Implementação em nuvem de uma aplicação IoT gerenciada por containers

PSI5120 - Tópicos de Computação em Nuvem

Ana Emília Hernandes Dib
Henrique Rosa
Yaney Gomez Correa

Agosto de 2023

1 Introdução

Atualmente muito se fala em computação em nuvem e Internet das Coisas. Neste trabalho, apresentamos estes conceitos e a hospedagem em um ambiente de contêiner de uma aplicação genérica de Internet das Coisas para fins de atividade didática.

1.1 Virtualização

Virtualização é uma técnica na área de tecnologia da informação que permite que um sistema computacional físico seja dividido em vários outros, chamados virtuais [1]. Cada máquina virtual possui um ambiente próprio, segregado de outros ambientes virtuais da máquina física.

A virtualização proporciona uma grande flexibilidade ao permitir que diversas aplicações sejam executadas em um mesmo *hardware* simultaneamente. Atualmente, a importância e quantidade de aplicações da virtualização aumentou bastante, desse modo os processadores recentes (Intel e AMD) já integram soluções de suporte à virtualização em sua arquitetura.

1.2 Computação em nuvem

Computação em nuvem é uma tecnologia que visa oferecer serviços de tecnologia da informação sob demanda. Esses serviços são oferecidos por meio de máquinas virtuais geralmente localizadas em grandes centros estrategicamente localizados pelo mundo, e permitem o acesso remoto do cliente ao ambiente virtual.

Exemplos de serviços em nuvem incluem:

- Software como Serviço (SaaS)
- Plataforma como Serviço (PaaS)
- Infraestrutura como Serviço (IaaS)

O Software como Serviço permite hospedar aplicações hospedadas na nuvem que podem ser acessadas e consumidas por usuários via internet. Como exemplos, podemos citar serviços de email e de calendário.

A Plataforma como Serviço funciona como um ambiente de desenvolvimento e testes, que inclui infraestrutura, servidores, armazenamento e rede, entre outros recursos úteis nesse contexto. Seu objetivo é dar suporte a outras aplicações e suas etapas: compilação, teste, implantação, gerenciamento e atualização.

A Infraestrutura como Serviço tem como objetivo fornecer recursos computacionais, de rede e armazenamento sob demanda, permitindo que o usuário escale a quantidade de recursos sob demanda. Com esse aprovisionamento flexível, é possível reduzir custos operacionais, visto que os recursos só são alocados conforme a demanda do usuário.

1.3 Conteinerização

Contêineres são um tipo de tecnologia de virtualização que permite empacotar, distribuir e executar aplicativos e suas dependências de forma isolada em um ambiente consistente. Eles encapsulam todos os componentes necessários para executar um aplicativo, incluindo bibliotecas, códigos e configurações, em um único pacote chamado de contêiner. Isso proporciona portabilidade, consistência e eficiência ao implantar aplicativos em diferentes ambientes, como desenvolvimento, teste e produção.

Contêineres de computação, como Docker e Kubernetes, surgiram como tecnologias promissoras para gerenciar e implantar aplicativos em sistemas IoT. Esses contêineres encapsulam componentes de software junto com suas dependências, fornecendo isolamento, portabilidade e escalabilidade [2].

1.4 Internet das Coisas

Nos últimos anos, a integração de soluções de Internet das Coisas (IoT) revolucionou vários domínios, desde casas inteligentes, cuidados de saúde à automação industrial, entre muitos outros. Quantidades crescentes de dispositivos, sensores e atuadores interconectados geram grandes quantidades de dados que exigem arquiteturas eficientes e escaláveis que lidam com a complexidade e diversidade desses sistemas.

O termo Internet das Coisas (*Internet of Things*, ou *IoT*, em inglês) é usado atualmente para descrever cenários em que objetos, dispositivos ou sensores possuem conexão com a Internet e capacidade computacional [3].

