PYTHON BRASIL [15]

SERVERLESS - ZERO TO HERO

By Giana de Almeida

ABOUT THE SPEAKER

INTEGRATION ENGINEER

Python community engager,
Pyladies Floripa,

PYTHON BRASIL [15]

About Leaf

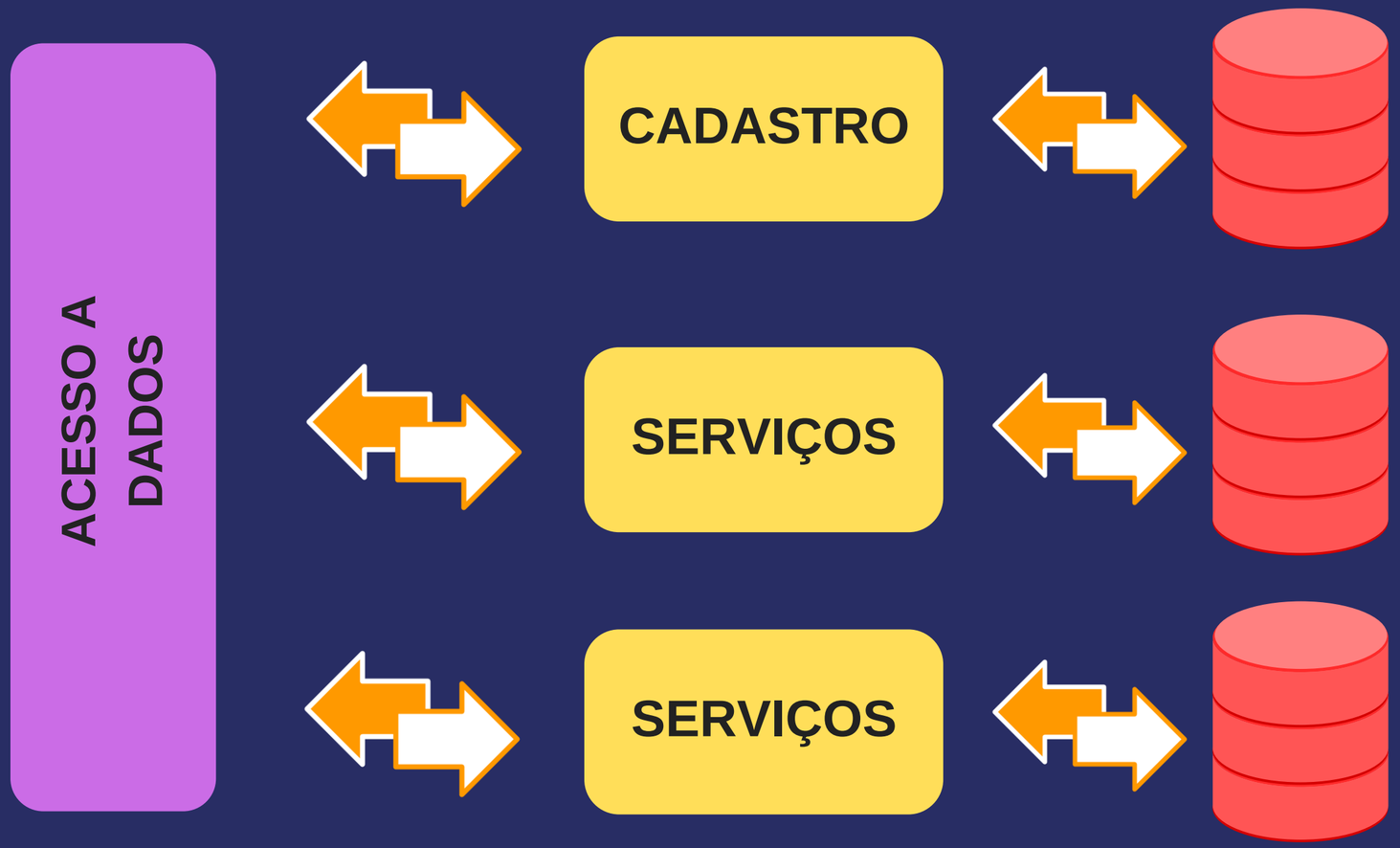
The easiest way to connect agriculture data across platforms.

PYTHON BRASIL [15]

APLICAÇÃO CENTRALIZADA

PYTHON BRASIL [15]

APLICAÇÃO DESCENTRALIZADA



PYTHON BRASIL [15]

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

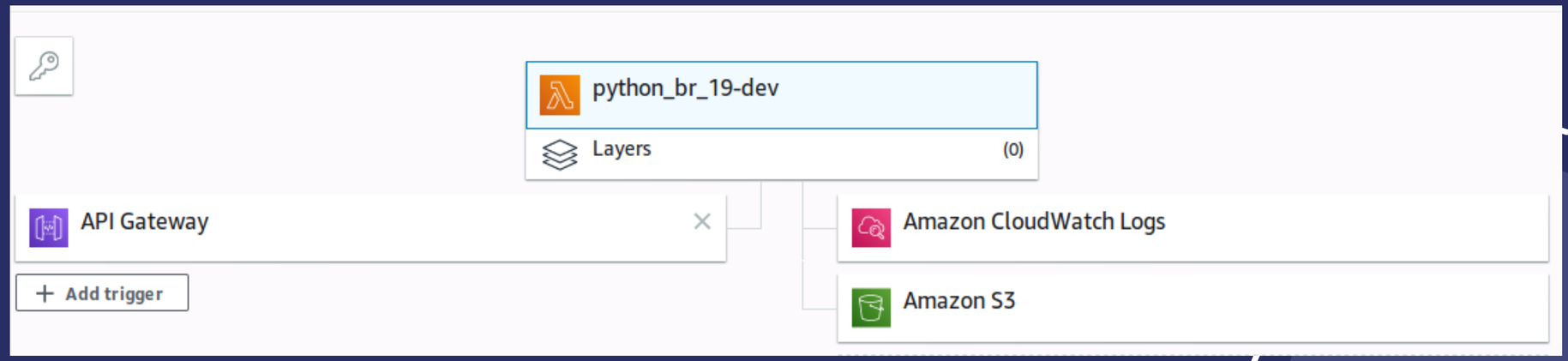
LIMITAÇÕES

- 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

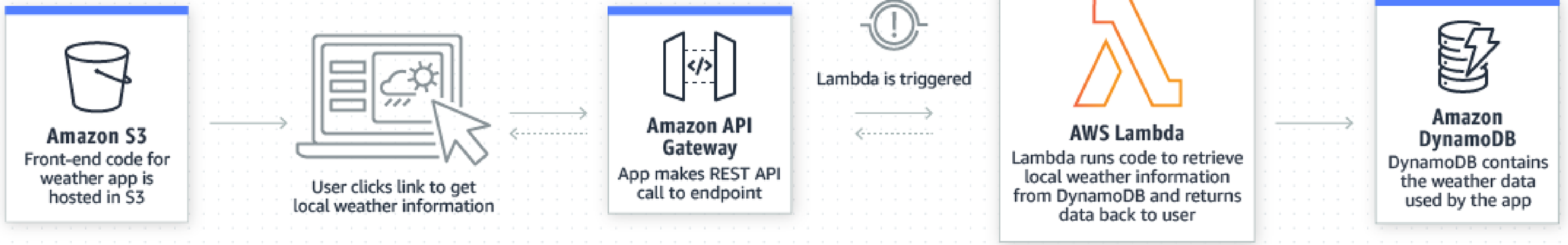
PYTHON BRASIL [15]

