Dissertação de Mestrado em

Engenharia Informática - Computação Móvel

Uma implementação opensource de um serviço de cloud do tipo IaaS

João Vitória Silva

Leiria, setembro de 2017

Dissertação de Mestrado em

Engenharia Informática - Computação Móvel

Uma implementação opensource de um serviço de cloud do tipo IaaS

João Vitória Silva

Dissertação de Mestrado realizada sob a orientação do Doutor Mário João Gonçalves Antunes, Professor Adjunto da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, setembro de 2017

Agradecimentos

Chegado a este momento que marca mais um passo na minha vida académica, é difícil quantificar todas as pessoas que tiveram uma papel mais ou menos fundamental na conclusão desta etapa.
Gostaria de começar por agradecer aos meus pais, pois foram eles que me deram a possibilidade, a força e o apoio necessário para finalizar esta etapa. Sei que demonstrarão este mesmo apoio e me continuarão a dar força nas diversas oportunidades que a vida me trará.
À Diana, por me dar força naquelas horas mais tardias em que a motivação e a força eram escassas.
Por fim a todos os amigos, docentes, não docentes ou conhecidos que tiveram um papel nesta etapa, quer tenha sido ele principal ou secundário.
Obrigado!

Esta página foi intencionalmente deixada em branco.

Resumo

A Internet é uma das infraestruturas à escala global de tecnologias de informação e comunicação, usada para pesquisa, trabalho, alojamento de conteúdos e visualização de conteúdos multimédia, tornando-se assim num dos principais focos tanto no mundo empresarial, como para o utilizador comum.

A forma como interagimos com todos os serviços disponíveis na Internet, e a própria forma como estes estão estruturados, está em constante mudança e reinvenção. Podemos observar esta evolução no modo como comunicamos pela Internet ao longo dos anos. Começou pela troca de emails de texto simples, estando atualmente num ponto em que há comunicação por videochamadas através do Skype, comunicação contínua pelo Facebook, publicação de fotos pelo Instagram, ou então conversação por fotos como acontece com o SnapChat. A cada vinte minutos no Facebook são partilhados um milhão de links, enviados dois milhões de pedidos de amizade e são enviadas três milhões de mensagens[1]. Cada vez há mais serviços deste tipo e estes mesmos serviços são cada vez mais utilizados no quotidiano dos utilizadores, gerando um aumento significativo de tráfego na rede, passando de 3.4 *Zettabyte* (ZB) em 2014 para 10.4 ZB em 2019 segundo dados da Cisco[2].

Este constante aumento de tráfego, de aplicações e de serviços implica remodelações constantes nas infraestruturas dos *Internet Service Provider* (ISP), de modo a que estas tenham capacidade de suportar o aumento constante do mesmo, apresentando soluções escaláveis, redundantes e seguras.

Atualmente, já existem várias soluções de várias companhias a oferecer e a implementar medidas que prometem colmatar estes desafios. Estas soluções, baseadas em tecnologias *cloud* em diversos *Data Center* (DC)s geograficamente distintos, não seguem uma linha única de desenvolvimento, oferecendo assim soluções proprietárias e comerciais dispares. Esta realidade criou uma incompatibilidade nos vários serviços que constituem a *cloud*, tornando assim o processo de integração e interoperabilidade entre as mesmas um desafio.

Com o objetivo de superar este desafio surgem várias soluções baseadas em tecnologias *opensource* e *standard*, tendo como objetivo oferecer uma solução *cloud* flexível, interope-

rável e dinâmica capaz de responder às necessidades do mercado e garantir a compatibilidade entre os mais diversos serviços que constituem a *cloud*.

Nesta dissertação, o objetivo é apresentar uma solução privada *cloud Infrastructure as a Service* (IaaS) que responda a este desafio, usando tecnologias totalmente *opensource* e *standard* de modo a provar a sua interoperabilidade e compará-la com uma solução proprietária e comercial, aplicando-a ao atual paradigma das pequenas e médias empresas portuguesas.

Após a apresentação da arquitetura testada para a implementação da plataforma privada *cloud* IaaS e posteriormente à comparação efetuada com uma solução equivalente proprietária e comercial, concluímos que apesar das potencialidades do uso de tecnologias *standard* e *opensource*, esta não é por vezes a escolha adequada para cenários com recursos humanos e tecnológicos limitados. Esta é uma realidade comum nas pequenas e médias empresas portuguesas e uma solução deste tipo traz um aumento de complexidade, quer de implementação quer de manutenção, incomportável para este prisma.

Palavras-chave: DataCenter, Cloud, IaaS, Virtualização, opensource

Esta página foi intencionalmente deixada em branco.

Abstract

The Internet is one of the global infrastructures of information and communication technologies, used for research, work, content hosting and content viewing thus becoming one of the main focal points in the business world, as well as for the average user.
The way we interact with all the services available on the Internet, and the way these are structured, are constantly changing and in reinvention. We can observe developments in the way we communicate over the Internet over the years. It started with exchange of plain text emails, currently being at a point where there is communication using video calls via Skype, continuous communication via Facebook, photos by Instagram, or photo conversation using SnapChat. Every twenty minutes on Facebook are shared a million links, sent two million friendship requests and sent three million messages[1]. More and more services of this type are increasing and being used in the daily lives of users, generating a significant increase of traffic in the network, going from 3.4 ZB in 2010 to 10.4 ZB in 2019 according to data from Cisco[2].
This constant increase in traffic, applications and services implies a constant remodeling of the ISP infrastructures, so that they are able to withstand the constant increase of traffic, providing scalable, redundant and secure solutions.
Currently there are several solutions from several companies that offer and implement measures which promise to meet these challenges. These solutions, based on cloud technologies in several geographically distinct DCs, do not follow a single line of development, thus offering each one a separate proprietary and commercial solution. This reality has created an incompatibility in the various services that constitute the cloud, making the process of integration and interoperability between them a challenge.
In order to overcome this challenge, a number of solutions based on open source and standard technologies are emerging, aiming to offer a flexible, interoperable and dynamic cloud solution capable of responding to the needs of the market and ensuring compatibility between the most diverse services that make up the cloud.
In this master thesis, the goal is to present a private cloud IaaS that responds to this challenge,

using completely opensource and standard technologies in order to prove its interoperability and to compare it with a proprietary and commercial solution, applying the current paradigm of Portuguese small and medium enterprises.
After the presentation of the tested architecture for the implementation of the private IaaS cloud platform and after the comparison made with a proprietary equivalent solution, we conclude that despite the potential of using standard and opensource techonologies, sometimes a solution of this type is not the appropriate choice for scenarios with limited human and techonologies resources. This is a common reality in Portuguese small and medium enterprises and a solution of this kind brings an increase in complexity, either of implementation or maintenance, unbearable for this prism.
Keywords: DataCenter, Cloud, IaaS, Virtualização, opensource

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Acrónimos

ACL *Access Control List*

API *Application Programming Interface*

Data Center Centro de Processamento de Dados

DB *Database*

DC *Data Center*

Capex *Capital Expeditures*

CPU *Central Processing Unit*

DHCP *Dynamic Host Configuration Protocol*

DNS *Domain Name System*

DPO *Data Protection Officer*

DRP *Disaster Recovery Plan*

FC *Fibre Channel*

FC *Fibre Channel*

FCIP *Fibre Channel over IP*

FCP *Fibre Channel Protocol*

FCoE *Fibre Channel over Ethernet*

GDPR *General Data Protection Regulation*

GNS3 *Graphical Network Simulator-3*

HTTP *Hypertext Transfer Protocol*

IaaS *Infrastructure as a Service*

IEEE *Institute of Electrical and Electronics Engineers*

iFCP *Fibre Channel Protocol*

IP *internet Protocol*

iSCSI *Internet Small Computer System Interface*

ISP *Internet Service Provider*

IETF *Internet Engineering Task Force*

MAC *Media Access Control*

NAT *Network Address Translation*

NFV *Network function virtualization*

NTP *Network Time Protocol*

OvS *Open vSwitch*

OF *OpenFlow*

OS *Operating Systems*

Opex *Operation Expenses*

NIST *National Institute of Standards and Technology*

PaaS *Platform as a Service*

PUE *Power Usage Effectiveness*

RAM *Random Access Memory*

SaaS *Software as a Service*

SAN *Storage Area Network*

SCSI *Small Computer System Interface*

SDN *Software Defined Networking*

SLA *Service Level Agreement*

SPICE *Simple Protocol for Independent Computing Environments*

TI Tecnologias de Informação

UPS *Uninterruptible Power Supply*

URL *Uniform Resource Locator*

VLAN *Virtual Local Area Network*

VM *Virtual Machine*

VNC *Virtual Network Computing*

VPN *Virtual Private Network*

VXLAN *Virtual Extensible Local Area Network*

ZB *Zettabyte*

Esta página foi intencionalmente deixada em branco.

Conteúdo

Capítulo 1

Introdução

Esta dissertação de mestrado insere-se no âmbito da unidade curricular de Dissertação, incidindo sobre temáticas como *Data Center* (DC)s, *cloud* e virtualização. Nesta dissertação apresentam-se conceitos fundamentais sobre as tecnologias e temáticas que irão ser abordadas no decorrer na mesma, apresentando também, um estado da arte sobre soluções já existentes.
A metodologia utilizada foi a pesquisa bibliográfica, enriquecida com cenários práticos com o objetivo de provar resultados e conceitos.

1.1 Motivação

Com a constante evolução e reinvenção das aplicações e serviços usados na e para a Internet, novos desafios são apresentados, como por exemplo a interoperabilidade de todos os componentes, a escalabilidade de recursos automaticamente, entre outros. De forma a ser possível colmatar estes novos desafios, novas tecnologias são criadas ou aprimoradas.
Cada vez são utilizados mais serviços e/ou aplicações que usam a Internet como recurso, aumentando assim o tráfego de informação existente no núcleo da rede e nas infraestruturas onde todos esses serviços e/ou aplicações estão armazenados e a funcionar. Devido à quantidade elevada destes serviços e/ou aplicações, é necessário adaptar todos os recursos da infraestrutura em questão de modo otimizado, garantindo ao utilizador segurança, redundância, alta disponibilidade, flexibilidade e escalabilidade. Garantindo todos estes pontos, é obtida uma solução em que o serviço e/ou aplicação irá funcionar sempre de forma estável e fluída.
Para assegurar que estas infraestruturas são capazes de suportar uma implementação com

estas premissas, é necessário usar um determinado número de tecnologias. São necessárias tecnologias de e para a *cloud* (soluções de IaaS e SDN), a própria infraestrutura (DCs) e soluções de virtualização para garantir a flexibilidade e a escalabilidade (virtualização de rede, armazenamento e sistemas).
Atualmente já existem várias soluções deste género, como as soluções apresentadas pela Amazon (Amazon EWS), Microsoft (Microsoft Azure), Google (Google Cloud Platform), entre outras. Apesar de já existirem várias soluções, a maioria são proprietárias, causando incompatibilidade na interligação e migração entre as várias soluções devido ao uso de tecnologias proprietárias díspares em cada uma das soluções.
Devido a esta incompatibilidade, existe uma grande comunidade, incluindo várias empresas, com o objetivo de criar uma solução *standard* que possa ser adaptada e melhorada por todos, de modo a erradicar esta incompatibilidade. Esta visão está ainda em desenvolvimento, sendo ainda necessário desmistificar a premissa de que o software proprietário e comercial, ou com um custo mais elevado, é melhor. A motivação desta dissertação centra-se neste aspeto, em que o objetivo é provar, com recurso a testes e comparações, que uma solução de IaaS usando apenas tecnologias *opensource*[1] e *standard*, terá as mesmas funcionalidades e a mesma qualidade que uma solução proprietária.

1.2 Desafios

Para incidir neste tema nesta dissertação, é necessário efetuar toda uma recolha e implementação de tecnologias e soluções existentes. Ao efetuar esta recolha e implementação observou-se um determinado número de desafios devido à própria natureza e maturidade das mesmas:

- **Maturidade das tecnologias;**

  Algumas das tecnologias, ou conceitos, abordadas nesta dissertação, como o SDN, são recentes, obtendo assim vários significados ou a não identificação correta dos componentes utilizados por parte da comunidade.

- **Documentação;**

  Documentação escassa em algumas situações, dificultando assim a compreensão e implementação.

[1] *Opensource* é o termo dado ao código disponibilizado de forma aberta, que qualquer indivíduo ou entidade pode usar, visualizar e efetuar contribuições ao mesmo sem qualquer custo.

- **Interligação de tecnologias;**

  Ao ser implementada uma solução deste tipo, é necessário usar várias tecnologias, sendo estas tecnologias dependentes umas das outras. Caso não exista essa interligação, ou o suporte para tal, a implementação deixa de ser possível ou o nível de dificuldade aumenta de forma considerável.

1.3 Abordagem

Como ponto de partida para esta dissertação, começou-se por fazer todo um estudo dos conceitos fundamentais necessários para a compreensão dos temas desenvolvidos.
De seguida, foi efetuado o levantamento de soluções existentes para cada um dos conceitos fundamentais abordados antes, comparando e detalhando várias soluções dentro de cada conceito fundamental de modo a defender as soluções escolhidas.
De modo a ser obtido um cenário o mais próximo da realidade, é apresentada uma arquitetura onde são descritas as escolhas efetuadas e a sua implementação no cenário. Nesta arquitetura é descrita a implementação do serviço de IaaS, de SDN, entre outros.
Após a arquitetura estar descrita, é efetuada uma comparação com uma arquitetura proprietária, o Microsoft Azure Stack, incidindo em pontos como número de funcionalidades, qualidade, desempenho, etc.
Por fim, são apresentadas as conclusões ao trabalho realizado onde será possível avaliar se a motivação desta dissertação é aplicável ou não.

1.4 Objetivos

Os objetivos definidos para a realização do trabalho são as seguintes:

- Providenciar e criar mecanismos para a configuração da solução de IaaS OpenStack;

  Neste ponto irá ser demonstrado a configuração de um DC de modo a suportar a implementação de uma *cloud* privada recorrendo à solução de IaaS OpenStack. Para este efeito são usadas várias tecnologias de virtualização, nomeadamente o VirtualBox e o GNS3.

- Demonstrar viabilidade de uma solução privada de *cloud* em pequenas e médias empresas;

- Demonstrar a interoperabilidade da solução com outras ferramentas e soluções *open-source*, sendo estas extensões à atual solução ou integração com outras soluções de *cloud* existentes.

  Com este ponto pretende-se demonstrar a compatibilidade da solução com ferramentas de terceiros, quer seja para adição de funcionalidades, quer para demonstrar as vantagens do uso de *standards*. Aqui será implementado um controlador externo de rede, baseado em tecnologias *opensource*, para gerir toda a rede e encaminhamento virtual associado ao cenário.

- Demonstração de uma solução com foco em virtualização de DCs em cenários considerados privados;

  Demonstração de uma solução onde os dados de uma dada organização se situam na sua infraestrutura e não numa infraestrutura de terceiros, onde a organização terá controlo total sobre a sua informação, obtendo as vantagens de uma solução baseada em serviços *cloud*.

1.5 Estrutura do documento

Este documento encontra-se estruturado da seguinte forma:

- Capítulo 1, onde é feita uma introdução às motivações, aos desafios, à abordagem e aos objetivos desta dissertação de mestrado.
- Capítulo 2, onde são descritos os conceitos fundamentais para esta dissertação de mestrado. Esta descrição de conceitos tem como objetivo facilitar a compreensão de todos os processos descritos neste documento.
- Capítulo 3, onde é efetuado um levantamento de soluções existentes para cada tópico apresentado no capítulo 2.
- Capítulo 4, onde será simulado um ambiente real de rede descrevendo esse cenário, a sua implementação e os seus equipamentos.
- Capítulo 5, onde serão apresentadas comparações levadas a cabo para fundamentar a motivação desta dissertação.
- Capítulo 6, serão aqui apresentadas as conclusões levadas a cabo ao longo desta dissertação.

Capítulo 2

Conceitos fundamentais

Neste capítulo irão ser descritos os conceitos mais relevantes para o tema desta dissertação. Neste levantamento irá ser considerado o conceito em geral e os conceitos chave dentro do mesmo, considerando tecnologias, aplicações na industria, entre outros.

2.1 *Cloud Computing*

O termo *cloud* está cada vez mais presente no nosso quotidiano, quer a nível pessoal, quer a nível empresarial, mas na maioria das vezes este termo não é compreendido na totalidade.

A origem do termo *cloud computing* é um pouco incerta. O termo *cloud* é usado no meio científico para descrever um aglomerado de objetos do mesmo tipo que juntos formam um outro objeto com uma designação diferente[10].

O termo *cloud* como conhecemos é relativamente recente. O mesmo teve origem e designações diferentes ao longo do tempo. Ao longo da história da computação o termo foi evoluindo. Começou com a designação de *time sharing* na década de 1970. Esta designação, tal como indica o nome, significava que uma dada empresa alugava a sua infraestrutura a terceiros durante um período de tempo, sendo esse período partilhado pelos clientes.

Mais tarde, antes da década de 1990, os ISPs forneciam aos seus clientes ligação à Internet usando uma tecnologia diferente da de hoje. Hoje o tráfego existente na Internet é entregue aos clientes usando um mecanismo designado de *packet switching* ao contrário de antigamente em que era usado um mecanismo designado de *circuit switching*, consultar figura 2.1 e 2.2 respetivamente.

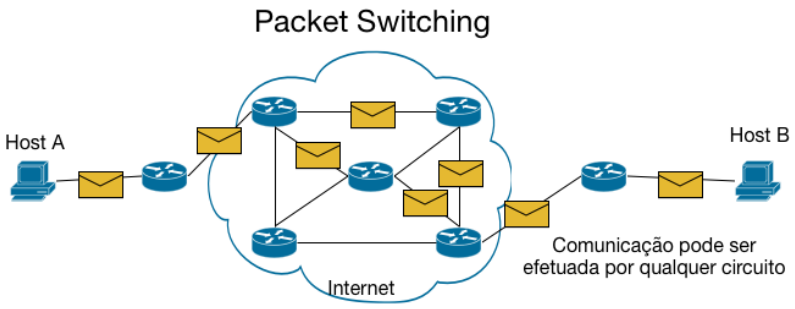

Figura 2.1: *Packet Switching*

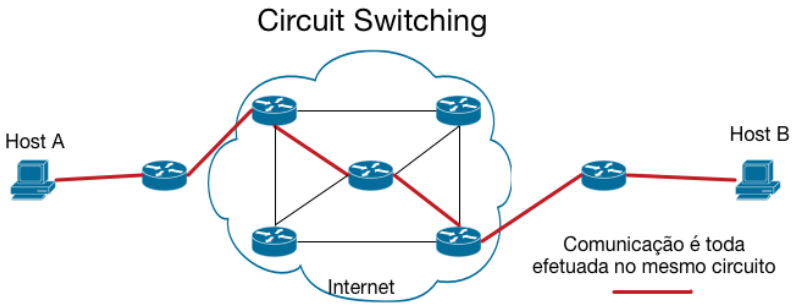

Figura 2.2: *Circuit Switching*

O mecanismo de *circuit switching* tinha algumas desvantagens como a necessidade de o tráfego ir apenas por um determinado circuito e, enquanto o circuito estivesse atribuído a uma ligação, o mesmo não poderia ser usado para outra ligação independentemente de o mesmo estar a trocar informação ou não[11].

Este mecanismo tornava-se pouco eficiente, havendo assim a necessidade de implementar um outro mais eficiente. Foi na década de 1990 que os ISPs introduziram o conceito de *Virtual Private Network* (VPN). Este conceito consistia em criar circuitos virtuais sobre os circuitos físicos existentes de forma a tornar a rede muito mais flexível e gerir a largura de banda existente de forma mais eficiente. Com o aparecimento desta nova tecnologia, os *providers* usaram pela primeira vez o símbolo da nuvem para designar um serviço ao cliente.

Só a partir do final da década de 2000 é que o termo *cloud computing* tomou o significado que hoje conhecemos[12][10][13].
O termo *cloud computing*, ou apenas *a cloud*, como hoje o conhecemos, é a disponibilização de serviços de forma ubíqua, rápida, simples e *on-demand* sobre a Internet de forma transparente para o utilizador.
Ao subscrevermos um serviço deste tipo, os dados são normalmente processados e guardados num local remoto fora do controlo da entidade cliente numa infraestrutura partilhada.

2.1.1 Características essenciais

De forma a podermos classificar um dado serviço como sendo de *Cloud Computing*, este terá de possuir cinco características essenciais[9]:

- ***On-demand self-service*;**

  Esta característica assenta na premissa de que um utilizador necessite de subscrever um determinado serviço, que este o possa fazer sem ser necessário entrar em contacto com outra pessoa. Esta subscrição encontra-se disponível *online* sendo a disponibilização feita quase sempre de imediato.

- ***Broad network access*;**

  Para podermos definir um serviço como pertencente ao *Cloud Computing*, este terá de ser acedido através de mecanismos comuns e bem conhecidos, como é o caso do *browser* ou *smartphone*.

- ***Resource pooling*;**

  Assenta na premissa de que os recursos dos serviços de *Cloud Computing* são partilhados por todos os clientes, sendo esses mesmos recursos alocados ou dealocados de forma dinâmica, tentando assim responder às necessidades dos clientes. Estes recursos são tanto físicos como virtuais. Ao existir esta dinâmica, o cliente não está ciente da localização ou controlo dos dados e aplicações que colocou na *cloud*.

  Se por sua vez um dito *provider* de serviços *cloud* disponibilizar ao cliente e apenas ao cliente certos recursos físicos, estes já não poderão ser considerados serviços de *cloud*, pois tal como foi referido para ser *cloud*, os recursos físicos são partilhados e apenas os recursos virtuais serão exclusivos para o cliente.

- ***Rapid elasticity*;**

O serviço e os recursos *cloud* são escaláveis. Caso seja necessário mais ou menos recursos, estes serão alocados dinamicamente, de forma automática e transparente para o utilizador. Aqui o objetivo é dar a ilusão ao utilizador de que os recursos são ilimitados.

- ***Measured service*.**

  Todos os sistemas e serviços em *cloud* são automaticamente controlados e otimizados pelo *provider*. Este controlo passa também pela quantidade de recursos exatos usados pelo utilizador. O mesmo pode ser útil em serviços em que se paga por recursos utilizados e não por mês, hora ou utilizador.

  Caso não seja este o tipo de controlo ou de utilização que o cliente pretenda ou queira contratualizar, o *provider* terá que disponibilizar esta opção.

2.1.2 Objetivos da *cloud*

Depois de ter sido definido todo o conceito de *cloud*, é necessário perceber realmente os grandes objetivos que a mesma disponibiliza e as suas vantagens, comparando-os com as soluções tradicionais já existentes. Oferecendo a *cloud* vários modelos, é necessário definir qual o melhor modelo a subscrever, por modo a rentabilizar da melhor forma possível o investimento. Posteriormente, deverá identificar-se se a mudança para a *cloud* irá satisfazer algum dos seguintes objetivos[9][14]:

- **Reduzir custos;**

  Uma das promessas da *cloud* é a redução de custos, mas será necessário avaliar se essa redução é significativa.

- **Obter flexibilidade de recursos;**

  Poderá ser importante para o tipo de negócio, obter um cenário em que os recursos são flexíveis e respondam da melhor forma aos requisitos dos clientes.

- **Aumentar a capacidade de armazenamento de dados;**

  Muitas empresas necessitam de uma capacidade elevada de armazenamento ou um constante aumento ou ajuste do mesmo devido ao uso de DBs e outros.

- **Evitar investimentos futuros em *hardware* ou *software*;**

  Ao subscrever um serviço de *cloud*, este poderá ser alterado e adaptado às necessidades do cliente, com ou sem a necessidade de investimento adicional.

- **Redundância dos dados e das ligações;**

  Os *providers* de *cloud* já contêm mecanismos de redundância, quer de dados, quer de ligações.

- **Melhorar o desempenho de acesso aos serviços do cliente;**

  Ao ter uma infraestrutura dedicada a este efeito, terá um melhor desempenho dos seus serviços;

- ***Backups* regulares;**

  Ao subscrever um serviço de *cloud* é garantido ao cliente que os seus dados são salvaguardados, sem preocupação ou custo adicional.

- **Gestão da infraestrutura;**

  Ao migrar os seus dados e as suas aplicações para a *cloud*, poderá focar-se apenas no desenvolvimento do seu serviço, não estando preocupado com a gestão da infraestrutura, o investimento e manutenção da mesma, poupando e otimizando o tempo para o que mais interessa.

Caso as necessidades do cliente se insiram em um ou mais destes tópicos gerais, então uma migração para a *cloud* será positivo.

2.1.3 Tipos de serviços

Ao falarmos de *cloud*, já se referiu que terá de responder a um determinado número de características e objetivos, mas dentro deste grande conceito, temos ainda outros. Estes outros conceitos são os que permitem definir o tipo de serviço que iremos obter da *cloud*, tendo estes finalidades e objetivos diferentes respondendo às necessidades de cada um. No total existem três principais tipos de serviços[9][14]:

- ***Software as a Service* (SaaS)**;

  Este serviço é o tipo de serviço que encontramos quando usamos por exemplo o Gmail[1], Hotmail[2] ou Dropbox[3]. Aqui o utilizador preocupa-se apenas em utilizar a dada aplicação e a gerir os aspetos dessa mesma aplicação. Todos os recursos que mantêm a aplicação não entram em contacto com o utilizador.

[1] O Gmail é o serviço de email da Google.

[2] À semelhança do Gmail, o Hotmail é também um serviço de email, mas da Microsoft.

[3] A Dropbox é um serviço de armazenamento online de ficheiros, podendo estes ser acedidos a partir de qualquer ponto.

- ***Platform as a Service* (PaaS)**;

  À semelhança do SaaS, aqui também a preocupação do utilizador é focada nas aplicações, mas ao invés de utilizar as mesmas, o *provider* da *cloud* disponibiliza ao utilizador todas as ferramentas para alojar, manter e oferecer a sua aplicação. O utilizador só terá de se preocupar com aspetos e configurações relacionadas com a sua aplicação. Mais uma vez, todos os recursos que mantêm a aplicação, não entram em contacto com o utilizador.

- ***Infrastructure as a Service* (IaaS)**.

  Neste serviço é dado ao utilizador acesso e controlo a todos os recursos computacionais de uma determinada máquina/infraestrutura, ou seja, aqui o utilizador também se preocupa com espaço em disco, memória, processador, *Operating Systems* (OS), etc e todo o software instalado e mantido nessa mesma máquina/infraestrutura.

Como foi referido, estes três são os principais tipos de serviços, no entanto no mundo da *cloud*, qualquer ferramenta pode ser considerada e colocada como um serviço. É possível também colocar DBaaS (Database as a Service), NaaS (Network as a Service), BaaS (Billing as a Service), entre outros.

2.1.4 Tipos de implementação

Como já definido anteriormente, *cloud* é o termo usado para a disponibilização de serviços com um determinado número de requisitos. No entanto, estes serviços podem ser prestados ao cliente de diversas formas. Destacam-se as seguintes[9][14]:

- ***Cloud* pública**;

  A *cloud* pública é o tipo de *cloud* mais comum. Aqui a utilização dos serviços é disponibilizada ao público, sendo detida, gerida e operada por uma entidade externa. Destacam-se os seguintes exemplos: Dropbox, Gmail e Hotmail.

- ***Cloud* privada**;

  Neste caso toda a infraestrutura de *cloud* é apenas usada pela empresa em questão, podendo a localização da mesma ser local ou num *Data Center* externo, gerida pela empresa, entidade externa ou uma combinação de ambas.

- ***Hybrid Cloud***;

Como o nome indica, trata-se de uma combinação de ambos os tipos de *cloud* descritos anteriormente. É uma combinação de duas ou mais infraestruturas mantendo a separação, sendo possível a portabilidade de dados ou aplicações quando necessário.

2.1.5 Benefícios da *cloud*

Na secção (2.1.4), foi descrito como a migração para a *cloud* pode trazer vários benefícios, tais como a redução de custos, de recursos humanos e de tempo. Quantificando:
Recentes estudos[15][16] demonstram que uma empresa movendo o seu negócio para a *cloud*, torna-se mais flexível, reduzindo custos ao nível das Tecnologias de Informação (TI), permitindo aos colaboradores acesso imediato à informação e aos serviços da empresa a partir de qualquer ponto, permitindo à empresa em questão focar-se apenas no seu negócio e não em problemáticas a nível das TI.
No mundo empresarial, o gestor terá de se preocupar com duas medidas importantes: o Opex e o Capex. O Opex é o custo operacional recorrente, ou seja, o custo subjacente para manter tal atividade e o Capex é o investimento inicial.
No caso do Opex é necessário ter em conta:

- Custo energético;
- Custo associado à manutenção dos sistemas AVAC[4];
- Licenças de *software*;
- Custo a nível de recursos humanos para manter a infraestrutura;
- Investimento em equipamentos adicionais, mas necessários para o correto funcionamento e sua manutenção;
- Tempo.

De seguida irá ser efetuada uma pequena comparação de custos ao nível do Opex e do Capex ao escolher uma solução *cloud* e uma local. Esta comparação tem em conta um custo aproximando em 36 meses, ou seja, três anos.
* *Uninterruptible Power Supply* (UPS)[6]

[4]AVAC constitui as tecnologias destinadas ao conforto ambiental interior, incluindo o aquecimento, ventilação e ar condicionado.

[6]UPS são sistemas que fornecem energia quando existe falha de energia na fonte principal.

Capex	Local	*Cloud*
Hardware (Server 8GB RAM, 2 CPU, 2TB)	2000€	0€
OS (Windows Server 2012 Standard)	700€	0€
Cabos	20€	0€
Apoio técnico de instalação (8h)	160€	0€
Total Capex	**2880€**	**0€**

Tabela 2.1: Custo Capex Local vs *Cloud*[9]

Capex	Local	*Cloud*
Custo elétrico servidor (400W)	1080€	0€
Custo elétrico AVAC (*Power Usage Effectiveness* (PUE)=1,75[5])	810€	0€
Custo AVAC	270€	0€
Sistemas *Uninterruptible Power Supply* (UPS)*	195€	0€
Apoio técnico à infraestrutura	5760€	0€
Ligação à Internet (Contrato com ISP)	1800€	0€
Subscrição de serviço *cloud*	0€	9176€
Total Opex	**12 795€**	**9176€**

Tabela 2.2: Custo Opex Local vs *Cloud*[9]

Observando as duas tabelas anteriores, tabela 2.1 e 2.2, podemos concluir que subscrever um serviço *cloud*, revela custos económicos inferiores a longo prazo, tendo ainda as vantagens descritas no sub capitulo inferior (2.1.4).

2.1.6 A problemática das *clouds* proprietárias

Existem várias empresas a disponibilizar serviços *cloud*. Cada uma destas empresas usa tecnologia proprietária ao nível das *Application Programming Interface* (API)s[7] para disponibilizar os seus serviços aos seus clientes. O facto de todas estas empresas disponibilizarem os mesmos tipos de serviço, mas usando tecnologias e implementações diferentes torna a comutação entre os mesmos dispendiosa, quer a nível de tempo, quer a nível monetário.
A necessidade de trocar os seus serviços de *provider*, pode incluir vários aspetos: um *provi-*

[7]Uma API trata-se de uma biblioteca/conjunto de código que fornece ao *developer* as ferramentas base para implementar um determinado conjunto de funcionalidades.

der disponibilizar os mesmos serviços, mas com um valor inferior, disponibilizar os mesmos serviços com mais funcionalidades, melhor qualidade de serviço, entre outros. No entanto a transição pode não ser simples. Quanto mais complexa for a aplicação/serviço, mais integração irá ter com a API do respetivo *provider*, resultando em mais alterações a serem efetuadas caso seja necessário mudar de *provider*, o qual terá uma API diferente com mecanismos diferentes.
Ao considerar uma mudança de *provider*, é necessário ter em conta várias considerações:

- Impacto desta mudança para os utilizadores da aplicação/serviço;
- Tempo necessário para efetuar a transição;
- Custos relativos a essa mudança desde:
  - Custo do novo *provider* relativamente ao antigo;
  - Custo ao nível dos recursos humanos para efetuar a transição.

Consequentemente, a escolha do *provider* é bastante importante, quer para o presente, quer para o futuro.

Cloud Broker

Os *cloud brokers* são uma das formas de lidar com o problema em questão. Ao falarmos de *cloud brokers*, podemos estar a referir-nos a dois tipos de serviços diferentes, mas com um objetivo comum, simplificar a escolha de *cloud provider* e a transição para a *cloud*. Existem dois tipos de *cloud brokers*[17]:

- Atuar como intermediário:
  - Explorar todas as necessidades do cliente;
  - Efetuar toda a pesquisa necessária para a escolha do *provider*;
  - Apresentar a melhor oferta no mercado com base nas necessidades do cliente, contando com as necessidades presentes e com projeções para necessidades futuras;
  - Poderá ser ou não o *cloud broker* a efetuar toda a parte de negociação com os *providers*.

  - Poderá, também o *cloud broker*, fornecer ao cliente serviços para efetuar toda a migração para a *cloud*, contando com mecanismos de encriptação e outros para garantir a confidencialidade dos dados.

- Fornecer ao cliente uma API intermediária:

  - Aqui o cliente não terá de se preocupar com o *provider* em si, apenas com o *cloud broker*;
  - Como na situação anterior, o *cloud broker* também irá explorar as necessidades do cliente e escolher o *provider* que melhor se enquadra nas necessidades;
  - O cliente irá desenvolver a sua aplicação/serviço com base na API do *cloud broker*;
  - Caso as necessidades do cliente alterem ou surja um *provider* com condições mais vantajosas, o *cloud broker* migra todo o conteúdo necessário para o novo *provider*;
  - Este processo é transparente para o cliente, pois o mesmo utiliza a API do *cloud broker* e não a API do *provider*.

Com esta solução, é garantida à empresa em questão, a escolha do melhor *provider* face às suas necessidades e caso o *cloud broker* o permita, um serviço intermediário que diminui o impacto ao comutar entre os diferentes *providers*. No entanto, a problemática mantêm-se, estando camuflada.

IaaS *open-source*

Já existem soluções para contornar o facto de cada um dos *providers* de *cloud* disponibilizar uma API própria.
Apesar de ser possível contornar o problema, é necessário compreender o problema em grande escala e tentar resolver o mesmo e não apenas camuflá-lo. Aqui entram os IaaS *opensource*.
Estes IaaS *opensource* têm como objetivo fornecer um conjunto de serviços ou aplicações para criar e gerir toda uma plataforma de *cloud computing*.
Uma empresa que opte por implementar um serviço *opensource*, obtém uma série de vantagens sobre o *closedsource*[8]. Devido a esta filosofia *opensource*, qualquer indivíduo ou

[8] *Closedsource* é o termo dado ao código proprietário de um dado indivíduo ou entidade, tendo como consequência o facto do mesmo ser fechado e não acessível a terceiros.

entidade pode adaptar a ferramenta em questão às suas necessidades ou adicionar funcionalidades de forma a tornar este projeto mais forte no mercado.
Em suma, ao invés de cada empresa criar a sua solução de *cloud computing*, cada uma *closedsource* e com APIs diferentes, o objetivo é criar uma ferramenta comum a todos, um *standard* para toda esta temática do *cloud computing*, eliminando este constrangimento.

2.2 *Data Centers*

Um DC é uma infraestrutura que engloba vários serviços num espaço físico próprio com gestão centralizada. Nesta infraestrutura estão incluídos todos os mecanismos necessários para o correto funcionamento dos mais variados serviços necessários para as empresas, desde toda a rede IP[9], serviços de rede como *Domain Name System* (DNS)[10], *Dynamic Host Configuration Protocol* (DHCP)[11] e outros, redes *Storage Area Network* (SAN), *Operating Systems* (OS) e os mais variados serviços, desde *file servers*, *messaging servers*, entre outros.
Toda esta infraestrutura conta com mecanismos de redundância e escalabilidade para garantir a total segurança dos dados existentes e garantir a continuidade, integridade e recuperação de dados em caso de catástrofe, sendo a causa da mesma natural ou atos maliciosos por parte de terceiros.[21]

2.2.1 Objetivos

O grande objetivo de um DC é criar uma infraestrutura com todos os aspetos necessários para suportar os serviços de TI de uma empresa. Dentro deste grande objetivo existem vários requisitos que um DC deverá cumprir e oferecer aos seus clientes. Um DC fornece aos seus clientes:

- **Suporte 24 sobre 7;**
- **Diminuição do custo associado à manutenção das TI;**

  Uma empresa que possuir uma infraestrutura própria terá um Opex elevado, visto que um dos objetivos dos DCs é baixar o Opex de uma dada empresa.

[9] O IP é um dos principais protocolos de comunicação na Internet.[18]
[10] O DNS é um sistema hierárquico distribuído que efetua a tradução de um domínio para um endereço IP.[19]
[11] O DHCP é o protocolo utilizado para atribuir dinamicamente parâmetros de rede a outras máquinas, como endereços IP, DNS *servers*, entre outros.[20]

- **Maior flexibilidade na criação e disponibilização de novos serviços;**

  Um DC, ao ter nas suas instalações técnicos especializados, torna a criação e/ou disponibilização de novos serviços mais flexível e eficiente, quer na instalação de *hardware* adicional ou suporte no existente.

- **Maior segurança dos dados devido às políticas de segurança exigidas numa infraestrutura deste tipo;**

  Um DC conta com políticas de segurança ao nível da segurança física, controlando a entrada e saída de todos os recursos humanos, cópia de segurança de todos os dados, entre outros.

- **Maior desempenho das máquinas;**

  Com controlo da temperatura ambiente e humidade, capacidade elétrica constante e redundante, proteção contra fogo e fumos e toda a cablagem necessária, é esperado o melhor desempenho possível.

- **Alta disponibilidade.**

  Redundância de hardware, quer de poder computacional, sistemas AVAC, energia elétrica, armazenamento, entre outros.

Em suma o objetivo de um DC, é fornecer uma solução à combinação de problemas e/ou limitações de TI existentes numa empresa de forma estruturada, flexível e *future proof*.[21]

2.2.2 Classificação

Com o conceito de DC e os seus objetivos definidos anteriormente, origina a definição da classificação dos vários tipos de DC existentes.
Geralmente, os objetivos são os mesmos, mas a forma como o serviço é disponibilizado ao cliente é que varia. Existem quatro tipos de DCs definidos em quatro *tiers*, sendo o *tier 1* o mais baixo e o *tier 4* o mais alto, o *tier* 4 mais robusto e menos suscetível a falhas.
Estes *tiers* são usados como referência na qualidade e no tipo de serviços que os mesmos oferecem. É de notar que todos os serviços disponíveis no *tier* 1 irão estar disponíveis no *tier* 2 e assim por diante. Por forma a um DC poder indicar que corresponde a um determinado *tier*, este terá de passar por uma certificação. Esta certificação pode ser efetuada por duas entidades, a ANSI/TIA (American National Standards Institute/Telecommunications Industry Association) ou o Uptime Institute. O Uptime Institute é uma entidade focada nesta

atividade, enquanto que a ANSI/TIA é um conjunto de entidades que cria *standards*, sendo que o *standard* ANSI/TIA-942 é o qual pelo que os DCs terão que passar. As características dos *tiers* são as seguintes[22]:

- *Tier* 1:
  - Ligações, equipamentos e *uplink* não redundantes;
  - 99.671% de disponibilidade.
- *Tier* 2:
  - *Tier* 1 + ligações e equipamentos redundantes;
  - 99.749% de disponibilidade.
- *Tier* 3:
  - *Tier* 1 + *Tier* 2 + redundância a nível energético nos equipamentos e nos *uplinks*;
  - 99.982% de disponibilidade.
- *Tier* 4:
  - *Tier* 1 + *Tier* 2 + *Tier* 3 + todos os componentes são redundantes a nível energético e a nível de falhas incluindo sistemas AVAC e outros;
  - 99.995% de disponibilidade.

