



FRAMEWORK FULL STACK

Texto base

11

Computing e configuração de conta no Python Anywhere

Prof. André Nascimento Maia
Prof. Caio Nascimento Maia
Prof. Antonio de Oliveira Dias

Resumo

Ambientes em nuvem são ótimas escolhas para aplicações modernas e se encaixam muito com a abordagem full stack. Neste texto, é apresentado como utilizar duas Platform as a Service (Python Anywhere e Vercel) como ambiente de produção para uma aplicação de front-end usando Next.js e outra aplicação de back-end usando Python e Flask.

11.1. Introdução

Uma *web application* criada para ser acessada por milhares de usuário não tem muita utilidade se não estiver disponível em um ambiente escalável e elástico para que todos consigam acessá-lo. Atualmente, aplicações são disponibilizadas em ambientes de nuvem.

Na construção de uma aplicação utilizando uma abordagem *full-stack* é necessário se valer o máximo possível de serviços prontos para o ganho de eficiência.

Neste texto, discutiremos sobre os ambientes de produção tanto para uma aplicação de *front-end* utilizando Next.js quanto de *back-end* utilizando Python e Flask, para um e-commerce fictício.

11.2. Computação em nuvem

Computação em nuvem é um termo utilizado para quando precisamos que recursos computacionais estejam disponíveis sob demanda. Diferentemente do modo clássico de comprar máquinas físicas e instalá-las nos ambientes e *data centers* adequados, a computação em nuvem permite que esses recursos sejam alugados

temporariamente. Obviamente que os recursos computacionais não são infinitos, mas empresas como Amazon Web Service e Google Cloud Platform mantêm um grande parque de *data centers* e administram e distribuem esses recursos conforme a demanda de seus clientes.

Recursos em nuvem geralmente estão disponíveis em 3 formas diferentes: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) ou *Software as a Service* (SaaS). IaaS permite grande flexibilidade e acesso em nível mais baixo, permitindo a configuração de máquinas virtuais e, por exemplo, acesso de nível administrativo para essas máquinas. Porém, criar grandes infraestruturas exige grande dedicação e tempo de desenvolvimento. Já PaaS não permite um nível de acesso tão baixo quanto IaaS, mas nos garantem grande eficiência e agilidade na disponibilização de ambientes computacionais. Por exemplo, para colocar uma *web application* e *web api* em produção, utilizando IaaS com um grande nível de qualidade e que suporte milhares de usuários seria necessário configurar toda a infraestrutura de máquina de forma escalável e disponível. Componentes como: Domain Name Server (DNS), Web Application Firewall (WAF), Content Delivery Network (CDN), Load Balancers, API Gateway e cluster de Kubernetes. Todos esses componentes utilizam uma grande quantidade de máquinas virtuais com redundância adequada entre áreas e *data centers* para suportar uma grande carga e garantir alta disponibilidade. Quando utilizamos uma solução PaaS, tudo isso geralmente já está configurado e pronto para ser utilizado em poucos cliques e procedimentos. SaaS já é um nível de abstração muito mais alto. Alguns exemplos de SaaS são Google Drive, Sales Force, Office 365, e vários outros.

No contexto de um e-commerce fictício, vamos utilizar duas PaaS que oferecem serviços em níveis gratuitos para poucos acessos em versões hobbistas. São Python Anywhere e Vercel.

11.2.1. Python Anywhere para back-end

Python Anywhere é uma plataforma de desenvolvimento baseada na nuvem que permite aos programadores escrever, testar e implantar aplicativos Python sem a necessidade de configurações complexas de servidores.

As etapas para rodar uma aplicação no Python Anywhere são simples e serão apresentados nas imagens a seguir.

A primeira etapa é criar um usuário e fazer login no domínio: <https://www.Python Anywhere.com/login/> - Figura 11.1 a seguir:

Figura 11.1. Login no Python Anywhere

Fonte: do Autor, 2023.

