UNIVERSIDADE FEEVALE

GUILHERME LINDEN

ANÁLISE EM CLOUD COMPUTING PARA APLICAÇÕES WEB: CLUSTER KUBERNETES x MÁQUINAS VIRTUAIS TRADICIONAIS

Anteprojeto de Trabalho de Conclusão

Novo Hamburgo

2020

GUILHERME LINDEN

ANÁLISE EM CLOUD COMPUTING PARA APLICAÇÕES WEB: CLUSTER KUBERNETES x MÁQUINAS VIRTUAIS TRADICIONAIS

Anteprojeto de Trabalho de Conclusão de Curso, apresentado como requisito parcial

à obtenção do grau de Bacharel em

Ciência da Computação pela

Universidade Feevale

Orientador: Juliano Varella de Carvalho

Novo Hamburgo

2020

RESUMO

A computação em nuvem vem ganhando cada vez mais adeptos, em grande parte devido a sua facilidade de utilização aliado ao baixo custo. Porém, os métodos tradicionais de virtualização que sempre foram utilizados pelas empresas quando se é necessário poder computacional, não vem conseguindo acompanhar a agilidade necessária pelos projetos atuais. Assim, foi desenvolvida a tecnologia de *containers* Docker, que aliada aos orquestradores de *cluster*, trazem mais agilidade e confiança para projetos modernos. Docker é um projeto *open-source* que engloba o código e dependências de uma aplicação e permite uma execução rápida e confiável. É uma abstração na camada de aplicação que executa processos isolados. Isso oferece agilidade para o desenvolvimento e execução de aplicações em nuvem, especialmente quando combinado com aplicações que se utilizam de micro serviços. Se utiliza do conceito de *containers* Linux, que foi quem introduziu essa ideia de isolamento dentro do sistema operacional. A tecnologia de *containers* vem sendo cada vez mais utilizada entre empresas de grande porte, por exemplo Twitter e Netflix já utilizam em grande escala em suas aplicações. Para aprimorar o controle sobre essas aplicações em *containers*, a Google em 2014 desenvolveu o Kubernetes, que é um serviço em *cluster* responsável por executar e administrar aplicações em *containers*. Dessa maneira, este trabalho tem como objetivo comparar máquinas virtuais tradicionais com *clusters* Kubernetes, através de testes de *benchmarking* em uma aplicação web e realizar análises relacionadas ao ciclo de desenvolvimento utilizando as duas arquiteturas. Assim, existem dados para verificar as vantagens e desvantagens de se utilizar as novas tecnologias baseadas em *containers* por meio da comparação direta entre as duas arquiteturas estudadas.

Palavras-chave: Docker, Container, Kubernetes, Cloud Computing, Benchmarking.

SUMÁRIO

MOTIVAÇÃO ...........................................................................................................................5

OBJETIVOS ..............................................................................................................................9

METODOLOGIA ....................................................................................................................10

CRONOGRAMA ....................................................................................................................12

BIBLIOGRAFIA ....................................................................................................................13

 MOTIVAÇÃO

Vive-se uma crescente utilização de poder computacional em nuvem, impulsionada em grande parte pela facilidade de uso, aliado a redução dos preços ao longo do tempo, trazendo um grande benefício para o consumidor quando comparado com a estruturação de um *datacenter* próprio (STANOVEVSKA-SLABEVA, 2010). Esse modelo de arquitetura computacional traz benefícios de escalonamento massivo, funciona como serviço e possibilita ao cliente pagar conforme o uso, tirando assim a necessidade da empresa de ter bens imobilizados (STANOVEVSKA-SLABEVA, 2010).

 A computação em nuvem consegue entregar todos os recursos necessários para a execução dos mais variados tipos de sistemas, oferecendo serviços de armazenamento, redes de computadores e poder computacional (STANOVEVSKA-SLABEVA, 2010). Existem então 3 cenários principais onde a computação em nuvem é utilizada: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e também Software como Serviço (SaaS).

