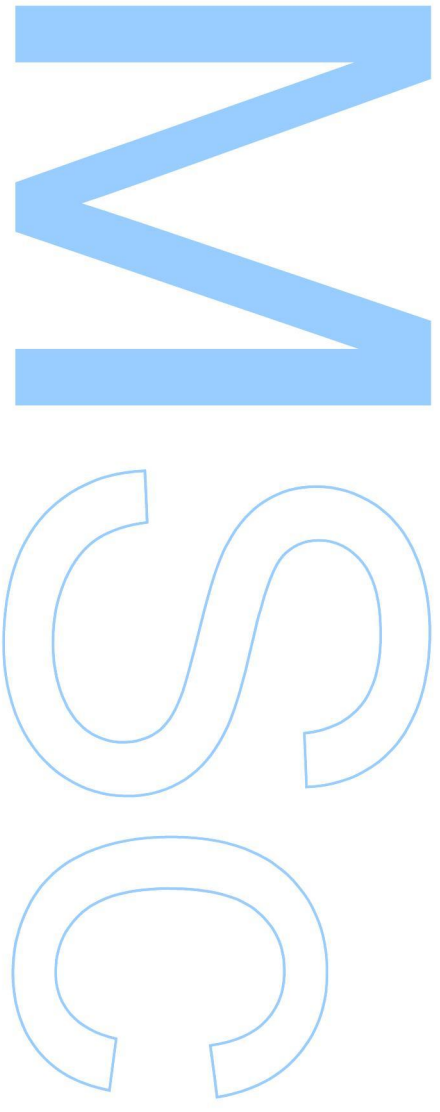
Infraestrutura Confiável para Cloud baseada em OpenStack

Duarte Miguel Petiz Ferreira
Mestrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2017

Orientador
Rolando Martins, Prof. Auxiliar Convidado, FCUP

Coorientador
Paulo Cavaleiro, Técnico Superior, FCUP



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

The Cloud Computing is a relatively new area in IT that uses recent advances in system design to offer state-of-the-art Cloud infrastructures. OpenStack is one of the most promising open source solutions in the world of cloud computing for Infrastructure as a Service (IaaS).

The main goal for this thesis is to conduct a comprehensive study of OpenStack in order to assess its ability to support a private cloud. More specially, this work intends to study, adopt and propose a high availability installation of this computational paradigm using OpenStack, by the Department of Computer Science, Faculty of Sciences of the University of Porto.

In a first phase, a study of the existing software solutions for the deployment of the Cloud IaaS model will be provided, including a comparative table with the prices of paid solutions and a comparative table among the technologies considered.

In a second phase, a high-availability experimental system cloud infrastructure will be deployed to support INESC-TEC computing resources. This testbed will be hosted in Computer Science Department of the Faculty of Sciences, University of Porto.

Resumo

A Cloud Computing é uma área relativamente nova em tenologias da informação (T.I.) que utiliza os mais recentes avanços no design de sistemas, oferecendo assim tecnologia de ponta para Infraestruturas de cloud. O OpenStack é uma das soluções open-source mais promissoras no mundo da Cloud Computing para Infraestrutura como Serviço (IaaS).

O principal objetivo desta tese é a realização de um estudo detalhado do OpenStack, com o objetivo de avaliar a sua capacidade para suportar uma cloud privada. Mais concretamente, este trabalho pretende estudar e propor um modelo de instalação de alta disponibilidade deste paradigma computacional, para o Departamento de Ciência da Computação da Faculdade de Ciências da Universidade do Porto.

Numa primeira fase será fornecido um estudo das soluções de software existentes para a implementação de um modelo de uma Infraestrutura como Serviço de Cloud, incluindo uma tabela com o preçário para as soluções comerciais, uma tabela comparativa entre as tecnologias consideradas.

Numa segunda fase, será implementada uma infraestrutura como serviço experimental de cloud, em alta disponibilidade, servindo de suporte aos recursos disponíveis para computação do INESC-TEC. Esta implementação será alojada no Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto.

Agradecimentos

Ao Professor Rolando Martins e ao Engenheiro Paulo Cavaleiro por todo o apoio que me deram para a elaboração desta tese. Ao meu grande amigo Alberto por toda a força e motivação que fez com que nunca baixasse os braços. À Marta por me apoiar em todos os momentos da minha vida. À minha Mãe por todo o esforço que fez para que eu me tornasse na pessoa que sou hoje. À Sofia e à Carlotinha por me fazerem perceber que pequenas coisas têm grande valor.

Obrigado

À minha mãe

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

API Application programming interface

AWS Amazon Web Service

CC Cluster Controller

CLC Cloud Controller

HDD Hard Disk Drive

IaaS Infraestruture as a Service

MoM Message Oriented Middleware

NC Node Controller

PaaS Plataform as a Service

P2S Point-To-Site

SaaS Software as a Service

SC Storage Controller

SDN Software-Defined Networking

SSD Solid State Drive

T.I. Tecnologias da Informação

VM Virtual Machine

VNIC Virtual Network Interface Controller

VPN Virtual Private Network

VRRP Virtual Router Redundancy Protocol

WC Walrus Storage

Capítulo 1

Introdução

Cada vez mais as tecnologias da informação (T.I.) têm vindo a herdar um papel central no seio das organizações, fazendo com que estas sejam praticamente dependentes das T.I., pelo que as infraestruturas informáticas tornaram-se como um ponto fulcral em qualquer organização para que esta prossiga com o seu correto funcionamento e desempenho.

Esta dependência das T.I., a par de reduzidos orçamentos disponíveis para investir em recursos e de mau aproveitamento dos recursos já por si disponíveis, originou um aumento da procura de soluções de consolidação dos recursos já existentes. Sendo assim, na procura de uma solução para a consolidação e correto aproveitamento de recursos anteriormente descrita surge uma solução: a adoção do modelo de Cloud Computing, seja esta do tipo privada, pública ou híbrida, de acordo com a preferência, recursos e orçamento disponíveis pela organização. O OpenStack [17] é o projecto open-source com mais destaque neste domínio.

Esta tese tem como objetivo estudar o funcionamento e infraestrutura necessária para a utilização do software OpenStack em alta disponibilidade e propor uma implementação da mesma para uma Cloud Privada com serviços redundantes, que servirá como base na Infraestrutura construída para o INESC TEC, alojada no *datacenter* do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto.

1.1 Cloud Computing

O termo *Cloud Computing* tem várias definições, sendo que a mais utilizada é a proposta pelo NIST (National Institute of Standarts and Technology) em 2011. Cloud Computing pode ser entendido como um agregado de diferentes tecnologias, como por exemplo, virtualização, serviços web, computação em grid, armazenamento, base de dados. A Cloud deve ser: 1) *On-demand self service*, cada utilizador pode alocar instâncias sem a intervenção do provedor do serviço; 2) *Broad network access*, é possível aceder às capacidades da Cloud a partir de diferentes plataformas, como telemóveis ou estações de trabalho, mesmo estas estando fora da rede local. No entanto as

empresas que disponibilizem este acesso têm que lidar com questões de segurança diferentes uma vez que podem sofrer ataques vindos do exterior; 3)*Resource pooling*, os recursos que o provedor do serviço possui são disponibilizados a diferentes utilizadores sendo que são reservados e libertados consoante a necessidade momentânea de cada utilizador; 4)*Rapid Elasticity*, é necessário que os recursos sejam elásticos para que possam ser rapidamente redimensionados consoante o utilizador necessite de ampliar, ou reduzir os mesmos recursos; 5)*Measured service*, os sistemas da Cloud controlam e otimizam a utilização de recursos. A monitorização é muito importante numa Cloud pois graças a esta é possível tomar decisões como por exemplo se os recursos alocados são os suficientes. É também importante quando se pretendem definir métricas para cada projeto, como por exemplo a duração da alocação dos recursos; [18].

Foram assim definidos três modelos de serviço que agrupam as diferentes utilizações da Cloud:

- Software as a Service (SaaS) - Este é o modelo mais simples e mais utilizado dos três. O SaaS é uma forma de distribuição de software, na qual o provedor é o responsável por toda a estrutura necessária para disponibilização do sistema. O utilizador apenas tem de configurar e utilizar o software que irá utilizar, evitando assim custos de manutenção em hardware e suporte, por exemplo, Google Docs.

- Platform as a Service (PaaS) - É uma plataforma de computação que consiste num serviço de alojamento e implementação de software suportado pelo fornecedor do serviço. Sendo assim, os programas não tem obrigatoriedade de serem instalados nas máquinas dos utilizadores, ultrapassando as restrições às configurações de hardware que cada utilizador poderia enfrentar, ou seja, o cliente não necessita de controlar a estrutura técnica da Cloud: rede, sistema operativo e armazenamento, por exemplo, OpenShift.

- Infrastructure as a Service (IaaS) - Os serviços de infraestrutura são tipicamente recursos de computação, armazenamento e rede que o utilizador tem disponíveis, sendo estes suportados com recurso à de virtualização e fornecidos através da Internet. O IaaS torna-se particularmente útil pois o custo associado é definido pela quantidade de recursos que são utilizados, ou seja o utilizador pode ampliar ou reduzir recursos consoante a sua necessidade, evitando assim o desperdício de recursos, por exemplo, OpenStack, Amazon EC2.

É possível que uma Cloud tenha quantidades avultadas de recursos uma vez que os recursos são distribuídos pela rede. Assim não só o utilizador final tem vantagens na utilização da Cloud, mas também as empresas, uma vez que passam a poder contratar um serviço de Cloud em vez da tradicional compra de hardware sendo estes pouco expansíveis devido às limitações inerentes aos recursos. Como tal as empresas podem assim evitar a contratação de equipas de especialistas e podem colocar de parte as preocupações relativas às atualizações dos sistemas. Como tal foram assim definidos quatro tipos de implementações de Clouds diferentes:

- Cloud Privada - É idealizado para uma organização que tem muitos utilizadores, sendo que toda a infraestrutura física pertence à organização. Pode também pertencer a um provedor

sendo que, caso seja este o caso, é necessário que exista um acordo em relação às aplicações que são implementadas e executadas na Cloud. Este é o tipo de Cloud que oferece mais segurança e transparência, mas é também o que requer o maior investimento (como por exemplo OpenStack numa infraestrutura privada).

- Cloud Comunitária - A infraestrutura da Cloud é partilhada por diversas organizações que partilhem os mesmos interesses, características ou preocupações (como por exemplo na política ou em uma missão), podendo assim partilhar os custos entre as organizações.

- Cloud Pública - A infraestrutura pertence ao provedor e os serviços estão disponíveis através de uma rede pública a vários utilizadores. Este tipo de Cloud é altamente escalável e tem um baixo custo. A maior diferença entre uma Cloud pública e uma privada assenta na segurança, uma vez que na Cloud pública os dados estão no poder de terceiros (por exemplo Amazon AWS).

- Cloud Híbrida - Com este tipo de Cloud torna-se possível que quando uma Cloud privada esgota os recursos que tem disponíveis veja os seus recursos ampliados recorrendo a uma Cloud pública (por exemplo expandir os recursos de uma Cloud privada OpenStack na Amazon AWS).

Capítulo 2

Estado da Arte

A Cloud Computing é uma área relativamente recente no mundo da computação, no entanto o estudo que tem sido feito em seu torno fez com que esta evoluísse de uma maneira excecional. Esta é vista como o futuro para as organizações uma vez que o utilizador além de se deixar de preocupar com o hardware, passando essa parte para o provedor de serviços sendo este o único responsável pela infraestrutura, passa a poder dar um aproveitamento diferente aos recursos que dispõe.

Num sistema tradicional os recursos são aprovisionados e configurados manualmente por uma pessoa responsável, enquanto que na Cloud está presente o conceito de multi-tenant, isto é, os recursos são aprovisionados automaticamente a partir de um conjunto de recursos disponíveis.

Para que a adoção da computação na Cloud seja possível foi necessário que as infraestruturas já existentes fossem adaptadas. Se já havia a preocupação que uma infraestrutura fosse redundante, com a computação na Cloud este passou a ser um ponto fulcral pois caso algum componente falhe, é necessário que exista um substituto imediato a assumir o papel do componente em falha. Sendo assim é necessário que seja possível escalar a infraestrutura de forma horizontal e não vertical, fazendo assim com que seja possível remover e acrescentar novas instâncias de forma a que estas possam assumir o papel das anteriormente existentes, eliminando assim a dependência de máquinas que possam falhar, tornando a infraestrutura tolerante a falhas.

Como tal foram criadas soluções para este paradigma. Embora as soluções existentes tenham funcionalidades diferentes, estas são obrigadas a apresentar algumas características comuns das quais não podem fugir designadamente: o serviço deve ser self-service, os recursos devem ser acessíveis a partir de qualquer dispositivo comum, existência de um conjunto de recursos disponíveis, elasticidade rápida e monitorização do serviço para que não haja abuso do uso de recursos partilhados, visíveis na figura 2.1.

Estas soluções podem ser comerciais ou comunitárias.

Figura 2.1: Figura representativa das características essenciais de uma Cloud. Imagem adaptada de [1].

2.1 Soluções Comerciais

O objetivo desta tese é a utilização de uma solução open source pelo que qualquer solução comercial está completamente excluída uma vez que dada a nossa intensiva utilização e necessidade de recursos computacionais juntamente com o facto de ser necessário licenciamento para a utilização e exploração das soluções comerciais, torna-se a médio e longo prazo economicamente rentável a aquisição de uma infraestrutura própria para o Departamento de Ciência de Computadores, da Faculdade de Ciências da Universidade do Porto juntamente com o INESC TEC.

2.1.1 Microsoft Azure

O Microsoft Azure é uma plataforma e infraestrutura (PaaS e IaaS) de Cloud Computing criada pela Microsoft para desenvolver, implementar e gerir aplicações e serviços alojados nos data centers da Microsoft. Esta plataforma é flexível uma vez que possui centenas de modelos previamente criados, sendo que esta possui alta compatibilidade com a mais ampla seleção de sistemas operativos, bases de dados e com múltiplas linguagens de programação. Tal como nas soluções de Cloud computing que serão descritas neste capitulo 2, o Azure é também escalável uma vez que o serviço contratado pode ver os seus recursos aumentados sem nenhuma necessidade de configuração extra.

Hoje em dia a infraestrutura da Cloud da Microsoft suporta mais de 1 bilião de utilizadores distribuídos por 140 países e tem suporte de 10 linguagens diferentes [2]. O Microsoft Azure é o principal concorrente ao Amazon AWS, entre as soluções pagas para Cloud Computing. Tanto a Microsoft Azure, como a Amazon AWS e a Google Cloud permitem que o utilizador recorra a *containers* Docker (tecnologia open source que permite criar, executar,testar e implementar

aplicações distribuídas dentro de contentores de software, sendo estes contentores um método de virtualização de um sistema operativo [19])

2.1.1.1 Computação

A Microsoft garante um serviço de alto desempenho na computação, com acesso a recursos escaláveis para que o utilizador consiga computar as tuas tarefas como análise de dados em grande escala, simulações e experiências em alta disponibilidade [20].

Com o Azure é possível criar e executar máquinas virtuais Linux e Windows em segundos, sendo que estas contam com um serviço de redimensionamento de recursos automático, para que o utilizador não tenha de pré-aprovisionar os recursos. Este oferece ainda um serviço para gestão de imagens de containers como DC/OS, Docker Swarm ou Kubernetes (Azure Container Registry) [21].

Existe ainda uma plataforma como serviço para Aplicações Web, suportando diversas linguagens de programação, com redimensionamento de recursos automático ou manual e possuindo diversos modelos de aplicação, contanto com uma série de software open source para que o utilizador apenas tenha de selecionar qual vai utilizar evitando a perda de tempo a instalá-lo [22].

2.1.1.2 Armazenamento

A Microsoft oferece vários tipos solução a nível de armazenamento como: discos, Blobs (objetos) e tabelas NoSQL [23]. A nível de discos conta com dois serviços diferentes: um Premium que utiliza Solid State Drives (SSDs) com um débito de 80 000 operações de IO por segundo e 2000 MB e outro normal com Hard Disk Drives (HDDs), aconselhado para testes. Tanto os SSD como os HDD são um tipo de dispositivo para armazenamento não volátil de dados, no entanto os SSD são construídos com base num circuito integrado semicondutor enquanto que os HDD são constituídos por discos metálicos com uma cobertura magnética. A nível de performance os SSD são superiores aos HDD, bem como também são mais resistentes a quedas [24]. O armazenamento de objetos é dimensionavel para dados não estruturados, com escalabilidade massiva e capacidade para exabytes. As tabelas NoSQL utilizam um tuplo chave-valor NoSQL e permitem que sejam armazenados petabytes de dados e estes sejam acedidos com uma grande performance.

2.1.1.3 Base de dados

A Base de Dados SQL da Microsoft Azure é um serviço de base de dados relacional baseado no já existente Microsoft SQL Server. Esta garante um ótimo desempenho, bem como escalabilidade sem períodos de indisponibilidade e proteção de dados. Existe também o Microsoft Azure SQL Data Warehouse que é uma base de dados que permite escalabilidade horizontal e é capaz de processar grandes volumes de dados, sejam estes do tipo relacional ou não relacional [25].

2.1.1.4 Rede

Ao se optar por Cloud Computing, a organização deixa de parte as preocupações maioritárias na configuração da rede, no entanto é útil poder ter controlo sobre a rede que utiliza na sua Cloud. Como tal com o Microsoft Azure cada utilizador possui um ambiente isolado e altamente seguro para executar as suas instâncias máquinas virtuais, podendo assim definir sub-redes e politicas de controlo de acesso às suas instâncias.

2.1.1.5 Segurança

Uma das estratégias utilizadas pela Microsoft para a melhoria continua da segurança é a utilização de uma equipa, *red team*, para que explorem possíveis vulnerabilidades, podendo assim corrigi-las, melhorar a segurança e melhorar a recuperação de informação caso seja perdida. Esta equipa tipicamente ataca as redes, a camada de aplicação e a plataforma. A Microsoft faz uma subdivisão em outros níveis de segurança: Infraestrutura, rede e dados.

A segurança da Infraestrutura divide em: segurança física, monitorização e antivírus. A Microsoft utiliza sistemas de deteção de intrusões, sistemas distribuídos para prevenção de *denial-of-service (DDoS)*, testes de penetração, análise de dados e *machine learning* para que se reforcem as defesas e se minimizem os riscos de ataques [2].

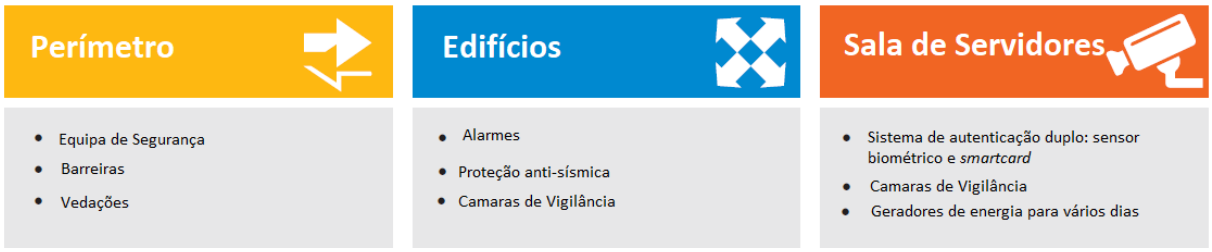


Figura 2.2: Figura representativa de segurança aplicada nas instalações dos datacenters da Microsoft. Imagem adaptada de [2]

.

Segurança da Rede

Num modelo de datacenter tradicional, os responsáveis por esse datacenter são quem controlam toda a rede, incluindo o acesso físico ao equipamento de rede. No modelo de Cloud as responsabilidades da proteção e gestão da rede são divididas entre o provedor e o utilizador e embora os utilizadores não tenham acesso físico aos equipamentos de rede podem implementar na sua Cloud firewalls ou VPNs. Assim, para proteger os seus clientes, a Microsoft implementou políticas de proteção de rede como isolamento de redes, redes virtuais e VPN.

Segurança dos Dados

O grande receio das empresas a quando uma migração para a Cloud é a segurança dos seus dados uma vez que ficam alojados na Cloud. Para contornar este obstáculo a Microsoft assumiu um compromisso para a proteção e privacidade dos seus dados. Como tal foram os primeiros a serem reconhecidos pelas autoridades de proteção de dados da União Europeia, devido ao rigoroso cumprimento da legislação, tendo sido também os primeiros a adotar a nova norma internacional de privacidade na Cloud, a ISO 27018 [26] [27].

Segurança da Identidade

A gestão da identidade e dos acessos à Cloud são a chave para a segurança de uma empresa e, por isso, a Microsoft usa rigorosos controles de gestão de identidade e acesso de forma a limitar e restringir o acesso aos dados de cada empresa. Eles utilizam um modelo de acesso privilegiado em que cada utilizador tem uma identidade, podendo essa identidade ter ou não acesso a certos dados, protegendo assim a informação do acesso por parte de pessoas não autorizadas.

2.1.2 Amazon AWS

A Amazon Web Services (AWS) é uma plataforma de serviços de Cloud Computing pertencente à Amazon.com, constituída por vários serviços e sendo esta escalável, permitindo às empresas reduzirem drasticamente as suas despesas em hardware focando-se no uso de serviços fornecidos pela Cloud, neste caso da Amazon.[28]. Os Principais Serviços da AWS são: computação, armazenamento, base de dados e redes.

Infraestrutura

A Cloud da Amazon Web Service atua em 38 zonas de disponibilidade e em 14 regiões geográficas dispersas pelo globo, com a previsão de mais 9 zonas de disponibilidade e 4 regiões que vão entrar em operação durante o próximo ano de 2017 [3]. Uma região é um local físico do mundo enquanto uma zona de disponibilidade é vista como um datacenter alojado num local físico, sendo que possui alimentação e redes redundantes, com instalações separadas. Por exemplo Frankfurt (região física) tem 2 zonas de execução. É possível visualizar a infraestrutura global da AWS de 2016 na imagem 2.3.

2.1.2.1 Computação

Uma empresa pode alugar os seus recursos computacionais, evitando assim gastar imenso dinheiro em hardware, alugando assim os recursos consoante a sua necessidade uma vez que estes são altamente escaláveis, não tendo que se preocupar com a manutenção dos mesmos. A AWS conta

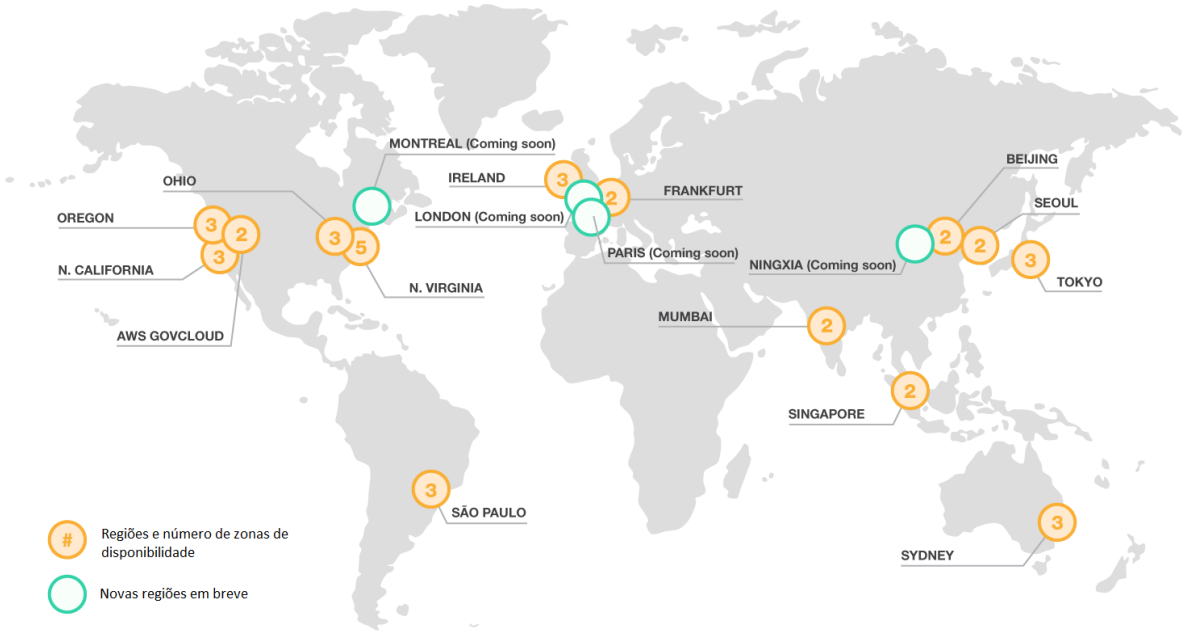


Figura 2.3: Figura representativa da Infraestrutura Global da AWS em 2016. Imagem adaptada de [3]

.

com mais de 70 serviços de infraestrutura [29] contando todos eles com escalabilidade automática e balanceamento de carga, sendo os principais:

- **Amazon EC2** - é um serviço da web que fornece capacidade de computação na cloud escalável em que os seus recursos totalmente controláveis pelo utilizador, permitindo que o utilizador apenas pague pelos recursos que realmente utilizar;
- **Amazon EC2 Container Service** - é um serviço que permite executar e gerir contentores do Docker, permitindo executar facilmente aplicações em um cluster do Amazon EC2. Com o ECR o utilizador não tem a necessidade de instalar, operar e escalar a sua própria infraestrutura de gestão de *clusters* [30]. Existe ainda o Amazon EC2 Container Registry que serve para armazenar e recuperar imagens do Docker [31].
- **AWS Lambda** - permite que sejam executadas praticamente todo o tipo de aplicações para back-end sem que seja necessário possuir ou gerir servidores, sendo que as aplicações são escaladas automaticamente caso necessário [32]. Com este serviço o utilizador apenas paga durante o tempo em que utilizou recursos, sendo que só é necessário carregar o seu código numa linguagem suportada (atualmente Node.js, Java, C e Python) e clicar em executar.

2.1.2.2 Armazenamento

A necessidade de uma empresa possuir um sítio seguro e confiável para armazenar a sua informação e com a possibilidade de obrigatoriedade de latências baixas consoante à localização geográfica de determinado utilizador, levou a que a Amazon criasse alguns serviços diferentes no que diz respeito ao armazenamento, à distribuição de conteúdo e backups, adequáveis às necessidades do utilizador.

- **Amazon S3** - é um serviço de armazenamento seguro na cloud e escalável. O Amazon S3 dispõe de uma interface de web simples para armazenar e recuperar qualquer quantidade de dados de qualquer parte da web. O Amazon S3 pode ser usado isoladamente ou com outros serviços da AWS, como o Amazon Elastic Compute Cloud (Amazon EC2) [33].
- **Amazon CloudFront** - é um serviço de rede para distribuição de conteúdo (CDN) global que acelera a distribuição do conteúdo que compõe os sites, Application programming interfaces (APIs), vídeo ou outros recursos da web a utilizadores finais, uma vez que utiliza uma distribuição geográfica diminuindo assim as latências de transporte e entrega de conteúdos [34].
- **Amazon EBS** - fornece volumes de armazenamento de blocos persistentes para uso em conjunto com instâncias do Amazon EC2 na cloud da AWS, sendo que cada volume EBS é replicado automaticamente, ficando assim protegidos contra eventuais falhas de componentes [35].

2.1.2.3 Base de dados

Qualquer organização necessita de uma base de dados na qual possa guardar os dados de forma persistente e organizada, sejam estes sobre colaboradores ou sobre os produtos que estas possuem. Desta forma o Amazon conta com alguns serviços de bases de dados, como a criação de uma base de dados de um tipo conhecido (como MySQL ou PostgreSQL) ou a migração de bases de dados sem que haja inatividade.

- **Amazon RDS** - serve para facilitar a configuração, operação e escalabilidade de bases de dados relacionais na Cloud, disponibilizando seis tipos de bases de dados conhecidos: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL e MariaDB [36].
- **Amazon DynamoDB** - é um serviço de bases de dados NoSQL (bases de dados não relacionais) rápido e flexível para todas as aplicações que precisam de latência constante abaixo de 10 milissegundos em qualquer escala [37]. Esta latência é obtida através do tempo em que é demorado a executar uma *query* dentro da Cloud. O tempo que o cliente demora a obter a resposta depende de vários fatores, como por exemplo distância geográfica.

- **Amazon ElastiCache** - é um serviço web para cache de dados na memória na cloud [38], que é compativel com o Redis (armazenamento e cache de dados rápidos) e o Memcached (um sistema de armazenamento em cache de objetos na memória [39]).

2.1.2.4 Rede

Uma organização ao optar por Cloud Computing deixa de parte as preocupações maioritárias na configuração do hardware, no entanto é útil poder ter controlo sobre a rede que utiliza na sua Cloud. Sendo assim existem serviços para o utilizador conseguir criar redes isoladas, podendo o utilizador manipular a sua rede sem interferir com outro utilizador. A rede é também um ponto suscetível a ataques numa organização, pelo que é necessário que exista um serviço que faça uma conexão segura entre o ambiente local e os serviços da Amazon.

- **Amazon VPC** - permite criar uma Cloud virtual privada, isolada logicamente, onde é possível executar recursos da AWS em uma rede virtual criada pelo utilizador. O utilizador tem gestão total sobre a sua rede virtual, incluindo a sua própria *pool* de endereços IP, criação de sub-redes e configuração de tabelas de encaminhamento e gateways de rede [40].
- **Amazon Direct Conect** - serve para estabelecer uma conexão de rede dedicada entre o ambiente local e a AWS podendo assim conectar de forma privada a AWS ao ambiente local da empresa, seja este um datacenter ou um escritório. Para tal é utilizada uma interface virtual privada [41].
- **Elastic Load Balancing** - serve para distribuir automaticamente o tráfego de entrada das aplicações nas várias instâncias do Amazon EC2. Proporciona dois tipos de balanceadores que fornecem alta disponibilidade: o Classic Load Balancer que encaminha o tráfego com base nas informações da aplicação e da rede, ideal para balanceamento de carga simples entre instâncias EC2, e o Application Load Balancer que encaminha o tráfego com base em informações avançadas da aplicação, como o conteúdo, ideal para aplicações que necessitem de recursos avançados de encaminhamento [42].

  O Elastic Load Balancing funciona em conjunto com o VPC permitindo que se aumente a segurança uma vez que é possível implementar uma arquitetura de camadas usando balanceadores de carga e encaminhar tráfego entre as diferentes camadas, sendo que caso alguma necessite de estar conectada à Internet apenas esta fica exposta. Este fornece ainda uma gestão integrada e desencriptação de certificados SSL permitindo que a carga seja aliviada entre instâncias. É ainda integrável com o AWS Certificate Manager para que seja facilitada a ativação do SSL/TLS (este é um protocolo que protege a troca de comunicação - email, navegação - HTTPS e a transferência de dados, via Internet [43]).
- **Amazon Route 53** - é um web service de Domain Name System (DNS) altamente disponível e escalável. Este serve para encaminhar os utilizadores finais para aplicações de Internet traduzindo nomes para endereços IP, usados para os computadores se conectarem [44].

