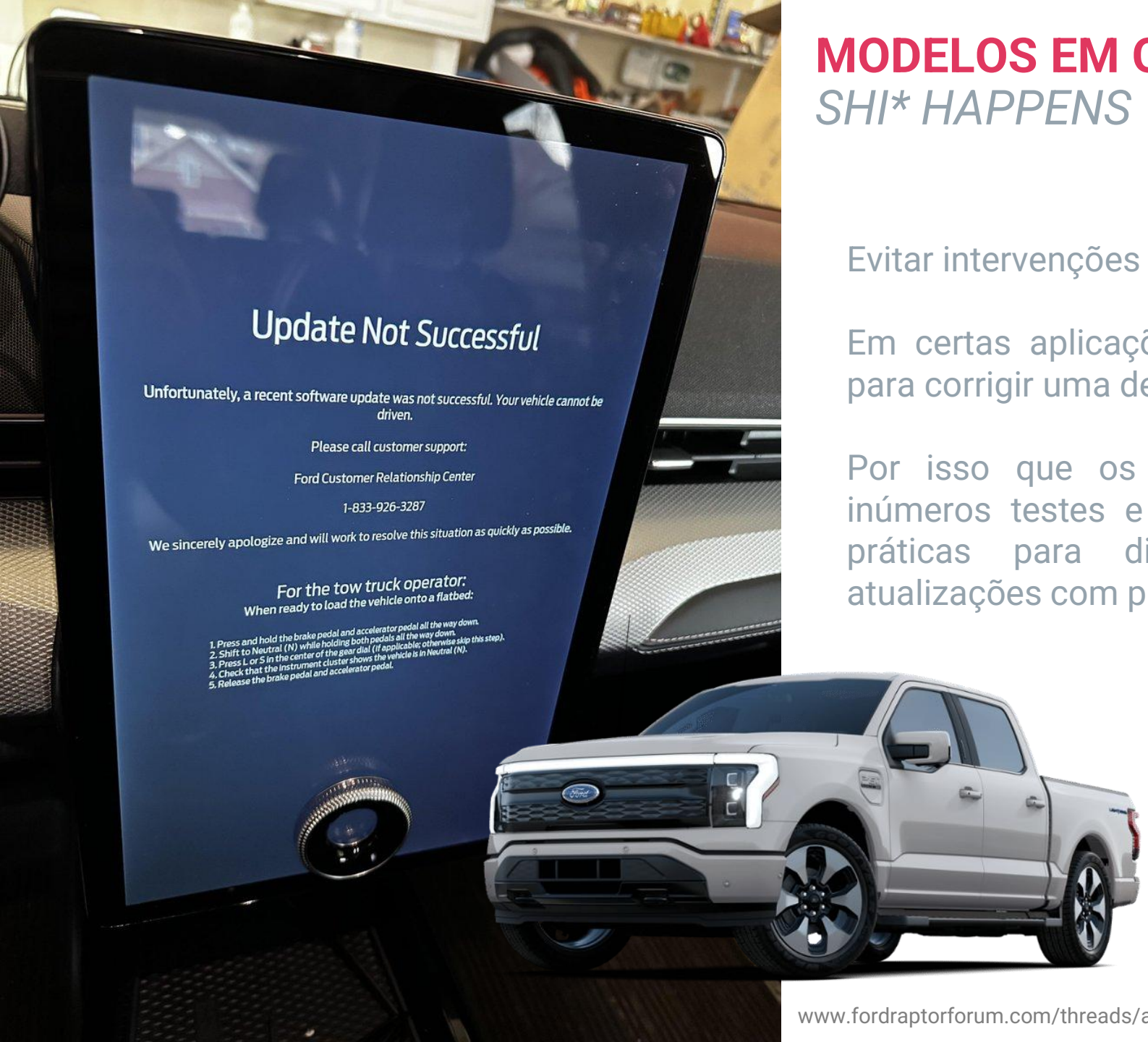
MODELOS EM OPERAÇÃO: NEM TUDO É API

Independentemente da aplicação, o maior desafio está em estabelecer um processo consistente e confiável para a atualização contínua dos modelos. Essa atualização pode ser necessária tanto para adaptação a novos dados quanto para a inclusão de novas funcionalidades, exigindo um fluxo bem definido que minimize riscos e preserve a qualidade do que está em produção.



MODELOS EM OPERAÇÃO

SHI HAPPENS*



Evitar intervenções manuais é primordial.

Em certas aplicações, o suporte pode levar muito tempo para corrigir uma determinada falha.

Por isso que os pacotes de atualização precisam ter inúmeros testes e serem escritos utilizando as melhores práticas para diminuir a possibilidade de termos atualizações com problemas.

www.fordraptorforum.com/threads/automatic-software-update-bricked-my-truck.96624/

DO DEVOPS: **TESTE** E **ANÁLISE ESTÁTICA**

Tanto para o serviço que utilizará o modelo quanto para o próprio modelo, é possível realizar uma série de testes antes de colocá-lo em produção. Esses testes ajudam a garantir que o comportamento esteja dentro do esperado, evitando falhas, inconsistências e impactos negativos na experiência do usuário ou nos resultados do negócio.

MODELO

SERVIÇO

DO DEVOPS: **TESTE** E **ANÁLISE ESTÁTICA**

FIAP MBA+

Tanto para o serviço que utilizará o modelo quanto para o próprio modelo, é possível realizar uma série de testes antes de colocá-lo em produção. Esses testes ajudam a garantir que o comportamento esteja dentro do esperado, evitando falhas, inconsistências e impactos negativos na experiência do usuário ou nos resultados do negócio.

MODELO

Número de parâmetros de entrada

Golden data (previsões já esperadas)

Validação de resultados (valores <0 por exemplo podem serem inválidos para certos tipos de regressão)

Tamanho do modelo

Tempo de resposta de predição

SERVIÇO

Parâmetros de entrada e saída

Utilização de parâmetros que minimizem potenciais problemas, por exemplo, uma biblioteca que faz uma requisição precisa ter um timeout

Validação de artefatos

Padrões de código

Padrões de desempenho em geral

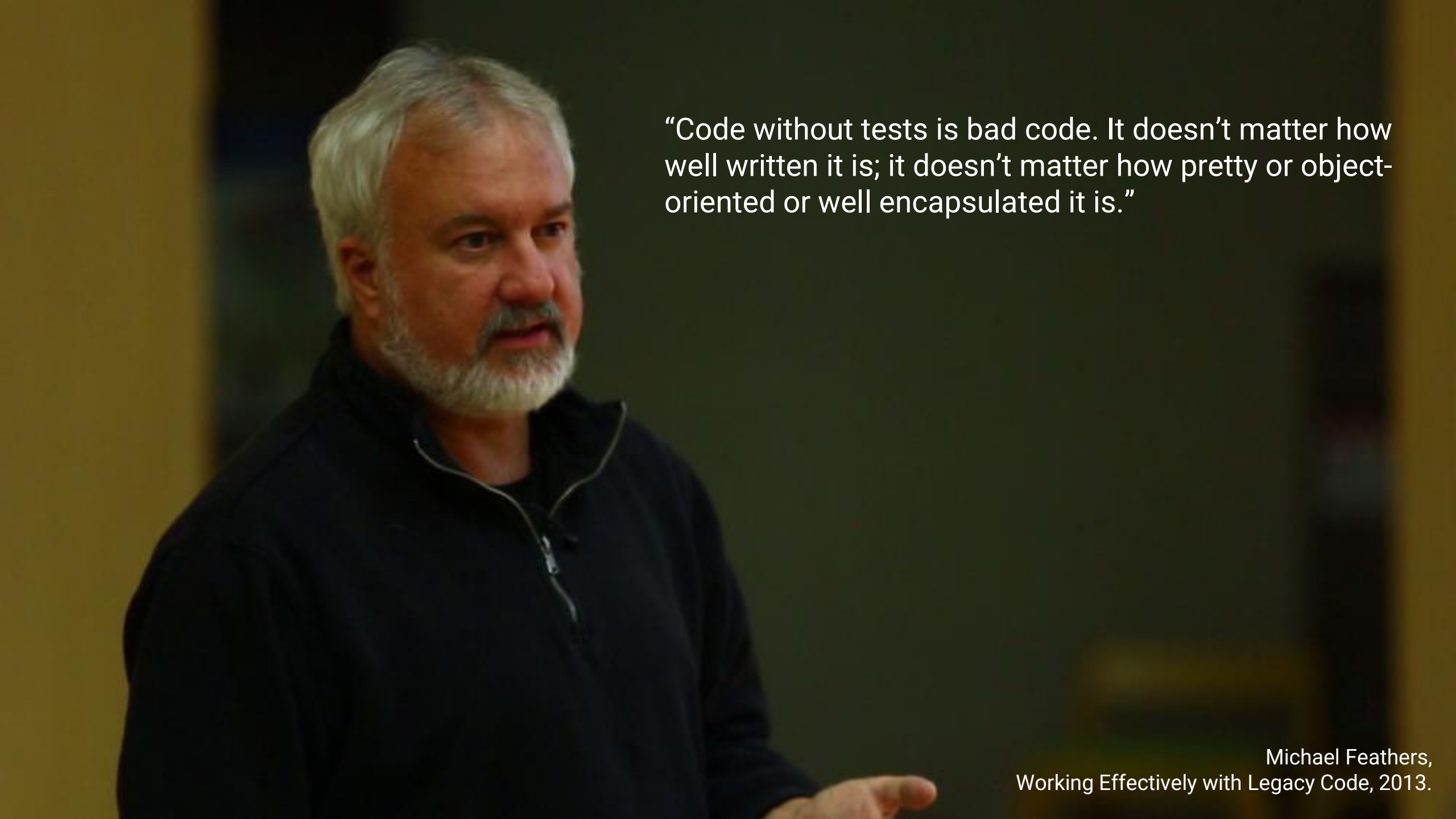
TESTES DE UNIDADE

DO DEVOPS: **TESTE DE UNIDADE**

FIAP MBA+

Um teste de unidade é uma forma de teste de software que se concentra na verificação individual de componentes isolados de um programa. O objetivo é validar se cada unidade (módulo, função, método, etc.) do software funciona conforme o esperado. Uma unidade pode ser uma função simples, uma classe ou até mesmo um conjunto de funções relacionadas

ISOLAMENTO	Cada unidade é testada de forma isolada, sem depender do estado ou do comportamento de outras partes do sistema.
AUTOMAÇÃO	Os testes de unidade são geralmente automatizados, o que significa que podem ser executados de forma rápida e repetida durante o desenvolvimento.
RASTREABILIDADE	Os resultados dos testes de unidade devem ser rastreáveis até o código específico da unidade que está sendo testada.
RAPIDEZ	Os testes de unidade são projetados para serem rápidos, permitindo que desenvolvedores os executem frequentemente durante o ciclo de desenvolvimento.
FACILIDADE DE DIAGNÓSTICO	Caso um teste de unidade falhe, deve ser fácil identificar a causa do problema para que os desenvolvedores possam corrigi-lo rapidamente.



“Code without tests is bad code. It doesn’t matter how well written it is; it doesn’t matter how pretty or object-oriented or well encapsulated it is.”

Michael Feathers,
Working Effectively with Legacy Code, 2013.

TESTE DE UNIDADE, UM CASO PRÁTICO

A biblioteca mais utilizada no mercado é a Pytest (https://docs.pytest.org/).
Assim como qualquer teste de unidade, o objetivo é validar por meio de asserções se o resultado obtido é o esperado.

```
# Conteúdo do arquivo test_soma.py

def soma(a, b):
    return a + b

def test_soma_numeros_positivos():
    resultado = soma(2, 3)
    assert resultado == 5, "A soma de 2 e 3 deveria ser 5"

def test_soma_numeros_negativos():
    resultado = soma(-2, -3)
    assert resultado == -5, "A soma de -2 e -3 deveria ser -5"

def test_soma_misto_positivo_negativo():
    resultado = soma(10, -5)
    assert resultado == 5, "A soma de 10 e -5 deveria ser 5"

def test_soma_zero():
    resultado = soma(0, 0)
    assert resultado == 0, "A soma de 0 e 0 deveria ser 0"
```

Se um teste falhar, deverá ser realizado uma verificação até que todos eles sejam cobertos.

Uma boa cobertura de testes é ter, ao mínimo, 80% de uma determinada aplicação. Isto implica que a maioria de todas as funções precisam constar estes testes.

Como os testes sempre crescem, quando há uma nova funcionalidade implementada é possível verificar, por meio dos testes de unidade, se não há nenhuma quebra das funcionalidades já existentes.

TESTE DE UNIDADE, UM CASO PRÁTICO

Após instalar o Pytest, podemos executar o comando para realizar o teste: “pytest –v”, o parâmetro utilizado indica apresentar verbosidade padrão sobre os testes.

```
========================================================================== test session starts ==========================================================================
platform win32 -- Python 3.7.9, pytest-7.4.4, pluggy-1.2.0 -- d:\github_projects\fiap-ds-mlops-start\.venv\scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\
collected 2 items

tests\app_test.py::test_api_return PASSED                                                                                                                  [ 50%]
tests\app_test.py::test_api_price_return PASSED                                                                                                            [100%]

========================================================================== 2 passed in 2.69s ==========================================================================
```

Quando um dos testes falha, é apresentada a mensagem negativa da asserção para investigação posterior.

```
================================================================================ FAILURES================================================================================
_____________________________________________________________________________ test_api_price_return _____________________________________________________________________________

    def test_api_price_return():
        result = app.get_product_price(1)
>       assert isinstance(result, (string)), "A variável não é numérica"
E       NameError: name 'string' is not defined

tests\app_test.py:12: NameError
========================================================================== short test summary info ==========================================================================
FAILED tests\app_test.py::test_api_price_return - NameError: name 'string' is not defined
========================================================================== 1 failed, 1 passed in 2.75s ==========================================================================
```

TESTE DE UNIDADE, UM CASO PRÁTICO

FIAP MBA+

Para estas funções, quais seriam bons testes de unidade?

```python
import requests


def get_product(id):
    results = requests.get("https://dummyjson.com/products/"+str(id))
    return results

def get_product_price(id):
    results = requests.get("https://dummyjson.com/products/"+str(id))
    data = results.json()
    return data["price"]
```

TESTE DE UNIDADE, UM CASO PRÁTICO

Os testes mais diretos do “caminho feliz”:

1. Teste baseado no código de sucesso 200.
2. Teste baseado no tipo de retorno.

```
def test_api_return():
  results = app.get_product(1)
  data = results.json()

  assert results.status_code == 200, "O retorno da conexão não foi bem sucedida"
  assert "id" in data, "Não existe chave id no retorno"
  assert data["id"] == 1, "Id enviado não é o mesmo recebido"


def test_api_price_return():
  result = app.get_product_price(1)
  assert isinstance(result, (int, float)), "O preço não é numérico"
```

TESTE DE UNIDADE, UM CASO PRÁTICO

FIAP MBA+

Outros testes “mocados” para condições específicas e importantes para testar:

1. Teste de erro baseado no status code.
2. Teste de erro baseado em time out.

```
def test_get_product_http_error_handling(mocker):
  mocker.patch("src.app.requests.get").return_value.status_code = 404
  result = app.get_product(2)
  assert result is None, "Valor não nulo em condição de erro."


def test_timeout_handling(mocker):
  mocker.patch("src.app.requests.get", side_effect=requests.exceptions.Timeout)
  result = app.get_product(3)
  assert result is None, "Valor não nulo em condição de erro."
```

DESAFIO 1

Utilizando o mesmo endpoint anterior (https://dummyjson.com/products/1), crie outro método para retornar o conteúdo da chave “images”.

Construa testes de unidade para validar a integridade deste resultado. Deve incluir as seguintes regras:

1. Deve retornar no mínimo uma imagem.
2. O formato da imagem deve ser JPG ou PNG,
3. O nome das imagens deve ser números, por exemplo 1.jpg, 2.jpg.

ANÁLISE ESTÁTICA (LINT)

ANÁLISE ESTÁTICA: **LINT**

Ferramentas de análise estática de código auxiliam os desenvolvedores na identificação e correção de problemas ainda na fase de desenvolvimento. Elas analisam o código-fonte em busca de padrões, aderência a estilos de codificação, possíveis erros e outras questões que possam comprometer a legibilidade, a manutenibilidade e a qualidade geral do software.

O termo "lint" originou-se do nome de uma ferramenta chamada "lint", desenvolvida nos anos 1970 para a linguagem de programação C. A ideia era que a ferramenta ajudaria a pentear o código em busca de problemas, assim como um "linter" de roupas remove partículas indesejadas.

ANÁLISE ESTÁTICA: **LINT**

CONFORMIDADE	Verificação se o código segue as convenções de formatação e estilo estabelecidas pela equipe de desenvolvimento ou por diretrizes específicas da linguagem.
ERROS POTENCIAIS	Identificação de erros lógicos, problemas de segurança, ou outras questões que podem causar comportamento inesperado durante a execução do programa.
VARIÁVEIS NÃO DECLARADAS	Detecção de variáveis que podem não ter sido declaradas ou que estão sendo usadas antes de serem inicializadas.
COMPLEXIDADE DE CÓDIGO	Avaliação da complexidade do código-fonte, como número de linhas de código em uma função ou a profundidade de aninhamento de estruturas de controle.

PYLINT, ANÁLISE ESTÁTICA EM PYTHON

O Pylint (https://pylint.readthedocs.io) é uma ferramenta de linting para o Python. Ele é projetado para analisar o código-fonte, identificar possíveis problemas, aplicar convenções de estilo e fornecer feedback aos desenvolvedores sobre a qualidade do código. O objetivo principal do Pylint é melhorar a legibilidade, a manutenibilidade e a qualidade geral do código Python.

Pylint
Star your Python code!

- **Verificação de Estilo de Código**: verifica se o código Python segue as convenções de estilo recomendadas pelo PEP 8 (https://peps.python.org/pep-0008/), que é o guia de estilo oficial para código Python.
- **Identificação de Erros Potenciais**: analisa o código em busca de possíveis erros lógicos, como uso incorreto de variáveis, chamadas de funções com número incorreto de argumentos, entre outros.
- **Métricas de Código**: fornece métricas relacionadas à complexidade do código, como o número de linhas de código em uma função, a profundidade de aninhamento e outros indicadores que podem ajudar a avaliar a qualidade do código.
- **Suporte a Convenções Personalizadas**: além das convenções do PEP 8, o Pylint para seguir convenções específicas da sua equipe ou projeto.
- **Integração com Editores de Código**: pode ser integrado com muitos editores de código, IDEs (Ambientes de Desenvolvimento Integrado) e ambientes de desenvolvimento, proporcionando feedback em tempo real enquanto você escreve código.

PONTUAÇÃO 0-10

REFERÊNCIAS (DOC. PYLINT)

PYLINT, ANÁLISE ESTÁTICA EM PYTHON

FIAP MBA+

Qual é a nota para o código abaixo?

```
import requestss


def get_product(id):
    results = requests.get("https://dummyjson.com/products/"+str(product_id), timeout=10)
    return results

def get_product_price(id):
    results = requests.get("https://dummyjson.com/products/"+str(product_id), timeout=10)
    data = results.json()
    return data["price"]

get_product(1)
get_product_price(1)
```

PYLINT, ANÁLISE ESTÁTICA EM PYTHON

FIAP MBA+

PONTUAÇÃO 0!

pylint app.py

```
************* Module app
src\app.py:14:0: C0304: Final newline missing (missing-final-newline)
src\app.py:1:0: C0114: Missing module docstring (missing-module-docstring)
src\app.py:1:0: E0401: Unable to import 'requestss' (import-error)
src\app.py:4:0: C0116: Missing function or method docstring (missing-function-docstring)
src\app.py:4:16: C0103: Argument name "id" doesn't conform to snake_case naming style (invalid-name)
src\app.py:4:16: W0622: Redefining built-in 'id' (redefined-builtin)
src\app.py:5:14: E0602: Undefined variable 'requests' (undefined-variable)
src\app.py:5:65: E0602: Undefined variable 'product_id' (undefined-variable)
src\app.py:4:16: W0613: Unused argument 'id' (unused-argument)
src\app.py:8:0: C0116: Missing function or method docstring (missing-function-docstring)
src\app.py:8:22: C0103: Argument name "id" doesn't conform to snake_case naming style (invalid-name)
src\app.py:8:22: W0622: Redefining built-in 'id' (redefined-builtin)
src\app.py:9:14: E0602: Undefined variable 'requests' (undefined-variable)
src\app.py:9:65: E0602: Undefined variable 'product_id' (undefined-variable)
src\app.py:8:22: W0613: Unused argument 'id' (unused-argument)
src\app.py:1:0: W0611: Unused import requestss (unused-import)

------------------------------------------------------------------
Your code has been rated at 0.00/10 (previous run: 0.00/10, +0.00)
```

PYLINT, ANÁLISE ESTÁTICA EM PYTHON

FIAP MBA+

E agora, depois dos ajustes?

```
"""Module responsible to get data from produts API"""
import requests


def get_product(product_id):
    """Get a product by id"""
    results = requests.get("https://dummyjson.com/products/"+str(product_id), timeout=10)
    return results

def get_product_price(product_id):
    """Get a product price by id"""
    results = requests.get("https://dummyjson.com/products/"+str(product_id), timeout=10)
    data = results.json()
    return data["price"]

get_product(1)
get_product_price(1)
```

PYLINT, ANÁLISE ESTÁTICA EM PYTHON

FIAP MBA+

PONTUAÇÃO 10!

```
------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 6.00/10, +4.00)
```

É comum não alcançar sempre o 10/10.

Muitas empresas adotam Linters específicos para cada time ou então flexibilizam a nota final para algo próximo a 10, sendo acima de 8 aceitável.

DESAFIO 2

No mesmo projeto utilizado no desafio anterior, crie mais um método responsável por baixar as imagens dos resultados obtidos da chave “imagem”.

Logo após execute o Lint padrão (baseado no PEP-8) e realize os ajustes necessários.

PARA SABER **MAIS**

https://www.youtube.com/watch?v=N96yiqUFwEo, 101 Master Class de Machine Learning Engineering do Nubank. Material indispensável para quem deseja prosseguir na área.
https://cursos.alura.com.br/course/python-tdd-explorando-testes-unitarios, Curso Python e TDD: explorando testes unitários.
https://wiki.python.org.br/GuiaDeEstilo, Guia de Estilo Python (PEP-8) em português.

FIAP MBA+

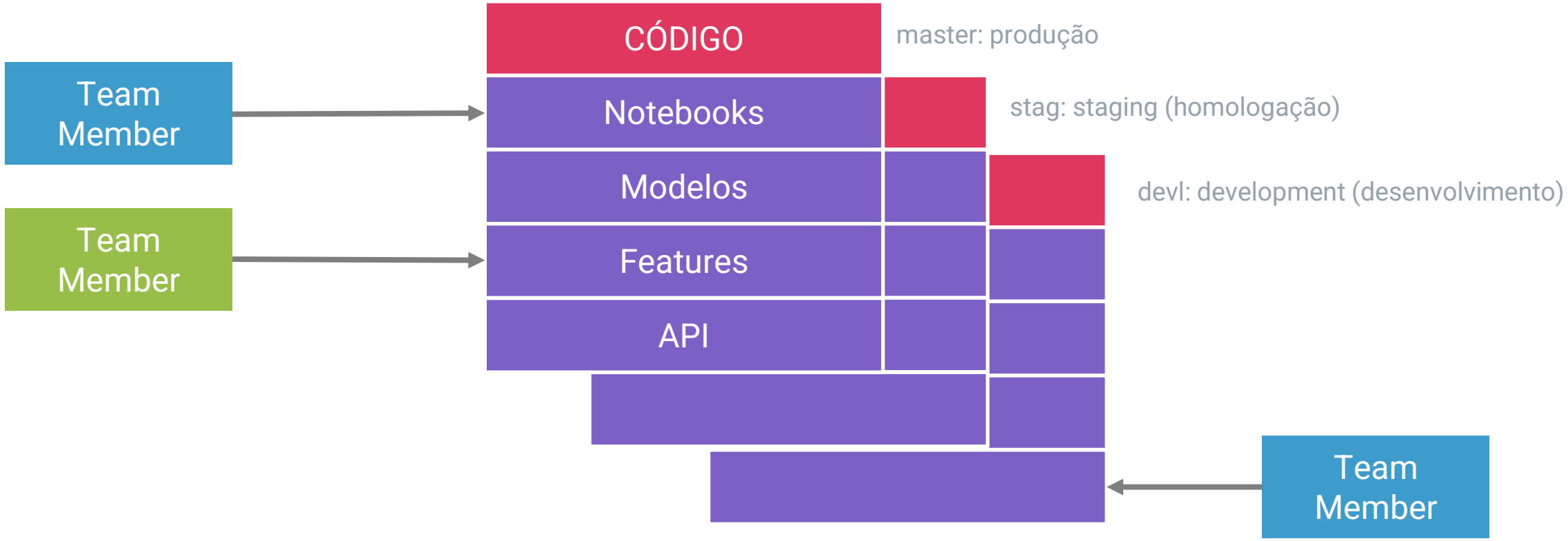
MACHINE LEARNING ENGINEERING

MBA EM DATA SCIENCE & IA

GERENCIAMENTO DE VERSÃO DE CÓDIGOS

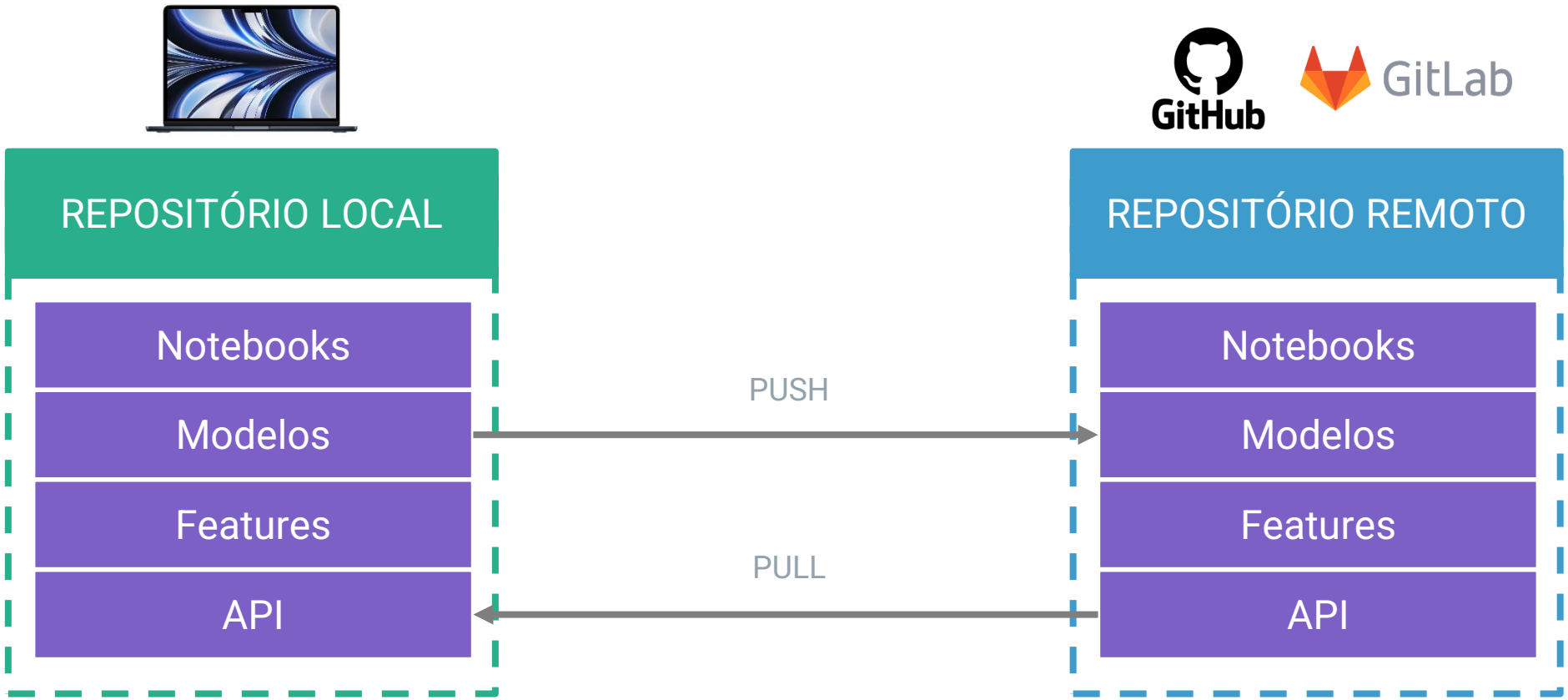
VERSIONAMENTO DE CÓDIGO

O versionamento de código permite controlar as alterações em uma base comum, possibilitando sua evolução contínua e a colaboração de múltiplos times. Esse controle contribui para minimizar conflitos, garantir a integridade do histórico de mudanças e manter a padronização do desenvolvimento ao longo do tempo.



REPOSITÓRIOS DE CÓDIGO

O repositório é o local onde são registradas as modificações e versões dos arquivos de uma determinada pasta do projeto. Após realizar as alterações necessárias, o próximo passo é enviá-las para um repositório remoto, permitindo que outras pessoas possam acessar, revisar e colaborar no mesmo código.



CÓDIGO **VERSIONÁVEL**

Nem todos os arquivos de um projeto precisam ser versionados. Devem ser incluídos apenas aqueles que serão utilizados em equipe, passíveis de testes, validação e eventual publicação em produção. Arquivos de uso individual, ambientes virtuais e arquivos sensíveis, como os que contêm senhas ou variáveis de ambiente (por exemplo, arquivos .env), devem ser explicitamente ignorados. No VS Code, há uma extensão chamada *Code Zombie* que gera automaticamente arquivos de ignore padronizados conforme a linguagem de programação utilizada, facilitando essa configuração.

https://marketplace.visualstudio.com/items?itemName=codezombiech.gitignore

ADICIONANDO ARQUIVOS

Para começarmos a trabalhar com o Git precisamos instalar no sistema operacional utilizado. No Windows, instalamos a partir desta fonte (https://www.git-scm.com/downloads) e para o Mac é recomendado utilizar o Brew (https://git-scm.com/download/mac).

1. CRIA REPOSITÓRIO LOCAL

git init

1. ADICIONA GITIGNORE (Opcional)

edite um arquivo .gitignore e adicione nele os arquivos ignorados

3. ADICIONA ARQUIVOS

git add .
adiciona todos os arquivos (pode ser adicionado arquivos específicos)

CONFIRMANDO ARQUIVOS

Após concluir todas as modificações, podemos confirmá-las por meio de um commit, preparando assim o envio para um repositório remoto compartilhado. Esse processo registra oficialmente as alterações no histórico do projeto e permite que outros colaboradores as integrem de forma segura.

CONFIRMAÇÃO

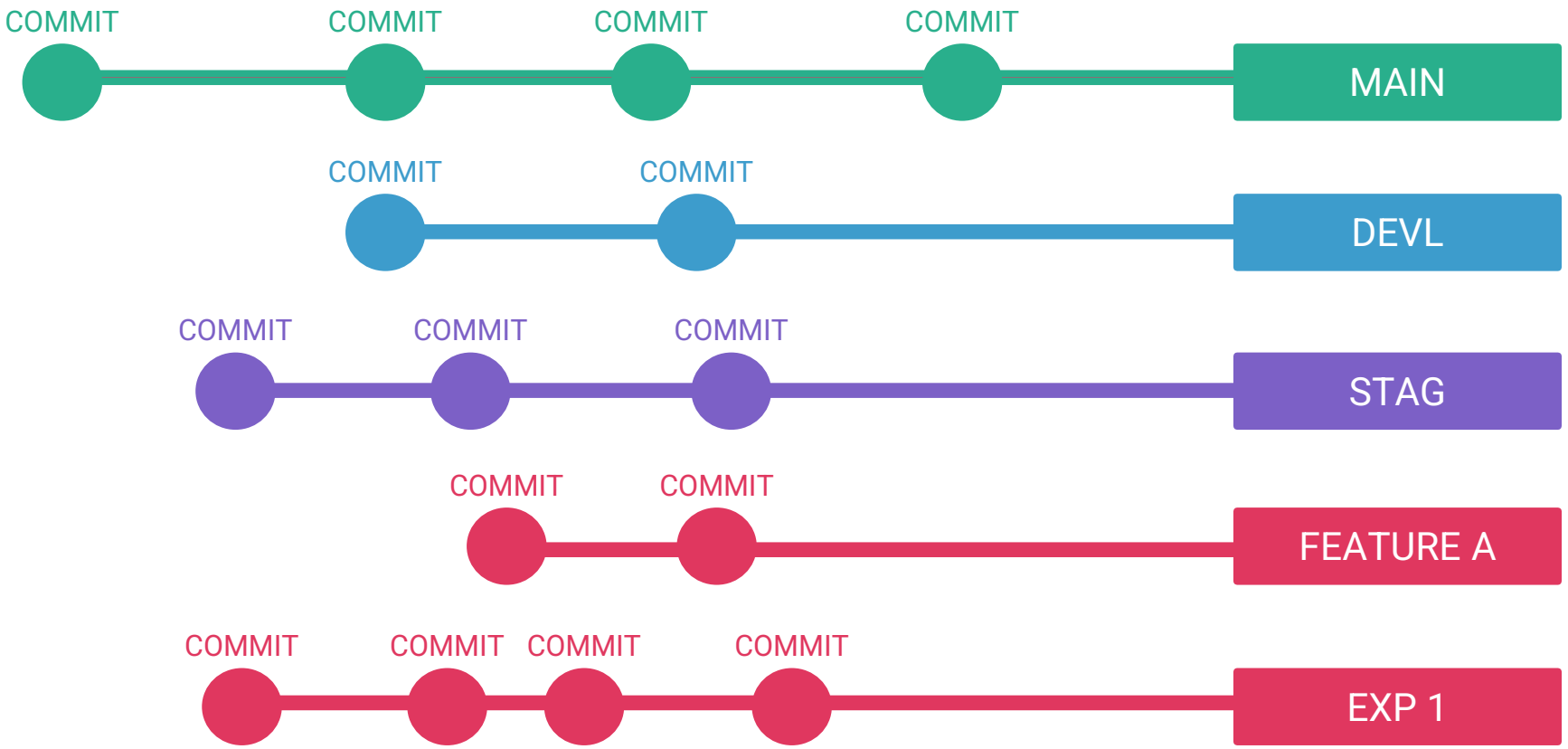
git commit -m mensagem
Cada confirmação requer uma mensagem para rastrear as razões das modificações.

HASH DE MODIFICAÇÃO ÚNICO (SHA-1)

O hash de modificação permite controlar cada uma as modificações. Assim é possível saber o que foi alterado e por qual pessoa seja para auditoria ou mesmo para sincronismo entre repositórios.

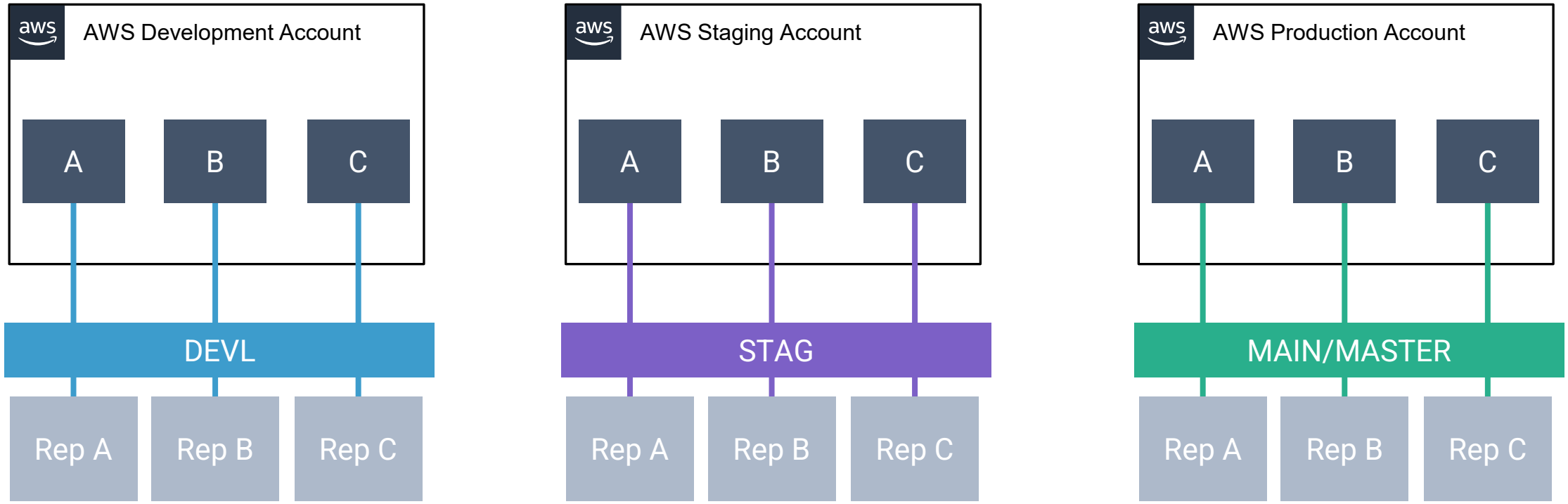
RAMIFICAÇÕES DO REPOSITÓRIO

As ramificações permitem que os membros de um time, mesmo atuando sobre a mesma base de código, possam trabalhar de forma independente, reduzindo o risco de conflitos de versão. Uma ramificação também pode ser utilizada para testes ou experimentos, sem a necessidade de que as alterações feitas nela sejam, necessariamente, integradas à versão de produção.



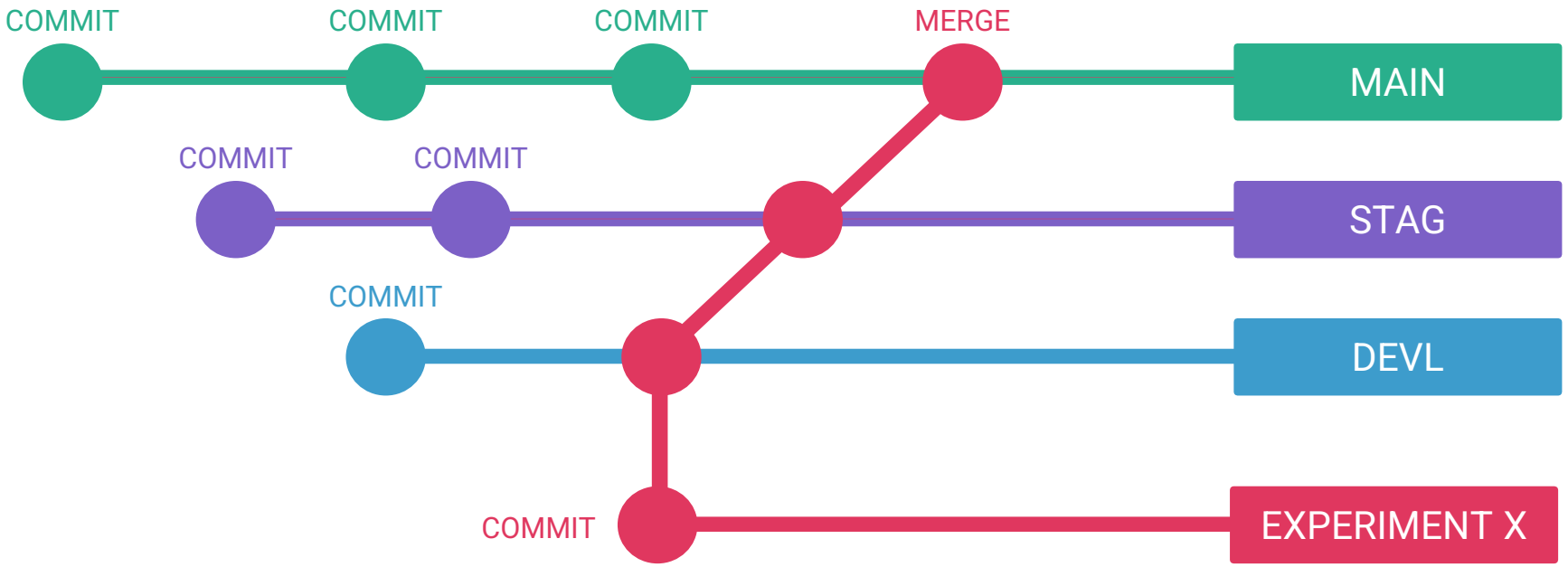
RAMIFICAÇÕES E AMBIENTES

As ramificações também são utilizadas como referência para os diferentes ambientes de um sistema. Por exemplo, a ramificação principal, geralmente chamada de master ou main, costuma estar associada ao ambiente de produção. A ramificação dev, de desenvolvimento, está ligada ao ambiente onde novas funcionalidades são construídas. Já a ramificação stag, de staging ou homologação, é usada para testes finais antes da liberação. Com essa estrutura de três ambientes, é possível desenvolver uma nova funcionalidade localmente no laptop do especialista, validá-la em um ambiente de desenvolvimento com condições similares às de produção, realizar testes em homologação com dados mais atualizados e, por fim, liberar para o ambiente produtivo.



GITFLOW E MERGES

As ramificações permitem que os integrantes de um time que trabalham na mesma base de código possam desenvolver de forma independente, sem preocupações imediatas com conflitos de versão. Uma ramificação também pode ser usada para testes ou experimentos, sem que isso implique, necessariamente, a publicação das modificações no ambiente de produção.



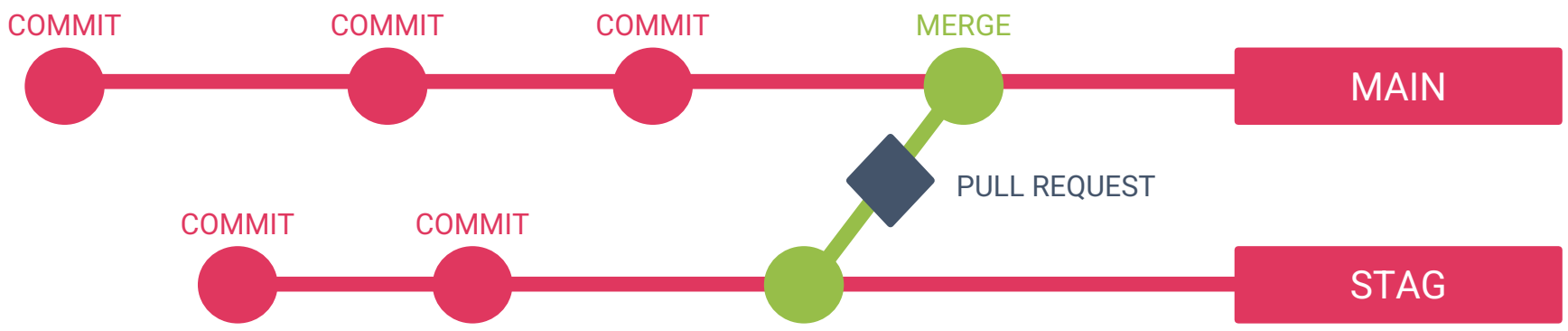
https://www.alura.com.br/artigos/git-flow-o-que-e-como-quando-utilizar

PULL REQUEST

O Pull Request (PR) pode ser opcional, pois se o desenvolvedor tiver permissões para “merge” na branch principal, ele pode consolidar as alterações. No entanto, como prática de desenvolvimento, o PR se torna uma forma de revisão e de explicação do desenvolvedor por qual razão ele quer consolidar a proposta dele de código em produção.

Membros do time mais seniores e revisores farão uma revisão detalhada para permitir a consolidação do código.

Utilizando automações de integração contínua e pipelines também é possível realizar ações ao submeter um PR, inclusive rejeitando-a quando a mesma não passar nos testes ou adicionando informações relevantes para o revisor analisar.



TEMPLATES DE PROJETO

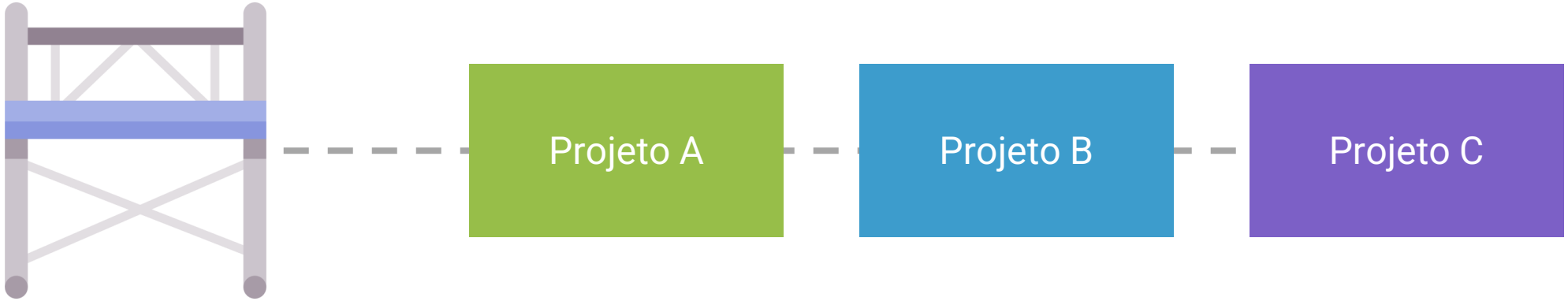
SCAFFOLDING DE PROJETOS DE ML

A padronização da estrutura de pastas em um projeto é essencial para facilitar o uso da base de código por toda a equipe.

Organizar o repositório com áreas específicas para Notebooks, scripts de treinamento, processamento de dados e outros componentes ajuda a manter os processos reproduzíveis e reduz o tempo de desenvolvimento em novas iniciativas.

Cada empresa ou departamento pode adotar um modelo próprio, de acordo com seus desafios e necessidades.

Uma forma prática de manter templates padronizados é com o uso do Cookiecutter, que permite criar e reutilizar estruturas de projeto de forma rápida.



TEMPLATE UTILIZANDO **COOKIECUTTER**

O Cookiecutter precisa ser instalado como pacote no Python (fora do ambiente virtual).

Por meio de comandos CLI passamos um template e ele constrói toda a estrutura de pastas.

```
python -m pip install --user pipx
python -m pipx ensurepath
```

Logo depois passamos o template a seguir. Este template é da DVC e pode ser adaptado para qualquer necessidade.

```
pipx install cookiecutter-data-science
```

No diretório em que você quer iniciar o projeto:

```
ccds
```

https://drivendata.github.io/cookiecutter-data-science/

ML **TEMPLATE**

Os dados ficam na pasta “data” separado por tipo de processamento.

Documentação ficam na pasta “docs” (readthedocs.com)

Modelos ficam na pasta “models”.

Experimentos com Notebooks ficam na pasta “notebooks”.

Manuais e referências adicionais como Papers ficam na pasta “references”.

Dentro da pasta “src” ficam os scripts para treinamento e processamento de features.

Alguns componentes ainda não precisamos utilizar e podemos remover, que é o caso do setup.py, tox.ini e Makefile. Eles são utilizados para compilar diferentes tarefas dentro do ambiente Python, e requerem um ambiente baseado em Linux.

```
├── LICENSE            <- Open-source license if one is chosen
├── Makefile           <- Makefile with convenience commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default mkdocs project; see www.mkdocs.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── pyproject.toml     <- Project configuration file with package metadata for
│                         laptop_pricing_brl and configuration for tools like black
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.cfg          <- Configuration file for flake8
│
└── laptop_pricing_brl   <- Source code for use in this project.
    │
    ├── __init__.py             <- Makes laptop_pricing_brl a Python module
    │
    ├── config.py               <- Store useful variables and configuration
    │
    ├── dataset.py              <- Scripts to download or generate data
    │
    ├── features.py             <- Code to create features for modeling
    │
    ├── modeling
    │   ├── __init__.py
    │   ├── predict.py          <- Code to run model inference with trained models
    │   └── train.py            <- Code to train models
    │
    └── plots.py                <- Code to create visualizations
```

VERSIONAR **MODELOS E FEATURES** ?

O Github possui um tier especial para armazenamento de Large Files de até 2GB.
Nos repositórios não é possível versionar e armazenar arquivos maiores do que 100 MB.
Artefatos de features e modelos podem ser armazenados em outras estruturas, como por exemplo em Object Storages que é o caso do S3 da AWS.

GitHub

COMO VERSIONAR EM LOCAIS DIFERENTES?

DVC FOR THE WIN!

O DVC (Data Version Control) resolve esse problema.
Utilizando a mesma estrutura de projetos do Git (seja no GitHub ou qualquer outra plataforma) ele mantem todas as modificações do projeto e adiciona outra plataforma de armazenamento de arquivos grandes, que é o caso de modelos, features ou qualquer outro artefato.

DAGSHUB TO RULE THEM ALL!

A plataforma DagsHub (dagshub.com) integra os principais serviços necessários para o gerenciamento moderno de projetos de machine learning e large language models (LLMs), oferecendo uma infraestrutura colaborativa que combina o controle de versão de dados e modelos via DVC, o versionamento de código com GitHub e o rastreamento de experimentos por meio do MLflow.

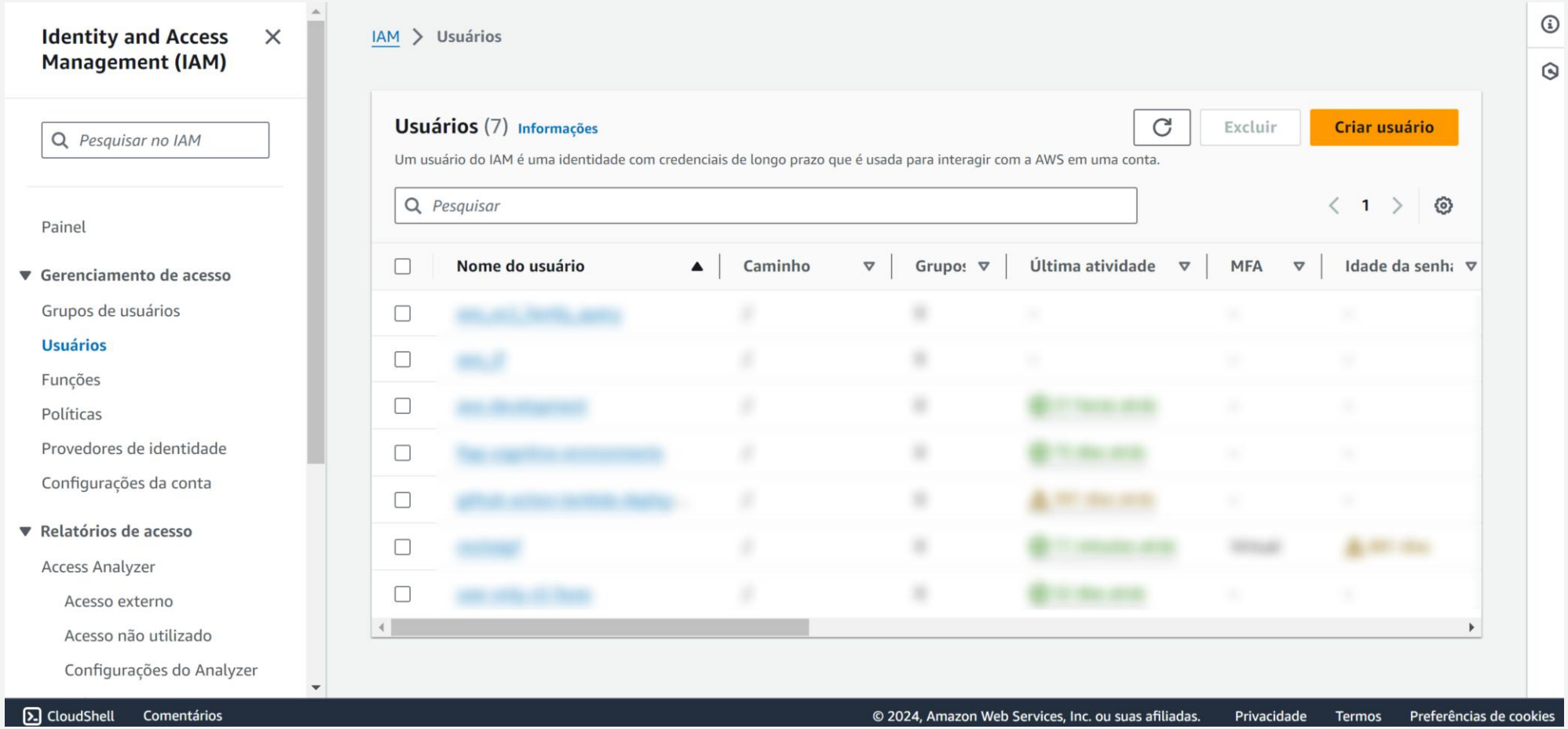
Optaremos por utilizá-la devido à sua praticidade, integração nativa com ferramentas amplamente adotadas no ecossistema de ciência de dados e por disponibilizar um plano gratuito (Community Version), adequado para fins de ensino e pesquisa acadêmica.

DagsHub

GitHub DVC mlflow™

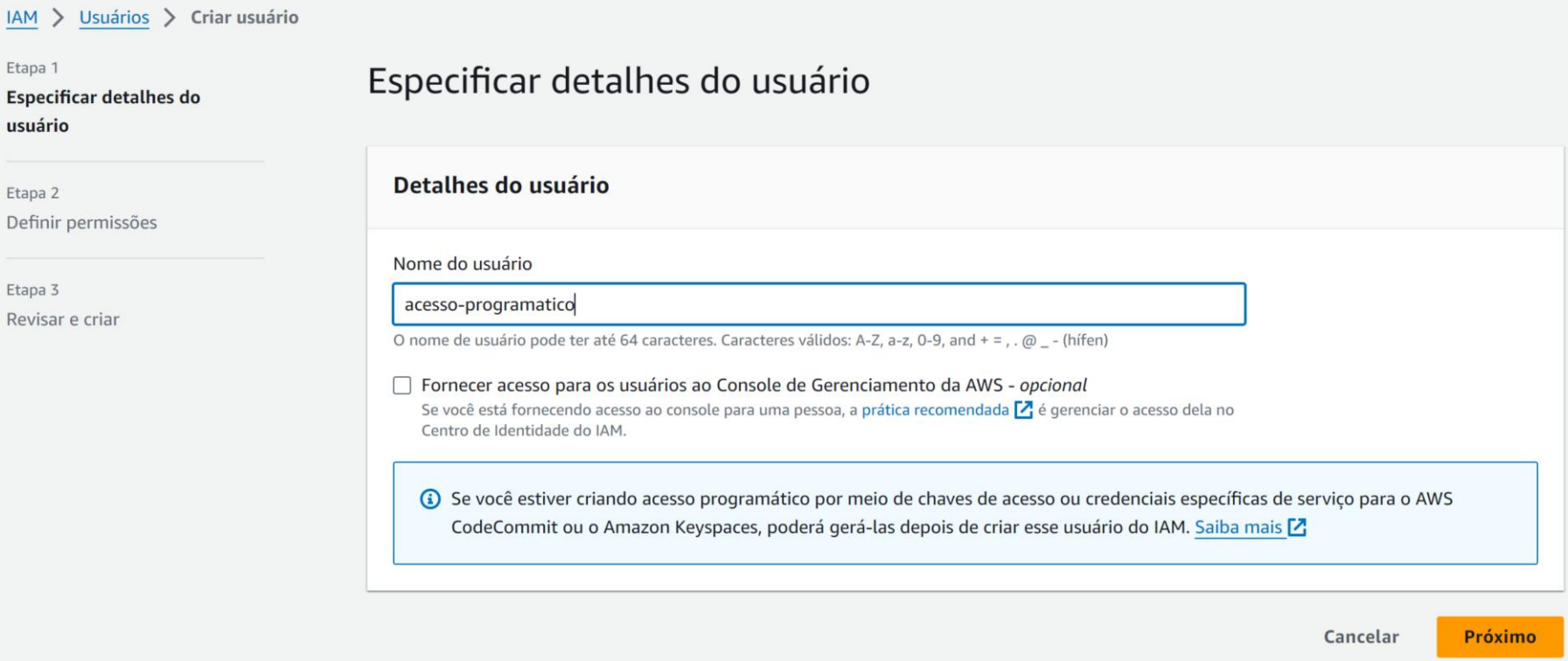
AWS IAM E CREDENCIAIS

Para acessar os recursos da AWS precisamos criar um usuário para acesso programático acessando o IAM.
Entre no Console, busque e entre no IAM.
Em “Usuários”, crie um usuário novo.



