IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

António Filipe Madeira Reis

novembro | 2015

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

AMBIENTE CLOUD OPENSTACK

ANTÓNIO FILIPE MADEIRA REIS

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Novembro / 2015

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

AMBIENTE CLOUD OPENSTACK

ANTÓNIO FILIPE MADEIRA REIS

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Novembro / 2015

ORIENTADOR: FERNANDO MELO RODRIGUES

COORIENTADOR: PEDRO MANUEL TEIXEIRA PINTO

Elementos Identificativos

Aluno

Nome: António Filipe Madeira Reis

Número: 1010521

Correio eletrónico: filipe-reis@sapo.pt

Curso: Engenharia Informática

Estabelecimento de Ensino

Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271220100 | Fax: 271220150

Orientador do Projeto:

Nome: Fernando Melo Rodrigues

Grau Académico: Mestre

Coorientador do Projeto:

Nome: Pedro Manuel Teixeira Pinto

Grau Académico: Mestre

Agradecimentos

Este espaço é dedicado àqueles que de alguma forma contribuíram para que este projeto fosse realizado.

Agradeço aos meus pais e irmã, pelo apoio e incentivo prestado durante o desenvolvimento deste projeto.

Um agradecimento ao meu orientador de projeto, Professor Fernando Melo Rodrigues, pelo profissionalismo, apoio e disponibilidade.

Um agradecimento ao meu coorientador, Engenheiro Pedro Pinto, pela ajuda e esforço para que fosse possível desenvolver um projeto sobre Cloud Computing, bem como a sua disponibilidade e dedicação prestada ao longo do projeto.

Por fim gostaria de agradecer aos meus colegas de curso, em especial, ao João Delgado pelo apoio prestado durante a realização do projeto.

Resumo

Este documento descreve o trabalho realizado no âmbito da Unidade Curricular Projeto de Informática, da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

O conceito de Cloud Computing tem ganho um enorme destaque nos últimos anos, no sentido em que tem vindo a alterar o paradigma de como as empresas consomem recursos de TI, permitindo-lhes um melhor aproveitamento do seu tempo e recursos devido às características fornecidas por um serviço Cloud. Este trabalho pretende expor o conceito de Cloud Computing, as suas características, os seus modelos de serviço e modelos de implementação. Após este estudo, procedeu-se ao desenvolvimento de uma arquitetura modelo para a disponibilização de serviços segundo este paradigma. Posteriormente foi implementada uma Cloud privada, enquadrada no modelo de serviço *Infrastructure as a Service*, através da tecnologia OpenStack. Foi também implementada a plataforma ownCloud, para *frontend* do serviço de armazenamento de ficheiros do OpenStack, assim como várias ferramentas para gestão e monitorização do ambiente Cloud. Por fim são apresentados testes realizados e resultados obtidos após a implementação da solução.

Palavras-chave:
Cloud Computing, OpenStack, IaaS, *Hosted Virtualization*, Linux.

Abstract

This document describes the work undertaken within the scope of the discipline of Computer Project, in the graduation of Computer Engineering from the School of Technology and Management of the Polytechnic Institute of Guarda.

The concept of Cloud Computing reached new heights in the past few years, changing the paradigm of how companies consume IT resources, namely by enhancing their utilization of time and resources. This project intends to explore the concept of Cloud Computing, it's characteristics, service models and deployment models. Following the study, a model architecture was designed to provide services according to this paradigm. Afterwards a private Cloud was deployed under the service model Infrastructure as a Service, through the OpenStack technology. It was also deployed the ownCloud platform as the frontend of the OpenStack storage service, together with tools for monitorization and management of the Cloud environment. Finally, the tests performed and respective results, after the deployment of the solution, are presented.

Keywords:
Cloud Computing, OpenStack, IaaS, Hosted Virtualization, Linux.

Índice

Índice de Figuras

Índice de Tabelas

Glossário de Abreviaturas

ACL	-	Access List
ADSL	-	Asymmetric Digital Subscriber Line
AMQP	-	Advanced Message Queueing Protocol
API	-	Application Programming Interface
AWS	-	Amazon Web Services
Capex	-	Capital Expenditure
CCNA	-	Cisco Certified Network Associate
Cisco UCS	-	Cisco Unified Computing System
CPU	-	Central Processing Unit
DHCP	-	Dynamic Host Configuration Protocol
DNS	-	Domain Name System
DoS	-	Denial of Service
EC2	-	Elastic Compute Cloud
ELK	-	Elastic Search, Logstash, Kibana
EPT	-	Extended Page Tables
EUA	-	Estados Unidos da América
FTP	-	File Transfer Protocol
HDD	-	Hard Disk Drive
HTTP	-	Hypertext Transfer Protocol
IaaS	-	Infrastructure as a Service
ICMP	-	Internet Control Message Protocol
IDS	-	Intrusion Detection System
ISP	-	Internet Service Provider
IP	-	Internet Protocol
iSCSI	-	Internet Small Computer Systems Interface
JSON	-	JavaScript Object Notation
KVM	-	Kernel-based Virtual Machine
LACP	-	Link Aggregation Control Protocol
LAN	-	Local Area Network
LPI	-	Linux Professional Institute
LTS	-	Long Term Support
LVM	-	Logical Volume Manager
NASA	-	National Aeronautics and Space Administration
NAT	-	Network Address Translation
NIST	-	National Institute of Standards and Technology
NoSQL	-	Non SQL / Non Relational
NRPE	-	Nagios Remote Plugin Executer
OPEX	-	Operation Expenditure
PaaS	-	Platform as a Service
PC	-	Personal Computer
PHP	-	PHP Hypertext Processor
QEMU	-	Quick Emulator
QoS	-	Quality of Service
RAM	-	Random Access Memory
REST	-	Representational State Transfer

S3	-	Simple Storage Service
SaaS	-	Software as a Service
SNMP	-	Simple Network Management Protocol
SO	-	Sistema Operativo
SPI	-	Stateful Packet Inspection
SQL	-	Structured Query Language
SSD	-	Solid State Drive
SSH	-	Secure Shell
STP	-	Spanning Tree Protocol
TCP	-	Transmission Control Protocol
TI	-	Tecnologias da Informação
UDP	-	User Datagram Protocol
URL	-	Uniform Resource Locator
VLAN	-	Virtual LAN
VM	-	Virtual Machine
VNC	-	Virtual Network Computing
VPN	-	Virtual Private Network
WAN	-	Wide Area Network
WSGI	-	Web Server Gateway Interface
YAML	-	YAML Ain’t Markup Language

1. Introdução

Cloud Computing é um conceito em expansão que veio inovar a forma como os recursos das TI são consumidos. Tem como principais vantagens a redução de custos, aprovisionamento de recursos de computação unilateralmente sem necessidade de intermediários, maior eficiência na utilização e gestão destes, a aderente elasticidade de aumento e redução de características consoante a necessidade do utilizador, o acesso é "universal" porque basta estar ligado à Internet para ter acesso a um serviço alojado na Cloud, entre outros. Este último ponto é interessante visto que o denominado "Internet of Things" está a emergir, onde se prevê que futuramente existam cada vez mais dispositivos ligados à Internet [1].

Este projeto consistiu na implementação de uma solução com base no conceito de Cloud Computing, enquadrado no modelo de serviço *Infrastructure as a Service*. O desenvolvimento deste projeto foi composto por duas fases.

Na primeira fase foi elaborada uma arquitetura modelo e identificadas as tecnologias a empregar para o fornecimento de serviços Cloud. A segunda fase consistiu na implementação de uma Cloud privada, com a finalidade de permitir o armazenamento, partilha de ficheiros e a criação de máquinas virtuais na Cloud. Foram também implementados sistemas de monitorização de forma a facilitar a gestão da infraestrutura. Esta solução destina-se a empresas (por norma grandes empresas ou enquadradas na área das TI) que tenham serviços com a necessidade de alta disponibilidade, autonomia, escalabilidade, controlo e soberania da informação.

1.1 Motivação

A principal motivação para a escolha deste projeto, deveu-se ao interesse em desenvolver conhecimentos e aptidões, na área de Cloud Computing e Administração de sistemas Linux. O facto de ser um tema pouco abordado no plano de estudos da licenciatura em Engenharia Informática também contribuiu na escolha deste. Com a evolução da Cloud também novas oportunidades de empregos têm aparecido e consequentemente a necessidade de recrutar profissionais com competências nesta área.

Outra das razões que contribuiu para optar por este tema é o facto de este estar enquadrado no âmbito de Redes de Computadores, sendo esta a área da Informática que mais me suscita interesse.

1.2 Objetivos

Os objetivos que se pretendem atingir com a elaboração deste projeto consistem inicialmente no estudo e projeção de uma arquitetura que forneça serviços Cloud, com as respetivas tecnologias a empregar para que esta possa garantir uma solução que respeite as características essenciais de um serviço de Cloud Computing.

De seguida e com base nos conhecimentos já adquiridos relativamente à arquitetura desenhada inicialmente, será projetada e implementada uma Cloud Privada, que permita satisfazer os serviços de armazenamento de dados, criação e gestão de máquinas virtuais na Cloud. Para tal será feita uma análise dos recursos disponíveis, de forma a obter a solução mais adequada, tendo em conta as limitações dos equipamentos de rede e computação.

1.3 Estrutura do Documento

Este documento compreende cinco capítulos para além do presente capítulo, organizando-se da seguinte forma:

- No segundo capítulo é descrito o Estado da Arte – onde é apresentado o conceito de Cloud Computing, as suas características, os modelos de serviço, os modelos de implementação e o respetivo estudo de várias soluções e *softwares* disponíveis para a sua implementação;
- No terceiro capítulo é descrita a Metodologia e Resultados esperados – onde são apresentadas as metodologias utilizadas no projeto, a calendarização das tarefas e os resultados esperados;
- No quarto capítulo são descritas as Tecnologias utilizadas – onde são apresentadas e descritas as tecnologias utilizadas ao longo do projeto;
- No quinto capítulo é descrita a Implementação da Solução – onde é apresentado detalhadamente o trabalho realizado ao longo do projeto com vista a implementação da solução e os respetivos resultados obtidos;
- No sexto capítulo são descritas as Conclusões – onde são feitas considerações finais sobre o trabalho realizado e trabalho a desenvolver no futuro com vista a melhorar o projeto.

2. Estado da Arte

2.1 Cloud Computing

O conceito de Cloud Computing data as suas primeiras referências em 1996 a partir de um documento interno da Compaq. No entanto a Cloud só ganhou destaque nos últimos anos e o principal motivo deve-se ao facto da forma como as tecnologias de telecomunicações têm evoluído na última década. Hoje em dia é comum qualquer residência ter Internet entre 10 Mbps a 200 Mbps, tendo as empresas ligações com mais largura de banda. Esta evolução na capacidade de transmissão de dados impulsionou o Cloud Computing ao patamar que se encontra hoje [1][14].

2.1.1 Definição de Cloud Computing

O autor do livro "Introdução ao Cloud Computing", António Miguel Ferreira introduz o tema da seguinte forma:

"Cloud Computing é um conceito, e não uma tecnologia, que está a revolucionar a forma como as pessoas e as empresas utilizam os recursos disponibilizados pelas TI.

A definição de Cloud assenta em cinco características essenciais, três modelos de serviço e quatro modelos de implementação." [1]

Definição de Cloud Computing segundo o NIST:

"Cloud Computing é um modelo que permite o acesso ubíquo, conveniente e a pedido através da rede, a um conjunto de recursos de computação partilhados (redes, servidores, armazenamento, aplicações, serviços, etc.), que podem ser rapidamente aprovisionados ou libertados, com um mínimo de esforço e sem interação com o fornecedor.

O modelo Cloud é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implementação." [2]

2.1.2 Características essenciais

As cinco características essenciais de Cloud Computing são [1][2]:

1. **On-demand self-service** (Autosserviço a pedido) – Um utilizador pode unilateralmente aprovisionar recursos de computação, por exemplo, obter tempo de acesso a um servidor ou espaço de armazenamento na rede, sem interagir com outra pessoa, em particular, com um comercial ou técnico de fornecedor de serviços que está a utilizar.
2. **Broad network access** (Acesso generalizado à rede) – As capacidades ou recursos estão disponíveis na rede (Internet, por exemplo) e podem ser acedidos através de mecanismos comuns (browser web, por exemplo) o que promove a utilização de plataformas de clientes comuns, como computadores, portáteis, *tablets*, *smartphones*, entre outros.
3. **Resource Pooling** (Acesso partilhado a recursos) – Os recursos de computação do fornecedor de serviços Cloud são partilhados por vários clientes, com recursos físicos ou virtualizados associados ou dissociados dinamicamente aos clientes, de acordo com o pedido dos mesmos. Há um certo sentido de independência da localização exata dos recursos, mas pode especificar a localização a um nível mais elevado de abstração (por exemplo, país, estado ou *datacenter*). Exemplos de recursos incluem, armazenamento, processamento, memória e largura de banda.
4. **Rapid elasticity** (Elasticidade rápida) – As capacidades (recursos) são disponibilizadas ou canceladas de forma elástica, em alguns casos automaticamente, para escalar o serviço, para cima ou para baixo de acordo com a procura. Para o cliente, as capacidades disponíveis parecem ilimitadas e podem ser solicitadas em qualquer quantidade e em qualquer altura.
5. **Measured service** (Serviço medido) – Os sistemas Cloud controlam e otimizam automaticamente a utilização de recursos através de mecanismos de medida, a um determinado nível de abstração de acordo com o tipo de serviço (por exemplo: espaço em disco, processamento, largura de banda e número de contas de utilizadores ativas). A utilização de recursos pode ser monitorizada, controlada e reportada, promovendo a transparência entre o utilizador do serviço e o fornecedor.

2.1.3 Modelos de Serviço

Os modelos de serviço definem o nível de abstração em que se encontra um serviço (Figura 1). Como referido anteriormente, são três os modelos de serviços associados à Cloud [1][2]:

1. **Infrastructure as a Service** (IaaS) (Infraestrutura como um Serviço) – A capacidade fornecida ao utilizador é o aprovisionamento de recursos de computação como processamento, armazenamento, memória, rede. O utilizador pode instalar e executar *software* (sistemas operativos ou outras aplicações). O utilizador não gere nem controla a infraestrutura mas tem controlo sobre o sistema operativo, aplicações e possivelmente, também sobre aspetos relacionados com a rede (IP, *host*-based *firewalls*, etc.). Em suma o IaaS dá a possibilidade ao utilizador de alojar os seus serviços ou dados (por exemplo, página web) sem ter que se preocupar com a instalação e gestão do *hardware*, apenas tem que gerir o sistema operativo e as aplicações que tem em funcionamento.
2. **Platform as a Service** (PaaS) (Plataforma como um Serviço) – A capacidade fornecida ao utilizador é a possibilidade de colocar na Cloud aplicações criadas ou adquiridas pelo utilizador, que usam linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo fornecedor. O utilizador não gere nem controla a infraestrutura de Cloud subjacente incluindo a rede, servidores, sistemas operativos e armazenamento, mas tem controlo sobre as aplicações instaladas por si e possivelmente também poderá controlar algumas configurações do ambiente onde estas estão alojadas. Resumidamente o PaaS dá a possibilidade a programadores de desenvolver as suas aplicações na Cloud, sem a preocupação da obtenção e gestão do *software* e *hardware* necessário para as executar. Um exemplo popular de PaaS é o *Heroku*, que tem suporte para as linguagens Java, Python, Ruby entre outras. Em Portugal existe o Jelastic da Lunacloud, com suporte para as linguagens Java, Python, PHP, Ruby e Node.js.

3. **Software as a Service** (SaaS) (Software como um Serviço) – A capacidade dada ao utilizador é de aceder a uma aplicação de *software* que é executado numa infraestrutura de computação. As aplicações são acessíveis através de vários dispositivos, como o caso mais comum de um *web browser*. O utilizador não gere nem controla a infraestrutura de computação, incluindo a rede, servidores, sistemas operativos, espaço em disco, memória, ou até mesmo capacidades individuais da aplicação. Pode possivelmente gerir aspetos relacionados com a configuração de utilizadores dessa aplicação.

   Resumidamente o SaaS dá-nos a possibilidade de aceder a aplicações sem a necessidade de as instalar no nosso computador, como por exemplo serviços de email existentes na Internet, como o Gmail e Hotmail, que para acedermos basta termos um *web browser* e uma ligação à Internet.

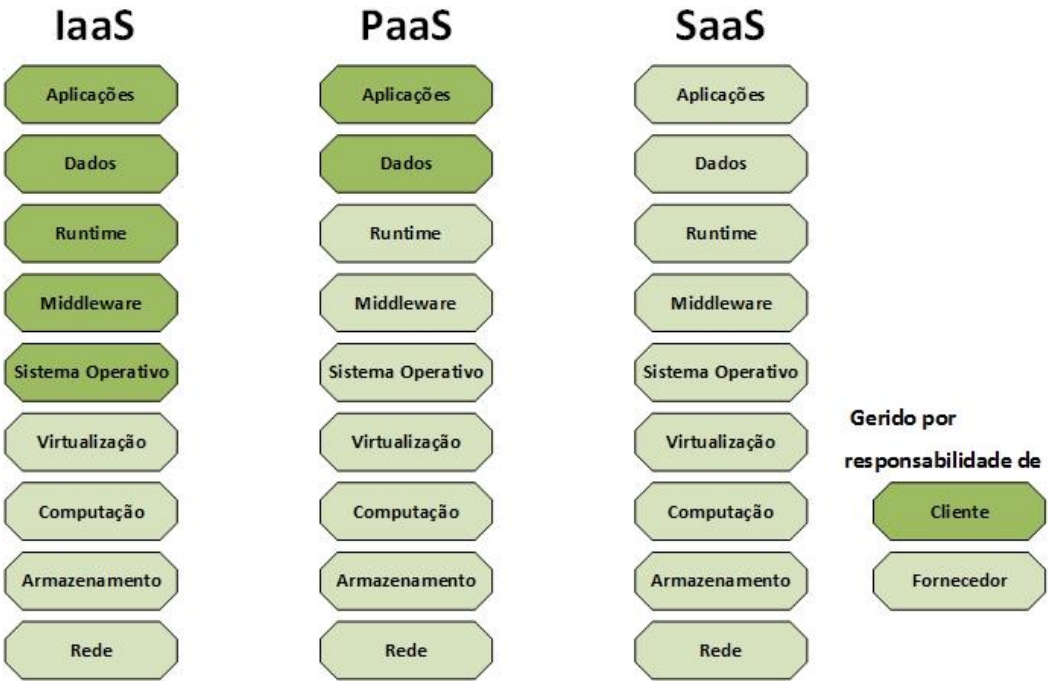


Figura 1 – Níveis de abstração dos vários modelos de serviço

Fonte: [1]

2.1.4 Modelos de implementação

Os modelos de implementação especificam a forma como o serviço nos é prestado. Existem dois modelos principais (Cloud pública e Cloud privada) e duas variantes (Cloud híbrida e Cloud de comunidade) ilustrados na Figura 2. Sendo assim temos os seguintes modelos de implementação [1][2]:

1. **Public Cloud** (Cloud pública) – A infraestrutura de Cloud é disponibilizada para utilização aberta pelo público em geral. Pode pertencer, ser gerida e operada por uma empresa, organização académica, governamental ou uma combinação destas. Esta Cloud está fisicamente localizada nas instalações do fornecedor Cloud.
2. **Private Cloud** (Cloud privada) – A infraestrutura de Cloud é disponibilizada para uso exclusivo de uma única organização (por exemplo, uma empresa), que inclui vários utilizadores (por exemplo, departamentos). Pode pertencer, ser gerida e operada pela própria organização, por uma entidade externa ou uma combinação de ambas e pode estar fisicamente localizada nas instalações dessa organização ou num *datacenter* externo.
3. **Hybrid Cloud** (Cloud híbrida) – A infraestrutura de Cloud é uma combinação de duas ou mais infraestruturas de Cloud (privada, pública, de comunidade), que continuam separadas, mas partilham tecnologia *standard* ou proprietária que permite a portabilidade dos dados ou aplicações (por exemplo, a possibilidade de uma aplicação alojada numa Cloud privada poder fazer *burst* – expandir a sua capacidade para uma Cloud pública).
4. **Community Cloud** (Cloud de comunidade) – A infraestrutura de Cloud é disponibilizada para o uso exclusivo de uma comunidade de utilizadores de empresas que partilham determinadas preocupações (por exemplo, requisitos de segurança, políticas, considerações de conformidade). Pode pertencer, ser gerida e operada por uma ou mais organizações da comunidade, por uma entidade externa ou uma combinação destas e pode estar fisicamente localizada nas instalações dessa organização ou num *datacenter* externo.

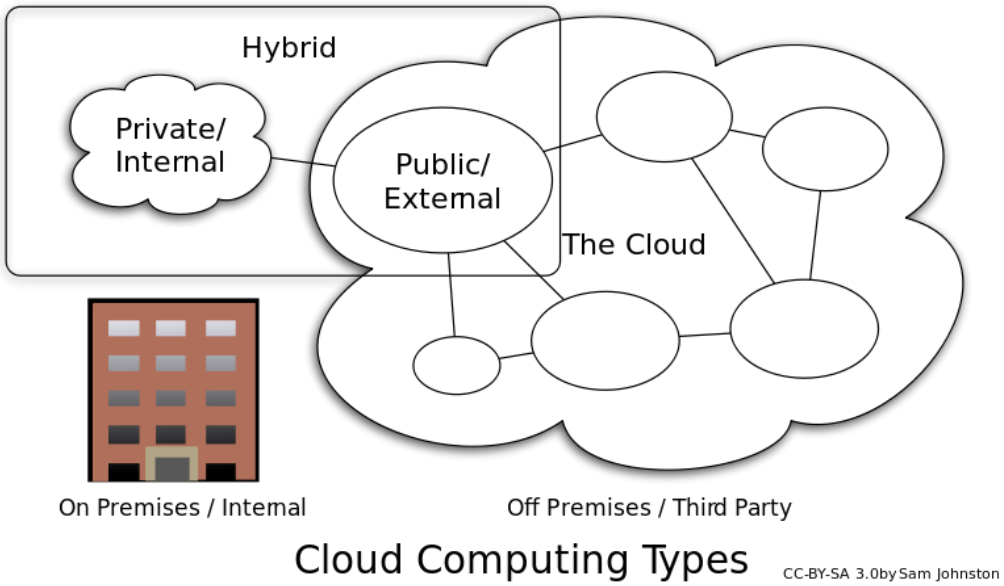


Figura 2 – Modelos de implementação de Cloud

Fonte: [14]

2.1.5 Virtualização

A virtualização é a tecnologia "core" do Cloud Computing. Esta tecnologia torna possível executar vários sistemas operativos e múltiplas aplicações no mesmo servidor em simultâneo [8].

"A virtualização é uma das tecnologias mais importantes da Cloud. É a virtualização que permite otimizar a utilização de recursos informáticos, os computadores e as aplicações, ao permitir que um conjunto de recursos sejam utilizados de forma partilhada, mais eficiente e separada, por vários utilizadores diferentes. A virtualização é muitas vezes confundida com a própria Cloud. Virtualização é uma tecnologia, Cloud é um conceito. Mas a virtualização é um componente fundamental da Cloud." [1]

2.1.6 Cloud Pública vs Cloud Privada

Quando pensamos em recursos computacionais em Cloud Computing, normalmente pensamos em serviços prestados por Clouds públicas tais como a AWS, Google e Rackspace, com a sua infraestrutura ou aplicações a serem partilhadas por milhões de clientes espalhados pelo mundo através da Internet. A principal diferença na infraestrutura de uma Cloud pública ou privada, é que a Cloud pública partilha os seus recursos com múltiplas organizações, enquanto que uma Cloud privada é implementada pela organização (ou por uma terceira entidade contratada a fazer essa implementação) e garante que os recursos computacionais são inteiramente dedicados aos serviços disponibilizados por esta [3][4][5][6].

