

# Construção de um Ambiente de Cloud Privada Utilizando Infraestrutura como Código (IaC)

**Hailson F. S. Junior, Michele Cristiane Barion, Carlos Eduardo Pagani**

Instituto Federal de São Paulo – Campus Hortolândia

hailson.fsj@gmail.com, michelefreitas@ifsp.edu.br, pagani@ifsp.edu.br

**Abstract.** *Cloud computing or cloud are increasingly common terms, which a short time ago seemed to be something far away to become one of the biggest technologies used by companies. Trends before cloud computing were limited to a certain class of users or focused on making available a specific demand on IT resources, today cloud-based is global. In this work, the private cloud was addressed and aims to build an entire cloud environment, using opensource software, providing computational resources on-demand, based on the OpenStack tool, using Infrastructure as Code (IaC), that is, the creation of the environment in a provisioned and automated way, using DevOps concepts.*

**Resumo.** *Computação em Nuvem ou cloud são termos cada vez mais comuns. O que a pouco tempo atrás parecia ser algo distante se tornou uma das maiores tecnologias utilizadas pelas empresas. Tendências anteriores à Computação em Nuvem foram limitadas a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI. Todavia, hoje Computação em Nuvem é algo global. Neste trabalho foi abordado a nuvem privada e tem como objetivo construir todo um ambiente de nuvem, utilizando software opensource, fornecendo recursos computacionais sob demanda, baseada na ferramenta OpenStack, com a utilização da Infraestrutura como Código (IaC), ou seja, a criação do ambiente de forma automatizada e provisionada utilizando conceitos de DevOps.*

## 1. Introdução

A Computação em Nuvem ou, como muitos conhecem, a *cloud computing*, tem sido uma das áreas que mais cresce no mercado e um dos principais temas relacionados a tecnologia, ainda mais considerando o período de pandemia de Coronavírus que fez com que o mercado global de infraestrutura de serviços (*IaaS*) crescesse em 50% no ano de 2020 [SHIMABUKURO 2021].

A ideia de comprar recursos computacionais sob demanda tem chamado a atenção de muitas empresas, já que com ela temos algumas vantagens como, por exemplo, maior agilidade, elasticidade, economia, provisionamento, automação e monitoramento. Podem ser citados como exemplos de Computação em Nuvem o *PaaS* (*Platform as a Service*), o *SaaS* (*Software as a Service*) e o *IaaS* (*Infrastructure as a Service*).

Neste trabalho será criado um ambiente de Computação em Nuvem no modelo *IaaS*, que fornece ao cliente a infraestrutura como um serviço, baseando-se no OpenStack, *software opensource* para construção de ambiente em nuvem. Para maior controle, segurança e agilidade, o ambiente será construído utilizando a *IaC* (*Infrastructure as Code*) onde temos todos os passos descritos em código e sendo executados de forma automatizada [MICROSOFT 2021].

A motivação deste trabalho parte da importância e da necessidade que muitas empresas possuem em ter um ambiente de Computação em Nuvem, já que novas tecnologias exigem um ambiente computacional mais robusto para execução de suas ferramentas, como também possibilite a segurança no armazenamento dos seus dados. O OpenStack é uma das ferramentas disponíveis para tal propósito que auxilia no gerenciamento de uma infraestrutura em nuvem, seja pública ou privada, provendo serviços para máquinas virtuais, *bare metal*<sup>1</sup> e *containers*<sup>2</sup>. Todavia, sua implementação é complexa, no entanto, por meio da Infraestrutura como Código (*IaC*), pode ser proposto o gerenciamento de uma infraestrutura com *scripts* que auxiliem em atividades que levariam muito tempo para serem executadas manualmente. O desenvolvimento desses *scripts* visa um processo mais rápido, seguro e controlado, facilitando a criação do ambiente.

Para abordagem do objetivo exposto, além da introdução, o presente trabalho está organizado em seis seções, ordenadas em: referencial teórico, trabalhos correlatos, metodologia, desenvolvimento, conclusão e trabalhos futuros, além de apresentar as referências utilizadas tanto na pesquisa sobre as metodologias adotadas quanto das ferramentas que foram fundamentais para o desenvolvimento.

## 2. Referencial Teórico

Nesta seção será apresentado o resumo de discussões de outros autores a respeito de conceitos que foram abordados neste trabalho.

### 2.1. Computação em Nuvem

Existem inúmeras definições para Computação em Nuvem, mas a mesma pode ser definida como a entrega de recursos computacionais sob demanda ou, como apresenta a *National Institute of Standards and Technology* (NIST), como:

[...] um modelo para permitir acesso onipresente, conveniente e sob demanda à rede a um *pool* compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços. Este modelo de nuvem é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação” [NIST 2011].

Com base na definição da NIST, a Computação em Nuvem permite acessar recursos computacionais a partir de qualquer lugar, sem a necessidade de comprar e manter servidores físicos e possibilitando o uso de serviços oferecidos por meio da Internet. Assim, evitam-se gastos desnecessários, maior agilidade em subir serviços e servidores em minutos, conforme a necessidade.

Para distinguir a Computação em Nuvem de outros modelos, observam-se cinco características essenciais [MEL e GRANCE 2011]:

1. Autoatendimento sob demanda: Um consumidor deve ser capaz de provisionar os recursos computacionais, conforme necessário;
2. Amplo acesso à rede: Possibilidade de acessar os recursos que estão disponíveis na rede a partir de qualquer dispositivo;

---

<sup>1</sup> Servidores dedicados.

<sup>2</sup> Virtualização a nível de Sistema Operacional. Recursos isolados.

3. Agrupamento de recursos: Recursos agrupados para atender vários consumidores, com recursos sendo atribuídos e retribuídos conforme a necessidade do consumidor, onde este tem a sensação de independência, sem ter uma ideia exata de onde estão alocados seus recursos por conta da camada de abstração;
4. Elasticidade rápida: Capacidade de provisionar os recursos com facilidade e rapidez, em alguns casos, automaticamente de acordo com a demanda;
5. Serviço mensurado: Monitoração e controle dos recursos que estão sendo utilizados.

Quanto aos tipos, a Computação em Nuvem pode ser caracterizada em 3 modelos, a qual cada um deles possui um tipo de serviço, níveis de gerenciamento e controle ao usuário. As subseções 2.1.1, 2.1.2 e 2.1.3 irão abordar esses tipos.

### **2.1.1. IaaS**

É o tipo de Computação em Nuvem, a qual a infraestrutura é fornecida ao usuário como um serviço, ou seja, são fornecidos os componentes básicos de uma infraestrutura. É o nível mais alto de flexibilidade e controle sobre os recursos, podendo ser fornecido, por exemplo, espaço para armazenamento, recursos de rede, e tudo o que compõe uma estrutura de tecnologia da informação. Com isso o cliente não precisa se preocupar com a compra, instalação e manutenção de servidores físicos, replicação de dados, *backup*, entre outras ações fundamentais para que a *IaaS* fique disponível. Assim, tudo será responsabilidade da empresa que está fornecendo a *IaaS* [AWS 2021], no entanto, o cliente é responsável pelo gerenciamento desses recursos no ambiente e configurações de serviços disponibilizados pelo provedor para segurança de seus dados, ou seja, o cliente é responsável pelos seus dados.

### **2.1.2. PaaS**

No segundo tipo de Computação em Nuvem, a plataforma como serviço irá fornecer um ambiente para criação, hospedagem e gerenciamento de *softwares*. Como apresentam [Chacon e Straub 2009]:

é um ambiente de computação em camadas de soluções como serviço. Facilitam a implantação de aplicações de menor custo e complexidade na compra e gestão do *hardware*, *software* e recursos de provisionamento de infraestrutura, fornecendo todas as facilidades necessárias para suportar o ciclo de vida completo de construção e entrega de aplicações *web* e serviços totalmente disponíveis a partir da Internet.

Com o *PaaS* o usuário pode subir sua aplicação sem ter que se preocupar com a parte de infraestrutura, com servidores e instalação de *softwares*, sistema operacional, etc. Utilizando o *PaaS* o cliente se preocupa apenas com o desenvolvimento da sua aplicação.

### **2.1.3. SaaS**

Neste modelo o provedor oferece *software* como serviço, ou seja, não precisa nem mesmo se preocupar com a implantação do *software*.

Utiliza-se o conceito de *SaaS* em praticamente todos os serviços de Internet que são consumidos, como uma ferramenta de busca na *web* (Google, Bing, Yahoo) ou seu e-mail [CHACON e STRAUB 2009].

A Figura 1 traz uma visão dos três tipos de serviços em computação na nuvem considerando o público alvo, além de exemplificar atividades que podem ser desempenhadas por cada um.



**Figura 1. Tipos de serviço em computação na nuvem [GLEB 2017]**

Além dos tipos de nuvem, a *cloud* pode ser apresentada em modelos de implementação, sendo nuvem privada, pública ou híbrida. As subseções 2.1.4, 2.1.5 e 2.1.6 abordam esse assunto.

#### **2.1.4. Nuvem privada**

É um ambiente exclusivo para uma única empresa, podendo ser construída e gerenciada pela própria organização que irá utilizá-la, ou seja,

é um modelo de implantação em nuvem sob demanda em que os serviços e a infraestrutura de Computação em Nuvem são hospedados de maneira privada, geralmente na própria intranet ou data center de uma empresa, por meio de recursos proprietários que não são compartilhados com outras organizações [VMWARE 2021].

#### **2.1.5. Nuvem pública**

Diferente da nuvem privada, no modelo público um provedor compartilha seus recursos com diversas organizações. Como apresentado pela [NIST 2011], a nuvem pública é aquela em que a infraestrutura e os recursos computacionais que a compõem são disponibilizados ao público em geral pela Internet e o cliente paga pelo recurso que for utilizado. Ela pertence e é operada por um provedor que entrega serviços em nuvem e, por definição, é externo às organizações dos consumidores.