* Iaas: nessa modalidade o provedor de serviço divide dinamicamente recursos computacionais como RAM (*random access memory*), CPU (*central processing unit*) e armazenamento, que podem ser utilizados por demanda pelos seus clientes.
* PaaS: a plataforma como serviço adiciona mais uma camada de abstração, entregando um software onde os sistemas do cliente podem funcionar em cima. A plataforma kubernetes é um exemplo de PaaS.
* SaaS: é uma alternativa a executar um software de terceiros localmente, assim esse software que está alocado em uma infraestrutura em nuvem pode ser utilizado pelo cliente remotamente (RODERO-MERINO, 2009).

 A tecnologia que permitiu esse movimento foi a de virtualização tradicional, pois através de um *hypervisor* divide os recursos físicos do servidor em máquinas virtuais totalmente isoladas. Isso permite que diferentes aplicações possam, por exemplo, utilizar diferentes sistemas operacionais. A computação em nuvem abstrai essa camada e entrega uma interface de fácil utilização, podendo assim adicionar ou remover recursos físicos com facilidade (STANOVEVSKA-SLABEVA, 2010).

Essa maneira de provisionar recursos utilizando máquinas virtuais tradicionais já é antiga, e traz alguns problemas para os sistemas e metodologias modernas que estão sendo aplicadas. Esse processo de provisionamento é lento quando comparado a ao provisionamento de um *container* Docker, por necessitar carregar um sistema operacional completo. Essas máquinas virtuais também apresentam baixa portabilidade, dificuldade para escalonar e também lentidão no processo de entrega de uma infraestrutura pronta para produção (ABDULLAH, 2019).

Outro método de provisionamento de infraestrutura é a utilização de *containers* Docker, que diferente das máquinas virtuais realiza virtualização a nível de sistema operacional, criando instâncias isoladas dentro de um mesmo sistema operacional. Os *containers* Docker foram idealizados para servirem apenas uma aplicação, eles consistem de uma imagem em execução com seu próprio sistema de bibliotecas e arquivos de configuração, utilizando-se do kernel Linux. Uma das principais diferenças do paradigma dos *containers* para a virtualização tradicional é que tanto o hardware quanto o sistema operacional são compartilhados entre todos os *containers* renderizados no sistema (MOUAT, 2016).

 Um *container* Docker é fundamentalmente um processo em execução que permite isolar recursos como sistemas de arquivos de outros processos em execução. Esse isolamento é possível devido a utilização de uma imagem Docker que é base para a execução de um *container*. Essas imagens podem ser armazenadas em um serviço de registro para reutilização futura e distribuição. O registro oficial é o Docker Hub, onde podem ser encontradas imagens oficiais para diversos serviços populares, como apache e nginx. (DOCKER INC, 2020). A Figura 1 ilustra como os containers são alocados quando comparados às máquinas virtuais. Cada *container* contém todos os pacotes e bibliotecas necessárias para o seu aplicativo ser executado, totalmente isolados e gerenciados pelo Docker *engine* (MOUAT, 2016).

 Figura 1 - Comparação de estrutura entre máquina virtual e container Docker

Fonte: THE KUBERNETES AUTHORS, 2020

Para gerenciar esses *containers* e prover um alto nível de disponibilidade e confiabilidade, a Google desenvolveu o Kubernetes no ano de 2014. Um *cluster* Kubernetes tem como objetivo orquestrar aplicações baseadas em *containers* em ambientes prontos para produção. Um *cluster* Kubernetes consiste de um número de servidores que em conjunto executam aplicações em *containers*, cada um desses servidores é chamado de nó. Cada nó possui *pods*, que são os componentes que executam os programas. Todos esses nós e *pods* são gerenciados pelo *control plane.* (THE KUBERNETES AUTHORS, 2020)

*Clusters* Kubernetes oferecem diversos serviços na sua plataforma, entre eles os que mais se destacam são:

* Balanceamento de carga e serviço de descoberta: os Kubernetes podem exportar um *container* através de um nome DNS (*domain name service*) ou diretamente pelo endereço IP. Se o tráfego para esse *container* for muito alto, ele consegue fazer o balanceamento de carga distribuindo o tráfego de rede a fim de manter os sistemas estáveis.
* Orquestração de armazenamento: possibilita montar diversos tipos de volumes de armazenamento, como armazenamento local ou *clouds* públicas.
* *Rollout* e *Rollbacks* automatizados: com o *cluster* Kubernetes você pode automatizar a criação e/ou a exclusão de *containers* para o seu processo de *deploy.*
* Distribuição de carga automática: ao ser especificado quanto recurso cada *container* necessita (RAM e CPU), o *cluster* Kubernetes distribui esses *containers* automaticamente entre os nós disponíveis.
* Auto cura: *containers* que apresentam erro são reiniciados automaticamente, ou se necessário, excluídos e criados novamente. (THE KUBERNETES AUTHORS, 2020)

 A comparação entre máquinas virtuais e *containers,* em ambientes com a necessidade de *load balancing,* entre mais de um servidor, já foi abordado no artigo de Abdullah et al. (2019), utilizando uma *cloud* privada e o orquestrador de *containers* *Docker Swarm*. Sendo assim, abre-se espaço para o avanço das pesquisas na área, utilizando tecnologias atuais consideradas padrões de mercado, como por exemplo, utilizar um provedor de *cloud* consolidado como o Google Cloud e comparar com um orquestrador de *containers* como o Kubernetes.

Portanto, esse trabalho visa verificar as diferenças, vantagens e desvantagens da utilização de *containers* dentro um *cluster* Kubernetes, quando comparado aos métodos tradicionais de virtualização e *load balancing.* A fim de mensurar tais diferenças será realizado um *benchmarking* de uma aplicação web, atendida por essas duas arquiteturas, por meio de um laboratório montado em nuvem, como ilustra a Figura 2.

Figura 2 - Esquema da infraestrutura onde os testes serão realizados



Fonte: elaborado pelo autor

OBJETIVOS

Objetivo geral:

 Comparar duas arquiteturas em *Cloud Computing* para servir aplicações web, analisando seus desempenhos, bem como as vantagens e desvantagens delas em relação ao ciclo de desenvolvimento e escalabilidade do software.

Objetivos específicos:

* Apropriar-se do conhecimento sobre as arquiteturas envolvidas no trabalho.
* Investigar ferramentas de *benchmarking*.
* Definir a ferramenta de Benchmarking a ser utilizada.
* Definir as métricas a serem utilizadas.
* Configurar as duas arquiteturas em ambiente *Cloud*.
* Mensurar e comparar o desempenho de cada arquitetura configurada.
* Analisar e comparar a escalabilidade das duas arquiteturas.
* Relatar as vantagens e desvantagens das duas abordagens para o ciclo de desenvolvimento de um software.

METODOLOGIA

A metodologia a ser seguida neste trabalho tem caráter de pesquisa aplicada, onde se fará uso de conhecimentos e métodos existentes com o intuito de alcançar os objetivos propostos. Primeiramente será realizada a pesquisa bibliográfica para o aprofundamento teórico sobre *clusters* Kubernetes e máquinas virtuais em ambientes de *cloud computing*.

Após a pesquisa teórica sobre os assuntos citados será realizada uma pesquisa sobre métodos de *benchmarking,* bem como as ferramentas disponíveis para a sua execução. Desta forma, serão definidos os testes a serem realizados no laboratório.

Com os testes definidos será necessário definir as métricas que serão coletadas e posteriormente analisadas. Essa definição deverá levar em conta o impacto do resultado da métrica para uma aplicação web em produção, também analisando e destacando o comportamento dessa métrica.

Assim que as pesquisas teóricas sobre o tema e a definição dos detalhes de execução dos testes forem finalizadas, será implementado em ambiente de *cloud computing* um laboratório para execução dos testes. Esse laboratório deve implementar uma aplicação web nos dois cenários estudados, a fim de poder comparar métricas entre o *cluster* Kubernetes e máquinas virtuais tradicionais.