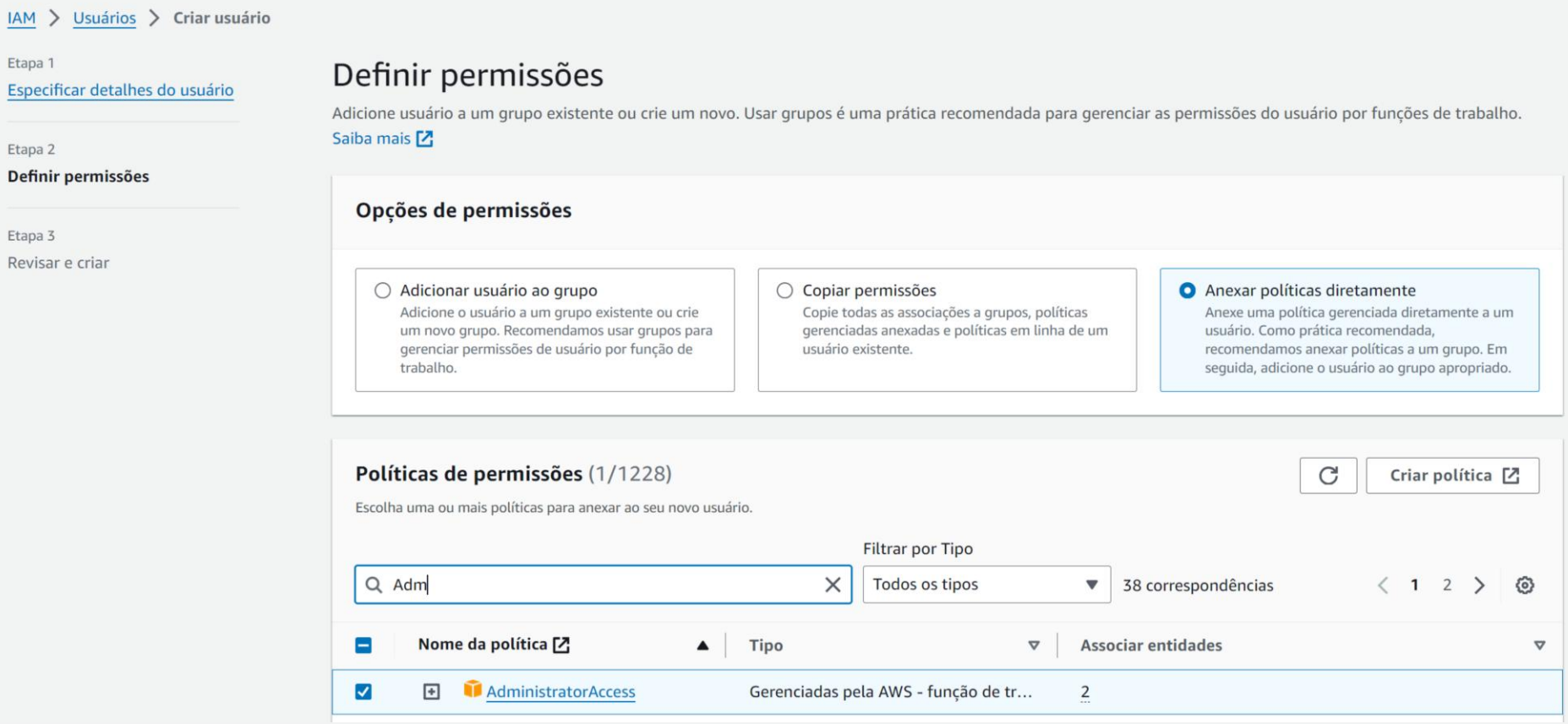
AWS IAM E CREDENCIAIS

Informe o nome do usuário.
Não marque acesso a console, pois esse usuário é apenas de uso programático.



AWS IAM E CREDENCIAIS

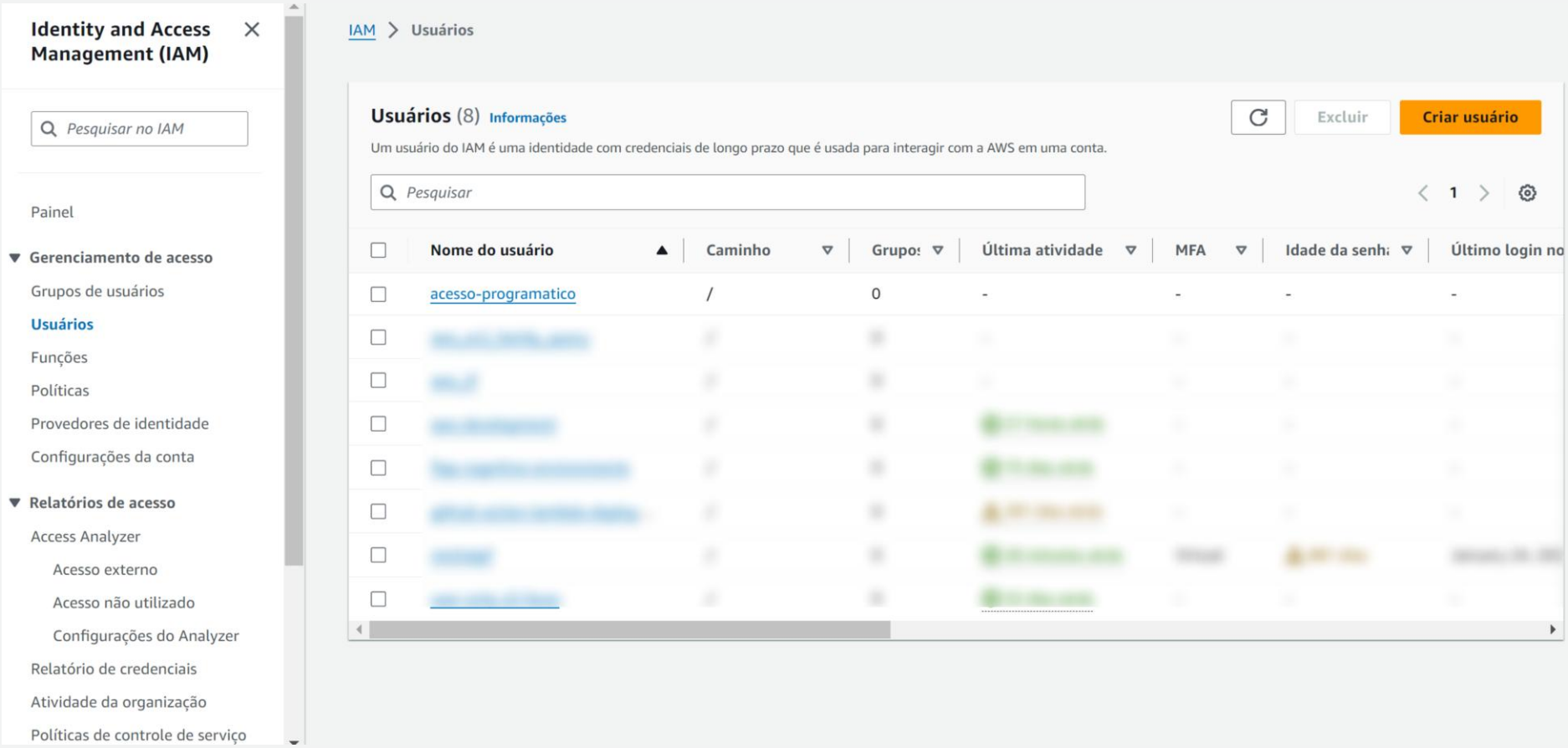
Clique em “Anexar políticas diretamente” e escolha a política “AdminstratorAccess”. E por fim, clicar em “Criar Usuário”.



Em sistemas produtivos devemos escolher exatamente quais recursos podemos ter acesso. Neste caso optamos o acesso administrador considerando o uso acadêmico e para simplificar o on-boarding.

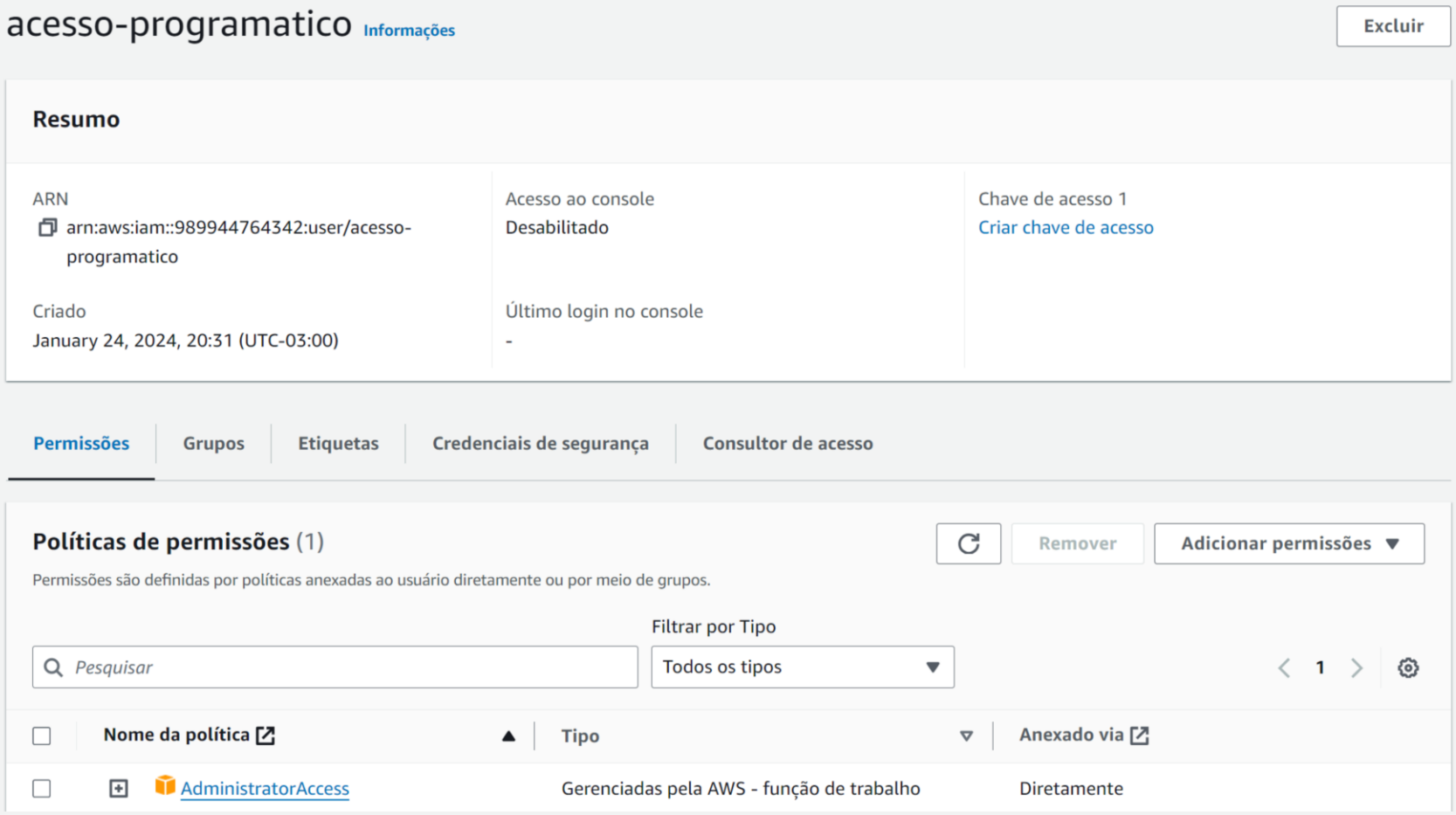
AWS IAM E CREDENCIAIS

Volte na tela principal e clique no usuário criado.



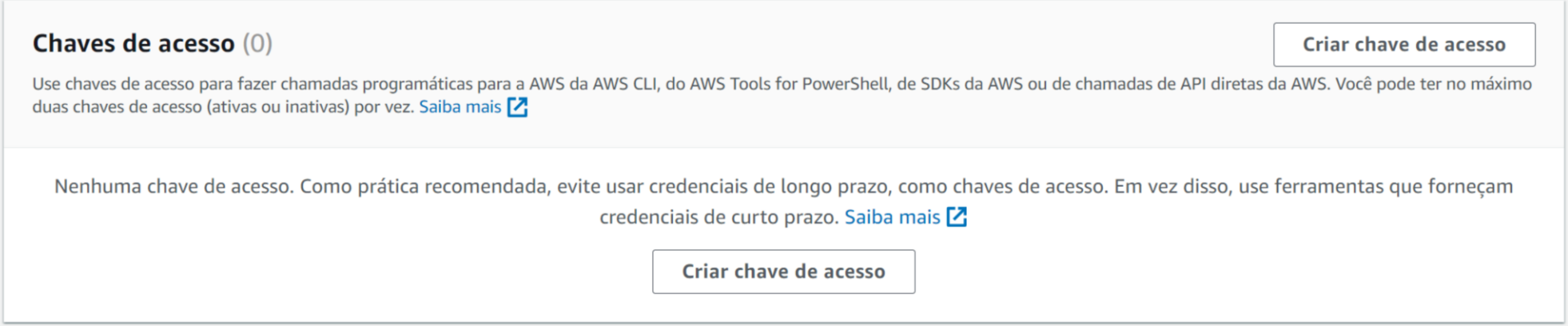
AWS IAM E CREDENCIAIS

Agora clique em “Credenciais de segurança”.



AWS IAM E CREDENCIAIS

Vá em “Chaves de acesso” e depois “Criar chave de acesso”.



AWS IAM E CREDENCIAIS

Escolha o caso de uso adequado, neste caso é para o CLI.

Práticas recomendadas e alternativas para chaves de acesso

Informações

Evite usar credenciais de longo prazo, como chaves de acesso, para melhorar sua segurança. Considere os seguintes casos de uso e alternativas.

Caso de uso

- (●) **Command Line Interface (CLI)**
  Você planeja usar essa chave de acesso para permitir que a AWS CLI acesse sua conta da AWS.
- ( ) **Código local**
  Você planeja usar essa chave de acesso para permitir que o código da aplicação em um ambiente de desenvolvimento local acesse a sua conta da AWS.
- ( ) **Aplicação em execução em um serviço computacional da AWS**
  Você planeja usar essa chave de acesso para permitir que o código da aplicação executado em um serviço computacional da AWS, como o Amazon EC2, o Amazon ECS ou o AWS Lambda, acesse a sua conta da AWS.

AWS IAM E CREDENCIAIS

Baixe o arquivo CSV das credenciais de acesso por segurança.

Recuperar chaves de acesso Informações

Chave de acesso

Se você perder ou esquecer sua chave de acesso secreta, não poderá recuperá-la. Em vez disso, crie uma nova chave de acesso e torne a chave antiga inativa.

Chave de acesso	Chave de acesso secreta
AKIA6M7KOK63G3BZYW6Y	********** Mostrar

Acesse as principais práticas recomendadas

- Nunca armazene sua chave de acesso em texto sem formatação, em um repositório de códigos ou em código.
- Desabilite ou exclua a chave de acesso quando não for mais necessária.
- Habilite permissões de menor privilégio.
- Alterne regularmente as chaves de acesso.

Para obter mais detalhes sobre o gerenciamento de chaves, consulte as Práticas recomendadas para o gerenciamento de chaves da AWS.

Baixar arquivo .csv | Concluído

AWS **CLI**

Para utilizar localmente o acesso a recursos da AWS precisamos instalar o AWS CLI (Command Line Interface).

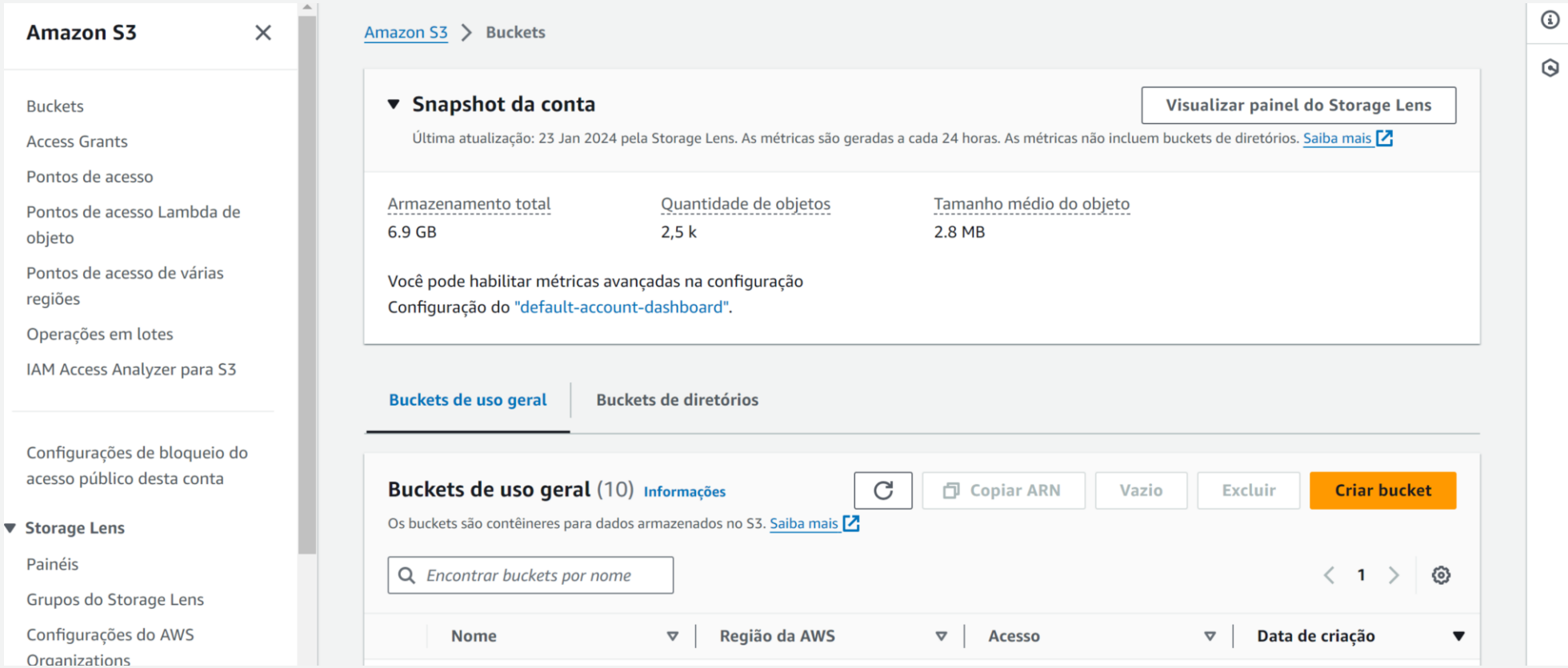
Depois que realizar a instalação precisamos configurar o CLI para acessar com as credenciais que foram criadas.

```
aws configure
AWS Access Key ID [None]: <access-key>
AWS Secret Access Key [None]: <secret-access-key>
Default region name [None]: us-east-1
Default output format [None]:
```

https://aws.amazon.com/pt/cli/

AWS S3: **CRIANDO** UM BUCKET

Os dados deverão estar em um bucket de S3, como se estivesse num Data Lake.
As atividades do cientista de dados vai ser processar estes dados e construir as features,
Portanto vamos criar 1 bucket (“Criar bucket”) para armazenar os dados em uma pasta específica “dataset”. E este mesmo bucket será utilizado pelo DVC para versionar o modelo e demais arquivos.



AWS S3: **CRIANDO** UM BUCKET

Deixe todas as opções padrão. Informe um nome e clique em “Criar bucket”.

Amazon S3 > Buckets > Criar bucket

Criar bucket Informações

Os buckets são contêineres para dados armazenados no S3. Saiba mais

Configuração geral

Região da AWS

Leste dos EUA (Norte da Virgínia) us-east-1

Tipo de bucket Informações

- (selecionado) **Propósito geral**
  Recomendados para a maioria dos casos de uso e padrões de acesso. Os buckets de uso geral são do tipo original do S3. Eles permitem uma combinação de classes de armazenamento que armazenam objetos de maneira redundante em várias zonas de disponibilidade.
- **Diretório** - *Novo*
  Recomendados para casos de uso de baixa latência. Esses buckets usam somente a classe de armazenamento do S3 Express One Zone, que fornece processamento mais rápido de dados em uma única zona de disponibilidade.

Nome do bucket Informações

myawsbucket

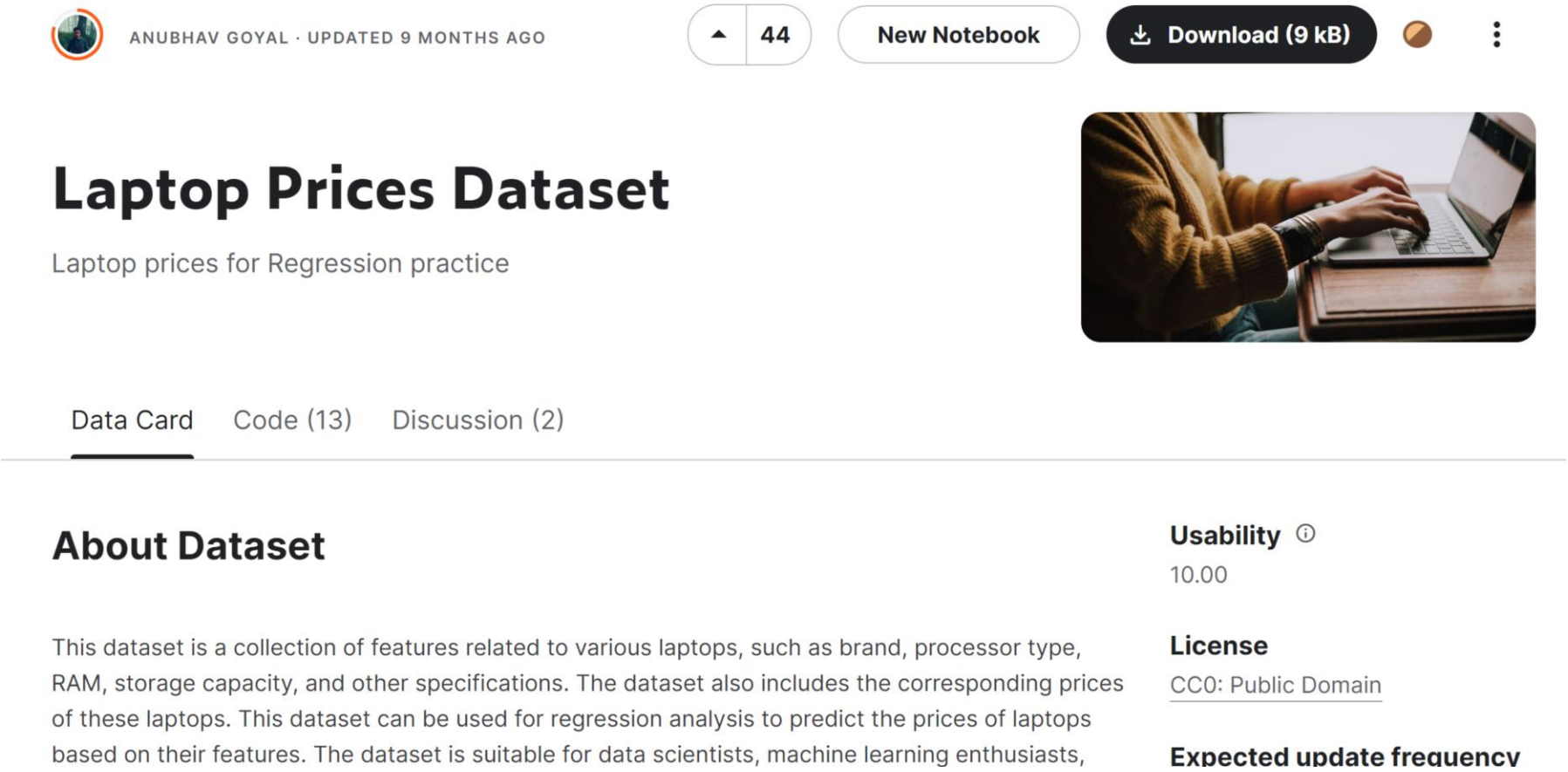
O nome do bucket deve ser exclusivo no namespace global e seguir as regras de nomenclatura do bucket. Veja as regras para nomenclatura de buckets

Copiar configurações do bucket existente - *opcional*

Somente as configurações de bucket na configuração a seguir são copiadas.

AWS S3: **UPLOAD**

Entre no bucket criado, clique em “Criar Pasta” e crie e pasta “dataset”.
Depois faça o upload do dataset que se encontra neste repositórido do Kaggle:
https://www.kaggle.com/datasets/anubhavgoyal10/laptop-prices-dataset



LAPTOP PRICING PREDICITON

Temos um desafio de negócio que consiste na compra de laptops usados. Para isso, será utilizado um modelo de predição de preço.

Esse modelo considerará diferentes atributos, como marca, processador, memória, armazenamento, entre outros.

Nossa primeira meta é versionar o projeto utilizando um template de machine learning e realizar um experimento inicial com um modelo simples baseado em regressão Ridge.

Os parâmetros mínimos de sucesso definidos são: $R^2 > 0,6$ (o modelo deve explicar pelo menos 60% da variância dos dados) e MAPE < 0,15 (erro percentual médio abaixo de 15%).

IMPORTÂNCIA DO **TEMPLATE**

O template que utilizamos tem uma convenção de que na pasta referente e dados processados e modelos eles não serão enviados no GitHub, mas sim no S3 da AWS.

Estas configurações de ignorar arquivos são importantes para manter os arquivos certos nos locais certos.

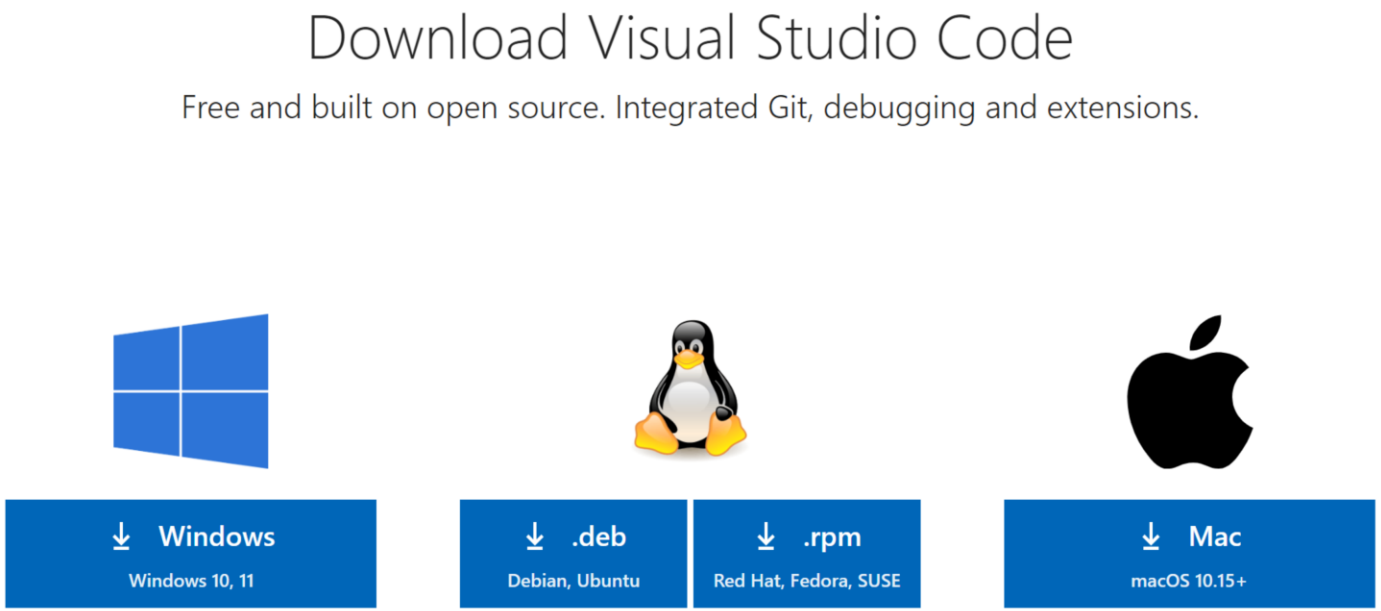
```
cookiecutter https://github.com/drivendata/cookiecutter-data-science

 [1/8] project_name (project_name): fiap-mlops-class-laptop-pricing
  [2/8] repo_name (fiap-mlops-class-laptop-pricing):
  [3/8] author_name (Your name (or your organization/company/team)): Michel Fernandes
  [4/8] description (A short description of the project.):
  [5/8] Select open_source_license
    1 - MIT
    2 - BSD-3-Clause
    3 - No license file
    Choose from [1/2/3] (1): 3
  [6/8] s3_bucket ([OPTIONAL] your-bucket-for-syncing-data (do not include 's3://')):
  [7/8] aws_profile (default):
  [8/8] Select python_interpreter
    1 - python3
    2 - python
    Choose from [1/2] (1): 1
```

VSCODE COMO IDE PADRÃO

Iremos utilizar o VS Code, pois a partir de agora trabalharemos com scripts e Notebooks que exigem controle sobre as versões dos pacotes utilizados.

O uso do Colab, por exemplo, implica em aceitar um ambiente gerenciado pelo Google, no qual não temos controle total sobre as versões do Python ou das bibliotecas instaladas, o que pode comprometer a reprodutibilidade dos experimentos.

Baixe o VSCode neste link: https://code.visualstudio.com/download



PYENV GERENCIE VERSÕES DO PYTHON

Dependendo os projetos que trabalhamos é comum utilizarmos diferentes versões do Python, seja por compatibilidades com plataformas de Cloud ou até mesmo bibliotecas.

Gerenciar manualmente diferentes versões do Python pode dar trabalho e gerar incompatibilidades, por isso fica a recomendação de utilizar o Pyenv (https://github.com/pyenv/pyenv) .

Com o Pyenv controlamos qual a versão Global do Python e também podemos instalar por ele mesmo versões diferentes.

Oficialmente não há suporte para Windows, mas há uma versão alternativa que pode ser utilizada por meio deste link: https://github.com/pyenv-win/pyenv-win/blob/master/docs/installation.md

```
pyenv install –l // Exibe as versões disponíveis para instalação
pyenv install 3.10.8 // Instala a versão 3.10.8
pyenv global 3.10.8 // Define a versão global como 3.10.8
```

Na disciplina vamos utilizar o Python 3.10 como padrão, portanto não há problema em baixar diretamente esta versão e utilizá-la.

VSCODE COMO IDE PADRÃO

Vamos abrir o VSCode para começarmos a trabalhar com o projeto.
Utilize a linha de comando abaixo para entrar no VSCode.

```
cd <pasta-do-projeto>
code .
```

Instale as seguintes extensões que ajudam no desenvolvimento:

1. Python
2. Jupyter
3. Docker
4. Gitignore
5. Github Actions

ARQUIVOS DE **DEPENDÊNCIAS**

Vamos utilizar o arquivo requirements.txt para listar as bibliotecas necessárias ao desenvolvimento do modelo, à exploração do dataset e à sincronização com repositórios locais e de experimentos.

```
black==25.1.0
flake8==7.2.0
isort==6.0.1
mkdocs==1.6.1
pytest==8.3.5
python-dotenv==1.1.0
notebook==7.4.2
pandas==2.2.3
numpy==2.2.5
scikit-learn==1.6.1
ipywidgets==8.1.7
ipykernel==6.29.5
dagshub==0.5.8
mlflow==2.22.0
dvc-s3==3.2.0
xgboost==3.0.0
lightgbm==4.0.0
catboost==1.2.8
```

As dependências podem variar conforme a evolução do projeto, especialmente quando novas bibliotecas forem adicionadas para criação, ajuste ou avaliação de modelos.

Utilizar bibliotecas sem versionamento no início do projeto facilita o gerenciamento de dependências e a identificação das versões mais compatíveis entre si. No entanto, uma vez que as bibliotecas estejam estáveis e validadas, é recomendado travar suas versões no requirements.txt para evitar problemas de compatibilidade em futuras execuções ou atualizações do ambiente.

INSTALANDO **AMBIENTE VIRTUAL**

Primeiro passo antes de começarmos a desenvolver nossos experimentos, será de construir um ambiente virtual que possua todos os pacotes que iremos trabalhar.

```
python -m venv .venv
.venv/scripts/activate ou source .venv/bin/activate para MacOS/Linux
```

Depois de instalar o ambiente virtual você deve notar o ambiente ativo por meio da linha de comando com o nome do diretório do ambiente.

```
PS D:\GitHub_Projects\fiap-ds-mlops-laptop-pricing> .\.venv\Scripts\activate
(.venv) PS D:\GitHub_Projects\fiap-ds-mlops-laptop-pricing>
```

Agora já podemos instalar as bibliotecas que iremos utilizar.
A medida que novas bibliotecas sejam necessárias, iremos adicioná-las e executar novamente o comando.

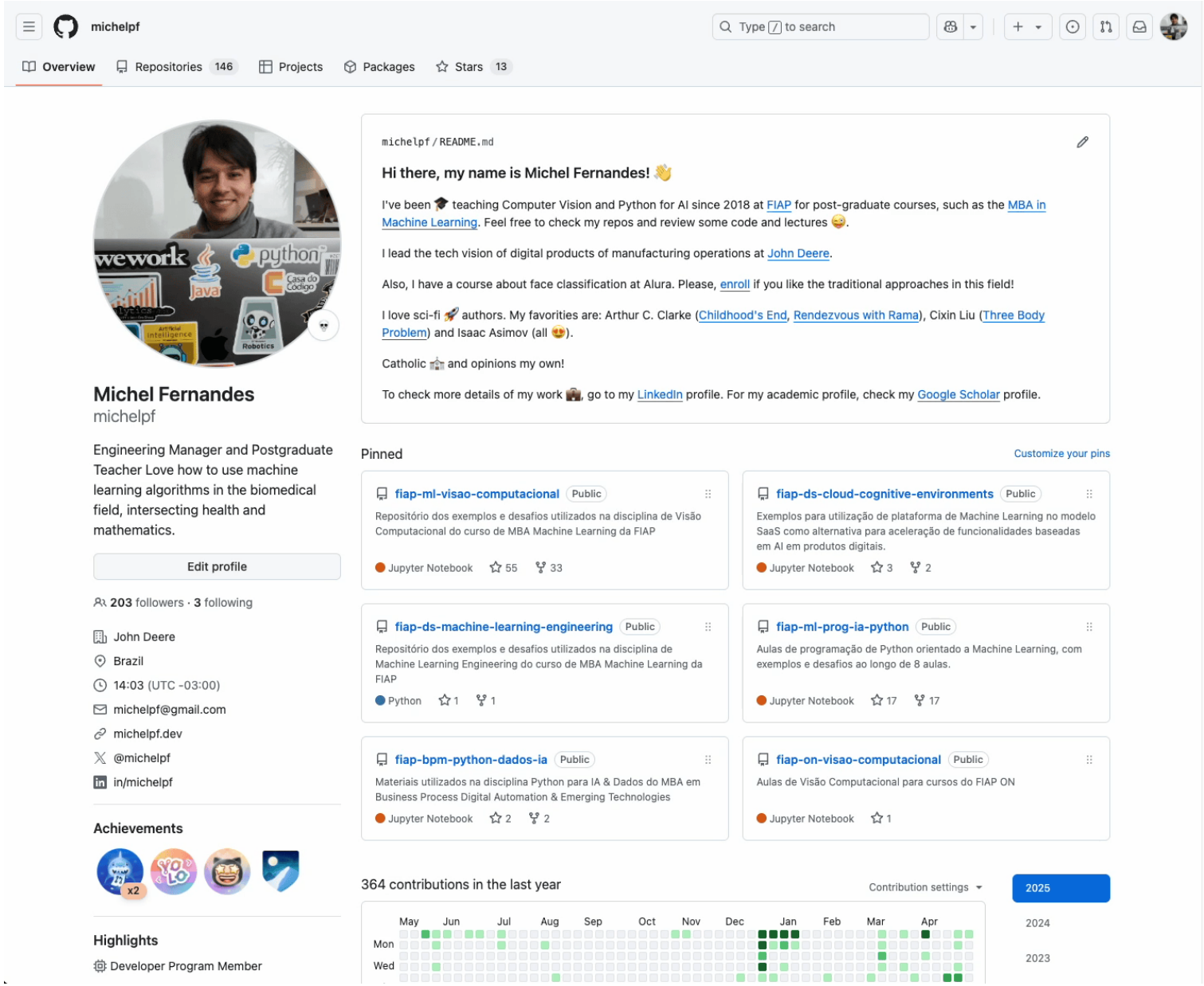
```
PS D:\GitHub_Projects\fiap-ds-mlops-laptop-pricing> .\.venv\Scripts\activate
(.venv) PS D:\GitHub_Projects\fiap-ds-mlops-laptop-pricing> pip install –r requirements.txt
```

CONFIGURANDO **GITHUB**

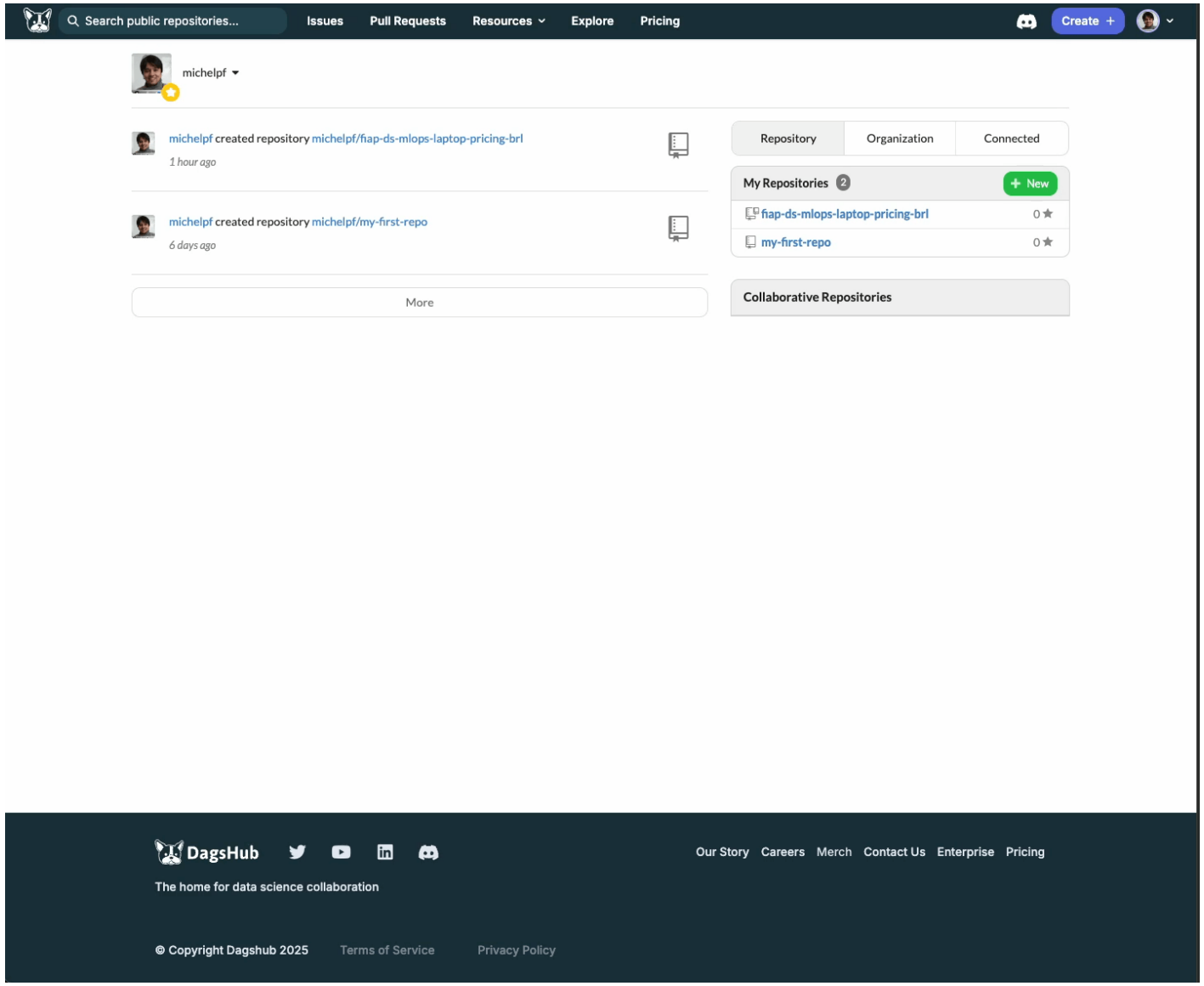
Primeiro vamos criar um novo repositório no GitHub.

Após isso podemos sincroniza-lo com o um repositório no DagsHub.

Por fim, sincronizaremos com o DVC utilizando o storage do DagsHub ou no S3.



CONFIGURANDO **DAGSHUB**



Agora com os repositórios criados, vamos carregar nosso template para depois iniciarmos o DVC.

CONFIGURANDO **GIT**

FIAP MBA+

Na pasta local vamos iniciar o repositório.

```
git init
```

Logo após precisamos indicar o repositório remoto que é o endereço dele no GitHub.

```
git remote add origin <endereço do repo>
```

Após isso adicionamos os arquivos e sincronizamos remotamente.

```
git add .
```

Confirmamos nossas modificações.

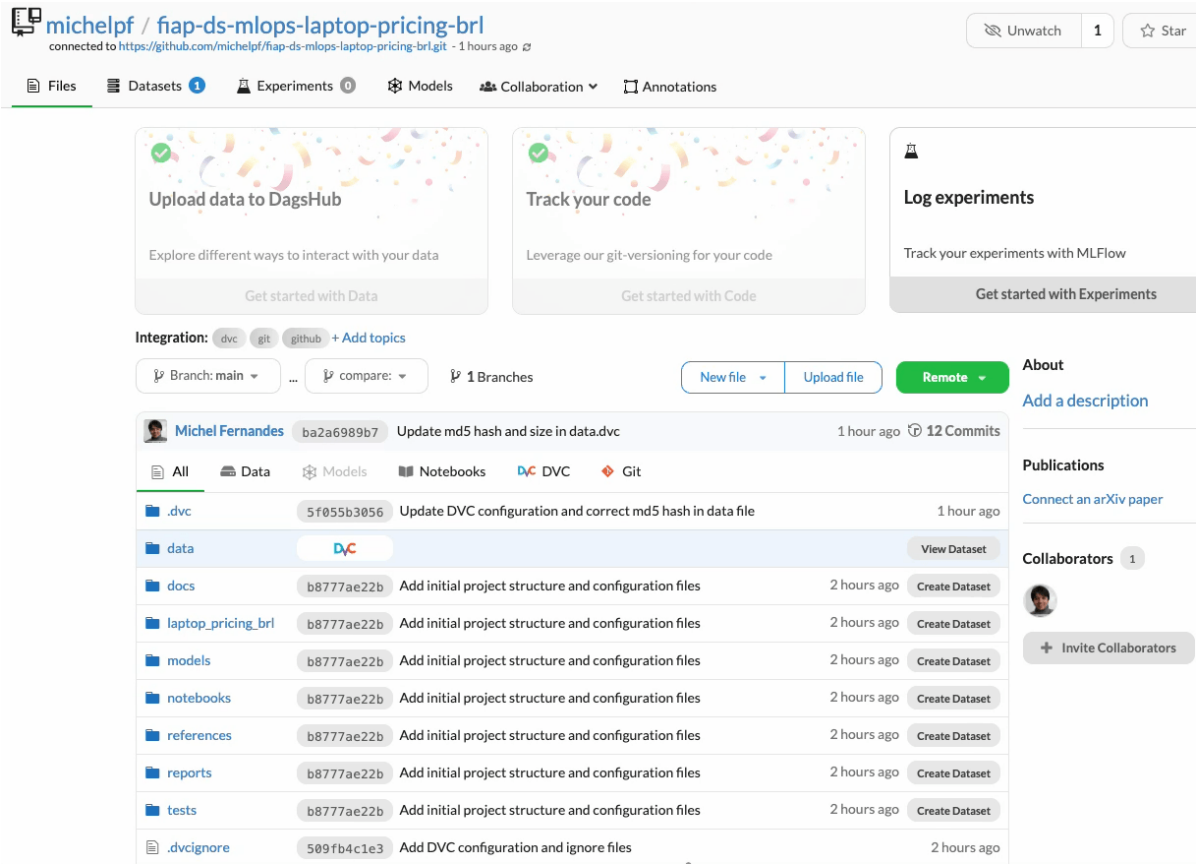
```
git commit –am “importação inicial”
```

E finamente sincronizamos.

```
git push origin main
```

CONFIGURANDO **DVC**

Agora que temos no DagsHub o repositório devidamente sincronizado, precisamos obter as credenciais do DagsHub storage que utiliza o DVC para sincronizar.



E então podemos iniciar o DVC.

```
dvc init
```

Adicionar o remote do DVC.

```
dvc remote add origin s3://dvc
dvc remote modify origin endpointurl url-do-dagshub
```

Adicionar credencais do DVC.

```
dvc remote modify origin --local access_key_id key
dvc remote modify origin --local secret_access_key
secret
```

Adicionar a pasta para o controle de versões do DVC

```
dvc add data
```

A partir de agora toda a mudança na pasta data precisa ser sincronizada com o repositório.

CONFIGURANDO **DVC**

Mais comandos adicionais.

Adicionar o controle de mudanças no Git para associarmos as mensagens de commit no DVC

```
git add data.dvc
```

Habilitar o auto-staging no DVC, ou seja adiciona no Git qualquer novo arquivo DVC.

```
dvc config core.autostage true
```

CONFIGURANDO **DVC**

Para criar um repositório local, basta inicializarmos ele com o comando abaixo:

```
git init
```

Vamos criar um repositório no Github e associar como repositório remoto:

```
git remote add origin <endereço do repo>
```

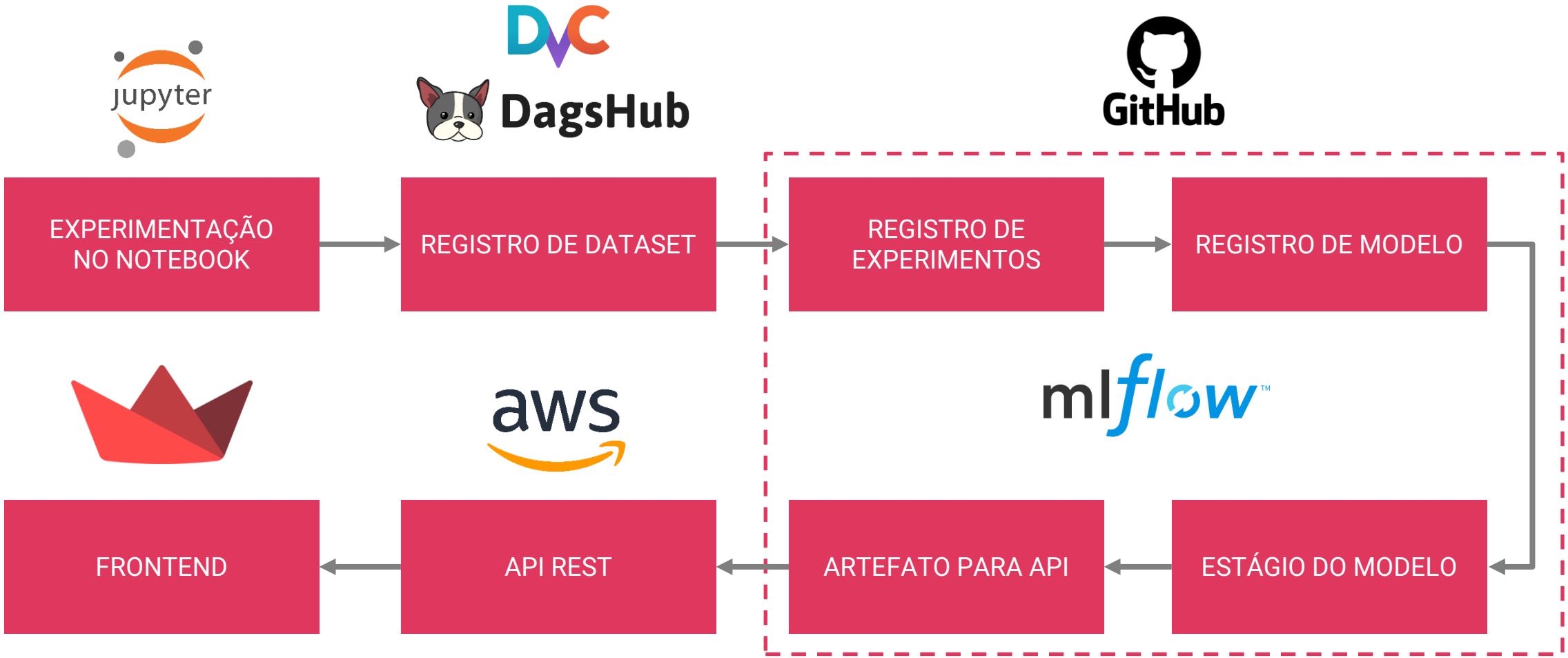
E agora inicializar o DVC.

```
dvc init
```

Logo após vamos adicionar um “remote” para os artefatos (arquivos grandes) no S3 da AWS.
Como já configuramos o AWS CLI, essa permissão será utilizada para sincronizar os arquivos com esse remote.

```
dvc remote add -d myremote s3://<bucket>/<key>
```

CICLO PARA CONSTRUIR UM MODELO



EXPLORANDO **DATASET E PRÉ-PROCESSANDO**

O dataset adaptado para uso em BRL está disponível neste repositório: https://github.com/michelpf/dataset-laptop-pricing.

Vamos baixá-lo e armazená-lo na pasta data/raw. A estrutura da pasta data será utilizada para organizar os dados, separando os brutos (raw) dos dados processados (processed).

Exploração Inicial e Processamento de Dados

Verificação e pré-processamento de dados para utilização de modelos.

```python
import pandas as pd
import numpy as np
import dagshub
from dagshub import upload_files
```

Python

```python
df = pd.read_csv("../data/raw/laptop-price-brl.csv")
df.head()
```

Python

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb	weight	warranty	Touchscreen	msoffice	Pri
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	23
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	26
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	26
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	32-bit	2 GB	Casual	No warranty	No	No	46
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	18

Nosso primeiro desafio será realizar o pré-processamento dos dados, padronizando e preparando o conjunto para o treinamento dos modelos.

Para essa atividade, vamos criar nosso primeiro notebook, chamado data-processing.ipynb, que ficará na pasta notebooks.

ENVIANDO DADOS PARA **DVC & DAGSHUB**

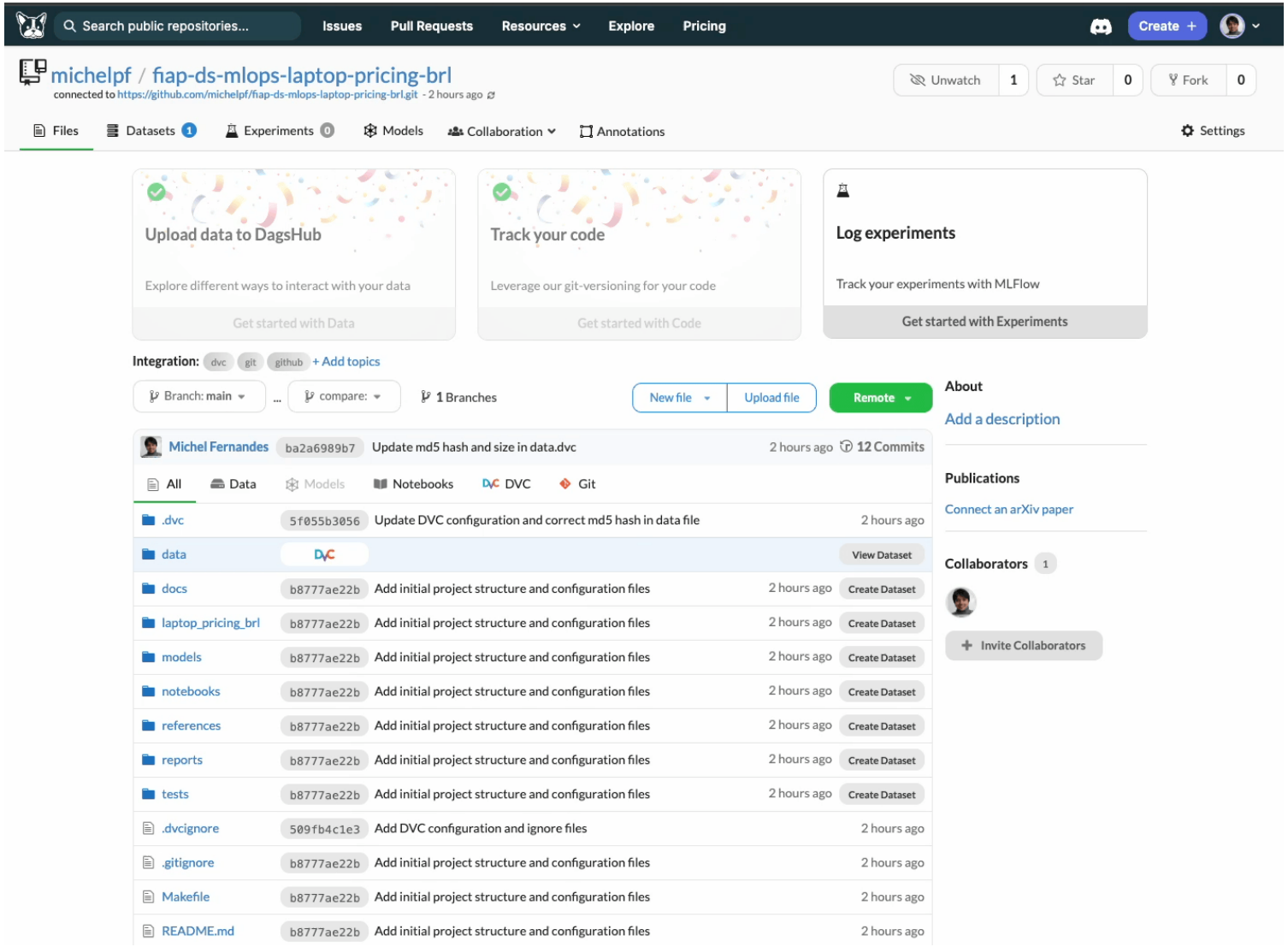
Após os arquivos terem sido gerados nas pastas “raw” e “processed” vamos fazer o seguinte:

```
dvc commit
```

Enviar dados para o repositório remoto toda a vez que houver modificações.

```
dvc remote default origin
dvc push
```

CRIANDO UM DATASET NO **DAGSHUB**



Com os datasets armazenados no DagsHub conseguimos concentrar uma única versão em diversos experimentos.