2.1.2.5 Segurança

Uma das maiores prioridades da AWS é a segurança que proporcionam à Cloud dos seus utilizadores, o datacenter e a arquitetura de rede foram criados para atender às maiores exigências de segurança dos utilizadores. A AWS garante que os utilizadores podem escalar e inovar sem deixar de manter um ambiente seguro e, uma vez que os clientes pagam apenas pelos serviços que utilizam, podem usufruir de toda a segurança necessária sem que tenham que ter despesas com segurança.

2.1.3 Google Cloud Platform

A Google Cloud Platform, tal como o nome indica, é uma plataforma de serviços de Cloud Computing pertencente à Google. O seu lema é: "deixe os inovadores inovar, deixe os programadores programar". A Google Cloud Platform permite migração de máquinas sem que se tenha de parar alguma mesmo quando estas estão a sofrer sobrecarga; permite ao utilizador escolher um tipo de configuração hardware em específico quando vai contratar o serviço, para que pague pelo hardware que realmente pretende; possui um sistema de backups contra desastres, acessível instantaneamente, com o preço de 1 cêntimo por gb/mês; dispõem de balanceadores de carga que permitem escalar 1M de utilizadores instantaneamente; prometem um tempo de arranque 1/5 mais rápido das máquinas virtuais do que na concorrência e têm presença em 75 pontos e mais de 33 países [45].

2.1.3.1 Computação

A Google oferece três opções a nível de computação, de forma a satisfazer as necessidades de cada utilizador: Google Compute Engine, Google Container Engine e Google App Engine.

O Google Compute Engine oferece máquinas virtuais com recursos personalizáveis, sendo que o utilizador pode optar por implementar o seu código diretamente na máquina ou utilizar containers para o fazer.

O Google Container Engine permite que o utilizador crie e utilize clusters de Kubernetes [46], sendo que estes fazes com que o utilizador consiga automaticamente escalar, fazer deployment ou simplesmente gerir aplicações que utilizem containers.

O Google App Engine é uma plataforma como serviço que permite que um programador apenas se foque no código, deixando os detalhes operacionais de implementação e gestão da infraestrutura do lado da Google.

2.1.3.2 Armazenamento

O Google Cloud Storage é um serviço de armazenamento de objetos altamente escalável, sendo este adequado a todos os tipos de dados não estruturados. A Google garante que o armazenamento dos objetos é seguro e altamente disponível, adequado até mesmo à mais extrema carga. Este possui apenas uma API que possibilita o acesso a três opções distintas: fluxo de alta transferência, análise de dados em grande quantidade, ou acesso a backups.

Este alto fluxo de elevada transferência deve-se ao facto de ser utilizado um sistema de armazenamento distribuído redundante, diminuindo assim as latências para acessos dispersos aos recursos vindos de diferentes partes do globo. O utilizador pode também optar por um armazenamento regional, isto é, deixar de ter o seu armazenamento distribuído redundantemente pelo globo, aumentando as latências no acesso e usufruindo de um custo mais reduzido.

Existe ainda um serviço chamado Coldline, de armazenamento com baixo custo criado para utilizadores que raramente tenham necessidade de aceder a esses dados, mas que caso necessitem, os tenham à disposição em milissegundos sendo este chamado de Coldline [47].

2.1.3.3 Base de dados

O Google Cloud SQL é um serviço dedicado especificamente a base de dados, sendo este totalmente gerido o que facilita a configuração, manutenção e gestão das base de dados relacionais MySQL do utilizador [48].

Big Data

Atualmente as aplicações e os utilizadores geram uma quantidade infinita de dados em cada utilização da Internet. A essa quantidade exorbitante de dados dá-se o nome de Big Data e atualmente este é visto como um negócio, sendo que cada vez mais se utiliza a computação na Cloud para fazer análise sobre este tipo de dados (data mining). Segundo a Google ao serem utilizadas as suas ferramentas para este tipo de dados, uma pesquisa que demoraria horas ou dias na concorrência, passa a demorar segundos [49].

2.1.3.4 Rede

O utilizador pode gerir a sua própria rede virtual, podendo assim conectar diferentes recursos de que disponha na Google Cloud Plataform. Dentro desta rede virtual o utilizador tem acesso a um conjunto de endereços IP que pode utilizar, rotas, firewalls, Virtual Private Network (VPN) e um Cloud Router [50].

2.1.3.5 Segurança

O modelo de segurança da Google foi desenvolvido ao longo dos últimos 15 anos com o objetivo de manter os seus utilizadores seguros nas suas aplicações como GMail e é um modelo end-to-end. A Google possui uma equipa especializada em segurança com mais de 500 membros, com o objetivo de manter a segurança da empresa e dos sistemas, tendo estes implementado por exemplo o modelo de certificado *SSL* por defeito [51]. E porque a segurança não pode ser só virtual, a Google também possui segurança a nível físico, nomeadamente cartões de acesso eletrónico, alarmes, barreiras de acesso, perímetros de segurança, detetores de metais e sensores biométricos. Todos os acessos a ambientes de produção e administração são controlados. Quando um disco é retirado de uma máquina é sujeito a destruição dos dados armazenados, sendo que caso surja algum erro ao limpar o disco e esta limpeza não seja possível, o disco é fisicamente destruído.

A segurança é um requisito do núcleo de qualquer aplicação da Google, como tal são utilizadas API's seguras, sendo o acesso apenas possível mediante a utilização canais seguros encriptados com *SSL/TLS*. São também utilizados sistemas de autenticação de 2 fatores para que caso um atacante descubra uma chave não consiga autenticar-se sem o segundo fator de autenticação. Todos os dados são cifrados com AES-256 e cada chave de cifra é igualmente cifrada com uma chave rotativa de um conjunto de chaves mestras [52]. É utilizado ainda um sistema de intrusões para aumentar a segurança. A Google possui uma série de certificações como a ISO 27001, a ISO 27017 ou a ISO 27018 [53].

2.1.4 Comparativo de Preços

Como descrito acima na presente secção, as três grandes empresas comerciais de Cloud Computing são muito semelhantes. Como tal decidimos simular a compra de uma máquina virtual em cada uma das empresas com 1 CPU, 1 GiB de memória RAM e 20 GiB de armazenamento, de forma a podermos comparar o preço entre as três.

	CPU	RAM (GiB)	Disco (GiB)	Preço por Hora (euros)
Azure	1	1	20	0.016
Amazon	1	1	20	0.014
Google	1	1	20	0.026

Tabela 2.1: Comparativo de preços de uma máquina virtual entre Azure, Amazon e Google.

Da tabela 2.1.4 é possivel verificar que a empresa com o serviço mais barato é a Amazon, com um preço de 0.014 euros por hora, equivalente a 10.416 euros por mês, 124.992 euros por ano.

2.2 Soluções open source

Foram analisadas as soluções open source mais conhecidas e utilizadas a nível mundial para implementação de um modelo de Cloud do tipo IaaS: OpenNebula, Eucalyptus, CloudStack, e OpenStack. Como o pretendido com esta tese de mestrado era a instalação do OpenStack, foi dado mais ênfase às soluções open source do que as soluções comerciais, para que possa existir uma comparação sobre se a escolha do OpenStack foi a mais acertada. É possível visualizar nas seguintes secções que o OpenStack é a solução open source com o maior desenvolvimento, apoio (tanto a nível da comunidade como a nível de suporte comercial), com maior compatibilidade com hypervisors e com a arquitetura mais modular e distribuída, pelo que esta é vista como a solução mais completa disponível gratuitamente.

2.2.1 OpenNebula

O OpenNebula começou por ser um projeto de investigação em 2005 de Ignacio M. Llorente e Rubén S. Montero [54] e atualmente é um software open source para implementação do modelo IaaS de Cloud do tipo Público, Privado e Híbrido. Este gere um conjunto de recursos como armazenamento, rede, virtualização ou políticas de segurança, que são utilizadas para a criação de máquinas virtuais em infraestruturas diferentes, combinando assim os recursos dos diferentes datacenters. É utilizada o algoritmo RSA. Este é recomendado para quem procura soluções pequenas/médias de Cloud Computing.

O OpenNebula têm duas funcionalidades chave: gestão virtual da infraestrutura e gestão da Cloud (visível na fig 2.4). Com a gestão virtual da infraestrutura o OpenNebula integra-se diretamente com os *hypervisors* (KVM, Xen ou VMware ESX) e tem completo controlo sobre os recursos virtuais e físicos, garantindo assim uma excelente capacidade de otimização e gestão de recursos em alta disponibilidade. Com a gestão da Cloud é possível criar múltiplos projetos e gerir de forma simples a Cloud através da sua interface gráfica.

Existem também utilizadores que utilizam o OpenNebula como uma camada superior numa infraestrutura já existente, como por exemplo a Amazon AWS. Assim os utilizadores conseguem facilmente criar vários utilizadores com acesso à Cloud (*multi-tenancy*) e conseguem também criar uma Cloud Híbrida caso disponham de uma infraestrutura privada que utilie o OpenNebula [4].

2.2.1.1 Principais Características e Componentes

O OpenNebula foi desenhado com alguns princípios subjacentes (fig 2.4), resultantes da investigação que decorreu sobre si. Este tem uma gestão centralizada de toda a infraestrutura virtualizada e física, a escalabilidade é rápida, permite total controlo da infraestrutura, suporta vários ambientes de execução, sendo possível a sua implementação para qualquer tipo de Cloud e

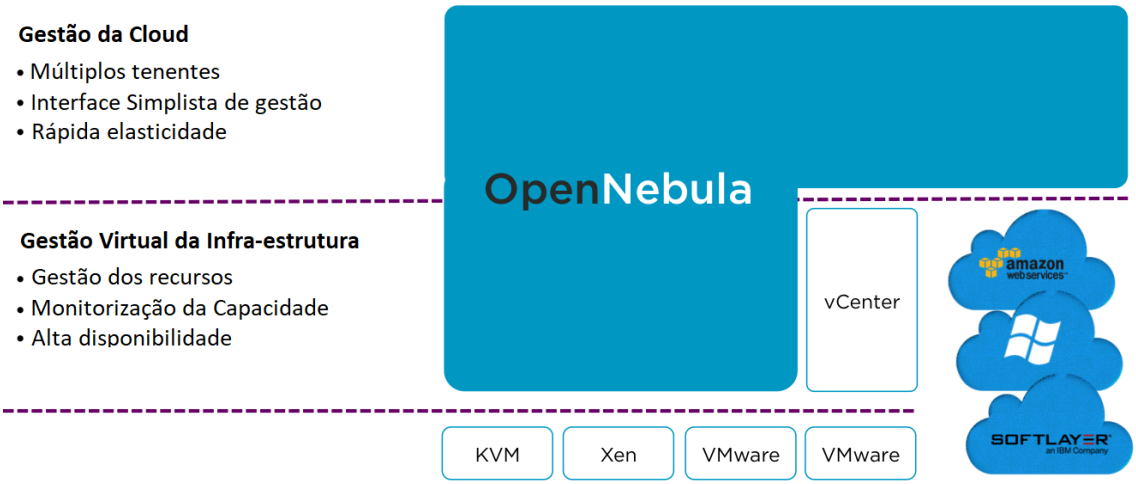


Figura 2.4: Imagem representativa da gestão de Cloud e da gestão da virtualização do OpenNebula. É visível a Cloud Management como camada superior de serviços como Amazon AWS ou Windows Azure, assim como é visível a integração com diferentes hypervisors. Imagem adaptada de [4]

é fácil de ser implementado em datacenter já existentes. Estas características são obtidas devido aos princípios implícitos pelos responsáveis do OpenNebula [4].

- **Abertura e Estabilidade -** Todo o software é aberto e está pronto para utilização em ambientes empresariais, sem que seja necessário utilizar algum tipo de extensão ou licença que diferencia versões open source de empresariais como acontece no caso do CentOS e da RedHat [55];
- **Flexibilidade e Portabilidade -** Adapta-se a todo o tipo de hardware presente nos mais diversos datacenters, evitando assim que se fique preso a um determinado fabricante de hardware;
- **Simplicidade -** Fácil de instalar e utilizar, uma vez que o OpenNebula é um produto e não um conjunto de serviços e ferramentas interligadas;
- **Escalabilidade -** Permite facilmente que os recursos sejam aumentados consoante a necessidade podendo assim ampliar as infraestruturas sem nenhum tipo de problema, garantindo assim ter um bom funcionamento durante anos sem que sejam necessárias manutenções muito agressivas;

Devido a tais características a escolha de hardware, a manutenção, a instalação e a utilização tornam-se simples. Com o OpenNebula apenas existem três componentes na arquitetura: armazenamento, rede e virtualização.

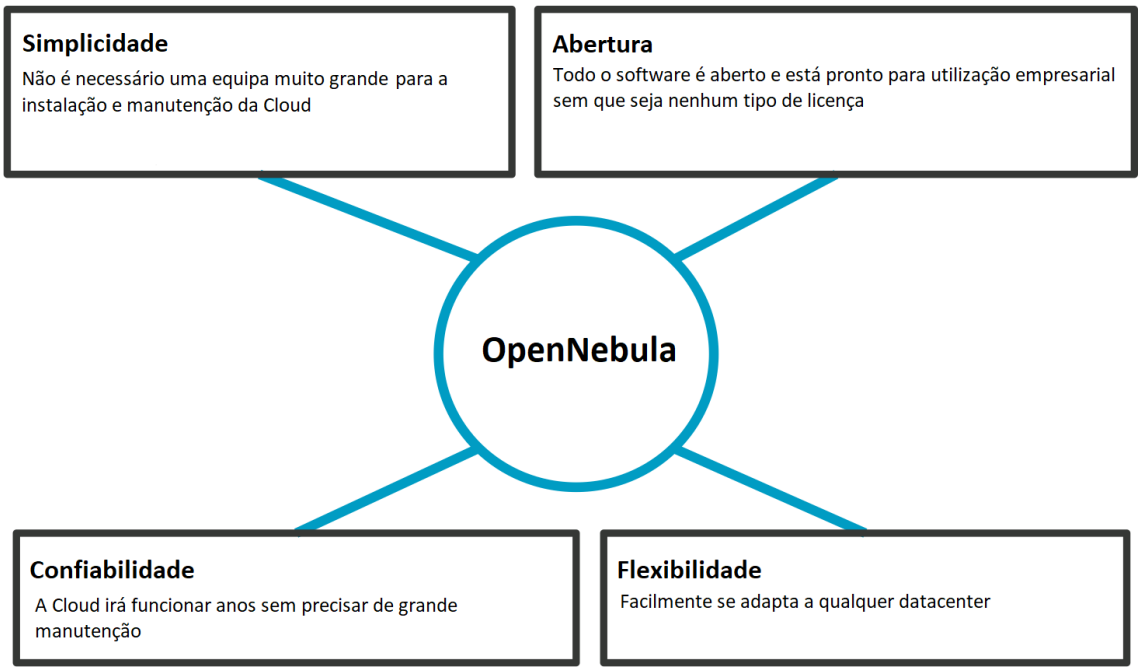


Figura 2.5: Imagem representativa da características do OpenNebula [5].

2.2.1.2 Segurança

O OpenNebula fornece um acesso seguro através de HTTPS, autenticação cifrada através do par utilizador/password, chave pública/privada SSH, X509 ou LDAP. Fornece ainda isolamento de redes para que as instâncias sejam separadas virtualmente caso necessário.

2.2.1.3 Compatibilidade com os Serviços Web da Amazon

O OpenNebula é totalmente compatível com os serviços web da Amazon, sendo que diversos utilizadores utilizam o OpenNebula como camada superior do AWS.

2.2.2 Eucalyptus

O Eucalyptus [56] é um software open source que começou a ser desenvolvido em 2008 pela Eucalyptus Systems, Inc. e é utilizado para a construção de serviços da Amazon [57]. Este software é compatível com o padrão de Cloud Computing do tipo Privado e Híbrido. Através deste é possível reservar-se uma *pool* de recursos como armazenamento, rede ou simplesmente recursos disponíveis para computação sendo que estes são dinamicamente elásticos, isto é, podem ser ampliados ou reduzidos consoante a carga que é necessária. Em 2014 Hewlett-Packard (HP) comprou o Eucalyptus com o objetivo de conseguir acelerar a adoção de Cloud Híbrida na empresa [58], acrescentado ao nome HPE Helion Eucalyptus.

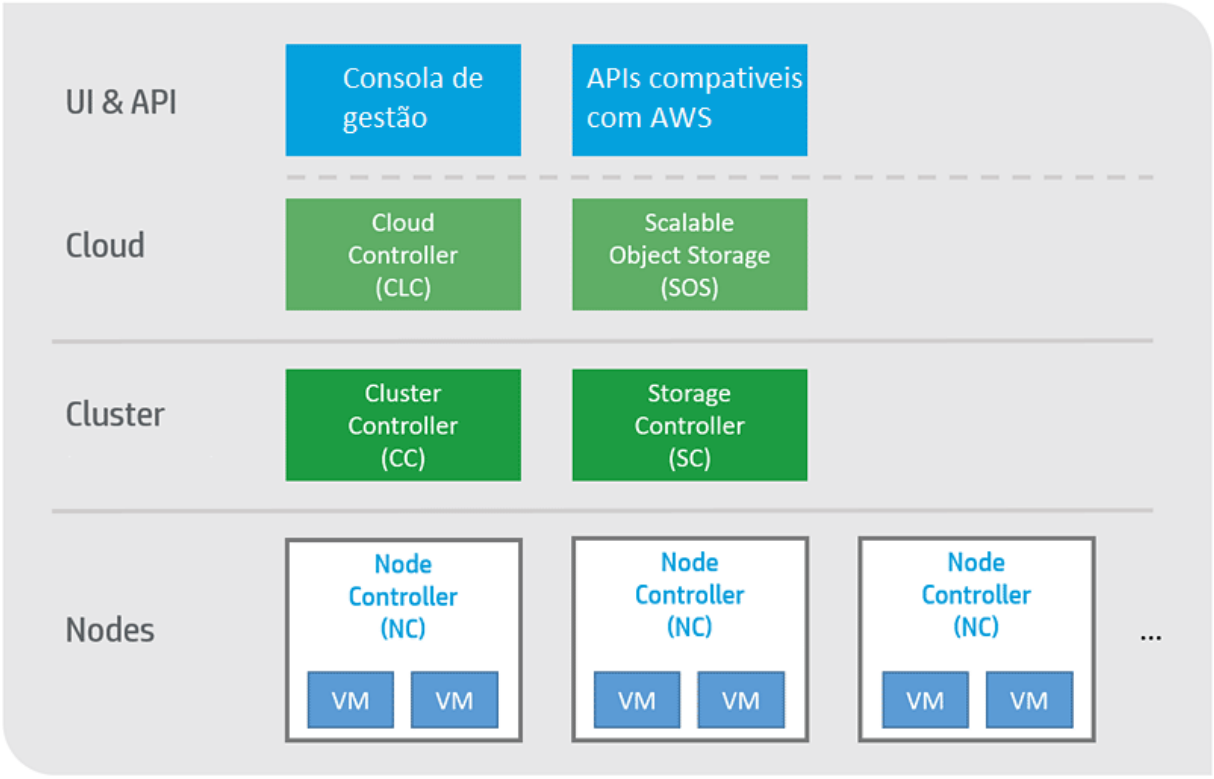


Figura 2.6: Imagem dos Componentes do Eucalyptus. Imagem adaptada de [6]

2.2.2.1 Principais Caracteristicas e Componentes

O Eucalyptus é constituído por diversos componentes que servem para interligar os diferentes serviços que constituem este software, nomeadamente: os nós, o cluster, a Cloud, e o interface gráfico e API. Através destes conseguimos que a comunicação entre eles ocorra, comunicação esta que é necessária para o funcionamento do sistema de Cloud do Eucalyptus.

- **Cloud Controller -** O CLC é um programa que oferece compatibilidade com as interfaces da EC2 da Amazon e tem como objetivo servir de interface de administração para a Cloud e gerir os recursos do sistema como armazenamento e rede, sendo que só pode existir um CLC por Cloud. O CLC é responsável pela autenticação, por enviar *reports* e por gerir as cotas atribuídas a cada instância.
- **Storage Controller -** O SC é um programa que tem como objetivos comunicar com o Cluster Controller e o Node Controller e gerir os blocos persistentes e snapshots de cada instância pertencente ao seu cluster, replicando-os, garantindo assim alta disponibilidade em caso de falha de algum componente. O SC tem um funcionamento equivalente ao Amazon Elastic Block Store (EBS) [35].
- **Walrus Storage -** O Walrus permite um armazenamento persistente obtido através de *get*/*put* em *HTTP*, em todas as máquinas virtuais da Cloud. É possível armazenar snapshots

de volumes, imagens ou dados de aplicações, sem nenhum tipo de restrição em relação ao seu tipo de dados. O seu funcionamento é equivalente ao AWS Simple Storage Service (S3) [33]. Apenas existe um Walrus por Cloud.

- **Cluster Controller -** O CC é o responsável por gerir a execução de instâncias e o SLA (Server Level Agreements) de cada Cluster. Este comunica com o Sotrage Controller e com o Node Controller.
- **Node Controller -** O NC aloja todas as instâncias e gere as redes virtuais dos *endpoints*. O NC faz download de imagens do Walrus e armazena-as em *cache*.

O Cloud Controller, o Storage Controller e o Walrus Storage estão implementados em JAVA. O Cluster Controller e o Node Controller estão implementados em C.

Arquitetura

Uma vez que a arquitetura do Eucalyptus (fig 2.7) é compatível com os serviços Web da Amazon é então possível que se obtenham Clouds Híbridas movendo instâncias da Cloud que utiliza Eucalyptus para a Amazon, e vice-versa. As diferenças a nível de hardware não influenciam o correto funcionamento devido à utilização de virtualização. A terminologia utilizada para as

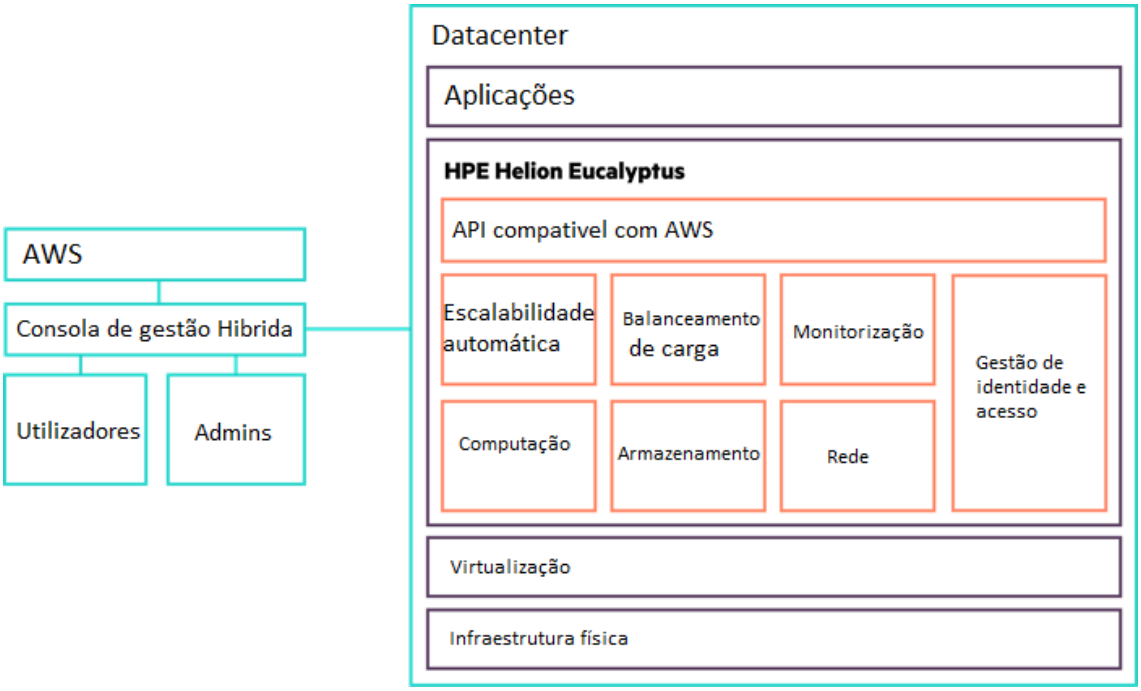


Figura 2.7: Imagem da arquitetura do Eucalyptus. Imagem adaptada de [7]

funcionalidades do Eucalyptus é:

- **Imagem (Eucalyptus EMI) -** Uma imagem é uma coleção de softwares e configurações que são empacotados para que possam ser replicados e utilizados diversas vezes na Cloud

Eucalyptus, quando necessário.

- **Instâncias -** Uma instância é uma imagem que se encontra a correr sendo que o Cloud Controller passa a gerir onde deve a imagem correr e lhe associa o espaço de armazenamento e os recursos de rede necessários.
- **Endereçamento IP -** Quando uma instância é criada é lhe associado um endereço IP, podendo este ser um endereço público ou privado. Caso a instância em questão seja uma instância de um serviço que necessita de um IP reservado, como por exemplo uma instância de um website, o Eucalyptus atribui-lhe um IP. Para as instâncias que não têm a necessidade de ter um IP constante, o Eucalyptus tem elasticidade sobre endereços IP, isto é, quando é necessário, é atribuído um endereço IP à instância que necessita, sendo que estes endereços elásticos podem ser transferidos de instâncias mesmo enquanto estas estão a correr.
- **Segurança -** São utilizados grupos de segurança TCP/IP, sendo que são partilhadas regras comuns na firewall entre os grupos. As instâncias são isoladas no Layer 2 (Data Link Layer) do TCP/IP prevenindo assim que sejam editadas configurações de rede de instâncias vizinhas.
- **Networking -** Na arquitetura do Eucalyptus foram desenhados 3 modos distintos de rede: *Managed Mode*, *System Mode*, *Static Mode*.

  Managed Mode - o Eucalyptus gere a rede local das instâncias;

  System Mode - o Eucalyptus atribui o endereço MAC a uma interface física de rede utilizando o Node Controller como bridge;

  Static Mode - o Eucalyptus atribui endereços IP às instâncias.

  Nos modos System e Static não existe elasticidade de endereços IP, grupos de segurança ou isolação de VM's.
- **Controlo de Acesso -** É atribuída uma identidade a um utilizador do Eucalyptus, sendo que as identidades podem ser agrupadas para controlo de acesso.

2.2.2.2 Segurança

O Eucalyptus fornece um acesso seguro através de HTTPS, autenticação cifrada através do par utilizador/password, chave pública/privada SSH ou LDAP. A transferência de dados entre uma Cloud Privada Eucalyptus e a Cloud Pública AWS é feita de forma cifrada para que os dados não sejam comprometidos.

2.2.2.3 Compatibilidade com os Serviços Web da Amazon

Dado ao facto da API da AWS estar implementada no topo do Eucalyptus (visível na figura 2.8) as aplicações a correr numa Cloud Eucalyptus e que tiram partido de alguma característica

especial da Cloud, conseguem correr na Amazon AWS, como por exemplo compatibilidade com Amazon Elastic Compute Cloud ou Amazon Simple Storage Service e elasticidade de recursos (*Autoscaling*).

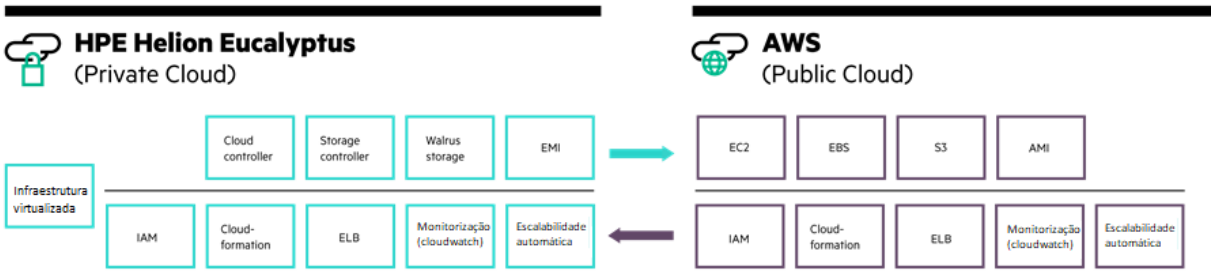


Figura 2.8: Representação da interação Eucalyptus - Amazon AWS. Imagem adaptada de [7]

2.2.3 CloudStack

O Apache CloudStack é um software open source pertencente ao Apache Software Foundation para a criação, gestão e instalação de serviços de infraestrutura pública e/ou privada na Cloud (IaaS) em alta disponibilidade (figura 2.9). Este utiliza *hypervisors* para virtualização, como KVM ou VMWare. A CloudStack é uma solução que inclui a *stack* de características que a maior parte das organizações procuram, como por exemplo gestão da computação, rede como um serviço, gestão de contas de utilizadores e uma API completa e aberta [59]. A infraestrututura de gestão do CloudStack é altamente escalável e pode servir de gestão a dezenas de milhares de hosts instalados em vários datacenters geograficamente distribuídos.

Para uma instalação mínima de produção utilizando CloudStack bastam duas máquinas: uma para executar o CloudStack Management Server e outra para ser utilizada como Infraestrutura da cloud, sendo que esta ultima instalação é mesmo uma infraestrutura mínima e simples. Podem ser configurados vários servidores de gestão, obtendo assim redundância e balanceamento de carga, sendo apenas necessário estes terem acesso a uma base de dados MySQL comum [60].

2.2.3.1 Principais Características e Componentes

O Apache CloudStack é um projeto baseado em Java que fornece gestão de servidores e agentes para *hypervisor hosts*, podendo assim criar-se uma Cloud do tipo IaaS.

