

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Infraestrutura Sob Demanda com OpenStack: Nuvem Privada de Máquinas Virtuais em Ambiente Acadêmico

Nathália Pimentel de Assis

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Marcelo Antonio Marotta

Brasília
2025



Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Infraestrutura Sob Demanda com OpenStack: Nuvem Privada de Máquinas Virtuais em Ambiente Acadêmico

Nathália Pimentel de Assis

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Marcelo Antonio Marotta (Orientador)
CIC/UnB

Prof. Dr. Sebastien R. M. J. Rondineau
Universidade de Brasília

Prof. Dr. Marcos F. Caetano
Universidade de Brasília

Prof. Dr. Guilherme Novaes Ramos
Coordenador do Curso de Engenharia da Computação

Brasília, 17 de julho de 2025

Agradecimentos

Finalizar esta monografia é um marco importante na minha trajetória, que só foi possível graças ao apoio e à companhia de pessoas especiais. Ao Prof. Dr. Marcelo Marotta, meu orientador, pela orientação atenta, paciência e dedicação em cada etapa. Agradeço ao Prof. Dr. Sebastien Rondineau, por me acolher no LabTelecom, confiar no meu trabalho e fornecer os recursos essenciais para este projeto. Ao Kauan, meu companheiro desde 2017, por me incentivar a continuar e acreditar em mim, mesmo nos momentos mais difíceis. À Camila, Gabriel, Gino, José, Júlia, Mário e Vivian, por cada conversa, apoio e momento que tornaram essa jornada mais leve. E, com todo o meu amor, às minhas cachorras, que são mais importantes para mim do que qualquer ser humano: Luna, Chanel, Pitchula, Pérola e Lilica. O carinho e a companhia de vocês foram meu sossego nos dias difíceis e minha alegria constante. Esta conquista também é de vocês.

Resumo

A Computação em Nuvem, especialmente no seu modelo de Nuvem Privada, representa uma solução estratégica para instituições de ensino superior que buscam otimizar o uso de recursos e prover ambientes de aprendizado prático para seus alunos. Contudo, a implantação de plataformas robustas como o OpenStack apresenta uma barreira técnica significativa devido à sua inerente complexidade, o que dificulta sua adoção em ambientes com recursos e equipes de TI limitados. Esta monografia aborda este desafio ao propor e validar um modelo prático para a implantação automatizada de uma infraestrutura de nuvem privada baseada em OpenStack, utilizando a ferramenta de automação Ansible com foco no reaproveitamento de hardware legado. Um protótipo funcional foi implementado em um *cluster* de cinco servidores revitalizados, onde o processo de instalação e configuração dos serviços OpenStack foi totalmente orquestrado por *playbooks* Ansible. Os resultados demonstram a eficácia do modelo, que realizou a implantação completa da infraestrutura em aproximadamente 45 minutos, e a validação funcional confirmou que a nuvem resultante estava plenamente operacional e apta a provisionar máquinas virtuais. Conclui-se que o modelo de automação proposto oferece uma solução viável, consistente e replicável que efetivamente reduz a barreira de entrada para a adoção de nuvens privadas em ambientes acadêmicos.

Palavras-chave: núvem privada, openstack, trabalho de conclusão de curso,computação em nuvem, automação de infraestrutura, ansible.

Abstract

Cloud Computing, particularly in its Private Cloud model, represents a strategic solution for higher education institutions seeking to optimize resource utilization and provide practical learning environments for students. However, the deployment of robust cloud platforms such as OpenStack presents a significant technical barrier due to its inherent complexity, which hinders its adoption in environments with limited IT resources and staff. This monograph addresses this challenge by proposing and validating a practical model for the automated deployment of a private OpenStack cloud infrastructure, leveraging the Ansible automation tool with a focus on repurposing legacy hardware. A functional prototype was implemented on a five-node cluster of revitalized servers, where the installation and configuration process of the OpenStack services was fully orchestrated by Ansible playbooks. The results demonstrate the model's effectiveness, achieving a full infrastructure deployment in approximately 45 minutes, while functional validation confirmed that the resulting cloud was fully operational and capable of provisioning virtual machines. It is concluded that the proposed automated deployment model offers a viable, consistent, and replicable solution that effectively lowers the barrier to entry for the adoption of private clouds in academic environments.

Keywords: cloud computing, private cloud, openstack, infrastructure automation, ansible

Sumário

Lista de Siglas e Abreviaturas **1**

1 Introdução **2**
1.1 Contextualização 2
1.1.1 O Paradigma da Computação em Nuvem 2
1.1.2 O Cenário Acadêmico e a Motivação do Projeto 3
1.2 O Problema de Pesquisa: A Barreira da Complexidade do OpenStack 4
1.2.1 A Causa Raiz: A Arquitetura Distribuída 4
1.2.2 O Desafio Prático: A Implantação Manual 4
1.2.3 A Consequência: Alta Barreira de Entrada 5
1.2.4 A Lacuna na Literatura 5
1.3 Proposta da Solução e Contribuições 5
1.4 Estrutura do Documento 6

2 Trabalhos Relacionados **7**
2.1 Fundamentação Teórica 7
2.1.1 Computação em Nuvem: Modelos e Características 7
2.1.2 Virtualização: A tecnologia habilitadora 8
2.1.3 OpenStack: A Plataforma de Gerenciamento de Nuvem 9
2.1.4 Automação de Infraestrutura como Código (IaC) 10
2.2 Análise dos Trabalhos Relacionados 11
2.2.1 Síntese da Lacuna na Literatura e Justificativa da Proposta 19

3 Proposta **21**
3.0.1 Arquitetura do Modelo de Implantação 23
3.0.2 Detalhamento do Processo de Automação com Ansible 24
3.0.3 Modelo de Configuração e Inventário 26

4 Prototipo **29**
4.0.1 Ambiente de Hardware e Software Utilizado 29

4.0.2 Metodologia de Implantação do Protótipo 30
4.0.3 Execução e Validação do Protótipo 33

5 Resultados **37**
5.0.1 Resultados da Automação da Implantação 37
5.0.2 Validação Funcional da Infraestrutura 38
5.0.3 Estudo de Caso: Provisionamento de Ambiente para Simulação 40
5.0.4 Discussão dos Resultados e Desafios 41

6 Conclusão **43**
6.1 Conclusões 43
6.2 Trabalhos Futuros 44

Referências **46**

Lista de Figuras

2.1 Modelos de Serviço e Implantação da Nuvem (Imagem adaptada de [1]). . . 8
2.2 Comparativo Arquitetural entre Bare Metal, Máquina Virtual e Containerização (Imagem adaptada de [2]). 9
2.3 Arquitetura Conceitual do OpenStack (Imagem retirada de [3]). 10

3.1 Diagrama da Arquitetura do Modelo de Implantação Proposto. Fonte: Elaborado pelo autor (2025). 28

4.1 Saída dos comandos de verificação de serviços, confirmando o estado operacional da nuvem. 34
4.2 Saída dos comandos de verificação dos agentes. 34
4.3 Saída do comando `openstack server show` para a instância de teste `test-cirros`. 36

5.1 Painel de Hipervisores do OpenStack Horizon, mostrando os cinco nós do cluster ativos e gerenciados pelo serviço Nova. Fonte: Elaborado pelo autor (2025). 39
5.2 Painel de Visão Geral do Projeto no Horizon, evidenciando o consumo de recursos após o provisionamento de uma instância. Fonte: Elaborado pelo autor (2025). 40

Lista de Tabelas

3.1 Levantamento dos equipamentos em desuso no Laboratório 23

4.1 Especificações Técnicas dos Servidores Utilizados na Nuvem Privada 30

Lista de Siglas e Abreviaturas

API Application Programming Interface (Interface de Programação de Aplicações)

CLI Command-Line Interface (Interface de Linha de Comando)

DHCP Dynamic Host Configuration Protocol

HA High Availability (Alta Disponibilidade)

HDD Hard Disk Drive (Unidade de Disco Rígido)

IaaS Infrastructure as a Service (Infraestrutura como Serviço)

IP Internet Protocol (Protocolo de Internet)

KVM Kernel-based Virtual Machine (Máquina Virtual Baseada em Kernel)

NIST National Institute of Standards and Technology

PaaS Platform as a Service (Plataforma como Serviço)

PERC PowerEdge RAID Controller

RAID Redundant Array of Independent Disks

SaaS Software as a Service (Software como Serviço)

SSH Secure Shell

VCL Virtual Computer Lab (Laboratório Virtual de Computação)

VM Virtual Machine (Máquina Virtual)

VMM Virtual Machine Monitor (Monitor de Máquina Virtual)

VPN Virtual Private Network (Rede Privada Virtual)

YAML YAML Ain't Markup Language

Capítulo 1

Introdução

A crescente demanda por recursos computacionais flexíveis e escaláveis, impulsionada pela transformação digital, consolidou a Computação em Nuvem como um paradigma fundamental na Tecnologia da Informação. Este capítulo introduz os conceitos essenciais que fundamentam este trabalho, partindo do contexto geral da Computação em Nuvem até a apresentação do problema específico e dos objetivos que esta monografia busca solucionar.

1.1 Contextualização

1.1.1 O Paradigma da Computação em Nuvem

A Computação em Nuvem é formalmente definida pelo National Institute of Standards and Technology (NIST) como um modelo que permite o acesso ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (ex: redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento [1]. Este modelo se sustenta em cinco características essenciais: autoatendimento sob demanda, amplo acesso à rede, pool de recursos, rápida elasticidade e serviço mensurado [1].

Os serviços de nuvem são entregues em três modelos principais: **Infraestrutura como Serviço (IaaS)**, que oferece os recursos mais básicos de computação, armazenamento e rede; **Plataforma como Serviço (PaaS)**; e **Software como Serviço (SaaS)** [1]. Os modelos de implantação, por sua vez, determinam como a infraestrutura é provisionada, sendo a **Nuvem Privada** o modelo onde os recursos são de uso exclusivo de uma única organização, garantindo maior controle e segurança [1]. O foco deste trabalho reside na implementação de uma nuvem privada que oferece IaaS.

A tecnologia habilitadora por trás da nuvem é a **virtualização**, a técnica que permite particionar um único sistema computacional físico em múltiplos ambientes de execução isolados, denominados Máquinas Virtuais (VMs) [4]. Essa virtualização é gerenciada por uma camada de software chamada *hypervisor* ou Monitor de Máquina Virtual (VMM), que se posiciona entre o hardware e o sistema operacional da VM [4].

Para orquestrar esses recursos virtualizados em larga escala, é necessária uma camada de gerenciamento de nuvem. Neste contexto, o **OpenStack** emergiu como a plataforma de código aberto mais proeminente para a criação de nuvens IaaS, sendo frequentemente descrito como um "sistema operacional de nuvem"[5, 6] capaz de controlar grandes pools de recursos de computação, armazenamento e rede [6].

1.1.2 O Cenário Acadêmico e a Motivação do Projeto

A aplicação da Computação em Nuvem no setor educacional tem sido motivada por desafios práticos e pela busca por ambientes de aprendizado mais flexíveis e eficientes. Instituições de ensino superior frequentemente enfrentam dificuldades com laboratórios de informática tradicionais, que implicam em altos custos de aquisição e manutenção de hardware especializado, além de limitações logísticas de espaço físico [7, 8]. Nesse cenário, a implementação de Laboratórios Virtuais (VCLs) através de uma nuvem privada surge como uma solução estratégica, permitindo oferecer aos estudantes ambientes de aprendizado acessíveis remotamente e disponíveis sob demanda, o que enriquece significativamente a experiência prática [9].

Adicionalmente, o ciclo de atualização tecnológica em universidades frequentemente leva à existência de um parque de máquinas que, embora mais antigo, ainda possui capacidade computacional relevante, mas que acaba em desuso. A implementação de uma nuvem privada, como demonstrado por estudos de caso em outras instituições federais, é uma abordagem sustentável e financeiramente inteligente para revitalizar esses equipamentos [10]. Ao agrupar os recursos de servidores heterogêneos, é possível criar uma infraestrutura robusta sem a necessidade de grandes investimentos em novo hardware.

Este foi precisamente o cenário que motivou o presente trabalho, observado no Laboratório de Telecomunicações (LabTelecom) da Universidade de Brasília. O laboratório possuía um conjunto de servidores da família Dell PowerEdge que estavam parados e subutilizados. Ao mesmo tempo, havia uma demanda recorrente por parte de alunos e pesquisadores por acesso a máquinas de alto desempenho, capazes de rodar simulações de engenharia pesadas em softwares como Ansys, Keysight e Altium Designer. Essas aplicações exigem não apenas poder de processamento, mas também ambientes de sistema operacional completos e isolados, especialmente Windows.

Diante deste contexto, a abordagem de prover Máquinas Virtuais (VMs) através de uma nuvem IaaS se mostrou a mais adequada. As VMs oferecem o nível de isolamento necessário, permitindo que cada aluno tenha um ambiente completo com privilégios administrativos para instalar e configurar softwares complexos, sem interferir nos outros usuários ou na infraestrutura subjacente [7]. Embora uma nuvem privada baseada em OpenStack que ofereça VMs seja a solução ideal para o problema apresentado, o caminho para sua implementação é repleto de desafios técnicos, que constituem o problema de pesquisa desta monografia.

1.2 O Problema de Pesquisa: A Barreira da Complexidade do OpenStack

Apesar de uma nuvem privada baseada em OpenStack ser a solução ideal para o cenário do LabTelecom, o caminho para sua implementação é repleto de desafios. A literatura e a prática demonstram de forma consistente que a alta complexidade técnica da plataforma representa a principal barreira para sua adoção, especialmente em ambientes acadêmicos com recursos limitados.

1.2.1 A Causa Raiz: A Arquitetura Distribuída

A principal fonte da complexidade do OpenStack reside em sua própria arquitetura. A plataforma não é um software monolítico, mas sim um conjunto de múltiplos serviços distribuídos e inter-relacionados — como Nova (computação), Neutron (rede), Keystone (identidade) e Glance (imagens) — que precisam ser instalados e configurados individualmente para interagirem de forma coesa [6]. Essa natureza modular, embora poderosa para garantir a escalabilidade, aumenta drasticamente a dificuldade da instalação inicial, pois cada componente possui seu próprio banco de dados, arquivos de configuração e dependências [11, 12].

1.2.2 O Desafio Prático: A Implantação Manual

Essa complexidade arquitetural se traduz em um desafio prático significativo durante a implantação manual. Guias técnicos, como o detalhado por Kumar et al. [11], expõem a longa e minuciosa sequência de passos que um administrador precisa executar. O processo envolve a execução de dezenas de comandos de terminal, a edição manual de múltiplos arquivos de configuração em diferentes diretórios, a criação de bancos de dados e a inicialização de serviços em uma ordem estrita de dependência [11]. Este processo, além

de demorado, é altamente suscetível a erros humanos de digitação ou de sequência, que podem comprometer toda a implantação.