#### **2.1.6. Nuvem híbrida**

O terceiro modelo é no formato híbrido, sendo uma forma de conectar recursos da nuvem pública e recursos internos. Como apresentam os autores [Aishwarya, Abdul e Vijayakumar 2015]:

Um modelo de Computação em Nuvem híbrida lida com a mistura de nuvem privada e pública para um trabalho mais proeminente e eficiente das nuvens. Em termos mais simples, o modelo híbrido é principalmente uma nuvem privada que permite a uma organização acessar uma nuvem pública como e quando necessário para o compartilhamento de informações.

### **2.2. OpenStack**

As próximas subseções irão abordar ferramentas que são usadas em ambientes da computação na nuvem.

OpenStack é uma plataforma *opensource*, sendo chamada de sistema operacional da nuvem. Tem como objetivo controlar grandes *pools* de recursos de computação, armazenamento e rede em um *datacenter*, sendo gerenciados e provisionados por meio de *APIs* com mecanismos de autenticação comuns [OPENSTACK 2022].

Com o OpenStack é possível realizar o gerenciamento de nuvens públicas e privadas, através de diversos serviços que devem ser instalados e configurados como, por exemplo, Keystone para autenticação, Horizon para *dashboard*, Glance para registro de imagens de máquinas virtuais, Placement API REST e Compute para hospedar e gerenciar sistemas de Computação em Nuvem, serviço Neutron para rede, Block Storage para armazenamento, entre outros.

### 2.3. DevOps

O DevOps é “a combinação de filosofias culturais, práticas e ferramentas que aumentam a capacidade de uma empresa de distribuir aplicativos e serviços em alta velocidade e, assim, otimizando e aperfeiçoando produtos em um ritmo mais rápido do que o das empresas que usam processos tradicionais de desenvolvimento de *software* e gerenciamento de infraestrutura. Essa velocidade permite que as empresas atendam melhor os seus clientes e consigam competir de modo mais eficaz no mercado” [AMAZON 2020].

No modo tradicional havia um grande problema no processo de desenvolvimento e entrega de um produto, tendo uma barreira entre as equipes de operações e de desenvolvimento, considerando que o time de operações tinha a responsabilidade de manter o ambiente estável e os desenvolvedores por outro lado constantemente estavam realizando mudanças. Com a cultura DevOps pode se dizer que não há mais uma barreira entre essas equipes, pois as duas trabalham juntas para entrega do produto [ATLASSIAN 2022], e isso envolve práticas e ferramentas de integração contínua, tais como entregas, micro serviços, testes, provisionamento e outras atividades. Assim, os desenvolvedores passam a conhecer um pouco do trabalho dos integrantes do time de operações e o time de operações conhece um pouco o trabalho do time de desenvolvimento.

### 2.4. Infraestrutura como Código

A Infraestrutura como Código (*IaC*) é o gerenciamento de infraestrutura associada às redes, máquinas virtuais, balanceadores de carga e topologia de conexão, em um modelo descritivo que usa o mesmo controle de versão que a equipe do DevOps para o código-fonte. Seguindo o princípio de que o mesmo código-fonte gera o mesmo binário, um modelo de *IaC* gera o mesmo ambiente toda vez que é aplicado. *IaC* é uma prática DevOps importante e é usada em conjunto com a entrega contínua [MICROSOFT 2021].

Enfatiza-se que muitos problemas são resolvidos com a prática da *IaC* [RED HAT 2020], afinal com ela o ambiente é fornecido de forma mais rápida e estável, tornando o processo mais seguro e evitando configurações manuais e lentas, além de toda configuração do ambiente ser escrita de código.

### 2.5. Ansible

O Ansible é um mecanismo *opensource* de automação de TI para provisionamento, gerenciamento de configurações, implantação de aplicações, orquestração e muitos outros processos da área [RED HAT 2021]. As tarefas são descritas em arquivos chamados de *playbook* utilizando formato de codificação de dados YAML<sup>3</sup>.

É descrito como uma ferramenta de gerenciamento, e tipicamente mencionado junto com outras ferramentas como Chef, Puppet e Salt. Quando é citado sobre gerenciamento de

---

<sup>3</sup> É um formato de serialização de dados legível por humanos e inspirado em linguagens, como C, Python e Perl. Ele é utilizado com a finalidade de fazer o armazenamento das informações.

configuração, tipicamente está associado à escrita de estado de servidores, ou seja, assegurar que de fato os servidores estão de acordo com o estado descrito, como se os pacotes corretos estão instalados, se as configurações de arquivos estão com os valores e as permissões esperadas, se o serviço correto está rodando, entre outras ações. Como outra ferramenta de gerenciamento de configurações, Ansible expõe uma linguagem específica de domínio *DSL*<sup>4</sup> usada para descrever o estado dos seus servidores [HOCHSTEIN e MOSER 2014].

## 2.6. Terraform

Terraform é uma ferramenta de infraestrutura como código (*IaC*) que permite construir, alterar e versionar infraestrutura com segurança e eficiência. Isso inclui componentes de baixo nível, como instâncias<sup>5</sup> de computação, armazenamento e rede, bem como componentes de alto nível, como entradas de *DNS* e recursos de *SaaS*. O Terraform pode gerenciar provedores de serviços existentes e soluções internas personalizadas [TERRAFORM 2021].

## 2.7. Git

Git é um sistema de controle de versão com distribuição gratuita e de código aberto para gerenciar projetos com velocidade e eficiência [GIT 2021]. Como apresentam os autores [Chacon e Straub 2009], o objetivo dessa ferramenta é trazer como meta a velocidade em um projeto simples, com forte suporte para desenvolvimento não-linear (milhares de ramos paralelos, completamente distribuído, podendo ser capaz de lidar com projetos grandes como o núcleo do Linux e com eficiência.

## 2.8. Modelo Incremental

Conforme apresenta [Sommerville 2011], o modelo incremental parte de uma metodologia para desenvolvimento de uma aplicação baseada em incrementos, ou seja, em partes que são desenvolvidas de forma linear, a qual o usuário traz *feedback* no final de cada incremento e diante disso, o *software* evolui e gera versões até que toda a aplicação seja desenvolvida e entregue.

Diante desse contexto, as características desse modelo são:

- Pelas entregas em formato de incrementos, sendo que cada um fornece parte da funcionalidade solicitada;
- Pela versão inicial que é o núcleo do produto, a qual os requisitos de usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais;
- Pela evolução que acontece quando são inseridas novas características ao desenvolvimento, considerando as sugestões do usuário a cada incremento;
- Pelas versões que são geradas e que podem ser planejadas de modo que os riscos técnicos possam ser administrados.

## 3. Trabalhos Correlatos

Na literatura podem ser encontrados vários trabalhos que tem como proposta central o uso da ferramenta OpenStack para criação de ambiente em nuvem, a qual para este artigo são expostos três desses trabalhos.

---

<sup>4</sup> Uma linguagem específica de domínio é uma linguagem de programação com um nível mais alto de abstração otimizado para uma classe específica de problemas. Uma *DSL* usa os conceitos e regras do campo ou domínio.

<sup>5</sup> Solicitação de máquinas virtuais.

- Os autores [Raja e Rabinson 2016] traz como objetivo a criação de um ambiente *IaaS* para *cloud* híbrida. O ambiente foi construído apresentando diferentes tipos de sistema operacional e *softwares* que foram disponibilizados aos usuários sem a necessidade de requisitos de sistema ao lado do usuário.
- Os autores [Girish e Guruprasad 2014] objetivaram a construção de um ambiente de nuvem privada. Os dois também tiveram como resultado um ambiente de Computação em Nuvem construído para subir imagens, instâncias, projetos, *flavors*<sup>6</sup> e serviços.
- O autor [Rodrigues Filho 2014] teve como objetivo em seu trabalho a implementação de uma infraestrutura de Computação em Nuvem escalável para ambientes *IaaS*. A solução foi encontrada por meio da ferramenta OpenStack que foi instalada através da ferramenta DevStack. Com a utilização do OpenStack houve redução de tempo de provisionamento de recursos, como armazenamento, conectividade, serviços, entre outros.

Todas as pesquisas apresentadas visam construir um ambiente de nuvem utilizando o OpenStack, disponibilizando recursos computacionais. Considerando a proposta exposta neste artigo e os trabalhos correlatos, a diferença é que a proposta irá se basear na utilização da Infraestrutura como Código, automatizando todo o processo de instalação, configuração do OpenStack e da criação de instâncias dentro do OpenStack e, assim, provisionando o ambiente tornando todo processo mais rápido, seguro e controlado.

Quanto ao trabalho do autor [Rodrigues Filho 2014], o que difere dos demais mencionados, é que ele utiliza o DevStack para automatizar o processo de instalação. Todavia o DevStack é utilizado apenas para ambiente de desenvolvimento, onde é instalado um OpenStack básico, como um demo ou ambiente de teste, sendo que toda instalação ocorre dentro de uma única máquina. O DevStack é ideal apenas para realização de testes antes de aplicar mudanças no ambiente de produção.

## 4. Metodologia

### 4.1. Delineamento do Trabalho

Esta seção apresenta o delinear da pesquisa, as etapas que foram seguidas para atingir o objetivo do trabalho.

Primeiramente foi realizada uma pesquisa bibliográfica buscando embasamento teórico quanto aos conceitos associados a proposta, estudo de trabalhos acadêmicos similares, além do estudo das ferramentas *opensource* para alcançar o objetivo proposto. Após a escolha das ferramentas foi realizado o desenvolvimento seguindo a metodologia incremental, desenvolvendo *scripts* para automatização de cada etapa necessária para implantação dos serviços que compõem um ambiente de *cloud*. Outro referencial importante e que fundamenta essa proposta é o manual do OpenStack.