O CloudStack funciona com vários hypervisor como por exemplo: XenServer/XCP, KVM, Hyper-V, e/ou VMware ESXi with vSphere e permite migração de VMs sem necessidade de se pararem as instâncias, não necessitando assim de interromper os serviços a correr. Possui uma interface web de fácil gestão da Cloud. Fornece aos utilizadores uma API nativa e uma API compatível com a AWS EC2 e AWS S3 [61]. O CloudStack possui uma interface de linha

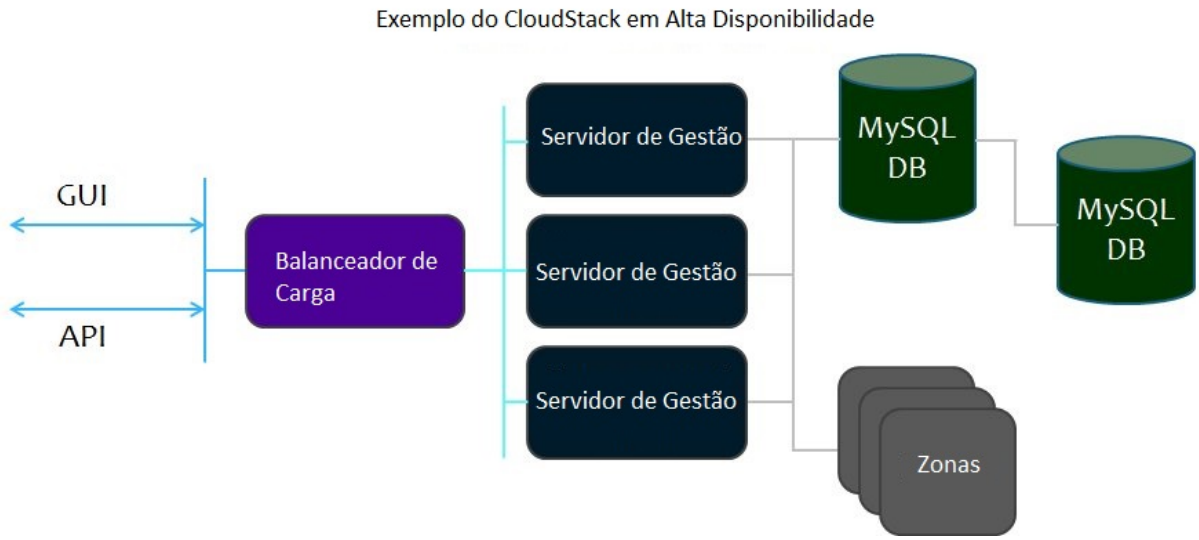


Figura 2.9: Figura representativa do CloudStack em Alta Disponibilidade. Imagem adaptada de [8]

.

de comando denominada CloudMonkey. Esta interface está escrita em python e serve para facilmente se criarem scripts de automação e administração complexa ou repetitiva de tarefas de gestão [62].

O CloudStack possui dois tipos de armazenamento: primário e secundário. O primário serve para armazenar instâncias em execução nos hypervisors e o secundário serve para armazenar templates, imagens ou ISOs. O CloudStack é compatível com Ceph [63] e também permite o controlo da rede como criação e gestão de VLANs, grupos de segurança, firewalls, routers virtuais e balanceamentos de carga. Permite ainda serviço de multi-tenancy e separação de contas, isto é, várias contas com vários níveis de privilégios de administração. É de salientar que este software é totalmente integrável com autenticação LDAP [64].

O CloudStack possui configuração automática: configura automaticamente o armazenamento e a rede de cada máquina virtual. O CloudStack gere internamente um conjunto de ferramentas virtuais para darem apoio à própria infraestrutura da cloud. Estas ferramentas permitem o controlo da rede como criação e gestão de VLANs, grupos de segurança, firewalls, routers virtuais, balanceamentos de carga e replicação.

O CloudStack é constituído por cinco componentes essenciais para o seu funcionamento: Load-Balancer, Cloud-API, Cloud-Acess, Cloud-Engine e Base de Dados.

2.2.3.2 Segurança

A segurança é sempre uma questão e preocupação fundamental em qualquer software utilizado por uma comunidade. Sendo assim as pessoas responsáveis e envolvidas no projeto Apache CloudStack criaram uma equipa de segurança responsável por lidar com todas as vulnerabilidades detetadas pela comunidade. A equipa de segurança da CloudStack trabalha proximamente com a equipa de segurança da Apache [65] [66].

O CloudStack fornece um acesso seguro através de HTTPS, autenticação cifrada através do par utilizador/password, chave pública/privada ou LDAP.

2.2.3.3 Compatibilidade com os Serviços Web da Amazon

O CloudStack é compatível com os serviços AWS EC2 e S3. Como este é um projeto ainda em desenvolvimento e com uma expectável margem de crescimento, prevê-se que este em breve seja compatível com mais serviços da AWS.

2.2.4 OpenStack

Já foram abordados vários softwares de implementação e gestão de Clouds do tipo de infraestrutura ao longo deste capitulo 2, no entanto o tema central desta tese é um software em específico: OpenStack. Este é um software open source para implementação e gestão de Clouds [17], capaz de gerir os componentes das múltiplas infraestruturas do OpenStack. O seu nome faz-se cumprir na integra, isto é, o OpenStack é um conjunto de software open source, usados para configurar e gerir toda a infraestrutura de computação e armazenamento da Cloud. Embora todo este projeto seja open source, existiram duas contribuições iniciais maioritárias no desenvolvimento deste projeto, a RackSpace (forneceu o seu projeto Cloud Files para tornar viável a implementação do Object Storage) e a NASA (forneceu acesso ao projeto Nebula, para que fosse possível implementar o lado da computação). A API do OpenStack é compatível com os serviços EC2 e S3 da Amazon AWS.

2.2.4.1 Principais Características e Componentes

As principais características do OpenStack são: tratar tudo como serviço, escalabilidade, uma interface web muito completa e um suporte gigante por parte da comunidade.

No OpenStack existem quatro serviços fundamentais que constituem o seu núcleo:

- Infraestrutura Computacional (Nova)
- Infraestrutura de Armazenamento (Swift)
- Gestão de Imagens (Glance)

- Infraestrutura de Rede (Neutron)

É possível que a gestão seja feita através da própria CLI do sistema, fazendo a autenticação utilizando as credenciais do keystone, ou utilizando o painel de gestão online (dashboard horizon), utilizando também as mesmas credenciais do keystone, sendo que a gestão se torna mais agradável e simples quando utilizado o dashboard horizon.

2.2.4.2 Segurança

O OpenStack permite que o administrador de sistemas defina o nível de encriptação de objetos uma vez que é possível utilizar-se AES 128/192/256, DES, ou RSA [67]. O OpenStack fornece um acesso seguro através de HTTPS, a autenticação é feita através do KeyStone, sendo este integravel com LDAP, OAuth, Open ID ou SAML.

2.2.4.3 Compatibilidade com os Serviços Web da Amazon

O OpenStack tal como o CloudStack, é compatível com os serviços AWS EC2 e S3. Como este é um projeto em constante desenvolvimento, prevê-se que este em breve seja compatível com mais serviços da AWS.

No entanto, neste capítulo não irei dar entrar em muito detalhe sobre o OpenStack nem sobre nenhum serviço, pois irá ser abordado detalhadamente no próximo capítulo.

2.3 Comparativo

Após identificadas as principais soluções do foro open source, realizou-se uma análise comparativa entre elas, podendo assim verificar de forma mais simplista as vantagens e desvantagens de cada e refletir sobre a escolha do OpenStack como a melhor opção.

Na tabela 2.2 é possível verificar alguma informação geral sobre cada um dos softwares, como a linguagem utilizada no desenvolvimento do software ou a compatibilidade com a Amazon AWS. É de notar que todos os softwares têm compatibilidade (total ou parcial) com os serviços da Amazon AWS. Isto acontece uma vez que a Amazon foi a primeira grande empresa a explorar a computação na Cloud, pelo que é vista como um padrão a seguir. Verifica-se que o Eucalyptus não possui uma API própria, ao contrário de todos os outros softwares. Como referido anteriormente, o Eucalyptus corre no seu topo a API da Amazon AWS.

Na tabela 2.3 é possível verificar os hypervisors que cada um dos softwares descritos suporta. Um hypervisor tem um papel fulcral num sistema de computação na cloud uma vez que são estes que permitem aplicar diversas técnicas de gestão e controlo de virtualização, como o tipo de armazenamento utilizado que é escolhido [68].

Tabela 2.1	Ano Criação	Licença	Linguagem Programação	Compatibilidade AWS
OpenNebula	2005	Apache 2.0	C, C++, Ruby, Java	Sim
Eucalyptus	2008	GPLv3	Java, C	Sim
CloudStack	2011	Apache 2.0	Java	Sim
OpenStack	2010	Apache 2.0	Python	Parcial

Tabela 2.2: Informações Gerais sobre os softwares open source abordados

Tabela 2.2	KVM	XEN	ESXi	Hyper-V	LXC	Docker
OpenNebula	Sim	Sim	Sim	Não	Não	Não
Eucalyptus	Sim	Não	Sim	Não	Não	Não
CloudStack	Sim	Sim	Sim	Não	Não	Não
OpenStack	Sim	Sim	Sim	Sim	Sim	Sim

Tabela 2.3: Diferentes hypervisors suportados pelos softwares open source abordados

Claramente de todos os softwares, o OpenStack é o que oferece maior destaque no que diz respeito à compatibilidade com hypervisors, sendo também o único compatível com o Hyper-V , o sistema de virtualização desenvolvido pela Microsoft [69] bem como é o único compatível com Docker, tecnologia open source que permite criar, executar, testar e implementar aplicações distribuídas dentro de contentores de software, sendo estes contentores um método de virtualização de um sistema operativo [19].

Na tabela 2.4 é possível verificar um comparativo entre as diferentes funcionalidades de rede, como suporte a VLAN, SDN, alocação de floating ip's (são endereços atribuídos pelo Neutron para o acesso às máquinas através da rede interna ou externa, reutilizáveis quando removidos de uma instância) ou grupos de segurança, funcionando estes como firewall, servindo para que se configure regras específicas para certas instâncias ou então se configure grupos de segurança de acordo com as regras de segurança pretendidas para uma ou várias instâncias virtuais. É possível verificar que todos os softwares suportam todas as funcionalidades comparadas na tabela 2.3, à exceção do Eucalyptus que não suporta redes definidas por software (SDN).

Tabela 2.3	VLAN	SDN	Floating IP	Grupos de Segurança
OpenNebula	Sim	Sim	Sim	Sim
Eucalyptus	Sim	Não	Sim	Sim
CloudStack	Sim	Sim	Sim	Sim
OpenStack	Sim	Sim	Sim	Sim

Tabela 2.4: Funcionalidades de rede suportadas pelos softwares open source abordados

O que se pretendia com esta tese era a instalação do OpenStack para a construção de uma Cloud do tipo IaaS, através desta comparação compreende-se claramente que o OpenStack é o que suporta mais funcionalidades no geral, embora a nível de compatibilidade com os serviços da AWS seja o que de momento oferece menos compatibilidade quando comparando com os restantes.

A este fator deve-se o facto de este ainda estar em desenvolvimento, no entanto espera-se que num futuro próximo este já seja totalmente compatível com a AWS [70]. As vantagens do OpenStack face aos outros não se resumem a vantagens a nível de compatibilidade: é o que tem a maior comunidade e possui uma arquitetura altamente modular. Como desvantagem do OpenStack surge uma instalação complexa, uma vez que não existe nenhuma ferramenta para uma instalação do tipo *one click*. Como tal é necessário conhecer cada um dos seus serviços na íntegra para que se perceba como os interligar. Devido a isto a uma curva de aprendizagem é muito elevada, face aos outros softwares open source, criando-se assim um desafio de grau elevado.

Capítulo 3

OpenStack

O OpenStack surgiu da procura de uma solução de IaaS, em 2010, através de uma colaboração entre a NASA e a RackSpace [71]. Devido ao projeto Nebula, da NASA, tinha sido desenvolvido uma ferramenta chamada Nova, utilizada pela NASA para computação e Cloud. Por outro lado a RackSpace tinha desenvolvido o Cloud Files, sendo este uma plataforma para armazenamento na Cloud, equivalente à plataforma S3 da Amazon. Aliando ambas as ferramentas, surgiu assim uma parceria de seu nome OpenStack. O Nova foi disponibilizado para este projeto, sob licença Apache [72] e o Cloud Files foi utilizado como base do Swift.

Atualmente o OpenStack pertence à OpenStack Fundation, e por ser open source conta com apoios gigantes da indústria e da comunidade, no seu desenvolvimento.

3.1 Descrição

O OpenStack pode ser visto como um sistema de operação em Cloud, possibilitando aos utilizadores aprovisionarem quantos recursos necessitarem desde que disponíveis, através de um dashboard (horizon), ou através de CLI. Nas versões iniciais do OpenStack, a sua arquitetura era muito pouco modular, sendo que os diversos componentes que o constituíam atuavam como um único módulo. Atualmente a sua arquitetura 3.1 é modular e distribuída. A versão mais recente e estável lançada (em Outubro de 2016) e a que foi utilizada para a implementação do modelo de Cloud proposto nesta tese é a versão Newton [73].

3.2 Serviços

O OpenStack tem uma arquitetura modular, sendo esta constituída por diversos serviços. Os principais serviços utilizados, visíveis na figura 3.1 são: Ceilometer, Cinder, Glance, Heat, Horizon, KeyStone, Neutron, Nova, Sahara, Swift, Trove.

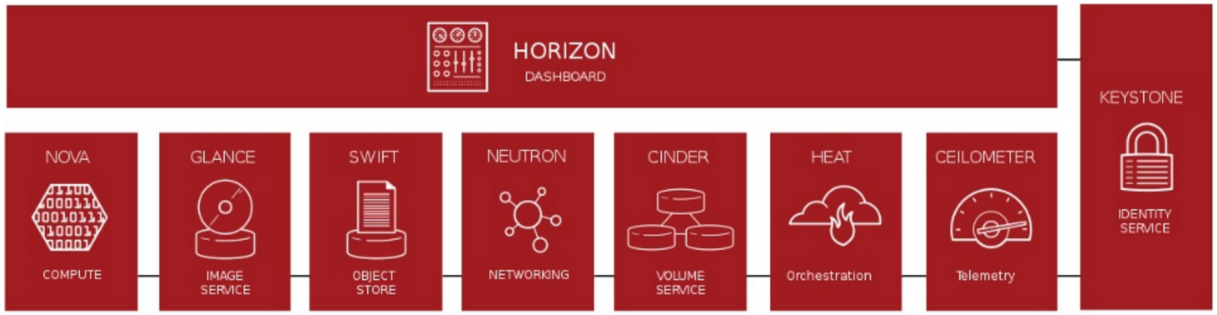


Figura 3.1: Figura representativa da arquitetura modular do OpenStack. Imagem retirada de [9].

3.2.1 Ceilometer

O Ceilometer é um serviço que tem como objetivo centralizar a monitorização e métrica de recursos para o OpenStack. Como tal este serviço recolhe, normaliza e transforma de forma eficiente os dados produzidos pelos restantes serviços do OpenStack. Este é também utilizado para definir um limite temporal na utilização dos recursos por parte dos utilizadores. É utilizado o MongoDB para o armazenamento dos dados. O serviço equivalente da Amazon AWS é o CloudWatch.

3.2.2 Cinder

O Cinder é um serviço que serve como solução de armazenamento em blocos. Este é responsável pelos volumes utilizados para armazenamento persistente e independente das instâncias de máquinas virtuais do OpenStack, sendo este similar ao Amazon EC2 Elastic Block Storage. Tipicamente este serviço é comparado a uma memória USB, podendo ser associado a diferentes instâncias. No entanto, apenas pode ser associado a uma instância de cada vez, tal como uma memória USB. Com o Cinder é possível utilizarem-se diversos tipos de drivers para backend, dependendo do dispositivo de armazenamento que se pretenda. Para a implementação desta tese foi utilizado um driver NFS. O serviço equivalente da Amazon AWS é o Elastic Block Storage (EBS).

3.2.3 Glance

O Glance é um serviço que serve para o registo e distribuição de imagens no OpenStack, imagens estas com os sistemas operativos a serem utilizados pelas instâncias de máquinas virtuais. Com o OpenStack é possível criar uma imagem de uma instância em execução, sendo que quando criada, esta é adicionada como uma imagem do Glance, podendo até ser utilizado este snapshot em outras máquinas virtuais.

Este serviço está dividido por três camadas sendo estas: a) uma base de dados que armazena

a informação produzida pelo registry; b) pela Glance Store, camada através da qual todas as operações passam dado fornecer uma interface para acesso ao armazenamento e base de dados das imagens; c) por um backend de armazenamento, sendo que o modo como as imagens são guardadas depende do driver utilizado. No nosso caso o driver utilizado foi o do Swift, sendo que o Glance é também compatível com Ceph. O serviço equivalente da Amazon AWS é o Amazon Machine Image (AMI).

3.2.4 Heat

O Heat é um serviço que serve para fazer instalação automática de aplicações distribuídas no OpenStack. Para tal são utilizados templates. Estes têm o mesmo formato dos templates utilizados pela Amazon para o mesmo tipo de serviço. Cada um destes deve ter a informação correspondente à Infraestrutura para a aplicação distribuída (como volumes, *floating ip's*, grupos de segurança), sendo que quando executado o serviço, este faz as chamadas necessárias às APIs dos diferentes serviços necessários para que a Infraestrutura seja criada. Caso seja necessário fazer uma alteração basta editar o ficheiro original de template. O serviço equivalente da Amazon AWS é o Cloud Formation.

3.2.5 Horizon

O Horizon é o *dashboard* oficial do OpenStack que serve para se gerirem serviços do OpenStack como o Nova, Neutron, Swift, Glance, etc, agilizando e facilitando o processo da gestão da Cloud. Através deste é possível aos administradores gerirem os diferentes recursos e aos utilizadores aprovisionarem os seus próprios recursos de forma *self-service* até a um certo limite definido para si, utilizando para tal esta interface web, em vez da tradicional linha de comandos, que poderia tornar-se complicada para simples utilizadores. Este é uma ferramenta completa que permite um controlo total de todos os serviços da Cloud, como criar *routers*, gerir e adicionar imagens, criar e importar chaves SSH para a autenticação de utilizadores nas instâncias, adição de volumes a instâncias, gestão dos grupos de segurança, atribuição ou remoção de *floating ip's* e criação de redes virtuais, o que torna o processo de gestão da Cloud mais ágil quando comparado com a linha de comandos. O serviço equivalente da Amazon AWS é o Console.

3.2.6 KeyStone

O KeyStone é o serviço responsável pela autenticação, gestão de identidades e projetos. Este serviço é crucial pois sustenta a verificação e listagem de serviços disponíveis a todos os outros serviços do OpenStack, sendo que este deve ser o primeiro serviço referente ao *core* a ser instalado. Para tal é utilizado um *token*, portanto a configuração do serviço reside maioritariamente na criação de utilizadores e regras de autorização para os serviços associados ao OpenStack, bem como na criação de contas de utilizador. Cada utilizador pode estar em vários projetos, sendo

que pode ter diferentes permissões para cada projeto, permissões estas que são geridas pelo KeyStone.

Existem 3 conceitos referentes ao KeyStone: inquilinos, regras e utilizadores. Um inquilino pode ser visto como uma espécie de projeto ao qual se associam recursos como utilizadores, imagens, redes e instâncias, associadas a esse mesmo projeto, podendo ser únicas sem que estejam disponíveis em mais nenhum projeto. Um utilizador pode pertencer a vários projetos, e pode ter permissões de administração ou ser apenas um utilizador normal, dependendo da regra que lhe foi atribuída. O serviço equivalente da Amazon AWS é o Identity and Access Management (IAM).

3.2.7 Neutron

O Neutron é o serviço de rede do OpenStack e foi criado para fornecer rede como um serviço (Software-Defined Networking - SDN) entre dispositivos de interface geridos por outros serviços, como por exemplo pelo Nova. Assim é possível que se abstraia a camada de controlo (onde é decidido para onde o tráfego deve ser dirigido) da camada de dados (onde se direciona o tráfego para o destino). Com isto garante-se uma enorme flexibilidade, podendo assim utilizar-se redes complexas em ambientes seguros multi-inquilinos. O facto de apenas ser possível utilizar um plugin no Neutron faz com que os dispositivos tenham que ser do mesmo fabricante, o que poderá ser visto como uma desvantagem. O Neutron é visto como tendo uma arquitetura modular, pelo que é possível a qualquer momento conectar um novo switch, ou configurar funções de firewall como serviço. Como tudo é definido através de software, temos um controlo completo de toda a infraestrutura de rede do OpenStack.

O Neutron é composto por vários componentes como neutron-api, neutron-conductor, neutron-compute, neutron-scheduler e BD, sendo que a camada de software no OpenStack é criada utilizando o Open vSwitch (OVS). O serviço equivalente da Amazon AWS é o Networking.

3.2.8 Nova

O Nova é um dos serviços principais e mais complexos do OpenStack. Permite iniciar e parar instâncias virtuais, atribuir e remover floating ip's, criar Snapshots de instâncias, criar e editar grupos de segurança e adicionar ou remover volumes. Na versão Newton o Nova divide-se em vários componentes como *nova-api, nova-conductor, nova-compute, nova-scheduler* e base de dados. No processo de criação de uma nova instância, após o utilizador escolher o *flavor* a utilizar, o *nova-schedule* escolhe o nó de computação que deve ser utilizado, de seguida o nova-compute obtém a imagem selecionada pelo utilizador a partir do Glance e, utilizando o *hypervisor*, procede à criação de uma nova instância. É também o *nova-compute* que cria um disco para servir como armazenamento da instância, no entanto este será destruído a quando a eliminação da instância. O serviço equivalente da Amazon AWS é o Compute.

3.2.9 Sahara

O Sahara é um serviço que é essencialmente dedicado a quem procura análise de grandes quantidades de dados (data mining). Com este serviço é possível rapidamente serem criados clusters para computação, facilitando assim o balanceamento da carga entre eles sem que seja necessário ao utilizador preocupar-se com a gestão dos clusters. Os dados podem ser obtidos através de diversas fontes, como bases de dados, Swift, HDFS ou NoSQL. O serviço equivalente da Amazon AWS é o Elastic Map Reduce (EMR).

3.2.10 Swift

O Swift é um serviço do OpenStack que também é conhecido como Object Storage. Este é visto como uma solução distribuída, redundante e altamente escalável de armazenamento, sendo que a autenticação e autorização para leitura e escrita de dados é gerida pelo KeyStone. Com o Swift é possível criar-se um cluster de armazenamento, destinado a grandes quantidades de dados não estruturados, como backups, imagens, documentos ou vídeos. O Swift foi implementado de forma análoga ao Amazon S3, e pode alojar objetos virtuais de tamanho ilimitado, desde que o hardware o permita. Como definição padrão, são criadas 3 réplicas para cada objeto armazenado no Swift. O serviço equivalente da Amazon AWS é o Simple Storage Service (S3).

3.2.11 Trove

O Trove é utilizado no OpenStack para base de dados como serviço. Este foi desenhado para que os utilizadores utilizem de forma rápida e eficiente os recursos de uma base de dados, seja esta relacional ou não, sem terem que se preocupar com tarefas de gestão complexas sobre a mesma. Os utilizadores podem assim aprovisionar várias instâncias de bases de dados conforme necessitem. Este serviço fornece isolamento de recursos proporcionando alto desempenho. Este serviço garante aos utilizadores elasticidade nas suas bases de dados quando necessário sendo que o utilizador vê o *cluster* como uma única base de dados. Este efetua cópias de segurança de forma automática e lida com falhas de nós de forma automática. O serviço equivalente da Amazon AWS é o RDS.

3.3 Implementação do modelo de Cloud privada utilizando o OpenStack

Após toda a fase de estudo do OpenStack e dos serviços que o constituem seguiu-se a fase de instalação num sistema de testes controlado para que, mais tarde, quando já tivessem sido resolvidos todos os problemas que pudessem eventualmente ocorrer durante instalação do

OpenStack nas máquinas de testes, se pudesse migrar para o ambiente de produção do INESC TEC.

Como um dos objetivos é a alta disponibilidade dos serviços, foram escolhidas 3 máquinas que servem como controllers, e 1 para compute, sendo que o pretendido para a cloud do INESC TEC serão 3 controllers, diferenciando apenas no número de nós de computação. Cada controlador contem os serviços descritos na secção 4, instalados e configurados da mesma forma que a descrita.

As máquinas utilizadas como teste têm todas as mesmas características: CPU AMD Athlon(tm) 64 X2 Dual Core Processor 5600+ de 2.8 Ghz e 4 Gb de RAM DDR2. A escolha destas máquinas deveu-se ao material que existia disponível. A versão do OpenStack escolhida foi a mais recente disponibilizada à comunidade, a Newton. A infraestrutura descrita e instalada está representada na figura 3.2.

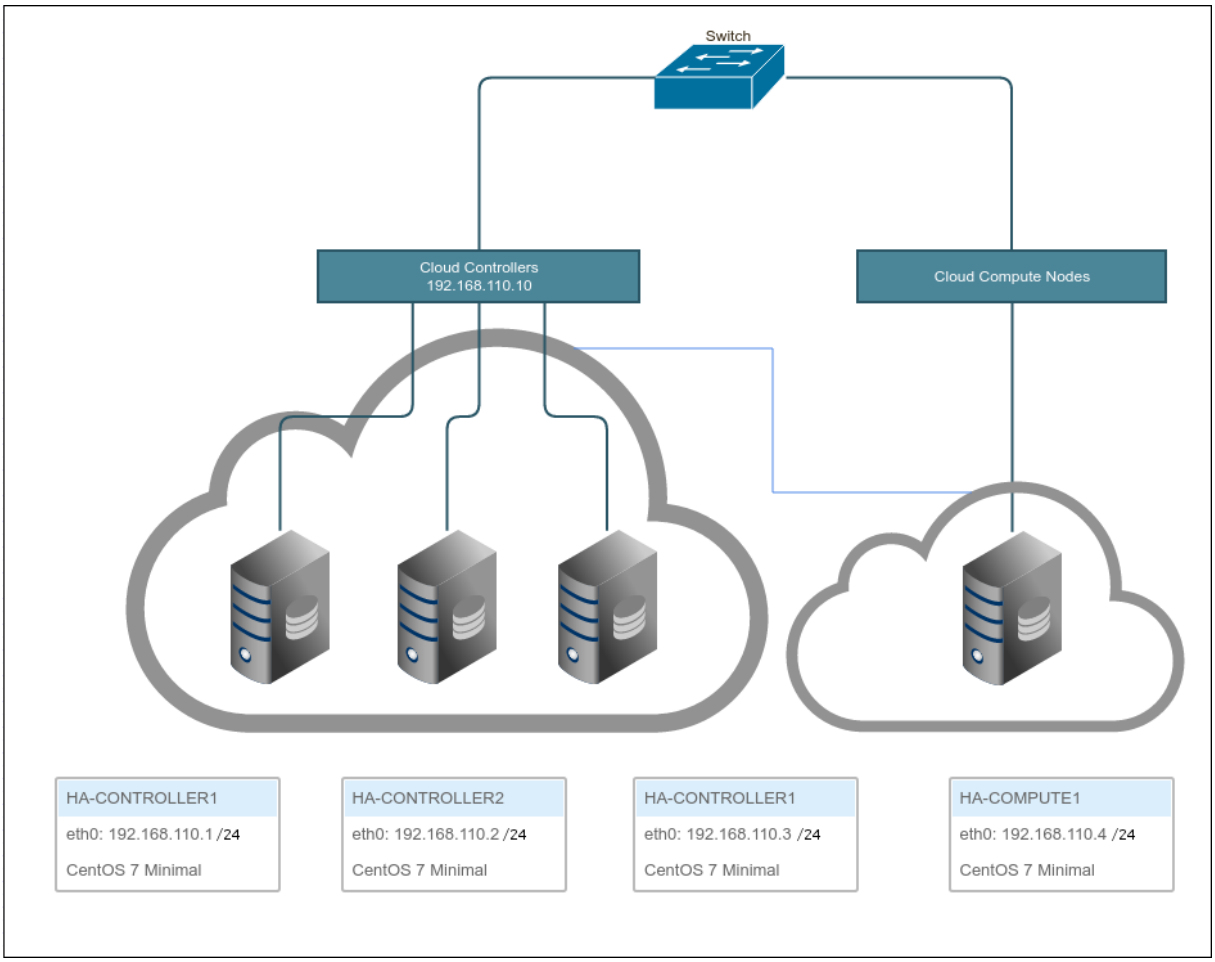


Figura 3.2: Esquema representativo da infraestrutura utilizada para a criação da Cloud de teste.

3.3.1 Organização dos Serviços

Para a instalação dos vários serviços do OpenStack foram criados clusters, isto é, a instalação de várias instâncias de cada serviço pelos diferentes controllers, utilizando um balanceador de carga para fazer a respetiva gestão da carga entre os controllers. Foi escolhido este modelo devido ao objetivo desta implementação ser a alta disponibilidade. Assim, com a utilização de um cluster aliado a balanceamento de carga, conseguimos cumprir o objetivo a que nos propusemos.

3.3.1.1 Balanceamento de Carga

De forma a que o tráfego fosse corretamente distribuído pelos diferentes controladores foi necessário recorrer a um mecanismo de balanceamento de carga para alta disponibilidade. A escolha recaiu sobre o software open-source recomendado para utilização conjunta com o OpenStack, o HAProxy. Este software executa uma monitorização ativa dos serviços, direcionando assim o tráfego para os serviços ativos, detetando possíveis falhas, e detetando-as, invalidando o serviço para o nó onde foi detetado, impedindo que seja enviado o tráfego para esse nó com o serviço inválido. O HAProxy suporta diferentes algoritmos de balanceamento, sendo que estes devem ser usados de acordo com a finalidade a que se destinam. No nosso caso em particular, foi utilizado *round robin*: utiliza todos os servidores de acordo com os seus pesos.

No entanto a utilização de um software para balancear a carga pode ser vista como um ponto de falha, uma vez que se o HAProxy falhar, o tráfego deixa de ser enviado. Para se solucionar este problema utilizou-se o Keepalived, que à semelhança do HAProxy, é também um software destinado a balanceamento de carga para alta disponibilidade. Com o Keepalived conseguimos criar um cluster de HAProxy. O Keepalived utiliza VRRP (Virtual Router Redundancy Protocol), permitindo que se crie um cluster do tipo Ativo/Passivo. Para tal é atribuido um IP Virtual (VIP) ao nó definido como master e, em caso de falha, o Keepalived automaticamente muda o endereço VIP para outro nó.

3.3.1.2 Base de dados

A nível de base de dados, a solução escolhida foi a utilização de um cluster MariaDB, utilizando para tal Percona XtraDB Cluster [74], sendo que este agrega num único pacote a biblioteca Galera, responsável pela alta-disponibilidade no MariaDB e o XtraBackup, responsável pela replicação entre nós do cluster. Com estes conseguimos que haja replicação das bases de dados de forma sincronizada sendo que os acessos são geridos pelo HAProxy. Como é conhecido um problema no Galera com o HAProxy [75], foi definido que apenas um nó do Galera é acedido, enquanto um dos outros apenas assume a função caso o anterior falhe, pelo que o modelo utilizado é o *active-hot standby one*.

3.3.1.3 Troca de Mensagens

Para que a comunicação seja conseguida entre os diferentes serviços do OpenStack é utilizado um *Message Oriented Middleware* (MoM), o RabbitMQ, que utiliza como base o protocolo AMQP [76]. Este é a solução proposta por defeito para a utilização do OpenStack e é vocacionado para soluções em alta disponibilidade. Como são conhecidos alguns problemas de compatibilidade entre o HAProxy e o RabbitMQ [76], os controllers devem ser fornecidos juntamente com os *rabbit_hosts e rabbit_ha_queues* conectados diretamente a um ou mais servidores que corram o RabbitMQ, sendo que eu caso de falha as re-conexões são tratadas automaticamente.

