FACULDADE DE TECNOLOGIA DE AMERICANA
Curso de Tecnologia em Segurança da Informação

José Guilherme Guimarães de Oliveira 0040971621034
Gabriel e Silva Botelho 0040971711026

KUBERNETES

Americana, SP
2019

FACULDADE DE TECNOLOGIA DE AMERICANA
Curso de Tecnologia em Segurança da Informação

José Guilherme Guimarães de Oliveira 0040971621034
Gabriel e Silva Botelho 0040971711026

KUBERNETES

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso de Tecnologia em Segurança da Informação, sob orientação do(a) .

Área de concentração: Desenvolvimento de Jogos

Americana, SP
2019

RESUMO

O Resumo é um elemento obrigatório em tese, dissertação, monografia e TCC, constituído de uma sequencia de frases concisas e objetivas, fornecendo uma visão rápida e clara do conteúdo do estudo. O texto deverá conter no máximo 500 palavras e ser antecedido pela referência do estudo. Também, não deve conter citações. O resumo deve ser redigido em parágrafo único, espaçamento simples e seguido das palavras representativas do conteúdo do estudo, isto é, palavras-chave, em número de três a cinco, separadas entre si por ponto e finalizadas também por ponto. Usar o verbo na terceira pessoa do singular, com linguagem impessoal, bem como fazer uso, preferencialmente, da voz ativa. Texto contendo um único parágrafo.

Palavras-chave: Palavra. Segunda Palavra. Outra palavra.

ABSTRACT

Elemento obrigatório em tese, dissertação, monografia e TCC. É a versão do resumo em português para o idioma de divulgação internacional. Deve ser antecedido pela referência do estudo. Deve aparecer em folha distinta do resumo em língua portuguesa e seguido das palavras representativas do conteúdo do estudo, isto é, das palavras-chave. Sugere-se a elaboração do resumo (Abstract) e das palavras-chave (Keywords) em inglês; para resumos em outras línguas, que não o inglês, consultar o departamento / curso de origem.

Keywords: Word. Second Word. Another word.

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação

SUMÁRIO

1 – INTRODUÇÃO 1
1.1 JUSTIFICATIVA 1
1.2 OBJETIVO 1
1.3 PROBLEMÁTICA 1
1.4 METODOLOGIA 1
1.5 ORGANIZAÇÃO DO TRABALHO 1

2 – FUNDAMENTAÇÃO TEÓRICA 2
2.1 VIRTUALIZAÇÃO 2
2.1.1 *HYPERVISOR* 2
2.1.2 VIRTUALIZAÇÃO COMPLETA E PARCIAL 2
2.1.3 VIRTUALIZAÇÃO DE HARDWARE 3
2.2 COUNTAINER LINUX 3
2.3 COUNTAINER ENGINE 4
2.4 ORQUESTRAÇÃO DE COUNTAINER 4

3 – KUBERNETES 5

4 – CONSIDERAÇÕES FINAIS 6

Referências 7

1 INTRODUÇÃO

Edite e coloque aqui o seu texto de introdução.

A Introdução é a parte inicial do texto, na qual devem constar o tema e a delimitação do assunto tratado, objetivos da pesquisa e outros elementos necessários para situar o tema do trabalho, tais como: justificativa, procedimentos metodológicos (classificação inicial), embasamento teórico (principais bases sintetizadas) e estrutura do trabalho, tratados de forma sucinta. Recursos utilizados e cronograma são incluídos quando necessário. Salienta-se que os procedimentos metodológicos e o embasamento teórico são tratados, posteriormente, em capítulos próprios e com a profundidade necessária ao trabalho de pesquisa.