UTILIZANDO UM DATASET NO **DAGSHUB**

FIAP MBA+

Load datasource

Download this notebook | Open in Colab

Use the processed datasource

```
# Install the DagsHub client
!pip install -q dagshub

import os
os.environ["DAGSHUB_CLIENT_HOST"] = "https://dagshub.com"
```

```
from dagshub.data_engine import datasources
ds = datasources.get('michelpf/fiap-ds-mlops-laptop-pricing-brl', 'processed')
```

Query your data

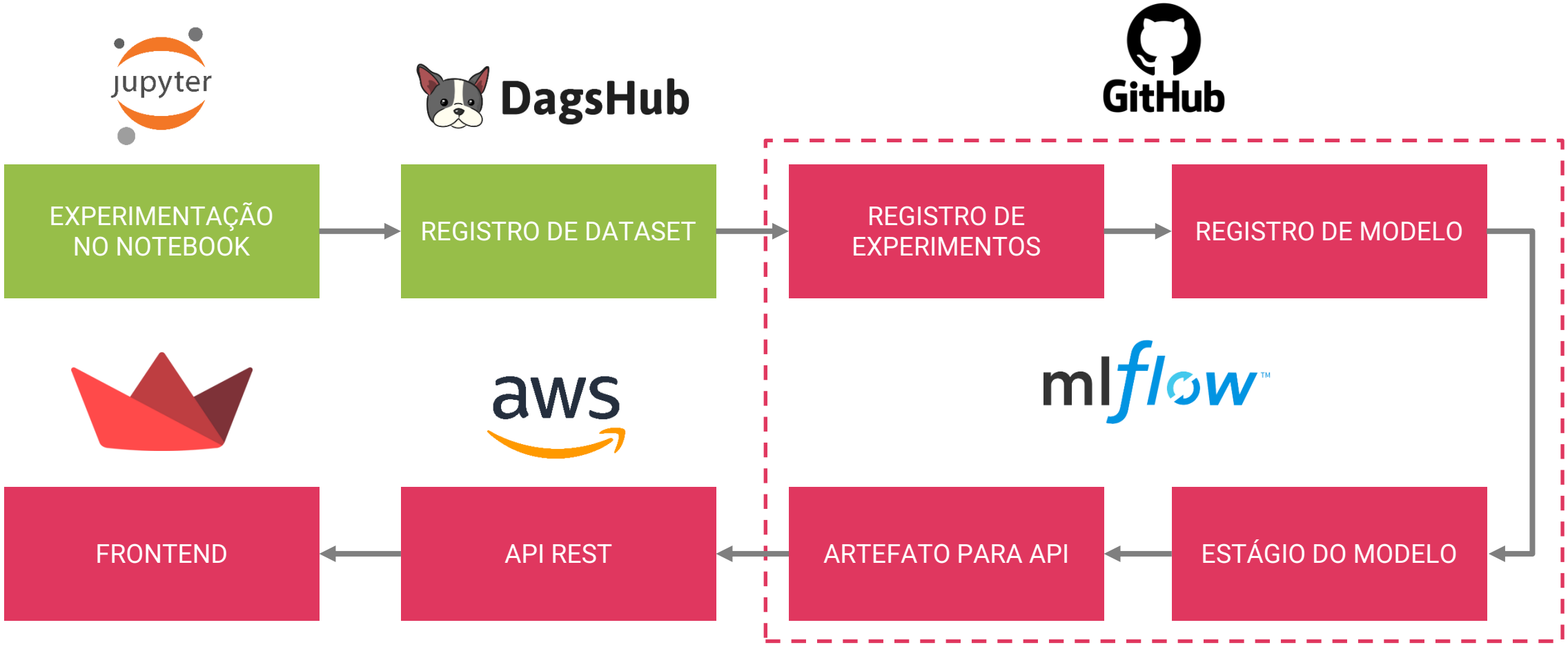
Datasources can be filtered using a pandas-like syntax.

For the full list of possible filters, see the docs.

```
# Example for simple query: get all data points with size larger than 1 byte
newQuery = ds["size"] > 1

# Display query results as data frame
newQuery.all().dataframe
```

	path	datapoint_id	dagshub_download_url	size
1	file0.png	41925882	https://dagshub.com/api/v1/repos/user/reponame...	116656
2	file1.jpeg	41925883	https://dagshub.com/api/v1/repos/user/reponame...	142949
3	file2.png	41925884	https://dagshub.com/api/v1/repos/user/reponame...	2982592
0	file3.gif	41925881	https://dagshub.com/api/v1/repos/user/reponame...	569995

CICLO PARA CONSTRUIR UM MODELO



DESAFIO 1

Elaborar a versão inicial do notebook de exploração e pré-processamento dos dados para posterior treinamento.

PARA SABER **MAIS**

https://dvc.org/doc, documentação da DVC e de todos os produtos que utilizaremos (DVC, DVCLive, Studio e GTO)
https://youtube.com/playlist?list=PL7WG7YrwYcnDBDuCkFbcyjnZQrdskFsBz&si=rI1SrP6S06Kc_OmS, Playlist de vídeos sobre MLOps da DVC

FIAP MBA+

MACHINE LEARNING ENGINEERING

MBA EM DATA SCIENCE & IA

RASTREAMENTO DE EXPERIMENTOS

EXPERIMENTOS COM **NOTEBOOK**

Antes de formalizar o experimento num script, ou seja, codifica-lo para que seja reproduzível em esteiras de CD/CI precisamos de um ambiente onde seja mais adequado atividades de:

- Exploração de dados
- Pré-processamento de dados
- Construção de features

A execução no VSCode garante que todas as bibliotecas estão condidas em dependências mapeadas.

Próximo passo é converter em scripts os passos citados anteriormente incluindo o treinamento.

Criamos uma branch nova para estas modificações

git checkout –b exploracao-inicial

Modelo de predição de preço de laptops

Os dados foram obtidos no dataset de Anubhal Goyal, disponibilizando no Kaggle por meio deste link. O arquivo original já foi baixado na pasta `data/raw`

```python
import pandas as pd
import numpy as np

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

%matplotlib inline

import matplotlib.pyplot as plt
```

[1] ✓ 0.9s Python

```python
df = pd.read_csv("../data/raw/laptopPrice.csv")
df.head()
```

[3] ✓ 0.0s Python

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb	weight	warranty	Touchscreen	msoffice
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	32-bit	2 GB	Casual	No warranty	No	No
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No

DESENVOLVENDO O **PRÉ-PROCESSAMENTO**

Antes de começarmos a experimentar diferentes tipos de modelos, precisamos identificar e selecionar os atributos mais relevantes, ou seja, realizar o processo de feature engineering.

Após essa etapa, registraremos tanto o dataset original quanto as versões pré-processadas, garantindo organização e rastreabilidade para os próximos passos do projeto.

Exploração Inicial e Processamento de Dados

Verificação e pré-processamento de dados para utilização de modelos.

```python
import pandas as pd
import numpy as np
import dagshub
from dagshub import upload_files
```

[1] Python

```python
df = pd.read_csv("../data/raw/laptop-price-brl.csv")
df.head()
```

[2] Python

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb	weight	warranty	Touchscreen	msoffice	Price	rating	Number of Ratings	Number of Reviews
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	2321	2 stars	3	0
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	2613	3 stars	65	5
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	2680	3 stars	8	1
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	32-bit	2 GB	Casual	No warranty	No	No	4689	3 stars	0	0
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	64-bit	0 GB	Casual	No warranty	No	No	1808	3 stars	0	0

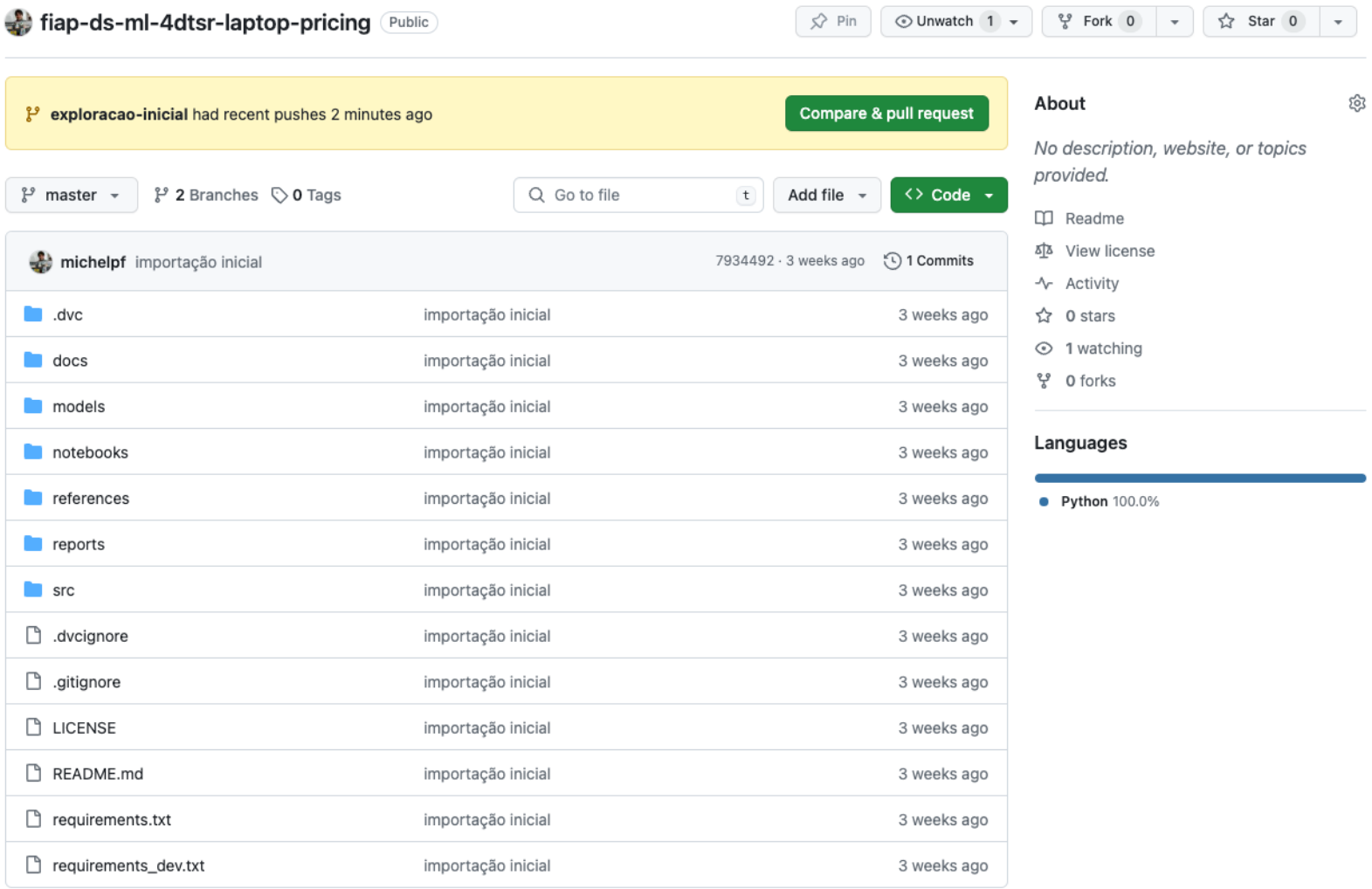
Pré-processamento

Padronização dos dados, consistindo nas etapas de:

Este notebook fica na pasta “notebooks” identificado como data-processing.

SUBMETENDO O **PULL-REQUEST**

Clicar em “Compare & pull request”.



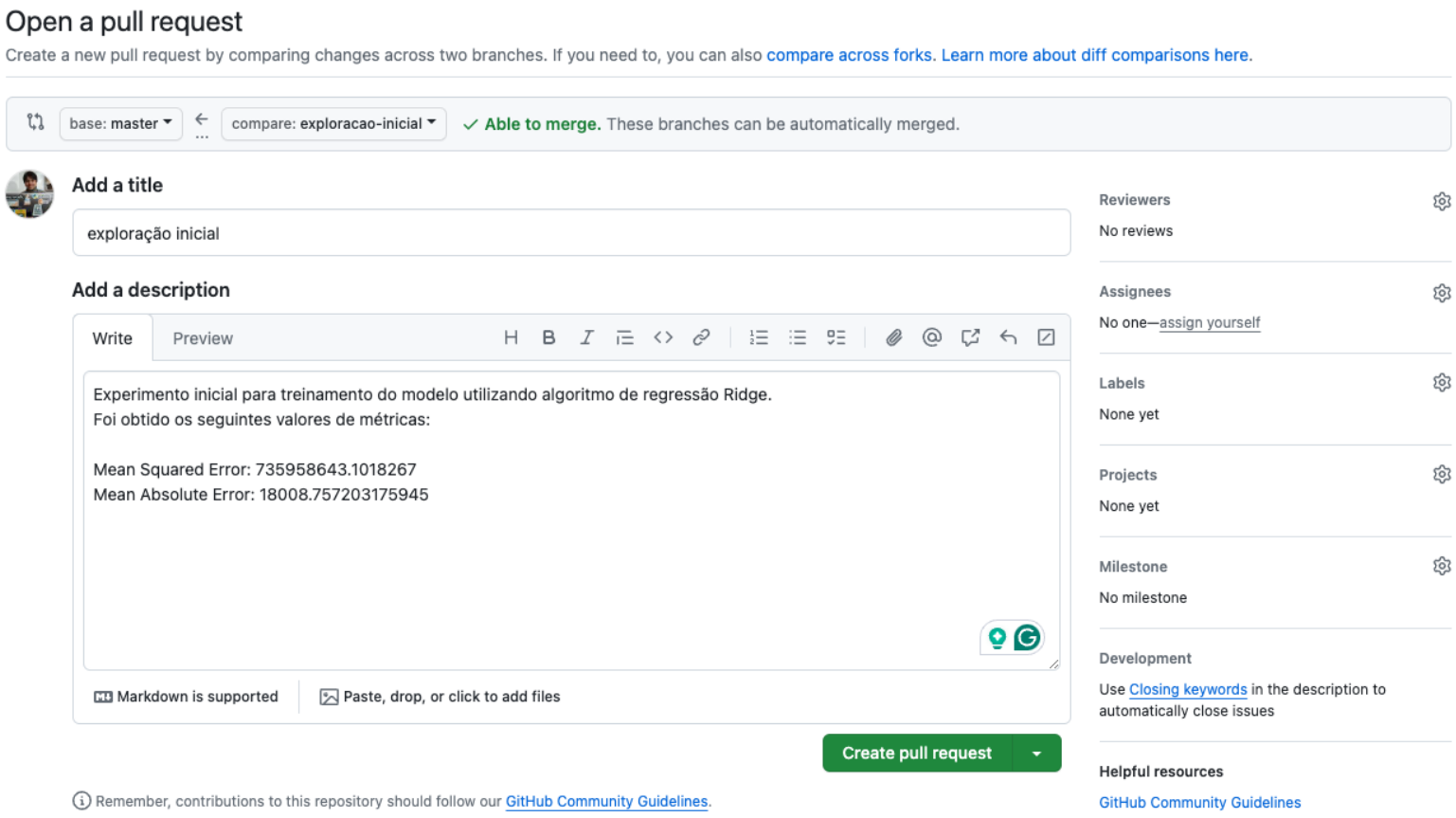
Podemos fazer as modificações na master? Sim.
Devemos? Não, pois o branch master precisa estar protegido e bem documentado quanto a qualquer alteração.

Somente trechos de código e modificações validadas e revisadas podem ser combinadas com a versão principal.

Isso se deve pois esta ramificação é que entregará o modelo em produção.

JUSTIFICANDO O **PULL REQUEST**

Precisamos convencer o revisor do repositório/projeto a aceitar as modificações. A depender do que se propõe modificar, valores de métricas são boas referências.

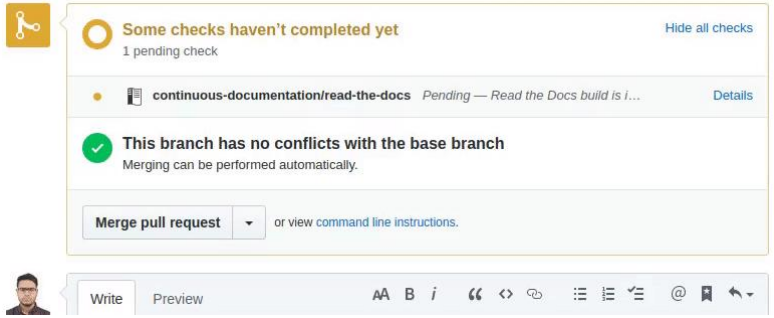


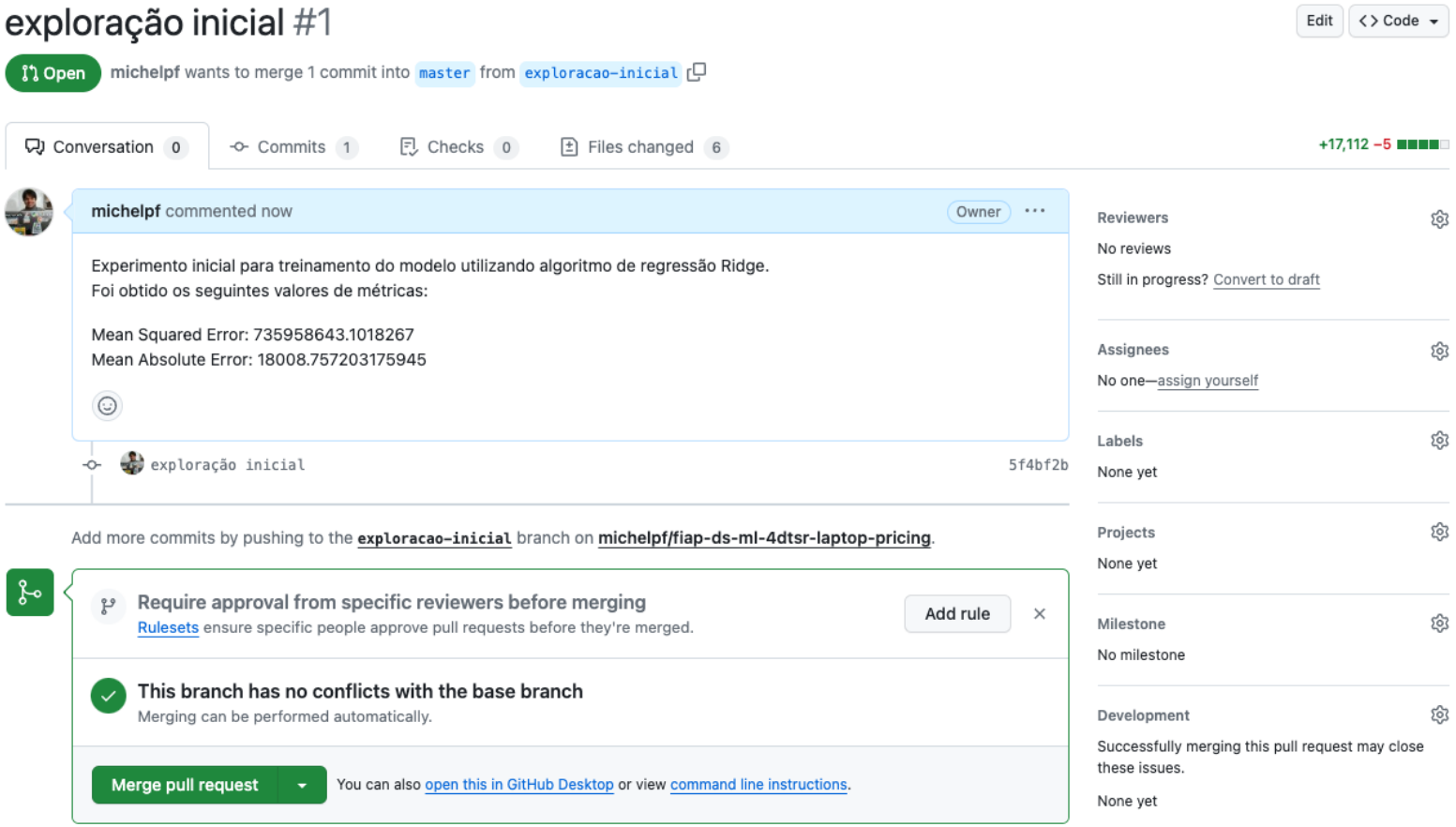
- Comparações antes e depois
- Referências das razões das escolhas
- Utilize os recursos disponíveis como imagens e anexos

VALIDAÇÕES E **TESTES**

Após submeter o Pull Request, se houver alguma ação personalizada, como testes de unidade, eles poderão ser executados e automaticamente invalidar sem a revisão.

Esta automação também pode incluir informações adicionais.

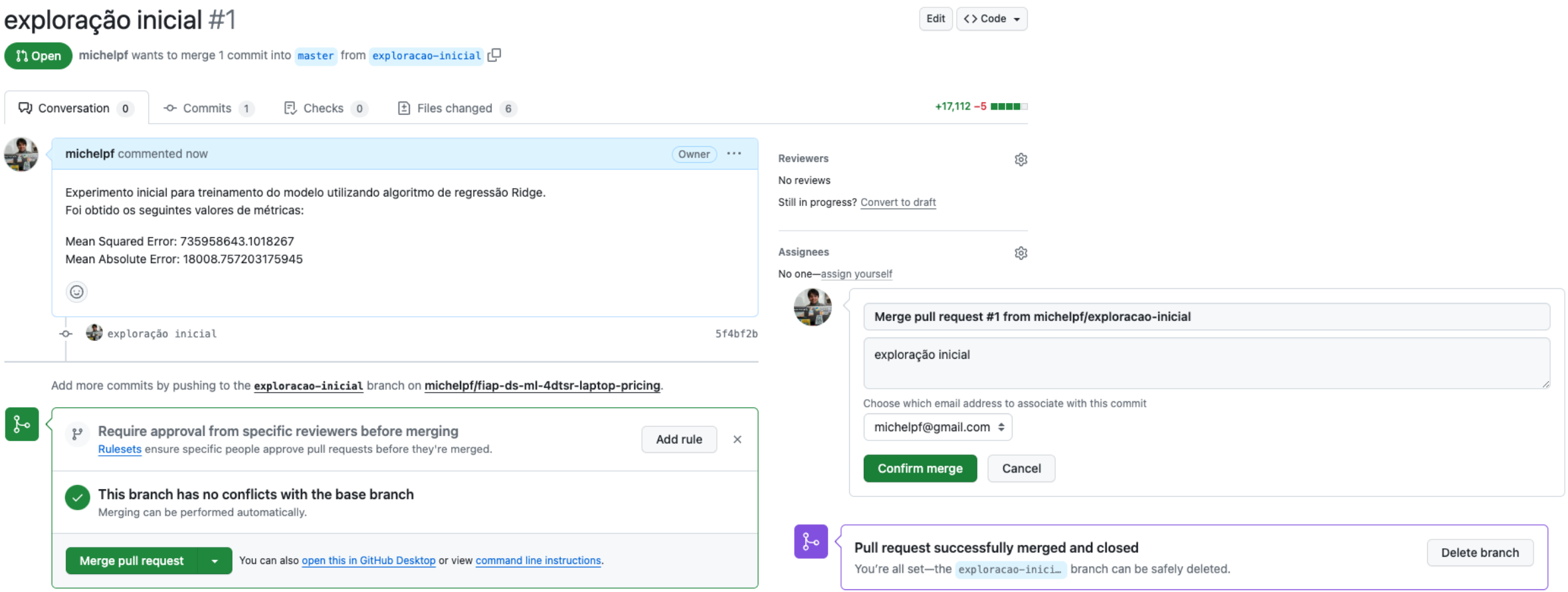




Com sinal verde, ou seja, todos os testes realizados com sucesso, é possível avançar com “merge” que irá combinar a branch “exploracao-inicial” com a “master”.

FINALIZANDO O **PULL REQUST**

Para finalizar o “Merge” confirmamos as alterações e também removemos a branch “Delete branch”.



ATUALIZANDO O **CÓDIGO**

FIAP MBA+

De volta ao nosso código local, vamos alterar a branch para a master, atualizá-la e depois remover a branch auxiliar.

Mudar a branch para master.

```
git checkout master
```

Obter as últimas modificações depois do merge do Pull Request

```
git pull origin master
```

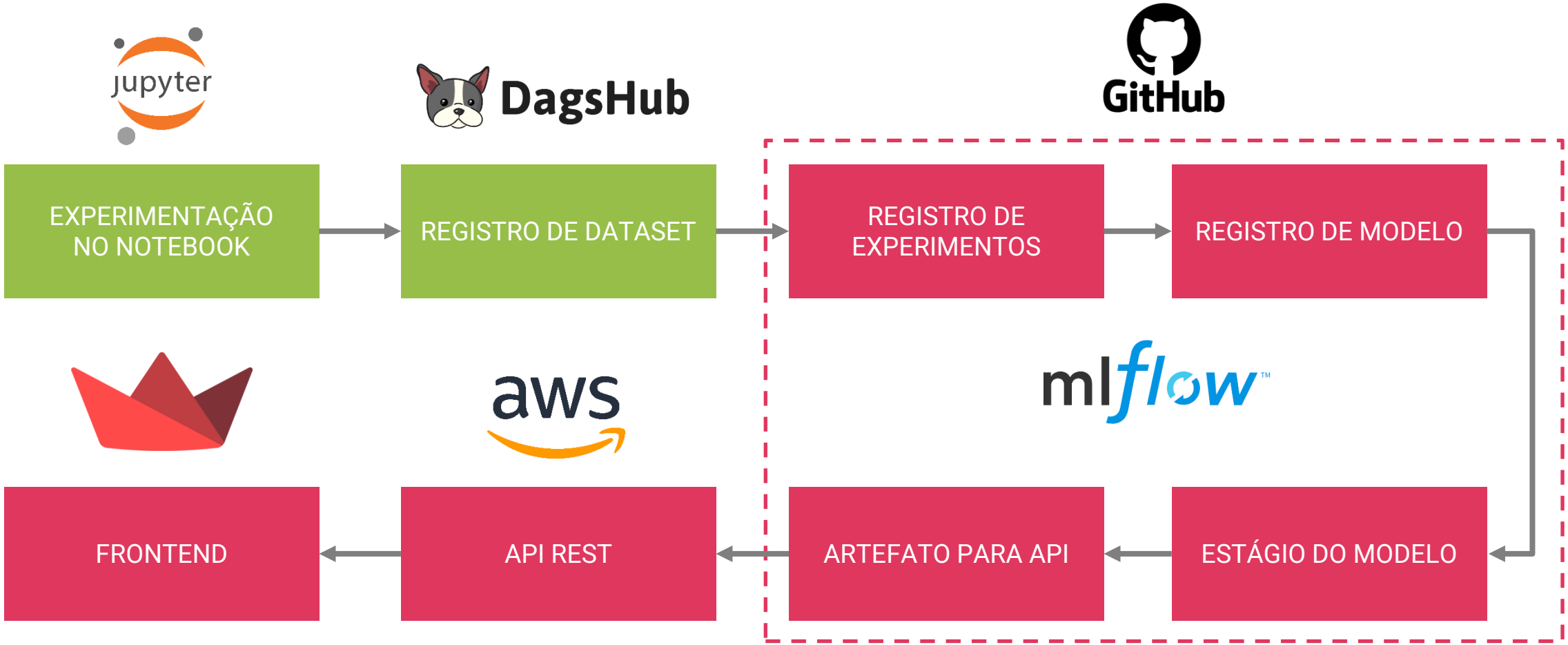
Remover a branch de testes

```
git branch –d exploracao-inicial
```

Caso as modificações não sejam sincronizadas pela main, o próprio Git informará que a branch a ser removida contém informações não sincronizadas, como um alerta.

EXPERIMENTOS COM MLFLOW

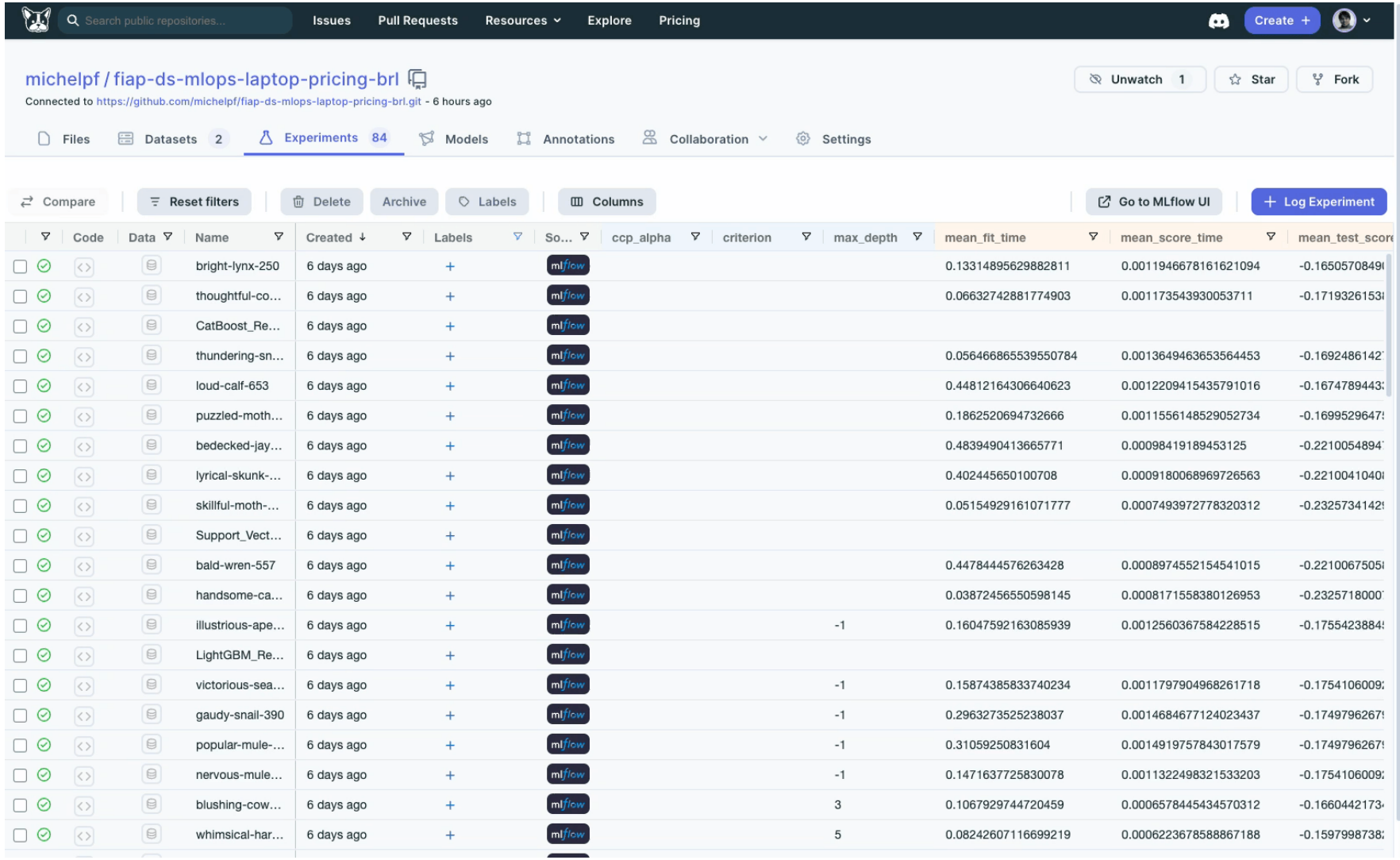
CICLO PARA CONSTRUIR UM MODELO



RASTREANDO **EXPERIMENTOS**

Os experimentos armazenam informações importantes durante o desenvolvimento do modelo, como métricas de desempenho, metas definidas e detalhes da execução dos treinamentos. Esses registros facilitam o acompanhamento da evolução e a comparação entre diferentes abordagens.

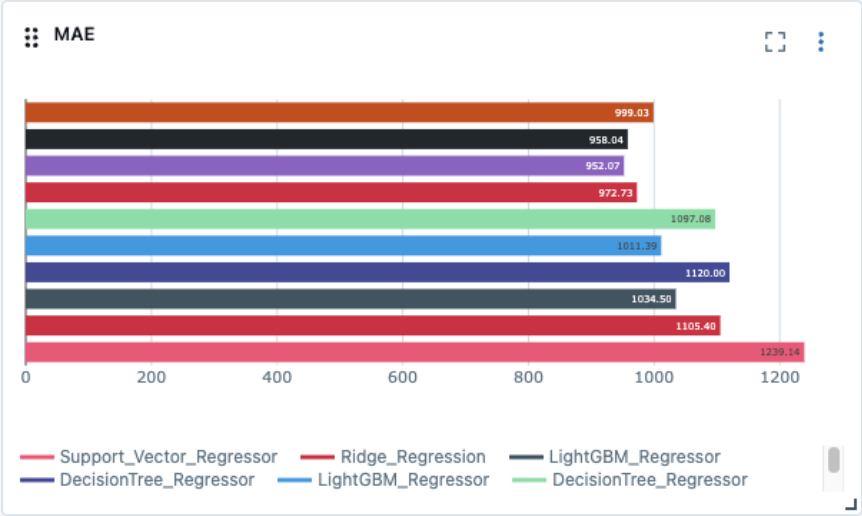
Embora o DagsHub ofereça um painel próprio para experimentação, optaremos por utilizar a interface visual do MLflow, que oferece mais recursos, especialmente para o registro de modelos e o download de artefatos gerados durante os experimentos.

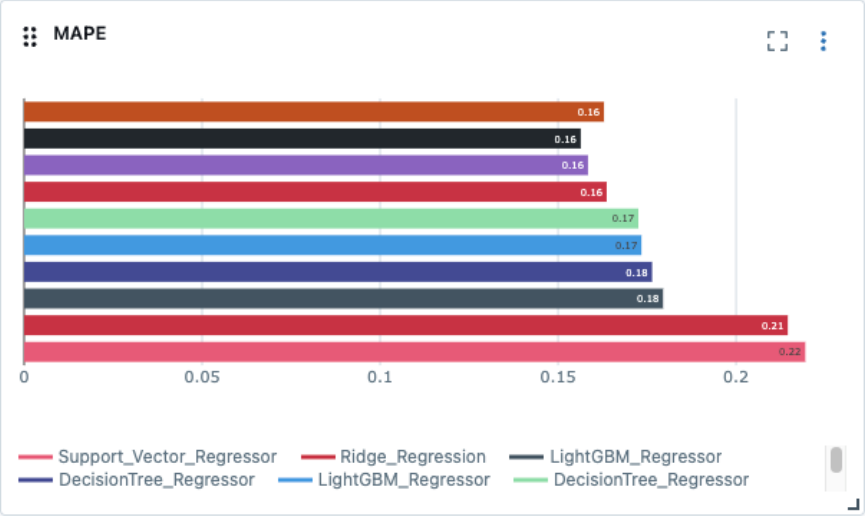


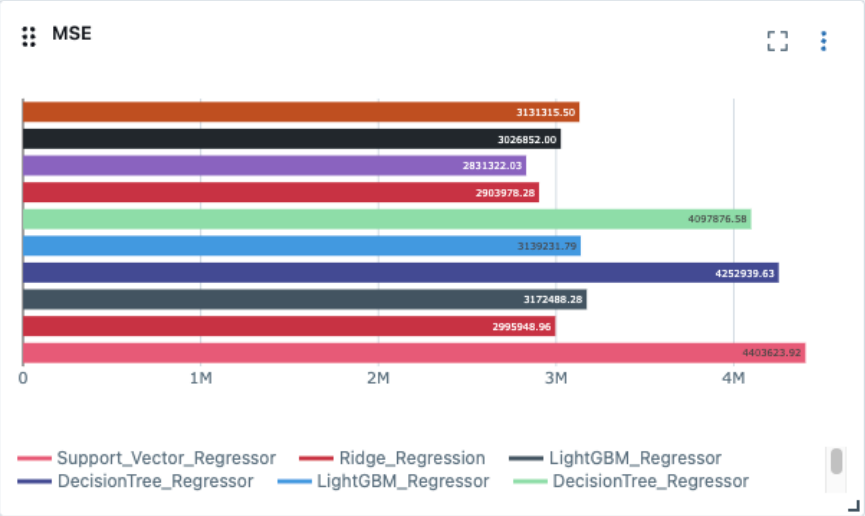
Code	Data	Name	Created	Labels	So...	ccp_alpha	criterion	max_depth	mean_fit_time	mean_score_time	mean_test_score
		bright-lynx-250	6 days ago	+	mlflow				0.13314895629882811	0.0011946678161621094	-0.16505708496
		thoughtful-co...	6 days ago	+	mlflow				0.06632742881774903	0.001173543930053711	-0.17193261538
		CatBoost_Re...	6 days ago	+	mlflow						
		thundering-sn...	6 days ago	+	mlflow				0.056466865539550784	0.0013649463653564453	-0.16924861427
		loud-calf-653	6 days ago	+	mlflow				0.44812164306640623	0.0012209415435791016	-0.16747894433
		puzzled-moth...	6 days ago	+	mlflow				0.1862520694732666	0.0011556148529052734	-0.16995296475
		bedecked-jay...	6 days ago	+	mlflow				0.4839490413665771	0.00098419189453125	-0.22100548947
		lyrical-skunk-...	6 days ago	+	mlflow				0.402445650100708	0.0009180068969726563	-0.22100410408
		skillful-moth-...	6 days ago	+	mlflow				0.05154929161071777	0.0007493972778320312	-0.23257341425
		Support_Vect...	6 days ago	+	mlflow						
		bald-wren-557	6 days ago	+	mlflow				0.4478444576263428	0.0008974552154541015	-0.22100675058
		handsome-ca...	6 days ago	+	mlflow				0.03872456550598145	0.0008171558380126953	-0.23257180007
		illustrious-ape...	6 days ago	+	mlflow			-1	0.16047592163085939	0.0012560367584228515	-0.17554238845
		LightGBM_Re...	6 days ago	+	mlflow						
		victorious-sea...	6 days ago	+	mlflow			-1	0.15874385833740234	0.0011797904968261718	-0.17541060092
		gaudy-snail-390	6 days ago	+	mlflow			-1	0.2963273525238037	0.0014684677124023437	-0.17497962675
		popular-mule-...	6 days ago	+	mlflow			-1	0.31059250831604	0.0014919757843017579	-0.17497962675
		nervous-mule...	6 days ago	+	mlflow			-1	0.1471637725830078	0.0011322498321533203	-0.17541060092
		blushing-cow...	6 days ago	+	mlflow			3	0.1067929744720459	0.0006578445434570312	-0.16604421734
		whimsical-har...	6 days ago	+	mlflow			5	0.08242607116699219	0.0006223678588867188	-0.15979987382

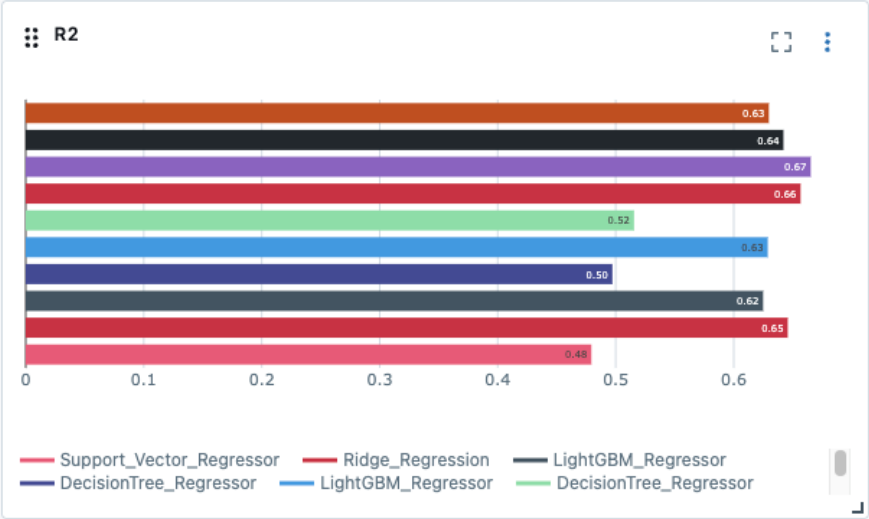
COMPARAÇÃO DE **MODELOS**

Na visão de gráfico podemos comparar as diferentes métricas.









CONFIGURAÇÃO DO RASTREAMENTO **EXPERIMENTO**

Para conduzir os experimentos, vamos utilizar um novo notebook. Antes de iniciar o desenvolvimento e a avaliação dos diferentes modelos, é necessário acessar o dataset já registrado. Definimos o dataset chamado processed como a base principal de features que será utilizada nos treinamentos.

Caminho do repostório no DagsHub. Nome do dataset

```
ds = datasources.get('michelpf/fiap-ds-mlops-laptop-pricing-brl', 'processed')
```

```
ds.all().dataframe
```

```
res = ds.head()
for dp in res:
  dataset_url = dp.download_url
```

Obtendo a URL para carregar no dataframe

```
df = pd.read_csv(dataset_url)
```

Dataset carregado no dataframe, pronto para uso nos modelos

CONFIGURAÇÃO DO RASTREAMENTO **EXPERIMENTO**

O próximo passo é iniciar o DagsHub, informando os dados do repositório e ativando o MLFlow.

```
dagshub.init(repo_owner='michelpf', repo_name='fiap-ds-mlops-laptop-pricing-brl', mlflow=True)
```

Para que o MLflow registre automaticamente todas as métricas durante os experimentos, podemos ativar o recurso de autolog. Essa funcionalidade é especialmente útil em modelos de deep learning, pois permite armazenar informações como o tempo de execução, a evolução da perda ao longo do treinamento e o número de épocas, facilitando a comparação entre diferentes execuções.

```
mlflow.autolog()
```

O recurso de autolog também dispara automaticamente um novo experimento no MLflow sempre que um modelo é treinado. Isso permite acompanhar a evolução em tempo real diretamente pelo portal, facilitando a análise por parte dos usuários que estiverem monitorando os experimentos.

CONFIGURAÇÃO DO RASTREAMENTO **EXPERIMENTO**

Apesar do *autolog* cobrir automaticamente todas as métricas, é boa prática definirmos explicitamente as métricas que precisamos armazenar, pois os modelos podem ter o outras métricas por padrão que nem sempre são as mesmas que são definidas como objetivo.

```
def evaluate_and_log_model(kind, model_name, model, X_test, y_test):
  predictions = model.predict(X_test)
  mse = mean_squared_error(y_test, predictions)
  mae = mean_absolute_error(y_test, predictions)
  r2 = r2_score(y_test, predictions)
  mape = mean_absolute_percentage_error(y_test, predictions)

  mlflow.log_metric("MSE", mse)
  mlflow.log_metric("MAE", mae)
  mlflow.log_metric("R2", r2)
  mlflow.log_metric("MAPE", mape)

  # Inferir a assinatura automaticamente
  signature = infer_signature(X_test, predictions)

  if kind == "catboost":
    mlflow.catboost.log_model(model, "model", signature=signature, input_example=X_test[:5])
  elif kind == "xgboost":
    mlflow.xgboost.log_model(model, "model", signature=signature, input_example=X_test[:5])
  elif kind == "lightgbm":
    mlflow.lightgbm.log_model(model, "model", signature=signature, input_example=X_test[:5])
  else:
    mlflow.sklearn.log_model(model, "model", signature=signature, input_example=X_test[:5])
```

CONFIGURAÇÃO DO RASTREAMENTO **EXPERIMENTO**

Para cada treinamento realizado, seja por meio de GridSearch ou não, podemos registrar métricas específicas no MLflow. O ideal é manter um conjunto de métricas padrão, o que permite comparar de forma consistente o desempenho entre diferentes modelos e abordagens.

Por tal razão que padronizamos a função “evaluate_and_log_model” para sempre rastrear as mesmas métricas.

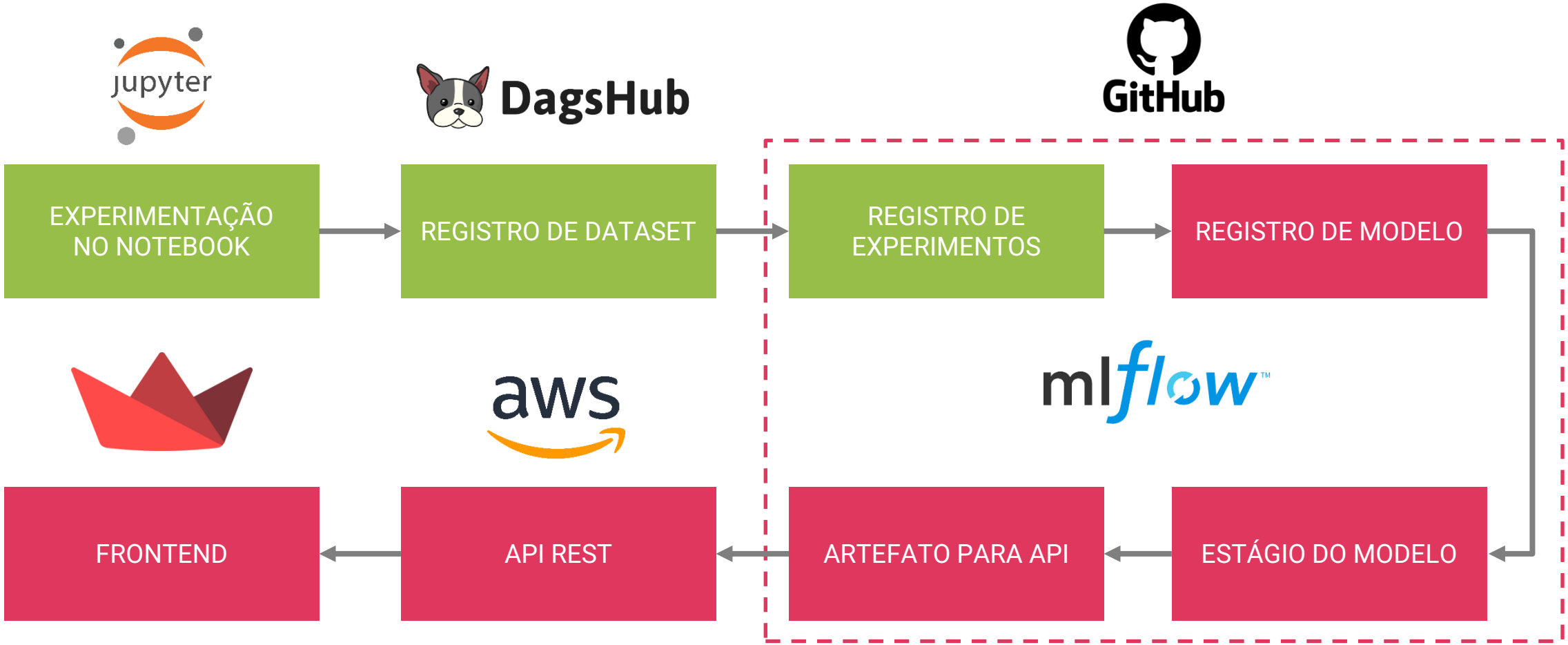
```
with mlflow.start_run(run_name="DecisionTree_Regressor"):
    param_grid = {
        'max_depth': [3, 5, 10, None],
        'min_samples_split': [2, 5, 10]
    }
    tree = DecisionTreeRegressor(random_state=42)
    grid_search = GridSearchCV(tree, param_grid, scoring=make_scorer(mean_absolute_percentage_error, greater_is_better=False), cv=5)
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_

    mlflow.log_param("best_max_depth", best_model.max_depth)
    mlflow.log_param("best_min_samples_split", best_model.min_samples_split)
    evaluate_and_log_model("sklearn", "Decision Tree Regressor", best_model, X_test, y_test)
```

CONFIGURAÇÃO DO RASTREAMENTO **EXPERIMENTO**

Mesmo utilizando GridSearch, o MLFlow é capaz de organizar as execuções mantendo as métricas somente do modelo com melhor desempenho.

Run Name	Created	Dataset	Duration	Source	Version	Models
XGBoost_Regressor	6 days ago	dataset (85a340bf) Train	1.5min	ipykern...	-	laptop-pricing-mo... +2
mysterious-elk-628	6 days ago	-	57.1s	ipykern...	-	-
nosy-bear-911	6 days ago	-	57.1s	ipykern...	-	-
rogue-turtle-779	6 days ago	-	57.1s	ipykern...	-	-
peaceful-shad-626	6 days ago	-	57.1s	ipykern...	-	-
casual-snipe-762	6 days ago	-	57.1s	ipykern...	-	-
Ridge_Regression	6 days ago	dataset (85a340bf) Train	41.2s	ipykern...	-	sklearn +1
DecisionTree_Regressor	6 days ago	dataset (85a340bf) Train	42.3s	ipykern...	-	sklearn +1
MLP_Regressor	6 days ago	dataset (85a340bf) Train	1.5min	ipykern...	-	sklearn +1
LightGBM_Regressor	6 days ago	dataset (85a340bf) Train	52.0s	ipykern...	-	lightgbm +1
Support_Vector_Regressor	6 days ago	dataset (85a340bf) Train	52.1s	ipykern...	-	sklearn +1
CatBoost_Regressor	6 days ago	dataset (85a340bf) Train	52.0s	ipykern...	-	sklearn +1
Ridge_Regression	5 days ago	dataset (9a6c211e) Train	41.4s	ipykern...	-	sklearn +1
DecisionTree_Regressor	5 days ago	dataset (9a6c211e) Train	53.1s	ipykern...	-	sklearn +1
MLP_Regressor	5 days ago	dataset (9a6c211e) Train	1.4min	ipykern...	-	sklearn +1
XGBoost_Regressor	5 days ago	dataset (9a6c211e) Train	1.5min	ipykern...	-	laptop-pricing-mod... +1
LightGBM_Regressor	5 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	lightgbm +1
Support_Vector_Regressor	5 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	sklearn +1
CatBoost_Regressor	5 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	sklearn +1

CICLO PARA CONSTRUIR UM MODELO



REGISTRO DE MODELOS

Os modelos são registrados de acordo com as especificações do arquivo de pipeline. Com os dados do artefato, incluindo métricas e gráficos do último modelo, ele será associado quando for realizado seu registro.

Os modelos podem ser registrados diretamente pela plataforma, em “Models”, como também via linha de comando. Vamos utilizar por linha de comando pois desta forma conseguimos automatizar via esteiras de entrega contínua.

Para registarmos precisamos do “run_id” do experimento que será promovido ao modelo.
A cada novo registro uma nova versão será gerada.

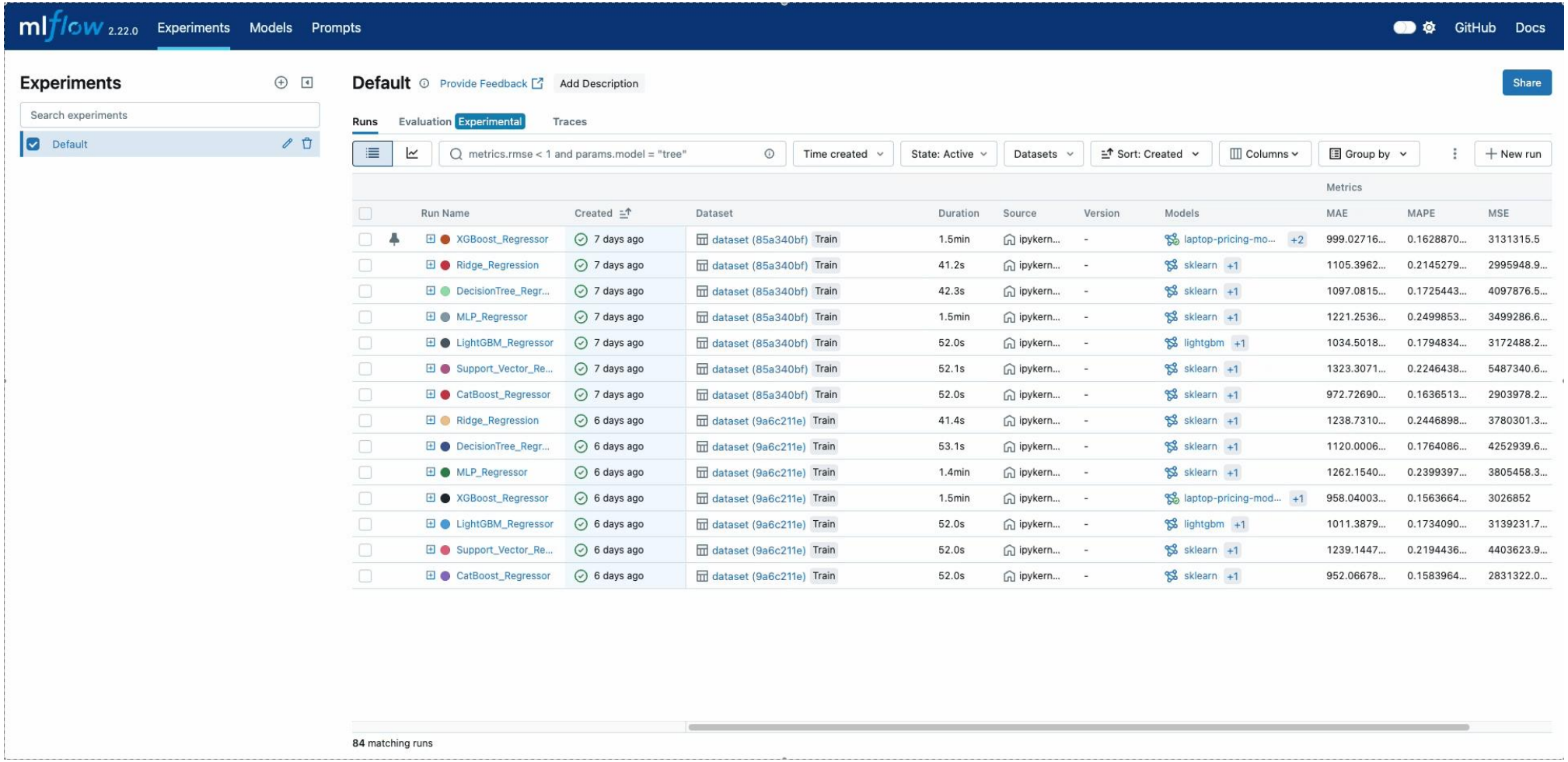
```
run_id = "211650eabbba4ef2a9cf2c7aed6fc36b"
mlflow.register_model(model_uri=f"runs:/{run_id}/model", name="laptop-pricing-model")
```

```
Registered model 'laptop-pricing-model' already exists. Creating a new version of this model... 2025/05/15 23:46:04 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version to finish creation. Model name: laptop-pricing-model, version 3 Created version '3' of model 'laptop-pricing-model'.
```

Cada versão nova é referente a uma nova etapa de treinamento que modificou o modelo. O rastreamento de versão é fundamental para o acompanhamento do modelo

REGISTRO DE MODELOS

A mesma operação também é possível fazer no plataforma do MLFlow. A limitação é que a geração de um modelo novo pelo console não permite iniciar a esteira de CD/CI para lançar uma implantação do modelo em uma API.



Run Name	Created	Dataset	Duration	Source	Version	Models	Metrics MAE	Metrics MAPE	Metrics MSE
XGBoost_Regressor	7 days ago	dataset (85a340bf) Train	1.5min	ipykern...	-	laptop-pricing-mo... +2	999.02716...	0.1628870...	3131315.5
Ridge_Regression	7 days ago	dataset (85a340bf) Train	41.2s	ipykern...	-	sklearn +1	1105.3962...	0.2145279...	2995948.9...
DecisionTree_Regr...	7 days ago	dataset (85a340bf) Train	42.3s	ipykern...	-	sklearn +1	1097.0815...	0.1725443...	4097876.5...
MLP_Regressor	7 days ago	dataset (85a340bf) Train	1.5min	ipykern...	-	sklearn +1	1221.2536...	0.2499853...	3499286.6...
LightGBM_Regressor	7 days ago	dataset (85a340bf) Train	52.0s	ipykern...	-	lightgbm +1	1034.5018...	0.1794834...	3172488.2...
Support_Vector_Re...	7 days ago	dataset (85a340bf) Train	52.1s	ipykern...	-	sklearn +1	1323.3071...	0.2246438...	5487340.6...
CatBoost_Regressor	7 days ago	dataset (85a340bf) Train	52.0s	ipykern...	-	sklearn +1	972.72690...	0.1636513...	2903978.2...
Ridge_Regression	6 days ago	dataset (9a6c211e) Train	41.4s	ipykern...	-	sklearn +1	1238.7310...	0.2446898...	3780301.3...
DecisionTree_Regr...	6 days ago	dataset (9a6c211e) Train	53.1s	ipykern...	-	sklearn +1	1120.0006...	0.1764086...	4252939.6...
MLP_Regressor	6 days ago	dataset (9a6c211e) Train	1.4min	ipykern...	-	sklearn +1	1262.1540...	0.2399397...	3805458.3...
XGBoost_Regressor	6 days ago	dataset (9a6c211e) Train	1.5min	ipykern...	-	laptop-pricing-mod... +1	958.04003...	0.1563664...	3026852
LightGBM_Regressor	6 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	lightgbm +1	1011.3879...	0.1734090...	3139231.7...
Support_Vector_Re...	6 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	sklearn +1	1239.1447...	0.2194436...	4403623.9...
CatBoost_Regressor	6 days ago	dataset (9a6c211e) Train	52.0s	ipykern...	-	sklearn +1	952.06678...	0.1583964...	2831322.0...

REGISTRO DE MODELOS

Histórico de versões para acompanhar as evoluções dos modelos juntamente com os tipos de algoritmos utilizados.