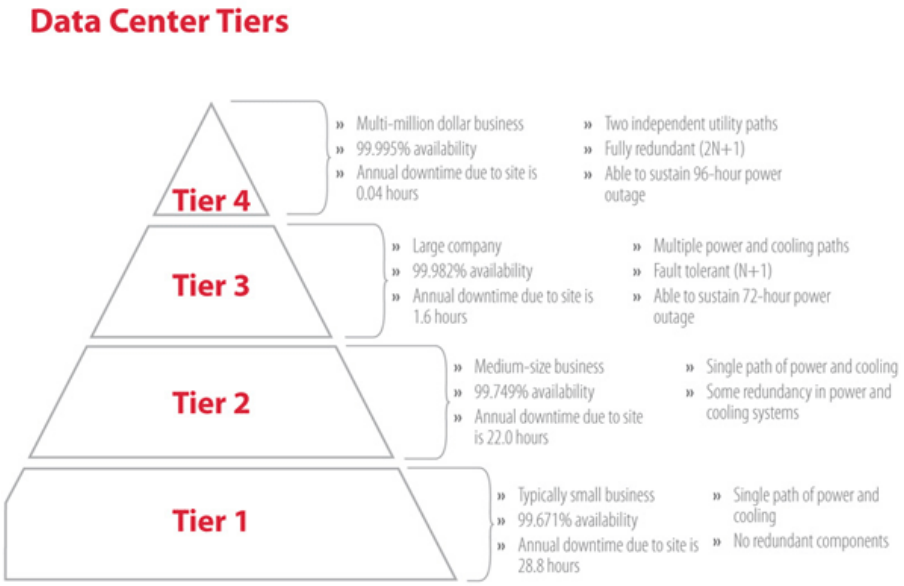

Figura 2.3: *Data Center Tiers*[3]

2.2.3 *Green Data Centers*

Um dos objetivos e requisitos de um DC, é fornecer energia elétrica, sistemas de controlo de humidade, temperatura, etc, no entanto estes serviços são dispendiosos ao nível de Opex e Capex. Devido às características destes serviços e também o impacto subjacente ao usar as mesmas no ambiente, cada vez mais é necessário ter em conta o impacto que estas infraestruturas podem ter no ambiente.
Esta iniciativa de um DC ser "amigo"do ambiente passa por implementar várias soluções, como por exemplo:

- **Minimizar o impacto visual e ambiental dos edifícios;**

  Em muito dos casos, os DC encontram-se situados em locais remotos devido às suas características ambientais, como a temperatura. Devido a essa localização remota, é necessário ter em conta o impacto da construção desta infraestrutura, visualmente, na flora e fauna envolventes.

  Também é necessário ter em conta o tipo de materiais usados na sua construção, devido às emissões e resíduos que possam existir.

- **Reciclar;**

  Como em qualquer outro edifício, neste também irá ser produzido lixo. É necessário ter em conta a separação do mesmo por forma a ser possível reciclar.

- **Filtrar todos os resíduos expelidos;**

  No caso dos sistemas AVAC ou sistemas de geradores de *backup* a combustíveis fosseis, é necessário colocar filtros que filtrem todas as impurezas expelidas.

- **Usar energias alternativas;**

  Ao invés de utilizar geradores e sistemas que usem combustíveis fosseis, devem usar-se soluções alternativas, como painéis solares, energia eólica, entre outros.

- **Usar meios de transporte elétricos;**

  Dentro do edifício e deslocações para o mesmo por parte de funcionários, é aconselhado o uso de veículos elétricos e/ou híbridos de modo a reduzir as emissões de CO2 para a atmosfera.

- **Usar equipamentos o mais energicamente eficientes possível.**

Qualquer um destes pontos é fundamental e importante para contribuir para um planeta mais sustentável, mas o mais importante será o último. É o que influencia muitos outros, como por exemplo o uso de mais ou menos energia elétrica para efetuar as mesmas operações, o facto da libertação de menos ou mais calor, que irá influenciar a necessidade de uso mais intensivo ou não dos sistemas AVAC que por sua vez irão influenciar o uso da energia elétrica.
Devido a toda esta problemática dos *Green Data Centers*, na escolha de um a utilizar, é tido em conta a sua eficiência energética, sendo esta calculada a partir da seguinte formula:
$PuE = \frac{Total-de-energia-usada-pelo-Data-Center}{Energia-dos-equipamentos-TI-usada-pelo-Data-Center}$
Atualmente, trata-se de um fator bastante importante na avaliação e decisão de um DC.

2.2.4 *Data Centers* e a *cloud*

Na secção 2.1 todos os dados ao serem disponibilizados numa *cloud* gerida por terceiros, ficam fora do controlo da entidade cliente e passam a partilhar toda uma infraestrutura. Esta infraestrutura é denominada de DC e a utilização do mesmo pode ser comparada à utilização da rede elétrica existente. No caso da rede elétrica, a mesma é usada de igual forma por todos, ou seja, a eletricidade chega a nós numa rede comum, mas não sabemos onde foi

gerada ou por onde passou. No caso dos DCs, podemos também aplicar esta analogia. Ao subscrevermos um serviço de *cloud*, sabemos que a informação estará alojada e a ser processada algures num DC, mas não sabemos onde nem quando, pois o mesmo ajusta-se de forma a responder às ditas necessidades e a informação pode estar disponível em DCs diferentes simultaneamente para responder às necessidades dos utilizadores da melhor forma possível. Também à semelhança da rede elétrica, o DC é uma infraestrutura comum a todos os seus utilizadores, partilhando os recursos da mesma.

Apesar de todas as vantagens referidas na secção 2.1.5 sobre mover os dados para uma *cloud*, pode não ser do interesse da empresa em questão assim o fazer. No entanto, pode ser do interesse da empresa ter todas as vantagens da *cloud* na sua implementação local.

Recordando, um DC é uma infraestrutura com um determinado número de requisitos e objetivos, possibilitando assim a qualquer empresa construir o seu próprio DC, mesmo que não seja com as mesmas dimensões ou com a mesma capacidade computacional e outros.

Após todos estes conceitos, conseguimos perceber a ligação entre os DCs e o *cloud computing*, os DCs disponibilizam uma infraestrutura física capaz de suportar toda a infraestrutura virtual proveniente da *cloud*.

2.3 Virtualização

Depois de ter sido definido o conceito de *Cloud Computing*, um *Data Center* e a sua interligação, é necessário definir também um último conceito, a virtualização.

A virtualização pode ser dividida, neste caso, em três áreas, virtualização de sistemas, de armazenamento e de rede (consultar figura 2.4).

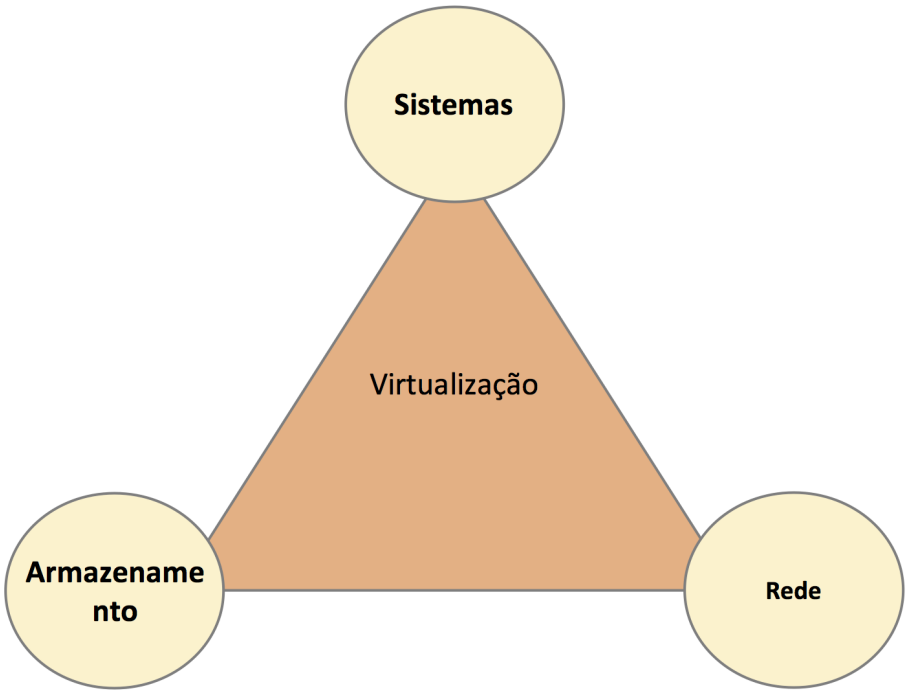

Figura 2.4: Triângulo Virtualização

2.3.1 Conceitos fundamentais

Tal como o nome indica, virtualização significa simular algo de carácter físico em ambiente não físico. Na área das TI, virtualizar significa aplicar algo que estaria em máquinas físicas e efetuar exatamente as mesmas operações, mas em ambiente virtual.
Na área das TI, este processo pode ser aplicado a pequenas coisas como criar discos virtuais para armazenamento, em que por exemplo num disco físico, criamos subdiscos (partições de disco), ou virtualizar uma *máquina* colocando o OS, os seus programas e todas as funcionalidades em ambiente virtual.
Esta tecnologia tem várias vantagens, como:

- Separação de tarefas por várias VMs;
- Caso haja algum problema numa das VMs, este não irá afetar as restantes;
- Caso haja uma aplicação ou serviço incompatível com o *Host OS*[12], é possível executá-la na mesma máquina física, mas numa VM com um OS diferente, ou seja, um

[12] *Host OS* é o OS instalado na máquina física.

Guest OS[13];

- Duplicar ou criar uma nova VM com base noutra já existente é simples, sem ser necessário voltar a instalar OS e as devidas aplicações ou serviços, tornando assim o *deploy* muito mais rápido;
- Conteúdo da VM fica isolado das restantes.

Contudo, é necessário ter algumas considerações se se optar por usar uma tecnologia como esta:

- É necessário ter uma máquina física que cumpra os seguintes requisitos:
  - *Central Processing Unit* (CPU)[14] com suporte para virtualização;
  - Memória *Random Access Memory* (RAM)[15] suficiente para o *Host OS* e para o *Guest OS*;
  - Espaço em disco para *Guest OS* e todas as suas aplicações e serviços.
- Abstração do utilizador a nível do físico e do virtual.

Devido à flexibilidade que esta tecnologia fornece, torna-se uma das tecnologias mais importantes no *Cloud Computing*, permitindo otimizar os recursos físicos de uma determinada infraestrutura e as suas aplicações de forma a permitir a utilização dos mesmos recursos físicos por vários utilizadores de forma eficiente e separada.

2.3.2 Técnicas de virtualização de sistemas

Os OSs como hoje os conhecemos, são desenhados e pensados para serem instalados e terem o melhor desempenho possível em máquinas físicas. No entanto, cada vez é mais comum o termo virtualização e em cada vez mais situações este é aplicável.
Ao usarmos um OS com uma arquitetura moderna, a arquitetura destes está desenhada para oferecer quatro tipos de privilégios aos OSs e às suas aplicações. Estes privilégios são conhecidos como Ring 0, 1, 2 e 3. Sendo o nível 0 o privilégio mais elevado, o OS espera receber este tipo de privilégio, tendo assim acesso aos recursos físicos da máquina, como

[13] *Guest OS* é o OS virtualizado e que corre sobre o *Host OS*.

[14] CPU é o componente do computador que executa todas as instruções necessárias do mesmo. É referido como o cérebro do computador.[23]

[15] RAM é um tipo de memória volátil que efetua a ponte entre todos os outros componentes do computador e o CPU. O CPU apenas consegue ler instruções que estejam na memória RAM.[24]

a memória. Se as aplicações instaladas naquela OS tem privilégios do tipo de Ring 3 e o *software* de virtualização, sendo também uma aplicação, como será possível dar privilégios do tipo de Ring 0 aos *guest OSs* a correr?
Para resolver esta questão são introduzidas três soluções[25]:

- *Paravirtualized Hypervisor*[16];
- *Full virtualized Hypervisor*;
- *Hybrid Hypervisors*.

Paravirtualized Hypervisor

Uma das soluções encontradas para solucionar esta questão, consiste em modificar o *kernel*[17] do OS, ajustando o mesmo para esta nova arquitetura. Ao efetuar esta alteração no *kernel*, o *guest OS* estará "ciente"desta virtualização e irá comunicar diretamente com a camada virtual criada pelo *hypervisor* existente para este fim.
Esta solução tem vários benefícios no desempenho da VM devido à inexistência da necessidade de traduzir todo o código binário criado pelo *guest OS*. No entanto, são necessários conhecimentos elevados para alterar o *kernel* de um OS (i.e. conhecimentos sólidos da(s) linguagem(ns) de programação usada(s), de todos os processos subjacentes ao mesmo, desde o funcionamento do *hardware* em questão, entre outros), muitas das vezes dificultado pelas atualizações do mesmo, sendo também muitas das vezes impossível alterar o *kernel* do mesmo devido a ser um OS proprietário, como é o caso do Windows ou Mac OSX.

Full virtualized Hypervisor

Com esta solução, é criada toda uma camada virtual de forma a simular uma máquina física. Assim o *guest OS* irá ter um funcionamento normal, estando ao cargo do *hypervisor* efetuar toda a tradução do código binário gerado pela *kernel* do *guest OS*. Face à solução anterior, esta poderá ter uma performance inferior, devido à necessidade de efetuar toda a tradução do código binário. No entanto, com esta solução, será possível virtualizar OSs proprietários, como é o caso do Windows ou do Mac OSX.
Apesar desta perda de desempenho devido à necessidade da tradução do código binário, novas soluções estão a aparecer para minimizar este impacto. Os fabricantes de *hardware*,

[16] *Hypervisor* é o nome dado à aplicação que gere toda a virtualização, incluindo a criação e a manutenção.
[17] O *kernel* é o componente do OS que interliga todos os outros e faz a ponte entre o *hardware* e o *software*.

como a Intel e a AMD, na construção dos seus CPUs, estão cada vez mais cientes da necessidade da virtualização, disponibilizando assim novas ferramentas e técnicas para melhorar e simplificar as técnicas existentes de virtualização. Com a tecnologia Intel Virtualization Tecnology (VT-x) e a AMD-V, é possível criar um novo nível de privilégios inferior ao Ring 0, onde os *hypervisors* possam correr e executar as instruções necessárias sem ser necessário esta tradução.
Este tipo de virtualização também é utilizado em soluções em que não existe um *host OS*. Isto é possível colocando o *hypervisor* a correr diretamente sobre o hardware, sendo muitas vezes designados de *bare metal hypervisors*. Esta solução tem as vantagens descritas anteriormente, mas sem o *overhead* extra que existe usando um *host OS*, aumentando assim a performance.

Hybrid Hypervisors

Esta solução, como o próprio nome indica, é uma solução híbrida, juntando assim o melhor das anteriores para obter a melhor performance possível.

2.3.3 Técnicas de virtualização de armazenamento

Ao falarmos de virtualização no mundo da computação, normalmente este conceito é apenas associado à virtualização de sistemas, ou seja, virtualização de uma máquina ou um OS. Não obstante, há mais do que a virtualização de sistemas. É cada vez maior a necessidade de alugar armazenamento e ter possibilidades de acedê-lo de forma rápida e simples.
Como foi descrito na secção 2.1.1, existem vários requisitos para um serviço ser considerado de *cloud* e ao alugarmos armazenamento na *cloud*, este terá de seguir os mesmos requisitos, tal como acontece na virtualização de sistemas. Por forma a conseguir seguir e cumprir todos estes requisitos, é necessário virtualizar este armazenamento, tornando a sua gestão o mais dinâmica possível, à semelhança da virtualização de sistemas.
O desafio encontrado aqui não é virtualizar o armazenamento, mas sim aceder aos mesmos de forma simples e remotamente. No caso de virtualizar um OS não existe este problema, pois é um OS com as mesmas características de um outro qualquer, ou seja, tem um IP, um endereço *Media Access Control* (MAC)[18], entre outros, ou seja, todos os recursos necessários para comunicar com outras máquinas na rede. Ao virtualizar armazenamento, o mesmo não acontece. Não existe nenhum OS pelo meio para efetuar toda a gestão e interligação com o mesmo.

[18] Um endereço MAC é o identificador de uma interface de rede na camada física da pilha protocolar TCP/IP.

Na secção 2.2, um DC contém redes *Storage Area Network* (SAN), sendo estas relevantes para a virtualização de armazenamento. Uma rede SAN é uma área de rede específica apenas para dispositivos de armazenamento, contando com um ou mais *storage array*[19]. Esta diferenciação existe para salvaguardar os dados em caso de catástrofe.

2.3.4 Técnicas de virtualização de rede

O termo *Software Defined Networking* (SDN) é relativamente recente, surgindo pela primeira vez em 2008. Criado pelas universidades de UC Berkeley e Stanford[26], o SDN surge como um conjunto de novas tecnologias com o objetivo de encarar a configuração de uma rede a um nível mais alto, dissociando-o assim do *hardware*, o controlo da rede. Com esta mudança de paradigma, a gestão da rede é simplificada, convergindo assim para um controlo centralizado. Com a implementação de SDN em redes informáticas, perde-se a necessidade de programação local a cada peça de *hardware*, sendo toda essa programação movida para um controlador, onde estarão a correr no mesmo, várias aplicações e serviços de rede[26]. Consultar figura 2.5.

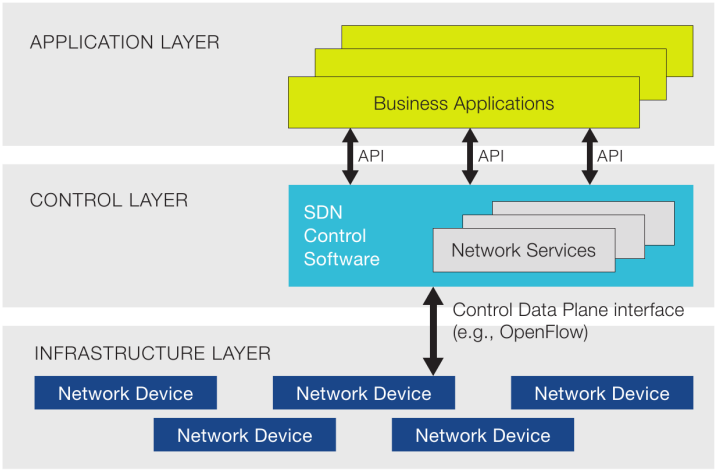

Figura 2.5: Arquitetura da tecnologia SDN[4]

A redução dos custos de manutenção de uma rede informática usando SDN e a necessidade constante de atualização do *hardware* são elementos de extrema importância para a temática

[19]Um *storage array* é o nome dado a um conjunto de dispositivos de armazenamento interligados entre si, possuindo capacidades de alta disponibilidade e redundância.

desenvolvida nesta dissertação, quer a nível da *cloud*, quer a nível de manutenção e gestão de toda a infraestrutura de um DC[26].

- **Data Center;**
  - Facilita a virtualização da rede devido à sua centralização;
  - Centro de dados fortemente escalável;
  - Melhor gestão da migração em tempo real de VMs;
  - Otimização do *hardware* de toda a rede do DC, permitindo assim uma redução de equipamentos, uma otimização de largura de banda e ainda redução do consumo energético dos equipamentos.
- **Cloud.**
  - Os recursos de toda a rede SDN poderão ser alocados de forma elástica;
  - Permite uma disponibilização mais rápida de serviços *cloud*.

Capítulo 3

Estado da arte

Neste capítulo irão ser aprofundados conceitos referidos no capítulo 2 de forma a fundamentar as decisões tomadas ao nível de tecnologias escolhidas e implementadas.

3.1 OpenStack

Relembrando a secção 2.1.6, existe a necessidade de criar um standard para toda a temática do *cloud computing*, surgindo assim o OpenStack como principal solução nas várias pesquisas efetuadas. O OpenStack é uma plataforma totalmente *open-source* de *cloud computing*, disponibilizando todas as ferramentas necessárias para configurar de raiz uma *cloud* pública ou privada.

Este projeto foi criado em 2010 em conjunto pela *National Aeronautics and Space Administration* (NASA)[1] e pela Rackspace Hosting[2], numa tentativa de superar o problema apresentado anteriormente. Atualmente, em 2017, trata-se de um projeto conceituado a nível mundial.

Todo o projeto é gerido pela OpenStack Foundation, uma organização sem fins lucrativos, com o objetivo de gerir tanto o desenvolvimento contínuo do projeto, como toda a comunidade à volta do mesmo, contando com o apoio de diversas empresas, como a Google, Cisco e IBM, e contribuições individuais por todo o globo.

O OpenStack contém várias ferramentas: criar e gerir instâncias de computação, gerir armazenamento, gerir todo o *networking* associado às VMs, entre outras. O OpenStack, ao disponibilizar ferramentas para criar todo um serviço *cloud*, disponibiliza também todas as

[1] A NASA é uma agência Norte Americana responsável pela pesquisa e desenvolvimento de tecnologias e programas de exploração espacial.

[2] Empresa Norte Americana conceituada mundialmente devido às suas soluções de *cloud computing*.

funcionalidades e características necessárias para ser considerado um serviço de *cloud computing* referidas na secção 2.1.1.
O OpenStack, por ser um conjunto de ferramentas para criar e manter todo um serviço de *cloud* sobre um *Data Center*, possibilita concluir que o OpenStack não se insere nos tipos de serviços de SaaS ou PaaS referidos anteriormente na secção 2.1.2, mas sim num serviço do tipo de IaaS[27][28][29].
Atualmente, encontra-se implementado em diferentes áreas, contando com grandes nomes como a PayPal, Yahoo, Cisco WebEx, CERN, American Express, Intel, entre muitos outros.

Propriedades do OpenStack

Sendo o OpenStack um serviço de IaaS, este terá de assegurar determinadas características de modo a ser considerado um serviço robusto. Estas características, ou propriedades do mesmo, são as seguintes:

- **Migração em tempo real;**

  Trata-se da capacidade de mover VMs de uma máquina física para outra de forma transparente enquanto a mesma se encontra a correr, migrando também toda a memória, configurações de rede e armazenamento associados à mesma.

  O OpenStack suporta dois tipos de migração em tempo real:

  - **Migração em tempo real baseado em armazenamento partilhado;**

    Suporta migração quando o *hypervisor* inicial e final necessitam de acesso ao armazenamento partilhado.

  - **Migração em tempo real baseado em bloco;**

    Suporta migração quando o *hypervisor* inicial e final não necessitam de acesso ao armazenamento partilhado.

- **Balanceamento de carga;**

  Trata-se da capacidade de dividir todo o tráfego por várias VMs de modo a conseguir obter o melhor desempenho. Este processo funcionará de forma transparente e em paralelo com o live migration.

  O OpenStack utiliza vários métodos para garantir esta funcionalidade, desde a utilização de migração em tempo real para distribuir a carga de forma uniforme e balanceamento de carga ao nível das VMs conseguido através do componente Neutron.

- **Tolerância a falhas;**

  Capacidade de todo o sistema ser tolerante a falhas, quer a nível de VMs, DBs ou máquinas físicas.

  O OpenStack pode efetuar tolerância a falhas usando os seguintes mecanismos:

  - Mecanismos de planeamento e *scripting* de forma a adaptar todas as VMs a esta falha, agrupamento de VMs do mesmo tipo em máquinas diferentes, mecanismos de configuração automática de novas VMs, entre outros;
  - Ao nível do armazenamento e de DBs, são usados mecanismos de replicação e sincronização.

- **Alta disponibilidade;**

  Usando todos os recursos mencionados anteriormente, ter a capacidade de garantir que existem sempre recursos disponíveis.

  É possível ter alta disponibilidade no OpenStack nos seus serviços *stateless*[3] e *statefull*[4]:

  - **Stateless;**

    Serviços como o Nova (computação), em que a alta disponibilidade é assegurada através de instâncias redundantes e balanceamento de carga entre as mesmas.

  - **Statefull.**

    Serviços como o Swift (armazenamento) ou DBs, a alta disponibilidade é assegurada através da replicação de dados e sincronização entre as várias replicações.

- **Segurança;**

  Garantir autenticação de todos os utilizadores no sistema de forma geral para obter o melhor controlo sobre os dados e a infraestrutura possíveis, confidencialidade e integridade dos dados e dos utilizadores usando VPNs e *firewalls*.

- **Compatibilidade.**

  Compatibilidade com outros serviços de *cloud computing*, como Amazon EC2 e S3, e a vários tipos de *hypervisors*, desde XenServer, KVM, etc.

[3] Serviços *stateless* são serviços que podem responder a pedidos sem ser necessário obter informação histórica ou de outros serviços.

[4] Serviços *statefull* são serviços que necessitam de informação de outros serviços ou informação histórica para que possam responder a pedidos.

3.2 CloudStack

À semelhança do OpenStack, o CloudStack é uma solução que segue a filosofia de criar um standard para a temática de *cloud computing*. Também à semelhança do OpenStack, o CloudStack segue uma política *open-source* suportado e mantido por uma comunidade, encontrando-se atualmente na versão 4.6.
Foi criado pela Cloud.com em maio de 2010 com cerca de 95% de todo o código *open-source* sobre a licença GPLv3[5][6]. Mais tarde em 2011, a empresa Citrix adquiriu a Cloud.com e continuou o desenvolvimento do projeto, tornando os restantes 5% de código *open-source*, também sob a licença GPLv3. Em 2012 a Citrix doou o projeto à Apache Software Foundation sobre a licença Apache Software License 2.0 (ASLv2)[7] e em 2013 tornou um *Top-Level Project* (TLP) por parte da mesma.
Atualmente não é um projeto com a dimensão do OpenStack, mas está a ganhar mais adeptos, devido à sua simplicidade.

Propriedades do CloudStack

À semelhança do OpenStack, o CloudStack também é um serviço de IaaS, tendo de assegurar certas características ou propriedades. Algumas destas são comuns ao OpenStack como:

- **Migração em tempo real;**

  A migração em tempo real no OpenStack poderá ter condições diferentes devido aos *hypervisors* utilizados, sendo assim importante a escolha do *hypervisor*. Por exemplo, ao usar o *hypervisor* KVM não é possível efetuar a migração se a VM em questão usar armazenamento local ou se a máquina inicial e a de destino estiverem em *clusters* diferentes. Ao usar os *hypervisors* XenServer ou VMWare estas condições já não existem.

- **Balanceamento de carga;**

  É possível balancear a carga de forma nativa nas máquinas físicas, mas caso seja necessário efetua-lá a nível virtual, será necessário usar *software* externo.

[5] GPLv3 é uma licença que define se um dado *software* é *copyleft*[30].

[6] *Copyleft* é o mesmo que *copyrighted*, mas ao invés de usar estes direitos por forma a restringir a um determinado número de pessoas, assegura que não há restrição a qualquer indivíduo e ao seu tipo de uso, ou seja, o utilizador tem total liberdade, desde que o produto resultante continue a seguir as mesmas bases.

[7] ASLv2 à semelhança do GPLv3 trata-se de uma licença para código *open-source*, mas no caso da GPLv3 se basearmos o nosso *software* em outro *software* que seja GPLv3, o código resultante também o será, sendo assim a entidade obrigada a disponibilizar o mesmo. No caso do ASLv2 esta obrigatoriedade não existe[31][32].

- **Tolerância a falhas;**

  O CloudStack, ao usar uma arquitetura centralizada através do Management Server, pode replicar este de forma a garantir continuidade caso um falhe. É possível serem efetuados também mecanismos semelhantes ao OpenStack.

- **Alta disponibilidade;**

  A alta disponibilidade no CloudStack é assegurada com mecanismos semelhantes ao do OpenStack.

- **Segurança;**

  Para além de suportar os mesmos mecanismos que o OpenStack, o CloudStack possibilita também o agrupamento e isolamento de tráfego usando grupos de segurança associados a grupos de VMs, sendo possível atribuir diferentes regras a cada um destes grupos.

- **Compatibilidade.**

O CloudStack ainda conta com outras duas propriedades:

- **Escalabilidade;**

  Capacidade de gerir grandes quantidades de *hosts* devido à estrutura hierárquica e aos *Management Servers*.

- **Extensão da plataforma a partir de APIs.**

  Possibilidade de criação de novos *plugins* e integração dos mesmos de forma nativa no CloudStack por parte dos *developers* por forma a estender as APIs existentes.

3.3 OpenStack vs CloudStack

Neste capítulo irá ser efetuada uma comparação entre o OpenStack e o CloudStack, focando pontos como:

- Comparação geral;
- Comparação funcional;
- Comparação das propriedades;

- Comparação da performance;
- Escolha do melhor serviço de IaaS.

3.3.1 Comparação geral

Ambas as soluções apresentam o mesmo modelo de negócio, com a mesma licença (gratuito com recurso à licença ASLv2), suportando os mesmos modelos de *cloud computing*. Ambos os serviços são baseados numa comunidade de utilizadores ou entidades, sendo todo o projeto gerido por uma organização, estando já inseridos em vários pontos no mercado, contando com o apoio de grandes empresas a nível mundial. Ao nível de implementação e arquitetura, existem diferenças consideráveis. O OpenStack tem uma arquitetura fragmentada, tendo como objetivo evitar barreiras na tecnologia, oferecer grande flexibilidade e extensibilidade. No entanto, esta arquitetura tem como consequência um aumento de complexidade na instalação e manutenção do serviço. O CloudStack por sua vez, ao fornecer uma arquitetura centralizada, torna a instalação e manutenção do serviço mais acessível, tendo como penalização uma inferior flexibilidade e extensibilidade no desenvolvimento do serviço. Consular tabela 3.1.

	OpenStack	***CloudStack***
Licença *open-source*	ASLv2	ASLv2
Tipos de IaaS	Público, Privado e Híbrido	Público, Privado e Híbrido
Instalação	Difícil (Muitos módulos e muita escolha)	Médio (Pouca instalação)
Arquitetura	Fragmentada	Controlador geral

Tabela 3.1: Comparação geral entre o CloudStack e o OpenStack

3.3.2 Comparação funcional

A nível da comparação funcional, podemos ter em conta os *hypervisors* disponíveis em cada plataforma, o tipo de gestão que cada um oferece, a comunidade, a frequência de atualizações, etc. Consultar tabela 3.2.

Ambos os serviços suportam os mesmos *hypervisors*, sendo possível suportar outros de forma não oficial. Ao nível da administração, é a maneira dos utilizadores administrarem todos os serviços relativos às plataformas, disponibilizando ambos uma interface Web e uma

Funcionalidade	OpenStack	*CloudStack*
Hypervisors open-source	Xen, KVM, Hyper-V, vSphere, LXC	Xen, KVM, Hyper-V vSphere, LXC
Administração	Interface Web e CLI	Interface WEB e CLI
Monitorização de utilizadores	Sim	Sim
Monitorização de recursos	Sim	Sim

Tabela 3.2: Comparação funcional entre o CloudStack e o OpenStack

interface ao nível da linha de comandos (CLI). Ambas as plataformas suportam a gestão e monitorização dos utilizadores e dos recursos.

Apesar de ser fundamental suportar todas as funcionalidades descritas em cima, é ainda mais fundamental se estas plataformas contiverem uma comunidade ativa devido à natureza *open-source* das mesmas, significando que têm um desenvolvimento ativo, atualizações regulares e uma boa documentação.

Ambas as plataformas são atualizadas regularmente, mas apenas a OpenStack tem uma política de datas específicas, ou seja, estas atualizações são realizadas duas vezes por ano simultaneamente com a distribuição de Linux Ubuntu, sendo uma das atualizações em abril e outra em setembro. O CloudStack apesar de não contar com uma política de atualizações com uma data bem definida, apresenta um desenvolvimento gradual e ativo.

Por último, é necessário avaliar a comunidade. Das duas, o OpenStack contém a maior comunidade com maior atividade de todos os projetos de IaaS de *cloud computing*. É de notar que a comunidade do CloudStack e a sua atividade no projeto está a crescer cada vez mais, estando a tornar-se um projeto de referência ao nível de *cloud computing* com cada vez mais participação no mercado. Ambos os projetos organizam eventos para juntar os seus *developers* e anunciar as novas funcionalidades.

Em suma, o OpenStack é o maior projeto de ambos, contando com uma comunidade maior e mais ativa, no entanto o CloudStack tem ganho importância nos últimos anos.

3.3.3 Comparação de propriedades

Outra comparação a ter em conta é a comparação de propriedades, considerando certas propriedades indispensáveis para um serviço do tipo de IaaS. Consultar tabela 3.3.

Ambos os serviços suportam as mesmas propriedades, sendo no entanto implementadas de formas distintas.

Propriedade	OpenStack	*CloudStack*
Migração em tempo real	Sim	Sim
Balanceamento de carga	Nível virtual e físico	Nível virtual e físico
Tolerância a falhas	Replicação de VMs e backup de dados	Replicação de VMs e backup de dados
Alta disponibilidade	Redundância e balanceamento de carga	Redundância e balanceamento de carga
Segurança	VPNs, *firewall*, user authentication, entre outros	VPNs, *firewall*, user authentication, entre outros
Compatibilidade	Amazon Services	Amazon Services

Tabela 3.3: Comparação de propriedades entre o CloudStack e o OpenStack

No caso do da migração em tempo real, o OpenStack suporta dois tipos, enquanto no CloudStack varia consoante o *hypervisor* utilizado.

Em balanceamento de carga ao nível físico, é efetuado usando mecanismos semelhantes em ambas as plataformas, usando migração em tempo real e replicação de VMs. No caso do nível virtual, ou seja, as VMs, no OpenStack todo este processo é garantido pelo componente Neutron, enquanto que no CloudStack não existe componente para tal, sendo assim necessário utilizar *software* externo para este fim.

Ao nível de tolerância a falhas, em ambos os serviços são configurados mecanismos de substituição de VMs ou serviços e replicação dos mesmos.

Para conseguir alta disponibilidade, são utilizadas várias instâncias do mesmo serviço de forma redundante e com balanceamento de carga por forma a garantir a continuidade caso uma das instâncias deixe de responder e garantir igual carga em todas as instâncias.

Ambas as plataformas suportam mecanismos de segurança por forma a garantir a mesma, confidencialidade e integridade dos dados. Estes mecanismos de segurança incluem VPNs, serviços de *firewall*, monitorização de utilizadores, entre outros.

Para tornar estas plataformas o mais dinâmicas possível, as duas plataformas suportam integração com as APIs da Amazon.

3.3.4 Comparação de desempenho

Após uma análise às funcionalidades e às propriedades dos dois serviços, é necessário uma comparação ao nível do desempenho. Para esta comparação têm-se por base o artigo *Bench-*

marking the Performance of OpenStack and CloudStack[33]. Neste artigo são avaliados os seguintes fatores:

- Tempo de criação e eliminação de VMs;
- Tempo de eliminação de VMs.

Na comparação destes dois fatores, são efetuadas as seguintes relações:

- Relação do CPU e RAM;
- Relação do tipo de armazenamento.

3.3.5 Conclusões

Considerando todas as comparações supra citadas, podemos observar que a nível de funcionalidades, ambas as plataformas estão equiparadas, no entanto ao nível do desempenho observa-se que a plataforma OpenStack é superior.
Em suma, se o desejado for utilizar uma plataforma com uma implementação simples, com fortes funcionalidades, mas que ao nível do desempenho seja inferior, o CloudStack é uma ótima plataforma. Caso o desejado seja o maior desempenho possível, mesmo que implique um maior conhecimento e um maior número de recursos, quer recursos humanos ou materiais, o OpenStack será uma melhor escolha.

3.4 Soluções de virtualização de sistemas existentes

Atualmente existem várias soluções no mercado, soluções *open-source* e outras proprietárias. A nível de soluções de virtualização de sistemas *open-source* temos:

- Oracle VirtualBox;
- KVM;
- XenServer;
- Entre outros.

Em soluções de virtualização de sistemas proprietárias existe:

- VMware vSphere (ESXi);

- VMware Fusion;
- VMware Workstation;
- VMware Player;
- Microsoft Hyper-V;
- Parallels Workstation;
- Parallels Desktop;
- Entre outros.

Como se pode observar, existem várias soluções, e dentro de cada empresa, a fornecer serviços deste tipo. Existem soluções para cada necessidade, desde soluções pessoais (Parallels Desktop, VMware Player e Oracle VirtualBox), a profissionais (VMware Fusion, VMware Workstation, Parallels Workstation e Microsoft Hyper-V) e ao nível de gestão de toda a infraestrutura de um DC (VMware vSphere, KVM, XenServer e Microsoft Hyper-V).

3.4.1 KVM vs XenServer vs VMware vSphere

Nesta dissertação irão ser consideradas apenas as soluções KVM, XenServer e VMware vSphere. KVM e XenServer, devido à flexibilidade que as mesmas oferecem ao serem soluções *open-source*, e o VMware vSphere de forma a oferecer uma comparação com um concorrente proprietário com uma grande quota de mercado e soluções fortes. Nestas três soluções irão ser considerados vários testes ao nível do desempenho[25].
Por forma a efetuar uma forte comparação, é usado como base o artigo *Evaluation of Different Hypervisors Performance in the Private Cloud with SIGAR Framework*[25], onde foram utilizadas várias ferramentas para aferir o desempenho de várias componentes dos *hypervisors*. Neste artigo são avaliados os seguintes fatores:

- Desempenho do CPU nos *hypervisors*;
- Desempenho da RAM nos *hypervisors*;
- Desempenho do armazanemanto nos *hypervisors*;
- Desempenho da comunicação em rede nos *hypervisors*.

Em suma, as soluções VMware vSphere e XenServer apresentam resultados muito próximos, estando assim equivalentes. Infelizmente, a solução KVM apresenta resultados significativamente inferiores, sendo assim uma escolha a não considerar quando o desempenho é um ponto chave.

Caso o objetivo seja uma solução *open-source*, como é o caso desta dissertação, o XenServer será uma ótima escolha, sendo o KVM uma alternativa.

3.5 Soluções de virtualização de armazenamento existentes

Como foi referido na secção 2.3.3, nos DCs existe uma separação física entre o armazenamento e o poder computacional, havendo assim a necessidade de existirem ferramentas/tecnologias que possibilitem a comunicação entre as mesmas.

Para possibilitar esta comunicação surgiram, entre outros, os seguintes protocolos:

- SCSI;
- FC.

3.5.1 *Small Computer System Interface* (SCSI)

O protocolo SCSI foi uma das primeiras soluções para esta problemática. Este protocolo foi desenhado e pensado para ser integrado em todas as soluções de redes SAN existentes para possibilitar toda a comunicação entre redes existentes com redes SAN *standard*. Desde que se tornou um *standard* em 1986, o SCSI já sofreu diversas iterações, sendo relevante para esta dissertação o iSCSI. O iSCSI assenta sobre o IP e tem como objetivo gerir todo o armazenamento remotamente a partir de qualquer ponto para qualquer ponto. Antes do iSCSI não era possível, devido a limitações da tecnologia ao nível da distância (25 metros)[34].

Atualmente qualquer OS moderno contem ferramentas para implementar uma solução deste género por parte do cliente. Qualquer máquina que deseje ligar-se a um disco virtual via iSCSI é chamado de iSCSI *Initiator* e o *storage array* desejado é chamado de iSCSI *Target*.

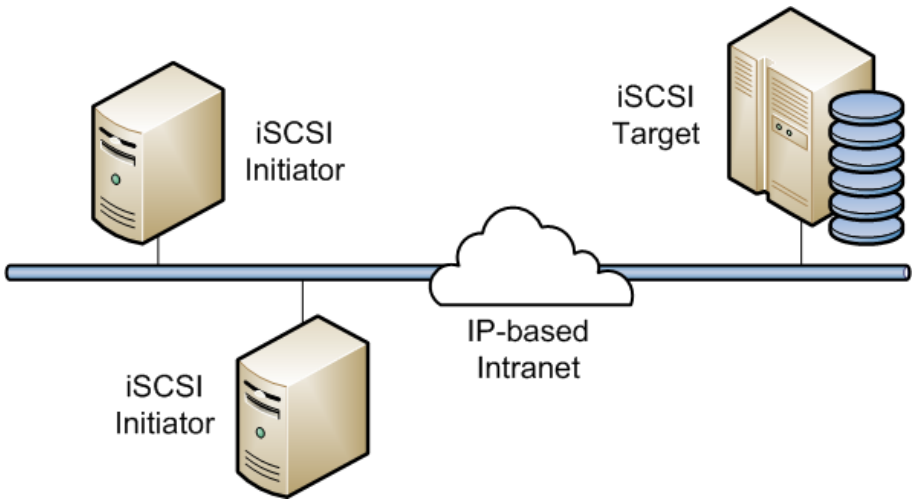

Figura 3.1: Funcionamento iSCSI[5]

De modo a ser possível efetuar esta ligação, o *storage array* terá de estar de alguma forma endereçado na rede. Este endereçamento é conseguido através da atribuição de um IP à *storage array*, sendo de seguida os discos virtuais endereçados com base neste IP. Este endereçamento pode ser efetuado das seguintes formas[35]:

- **Formato iqn.;**
  - Literal iqn;

    iSCSI *Qualified Name*
  - Data;

    Ano e mês (yyyy-mm) em que a dada empresa adquiriu o domínio.
  - Domínio da empresa representado de forma oposta;

    Se o domínio da empresa em questão for por exemplo example.com, irá ficar neste caso com.example.
  - Após o :, caminho para um disco, virtual ou não, no *storage array*. OPCIONAL.

```
                Naming      String defined by
 Type  Date      Auth       "example.com" naming authority
+--++------+ +----------+ +------------------------------+
|  ||      | |          | |                              |
```

```
iqn.1992-01.com.example:storage:diskarrays-sn-a8675309
iqn.1992-01.com.example
iqn.1992-01.com.example:storage.tape1.sys1.xyz
iqn.1992-01.com.example:storage.disk2.sys1.xyz
```

- **Formato eui..**

  O endereço EUI, usa o formato *standard* EUI-64 da *Institute of Electrical and Electronics Engineers* (IEEE)[8], sendo o mesmo composto por um identificador de 64-bit.