CONHECENDO OS RECURSOS

- Funções lambda
- API Gateway
- S3
- CloudWatch

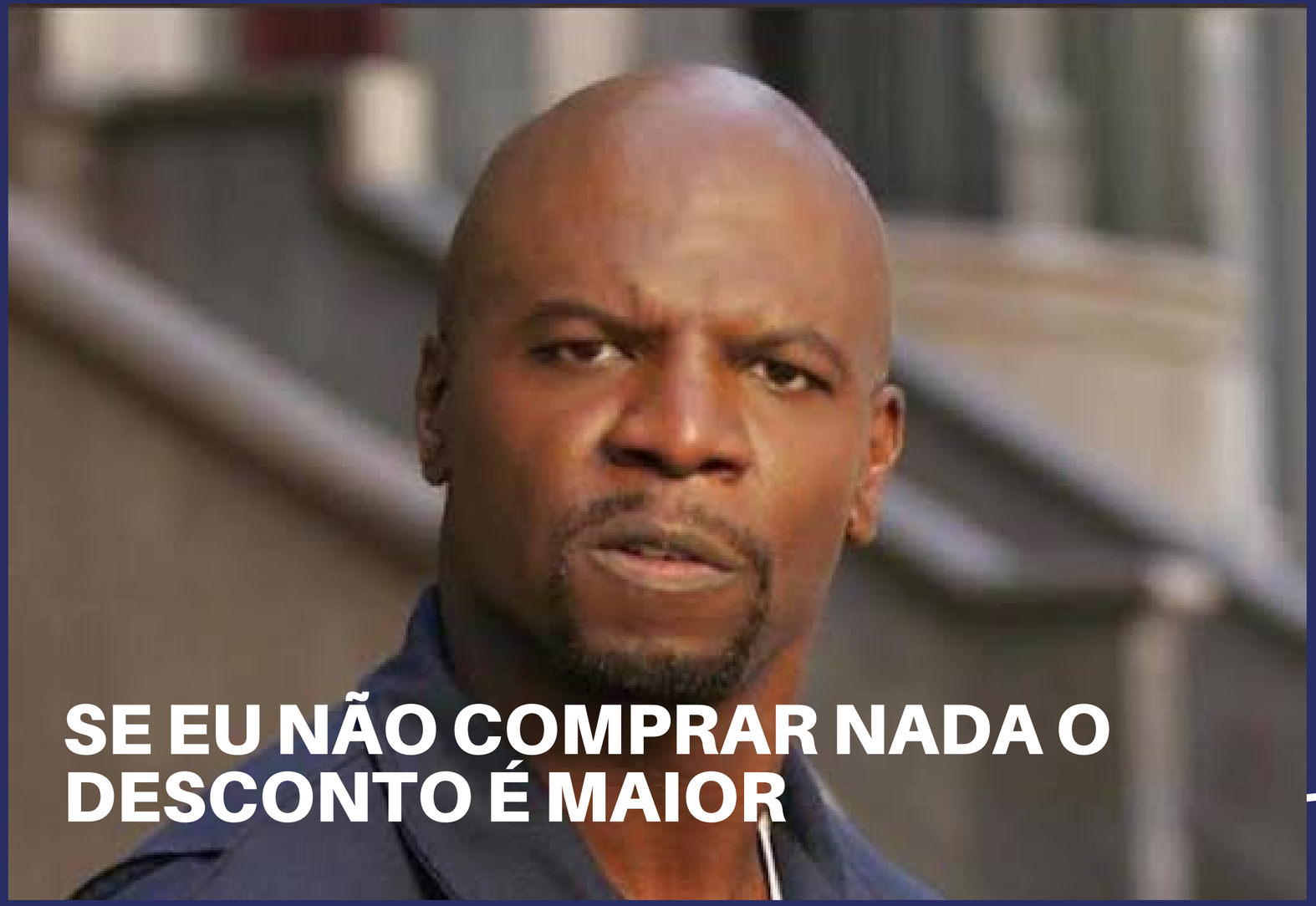


PYTHON BRASIL [15]



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

PYTHON BRASIL [15]

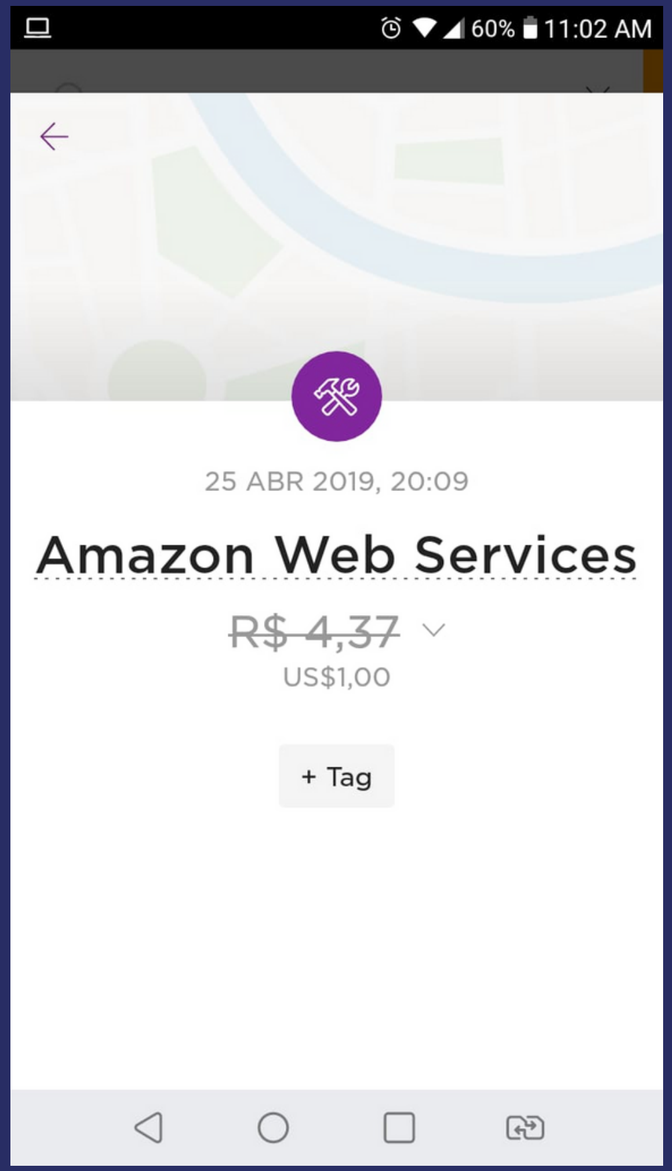


PYTHON BRASIL [15]

TAXA DE $1,

AWS EXTORNOU

E apenas para confirmar o cartão



PYTHON BRASIL [15]

CRIANDO UM ALARME

- Evita cobranças caso você não tenha como pagar
- Usando os recursos gratuítos 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---Serveless/blob/master/tutoriais/alarmes_aws.md

PYTHON BRASIL [15]

CONHECENDO O FRAMEWORK

Usaremos o framework **Chalice**

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

Instalar:

- Python3
- Pip3
- Chalice

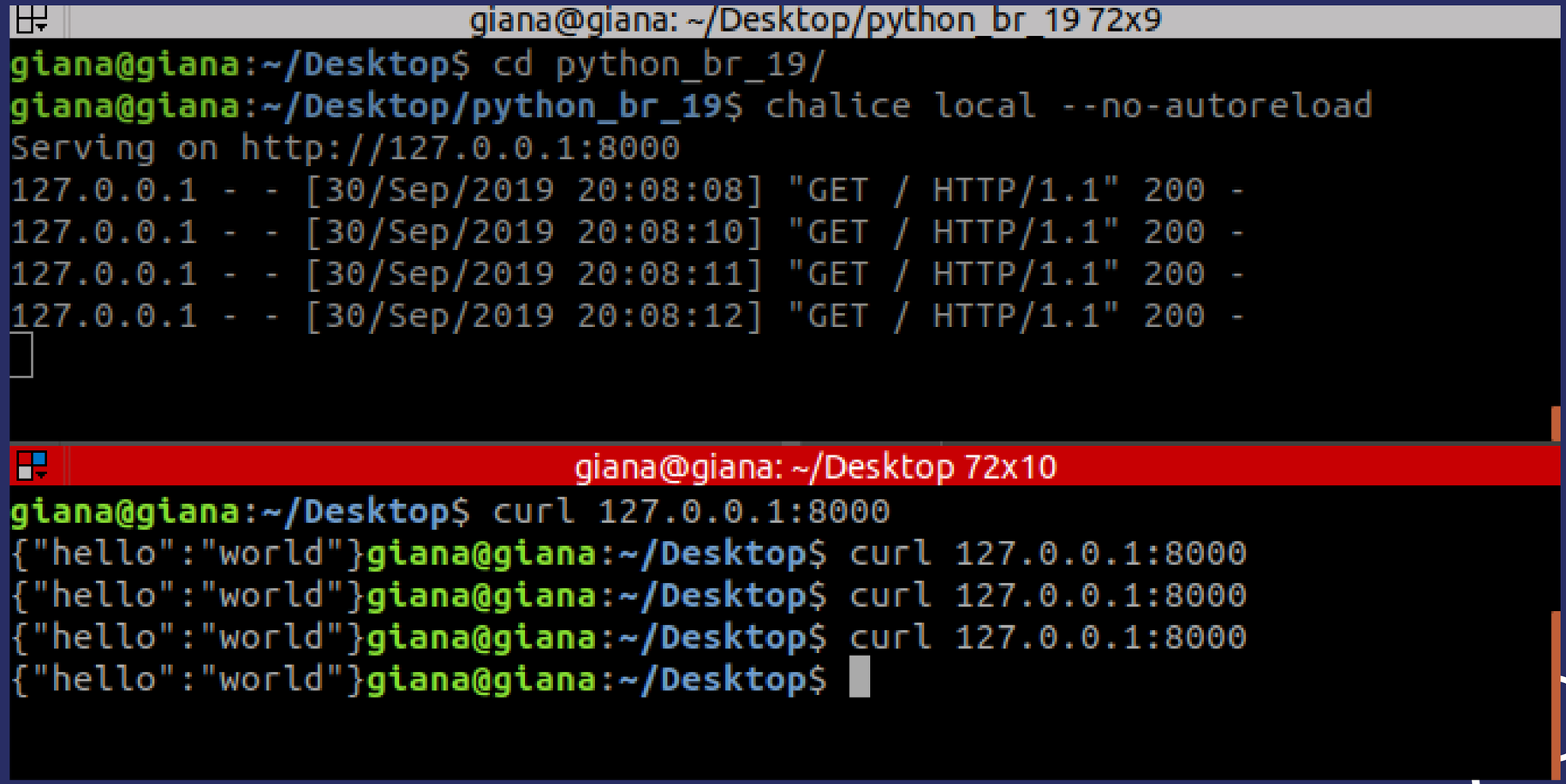
PYTHON BRASIL [15]

```
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
```

