

PYTHON BRASIL [15]

# SERVERLESS - ZERO TO HERO

By Giana de Almeida

A portrait of a woman with long dark hair, smiling, wearing a black jacket over a blue patterned top. The background is a solid light blue.

# ABOUT THE SPEAKER

**INTEGRATION ENGINEER**

Python community engager,  
Pyladies Floripa,

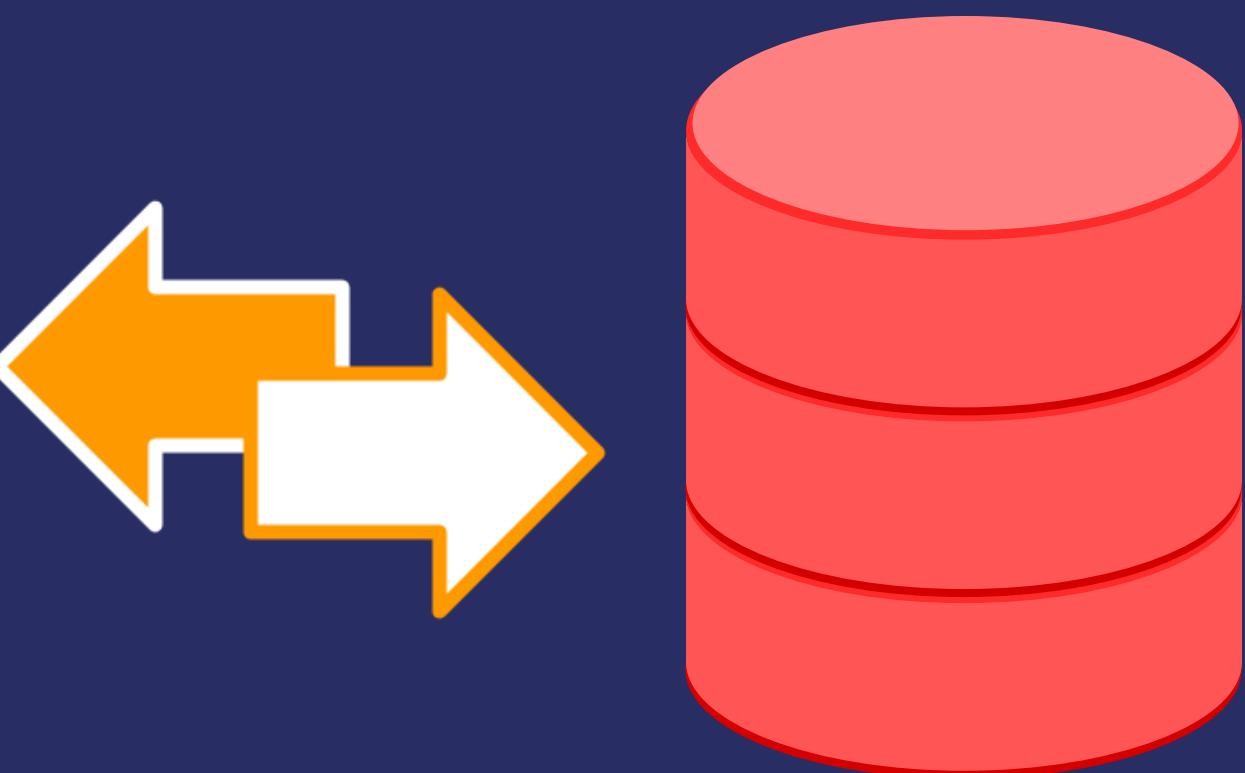
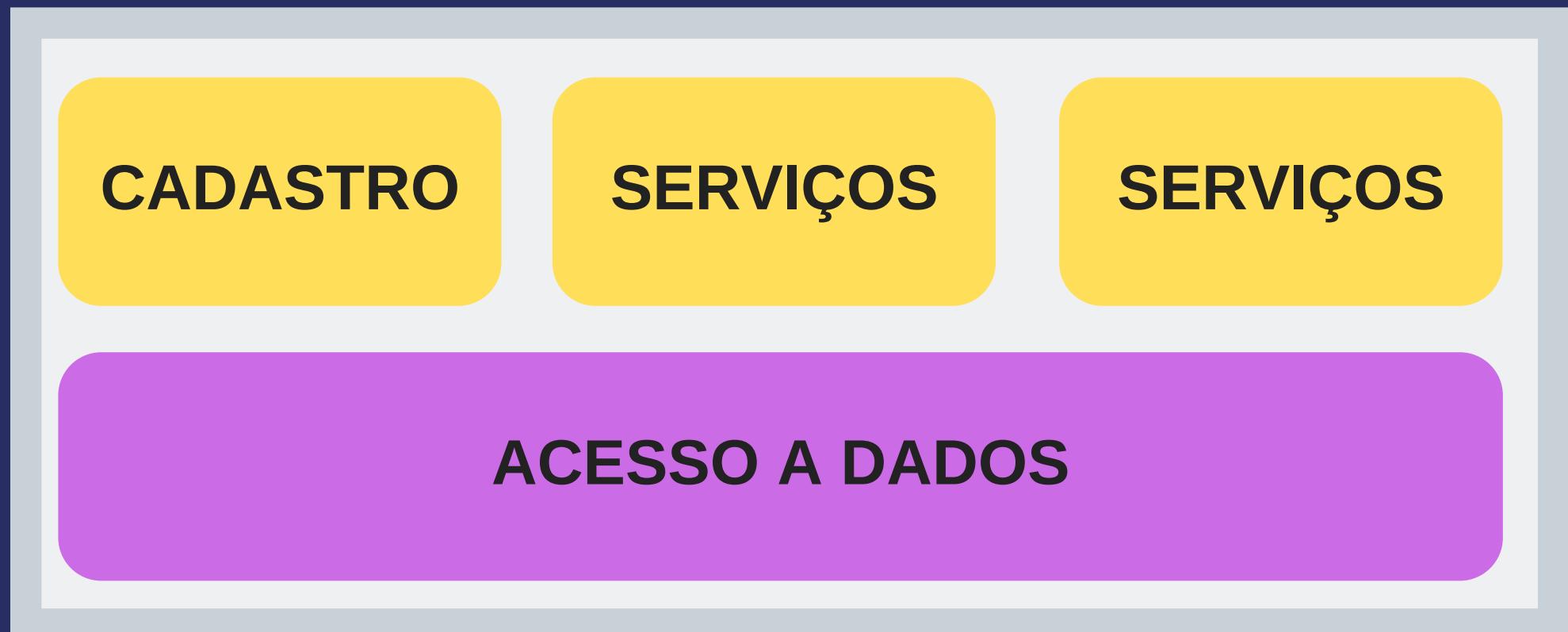
# About Leaf

The easiest way to connect  
agriculture data across platforms.



