UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Departamento de Computação

MATHEUS TEIXEIRA MATTIOLI

Estudo de ferramentas de automação para computação em nuvem

SÃO CARLOS-SP

2023

MATHEUS TEIXEIRA MATTIOLI

Estudo de ferramentas de automação para computação em nuvem

Trabalho de conclusão de curso apresentado ao Departamento de Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientação Prof. Dr. Hélio Crestana Guardia

SÃO CARLOS-SP

2023

Dedico este trabalho a todos os professores que participaram da minha formação e a minha família, que sempre me apoiou em todas as minhas escolhas.

"A teoria por trás do código aberto é simples. No caso de um sistema operacional, o código-fonte – as instruções de programação subjacentes ao sistema – é gratuito. Qualquer um pode melhorá-lo, alterá-lo, explorá-lo. Mas essas melhorias, mudanças e explorações devem ser disponibilizadas gratuitamente. Pense Zen. O projeto não pertence a ninguém, é de todos. Quando um projeto é aberto, há uma melhoria rápida e contínua. Com equipes de colaboradores trabalhando em paralelo, os resultados podem acontecer com muito mais rapidez e sucesso do que se o trabalho estivesse sendo realizado a portas fechadas.

(Linus Torvalds)

Resumo

A computação surgiu para facilitar as ações da vida humana, desde a criação de planilhas eletrônicas até a facilitação no acesso à informação. Com o modelo de computação em nuvem, ela oferece soluções como serviços muitas vezes virtuais para facilitar o acesso de empresas ou usuários a armazenamento remoto persistente ou até mesmo infraestrutura. Por exemplo, pode-se considerar um local remoto para armazenamento de arquivos, ou, uma infraestrutura distante que pode ser utilizada em alguma aplicação virtual. Visando construir uma estrutura para oferecer computação em nuvem, este trabalho propõe estudar e implementar técnicas de automação para ambientes em nuvem. Fazendo uso de técnicas como infraestrutura como código e metal como serviço o objetivo é automatizar essa implementação para se tornar rápida e escalável, algo que é imprescindível no mundo contemporâneo. Como resultado é obtido um estudo sobre o uso das principais ferramentas disponíveis como software livre.

Palavras-chave: Computação em Nuvem; Infraestrutura como Código; IaC; Heat; Juju; Puppet; IaaS; Chef; Nuvem; OpenStack; MaaS.

Abstract

Computing emerged to facilitate the actions of human life, from the creation of electronic spreadsheets to information access. Bringing it to cloud computing, it offers solutions such as virtual services to facilitate access by companies or users to persistent remote storage or even infrastructure. For example, one can consider a remote location for storing files, or a distant infrastructure that can be used in some virtual application. Aiming to build a structure that offers cloud computing, this work proposes studying and implementing automation techniques for cloud environments. Making use of techniques such as infrastructure as code and metal as a service, the objective is automating this implementation, making it fast and scalable, something that is essential in the modern world. As a result, we present a study on the use of the main open source tools avaiable as open software.

Keywords: Cloud Computing; Infrastructure as Code; Iac; Heat; Juju; Puppet; IaaS; Chef; Cloud; OpenStack; MaaS.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A computação em nuvem é uma área extremamente abrangente e complexa de se definir sucintamente. No entanto, uma de suas principais funcionalidades é oferecer soluções para aplicações com necessidades variáveis de recursos computacionais por meio de servidores, protocolos de comunicação em rede e infraestrutura, como serviços, o que é conhecido como IaaS (*Infrastructure as a Service* - Infraestrutura como Serviço). Sendo assim, é um dos modelos fundamentais de serviços de computação em nuvem, onde os provedores fornecem recursos de infraestrutura virtualizados através da internet. Isso inclui servidores, armazenamento, redes e outros componentes como *block storage*, permitindo que os usuários aluguem esses recursos de forma flexível, escalonável e sob demanda. Em vez de investir em hardware próprio, os clientes podem provisionar e gerenciar serviços de infraestrutura conforme suas necessidades, pagando apenas pelo uso que fizerem. Isso oferece maior agilidade, redução de custos e escalabilidade para empresas e desenvolvedores que buscam executar suas aplicações de forma eficiente na nuvem. Porém, caso contratar um serviço IaaS seja um problema, há a opção de investir na construção de sua própria nuvem utilizando uma tecnologia de código aberto e gratuita conhecida como OpenStack [1]. Essa plataforma é responsável por implementar nuvens IaaS, oferecendo flexibilidade e controle sobre a infraestrutura de computação em nuvem interna da empresa.

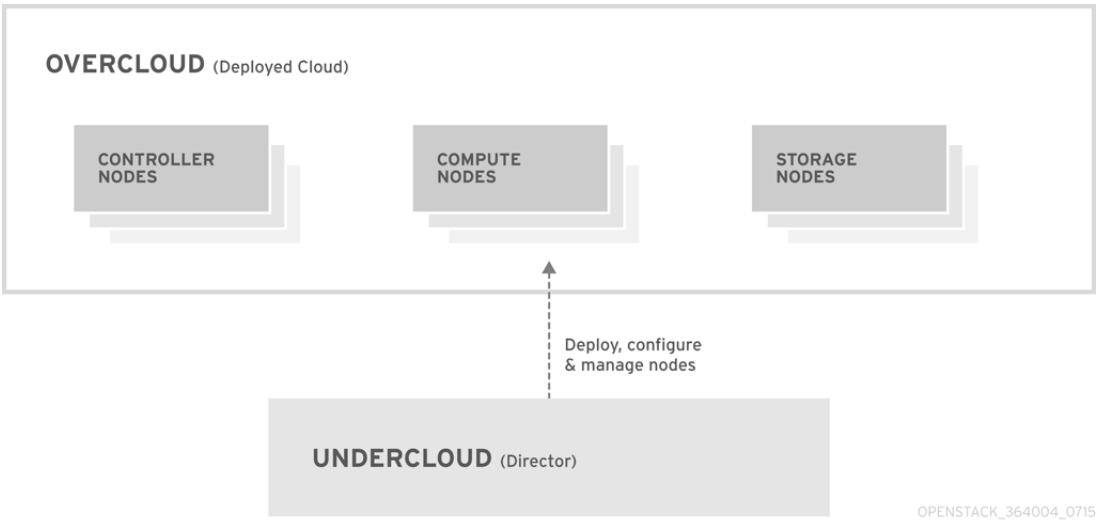
Muitas arquiteturas para construção de nuvem interna ou privada podem ser pensadas com relação à organização das máquinas físicas, por exemplo, uma única máquina física, rodando máquinas virtuais provendo serviços específicos de IaaS. A Figura 1 mostra um esquemático de uma configuração com quatro máquinas (nós), das quais, o *compute node* (nó de computação) é responsável por fornecer recursos de servidores, o *storage node* (nó de armazenamento) provê serviços de armazenamento como *block storage* e *object storage* e o *controller node* (nó controlador) responsável por gerenciar e monitorar consumo dos recursos da infraestrutura.

O artigo de Yamato (2017) aborda a implementação de uma IaaS utilizando arquiteturas *bare metal*, onde cada servidor é um hardware físico independente. Neste caso, argumenta-se que isso oferece vantagens significativas, principalmente para cargas de trabalho que exigem alto desempenho, baixa latência e recursos dedicados, por conta da ausência das camadas de virtualização. Nessa configuração, cada máquina pode assumir o papel dos nós representados na Figura 1. De acordo com o artigo *"Cloud Computing: An Overview"* de Qian et al. (2009), uma configuração comum para os servidores é designá-los

[1] https://www.redhat.com/pt-br/topics/openstack

[2] Disponível em: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.0/html-single/director_installation_and_usage/index#sect-Overcloud. Acesso em 30 Jul. 2023

Figura 1 – Exemplo de organização das máquinas físicas para construção de uma nuvem.



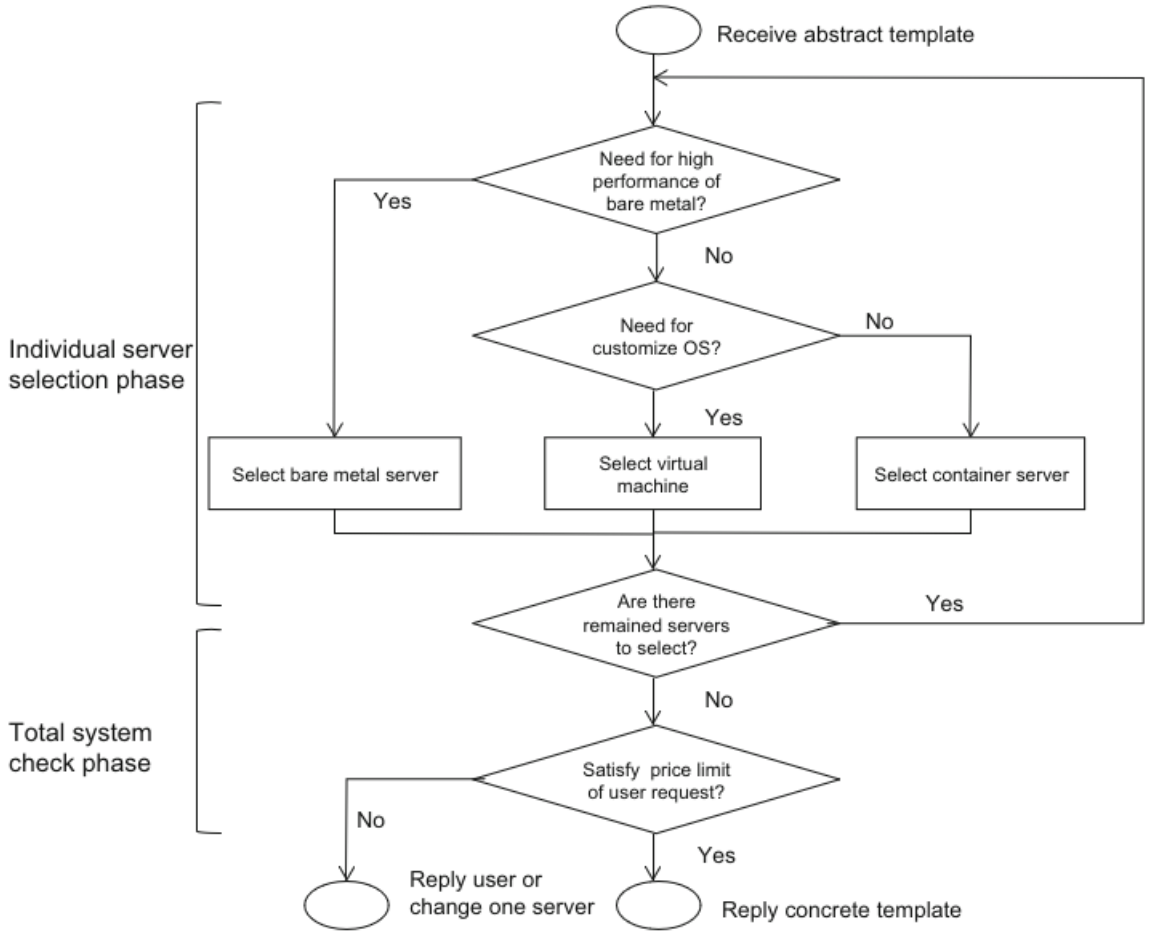
RedHat OpenStack documentation [2]

para tarefas específicas, como aplicação, plataforma e recursos. Por exemplo, em uma aplicação *web* com três camadas - banco de dados, aplicação e apresentação - cada servidor pode ser atribuído a uma camada. Dessa forma, o servidor de recursos seria responsável por fornecer serviços de acesso ao banco de dados, o servidor de aplicação teria os *softwares* necessários para implementar as regras de negócio e o servidor de plataforma cuidaria da apresentação da aplicação *web*.

Portanto, para implementação de infraestrutura como serviço, existem diferentes formas de utilizar os recursos dos servidores. As principais são através da virtualização, que envolve a criação de várias máquinas virtuais para otimizar o aproveitamento do hardware de uma única máquina, e a técnica conhecida como *bare metal*, na qual não há virtualização e todo o poder da máquina é direcionado para um único propósito específico. Essas abordagens oferecem vantagens distintas e são escolhidas de acordo com as necessidades e requisitos específicos de cada aplicação ou projeto. A Figura 2 apresenta um fluxograma das diferentes possibilidades a serem consideradas no planejamento da construção de uma nuvem IaaS ou Maas (*Metal as a Service* - Metal como serviço).

Embora a computação em nuvem ofereça inúmeras vantagens, é essencial realizar um planejamento adequado antes de implementar uma solução para utilização de serviços em nuvem. Por exemplo, ao optar por IaaS, muitas vezes é necessário recriar e reconfigurar ambientes virtuais por conta de questões de escalabilidade do serviço que está sendo implementado nessa infraestrutura de software. Esse processo exige a execução de uma quantidade considerável de comandos em um terminal, a fim de configurar novamente cada máquina virtual com os serviços adequados para a aplicação desenvolvida. Como resultado, um tempo valioso é desperdiçado no lançamento da aplicação, aumentando o risco de erros humanos ao digitar tantos comandos necessários para criar a solução em

Figura 2 – Fluxograma para escolha de arquitetura da nuvem.



Fonte: Yamato (2017).

uma nuvem IaaS.

O mesmo problema de escalabilidade vale para uma nuvem MaaS. Por exemplo, imagine que uma empresa queira aumentar sua operação de 10 máquinas físicas para 100. Uma nuvem desse estilo permite, por exemplo, instalar sistemas operacionais automaticamente em todas essas máquinas a partir de uma já configurada.

Dado este contexto, destaca-se a importância de uma técnica fundamental para garantir integração e desenvolvimento contínuo, conhecida como IaC (*Infrastructure as Code* - Infraestrutura como Código), aplicável em nuvens que oferecem IaaS. A abordagem de IaC envolve a construção da infraestrutura de software para uma aplicação por meio de arquivos de códigos, nos quais, dependendo da tecnologia utilizada, a sintaxe pode ser similar à YAML, Ruby, Golang, entre outras linguagens. Ao empregar esses arquivos de código, é possível criar essas infraestruturas (máquinas virtuais) ou *stacks* de forma imediata, e também replicá-las facilmente, bastando apenas a reexecução do arquivo. Isso significa que a configuração de toda a infraestrutura é tratada como código, o que oferece diversos benefícios. Ao invés de depender de procedimentos manuais propensos a erros e demorados, o uso da IaC permite automatizar todo o processo de implantação da

infraestrutura, tornando-o mais confiável, consistente e ágil.