Uma empresa que queira implementar uma Cloud terá de pensar primeiro o quão crítica é a sua informação e as suas aplicações. Também terá de considerar qualquer regulação ou requisitos de proteção de dados relevantes à sua organização. Por exemplo, organizações nas áreas financeiras ou saúde (bancos, hospitais), têm que obedecer a leis rigorosas sobre o controlo, segurança e soberania dos seus dados, logo uma Cloud privada será a melhor opção neste exemplo [4][5].

Segundo vários autores, as vantagens e desvantagens de uma Cloud pública são as seguintes [1][3][4][5][7]:

Vantagens:

- **Fácil escalabilidade** – Uma das maiores vantagens do uso de uma Cloud pública, é o facto de os seus serviços terem capacidades de computação quase ilimitadas, os recursos são disponibilizados a pedido autonomamente, logo qualquer necessidade de os aumentar ou reduzir, é feito facilmente e eficientemente;
- **Opex[1] vs Capex[2]** – Para start-ups, pequenas e médias empresas que não queiram investir em *hardware*, licenças de *software* (sistema operativo), gestão e manutenção dos equipamentos, pagando apenas o custo operacional, a Cloud pública é solução mais eficiente a nível económico;

[1] Opex – Custos operacionais de forma a manter um sistema ou produto (por exemplo, manutenção dos equipamentos e despesas de eletricidade destes).
[2] Capex – Investimento em bens de capital para implementar ou melhorar, um sistema ou produto (por exemplo, custo relativo à aquisição de equipamentos e imóveis).

- **Backup e recuperação de dados** – O processo de realizar backups deixa de ser uma preocupação porque essa parte é feita pelo fornecedor do serviço Cloud. Os vários fornecedores de Cloud oferecem soluções de *backup*/recuperação de dados flexíveis e de confiança. Em alguns casos a Cloud é usada simplesmente como um repositório de *backups*;
- **Resiliência e Redundância** – A extensa rede de servidores envolvidos numa Cloud pública garante uma maior resiliência nos serviços, mesmo que alguns sectores de um datacenter possam falhar, os balanceadores de cargas simplesmente redistribuem o tráfego pelos sectores não afetados, tornando reduzida a probabilidade de uma Cloud pública não estar disponível.

Desvantagens:

- **Segurança e soberania da informação** – Quando se fala em segurança como desvantagem de uma Cloud pública não significa que esta seja insegura, antes pelo contrário, a desvantagem em relação à segurança, tem a ver com a soberania da informação, pelo facto de certas organizações terem de obedecer a leis restritas relativas ao tratamento e armazenamento dos seus dados. Outro fator é que como uma Cloud pública não tem restrições geográficas, os nossos dados podem estar alojados num fornecedor sediado num país que tenha leis, que obriguem os fornecedores a facultar informações dos seus clientes aos seus governos (por exemplo, a lei "Patriot Act" criada nos EUA que obriga as empresas americanas, mesmo que tendo subsidiárias noutros países, a divulgar ao governo americano informações relativas aos seus clientes sem o conhecimento destes, caso o governo o solicite);
- **Dependência do fornecedor** – Como cada fornecedor tem a sua própria API e ainda não existir um *standard* (apesar de existirem esforços neste sentido, como por exemplo o OpenStack promovido pela Rackspace e NASA), a migração entre fornecedores por vezes torna-se difícil e complicada.

Segundos vários autores, as vantagens e desvantagens de uma Cloud privada são as seguintes [3][4][5][6]:

Vantagens:

- **Maior segurança** – A grande vantagem de uma Cloud privada é a possibilidade de implementar segurança mais específica às necessidades da organização que a está a implementar (por exemplo, um banco). Os servidores estão dentro das instalações dessa organização, ou numa entidade externa de confiança e são geridos pela equipa de TI dessa organização, ou pela entidade externa;
- **Melhor performance** – Ao contrário de uma Cloud pública, numa Cloud privada os recursos só serão acedidos e utilizados pela própria organização, o que garante um melhor aproveitamento e gestão dos mesmos;
- **Custo vs Flexibilidade** – Uma Cloud privada pode ser implementada em torno de grandes infraestruturas dedicadas, com a organização a instalar os seus próprios servidores e *hardware* de armazenamento, o que implica um investimento inicial significativo. Positivamente isto permite uma maior flexibilidade em deslocar cargas de trabalho entre os servidores quando existem picos de utilização ou são implementadas novas aplicações.

Desvantagens:

- **Custos iniciais elevados** - A maior desvantagem no desenvolvimento de uma Cloud privada é a necessidade de um maior investimento inicial, estando este custo relacionado com a implementação da mesma. Existe ainda a necessidade de contratar pessoal especializado para a instalação, configuração e gestão desta, custos na manutenção dos equipamentos e aquisição licenças de *software*;
- **Maior dificuldade de acesso** - Por vezes poderá ser mais difícil aceder à informação, quando os utilizadores estão fora das instalações, devido às medidas de segurança implementadas.

2.2 Fornecedores de serviços Cloud

Visto que o tema do projeto se destina à implementação de uma solução que forneça serviços Cloud (armazenamento e Cloud *servers*), procede-se a apresentação de algumas organizações presentes no mercado com soluções Cloud (com destaque em serviços IaaS).

2.2.1 Amazon Web Services (AWS)

A Amazon é um dos maiores fornecedores de serviços Cloud à escala mundial. Os serviços disponibilizados são vários e a infraestrutura está sediada em 11 regiões geográficas distribuídas pelo mundo. É principalmente conhecida por serviços tais como o *Amazon Elastic Compute Cloud* (EC2 - Figura 3) que permite a criação de *Cloud servers* e o *Amazon Simple Storage Service* (S3) que fornece *web services* para o armazenamento de dados na Cloud. A infraestrutura da AWS é usada em serviços populares tais como a Dropbox e Foursquare [23][24].

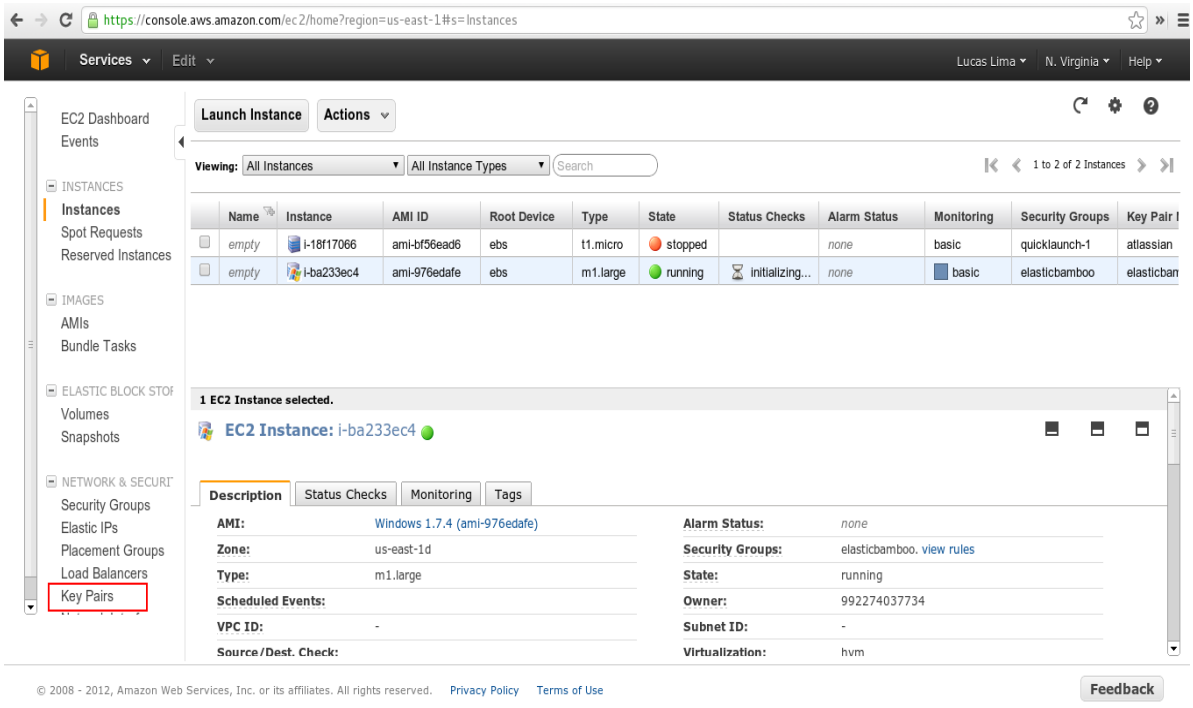


Figura 3 - Dashboard do EC2 da Amazon Web Services

Fonte: [9]

Segundo vários utilizadores, as vantagens e desvantagens da AWS são [25][26]:

Vantagens:

- Preços competitivos e a possibilidade de testar vários serviços gratuitamente com algumas restrições;
- Uma vasta lista de *developers* de aplicações;
- Vários *datacenters* distribuídos pelo mundo;
- Grande variedade de serviços prestados.

Desvantagens:

- Pior performance em comparação à Rackspace;
- Débito por armazenamento não utilizado.

2.2.2 Rackspace

A Rackspace é outro grande fornecedor de serviços Cloud a nível mundial, que disponibiliza uma grande variedade de serviços na Cloud baseados em "Utility Computing", que se trata de um modelo de fornecimento de recursos e gestão de infraestrutura disponíveis ao utilizador consoante a sua necessidade. Cobra por essa quantidade específica em vez de uma taxa fixa. A infraestrutura está sediada em 6 regiões geográficas distribuídas pelo mundo [27][28].

Segundo vários utilizadores, as vantagens e desvantagens da Rackspace são [25][26]:

Vantagens:

- Extremamente focada no serviço ao utilizador;
- Melhor para médias empresas/organizações;
- Serviço mais estável que a AWS (com menores interrupções de serviço).

Desvantagens:

- Por norma os serviços são mais caros que a AWS;
- Mais pequena que a AWS, oferece uma menor extensão de *datacenters* pelo mundo;
- Menor lista de *developers* de aplicações.

A Figura 4 apresenta a *Dashboard* dos serviços de *hosting* na Cloud da Rackspace.

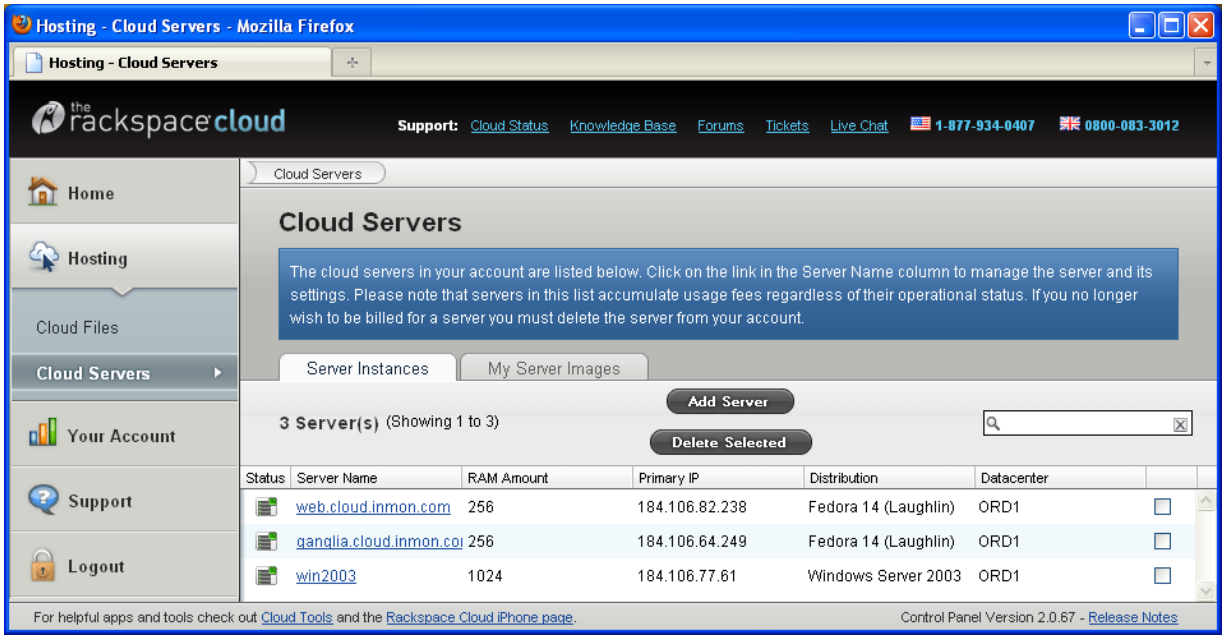


Figura 4 - Dashboard Rackspace

Fonte: [10]

2.2.3 Lunacloud

A Lunacloud é um fornecedor de serviços Cloud que opera principalmente na Europa. Fundada em Portugal por António Miguel Ferreira e Charles Nasser em 2011, a Lunacloud disponibiliza três serviços base. *Cloud Servers* e *Cloud Storage*, que permitem a criação de servidores virtuais escaláveis e armazenamento de dados através da Internet. *Cloud Jelastic*, que fornece uma plataforma para o desenvolvimento e alojamento de aplicações, suportando várias linguagens de programação como Python, Java entre outras. A infraestrutura está sediada em 2 regiões geográficas na Europa, sendo estas Lisboa e Paris [29][30].

Segundo uma análise feita pelo Cloud Spectator em 2012, as vantagens e desvantagens da Lunacloud são [30][31]:

Vantagens:

- Preços competitivos e facilidade de localização para clientes no continente Europeu;
- Simples e acessível para utilizadores com menos experiência nas TI;
- Melhor desempenho que a Rackspace e AWS no serviço de Cloud *servers*.

Desvantagens:

- Menor variedade de serviços prestados quando comparada com a AWS e Rackspace;
- Mais pequena que a AWS e Rackspace estando de momento apenas presente no continente Europeu.

A Figura 5 representa a *Dashboard* da Lunacloud.

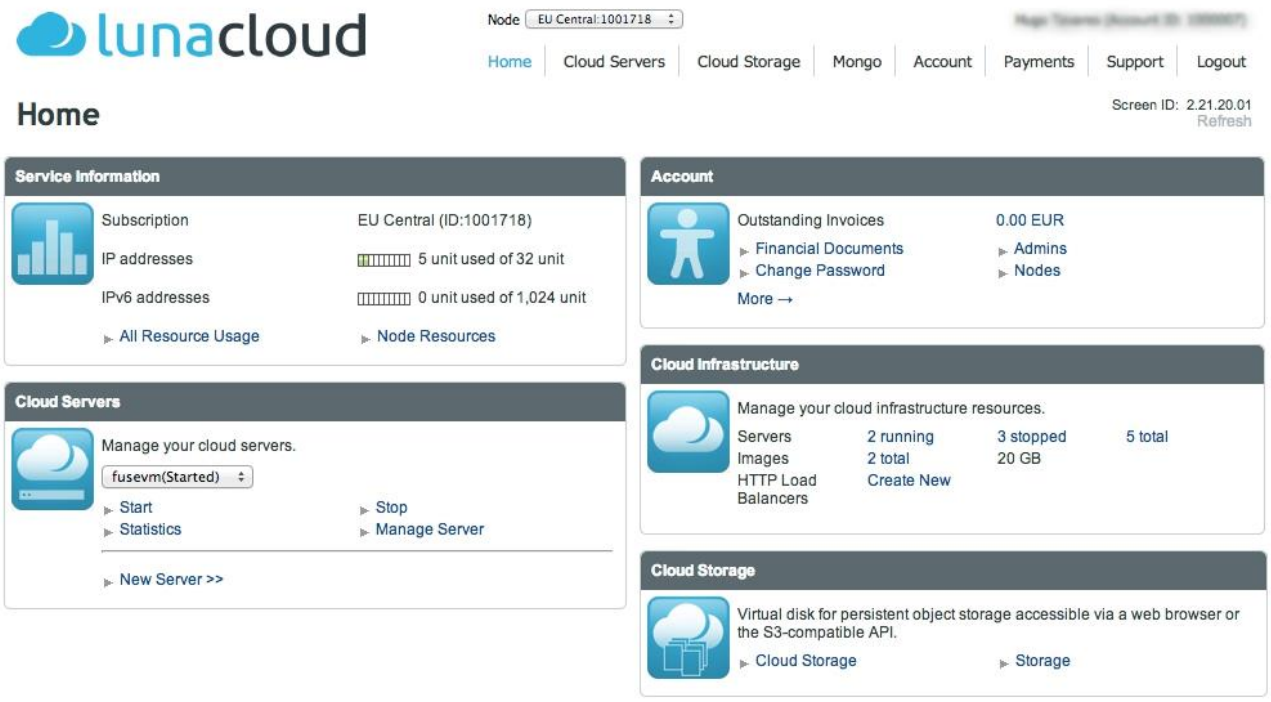


Figura 5 – Dashboard da Lunacloud

Fonte: [11]

2.2.4 Cisco

A Cisco é uma empresa multinacional cuja principal atividade é o fornecimento de soluções e equipamentos na área de redes. Foi fundada em 1984 e na sua primeira fase produziam apenas *routers.* Atualmente a Cisco para além da produção e venda de equipamento de redes (*Routers, Switches etc..)* também fornece soluções para redes (segurança, mobilidade, Cloud etc..). Relativamente a soluções Cloud, a Cisco oferece serviços para a criação e desenvolvimento de ambientes Cloud (públicas, privadas e híbridas) utilizando a tecnologia OpenStack, podendo o utilizador gerir a infraestrutura e a Cisco implementar a solução (*Cisco UCS* para *Red Hat OpenStack Platform*). Pode também optar que a implementação e gestão da infraestrutura seja feita pela Cisco (*Cisco Metapod)*. Ou optar por utilizar uma infraestrutura pública dentro do ecossistema da *Cisco Cloud Services* [17][18].

Outras empresas como a Canonical, Red Hat, Mirantis têm serviços semelhantes à Cisco para a implementação e gestão de ambientes Cloud (privadas/híbridas), principalmente no modelo de serviço IaaS.

Segundo vários autores, as vantagens e desvantagens dos serviços Cloud prestados pela Cisco são [18][32]:

Vantagens:

- Maior rapidez na implementação da Cloud sem necessidade de recrutar/formar *staff* especializado em OpenStack;
- Escalabilidade e alta performance;
- Alta redundância e assistência especializada;
- Alta interoperabilidade.

Desvantagens:

- Necessidade de formação do *staff* para trabalhar com os sistemas Cisco;
- Custo inicial mais elevado quando comparada às soluções disponibilizadas pelos fornecedores de Clouds públicas.

2.3 Plataformas de *Software* de desenvolvimento Cloud Computing

Para a criação de um ambiente Cloud existem várias ferramentas disponíveis para o seu desenvolvimento, algumas *open-source,* outras que requerem licenças pagas. Relativamente à implementação da solução, foram consideradas apenas ferramentas *open-source*. De seguida são identificadas e especificadas algumas destas ferramentas.

2.3.1 OpenStack

O OpenStack é uma plataforma de *software* para desenvolvimento e gestão de ambientes Cloud. Foi criado em 2010 num projeto conjunto entre a Rackspace e a NASA, sendo gerido atualmente pela OpenStack Foundation. É principalmente implementado pelas empresas no modelo de serviço IaaS, a tecnologia é baseada em Python e consiste num grupo de projetos/componentes inter-relacionados que controlam grandes quantidades de recursos (processamento, armazenamento, rede etc..). Estes recursos podem ser geridos pelos utilizadores graficamente através de uma interface *web* (Figura 6), ou através de linha de comandos pela OpenStack API. O OpenStack é utilizado como ambiente de produção de empresas tais como: American Express, Asus, BMW, Disney, Intel, NASA, Orange, Rackspace, Yahoo, Walmart, entre outras [12][13].

Segundo vários autores, as vantagens e desvantagens do OpenStack são [15][16][19][33]:
Vantagens:

- **Suporta vários *Hypervisors*** - O OpenStack tem suporte para QEMU/KVM, sendo o KVM o *default* e recomendado, conta também com suporte limitado para VMware ESX, Citrix Xen e Microsoft Hyper-V, a partir da versão Kilo tem também já algum suporte para *bare-metal servers (*projeto *Ironic);*
- **Vastas funcionalidades de Rede -** A *framework* de rede do OpenStack tem funcionalidades como balanceamento de cargas, sistemas de deteção de intrusões (IDS), *firewall*, entre outros, que garantem a redundância e resiliência da rede OpenStack;

- **Grande suporte por parte da comunidade** - O OpenStack é a plataforma *open-source* mais utilizada para o desenvolvimento de ambientes Cloud, com uma comunidade de utilizadores e *developers* superior às outras plataformas, contando também com o apoio e contribuição de grandes empresas como a Cisco, Dell, IBM, Rackspace entre outros;
- **Compatibilidade com a AWS –** A API do OpenStack tem compatibilidade com os serviços EC2 e S3 da Amazon Web Services, o que permite às empresas a possibilidade de implementar Clouds híbridas com maior facilidade.

Desvantagens:

- **Difícil configuração e implementação** - Visto que o OpenStack tem uma implementação fragmentada, através da incorporação de vários módulos, é necessário um bom conhecimento da tecnologia e tempo para desenvolver a solução desejada;
- **Necessidade de *staff* especializado** - Tirando uma solução paga (como a Canonical, Red Hat, Mirantis, Cisco, entre outros oferecem), a implementação e gestão do ambiente OpenStack requer pessoal especializado. A Mirantis, Rackspace e Red Hat, oferecem formação e certificação em OpenStack.

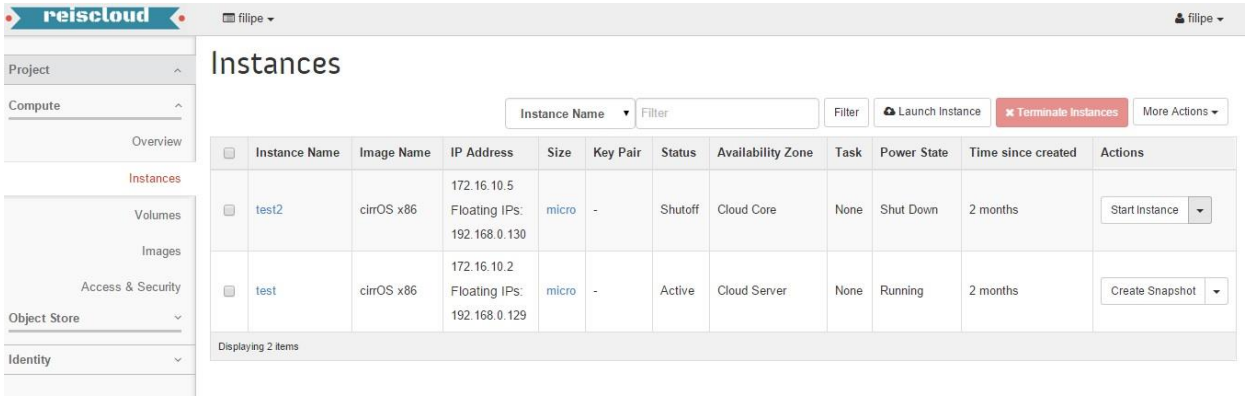


Figura 6 - Dashboard do OpenStack

Fonte: Fonte própria

2.3.2 CloudStack

O CloudStack é uma plataforma de *software open-source* destinada a criar, gerir e implementar serviços Cloud no modelo de serviço IaaS. Foi inicialmente desenvolvido pela *Cloud.com* em 2010 e adquirido em 2011 pela Citrix. O CloudStack é baseado em Java e permite a gestão de vários recursos computacionais através de uma *interface web* (Figura 7) ou linha de comandos via API. O CloudStack é utilizado como sistema de produção de empresas como: Apple, China Telecom, Fujitsu, Huawei, Orange, entre outras [20][21].