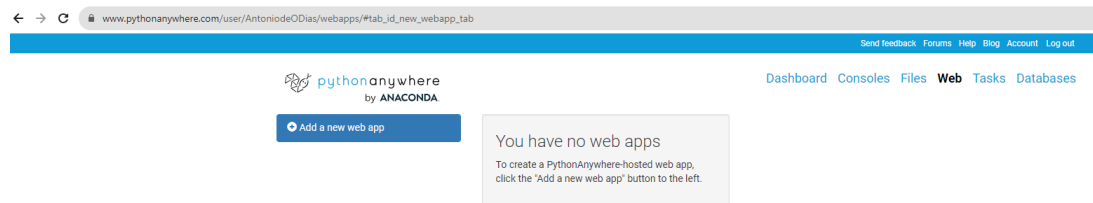
Após o login, o sistema redireciona para a página de dashboard, como na Figura 11.2, a seguir:

Figura 11.2. Dashboard do Python Anywhere

Fonte: do Autor, 2023.

Em seguida deve ser feita a criação da aplicação web. Para isso, basta clicar na opção Web da console no canto superior direito. O sistema redireciona o usuário para a página de criação de aplicação, como mostra a Figura 11.3.

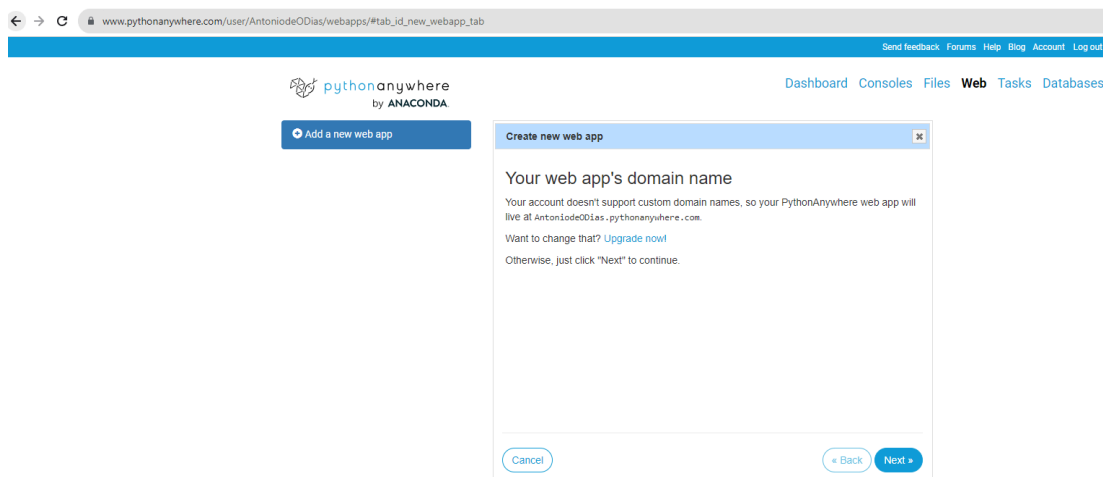
Figura 11.3. Console do Python Anywhere



Fonte: do Autor, 2023.

