

- Back-End(<https://imasters.com.br/back-end>)
 - Mobile(<https://imasters.com.br/mobile>)
 - Front End(<https://imasters.com.br/front-end>)
 - DevSecOps(<https://imasters.com.br/devsecops>)
 - Design & UX(<https://imasters.com.br/design-ux>)
 - Data(<https://imasters.com.br/data>)
 - APIs e Microserviços(<https://imasters.com.br/apis-microservicos>)
 - Crypto(<https://cryptobusinessreview.com/>)


BANCO DE DADOS

9 JAN, 2018

Docker Compose: O que é? Para que serve? O que come?

 (<https://www.facebook.com/sharer?u=https://imasters.com.br/banco-de-dados/docker-compose-o-que-e-para-que-serve-o-que-come>)

 (<https://twitter.com/share?url=https://imasters.com.br/banco-de-dados/docker-compose-o-que-e-para-que-serve-o-que-come>)

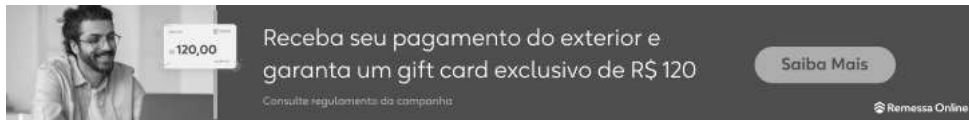
 (<https://www.linkedin.com/shareArticle?url=https://imasters.com.br/banco-de-dados/docker-compose-o-que-e-para-que-serve-o-que-come>)

COMPARTILHE!

CRISTIAN TRUCCO
([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))
Tem 6 artigos publicados com 3600 visualizações desde 2017



PUBLICIDADE



(https://lp.remessaonline.com.br/receba-do-exterior?utm_medium=banner_campanha_devs&utm_source=portal_imaster&utm_campaign=top_small-pj_devs)



CRISTIAN TRUCCO ([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))

6 

DevOps na Concreta, estudou Analise e Desenvolvimento Sistemas, usando containers e tentando automatizar quase tudo com alguns truques! Entusiasta de cloud, marombeiro desde pequeno e jogador profissional de Street Fighter 5 nas horas vagas.

LEIA MAIS ([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))

17 JAN, 2019

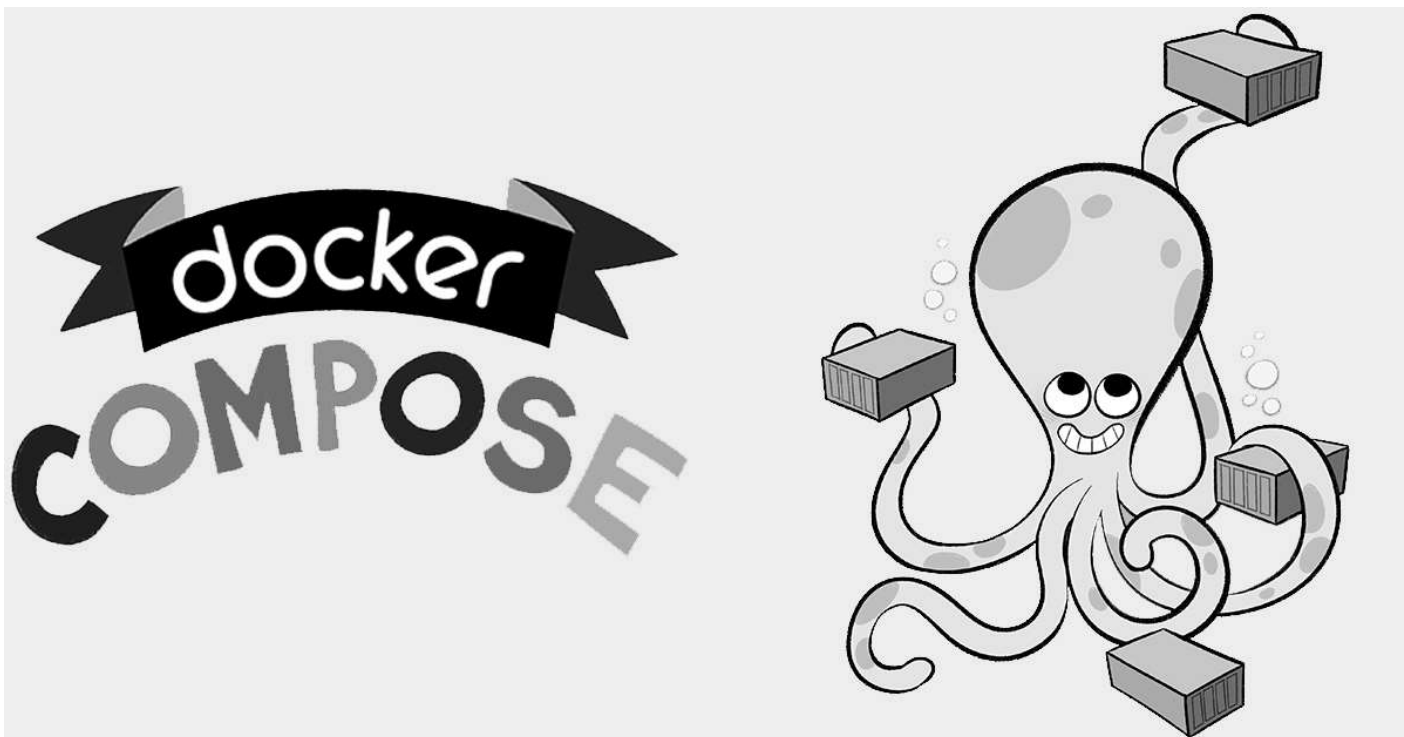
Automatize a sua estrutura com Terraform (<https://imasters.com.br/devsecops/automatize-sua-estrutura-com-terraform>)