Essa abordagem também facilita a colaboração entre equipes de desenvolvimento, operações e outros setores, já que o código da infraestrutura é versionado, revisado e mantido no mesmo sistema de controle de código-fonte utilizado pelo software da aplicação, conforme indica o artigo de Romero et al. (2022). Isso proporciona um ambiente coeso, possibilitando um gerenciamento mais eficiente e uma maior compreensão do estado da infraestrutura das máquinas virtuais em cada etapa do desenvolvimento e implantação. Portanto, com a IaC, mudanças podem ser implementadas de forma controlada, documentada e auditável, contribuindo para a redução de falhas e na facilitação de práticas de DevOps, onde o desenvolvimento e a operação de sistemas são integrados para melhorar a agilidade e a qualidade do produto final. Sendo assim, a IaC é capaz de garantir uma base sólida para a execução de serviços na nuvem com eficiência, consistência e escalabilidade.

Para nuvens MaaS, existe uma tecnologia de mesmo nome, desenvolvida pela empresa Canonical responsável por, interativamente em sua interface, ou através de comandos na CLI, promover automação no fornecimento das máquinas físicas.

Soluções em nuvem podem ser caras para empresas, dessa forma, implementar sua própria infraestrutura pode ser o caminho. Porém, isto não é uma tarefa fácil, pois existe todo um trabalho de inventariação dos equipamentos, planejamento, configuração e o uso de alguma ferramenta de automação. Este último é um problema, pois, apesar das vantagens que a automação traz para a computação em nuvem, também existem algumas desvantagens. Por exemplo, de acordo com Artac et al. (2018), há dificuldades em escrever e compreender o código utilizado em técnicas IaC. Além disso, Guerriero et al. (2019) afirmam que a diversidade de tecnologias para orquestração de IaaS e MaaS pode ser um desafio na indústria. E Edwards (2022) aponta algumas das ferramentas mais populares de IaC na indústria. Então, dentre essas várias tecnologias disponíveis para a criação de infraestruturas em nuvem, seja na etapa através de código, no provisionamento das máquinas físicas ou até documentação do inventário, destacam-se:

- NetBox é uma plataforma de código aberto usada para gerenciamento de infraestrutura de rede. Esta plataforma fornece recursos para rastrear, documentar e automatizar o gerenciamento de ativos de rede, como dispositivos, endereços IP e cabos, tornando-o uma ferramenta valiosa para administradores de rede e data centers.

- Heat é um projeto da plataforma OpenStack que oferece recursos para criação e gerenciamento de recursos da IaaS. Heat permite a definição de modelos (*templates)* que descrevem os componentes da infraestrutura virtual, como instâncias de máquinas virtuais, redes virtuais, volumes de armazenamento e muito mais.

- Juju é uma ferramenta desenvolvida pela Canonical que possibilita a modelagem,

configuração e implantação de serviços em nuvem de forma simples e rápida. Esta ferramenta utiliza conceitos de *charms* que representam unidades de software e suas dependências, permitindo a criação de ambientes complexos com facilidade. Atuando como um mecanismo de automação na disponibilização das nuvens IaaS como o OpenStack.

- Terraform é uma ferramenta da HashiCorp que permite a criação, mudança e versionamento da infraestrutura de software para aplicações como código, IaC. Terraform suporta diversas provedoras de nuvem, como AWS, Azure, Google Cloud, entre outras, tornando-o uma escolha versátil para gerenciar recursos em ambientes *multicloud*.

- O próprio Metal as a Service (MaaS), também da Canonical, é uma tecnologia importante para automação, que permite o gerenciamento e a implantação automatizada de servidores físicos, também conhecidos como "*bare metal servers*". A principal função dessa ferramenta é tornar mais eficiente e ágil o processo de provisionamento e gerenciamento de hardware em data centers ou ambientes de computação em nuvem. O funcionamento da ferramenta MaaS envolve vários componentes que trabalham em conjunto para automatizar as tarefas de implantação e configuração dos servidores físicos.

- *Puppet*, construído pela Puppet Inc, é uma ferramenta IaC popular que permite a automação e o gerenciamento de configurações de máquinas virtuais de forma declarativa, o que significa que os usuários descrevem o estado desejado do sistema em vez de detalhar o processo para chegar a esse estado. Uma tecnologia desse último tipo citado, conhecido como paradigma imperativo, é Heat. Na ferramenta Puppet há uma linguagem específica para descrever as configurações da infraestrutura, baseada em uma sintaxe simples e fácil de entender, permitindo que os usuários expressem suas intenções de forma clara e concisa e reutilizem código através da modularização disponível.

- A tecnologia *Chef*, possibilita a automação e o gerenciamento de configurações de infraestrutura de forma declarativa e programática, por meio de *recipes* e *cookbooks*. Permitindo que os administradores descrevam o estado desejado da infraestrutura em um código, e, fazendo analogia à receitas de cozinha, Chef se encarrega de aplicar e manter essa configuração de maneira consistente em todas as máquinas virtuais do ambiente da aplicação.

Todas essas tecnologias são de código aberto (*open source*) e, vale ressaltar, que mesmo facilitando a implementação de ambientes de computação em nuvem usando infraestrutura como código ou promovendo outras automações, é importante reconhecer que cada uma

possui suas próprias peculiaridades de instalação e recursos providos, cada uma podendo atuar em fases específicas da implantação da arquitetura da nuvem. A escolha da ferramenta mais adequada dependerá de diversos fatores, tais como as necessidades específicas do projeto, as preferências da equipe de desenvolvimento e as características do ambiente de nuvem em que será implantada a solução.

Como foi visto, existem muitas decisões para serem tomadas na construção de uma solução de nuvem IaaS. Neste caso, pode-se definir as etapas:

- Organização/arquitetura das máquinas físicas, parte em que são planejadas as configurações de *networking* nos *switches* e computadores adquiridos;
- Implantação de uma infraestrutura de software em um conjunto de nós, isto é, configuração e instalação dessas máquinas com sistemas operacionais e dependências para o fornecimento da nuvem;
- Implantação da infraestrutura de software para uma aplicação, onde IaaS já está pronta e os serviços podem ser utilizados para estruturar as aplicações da empresa em máquinas virtuais.

Dessa forma, considerando a ampla variedade de ferramentas de automação disponíveis, este trabalho apresenta um estudo sobre a construção de uma nuvem IaaS utilizando algumas dessas ferramentas para automatizar as etapas de implantação de uma infraestrutura de software em um conjunto de nós e implantação da infraestrutura de software para uma aplicação. O foco foi direcionado a verificar e analisar todo o processo de preparação das máquinas físicas, para, a partir delas, configurar uma nuvem oferecendo IaaS, por meio das tecnologias MaaS, Juju e Heat.

O objetivo principal dessa pesquisa é avaliar e julgar quais ferramentas de IaC e automação são mais pertinentes e adequadas para utilização no projeto de extensão "Pesquisa e desenvolvimento em tecnologias para data centers utilizando virtualização" em parceria DC/UFSCar e LuizaLabs. Por meio desse estudo, busca-se identificar as tecnologias mais eficientes, funcionais e alinhadas às necessidades específicas do projeto, a fim de garantir uma infraestrutura ágil, confiável e escalável para a realização dos objetivos propostos no âmbito do projeto de extensão.

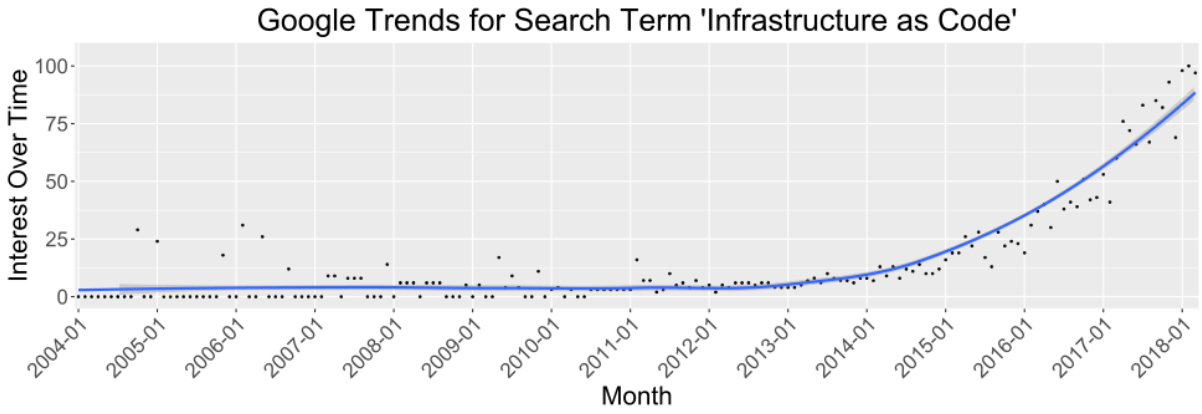
Ao final, é obtido como resultado um relatório apresentando a etapa de atuação de cada tecnologia na construção da nuvem. Também são construídas tabelas comparativas entre cada tecnologia, indicando a funcionalidade, os recursos, características relevantes e etapa de atuação na construção da infraestrutura das ferramentas MaaS, JuJu, Heat, Chef e Puppet.

2 Revisão Bibliográfica

O mundo contemporâneo demanda agilidade nas mais diversas esferas do trabalho humano, mas desde a produção agrícola até a científica, muita dessa agilidade só é possível por conta do nível de automação proporcionado por avanços tecnológicos. Portanto, no ramo da tecnologia a velocidade das aplicações desenvolvidas deve ser sempre um fator levado em consideração. Nesse sentido, trazendo para o campo da computação em nuvem, a capacidade de automatizar a construção e configuração da nuvem atualmente é crucial para as empresas.

As ferramentas de IaC [1] [2] e MaaS [3] permitem trazer para esse campo automação, padronização e versionamento da infraestrutura demandada para implementar métodos ágeis tão importantes na contemporaneidade. Em outras palavras, IaC é um caminho muito importante para adotar as políticas de CI/CD (*Continuous Integration/Continuous Development* - Integração Contínua/Desenvolvimento Contínuo) e DevOps (ARTAC et al., 2017). Por conta disso, muitas pesquisas estão surgindo recentemente neste assunto relativamente novo; a Figura 3 comprova essa afirmação, na qual é possível verificar o crescimento do interesse através de pesquisas no Google com o passar dos anos.

Figura 3 – Crescimento do interesse em IaC ao longo dos anos.



Fonte: Rahman, Mahdavi-Hezaveh e Williams (2019).

Também é possível encontrar pesquisas de mapeamento sistemático e revisão sistemática da literatura, o que é muito importante para destacar a importância científica desse campo. Então, a partir das *strings* de busca

- "(IaC OR Infrastrucutre as Code) AND (IaaS OR Cloud Computing OR DevOps)"

[1] https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac
[2] https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code
[3] https://maas.io/how-it-works

- "(IaaS OR Bare Metal) AND (Architecture OR Scheme OR Configuration)"

foram encontrados os mapeamentos sistemáticos de Rahman, Mahdavi-Hezaveh e Williams (2019) e Kumara et al. (2021). Os demais trabalhos escolhidos para serem apresentados foram obtidos aplicando as técnicas de métodologia científica *forward snowballing* e *backward snowballing* nos dois artigos separados anteriormente.

No geral, adiantando que após a análise de toda a bibliografia encontrada, pode-se concluir que métodos de automação em nuvem, seja ele IaC ou MaaS, são um tópico quente e estado da arte para empresas que se preocupam com seus servidores. Isto decorre do fato de proverem implementação de métodos ágeis e habilitam práticas de monitoramento, teste e balanceamento de carga nesses sistemas. Também, por conta de ser uma área nova e emergente, a quantidade e variedade de pesquisas que podem ser feitas são variadas e algumas nunca exploradas, como a capacitação de pessoas na área. A maior parte dos estudos encontrados é de natureza comparativa, ou abordando a implementação de ferramentas já existentes ou promovendo a criação de novas.

2.1 Trabalhos Relacionados

Nesta subseção são apresentados os principais trabalhos relacionados encontrados resultantes da aplicação das técnicas citadas. Para começar, é importante frisar a importância do termo *"as code"* do nome IaC, pois, ao escrever uma infraestrutura em forma de código é possível abordar todas as automações citadas no início dessa revisão bibliográfica. Nessa linha de raciocínio é importante citar o trabalho de Parnin et al. (2017) para relevar a importância dessa prática. Intitulado de *"The Top 10 Adages in Continuous Deployment"*, o artigo foi construído a partir de um encontro de verão entre pesquisadores de engenharia de software e ele conta sobre as principais políticas adotadas e almejadas por grandes empresas para proporcionar CI/CD e os principais problemas encontrados, dentre esses problemas, um que é muito relevante é o seguinte:

> Empresas rapidamente superam a infraestrutura para armazenar dados relacionados a métricas. A Netflix inicialmente coletava 1,2 milhão de métricas relacionadas aos seus serviços de streaming, mas isso logo aumentou para 1 bilhão de métricas. Não apenas o armazenamento de dados em memória não conseguia mais acompanhar, mas a empresa também teve que considerar mais cuidadosamente quais dados eram essenciais para experimentação. Um dos participantes comentou: "Você precisa de um doutorado para escrever uma consulta de análise de dados."É necessário um investimento significativo tanto em telemetria quanto em análise. Investir nesses esforços separadamente pode ser caro. (PARNIN et al., 2017, tradução nossa)

Fica claro que isso é um problema de escalabilidade da infraestrutura para adequar a grande quantidade de dados obtida a partir do crescimento da empresa, como o caso

da Netflix. Uma solução para isso, segundo o artigo, é implementar uma política para automatizar a atualização da infraestrutura chamado de *deployment strategy*, logo, pode-se perceber que a solução seria utilizar IaC.

Por conta da área ter uma grande quantidade de tecnologias existem alguns estudos comparativos na literatura, como apresentado em Jiang e Adams (2015), o qual faz um estudo empírico muito importante aonde mais de 265 projetos OpenStack foram analisados, revelando que os arquivos de especificação de infraestrutura são grandes e mudam frequentemente, de forma que isso é um potencial vetor para introdução de bugs. Além disso, o estudo aponta que os arquivos de IaC estão fortemente acoplados com outros arquivos do projeto, especialmente arquivos de teste, o que aumenta a complexidade e o risco de introduzir problemas no ambiente de teste ou implantação do sistema de software.