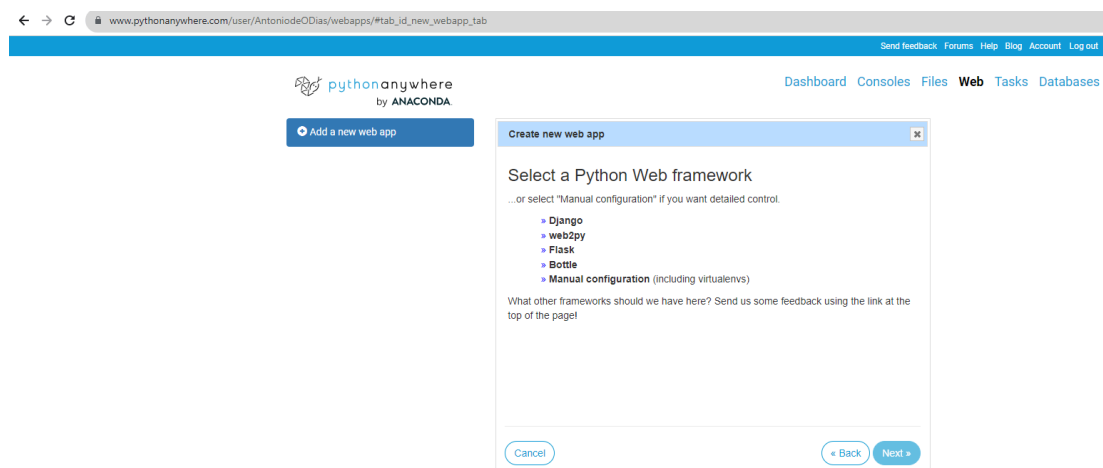
O usuário deve clicar na opção Add new Web App na parte superior esquerda e, em seguida, clicar em “Next”. Veja na Figura 11.4. e 11.5., respectivamente.

Figura 11.4. Criação de aplicação no Python Anywhere



Fonte: do Autor, 2023.

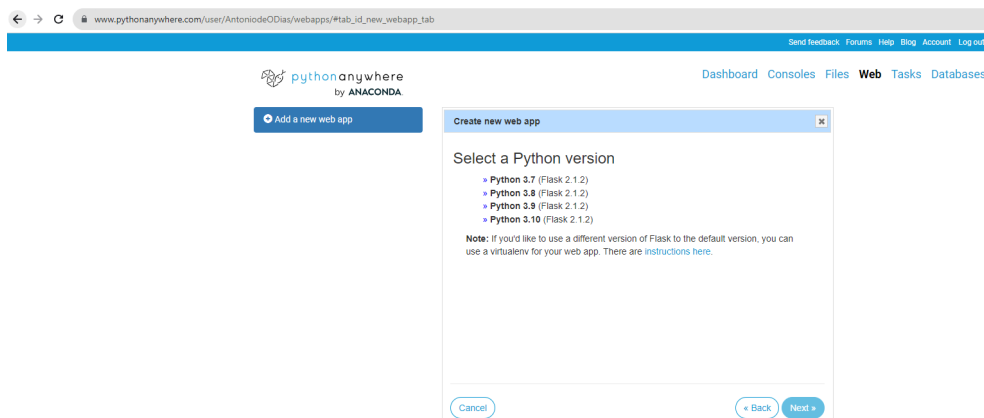
Figura 11.5 seleção de linguagem no Python Anywhere



Fonte: do Autor, 2023.

Escolha a opção Flask (no exemplo foi escolhido a versão 3.8.) - Figura 11.6. abaixo.

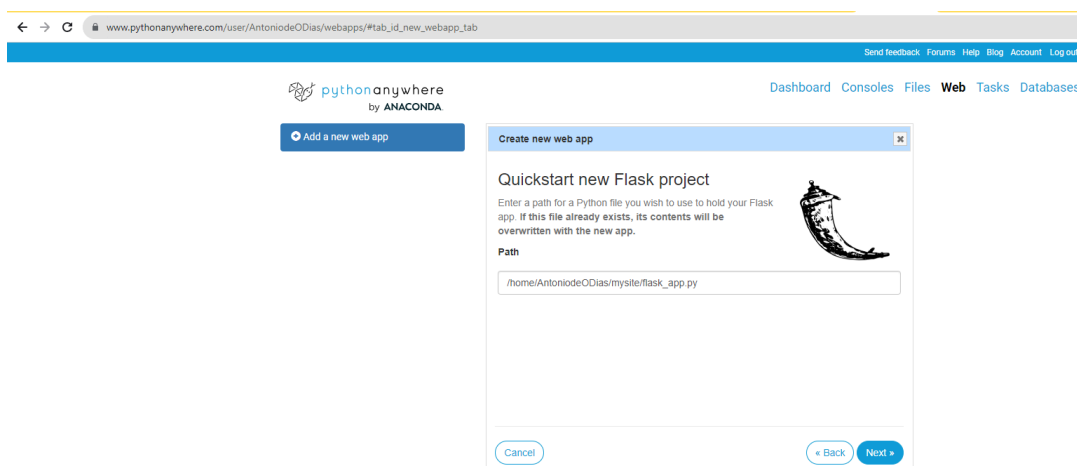
Figura 11.6. Seleção da versão do python no Python Anywhere