3.3.1.4 Armazenamento

O serviço recomendado pelo OpenStack para o backend de armazenamento recomendado é o Swift, no entanto este é compatível com vários backends de armazenamento como por exemplo Ceph. Neste modelo de instalação do OpenStack proposto é utilizado como backend o Swift, que tal como já anteriormente descrito em 3.2.10, idealizado para armazenamento em grande escala e pode ser aumentado infinitamente, desde que existam recursos para tal. O facto de a nossa escolha ter recaído sobre o Swift deveu-se ao facto de se pretender conhecer a integração de todos os serviços do OpenStack, pelo que deste modo foi necessário perceber como integrar três backends distintos: Swift, Glance e Cinder.

3.3.1.5 Autenticação, Funções e Listagem de Serviços

O KeyStone, tal como já referido em 3.2.6, não serve apenas para autenticação. Este atua com quatro tipos diferentes de informação: autenticação (*tokens)*, listagem de serviços (*catalog)*, funções de utilizadores (*assignement)* e identificação (*identity)*. Assim para a criação de *tokens* foi utilizado o plugin PKI (*Public Key Infrastructure)*, que embora não tenha grande relevância para a instalação do OpenStack num ambiente de teste, será deveras importante quando se passar para a instalação nos servidores de produção, fazendo com que a segurança seja aumentada, pois os *tokens* passam a ser assinados digitalmente. Para o armazenamento dos restantes tipos de informação foi utilizado MySQL como backend pois nenhum destes oferecia incompatibilidade ao mesmo, sendo inclusive o recomendado.

3.3.1.6 Arquitetura de Rede

Após a análise efetuada sobre o Neutron decidiu-se optar pela utilização do *plugin* ML2 (Modular Layer 2) juntamente com o agente Open vSwitch. O ML2 é um *framework* que permite que sejam utilizadas simultaneamente diferentes tecnologias da camada 2 de rede dispostas por datacenters de todo o mundo. Este plugin é compatível com três agentes: OpenvSwitch, linuxbridge, e hyperv L2. A escolha recaiu sobre o agente OpenvSwitch tendo sido esta meramente opcional devido à comunidade recomendar maioritariamente o OpenvSwitch. O plugin ML2 utiliza o agente para

criar um switch virtual, sendo que através deste cria uma camada do tipo *layer 2* juntamente com *bridges*, aumentando assim a simplicidade na camada de dados (*layer 2)* e permitindo que se conectem facilmente instâncias.

3.3.2 Estratégias para alta disponibilidade utilizadas

A tabela 3.1 reflete um resumo da estratégia de alta disponibilidade descrita na secção 3.3.1 utilizada sobre cada um dos serviços detalhados na secção 3.2, tendo sido estes devidamente instalados e configurados em cada controlador. Através da tabela conseguimos perceber que tipo de estratégia é aplicada a cada serviço para se obter alta disponibilidade. É visível que na sua grande maioria, a alta disponibilidade dos serviços é obtida recorrendo ao HAProxy. No entanto como já descrito anteriormente em 3.2 nem sempre foi possível recorrer ao HAProxy pelo que tivemos que encontrar uma solução para quando o seu uso não era viável. A solução passou pelo uso de keepalived, redis e a utilização de clusters.

	HAProxy	**Cluster**	**Keepalived**	**Redis**
MySQL	X	X		
RabbitMQ		X		
HAProxy			X	
MongoDB		X		
Redis		X		
KeyStone	X			
Glance	X			
Nova	X			
Cinder	X			
Neutron	X			
Horizon	X			
Ceilometer	X			X
Heat	X			
Swift	X			
Sahara	X			
Trove	X			

Tabela 3.1: Tipos de estratégias HA utilizadas pelos serviços instalados

Capítulo 4

Implementação dos Controladores

A proposta de implementação que se segue está sub-dividida em duas partes: serviços do OpenStack que pertencem ao seu core e serviços do OpenStack que não pertencem ao seu core. Inicialmente instalamos os serviços non-Core, sendo que de seguida instalamos os serviços do Core do OpenStack. Os serviços estão ilustrados na tabela 4.

Core	**Non-Core**
KeyStone	HAProxy
Glance	Keepalived
Cinder	Galera
Swift	RabbitMQ
Neutron	MemCached
Nova	Redis
Ceilometer	MongoDB
Heat	
Horizon	
Trove	
Sahara	

Tabela 4.1: Serviços pertencentes ou não ao Core do OpenStack a serem instalados

No capítulo anterior em 3.2 foi ilustrada a arquitetura de rede que foi utilizada. Para tal foram utilizados três servidores com a função de controladores, e um servidor com a função de nó para computação. Para se aceder às máquinas para se fazerem as configurações iniciais de cada interface foi utilizada a rede do departamento atribuída por DHCP, sendo que após as configurações iniciais esta deixou de ser utilizada. Da imagem referida anteriormente podemos dividi-la em duas componentes: a dos controladores e a da computação. Para este capítulo a componente relevante é a representada na figura 4.1, em que podemos verificar os três servidores (HA-CONTROLLER 1,2,3), sendo que a sua interligação é feita através do HAProxy (4.1) e do seu endereço IP virtual obtido pelo keepalived (4.2) e cada controlador possui instalado os serviços referidos na tabela 4. O HAProxy (High Availability Proxy) e o Keepalived são ambos

softwares gratuitos, sendo que o primeiro é utilizado para alta disponibilidade, balanceamento de carga e proxy para aplicações que utilizam TCP e HTTP [77] e o segundo é utilizado em alta disponibilidade para infraestruturas baseadas em Linux.

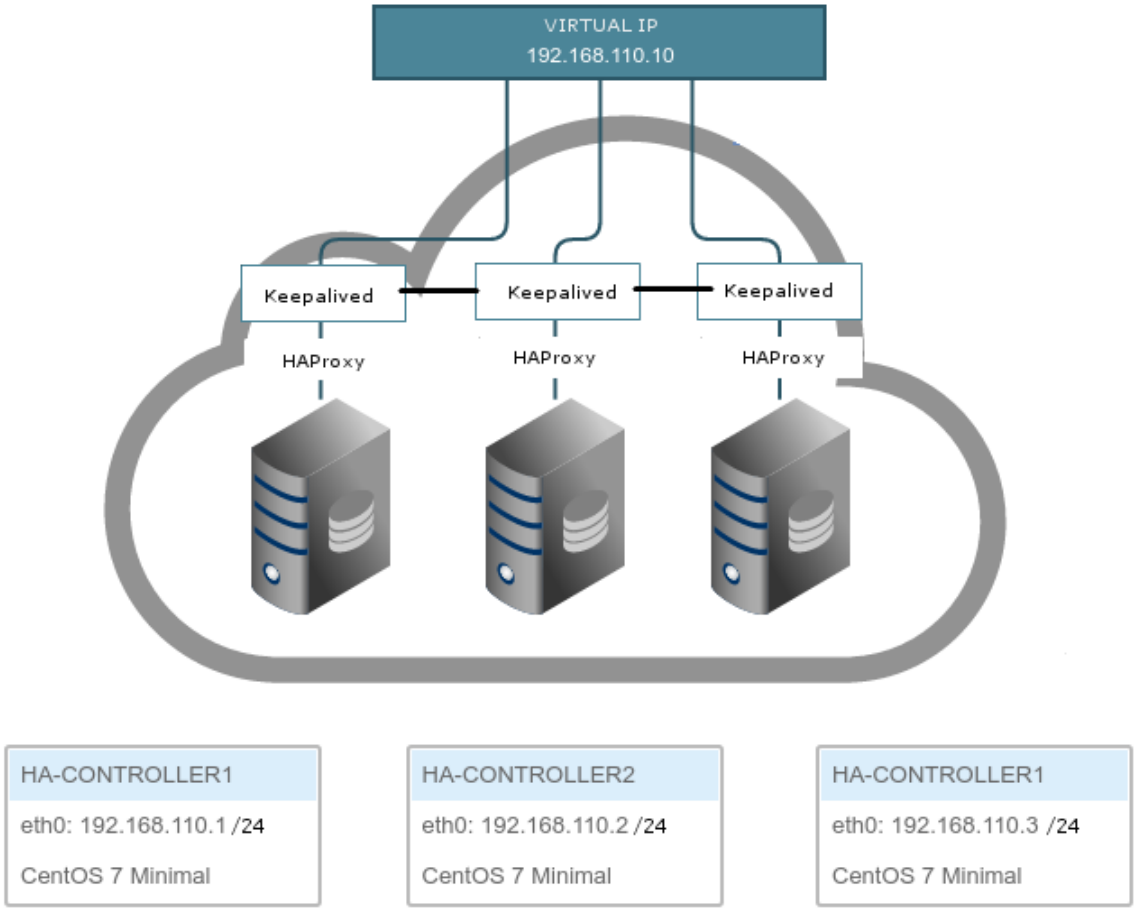


Figura 4.1: Figura representativa da componente dos controladores instalada.

Estes servidores têm o seguinte endereço IP e respetivo nome na rede 192.168.110.0/24:

```
192.168.110.1  HA-CONTROLLER1
192.168.110.2  HA-CONTROLLER2
192.168.110.3  HA-CONTROLLER3
192.168.110.4  HA-COMPUTE1

192.168.110.10 VIP
```

Como pré requisito para a instalação de qualquer serviço é necessário adicionar o repositório do RDO [78] para conseguirmos instalar os pacotes.

```
1 yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
```

É importante referir que à medida que se vão instalando os serviços é possível acompanhar a sua correta configuração através do interface online do HAProxy após este estar configurado. O termo *ADDRESS* deverá ser substituído pelo endereço IP do respetivo controlador.

4.1 HAProxy

O HAProxy (High Availability Proxy) é um software gratuito para alta disponibilidade, balanceamento de carga e proxy para aplicações que utilizam TCP e HTTP [77]. O HAProxy assume um papel crucial na nossa implementação, sendo que na configuração dos serviços foi-lhe configurado cada serviço por controlador. O balanceamento de carga pode ser feito através do algoritmo de round-robin, leastconn ou source. O round-robin é o algoritmo por defeito utilizado no HAProxy, sendo que a escolha do servidor é feita por turnos. O leastconn é um algoritmo em que o servidor escolhido é o que tem menos ligações, sendo este recomendado para conexões longas. O source é um algoritmo em que o cliente utiliza sempre o mesmo servidor por defeito. Para a nossa implementação o método mais indicado é o round-robin uma vez pretendemos acessos aos diferentes servidores de forma balanceada e com conexões de curta duração. Caso um controlador esteja em falha, o HAProxy coloca-o em estado de falha e deixa de encaminhar pedidos por ele, até que este esteja recuperado e operacional.

Um esquema simples do HAProxy está representado na figura 4.2 em que é possível verificar que um utilizador efetua um pedido através do endereço IP virtual e este entrega o pedido ao HAProxy disponível que por sua vez encaminha o tráfego de acordo com o algoritmo de balanceamento de carga, sendo que posteriormente devolve a resposta ao utilizador. Deste modo é possível suportar a falha de até dois controladores, numa dada janela temporal.

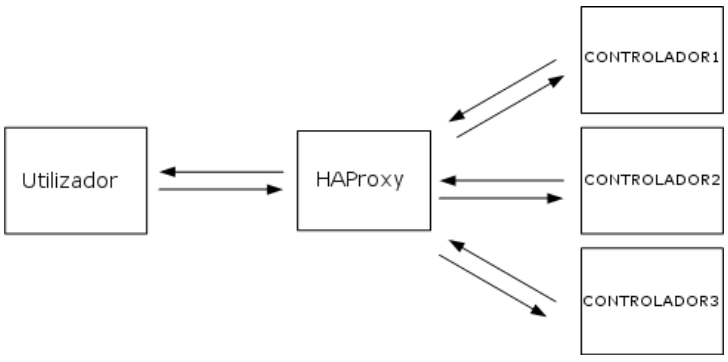


Figura 4.2: Figura representativa do HAProxy na implementação efetuada.

Os seguintes passos para a instalação do HAProxy devem ser executados em todos os controladores até que seja afirmado o contrário. O bloco de código [1-3] apresentado de seguida instala os pacotes necessários para a execução bem como ativa as opções de encaminhamento de pacotes entre diferentes controladores e permite a ligação dos serviços aos diferentes controladores.

```
yum install -y haproxy openstack-selinux
echo net.ipv4.ip_nonlocal_bind=1 >> /etc/sysctl.d/haproxy.conf
echo 1 > /proc/sys/net/ipv4/ip_nonlocal_bind
```

O bloco de código [1-19] apresentado de seguida define os parâmetros de atuação do HAProxy, o tempo sem resposta máximo, o número máximo de conexões, a localização da socket de dados, o endereço do controlador, o protocolo de comunicação e controlo de transmissão. Cada um dos blocos de código [20-29], [31-41], [43-52], [54-63], [65-72], [74-81], [83-90], [92-99], [101-108], [110-118], [120-127], [129-136], [138-148], [150-157], [159-166], [168-175], [177-186], [188-195] e [197-204] apresentados de seguida é responsável por receber os pedidos de cada cliente para cada serviço que o OpenStack irá utilizar. Por exemplo para o bloco [20-31] é definido o endereço VIP com a porta associada ao serviço de base de dados, o tempo sem resposta máximo e o algoritmo de balanceamento utilizado (round-robin).

```
cat > /etc/haproxy/haproxy.cfg << EOF
global
daemon
stats socket /var/lib/haproxy/stats
defaults
mode tcp
maxconn 10000
timeout connect 5s
timeout client 30s
timeout server 30s
listen monitor
bind ADDRESS:9300
mode http
monitor-uri /status
stats enable
stats uri /admin
stats realm Haproxy\ Statistics
stats refresh 5s

frontend vip-db
bind 192.168.110.10:3306
timeout client 90m
default_backend db-vms-galera
backend db-vms-galera
balance roundrobin
timeout server 90m
server controlador1 192.168.110.1:3306 check inter 1s
server controlador2 192.168.110.2:3306 check inter 1s
server controlador3 192.168.110.3:3306 check inter 1s

frontend vip-rabbitmq
```

```
32 option clitcpka
33 bind 192.168.110.10:5672
34 timeout client 900m
35 default_backend rabbitmq-vms
36 backend rabbitmq-vms
37 balance roundrobin
38 timeout server 900m
39 server controlador1 192.168.110.1:5672 check inter 1s
40 server controlador2 192.168.110.2:5672 check inter 1s
41 server controlador3 192.168.110.3:5672 check inter 1s
42
43 frontend vip-keystone-admin
44 bind 192.168.110.10:35357
45 default_backend keystone-admin-vms
46 timeout client 600s
47 backend keystone-admin-vms
48 balance roundrobin
49 timeout server 600s
50 server controlador1 192.168.110.1:35357 check inter 1s
51 server controlador2 192.168.110.2:35357 check inter 1s
52 server controlador3 192.168.110.3:35357 check inter 1s
53
54 frontend vip-keystone-public
55 bind 192.168.110.10:5000
56 default_backend keystone-public-vms
57 timeout client 600s
58 backend keystone-public-vms
59 balance roundrobin
60 timeout server 600s
61 server controlador1 192.168.110.1:5000 check inter 1s
62 server controlador2 192.168.110.2:5000 check inter 1s
63 server controlador3 192.168.110.3:5000 check inter 1s
64
65 frontend vip-glance-api
66 bind 192.168.110.10:9191
67 default_backend glance-api-vms
68 backend glance-api-vms
69 balance roundrobin
70 server controlador1 192.168.110.1:9191 check inter 1s
71 server controlador2 192.168.110.2:9191 check inter 1s
72 server controlador3 192.168.110.3:9191 check inter 1s
73
74 frontend vip-glance-registry
75 bind 192.168.110.10:9292
76 default_backend glance-registry-vms
77 backend glance-registry-vms
78 balance roundrobin
```

```
79 server controlador1 192.168.110.1:9292 check inter 1s
80 server controlador2 192.168.110.2:9292 check inter 1s
81 server controlador3 192.168.110.3:9292 check inter 1s
82
83 frontend vip-cinder
84 bind 192.168.110.10:8776
85 default_backend cinder-vms
86 backend cinder-vms
87 balance roundrobin
88 server controlador1 192.168.110.1:8776 check inter 1s
89 server controlador2 192.168.110.2:8776 check inter 1s
90 server controlador3 192.168.110.3:8776 check inter 1s
91
92 frontend vip-swift
93 bind 192.168.110.10:8080
94 default_backend swift-vms
95 backend swift-vms
96 balance roundrobin
97 server controlador1 192.168.110.1:8080 check inter 1s
98 server controlador2 192.168.110.2:8080 check inter 1s
99 server controlador3 192.168.110.3:8080 check inter 1s
100
101 frontend vip-neutron
102 bind 192.168.110.10:9696
103 default_backend neutron-vms
104 backend neutron-vms
105 balance roundrobin
106 server controlador1 192.168.110.1:9696 check inter 1s
107 server controlador2 192.168.110.2:9696 check inter 1s
108 server controlador3 192.168.110.3:9696 check inter 1s
109
110 frontend vip-nova-vnc-novncproxy
111 bind 192.168.110.10:6080
112 default_backend nova-vnc-novncproxy-vms
113 backend nova-vnc-novncproxy-vms
114 balance roundrobin
115 timeout tunnel 1h
116 server controlador1 192.168.110.1:6080 check inter 1s
117 server controlador2 192.168.110.2:6080 check inter 1s
118 server controlador3 192.168.110.3:6080 check inter 1s
119
120 frontend nova-metadata-vms
121 bind 192.168.110.10:8775
122 default_backend nova-metadata-vms
123 backend nova-metadata-vms
124 balance roundrobin
125 server controlador1 192.168.110.1:8775 check inter 1s
```

```
126 server controlador2 192.168.110.2:8775 check inter 1s
127 server controlador3 192.168.110.3:8775 check inter 1s
128
129 frontend vip-nova-api
130 bind 192.168.110.10:8774
131 default_backend nova-api-vms
132 backend nova-api-vms
133 balance roundrobin
134 server controlador1 192.168.110.1:8774 check inter 1s
135 server controlador2 192.168.110.2:8774 check inter 1s
136 server controlador3 192.168.110.3:8774 check inter 1s
137
138 frontend vip-horizon
139 bind 192.168.110.10:80
140 timeout client 180s
141 default_backend horizon-vms
142 backend horizon-vms
143 balance roundrobin
144 timeout server 180s
145 cookie SERVERID insert indirect nocache
146 server controlador1 192.168.110.1:80 check inter 1s
147 server controlador2 192.168.110.2:80 check inter 1s
148 server controlador3 192.168.110.3:80 check inter 1s
149
150 frontend vip-heat-cfn
151 bind 192.168.110.10:8000
152 default_backend heat-cfn-vms
153 backend heat-cfn-vms
154 balance roundrobin
155 server controlador1 192.168.110.1:8000 check inter 1s
156 server controlador2 192.168.110.2:8000 check inter 1s
157 server controlador3 192.168.110.3:8000 check inter 1s
158
159 frontend vip-heat-cloudw
160 bind 192.168.110.10:8003
161 default_backend heat-cloudw-vms
162 backend heat-cloudw-vms
163 balance roundrobin
164 server controlador1 192.168.110.1:8003 check inter 1s
165 server controlador2 192.168.110.2:8003 check inter 1s
166 server controlador3 192.168.110.3:8003 check inter 1s
167
168 frontend vip-heat-srv
169 bind 192.168.110.10:8004
170 default_backend heat-srv-vms
171 backend heat-srv-vms
172 balance roundrobin
```

```
173 server controlador1 192.168.110.1:8004 check inter 1s
174 server controlador2 192.168.110.2:8004 check inter 1s
175 server controlador3 192.168.110.3:8004 check inter 1s
176
177 frontend vip-ceilometer
178 bind 192.168.110.10:8777
179 timeout client 90s
180 default_backend ceilometer-vms
181 backend ceilometer-vms
182 balance roundrobin
183 timeout server 90s
184 server controlador1 192.168.110.1:8777 check inter 1s
185 server controlador2 192.168.110.2:8777 check inter 1s
186 server controlador3 192.168.110.3:8777 check inter 1s
187
188 frontend vip-sahara
189 bind 192.168.110.10:8386
190 default_backend sahara-vms
191 backend sahara-vms
192 balance roundrobin
193 server controlador1 192.168.110.1:8386 check inter 1s
194 server controlador2 192.168.110.2:8386 check inter 1s
195 server controlador3 192.168.110.3:8386 check inter 1s
196
197 frontend vip-trove
198 bind 192.168.110.10:8779
199 default_backend trove-vms
200 backend trove-vms
201 balance roundrobin
202 server controlador1 192.168.110.1:8779 check inter 1s
203 server controlador2 192.168.110.2:8779 check inter 1s
204 server controlador3 192.168.110.3:8779 check inter 1s
205 EOF
```

4.2 Keepalived

Keepalived é um software para alta disponibilidade para infraestruturas baseadas em Linux. A alta disponibilidade é obtida utilizando o protocolo Virtual Router Redundancy Protocol (VRRP) [79]. O VRRP é um protocolo que permite a criação de um router virtual, permitindo assim que se elimine um ponto de falha comparando com a utilização de um router estático, pois este se ao final de um tempo de timeout não obtiver resposta, envia o tráfego por outro caminho. É possível visualizar esquema do funcionamento do Keepalived na figura 4.3. Uma vez que o HAProxy está instalado em três controladores é necessário escolher qual é o controlador por onde

o tráfego é encaminhado. Assim com o Keepalived conseguimos criar um cluster que garante que o tráfego passa por um controlador funcional, e não por um em falha.

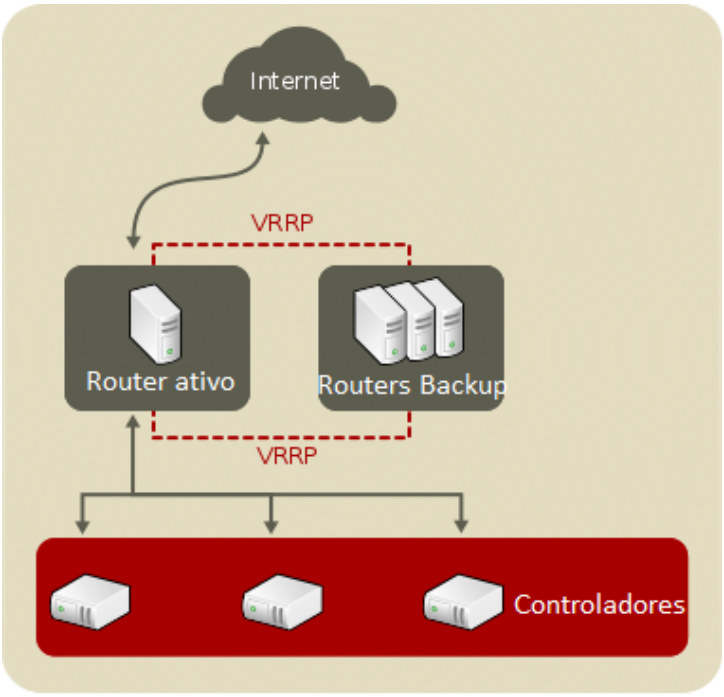


Figura 4.3: Figura representativa do Keepalived na implementação efetuada, adaptada de [10]

Os seguintes passos para a instalação do Keepalived devem ser executados em todos os controladores até que seja afirmado o contrário:

```
1 yum -y install keepalived psmisc
```

O bloco de código [3-5] apresentado de seguida corre a cada dois segundos uma verificação do estado do haproxy no controlador, e caso este esteja em falha, o keepalived desliga-o. O bloco de código [8-19] contém a configuração necessária para um VRRP. Neste bloco é definida a interface de rede utilizada, o estado (master ou backup), o número identificativo escolhido para o router (virtual router id) e a prioridade, que é um valor compreendido entre 0 e 255.

É necessário substituir o termo ESTADO em cada um dos controladores, sendo que no HA-CONTROLLER1 este deve ser MASTER, e no HA-CONTROLLER2 e HA-CONTROLLER3 este deve ser BACKUP.

```
1 cat > /etc/keepalived/keepalived.conf << EOF
2
3 vrrp_script chk_haproxy {
4     script "/usr/bin/killall -9 haproxy"
5     interval 2
6 }
7
```

```
vrrp_instance VI_PUBLIC {
    interface eth0
    state ESTADO
    virtual_router_id 50
    priority 254
    virtual_ipaddress {
        192.168.110.10 dev br-eth0
    }
    track_script {
    chk_haproxy
    }
}
vrrp_sync_group VG1
group {
    VI_PUBLIC
}
EOF
```

Após isto o Keepalived encontra-se instalado, pelo que o devemos ligar e adicionar aos serviços que arrancam com o sistema automaticamente.

```
systemctl start keepalived
systemctl enable keepalived
```

4.3 Galera

Esta secção é referente à instalação do Cluster MariaDB [80] (é um tipo de base de dados relacional, que esta organiza os dados em tabelas) juntamente com Percona XtraDB Cluster, sendo que este último é o que agrega num único pacote a biblioteca Galera [81]. A biblioteca galera serve para a sincronização paralela de base de dados pertencentes a clusters, fazendo assim com que todas as bases de dados pertencentes ao cluster se encontrem igualmente sincronizadas, levando assim à redundância de dados. Na figura 4.4 é possível de perceber o funcionamento do cluster galera na nossa implementação. Quando existe um pedido o HAProxy encaminha o tráfego para um nó de SQL, sendo que o endereço VIP pode estar mapeado em qualquer um dos três controladores. Quando são feitas alterações na base de dados, a que sofreu escrita automaticamente sincroniza as mudanças com as outras.

Os seguintes passos para a instalação devem ser executados em todos os controladores até que seja afirmado o contrário:

```
cat > /etc/yum.repos.d/mariadb.repo << EOF
[mariadb]
name = MariaDB
```

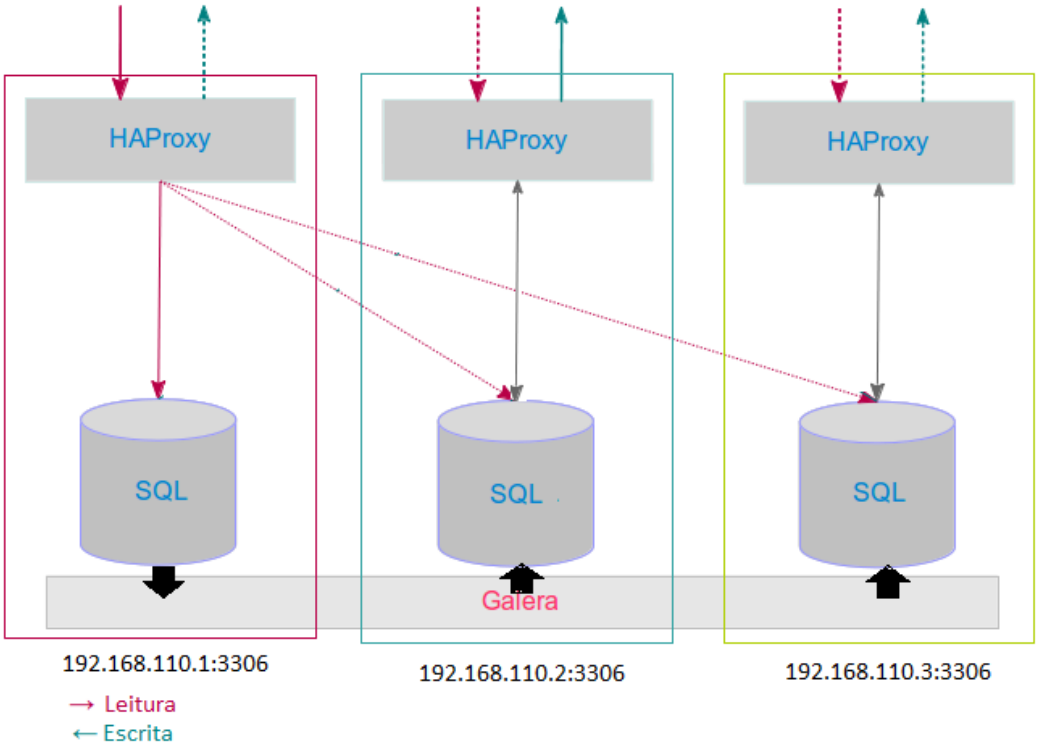


Figura 4.4: Figura representativa da replicação em um cluster Galera gerida pelo HAProxy.

```
4 baseurl = http://yum.mariadb.org/10.1/centos7-amd64
5 gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
6 gpgcheck=1
7 EOF
8
9 yum install rsync nmap lsof perl-DBI nc
10 yum install MariaDB-server MariaDB-client MariaDB-compat galera socat jemalloc
```

De seguida devemos arrancar o MySQL e criar um utilizador para o ClusterCheck. No bloco de código [8-9] seguidamente apresentado definimos um tamanho máximo para a abertura de um ficheiro, recomendado pela comunidade OpenStack.

```
1 systemctl start mysqld
2 mysql -e "CREATE USER 'clustercheck'@'localhost' IDENTIFIED BY 'redhat';"
3 systemctl stop mysqld
4
5 mkdir -p /etc/systemd/system/mariadb.service.d/
6
7 cat > /etc/systemd/system/mariadb.service.d/limits.conf << EOF
8 [Service]
```

```
LimitNOFILE=16384
EOF
```

Na seguinte fase deve ser configurado cada controlador individualmente como exemplificado de seguida. No bloco de código [1-14] apresentado de seguida é possível visualizar alguns parâmetros como: wsrep_on (ativa a replicação), wsrep_provider (define o caminho para o plugin de replicação), wsrep_cluster_address (define o tipo de backend e o endereço IP), wsrep_cluster_name (define o nome para o cluster), wsrep_sst_method (define o metodo de sincronização das bases de dados entre diferentes nós), o tipo de armazenamento por defeito (InnoDB, recomendado pela comunidade OpenStack), o tipo de bloqueio para escrita (2 - utilizado conjuntamente com ROW, sendo este faz o registos dos eventos que afetam as linhas de cada tabela individualmente) e o endereço do controlador.

No HA-CONTROLLER1:

```
cat > /etc/my.cnf.d/server.cnf << EOF
[galera]
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address='gcomm://'
wsrep_cluster_name='galera_cluster'
wsrep_node_address='192.168.110.1'
wsrep_node_name='galera1'
wsrep_sst_method=rsync
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=192.168.110.1
EOF
```

No HA-CONTROLLER2:

```
cat > /etc/my.cnf.d/server.cnf << EOF
[galera]
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address='gcomm://192.168.110.1'
wsrep_cluster_name='galera_cluster'
wsrep_node_address='192.168.110.2'
wsrep_node_name='galera2'
wsrep_sst_method=rsync
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=192.168.110.2
EOF
```

No HA-CONTROLLER3:

```
cat > /etc/my.cnf.d/server.cnf << EOF
[galera]
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address='gcomm://192.168.110.1,192.168.110.2'
wsrep_cluster_name='galera_cluster'
wsrep_node_address='192.168.110.3'
wsrep_node_name='galera3'
wsrep_sst_method=rsync
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=192.168.110.3
EOF
```

Os próximos passos devem ser executados nos três controladores.