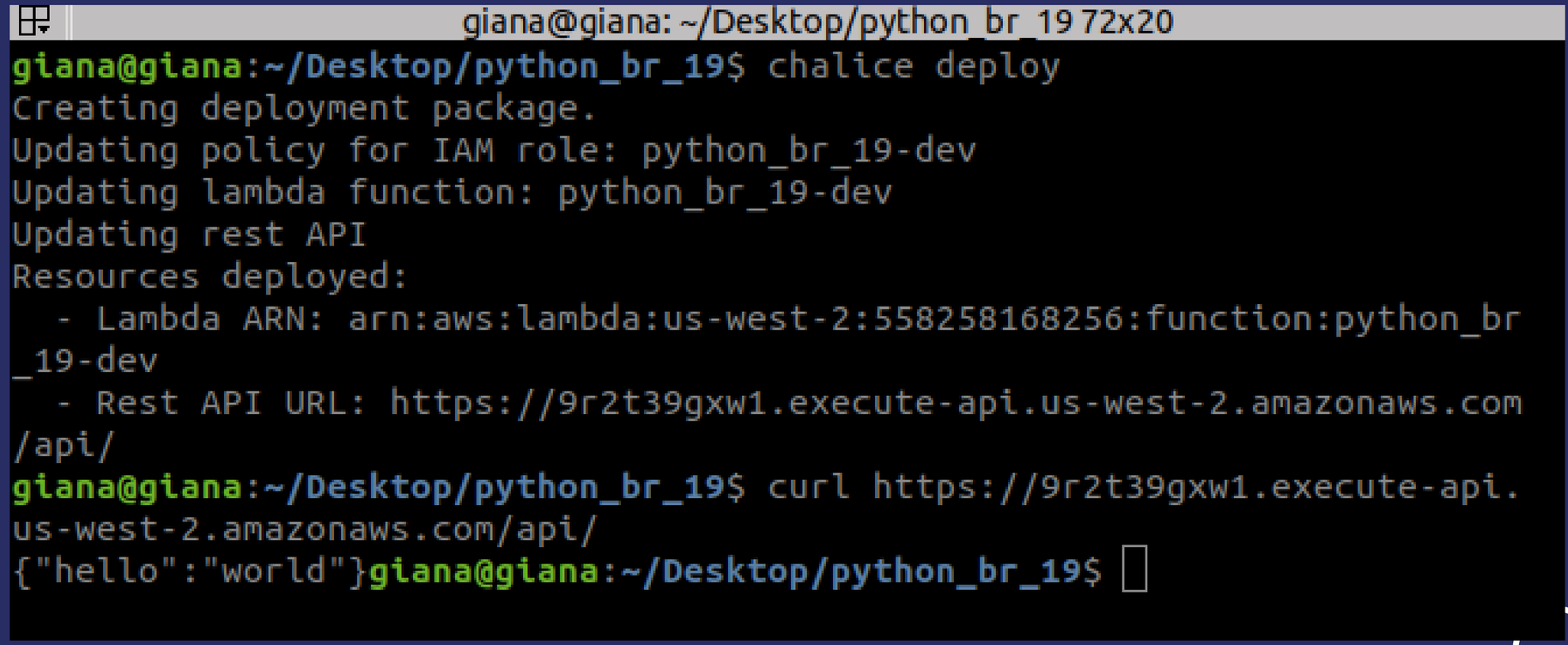
PYTHON

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

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.
#
```

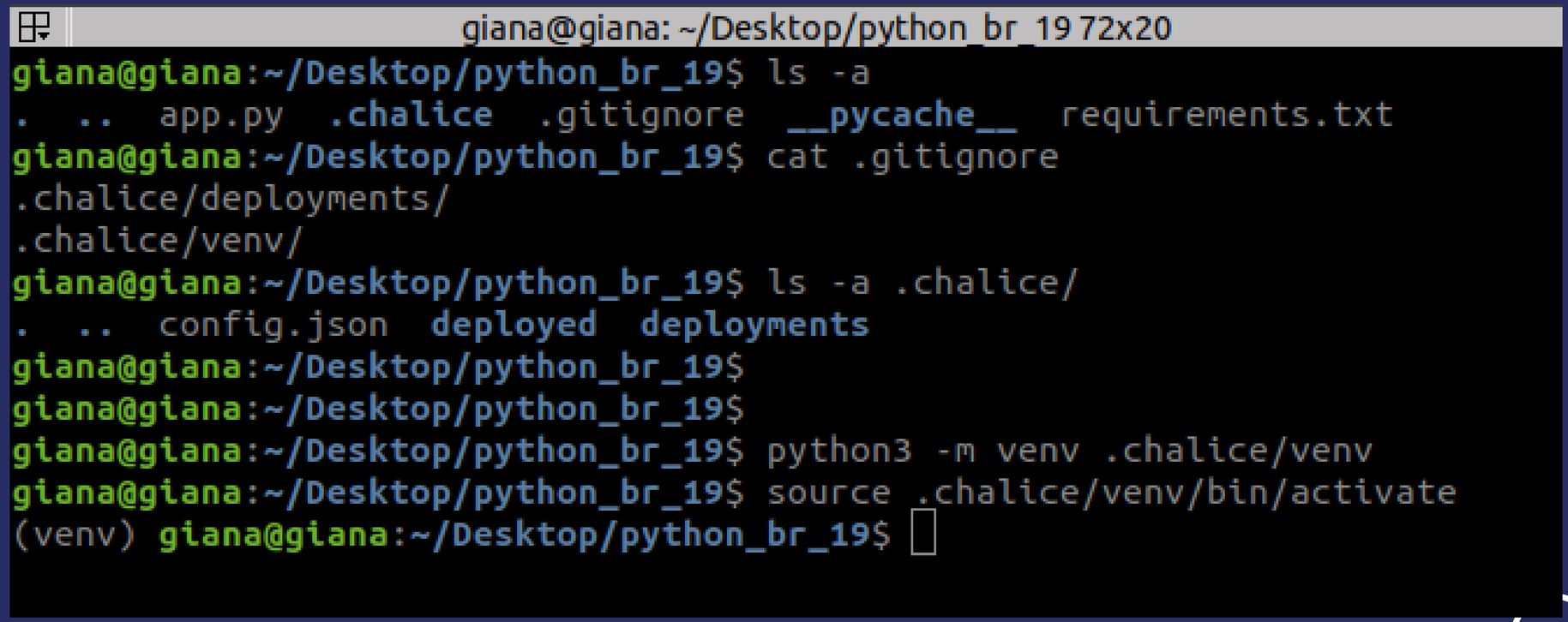
PYTHON BRASIL [15]



PYTHON BRASIL [15]



PYTHON BRASIL [15]

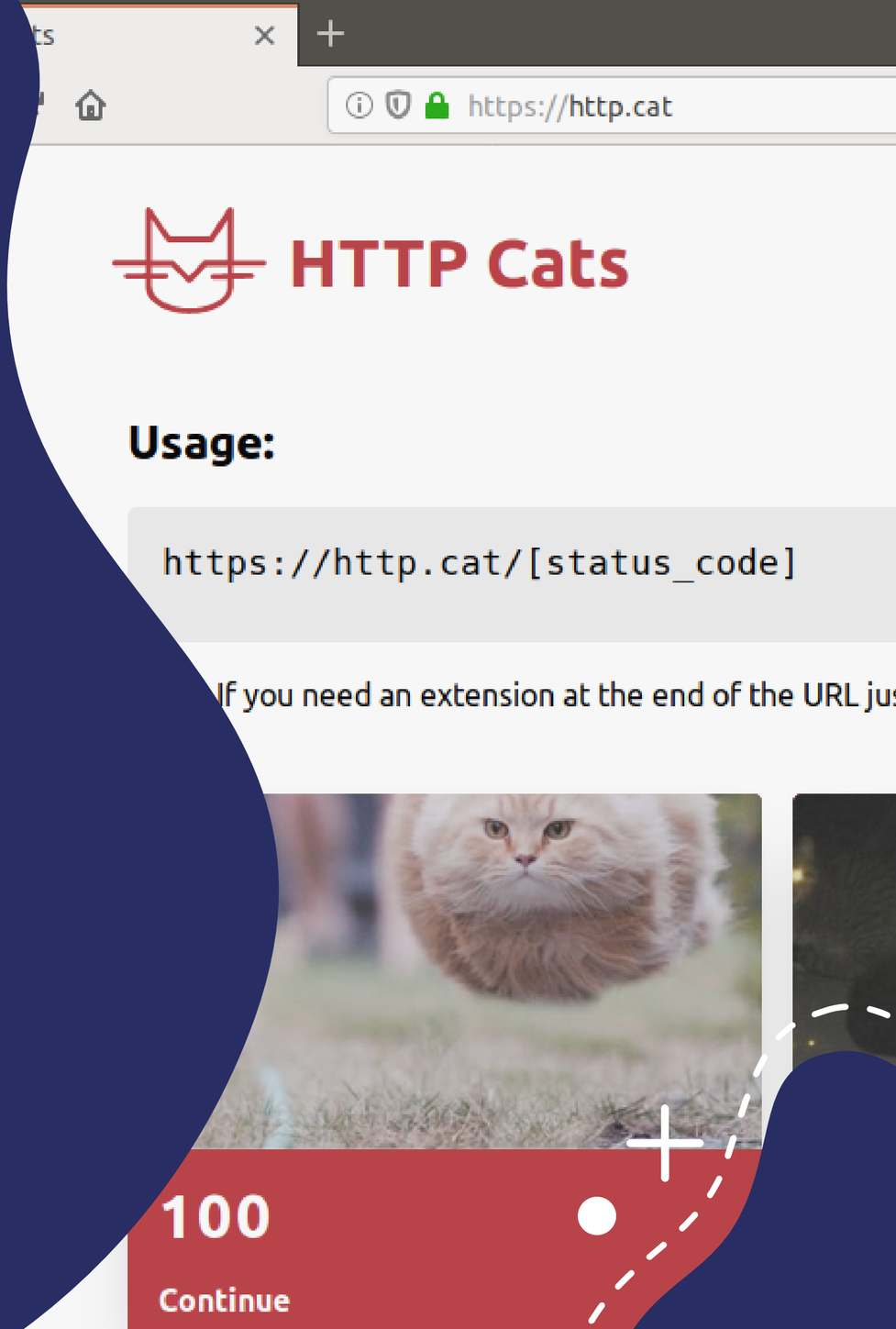


PYTHON BRASIL [15]

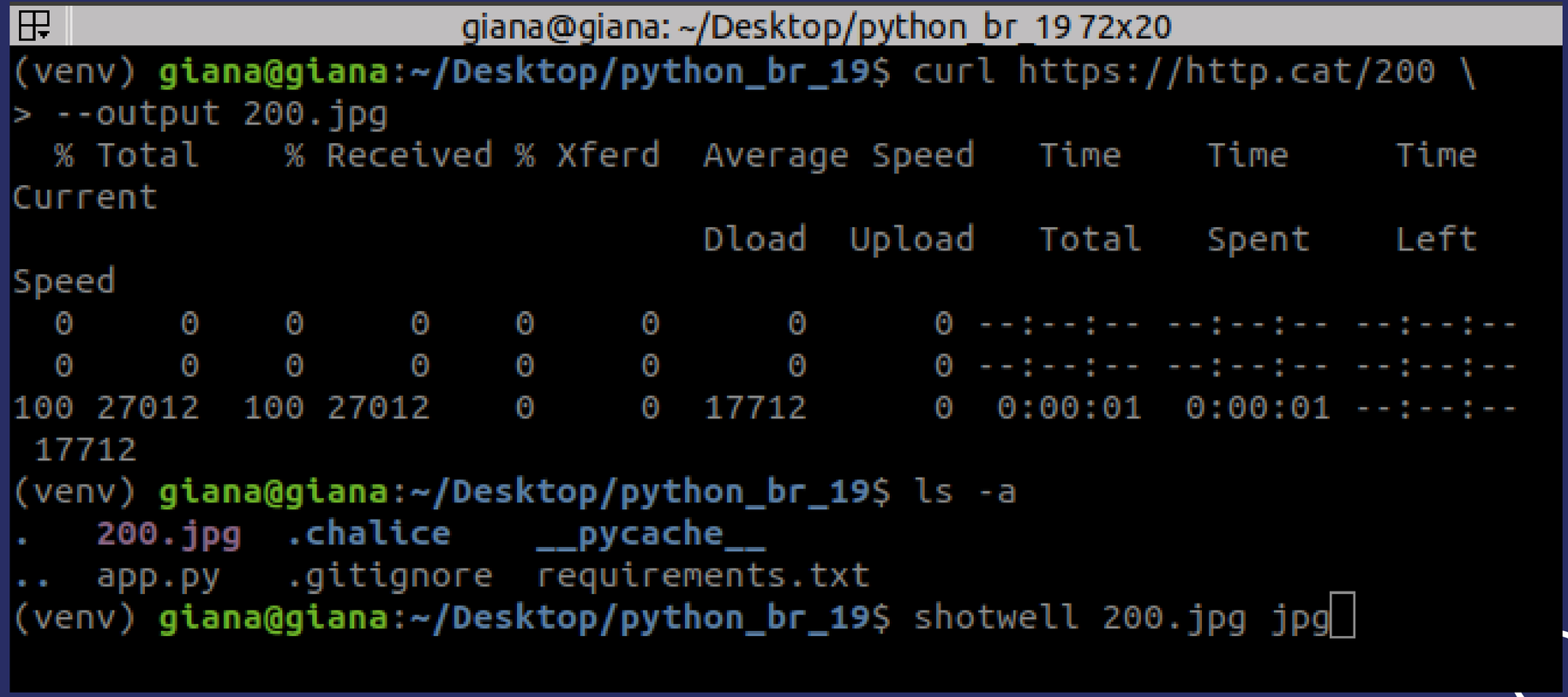
Exercício

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

Usando é claro, uma API



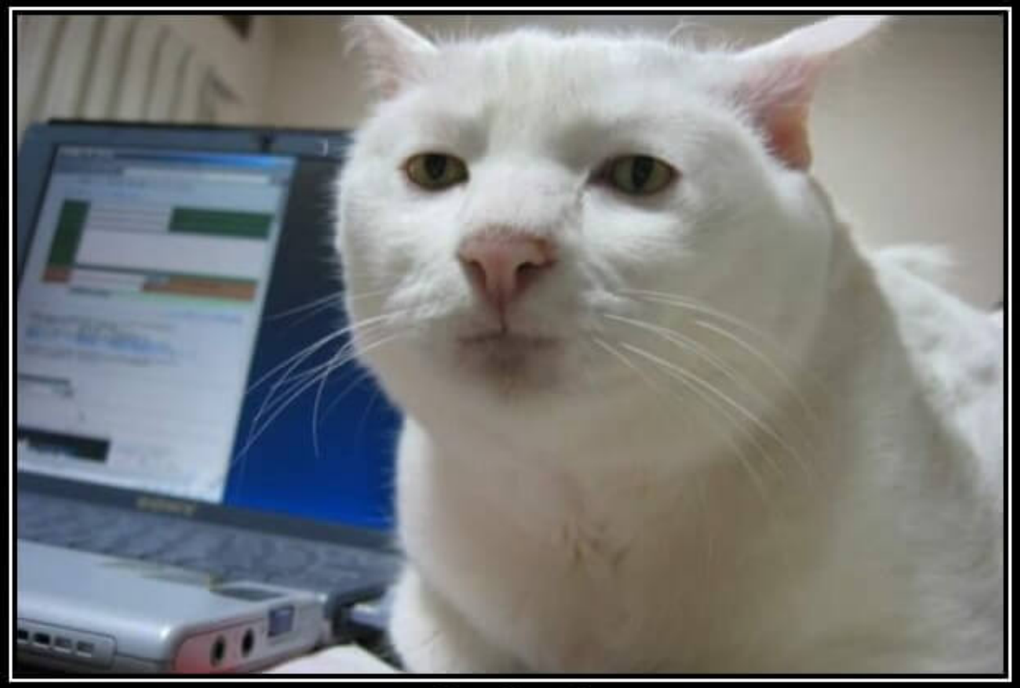
