Virtualização Do Ambiente De Desenvolvimento

Gustavo Morini[1], William Roberto Pelissari[1], Anderson Burnes[1]

[1]Faculdade Alfa – Umuarama – PR – Brasil

gustavommh@gmail.com, wrpelissari@gmail.com, burnes@professorburnes.com

Abstract. *This article describes some of the advantages of using virtualization in the development environment, its main commands, and how to build a functional environment for web development, making distinct technologies communicate in a fully-independent and fully operating-system environment.*

Resumo. *Este artigo descreve algumas vantagens de se utilizar a virtualização no ambiente de desenvolvimento, seus principais comandos e como montar um ambiente funcional para o desenvolvimento web, fazendo tecnologias distintas se comunicarem formando um ambiente completo e totalmente independente do sistema operacional.*

1. Introdução

Compreender as principais características para se criar um ambiente de desenvolvimento virtualizado, customizável com Docker e containers. Analisar como gerenciar os containers utilizando Docker, montar um ambiente de desenvolvimento Web funcional, configurar containers individuais interligados entre si, demonstrar o funcionamento de um projeto rodando dentro do ambiente virtual.

O ambiente de desenvolvimento se mostra muitas vez bem específico para cada tipo de projeto, configurar apenas uma vez e repassar para toda a equipe de trabalho funcionando exatamente igual em todas as máquinas, independente de sistema operacional vem a ser uma das justificativas de se usar Docker e containers.

Um ambiente convencional precisar ser totalmente refeito sempre que é preciso realizar uma manutenção na máquina hospedeira, dependendo da quantidade de elementos utilizados no ambiente de desenvolvimento essa pode ser uma tarefa árdua.

A virtualização com Docker chega com o intuito de simplificar esta tarefa, mantendo um nível profissional de segurança, confiabilidade e padronização, sendo para um único desenvolvedor ou uma grande equipe, facilmente customizável e replicável.

2. Apresentação

Segundo (Gomes & Souza, 2015), Docker é uma ferramenta que possibilita gerenciar aplicações distintas dentro de ambientes isolados denominados containers. É uma alternativa para desenvolvedores criarem com rapidez e facilidade seu ambiente de desenvolvimento e realizar todas as tarefas rotineiras, inclusive testes com total segurança. A principal funcionalidade é proporcionar ambientes isolados dentro da mesma máquina, independente de sistema operacional, disponível localmente ou remotamente através de mapeamento de portas. (Henrique Rocha Silva & — Licenciatura Orientador André Costa Drummond, 2017), descreve que um contêiner Docker basicamente, consiste em manter tudo o que é necessário para a aplicação ser

executada, independente do sistema operacional nativo, compartilhando apenas o kernel da máquina hospedeira, dentro do container existe o sistema operacional, configurações do usuário e metadados, no entanto a imagem Docker é somente leitura, criando assim uma nova instancia sempre que necessário. Quando o Docker inicia um container a partir de uma imagem ele adiciona uma camada de leitura e escrita na parte superior da imagem no qual a aplicação pode ser executada.

3. Problematização

Quando pensamos em desenvolvimento web, precisamos de um servidor que interprete os códigos, além de um banco de dados onde seja possível popular os dados da aplicação. Geralmente é utilizado um ambiente local antes de passar para o ambiente de produção na nuvem. A instalação desses recursos pode ser um tanto trabalhoso, devido a utilizar várias tecnologias distintas, que trabalharem juntas.

Vamos montar com Docker, um ambiente de desenvolvimento, rodando virtualmente os programas necessários para o aplicativo funcionar, de fácil manutenção e distribuição para outras pessoas da equipe, sendo recriado com apenas um comando.

4. Metodologia

Para a realização do estudo é necessário a instalação da ferramenta Docker e Docker-Compose, que auxilia na orquestração dos containers. Para os sistemas Windows e Mac existe um instalador pronto, já no caso dos sistemas baseados em Linux é possível instalar via terminal seguindo orientações do próprio site do desenvolvedor. Também foi utilizado uma aplicação de exemplo desenvolvida em PHP, com ajuda do framework Codeigniter, tendo uma conexão em MariaDB e utilizando o PHPmyAdmin para gerenciamento do banco de dados.

5. Resultados

Assim que instalado a ferramenta Docker é possível utilizar seus comandos diretamente no terminal do sistema, a (tabela 1) referência a função e seu respectivo comando no terminal, primeiramente vamos verificar se a instalação foi realizada corretamente exibindo a verão instalada no sistema. Assim podemos então criar o container à partir de uma imagem base, no caso o banco de dados MariaDB, neste comando se define o modo de execução, nome e senha de root para o novo container.