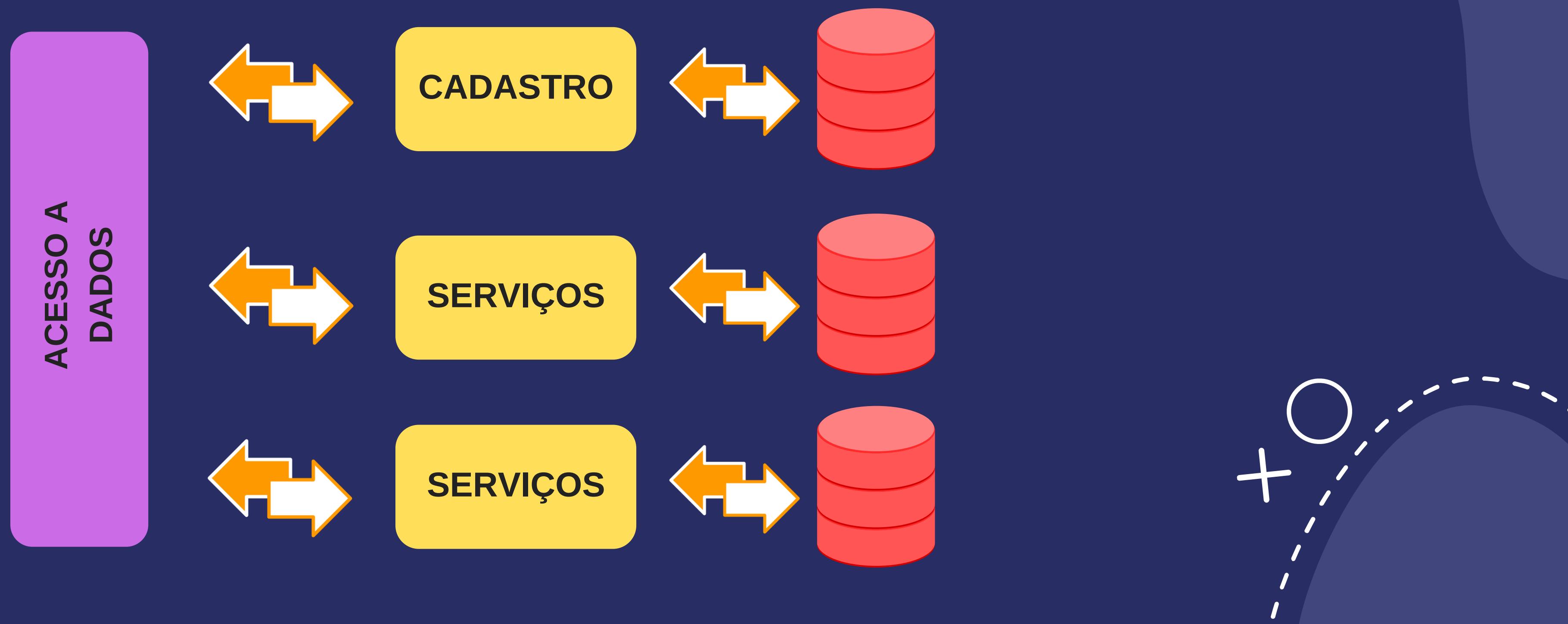
# PYTHON BRASIL [15]

## APLICAÇÃO CENTRALIZADA



# PYTHON BRASIL [15]

## APLICAÇÃO DESCENTRALIZADA



# serverless:

é um modelo de execução onde o provedor de cloud será o responsável por executar pedaços de código com recursos que irão ser alocados dinamicamente e cobrando apenas pelos recursos usados para executar aquele código em específico.





Fonte: Imagem de SOUZA. E.

## VANTAGENS

- Sem preocupações com infraestrutura, foco na aplicação
- Processamento em paralelo
- Baixo custo
- Códigos menores
- Escalável
- Facilidade de combinar recursos da Cloud
- Se uma função falha, não compromete a disponibilidade de outros recursos da aplicação

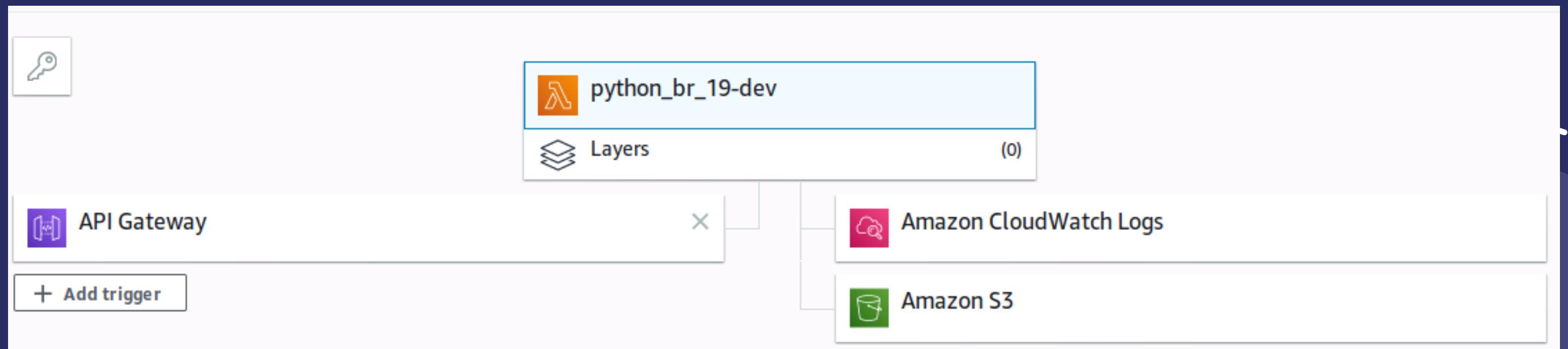
## LIMITAÇÕES

- Não permite processamentos longos, o ideal é dentro do tempo de 5 minutos
- Dificuldade de trocar de Cloud sem ter que fazer do zero, devido a frameworks e recursos, como proxy e armazenamento por exemplo
- Limitação nas configurações de infra, permitindo apenas a personalização da linguagem de programação, memória, time out e permissões