Registered Models ›

laptop-pricing-model

Created Time: 05/14/2025, 10:45:19 PM Last Modified: 05/22/2025, 09:52:51 PM

› Description Edit

› Tags

⌄ Versions Compare

New model registry UI

Version	Registered at	Created by	Tags	Aliases	Description
Version 5	05/22/2025, 09:52:51 PM		Add	Add	
Version 4	05/22/2025, 09:47:53 PM		Add	Add	
Version 3	05/15/2025, 11:46:04 PM		Add	Add	
Version 2	05/15/2025, 10:48:22 PM		Add	Add	
Version 1	05/14/2025, 10:45:20 PM		Add	Add	

1

REGISTRO DE MODELOS

Detalhes sobre descrição, tags e entrada e saída de dados esperados.

Registered Models › laptop-pricing-model ›

Version 5

Registered At: 05/22/2025, 09:52:51 PM Last Modified: 05/22/2025, 09:52:51 PM Source Run: XGBoost_Regressor

Stage (deprecated): None

Description Edit

Tags

Schema

Name	Type
Inputs (1)	
- (required)	Tensor (dtype: int64, shape: [-1,30])
Outputs (1)	
- (required)	Tensor (dtype: float32, shape: [-1])

Os testes de unidade garantem que mesmo que uma implantação seja disparada, o modelo novo não é lançado em produção até que os dados de entrada da API sejam ajustado, no caso de haver alterações.

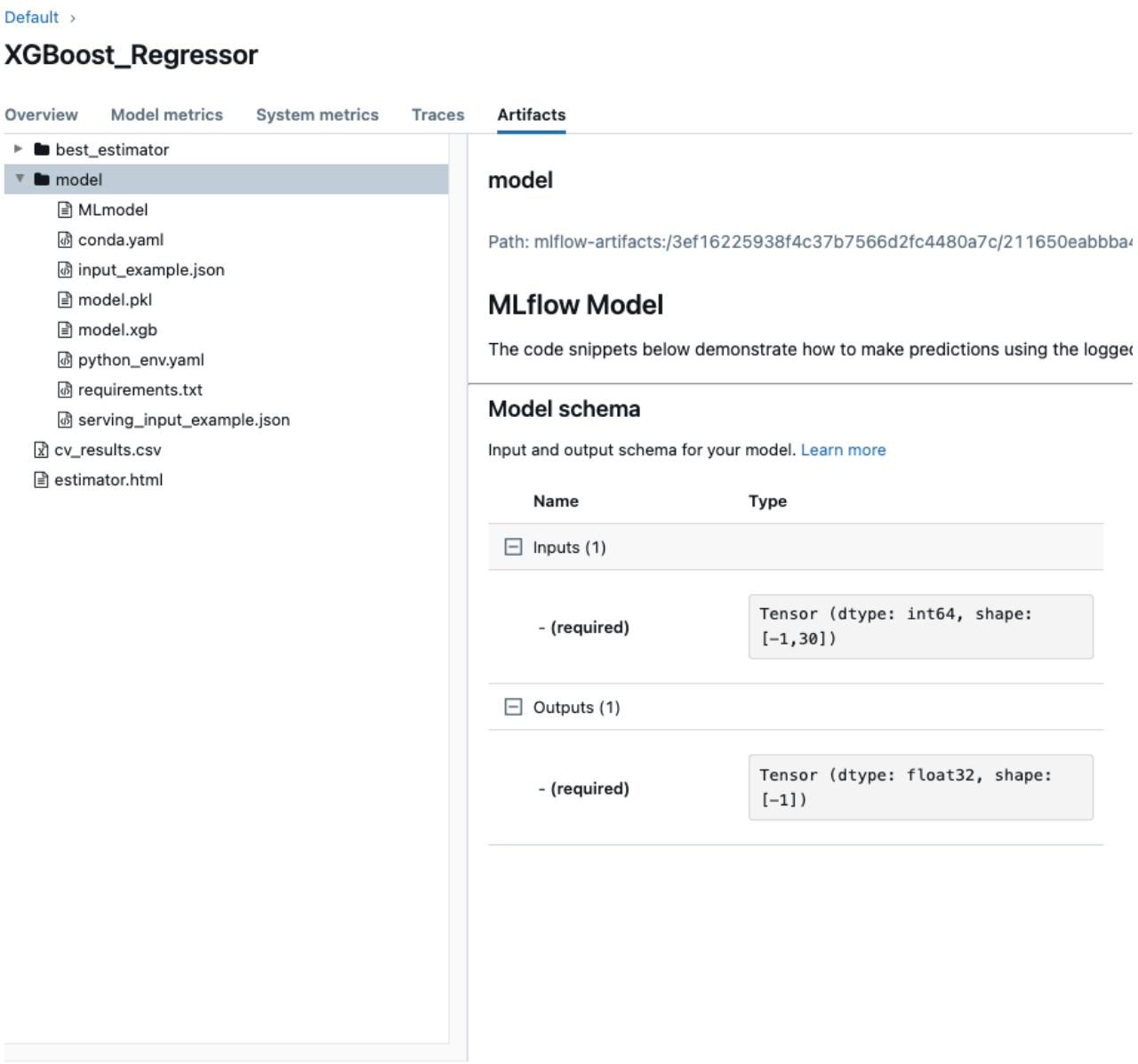
Quando implantamos um modelo novo em uma API existente precisamos ter cuidado com os dados de entrada e saída. Se for incluído ou removido alguma coluna de dados pelo modelo a API precisa estar preparada para enviar os dados no mesmo padrão.

ARTEFATOS DE MODELOS

No registro de modelos temos instruções de como acessá-lo. Podemos baixar o arquivo binário do modelo ou simplesmente carregar diretamente em tempo de execução.

Como nossa API deverá ter uma ajuste nos dados de entrada vamos optar por baixar o arquivo binário (Pickle) e integrar diretamente.

Carregar diretamente em tempo de execução não é recomendável pois a API não pode estar preparada para alterações de entrada. Prefira uma arquitetura imutável para evitar surpresas em produção.



ARTEFATOS DE MODELOS

A forma mais simples de acessar o modelo registrado é via a função “load_model”.

Indicada para uso experimental em Notebooks ou ambientes que não requer integrações com outros sistemas.

Validate the model before deployment

Run the following code to validate model inference works on the example input data and logged model dependencies, prior to deploying it to a serving endpoint

```
import mlflow
from mlflow.models import Model

model_uri = 'runs:/211650eabbba4ef2a9cf2c7aed6fc36b/model'
# The model is logged with an input example
pyfunc_model = mlflow.pyfunc.load_model(model_uri)
input_data = pyfunc_model.input_example

# Verify the model with the provided input data using the logged
dependencies.
# For more details, refer to:
# https://mlflow.org/docs/latest/models.html#validate-models-before-
deployment
mlflow.models.predict(
    model_uri=model_uri,
    input_data=input_data,
    env_manager="uv",
)
```

ARTEFATOS DE MODELOS

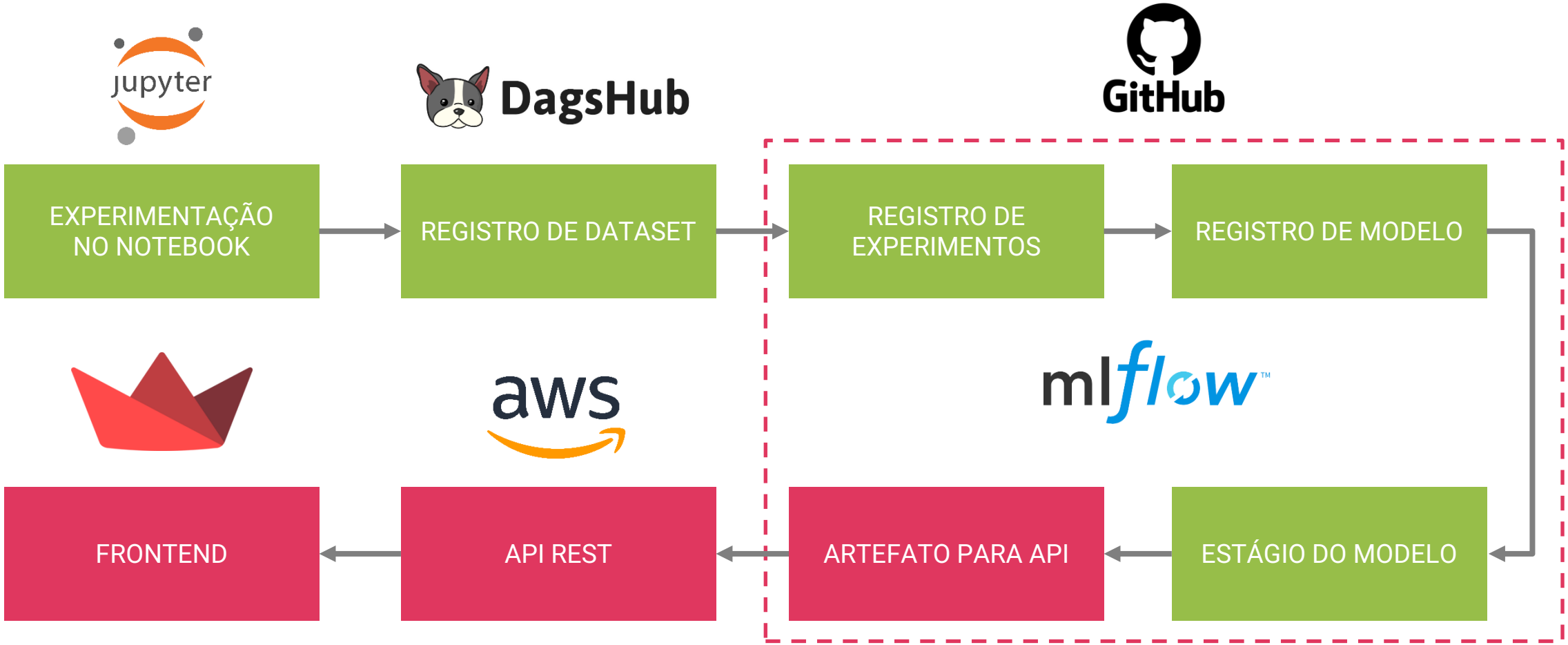
Prefira baixar diretamente o arquivo binário e incorporá-lo na API.

1. Procure pela última versão
2. Baixe o modelo padronizado (model.pkl)
3. Armazene os metadados do modelo para rastreamento de versão
4. Após isso carregue na API em produção quando for gerado uma nova imagem do container.

Este script pode ser colocado nas esteiras de CD/CI durante o registro de um modelo novo ou devido a alguma alteração no código-fonte da API.

```python
mlflow.set_tracking_uri("https://dagshub.com/michelpf/fiap-ds-mlops-laptop-pricing-brl.mlflow")

# Configurações
model_name = "laptop-pricing-model"
artifact_relative_path = "model/model.pkl"

client = MlflowClient()

# 1. Buscar todas as versões do modelo
versions = client.search_model_versions(f"name='{model_name}'")

# 2. Obter a versão mais recente
latest = max(versions, key=lambda v: int(v.version))

# 3. Baixar o artefato
download_path = client.download_artifacts(
  run_id=latest.run_id,
  path=artifact_relative_path,
  dst_path="."
)

model_metadata = {
"model_name": model_name,
"version": latest.version,
"run_id": latest.run_id,
"source": latest.source,
"downloaded_at": datetime.now().isoformat()
}

with open("model/model_metadata.json", "w") as f:
json.dump(model_metadata, f, indent=2)
```

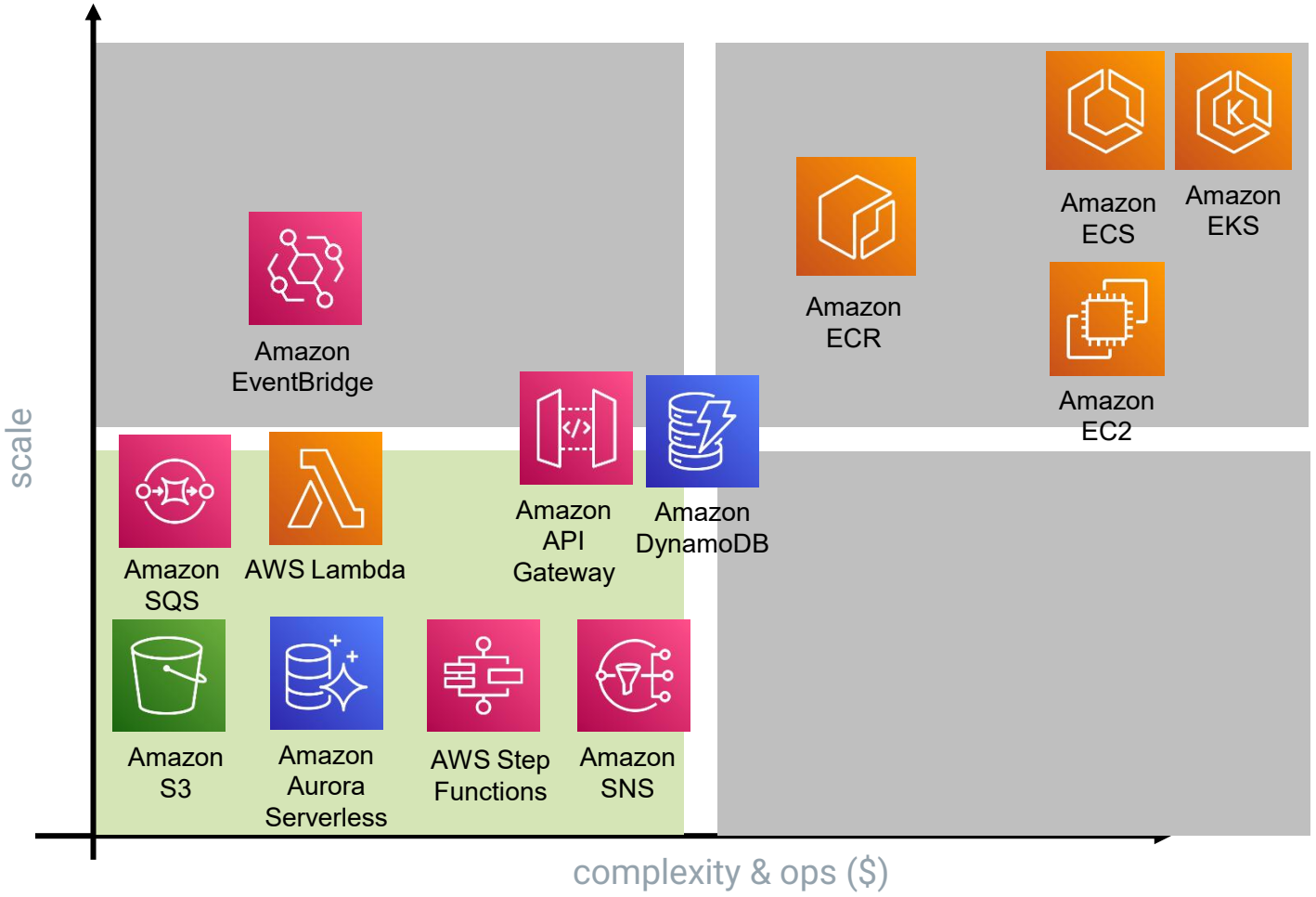
CICLO PARA CONSTRUIR UM MODELO



API REST EM CLOUD

SERVERLESS

Antes de começar qualquer coisa: a arquitetura se adequa às capacidades tecnológicas da equipe?
Nós exploramos a capacidade de Serverless?
Quanto a estimativa do calculadora indica?

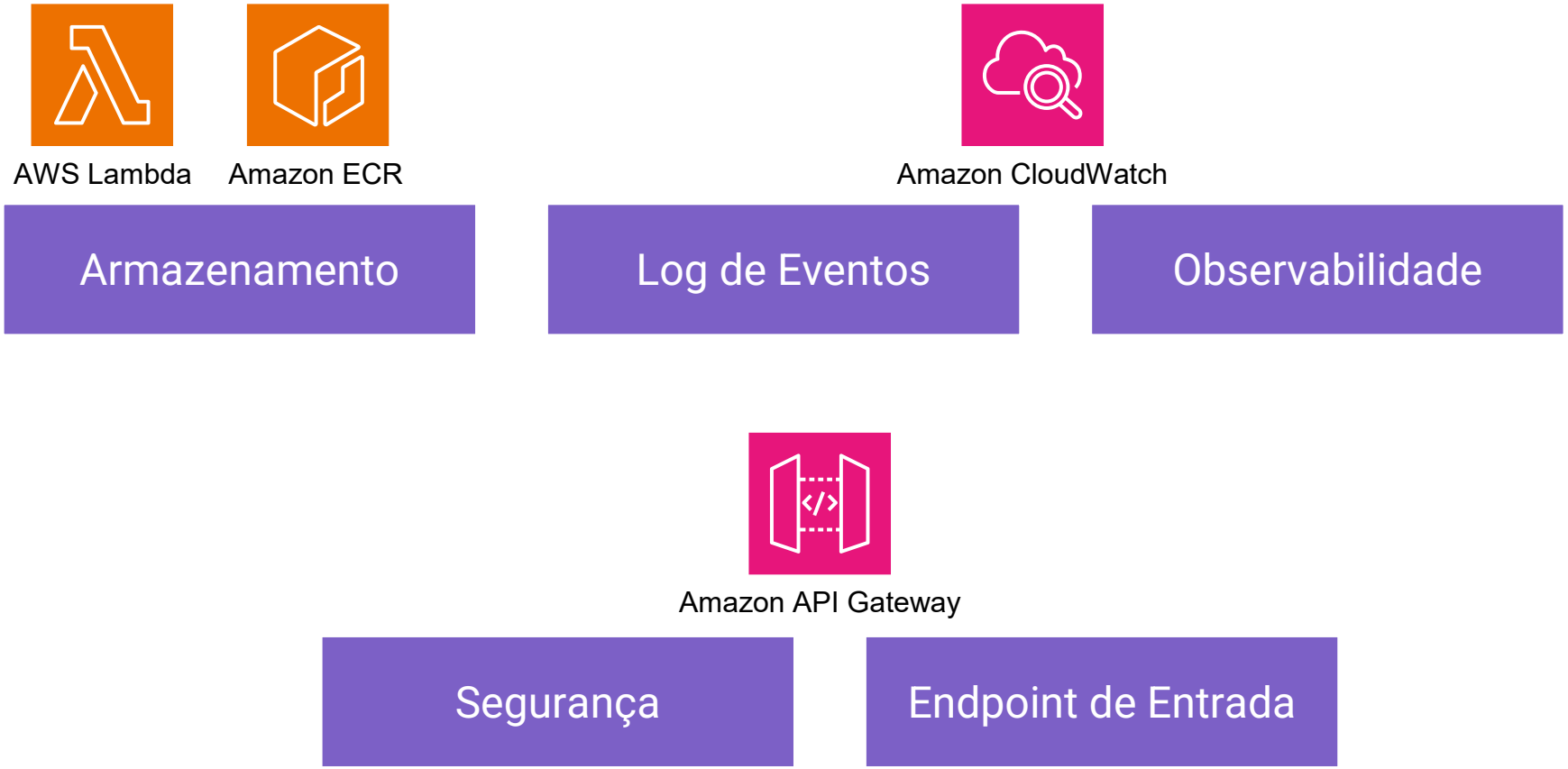


10 MM req/month

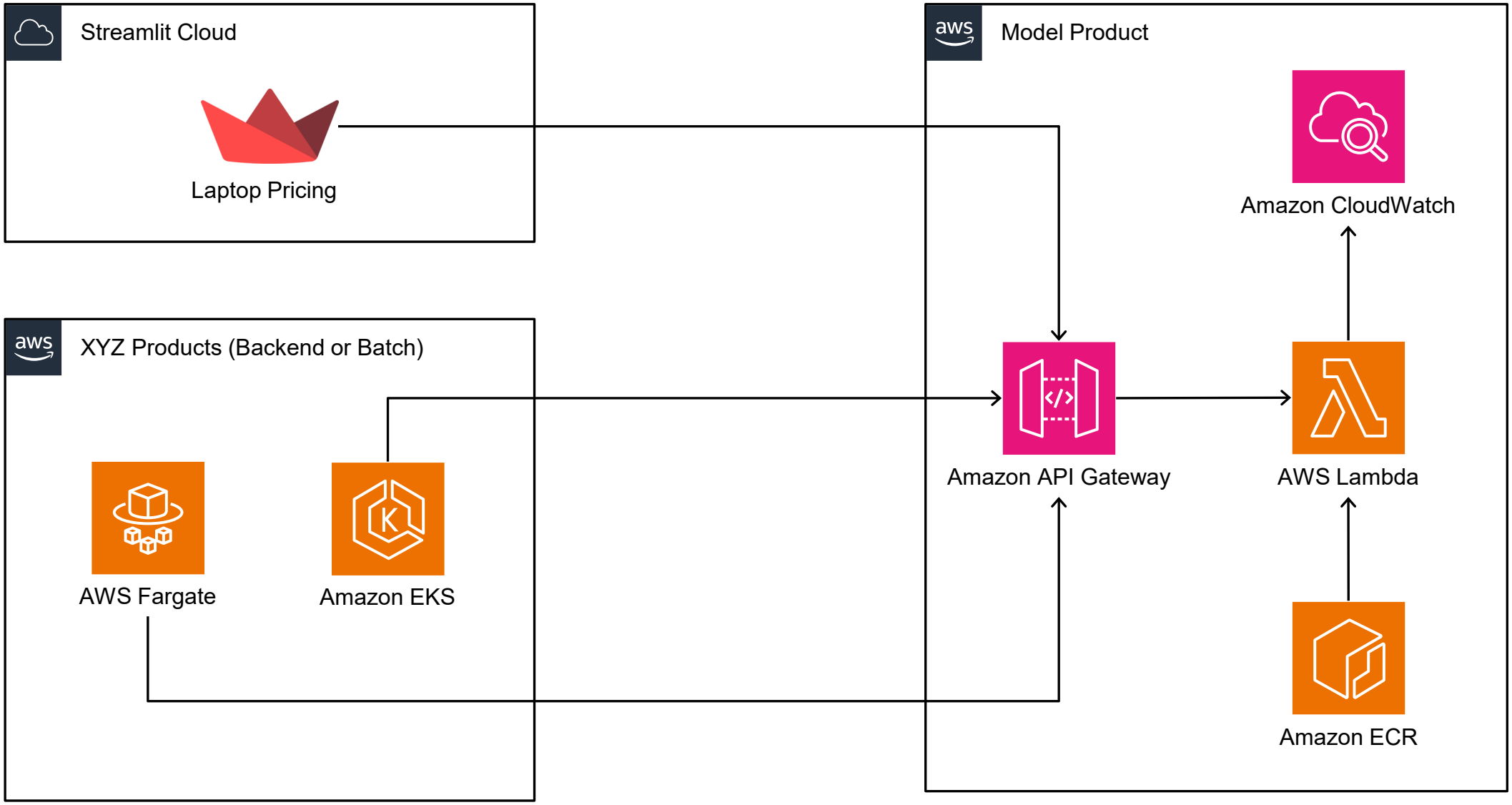
Lambda has 1 MM/month requests always free tier

BLUEPRINT

Iniciando nossa análise para uma API REST precisamos considerar alguns pontos importantes



SYSTEM **DESIGN**



FUNÇÃO **LAMBDA**

Uma função Lambda da AWS é um tipo de serviço de computação serverless. Ela permite que você execute código sem precisar provisionar ou gerenciar servidores.

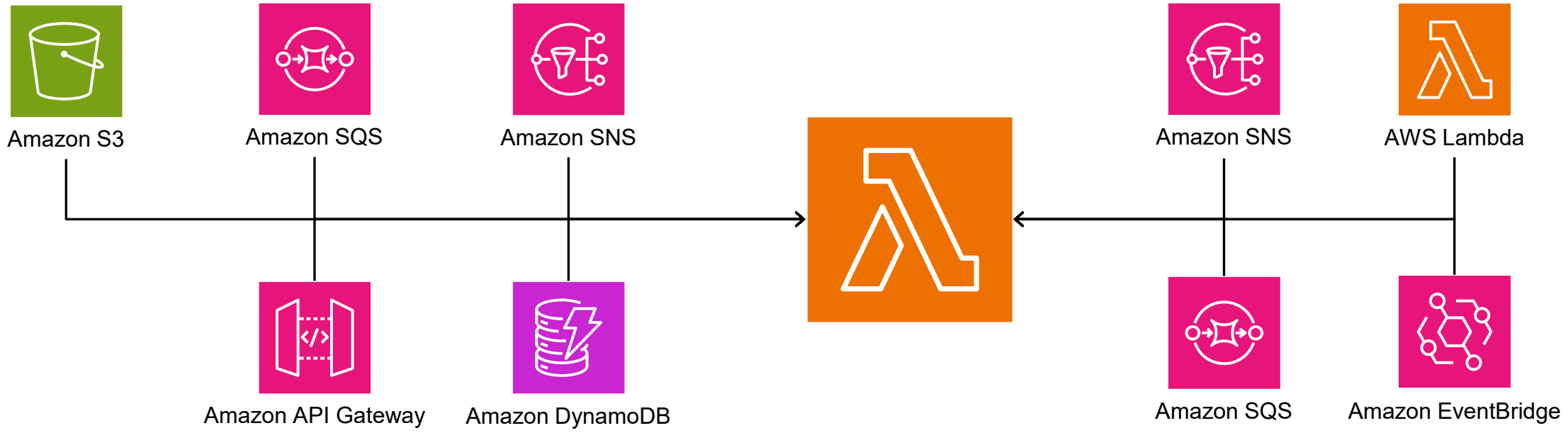
Permite executar código em resposta a eventos, como alterações em dados no Amazon S3, atualizações em tabelas do Amazon DynamoDB, chamadas de API no Amazon API Gateway, entre outros. As funções Lambda podem ser escritas em várias linguagens de programação, incluindo Python, Node.js, Java, C#, e outras.

Principais características:

- **Serverless**: Você não precisa provisionar nem gerenciar servidores. A AWS cuida da infraestrutura, permitindo que o desenvolvedor se concentre apenas no código.
- **Escalabilidade Automática**: As funções Lambda são dimensionadas automaticamente, o que significa que elas podem lidar com qualquer quantidade de tráfego, desde algumas chamadas por mês até milhões de chamadas por segundo.
- **Pagamento por Uso**: Você paga apenas pelo tempo de execução das funções Lambda e pelos recursos computacionais consumidos durante a execução do código. Não há taxas mínimas nem custos fixos.
- **Integração com Serviços AWS**: As funções Lambda podem ser facilmente integradas com outros serviços da AWS, como S3, DynamoDB, API Gateway, SNS, SQS, e muitos outros, o que permite criar aplicações complexas e altamente escaláveis.

FUNÇÃO **LAMBDA**

As funções Lambdas também podem ser iniciadas por outros componentes, como um API Gateway, para processamento síncrono ou outras formas assíncronas, como eventos, filas ou ações baseadas em eventos específicos do S3, como criação ou modificação de objetos, por exemplo.
Os eventos são disparados para iniciar e também há eventos de pós-processamento, seja num evento de erro ou sucesso para notificações ou ações encadeadas.



MODOS DE **DEPLOY: CONTAINER**

Uma função Lambda pode ser implantada das seguintes maneiras:

1. Programar diretamente no Console. Opção limita a instalação de pacotes adicionais além do Boto3.
2. Programar localmente (VSCode, por exemplo), depois enviar o arquivo compactado com as dependências.
3. Criar uma imagem de contêiner num registro e associar a imagem a uma função.

Implantações padrão (não por contêiner) tem a limitação de tamanho máximo de 250MB (descompactado, com 50MB compactado).

Por contêiner possui limite de 10GB de armazenamento.

MODOS DE **DEPLOY: PADRÃO**

No deploy local precisamos separar numa pasta temporária os arquivos para compactar. Note que as bibliotecas utilizadas precisam estar na raiz e não na pasta do ambiente virtualizado, por isso instalamos as dependências com o parâmetro “-t” de target.

```
mkdir deploy
pip install -r .\requirements.txt -t deploy
copy app.py deploy
cd deploy
zip -r function.zip .
```

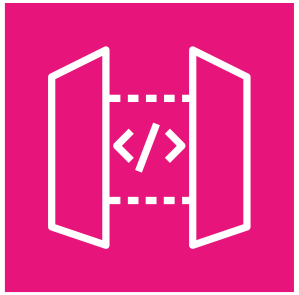
Após compactarmos, enviamos o arquivo pela Console da AWS, em “Código”, depois “Fazer Upload” e então de arquivo local. Com a implantação terminada, a pasta deploy pode ser removida.

Ou utilizar o comando abaixo para carregar via AWS CLI.

```
aws lambda update-function-code --function-name detectar-face --zip-file fileb://function.zip
```

API **GATEWAY**

O Amazon API Gateway é um serviço da AWS que facilita a criação, publicação, manutenção, monitoramento e proteção de APIs (Application Programming Interfaces) RESTful e WebSocket para aplicativos na nuvem.



Ele atua como uma porta de entrada para os aplicativos acessarem os dados, a lógica de negócios ou os serviços de back-end de forma segura e escalável.

Principais características:

- **Criação de APIs RESTful e WebSocket:** suporta a criação de APIs RESTful tradicionais, bem como APIs WebSocket para aplicativos que necessitam de comunicação bidirecional em tempo real.
- **Integração com Serviços AWS e HTTP/S:** integração de APIs com serviços da AWS, como Lambda, DynamoDB, S3, entre outros, além de serviços HTTP/S externos.
- **Gerenciamento de Tráfego e Versões:** permite controlar o tráfego para diferentes versões de suas APIs, além de oferecer suporte para testes e implantações canário.
- **Autenticação e Autorização:** oferece recursos avançados de autenticação e autorização, incluindo integração com o AWS Identity and Access Management (IAM), tokens JWT (JSON Web Tokens), Cognito, entre outros.
- **Monitoramento e Logging:** fornece métricas detalhadas, registros de acesso e monitoramento em tempo real, permitindo que você monitore o desempenho e a integridade de suas APIs.
- **Segurança:** protege as APIs contra ameaças comuns, como ataques de DDoS (Distributed Denial of Service) e injeção de SQL, além de oferecer suporte para criptografia SSL/TLS.

ELASTIC CONTAINER REGISTRY (**ECR**)

O Amazon Elastic Container Registry (ECR) é um serviço gerenciado da AWS que permite armazenar, gerenciar e implantar imagens de contêineres Docker.

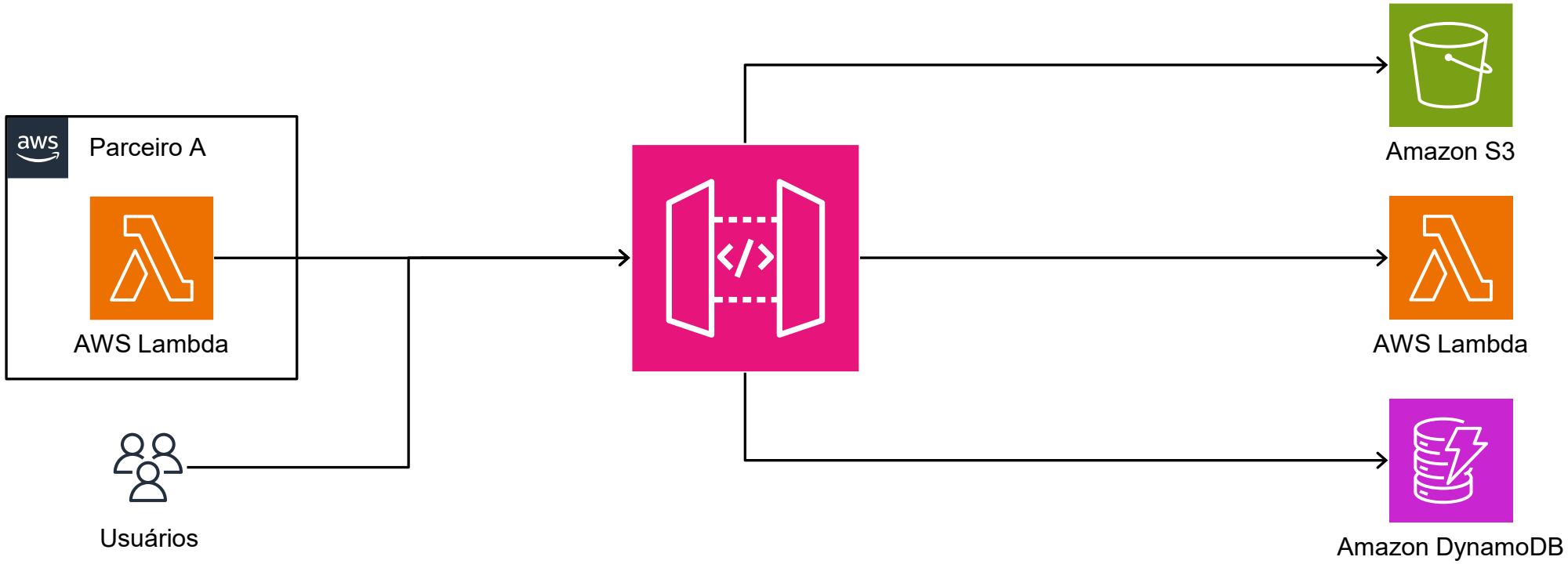
Fornece um repositório seguro e privado para armazenar imagens de contêineres Docker. Ele é integrado com outros serviços da AWS, como o Amazon ECS (Elastic Container Service) e o Amazon EKS (Elastic Kubernetes Service), facilitando a implantação de contêineres em escala na nuvem da AWS.

Principais características:

- **Repositório de Imagens Docker:** fornece repositórios privados para armazenar e gerenciar imagens de contêineres Docker.
- **Segurança e Controle de Acesso:** as imagens do ECR são armazenadas de forma segura e privada. É possível controlar o acesso aos repositórios usando políticas de acesso baseadas em IAM (Identity and Access Management) da AWS, garantindo que apenas usuários autorizados possam acessar e implantar imagens de contêineres.
- **Integração com Serviços AWS:** é totalmente integrado com outros serviços da AWS, como o Amazon ECS, o Amazon EKS e o AWS Lambda, permitindo que você implante facilmente contêineres em escala na nuvem da AWS.
- **Escalabilidade e Disponibilidade:** é altamente escalável e oferece alta disponibilidade, permitindo armazenar e implantar imagens de contêineres em escala, mesmo para cargas de trabalho de missão crítica.
- **Integração com Ferramentas de Desenvolvimento:** O ECR é compatível com várias ferramentas de desenvolvimento e CI/CD (Continuous Integration/Continuous Deployment), como Docker CLI, AWS CLI, Jenkins, GitLab CI, e muitas outras, facilitando a integração com seus fluxos de trabalho de desenvolvimento existentes.

API **GATEWAY**

Como o próprio nome diz, o termo “Gateway” é uma porta de entrada para os serviços da AWS, garantindo segurança e flexibilidade para integrações com outros componentes.
Apesar do Lambda ser a mais comum, é possível também integrá-lo com o DynamoDB e o S3, diretamente.
Por ser um serviço público, ele pode ser acessado tanto internamente quanto externamente.



S3: SIMPLE STORAGE SERVICE

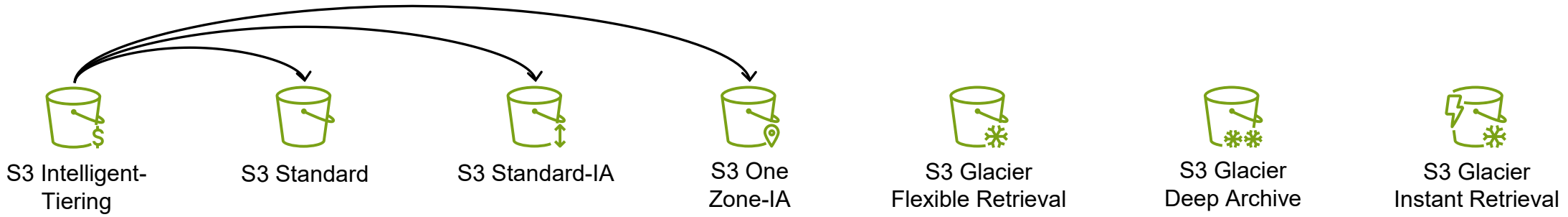
O Amazon S3 é um serviço de armazenamento de objetos da AWS que oferece escalabilidade, disponibilidade, segurança e desempenho para armazenar e recuperar dados na nuvem.
É um serviço de armazenamento de objetos altamente escalável e durável, projetado para armazenar e recuperar grandes volumes de dados de forma segura e eficiente. Ele é amplamente utilizado para uma variedade de casos de uso, como armazenamento de backups, hospedagem de arquivos estáticos para websites, distribuição de conteúdo, armazenamento de dados para aplicativos móveis e análises de big data, entre outros.

Principais características:

- **Armazenamento de Objetos:** armazena dados na forma de objetos, que podem ser arquivos de qualquer tipo, desde documentos e imagens até vídeos e backups de banco de dados.
- **Escalabilidade e Disponibilidade:** Ele é altamente escalável e pode acomodar desde alguns gigabytes até vários petabytes de dados. Além disso, o S3 é projetado para oferecer alta disponibilidade, com durabilidade de 99,999999999% (onze noves) dos objetos armazenados.
- **Modelo de Consistência**: O S3 oferece um modelo de consistência forte para todas as operações de leitura, garantindo que os objetos recuperados sejam os mais recentes e consistentes.
- **Controle de Acesso:** Ele fornece recursos avançados de controle de acesso, permitindo que você defina políticas granulares de permissões de acesso aos seus dados usando o IAM (Identity and Access Management) da AWS.
- **Segurança:** O S3 oferece várias camadas de segurança para proteger seus dados, incluindo criptografia em repouso e em trânsito, controle de acesso baseado em políticas, e integração com outros serviços de segurança da AWS.
- **Integração com Outros Serviços:** O S3 pode ser facilmente integrado com outros serviços da AWS, como Lambda, API Gateway, CloudFront, e serviços de processamento de big data, como o Amazon EMR e o Amazon Athena.

S3: SIMPLE STORAGE SERVICE

Por ser um serviço de armazenamento de objetos, ele possui classes de armazenamento associado a cada caso de uso, combinando performance e custo. A ordem das classes é para o dado armazenado mais quente (mais utilizado) para o mais frio (menos utilizado)



Intelligent-Tiering: é projetada para otimizar automaticamente os custos e o desempenho, movendo os objetos entre as camadas de armazenamento frequentemente acessadas e as camadas de armazenamento infrequentemente acessadas, com base nos padrões de acesso aos dados ao longo do tempo.
S3 Standard: classe padrão do S3, oferecendo alta disponibilidade, durabilidade e desempenho para dados frequentemente acessados.
Standard-IA: oferece os mesmos benefícios do Standard, mas com custos mais baixos para dados que são acessados com menos frequência.
One Zone-IA: oferece os mesmos benefícios do S3 Standard-IA, mas armazena os dados em apenas uma zona de disponibilidade, tornando-a mais econômica.
Glacier Flexible Retrieval: classe de armazenamento de "glacier" que oferece uma recuperação flexível dos dados, permitindo que você recupere os dados em minutos a horas.
Glacier Deep Archive: classe de armazenamento de "glacier" é projetada para dados que são acessados raramente e exigem uma recuperação ainda mais econômica do que o S3 Glacier Flexible Retrieval.
Glacier Instant Retrieval: esta é uma opção de recuperação instantânea para dados armazenados no S3 Glacier, permitindo que recuperar os dados em segundos.

S3: PRINCIPAIS COMANDOS DA API

Para realizar os comandos abaixo você precisa configurar as chaves de acesso via AWS CLI ou especificá-las no cliente. Esta última opção não é recomendada por expor as chaves em base de código.

```python
import boto3

# Crie um cliente S3
s3_client = boto3.client('s3')


# Listar todos os buckets
response = s3_client.list_buckets()
for bucket in response['Buckets']:
    print(bucket['Name'])


# Criar um novo bucket
bucket_name = 'nome-do-seu-bucket'
s3_client.create_bucket(Bucket=bucket_name)


# Enviar um arquivo para um bucket
file_path = 'caminho/para/seu/arquivo.txt'
object_key = 'nome-do-arquivo-no-s3.txt'
bucket_name = 'nome-do-seu-bucket'
s3_client.upload_file(file_path, bucket_name, object_key)
```

```python
import boto3

# Defina suas chaves de acesso
ACCESS_KEY_ID = 'sua_access_key_id'
SECRET_ACCESS_KEY = 'sua_secret_access_key'

# Crie um cliente S3 com as chaves de acesso
s3_client = boto3.client(
    's3',
    aws_access_key_id=ACCESS_KEY_ID,
    aws_secret_access_key=SECRET_ACCESS_KEY
)
```

Utilize com cautela esta opção de cliente.

S3: PRINCIPAIS COMANDOS DA API

Bucket é o nome reservatório dos arquivos. Object é o nome dos arquivos que podem incluir diretórios, por exemplo: “arquivo.txt” ou “arquivos/arquivo.txt” é um objeto da mesma forma.
É possível criar pastas sem objetos diretamente no console.

```
# Listar todos os objetos em um bucket
bucket_name = 'nome-do-seu-bucket'
response = s3_client.list_objects_v2(Bucket=bucket_name)
for obj in response['Contents']:
    print(obj['Key'])
```

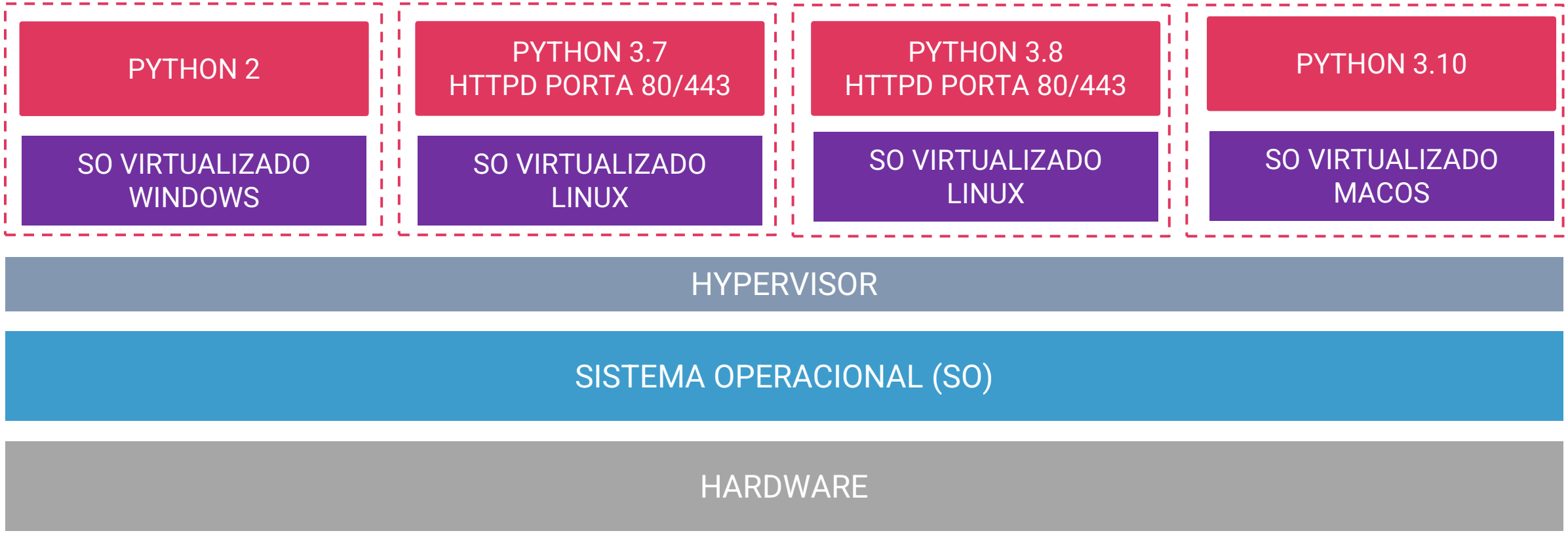
```
# Baixar um arquivo do S3
object_key = 'nome-do-arquivo-no-s3.txt'
bucket_name = 'nome-do-seu-bucket'
local_file_path = 'caminho/para/salvar/o/arquivo.txt'
s3_client.download_file(bucket_name, object_key, local_file_path)
```

```
# Excluir um objeto do S3
object_key = 'nome-do-arquivo-no-s3.txt'
bucket_name = 'nome-do-seu-bucket'
s3_client.delete_object(Bucket=bucket_name, Key=object_key)
```

CONTEINERIZAÇÃO

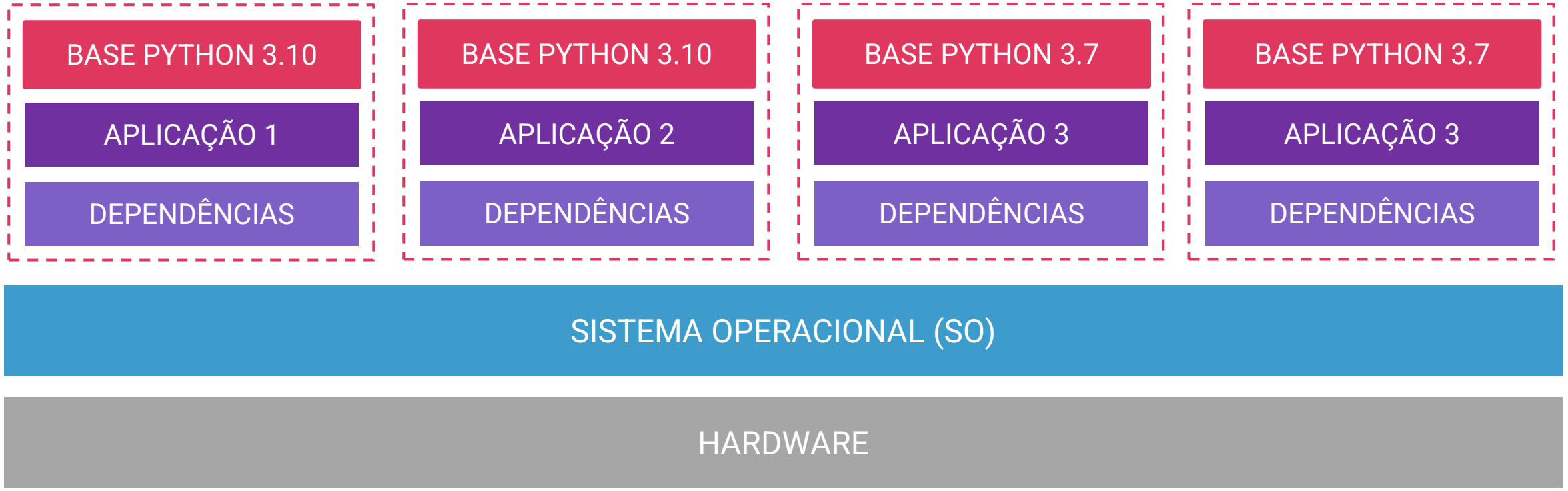
VIRTUALIZAÇÃO DE INFRAESTRUTURA

Conforme as aplicações vão crescendo e evoluindo, precisam de recursos e isolamento para trabalharem para extrair o máximo de seus recursos. Utilizando um servidor físico torna limitado essa flexibilidade pois teríamos que utilizar apenas um único sistema operacional, não podemos utilizar a mesma porta e também as versões e runtimes de interpretadores ficariam fixados em uma única versão.
Para resolver este problema as máquinas virtuais, como o EC2 da AWS foram criadas.



CONTEINEIRIZAÇÃO DE APLICAÇÕES

Com a conteinerização de aplicações alcançamos os mesmos objetivos principais da virtualização, com um custo menor pois não precisamos instalar novos sistemas operacionais.
Imagens formados por camadas ainda ajudam a ser mais eficiente no espaço físico e memória ao promover maior reutilização.
Da mesma forma como nas máquinas virtuais, os contêineres alcançam os mesmos níveis de isolamento, como processos, rede, memória, rede, armazenamento, e inclusive de kernel (como se fosse outro host).



EXECUTANDO CONTAINERS

Uma vez instalado o Docker podemos executar containers diretamente na máquina de desenvolvimento. Para verificação se deu tudo certo, vamos iniciar o “hello world” do Docker.

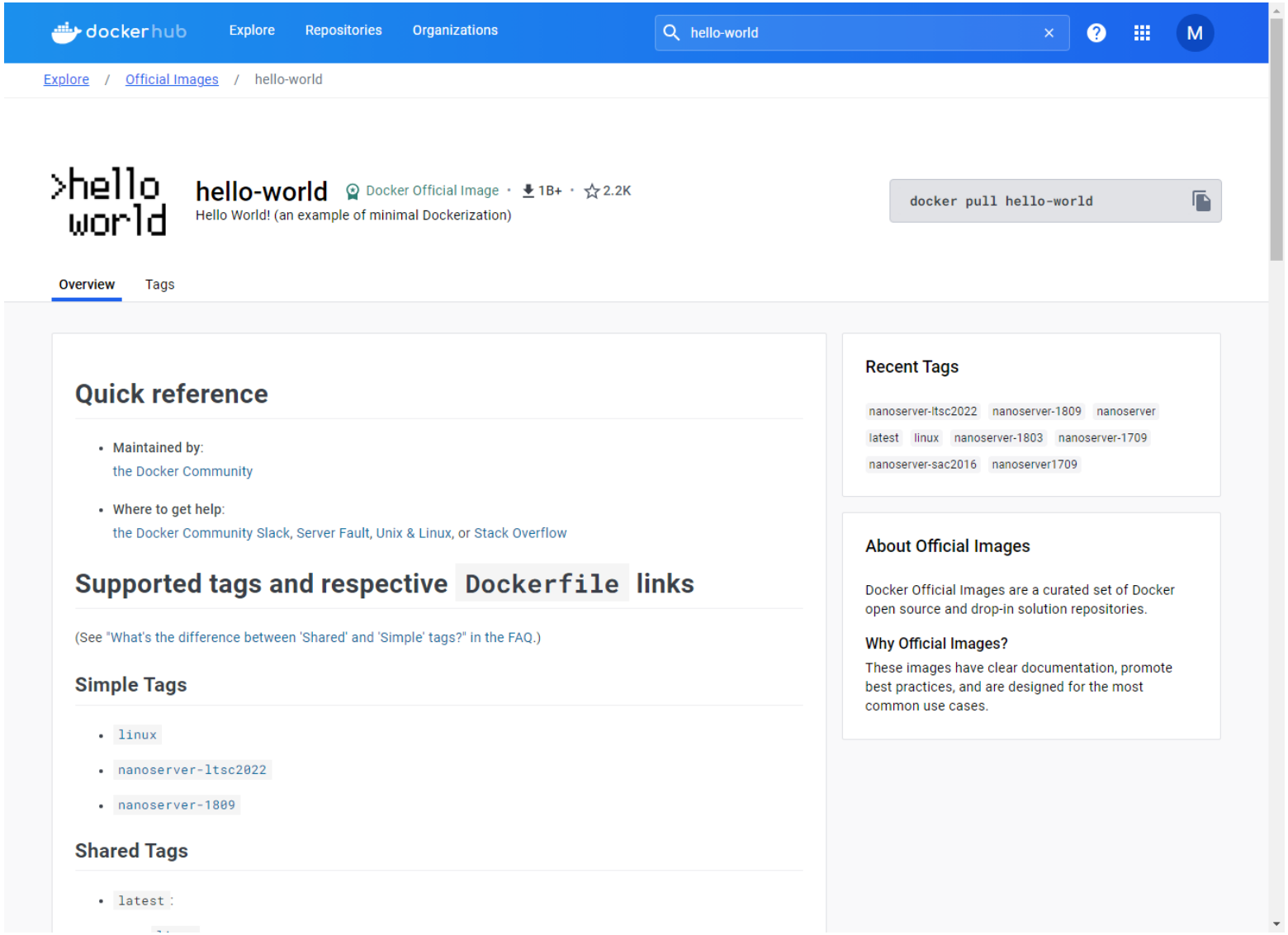
```
docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Obtendo imagem de um registry (DockerHub)

O comando docker run automaticamente faz um pull da imagem quando ela não é encontrada no DockerHub.

CONTAINER **REGISTRY**: **DOCKERHUB**



Assim como códigos são versionados no GitHub, containers são versionados no DockerHub, um registro de containers.

Neste registro público podemos encontrar diversas imagens de containers para uso, baseado nas mais diversas necessidades, como versões de Python específicas, sistema operacional, e até mesmo imagens reduzidas para diminuir o tamanho para implantações que utilize o mínimo necessário.

https://hub.docker.com/_/hello-world

COMANDOS PRINCIPAIS

Listar todos os containers em execução ou não.

```
docker container ls –a

CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
9ab4e976549c   hello-world   "/hello"   10 seconds ago   Exited (0) 6 seconds ago              musing_gagarin
1b97f144ea22   hello-world   "/hello"   7 hours ago      Exited (0) 7 hours ago                fervent_pare
```

O container sempre vai executar um comando especificado dentro do “Dockerfile”. Após ele executar o comando ele termina a execução.
Servidores web contém comando que inicia um processo aguardando conexões, neste caso o container fica executando indefinidamente.

Executar comandos dentro da imagem do docker. Exemplo: exibir o conteúdo da pasta var dentro do container ubuntu.

```
docker run ubuntu ls var

backups
cache
lib
local
lock
log
mail
opt
run
spool
tmp
```

COMANDOS PRINCIPAIS

Mapeamento de portas. O “-d” é de detached (para não travar o console), 8080 é a porta local mapeada para a porta 80 do container.

```
docker run –d –p 8080:80 dockersamples/static-site
```

```
CONTAINER ID   IMAGE     COMMAND      CREATED              STATUS              PORTS     NAMES
331325df8538   ubuntu    "sleep 1d"   About a minute ago   Up About a minute             cool_liskov
```

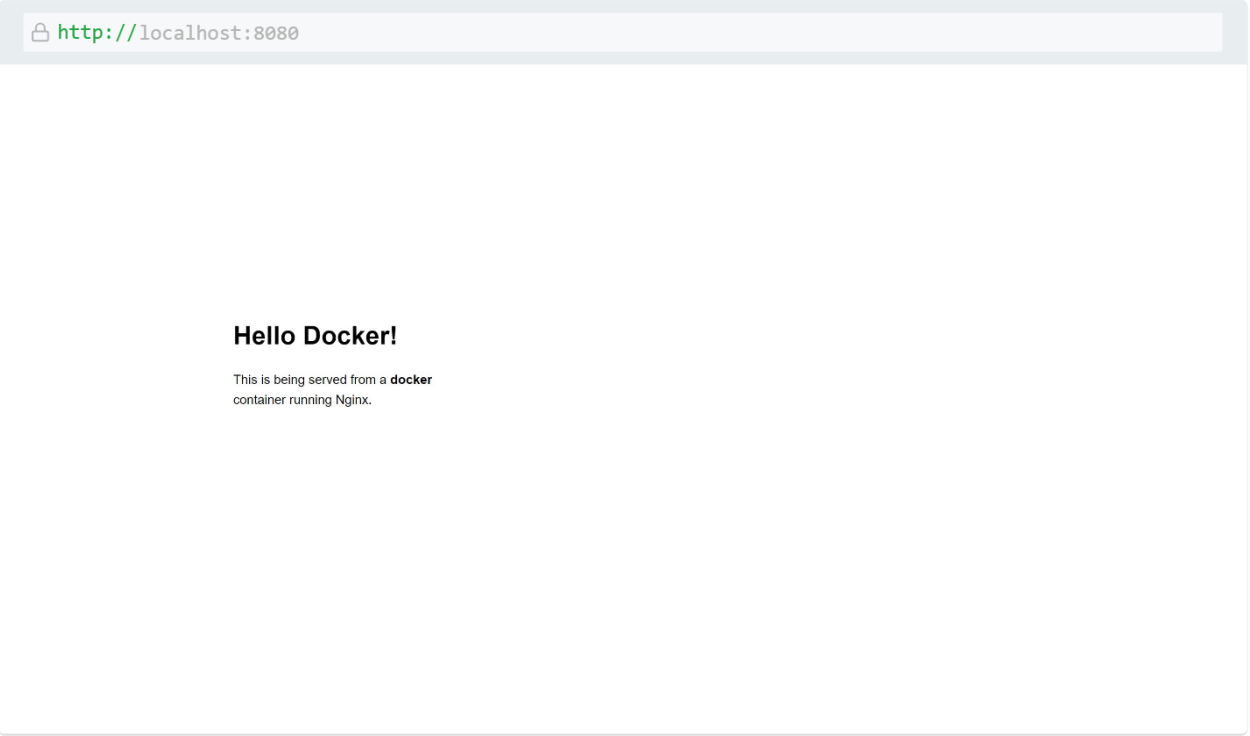
Exibir o mapeamento de porta

```
docker port 6b35c0cc8acb
```

```
80/tcp -> 0.0.0.0:8080
```

Com este mapeamento conseguimos acessar o conteúdo do container da máquina host.

Com todos os isolamentos existentes, o mapeamento de porta torna possível que a porta 8080 seja redirecionada para dentro do container para a porta 80.



COMANDOS PRINCIPAIS

FIAP MBA+

Containers em execução.