PYTHON BRASIL [15]



PYTHON

200.jpg (~/Desktop/python_br_19) - Shotwell

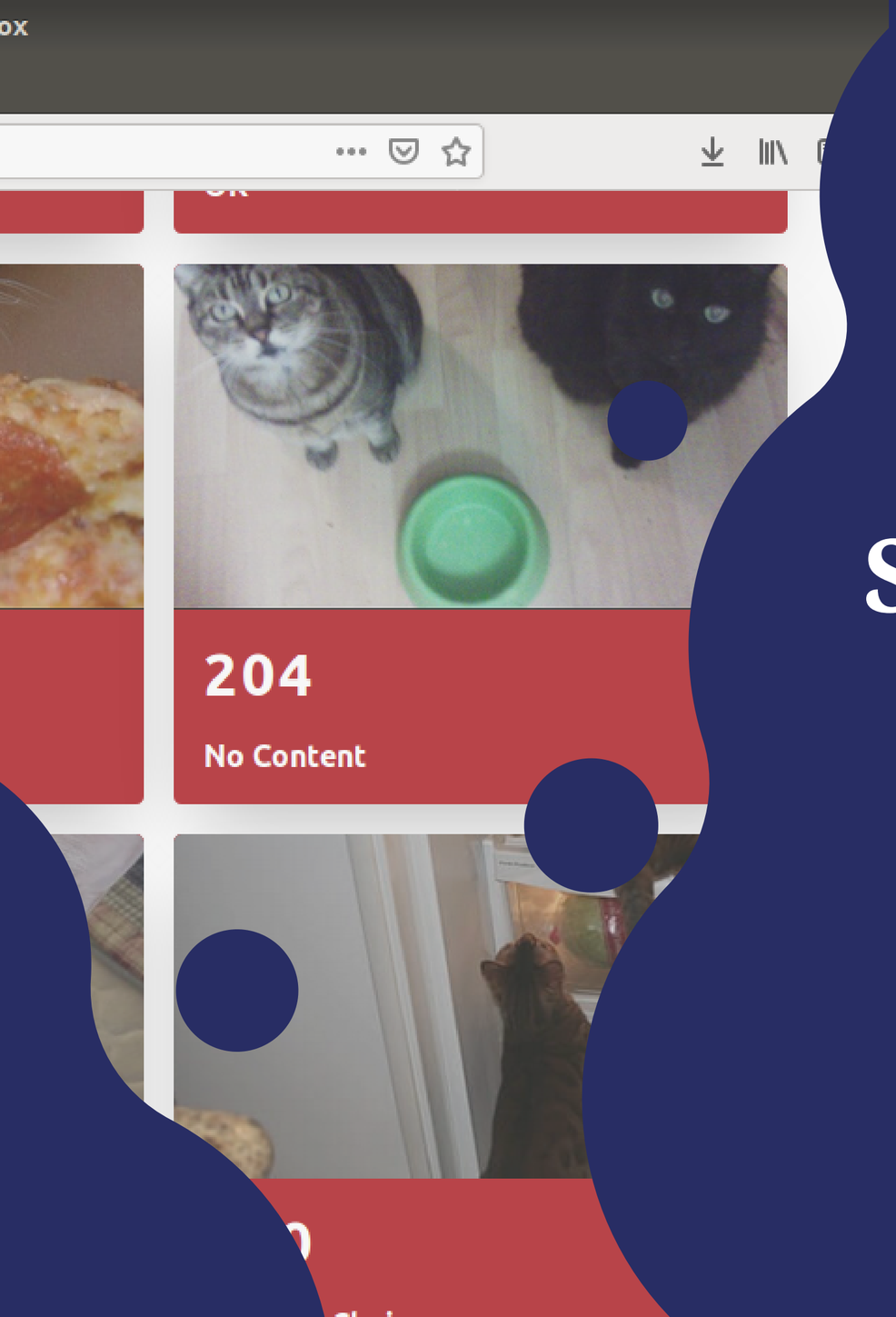
File Edit View Photo Help



200

OK

Rotate Crop Straighten Red-eye Adjust Enhance



PYTHON BRASIL [15]

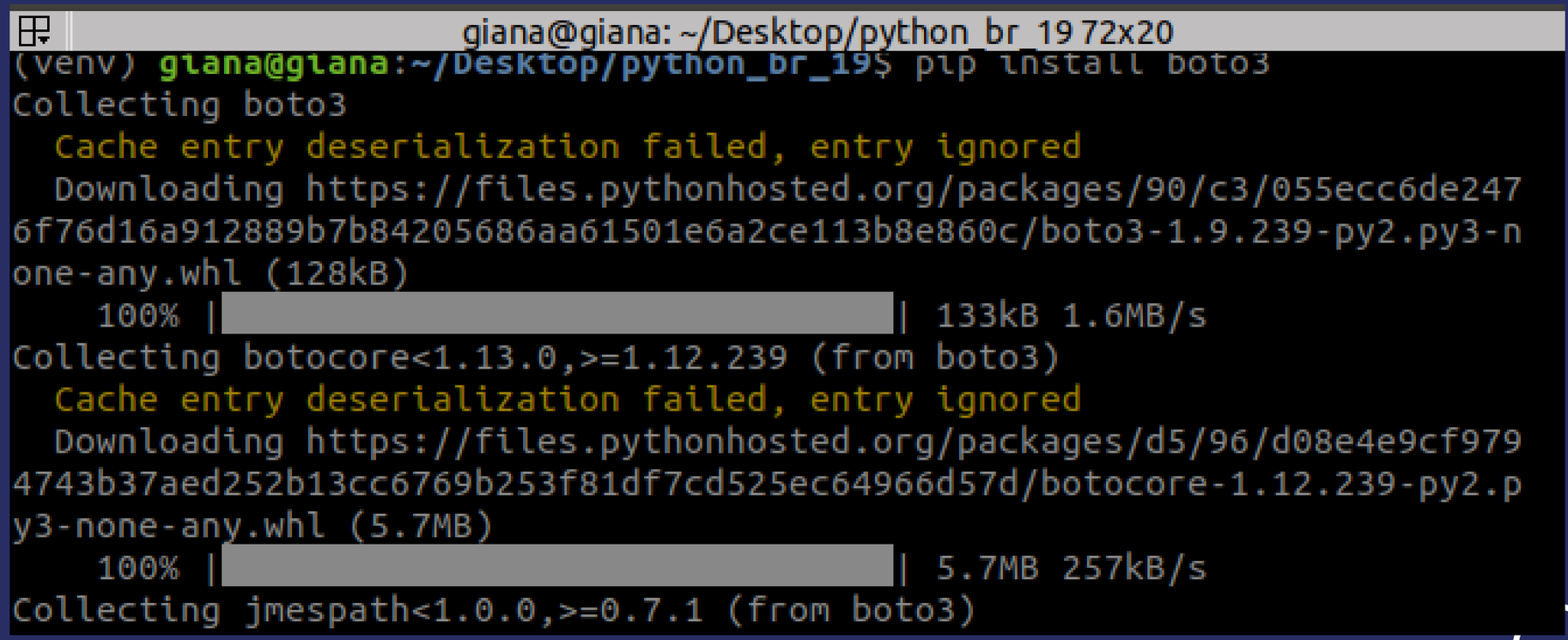
SALVANDO AS FOTOS NA AWS

Armazenamento

Conhecendo o AWS S3 e aprendendo sobre a biblioteca boto3

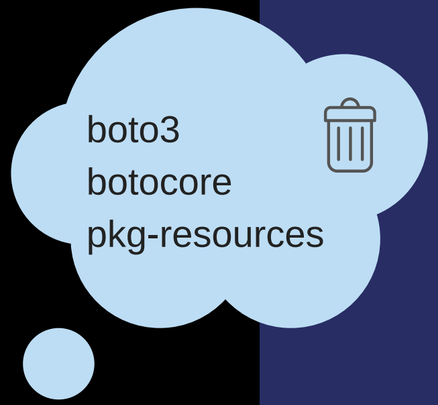
PYTHON BRASIL [15]

PYTHON BRASIL [15]



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$
```



