

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
FACULDADE DE TECNOLOGIA SENAI/SC FLORIANÓPOLIS
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

FELIPE MENDONÇA RUHLAND

ESTUDO SOBRE O USO DO DOCKER NA EXECUÇÃO DE APLICAÇÕES WEB

Florianópolis/SC

2016

FELIPE MENDONÇA RUHLAND

ESTUDO SOBRE O USO DO DOCKER NA EXECUÇÃO DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia do SENAI Florianópolis como requisito parcial para obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Professor Orientador: Paulo Bueno.

Florianópolis/SC

2016

FELIPE MENDONÇA RUHLAND

ESTUDO SOBRE O USO DO DOCKER NA EXECUÇÃO DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia do SENAI Florianópolis como requisito parcial para obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**APROVADA PELA COMISSÃO EXAMINADORA
EM FLORIANÓPOLIS, ?? DE JULHO DE 2016**

Prof. Bobiquins Estevão de Mello, Me. (SENAI/SC)
Coordenador do Curso

Profa. Jaqueline Voltolini de Almeida, Me. (SENAI/SC)
Coordenador de TCC

Prof. Paulo Bueno, Dr. (SENAI/SC)
Orientador

Prof. Fulado de tal, Me. (SENAI/SC)
Examinador

Dedico à mamãe ao papai, à vovó e ao vovô!!!

AGRADECIMENTOS

A Deus por xxxx

Agradecimentos especiais à minha família ...

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível”
(CHARLES CHAPLIN)

Ruhland, Felipe. **Estudo sobre container linux para execução de aplicações web**. Florianópolis, 2016. 26f. Trabalho de Conclusão de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas - Curso Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia do SENAI, Florianópolis, 2016.

RESUMO

Segundo a NBR6028:2003, o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (...) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: Latex. Abntex. Editoração de texto.

SOBRENOME, Nome. **Título do trabalho.** Florianópolis, 2013. 89f. Trabalho de Conclusão de Curso Superior de Tecnologia em Redes de Computadores - Curso Redes de Computadores. Faculdade de Tecnologia do SENAI, Florianópolis, 2013.

ABSTRACT

This is the english abstract.

Key-words: Latex. Abntex. Text editoration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

456	Isto é um número
123	Isto é outro número
BB	bom e barato

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
\in	Pertence

SUMÁRIO

1 INTRODUÇÃO	13
1.1 JUSTIFICATIVA	14
1.2 OBJETIVOS	14
1.2.1 Objetivo geral	14
1.2.2 Objetivos específicos	14
1.3 METODOLOGIA	15
1.4 ESTRUTURA DO TRABALHO	15
2 REVISÃO DA LITERATURA	16
2.1 Introdução ao Docker	16
2.1.1 Volumes	16
2.1.1.1 Volume vinculado ao host	16
2.1.2 Ambiente de Desenvolvimento e Homologação	17
2.1.3 Instalação do Software	17
2.1.4 Escalabilidade	17
2.1.5 Entregabilidade	18
2.1.6 Segurança	18
2.1.7 O que é Docker?	18
2.2 Diferenças entre servidor dedicado, virtualizado e containerizado	19
3 PROCEDIMENTOS METODOLÓGICOS	20
4 RESULTADOS E DISCUSSÕES	21
4.1 Desenvolvimento	21
4.2 Distribuição	21
4.3 Publicação	21
5 CONCLUSÃO	23
REFERÊNCIAS	24
APÊNDICE A Código fonte	25
ANEXO A Pesquisa IBGE	26

1 INTRODUÇÃO

Mais de dois terços das aplicações web rodam em ambiente unix, segundo W3Techs (2016). Os sistemas são executados em servidores dedicados ou virtualizados. Os servidores dedicados são a maneira mais natural de servir uma aplicação web. São máquinas físicas, normalmente em datacenters, com um sistema operacional linux e com todo o hardware disponível. Entretanto, executar uma aplicação web em um servidor dedicado acaba por desperdiçar muitos recursos do mesmo. Para combater este desperdício, trabalha-se há décadas para aperfeiçoar um servidor que consiga evitar o desperdício, com a divisão dos recursos.