```
docker ps

CONTAINER ID   IMAGE     COMMAND      CREATED              STATUS              PORTS     NAMES
331325df8538   ubuntu    "sleep 1d"   About a minute ago   Up About a minute             cool_liskov
```

Interromper a execução.

```
docker stop 331325df8538

331325df8538
```

Reiniciar a execução.

```
docker start 331325df8538

331325df8538
```

Remover container (após interrompê-lo). Perderá todos os dados armazenados.

```
docker rm 331325df8538

331325df8538
```

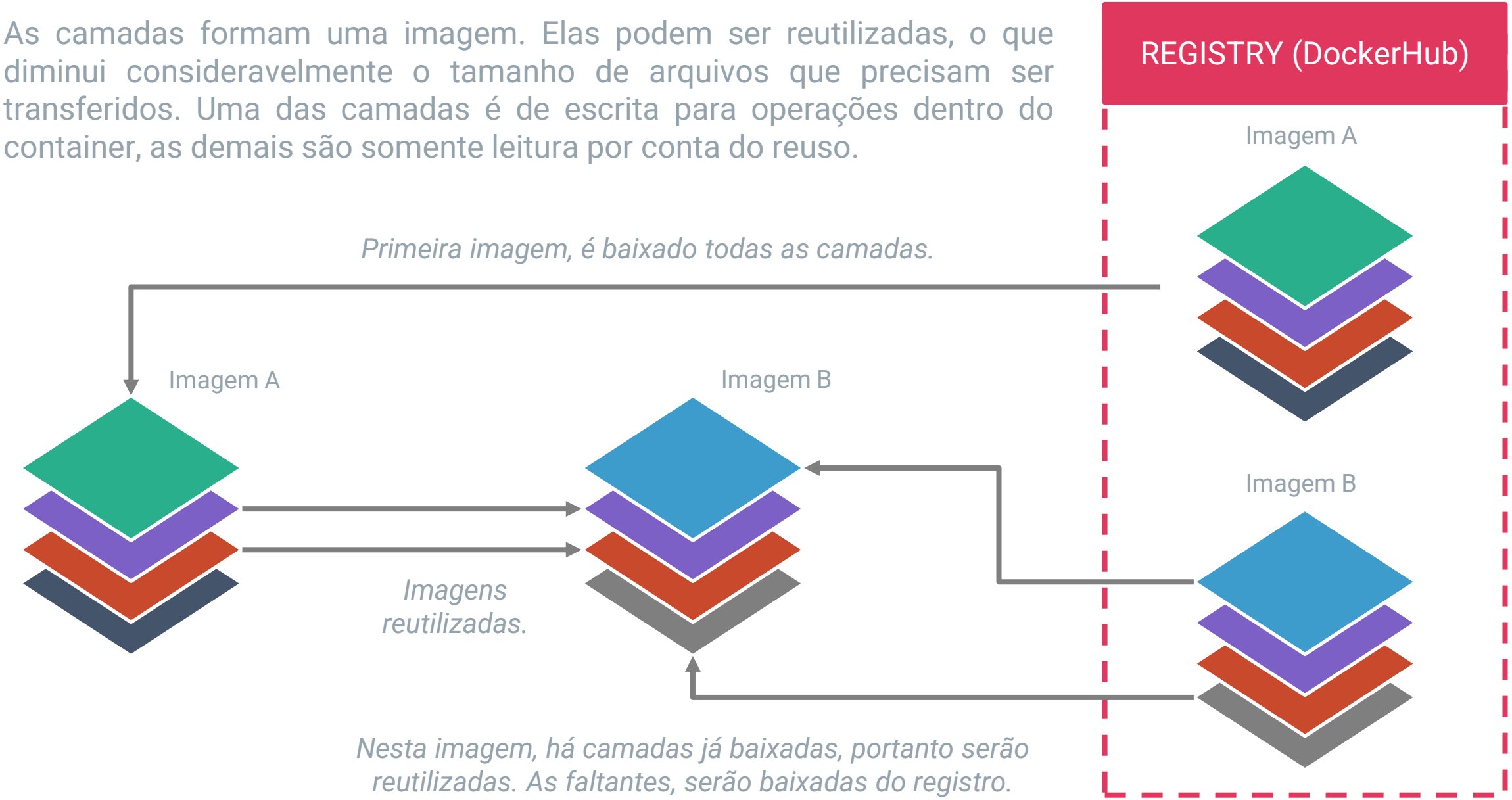
Modo iterativo. Executar comandos dentro do container.

```
docker exec -it 331325df8538 bash

root@331325df8538:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@331325df8538:/#
```

Experimente instalar o Python (apt update | apt install python3).

CAMADAS DE IMAGENS

As camadas formam uma imagem. Elas podem ser reutilizadas, o que diminui consideravelmente o tamanho de arquivos que precisam ser transferidos. Uma das camadas é de escrita para operações dentro do container, as demais são somente leitura por conta do reuso.



DOCKERFILE

O Dockerfile é como se fosse a “receita” do container. Neste arquivo é especificado a imagem, mapeamento de portas, comandos de cópia de arquivos além de processos que precisam ser executados.

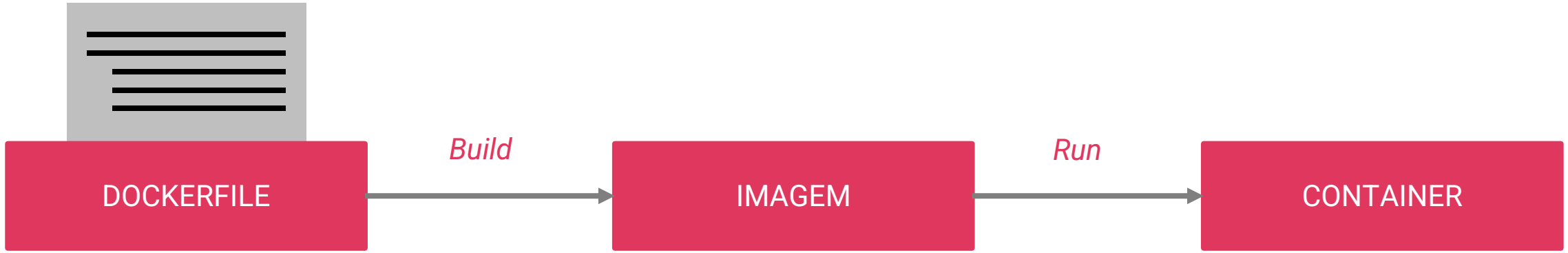


Imagem base
Cópia de arquivos (código fonte, modelo)
Instalação de dependências
Inicialização

DOCKERFILE PARA IMAGEM DE CONTAINER

O arquivo Docker file é como se fosse uma receita para montarmos uma imagem customizada, copiando arquivos necessário, baixando uma imagem de base e configurando aplicações para serem executadas ao iniciar sua execução.

```
# Use a imagem oficial Python como base
FROM python:3

# Copy requirements.txt
WORKDIR /app

# Install the specified packages
COPY ..

# Invoca o compando python app/app.py
CMD [ "python", "app/app.py" ]
```

Obtendo imagem de um registry (DockerHub)

Pasta de trabalho, onde os arquivos serão copiados

Instala as dependências

Executa o comando ao iniciar

https://docs.docker.com/engine/reference/builder/

ANATOMIA DA **API**

A função básica da API é retornar via interface RESTFUL a previsão de preço de um determinado modelo de laptop, utilizando a última versão do modelo em produção.

CARREGAR MODELO	A imagem do container deve conter a última versão do modelo. Toda a vez que o modelo for atualizado, uma nova versão da imagem precisa ser gerada.
CONVERTER PAYLOAD PARA FEATURES	O uso do one-hot encoding não é trivial. Afim de manter uma boa usabilidade, utilizaremos um payload no formato JSON e faremos a conversão na própria API.
REALIZAR INFERÊNCIA	Executar o modelo, associando sua versão utilizada.
ARMAZENAR REQUISIÇÕES	Salvar em um arquivo CSV incremental para análises de desvio de dados e modelo para acompanhamento.
LOGAR EVENTOS E MÉTRICAS	Acompanhamento de métricas de infraestrutura e distribuição dos dados do modelo. Log granular e dashboards consolidados para observabilidade.

CARREGAR O MODELO

No início do código principal (app.py) incluiremos as importações necessárias para a carga do modelo serializado, acesso a serviços da AWS como o log de eventos Cloudwatch e o componente datetime para armazenamento de timestamp dos eventos da API.

Note que como controlamos a versão do modelo por tags, não faria sentido embutir este dado como metadado do próprio modelo. Por tal razão utilizaremos um arquivo texto que informa a versão utilizada do modelo.

```
import boto3
import joblib
import json

model = joblib.load("model/model.pkl")
with open("model/model_metadata.json", "r") as f:
model_info = json.load(f)

cloudwatch = boto3.client("cloudwatch")
```

model_metadata.json


```
{
"model_name": "laptop-pricing-model",
"version": "5",
"run_id": "211650eabbba4ef2a9cf2c7aed6fc36b",
"source": "mlflow-
artifacts:/3ef16225938f4c37b7566d2fc4480a7c/211650eabbba4ef2a9cf2c7
aed6fc36b/artifacts/model",
"downloaded_at": "2025-06-01T14:09:58.259050"
}
```

CONVERTER PAYLOAD EM **FEATURES**

```
def prepare_payload(data):

 data_processed = []
 data_processed.append(int(data["ram_gb"]))
 data_processed.append(int(data["ssd"]))
 data_processed.append(int(data["hdd"]))
 data_processed.append(int(data["graphic_card"]))
 data_processed.append(int(data["warranty"]))
 data_processed.append(1) if data["brand"] == "asus" else data_processed.append(0)
 data_processed.append(1) if data["brand"] == "dell" else data_processed.append(0)
 data_processed.append(1) if data["brand"] == "hp" else data_processed.append(0)
 data_processed.append(1) if data["brand"] == "lenovo" else data_processed.append(0)
 data_processed.append(1) if data["brand"] == "other" else data_processed.append(0)
 data_processed.append(1) if data["processor_brand"] == "amd" else data_processed.append(0)
 data_processed.append(1) if data["processor_brand"] == "intel" else data_processed.append(0)
 data_processed.append(1) if data["processor_brand"] == "m1" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "core i3" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "core i5" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "core i7" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "other" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "ryzen 5" else data_processed.append(0)
 data_processed.append(1) if data["processor_name"] == "ryzen 7" else data_processed.append(0)
 data_processed.append(1) if data["os"] == "other" else data_processed.append(0)
 data_processed.append(1) if data["os"] == "windows" else data_processed.append(0)
 data_processed.append(1) if data["weight"] == "casual" else data_processed.append(0)
 data_processed.append(1) if data["weight"] == "gaming" else data_processed.append(0)
 data_processed.append(1) if data["weight"] == "thinnlight" else data_processed.append(0)
 data_processed.append(1) if data["touchscreen"] == "0" else data_processed.append(0)
 data_processed.append(1) if data["touchscreen"] == "1" else data_processed.append(0)
 data_processed.append(1) if data["ram_type"] == "ddr4" else data_processed.append(0)
 data_processed.append(1) if data["ram_type"] == "other" else data_processed.append(0)
 data_processed.append(1) if data["os_bit"] == "32" else data_processed.append(0)
 data_processed.append(1) if data["os_bit"] == "64" else data_processed.append(0)

 return data_processed
```

Os dados categóricos viram 0 ou 1 para cada coluna de treinamento.

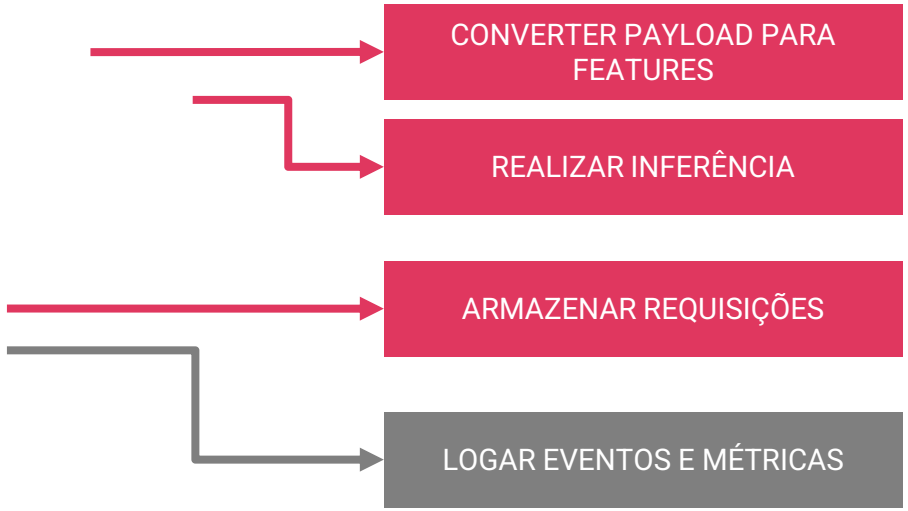
Os dados numéricos permanecem como tal, porém com o tipo de dado convertido.

Ao final teremos um vetor de entrada puramente numérico, o que é esperado para interferência de um modelo.

REALIZAR **INFERÊNCIA**

A parte principal da API, onde é chamado os demais módulos, carregado o modelo e executado.

```
def handler(event, context):

print(event)
data = event["data"]
print(data)
data_processed = prepare_payload(data)
prediction = model.predict([data_processed])

print(prediction)

prediction = int(prediction[0])

write_real_data(data, prediction)
input_metrics(data, prediction)

return {
'statusCode': 200,
'prediction': int(prediction),
'version': model_version
}
```



As saídas em console (print) são automaticamente armazenadas como registro do Cloudwatch.

Opcionalmente podemos utilizar alguma classe de Logging já existente do Python para um controle mais flexível do que deverá ser armazenado.

ARMAZENAR REQUISIÇÕES

Análises de desvios de dados são essenciais depois que um modelo é colocado em produção. Por este motivo precisamos de uma forma de armazenar cada requisição em um arquivo CSV num bucket S3.

```
def write_real_data(data, prediction):

  now = datetime.now()
  now_formatted = now.strftime("%d-%m-%Y %H:%M")

  file_name = f"{now.strftime('%Y-%m-%d')}_laptop_prediction_data.csv"
  data["price"] = prediction
  data["timestamp"] = now_formatted
  data["model_version"] = model_version
  s3 = boto3.client("s3")
  bucket_name = "fiap-ds-mlops"
  s3_path = "laptop-prediction-real-data"

  try:
    existing_object = s3.get_object(Bucket=bucket_name, Key=f'{s3_path}/{file_name}')
    existing_data = existing_object['Body'].read().decode('utf-8').strip().split('\n')
    existing_data.append(','.join(map(str, data.values())))
    updated_content = '\n'.join(existing_data)
  except s3.exceptions.NoSuchKey:
    # Se o arquivo não existir, cria um novo
   updated_content = ','.join(data.keys()) + '\n' + ','.join(map(str, data.values()))


  s3.put_object(Body=updated_content, Bucket=bucket_name, Key=f'{s3_path}/{file_name}')
```

O arquivo será acumulado por dia.

Adicionando colunas além do treinamento: timestamp da execução, previsão e versão do modelo.

O S3 não contém uma função de acumular (append), por isso o arquivo é lido antes de escrever com os dados novos.

Se o arquivo não existir (novo dia), será criado um novo.
Note que estamos escrevendo as colunas dos dados como primeira linha.

Esta arquitetura pode evoluir para uma solução baseado em eventos e um processador serializar a escrita/leitura do arquivo CSV para acumular dados e evitar, apesar de raro, eventuais acessos simultâneos.

LOGAR EVENTOS E MÉTRICAS

As métricas personalizadas do Cloudwatch permite que enviemos informações das requisições e assim acompanhar a distribuição dos dados consumidos em tempo real. Já as métricas de infraestrutura já são calculadas automaticamente, sem a necessidade de enviar,.

```
def input_metrics(data, prediction):
 cloudwatch.put_metric_data(
 MetricData = [
 {
 'MetricName': 'Price Prediction’,
 'Value': prediction,
 'Dimensions': [{'Name': "Currency", 'Value': "INR"}]
 },
 ], Namespace='Laptop Pricing Model’)

 for key, value in data.items():
  cloudwatch.put_metric_data(
  MetricData = [
  {
  'MetricName': 'Latptop Feature ‘,
  'Value': 1,
  'Unit': 'Count’,
  'Dimensions': [{'Name': key, 'Value': str(value)}]
  },
  ], Namespace='Laptop Pricing Features')
```

Métrica numérica

Métrica cumulativa

DOCKERFILE DA API PARA SERVIR UM MODELO

A função para consumir a API e servir numa API será realizada por um Lambda. Por isso já existe uma imagem base oficial da AWS que contém tudo o que é necessário para o container funcionar nesta condição.

```
# Use a imagem oficial Python como base
FROM public.ecr.aws/lambda/python:3.10
```

Obtendo imagem de um registry (Public ECR)

```
# Copy requirements.txt
COPY requirements.txt ${LAMBDA_TASK_ROOT}
```

Copia arquivo de dependências local para o caminho onde a função é armazenada

```
# Install the specified packages
RUN pip install -r requirements.txt
```

Instala as dependências

```
# Copy function code
COPY src/app.py ${LAMBDA_TASK_ROOT}
```

Copia script Python da função

```
# Copy model file
COPY model/model.pkl ${LAMBDA_TASK_ROOT}/model/
```

Copia modelo binário

```
# Copy version file
COPY model/model_metadata.json ${LAMBDA_TASK_ROOT}/model
```

Copia metadado do modelo

```
# Set the CMD to your handler (could also be done as a
parameter override outside of the Dockerfile)
CMD [ "app.handler" ]
```

Comando padrão a ser executado ao iniciar o container. Indica o arquivo .py seguido do nome da função

https://docs.docker.com/engine/reference/builder/

EXECUTANDO CONTAINER **LOCAL**

Depois de criar a especificação, vamos montar a imagem, construir o container, copiar alguns arquivos para teste local e abrir o modo iterativo.

```
docker build --platform linux/amd64 -t laptop-pricing-api-brl .
[+] Building 889.8s (11/11) FINISHED
docker:default
 => [internal] load .dockerignore
0.3s
 => => transferring context: 2B
…
```

O Platform precisa ser o mesmo do runner escolhido ao criar uma Lambda

Executar o container no modo detached

```
docker run -d laptop-pricing-api
```

Copiar os arquivos de teste para invocar a função.

```
docker cp test.py d23b4b71a097:/var/task
docker cp data.json d23b4b71a097:/var/task
```

Abrir o container no modo interativo.

```
docker exec -it d23b4b71a097 bash
```

Instalar o AWS CLI e configurar

```
yum update –y
yum install awscli -y

aws configure
```

Em produção não precisamos instalar o AWS CLI pois as permissões são concedidas no próprio Lambda. Não será levado em produção estes arquivos adicionais, eles servem apenas para testar localmente.

Finalmente executar o teste

```
python test.py
```

CRIANDO UM **REGISTRY PRIVADO**

Aplicações corporativas utilizam um repositório de registro privado de forma semelhante com o versionamento dos códigos.

Quando for criar um repositório, marque a opção privado e indique o nome do repositório.

Cada repositório é dedicado uma imagem de container.

OBTENDO COMANDOS DO **REGISTRY**

FIAP MBA+

Comandos push para laptop-pricing-4dtsr

macOS / Linux | Windows

Verifique se você tem a versão mais recente do AWS CLI e do Docker instalada. Para obter mais informações, consulte Primeiros passos com o Amazon ECR.

Use as etapas a seguir para autenticar e enviar uma imagem para o repositório. Para obter outros métodos de autenticação do registro, incluindo o auxiliar de credenciais do Amazon ECR, consulte Autenticação do registro.

1. Recupere um token de autenticação e autentique seu cliente Docker em seu registro.
   Use o AWS CLI:
   ```
   aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 989944764342.dkr.ecr.us-east-1.amazonaws.com
   ```
   Observação: se você receber um erro ao usar o AWS CLI, verifique se você tem a versão mais recente do AWS CLI e do Docker instalada.
2. Crie sua imagem do Docker usando o comando a seguir. Para obter informações sobre como criar um arquivo Docker do zero, consulte as instruções aqui. Você pode pular esta etapa se sua imagem já foi criada.
   ```
   docker build -t laptop-pricing-4dtsr .
   ```
3. Depois que a compilação for concluída, marque sua imagem para que você possa enviá-la para este repositório:
   ```
   docker tag laptop-pricing-4dtsr:latest 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-4dtsr:latest
   ```
4. Execute o seguinte comando para enviar essa imagem para o repositório da AWS recém-criado:
   ```
   docker push 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-4dtsr:latest
   ```

Fechar

Amazon ECR > Registro privado > Repositórios > laptop-pricing-4dtsr

laptop-pricing-4dtsr

Visualizar comandos push | Editar

Imagens (1)

Excluir | Detalhes | Verificar

Pesquisar artefatos

Etiqueta de imagem	Tipo de artefato	Enviado por push em	Tamanho (MB)	URI de imagem	Resumo	Status da verificação	Vulnerabilidades
latest	Image	21 de fevereiro de 2024, 23:11:11 (UTC-03)	1023.69	Copiar URI	sha256:9[illegible]bc10411609d1...	-	-

Com o respositório criado, obtemos os comandos para autenticar e enviar a imagem local para o repositório remoto.

Os comandos são obtidos ao clicar em “Visualizar comandos push” no painel do repositório.

CARREGANDO IMAGEM NO **REGISTRY**

FIAP MBA+

Após validarmos que o container foi executado como esperado e a imagem está validada, precisamos enviar para um registro para que possa ser utilizada em novas implantações
Para aplicações corporativas utilizamos registros de imagem privados, como o Elastic Container Registry (ECR da AWS. O DockerHub é outra possibilidade, mas neste caso para imagens públicas. O ECR também tem repositórios públicos para distribuição de imagens padrão compatíveis e validadas com os serviços da AWS.

Primeiro incluímos o tag com o nome da imagem e sua versão. É comum utilizarmos o nome “latest” para indicar que é a última versão disponível (e atualizada).

```
docker tag laptop-pricing-api:latest 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-api:latest
```

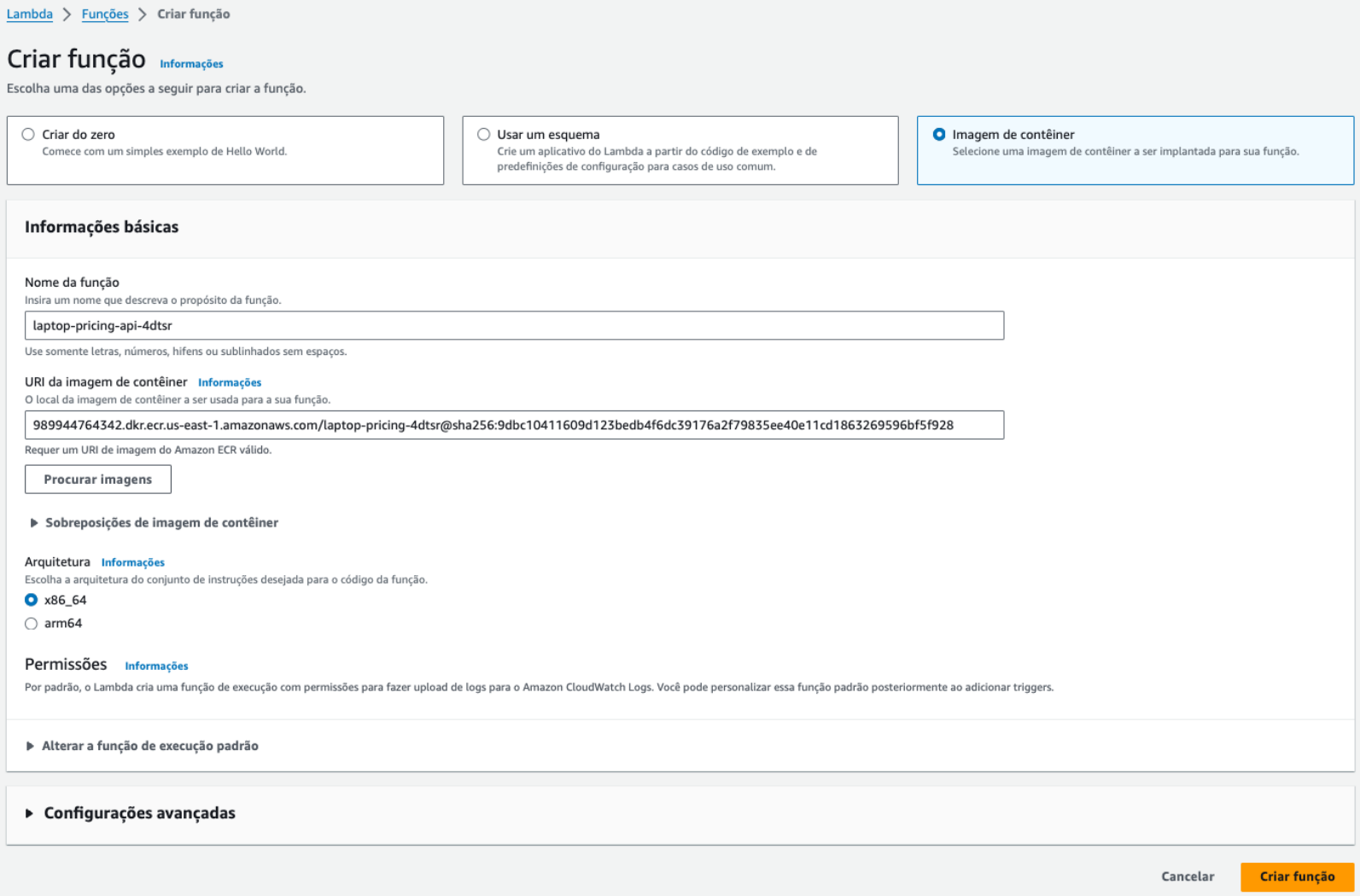
Depois iremos enviar os dados para o registro. Alguns repositórios necessitam de login e autenticação antes, como o Docker. Como estamos com o AWS CLI já autenticado com as devidas permissões, o acesso já está autorizado, mas precisamos enviar o comando para autenticar.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 989944764342.dkr.ecr.us-east-1.amazonaws.comD
```

```
docker push 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-api:latest
```

Existem serviços baseados em containers da qual quando há uma nova versão “latest” no repositório inicia o processo de entrega contínua atualizando automaticamente o recurso. Não é o caso das funções Lambda. É necessário atualizar via comando a nova versão da imagem mesmo que seja a mesma.

CRIANDO **LAMBDA CONTEINER**



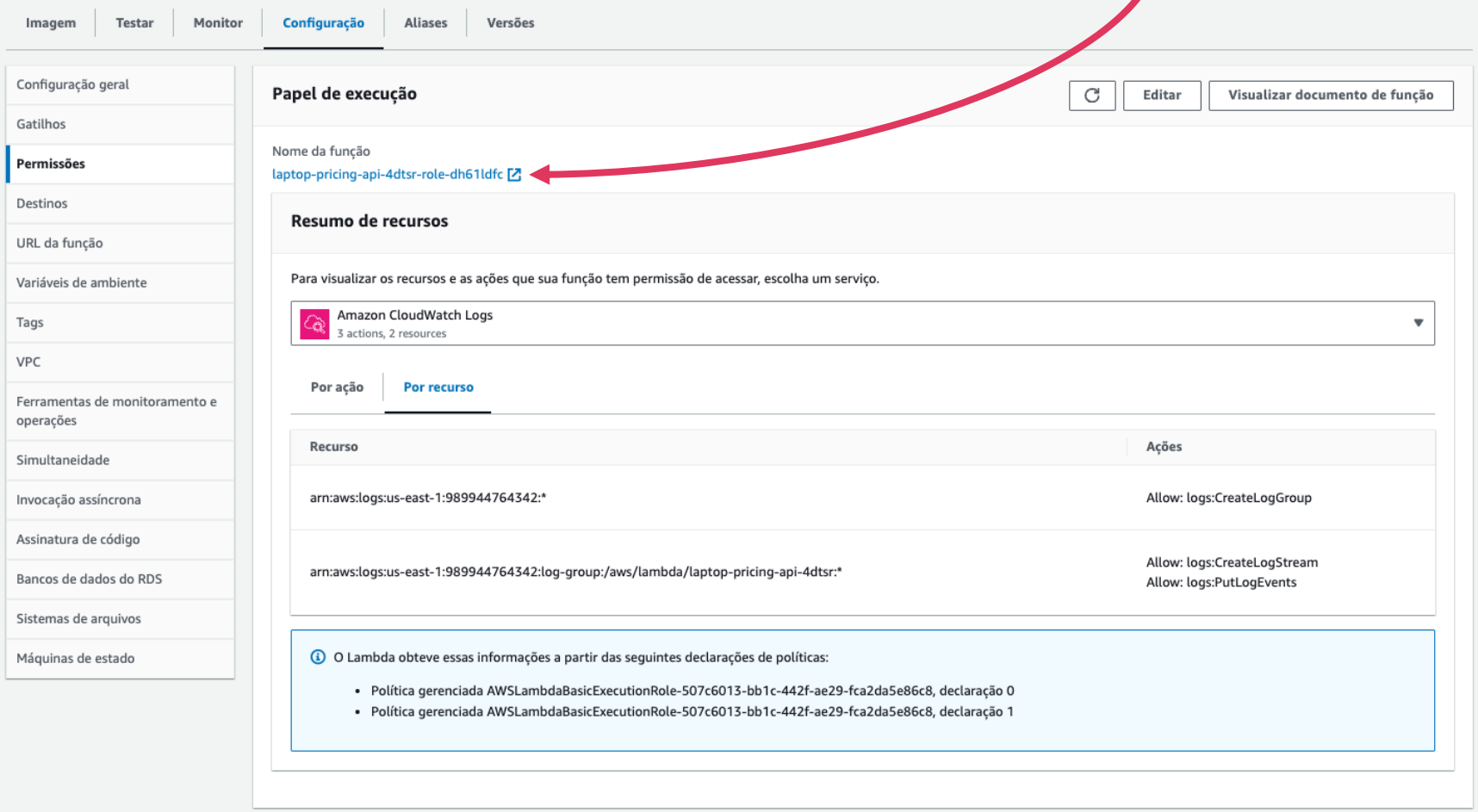
Para criar um Lambda baseado em contêiner selecionamos a opção “Imagem de contêiner”.

Logo depois indicamos qual a URL da imagem, buscando pelos repositórios disponíveis.

Depois selecionamos a arquitetura, recomenda-se ARM64 por ser mais eficiente computacionalmente e menos custosa.

ADICIONANDO **PERMISSÕES**

Após a função ter sido implantada, precisamos incluir as permissões adicionais (além do padrão) que permitirá adicionar métricas e alarmes customizados além de escrita em bucket S3.
Vamos clicar no nome da função abaixo para ir ao IAM realizar as modificações.



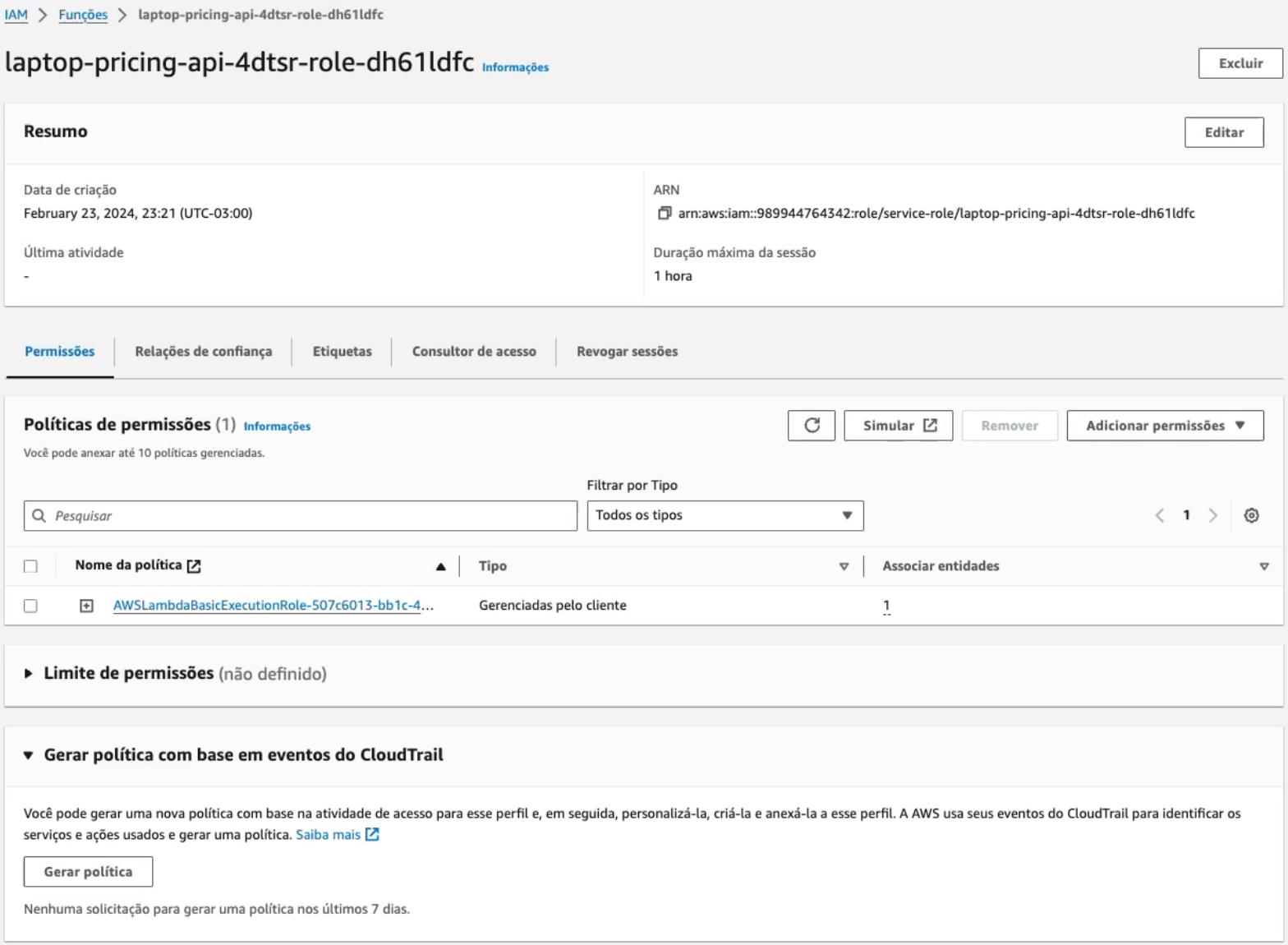
ADICIONANDO **PERMISSÕES**

Criando políticas em linha incluímos as permissões que precisamos:

No Cloudwatch: PutMetricAlarm e PutMetricData.

No S3: acesso de leitura e escrita (para simplificar adicionamos todas as permissões).

Em cloud utilizamos sempre a regra do menor privilégio. Ou seja, as permissões precisam ser exatamente as mínimas necessárias tanto de operações (o quê) e também onde (qual componente, por exemplo bucket).

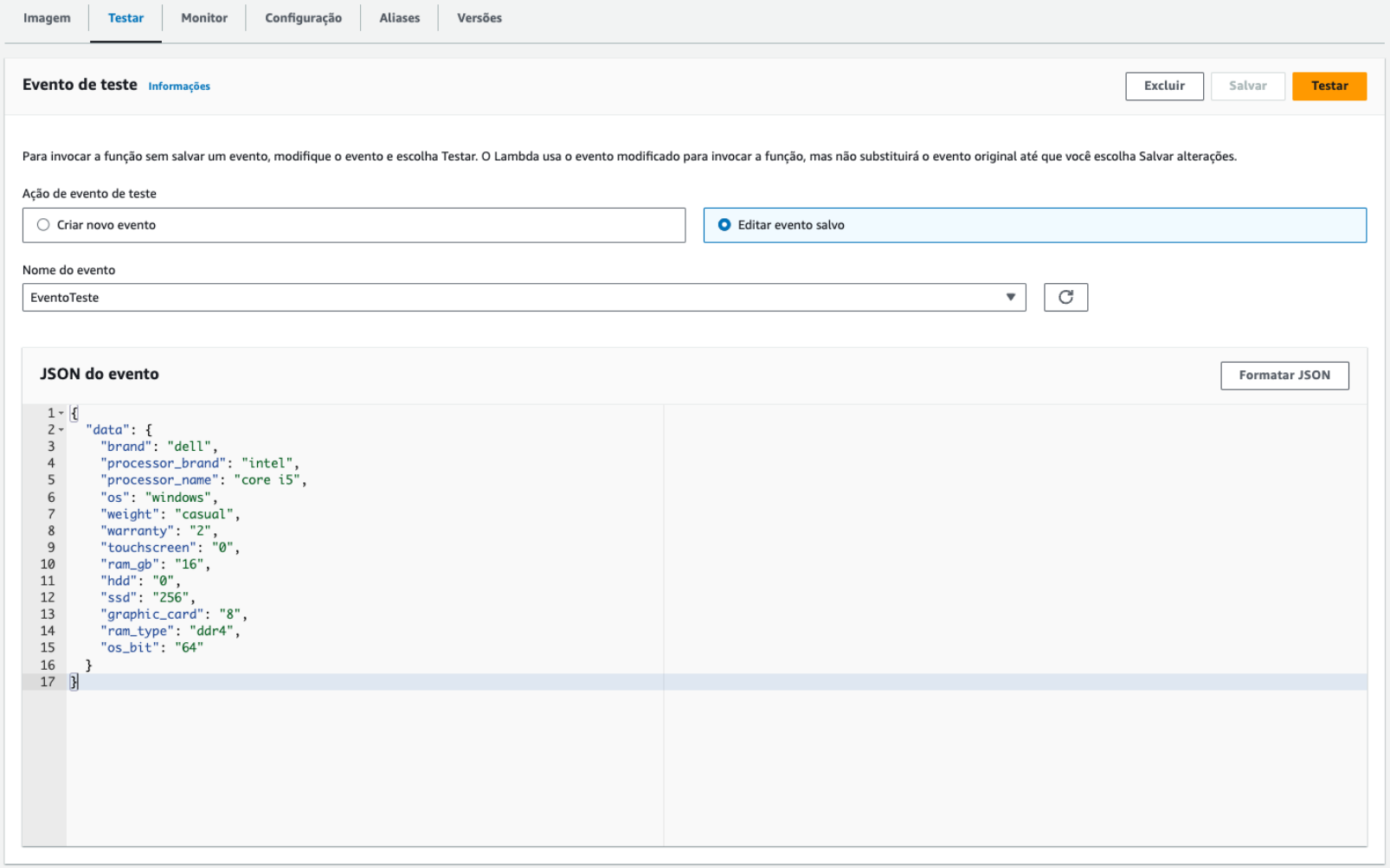


TUDO PRONTO: **EVENTO DE TESTE**

A API está pronta para uso.
Nosso primeiro teste será na console do Lambda com um evento de teste. Será utilizado o mesmo payload que a API irá receber.

Payload

```
{
  "data": {
    "brand": "dell",
    "processor_brand": "intel",
    "processor_name": "core i5",
    "os": "windows",
    "weight": "casual",
    "warranty": "2",
    "touchscreen": "0",
    "ram_gb": "16",
    "hdd": "0",
    "ssd": "256",
    "graphic_card": "8",
    "ram_type": "ddr4",
    "os_bit": "64"
  }
}
```



ANALISANDO **TESTE**

O contêiner quando é recém implantado pode apresentar um erro de timeout pois o padrão do tempo de execução máxima é de 3 segundos. Caso apresente esse erro, tente novamente.
Com o contêiner pronto para execução, o retorno esperado é código 200 (statusCode) e o valor da predição no JSON de resposta.

Imagem | Testar | Monitor | Configuração | Aliases | Versões

Execução da função: bem-sucedida (logs)

▼ Detalhes

A área abaixo mostra os últimos 4 KB do log de execução.

```
{
  "statusCode": 200,
  "prediction": 79922,
  "version": "v0.0.1"
}
```

Resumo

Código SHA-256 9dbc10411609d123bedb4f6dc39176a2f79835ee40e11cd1863269596bf5f928	Execution time há 11 segundos (23 de fevereiro de 2024 às 23:35 BRT)
ID da solicitação 86114b5d-d583-4db2-bc17-1600a6b5e97d	Versão da função $LATEST
Duração 1243.75 ms	Período cobrado 1244 ms
Recursos configurados 128 MB	Memória máxima usada 127 MB

Resultado de saída do log

A seção abaixo mostra as chamadas de registro em log no seu código. Clique aqui para visualizar o grupo de logs do CloudWatch correspondente.

```
START RequestId: 86114b5d-d583-4db2-bc17-1600a6b5e97d Version: $LATEST
{'data': {'brand': 'dell', 'processor_brand': 'intel', 'processor_name': 'core i5', 'os': 'windows', 'weight': 'casual', 'warranty': '2', 'touchscreen': '0', 'ram_gb': '16', 'hdd': '0', 'ssd': '256',
'graphic_card': '8', 'ram_type': 'ddr4', 'os_bit': '64'}}
LambdaContext([aws_request_id=86114b5d-d583-4db2-bc17-1600a6b5e97d,log_group_name=/aws/lambda/laptop-pricing-api-
4dtsr,log_stream_name=2024/02/24/[$LATEST]640c3e8a2fb644bdace24608edcb6bcf,function_name=laptop-pricing-api-
4dtsr,memory_limit_in_mb=128,function_version=$LATEST,invoked_function_arn=arn:aws:lambda:us-east-1:989944764342:function:laptop-pricing-api-
4dtsr,client_context=None,identity=CognitoIdentity([cognito_identity_id=None,cognito_identity_pool_id=None])])
{'brand': 'dell', 'processor_brand': 'intel', 'processor_name': 'core i5', 'os': 'windows', 'weight': 'casual', 'warranty': '2', 'touchscreen': '0', 'ram_gb': '16', 'hdd': '0', 'ssd': '256',
'graphic_card': '8', 'ram_type': 'ddr4', 'os_bit': '64'}
Prediction: 79922
```

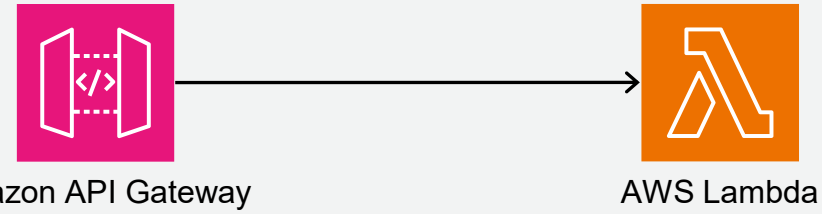
Resposta da função (API)

Log dos últimos eventos.

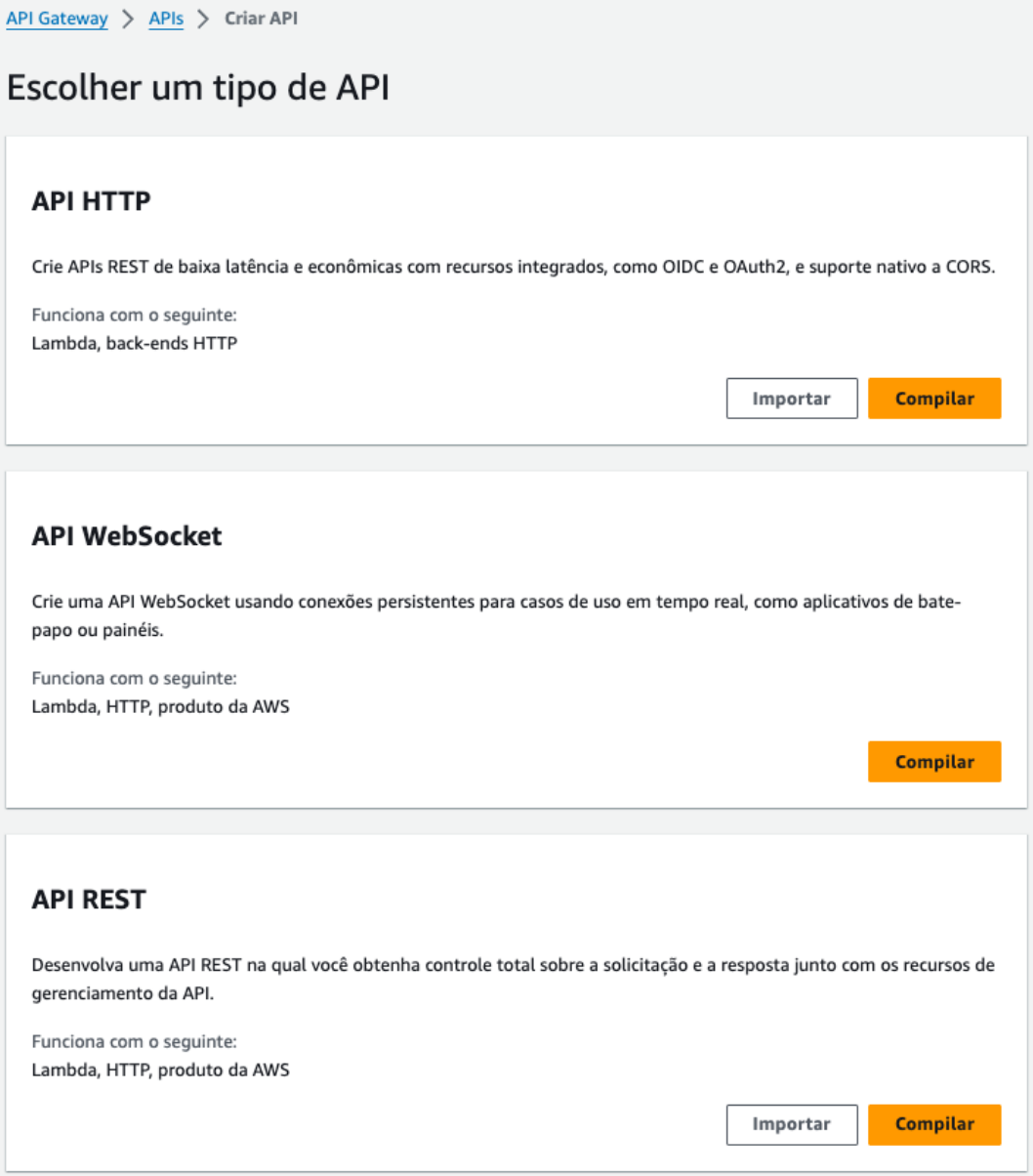
ETAPA FINAL DA API: **API GATEWAY**

Com o Lambda funcionando (função), precisamos integrar num API Gateway para um controle mais preciso de acesso, cotas de uso e mais segurança.

O API Gateway fornecerá a URL de acesso para ser consumido pelo frontend ou outros serviços.



Nosso tipo de API Gateway é API REST.



CARREGANDO IMAGEM NO **REGISTRY**

Depois de selecionar o tipo de API para criar podemos criar uma nova (do zero) ou utilizar algum acelerador disponível, como importar da especificação OpenAPI (antigo Swagger), clonar uma existente ou utilizar uma de exemplo.

Preencher o nome da API (ela ficará exposta no endereço do endpoint, por isso use um nome identificável e sugestivo).

Utilize a descrição para informar detalhes da API, se necessário.

API Gateway > APIs > Criar API > Criar API REST

Criar API REST

Detalhes da API

- (●) **API nova** — Crie uma nova API REST.
- ( ) **Clonar a API existente** — Crie uma cópia de uma API nessa conta da AWS.
- ( ) **Importar API** — Importe uma API de uma definição de OpenAPI.
- ( ) **API de exemplo** — Saiba mais sobre o API Gateway com uma API de exemplo.

Nome da API

LaptopPricing4DTSR

Descrição: *opcional*

Tipo de endpoint de API

As APIs regionais são implantadas na região atual da AWS. As APIs otimizadas para a borda direcionam as solicitações para o ponto de presença do CloudFront mais próximo. As APIs privadas só podem ser acessadas de VPCs.

Regional

Cancelar | Criar API

CRIAR UMA ROTA OU **RECURSO**

O primeiro passo é criar um recurso que servirá de rota para acesso.
Por exemplo, o nosso endpoint com o recurso “predict” será assim: https://<endereço do endpoint>/predict

API Gateway > APIs > Recursos - LaptopPricing4DTSR (z07uelgg6l) > Criar recurso

Criar recurso

Detalhes do recurso

Recurso proxy Informações
Os recursos de proxy lidam com solicitações para todos os sub-recursos. Para criar um recurso proxy, use um parâmetro de caminho que termine com um sinal de mais, por exemplo, {proxy+}.

Caminho do recurso: /

Nome do recurso: predict

CORS (Compartilhamento de recursos de origem cruzada) Informações
Crie um método OPTIONS que permita todas as origens, todos os métodos e vários cabeçalhos comuns.

Cancelar | Criar recurso

CRIANDO UM **MÉTODO**

Com a API e o recurso criado, podemos finalmente criar o método de consumo (“Criar método”).

No método incluiremos a integração com o Lambda para sua execução.

API Gateway > APIs > Recursos - LaptopPricing4DTSR (z07uelgg6l)

Recursos

Ações de API | Implantar API

Criar recurso

/

/predict

Detalhes do recurso

Excluir | Atualizar a documentação | Habilitar CORS

Caminho
/predict

ID do recurso
tplgon

Métodos (0)

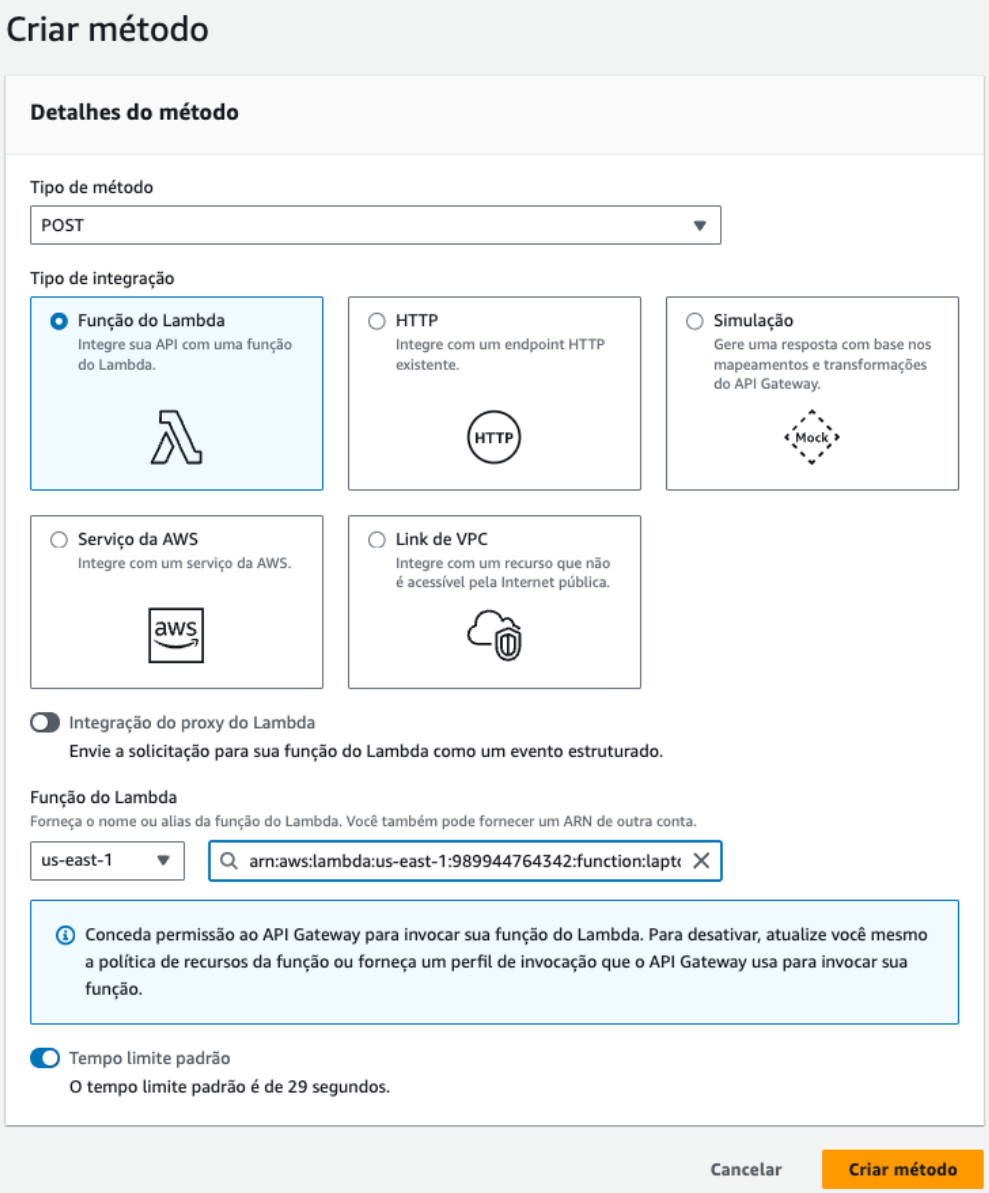
Excluir | Criar método

Tipo de método	Tipo de integração	Autorização	Chave de API

Nenhum método

Nenhum método definido.

INTEGRANDO O **MÉTODO**



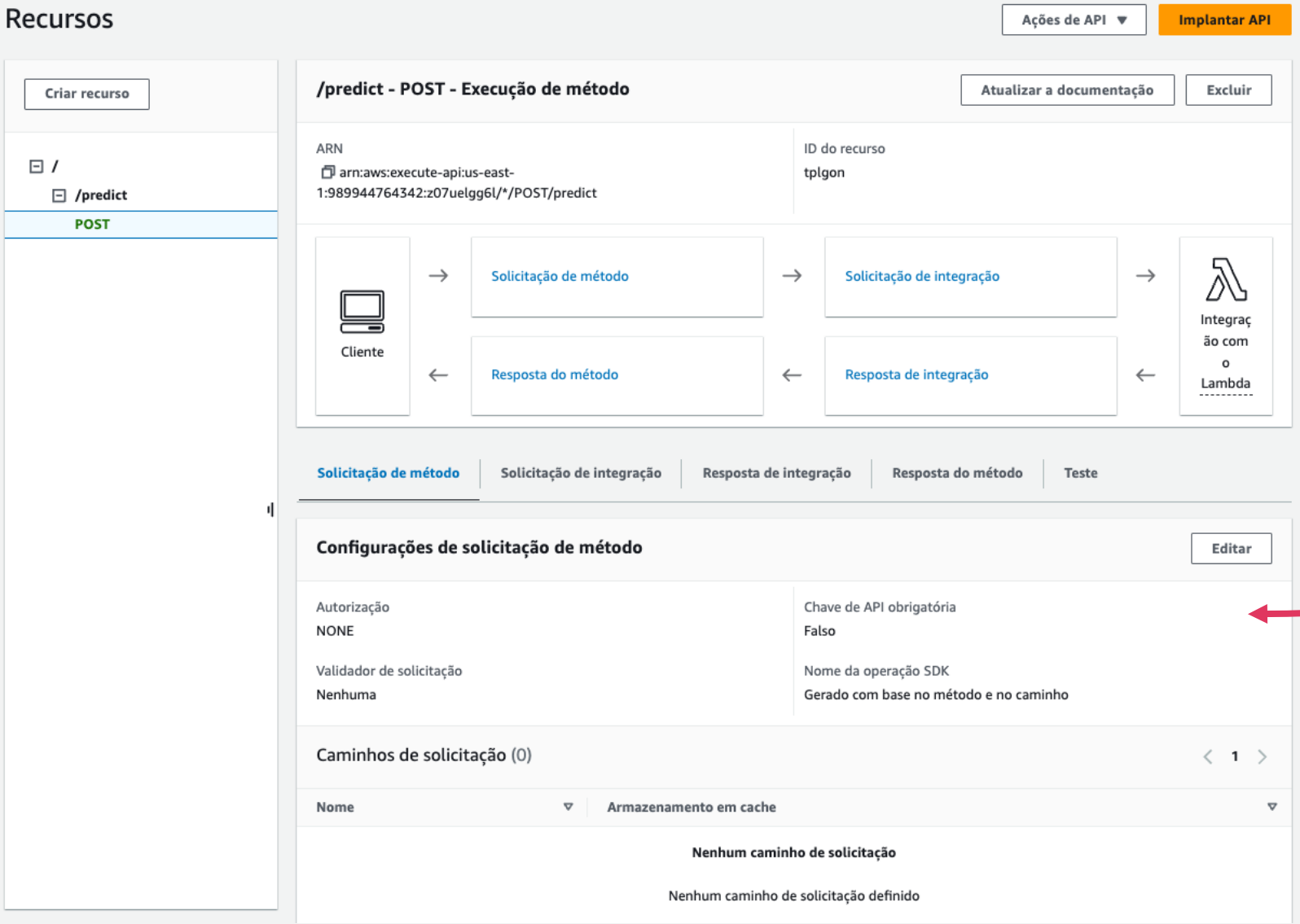
O método pode ser integrado com os serviços da AWS ou com Lambda, que é a integração mais comum.

