Tutorial de Deploy por aí é o que não falta, a maioria em inglês. Esse que estou criando é pra engrossar o caldo de deploys em português. Esse é um Guia Definitivo Rápido, ou não tão rápido, para fazer Deploy Django com Python 3. É um deploy para Kids.

A dificuldade de fazer um deploy reside nos detalhes. No fundo é fácil se você está familiarizado com as partes envolvidas. Você precisa saber fazer uma autenticação ssh, estar acostumado com a linha de comando, conhecer linux, saber configurar o

projeto, entender o que é servir arquivos estáticos, gunicorn…. tá, tá… nunca é fácil e muito menos rápido, justamente por isso criaram um monte de ferramentas pra deploy. E hoje com Ansible, Docker e *whatever kids are using these days* fica fácil fazer o deploy mas muito abstrato entender o funcionamento.

Em alguns anos esse post será obsoleto pra sempre, com serverless e tudo mais acho que pouca gente vai querer saber como fazer um deploy django dessa forma. Mas, mesmo assim, se ajudar uma pessoa já está bom. Será um tutorial **Old-Style**.

O Servidor

Vou imaginar que você não tem um servidor, não tem uma conta na AWS, nem DigitalOcean, nem Linode, nada… Você pode criar uma conta em um deles, lançar uma máquina com as configurações que quiser (Na AWS é tudo mais complicado para quem não está acostumado, se for sua primeira vez, prefira outro).

Para esse tutorial estou falando de Ubuntu 16.04, que é o servidor que você mais vai ver por aí nesse momento nesses serviços. Você pode escolher um Debian qualquer também.

Configuração inicial

Configure o timezone do servidor

```
sudo locale-gen --no-purge --lang pt_BR
sudo dpkg-reconfigure tzdata
```

Atualize os pacotes:

```
sudo apt-get update
sudo apt-get -y upgrade
```

Instalando Python 3.6 no lugar do Python 3.5

Agora substitua o Python 3.5 instalado, pelo Python3.6 (O Ubuntu que indiquei, ele vem com Python 3.5.1)

```
sudo apt-get update
sudo add-apt-repository ppa:jonathonf/python-3.6
sudo apt-get install python3.6
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.5 1
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.6 2
```


Você pode escolher qual versão do Python o SO vai usar ao chamar python3 com:

```
sudo update-alternatives --config python3
```

e você se enrolar, dê uma olhada aqui.

nstale os requisistos do Sistema Operacional

Aqui tem alguns pacotes que eu sempre uso em um deploy.

```
sudo apt-get install python3-pip nginx supervisor git git-core libpq-dev python-dev
python-virtualenv
```

Seu projeto pode ter outros requirements do SO pra instalar.

VirtualEnvWrapper para o Python3

Eu gosto muito do VirtualEnvWrapper, acho simples de começar um virtualenv, e deixa todos os meus virtualenvs no mesmo lugar, mas isso é escolha pessoal, se você não gosta, faça como está acostumado

Você instala o virtualenvwrapper, depois define a pasta dos seus virtualenvs (WORKON_HOME).

Para usar com múltiplos Pythons você vai precisar definir VIRTUALENVWRAPPER_PYTHON. No caso estou usando sempre o padrão que defini para o comando python3. Isso não é um problema porque você pode criar um virtualenv depois apontando qual python aquele virtualenv vai usar.

```
sudo pip3 install virtualenvwrapper
echo 'export WORKON_HOME=~/Envs' >> ~/.bashrc
echo ‘export VIRTUALENVWRAPPER_PYTHON=`which python3`’ >> ~/.bashrc
echo 'source /usr/local/bin/virtualenvwrapper.sh' >> ~/.bashrc
source ~/.bashrc
```


:rie seu VirtualEnv apontando qual python aquele virtualenv irá usar

```
mkvirtualenv nome_venv --python=python3
```

) VirtualEnvWrapper é muito fácil, para entrar em um Virtualenv que você criou, ocê pode usar:

```
workon nome_venv
```

ara sair do virtualenv:

```
deactivate
```

Para **excluir** um virtualenv:

```
rmvirtualenv nome_venv
```

Gere as Chaves SSH para autenticar no GitHub

Você não quer (e nem deveria) escrever a senha pra fazer o *git pull* do seu projeto no servidor.

