Instalação e Configuração de Servidores

Prof. Diego Cirilo

Aula 13: Deploy de um sistema Django

Deploy

- *Implantar*
- Enviar seu sistema para um servidor de *produção*
- Desenvolvimento/Produção

Preparação

- Clone do seu *virtualenv*
- Na máquina de desenvolvimento:
  - `pip freeze > requirements.txt`
- No servidor:
  - `pip install -r requirements.txt`
- **ATENÇÃO**: nunca reutilize seu *venv*, crie um em cada máquina que usar.

Preparação

- Verifique que o *nginx* e o *MySQL* estão funcionando corretamente (aulas anteriores)
- Instale as dependências:

```
$ sudo apt-get install python3-pip python3-venv libmariadb-dev pkg-config
```

Preparação

- Copie a pasta do seu projeto Django para sua *home* (use o Filezilla ou Git)
- Re-crie o *venv* dentro da pasta do seu projeto e ative-o (perceba que o comando do *venv* é diferente do *Windows*):

```
python3 -m venv venv
source venv/bin/activate
```

- Instale as dependências do seu projeto:

```
pip install -r requirements.txt
```

- Instale também as novas dependências para o servidor

```
$ pip install mysqlclient gunicorn
```

Configure o banco de dados

- Acesse o MySQL como administrador via terminal, MySQL Workbench, phpMyAdmin, etc;
- Execute:

```
CREATE DATABASE <banco-do-app> CHARACTER SET 'utf8';
CREATE USER <user-do-app>;
GRANT ALL ON <banco-do-app>.* TO '<user-do-app>'@'localhost' IDENTIFIED BY '<senha-do-user-do-app>';
quit
```

> **Substitua o `<exemplo>` com suas informações**

Modifique as configurações

- No arquivo `settings.py`;
- Mudamos o modo `DEBUG` para `False`;
- Configuramos um `SECRET_KEY` específico para produção;
    - Nunca adicione o `SECRET_KEY` de prod. ao repositório!!
    - Documentação

Modifique as configurações

- Configure o banco de dados, o *SQLite* não é ideal para produção;
- Em `ALLOWED_HOSTS` colocamos os IPs e domínios do nosso servidor;
- É uma boa ideia manter os arquivos estáticos e *media* separados do código;
- Dessa maneira não precisamos liberar acesso ao nosso diretório;
- Sugestão: manter `media` e `static` em `/var/www/html/nomedoprojeto/`.

Modifique as configurações

```
SECRET_KEY = "minha chave super segura e longa"

DEBUG = False

ALLOWED_HOSTS = ['127.0.0.1', 'localhost', 'meusite.com', '111.111.111.111'] #exemplo!

...

DATABASES = {
    'default': {
         'ENGINE': 'django.db.backends.mysql',
         'OPTIONS': {
             'sql_mode': 'traditional',
         },
         'NAME': '<banco-do-app>',
         'USER': '<user-do-app>',
         'PASSWORD': '<senha-do-user-do-app>',
         'HOST': 'localhost',
         'PORT': '3306',
     }
}

...

STATIC_URL = "/static/"
STATIC_ROOT="/var/www/html/nomedoprojeto/static/"

MEDIA_URL="/media/"
MEDIA_ROOT="/var/www/html/nomedoprojeto/media/"
```

Arquivos estáticos

- Crie os diretórios necessários em `/var/www/html`
- Mude o proprietário do diretório para seu usuário:

```
sudo chown -R usuario:usuario
/var/www/html/nomedoprojeto
```

Backend de Email

- Podemos configurar um servidor de emails local;
- Porém hoje em dia qualquer servidor de email *novo* cai em filtros de *spam*;
- Podemos usar um servidor de emails *famoso* através do SMTP;
- *Simple Mail Transfer Protocol*.

Backend de Email

- Exemplo gmail

```
EMAIL_BACKEND = ‘django.core.mail.backends.smtp.EmailBackend’
EMAIL_HOST = ‘smtp.gmail.com’
EMAIL_USE_TLS = True
EMAIL_PORT = 587
EMAIL_HOST_USER = ‘seuemail@gmail.com’
EMAIL_HOST_PASSWORD = ‘seu TOKEN!! (ñ é senha)’
```

Migrations

- Faça as *migrations*

```
python manage.py makemigrations
python manage.py migrate
```

- Colete os arquivos estáticos para a pasta de arquivos estáticos

```
python manage.py collectstatic
```

- Caso necessário, crie também o *superuser* do seu sistema

```
python manage.py createsuperuser
```

- Execute o servidor de teste para saber se o sistema ainda sobe

```
python manage.py runserver
```