## CONHECENDO OS RECURSOS

- Funções lambda
- API Gateway
- S3
- CloudWatch



# PYTHON BRASIL [15]

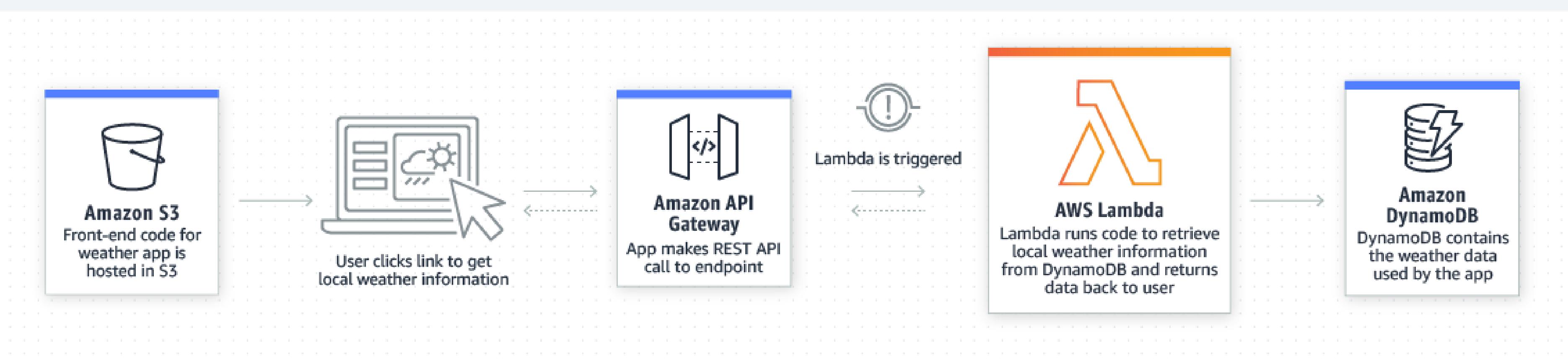


**Amazon Lambda - AWS**

Com o Amazon Lambda você paga apenas pelo tempo de computação utilizado. Execute códigos sem provisionar ou gerenciar servidores. Com o Lambda, você pode executar o código para praticamente qualquer tipo de aplicativo ou serviço de back-end. Confira o preço do Amazon...

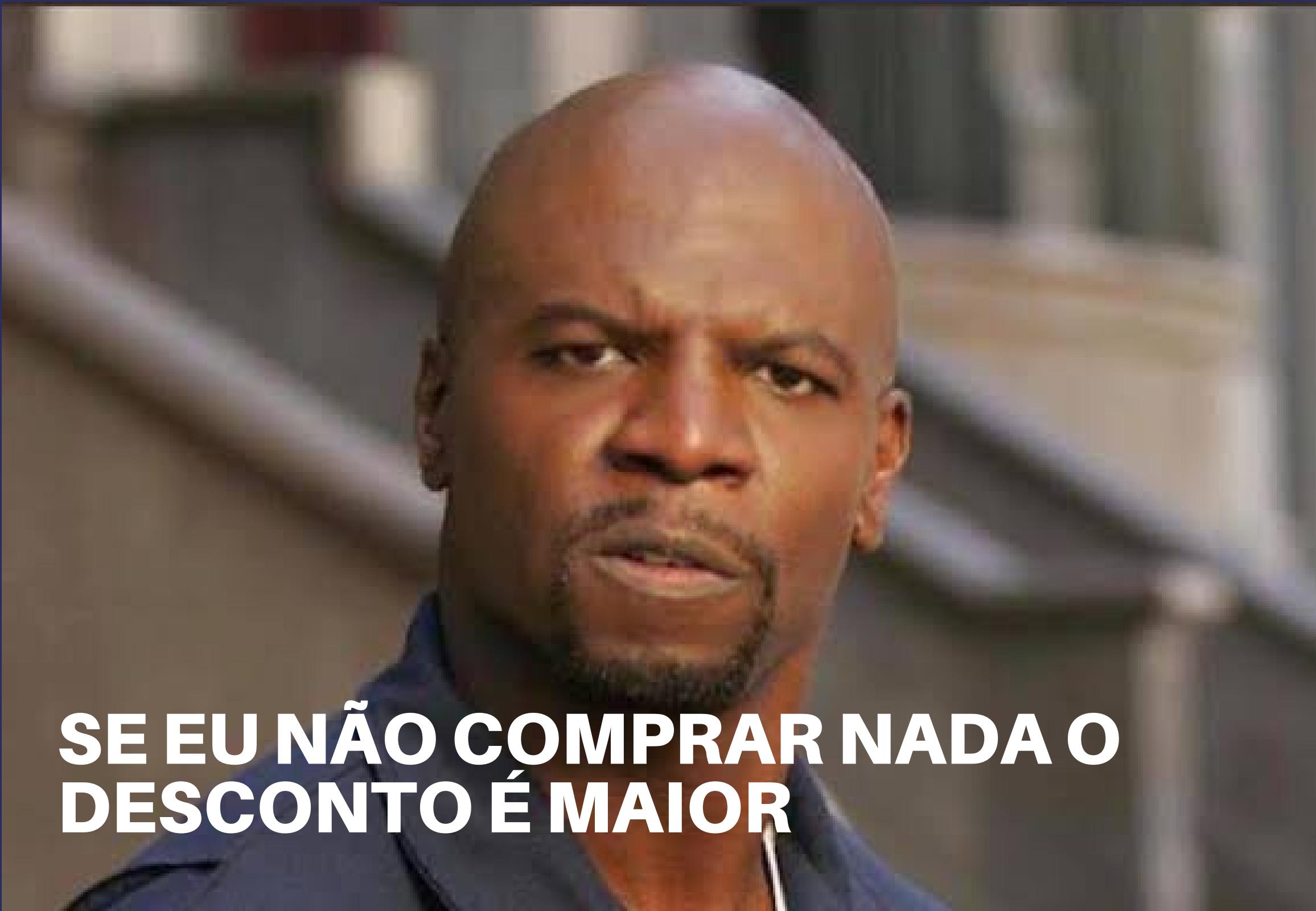
 Amazon Web Services





Esse é um modelo parecido com o que vamos aprender hoje :)