De seguida iremos ativar o arranque automático dos serviços bem como os iremos iniciar.

```
systemctl daemon-reload
systemctl enable haproxy
systemctl start haproxy
systemctl start mariadb
```

O próximo passo consiste em criar os utilizadores para os diferentes serviços que irão ser instalados. Estes deverão ser criados apenas no HA-CONTROLLER1. É possível organizar o script abaixo por blocos, sendo que um bloco corresponde à criação da base de dados do serviço e do utilizador com as respetivas permissões para cada serviço. Por exemplo para o bloco [4-10] é criada a base de dados *keystone*, com o nome de utilizador *keystone* e password *keystonetest*. É criado um utilizador para cada controlador.

```
mysql
use mysql;

CREATE DATABASE keystone;
create user 'keystone'@'HA-CONTROLLER1' identified by 'keystonetest';
create user 'keystone'@'HA-CONTROLLER2' identified by 'keystonetest';
create user 'keystone'@'HA-CONTROLLER3' identified by 'keystonetest';
GRANT ALL ON keystone.* TO 'keystone'@'HA-CONTROLLER2' IDENTIFIED BY 'keystonetest';
GRANT ALL ON keystone.* TO 'keystone'@'HA-CONTROLLER3' IDENTIFIED BY 'keystonetest';
GRANT ALL ON keystone.* TO 'keystone'@'HA-CONTROLLER1' IDENTIFIED BY 'keystonetest';

```

```
12 CREATE DATABASE glance;
13 create user 'glance'@'HA-CONTROLLER1' identified by 'glancetest';
14 create user 'glance'@'HA-CONTROLLER2' identified by 'glancetest';
15 create user 'glance'@'HA-CONTROLLER3' identified by 'glancetest';
16 GRANT ALL ON glance.* TO 'glance'@'HA-CONTROLLER1' IDENTIFIED BY 'glancetest';
17 GRANT ALL ON glance.* TO 'glance'@'HA-CONTROLLER2' IDENTIFIED BY 'glancetest';
18 GRANT ALL ON glance.* TO 'glance'@'HA-CONTROLLER3' IDENTIFIED BY 'glancetest';
19
20 CREATE DATABASE cinder;
21 create user 'cinder'@'HA-CONTROLLER1' identified by 'cindertest';
22 create user 'cinder'@'HA-CONTROLLER2' identified by 'cindertest';
23 create user 'cinder'@'HA-CONTROLLER3' identified by 'cindertest';
24 GRANT ALL ON cinder.* TO 'cinder'@'HA-CONTROLLER1' IDENTIFIED BY 'cindertest';
25 GRANT ALL ON cinder.* TO 'cinder'@'HA-CONTROLLER2' IDENTIFIED BY 'cindertest';
26 GRANT ALL ON cinder.* TO 'cinder'@'HA-CONTROLLER3' IDENTIFIED BY 'cindertest';
27
28 CREATE DATABASE neutron;
29 create user 'neutron'@'HA-CONTROLLER1' identified by 'neutrontest';
30 create user 'neutron'@'HA-CONTROLLER2' identified by 'neutrontest';
31 create user 'neutron'@'HA-CONTROLLER3' identified by 'neutrontest';
32 GRANT ALL ON neutron.* TO 'neutron'@'HA-CONTROLLER1' IDENTIFIED BY 'neutrontest';
33 GRANT ALL ON neutron.* TO 'neutron'@'HA-CONTROLLER2' IDENTIFIED BY 'neutrontest';
34 GRANT ALL ON neutron.* TO 'neutron'@'HA-CONTROLLER3' IDENTIFIED BY 'neutrontest';
35
36 CREATE DATABASE nova;
37 create user 'nova'@'HA-CONTROLLER1' identified by 'novatest';
38 create user 'nova'@'HA-CONTROLLER2' identified by 'novatest';
39 create user 'nova'@'HA-CONTROLLER3' identified by 'novatest';
40 GRANT ALL ON nova.* TO 'nova'@'HA-CONTROLLER2' IDENTIFIED BY 'novatest';
41 GRANT ALL ON nova.* TO 'nova'@'HA-CONTROLLER3' IDENTIFIED BY 'novatest';
42 GRANT ALL ON nova.* TO 'nova'@'HA-CONTROLLER1' IDENTIFIED BY 'novatest';
43
44 CREATE DATABASE nova_api;
45 create user 'nova_api'@'HA-CONTROLLER1' identified by 'novatest';
46 create user 'nova_api'@'HA-CONTROLLER2' identified by 'novatest';
47 create user 'nova_api'@'HA-CONTROLLER3' identified by 'novatest';
48 GRANT ALL ON nova_api.* TO 'nova_api'@'HA-CONTROLLER2' IDENTIFIED BY 'novatest';
49 GRANT ALL ON nova_api.* TO 'nova_api'@'HA-CONTROLLER3' IDENTIFIED BY 'novatest';
50 GRANT ALL ON nova_api.* TO 'nova_api'@'HA-CONTROLLER1' IDENTIFIED BY 'novatest';
51
52 CREATE DATABASE heat;
53 create user 'heat'@'HA-CONTROLLER1' identified by 'heattest';
54 create user 'heat'@'HA-CONTROLLER2' identified by 'heattest';
55 create user 'heat'@'HA-CONTROLLER3' identified by 'heattest';
56 GRANT ALL ON heat.* TO 'heat'@'HA-CONTROLLER2' IDENTIFIED BY 'heattest';
57 GRANT ALL ON heat.* TO 'heat'@'HA-CONTROLLER3' IDENTIFIED BY 'heattest';
58 GRANT ALL ON heat.* TO 'heat'@'HA-CONTROLLER1' IDENTIFIED BY 'heattest';
```

```
59
60 CREATE DATABASE sahara;
61 create user 'sahara'@'HA-CONTROLLER1' identified by 'saharatest';
62 create user 'sahara'@'HA-CONTROLLER2' identified by 'saharatest';
63 create user 'sahara'@'HA-CONTROLLER3' identified by 'saharatest';
64 GRANT ALL ON sahara.* TO 'sahara'@'HA-CONTROLLER2' IDENTIFIED BY 'saharatest';
65 GRANT ALL ON sahara.* TO 'sahara'@'HA-CONTROLLER3' IDENTIFIED BY 'saharatest';
66 GRANT ALL ON sahara.* TO 'sahara'@'HA-CONTROLLER1' IDENTIFIED BY 'saharatest';
67
68 CREATE DATABASE trove;
69 create user 'trove'@'HA-CONTROLLER1' identified by 'trovetest';
70 create user 'trove'@'HA-CONTROLLER2' identified by 'trovetest';
71 create user 'trove'@'HA-CONTROLLER3' identified by 'trovetest';
72 GRANT ALL ON trove.* TO 'trove'@'HA-CONTROLLER2' IDENTIFIED BY 'trovetest';
73 GRANT ALL ON trove.* TO 'trove'@'HA-CONTROLLER3' IDENTIFIED BY 'trovetest';
74 GRANT ALL ON trove.* TO 'trove'@'HA-CONTROLLER1' IDENTIFIED BY 'trovetest';
75
76 FLUSH PRIVILEGES;
77
78 quit
79 mysqladmin flush-hosts
```

O próximo e último passo na instalação e configuração do Cluster consiste na configuração de um *script* em todos os controladores para verificar se o cluster está devidamente sincronizado. Para este script funcionar corretamente é necessário que exista um utilizador com acesso às tabelas da base de dados (bloco [9-11]) e que este conheça, para que possa comparar se a sincronização está corretamente efetuada ou não.

```
1 yum install git
2 git clone https://github.com/olafz/percona-clustercheck
3
4 cd percona-clustercheck
5 mv clustercheck /bin/usr
6 cd /bin/usr
7 chmod 755 clustercheck
8
9 mysql
10 GRANT PROCESS ON *.* TO 'clustercheckuser'@'localhost' IDENTIFIED BY
       'clustercheckpassword!' ;
11 quit
12
13 cat > /usr/bin/clustercheck_best << EOF
14 !/bin/bash
15 /usr/bin/clustercheck clustercheckuser clustercheckpassword!
16 EOF
17
```

```
18 chmod 755 clustercheck_best
```

No bloco de código [4-15] de seguida apresentamos as configurações para o xinetd, que é um serviço para controlo de acessos a serviços disponíveis na rede. Este foi utilizado para fazer a verificação do cluster de base de dados entre os diferentes controladores. Na configuração do serviço foi criado um grupo *mysqlchk* com algumas opções como a porta que este utiliza, o tipo de socket utilizada, o servidor (especifica qual o binário executavel a dar inicio), registo de log em caso de falha, o wait define se este é multi-threaded (no) ou single-threaded (yes) e o disable que especifica se o serviço está inativo (yes) ou ativo (no).

```
1 yum install xinetd
2 cat > /etc/xinetd.d/mysqlchk << EOF
3
4 service mysqlchk
5 {
6 disable = no
7 socket_type = stream
8 port = 9200
9 wait = no
10 user = nobody
11 server = /usr/bin/clustercheck_best
12 log_on_failure += USERID
13 only_from = 0.0.0.0/0
14 per_source = UNLIMITED
15 }
16 EOF
```

O bloco de código abaixo [1-3] serve para darmos permissão ao serviço mysqlchk do xinetd de comunicar por TCP na porta 9200. Por outro lado comentamos o wap-wsp uma vez que este não é utilizado e estaria a utilizar a mesma porta que o mysqlchk, o que resultaria num conflito. Adicionar a primeira linha e comentar as duas últimas no ficheiro /etc/services.

```
1 mysqlchk 9200/tcp
2 wap-wsp 9200/tcp
3 wap-wsp 9200/udp
4 service xinetd start
```

4.4 RabbitMQ

O RabbitMQ é um MoM - Message Oriented Middleware (serve para troca de mensagens entre sistemas distribuídos) que implementa o protocolo Advanced Message Queuing Protocol (AMQP) [82] que serve para orientação de mensagens, filas, e encaminhamento de mensagens.

Assim ao utilizarmos o RabbitMQ conseguimos que seja possível a troca de mensagens entre os controladores e os nós de computação, uma vez o RabbitMQ faz a tradução das mensagens para o formato que cada serviço está à espera. Ao ser utilizado um cluster RabbitMQ as mensagens são espelhadas por todos os nós. As filas comportam-se por padrão como se estivessem em apenas em um único nó, mas estão espelhadas em todos os nós, tornando-se tolerante a falhas de forma transparente e sem perda de mensagens caso seja mantida pelo menos uma das filas [83]. Um exemplo troca de mensagens no OpenStack entre o controlador e um nó de computação pode ser visto na figura 4.5.

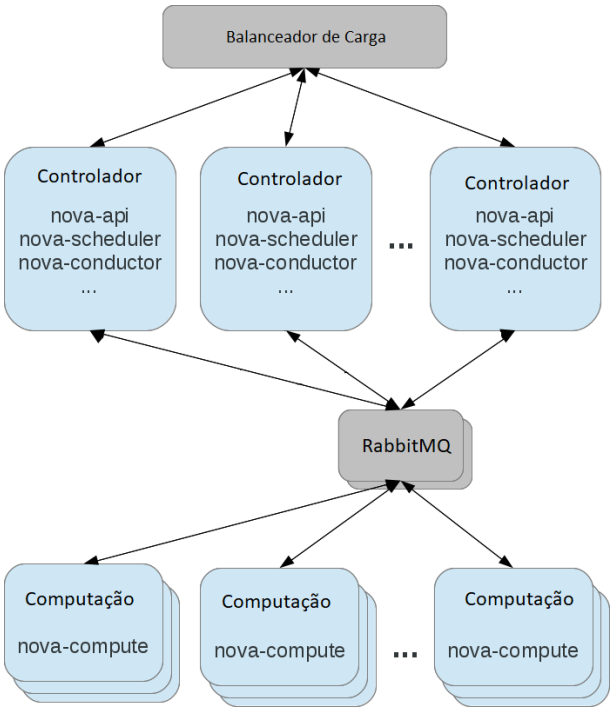


Figura 4.5: Esquema representativo da troca de mensagens com o MoM RabbitMQ entre o Nova em um controlador e em um nó de computação.

Os seguintes passos para a instalação do RabbitMQ devem ser executados em todos os controladores até que seja afirmado o contrário:

```
1 yum -y install rabbitmq-server
```

É necessário criar o *erlang cookie* e distribuí-lo pelos restantes controladores. O *erlang cookie* é uma série de caracteres alfanuméricos e atua como chave e serve para que dois ou mais nós consigam comunicar entre si. Como tal todos os nós devem possuir o mesmo *erlang cookie*. Este é gerado automaticamente uma única vez quando o serviço é iniciado pela primeira vez. Como tal iremos criá-lo no HA-CONTROLLER1 e distribuí-lo aos restantes para que todos consigam comunicar entre si:

```
1 cat > /etc/rabbitmq/rabbitmq-env.conf << EOF
2 NODE_IP_ADDRESS=192.168.110.1
```

```
3 EOF
4
5 systemctl start rabbitmq-server
6 systemctl stop rabbitmq-server
7
8 scp -p /var/lib/rabbitmq/.erlang.cookie HA-CONTROLLER2:/var/lib/rabbitmq
9 scp -p /var/lib/rabbitmq/.erlang.cookie HA-CONTROLLER3:/var/lib/rabbitmq
```

Os seguintes passos para a instalação do RabbitMQ devem ser executados nos controladores 2 e 3, e têm como objetivo dar as permissões certas ao *erlang cookie*:

```
1 chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie
2
3 cat > /etc/rabbitmq/rabbitmq-env.conf << EOF
4 NODE_IP_ADDRESS=ADDRESS
5 EOF
```

O próximo passo é editar o ficheiro de configuração principal do RabbitMQ, acrescentando ao mesmo os nós constituintes do cluster. Para tal devemos editar o ficheiro de configuração em todos os nós (bloco [2-16]). Neste bloco é definido o utilizador e a palavra pass por defeito, reutilização do endereço para que este fique em alta disponibilidade, o ativar do serviço em background, É ainda necessário adicionar os parâmetros TCP do keepalive (bloco [20-24], sendo que iremos definir que o keepalive espera 5 segundos até ter atividade na socket antes de enviar uma prova e reenvia-a a cada 1 segundo. No bloco [12-14] definimos a porta de comunicação do RabbitMQ com os serviços.

```
1 cat > /etc/rabbitmq/rabbitmq.config << EOF
2 [
3 {rabbit, [
4 {cluster_nodes, {['rabbit@HA-CONTROLLER1', 'rabbit@HA-CONTROLLER2',
      'rabbit@HA-CONTROLLER3'], disc}},
5 {default_user, <<"guest">>},
6 {default_pass, <<"guest">>},
7 {tcp_listen_options, [
8     {reuseaddr, true},
9     {exit_on_close, false},
10    {keepalive, true}]}
11 ]},
12 {kernel, [
13    {inet_dist_listen_max, 44001},
14    {inet_dist_listen_min, 44001}
15 ]}
16 ].
17 EOF
18
```

```
19
20 cat > /etc/sysctl.d/tcpka.conf << EOF
21 net.ipv4.tcp_keepalive_intvl = 1
22 net.ipv4.tcp_keepalive_probes = 5
23 net.ipv4.tcp_keepalive_time = 5
24 EOF
25
26 systemctl enable rabbitmq-server
27 systemctl start rabbitmq-server
```

Para finalizar a instalação do RabbitMQ iremos verificar o estado do cluster, isto é, se ficou bem criado, e no HA-CONTROLLER1 iremos definir o modo de alta disponibilidade para todas as filas de mensagens, passando assim estas a serem espelhadas pelos outros nós.

```
1 rabbitmqctl cluster_status
2 rabbitmqctl set_policy HA '^(?!amq\.).*' '{"ha-mode": "all"}'
```

4.5 MemCached

O MemCached é um sistema de cache de objetos de memória destinado à aceleração de aplicações web de forma a aliviar a carga sobre as bases de dados. Ao ser utilizado no OpenStack podemos reduzir o número de acessos às bases de dados, como por exemplo para obter tokens de autenticação, conseguindo assim uma melhor performance devido à ausência de necessidade de consulta à base de dados[84].

A instalação do MemCached é deveras simples e não exige nenhuma configuração em especial. Apenas é necessário correr os seguintes comandos em todos os controladores:

```
1 yum install -y memcached
2 systemctl start memcached
3 systemctl enable memcached
```

Os valores utilizados são os configurados por defeito, pelo que não foi necessária nenhuma alteração a estes para o correcto funcionamento do OpenStack com o MemCached.

4.6 Redis

O Redis é utilizado no OpenStack pelo Ceilometer devido a ser uma estrutura que guarda tuplos chave/valor com cache, desta forma os dados anteriormente carregados ficam disponíveis em cache o que dispensa que seja feita uma nova chamada enquanto estes estiverem guardados e

forem consistentes. Este é usado em alta disponibilidade pois quando se escreve para um nó master este replica a informação para os slaves, sendo que caso o nó master falhe, o Sentinel altera o papel de um slave para passar a ser o master. Os seguintes passos para a instalação do Redis devem ser executados em todos os controladores.

O seguinte bloco de código tem como objetivo substituir todas as ocorrências de *bind 127.0.0.1* por *bind 127.0.0.1 ADDRESS*, para que o redis saiba qual é o seu endereço respectivo.

```
1 yum install -y redis
2 sed --in-place 's/bind 127.0.0.1/bind 127.0.0.1 ADDRESS/' /etc/redis.conf
```

O seguinte bloco de código deverá ser executado no HA-CONTROLLER 2,3 e serve para que seja definido qual é o endereço IP do redis master, neste caso é o HA-CONTROLLER1.

```
1 echo slaveof 192.168.110.1 6379 >> /etc/redis.conf
```

O próximo passo consiste na configuração do *Sentinel*, sendo que o objetivo deste é deixar o nó master inativo caso este falhe. Esta configuração deverá ser aplicada aos três controladores. Da análise do bloco de código [1-9] de seguida podemos verificar que é definido aqui que o nó master (monitor) é o HA-CONTROLLER1 (192.168.110.1), que a porta utilizada é a 6379 e que possui 2 slaves. 30000 milisegundos é o tempo para que se espera para concluir que um Sentinel está em falha, sendo que caso este não recupere ao final de 180000 milisegundos um slave fica automaticamente promovido a master. A variavel *parallel-syncs* define o número de slaves que têm que passar a comunicar com o novo master, caso o inicial fique em falha.

```
1 cat > /etc/redis-sentinel.conf << EOF
2 sentinel monitor mymaster 192.168.110.1 6379 2
3 sentinel down-after-milliseconds mymaster 30000
4 sentinel failover-timeout mymaster 180000
5 sentinel parallel-syncs mymaster 1
6 min-slaves-to-write 1
7 logfile /var/log/redis/sentinel.log
8 EOF
```

Para finalizar apenas nos falta ativar o serviço e garantir que este é automaticamente ativo no arranque do sistema:

```
1 systemctl enable redis
2 systemctl start redis
3 systemctl enable redis-sentinel
4 systemctl start redis-sentinel
```

4.7 MongoDB

O MongoDB é um software destinado a bases de dados do tipo NoSQL [85]. Como no OpenStack os serviços de telemetria e monitorização utilizam dados não relacionais, é necessário o uso de um software para bases de dados deste tipo. Devido a isto e sendo este o recomendado pelo OpenStack, optou-se pela sua utilização em alta disponibilidade. A alta disponibilidade no MongoDB é obtida quando existe um nó master e outros nós slaves, sendo que como a informação é replicada em todos os nós, quando é recebido um pedido o nó master delega um nó slave para receber o pedido, esquematizado na figura 4.6. Caso o nó slave que estava responsável de tratar do pedido falhe, o nó master encarrega outro nó slave de responder ao pedido.

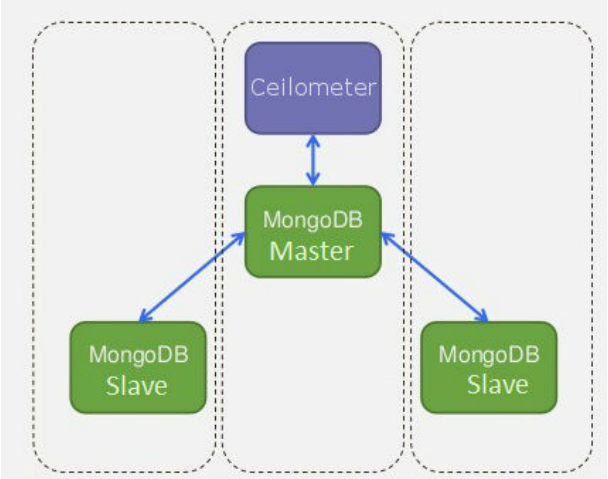


Figura 4.6: Esquema representativo da delegação da base de dados no MongoDB, adaptado de: [11].

Os seguintes passos para a instalação do MongoDB devem ser executados em todos os controladores. Na linha 1 do seguinte bloco de código procedemos à instalação dos serviços do mongodb. Na linha 2 indicamos o conjunto de réplicas com o nome *ceilometer*. Nas linhas 4 e 5 ativamos o serviço e garantimos que este é automaticamente iniciado com o sistema no arranque. Na linha 6 substituímos todas as ocorrências de bind_ip.* por bind_ip = 0.0.0.0 para que sejam permitidas conexões de qualquer endereço IP local do controlador.

```
1 yum install -y mongodb mongodb-server
2 echo "replSet = ceilometer" >> /etc/mongod.conf
3
4 systemctl start mongod
5 systemctl enable mongod
6 sed -i -e 's#bind_ip.*#bind_ip = 0.0.0.0#g' /etc/mongod.conf
```

De seguida, no HA-CONTROLLER1, vamos iniciar o cluster (linha 2) e adicionar ao cluster o HA-CONTROLLER2 e HA-CONTROLLER3 (linha 4 e 5). A linha 3 faz com que se espere dez

segundos antes de adicionarmos os nós ao cluster, para garantir que este cluster já foi iniciado quando se for adicionar os nós.

```
1 mongo
2 rs.initiate()
3 sleep(10000)
4 rs.add("HA-CONTROLLER2");
5 rs.add("HA-CONTROLLER3");
```

O próximo e último passo na configuração do MongoDB consiste em verificar se os nós do cluster foram bem adicionados. Para tal podemos correr os seguintes comandos em qualquer controlador. Quando o *stateStr* aparecer como *PRIMARY* ou *SECONDARY* podemos sair do MongoDB

```
1 rs.status()
2 quit()
```

4.8 KeyStone

Como foi anteriormente descrito no capítulo 3, na secção 3.2.6, o KeyStone é o serviço responsável pela autenticação, gestão de identidades e projetos no OpenStack. Na imagem 4.7 é visível um exemplo do papel do KeyStone no processo de troca de mensagens na criação de uma máquina virtual uma vez que os serviços antes de procederem a alguma operação comunicam com o KeyStone para fazer uma validação da autenticação e da autorização desse pedido.

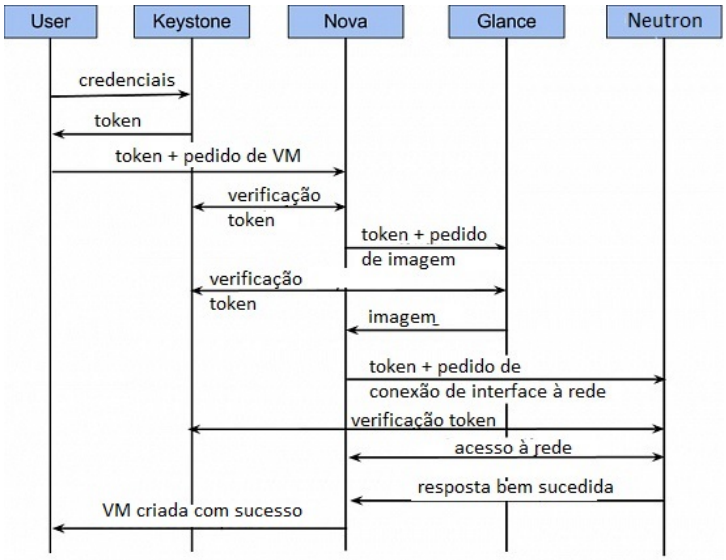


Figura 4.7: Imagem representativa do processo de criação de uma VM [12].

Da figura anterior é possível analisar a atuação do KeyStone: 1) o utilizador envia as suas

credenciais ao KeyStone e este devolve-lhe um *token* de autenticação; 2) o utilizador envia o *token* que lhe foi devolvido juntamente com o pedido de criação de uma nova instância, sendo que o KeyStone o dirige ao Nova; 3) o Nova e o KeyStone fazem uma troca de *tokens* de autenticação, sendo que este ultimo verifica se o *token* do Nova está autorizado; 4) uma vez autorizado, uma Nova envia o *token* ao Glance e efetua um pedido de imagem para a criação da instância para o utilizador; 5) o Glance comunica com o KeyStone para verificar se o *token* do Nova está correto e caso esteja, envia a imagem ao Nova; 6) após recebida a imagem do Glance, o Nova envia o seu *token* ao Neutron e faz-lhe um pedido de conexão à interface de rede; 7) o Neutron comunica com o KeyStone e valida o *token* do Nova, caso este esteja correto permite o acesso à rede por parte do Nova; 8) o Nova finaliza a criação da VM e comunica ao utilizador.

Os seguintes passos para a instalação do KeyStone devem ser executados em todos os controladores.

```
1 yum install -y openstack-keystone openstack-utils openstack-selinux httpd mod_wsgi
      python-openstackclient
```

De seguida, no HA-CONTROLLER1, vamos gerar o *token* de serviço e distribuí-lo pelos HA-CONTROLLER2 e HA-CONTROLLER3. Para tal geramos uma chave recorrendo ao openssl, aleatoria hexadecimal, de comprimento 10 unidades e procedemos ao envio para os outros controladores usando *secure copy* (tunel seguro ssh para transferência de dados).

```
1 export SERVICE_TOKEN=$(openssl rand -hex 10)
2 echo $SERVICE_TOKEN > /root/keystone_service_token
3 scp /root/keystone_service_token root@HA-CONTROLLER2:/root
4 scp /root/keystone_service_token root@HA-CONTROLLER3:/root
```

O próximo passo consiste em configurar o servidor de *apache* e o keystone. Esta configuração deve ser feita em todos os controladores. Na linha 1 procedemos à copia do ficheiro wsgi (Web Server Gateway Interface) do KeyStone para a pasta de configuração do apache, uma vez que este gateway é o que faz a ponte entre o KeyStone e o serviço. No bloco de código [2-6] procedemos à substituição das ocorrências necessárias para que o ficheiro de configuração contenha o endereço IP do controlador e as portas para o qual permitirá comunicação.

```
1 cp /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/
2 sed -i -e 's/apache2/httpd/g' /etc/httpd/conf.d/wsgi-keystone.conf
3 sed -i -e 's/VirtualHost \*/VirtualHost ADDRESS/g' /etc/httpd/conf.d/wsgi-keystone.conf
4 sed -i -e 's/Listen 5000/Listen ADDRESS:5000/g' /etc/httpd/conf.d/wsgi-keystone.conf
5 sed -i -e 's/Listen 35357/Listen ADDRESS:35357/g' /etc/httpd/conf.d/wsgi-keystone.conf
6 sed -i -e 's/^Listen.*/Listen ADDRESS:80/g' /etc/httpd/conf/httpd.conf
```

No seguinte bloco de código [1-11] procedemos à configuração de alguns parâmetros necessários no ficheiro de configuração como: *token* do KeyStone, controladores RabbitMQ para troca de mensagens e ativação de alta disponibilidade, caminho para a conexão à base de dados bem como

adição de uma variável para as tentativas máximas de conexão sejam ilimitadas, e endereço de público e de administração, que no nosso caso será o mesmo.

```
1 export SERVICE_TOKEN=$(cat /root/keystone_service_token)
2 openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token
      $SERVICE_TOKEN
3 openstack-config --set /etc/keystone/keystone.conf DEFAULT rabbit_hosts
      HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
4 openstack-config --set /etc/keystone/keystone.conf DEFAULT rabbit_ha_queues true
5 openstack-config --set /etc/keystone/keystone.conf eventlet_server admin_endpoint
      'http://192.168.110.10:%(admin_port)s/'
6 openstack-config --set /etc/keystone/keystone.conf eventlet_server public_endpoint
      'http://192.168.110.10:%(public_port)s/'
7 openstack-config --set /etc/keystone/keystone.conf database connection
      mysql://keystone:keystonetest@192.168.110.10/keystone
8 openstack-config --set /etc/keystone/keystone.conf database max_retries -1
9 openstack-config --set /etc/keystone/keystone.conf DEFAULT public_bind_host ADDRESS
10 openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_bind_host ADDRESS
11 openstack-config --set /etc/keystone/keystone.conf token driver
      keystone.token.persistence.backends.sql.Token
```

O próximo passo consiste em distribuir pelos controladores os ficheiros criados pelo plugin PKI (Public Key Infrastructure). Os seguintes passos devem ser realizados no HA-CONTROLLER1. Na linha 1 são gerados os ficheiros e na linha 2 é feita uma mudança recursiva do dono dos ficheiros para o utilizador KeyStone com o grupo KeyStone, assim garantimos que este é o único dono dos ficheiros. A linha 3 é feita uma sincronização das bases de dados com as alterações executadas no KeyStone sendo que por fim são enviados os ficheiros aos outros controladores.

```
1 keystone-manage pki_setup --keystone-user keystone --keystone-group keystone
2 chown -R keystone:keystone /var/log/keystone /etc/keystone/ssl/
3 su keystone -s /bin/sh -c "keystone-manage db_sync"
4 cd /etc/keystone/ssl
5 tar cvfz /tmp/keystone_ssl.tgz *
6 scp /tmp/keystone_ssl.tgz HA-CONTROLLER:/tmp
7 scp /tmp/keystone_ssl.tgz HA-CONTROLLER:/tmp
```

De seguida, no HA-CONTROLLER2 e no HA-CONTROLLER3 iremos atualizar os ficheiros para os que criamos no HA-CONTROLLER1 anteriormente, uma vez que estes necessitam de manter a coerência para que a autenticação seja igual em todos os controladores. Como tal iremos extrair os ficheiros para o diretório correto (linhas 1,2 3 4), mudar novamente o dono dos ficheiros para o KeyStone e proceder ao restauro do contexto de segurança dos ficheiros (linha 5) para que estes herdem os atributos do utilizador KeyStone. Por fim criamos o ficheiro de log e colocamos o utilizador KeyStone com o seu dono, para que sejam efetuados os registos das ocorrências do KeyStone.

```
mkdir -p /etc/keystone/ssl
cd /etc/keystone/ssl
tar xvfz /tmp/keystone_ssl.tgz
chown -R keystone:keystone /var/log/keystone /etc/keystone/ssl/
restorecon -Rv /etc/keystone/ssl
touch /var/log/keystone/keystone.log
chown keystone:keystone /var/log/keystone/keystone.log
```

É necessário adicionar uma tarefa de limpeza de *tokens* expirados. Esta tarefa é necessária para se garantir que apenas são utilizados *tokens* válidos e não expirados, aumentando assim a segurança. Para tal basta adicionar uma tarefa aos cron jobs. Após isso também já podemos arrancar o serviço do apache. Estas duas tarefas devem ser realizadas em todos os controladores.

```
1 echo "1 * * * * keystone keystone-manage token_flush
      >>/var/log/keystone/keystone-tokenflush.log 2>&1" >> /etc/crontab
2 systemctl start httpd
3 systemctl enable httpd
```

O próximo passo na configuração do KeyStone consiste em criar os *endpoints*, os serviços e os utilizadores para todas as API dos serviços do OpenStack. Para o bloco de código seguinte, na linha 1 verificamos a exportação do *token* de autenticação para as variáveis de ambiente; nas linhas [3-9] criamos o serviço de identidade do OpenStack, e indicamos quais são os URLs a serem acedidos para a autenticação, bem como os utilizadores, o nome do projeto e os papeis, quer para gestão, quer para os serviços comunicarem. Os restantes blocos de código seguem todos a mesma estrutura, sendo um bloco por cada serviço. Por exemplo o bloco [11-16] é referente ao Glance: para este serviço é criado o utilizador Glance, com a password glancetest bem como é lhe atribuído um papel de serviço e é adicionado o URL de comunicação público/interno/administração, que no nosso caso é o mesmo.