O artigo dos pesquisadores Carvalho e Araujo (2020) faz uma comparação entre Cloudify, Heat, CloudFormation, Terraform e Cloud Assembly como orquestradores de *"Multiclouds"*, apontando o melhor para este cenário de uso. Os artigos de Scheuner et al. (2014) e Scheuner et al. (2015) produzem uma ferramenta de *benchmarking* entre ambientes IaaS baseada em IaC para performar essa análise, seu nome é *Cloud WorkBench*. E Miglierina (2014) propõe uma revisão das principais tecnologias IaC disponíveis no mercado.

Agora, em relação às ferramentas mais utilizadas, o mapeamento sistemático de Rahman, Mahdavi-Hezaveh e Williams (2019) aponta como as mais proeminentes Chef e Puppet, que são mencionados em muitas pesquisas recentes. Fato que é comprovado na tabela da Figura 4, a qual apresenta o número de vezes que determinada ferramenta aparece em publicações de um determinado ano, dentre os artigos selecionados pelo mapeamento. Ainda nesta figura, é importante destacar a presença da tecnologia Juju e a ausência de Heat. Essa ausência pode ser explicada pelo fato dessa tecnologia poder ser integrada e até substituída facilmente tanto pelo Chef, quanto pelo Puppet.

Como vimos, em geral, escrever uma infraestrutura como código é vantajoso para a engenharia de software, porém alguns artigos vão além. Segundo Artac et al. (2017), IaC pode ser considerado um campo de estudo em DevOps. Já Wettinger, Breitenbücher e Leymann (2014) criaram uma linguagem chamada de *"DevOpSlang"*. Por fim, o artigo *"Don't Install Software by Hand"* de Spinellis (2012) diz que:

Figura 4 – Quantidade de vezes em que a tecnologia é utilizada em artigos por ano.

Tool	2012	2013	2014	2015	2016	2017
ABS Modeling Language	0	0	0	1	0	0
Ansible	0	0	0	1	0	0
Argon	0	0	0	0	0	2
Charon	0	1	0	0	0	0
Chef	0	2	3	3	0	1
ConfigValidLang	0	0	0	0	0	1
DevOpsLang	0	0	1	0	0	0
Foreman	0	0	0	0	1	0
Juju	0	0	1	2	0	0
Omnia	0	0	0	0	0	1
Puppet	0	0	0	2	3	1
Vagrant	0	0	1	0	0	0

Fonte: Rahman, Mahdavi-Hezaveh e Williams (2019).

> Um importante facilitador do DevOps são as ferramentas de gestão de configuração de sistemas de TI. Tais ferramentas permitem controlar e automatizar a configuração de todos os elementos que compõem um sistema de TI: hosts, software instalado, usuários, serviços em execução, arquivos de configuração, tarefas agendadas, redes, armazenamento, monitoramento e segurança. (Note que, embora sejam comumente chamadas de ferramentas de gestão de configuração, elas não estão relacionadas às ferramentas de gestão de configuração de software, como Subversion e Git, que usamos para gerenciar revisões de nosso software.) [...] Sua principal função é automatizar a configuração de um sistema. Você escreve algumas regras que expressam como um sistema de TI deve ser configurado, e a ferramenta configurará o sistema de acordo com essas regras. (SPINELLIS, 2012, tradução nossa)

Alguns autores foram além e propuseram *frameworks* e ferramentas. A saber, Hanappi, Hummer e Dustdar (2016) trabalham um problema muito comum nessa área chamado de idempotência, que é a prática de reaplicar múltiplas vezes o mesmo script e resultar sempre no mesmo estado ou efeito, independentemente do número de vezes que é executado, garantindo que as operações possam ser repetidas sem causar efeitos colaterais indesejados ou duplicação de recursos. Por exemplo, se uma solicitação de criação de uma instância de máquina virtual for idempotente, não importa quantas vezes a solicitação seja enviada; a instância será criada apenas uma vez, evitando a duplicação não intencional de recursos. Para isso, eles apresentam uma *frameworks* conceitual aplicável nos códigos Puppet.

Em uma empresa, pode ocorrer a mudança do serviço que implementa IaC por algum motivo e, segundo Sandobalin, Insfran e Abrahao (2017b), isso acaba sendo um

desafio técnico importante para apoiar as ativades de CI/CD. Por conta disso, esses mesmos autores lançaram mão de uma ferramenta responsável por traduzir o estado final produzido pelos códigos em *scripts* que trabalham com a plataforma Amazon Web Services (AWS).

Construir uma infraestrutura por meio de *scripts* e *frameworks* automatizados também é um campo de pesquisa muito interessante e útil nesta área. Os artigos de Singh et al. (2015), Hintsch, Görling e Turowski (2015), Aiftimiei et al. (2017) e Sandobalin, Insfran e Abrahao (2017a) tratam desse assunto por meio de tecnologias já estabelecidas no mercado ou criadas para a pesquisa. Dando destaque para o primeiro citado, ele conta sobre a construção de uma nuvem oferecendo IaaS, propondo um *design* para a arquitetura e explicando a utilização da ferramenta Ansible neste contexto.

> [...] Para esse propósito, o Ansible é utilizado como um motor de automação da infraestrutura que automatiza o provisionamento da nuvem. A principal razão para usar o Ansible como nossa ferramenta de gerenciamento de configuração é o fato de ele ser projetado para implantações em vários níveis. O Ansible monta a infraestrutura contando como os servidores se inter-relacionam, em vez de apenas um único servidor por vez. Um conceito-chave do Ansible é o uso de arquivos de inventário, que são considerados como um único ponto de verdade. Além disso, a infraestrutura em nuvem consiste em máquinas virtuais que são criadas e desativadas com muita frequência, tornando a manutenção de um único arquivo de inventário em um ambiente dinamicamente provisionado uma tarefa complicada. No entanto, o Ansible possui todas as funcionalidades para lidar com essas situações, desde arquivos de inventário dinâmicos até módulos de automação completos. (SINGH et al., 2015, tradução nossa)

Portanto, após essa revisão da bibliográfica, pode-se perceber que realmente a área é atual e importante para o mercado, por conta de muitos dos motivos citados, mas principalmente, por proporcionar agilidade às empresas de tecnologia no contexto do mundo contemporâneo e globalizado.

3 Tecnologias Escolhidas

A partir dos estudos realizados, este trabalho considerou que uma configuração interessante para construir nuvem privada são as duas ferramentas da Canonical, JuJu e MaaS, para a etapa de implantação da infraestrutura de sistema operacional e softwares nos nós, pois trazem facilidade e automação para disponibilização do ambiente OpenStack. A partir da instalação dos componentes OpenStack e preparação da infraestrutura como serviço, pode-se utilizar Heat na etapa de implantação da infraestrutura de software para uma aplicação, isto é, o fornecimento de máquinas e serviços já no IaaS. Esta última etapa ocorre por meio das *stacks* criadas por modelos escritos em *Heat Template File* (HOT). Então, em suma, em função dos estudos e análises realizados, propões-se criar um experimento que consiste em utilizar, respectivamente, *MaaS* e *JuJu* para definir, criar e gerenciar as nós com sistemas operacionais e provisionar os componentes necessários para instalação da nuvem *OpenStack*. Depois de alcançar a infraestrutura como serviço, busca-se exercitar a prática de infraestrutura como código através de *scripts* HOT.

As próximas subseções descrevem em detalhes as tecnologias utilizadas e sua atuação na construção da nuvem.

3.1 MaaS

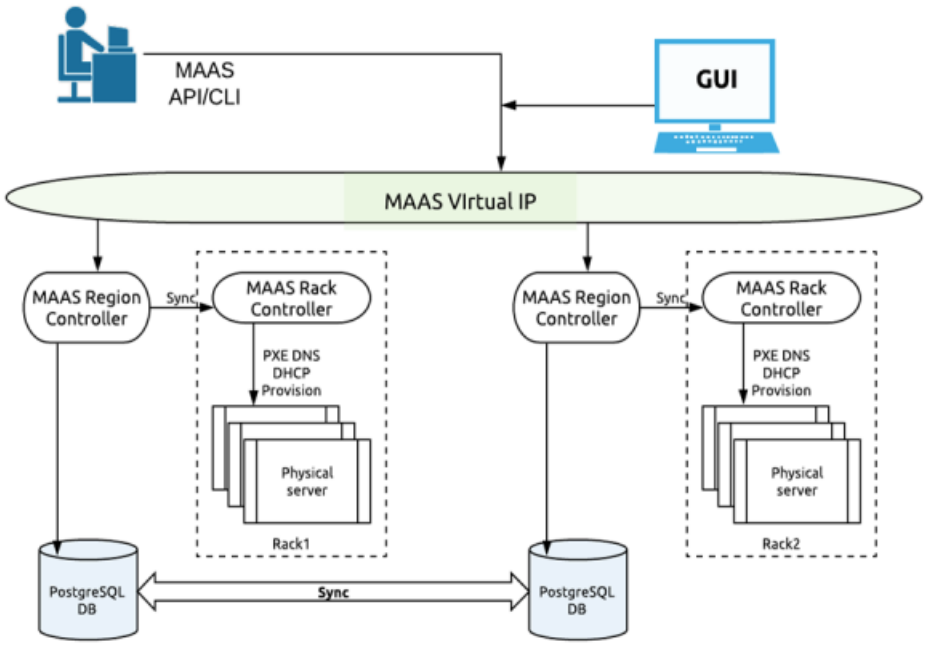
Metal as a Service é uma tecnologia que oferece uma abordagem automatizada para o gerenciamento e a implantação de hardware físico, como servidores, *switches* e roteadores, em ambientes de nuvem e *data centers*. Ao tratar o hardware como um serviço, esse serviço simplifica e agiliza o processo de provisionamento e configuração desses componentes físicos. A Figura 5 descreve a arquitetura geral desse serviço, destacando o uso dos protocolos PXE, DNS e DHCP para provisionar as máquinas físicas com sistemas operacionais e outros serviços pelo *cloud-init*.

Além disso, a ferramenta MaaS permite automatizar tarefas de inicialização, provisionamento e monitoramento de dispositivos físicos. No nosso caso, MaaS é utilizado para configuração da rede virtual e para a instalação dos sistemas operacionais nas máquinas do laboratório.

Por conta dessa automação no comissionamento de máquinas físicas com os sistemas operacionais e possíveis configurações iniciais dentro desse SO possibilitadas pelo *cloud-init*, essa tecnologia torna a gestão de hardware tão dinâmica quanto a computação em

[1] Disponível em: https://infohub.delltechnologies.com/l/reference-architecture-canonical-charmed-openstack-ussuri-on-dell-emc-hardware/high-availability-in-maas. Acesso em 14 Ago. 2023

Figura 5 – Arquitetura geral do MaaS para N Racks.



Fonte: *Dell Technologies* [1].

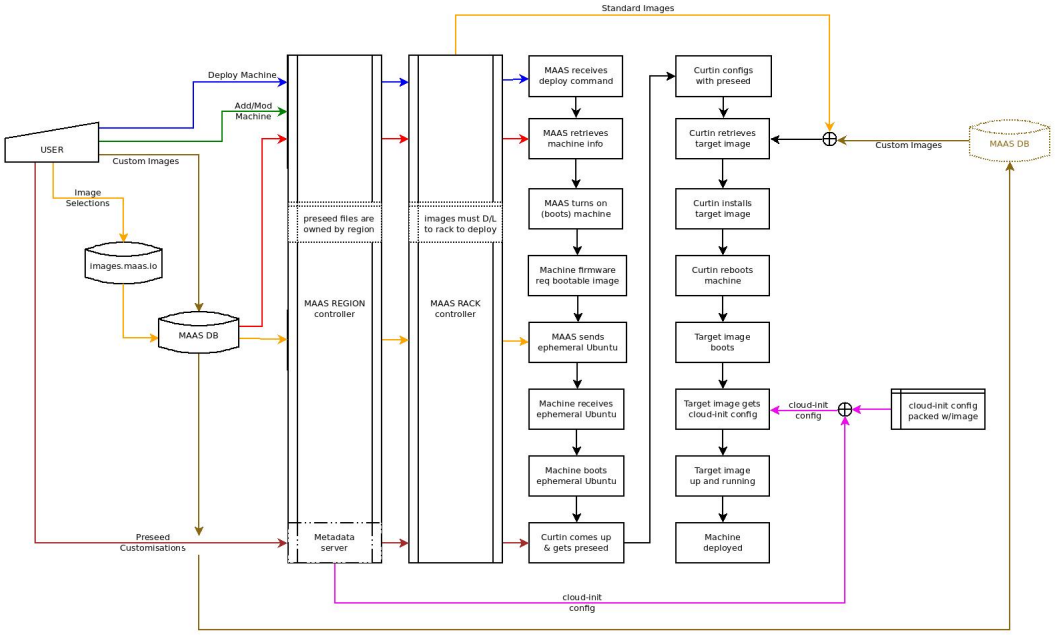
nuvem, aumentando a eficiência, reduzindo erros e acelerando o ciclo de implementação. Também é fácil visualizar possibilidades de escalabilidade na infraestrutura que essa tecnologia oferece. A Figura 6 evidencia o funcionamento do provisionamento de imagens de sistemas operacionais para as máquinas físicas, mostrando todas as etapas envolvidas desde a consulta ao banco de dados (se não for uma imagem customizada) até a etapa de comissionamento, a figura também apresenta o trajeto do *cloud-init*.

Principais características e funções providas pelo sistema MaaS, segundo a documentação:

- **Provisionamento Rápido.** É possível provisionar servidores e outros componentes físicos em minutos, em vez de horas ou dias.
- **Automatização de Configuração:** MaaS permite definir modelos de configuração que podem ser aplicados automaticamente a dispositivos físicos, garantindo que todos os componentes sejam configurados de maneira consistente.
- **Escalabilidade e Flexibilidade.** Permite a rápida adição ou remoção de hardware conforme a demanda do ambiente. Isso é particularmente útil em ambientes onde a carga de trabalho varia.
- **Gestão Centralizada.** Os administradores podem gerenciar todos os recursos físicos a partir de um único painel de controle, o que simplifica o monitoramento, a atualização e a manutenção.

[2] Disponível em: https://maas.io/docs/about-images. Acesso em 14 Ago. 2023

Figura 6 – Caminho da imagem: do banco de dados até as máquinas físicas.



Fonte: *MaaS documentation* [2].

- **Suporte a Diversos Fabricantes.** Suporta uma variedade de fabricantes de hardware e é compatível com diferentes tipos de dispositivos, desde servidores até dispositivos de rede.

- **Integração com Outras Tecnologias.** Pode ser integrado a outras ferramentas de orquestração e gerenciamento, como o OpenStack, permitindo uma automação mais abrangente.

