
Aplicativos Docker

Fernando Anselmo

<http://fernandoanselmo.orgfree.com/wordpress/>

Versão 1.0 em 15 de agosto de 2020

1 Aplicativos por Ordem Alfabética

```
1
2      ##
3      ## ## ##
4      ## ## ## ## ## ##
5      /oooooooooooooooooooo\___/ ===
6      /
7  ~ { ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ / ~ ~ ~ ~ ~
8      \___ o
9      \
10     \___\
11
```

1.1 Apache

```
$ docker run -d -p 8080:80 -v /home/usuario/[pastaAssociativa]:/var/www/html
--name meu-apache nimmis/apache-php7
```

1.2 Camunda

```
$ docker run --name meu-camunda -p 8070:8070 -p 8090:8090 -p 9090:9090
camunda/camunda-bpm-workbench
```

1.3 Caravel

```
$ docker pull amancevice/caravel
$ docker run --name caravel -d -p 8088:8088 amancevice/caravel
$ docker exec -it caravel demo
```

Usuário: admin — admin

Inicialização da Base de Dados:

```
$ docker run --detach --name caravel ... amancevice/caravel
$ docker exec -it caravel caravel-init
```

Determinar aonde será armazenado o banco de dados do Caravel; podemos escolher entre SQLite, MySQL, PostgreSQL ou Redshift. Utilizar a variável de ambiente *SQLALCHEMY_DATABASE_URI* para apontar o Caravel para SGBD corretamente. Certificar de definir *SECRET_KEY* ao criar o contêiner.

SQLite

```
1 $ docker run --detach --name caravel \
2 --env SECRET_KEY="mySUPERsecretKEY" \
3 --env SQLALCHEMY_DATABASE_URI="sqlite:///home/caravel/db/caravel.db" \
4 --publish 8088:8088 \
5 --volume [homeUsuario]/caravel:/home/caravel/db \
6 amancevice/caravel
```

MySQL:

```
1 $ docker run --detach --name caravel \
2 --env SECRET_KEY="mySUPERsecretKEY" \
3 --env SQLALCHEMY_DATABASE_URI="mysql://user:pass@host:port/db" \
4 --publish 8088:8088 \
5 amancevice/caravel
```

PostgreSQL:

```
1 $ docker run --detach --name caravel \
2 --env SECRET_KEY="mySUPERsecretKEY" \
3 --env SQLALCHEMY_DATABASE_URI="postgresql://user:pass@host:port/db" \
4 --publish 8088:8088 \
5 amancevice/caravel
```

Redshift:

```
1 $ docker run --detach --name caravel \
2 --env SECRET_KEY="mySUPERsecretKEY" \
3 --env SQLALCHEMY_DATABASE_URI="redshift+psycopg2://username@host.amazonaws.com:5439/db" \
4 --publish 8088:8088 \
5 amancevice/caravel
```

1.4 DokuWiki

```
$ docker network create dokuwiki-tier
$ docker run -d -p 80:80 -p 443:443 --name dokuwiki --net kuwiki-tier
--volume /home/fernando/dokuwiki-persistence:/bitnami bitnami/dokuwiki:latest
```

1.5 ElasticSearch

```
$ docker run -d -p 9200:9200 -p 9300:9300 -it -h elasticsearch
--name elasticsearch elasticsearch
```

Testar ElasticSearch: `$ curl http://localhost:9200`

```
$ docker run -d -p -p5601:5601 -h kibana --link elasticsearch:elasticsearch
kibana
```

Testar Kibana: `http://localhost:5601`

```
$ docker run -d -p -p5601:5601 -h kibana --link elasticsearch:elasticsearch
kibana
```

```
$ docker run -h logstash --name logstash --link elasticsearch:elasticsearch
-it -rm -v "$PWD":/config-dir logstash -f /config-dir/logstash.conf
```

Arquivo logstash.conf:

```
1 input {
2   stdin {}
3 }
4 output {
5   elasticsearch { hosts => ["elasticsearch:9200"] }
6 }
```

Outro exemplo:

```
1 input {
2   tcp {
3     port => 9500
4   }
5 }
6 output {
7   elasticsearch { hosts => ["elasticsearch:9200"] }
8 }
```

1.6 Ember

Instalar a imagem:

```
$ docker pull danlynn/ember-cli:2.12.1
```

Colocar o comando no Bash:

```
$ nano /.bashrc
```

E adicionar a linha ao final do arquivo:

```
alias emb='docker run -it --rm -v $(pwd):/myapp danlynn/ember-cli:2.12.1'
```

Sair e entrar novamente na tela de comandos.