1.2.3 A Consequência: Alta Barreira de Entrada

A alta barreira de entrada imposta pela complexidade da instalação é tão significativa que, em muitos casos, até mesmo pesquisadores buscam alternativas simplificadas para viabilizar seus experimentos. No estudo de caso de Kabiri e Wannous [9], por exemplo, os autores optaram por utilizar o `DevStack` para implantar o ambiente de laboratório virtual. O `DevStack` é uma ferramenta projetada para criar rapidamente ambientes de desenvolvimento, mas que não é recomendada ou adequada para um ambiente de produção devido à sua simplicidade e falta de robustez [9]. A escolha por essa ferramenta de teste evidencia a percepção de que uma instalação de produção manual seria um esforço proibitivo para o escopo daquele trabalho, ilustrando perfeitamente a barreira que impede que os benefícios da nuvem sejam alcançados.

1.2.4 A Lacuna na Literatura

Portanto, a análise da literatura revela um claro paradoxo: a plataforma mais poderosa e flexível para a criação de nuvens privadas é também uma das mais complexas de se implantar. A lacuna, portanto, não está na ausência de casos de uso ou na validação do valor do OpenStack, mas sim na falta de um modelo prático e acessível que resolva o desafio fundamental do **"Dia 1"**: como construir a nuvem de forma eficiente, repetível e confiável. É precisamente para preencher esta lacuna que a presente monografia se insere.

1.3 Proposta da Solução e Contribuições

Para endereçar a significativa barreira técnica imposta pela complexidade de implantação do OpenStack, esta monografia propõe e valida um modelo de engenharia para a criação de nuvens privadas em ambientes acadêmicos com recursos limitados. A solução transforma o processo de instalação, tradicionalmente manual e propenso a erros, em um fluxo de trabalho padronizado, automatizado e confiável. A proposta está fundamentada em uma abordagem prática que vai desde a preparação da infraestrutura física até a orquestração do software, resolvendo o problema da complexidade e garantindo um resultado consistente e replicável.

A base do modelo consiste na revitalização de equipamentos em desuso, abrangendo desde a preparação física dos servidores, com a configuração de arranjos de disco (RAID), até a alocação estratégica de papéis — Nó de Controle e Nós de Computação — para

otimizar o uso do conjunto heterogêneo de hardware disponível. Sobre esta base, uma arquitetura de rede estável e com endereçamento IP estático é implementada para garantir a comunicação confiável entre todos os componentes. O pilar final, que efetivamente constrói a nuvem, é a automação completa do processo de instalação utilizando a ferramenta Ansible, que orquestra todo o complexo fluxo de configuração dos serviços OpenStack por meio de *playbooks*.

Ao aplicar este modelo, o presente trabalho oferece as seguintes contribuições principais para a área:

- A investigação sistemática e a documentação dos desafios e da complexidade inerentes à implantação manual de uma nuvem OpenStack em um ambiente com múltiplos nós.
- O desenvolvimento de um conjunto de *scripts* de automação (playbooks Ansible) que orquestra a instalação e a configuração dos principais serviços do OpenStack (Keystone, Glance, Nova, Neutron e Horizon), servindo como um modelo prático e customizável.
- A validação do modelo proposto através da implantação de um protótipo de nuvem totalmente funcional sobre um cluster de cinco servidores legados, demonstrando a drástica redução no tempo e no esforço manual em comparação com os processos tradicionais.
- A apresentação de um modelo documentado e replicável que pode ser adaptado por outras instituições de ensino, efetivamente reduzindo a barreira de entrada para a adoção de soluções de nuvem privada.

1.4 Estrutura do Documento

O restante desta monografia está organizado da seguinte forma. O Capítulo 2 apresenta a Fundamentação Teórica e os Trabalhos Relacionados, detalhando os conceitos tecnológicos e analisando a literatura relevante para contextualizar o problema. O Capítulo 3 detalha a Proposta da Solução, descrevendo o modelo conceitual da automação com Ansible. O Capítulo 4 descreve a Implementação do Protótipo, abordando os passos práticos da revitalização do hardware e da execução da automação. O Capítulo 5 apresenta e analisa os Resultados obtidos com a implementação, validando a eficácia do protótipo. Por fim, o Capítulo 6 apresenta as Conclusões do trabalho e delineia as direções para Trabalhos Futuros.

Capítulo 2

Trabalhos Relacionados

Para uma compreensão aprofundada da solução proposta, este capítulo estabelece o embasamento teórico e analisa o estado da arte da literatura. A primeira seção, *Fundamentação Teórica*, será dedicada a apresentar os conceitos tecnológicos essenciais — desde os princípios de Computação em Nuvem e virtualização até a arquitetura da plataforma OpenStack e o paradigma de automação. A segunda seção, *Análise dos Trabalhos Relacionados*, irá sintetizar e comparar os estudos relevantes de forma temática, agrupando-os por foco para construir um argumento coeso. Ao final, esta análise consolidada visa reforçar de maneira clara e estruturada a lacuna na literatura que esta monografia se propõe a endereçar.

2.1 Fundamentação Teórica

Para uma compreensão aprofundada dos desafios e da solução proposta nesta monografia, esta seção apresenta os conceitos tecnológicos fundamentais que servem de alicerce para o trabalho. Serão abordados os princípios de Computação em Nuvem conforme definidos pelo NIST, as tecnologias de virtualização que a viabilizam, a arquitetura da plataforma de gerenciamento OpenStack e, por fim, o paradigma de automação de infraestrutura.

2.1.1 Computação em Nuvem: Modelos e Características

A Computação em Nuvem é formalmente definida pelo NIST como um modelo para o acesso sob demanda a um conjunto compartilhado de recursos computacionais configuráveis [1]. Este modelo se sustenta em cinco características essenciais: **autoatendimento sob demanda**, **amplo acesso à rede**, **pool de recursos**, **rápida elasticidade** e **serviço mensurado** [1]. Os serviços são entregues em três modelos principais: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). O foco deste trabalho reside no modelo IaaS, que oferece os recursos mais básicos

(computação, armazenamento e rede) sobre os quais o consumidor pode implantar sistemas operacionais e aplicações [1, 6]. O modelo de implantação abordado é o de Nuvem Privada, onde a infraestrutura é de uso exclusivo de uma única organização, garantindo maior controle e segurança [1].

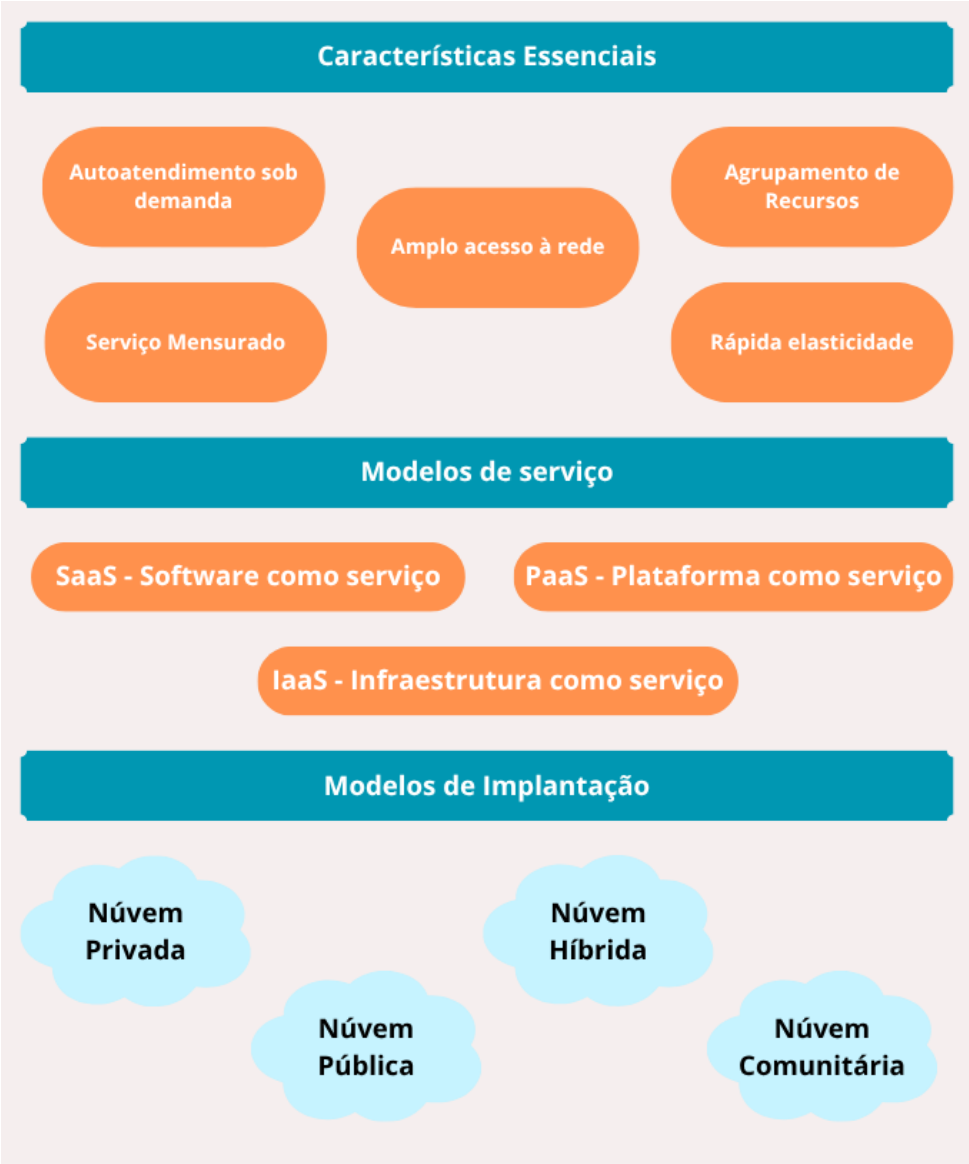


Figura 2.1: Modelos de Serviço e Implantação da Nuvem (Imagem adaptada de [1]).

2.1.2 Virtualização: A tecnologia habilitadora

A tecnologia que torna a computação em nuvem viável é a virtualização. A virtualização é a técnica que permite particionar um único sistema computacional físico em múltiplos ambientes de execução isolados, denominados Máquinas Virtuais (VMs) [4]. Cada VM se comporta como um computador completo e independente, com seu próprio sistema

operacional e aplicações [4]. Essa virtualização é gerenciada por uma camada de software chamada *hypervisor* ou Monitor de Máquina Virtual (VMM). Existem duas abordagens principais para hypervisors: o Tipo 1, que roda diretamente sobre o hardware (bare-metal), e o Tipo 2, que roda como uma aplicação sobre um sistema operacional hospedeiro [4]. Para ambientes de produção, o Hipervisor Tipo 1, como o KVM, é a escolha preferencial por seu desempenho e eficiência superiores.

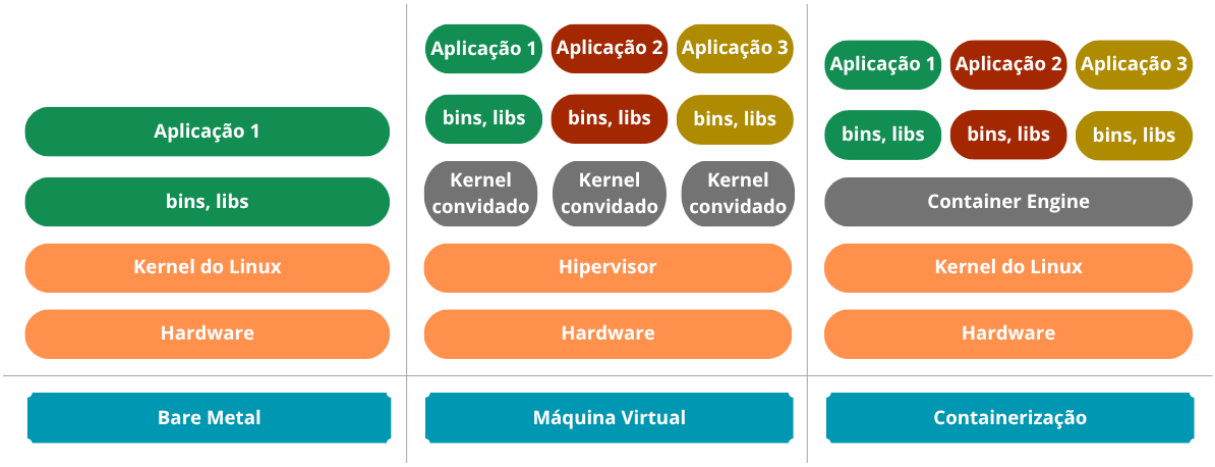


Figura 2.2: Comparativo Arquitetural entre Bare Metal, Máquina Virtual e Containerização (Imagem adaptada de [2]).

2.1.3 OpenStack: A Plataforma de Gerenciamento de Nuvem

É crucial distinguir entre a tecnologia de virtualização e a plataforma de gerenciamento. O OpenStack não é um hipervisor, mas sim um **sistema operacional de nuvem** de código aberto, projetado para controlar e orquestrar grandes pools de recursos de computação, armazenamento e rede [5, 6]. Sua arquitetura, ilustrada na Figura 2.3, é inerentemente modular e composta por um conjunto de serviços (projetos) que se comunicam via APIs para entregar uma solução IaaS completa e robusta.

A Figura 2.3 demonstra a complexidade e a riqueza do ecossistema OpenStack. Os componentes centrais, utilizados nesta monografia, incluem o **Nova** (Compute), que gerencia o ciclo de vida das máquinas virtuais; o **Neutron** (Network), que provê o serviço de rede virtual; o **Glance** (Image), que atua como um registro para as imagens de VMs; e o **Keystone** (Identity), que centraliza a autenticação e autorização para todos os outros serviços. Além desses, a arquitetura completa pode incluir componentes para orquestração de infraestrutura (**Heat**), armazenamento de objetos (**Swift**), armazenamento de blocos (**Cinder**) e processamento de Big Data (**Sahara**), entre outros. A necessidade de configurar e garantir a comunicação correta entre todos esses serviços é a principal fonte da complexidade da implantação do OpenStack.

