**UNIVERSIDADE FEDERAL DE RORAIMA**

**CENTRO DE CIÊNCIA E TECNOLOGIA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**DCC403– SISTEMAS OPERACIONAIS I – 2025**

**PROF. DR. HEBERT OLIVEIRA ROCHA**

**ARTHUR CORREIA DE OLIVEIRA RAMOS**

**CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA**

**Virtualização de Sistemas Operacionais – VB e Containers**

**BOA VISTA, RR**

**2025**

**ARTHUR CORREIA DE OLIVEIRA RAMOS**

**CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA**

**Virtualização de Sistemas Operacionais – VB e Containers**

Trabalho da disciplina de Sistemas Operacionais I do ano de 2025.1 apresentado à Universidade Federal de Roraima do curso de Bacharelado em ciência da computação.

Docente: Prof. Dr. Hebert O. Rocha

**BOA VISTA, RR**

**2025**

LISTA DE ABREVIATURAS E SIGLAS

DSATUR *Degree of saturation*

SUMÁRIO

[1. Introdução 5](#_Toc444681825)

[2. Algoritmos 5](#_Toc444681826)

[2.1. Algoritmo Guloso 5](#_Toc444681827)

[2.1.1. Pseudocódigo 5](#_Toc444681828)

[2.2. Algoritmo Dsatur 6](#_Toc444681829)

[2.2.1. Pseudocódigo 6](#_Toc444681830)

[3. Avaliação Experimental 8](#_Toc444681831)

# Introdução

O presente relatório documenta o desenvolvimento de um projeto de virtualização e containerização que visa criar uma rede de computadores virtuais com comunicação entre Máquinas Virtuais e Containers. O projeto aborda desde a instalação das ferramentas até a implementação de sistemas escaláveis e análise comparativa das tecnologias.

# Processo de Instalação de Software para Criação de Máquinas Virtuais e Containers

# Instalação do VirtualBox

Sistema Ubuntu/Debian:

sudo apt update

sudo apt install virtualbox

Sistema Fedora:

sudo dnf install VirtualBox

# Instalação do Docker

Sistema Ubuntu**:**

bash

sudo apt update

sudo apt install docker.io

sudo systemctl start docker

sudo systemctl enable docker

Sistema Fedora:

bash

sudo dnf install docker

sudo systemctl start docker

sudo systemctl enable docker

**Verificação da Instalação:**

bash

docker --version

# Criando um Cluster com Máquinas Virtuais

# Crie 3 VMs no VirtualBox com Ubuntu Server ou Fedora Server.

# Configure o modo de rede como Rede Interna ou Adaptador de Rede Paravirtualizado (VirtIO).

# Configure IPs estáticos em cada VM, por exemplo:

VM1: 192.168.100.10

VM2: 192.168.100.11

VM3: 192.168.100.12

# Teste o ping entre elas.

ping 192.168.100.11

# **Criando um Cluster de Co**ntainers (D**ocker Swarm)**

Passos**:**

# Em um dos nós (gerenciador)

docker swarm init --advertise-addr 192.168.100.10

# Nos outros nós (workers), use o token exibido:

docker swarm join --token <TOKEN> 192.168.100.10:2377

# Verifique os nós no gerenciador:

docker node ls

# **Vantagens e desvantagens**

No contexto da computação moderna, especialmente em ambientes de desenvolvimento e infraestrutura em nuvem, duas tecnologias se destacam para o empacotamento e execução de aplicações: **containers** e **máquinas virtuais (VMs)**. Ambas permitem a criação de ambientes isolados, mas funcionam de formas diferentes e oferecem vantagens e desvantagens distintas.

# Desempenho

As máquinas virtuais são mais pesadas, pois cada instância executa um sistema operacional completo sobre um hipervisor. Esse modelo, apesar de oferecer isolamento total, exige mais recursos computacionais e impacta no desempenho. Os containers, por outro lado, são mais leves e rápidos. Isso acontece porque eles compartilham o kernel do sistema operacional do host, eliminando a necessidade de um SO completo para cada instância.