1.4.1 Protocolos IoT

Muitos padrões de IoT são propostos para facilitar e simplificar o trabalho de programadores de aplicativos e provedores de serviços. Diferentes grupos foram criados para fornecer protocolos de suporte à IoT, incluindo esforços liderados pelo World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPCglobal, Institute of Electrical and Electronics Engineers (IEEE) e European Telecommunications Standards Instituto (ETSI) [4].

Nas implementações de IoT, os protocolos sem fio são mais comumente usados, podem ser instalados mesmo em locais que não possuem os principais requisitos para sensores com fio, como energia, cabeamento de comunicação, etc, além disso, em uma rede de sensores sem fio, é mais fácil para os nós serem adicionados, removidos ou realocados sem reconsiderar a estrutura de toda a rede. Em outras implementações, no entanto, uma rede de sensores com fio é preferida, pois essas redes são mais confiáveis, mais seguras e oferecem velocidades de transmissão de dados mais altas. Por exemplo, em implementações de IoT em um hospital, onde a confiabilidade e a velocidade são fatores importantes para salvar a vida de um paciente, os sensores com fio são preferíveis e os requisitos para sua instalação podem ser planejados

durante o projeto inicial do hospital (cabeamento, cabos de fornecimento de energia, etc.) [5].

De acordo com [3], existem quatro modelos básicos de comunicação utilizados com dispositivos IoT: comunicações dispositivo para dispositivo, dispositivo para nuvem, dispositivo para ponte de ligação (*gateway*, em inglês), e modelo de compartilhamento de processos internos.

A Figura 1 mostra os protocolos usados em uma arquitetura IoT (ITU-T (International Telecommunications Union - Telecommunication Standardization Sector) composta por quatro camadas; a camada superior ou primeira é a camada de aplicativo IoT que contém a interface do usuário do aplicativo, a segunda camada são os serviços e camada de suporte de aplicativo, a terceira camada é a camada de rede que contém os recursos de rede e transporte, a camada inferior é a camada de dispositivo, que contém os gateways e o hardware e sensores e tags RFID e outros. Os protocolos descritos a seguir

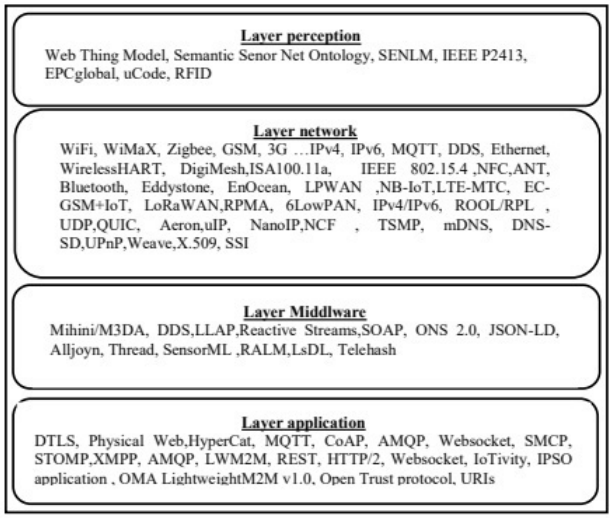

Figura 1: Protocolos de IoT. Fonte: [4].

são alguns dos mais utilizados na camada de aplicação:

- MQTT: O protocolo Message Queuing Telemetry Transport (MQTT) é um protocolo de mensagens para publicação e assinatura que funciona no modelo cliente/servidor muito simples e é executado sobre TCP/IP ou outros protocolos. É mais adequado para ambientes restritos, como

em IoT, porque é aberto, leve e facilmente implementável. Em infraestruturas críticas e aplicações com informações sensíveis, o MQTT pode funcionar e oferecer serviços avançados de segurança com o uso de recursos específicos recomendados.

- CoAP: O Protocolo de Aplicação Restrita (CoAP) é definido como um protocolo de transferência web especializado na RFC 7252. É um protocolo leve, com baixa taxa de transmissão, proposto para uso com nós restritos e redes restritas. O design é apropriado para aplicações máquina-a-máquina (M2M), como gerenciamento da cadeia de suprimentos e medidores inteligentes para rastrear o consumo de energia. Ele pode interagir muito bem com HTTP, o que facilita a integração com a Web. Mas o CoAP não é um protocolo seguro, e isso é uma séria desvantagem.