Segundo vários autores as vantagens e desvantagens do CloudStack são [15][16][19][33]:

Vantagens:

- **Implementação mais simplificada –** Relativamente ao OpenStack, a dificuldade de implementação é uma desvantagem, no caso do CloudStack é uma vantagem, porque a sua arquitetura é mais centralizada e simples de implementar;
- **Documentação detalhada –** A documentação disponibilizada pelo CloudStack para a sua implementação é bem estruturada, fácil de acompanhar e compreender, com um bom nível de detalhe;
- **Interface gráfica bastante desenvolvida –** A *Dashboard* do CloudStack permite o acesso a grande parte das funcionalidades disponíveis através da API e de forma fácil e intuitiva;
- **Suporta vários *Hypervisors* –** O CloudStack suporta vários *Hypervisors*, entre eles o Citrix XenServer, Oracle VM, VMware, KVM e vSphere.

Desvantagens:

- **Arquitetura rígida –** Como a arquitetura do CloudStack é mais centralizada e facilita uma implementação mais *standard,* também tem como desvantagem a limitação da flexibilidade desta implementação para casos mais específicos, tornando-se assim necessário a aquisição de conhecimentos mais profundos para determinadas implementações;

- **Menor mercado e comunidade** – Visto que o CloudStack é uma tecnologia mais recente quando comparada ao OpenStack, tem uma comunidade e suporte inferior. Relativamente ao mercado empresarial, o CloudStack está atrás do OpenStack tanto a nível de suporte como apoio das grandes corporações, tendo em conta que o OpenStack é a tecnologia *open-source* mais adotada.

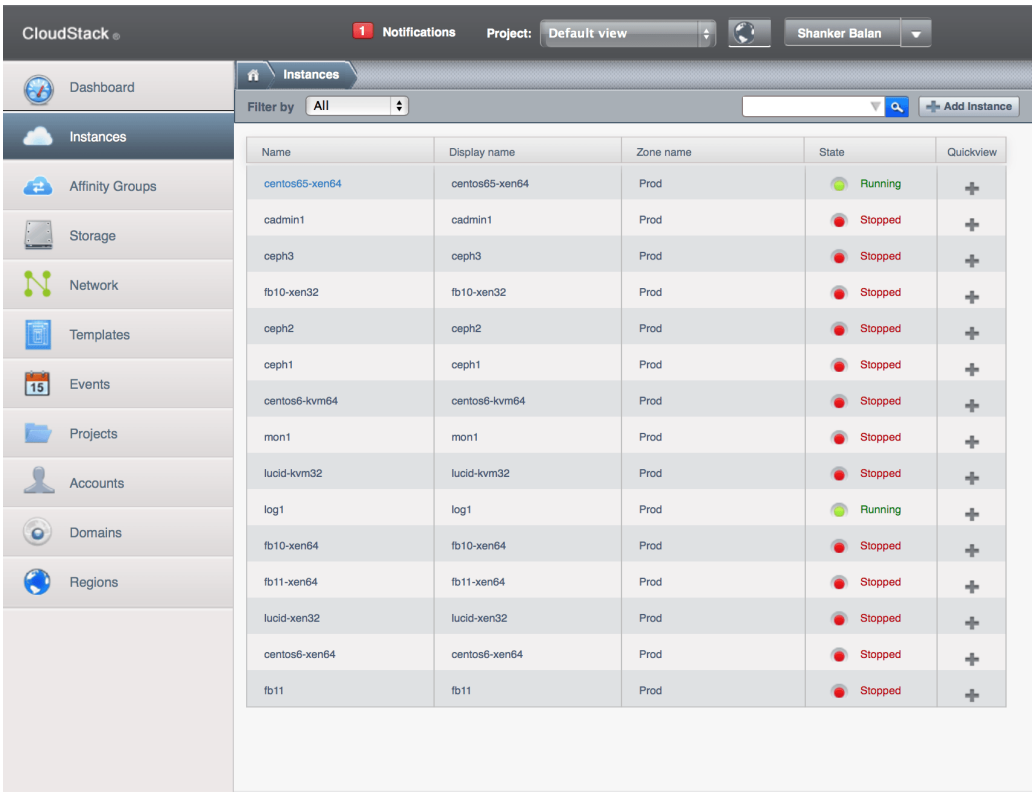


Figura 7 - Dashboard do CloudStack

Fonte: [22]

2.3.3 Escolha da Plataforma de *Software*

Após a identificação e descrição das melhores plataformas de *software open-source* para o desenvolvimento de uma solução Cloud no modelo de serviço IaaS é necessário verificar se estas cumprem os requisitos necessários para a implementação da solução desejada (Tabela 1). Os requisitos necessários são os seguintes:

- Possibilidade de virtualização com e sem suporte de aceleração por *hardware;*
- Possibilidade de criação de máquinas virtuais e armazenamento de dados onde os recursos são garantidos (CPU, RAM, espaço em disco);

- Redundância e alta disponibilidade dos dados;
- Elasticidade dos recursos aprovisionados consoante a necessidade do utilizador;
- Gestão centralizada;
- Compatibilidade com uma plataforma de armazenamento de dados *Open-Source* (ownCloud, Pydio, Seafile) para que funcione como o *backend* da mesma;
- API simples de forma a facilitar a criação de *scripts* para gerir o ambiente Cloud;
- Fácil implementação sobre *commodity hardware*[3].

Tabela 1 - Tabela de Comparação entre OpenStack e CloudStack [13][21]

	OpenStack	**CloudStack**
Sistemas Operativos *Guest* Suportados	Windows *Server* e Linux	Windows *Server* e Linux
Suporte a virtualização assistida por *hardware* ou *software based*	Sim	Pouco suporte caso o *host* não suporte aceleração por *hardware*
Networking	Auto alocação de endereços IP e DNS (DHCP), NAT, VLANs, VPNs, IDS, balanceamento de cargas, *firewalls*	Auto alocação de endereços IP e DNS (DHCP), NAT, VLANs, VPNs, IDS, balanceamento de cargas, *firewalls*
Gestão por *API* ou via *Web*	Sim	Sim
Alta Disponibilidade	Sim	Sim
Criação de *Snapshots*	Sim	Sim
Compatibilidade com AWS	Sim	Sim
Autosserviço a pedido	Via *interface Web* ou via *API*	Via *interface Web* ou via *API*
Compatibilidade com uma plataforma de armazenamento de dados *Open-Source*	ownCloud	Não

[3] *Commodity hardware* – Trata-se de *hardware* de baixo custo e sem características específicas (por exemplo, *hardware* disponível na maioria das superfícies comerciais).

Realizado o estudo das plataformas de *software* em relação às potencialidades e enquadramento na solução a implementar, optou-se por escolher o *OpenStack* devido à sua maior abrangência no mercado, maior suporte da comunidade e compatibilidade para funcionar como *backend* do ownCloud. Concluindo, este cumpre todos os requisitos estabelecidos anteriormente para obter os serviços Cloud desejados.

2.4 Plataforma de Armazenamento de Dados

Apesar do OpenStack já contar com uma *interface web* onde é possível ao utilizador fazer *download* e *upload* de ficheiros, esta é bastante básica e limitada. Para uma melhor interação com o utilizador, existem várias plataformas de armazenamento de dados (ao estilo da Dropbox*)*, entre as mais populares estão o ownCloud, Pydio e Seafile. Todos estes são *softwares* de sincronização de ficheiros com uma *interface web* simples e completa, que permitem a partilha dos ficheiros com outros utilizadores. Como apenas o ownCloud tem já integrada a funcionalidade de funcionar como *frontend* do OpenStack, foi este o escolhido para esta tarefa.

2.4.1 ownCloud

O ownCloud é um *software open-source* gratuito que permite o armazenamento e sincronização de ficheiros. A sua *interface* e funcionamento é muito semelhante à Dropbox (Figura 8), é possível instala-lo e opera-lo numa Cloud ou servidor privado em que a limitação do espaço de armazenamento é o total disponível na infraestrutura. Foi desenvolvido em 2010 por Frank Karlistschek, com a finalidade de fornecer uma alternativa aos sistemas de armazenamento com código proprietário. O ownCloud, devido à sua popularidade dentro da sua área, possui uma vasta lista de *developers* de aplicações. É possível adicionar armazenamento externo via Dropbox, OpenStack, FTP, Google Docs, S3, entre outros. Vem integrada também uma aplicação que permite a encriptação dos dados do lado do servidor, de forma a fornecer maior segurança e privacidade [34][35].

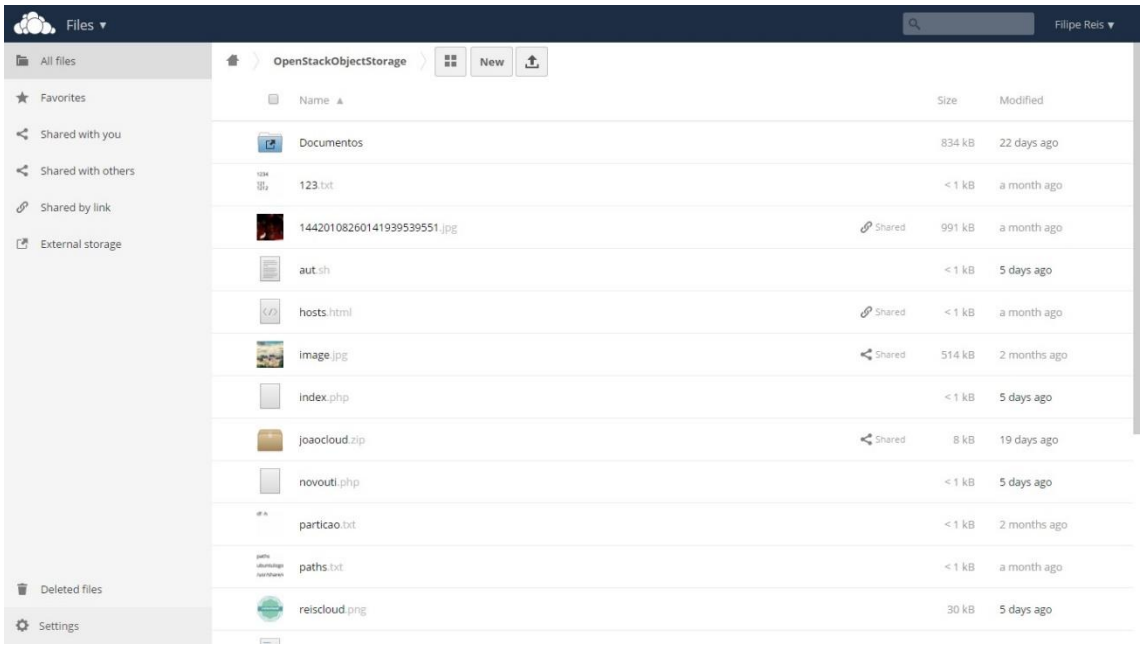


Figura 8 - Interface Web ownCloud

Fonte: Fonte própria

3. Metodologia e Resultados Esperados

3.1 Metodologia

Para a realização deste projeto foi desenvolvido o estudo das tecnologias *open-source* que permitem o desenvolvimento de ambientes Cloud, verificando se cada uma destas tecnologias cumpria os requisitos necessários para a implementação da solução desejada. Após este estudo foi escolhida a que melhor se enquadrava nestes requisitos (Capítulo 2). De seguida foi estudada a arquitetura e funcionamento desta tecnologia, para desenhar na primeira fase, uma arquitetura modelo, com vista o fornecimento dos serviços Cloud desejados. Por fim, foi implementada e testada uma solução que fornece estes serviços, de acordo com as limitações de *hardware* existentes no ambiente. Assim sendo, a metodologia utilizada para implementar e testar a solução foi a seguinte:

- Análise dos requisitos do projeto;
- Estudo de tecnologias *Open-Source* utilizadas para o desenvolvimento de ambientes Cloud (OpenStack, CloudStack);
- Decidir que tecnologia melhor se enquadrava nos requisitos necessários à implementação da solução;
- Estudo da arquitetura do OpenStack;
- Desenvolvimento de uma arquitetura modelo segundo a tecnologia OpenStack;
- Verificação dos recursos computacionais existentes e distribuição dos componentes OpenStack entre estes de forma a suportar a solução pretendida;
- Desenho e configuração da rede;
- Implementação do OpenStack e realização de testes cada vez que um componente era implementado;
- Criação de scripts *bash* de forma a simplificar a gestão e interação com a API do OpenStack;
- Implementação da plataforma ownCloud para funcionar como *frontend* do armazenamento de dados da Cloud;
- Implementação de ferramentas de monitorização da rede e gestão dos *hosts*;

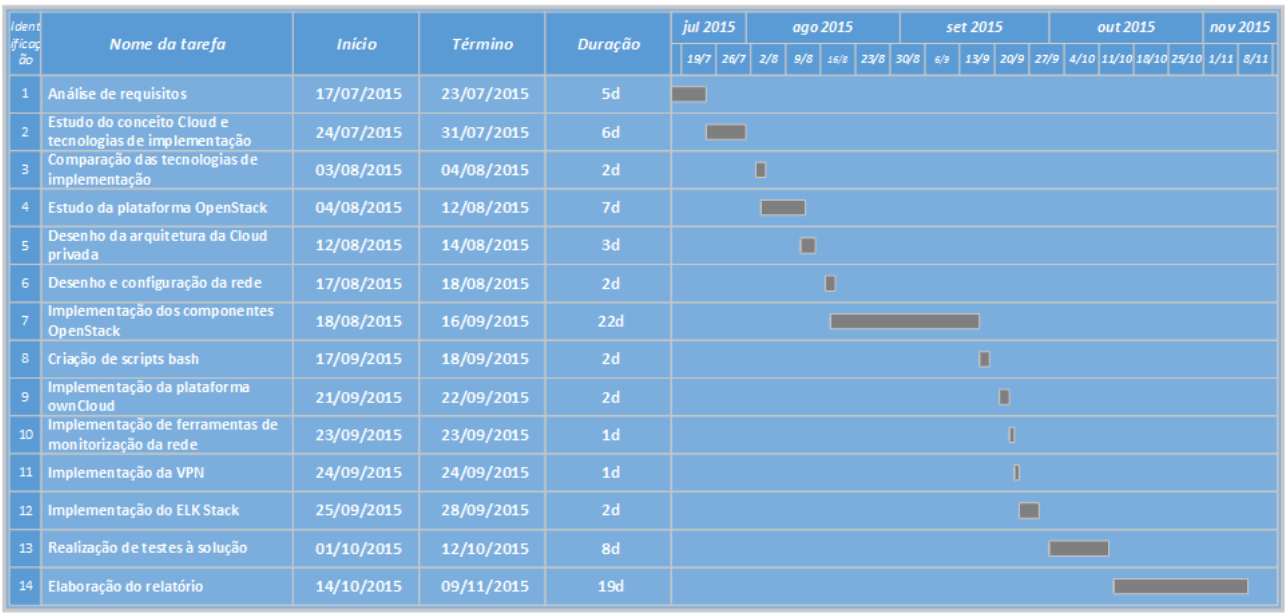
- Criação de uma VPN para acesso à Cloud a partir de qualquer local com ligação à Internet;
- Implementação do ELK Stack para gestão e visualização dos *logs*;
- Realização de testes à solução.

3.2 Descrição das tarefas

As tarefas principais foram:

- Tarefa 1 – Análise dos requisitos do projeto;
- Tarefa 2 – Estudo do conceito Cloud e tecnologias de implementação;
- Tarefa 3 – Comparação e escolha da tecnologia que melhor se adequava ao projeto;
- Tarefa 4 – Estudo da plataforma OpenStack e elaboração de uma arquitetura modelo;
- Tarefa 5 – Verificação dos recursos computacionais disponíveis e distribuição dos componentes OpenStack nestes;
- Tarefa 6 – Desenho e configuração da rede;
- Tarefa 7 – Implementação dos componentes OpenStack;
- Tarefa 8 – Criação de scripts em *bash*;
- Tarefa 9 – Implementação da plataforma ownCloud;
- Tarefa 10 – Implementação de ferramentas de monitorização da rede;
- Tarefa 11 – Implementação de uma VPN;
- Tarefa 12 – Implementação do ELK Stack;
- Tarefa 13 – Realização de testes à solução;
- Tarefa 14 – Elaboração do relatório.

A Figura 9 apresenta o Mapa de Gantt que ilustra o desenvolvimento das várias tarefas.



Identificação	Nome da tarefa	Início	Término	Duração
1	Análise de requisitos	17/07/2015	23/07/2015	5d
2	Estudo do conceito Cloud e tecnologias de implementação	24/07/2015	31/07/2015	6d
3	Comparação das tecnologias de implementação	03/08/2015	04/08/2015	2d
4	Estudo da plataforma OpenStack	04/08/2015	12/08/2015	7d
5	Desenho da arquitetura da Cloud privada	12/08/2015	14/08/2015	3d
6	Desenho e configuração da rede	17/08/2015	18/08/2015	2d
7	Implementação dos componentes OpenStack	18/08/2015	16/09/2015	22d
8	Criação de scripts bash	17/09/2015	18/09/2015	2d
9	Implementação da plataforma ownCloud	21/09/2015	22/09/2015	2d
10	Implementação de ferramentas de monitorização da rede	23/09/2015	23/09/2015	1d
11	Implementação da VPN	24/09/2015	24/09/2015	1d
12	Implementação do ELK Stack	25/09/2015	28/09/2015	2d
13	Realização de testes à solução	01/10/2015	12/10/2015	8d
14	Elaboração do relatório	14/10/2015	09/11/2015	19d

Figura 9 - Mapa de Gantt

Fonte: Fonte própria

3.3 Resultados esperados

Após a implementação da solução esperam-se os seguintes resultados:

- Facilidade na criação de máquinas virtuais unilateralmente, a pedido do utilizador, onde os recursos são garantidos;
- Facilidade no armazenamento e partilha de dados entre vários utilizadores;
- Redundância e alta disponibilidade dos dados;
- Escalabilidade dos recursos computacionais consoante a necessidade do utilizador;
- Gestão centralizada;
- Acesso à Cloud a partir de qualquer local através da Internet;
- Monitorização da rede.

4. Tecnologias utilizadas

Neste capítulo são identificadas e caracterizadas as tecnologias utilizadas ao longo do projeto.

4.1 OpenStack

O OpenStack tem suporte para todos os tipos de ambiente Cloud. A tecnologia tem uma arquitetura "modular" com vários componentes que se inter-relacionam para proporcionar os serviços desejados, cada componente tem um "nome de código*"* e uma API própria de forma a facilitar a sua integração. Relativamente à distribuição escolhida, optou-se pela "Kilo", lançada em 30 de Abril de 2015.

De seguida é apresentado e descrito cada um dos componentes/serviços escolhidos e tecnologias complementares.

4.1.1 MySQL

O MySQL é um sistema de gestão de bases de dados *Open-Source* que utiliza a linguagem SQL. Foi utilizado no projeto para armazenar dados de componentes/serviços do OpenStack, cada um destes componentes tem uma base de dados própria armazenada no MySQL *Server*, com a exceção do Ceilometer.

4.1.2 RabbitMQ

O RabbitMQ é um mediador de mensagens *open-source* que implementa o protocolo AMQP[4]. O RabbitMQ é a tecnologia responsável pela troca de mensagens e comunicação entre *nodes* dentro do *cluster* OpenStack [42].

4.1.3 Keystone

O Keystone é o componente responsável pela autenticação e autorização, de todos os componentes e utilizadores integrados no ambiente Cloud. Implementa políticas de gestão e catalogação de serviços, permite o registo de projetos e utilizadores. Autenticação dos utilizadores e fornecimento de *tokens*[5] para autorização, permitindo o acesso destes aos recursos computacionais.

[4] AMQP – É um protocolo de gestão de filas de espera de mensagens, que permite interoperabilidade entre aplicações distintas.
[5] *Token* – É uma *string* alfa numérica de texto usada para aceder aos recursos e APIs do OpenStack.

É também responsável pela catalogação e gestão dos *endpoints*[6] dos componentes [43].

A autenticação permite ao Keystone reconhecer quem é o utilizador, que confirma se o utilizador é quem diz ser através da validação de credenciais fornecidas pelo utilizador (nome de utilizador e *password*), após confirmada a identidade do utilizador fornece um *token* que permite o pedido de recursos computacionais, este *token* também determina os privilégios deste utilizador no sistema (por exemplo administrador, utilizador).

4.1.4 Glance

O Glance é o componente que fornece serviços de registo, upload e gestão de imagens de servidores ou sistemas operativos. Tem a capacidade de copiar e armazenar *snapshots* (backups) das máquinas virtuais. Estas imagens podem ser armazenadas numa localização específica ou no componente de armazenamento do OpenStack (Swift) [44].

4.1.5 Nova

O Nova é o componente de computação do OpenStack, desenhado para fornecer acesso a recursos computacionais escaláveis, de forma partilhada e com autosserviço a pedido. O Nova é o serviço responsável pela criação e gestão das máquinas virtuais, com suporte para várias tecnologias de virtualização. Interage com o MySQL *Server* para armazenar os dados relativos aos seus serviços, Keystone para autenticação, Glance para acesso às imagens dos sistemas operativos que vão ser executados nas máquinas virtuais e com o Horizon para uma *interface web* de gestão destas. A sua arquitetura é desenhada para escalar horizontalmente em *hardware standard.* O projeto Nova consiste em vários subcomponentes, cada um com uma função específica. De seguida são identificados e caracterizados os subcomponentes utilizados no projeto [45]:

- **Nova-api service -** Responde e aceita chamadas da API por parte do utilizador. Permite também compatibilidade com a API EC2 da AWS;

[6] *Endpoint* – Trata-se de um endereço de rede, por norma *URL*, para o acesso a um determinado serviço.