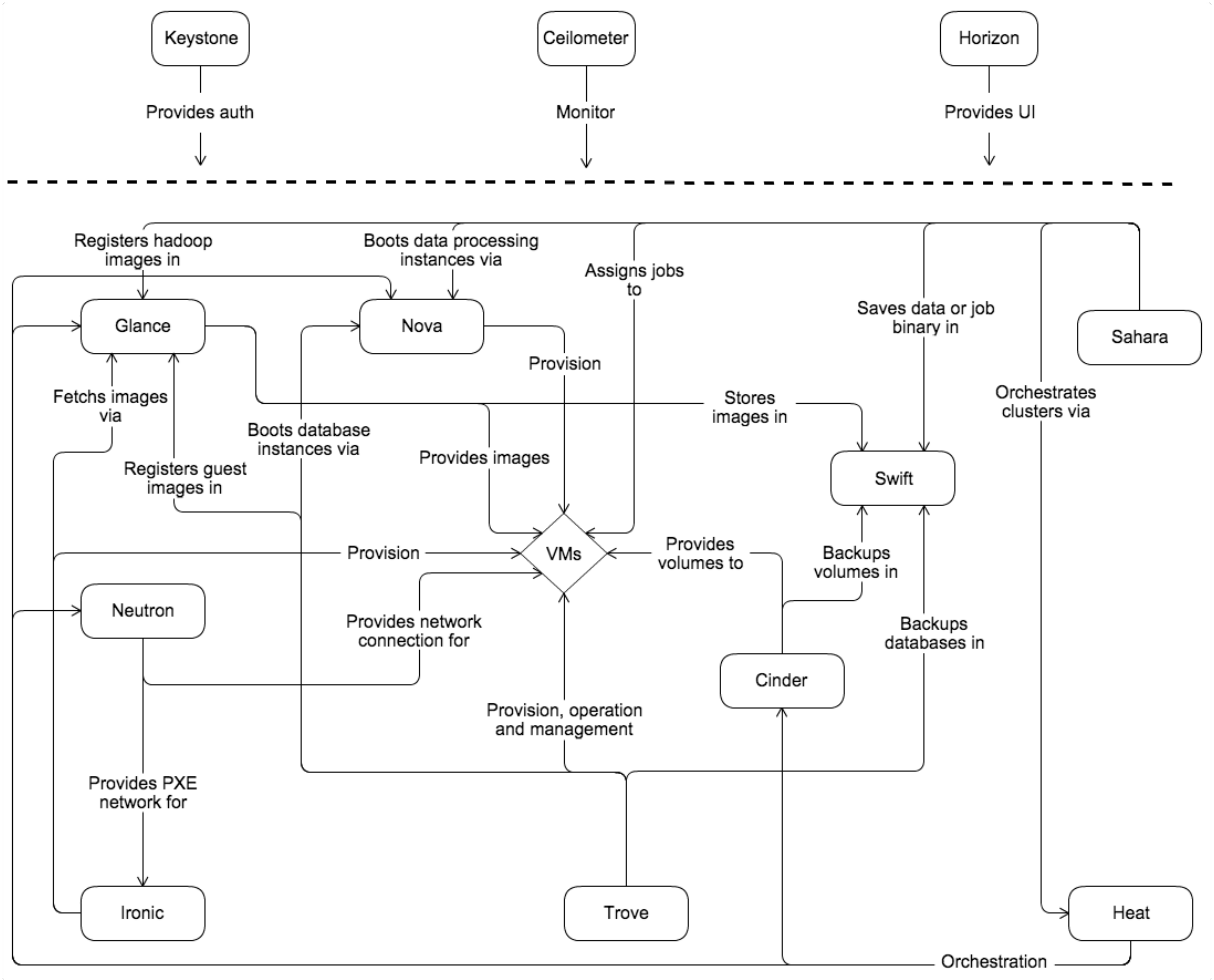


Figura 2.3: Arquitetura Conceitual do OpenStack (Imagem retirada de [3]).

2.1.4 Automação de Infraestrutura como Código (IaC)

A complexidade da implantação de plataformas como o OpenStack levou ao desenvolvimento do paradigma de Infraestrutura como Código (IaC). IaC é a prática de gerenciar e provisionar a infraestrutura de TI através de arquivos de definição legíveis por máquina (código), em vez de configuração manual. Ferramentas de automação como o Ansible e o Chef são os principais exemplos desta abordagem [13]. O **Ansible**, ferramenta escolhida nesta monografia, destaca-se por sua arquitetura *agentless* (sem agentes), comunicando-se com os nós gerenciados via SSH. A automação é definida em *playbooks*, escritos na linguagem de marcação YAML, que descrevem o estado final desejado para o sistema. Essa abordagem declarativa simplifica o gerenciamento e garante a invariabilidade, ou seja, a aplicação repetida do mesmo playbook resulta sempre no mesmo estado do sistema, sem gerar erros.

2.2 Análise dos Trabalhos Relacionados

Com os conceitos fundamentais estabelecidos na seção anterior, esta seção se dedica a analisar, de forma sintética e comparativa, os trabalhos relevantes que contextualizam esta monografia. Em vez de apresentar os estudos individualmente, a análise será agrupada em categorias temáticas que, em conjunto, constroem a justificativa para a presente proposta. A abordagem visa mapear as principais correntes de pesquisa na área, desde a validação do valor de nuvens privadas na academia até os desafios técnicos de sua implantação.

Para justificar a escolha da plataforma utilizada nesta monografia, é fundamental analisar estudos que comparam as soluções de código aberto disponíveis, uma vez que o custo de alternativas comerciais é frequentemente um impeditivo para ambientes acadêmicos e de pesquisa [12]. O trabalho de Wen et al. [12] cumpre este papel ao realizar um estudo comparativo detalhado entre as duas principais plataformas de gerenciamento de nuvem IaaS da época: OpenStack e OpenNebula. O objetivo dos autores é apresentar as funcionalidades de cada uma e compará-las sob diversas perspectivas, a fim de orientar a decisão de qual plataforma é mais adequada para diferentes necessidades de implantação [12].

A comparação realizada por Wen et al. [12] abrange múltiplos critérios, incluindo a origem e o apoio da comunidade, a arquitetura de cada solução, o suporte a diferentes hipervisores e as abordagens de segurança. O estudo destaca a arquitetura de serviços do OpenStack (composta pelos projetos Nova, Swift e Glance) e seu rápido crescimento impulsionado por um forte ecossistema de empresas [12]. Em contraste, o OpenNebula é apresentado como uma ferramenta com mais tempo de maturidade e uma arquitetura flexível, com forte presença em institutos de pesquisa e universidades. Com base nessa análise, os autores recomendam o OpenStack como uma escolha mais adequada para empresas que buscam encapsular serviços, enquanto o OpenNebula seria ideal para ambientes de pesquisa que valorizam a flexibilidade para integrar infraestruturas existentes [12].

A contribuição deste estudo é valiosa, pois fornece um guia claro para a tomada de decisão estratégica sobre qual ecossistema de código aberto adotar como base para uma nuvem privada. A análise de Wen et al. [12] ajuda a compreender as filosofias de projeto e os pontos fortes de cada plataforma. Contudo, as recomendações oferecidas são de alto nível, baseadas em perfis de organizações (empresa vs. pesquisa) e não se aprofundam nos desafios operacionais práticos do gerenciamento de uma nuvem. O trabalho não aborda, por exemplo, a complexidade da implantação da plataforma, um pré-requisito fundamental para qualquer caso de uso.

Portanto, um estudo comparativo como o de Wen et al. [12] é essencial para a fase de planejamento e justifica a escolha do OpenStack como plataforma. Uma vez que a plataforma é escolhida, o passo seguinte é analisar a complexidade de sua implementação

manual, um desafio detalhado em guias práticos como o de Kumar et al. [11], que será discutido na sequência.

O trabalho de Kumar et al. [11] serve como um excelente exemplo prático da complexidade de uma implantação manual. O artigo, que também realiza uma breve comparação entre plataformas, dedica uma porção significativa do seu conteúdo para apresentar um guia detalhado, um tutorial passo a passo, para a instalação do OpenStack (versão Icehouse) em um sistema RHEL. O objetivo é fornecer aos administradores iniciantes um roteiro com os comandos exatos necessários para configurar um ambiente de nuvem funcional, desde a instalação dos pacotes até a criação de uma máquina virtual com conectividade de rede [11].

A metodologia apresentada é inteiramente baseada na execução sequencial de comandos na linha de comando (CLI). Os autores detalham cada passo, desde a configuração inicial do sistema operacional até a criação de redes, sub-redes e roteadores virtuais utilizando o cliente do ‘quantum‘ (o antigo nome do Neutron) [11]. O guia ilustra a necessidade de executar dezenas de comandos distintos, editar múltiplos arquivos de configuração e gerenciar os vários serviços do OpenStack para que eles se comuniquem corretamente. Este processo manual reflete a natureza distribuída e modular da plataforma, onde cada peça deve ser cuidadosamente configurada.

A contribuição prática deste tipo de guia é seu valor instrutivo para um novo administrador, que pode seguir o roteiro para aprender a lógica fundamental por trás da criação de uma nuvem. No entanto, é precisamente essa natureza manual que revela a principal lacuna e o problema que esta monografia aborda. A abordagem apresentada por Kumar et al. [11], embora funcional para um único ambiente de teste, não é escalável nem eficiente para um cenário de produção multiusuário, como o de uma universidade. O processo manual é demorado, altamente suscetível a erros de digitação e inconsistências, tornando-se um gargalo operacional intransponível.

Portanto, a complexidade do processo manual, tão bem demonstrada por Kumar et al. [11], é o problema central que justifica a busca por métodos mais eficientes. A literatura, ciente deste desafio, evoluiu para propor soluções baseadas em automação. O "OpenStack Cloud Computing Cookbook"[13], por exemplo, aborda este mesmo problema, mas utilizando a ferramenta Chef para automatizar a implantação, como será discutido a seguir.

A complexidade do processo manual, tão bem demonstrada por guias como os de Kumar et al. [11], é um problema tão conhecido que a literatura técnica evoluiu para propor soluções baseadas em automação. O livro "OpenStack Cloud Computing Cookbook"de Jackson e Bunch [13], por exemplo, dedica um capítulo inteiro, o décimo, para o tema de *"Automating OpenStack Installations"*. O objetivo dos autores é apresentar um método

para automatizar a implantação de uma nuvem OpenStack utilizando a ferramenta de gerenciamento de configuração Chef [13].

A abordagem proposta no livro consiste em utilizar a infraestrutura do Chef, que é baseada em um modelo cliente-servidor, para definir o estado desejado de cada componente do OpenStack de forma programática. Para isso, são utilizados *cookbooks*, que são conjuntos de *recipes* (receitas) que detalham as tarefas a serem executadas, como instalar um pacote ou configurar um serviço [13]. Esses *recipes* são então agrupados em *roles* (papéis), como *controller* ou *compute*, que são atribuídos aos nós da nuvem. O servidor Chef, portanto, orquestra a execução remota desses passos nos nós clientes, garantindo uma instalação consistente e padronizada.

A grande contribuição de trabalhos como este é validar que a automação da implantação é, por si só, um tópico de pesquisa e engenharia relevante e complexo. Ao detalhar uma solução com Chef, os autores demonstram um caminho para superar a barreira da instalação manual. Contudo, a escolha pelo Chef implica a adoção de sua arquitetura cliente-servidor, que exige a instalação de um agente em cada nó, e uma curva de aprendizado para dominar sua sintaxe baseada em Ruby. Em contraste, a presente monografia opta pela ferramenta Ansible, que se justifica por sua simplicidade, sintaxe baseada em YAML e, crucialmente, por sua arquitetura sem agentes (*agentless*), que se comunica via SSH. Essas características tornam o Ansible uma escolha mais leve e, frequentemente, mais adequada para equipes de TI enxutas, como as de ambientes acadêmicos.

Portanto, o trabalho de Jackson e Bunch [13] é fundamental por estabelecer a automação da implantação como uma solução consolidada para a complexidade operacional do OpenStack. Contudo, a complexidade da implantação não se resume apenas a desafios operacionais; ela carrega consigo riscos significativos de segurança. Uma instalação manual, propensa a erros, pode facilmente resultar em configurações inseguras, e a literatura também se dedica a investigar essas vulnerabilidades. A revisão crítica realizada por Albaroodi et al. [14], que será discutida a seguir, aprofunda-se nessas fraquezas, reforçando a necessidade de um processo de implantação que seja não apenas eficiente, mas também padronizado e seguro.

Além da complexidade operacional, a segurança é outro desafio crítico na implantação do OpenStack, como detalhado na revisão de Albaroodi et al. [14]. O trabalho apresenta uma análise aprofundada das fraquezas e dos problemas de segurança inerentes à plataforma, destacando que, embora o OpenStack ofereça múltiplos mecanismos de proteção, uma configuração incorreta pode expor toda a infraestrutura a riscos significativos. O objetivo do artigo é justamente examinar o estado da segurança no OpenStack e os desafios que as organizações enfrentam [14].

A análise dos autores cobre diversas camadas, desde a segurança no nível da API, que

em versões mais antigas poderia trafegar credenciais em texto plano, até a importância do serviço de identidade Keystone. O estudo de Albaroodi et al. [14] ressalta que o Keystone, apesar de ser o pilar da segurança, depende de uma configuração rigorosa de papéis (*roles*) e políticas para ser eficaz. Falhas na configuração podem levar à escalada de privilégios, onde o administrador de um único projeto poderia, por exemplo, obter controle sobre todo o sistema. A pesquisa também aponta para a falta de requisitos de complexidade de senhas e para o armazenamento inseguro de credenciais em versões passadas da plataforma.

A principal contribuição deste trabalho é a consolidação das principais vulnerabilidades e pontos de atenção em segurança para administradores de nuvens OpenStack. Ao fazer isso, o artigo implicitamente reforça a necessidade de um processo de implantação padronizado e seguro. A lacuna identificada não é uma nova tecnologia, mas sim a dificuldade de se aplicar consistentemente as melhores práticas de segurança em um ambiente tão complexo. Um processo de implantação manual, como o descrito por Kumar et al. [11], é altamente propenso a erros que podem resultar em brechas de segurança.

Portanto, a análise de segurança de Albaroodi et al. [14] adiciona uma dimensão crítica à justificativa para a automação: ela não só simplifica a implantação, mas é fundamental para garantir a segurança e a consistência da configuração. Com a complexidade e os riscos da implantação estabelecidos, e com a automação posicionada como a solução, é possível focar nos resultados práticos que essa infraestrutura segura e bem construída permite alcançar. O estudo de caso de Pereira et al. [10], que parte de uma prova de conceito em pequena escala, representa precisamente esta etapa e será discutido a seguir.

O trabalho de Pereira et al. [10] apresenta um estudo de caso prático sobre a implantação de uma infraestrutura de nuvem utilizando OpenStack no campus do CEFET/RJ em Nova Iguaçu. O objetivo principal do estudo foi prover um ambiente de nuvem que pudesse ser utilizado por projetos desenvolvidos no campus, otimizando o uso dos recursos físicos já disponíveis e evitando a ociosidade computacional [10]. Para gerenciar o acesso a esses recursos, os autores também desenvolveram um portal web externo, que trata as requisições dos usuários de forma semi-automática [10]. Esta abordagem representa um passo inicial comum em ambientes acadêmicos para validar a tecnologia.

A metodologia adotada consistiu na instalação de todos os serviços do OpenStack em um único servidor físico para a realização de testes de desempenho [10]. Nessa infraestrutura contida, os autores avaliaram os tempos de criação e exclusão de máquinas virtuais, o desempenho da comunicação de rede através de testes de download de arquivos sob contenção e a performance da CPU em cenários de estresse e com provisionamento excedente de núcleos virtuais [10]. Os resultados forneceram uma base de dados inicial sobre o comportamento da plataforma em um ambiente de pequena escala, mostrando,

por exemplo, que a taxa de transferência da rede diminui consideravelmente com a adição de tráfego concorrente [10].