- REST: O Representational State Transfer (REST) é um estilo arquitetônico híbrido para sistemas hipermídia distribuídos. Inclui um conjunto de regras que descrevem os princípios orientadores da engenharia de software para construir um aplicativo com certas restrições. É utilizado para a construção de serviços web, também chamados de RESTful. REST inclui a) a restrição cliente-servidor, b) a restrição sem estado, que alcança visibilidade, confiabilidade e escalabilidade, c) a restrição de cache, que melhora a eficiência da rede, d) um conjunto de quatro restrições para uma interface uniforme entre os componentes, e) restrições do sistema em camadas, e f) a restrição opcional do código sob demanda.

- XMPP: O Extensible Messaging and Presence Protocol (XMPP) é uma tecnologia XML aberta para comunicação em tempo real. É usado para mensagens instantâneas, presença e colaboração. Presença especifica que uma entidade está pronta para envio de mensagens. O design aberto do XMPP facilita mudanças e permite seu recurso extensível, que atende a uma implementação de IoT.

- AMQP: O Advanced Message Queuing Protocol (AMQP) é um padrão aberto adequado para mensagens de negócios entre aplicativos, que opera de forma assíncrona em diferentes organizações e plataformas. É um protocolo de nível de fio que permite mensagens comerciais confiáveis. Algumas das principais características incluídas no design do AMQP visam garantir segurança, confiabilidade e interoperabili-

dade.

A seleção dos protocolos a serem utilizados pode ser baseada em diversos fatores como heterogeneidade do hardware, consumo de energia, velocidade de transmissão, distância de transmissão necessária em cada aplicação e muitos outros.

Protocol Name	Network Standard	Latest Version (Year)	Transport Protocol	Messaging Model	Architecture	Security and QoS	Application Levels
CoAP	IETF, Eclipse Foundation	RFC 8323 (2018)	UDP	Request/response	Tree	Both	Utility field
MQTT	OASIS, Eclipse Foundations	MQTT version 5.0 (2018)	TCP	Publish/subscribe and request/response	Tree	Both	IoT messaging
XMPP	(RFC 3920-RFC 3923) RFC 4622, RFC 4854, RFC 4979, RFC 6122	XMPP v 1.0.1, XEP-0128 (2019)	TCP	Publish/subscribe and request/response	Client-server	Security	Remote management
AMQP	OASIS, ISO/IEC	AMQP v 2.5.0 (2019)	TCP	Publish/subscribe	P2P	Both	Enterprise integration
DDS	OMG	DDS v.1.4 (2015)	TCP/UDP	Publish/subscribe and request/response	Bus	QoS	Military
LoWPAN	IEEE 802.15.4	6Lo-BLEMesh (2019)	TCP	Publish/subscribe	Star, mesh	Security	Structural monitoring
BLE	802.15.1	6Lo-BLEMesh (2019), MRT-BLE (2018)	TCP/UDP	Publish/subscribe and request/response	Star	Security	Wearable devices
Zigbee	IEEE 802.15.4	Zigbee 3.0 (2018)	UDP	Publish/subscribe and request/response	Star, mesh, hybrid	Security	Consumer electronics

Figura 2: Características de alguns protocolos de comunicação IoT. Fonte: [6].

1.4.2 Aplicaçãoes da IoT

A Internet das Coisas pode ser usada em muitos aspectos diferentes da vida, tanto no setor privado quanto no público. De acordo com a Cisco, mais de 500 bilhões de dispositivos serão conectados à Internet até 2030 [7]. A versatilidade da IoT a torna uma opção atraente para tantas empresas, organizações, órgãos governamentais e o cotidiano das pessoas que estão incorporando cada vez mais aplicativos de acordo com suas necessidades.