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

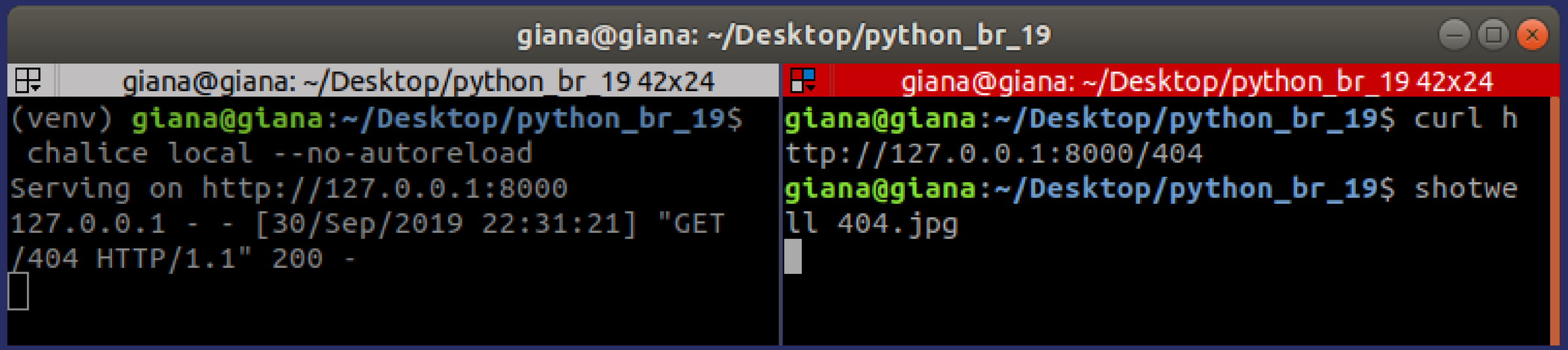
```
(venv) giana@giana:~/Desktop/python_br_19$ cat .chalice/config.json
{
  "version": "2.0",
  "app_name": "python_br_19",
  "environment_variables": {
    "BASE_URL": "https://http.cat",
    "FILE_TYPE": "jpg"
  },
  "stages": {
    "dev": {
      "api_gateway_stage": "api"
    }
  }
}
(venv) giana@giana:~/Desktop/python_br_19$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

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]



PYTHON E

404.jpg (~/Desktop/python_br_19) - Shotwell

File Edit View Photo Help

404

Not Found

Rotate Crop Straighten Red-eye Adjust Enhance

giana

(venv) gia
chalice l
Serving on
127.0.0.1
/404 HTTP/

19 42x24
$ curl h

$ shotwe

PYTHON BRASIL [15]

giana@giana: ~/Desktop/python_br_19 87x24

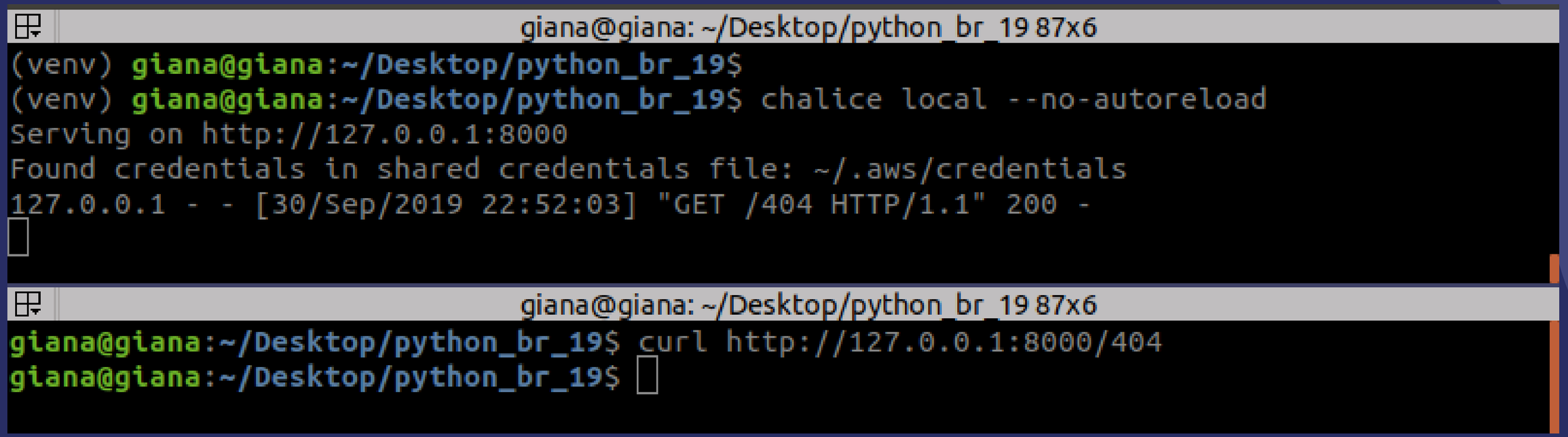
```
(venv) giana@giana:~/Desktop/python_br_19$ mkdir chalicelib
(venv) giana@giana:~/Desktop/python_br_19$ touch chalicelib/upload.py
(venv) giana@giana:~/Desktop/python_br_19$ nano chalicelib/upload.py
(venv) giana@giana:~/Desktop/python_br_19$ cat chalicelib/upload.py
import logging
import boto3
from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

