Documentacao Dr. Down

Release 0.1

EPS/MDS

Apr 24, 2018

CONTENTS

Conteúdo:

CHAPTER

ONE

INSTALANDO

Observação: Recomendamos, e o processo abaixo foi executado, uma máquina Ubuntu/Linux com a distribuição `Ubuntu 16.04.4 LTS`.

O primeiro passo é fazer o clone do projeto pelo GitHub (tenha certeza de ter o `git` instalado em sua máquina):

```
$ git clone https://github.com/fga-gpp-mds/2018.1-Dr-Down.git
```

Para rodar a aplicação tenha certeza de ter algumas dependências instaladas. Existem dois scripts que auxiliam o você nessa etapa. Para fazer a instalação basta rodar (partindo que está na pasta base após clone) os seguintes `shell scripts`:

```
$ sudo bash utility/install_os_dependencies.sh arg
$ sudo bash utility/install_python_dependencies.sh
```

No primeiro script será necessário dizer qual é o `arg` da operação que deseja fazer, as funções disponíveis são:

- list
- help
- install
- upgrade

Certifique de ter instalado também:

- docker
- docker-compose

E por fim, agora para rodar a aplicação basta rodar o seguinte comando no seu terminal:

```
$ docker-compose -f local.yml up --build
```

Com isso as imagens serão baixadas e geradas na sua máquina e você poderá acessar a aplicação pelo seu navegador no endereço `127.0.0.1:8000`.

Observação: Caso deseje parar os containers basta usar a combinação **CTRL+C** no terminal que está rodando a aplicação, ou, caso esteja rodando em *backgroud* executar o comando:

```
$ docker-compose -f local.yml down
```

1.1 Configurando seu usuário:

- Para criar um **conta de usuário normal**, vá em Criar Conta e preencha os campos. Assim que você submeter suas informações, você verá uma página de "Verificar seu endereço de E-mail". Vá no seu terminal, no seu console você verá uma mensagem de email de verificação. Copie o link para seu negador. Agora o E-mail do usuário deve ser verificado e pronto para ser usado.
- Para criar uma **conta super usuário**, use esse comando:

```
$ docker-compose -f local.yml run --rm django python manage.py createsuperuser
```

Por conveniência, você pode manter o seu usuário normal logado no Chrome e seu super usuário (administrador) logado no Firefox (ou similar), assim você consegue ver como o site se comporta em ambos usuários.

Observação: O tutorial acima mostra como instalar e rodar a máquina em ambiente de desenvolvimento, para ambiente de produção, verifique o deploy.

CHAPTER

TWO

DEPLOY

2.1 Homologação (*staging*)

O deploy em ambiente de testes foi feito com o auxílio de algumas documentações, principalmente a que pode ser acessada pelo projeto do cookiecutter, na seção Deploy com Docker, atentado as particularidades, dado ao fato desse ser o ambiente de homologação da ferramenta.

Para o deploy é necessário:

- Um droplet do Digital Ocean (Ubuntu 16.04.4 LTS);
- Um domínio (foi usado o provedor Freenom);

Observação: A equipe decidiu por usar as máquinas (droplets) do *Digital Ocean* para o deploy, mas uma máquina com um IP público já é apropriada para o processo.

2.1.1 Configurando o domínio

Com uma máquina em mãos o primeiro passo é configurar o seu domínio para que o nome o qual você registrou possa apontar para a máquina (IP) a qual você irá fazer o deploy da aplicação. Neste passo foi seguida a documentação do *Digital Ocean*, que apresenta como configurar o DigitalOcean DNS. Em poucos passos o que foi feito:

- Adicionado o domínio registrado no *Digital Ocean*;
- Adicionado os registros do DNS;
- E configurado os *Nameservers* no site do registro do domínio.

Com isso sua máquina já estará mapeada no domínio registrado e já será possível acessá-la via URL do domínio.

2.1.2 Fazendo o deploy na máquina

Para fazer o deploy na sua máquina, que agora já tem o IP registrado num domínio, se atente em ter as seguintes dependências instaladas:

- Nginx
- Docker
- Docker Compose

Acesse sua máquina via SSH e faça os passos de instalação.

Agora é necessário configurar o Nginx para que ele possa mapear as portas e servir os arquivos da sua instalação. Aconselhamos este tutorial ou se preferir esta documentação, que é o exemplo encontrado no site oficial do Nginx. Mas é importante ressaltar que ao instalar o *nginx* na sua máquina, o mesmo cria um arquivo *default* de configuração

e a ferramenta do Dr. Down já trabalha automaticamente com essa porta, logo com a configuração *default* a aplicação deve rodar sem problema, caso precise mudar ou especificar algo diferente, leia os documentos supracitados. Com isto certifique que:

- Tem as dependências instaladas;
- As instruções de instalação do software foram seguidas;
- Os *containers* docker estão rodando em máquina (status *Up*).