Atualmente, há um número crescente de plataformas oferecendo soluções IoT como Oracle IoT, Google Cloud IoT, Amazon AWS IoT Core e Microsoft Azure IoT Hub. Existem muitos campos de aplicações de IoT, alguns mais fortemente desenvolvidos do que outros, podemos citar aplicações em: agricultura, uso do consumidor, manufatura, varejo, transporte, serviços públicos, energia, vestuário, monitoramento de tráfego, hospitalidade, smart

grid e economia de energia, abastecimento de água, casas inteligentes, cidades inteligentes, controle inteligente de poluição, entre outros.

1.4.3 IoT em contêineres

A integração de contêineres em sistemas de Internet das Coisas (IoT) cria muitos desafios e oportunidades na computação em nuvem. O gerenciamento de recursos é o principal fator para utilizar totalmente o potencial da computação Edge/Fog para executar aplicativos IoT críticos e em tempo real. No entanto, o gerenciamento, a implantação, a verificação de integridade e a escalabilidade de um grande número de contêineres são questões desafiadoras [8]. A tecnologia de contêineres está sendo predominantemente usada pela indústria e pela academia para executar inumeráveis aplicações, incluindo sistemas IoT. Em comparação com as máquinas físicas e virtuais, os contêineres são leves, fáceis de implantar e permitem tempos curtos de inicialização, expansão e migração.

2 A aplicação IoT

O uso de contêineres na Internet das Coisas (IoT) traz uma abordagem inovadora para superar desafios complexos. Em um mundo repleto de dispositivos variados e sistemas operacionais distintos, os contêineres fornecem uma maneira consistente e eficiente de empacotar aplicativos e seus requisitos. Eles isolam esses aplicativos, garantindo que eles funcionem uniformemente em diferentes dispositivos, enquanto preservam a segurança e a escalabilidade.

A ideia central dos contêineres é a criação de ambientes autossuficientes que englobam tudo o que um aplicativo precisa para funcionar. Isso elimina a complexidade de configurar cada dispositivo individualmente. Quando aplicado a IoT, o uso de contêineres é particularmente valioso em dispositivos de borda, como gateways ou Raspberry Pi. Nesses dispositivos, os contêineres hospedam módulos que processam dados de sensores e executam análises preliminares. Esse processamento local reduz a latência, melhora a privacidade dos dados e alivia a carga da rede, já que apenas informações relevantes são enviadas para a nuvem.

Ao utilizar contêineres em IoT, cria-se uma arquitetura flexível em que os dispositivos de borda executam tarefas críticas e tomam decisões locais, en-

quanto a nuvem é usada para análises mais complexas e armazenamento a longo prazo. Essa abordagem híbrida combina a agilidade dos dispositivos de borda com a capacidade de processamento e armazenamento escaláveis da nuvem.

A aplicação prática de contêineres em IoT é ilustrada quando consideramos um cenário onde um sensor de temperatura e umidade em uma fábrica é conectado a uma Raspberry Pi. A Raspberry Pi, usando contêineres, pode processar os dados coletados pelo sensor localmente, executando análises e decisões relevantes. Isso permite que apenas informações pertinentes sejam enviadas para a nuvem, economizando recursos de rede e otimizando a comunicação entre a borda e a nuvem. Em última análise, a utilização de contêineres em IoT simplifica a complexidade do desenvolvimento, implantação e gerenciamento de aplicativos, oferecendo uma abordagem eficaz para lidar com a diversidade de dispositivos e sistemas operacionais encontrados no cenário de IoT.

3 Tutorial

Para o Hands-on criaremos um aplicativo para monitorar temperatura e umidade de um dispositivo Raspberry PI usando contêineres.

3.1 Pré-requisitos

- Uma conta do Azure e uma inscrição com créditos. Criar uma conta gratuita em: `https://azure.microsoft.com/pt-br/free/`
- Dispositivo IoT Edge (usamos um Raspberry Pi Model B).

3.2 Hands-on

O roteiro será o seguinte [1]:

1. Criar um IoT Hub.

[1] Para as linhas de comando iniciadas em "az", podem ser usados o terminal do Linux, do Power Shell ou Cloud Shell no Portal do Azure. Para os dois primeiros, é necessário instalar o Azure CLI. (`https://learn.microsoft.com/en-us/cli/azure/install-azure-cli`).