Serão realizados testes variados nas duas arquiteturas implementadas no laboratório, sempre seguindo os mesmos procedimentos nos dois cenários para obtermos um resultado comparativo fidedigno. Os testes de *benchmarking* visam analisar diferentes parâmetros (métricas), relacionados a *performance* de uma aplicação web.

Também será realizada uma análise sobre as duas arquiteturas no âmbito do ciclo de desenvolvimento e escalabilidade, verificando as vantagens e desvantagens apresentadas. Essa análise busca verificar características que não são demonstradas através de um teste de performance mas que são relevantes quando realizamos uma comparação entre duas arquiteturas diferentes para uma mesma aplicação.

Devido às etapas de análise e comparação citadas anteriormente, essa é uma pesquisa qualitava e quantitativa, visto que realizaremos análises de dados coletadas em testes de *benchmarking* e análises qualitativas sobre o ciclo de desenvolvimento.

Por fim, será realizada uma análise comparativa dos dois cenários, a fim de compreender qual tem melhores resultados no cenário estudado, levando em consideração as diversas métricas coletadas e também as análises teóricas. É vantajosa a utilização de *clusters* Kubernetes para aplicações web quando temos a opção já consolidada de máquinas virtuais tradicionais?

CRONOGRAMA

Trabalho de Conclusão I

|  |  |
| --- | --- |
| Etapas  | Meses |
| Ago | Set | Out | Nov |
| Anteprojeto |  |  |  |  |
| Pesquisa bibliográfica sobre *cloud computing* |  |  |  |  |
| Pesquisa bibliográfica sobre *containers* Docker e *clusters* Kubernetes |  |  |  |  |
| Pesquisar ferramentas e métodos de *benchmarking*, bem como suas principais métricas |  |  |  |  |
| Selecionar o provedor *cloud* a ser utilizado |  |  |  |  |
| Elaborar TC 1 |  |  |  |  |

Trabalho de Conclusão II

|  |  |
| --- | --- |
| Etapas  | Meses |
| Mar | Abr | Mai | Jun |
| Montar a infraestrutura em nuvem  |  |  |  |  |
| Configurar a aplicação web |  |  |  |  |
| Realizar os testes de *benchmarking* na aplicação web |  |  |  |  |
| Analisar e comparar as duas arquiteturas em relação ao ciclo de desenvolvimento |  |  |  |  |
| Analisar e comparar as métricas geradas pelo teste de *benchmarking* |  |  |  |  |
| Montar uma visualização clara para essas métricas analisadas |  |  |  |  |
| Elaborar TC II |  |  |  |  |

BIBLIOGRAFIA

STANOVEVSKA-SLABEVA, K.; WOZNIAK, T. **Cloud Basics - An Introduction to Cloud Computing**. Grid and Cloud Computing, Springer, Berlin, p. 47 - 61, 2010.

RODERO-MERINO, L.; VAQUERO, L. **A Break in the Clouds: Towards a Cloud Definition**. ACM SIGCOMM Computer Communication Review, p. 50 - 54, 2009

ABDULLAH, M; IQBAL, W; BUCKARI, F. **Containers vs Virtual Machines for Auto-scaling Multi-tier Applications Under Dynamically Increasing Workloads**. International Conference on Intelligent Technologies and Applications,2019.

MOUAT, Adrian.**Usando Docker: Desenvolvendo e implantando software com containers**. [s.l.] O’Reilly Media, 2015

DOCKER INC. **What is a Container?**. Disponível em: <https://www.docker.com/resources/what-container/>. Acesso em: 2 ago. 2020

RAD, B; BHATTI, H; AHMADI, M. **An Introduction to Docker and Analysis of its Performance**. IJCSNS International Journal of Computer Science and Network Security, p. 228-235, 2017..

THE KUBERNETES AUTHORS. **What is Kubernetes?**. Disponível em: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>. Acesso em: 7 ago. 2020

ALKSNIS G; SILVA, V; KIRIKOVA, M. **Containers for Virtualization: An Overview**. The Journal of Applied Computer Systems, p. 21 - 27, 2018.