Portanto, MaaS é uma tecnologia que torna o gerenciamento e a implantação de hardware físico tão ágeis e automatizados quanto a implantação de recursos virtuais em ambientes de nuvem. Isso é especialmente benéfico em cenários que envolvem a gestão de grandes quantidades de hardware físico, garantindo eficiência, padronização e redução de complexidade. Para experimentar e avaliar essas questões, no experimento realizado por este trabalho, o sistema MaaS foi instalado em um nó *host* de controle da arquitetura e a partir daí pode comissionar outros nós seguindo por meio da CLI ou interface gráfica.

3.2 JuJu

Juju é uma plataforma de automação de orquestração de aplicativos desenvolvida pela Canonical, a mesma empresa por trás do sistema operacional Ubuntu. O nome “Juju” é derivado da palavra “conjurar”, refletindo sua capacidade de conjurar e orquestrar softwares de maneira eficiente para criar e gerenciar ambientes de aplicativos complexos.

Mais especificamente, essa tecnologia é capaz de instalar e configurar programas como, por exemplo postgresql, por meio de *scripts* de instalação, que podem ser generalizados para a implantação de ambientes mais complexos como o OpenStack.

O objetivo principal da ferramenta Juju é simplificar e automatizar o processo de implantação, gerenciamento e escalabilidade de aplicativos em nuvens públicas, privadas ou híbridas, bem como em ambientes de data center. Ele permite que os desenvolvedores e administradores definam modelos de aplicativos usando *charms* (encantos/encantamentos), que são pacotes reutilizáveis de *scripts*, configurações e metadados que descrevem as características e dependências de um aplicativo que será instalado pela ferramenta.

O processo de implantação com Juju envolve o uso desses *charms* para descrever como cada componente do aplicativo deve ser configurado e conectado. A partir disso, esses componentes são instalados em um ou vários servidores, configurando as dependências, os ajustes de rede e outros parâmetros necessários.

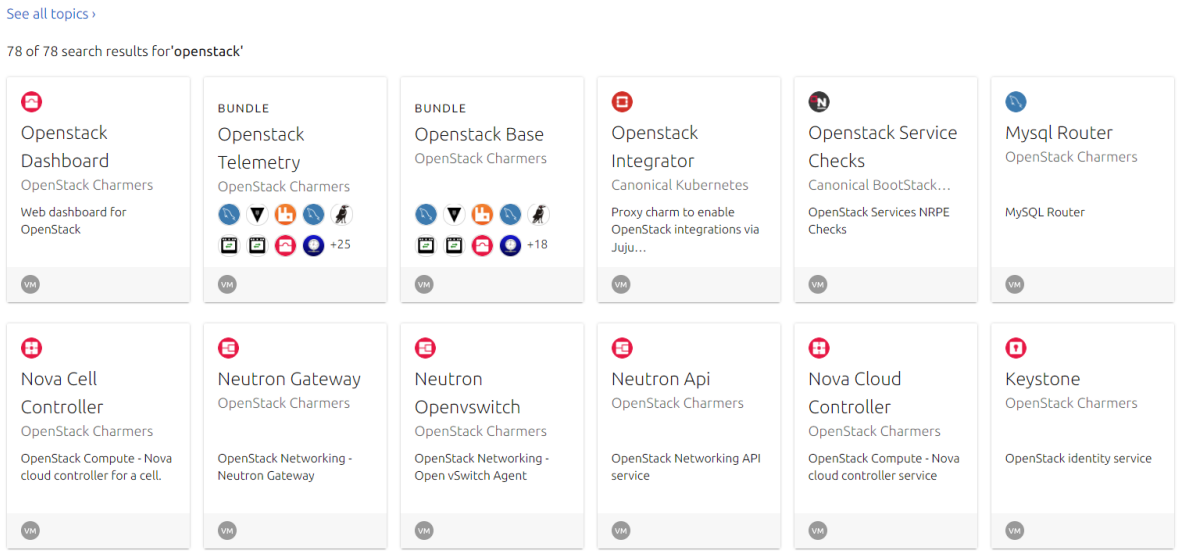
Para simplificar, é possível listar os principais recursos e benefícios providos por Juju:

- **Orquestração Flexível.** permite criar arquiteturas de aplicativos complexas com facilidade, escalando os componentes conforme necessário.
- **Reusabilidade.** *Charms* são reutilizáveis, o que significa que você pode aproveitar modelos prontos para aplicativos populares ou criar seus próprios.
- **Integração com Diversas Plataformas.** Suporta várias nuvens, como Amazon Web Services, Microsoft Azure, Google Cloud Platform, além de ambientes OpenStack, VMware e outros.
- **Automatização de Ciclo de Vida.** Além da implantação inicial, também ajuda a gerenciar atualizações, ajustes de configuração, escalabilidade e até mesmo migração de aplicativos.
- **Facilidade de Uso.** Com sua abordagem baseada em modelos, simplifica a complexidade da configuração de aplicativos, reduzindo erros humanos e acelerando o tempo de implementação (DevOps).

Sendo assim, é uma ferramenta poderosa para a etapa de configuração e implantação de software nos nós da arquitetura, pois, com a automatização provida torna o gerenciamento de aplicativos em "*data centers*" mais eficientes, permitindo que as equipes de TI se concentrem em criar valor através dos aplicativos, em vez de lidar com a complexidade da infraestrutura. Para este projeto, é particularmente interessante ressaltar que a instalação das dependências dos serviços OpenStack foi realizada através desses pacotes com conjuntos de *scripts* que instalam por exemplo, bancos de dados, linguagens, serviços

"*brokers*" para comunicação entre os serviços do OpenStack e as configurações necessárias para essa troca de mensagem. Na figura 7, é possível verificar a grande quantidade de "encantamentos" que a página *charmhub*[3] oferece para OpenStack.

Figura 7 – Lista de pacotes para instalação de OpenStack com JuJu.



Fonte: *Charmhub* [4].

3.3 OpenStack

OpenStack é uma plataforma de código aberto projetada para criar e gerenciar ambientes de nuvem pública e privada, no caso desse experimento, uma nuvem privada IaaS. Esta plataforma fornece um conjunto de serviços e ferramentas que permitem aos usuários criar e gerenciar recursos como computação, armazenamento e rede em uma infraestrutura de nuvem escalável e flexível.

A principal característica dessa plataforma é sua natureza modular, que permite aos usuários escolher e combinar os serviços necessários para atender às suas necessidades específicas. Cada serviço é projetado para funcionar de maneira independente, mas também pode ser integrado com outros serviços para criar soluções de nuvem abrangentes. A comunicação entre esses serviços ocorre por meio de mensagens em um esquema "*publisher/listener*" utilizando *rabbitmq* [5] como o intermediador.

A seguir são listados os principais componentes de OpenStack:

- **Nova:** O serviço de computação, que gerencia a criação e o gerenciamento de instâncias de máquinas virtuais.

[3] https://charmhub.io/
[4] Disponível em: https://charmhub.io/. Acesso em 14 Ago. 2023
[5] https://www.rabbitmq.com/

- **Neutron:** O serviço de rede, que gerencia a configuração e a conexão de redes virtuais.
- **Cinder:** O serviço de armazenamento, que oferece armazenamento em bloco (*block storage*) para instâncias de máquinas virtuais.
- **Swift:** O serviço de armazenamento em objetos (*object storage*), que fornece armazenamento escalável e distribuído para dados não estruturados.
- **Keystone:** O serviço de autenticação e autorização, que gerencia as credenciais dos usuários e controla o acesso aos recursos.
- **Glance:** O serviço de imagens, que armazena e gerencia imagens de máquinas virtuais usadas para criar instâncias.
- **Horizon:** A interface de usuário baseada na web, que permite aos usuários e administradores gerenciar e monitorar recursos na nuvem.
- **Heat:** O serviço de orquestração, que permite a criação e o gerenciamento de recursos usando modelos predefinidos.
- **Ceilometer:** O serviço de coleta de métricas e telemetria, que monitora o uso e o desempenho dos recursos na nuvem.
- **Trove, Sahara, Zaqar, Manila, entre outros:** Outros serviços opcionais que adicionam funcionalidades específicas, como banco de dados como serviço, processamento de dados em lote, mensagens e compartilhamento de arquivos.

Um diagrama geral de uma possível organização desses componentes é apresentado na Figura 8.

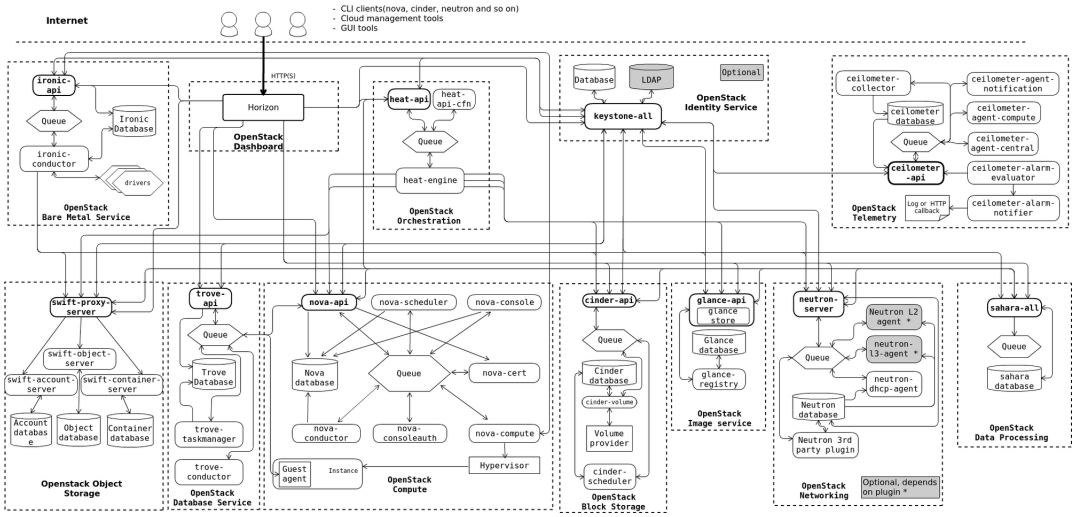
3.4 Heat

OpenStack Heat é um projeto de código aberto dentro da plataforma OpenStack que fornece serviços de orquestração de infraestrutura como código. O objetivo principal do serviço Heat é automatizar e gerenciar a etapa implantação e provisionamento de uma infraestrutura de software para uma aplicação de maneira eficiente e consistente. Na Figura 9, é possível perceber que o Heat (orquestrador) pode utilizar serviços como Neutron, Glance, Cinder, Nova, Swift e Ceilometer nos seus modelos.

[6] Disponível em: https://docs.openstack.org/install-guide/get-started-logical-architecture.html. Acesso em 16 Ago. 2023

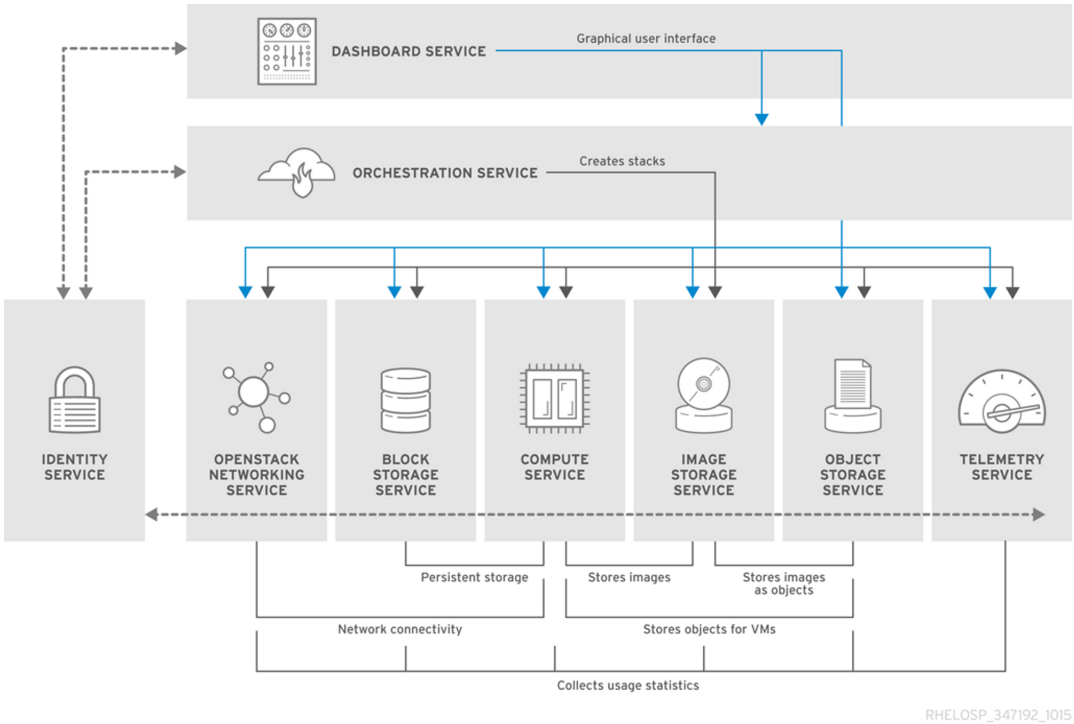
[7] Disponível em: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.0/html-single/product_guide/index. Acesso em 16 Ago. 2023

Figura 8 – O diagrama mostra a arquitetura mais comum, mas não a única possível, para uma nuvem OpenStack.



Fonte: *Documentação do OpenStack* [6].

Figura 9 – Principais serviços disponíveis no OpenStack.



Fonte: *Dococumentação do Red Hat OpenStack* [7].

A orquestração de infraestrutura é um componente essencial na computação em nuvem, pois permite definir, criar e gerenciar recursos complexos de maneira padronizada, repetível e automatizada. Deste modo, é possível aos usuários criar descrições em forma de modelos para definir a infraestrutura necessária para suas aplicações ou serviços. Esses modelos podem incluir máquinas virtuais, redes, armazenamento, balanceadores de carga e outros recursos.

Principais características e funções do serviço Heat:

- **Modelos em Formato de Template.** Utiliza templates para descrever a infraestrutura desejada. Esses templates são escritos em linguagens como o formato YAML ou JSON, permitindo que os usuários especifiquem detalhes da infraestrutura de maneira declarativa.
- **Automação e Padronização.** É possível automatizar a criação, a configuração e a gestão de recursos em nuvem, o que garante a padronização e reduz a possibilidade de erros humanos.
- **Orquestração de Recursos.** Gerencia a ordem correta de criação e interconexão dos recursos, garantindo que todos os componentes estejam corretamente configurados e prontos para uso.
- **Escalabilidade e Elasticidade.** Permite definir políticas de escalabilidade e elasticidade, onde recursos adicionais são provisionados ou desativados automaticamente em resposta a mudanças na carga de trabalho (Funciona em conjunto com a tecnologia Ceilometer).
- **Integração com Outros Projetos OpenStack.** Pode integrar-se com outros serviços do OpenStack, como o Nova (para provisionamento de instâncias) e o Neutron (para configuração de rede).
- **Orquestração Multi-Cloud.** Não é restrito a uma única nuvem, possibilitando a orquestração de recursos em múltiplas plataformas de nuvem pública e privada.