1.1 JUSTIFICATIVA

1.2 OBJETIVO

1.3 PROBLEMÁTICA

1.4 METODOLOGIA

1.5 ORGANIZAÇÃO DO TRABALHO

Normalmente ao final da introdução é apresentada, em um ou dois parágrafos curtos, a organização do restante do trabalho acadêmico. Deve-se dizer o quê será apresentado em cada um dos demais capítulos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 VIRTUALIZAÇÃO

Virtualização, basicamente, é a técnica de separar aplicação e sistema operacional dos componentes físicos. Por exemplo, uma máquina virtual possui aplicação e sistema operacional como um servidor físico, mas estes não estão vinculados ao software e pode ser disponibilizado onde for mais conveniente. Uma aplicação deve ser executada em um sistema operacional em um determinado software. Com virtualização de aplicação ou apresentação, estas aplicações podem rodar em um servidor ou ambiente centralizado e ser deportada para outros sistemas operacionais e hardwares.

2.1.1 *HYPERVISOR*

O *hypervisor* é um *firmware* ou hardware que cria e roda máquinas virtuais (*VMs*). O computador no qual o *hypervisor* roda uma ou mais VMs é chamado de máquina hospedeira (*host*), e cada *VM* é chamada de máquina convidada (*guest*). O *hypervisor* se apresenta aos sistemas operacionais convidados como uma plataforma de virtualização e gerencia a execução dos sistemas operacionais convidados.

Existem dois tipos de hypervisor, o primeiro tipo é conhecido como bare matal, onde o próprio sistema operacional que gerencia as VM, podemos citar o VMware: ESX, Xem, CubeOS, Microsoft Hyper-V. O segudo modelo é o hosted onde o hypervisor se encontra encima do sistema operacional, seja Linux, Windowns ou MacOS, podemos citar o virtualbox, VMware: Workstation, QEMU, OVirt.

2.1.2 VIRTUALIZAÇÃO COMPLETA E PARCIAL

Como o próprio nome sugere, a virtualização completa realiza toda a abstração do sistema físico, com o objetivo de fornecer ao sistema operacional hóspede uma réplica do hardware virtualizado pelo hospedeiro. Este tipo dispensa a necessidade de modificar o SO convidado, que trabalha desconhecendo que há virtualização.

Com a virtualização total, as instruções não críticas são executadas diretamente no hardware, enquanto as instruções críticas são interceptadas e executadas pela hypervisor. O sistema operacional visitante, no entanto, sequer tem o conhecimento de que está sendo executado sobre o hypervisor.

Jã um SO convidado paravirtualizado tem a assistência de um compilador inteligente que atua na substituição de instruções de SO não virtualizáveis por hiperchamadas (hypercalls) quando for executar uma instrução sensível. Tal procedimento poupa o desempenho, quando comparado ao que foi descrito na virtualização total.

Em relação aos dispositivos de E/S, a paravirtualização permite que as máquinas virtuais usem os drivers do dispositivo físico real sob o controle do hipervisor, o que reduz os problemas de compatibilidade.

2.1.3 VIRTUALIZAÇÃO DE HARDWARE

2.2 COUNTAINER LINUX

Um container Linux é um conjunto de um ou mais processos organizados isoladamente do sistema. Todos os arquivos necessários à execução de tais processos são fornecidos por uma imagem distinta. Na prática, os containers Linux são portáteis e consistentes durante toda a migração entre os ambientes de desenvolvimento, teste e produção. Essas características os tornam uma opção muito mais rápida do que os pipelines de desenvolvimento, que dependem da replicação dos ambientes de teste tradicionais.

Embora os containers não tenham se originado no Linux, é no mundo open source que as melhores ideias são emprestadas, modificadas e aprimoradas. Foi o que aconteceu com os containers.

A ideia do que atualmente chamamos de tecnologia de containers surgiu inicialmente no ano 2000 como jails do FreeBSD, uma tecnologia que permite particionar um sistema FreeBSD em vários subsistemas ou celas (por isso o nome "jails"). Os jails foram desenvolvidos como ambientes seguros que podiam ser compartilhados por um administrador de sistemas com vários usuários internos ou externos à empresa. O propósito do jail era a criação de processos em um ambiente modificado por chroot (no qual o acesso ao sistema de arquivos, rede e usuários é virtualizado), que não pudesse escapar ou comprometer o sistema como um todo. A implementação dele era limitada, e os métodos de escape do ambiente em jail foram descobertos com o tempo.