A contribuição do estudo de Pereira et al. [10] é a demonstração da viabilidade e a análise de desempenho de uma implementação simples de OpenStack, servindo como uma prova de conceito valiosa. No entanto, a natureza da implementação em um único *host* é sua principal limitação, pois não aborda as complexidades inerentes a um ambiente de produção distribuído e escalável, como a comunicação entre múltiplos nós de computação ou a gestão de um *cluster* de armazenamento. A avaliação de rede, embora útil, se restringe a um teste de vazão e não explora os desafios de configuração de redes isoladas para múltiplos projetos, roteamento entre redes virtuais ou políticas de segurança, que são aspectos críticos em uma nuvem maior.

Portanto, o trabalho de Pereira et al. [10] funciona como um excelente exemplo de uma prova de conceito fundamental que valida a tecnologia em um ambiente mínimo. Uma vez que a viabilidade da plataforma é confirmada, mesmo em pequena escala, o próximo passo lógico na literatura é evoluir para uma aplicação funcional que entregue valor direto aos usuários. A criação de um Laboratório Virtual (VCL) completo para estudantes, que é exatamente a proposta apresentada por Arora et al. [7], representa esta etapa e será discutida a seguir.

O estudo de Arora et al. [7] representa a evolução da prova de conceito para uma aplicação funcional, ao apresentar um modelo para a criação e interação de laboratórios virtuais (VCLs) utilizando a plataforma OpenStack com o hipervisor KVM. A principal motivação do trabalho é endereçar o alto custo de hardware e software em instituições de ensino, ao mesmo tempo em que se busca suprir a necessidade de experiência prática para os estudantes [7]. A proposta foca em prover uma solução de Infraestrutura como Serviço (IaaS) que permita aos alunos e professores acessar máquinas virtuais (VMs) de forma remota e sob demanda.

Para alcançar seu objetivo, a arquitetura proposta detalha a implantação de componentes essenciais do OpenStack, como o Horizon para a interface gráfica e o Keystone para o gerenciamento de identidade e segurança. O trabalho demonstra a viabilidade de se oferecer um ambiente onde os estudantes possuem privilégios administrativos sobre suas VMs, permitindo a instalação e configuração de diversos sistemas operacionais e softwares, como em pilhas LAMP ou WAMP [7]. A utilização do KVM como hipervisor é justificada por ser uma tecnologia de virtualização robusta e nativa do ecossistema Linux. A solução final permitiu à universidade otimizar o uso de seus laboratórios físicos e oferecer uma experiência de aprendizado mais rica e acessível [7].

A grande contribuição do trabalho de Arora et al. [7] reside na demonstração prática de um VCL funcional e na validação de seus benefícios para o ambiente educacional. Os

autores mostram que é possível conceder autonomia aos estudantes sem comprometer a segurança da rede da instituição, utilizando os mecanismos nativos do OpenStack [7]. No entanto, a análise do trabalho revela que seu foco está primariamente na aplicação final e no ciclo de vida do *hardware virtualizado* (as VMs). O estudo pressupõe a existência de uma plataforma OpenStack já implantada e funcional, não detalhando ou abordando os desafios de engenharia do complexo processo de instalação inicial.

Dessa forma, o estudo de Arora et al. [7] estabelece um modelo funcional para um laboratório virtual, focando nos benefícios práticos para o usuário. Uma vez que a plataforma é demonstrada como funcional para um caso de uso real, torna-se crucial validar se ela de fato cumpre seu objetivo principal: melhorar o aprendizado. A avaliação do impacto pedagógico de um VCL é precisamente o foco do trabalho de Kabiri e Wannous [9], que será analisado a seguir.

O trabalho de Kabiri e Wannous [9] foca em um aspecto crucial para a adoção de novas tecnologias no meio acadêmico: a validação de seu impacto pedagógico. O objetivo principal do estudo foi criar um laboratório virtual baseado em OpenStack e, mais importante, realizar uma avaliação experimental para medir a eficácia dessa ferramenta na melhoria da experiência de aprendizado prático dos estudantes [9]. A motivação surgiu da necessidade de adicionar um componente prático a uma solução de *e-learning* anterior que, segundo a opinião dos próprios alunos, carecia de suporte para atividades de laboratório [9].

Para realizar o estudo, os autores implementaram um laboratório virtual em um único servidor, utilizando DevStack para uma instalação de teste do OpenStack [9]. A metodologia de avaliação envolveu um experimento com um grupo de 16 estudantes dos cursos de Redes de Computadores e Administração de Sistemas [9]. O progresso do aprendizado foi quantificado através da aplicação de testes de nivelamento antes e depois da utilização do laboratório, além de um questionário para coletar a percepção dos alunos sobre a utilidade e a facilidade de uso da solução proposta [9].

A contribuição mais significativa de Kabiri e Wannous [9] é a demonstração empírica do valor educacional de um VCL baseado em OpenStack. Os resultados mostraram uma melhoria média de 17.1% nas notas dos estudantes no pós-teste, validando que a ferramenta contribui positivamente para o aprendizado prático [9]. No entanto, a implementação técnica utilizada foi uma prova de conceito em um único servidor, que não seria adequada para um ambiente de produção em larga escala. O foco do trabalho está nos resultados pedagógicos, e não na arquitetura, escalabilidade ou automação da infraestrutura de nuvem subjacente, reforçando a separação entre a pesquisa sobre eficácia educacional e a engenharia de uma plataforma robusta.

Portanto, o estudo de Kabiri e Wannous [9] fornece a justificativa pedagógica para o

investimento em laboratórios virtuais. Com o valor educacional comprovado, o desafio seguinte é transformar o protótipo de sucesso em um serviço robusto e de larga escala. Para isso, é necessário projetar uma infraestrutura resiliente e com alta disponibilidade, capaz de suportar toda a universidade, um tema detalhado no guia arquitetural de Zhao et al. [15], que será discutido na sequência.

O trabalho de Zhao et al. [15] aborda a construção de uma plataforma de nuvem privada para recursos educacionais com um foco acentuado na resiliência e na otimização da gestão de recursos em larga escala. A motivação dos autores parte da constatação de que o crescimento universitário torna os modelos de TI tradicionais insuficientes, e que plataformas de nuvem precisam ser robustas e seguras para serem viáveis em um nível de produção [15]. A solução proposta detalha um esquema de construção para uma nuvem OpenStack que utiliza o sistema de armazenamento distribuído Ceph, visando unificar o gerenciamento dos recursos e oferecê-los em um modelo de autosserviço para professores e alunos.

Para atingir um alto grau de robustez, a arquitetura proposta por Zhao et al. [15] é meticulosamente planejada com foco em alta disponibilidade (HA). Os autores descrevem uma implementação distribuída com múltiplos nós, onde cada tipo de serviço (controle, rede, armazenamento) é redundante para evitar pontos únicos de falha [15]. Tecnologias como Haproxy e Keepalived são empregadas para o balanceamento de carga e *failover* dos serviços de controle, enquanto o próprio Ceph garante a integridade e a disponibilidade dos dados através de sua arquitetura de armazenamento distribuído. Esta abordagem visa garantir a continuidade dos serviços educacionais, mesmo diante de falhas de hardware ou software em componentes individuais da nuvem [15].

A principal contribuição do estudo de Zhao et al. [15] é fornecer um guia arquitetural detalhado para a construção de uma infraestrutura de nuvem privada que não é apenas funcional, mas também altamente disponível e resiliente, um pré-requisito para qualquer serviço em ambiente de produção. No entanto, a análise do trabalho demonstra que seu foco está no *design* da arquitetura final, e não no *processo* de sua implementação. O artigo descreve em detalhes *o que* uma nuvem resiliente deve ter, mas não oferece um modelo para automatizar a complexa tarefa de implantar e configurar todos esses nós e serviços de HA.

Dessa forma, o trabalho de Zhao et al. [15] resolve o problema de como projetar uma fundação de nuvem estável e resiliente. Com a arquitetura de produção definida, a otimização do desempenho torna-se a próxima prioridade. É preciso mergulhar nos detalhes técnicos para responder a questões cruciais como: qual tecnologia de virtualização oferece a menor sobrecarga e qual tipo de *host* é mais adequado para cada carga de trabalho? Essas questões são o foco de estudos de desempenho aprofundados, como os de

Younge et al. [16] e Kominos et al. [17], que serão discutidos a seguir.

Com a infraestrutura de produção delineada, a análise de desempenho das tecnologias subjacentes torna-se imperativa para garantir que a nuvem possa atender a cargas de trabalho exigentes. Esta vertente da literatura foca em quantificar a sobrecarga (*overhead*) introduzida pelas camadas de virtualização. Trabalhos como os de Younge et al. [16] e Kominos et al. [17] são fundamentais, pois analisam, respectivamente, o desempenho do hipervisor e os diferentes tipos de hosts que podem ser provisionados sobre ele.

O estudo de Younge et al. [16] foca na comparação de desempenho de diferentes hipervisores para ambientes de Computação de Alto Desempenho (HPC). Utilizando *benchmarks* padronizados, o estudo conclui que o KVM é a escolha ideal para suportar aplicações HPC em nuvem, devido ao seu desempenho próximo ao nativo e à baixa variância nos resultados, um fator crucial para a Qualidade de Serviço (QoS) [16]. Essa validação empírica solidifica a escolha do KVM como a tecnologia de virtualização base para nuvens que necessitam de alta performance e previsibilidade.

De forma complementar, o estudo de Kominos et al. [17] avalia o desempenho não apenas do hipervisor, mas dos diferentes tipos de hosts que o OpenStack pode provisionar: *Bare-metal*, Máquinas Virtuais e Contêineres. A análise quantifica os *trade-offs* entre desempenho bruto e flexibilidade, mostrando que contêineres oferecem uma performance quase nativa com tempos de inicialização muito mais rápidos, enquanto VMs apresentam uma sobrecarga mais significativa em tarefas de I/O de disco e CPU [17]. Juntos, esses estudos fornecem um guia robusto para a seleção dos componentes da camada de computação, mas partem do pressuposto de que a plataforma já está implantada e funcional.

No entanto, a otimização da camada de computação é apenas parte da solução. O foco desses trabalhos na performance do *host* individual não aborda como a própria operação da infraestrutura OpenStack, com seus múltiplos serviços em segundo plano, pode interferir no desempenho de aplicações sensíveis. Mesmo com o hipervisor e o tipo de host mais performáticos, a interação com a plataforma de nuvem pode gerar gargalos inesperados, como será visto na análise de interferência sutil realizada por Bystrov et al.

O trabalho de Bystrov et al. [18] aprofunda a análise de desempenho em nuvens OpenStack, investigando um cenário complexo que *benchmarks* padrão geralmente não cobrem. O objetivo do estudo é analisar a performance de um software de alta intensidade computacional e de comunicação, baseado no Método dos Elementos Discretos (DEM) e paralelizado com MPI, quando implantado como um serviço (SaaS) na nuvem [18]. A motivação é entender como a infraestrutura de nuvem e as questões de comunicação afetam o desempenho de computações paralelas, um tópico que, segundo os autores, permanece pouco claro na literatura [18].

A metodologia adotada envolve a implantação de um software científico real em uma

nuvem OpenStack, comparando seu desempenho em máquinas virtuais KVM e contêineres Docker com a execução em hardware nativo [18]. A análise vai além da medição de sobrecarga (*overhead*) total e investiga as fontes dessa perda de desempenho, focando no impacto da comunicação MPI e no mapeamento dos processos da aplicação aos recursos *multicore* dos nós físicos [18]. Os autores demonstram que otimizar o mapeamento da aplicação pode reduzir a comunicação entre os nós em mais de 34% e aumentar significativamente a eficiência paralela [18].

A principal contribuição do estudo de Bystrov et al. [18] é o diagnóstico preciso de um problema sutil: a sobrecarga da infraestrutura OpenStack (que pode chegar a 13.7%) é causada majoritariamente pelo aumento do desbalanceamento de carga da aplicação paralela. Esse desbalanceamento ocorre porque os processos de serviço do próprio OpenStack, executados em segundo plano, competem por tempo de CPU com os processos da aplicação [18]. A limitação do trabalho, no entanto, é que a solução explorada para este problema ocorre no nível da aplicação (re-mapeamento de processos), e não no nível da infraestrutura, e para ser estudada, pressupõe que a complexa etapa de implantação da nuvem já foi superada.

Portanto, a análise de Bystrov et al. [18] é fundamental por expor a complexa e delicada interação entre aplicações de alto desempenho e os serviços da plataforma de nuvem. Ela revela que o desempenho não depende apenas da virtualização, mas de toda a operação da nuvem. Com a conclusão desta e das análises anteriores, que cobriram desde a escolha da plataforma até os desafios de implementação e performance, formamos um panorama coeso sobre o estado da arte, cujas lacunas recorrentes serão sintetizadas a seguir para justificar de forma definitiva a proposta desta monografia.

2.2.1 Síntese da Lacuna na Literatura e Justificativa da Proposta

A análise aprofundada dos trabalhos apresentados nas seções anteriores permite traçar um panorama claro sobre o estado da arte na utilização de nuvens OpenStack e, ao mesmo tempo, revela um paradoxo central. A literatura é robusta em estabelecer a plataforma como uma solução IaaS de código aberto de referência [12, 6] e em validar seu imenso valor para ambientes acadêmicos, seja para otimizar o uso de hardware [10], prover laboratórios virtuais flexíveis [7] ou gerar ganhos pedagógicos mensuráveis [9]. O consenso é que o resultado final (uma nuvem privada funcional) é um objetivo estratégico e altamente desejável para instituições de ensino.

Contudo, a mesma literatura evidencia, de forma consistente, a significativa complexidade técnica e a alta barreira de entrada para se alcançar esses resultados. A arquitetura

inerentemente modular do OpenStack torna seu processo de implantação manual uma tarefa longa, minuciosa e propensa a erros, como detalhado por guias como o de Kumar et al. [11]. Este desafio é agravado por questões de segurança, onde uma configuração manual incorreta pode facilmente gerar vulnerabilidades críticas, conforme a revisão de Albaroodi et al. [14]. A dificuldade de implantação é tão pronunciada que a própria literatura já propõe soluções de automação como um campo de estudo relevante [13] e mostra pesquisadores utilizando ferramentas de teste como o DevStack para contornar a complexidade da instalação de produção [9].

A lacuna na literatura reside, portanto, precisamente nesta desconexão entre o objetivo estratégico desejado (os benefícios da nuvem) e a capacidade operacional para alcançá-lo (a complexidade da implantação). Falta um modelo prático e documentado que utilize ferramentas de automação modernas e acessíveis, como o Ansible, para transformar o processo de implantação de um desafio hercúleo em um fluxo de trabalho gerenciável e repetível. Esta lacuna é especialmente crítica para instituições acadêmicas, que frequentemente possuem equipes de TI enxutas e buscam, como no caso deste trabalho, reaproveitar hardware existente para construir sua infraestrutura.