  Um endereço EUI pode ser formado de três formas diferentes[36]:

  – Uma concatenação de um endereço *Organizationally Unique Identifier* (OUI) de 24-bit atribuído pela IEEE Registration Authority (IEEE RA) com um endereço de 40-bit atribuído pela organização em questão seguindo a norma OUI.

  – Uma concatenação de um endereço de 28-bit atribuído pela IEEE RA com um endereço de 36-bit atribuído pela organização em questão.

  – Uma concatenação de um endereço OUI-36 de 36-bit atribuído pela IEEE RA com um endereço de 28-bit atribuído pela organização em questão seguindo a norma OUI-36.

```
+--++---------------+
|  ||               |
eui.02004567A425678D
```

3.5.2 *Fibre Channel* (FC)

Ao referirmos *Fibre Channel* (FC), podemos referirmo-nos ao protocolo de transporte *Fibre Channel Protocol* (FCP) ou à tecnologia FC.
O FC é uma tecnologia de rede de alta velocidade (até 16Gb por segundo), tendo como objetivo conectar e interligar armazenamento. Tornando-se um *standard* ANSI em 1994, atualmente é uma tecnologia comum associada às redes SAN.
Por outro lado, o protocolo de transporte FCP tem como objetivo transportar comandos SCSI sobre uma rede FC.

[8] IEEE é uma associação profissional com o objetivo de promover o avanço tecnológico, científico e educacional ao nível da eletrónica, eletrotecnia, informática e telecomunicações.

Existem outras aplicações de FC na indústria, como o Internet *Fibre Channel Protocol* (iFCP)[9], *Fibre Channel over IP* (FCIP)[10], *Fibre Channel over Ethernet* (FCoE)[11], entre outros.

3.6 Soluções de virtualização de rede existentes

O SDN é um conjunto de tecnologias, tendo cada uma destas uma função especifica dentro de toda a gestão de uma rede. Consultar figura 3.2.

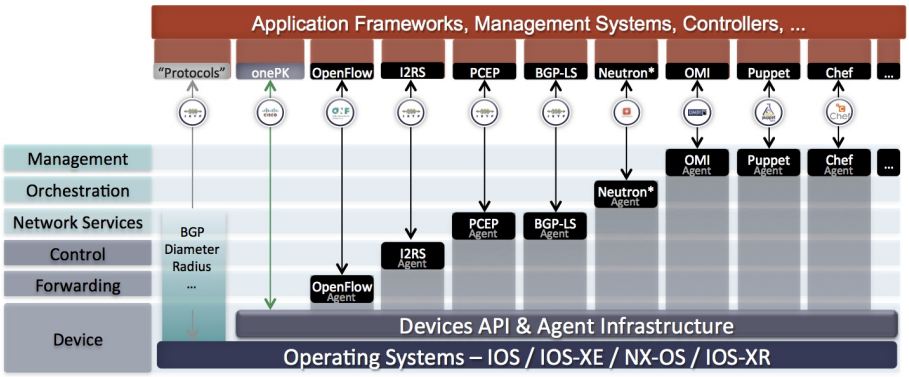

Figura 3.2: Arquitetura da tecnologia SDN[6]

O *OpenFlow* (OF) é um dos protocolos fundamentais para que seja possível implementar uma correta solução de SDN. O objetivo do OF é retirar do *hardware* de encaminhamento todo o trabalho extra, como por exemplo, balanceamento de carga, limitar tráfego na rede entre máquinas, *firewall*, entre outros e deixar apenas a função de encaminhamento. Ao ser retirado todo este processamento extra, é possível a um protocolo como o OF interagir com a(s) tabela(s) de encaminhamento e adaptar toda a rede consoante as necessidades presentes. Atualmente encontra-se na versão 1.6 tendo sido esta disponibilizada em 09/2016[40].
Trata-se de um protocolo de comunicação, que permite identificar padrões na rede através de regras pré-definidas, estáticas ou dinâmicas, programadas pelo controlador. O OF, ao

[9] O iFCP é um protocolo de rede ponto-a-ponto com o objetivo de fornecer funcionalidades FC a dispositivos FC sobre uma rede IP usando mecanismos de *routing*. O iFCP é um *standard Internet Engineering Task Force* (IETF)[37].

[10] O FCIP é um protocolo de rede ponto-a-ponto que possibilita a comunicação entre dois equipamentos FC sobre IP usando mecanismos de *tunneling*. O FCIP é um *standard* IETF[38].

[11] O FCoE é um protocolo de rede ponto-a-ponto com o objetivo de fornecer funcionalidades FC a dispositivos FC sobre uma rede *Ethernet* usando mecanismos de *routing*[39].

permitir esta flexibilidade e esta separação entre o controlo de fluxo e o encaminhamento, obtém uma gestão de toda a rede mais sofisticada, centralizada e simplificada[26].
Para além das vantagens já apresentadas aqui, o OF é uma solução *open-source*.

3.6.1 Funcionamento do *OpenFlow* (OF)

O OF é constituído por quatro módulos[26]:

- **Controlador;**

  Toma decisões na rede, encaminhando o tráfego para os agentes OF.

- ***Northbound* APIs;**

  Estas APIs efetuam a ligação entre o controlador e as aplicações configuradas, como a *firewall*, balanceamento de carga, entre outras.

- **Agentes OF;**

  Equipamento de rede com capacidade de comunicação OF, comunicando assim com o controlador e adaptando-se às suas instruções.

- **Troca de mensagens.**

  Troca de mensagens entre o controlador e os agentes OF.

Ao usar um protocolo com as características do OF, toda a rede ficará programada e programável, permitindo assim um controlo muito específico, alterações em tempo real, baseando-se no utilizador ou no nível de sessão apresentado. O encaminhamento tradicional IP presente em cada um dos equipamentos de encaminhamento não permite esta flexibilidade. Esta limitação existe devido à premissa que todo o tráfego, ou fluxos de dados, entre dois *hosts*, terá de seguir o mesmo caminho através da rede, independentemente das suas diferentes necessidades.

3.6.2 Encaminhamento OpenFlow vs Encaminhamento IP tradicional

Ao ser usado encaminhamento OF ao invés de encaminhamento IP tradicional, obtém-se as seguintes vantagens[26]:

- **Controlo sobre equipamentos de vários vendedores;**

O OF, ao ser um protocolo *open-source*, permite que qualquer vendedor o implemente nos seus equipamentos. Assim, basta o equipamento ser compatível com OF, para ser possível implementar numa solução deste tipo. Esta abordagem impede *vendor lock-in*[12], sendo assim mais simples e rápido lançar updates na rede e configurar novos equipamentos.

- **Complexidade reduzida através de automação de tarefas;**

  Devido à flexibilidade, gestão centralizada e rede programável que o OF fornece, é possível desenvolver ferramentas com o intuito de automatizar a rede. Estas ferramentas oferecem uma menor interação na rede por parte dos administradores de rede, baixando assim os recursos humanos necessários para manter uma rede informática e também baixar a instabilidade na rede devido a erros humanos.

  Para além desta automação, em aplicações *cloud*, estas podem ser geridas a partir de sistemas de orquestração e de aprovisionamento inteligente, reduzindo assim o conhecimento técnico de quem gere as mesmas e aumentar a agilidade do negócio em questão.

- **Maior taxa de inovação;**

  Capacidade de implementação de novos serviços e novas funcionalidades na rede ou nas aplicações sobre a mesma, de forma mais rápida, devido à resposta em tempo real às necessidades do serviço.

- **Maior fiabilidade e segurança da rede;**

  Configuração e alteração da topologia da rede de forma simultânea em todos os equipamentos, reduzindo assim a necessidade de configuração a cada equipamento. Com esta característica, elimina-se a incoerência criada na rede e os problemas humanos associados devido à configuração individual dos equipamentos.

- **Controlo mais complexo sobre a rede;**

  O controlo por fluxo que o OF fornece, permite aos administradores de rede aplicar atualizações na rede de forma muito específica, abstrata e automatizada. Introduz suporte multi-locação aos administradores de rede, mantendo e alterando o isolamento de tráfego, segurança e gestão de recursos enquanto os utilizadores usam a rede.

[12] *Vendor lock-in* é o termo utilizado quando uma solução está associada a apenas um vendedor, devido à tecnologia proprietária (*closed-source*).

- **Melhorar experiência para os utilizadores.**

  Com a possibilidade de um controlo específico sobre cada fluxo de dados, é possível dar ao utilizador uma melhor qualidade de serviço. Esta melhoria é possível através da adaptação do tráfego e da qualidade de um serviço, baseando-se no tipo de subscrição de um utilizador, adaptando a qualidade do streaming de vídeo através da largura de banda disponível naquele momento (*throughtput*) e não no tipo de ligação ou largura de banda que subscrevemos.

3.6.3 Soluções OpenFlow *opensource*

Atualmente já existem várias soluções OF tanto proprietárias como *opensource*. Em 2015 a *Open Networking Foundation* anunciou um novo *website* (`http://opensourcesdn.org`) com o objetivo de criar um ponto central para o desenvolvimento e repositório de soluções SDN livres de código proprietário[41]. Neste *website* a *Open Networking Foundation* filtra e apresenta as melhores soluções criadas pela comunidade. Ao nível de soluções OF são apresentadas as seguintes soluções[42]:

- **NOX**;

  Primeiro controlador de OF disponibilizado. Atualmente já não se encontra em desenvolvimento, mas serve como base para novos desenvolvimentos.

- **POX**;

  O POX é um controlador criado pelos mesmos indivíduos do NOX (Universidade de Stanford). Atualmente é um dos controladores com mas desenvolvimento.

- **ODENOS (O3 Orchestrator Suite), Trema e Ryu**;

  Ao contrário do NOX e do POX, o ODENOS, o Trema e o Ryu, não são controladores, mas sim bibliotecas de *software* (ou *frameworks*) para o desenvolvimento de controladores OF.

- **ONOS: Open Network Operating System**;

  O ONOS é o OS construído de raiz para toda a temática de SDN, oferecendo funções de controlador de OF, alta disponibilidade, desempenho e escalabilidade.

- **OpenDaylight**; A OpenDaylight é uma plataforma com o objetivo de disponibilizar ferramentas de SDN e *Network function virtualization* (NFV)[13] em redes de computa-

[13]NFV é uma arquitetura de rede que consiste na virtualização de um conjunto inteiro de funções de rede.

dores. Oferece ferramentas como controlador de OF, *plugin* para o OpenStack Neutron, entre outros.

- **Project Floodlight: An Open SDN Controller**.

  O controlador Floodlight é um controlador de OF com uma API *Hypertext Transfer Protocol* (HTTP)[14] RESTful, suportando interligação com o OpenStack através de de um *plugin* para o Neutron.

NOX

Devido ao NOX já não se encontrar em desenvolvimento (estando o seu *website* fora de serviço), não é uma opção viável para a implementação desenvolvida nesta dissertação.
No entanto, é uma ótima ferramenta para compreensão do protocolo OF e os mecanismos em volta do mesmo[43][42].

POX

O POX é baseado em Python, suportando a versão 1.0 do protocolo OF e algumas funcionalidades da versão 1.1.
O POX pode ser utilizado com vários intuitos. Podem ser utilizadas as APIs do mesmo para criar um novo controlador de OF, ou uma ferramenta que inclua um controlador de OF, ou então pode ser utilizado como um tópico controlador de OF, permitindo:

- Deteção automática de cenários com OF, permitindo assim uma otimização do controlador para a tarefa em questão;
- Deteção automática de equipamentos OF na rede usando *layer 2* (mapeamento de endereços MAC, apenas domínio de *broadcast* no qual está inserido) ou usando *layer 3* (mapeamento de endereços IP, suporte a domínios de *broadcast* distintos);
- Gestão de fluxos de dados entre os agentes de OF:
  - Novas conexões a agentes de OF ou novos fluxos de dados;
  - Termino de conexões a agentes de OF ou termino de fluxos de dados;
  - Alteração de portos em fluxos de dados existentes;
  - Entre outros.

[14] O HTTP é o protocolo aplicacional base usado na troca de informação na *web*

- Criação, alteração ou eliminaçao de tabelas de fluxos de dados;
- Suporte a mecanismos de *firewall* usando OF;
- Suporte a mecanismos de *Access Control List* (ACL)[15] usando OF;
- Suporte a marcação e identificação de tráfego recorrendo a:
  - Endereços MAC;
  - Endereços IP;
  - Protocolo de transporte (UDP ou TCP);
  - Porto;
  - Protocolo aplicacional (HTTP, FTP, etc).
- Suporte a extensões.

Apesar de o POX apresentar todas as características relevantes ao nível do protocolo OF para a implementação desenvolvida nesta dissertação, há alguns aspetos negativos no projeto e alguns conhecimentos em outras ferramentas, como[44]:

- Não disponibiliza uma interface gráfica, não existindo assim um modo simples e intuitivo de configurar o controlador e as regras:
  - Conhecimentos sobre terminal de modo a ser possível gerir o controlador (ligar, desligar, alterar parâmetros, etc);
  - Conhecimentos ao nível da linguagem de programação Python, de modo a ser possível configurar vários aspetos do controlador (configuração de *firewall*, marcação de pacotes, alterar tabelas de fluxos de dados, entre muitos outros).
- Não existe interligação nativa com o OpenStack.

ODENOS, Trema e Ryu

Devido à natureza destes dois projetos, em que o objetivo é disponibilizar ferramentas para criar, testar e lançar novos controladores de OF e não a disponibilização de um controlador OF funcional e bem documentado, estas duas soluções não são viáveis para a implementação desenvolvida nesta dissertação[45][46][47].

[15] ACL é uma lista de regras que define as permissões de um utilizador/máquina a um determinado componente do serviço.

ONOS: Open Network Operating System

O ONOS tem como objetivo oferecer aos seus utilizadores um controlador de OF funcional e bem documentado de forma a oferecer uma boa experiência tanto ao utilizador como ao *developer*.
Ao nível de funcionalidades, o ONOS oferece funcionalidades semelhantes ao POX:

- Gestão de fluxos de dados entre os agentes de OF;
- Criação, alteração ou eliminação de tabelas de fluxos de dados;
- Suporte a mecanismos de *firewall* usando OF;
- Suporte a mecanismos de ACL usando OF;
- Suporte a balanceamento de carga;
- Suporte a marcação e identificação de tráfego.

Oferece também:

- Interface gráfica simples e intuitiva para a gestão do controlador;
- Solução de alta disponibilidade.

No entanto, como no caso do POX, o ONOS apresenta aspetos negativos e necessita de vários conhecimentos em outras ferramentas, como[48]:

- Conhecimentos sobre terminal de modo a ser possível:
  - Instalar dependências;
  - Instalar ONOS;
- Conhecimentos de JSON para possível configuração/compreensão de configurações;
- Não existe interligação nativa com o OpenStack.

OpenDaylight

O OpenDaylight não pode ser visto como um simples controlador de OF, mas sim uma plataforma que contém várias funcionalidades relevantes para uma implementação de SDN. Dentro de muitas funcionalidades, o OpenDaylight contém as seguintes, relevantes para esta dissertação:

- Suporte ao protocolo OF:
  - Suporte a implementação como controlador;
    - Gestão de fluxos de dados entre os agentes de OF;
    - Criação, alteração ou eliminação de tabelas de fluxos de dados;
    - Suporte a mecanismos de *firewall* usando OF;
    - Suporte a mecanismos de ACL usando OF;
    - Suporte a balanceamento de carga;
    - Suporte a marcação e identificação de tráfego.
  - Suporte a implementação como *switch*.
- Interface gráfica simples e intuitiva;
- Interligação com o OpenStack Neutron;
- Serviço de VPN;
- Mecanismos de segurança;
- Entre outros.

Oferece também uma ótima documentação, quer na configuração de toda a plataforma, como a integração da mesma com o OpenStack Neutron.
Para além de todas as funcionalidades que oferece, oferece também uma arquitetura modular com o mesmo objetivo que o OpenStack. Apenas são instaladas/ativadas as funcionalidades necessárias de forma a tornar a ferramenta o mais dinâmica possível.
É também uma ferramenta com um desenvolvimento muito ativo, libertando uma nova versão com novas funcionalidades e erros corrigidos a cada seis meses.
É também necessário, como em outras ferramentas, vários outros conhecimentos, como[49]:

- Conhecimentos sobre terminal de modo a ser possível:
  - Instalar dependências;
  - Instalar OpenDaylight;

Project Floodlight: An Open SDN Controller

À semelhança do POX e do ONOS, o Floodlight é um controlador de OF com o objetivo de oferecer um controlador funcional e bem documentado. Atualmente encontra-se na versão 1.2 (libertada a 07/02/2016)[7] baseado na linguagem de programação Java.
Trata-se de um projeto com desenvolvimento ativo, mas ao contrário do OpenDaylight, não existe um período de tempo definido entre versões.
Ao nível de funcionalidades o Floodlight oferece funcionalidades semelhantes ao POX e ao ONOS:

- Gestão de fluxos de dados entre os agentes de OF;
- Criação, alteração ou eliminação de tabelas de fluxos de dados;
- Suporte a mecanismos de *firewall* usando OF;
- Suporte a mecanismos de ACL usando OF;
- Suporte a balanceamento de carga;
- Suporte a marcação e identificação de tráfego.

Oferece também:

- Suporte a interligação com o OpenStack;
- Suporta tipologias de rede híbridas, onde nem todos os equipamentos suportam OF (consultar figura 3.3).

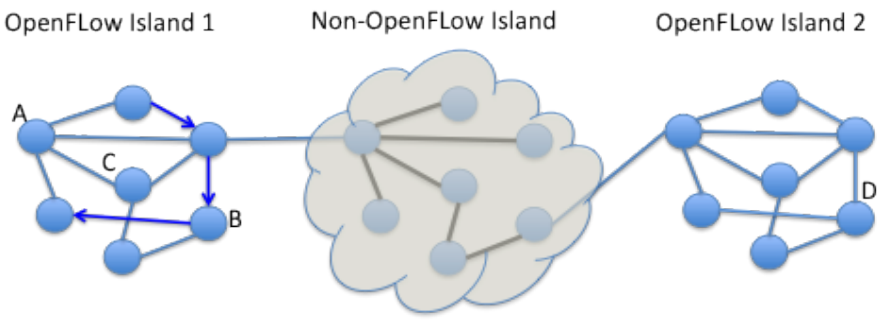

Figura 3.3: Tipologia de rede híbrida suportada pelo Floodlight[7]

No entanto, como no caso do POX e do ONOS, o Floodlight apresenta aspetos negativos e necessita de vários conhecimentos em outras ferramentas, como[7]:

- Não disponibiliza uma interface gráfica, não existindo assim um modo simples e intuitivo de configurar o controlador e as regras:
  - Conhecimentos sobre terminal de modo a ser possível gerir o controlador (ligar, desligar, alterar parâmetros, etc);
  - Conhecimentos ao nível do serviço cURL de modo a ser possível criar, alterar ou eliminar regras no controlador.
- Toda a componente de interligação com o OpenStack é baseada na API antiga do mesmo, não sendo uma opção viável para as novas versões de OpenStack.
- Algumas funcionalidades não se encontram totalmente funcionais como é possível observar na documentação do mesmo.

  > *The code is not considered complete at this time, but it supports basic creation and use of load balancer for icmp, tcp, and udp services.*

3.7 Conclusões

Depois de várias soluções terem sido estudadas, é possível concluir que existem várias hipóteses a considerar numa implementação como esta.
Existem outras soluções disponíveis, mas com este estado da arte o objetivo foi um foco nas soluções mais completas e com uma maior posição no mercado, de modo a ser obtida uma implementação robusta, completa e *future proof*.

Capítulo 4

Arquitectura e implementação

Neste capitulo, irá ser apresentada a arquitetura definida para todas as comparações e demonstrações, a implementação da mesma e os passos intermédios escolhidos. Com esta implementação, pretende-se chegar o mais próximo possível de uma implementação real.
De modo a tornar o cenário o mais próximo da realidade possível, foi criado um ambiente virtual em *Graphical Network Simulator-3* (GNS3) e em VirtualBox, simulando um DC e um cliente. Neste cenário foi também tido em conta toda a configuração de encaminhamento entre todas as máquinas de forma redundante. Consular figura 4.1.
Para toda a gestão da infraestrutura do DC foi escolhido o OpenStack devido ao seu melhor desempenho, suporte e comunidade face ao CloudStack e o OpenDaylight como controlador de OF devido a ser o mais completo de todas as soluções apresentadas.

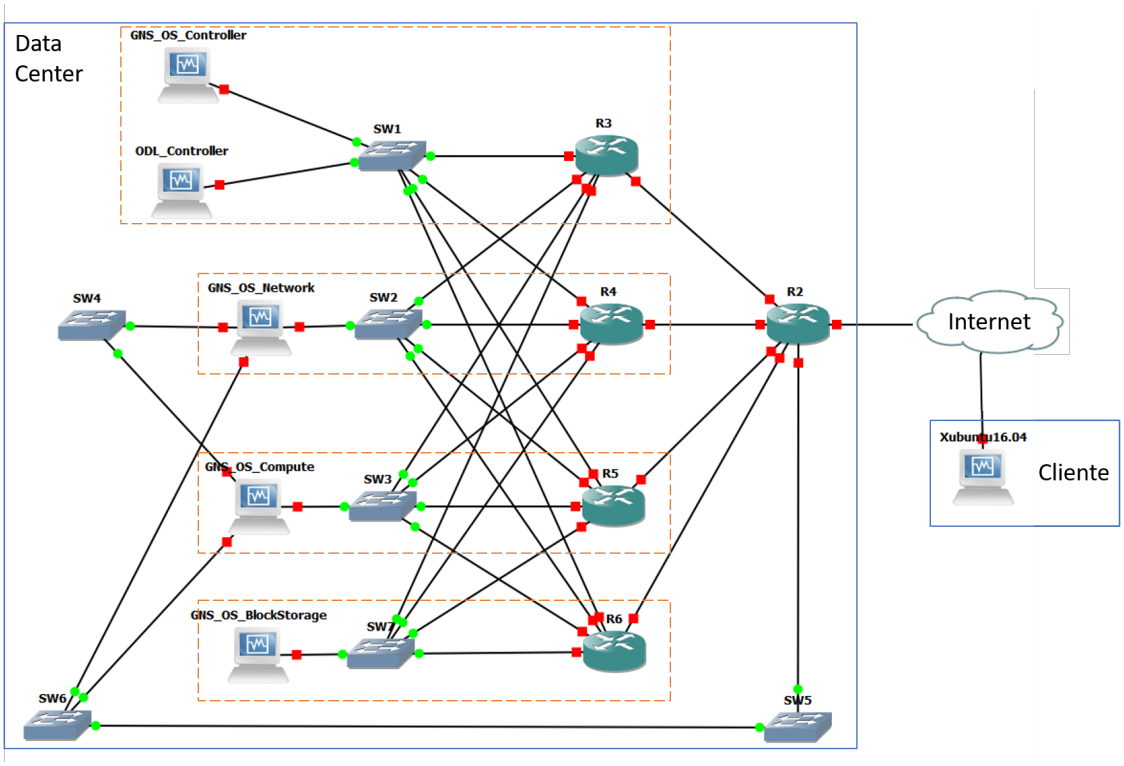

Figura 4.1: Cenário de rede virtual em GNS3 e VirtualBox

4.1 *Graphical Network Simulator-3*

O *software Graphical Network Simulator-3* (GNS3) irá ser a ferramenta onde toda a arquitetura irá ser desenvolvida e onde todos os testes irão ser realizados.

Trata-se de uma ferramenta com o objetivo de virtualizar/emular uma rede de computadores, englobando aspetos como encaminhamento de *layer 2* e de *layer 3*. Consoante os recursos do *host* onde o GNS3 irá ser executado, as aplicações poderão ser distintas, podendo ser possível virtualizar/emular redes simples de pequena dimensão a redes de larga escala, ex. simulação de redes em cenários de DC, simulação de redes de telecomunicações em larga escala, entre outros.

É uma ferramenta altamente flexível ao nível de implementação devido ao suporte *multi--vendor* (mais de 20 vendedores e os seus OSs são suportados no GNS3), oferecendo uma simulação em tempo real de toda uma infraestrutura de redes de computador sem a necessidade de *hardware* de rede. Para além de toda esta flexibilidade, o GNS3 permite ainda interligar toda uma implementação a uma rede real, de modo a ser possível testar todo o

cenário antes de uma implementação em ambiente de produção[50].
Para além do suporte *multi-vendor* com os principais construtores de equipamentos de encaminhamento, o GNS3 também suporta a interligação com ferramentas de virtualização de sistemas.
É uma ferramenta que suporta extensões ou módulos de modo a ser possível extender as funcionalidades da mesma[51].

4.2 VirtualBox

O VirtualBox é um *software* de virtualização totalmente *opensource*. Esta ferramenta irá servir de base para todas as VMs criadas para implementar a solução de OpenStack.
Graças à sua compatibilidade com os principais OSs (Windows, macOS e Linux), a presença das principais funcionalidades esperadas, a constante adição de novas e o facto de ser totalmente *opensource*, o VirtualBox é uma das escolhas principais não só para a virtualização de sistemas em ambientes pessoais, mas também em ambientes empresariais.
O VirtualBox suporta um diverso número de *guest* OSs, desde o Windows NT 4.0 a OpenBSD.
É de realçar que para além da flexibilidade da ferramenta, o VirtualBox suporta *nested virtualization*[1], um ponto importante no decorrer da implementação desta dissertação.
Em constante evolução e atualização, o VirtualBox à semelhança do OpenStack, é gerido por uma entidade (Oracle), sendo mantido pela comunidade[52].
o VirtualBox é uma das ferramentas de virtualização de sistemas que oferece interligação com o GNS3.

4.3 Endereçamento IP *Data Center* (DC)

Neste ponto irão ser identificados os diferentes tipos de encaminhamento implementados na arquitetura desenvolvida.

4.3.1 Endereçamento físico

Como referido anteriormente na introdução do capítulo 4, um dos objetivos desta arquitetura é ser o mais próximo possível de uma implementação real de um DC. Para este efeito foram escolhidos os seguintes fatores na configuração do endereçamento (consultar figura 4.2):

[1] Capacidade de executar VMs sobre VMs.

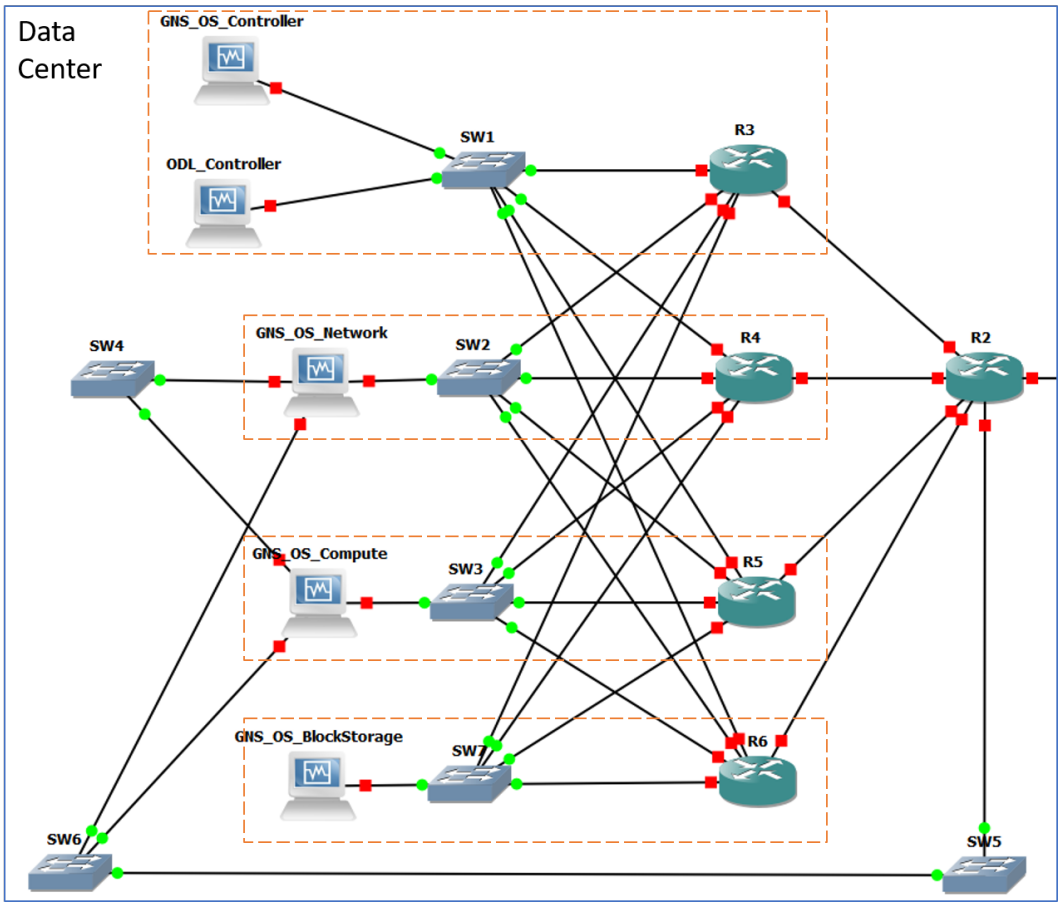

Figura 4.2: Endereçamento IP de simulação de ambiente DC

- Simulação de equipamentos de acesso ao exterior (equipamento R1);
  - Um cenário considerado normal numa topologia deste tipo, seria vários equipamentos de acesso ao exterior, sendo cada um destes de *providers* distintos. Neste caso foi apenas um configurado, devido ao limite de recursos disponíveis no *host* de testes.
- Simulação de equipamentos intermédios entre os vários bastidores do DC e os equipamentos de acesso ao exterior (equipamento R2);
  - À semelhança do equipamento anterior, é esperado o uso de vários dispositivos intermédios de modo a garantir a redundância e a alta disponibilidade do serviço, no entanto, devido aos recursos limitados do *host* de testes, apenas um equipamento é utilizado para demonstrar.

- Simulação de equipamentos de acesso a cada bastidor (equipamentos R3, R4, R5 e R6), de modo a garantir domínios de *broadcast* distintos por bastidor;
  - Redundância e alta disponibilidade configurada nos vários bastidores.
- Simulação de equipamentos de *layer* 2 (equipamentos SW1, SW2, SW3 e SW7).
  - Estes equipamentos poderão ser facultativos no caso dos equipamentos de *layer* 3 de acesso ao bastidor, também efetuaram encaminhamento de *layer* 2;
  - À semelhança de outros equipamentos, é esperado o uso de vários dispositivos para efeitos de redundância e alta disponibilidade, mas devido aos recursos limitados do *host* de testes, apenas um equipamento é utilizado para demonstrar.

Um dos fatores de um encaminhamento DC bem definido e estruturado, é o uso sub-endereçamento para otimização:

- Uso de redes com máscara de rede[2] /30 entre os equipamentos R2-R3, R2-R4, R2-R5 e R2-R6;
- Uso de redes com máscara de rede /24 para as redes dos bastidores.

  Estas redes poderiam também ter máscaras de rede /30 devido ao uso de poucos *hosts* nos testes efetuados. No entanto, por norma neste tipo de cenário, os bastidores costumam conter vários servidores, muitas das vezes tendo estes VMs a executar sobre os mesmos, sendo assim mais fidedigno o uso de redes de maior dimensão;
- Uso de redes com máscara de rede /24 nas redes de apoio ao OpenStack, i.e. rede pertencente ao *switch* SW4 e SW6.

  O equipamento SW4 irá servir de suporte à plataforma de OpenStack para a gestão e configuração das redes virtuais e instâncias, e o equipamento de SW6 irá conter a rede onde todos os IPs flutuantes das instâncias irão ser criados.

A configuração destes equipamentos encontra-se disponível no anexo A.

4.3.2 Endereçamento virtual

Para além de todo o encaminhamento físico existente na arquitetura, também irá existir encaminhamento virtual associado às redes virtuais criadas para dar suporte às VMs criadas.

[2]Máscara de rede é um número de 32 bits usado indicar o tamanho de uma rede, i.e. o número de *hosts*.

Este encaminhamento poderá ser local, ou seja, apenas entre equipamentos na mesma subrede, entre várias redes locais virtuais ou entre as redes virtuais e a rede física existente na arquitetura, garantindo assim um IP público às VMs existentes (consultar 4.3).

```
testes@networkTestes:~$ sudo ovs-vsctl show
0fd910cc-a8e4-4690-897e-3b2e18a7a367
    Manager "tcp:192.168.10.11:6640"
        is_connected: true
    Manager "ptcp:6640:127.0.0.1"
    Bridge br-int
        Controller "tcp:192.168.10.11:6653"
            is_connected: true
        fail_mode: secure
        Port "tund47a245a938"
            Interface "tund47a245a938"
                type: vxlan
                options: {key=flow, local_ip="10.0.1.21", remote_ip="10.0.1.31"}
        Port br-int
            Interface br-int
                type: internal
        Port "tapc634da98-c7"
            Interface "tapc634da98-c7"
                type: internal
        Port "tap8c50b0eb-12"
            Interface "tap8c50b0eb-12"
                type: internal
    Bridge br-provider
        Port br-provider
            Interface br-provider
                type: internal
        Port "enp0s9"
            Interface "enp0s9"
    Bridge br-tun
        Port br-tun
            Interface br-tun
                type: internal
    ovs_version: "2.6.1"
```

Figura 4.3: Redes virtuais representadas no Open vSwitch

De modo a providenciar um IP público, ou IP flutuante, às VMs, com o objetivo de publicar serviços ou aceder à VM a partir de qualquer ponto, sem ser necessário qualquer ligação com a infraestrutura ou com a solução de OpenStack, é necessário que ambas as máquinas de *compute* e de *network*, contenham uma interface fisica dedicada para este efeito. Esta interface física não terá qualquer IP, tendo sim interfaces virtuais associadas à mesma através do Open vSwitch. Com esta abordagem, irá ser criada uma interface virtual sobre estas interfaces físicas para cada IP público atribuído às VMs. Estas interfaces virtuais já terão IPs atribuídos.

À semelhança de providenciar IPs flutuantes às VMs, para providenciar redes virtuais às VMs é também necessário uma interface dedicada para este efeito. A esta interface estará associada uma rede física comum aos nós *compute* e de *network* onde irá ser efetuada toda a configuração e troca de mensagens entre os serviços. Este segmento de rede irá ser o elo de ligação entre os módulos de Neutron e de Nova existentes nestes dos nós. Para cada VM criada, será também criada uma interface do tipo *tap* no serviço de Open vSwitch, como é possível observar na figura 4.3.

4.4 Implementação OpenStack

Existem várias formas de configurar uma infraestrutura OpenStack, desde implementações simples numa VM a implementações mais complexas, usando várias nós físicos. A OpenStack Foundation através da comunidade, disponibiliza uma ferramenta designada de *DevStack*[53], com o objetivo de configurar rapidamente e facilmente uma solução de OpenStack. Devido a toda a simplicidade associada, não é aconselhado uma implementação deste tipo num ambiente de produção. É também necessário salientar que a maioria das configurações e soluções apresentadas para o DevStack não são totalmente testadas, não garantindo assim a fiabilidade da ferramenta.

Como é possível observar na documentação oficial do DevStack na secção *Overview*:

> *DevStack has evolved to support a large number of configuration options and alternative platforms and support services. That evolution has grown well beyond what was originally intended and the majority of configuration combinations are rarely, if ever, tested. DevStack is not a general OpenStack installer and was never meant to be everything to everyone.*

Devido a estas características da ferramenta DevStack, e com o objetivo de tornar este cenário o mais próximo da realidade possível, foi então escolhido implementar uma configuração mais robusta.

Nesta configuração o OpenStack será instalado por módulos sendo a instalação de cada um destes módulos independente. É possível visualizar a hierarquia destes módulos na figura 4.4.

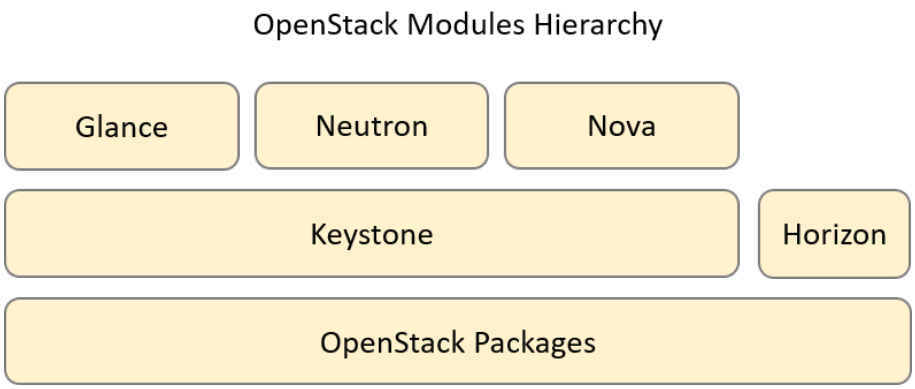

Figura 4.4: Hierarquia dos módulos do OpenStack

Estes módulos irão ser explicados com mais detalhe de seguida.

4.4.1 Requisitos de *hardware*

Na documentação do OpenStack para o ambiente escolhido[8], o cenário demonstrado não complementa a gestão dos *switchs* virtuais através do *Open vSwitch* (OvS). Como é relevante para a interligação com o controlador de OF ter uma gester compatível com OF, o cenário será adaptado para essa realidade.
No mínimo, será necessário um nó *controller*, um nó *network* e um nó *compute*. Há a possibilidade de configurar outros nós do tipo *compute*, nós do tipo *block storage* e nós do tipo *object storage*. Estes nós tem como objetivo: (Consultar figura 4.5)

- **Nó *controller*:** Este nó tem como objetivo ser o centro da implementação OpenStack, tendo assim todos os serviços base do mesmo, servindo como ponto central de gestão.

  Neste nó estarão implementados serviços como o Keystone (*identity service*), Glance (*image Service*), componentes de gestão do Nova (*compute service*), componentes de gestão do Neutron (*networking service*) e o Horizon (*dashboard service*). Conta também com serviços de DB (MySQL e NoSQL), *message queue* e de *Network Time Protocol* (NTP)[3].

  No nó *controller* também podem correr outros serviços, dependendo do tipo de instalação e configuração pretendidos, como por exemplo componentes de gestão do Swift (*object storage service*), componentes de gestão do Cinder (*block storage service*),

[3]O NTP é um protocolo para sincronização dos relógios de computador.

componentes de gestão e agentes do Ceilometer (*telemetry service*) e o Heat (*orchestration service*).

- **Nó *network*:** Este nó tem como objetivo ser o gestor de rede virtual da infraestrutura OpenStack. Aqui irão estar implementados vários agentes de rede (Neutron).

- **nó *compute*:** Este nó tem como objetivo correr um ou mais *hypervisors* de forma a operar todas as instâncias existentes. À semelhança do nó *network*, também existem agentes do Neutron.

- **nó *block storage*:** Este nó tem como objetivo armazenar todos os discos que o serviço Cinder provisiona às instâncias. O uso deste nó é opcional.

- **nó *object storage*:** Este nó tem como objetivo armazenar todos os discos que o serviço Swift utiliza para armazenar *accounts*, *containers*, e *objects*. O uso deste nó é opcional.