Em resumo, OpenStack Heat é um serviço crucial para automatizar a criação e a gestão de infraestrutura em nuvem, permitindo aos usuários definirem suas necessidades de recursos por meio de modelos e garantindo que esses recursos sejam implantados de maneira eficiente e coordenada. Isso simplifica a implantação de aplicativos complexos e otimiza a utilização dos recursos de nuvem. Por exemplo, para criar uma ou várias máquinas virtuais podemos usar o modelo abaixo:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute
instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
```

```
      key_name: my_key
      image: F18-x86_64-cfntools
      flavor: m1.small
```

Se o nome dado a esse arquivo for exemplo.yaml, pode-se subir uma VM por meio do comando.

```
$ openstack stack create --template exemplo.yaml nome_stack
```

No geral, segundo a documentação [8], pode-se definir parâmetros a serem utilizados pelos recursos (*stacks*) que são criados, criando ambientes mais complexos, com a possibilidade de se definir até uma roteadores virtuais e IP flutuantes. No exemplo abaixo é passado um modelo para se definir uma chave para ser utilizada pelo serviço de autenticação por SSH, a imagem de algum SO e o *flavor*, o "hardware", da máquina virtual:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute
    instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
```

[8] https://docs.openstack.org/heat/rocky/template_guide/hot_guide.html

```
flavor: { get_param: instance_type }
```

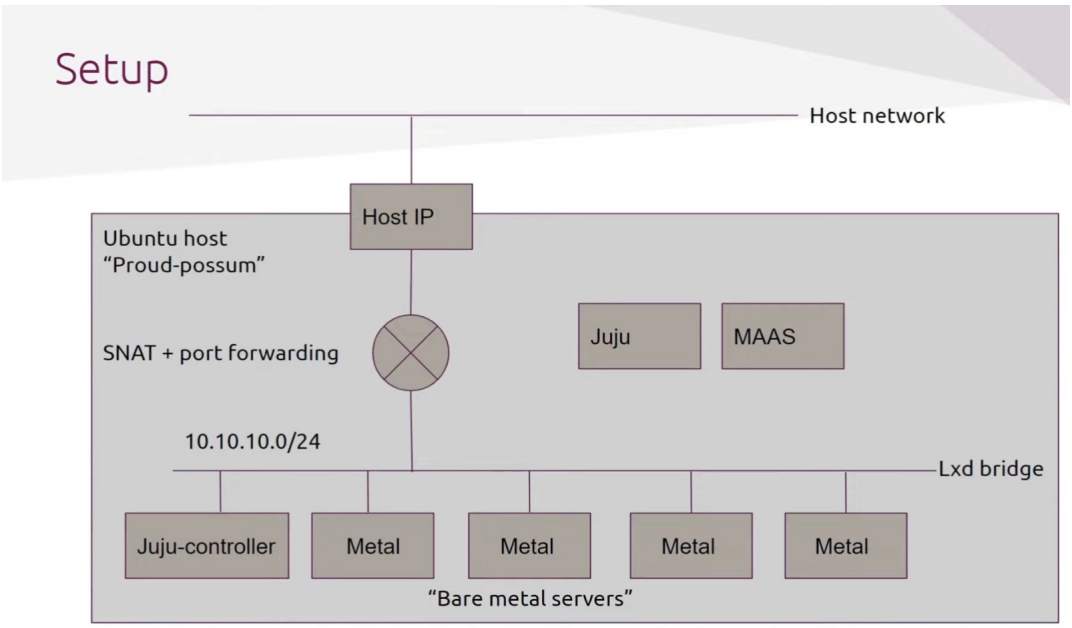
4 Desenvolvimento

Considerando as experimentações com a plataforma de computação em nuvem OpenStack e as ferramentas de automatização realizados neste trabalho, o ambiente preparado para o testes das tecnologias foi inspirado nos tutoriais da documentação OpenStack (2023) e do tutorial de Gisli (2022). Começando a configuração, uma máquina virtual com Ubuntu 22.04 LTS foi preparada em uma nuvem pública para para ser *host* MaaS e acomodar cinco máquinas virtuais disponibilizadas por KVM com as seguintes configurações para o controlador Juju e nós de computação, respectivamente:

- 1 x nó controlador Juju: 4GiB RAM, 2 CPUs, 1 NIC, 1 x 40GiB armazenamento.
- 4 x nós de computação: 8GiB RAM, 2 CPUs, 1 NIC, 1 x 80GiB armazenamento.

As configurações de rede e a organização dos nós podem ser melhor compreendidas na Figura 10. Nela, é possível perceber que a rede foi configurada com SNAT e *port forwarding* a partir do endereço IP da máquina anfitriã e utiliza uma ponte lxd, que será utilizada pelo MaaS para fazer configurações de IP nas máquinas comissionadas, para a partir disso, acessá-las via PXE.

Figura 10 – Arquitetura dos computadores planejada para criação da nuvem IaaS.



Fonte: Figura adaptada de Gisli (2022).

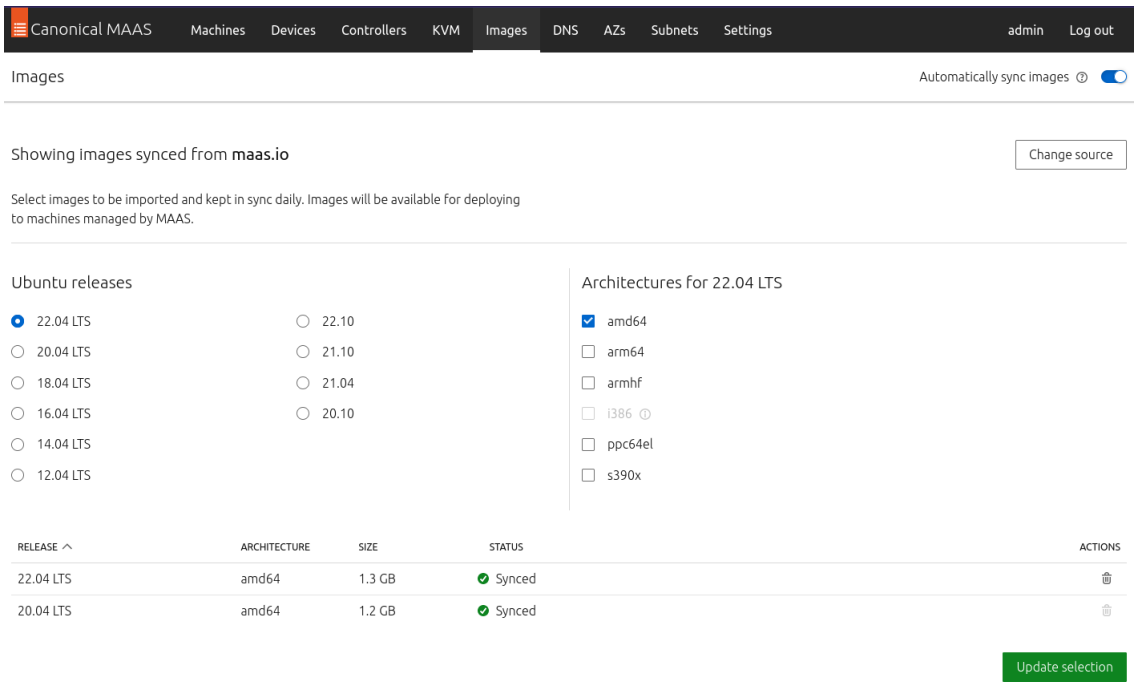
A partir dessas decisões iniciais, a instalação do MaaS na máquina anfitriã é trivial seguindo os guias, permitindo enfim acessar o cliente web da Figura 11 para começar a criação e comissionamento das máquinas virtuais da infraestrutura com alguma das

imagens disponíveis. A Figura 12 apresenta a máquina anfitriã do MaaS. O *host* KVM foi criado através do comando a seguir:

```
maas admin vm-hosts create password=password \
type=lxd power_address=https://${IP_ADDRESS}:8443 \
project=maas | jq '.id')
```

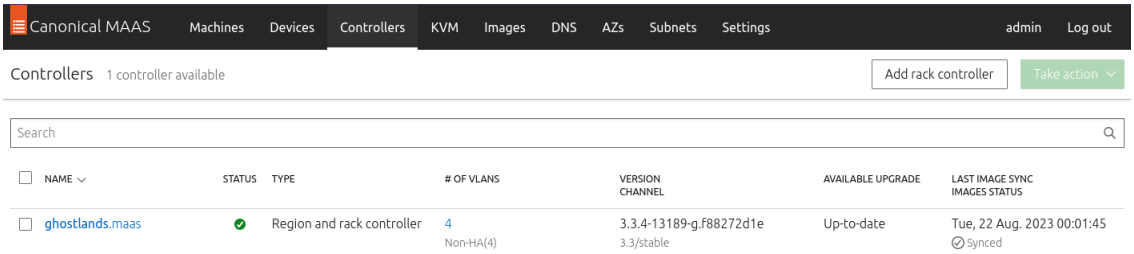
e os nós foram criados nesse KVM com as configurações descritas em 6. A Figura 13 indica na interface a utilização dos recursos pelas máquinas. Na Figura 14 estão identificadas as máquinas da infraestrutura reconhecidas pelo MaaS e prontas para poderem ser usadas pelo Juju. A Figura 15 apresenta a configuração de rede feita.

Figura 11 – Cliente web MaaS com as imagens disponíveis.



Fonte: MaaS Web Interface.

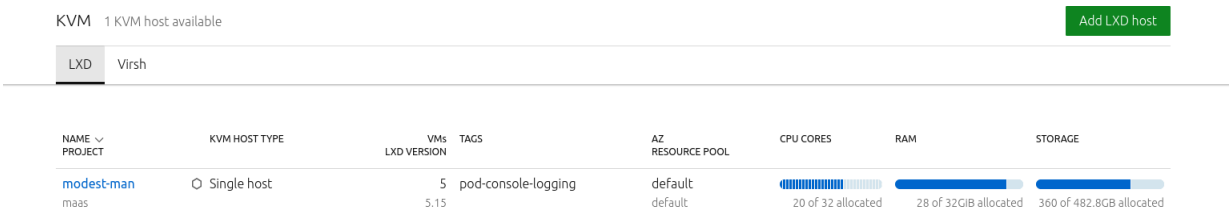
Figura 12 – Cliente web MaaS apresentando seu “rack controller”.



Fonte: MaaS Web Interface.

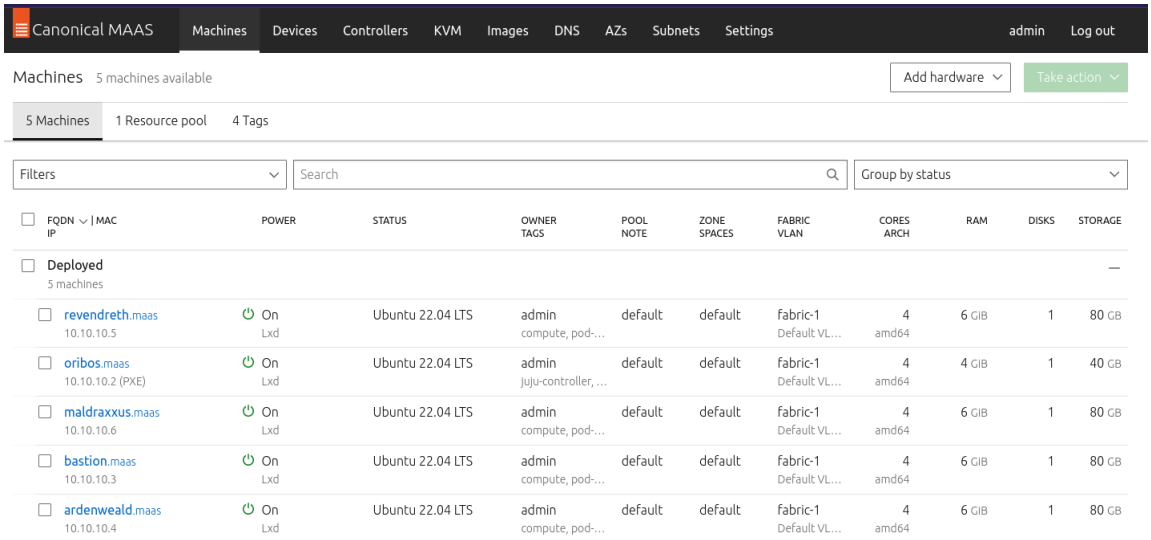
A próxima etapa dos materiais de referência usados nos testes deste trabalho apontam para a instalação do Juju na máquina “controller-juju”. Isto é algo que pode ser feito tranquilamente por meio do MaaS, chegando rapidamente ao estado da Figura 16,

Figura 13 – Recursos consumidos pela infraestrutura.



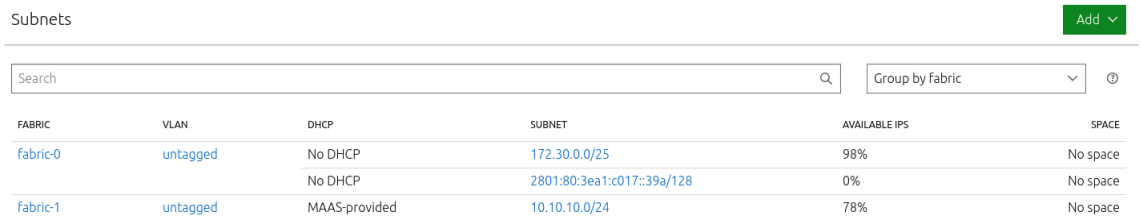
Fonte: MaaS Web Interface.

Figura 14 – Cliente web MaaS apresentando o *controller-juju* e os *computes* para a nuvem.



Fonte: MaaS Web Interface.

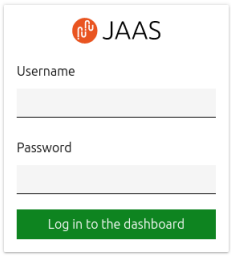
Figura 15 – Configuração de rede no MaaS.



Fonte: MaaS Web Interface.