Criar o contêiner:

```
$ docker run --name meu-ember -it -v $(pwd):/myapp -p 4200:4200 -p 49153:49153
```

```
danlynn/ember-cli:2.12.1 bash
```

Sair com exit:

```
# exit
```

CRIAR O PROJETO

Criar uma pasta:

```
$ mkdir novo_proj
```

```
$ cd novo_proj
```

Iniciar o projeto:

```
$ emb ember init
```

```
$ emb npm install
```

```
$ emb bower --allow-root install
```

Executar o container:

```
$ docker start meu-ember
```

```
$ docker exec -it meu-ember bash
```

E iniciar o servidor:

```
# ember server
```

Ao sair:

```
$ docker stop meu-ember
```

1.7 Grafana

```
$ docker run -d -p 3000:3000 --name meu-grafana grafana/grafana
```

Padrão: admin — admin

Mudar para: admin — grafana

1.8 Ignite

```
$ docker run -d -v $PWD/ignite_work_dir:/persistence  
-e IGNITE_WORK_DIR=/persistence --name meu-ignite apacheignite/ignite
```

```
$ docker run -it --net=host -e "CONFIG_URI=https://raw.githubusercontent.com/  
apache/ignite/master/examples/config/example-cache.xml"apacheignite/ignite
```

1.9 JLNP - Java Loading Network Protocol

Dockerfile:

```
1 # from your favorite base image which includes the latest Java, this example uses Debian  
2 FROM <your-java-base-image>  
3  
4 # xorg and sudo is needed to run X as non-root  
5 RUN apt-get update && \
```

```

6 apt-get install -y xorg sudo
7
8 # run X as non-root
9 RUN export uid=1000 gid=1000 && \
10 mkdir -p /home/dockeruser && \
11 echo "dockeruser:x:${uid}:${gid}:Developer,,,:/home/dockeruser:/bin/bash" >> /etc/passwd
    && \
12 echo "dockeruser:x:${uid}:" >> /etc/group && \
13 echo "dockeruser ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/dockeruser && \
14 chmod 0440 /etc/sudoers.d/dockeruser && \
15 chown ${uid}:${gid} -R /home/dockeruser
16
17 USER dockeruser
18 ENV HOME /home/dockeruser

```

Para execução é necessário permitir os Clientes X:

```

$ docker run -ti --rm -e DISPLAY=$DISPLAY
-v /tmp/.X11-unix/:/tmp/.X11-unix/
-v /<directory-containing-jnlp-file>:/jnlp
<your-container> /bin/bash
dockeruser@62ed23a5ecf8:/ $ javaws jnlp/dynamictree_webstart.jnlp

```

1.10 Meteor

Baixar a imagem de desenvolvimento:

```
$ docker pull danieldent/meteor
```

Criar o contêiner:

```

$ docker run -it --rm -p 3000:3000 --link some-mongo:mongo -e
MONGO_URL=mongodb://mongo:27017/appdb
-v "$(pwd)"/app danieldent/meteor meteor create [app]

```

Entrar no diretório:

```
$ cd [app]
```

Entrar no diretório:

```

$ docker run -it --rm -p 3000:3000 --link some-mongo:mongo -e
MONGO_URL=mongodb://mongo:27017/appdb -v "$(pwd)"/app danieldent/meteor meteor

```

Criar base no MongoDB:

```

1 Todos = new Mongo.Collection('Todos');
2 Todos.find().fetch(); // Traz os registros
3 Todos.insert({campo: 'valor', criadoEm: new Date()});
4
5 meteor add accounts-ui accounts-password
6 Adicionar no template: {{> loginButtons}}
7
8 {{#if currentUser}}
9 Estou logado
10 {{else}}
11 Favor se logar
12 {{/if}}

```

Ctrl+C duas vezes para parar.