É para construir a ponte sobre essa lacuna que a presente monografia se insere. O modelo para a implantação automatizada de uma nuvem privada OpenStack, que será detalhado a seguir, ataca diretamente o problema da complexidade da instalação. Ao encapsular os passos manuais em um processo automatizado, a proposta visa tornar a criação de uma infraestrutura de nuvem privada uma tarefa mais acessível, rápida e confiável. Com a necessidade da solução firmemente estabelecida pela análise do estado da arte, o capítulo seguinte procederá ao detalhamento técnico do modelo de automação proposto.

Capítulo 3

Proposta

Conforme estabelecido pela análise da literatura, existe uma lacuna significativa entre o valor estratégico de uma nuvem privada OpenStack em ambientes acadêmicos e a complexidade técnica de sua implantação. A presente proposta visa endereçar este desafio através de um modelo prático para a implantação automatizada de uma infraestrutura de nuvem, projetado especificamente para o contexto de reaproveitamento de hardware legado em laboratórios universitários. Este capítulo detalha a solução proposta, abordando seus objetivos e requisitos, a arquitetura de implantação, o processo de automação com a ferramenta Ansible e o modelo de configuração adotado.

Objetivos da Proposta

Considerando o problema da complexidade da implantação do OpenStack, que representa uma barreira para ambientes acadêmicos, e a necessidade de otimizar o uso de hardware para atender à demanda por recursos computacionais para simulações estudantis, esta proposta tem como objetivo geral:

> Propor e validar um modelo prático para a implantação automatizada de uma nuvem privada IaaS baseada em OpenStack, focado no reaproveitamento de hardware em um ambiente de laboratório.

Para alcançar este objetivo principal, foram definidos os seguintes objetivos específicos:

- **Revitalizar a Infraestrutura Física:** Detalhar o processo de preparação e revitalização de servidores heterogêneos em desuso, incluindo a configuração de arranjos de disco (RAID) e a definição de uma topologia de rede base para a nuvem.
- **Automatizar a Implantação da Plataforma:** Desenvolver um conjunto de *scripts* de automação utilizando a ferramenta Ansible para orquestrar a instalação e a configuração completas da plataforma OpenStack, garantindo uma implantação consistente, padronizada e repetível entre os nós.

- **Validar a Funcionalidade da Nuvem:** Demonstrar a eficácia do modelo de implantação através da criação de uma nuvem privada funcional, capaz de provisionar máquinas virtuais com os sistemas operacionais (Windows e Ubuntu) e as configurações levantadas junto aos alunos do laboratório.
- **Documentar um Modelo Replicável:** Apresentar o processo completo como um modelo documentado que possa ser seguido por outras instituições de ensino que enfrentam desafios similares de recursos e que buscam implementar suas próprias soluções de nuvem privada de forma eficiente.

Requisitos da Proposta

Para garantir que o modelo proposto atenda tanto aos objetivos operacionais de automação quanto às necessidades práticas dos usuários finais (os alunos), foram definidos os seguintes requisitos funcionais e não-funcionais.

Requisitos Funcionais Os requisitos funcionais descrevem o que o sistema, como um todo, deve ser capaz de fazer. Com base no levantamento realizado junto aos estudantes, a infraestrutura de nuvem resultante da implantação deve ser capaz de:

- Provisionar Máquinas Virtuais (VMs) sob demanda para os alunos do laboratório.
- Suportar a instalação e execução de sistemas operacionais Microsoft Windows de 64-bits, especificamente Windows 10 e Windows 11 (versões Professional, Enterprise ou Education) ou suas contrapartes de servidor (Windows Server 2019/2022).
- Executar de forma satisfatória os seguintes softwares de simulação, que são essenciais para as atividades acadêmicas:
  - Ansys HFSS;
  - Keysight ADS (Advanced Design System);
  - Altium Designer 24.

Requisitos Não-Funcionais Os requisitos não-funcionais descrevem as qualidades, restrições e características de como o sistema deve operar.

- **Desempenho de GPU:** Para atender às exigências de hardware dos softwares de simulação, as VMs provisionadas devem ter acesso a recursos de processamento gráfico. O modelo deve suportar a alocação de Unidades de Processamento Gráfico (GPUs) dedicadas às VMs, com capacidade mínima de 4 GB de VRAM e desempenho compatível com as recomendações dos fabricantes de software. Isso implica a necessidade de configuração de *GPU Passthrough* no nível do hipervisor KVM.

- **Gerenciamento de Licenças:** O ambiente virtual deve ser estável e configurado de forma a não violar os termos de licenciamento dos softwares utilizados. A solução deve permitir que as VMs se comuniquem com servidores de licença centralizados da universidade e, se necessário, manter características de hardware virtual (como endereço MAC) consistentes para licenças vinculadas a um *host* específico.
- **Automação da Implantação:** O processo de instalação e configuração da própria plataforma OpenStack nos servidores físicos deve ser automatizado por meio de *scripts* Ansible. Este requisito visa garantir a consistência, a repetibilidade e a aplicação de uma linha de base de segurança em toda a infraestrutura, reduzindo a complexidade e o risco de erro humano.
- **Idempotência:** A execução repetida dos scripts de automação Ansible não deve alterar o estado do sistema se nenhuma mudança de configuração for especificada, garantindo a estabilidade em operações de manutenção.
- **Restrição de Hardware:** A solução completa deve ser implementada utilizando como base a coleção de servidores disponíveis no laboratório, conforme o levantamento físico realizado.

3.0.1 Arquitetura do Modelo de Implantação

A proposta desta monografia foi concebida a partir de um cenário prático e comum em ambientes acadêmicos: a existência de um conjunto de servidores e equipamentos de hardware legados, em desuso, em um laboratório de pesquisa. O primeiro passo consistiu, portanto, em um levantamento completo dos equipamentos que poderiam ser revitalizados para servir de base para a futura nuvem privada. A Tabela 3.1 detalha os equipamentos que estavam parados e foram inventariados para este projeto.

Tabela 3.1: Levantamento dos equipamentos em desuso no Laboratório

Equipamento	**Qtde**	**Especificações e Descrição**
Dell PowerEdge 1950 1U	1	Processador Intel Xeon; suporte a DDR2/DDR3.
Dell PowerEdge R610 1U	2	Processador Intel Xeon; suporte a DDR2/DDR3.
Dell PowerEdge 2950 2U	1	Processador Intel Xeon; suporte a DDR2/DDR3.
Dell PowerEdge 2900 4U	3	Processador Intel Xeon; suporte a DDR2/DDR3.
Dell PowerEdge R900 4U	3	Processador Intel Xeon; suporte a DDR2/DDR3.

O processo inicial envolveu a preparação física dos servidores selecionados desta lista, que incluiu limpeza interna, instalação de módulos de memória RAM e discos rígidos adicionais, aplicação de nova pasta térmica nos processadores e a configuração de arranjos de disco (RAID) para garantir redundância e desempenho. Após esta análise, cinco servidores da família Dell PowerEdge foram escolhidos para compor a infraestrutura inicial da nuvem. Com base no inventário apresentado, o processo inicial envolveu a preparação física dos servidores Dell PowerEdge selecionados. Esta etapa incluiu a limpeza interna, a instalação de módulos de memória e discos rígidos, a aplicação de nova pasta térmica nos processadores e a configuração de arranjos de disco (RAID) via controladora de hardware PERC para garantir redundância. O plano inicial era padronizar todos os nós com arranjos em RAID 6 para maximizar a proteção de dados. Contudo, durante a preparação, constatou-se que um dos servidores possuía apenas duas unidades de disco disponíveis, uma restrição física que impossibilitava a criação de um RAID 6. Diante deste cenário, foi adotada uma configuração em RAID 1 (espelhamento) para este nó específico, garantindo a redundância possível dentro das limitações do hardware. A arquitetura lógica foi então projetada para otimizar o uso deste conjunto de hardware heterogêneo. Foi realizada uma análise de *trade-offs* para a alocação dos papéis de **Nó controlador (*Controller Node*)** e **Nós de computação (*Compute Nodes*)**. O servidor 'lab103' foi estrategicamente designado como Nó controlador por apresentar um equilíbrio entre desempenho e capacidade de memória RAM (15 GiB), sendo robusto o suficiente para gerenciar os serviços da plataforma sem comprometer excessivamente os recursos disponíveis para as VMs. Os servidores com maior capacidade de memória, como o 'lab101' (47 GiB) e o 'lab104' (54 GiB), foram designados como Nós de computação para maximizar a quantidade de simulações pesadas que poderiam ser executadas simultaneamente, atendendo diretamente ao objetivo principal do projeto.

A topologia de rede foi estabelecida conectando todos os cinco servidores a um *switch* de gerência através de suas interfaces de rede primárias (eno1). Para garantir a estabilidade e a comunicação entre os componentes do OpenStack, foi implementado um plano de endereçamento IP estático, utilizando a faixa de rede de `192.168.1.100` a `192.168.1.200`, que foi liberada no *firewall* do laboratório. O acesso remoto ao ambiente para os usuários foi garantido através da VPN da instituição.

3.0.2 Detalhamento do Processo de Automação com Ansible

Para resolver a complexidade da implantação manual e garantir um processo padronizado e repetível, o núcleo desta proposta é a utilização da ferramenta de automação Ansible. O processo de implantação foi encapsulado em um conjunto de *playbooks*, *roles* e *templates*,

que orquestram todas as tarefas necessárias de forma sequencial. Esta seção detalha a estrutura e a lógica de execução desta automação.

Estrutura do Projeto Ansible

O projeto de automação foi organizado de forma modular para facilitar a manutenção e a reutilização do código, seguindo as melhores práticas recomendadas pela comunidade Ansible. A estrutura, conforme foi detalhado no inventário de arquivos do projeto, utiliza *roles* para encapsular a lógica de cada componente, um arquivo de inventário para definir os *hosts*, e uma pasta `group_vars` para centralizar as variáveis de configuração.

Os principais componentes da estrutura são:

- ***Playbooks* Principais:** Dois arquivos YAML principais orquestram todo o fluxo: `install_openstack_controller.yaml` e `install_openstack_compute.yaml`. O primeiro é responsável por toda a configuração do Nó controlador, enquanto o segundo é aplicado a todos os Nós de computação.
- **Roles:** O código foi dividido em duas *roles* principais: `controller` e `compute`. Cada *role* contém, em sua subpasta `tasks/main.yml`, a sequência de tarefas específicas para configurar aquele tipo de nó. A pasta `templates/` dentro de cada *role* armazena os modelos dos arquivos de configuração (ex: `nova.conf.j2`, `keystone.conf.j2`), que são preenchidos com as variáveis durante a execução.
- **Inventário:** O arquivo de inventário (`hosts.ini` ou `inventory.yaml`) define os grupos de servidores. Para este projeto, foram definidos dois grupos: `[controller]`, contendo o host local (‘192.168.1.103‘), e `[compute]`, contendo os quatro servidores dedicados à computação (‘192.168.1.100‘ a ‘192.168.1.104‘).
- **Variáveis Centralizadas:** Todas as variáveis de configuração, como senhas, endereços IP e a versão do OpenStack (Caracal), são definidas no arquivo `group_vars/all.yml`. Esta prática de separar a lógica da configuração em si é fundamental para a segurança e reusabilidade do modelo.

Fluxo de Execução da Automação

O processo de implantação automatizada é executado a partir do Nó controlador em duas fases distintas, garantindo que as dependências entre os serviços do OpenStack sejam respeitadas.

Fase 1 - Implantação do Nó de Controle: A primeira fase é iniciada pela execução do playbook `install_openstack_controller.yaml`. Este playbook aplica a *role*

`controller` ao host local (‘localhost‘). As tarefas dentro desta *role* são responsáveis por instalar e configurar todos os serviços de suporte (MariaDB, RabbitMQ) e os serviços centrais de gerenciamento do OpenStack, como Keystone, Glance, as APIs do Nova, o servidor Neutron e o dashboard Horizon. Ao final desta fase, o Nó controlador está totalmente funcional e pronto para gerenciar o restante da nuvem.

Fase 2 - Implantação dos Nós de Computação: A segunda fase é disparada pela execução do playbook `install_openstack_compute.yaml`. Este playbook aplica a *role* `compute` a todos os hosts definidos no grupo `[compute]` do inventário. As tarefas desta *role* instalam os pacotes necessários em cada nó (como ‘nova-compute‘ e ‘neutron-linuxbridge-agent‘), configuram a comunicação com os serviços centralizados no Nó controlador e iniciam o serviço de hipervisor. Ao término desta fase, os Nós de computação se registram automaticamente no Nó controlador e estão prontos para hospedar máquinas virtuais.

Procedimentos Pós-Automação: Com a infraestrutura totalmente implantada pela automação, alguns procedimentos finais são realizados manualmente. Isso inclui o acesso ao *dashboard* Horizon para verificação do status dos serviços, o upload de imagens de sistema operacional específicas (como a do Windows Server) para o serviço Glance, e a criação de projetos, redes e usuários iniciais para validar a funcionalidade completa do ambiente de nuvem.

3.0.3 Modelo de Configuração e Inventário

Para garantir a flexibilidade e a reusabilidade do modelo de automação, um dos princípios de design adotados foi a separação estrita entre a *lógica* de automação (contida nos playbooks e roles) e a *configuração* do ambiente (contida em arquivos de inventário e de variáveis). Esta abordagem permite que a solução seja facilmente adaptada para diferentes cenários de hardware ou requisitos de segurança, alterando apenas os arquivos de configuração, sem a necessidade de modificar o código dos *playbooks*.

Arquivo de Inventário

O inventário do Ansible é o componente responsável por mapear a arquitetura lógica da nuvem para os servidores físicos. Para este projeto, foi utilizado um arquivo de inventário no formato INI (`hosts.ini`) para definir os grupos de *hosts*. Conforme ilustrado no código de exemplo abaixo, foram criados dois grupos principais: `[controller]`, que contém o nó que centralizará os serviços de gerenciamento, e `[compute]`, que agrupa todos os servidores que atuarão como nós de computação.

```
[controller]
localhost ansible_connection=local ...

[compute]
192.168.1.100 ansible_user=lab100 ...
192.168.1.101 ansible_user=lab101 ...
192.168.1.102 ansible_user=lab102 ...
192.168.1.104 ansible_user=lab104 ...
```

Essa estrutura permite que os *playbooks* apliquem as *roles* corretas a cada grupo de servidores de forma direcionada. Para adaptar o modelo a uma nova infraestrutura, bastaria ao administrador editar este arquivo, modificando os endereços IP e as credenciais de acesso para os novos servidores.

Arquivo de Variáveis Centralizadas

Todas as configurações sensíveis e os parâmetros que podem variar entre implantações foram externalizados para um arquivo de variáveis centralizado, localizado em `group_vars/all.yml`. Esta prática é fundamental para a segurança, pois evita que senhas sejam armazenadas diretamente no código dos *playbooks*, e aumenta a manutenibilidade do projeto.

O arquivo de variáveis define parâmetros críticos, como:

- **Credenciais:** Senhas para o usuário 'admin' do OpenStack, para o banco de dados MySQL e para o serviço de mensageria RabbitMQ.
- **Configuração da Rede:** O endereço IP do nó de controle, que é utilizado como endpoint para todos os outros serviços.
- **Versão do OpenStack:** A versão específica a ser implantada (ex: 'caracal'), permitindo que a automação seja facilmente atualizada para futuras versões do OpenStack com a alteração de uma única variável.