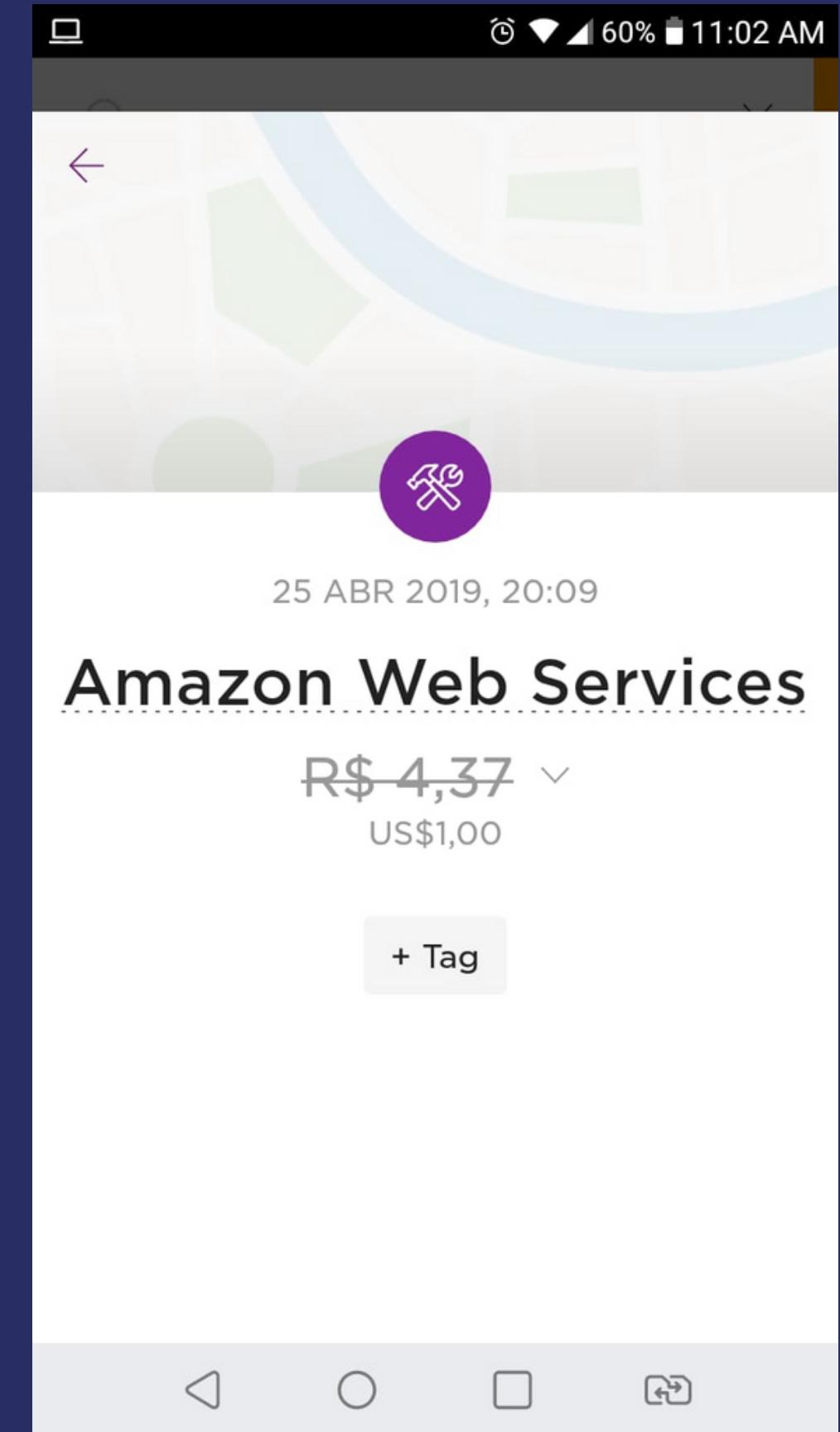
# PYTHON BRASIL [15]



**SE EU NÃO COMPRAR NADA O  
DESCONTO É MAIOR**



# PYTHON BRASIL [15]



# CRIANDO UM ALARME

- Evita cobranças caso você não tenha como pagar
- Usando os recursos gratuitos da AWS dá para aprender muita coisa
- Com esses recursos dá para criar muitas coisas
- Sejam cientistas, testem suas hipóteses
- Acesse o tutorial no Github: [https://github.com/GianaAlmeida/Python-Brasil-19---Serverless/blob/master/tutoriais/alarmes\\_aws.md](https://github.com/GianaAlmeida/Python-Brasil-19---Serverless/blob/master/tutoriais/alarmes_aws.md)



## CONHECENDO O FRAMEWORK

Usaremos o framework **Chalice**

- Alto nível
- Rápido deploy
- Criando um endpoint em um minuto

Instalar:

- Python3
- Pip3
- Chalice



## PYTHON BRASIL [15]

```
giana@giana:~/Desktop$ chalice --version
chalice 1.8.0, python 3.6.8, linux 5.0.0-29-generic
giana@giana:~/Desktop$ chalice new-project python_br_19
giana@giana:~/Desktop$ ls
python_br_19
giana@giana:~/Desktop$ ls python_br_19/
app.py  requirements.txt
```

```
giana@giana:~/Desktop$ cat python_br_19/app.py
from chalice import Chalice

PYTHON app = Chalice(app_name='python_br_19')

@app.route('/')
def index():
    return {'hello': 'world'}


# The view function above will return {"hello": "world"}
# whenever you make an HTTP GET request to '/'.
#
# Here are a few more examples:
#
# @app.route('/hello/{name}')
# def hello_name(name):
#     # '/hello/james' -> {"hello": "james"}
#     return {'hello': name}
#
# @app.route('/users', methods=['POST'])
# def create_user():
#     # This is the JSON body the user sent in their POST request.
#     user_as_json = app.current_request.json_body
#     # We'll echo the json body back to the user in a 'user' key.
#     return {'user': user_as_json}
#
# See the README documentation for more examples.
#
```

PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x9
giana@giana:~/Desktop$ cd python_br_19/
giana@giana:~/Desktop/python_br_19$ chalice local --no-autoreload
Serving on http://127.0.0.1:8000
127.0.0.1 - - [30/Sep/2019 20:08:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Sep/2019 20:08:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Sep/2019 20:08:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Sep/2019 20:08:12] "GET / HTTP/1.1" 200 -
[