Configure o *gunicorn*

- *Gunicorn* - Unicórnio Verde: servidor HTTP para Python.
- Processa e encaminha as requisições entre o servidor web e a aplicação Python
- Disponibiliza um *socket* para que o *nginx* possa se comunicar com Django.
- Deve rodar como um *daemon* no sistema

Gunicorn Daemon e socket

- Crie um arquivo <projeto>_gunicorn.socket

```
sudo nano /etc/systemd/system/<projeto>_gunicorn.socket
```

- Com o seguinte conteúdo:

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/<projeto>_gunicorn.sock

[Install]
WantedBy=sockets.target
```

Gunicorn Daemon

- Crie um arquivo <projeto>_gunicorn.service

```
sudo nano /etc/systemd/system/<projeto>_gunicorn.service
```

- Com o seguinte conteúdo:

```
[Unit]
Description=gunicorn service
After=network.target
Requires=<projeto>_gunicorn.socket

[Service]
User=<user>
Group=www-data
WorkingDirectory=/home/<user>/<projeto-django>/
ExecStart=/home/<user>/<projeto-django>/venv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/<projeto-django>.sock config.wsgi:application

[Install]
WantedBy=multi-user.target
```

Gunicorn Daemon

- Habilite o *daemon*

```
sudo systemctl enable <app>_gunicorn.service
sudo systemctl start <app>_gunicorn.service
sudo systemctl status <app>_gunicorn.service
```

- Sempre que fizer alterações do seu sistema, reinicie os *daemons*

```
sudo systemctl daemon-reload
sudo systemctl restart <app>_gunicorn.service
```

Configurando o Nginx

- Crie um novo arquivo de configuração do nginx

```
sudo nano /etc/nginx/sites-available/<projeto-django>
```

- Adicione os seguintes conteúdos, alterando o que estiver entre `< >`:

```
server {
       listen 80;
       server_name 127.0.0.1;
       location = /favicon.ico {access_log off;log_not_found off;}

        location /static/ {
            alias /var/www/html/<projeto-django>/static/;
        }

        location /media/ {
            alias /var/www/html/<projeto-django>/media/;
        }

        location / {
            include proxy_params;
            proxy_pass http://unix:/run/<projeto-django>_gunicorn.sock;
        }
    }
```

Configurando o Nginx

- Habilite o site (desabilite o default se for necessário)

```
sudo ln -s /etc/nginx/sites-available/<projeto-django> /etc/nginx/sites-enabled
```

- Verifique se está tudo ok

```
sudo nginx -t
```

Se não houver nenhum erro, reinicie o nginx

```
sudo systemctl restart nginx
```

- A aplicação deve estar disponível.

Solucionando erros

- Leia o *status* do *nginx* com o `systemctl status`
- Leia o *log* de erros do *nginx* para mais informações usando:

```
sudo tail /var/log/nginx/error.log
```

Upload de Arquivos

- O *nginx* impõe um limite de 1 MB no tamanho dos uploads;
- Caso o usuário tente um upload maior, acontece o erro 413 (*Request Entity Too Large*);
- É possível aumentar usando a configuração `client_max_body_size tamanho`
- Ex. `client_max_body_size 10M`
- Podemos adicionar essa configuração em `/etc/nginx/sites-available/meusite`;
- Reiniciamos o servidor para aplicar as modificações.

Upload de Arquivos

- Exemplo `/etc/nginx/sites-available/meusite`

```
server {
...
    server_name _;

    client_max_body_size 10M;

...
    location / {
        ...
    }
...
}
```

Múltiplos projetos

- É possível rodar mais de um projeto em um mesmo servidor;
- Caso haja domínios configurados, é possível mudar o `server_name` no *nginx*;
- Ou usar `subdiretórios`, como `meusite.com/projeto1`, `meusite.com/projeto2`;
- Pode causar problemas caso as *urls* do sistema não estejam perfeitas!

Múltiplos projetos

- Exemplo (nginx)

```
location /projeto1/static/ {
    alias /var/www/html/projeto1/static/;
}

location /projeto1/static/ {
    alias /var/www/html/projeto1/media/;
}

location /projeto1/ {
    include proxy_params;
    proxy_set_header SCRIPT_NAME "/projeto1";
    proxy_pass http://unix:/run/projeto1_gunicorn.sock;
}
```

Atualizando o projeto

- Para atualizar do Git;

```
cd pasta-do-projeto
git pull origin main
source venv/bin/activate
pip install -r requirements.txt
python manage.py migrate
python manage.py collectstatic
sudo systemctl restart projeto_gunicorn.service
sudo systemctl restart nginx
```

Referências

- https://realpython.com/django-nginx-gunicorn/
- https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu
- https://docs.gunicorn.org/en/latest/deploy.html
- https://docs.djangoproject.com/en/5.1/howto/static-files/deployment/

Dúvidas?