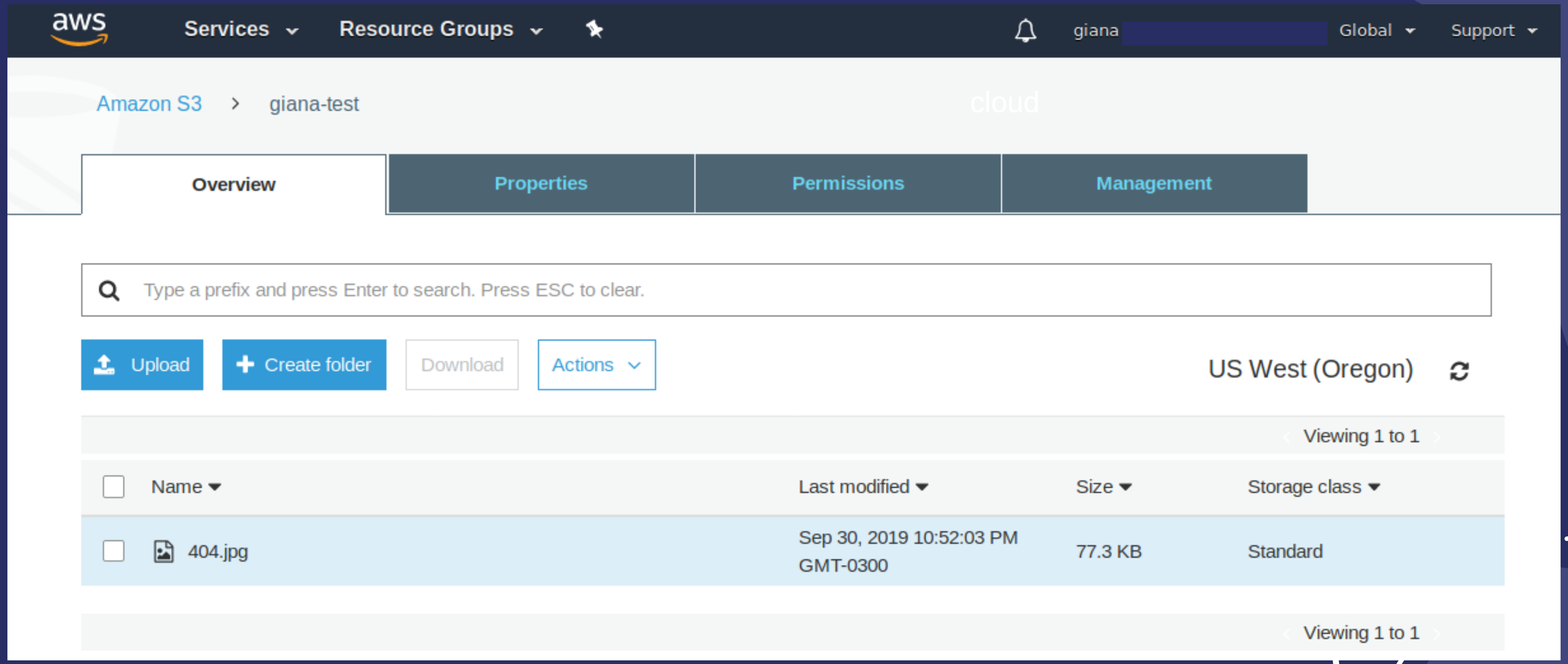
giana@giana: ~/Desktop/python_br_19 87x25

```
(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]



PYTHON BRASIL [15]

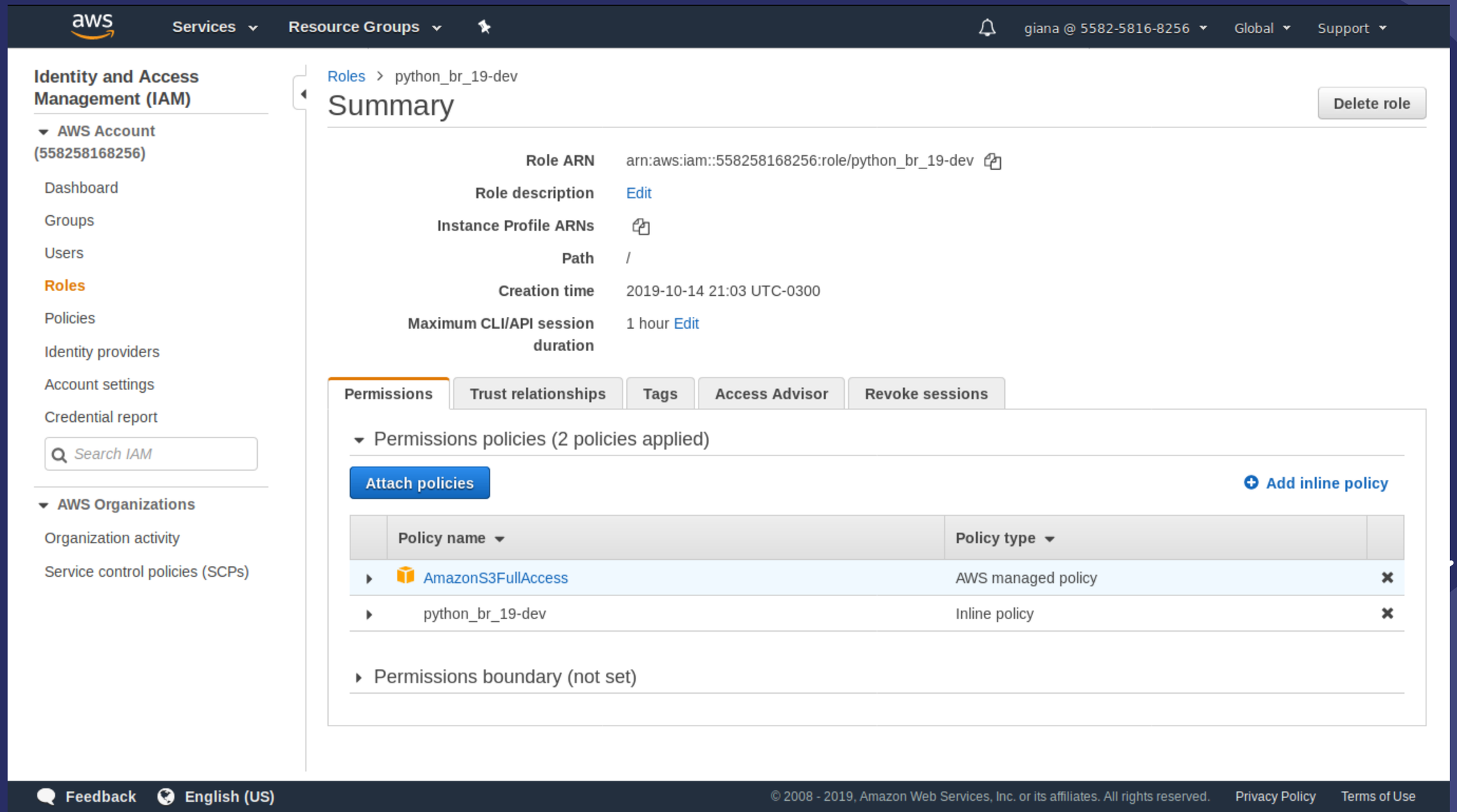


PYTHON BRASIL [15]

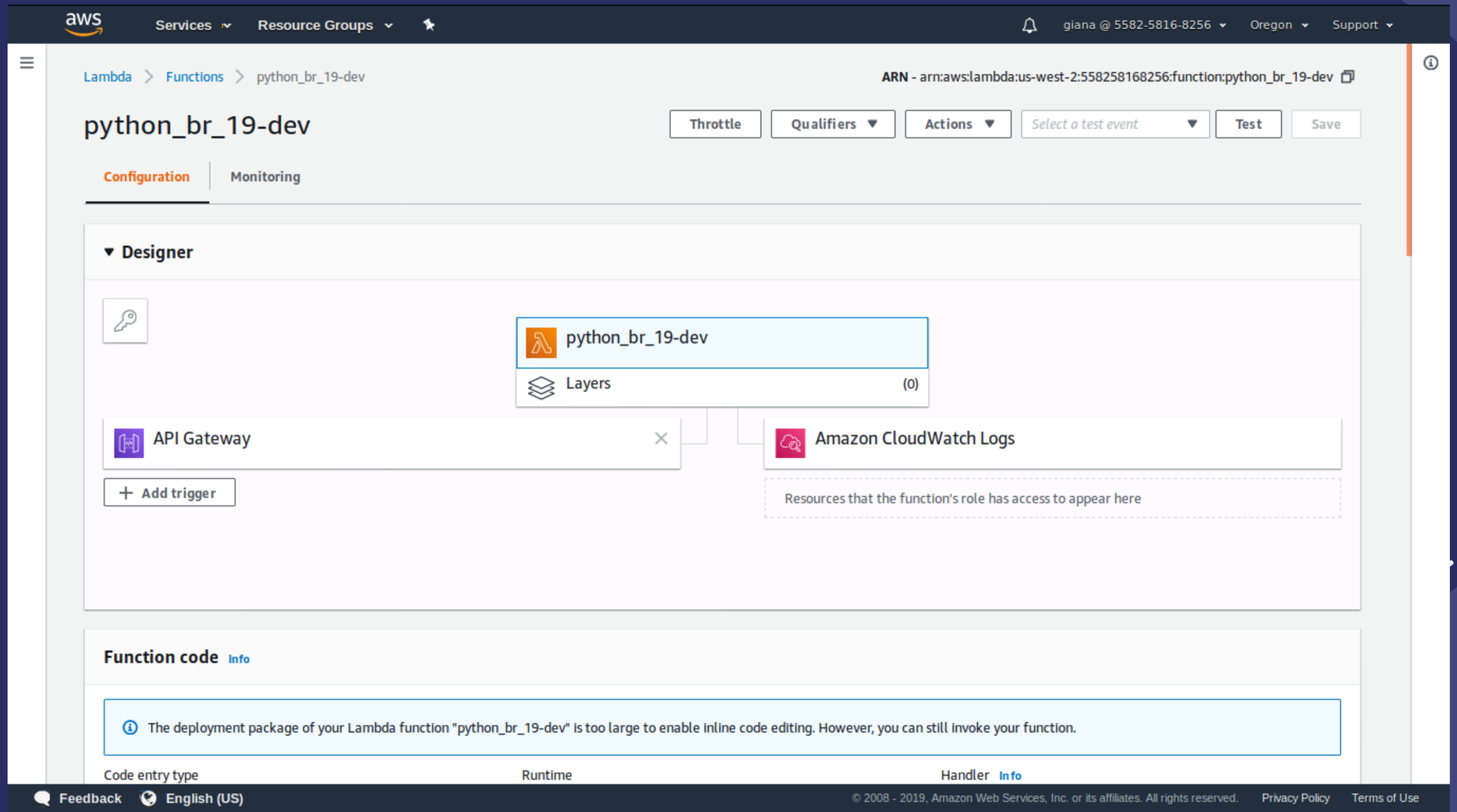
giana@giana: ~/Desktop/python_br_19 80x34