giana@giana: ~/Desktop 72x10
giana@giana:~/Desktop$ curl 127.0.0.1:8000
{"hello":"world"}giana@giana:~/Desktop$ curl 127.0.0.1:8000
{"hello":"world"}giana@giana:~/Desktop$ curl 127.0.0.1:8000
{"hello":"world"}giana@giana:~/Desktop$ curl 127.0.0.1:8000
{"hello":"world"}giana@giana:~/Desktop$
```

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
giana@giana:~/Desktop/python_br_19$ chalice deploy
Creating deployment package.
Updating policy for IAM role: python_br_19-dev
Updating lambda function: python_br_19-dev
Updating rest API
Resources deployed:
  - Lambda ARN: arn:aws:lambda:us-west-2:558258168256:function:python_br
    _19-dev
  - Rest API URL: https://9r2t39gxw1.execute-api.us-west-2.amazonaws.com
    /api/
giana@giana:~/Desktop/python_br_19$ curl https://9r2t39gxw1.execute-api.
us-west-2.amazonaws.com/api/
{"hello": "world"}giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]

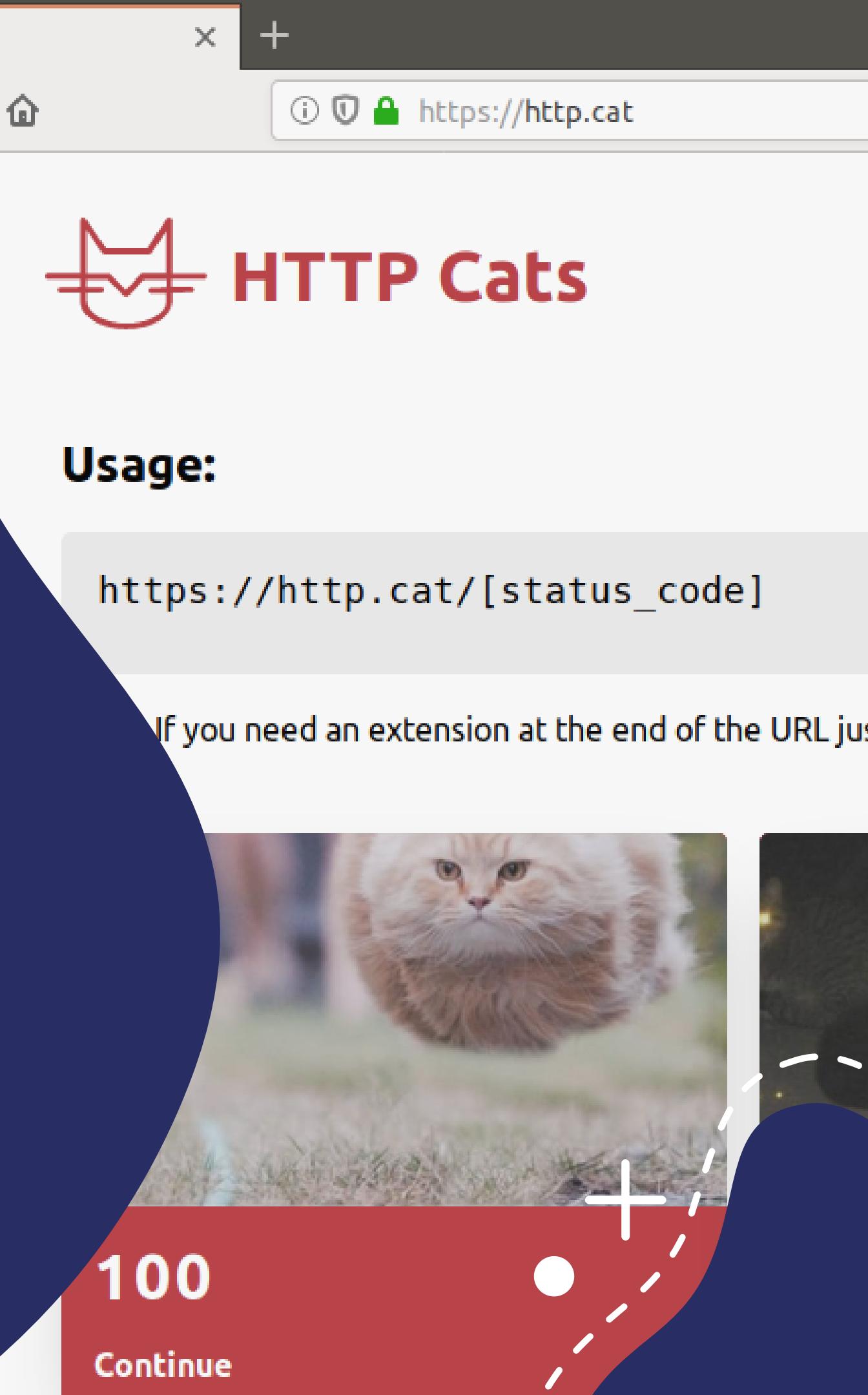
```
giana@giana: ~/Desktop/python_br_19 72x20
giana@giana:~/Desktop/python_br_19$ ls -a
. . . app.py .chalice .gitignore __pycache__ requirements.txt
giana@giana:~/Desktop/python_br_19$ cat .gitignore
.chalice/deployments/
.chalice/venv/
giana@giana:~/Desktop/python_br_19$ ls -a .chalice/
. . . config.json deployed deployments
giana@giana:~/Desktop/python_br_19$
giana@giana:~/Desktop/python_br_19$
giana@giana:~/Desktop/python_br_19$ python3 -m venv .chalice/venv
giana@giana:~/Desktop/python_br_19$ source .chalice/venv/bin/activate
(venv) giana@giana:~/Desktop/python_br_19$ 
```

# PYTHON BRASIL [15]

*Exercício*

nosso caso de uso será  
aprender os verbos http com  
foto de gatos

*Usando é claro, uma API*



100

Continue

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
(venv) giana@giana:~/Desktop/python_br_19$ curl https://http.cat/200 \
> --output 200.jpg
% Total    % Received % Xferd  Average Speed   Time     Time     Time
Current                                         Dload  Upload   Total   Spent   Left
Speed
0      0      0      0      0      0      0      0  ---:---:---  ---:---:---  ---:---:---
0      0      0      0      0      0      0      0  ---:---:---  ---:---:---  ---:---:---
100 27012  100 27012    0      0  17712      0  0:00:01  0:00:01  ---:---:---
17712
(venv) giana@giana:~/Desktop/python_br_19$ ls -a
.  200.jpg  .chalice  __pycache__
.. app.py   .gitignore requirements.txt
(venv) giana@giana:~/Desktop/python_br_19$ shotwell 200.jpg jpg
```