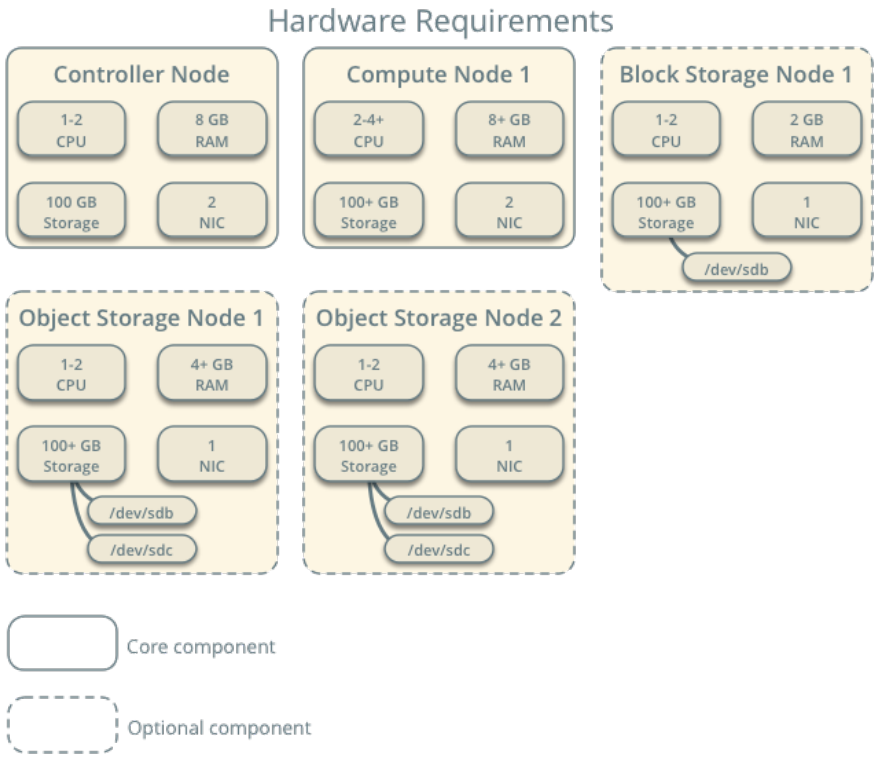

Figura 4.5: Requisitos minimos de *hardware* para o OpenStack[8]

Como é possível observar na figura 4.5, os requisitos físicos mínimos variam de nó paro nó devido aos diferentes tipos de utilização que cada nó tem.

4.4.2 Requisitos de rede

Ao nível de como toda a infraestrutura de rede virtual do OpenStack se pode implementar, existem dois tipos. Estes dois tipos de implementação assentam sobre o componente Neutron, tendo uma implementação mais funcionalidades que outra. Os dois tipos de implementação são:

- **Configuração simplificada de rede usando o Neutron (*provider networks*):** Esta implementação recorre ao serviço Neutron, da forma mais simples, disponibilizando apenas soluções de *switching/bridging L2* e segmentação de redes *Virtual Local Area Network* (VLAN)[4]. Nesta implementação o Neutron recorre a ligações físicas para todo o encaminhamento, efetuando *bridge* das redes virtuais para as físicas.

- **Configuração avançada de rede usando o Neutron (*self-service networks*:** Esta implementação adiciona soluções de *routing L3* permitindo redes *Virtual Extensible Local Area Network* (VXLAN)[5]. Aqui em vez de ser necessário recorrer às ligações físicas para obter encaminhamento, a própria rede virtual tem capacidades de encaminhamento, efetuando *Network Address Translation* (NAT)[6] para as ligações físicas. Com esta configuração é possível implementar serviços na infraestrutura do tipo VPNaaS, FWaaS e LBaaS.

No âmbito desta dissertação em que é relevante a implementação de funcionalidade de *layer 3*, é necessário optar pela opção de *self-service networks*.
Os requisitos de rede variam de nó para nó:

- ***Controller*, *Block Storage* e *Object Storage*:**

  - Uma interface de rede que irá garantir o acesso exterior (Internet) e o encaminhamento entre os vários nós OpenStack.

- ***Network* e *Compute*:**

[4]Uma rede VLAN é o equivalente a uma rede de área local, mas virtual, sendo assim possível, por exemplo, configurar várias VLANs na mesma ligação física, ou na mesma porta física.

[5]Uma rede VXLAN é a solução para o fato de só ser possível ter 4096 VLANs, sendo assim possível com a tecnologia VXLAN ter aproximadamente 16 milhões de redes virtuais.

[6]NAT é uma técnica que consiste em reescrever, utilizando-se de uma tabela hash, os endereços IP de origem de um pacote.

– Uma interface de rede que irá garantir o acesso exterior (Internet) e o encaminhamento entre as várias nós OpenStack;

– Uma interface de rede para redes virtuais do tipo túnel VXLAN;

– Uma interface de rede que irá garantir o acesso exterior (Internet) às VMs através de NAT e IPs flutuantes.

4.4.3 Requisitos de *software*

Nesta secção irá ser descrito todo o *software* necessário para uma correta implementação de OpenStack.

Software base

Antes da instalação dos componentes do OpenStack, é necessário instalar as suas dependências. Estas dependências são um servidor NTP de modo a que os relógios das nós OpenStack estejam sincronizados, serviços de DB SQL instalados para armazenamento de informação relativa aos serviços do OpenStack e um serviço de *message queue* de forma a coordenar toda a troca de informação entre os vários serviços da infraestrutura OpenStack[8]. É possível verificar a relação de todos estes na figura 4.6.

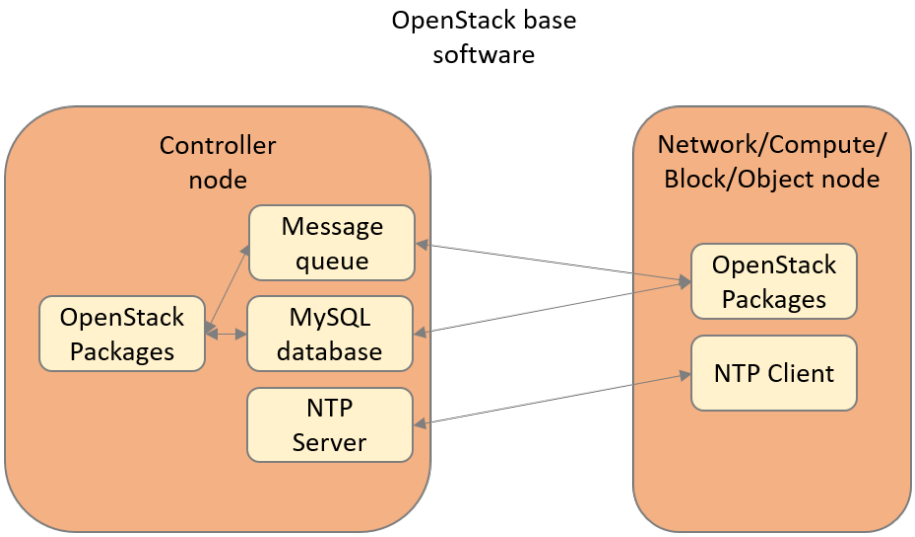

Figura 4.6: Diagrama software base OpenStack

OpenStack Keystone (identity)

O serviço Keystone (identity) fornece à infraestrutura um ponto central de autenticação e *catalog services* para todos os outros serviços. Dispõe de três componentes[8] (consultar figura 4.7).

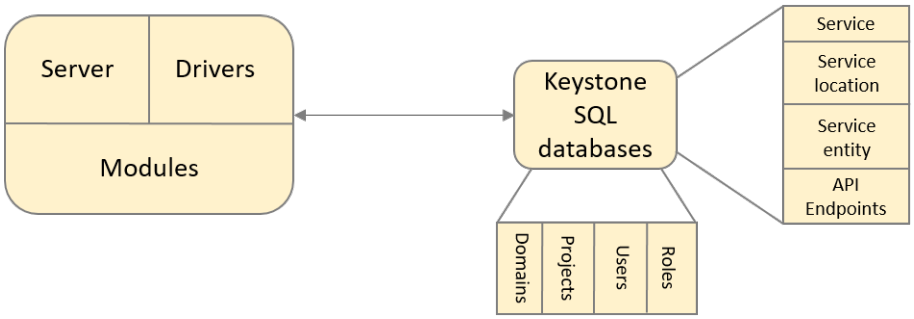

Figura 4.7: Diagrama Keystone (identity)

Estes três componentes têm como função:

- ***Server*:** Servidor central que fornece autenticação a todos os outros serviços através de uma interface RESTful.

- ***Drivers*:** Ferramenta usada pelo Keystone para aceder a informação alojada fora de serviços do OpenStack, ex. MySQL, entre outros.

- ***Modules*:** Módulos de *Middleware*[7] que intercetam os pedidos de autenticação, extraindo toda a informação relevante encaminhando-a para o *Server*.

Para além dos três componentes descritos em cima, o Keystone armazena ainda numa DB(s)My SQL externa ao OpenStack, toda a informação relativa ao mesmo. Nesta DB são guardados dois tipos de informação (consultar figura 4.7):

- Informação relativa a outros serviços OpenStack instalados na infraestrutura;

- Informação relativa à autenticação desses mesmos serviços no Keystone.

[7] *Middleware* é um programa que efetua a medição entre um outro programa e as demais aplicações. É fundamental para efetuar a transição entre aplicações/programas que usem protocolos ou tecnologias diferentes.

OpenStack Glance (image)

O serviço Glance (image) fornece à infraestrutura uma API com o objetivo de gerar discos ou imagens de VMs e *metadata* sobre as imagens disponíveis na infraestrutura. Dispõe de dois componentes[8] (consultar figura 4.8).

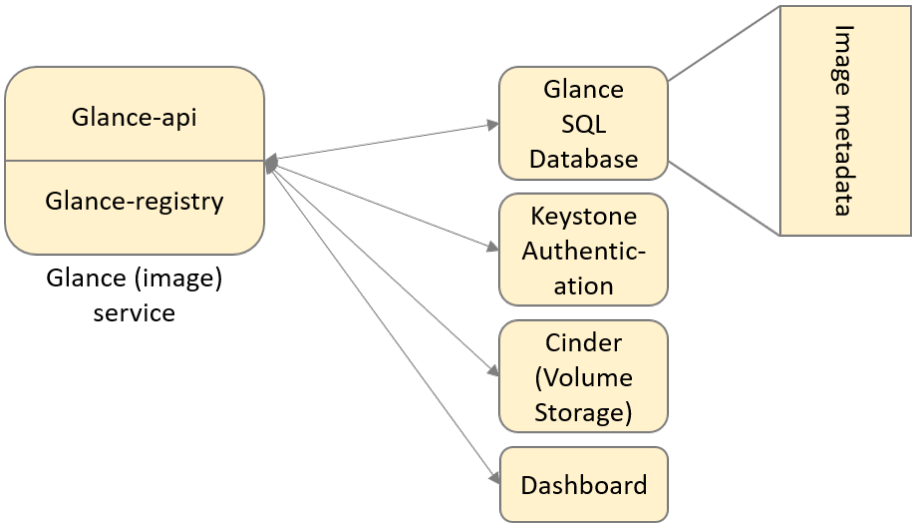

Figura 4.8: Diagrama Glance (image)

Estes dois componentes têm como função:

- ***Glance-api*:** Toda a API relativa ao serviço, com funcionalidades como criar, armazenar, descobrir, entre outras, imagens;
- ***Glance-registry*:** Ferramenta usada pelo Glance para armazenar, processar e retirar *metadata* sobre imagens na infraestrutura.

Para além dos dois componentes descritos anteriormente, o Glance armazena ainda numa(s) DB(s) SQL externa(s) ao OpenStack, toda a informação relativa ao mesmo, incluindo toda a *metadata* relativa às imagens disponíveis na infraestrutura.
Dispõe também da capacidade de armazenar todas as imagens em vários tipos de armazenamento, normal file systems, Swift (object storage), RADOS block devices, HTTP, e Amazon S3.
Existe também comunicação bidirecional com o serviço Horizon, com o objetivo de oferecer aos utilizadores da infraestrutura uma interface gráfica para gestão de todas as imagens existentes e a criação de novas.

Por fim, necessita também de comunicação bidirecional com o serviço Keystone para toda a autenticação na infraestrutura.

OpenStack Nova (compute)

O serviço Nova (compute) fornece à infraestrutura uma API com o objetivo de gerir todas as instâncias de VMs existentes na infraestrutura. Dispõe de catorze componentes[8] (consultar figura 4.9).

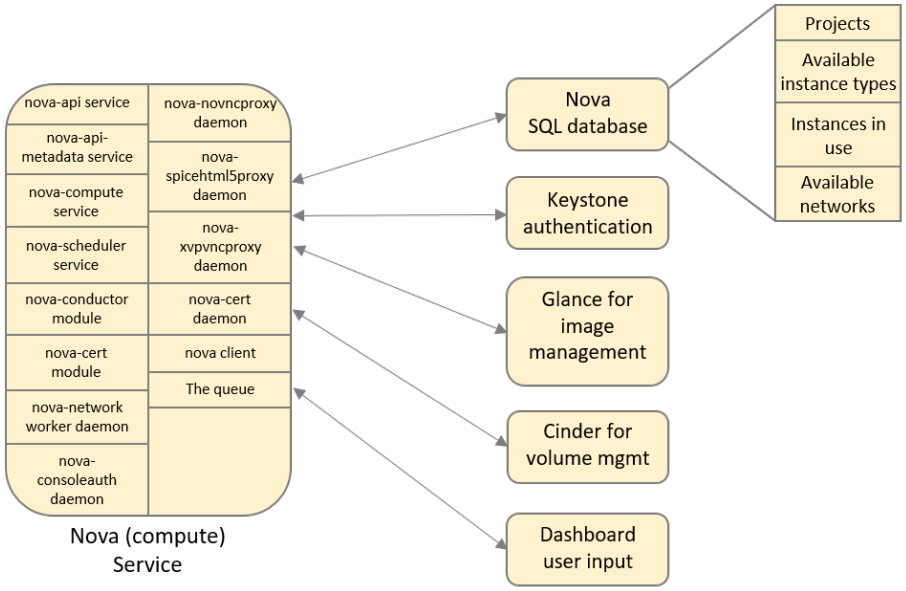

Figura 4.9: Diagrama Nova (compute)

Estes catorze componentes têm como função:

- ***Nova-api service* e *nova-client*:** API com o objetivo de oferecer ao utilizador ferramentas para interagir e gerir todas as VMs existentes na infraestrutura;
- ***Nova-api-metadata service*:** Trata de todos os pedidos sobre *metadata* efetuados pelas instâncias existentes na infraestrutura;
- ***Nova-compute service*:** Módulo de *Middleware* que gere as instâncias de VMs no *hypervisor* através das APIs do mesmo;
- ***Nova-scheduler service*:** Determina em que nó do tipo *compute* irá ser alocado um novo pedido de instância;

- **Módulo *nova-conductor*:** Módulo que gere todas as interações entre a ferramenta Nova e a DB;
- **Módulo *nova-cert* e *daemon nova-cert*:** Módulo que cria certificados. Necessário ao usar a API Amazon EC2;
- ***Daemon nova-network worker*:** Funciona em paralelo com o ***nova-compute service***, gerindo toda a rede associada à nova instância, desde encaminhamento entre esta e outras instâncias existentes, entre outras;
- ***Daemon nova-consoleauth*:** Gere o acesso às instâncias através de consola, atribuindo *tokens* aos clientes;
- ***Daemon nova-novncproxy*:** Cria um *proxy* para acesso às instâncias através de conexões *Virtual Network Computing* (VNC)[8] através de um *browser*;
- ***Daemon nova-spicehtml5proxy*:** Cria um *proxy* para acesso às instâncias através de conexões *Simple Protocol for Independent Computing Environments* (SPICE)[9] através de um *browser*;
- ***Daemon nova-xvpvncproxy*:** Cria um *proxy* para acesso às instâncias através de conexões VNC através de uma ferramenta especifica da OpenStack;
- ***The queue*:** Módulo usado para transferir mensagens (RabbitMQ) entre *daemons*.

Para além dos treze componentes previamente descritos, o Nova armazena ainda numa(s) DB(s) SQL externa(s) ao OpenStack, toda a informação relativa ao mesmo, incluindo toda a informação relativa às instâncias disponíveis na infraestrutura, entre outras.
O Nova comunica bidirecionalmente com o serviço Glance para acesso a imagens de forma a ser possível criar novas VMs e instâncias com base nessas imagens.
O Nova comunica bidirecionalmente com o serviço Cinder para acesso aos volumes das instâncias.
Existe também comunicação bidirecional com o serviço Horizon, com o objetivo de oferecer aos utilizadores da infraestrutura uma interface gráfica para gestão de todas as VMs existentes e a criação de novas.
Necessita também de comunicação bidirecional com o serviço Keystone para toda a autenticação na infraestrutura.

[8]Uma conexão VNC é uma ferramenta gráfica que permite o controlo remoto de umo nó através de outra.
[9]Uma conexão SPICE é uma ferramenta com o mesmo objetivo da conexão VNC.

Por fim, é de salientar que o serviço Nova necessita de estar instalado quer no nó *controller*, quer nos nós *compute*. A função do Nova difere do nó *controller* para o nó *compute*, estando o nó *controller* encarregue de toda monitorização e gestão do mesmo, interligando-o com os restantes serviços. Os nós *computes* tem como função gerir todas as instâncias existentes nos *hypervisors* disponíveis.

OpenStack Neutron (network)

O serviço Neutron (network) gere toda a rede virtual existente numa infraestrutura deste tipo. Gere questões como o encaminhamento, entre diferentes redes virtuais e físicas, encaminhamento entre as diversas VMs, entre outros. Dispõe de três componentes[8] (consultar figura 4.10).

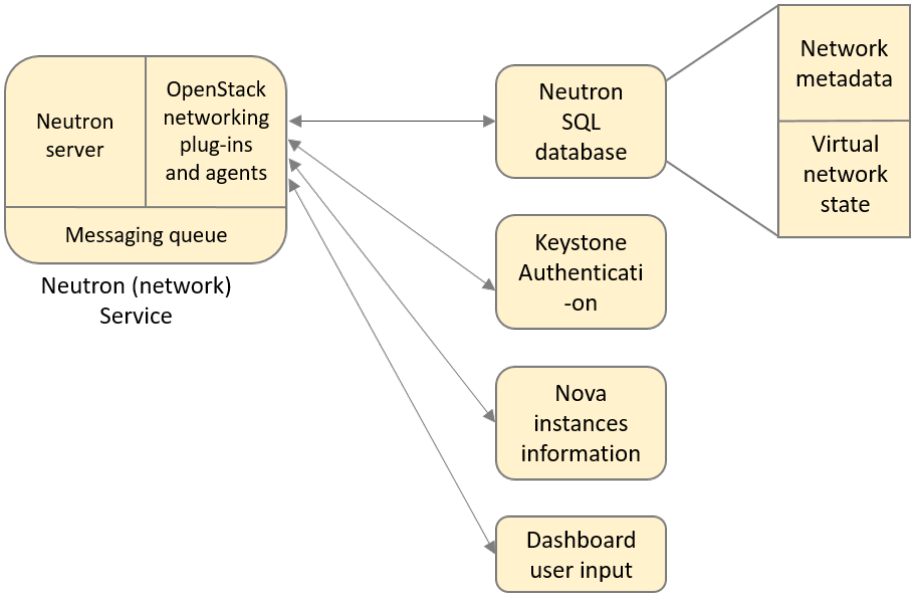

Figura 4.10: Diagrama Neutron (network)

Estes três componentes têm como função:

- ***Neutron server*:** Recebe pedidos da API do Neutron, encaminhando-os para o ***plug-in*** de rede OpenStack correto;
- ***OpenStack networking plug-ins and agents*:** Trata de questões específicas de rede, como cria ou elimina interfaces de redes virtuais para as VMs, cria ou elimina redes ou subredes, fornecendo encaminhamento IP;

- ***Messaging queue*:** Encaminha a informação entre o ***Neutron server*** e os ***plug-ins***.

Para além dos três componentes previamente descritos, o Neutron armazena ainda numa(s) DB(s) SQL externa(s) ao OpenStack, toda a informação relativa ao mesmo, incluindo toda a informação relativa às redes virtuais existentes, interfaces de rede das mesmas, entre outras.
O Neutron comunica bidirecionalmente com o serviço Nova de modo a garantir conectividade e encaminhamento entre as instâncias.
Necessita também de comunicação bidirecional com o serviço Keystone para toda a autenticação na infraestrutura.
Existe também comunicação bidirecional com o serviço Horizon, com o objetivo de oferecer aos utilizadores da infraestrutura uma interface gráfica para gestão de todas as redes e routers virtuais existentes e a criação de novas(os).
Semelhante ao serviço Nova, o Neutron também necessita de estar instalado em diversos nós. O nó *controller* está encarregue de toda monitorização e gestão do mesmo, interligando-o com os restantes serviços. O nó *network* irá conter os serviços de DHCP, agente e outros para dar suporte às instâncias do OpenStack. Os restantes nós (*compute*, *block storage* e *object storage*) necessitam de ter um agente configurado nas mesmas, de modo que alterações na topologia da rede possam ser adquiridas por todos os nós.

OpenStack Horizon (dashboard)

O serviço Horizon (dashboard) disponibiliza ao utilizador final uma interface gráfica web para interagir com a infraestrutura. O Horizon usa várias APIs do OpenStack para comunicar com os vários serviços existentes, como o Nova ou Neutron, disponibilizando a informação dos mesmos de forma compreensível para o utilizador final. É a partir deste serviço que é possível criar graficamente novas redes virtuais, novas VMs e novas instâncias das mesmas, entre outros[8].

4.4.4 Arquitetura desenvolvida

De forma a simplificar todo o processo e não sobrecarregar o nó no qual todo o cenário virtual está a ser implementado, foi escolhida uma arquitetura base para o OpenStack, facilmente escalável e alterada. Esta arquitetura consiste em cinco VMs com as seguintes características (consultar tabelas 4.1 e 4.2):
Cada um destes nós tem funções diferentes, sendo o nó *GNS_OS_Controller* para atuar como nó *controller* de toda a solução OpenStack, o nó *GNS_OS_Network* como controlador das

Nome nó	OS	RAM	CPU	Disco(s)
GNS_OS_Controller	Ubuntu Server 16.04.2	4096Mb	2 CPU virtual	20Gb
GNS_OS_Network	Ubuntu Server 16.04.2	2048Mb	2 CPU virtual	20Gb
GNS_OS_Compute	Ubuntu Server 16.04.2	2048Mb	2 CPU virtual	20Gb
GNS_OS_BlockStorage	Ubuntu Server 16.04.2	2048Mb	2 CPU virtual	2x20Gb
GNS_ODL_Controller	Ubuntu Server 16.04.2	2048Mb	2 CPU virtual	20Gb

Tabela 4.1: Características das VMs do cenário de testes - Parte 1

Nome nó	Interfaces de rede
GNS_OS_Controller	1
GNS_OS_Network	3
GNS_OS_Compute	3
GNS_OS_BlockStorage	1
GNS_ODL_Controller	1

Tabela 4.2: Características das VMs do cenário de testes - Parte 2

redes virtuais do serviço de OpenStack e o nó *GNS_OS_Compute* como *host* para instâncias criadas através do serviço de OpenStack. Para além destes três tipos de nó considerados essenciais para o correto funcionamento da solução, também foi configurado um nó do tipo *Block Storage* denominado de *GNS_OS_BlockStorage*, de modo a existir a possibilidade de criar e associar volumes lógicos virtuais às VMs criadas. De modo a demonstrar a interoperabilidade desta solução, foi também criado um nó designado de *GNS_ODL_Controller* que irá conter o controlador externo de OF.

Ao nível do *hypervisor* escolhido para correr no nó *compute*, foi escolhido o QEMU. Apesar das limitações de usar uma ferramenta bastante limitada, é a tecnologia suportada devido ao uso de *nested virtualization* no cenário implementado.

Com a escolha desta arquitetura, é possível desenvolver vários testes sem sobrecarregar em demasia o *host* de testes. Também é possível posteriormente adicionar outros nós, como por exemplo, mais nós *compute*, *block/object storage*, nó(s) de redundância, entre outros. É também importante referir que a atual arquitetura não tem qualquer implementação de redundância ou *disaster ricovery* na solução de OpenStack devido à limitação de recursos disponíveis no *host* de testes. É de salientar que é possível e suportado pelo OpenStack a configuração de cenários de redundância e *disaster recovery*.

4.4.5 Instalação da arquitetura OpenStack

De modo a simplificar todo o processo de instalação e automatizar todo o processo, foram criados vários *scripts* de instalação. Estes *scripts* requerem no mínimo três nós, com acesso à Internet e com o OS Ubuntu Server 16.04. Estes *scripts* estão disponíveis no anexo B.
Todo este processo foi pensado de modo a retirar todo o *overhead* ao utilizador de configuração de serviços, alteração de ficheiros, entre outros. Devido então a este processo, os *scripts* estão organizados por tipo de serviço, ou seja, *controller*, *network*, *compute* e *block storage*. Estão também preparados para serem executados individualmente ou em cadeia, traduzindo para o utilizador todo o processo a decorrer. Com esta automatização é possível:

- Configuração automática das interfaces de rede;
  - Interface de acesso à Internet e gestão do OpenStack (necessário em todos ogs nós);
  - Interface de gestão de redes virtuais do OpenStack (necessário em nós do tipo *network* e *compute*);
  - Interface com rede e IPs públicos a disponibilizar às VMs para acesso remoto, entre outros (necessário em nós do tipo *network* e *compute*).
- Configuração automática do ficheiro de *hostname* do nó;
- Configuração automática do ficheiro de *hosts* do nó;
- Configuração de *passwords* no ato de instalação;
- Configuração de requisitos para o OpenStack;
- Configuração do OpenStack e os módulos especificados com base nas entradas anteriores do utilizador.

Para simplificar a compreensão e organização dos mesmos, estes estão organizados por fases:

- Tipo de nó (*controller*, *network*, *compute* e *block storage*):
  - *Pre-installation*;
  - *Installation*;
  - *Modules*.

Com esta organização pretende-se que as várias fases de instalação estejam bem delineadas e claras. Pré instalação para preparar todo o sistema para a instalação de OpenStack, como por exemplo, a configuração das interfaces de rede. A instalação onde todo o OpenStack e os módulos escolhidos pelo utilizador serão instalados, configurados e testados.

Hierarquia dos *scripts*

Depois de explicada a organização dos *scripts*, irá ser explicada a hierarquia e interligação dos mesmos.
Todos os *scripts* disponibilizados podem ser executados de forma individual. Esta abordagem foi considerada devido ao fato de o utilizador poder desejar ter um controlo mais fino sobre a instalação. Porém, existe um tipo de *script* que não tem qualquer ligação com outros, este *script* é:

- Tipo de nó (*controller*, *network*, *compute* e *block storage*):
  - *Pre-installation*

Este tipo de *script* não tem qualquer ligação com outro devido às funções que desempenha. Devido a efetuar atualizações ao sistema, incluindo ao *kernel* do mesmo, é necessário um reinicio do nó para uma correta instalação do OpenStack.
Tal como o nome indica, este *script* deverá ser o primeiro a ser executado, uma vez que os restantes serviços dependem deste, como por exemplo, a configuração das interfaces de rede.
Relativamente a todos os outros, estes podem ser executados de modo interligado ou de modo individual. O modo interligado tem ainda quatro modos de instalação:

- *all* - Instala o serviço base de OpenStack, as suas dependências, os módulos essenciais (Keystone, Glance, Nova, Neutron e Horizon) e os módulos opcionais (Cinder);
- *base* - Instala apenas serviço base de OpenStack e as suas dependências, não instalando qualquer módulo;
- *required* - Instala o serviço base do OpenStack, as suas dependências e os módulos essenciais (Keystone, Glance, Nova, Neutron e Horizon);
- *optional* - Instala os módulos de OpenStack considerados opcionais (Cinder), sendo requerido a instalação do serviço base do OpenStack e os seus módulos essenciais.

É de salientar que é imperativo manter a ordem de instalação dos módulos caso se opte por instalação individual. Esta ordem é necessária devido ao fato de certos módulos dependerem de outros, podendo assim causar erros na instalação e no correto funcionamento do OpenStack. Esta ordem é respeitada caso se opte pela execução interligada. A ordem correta é:

- *Controller*
  - *Pre-installation*;
  - *Installation*
    - Núcleo do OpenStack;
    - Serviços dos quais o OpenStack depende (DB, entre outros);
  - *Modules*;
    - Keystone;
    - Glance;
    - Nova;
    - Neutron;
    - Horizon;
    - Cinder.
- *Network*
  - *Pre-installation*;
  - *Installation*
    - Núcleo do OpenStack;
  - *Modules*;
    - Neutron.
- *Compute*
  - *Pre-installation*;
  - *Installation*
    - Núcleo do OpenStack;
  - *Modules*;
    - Nova;

        * Neutron.

- *Block Storage*

    - *Pre-installation*;
    - *Installation*
        * Núcleo do OpenStack;
    - *Modules*;
        * Cinder.

A instalação dos módulos varia em cada tipo de nó (*Controller*, *Network*, *Compute* e *Block Storage*).

4.5 Implementação controlador de *OpenFlow* (OF)

Como já foi referido anteriormente, irá ser implementada uma solução de controlador *OpenFlow* (OF), de modo a ser possível controlar, aplicar regras e outros a toda a rede virtual do OpenStack e as suas sub-redes.

Para este fim, foi escolhido o controlador de OF OpenDaylight devido à sua integração nativa com o OpenStack e também devido a ser um controlador com bastantes funcionalidades e um desenvolvimento estável e contínuo.

Para a instalação do controlador, poderá ser criada uma nova VM para o efeito ou usar uma já existente.

Antes da instalação do controlador de OF e toda a sua interligação com o OpenStack, é necessário preparar o OpenStack para a interligação com o mesmo. Com a interligação do controlador de OF com o OpenStack, é necessário garantir que todas as redes virtuais existentes na infraestrutura se encontram eliminadas. Este processo é necessário, pois ao existir uma nova peça no processo de gestão de rede, é necessário que essa peça esteja ciente de todos os componentes criados (consultar figura 4.11).

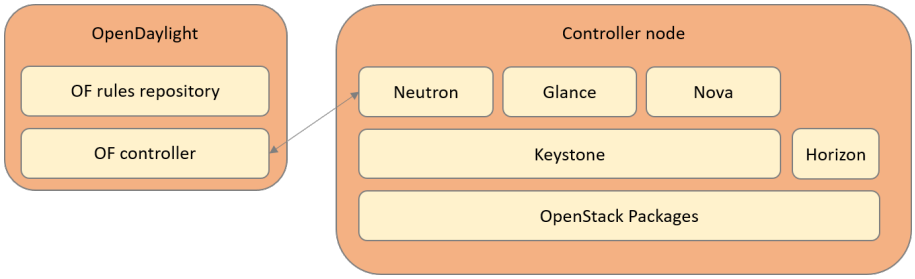

Figura 4.11: Diagrama interligação Neutron - OpenDaylight

Para o âmbito desta dissertação de modo a ser o mais próximo possível de um cenário real, o controlador de OF foi instalado em uma máquina distinta no segmento de rede e no bastidor virtual de controladores, onde também se encontra o controlador da solução de OpenStack.

4.5.1 Configuração controlador de *OpenFlow* (OF)

Como foi referido anteriormente, o OpenDaylight é um controlador de OF em constante desenvolvimento. Para a arquitetura desenvolvida no âmbito desta dissertação, foi escolhida a versão Boron-SR4. O OpenDaylight tem um ciclo de desenvolvimento de 8 meses para cada versão, sendo esta versão atualizada aproximadamente de 2 em 2 meses (versões SRX), lançado uma nova versão a cada 6 meses.
Os requisitos para a execução do controlador variam de versão para versão, sendo estes os requisitos necessários para a execução da versão Boron-SR4:

- Java 1.8.0 ou superior;
- Nós de OpenStack (network e compute) com versão de Open vSwitch superior ou igual a 2.5.0.

É possível transferir do site oficial do OpenDaylight uma versão preparada para ser implementada, sendo apenas necessário efetuar o *unpack* do conteúdo.
Após o conteúdo ser *unpacked*, irá resultar uma diretoria com todos os ficheiros necessários para a execução da ferramenta. Relevantes para a execução da ferramenta são os seguintes ficheiros:

- Diretoria "bin-> ficheiro "start":

  Permite iniciar o serviço, ficando o mesmo à espera de ligações no porto 6640.

- Diretoria "bin-> ficheiro "stop":

  Permite parar o serviço. Não recomendado com solução de OpenStack em execução.

- Diretoria "bin-> ficheiro "status":

  Imprime para o ecrã o estado do serviço.

- Diretoria "bin-> ficheiro "client":

  Efetua a autenticação no serviço usando o utilizador padrão "karaf". Aqui é obtida a consola do serviço onde é possível efetuar várias ações, incluindo a instalação/desinstalação de componentes.

Após os requisitos do serviço estarem garantidos e o conteúdo *unpacked*, basta executar o serviço e efetuar a autenticação no serviço, instalando as componentes necessárias. Estas componentes são:

- odl-netvirt-openstack:

  Módulo de integração com o OpenStack.

- odl-dlux-core:

  Interface web para visualização de agentes registados no serviço (agentes de Open vSwitch). Esta interface encontra-se disponível no seguinte endereço:

  `http://<ip_controladorODL>:8181/index.html`

  Por defeito, as credenciais do serviço são:

  - Utilizador: admin
  - Password: admin

- odl-mdsal-apidocs:

  Documentação de APIs instaladas.

De modo a instalar estas componentes, basta executar os seguintes comandos:

```
cd distribution-karaf-0.5.4-Boron-SR4
./bin/start
./bin/client
feature:install odl-netvirt-openstack odl-dlux-core odl-mdsal-apidocs
```

O tempo de instalação destas componentes irá variar consoante os recursos da máquina.

4.5.2 Integração controlador de *OpenFlow* (OF) com o OpenStack

Esta interligação deverá ser efetuada no processo de configuração do serviço de OpenStack, pois é neste momento que ainda não existe nenhuma configuração de rede criada. Caso já exista, será necessário efetuar os seguintes passos:

- Eliminar possíveis instâncias criadas:

  ```
  nova list
  nova delete <instance names>
  ```

- Eliminar possíveis redes e *routers* criados:

  ```
  neutron router-list
  neutron router-port-list <router name>
  neutron router-interface-delete <router name> <subnet ID or name>
  neutron router-clear-gateway <router name>
  neutron router-delete <router name>
  neutron subnet-list
  neutron subnet-delete <subnet name>
  neutron net-list
  neutron net-delete <net name>
  ```

- Verificar se todas as configurações foram eliminadas:

  ```
  neutron port-list
  ```

  Não deverá retornar nada. Caso retorne, repetir os passos anteriores.

Após todas as configurações de rede estarem eliminadas, é necessário configurar a solução de OpenStack para a utilização do controlador externo de rede. Esta configuração assegura a interligação do OpenStack com o controlador de OF externo. É de salientar a importância da instalação do módulo de OpenDaylight antes de se efetuar qualquer alteração, pois sem este o módulo Neutron da solução de OpenStack não irá ser executado corretamente. Este módulo encontra-se disponível para transferência, sendo necessário executar os seguintes passos para instalar o mesmo (o pacote de instalação varia consoante a versão da solução de OpenStack implementada):

```
git clone https://github.com/openstack/networking-odl -b stable/newton
cd networking-odl
python setup.py install
```

À semelhança da instalação do serviço de OpenStack (consultar secção 4.4.5), de modo a simplificar o processo de toda a configuração do controlador de OF com a solução de OpenStack, foram criados vários *scripts* de configuração para todas os nós de OpenStack que necessitem do módulo de Newton (Controller, Network e Compute).

- *Pre-installation*;
- *Installation*;

Com esta automatização é possível:

- Configuração automática do ficheiro de *hosts* da máquina;
- Configuração do serviço de OpenStack para a interligação com o controlador de OF;
  - Alteração de ficheiros de configuração do serviço (máquina *controller*, *network* e *compute*);
  - Reposição da DB do serviço de Neutron com as novas configurações (máquina *controller*);
  - Configuração do serviço de OvS;
- Configuração de requisitos para OpenDaylight, i.e. a desabilitação dos serviços base do OpenStack que asseguram o controlo de todas as componentes de rede virtual:
  - Agente de Open vSwitch do módulo de Neutron nas máquinas Network e Compute;
  - Agente de encaminhamento de *layer 3* na máquina network.
- Instalação do módulo de OpenDaylight para a solução de OpenStack.
- Configuração da utilização do serviço de OpenDaylight na solução de OpenStack.

Estes *scripts* estão disponíveis no anexo C.
Após a integração dos dois serviços estar finalizada, é possível visualizar a sua integração nas máquinas *network* e *compute*.

```
sudo ovs-vsctl show
```

Com o resultado deste comando será possível observar que o serviço de Open vSwitch está conectado a um controlador externo, este que será o controlador externo de OF, o OpenDaylight.

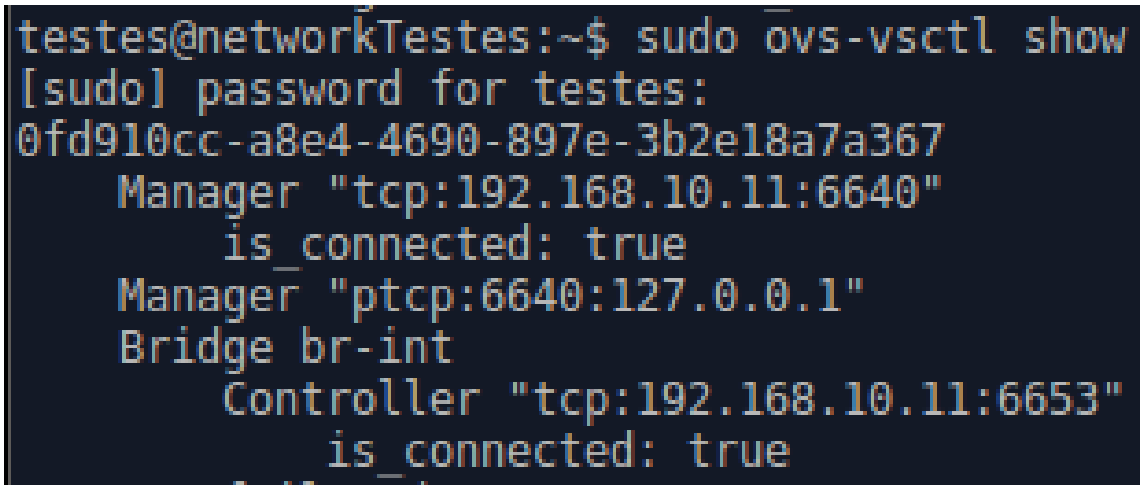

Figura 4.12: Integração de OpenDaylight com o Open vSwitch

4.5.3 Interface gráfica OpenDaylight

O OpenDaylight contém uma interface gráfica contemplada na arquitetura desta dissertação onde é possível visualizar os clientes que estão conectados com o serviço. Estes clientes serão os clientes de Open vSwitch em cada uma das máquinas *network* e *compute*. Como no âmbito desta dissertação só foi possível configurar um de cada devido a limitações de recursos como já foi referido anteriormente, nesta interface gráfica irão surgir dois clientes (consultar figura 4.13).