Fonte: do Autor, 2023.

Nesta etapa, deve-se ver o nome do arquivo para a aplicação Flask - o nome pode ser editado caso seja necessário. Clique em “Next” e o sistema criará a aplicação web - Figura 11.7. a seguir:

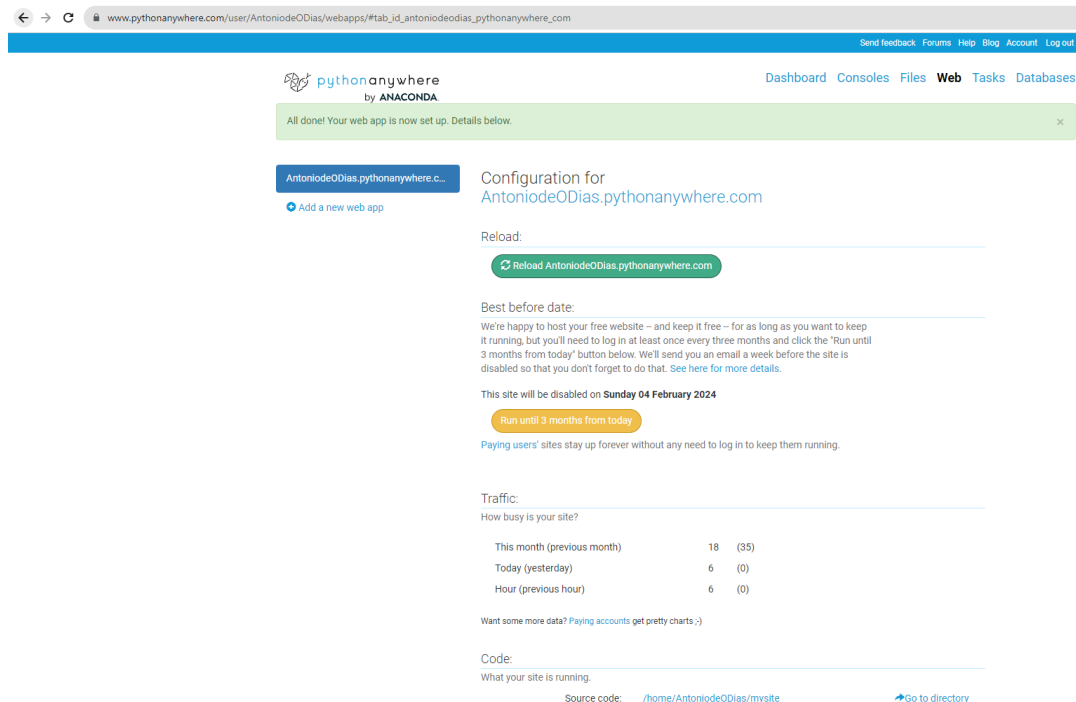
Figura 11.7. Nome do arquivo no Python Anywhere



Fonte: do Autor, 2023.

A tela do console é novamente apresentada e desta vez temos o link da aplicação web, logs, opções para desativar a aplicação ou deletá-la. Veja na Figura 18.8.

Figura 11.8 console com aplicação web criada no Python Anywhere



Fonte: do Autor, 2023.