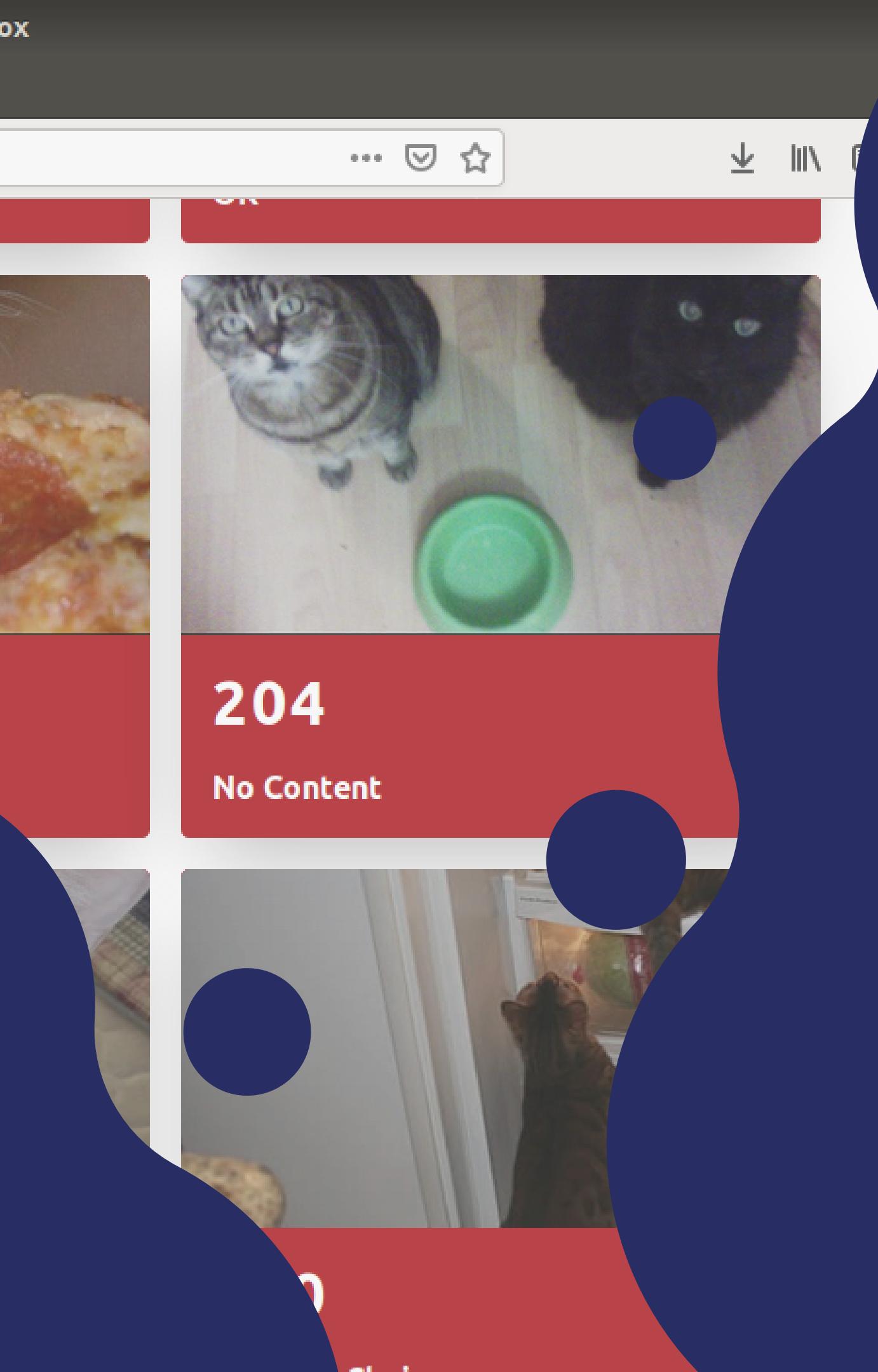
# PYTHON



# SALVANDO AS FOTOS NA AWS

*Armazenamento*

Conhecendo o AWS S3 e  
aprendendo sobre a  
biblioteca boto3



# PYTHON BRASIL [15]



**Armazenamento S3 - Simple Storage Service**

O armazenamento S3 da AWS ajuda você com backup e arquivamento, big data e mais. O AWS S3 oferece recursos de segurança e conformidade que cumprem os requisitos normativos mais rigorosos. Clientes podem gerenciar dados de otimização de custo, controle de acesso e conformidade.

 Amazon Web Services



# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
(venv) giana@giana:~/Desktop/python_br_19$ pip install boto3
Collecting boto3
  Cache entry deserialization failed, entry ignored
    Downloading https://files.pythonhosted.org/packages/90/c3/055ecc6de247
    6f76d16a912889b7b84205686aa61501e6a2ce113b8e860c/boto3-1.9.239-py2.py3-n
    one-any.whl (128kB)
      100% |██████████| 133kB 1.6MB/s
Collecting botocore<1.13.0,>=1.12.239 (from boto3)
  Cache entry deserialization failed, entry ignored
    Downloading https://files.pythonhosted.org/packages/d5/96/d08e4e9cf979
    4743b37aed252b13cc6769b253f81df7cd525ec64966d57d/botocore-1.12.239-py2.p
    y3-none-any.whl (5.7MB)
      100% |██████████| 5.7MB 257kB/s
Collecting jmespath<1.0.0,>=0.7.1 (from boto3)
```

## PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
Installing collected packages: urllib3, jmespath, docutils, six, python-
dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.9.239 botocore-1.12.239 docutils-0.15.2 j
mespath-0.9.4 python-dateutil-2.8.0 s3transfer-0.2.1 six-1.12.0 urllib3-
1.25.6
(venv) giana@giana:~/Desktop/python_br_19$ pip freeze
boto3==1.9.239
botocore==1.12.239
docutils==0.15.2
jmespath==0.9.4
pkg-resources==0.0.0
python-dateutil==2.8.0
s3transfer==0.2.1
six==1.12.0
urllib3==1.25.6
(venv) giana@giana:~/Desktop/python_br_19$ cat requirements.txt
(venv) giana@giana:~/Desktop/python_br_19$ pip freeze > requirements.txt

(venv) giana@giana:~/Desktop/python_br_19$
```

boto3  
botocore  
pkg-resources



# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
(venv) giana@giana:~/Desktop/python_br_19$ cat .chalice/config.json
{
    "version": "2.0",
    "app_name": "python_br_19",
    "stages": {
        "dev": {
            "api_gateway_stage": "api"
        }
    }
}
(venv) giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 72x20
(venv) giana@giana:~/Desktop/python_br_19$ cat .chalice/config.json
{
    "version": "2.0",
    "app_name": "python_br_19",
    "environment_variables": {
        "BASE_URL": "https://http.cat",
        "FILE_TYPE": "jpg"
    },
    "stages": {
        "dev": {
            "api_gateway_stage": "api"
        }
    }
}
(venv) giana@giana:~/Desktop/python_br_19$ 
```



Usage:

[https://http.cat/\[status\\_code\]](https://http.cat/[status_code])

## PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 87x24
(venv) giana@giana:~/Desktop/python_br_19$ cat app.py
import os
import requests

from chalice import Chalice

app = Chalice(app_name='python_br_19')

BASE_URL = os.environ['BASE_URL']
FILE_TYPE = os.environ['FILE_TYPE']

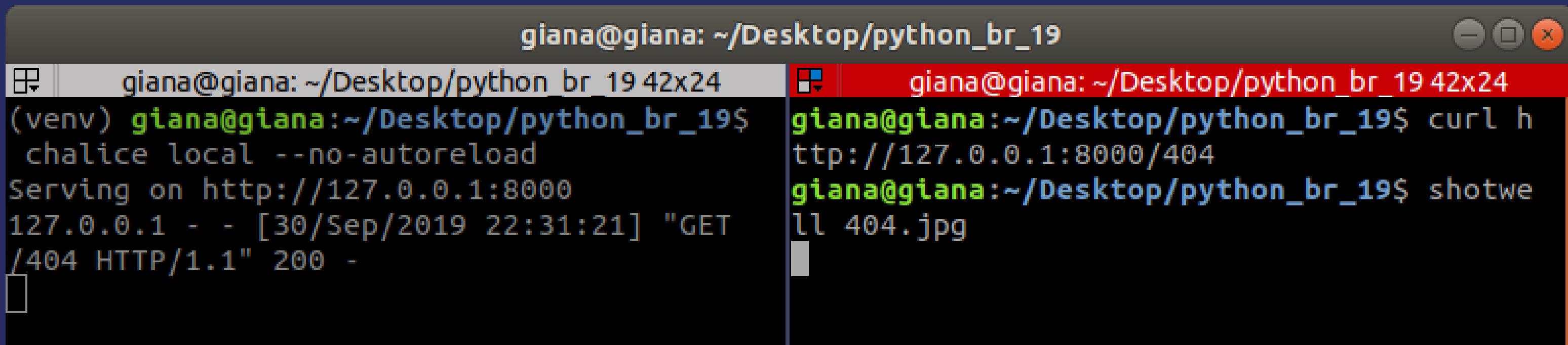
@app.route('/{code}')
def index(code):
    url = f'{BASE_URL}/{code}.{FILE_TYPE}'
    r = requests.get(url)

    with open(f'./{code}.{FILE_TYPE}', 'wb') as f:
        f.write(r.content)

    return {'status': r.status_code}

(venv) giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]