Para este trabalho foram implementados os serviços essenciais para o funcionamento do ambiente e o serviço de *block storage* e *object storage* que são opcionais. Após a construção do ambiente foi realizada a automatização de criação de instâncias. Testes foram realizados após a conclusão de cada etapa para validação.

---

<sup>6</sup> Templates de máquinas virtuais.

Toda a proposta foi implementada e testada através do sistema operacional Ubuntu versão 20.04.

## 5. Desenvolvimento

Como mencionado na metodologia, o desenvolvimento deste trabalho seguirá o modelo incremental através de nove incrementos, sendo apresentados no decorrer desta seção.

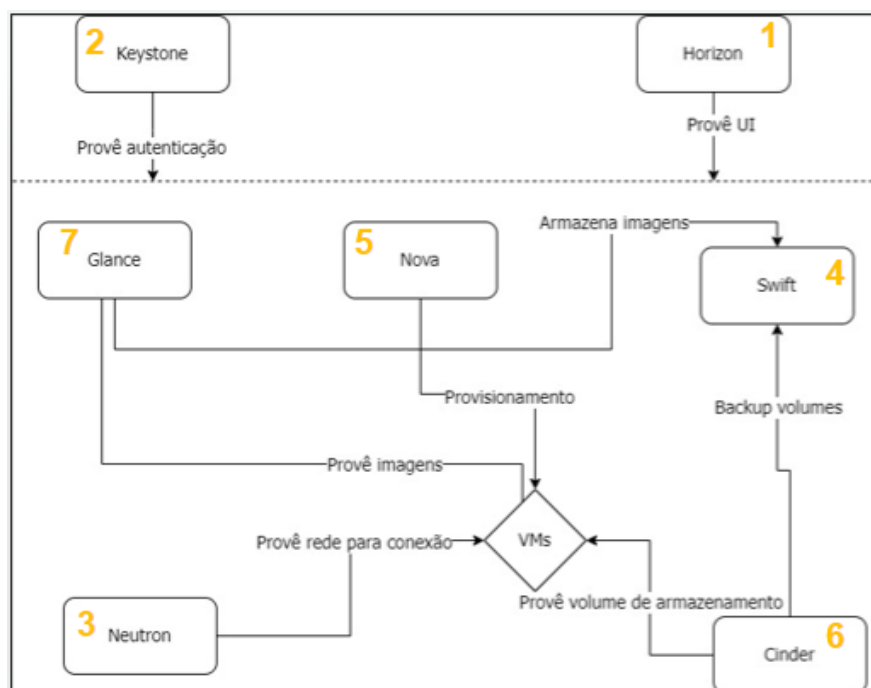
Considerando que a proposta se dá no tipo de Computação em Nuvem *IaaS*, será utilizada a plataforma OpenStack. A Tabela 1 traz a relação de serviços que foram implementados.

**Tabela 1. Serviços do OpenStack implementados**

Serviço	Função
Keystone	Serviço de identidade que provê autenticação, descoberta de serviço e autorização de multilocação distribuída.
Horizon	Serviço que provê uma interface de usuário <i>web</i> para os serviços do OpenStack.
Glance	Permite que os usuários descubra, registre e recupere imagens de máquinas virtuais.
Placement	Serviço <i>HTTP</i> para gerenciar, selecionar e reivindicar provedores de classes de inventário que representam os recursos disponíveis na <i>cloud</i> .
Nova	Realiza o gerenciamento do sistema de Computação em Nuvem.
Neutron	Serviço que permite criar e conectar dispositivos de interface gerenciados por outros serviços do OpenStack a redes.
Cinder	Serviço de armazenamento em bloco para instâncias.
Swift	Serviço de armazenamento de objetos.

Quanto a codificação e conforme abordado na metodologia, foi seguida a própria documentação do [OpenStack 2022] para realização das etapas que já estão sequenciadas na ordem de implementação. O documento traz os requisitos que fundamentam cada incremento e validação dos resultados propostos.

O diagrama de fluxo apresentado na Figura 2, traz uma visão geral da proposta.



**Figura 2. Diagrama de fluxo dos serviços**



Considerando a legenda numérica da Figura 2, foi feita uma exploração dos fluxos para melhor entendimento do diagrama.

1. O Horizon é um serviço de interface *web* que será disponibilizado ao usuário para criar, deletar e editar recursos na nuvem. Conforme mostra o diagrama, este serviço se comunica com todos os outros serviços implementados, pois o usuário irá gerenciar os recursos na nuvem pelo *dashboard*, apesar de também ser possível gerenciar por linha de comando.
2. O serviço Keystone é um dos mais importantes, todos os serviços se comunicam entre si por causa desse serviço, e com ele é possível que o usuário se autentique pela interface *web* do Horizon. Como mostra o diagrama, todos os outros serviços se comunicam com o Keystone para se autenticar. Os dados de acesso são gravados em um banco de dados MySQL.
3. O Neutron, serviço de rede do ambiente, se comunica com o Keystone para autenticação e é acessado pelo Horizon. O Neutron possui alguns componentes, como o Neutron-Server que aceita e faz o roteamento de requisições de *API* para o *plugin* de rede do OpenStack adequado para a ação. O *Queue* roteia informações entre o Neutron-Server e os *plugins*.
4. O Swift, serviço de *object storage*, é acessado pelo Horizon e se comunica com o Keystone para autenticação. O Swift possui alguns componentes, como *swift-proxy-server* que aceita requisições para *upload* de arquivos, modificação de *metadata* e criação de *containers*. O *swift-proxy-server* se comunica com *swift-account-server* que gerencia contas com armazenamento de objeto, se comunica com o *swift-container-server* que gerencia o mapeamento de *containers* ou pastas, dentro do *Object Storage* e se comunica com o *swift-object-server* que gerencia objetos reais, como arquivos, nos nós de armazenamento.
5. O Nova Compute interage com o Keystone para autenticação, com o *placement* para rastreamento e seleção de inventário de recursos, serviço *glance* para imagens de disco, e *dashboard* para interface de usuário.
6. O Cinder interage com o Keystone para autenticação e com o Horizon para interface de usuário. O *cinder-volume* interage diretamente com o serviço de block storage e processos como o *cinder-scheduler*, interage com esses processos por meio de uma fila de mensagem, o *queue*.
7. O Glance, serviço de imagens de instâncias, se comunica com o Keystone para autenticação e o Horizon para interface de usuário. O *glance-api* recebe e aceita chamadas de *API* para descoberta, recuperação e armazenamento de imagens.

### 5.1. Softwares Utilizados

Para alcançar o objetivo do trabalho foram utilizadas ferramentas *opensource*, conforme apresenta a Tabela 2.

**Tabela 2. Softwares utilizados para atingir os objetivos**

Nome	Descrição
[OpenStack 2022]	Plataforma <i>opensource</i> a ser implementada para criar e gerenciar a nuvem privada.
[Ansible 2021]	Ferramenta <i>opensource</i> a ser utilizada para automatizar todo o processo de instalação, configuração e ações necessárias para criar o ambiente.

[Terraform 2021]	Ferramenta <i>opensource</i> para criar recursos no ambiente por meio de código.
[Git 2021]	Sistema de controle de versão de arquivos a ser utilizado para controle de versão das <i>playbooks</i> e arquivos criados.

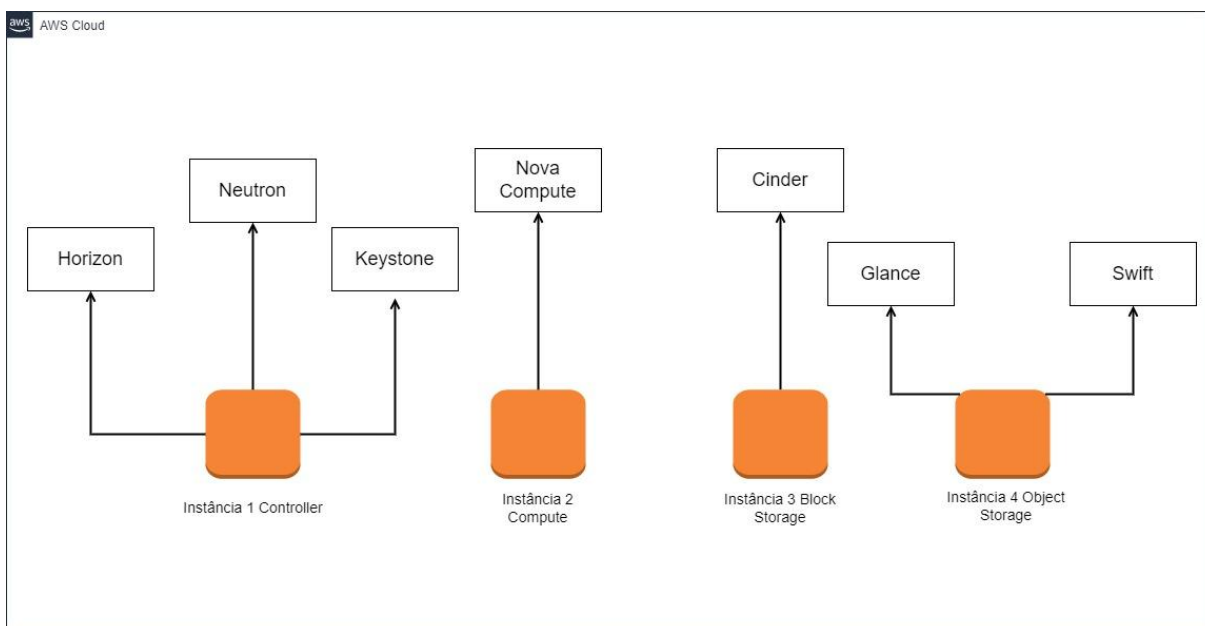
## 5.2. Hosts

Para a construção do ambiente foram utilizados quatro *hosts*, sendo: um *controller*, um *compute*, um *block storage* e um *object storage*. Foram criados na *Amazon Web Services* (AWS) para fins de teste. As características das instâncias são apresentadas na Tabela 3.