A separação entre o inventário de hosts e as variáveis de configuração constitui o núcleo do modelo proposto, garantindo que a solução seja não apenas uma automação de tarefas, mas um verdadeiro *framework* de implantação, adaptável a diferentes realidades de hardware e requisitos específicos de cada ambiente acadêmico. Com a proposta conceitual agora detalhada, o próximo capítulo descreverá a implementação prática deste modelo e os resultados obtidos.

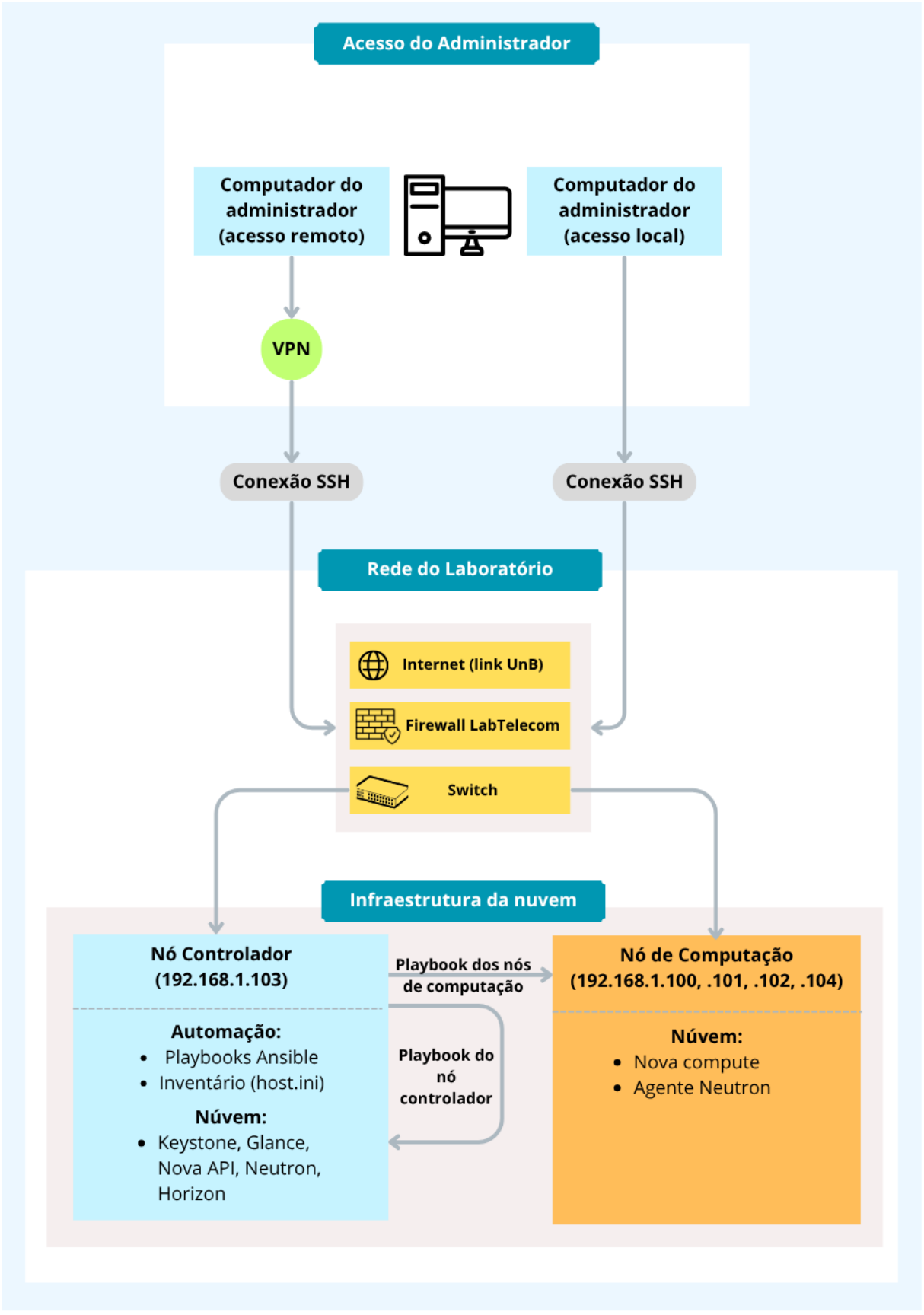


Figura 3.1: Diagrama da Arquitetura do Modelo de Implantação Proposto. Fonte: Elaborado pelo autor (2025).

Capítulo 4

Prototipo

O capítulo anterior apresentou o modelo conceitual para a implantação automatizada de uma nuvem privada OpenStack. O presente capítulo descreve a implementação prática e a validação deste modelo através da construção de um protótipo funcional em um ambiente de laboratório. A seguir, são detalhados o ambiente de hardware e software utilizado, os procedimentos manuais de preparação da infraestrutura que serviram de pré-requisito para a automação e, por fim, a execução do protótipo baseado em Ansible para a implantação da nuvem. O objetivo é demonstrar a aplicabilidade e a eficácia da proposta em um cenário real de reaproveitamento de hardware legado.

4.0.1 Ambiente de Hardware e Software Utilizado

O protótipo do modelo de implantação automatizada foi implementado e validado em um ambiente de laboratório, utilizando um conjunto de cinco servidores Dell PowerEdge heterogêneos que foram revitalizados para este projeto. Esta seção detalha a configuração final do hardware e do software que compõem a infraestrutura da nuvem privada.

Infraestrutura de Hardware e Configuração de Disco

A infraestrutura física é composta por cinco servidores que, antes da implantação, passaram por um processo de preparação, incluindo limpeza, instalação de componentes e configuração de arranjos de disco (RAID) via controladora de hardware PERC. A configuração de disco foi adaptada às restrições de cada equipamento: o Nó de Controle (`lab103`) e o Nó de Computação `lab102` (que possuía limitação de discos) foram configurados com RAID 1 (espelhamento) para redundância. Os demais Nós de Computação foram configurados com RAID 6 para um equilíbrio entre capacidade e proteção de dados. A Tabela 4.1 resume as especificações detalhadas de cada nó utilizado.

Tabela 4.1: Especificações Técnicas dos Servidores Utilizados na Nuvem Privada

Papel	**Hostname**	**Modelo da CPU**	**Núcleos**	**RAM Total**
Nó de Controle	lab103	Intel Xeon E5410 @ 2.33GHz	4	15 GiB
Nó de Computação	lab100	Intel Xeon E5410 @ 2.33GHz	4	23 GiB
Nó de Computação	lab101	Intel Xeon E5410 @ 2.33GHz	4	47 GiB
Nó de Computação	lab102	Intel Xeon E5410 @ 2.33GHz	4	31 GiB
Nó de Computação	lab104	Intel Xeon E7430 @ 2.13GHz	1	54 GiB

Ambiente de Software

Para garantir a compatibilidade e a estabilidade da implantação, todo o ambiente de software foi padronizado. As especificações são as seguintes:

- **Sistema Operacional Base:** Todos os cinco servidores foram provisionados com o sistema operacional **Ubuntu Server 22.04 LTS**, escolhido por seu amplo suporte da comunidade, documentação robusta e compatibilidade nativa com os pacotes do OpenStack.
- **Plataforma de Nuvem:** O alvo da automação foi a implantação do **OpenStack versão 2024.1 (Caracal)**, utilizando os pacotes disponíveis no repositório oficial Cloud Archive para o Ubuntu.
- **Ferramenta de Automação:** Todo o processo de implantação foi orquestrado utilizando a ferramenta **Ansible**, em sua versão instalada no Nó de Controle. Os *playbooks* foram desenvolvidos em sintaxe YAML, utilizando módulos nativos do Ansible para gerenciamento de pacotes, *templates* e serviços.

4.0.2 Metodologia de Implantação do Protótipo

A implementação do protótipo foi dividida em duas grandes etapas, seguindo uma abordagem sistemática para garantir a robustez e a repetibilidade do processo. A primeira etapa consistiu em uma fase de preparação manual, para garantir que a infraestrutura física e o sistema operacional base estivessem prontos e padronizados. A segunda etapa, o núcleo do trabalho, foi a execução da automação para a implantação dos serviços OpenStack.

Preparação Manual da Infraestrutura (Pré-requisitos)

Revitalização do Hardware: Cada um dos cinco servidores Dell PowerEdge selecionados foi fisicamente inspecionado, teve seus componentes internos limpos, e recebeu a instalação de módulos de memória e discos rígidos. Adicionalmente, foi aplicada nova pasta térmica nos processadores.

Instalação do SO e Configuração de Rede: O sistema operacional **Ubuntu Server 22.04 LTS** foi instalado manualmente em todos os cinco nós. Em seguida, foi executado um processo meticuloso para configurar um endereço IP estático em cada servidor, detalhado a seguir.

1. **Desabilitar o controle de rede do `cloud-init`:** Para assegurar que as definições manuais não fossem sobrescritas, o controle de rede do utilitário foi desabilitado em cada nó.

   ```
   sudo mkdir -p /etc/cloud/cloud.cfg.d
   echo "network: {config: disabled}"
   | sudo tee /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg
   ```

2. **Desativar a configuração padrão do Netplan:** O arquivo de configuração gerado pela instalação inicial foi renomeado para desativá-lo.

   ```
   sudo mv /etc/netplan/50-cloud-init.yaml
   /etc/netplan/50-cloud-init.yaml.bak
   ```

3. **Criar e aplicar o novo arquivo de configuração Netplan:** Um novo arquivo, `/etc/netplan/01-static-ip.yaml`, foi criado com as definições de IP estático.

   ```
   # Exemplo de configuração para um nó
   network:
     version: 2
     ethernets:
       eno1:
         dhcp4: false
         addresses:
           - 192.168.1.101/24
         routes:
   ```

```
    - to: default
      via: 192.168.1.1
  nameservers:
    addresses: [8.8.8.8, 8.8.4.4]
```

Após a criação, as permissões do arquivo foram ajustadas com `sudo chmod 600` e a configuração ativada com `sudo netplan apply`.

Configuração de Reservas no Firewall: Para garantir uma camada adicional de robustez, foi configurada uma reserva de endereço no servidor DHCP central do LabTelecom (FreeBSD). Este procedimento garante que, mesmo que um nó tente obter um IP via DHCP, ele sempre receberá o endereço correto. Para cada servidor, um bloco de *host* foi adicionado no arquivo `/usr/local/etc/dhcpd.conf`, vinculando o endereço MAC ao seu respectivo IP estático, como no exemplo abaixo.

```
# Exemplo de reservas de IP no servidor DHCP
host lab101 {
    hardware ethernet 00:22:ww:xx:yy:zz;
    fixed-address 192.168.1.101;
}
# ... (entradas para os outros hosts)
```

Após a edição, o serviço DHCP foi reiniciado com `sudo service isc-dhcpd restart` e os *leases* foram verificados para confirmar a correta atribuição.

Implementação dos Artefatos de Automação

Com a infraestrutura base preparada, a fase seguinte consistiu na implementação dos artefatos de automação que constituem o núcleo deste trabalho.

Estrutura do Projeto Ansible: O protótipo de automação foi desenvolvido com uma estrutura de diretórios modular que separa a lógica da configuração. Esta organização, localizada no diretório `~/ansible-openstack/` do Nó de Controle, facilita a manutenção e a reutilização do código. Os principais componentes são os *playbooks* principais, um inventário de *hosts*, variáveis centralizadas e as *roles* `controller` e `compute`.

Inventário e Variáveis: O arquivo `hosts.ini` foi utilizado para mapear os servidores aos seus papéis lógicos, enquanto o arquivo `group_vars/all.yml` centralizou todas as configurações sensíveis, como senhas e a *release* do OpenStack, garantindo um modelo seguro e flexível.

Lógica da Automação em Roles: A lógica de configuração foi encapsulada em duas *roles* principais. A *role* `controller` contém as tarefas para instalar e configurar os serviços centrais da nuvem (Keystone, Glance, etc.), enquanto a *role* `compute` é responsável por instalar o hipervisor e os agentes de serviço nos nós de computação.

Orquestração com Playbooks: Os *playbooks* `install_openstack_controller.yaml` e `install_openstack_compute.yaml` atuam como o ponto de entrada, orquestrando a execução das *roles* nos grupos de *hosts* corretos e na ordem de dependência apropriada, garantindo uma implantação faseada e controlada.

4.0.3 Execução e Validação do Protótipo

Com a infraestrutura física preparada e os artefatos de automação desenvolvidos, a etapa final consistiu na execução do protótipo e na subsequente validação da nuvem OpenStack implantada. Esta seção descreve o processo de execução dos *playbooks* Ansible e os testes funcionais realizados para verificar se o ambiente estava totalmente operacional e apto a cumprir seus objetivos.

Execução da Automação com Ansible

O processo de implantação foi iniciado a partir do terminal do Nó de Controle (‘lab103‘), com a execução sequencial dos dois *playbooks* principais. Primeiramente, o *playbook* do controlador foi invocado com o seguinte comando:

```
ansible-playbook -i hosts.ini install_openstack_controller.yaml
```

A execução deste *playbook* orquestrou a instalação e configuração de todos os serviços centrais no Nó de Controle. Após sua conclusão bem-sucedida, o segundo *playbook* foi executado para provisionar os nós de computação:

```
ansible-playbook -i hosts.ini install_openstack_compute.yaml
```

Este comando aplicou a configuração necessária aos quatro nós de computação em paralelo. A execução completa de ambos os *playbooks*, , . Este resultado, por si só, valida um dos principais benefícios do modelo proposto: a drástica redução no tempo e no esforço de implantação em comparação com o processo manual, que poderia levar várias horas ou dias.