- **Nova-compute service** - É o serviço responsável por criar ou terminar as máquinas virtuais através de APIs de *hypervisors* (por exemplo, libvirt para KVM/QEMU, XenAPI para XenServer, entre outras), o *hypervisor* escolhido foi o KVM/QEMU (que é um "*hypervisor* do tipo 2"[7]). O nova-compute interage com a base de dados para criar, atualizar o estado ou terminar uma máquina virtual;
- **Nova-cert daemon** – Permite a criação e gestão de certificados x509 (por exemplo, chaves públicas e privadas para acesso às máquinas virtuais via SSH);
- **Nova-conductor module -** É o mediador de interações entre o Nova e a base de dados. É responsável pela ligação do nova-compute à base de dados alocada no MySQL *Server*;
- **Nova-consoleauth daemon -** Fornece *tokens* para os utilizadores serem autorizados a aceder ao *proxy* das consolas virtuais das máquinas virtuais;
- **Nova-network -** Permite a criação de redes privadas para as máquinas virtuais, existem dois tipos de endereços IP atribuídos às máquinas virtuais. Os endereços IP privados, que são atribuídos à máquina virtual na sua criação, consoante a rede privada a que esta pertence. Endereços IP "públicos" (*Floating* IP*)*, que permitem que estas máquinas possam ser acedidas desde o exterior da rede (podem variar consoante a preferência do utilizador). A alocação dos endereços IP pode ser dinâmica (DHCP) ou estática. O Nova-network também faz a tradução dos endereços públicos para privados e vice-versa (NAT), das máquinas virtuais. Permite a criação de regras de *firewall*, criação de VLANs e implementa redundância relativamente à rede, quando distribuído entre todos os nodes que façam computação.
- **Nova-novncproxy daemon -** Permite o acesso às máquinas virtuais em execução, através de uma ligação VNC via *interface web* (Horizon);
- **Nova-scheduler service -** É o serviço responsável por receber e gerir filas de espera de pedidos relativos às máquinas virtuais. Determina também em que *host* estas irão ser executadas.

[7] *Hypervisor* Tipo 2 (Hosted) – Este tipo de *hypervisor* é executado por cima do sistema operativo.

4.1.6 Cinder

O Cinder é o componente que garante o armazenamento persistente de uma máquina virtual (volume) e interage com o Nova para fornecer e gerir volumes. Também permite a criação de *snapshots* dos volumes. O Cinder funciona com base num gestor de volumes, como o LVM ou Ceph, optou-se pelo LVM pela maior simplicidade de implementação. Outra vantagem do Cinder é a facilidade de migrar máquinas virtuais de *host* quando estas falharem, garantindo assim redundância, visto que por norma os volumes de dados estarão armazenados em *nodes* diferentes do *host* onde as máquinas virtuais estão em execução. O Cinder consiste em quatro subcomponentes [41]:

- **Cinder-api –** Aceita pedidos da API e encaminha-os para o Cinder-volume para efetuar o pedido;
- **Cinder-volume –** Responde a pedidos de leitura e escrita enviados para o Cinder. Interage com o gestor de volumes escolhido para o armazenamento dos dados das máquinas virtuais;
- **Cinder-scheduler daemon –** Funciona de forma semelhante ao nova-scheduler, seleciona o melhor *node* para armazenar um volume;
- **Cinder-backup daemon –** Permite a criação de *backups* dos volumes de forma simples e rápida através da API. Estes *backups* poderão ser armazenados no serviço de armazenamento de objetos do OpenStack (Swift), se o Cinder for assim configurado.

4.1.7 Swift

O Swift é o componente responsável pelo armazenamento de objetos. É altamente escalável e pode gerir grandes quantidades de dados através de uma API RESTful HTTP.

Implementa um princípio de "*consistent hashing*[8]*"* para a criação de *rings*. Cada *ring* representa o espaço de computação de todos os valores *hash* divididos em partes equivalentes, onde cada parte deste espaço é chamada de partição. Por exemplo, se quisermos armazenar objetos e distribui-los por 3 nodes, teremos que dividir o espaço *hash* em 3. Ou seja, iriamos ter 3 partições no mínimo, mas como queremos ter um serviço facilmente escalável e garantir suavidade na movimentação de dados pela rede, convém termos bastantes mais partições do que as necessárias, de forma a manter estabilidade no mapeamento *hash* [46].

A arquitetura do OpenStack Swift é dividida em três categorias de *rings*: conta, contentor e objetos. Uma conta contém contentores, e cada contentor contém objetos. Cada vez que um componente precisa de efetuar uma operação relativa a um objeto, contentor ou conta, este precisa de interagir com o *ring* apropriado de forma a determinar a sua localização no *cluster.* Cada *ring* tem um mapeamento do tipo: zona, dispositivo físico, endereço IP para envio da réplica, peso desse *node* (prioridade) e número de partições (que serão distribuídas consoante o peso desse *node*) [46].

Ao criar estes três *rings* e ao adicionar cada *node* a estes *rings,* é criada a "tabela" de *rings,* que irá conter os possíveis *nodes*, que irão receber várias réplicas (configurado na implementação) de cada objeto que seja recebido. É graças à criação destes *rings* que é implementada a redundância e alta disponibilidade dos dados.

O Swift consiste em quatro subcomponentes [41][46]:

- **Swift-proxy-server** – O servidor proxy é o responsável por responder a pedidos de *download*/*upload* de ficheiros e criação de contentores. Para cada pedido irá verificar os *rings* e encaminhar as várias réplicas para os *nodes* escolhidos;
- **Swift-account-server** – Faz a gestão das contas definidas no serviço de armazenamento de dados;
- **Swift-container-server –** Faz a gestão do mapeamento dos contentores (diretórios) definidos no serviço de armazenamento de dados;
- **Swift-object-server** – Faz a gestão dos objetos (ficheiros) nos *nodes* de armazenamento definidos.

Para a sincronização de ficheiros e diretórios entre *nodes* foi implementado o rsync que é um *software open-source* utilizado em sistemas Unix para esta finalidade.

[8] Hashing – O conceito de hashing baseia-se na transformação de uma grande quantidade de dados numa pequena quantidade de informações.

4.1.8 Horizon

O Horizon é a *Dashboard* do OpenStack, uma *interface web* que permite uma gestão simples e centralizada de todos os serviços e recursos disponíveis no projeto. É possível personalizar o seu aspeto (seja colocar o logotipo da empresa ou alterar as cores). Um utilizador comum tem apenas acesso aos recursos disponíveis no(s) projeto(s) que está integrado, enquanto que o administrador tem acesso à visualização e gestão de todo o ambiente implementado.

O servidor *web* escolhido para a implementação do Horizon foi o Apache, que é um dos mais populares em todo o mundo.

4.1.9 Ceilometer

O Ceilometer é o componente responsável criação de métricas de dados e criação de alarmes para melhor monitorização e gestão do ambiente. Estas métricas podem recolher informação relativa a vários recursos/serviços dentro do ambiente Cloud, podendo também servir para a criação de estatísticas, de forma a facilitar a monitorização do ambiente.

O Ceilometer utiliza uma base de dados NoSQL, que no fundo é uma base de dados não relacional com finalidade de armazenar grandes quantidades de dados, que não tenham grande complexidade. Para tal foi utilizado o MongoDB [41].

4.1.10 MongoDB

O MongoDB é uma plataforma de bases de dados orientada à documentação (armazenamento de grandes quantidades de dados) *Open-Source* do tipo NoSQL. O MongoDB é utilizado para armazenar os dados das métricas obtidas pelo Ceilometer.

4.1.11 Heat

O Heat é o componente responsável por "orquestrar" a Cloud de forma mais eficiente, através da criação de *templates* que permitem a automação dos serviços, aplicações e recursos presentes na infraestrutura. Estes *templates* podem ser escritos nos formatos YAML ou JSON, que viabilizam a criação/gestão de *stacks*[9], auto elasticidade e alta disponibilidade das máquinas virtuais. Viabiliza também uma *AWS API Query* compatível com a *AWS CloudFormation* [41][47].

4.2 GNU/Linux

GNU/Linux é o termo utilizado para referir sistemas operativos que utilizam o *kernel* Linux, desenvolvido por Linus Torvalds. O OpenStack trabalha em cima de uma distribuição GNU/Linux, sendo a mais popular e com maior suporte o Ubuntu da Canonical, por esses motivos escolheu-se a versão Ubuntu 14.04 LTS para sistema operativo dos *hosts* da Cloud [48].

4.2.1 Bash

O Bash é uma linguagem de comandos Unix que permite a criação de *scripts*, que são sequências de comandos/instruções a serem executadas pela *shell*. É utilizada a "tag" #!/bin/bash para a *shell* saber que interpretador utilizar. Durante o desenvolvimento do projeto foram criados scripts de forma a facilitar a interação e gestão do OpenStack através da sua API.

4.3 Rede/Segurança

4.3.1 OpenVPN

O OpenVPN é um *software open-source* para a criação de VPNs. Foi implementado para que qualquer utilizador da Cloud possa aceder à rede e aos recursos de forma privada e segura, o acesso é estabelecido através da criação de um *tunnel* cifrado. Foram também criadas regras na *firewall* (IPTables) onde está hospedado o servidor OpenVPN para permitir e encaminhar o tráfego vindo deste *tunnel* para a rede [49].

[9] Stacks – Trata-se de um grupo de objetos/recursos que irão ser criados pelo Heat definidos nos templates.

4.3.2 ClamAV

O ClamAV é um antivírus *open-source* desenvolvido inicialmente pela Sourcefire (que foi adquirida pela Cisco em 2013) para sistemas Unix. Apesar dos sistemas operativos Linux serem mais seguros e existirem menos ameaças é aconselhável ter pelo menos um antivírus a ser executado nos *hosts*. Para tal foi instalado o ClamAV em todos os *hosts* da Cloud [50].

4.4 Monitorização e Gestão da rede

4.4.1 Nagios

O Nagios é uma ferramenta *open-source* que permite monitorizar sistemas, serviços e redes. A monitorização é feita através da utilização de diversos protocolos (ICMP, SSH, HTTP, SNMP, entre outros). Sempre que algum serviço ou equipamento falha o Nagios envia um alerta, que pode ser configurado para que seja enviado para o *mail*, de forma a não ser necessário estar sempre a verificar a *interface web* ou *logs*, para estar ao corrente do estado do ambiente. Foi implementado para facilitar a gestão da Cloud e monitorização desta [51].

4.4.2 Cacti

O Cacti é uma ferramenta *open-source* de monitorização de serviços e redes semelhante ao Nagios com a diferença que os dados recolhidos pelo Cacti, são utilizados para gerar gráficos relativos a vários elementos (por exemplo, largura de banda que está a passar numa *interface*, uso de CPU, entre outros). Foi implementado para facilitar a gestão da Cloud e monitorização desta [52].

4.4.3 Ajenti

O Ajenti é uma ferramenta *open-source* para gestão gráfica de servidores. Permite gerir vários serviços como Apache, Cron, Firewall, MySQL, entre outros. Permite ainda gerir processos, configurações de rede, aceder a *logs* através de uma *interface web*. Foi implementado para facilitar a gestão dos *hosts* da Cloud.

4.5 Plataforma de Armazenamento

4.5.1 ownCloud

O ownCloud, como referido anteriormente (Capítulo 2), é um *software open-source* que permite o armazenamento e sincronização de ficheiros, com uma interação muito semelhante à Dropbox. Foi implementado com a finalidade de trabalhar como *frontend* do OpenStack Swift, de forma a simplificar e melhorar, o serviço de armazenamento e partilha de dados. O ownCloud também é utilizado para encriptar os dados alojados na Cloud.

4.6 Armazenamento e gestão de *Logs*

4.6.1 Rsyslog

O Rsyslog é uma ferramenta *open-source* para sistemas Unix que permite o envio de *logs* sobre uma rede IP. Funciona com base no syslog, mas com funcionalidades mais avançadas, como maior capacidade de filtragem e transporte via TCP/UDP. É utilizado para enviar os *logs* de vários serviços de todos *nodes* da Cloud para um servidor onde estes serão armazenados.

4.6.2 ELK Stack

O ELK Stack consiste na junção três ferramentas bastantes populares para a gestão de *logs*, que são estas o Elastic Search, o Logstash e o Kibana.

O Elastic Search foi utilizado para fazer pesquisas rápidas e eficientes aos *logs*. O Logstash é utilizado para receber, armazenar, filtrar e analisar os *logs*. O Kibana fornece uma *interface web* para visualização e gestão dos *logs* [53].

5. Implementação da Solução

5.1 Arquitetura modelo

A arquitetura de *cluster* OpenStack é desenhada para ser horizontalmente escalável em grande escala. Os *nodes* deste *cluster* podem ser denominados dos seguintes tipos:

- ***Controller node* (Controlador)** - O Controlador pode ser considerado como o operador do ambiente Cloud, recebe pedidos e envia tarefas a desempenhar aos outros *nodes.* Também é o responsável por fornecer uma gestão centralizada do sistema Cloud via API ou *Dashboard*. Por norma faz a gestão de vários serviços como, bases de dados, filas de mensagens, autenticação, *endpoints*, agendamento e escolha dos *nodes* a efetuar determinadas tarefas.
- ***Task node* (Nó de tarefa)** - Este tipo de *node* tem funções específicas dentro do *cluster*, tais como, computação de máquinas virtuais, armazenamento de ficheiros ou volumes de dados, atribuição e gestão de serviços de rede entre outras.

Os serviços presentes no ambiente deverão estar separados e bem distribuídos, de forma a garantir alta disponibilidade e redundância destes. Poderão ser utilizados balanceadores de cargas externos para dividir a carga dos serviços por múltiplos servidores, ou garantir redundância caso alguns falhem. Por norma uma implementação OpenStack já contém alguns componentes para fazer balanceamento de cargas, como *proxys* a nível do serviço de armazenamento e rede [37].

Um fornecedor de serviços Cloud poderá ter mais que um *datacenter*, onde dentro deste poderão existir vários *clusters*, a distribuição e finalidade destes é relativo aos serviços que se pretendem garantir.

De seguida é apresentado um diagrama de uma arquitetura modelo para a implementação de um ambiente Cloud que garanta redundância e alta disponibilidade dos seus serviços (Figura 10).

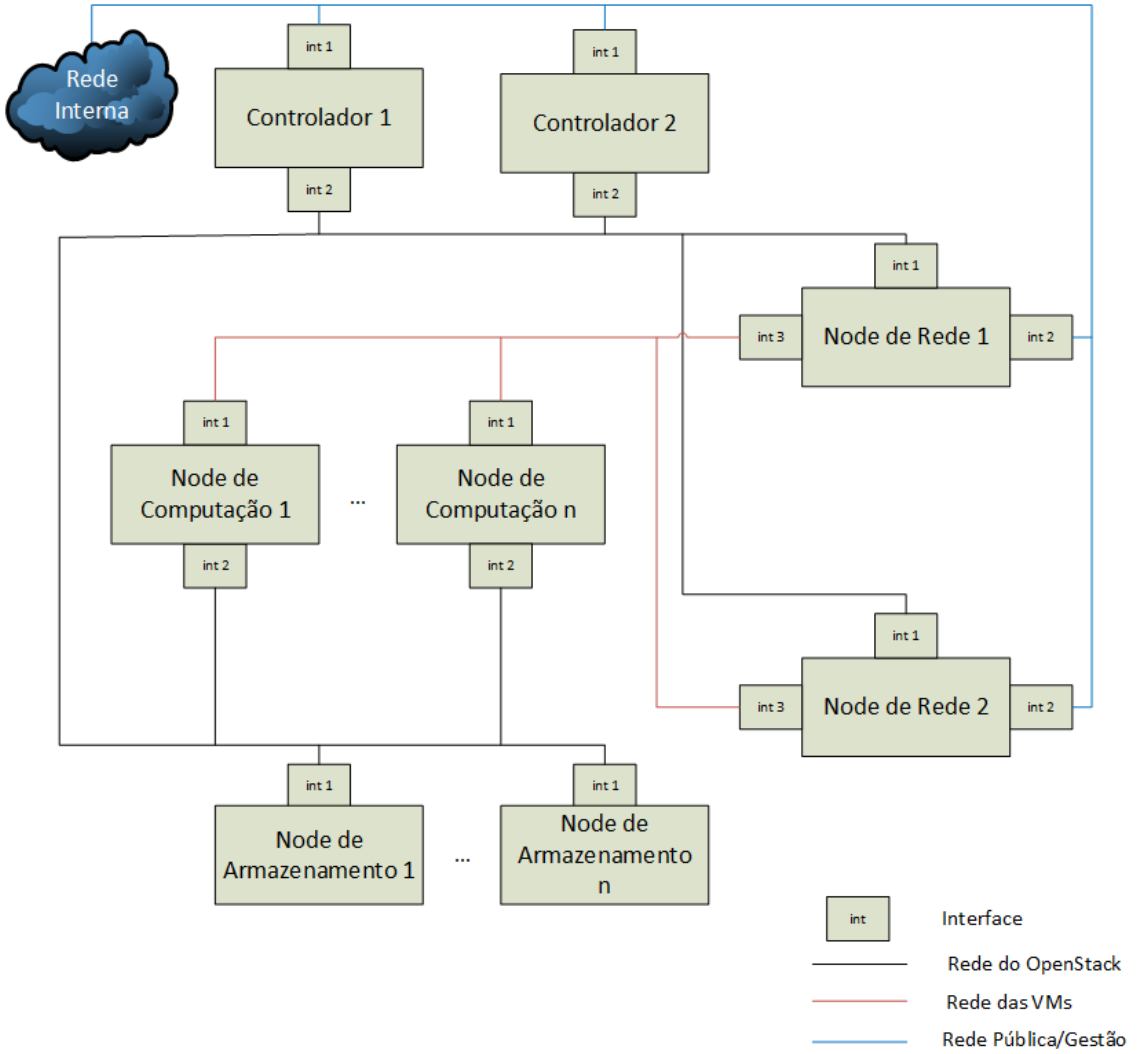


Figura 10 - Arquitetura modelo OpenStack

Fonte: Fonte própria

Legenda:

- ***Node* de Rede (*Network Node)*** - Este *node* é responsável pela gestão de todas as redes virtuais através de *routers* lógicos e *virtual switches*, permite também integrar serviços como *firewalls*, balanceadores de cargas e VPNs.
- ***Node* de Armazenamento (*Storage Node)*** - Este *node* pode ter três finalidades, armazenamento de ficheiros, armazenamento de volumes das máquinas virtuais e armazenamento das imagens dos sistemas operativos. Para alto desempenho, cada um destes *nodes* apenas deverá integrar uma destas finalidades.

- ***Node* de Computação *(Compute Node)*** - Este *node* apenas faz a computação das máquinas virtuais segundo o *hypervisor* definido. O armazenamento destas máquinas virtuais estará alojado em *nodes* diferentes (*nodes* de armazenamento). Logo é garantida redundância, porque no caso de um *node* de computação parar de funcionar, o processo de migração das máquinas virtuais que estivessem a ser executadas nesse *node* torna-se simples, visto que os seus dados não estão a ser alojados no mesmo *node*.

5.2 Análise do *hardware* e Arquitetura da Solução

Antes de iniciar a implementação do ambiente Cloud procedeu-se a análise dos recursos computacionais disponíveis e desenho da arquitetura da solução.

5.2.1 *Hardware*

De forma a fazer uma melhor distribuição dos componentes/serviços OpenStack, foi elaborada uma análise às potencialidades e limitações do *hardware* disponível, para o desenvolvimento deste projeto. De seguida são apresentadas as características do *hardware* (Tabela 2):

Tabela 2 - Características do hardware

	Tipo	Processador	Cores	RAM	Armazenamento	Virtualização assistida por *hardware*
PC-1	*Desktop*	i5 4460 3.2 GHz	4	16 GB	SSD 128 GB	VT-x, VT-D, EPT
PC-2	Portátil	Intel T4200 2 GHz	2	4 GB	HDD 250 GB + HDD 1 TB	Não
PC-3	Portátil	Intel T2600 2.16 GHz	2	2 GB	HDD 128 GB + HDD 1 TB	Não
PC-4	*Desktop*	AMD Athlon 1.25 GHz	1	1 GB	HDD 30GB + HDD 40GB	Não

5.2.2 Arquitetura da Solução

Com o *hardware* disponível, houve a necessidade de emparelhar componentes/serviços OpenStack. Foi desenhada uma arquitetura com a melhor distribuição possível, tendo em conta os recursos computacionais disponíveis. No entanto, devido a limitações de *hardware*, apenas foi possível implementar um Controlador para o ambiente, o que terá como precedente um ponto de falha central, isto é, se o Controlador falhar, automaticamente os outros serviços na Cloud deixam de funcionar, até o Controlador estar novamente operacional.

A distribuição ideal de componentes foi a seguinte:

- **PC-1 (Controlador)** – O PC-1 foi escolhido para Controlador, armazenamento de volumes e computação de máquinas virtuais, por ser o PC com maior poder computacional e é o único que o processador suporta tecnologias de virtualização assistida por *hardware* (VT-x, VT-d, EPT)*;*
- **PC-2 (Nova)** - O PC-2 foi escolhido para armazenamento de volumes, armazenamento de dados e computação de máquinas virtuais;
- **PC-3 (Swift)** – O PC-3 foi escolhido para armazenamento de dados, implementação do ownCloud e o servidor OpenVPN;
- **PC-4 (Gestor)** – O PC-4 foi escolhido para armazenamento de dados e implementação de ferramentas de monitorização (Nagios, Cacti).

A Figura 11 apresenta a distribuição dos serviços pelos *nodes* da Cloud.

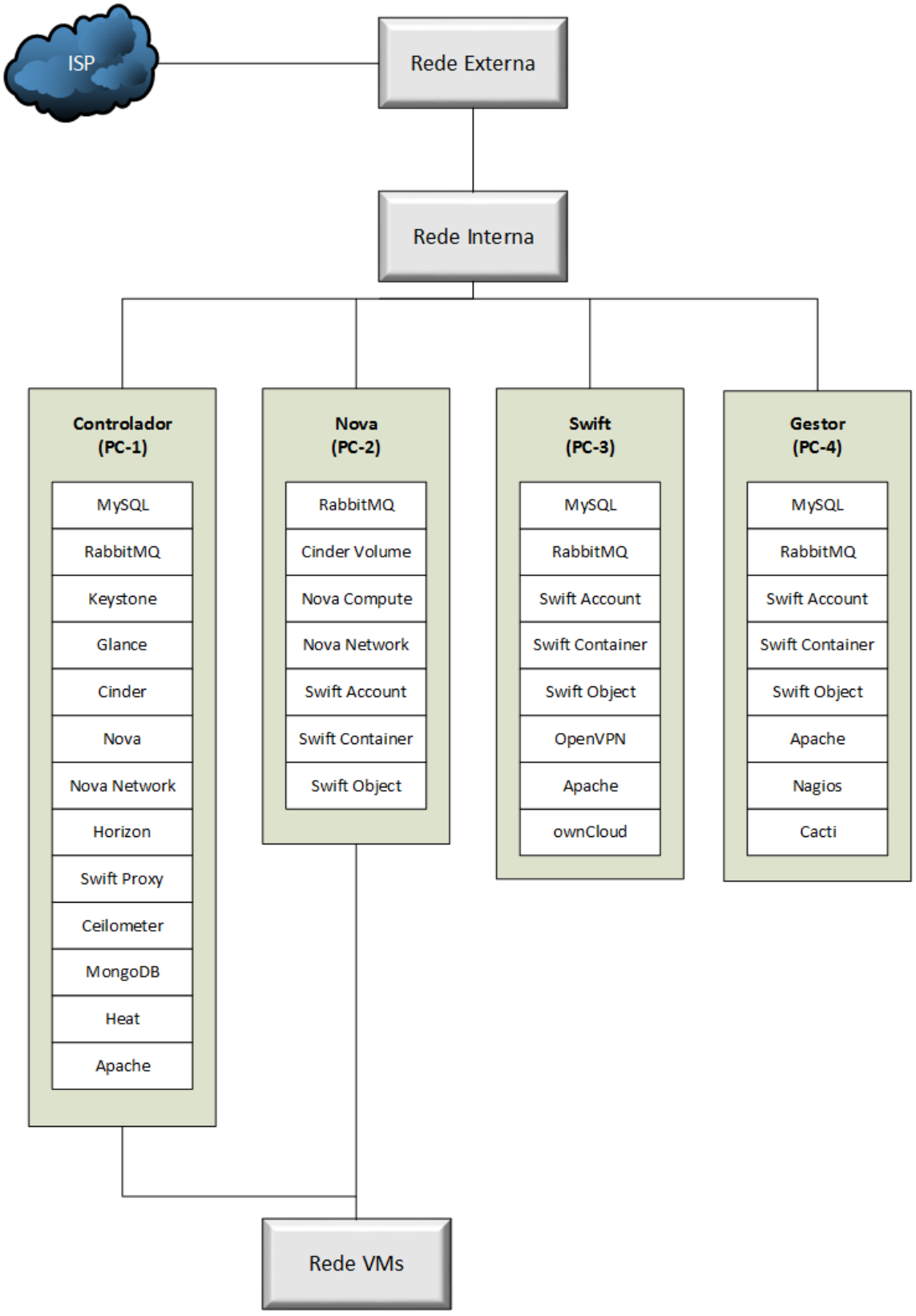


Figura 11 - Arquitetura da Cloud Privada

Fonte: Fonte própria

5.3 Configuração da Rede

Como representado na Figura 11, o ambiente Cloud consiste em três redes e um *tunnel* VPN para permitir o acesso aos recursos computacionais a partir do exterior da Cloud. As Figura 12 e Figura 13 apresentam o diagrama da rede.