**Tabela 3. Instâncias criadas para construção do ambiente**

	Node 1	Node 2	Node 3	Node 4
<b>Hostname</b>	controller	compute	block1	object1
<b>Processador</b>	2 vCPUs, 2.3 GHz	2 vCPUs, 2.3 GHz	2 vCPUs, 2.3 GHz	2 vCPUs, 2.3 GHz
<b>Memória</b>	8 GiB	8 GiB	4 GiB	4 GiB
<b>Armazenamento</b>	30 GB	30 GB	50 GB	50 GB

O ambiente de teste criado na AWS com as 4 instâncias de máquinas virtuais é apresentado na Figura 3, considerando os serviços que foram implementados em cada instância. Enfatiza-se que todos os serviços foram explorados através da Figura 2.



**Figura 3. Diagrama ambiente de teste**

## 5.3. *Playbook*<sup>7</sup> Principal

Para implantação do OpenStack é necessário a execução de uma única *playbook* Ansible pelo usuário. Essa *playbook* chama diversas outras tarefas conhecidas pelo Ansible como “*roles*”, como apresenta a Figura 4.

<sup>7</sup> Conjunto de tarefas a serem executadas pelo Ansible escritas em um mesmo arquivo no formato YAML.

```
playbook-deploy.yml
OpenStack-Ansible-Prod > ubuntu > 01-deploy-openstack > playbook-deploy.yml
You, seconds ago | 2 authors (HailsonJunior and others)
1 --- # https://docs.openstack.org/install-guide
2 - name: Deploy OpenStack
3   hosts: 'all'
4   any_errors_fatal: true
5   become: yes
6   become_method: sudo
7   roles:
8     - { role: prepare-environment, tags: ["prepare-environment"]}
9     - { role: keystone, tags: ["keystone"]}
10    - { role: horizon, tags: ["horizon"]}
11    - { role: glance, tags: ["glance"]}
12    - { role: placement, tags: ["placement"]}
13    - { role: nova, tags: ["nova"]}
14    - { role: neutron, tags: ["neutron"]}
15    - { role: cinder, tags: ["cinder"]}
16    - { role: swift, tags: ["swift"]}
17
18 ...
```

Figura 4. *Playbook* principal

Observa-se que antes de informar quais *roles* serão necessárias, alguns campos são definidos na *playbook*, sendo: o nome da tarefa, quais *hosts* a *playbook* será executada, se o usuário deseja que o Ansible pare a execução no caso de qualquer erro que ocorra durante a execução (opcional), se a *playbook* será executada com privilégios de usuário *root*, o método de execução e por fim a relação das *roles* de acordo com a ordem desejada.

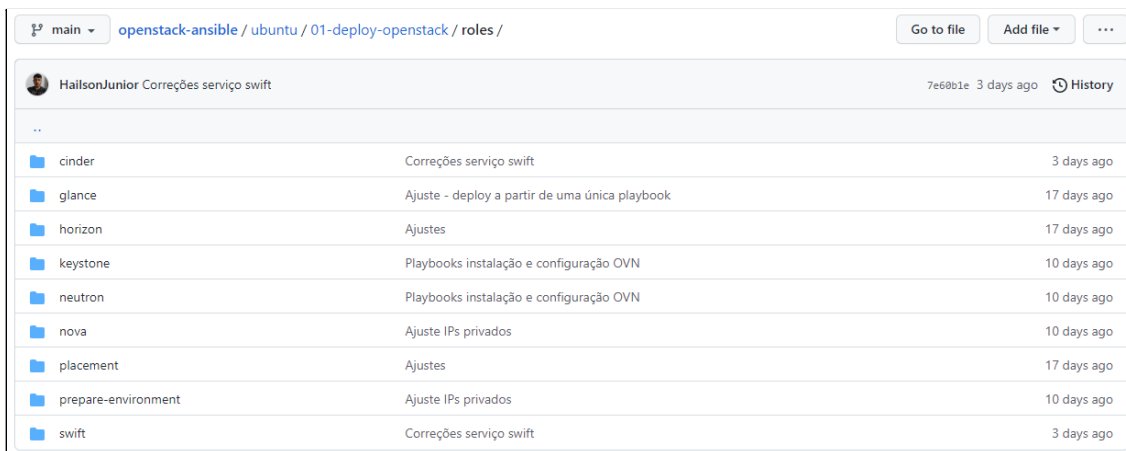
Para que o Ansible saiba em quais *hosts* serão desempenhadas suas tarefas é necessária a criação de um inventário informando o endereço desses *hosts*. É possível criar inventários de diversos formatos, utilizando variáveis, grupos e subgrupos, conforme necessário. Para essa proposta, foi utilizado um modelo simples, pois consiste em um inventário com apenas quatro *hosts*, conforme apresenta a Figura 5. Entre colchetes é informado o nome de um grupo, sendo que nesse exemplo há *controller*, *compute*, *block* e *object*. Cada grupo contém um *host* que será conhecido pelo Ansible pela variável informada antes de “*ansible\_host*” (variável que recebe o endereço *IP* da instância), como *node1*, *node2*, *node3* e *node4*. Esses são apenas identificadores, sendo o que conta realmente é o endereço *IP*. Cada grupo pode receber um ou mais *hosts*.

```
inventory.ini
OpenStack-Ansible-Prod > ubuntu > 01-deploy-openstack > inventory.ini
HailsonJunior, a week ago | 1 author (HailsonJunior)
1 [controller]
2 node1 ansible_host=172.31.82.248
3
4 [compute]
5 node2 ansible_host=172.31.87.49
6
7 [block]
8 node3 ansible_host=172.31.91.50
9
10 [object]
11 node4 ansible_host=172.31.82.229
```

Figura 5. Inventário Ansible

Para executar a *playbook* o usuário pode utilizar o seguinte comando: “*ansible-playbook -v -i inventory.ini playbook-deploy.yml*”.

Como está descrito na *playbook* principal, o Ansible buscará na pasta “*roles*”, as tarefas que foram informadas, conforme apresenta a Figura 6. As informações dessas tarefas serão explicadas com mais detalhes nas próximas subseções.



Role Name	Description	Created
..		
cinder	Correções serviço swift	3 days ago
glance	Ajuste - deploy a partir de uma única playbook	17 days ago
horizon	Ajustes	17 days ago
keystone	Playbooks instalação e configuração OVN	10 days ago
neutron	Playbooks instalação e configuração OVN	10 days ago
nova	Ajuste IPs privados	10 days ago
placement	Ajustes	17 days ago
prepare-environment	Ajuste IPs privados	10 days ago
swift	Correções serviço swift	3 days ago

**Figura 6. Relação das Roles criadas**

#### 5.4. Preparando o Ambiente

Esta subseção representa o primeiro incremento da proposta. A primeira *role* a ser executada pelo Ansible é a “prepare-environment”, que tem como função a configuração dos *hosts* para preparação do ambiente. O [Anexo 1] apresenta quais *playbooks* compõem essa *role*.

Nessa primeira etapa diversos passos foram automatizados para configuração do ambiente, sendo: 1. *update* dos *hosts*, 2. configuração do *hostname* de cada instância, 3. configuração para que os *hosts* se comuniquem entre si por meio de nome sem precisar passar *IP*, 4. instalação e configuração do Chrony (implementação do *NTP*) para sincronização de relógio, 5. adição dos pacotes do Wallaby (uma das distribuições do OpenStack), 6. instalação do Nova Compute para provisionamento de instâncias, 7. instalação e configuração do banco de dados com MariaDB, 8. instalação do Rabbitmq Server para fila de mensagens realizando operações e informações de status entre os serviços, 9. instalação do Memcached para armazenar tokens em cache, 10. instalação e configuração do serviço Etcad para armazenamento de chave-valor<sup>8</sup>.

A Figura 7 apresenta um exemplo de *playbook* Ansible criada para instalação e configuração do *etcd*.

<sup>8</sup> Modelo de armazenamento de dados que indexa os dados a uma chave.

```

HailsonJunior, há 6 meses | 1 autor (HailsonJunior)
1 ---
2 - name: Install package etcd
3   apt:
4     name: etcd
5     state: latest
6   when: "'node1' in inventory_hostname"
7   tags: install-etcd
8
9 - name: Edit etcd conf file
10  blockinfile:
11    path: /etc/default/etcd
12    block: |
13      ETCD_NAME="controller"
14      ETCD_DATA_DIR="/var/lib/etcd"
15      ETCD_INITIAL_CLUSTER_STATE="new"
16      ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
17      ETCD_INITIAL_CLUSTER="controller=http://{{ ip_controller }}:2380"
18      ETCD_INITIAL_ADVERTISE_PEER_URLS="http://{{ ip_controller }}:2380"
19      ETCD_ADVERTISE_CLIENT_URLS="http://{{ ip_controller }}:2379"
20      ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
21      ETCD_LISTEN_CLIENT_URLS="http://{{ ip_controller }}:2379"
22  when: "'node1' in inventory_hostname"
23  tags: etcd-configure
24
25 - name: Restart and enable etcd service
26  service:
27    name: etcd
28    state: restarted
29    enabled: yes
30  when: "'node1' in inventory_hostname"
31  tags: restart-etcd
32
33 ...

```

Figura 7. Playbook para instalação do etcd

Outro arquivo importante é o que se encontra no caminho “defaults/main.yml” de cada uma das *roles*. A Figura 8 apresenta esse arquivo da role *prepare environment*.

```

HailsonJunior, mês passado | 1 autor (HailsonJunior)
1 ---
2 ## Nodes IPs
3 ip_controller: 172.31.82.248
4 ip_compute: 172.31.87.49
5 ip_block: 172.31.91.50
6 ip_object: 172.31.82.229
7
8 ## Network
9 network: 172.31.0.0/16
10
11 ## NTP Server
12 NTP_SERVER: localhost
13
14 ## Passwords
15 db_root_pass: openstack
16 RABBIT_PASS: openstack

```

Figura 8. Playbook de variáveis

## 5.5. Keystone - Serviço de Identidade

Esta subseção representa o segundo incremento da proposta. O Keystone é um dos principais serviços do OpenStack, tornando possível que o usuário consiga se autenticar no ambiente, fornecendo um único ponto de integração para gerenciar a autenticação. O [Anexo 2] mostra pelo *dashboard* da plataforma, a opção *identity* do menu que tem como finalidade a criação de projetos, usuários, grupos, funções e credenciais utilizando o serviço Keystone. Já o [Anexo 3] mostra as *playbooks* criadas para implantação desse serviço.