Validação Funcional da Infraestrutura

Após o término da automação, a primeira etapa de validação foi verificar o estado dos serviços do OpenStack. Para interagir com a API da nuvem via linha de comando, foi necessário carregar as credenciais de administrador, utilizando o *script* gerado pelo Keystone durante a instalação:

```
source admin-openrc.sh
```

Com o ambiente configurado, o comando `openstack compute service list` foi executado para verificar o status dos serviços do Nova. A saída do comando confirmou que o *scheduler*, o *conductor* e os quatro serviços `nova-compute` (um para cada nó de computação) estavam ativos e em estado *up*, indicando que os nós de computação haviam se registrado com sucesso no controlador. De forma similar, o comando `openstack network agent list` foi utilizado para validar que os agentes de rede estavam ativos e operacionais.

```
lab103@lab103:~$  source ~/admin-openrc.sh
lab103@lab103:~$ openstack compute service list
+--------------------------------------+----------------+--------+----------+---------+-------+----------------------------+
| ID                                   | Binary         | Host   | Zone     | Status  | State | Updated At                 |
+--------------------------------------+----------------+--------+----------+---------+-------+----------------------------+
| [redacted]                           | nova-conductor | lab103 | internal | enabled | up    | 2025-06-30T00:09:07.000000 |
|                                      | nova-scheduler | lab103 | internal | enabled | up    | 2025-06-30T00:08:58.000000 |
|                                      | nova-compute   | lab101 | nova     | enabled | up    | 2025-06-30T00:09:01.000000 |
|                                      | nova-compute   | lab102 | nova     | enabled | up    | 2025-06-30T00:09:02.000000 |
|                                      | nova-compute   | lab103 | nova     | enabled | down  | 2025-05-12T01:23:06.000000 |
|                                      | nova-compute   | lab104 | nova     | enabled | up    | 2025-06-30T00:09:05.000000 |
|                                      | nova-compute   | lab100 | nova     | enabled | up    | 2025-06-30T00:09:04.000000 |
+--------------------------------------+----------------+--------+----------+---------+-------+----------------------------+
```

Figura 4.1: Saída dos comandos de verificação de serviços, confirmando o estado operacional da nuvem.

```
lab103@lab103:~$ openstack network agent list
+--------------------------------------+--------------------+--------+-------------------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host   | Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+--------+-------------------+-------+-------+---------------------------+
| [redacted]                           | Metadata agent     | lab104 | None              | :-)   | UP    | neutron-metadata-agent    |
|                                      | Metadata agent     | lab100 | None              | :-)   | UP    | neutron-metadata-agent    |
|                                      | Linux bridge agent | lab101 | None              | :-)   | UP    | neutron-linuxbridge-agent |
|                                      | Linux bridge agent | lab103 | None              | :-)   | UP    | neutron-linuxbridge-agent |
|                                      | Open vSwitch agent | lab103 | None              | XXX   | UP    | neutron-openvswitch-agent |
|                                      | Metadata agent     | lab103 | None              | :-)   | UP    | neutron-metadata-agent    |
|                                      | Metadata agent     | lab101 | None              | :-)   | UP    | neutron-metadata-agent    |
|                                      | Metadata agent     | lab102 | None              | XXX   | UP    | neutron-metadata-agent    |
|                                      | DHCP agent         | lab103 | nova              | :-)   | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+--------+-------------------+-------+-------+---------------------------+
```

Figura 4.2: Saída dos comandos de verificação dos agentes.

Validação do Caso de Uso Final

A validação final do protótipo foi realizada em duas etapas sequenciais. A primeira consistiu em um teste de funcionalidade base da nuvem, utilizando uma imagem mínima e leve. Uma vez confirmada a operacionalidade dos serviços *core*, a segunda etapa focou

na validação do caso de uso principal: o provisionamento de máquinas virtuais Windows para atender aos requisitos dos alunos.

Teste de Funcionalidade com CirrOS Para realizar um teste rápido e de baixo custo de recursos, foi utilizada a imagem do CirrOS, uma distribuição Linux mínima projetada especificamente para ambientes de nuvem. Primeiramente, a imagem do CirrOS foi carregada no serviço Glance. Como esta imagem possui requisitos de hardware muito baixos, foi necessário criar um novo *flavor* no OpenStack, nomeado `flavor-cirros`, com especificações modestas (ex: 1 vCPU, 512 MB de RAM e 1 GB de disco).

Com a imagem e o *flavor* prontos, uma instância de teste, nomeada `test-cirros`, foi provisionada através da linha de comando. A Figura 4.3 apresenta a saída do comando `openstack server show test-cirros`, que detalha os atributos da máquina virtual durante e após o seu provisionamento. O sucesso na criação desta instância, que obteve um endereço IP e atingiu o estado *UP*, serviu como a primeira validação completa da cadeia de serviços: Keystone (autenticação), Glance (imagem), Neutron (rede) e Nova (computação) estavam todos interagindo corretamente.

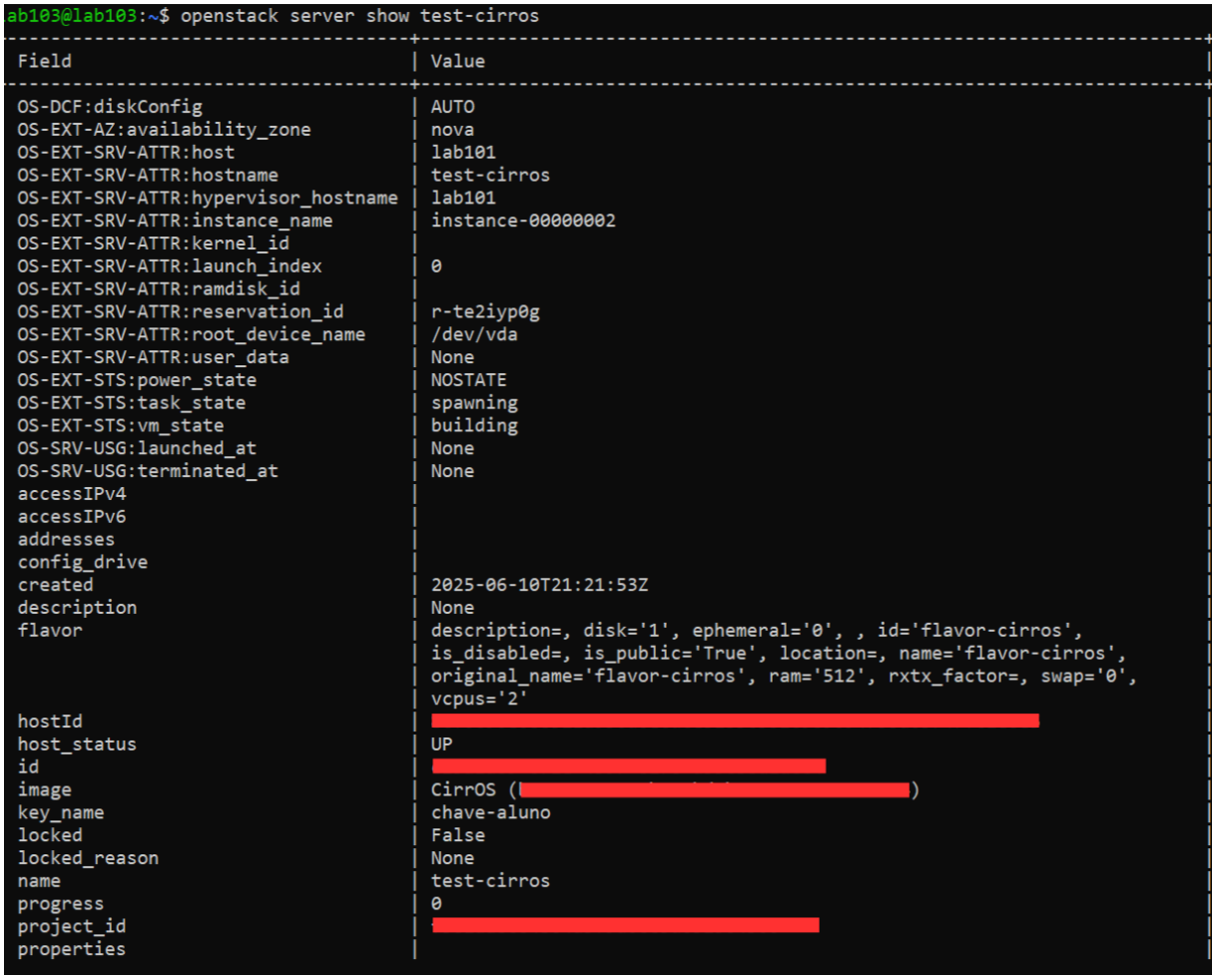


Figura 4.3: Saída do comando `openstack server show` para a instância de teste `test-cirros`.

Capítulo 5

Resultados

O capítulo anterior descreveu o modelo de automação proposto e a metodologia prática utilizada para implementar um protótipo funcional a partir de hardware legado. O presente capítulo tem como objetivo apresentar e analisar os resultados obtidos com esta implementação. A análise será dividida em duas vertentes principais: primeiramente, serão apresentados os resultados quantitativos e qualitativos da validação funcional da infraestrutura, demonstrando que o processo de automação foi bem-sucedido. Em seguida, será detalhado um estudo de caso que valida o principal objetivo do projeto: o provisionamento de um ambiente de computação para atender às necessidades de simulação dos alunos. Por fim, será realizada uma discussão sobre as implicações práticas da solução e os desafios encontrados durante o desenvolvimento.

5.0.1 Resultados da Automação da Implantação

A validação primária do modelo proposto consistiu em medir a eficácia e a eficiência do processo de implantação automatizada. Os resultados desta fase são analisados tanto do ponto de vista quantitativo, através do tempo de execução, quanto qualitativo, pela verificação da consistência e da idempotência da solução.

Análise Quantitativa: Tempo de Implantação

O principal indicador de desempenho para a automação é o tempo total necessário para provisionar a infraestrutura completa. A execução dos playbooks Ansible foi cronometrada a partir do início do primeiro playbook (`install_openstack_controller.yaml`) até a conclusão do segundo (`install_openstack_compute.yaml`). O processo completo de implantação para a infraestrutura de cinco nós (um controlador e quatro de computação) levou um total de aproximadamente 2 horas.

Este resultado valida um dos principais objetivos da proposta: a drástica redução no esforço e no tempo de implantação. Uma instalação manual, seguindo um guia como o de Kumar et al. [11], é um processo que poderia levar várias horas ou mesmo dias de trabalho de um administrador, além de ser altamente suscetível a erros de digitação e configuração que demandariam tempo adicional de depuração. A automação, portanto, não apenas acelera a implantação inicial, mas também garante a repetibilidade do processo, permitindo que a mesma infraestrutura possa ser recriada de forma idêntica em um tempo previsível.

Análise Qualitativa: Consistência

Além da velocidade, um requisito não funcional crítico para um modelo de automação robusto é a idempotência. Para validar esta característica, os playbooks foram executados uma segunda vez em um ambiente já configurado. Conforme esperado, o Ansible não realizou nenhuma alteração na infraestrutura, reportando o status `ok` para a maioria das tarefas, em vez do status `changed`. Este comportamento confirma que a automação pode ser executada de forma segura para operações de manutenção ou para garantir que a configuração dos nós não tenha se desviado do estado desejado, sem o risco de causar erros ou interrupções.

A consistência da implantação foi validada na etapa seguinte, através da verificação funcional dos serviços, onde se confirmou que todos os nós de computação se registraram corretamente no controlador e que a plataforma estava pronta para o provisionamento de máquinas virtuais. Em conjunto, estes resultados demonstram que o modelo de automação proposto é não apenas eficiente, mas também confiável e robusto.

5.0.2 Validação Funcional da Infraestrutura

Após a conclusão do processo de implantação automatizada, foi realizado um conjunto de testes funcionais para validar o estado operacional da nuvem privada. A validação foi conduzida tanto pela interface gráfica (Horizon) quanto pela linha de comando (CLI), com o objetivo de verificar a integridade dos serviços e a capacidade de provisionamento de máquinas virtuais.

Validação da Integração dos Nós

O primeiro e mais fundamental teste consistiu em verificar se todos os nós de computação foram corretamente registrados pelos serviços do Nó de Controle, confirmando o sucesso da configuração distribuída. A Figura 5.1, uma captura de tela da aba “Hipervisores” do painel de administração do Horizon, demonstra que todos os cinco nós do cluster (`lab100`

a `lab104`) foram detectados e estavam reportando corretamente seu status e a utilização de recursos (RAM e armazenamento local) para o serviço Nova. Este resultado valida o sucesso da comunicação entre o controlador e os nós de computação, uma etapa crítica da automação.

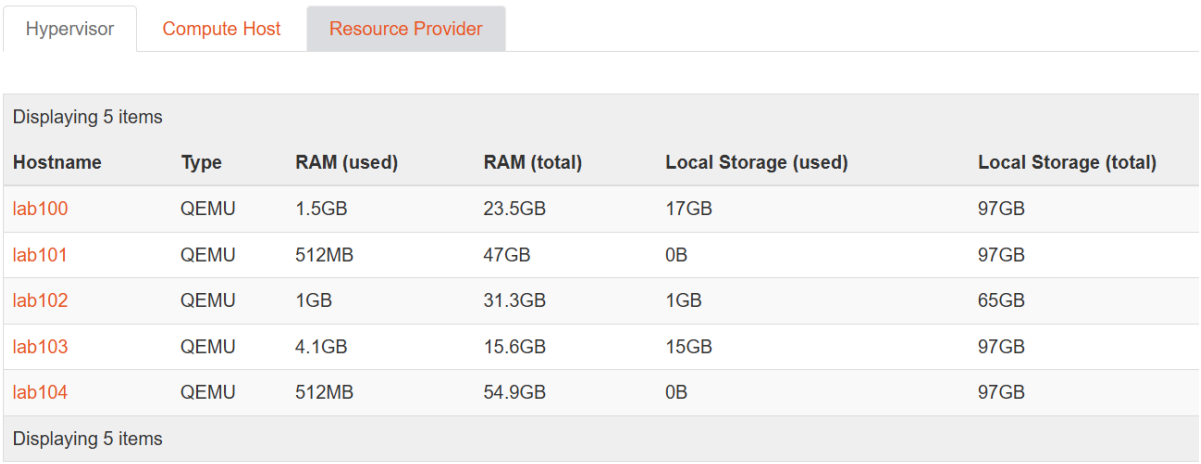


Figura 5.1: Painel de Hipervisores do OpenStack Horizon, mostrando os cinco nós do cluster ativos e gerenciados pelo serviço Nova. Fonte: Elaborado pelo autor (2025).

Validação dos Serviços via CLI e API

Para uma validação mais aprofundada, foram utilizados os clientes de linha de comando do OpenStack. Após carregar as credenciais de administrador com o script `admin-openrc.sh`, comandos como `openstack compute service list` foram executados para verificar o estado (‘up‘) de cada serviço individual.

Para testar a funcionalidade da API de ponta a ponta, uma instância de teste, nomeada `test-cirros`, foi criada. A Figura 4.3 apresenta a saída no terminal do comando `openstack server show test-cirros`. O retorno bem-sucedido desta consulta, exibindo metadados detalhados sobre a instância (como o ID da imagem, flavor, hostID e status), confirma que as APIs dos serviços Keystone, Glance e Nova estavam integradas e se comunicando corretamente.

Validação do Caso de Uso Final

A validação final consistiu em verificar se a nuvem atendia ao seu requisito funcional primário: o provisionamento de recursos para um projeto. A Figura 5.2, capturada da tela de “Visão Geral” de um projeto no Horizon, fornece uma prova visual clara do sucesso deste teste. Os medidores de “Limit Summary” confirmam a alocação de recursos, mostrando o uso de “1 de 10“ instâncias permitidas e “2 de 20“ VCPUs disponíveis. Este resultado

demonstra que um tenant (projeto) é capaz de consumir recursos da nuvem conforme as quotas definidas, validando que o protótipo estava totalmente operacional e pronto para a criação de ambientes para os alunos.

