Virtualização Do Ambiente De Desenvolvimento

Gustavo Morini1

1 Faculdade Alfa – Umuarama – PR – Brasil

gustavommh@gmail.com

**Abstract.** This article describes some of the advantages of using virtualization with Docker in the development environment. Demonstrates its main commands and how to set up a functional environment for web development, making distinct environments communicate with each other in a complete and totally independent environment of the host operating system.

**Resumo.** Este artigo descreve algumas vantagens de se utilizar a virtualização com Docker no ambiente de desenvolvimento. Demonstra seus principais comandos e como montar um ambiente funcional para o desenvolvimento web, fazendo ambientes distintos comunicarem entre si formando um ambiente completo e totalmente independente do sistema operacional hospedeiro.

# 1. Objetivo

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.

# 2. Justificativa

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.

# 3. Estrutura

….

# 4. Fundamentação Teórica

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 trabalho, para 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.

# 6. Materiais e Métodos

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.

# 7. Desenvolvimento

Quando pensamos em desenvolvimento web, com PHP e MYSQL, precisamos de um servidor local que interprete os códigos PHP, além de um banco de dados local onde seja possível popular os dados da aplicação.

A instalação desses recursos pode ser um tanto trabalhoso, devido a utilizar várias tecnologias distintas, que trabalharem juntas, existem aplicativos como Xampp, WampServer, EasyPHP, que facilitam essa tarefa, porém não seria a alternativa ideal para um ambiente de desenvolvimento mais profissional.

Vamos montar com Docker, um ambiente de desenvolvimento profissional, 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.

Foi utilizado o Docker no sistema operacional Linux Mint 19.1 64 bits, dentro do terminal instalamos os pacotes, “sudo apt install docker.io docker-compose”, verificamos se tudo foi instalado corretamente, “docker –v && docker-compose -v”. Estes comandos vão mostrar a versão de cada aplicativo instalado no sistema, desta forma verificamos que tudo está correto e podemos prosseguir.

Utilizando apenas o Docker poderíamos iniciar um banco de dados rapidamente, “sudo docker run --name teste-mariadb -e MYSQL\_ROOT\_PASSWORD=mypass -d mariadb”. Este comando está pedindo para o docker rodar um novo container, “run” como nome “testemariadb” passando uma variável de ambiente “-e” que diz que a senha de root será “mypass” rodando em modo demon “-d” chamando a imagem mariadb. Assim que executado o comando retorna o id do novo container, “cafe7b8a15b94fbd47147bc12be331028fb80c328e4fd14526146b50147fe512” demonstrando que está rodando em modo demon, o container também pode ser gerenciado pelo nome que foi definido no momento da criação “teste-mariadb”.

Agora podemos acessar o container para rodar os comandos SQL, “sudo docker exec -it teste-mariadb bash” o comando faz o Docker executar “exec” em modo interativo “-it” dentro do container “teste-mariadb” o “bash” terminal do sistema. Em nosso terminal ao invés do nome de usuário local “gustavo@linux:~$” passamos a utilizar o usuário “root@cafe7b8a15b9:/#” significa que estamos dentro do container como root podendo rodar os comandos necessários para criação do banco.

Voltando para o ambiente local podemos inspecionar o container “sudo docker inspect teste-mariadb” mostrando todas as informações desse container, inclusive "IPAddress": "172.17.0.2" que é a formata de acessar o container externamente.

Podemos listar os containers rodando na máquina, “sudo docker container ls”, parar o container utilizando seu nome ou id, “sudo docker container stop teste-mariadb”, iniciá-lo da mesma forma, “sudo docker container start teste-mariadb”, listar todos os containers existentes mesmo que não estejam em execução, “sudo docker container ls -a” e se deletar o container quando for necessário “sudo docker container rm teste-mariadb”.

Este são os passos básicos para criação e gerenciamento de um container com Docker, poderiamos criar mais de um conteiner e interligá-los, mas seria algo muito verboso que não tem a necessidade, ainda mais que a proposta é facilitar a vida 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. Em nosso projeto foi utilizado uma aplicação em PHP e MYSQL criada com o framework “CodeIgniter”, existem várias formas de se criar o ambiente, utilizamos a mais simples que foi encontrar a imagem do framework para Docker e integrar em nosso projeto.

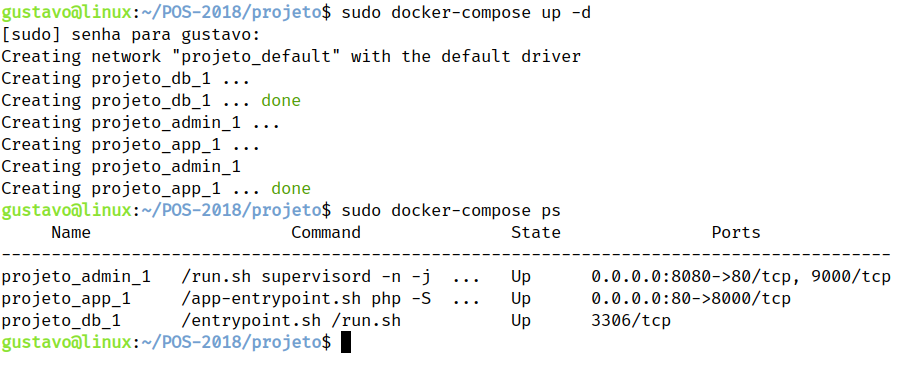
Dentro de uma pasta vazia vamos criar um arquivo “docker-compose.yml” este será responsável por subir nossos serviços, o arquivo depende de uma edentação correta para compreender em qual camada estão os comandos e como interpretá-los.