Com essas *playbooks* foi possível automatizar a configuração do banco de dados através da criação do *database* do Keystone, dos usuários e seus privilégios, além da inserção dos seus respectivos dados. Após a configuração do *database* foi realizada a configuração do Apache informando o nome do servidor *controller* e a criação de domínio, projeto, usuário e *role* no OpenStack para validação do serviço de identidade.

## 5.6. Horizon - Serviço de Dashboard

Esta subseção representa o terceiro incremento da proposta. Após a automatização da preparação do ambiente, instalação e configuração de diversos serviços, foi realizada a criação de *playbooks* para implementação do Horizon, serviço de *dashboard* que provê uma interface *web* para visualizar e gerenciar recursos no ambiente, como apresenta o [Anexo 4].

As *playbooks* criadas realizam a instalação do serviço e da configuração. Consiste em configurar o *dashboard* a utilizar os serviços do OpenStack no *controller node*; permitir acesso dos *hosts* ao *dashboard*; configuração do serviço Memcached Storage; habilitação da versão 3 do *identity API*, habilitação de domínio de suporte múltiplo, configuração de domínio, *role* e *time zone*. O [Anexo 5] apresenta as *playbooks* que foram criadas para a realização dessas tarefas. A ordem em que serão executadas está sendo especificado na *playbook main.yml*.

## 5.7. Glance - Serviço de Imagem

Esta subseção representa o quarto incremento da proposta. Permite encontrar, registrar e recuperar imagens de máquinas virtuais. Essas imagens podem ser gravadas em diversos locais como em sistema de arquivos e sistemas de armazenamento de objetos ou *Block Storage*. As *playbooks* apresentadas no [Anexo 6] realiza a criação do banco de dados do Glance, além do usuário e do seu perfil como administrador. Também é criado o serviço e os *endpoints*, além da instalação do serviço e dos diversos ajustes em seu arquivo de configuração. Após esses passos é realizado o teste do serviço realizando o *download* de uma imagem, *upload* e listagem. O [Anexo 7] apresenta o menu de imagens do Glance pelo *dashboard*.

## 5.8. Placement - API REST

Esta subseção representa o quinto incremento da proposta. O serviço *Placement* é um *API REST* e modelo de dados utilizado para rastrear usos de provedores de recursos [OPENSTACK 2021], como um nó de computação. O *Placement* verifica a utilização de recursos de cada provedor.

O [Anexo 8] mostra as *playbooks* criadas para implementar o serviço *Placement*. Essas *playbooks* realizam o processo de criação da base de dados do serviço e sua configuração, criação do serviço no OpenStack, configuração do usuário, *endpoints*, instalação do *Placement*, demais configurações necessárias para seu funcionamento e teste.

## 5.9. Nova - Serviço de Computação

Esta subseção representa o sexto incremento da proposta. Para hospedagem e gerenciamento do sistema de Computação em Nuvem foi implementado o serviço *Nova Compute*.

O OpenStack *Compute* é uma parte muito importante para um sistema de Infraestrutura como Serviço e consiste nos seguintes componentes: nova-api service para aceitar e responder chamadas de API de computação do usuário final; nova-api-metadata service para aceitar solicitação de metadata por instâncias; nova-compute service um *daemon worker*<sup>9</sup> que cria e encerra instâncias de máquinas virtuais por meio de APIs de *hypervisor* (libvirt ou QEMU, por exemplo); nova-scheduler service que recebe solicitações de instância de máquina virtual da fila e determina em qual *host* ela será executada; nova-conductor module que faz a mediação das interações entre nova-compute e o banco de dados; nova-novncproxy daemon que provê um proxy para acessar instâncias em execução através de conexão VNC; nova-spicehtml5proxy daemon que provê um proxy para acessar instâncias em execução através de uma conexão SPICE; queue um hub central para controle de mensagens entre daemons e banco de dados SQL para armazenar estados de tempo de construção e tempo de execução de uma instância.

O [Anexo 9] apresenta as *playbooks* criadas para automatizar a instalação e configuração do Nova e todos seus componentes.

## 5.10. Neutron - Serviço de Rede

Esta subseção representa o sétimo incremento da proposta. Após automatizar a instalação e configuração do serviço de computação, foram criadas as *playbooks* para o serviço de rede do ambiente, o Neutron. Assim, possibilitará a criação de rede, sub-rede, roteadores, conexão de dispositivos de interface gerenciados por outros serviços OpenStack às redes.

A implementação do Neutron inclui os seguintes componentes: neutron-server que aceita e roteia solicitações de API para *plug-in* OpenStack *Networking* apropriado para a ação; OpenStack *Networking plug-ins and agents* que conecta e desconecta portas, cria redes ou sub-redes e fornece endereços IP; *Messaging queue* usado para rotear informações entre o servidor neutron e vários *agents*. O Neutron se comunica com o OpenStack *Compute* para prover rede e conectividade para as instâncias.

O [Anexo 10] apresenta as *playbooks* criadas para automatizar a implementação do serviço Neutron e o [Anexo 11] apresenta pelo *dashboard* a utilização do serviço.

## 5.11. Cinder - Serviço de Block Storage

Esta subseção representa o oitavo incremento da proposta. Para o serviço de *Block Storage* foi utilizado o serviço Cinder, que fornece dispositivos de armazenamento em bloco para as instâncias do ambiente.

O serviço de *Block Storage* consiste nos seguintes componentes: cinder-api que aceita solicitações de API e as encaminha para o volume cinder para realizar ação; cinder-volume que interage diretamente com o serviço *Block Storage*; cinder-scheduler daemon para selecionar o nó do provedor de armazenamento ideal para criar o volume; cinder-backup daemon para prover backup de volumes de qualquer tipo para um provedor de armazenamento de backup; *Messaging queue* para rotear informações entre processos *Block Storage*.

---

<sup>9</sup> Um programa de computador que executa como um processo em plano de fundo.

O [Anexo 12] mostra as *playbooks* criadas para automatizar a instalação e configuração do serviço e o [Anexo 13] apresenta a utilização do serviço pelo *dashboard*.

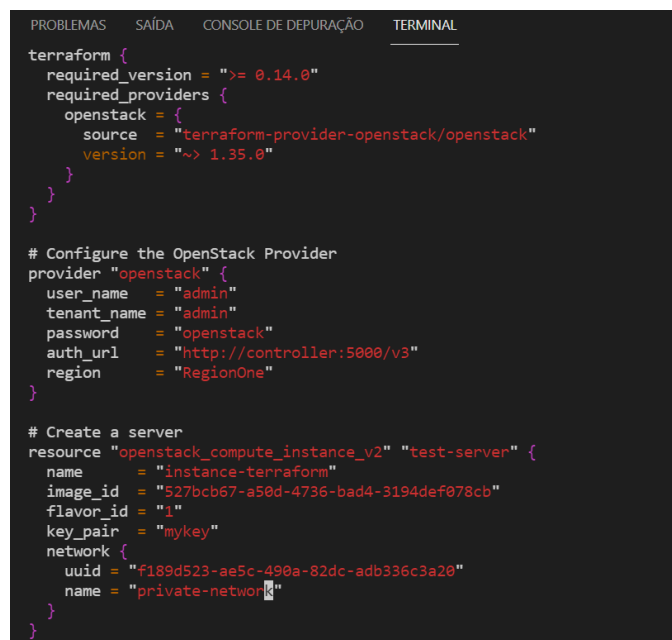
## 5.12. Swift- Serviço de Object Storage

Esta subseção representa o nono incremento da proposta. Para armazenamento de objetos foi implementado o serviço Swift que consiste nos seguintes componentes: *swift-proxy-server* que aceita solicitações *HTTP* para fazer *upload* de arquivos; modificar metadados e criar contêineres<sup>10</sup>; *swift-container-server* para gerenciamento e mapeamento de contêineres ou pastas dentro do *Object Storage*; *swift-object-server* para gerenciar objetos atuais nos *nodes storage*; *WSGI middleware* que lida com autenticação; *swift client* que permite que os usuários enviem comandos à *API REST* por meio de um *client* de linha de comando autorizado; *swift-init script* que inicializa a construção do arquivo *ring* pegando nomes de *daemons* como parâmetros e oferece comandos; *swift-recon* uma ferramenta *CLI*<sup>11</sup> usada para recuperar várias métricas e informações de telemetria sobre um *cluster* que foi coletado pelo *middleware* *swift-recon* e *swift-ring-builder* utilitário de construção e reequilíbrio do anel de armazenamento.

O [Anexo 14] mostra as *playbooks* criadas para automatizar a instalação e configuração do Swift e o [Anexo 15] mostra a utilização do serviço pelo *dashboard*.

