

FIAP

NABA



# COGNITIVE ENVIRONMENTS

DATA SCIENCE & AI MBA



# REKOGNITION **MODERATION LABELS**

O **Rekognition Moderation Labels** ajudam a identificar conteúdo potencialmente inadequado em imagens.

Esses rótulos são gerados por algoritmos treinados para reconhecer características associadas a diferentes tipos de conteúdo inadequado, como nudez, violência, drogas, álcool, entre outros. Ao analisar uma imagem, o modelo pode atribuir um ou mais desses rótulos à imagem, indicando a presença de conteúdo considerado sensível ou impróprio.

O serviço é útil em uma variedade de cenários, incluindo redes sociais, plataformas de compartilhamento de conteúdo, fóruns online e aplicativos que desejam manter um ambiente seguro para os usuários. Ao detectar automaticamente conteúdo potencialmente inadequado, esses rótulos ajudam na moderação e filtragem de conteúdo, permitindo que as empresas e desenvolvedores tomem medidas adequadas, como remover ou ocultar conteúdo sensível.

# REKOGNITION **MODERATION LABELS**

A detecção de conteúdos explícitos pode ser regulada por meio de intervalos de confiança menor ou maior, de acordo com a sensibilidade de cada caso de uso.



# MODERATION LABELS: INFERÊNCIA

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_KEY, aws_secret_access_key=
ACCESS_SECRET)

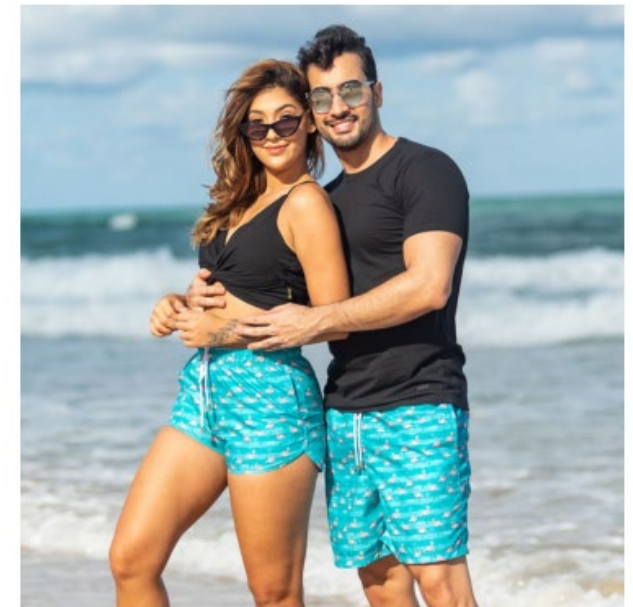
# criar cliente
client = session.client('rekognition', region_name=REGION)
```

```
with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_moderation_labels(
    Image={'Bytes': bytes_img})
```

response

```
{'ModerationLabels': [], 'ModerationModelVersion': '7.0', 'ContentTypes': [], 'ResponseMetadata':
{'RequestId': 'f6d5b930-bc45-49c3-ac2e-8601e565b689', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': 'f6d5b930-bc45-49c3-ac2e-8601e565b689', 'content-type': 'application/x-amz-json-1.1',
'content-length': '72', 'date': 'Thu, 08 Feb 2024 00:43:36 GMT'}, 'RetryAttempts': 0}}
```





# MODERATION LABELS: INFERÊNCIA

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

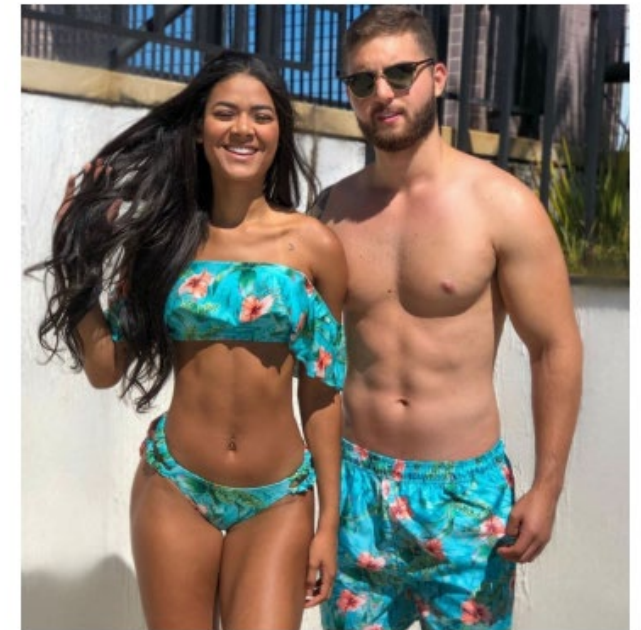
```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_KEY, aws_secret_access_key= ACCESS_SECRET)
# criar cliente
client = session.client('rekognition', region_name=REGION)

with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_moderation_labels(
    Image={'Bytes': bytes_img})

response

{'ModerationLabels': [{'Confidence': 97.43789672851562,
  'Name': 'Non-Explicit Nudity of Intimate parts and Kissing',
  'ParentName': '', 'TaxonomyLevel': 1},
  {'Confidence': 97.43789672851562, 'Name': 'Non-Explicit Nudity',
  'ParentName': 'Non-Explicit Nudity of Intimate parts and Kissing', 'TaxonomyLevel': 2},
  {'Confidence': 97.43789672851562, 'Name': 'Exposed Male Nipple',
  'ParentName': 'Non-Explicit Nudity', 'TaxonomyLevel': 3},
  {'Confidence': 58.55329895019531, 'Name': 'Swimwear or Underwear', 'ParentName': '', 'TaxonomyLevel': 1},
  {'Confidence': 58.55329895019531, 'Name': 'Female Swimwear or Underwear',
  'ParentName': 'Swimwear or Underwear', 'TaxonomyLevel': 2}]]...
```



# MODERATION LABELS: INFERÊNCIA

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_KEY, aws_secret_access_key= ACCESS_SECRET)
# criar cliente
client = session.client('rekognition', region_name=REGION)

with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_moderation_labels(
    Image={'Bytes': bytes_img})

response

{'ModerationLabels': [{ 'Confidence': 82.22530364990234,
    'Name': 'Weapons',
    'ParentName': 'Violence', 'TaxonomyLevel': 2},
  { 'Confidence': 82.22530364990234,
    'Name': 'Violence', 'ParentName': '', 'TaxonomyLevel': 1},
  { 'Confidence': 78.94529724121094, 'Name': 'Weapon Violence',
    'ParentName': 'Graphic Violence', 'TaxonomyLevel': 3},
  { 'Confidence': 78.94529724121094, 'Name': 'Graphic Violence',
    'ParentName': 'Violence', 'TaxonomyLevel': 2...
```





# REKOGNITION: **CUSTOM LABELS**

Amazon Rekognition Custom Labels é uma ferramenta para treinar modelos de visão computacional personalizados, permitindo a criação soluções de inteligência artificial adaptadas às suas necessidades específicas.

**Treinamento Personalizado de Modelos:** permite treinar modelos de machine learning personalizados usando conjuntos de dados próprios. Isso permite que você ensine ao modelo para reconhecer objetos ou padrões específicos relevantes para o seu caso de uso, em vez de depender de modelos genéricos pré-treinados.

**Detecção de Objetos e Classificação de Imagens:** treinamento de modelos para realizar tarefas como detecção de objetos em imagens, onde o modelo identifica e delimita áreas que contêm objetos específicos, ou classificação de imagens, onde o modelo atribui rótulos ou tags a imagens com base em seu conteúdo.

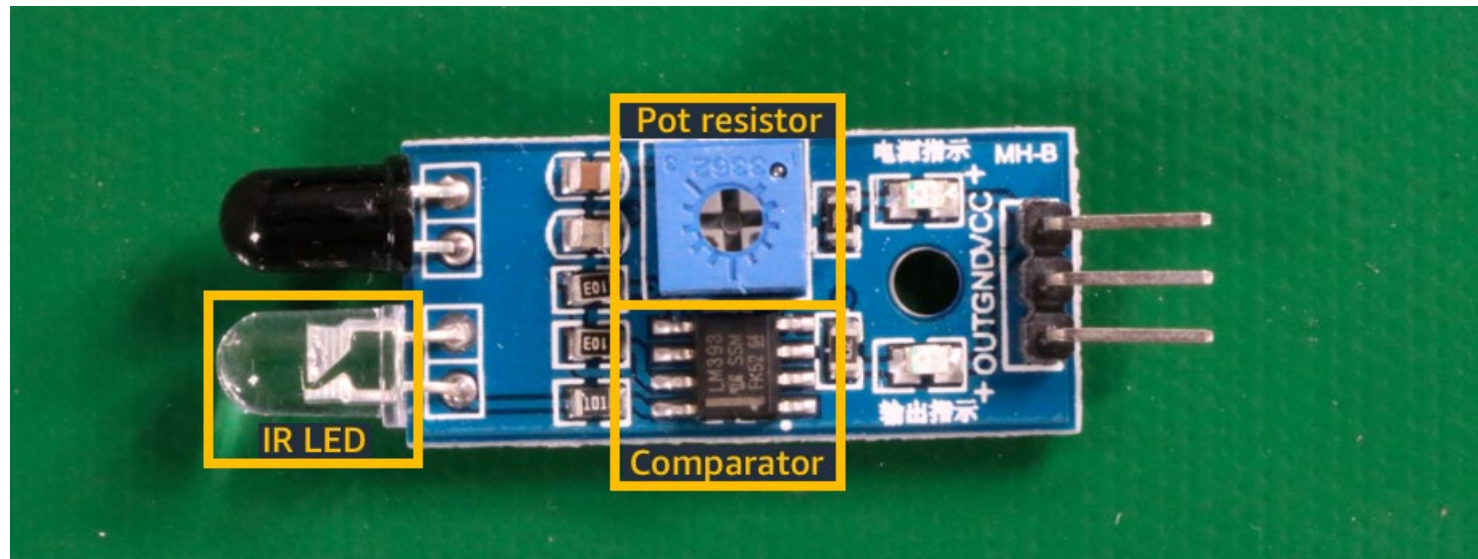
**Facilidade de Uso:** possui interface fácil de usar para treinar, avaliar e implantar modelos de machine learning personalizados. Você pode interagir com o serviço por meio da console da AWS, da linha de comando ou da API.

**Integração com AWS:** se integra perfeitamente a outros serviços da AWS, como Amazon S3 para armazenamento de dados de treinamento, AWS Lambda para processamento de eventos em tempo real e Amazon SageMaker para treinamento avançado de modelos.

**Alta Precisão e Escalabilidade:** é projetado para fornecer resultados precisos e escaláveis, mesmo com grandes volumes de dados e imagens. Ele utiliza algoritmos de deep learning e infraestrutura de computação em nuvem da AWS para alcançar desempenho e precisão.