Nos últimos anos, usou-se muito as máquinas virtuais para aproveitar melhor os recursos dos servidores. Elas funcionam como novas máquinas dentro da máquina física e podem ser incluídas na rede como se fossem uma máquina física. Desta maneira, um servidor dedicado passa a ser múltiplas máquinas, com seus próprios recursos e totalmente isoladas, que traz ainda mais segurança para os administradores de sistemas.

Com esta estratégia, os VPS (Servidores privados virtuais) tornaram muito mais acessíveis ao público em geral, pois era possível contratar um pequeno servidor virtualizado e ter total controle das configurações desde servidor. Isso colaborou com pequenos empreendedores que puderam expor seu trabalho de maneira mais econômica e, com isso, abrir um leque de possibilidades para novas empresas.

A máquina virtual, por padrão, funciona como uma máquina totalmente nova. Ou seja, ela possui sistema operacional próprio, permissões individuais e recursos compartilhados com a máquina anfitriã. É possível instalar diversas versões do kernel, por exemplo, sem que uma interfira na outra. Contudo, observa-se que para cada máquina virtual criada num servidor dedicado existe uma sobrecarga do sistema operacional, pois além do sistema instalado na máquina anfitriã, cada máquina virtual possui seu sistema individual. Sabe-se, também, que o sistema operacional necessita de uma série de recursos para o bom funcionamento, de maneira que os recursos não podem ser muito ínfimo para não ocorrer problemas. Em paralelo às máquinas virtuais, existe uma outra alternativa, mais leve, chamada de container.

Containers linux existem com a finalidade de isolar ambientes dentro de um sistema operacional linux. Eles não são máquinas virtuais, mas também conseguem restringir acesso, definir recursos próprios, mas não sobrepõe o sistema operacional. Ele usa o sistema da máquina anfitriã e, com isso, consegue utilizar menos recursos que a máquina virtual. Em razão dessas vantagens, muitos estudam para deixar a criação e manutenção desses containers uma tarefa mais fácil para os profissionais de TI. O caso mais conhecido nos últimos anos é da ferramenta Docker, que descomplicou a maneira de criar e gerir containers dentro de um sistema operacional. Pode-se dizer que o projeto é um sucesso, pois recebe mais utilizadores a cada dia e já possui

investimentos na casa dos milhões de dólares.

Fala-se muito na arquitetura de microserviço, que traz vantagens para o desenvolvedor, por ter um escopo reduzido, facilita a criação e execução de testes, deploy e inúmeros outros fatores. Essa nova abordagem, traz consigo a ideia de executar o microserviço em um container para simplificar a infraestrutura, de modo a facilitar a escalabilidade da aplicação e a sua manutenção. Com este pensamento, pode-se criar milhares de containers com a mesma aplicação, em ambientes isolados, independentes e seguros.

Este projeto tem por objetivo fazer um estudo sobre a tecnologia de containers linux para a execução de aplicações web, em especial o Docker.

1.1 JUSTIFICATIVA

Em razão do Docker, tem-se discutido muito o assunto de containers para execução de aplicações web, com muitos olhares positivos e muita adesão. Acredita-se que é um assunto muito relevante, pois já chamou a atenção dos gigantes da TI, como Google, Red Hat e Microsoft. Já existem inúmeros serviços que utilizam a ideia de container para execução de aplicações e muitas empresas já confiam neste conceito. Inclusive, a grande justificativa que se traz, a princípio, é a vantagem de ter uma aplicação que roda da mesma forma em desenvolvimento, testes, integração e produção. Não existe mais a desculpa que o ambiente não estava identico, pois agora, o ambiente é reduzido a um container linux.

Conforme descrito, os estudos focam a solução Docker, por ser o grande responsável pelo assunto atualmente e por ter chamado tanta atenção dos profissionais de TI, uma vez que o Docker é bem quisto por todas as etapas do desenvolvimento de software.

1.2 OBJETIVOS

Nesta seção são apresentados os objetivos do presente trabalho.

1.2.1 Objetivo geral

Estudo da solução de container linux para a execução de aplicações web.

1.2.2 Objetivos específicos

1. Introdução ao Docker
2. Diferenças do servidor dedicado, virtualizado e container
3. Estudo do funcionamento do Docker

4. Vantagens em executar aplicações em container
5. Exemplos de utilização

1.3 METODOLOGIA

Dizer qual metodologia de trabalho será usada.

1.4 ESTRUTURA DO TRABALHO

Explicar como o trabalho está estruturado.