## 5.13. Validação - criando instância com Terraform

Para criar instâncias no ambiente utilizando infraestrutura como código e validação, foi escrito o código apresentado na Figura 9 que se conecta ao ambiente e realiza a criação de uma instância como o nome “instance-terraform”. A Figura 10 mostra pelo *dashboard* os *logs* dessa instância, mostrando que ela foi criada.



```
PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL
terraform {
  required_version = ">= 0.14.0"
  required_providers {
    openstack = {
      source = "terraform-provider-openstack/openstack"
      version = "~> 1.35.0"
    }
  }
}

# Configure the OpenStack Provider
provider "openstack" {
  user_name = "admin"
  tenant_name = "admin"
  password = "openstack"
  auth_url = "http://controller:5000/v3"
  region = "RegionOne"
}

# Create a server
resource "openstack_compute_instance_v2" "test-server" {
  name = "instance-terraform"
  image_id = "527bcb67-a50d-4736-bad4-3194def078cb"
  flavor_id = "1"
  key_pair = "mykey"
  network {
    uuid = "f189d523-ae5c-490a-82dc-adb336c3a20"
    name = "private-network"
  }
}
```

Figura 9. Código Terraform para criar máquinas virtuais

<sup>10</sup> Pastas para armazenamento de objetos.

<sup>11</sup> Fazendo uma comparação entre ferramenta CLI e GUI, a CLI permite que os usuários digitem o comando manual para executar a tarefa desejada, enquanto nos usuários da GUI os recursos visuais interagem com o sistema operacional, como botões, ícones, imagens, entre outros..



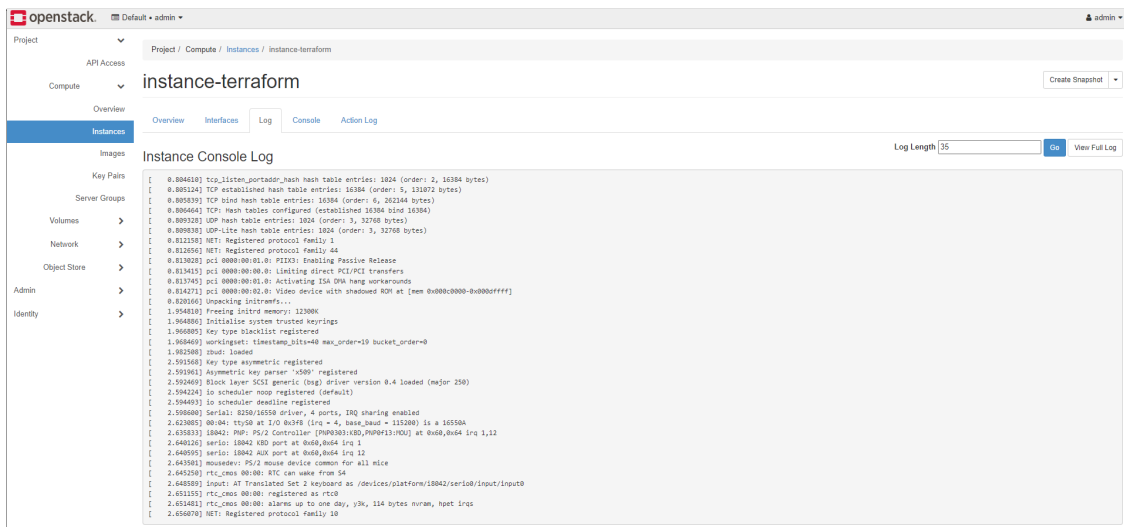


Figura 10. Logs da instância

## 6. Conclusão e trabalhos futuros

Os objetivos deste trabalho foram alcançados, considerando os testes apresentados na subseção 5.13, já que foi possível realizar todas as tarefas necessárias para construção do ambiente utilizando Infraestrutura como Código. Foi implementado um modelo de Computação em Nuvem *IaaS*, fornecendo infraestrutura como serviço, baseando-se no OpenStack.

Com esta solução é possível a construção do ambiente com maior velocidade, já que a execução da *playbook* levou 16 minutos para terminar comparado a horas de trabalho que levaria em um processo manual. Outro aspecto positivo é o maior controle e segurança quanto ao mesmo processo executado manualmente, afinal a automatização traz a probabilidade de zerar o número de erros humanos na execução de cada passo, além da diminuição da complexidade. E esse pode ser destacado como um diferencial dos trabalhos correlatos apresentados nesse artigo. É importante enfatizar que o tempo calculado foi definido através da hora inicial e a hora final da execução.

Para interessados, o trabalho está sendo disponibilizado na plataforma GitHub através do link <https://github.com/HailsonJunior/openstack-ansible>.

Para o desenvolvimento do trabalho foi fundamental o conteúdo das seguintes disciplinas do curso: Algoritmos de Programação, Arquitetura de Computadores, Sistemas Operacionais, Redes de Computadores, Serviços de Rede e Engenharia de *Software*.

Como trabalho futuro propõe-se a implementação do projeto em um ambiente real, afinal a proposta se deu na execução de um ambiente de teste, com serviços limitados. Uma segunda proposta seria o desenvolvimento de *playbooks* para implementação dos outros serviços do OpenStack, além do desenvolvimento das mesmas *playbooks* para outros sistemas operacionais e teste de outras ferramentas diferentes do OpenStack, como por exemplo, OpenShift.

## Referências

Aishwarya S., Abdul Q., Vijayakumar V. Era of Cloud Computing: A New Insight To Hybrid Cloud, Elsevier B.V, 2011. Disponível em:

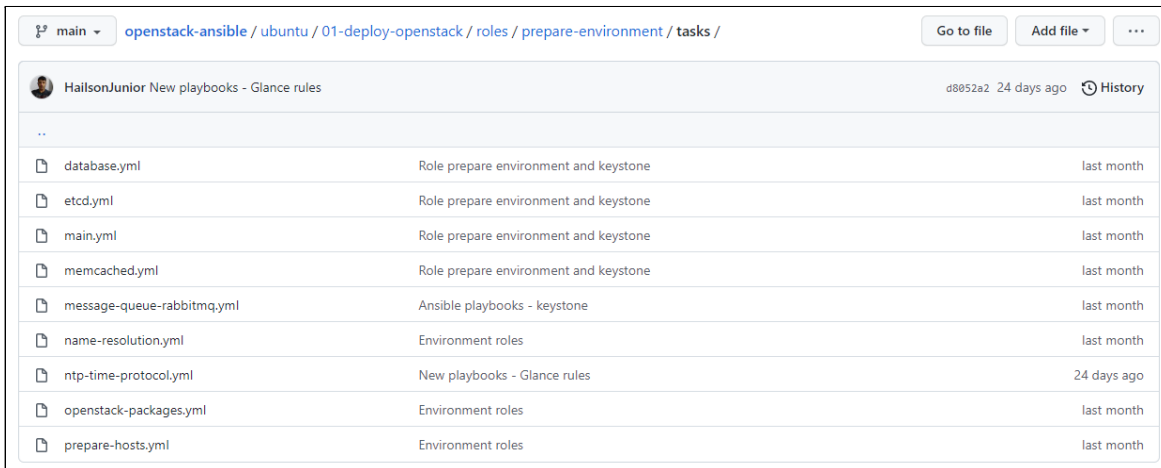
- <<https://www.sciencedirect.com/science/article/pii/S1877050915005608>>. Acesso em: 23 ago. 2021.
- Amazon. (2020) O que é o DevOps. Disponível em:  
<<https://aws.amazon.com/pt/devops/what-is-devops/>>. Acesso em: 06 out. 2021.
- Atlassian. (2022) O que é DevOps. Disponível em: < <https://www.atlassian.com/br/devops>>.  
Acesso em: 11 jan. 2022.
- AWS. (2022) O que é cloud computing. Disponível em:  
<<https://aws.amazon.com/pt/what-is-cloud-computing/>>. Acesso em: 11 jan. 2022.
- Chacon, S.; Straub, B. (2009) DUALTEC. Entendendo as camadas do *cloud computing*. Disponível em:  
<[http://lt.idg.com.br/Dualtec/Dualtec\\_whitepaper\\_as\\_camadas\\_da\\_nuvem.pdf](http://lt.idg.com.br/Dualtec/Dualtec_whitepaper_as_camadas_da_nuvem.pdf)>. Acesso em: 23 ago. 2021.
- Gleb, B. (2017) Choosing the Right Cloud Service: IaaS, PaaS, or SaaS. Disponível em:  
<<https://rubygarage.org/blog/iaas-vs-paas-vs-saas>>. Acesso em: 16 abr. 2019.
- Git. (2021) Git Fast Version Control. Disponível em: <<https://git-scm.com>>. Acesso em: 06 out. 2021.
- Guruprasad H. S.; Girish L. S. (2014) Building Private Cloud using OpenStack, International Journal of engineering research & technology (IJERT) Volume 3, Issue 3, Mai – Jun 2014. Disponível em:  
<[https://www.researchgate.net/publication/274388772\\_Building\\_Private\\_Cloud\\_using\\_OpenStack](https://www.researchgate.net/publication/274388772_Building_Private_Cloud_using_OpenStack)>. Acesso em: 20 ago. 2021.
- Hochstein, L.; Moser, R. (2014) Ansible Up & Running: Automating Configuration Management and Deployment the Easy Way. 2ª edição. United State of America: O'Reilly Media.
- Jansen W., Grance T. (2011) Guidelines on Security and Privacy in Public Cloud Computing, National Institute of Standards and Technology, dezembro 2011. Disponível em:  
<<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-144.pdf>>. Acesso em: 23 ago. 2021.
- Mell P. (NIST), Grance T. (NIST). (2011) The NIST Definition of Cloud Computing. SP 800-145. Disponível em: < <https://csrc.nist.gov/publications/detail/sp/800-145/final> >.  
Acesso em: 20 ago. 2021.
- Microsoft. (2021) O que é a Infraestrutura como Código. Disponível em:<<https://docs.microsoft.com/pt-br/devops/deliver/what-is-infrastructure-as-code/>>.  
Acesso em: 06 out. 2021.
- Raja, B. J., Rabinson, V. K. (2016) IaaS for Private and Public Cloud using Openstack, International Journal of engineering research & technology (IJERT) Volume 05, Issue 04 (abril 2016). Disponível em: <<http://dx.doi.org/10.17577/IJERTV5IS040191>>. Acesso em: 20 ago. 2021.