Estas configurações devem ser feitas no HA-CONTROLLER1.

```
1  export OS_TOKEN=$(cat /root/keystone_service_token)
2
3  openstack service create --name=keystone --description="Keystone Identity Service" identity
4  openstack endpoint create --publicurl 'http://192.168.110.10:5000/v2.0' --adminurl
       'http://192.168.110.10:35357/v2.0' --internalurl 'http://192.168.110.10:5000/v2.0'
       --region regionOne keystone
5  openstack user create --password keystonetest admin
6  openstack role create admin
7  openstack project create admin
8  openstack role add --project admin --user admin admin
9  openstack project create --description "Services Tenant" services
10
11 -- GLANCE
12 openstack user create --password glancetest glance
```

```
13 openstack role add --project services --user glance admin
14 openstack service create --name=glance --description="Glance Image Service" image
15 openstack endpoint create --publicurl 'http://192.168.110.10:9292' --adminurl
       'http://192.168.110.10:9292' --internalurl 'http://192.168.110.10:9292' --region regionOne
       glance
16
17 -- CINDER
18 openstack user create --password cindertest cinder
19 openstack role add --project services --user cinder admin
20 openstack service create --name=cinder --description="Cinder Volume Service" volume
21 openstack endpoint create --publicurl "http://192.168.110.10:8776/v1/\$(tenant_id)s"
       --adminurl "http://192.168.110.10:8776/v1/\$(tenant_id)s" --internalurl
       "http://192.168.110.10:8776/v1/\$(tenant_id)s" --region regionOne cinder
22 openstack service create --name=cinderv2 --description="OpenStack Block Storage" volumev2
23 openstack endpoint create --publicurl "http://192.168.110.10:8776/v2/\$(tenant_id)s"
       --adminurl "http://192.168.110.10:8776/v2/\$(tenant_id)s" --internalurl
       "http://192.168.110.10:8776/v2/\$(tenant_id)s" --region regionOne cinderv2
24
25 -- SWIFT
26 openstack user create --password swifttest swift
27 openstack role add --project services --user swift admin
28 openstack service create --name=swift --description="Swift Storage Service" object-store
29 openstack endpoint create --publicurl "http://192.168.110.10:8080/v1/AUTH_\$(tenant_id)s"
       --adminurl "http://192.168.110.10:8080/v1" --internalurl
       "http://192.168.110.10:8080/v1/AUTH_\$(tenant_id)s" --region regionOne swift
30
31 -- NEUTRON
32 openstack user create --password neutrontest neutron
33 openstack role add --project services --user neutron admin
34 openstack service create --name=neutron --description="OpenStack Networking Service"
       network
35 openstack endpoint create --publicurl "http://192.168.110.10:9696" --adminurl
       "http://192.168.110.10:9696" --internalurl "http://192.168.110.10:9696" --region
       regionOne neutron
36
37 -- NOVA
38 openstack user create --password novatest compute
39 openstack role add --project services --user compute admin
40 openstack service create --name=compute --description="OpenStack Compute Service"
       compute
41 openstack endpoint create --publicurl "http://192.168.110.10:8774/v2/\$(tenant_id)s"
       --adminurl "http://192.168.110.10:8774/v2/\$(tenant_id)s" --internalurl
       "http://192.168.110.10:8774/v2/\$(tenant_id)s" --region regionOne compute
42
43 -- HEAT
44 openstack user create --password heattest heat
45 openstack role add --project services --user heat admin
```

```
46 openstack service create --name=heat --description="Heat Orchestration Service" orchestration
47 openstack endpoint create --publicurl "http://192.168.110.10:8004/v1/%(tenant_id)s"
       --adminurl "http://192.168.110.10:8004/v1/%(tenant_id)s" --internalurl
       "http://192.168.110.10:8004/v1/%(tenant_id)s" --region regionOne heat
48 openstack service create --name=heat-cfn --description="Heat CloudFormation Service"
       cloudformation
49 openstack endpoint create --publicurl "http://192.168.110.10:8000/v1" --adminurl
       "http://192.168.110.10:8000/v1" --internalurl "http://192.168.110.10:8000/v1" --region
       regionOne heat-cfn
50
51 -- CEILOMETER
52 openstack user create --password ceilometertest ceilometer
53 openstack role add --project services --user ceilometer admin
54 openstack role create ResellerAdmin
55 openstack role add --project services --user ceilometer ResellerAdmin
56 openstack service create --name=ceilometer --description="OpenStack Telemetry Service"
       metering
57 openstack endpoint create --publicurl "http://192.168.110.10:8777" --adminurl
       "http://192.168.110.10:8777" --internalurl "http://192.168.110.10:8777" --region
       regionOne ceilometer
58
59 -- SAHARA
60 openstack user create --password saharatest sahara
61 openstack role add --project services --user sahara admin
62 openstack service create --name=sahara --description="Sahara Data Processing"
       data-processing
63 openstack endpoint create --publicurl "http://192.168.110.10:8386/v1.1/%(tenant_id)s"
       --adminurl "http://192.168.110.10:8386/v1.1/%(tenant_id)s" --internalurl
       "http://192.168.110.10:8386/v1.1/%(tenant_id)s" --region regionOne sahara
64
65 -- TROVE
66 openstack user create --password trovetest trove
67 openstack role add --project services --user trove admin
68 openstack service create --name=trove --description="OpenStack Database Service" database
69 openstack endpoint create --publicurl "http://192.168.110.10:8779/v1.0/%(tenant_id)s"
       --adminurl "http://192.168.110.10:8779/v1.0/%(tenant_id)s" --internalurl
       "http://192.168.110.10:8779/v1.0/%(tenant_id)s" --region regionOne trove
```

O último passo da configuração do KeyStone é criar o ficheiro keystonerc para simplificar a configuração dos próximos serviços. Este ficheiro contém as variáveis de ambiente necessárias para a autenticação no KeyStone como administrador, pelo que para nos autenticarmos no KeyStone a partir dos controladores passa apenas a ser necessário carregar este ficheiro para que as variáveis sejam lidas. Esta configuração deverá ser executada em todos os controladores.

```
1 cat > /root/keystonerc_admin << EOF
2 export OS_USERNAME=admin
```

```
export OS_TENANT_NAME=admin
export OS_PROJECT_NAME=admin
export OS_REGION_NAME=regionOne
export OS_PASSWORD=keystonetest
export OS_AUTH_URL=http://192.168.110.10:35357/v2.0/
export PS1='[\u@\h \W(keystone_admin)]\$ '
EOF
```

4.9 Glance

Como foi anteriormente descrito no capítulo 3, na secção 3.2.3, o Glance é um serviço que serve para o registo e distribuição de imagens no OpenStack, imagens estas com os sistemas operativos a serem utilizados pelas instâncias de máquinas virtuais. Quando um utilizador faz um pedido de uma imagem, o KeyStone valida a autorização e encaminha o pedido ao Nova, que por sua vez verifica a autenticação e trata o pedido. Este procedimento é visível na figura anteriormente apresentada em 4.7. Quando um utilizador pretende carregar ou apagar uma imagem o processo é o mesmo.

O Glance é constituído por 4 componentes que necessitam de ser configurados:

- Glance-api - trata das chamadas à API como pedidos de descoberta, carregamento ou apagar de imagens;
- Glance-registry - Armazena, processa e recupera metadados para as imagens. Estes incluem dados como tamanho e tipo;
- Base de dados - Armazena os metadados;
- Storage repository - Armazena as Imagens;

Os seguintes passos para a instalação do Glance devem ser executados em todos os controladores.

```
yum install -y openstack-glance openstack-utils openstack-selinux nfs-utils
```

De seguida necessitamos de configurar o glance-api e o glance-registry. O bloco de código [1-13] é referente à configuração do glance-api: neste é lhe dito qual é o caminho para a conexão da base de dados, o número de tentativas de conexão que se podem fazer à base de dados, os dados referentes à autenticação no KeyStone como URL, plugin de autenticação (password), username, password e nome do projeto (services). É também configurado o tipo de driver de notificações (messaging), o endereço IP do controlador, o endereço IP VIP e o hostname dos controladores que tem os *hosts* do RabbitMQ. O bloco de código [14-22] é referente à configuração

do glance-registry, sendo que os parâmetros a configurar são os mesmos que para o bloco de código anterior, à exceção de que não se indica o hostname dos controladores que tem os *hosts* do RabbitMQ.

```
1 openstack-config --set /etc/glance/glance-api.conf database connection
      mysql://glance:glancetest@192.168.110.10/glance
2 openstack-config --set /etc/glance/glance-api.conf database max_retries -1
3 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
4 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken auth_plugin password
5 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
6 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken username glance
7 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken password glancetest
8 openstack-config --set /etc/glance/glance-api.conf keystone_authtoken project_name services
9 openstack-config --set /etc/glance/glance-api.conf DEFAULT notification_driver messaging
10 openstack-config --set /etc/glance/glance-api.conf DEFAULT bind_host ADDRESS
11 openstack-config --set /etc/glance/glance-api.conf DEFAULT registry_host 192.168.110.10
12 openstack-config --set /etc/glance/glance-api.conf oslo_messaging_rabbit rabbit_hosts
      HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
13 openstack-config --set /etc/glance/glance-api.conf oslo_messaging_rabbit rabbit_ha_queues
      true
14 openstack-config --set /etc/glance/glance-registry.conf database connection
      mysql://glance:glancetest@192.168.110.10/glance
15 openstack-config --set /etc/glance/glance-registry.conf database max_retries -1
16 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
17 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken auth_plugin
      password
18 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
19 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken username glance
20 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken password
      glancetest
21 openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken project_name
      services
22 openstack-config --set /etc/glance/glance-registry.conf DEFAULT bind_host ADDRESS
```

O próximo passo consiste em sincronizar as bases de dados. Para tal no HA-CONTROLLER1 devemos executar o seguinte comando:

```
1 su glance -s /bin/sh -c "glance-manage db_sync"
```

Como o Glance é o serviço responsável pela gestão e armazenamento de imagens torna-se necessário configurar um *backend* em cada um dos controladores. O *backend* escolhido, já descrito anteriormente, é o NFS. Como tal em cada um dos controladores iremos configurar um servidor

de NFS de modo a termos as imagens replicadas pelos três controladores, para que em caso de falha de algum dos controladores estas continuem disponíveis.

Como tal na linha 1 criamos o diretório que será utilizado e no bloco [3-10] iniciamos e garantimos que os serviços são arrancados com o sistema. No bloco [12-14] indicamos qual o diretório que pode ser acedido remotamente e as permissões com que este pode ser acedido e manipulado. O bloco [16-20] serve para que a pasta remota do nosso controlador, bem como a dos outros dois controladores seja montada automaticamente com o sistema. Por fim nas linhas [22-24] atribuímos o Glance como dono do diretório, reiniciamos o servidor nfs e montamos as diretorias remotas que foram adicionadas ao ficheiro *fstab*.

```
1 mkdir /opt/glance_shared
2 chmod -R 777 /opt/glance_shared
3 systemctl enable rpcbind
4 systemctl enable nfs-server
5 systemctl enable nfs-lock
6 systemctl enable nfs-idmap
7 systemctl start rpcbind
8 systemctl start nfs-server
9 systemctl start nfs-lock
10 systemctl start nfs-idmap
11
12 cat > /etc/exports << EOF
13 /opt/glance_shared 192.168.110.0/24(rw,sync,no_root_squash,no_all_squash)
14 EOF
15
16 cat > /etc/fstab << EOF
17 192.168.110.1:/opt/glance_shared /var/lib/glance nfs defaults 0 0
18 192.168.110.2:/opt/glance_shared /var/lib/glance nfs defaults 0 0
19 192.168.110.3:/opt/glance_shared /var/lib/glance nfs defaults 0 0
20 EOF
21
22 chown glance:nobody /var/lib/glance
23 systemctl restart nfs-server
24 mount -a
```

Para finalizar a configuração, só nos resta arrancar os serviços em todos os controladores e garantir novamente que a base de dados está sincronizada. Para sincronizar a base de dados basta executar o comando num controlador apenas.

```
1 systemctl start openstack-glance-registry
2 systemctl start openstack-glance-api
3 systemctl enable openstack-glance-registry
4 systemctl enable openstack-glance-api
5 glance-manage db_sync
```

Para garantir que o Glance ficou bem instalado e configurado, podemos fazer um teste rápido com a imagem do Cirros. Caso seja possível adicionar e listar a imagem do Cirros, então a instalação foi bem sucedida. Este teste pode ser feito em qualquer um dos três controladores.

```
1 . /root/keystonerc_admin
2 wget http://download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img
3 glance image-create --name "cirros" --disk-format qcow2 --container-format bare --file
      cirros-0.3.3-x86_64-disk.img --visibility public
4
5 glance image-list
```

4.10 Cinder

Como foi anteriormente descrito no capítulo 3, na secção 3.2.2, o Cinder é um serviço que serve como solução de armazenamento em blocos. Este é particularmente útil pois pode ser adicionado e removido sempre que necessário de uma instância. Utilizou-se para tal um backend NFS. A arquitetura do Cinder é constituída por seis componentes e está visível na figura 4.8

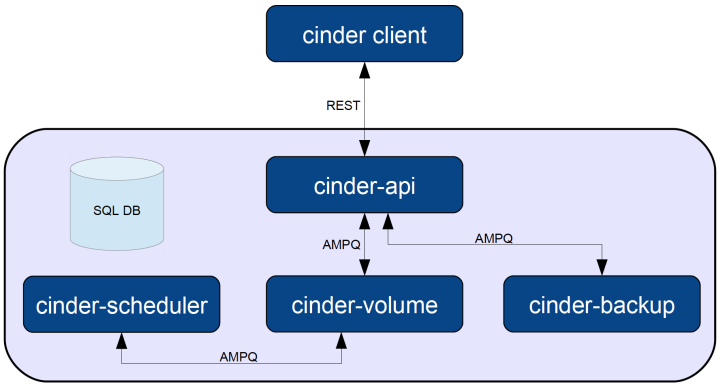


Figura 4.8: Imagem representativa da arquitetura do Cinder [13].

- Base de dados - armazena as informações;
- cinder client - potencial componente externo que faz pedidos à api;
- cinder-api - componente que recebe os pedidos http, os converte e comunica com as outras componentes;
- cinder-scheduler - decide onde alojar cada volume;
- cinder-volume - mapeia os blocos atribuídos ao volume;
- cinder-backup - responsável pelo backup dos volumes;

O bloco [3-8] contém o caminho para a conexão da base de dados, o número de tentativas de conexão que se podem fazer à base de dados, o tipo de estratégia que será utilizada para autenticação, o endereço IP do controlador, o endereço IP, o endereço de transporte de mensagens do RabbitMQ e o endereço da servidor do glance api; o bloco [9-24] contém dados referentes à autenticação no KeyStone como URL, plugin de autenticação (password), username, password e nome do projeto (services). O bloco [24-39] contém o tipo de driver de notificações (messaging), o hostname dos controladores que tem os *hosts* do RabbitMQ, os servidores de memcache disponíveis, o endereço do Glance Os seguintes passos para a instalação do Cinder devem ser executados em todos os controladores.

```
1 yum install -y openstack-cinder openstack-utils openstack-selinux python-memcached
2
3 openstack-config --set /etc/cinder/cinder.conf database connection
      mysql://cinder:cindertest@192.168.110.10/cinder
4 openstack-config --set /etc/cinder/cinder.conf database max_retries -1
5 openstack-config --set /etc/cinder/cinder.conf DEFAULT auth_strategy keystone
6 openstack-config --set /etc/cinder/cinder.conf DEFAULT my_ip ADDRESS
7 openstack-config --set /etc/cinder/cinder.conf DEFAULT transport_url rabbit://192.168.110.10
8 openstack-config --set /etc/cinder/cinder.conf DEFAULT glance_api_servers
      http://ADDRESS:9292
9 openstack-config --set /etc/cinder/cinder.conf DEFAULT auth_strategy keystone
10 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000
11 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_url
      http://192.168.110.10:35357
12 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken memcached_servers
      192.168.110.10:11211
13 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_type password
14 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken project_domain_name
      default
15 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken user_domain_name default
16 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken project_name service
17 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken username cinder
18 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken password cindertest
19 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
20 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_plugin password
21 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
22 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken username cinder
23 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken password cindertest
24 openstack-config --set /etc/cinder/cinder.conf keystone_authtoken project_name services
25 openstack-config --set /etc/cinder/cinder.conf DEFAULT notification_driver messaging
26 openstack-config --set /etc/cinder/cinder.conf DEFAULT glance_host 192.168.110.10
27 openstack-config --set /etc/cinder/cinder.conf DEFAULT memcache_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
```

```
28 openstack-config --set /etc/cinder/cinder.conf oslo_messaging_rabbit rabbit_hosts
       HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
29 openstack-config --set /etc/cinder/cinder.conf oslo_messaging_rabbit rabbit_ha_queues true
```

O próximo passo consiste configurar o driver NFS para o Cinder. Nesta instalação utilizamos o servidor já instalado e configurado para o Glance, uma vez que como estamos a utilizador os mesmos controladores iremos reaproveitar a configuração já feita anteriormente. Apenas são necessárias alguns acrescentos: nas linhas [1-3] é criado o ficheiro que contém o caminho para o diretório remoto NFS. No bloco [7-10] é adicionado ao ficheiro de configuração do Cinder o caminho para o ficheiro que contém o diretório remoto, as opções do NFS e o driver utilizado pelo Cinder.

```
1 cat > /etc/cinder/nfs_exports << EOF
2 ADDRESS:/opt/glance_shared
3 EOF
4
5 chown root:cinder /etc/cinder/nfs_exports
6 chmod 0640 /etc/cinder/nfs_exports
7 openstack-config --set /etc/cinder/cinder.conf DEFAULT nfs_shares_config
      /etc/cinder/nfs_exports
8 openstack-config --set /etc/cinder/cinder.conf DEFAULT nfs_sparsed_volumes true
9 openstack-config --set /etc/cinder/cinder.conf DEFAULT nfs_mount_options v3
10 openstack-config --set /etc/cinder/cinder.conf DEFAULT volume_driver
      cinder.volume.drivers.nfs.NfsDriver
```

Para finalizar a configuração, só nos resta garantir que a base de dados está sincronizada (linha 1) bem como arrancar os serviços (bloco [2-7]). A sincronização da base de dados deve ser feita apenas em um dos controladores.

```
1 su cinder -s /bin/sh -c "cinder-manage db sync"
2 systemctl start openstack-cinder-api
3 systemctl start openstack-cinder-schedule
4 systemctl start openstack-cinder-volume
5 systemctl enable openstack-cinder-api
6 systemctl enable openstack-cinder-scheduler
7 systemctl enable openstack-cinder-volume
```

Para garantir que o Cinder ficou bem instalado e configurado, podemos fazer um teste rápido, criando para tal um volume. Caso seja possível criar, estender e apagar o volume, então é porque o Cinder está a funcionar corretamente.

```
1 . /root/keystonerc_demo
2 cinder create --display-name test 1
3 cinder extend test 4
4 cinder delete test
```

4.11 Swift

Como foi anteriormente descrito no capítulo 3, na secção 3.2.10, o Swift é um serviço do OpenStack que também é conhecido como Object Storage e é a solução de armazenamento permanente que as instâncias do OpenStack utilizam. A alta disponibilidade neste serviço é crucial uma vez que caso algum dos discos fique em falha é necessário que exista uma ou várias cópias do mesmo para que a sua recuperação seja possível.

Os seguintes passos para a instalação do Swift devem ser executados em todos os controladores. É necessário uma partição adicional ou um disco rígido adicional para cada controlador para se puder usar o Swift, uma vez que este é o serviço de armazenamento e utiliza xfs como sistema de ficheiros.

```
yum install -y openstack-swift-object openstack-swift-container
    openstack-swift-account openstack-swift-proxy openstack-utils rsync xfsprogs
```

O próximo passo consiste criar o sistema de ficheiros xfs (linha 1) para cada disco adicional e deixar o sistema de ficheiros devidamente montado (linha 3) e pronto a ser utilizado (linha 4).

```
1 mkfs.xfs /dev/sdc
2 mkdir -p /srv/node/vdb
3 echo "/dev/sdc /srv/node/vdb xfs defaults 1 2" >> /etc/fstab
4 mount -a
5 chown -R swift:swift /srv/node
6 restorecon -R /srv/node
```

De seguida é necessário configurar os serviços de conta, *container* e de objeto bem como proceder à sua inicialização. Para cada um destes serviços são configuradas duas coisas: endereço do controlador e dispositivo de armazenamento. Por exemplo o bloco [1-2]. O dispositivo de armazenamento é o sistema de ficheiros que montamos anteriormente. O bloco [9-14] inicia os serviços e garante que estes são arrancados quando o sistema.

```
1 openstack-config --set /etc/swift/object-server.conf DEFAULT bind_ip ADDRESS
2 openstack-config --set /etc/swift/object-server.conf DEFAULT devices /srv/node
3 openstack-config --set /etc/swift/account-server.conf DEFAULT bind_ip ADDRESS
4 openstack-config --set /etc/swift/account-server.conf DEFAULT devices /srv/node
5 openstack-config --set /etc/swift/container-server.conf DEFAULT bind_ip ADDRESS
6 openstack-config --set /etc/swift/container-server.conf DEFAULT devices /srv/node
7 chown -R root:swift /etc/swift
8
9 systemctl start openstack-swift-account
10 systemctl start openstack-swift-container
11 systemctl start openstack-swift-object
12 systemctl enable openstack-swift-account
```

```
13 systemctl enable openstack-swift-container
14 systemctl enable openstack-swift-object
```

O próximo passo será configurar o *proxy server* e o *object expirer*. Para o *proxy-server* (bloco [1-8]) é necessário configurar o endereço de autenticação, o plugin de autenticação, o utilizador e password (swift/swifttest), o nome do projeto (services), o endereço dos servidores de memcache e o endereço do controlador. Para o *object expirer* (bloco [9-10]) apenas é necessário configurar os endereços dos servidores de memcache e o número de objetos de replicação (concurrency).

```
1 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken auth_uri
      http://192.168.110.10:5000/
2 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken auth_plugin password
3 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken auth_url
      http://192.168.110.10:35357/
4 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken username swift
5 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken password swifttest
6 openstack-config --set /etc/swift/proxy-server.conf filter:authtoken project_name services
7 openstack-config --set /etc/swift/proxy-server.conf filter:cache memcache_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
8 openstack-config --set /etc/swift/proxy-server.conf DEFAULT bind_ip ADDRESS
9 openstack-config --set /etc/swift/object-expirer.conf filter:cache memcache_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
10 openstack-config --set /etc/swift/object-expirer.conf object-expirer concurrency 3
```

O último passo na configuração é iniciar os serviços nos controladores e garantir que estes serviços são automaticamente iniciados quando o sistema arranca.

```
1 systemctl start openstack-swift-proxy
2 systemctl enable openstack-swift-proxy
3 systemctl start openstack-swift-object-expirer
4 systemctl enable openstack-swift-object-expirer
```

Para garantir que o Swift ficou bem instalado e configurado, podemos fazer um teste rápido: para tal iremos copiar um ficheiro para o armazenamento do Swift; caso seja fazer *upload* do ficheiro e o seu respetivo *download* significa que o Swift está a funcionar corretamente.

```
1 . /root/keystonerc_admin
2 swift list
3 swift upload test /tmp/cirros-0.3.3-x86_64-disk.img
4 swift list
5 swift list test
6 swift download test tmp/cirros-0.3.3-x86_64-disk.img
```

4.12 Neutron

Como foi anteriormente descrito no capítulo 3, na secção 3.2.7, o Neutron é o serviço de rede do OpenStack e foi criado para fornecer rede como um serviço (Software-Defined Networking - SDN) entre dispositivos de interface geridos por outros serviços, como por exemplo pelo Nova. A arquitetura do neutron é constituída por cinco componentes e está visível na figura 4.9.

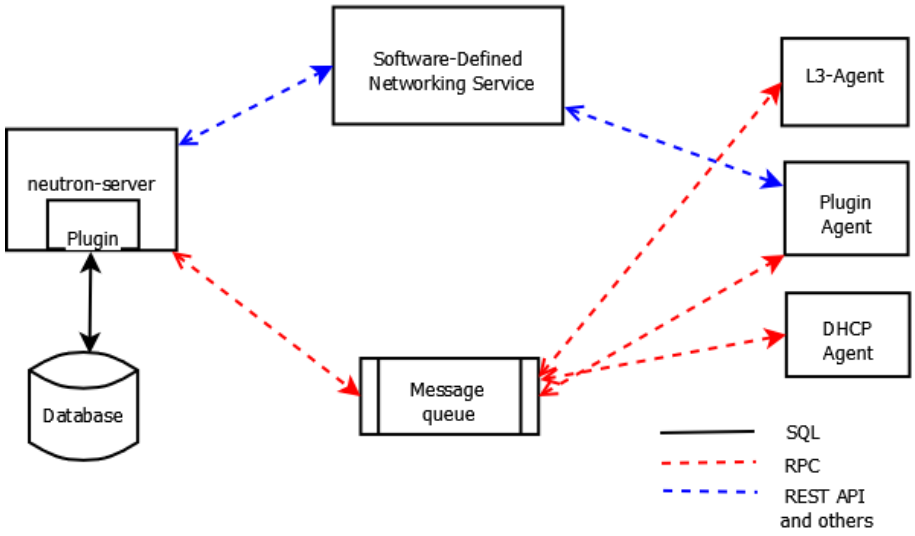


Figura 4.9: Imagem representativa da arquitetura do Neutron [14].