2 REVISÃO DA LITERATURA

O processo de desenvolvimento de software é um trabalho complexo e depende de muitas etapas até a conclusão <<http://www.devmedia.com.br/atividades-basicas-ao-processo-de-desenvolvimento-de-5413>>.

2.1 Introdução ao Docker

2.1.1 Volumes

Por padrão, os containers são criados com caráter imutável, segundo Nickoloff (2016, p.?). Matthias e Kane (2015, p. 66) ensinam que os containers possuem armazenamento de natureza efêmera e o espaço em disco alocado não é apropriado para a tarefa desejada. Para solucionar estes problemas, existem os volumes. Estes podem ser definidos na criação da imagem e montados no momento da criação do container. Ou seja, para containers que necessitem de armazenamento de dados, é possível criar um volume para armazenar estes dados. Um volume funciona como um container com o único intuito de persistir os dados e disponibilizar se for a vontade desejada.

Conforme Mouat (2015, p. 53), os volumes ainda podem ser utilizados para compartilhar dados entre o host e os containers, bem como entre dois containers. Isso permite o poliforfismo entre os containers, pois imagens criadas de forma genérica podem ser utilizadas em conjunto com volumes para modificar o comportamento dos containers conforme os dados do volume, segundo Nickoloff (2016, p. 74). Uma imagem criada para servir conteúdo estático pode se aproveitar desse padrão de projeto para generalizar o seu uso, uma vez que seu comportamento será alterado conforme for definido o seu volume, por exemplo. É possível definir um volume com propriedade de *somente leitura* para manter o isolamento e assegurar que nenhum dado será modificado. Existem dois tipos de volumes no ecossistema Docker: Volume vinculado à máquina anfitriã e o volume gerenciado pelo Docker.

2.1.1.1 Volume vinculado ao host

Os volumes vinculados ao host são pontos de montagem do disco da máquina anfitriã no disco do container. Funciona como um diretório compartilhado e é uma ótima forma para compartilhar os arquivos entre a máquina anfitriã e o container.

2.1.2 Ambiente de Desenvolvimento e Homologação

Um grande desafio que os desenvolvedores enfrentam é a configuração do ambiente de desenvolvimento, pois o este ambiente acaba por ser simplificado em comparação ao ambiente de produção. É muito comum o software correr de maneira estável em desenvolvimento, mas não obter o mesmo resultado em produção. Seja banco de dados em memória, configurações padrão ou uma arquitetura mais modesta, os ambiente de desenvolvimento são mais simples para garantir que o desenvolvedor não perca tempo com configurações e ajustes e passa a focar no objetivo principal que é a implementação de funcionalidades. Grandes empresas dispõe de um ambiente de homologação que precede o ambiente de produção. Isto ocorre, de maneira geral, para prevenir erros e desacertos entre ambientes. Este procedimento sempre foi muito bem aceito <<http://www.profissionaisti.com.br/2013/06/a-importancia-de-um-ambiente-de-homologacao/>>, mas será que o ambiente de homologação é realmente necessário?

O ambiente de homologação exige, também, uma instalação completa. Exige instalação do software, de suas dependências, alterações de banco de dados e configurações de ambiente. Mesmo que seja utilizada uma ferramenta de automatização para a instalação, um ambiente de homologação exige atenção e manutenção para operar normalmente.

2.1.3 Instalação do Software

Após a implementação de uma ou mais funcionalidade, uma nova versão é gerada e deve ser lançada no ambiente de homologação ou produção. Normalmente este processo é arduo e depende de uma equipe especializada para concluir esta etapa. A equipe de desenvolvimento deve passar detalhes da implementação e as peculiaridades da versão. Qualquer erro ocorrido deve ser reiniciado o processo de instalação e pode provocar consequências irreversíveis, como corrupção de dados. Após a instalação, pode ocorrer a necessidade de ampliar os servidores para servir um tráfego maior de usuários e isso acarreta na instalação em novas máquinas. Mas, é claro, que essa instalação não acontece há tempo de suprir a necessidade de tráfego.