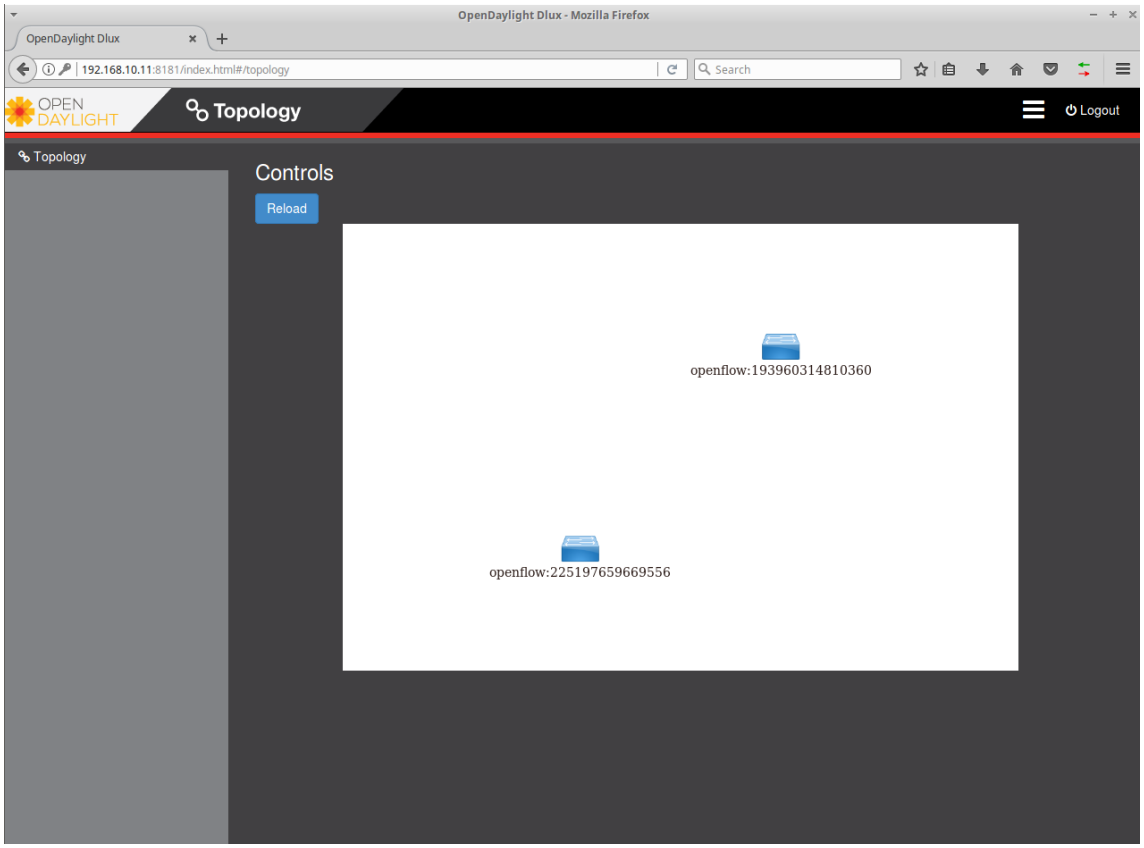

Figura 4.13: Interface gráfica OpenDaylight

Caso sejam adicionados nós adicionais *network* e *compute* à solução de OpenStack, estes serão representados nesta interface gráfica.

4.5.4 cURL

O cURL é uma ferramenta de terminal usada para transferir dados entre dois pontos a partir de uma sintaxe em forma de *Uniform Resource Locator* (URL)[10], suportando vários protocolos[55]. No âmbito desta dissertação, é a ferramenta que possibilita a adição e remoção de regras no nosso controlador. Funciona através de pedidos GET e POST do HTTP utilizando um formato predefinido e conhecido (varia consoante o controlador usado).
Atualmente a ferramenta cURL encontra-se na versão 7.55.1 [55] e contém versões para uma grande variedade de sistemas operativos.

[10]Mais conhecido como endereço de web é uma frase especifica que segue um determinado conjunto de regras que faz referência a um recurso, como uma página web, uma foto alojada na web, etc.[54]

Após toda a configuração de integração do OpenStack com o OpenDaylight estar finalizada, é possível verificar o sucesso da mesma executando o cURL para comparar a informação armazenada no controlador externo de OF com a informação armazenada na DB do módulo de Neutron.

- Controlador externo de OF:

```
curl -u admin:admin \
http://192.168.10.11:8080/controller/nb/v2/neutron/networks
{
   "networks" : [ {
      "id" : "d3ac3a58-e66f-462c-b3af-d58ddc2a5de9",
      "admin_state_up" : true,
      "status" : "ACTIVE",
      "tenant_id" : "e1f59fcc9200456c9e46cbab4103dfb7",
      "name" : "selfservice1",
      "shared" : false,
      "router:external" : false,
      "provider:network_type" : "vxlan",
      "provider:segmentation_id" : "72",
      "segments" : [ ]
   }, {
      "id" : "58edee40-dfee-495f-8340-581a5ec16c59",
      "admin_state_up" : true,
      "tenant_id" : "0093ee8111034a26862af3a0154bf1c1",
      "name" : "provider1",
      "shared" : true,
      "router:external" : true,
      "provider:network_type" : "flat",
      "provider:physical_network" : "provider",
      "segments" : [ ]
   } ]
```

- DB do módulo de Neutron

```
testes@controllerTestes:~$ neutron net-list
+--------------------------------------+-------------+-----------------------------------------------------+
| id                                   | name        | subnets                                             |
+--------------------------------------+-------------+-----------------------------------------------------+
| 58edee40-dfee-495f-8340-581a5ec16c59 | provider1   | 055d10e6-5367-4657-a081-988090b114f5 192.168.3.0/24 |
| d3ac3a58-e66f-462c-b3af-d58ddc2a5de9 | selfservice1| 53f18ecd-a75c-44ae-a09d-3a11578d6608 192.168.100.0/24 |
+--------------------------------------+-------------+-----------------------------------------------------+
```

Figura 4.14: Redes virtuais armazenadas na DB do modulo de Neutron

Como é possível verificar através dos resultados obtidos das duas ferramentas, a informação é idêntica. OS IDs das redes virtuais são idênticos bem como os tipos de rede (provider=provider e vxlan=selfservice).

Capítulo 5

Análise comparativa

Neste capítulo será apresentada uma comparação qualitativa entre a arquitetura implementada e o serviço de IaaS da Microsoft, denominado de Microsoft Azure.
O Microsoft Azure é um serviço *cloud* com foco em serviços SaaS, PaaS e IaaS. À semelhança do OpenStack, o Azure disponibiliza todas as ferramentas necessárias para configurar e gerir uma infraestrutura num ambiente de *cloud* pública.
O Azure foi disponibilizado em 1 de fevereiro de 2010, tendo crescido quer a nível de maturidade quer a nível de funcionalidades. Atualmente conta com utilização de cerca de 90% das empresas da Fortune 500, 42 DCs em diferentes localizações do planeta, sendo atualmente estes DCs os mais certificados a nível de segurança para serviços *cloud* do mundo[56].
Com o objetivo de disponibilizar verdadeiros cenários de *cloud* híbrida, a Microsoft recentemente anunciou o Microsoft Azure Stack, sendo este uma extensão da *cloud* pública de Azure para ambiente *on-premises*[1]. Com o Azure Stack é possível não só gerir um ambiente IaaS, mas também fornecer serviços do Azure na infraestrutura interna, algo que não é possível oferecer com uma implementação híbrida entre System Center Virtual Machine Manager[2] e Azure.
O Azure Stack encontra-se atualmente em versão beta.

5.1 Microsoft Azure Stack vs OpenStack

Neste ponto será apresentada uma comparação qualitativa entre a solução de OpenStack e a solução de Azure Stack. Esta comparação qualitativa terá como base os seguintes aspetos:

[1] *On-premises* é o termo usado para indicar que um recurso se encontra nas instalações.
[2] O System Center Virtual Machine Manager é uma solução de gestão IaaS para *clouds* privadas.

- Instalação e configuração;
- Gestão e manutenção;
- Documentação e comunidade;
- Compatibilidade e extensibilidade;
- Custo e suporte;
- Adoção e aceitação pela indústria.

Para efetuar esta comparação qualitativa teve-se por base o conhecimento adquirido no decorrer da implementação da solução OpenStack no âmbito desta dissertação e a experiência profissional em tecnologias Microsoft, inclusive Microsoft Azure, e a interação com várias organizações de vários setores.
Uma comparação quantitativa entre a solução de OpenStack e de Azure Stack não será apresentada devido aos elevados recursos necessários para executar a solução de Azure Stack num ambiente de testes local. Estes requisitos são:

- 96Gb de RAM;
- 12 cores de CPU físicos;
- Não suporta *nested virtualization*.

5.1.1 Instalação e configuração

Apesar de ambas as plataformas conterem essencialmente as mesmas funcionalidades a nível de IaaS, a instalação e implementação varia consideravelmente. O serviço de OpenStack é bastante mais complexo, quer comparando com o serviço de Azure ou Azure Stack. Sendo o Azure um serviço de *cloud* público, este é gerido por terceiros (Microsoft), retirando assim a complexidade de configuração. O Azure Stack, apesar de ser um serviço *on-premises*, a complexidade de configuração do mesmo é mínima, uma vez que a Microsoft disponibiliza ferramentas para instalar e configurar a ferramenta de forma automática, sendo apenas necessário garantir os requisitos mínimos. É de salientar que o serviço de Azure Stack é exclusivamente um serviço *cloud* híbrido, sendo assim necessário uma subscrição de Azure válida e a interligação da mesma com o Azure Stack para que o serviço funcionar em pleno. Ao contrário, o serviço de OpenStack é bastante complexo de implementar e configurar, devido à sua estrutura modular, conhecimentos de ambientes Linux, de redes de computadores,

ferramentas associadas ao serviço, desde base de dados, *hypervisors*, entre outros. Apesar de existirem várias ferramentas que otimizam este processo, será sempre necessário existirem os conhecimentos necessários dentro da organização para efetuar o *throubleshooting* em caso de erros ou falhas na instalação.

5.1.2 Gestão e manutenção

À semelhança do ponto anterior, a gestão e manutenção do serviço é bastante mais simplificada nos serviços de Azure. Como referido, sendo o serviço de Azure um serviço de *cloud* público, este é gerido por terceiros (Microsoft), retirando assim a complexidade de gestão e manutenção. O Azure Stack devido ao foco da Microsoft em ferramentas de gestão simples e totalmente através de interfaces gráficas, agilizam bastante o processo, uma vez que todo o processo é realizado graficamente, não existindo assim a necessidade de atualizar ficheiros manualmente, atualizar repositórios, etc.
Devido à natureza do serviço de OpenStack e face à liberdade que o mesmo oferece na interligação com serviços/ferramentas externas, diferentes tipos de *hypervisor*, diferentes arquiteturas, entre outros, causa entropia no ato de atualizar, gerir e manter a infraestrutura, uma vez que existem vários fatores a limitar o sucesso, desde compatibilidade entre versões, processos de atualizações, atualizações de repositórios, validação de preferências em ficheiros de configuração antes de atualização, estes poderão mudar de versão para versão, entre outros.

5.1.3 Documentação e comunidade

Para ambas as soluções, a documentação e a comunidade são pontos essenciais a considerar, uma vez que poderão ser os primeiros pontos a consultar ou verificar na configuração e na manutenção da solução.
Quer usando a solução de Azure ou OpenStack, iremos obter bons resultados, pois tanto a comunidade como a documentação é acessível, clara e objetiva. No entanto devido, à natureza da solução de OpenStack, mais concretamente no nível de conhecimento necessário para entender todas as componentes do serviço e as componentes que integram o mesmo, a documentação torna-se insuficiente expondo componentes, configurações e/ou funcionamento das componentes de modo implícito, podendo assim levar à não compreensão do conteúdo exposto. Esta complexidade associada às várias componentes e a interligação entre as mesmas, aliada à natureza do OS (Linux) em qual o serviço OpenStack assenta, poderá ditar o

sucesso da implementação e do ciclo de vida da solução em uma dada organização.
Tendo em conta que o serviço de Azure Stack ainda se encontra em versão beta, apesar do fabricante disponibilizar a devida documentação, não é esperado o mesmo tipo de comunidade e de *feedback*, uma vez que os adotantes da tecnologia atualmente se concentram em cenários de prova de conceito e de validação da tecnologia. É importante também salientar que o Azure Stack ainda não contém todas as suas funcionalidades disponíveis para uso e as que disponibiliza poderão não estar finalizadas ou devidamente funcionais.

5.1.4 Compatibilidade e extensibilidade

Como referido anteriormente, o objetivo da criação do serviço de OpenStack é criar uma plataforma de *cloud computing* totalmente independente do vendedor ou da tecnologia, possibilitando assim liberdade na escolha de tecnologia e de *provider*. No serviço de Azure e de Azure Stack, não existe essa premissa, uma vez que se trata de um serviço proprietário. Um desses exemplos é o único tipo de formato de discos virtuais suportado ser o VHD. Caso queiramos importar um formato diferente, este terá de ser convertido para VHD, antes de qualquer outra operação. Embora este processo seja suportado pelo fabricante e o mesmo disponibilize ferramentas com esse objetivo, é perdida a flexibilidade obtida na solução de OpenStack.
Um outro ponto a focar, é mais uma vez a flexibilidade de suportar qualquer OS (passível de ser virtualizado) na solução de OpenStack. No caso da solução de Azure ou Azure Stack, existirá sempre a limitação dos OSs disponibilizados pelo fabricante. Apesar de ser uma limitação no serviço de Azure ou Azure Stack, estes já oferecem uma vasta oferta de soluções.
Ao nível da extensibilidade é possível focar três pontos:

- Extensibilidade da solução em si;
- Integração da solução com outras ferramentas e/ou soluções;
- Adição de funcionalidades ao serviço através de ferramentas externas.

Extensibilidade da solução em si

Ao referir extensibilidade da solução, é a possibilidade de adicionar funcionalidades ao core da solução com base numa dada necessidade. Devido à natureza *opensource* da solução de OpenStack, é possível efetuar esta extensibilidade com os recursos e o conhecimento certo.

Considerando a solução de Azure ou Azure Stack, não é possível efetuar esta extensibilidade devido à natureza proprietária da solução. Aqui essa função caberá ao fabricante.

Integração da solução com outras ferramentas e/ou soluções

Neste ponto é referido a capacidade de integrar a solução com outras ferramentas e/ou soluções. Esta integração pode ser com base em ferramentas e/ou soluções organizacionais ou de terceiros. Estas ferramentas e/ou soluções irão consumir APIs das soluções de modo a consumir as mesmas.
Tanto o OpenStack como o Azure beneficiam desta integração e oferecem várias APIs para este fim. No caso do Azure Stack em que é sempre necessário interligar o mesmo com uma subscrição de Azure, este irá usufruir das mesmas APIs do Azure.

Adição de funcionalidades ao serviço através de ferramentas externas

Tanto a solução de OpenStack como a solução de Azure/Azure Stack suportam a adição de funcionalidades através da adição de extensões. Com esta adição, é possível adicionar funcionalidades ou serviços que não estão integrados com a solução.
Ambas as soluções dispõem de um ponto central onde é possível listar e observar todas as extensões e adiciona-las à subscrição, se necessário. Embora possam não existir as mesmas extensões nas duas soluções, ambas dispõem de uma lista considerável de extensões.

5.1.5 Custo e suporte

Ao ser dado ênfase à solução de OpenStack, podemos optar por dois caminhos, implementar a solução sem a contribuição adicional de terceiros, ou contratar um serviço externo que implemente e mantenha a solução. No caso da escolha da implementação da solução sem a contribuição adicional de terceiros, nunca será obtido o mesmo nível de suporte, uma vez que não existe uma entidade ao qual recorrer. Irá existir sempre a possibilidade de recorrer diretamente à comunidade do serviço, no entanto não será comparável à opção de contratar um serviço externo, ao qual deverá estar associado um serviço de suporte com determinados *Service Level Agreement* (SLA)s.
No caso do Azure ou do Azure Stack, estes serviços terão associados um dado valor que irá variar consoante o tipo de contrato, SLAs contratados, quantidade, entre outros, mas estará sempre implícito um contrato de suporte.

Considerando uma implementação da solução de OpenStack com a contratação de um serviço externo, é de salientar que esse *provider* não será o fabricante da solução, mas sim um parceiro certificado pelo mesmo, ao contrário do Azure e do Azure Stack em que o suporte é fornecido pelo fabricante.

Comparando o custo das duas soluções, terão sempre de se quantificar os benefícios na implementação de um cenário de *cloud* de modo a ser possível comparar o Opex e Capex. Como é observado e exemplificado no capítulo 2.1.5, a escolha de implementação de um DC numa dada organização traduz-se sempre num investimento inicial considerável, sendo este amortizado ao longo dos anos, e num investimento constante de modo a manter a infraestrutura, ao contrário de uma solução *cloud* em que esse investimento inicial é mínimo comparando, sendo relativamente uniforme no decorrer do tempo. É possível também afirmar que após um período de tempo o valor de *opex* da *cloud* irá ultrapassar o valor de Opex e de Capex estimados inicialmente para um DC local, no entanto um DC local nunca terá a mesma flexibilidade que uma *cloud* oferece.

Em suma, caso a organização em questão tenha a capacidade, quer de recursos humanos quer de conhecimento técnico para manter uma solução com a complexidade do OpenStack, será uma possibilidade implementar e manter a solução, mas caso a organização não tenha essa capacidade tal cenário não é recomendado, sendo sempre uma solução com um contrato de suporte associado um requisito indispensável. Efetuar uma comparação genérica de custo entre as duas soluções não é viável, pois as necessidades variam de organização para organização, tornando o custo um fator secundário quando é necessário responder se uma dada solução responde às necessidades impostas.

5.1.6 Adoção e aceitação pela indústria

Ambas as soluções apresentam uma adoção e aceitação exemplar por parte da indústria, contando por exemplo o Azure com 90% das empresas do Fortune 500 na sua *cloud* e o OpenStack com a sua utilização no core de empresas mundialmente reconhecidas, como é o caso da HP ou a PayPal[56][57]. São ambas consideradas duas soluções *standard* na temática da *cloud*.

É de notar que o serviço de Azure Stack ainda se encontra numa versão de testes, não existindo assim forma de apresentar a sua adoção pela indústria.

5.1.7 Conclusões da comparação qualitativa

Após serem apresentados os vários pontos de comparação, não é possível qualificar de forma clara qual destas soluções é a melhor ou a mais completa a nível de funcionalidades. É sim possível concluir que são duas tecnologias para dois públicos alvos distintos.
O Azure Stack devido à sua vertente de configuração e manutenção mais simplificada, irá apelar a organizações com recursos mais limitados, em que uma solução deste tipo irá adicionar vantagens e simplificar vários processos dentro da organização em vez de adicionar. Por sua vez, o OpenStack irá apelar a organizações com determinados requisitos tecnológicos, ou outros, que não são respondidos pelo Azure Stack, sendo assim necessário optar por um serviço mais flexível e que possa ser adaptado às suas necessidades.
É de salientar que o serviço Azure Stack não suporta uma implementação de *cloud* puramente privada, sendo assim um ponto negativo face ao OpenStack, que suporta os diversos tipos de *cloud* apresentados anteriormente (privada, híbrida e pública).

5.2 *General Data Protection Regulation* (GDPR)

O *General Data Protection Regulation* (GDPR) foi aprovado em 14 de abril de 2016 pelo Parlamento Europeu com o objetivo de reforçar e unificar a proteção de dados para todos os indivíduos da União Europeia. A partir de 25 de maio de 2018 esta nova regulamentação irá entrar oficialmente em vigor após o período de adaptação e preparação de dois anos. Como é possível observar no portal do GDPR[58], esta regulamentação é a maior alteração na proteção de dados pessoais e privados nos últimos 20 anos, resultando em multas consideráveis caso as organizações não se encontrem em conformidade.

> *The EU General Data Protection Regulation (GDPR) is the most important change in data privacy regulation in 20 years (...) organizations in non-compliance will face heavy fines.*

Esta é uma regulamentação que irá afetar não só organizações e indivíduos sediados na União Europeia, mas também organizações e indivíduos que efetuem negócios na União Europeia. De um modo geral, esta regulamentação trará alterações em quatro pontos chave no normal funcionamento das organizações[59]:

- Privacidade pessoal;
- Monitorização e alarmística;

- Políticas transparentes;
- Alterações na atual hierarquia da organização.

Apesar da regulamentação ser baseada em processos e em obrigações, estes terão de ser suportados por tecnologia, ficando a cargo da organização a escolha da tecnologia, uma vez que a regulamentação é agnóstica à tecnologia.

Privacidade pessoal

Com a entrada do GDPR um indivíduo terá poder de decisão sobre os seus dados pessoais, sendo assim a organização em questão, obrigada a ter mecanismos implementados para tal. O indivíduo terá direito a:

- Aceder e visualizar os seus dados pessoais;
- Corrigir erros nos seus dados pessoais;
- Exportar os seus dados pessoais;
- Negar o processamento dos seus dados pessoais;
- Apagar definitivamente os seus dados pessoais.

Apesar de vários destes pontos já estarem implementados em várias organizações e em vários serviços, obrigará à criação e/ou adaptação de novos processos internos, uma vez que estes direitos são transversais à organização e aos seus serviços. Isto é, caso um indivíduo expresse o desejo de eliminar os seus dados pessoais de um dado serviço, estes terão de ser eliminados em todas as componentes da infraestrutura da organização e do serviço, contando por exemplo com sistemas de backups, ou seja, a organização terá de garantir a implementação de processos que garantam a indexação, a modificação e a eliminação transversalmente. É de salientar que todos estes processos não envolvem apenas serviços que tenham como alvo utilizadores finais, mas sim qualquer informação pessoal, profissional ou pública a cargo de uma organização. Como exemplo, poderá ser um currículo entregue a uma dada organização, que contará com informações pessoais, como é o caso do nome, da data de nascimento, número de telefone, email, entre outos.
Como é possível observar no portal do GDPR[58]:

> *(...) personal data is any information relating to an individual, whether it relates to his or her private, professional or public life. It can be anything from a*

name, a home address, a photo, an email address, bank details, posts on social networking websites, medical information, or a computer's IP address.

Monitorização e alarmística

Para além de garantir direitos sobre os dados pessoais de um dado indivíduo, a organização terá também a obrigação de garantir que esses dados estão protegidos consoante os *standard* da indústria e que possui as ferramentas necessárias para monitorizar a sua infraestrutura e detetar possíveis vulnerabilidades e ataques. Com base nestes pontos a organização terá:

- Proteger os dados pessoais com os níveis de segurança adequados;
- Notificar as autoridades competentes em caso de ataques e de dados comprometidos;
- Notificar coerentemente e explicitamente os indivíduos do objetivo do processamento e uso dos seus dados pessoais;
- Relatórios detalhados sobre os dados pessoais processados.

Ao ser referida a proteção de dados pessoais, não é apenas garantir que estes estão devidamente encriptados ou que as DBs onde estão armazenadas estão devidamente seguras, mas também o acesso aos mesmos, isto é, uma vez que o paradigma de perímetro de segurança de uma organização já não é apenas a *firewall* da organização, mas também as credenciais do colaborador, é por exemplo necessário garantir que apenas colaboradores com as devidas permissões poderão aceder aos dados e garantir que não existem acessos não autorizados.
Para a monitorização, é esperada a capacidade das organizações monitorizarem a sua infraestrutura sendo possível, no caso de deteção de ataque/vulnerabilidade que cause fuga de informação, detetar a causa, se a mesma foi alargada para outros sistemas (movimento lateral) e mitigar a(s) falha(s). Com esses resultados, será necessário apresentar relatórios às entidades competentes no prazo de 72 horas.
Para a proteção da informação pessoal de um dado indivíduo, é necessário informar atempadamente e de forma clara ao indivíduo, os objetivos do processamento dessa informação, mantendo relatórios detalhados desse processamento.

Políticas transparentes

Este ponto surge no seguimento dos dois anteriores, complementando-os. Como referido nos pontos anteriores, a organização terá de fornecer ao indivíduo os seus objetivos na obtenção e processamento da sua informação pessoal, focando em:

- Fornecer aviso claro sobre a coleta de dados;
- Identificar os processos associados ao processamento dos dados pessoais através de tópicos relevantes e casos de uso;
- Definição e identificação de políticas de retenção e eliminação de dados pessoais.

Como referido nos pontos anteriores, a organização terá de informar atempadamente e de forma clara ao indivíduo, os objetivos do processamento da sua informação pessoal, informando também o utilizador as suas políticas de retenção e de eliminação dos dados, ou seja, por exemplo caso os dados sejam apagados após um período temporal de inatividade.

Alterações na atual hierarquia da organização

Ao serem introduzidas novas regulamentações que trarão impacto nos processos de uma organização, é sempre necessário garantir a formação das equipas afetadas, rever atuais processos e aplicar as devidas remediações. O GDPR não é exceção levando à reestruturação da infraestrutura das TI, alterações no departamento jurídico, financeiro, comercial, entre outros.

> ***NÃO HÁ COMO ESCAPAR-LHE*** *- O Regulamento Europeu Geral de Proteção de Dados (GDPR) está a chegar e, independentemente de qual for o seu papel na empresa onde trabalha, quer seja em RH ou Marketing, Juridico, Segurança de informação e TI, este irá afetá-lo e terá um impacto no seu dia de trabalho. Se manusear, reter ou utilizar dados pessoais, quer sejam informações de funcionários ou informações de clientes, potenciais e atuais, o GDPR vai trazer mudanças nas suas práticas de trabalho e cabe-lhe a si implementá-las.* - Kaspersky[60]

Este impacto deve-se essencialmente a:

- Formação a todos os colaboradores envolvidos;
- Auditar atuais processos e atualizá-los de acordo com a nova regulamentação;
- Definição do cargo de *Data Protection Officer* (DPO) em organizações com mais de 250 colaboradores ou autoridades públicas[61];
- Auditar fornecedores, parceiros, entre outros.

Apesar da formação de todos os colaboradores envolvidos ser uma prática comum, é importante salientar a importância deste ponto para a correta implementação das novas políticas e de modo a garantir a conformidade da organização perante a regulamentação.
A nomeação de um DPO é algo essencial, uma vez que este será o indivíduo responsável por garantir a conformidade da organização perante a regulamentação, auditar os atuais processos, entre outros. O DPO será também responsável por assegurar que os devidos fornecedores, parceiros, entre outros, estarão também em conformidade com o GDPR. Um bom exemplo é o caso em que a informação da organização (DB, VMs, etc) se encontra alojada em serviços externos. Neste caso cabe à organização perceber se estes serviços estão em conformidade com o GDPR, uma vez que a não conformidade pode levar à aplicação de multas e sanções.

5.3 O *General Data Protection Regulation* (GDPR), o OpenStack e o Azure

Com a entrada do GDPR em vigor no dia 25 de maio de 2018, esta será uma realidade para todas as organizações que efetuem negócios em território europeu. A não conformidade com esta regulamentação poderá resultar em multas e sanções elevadíssimas, podendo chegar a 4% do valor da faturação de uma organização ou 20 milhões de euros, dependendo de qual for maior. Este é um valor que poderá ditar a continuidade de uma organização caso a sanção seja aplicada, sendo atualmente um dois principais pontos (ou até o principal ponto) a aplicar nas organizações.

> *Como um treinador pessoal, a Kaspersky Lab pode ajudar a definir um plano de ataque, para que a sua equipa fique em boas condições e esteja pronta para o GDPR.* - Kaspersky[60]

> *You can count on the fact that Google is committed to GDPR compliance across G Suite and Google Cloud Platform services. We are also committed to helping our customers with their GDPR compliance journey by providing them with the robust privacy and security protections we have built into our services and contracts over the years.* - Google[62]

> *We are committed to GDPR compliance across our cloud services when enforcement begins May 25, 2018, and provide GDPR related assurances in our contractual commitments.* - Microsoft[59]

> *At AWS, security, data protection, and compliance are our top priorities, and we will continue to work vigilantly to ensure that our customers are able to enjoy the benefits of AWS securely, compliantly, and without disruption in Europe and around the world. As we head toward May 2018, we will share more news and resources with you to help you comply with the GDPR.* - Amazon AWS[63]

Estas são apenas 4 afirmações de 4 *providers* distintos de serviços *cloud* onde se pode observar que a tendência e o objetivo é apenas um - a conformidade com o GDPR.
Analisando o serviço de Azure e todas as suas componentes, é possível, por exemplo, observar:

- Os DCs da Microsoft são os mais certificados, comparando com outros *providers* de serviços semelhantes.

  Dependendo do tipo de negócio de uma dada organização, a conformidade com certas certificações poderá ser um fator decisivo.

- Plataforma de gestão de identidades (Azure Active Directory) em conformidade com as novas regulamentações do GDPR:

  - Disponibilização de mecanismos fortes de autenticação;
  - Monitorização e gestão de identidades;
  - Alarmística baseada em mecanismos de análise comportamental;
  - Entre outros.

  Serviços paralelos ao Azure, como é o caso do Office 365 ou do Windows 10, beneficiam dos mesmos avanços e das mesmas tecnologias implementadas para a conformidade com o GDPR.

Estes são apenas alguns exemplos dos esforços da Microsoft nesta temática, mas é também possível verificar que o objetivo da Microsoft é oferecer ao mercado um conjunto de serviços chave na mão, que permitem aos seus clientes construir e manter uma infraestrutura sem a preocupação constante de manter essa infraestrutura em conformidade e atualizada, uma vez que alguém o faz por eles.
No caso do OpenStack, esta abordagem não é possível, uma vez que toda a infraestrutura terá de ser mantida pela organização. Um bom exemplo desta temática, é a procura do termo GDPR no site oficial do OpenStack. São obtidos seis resultados sobre esta temática (consultar figura 5.1) e em nenhum destes resultados é obtida uma clarificação de como manter

a infraestrutura em conformidade. Um outro bom exemplo, é a implementação de autenticação forte na infraestrutura de OpenStack. Como é possível observar na documentação do OpenStack:

> *(...) The Identity service supports external authentication services through the Apache web server that can provide this functionality. Servers may also enforce client-side authentication using certificates.*

À semelhança de outros pontos do OpenStack, esta é uma realidade possível de se aplicar, mas será sempre necessário existir na organização as competências para implementar e manter mais um serviço. Esta é uma problemática transversal a todo o serviço de OpenStack.

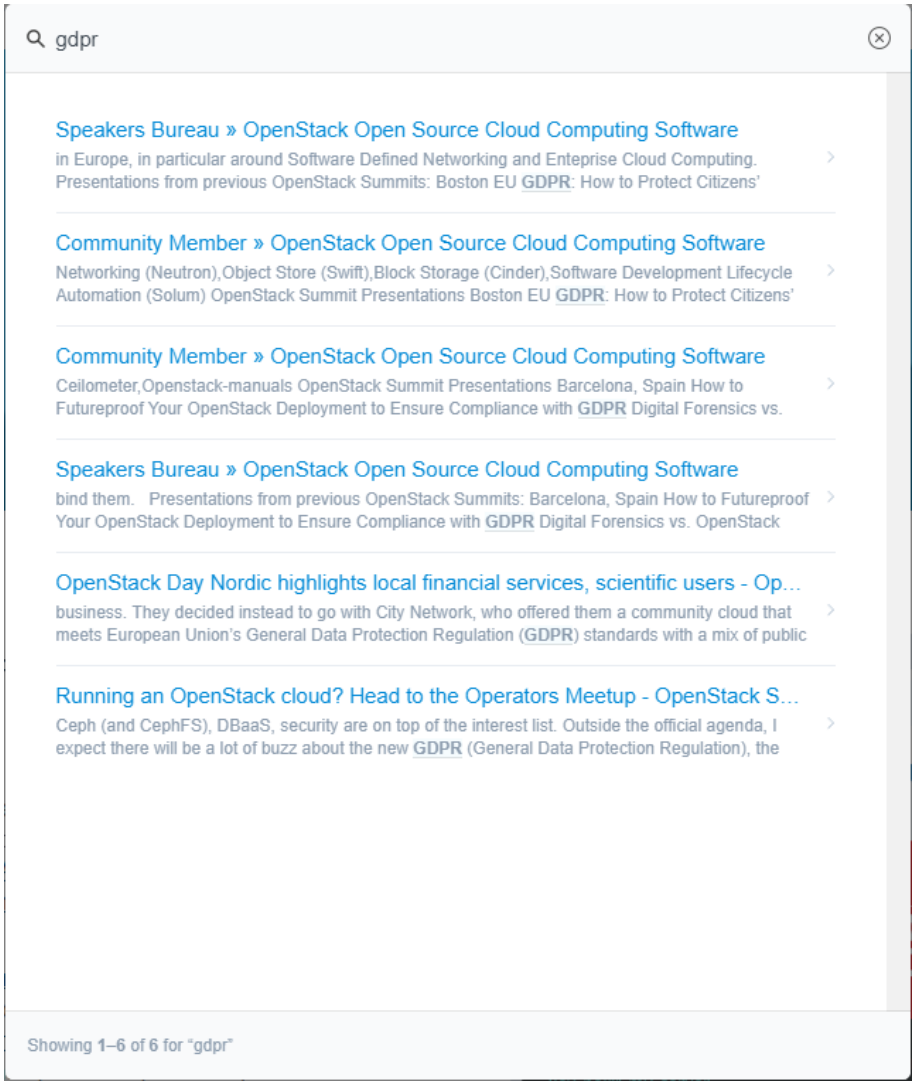

Figura 5.1: Resultados GDPR documentação OpenStack

Capítulo 6

Conclusão

Nesta dissertação é abordada a temática do uso de tecnologias não *standard* e comerciais nos diversos *providers* de serviços *cloud* disponíveis atualmente no mercado. É também abordada, e demonstrada, uma solução criada em resposta a esta temática, o OpenStack. Para além de serem abordados os prós e os contras desta solução, é também efetuada uma comparação qualitativa com um dos principais *providers* de serviços *cloud*, o Microsoft Azure. Concluímos que o OpenStack se trata de uma tecnologia que desempenha as funções que disponibiliza de modo exemplar, garantido um nível equiparável de funcionalidades, quer em qualidade quer em quantidade, quando comparado com os principais *providers* de serviços *cloud* com tecnologias proprietárias, e que aplica tecnologias *standard* nos seus diversos módulos, garantindo assim interoperabilidade com outros serviços similares. Esta interoperabilidade é demonstrada no âmbito desta dissertação com a integração do controlador de OF OpenDaylight com a solução de OpenStack. O sucesso desta tecnologia é comprovado pelo uso diversificado e pela aceitação da mesma pelo mundo empresarial.

Um dos outros objetivos traçados para esta dissertação, seria a criação de mecanismos de configuração da solução OpenStack, de modo a ser possível configurar a mesma sem a necessidade de conhecer detalhadamente a solução ou a complexidade associada à mesma. Este objetivo foi concluído, tendo sido apresentados no decorrer desta dissertação esses mesmos mecanismos. No entanto, à semelhança da integração do controlador OpenDaylight com o OpenStack e como foi referido ao longo desta dissertação, o OpenStack é uma solução bastante complexa, levando a que a sua implementação por vezes não seja simples. Podemos observar esta complexidade na implementação da solução, bem como na integração com o controlador OpenDaylight, não só pelas várias peças que representam a solução, mas também pela documentação escassa e muitas vezes incompleta no sentido em que toma certos

tópicos como garantidos por parte do implementador. Um destes exemplos é a não identificação da necessidade da instalação do módulo de OpenDaylight para a solução de OpenStack na documentação do OpenDaylight, levando assim a uma implementação incompleta da tecnologia, trazendo vários constrangimentos na execução da mesma.
Apesar da solução de OpenStack se apresentar como um *standard* no mercado e uma solução de referência no mundo do *opensource*, consideramos que esta é uma solução para um nicho de mercado. Esta consideração deve-se à complexidade associada à manutenção do serviço e aos recursos necessários envolventes, levando a que a organização em causa tenha total consciência sobre os riscos e esforço necessários para operar a solução. Uma outra consideração é o facto do OpenStack se tratar das poucas soluções disponíveis no mercado totalmente baseada em tecnologias e protocolos *standard* capaz de acomodar uma infraestrutura *cloud* num modelo puramente privado. Consideramos também que não existe uma tecnologia proprietária que se equipare ao OpenStack capaz de acomodar uma *cloud* puramente privada. Tecnologias como o Microsoft Azure Stack, apesar de terem como objetivo a sua instalação na infraestrutura local da organização, tem como principal foco a implementação de um cenário *cloud* híbrido. Será sempre necessário a sua ligação com o seu equivalente público de modo a ser possível implementar a solução. No caso do OpenStack, este não requer qualquer ligação com uma *cloud* pública, sendo possível criar vários sites geograficamente distintos de modo a obter um cenário de redundância como o Microsoft Azure oferece.
Historicamente, e tendo em conta o que o mercado empresarial já demonstrou, uma grande percentagem do sucesso de uma dada solução não é diretamente proporcional com a quantidade de funcionalidades que oferece, com a sua qualidade ou com o facto de mitigar ou não uma problemática existente no segmento em questão, mas sim se essa solução soluciona uma determinada problemática levantada pelos seus clientes. Desta forma, e tendo em conta a entrada em vigor do GDPR, a principal preocupação no mundo empresarial, será a conformidade com esta regulamentação.
Olhando para o OpenStack, podemos concluir que esta solução não se enquadra nesta tendência histórica e que as organizações consideradas comuns, em que os seus recursos, quer humanos, quer tecnológicos, já são limitados e muitas das vezes sobrecarregados com as atuais tecnologias, não irão considerar uma implementação complexa que irá necessitar de recursos e competências extra. Consideramos que o paradigma das atuais pequenas e médias empresas portuguesas se enquadra neste prisma, concluindo assim que a implementação de OpenStack poderá não ser a mais assertiva para esta realidade. No entanto, caso seja imperativo implementar uma solução privada de *cloud*, consideramos que o OpenStack se trata de uma solução exemplar, tal como demonstrado pela adoção do mercado.

Bibliografia

[1] Estatísticas facebook. acedido a 23/09/2017 em http://www.statisticbrain.com/facebook-statistics/.

[2] *Global Cloud Index* (gci). acedido a 23/09/2017 em http://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/index.html#~Forecast.

[3] *Data Center tiers image*. acedido a 23/09/2017 em http://www.eci.com/blog/images/8-3-10_Data-Center-Tiers.gif.

[4] Imagem arquitetura SDN. acedido a 23/09/2017 em https://www.sdxcentral.com/wp-content/uploads/2015/03/sdn-architecture.png.

[5] iSCSI image. acedido a 23/09/2017 em https://docs.oracle.com/cd/E37670_01/E41138/html/images/iSCSI.png.

[6] Imagem protocolos SDN. acedido a 23/09/2017 em http://4.bp.blogspot.com/-96O1GgT9jbQ/VeQYE6Kh5SI/AAAAAAAAGSk/wkApt_SA8VQ/s1600/photo4.jpg.

[7] *Release notes* floodlight. acedido a 23/09/2017 em https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+v1.2.

[8] Guia de instalação de openstack em ambiente ubuntu. acedido a 23/09/2017 em http://docs.openstack.org/newton/install-guide-ubuntu/.

[9] António Miguel Ferreira. *Introdução ao Cloud Computing*, chapter 2, pages 13–22. FCA - Editora de Informática, Lda, março 2015.

[10] *Cloud Computing*. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Cloud_computing.

[11] *Packet Switching vs Circuit Switching*. acedido a 23/09/2017 em http://www.computerworld.com/article/2593382/networking/networking-packet-switched-vs-circuit-switched-networks.html.

[12] *Cloud Computing* pela ibm. acedido a 23/09/2017 em http://www.ibm.com/cloud-computing/us/en/what-is-cloud-computing.html.

[13] Breve história dos *Data Centers*. acedido a 23/09/2017 em http://www.rackspace.com/blog/datacenter-evolution-1960-to-2000/.

[14] *The NIST Definition of Cloud Computing*. acedido a 23/09/2017 em http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[15] *SMBs that embrace Cloud enjoy more revenue: MYOB*. acedido a 23/09/2017 em http://www.arnnet.com.au/article/459981/smbs_embrace_cloud_enjoy_more_revenue_myob_/.

[16] *SMBs losing billions due to ineffective IT management*. acedido a 23/09/2017 em http://www.dynamicbusiness.com.au/news/smbs-losing-billions-due-to-ineffective-it-management-24042013.html.

[17] *Cloud Brokers*. acedido a 23/09/2017 em http://searchcloudprovider.techtarget.com/definition/cloud-broker.

[18] IP. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Internet_Protocol.

[19] DNS. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Domain_Name_System.

[20] DHCP. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol.

[21] Mauricio Arregoces and Maurizio Portolani. *Data Center Fundamentals*. Number ISBN: 1-58705-023-4. Cisco Press, Cisco Press, 800 East 96th Street, Indianapolis, IN 46240 USA, December 2003.

[22] *Data Center tiers*. acedido a 23/09/2017 em https://pt.uptimeinstitute.com/TierCertification/ e http://www.tia-942.org/content/162/289/About_Data_Centers.

[23] CPU. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Central_processing_unit.

[24] RAM. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/Random-access_memory.

[25] P. Vijaya Vardhan Reddy. Evaluation of different hypervisors performance in the private cloud with sigar framework. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 5(2):60–66, 2014.

[26] João Pedro Leal Santos João Vitória Silva. *Gestão de rede recorrendo a Software Defined Networking e Openflow - uma abordagem Opensource*. PhD thesis, Escola Superior de Tecnologia e Gestão Instituto Politécnico de Leiria, julho 2014.

[27] Documentação openstack. acedido a 23/09/2017 em http://docs.openstack.org.

[28] Breve introdução ao openstack. acedido a 23/09/2017 em urlhttp://opensource.com/resources/what-is-openstack.

[29] Marketplace openstack. acedido a 23/09/2017 em http://www.openstack.org/marketplace/.