```
giana@giana: ~/Desktop/python_br_19
giana@giana: ~/Desktop/python br_19 42x24
(venv) giana@giana:~/Desktop/python_br_19$ chalice local --no-autoreload
Serving on http://127.0.0.1:8000
127.0.0.1 - - [30/Sep/2019 22:31:21] "GET
/404 HTTP/1.1" 200 -
giana@giana: ~/Desktop/python_br_19
giana@giana: ~/Desktop/python br_19 42x24
giana@giana:~/Desktop/python_br_19$ curl h
ttp://127.0.0.1:8000/404
giana@giana:~/Desktop/python_br_19$ shotwe
ll 404.jpg
```

P Y T H O N E

404.jpg (~/Desktop/python\_br\_19) - Shotwell

File Edit View Photo Help



404

Not Found

Rotate Crop Straighten Red-eye Adjust Enhance

← →

19 42x24  
\$ curl h  
\$ shotwe

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 87x24
(venv) giana@giana:~/Desktop/python_br_19$ mkdir chalicelib
(venv) giana@giana:~/Desktop/python_br_19$ touch chalicelib/upload.py
(venv) giana@giana:~/Desktop/python_br_19$ nano chalicelib/upload.py
(venv) giana@giana:~/Desktop/python_br_19$ cat chalicelib/upload.py
import logging
import boto3
from botocore.exceptions import ClientError

def upload_file(file_name, bucket_name, object_name=None):
    if object_name is None:
        object_name = file_name

    s3_client = boto3.client('s3')

    try:
        response = s3_client.upload_file(file_name, bucket_name, object_name)
    except ClientError as e:
        logging.error(e)
        return False

    return True
(venv) giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 87x25
(venv) giana@giana:~/Desktop/python_br_19$ cat app.py
import os
import requests

from chalice import Chalice
from chalicelib.upload import upload_file

app = Chalice(app_name='python_br_19')

BASE_URL = os.environ['BASE_URL']
FILE_TYPE = os.environ['FILE_TYPE']

@app.route('/{code}')
def index(code):
    url = f'{BASE_URL}/{code}.{FILE_TYPE}'
    r = requests.get(url)

    with open(f'./{code}.{FILE_TYPE}', 'wb') as f:
        f.write(r.content)
        upload_file(f'{code}.{FILE_TYPE}', 'giana-test')

    return {'status': r.status_code}

(venv) giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]

```
giana@giana:~/Desktop/python_br_19 87x6
(venv) giana@giana:~/Desktop/python_br_19$ chalice local --no-autoreload
Serving on http://127.0.0.1:8000
Found credentials in shared credentials file: ~/.aws/credentials
127.0.0.1 - - [30/Sep/2019 22:52:03] "GET /404 HTTP/1.1" 200 -
[ ]
```

```
giana@giana:~/Desktop/python_br_19 87x6
giana@giana:~/Desktop/python_br_19$ curl http://127.0.0.1:8000/404
giana@giana:~/Desktop/python_br_19$ [ ]
```

# PYTHON BRASIL [15]

The screenshot shows the AWS S3 console interface. At the top, the navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a star icon, a notification bell, the user name 'giana', 'Global' dropdown, and 'Support' dropdown. Below the navigation bar, the path 'Amazon S3 > giana-test' is displayed, with 'cloud' written above 'giana-test'. A search bar contains the placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', 'Create folder', 'Download', and 'Actions'. To the right of these buttons, the region 'US West (Oregon)' and a refresh icon are shown. The main content area displays a table of files in the 'giana-test' bucket. The table has columns: a checkbox, 'Name' (with a dropdown arrow), 'Last modified', 'Size', and 'Storage class'. One file is listed: '404.jpg' (checkbox checked, image thumbnail, last modified Sep 30, 2019 10:52:03 PM GMT-0300, size 77.3 KB, storage class Standard). Below the table, a message indicates 'Viewing 1 to 1'.

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	404.jpg	Sep 30, 2019 10:52:03 PM GMT-0300	77.3 KB	Standard

# PYTHON BRASIL [15]

```
giana@giana: ~/Desktop/python_br_19 80x34
(venv) giana@giana:~/Desktop/python_br_19$ cat app.py
import os
import requests

from chalice import Chalice
from chalicelib.upload import upload_file

app = Chalice(app_name='python_br_19')

BASE_URL = os.environ['BASE_URL']
FILE_TYPE = os.environ['FILE_TYPE']

@app.route('/{code}')
def index(code):
    url = f'{BASE_URL}/{code}.{FILE_TYPE}'
    r = requests.get(url)

    folder = '/tmp/'
    if not os.path.exists(folder):
        os.system(f'mkdir -m 777 {folder} && touch {folder}/{code}.{FILE_TYPE}')

    with open(f'{folder}/{code}.{FILE_TYPE}', 'wb') as f:
        f.write(r.content)
        os.system(f'chmod 777 {folder}/{code}.{FILE_TYPE}')
        upload_file(f'{folder}/{code}.{FILE_TYPE}', 'giana-test')