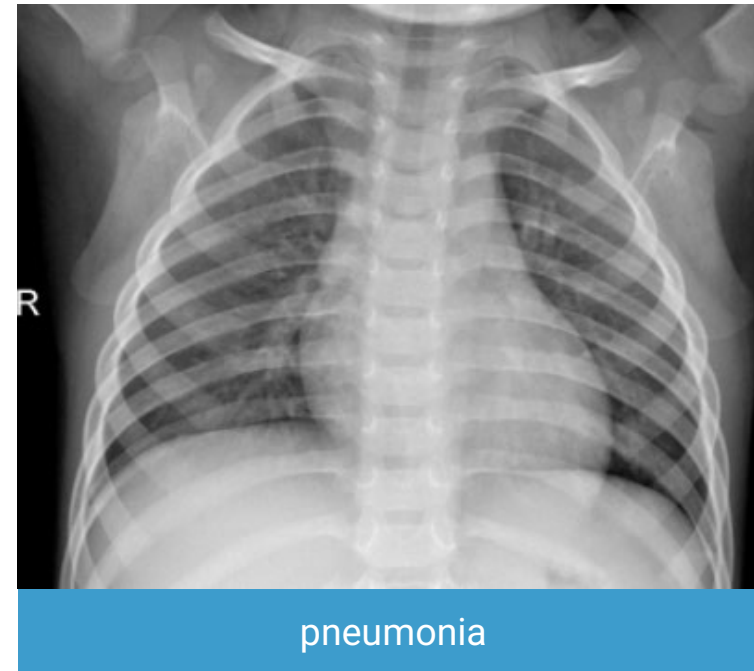
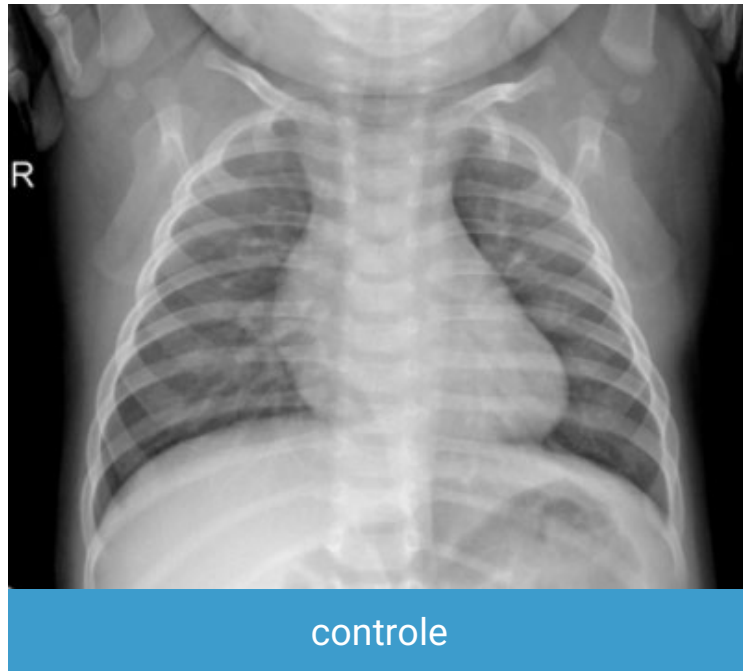
# REKOGNITION: **CUSTOM LABELS**

Detecção de objetos permite identificar na imagem regiões específicas e delimitar por caixas delimitadoras. Em uma imagem é possível ter inúmeras classes de objetos.



# REKOGNITION: **CUSTOM LABELS**

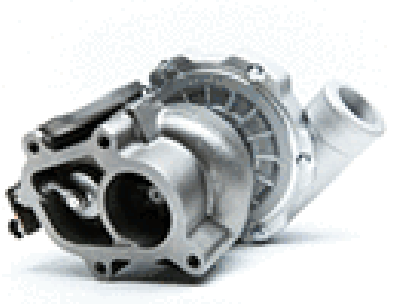
A classificação baseado em rótulos analisa a imagem como um todo, atribuindo uma classe para cada uma delas. No caso abaixo, uma única classe classifica uma imagem de raio-x de pulmão de uma pessoa controle (sem enfermidade) contra uma outra com pneumonia.



# REKOGNITION: **CUSTOM LABELS**

É possível combinar a API padrão, detector de objetos, junto com o Custom Labels.

A primeira análise pode servir para encontrar uma região de interesse, e então aplicar o segundo modelo para detalhar o objeto.



Machine Parts

Turbocharger



Machine Parts

Torque Converter



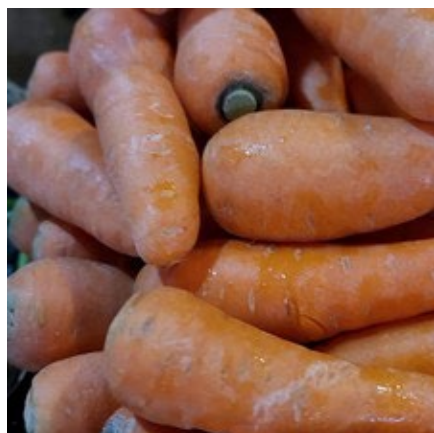
Machine Parts

Crankshaft

Rekognition Object and Scene Labels

Rekognition Custom Labels

# CLASSIFICADOR DE IMAGENS COM CUSTOM LABELS



Para a classificação de imagens, vamos utilizar um conjunto de exemplos de tipos de vegetais. Cada tipo possui 1000 imagens e o nome do diretório é o rótulo.

Este modelo não requer muitas imagens, começando com 30 é possível ter ótimos resultados (precisão > 90%).

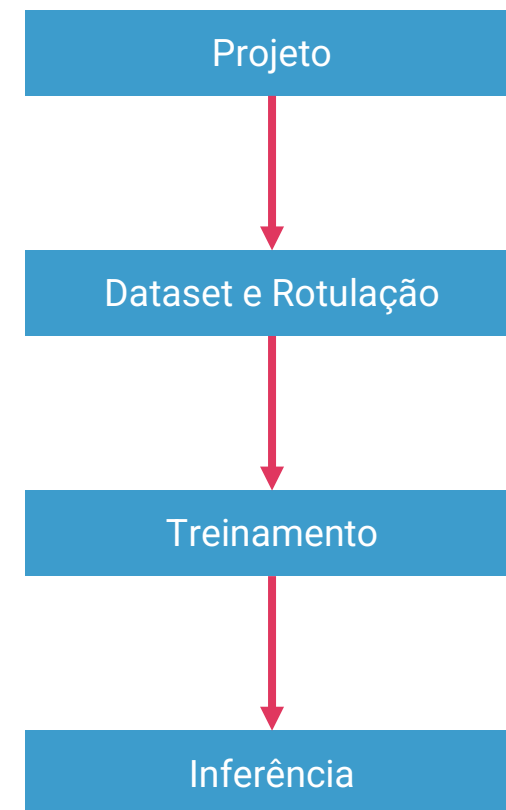
Fonte [Kaggle](#) também disponível neste [repositório](#).

# CLASSIFICAÇÃO DE IMAGENS

Para criar um projeto de classificação de imagens, acessamos pelo console o serviço Rekognition, depois no menu lateral acessamos o Custom Labels.

Após entrar no painel do Custom Labels, criamos um projeto novo que servirá de base para todo o pipeline de construção do modelo

The screenshot shows the Amazon Rekognition Custom Labels console. On the left is a sidebar with the title 'Amazon Rekognition Custom Labels' and a close button. Below the title are links: 'Comece a usar', 'Projetos' (with a sub-link 'Atualizado'), 'Definição de preço' (with an external link icon), and 'Documentação' (with an external link icon). The main content area is titled 'Custom Labels > Create project' and 'Criar projeto' (with an 'Info' link). A light blue informational box contains an 'i' icon, the title 'Criar projeto', and a close button. The text inside the box explains that creating a project is the first step in building a custom model. Below this box is a section titled 'Detalhes do projeto' containing a text input field for 'Nome do projeto' with the value 'Transportes'. A note below the input field states: 'O nome do projeto não pode ter mais de 63 caracteres. Ele só pode conter caracteres alfanuméricos, sem espaços nem caracteres especiais.' At the bottom right of the form are two buttons: 'Cancelar' and 'Criar projeto' (in orange).





# CLASSIFICAÇÃO DE IMAGENS: DATASET

A primeira etapa é de criação e rotulação do dataset.

As imagens utilizadas podem estar em um bucket do S3 ou serem enviadas a partir do computador (via upload). Para este projeto vamos enviar as imagens por upload.

## Transportes<sup>Info</sup>

### ▼ Como funciona

#### Criação do seu conjunto de dados



##### 1. Criar conjunto de dados

Um conjunto de dados é uma coleção de imagens e rótulos de imagens que você usa para treinar ou testar um modelo.

**Criar conjunto de dados**



##### 2. Rótulos de imagens

Os rótulos identificam objetos, cenas ou conceitos em uma imagem inteira, ou identificam a localização de objetos em uma imagem.

**Adicionar rótulo**

#### Treinamento do seu modelo



##### 3. Treinar modelo

Dependendo do conjunto de dados de treinamento, o modelo de treinamento localiza cenas e conceitos em nível de imagem ou define a localização de objetos.

**Treinar modelo**

#### Avaliação do seu modelo



##### 4. Verifique as métricas de desempenho

As métricas de desempenho informam se o modelo precisa de treinamento adicional antes de você usá-lo.

**Verificar métricas**

# CLASSIFICAÇÃO DE IMAGENS: DATASET

Após enviarmos as imagens, precisamos rotular cada uma delas.

No menu lateral temos filtros para imagens rotuladas, não rotuladas ou com erro.

Para iniciar a rotação clicamos em “Iniciar rotulagem”.

*Se não for especificado conjuntos de dados separados para treinamento e testes, a plataforma vai separar 80% das imagens para treinamento e 20% para teste.*

[Custom Labels](#) > [Projects](#) > [Transportes](#) > Dataset

Conjunto de dados [Info](#)

Iniciar rotulagem

Ações ▼

Treinar modelo

## ▼ Preparação do seu conjunto de dados



### 1. Review dataset

Verify that your images are labeled correctly. If the dataset needs more images, choose Actions and then the appropriate dataset under Add Images. [Saiba mais](#)



### 2. Add labels

You add labels for each type of object, scene, or concept in your dataset. To add or modify labels, choose Start labeling and then choose Edit labels. [Saiba mais](#)



### 3. Label images

Choose the images that you want to label. If you need to label an entire image, choose Assign labels and assign image-level labels. If you need to label object locations, Choose Draw bounding boxes. Then draw bounding boxes around objects and assign labels. Choose Save changes to finish. [Saiba mais](#)



### 4. Train model

After your datasets are ready, Choose Train model to train your model. Then, evaluate and use the model to find objects, scenes, and concepts in new images. [Saiba mais](#)

## Rótulos

Adicionar rótulo

☒ Imagens (10)

☐ Rotuladas (0)

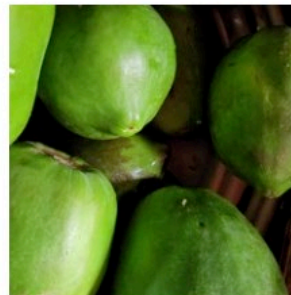
☐ Não rotuladas (10)

☐ Erros (0)

## Imagens (120)

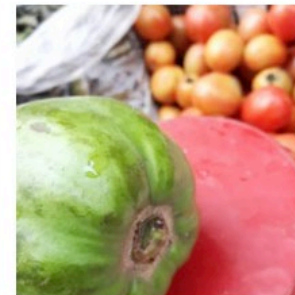
< 1 2 3 ... >

0001.jpg



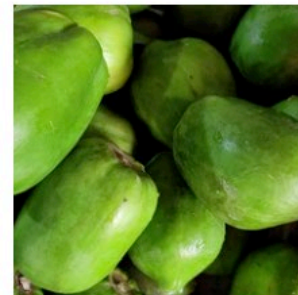
☒ Papaya ✕

0002.jpg



☒ Papaya ✕

0003.jpg



☒ Papaya ✕

# CLASSIFICAÇÃO DE IMAGENS: ROTULAGEM

Primeiro, precisamos clicar em “Gerenciar rótulos” para adicionar os rótulos para as imagens. Em nosso projeto iremos adicionar um rótulo para cada tipo de veículo de transporte: “truck”, “car” e “bus”.

The image illustrates the process of adding labels in a web application. It shows a sidebar on the left and a modal window on the right.

**Sidebar (Rótulos):**

- Gerenciar rótulos** (button)
- ☒ **Imagens (120)**
- ☐ **Rotuladas (120)**
- ☐ **Não rotuladas (0)**
- ☐ **Erros (0)**
- 
- ☐ **Tomate (24)**
- ☐ **Batata (24)**
- ☐ **Abóbora (24)**
- ☐ **Cenoura (24)**
- ☐ **Papaya (24)**

**Modal (Gerenciar rótulos):**

Rótulos são objetos, cenas ou conceitos que o modelo será treinado para identificar nas imagens.