O método para envio de dados escolhido por padrão é o POST.

Selecionaremos o Lambda e posteriormente o ARN da função de acordo com a região selecionada (us-east-1).

Mantemos o tempo limite de 29s (padrão do API Gateway), isso significa que se a requisição levar mais do que esse tempo retornará time out.

DEFININDO **MÉTODO**



O API Gateway está quase terminado.

Precisamos implantar uma versão para poder associar a um plano de consumo de chaves de acesso.

Note que no momento estamos sem chave de acesso associada a API.

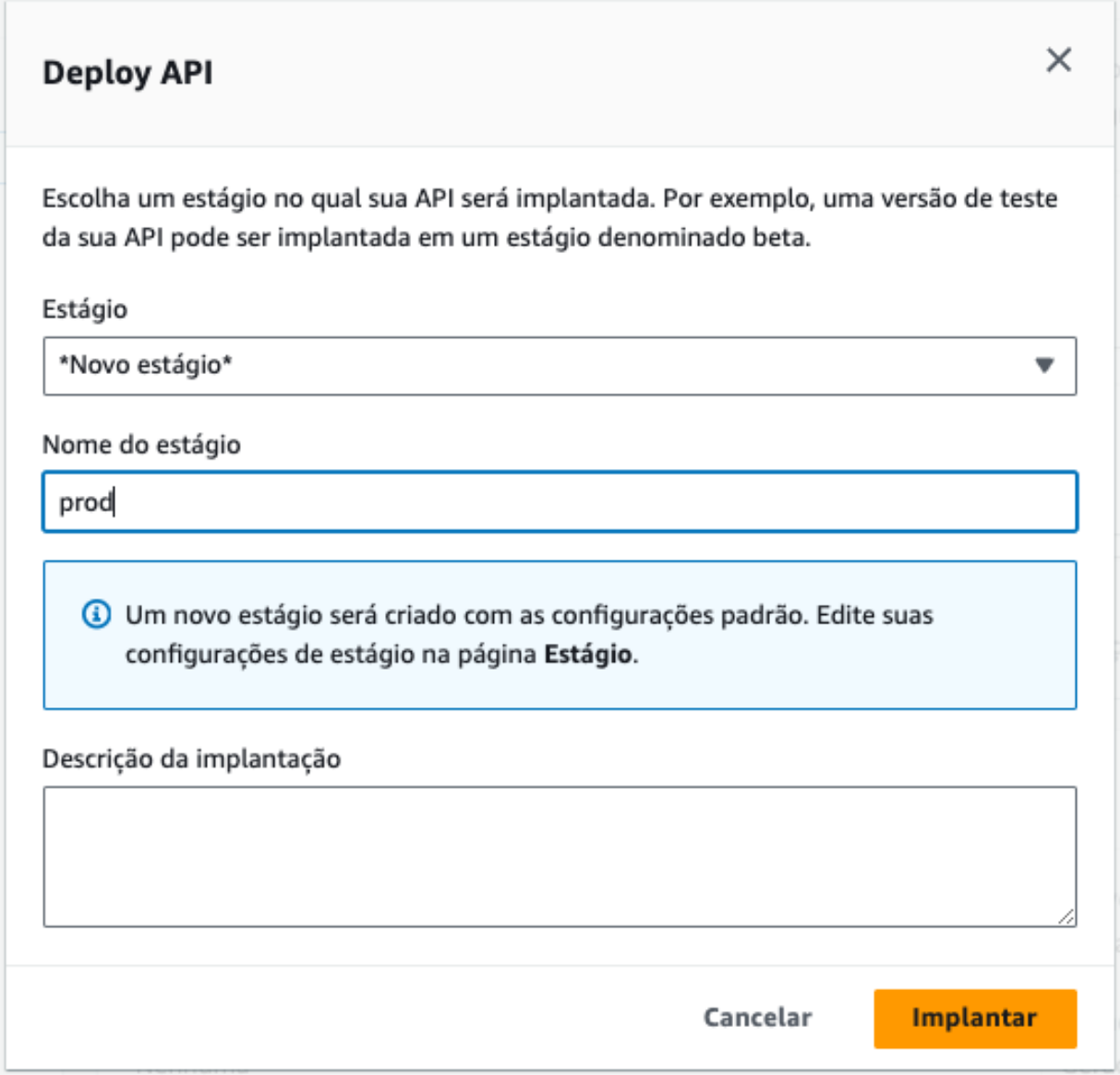
DEFININDO **ESTÁGIO**

Os estágios de uma API são como se fossem ambientes separados.

Podem ser utilizados para propósitos diferentes como desenvolvimento, produção, qualidade, etc.

No nosso caso vamos criar apenas um que será o de produção. Escolhemos o nome “prod”.

O nome do estágio vai ser incluído na definição do endereço do endpoint de acesso, portanto prefira nomes adequados para melhor identificação.



CRIANDO UM **PLANO DE USO**

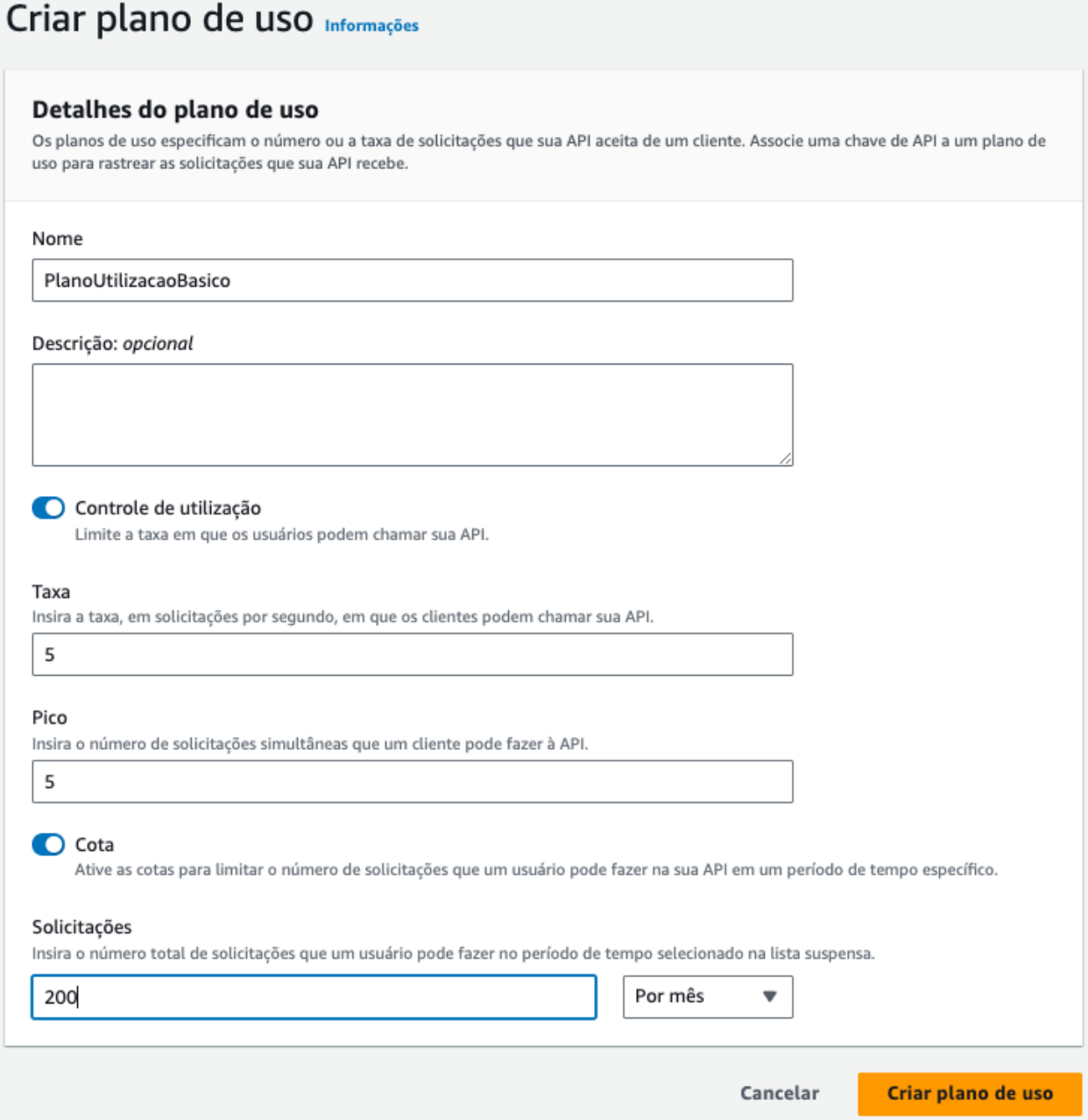
Planos de uso definem características importantes de acesso a uma API.

Definições de:

- Taxa de solicitações por segundo
- Pico de acessos simultâneos
- Cotas de acesso

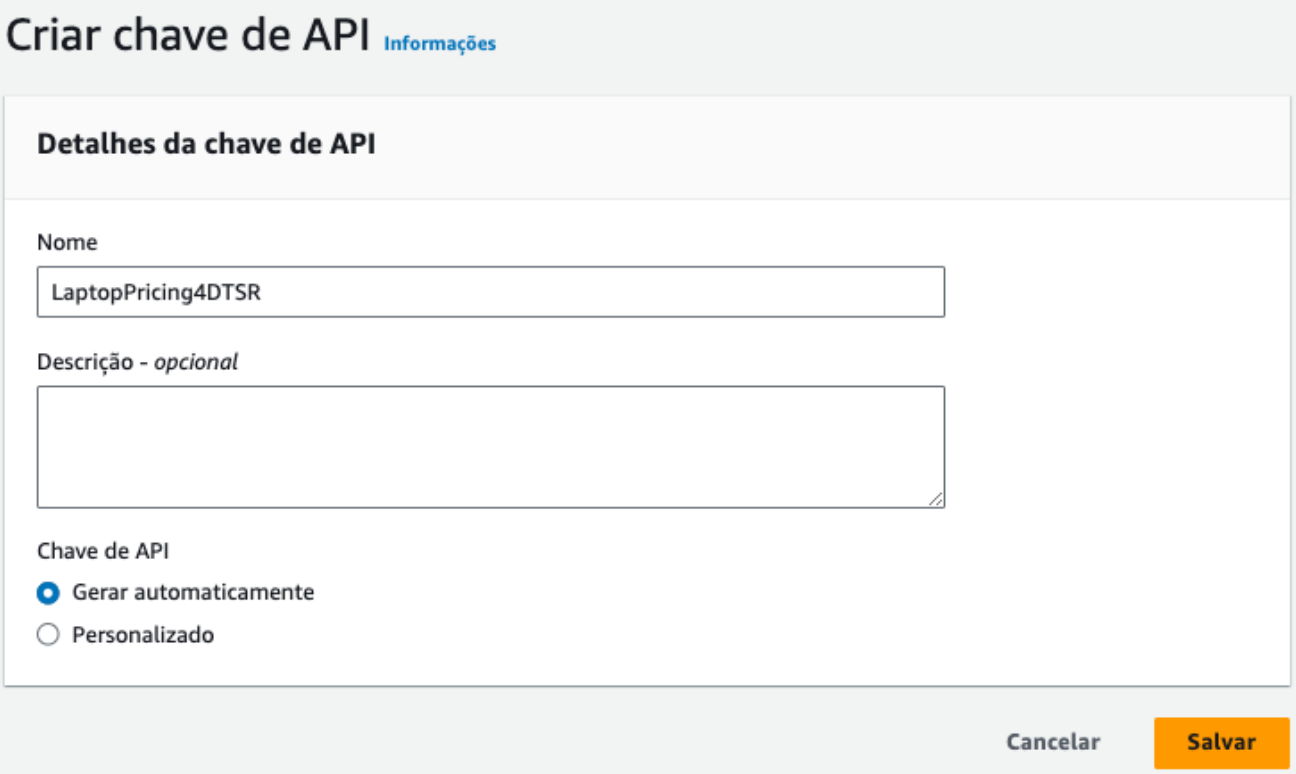
Ajudam a controlar aspectos do uso de seu serviço, especialmente quando há solicitações de serviços custosos como os LLMs (apesar de não ser o nosso caso no momento).

Na AWS o plano de uso é associado a chaves de acesso que por sua vez é associado um estágio de API.



CRIANDO **CHAVES DE ACESSO**

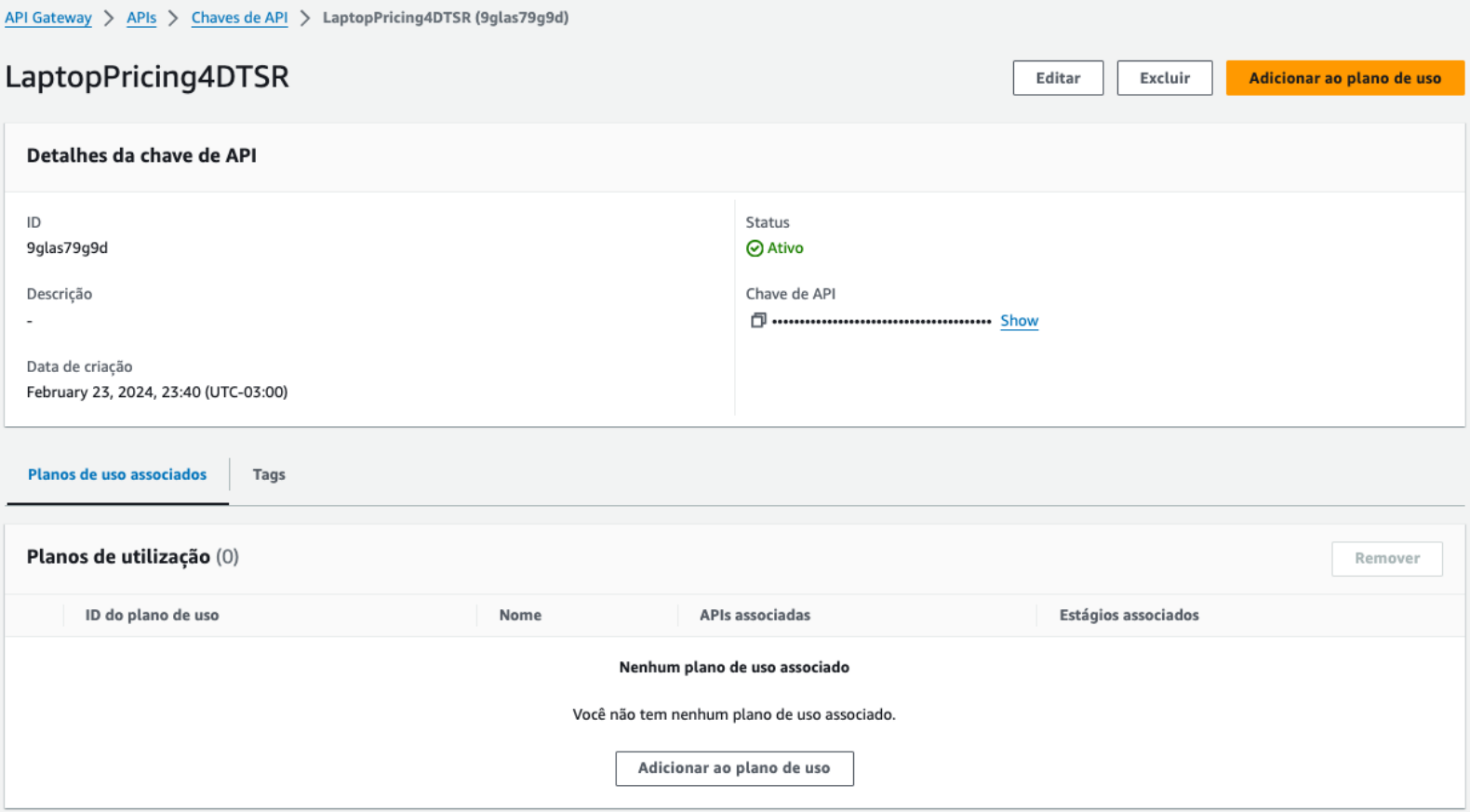
A chave de acesso pode ser gerada automaticamente ou gerada personalizada.
Chaves de acesso garantem nível de segurança de acesso da qual quem tem direito de utilizar o serviço as utiliza para autenticação.



Na requisição do serviço deverá ser incluída no header a chave x-api-key com o valor da senha (chave de acesso).

CRIANDO **CHAVES DE ACESSO**

Após criar uma chave de acesso, temos que associar ao plano de uso que criamos anteriormente. Clicamos em “Adicionar ao plano de uso”.



CRIANDO **CHAVES DE ACESSO**

FIAP MBA+

API Gateway > APIs > Chaves de API > LaptopPricing4DTSR (9glas79g9d) > Adicionar ao plano de uso

Adicionar ao plano de uso

Detalhes da chave de API

ID
9glas79g9d

Descrição
-

Data de criação
February 23, 2024, 23:40 (UTC-03:00)

Status
Ativo

Chave de API
•• Show

Adicionar chave ao plano de uso

Plano de uso
PlanoUtilizacaoBasico

Cancelar Salvar

E então associamos ao plano de uso existente.

ADICIONANDO O PLANO DE UTILIZAÇÃO AO **ESTÁGIO**

Após o plano de utilização criado, com as chaves de acesso criadas e associadas, o próximo passo é associar a um estágio de API.

API Gateway > APIs > Planos de utilização > PlanoUtilizacaoBasico

PlanoUtilizacaoBasico

Ações ▼ | Exportar dados de uso

Detalhes do plano de uso

ID do plano de uso 4if08j	Taxa 5 solicitações por segundo
Descrição -	Pico 5 solicitações
Código do produto do AWS Marketplace -	Cota 200 solicitações por mês

Estágios associados | Chaves de API associadas | Tags

Estágios associados (0) Informações

Editar | Remover | Adicionar estágio

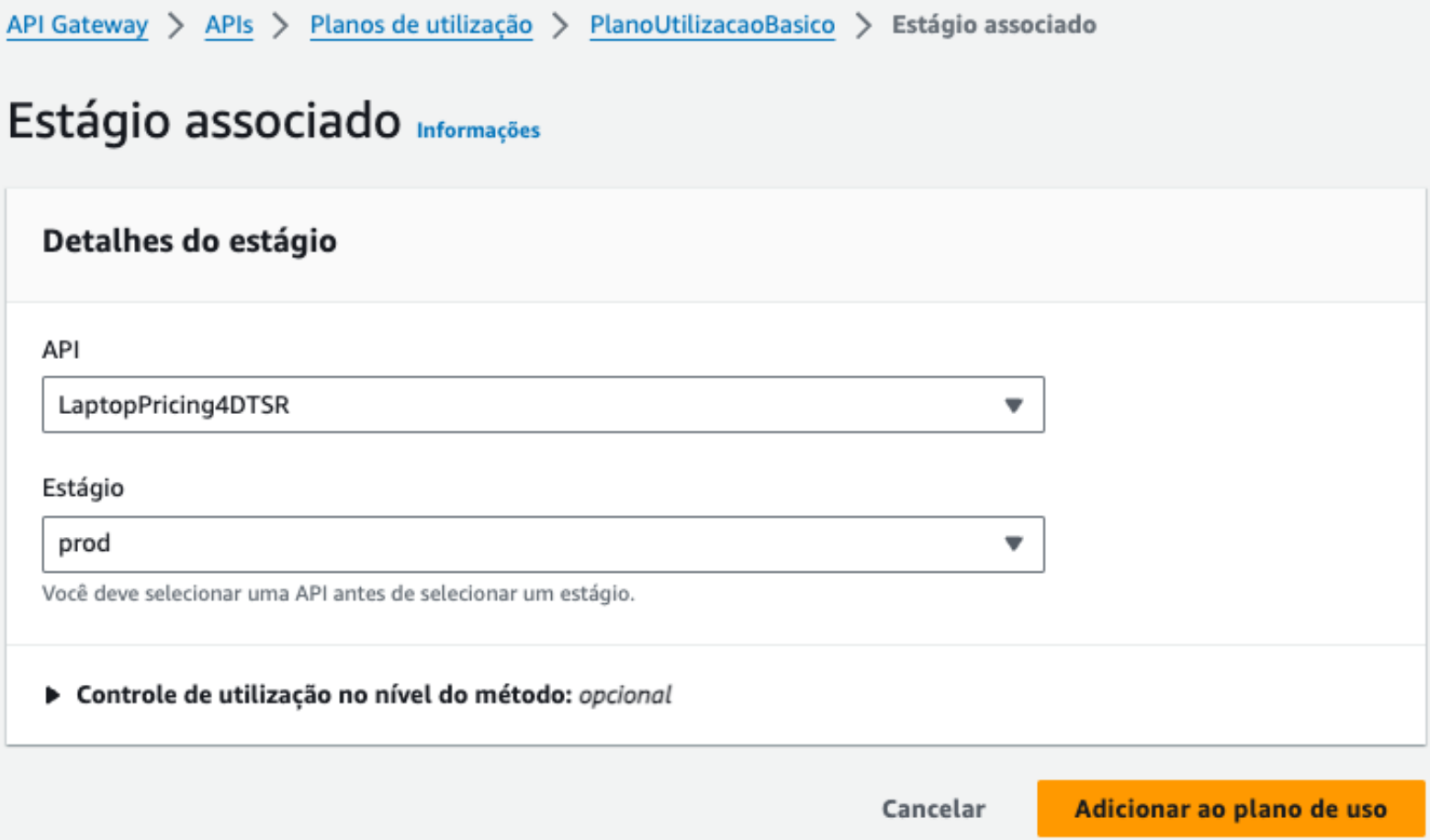
API	Estágio	Controle de utilização do método

Nenhuma estágio

Você não tem nenhum estágio.

Adicionar estágio da API

CARREGANDO IMAGEM NO **REGISTRY**



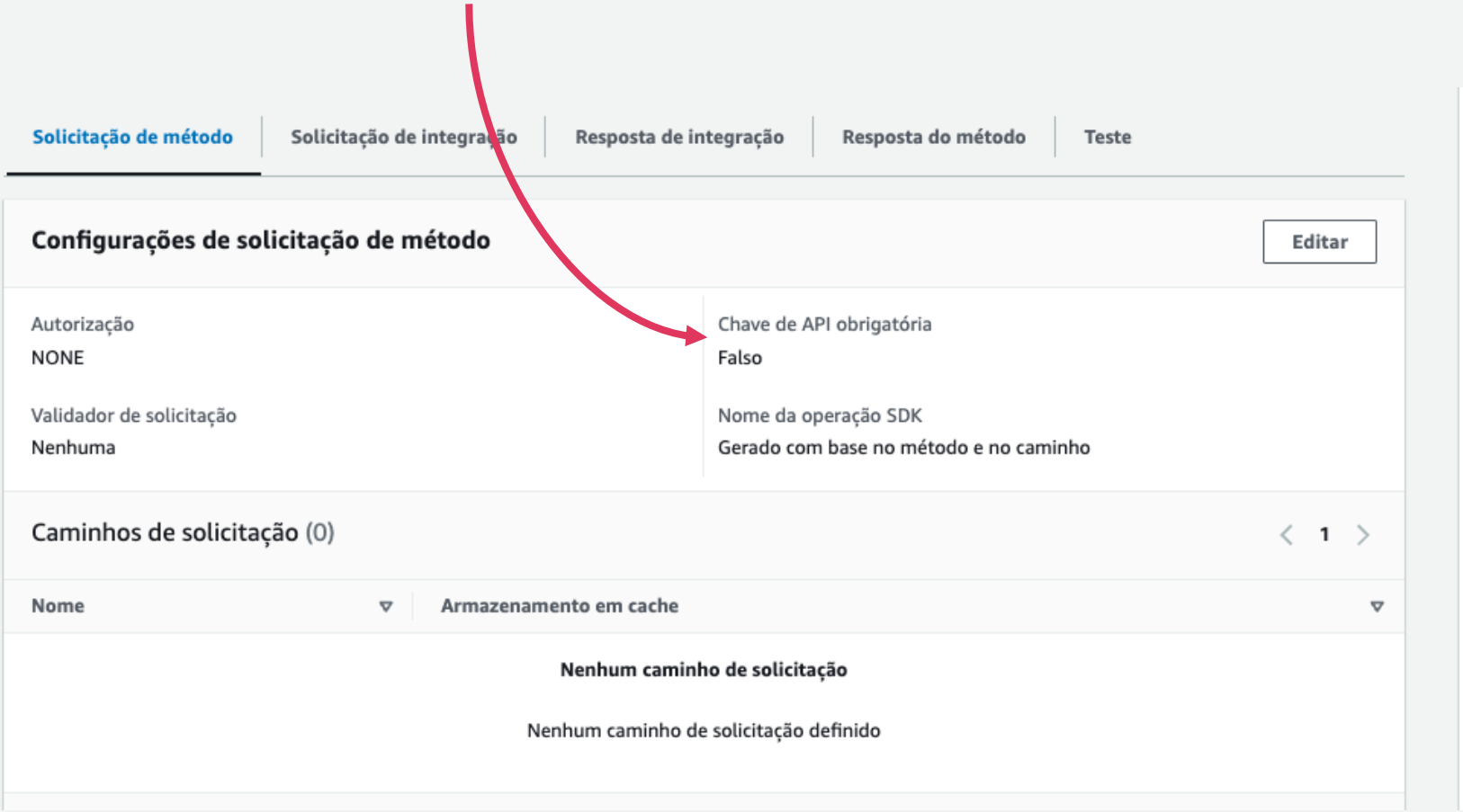
O estágio associado ao plano de uso está também associado a API da qual o estágio foi criado.

Desta forma, o plano de uso passa a ser configurado para a API que criamos no estágio selecionado.

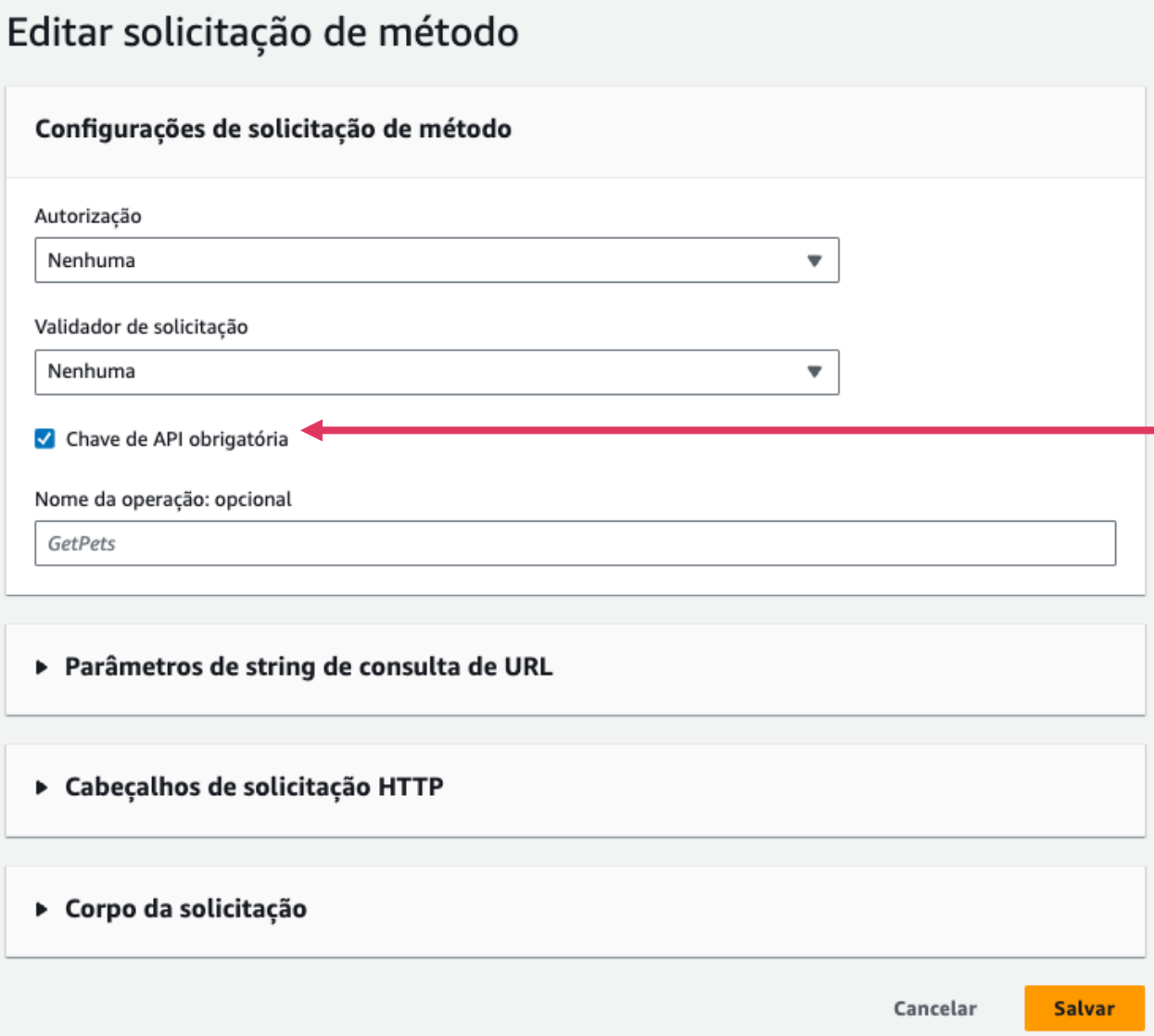
No nosso caso a API “LaptopPricing4DTSR” do estágio “prod” foi associado ao plano de uso “PlanoUtilizacaoBasico”.

ATIVANDO USO DE **CHAVES NA API**

Depois que criamos o plano de uso, chaves de acesso e associamos ao estágio e API que criamos, precisamos ativar esta funcionalidade.
Por padrão as chaves de acesso não são ativadas na API.



ATIVANDO **CHAVE DE ACESSO NA REQUISIÇÃO**

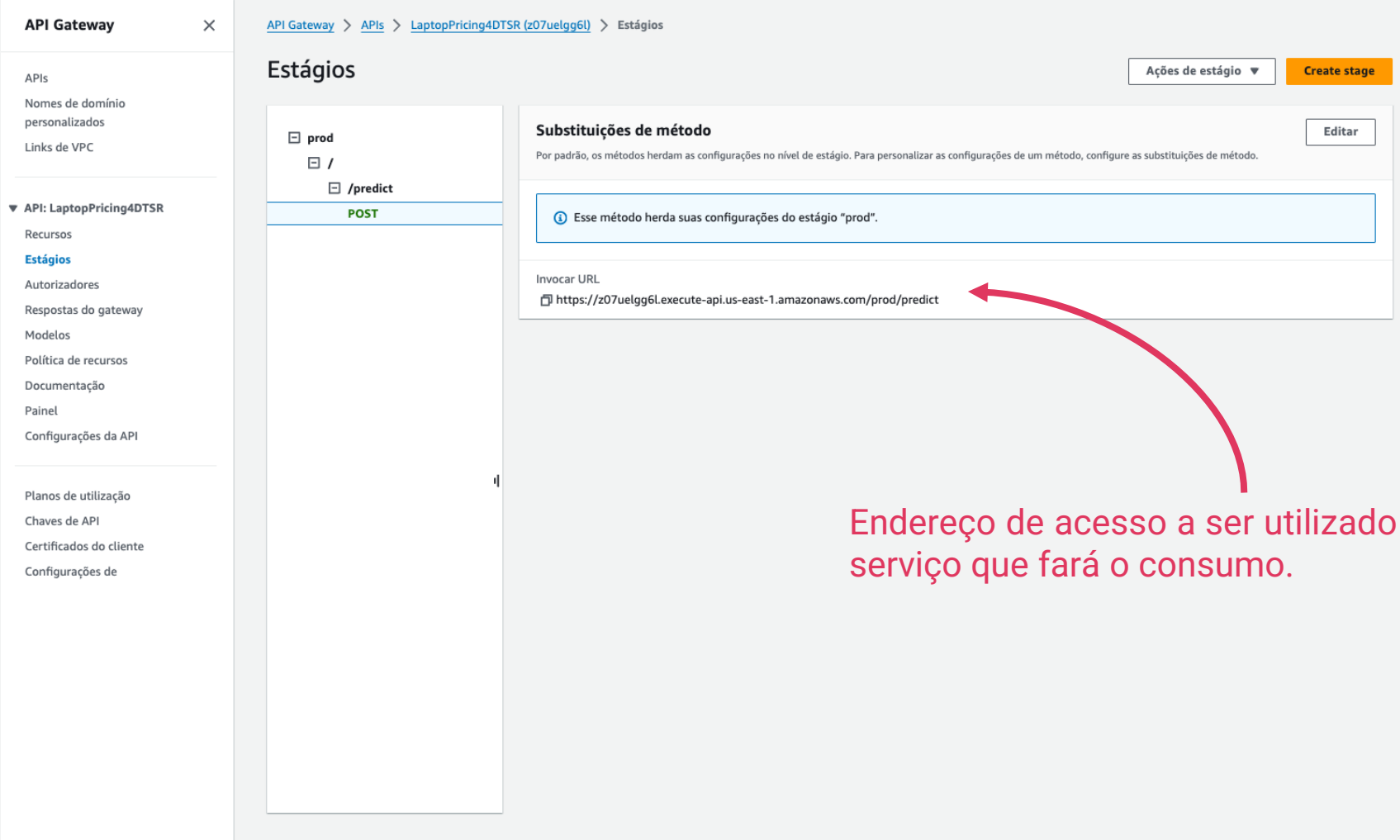


Para ativar o uso das chaves, basta clicarmos em “Chave de API obrigatória”.

Isso fará que todas as requisições tenha a chave criada anteriormente.

DEFINIÇÃO DE ENDEREÇO DE **ENDPOINT**

O endereço de endpoint do método criado fica em “Estágios”, depois clicamos no método e no verbo configurado “Post”.



Endereço de acesso a ser utilizado no frontend ou serviço que fará o consumo.

TESTANDO A API NO **TALEND**

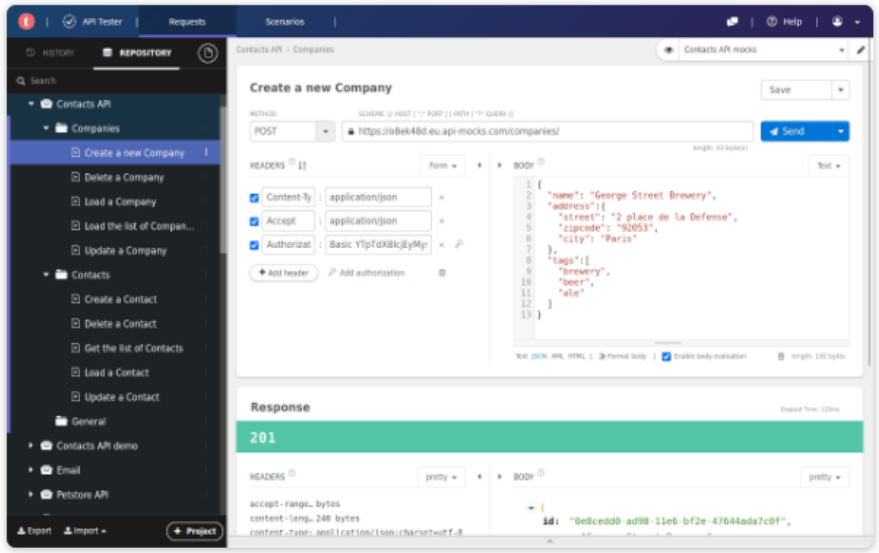
O Talend é uma extensão do Chrome utilizada para testes de API.
Ele realiza requests com possibilidades de configuração de cabeçalho, payload, etc. conforme cada serviço necessita.

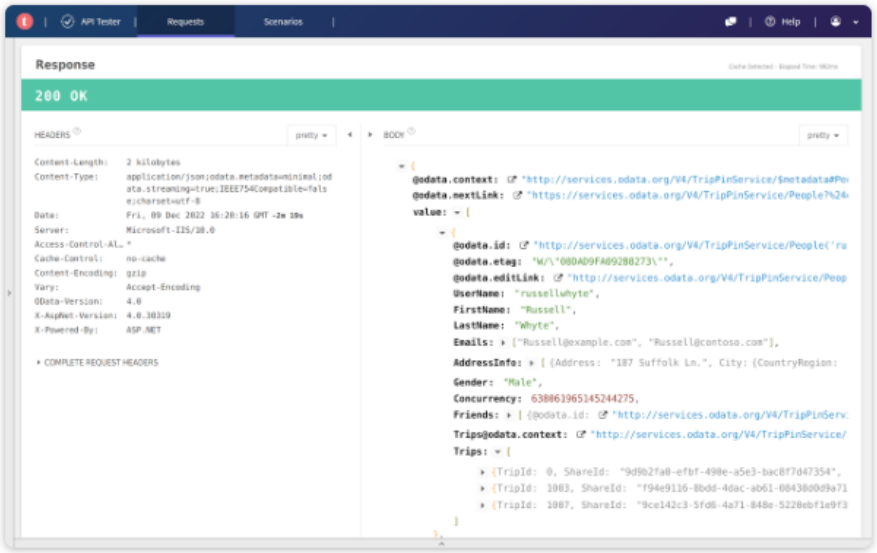
Talend API Tester - Free Edition

Remover do Chrome

Em destaque 4,8 ★ (4,2 mil notas)

Extensão | DevTools | 600.000 usuários



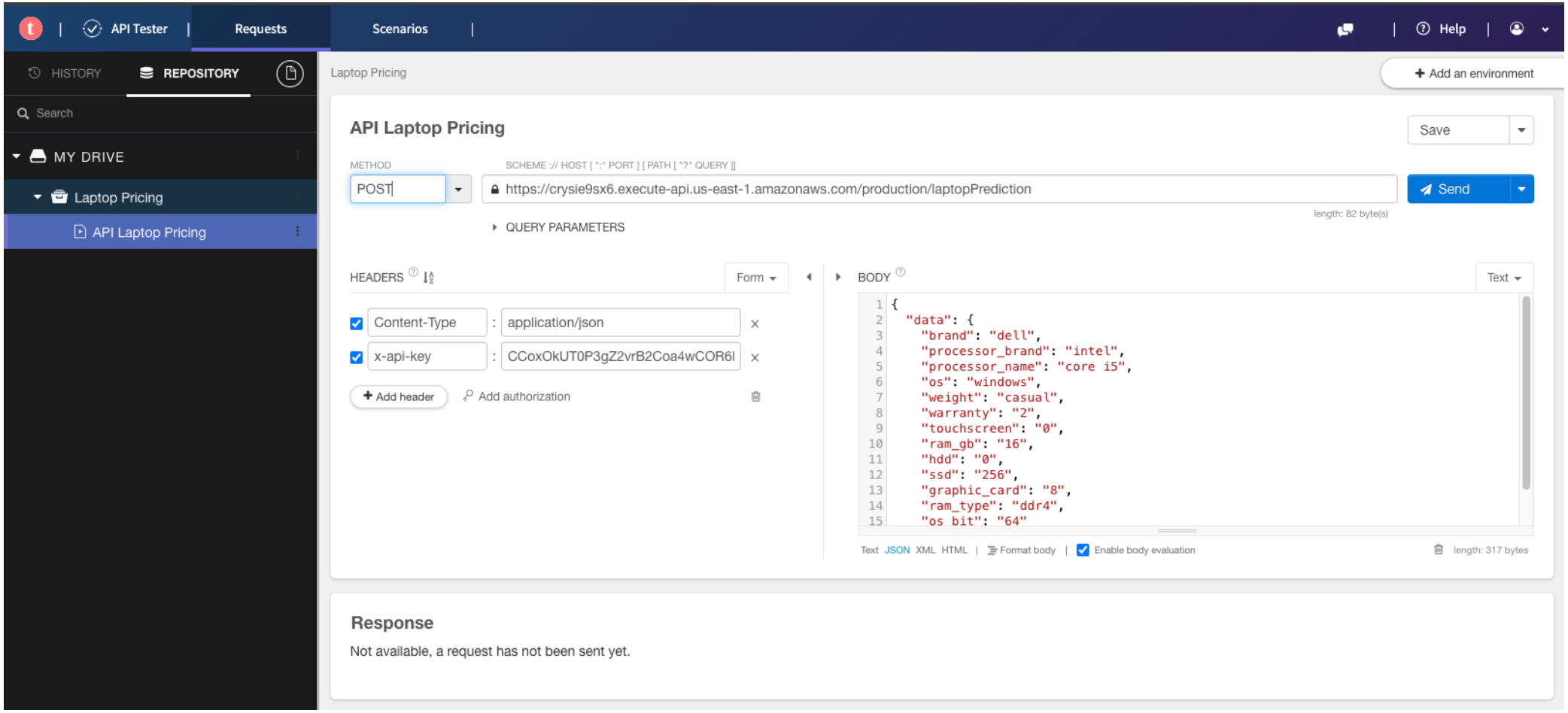


Antes de liberar a API para consumo, testamos no Talend para termos certeza de que tudo está funcionando como esperado.

https://chromewebstore.google.com/detail/talend-api-tester-free-ed/

TESTANDO A API NO **TALEND**

Configuramos o endereço do endpoint. Depois o header de chave (senha) com o parâmetro “x-api-key” e o contente type JSON. Por fim, editamos o body com o mesmo valor do evento de teste no Lambda.



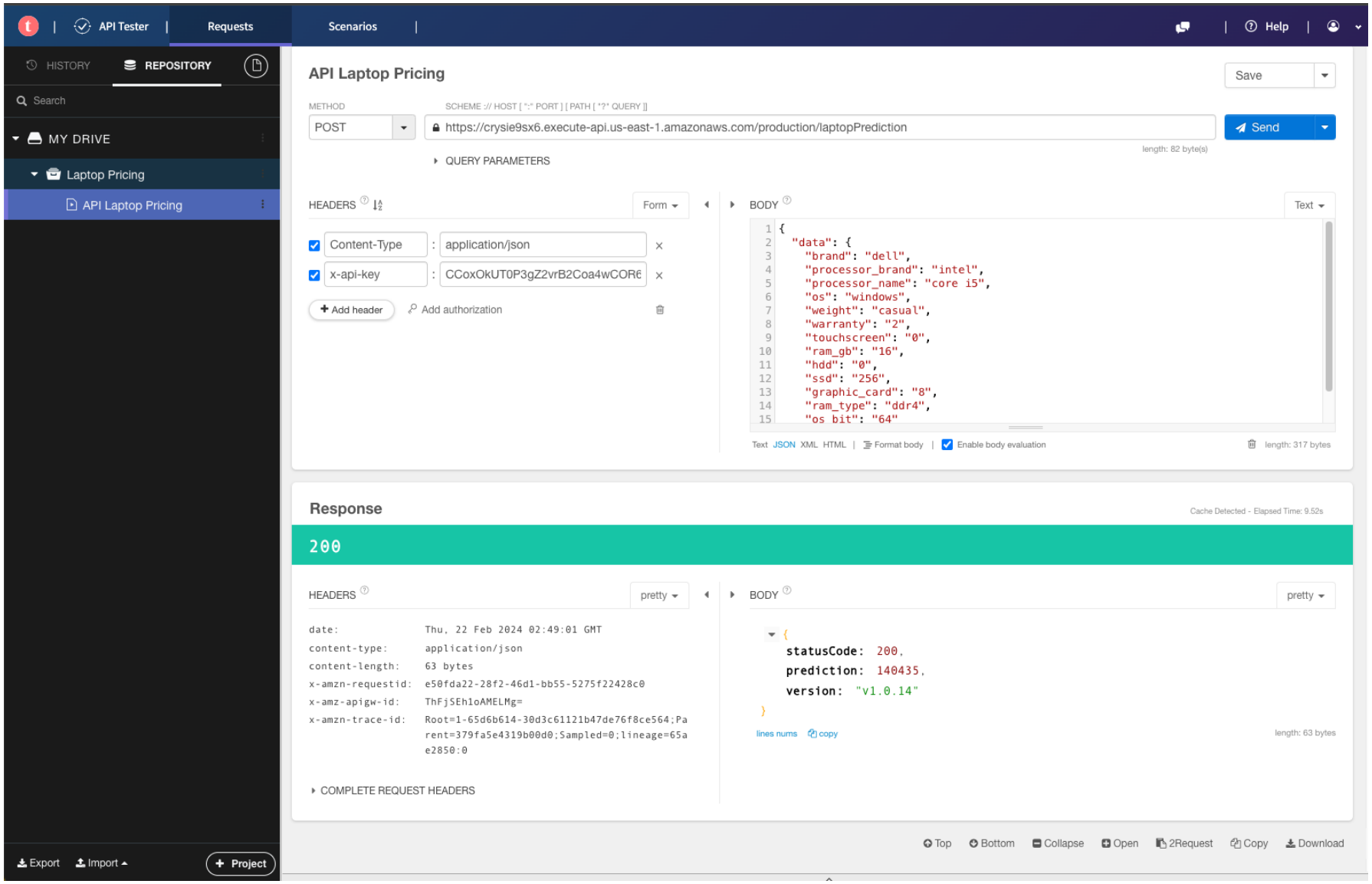
TESTANDO A API NO **TALEND**

Clicamos em “Send” e analisamos o retorno da API.

Primeiro o código de resposta, 200 sinaliza que foi processado com sucesos.

Depois o valor de retorno traz conforme o esperado: código de status, predição e a versão do modelo.

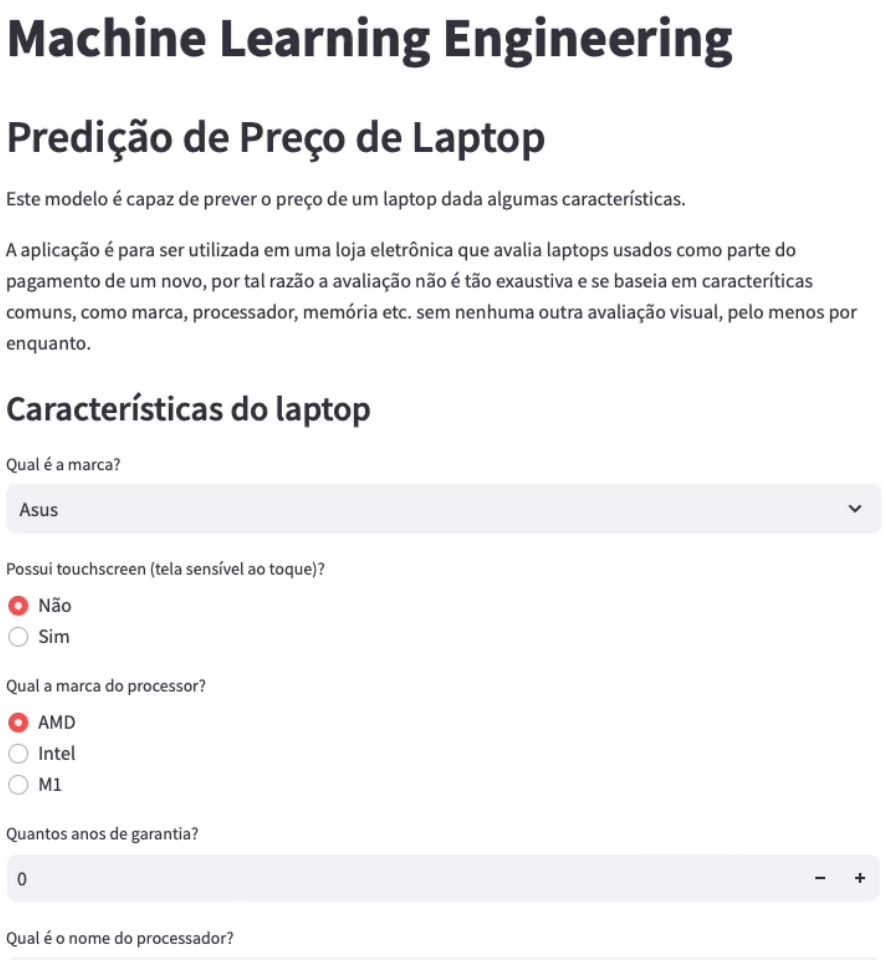
Tudo pronto para integrarmos com o frontend.



CICLO PARA CONSTRUIR UM MODELO

Streamlit

APP DE INTERAÇÃO: **FRONTEND**



A aplicação de interação pode envolver diferentes formas de frontend, como uma aplicação web ou até mesmo um aplicativo móvel em smartphones.

Vamos utilizar o Streamlit que nos servirá de frontend e o serviço será consumido remotamente por meio de uma API que desenvolvemos anteriormente.

STREAMLIT

É uma biblioteca de código aberto em Python que permite a criação rápida e fácil de aplicativos da web interativos para ciência de dados e aprendizado de máquina. Com ele é possível transformar scripts Python em aplicativos da web interativos com apenas algumas linhas de código, sem a necessidade de conhecimento em desenvolvimento web.

- **Simplicidade de Uso**: foi projetado para ser simples e intuitivo, permitindo que os desenvolvedores criem aplicativos da web interativos com apenas algumas linhas de código Python.
- **Reatividade em Tempo Real**: os aplicativos criados são reativos em tempo real, o que significa que as atualizações nos elementos da interface do usuário são refletidas instantaneamente, sem a necessidade de recarregar a página.
- **Ampla Gama de Widgets**: oferece uma ampla variedade de widgets interativos, como botões, caixas de seleção, barras deslizantes e gráficos, que podem ser facilmente integrados aos aplicativos para torná-los mais dinâmicos e envolventes.
- **Integração com Bibliotecas de Visualização**: integra perfeitamente com bibliotecas populares de visualização de dados, como Matplotlib, Plotly e Altair, permitindo que os desenvolvedores criem visualizações impressionantes com apenas algumas linhas de código.
- **Suporte para Modelos de Machine Learning**: os aplicativos podem incluir modelos de machine learning treinados e fornecer uma interface amigável para os usuários interagirem com esses modelos, inserindo dados de entrada e visualizando os resultados.

COMPOENTES: **WIDGETS VISUAIS**

Para coletarmos a imagem, vamos utilizar componentes de entrada do Streamlit. Eles ficam na documentação como “Input Widgets”, nesta [documentação](#).

Vamos utilizar o componente “camera_input” para coletar a imagem da câmera do dispositivo.

Number input

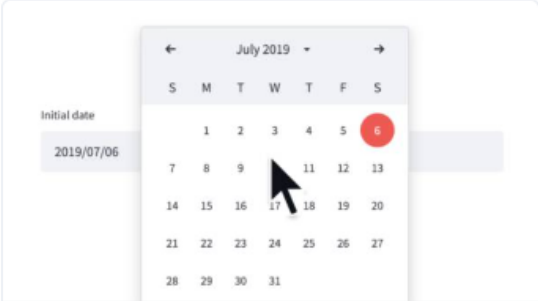
Display a numeric input widget.

```
choice = st.number_input("Pick a numbe
```

Text area

Display a multi-line text input widget.

```
text = st.text_area("Text to translate
```



Date input

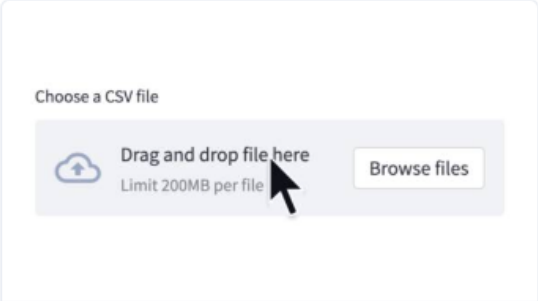
Display a date input widget.

```
date = st.date_input("Your birthday")
```

Time input

Display a time input widget.

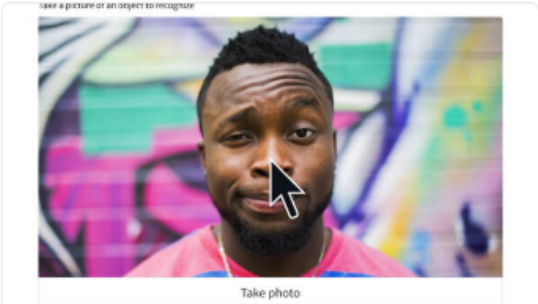
```
time = st.time_input("Meeting time")
```



File uploader

Display a file uploader widget.

```
data = st.file_uploader("Upload a CSV"
```



Camera input

Display a widget that allows users to upload images directly from a camera.

```
image = st.camera_input("Take a pictur
```

APP DE INTERAÇÃO: **FRONTEND**

Durante a chamada ao serviço externo, precisamos notificar o usuário que há um processamento pendente. Neste caso utilizamos o componente “spinner”.

Spinner

Temporarily displays a message while executing a block of code.

```
with st.spinner("Please wait..."):
  do_something_slow()
```

APP DE INTERAÇÃO: **FRONTEND**

E finalmente para a exibição dos diferentes estados, sucesso ou erro, utilizaremos os componentes de status “success” e “error”.

Ou ainda podemos utilizar uma saída em tela no formato Markdown.

Error running analysis

Error box

Display error message.

```
st.error("We encountered an error")
```

Trimmed input to fit buffer

Warning box

Display warning message.

```
st.warning("Unable to fetch image. Ski
```

Last datapoint is for partial data

Info box

Display an informational message.

```
st.info("Dataset is updated every day
```

Analysis finished in 3.14s

Success box

Display a success message.

```
st.success("Match found!")
```

ValueError: operands could not be broadcast together with shapes (10,2) (2,10)

Traceback:

```
File "/Users/test/Projects/streamlit/streamlit/docs/api-examples-source/except
    np.random.randn(10, 2) + np.random.randn(2, 10)
```

Exception output

Display an exception.

```
e = RuntimeError("This is an exception
st.exception(e)
```

DESAFIO 1

Crie uma aplicação no Streamlit que consuma a API criada anteriormente e envie os parâmetros esperados.
Utilize “Secrets” para não expor publicamente o endpoint da API.

UTILIZAÇÃO DE **SECRETS**

Qualquer dado que não precise ser exposto publicamente como chaves e endpoints devem ser mantidos em variáveis de ambiente ou em ambientes com secrets.

O Streamlit possui esta área em seu ambiente de cloud e pode ser utilizado no projeto a parte. Isto é, tais informações secretas fica no seu repositório local mas não no público.

Para armazená-lo, é necessário criar uma pasta “.streamlit” e dentro dela um arquivo “secrets.toml”.

A seguir incluímos o endpoint do serviço (Lambda Function).

```
API-ENDPOINT = "https://u2dspwvgdq6dwi6qj26lyl5y5e0hcqzq.lambda-url.us-east-1.on.aws"
```

BIBLIOTECAS E **DEPENDÊNCIAS**

Nossas dependências são poucas pois toda a complexidade ficou na API REST.
Utilizaremos somente própria biblioteca de componentes do Streamlit e outra para acesso a APIs externas (Requests).

```
streamlit==1.30.0
requests==2.31.0
```

Cravar o versionamento evita atualizações mais recentes que podem quebrar a compatibilidade do nosso projeto.