# Isolamento

As VMs oferecem um isolamento completo, já que cada uma roda seu próprio kernel. Isso proporciona maior segurança e independência entre instâncias. Já os containers possuem isolamento parcial, pois compartilham o mesmo kernel do host. Apesar disso, o nível de isolamento dos containers costuma ser suficiente para a maioria das aplicações, especialmente em ambientes controlados.

# Portabilidade

A portabilidade é um dos principais pontos fortes dos containers. Como uma imagem de container contém tudo o que a aplicação precisa (código, bibliotecas e configurações), ela pode ser executada em qualquer lugar onde o Docker (ou outro runtime de containers) esteja instalado. Máquinas virtuais, por dependerem de hipervisores e arquivos de imagem maiores, são menos portáteis e mais difíceis de mover entre ambientes diferentes.

# Tempo de Inicialização

Containers se destacam também na velocidade de inicialização. Como não precisam carregar um sistema operacional completo, eles iniciam em poucos segundos ou até milissegundos. Já as VMs podem levar vários segundos ou minutos para estarem prontas, o que pode ser uma desvantagem em ambientes que exigem escalabilidade rápida.

# Uso de Recursos

Em termos de uso de recursos, os containers são significativamente mais eficientes. Compartilhando recursos do host e ocupando menos espaço, eles permitem maior densidade de aplicações rodando em um mesmo servidor. Máquinas virtuais, por sua vez, consomem mais memória, processamento e armazenamento por conta do seu modelo completo de virtualização.

# **Sistema Web Escalável com Containers**

Estrutura:

NGINX (Proxy reverso)

Aplicação Web (Ex: Flask ou Node.js)

Banco de Dados (Ex: PostgreSQL)

Docker Compose: docker-compose.yml

docker compose atua como uma ferramenta que permite executar varias acoes tendo independente do SO da maquina.

version: '3'

services:

web:

image: node:18

volumes:

- ./app:/app

working\_dir: /app

command: node server.js

ports:

- "3000:3000"

db:

image: postgres:15

environment:

POSTGRES\_USER: user

POSTGRES\_PASSWORD: pass

POSTGRES\_DB: appdb

volumes:

- pgdata:/var/lib/postgresql/data

nginx:

image: nginx:alpine

ports:

- "80:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

volumes:

pgdata:

nginx.conf

events {}

http {

server {

listen 80;

location / {

proxy\_pass http://web:3000;

}

}

}

# **Gerenciamento de Containers**

# Sugestão: Portainer

# Instalação:

# docker volume create portainer\_data

# docker run -d -p 9000:9000 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data portainer/portainer-ce

# Acesse: http://localhost:9000

# **Sistemas Operacionais com foco em Virtualização**

# Proxmox VE

# Foco em virtualização KVM e containers LXC.

# Interface web integrada.

# Ubuntu Server com KVM

# Suporte a libvirt, KVM e virt-manager.

# Fedora Server com Cockpit

# Ferramenta web para administração, incluindo máquinas virtuais.

# **Conclusão**

# A implementação deste projeto permitiu uma análise prática comparativa entre virtualização e containerização. A configuração dos clusters evidenciou que máquinas virtuais demandam recursos substanciais devido ao overhead de sistemas operacionais completos, enquanto containers demonstraram eficiência superior com menor consumo de recursos e inicialização mais rápida.

# O desenvolvimento do sistema web escalável com Docker Compose revelou a potência dos containers para arquiteturas de microserviços, facilitando a separação em componentes distintos (NGINX, Node.js e PostgreSQL) e proporcionando escalabilidade eficiente. O Portainer ofereceu interface de gerenciamento centralizada e intuitiva para todo o ambiente containerizado.

# A análise dos sistemas operacionais especializados (Proxmox VE, Ubuntu Server com KVM e Fedora Server com Cockpit) demonstrou a maturidade das soluções disponíveis, cada uma adequada para contextos específicos de aplicação.

# Os resultados confirmaram que ambas as tecnologias são complementares. Máquinas virtuais permanecem essenciais para cenários que exigem isolamento completo e diferentes sistemas operacionais, enquanto containers são ideais para desenvolvimento ágil e arquiteturas cloud-native. Este projeto proporcionou base sólida para futuras implementações profissionais em modernização de aplicações e otimização de recursos computacionais.

**REFERÊNCIAS**