O nome do rótulo não pode ter mais de 100 caracteres. Cada rótulo deve ser atribuído a um volume de 1 a 1000 imagens.

# CLASSIFICAÇÃO DE IMAGENS: ROTULAGEM

**Atribuir rótulo no nível da imagem às imagens selecionadas** ✕

Rótulos são objetos, cenas ou conceitos que o modelo será treinado para identificar nas imagens.

Selecionar rótulos


✕

Cancelar Atribuir

Imagens (1/120)
Atribuir rótulos em nível de imagem
Caixas delimitadoras


< 1 2 3 ... >

0001.jpg ☒




☒ Papaya ✕

0002.jpg ☐



☐ Papaya ✕

0003.jpg ☐



☐ Papaya ✕

Depois dos rótulos devidamente adicionados, precisamos atribuir as imagens. Selecione a imagem desejada, depois vá em “Atribuir rótulos em nível de imagem”.

# CLASSIFICAÇÃO DE IMAGENS: TREINAMENTO

[Custom Labels](#) > Train model

## Treinar modelo



### Train model



Para treinar seu modelo, o Amazon Rekognition Custom Labels usa o conjunto de dados de treinamento e o conjunto de dados de teste do projeto. Você pode adicionar tags para ajudar a rastrear seus modelos. Você também pode criptografar suas imagens com sua própria chave do AWS Key Management Service.

### Detalhes do treinamento [Info](#)

#### Escolher projeto

O Amazon Rekognition Custom Labels treina uma nova versão do modelo dentro do projeto escolhido.

arn:aws:rekognition:us-east-1:989944764342:project/Transportes/1707064275115

### Tags [Info](#)

Uma tag é um rótulo que você pode atribuir ao seu modelo. Cada tag consiste em uma chave e um valor opcional.

Nenhuma tag associada ao recurso.

Adicionar nova tag

Você pode adicionar até 50 tags a mais.

### Criptografia de dados de imagem

Seus dados são criptografados por padrão com uma chave gerenciada pela AWS. Para escolher uma chave diferente, personalize suas configurações de criptografia. [Saiba mais](#)

☐ Personalizar configurações de criptografia (avanzado)

Após todas as imagens forem rotuladas, estamos prontos para a etapa de treinamento.

Confirme o projeto.

Depois clique em “Treinar Modelo”.

*O treinamento não permite customizações sobre o tempo de treinamento. Ele é feito de acordo com o número de imagens. Pode demorar 30 minutos até 24 horas.*

Cancelar

Treinar modelo

# CLASSIFICAÇÃO DE IMAGENS: TREINAMENTO

Acompanhe o progresso do treinamento na guia “Modelos”.

Quando o status do modelo alterar para “COMPLETED”, ele já pode ser utilizado para as inferências (antes é recomendado verificar as métricas do treinamento para assegurar que cumprem o mínimo requerido para sua utilização).

| Status do modelo ▾ | Mensagem de status ▾       |
|--------------------|----------------------------|
| TRAINING_COMPLETED | The model is ready to run. |

[Custom Labels](#) > [Projects](#) > Transportes

## Transportes<sup>Info</sup>

### ▼ Como funciona

#### Criação do seu conjunto de dados



##### 1. Criar conjunto de dados

Um conjunto de dados é uma coleção de imagens e rótulos de imagens que você usa para treinar ou testar um modelo.

✔ Criado



##### 2. Rótulos de imagens

Os rótulos identificam objetos, cenas ou conceitos em uma imagem inteira, ou identificam a localização de objetos em uma imagem.

Adicionar rótulo

#### Treinamento do seu modelo



##### 3. Treinar modelo

Dependendo do conjunto de dados de treinamento, o modelo de treinamento localiza cenas e conceitos em nível de imagem ou define a localização de objetos.

Treinar modelo

#### Avaliação do seu modelo



##### 4. Verifique as métricas de desempenho

As métricas de desempenho informam se o modelo precisa de treinamento adicional antes de você usá-lo.

Verificar métricas

### Detalhes do projeto

Nome do projeto  
Transportes

Criado  
February 04, 2024 at 13:31:16 (UTC-03:00)

Conjunto de dados  
3 rótulos de treinamento, 24 imagens de treinamento, 3 rótulos de teste, 7 imagens de teste

Modelos  
1

### Modelos (1)

Excluir modelo

Download validation results ▾

🔍 Encontrar recursos

< 1 ... >

| <input type="checkbox"/> | Nome ▾                          | Data de criação ▾ | Conjunto de dados de treinamento ▾ | Testar conjunto de dados ▾ | Performance do modelo (Pontuação F1) ▾ | Status do modelo ▾   | Mensagem de status ▾        |
|--------------------------|---------------------------------|-------------------|------------------------------------|----------------------------|--|----------------------|-----------------------------|
| <input type="checkbox"/> | Transportes.2024-02-04T13.52.02 | February 04, 2024 |                                    |                            | N/A                                    | TRAINING_IN_PROGRESS | The model is being trained. |



# CLASSIFICAÇÃO DE IMAGENS: MÉTRICAS

Pontuação F1Info

1.000

Average precisionInfo

1.000

Overall recallInfo

1.000

Date completed

May 02, 2025

Trained in 0.539 hours

Conjunto de dados de treinamento

5 labels, 120 images

Conjunto de dados de teste

5 labels, 30 images

Desempenho por rótulo (5)

Find labels

<

1

>


| Label name | ▲ | F1 score | ▼ | Test images | ▼ | Precision | ▼ | Recall | ▼ | Assumed threshold | ▼ |
|------------|---|----------|---|-------------|---|-----------|---|--------|---|-------------------|---|
| Abóbora    |   | 1.000    |   | 6           |   | 1.000     |   | 1.000  |   | 0.978             |   |
| Batata     |   | 1.000    |   | 6           |   | 1.000     |   | 1.000  |   | 0.235             |   |
| Cenoura    |   | 1.000    |   | 6           |   | 1.000     |   | 1.000  |   | 0.995             |   |
| Papaya     |   | 1.000    |   | 6           |   | 1.000     |   | 1.000  |   | 0.999             |   |
| Tomate     |   | 1.000    |   | 6           |   | 1.000     |   | 1.000  |   | 0.983             |   |

As métricas indicam o quão bom o modelo foi avaliado com as imagens de teste. Como utilizamos poucas imagens, os indicadores ficaram mais positivos.

Um ponto importante é utilizar o limiar recomendável (assumed threshold) para assumir que determinada imagem é da classe respectiva.


# CLASSIFICAÇÃO DE IMAGENS: INFERÊNCIA

[Avaliação](#) | [Detalhes do modelo](#) | [Usar modelo](#) | [Tags](#)

 Usar modelo ×

Use a guia Usar modelo para iniciar e interromper o modelo. São fornecidos códigos de exemplo para usar o modelo treinado.

### Iniciar ou interromper o modelo

 Interrompido

O modelo não está em execução. Para começar a executar o modelo, escolha Iniciar modelo ou use o código de exemplo em Use seu modelo. Você pode usar o modelo para encontrar rótulos personalizados em imagens.

Selecionar número de unidades de inferência

Selecione um número maior de unidades de inferência para aumentar a taxa de transferência do modelo. Você será cobrado por cada unidade de inferência adicional usada.

1 inference unit ▼

### Use seu modelo

Nome de Recurso Amazon (ARN)  
arn:aws:rekognition:us-east-1:989944764342:project/Transportes/version/Transportes.2024-02-04T13.52.02/1707065521996

▶ Código da API

Para utilizar o modelo é necessário clicar em “Inciar” e confirmar quantas unidades de inferência deverá ser utilizado. Este valor está associado ao taxa de transferência da inferência. Sempre começamos com 1 e vamos ajustando conforme a demanda.

*Os custos de inferência é de 4 USD /hora, por isso quando não for utilizar interrompa e torne a ligar quando for utilizar. A ativação pode levar 30 minutos.*

# CLASSIFICAÇÃO DE IMAGENS: INFERÊNCIA

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
# criar cliente
client = session.client('rekognition', region_name=region)

model_arn = 'arn:aws:rekognition:us-east-1:989944764342:project/Transportes/version/Transportes.2024-02-04T13.52.02/1707065521996'

path = "aula-2-audio-imagem-transcricao/imagens/onibus.jpeg"

with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_custom_labels(
    Image={'Bytes': bytes_img}, MinConfidence=50, ProjectVersionArn=model_arn)