2. Registrar um dispositivo Edge no IoT Hub.
3. Obter a Connection String usada pelo dispositivo.
4. Instalar o Edge Runtime no dispositivo Edge.
5. Configurar o dispositivo com sua Connection String.
6. Realizar o deploy de módulos para o dispositivo.
7. Monitorar as mensagens vindas do módulo no IoT Hub.

3.2.1 Criar um IoT Hub

Azure portal: `https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal?view=iotedge-1.4#create-an-iot-hub`

- Fazer login na conta Azure:

```
$ az login
```

- Adicionar a extensão azure-iot para gerenciar serviços de IoT pelo CLI:

```
$ az extension add --name azure-iot
```

- Criar um grupo de recursos:

```
$ az group create -l brazilsouth -n MyResourceGroup
```

- Criar um IoT Hub:

```
$ az iot hub create --resource-group
  MyResourceGroup --name MyIotHub --sku F1 --
  partition-count 2
```

3.2.2 Registrar um dispositivo Edge no IoT Hub

- Criar um dispositivo no Iot Hub:

```
$ az iot hub device-identity create --device-id
  MyRasp --hub-name MyIotHub --edge-enabled
```

3.2.3 Obter a Connection String usada pelo dispositivo

- Recuperar a string de conexão:

```
$ az iot hub device-identity connection-string show
    --device-id MyRasp --hub-name MyIotHub
```

3.2.4 Instalar o Edge Runtime no dispositivo Edge (Raspberry Pi)

- Baixar os pacotes de configuração da Microsoft:

```
$ curl https://packages.microsoft.com/config/debian
   /stretch/multiarch/prod.list > ./microsoft-prod.
   list
```

- Copiar para o diretório correto (sources.list.d):

```
$ sudo cp ./microsoft-prod.list /etc/apt/sources.
   list.d/
```

- Instalar a chave pública GPG Microsoft:

```
$ curl https://packages.microsoft.com/keys/
   microsoft.asc | gpg --dearmor > microsoft.gpg
```

```
$ sudo cp ./microsoft.gpg /etc/apt/trusted.gpg.d/
```

- Atualizar os repositórios:

```
$ sudo apt-get update && sudo apt-get upgrade
```

- Baixar o script de instalação do Docker:

```
$ curl -fsSL https://get.docker.com -o get-docker.
   sh
```

```
$ sudo sh get-docker.sh
```

- Adicionar o usuário ao grupo Docker:

```
$ sudo usermod -aG docker Pi
```

- Atualizar as mudanças:

```
$ logout
```

- Para verificar se a instalação foi bem sucedida e testar o funcionamento correto do Docker:

```
$ docker version
```

```
$ docker run hello-world
```

- Instalar o IoT Edge:

```
$ sudo apt-get install aziot-edge --fix-missing
```

3.2.5 Configurar o Raspberry com a string de conexão

- Configurar a conexão com a string definida anteriormente

```
$ sudo iotedge config mp --connection-string '
  PASTE_DEVICE_CONNECTION_STRING_HERE'
```

- Atualizar as mudanças

```
$ sudo iotedge config apply
```

- Verificar o status do serviço IoT Edge

```
$ sudo iotedge system status
```

- Verificar as conexões e conectividade

```
$ sudo iotedge check
```

- Visualizar os módulos em execução

```
$ sudo iotedge list
```

3.2.6 Realizar o deploy de módulos para o dispositivo

Azure portal: `https://azuremarketplace.microsoft.com/en-us/marketplace/apps/category/internet-of-things?page=1&subcategories=iot-edge-modules`