Código inicial de teste:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

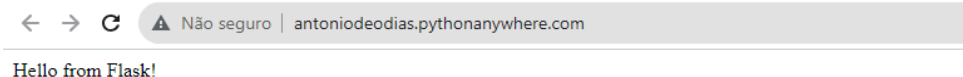
def hello_world():

    return 'Hello from Flask!'
```

Após esta configuração, deve-se clicar em Reload (em verde) para o sistema carregar a aplicação. Após estas etapas a aplicação está pronta para ser testada. Basta clicar no link abaixo da opção “Configuration for”.

No exemplo da Figura 11.9, o link está com o nome “AntoniioDias.Python Anywhere.com”. Assim podemos abrir a aplicação e verificar o seu funcionamento.

Figura 11.9 aplicação web rodando no Python Anywhere



A screenshot of a web browser window. The address bar shows a warning icon, the text 'Não seguro' (Not secure), and the URL 'antoniodeodias.pythonanywhere.com'. Below the address bar, the page content displays 'Hello from Flask!'.

Hello from Flask!

Fonte: do Autor, 2023.

Com isso temos a aplicação rodando no servidor sem qualquer custo.

Neste mesmo site podemos ver os logs ou editar os arquivos da aplicação.

Figura 11.10 configuração e logs do Python Anywhere

← → ↻ www.pythonanywhere.com/user/AntoniodeODias/webapps/#tab_id_antoniodeodias.pythonanywhere.com

Code:
What your site is running.

Source code: </home/AntoniodeODias/mysite> [Go to directory](#)
Working directory: </home/AntoniodeODias/> [Go to directory](#)
WSGI configuration file: </var/www/antoniodeodias.pythonanywhere.com/wsgi.py>
Python version: 3.8 [Python version](#)

Virtualenv:
Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.
[Enter path to a virtualenv, if desired](#)

Log files:
The first place to look if something goes wrong.

Access log: [antoniodeodias.pythonanywhere.com.access.log](https://antoniodeodias.pythonanywhere.com/access.log)
Error log: [antoniodeodias.pythonanywhere.com.error.log](https://antoniodeodias.pythonanywhere.com/error.log)
Server log: [antoniodeodias.pythonanywhere.com.server.log](https://antoniodeodias.pythonanywhere.com/server.log)
Log files are periodically rotated. You can find old logs here: </var/log>

Static files:
Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
Enter URL	Enter path	

Security:
An HTTPS certificate is needed so that people can access your site securely. We automatically provide a certificate for [AntoniodeODias.pythonanywhere.com](#).

HTTPS certificate: [Automatically provided for this hostname](#)

You need to **Reload your web app** to activate any changes made below.

Forcing HTTPS means that anyone who goes to your site using the insecure http URL will immediately be redirected to the secure https one. [More information here](#).

Fonte: do Autor, 2023.

Lembrando que no Python Anywhere só é possível fazer o deploy de uma aplicação web na opção grátis, para fazer deploy de uma nova aplicação, deve ser apagada a aplicação anterior ou migrar a conta para uma opção paga.