response

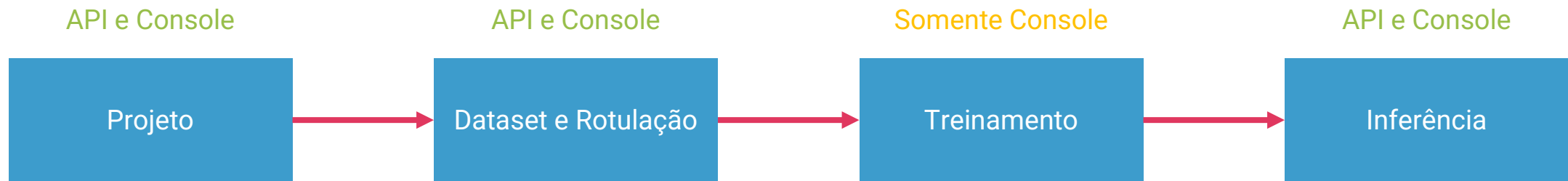
{'CustomLabels': [{'Name': 'Abóbora', 'Confidence': 98.55899810791016}],
 'ResponseMetadata': {'RequestId': '1901e84d-dcb5-4205-bb2c-9b9fd9201167',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '1901e84d-dcb5-4205-bb2c-9b9fd9201167',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '69',
   'date': 'Fri, 02 May 2025 01:03:25 GMT'},
  'RetryAttempts': 0}}
```



# ETAPAS E AÇÕES PARA CRIAÇÃO DE MODELO

Nem todas as etapas podem ser criadas programaticamente.

A criação do projeto, do dataset e da anotação são acionáveis pelo SDK. O treinamento não. Já o endpoint é possível via SDK iniciar e interromper.



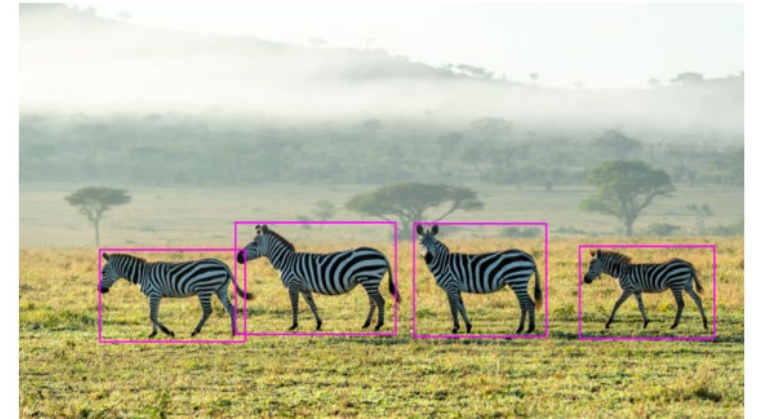
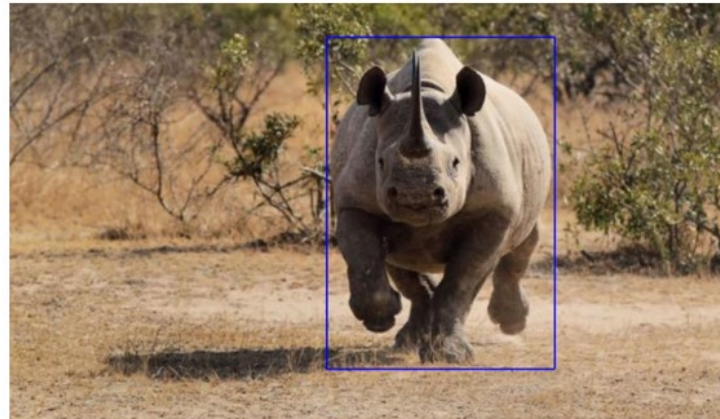
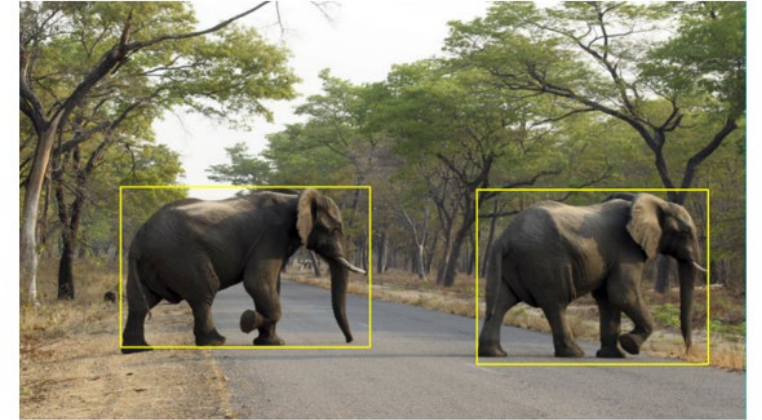
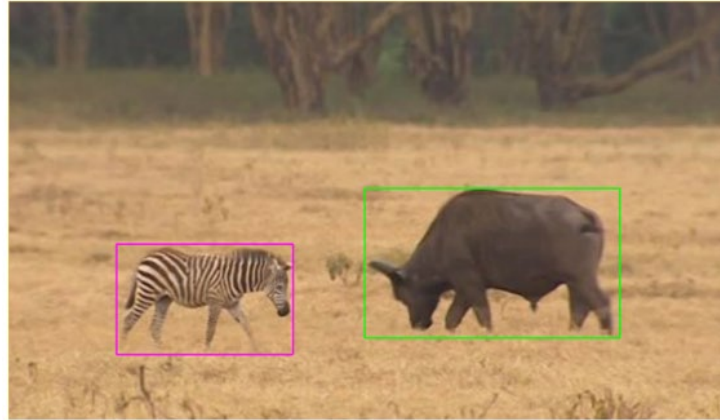
*O uso de SDK é mais recomendável no envio do dataset especialmente para rotulagem existente e não ter que realiza-las novamente de forma manual.*

# CLASSIFICADOR DE OBJETOS COM CUSTOM LABELS

Utilizaremos uma base de imagens com diferentes animais selvagens.

Por se tratar de um classificador de objetos, teremos além das imagens as regiões delimitadoras para cada classe.

Fonte [Kaggle](#) também disponível neste [repositório](#).





# CLASSIFICAÇÃO DE OBJETOS: DATASET

O processo de envio de imagens e rotulagem é o mesmo. A única diferença é que agora vamos utilizar a rotulagem por “Caixas delimitadoras”, que irá requerer anotações baseadas em um retângulo delimitador.


Imagens (1/15)

Atribuir rótulos em nível de imagem

Caixas delimitadoras


< 1 2 ... >

348.jpg ☒



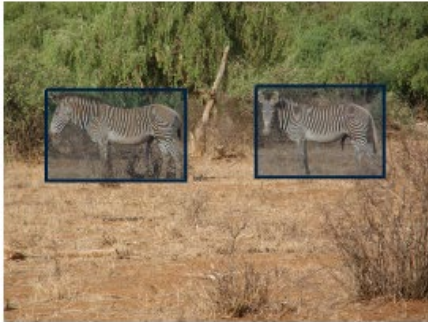
☐ zebra ✕

349.jpg ☐



☐ zebra ✕

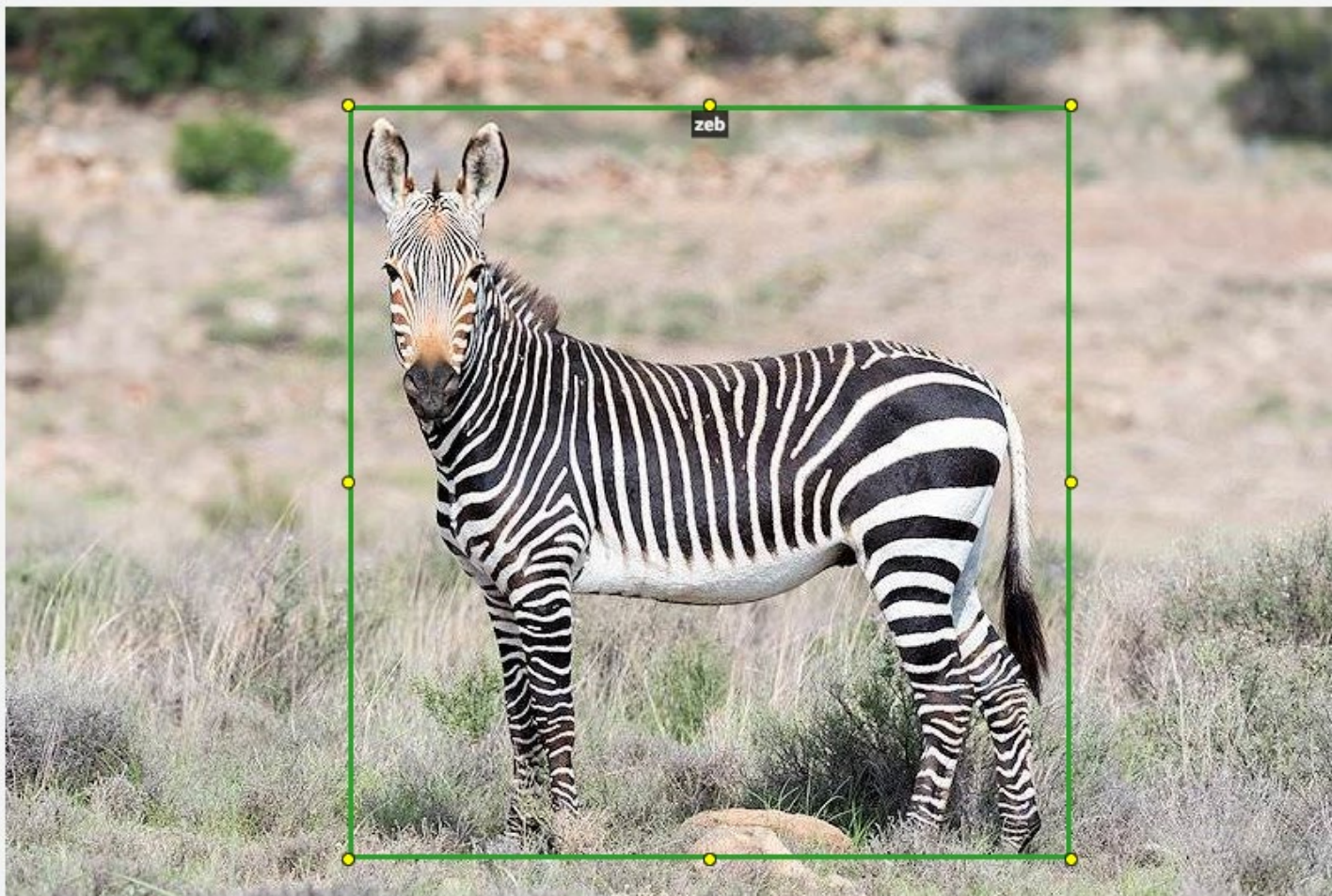
350.jpg ☐



☐ zebra ✕



# CLASSIFICAÇÃO DE OBJETOS: **ROTULAGEM**



Cada objeto precisa estar dentro de uma caixa delimitadora.

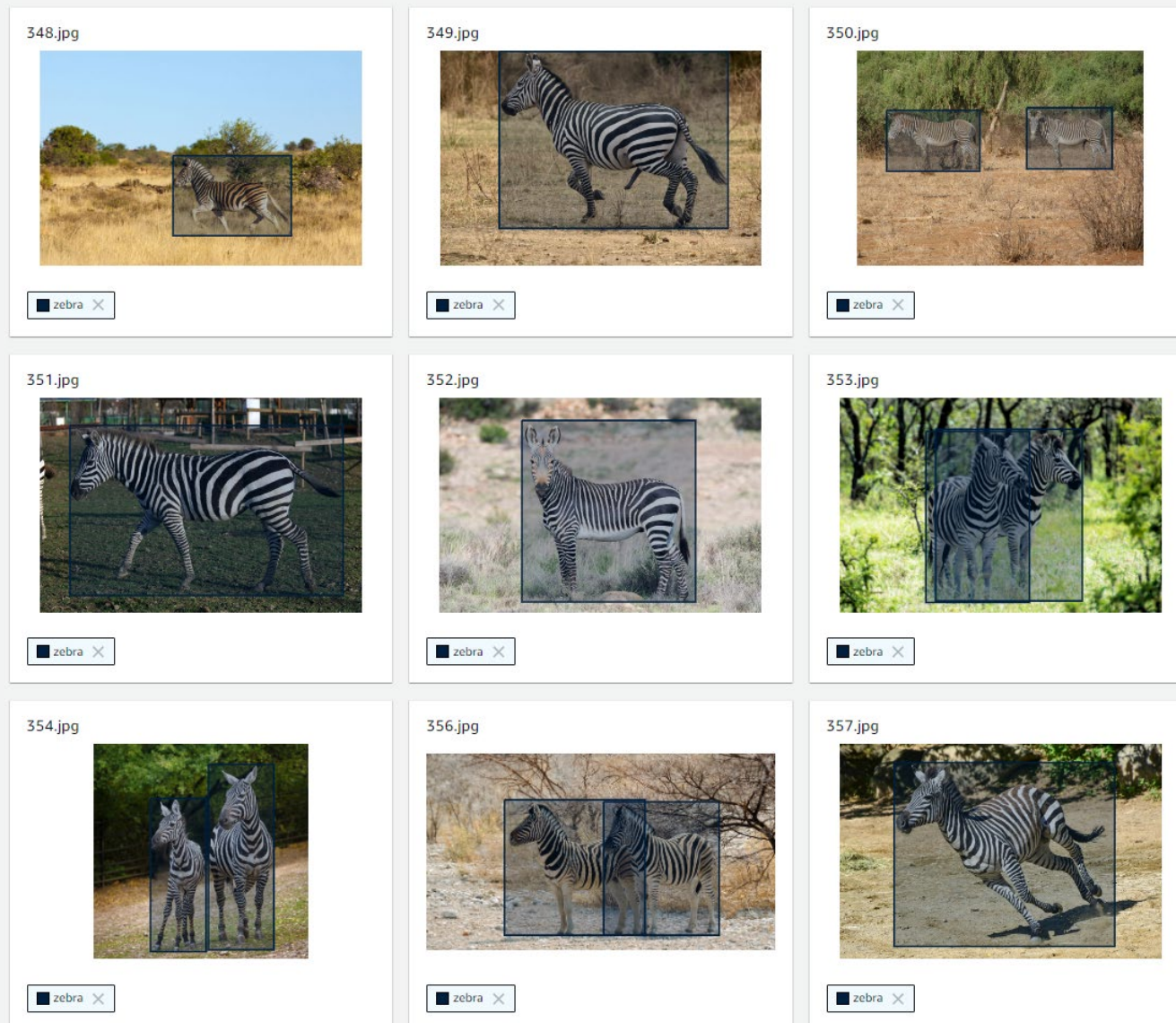
Se houver 2 objetos, terão 2 caixas delimitadoras.

Não há problema se houver sobreposições.

# CLASSIFICAÇÃO DE OBJETOS: **ROTULAGEM**

Ao final do processo, esperamos ter todas as imagens rotuladas e com as caixas delimitadoras em todas elas.

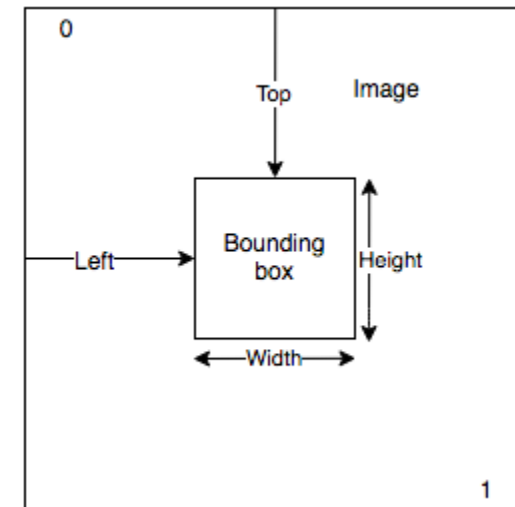
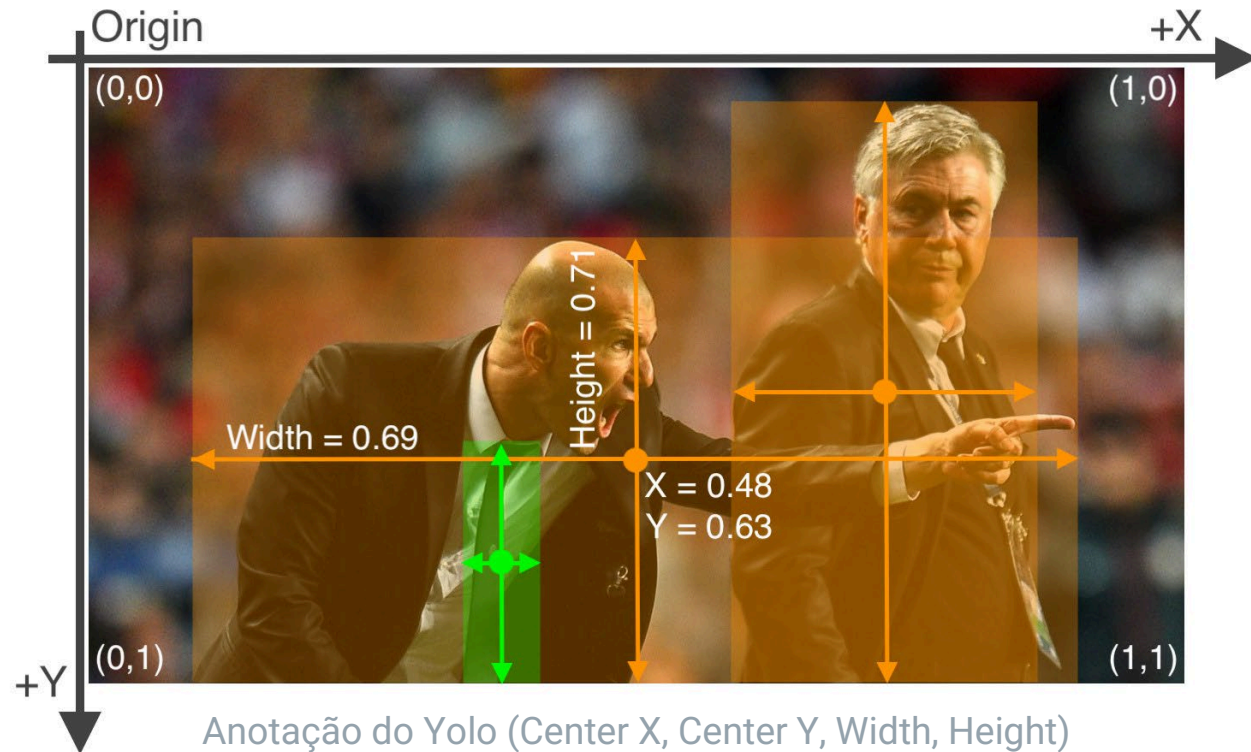
Os passos de treinamento, verificação de métricas e inferência seguem o mesmo do classificador de imagens.





# CLASSIFICAÇÃO DE OBJETOS: **ROTULAGEM POR SDK**

A anotação do conjunto de dados utiliza um padrão conhecido por Yolo. Cada anotação pode ter um padrão diferente. O Rekognition precisa de uma anotação com dados absolutos, ou seja, comprimento, altura e posicionamento da imagem referente aos eixos x e y.



# CLASSIFICAÇÃO DE OBJETOS: ROTULAGEM POR SDK

A conversão de um tipo de anotação para outra necessita do tamanho da imagem, pois assim se calcula os valores absolutos a partir dos relativos.