2 MAR, 2018

Tudo o que você precisa saber sobre o Kubernetes – Parte 02 (<https://imasters.com.br/desenvolvimento/tudo-o-que-voce-precisa-saber-sobre-o-kubernetes-parte-02>)

27 FEV, 2018

Tudo o que você precisa saber sobre Kubernetes – Parte 01 (<https://imasters.com.br/desenvolvimento/tudo-o-que-voce-precisa-saber-sobre-kubernetes-parte-01>)



Docker Compose é o orquestrador de containers da Docker. E como funciona um orquestrador em uma orquestra? Ele rege como uma banda deve se comportar/tocar durante uma determinada apresentação ou música.

Com o Docker Compose é a mesma coisa, mas os maestros somos nós! Nós que iremos reger esse comportamento através do arquivo chamado `docker-compose`, semelhante ao `Dockerfile`, escrito em **YAML** (<http://yaml.org/>) (acrônimo recursivo para **Y**AML **A**in't **M**arkup **L**anguage) é um formato de codificação de dados legíveis por humanos, o que torna fácil de ler e entender o que um Compose faz! Mais informações na [Wikipedia](https://pt.wikipedia.org/wiki/YAML) (<https://pt.wikipedia.org/wiki/YAML>).

Um exemplo prático de como funciona o Docker Compose é: imagine que temos uma aplicação Java ou PHP e que essa aplicação depende de um banco de dados MySQL e, para disponibilizar essa aplicação na internet, queremos utilizar um proxy na frente. O cenário é bem típico e um cara de infraestrutura configura esse ambiente fácil em menos de um dia. Beleza, esse é o ponto!

Agora imagina que para cada cliente, precisamos realizar esse setup pelo menos umas três vezes: um para Desenvolvimento, Homologação e outro para Produção. Complicou um pouco, né?

Outro detalhe, se esses ambientes estão em cloud e existe uma Instância/VM para cada aplicação, isso pode gerar muito custo, recurso desperdiçado e tempo desnecessário gasto, sem contar o trabalho que dá para montar essa infra toda vez que surge um projeto novo. E no mundo de hoje, precisamos ser ágeis para entregar valor ao cliente.

Outra grande desvantagem desse processo é que ele é altamente manual, logo existe uma possibilidade de acontecer um erro humano.



Bom, falei isso tudo para entender qual a “dor” que tínhamos na infra e explicar como o Docker Compose pode ajudar nessa árdua tarefa!