- Shimabukuro, I. (2020) Com a pandemia de coronavírus, o mercado global de *IaaS* cresceu quase 50% em 2020. Disponível em:  
<<https://olhardigital.com.br/2021/06/30/pro/mercado-global-de-iaas-cresceu-quase-50-por-cento-em-2020/>>. Acesso em 10 dez. 2021.
- OpenStack. (2021) What is OpenStack. Disponível em:  
<<https://www.openstack.org/software>>. Acesso em: 05 out. 2021.
- OpenStack. (2022) OpenStack Install Guide. Disponível em:  
<<https://docs.openstack.org/install-guide>>. Acesso em 05 dez. 2021.
- Red Hat. (2021) Noções básicas do Ansible. Disponível em:  
<<https://www.redhat.com/pt-br/topics/automation/learning-ansible-tutorial> >. Acesso em: 06 out. 2021.
- Red Hat. (2020) O que é Infraestrutura como Código. Disponível em:  
<<https://www.redhat.com/pt-br/topics/automation/what-is-infrastructure-as-code-iac#vantagens-da-iac>>. Acesso em 11 jan. 2022.
- Rodrigues Filho, J. S. (2014) Implantação de uma infraestrutura de Computação em Nuvem *IAAS* com openstack, Faculdade Antonio Meneghetti. Disponível em:  
<<http://repositorio.faculdadeam.edu.br/xmlui/handle/123456789/96>>. Acesso em: 20 ago. 2021.
- Sommerville, I. (2011). "Engenharia de Software". Editora Pearson, 9ª Edição.
- Terraform. (2021) Introduction to Terraform. Disponível em:  
<<https://www.terraform.io/intro/index.html>>. Acesso em: 06 out. 2021.
- VMware. (2021) Nuvem privada. Disponível em:  
<<https://www.vmware.com/br/topics/glossary/content/private-cloud.html> >. Acesso em: 26 nov. 2021.

# ANEXO I - Serviços OpenStack

## 1. Preparação do ambiente

A *playbooks* da role de preparação de ambiente farão as primeiras configurações necessárias para construir o ambiente, alterando *hostname* das máquinas, instalação de banco de dados e outros serviços.



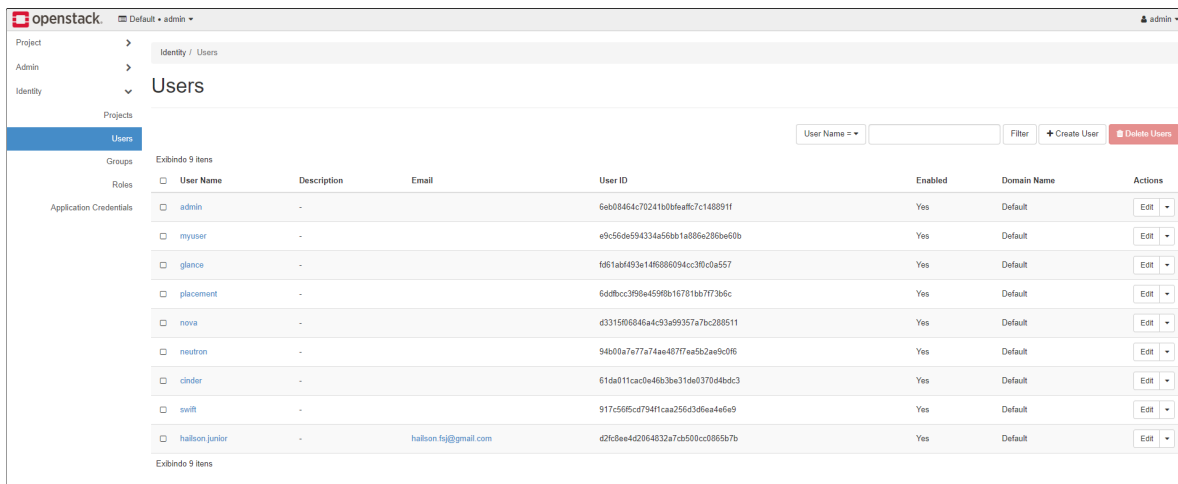
The screenshot shows the OpenStack Ansible roles page. The breadcrumb navigation is: main > openstack-ansible / ubuntu / 01-deploy-openstack / roles / prepare-environment / tasks / . There are buttons for 'Go to file', 'Add file', and '...'. The page title is 'HailsonJunior New playbooks - Glance rules' with a timestamp 'd8852a2 24 days ago' and a 'History' link. Below the title is a table listing roles:

Role Name	Description	Last Modified
database.yml	Role prepare environment and keystone	last month
etcd.yml	Role prepare environment and keystone	last month
main.yml	Role prepare environment and keystone	last month
memcached.yml	Role prepare environment and keystone	last month
message-queue-rabbitmq.yml	Ansible playbooks - keystone	last month
name-resolution.yml	Environment roles	last month
ntp-time-protocol.yml	New playbooks - Glance rules	24 days ago
openstack-packages.yml	Environment roles	last month
prepare-hosts.yml	Environment roles	last month

Figura 1. Role – Preparando ambiente

## 2. Serviço de identidade – Keystone

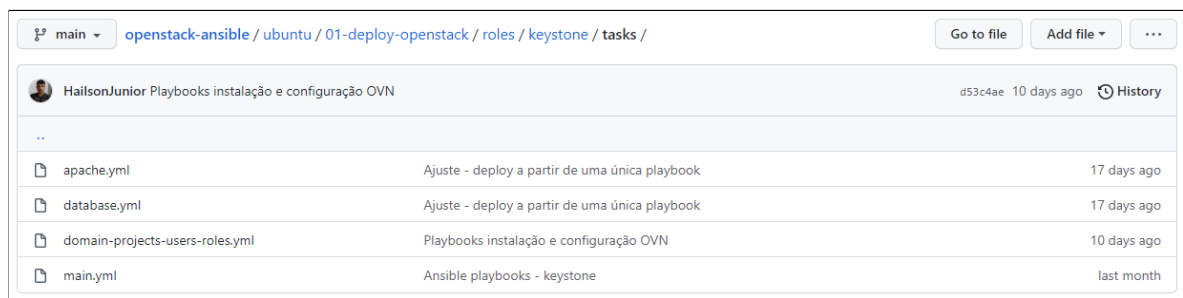
A role keystone possui as *playbooks* necessárias para instalar e configurar o serviço de identidade que servirá para autenticação no ambiente.



The screenshot shows the OpenStack Identity Users dashboard. The breadcrumb navigation is: openstack > Default > admin > Identity / Users. The page title is 'Users'. There are search and filter options: 'User Name =', 'Filter', '+ Create User', and 'Delete Users'. Below the title is a table listing users:

User Name	Description	Email	User ID	Enabled	Domain Name	Actions
admin	-	-	6eb08464c70241b0bfeaf67c148891f	Yes	Default	Edit
myuser	-	-	e9c56de594334a56bb1a886a288be60b	Yes	Default	Edit
glance	-	-	f651abf493e1485888094cc380c0a557	Yes	Default	Edit
placement	-	-	6d0b0cc3f98a4598b16781bb7773b6c	Yes	Default	Edit
nova	-	-	d3315f06846a4c93a99957a7bc280511	Yes	Default	Edit
neutron	-	-	94b00a7e77a74ae4877ef5e2ae9c0f6	Yes	Default	Edit
clinder	-	-	61da011cac0e46b3e314de037044bdc3	Yes	Default	Edit
swift	-	-	917c56f5cd794f1caa256d3d8ea4e6e9	Yes	Default	Edit
hailson junior	-	hailson.fsj@gmail.com	d2f68ea4d2064832a7cb500cc0865b7b	Yes	Default	Edit

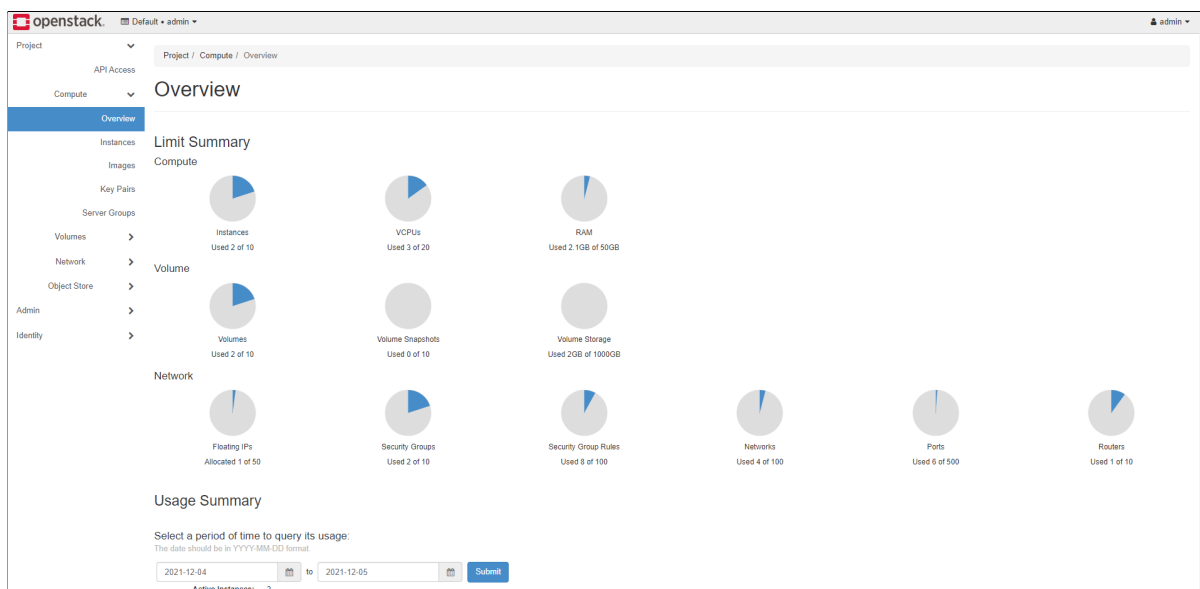
Figura 2. Dashboard – Serviço de Identidade



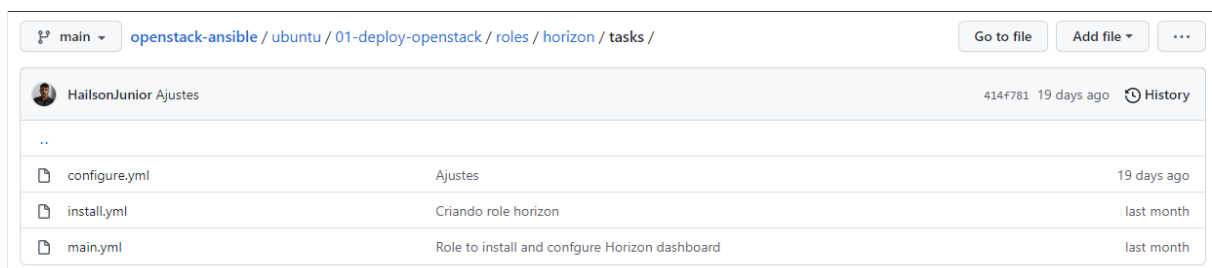
**Figura 3. Role – Serviço de Identidade**

### **3. Serviço de *dashboard* – Horizon**

O serviço horizon foi implementado para prover uma interface *web* ao usuário para facilitar o gerenciamento de recursos.