```
def convert_to_rekognition_format(box, image_width, image_height, class_id):
    center_x, center_y, width, height = box
    abs_width = int(width * image_width)
    abs_height = int(height * image_height)
    abs_x = int(center_x * image_width - (abs_width / 2))
    abs_y = int(center_y * image_height - (abs_height / 2))
    return {
        'width': abs_width,
        'height': abs_height,
        'left': abs_x,
        'top': abs_y,
        'class_id': class_id
    }
```

*O parâmetro box compõe os dados no formato Yolo. Apesar deste formato também trazer o id da classe, optamos por criar nosso próprio identificador.*

*Se decidirmos fazer a anotação manual não é necessário os passos programáticos, mas levará muito mais tempo e não é recomendado quando já tem a anotação pronta.*

# CLASSIFICAÇÃO DE OBJETOS: **MANIFESTO**

```
{
  "source-ref":"s3://bucket/image",
  "BB":{
    "annotations":[
      {"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
      {"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
      {"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
      {"left":2634,"top":1051,"width":673,"height":338,"class_id":3}
    ],
    "image_size":[
      {"width":4000,"height":2667,"depth":3}
    ]
  },
  "BB-metadata":{
    "job-name":"labeling-job/BB",
    "class-map":{
      "0":"comparator",
      "1":"pot_resistor",
      "2":"ir_phototransistor",
      "3":"ir_led"},
    "human-annotated":"yes",
    "objects":[
      {"confidence":1},
      {"confidence":1},
      {"confidence":1},
      {"confidence":1}
    ],
    "creation-date":"2021-06-22T10:11:18.006Z",
    "type":"groundtruth/object-detection"
  }
}
```

*Este é o arquivo de manifesto de exemplo para cada imagem. Nele contém as informações de anotação manual que precisam ser repassadas para ser reconhecidas no conjunto de dados.*

*As imagens precisam estar em um bucket do S3 obrigatoriamente.*

# CLASSIFICAÇÃO DE OBJETOS: ROTULAGEM POR SDK

Vamos separar uma lista de 20 imagens para o treinamento.

```
image_list = [  
    "dataset-animals/images/buffalo/001.jpg", "dataset-animals/images/buffalo/002.jpg",  
    "dataset-animals/images/buffalo/003.jpg", "dataset-animals/images/buffalo/004.jpg",  
    "dataset-animals/images/buffalo/005.jpg", "dataset-animals/images/buffalo/006.jpg",  
    "dataset-animals/images/buffalo/007.jpg", "dataset-animals/images/buffalo/008.jpg",  
    "dataset-animals/images/buffalo/009.jpg", "dataset-animals/images/buffalo/010.jpg",  
    "dataset-animals/images/buffalo/011.jpg", "dataset-animals/images/buffalo/012.jpg",  
    "dataset-animals/images/buffalo/013.jpg", "dataset-animals/images/buffalo/014.jpg",  
    "dataset-animals/images/buffalo/015.jpg", "dataset-animals/images/buffalo/016.jpg",  
    "dataset-animals/images/buffalo/017.jpg", "dataset-animals/images/buffalo/018.jpg",  
    "dataset-animals/images/buffalo/019.jpg", "dataset-animals/images/buffalo/020.jpg"  
]
```

*Neste dataset, cada nome de imagem possui o mesmo nome para a anotação num arquivo txt. Portanto vamos construir uma função para obter a anotação dado o caminho da imagem.*

```
def get_annotation(path):  
    annotation = path.replace(".jpg", ".txt")  
    with open(annotation, 'r') as arquivo:  
        conteudo = arquivo.read()  
  
    return conteudo
```



# NOVO PROJETO E DATASET **POR SDK**

Vamos criar o projeto e o conjunto de dados programaticamente.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
# criar cliente
client = session.client('rekognition', region_name=region)

project_name = 'Buffalo'

response = client.create_project(ProjectName=project_name)
project_arn = response['ProjectArn']

dataset_name = 'Buffalo_Images'
response = client.create_dataset(ProjectArn=project_arn, DatasetType='TRAIN')

dataset_arn = response['DatasetArn']
```

```
with open('arquivo.json', 'w') as arquivo:
    for image in image_list:

        object_image = {}
        bounding_boxes = []
        objects = []

        object_image["source-ref"] = "s3://fiap-cognitive-platforms/datasets/animals/buffalo/"+image.split("/")[3]

        imagem = mpimg.imread(image)
        image_height, image_width, _ = imagem.shape
        annotations = get_annotation(image)

        for annotation in annotations.split("\n"):
            box = float(annotation.split(' ')[1]), float(annotation.split(' ')[2]), float(annotation.split(' ')[3]),
float(annotation.split(' ')[4])
            new_format = convert_to_rekognition_format(box, image_width, image_height, 1)
            bounding_boxes.append(new_format)
            objects.append({"confidence": 1})

        object_image["BB"] = {"annotations": bounding_boxes, "image_size": [{"width":image_width, "height":image_height,
"depth":3}]}
        object_image["BB-metadata"] = {"job-name":"labeling-job/BB", "class-map":{"1":"buffalo"}, "human-annotated": "yes",
"objects": objects, "creation-date": "2024-02-04T10:11:18.006Z", "type": "groundtruth/object-detection"}

        json_string = json.dumps(object_image)
        arquivo.write(json_string + "\n")
```

*Agora vamos consolidar no arquivo de manifesto do modelo cada imagem com as respectivas anotações.*

# CARGA DE DADOS E MANIFESTO **POR SDK**

Vamos preparar o payload, adicionando o “GroundTruth” em seguida enviando os arquivos no formato string (Base64).

```
with open('arquivo.json', 'r') as arquivo:
    conteudo_json = arquivo.read()

json_string = json.dumps(conteudo_json)
bytes_data = json.loads(json_string)

object_images = { "GroundTruth" : bytes_data }
```

O próximo passo é enviar o payload com o manifesto. Como são muitas imagens enviadas de uma vez, há um pequeno delay para o processamento. Portanto verifique o comando “list\_dataset\_entries” até que os dados apareçam, indicando que eles foram carregados com sucesso.

```
response = client.update_dataset_entries(DatasetArn = dataset_arn, Changes = object_images)
response = client.list_dataset_entries(DatasetArn=dataset_arn, MaxResults=100)
response
```

```
04T10:11:18.006Z", "type": "groundtruth/object-detection", "cl-metadata": {"is_labeled": true}}, { "source-
ref": "s3://fiap-cognitive-
platforms/datasets/animals/buffalo/009.jpg", "BB": {"annotations": [{"width": 301, "height": 198, "left": 75, "top": 88, "class_id
": 1}], "image_size": [{"width": 450, "height": 353, "depth": 3}], "BB-metadata": {"job-name": "labeling-job/BB", "class-
map": {"1": "buffalo"}, "human-annotated": "yes", "objects": [{"confidence": 1}], "creation-date": "2024-02-
04T10:11:18.006Z", "type": "groundtruth/object-detection", "cl-metadata": {"is_labeled": true}}, {
```

# REVISÃO DO CONJUNTO DE DADOS

001.jpg



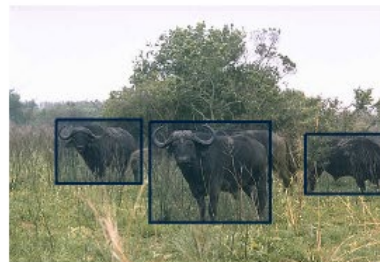
■ buffalo ✕

002.jpg



■ buffalo ✕

005.jpg



■ buffalo ✕

006.jpg



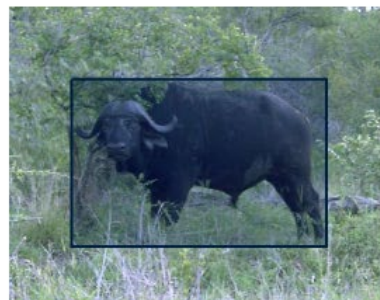
■ buffalo ✕

007.jpg



■ buffalo ✕

009.jpg



■ buffalo ✕

Após o processo de criação e carga das referências de arquivos (incluindo o manifesto), eles devem aparecer no Console para revisão.