Código do Exemplo:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def nao_entre_em_panico():
    qtdTotal = 100
    primos = "1,2,"
    candPrimo = 3
    qtdEncontrados = 2
    ehPrimo = 1

    while qtdEncontrados < qtdTotal:
        for i in range(2, candPrimo):
            if candPrimo % i == 0:
                ehPrimo = 0
                break

        if ehPrimo == 1:
            primos = primos + str(candPrimo) + ","
            qtdEncontrados += 1
            if qtdEncontrados % 10 == 0:
                primos += "<br>"
            ehPrimo = 1
            candPrimo += 2

    return primos

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

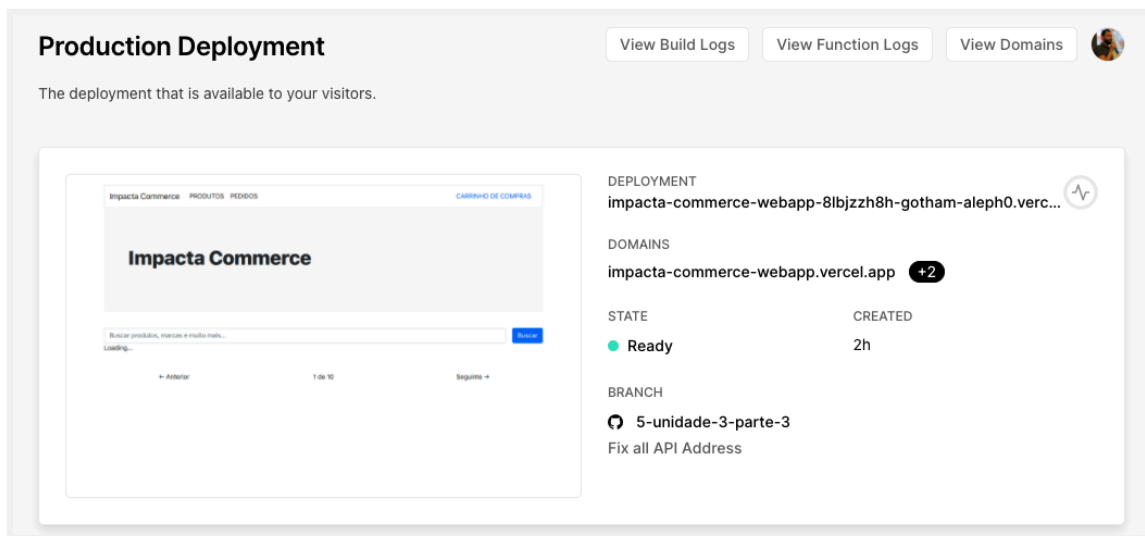
11.2.2. Vercel para front-end

A Vercel é uma plataforma em nuvem que tem como objetivo permitir que os times de *front-end* executem o seu melhor trabalho, ou seja, utilizem a plataforma como serviço e se concentrem em desenvolver código para as aplicações em *front-end* sem investir tempo com infraestrutura.

O processo para entregar uma aplicação no ambiente da Vercel é também tão simples quanto o processo da Python Anywhere. Basta fazer o login e permitir o acesso a um repositório Git do Github, por exemplo. Depois de selecionar o repositório, é necessário informar algumas configurações básicas como: nome do projeto; *framework*, no nosso caso é o Next.js; e o diretório raiz da aplicação. Após informados os parâmetros, o próximo clique em Deploy e todo o processo se inicia automaticamente.

O processo, quando finalizado com sucesso, redireciona o usuário para a visão geral do entrega em produção conforme a Figura 11.11.

Figura 11.11. Resultado de entrega em produção na Vercel.



Fonte: autores, 2022.

Assim como no Python Anywhere a Vercel disponibiliza um nome de domínio automaticamente baseado no nome do seu projeto. Neste caso, o endereço disponibilizado é o <<https://impacta-commerce-webapp.vercel.app>>.

11.3. Conclusões

Utilizar *Platform as a Service* (PaaS) se encaixa muito com a filosofia e abordagem do desenvolvimento de aplicações utilizando uma abordagem *full stack*. Esse caminho garante grande eficiência e uma qualidade muito alta com pouco trabalho. Obviamente os custos aparentam ser altos, porém, olhando bem de perto, o custo para manter uma grande equipe de infraestrutura e todo o maquinário necessário para ter a mesma qualidade é tão elevado quanto quando utilizamos uma PaaS como Python Anywhere ou Vercel.

Referências

Gunicorn. **WSGI server — Gunicorn 20.1.0 documentation.** Gunicorn - WSGI server, 2022. Disponível em: <<https://docs.gunicorn.org/en/latest/index.html>>. Acesso em 09 fev. 2022.

Anaconda. **Sobre Python Anywhere, uma empresa Anaconda.** Python Anywhere, 2023. Disponível em: <<https://www.Python Anywhere.com/>>. Acesso em 09 nov. 2023.