[30] Gplv3. acedido a 23/09/2017 em http://www.gnu.org/licenses/quick-guide-gplv3.html.

[31] Aslv2. acedido a 23/09/2017 em http://www.apache.org/licenses/LICENSE-2.0.

[32] Aslv2 vs gplv3. acedido a 23/09/2017 em http://www.apache.org/licenses/GPL-compatibility.html.

[33] Bo Yuan Aaron Paradowski, Lu Liu. Benchmarking the performance of openstack and cloudstack. *IEEE 17th International Symposium on Object/Component-Oriented Real-Time Distributed Computing*, 2014.

[34] Rfc 3720 - iSCSI. acedido a 23/09/2017 em https://tools.ietf.org/html/rfc3720.

[35] Rfc 3721 - iSCSI naming and discovery. acedido a 23/09/2017 em https://tools.ietf.org/html/rfc3721.

[36] Eui-64. acedido a 23/09/2017 em https://ccie.lol/wp-content/uploads/2016/10/eui64.pdf.

[37] Rfc 4172 - iFCP. acedido a 23/09/2017 em https://tools.ietf.org/html/rfc4172.

[38] Rfc 3821 - FCIP. acedido a 23/09/2017 em https://tools.ietf.org/html/rfc3821.

[39] Rfc 6847 - FCoE. acedido a 23/09/2017 em https://tools.ietf.org/html/rfc6847.

[40] Notas de lançamento do protocolo OF 1.6. acedido em 23/09/2017 em https://www.opennetworking.org/software-defined-standards/specifications/.

[41] *OpenSource* SDN. acedido em 23/09/2017 em http://opensourcesdn.org.

[42] Controladores OF *OpenSource*. acedido a 23/09/2017 em http://opensourcesdn.org/sdn-links/.

[43] Nox *website*. acedido a 23/09/2017 em http://www.noxrepo.org.

[44] Pox wiki. acedido a 23/09/2017 em https://openflow.stanford.edu/display/ONL/POX+Wiki.

[45] Odenos *framework website*. acedido a 23/09/2017 em http://o3project.github.io/odenos/.

[46] Trema *framework website*. acedido a 23/09/2017 em https://trema.github.io/trema/.

[47] Ryu *framework website*. acedido a 23/09/2017 em https://osrg.github.io/ryu/.

[48] Onos wiki. acedido a 23/09/2017 em https://wiki.onosproject.org/display/ONOS15/User%27s+Guide.

[49] Opendaylight *website*. acedido a 23/09/2017 em http://docs.opendaylight.org/en/stable-boron/getting-started-guide/karaf_features.html.

[50] GNS3 *website*. acedido a 23/09/2017 em https://www.gns3.com/software.

[51] Mário Antunes Rodrigo Emiliano. Automatic network configuration in virtualized environment using gns3. *ICCSE 2015*, pages 25–30, julho 2015.

[52] Virtualbox *website*. acedido a 23/09/2017 em https://www.virtualbox.org/.

[53] Documentação devstack. acedido a 23/09/2017 em http://docs.openstack.org/developer/devstack/.

[54] URL. acedido a 23/09/2017 em https://en.wikipedia.org/wiki/URL.

[55] curl. acedido a 23/09/2017 em https://curl.haxx.se/.

[56] Dados microsoft azure. acedido a 23/09/2017 em https://azure.microsoft.com/pt-pt/?&wt.mc_id=AID623294_SEM_&gclid=CjwKCAjw_dTMBRBHEiwApIzn_OoTkDod1v96ts0VahBs0yHb4t_Zg9nDMFANMsRlULz44EGnnjD2RhoCNG8QAvD_BwE.

[57] Página oficial openstack. acedido a 23/09/2017 em https://www.openstack.org.

[58] Portal do GDPR. acedido a 23/09/2017 em http://www.eugdpr.org/.

[59] Pontos chaves no GDPR. acedido a 23/09/2017 em https://www.microsoft.com/en-us/TrustCenter/Privacy/gdpr/default.aspx.

[60] Kaspersky e o GDPR. acedido a 23/09/2017 em https://www.kaspersky.pt/gdpr.

[61] DPO. acedido a 23/09/2017 em http://www.eudataprotectionregulation.com/data-protection-officer.

[62] Google e o GDPR. acedido a 29/08/2017 em https://www.google.com/cloud/security/gdpr/.

[63] Amazon aws e o GDPR. acedido a 23/09/2017 em https://aws.amazon.com/pt/blogs/security/aws-and-the-general-data-protection-regulation/.

Apêndice A

Configurações de equipamentos de rede

Neste anexo encontram-se as configurações dos equipamentos de rede usados na arquitetura definida.
As configurações dos equipamentos de rede estão também disponíveis em `https://github.com/xonaecom/configureopenstack`.

A.1 Equipamento R1

```
Building configuration...

Current configuration : 1426 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
```

```
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
!
ip tcp synwait-time 5
!
interface FastEthernet0/0
 ip address dhcp
 ip nat outside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.0.1 255.255.255.252
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.2.254 255.255.255.0
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet2/0
 no ip address
 ip virtual-reassembly
 shutdown
 duplex auto
 speed auto
!
```

```
router rip
 version 2
 network 192.168.0.0
 network 192.168.2.0
 default-information originate
 no auto-summary
!
no ip http server
no ip http secure-server
ip forward-protocol nd
!
ip nat inside source list NAT interface FastEthernet0/0 overload
!
ip access-list standard NAT
 permit 192.168.0.0 0.0.0.3
 permit 192.168.2.0 0.0.0.255
 permit 192.168.3.0 0.0.0.255
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
end
```

A.2 Equipamento R2

```
Building configuration...

Current configuration : 2313 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
!
ip tcp synwait-time 5
!
interface FastEthernet0/0
 ip address 192.168.0.2 255.255.255.252
 ip nat outside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.1.1 255.255.255.252
 ip nat inside
```

```
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.1.5 255.255.255.252
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.1.9 255.255.255.252
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet3/0
 ip address 192.168.3.254 255.255.255.0
 ip nat outside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet4/0
 ip address 192.168.1.13 255.255.255.252
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.0 0.0.0.3 area 0
```

```
 network 192.168.1.4 0.0.0.3 area 0
 network 192.168.1.8 0.0.0.3 area 0
 network 192.168.1.12 0.0.0.3 area 0
 default-information originate
!
router rip
 version 2
 network 192.168.0.0
 network 192.168.3.0
 no auto-summary
!
no ip http server
no ip http secure-server
ip forward-protocol nd
!
ip nat inside source list NAT interface FastEthernet0/0 overload
ip nat inside source static tcp 192.168.10.1 6080 interface FastEthernet0/0 6080
ip nat inside source static tcp 192.168.10.1 80 interface FastEthernet0/0 80
ip nat inside source static tcp 192.168.10.1 443 interface FastEthernet0/0 443
!
ip access-list standard NAT
 permit 192.168.1.0 0.0.0.3
 permit 192.168.1.4 0.0.0.3
 permit 192.168.1.8 0.0.0.3
 permit 192.168.10.0 0.0.0.255
 permit 192.168.20.0 0.0.0.255
 permit 192.168.30.0 0.0.0.255
 permit 192.168.1.12 0.0.0.3
 permit 192.168.40.0 0.0.0.255
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
```

```
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
!
end
```

A.3 Equipamento R3

```
Building configuration...

Current configuration : 1357 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R3
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
```

```
!
ip tcp synwait-time 5
!
interface FastEthernet0/0
 ip address 192.168.1.2 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.10.254 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.20.254 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.30.254 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet3/0
 ip address 192.168.40.254 255.255.255.0
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
 network 192.168.20.0 0.0.0.255 area 0
 network 192.168.30.0 0.0.0.255 area 0
 network 192.168.40.0 0.0.0.255 area 0
```

```
!
no ip http server
no ip http secure-server
ip forward-protocol nd
!
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
end
```

A.4 Equipamento R4

```
Building configuration...

Current configuration : 1357 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R4
```

```
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
!
ip tcp synwait-time 5
!
interface FastEthernet0/0
 ip address 192.168.1.6 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.10.253 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.20.253 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.30.253 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet3/0
 ip address 192.168.40.253 255.255.255.0
```

```
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.4 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
 network 192.168.20.0 0.0.0.255 area 0
 network 192.168.30.0 0.0.0.255 area 0
 network 192.168.40.0 0.0.0.255 area 0
!
no ip http server
no ip http secure-server
ip forward-protocol nd
!
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
end
```

A.5 Equipamento R5

```
Building configuration...

Current configuration : 1358 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R5
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
!
ip tcp synwait-time 5
!
interface FastEthernet0/0
 ip address 192.168.1.10 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.10.252 255.255.255.0
 duplex auto
 speed auto
!
```

```
interface FastEthernet1/0
 ip address 192.168.20.252 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.30.252 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet3/0
 ip address 192.168.40.252 255.255.255.0
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.8 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
 network 192.168.20.0 0.0.0.255 area 0
 network 192.168.30.0 0.0.0.255 area 0
 network 192.168.40.0 0.0.0.255 area 0
!
no ip http server
no ip http secure-server
ip forward-protocol nd
!
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
```

```
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
end
```

A.6 Equipamento R6

```
Building configuration...

Current configuration : 1359 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R6
!
boot-start-marker
boot-end-marker
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
!
ip cef
no ip domain lookup
!
ip tcp synwait-time 5
!
```

```
interface FastEthernet0/0
 ip address 192.168.1.14 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.10.251 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.20.251 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.30.251 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet3/0
 ip address 192.168.40.251 255.255.255.0
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.12 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
 network 192.168.20.0 0.0.0.255 area 0
 network 192.168.30.0 0.0.0.255 area 0
 network 192.168.40.0 0.0.0.255 area 0
!
no ip http server
no ip http secure-server
```

```
ip forward-protocol nd
!
no cdp log mismatch duplex
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
end
```

Apêndice B

Scripts de instalação OpenStack

Todos os *scripts* estão também disponíveis em
`https://github.com/xonaecom/configureopenstack`.

B.1 *Scripts* nó *controller*

B.1.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function ipValidation() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 \\
      && ${ip[3]} -le 255 ]]; then
        stat=$?
      fi
```

```
  fi
  echo $stat
}

function buildInterfacesFile() {
  echo "# Keep this interface as you see it
# The loopback network interface
auto lo
iface lo inet loopback

# Controller node network interface for internet access
auto enp0s3
iface enp0s3 inet static
  address $OSIPAddress
  netmask $OSNetmask
  dns-nameservers 8.8.8.8 8.8.4.4" >> interfaces
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "  gateway ${OSGateway[0]}" >> interfaces
  else
    value=100
    for i in "${OSGateway[@]}"
  do
  echo "  up ip route add default via $i dev enp0s3 metric $value" >> \\
  interfaces
  value=$((value+100))
done
  fi
}

function buildHostnameFile() {
  echo "$controllerHostname" >> hostname
}

function buildHostsFile() {
  echo "127.0.0.1 localhost
```

```
# 127.0.1.1 $controllerHostname

# Change the name of the hosts to your needs
# controller
$OSIPAddress  $controllerHostname

# network
# 10.0.0.21 network

# compute1
# 10.0.0.31 compute1

# block1
# 10.0.0.41 block1

# object1
# 10.0.0.51 object1

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters" >> hosts
}

function buildConfigFile() {
  echo "[controller]
OSIP=$OSIPAddress
OSNetmask=$OSNetmask" >> ../configFileController.txt
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "OSGateway1=${OSGateway[0]}" >> ../configFileController.txt
  else
    value=1
    for i in "${OSGateway[@]}"
  do
  echo "OSGateway$value=$i" >> ../configFileController.txt
  value=$((value+1))
```

```
done
  fi
echo "ControllerHostname=$controllerHostname
" >> ../configFileController.txt
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "This script will verify your system compability."
echo "It's recommended to use a clean install."
echo
echo "This script will replace your /etc/network/interfaces, /etc/hosts and \\
/etc/network/interfaces files."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
```

```
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Updating system
echo "2 - Updating system"
sleep 2
apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null

# Replacing files
echo "3 - Replacing files"
sleep 2
OSGateway=()

# Replacing /etc/network/interfaces file
echo "3.1 - Replacing /etc/network/interfaces file"
echo "This script, by default, will use the 10.0.0.0/24 network, 10.0.0.1 \\
IPv4 address and three gateways (10.0.0.254, 10.0.0.253 and\\
10.0.0.252) for the controller node management interface."
read -r -p "Use default? [y/N]" userInputResponseOSManagementNetwork
if [[ $userInputResponseOSManagementNetwork =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default network (10.0.0.0/24)"
  OSIPAddress=10.0.0.1
  OSNetmask=255.255.255.0
  OSGateway+=('10.0.0.254')
  OSGateway+=('10.0.0.253')
  OSGateway+=('10.0.0.252')
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the controller node management interface"
    read -r -p "Controller node IP address: " userInputNewControllerIP
```

```
  read -r -p "Netmask: " userInputNewOSNetmask
read -r -p "Multiple gateways? [y/N]" userInputResponseMultiGateways
  if [[ $userInputResponseMultiGateways =~ ^([yY][eE][sS]|[yY])$ ]]; then
    validNumber=true
    while $validNumber; do
      read -r -p "How many gateways? " userInputResponseGatewaysNumber
      if [[ $userInputResponseGatewaysNumber =~ ^-?[0-9]+$ ]]; then
        aux=0
        while [ $aux -lt $userInputResponseGatewaysNumber ]; do
          aux=$((aux+1))
          read -r -p "Controller node gateway address $aux: " \\
          userInputNewControllerGateway
          OSGateway+=("$userInputNewControllerGateway")
        done
        found=0
        for i in "${OSGateway[@]}"
        do
          if [[ $(ipValidation $i) -ne 0 ]]; then
            found=1
          fi
        done
        if [[ $found -eq 0 ]]; then
          if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
          [[ $(ipValidation $userInputNewControllerIP) -eq 0 ]]; then
            validNumber=false
            validInputOS=false
            OSIPAddress=$userInputNewControllerIP
            OSNetmask=$userInputNewOSNetmask
          fi
        fi
      fi
    done
  else
    read -r -p "Controller node gateway address: " \\
    userInputNewControllerGateway
```

```
      if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
      [[ $(ipValidation $userInputNewControllerIP) -eq 0 ]] && \\
      [[ $(ipValidation $userInputNewControllerGateway) -eq 0 ]]; then
         validInputOS=false
         OSIPAddress=$userInputNewControllerIP
         OSNetmask=$userInputNewOSNetmask
         OSGateway+=("$userInputNewControllerGateway")
      fi
    fi
  done
fi
buildInterfacesFile
mv interfaces /etc/network/interfaces

# Replacing /etc/hostname and /etc/hosts files
echo "3.2 - Replacing /etc/hostname and /etc/hosts files"
echo "This script, by default, will use the following hostname \"controller\

read -r -p "Use default? [y/N]" userInputResponseHostname
if [[ $userInputResponseHostname =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default hostname"
  controllerHostname="controller"
else
  read -r -p "Insert the new hostname: " userInputNewHostname
  controllerHostname=$userInputNewHostname
fi
buildHostnameFile
buildHostsFile
mv hostname /etc/hostname
mv hosts /etc/hosts

#  Generating config file
echo "4 - Generating config file"
buildConfigFile
```

```
# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them\\
to the hosts file"
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install OpenStack and its modules without \\
restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \\
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

B.1.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerIP=${myarray[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray <<< "$line"
        controllerHostname=${myarray[1]}
      fi
    fi
  done < "$confFilePath"
```

```
}

function buildMySQLFile() {
  echo "[mysqld]
bind-address = $controllerIP

default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8" >> 99-openstack.cnf
}

function storeServiceModulesDataConfigFile() {
  if [[ "$1" != "optional" ]]; then
    echo "[passwords]
MDBPass=$userInputMDBPass
RMQUser=$userInputRMQUser
RMQPass=$userInputRMQPass" >> "$confFilePath"
    if [[ "$1" != "base" ]]; then
      echo "KeystonePass=$userInputKeystonePass
KeystoneDBPass=$userInputKeystoneDBPass
KeystoneRegion=$userInputKeystoneRegion
KeystoneDomain=$userInputKeystoneDomain
AdminPass=$userInputKeystoneAdminPass
DemoPass=$userInputKeystoneDemoPass
GlancePass=$userInputGlancePass
GlanceDBPass=$userInputGlanceDBPass
NovaPass=$userInputNovaPass
NovaDBPass=$userInputNovaDBPass
NeutronPass=$userInputNeutronPass
NeutronDBPass=$userInputNeutronDBPass
NeutronSharedSecret=$userInputNeutronSharedSecret
HorizonPass=$userInputHorizonPass" >> "$confFilePath"
    fi
```

```
  fi
  if [[ "$1" == "all" ]] || [[ "$1" == "optional" ]]; then
    echo "CinderPass=$userInputCinderPass
CinderDBPass=$userInputCinderDBPass
SwiftPass=$userInputSwiftPass
HeatPass=$userInputHeatPass
HeatDBPass=$userInputHeatDBPass
CeilometerPass=$userInputCeilometerPass
CeilometerDBPass=$userInputCeilometerDBPass" >> "$confFilePath"
  fi
}

function buildAdminFile() {
  echo "export OS_PROJECT_DOMAIN_ID=$userInputKeystoneDomain
export OS_USER_DOMAIN_ID=$userInputKeystoneDomain
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=$userInputKeystoneAdminPass
export OS_AUTH_URL=http://$controllerHostname:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2" >> ../admin-openrc.sh
}

function buildDemoFile() {
  echo "export OS_PROJECT_DOMAIN_ID=$userInputKeystoneDomain
export OS_USER_DOMAIN_ID=$userInputKeystoneDomain
export OS_PROJECT_NAME=demo
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=$userInputKeystoneDemoPass
export OS_AUTH_URL=http://$controllerHostname:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2" >> ../demo-openrc.sh
}
```

```
########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
else
  if [[ "$1" != "required" ]] && [[ "$1" != "all" ]] && [[ "$1" != \\
  "optional" ]]  &&   [[ "$1" != "base" ]] && [[ "$1" != "help" ]]; then
    if [[ "$1" ==  ]]; then
      echo "You need to specify one argument"
    else
      echo "Option -$1- not valid"
    fi
    echo "Rerun script with a valid option (base, optional, required, all \\
    or help)"
    exit
  else
    if [[ "$1" == "help" ]]; then
      echo "Run the script with the following options:"
      echo "  help - Get script options"
      echo "  base - Only OpenStack core"
      echo "  required - OpenStack core and Glance, Horizon, Keystone,\\
       Neutron,  Nova modules"
      echo "  all - OpenStack core and all modules"
      echo "  optional - Only optional modules Cinder"
      exit
    else
      echo "Running script with -$1- enabled"
      echo
      sleep 2
    fi
  fi
fi
```

```
# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install all the components needed for a correct \\
controller OpenStack installation."
if [[ "$1" == "required" ]]; then
  echo "It'll be installed the following OpenStack modules: Glance, Horizon, \\
  Keystone, Neutron and Nova.\n"
else
  if [[ "$1" == "all" ]]; then
    echo "It'll be installed the following OpenStack modules: Cinder, \\
    Glance, Horizon,"
    echo "Keystone, Neutron, Nova and Swift.\n"
  else
    if [[ "$1" == "optional" ]]; then
      echo "It'll be installed the following OpenStack modules: Cinder.\n"
    fi
  fi
fi
echo "This is a controller script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Resuming installation with -$1- option enabled"
else
  exit
fi

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 14.04
```

```
echo "1.1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi

# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
getVariablesFromConfigFile

# Getting user input data and storing it in the config file
echo "1.3 - Getting data and storing it in the config file"
if [[ "$1" != "optional" ]]; then
  adminToken=‘openssl rand -hex 10‘
  userInputMDBPass=‘openssl rand -hex 10‘
  userInputRMQUser="testesRabbit"
  userInputRMQPass=‘openssl rand -hex 10‘
  if [[ "$1" != "base" ]]; then
userInputKeystonePass=‘openssl rand -hex 10‘
userInputKeystoneDBPass=‘openssl rand -hex 10‘
    userInputKeystoneRegion="RegionOne"
    userInputKeystoneDomain="default"
    userInputKeystoneAdminPass="testesAdmin"
    userInputKeystoneDemoPass="testesDemo"
    #userInputKeystoneAdminPass=‘openssl rand -hex 10‘
    #userInputKeystoneDemoPass=‘openssl rand -hex 10‘
    userInputGlancePass=‘openssl rand -hex 10‘
    userInputGlanceDBPass=‘openssl rand -hex 10‘
    userInputNovaPass=‘openssl rand -hex 10‘
```

```
    userInputNovaDBPass=‘openssl rand -hex 10‘
    userInputNeutronPass=‘openssl rand -hex 10‘
    userInputNeutronDBPass=‘openssl rand -hex 10‘
    userInputNeutronSharedSecret=‘openssl rand -hex 10‘
    userInputHorizonPass=‘openssl rand -hex 10‘
  fi
fi
if [[ "$1" == "all" ]] || [[ "$1" == "optional" ]]; then
  userInputCinderPass=‘openssl rand -hex 10‘
  userInputCinderDBPass=‘openssl rand -hex 10‘
  userInputSwiftPass=‘openssl rand -hex 10‘
  userInputHeatPass=‘openssl rand -hex 10‘
  userInputHeatDBPass=‘openssl rand -hex 10‘
  userInputCeilometerPass=‘openssl rand -hex 10‘
  userInputCeilometerDBPass=‘openssl rand -hex 10‘
fi
storeServiceModulesDataConfigFile $1

buildAdminFile
buildDemoFile

if [[ "$1" != "optional" ]]; then
  # Installing Chrony packages
  echo "2 - Installing Chrony NTP server"
  sleep 2
  apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null
  apt-get install chrony -y > /dev/null
  sed -i 's/pool 2.debian.pool.ntp.org offline \\
  iburst/server 0.debian.pool.ntp.org  iburst/' /etc/chrony/chrony.conf
  sed -i '/server 0.debian.pool.ntp.org iburst/a \server 1.debian.pool.ntp.org \\
  iburst' /etc/chrony/chrony.conf
  sed -i '/server 1.debian.pool.ntp.org iburst/a \server 2.debian.pool.ntp.org \\
  iburst' /etc/chrony/chrony.conf
  sed -i '/server 2.debian.pool.ntp.org iburst/a \server 3.debian.pool.ntp.org \\
  iburst' /etc/chrony/chrony.conf
```

```
  #cp chrony.conf /etc/chrony/chrony.conf
  echo "allow 192.168.10.0/24
allow 192.168.20.0/24
allow 192.168.30.0/24" >> /etc/chrony/chrony.conf

  # Restarting Chrony server
  echo "2.1 - Restarting Chrony server"
  service chrony restart

  # Installing OpenStack packages
  echo "3 - Installing OpenStack packages"
  sleep 2
  echo "3.1 - Checking pre-requisites and updating repositories to OpenStack \\
  newton version"
  apt-get install software-properties-common -y > /dev/null
  add-apt-repository cloud-archive:newton -y
  apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null
  echo "3.2 - OpenStack packages"
  apt-get install python-openstackclient -y > /dev/null

  # Installing SQL database packages
  echo "4 - Installing MySQL database packages"
  sleep 2
  #apt-get install debconf-utils -y > /dev/null
  #debconf-set-selections <<< "mysql-server mysql-server/root_password \\
  password $userInputMDBPass"
  #debconf-set-selections <<< "mysql-server mysql-server/root_password_again \\
  password $userInputMDBPass"
  apt-get install mariadb-server python-pymysql expect -y > /dev/null

  echo "4.1 - Creating necessary file - /etc/mysql/conf.d/mysqld_openstack.cnf"
  buildMySQLFile
  mv 99-openstack.cnf /etc/mysql/mariadb.conf.d/99-openstack.cnf
  chown root:root /etc/mysql/mariadb.conf.d/99-openstack.cnf
  chmod 644 /etc/mysql/mariadb.conf.d/99-openstack.cnf
```

```
  echo "4.2 - Restarting MySQL server"
  service mysql restart

  echo "4.3 - Securing MySQL installation - script mysql_secure_installation"

#expect \"Set root password?\"
#send \"y\r\"

#expect \"New password:\"
#send \"$userInputMDBPass\r\"

#expect \"Re-enter new password:\"
#send \"$userInputMDBPass\r\"
  SECURE_MYSQL=$(expect -c "

set timeout 10
spawn mysql_secure_installation

expect \"Enter current password for root (enter for none):\"
send \"\r\"

expect \"Set root password?\"
send \"n\r\"

expect \"Remove anonymous users?\"
send \"y\r\"

expect \"Disallow root login remotely?\"
send \"y\r\"

expect \"Remove test database and access to it?\"
send \"y\r\"

expect \"Reload privilege tables now?\"
```

```
send \"y\r\"

expect eof
")

  echo "$SECURE_MYSQL"

  # Installing RabbitMQ packages
  echo "5 - Installing RabbitMQ packages"
  sleep 2
  apt-get install rabbitmq-server -y > /dev/null
  rabbitmqctl add_user $userInputRMQUser $userInputRMQPass
  rabbitmqctl set_permissions $userInputRMQUser ".*" ".*" ".*"

  # Installing memcached packages
  echo "6 - Installing memcached packages"
  sleep 2
  apt-get install memcached python-memcache -y > /dev/null
  sed -i "s/-l 127.0.0.1/-l $controllerIP/" /etc/memcached.conf
  service memcached restart

  if [[ "$1" != "base" ]]; then
    #Installing OpenStack Keystone module
    echo "7 - Installing Keystone module"
    sleep 2
    ../Modules/./os-controller-identity.sh following

    #Installing OpenStack Glance module
    echo "8 - Installing Glance module"
    sleep 2
    ../Modules/./os-controller-image.sh following

    #Installing OpenStack Nova module
    echo "9 - Installing Nova module"
    sleep 2
```

```
    ../Modules/./os-controller-compute.sh following

    #Installing OpenStack Neutron module
    echo "10 - Installing Neutron module"
    sleep 2
    ../Modules/./os-controller-network.sh following

    #Installing OpenStack Horizon module
    echo "11 - Installing Horizon module"
    sleep 2
    ../Modules/./os-controller-dashboard.sh following
  fi
fi
if [[ "$1" != "required" ]] && [[ "$1" != "base" ]]; then
  #Installing OpenStack Cinder module
  echo "12 - Installing Cinder module"
  sleep 2
  ../Modules/./os-controller-blockStorage.sh following

  #Installing OpenStack Swift module
  #echo "14 - Installing Swift module"
  #sleep 2
  #read -p "Pause: "
  #../Modules/./os-controller-objectStorage.sh following
  #../Modules/Swift/./os-pos-controller-objectStorage.sh following

  #Installing OpenStack Heat module
  #echo "15 - Installing Heat module"
  #sleep 2
  #read -p "Pause: "
  #../Modules/./os-controller-orchestration.sh following

  #Installing OpenStack Ceilometer module
  #echo "16 - Installing Ceilometer module"
  #sleep 2
```

```
  #read -p "Pause: "
  #../Modules/./os-controller-telemetry.sh following
fi

echo "END"
```

B.1.3 *Script* instalação de módulo Keystone

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"adminToken="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      adminToken=${myarray[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerHostname=${myarray1[1]}
      else
        if [[ $line == *"MDBPass="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerMDBPass=${myarray2[1]}
        else
          if [[ $line == *"KeystonePass="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystonePass=${myarray3[1]}
          else
            if [[ $line == *"KeystoneDBPass="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerKeystoneDBPass=${myarray4[1]}
            else
              if [[ $line == *"KeystoneRegion="* ]]; then
```

```
            IFS='=' read -a myarray5 <<< "$line"
            controllerKeystoneRegion=${myarray5[1]}
          else
            if [[ $line == *"KeystoneDomain="* ]]; then
              IFS='=' read -a myarray6 <<< "$line"
              controllerKeystoneDomain=${myarray6[1]}
            else
              if [[ $line == *"AdminPass="* ]]; then
                IFS='=' read -a myarray7 <<< "$line"
                controllerKeystoneAdminPass=${myarray7[1]}
              else
                if [[ $line == *"DemoPass="* ]]; then
                  IFS='=' read -a myarray8 <<< "$line"
                  controllerKeystoneDemoPass=${myarray8[1]}
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePath"
}

function storeServiceModulesDataConfigFile() {
  echo "KeystonePass=$userInputKeystonePass
KeystoneDBPass=$userInputKeystoneDBPass
KeystoneRegion=$userInputKeystoneRegion
KeystoneDomain=$userInputKeystoneDomain
AdminPass=$userInputKeystoneAdminPass
DemoPass=$userInputKeystoneDemoPass" >> $confFilePath
}
```

```
function buildAdminSourceFile() {
  echo "export OS_PROJECT_DOMAIN_NAME=$controllerKeystoneDomain
export OS_USER_DOMAIN_NAME=$controllerKeystoneDomain
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=$controllerKeystoneAdminPass
export OS_AUTH_URL=http://$controllerHostname:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2" >> admin-openrc
}

function buildDemoSourceFile() {
  echo "export OS_PROJECT_DOMAIN_NAME=$controllerKeystoneDomain
export OS_USER_DOMAIN_NAME=$controllerKeystoneDomain
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=$controllerKeystoneDemoPass
export OS_AUTH_URL=http://$controllerHostname:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2" >> demo-openrc
}

function buildKeystoneFile() {
  echo "[DEFAULT]
admin_token = $adminToken
log_dir = /var/log/keystone

[database]
connection = mysql+pymysql://keystone:$controllerKeystoneDBPass@\\
$controllerHostname/keystone

[token]
provider = fernet

[extra_headers]
```

```
Distribution = Ubuntu" >> keystone.conf
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (following) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Keystone OpenStack controller module."
  echo "This is a controller script."
  echo "Change the provided files to your needs."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This Ubuntu version isn't 14.04."
```

```
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Installing keystone prerequisites
if [[ "$1" != "following" ]]; then
  echo "2 - Installing prerequisites"
else
  echo "7.1 - Installing prerequisites"
fi
sleep 2

# Getting user input data and storing it in the config file
confFilePath=$(find / -name configFileController.txt)
if [[ "$1" != "following" ]]; then
  echo "2.1 - Getting user input data and storing it in the config file"
  echo "SERVICE PASSWORDS"
  echo "It is recommended that you use distinct passwords for each module \\
  and service"
  read -r -p "Keystone (Identity) server password: " userInputKeystonePass
  read -r -p "Keystone (Identity) DB password: " userInputKeystoneDBPass
  read -r -p "Keystone (Identity) region name (leave empty for default \\
  (RegionOne)): " userInputKeystoneRegion
  if [[ "$userInputKeystoneRegion" ==  ]]; then
    $userInputKeystoneRegion = RegionOne
  fi
  read -r -p "Keystone (Identity) domain (leave empty for default (default)): \\
  " userInputKeystoneDomain
  if [[ "$userInputKeystoneDomain" ==  ]]; then
    $userInputKeystoneDomain = default
```

```
  fi
  read -r -p "Keystone (Identity) admin user password: " \\
  userInputKeystoneAdminPass
  read -r -p "Keystone (Identity) demo user password: " \\
  userInputKeystoneDemoPass
  storeServiceModulesDataConfigFile
fi
getVariablesFromConfigFile

# Database commands
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating MySQL database for Keystone"
else
  echo "7.1.1 - Creating MySQL database for Keystone"
fi

user=root
database=keystone
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'localhost' IDENTIFIED BY '$controllerKeystoneDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'\%' IDENTIFIED BY '$controllerKeystoneDBPass';"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'localhost' IDENTIFIED BY '\\
$controllerKeystoneDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'\%' IDENTIFIED BY '
$controllerKeystoneDBPass';"
```

```
# Configuring keystone components
if [[ "$1" != "following" ]]; then
  echo "3 - Configuring components"
else
  echo "7.2 - Configuring components"
fi
sleep 2
#buildKeystoneOverrideFile
#mv keystone.override /etc/init/keystone.override

# keystone.conf configuration
apt-get install keystone -y > /dev/null
buildKeystoneFile
mv keystone.conf /etc/keystone/keystone.conf
chown root:root /etc/keystone/keystone.conf
chmod 644 /etc/keystone/keystone.conf
su -s /bin/sh -c "keystone-manage db_sync" keystone

# Initialize Fernet key repositories
if [[ "$1" != "following" ]]; then
  echo "3.1 - Initialize Fernet key repositories"
else
  echo "7.2.1 - Initialize Fernet key repositories"
fi
keystone-manage fernet_setup --keystone-user keystone \\
--keystone-group keystone
keystone-manage credential_setup --keystone-user keystone \\
--keystone-group keystone

# Bootstrap the Identity service
if [[ "$1" != "following" ]]; then
  echo "3.2 - Bootstrap the Identity service"
else
  echo "7.2.2 - Bootstrap the Identity service"
fi
```

```
sleep 2
keystone-manage bootstrap --bootstrap-password $controllerKeystoneAdminPass \\
--bootstrap-admin-url http://$controllerHostname:35357/v3/ \\
--bootstrap-internal-url http://$controllerHostname:35357/v3/ \\
--bootstrap-public-urlv http://$controllerHostname:5000/v3/ \\
--bootstrap-region-id RegionOne

# Configure the Apache HTTP server
if [[ "$1" != "following" ]]; then
  echo "3.3 - Configure the Apache HTTP server"
else
  echo "7.2.3 - Configure the Apache HTTP server"
fi
sleep 2
echo "ServerName $controllerHostname" >> /etc/apache2/apache2.conf
#ln -s /etc/apache2/sites-available/wsgi-keystone.conf \\
/etc/apache2/sites-enabled

# Finalize the installation
if [[ "$1" != "following" ]]; then
  echo "3.4 - Finalize the installation"
else
  echo "7.2.4 - Finalize the installation"
fi
sleep 2
service apache2 restart
rm -f /var/lib/keystone/keystone.db

export OS_USERNAME=admin
export OS_PASSWORD=$controllerKeystoneAdminPass
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=$controllerKeystoneDomain
export OS_PROJECT_DOMAIN_NAME=$controllerKeystoneDomain
export OS_AUTH_URL=http://$controllerHostname:35357/v3
export OS_IDENTITY_API_VERSION=3
```

```
# Create a domain, projects, users, and roles
if [[ "$1" != "following" ]]; then
  echo "3.5 - Create a domain, projects, users, and roles"
else
  echo "7.3 - Create a domain, projects, users, and roles"
fi
sleep 2
openstack project create --domain $controllerKeystoneDomain --description \\
"Service Project" service
openstack project create --domain $controllerKeystoneDomain --description \\
"Demo Project" demo
openstack user create --domain $controllerKeystoneDomain --password \\
$controllerKeystoneDemoPass demo
openstack role create user
openstack role add --project demo --user demo user

unset OS_USERNAME
unset OS_PASSWORD
unset OS_PROJECT_NAME
unset OS_USER_DOMAIN_NAME
unset OS_PROJECT_DOMAIN_NAME
unset OS_AUTH_URL
unset OS_IDENTITY_API_VERSION

buildAdminSourceFile
buildDemoSourceFile

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.1.4 *Script* instalação de módulo Glance

```
#!/bin/bash
```

```
#########
# Functions declared here
function getVariablesFromConfigFile() {
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    else
      if [[ $line == *"MDBPass="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerMDBPass=${myarray1[1]}
      else
        if [[ $line == *"KeystoneRegion="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerKeystoneRegion=${myarray2[1]}
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystoneDomain=${myarray3[1]}
          else
            if [[ $line == *"GlancePass="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerGlancePass=${myarray4[1]}
            else
              if [[ $line == *"GlanceDBPass="* ]]; then
                IFS='=' read -a myarray5 <<< "$line"
                controllerGlanceDBPass=${myarray5[1]}
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePath"
```

```
}

function storeServiceModulesDataConfigFile() {
  echo "GlancePass=$userInputGlancePass
GlanceDBPass=$userInputGlanceDBPass" >> $confFilePath
}

function buildGlanceAPIFile() {
  echo "[DEFAULT]
verbose = True

[database]
connection = mysql+pymysql://glance:$controllerGlanceDBPass@\\
$controllerHostname/glance
backend = sqlalchemy

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = glance
password = $controllerGlancePass