Gere as chaves SSH

```
cd ~/.ssh
ssh-keygen -t rsa -b 4096 -C "dev@email.com"
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
```

Veja e copie o conteúdo da sua chave pública (**id_rsa.pub**)


```
cat ~/.ssh/id_rsa.pub
```

)epois entre no seu github, em Settings > SSH and GPG Keys. Clique em New SH Key, dê um nome pra essa chave, como (“chaves do servidor teste”) e em Key ole o conteúdo da chave pública id_rsa.pub

Faça o clone do seu projeto Django

:opie o link SSH do Github para fazer o clone, no caso estou usando um projeto ue encontrei agora pra exemplo

```
git clone git@github.com:kirpit/django-sample-app.git
```

Entre na pasta do projeto e instale os requirements do projeto.

Lembre-se de estar no virtual env correto.

```
cd django-sample-app/
pip install -r requirements.txt
```

Agora faça as alterações que forem necessárias para o deploy django, como criar um arquivo de settings_local, banco de dados, ou qualquer outra coisa específica do seu projeto.

Depois de tudo certo, você ainda precisa rodar as migrações e gerar os arquivos estáticos (se estiver usando)

```
python manage.py migrate
python manage.py collectstatic
```

Configurando o NGINX

Nginx, assim como o Apache, tem um mundo inteiro só deles, nesse momento você precisa conhecer o básico.

Existe um diretório /etc/nginx/sites-available/ onde ficão os arquivos de configuração de sites disponíveis para o nginx servir e existe o diretório /etc/nginx/sites-enabled/ que é a mesmíssima coisa, mas o que estiver aqui Nginx estará servindo mesmo.

ormalmente você cria o arquivo de configuração no sites-available/ e cria um link imbólico para o sites-enabled/

lós vamos fazer isso. Primeiramente, vou excluir o site default do nginx

```
sudo rm /etc/nginx/sites-enabled/default
```

gora crie o arquivo de configuração para o seu site. (Se você não está acostumado om o VIM, use troque vi por nano)

```
sudo vi /etc/nginx/sites-available/meusite
```

No conteúdo do arquivo, coloque isto, mudando os caminhos necessários:

```
server {
 listen 80;
 access_log /home/usuario/logs/access.log;
 error_log /home/usuario/logs/error.log;

 server_name nome-site.com.br;

 location / {
 proxy_pass http://127.0.0.1:8000;

 #As proximas linhas passam o IP real para o gunicorn nao achar que sao acessos locais
 proxy_pass_header Server;
 proxy_set_header X-Forwarded-Host $server_name;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header Host $http_host;


 }
```

```
 location /static {
   alias /home/usuario/caminho_projeto/static/;
 }
}
```

Agora crie o link simbólico para o sites-enabled:


```
sudo ln -s /etc/nginx/sites-available/meusite /etc/nginx/sites-enabled/meusite
```

Reinicie o Nginx:

```
sudo service nginx restart
```

Se você configurou tudo direitinho até aqui, ao acessar o site você verá uma página com um erro 502 Bad Gateway do próprio nginx)

Isso acontece porque ainda não tem nada aqui http://127.0.0.1:8000

Vamos colocar o site pra rodar nessa porta pelo gunicorn

Configurando o Gunicorn

Alguém vivo até essa parte? Cansa não, falta pouco.

No seu virtualenv (lembra workon nome_env?) instale o gunicorn

```
pip install gunicorn
```

Na pasta do seu projeto crie um arquivo chamado **gunicorn_conf** com:

```
bind = "127.0.0.1:8000"
logfile = "/home/usuario/logs/gunicorn.log"
workers = 3
```

Agora se você rodar o gunicorn, você vai ver seu site rodando:

```
/home/usuario/Envs/nome_venv/bin/gunicorn projeto.wsgi:application -c gunicorn_conf
```

Mas o que você pretende fazer? Rodar esse comando dentro de um screen e ir embora? Não dá né! Então, você vai usar o Supervisor pra controlar o funcionamento do gunicorn.

Configurando o Supervisor

Crie o seguinte arquivo de configuração

```
sudo vi /etc/supervisor/conf.d/gunicorn.conf
```

Com o seguinte conteúdo:

```
[program:gunicorn]
command=/home/usuario/Envs/nome_venv/bin/gunicorn projeto.wsgi:application -c /home/us
directory=/home/usuario/projeto/projeto-django
user=usuario
autostart=true
autorestart=true
redirect_stderr=true
```

Depois é só avisar o supervisor que existe um novo processo que ele precisa controlar da seguinte forma:

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl restart gunicorn
```

E voilá! Um site rodando você terá!

Conclusão

Existem muito mais coisas envolvidas no processo de um deploy. Você precisa configurar um firewall, provavelmente precisará servir mais pastas estáticas, etc, etc, etc… Mas precisa começar por algum lugar.

Não acredito que fiz um post inteiro sem colocar nenhum gif no meio, então só pra terminar, **PRESTE ATENÇÃO** em **TODOS os caminhos** que eu coloquei acima, você vai ter que usar os seus próprios paths corretamente

Oops..

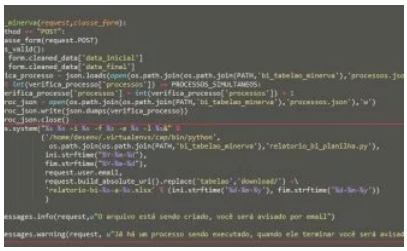
Also published on Medium.

Relacionado

Deploy de aplicações SSR utilizando o Supervisord

Se você está construindo aplicações que renderizam do lado do servidor, o famoso SSR, utiliznado React (com o next.js) ou Vue.js (com o nuxt.js), você vai precisar de alguma ferramenta para controle do processos quando for fazer o deploy. Eu escolhi fazer o deploy de aplicações SSR utilizando o Supervisord,…

Populando um campo novo não-nulo no Django



Tarefas demoradas de forma assíncrona com Django e Celery

Deixe uma resposta

Digite seu comentário aqui...

14 COMENTÁRIOS

Thiago

julho 29, 2017

Parabéns! Depois cria um mais avançado…

Responder a Thiago

ffreitasalves
julho 30, 2017

Valeu Thiago!

Estou pensando em talvez criar uma série, uma outra pessoa pediu no facebook pra falar de outros temas também, o que você acha que posso acrescentar no próximo?

Obrigado pela sugestão!

Responder a ffreitasalves

Thiago
agosto 8, 2017

Deploy automatizado, https, e backup do DB, eu acho 🙂

Responder a Thiago

ffreitasalves
agosto 8, 2017

Boa Thiago, vou escrever, valeu pela sugestão!

Responder a ffreitasalves

Thiago
agosto 8, 2017

Sugestões para artigo:
Cache
Otimização das querys
API com Django
Django com algum framework front-end do momento (Vue ou React) 🙂

Responder a Thiago

Marcelo Augusto
agosto 10, 2018

nossa o que tem o python e o django de facilidade tem o deploy de dificuldade absudo!! mas agradeço pela sua boa vontade em explicar

Responder a Marcelo

ffreitasalves
agosto 10, 2018

É Marcelo, sabendo os pormenores da pra ver que tem sim muita coisa pra fazer pra colocar no ar.
Mas, como eu disse, esse tutorial é old school.
Vou criar um simplificando o deploy com ansible.

Responder a ffreitasalves

Diego Muriel
dezembro 12, 2018

Olá! No exemplo de configuração do NGINX está faltando uma ‘}’ ao final do arquivo. Parabéns pelo artigo!

Responder a Diego

ffreitasalves
dezembro 12, 2018

Vc tem razão.
Adicionei, muitissimo obrigado

Responder a ffreitasalves

Thiago Brito
janeiro 30, 2019

Excelente, depois posta como fazer do modo fácil.. apesar, que isso já é facil

Responder a Thiago

Wagner Cateb
março 7, 2019

Muito bom mesmo! Já vi várias explicações mas você fez ótimos comentários esclarecendo o porque de algumas coisas, ou opções pessoais. Muito obrigado!!!

Responder a Wagner

diego oliveira
junho 11, 2019

Parabéns e obrigado por compartilhar seu conhecimento.
Esse gunicorn eu não conhecia, utilizo Apache e o WSGI do Django.

Responder a diego

Patterson Junior
julho 18, 2019

Show!

Responder a Patterson

ffreitasalves
julho 22, 2019

Você é um dos comentaristas mais antigos desse blog!

Responder a ffreitasalves

WEBMENTIONS

Deploy de aplicações SSR utilizando o Supervisord - Fernando Alves julho 22, 2019
…] é mais conhecido em um contexto geral, não só de quem é do mundo do node. Se você já viu meu uia para Deploy Django e Python 3, você já usou o […]

eploy for kids - Django & Python 3 - Fernando Alves julho 22, 2019
…] here is a lot of tutorials out there, especially in English. Here it goes another one. I wrote it iginally in Portuguese. […]

ONTEÚDO RELACIONADO POR TAG DEPLOY DJANGO GUNICORN NGINX PORTUGUÊS
YTHON PYTHON3 VIRTUALENV