2.1.4 Escalabilidade

O aumento repentino de tráfego em aplicações web demandam muita estratégia da equipe de infraestrutura para que os usuários não sintam lentidão ou não recebam resposta do servidor. Essa estratégia deve conter um rápido ataque para suprir esta necessidade e não pode contar com a instalação de novas máquinas físicas, pois não haveria tempo hábil para tal. Nos dias de hoje, uma empresa que oferece serviços como produto não pode deixar de prestar o serviço sob pena de ter o descrédito pelos usuários e resultar no término dos negócios. Portanto, esses casos exigem que a equipe de infraestrutura possua condições de ampliar o servidor web.

2.1.5 Entregabilidade

A dificuldade de instalar um software normalmente é medida pela frequência que ela ocorre. Quando ocorre frequentemente, deixa a instalação simples e rápida. Se ocorre com pouca frequência, deixa a instalação complexa e demorada. A forma tradicional de instalação normalmente é mais dolorosa e pouco frequente, em razão de diversos fatores como dependências, scripts de migração de banco de dados e etc. Assim, a entregabilidade é prejudicada, pois uma nova funcionalidade leva tempo para ser instalada em produção.

2.1.6 Segurança

Nos servidores físicos são comuns a execução de várias aplicações web na mesma máquina para utilizar o máximo de recurso possível. Entretanto, isso pode ser muito perigoso, pois uma vulnerabilidade em uma aplicação pode dar acesso à máquina que rodam as outras. Da mesma forma, um vazamento de memória pode impedir o bom funcionamento das demais aplicações e acabar derrubando as demais.

2.1.7 O que é Docker?

Docker é uma ferramenta de linha de comando, que é executada em plano de fundo, e promove um servidor remoto para simplificar a experiência de instalar, executar, publicar e remover software, segundo Nickoloff (2016, p.6, tradução livre). Possibilita que um software seja posto em um container, junto com suas dependências, em uma unidade padrão de desenvolvimento de software, conforme Docker (2016, tradução livre). Desta forma, pode-se garantir que o software sempre vai se comportar da mesma maneira, independente do ambiente que for executado.

Em março de 2013, o Docker foi lançado como um projeto de código aberto pela dotCloud, empresa que gerencia um serviço de plataforma na nuvem. Com o intuito de oferecer um serviço melhor e mais competitivo, o fundador e CEO da dotCloud desde 2010, Solomon Hykes, criou o Docker. Em pouco tempo, a ferramenta caiu nos braços da comunidade de desenvolvedores, inclusive grandes empresas de tecnologia como Red Hat, IBM, Google e Cisco, que ajudaram no desenvolvimento do produto, de acordo com TechTarget (2016, tradução livre).

O ano de 2014 foi um ano muito especial para o Docker, pois foi o ano que recebeu grandes investimentos da Greylock Partners e Sequoia Capital e passou a ter um valor de mercado em US\$ 400M (Quatrocentos milhões de dólares). Em 2015, não foi muito diferente, houveram mais duas rodadas de investimentos que totalizou US\$ 180M (Cento e oitenta milhões de dólares), conforme CrunchBase (2016, tradução livre).

Entre as características marcantes da ferramenta pode-se considerar a leveza, pois compartilha recursos do sistema operacional; abertura, pois pode ser executado na grande maioria

dos servidores linux e com versões instáveis para OSX e Windows; seguro, pois o container promove mais uma barreira protetora para o ambiente, segundo Docker (2016, tradução livre).

2.2 Diferenças entre servidor dedicado, virtualizado e containerizado

3 PROCEDIMENTOS METODOLÓGICOS

Este capítulo aborda os métodos e as técnicas utilizadas neste trabalho, de modo que permitiram o tratamento do tema proposto.

O estudo de caso foi feito de maneira descritiva, no qual a literatura foi analisada e interpretada para a elaboração de um parecer final sobre o assunto. O procedimento utilizado foi a pesquisa bibliográfica, para o melhor entendimento do funcionamento da ferramenta e seus efeitos no ambiente de desenvolvimento de software.

Ainda, buscou-se a utilização do método qualitativo para a elaboração do trabalho. Por se tratar de um tema extremamente novo, utilizou-se os livros mais atuais, para manter a coerência entre o estado atual da ferramenta, artigos de documentação da própria ferramenta, que é de excelente qualidade, e artigos na internet que dissertam sobre o tema com um foco mais prático.

4 RESULTADOS E DISCUSSÕES

Neste capítulo, dissertar-se-á sobre as vantagens em utilizar a ferramenta Docker em todas as etapas de desenvolvimento de software, em desenvolvimento, distribuição e publicação.