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

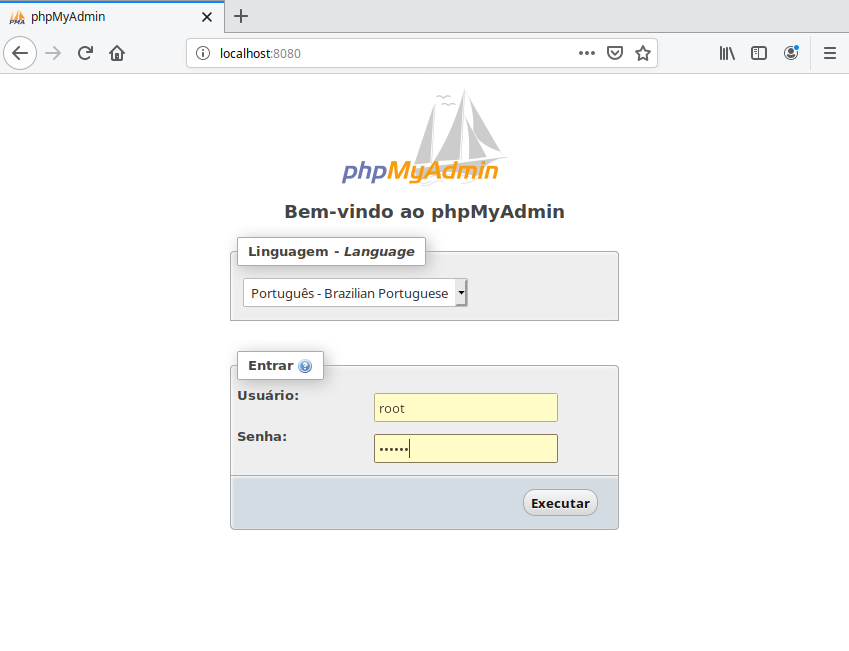
A figura 01 mostra como a organização é feita, a edentação é fundamental para este tipo de arquivo, através dela que define como cada comando deve ser interpretado, inicia com a versão dos comandos, após os serviços a serem criados, “db” é o nome do serviço, pode ser qualquer um, usamos db para ficar 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.

Feito isso criamos a uma pasta dentro do mesmo diretório chamada myapp, pelo terminal “mkdir myapp”, nela vamos colocar os arquivos da aplicação. Na aplicação existe o arquivo de banco de dados “database.php” onde devemos setar as informações de acordo com o container bd, “hostname:db, username:root, password:mypass, database:controle”. Neste momento já é possível subir os serviços, no diretório onde está o arquivo docker-compose.yml iremos rodar o comando “sudo docker-compose up -d”, a primeira execução demora um pouco por ter que baixar dos repositórios online as imagens necessárias para execução dos serviços descritos, após isso a iniciação é feita em poucos segundos, praticamente instantaneamente. Após é possível listar os container ativos com o comando “sudo docker-compose ps” como mostra a figura 02.

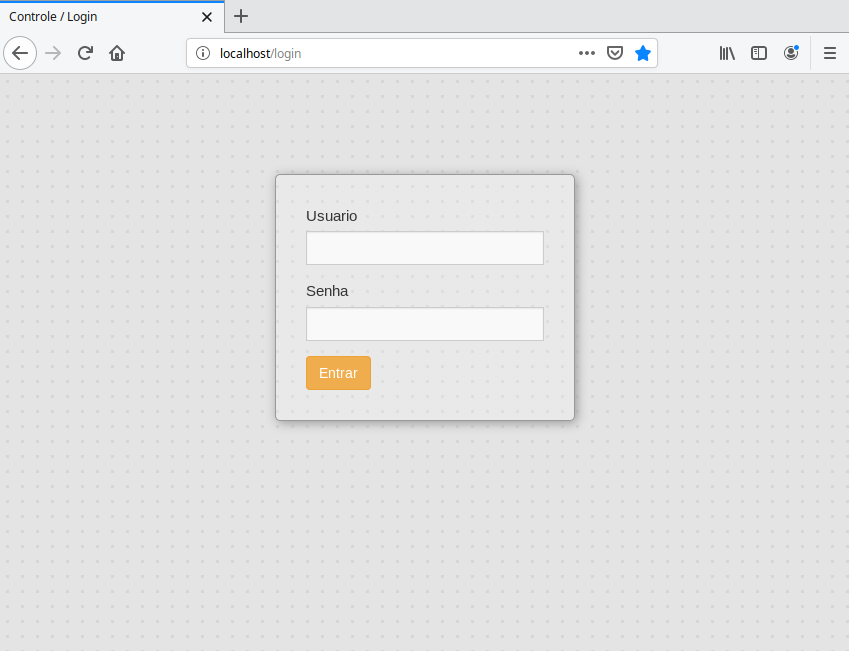


**Figura 2. Subindo e listando containers.**

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 configurada, para parar os containers todos de uma vez use o comando “sudo docker-compose stop” e iniciar todos novamente “sudo docker-compose start -d”. Assim foi concluído o estudo do ambiente de desenvolvimento virtual em Docker.

# 8. Análise dos Resultados e Discussão

Analisamos que um primeiro contato com a ferramenta pode desmotivar o desenvolvedor por exigir um conhecimento específico de alguns comandos e regras, mas assim que o primeiro projeto é criado nota-se que é um gerenciamento simples, que com um pouco de estudo é possível replicar ambientes complexos com muitas configurações em apenas um arquivo central.

# 9. Conclusões e Trabalhos Futuros

Concluímos que a ferramenta é sim funcional, 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.

# References

Boulic, R. and Renault, O. (1991) “3D Hierarchies for Animation”, In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) “Motion Capture White Paper”, <http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html>, December.

Holton, M. and Alexander, S. (1995) “Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials”, Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.