Quando se utiliza o termo run sempre criará um novo container, após a criação podemos listá-los para mais detalhe, é possível acessá-lo e rodar comandos específicos dentro deles, inspecionar a fundo mostrando todas as informações referentes a ele com o comando inspect. Parar sua execução, iniciá-lo sem a necessidade de criá-lo novamente, listar todos os containers criados na máquina e removê-los se necessário.

Tabela 1. Comandos Docker.

Função	Comando do terminal
Verificar Instalação	*docker --version*
Criar um container à partir de uma imagem	*docker run -d --name teste-db -e MARIADB_ROOT_PASSWORD=mypass mariadb*
Verifica containers em execução	*docker container ls*
Acessar terminal do container	*docker exec -it teste-db bash*
Acessar o mysql dentro do container	*mysql -u root -p*
Inspecionar o container em execução	*docker inspect teste-db*
Parar o container	*docker container stop teste-db*
Iniciar novamente o container criado	*docker container start teste-db*
Listar todos os containers existentes na máquina	*docker container ls -a*
Remover o container	*docker container rm teste-db*

Este são os passos básicos para criação e gerenciamento de um container com Docker, poderíamos criar mais de um contêiner e interligá-los, mas seria algo muito verboso que não tem a necessidade, ainda mais que a proposta é facilitar a experiência do desenvolvedor, assim vamos utilizar para criar o ambiente completo o Docker-compose, que à partir de uma lista de comandos interpreta e passa para o Docker executá-los de maneira simples.

Dentro de uma pasta vazia vamos criar um arquivo “docker-compose.yml” este será responsável por subir nossos serviços, o arquivo depende de uma endentação correta para compreender em qual camada estão os comandos e como interpretá-los.

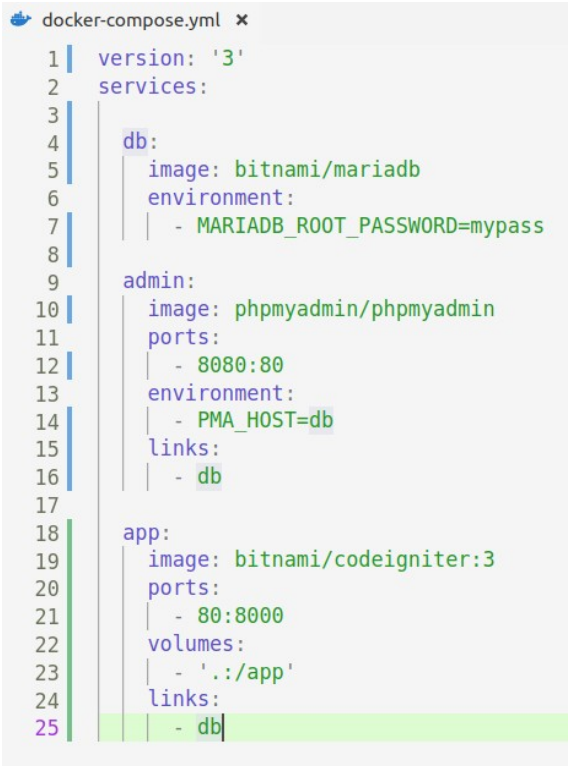

Figura 1. Estrutura do arquivo docker-compose.yml.

A figura 01 mostra como a organização é feita, a endentação é fundamental para este tipo de arquivo, através dela que define como cada comando deve ser interpretado, inicia com a versão dos comandos, após os serviços a serem criados, “db” é o nome do serviço, pode ser qualquer um, usamos db para ficar mais didático, “image” é qual imagem o docker vai procurar para espelhar e criar o container db, “environment” são as variáveis globais que podemos passar no momento da estanciação, no caso define a senha de root para “mypass”. Próximo serviço “admin” referencia a imagem do “phpmyadmin”, realizando um mapeamento de portas para não haver conflito com o app, sua variável global é “PMA_HOST” que define o nome do host que irá se conectar, perceba que é o mesmo nome do serviço de banco de dados criado acima, “links” apenas define que irá se conectar com o serviço db. Já o serviço “app” é a nossa aplicação, utilizando a imagem do framework que já tem o ambiente pré-configurado, define a porta a ser utilizada local e virtual, “volumes” é muito importante, define onde irão ficar os arquivos localmente sendo espelhados dentro do container, e novamente diz para se conectar com o serviço db.

No mesmo diretório criamos a pasta myapp nela vamos colocar os arquivos da aplicação. Dentro da aplicação existe o arquivo de configuração do banco de dados chamado database.php onde devemos preencher os dados de acordo com o serviço bd do arquivo docker-compose.yml como demostrado na (figura 2).

```
78      'hostname' => 'db',
79      'username' => 'root',
80      'password' => 'mypass',
81      'database' => 'controle',
```

Figura 2. Arquivo database.php.