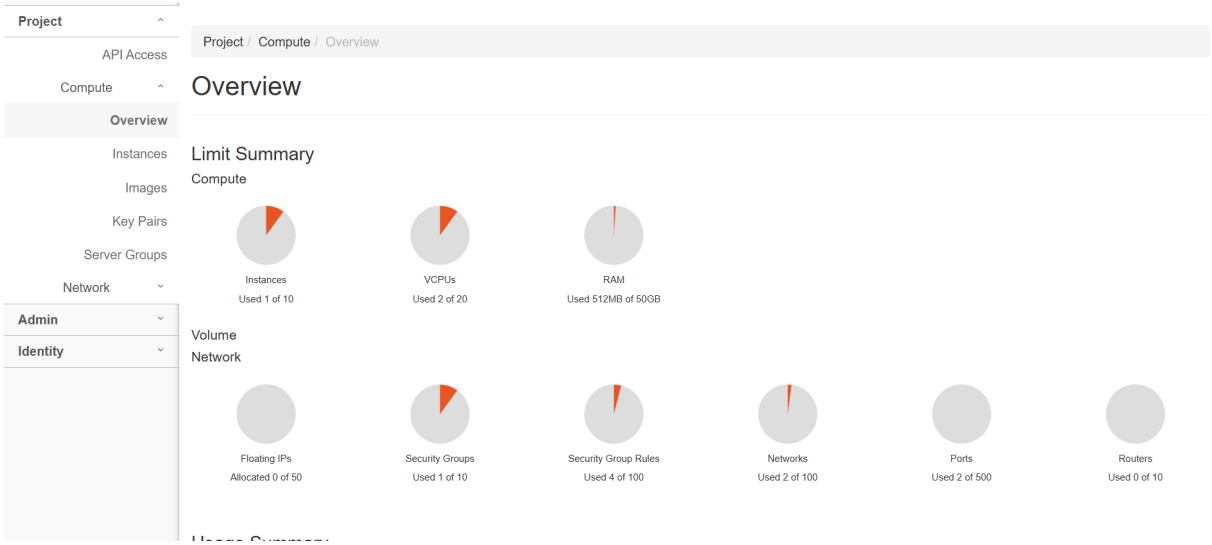


Figura 5.2: Painel de Visão Geral do Projeto no Horizon, evidenciando o consumo de recursos após o provisionamento de uma instância. Fonte: Elaborado pelo autor (2025).

5.0.3 Estudo de Caso: Provisionamento de Ambiente para Simulação

Com a infraestrutura de nuvem validada e funcional, a etapa final do protótipo foi a realização de um estudo de caso para demonstrar a capacidade da solução em atender ao seu requisito principal: prover um ambiente de computação adequado para as simulações e projetos dos alunos do laboratório.

Cenário de Teste e Procedimento

O cenário de teste consistiu em simular o provisionamento de um laboratório virtual para um aluno, criando um ambiente de sistema operacional padrão que pudesse servir de base para a instalação dos softwares de simulação. Como muitos dos programas de engenharia e pesquisa utilizados no ambiente acadêmico são compatíveis ou otimizados para Linux, o sistema operacional escolhido para a validação foi o Ubuntu Server.

O procedimento para a execução do estudo de caso foi o seguinte:

1. **Preparação do Ambiente:** Primeiramente, uma imagem oficial do sistema operacional **Ubuntu Server 22.04 LTS** foi preparada. A imagem, no formato `.qcow2` para otimização com o hipervisor KVM, foi então carregada para o serviço Glance,

tornando-se disponível para todos os projetos na nuvem. Um *flavor* (tipo de instância), nomeado `ubuntu`, foi criado com especificações adequadas (2 vCPUs, 4 GB de RAM, 40 GB de disco).

2. **Provisionamento da Máquina Virtual via CLI:** A validação de ponta a ponta foi realizada utilizando a Interface de Linha de Comando (CLI) do OpenStack. A escolha pela CLI, em vez da interface gráfica Horizon, foi deliberada, pois uma operação bem-sucedida via CLI valida de forma mais fundamental a integridade de todas as APIs do sistema. O comando `openstack server create` foi utilizado para provisionar a instância, conforme demonstrado no vídeo da defesa.

3. **Acesso e Verificação:** Após o provisionamento, o acesso à máquina virtual foi estabelecido via SSH, utilizando o par de chaves previamente configurado no OpenStack.

Resultado e Validação do Caso de Uso

A máquina virtual Ubuntu Server foi provisionada com sucesso em aproximadamente 2 minutos, atingindo o estado *ACTIVE*, como verificado pelo comando `openstack server list`. O acesso via SSH à instância foi bem-sucedido, confirmando que a rede virtual foi configurada corretamente e que a VM estava plenamente operacional.

O sucesso deste estudo de caso valida que o modelo de implantação automatizada proposto resultou em uma infraestrutura de nuvem plenamente capaz de atender aos requisitos funcionais do projeto. Demonstra que, a partir de hardware legado e de um processo de implantação automatizado, é possível criar um ambiente de nuvem robusto e apto a prover os recursos de computação necessários para as atividades de ensino e pesquisa do laboratório, resolvendo o problema original da competição por máquinas físicas.

5.0.4 Discussão dos Resultados e Desafios

A implementação do protótipo e a subsequente validação funcional e de caso de uso confirmam o sucesso do modelo proposto. Os resultados demonstram que é tecnicamente viável, através de um processo de implantação automatizado, transformar um conjunto de hardware legado e heterogêneo em uma infraestrutura de nuvem privada IaaS robusta e funcional. A drástica redução no tempo de implantação, de dias ou horas de trabalho manual para menos de uma hora de execução de scripts, valida a automação como a solução para a barreira de entrada da complexidade do OpenStack.

Apesar do sucesso do protótipo, o processo de implementação revelou desafios práticos que são inerentes a projetos em ambientes reais e não controlados. O principal desafio encontrado foi a heterogeneidade do hardware disponível. O plano inicial de padronizar

todos os nós com arranjos de disco em RAID 6 teve de ser adaptado devido às restrições físicas do servidor `lab102`, que possuía um número limitado de baias de disco. A necessidade de adaptar a configuração de RAID para este nó específico, optando por um RAID 1, evidencia a importância de um modelo de automação flexível, mas também o valor de um levantamento de hardware minucioso na fase de planejamento.

Outro desafio significativo foi a depuração dos templates Jinja2 do Ansible, especialmente para os arquivos de configuração do serviço de rede, Neutron. Garantir a sintaxe correta e a compatibilidade das configurações com a versão Caracal do OpenStack exigiu um ciclo iterativo de testes e ajustes, reforçando que, embora a automação elimine o erro humano na execução, a criação da lógica de automação em si é uma tarefa de engenharia complexa e que demanda conhecimento aprofundado da plataforma.

Finalmente, é importante reconhecer as limitações do presente trabalho. O protótipo focou na automação da implantação inicial, também conhecida como operações de "Dia 1". A automação de tarefas de gerenciamento contínuo, como o monitoramento de desempenho dos nós, a aplicação de atualizações de segurança e o escalonamento dinâmico da infraestrutura (operações de "Dia 2"), não fez parte do escopo desta monografia. Estas áreas representam oportunidades claras e valiosas para trabalhos futuros, que podem se basear na infraestrutura sólida estabelecida por este projeto.

Capítulo 6

Conclusão

Este capítulo finaliza a presente monografia, apresentando as conclusões gerais obtidas com o desenvolvimento do trabalho, reforçando suas contribuições e, em seguida, delineando uma série de possíveis melhorias e direções para pesquisas futuras. O objetivo é consolidar os resultados alcançados e contextualizar o projeto dentro de um ciclo de evolução contínua.

6.1 Conclusões

O objetivo principal desta monografia foi propor e validar um modelo prático para a implantação automatizada de uma nuvem privada IaaS baseada em OpenStack, com foco no reaproveitamento de hardware legado em um ambiente acadêmico. A análise da literatura confirmou a existência de um paradoxo no estado da arte: enquanto o valor de nuvens privadas para o ensino e a pesquisa é amplamente reconhecido, a alta complexidade da implantação manual do OpenStack representa uma barreira significativa para muitas instituições.

A solução proposta, fundamentada em um modelo de automação com Ansible, demonstrou ser eficaz para superar essa barreira. O protótipo implementado em um cluster de cinco servidores revitalizados validou que é possível construir uma nuvem funcional de forma rápida, consistente e repetível. Os resultados, que incluem um tempo de implantação de aproximadamente 45 minutos e a validação funcional completa da plataforma, confirmam que os objetivos do trabalho foram atingidos com sucesso.

As principais contribuições deste trabalho para a área são:

- A investigação sistemática e a documentação dos desafios da implantação manual de uma nuvem OpenStack em um ambiente com múltiplos nós e hardware heterogêneo.

- O desenvolvimento de um conjunto de *playbooks* Ansible que orquestra a instalação e a configuração dos principais serviços do OpenStack, servindo como um modelo prático e customizável.
- A validação do modelo através da implantação de um protótipo funcional sobre hardware legado, demonstrando a drástica redução no tempo e no esforço manual.
- A apresentação de um modelo documentado e replicável que pode ser adaptado por outras instituições, efetivamente reduzindo a barreira de entrada para a adoção de soluções de nuvem privada.

Conclui-se, portanto, que a automação da implantação é um passo fundamental para tornar os benefícios da Computação em Nuvem mais acessíveis a ambientes com recursos limitados. A solução apresentada não apenas resolve um problema técnico, mas também habilita novas possibilidades de ensino e pesquisa ao prover uma infraestrutura robusta de forma eficiente e de baixo custo.

6.2 Trabalhos Futuros

Embora o protótipo seja funcional, ele representa a fundação sobre a qual melhorias significativas podem ser construídas. As principais direções para a evolução do projeto, focadas nas operações de "Dia 2"e na robustez do ambiente, são detalhadas a seguir:

- **Implementação de Alta Disponibilidade (HA):** A melhoria mais crítica seria a implementação de um cluster de Nós de Controle para eliminar o ponto único de falha, utilizando ferramentas como Pacemaker e HAProxy para garantir a continuidade dos serviços.
- **Evolução da Infraestrutura de Armazenamento:** A adoção de uma solução de armazenamento distribuído, como o Ceph, permitiria a criação de um pool de armazenamento centralizado e resiliente, habilitando funcionalidades de produção como a migração de VMs ao vivo.
- **Automação de Operações de "Dia 2":** Expandir o escopo da automação para incluir o gerenciamento contínuo da nuvem. Isso envolveria o desenvolvimento de playbooks para a implementação de um stack de monitoramento (com Prometheus e Grafana) e para a automação de processos de atualização e aplicação de patches de segurança.
- **Automação dos Testes de Validação:** Um trabalho futuro de grande valor seria o desenvolvimento de um playbook Ansible dedicado à validação. Este playbook

seria executado após a implantação e realizaria automaticamente os testes funcionais, como a criação de uma rede de teste, o provisionamento de uma VM CirrOS, a verificação de sua conectividade e, por fim, a exclusão de todos os recursos, gerando um relatório final e criando um ciclo de implantação e validação totalmente automatizado.

Referências

[1] Mell, Peter e Timothy Grance: *The nist definition of cloud computing*. Special publication 800-145, National Institute of Standards and Technology, 2011. viii, 2, 7, 8

[2] Takahashi, Kimitoshi: *A Study on Portable Load Balancer for Container Clusters*. Tese de Doutoramento, setembro 2019. viii, 9

[3] OpenStack Foundation: *Openstack kilo conceptual architecture*. Online, 2015. Disponível em: `https://docs.openstack.org/kilo/install-guide/common/environment-conceptual-arch.html`. viii, 10

[4] Carissimi, Alexandre: *Virtualização: da teoria a soluções*. Em *XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Livro Texto dos Minicursos*, páginas 174–206, Porto Alegre, RS, Brasil, 2008. Sociedade Brasileira de Computação. 3, 8, 9

[5] Sahasrabudhe, Shalmali Suhas e Shilpa S. Sonawani: *Comparing openstack and vmware*. Em *2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC)*. IEEE, 2014. 3, 9

[6] Sefraoui, Omar, Mohammed Aissaoui e Mohsine Eleuldj: *Openstack: Toward an open-source solution for cloud computing*. International Journal of Computer Applications, 55(3):38–42, 2012. 3, 4, 8, 9, 19

[7] Arora, Deepak, Varun Kumar, B. Jagdeep e Prabhat Verma: *Proposed model for virtual labs interaction with openstack integration using kvm hypervisor*. International Journal of Scientific Technology Research, 3(7):64–68, 2014. 3, 4, 15, 16, 19

[8] Bhatia, Gaurav, Ismail Al Noutaki, Sheikha Al Ruzeiqi e Jawaher Al Maskari: *Design and implementation of private cloud for higher education using openstack*. Em *2018 Majan International Conference (MIC)*, páginas 1–6. IEEE, 2018. 3

[9] Kabiri, Mohammad Nazim e Muhammad Wannous: *An experimental evaluation of a cloud-based virtual computer laboratory using openstack*. Em *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, páginas 667–672, 2017. 3, 5, 16, 19, 20

[10] Pereira, André Igor, Jonas Henrique Gusmão de Oliveira, Ubiratam de Paula e Rafaelli Coutinho: *Implantação de uma infraestrutura de nuvem usando openstack*. Em *Anais de Evento (não especificado)*, 2018. 3, 14, 15, 19

[11] Kumar, Rakesh, Neha Gupta, Shilpi Charu, Kanishk Jain e Sunil Kumar Jangir: *Open source solution for cloud computing platform using openstack*. International Journal of Computer Science and Mobile Computing, 3(5):89–98, 2014. 4, 12, 14, 20, 38

[12] Wen, Xiaolong, Genqiang Gu, Qingchun Li, Yun Gao e Xuejie Zhang: *Comparison of open-source cloud management platforms: Openstack and opennebula*. Em *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, páginas 2457–2461, 2012. 4, 11, 19

[13] Jackson, Kevin e Cody Bunch: *OpenStack Cloud Computing Cookbook*. Packt Publishing, Birmingham, UK, second edição, 2013. 10, 12, 13, 20

[14] Albaroodi, Hala, Selvakumar Manickam e Parminder Singh: *Critical review of openstack security: Issues and weaknesses*. Journal of Computer Science, 10(1):23–33, 2014. 13, 14, 20

[15] Zhao, Linchang, Guoqing Hu e Yongchi Xu: *Educational resource private cloud platform based on openstack*. Computers, 13(9):241, 2024. 17

[16] Younge, Andrew J., Robert Henschel, James T. Brown, Gregor von Laszewski, Judy Qiu e Geoffrey C. Fox: *Analysis of virtualization technologies for high performance computing environments*. Em *2011 IEEE 4th International Conference on Cloud Computing*, páginas 9–16, 2011. 17, 18

[17] Kominos, Charalampos Gavriil, Nicolas Seyvet e Konstantinos Vandikas: *Bare-metal, virtual machines and containers in openstack*. Em *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, páginas 36–43, 2017. 17, 18

[18] Bystrov, Oleg, Ruslan Pacevič e Arnas Kačeniauskas: *Performance of communication- and computation-intensive saas on the openstack cloud*. Applied Sciences, 11(16):7379, 2021. 18, 19