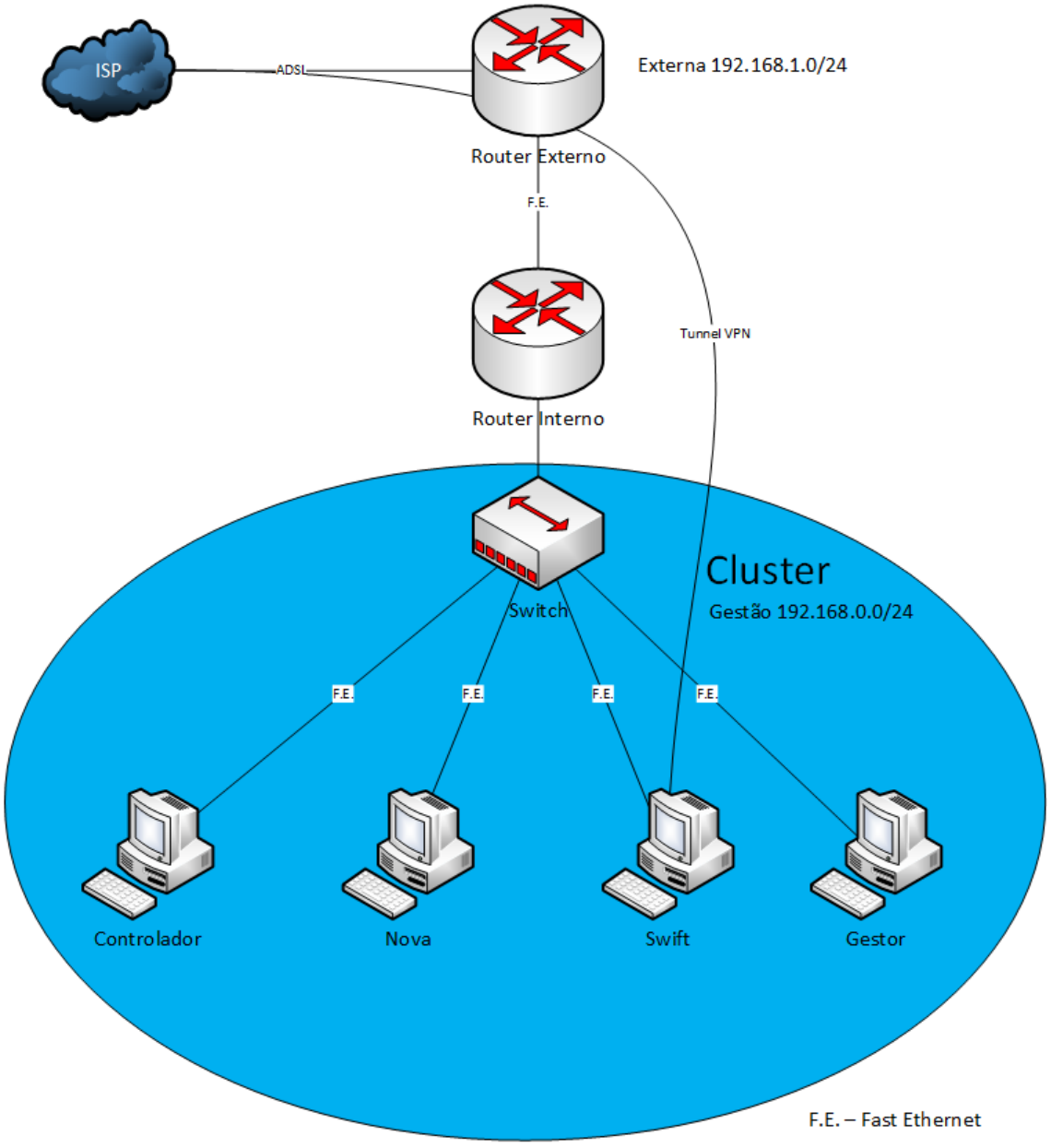


Figura 12 - Diagrama da rede

Fonte: Fonte própria

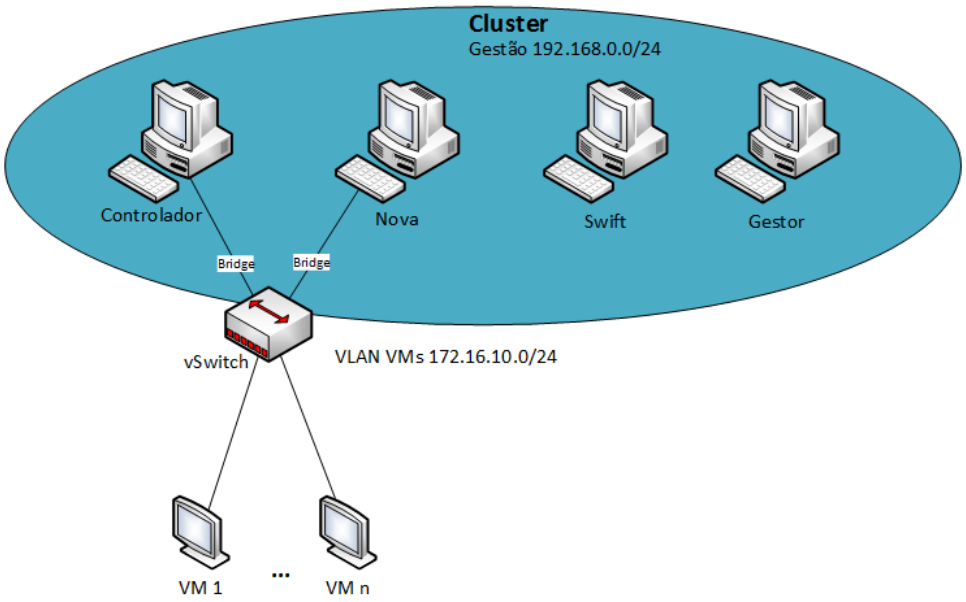


Figura 13 - Diagrama do Cluster

Fonte: Fonte própria

Esta solução foi implementada num ambiente residencial, logo foi necessário a criação de uma rede de gestão através da adição de mais um *router*, de forma a isolar a Cloud da rede externa, que permite apenas o acesso à Internet de familiares e dos *hosts* da Cloud. Para tal foi criada uma *Access List* no *router* interno, que nega o tráfego entre os *hosts* da rede externa e rede de gestão, permitindo apenas o tráfego dos *hosts* da Cloud para a Internet.

As redes criadas, o equipamento de rede utilizado e os endereços IP atribuídos foram os seguintes (*Tabela 3*, *Tabela 4*, *Tabela 5*):

Tabela 3 - Endereços das redes

	Rede	Mascara	Gateway	*Hosts* Disponíveis	Interface
Rede Externa	192.168.1.0	255.255.255.0	192.168.1.1	254	--
Rede Gestão	192.168.0.0	255.255.255.0	192.168.0.1	254	--
Rede VMs	172.16.10.0	255.255.255.0	172.16.10.1	254	br100
Tunnel VPN	10.8.0.0	255.255.255.0	10.8.0.1	254	tun0

Tabela 4 - Equipamento ativo de rede

	Modelo	Interfaces	Rede	IP	Características	Segurança
Router Interno	Technicolor TG582n	4 x 10/100 Mbps LAN	192.168.1.0/24	192.168.1.1	QoS, DHCP, Port-Forward	IDS, SPI Firewall
Router Externo	TP-Link TL-WR740N	4 x 10/100Mbps LAN + 1 x 10 10/100Mbps WAN	192.168.0.0/24	192.168.0.1	QoS, DHCP, Port-Forward, ACLs	DoS, SPI Firewall
Switch	TP-Link TL-SF1008D	8 x 10/100 Mbps	192.168.0.0/24	--	--	--

Tabela 5 - Endereços IP atribuídos

	Rede	IP	Gateway	Interface
Controlador	192.168.0.0/24	192.168.0.41	192.168.0.1	eth0
Nova	192.168.0.0/24	192.168.0.40	192.168.0.1	eth0
Swift	192.168.0.0/24	192.168.0.42	192.168.0.1	eth0
Gestor	192.168.0.0/24	192.168.0.43	192.168.0.1	eth0

Foi instalada a distribuição Linux Ubuntu 14.04 nestes quatro *hosts*, a configuração da rede foi feita previamente ao início da implementação do OpenStack. Foi levado em conta que a atribuição dos endereços IP nestes hosts, teve que ser feita estaticamente, porque o OpenStack não suporta aplicações gráficas como o Network Manager. Logo a configuração da rede foi efetuada no ficheiro /etc/network/interfaces (**Anexo A1**). Foi testado o cenário recorrendo ao protocolo ICMP, enviando *pings* para todos os *nodes* e para o exterior da rede, a partir de cada um destes e recebendo as respetivas respostas aos *pings*.

5.4 Implementação do OpenStack

Após a configuração da rede em todos os *nodes* procedeu-se à implementação do ambiente Cloud através da tecnologia OpenStack. Toda a implementação foi feita através da documentação disponibilizada pelo OpenStack [36][38][39][40][41].

Para a implementação e configuração de *clusters* OpenStack, é recomendado o uso de ferramentas de gestão e configuração, como o Chef, Puppet, Juju, Ansible, entre outras. Na implementação desta solução não existiu necessidade de utilizar uma destas ferramentas devido à pequena dimensão do *cluster*.

Primeiro foi instalado o arquivo ubuntu-cloud-keyring e adicionado o repositório da distribuição Kilo em todos os *nodes*.

- *apt-get install ubuntu-cloud-keyring*
- *echo "deb http://ubuntu-cloud.archive.cannonical.com/ubuntu" \ "trusty-updates/kilo main" > /etc/apt/sources.list.d/cloudarchive-kilo.list*
- *apt-get update && apt-get dist-upgrade*

5.4.1 Instalação e Configuração do MySQL Server

Como grande parte dos serviços utilizam uma base de dados SQL para armazenar a informação, foi instalado o MySQL *Server* e suporte deste com Python no Controlador.

- *apt-get install mysql-server python-mysqldb*

De seguida foi editada a configuração do MySQL *Server* de forma a ficar associado no endereço IP do Controlador (por omissão trabalha no *localhost*), permitindo assim o acesso dos *nodes* à base de dados. Foi também adicionado o motor de armazenamento InnoDB[10] e ativada a codificação UTF-8, a partir da adição das seguintes linhas à secção **[mysqld]** do ficheiro de configuração /etc/mysql/my.cnf (Figura 14):

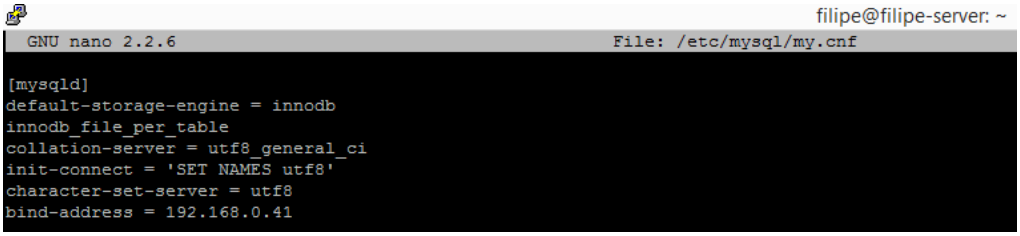


Figura 14 - Configuração MySQL

Fonte: Fonte própria

[10] InnoDB – É um motor de armazenamento MySQL desenhado para garantir alto desempenho e alta fiabilidade.

5.4.2 Instalação e configuração do RabbitMQ

Foi instalado o RabbitMQ para coordenar operações e receber/enviar informações de estados, entre os vários serviços OpenStack. Foi adicionado o utilizador openstack (Figura 15) e garantidas permissões de acesso, leitura e escrita através dos seguintes comandos:

- *apt-get install rabbitmq-server*
- *rabbitmqctl add_user openstack <password>*
- *rabbitmqctl set_permissions openstack ".*" ".*" ".*"*

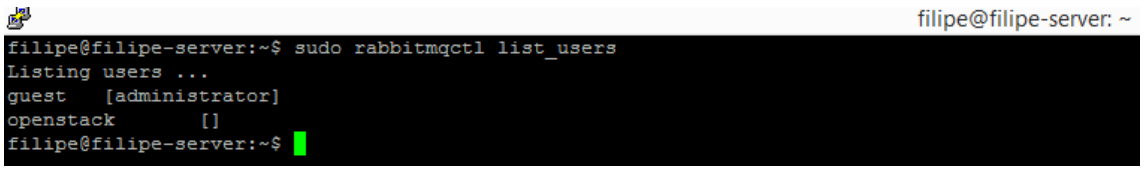


Figura 15 – Verificação das configurações feitas ao RabbitMQ

Fonte: Fonte própria

5.4.3 Instalação e configuração do Keystone

Antes de instalar e configurar o Keystone foi necessário criar a base de dados para este serviço e atribuir privilégios. Para todos os outros componentes que necessitaram base de dados, efetuou-se o mesmo processo, alterando apenas o nome do componente/serviço e *password*:

- *mysql –u root –p*
- *CREATE DATABASE keystone;*
  GRANT ALL PRIVILEGES ON keystone. TO 'keystone'@'localhost' IDENTIFIED BY '<password >';*
- *GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '<password >';*

Após a criação da base de dados foram instalados e configurados os componentes do Keystone. Como o OpenStack é escrito em Python foi necessário proceder à criação de um servidor HTTP com o módulo WSGI[11] ativado. Para servir pedidos nos portos 5000, 35357 e instalação dos respetivos pacotes (**Anexo B1**).

[11] WSGI – Faz a ligação entre servidores *web* e aplicações ou *frameworks* escritas em Python.

Para que fosse possível adicionar utilizadores, projetos e privilégios ao Keystone pela primeira vez, foi necessário efetuar a ligação à base de dados, e adicionar um *token* de *admin* que será eliminado posteriormente por questões de segurança. Assim foi necessário adicionar as seguintes linhas ao ficheiro de configuração /etc/keystone/keystone.conf (Figura 16):

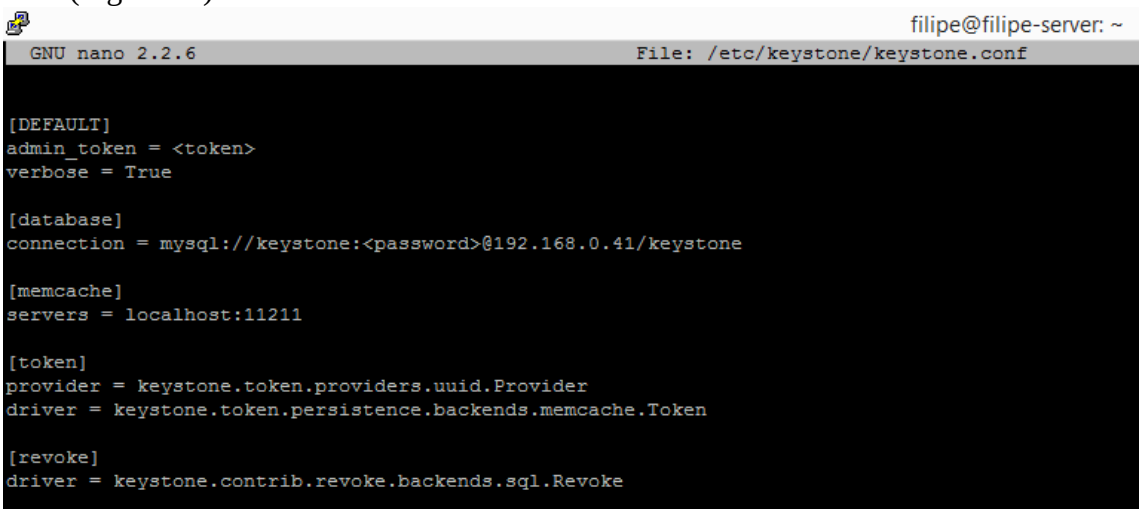


Figura 16 - Configuração Keystone

Fonte: Fonte própria

Nestas linhas é escolhido o *token* de *admin*, feita a ligação à base de dados e adicionado o servidor memcached[12], para maior velocidade nos pedidos feitos ao Keystone. Após a configuração inicial foi necessário "popular" e sincronizar o Keystone à base de dados criada para este serviço com as respetivas tabelas através dos seguintes comandos:

- *su -s /bin/sh -c "keystone-manage db_sync" keystone*
- *service apache2 restart*

De seguida são criadas as variáveis de ambiente relativas ao *token* e *endpoint* para primeira autenticação na API como administrador para adicionar o utilizador *admin* e respetivo projeto para administração da Cloud:

- *export OS_TOKEN = <token>*
- *export OS_URL = http://192.168.0.41:35357/v2.0*

[12] Memcached – O Memcached permite alocar dinamicamente dados e objetos, de vários serviços na RAM de forma mais eficiente.

Criação do serviço de Identidade, que gere os *endpoints* para comunicação de serviços:

- *openstack service create --name keystone --description "OpenStack Identity" identity*
- *openstack endpoint create *
  *--publicurl http://192.168.0.41:5000/v2.0 *
  *--internalurl http://192.168.0.41:5000/v2.0 *
  *--adminurl http://192.168.0.41:35357/v2.0 *
  --region RegionOne identity

Criação do projeto, utilizador e atribuição de privilégios de *admin*:

- *openstack project create --description "Administração" admin*
- *openstack user create --password-prompt admin*
- *openstack role create admin*
- *openstack role add --project admin --user admin admin*

Foi criado posteriormente da mesma forma, o projeto "Service" que contém o utilizador de cada serviço. Foi também adicionado o utilizador e projeto "filipe" com privilégios de "utilizador" (apenas pode gerir e visualizar os recursos disponibilizados pelo administrador neste projeto). Foram criados *scripts* para a autenticação com o Keystone via API, relativos aos utilizadores *admin* e *fili*pe (**Anexo C1**).

Verificação do serviço de identidade (Keystone), ao efetuar a autenticação como *admin* ou como *filipe*, e verificação de atribuição de *tokens* (Figura 17, Figura 18):

```
filipe@filipe-server:~/Documents$ source admincred.sh
filipe@filipe-server:~/Documents$ openstack project list
+----------------------------------+-------------+
| ID                               | Name        |
+----------------------------------+-------------+
| 13275bcc3cca42679b8fd86a11fd870d | script      |
| 1ba49d4fbae347fbb40b5a970c107fbd | service     |
| 5c6eea327b8242fab4ee4c0822a6454f | owncloud    |
| 8313936762594b3681f516ebbdcfb6e0 | filipe      |
| c0089b0d9e754872abf4f9fefadfd573 | admin       |
| daf7b41dd43b4d24972db1e4636f51e4 | joaodelgado |
+----------------------------------+-------------+
filipe@filipe-server:~/Documents$ source filipecred.sh
filipe@filipe-server:~/Documents$ openstack project list
ERROR: openstack You are not authorized to perform the requested action: admin_required (HTTP 403)
filipe@filipe-server:~/Documents$
```

Figura 17 - Verificação do serviço de autenticação

Fonte: Fonte própria

```
filipe@filipe-server:~/Documents$ openstack token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2015-11-02T16:10:09.766265Z      |
| id         | e8188c81bf8e4b119966248056c33133 |
| project_id | 8313936762594b3681f516ebbdcfb6e0 |
| user_id    | bb065f4a29c249358f442171cd9d6c13 |
+------------+----------------------------------+
filipe@filipe-server:~/Documents$
```

Figura 18 - Verificação da atribuição de tokens

Fonte: Fonte própria

5.4.4 Instalação e configuração do Glance

Foi criada uma base de dados para o serviço de armazenamento de imagens (Glance), da mesma forma do Keystone. Foi também criado o utilizador Glance e associado ao projeto "Service" no Keystone para autenticação do serviço. Foi criado o *endpoint* do Glance para comunicar no porto 9292 do Controlador. De seguida foram instalados os pacotes do Glance e feitas as configurações do serviço nos ficheiros *glance-api.conf* e *glance-registry.conf* (**Anexo B2**).

Verificação do serviço de armazenamento de imagens (Glance - Figura 19):

```
filipe@filipe-server:~/Documents/imagens$ glance image-create --name "Debian" --file debian-8.2.0-openstack-amd64.qcow2 \
>   --disk-format qcow2 --container-format bare --visibility public --progress
[=============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 6ee65cf230d990a05faa50038e3b6bc9     |
| container_format | bare                                 |
| created_at       | 2015-11-03T21:26:32Z                 |
| disk_format      | qcow2                                |
| id               | 6ba4dbd6-8c91-45f2-ba8d-efd2b3a6b32f |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | Debian                               |
| owner            | c0089b0d9e754872abf4f9fefadfd573     |
| protected        | False                                |
| size             | 490775040                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2015-11-03T21:26:34Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
filipe@filipe-server:~/Documents/imagens$ glance image-list
+--------------------------------------+-------------+
| ID                                   | Name        |
+--------------------------------------+-------------+
| 6ba4dbd6-8c91-45f2-ba8d-efd2b3a6b32f | Debian      |
| 53aa4af3-b972-46b5-93d9-0b3d09730fb0 | Fedora x386 |
| 8f4b68c2-714a-4483-82c3-a947dc8adbfb | cirrOS x386 |
| adb187ca-50bb-4904-86cf-1184f6633510 | ubuntu x386 |
| dcc4fb1b-5bea-42b9-864e-f2e7158cd77a | CentOS      |
| 7a134b45-0c63-4a22-8180-d746f0134793 | Fedora      |
| 9f72de0d-6ce8-4f66-9c9e-561ccbc984ca | openSUSE    |
| 31f2757c-f073-490f-b742-dda597f312bb | RHEL        |
| 8681e890-a7f7-4106-8c6c-2dd65c6a0014 | ubuntu      |
| d77a9a35-8dc7-497c-8923-3243cc33b57a | cirrOS      |
+--------------------------------------+-------------+
filipe@filipe-server:~/Documents/imagens$
```

Figura 19 – Armazenamento da imagem do SO Linux Debian

Fonte: Fonte própria

5.4.5 Instalação e configuração do Nova

Foi implementado o Nova para fazer computação das máquinas virtuais através do KVM/QEMU, que se enquadram num *hypervisor* do tipo 2 (*hosted*). A configuração da rede das máquinas virtuais foi feita através do Nova-Network devido a limitações de *hardware* e maior simplicidade de implementação, porque o ideal seria usar o Neutron[13].