1.11 Node.js

Baixar a imagem:

```
$ docker pull node
```

Contêiner associado ao MongoDB:

```
$ docker run -it --name meu-node -v "$(pwd)":/data -w /data -p 3000:3000  
--link some-mongo:mongo node bash
```

Atachar:

```
$ docker attach --sig-proxy=false meu-node
```

Desatachar:

Ctrl + P + Q

1.12 Phoneygap

Baixar a imagem:

```
$ docker pull nmaas87/webratio-phoneygap
```

Criar o contêiner:

```
$ docker run -v <application-parent-dir>:/data nmaas87/webratio-phoneygap  
phoneygap create <application-name>
```

Rodar o servidor do Phoneygap:

```
$ docker run -d -p <port>:3000 -v  
<application-dir>:/data nmaas87/webratio-phoneygap phoneygap serve -p 3000
```

Construir para o Android:

```
$ docker run -v <application-dir>:/data nmaas87/webratio-phoneygap  
phoneygap build android
```

1.13 Portainer

Baixar a imagem:

```
$ docker pull portainer/portainer
```

Criar o contêiner:

```
$ docker volume create portainer_data  
$ docker run -d -p 9000:9000 --name meu-portainer  
-v /var/run/docker.sock:/var/run/docker.sock  
-v portainer_data:/data portainer/portainer
```

1.14 Selenium

Instalar:

```
$ docker run -d -p 4444:4444 -p 5900:5900 --name  
meu-selenium selenium/standalone-firefox
```

Página Principal:

```
https://github.com/SeleniumHQ/docker-selenium  
http://0.0.0.0:4444/grid/console
```

1.15 Swagger

Baixar a imagem:

```
$ docker pull swaggerapi/swagger-ui
```

Criar o contêiner:

```
$ docker run -d -p 80:8080 -v [homeUsuario]/Aplicativos/swagger:/usr/share/nginx/html  
--name meu-swagger swaggerapi/swagger-ui
```

Visualização

1. Instalar globalmente o pacote http-server:

```
$ sudo npm install -g http-server
```

2. Na pasta do arquivo ativar o server com o parâmetro CORS (evita erro Cross-Reference):

```
$ http-server --cors
```

3. Abrir o Swagger UI

```
http://localhost:8080/[nome].json
```

1.16 Vue.js

Primeiro é necessário instalar uma extensão do chrome chamada: *Vue.jsdevtools*

Instalar: *npm* e *nodejs*

1. Baixar a imagem:

```
$ docker pull amurf/docker-vue-cli
```

2. Editar o bash init:

```
$ nano /.bashrc
```

3. Adicionar as seguintes linhas

```
$ alias vue='docker run -it --rm -v "$PWD:$PWD-w "$PWD"amurf/docke  
vue'
```

4. O node responde ao seguinte comando:

```
$ node (para sair .exit)
```

5. E o Vue.js responde ao seguinte comando:

```
$ vue
```

6. Ver as templates oficiais para a criação de projetos:

```
$ vue list
```

7. Criar um projeto:

```
$ vue init [template|webpack] [nomeProjeto]
```

8. Opções:

```
1 Nome do Projeto?
2 Descrição do Projeto?
3 Autor?
4 Vue Build? [Runtime-only - extensões .vue]
5 vue-router? No
6 ESLint? Yes [auxilia a manter o código correto]
7 Padrão de Projeto? Standard
8 Teste Unitarios? No
9 Teste? No
```

9. Depois do projeto gerado:

```
$ cd [nome projeto]
```

```
$ sudo npm install $ npm run dev
```

1.17 Yo.js

Baixar a imagem:

```
$ docker pull alexagency/angular-yeoman
```

Executar o contêiner:

```
$ docker run -it --rm --link some-mongo:mongo -p 9000:9000 -p 3000:3000 -p 3001:3001
-v $(pwd)/angular:/app alexagency/angular-yeoman
```

Pulo do Gato: Criou uma pasta /angular - usar chmod 777

```
$ yo angular
```

```
$ yo angular-fullstack
```

```
$ yo gulp-angular
```

Angular e FullStack:

```
$ grunt build --force
```

```
$ grunt test
```

```
$ sed -i s/localhost/0.0.0.0/g Gruntfile.js
```

```
$ grunt serve --force
```

```
http://localhost:9000
```

Gulp:

```
$ gulp build
```

```
$ gulp test
```

```
$ gulp serve
```

```
http://localhost:3000 ou http://localhost:3001
```

NodeJS, arquivo angular/server/config/environment/development.js e production.js

```
MongoClient.connect('mongodb://...')
```


1.18 Zope/Plone

Baixar a imagem:

```
$ docker pull plone
```

Executar o contêiner:

```
$ docker run -p 8080:8080 plone
```