[paste_deploy]
flavor = keystone

[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

```
[image_format]
disk_formats = ami,ari,aki,vhd,vhdx,vmdk,raw,qcow2,vdi,iso,root-tar" \\
>> glance-api.conf
}

function buildGlanceRegistryFile() {
  echo "[DEFAULT]
verbose = True

[database]
connection = mysql+pymysql://glance:$controllerGlanceDBPass@\\
$controllerHostname/glance
backend = sqlalchemy

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = glance
password = $controllerGlancePass

[paste_deploy]
flavor = keystone" >> glance-registry.conf
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi
```

```
if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 14.04. Other versions \\
  weren't tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Glance OpenStack controller module."
  echo "This is a controller script."
  echo "Change the provided files to your needs."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"14.04"* ]]; then
    echo "This ubuntu version isn't 14.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi
```

```
# Installing glance prerequisites
if [[ "$1" != "following" ]]; then
  echo "2 - Installing prerequisites"
else
  echo "8.1 - Installing prerequisites"
fi

# Getting user input data and storing it in the config file
confFilePath=$(find / -name configFileController.txt)
if [[ "$1" != "following" ]]; then
  echo "2.1 - Getting user input data and storing it in the config file"
  echo "SERVICE PASSWORDS"
  echo "It is recommended that you use distinct passwords for each \\
  module and service"
  read -r -p "Glance (Image) server password: " userInputGlancePass
  read -r -p "Glance (Image) DB password: " userInputGlanceDBPass
  storeServiceModulesDataConfigFile
fi
getVariablesFromConfigFile

# Database commands
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating MySQL database for Glance"
else
  echo "8.1.1 - Creating MySQL database for Glance"
fi
user=root
database=glance
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'localhost' IDENTIFIED BY '$controllerGlanceDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
```

```
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'\%' IDENTIFIED BY '$controllerGlanceDBPass';"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --database="$database" --execute="GRANT \\
ALL PRIVILEGES ON $database.* TO '$database'@'localhost' IDENTIFIED \\
BY '$controllerGlanceDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT \\
ALL PRIVILEGES ON $database.* TO '$database'@'\%' IDENTIFIED \\
BY '$controllerGlanceDBPass';"


# Creating service credentials
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating service credentials"
else
  echo "8.1.2 - Creating service credentials"
fi
sourcePathAdmin=$(find / -name admin-openrc)
. $sourcePathAdmin
openstack user create --domain $controllerKeystoneDomain --password \\
$controllerGlancePass glance
openstack role add --project service --user glance admin
openstack service create --name glance --description "OpenStack Image \\
service" image

# Creating image service API endpoints
if [[ "$1" != "following" ]]; then
  echo "2.3 - Creating image service API endpoints"
else
  echo "8.1.3 - Creating image service API endpoints"
fi
openstack endpoint create --region $controllerKeystoneRegion image\\
 public http://$controllerHostname:9292
openstack endpoint create --region $controllerKeystoneRegion image \\
internal http://$controllerHostname:9292
```

```
openstack endpoint create --region $controllerKeystoneRegion image \\
admin http://$controllerHostname:9292

# Configuring glance components
if [[ "$1" != "following" ]]; then
  echo "2.4 - Configuring components"
else
  echo "8.1.4 - Configuring components"
fi
apt-get install glance -y > /dev/null
buildGlanceAPIFile
buildGlanceRegistryFile
mv glance-api.conf /etc/glance/glance-api.conf
mv glance-registry.conf /etc/glance/glance-registry.conf
chown glance:glance /etc/glance/glance-api.conf
chown glance:glance /etc/glance/glance-registry.conf
chmod 644 /etc/glance/glance-api.conf
chmod 644 /etc/glance/glance-registry.conf
su -s /bin/sh -c "glance-manage db_sync" glance

# Restaring services and finishing installation
if [[ "$1" != "following" ]]; then
  echo "3 - Restaring services and finishing installation"
else
  echo "8.2 - Restaring services and finishing installation"
fi
service glance-registry restart
service glance-api restart
rm -f /var/lib/glance/glance.sqlite

. $sourcePathAdmin
wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
openstack image create "cirros" --file cirros-0.3.4-x86_64-disk.img \\
--disk-format qcow2 --container-format bare --public
openstack image list
```

```
if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.1.5 *Script* instalação de módulo Nova

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray1 <<< "$line"
      controllerOSIP=${myarray1[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray2 <<< "$line"
        controllerHostname=${myarray2[1]}
      else
        if [[ $line == *"MDBPass="* ]]; then
          IFS='=' read -a myarray3 <<< "$line"
          controllerMDBPass=${myarray3[1]}
        else
          if [[ $line == *"RMQUser="* ]]; then
            IFS='=' read -a myarray4 <<< "$line"
            controllerRMQUser=${myarray4[1]}
          else
            if [[ $line == *"RMQPass="* ]]; then
              IFS='=' read -a myarray5 <<< "$line"
              controllerRMQPass=${myarray5[1]}
            else
              if [[ $line == *"KeystoneRegion="* ]]; then
```

```
                IFS='=' read -a myarray6 <<< "$line"
                controllerKeystoneRegion=${myarray6[1]}
              else
                if [[ $line == *"KeystoneDomain="* ]]; then
                  IFS='=' read -a myarray7 <<< "$line"
                  controllerKeystoneDomain=${myarray7[1]}
                else
                  if [[ $line == *"NovaPass="* ]]; then
                    IFS='=' read -a myarray8 <<< "$line"
                    controllerNovaPass=${myarray8[1]}
                  else
                    if [[ $line == *"NovaDBPass="* ]]; then
                      IFS='=' read -a myarray9 <<< "$line"
                      controllerNovaDBPass=${myarray9[1]}
                    fi
                  fi
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePath"
}

function storeServiceModulesDataConfigFile() {
  echo "NovaPass=$userInputNovaPass
NovaDBPass=$userInputNovaDBPass" >> $confFilePath
}

function buildNovaFile() {
  echo "[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
```

```
lock_path=/var/lock/nova
rootwrap_config=/etc/nova/rootwrap.conf
verbose=True
transport_url = rabbit://$controllerRMQUser:$controllerRMQPass@\\
$controllerHostname
auth_strategy = keystone
my_ip = $controllerOSIP
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[api_database]
connection = mysql+pymysql://nova:$controllerNovaDBPass@\\
$controllerHostname/nova_api

[database]
connection = mysql+pymysql://nova:$controllerNovaDBPass@\\
$controllerHostname/nova

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = nova
password = $controllerNovaPass

[vnc]
vncserver_listen = \$my_ip
vncserver_proxyclient_address = \$my_ip

[glance]
api_servers = http://$controllerHostname:9292
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp

[wsgi]
api_paste_config = /etc/nova/api-paste.ini" >> nova.conf
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 14.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Nova OpenStack controller module."
  echo "This is a controller script."
  echo "Change the provided files to your needs."
  read -r -p "Do you wish to continue? [y/N]" response

  # Checking linux distribuion version. Must be Ubuntu 14.04
  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
```

```
  fi

  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"14.04"* ]]; then
    echo "This ubuntu version isn't 14.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Installing nova prerequisites
if [[ "$1" != "following" ]]; then
  echo "2 - Installing prerequisites"
else
  echo "9.1 - Installing prerequisites"
fi

# Getting user input data and storing it in the config file
confFilePath=$(find / -name configFileController.txt)
if [[ "$1" != "following" ]]; then
  echo "2.1 - Getting user input data and storing it in the config file"
  echo "SERVICE PASSWORDS"
  echo "It is recommended that you use distinct passwords for each \\
  module and service"
  read -r -p "Nova (Compute) server password: " userInputNovaPass
  read -r -p "Nova (Compute) DB password: " userInputNovaDBPass
  storeServiceModulesDataConfigFile
fi
getVariablesFromConfigFile
```

```
# Database commands
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating MySQL database for Nova"
else
  echo "9.1.1 - Creating MySQL database for Nova"
fi
user=root
database=nova
database_api=nova_api
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database;"
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database_api;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'localhost' IDENTIFIED BY '$controllerNovaDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'\%' IDENTIFIED BY '$controllerNovaDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database_api" --execute="GRANT ALL PRIVILEGES ON \\
$database_api.* TO '$database'@'localhost' IDENTIFIED BY '\\
$controllerNovaDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database_api" --execute="GRANT ALL PRIVILEGES ON \\
$database_api.* TO '$database'@'\%' IDENTIFIED BY '\\
$controllerNovaDBPass';"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --execute="CREATE DATABASE $database_api;"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'localhost' IDENTIFIED BY '\\
$controllerNovaDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'\%' IDENTIFIED BY
'$controllerNovaDBPass';"
```

```
mysql --user="$user" --database="$database_api" --execute="GRANT \\
ALL PRIVILEGES ON $database_api.* TO '$database'@'localhost' \\
IDENTIFIED BY '$controllerNovaDBPass';"
mysql --user="$user" --database="$database_api" --execute="GRANT \\
ALL PRIVILEGES ON $database_api.* TO '$database'@'\%' IDENTIFIED \\
BY '$controllerNovaDBPass';"

# Create the service entity and API endpoints
if [[ "$1" != "following" ]]; then
  echo "2.3 - Create the service entity and API endpoints"
else
  echo "9.1.2 - Create the service entity and API endpoints"
fi
sourcePathAdmin=$(find / -name admin-openrc)
. $sourcePathAdmin
openstack user create --domain $controllerKeystoneDomain --password \\
$controllerNovaPass nova
openstack role add --project service --user nova admin
openstack service create --name nova --description "OpenStack \\
Compute" compute
openstack endpoint create --region $controllerKeystoneRegion compute \\
public http://$controllerHostname:8774/v2.1/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion compute \\
internal http://$controllerHostname:8774/v2.1/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion compute \\
admin http://$controllerHostname:8774/v2.1/\%\(tenant_id\)s

# Configuring nova components
if [[ "$1" != "following" ]]; then
  echo "3 - Configuring components"
else
  echo "9.2 -Configuring components"
fi
apt-get install nova-api nova-conductor nova-consoleauth nova-novncproxy \\
nova-scheduler -y > /dev/null
```

```
buildNovaFile
mv nova.conf /etc/nova/nova.conf
chown nova:nova /etc/nova/nova.conf
chmod 640 /etc/nova/nova.conf

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "4 - Finalizing installation"
else
  echo "9.3 - Finalizing installation"
fi
su -s /bin/sh -c "nova-manage api_db sync" nova
su -s /bin/sh -c "nova-manage db sync" nova

service nova-api restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart
service nova-novncproxy restart
rm -f /var/lib/nova/nova.sqlite

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.1.6 *Script* instalação de módulo Newton

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
```

```
  IFS='=' read -a myarray1 <<< "$line"
  controllerHostname=${myarray1[1]}
else
  if [[ $line == *"MDBPass="* ]]; then
    IFS='=' read -a myarray2 <<< "$line"
    controllerMDBPass=${myarray2[1]}
  else
    if [[ $line == *"RMQUser="* ]]; then
      IFS='=' read -a myarray3 <<< "$line"
      controllerRMQUser=${myarray3[1]}
    else
      if [[ $line == *"RMQPass="* ]]; then
        IFS='=' read -a myarray4 <<< "$line"
        controllerRMQPass=${myarray4[1]}
      else
        if [[ $line == *"KeystoneRegion="* ]]; then
          IFS='=' read -a myarray5 <<< "$line"
          controllerKeystoneRegion=${myarray5[1]}
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray6 <<< "$line"
            controllerKeystoneDomain=${myarray6[1]}
          else
            if [[ $line == *"NovaPass="* ]]; then
              IFS='=' read -a myarray7 <<< "$line"
              controllerNovaPass=${myarray7[1]}
            else
              if [[ $line == *"NeutronPass="* ]]; then
                IFS='=' read -a myarray8 <<< "$line"
                controllerNeutronPass=${myarray8[1]}
              else
                if [[ $line == *"NeutronDBPass="* ]]; then
                  IFS='=' read -a myarray9 <<< "$line"
                  controllerNeutronDBPass=${myarray9[1]}
                else
```

```
 if [[ $line == *"NeutronSharedSecret="* ]]; then
   IFS='=' read -a myarray <<< "$line"
   controllerNeutronSharedSecret=${myarray[1]}
else
             if [[ $line == *"ODControllerIPAddress="* ]]; then
    IFS='=' read -a myarray10 <<< "$line"
    ODControllerIP=${myarray10[1]}
  fi
fi
         fi
                  fi
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePath"
}

function storeServiceModulesDataConfigFile() {
  echo "NeutronPass=$userInputNeutronPass
NeutronDBPass=$userInputNeutronDBPass
NeutronSharedSecret=$userInputNeutronSharedSecret" >> $confFilePath
}

function buildNeutronFile() {
  echo "[DEFAULT]
auth_strategy = keystone
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
dhcp_agent_notification = True
state_path = /var/lib/neutron
```

```
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
transport_url = rabbit://$controllerRMQUser:$controllerRMQPass@\\
$controllerHostname

[agent]
root_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf

[database]
connection = mysql+pymysql://neutron:$controllerNeutronDBPass@\\
$controllerHostname/neutron

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = neutron
password = $controllerNeutronPass

[nova]
auth_url = http://$controllerHostname:35357
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
region_name = $controllerKeystoneRegion
project_name = service
username = nova
password = $controllerNovaPass

[oslo_concurrency]
lock_path = \$state_path/lock" >> neutron.conf
```

```
}

function buildML2File() {
  echo "[DEFAULT]

[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
#mechanism_drivers = linuxbridge,l2population
mechanism_drivers = openvswitch,linuxbridge,l2population
extension_drivers = port_security

[ml2_type_flat]
flat_networks = provider

[ml2_type_vlan]
network_vlan_ranges = provider

[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
enable_ipset = true
#enable_security_group = True
#firewall_driver = neutron.agent.linux.iptables_firewall.\\
IptablesFirewallDriver" >> ml2_conf.ini
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi
```

```
if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions \\
  weren't tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Neutron OpenStack controller \\
  module."
  echo "This is a controller script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This ubuntu version isn't 16.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi
```

```
# Installing prerequisites
if [[ "$1" != "following" ]]; then
  echo "2 - Installing prerequisites"
else
  echo "10.1 - Installing prerequisites"
fi

# Getting user input data and storing it in the config file
confFilePath=$(find / -name configFileController.txt)
if [[ "$1" != "following" ]]; then
  echo "2.1 - Getting user input data and storing it in the config file"
  echo "SERVICE PASSWORDS"
  echo "It is recommended that you use distinct passwords for each \\
  module and service"
  read -r -p "Neutron (Network) server password: " \\
  userInputNeutronPass
  read -r -p "Neutron (Network) DB password: " \\
  userInputNeutronDBPass
  read -r -p "Neutron (Network) shared secret: " \\
  userInputNeutronSharedSecret
  storeServiceModulesDataConfigFile
fi
getVariablesFromConfigFile

# Database commands
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating MySQL database for Neutron"
else
  echo "10.1.1 - Creating MySQL database for Neutron"
fi
user=root
database=neutron
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
```

```
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'localhost' IDENTIFIED BY '$controllerNeutronDBPass';"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
'$database'@'\%' IDENTIFIED BY '$controllerNeutronDBPass';"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'localhost' IDENTIFIED BY '\\
$controllerNeutronDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'\%' IDENTIFIED BY '\\
$controllerNeutronDBPass';"

# Create the service entity and API endpoints
if [[ "$1" != "following" ]]; then
  echo "2.3 - Create the service entity and API endpoints"
else
  echo "10.1.2 - Create the service entity and API endpoints"
fi
sourcePathAdmin=$(find / -name admin-openrc.sh)
. $sourcePathAdmin
openstack user create --domain $controllerKeystoneDomain --password \\
$controllerNeutronPass neutron
openstack role add --project service --user neutron admin
openstack service create --name neutron --description \\
"OpenStack Networking" network
openstack endpoint create --region $controllerKeystoneRegion network \\
public http://$controllerHostname:9696
openstack endpoint create --region $controllerKeystoneRegion network \\
internal http://$controllerHostname:9696
openstack endpoint create --region $controllerKeystoneRegion network \\
admin http://$controllerHostname:9696

# Networking Option 2: Self-service networks
# Configuring neutron components
```

```
if [[ "$1" != "following" ]]; then
  echo "3 - Configuring components"
else
  echo "10.2 - Configuring components"
fi
apt-get install neutron-server neutron-plugin-ml2 python-neutronclient -y \\
> /dev/null

# Adapting files
if [[ "$1" != "following" ]]; then
  echo "3.1 - Adapting files"
else
  echo "10.2.1 - Adapting files"
fi
echo "[neutron]
url = http://$controllerHostname:9696
auth_url = http://$controllerHostname:35357
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
region_name = $controllerKeystoneRegion
project_name = service
username = neutron
password = $controllerNeutronPass

service_metadata_proxy = True
metadata_proxy_shared_secret = $controllerNeutronSharedSecret"  >>\\
 /etc/nova/nova.conf

# Copying files
if [[ "$1" != "following" ]]; then
  echo "3.2 - Copying files"
else
  echo "10.2.2 - Copying files"
fi
```

```
buildNeutronFile
buildML2File
mv neutron.conf /etc/neutron/neutron.conf
chown root:neutron /etc/neutron/neutron.conf
chmod 640 /etc/neutron/neutron.conf
mv ml2_conf.ini /etc/neutron/plugins/ml2/ml2_conf.ini
chown root:neutron /etc/neutron/plugins/ml2/ml2_conf.ini
chmod 644 /etc/neutron/plugins/ml2/ml2_conf.ini

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "4 - Finalizing installation"
else
  echo "10.3 - Finalizing installation"
fi
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \\
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
service nova-api restart
service neutron-server restart
rm -f /var/lib/neutron/neutron.sqlite

#route
#sleep 5
#cat /etc/network/interfaces
#sleep 5

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.1.7 *Script* instalação de módulo Cinder

```
#!/bin/bash
```

```
#########
# Functions declared here
function getVariablesFromConfigFile() {
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray1 <<< "$line"
      controllerOSIP=${myarray1[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray2 <<< "$line"
        controllerHostname=${myarray2[1]}
      else
        if [[ $line == *"MDBPass="* ]]; then
          IFS='=' read -a myarray3 <<< "$line"
          controllerMDBPass=${myarray3[1]}
        else
          if [[ $line == *"RMQUser="* ]]; then
            IFS='=' read -a myarray4 <<< "$line"
            controllerRMQUser=${myarray4[1]}
          else
            if [[ $line == *"RMQPass="* ]]; then
              IFS='=' read -a myarray5 <<< "$line"
              controllerRMQPass=${myarray5[1]}
            else
              if [[ $line == *"KeystoneRegion="* ]]; then
                IFS='=' read -a myarray6 <<< "$line"
                controllerKeystoneRegion=${myarray6[1]}
              else
                if [[ $line == *"KeystoneDomain="* ]]; then
                  IFS='=' read -a myarray7 <<< "$line"
                  controllerKeystoneDomain=${myarray7[1]}
                else
                  if [[ $line == *"CinderPass="* ]]; then
                    IFS='=' read -a myarray8 <<< "$line"
                    controllerCinderPass=${myarray8[1]}
```

```
                else
                  if [[ $line == *"CinderDBPass="* ]]; then
                    IFS='=' read -a myarray9 <<< "$line"
                    controllerCinderDBPass=${myarray9[1]}
                  fi
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePath"
}

function storeServiceModulesDataConfigFile() {
  echo "CinderPass=$userInputCinderPass
CinderDBPass=$userInputCinderDBPass" >> $confFilePath
}

function buildCinderFile() {
  echo "[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes

rpc_backend = rabbit
```

```
auth_strategy = keystone

my_ip = $controllerOSIP

#notification_driver = messagingv2
transport_url = rabbit://$controllerRMQUser:$controllerRMQPass@\\
$controllerHostname

[database]
connection = mysql+pymysql://cinder:$controllerCinderDBPass@\\
$controllerHostname/cinder

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_plugin = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = cinder
password = $controllerCinderPass

[oslo_concurrency]
lock_path = /var/lib/cinder/tmp" >> cinder.conf
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi
```

```
if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions \\
  weren't tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Cinder OpenStack controller module."
  echo "This is a controller script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This ubuntu version isn't 16.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Installing prerequisites
```

```
if [[ "$1" != "following" ]]; then
  echo "2 - Installing prerequisites"
else
  echo "12.1 - Installing prerequisites"
fi

# Getting user input data and storing it in the config file
confFilePath=$(find / -name configFileController.txt)
if [[ "$1" != "following" ]]; then
  echo "2.1 - Getting user input data and storing it in the config file"
  userInputCinderPass=‘openssl rand -hex 10‘
  userInputCinderDBPass=‘openssl rand -hex 10‘
  storeServiceModulesDataConfigFile
fi
getVariablesFromConfigFile

# Database commands
if [[ "$1" != "following" ]]; then
  echo "2.2 - Creating MySQL database for Cinder"
else
  echo "12.1.1 - Creating MySQL database for Cinder"
fi
user=root
database=cinder
#mysql --user="$user" --password="$controllerMDBPass" --execute=\\
"CREATE DATABASE $database;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
’$database’@’localhost’ IDENTIFIED BY ’$controllerCinderDBPass’;"
#mysql --user="$user" --password="$controllerMDBPass" --database=\\
"$database" --execute="GRANT ALL PRIVILEGES ON $database.* TO \\
’$database’@’\%’ IDENTIFIED BY ’$controllerCinderDBPass’;"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO ’$database’@’localhost’ IDENTIFIED BY ’\\
```

```
$controllerCinderDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT ALL \\
PRIVILEGES ON $database.* TO '$database'@'\%' IDENTIFIED BY '\\
$controllerCinderDBPass';"

# Create the service entity and API endpoints
if [[ "$1" != "following" ]]; then
  echo "2.3 - Create the service entity and API endpoints"
else
  echo "12.1.2 - Create the service entity and API endpoints"
fi
sourcePathAdmin=$(find / -name admin-openrc.sh)
. $sourcePathAdmin
openstack user create --domain $controllerKeystoneDomain --password \\
$controllerCinderPass cinder
openstack role add --project service --user cinder admin
openstack service create --name cinder --description \\
"OpenStack Block Storage" volume
openstack service create --name cinderv2 --description \\
"OpenStack Block Storage" volumev2
openstack endpoint create --region $controllerKeystoneRegion volume \\
public http://$controllerHostname:8776/v1/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion volume \\
internal http://$controllerHostname:8776/v1/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion volume \\
admin http://$controllerHostname:8776/v1/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion volumev2 \\
public http://$controllerHostname:8776/v2/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion volumev2 \\
internal http://$controllerHostname:8776/v2/\%\(tenant_id\)s
openstack endpoint create --region $controllerKeystoneRegion volumev2 \\
admin http://$controllerHostname:8776/v2/\%\(tenant_id\)s

# Configuring cinder components
if [[ "$1" != "following" ]]; then
```

```
  echo "3 - Configuring components"
else
  echo "12.2 - Configuring components"
fi
apt-get install cinder-api cinder-scheduler -y > /dev/null
buildCinderFile
mv cinder.conf /etc/cinder/cinder.conf
chown cinder:cinder /etc/cinder/cinder.conf
chmod 644 /etc/cinder/cinder.conf

su -s /bin/sh -c "cinder-manage db sync" cinder

echo "
[cinder]
os_region_name = $controllerKeystoneRegion" >> /etc/nova/nova.conf

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "4 - Finalizing installation"
else
  echo "12.3 - Finalizing installation"
fi
service nova-api restart
service cinder-scheduler restart
service cinder-api restart
rm -f /var/lib/cinder/cinder.sqlite

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.2 *Scripts* nó *compute*

B.2.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  configFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerOSIP=${myarray[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerHostname=${myarray1[1]}
      else
        if [[ $line == *"ODHostname="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          ODControllerHostname=${myarray2[1]}
        else
          if [[ $line == *"ODControllerIPAddress="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            ODControllerIPAddress=${myarray3[1]}
          fi
        fi
      fi
    fi
  done < "$configFilePathController"
}

function getVariablesFromNetworkConfigFile() {
  configFilePathNetwork=$(find / -name configFileNetwork.txt)
```

```
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      networkOSIP=${myarray[1]}
    else
      if [[ $line == *"NetworkHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        networkHostname=${myarray1[1]}
      fi
    fi
  done < "$configFilePathNetwork"
}

function ipValidation() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 && \\
      ${ip[3]} -le 255 ]]; then
        stat=$?
      fi
  fi
  echo $stat
}

function buildInterfacesFile() {
  echo "# Keep this interface as you see it
# The loopback network interface
auto lo
iface lo inet loopback
```

```
# Compute node network interface for internet access
auto enp0s3
iface enp0s3 inet static
  address $OSIPAddress
  netmask $OSNetmask
  dns-nameservers 8.8.8.8 8.8.4.4" >> interfaces
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "  gateway ${OSGateway[0]}" >> interfaces
  else
    value=100
    for i in "${OSGateway[@]}"
  do
  echo "  up ip route add default via $i dev enp0s3 metric $value" \\
  >> interfaces
  value=$((value+100))
done
  fi

 echo "
# Compute node network interface for tunnel network
auto enp0s8
iface enp0s8 inet static
  address $OSIPAddressTunnel
  netmask $OSNetmaskTunnel

# Compute node network interface for VLAN network
auto enp0s9
iface enp0s9 inet manual
  up ip link set dev \$IFACE up
  down ip link set dev \$IFACE down" >> interfaces
}

function buildHostnameFile() {
  echo "$computeHostname" >> hostname
```

```
}

function buildHostsFile() {
  echo "127.0.0.1 localhost
# 127.0.1.1 $computeHostname

# Change the name of the hosts to your needs
# controller
$controllerOSIP       $controllerHostname

# controller OD
$ODControllerIPAddress   $ODControllerHostname

# network
$networkOSIP     $networkHostname

# compute1
$OSIPAddress      $computeHostname

# block1
# 10.0.0.41       block1

# object1
# 10.0.0.51       object1

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters" >> hosts
}

function buildConfigFile() {
  echo "[compute]
OSIP=$OSIPAddress
OSNetmask=$OSNetmask" >> ../configFileCompute.txt
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
```

```
echo "OSGateway1=${OSGateway[0]}" >> \\
../configFileCompute.txt
  else
    value=1
    for i in "${OSGateway[@]}"
  do
  echo "OSGateway$value=$i" >> ../configFileCompute.txt
  value=$((value+1))
done
  fi
echo "OSIPTunnel=$OSIPAddressTunnel
OSNetmaskTunnel=$OSNetmaskTunnel
ComputeHostname=$computeHostname
" >> ../configFileCompute.txt
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions \\
weren't tested."
echo "This script will verify your system compability."
echo "It's recommended to use a clean install."
echo
echo "This script will replace your /etc/network/interfaces, \\
/etc/hosts and /etc/network/interfaces files."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
```

```
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 14.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Updating system
echo "2 - Updating system"
sleep 2
apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null

# Verifying and importing data from config file
echo "3 - Verifying and importing data from config file"
getVariablesFromControllerConfigFile
getVariablesFromNetworkConfigFile

# Replacing files
echo "4 - Replacing files"
echo "This script, by default, will use the 10.0.0.0/24 network, 10.0.0.31 \\
IPv4 address and three gateways (10.0.0.254, 10.0.0.253 and 10.0.0.252) \\
for the compute node management interface"
read -r -p "Use default? [y/N]" userInputResponseOSManagementNetwork
if [[ $userInputResponseOSManagementNetwork =~ ^([yY][eE][sS]|[yY])$ ]]; then
```

```
  echo "Using default network (10.0.0.0/24)"
  OSIPAddress=10.0.0.31
  OSNetmask=255.255.255.0
  OSGateway+=('10.0.0.254')
  OSGateway+=('10.0.0.253')
  OSGateway+=('10.0.0.252')
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the compute node management interface"
    read -r -p "Compute node IP address: " userInputNewNetworkIP
    read -r -p "Netmask: " userInputNewOSNetmask
  read -r -p "Multiple gateways? [y/N]" userInputResponseMultiGateways
    if [[ $userInputResponseMultiGateways =~ ^([yY][eE][sS]|[yY])$ ]]; then
      validNumber=true
      while $validNumber; do
        read -r -p "How many gateways? " userInputResponseGatewaysNumber
        if [[ $userInputResponseGatewaysNumber =~ ^-?[0-9]+$ ]]; then
          aux=0
          while [ $aux -lt $userInputResponseGatewaysNumber ]; do
            aux=$((aux+1))
            read -r -p "Compute node gateway address $aux: " \\
            userInputNewNetworkGateway
            OSGateway+=("$userInputNewNetworkGateway")
          done
          found=0
          for i in "${OSGateway[@]}"
          do
            if [[ $(ipValidation $i) -ne 0 ]]; then
              found=1
            fi
          done
          if [[ $found -eq 0 ]]; then
            if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
            [[ $(ipValidation $userInputNewNetworkIP) -eq 0 ]]; then
```

```
              validNumber=false
              validInputOS=false
              OSIPAddress=$userInputNewNetworkIP
              OSNetmask=$userInputNewOSNetmask
            fi
          fi
        fi
      done
    else
      read -r -p "Network node gateway address: " userInputNewNetworkGateway
      if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && [[ \\
      $(ipValidation $userInputNewNetworkIP) -eq 0 ]] && [[ $(ipValidation \\
      $userInputNewNetworkGateway) -eq 0 ]]; then
          validInputOS=false
          OSIPAddress=$userInputNewNetworkIP
          OSNetmask=$userInputNewOSNetmask
          OSGateway+=("$userInputNewNetworkGateway")
        fi
    fi
  done
fi
echo "This script, by default, will use the 10.0.1.0/24 network and the \\
10.0.1.31 IPv4 address for the network node tunnel interface"
read -r -p "Use default? [y/N]" userInputResponseOSTunnelNetwork
if [[ $userInputResponseOSTunnelNetwork =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default network (10.0.1.0/24)"
  OSIPAddressTunnel=10.0.1.31
  OSNetmaskTunnel=255.255.255.0
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the network node tunnel interface"
    read -r -p "Compute node tunnel IP address: " userInputNewTunnelIP
    read -r -p "Tunnel network netmask: " userInputNewTunnelNetmask
    if [[ $(ipValidation $userInputNewTunnelIP) -eq 0 ]] && [[ $(ipValidation \\
```

```
    $userInputNewTunnelNetmask) -eq 0 ]]; then
      validInputOS=false
    fi
  done
  OSIPAddressTunnel=$userInputNewTunnelIP
  OSNetmaskTunnel=$userInputNewTunnelNetmask
fi
buildInterfacesFile
mv interfaces /etc/network/interfaces

# Replacing /etc/hostname and /etc/hosts files
echo "4.2 - Replacing /etc/hostname and /etc/hosts files"
echo "This script, by default, will use the following hostname \"compute1\

read -r -p "Use default? [y/N]" userInputResponseHostname
if [[ $userInputResponseHostname =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default hostname"
  computeHostname="compute1"
else
  read -r -p "Insert the new hostname: " userInputNewHostname
  computeHostname=$userInputNewHostname
fi
buildHostnameFile
buildHostsFile
mv hostname /etc/hostname
mv hosts /etc/hosts

#  Generating config file
echo "5 - Generating config file"
buildConfigFile

# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them \\
to the hosts file"
echo "The system need a reboot to apply the changes."
```

```
echo "It is not recommended to install OpenStack and its modules without \\
restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \\
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

B.2.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    fi
  done < "$confFilePathController"
}
#########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
else
  if [[ "$1" != "required" ]] && [[ "$1" != "all" ]] && [[ "$1" != \\
```

```
  "optional" ]] && [[ "$1" != "base" ]] && [[ "$1" != "help" ]]; then
    if [[ "$1" ==  ]]; then
      echo "You need to specify one argument"
    else
      echo "Option -$1- not valid"
    fi
    echo "Rerun script with a valid option (base, optional, required, \\
    all or help)"
    exit
  else
    if [[ "$1" == "help" ]]; then
      echo "Run the script with the following options:"
      echo "  help - Get script options"
      echo "  base - Only OpenStack core"
      echo "  required - OpenStack core, Neutron and Nova modules"
      echo "  all - OpenStack core and all modules"
      echo "  optional - Only optional modules - Cinder"
      exit
    else
      echo "Running script with -$1- enabled"
      echo
      sleep 2
    fi
  fi
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install all the components needed for a correct \\
controller OpenStack installation."
if [[ "$1" == "required" ]]; then
  echo "It'll be installed the following OpenStack modules: Nova and Neutron"
else
  if [[ "$1" == "all" ]]; then
```

```
    echo "It'll be installed the following OpenStack modules: Nova, \\
    Neutron and Cinder"
  else
    if [[ "$1" == "optional" ]]; then
      echo "It'll be installed the following OpenStack modules: Cinder.\n"
    fi
  fi
fi
echo "This is a compute script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Resuming installation with -$1- option enabled"
else
  exit
fi

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 14.04
echo "1.1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi
```

```
# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
getVariablesFromControllerConfigFile

if [[ "$1" != "optional" ]]; then
  # Install Chrony packages
  echo "2 - Installing Chrony NTP server"
  sleep 2
  sudo apt-get update > /dev/null
  apt-get install chrony -y > /dev/null

  echo "2.1 - Editing Chrony conf file"
  sed -i "s/pool 2.debian.pool.ntp.org offline iburst/server \\
  $controllerHostname iburst/" /etc/chrony/chrony.conf

  echo "2.2 - Restarting Chrony server"
  service chrony restart

  # Install OpenStack packages
  echo "3 - Installing OpenStack packages"
  echo "3.1 - Checking pre-requisites and updating repositories to OpenStack \\
  newton version"
  apt-get install software-properties-common -y > /dev/null
  add-apt-repository cloud-archive:newton -y
  apt-get update > /dev/null  && apt-get dist-upgrade -y > /dev/null
  echo "3.2 - OpenStack packages"
  apt-get install python-openstackclient -y > /dev/null

  if [[ "$1" != "base" ]]; then
    #Installing OpenStack Nova module
    echo "4 - Installing Nova module"
    sleep 2
    ../Modules/./os-compute-compute.sh following

    #Installing OpenStack Neutron module
```

```
    echo "5 - Installing Neutron module"
    sleep 2
    ../Modules/./os-compute-network.sh following
  fi
fi
if [[ "$1" != "required" && "$1" != "base" ]]; then
  #Installing OpenStack Block Storage module
  echo "6 - Installing Block Storage module"
  sleep 2
  ../Modules/./os-compute-blockstorage.sh following
fi

echo "END"
```

B.2.3 *Script* instalação de módulo Nova

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFileController() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    else
      if [[ $line == *"RMQUser="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerRMQUser=${myarray1[1]}
      else
        if [[ $line == *"RMQPass="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerRMQPass=${myarray2[1]}
```

```
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystoneDomain=${myarray3[1]}
          else
            if [[ $line == *"NovaPass="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerNovaPass=${myarray4[1]}
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePathController"
}

function getVariablesFromConfigFileCompute() {
  confFilePathCompute=$(find / -name configFileCompute.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ComputeHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      computeHostname=${myarray[1]}
    else
      if [[ $line == *"OSIP="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        computeOSIP=${myarray1[1]}
      else
        if [[ $line == *"OSIPTunnel="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          computeOSIPTunnel=${myarray2[1]}
        fi
      fi
    fi
  done < "$confFilePathCompute"
}
```

```
function buildNovaFile() {
  echo "[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
rootwrap_config=/etc/nova/rootwrap.conf
verbose=True
auth_strategy = keystone
my_ip = $computeOSIP
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
transport_url = rabbit://$controllerRMQUser:$controllerRMQPass@\\
$controllerHostname

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = nova
password = $controllerNovaPass

[vnc]
enabled = True
vncserver_listen = \$my_ip
vncserver_proxyclient_address = \$my_ip
novncproxy_base_url = http://$controllerHostname:6080/vnc_auto.html

[glance]
api_servers = http://$controllerHostname:9292
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp

[wsgi]
api_paste_config = /etc/nova/api-paste.ini" >> nova.conf
}

function buildNovaComputeFile() {
  if [[ "$virtualCap" == "0" ]]; then
    echo "[DEFAULT]
compute_driver=libvirt.LibvirtDriver
[libvirt]
virt_type=qemu" >> nova-compute.conf
  else
    echo "[DEFAULT]
compute_driver=libvirt.LibvirtDriver
[libvirt]
virt_type=kvm" >> nova-compute.conf
  fi
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (following) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
```

```
  echo "This script was tested in Ubuntu 14.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Nova OpenStack compute module."
  echo "This is a compute script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"14.04"* ]]; then
    echo "This ubuntu version isn't 14.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Installing and configuring components
if [[ "$1" != "following" ]]; then
  echo "2 - Installing and configuring components"
else
  echo "4.1 - Installing and configuring components"
fi
getVariablesFromConfigFileController
getVariablesFromConfigFileCompute
```

```
apt-get install nova-compute -y > /dev/null
buildNovaFile
mv nova.conf /etc/nova/nova.conf
chown nova:nova /etc/nova/nova.conf
chmod 640 /etc/nova/nova.conf

virtualCap=$(egrep -c '(vmx|svm)' /proc/cpuinfo)
buildNovaComputeFile
mv nova-compute.conf /etc/nova/nova-compute.conf
chown nova:nova /etc/nova/nova-compute.conf
chmod 640 /etc/nova/nova-compute.conf

# Installing and configuring components
if [[ "$1" != "following" ]]; then
  echo "3 - Finalizing installation"
else
  echo "4.2 - Finalizing installation"
fi
service nova-compute restart
rm -f /var/lib/nova/nova.sqlite

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.2.4 *Script* instalação de módulo Neutron

```
 #!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFileController() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
```

```
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    else
      if [[ $line == *"RMQUser="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerRMQUser=${myarray1[1]}
      else
        if [[ $line == *"RMQPass="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerRMQPass=${myarray2[1]}
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystoneDomain=${myarray3[1]}
          else
            if [[ $line == *"KeystoneRegion="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerKeystoneRegion=${myarray4[1]}
            else
              if [[ $line == *"NeutronPass="* ]]; then
                IFS='=' read -a myarray5 <<< "$line"
                controllerNeutronPass=${myarray5[1]}
              else
if [[ $line == *"ODControllerIPAddress="* ]]; then
              IFS='=' read -a myarray6 <<< "$line"
              ODControllerIP=${myarray6[1]}
else
                  if [[ $line == *"NeutronDBPass="* ]]; then
                    IFS='=' read -a myarray7 <<< "$line"
                    controllerNeutronDBPass=${myarray7[1]}
                  else
if [[ $line == *"NeutronSharedSecret="* ]]; then
  IFS='=' read -a myarray <<< "$line"
      controllerNeutronSharedSecret=${myarray[1]}
```

```
fi
        fi
                fi
   fi
           fi
         fi
        fi
      fi
    fi
  done < "$confFilePathController"
}

function getVariablesFromConfigFileCompute() {
  confFilePathCompute=$(find / -name configFileCompute.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
if [[ $line == *"OSIP="* ]]; then
  IFS='=' read -a myarray <<< "$line"
  computeOSIP=${myarray[1]}
else
  if [[ $line == *"OSIPTunnel="* ]]; then
IFS='=' read -a myarray1 <<< "$line"
computeOSIPTunnel=${myarray1[1]}
  else
    if [[ $line == *"ComputeHostname="* ]]; then
      IFS='=' read -a myarray2 <<< "$line"
      computeHostname=${myarray2[1]}
        fi
      fi
    fi
  done < "$confFilePathCompute"
}

function buildNeutronFile() {
  echo "[DEFAULT]
auth_strategy = keystone
```

```
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

state_path = /var/lib/neutron

#notify_nova_on_port_status_changes = True
#notify_nova_on_port_data_changes = True
transport_url = rabbit://$controllerRMQUser:$controllerRMQPass@\\
$controllerHostname

[agent]
root_helper = sudo /usr/bin/neutron-rootwrap \\
/etc/neutron/rootwrap.conf

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = neutron
password = $controllerNeutronPass

[oslo_concurrency]
lock_path = \$state_path/lock" >> neutron.conf
}

function buildOvSAgentFile() {
  echo "[DEFAULT]

[ovs]
bridge_mappings = provider:br-provider
```

```
local_ip = $computeOSIPTunnel

[agent]
tunnel_types = vxlan
l2_population = True

[securitygroup]
firewall_driver = iptables_hybrid" >> openvswitch_agent.ini
}

function buildML2AgentFile() {
echo "[DEFAULT]

[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
#mechanism_drivers = linuxbridge,l2population
mechanism_drivers = openvswitch,linuxbridge,l2population
extension_drivers = port_security

[ml2_type_flat]
flat_networks = provider

[ml2_type_vlan]
network_vlan_ranges = provider

[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
enable_ipset = true
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.\\
IptablesFirewallDriver" >> ml2_conf.ini
}
```

```
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 14.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Neutron OpenStack compute module."
  echo "This is a compute script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"14.04"* ]]; then
    echo "This ubuntu version isn't 14.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
```

```
      echo
    else
      exit
    fi
  fi
fi

# Configuring neutron components
if [[ "$1" != "following" ]]; then
  echo "2 - Configuring components"
else
  echo "5.1 - Configuring components"
fi
getVariablesFromConfigFileController
getVariablesFromConfigFileCompute

apt-get install neutron-common neutron-plugin-ml2 \\
neutron-plugin-openvswitch-agent -y > /dev/null
buildNeutronFile
buildOvSAgentFile
buildML2AgentFile
echo "
[neutron]
url = http://$controllerHostname:9696
auth_url = http://$controllerHostname:35357
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
region_name = $controllerKeystoneRegion
project_name = service
username = neutron
password = $controllerNeutronPass" >> /etc/nova/nova.conf
mv neutron.conf /etc/neutron/neutron.conf
chown root:neutron /etc/neutron/neutron.conf
chmod 640 /etc/neutron/neutron.conf
```

```
mv openvswitch_agent.ini \\
/etc/neutron/plugins/ml2/openvswitch_agent.ini
chown root:neutron\\
 /etc/neutron/plugins/ml2/openvswitch_agent.ini
chmod 640 /etc/neutron/plugins/ml2/openvswitch_agent.ini
mv ml2_conf.ini /etc/neutron/plugins/ml2/ml2_conf.ini
chown root:neutron /etc/neutron/plugins/ml2/ml2_conf.ini
chmod 640 /etc/neutron/plugins/ml2/ml2_conf.ini

ovs-vsctl add-br br-provider
ovs-vsctl add-port br-provider enp0s9
ovs-vsctl add-br br-tun

service nova-compute restart
service neutron-openvswitch-agent restart

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "3 - Finaling installation"
else
  echo "5.2 - Finaling installation"
fi