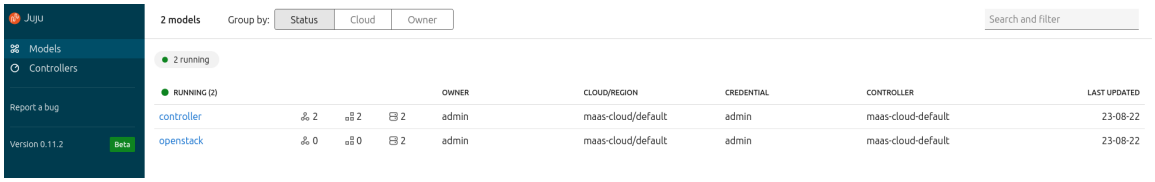
a tela de login do seu cliente web. Para saber as credenciais que podem ser usadas em acessos à plataforma instalada basta rodar o comando "juju dashboard" na CLI e inserir os valores retornados. Após conseguir esse acesso, é possível criar um modelo, conforme o da Figura 17 para começar a instalação dos componentes de forma automatizada nos nós da infraestrutura. Vale notar nessa figura a presença do modelo "*controller*", no qual está presente a unidade que constrói o *dashboard* na máquina "controller-juju" e o modelo openstack. A Figura 18 mostra um exemplo de relacionamento construído por um *charm* que inicializa a interface web.

Figura 16 – Página de Login ao Juju dashboard.



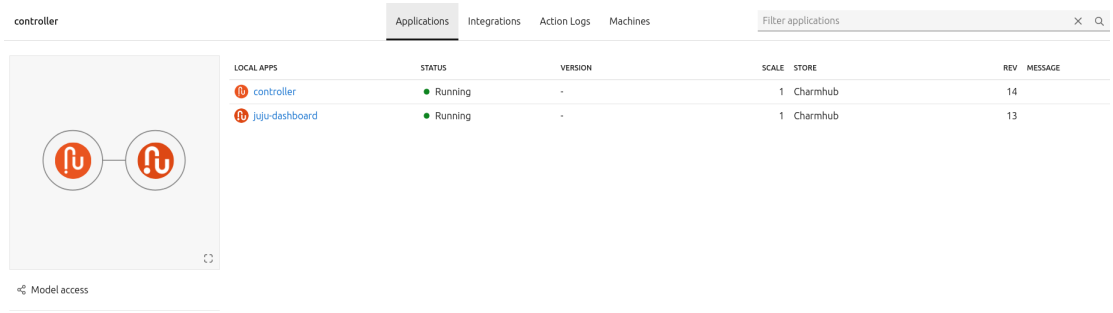
Fonte: Juju Web Interface.

Figura 17 – Modelos *controller* e *openstack*. No controller está o acesso ao dashboard.



Fonte: Juju Web Interface.

Figura 18 – Modelos *controller* e *openstack*. No controller está o acesso ao dashboard.



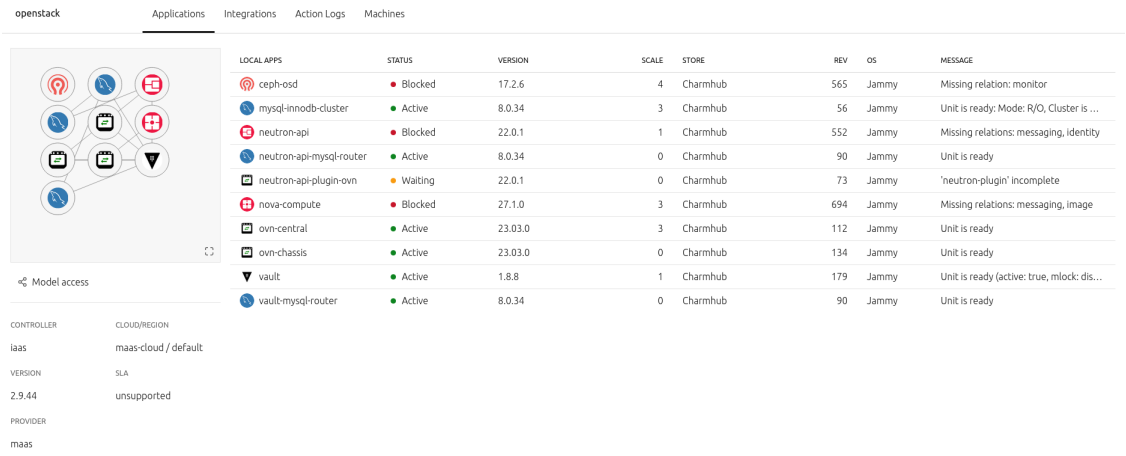
Fonte: Juju Web Interface.

A próxima etapa dos testes tratou da instalação do OpenStack no modelo openstack por meio de *scripts charm*. Os principais comandos executados para esse processo de instalação são “juju deploy <pacote> –to <máquina1,máquina2,...>” e “juju add-relation <pacote1> <pacote2>. O primeiro realiza a instalação do componente em máquinas específicas da arquitetura, podendo até mesmo realizar a conteinerização delas por meio do comando “juju deploy <pacote> –to <lxd:máquina1,lxd:máquina2,...>. E o segundo serve para indicar que existe um relacionamento entre pacotes. Seguindo o passo a passo da documentação OpenStack (2023), o estado das Figuras 19, 20 e 21 é alcançado. Em cada figura é possível verificar o esquemático dos componentes e suas interligações, sendo que os serviços que estão bloqueados ainda não receberam os relacionamentos ou pacotes requisitados.

Ao final, após conclusão dos tutoriais seguidos é possível chegar à configuração da Figura 22. Porém isto não foi atingido neste trabalho.

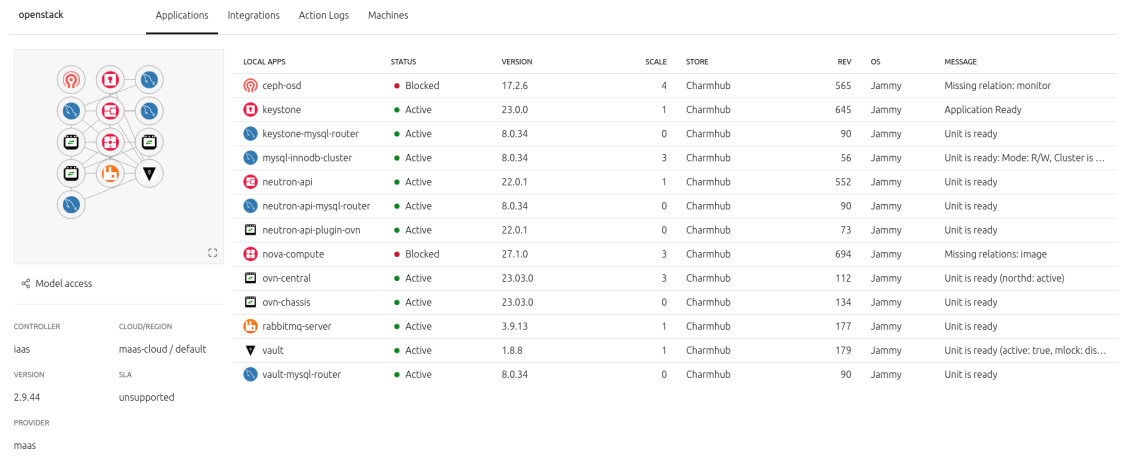
[1] Disponível em: https://infohub.delltechnologies.com/l/reference-architecture-canonical-charmed-

Figura 19 – Estado inicial da instalação do OpenStack.



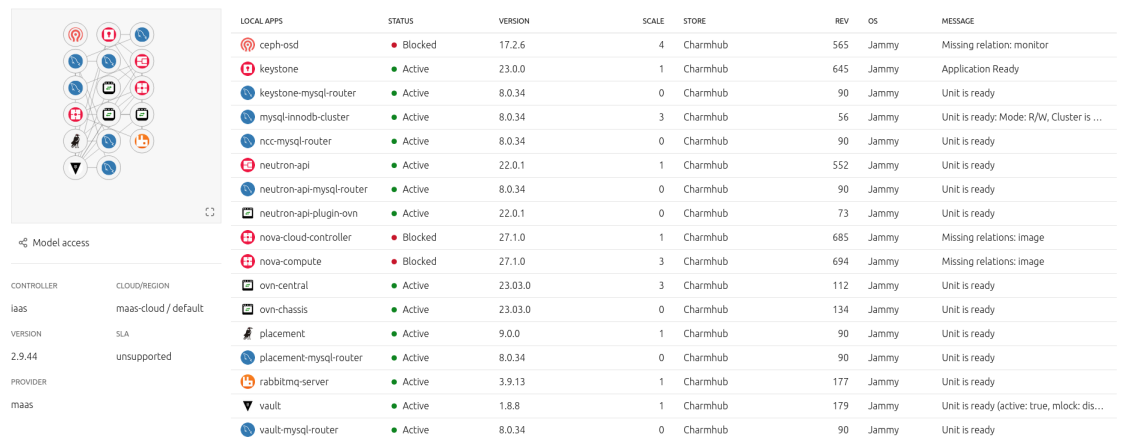
Fonte: Juju Web Interface.

Figura 20 – Estado intermediário da instalação do OpenStack.



Fonte: Juju Web Interface.

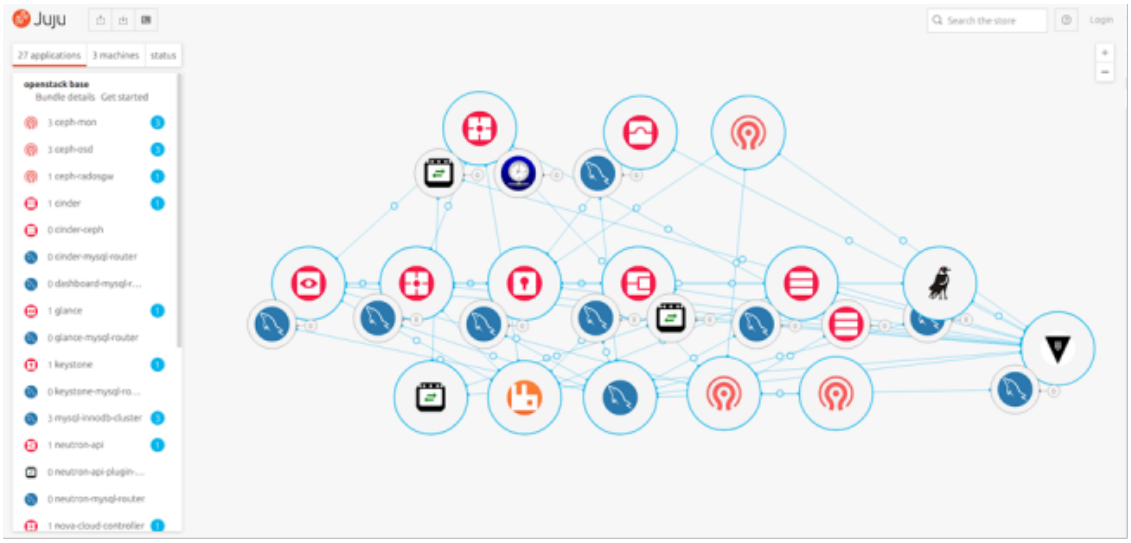
Figura 21 – Estado quase final da instalação do OpenStack.



Fonte: Juju Web Interface.

openstack-ussuri-on-dell-emc-hardware/juju-components. Acesso em 25 Ago. 2023

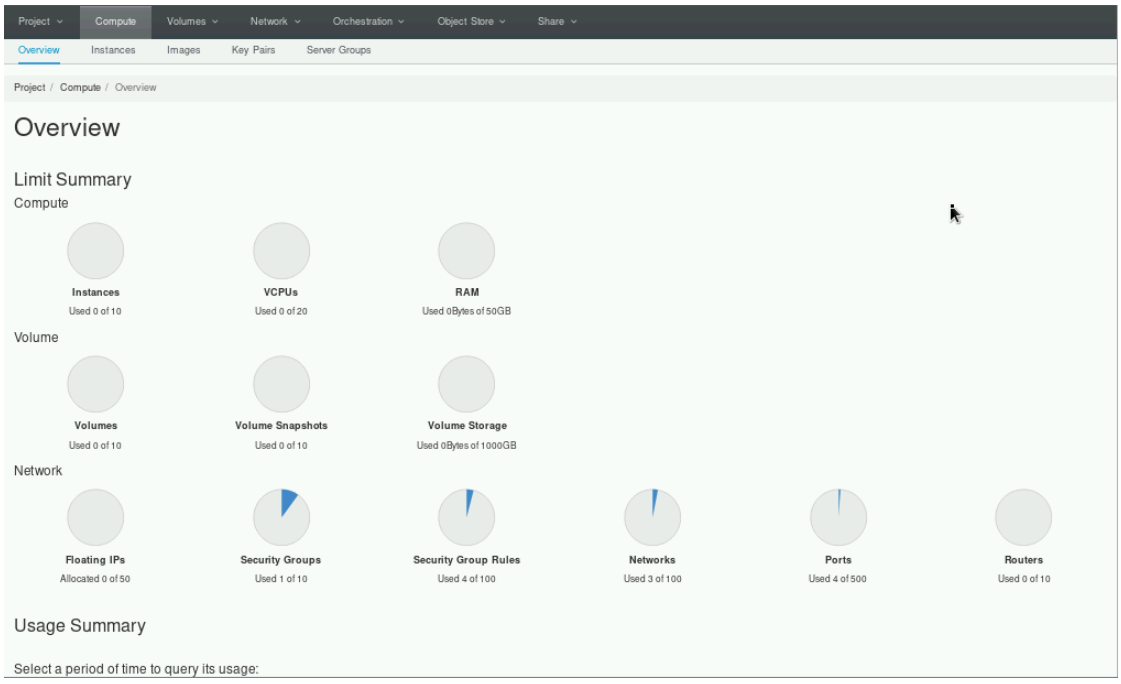
Figura 22 – Configuração final da instalação do OpenStack.



Fonte: Dell Technologies [1].

Então, para continuar o experimento proposto e utilizar Heat para criação de *stacks*, foi utilizada uma plataforma IaaS OpenStack já construída. Por meio de seu dashboard (horizon), ilustrado na Figura 23, as *stacks* por meio da aba de orquestração, a Figura 24 indica como esse serviço funciona na interface. Basta configurar um arquivo .yml com a sintaxe HOT e importar este arquivo na interface. Após esse processo, os serviços iniciam o trabalho e a *stack* fica em estado de criação, conforme indicado na Figura 25.