- No Azure Portal clique em **Set Modules**. Vá na seção **IoT Edge Modules**, clique em **Add** e selecione a opção **Marketplace Module** para buscar uma imagem do Marketplace. Digite **Simulated** na janela de busca e selecione o módulo **Simulated Temperature Sensor.**
- Na seção **IoT Edge Modules**, clique em **"Runtime Settings"** e modifique a versão dos módulos Edge Agent e Edge Hub para a mais recente e aplique as mudanças.
- Na seção **"Routes"** temos duas rotas já pré-configuradas. A primeira é a padrão e direciona todas as mensagens de qualquer módulo emissor de mensagens para $upstream, que é o IoT Hub. A segunda rota foi criada automaticamente quando selecionamos o módulo de simulação do Marketplace e direciona o que vem deste módulo para o IoT Hub. Como essas rotas acabam sendo redundantes, podemos excluir a primeira clicando no ícone da lixeira ao lado da rota.
- Clique em Review + Create. Nessa etapa o IoT Hub terá construído o arquivo "Deployment Manifest"[2] no formato JSON (Edge Agent é o módulo responsável pelo monitoramento e gerenciamento dos módulos e o edgeHub com a especificação das rotas que gerencia a comunicação interna e externa)
- Verificar status dos módulos no dispositivo:

```
$ sudo iotedge list
```

```
$ sudo iotedge check
```

3.2.7 Monitorar as mensagens vindas do módulo no IoT Hub

- Verificar os módulos no dispositivo:

```
$ sudo iotedge logs SimulatedTemperatureSensor -f
```

- Verificar os eventos no IoT Hub pelo computador:

```
$ az iot hub monitor-events -n IOT_HUB_NAME
```

[2]Para o deploy de módulos no dispositivo será utilizado o arquivo "Deployment Manifest"do IoT Hub.

3.2.8 Limpar o ambiente de trabalho e liberar os recursos

1. Desinstalar o IoT Edge Runtime do Raspberry

```
$ sudo apt-get remove --purge aziot-edge
```

2. Listar containers

```
$ docker ps -a
```

3. Deletar contêiners

```
$ sudo docker rm -f CONTAINER ID
```

4. Deletar containers engine Docker do Raspberry (Opcional)

```
$ sudo apt-get purge docker-ce
```

```
$ sudo apt-get purge docker-ce-cli
```

5. Deletar arquivos, imagens, contêineres

```
$ sudo rm -rf /var/lib/docker
```

Todas as informações, códigos gerados podem ser encontrados no repositório GitHub: `https://github.com/Henriquer88/Conteiner_Edge`

4 Análise

Contêineres e máquinas virtuais (*Virtual Machines*, VMs, em inglês) são tecnologias de virtualização de recursos muito semelhantes, onde os recursos do sistema podem ser "virtualizados" e representados como múltiplos recursos. É perfeitamente possível utilizar containers e máquinas virtuais em conjunto, embora na prática esse tipo de situação possa não ocorrer. Se você tiver requisitos de hardware específicos para o seu projeto ou estiver desenvolvendo em uma plataforma de hardware e precisar direcionar outra, você precisará usar uma máquina virtual.

A maioria dos outros requisitos "somente software" podem ser atendidos usando contêineres. A principal diferença entre contêineres e máquinas virtuais é que as máquinas virtuais virtualizam uma máquina inteira até as

camadas de hardware, e os contêineres virtualizam apenas as camadas de software acima do nível do sistema operacional.

Algumas vantagens e desvantagens de ambas as tecnologias são mostradas na tabela 1.

Ambas as tecnologias tendem a ser confundidas e muitas vezes são difíceis de diferenciar porque visam abstrair e dissociar o hardware subjacente do aplicativo. Portanto, existem diferenças, Figura 3, na forma em que atingem os objetivos estabelecidos.

Characteristic	Containers	VMs
Virtualization	OS Kernel	Hardware
OS	Shared	Dedicated
Size	Small	Large
Boot time	Low	High
Performance	Lighter	Heavy
Security	Access Control	Hypervisor
Communication	Ethernet	IPC
QoS	Better	Average

Figura 3: Comparação entre contêineres e máquinas virtuais. Fonte: [9].