O Controlador numa arquitetura bem distribuída não faz computação de máquinas virtuais, neste caso optou-se por utiliza-lo para este fim, por ser o único *host* com suporte para aceleração de virtualização por *hardware*, e por ser o *host* com maior poder computacional. O outro *host* escolhido para fazer computação foi o Nova *node* que integra o *hypervisor* QEMU sem o *fork* de KVM, que também irá executar o componente Nova-Network para aliviar algum tráfego do Controlador (balanceamento de cargas).

[13] Neutron – É um componente de rede do OpenStack semelhante ao Nova-Network mas com funcionalidades mais avançadas.

Foi então criada a base de dados, utilizador (Nova) para o serviço de computação, e criação do *endpoint* para comunicação com este serviço, alojado no porto 8774 do Controlador. De seguida foi efetuada a instalação dos respetivos pacotes e configurações (**Anexo B3**).

Após a instalação e configuração do Nova foi criada a rede das máquinas virtuais:

- *nova network-create v-net --bridge br100 --multi-host T --vlan 10 --fixed-range-v4 172.16.10.0/24*

Esta rede virtual é criada através do *bridge-utils* (Linux-Bridge, Figura 20), que utiliza a API de virtualização *libvirt*, para fazer a ligação (br100) entre a *interface* física do *host* (eth0) à *interface* virtual dos *guests* (VMs – exemplo, vnet0), permitindo assim o acesso à rede destes. É possível configurar também *link-aggregation* (LACP), VLANs para isolar o tráfego entre VMs e *spanning-tree* (STP). O Nova-network, se configurado para esta finalidade faz este processo automaticamente.

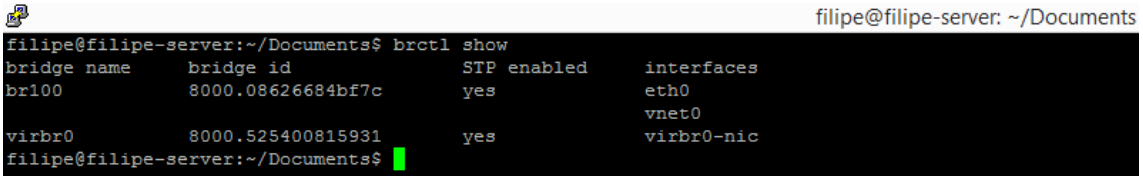


Figura 20 - Interface Bridge br100 que liga os interfaces eth0 e vnet0

Fonte: Fonte própria

Configuração da *pool*, para atribuição de endereços públicos às máquinas virtuais (Floating IP):

- *nova-manage floating create --pool=nova –ip_range=192.168.0.128/25*

Verificação do serviço de computação (Figura 21):

```
filipe@filipe-server:~/Documents$ source admincred.sh
filipe@filipe-server:~/Documents$ nova service-list
+----+------------------+--------------+------------+---------+-------+----------------------------+-----------------+
| Id | Binary           | Host         | Zone       | Status  | State | Updated_at                 | Disabled Reason |
+----+------------------+--------------+------------+---------+-------+----------------------------+-----------------+
| 1  | nova-consoleauth | filipe-server | internal   | enabled | up    | 2015-11-05T19:20:40.000000 | -               |
| 2  | nova-cert        | filipe-server | internal   | enabled | up    | 2015-11-05T19:20:40.000000 | -               |
| 3  | nova-scheduler   | filipe-server | internal   | enabled | up    | 2015-11-05T19:20:45.000000 | -               |
| 4  | nova-conductor   | filipe-server | internal   | enabled | up    | 2015-11-05T19:20:48.000000 | -               |
| 5  | nova-compute     | filipe-server | Controlador | enabled | up    | 2015-11-05T19:20:48.000000 | -               |
| 6  | nova-network     | filipe-server | internal   | enabled | up    | 2015-11-05T19:20:42.000000 | -               |
| 7  | nova-compute     | filipe-core  | Nova       | enabled | up    | 2015-11-05T19:20:48.000000 | -               |
| 8  | nova-network     | filipe-core  | internal   | enabled | up    | 2015-11-05T19:20:45.000000 | -               |
+----+------------------+--------------+------------+---------+-------+----------------------------+-----------------+
filipe@filipe-server:~/Documents$
```

Figura 21 - Verificação do estado dos componentes do Nova

Fonte: Fonte própria

Foram também criados *scripts* (Figura 22, Figura 23) para facilitar a interação com a API como criar novos utilizadores, novas máquinas virtuais, entre outros (**Anexo C**).

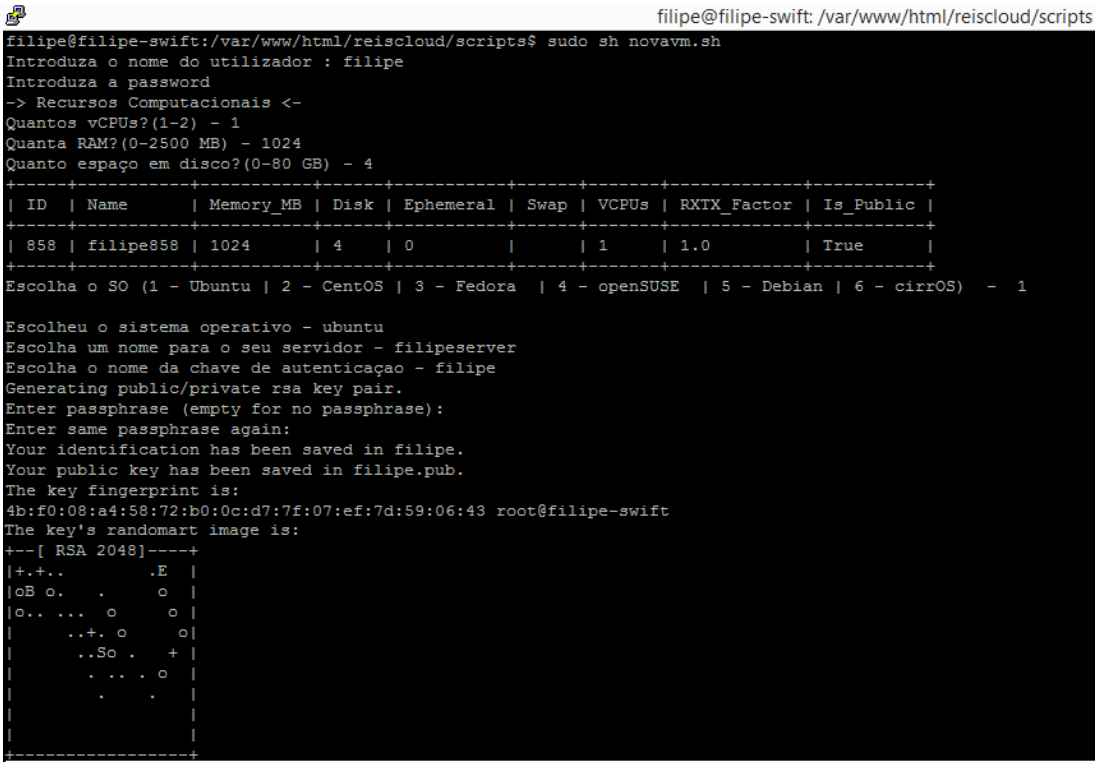


Figura 22 - Criação de uma máquina virtual via bash script

Fonte: Fonte própria

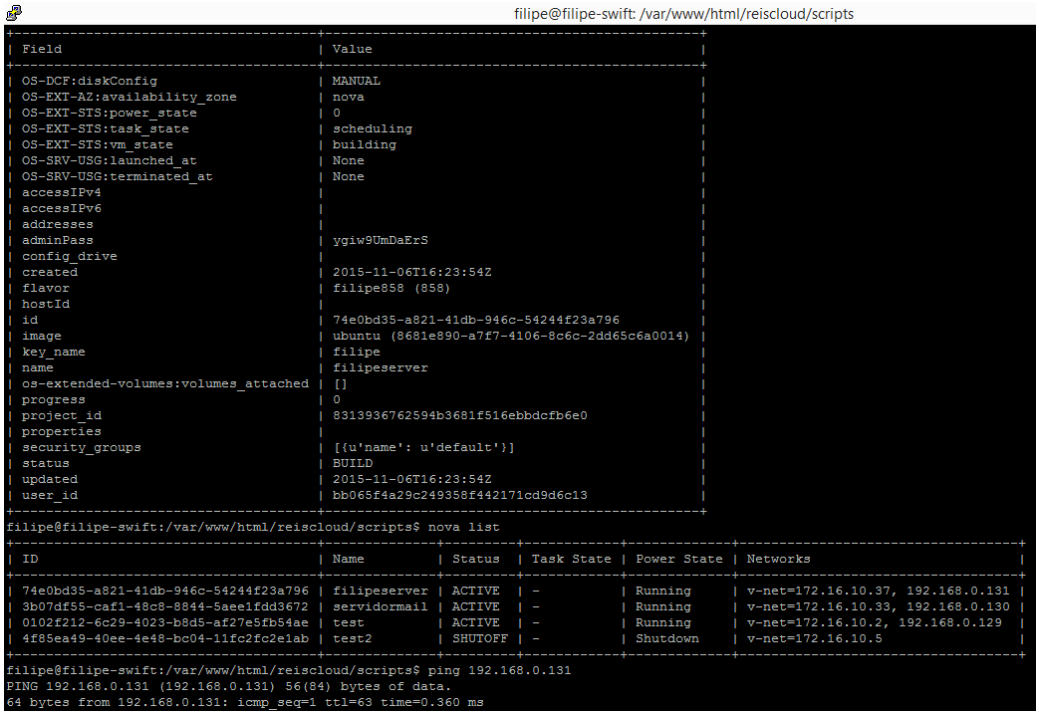


Figura 23 - Máquina virtual criada com sucesso

Fonte: Fonte própria

5.4.6 Instalação e configuração do Cinder

Foi implementado o Cinder para fazer o armazenamento de volumes de dados das máquinas virtuais. Para tal foram criadas duas partições de 25GB cada (sda5/sda6) nos *hosts* Controlador e Nova, o LVM vai ser a ferramenta responsável por aprovisionar volumes lógicos nestas partições que são fornecidos às máquinas virtuais via iSCSI[14].

Para a implementação deste serviço foi criada a base de dados, utilizador e *endpoint* para o Cinder alojado no porto 8776. De seguida foi efetuada a instalação dos respetivos pacotes e configurações (**Anexo B4**).

Verificação do serviço de armazenamento de volumes (Cinder - Figura 24, Figura 25):

```
filipe@filipe-server:~/Documents$ cinder service-list
+------------------+------------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |       Host       | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+------------------+------+---------+-------+----------------------------+-----------------+
|  cinder-backup   |    filipe-core   | nova | enabled |   up  | 2015-11-09T00:56:09.000000 |       None      |
|  cinder-backup   |   filipe-server  | nova | enabled |   up  | 2015-11-09T00:56:10.000000 |       None      |
| cinder-scheduler |   filipe-server  | nova | enabled |   up  | 2015-11-09T00:56:07.000000 |       None      |
|  cinder-volume   |  filipe-core@lvm | nova | enabled |   up  | 2015-11-09T00:56:10.000000 |       None      |
|  cinder-volume   |   filipe-server  | nova | enabled |  down | 2015-10-21T20:41:01.000000 |       None      |
|  cinder-volume   | filipe-server@lvm | nova | enabled |   up  | 2015-11-09T00:56:11.000000 |       None      |
+------------------+------------------+------+---------+-------+----------------------------+-----------------+
filipe@filipe-server:~/Documents$ source filipecred.sh
filipe@filipe-server:~/Documents$ cinder create --name volumeteste 4
+---------------------------------------+--------------------------------------+
|                Property               |                Value                 |
+---------------------------------------+--------------------------------------+
|              attachments              |                  []                  |
|           availability_zone           |                 nova                 |
|                bootable               |                false                 |
|          consistencygroup_id          |                 None                 |
|               created_at              |      2015-11-09T00:57:18.000000      |
|              description              |                 None                 |
|               encrypted               |                False                 |
|                   id                  | db4e844f-6beb-4317-8889-1cd5ec467d81 |
|               metadata                |                  {}                  |
|              multiattach              |                False                 |
|                   name                |             volumeteste              |
|      os-vol-tenant-attr:tenant_id     |   8313936762594b3681f516ebbdcfb6e0   |
|   os-volume-replication:driver_data   |                 None                 |
| os-volume-replication:extended_status |                 None                 |
|           replication_status          |               disabled               |
|                   size                |                  4                   |
|              snapshot_id              |                 None                 |
|              source_volid             |                 None                 |
|                 status                |               creating               |
|                user_id                |   bb065f4a29c249358f442171cd9d6c13   |
|              volume_type              |                 None                 |
+---------------------------------------+--------------------------------------+
```

Figura 24- Verificação do estado dos componentes do Cinder e criação de um volume

Fonte: Fonte própria

```
filipe@filipe-server:~/Documents$ cinder list
+--------------------------------------+-----------+--------------+------+-------------+----------+--------------------------------------
----------+
|                  ID                  |   Status  |     Name     | Size | Volume Type | Bootable |             Attached to
          |
+--------------------------------------+-----------+--------------+------+-------------+----------+--------------------------------------
----------+
| 7213c5cc-0a4b-4eff-b34a-daaf302e42a5 |   in-use  |   volume1    |  1   |     None    |  false   | 4f85ea49-40ee-4e48-bc04-11f
c2fc2e1ab |
| 80a2560b-8a9b-4b74-90d3-5c80a5812ae6 |   in-use  | servidormail |  4   |     lvm2    |   true   | 3b07df55-caf1-48c8-8844-5ae
e1fdd3672 |
| db4e844f-6beb-4317-8889-1cd5ec467d81 | available | volumeteste  |  4   |     None    |  false   |
          |
+--------------------------------------+-----------+--------------+------+-------------+----------+--------------------------------------
----------+
filipe@filipe-server:~/Documents$
```

Figura 25 - Volume criado com sucesso

Fonte: Fonte própria

[14] iSCSI – Permite a ligação e transferência de dados entre computadores e periféricos através de redes IP.

5.4.7 Instalação e configuração do Swift

Foi implementado o Swift para o armazenamento de ficheiros/objetos. O Swift tem como requisito, que o disco rígido onde é efetuado o armazenamento apenas tenha uma partição a conter todo o espaço desse disco. Para esse efeito foram utilizados dois discos externos de 1 TB no Nova e Swift *node*, e um disco de 40GB no Gestor *node.* O Controlador irá executar o Swift-Proxy, que é o responsável pela gestão de pedidos relativos ao armazenamento nestes *nodes*. Para a implementação deste serviço foi criado o utilizador e *endpoint* para o Swift alojado no porto 8080. De seguida foi efetuada a instalação dos respetivos pacotes e configurações (**Anexo B5**).

Nos *nodes* destinados ao armazenamento dos dados, foram instalados os respetivos pacotes e feitas as configurações para este serviço (**Anexo B5**), da mesma forma para todos, alterando apenas o endereço IP de cada um destes *nodes*.

O sistema de ficheiros escolhido para os discos destinados ao armazenamento, foi o XFS que é o recomendado pelo OpenStack, apesar de qualquer sistema de ficheiros com *extended attributes* também ser suportado. Este processo foi feito da seguinte forma:

- *mkfs.xfs /dev/sdb1*
- *mkdir –p /srv/node/sdb1*
- *mount /srv/node/sdb1*

Foi adicionada a seguinte linha ao ficheiro /etc/fstab de forma a que o sistema faça *mount* do disco no diretório escolhido sempre que seja inicializado:

- */dev/sdb1 /srv/node/sdb1 xfs noatime, nodiratime, nobarrier, logbuffs=8 0 2*

De seguida procedeu-se à criação dos *rings* do Swift, que resumidamente, fazem o mapeamento entre os componentes dos *nodes* de armazenamento e o disco rígido. Para a criação destes *rings*, existem três parâmetros, sendo o primeiro o número de partições (*hashing* e não partições de disco*)*, quantas mais partições melhor será a distribuição, mas também irá colocar maior carga nesse *host*. Uma boa regra é ter cerca de 100 partições ou mais por disco, logo o valor escolhido para o primeiro parâmetro foi 7 (2^7=128). O segundo parâmetro define o número de réplicas, o OpenStack recomenda três, mas nesta solução não é funcional porque só existem três *nodes* de armazenamento.

Visto que, se um falha segundo a arquitetura deste serviço não será possível fazer o *upload* de um ficheiro, porque este obriga a que pelo menos três *hosts* recebam esse ficheiro. Logo escolheram-se duas réplicas, de forma que, mesmo que um *node* falhe, seja possível enviar os dados para os outros dois. O terceiro parâmetro é o tempo em horas antes de uma partição específica poder ser movida em sucessão. Logo a criação dos *rings* foi elaborada da seguinte forma:

- *swift-ring-builder account.builder create 7 2 1*
- *swift-ring-builder container.builder create 7 2 1*
- *swift-ring-builder object.builder create 7 2 1*

Adição dos *nodes* de armazenamento aos *rings* (o último parâmetro significa o peso deste *node):*

- *swift-ring-builder account.builder add r1z1-192.168.0.40:6002/sdb1 100*
- *swift-ring-builder account.builder add r1z1-192.168.0.42:6002/sdb1 100*
- *swift-ring-builder account.builder add r1z1-192.168.0.43:6002/sdb1 100*
- *swift-ring-builder account.builder rebalance*
- *swift-ring-builder container.builder add r1z1-192.168.0.40:6001/sdb1 100*
- *swift-ring-builder container.builder add r1z1-192.168.0.42:6001/sdb1 100*
- *swift-ring-builder container.builder add r1z1-192.168.0.43:6001/sdb1 100*
- *swift-ring-builder container.builder rebalance*
- *swift-ring-builder object.builder add r1z1-192.168.0.40:6000/sdb1 100*
- *swift-ring-builder object.builder add r1z1-192.168.0.42:6000/sdb1 100*
- *swift-ring-builder object.builder add r1z1-192.168.0.43:6000/sdb1 100*
- *swift-ring-builder object.builder rebalance*

Como foram definidas duas réplicas, o espaço total para armazenamento de dados será de 1,20TB ((1 + 1 + 0,40) / 2). Foi verificada a criação dos *rings* (Figura 26).

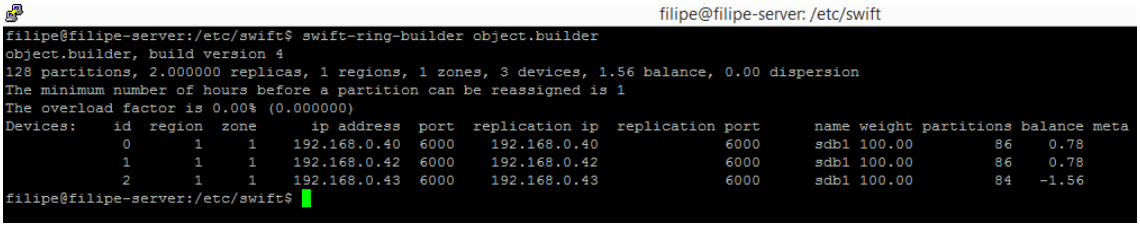


Figura 26 - Ring de objetos já criado com os respetivos nodes de armazenamento

Fonte: Fonte própria

Verificação do serviço de armazenamento de objetos (Swift - Figura 27):

```
filipe@filipe-server: ~
filipe@filipe-server:~/Music$ swift -V 3 stat
                     Account: AUTH_8313936762594b3681f516ebbdcfb6e0
                  Containers: 2
                     Objects: 209
                       Bytes: 404362209
X-Account-Project-Domain-Id: default
                  Connection: keep-alive
                 X-Timestamp: 1440265310.01021
                  X-Trans-Id: tx463b141951134a2eb6ed7-0056401332
                Content-Type: text/plain; charset=utf-8
               Accept-Ranges: bytes
filipe@filipe-server:~/Music$ swift -V 3 upload teste newstuff.txt
newstuff.txt
filipe@filipe-server:~/Music$ cd ..
filipe@filipe-server:~$ swift -V 3 download teste newstuff.txt
newstuff.txt [auth 0.126s, headers 0.166s, total 0.167s, 0.001 MB/s]
```

Figura 27 - Verificação do funcionamento do Swift através do upload e download de um ficheiro via API

Fonte: Fonte própria

5.4.8 Instalação e configuração do Horizon

Foi implementado o Horizon, que permite a gestão centralizada dos serviços e recursos computacionais aos utilizadores e administradores via *interface web*. Esta *interface* funciona à base de interações *web* com o Controlador através da API OpenStack. Assim é possível criar uma *interface* diferente para os utilizadores, como um exemplo simples elaborado em PHP, que interage com um *script* em *bash* para a criação de novos utilizadores (**Anexo C5**). Foram então instalados os pacotes do Horizon e feitas as respetivas configurações (**Anexo B6**).