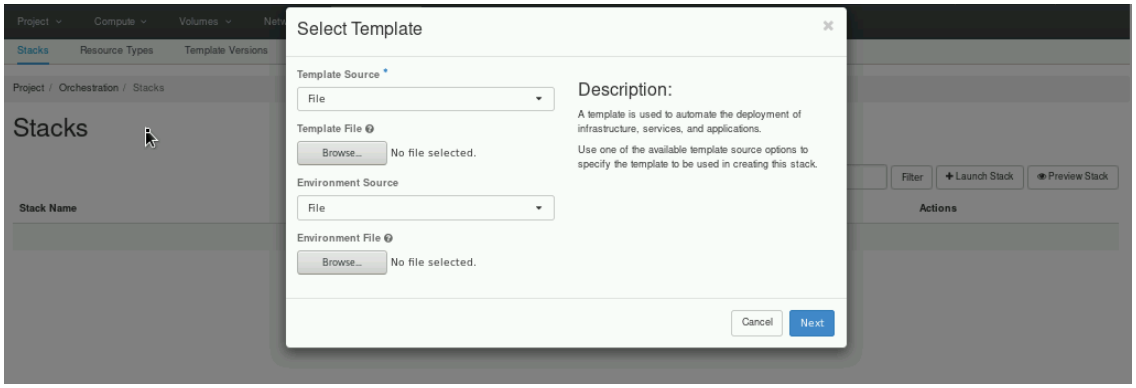
Figura 23 – Visão geral da interface OpenStack.



Fonte: OpenStack Web Interface.

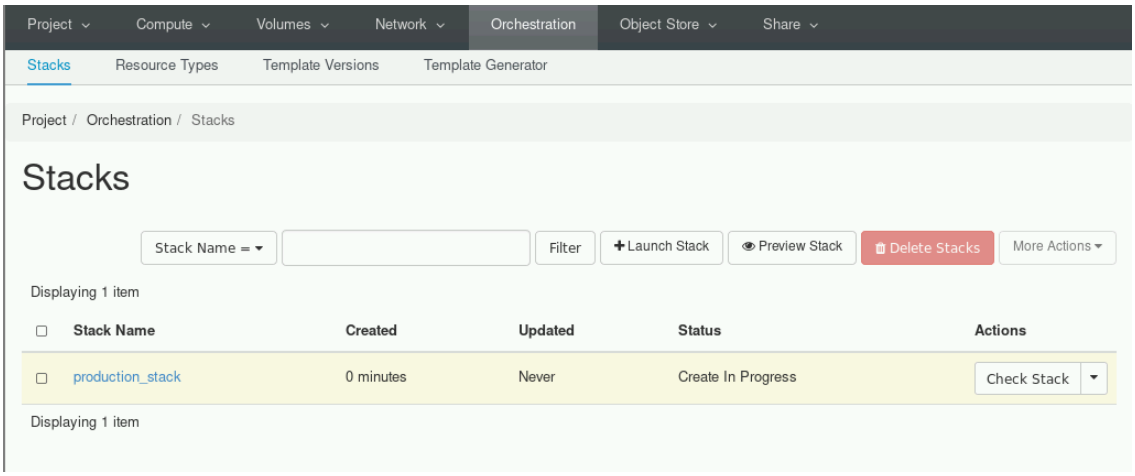
Depois de todo esse processo, os componentes especificados foram criados auto-

Figura 24 – Serviço de orquestração Heat na interface.



Fonte: OpenStack Web Interface.

Figura 25 – Estado de criação da infraestrutura.

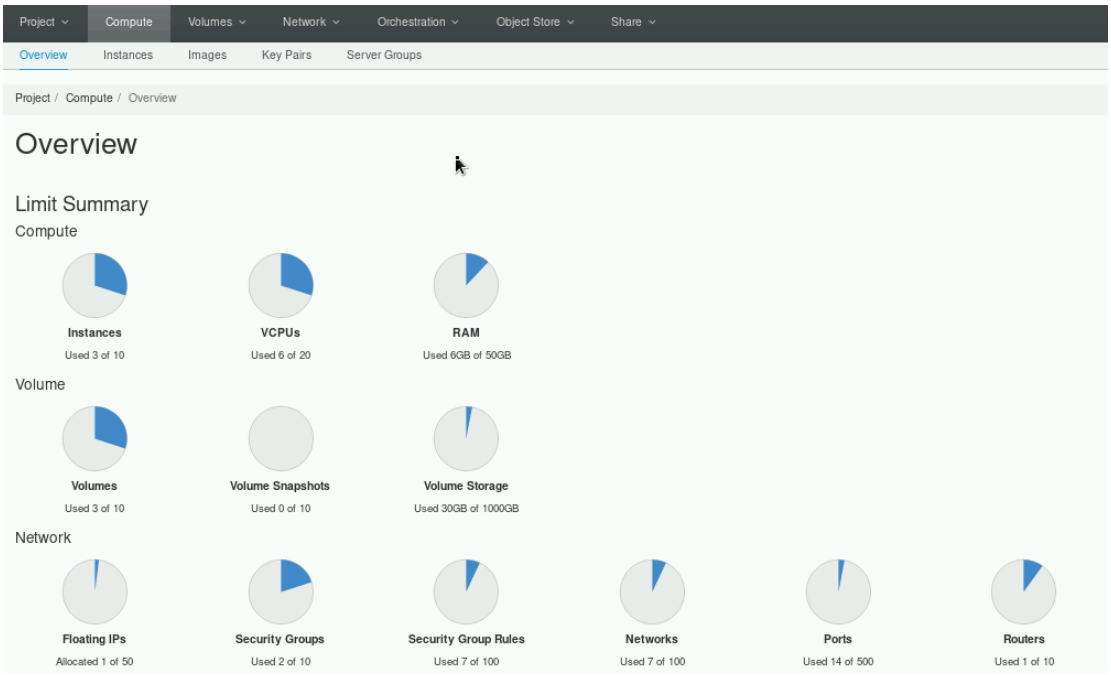


Fonte: OpenStack Web Interface.

maticamente pelos serviços OpenStack e é possível analisar na interface que os recursos realmente foram consumidos. A Figura 26 aponta essa diferença com a Figura 23 que mostra o estado inicial. Também é possível verificar a topologia da rede construída 27.

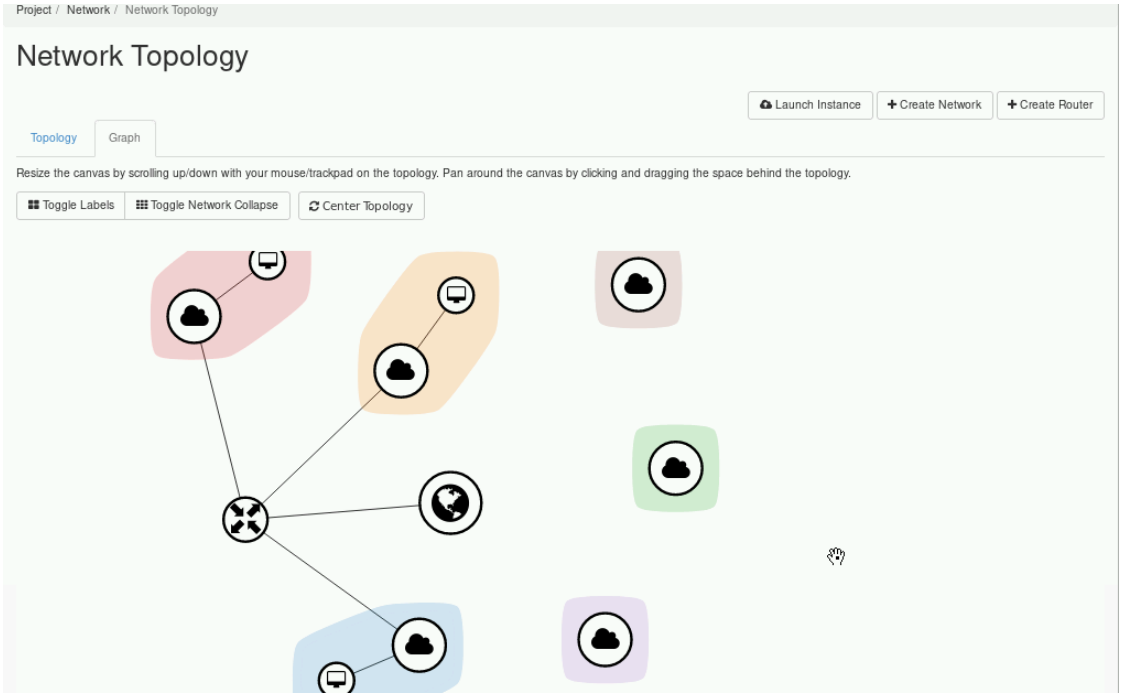
Por fim, mesmo que os experimentos realizados neste estudo tenham se limitado à criação do ambiente de computação em nuvem com OpenStack, foi possível validar a funcionalidade de todas as tecnologias selecionadas para análise.

Figura 26 – Visão geral da interface OpenStack com os recursos consumidos.



Fonte: OpenStack Web Interface.

Figura 27 – Visão geral da topologia construída pela *stack*.



Fonte: OpenStack Web Interface.

5 Resultados

A partir dos estudos e das experimentações realizados com as tecnologias descritas na seção de desenvolvimento deste trabalho, é possível entender que a construção de uma nuvem a partir de máquinas físicas possui etapas e algumas delas podem e devem ser automatizadas por tecnologias disponíveis na atualidade. Pois, como discutido, isto traz inúmeras vantagens para o ambiente operacional de empresas.

Explicitando a etapa em que cada tecnologia abordada pode atuar e como ela faz isso, vê-se que MaaS pode ser responsável por automatizar a etapa inicial da construção da nuvem, etapa em que os nós da infraestrutura ainda estão sem nenhuma "vida". Nos testes realizados esta foi responsável por comissionar as máquinas com "Ubuntu Server 22.04 LTS" e realizar configurações básicas na rede como criação de *Subnets*, *tags* e *Vlans*.

Com todas as máquinas operantes e as configurações de redes realizadas, Juju atua ainda na fase de consolidação da estrutura para o ambiente IaaS. Por meio de arquivos YAML disponibilizados pelos pacotes de *charms* OpenStack, as dependências de sistema de cada um dos serviços são instaladas, além disso, eventuais configurações de rede também são providas por esses *scripts*. A Figura 8 mostra a arquitetura lógica de alguns serviços do OpenStack e as tecnologias e comunicações necessárias.

Com o OpenStack instalado e operante, todos os serviços podem ser utilizados pelo orquestrador na etapa de implantação da infraestrutura de software para a construção de um aplicativo. Das muitas tecnologia disponíveis no mercado, Heat foi a estudada para o projeto por conta da facilidade de instalação, já que é um serviço básico do OpenStack. Então, por meio dos seus modelos HOT, máquinas virtuais, roteadores virtuais, IPs flutuantes, chaves SSH e armazenamento são configurados para fornecer infraestrutura para alguma aplicação que utiliza a infraestrutura como serviço da computação em nuvem.

Como resultado do estudo realizado neste trabalho é possível compreender a etapa de atuação de cada uma delas e suas principais características. A Tabela 1 resume o resultado do estudo teórico para cada tecnologia. Para facilitar a descrição da etapa de atuação de cada tecnologia, a parte de implantação da infraestrutura de software nos nós e fornecimento de tecnologias básicas para a nuvem é chamada de "etapa 1" na tabela e a implantação de uma infraestrutura virtual para um aplicativo de "etapa 2" na tabela.

Mesmo que algumas dessas tecnologias não tenham sido utilizadas neste trabalho, é interessante apontar o estudo realizado sobre elas, pois podem ser utilizadas no projeto de extensão no futuro. A seguir, são apontadas características importantes para essas tecnologias de orquestração que não foram utilizadas nesse experimento. Isto é feito com base nos estudos realizados e no artigo de Edwards (2022).

Tabela 1 – Comparação entre as principais tecnologias de automação estudadas

Tecnologia	**Fase de Atuação**	**Função**	**Open Source**
MaaS	Etapa 1	Configuração nós físicos	Sim
Juju	Etapa 1	Configuração de dependências	Sim
Heat	Etapa 2	Orquestração	Sim
Chef	Etapa 2	Orquestração	Sim
Puppet	Etapa 2	Orquestração	Sim
Terraform	Etapa 2	Orquestração	Sim

- **Terraform**. Tem capacidade de realizar operações multi-clouds e sua sintaxe é relativamente simples, como se vê num exemplo apresentado no Código 5. A linguagem é declarativa, o que permite visualizar o estado da aplicação antes de efetivamente lançá-la.

- **Chef**. É uma tecnologia com bastante documentação e modelos (*recipes*) prontos na literatura, trazendo facilidades de escalabilidade. Um aspecto a considerar é que sua instalação é complicada e a escrita de seus *scripts* depende de conhecimentos da linguagem Ruby.

- **Puppet**. Muito parecido com Chef nas características relevantes, e necessita conhecimentos em PuppetDSL para escrita dos códigos.

```
\\ Exemplo de atribuicao de imagem em Terraform.
storage_profile_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "16.04-LTS"
    version   = "latest"
 }
```

6 Próximos Passos

Dando continuidade aos estudos iniciados com este trabalho, pretende-se aplicar os conhecimentos produzidos neste trabalho para o laboratório do projeto de extensão no DC/UFSCar, onde há 3 máquinas físicas e um comutador. Cada máquina tem as seguintes configurações 6.