O Compose File

Nesse arquivo Compose que mencionei no início do texto, descrevemos a infraestrutura como código e como ela vai se comportar ao ser iniciado. Se digo que preciso de um banco de dados para rodar minha aplicação Java/Php, descrevo isso no meu Compose e digo que minha aplicação depende do meu container de banco de dados MySQL para rodar.

Outra coisa legal, é que podemos definir o comportamento que o Docker vai ter caso um dos containers venha a falhar. Descrevo no Compose que, caso o banco de dados falhe por algum motivo, o Docker que suba outro imediatamente. Consigo isolar essas aplicações em uma rede separada e quais volumes essas aplicações vão utilizar para persistir os dados. Vamos subir todos esses serviços descritos no Compose com apenas um comando.

“Essa automatização é muito bacana, a forma que temos de orquestrar os serviços de forma declarativa!” (YAML™) [Version 1.2 \(http://www.yaml.org/spec/1.2/spec.html\)](http://www.yaml.org/spec/1.2/spec.html).

A imagem abaixo ilustra como é um arquivo do Docker Compose e como declaramos os serviços:

```
1  version: '3'
2
3  services:
4    db:
5      image: mysql:5.7
6      volumes:
7        - db_data:/var/lib/mysql
8      restart: always
9      environment:
10        MYSQL_ROOT_PASSWORD: wordpress
11        MYSQL_DATABASE: wordpress
12        MYSQL_USER: wordpress
13        MYSQL_PASSWORD: wordpress
14
15    wordpress:
16      depends_on:
17        - db
18      image: wordpress:latest
19      ports:
20        - "8000:80"
21      restart: always
22      environment:
23        WORDPRESS_DB_HOST: db:3306
24        WORDPRESS_DB_USER: wordpress
25        WORDPRESS_DB_PASSWORD: wordpress
26  volumes:
27    db_data:
```

O que consigo fazer no Compose?

Outro ponto interessante para comentar, são as variáveis de ambiente, podemos configurar no Compose usando o environment, passando as variáveis que serão usadas por nossa aplicação em determinado ambiente, quando os serviços estiverem subindo.

No caso do banco de dados em nosso exemplo, passamos o host, porta, usuário e senha do banco de dados que o WordPress vai usar para poder instalar e depois funcionar.

Em resumo, utilizando o Docker Compose, em vez de o administrador executar o docker run na mão para cada container e subir os serviços separados, linkando os containers das aplicações manualmente, temos um único arquivo que vai fazer essa orquestração e vai subir os serviços/containers de uma só vez. Isso diminui a responsabilidade do Sysadmin ou do desenvolvedor de ter que gerenciar o deploy e se preocupar em rodar todos esses comandos para ter a sua aplicação rodando com todas as suas dependências.

Neste link (<https://github.com/concrete-cristian-trucco/wordpress-mysql-compose.git>), temos um arquivo Docker Compose de exemplo que vamos executar para ver como funciona. Na prática, vamos usar o Play With Docker (ou simplesmente 'PWD') para testar o Docker Compose.

O PWD nada mais é que uma aplicação rodando o Docker com todos os serviços deste já instalados, simulando uma Vm Linux. Isso é conhecido como DIND (Docker in Docker) e tudo vai ser via navegador web, sem precisar instalar nada nas nossas máquinas para rodar o exemplo. Bacana, né? Agora chega de historinha e vamos ver esse cara funcionando.

Acessem este [link](http://play-with-docker.com/) (<http://play-with-docker.com/>).

Vamos ter uma tela semelhante a essa, então clique no botão <add New Instance> para lançar uma "VM" no PWD.



De posse desse terminal já podemos dar o primeiro comando, o [Git Clone \(https://github.com/concrete-cristian-trucco/wordpress-mysql-compose.git\)](https://github.com/concrete-cristian-trucco/wordpress-mysql-compose.git) para clonar o repositório do Github.