Conforme exemplo anterior, o fluxo segue normalmente via console para treinamento, análise de métricas e inferência.

# CLASSIFICAÇÃO DE OBJETOS: **INFERÊNCIA**

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
# criar cliente
client = session.client('rekognition', region_name=region)

model_arn='arn:aws:rekognition:us-east-1:989944764342:project/Animais/version/Animais.2024-02-04T21.13.57/1707092036915'

path = "aula-2-audio-imagem-transcricao/imagens/zebra.jpg"

with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_custom_labels(
    Image={'Bytes': bytes_img}, MinConfidence=50, ProjectVersionArn=model_arn)

response

{'CustomLabels': [{'Name': 'zebra',
  'Confidence': 89.15499877929688,
  'Geometry': {'BoundingBox': {'Width': 0.8374999761581421,
    'Height': 0.9529100060462952,
    'Left': 0.14809000492095947,
    'Top': 0.041600000113248825}}}],
  ...
```



# CLASSIFICAÇÃO DE OBJETOS: INFERÊNCIA

Exemplo de inferência. Note que podemos especificar confiança mínima e avaliar o valor vindo do modelo.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
# criar cliente
client = session.client('rekognition', region_name=region)

model_arn='arn:aws:rekognition:us-east-1:989944764342:project/Buffalo/version/Buffalo.2024-02-05T20.55.12/1707177312926'

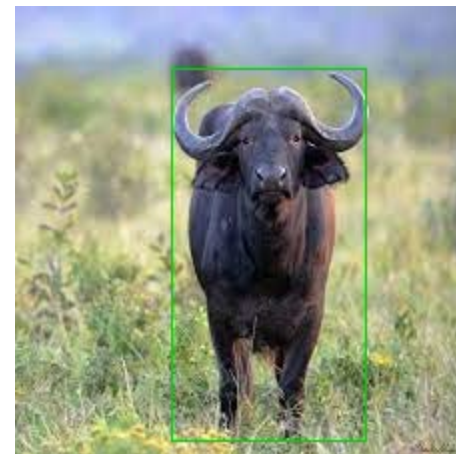
path = "aula-2-audio-imagem-transcricao/imagens/bufalo.jpg"

with open(path, "rb") as file:
    img = file.read()
    bytes_img = bytearray(img)

response = client.detect_custom_labels(
    Image={'Bytes': bytes_img}, MinConfidence=50, ProjectVersionArn=model_arn)

response

{'CustomLabels': [{'Name': 'buffalo',
  'Confidence': 94.17300415039062,
  'Geometry': {'BoundingBox': {'Width': 0.42895999550819397,
    'Height': 0.8270099759101868,
    'Left': 0.3497900068759918,
    'Top': 0.14168000221252441}}}],
  ...
```





# DESAFIO 1

Crie um modelo para classificar diferentes tipos de plantações (considere no mínimo 3 classes). Busque alcançar valores de precisão, *recall* e *average precision* maiores do que 90%.

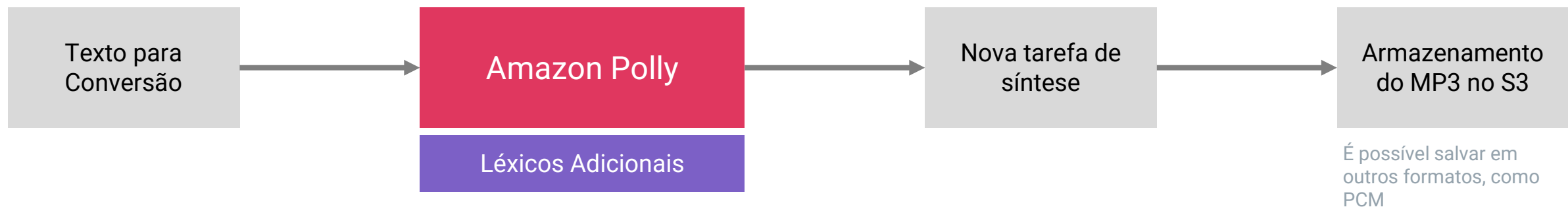
Utilize o dataset disponibilizado neste repositório <https://github.com/michelpf/dataset-agricultural-crops>.

# AMAZON POLLY: TIPOS DE EXECUÇÃO

Serviço para converter texto em áudio.

Suporta diversos idiomas, dentro os quais o português do Brasil.

É compatível com a linguagem de marcação SSML.



É possível customizar as vozes de fala e também incluir léxicos para ser utilizado nas tarefas de sintetização. Exemplo comum é o uso de léxicos para lidar com siglas e ao ser sintetizado ser falado o que significa a sigla, por exemplo: ao citar o BNDES, o áudio poderá dizer Banco Nacional de Desenvolvimento ou invés das letras B, N, D, E S.

Outra aplicação dos léxicos é o uso de grafema (representação das letras) e fonemas (representação do som) específicos de cada idioma ou região.

|   |                                    |
|---|------------------------------------|
| Adicionando uma pausa   | <break>                            |
| Enfatizando palavras  | <emphasis>                         |
| Especificando outro idioma para palavras específicas          | <lang>                             |
| Colocando uma marcação personalizada em seu texto             | <mark>                             |
| Adicionando uma pausa entre parágrafos                        | <p>                                |
| Usando pronúncia fonética                                     | <phoneme>                          |
| Controlando volume, taxa de fala e tom                        | <prosody>                          |
| Definindo uma duração máxima para a síntese da fala           | <prosody amazon:max-duration>      |
| Adicionando uma pausa entre frases                            | <s>                                |
| Controlando como tipos especiais de palavras são pronunciados | <say-as>                           |
| Identificando texto aprimorado com SSML                       | <speak>                            |
| Pronunciando siglas e abreviações                             | <sub>                              |
| Melhorando a pronúncia ao especificar partes do discurso      | <w>                                |
| Adicionando o som da respiração                               | <amazon:auto-breaths>              |
| Estilo de fala de apresentador de notícias                    | <amazon:domain name="news">        |
| Adicionando compressão de faixa dinâmica                      | <amazon:effect name="drc">         |
| Falando suavemente  | <amazon:effect phonation="soft">   |
| Controlando o timbre  | <amazon:effect vocal-tract-length> |
| Sussurrando   | <amazon: effect name="whispered">  |

# AMAZON POLLY: TIPOS DE EXECUÇÃO

Exemplo de utilização, diretamente no Console.

A tag “speak” indica o trecho de fala.

A tag “break” indica a pausa na fala e a duração.

A tag “sub” indica a narração com o significado da sigla ao invés a fala das letras individuais.

A tag “amazon” refere-se a efeitos especiais da Amazon, o valor “whispered” indica que a voz falada terá um volume de menor intensidade, como um cochicho.

```
<speak>
```

```
Boa a noite a todos.<break time="1s"/>Hoje, a aula de Plataformas Cognitivas na
```

```
<sub alias="fiap">FIAP</sub> será realizada com o professor Michel.
```

```
Espero que tenham uma boa aula!
```

```
<amazon:effect name="whispered">"Ah, é uma das minhas disciplinas favoritas!"</amazon:effect>
```

```
</speak>
```



# AMAZON POLLY: TIPOS DE EXECUÇÃO

Execução por API, padrão.

```
# abrir sessão
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)

# criar cliente
client = session.client("polly", region_name=region)

# iniciat requisição para atividade de geração de áudio
response = client.start_speech_synthesis_task(
    Engine='neural',
    LanguageCode='pt-BR',
    OutputFormat='mp3',
    OutputS3BucketName='fiap-cognitive-platforms',
    OutputS3KeyPrefix='text-to-audio/',
    Text='Olá, isso é um teste de fala do Polly.',
    TextType='text',
    VoiceId='Camila'
)
```

Resultado armazenado no S3

```
{'ResponseMetadata': {'RequestId': 'ca50a8b7-5d2e-4062-b09a-eca9835ae3c3', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'ca50a8b7-5d2e-4062-b09a-eca9835ae3c3', 'content-type': 'application/json', 'content-length': '527', 'date': 'Sat, 05 Aug 2023 20:39:30 GMT'}, 'RetryAttempts': 0}, 'SynthesisTask': {'Engine': 'neural', 'TaskId': '3ecf5549-0d86-45e4-9f15-e5e3188d04af', 'TaskStatus': 'scheduled', 'OutputUri': 'https://s3.us-east-1.amazonaws.com/fiap-cognitive-platforms/text-to-audio/.3ecf5549-0d86-45e4-9f15-e5e3188d04af.mp3', 'CreationTime': datetime.datetime(2023, 8, 5, 20, 39, 31, 287000, tzinfo=tzlocal()), 'RequestCharacters': 38, 'OutputFormat': 'mp3', 'TextType': 'text', 'VoiceId': 'Camila', 'LanguageCode': 'pt-BR'}}
```



# AMAZON POLLY: TIPOS DE EXECUÇÃO

Execução por API, utilizando SSML.

```
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
client = session.client("polly", region_name=region)
texto_ssml = """
<say>
  Boa a noite a todos.<break time="1s"/>Hoje, a aula de Plataformas Cognitivas na
  <sub alias="fiap">FIAP</sub> será realizada com o professor Michel.
  Espero que tenham uma boa aula!
  <amazon:effect name="whispered">"Ah, é uma das minhas disciplinas favoritas!"</amazon:effect>
</say>
"""

response = client.start_speech_synthesis_task(
    Engine='neural',
    LanguageCode='pt-BR',
    OutputFormat='mp3',
    OutputS3BucketName='fiap-cognitive-platforms',
    OutputS3KeyPrefix='text-to-audio/',
    Text=texto_ssml,
    TextType='ssml',
    VoiceId='Camila'
)
```

Resultado armazenado no S3

```
{'ResponseMetadata': {'RequestId': '9caee327-9c6d-4500-b86b-8c8efd12a3e7', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '9caee327-9c6d-4500-b86b-8c8efd12a3e7', 'content-type': 'application/json', 'content-length': '528', 'date': 'Mon, 07 Aug 2023 00:56:20 GMT'}, 'RetryAttempts': 0}, 'SynthesisTask': {'Engine': 'neural', 'TaskId': '61eedc2b-425a-4c67-8d8d-25ff6ff0765b', 'TaskStatus': 'scheduled', 'OutputUri': 'https://s3.us-east-1.amazonaws.com/fiap-cognitive-platforms/text-to-audio/.61eedc2b-425a-4c67-8d8d-25ff6ff0765b.mp3', 'CreationTime': datetime.datetime(2023, 8, 7, 0, 56, 20, 744000, tzinfo=tzlocal()), 'RequestCharacters': 184, 'OutputFormat': 'mp3', 'TextType': 'ssml', 'VoiceId': 'Camila', 'LanguageCode': 'pt-BR'}}
```

# CLOUD API: **AMAZON TRANSCRIBE**

Serviço dedicado a transcrever voz em texto, a partir de arquivos MP3 (comporta outros formatos). Capaz de identificar automaticamente o idioma como também os narradores, quando há mais do que um.

Permite que seja criado modelo customizado a partir de textos e gravações específicas, apesar do modelo padrão ter excelentes resultados.