Caso todos os passos acima tenham sido feitos, agora, basta acessar a URL do seu domínio que o site Dr. Down deverá estar disponível para o ambiente de homologação.

Observação: Caso enfrente algum problema, sempre verique os logs de máquina, e não hesite em nos contatar via Issue no GitHub, faremos o possível para ajudar.

Observação: Os comandos de máquina como *makemigrations*, *migrate*, *shell*, *createsuperuser* e etc, devem ser feitos na máquina host da aplicação, assim como a verficação de usuários por e-mail deverá ser feita como o padrão em desenvolvimento (pegar o link de verificação gerado pelo *output* do console da aplicação).

2.1.3 Deploy Contínuo

A aplicação Dr. Down tem um pipeline de deploy contínuo para o ambiente de homologação, ele é executado junto com os testes e a *build* nos *jobs* do Travis CI, caso queira manter o mesmo pipeline usado por nós modifique o arquivo `.travis.yml` que fica na pasta raíz do projeto, se atentando em configurar de acordo com as suas necessidades os seguintes parâmetros:

- Usuário do Docker Hub;
- Senha de acesso ao Docker Hub;
- IP da máquina de deploy;
- Senha de acesso a máquina de deploy.

Com esses parâmetros devidamente configurados, o deploy contínuo deve funcionar normalmente no seu contexto.

2.2 Produção (*production*)

O deploy em ambiente de produção ainda está em fase de teste, portanto ainda não será documentado aqui.

CHAPTER

THREE

TESTES

Nesta seção iremos demonstrar alguns processos de teste da aplicação Dr. Down, além de mostrar como você pode escrever os seus próprios testes. A aplicação Dr. Down vem com duas suítes de teste python instaladas e configuradas, sendo uma o py.test e a outra o coverage. Antes de explicar por menor as suítes de teste vamos ver como são feitos os testes em python/Django.

3.1 Testes python/Django

O Django provê ao desenvolvedor diversas formas de testar sua aplicação, diversas ferramentas *third-party* também estão disponíveis para auxiliar na tarefa dos testes. Testar uma aplicação web é uma tarefa complexa, porque uma aplicação web é feita por diversas camadas lógicas – vai do nível de uma requisição HTTP, a uma validação e processamento de um formulário, até a renderização de um template. Com a execução de teste em Django e dispondo de diversas utilidades, você pode simular requisições, inserir informações testes, inspecionar saídas de sua aplicação e geralmente verificar se seu código está fazendo o que ele deveria fazer. Por padrão os testes Django devem estar na pasta `tests/` do seu app, com isso as ferramentas reconhecem como teste do seu código e são executadas. Algumas documentações de referência são:

3.1.1 Escrevendo e Executando Testes

3.1.2 Testes Unitários

3.2 Testando Dr. Down

3.2.1 py.test:

Para apenas rodar a suíte de testes com o py.test basta executar o seguinte comando:

```
$ docker-compose -f local.yml run --rm django py.test
```

Observação: No próprio terminal será mostrado o *output* dos testes rodados. Outras configurações, *flags* e modos de uso do py.test podem ser verificadas na documentação do py.test.

3.2.2 coverage:

O Coverage é uma ferramenta auxiliar que permite gerar relatórios e verifcar a cobertura de código a partir dos testes executados. Tanto que para isso ele usa o auxílio do py.test, no entanto, há forma de executar apenas o coverage, ou até usar de outros *frameworks* de teste, mas nesse projeto utilizamos ele em conjunto com o py.test, sua execução se dá da seguinte forma:

```
$ docker-compose -f local.yml run --rm django coverage run -m py.test
$ docker-compose -f local.yml run --rm django coverage html
$ firefox htmlcov/index.html
```

Com estes comandos o coverage irá executar o testes com auxílio do py.test, vai utilizar os dados dele como input para gerar a cobertura de testes e mostrar ao usuário.

Observação: Será gerada uma pasta `htmlcov/` que conterá o `index.html`, este arquivo contem o relatório extraído dos testes rodados, no exemplo acima foi utilizado o navegador *Mozilla Firefox* para a abertura do arquivo HTML gerado. Outras configurações, *flags* e modos de uso do coverage podem ser verificadas na documentação do coverage.

Dúvidas ou problemas, acesse nosso github e fique a vontade para abrir uma *issue* para que possamos esclarecer!

CHAPTER

FOUR

INDÍCES E TABELAS

- genindex
- modindex
- search