USO DE **MARKDOWN**

O Streamlit permite o uso de Markdown na forma de comentário na aplicação Python. Ela renderiza como texto na aplicação.

```
"""
# Machine Learning Engineering

## Predição de Preço de Laptop

Este modelo é capaz de prever o preço de um laptop dada algumas características.

A aplicação é para ser utilizada em uma loja eletrônica que avalia laptops usados como parte do pagamento de um
novo,
por tal razão a avaliação não é tão exaustiva e se baseia em caracteríticas comuns, como marca, processador,
memória etc.
sem nenhuma outra avaliação visual, pelo menos por enquanto.

### Características do laptop
"""
```

FILOSOFIA **PROCEDURAL**

O fluxo de aplicação é sequencial e procedural. Basta incluindo os componentes e validando logo após se são válidos para o programa continuar.

```
brand_option = st.selectbox(
"Qual é a marca?",
("Asus", "Dell", "HP", "Lenovo", "Outro"))

touchscreen = st.radio(
"Possui touchscreen (tela sensível ao toque)?",
["Não", "Sim"])

processor_brand = st.radio(
"Qual a marca do processor?",
["AMD", "Intel", "M1"])

warranty = st.number_input("Quantos anos de garantia?", step=1, placeholder="Coloque 0 se não houver garantina.")

...

weight = st.radio(
"Qual o peso estimado?",
["Casual", "Gaming", "Thinlight"],
captions = ["Peso padrão", "Pesado", "Leve"])

ram_type = st.radio(
"Qual o tipo da memória RAM?",
["DDR4", "Outro"])
```

Inicialmente vamos incluindo os componentes e obtendo do usuário suas seleções.

Para dados categóricos, utilizamos caixa de seleção.

Para dados numéricos utilizamos input de dados.
Neste componente ainda podemos regular o step, ou seja, de quanto em quanto é realizado a iteração.
Valores de memória podem subir em múltiplos 8 por exemplo.

FILOSOFIA **PROCEDURAL**

Com a entrada dos dados devemos montar o payload com as chaves conforme é esperado pela API. Note que decidimos deixar a conversão deste formato para features (one-hot encoding) a ser feito na API.

```
if ram_type == "Outro":
ram_type = "other"


if touchscreen=="Sim":
touchscreen = "1"
else:
touchscreen = "0"

payload = { "data" : {
"brand": brand_option.lower(),
"processor_brand": processor_brand.lower(),
"processor_name": brand_option.lower(),
"os": os_brand.lower(),
"weight": weight.lower(),
"warranty": warranty,
"touchscreen": touchscreen,
"ram_gb": ram_size,
"hdd": hdd_size,
"ssd": ssd_size,
"graphic_card": graphic_card_option,
"ram_type": ram_type.lower(),
"os_bit": os_bit
}
}
```

Payload em formato dicionário. Note que precisaremos converter para JSON na requisição para a API.

```
if st.button("Estimar Preço"):
with st.spinner("Processando..."):
get_prediction(payload)
```

Neste parte iremos acionar a função que irá chamar a API remota.

FILOSOFIA **PROCEDURAL**

Depois da requisição for enviada, validamos se o código é de sucesso. Esta referência fornece todos os status codes. Por padrão 200 é sucesso.

```
def get_prediction(data):

print(json.dumps(data))

endpoint = st.secrets["API-ENDPOINT"]
headers = {'x-api-key': st.secrets["API-KEY"], 'Content-Type': 'application/json'}

response = requests.post(endpoint, data=json.dumps(data), headers=headers)
if response.status_code == 200:
result = (response.json())
print(result)

"""
### Preço estimado para compra

De acordo com os dados fornecidos, seu laptop poderá ser comprado pelo seguinte valor abaixo.
"""

locale.setlocale(locale.LC_ALL, 'pt_BR')
predicted_value_formatted = locale.format_string("%d", result['prediction'], grouping=True)

st.markdown("Valor para compra: **" + str(predicted_value_formatted) + " IND (India Rupee)**.")
else:
st.markdown("Houve um problema na consulta. Revise os dados.")
```

Se o código for 200, tudo certo.
Do contrário, houve algum erro no processamento da API.

Afim de ter o padrão brasileiro de número inteiro e decimal, precisamos incluir o Locale. E ainda, por questões específicas da plataforma do Streamlit, se decidirmos publicar esta aplicação precisamos incluir um arquivo package.txt e nele incluir a linha locales-all para que o locale brasileiro seja ativado.

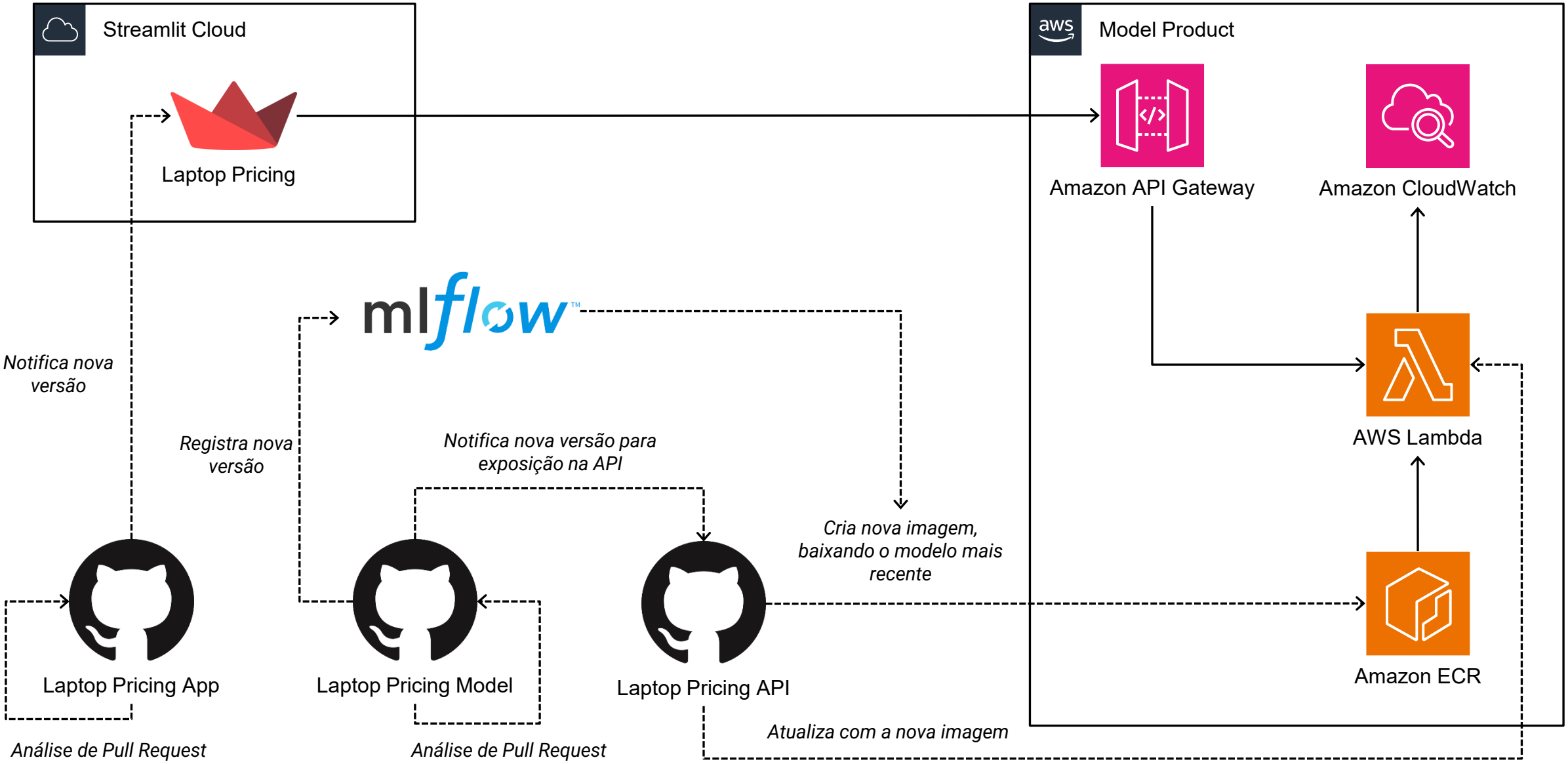
CICLO PARA CONSTRUIR UM MODELO

PARA SABER **MAIS**

https://dvc.org/doc, documentação da DVC e de todos os produtos que utilizaremos (DVC, DVCLive, Studio e GTO)
https://youtube.com/playlist?list=PL7WG7YrwYcnDBDuCkFbcyjnZQrdskFsBz&si=rI1SrP6S06Kc_OmS, Playlist de vídeos sobre MLOps da DVC
https://cursos.alura.com.br/course/streamlit-construindo-dashboard-interativo, Curso de Streamlit para construção de dashboards
https://www.youtube.com/watch?v=wxLvvMxzc1Q, Containers, Docker e Kubernetes com Giovanni Bassi

PIPELINES DE INTEGRAÇÃO E ENTREGA CONTÍNUA

CD/CI **FLOW** BLUEPRINT



GitHub Actions

GITHUB **ACTIONS**

GitHub Actions é um serviço de automação que permite automatizar fluxos de trabalho diretamente do seu repositório GitHub. Principais recursos:

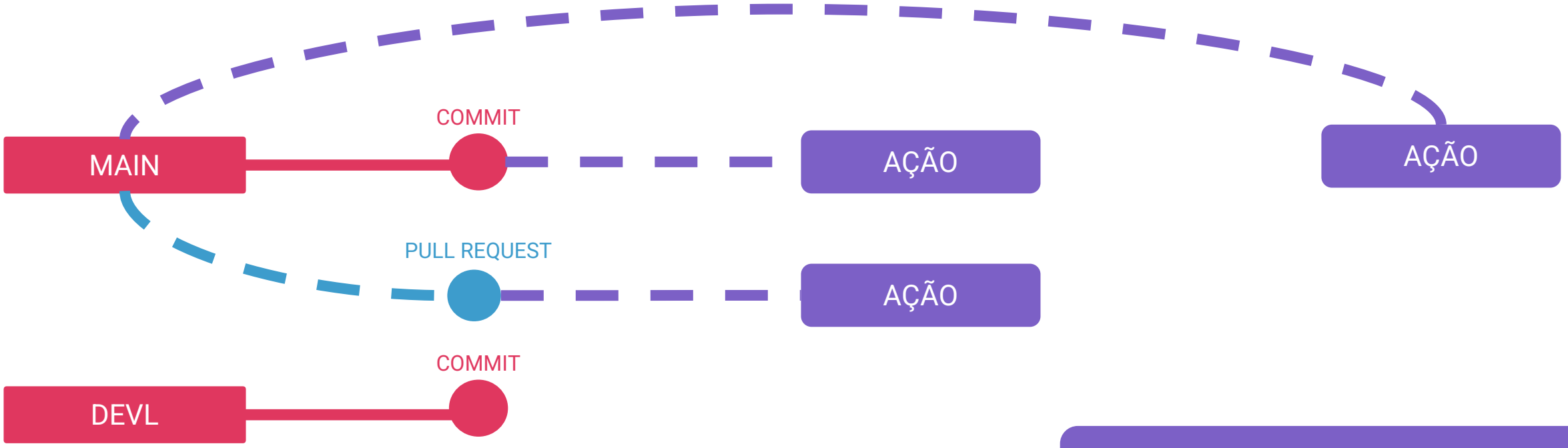
- Automatização de Fluxos de Trabalho: permite criar fluxos de trabalho automatizados para construir, testar, empacotar, liberar e implantar seu código.
- Integração Contínua (CI): é possível configurar CI para construir e testar automaticamente seu código sempre que houver um push ou uma pull request no repositório.
- Implantação Contínua (CD): é possível configurar implantações contínuas para enviar automaticamente seu código para ambientes de produção ou de teste após a aprovação dos testes de CI.
- Eventos do GitHub: GitHub Actions responde a uma variedade de eventos do GitHub, como push, pull requests, issues, entre outros, permitindo desencadear ações específicas em resposta a esses eventos.
- Reutilização de Ações: é possível compartilhar ações reutilizáveis, ou seja, pequenos trechos de código que podem ser utilizados em diferentes fluxos de trabalho, facilitando a composição de processos complexos.
- Infraestrutura Escalável: é executado em infraestrutura escalável fornecida pelo GitHub, o que significa que você não precisa se preocupar com provisionamento ou gerenciamento de servidores.

Mesmo o os runners do GitHub Actions serem gerenciados pelo GitHub é possível instalar runners dedicados por meio de agentes. Com isso podemos ter diferentes tipos de instância até mesmo com tipos de CPU e GPU adequadas para as atividades Machine Learning.

https://docs.github.com/pt/actions
https://dvc.ai/blog/cml-self-hosted-runners-on-demand-with-gpus

ESTRATÉGIAS DE **ACTIONS**

Cada necessidade requer um tipo diferente de ação.
Construir um modelo, montar uma imagem de contêiner, implantar um serviço de nuvem podem diferenciar um fluxo de outro. Até mesmo o tipo de ação, um Pull Request ou atualizações em uma determinada ramificação.



As ações visam garantir testes mínimos para avançar para etapas de merge e implantação do código.

PASSOS COMUNS DE **ACTIONS**

Cada necessidade requer um tipo diferente de ação.
Construir um modelo, montar uma imagem de contêiner, implantar um serviço de nuvem podem diferenciar um fluxo de outro. Até mesmo o tipo de ação, um Pull Request ou atualizações em uma determinada ramificação.

Análise de Pull Request

VERIFICAR E INSTALAR DEPENDÊNCIAS

EXECUTAR PIPELINE DE VALIDAÇÃO → TESTES DE UNIDADE | VERIFICAÇÃO DO MODELO | EXECUÇÃO | GOLDEN DATA

COLETAR MÉTRICAS

VALIDAÇÃO ANTES DE COMBINAR O CÓDIGO COM PRODUÇÃO

A análise de Pull Request é em outra branch que não a de produção (master).
As validações mínimas do pipeline de fluxo de dados e a análise de métricas garantem que não haja alterações que possam quebrar o modelo.
Na hipótese dos testes serem alterados, uma análise mais criteriosa precisa ser feita durante a revisão.
Após a aprovação do Pull Request o modelo segue para uma nova rodada de treinamento e registro do modelo.

ACTION PARA **PULL REQUEST**

FIAP MBA+

```yaml
name: MLflow Model Report

on:
pull_request:
branches: [main]

permissions:
contents: read
pull-requests: write

jobs:
report:
runs-on: ubuntu-latest

steps:
- name: Checkout code
uses: actions/checkout@v4

- name: Setup Python
uses: actions/setup-python@v5
with:
python-version: '3.10'

- name: Install dependencies
run: pip install -r requirements.txt

- name: Unit testing
run: pytest
```

Execução somente em Pull Request

Permissões para escrita de sumário do Action, modificações de Pull Request

Sistema operacional do container.

Obtendo arquivos do repositório

Obtendo Python 3.10

Instalando dependências de produção

Realizando testes de unidade

ACTION PARA **PULL REQUEST**

```yaml
- name: Generate MLflow Report
run: python reports/report.py
```

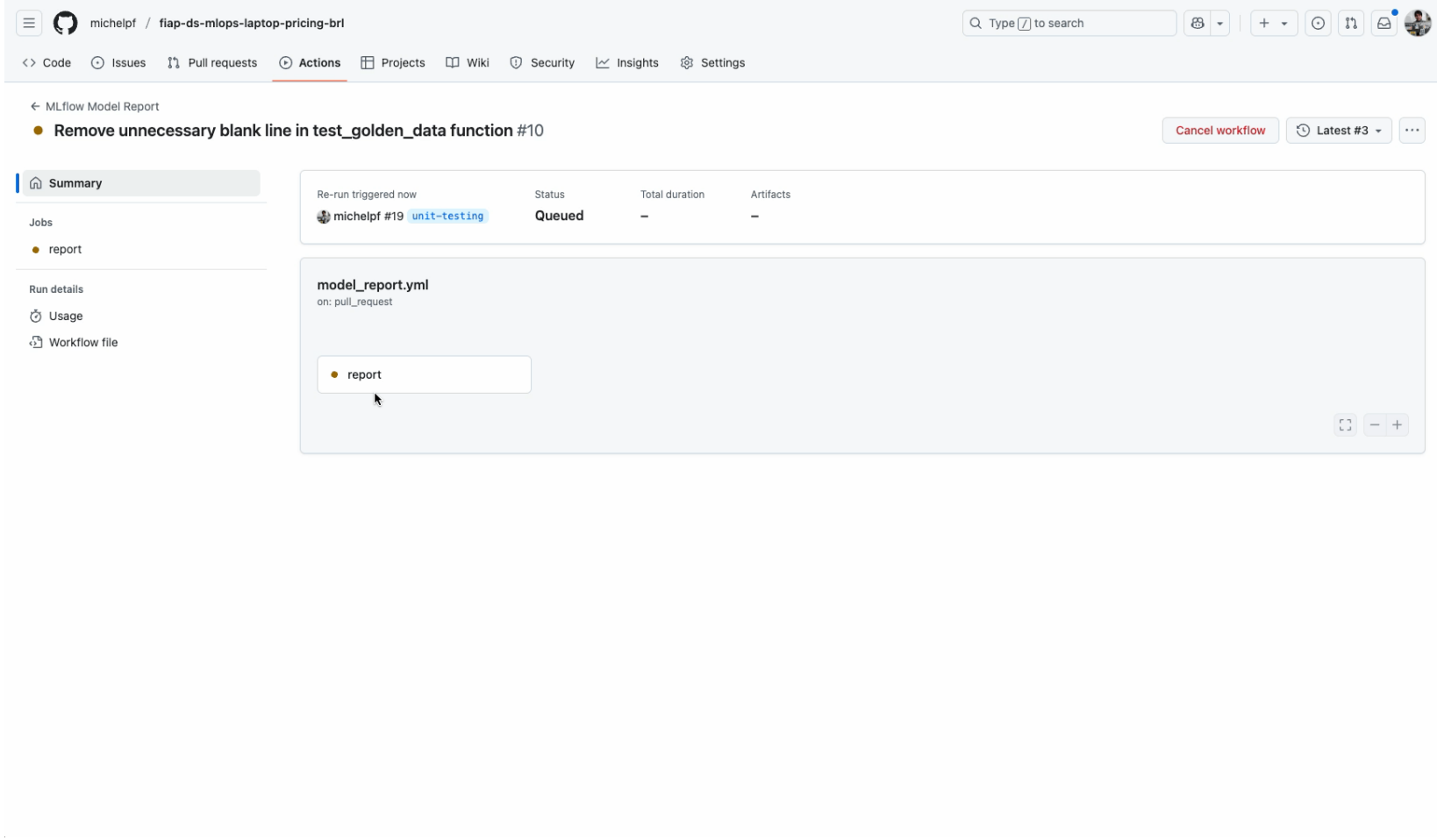
Gerando o relatório de métricas

```yaml
- name: Comment on Pull Request
uses: marocchino/sticky-pull-request-comment@v2
with:
header: mlflow-report
path: mlflow_report.md
```

Exibindo o relatório no Pull Request

ACTION PARA **PULL REQUEST**

Cada passo (step) é executado sequencialmente. Quando algum passo falha (código de status diferente de 0), a pipeline é interrompida até que uma nova execução sem erro seja realizada. Com isso nenhuma modificação é lançada em produção.

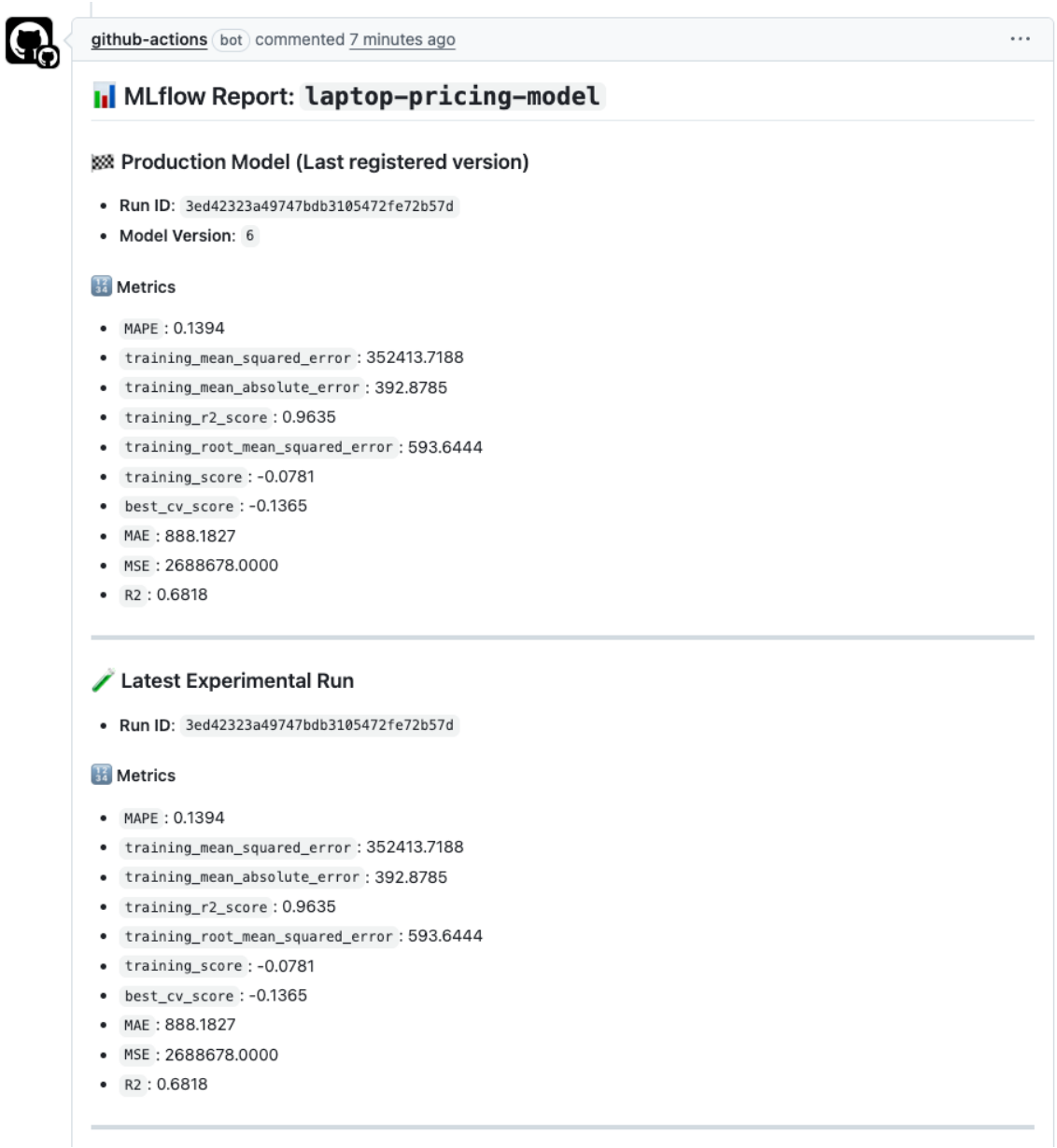


ACTION PARA **PULL REQUEST**

O relatório de métricas do modelo serve para revisar se o novo modelo registrado cumpre com os objetivos estabelecidos pela governança.

Integrando na tela de Pull Request podemos facilmente verificar as variações e decidir se a nova abordagem deve ser incorporada no código principal.

Caso seja incorporado, será disparado uma notificação para atualizar o modelo servido pela API.



github-actions bot commented 7 minutes ago

📊 MLflow Report: `laptop-pricing-model`

🏁 Production Model (Last registered version)

- Run ID: `3ed42323a49747bdb3105472fe72b57d`
- Model Version: `6`

🔢 Metrics

- `MAPE` : 0.1394
- `training_mean_squared_error` : 352413.7188
- `training_mean_absolute_error` : 392.8785
- `training_r2_score` : 0.9635
- `training_root_mean_squared_error` : 593.6444
- `training_score` : -0.0781
- `best_cv_score` : -0.1365
- `MAE` : 888.1827
- `MSE` : 2688678.0000
- `R2` : 0.6818

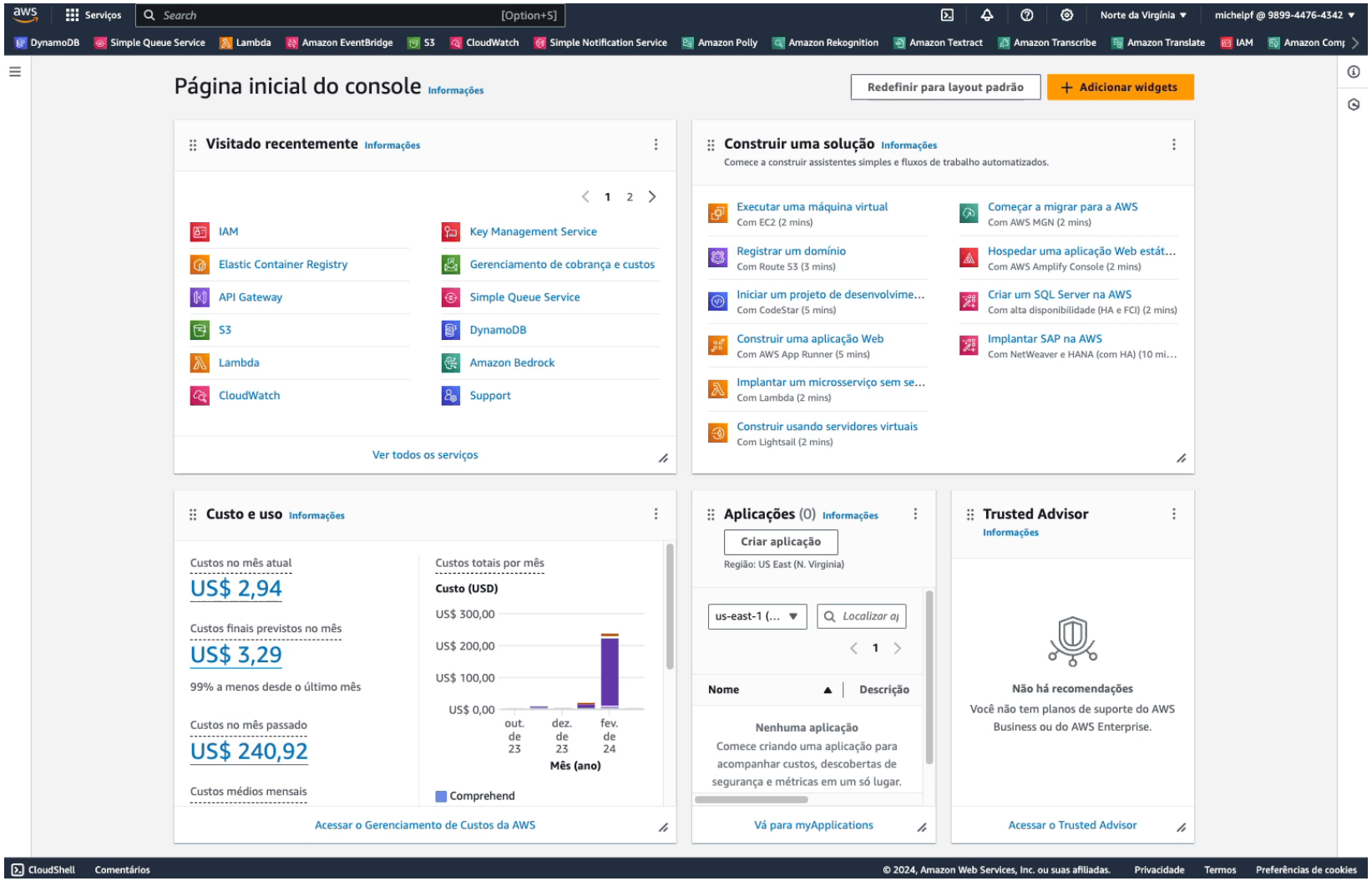
🧪 Latest Experimental Run

- Run ID: `3ed42323a49747bdb3105472fe72b57d`

🔢 Metrics

- `MAPE` : 0.1394
- `training_mean_squared_error` : 352413.7188
- `training_mean_absolute_error` : 392.8785
- `training_r2_score` : 0.9635
- `training_root_mean_squared_error` : 593.6444
- `training_score` : -0.0781
- `best_cv_score` : -0.1365
- `MAE` : 888.1827
- `MSE` : 2688678.0000
- `R2` : 0.6818

ADICIONADO PROVEDOR **OIDC PARA GITHUB**



Para evitarmos salvar credenciais de acesso em secrets, a AWS permite validar credenciais de acesso utilizado OpenID Connect. Com este método, as ações das pipelines do GitHub acessam os recursos da AWS conforme liberação de política.
A configuração deste tipo de identidade é feita por essa URL de provedor: https://token.actions.githubusercontent.com.

https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services

ADICIONADO PROVEDOR **OIDC PARA GITHUB**

Na Role de acesso pelo Github Actions, é necessário incluir a seguinte relação de confiança.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::989944764342:oidc-provider/token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
        },
        "StringLike": {
          "token.actions.githubusercontent.com:sub": "repo:michelpf/*"
        }
      }
    }
  ]
}
```

https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services

ACTIONS PARA NOTIFICAÇÃO DE IMPLANTAÇÃO DE API

Após um Pull Request for aceito, as novas configurações estão prontas para mesclarem com o código em produção. Após isso a esteira de CI/CD vai notificar a API para implantar uma nova versão.

Como os models ficam registrados no MLFlow não é necessário nenhum passo adicional de build.

ACTIONS PARA TREINAMENTO DO MODELO

```yaml
name: Notify API Deployment

on:
push:
branches:
- main
paths:
- 'notebooks/**'

jobs:
notify_api_repository:
name: "Notification to API Repository"
runs-on: ubuntu-latest

steps:
- name: Checkout code
uses: actions/checkout@v4

- name: Setup Python
uses: actions/setup-python@v5
with:
python-version: '3.10'

- name: Install dependencies
run: pip install -r requirements.txt
```

Execução somente quando houver atualizações na ramificação main e em pastas específicas

Sistema operacional do container.

Downloado do repositório

Instalação do Python

Instalação de dependências

ACTIONS PARA TREINAMENTO DO MODELO

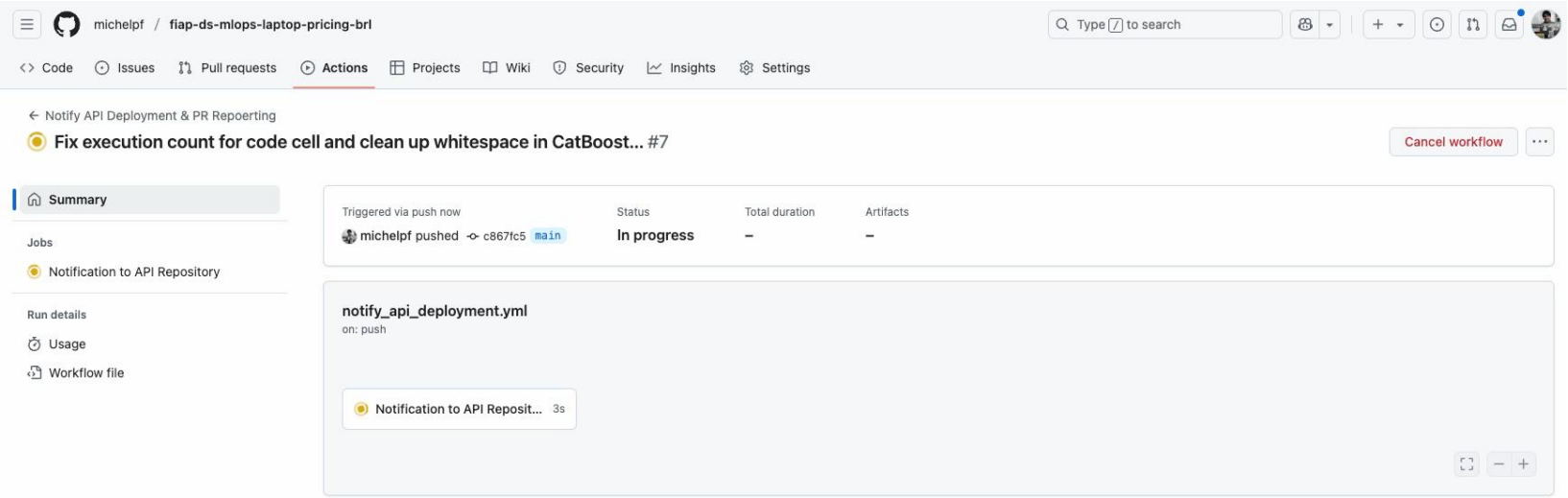
```
- name: Unit testing
run: pytest
```

Execução dos testes de unidade

```
- name: Deploy Stage
uses: actions/github-script@v6
with:
github-token: ${{ secrets.ACCESS_TOKEN }}
script: |
await github.rest.actions.createWorkflowDispatch({
owner: "michelpf",
repo: "fiap-ds-mlops-api-laptop-pricing-brl",
workflow_id: "deploy.yml",
ref: "main"
})
```

Notificar outro repositório para implantação.

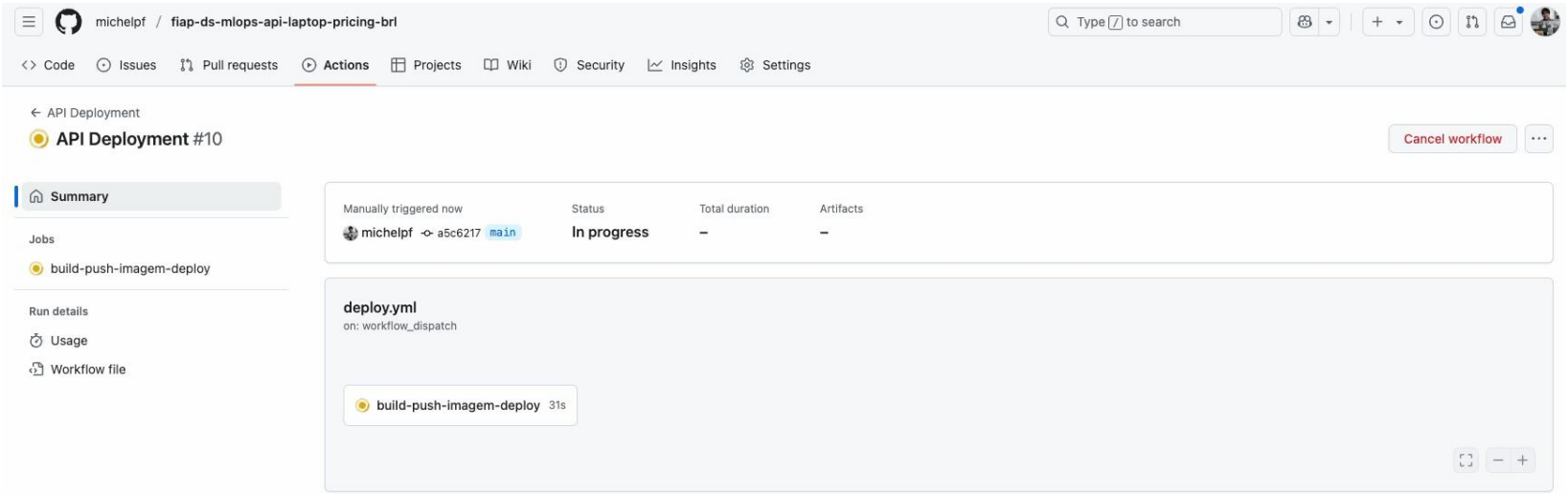
ACTIONS PARA TESTES E NOTIFICAÇÃO DE IMPLANTAÇÃO



Após os testes de unidade, a pipeline avisa a outro repositório para implantar uma nova versão da API com o modelo atualizado.

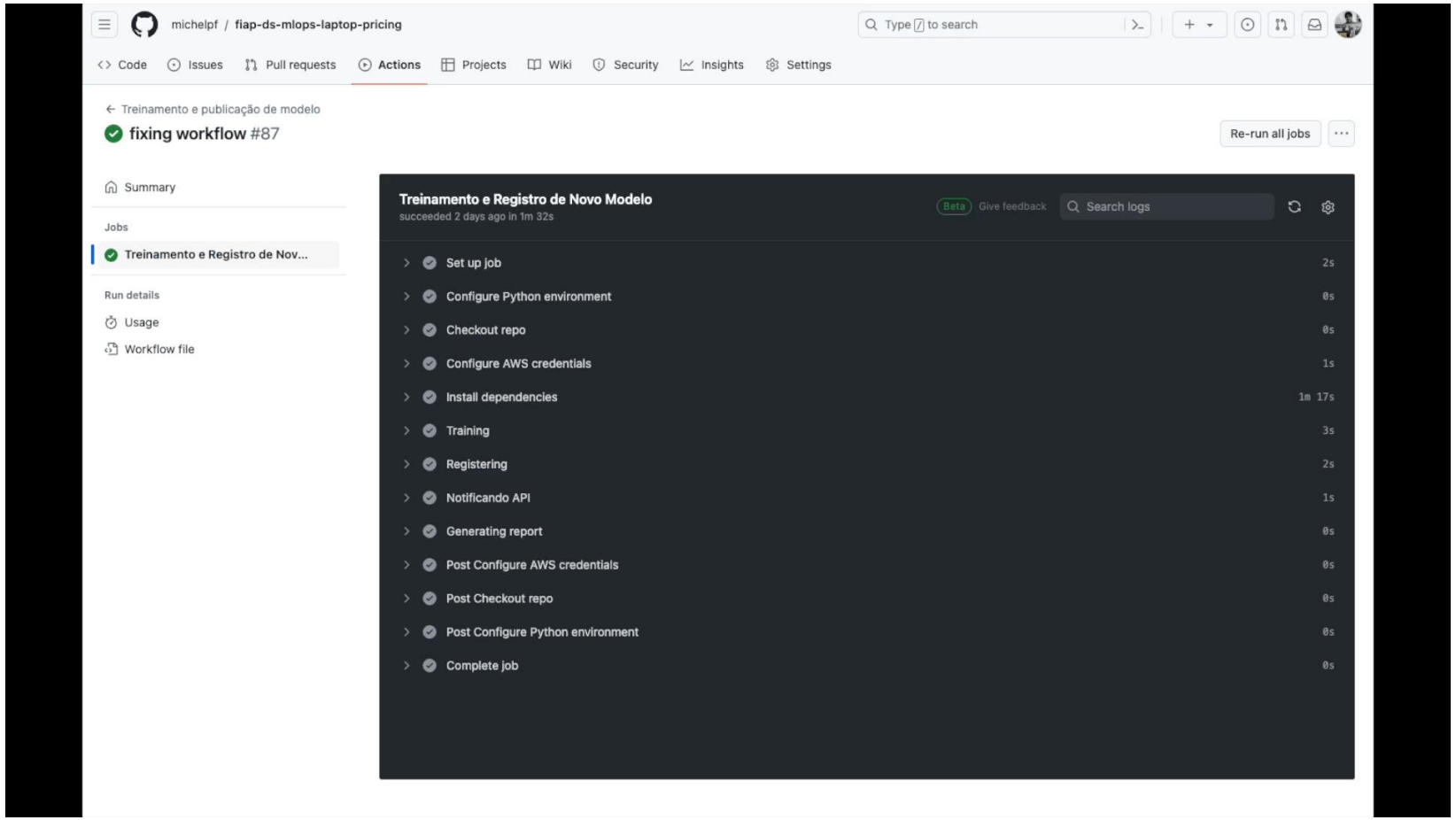
ACTIONS PARA IMPLANTAÇÃO DA API

Note o “workflow dispatch”, isso indica que a pipeline foi disparada por outra pipeline.



TOKEN DE ACESSO PARTICULAR

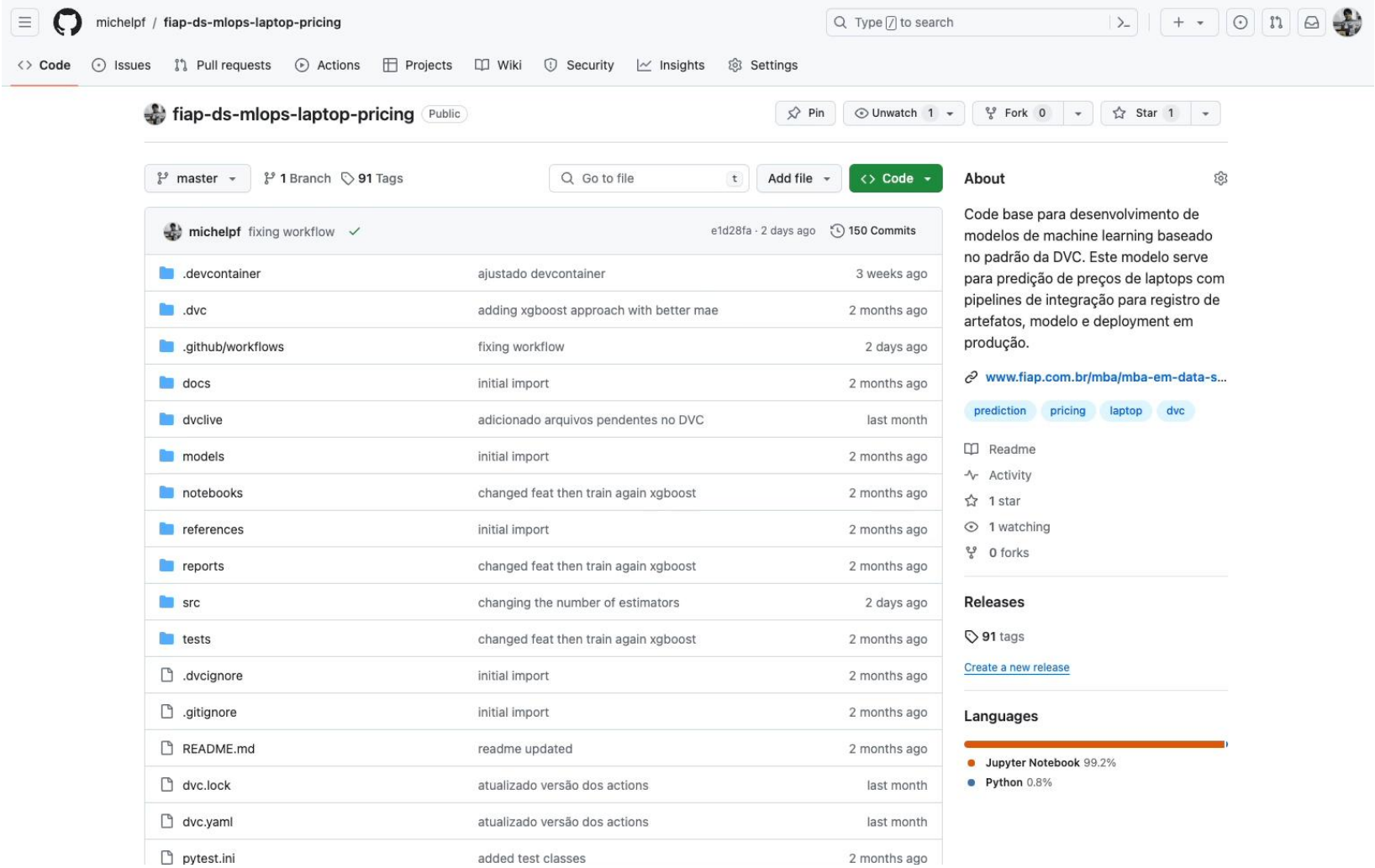
Para ações específicas como de iniciar uma action em outro repositório é necessário permissões extras, ou seja, além da do próprio repositório da qual já é oferecida um token padrão (GITHUB_TOKEN) sem a necessidade da criação de um novo. Neste caso precisamos criar um token particular de acesso.



SECRET PARA USO EM UMA ACTION

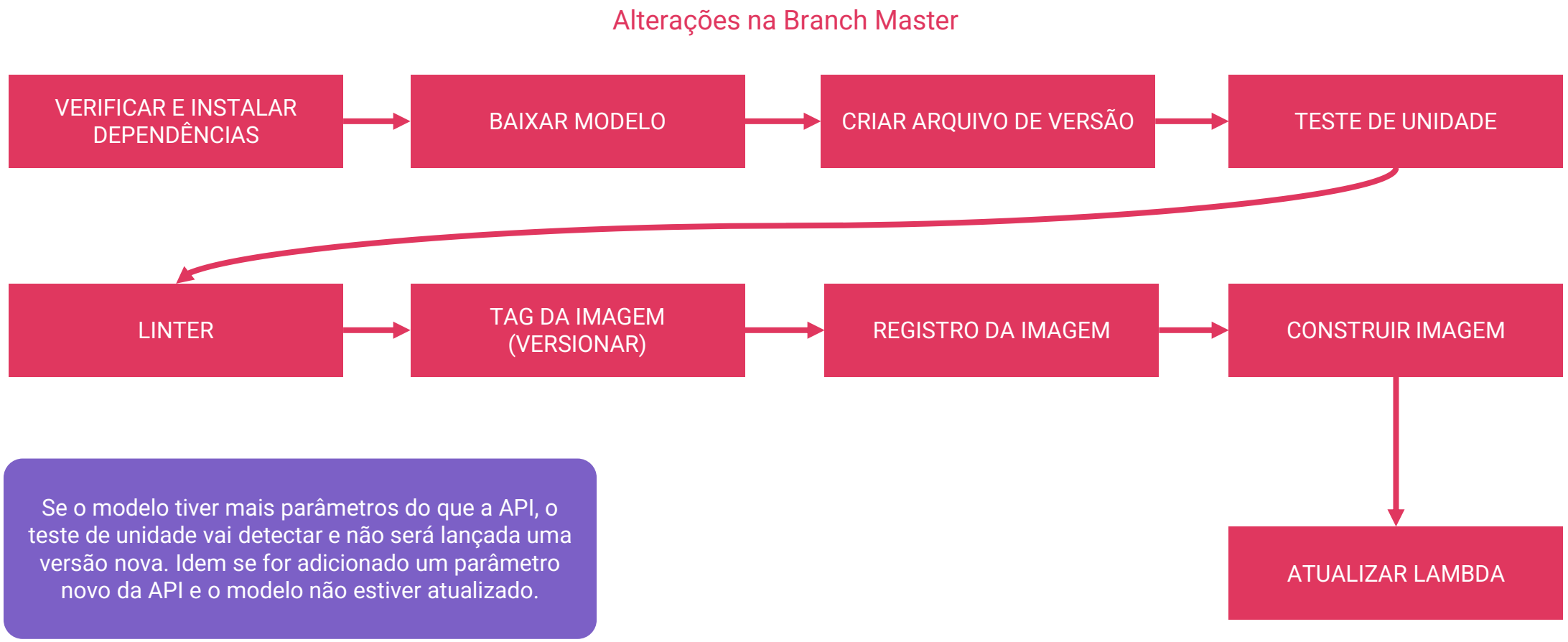
Depois de criar um acesso pessoal de token, precisamos carrega-lo como secret do repositório a ser utilizado em uma action.

O nome da secret precisa ser o mesmo a ser referenciado no workflow, ou seja, o secret “ACCESS_TOKEN” é referenciado no workflow como “secrets.ACCESS_TOKEN”.



ACTIONS PARA IMPLANTAÇÃO DA API

Após todos os testes do modelo e treinamento estiverem concluídos, passamos a notificar o repositório da API para uma nova implantação.
O repositório da API também pode disparar novas implantações que não dependem de uma nova versão do modelo, para por exemplo, melhorar a performance do modelo e segurança.



ACTIONS PARA IMPLANTAÇÃO DA API

FIAP MBA+

Podemos criar uma action dedicada para Pull Request, indo até o build da imagem do container.

```yaml
name: Build image then push to registry for deployment

on:
  workflow_dispatch:
  push:
    branches: [ "master" ]

jobs:
  build-push-imagem-deploy:
    runs-on: ubuntu-latest

    permissions:
      id-token: write

    steps:
      - name: Checkout repo
        uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::989944764342:role/GithubActionsRole
          aws-region: us-east-1

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2

      ...
```

Execução somente quando houver atualizações na ramificação master

Permissão de utilizar o token para autenticação com serviços externos via OIDC com a AWS.

Ação para baixar o conteúdo do repositório (clonagem, com todos os dados).

Ação para configurar autenticação na AWS sem precisar informar chaves de segurança, utilizando OIDC (Open ID Connect) e Assume Role.

Ação para configurar credenciais de acesso no ECR.

ACTIONS PARA IMPLANTAÇÃO DA API

FIAP MBA+

```
...

  - name: Configure Python environment
    uses: actions/setup-python@v5
    with:
      python-version: "3.10"

  - name: Install dependencies
    run: python -m pip install -r requirements.txt

  - name: Model download
    run: python –m model_dowloader

  - name: Unit testing
    run: pytest

  - name: Linting
    run: pylint src

  - name: Build image
    run: docker build --platform linux/amd64 -t laptop-pricing-api .
```

Ação para configurar o ambiente Python para o fluxo da pipeline.

Instalação dependências do projeto.

Baixar o último modelo disponível em produção.

Realizar testes de unidade.

Realizar análise estática (Lint)

Cria uma imagem Docker utilizando o Dockerfile.

ACTIONS PARA IMPLANTAÇÃO DA API

```
...

  - name: Tag to ECR
    run: docker tag laptop-pricing-api:latest 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-
api:latest
```

Inclui tag na imagem criada para apontar no registro de containers.

```
  - name: Register in ECR
    run: docker push 989944764342.dkr.ecr.us-east-1.amazonaws.com/laptop-pricing-api:latest
```

Envia o modelo para o ECR com base na tag.