# AMAZON TRANSCRIBE: EXECUÇÃO DA API

```
session = boto3.Session(aws_access_key_id=ACCESS_ID, aws_secret_access_key= ACCESS_KEY)
client = session.client("transcribe", region_name=region)

response = client.start_transcription_job(
    TranscriptionJobName="PodcastDecodeTranscription",
    LanguageCode="pt-BR",
    MediaFormat="mp3",
    Media={
        "MediaFileUri": "s3://fiap-cognitive-platforms/audio/podcast-decode-fiap/decode-28.mp3",
    },
    OutputBucketName="fiap-cognitive-platforms",
    OutputKey="audio/podcast-decode-fiap/transcription/",
    Settings={
        'ShowSpeakerLabels': True,
        'MaxSpeakerLabels': 3,
        'ChannelIdentification': True,
    },
    JobExecutionSettings={
        "AllowDeferredExecution": True,
        "DataAccessRoleArn": "arn:aws:iam::989944764342:role/ComprehendJob"
    },
    Subtitles={
        'Formats': [
            'vtt','srt',
        ],
    }
)
```

Cada transcrição é um batch job que leva alguns minutos para a atividade, ou seja, é um processamento assíncrono.

Detalhes como idioma de transcrição e número de narradores ajudam na atividade e diminuem potenciais falhas no processo.

O `DataAccessRoleArn`, se executado do Colab (ou a partir de outro usuário), precisa conter as permissões citadas no começo da aula.

Podemos utilizar a mesma role utilizada anteriormente (“ComprehendJob”).

# AMAZON TRANSCRIBE: EXECUÇÃO DA API

```
response = client.get_transcription_job(  
    TranscriptionJobName="PodcastDecodeTranscription"  
)
```

response

```
{{'TranscriptionJob': {'TranscriptionJobName': 'PodcastDecodeTranscription',  
    'TranscriptionJobStatus': 'IN_PROGRESS',  
    'LanguageCode': 'pt-BR',  
    'MediaFormat': 'mp3',  
    'Media': {'MediaFileUri': 's3://fiap-cognitive-platforms/audio/podcast-decode-fiap/decode-28.mp3'},  
    'StartTime': datetime.datetime(2023, 8, 8, 0, 57, 1, 137000, tzinfo=tzlocal()),  
    'CreationTime': datetime.datetime(2023, 8, 8, 0, 57, 1, 105000, tzinfo=tzlocal()),  
    'Settings': {'ShowSpeakerLabels': True,  
    'MaxSpeakerLabels': 3,  
    'ChannelIdentification': True},  
    'Subtitles': {'Formats': ['vtt', 'srt']}},  
    'ResponseMetadata': {'RequestId': 'c670be98-0e65-482b-9269-d519c8b32159',  
    'HTTPStatusCode': 200,  
    'HTTPHeaders': {'x-amzn-requestid': 'c670be98-0e65-482b-9269-d519c8b32159',  
    'content-type': 'application/x-amz-json-1.1',  
    'content-length': '440',  
    'date': 'Tue, 08 Aug 2023 00:57:00 GMT'},  
    'RetryAttempts': 0}}
```

A resposta da requisição da transcrição retornará o status de envio do job.

Utilizaremos o nome do job (*TranscriptionJobName*) para consultar o status da transcrição.

Quando o status for *COMPLETED* podemos obter o resultado no bucket informado como saída.

# 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:

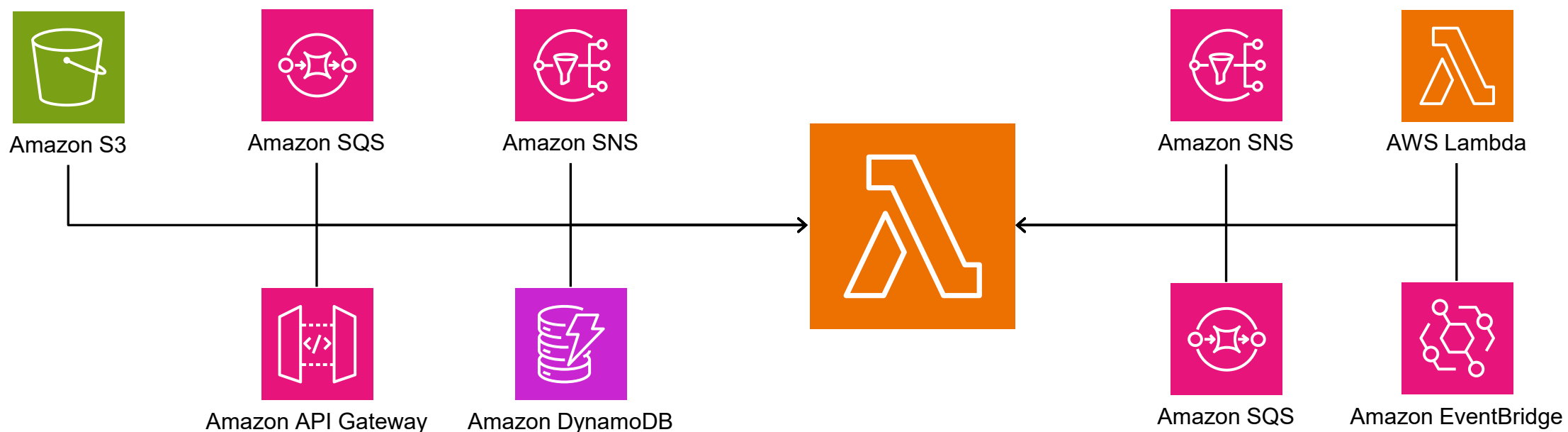
- **Serverless:** Você não precisa provisionar nem gerenciar servidores. A AWS cuida da infraestrutura, permitindo que o desenvolvedor se concentre apenas no código.
- **Escalabilidade Automática:** As funções Lambda são dimensionadas automaticamente, o que significa que elas podem lidar com qualquer 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**: 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
cp 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
```



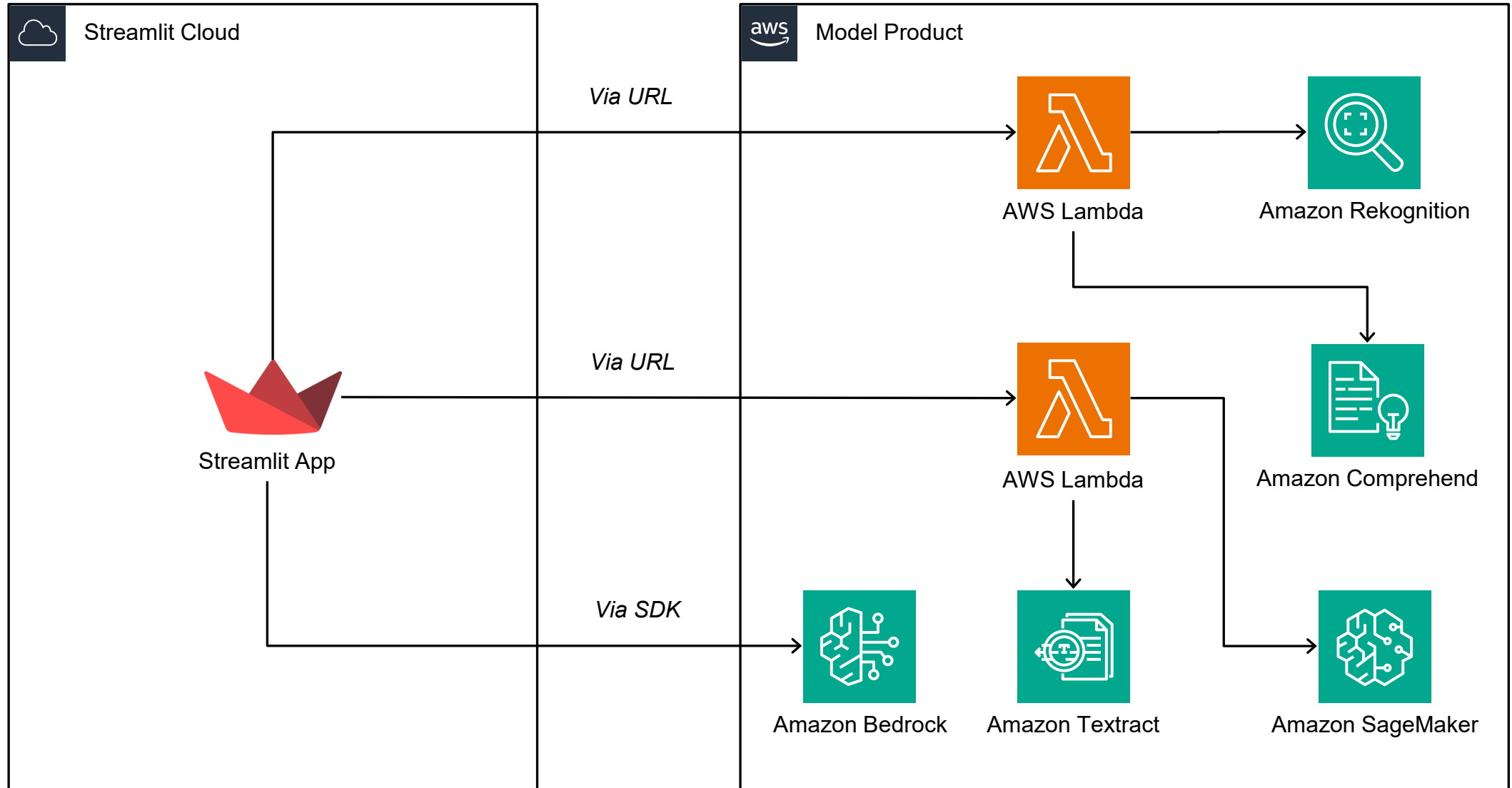
# Streamlit



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.

# SYSTEM DESIGN





# FACE DETECTOR

Esta aplicação desenvolvida no Streamlit acessa um endpoint externo (Lambda Function) enviando uma imagem de uma câmera para validações.

É validado expressões significativas como raiva, alegria, etc. O intuito é ter uma foto neutra para ser utilizado no documento de validação de instituições financeiras.

Foi utilizado o Amazon Rekognition para as análises faciais acessada pelo backend e exibidas na camada de interação Streamlit (como frontend).

## Cognitive Platforms

### Detecção de Foto para Documentos

Este aplicativo visa detectar face para emissão de documentos, validando as seguintes condições e características:

- Existência de face
- Verificação de neutralidade via análise sentimental

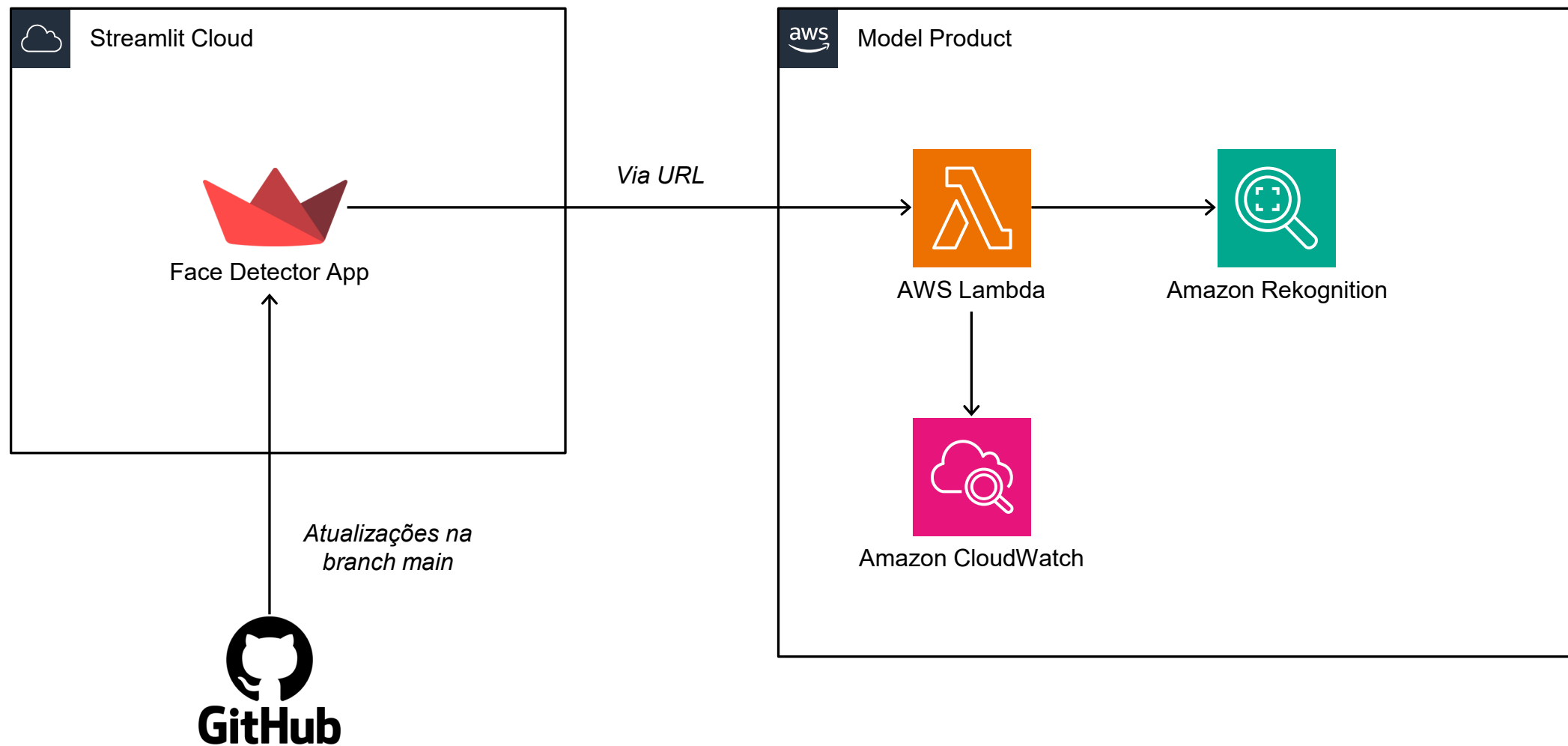
Estando as duas condições validadas, a foto está aprovada para emissão de documento.

Tire sua foto. Procure centralizar seu rosto.



Take Photo

# FACE DETECTOR: **SYSTEM DESIGN**




<https://github.com/michelpf/fiap-ds-face-detector-app>

<https://github.com/michelpf/fiap-ds-face-detector-backend>

# APP DE INTERAÇÃO: FRONTEND

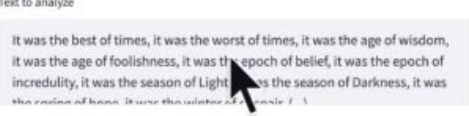
Para coletarmos a imagem, vamos utilizar componentes de entrada do Streamlit. Eles ficam na documentação como “Input Widgets”, nesta [documentação](#).

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 number")
```