Verificação da *Dashboard* (Horizon - Figura 28) e funcionamento desta (**Anexo G**):

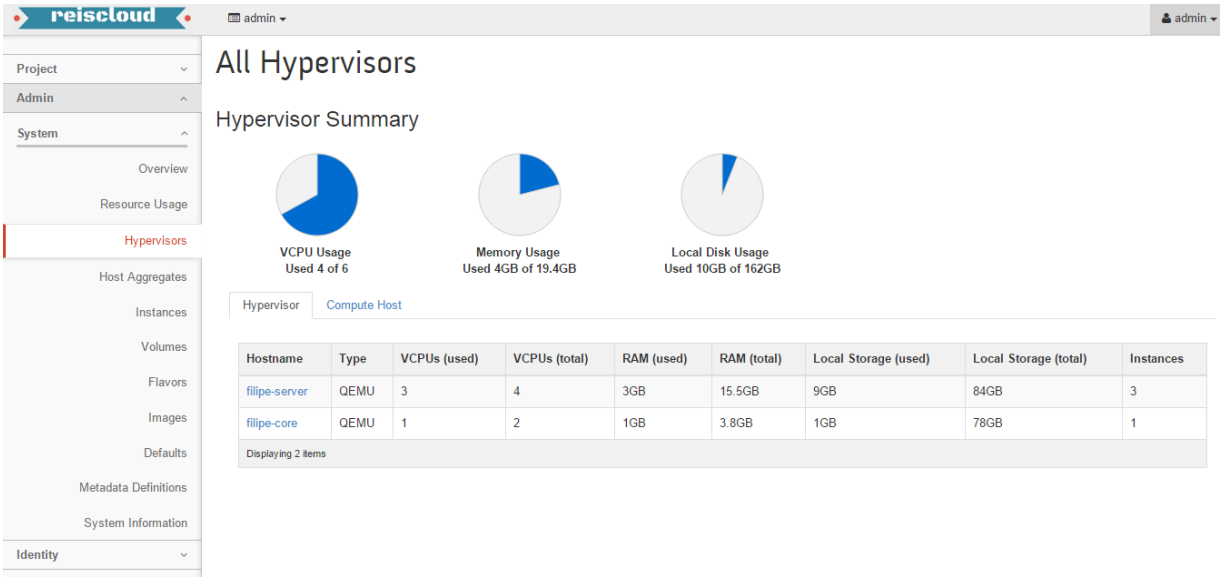


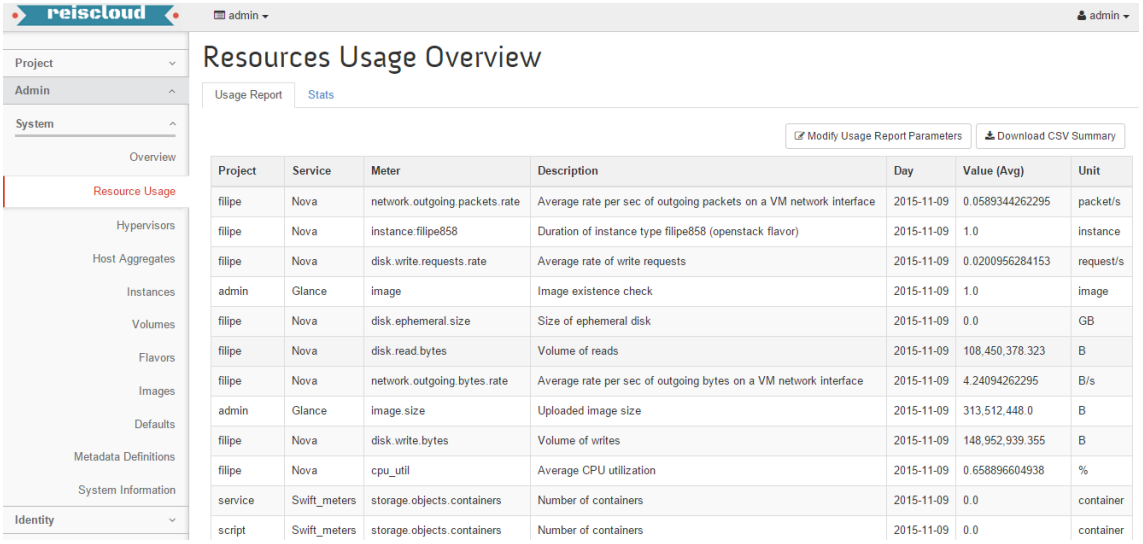
Figura 28 – Verificação do funcionamento da interface web do OpenStack

Fonte: Fonte própria

5.4.9 Instalação e configuração do Ceilometer

Foi implementado o Ceilometer, para a recolha de métricas e criação de alarmes relativos aos serviços presentes no ambiente Cloud. É possível visualizar estas métricas e respetivas estatísticas via API ou através da *Dashboard* do OpenStack. Para a implementação deste serviço foi criada a base de dados no MongoDB, utilizador e *endpoint* alojado no porto 8777. Foram instalados os pacotes do serviço Ceilometer e respetivas configurações (**Anexo B7**).

Verificação do serviço de métricas e alarmes (Ceilometer - Figura 29):



Project	Service	Meter	Description	Day	Value (Avg)	Unit
filipe	Nova	network.outgoing.packets.rate	Average rate per sec of outgoing packets on a VM network interface	2015-11-09	0.0589344262295	packet/s
filipe	Nova	instance:filipe858	Duration of instance type filipe858 (openstack flavor)	2015-11-09	1.0	instance
filipe	Nova	disk.write.requests.rate	Average rate of write requests	2015-11-09	0.0200956284153	request/s
admin	Glance	image	Image existence check	2015-11-09	1.0	image
filipe	Nova	disk.ephemeral.size	Size of ephemeral disk	2015-11-09	0.0	GB
filipe	Nova	disk.read.bytes	Volume of reads	2015-11-09	108,450,378.323	B
filipe	Nova	network.outgoing.bytes.rate	Average rate per sec of outgoing bytes on a VM network interface	2015-11-09	4.24094262295	B/s
admin	Glance	image.size	Uploaded image size	2015-11-09	313,512,448.0	B
filipe	Nova	disk.write.bytes	Volume of writes	2015-11-09	148,952,939.355	B
filipe	Nova	cpu_util	Average CPU utilization	2015-11-09	0.658896604938	%
service	Swift_meters	storage.objects.containers	Number of containers	2015-11-09	0.0	container
script	Swift_meters	storage.objects.containers	Number of containers	2015-11-09	0.0	container

Figura 29 - Métricas recolhidas pelo Ceilometer

Fonte: Fonte própria

5.4.10 Instalação e configuração do Heat

O Heat permite "orquestrar" os recursos e aplicações presentes na Cloud de forma mais eficiente. Foi implementado mais tarde na solução, por curiosidade e para efetuar testes às capacidades deste componente. Permite funcionalidades avançadas como escalabilidade automática de recursos ou serviços, consoante o estipulado pelo utilizador nos *templates*. Estes *templates*, viabilizam a criação de *stacks*, onde a partir destes podemos criar ou gerir por exemplo, um elevado número de máquinas virtuais ou volumes de dados, de forma massiva, e realizar operações em todas estas simultaneamente a partir de um *template* de ambiente. Para a implementação deste serviço foi criada a base de dados, utilizador e *endpoints* à escuta nos portos 8004 e 8000. Foram instalados os pacotes do serviço Heat e respetivas configurações (**Anexo B8**).

Para verificar o Heat foi criada uma *stack* simples de apenas duas máquinas virtuais, onde é possível "orquestrar" ambas simultaneamente.

Verificação do Heat (Figura 30):

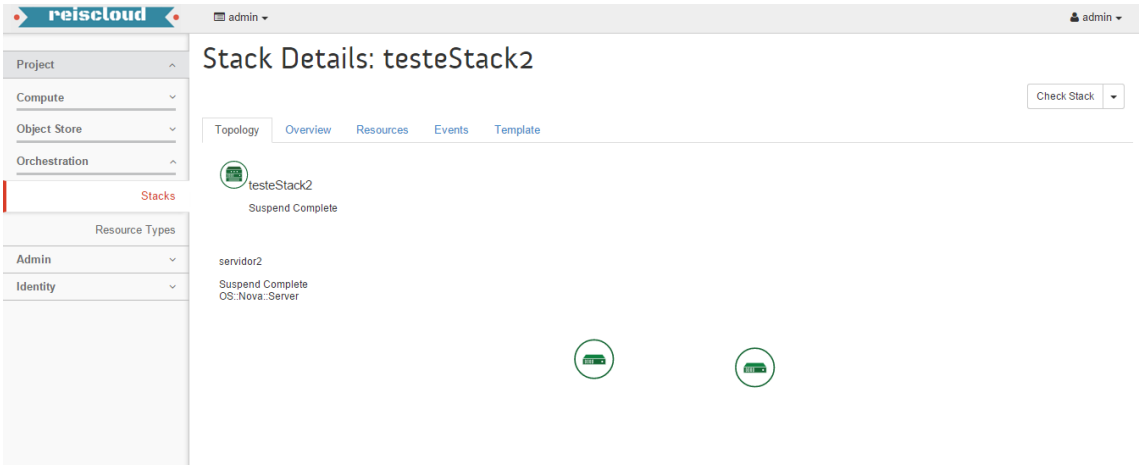


Figura 30 - Stack de duas máquinas virtuais criados pelo Heat

Fonte: Fonte própria

5.5 Implementação do ownCloud

Após a implementação do OpenStack e verificação do funcionamento dos componentes instalados, optou-se por implementar o ownCloud, para servir de *interface web* ao armazenamento de ficheiros no ambiente Cloud. A implementação foi feita seguindo a documentação do ownCloud no Swift *node* (**Anexo D**), e após esta ter sido concluída apenas foi necessário ativar a aplicação "*External Storage*" para fazer a ligação ao OpenStack Swift (Figura 31) [54].

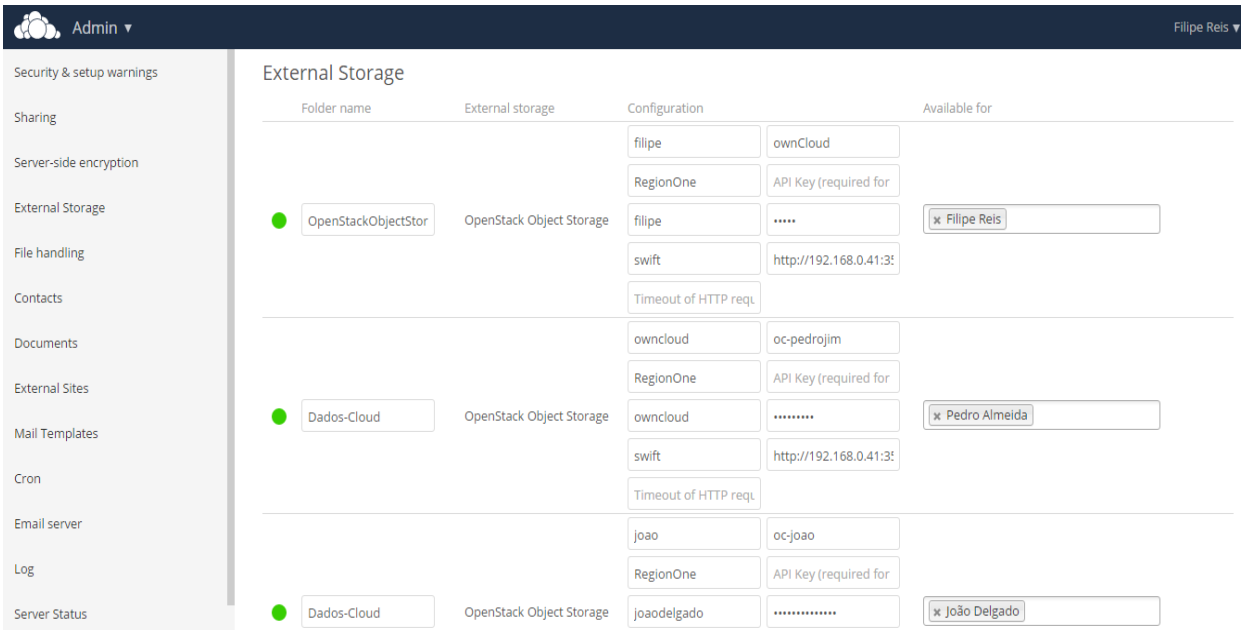


Figura 31 - Adição do OpenStack Swift como backend do ownCloud

Fonte: Fonte própria

Também foi utilizado o ownCloud para fazer a encriptação dos ficheiros enviados para a Cloud (Figura 32), ativando também a aplicação de "*Server Side Encryption"*, e criação de uma chave de recuperação de dados.

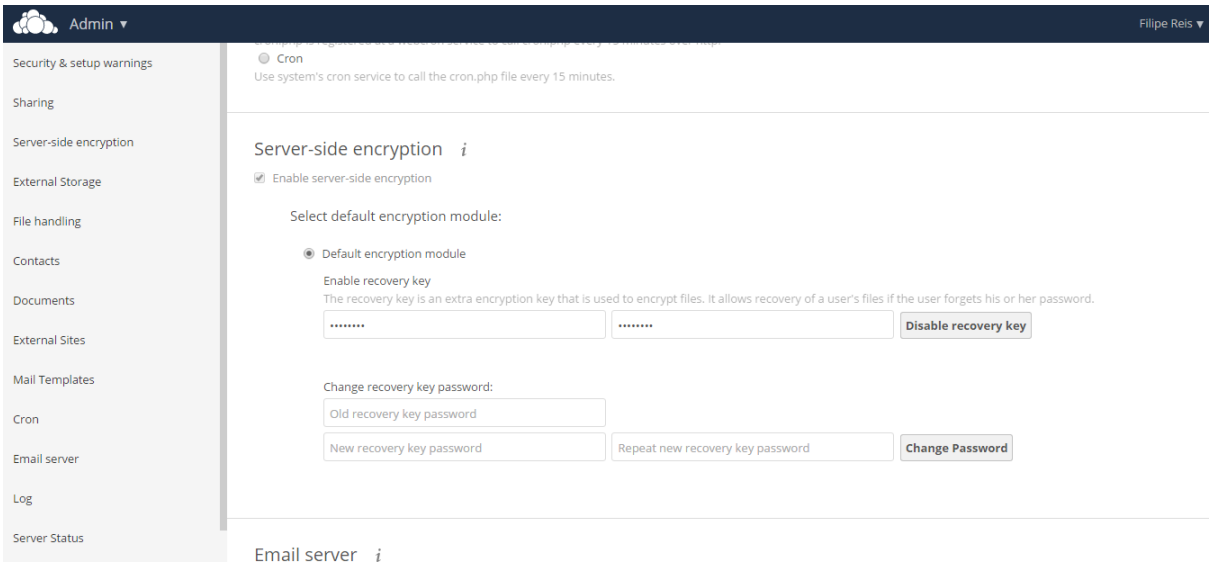


Figura 32 - Ativação do serviço de encriptação dos ficheiros

Fonte: Fonte própria

O funcionamento do ownCloud (Figura 33) a nível de utilizador é muito simples e intuitivo (semelhante à Dropbox), com funcionalidades de partilha de ficheiros, também é possível instalar várias aplicações desenvolvidas pela comunidade, como, listas de contactos, galerias de imagens, calendários para criação de eventos, escrita de documentos em simultâneo com outros utilizadores, entre outros.

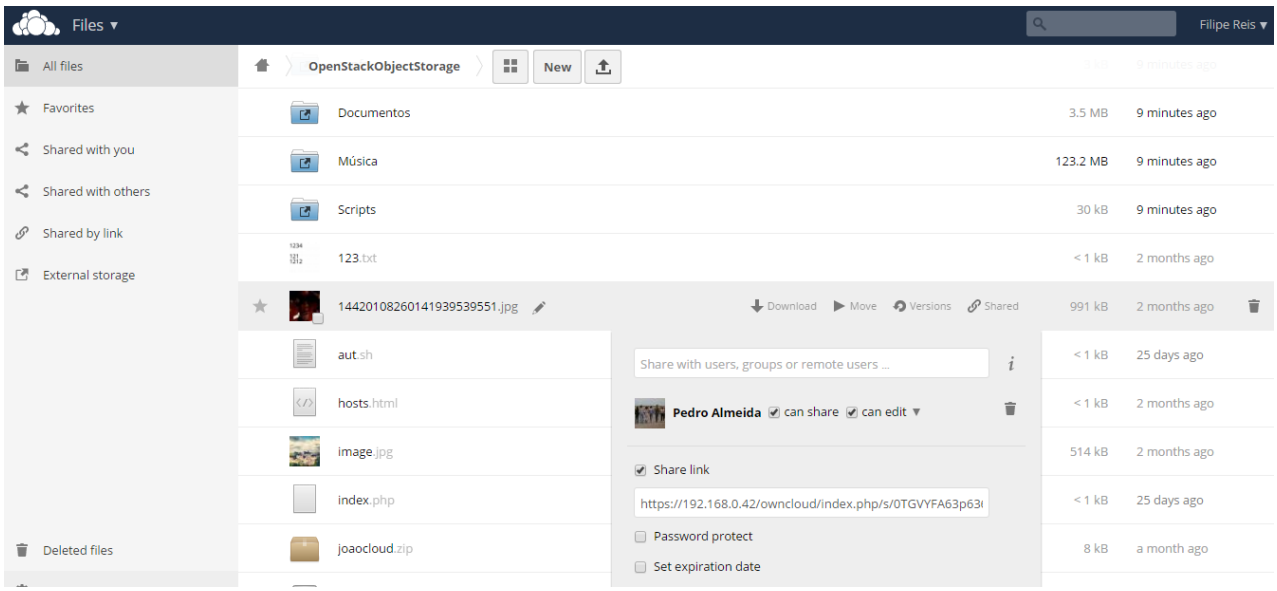


Figura 33 – Interface do ownCloud

Fonte: Fonte própria

5.6 Implementação de ferramentas de monitorização e gestão

Para facilitar a monitorização e gestão do ambiente Cloud, foram implementadas as ferramentas Nagios, Cacti no Gestor *node* e Ajenti em todos os *nodes* para gestão gráfica destes (**Anexo E**). Foi instalado o SNMP e SNMPD para a criação da *community*, e recolha de dados relativos aos recursos computacionais a serem utilizados pelos *hosts*. A implementação destas ferramentas foi feita seguindo a documentação disponibilizada pela comunidade [55][56][57].

De seguida são apresentadas algumas imagens destas ferramentas já em funcionamento (Figura 34, Figura 35, Figura 36):

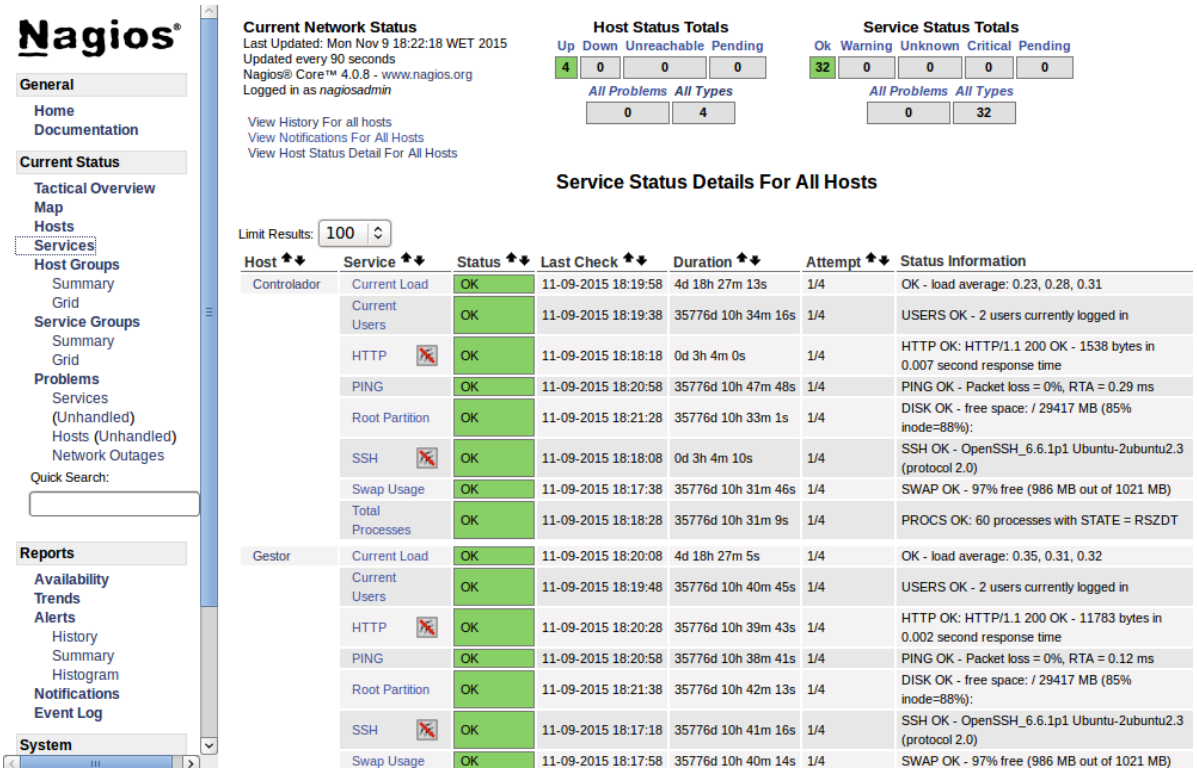


Figura 34 - Interface do Nagios

Fonte: Fonte própria

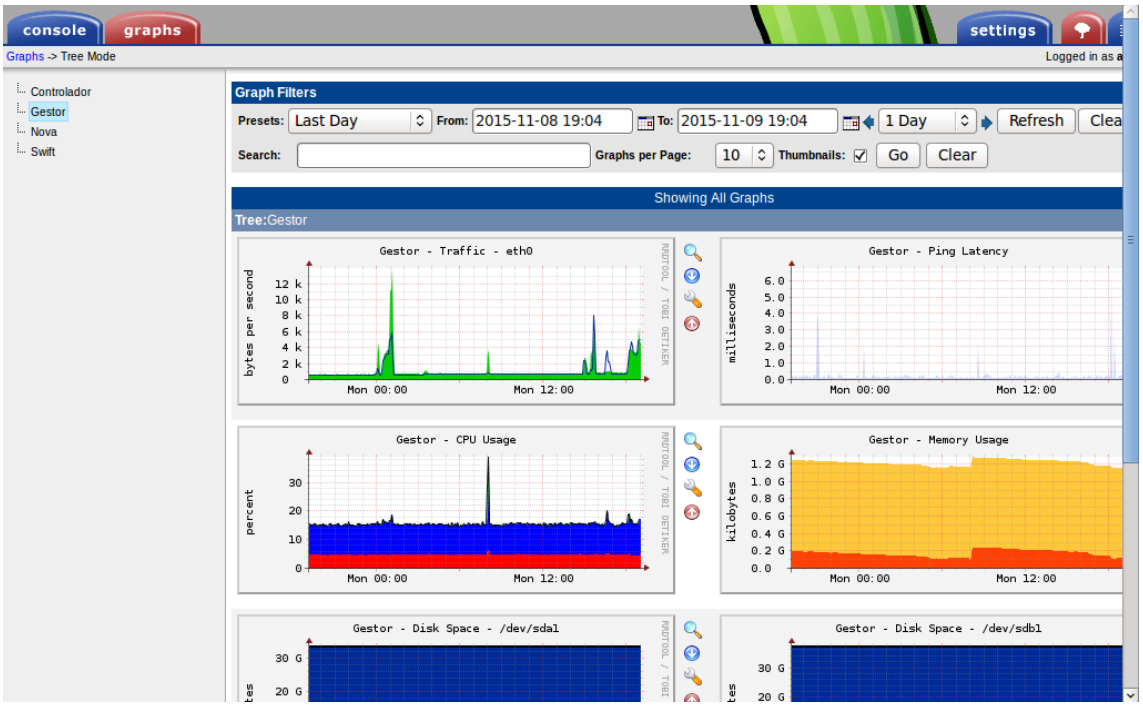


Figura 35 - Interface do Cacti

Fonte: Fonte própria

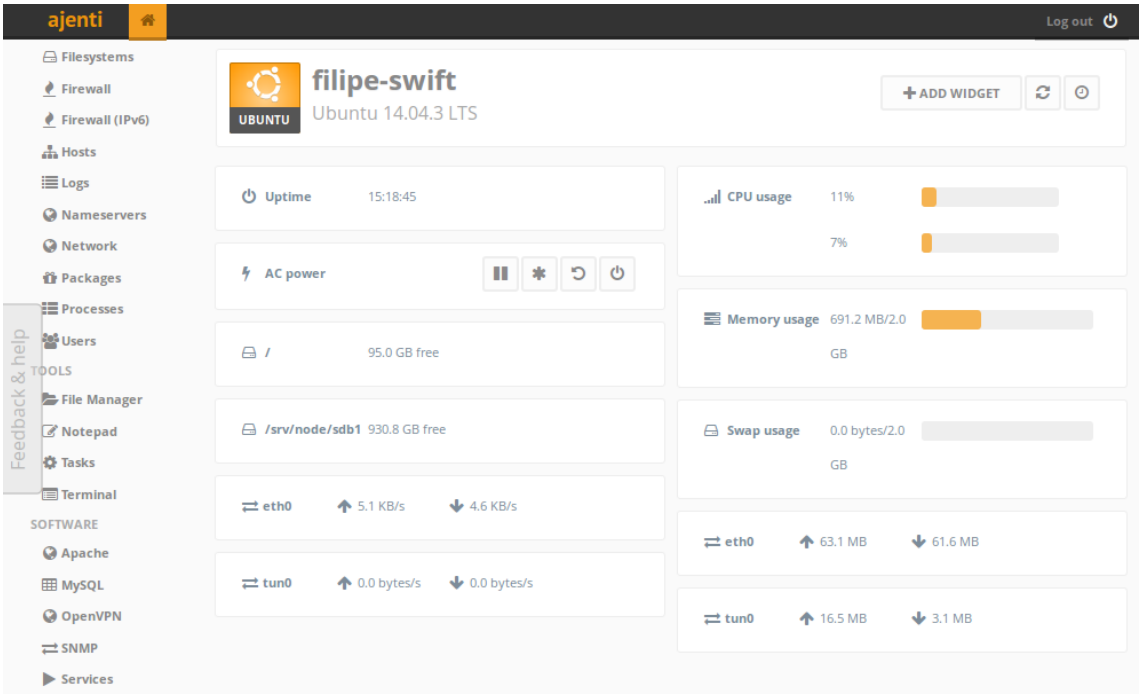


Figura 36 - Interface do Ajenti

Fonte: Fonte própria

Foi também criada uma página *web* para permitir um acesso simples às várias tecnologias implementadas no ambiente Cloud (Figura 37).