A saída do comando vai ser semelhante a esta:

```
[node1] (local) root@10.0.5.3 -
$ git clone https://github.com/concrete-cristian-trucco/wordpress-mysql-compose.git
Cloning into 'wordpress-mysql-compose'...
remote: Counting objects: 29, done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 29 (delta 11), reused 10 (delta 1), pack-reused 0
Unpacking objects: 100% (29/29), done.
[node1] (local) root@10.0.5.3 -
$
```

Agora precisamos entrar no diretório que baixamos do Git:

```
1 | cd wordpress-mysql-compose/
```

Após entrar no diretório, podemos listar os arquivos com o comando `ls -l` para ver o conteúdo desse diretório:

```
1 | ls -l
2 | 1
3 | ls -l
```

```
$ ls -l
total 52
-rw-r--r-- 1 root root      15 Oct 6 12:50 Dockerfile
-rw-r--r-- 1 root root     275 Oct 6 12:50 README.md
-rw-r--r-- 1 root root   35351 Oct 6 12:50 Wordpress_Docker.png
-rwxr-xr-x 1 root root     193 Oct 6 12:50 deployStack.sh
-rw-r--r-- 1 root root     546 Oct 6 12:50 docker-compose.yml
```

E por fim, estando no mesmo diretório do arquivo do `docker-compose.yml`, podemos rodar o comando que vai executar de fato o Compose:

```
1 | docker-compose up
```

Inicialmente, o Docker vai baixar as imagens que vão ser usadas nesse compose. Pode demorar alguns minutos na primeira vez, depois o comando é bem rápido.

Rodando o comando dessa forma, o Docker joga todas as informações para a tela do terminal. Podemos colocar a flag `-d` para termos o terminal funcional, enquanto o compose roda em segundo plano:

```
1 | docker-compose up -d
```

```
$ docker-compose up -d
Starting wordpressmysqlcompose_db_1 ...
Starting wordpressmysqlcompose_db_1 ... done
Starting wordpressmysqlcompose_wordpress_1 ...
Starting wordpressmysqlcompose_wordpress_1 ... done
```

Como podemos ver se o Compose está rodando e se de fato a quantidade de containers que definimos e as portas que expomos está ok? Com o comando `docker-compose ps`.

```

(node1) (local) root@10.0.5.3 ~/wordpress-mysql-compose
$ docker-compose ps

```

Name	Command	State	Ports
wordpressmysqlcompose_db_1	docker-entrypoint.sh mysqld	Up	3306/tcp
wordpressmysqlcompose_wordpress_1	docker-entrypoint.sh apach ...	Up	0.0.0.0:8000->80/tcp

Esse comando, como é possível ver na imagem, vai listar o nome da minha aplicação, os status desses serviços e as portas disponíveis para cada container.

OBS: Os comandos do Docker Compose só vão funcionar passando o caminho do diretório ou estando no mesmo diretório do arquivo do docker-compose.yml

Voltando: o legal do PWD é que assim que os containers sobem, as portas que definimos no compose aparecem disponíveis no Dashboard do PWD.

Abaixo, a porta 8000 disponível como link.

0a8f6488_node1

IP

10.0.5.3

8000

Memory

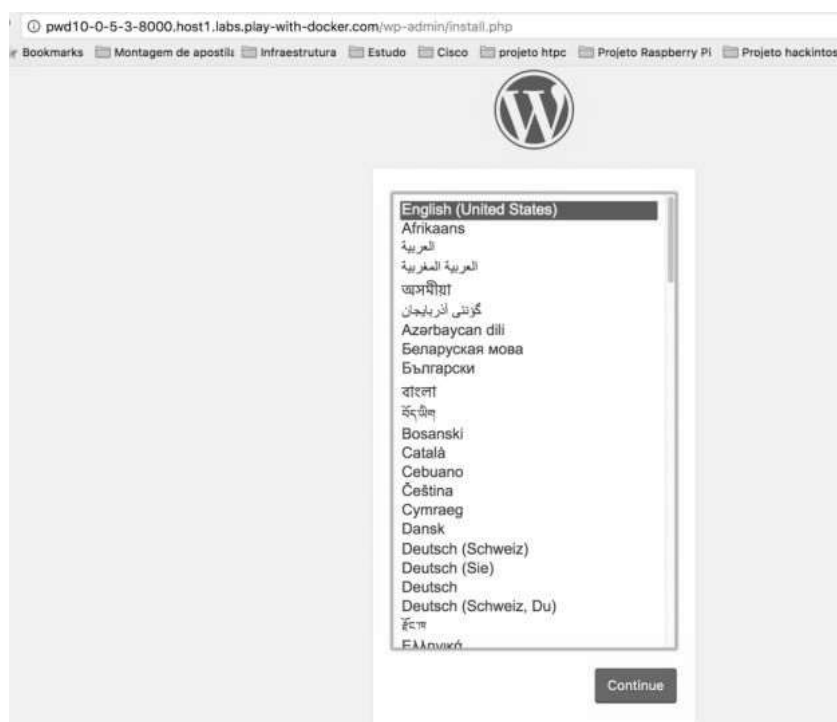
31.71% (1.267GiB / 3.996GiB)