- neutron-server - É executado no nó da rede para responder à API. Ele é o responsável pelo encaminhamento de cada porta. É necessário acesso a uma BD persistente, feito através de plugins, que comunicam com a BD através do RabbitMQ;
- plugin agent - Corre em cada nó de computação para gerir o switch virtual (OpenvSwitch);
- L3-agent - Responsável pelo encaminhamento L3 / NAT para as VMs acederem à rede externa;
- DHCP agent - Responsável pelo servidor DHCP que cada projeto dispõe;
- SDN service - Fornece serviços de rede adicionais a inquilinos;

Os seguintes passos para a instalação do Neutron devem ser executados em todos os controladores.

```
yum install -y openstack-neutron openstack-neutron-openvswitch
    openstack-neutron-ml2 openstack-utils openstack-selinux
```

Na configuração do *Neutron-Server* é necessário interliga-lo com o KeyStone (bloco [3-10]), com o Nova (bloco [16-26]) e com o plugin ML2 (bloco [27-47]). É também necessário indicar qual o endereço dos controladores do RabbitMQ (linhas 13-15). É também necessário indicar o caminho do endereço para a conexão à base de dados. A configuração para o KeyStone é idêntica a todas as já realizadas anteriormente. Para o configurar com o Nova é necessário indicar o nome da região (regionOne), o nome do projeto (services), o tipo de autenticação bem como o utilizador e a password (nova/novatest) e o URL do serviço. Para a configuração do ML2 foi utilizada a configuração por defeito proposta pela redhat [86] para a utilização de ML2: o type_drivers (linha 33) possui a lista de drivers que são carregados, tenant_network_types (linha 34) possui o tipo de Vlan a ser utilizada na rede inquilina, o mechanism_drivers (linha 35) possui o tipo de driver utilizado, o ml2_type_flat (linha 36) possui a lista de nomes com os quais as redes podem ser criadas (* para todos), o ml2_type_vxlan (linha 37) possui a gama de IDs VXLAN disponíveis para alocação de rede, o ml2_type_vxlan (linha 39) possui o endereço do servidor de multicast e o securitygroup (linhas 40 e 41) permite ativar os grupos de segurança e indicar o driver a ser utilizado como firewall (neutron.agent.linux.iptables_firewall).

```
1  openstack-config --set /etc/neutron/neutron.conf DEFAULT bind_host ADDRESS
2  openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
3  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_uri
       http://192.168.110.10:5000/
4  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_plugin password
5  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_url
       http://192.168.110.10:35357/
6  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken memcached_servers
       192.168.110.10:11211
7  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken username neutron
8  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken password neutrontest
9  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken project_name services
10 openstack-config --set /etc/neutron/neutron.conf DEFAULT keystone_authtoken auth_type
       password
11 openstack-config --set /etc/neutron/neutron.conf database connection
       mysql://neutron:neutrontest@192.168.110.10:3306/neutron
12 openstack-config --set /etc/neutron/neutron.conf database max_retries -1
13 openstack-config --set /etc/neutron/neutron.conf DEFAULT notification_driver
       neutron.openstack.common.notifier.rpc_notifier
14 openstack-config --set /etc/neutron/neutron.conf oslo_messaging_rabbit rabbit_hosts
       HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
15 openstack-config --set /etc/neutron/neutron.conf oslo_messaging_rabbit rabbit_ha_queues
       true
16 openstack-config --set /etc/neutron/neutron.conf nova nova_region_name regionOne
17 openstack-config --set /etc/neutron/neutron.conf nova project_domain_id default
```

```
18 openstack-config --set /etc/neutron/neutron.conf nova project_name services
19 openstack-config --set /etc/neutron/neutron.conf nova user_domain_id default
20 openstack-config --set /etc/neutron/neutron.conf nova password novatest
21 openstack-config --set /etc/neutron/neutron.conf nova username nova
22 openstack-config --set /etc/neutron/neutron.conf nova auth_url http://192.168.110.10:35357/
23 openstack-config --set /etc/neutron/neutron.conf nova auth_plugin password
24 openstack-config --set /etc/neutron/neutron.conf nova auth_type = password
25 openstack-config --set /etc/neutron/neutron.conf DEFAULT
       notify_nova_on_port_status_changes True
26 openstack-config --set /etc/neutron/neutron.conf DEFAULT
       notify_nova_on_port_data_changes True
27 openstack-config --set /etc/neutron/neutron.conf DEFAULT core_plugin ml2
28 openstack-config --set /etc/neutron/neutron.conf DEFAULT service_plugins router
29 openstack-config --set /etc/neutron/neutron.conf DEFAULT router_scheduler_driver
       neutron.scheduler.l3_agent_scheduler.ChanceScheduler
30 openstack-config --set /etc/neutron/neutron.conf DEFAULT transport_url
       rabbit://HA-CONTROLLER1:5672,HA-CONTROLLER2:5672,HA-CONTROLLER3:5672
31
32 ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
33 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 type_drivers
       local,gre,flat,vxlan,vlan
34 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 tenant_network_types vxlan
35 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 mechanism_drivers
       openvswitch
36 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_flat flat_networks \*
37 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre
       tunnel_id_ranges 10:10000
38 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_vxlan vni_ranges
       10:10000
39 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_vxlan vxlan_group
       224.0.0.1
40 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup
       enable_security_group True
41 openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup firewall_driver
       neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

No nó HA-CONTROLLER1 iremos sincronizar a base de dados para que a informação fique redundante e consistente nos outros controladores:

```
1 neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file
      /etc/neutron/plugin.ini upgrade head
```

O próximo passo será ligar o serviço do *neutron-server* bloco [1-4] e configurar o *OpenvSwitch* bem como os agentes, quer do *OpenvSwitch* quer do *DHCP* bloco [36-67]. A configuração utilizada é a proposta por defeito pela redhat para a utilização do OpenvSwitch com o Neutron, disponivel em [86]. Para esta é necessário proceder a algumas alterações como por exemplo

o nome da interface física dos nossos controladores. A interligação com o KeyStone é feita do mesmo modo que os outros serviços anteriormente descritos. É também necessário definir o número de controladores que temos em *send_arp_for_ha* bem como é necessário alterar o endereço de *metadata* para o endereço do nosso controlador VIP. Para que se consiga utilizar o *OpenvSwitch* é necessário preparar a nossa interface de rede para tal definindo-lhe o tipo para *ovs* (bloco [6-17]) e é necessário criar uma *bridge* sobre essa interface de rede para que seja essa *bridge* (bloco [20-32]). É obrigatório definir o tipo de driver que utilizamos na interface, uma vez que o nosso escolhido foi o OpenvSwitch então devemos defini-lo em *interface_driver* (linha 60).

Deverá ser realizado em todos os nós:

```
1 systemctl start neutron-server
2 systemctl enable neutron-server
3 systemctl enable openvswitch
4 systemctl start openvswitch
5
6 cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
7 DEVICE=eth0
8 ONBOOT=yes
9 DEVICETYPE=ovs
10 TYPE=OVSPort
11 OVS_BRIDGE=br-eth0
12 ONBOOT=yes
13 BOOTPROTO=none
14 VLAN=yes
15 MTU="1500"
16 NM_CONTROLLED=no
17 EOF
18
19
20 cat > /etc/sysconfig/network-scripts/ifcfg-br-eth0 << EOF
21 DEVICE=br-eth0
22 DEVICETYPE=ovs
23 OVSBOOTPROTO=none
24 TYPE=OVSBridge
25 ONBOOT=yes
26 BOOTPROTO=static
27 NETWORK=192.168.110.0
28 NETMASK=255.255.255.0
29 IPADDR=ADDRESS
30 MTU="1500"
31 NM_CONTROLLED=no
32 EOF
33
34 systemctl restart network
35
```

```
36 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent tunnel_types
       vxlan
37 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent
       vxlan_udp_port 4789
38 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs local_ip
       ADDRESS
39 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs enable_tunneling
       True
40 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs integration_bridge
       br-int
41 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs tunnel_bridge
       br-tun
42 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs bridge_mappings
       physnet1:br-eth0
43 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs
       network_vlan_ranges physnet1
44 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini securitygroup
       firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
45 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent l2_population
       False
46
47 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT auth_strategy keystone
48 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT auth_url
       http://192.168.110.10:35357/v2.0
49 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT auth_host 192.168.110.10
50 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT auth_region regionOne
51 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT admin_tenant_name
       services
52 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT admin_user neutron
53 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT admin_password
       neutrontest
54 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT nova_metadata_ip
       192.168.110.10
55 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT nova_metadata_port 8775
56 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT
       metadata_proxy_shared_secret metatest
57 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT metadata_workers 4
58 openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT metadata_backlog 2048
59
60 openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT interface_driver
       neutron.agent.linux.interface.OVSInterfaceDriver
61 openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT dhcp_driver
       neutron.agent.linux.dhcp.Dnsmasq
62 openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT enable_isolated_metadata True
63 openstack-config --set /etc/neutron/l3_agent.ini DEFAULT interface_driver
       neutron.agent.linux.interface.BridgeInterfaceDriver
64 openstack-config --set /etc/neutron/l3_agent.ini DEFAULT handle_internal_only_routers True
```

```
65 openstack-config --set /etc/neutron/l3_agent.ini DEFAULT send_arp_for_ha 3
66 openstack-config --set /etc/neutron/l3_agent.ini DEFAULT metadata_ip 192.168.110.10
67 openstack-config --set /etc/neutron/l3_agent.ini DEFAULT external_network_bridge
```

O último passo na configuração do Neutron é ligar os serviços anteriormente instalados e garantir que estes são iniciados automaticamente no arranque do sistema.

```
1 systemctl start neutron-openvswitch-agent
2 systemctl start neutron-dhcp-agent
3 systemctl start neutron-l3-agent
4 systemctl start neutron-metadata-agent
5 systemctl enable neutron-openvswitch-agent
6 systemctl enable neutron-dhcp-agent
7 systemctl enable neutron-l3-agent
8 systemctl enable neutron-metadata-agent
9 systemctl enable neutron-ovs-cleanup
```

Quando se reinicia o cluster inteiro, caso o Galera não ligue corretamente, o neutron-server ficará à espera até ocorrer o timeout e este falhar. Para contornar este problema podemos criar um ficheiro que o reiniciará sempre que este falhar a arrancar.

```
1 cat > /etc/systemd/system/neutron-server.service.d/restart.conf << EOF
2 [Service]
3 Restart=on-failure
```

4.13 Nova

Como foi anteriormente descrito no capítulo 3, na secção 3.2.8, o Nova é o serviço que instalado nos controladores e nos nós de computação permite iniciar e parar instâncias virtuais, atribuir e remover floating ip's, criar Snapshots de instâncias, criar e editar grupos de segurança e adicionar ou remover volumes. Este é o serviço mais importante do OpenStack pois juntamente com o KeyStone é o que controla toda a infraestrutura. Anteriormente à existência do Neutron, o Nova era também o serviço que controlava a rede.

A arquitetura do nova é constituída por cinco componentes:

- Base de dados - serve para armazenamento de dados;
- nova-api - recebe pedidos http e comunica com outros componentes;
- nova-scheduler - decide onde vai ser alojada cada instância;
- nova-compute - gere a comunicação entre o hypervisor e as instâncias;

- nova-conductor - trata de pedidos como criar uma instância;

Os seguintes passos para a instalação do Nova devem ser executados em todos os controladores.

```
yum install -y openstack-nova-console openstack-nova-novncproxy openstack-utils
    openstack-nova-api openstack-nova-conductor openstack-nova-scheduler
    python-cinderclient python-memcached
```

Na sua configuração devem ser indicados os endereços de memcache (linha 1); configurar o acesso por VNC às instâncias (bloco [2-5]); indicar o caminho do endereço para a conexão à base de dados (linha 6) bem como o número máximo de tentativas de conexão (linha 7); o tipo de estratégia utilizada para autenticação (linha 8); o endereço para onde o nó de computação vai comunicar, o de meta dados e a respetiva porta e o do Glance (bloco [9-13]); o tipo de rede, o driver de firewall e de virtualização utilizado e os grupos de segurança (bloco [14-17]); o RabbitMQ (bloco [20-21]); o Neutron (bloco [22-33]) onde são indicados o utilizador e a password, a região, o nome do projeto e o endereço do Neutron; é ainda indicado o caminho do endereço para a conexão à base de dados do nova-api, pois este componente utiliza uma base de dados separada e por fim, são indicados os dados de autenticação do Nova, bem como nome do projeto e URL de autenticação (bloco [35-42]). O próximo passo consiste em configurar o *Nova* no controlador.

```
1 openstack-config --set /etc/nova/nova.conf DEFAULT memcached_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
2 openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy_host ADDRESS
3 openstack-config --set /etc/nova/nova.conf vnc novncproxy_base_url
      http://192.168.110.10:6080/vnc_auto.html
4 openstack-config --set /etc/nova/nova.conf vnc vncserver_proxyclient_address ADDRESS
5 openstack-config --set /etc/nova/nova.conf vnc vncserver_listen ADDRESS
6 openstack-config --set /etc/nova/nova.conf database connection
      mysql://nova:novatest@192.168.110.10/nova
7 openstack-config --set /etc/nova/nova.conf database max_retries -1
8 openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
9 openstack-config --set /etc/nova/nova.conf DEFAULT osapi_compute_listen ADDRESS
10 openstack-config --set /etc/nova/nova.conf DEFAULT metadata_host ADDRESS
11 openstack-config --set /etc/nova/nova.conf DEFAULT metadata_listen ADDRESS
12 openstack-config --set /etc/nova/nova.conf DEFAULT metadata_listen_port 8775
13 openstack-config --set /etc/nova/nova.conf DEFAULT glance_host 192.168.110.10
14 openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
      nova.network.neutronv2.api.API
15 openstack-config --set /etc/nova/nova.conf DEFAULT firewall_driver
      nova.virt.firewall.NoopFirewallDriver
16 openstack-config --set /etc/nova/nova.conf libvirt vif_driver
      nova.virt.libvirt.vif.LibvirtGenericVIFDriver
17 openstack-config --set /etc/nova/nova.conf DEFAULT security_group_api neutron
18 openstack-config --set /etc/nova/nova.conf cinder cinder_catalog_info
      volume:cinder:internalURL
```

```
openstack-config --set /etc/nova/nova.conf conductor use_local false
openstack-config --set /etc/nova/nova.conf oslo_messaging_rabbit rabbit_hosts
    HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
openstack-config --set /etc/nova/nova.conf oslo_messaging_rabbit rabbit_ha_queues True
openstack-config --set /etc/nova/nova.conf neutron service_metadata_proxy True
openstack-config --set /etc/nova/nova.conf neutron metadata_proxy_shared_secret metatest
openstack-config --set /etc/nova/nova.conf neutron url http://192.168.110.10:9696/
openstack-config --set /etc/nova/nova.conf neutron project_domain_id default
openstack-config --set /etc/nova/nova.conf neutron project_name services
openstack-config --set /etc/nova/nova.conf neutron user_domain_id default
openstack-config --set /etc/nova/nova.conf neutron username neutron
openstack-config --set /etc/nova/nova.conf neutron password neutrontest
openstack-config --set /etc/nova/nova.conf neutron auth_url http://192.168.110.10:35357/
openstack-config --set /etc/nova/nova.conf neutron auth_uri http://192.168.110.10:5000/
openstack-config --set /etc/nova/nova.conf neutron auth_plugin password
openstack-config --set /etc/nova/nova.conf neutron region_name regionOne
openstack-config --set /etc/nova/nova.conf api_database connection
    mysql://nova_api:novatest@192.168.110.10/nova_api
openstack-config --set /etc/nova/nova.conf api_database max_retries -1
openstack-config --set /etc/nova/api-paste.ini filter:authtoken auth_plugin password
openstack-config --set /etc/nova/api-paste.ini filter:authtoken auth_url
    http://192.168.110.10:35357/
openstack-config --set /etc/nova/api-paste.ini filter:authtoken username compute
openstack-config --set /etc/nova/api-paste.ini filter:authtoken password novatest
openstack-config --set /etc/nova/api-paste.ini filter:authtoken project_name services
openstack-config --set /etc/nova/api-paste.ini filter:authtoken auth_uri
    http://192.168.110.10:5000/
```

No nó HA-CONTROLLER1 iremos sincronizar as bases de dados:

```
su -s /bin/sh -c "nova-manage api_db sync" nova
su nova -s /bin/sh -c "nova-manage db sync"
```

O ultimo passo será ligar os serviços do *nova* e garantir que são automaticamente ligados com o arranque do sistema. Deverá ser realizado em todos os nós:

```
systemctl start openstack-nova-consoleauth
systemctl start openstack-nova-novncproxy
systemctl start openstack-nova-api
systemctl start openstack-nova-scheduler
systemctl start openstack-nova-conductor
systemctl enable openstack-nova-consoleauth
systemctl enable openstack-nova-novncproxy
systemctl enable openstack-nova-api
systemctl enable openstack-nova-scheduler
systemctl enable openstack-nova-conductor
```

4.14 Ceilometer

Como foi anteriormente descrito no capítulo 3, na secção 3.2.1, o Ceilometer é um serviço que tem como objetivo centralizar a monitorização e métrica de recursos para o OpenStack. Este é responsável pela recolha de dados sobre as máquinas e tem também a capacidade de serem definidas métricas como por exemplo tempo máximo de utilização de uma instância, sendo que ao fim a instância é desligada ou apagada.

Os seguintes passos para a instalação do Ceilometer devem ser executados em todos os controladores.

```
yum install -y openstack-ceilometer-api openstack-ceilometer-central
    openstack-ceilometer-collector openstack-ceilometer-common
    openstack-ceilometer-alarm python-ceilometer python-ceilometerclient
    python-redis
```

Como em todos os serviços anteriormente apresentados é necessário configurar a autenticação com o KeyStone (bloco [1-8]). É também necessário indicar os servidores de memcache (linha 10), bem como indicar os controladores onde o atua o RabbitMQ (bloco [11-13]). São definidos um conjunto de informações para autenticação na telemetria como utilizador, password, endereço IP de autenticação, nome do projeto, nome de região, e endereço interno de autenticação (bloco [14-22]). É ainda configurada na API do Ceilometer o endereço IP do controlador e a porta, bem como o caminho para o endereço de conexão à base de dados MongoDB, assim como o número máximo de tentativas de autenticação (bloco [23-28]). Por fim é adicionado ao ficheiro de configuração o URL do Redis, tornando assim o serviço em alta disponibilidade (linha 30 e 31).

```
1 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
2 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_type password
3 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
4 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken username
      ceilometer
5 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken password
      ceilometertest
6 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken project_name
      services
7 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken
      project_domain_name default
8 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken
      user_domain_name default
9 openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT rpcbackend rabbit
10 openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT memcache_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
```

```
openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT auth_strategy keystone
openstack-config --set /etc/ceilometer/ceilometer.conf oslo_messaging_rabbit rabbit_hosts
    HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
openstack-config --set /etc/ceilometer/ceilometer.conf oslo_messaging_rabbit
    rabbit_ha_queues true
openstack-config --set /etc/ceilometer/ceilometer.conf publisher telemetry_secret
    ceilometersecret
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials auth_type password
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials username ceilometer
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials tenant_name
    services
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials password
    ceilometertest
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials auth_url
    http://192.168.110.10:5000/v3
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials
    project_domain_name ceilometertest
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials region_name
    regionOne
openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials interface internalURL
openstack-config --set /etc/ceilometer/ceilometer.conf api host ADDRESS
openstack-config --set /etc/ceilometer/ceilometer.conf api port 8777
openstack-config --set /etc/ceilometer/ceilometer.conf database connection
mongodb://HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3:27017/ceilometer
?replicaSet=ceilometer
openstack-config --set /etc/ceilometer/ceilometer.conf database max_retries -1

openstack-config --set /etc/ceilometer/ceilometer.conf coordination backend_url
    'redis://HA-CONTROLLER1:26379?sentinel=mymaster&sentinel_fallback=
HA-CONTROLLER2:26379&sentinel_fallback=HA-CONTROLLER3:26379'
```

Para finalizar a configuração do Ceilometer resta apenas editar dois ficheiros em todos os controladores, sendo que iremos corrigir duas linhas de código que estavam a falhar. no */usr/lib/systemd/system/openstack-ceilometer-api.service* editar a linha original equivalente para:

```
ExecStart=/usr/bin/ceilometer-api"
```

e no /usr/bin/ceilometer-api editar as linhas originais equivalentes para:

```
parser.add_argument('--port', '-p', type=int, default=8777,
server = wss.make_server(my_ip, args.port, build_wsgi_app())
```

O último passo será ligar os serviços do *Ceilometer* e garantir que os serviços são iniciados automaticamente no arranque do sistema. Deverá ser realizado em todos os nós:

```
systemctl start openstack-ceilometer-central
```

```
2 systemctl enable openstack-ceilometer-central
3 systemctl start openstack-ceilometer-collector
4 systemctl enable openstack-ceilometer-collector
5 systemctl start openstack-ceilometer-api
6 systemctl enable openstack-ceilometer-api
7 systemctl start openstack-ceilometer-notification
8 systemctl enable openstack-ceilometer-notification
```

4.15 Heat

Como foi anteriormente descrito no capítulo 3, na secção 3.2.4, o Heat é um serviço que utiliza templates para fazer instalação automática de aplicações distribuídas no OpenStack. Os recursos da infraestrutura que podem ser descritos no template incluem servidores, ip's flutuantes, volumes, grupos de segurança e usuários. Este é também integrável com o Puppet, que é uma ferramenta para configuração de software em máquinas. Se for necessário alterar a infraestrutura apenas é necessário alterar o template do Heat, pois este adapta-se facilmente a qualquer alteração na infraestrutura. A arquitetura do heat é constituída por três componentes:

- heat-api - é uma API em REST que recebe pedidos e processa-os;
- heat-api-cfn - é uma API que é compativel com a AWS CloudFormation e processa pedidos e envia-os ao heat-engine;
- heat-engine - faz o trabalho principal que é executar os templates;

Os seguintes passos para a instalação do Heat devem ser executados em todos os controladores.

```
yum install -y openstack-heat-engine openstack-heat-api openstack-heat-api-cfn
    openstack-heat-api-cloudwatch python-heatclient openstack-utils
    python-glanceclient
```

Para que todos os utilizadores consigam criar *Heat Stacks* é necessário criar um domínio no KeyStone para o Heat. Para tal, no HA-CONTROLLER1 iremos criar esse mesmo domínio. No bloco [1-3] procedemos à criação de um papel chamado *heat_stack_user* e obtemos o *TOKEN_ID*. No bloco [5-7] procedemos à criação de um domínio chamado *heat*, à criação de um utilizador com a password *heattest* associado com o domínio anterior e por fim definimos esse utilizador como administrador desse domínio. Deverá ser substituído o termo *TOKEN_ID* pelo ID que será gerado.

```
1 . /root/keystonerc_admin
2 openstack role create heat_stack_user
3 openstack token issue
```

```
4
5 openstack --os-token=TOKEN_ID --os-url=http://192.168.110.10:5000/v3
      --os-identity-api-version=3 domain create heat --description "Owns users and projects
      created by heat"
6 openstack --os-token=TOKEN_ID --os-url=http://192.168.110.10:5000/v3
      --os-identity-api-version=3 user create --password heattest --domain heat
      --description "Manages users and projects created by heat" heat_domain_admin
7 openstack --os-token=TOKEN_ID --os-url=http://192.168.110.10:5000/v3
      --os-identity-api-version=3 role add --user heat_domain_admin --domain heat admin
```

O próximo passo consiste configurar o *Heat* em todos os controladores. No bloco [1-3] utilizamos os dados que criamos anteriormente como dominio, utilizador e password; no bloco [4-5] indicamos o caminho para o endereço de conexão à base de dados bem como o número máximo de tentativas de conexão; no bloco [6-7] configuramos a ligação com o KeyStone, tal como foi feito nos serviços anteriores. Na linha 17 indicamos quais são os servidores de memcache; no bloco [18-19] configuramos o heat_api, indicando-lhe o endereço do controlador; na linha 20 indicamos qual o endereço do servidor de metadados e no bloco [21-24] indicamos a configuração do RabbitMQ para a troca de mensagens com o Heat.

```
1 openstack-config --set /etc/heat/heat.conf DEFAULT stack_domain_admin_password heattest
2 openstack-config --set /etc/heat/heat.conf DEFAULT stack_domain_admin
      heat_domain_admin
3 openstack-config --set /etc/heat/heat.conf DEFAULT stack_user_domain_name heat
4 openstack-config --set /etc/heat/heat.conf database connection
      mysql://heat:heattest@192.168.110.10/heat
5 openstack-config --set /etc/heat/heat.conf database max_retries -1
6 openstack-config --set /etc/heat/heat.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
7 openstack-config --set /etc/heat/heat.conf keystone_authtoken auth_plugin password
8 openstack-config --set /etc/heat/heat.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
9 openstack-config --set /etc/heat/heat.conf keystone_authtoken username heat
10 openstack-config --set /etc/heat/heat.conf keystone_authtoken password heattest
11 openstack-config --set /etc/heat/heat.conf keystone_authtoken project_name services
12 openstack-config --set /etc/heat/heat.conf keystone_authtoken identity_uri
      http://192.168.110.10:35357
13 openstack-config --set /etc/heat/heat.conf keystone_authtoken admin_tenant_name services
14 openstack-config --set /etc/heat/heat.conf keystone_authtoken admin_user heat
15 openstack-config --set /etc/heat/heat.conf keystone_authtoken admin_password heattest
16 openstack-config --set /etc/heat/heat.conf ec2authtoken auth_uri
      http://192.168.110.10:5000/v2.0
17 openstack-config --set /etc/heat/heat.conf DEFAULT memcache_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
18 openstack-config --set /etc/heat/heat.conf heat_api bind_host ADDRESS
19 openstack-config --set /etc/heat/heat.conf heat_api_cfn bind_host ADDRESS
20 openstack-config --set /etc/heat/heat.conf DEFAULT heat_metadata_server_url
```

```
    192.168.110.10:8000
21 openstack-config --set /etc/heat/heat.conf oslo_messaging_rabbit rabbit_hosts
    HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
22 openstack-config --set /etc/heat/heat.conf oslo_messaging_rabbit rabbit_ha_queues true
23 openstack-config --set /etc/heat/heat.conf DEFAULT rpc_backend rabbit
24 openstack-config --set /etc/heat/heat.conf DEFAULT notification_driver
    heat.openstack.common.notifier.rpc_notifier
```

Para finalizar a configuração do Heat resta apenas sincronizar a base de dados para garantir que os dados são redundantes pelas bases de dados dos controladores. O seguinte comando deverá ser executado em apenas um controlador:

```
1 su heat -s /bin/sh -c "heat-manage db_sync"
```

Como último passo iremos iniciar os serviços referentes ao Heat e garantir que estes são iniciados automáticamente no arranque do sistema de todos os controladores:

```
1 systemctl start openstack-heat-api
2 systemctl start openstack-heat-api-cfn
3 systemctl start openstack-heat-api-cloudwatch
4 systemctl start openstack-heat-engine
5 systemctl enable openstack-heat-api
6 systemctl enable openstack-heat-api-cfn
7 systemctl enable openstack-heat-api-cloudwatch
8 systemctl enable openstack-heat-engine
```

4.16 Horizon

Como foi anteriormente descrito no capítulo 3, na secção 3.2.5, o Horizon é o *dashboard* oficial do OpenStack que serve para se gerirem serviços do OpenStack como o Nova, Neutron, Swift e Glance, agilizando e facilitando o processo da gestão da Cloud. Os seguintes passos para a instalação do Horizon-Dashboard devem ser executados em todos os controladores. Foi instalado e configurado o MemCached para que um utilizador possa rapidamente aceder à informação já consultada sem que os pedidos tenham que ser feitos novamente. O Horizon apenas recorre ao KeyStone para verificar as respetivas autenticações e autorizações e caso se verifique, utiliza a sua API para fazer os pedidos e obter as respostas das funcionalidades que pretende utilizar.

```
yum install -y mod_wsgi httpd mod_ssl python-memcached openstack-dashboard
```

É necessário definir uma chave secreta. Para tal iremos gerar uma chave aleatória e iremos substituir o termo **VALUE** pela chave gerada. A chave deverá ser gerada no HA-CONTROLLER1, e deverá ser utilizada pelos três controladores. A chave gerada será aleatória

utilizando hexadecimal, de comprimento 10, recorrendo ao openssl para a geração da mesma (linha 1). A linha 2 procede à alteração das ocorrências do termo *SECRET_KEY.**, substituindo o que encontra no ficheiro /etc/openstack-dashboard/local_settings por *SECRET_KEY = 'VALUE'*.

```
1 openssl rand -hex 10
2 sed -i -e "sSECRET_KEY.*SECRET_KEY = 'VALUE'g"
      /etc/openstack-dashboard/local_settings
```

O próximo passo consiste configurar o *local_settings* e o *httpd.conf* em todos os controladores. O que se pretende com o bloco de código [1-6] é que seja feita uma substituição de todas as ocorrências de: *ALLOWED_HOSTS* por * para que este seja alcançável por todos, de *memcached.MemcachedCache, LOCATION* o endereço dos hostname de memcache, de *OPENSTACK_HOST* para *192.168.110.10* e de *LOCAL_PATH* para */var/lib/openstack-dashboard*.

```
1 sed -i -e "s#ALLOWED_HOSTS.*ALLOWED_HOSTS = ['*',]#g" \
2 -e "s#locmem.LocMemCache'#memcached.MemcachedCache',\n\t'LOCATION' : [
      'HA-CONTROLLER1:11211', 'HA-CONTROLLER2:11211', 'HA-CONTROLLER3:11211',
      ]#g" \|
3 -e 's#OPENSTACK_HOST =.*#OPENSTACK_HOST = "192.168.110.10"#g' \
4 -e "s#^LOCAL_PATH.*#LOCAL_PATH = '/var/lib/openstack-dashboard'#g" \
5 /etc/openstack-dashboard/local_settings
```

Como último passo iremos iniciar os serviços referentes ao Horizon em todos os controladores:

```
1 systemctl daemon-reload
2 systemctl restart httpd
```

4.17 Trove

Como foi anteriormente descrito no capítulo 3, na secção 3.2.11, o Trove é utilizado no OpenStack para base de dados como serviço. Assim os utilizadores podem de forma rápida e eficiente utilizar os recursos de uma base de dados, sem terem que se preocupar com tarefas de gestão complexas sobre a mesma.

A arquitetura do trove é constituída por três componentes e está visível na figura 4.10.

- trove-api - fornece uma API REST para gerir as instâncias do Trove.;
- trove-conductor - responsável por gerir o ciclo de vida das instâncias e executar operações nas bases de dados das instâncias;

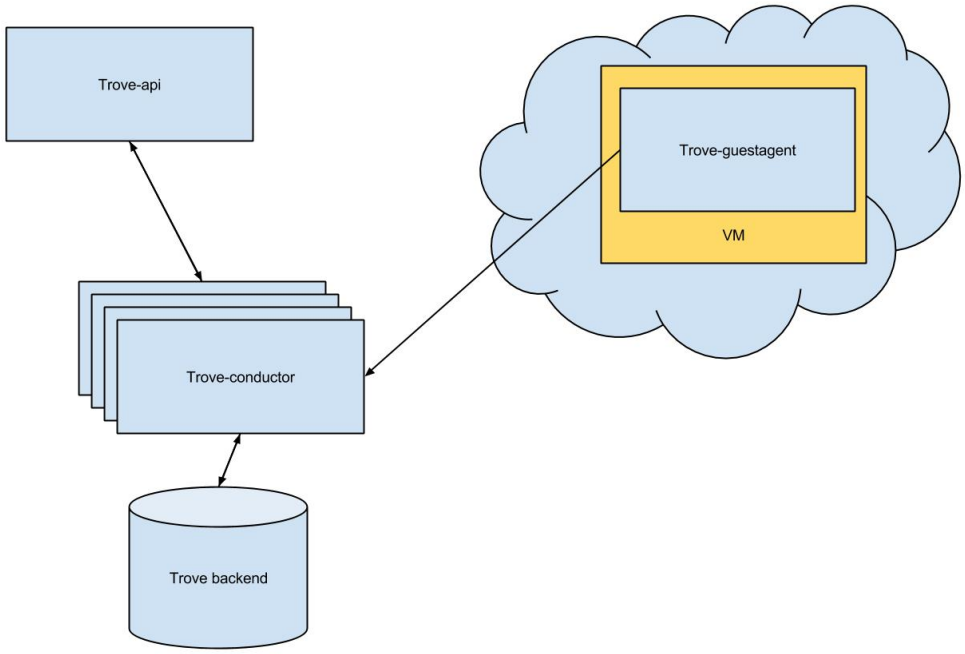


Figura 4.10: Imagem representativa da arquitetura do Neutron [15].

- trove-guestagent - responsável por gerir e executar operações na própria base de dados de convidado. Ele espera por mensagens RPC que recebe através do barramento e executa a operação solicitada;

Os seguintes passos para a instalação do Trove devem ser executados em todos os controladores a menos que seja dito algo em contrário.

```
yum install -y openstack-trove python-troveclient
```

Deverá substituir-se o termo *SERVICES_TENANT* pelo ID respetivo gerado pelo KeyStone. No bloco [1-4] são configurados os dados gerais do trove como endereço do controlador, diretorio onde vão ser guardados os logs, URL de autenticação e nome da região. No bloco [5-8] é configurado o acesso ao RabbitMQ para a troca de mensagens. No bloco [9-16] é indicamos o caminho para o endereço de conexão à base de dados bem como o número máximo de tentativas de conexão assim como as configurações anteriormente referidas sobre o KeyStone como tipo de autenticação, utilizador, password, nome do projeto e URL. No bloco [17-31] é configurado o *trove-conductor*, sendo que é necessário indicar o acesso ao RabbitMQ para a troca de mensagens, o endereço para o qual o Trove deve autenticar-se, o nome do log que será gerado, o endereço de conexão à base de dados bem como o número máximo de tentativas e os dados do KeyStone, tal como referido anteriormente. No bloco [32-50] é configurado o *trove-taskmanager*, sendo que a sua configuração é idêntica à referida nos blocos anteriores, com a diferença de que na linha 50 é

definido o driver de rede. A linha 51 serve para configurar o tipo de base de dados padrão uma vez que o painel não o permite. A instalação e configuração do trove foi baseada na proposta pela redhat em [87].