Neste momento é possível iniciar nossos serviços descritos no docker-compose.yml, seguindo a (tabela 2) é possível acompanhar a função e cada comando executado no terminal, vamos iniciar os containers, onde uma primeira execução é feita buscando na nuvem as imagens referenciadas, sua primeira execução pode demorar um pouco de acordo com a conexão e a quantidade de imagens a serem baixadas, após isso a iniciação é feita em poucos segundos. Após concluir vamos listar os containers exibindo detalhes como nome, id e portas onde estão rodando. O docker-compose já inicia para e remove todos os containers de uma só vez com apenas um comando.

Tabela 1. Comandos Docker.

Função	Comando terminal
Iniciar todos os containers	*docker-compose up -d*
Listar todos os containers	*docker-compose ps*
Parar todos os containers	*docker-compose stop*
Remove todos os containers	*docker-compose down*

Na lista de containers é possível observar seu nome, o comando interno que está sendo executado, o estado e em qual porta está rodando e o redirecionamento que está sendo utilizado. O phpmyadmin pode ser acessado na URL: http://localhost:8080, Figura 03 e a aplicação URL: http://localhost, Figura 04.

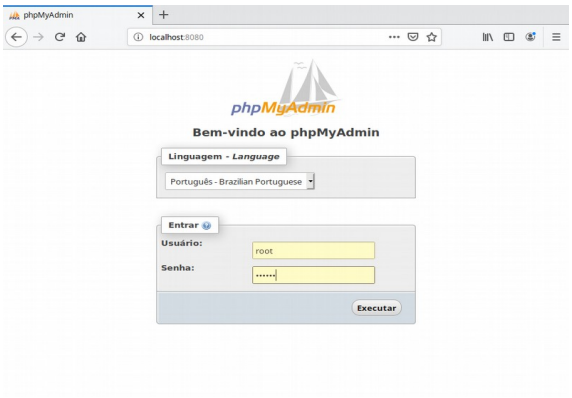

Figura 3. http://localhost:8080.

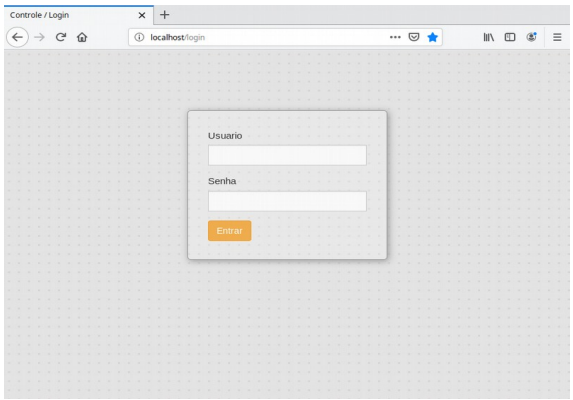

Figura 4. http://localhost.

Este é o ambiente completo rodando em Docker, todas as alterações feitas na pasta

myapp serão repassadas para o container interpretando internamente e devolvendo na URL definida no arquivo docker-compose.yml. Assim foi concluído o ambiente de desenvolvimento virtual em Docker.

6. Conclusão

Concluímos que a ferramenta é sim funcional, necessita o conhecimento de alguns comandos básicos, com um gerenciamento simples, com pouco estudo é possível criar e replicar ambientes complexos em apenas um arquivo central, executou com perfeição o trabalho proposto, mantém um sistema operacional limpo, livre de aplicações que não utilizamos no dia a dia, como ferramentas de testes, que podem ser criadas e apagadas rapidamente sem qualquer risco de conflito com outros aplicativos instalados. Mantém um nível de segurança alto, podendo manter informações dentro e fora do container, criar redes específicas isoladas da rede padrão. Por ser virtual e necessitar de adicionais para funcionar corretamente consome mais recursos da máquina host, tanto em disco quanto em processamento do que um ambiente local , mas menos que uma máquina virtual convencional que recria todo o sistema operacional do zero.

7. Referências

Gomes, R., & Souza, R. (2015). Docker – Infraestrutura como código, com autonomia e replicabilidade.

Henrique Rocha Silva, F., & — Licenciatura Orientador André Costa Drummond, C. (2017). Avaliação de Desempenho de Contêineres Docker para Aplicações do Supremo Tribunal Federal. Retrieved from http://bdm.unb.br/bitstream/10483/17796/1/2017_FlavioHenriqueSilva_tcc.pdf