**Text area**  
Display a multi-line text input widget.

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



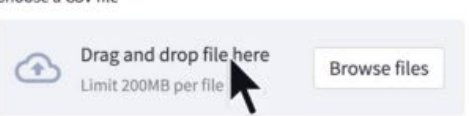
**Date input**  
Display a date input widget.

```
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 picture")
```

# APP DE INTERAÇÃO: **BACKEND**

Nosso backend será em Lambda Function, com acesso por endpoint externo conhecido como Lambda URL.

A função precisa ser o mais simples o possível, para que o processamento seja rápido.

```
def lambda_handler(event, context):  
    # Verificando se o request veio de URL  
    if "headers" in event:  
        event = event["body"]  
  
        # Remove as barras invertidas de escape e caracteres  
        adicionais  
        event = event.replace("\\", "")  
        event = event.strip('\"')  
  
        # Converte a string para um dicionário Python  
        event = json.loads(event)  
  
    print(event)
```

*Toda a função Lambda precisa ter um handler (função) com 2 parâmetros obrigatórios.*

*Utilizamos com mais frequência o event. Seja por requisição externa (URL) ou interna.*

*Por isso, precisamos validar a origem. Se há header, é externa e precisamos extrair o body. Do contrário, o body é o próprio event.*

# APP DE INTERAÇÃO: **BACKEND**

```
# Verifica se o evento contém a imagem em base64
if 'image_base64' in event:
    # Decodifica a imagem base64
    image_base64 = event['image_base64']
    image_bytes = base64.b64decode(image_base64)

    # Define o caminho e o nome do arquivo temporário
    temp_image_path = '/tmp/temp_image.png'

    # Salva a imagem em um arquivo temporário
    with open(temp_image_path, 'wb') as f:
        f.write(image_bytes)

    # Inicializa o cliente para a AWS Rekognition
    rekognition_client = boto3.client('rekognition')

    # Chama a API de análise facial da AWS Rekognition
    with open(temp_image_path, 'rb') as f:
        response = rekognition_client.detect_faces(
            Image={
                'Bytes': f.read()
            },
            Attributes=['ALL']
        )
```

*Validamos se no payload há uma chave "image\_base64". Iremos receber a imagem como string neste padrão.*

*Decodificamos a imagem e armazenamos numa pasta temporária.*

*Abrimos o arquivo em bytes (binário) e chamamos o serviço do Rekognition.*

*Como o serviço já possui as permissões, não precisamos incluir as chaves de acesso como fazemos no laboratório (Colab).*

# APP DE INTERAÇÃO: **BACKEND**

```
# Processa a resposta da API
if 'FaceDetails' in response:
    face_details = response['FaceDetails']

    if len(face_details) > 1:

        return {
            'statusCode': 400,
            'body': "Há mais de uma face neste rosto."
        }

# Realizar validação se alguma emoção tem maior que 70% de confiança

emotions = response['FaceDetails'][0]["Emotions"]
emotion_max = obter_maior_confianca(emotions)

if emotion_max["Confidence"] > 0.7 and emotion_max["Type"] != "CALM":
    return {
        'statusCode': 400,
        'body': "Rosto com emoção predominante (" + emotions_translate[emotion_max["Type"]] + ")."
    }

return {
    'statusCode': 200,
    'body': {"boundingBox": response['FaceDetails'][0]["BoundingBox"], "reason": "Face validada com
sucesso."}
}
```

*Precisamos fazer a validação da emoção com maior intensidade. Se for calmo ignoramos, do contrário, precisamos informar que não é permitido. Também não é permitido mais de uma face.  
Estando tudo certo, enviamos os dados do BoundingBox.*



```
else:
    return {
        'statusCode': 400,
        'body': 'Não foram encontrados detalhes faciais na imagem.'
    }
else:
    return {
        'statusCode': 400,
        'body': 'Por favor, forneça a imagem em base64.'
    }
```

*Se não for encontrado a chave da imagem em base64 ou se nenhum rosto for encontrado, retornamos com erro.*

```
emotions_translate = {
    "ANGRY": "nervoso",
    "DISGUSTED": "enjoado",
    "FEAR": "medo",
    "CALM": "calmo",
    "SAD": "triste",
    "SURPRISED": "surpreso",
    "CONFUSED": "confuso",
    "HAPPY": "feliz"
}
```

*Dicionário para tradução das emoções para português e ser incluída nas mensagens.*

# APP DE INTERAÇÃO: **BACKEND**

```
def obter_maior_confianca(lista):  
    # inicializa a maior confiança e o dicionário com maior confiança como None  
    maior_confianca = None  
    dicionario_maior_confianca = None  
  
    # percorre cada dicionário na lista  
    for dicionario in lista:  
        confianca = dicionario["Confidence"]  
  
        # verifica se é a maior confiança encontrada até o momento  
        if maior_confianca is None or confianca > maior_confianca:  
            maior_confianca = confianca  
            dicionario_maior_confianca = dicionario  
  
    return dicionario_maior_confianca
```

*Função auxiliar para determinar a emoção com maior intensidade de confiança. E assim estabelecer como a marcante de uma determinada imagem.*

# 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”.



Please wait...

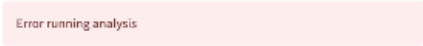




## Spinner

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”.

|  |   |   |
|--|---|---|
|  |  |  |
| <b>Error box</b><br>Display error message.   | <b>Warning box</b><br>Display warning message.                                      | <b>Info box</b><br>Display an informational message.                                |
| <pre>st.error("We encountered an error")</pre>                                     | <pre>st.warning("Unable to fetch image. Ski")</pre>                                 | <pre>st.info("Dataset is updated every day")</pre>                                  |
|  |  |   |
| <b>Success box</b><br>Display a success message.                                   | <b>Exception output</b><br>Display an exception.                                    |   |
| <pre>st.success("Match found!")</pre>  | <pre>e = RuntimeError("This is an exception")<br/>st.exception(e)</pre>             |   |

# 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://u2dspwgdq6dwi6qj26l5y5e0hcqzq.lambda-url.us-east-1.on.aws"
```

# BIBLIOTECAS E **DEPENDÊNCIAS**

O projeto com Streamlit vai utilizar ela mesma e bibliotecas adicionais para outras manipulações. Em nosso caso precisaremos recortar a imagem com os dados do retângulo delimitador (região de interesse), neste caso utilizaremos o Pillow.

Além disso precisamos da biblioteca para acesso a APIs externas (Requests).

```
streamlit == 1.30.0  
requests == 2.31.0  
pillow == 9.5.0
```

*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.

```
"""  
# Cognitive Platforms  
  
## Detecção de Foto para Documentos  
  
Este aplicativo visa detectar face para emissão de documentos,  
validando as seguintes condições e características:  
  
* Existência de face  
* Verificação de neutralidade via análise sentimental  
  
Estando as duas condições validadas, a foto está aprovada para emissão de documento.  
"""
```



# 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.

```
foto = st.camera_input("Tire sua foto. Procure centralizar seu rosto.")
```

```
if foto:
```

```
    with st.spinner('Validando condições...'):
```

```
        bytes_data = foto.getvalue()
        img_base64 = base64.b64encode(bytes_data).decode('utf-8')
```

```
        payload = {
            "image_base64": img_base64
        }
```

```
        payload = json.dumps(payload)
```

```
        # URL da sua API Gateway
        endpoint = st.secrets["API-ENDPOINT"]
```

```
        response = requests.post(endpoint, json=payload)
```

*O Lambda não lida com arquivos binários diretamente, por isso vamos enviar como String utilizando Base64.*

*A URL está como Secret, tanto local quando em cloud.*

# 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.

```
# Verifique se a solicitação foi bem-sucedida
if response.status_code == 200:
    response_data = json.loads(response.text)

    imagem = Image.open(foto)

    largura_total, altura_total = imagem.size

    # Define as coordenadas do retângulo de corte com base nas medidas fornecidas
    left = int(response_data["boundingBox"]["Left"] * largura_total)
    top = int(response_data["boundingBox"]["Top"] * altura_total)
    width = int(response_data["boundingBox"]["Width"] * largura_total)
    height = int(response_data["boundingBox"]["Height"] * altura_total)

    # Recorta a região de interesse (ROI) da imagem
    regioao_recortada = imagem.crop((left, top, left+width, top+height))

    st.image(regiao_recortada)

    st.success('Foto validada com sucesso! Emissão autorizada.', icon="✅")
else:
    st.error(response.text, icon="🚨")
```

*Se o código for 200, tudo certo.  
Com os dados fornecidos do Bounding Box,  
podemos recortar a imagem no rosto.*

*Com a imagem  
recortada, vamos exibi-  
la como imagem.*

# PARA SABER **MAIS**

<https://cursos.alura.com.br/course/pandas-limpeza-tratamento-dados>, Tratamento de dados com Pandas

<https://cursos.alura.com.br/course/machine-learning-utilizando-chatgpt-assistente>, Utilizando ChatGPT como assistente em Machine Learning

<https://cursos.alura.com.br/course/streamlit-construindo-dashboard-interativo>, Curso de Streamlit para construção de dashboards

FIAP