```
(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$
```

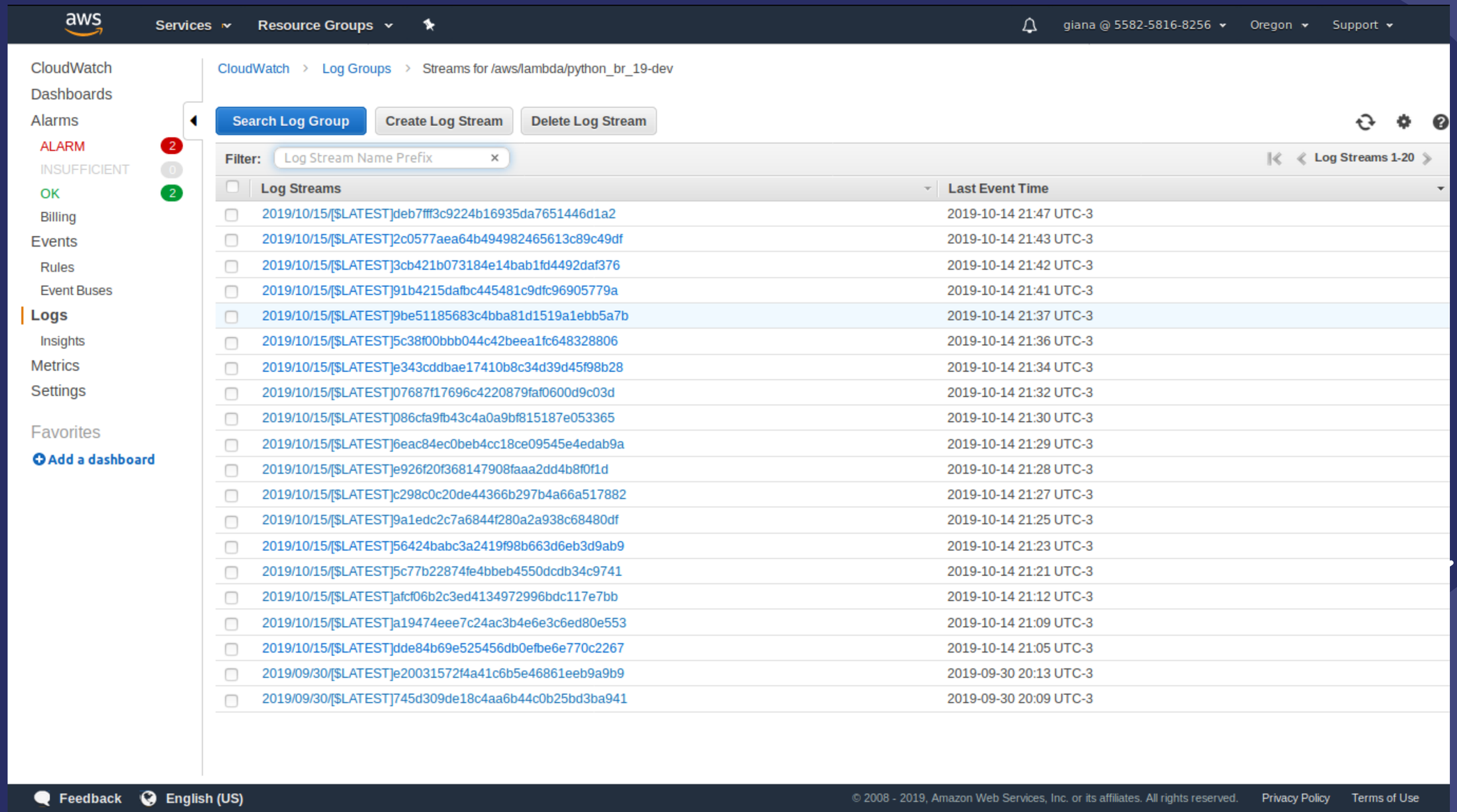
PYTHON BRASIL [15]



PYTHON BRASIL [15]



PYTHON BRASIL [15]



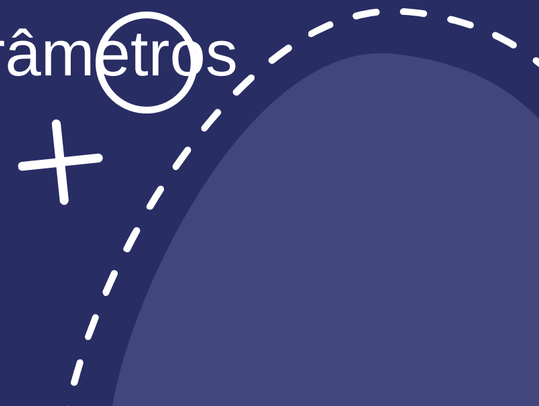
PYTHON BRASIL [15]

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.



PYTHON BRASIL [15]

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

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>. Acesso em: 13 set. 2019.

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%B5es-8606e7896c57>. Acesso em: 13 set. 2019.