```
1 openstack-config --set /etc/trove/trove.conf DEFAULT bind_host ADDRESS
2 openstack-config --set /etc/trove/trove.conf DEFAULT log_dir /var/log/trove
3 openstack-config --set /etc/trove/trove.conf DEFAULT trove_auth_url
      http://192.168.110.10:35357/v2.0
4 openstack-config --set /etc/trove/trove.conf DEFAULT os_region_name regionOne
5 openstack-config --set /etc/trove/trove.conf oslo_messaging_rabbit rabbit_hosts
      HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
6 openstack-config --set /etc/trove/trove.conf oslo_messaging_rabbit rabbit_ha_queues true
7 openstack-config --set /etc/trove/trove.conf oslo_messaging_rabbit rabbit_password guest
8 openstack-config --set /etc/trove/trove.conf DEFAULT rpc_backend rabbit
9 openstack-config --set /etc/trove/trove.conf database connection
      mysql://trove:trovetest@192.168.110.10/trove
10 openstack-config --set /etc/trove/trove.conf database max_retries -1
11 openstack-config --set /etc/trove/trove.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
12 openstack-config --set /etc/trove/trove.conf keystone_authtoken auth_plugin password
13 openstack-config --set /etc/trove/trove.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
14 openstack-config --set /etc/trove/trove.conf keystone_authtoken username trove
15 openstack-config --set /etc/trove/trove.conf keystone_authtoken password trovetest
16 openstack-config --set /etc/trove/trove.conf keystone_authtoken project_name services
17 openstack-config --set /etc/trove/trove-conductor.conf DEFAULT trove_auth_url
      http://192.168.110.10:35357/v2.0
18 openstack-config --set /etc/trove/trove-conductor.conf DEFAULT os_region_name regionOne
19 openstack-config --set /etc/trove/trove-conductor.conf DEFAULT log_file
      trove-conductor.log
20 openstack-config --set /etc/trove/trove-conductor.conf oslo_messaging_rabbit rabbit_hosts
      HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
21 openstack-config --set /etc/trove/trove-conductor.conf oslo_messaging_rabbit
      rabbit_ha_queues true
22 openstack-config --set /etc/trove/trove-conductor.conf oslo_messaging_rabbit
      rabbit_password guest
23 openstack-config --set /etc/trove/trove-conductor.conf DEFAULT rpc_backend rabbit
24 openstack-config --set /etc/trove/trove-conductor.conf database connection
      mysql://trove:trovetest@192.168.110.10/trove
25 openstack-config --set /etc/trove/trove-conductor.conf database max_retries -1
26 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken auth_uri
      http://192.168.110.10:5000/
27 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken auth_plugin
      password
28 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken auth_url
      http://192.168.110.10:35357/
29 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken username trove
```

```
30 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken password trovetest
31 openstack-config --set /etc/trove/trove-conductor.conf keystone_authtoken project_name
       services
32 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT trove_auth_url
       http://192.168.110.10:35357/v2.0
33 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT os_region_name
       regionOne
34 openstack-config --set /etc/trove/trove-taskmanager.conf oslo_messaging_rabbit
       rabbit_hosts HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
35 openstack-config --set /etc/trove/trove-taskmanager.conf oslo_messaging_rabbit
       rabbit_ha_queues true
36 openstack-config --set /etc/trove/trove-taskmanager.conf oslo_messaging_rabbit
       rabbit_password guest
37 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT rpc_backend rabbit
38 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT nova_proxy_admin_user
       trove
39 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT
       nova_proxy_admin_pass trovetest
40 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT
       nova_proxy_admin_tenant_name SERVICES_TENANT_ID
41 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT log_file
       trove-taskmanager.log
42 openstack-config --set /etc/trove/trove-taskmanager.conf database connection
       mysql://trove:trovetest@192.168.110.10/trove
43 openstack-config --set /etc/trove/trove-taskmanager.conf database max_retries -1
44 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken auth_uri
       http://192.168.110.10:5000/
45 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken auth_plugin
       password
46 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken auth_url
       http://192.168.110.10:35357/
47 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken username trove
48 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken password
       trovetest
49 openstack-config --set /etc/trove/trove-taskmanager.conf keystone_authtoken project_name
       services
50 openstack-config --set /etc/trove/trove-taskmanager.conf DEFAULT network_driver
       trove.network.neutron.NeutronDriver
51 openstack-config --set /etc/trove/trove.conf DEFAULT default_datastore mysql
52 openstack-config --set /etc/trove/trove.conf DEFAULT network_label_regex .*
```

O próximo passo consiste em sincronizar a base de dados dos diferentes controladores para que as bases de dados dos diferentes controladores fiquem redundantes. O seguinte comando deverá ser executado em apenas um dos controladores:

```
1 su trove -s /bin/sh -c "trove-manage db_sync"
2 trove-manage datastore_update mysql ''
```

Como último passo iremos iniciar e garantir que os serviços referentes ao Trove são iniciados no arranque do sistema em todos os controladores:

```
1 systemctl enable openstack-trove-api
2 systemctl enable openstack-trove-taskmanager
3 systemctl enable openstack-trove-conductor
4 systemctl start openstack-trove-api
5 systemctl start openstack-trove-taskmanager
6 systemctl start openstack-trove-conductor
```

4.18 Sahara

Como foi anteriormente descrito no capítulo 3, na secção 3.2.9, o Sahara é um serviço que é essencialmente dedicado a quem procura análise de grandes quantidades de dados (data mining). A arquitetura do Sahara é constituída por vários componentes como:

- Secure Storage Access Layer - mantém os dados de autenticação como password e chaves privadas num local seguro;
- Provisioning Engine - componente responsável pela comunicação com o Nova;
- Vendor Plugins - mecanismo responsável pela execução de frameworks de processamento de dados;
- Elastic Data Processing - responsável pelo agendamento e gestão de processamento de dados em clusters Sahara;
- Sahara pages - uma GUI localizada no Horizon;

Os seguintes passos para a instalação do Sahara devem ser executados em todos os controladores a menos que seja dito algo em contrário. No bloco [3-5] são configurados o endereço do controlador, o uso de floating ips e o uso de neutron para que seja possível trocar grandes dados pela rede entre diferentes instâncias. No bloco [6-10] são definidos os controladores onde o RabbitMQ está instalado, a porta utilizada bem como o uso de filas em alta disponibilidade. Na linha 12 indicamos o caminho para o endereço de conexão à base de dados. No bloco [13-21] configuramos os dados referentes à ligação com o KeyStone, como foi feito nos serviços anteriores. Na linha 22 indicamos o ficheiro de log que será utilizado para o registo de ocorrências no sahara.

```
1 yum install -y openstack-sahara-api openstack-sahara-engine openstack-sahara-common
      openstack-sahara python-saharaclient
2
```

```
openstack-config --set /etc/sahara/sahara.conf DEFAULT host ADDRESS
openstack-config --set /etc/sahara/sahara.conf DEFAULT use_floating_ips True
openstack-config --set /etc/sahara/sahara.conf DEFAULT use_neutron True
openstack-config --set /etc/sahara/sahara.conf DEFAULT rpc_backend rabbit
openstack-config --set /etc/sahara/sahara.conf oslo_messaging_rabbit rabbit_hosts
    HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
openstack-config --set /etc/sahara/sahara.conf oslo_messaging_rabbit rabbit_port 5672
openstack-config --set /etc/sahara/sahara.conf oslo_messaging_rabbit rabbit_use_ssl False
openstack-config --set /etc/sahara/sahara.conf oslo_messaging_rabbit rabbit_ha_queues true
openstack-config --set /etc/sahara/sahara.conf DEFAULT notification_topics notifications
openstack-config --set /etc/sahara/sahara.conf database connection
    mysql://sahara:saharatest@192.168.110.10/sahara
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken auth_uri
    http://192.168.110.10:5000/
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken auth_plugin password
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken auth_url
    http://192.168.110.10:35357/
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken username sahara
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken password saharatest
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken project_name services
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken admin_tenant_name
    services
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken admin_user sahara
openstack-config --set /etc/sahara/sahara.conf keystone_authtoken admin_password saharatest
openstack-config --set /etc/sahara/sahara.conf DEFAULT log_file /var/log/sahara/sahara.log
```

O próximo passo consiste em sincronizar a base de dados dos diferentes controladores para garantir que os dados entre as bases de dados dos diferentes controladores são redundantes. O seguinte comando deverá ser executado no HA-CONTROLLER1:

```
sahara-db-manage --config-file /etc/sahara/sahara.conf upgrade head
```

Como último passo iremos iniciar e garantir que os serviços referentes ao Sahara são iniciados automaticamente no arranque do sistema em todos os controladores:

```
systemctl enable openstack-sahara-api
systemctl enable openstack-sahara-engine
systemctl start openstack-sahara-api
systemctl start openstack-sahara-engine
```

4.19 Dificuldades Encontradas

Durante a implementação dos controladores foram surgindo algumas dificuldades, sendo que a principal foi inexistência de documentação atual sobre como configurar os serviços necessários. O facto de a cada lançamento de uma nova versão do OpenStack parte da sua API alterar leva a que a bibliografia que se encontrava estivesse desatualizada e não fosse funcional para a versão utilizada (Newton). Outro problema encontrado foi nos recursos disponíveis: os serviços do OpenStack são pesados, pelo que estes demoravam alguns minutos a iniciar nos nossos controladores, chegando a falhar às vezes por excederem o tempo de timeout. Este problema surgiu no final da configuração do serviço Neutron, tendo esta situação sido contornada quando se aumentou a memória RAM dos controladores. Outro problema encontrado foi na implementação do Cluster MariaDB. Isto aconteceu porque a sua alta disponibilidade é do tipo Ativo-Passivo, sendo que se o nó principal falhar e um secundário assumir o papel de principal, é necessário restaurar a base de dados do que era anteriormente o principal, pois a sincronização é feita em cadeia.

Capítulo 5

Nós de Computação

Ao longo do capitulo 4 exploramos a arquitetura dos componentes e demonstramos como instalar e configurar os controladores do OpenStack para a versão Newton. No entanto os controladores só por si não chegam, pelo que é necessário adicionarem-se nós de computação à infraestrutura para que esta tenha recursos disponíveis para os seus utilizadores usufruírem. Não existe limite de nós de computação no OpenStack [88], ou seja desde que existam recursos para se adicionar, podemos fazê-lo. No entanto devemos ter em atenção à versão que estamos a utilizar em caso de se adicionar um nó posterior à instalação inicial, uma vez que pode ter sido lançada uma nova versão desde que este tinha sido instalado.

Os passos de instalação e configuração de seguida apresentados devem ser executados sempre que se pretenda adicionar um nó de computação à infraestrutura anteriormente configurada.

5.1 Implementação

A configuração de um nó de computação, contrariamente à configuração de um controlador, é bastante simples uma vez que para este funcionar corretamente apenas necessita que seja instalado e configurado o Nova, o Neutron e, opcionalmente, o Ceilometer.

Inicialmente deverá ser adicionado o repositório da RDO (linha 1) bem como instalado os softwares necessários para que este nó comunique com os controladores (linha 2). Para tal deverá ser instalado o software que se apresenta abaixo. No bloco [4-6] garantimos que o serviço do openvswitch será iniciado e adicionamos uma bridge do tipo ovs, que será necessária para o Neutron. Ao longo desta instalação o termo *ADDRESS* deverá ser substituído pelo endereço IP do respetivo nó de computação.

```
1 yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
2 yum install -y openstack-nova-compute openstack-utils python-cinder
    openstack-neutron-openvswitch openstack-ceilometer-compute openstack-neutron
    openstack-selinux
```

```
3
4 systemctl enable openvswitch
5 systemctl start openvswitch
6 ovs-vsctl add-br br-int
```

O próximo passo consiste configurar os serviços no nó de computação, para que este saiba quem são os controladores que possuem os serviços necessários. Neste caso os serviços a configurar serão apenas o Nova, o Neutron e o Ceilometer. O bloco [1-34] contém a configuração referente ao Nova: neste são indicados quais os servidores de memcache (linha 1); o bloco [2-5] contém o endereço e porta do VNC e o endereço do controlador, para que seja permitido acesso remoto à consola de cada instância; o bloco [5-6] contém o endereço de conexão à base de dados bem como o número máximo de tentativas e como é feita a autenticação (KeyStone); na linha 6 é indicado o endereço IP do Glance. Na linha 11 é indicado o driver que será utilizado para a virtualização e na linha 15 são indicados os controladores que contém o serviço do RabbitMQ; no bloco [16-28] são indicadas as configurações de rede para que o Nova consiga comunicar com o Neutron, como endereço do Neutron, username, password, nome do projeto e região.

A configuração do Neutron surge no bloco [36-55]. Aqui são indicados os parâmetros que tinham sido anteriormente configurados nos nós controladores como tipo de estratégia utilizada para autenticação, bem como os dados de autenticação (bloco [36-42]); dados dos controladores que contém o RabbitMQ bem como o driver que é utilizado para as notificações (bloco [43-45]); no bloco [46-55] são indicados os parâmetros da configuração do OpenvSwitch, como o tipo de Vlan utilizada (vxlan), a bridge de integração e a de túnel (a bridge de tunel é a que permite ligação entre os controladores e os nós de computação e a bridge de integração é a que faz a comunicação com o tunel e o controlador ou com o nó de computação) - é possível visualizar como é feita a troca de pacotes na rede na figura 5.1) e o driver de firewall utilizado.

No bloco [57-74] surge a configuração do Ceilometer em que para este são indicados os parâmetros que tinham sido anteriormente configurados nos nós controladores como tipo de estratégia utilizada para autenticação, bem como os dados de autenticação (bloco [57-62]), os servidores de memcache (linha 63), na linha 64 são indicados os controladores que contém o serviço do RabbitMQ, no bloco [66-70] são definidos os dados de autenticação do ceilometer para a telemetria, no bloco [71-74] é indicado o endereço para a conexão com a base de dados MongoDB do Ceilometer e o número máximo de tentativas de autenticação.

No bloco [76-80] são definidos os parâmetros TCP para o Keepalive, isto é: na linha 77 e 78 os parâmetros são expressos em segundos e significa que as rotinas do Keepalive esperam cinco segundos e são reenviadas a cada segundo que passa. Se não for recebido nenhum ACK (liinha 79) passado outros cinco segundos a ligação é dada como falhada e é mudado de controlador de keepalive.

```
1 openstack-config --set /etc/nova/nova.conf DEFAULT memcached_servers
      HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
2 openstack-config --set /etc/nova/nova.conf vnc vncserver_proxyclient_address ADDRESS
```

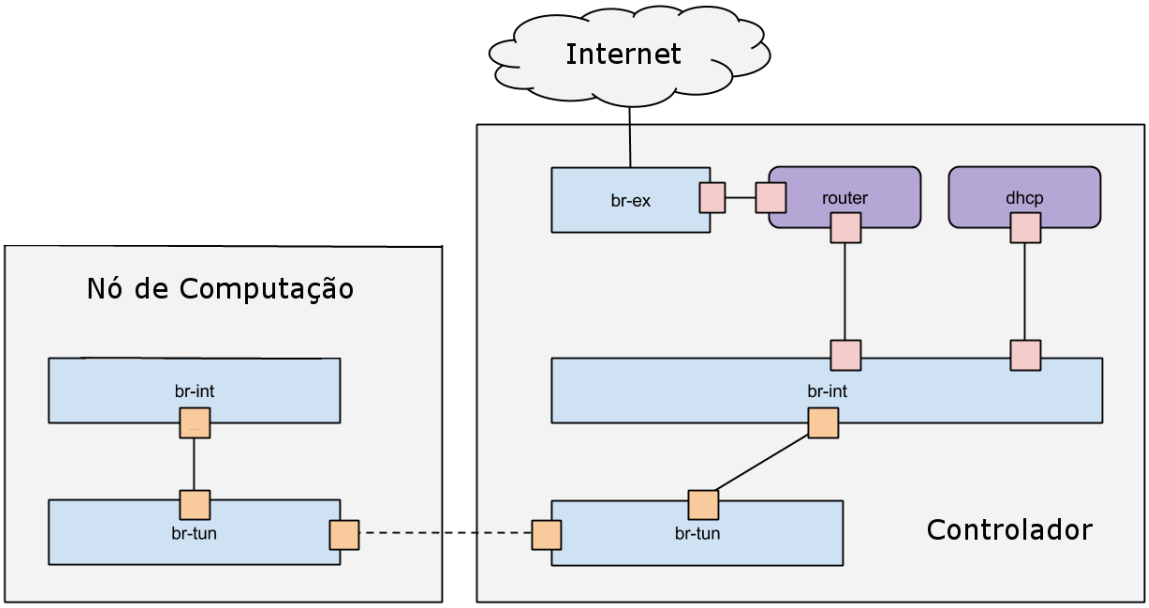


Figura 5.1: Representação das Bridges no Neutron entre o nó de computação e o controlador. Imagem adaptada de [16].

```
3 openstack-config --set /etc/nova/nova.conf vnc vncserver_listen 0.0.0.0
4 openstack-config --set /etc/nova/nova.conf vnc novncproxy_base_url
      http://192.168.110.10:6080/vnc_auto.html
5 openstack-config --set /etc/nova/nova.conf database connection
      mysql://nova:novatest@192.168.110.10/nova
6 openstack-config --set /etc/nova/nova.conf database max_retries -1
7 openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
8 openstack-config --set /etc/nova/nova.conf glance host 192.168.110.10
9 openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
      nova.network.neutronv2.api.API
10 openstack-config --set /etc/nova/nova.conf DEFAULT firewall_driver
      nova.virt.firewall.NoopFirewallDriver
11 openstack-config --set /etc/nova/nova.conf libvirt vif_driver
      nova.virt.libvirt.vif.LibvirtGenericVIFDriver
12 openstack-config --set /etc/nova/nova.conf DEFAULT security_group_api neutron
13 openstack-config --set /etc/nova/nova.conf cinder cinder_catalog_info
      volume:cinder:internalURL
14 openstack-config --set /etc/nova/nova.conf conductor use_local false
15 openstack-config --set /etc/nova/nova.conf oslo_messaging_rabbit rabbit_hosts
      HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
16 openstack-config --set /etc/nova/nova.conf oslo_messaging_rabbit rabbit_ha_queues True
17 openstack-config --set /etc/nova/nova.conf neutron service_metadata_proxy True
18 openstack-config --set /etc/nova/nova.conf neutron metadata_proxy_shared_secret metatest
19 openstack-config --set /etc/nova/nova.conf neutron url http://192.168.110.10:9696/
```

```
20 openstack-config --set /etc/nova/nova.conf neutron project_domain_id default
21 openstack-config --set /etc/nova/nova.conf neutron project_name services
22 openstack-config --set /etc/nova/nova.conf neutron user_domain_id default
23 openstack-config --set /etc/nova/nova.conf neutron username neutron
24 openstack-config --set /etc/nova/nova.conf neutron password neutrontest
25 openstack-config --set /etc/nova/nova.conf neutron auth_url http://192.168.110.10:35357/
26 openstack-config --set /etc/nova/nova.conf neutron auth_uri http://192.168.110.10:5000/
27 openstack-config --set /etc/nova/nova.conf neutron auth_plugin password
28 openstack-config --set /etc/nova/nova.conf neutron region_name regionOne
29 openstack-config --set /etc/nova/nova.conf libvirt nfs_mount_options v3
30 openstack-config --set /etc/nova/nova.conf DEFAULT transport_url
       rabbit://rabbit@HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
31 openstack-config --set /etc/nova/nova.conf DEFAULT my_ip ADDRESS
32 openstack-config --set /etc/nova/nova.conf DEFAULT use_neutron True
33 openstack-config --set /etc/nova/nova.conf vnc enabled True
34 openstack-config --set /etc/nova/nova.conf glance api_servers http://192.168.110.10:9292
35
36 openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
37 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_uri
       http://192.168.110.10:5000/
38 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_plugin password
39 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_url
       http://192.168.110.10:35357/
40 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken username neutron
41 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken password neutrontest
42 openstack-config --set /etc/neutron/neutron.conf keystone_authtoken project_name services
43 openstack-config --set /etc/neutron/neutron.conf oslo_messaging_rabbit rabbit_hosts
       HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
44 openstack-config --set /etc/neutron/neutron.conf oslo_messaging_rabbit rabbit_ha_queues
       true
45 openstack-config --set /etc/neutron/neutron.conf DEFAULT notification_driver
       neutron.openstack.common.notifier.rpc_notifier
46 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent tunnel_types
       vxlan
47 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent
       vxlan_udp_port 4789
48 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs enable_tunneling
       True
49 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs tunnel_id_ranges
       1:1000
50 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs
       tenant_network_type vxlan
51 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs integration_bridge
       br-int
52 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs tunnel_bridge
       br-tun
53 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs local_ip
```

```
    ADDRESS
54 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini securitygroup
    firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
55 openstack-config --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent l2_population
    False
56
57 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_uri
    http://192.168.110.10:5000/
58 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_plugin
    password
59 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_url
    http://192.168.110.10:35357/
60 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken username
    ceilometer
61 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken password
    ceilometertest
62 openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken project_name
    services
63 openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT memcache_servers
    HA-CONTROLLER1:11211,HA-CONTROLLER2:11211,HA-CONTROLLER3:11211
64 openstack-config --set /etc/ceilometer/ceilometer.conf oslo_messaging_rabbit rabbit_hosts
    HA-CONTROLLER1,HA-CONTROLLER2,HA-CONTROLLER3
65 openstack-config --set /etc/ceilometer/ceilometer.conf oslo_messaging_rabbit
    rabbit_ha_queues true
66 openstack-config --set /etc/ceilometer/ceilometer.conf publisher telemetry_secret
    ceilometersecret
67 openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials os_auth_url
    http://192.168.110.10:5000/v2.0
68 openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials os_username
    ceilometer
69 openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials os_tenant_name
    services
70 openstack-config --set /etc/ceilometer/ceilometer.conf service_credentials os_password
    ceilometertest
71 openstack-config --set /etc/ceilometer/ceilometer.conf
72 database connection mongodb://HA-CONTROLLER1,HA-CONTROLLER
73 2,HA-CONTROLLER3:27017/ceilometer?replicaSet=ceilometer
74 openstack-config --set /etc/ceilometer/ceilometer.conf database connection max_retries -1
75
76 cat > /etc/sysctl.d/tcpka.conf << EOF
77 net.ipv4.tcp_keepalive_time = 5
78 net.ipv4.tcp_keepalive_intvl = 1
79 net.ipv4.tcp_keepalive_probes = 5
80 EOF
```

Para finalizar apenas nos falta iniciar e garantir que os serviços instalados são iniciados automaticamente no arranque do sistema do nó de computação:

```
systemctl start libvirtd
systemctl start neutron-openvswitch-agent
systemctl enable neutron-openvswitch-agent
systemctl enable neutron-ovs-cleanup
systemctl start openstack-ceilometer-compute
systemctl enable openstack-ceilometer-compute
systemctl start openstack-nova-compute
systemctl enable openstack-nova-compute
```

Posto isto temos o nosso nó de computação totalmente configurado e já disponível para ser utilizado.

5.2 Dificuldades Encontradas

Na implementação do nó de computação não foi sentida nenhuma dificuldade uma vez que apenas foi necessário seguir a lógica já utilizada anteriormente para configuração dos controladores.

Capítulo 6

Conclusão

Com a conclusão deste projeto propusemos a adoção de um sistema de Cloud Computing do tipo privado, o que possibilitará um melhor aproveitamento dos recursos do INESC TEC, sejam estes tecnológicos (re-aproveitamento dos recursos existentes), humanos (para a manutenção dos recursos basta um administrador de sistemas,sendo que no aprovisionamento dos recursos este funciona de forma self-service) ou financeiros (deixa de existir a necessidade de se adquirir recursos tecnológicos a cada novo projeto de investigação).

Devido à sua enorme comunidade de desenvolvimento associada, o OpenStack revelou-se uma escolha acertada graças à sua estabilidade, modularidade, compatibilidade e funcionalidade.

O facto do OpenStack ter sido implementado em máquinas de teste seguindo todos os requisitos exigidos para este projeto, fez com que a implementação final nos servidores destinados para produção e investigação do INESC TEC tivesse concluída sem problemas graves, uma vez que as falhas e dificuldades de instalação tinham sido superadas e tendo sido construída a documentação de implementação, a instalação tornou-se simples e sem surpresas.

Assim sendo, conseguimos concluir praticamente todas as metas propostas para esta tese, nomeadamente a instalação e configuração da versão em vigor mais recente do OpenStack em alta disponibilidade e com tolerância a falhas podendo assim salientar o facto de esta ser uma infraestrutura Confiável, bem como a construção de documentação associada, conseguindo assim descomplicar-se o processo de instalação e configuração em alta disponibilidade, uma vez que a documentação existente ao nosso alcance era praticamente nula, sendo que apenas não foi implementada a Cloud híbrida que ambicionávamos, ficando esta assim para trabalho a realizar no futuro.

Através deste projeto consegui ter um contacto mais intenso com a minha área de eleição: administração de sistemas e gestão de redes. Este contacto permitiu a aquisição de conhecimento e experiência em matérias das quais nunca tinha tido oportunidade de trabalhar, como Cloud Computing, redundância e alta disponibilidade que, sem margem para dúvidas, serão importantes para a minha carreira profissional nesta área.

6.1 Trabalho Futuro

Estima-se que para um futuro próximo o objetivo passe por interligar a nossa Cloud privada com uma Cloud do tipo público, como por exemplo a AWS. Para tal, pretende-se utilizar os *plugins* já desenvolvidos pelo projeto OpenStack para tal função.

Gostaríamos também de desenvolver uma interface web, mais simplista que a padrão do OpenStack (Horizon Dashboard), uma vez que esta pode ser vista como complexa para um investigador normal, dado que este só pretende lançar, iniciar e parar uma instância, não necessitando do outro elevado número de funções disponíveis no Horizon Dashboard.

Bibliografia

[1] Tomislav Bronzin. Business intelligence vnext or how cloud computing is (not) changing the way we do bi, nov 2016.

[2] Microsoft. Microsoft trust center, apr 2015.

[3] Amazon. Infraestrutura global da aws, nov 2016.

[4] OpenNebula. Which functionality does opennebula provide?, oct 2016.

[5] OpenNebula. Why opennebula?, oct 2016.

[6] Wikipedia. Eucalyptus, oct 2016.

[7] Hewlett-Packard. Hp, oct 2016.

[8] ShapeBlue. Cloudstack 101, nov 2016.

[9] RedHat. Open cloud infrastructure built for the enterprise, apr 2017.

[10] RedHat. Keepalived scheduling, jun 2017.

[11] MongoDB. Mongodb, jun 2017.

[12] Mirantis. What is this keystone anyway?, jun 2017.

[13] OpenStack. Cinder system architecture, jun 2017.

[14] OpenStack. Networking architecture, jun 2017.

[15] RedHat. Architecture guide, jun 2017.

[16] RDO PROJECT. Networking in too much detail, jun 2017.

[17] Rackspace Cloud Computing. Openstack, oct 2016.

[18] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[19] Amazon. O que é o docker?, nov 2016.

[20] Microsoft Azure. Computação de alto desempenho, dec 2016.

[21] Microsoft Azure. Container registry, dec 2016.

[22] Microsoft Azure. Servico de aplicacoes, dec 2016.

[23] Microsoft Azure. Storage, dec 2016.

[24] Rodney A DeKoning, Gerald J Fredin, and Donald R Humlicek. Apparatus and method to provide virtual solid state disk in cache memory in a storage controller, May 20 2003. US Patent 6,567,889.

[25] Microsoft Azure. Base de dados sql, oct 2016.

[26] Microsoft Azure. O que é o azure, oct 2016.

[27] ISO/IEC. Iso/iec 27018:2014, oct 2016.

[28] Amazon. O que é a aws?, nov 2016.

[29] Amazon. Cloud compute with aws, nov 2016.

[30] Amazon. Amazon ec2 container service, nov 2016.

[31] Amazon. Amazon ec2 container registry, nov 2016.

[32] Amazon. Aws lambda, nov 2016.

[33] Amazon. Amazon s3, oct 2016.

[34] Amazon. Amazon cloudfront – rede de entrega de conteúdo (cdn), nov 2016.

[35] Amazon. Amazon elastic block store (ebs), oct 2016.

[36] Amazon. Amazon relational database service (rds), nov 2016.

[37] Amazon. Amazon dynamodb, nov 2016.

[38] Amazon. Amazon elasticache, nov 2016.

[39] Bogdan George Tudorica and Cristian Bucur. A comparison between several nosql databases with comments and notes. In *2011 RoEduNet International Conference 10th Edition: Networking in Education and Research*, pages 1–5. IEEE, 2011.

[40] Amazon. Amazon virtual private cloud (vpc), nov 2016.

[41] Amazon. Aws direct connect, nov 2016.

[42] Amazon. Elastic load balancing, nov 2016.

[43] William Stallings. *Cryptography and network security: principles and practices*. Pearson Education India, 2006.

[44] Amazon. Amazon route 53, nov 2016.

[45] Google. Google cloudplataform, nov 2016.

[46] Kubernetes. Kubernetes, dec 2016.

[47] Google. Storage classes, dec 2016.

[48] Google. Cloud sql, dec 2016.

[49] Google. Seriously powerful data analytics services, nov 2016.

[50] Google. Cloud virtual network, nov 2016.

[51] Google. Google cloud platform security, nov 2016.

[52] Google. Encryption at rest in google cloud platform, nov 2016.

[53] Google. Google cloud platform security, nov 2016.

[54] Rafael Moreno-Vozmediano, Rubén S Montero, and Ignacio M Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45 (12):65–72, 2012.

[55] OpenNebula. Red hat + centos, oct 2016.

[56] Inc. Eucalyptus Systems. Eucalyptus, oct 2016.

[57] Amazon. Amazon web services, oct 2016.

[58] Hewlett-Packard. Hp acquires eucalyptus, sep 2014.

[59] Apache CloudStack. Apache cloudstack^{TM}, nov 2016.

[60] Apache CloudStack. Apache cloudstack faq, nov 2016.

[61] Apache CloudStack. Apache cloudstack about, nov 2016.

[62] Apache CloudStack. Cloudstack cloudmonkey cli, nov 2016.

[63] Ceph. Ceph, nov 2016.

[64] Apache CloudStack. Apache cloudstack features, nov 2016.

[65] Apache. Apache security, nov 2016.

[66] Apache CloudStack. Apache cloudstack security, nov 2016.

[67] Rackspace Cloud Computing. Openstack, nov 2016.

[68] Search Server Virtualization. What is hypervisor?, nov 2016.

[69] Microsoft. Virtualization, nov 2016.

[70] cloudscaling. Openstack's future depends on embracing amazon. now., jun 2017.

[71] NASA. Nebula, nasa, and openstack, mar 2017.

[72] Apache. Apache license, fev 2017.

[73] OpenStack. Newton, mar 2017.

[74] Percona. Bootstraping the cluster, apr 2017.

[75] Jay Pires. [openstack-dev] [nova][neutron][mysql], jun 2017.

[76] OpenStack. Openstack docs, jun 2017.

[77] haproxy. Haproxy, jun 2017.

[78] RDO PROJECT. Rdo, mar 2017.

[79] Keepalived. Keepalived for linux, jun 2017.

[80] MariaDB. What is mariadb galera cluster?, jun 2017.

[81] MariaDB. Mariadb galera, jun 2017.

[82] RabbitMQ. Rabbitmq, jun 2017.

[83] RabbitMQ. Reliability guide, jun 2017.

[84] MemCached. Memcached, jun 2017.

[85] MongoDB. Mongodb, jun 2017.

[86] RedHat. Chapter 2. openstack networking concepts, jun 2017.

[87] RedHat. 3.3. openstack database-as-a-service (trove), jun 2017.

[88] Red Hat. Deployment limits for red hat openstack platform, jun 2017.