    return {'status': r.status_code}
(venv) giana@giana:~/Desktop/python_br_19$
```

# PYTHON BRASIL [15]

Screenshot of the AWS IAM Roles page for the role "python\_br\_19-dev".

The left sidebar shows the IAM navigation menu, and the top bar shows the user "giana @ 5582-5816-8256" and other account details.

**Summary**

Role ARN	arn:aws:iam::558258168256:role/python_br_19-dev
Role description	<a href="#">Edit</a>
Instance Profile ARNs	<a href="#">Edit</a>
Path	/
Creation time	2019-10-14 21:03 UTC-0300
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

**Permissions** (selected tab) | [Trust relationships](#) | [Tags](#) | [Access Advisor](#) | [Revoke sessions](#)

**Permissions policies (2 policies applied)**

Policy name	Policy type	Actions
AmazonS3FullAccess	AWS managed policy	<a href="#">X</a>
python_br_19-dev	Inline policy	<a href="#">X</a>

**Permissions boundary (not set)**

[Attach policies](#) | [+ Add inline policy](#)

**Feedback** | **English (US)** | © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. | [Privacy Policy](#) | [Terms of Use](#)

# PYTHON BRASIL [15]

AWS Services Resource Groups 🔍 giana @ 5582-5816-8256 Oregon Support

Lambda > Functions > python\_br\_19-dev ARN - arn:aws:lambda:us-west-2:558258168256:function:python\_br\_19-dev ⓘ

### python\_br\_19-dev

Throttle Qualifiers Actions Select a test event Test Save

Configuration Monitoring

Designer

python\_br\_19-dev

Layers (0)

API Gateway

+ Add trigger

Amazon CloudWatch Logs

Resources that the function's role has access to appear here

Function code Info

The deployment package of your Lambda function "python\_br\_19-dev" is too large to enable inline code editing. However, you can still invoke your function.

Code entry type Runtime Handler Info

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

# PYTHON BRASIL [15]

The screenshot shows the AWS CloudWatch Logs interface. The left sidebar has a 'Logs' section selected, showing various log streams. The main area displays a list of log streams for the '/aws/lambda/python\_br\_19-dev' function. Each stream entry includes a checkbox, the stream name, and the last event time.

Log Stream	Last Event Time
2019/10/15[\$LATEST]deb7fff3c9224b16935da7651446d1a2	2019-10-14 21:47 UTC-3
2019/10/15[\$LATEST]2c0577aea64b494982465613c89c49df	2019-10-14 21:43 UTC-3
2019/10/15[\$LATEST]3cb421b073184e14bab1fd4492daf376	2019-10-14 21:42 UTC-3
2019/10/15[\$LATEST]91b4215dafbc445481c9dfc96905779a	2019-10-14 21:41 UTC-3
2019/10/15[\$LATEST]9be51185683c4bba81d1519a1ebb5a7b	2019-10-14 21:37 UTC-3
2019/10/15[\$LATEST]5c38f00bbb044c42beea1fc648328806	2019-10-14 21:36 UTC-3
2019/10/15[\$LATEST]e343cddbae17410b8c34d39d45f98b28	2019-10-14 21:34 UTC-3
2019/10/15[\$LATEST]07687f17696c4220879faf0600d9c03d	2019-10-14 21:32 UTC-3
2019/10/15[\$LATEST]086cf9fb43c4a0a9bf815187e053365	2019-10-14 21:30 UTC-3
2019/10/15[\$LATEST]6eac84ec0beb4cc18ce09545e4edab9a	2019-10-14 21:29 UTC-3
2019/10/15[\$LATEST]e926f20f368147908faaa2dd4b8f0f1d	2019-10-14 21:28 UTC-3
2019/10/15[\$LATEST]c298c0c20de44366b297b4a66a517882	2019-10-14 21:27 UTC-3
2019/10/15[\$LATEST]9a1edc2c7a6844f280a2a938c68480df	2019-10-14 21:25 UTC-3
2019/10/15[\$LATEST]56424bab3a2419f98b663d6eb3d9ab9	2019-10-14 21:23 UTC-3
2019/10/15[\$LATEST]5c77b22874fe4bbeb4550dcdb34c9741	2019-10-14 21:21 UTC-3
2019/10/15[\$LATEST]afcf06b2c3ed4134972996bdc117e7bb	2019-10-14 21:12 UTC-3
2019/10/15[\$LATEST]a19474eeee7c24ac3b4e6e3c6ed80e553	2019-10-14 21:09 UTC-3
2019/10/15[\$LATEST]dde84b69e525456db0efbe6e770c2267	2019-10-14 21:05 UTC-3
2019/09/30[\$LATEST]e20031572f4a41c6b5e46861eeb9a9b9	2019-09-30 20:13 UTC-3
2019/09/30[\$LATEST]745d309de18c4aa6b44c0b25bd3ba941	2019-09-30 20:09 UTC-3

## BOAS PRÁTICAS

- Tempo de execução de até 5 minutos (permite até 15 minutos mas quebra um pouco o modelo do serveless)
- Redes privadas virtuais (VPCs) para APIs (endpoints) externos
- Se não vai usar no modelo de API, se é uma "API" interna, trocar routes por function
- Serviços acionados por eventos
- Testes automatizados
- Usar duas zonas de disponibilidade, se falhar em um local no outro estará funcionando
- Muito DevOps



## RECOMENDAÇÕES AWS

- Separe o manipulador do Lambda da lógica central. Isso permite que você crie uma função mais fácil para teste de unidade.
- Certifique-se de que qualquer configuração ou dependências externalizadas recuperadas pelo código sejam armazenadas e referenciadas localmente após a execução. Limite a reinicialização de variáveis/objetos em cada invocação.
- Use Variáveis de ambiente do AWS Lambda para transmitir parâmetros operacionais para sua função.

## RECOMENDAÇÕES AWS

- Use Variáveis de ambiente do AWS Lambda para transmitir parâmetros operacionais para sua função.
- Minimize o tamanho do pacote de implantação para as necessidades de seu tempo de execução.
- Minimize a complexidade de suas dependências.
- Evite usar código recursivo



# *Leaf Agriculture*

ITS ALL ABOUT THE JOURNEY  
KEEP CALM AND ENJOY



PYTHON BRASIL [15]

# SERVERLESS - ZERO TO HERO

By Giana de Almeida

Thank  
you  
guys

## REFERENCIAS

- AWS. **Boto3 Documentation - Upload files.** 2019. Disponível em: <<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-uploading-files.html>>. Acesso em: 13 set. 2019.
- SARYERWINNIE., James. **Chalice Read The Docs.** 2016. Disponível em: <<https://chalice.readthedocs.io/en/latest/topics/configfile.html#id1>>. Acesso em: 13 set. 2019.
- AWS. **Melhores práticas para trabalhar com funções do AWS Lambda.** Disponível em: <[https://docs.aws.amazon.com/pt\\_br/lambda/latest/dg/best-practices.html](https://docs.aws.amazon.com/pt_br/lambda/latest/dg/best-practices.html)>. Acesso em: 13 set. 2019.

## PYTHON BRASIL [15]

# REFERENCIAS

- SHARMA, Saurabh. **Building Serverless Python Apps Using AWS Chalice.** 2018. Disponível em: <<https://realpython.com/aws-chalice-serverless-python/>>. Acesso em: 13 set. 2019.
- HAQ, Siraj Ul. **Introduction to Monolithic Architecture and MicroServices Architecture.** 2018. Disponível em: <<https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63>>. Acesso em: 13 set. 2019.
- APIS, Public. **Public APIs Build Status.** Disponível em: <<https://github.com/public-apis/public-apis>>. Acesso em: 13 set. 2019.

# REFERENCIAS

- FIDELIS, Matheus. **13 coisas que aprendi em 1 ano usando Serverless em produção.** 2019. Disponível em: <<https://medium.com/@fidelissauro/13-coisas-que-aprendi-em-1-ano-usando-serverless-em-produ%C3%A7%C3%A3o-40e4e5e50470>>. Acesso em: 13 set. 2019.
- SOUZA, Evandro. **Serverless — mais foco nas suas aplicações.** 2018. Disponível em: <<https://medium.com/trainingcenter/serverless-mais-foco-nas-suas-aplica%C3%A7%C3%A3o-8606e7896c57>>. Acesso em: 13 set. 2019.