CPU

0.12%

Quando clicar no link da porta, automaticamente o PWD redireciona para a aplicação, que no nosso caso é um WordPress.

Tela inicial da instalação do WordPress:



Após realizar o Wizard de instalação do WordPress, temos uma aplicação real funcionando em menos de 5 minutos. Esse é o barato de infraestrutura como código, containers e Docker!

Acessando o WordPress:



CRISTIAN TRUCCO ([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))
14 FEV. 2018





CRISTIAN TRUCCO ([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))
9 JAN, 2018



Docker Compose: O que é? Para que serve? O que come? (<https://imasters.com.br/banco-de-dados/docker-compose-o-que-e-para-que-serve-o-que-come>)



SAIBA MAIS
([HTTPS://IMASTERS.COM.BR/PERFIL/CRISTIANTRUCCO](https://imasters.com.br/perfil/cristiantrucco))

Cristian Trucco



(<https://www.concrete.com.br/>)



(<mailto:cristian.trucco@concrete.com.br>)

6 Artigo(s)

DevOps na Concrete, estudou Analise e Desenvolvimento Sistemas, usando containers e tentando automatizar quase tudo com alguns truques! Entusiasta de cloud, marombeiro desde pequeno e jogador profissional de Street Fighter 5 nas horas vagas.

1 comentário

Classificar por **Mais antigos**



Adicione um comentário...



Uigor Marshall

Excelente artigo, me serviu de base para um estudo que estou fazendo, porém não consegui finalizar, poderia me ajudar?

<https://hub.docker.com/r/uigormarshall/node-simple-api/>

não consigo conectar com o postgres e recebo o erro abaixo:

Unhandled rejection Error: connect ECONNREFUSED 127.0.0.1:5432

marshall-app-dev | at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1174:14)

Curtir · Responder · 4 a

Plugin de comentários do Facebook

Este projeto é mantido e patrocinado pelas empresas



(<https://apiki.com/>)



(<http://www.dialhost.com.br>)



(<https://huaweicloud.imasters.com.br/>)



(<https://www.idexo.com.br/>)



(<https://www.maxipago.com/>)



(<https://dev.paygo.com.br/>)



(<https://www.userede.com.br/>)



(<https://www.schoolofnet.com/cursos/gratuitos/>)
(<https://developers.totvs.com/>)
utm_source=imasters&utm_medium=patrocinio&utm_campaign=institucional_patrocin
institucional&utm_content=institucional_patrocinio_imasters_link-
institucional)



ASSINE NOSSA Newsletter

Fique em dia com as novidades do iMasters! Assine nossa newsletter e receba
conteúdos especiais curados por nossa equipe



Qual é o seu e-mail?

ASSINAR



SOBRE O IMASTERS ([HTTPS://IMASTERS.COM.BR/P/SOBRE-O-IMASTERS](https://imasters.com.br/p/sobre-o-imasters))

POLÍTICA DE PRIVACIDADE ([HTTPS://IMASTERS.COM.BR/P/POLITICA-DE-PRIVACIDADE](https://imasters.com.br/p/politica-de-privacidade))

FALE CONOSCO ([HTTPS://IMASTERS.COM.BR/FALE-CONOSCO/](https://imasters.com.br/fale-conosco/))

QUERO SER AUTOR ([HTTPS://IMASTERS.COM.BR/P/QUERO-SER-AUTOR](https://imasters.com.br/p/quero-ser-autor))

FÓRUM ([HTTPS://FORUM.IMASTERS.COM.BR/](https://forum.imasters.com.br/))

CBR ([HTTPS://CRYPTOBUSINESSREVIEW.COM/](https://cryptobusinessreview.com/))