**Figura 4. Horizon – Dashboard**



**Figura 5. Role – Horizon**

### **4. Serviço de imagem – Glance**

A role Glance possui as *playbooks* que serão utilizadas para implementar o serviço de imagens de instâncias.

Task Name	Description	Count	Time
HailsonJunior Ajuste - deploy a partir de uma única playbook			
488694		19 days ago	History
..			
database.yml	Ajuste - deploy a partir de uma única playbook		19 days ago
install-and-configure.yml	Ajuste - deploy a partir de uma única playbook		19 days ago
main.yml	New playbooks - Glance rules		26 days ago
service-credentials.yml	Ajuste - deploy a partir de uma única playbook		19 days ago
verify-glance-service.yml	Ajuste - deploy a partir de uma única playbook		19 days ago

**Figura 6. Role – Glance**

Proprietário	Nome	Tipo	Status	Visibilidade	Protegido	Formato de Disco	Tamanho
admin	cirros	Image	Ativo	Público	Não	QCOW2	12.13 MB

**Figura 7. Dashboard – Glance**

## **5. Serviço de rastreamento de provedores – Placement**

As *playbooks* da *role* placement implementarão o Placement para por meio dele fazer o rastreamento dos provedores de recursos da *cloud*, rastreando o estoque e o uso de cada provedor.

Task Name	Description	Count	Time
HailsonJunior Ajustes			
414781		22 days ago	History
..			
database.yml	Ajuste - deploy a partir de uma única playbook		22 days ago
install-and-configure.yml	Ajustes		22 days ago
main.yml	Placement role		26 days ago
user-and-endpoints.yml	Ajustes		22 days ago
verify-placement-service.yml	Ajustes		22 days ago

**Figura 8. Role – Placement**

## **6. Serviço de hospedagem e gerenciamento de nuvem – Nova**

Por meio das *playbooks* apresentadas na Figura 9 será implementado o serviço de computação, o Nova, que faz a hospedagem e o gerenciamento do ambiente.

main ▾ openstack-ansible / ubuntu / 01-deploy-openstack / roles / nova / tasks /			Go to file	Add file ▾	...
HailsonJunior Correcoes		078ea56	yesterday	History	
..					
compute-finalize-installation.yml	Compute Nova role	25 days ago			
compute-install-and-configure.yml	Correcoes	yesterday			
controller-database.yml	Compute Nova role	25 days ago			
controller-finalize-installation.yml	Compute Nova role	25 days ago			
controller-install-and-configure.yml	Correcoes	yesterday			
controller-service-credentials.yml	Compute Nova role	25 days ago			
main.yml	Compute Nova role	25 days ago			
verify-operation.yml	Compute Nova role	25 days ago			

**Figura 9. Role – Nova**

## 7. Serviço de rede – Neutron

As *playbooks* da *role* Neutron da Figura 10 realizam os passos necessários para implementação do Neutron, serviço de rede.

main ▾ openstack-ansible / ubuntu / 01-deploy-openstack / roles / neutron / tasks /			Go to file	Add file ▾	...
HailsonJunior Ajustes		8a26a8d	12 hours ago	History	
..					
compute-finalize-instalation.yml	Role neutron service	22 days ago			
compute-install-and-configure.yml	Role neutron service	22 days ago			
compute-ovn.yml	Playbooks instalação e configuração OVN	15 days ago			
compute-self-service-networks.yml	Role neutron service	22 days ago			
controller-database.yml	Role neutron service	22 days ago			
controller-finalize-instalation.yml	Role neutron service	22 days ago			
controller-ovn.yml	Playbooks instalação e configuração OVN	15 days ago			
controller-self-service-networks.yml	Correcoes	yesterday			
controller-service-credentials.yml	Role neutron service	22 days ago			
main.yml	Ajustes	12 hours ago			
verify-operation.yml	Role neutron service	22 days ago			

**Figura 10. Role – Neutron**

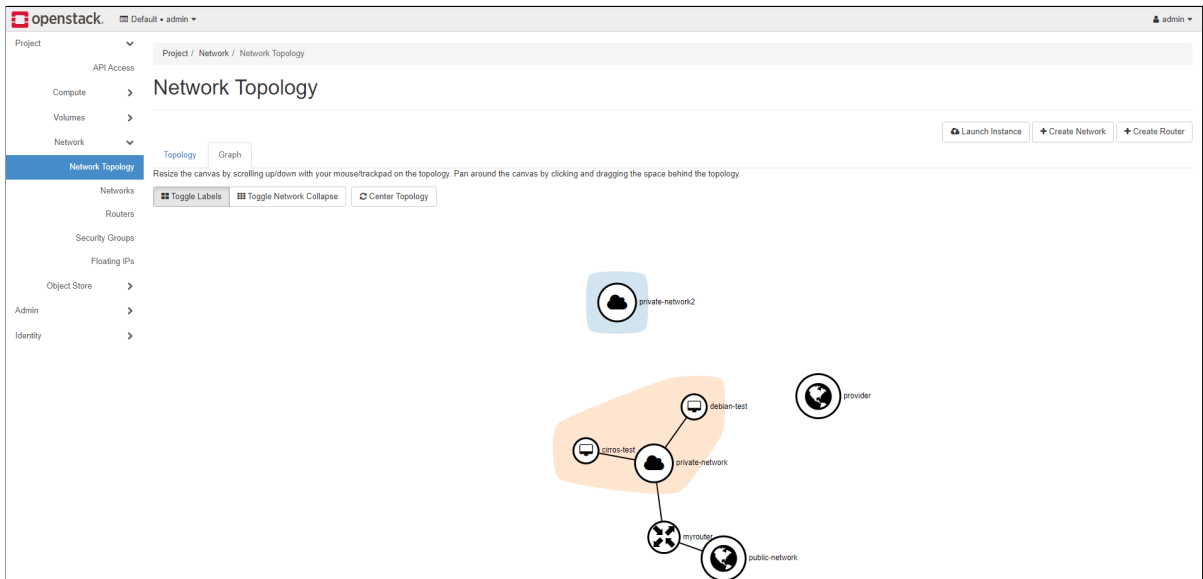


Figura 11. Dashboard – Neutron

## 8. Serviço de Block Storage – Cinder

Para armazenamento em bloco, a *role* cinder faz a implementação do serviço Cinder que permite esse tipo de armazenamento.

Name	Role	Date
controller-database.yml	Role block e object storage	15 days ago
controller-install-and-configure.yml	Role block e object storage	15 days ago
controller-service-credentials.yml	Role block e object storage	15 days ago
main.yml	Role block e object storage	15 days ago
storage-install-and-configure.yml	Correções serviço swift	8 days ago

Figura 12. Role – Cinder

Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
myvolume	-	1GiB	Available	-	__DEFAULT__		nova	No	No	Edit Volume
volume1	-	1GiB	Available	-	__DEFAULT__		nova	Yes	No	Edit Volume

Figura 13. Dashboard – Cinder

## 9. Serviço de Object Storage – Swift

As *playbooks* da Figura 14 implementam o *Swift*, serviço de armazenamento de objetos, como imagens, vídeos, arquivo de texto, entre outros.



main ▾ openstack-ansible / ubuntu / 01-deploy-openstack / roles / swift / tasks / Go to file Add file ▾ ...

HailsonJunior Ajustes 8a26a8d 12 hours ago History

..

controller-create-and-distribute-initial-rings.yml	Role block e object storage	15 days ago
controller-proxy-server.yml	Correções serviço swift	8 days ago
controller-service-credentials.yml	Role block e object storage	15 days ago
finalize-installation.yml	Correcoes	yesterday
main.yml	Ajustes	12 hours ago
object-install-and-conf-components.yml	Correções serviço swift	8 days ago
object-prerequisites.yml	Role block e object storage	15 days ago
storage-cinder-backup.yml	Role block e object storage	15 days ago
swift-with-keystone.yml	Role block e object storage	15 days ago
verify.yml	Ajustes	12 hours ago

**Figura 14. Role – Swift**

openstack. Default • admin ▾ admin ▾

Project / Object Store / Containers

## Containers

+ Contêiner

container1

Exibindo 2 itens

Nome	Tamanho
Pasta	
Artigo - Hailson.docx	1.75 MB

Exibindo 2 itens

container1

- Contagem de Objetos: 2
- Tamanho: 1.75 MB
- Data Criada: Dec 5, 2021
- Storage Policy: Policy-0
- Acesso Público: Desativado

**Figura 15. Dashboard – Swift**