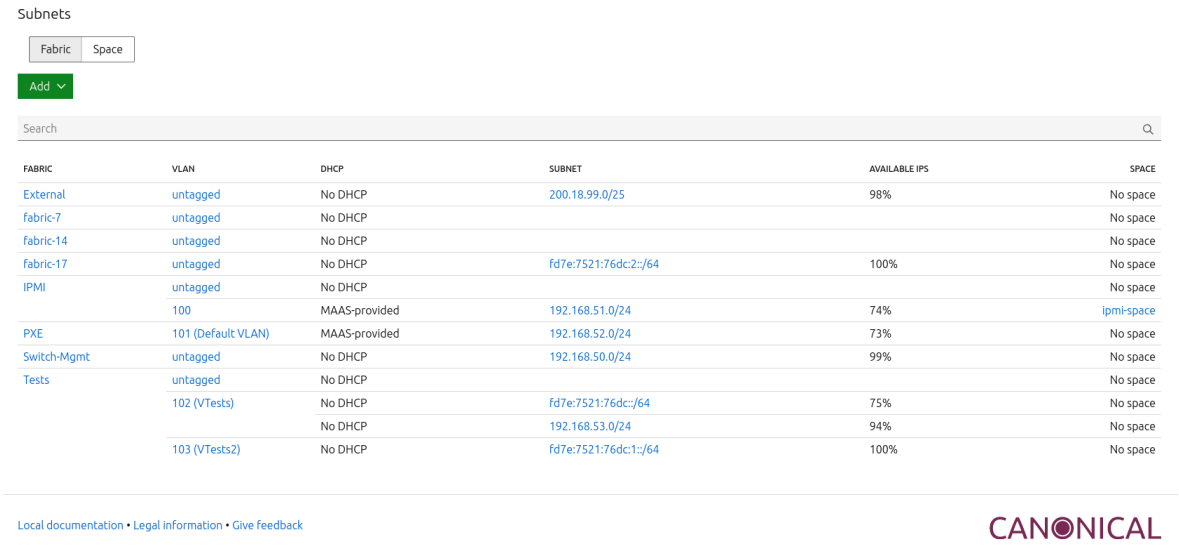
- Motherboard: OX947H Dell Inc..
- CPU: 4x Intel Xeon X7460.
- Ethernet Adapter: 4x Embedded BCM5708; 4x Embedded BCM5709.
- Video Adapter: ATI ES1000 Video
- RAM: 32x NANYA 4GB PC2-5300 DDR2 FBD MEMORY
- Maximum Capacity: 4x 256GB
- Slot: 4x (x8 PCI Express)
- Connector: 10x USB; 2x DB-15 female; DB-9 Male; 4x RJ-45

Tendo em vista as configurações pretendidas para a infraestrutura disponível, uma máquina foi configurada com três discos rígidos para atuar como controlador Juju, host MaaS e *block storage*. Seu esquema de partições foi dividido entre dois discos rígidos de 256 GB configurados em RAID 1 para o sistema operacional Ubuntu Server 22.04 LTS e os programas de gerenciamento (MaaS e Juju), e um disco rígido de 2 TB somente para armazenamento em bloco. As outras duas são os nossos nós de computação.

A arquitetura da rede foi construída por outros membros do projeto. Além disso, a tecnologia IPMI para os controladores BMCs *iDRAC* dessas máquinas *Dell* possibilita ligar, desligar e definir a ordem de boot dos nós *computes* remotamente. Na Figura 28 está a configuração feita na rede pelo MaaS, com todas as fábricas, vlans e subnets configuradas. A Figura 29 mostra as imagens disponibilizadas, e a Figura 30 apresenta o *bare metal* provisionado até o momento.

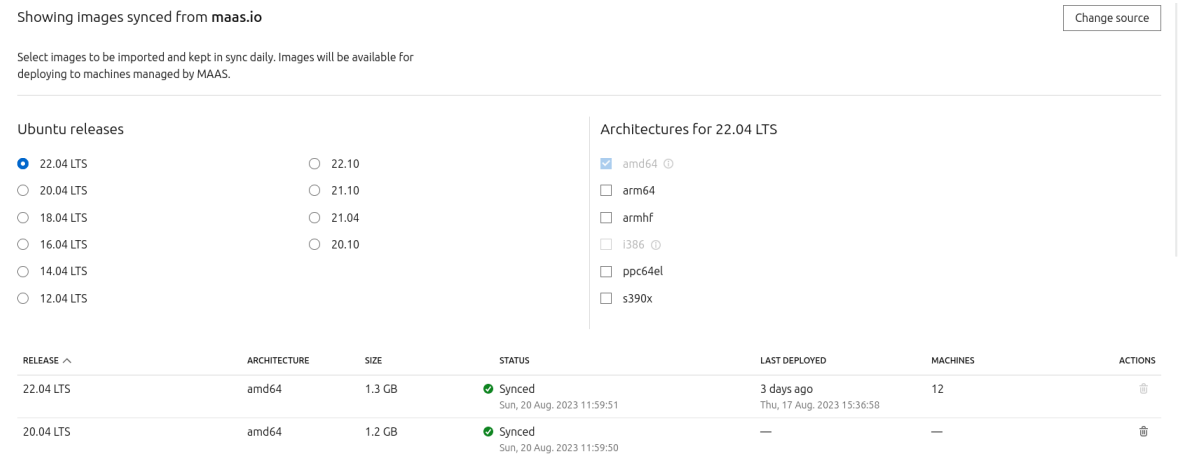
Apesar disso, ainda faltam muitos passos para consolidação dessa infraestrutura. Conforme abordado no desenvolvimento, não temos ainda, por exemplo, o nó controlador de juju.

Figura 28 – Subnets no MaaS.



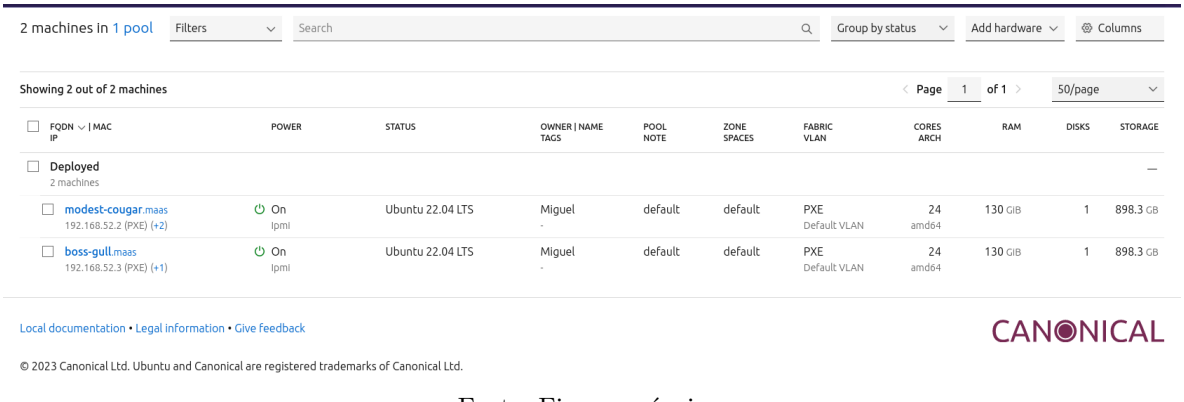
Fonte: Figura própria.

Figura 29 – Imagens de SO's no MaaS.



Fonte: Figura própria.

Figura 30 – Máquinas físicas provisionadas pelo MaaS.



Fonte: Figura própria.

7 Conclusão

Foi observado que a área de computação em nuvem é extremamente vasta, com inúmeras opções disponíveis em suas diversas camadas de desenvolvimento. Cada detalhe deve ser cuidadosamente considerado e configurado de acordo com as necessidades do projeto. No entanto, às vezes, o ambiente em nuvem projetado pode não atender plenamente aos requisitos, o que pode exigir uma recriação do mesmo. Nesse contexto, as tecnologias de automação da infraestrutura, seja ela IaC ou automação do *bare metal*, surgem como um conceito fundamental na computação em nuvem, permitindo automatizar, padronizar e versionar o processo de criação da infraestrutura para ambientes IaaS. Com o auxílio de ferramentas adequadas, é possível até mesmo automatizar a utilização dos serviços oferecidos por plataformas IaaS.

Conforme apresentado ao longo do texto, as ferramentas de infraestrutura como código são amplamente utilizadas na indústria e desempenham um papel fundamental no mercado de serviços em nuvem atualmente, permitindo a implementação de Integração Contínua (CI), Desenvolvimento Contínuo (CD) e práticas DevOps nessas empresas. Além disso, outras tecnologias de automação de infraestrutura também são relevantes, como o próprio MaaS da Canonical. Através de pesquisas na literatura e análise de materiais disponíveis, chegou-se à conclusão de que as ferramentas de IaC mais adequadas para o projeto foram Juju, também da Canonical, e Heat do OpenStack.

Dessa forma, após definição das tecnologias, ocorreu a etapa de experimentação na construção de uma infraestrutura, na qual foi possível validar a funcionalidade de MaaS, JuJu e Heat, elaborando um estudo comparativo de suas funcionalidades e características relevantes dentro do processo de concepção do ambiente em nuvem e de máquinas virtuais.

Porém, mesmo trabalhando com algumas ferramentas, uma limitação deste trabalho foi a gama limitada delas testadas. Para este trabalho de conclusão de curso foram utilizadas apenas três das várias disponíveis. Dito isto, um dos objetivos para o futuro do projeto é abordar e documentar mais delas, como Puppet e Chef, as quais, como foi visto, têm bastante documentação na literatura. Também um trabalho futuro é utilizar a configuração estudada para o laboratório deste projeto.

Após todas essas etapas e resultados apresentados, foi possível validar o funcionamento das tecnologias de automação na tentativa de construção de uma infraestrutura como serviço utilizando ferramentas de automação. Com a etapa de construção das máquinas virtuais com orquestração feita em cima da IaaS OpenStack, foi possível verificar que de fato, tecnologias de automação para construção de uma nuvem facilitam o processo e garantem uma infraestrutural ágil, confiável e passível de ser escalada para os futuros

objetivos do projeto.

Referências

AIFTIMIEI, C. et al. Cloud environment automation: from infrastructure deployment to application monitoring. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2017. v. 898, n. 8, p. 082016. Citado na página 11.

ARTAC, M. et al. Infrastructure-as-code for data-intensive architectures: a model-driven development approach. In: IEEE. *2018 IEEE international conference on software architecture (ICSA)*. [S.l.], 2018. p. 156–15609. Citado na página 4.

ARTAC, M. et al. Devops: introducing infrastructure-as-code. In: IEEE. *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. [S.l.], 2017. p. 497–498. Citado 2 vezes nas páginas 7 e 9.

CARVALHO, L. R. D.; ARAUJO, A. P. F. de. Performance comparison of terraform and cloudify as multicloud orchestrators. In: IEEE. *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. [S.l.], 2020. p. 380–389. Citado na página 9.

EDWARDS, A. *Infrastructure as Code (IaC): Comparing the Tools*. 2022. <https://techcommunity.microsoft.com/t5/itops-talk-blog/infrastructure-as-code-iac-comparing-the-tools/ba-p/3205045>. Acesso em: 05 Ago. 2023. Citado 2 vezes nas páginas 4 e 31.

GISLI, A. *Bare metal Kubernetes with MAAS, LXD, and Juju*. 2022. <https://github.com/antongisli/maas-baremetal-k8s-tutorial>. Acesso em: 21 Ago. 2023. Citado na página 23.

GUERRIERO, M. et al. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In: IEEE. *2019 IEEE international conference on software maintenance and evolution (ICSME)*. [S.l.], 2019. p. 580–589. Citado na página 4.

HANAPPI, O.; HUMMER, W.; DUSTDAR, S. Asserting reliable convergence for configuration management scripts. In: *Proceedings of the 2016 ACM SIGPLAN international conference on object-oriented programming, systems, languages, and applications*. [S.l.: s.n.], 2016. p. 328–343. Citado na página 10.

HINTSCH, J.; GÖRLING, C.; TUROWSKI, K. Modularization of software as a service products: A case study of the configuration management tool puppet. In: IEEE. *2015 International Conference on Enterprise Systems (ES)*. [S.l.], 2015. p. 184–191. Citado na página 11.

JIANG, Y.; ADAMS, B. Co-evolution of infrastructure and source code-an empirical study. In: IEEE. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. [S.l.], 2015. p. 45–55. Citado na página 9.

KUMARA, I. et al. The do's and don'ts of infrastructure code: A systematic gray literature review. *Information and Software Technology*, Elsevier, v. 137, p. 106593, 2021. Citado na página 8.

MIGLIERINA, M. Application deployment and management in the cloud. In: IEEE. *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. [S.l.], 2014. p. 422–428. Citado na página 9.

OPENSTACK. *OpenStack Charms Deployment Guide*. 2023. <https://docs.openstack.org/project-deploy-guide/charm-deployment-guide/latest/install-maas.html>. Acesso em: 21 Ago. 2023. Citado 2 vezes nas páginas 23 e 26.

PARNIN, C. et al. The top 10 adages in continuous deployment. *IEEE Software*, IEEE, v. 34, n. 3, p. 86–95, 2017. Citado na página 8.

QIAN, L. et al. Cloud computing: An overview. In: SPRINGER. *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*. [S.l.], 2009. p. 626–631. Citado na página 1.

RAHMAN, A.; MAHDAVI-HEZAVEH, R.; WILLIAMS, L. A systematic mapping study of infrastructure as code research. *Information and Software Technology*, Elsevier, v. 108, p. 65–77, 2019. Citado 4 vezes nas páginas 7, 8, 9 e 10.

ROMERO, E. E. et al. Integration of devops practices on a noise monitor system with circleci and terraform. *ACM Transactions on Management Information Systems (TMIS)*, ACM New York, NY, v. 13, n. 4, p. 1–24, 2022. Citado na página 4.

SANDOBALIN, J.; INSFRAN, E.; ABRAHAO, S. End-to-end automation in cloud infrastructure provisioning. 2017. Citado na página 11.

SANDOBALIN, J.; INSFRAN, E.; ABRAHAO, S. An infrastructure modelling tool for cloud provisioning. In: IEEE. *2017 IEEE international conference on services computing (SCC)*. [S.l.], 2017. p. 354–361. Citado na página 10.

SCHEUNER, J. et al. Cloud workbench: Benchmarking iaas providers based on infrastructure-as-code. In: *Proceedings of the 24th International Conference on World Wide Web*. [S.l.: s.n.], 2015. p. 239–242. Citado na página 9.

SCHEUNER, J. et al. Cloud work bench–infrastructure-as-code based cloud benchmarking. In: IEEE. *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. [S.l.], 2014. p. 246–253. Citado na página 9.

SINGH, N. K. et al. Automated provisioning of application in iaas cloud using ansible configuration management. In: IEEE. *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*. [S.l.], 2015. p. 81–85. Citado na página 11.

SPINELLIS, D. Don't install software by hand. *IEEE software*, IEEE, v. 29, n. 4, p. 86–87, 2012. Citado na página 9.

WETTINGER, J.; BREITENBÜCHER, U.; LEYMANN, F. Devopslang–bridging the gap between development and operations. In: SPRINGER. *Service-Oriented and Cloud Computing: Third European Conference, ESOCC 2014, Manchester, UK, September 2-4, 2014. Proceedings 3*. [S.l.], 2014. p. 108–122. Citado na página 9.

YAMATO, Y. Performance-aware server architecture recommendation and automatic performance verification technology on iaas cloud. *Service Oriented Computing and Applications*, Springer, v. 11, p. 121–135, 2017. Citado 2 vezes nas páginas 1 e 3.