```
  - name: Update Lambda
    run: aws lambda update-function-code --function-name laptop-pricing-api --image-uri 989944764342.dkr.ecr.us-
east-1.amazonaws.com/laptop-pricing-api:latest
```

Atualiza o Lambda para a versão mais atual da imagem do contêiner.

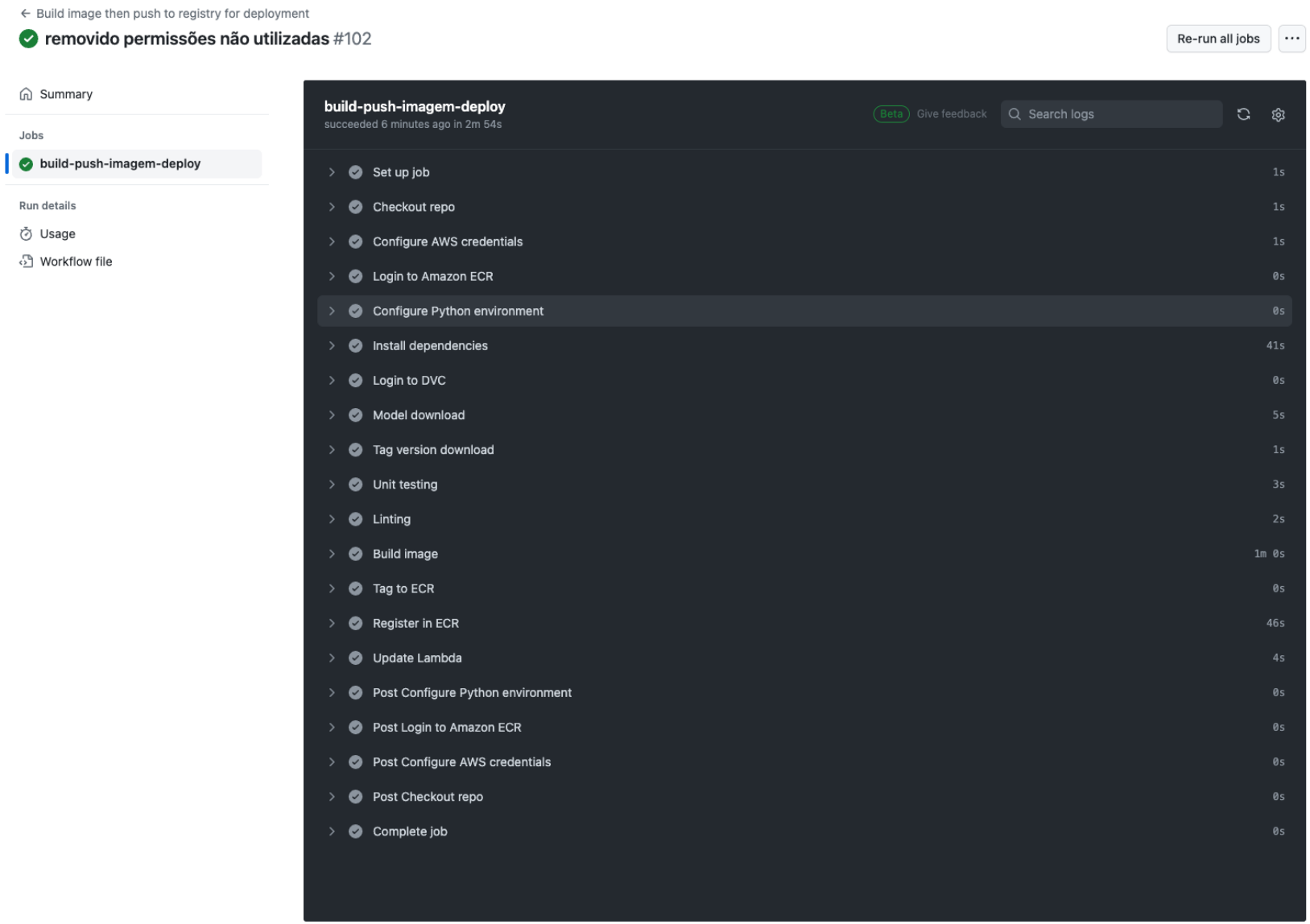
ACTIONS PARA IMPLANTAÇÃO DA API

Passos utilizados neste workflow de implantação.

Cada um deles tem o log de das ações realizadas e o resultado das verificações.

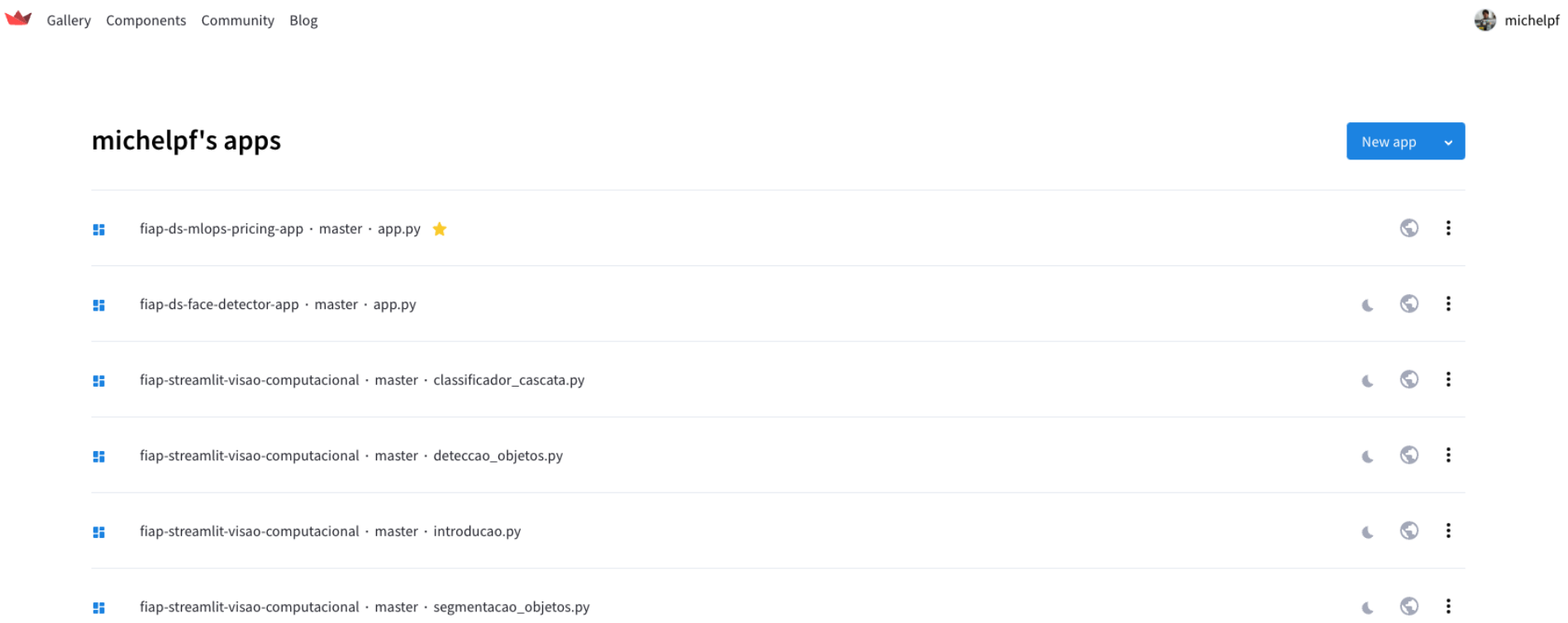
Quando todas elas são executadas com sucesso, a pipeline por sua vez é considerada executada com sucesso como um todo.

Qualquer erro durante a implantação interrompe o processo e não é lançada uma nova versão em produção.



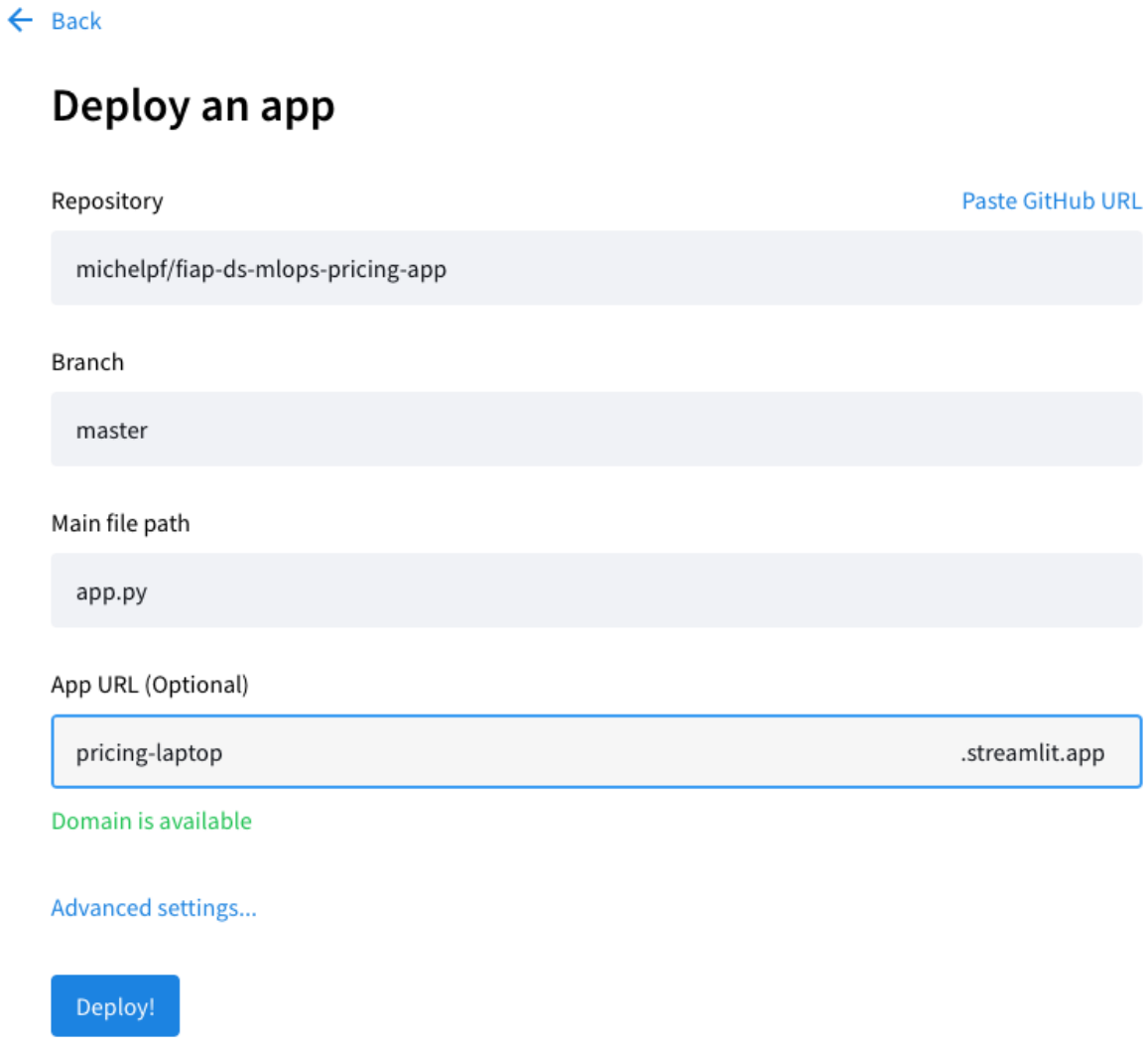
FRONTEND NO **STREAMLIT**

A plataforma Streamlit (https://streamlit.io/) permite a implantação de aplicativos em sua cloud de forma gratuita. Registre com sua conta do Google ou GitHub e tenha acesso ao painel de apps.



Clique em “New app” para configurar um novo aplicativo.

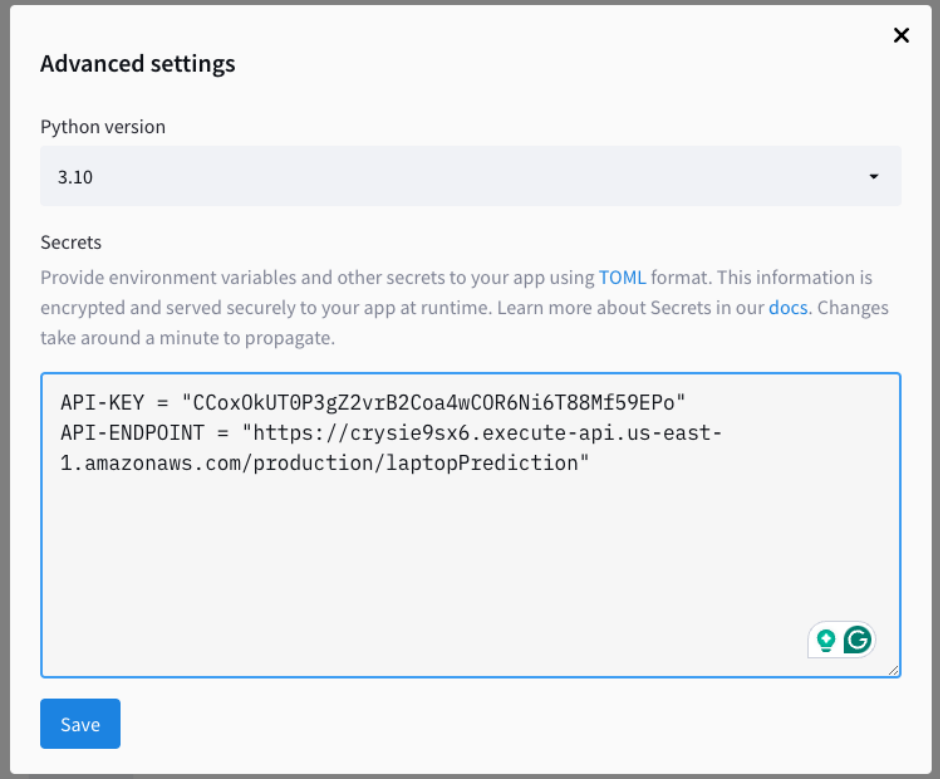
IMPLANTAÇÃO DO FRONTEND



Para configurar uma nova aplicação informe o nome do repositório, a ramificação, o nome do arquivo de script e por fim, defina um domínio.

UTILIZANDO SECRETS

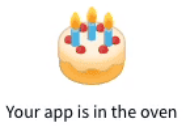
Não podemos deixar em um repositório público as informações de endpoint de API e sua chave de acesso. Por isso definimos como Secrets.
Configuramos também a versão do Python a ser utilizada.
Para nossa aplicação que utiliza o separador de milhar e casas decimais baseado na localização, precisamos definir o arquivo packages.txt para inserir a extensão a nível de sistema operacional “locales-all”.



IMPLANTAÇÃO DO FRONTEND

Após as alterações na ramificação master, elas são replicadas para o Streamlit cloud.

Podemos acompanhar a instalação pelo próprio painel da aplicação, seja quando houver uma atualização ou seja quando a aplicação for restaurada do estado de hibernação por inatividade.




```
Collecting tzdata>=2022.7
  Downloading tzdata-2024.1-py2.py3-none-any.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl
Collecting pygments<3.0.0,>=2.13.0
  Downloading pygments-2.17.2-py3-none-any.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
Collecting markdown-it-py>=2.2.0
  Downloading markdown_it_py-3.0.0-py3-none-a
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
Collecting smmap<6,>=3.0.1
  Downloading smmap-5.0.1-py3-none-any.whl (2
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.5-cp39-cp39-many
Collecting attrs>=22.2.0
  Downloading attrs-23.2.0-py3-none-any.whl (
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
Collecting jsonschema-specifications>=2023.03
  Downloading jsonschema_specifications-2023.
Collecting rpds-py>=0.7.1
  Downloading rpds_py-0.18.0-cp39-cp39-manyli
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
Collecting referencing>=0.28.4
  Downloading referencing-0.33.0-py3-none-any
Collecting mdurl~=0.1
  Downloading mdurl-0.1.2-py3-none-any.whl (1
Installing collected packages: pytz, zipp, wa
Successfully installed MarkupSafe-2.1.5 altai
WARNING: You are using pip version 22.0.3; ho
You should consider upgrading via the '/home/
Checking if Streamlit is installed
Found Streamlit version 1.30.0 in the environ
[02:46:20] 🐍 Python dependencies were instal
Check if streamlit is installed
Streamlit is already installed
[02:46:21] 📦 Processed dependencies!
[02:47:34] 🔌 Disconnecting...
master  michelpf/fiap-ds-mlops-pricing-app/...
```

OBSERVABILIDADE E MONITORAMENTO

OBSERVABILIDADE E MONITORAMENTO

FIAP MBA+

Amazon CloudWatch

Em ambientes de desenvolvimento de software, especialmente no contexto de APIs e modelos de Machine Learning, é essencial compreender a diferença entre observabilidade e monitoramento. Enquanto o monitoramento se concentra em acompanhar métricas externas e garantir o desempenho do sistema, a observabilidade visa entender o comportamento interno do sistema através da instrumentação adequada.

Monitoramento

- Acompanha continuamente métricas e KPIs.
- Coleta dados em tempo real ou em intervalos regulares.
- Analisa dados para detectar padrões, tendências ou anomalias.
- Garante o funcionamento esperado do sistema.

reativo | o quê? | quando?

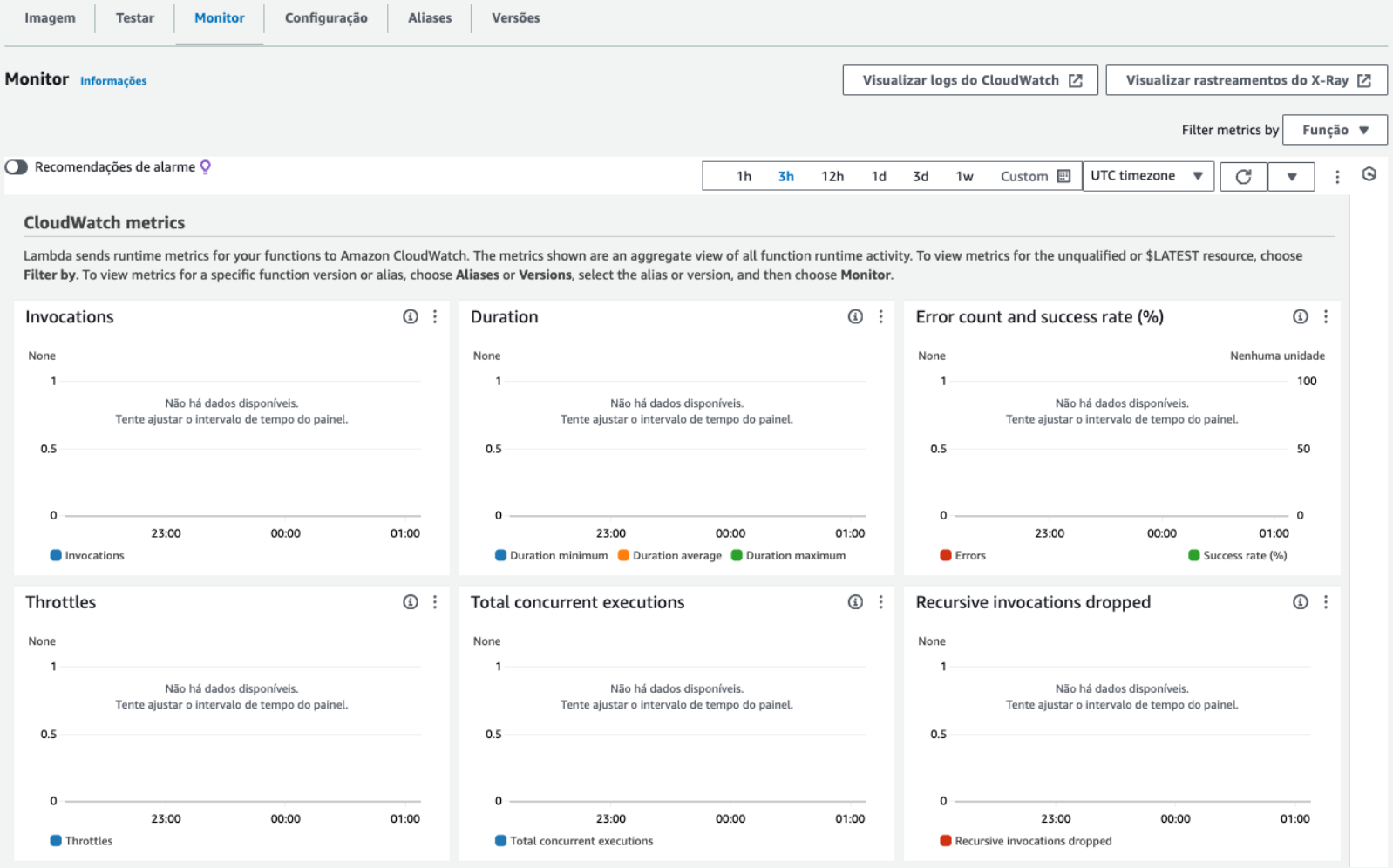
Observabilidade

- Capacidade de entender internamente o funcionamento do sistema.
- Baseia-se na instrumentação adequada para geração de dados internos.
- Permite identificar problemas e depurar falhas com facilidade.
- Foco no entendimento do comportamento interno do sistema.

proativo | por quê? | como?

PAINÉIS DE MONITORAMENTO

Todos os serviços da AWS possuem painéis dedicados para cada um deles. Tais painéis dão uma visão rápida e completa das principais métricas do serviço. O painel abaixo é da função Lambda da função de predição.
Para detalhamento de monitoração, precisamos acessar o “Visualizar logs do CloudWatch”.



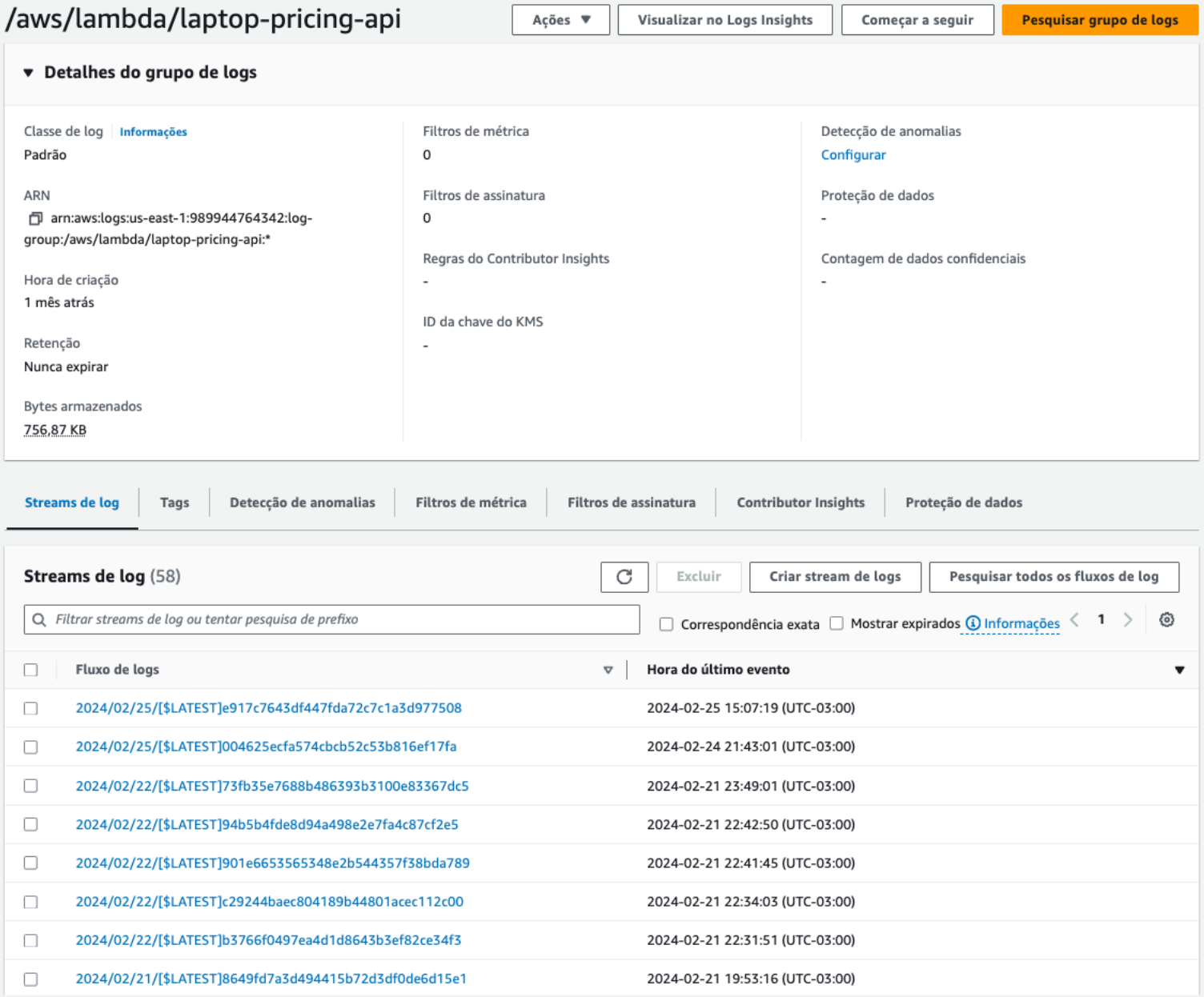
LOGS DE MONITORAMENTO

O painel do log do CloudWatch apresenta ferramentas para pesquisa e visualização dos logs.

Os streams de log é onde ficam armazenados todos os eventos.

Utilizamos os logs para investigações de causa raiz ou de condições específicas do comportamento de um determinado serviço.

Por tal razão deve-se ter muita cautela ao escolher quais mensagens devem ser logadas, para balancear investigações com excesso de dados e impactar o custo caso o armazenamento seja muito grande (aplicações em grande escala).



STREAM DE LOG

No exemplo abaixo podemos verificar como é composto um stream de logs.
São incluídos mensagens do sistema, como o “START”, “END” e “REPORT” da função Lambda e também as mensagens que a função exibe de acordo com a definição da API. Payload, Context e Prediction são exemplos de logs do serviço desenvolvido.

CloudWatch > Grupos de logs > /aws/lambda/laptop-pricing-api > 2024/02/25/[$LATEST]e917c7643df447fda72c7c1a3d977508

Eventos de log

Ações | Começar a seguir | Criar filtro de métrica

Você pode usar a barra de filtros abaixo para procurar e combinar termos, frases ou valores nos seus eventos de log. Saiba mais sobre padrões de filtro

Filtrar eventos | Limpar | 1m | 30m | 1h | 12h | Personalizado | Fuso horário local | Exibir

Timestamp	Mensagem
	Não há eventos mais antigos no momento. *Tentar novamente*
2024-02-25T15:07:13.344-03:00	/var/lang/lib/python3.10/site-packages/joblib/_multiprocessing_helpers.py:46: UserWarning: [Errno 38] Function not implemented. jo…
2024-02-25T15:07:13.344-03:00	warnings.warn('%s. joblib will operate in serial mode' % (e,))
2024-02-25T15:07:16.873-03:00	START RequestId: bf43d96a-8c5f-42d0-8d7b-4592c703e976 Version: $LATEST
2024-02-25T15:07:16.874-03:00	{'data': {'brand': 'dell', 'processor_brand': 'intel', 'processor_name': 'core i5', 'os': 'windows', 'weight': 'casual', 'warranty…
2024-02-25T15:07:16.874-03:00	LambdaContext([aws_request_id=bf43d96a-8c5f-42d0-8d7b-4592c703e976,log_group_name=/aws/lambda/laptop-pricing-api,log_stream_name=2…
2024-02-25T15:07:16.874-03:00	{'brand': 'dell', 'processor_brand': 'intel', 'processor_name': 'core i5', 'os': 'windows', 'weight': 'casual', 'warranty': '2', '…
2024-02-25T15:07:16.938-03:00	Prediction: 127923
2024-02-25T15:07:19.801-03:00	END RequestId: bf43d96a-8c5f-42d0-8d7b-4592c703e976
2024-02-25T15:07:19.801-03:00	REPORT RequestId: bf43d96a-8c5f-42d0-8d7b-4592c703e976 Duration: 2926.52 ms Billed Duration: 7021 ms Memory Size: 128 MB Max Memor…
	Não há eventos mais recentes no momento. *Nova tentativa automática pausada.* *Retomar*

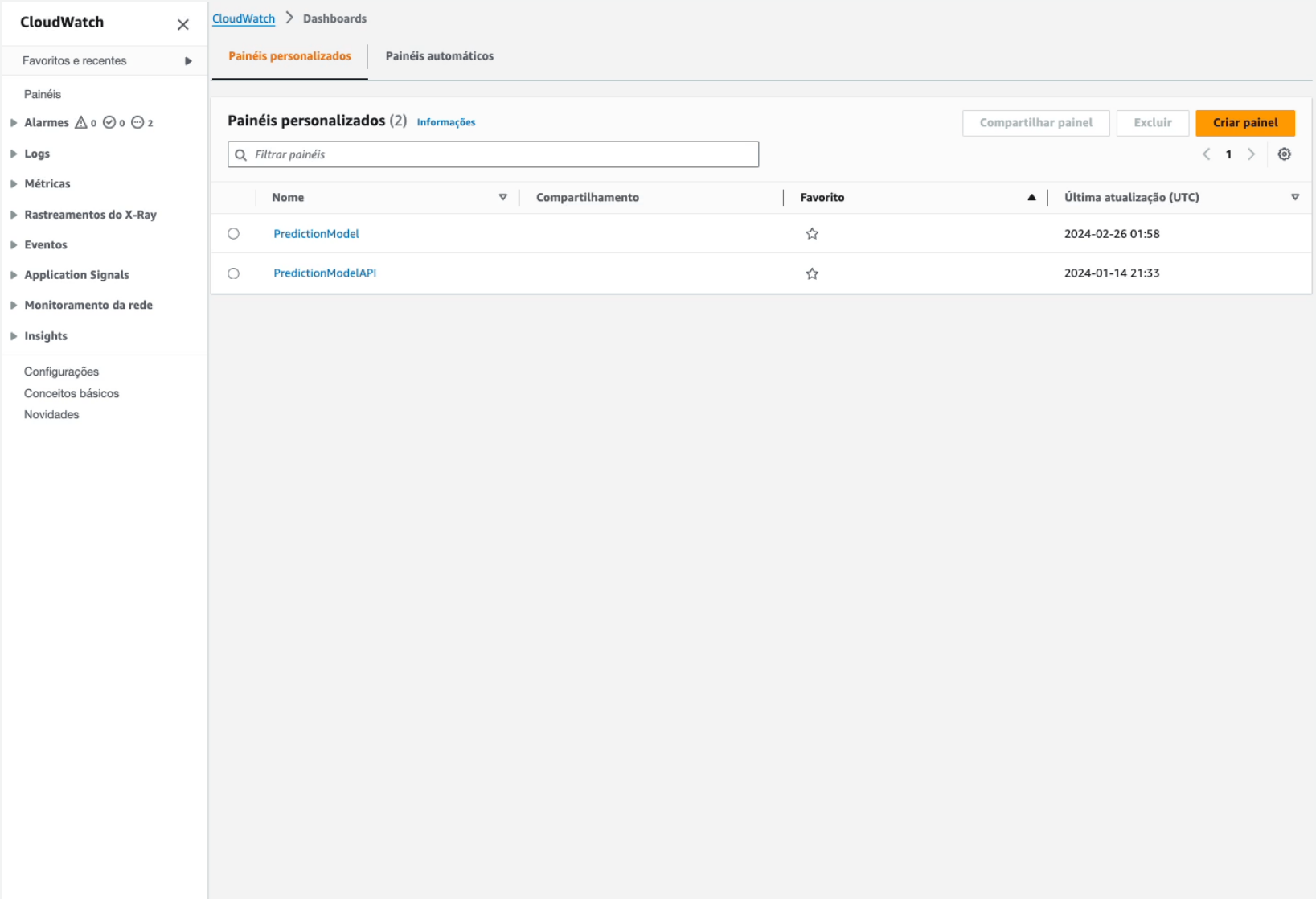
Utilizar a saída de console utilizando o comando “print” é o mais simples possível. Para controlar melhor diferentes granularidades considere utilizar o Python Logging (https://docs.python.org/pt-br/3/howto/logging.html).

PAINÉIS PERSONALIZADO DO MODELO

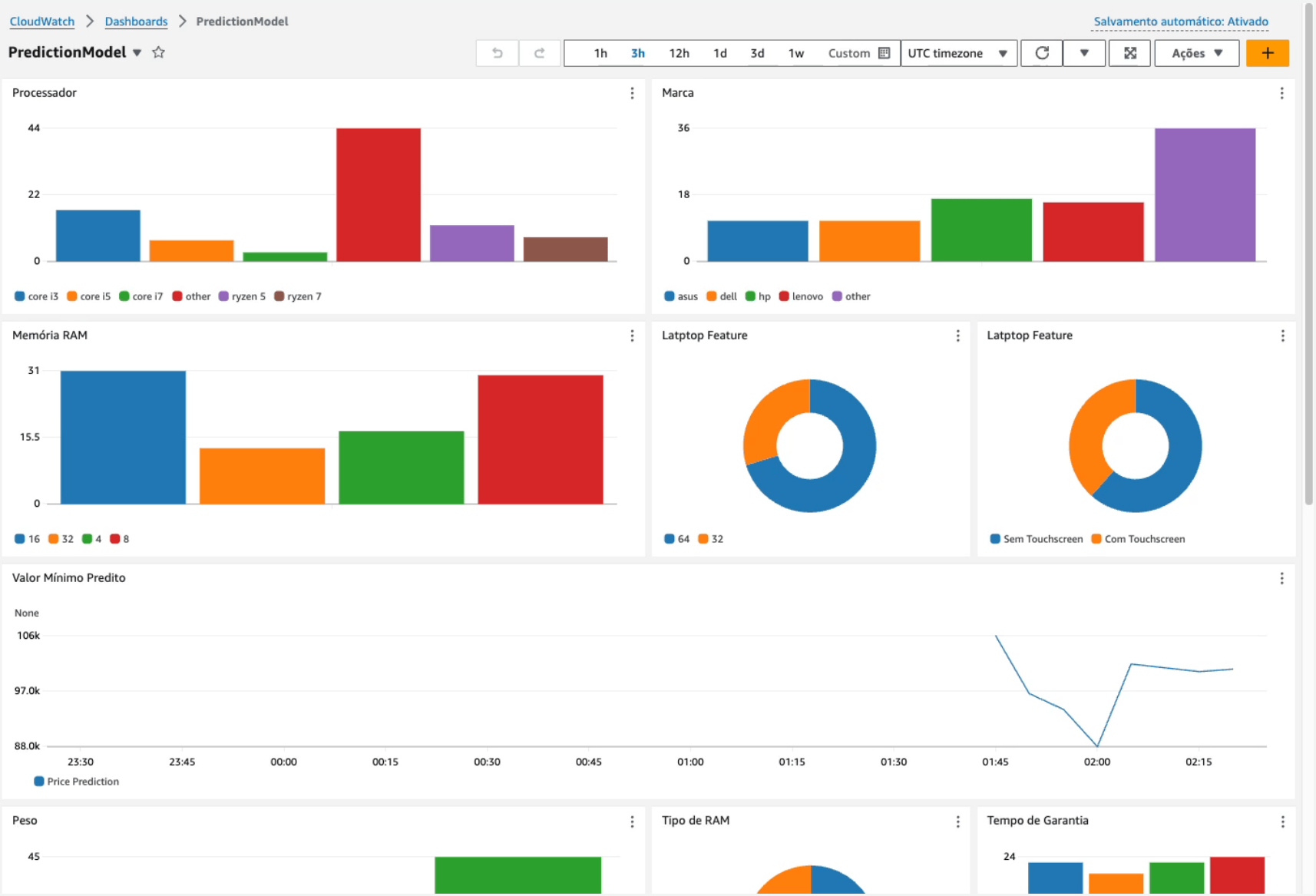
As métricas customizadas que criamos e enviamos em cada requisição traz dados quantitativos dos features e assim conseguimos acompanhar eventuais grandes desvios.

Também é possível acompanhar a média de predições, caso o modelo apresente algum desvio significativo.

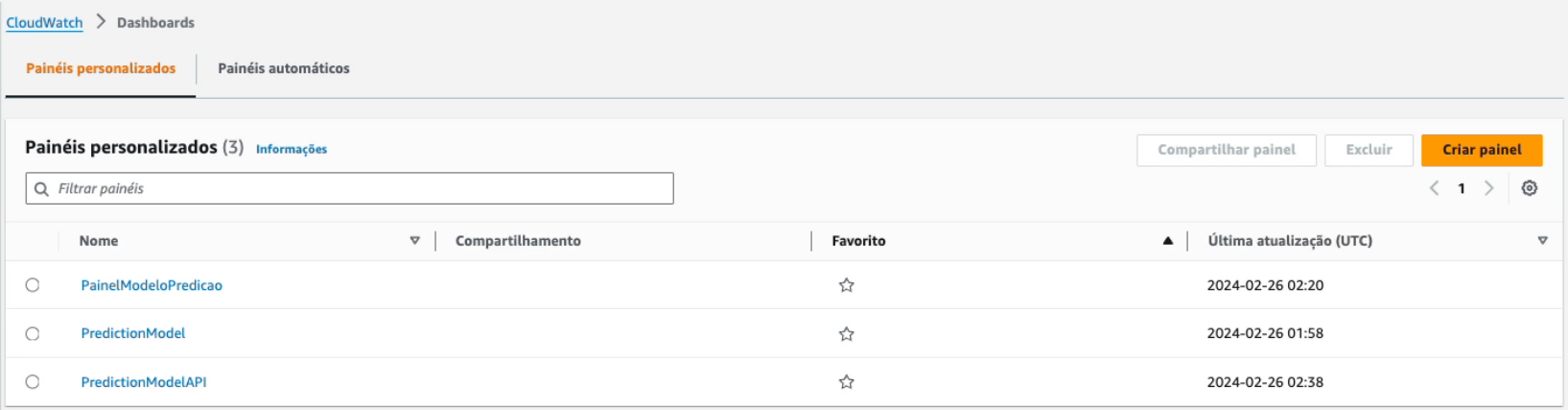
Estas visões são de curto prazo, para acompanhamento diário. Análises mais completas de desvios de dados sevem ser feitas num contexto estatístico mais amplo que veremos mais adiante.



EXEMPLO DE PAINEL DO MODELO COMPLETO



PAINÉIS PERSONALIZADO DO SERVIÇO

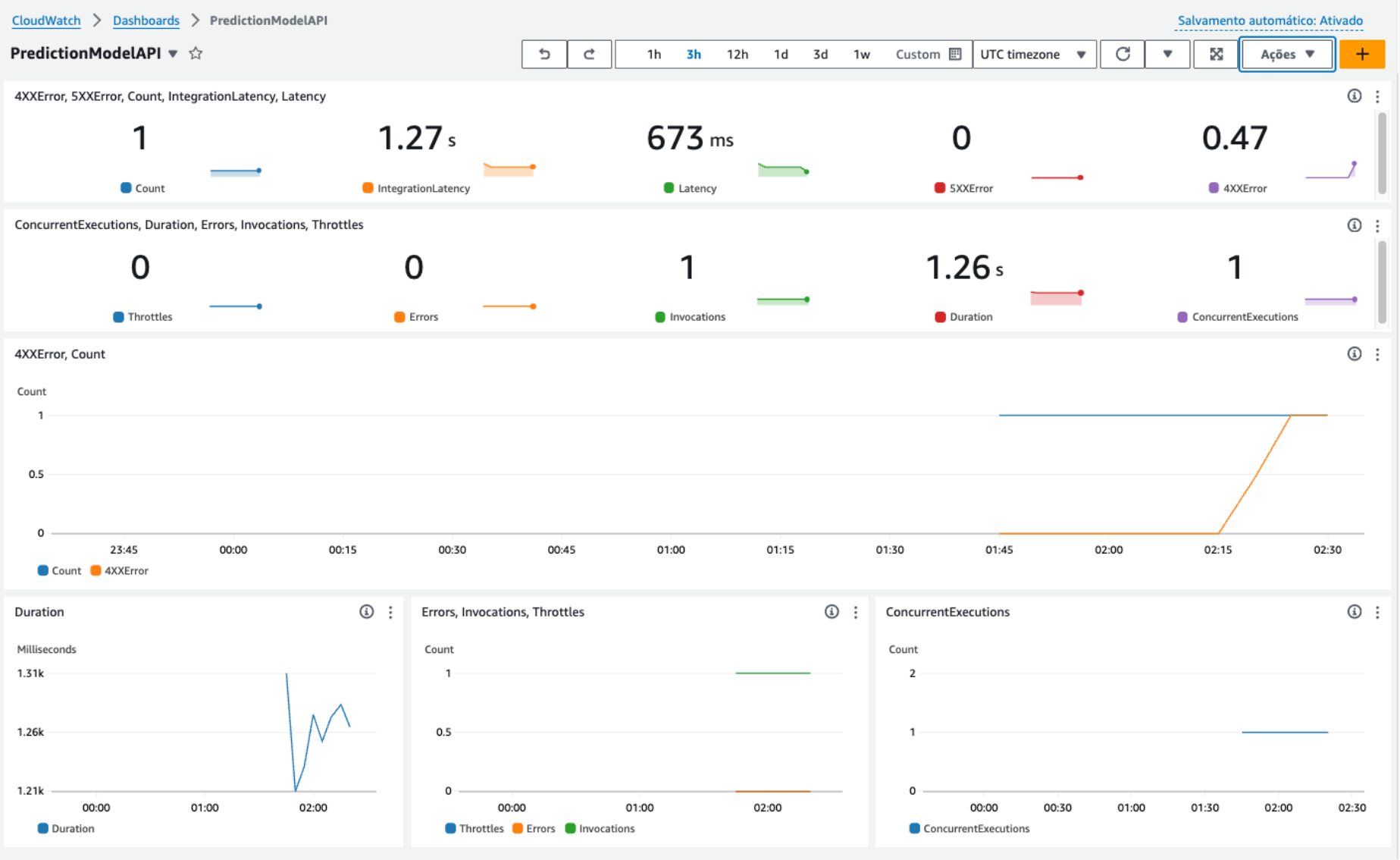


As métricas de serviço estão associadas aos componentes da AWS utilizados.

Na nossa arquitetura, o API Gateway serve de porta de entrada das solicitações, para depois enviar ao Lambda. Com isso, é importante entender a quantidade de requisições, latência para responder o serviço, quantidade de erros, tempo de execução, etc.

O entendimento de como a topologia funciona pode ajudar na identificação de erros e gargalos no serviço.

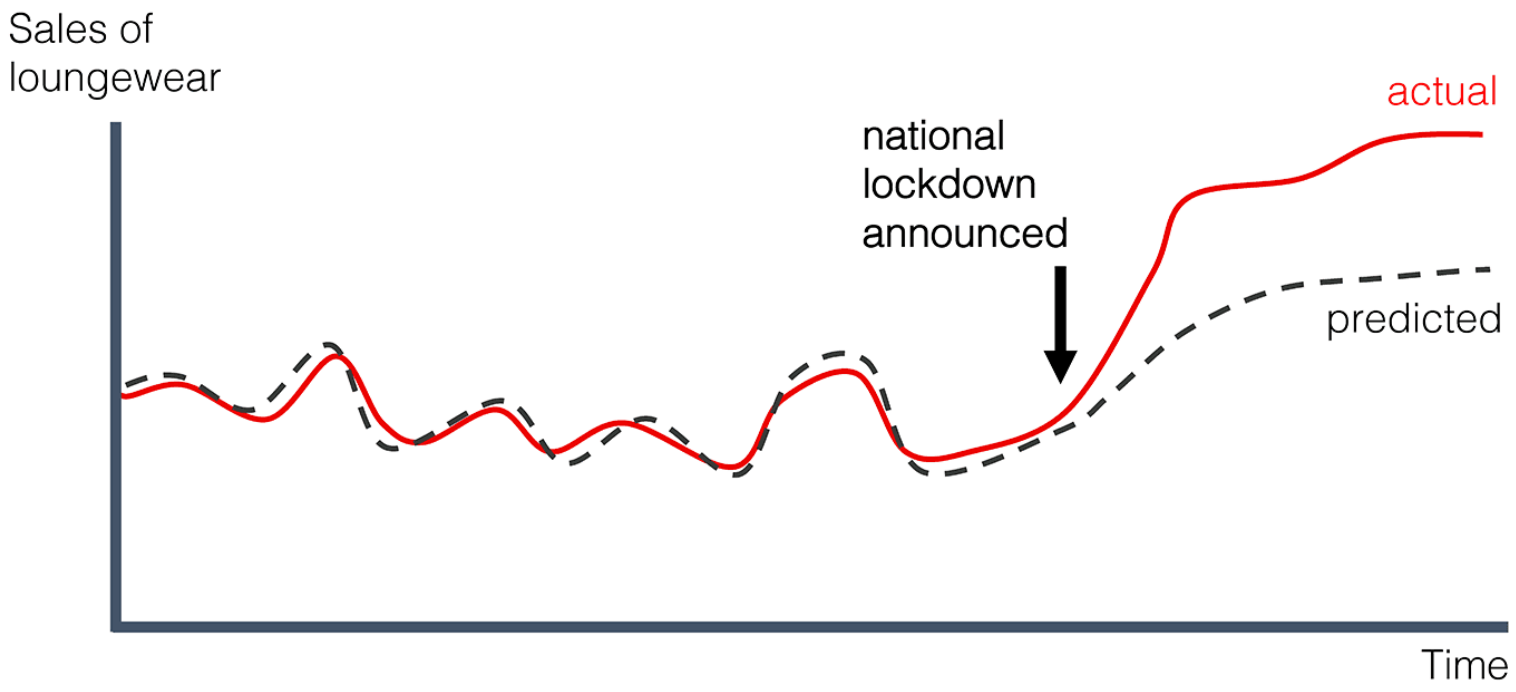
EXEMPLO DE PAINEL DO SERVIÇO



DESVIO DE MODELO E DADOS

DESVIO DE MODELO

Mudança gradual ou súbita nas características dos dados de entrada em um modelo de machine learning após sua implementação. Isso pode ocorrer devido a várias razões, como mudanças nas preferências dos usuários, alterações nos padrões de comportamento, ou até mesmo falhas nos sistemas de coleta de dados.



Data & Concept Drift pode impactar negativamente o desempenho do modelo, pois ele foi treinado em dados históricos que podem não mais representar com precisão a realidade atual.

Como resultado, o modelo pode produzir previsões menos precisas ou confiáveis.

Conceito | Dados

https://blog.nimblebox.ai/machine-learning-model-drift

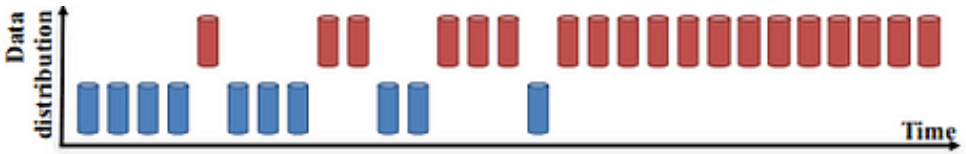
DESVIO DE DADOS & CONCEITO

Desvio de Dados: As entradas mudaram, tornando o modelo treinado irrelevante para os novos dados. Um exemplo disso é a mudança na distribuição etária dos usuários de determinado aplicativo ao longo do tempo. Assim, um modelo treinado em uma distribuição etária específica, utilizada para estratégias de marketing, precisará ser ajustado à medida que a mudança na faixa etária afetará as estratégias de marketing.

Desvio de Conceito: A distribuição dos dados permanece a mesma, mas a relação entre as entradas e saídas é diferente. Isso significa que o que estamos tentando prever mudou. Um exemplo comum é a detecção de spam: os spammers adotam novas táticas ao longo do tempo, exigindo que os filtros de spam sejam ajustados para reconhecer esses novos padrões.

https://kdimensions.com/
https://www.iguazio.com/questions/what-is-the-difference-between-data-drift-and-concept-drift/

DESVIO DE CONCEITO

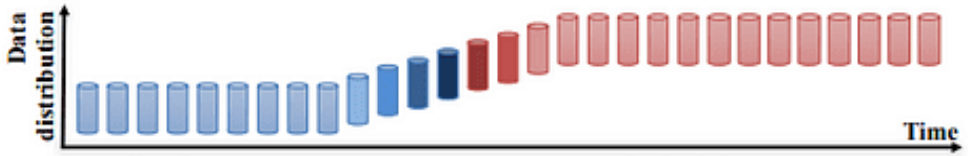


Gradual Drift:
A new concept gradually replaces an old one over a period of time.

Em um desvio gradual de conceito, a relação entre a entrada e o alvo pode mudar lentamente e de forma sutil. Isso pode resultar em uma queda gradual no desempenho de um modelo de aprendizado de máquina, à medida que o modelo se torna menos preciso ao longo do tempo.

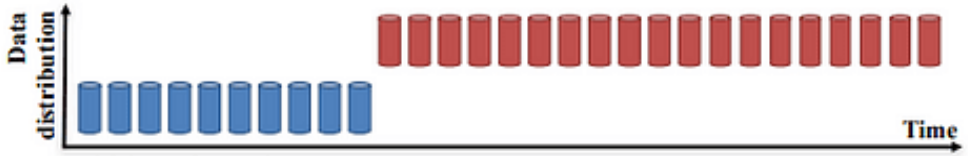
Um exemplo de desvio gradual de conceito é o comportamento fraudulento. Os fraudadores tendem a entender como o sistema de detecção de fraudes funciona e mudam seu comportamento ao longo do tempo para escapar do sistema.

Portanto, um modelo de aprendizado de máquina treinado com dados históricos de transações fraudulentas não preverá com precisão as mudanças graduais no comportamento dos fraudadores.



Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.

O desvio incremental ocorre quando a relação entre a variável alvo e a entrada muda gradualmente ao longo do tempo, o que geralmente ocorre devido a mudanças no processo de geração de dados.



Sudden Drift:
A new concept occurs within a short time.

Desvio repentino de conceito ocorre quando a relação entre as variáveis independentes e dependentes muda de forma repentina.

Um exemplo é o surgimento repentino da pandemia de COVID-19.

A ocorrência da pandemia mudou repentinamente a relação entre a variável alvo e as características em diferentes áreas, de modo que um modelo preditivo treinado com dados prévios não será capaz de prever com precisão durante o período da pandemia.

Reoccurring Concepts:
An old concept may reoccur after some time.

Desvio sazonal. Um exemplo típico é o aumento nas vendas durante o Natal ou a Black Friday.

Um modelo de aprendizado de máquina que não leva em conta essas mudanças sazonais acabará fornecendo previsões imprecisas para essas variações sazonais.

https://arxiv.org/abs/2004.05785

ABORDAGENS PARA **DETECÇÃO**

As técnicas abaixo algumas ferramentas relevantes para a análise e comparação de distribuições estatísticas. Amplamente empregadas na estatística e na análise de dados, essas abordagens são essenciais para quantificar a similaridade ou disparidade entre diferentes distribuições de probabilidade.

Teste de Kolmogorov-Smirnov: é uma técnica estatística usada para determinar se duas distribuições de dados diferem significativamente uma da outra. Em termos simples, ele mede a similaridade entre duas distribuições de probabilidade. Pode ser aplicado para comparar a distribuição das previsões do modelo com a distribuição dos valores reais.

Índice de Estabilidade Populacional: é uma métrica que avalia a estabilidade das distribuições de variáveis ao longo do tempo. É comumente usados em modelos de score de crédito e outras aplicações onde é importante monitorar se a distribuição dos dados mudou significativamente entre períodos. O PSI pode ser usado para identificar desvios nos dados de treinamento e dados em tempo real.

Distância de Wasserstein: é uma métrica que quantifica a dissimilaridade entre duas distribuições de probabilidade. Ela é especialmente útil quando as distribuições têm formas diferentes ou quando há uma correspondência entre elementos das duas distribuições. Em modelos de machine learning, a distância de Wasserstein pode ser usada para comparar a distribuição das previsões do modelo com a distribuição dos valores reais.

Divergência de Kullback-Leibler: é uma medida da diferença entre duas distribuições de probabilidade. Ela quantifica o quanto uma distribuição difere de outra. A divergência de Kullback-Leibler pode ser usada para avaliar a discrepância entre a distribuição das previsões do modelo e a distribuição dos valores reais.

EVIDENTLY

A biblioteca Evidently é uma biblioteca de código aberto projetada para ajudar na validação e interpretação de modelos de machine learning. Ela é usada principalmente para avaliar a performance e a interpretabilidade de modelos, incluindo os casos de desvio de dados e de modelo, dentro de monitoramento de modelos.

Principais recursos:

Análise de desempenho do modelo: fornece métricas e visualizações para avaliar a qualidade preditiva de um modelo. Isso inclui métricas comuns, como precisão, recall, F1-score, entre outras, além de gráficos que ajudam a entender como o modelo se comporta em diferentes partes do conjunto de dados.

Análise de bias e equidade: ajuda a identificar e quantificar vieses e disparidades nos resultados do modelo, permitindo uma análise mais profunda da equidade do modelo em diferentes subgrupos da população.

Interpretabilidade do modelo: oferece ferramentas para interpretar e explicar as decisões do modelo, incluindo a importância de cada feature, análise de feature interactions e visualizações que ajudam a entender como as features influenciam as previsões do modelo.

Monitoramento contínuo do modelo: pode ser usada para monitorar a performance de modelos em produção ao longo do tempo, ajudando a detectar desvios de performance e a tomar medidas corretivas quando necessário.

UTILIZANDO **EVIDENTLY**

FIAP MBA+

Instalando a biblioteca.

```
!pip install evidently==0.4.15
```

Baixando os dados de análise.

```
!git clone https://github.com/michelpf/fiap-ds-mlops-laptop-pricing-model-drift
df_training = pd.read_csv("training-data/laptop-pricing.csv")
df_training.head()
```

Gerando desvio artificial em 5 desvios padrão.

```
df_training.rename(columns={'price': 'target'}, inplace=True)
df_training['prediction'] = df_training['target'].values + np.random.normal(0, 5,
df_training.shape[0])
```

Obtendo amostras para 2 conjuntos, um de treinamento que será utilizado como referência e outro para produção que será utilizado como dado atual.

```
training = df_training.sample(n=500, replace=False)
production = df_training.sample(n=500, replace=False)
```

https://github.com/michelpf/fiap-ds-mlops-laptop-pricing-model-drift

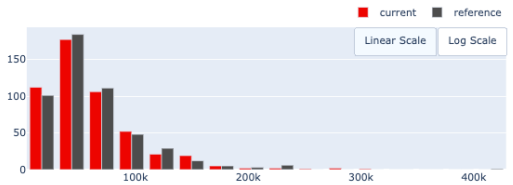
ANÁLISE DE **DADOS**

Obtendo relatório visual para análise.

```
data_quality_report = Report(metrics=[DataQualityPreset()])
data_quality_report.run(reference_data=training, current_data=production)
data_quality_report
```

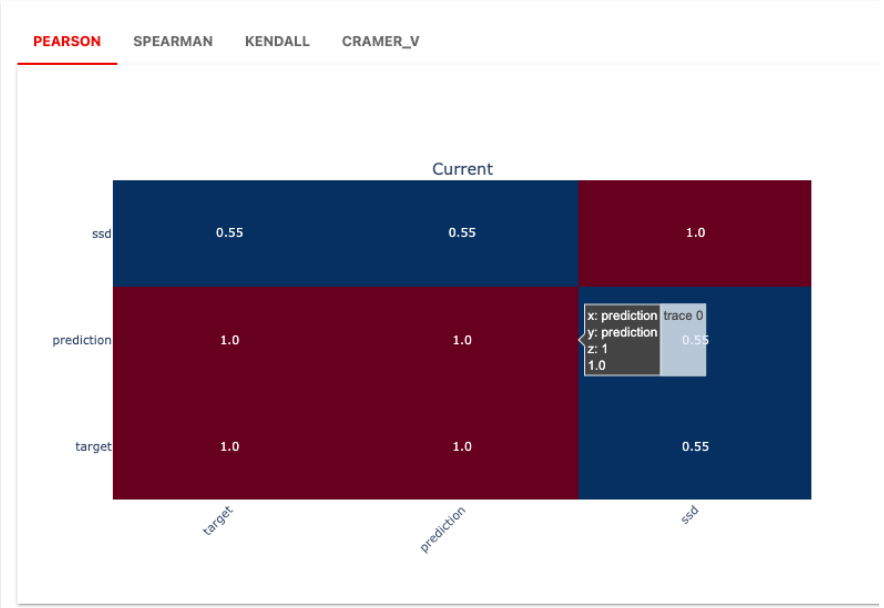
target
num

	current	reference
count	500	500
mean	74654.32	75470.75
std	41847.27	42767.24
min	16990.0	16990.0
25%	44990.0	46990.0
50%	63490.0	64990.0
75%	86240.0	89435.0
max	309990.0	441990.0
unique	290 (58.0%)	298 (59.6%)
most common	64990.0 (1.8%)	64990.0 (2.4%)
missing	0 (0.0%)	0 (0.0%)
infinite	0 (0.0%)	0 (0.0%)



Análise para cada coluna do dataset

Análise de correlação entre os atributos.

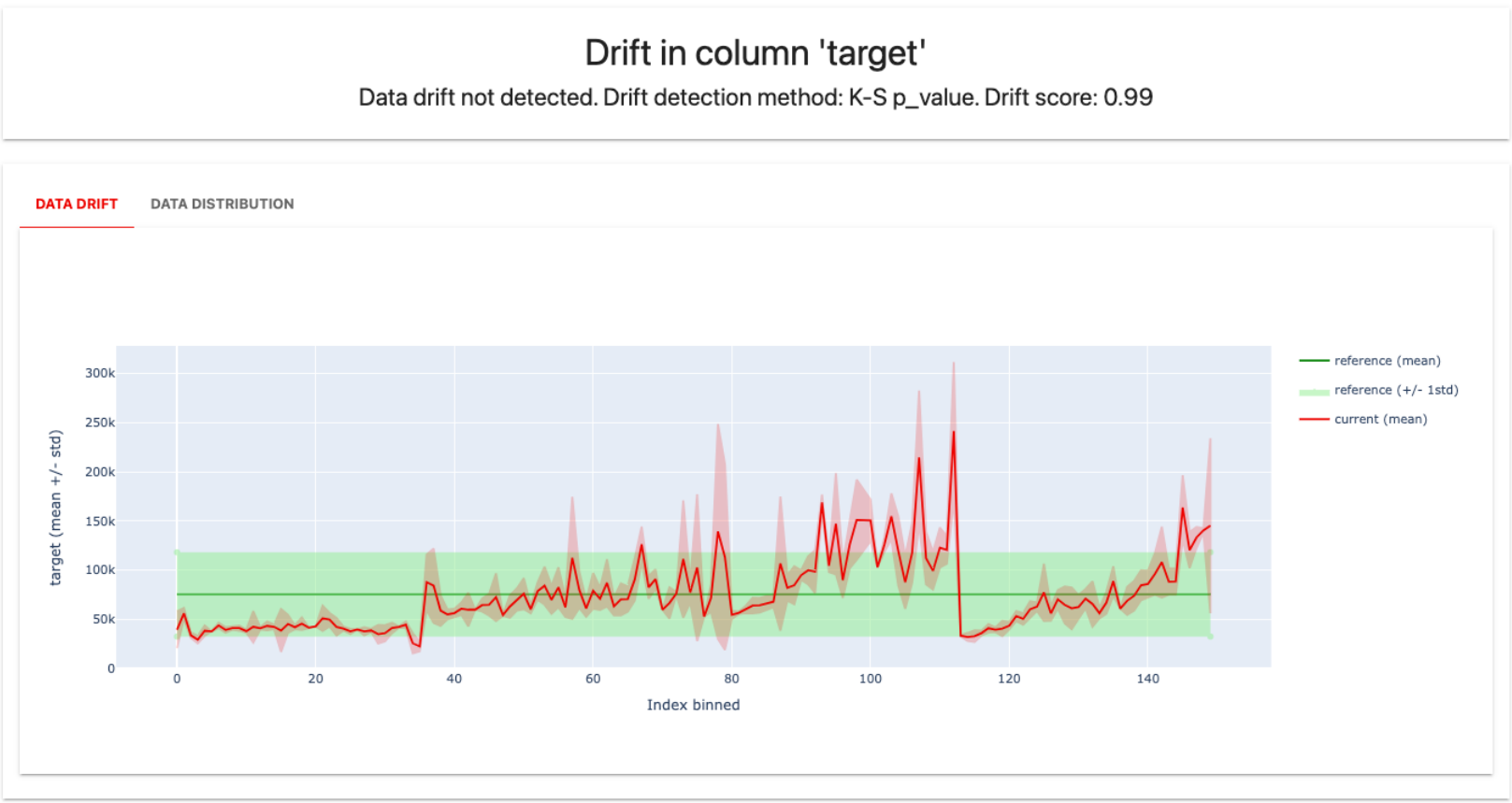


ANÁLISE DE **DESVIO DE MODELO (ALVO)**

FIAP MBA+

Obtendo relatório visual para análise.

```
report = Report(metrics=[TargetDriftPreset()])
report.run(reference_data=training, current_data=production)
report
```



ANÁLISE DE **DESVIO DE DADOS (CONCEITO)**

FIAP MBA+

Obtendo relatório visual para análise.

```
report = Report(metrics=[DataDriftPreset()])
report.run(reference_data=training, current_data=production)
report
```

Dataset Drift

Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5

15	0	0.0
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 0.0% of columns (0 out of 15).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
target	num			Not Detected	K-S p_value	0.98968
prediction	num			Not Detected	K-S p_value	0.98968
ssd	num			Not Detected	K-S p_value	0.995953

Este relatório pode ser disponibilizado em formato JSON que facilita a integração com sistemas de monitoramento para notificações de desvios de dados relavantes.

```
report.json()
```

Os testes estatísticos são definidos automaticamente baseado no tamanho e tipo do dataset.

É possível determinar testes específicos dos diversos disponíveis.

CONCLUSÕES SOBRE DESVIO DE MODELOS E DADOS

Tamanho do desvio que deseja detectar: dependendo do caso apenas vale investir somente em mudanças grandes. Em outros, até mesmo a uma pequena. A definição de "grande" e "pequeno" variaria com base nos seus dados e a criticidade do negócio envolvido.

Tamanho das amostras para comparação: como diferentes testes fornecem resultados diferentes dependendo do tamanho da amostra, você deve considerar isso para o tipo de teste a ser utilizado.

O custo da queda no desempenho do modelo: se cada erro for caro, você pode querer escolher um teste mais sensível, mesmo que isso resulte em alertas de falsos positivos.

Se a precisão for crucial, escolha o Teste de Kolmogorov-Smirnov (KS).

Se você deseja detectar um desvio maior, escolha o Teste de Wasserstein (WD). Você pode definir um limiar de acordo com o número de desvios padrão que você assume que a distribuição deve mudar para se qualificar como um desvio.

PARA SABER **MAIS**

https://www.evidentlyai.com/blog/data-drift-detection-large-datasets, artigo sobre observabilidade de modelos e desvios, da Evently AI.
https://cursos.alura.com.br/course/observabilidade-aws-utilizando-cloudwatch, curso de Observabilidade com Cloudwatch