Em pouquíssimo tempo, mais tecnologias foram combinadas para tornar essa abordagem isolada uma realidade. Os grupos de controle (cgroups) são uma funcionalidade de kernel que controla e limita o uso de recursos por um processo ou grupo de processos. E o systemd, um sistema de inicialização que configura o espaço do usuário e gerencia processos, é usado pelo cgroups para dar mais controle sobre os processos isolados. Ambas as tecnologias, além de adicionarem um controle geral ao Linux, serviram como estrutura para a separação eficaz de ambientes.

Os avanços em namespaces de kernel representaram a próxima etapa na criação dos containers. Com os namespaces de kernel, absolutamente tudo, desde IDs de processos a nomes de rede, puderam ser virtualizados em um kernel Linux. Um dos avanços mais recentes, os namespaces de usuários "permitem realizar mapeamentos de IDs de usuários e grupos por namespace. No contexto dos containers, isso significa que os usuários e grupos podem ter privilégios para realizar determinadas operações dentro de um container, sem ter esses mesmos privilégios fora dele". O projeto Linux Containers (LXC) contribuiu com as ferramentas,

bibliotecas, associações de linguagens e modelos necessários para esses avanços, o que melhorou a experiência do usuário na utilização de containers. O LXC tornou mais fácil para os usuários iniciar containers com uma interface de linha de comando simples.

2.3 COUNTAINER ENGINE

O Docker adicionou muitos dos novos conceitos e ferramentas: uma interface de linha de comando simples para executar e criar novas imagens em camadas, um daemon de servidor, uma biblioteca de imagens de container pré-criadas e o conceito de servidor de registros. Combinadas, essas tecnologias possibilitaram aos usuários criar novos containers em camadas com rapidez e compartilhá-los facilmente com outras pessoas.

Com o Docker, é possível lidar com os containers como se fossem máquinas virtuais modulares e extremamente leves. Além disso, os containers oferecem maior flexibilidade para você criar, implantar, copiar e migrar um container de um ambiente para outro.

2.4 ORQUESTRAÇÃO DE COUNTAINER

3 KUBERNETES

O Kubernetes é uma plataforma portátil, extensível e de código aberto para gerenciar cargas de trabalho e serviços em contêiner, que facilita a configuração declarativa e a automação. Possui um ecossistema grande e de rápido crescimento. Os serviços, suporte e ferramentas do Kubernetes estão amplamente disponíveis.

O nome Kubernetes é originário do grego, significando timoneiro ou piloto. O Google deu origem ao projeto Kubernetes em 2014. O Kubernetes se baseia em uma década e meia de experiência que o Google tem em executar cargas de trabalho de produção em grande escala , combinadas com as melhores idéias e práticas da comunidade.

Para trabalhar com o Kubernetes, use os objetos da API do Kubernetes para descrever o estado desejado do cluster : quais aplicativos ou outras cargas de trabalho você deseja executar, quais imagens de contêineres eles usam, o número de réplicas, quais recursos de rede e disco você deseja disponibilizar e Mais. Você define o estado desejado criando objetos usando a API do Kubernetes, normalmente por meio da interface da linha de comandos kubectl. Você também pode usar a API Kubernetes diretamente para interagir com o cluster e definir ou modificar o estado desejado.

Depois de definir o estado desejado, o Kubernetes Control Plane faz com que o estado atual do cluster corresponda ao estado desejado por meio do Pod Lifecycle Event Generator (PLEG). Para fazer isso, o Kubernetes executa várias tarefas automaticamente - como iniciar ou reiniciar contêineres, dimensionar o número de réplicas de um determinado aplicativo e muito mais.

4 CONSIDERAÇÕES FINAIS

Referências