Automação de ambientes utilizando Docker

PROPOSTA DE TRABALHO DE CONCLUSÃO DE CURSO

Jefferson Heckler Stachelski
jeffhsta@riseup.net
Curso de Sistemas de Informação
Centro Universitário Ritter Dos Reis - UNIRITTER

Guilherme Lacerda
Professor Orientador

1. Relevância do trabalho

Muitos desenvolvedores de *software* trabalham usando o seu computador como ambiente principal para rodar e testar o *software* durante o processo de desenvolvimento do *software*. Assim esse ambiente constuma ser diferente dos ambientes dos servidores onde o *software* irá ser instalado, como os ambientes de desenvolvimento, homologação e produção. Com isso em alguns casos ocorrem problemas de imcompatibilidade do *software* entre os ambientes, geralmente causando por versões de sistemas operacionais ou ferramentas diferentes. O uso de *containers* por desenvolvedores é de apenas 17%, 56% estão considerando a adoção da prática e 27% não utilizam e não possuem planos de adoção[2].

A *automação* e *virtualização* de processos de *configuração de ambientes* e de *deploy* tem se tornado uma prática cada vez mais comum[2]. Essa prática acaba minimizando as falhas de imcompatibilidade do *software* nos ambientes no qual esse é implantado[3]. A *virtualização* é uma prática adotada muitas vezes no processo de desenvolvimento fazendo com que o *software* seja desenvolvido e testado sobre esse ambiente virtualizado, afim de que mantenha-se a compatibilidade com o ambiente de produção, em muitos casos usa-se ferramentas de provisionamento do ambiente em maquinas virtuais como por exemplo *Cheff*, *Puppet* entre outros[4]. Com a *virtualização* no processo de desenvolvimento acaba-se reduzindo alguns riscos e até mesmo facilitando quando há alguma mudança no ambiente de produção, por exemplo uma atualização de uma biblioteca ou qualquer outra dependencia do *software*[3].

Porém o consumo de recursos do computador tem se mostrado muito elevado para possuir esse ambiente virtualizado, além de o processo de configuração desse ambiente ser muitas vezes demorado, de acordo com a complexidade e o número de dependências que esse possui. Como alternativa a *virtualização* temos *containers*. *Containers* é o mecanismo utilizado pela ferramente chamada *Docker*, que ao invés de virtualizar um sistema operacional, esse mecanismo utiliza muitos dos componentes ja existentes no computador, e executa outros componentes a fim de isolar os processos executados dentro do *container*[8], fazendo assim essa uma alternativa, mais rápida e que consume menos recursos do computador[2]. Um bom caso de exemplo seria um ambiente onde estão rodando *microservices*, cujos são compostoso por outros serviços externos, no qual cada *microservice* deve ser resiliente, flexivel, mínimo e completo [1]. *Docker* torna-se util também em cenários onde aplicações dependem umas das outras, mas po algum motivo elas executão em diferentes versões de linguagens, biblioteca ou outra dependencia, podendo rodar isoladamente cada aplicação em diferentes *containers*[5].

No contexto abordado acima, é uma área emergente de trabalhos que apontam a vantagem do uso do *Docker* como uma solução alternativa a *virtualização*[7]. Quando trata-se desse tópico geralmente são discutidos ferramentas de automação que criam uma maquina virtual e a configuram, quando que esse processo pode possuir menos tarefas repetitivas e um processo de criação e configuração de um ambiente de forma mais rapido e menos complexa.

2. Objetivos

2.1. Objetivo geral

Este trabalho propõem um estudo de caso da aplicação do *Docker* em aplicação corporativa analisando características não funcionais (performance, segurança, isolamento, tempo de liberação) compado com maquinas virtuais.