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.2.5 *Script* instalação de módulo Cinder

```
#!/bin/bash

#########
# Functions declared here
```

```
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will configure nova for cinder module."
  echo "This is a compute script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This ubuntu version isn't 16.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
```

```
      echo
    else
      exit
    fi
  fi
fi

# Configuring neutron components
if [[ "$1" != "following" ]]; then
  echo "2 - Configuring components"
else
  echo "6.1 - Configuring components"
fi

# Setting up LVM
sed -i '/devices {/a \filter = [ "a/sda/", "r/.*/"]' /etc/lvm/lvm.conf

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.3 *Scripts* nó *network*

B.3.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
```

```
      controllerOSIP=${myarray[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerHostname=${myarray1[1]}
  else
if [[ $line == *"ODHostname="* ]]; then
  IFS='=' read -a myarray2 <<< "$line"
  ODControllerHostname=${myarray2[1]}
else
  if [[ $line == *"ODControllerIPAddress="* ]]; then
    IFS='=' read -a myarray3 <<< "$line"
    ODControllerIPAddress=${myarray3[1]}
  fi
fi
      fi
    fi
  done < "$confFilePathController"
}

function ipValidation() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 \\
      && ${ip[3]} -le 255 ]]; then
        stat=$?
      fi
  fi
  echo $stat
```

```
}

function buildInterfacesFile() {
  echo "# Keep this interface as you see it
# The loopback network interface
auto lo
iface lo inet loopback

# Network node network interface for internet access
auto enp0s3
iface enp0s3 inet static
  address $OSIPAddress
  netmask $OSNetmask
  dns-nameservers 8.8.8.8 8.8.4.4" >> interfaces
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "  gateway ${OSGateway[0]}" >> interfaces
  else
    value=100
    for i in "${OSGateway[@]}"
  do
  echo "  up ip route add default via $i dev enp0s3 metric \\
  $value" >> interfaces
  value=$((value+100))
done
  fi

 echo "
# Network node network interface for tunnel network
auto enp0s8
iface enp0s8 inet static
  address $OSIPAddressTunnel
  netmask $OSNetmaskTunnel

# Network node network interface for VLAN network
auto enp0s9
```

```
iface enp0s9 inet manual
  up ip link set dev \$IFACE up
  down ip link set dev \$IFACE down" >> interfaces
}

function buildHostnameFile() {
  echo "$networkHostname" >> hostname
}

function buildHostsFile() {
  echo "127.0.0.1 localhost
# 127.0.1.1 $networkHostname

# Change the name of the hosts to your needs
# controller
$controllerOSIP       $controllerHostname

# controller ODL
$ODControllerIPAddress   $ODControllerHostname

# network
$OSIPAddress     $networkHostname

# compute1
# 10.0.0.31       compute1

# block1
# 10.0.0.41       block1

# object1
# 10.0.0.51       object1

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters" >> hosts
```

```
}

function buildConfigFile() {
  echo "[network]
OSIP=$OSIPAddress
OSNetmask=$OSNetmask" >> ../configFileNetwork.txt
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "OSGateway1=${OSGateway[0]}" >> ../configFileNetwork.txt
  else
    value=1
    for i in "${OSGateway[@]}"
  do
  echo "OSGateway$value=$i" >> ../configFileNetwork.txt
  value=$((value+1))
done
  fi
echo "OSIPTunnel=$OSIPAddressTunnel
OSNetmaskTunnel=$OSNetmaskTunnel
NetworkHostname=$networkHostname" >> ../configFileNetwork.txt
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
tested."
echo "This script will verify your system compability."
echo "It's recommended to use a clean install."
echo
echo "This script will replace your /etc/network/interfaces, /etc/hosts \\
```

```
and /etc/network/interfaces files."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Updating system
echo "2 - Updating system"
sleep 2
apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null
getVariablesFromControllerConfigFile

# Replacing files
echo "3 - Replacing files"
sleep 2
OSGateway=()

# Replacing /etc/network/interfaces file
```

```
echo "3.1 - Replacing /etc/network/interfaces file"
echo "This script, by default, will use the 10.0.0.0/24 network, \\
10.0.0.21 IPv4 address and three gateways (10.0.0.254, \\
10.0.0.253 and 10.0.0.252) for the network node management \\
interface"
read -r -p "Use default? [y/N]" \\
userInputResponseOSManagementNetwork
if [[ $userInputResponseOSManagementNetwork =~ \\
^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default network (10.0.0.0/24)"
  OSIPAddress=10.0.0.21
  OSNetmask=255.255.255.0
  OSGateway+=('10.0.0.254')
  OSGateway+=('10.0.0.253')
  OSGateway+=('10.0.0.252')
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the network node management \\
    interface"
    read -r -p "Network node IP address: " userInputNewNetworkIP
    read -r -p "Netmask: " userInputNewOSNetmask
  read -r -p "Multiple gateways? [y/N]" \\
  userInputResponseMultiGateways
    if [[ $userInputResponseMultiGateways =~ ^([yY][eE][sS]|[yY])$ ]]; then
      validNumber=true
      while $validNumber; do
        read -r -p "How many gateways? " userInputResponseGatewaysNumber
        if [[ $userInputResponseGatewaysNumber =~ ^-?[0-9]+$ ]]; then
          aux=0
          while [ $aux -lt $userInputResponseGatewaysNumber ]; do
            aux=$((aux+1))
            read -r -p "Network node gateway address $aux: "  \\
            userInputNewNetworkGateway
            OSGateway+=("$userInputNewNetworkGateway")
```

```
            done
            found=0
            for i in "${OSGateway[@]}"
            do
              if [[ $(ipValidation $i) -ne 0 ]]; then
                found=1
              fi
            done
            if [[ $found -eq 0 ]]; then
              if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
              [[ $(ipValidation $userInputNewNetworkIP) -eq 0 ]]; then
                validNumber=false
                validInputOS=false
                OSIPAddress=$userInputNewNetworkIP
                OSNetmask=$userInputNewOSNetmask
              fi
            fi
          fi
        done
      else
        read -r -p "Network node gateway address: " userInputNewNetworkGateway
        if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
        [[ $(ipValidation $userInputNewNetworkIP) -eq 0 ]] && \\
        [[ $(ipValidation $userInputNewNetworkGateway) -eq 0 ]]; then
            validInputOS=false
            OSIPAddress=$userInputNewNetworkIP
            OSNetmask=$userInputNewOSNetmask
            OSGateway+=("$userInputNewNetworkGateway")
          fi
      fi
    done
fi
echo "This script, by default, will use the 10.0.1.0/24 network and the \\
10.0.1.21 IPv4 address for the network node tunnel interface"
read -r -p "Use default? [y/N]" userInputResponseOSTunnelNetwork
```

```
if [[ $userInputResponseOSTunnelNetwork =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default network (10.0.1.0/24)"
  OSIPAddressTunnel=10.0.1.21
  OSNetmaskTunnel=255.255.255.0
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the network node tunnel interface"
    read -r -p "Network node tunnel IP address: " userInputNewTunnelIP
    read -r -p "Tunnel network netmask: " userInputNewTunnelNetmask
    if [[ $(ipValidation $userInputNewTunnelIP) -eq 0 ]] && [[ $(ipValidation \\
    $userInputNewTunnelNetmask) -eq 0 ]]; then
      validInputOS=false
    fi
  done
  OSIPAddressTunnel=$userInputNewTunnelIP
  OSNetmaskTunnel=$userInputNewTunnelNetmask
fi
buildInterfacesFile
mv interfaces /etc/network/interfaces

# Replacing /etc/hostname and /etc/hosts files
echo "3.2 - Replacing /etc/hostname and /etc/hosts files"
echo "This script, by default, will use the following hostname \"network\

read -r -p "Use default? [y/N]" userInputResponseHostname
if [[ $userInputResponseHostname =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default hostname"
  networkHostname="network"
else
  read -r -p "Insert the new hostname: " userInputNewHostname
  networkHostname=$userInputNewHostname
fi
buildHostnameFile
buildHostsFile
```

```
mv hostname /etc/hostname
mv hosts /etc/hosts

#  Generating config file
echo "4 - Generating config file"
buildConfigFile

# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them \\
to the hosts file"
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install OpenStack and its modules without \\
restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \\
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

B.3.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
```

```
    fi
  done < "$confFilePathController"
}
#########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
else
  if [[ "$1" != "required" ]] && [[ "$1" != "base" ]] && [[ "$1" != "help" ]]; then
    if [[ "$1" ==  ]]; then
      echo "You need to specify one argument"
    else
      echo "Option -$1- not valid"
    fi
    echo "Rerun script with a valid option (base, all or help)"
    exit
  else
    if [[ "$1" == "help" ]]; then
      echo "Run the script with the following options:"
      echo "  help - Get script options"
      echo "  base - Only OpenStack core"
      echo "  all - OpenStack core and all modules"
      exit
    else
      echo "Running script with -$1- enabled"
      echo
      sleep 2
    fi
  fi
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
```

```
tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install all the components needed for a correct \\
network OpenStack installation."
if [[ "$1" == "all" ]]; then
  echo "It'll be installed the following OpenStack modules: Neutron"
fi
echo "This is a network script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Resuming installation with -$1- option enabled"
else
  exit
fi

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 14.04
echo "1.1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi

# Verifying and importing data from config file
```

```
echo "1.2 - Verifying and importing data from config file"
getVariablesFromControllerConfigFile

if [[ "$1" != "optional" ]]; then
  # Install Chrony packages
  echo "2 - Installing Chrony NTP server"
  sleep 2
  sudo apt-get update > /dev/null
  apt-get install chrony -y > /dev/null

  echo "2.1 - Editing Chrony conf file"
  sed -i "s/pool 2.debian.pool.ntp.org offline iburst/server \\
  $controllerHostname iburst/" /etc/chrony/chrony.conf

  echo "2.2 - Restarting Chrony server"
  service chrony restart

  # Install OpenStack packages
  echo "3 - Installing OpenStack packages"
  echo "3.1 - Checking pre-requisites and updating repositories to \\
  OpenStack newton version"
  apt-get install software-properties-common -y > /dev/null
  add-apt-repository cloud-archive:newton -y
  apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null
  echo "3.2 - OpenStack packages"
  apt-get install python-openstackclient -y > /dev/null

  if [[ "$1" != "base" ]]; then
    #Installing OpenStack Neutron module
    echo "4 - Installing Neutron module"
../Modules/./os-network-network.sh following
    sleep 2
  fi
fi
```

```
echo "END"
```

B.3.3 *Script* instalação de módulo Neutron

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    else
      if [[ $line == *"RMQUser="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerRMQUser=${myarray1[1]}
      else
        if [[ $line == *"RMQPass="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerRMQPass=${myarray2[1]}
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystoneDomain=${myarray3[1]}
          else
            if [[ $line == *"KeystoneRegion="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerKeystoneRegion=${myarray4[1]}
            else
              if [[ $line == *"NeutronPass="* ]]; then
                IFS='=' read -a myarray5 <<< "$line"
                controllerNeutronPass=${myarray5[1]}
```

```
              else
                if [[ $line == *"NeutronSharedSecret="* ]]; then
                 IFS='=' read -a myarray6 <<< "$line"
                 controllerNeutronSharedSecret=${myarray6[1]}
else
   if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray7 <<< "$line"
      ODControllerIP=${myarray7[1]}
                  else
                    if [[ $line == *"NeutronDBPass="* ]]; then
                      IFS='=' read -a myarray8 <<< "$line"
                      controllerNeutronDBPass=${myarray8[1]}
                    else
                      if [[ $line == *"NovaPass="* ]]; then
                        IFS='=' read -a myarray7 <<< "$line"
                        controllerNovaPass=${myarray7[1]}
                      fi
                    fi
        fi
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePathController"
}

function getVariablesFromNetworkConfigFile() {
  confFilePathNetwork=$(find / -name configFileNetwork.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
if [[ $line == *"OSIPTunnel="* ]]; then
  IFS='=' read -a myarray <<< "$line"
  networkOSTunnelIP=${myarray[1]}
```

```
else
  if [[ $line == *"OSIP="* ]]; then
IFS='=' read -a myarray <<< "$line"
    networkOSIP=${myarray[1]}
  fi
fi
  done < "$confFilePathNetwork"
}

function buildNeutronFile() {
  echo "[DEFAULT]
auth_strategy = keystone
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
state_path = /var/lib/neutron
#notify_nova_on_port_status_changes = True
#notify_nova_on_port_data_changes = True
transport_url = rabbit://$controllerRMQUser:\\
$controllerRMQPass@$controllerHostname

[agent]
root_helper = sudo /usr/bin/neutron-rootwrap \\
/etc/neutron/rootwrap.conf

#[database]
#connection = mysql+pymysql://neutron:$controllerNeutronDBPass\\
@$controllerHostname/neutron

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
```

```
user_domain_name = $controllerKeystoneDomain
project_name = service
username = neutron
password = $controllerNeutronPass

#[nova]
#auth_url = http://$controllerHostname:35357
#auth_type = password
#project_domain_name = $controllerKeystoneDomain
#user_domain_name = $controllerKeystoneDomain
#region_name = $controllerKeystoneRegion
#project_name = service
#username = nova
#password = $controllerNovaPass

[oslo_concurrency]
lock_path = \$state_path/lock" >> neutron.conf
}

function buildML2File() {
  echo "[DEFAULT]

[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
#mechanism_drivers = linuxbridge,l2population
mechanism_drivers = openvswitch,linuxbridge,l2population
extension_drivers = port_security

[ml2_type_flat]
flat_networks = provider

[ml2_type_vlan]
network_vlan_ranges = provider
```

```
[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
enable_ipset = true" >> ml2_conf.ini
}

function buildL3AgentFile() {
  echo "[DEFAULT]
interface_driver = openvswitch
external_network_bridge =
verbose = True" >> l3_agent.ini
}

function buildOvSAgentFile() {
  echo "[DEFAULT]

[agent]
tunnel_types = vxlan
l2_population = True

[ovs]
local_ip = $networkOSTunnelIP
bridge_mappings = provider:br-provider

[securitygroup]
firewall_driver = iptables_hybrid" >> openvswitch_agent.ini
}

function buildDHCPAgentFile() {
  echo "[DEFAULT]
interface_driver = openvswitch
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
force_metadata  = true" >> dhcp_agent.ini
```

```
}

function buildMetadataAgent() {
  echo "[DEFAULT]
nova_metadata_ip = $controllerHostname
metadata_proxy_shared_secret = $controllerNeutronSharedSecret" \\
>> metadata_agent.ini
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Neutron OpenStack compute module."
  echo "This is a compute script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
```

```
  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This ubuntu version isn't 16.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Configuring neutron components
if [[ "$1" != "following" ]]; then
  echo "2 - Configuring components"
else
  echo "5.1 - Configuring components"
fi
getVariablesFromControllerConfigFile
getVariablesFromNetworkConfigFile

apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent \\
neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent \\
python-neutronclient -y > /dev/null
buildNeutronFile
buildML2File
buildDHCPAgentFile
buildMetadataAgent
buildL3AgentFile
buildOvSAgentFile
mv neutron.conf /etc/neutron/neutron.conf
chown root:neutron /etc/neutron/neutron.conf
```

```
chmod 640 /etc/neutron/neutron.conf
mv ml2_conf.ini /etc/neutron/plugins/ml2/ml2_conf.ini
chown root:neutron /etc/neutron/plugins/ml2/ml2_conf.ini
chmod 644 /etc/neutron/plugins/ml2/ml2_conf.ini
mv dhcp_agent.ini /etc/neutron/dhcp_agent.ini
chown root:neutron /etc/neutron/dhcp_agent.ini
chmod 640 /etc/neutron/dhcp_agent.ini
mv metadata_agent.ini /etc/neutron/metadata_agent.ini
chown root:neutron /etc/neutron/metadata_agent.ini
chmod 640 /etc/neutron/metadata_agent.ini
mv l3_agent.ini /etc/neutron/l3_agent.ini
chown root:neutron /etc/neutron/l3_agent.ini
chmod 640 /etc/neutron/l3_agent.ini
mv openvswitch_agent.ini \\
/etc/neutron/plugins/ml2/openvswitch_agent.ini
chown root:neutron \\
/etc/neutron/plugins/ml2/openvswitch_agent.ini
chmod 640 /etc/neutron/plugins/ml2/openvswitch_agent.ini

ovs-vsctl add-br br-provider
ovs-vsctl add-port br-provider enp0s9
ovs-vsctl add-br br-tun


service neutron-l3-agent restart
service neutron-openvswitch-agent restart
service neutron-dhcp-agent restart
service neutron-metadata-agent restart

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "3 - Finaling installation"
else
  echo "5.2 - Finaling installation"
fi
```

```
if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

B.4 *Scripts* nó *block storage*

B.4.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  configFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerOSIP=${myarray[1]}
    else
      if [[ $line == *"ControllerHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerHostname=${myarray1[1]}
      fi
    fi
  done < "$configFilePathController"
}

function getVariablesFromNetworkConfigFile() {
  configFilePathNetwork=$(find / -name configFileNetwork.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      networkOSIP=${myarray[1]}
```

```
    else
      if [[ $line == *"NetworkHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        networkHostname=${myarray1[1]}
      fi
    fi
  done < "$configFilePathNetwork"
}

function getVariablesFromComputeConfigFile() {
  configFilePathCompute=$(find / -name configFileCompute.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      computeOSIP=${myarray[1]}
    else
      if [[ $line == *"ComputeHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        computeHostname=${myarray1[1]}
      fi
    fi
  done < "$configFilePathCompute"
}

function ipValidation() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 && \\
      ${ip[3]} -le 255 ]]; then
```

```
        stat=$?
      fi
  fi
  echo $stat
}

function buildInterfacesFile() {
  echo "# Keep this interface as you see it
# The loopback network interface
auto lo
iface lo inet loopback

# Block storage node network interface for internet access
auto enp0s3
iface enp0s3 inet static
  address $OSIPAddress
  netmask $OSNetmask
  dns-nameservers 8.8.8.8 8.8.4.4" >> interfaces
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
  echo "  gateway ${OSGateway[0]}" >> interfaces
  else
    value=100
    for i in "${OSGateway[@]}"
  do
  echo "  up ip route add default via $i dev enp0s3 metric \\
  $value" >> interfaces
  value=$((value+100))
done
  fi
}

function buildHostnameFile() {
  echo "$blockstorageHostname" >> hostname
}
```

```
function buildHostsFile() {
  echo "127.0.0.1 localhost
# 127.0.1.1 $blockstorageHostname

# Change the name of the hosts to your needs
# controller
$controllerOSIP       $controllerHostname

# network
$networkOSIP     $networkHostname

# compute1
$computeOSIP      $computeHostname

# block1
$OSIPAddress  $blockstorageHostname

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters" >> hosts
}

function buildConfigFile() {
  echo "[blockstorage]
OSIP=$OSIPAddress
OSNetmask=$OSNetmask" >> ../configFileController.txt
  if [[ ${#OSGateway[@]} -eq 1 ]]; then
echo "OSGateway1=${OSGateway[0]}" >> \\
../configFileController.txt
  else
    value=1
    for i in "${OSGateway[@]}"
  do
  echo "OSGateway$value=$i" >> ../configFileController.txt
  value=$((value+1))
```

```
done
  fi
echo "blockstorageHostname=$blockstorageHostname
" >> ../configFileBlockStorage.txt
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions \\
weren't tested."
echo "This script will verify your system compability."
echo "It's recommended to use a clean install."
echo
echo "This script will replace your /etc/network/interfaces, \\
/etc/hosts and /etc/network/interfaces files."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
```

```
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Updating system
echo "2 - Updating system"
sleep 2
apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null

# Verifying and importing data from config file
echo "3 - Verifying and importing data from config file"
getVariablesFromControllerConfigFile
getVariablesFromNetworkConfigFile
getVariablesFromComputeConfigFile

# Replacing files
echo "4 - Replacing files"
sleep 2
OSGateway=()

# Replacing /etc/network/interfaces file
echo "4.1 - Replacing /etc/network/interfaces file"
echo "This script, by default, will use the 10.0.0.0/24 network, \\
10.0.0.1 IPv4 address and three gateways (10.0.0.254, 10.0.0.253 \\
and 10.0.0.252) for the block storage node management interface."
read -r -p "Use default? [y/N]" userInputResponseOSManagementNetwork
if [[ $userInputResponseOSManagementNetwork =~ \\
^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default network (10.0.0.0/24)"
  OSIPAddress=10.0.0.1
  OSNetmask=255.255.255.0
```

```
  OSGateway+=('10.0.0.254')
  OSGateway+=('10.0.0.253')
  OSGateway+=('10.0.0.252')
else
  validInputOS=true
  while $validInputOS; do
    echo "Insert the new values for the block storage node management interface"
    read -r -p "Block Storage node IP address: " userInputNewControllerIP
    read -r -p "Netmask: " userInputNewOSNetmask
  read -r -p "Multiple gateways? [y/N]" userInputResponseMultiGateways
    if [[ $userInputResponseMultiGateways =~ ^([yY][eE][sS]|[yY])$ ]]; then
      validNumber=true
      while $validNumber; do
        read -r -p "How many gateways? " userInputResponseGatewaysNumber
        if [[ $userInputResponseGatewaysNumber =~ ^-?[0-9]+$ ]]; then
          aux=0
          while [ $aux -lt $userInputResponseGatewaysNumber ]; do
            aux=$((aux+1))
            read -r -p "Block Storage node gateway address $aux: " \\
            userInputNewControllerGateway
            OSGateway+=("$userInputNewControllerGateway")
          done
          found=0
          for i in "${OSGateway[@]}"
          do
            if [[ $(ipValidation $i) -ne 0 ]]; then
              found=1
            fi
          done
          if [[ $found -eq 0 ]]; then
            if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && [[ \\
            $(ipValidation $userInputNewControllerIP) -eq 0 ]]; then
              validNumber=false
              validInputOS=false
              OSIPAddress=$userInputNewControllerIP
```

```
              OSNetmask=$userInputNewOSNetmask
            fi
          fi
        fi
      done
    else
      read -r -p "Block Storage node gateway address: "\\
       userInputNewControllerGateway
      if [[ $(ipValidation $userInputNewOSNetmask) -eq 0 ]] && \\
      [[ $(ipValidation $userInputNewControllerIP) -eq 0 ]] && \\
      [[ $(ipValidation $userInputNewControllerGateway) -eq 0 ]]; then
         validInputOS=false
         OSIPAddress=$userInputNewControllerIP
         OSNetmask=$userInputNewOSNetmask
         OSGateway+=("$userInputNewControllerGateway")
      fi
    fi
  done
fi
buildInterfacesFile
mv interfaces /etc/network/interfaces

# Replacing /etc/hostname and /etc/hosts files
echo "4.2 - Replacing /etc/hostname and /etc/hosts files"
echo "This script, by default, will use the following hostname \\
\"blockstorage\
read -r -p "Use default? [y/N]" userInputResponseHostname
if [[ $userInputResponseHostname =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Using default hostname"
  blockstorageHostname="blockstorage"
else
  read -r -p "Insert the new hostname: " userInputNewHostname
  blockstorageHostname=$userInputNewHostname
fi
buildHostnameFile
```

```
buildHostsFile
mv hostname /etc/hostname
mv hosts /etc/hosts

#  Generating config file
echo "4 - Generating config file"
buildConfigFile

# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them \\
to the hosts file"
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install OpenStack and its modules without \\
restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \\
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

B.4.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
```

```
      controllerHostname=${myarray[1]}
    fi
  done < "$confFilePathController"
}
#########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
else
  if [[ "$1" != "required" ]] && [[ "$1" != "base" ]] && [[ "$1" != "help" ]]; \\
  then
    if [[ "$1" ==  ]]; then
      echo "You need to specify one argument"
    else
      echo "Option -$1- not valid"
    fi
    echo "Rerun script with a valid option (base, all or help)"
    exit
  else
    if [[ "$1" == "help" ]]; then
      echo "Run the script with the following options:"
      echo "  help - Get script options"
      echo "  base - Only OpenStack core"
      echo "  all - OpenStack core and all modules"
      exit
    else
      echo "Running script with -$1- enabled"
      echo
      sleep 2
    fi
  fi
fi
```

```
# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install all the components needed for a correct \\
network OpenStack installation."
if [[ "$1" != "base" ]]; then
  echo "It'll be installed the following OpenStack modules: Block Storage"
fi
echo "This is a Block Storage script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Resuming installation with -$1- option enabled"
else
  exit
fi

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 14.04
echo "1.1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi
```

```
# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
getVariablesFromControllerConfigFile

if [[ "$1" != "optional" ]]; then
  # Install Chrony packages
  echo "2 - Installing Chrony NTP server"
  sleep 2
  sudo apt-get update > /dev/null
  apt-get install chrony -y > /dev/null

  echo "2.1 - Editing Chrony conf file"
  sed -i "s/pool 2.debian.pool.ntp.org offline iburst/server \\
  $controllerHostname iburst/" /etc/chrony/chrony.conf

  echo "2.2 - Restarting Chrony server"
  service chrony restart

  # Install OpenStack packages
  echo "3 - Installing OpenStack packages"
  echo "3.1 - Checking pre-requisites and updating repositories to \\
  OpenStack newton version"
  apt-get install software-properties-common -y > /dev/null
  add-apt-repository cloud-archive:newton -y
  apt-get update > /dev/null && apt-get dist-upgrade -y > /dev/null
  echo "3.2 - OpenStack packages"
  apt-get install python-openstackclient -y > /dev/null

  if [[ "$1" != "base" ]]; then
    #Installing OpenStack Block Storage module
    echo "4 - Installing Block Storage module"
../Modules/./os-blockstorage-blockstorage.sh following
    sleep 2
  fi
```

```
fi

echo "END"
```

B.4.3 *Script* instalação de módulo Cinder

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromControllerConfigFile() {
  confFilePathController=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ControllerHostname="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      controllerHostname=${myarray[1]}
    else
      if [[ $line == *"RMQUser="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerRMQUser=${myarray1[1]}
      else
        if [[ $line == *"RMQPass="* ]]; then
          IFS='=' read -a myarray2 <<< "$line"
          controllerRMQPass=${myarray2[1]}
        else
          if [[ $line == *"KeystoneDomain="* ]]; then
            IFS='=' read -a myarray3 <<< "$line"
            controllerKeystoneDomain=${myarray3[1]}
          else
            if [[ $line == *"KeystoneRegion="* ]]; then
              IFS='=' read -a myarray4 <<< "$line"
              controllerKeystoneRegion=${myarray4[1]}
            else
              if [[ $line == *"CinderPass="* ]]; then
```

```
                IFS='=' read -a myarray5 <<< "$line"
                controllerCinderPass=${myarray5[1]}
              else
                if [[ $line == *"CinderDBPass="* ]]; then
                  IFS='=' read -a myarray8 <<< "$line"
                  controllerCinderDBPass=${myarray8[1]}
                fi
              fi
            fi
          fi
        fi
      fi
    fi
  done < "$confFilePathController"
}

function getVariablesFromNetworkConfigFile() {
  confFilePathBS=$(find / -name configFileBlockStorage.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"OSIP="* ]]; then
    IFS='=' read -a myarray <<< "$line"
      bsOSIP=${myarray[1]}
    fi
  done < "$confFilePathBS"
}

function buildCinderFile() {
  echo "[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
```

```
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes

my_ip = $bsOSIP

enabled_backends = lvm

glance_api_servers = http://$controllerHostname:9292

transport_url = rabbit://$RMQUser:$RMQPass@$controllerHostname

[database]
connection = mysql+pymysql://cinder:$controllerCinderDBPass@\\
$controllerHostname/cinder

[keystone_authtoken]
auth_uri = http://$controllerHostname:5000
auth_url = http://$controllerHostname:35357
memcached_servers = $controllerHostname:11211
auth_type = password
project_domain_name = $controllerKeystoneDomain
user_domain_name = $controllerKeystoneDomain
project_name = service
username = cinder
password = $controllerCinderPass

[lvm]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = tgtadm

[oslo_concurrency]
lock_path = /var/lib/cinder/tmp" >> cinder.conf
```

```
}
#########

if [[ "$1" !=  ]] && [[ "$1" != "following" ]]; then
  echo "Option -$1- not valid"
  echo "Rerun script with a valid option (main) or without arguments"
  exit
fi

if [[ "$1" != "following" ]]; then
  # Sudo execution verifycation
  if [[ "$EUID" -ne 0 ]]; then
    echo "Please run this script as root."
    exit
  fi
  # Warnings for the user
  echo "This script was tested in Ubuntu 16.04. Other versions weren't \\
  tested."
  echo "You linux distribution will be tested for compatibility.\n"
  echo "This script will install the Cinder OpenStack Block Storage module."
  echo "This is a Block Storage script."
  read -r -p "Do you wish to continue? [y/N]" response

  if [[ $response =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi

  # Checking linux distribuion version. Must be Ubuntu 14.04
  echo "1 - Checking your linux distribution"
  UV=$(lsb_release -r)
  if [[ "$UV" != *"16.04"* ]]; then
    echo "This ubuntu version isn't 16.04."
    read -r -p "Do you wish to continue? [y/N]" responseVersion
```

```
    if [[ $responseVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
      echo
    else
      exit
    fi
  fi
fi

# Configuring neutron components
if [[ "$1" != "following" ]]; then
  echo "2 - Configuring components"
else
  echo "5.1 - Configuring components"
fi
getVariablesFromControllerConfigFile
getVariablesFromNetworkConfigFile

# Setting up LVM
pvcreate /dev/sdb
vgcreate cinder-volumes /dev/sdb
sed -i '/devices {/a \filter = [ "a/sda/", "a/sdb/", "r/.*/"]' \\
/etc/lvm/lvm.conf

apt-get install cinder-volume -y > /dev/null
buildCinderFile
mv cinder.conf /etc/cinder/cinder.conf
chown cinder:cinder /etc/cinder/cinder.conf
chmod 644 /etc/cinder/cinder.conf

# Finalizing installation
if [[ "$1" != "following" ]]; then
  echo "3 - Finaling installation"
else
  echo "5.2 - Finaling installation"
fi
```

```
sleep 2
service tgt restart
service cinder-volume restart

if [[ "$1" != "following" ]]; then
  echo "END"
fi
```

Apêndice C

Scripts de integração OpenDaylight

Todos os *scripts* estão também disponíveis em
`https://github.com/xonaecom/configureopenstack`.

C.1 *Scripts* nó *controller*

C.1.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function ipValidation() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 && ${ip[3]}
  -le 255 ]]; then
        stat=$?
      fi
```

```
  fi
  echo $stat
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "This script will verify your system compability."
echo "Use this script on a OpenStack controller node."
echo
echo "This script will replace your /etc/hosts file."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
```

```
    exit
  fi
fi

# Replacing files
echo "2 - Replacing files"
sleep 2

# Replacing /etc/hostname and /etc/hosts files
echo "2.1 - Adding ODL hosts attribute"
echo "Insert OpenDaylight controller information"
read -r -p "OpenDaylight controller hostname: " userInputODHostname
ODControllerHostname=$userInputODHostname
validInputODIP=true
  while $validInputODIP; do
    read -r -p "OpenDaylight controller IP: " userInputODIP
    if [[ $(ipValidation $userInputODIP) -eq 0 ]]; then
      ODControllerIPAddress=$userInputODIP
      validInputODIP=false
    fi
  done
echo "
# OpenDaylight controller
$ODControllerIPAddress  $ODControllerHostname" >> /etc/hosts

# adding additional variables to config file
echo "2.2 - Adding ODL hosts attribute"
sleep 2
configFile=$(find / -type f -name "configFileController.txt")
echo "ODControllerIPAddress=$ODControllerIPAddress
ODHostname=$ODControllerHostname" >> $configFile

# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them \
to the hosts file"
```

```
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install ODL controller without restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

C.1.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      ODControllerIPAddress=${myarray[1]}
    else
      if [[ $line == *"NeutronDBPass="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerNeutronDBPass=${myarray1[1]}
      fi
    fi
  done < $confFilePath
}
########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
```

```
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install the ODL controller OpenStack related attributes."
echo "This is a controller script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 16.04
echo "1.1 - Checking your linux distribution"
sleep 2
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi

# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
sleep 2
getVariablesFromConfigFile
```

```
#Installing OpenStack Neutron module
echo "2 - Installing ODL Newton library"
sleep 2
apt-get update && apt-get install git -y > /dev/null
git clone https://github.com/openstack/networking-odl -b stable/newton
cd networking-odl
python setup.py install
cd ..

echo "3 - Applying ODL settings to Neutron Server"
sleep 2
sed -i 's/service_plugins = router/service_plugins = odl-router/' \
/etc/neutron/neutron.conf
sed -i 's/mechanism_drivers = openvswitch,l2population/mechanism_drivers = \
opendaylight/' /etc/neutron/plugins/ml2/ml2_conf.ini
echo "[ml2_odl]
url = http://$ODControllerIPAddress:8080/controller/nb/v2/neutron
password = admin
username = admin" >> /etc/neutron/plugins/ml2/ml2_conf.ini

# Resetting Neutron databases
echo "5 - Resetting Neutron databases"
user=root
database=neutron
mysql --user="$user" --execute="DROP DATABASE IF EXISTS $database;"
mysql --user="$user" --execute="CREATE DATABASE $database;"
mysql --user="$user" --database="$database" --execute="GRANT ALL PRIVILEGES \
ON $database.* TO '$database'@'localhost' IDENTIFIED BY \
'$controllerNeutronDBPass';"
mysql --user="$user" --database="$database" --execute="GRANT ALL PRIVILEGES \
ON $database.* TO '$database'@'%' IDENTIFIED BY '$controllerNeutronDBPass';"
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron

# Restarting services
```

```
echo "5 - Restarting services"
sleep 2
service neutron-server restart

echo "END"
```

C.2 *Scripts* nó *network*

C.2.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      ODControllerIPAddress=${myarray[1]}
    else
      if [[ $line == *"ODHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        ODControllerHostname=${myarray1[1]}
      fi
    fi
  done < "$confFilePath"
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi
```

```
# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "This script will verify your system compability."
echo "Use this script on a OpenStack controller node."
echo
echo "This script will replace your /etc/hosts file."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Replacing files
echo "2 - Replacing files"
getVariablesFromConfigFile
sleep 2

# Replacing /etc/hostname and /etc/hosts files
```

```
echo "2.1 - Adding ODL hosts attribute"
echo "
# OpenDaylight controller
$ODControllerIPAddress  $ODControllerHostname" >> /etc/hosts

# Rebooting and final warnings
echo "As you add more servers to the configuration, you need to add them to \
the hosts file"
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install ODL controller without restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

C.2.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      ODControllerIPAddress=${myarray[1]}
    else
      if [[ $line == *"NeutronDBPass="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerNeutronDBPass=${myarray1[1]}
```

```
      fi
    fi
  done < "$confFilePath"
}

function getVariablesFromNetworkConfigFile() {
  confFilePathNetwork=$(find / -name configFileNetwork.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
if [[ $line == *"OSIPTunnel="* ]]; then
  IFS='=' read -a myarray <<< "$line"
  networkOSTunnelIP=${myarray[1]}
fi
  done < "$confFilePathNetwork"
}
########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install the ODL controller OpenStack related attributes."
echo "This is a controller script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 16.04
```

```
echo "1.1 - Checking your linux distribution"
sleep 2
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi

# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
sleep 2
getVariablesFromConfigFile
getVariablesFromNetworkConfigFile

#Installing OpenStack Neutron module
echo "2 - Installing ODL Newton library"
sleep 2
apt-get update && apt-get install git -y > /dev/null
git clone https://github.com/openstack/networking-odl -b stable/newton
cd networking-odl
python setup.py install
cd ..

echo "3 - Applying ODL settings to Neutron Server"
sleep 2
sed -i 's/service_plugins = router/service_plugins = odl-router/' \
/etc/neutron/neutron.conf
sed -i 's/mechanism_drivers = openvswitch,l2population/mechanism_drivers = \
opendaylight/' /etc/neutron/plugins/ml2/ml2_conf.ini ml2
echo "
```

```
[OVS]
ovsdb_interface = vsctl" >> /etc/neutron/dhcp_agent.ini
echo "
[ml2_odl]
url = http://$ODControllerIPAddress:8080/controller/nb/v2/neutron
password = admin
username = admin" >> /etc/neutron/plugins/ml2/ml2_conf.ini

# Disabling services
echo "4 - Disabling services"
systemctl disable neutron-openvswitch-agent.service
systemctl disable neutron-l3-agent.service

# Resetting OvS
echo "5 - Resetting OvS"
service openvswitch-switch stop
rm -rf /var/log/openvswitch/*
rm -rf /etc/openvswitch/conf.db
service openvswitch-switch start
ovsID=$(ovs-vsctl show)
IFS=' ' read -a ovsArray <<< "$ovsID"
ovs-vsctl set-manager tcp:$ODControllerIPAddress:6640
ovs-vsctl set Open_vSwitch ${ovsArray[0]} other_config=\
{"local_ip"="$networkOSTunnelIP"}
ovs-vsctl add-br br-provider
ovs-vsctl add-port br-provider enp0s9
ovs-vsctl add-br br-tun

# Restarting services
echo "5 - Restarting services"
service neutron-dhcp-agent restart

echo "END"
```

C.3 *Scripts* nó *network*

C.3.1 *Scripts* pré instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      ODControllerIPAddress=${myarray[1]}
    else
      if [[ $line == *"ODHostname="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        ODControllerHostname=${myarray1[1]}
      fi
    fi
  done < "$confFilePath"
}
########

# Sudo execution
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "This script will verify your system compability."
echo "Use this script on a OpenStack controller node."
echo
```

```
echo "This script will replace your /etc/hosts file."
echo "This script WILL need user input"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

if [[ $userInputInitialPrompt =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "Installation initiated"
else
  exit
fi

# Checking your linux distribution
echo "1 - Checking your linux distribution"
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo "Resuming installation with untested Ubuntu version"
  else
    exit
  fi
fi

# Replacing files
echo "2 - Replacing files"
getVariablesFromConfigFile
sleep 2

# Replacing /etc/hostname and /etc/hosts files
echo "2.1 - Adding ODL hosts attribute"
echo "
# OpenDaylight controller
$ODControllerIPAddress  $ODControllerHostname" >> /etc/hosts

# Rebooting and final warnings
```

```
echo "As you add more servers to the configuration, you need to add them \
to the hosts file"
echo "The system need a reboot to apply the changes."
echo "It is not recommended to install ODL controller without restarting."
read -r -p "Do you wish to reboot now or later? [y for now/N for later]" \
responseReboot
if [[ $responseReboot =~ ^([yY][eE][sS]|[yY])$ ]]; then
  echo "The system will now reboot."
  reboot
else
  echo "END"
fi
```

C.3.2 *Scripts* instalação

```
#!/bin/bash

#########
# Functions declared here
function getVariablesFromConfigFile() {
  confFilePath=$(find / -name configFileController.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
    if [[ $line == *"ODControllerIPAddress="* ]]; then
      IFS='=' read -a myarray <<< "$line"
      ODControllerIPAddress=${myarray[1]}
    else
      if [[ $line == *"NeutronDBPass="* ]]; then
        IFS='=' read -a myarray1 <<< "$line"
        controllerNeutronDBPass=${myarray1[1]}
      fi
    fi
  done < "$confFilePath"
}

function getVariablesFromConfigFileCompute() {
```

```
  confFilePathCompute=$(find / -name configFileCompute.txt)
  while IFS='' read -r line || [[ -n "$line" ]]; do
  if [[ $line == *"OSIPTunnel="* ]]; then
IFS='=' read -a myarray <<< "$line"
computeOSIPTunnel=${myarray[1]}
  fi
  done < "$confFilePathCompute"
}
########

# Sudo execution and argument verifycation
if [[ "$EUID" -ne 0 ]]; then
  echo "Please run this script as root."
  exit
fi

# Warnings for the user
echo "This script was tested in Ubuntu 16.04. Other versions weren't tested."
echo "You linux distribution will be tested for compatibility.\n"
echo "This script will install the ODL controller OpenStack related attributes."
echo "This is a controller script."
echo "This script WILL need user input.\n"
echo "Do not change the order of the provided directories or files.\n"
read -r -p "Do you wish to continue? [y/N]" userInputInitialPrompt

# Executing verifications
echo "1 - Executing verifications"

# Checking linux distribuion version. Must be Ubuntu 16.04
echo "1.1 - Checking your linux distribution"
sleep 2
UV=$(lsb_release -r)
if [[ "$UV" != *"16.04"* ]]; then
  echo "This ubuntu version isn't 16.04."
  read -r -p "Do you wish to continue? [y/N]" userInputUbuntuVersion
```

```
  if [[ $userInputUbuntuVersion =~ ^([yY][eE][sS]|[yY])$ ]]; then
    echo
  else
    exit
  fi
fi

# Verifying and importing data from config file
echo "1.2 - Verifying and importing data from config file"
sleep 2
getVariablesFromConfigFile
getVariablesFromConfigFileCompute

#Installing OpenStack Neutron module
echo "2 - Installing ODL Newton library"
sleep 2
apt-get update && apt-get install git -y > /dev/null
git clone https://github.com/openstack/networking-odl -b stable/newton
cd networking-odl
python setup.py install
cd ..

#Applying ODL settings to Neutron Server
echo "3 - Applying ODL settings to Neutron Server"
sleep 2
sed -i 's/service_plugins = router/service_plugins = odl-router/' \
/etc/neutron/neutron.conf
sed -i 's/mechanism_drivers = openvswitch,l2population/mechanism_drivers = \
opendaylight/' /etc/neutron/plugins/ml2/ml2_conf.ini
echo "
[ml2_odl]
url = http://$ODControllerIPAddress:8080/controller/nb/v2/neutron
password = admin
username = admin" >> /etc/neutron/plugins/ml2/ml2_conf.ini
```

```
# Disabling services
echo "4 - Disabling services"
sudo systemctl disable neutron-openvswitch-agent.service

# Resetting OvS
echo "5 - Resetting OvS"
service openvswitch-switch stop
rm -rf /var/log/openvswitch/*
rm -rf /etc/openvswitch/conf.db
service openvswitch-switch start
ovsID=$(ovs-vsctl show)
IFS=' ' read -a ovsArray <<< "$ovsID"
ovs-vsctl set-manager tcp:$ODControllerIPAddress:6640
ovs-vsctl set Open_vSwitch ${ovsArray[0]} other_config=\
{"local_ip"="$computeOSIPTunnel"}
ovs-vsctl add-br br-provider
ovs-vsctl add-port br-provider enp0s9
ovs-vsctl add-br br-tun

echo "END"
```