4.1 Desenvolvimento

Em princípio, notou-se que as aplicações web atuais estão muito complexas e com muitas dependências. Isso torna o desenvolvimento de software um pouco lento, pois são inúmeras configurações que o profissional deve se atentar para poder executar o seu trabalho de maneira desimpedida, segundo Nickoloff (2016, p.14). Para isso, tem-se o Docker Compose, uma ferramenta que auxilia a criação de serviços interconectados, configurável por arquivos *yaml* e que são controláveis por um programa de linha de comando, conforme Nickoloff (2016, p.232).

Além de evitar que o desenvolvedor seja obrigado a instalar serviços como banco de dados, gerenciador de fila, cache, entre outros, o Docker Compose se encarrega de orquestrar estes serviços de forma automatizada e interligadas, utilizando links para simplificar a conexão entre elas, de acordo com Matthias e Kane (2015, p111). Sem sombra de dúvidas, é a melhor maneira de criar um ambiente de desenvolvimento do zero. Isso torna a iniciação de novos desenvolvedores mais rápida e com menos tempo para adaptação, desde que saiba o básico de Docker.

4.2 Distribuição

Para a distribuição do software, sabe-se que é, também, uma tarefa muito árdua, que na grande maioria dos casos necessita de uma equipe especializada no assunto. Com Docker Registry este trabalho foi facilitado, de modo que a distribuição das imagens são feitas de maneira automática e podem ser construídas de maneira muito rápida, com o uso de imagens base, como explica Mouat (2015, p.44). Lembra-se, contudo, que as boas práticas ensinam que as imagens só devem ser publicadas no registry após os testes unitários e de aceitação.

4.3 Publicação

Por fim, o deploy deve ser automático e não deve apresentar problemas se for utilizado o Docker Swarm, que é uma ferramenta de orquestração para construção de cluster de serviços docker, como ensina Nickoloff (2016, p.255). Ele consegue atualizar o cluster de serviços Docker de forma rápida e segura, pois nossos serviços web devem operar com conexão criptografada, segundo Matthias e Kane (2015, p.129).

Desta forma, concluí-se que a ferramenta Docker traz benefícios em todas as etapas de desenvolvimento de software, com a sua distribuição de imagens, publicação em produção e manutenção do cluster de serviços.

5 CONCLUSÃO

As conclusão do trabalho são apresentadas aqui.

REFERÊNCIAS

CRUNCHBASE. *Docker*. 2016. Disponível em: <<https://www.crunchbase.com/organization/docker>>. Acesso em: 19 de maio de 2016. Citado na página 18.

DOCKER. *What is Docker?* 2016. Disponível em: <<https://www.docker.com/what-docker>>. Acesso em: 19 de maio de 2016. Citado 2 vezes nas páginas 18 e 19.

MATTHIAS, K.; KANE, S. *Docker up and running*. [S.l.]: O Reilly Media, Inc, 2015. Citado 2 vezes nas páginas 16 e 21.

MOUAT, A. *Using Docker*. [S.l.]: O Reilly Media, Inc, 2015. Citado 2 vezes nas páginas 16 e 21.

NICKOLOFF, J. *Docker in action*. [S.l.]: Shelter Island, 2016. Citado 3 vezes nas páginas 16, 18 e 21.

TECHTARGET. *brief history of Docker Containers overnight success*. 2016. Disponível em: <<http://searchservvirtualization.techtarget.com/feature/A-brief-history-of-Docker-Containers-overnight-success>>. Acesso em: 19 de maio de 2016. Citado na página 18.

W3TECHS. *W3Techs*. 2016. Disponível em: <https://w3techs.com/technologies/overview/operating_system/all/>. Acesso em: 19 de maio de 2016. Citado na página 13.

APÊNDICE A CÓDIGO FONTE

Código de minha autoria. O apêndice é opcional ao TCC e deve ser elaborado pelo próprio autor. Destina-se a complementar as ideias, sem prejuízo do tema do trabalho. Segue um exemplo:

```
#include <stdio.h>

int main() {
    printf("Ola mundo !\n");
    return 0;
}
```

ANEXO A PESQUISA IBGE

O anexo é opcional ao TCC e são informações não elaboradas pelo próprio autor, mas que tem como objetivo complementar as ideias, sem prejuízo do tema do relatório.