2.2. Objetivos específicos

Entre os objetivos específicos deste trabalho pode-se destacar:

- Estudar as ferramentas de virtualização de *software*, como por exemplo, VirtualBox, VWare, Hypervision entre outros. Conhecer as ferramentas existentes no mercado e as mas utilizadas pelas empresas.
- Estudar o *Docker* a nivel prático, entender como funciona a ferramenta e quais são os recursos e funcionalidades disponiveis.
- Criar uma aplicação web simples, contendo pelo menos uma página dinâmica, mas com dependencia de pelo menos duas aplicações externas.
- Automatizar a criação do ambiente do projeto citado acima utilizando uma ferramenta de virtualização previamente estudada.
- Automatizar a criação do ambiente como citado no item acima, porém utilizando o *Docker*. Possuindo assim um cenário onde simula-se a migração de um ambiente virtualizado para utilização de *containers*
- Prover um estudo inicial através de um questionário sobre a diferença entre automatizar ambientes com ferramentas de virtualização e ambientes com *Docker*. Esse questionário será aplicado a desenvolvedores com algum nivel de expericência em *Docker* a fim de prover obter uma visão geral do *Docker* pelos profissionais da area de desenvolvimento de *software*.

3. Solução proposta

Este trabalho visa avaliar os beneficios de utilizar *containers Docker* como método de automatizar os ambientes ao invés do uso de maquinas virtuais. Esse trabalho difere-se da literatura de Kyoung-Taek Seo, H. Hwang, I Moon, O. Kwon and B. Kim[6], pois expecificamente busca-se avaliar a redução de complexidade, desacoplamento dos componentes e a queda de uso de recursos do computador ao utilizar a abordagem de *containers* como base de provisionamento do ambiente de uma aplicação corporativa.

Os experimentos serão conduzidos através de um formulário com perguntas relacionadas a adoção de *containers* por um time de desenvolvimento de *software*, cujo adotou a utilização de *Docker* com a finalidade de melhorar o processo de automação de ambientes de desenvolvimento, testes e produção. O questionário será aplicado a um time de pelo menos cinco desenvolvedores, as perguntas do formulário estarão relacionadas a nivel de complexidade, tempo de resposta e recursos utilizados do computador antes e após a adoção do *Docker* no projeto.

4. Cronograma de desenvolvimento

4.1. Trabalho de conclusão de curso I

A Tabela 1 apresenta o cronograma de desenvolvimento do trabalho conforme a numeração das atividades abaixo:

1. Levantamento das variáveis que diferenciam utilização de maquinas virtuais e *containers*;
2. Elaboração do formulário com as perguntas relavantes com base no item acima;
3. Validação do formulário elaborado com o orientador e desenvolvedores a fim de coletar feedbacks;
4. Elaboração da versão final do formulário;
5. Redação do artigo;
6. Submissão do artigo para a banca;
7. Defesa do trabalho.

Tabela 1. Cronograma de atividades para o trabalho de conclusão I.

Atividade	3/16	4/16	5/16	6/16	7/16
1	x	x			
2		x	x		
3			x	x	
4			x	x	
5				x	x
6				x	x
7					x

4.2. Trabalho de conclusão de curso II

A Tabela 2 apresenta o cronograma de desenvolvimento do trabalho conforme a numeração das atividades abaixo:

1. Definição das ferramentas que serão utilizadas para virtualização e provisionamento;
2. Implementação da aplicação de demonstração com algumas dependências (como descrito nos objectivos específicos2.2);
3. Implementação do processo automatizado de provisionamento do ambiente virtuzalido;
4. Implementação do processo automatizado de provisionamento do ambiente utilizando *Docker*;
5. Análise comparativa dos resultados com relação à trabalhos relacionados;
6. Redação do artigo;
7. Submissão do artigo para a banca;
8. Defesa do trabalho.

Tabela 2. Cronograma de atividades para o trabalho de conclusão II.

Atividade	7/16	8/16	9/16	10/16	11/16	12/16
1	x	x				
2	x	x	x	x		
3		x	x	x	x	
4			x	x	x	
5				x	x	
6			x	x	x	
7					x	
8						x

Referências

[1] J. Bugwadia. Microservices: Five architectural constraints, 2015. 1

[2] J. Esposito. The dzone guide to continuous delivery volume iii, 2016. 1

[3] M. Fowler. Continuous integration. 2006. 1

[4] M. Fowler. Infrastructure as code, 2016. 1

[5] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014. 1

[6] K.-T. Seo, H. Hwang, I. Moon, O. Kwon, and B. Kim. Performance comparison analysis of linux container and virtual machine for building cloud. *Advanced Science and Technology Letters*, 66:105–111, 2014. 2

[7] J. Sitakange. Infrastructure as code: A reason to smile, 2014. 1

[8] ThoughtWorks. Docker for builds, 2015. 1