A necessidade de contêineres surge do fato de que os desenvolvedores tiveram que enfrentar uma tarefa difícil de portar aplicativos do ambiente de desenvolvimento para ambientes de teste/laboratório e, finalmente, para ambientes de produção. Cada vez que um aplicativo é transferido para um novo ambiente, ele enfrenta problemas de migração porque normalmente não existem dois ambientes semelhantes em termos de recursos de software e hardware. Assim, para superar essa desvantagem e trazer mais eficiência operacional, entra em cena a tecnologia de contêineres [9]. No entanto, é provável que uma das duas opções satisfaça as suas necessidades se tiver em conta os recursos e compensações necessárias.

Tabela 1: Vantagens e desvantagens de contêineres e VMs

Recurso	Vantagens	Desvantagem
Contêiners	•Velocidade de iteração: como os contêineres são leves e incluem apenas software de alto nível, eles são muito rápidos para modificar e iterar. • Ecossistema estável: a maioria dos sistemas de tempo de execução de contêineres oferece um repositório público hospedado de contêineres pré-construídos com aplicativos de software que podem ser baixados e executados instantaneamente, economizando tempo das equipes de desenvolvimento.	•Vulnerabilidades de host compartilhado: todos os contêineres compartilham o mesmo sistema de hardware subjacente abaixo da camada do sistema operacional; é possível que uma vulnerabilidade em um contêiner saia do contêiner e afete o hardware compartilhado.
Máquinas Virtuais (VMs)	• Segurança de isolamento total: as máquinas virtuais funcionam isoladamente como um sistema completamente independente. Isso significa que as máquinas virtuais são imunes a qualquer vulnerabilidade ou interferência de outras máquinas virtuais em um host compartilhado.• Desenvolvimento interativo: As máquinas virtuais são dinâmicas e podem ser desenvolvidas de forma interativa. Uma vez especificada a definição básica de hardware para uma máquina virtual, a máquina virtual pode ser tratada como um computador básico, enquanto os contêineres são geralmente definições estáticas das dependências esperadas.	• Velocidade de iteração: o desenvolvimento e a reconstrução de máquinas virtuais consomem muito tempo, demorando tempo para regenerar quaisquer alterações em um instantâneo de máquina virtual. • Custo do tamanho do armazenamento: As máquinas virtuais podem ocupar muito espaço de armazenamento, levando a problemas de falta de espaço na máquina host das máquinas virtuais.

Referências

[1] Manoel Veras. *Virtualização*. Brasport, 2011.

[2] Ihor Zakutynskyi. Finding the optimal number of computing containers in iot systems: Application of mathematical modeling methods. *Electronics and Control Systems*, 2:9–14, 6 2023.

[3] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things: An overview. *The internet society (ISOC)*, 80:1–50, 2015.

[4] Sakina Elhadi, Abdelaziz Marzak, Nawal Sael, and Soukaina Merzouk. Comparative study of iot protocols. *Smart Application and Data Analysis for Smart Cities (SADASC'18)*, 2018.

[5] Apostolos Gerodimos, Leandros Maglaras, Mohamed Amine Ferrag, Nick Ayres, and Ioanna Kantzavelou. Iot: Communication protocols and security threats, 1 2023.

[6] Md Milon Islam, Sheikh Nooruddin, Fakhri Karray, and Ghulam Muhammad. Internet of things: Device capabilities, architectures, protocols, and smart applications in healthcare domain. *IEEE Internet of Things Journal*, 10:3611–3641, 2 2023.

[7] Quy Vu Khanh, Nam Vi Hoai, Linh Dao Manh, Anh Ngoc Le, and Gwanggil Jeon. Wireless communication technologies for iot in 5g: Vision, applications, and challenges. *Wireless Communications and Mobile Computing*, 2022, 2022.

[8] Zhiyu Wang, Mohammad Goudarzi, Jagannath Aryal, and Rajkumar Buyya. Container orchestration in edge and fog computing environments for real-time iot applications. 3 2022.

[9] Tamanna Siddiqui, Shadab Alam Siddiqui, and Najeeb Ahmad Khan. Comprehensive analysis of container technology. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 218–223, 2019.