Figura 37 – Página web para acesso às interfaces dos serviços da Cloud

Fonte: Fonte própria

5.7 *Logging* e *Troubleshooting*

A leitura de *logs* foi de grande importância durante a implementação de toda a solução, grande parte dos problemas que surgiram foram resolvidos através da leitura destes. Em alguns casos também foi ativada a opção "*debug=true"* para receber mais informação acerca de certos serviços. Os *logs* têm níveis de severidade, que vai de menos para mais, da seguinte forma, *debug, info, audit, warning, error, critical, trace*.

Cada serviço tem o seu respetivo *log*, logo a quantidade de *logs* no ambiente torna-se bastante extensa, para facilitar a gestão e leitura destes, foi decidido implementar o projeto *ELK Stack* (**Anexo F**) [38][58].

Estas ferramentas permitem o armazenamento dos *logs* num servidor, dedicado a este fim. Foi criada uma máquina virtual para receber e armazenar os *logs* de vários serviços da Cloud. O ELK *Stack* permite armazenar, filtrar e pesquisar informação específica de *logs*, possibilitando também, gerir e visualizar todos estes através de uma *interface web* (Kibana).

A Figura 38 apresenta a interface do Kibana após a implementação do ELK Stack.

Figura 38 - Interface do Kibana para gestão centralizada dos logs

Fonte: Fonte própria

5.8 Implementação da VPN

Foi implementada uma VPN através do *software* OpenVPN, para permitir um acesso seguro à Cloud a partir do exterior. O servidor OpenVPN foi hospedado no Swift *node* seguindo a documentação fornecida pelo OpenVPN e comunidade [59]. Foi criado um *hostname* (reiscloud.ddns.net) através de *Dynamic DNS* (**Anexo A2**), no *Router-Interno*, para mapear o endereço IP público da Cloud, visto que este endereço fornecido pelo ISP, é dinâmico. Após a configuração do servidor VPN (**Anexo A3**), foi criado um *script* que é executado quando é inicializado o sistema operativo, para criar regras na *firewall*, que permitem que o tráfego do *tunnel*, aceda à LAN e à Internet (**Anexo C2**). Esta configuração também podia ter sido efetuada através do serviço IPTables-Persistent, mas optou-se por este método visto que este proporciona maior flexibilidade de alterar as regras da *firewall* rapidamente caso seja necessário.

Verificação da VPN (Figura 39):

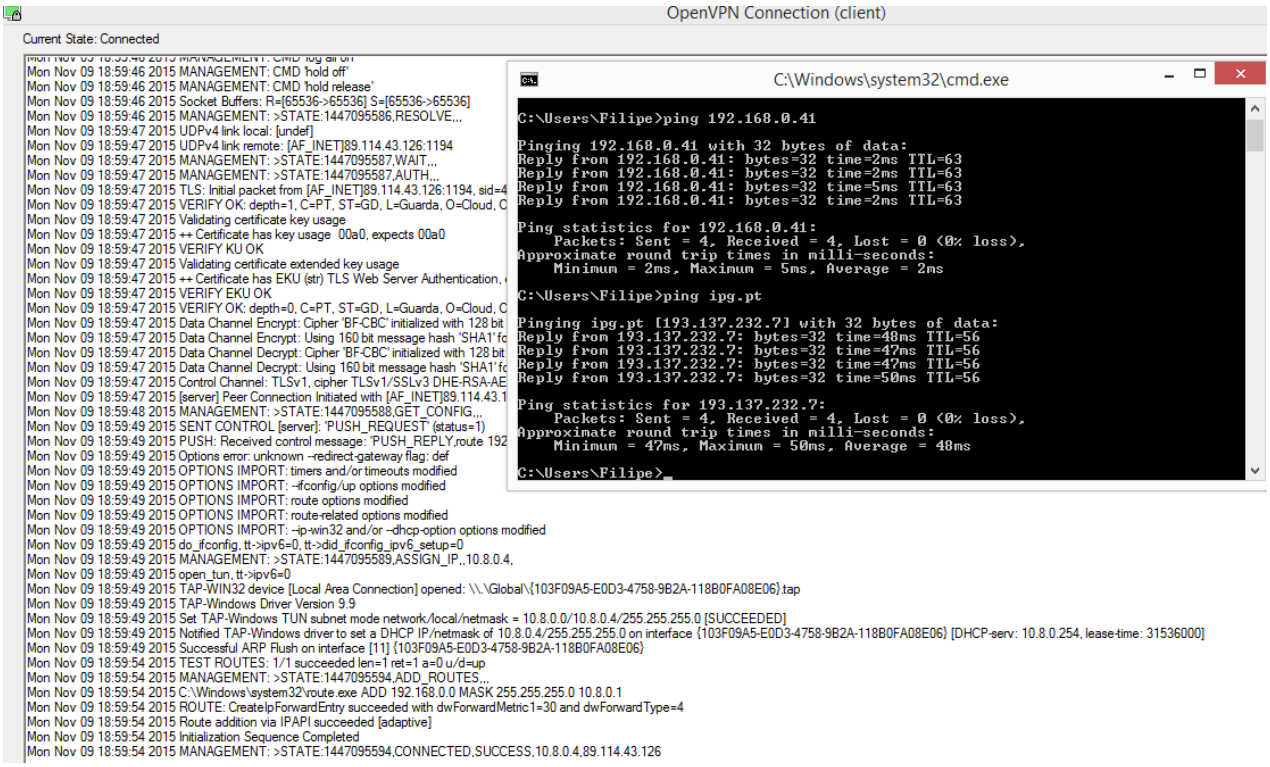


Figura 39 - Ligação à VPN e testes de ligação para dentro e fora da rede

Fonte: Fonte própria

5.9 Testes e resultados obtidos

Após a solução ter sido implementada, procedeu-se à realização de vários testes, para verificar se a solução obtida, tinha alcançado os requisitos inicialmente definidos.

Com a conclusão dos testes realizados à solução, foram obtidos os seguintes resultados:

- A gestão do ambiente Cloud é simples e centralizada;
- É possível criar máquinas virtuais de forma simples e unilateralmente;
- Aumento e redução dos recursos das máquinas virtuais, consoante a necessidade do utilizador de forma simples e eficiente;
- Armazenamento e partilha de ficheiros de forma simples e intuitiva (ownCloud);
- Os dados armazenados na Cloud garantem redundância e alta disponibilidade;
- É possível aceder ao ambiente Cloud de forma segura e privada a partir do exterior desta;
- É possível criar métricas e alarmes para os vários serviços presentes na Cloud;
- É possível migrar máquinas virtuais de *host*, apesar da *performance* no Nova *node* ser bastante inferior ao Controlador;
- Fácil criação de *backups* das máquinas virtuais e volumes a partir da criação de *snapshots* ou pelo componente Cinder-backup;
- Maior facilidade em monitorizar e gerir o ambiente (Nagios, Cacti, Ajenti, ELK).

6. Conclusões

Durante o desenvolvimento deste projeto, foram adquiridos conhecimentos sobre temas não abordados ao longo da licenciatura, tais como, o conceito de Cloud Computing e aderentes tecnologias, como virtualização e computação distribuída. Outras aptidões adquiridas durante a licenciatura e certificados Cisco (CCNA, LPI-0 e LPI-1), foram de grande importância para atingir os objetivos previstos no início do projeto.

Durante a implementação da solução surgiram vários inconvenientes, como o insucesso na instalação e configuração do componente de rede, Neutron, por limitações de *hardware* (configuração um pouco rígida que pede pelo menos duas placas de rede). Que se tentou contornar sem sucesso com a criação de *interfaces* virtuais, devido à falta de experiência/conhecimentos e pouca documentação existente nesta matéria. Este obstáculo foi superado ao escolher o Nova Network, como módulo de rede das máquinas virtuais, devido à sua maior simplicidade de funcionalidades e arquitetura. Outros inconvenientes como o não/mau funcionamento de serviços (bugs/erros) após a implementação destes, foram superados através da leitura dos respetivos *logs*, e posterior *troubleshooting*. Leitura da documentação sobre *troubleshooting* fornecida pelo OpenStack, e ajuda fornecida pela comunidade.

Pode-se concluir que as metas definidas inicialmente foram alcançadas com êxito, o desenvolvimento deste projeto permitiu adquirir conhecimentos sobre Cloud Computing, uma área que se encontra em crescimento, já com bastante impacto no mundo empresarial, em grande parte devido à maior facilidade de gestão e redução de custos que este conceito proporciona.

6.1 Trabalho Futuro

No geral, os objetivos inicialmente estabelecidos foram cumpridos, mas durante a implementação da solução existiram certos complementos "extra" adicionados, de forma a tornar o projeto mais completo e a facilitar a gestão do ambiente Cloud. No entanto por limitações de tempo e *hardware*, há certas funcionalidades/serviços que não foram melhorados. O trabalho futuro a realizar, de modo a melhorar a solução, seria o seguinte:

- Implementação de pelo menos mais um Controlador para redundância, caso este falhe;
- Com mais um Controlador, deverá ser utilizado o Swift como *backend* do Glance;
- Implementação de mais um *node* apenas para o armazenamento de volumes, e um *node* apenas para computação, também para garantir maior redundância das máquinas virtuais;
- Implementação do Ceph para armazenamento dos volumes, de forma a garantir maior redundância destes;
- Implementação de um servidor de e-mail, para enviar alarmes do Nagios e melhorar as configurações deste;
- Implementação de um *node* exclusivamente para a rede das máquinas virtuais e eventualmente trocar o Nova-Network pelo Neutron;
- Testar melhor o Heat e criar *templates* com utilidades mais "reais";
- Adição de alarmes de vários serviços e métricas mais específicas aos recursos a serem consumidos;
- Melhorar o serviço de gestão de *logs*, ao aproveitar melhor as funcionalidades proporcionadas pelo ELK Stack.

Bibliografia

[1] Ferreira, António Miguel. *"Introdução ao Cloud Computing"*. : FCA, Março, 2015.

[2] Mell, Peter e Grance, Timothy. "The NIST Definition of Cloud Computing". *NIST*. [Online] Setembro de 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [Acedido: Agosto-2015].

[3] "The Pros and Cons of Public and Private Clouds". *Infusionsoft.* [Online] http://www.infusionsoft.com/blog/pros-and-cons-public-and-private-clouds [Acedido: Setembro-2015].

[4] Savvas, Antony. "The benefits of private Cloud Computing". *ITProPortal.* [Online] Maio de 2014. http://www.itproportal.com/2014/05/12/the-benefits-of-private-cloud-computing/ [Acedido: Setembro-2015].

[5] "The benefits of public Cloud Computing". *ITProPortal.* [Online] Maio de 2014. http://www.itproportal.com/2014/05/07/benefits-public-cloud-computing/ [Acedido: Setembro-2015].

[6] Santos, Sandra Sergi. "Advantages and options of private cloud computing". *IBM.* [Online] Abril de 2012. http://www.ibm.com/developerworks/rational/library/private-cloud-advantages-options/ [Acedido: Setembro-2015].

[7] Tsagklis, Ilyas. "Cloud computing pros and cons". *Javacodegeeks.* [Online] Abril de 2013. http://www.javacodegeeks.com/2013/04/advantages-and-disadvantages-of-cloud-computing-cloud-computing-pros-and-cons.html [Acedido: Setembro-2015].

[8] Angeles, Sara. "Virtualization vs Cloud Computing". *BusinessNewsDaily.* [Online] Janeiro de 2014. http://www.businessnewsdaily.com/5791-virtualization-vs-cloud-computing.html [Acedido: Setembro-2015].

[9] Lima, Lucas. [Online] https://confluence.atlassian.com/display/BAMKB/Obtaining+AWS+Key+Pair+to+access+Amazon+Elastic+Instances [Acedido: Setembro-2015].

[10] Peter. "Rackspace cloudservers". [Online] http://blog.sflow.com/2011/01/rackspace-cloudservers.html [Acedido: Setembro-2015].

[11] Meyer, David. "Launchpad finalist Lunacloud offers strong IaaS localization in Europe and beyond". *Gigaom.* [Online] Agosto de 2013. https://gigaom.com/2013/08/21/launchpad-finalist-lunacloud-offers-strong-iaas-localization-in-europe-and-beyond/ [Acedido: Setembro-2015].

[12] *OpenStack.* [Online] https://www.openstack.org/software/ [Acedido: Junho-2015].

[13] "OpenStack". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/OpenStack [Acedido: Junho-2015].

[14] "Cloud Computing". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Cloud_computing [Acedido:Maio-2015].

[15] "A Game of Stacks: OpenStack vs CloudStack". *GetFileCloud.* [Online] Fevereiro de 2014. https://www.getfilecloud.com/blog/2014/02/a-game-of-stacks-openstack-vs-cloudstack/ [Acedido: Junho-2015].

[16] Kleyman, Bill. "Understanding ClouStack, OpenStack, and the Cloud API". *DatacenterKnowledge.* [Online] Fevereiro de 2015. http://www.datacenterknowledge.com/archives/2015/02/23/openstack-vs-cloudstack-the-platforms-and-the-cloud-apis/ [Acedido: Maio-2015].

[17] "Cisco Systems". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Cisco_Systems [Acedido: Outubro 2015].

[18] "OpenStack at Cisco". *Cisco.* [Online] http://www.cisco.com/c/en/us/solutions/data-center-virtualization/openstack-at-cisco/index.html [Acedido: Outubro-2015].

[19] Gaudreault, François. "CloudStack vs OpenStack". *Cloudops.* [Online] Fevereiro de 2013. http://www.cloudops.com/2013/02/cloudstack-vs-openstack-a-personal-experience/ [Acedido: Junho 2015].

[20] "Apache CloudStack". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Apache_CloudStack [Acedido: Junho-2015].

[21] "About CloudStack". *CloudStack.* [Online] https://cloudstack.apache.org/ [Acedido Junho-2015].

[22] "CloudStack CentOS Template". *Shankerbalan.* [Online] http://shankerbalan.net/blog/cloudstack-centos-template-install-from-iso/ [Acedido: Junho-2015].

[23] "Amazon Web Services". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Amazon_Web_Services [Acedido: Junho-2015].

[24] "Cloud Computing with Amazon Web Services". *Amazon Web Services.* [Online] https://aws.amazon.com/what-is-aws/ [Acedido: Junho-2015].

[25] "Rackspace vs Amazon EC2". *SoftwareInsider.* [Online] http://cloud-computing.softwareinsider.com/saved_compare/Rackspace-vs-Amazon-EC2 [Acedido: Junho-2015].

[26] "Rackspace vs AWS". *Quora.* [Online] https://www.quora.com/Rackspace-Cloud-vs-Amazon-Cloud-%E2%80%94-which-one-is-better-And-why [Acedido: Junho-2015].
[27] "Rackspace". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Rackspace [Acedido: Junho-2015].
[28] "About". *Rackspace.* [Online] http://www.rackspace.com/pt/about [Acedido: Junho-2015].
[29] Lunacloud. *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Lunacloud [Acedido: Junho-2015].
[30] *Lunacloud.* [Online] http://www.lunacloud.com/pt/ [Acedido: Junho-2015].
[31] "Lunacloud, Amazon EC2, and Rackspace Cloud Compared". *CloudSpectator.* [Online] Dezembro de 2012. http://cloudspectator.com/lunacloud-amazon-ec2-and-rackspace-cloud-compared/ [Acedido: Junho-2015].
[32] Cisco Metapod. *Cisco.* [Online] http://www.cisco.com/c/en/us/products/cloud-systems-management/metapod/index.html [Acedido: Agosto-2015].
[33] Zhang, Victor. "CloudStack vs OpenStack". *Slideshare.* [Online] Março de 2015. http://pt.slideshare.net/opendecoy/cloud-stack-vs-openstack1 [Acedido: Junho-2015].
[34] "ownCloud features". *ownCloud.* [Online] https://owncloud.org/features/ [Acedido: Junho-2015].
[35] "ownCloud". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/OwnCloud [Acedido: Junho-2015].
[36] "OpenStack Docs". *OpenStack.* [Online] http://docs.openstack.org/kilo/ [Acedido: Julho-2015].
[37] "OpenStack Architecture Design Guide". *OpenStack.* [Online] http://docs.openstack.org/arch-design/content/ [Acedido: Julho-2015].
[38] "OpenStack Operations Guide". *OpenStack.* [Online] http://docs.openstack.org/openstack-ops/content/ [Acedido: Julho-2015].
[39] "OpenStack Cloud Administrator Guide". *OpenStack.* [Online] http://docs.openstack.org/admin-guide-cloud/ [Acedido: Agosto-2015].
[40] "OpenStack CLI Reference". *OpenStack.* [Online] http://docs.openstack.org/cli-reference/content/ [Acedido: Agosto-2015].
[41] "OpenStack Kilo Installation Guide". *OpenStack.* [Online] http://docs.openstack.org/kilo/install-guide/install/apt/content/ [Acedido: Julho-2015].

[42] "RabbitMQ". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/RabbitMQ [Acedido: Agosto-2015].

[43] "Keystone Architecture". *OpenStack.* [Online] http://docs.openstack.org/developer/keystone/architecture.html [Acedido: Agosto-2015].

[44] "Glance Architecture". *OpenStack.* [Online] http://docs.openstack.org/developer/glance/architecture.html [Acedido: Agosto-2015].

[45] "Nova System Architecture". *OpenStack.* [Online] http://docs.openstack.org/developer/nova/architecture.html [Acedido: Agosto-2015].

[46] "Swift Architectural Overview". *OpenStack.* [Online] http://docs.openstack.org/developer/swift/overview_architecture.html [Acedido: Agosto-2015].

[47] Lowe, Scott. "An Introduction to OpenStack Heat". *Blog.* [Online] Maio de 2014. http://blog.scottlowe.org/2014/05/01/an-introduction-to-openstack-heat/ [Acedido: Outubro-2015].

[48] "OpenStack User Survey". *OpenStack.* [Online] Outubro de 2015. http://www.openstack.org/assets/survey/Public-User-Survey-Report.pdf [Acedido: Outubro-2015]

[49] "OpenVPN". *Wikipedia.* [Online] https://en.wikipedia.org/wiki/OpenVPN [Acedido: Agosto-2015].

[50] Clam AntiVirus. *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Clam_AntiVirus [Acedido: Agosto-2015].

[51] "What Nagios Provides". *Nagios.* [Online] https://www.nagios.org/about/overview/ [Acedido: Setembro-2015].

[52] "What is Cacti". *Cacti.* [Online] http://www.cacti.net/what_is_cacti.php [Acedido: Setembro-2015].

[53] Vanderzyden, John. "Welcome to the ELK Stack". *Qbox.* [Online] Julho de 2015. https://qbox.io/blog/welcome-to-the-elk-stack-elasticsearch-logstash-kibana [Acedido: Novembro-2015].

[54] "ownCloud Manual Installation". *ownCloud.* [Online] https://doc.owncloud.org/server/7.0/admin_manual/installation/source_installation.html [Acedido: Setembro-2015].

[55] Anicas, Michael. "How to install Nagios". *Digital Ocean.* [Online] Março de 2015. https://www.digitalocean.com/community/tutorials/how-to-install-nagios-4-and-monitor-your-servers-on-ubuntu-14-04 [Acedido: Setembro-2015].

[56] "How to install Cacti". *Unixmen.* [Online] Janeiro de 2015. http://www.unixmen.com/install-cacti-ubuntu-14-04/ [Acedido: Setembro-2015].

[57] "Ajenti Quick Install". *Ajenti.* [Online] http://support.ajenti.org/topic/349868-installing-on-ubuntu/ [Acedido: Setembro-2015].

[58] "OpenStack Lumberjack". *OpenStack.Prov2n.* [Online] Junho de 2014. http://openstack.prov12n.com/openstack-lumberjack-part-3-logstash-and-kibana/ [Acedido: Novembro-2015].

[59] James. "How to Set Up an OpenVPN Server". *Digital Ocean.* [Online] Janeiro de 2015. https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-ubuntu-14-04 [Acedido: Setembro-2015].

Anexos

Anexo A – Configurações da rede

Anexo B – Configurações de componentes OpenStack

Anexo C – Criação de Scripts

Anexo D – Implementação do ownCloud

Anexo E – Implementação das ferramentas de monitorização

Anexo F – Implementação do ELK Stack

Anexo G – Funcionamento da Dashboard OpenStack

Anexo A

Configurações da rede

A1 – Configuração dos endereços de rede dos *hosts*

Editar ficheiro /etc/network/interfaces para atribuir configurações de rede:

- *nano /etc/network/interfaces*

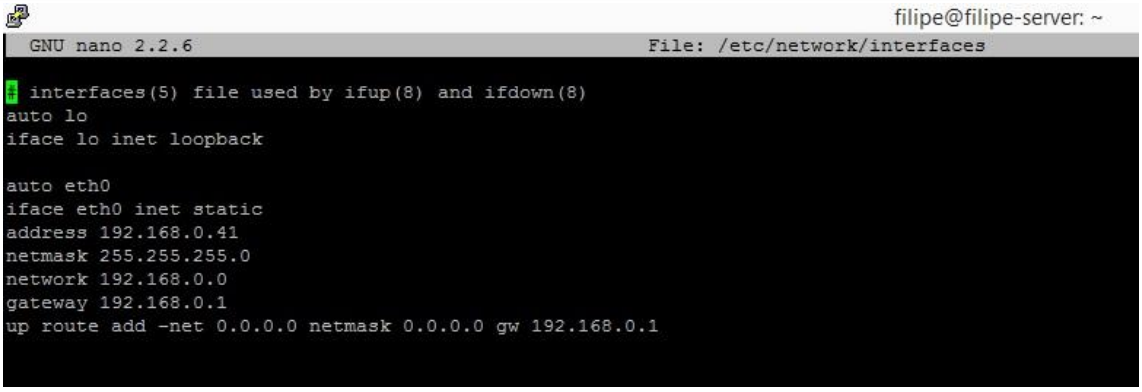


Figura 40 - Configurações de rede do Controlador

Fonte: Fonte própria

Para os restantes *hosts* foi feita a mesma configuração alterando apenas o endereço IP de cada um deles segundo a Tabela 5.

A2 – Configuração de Dynamic DNS no Router Externo

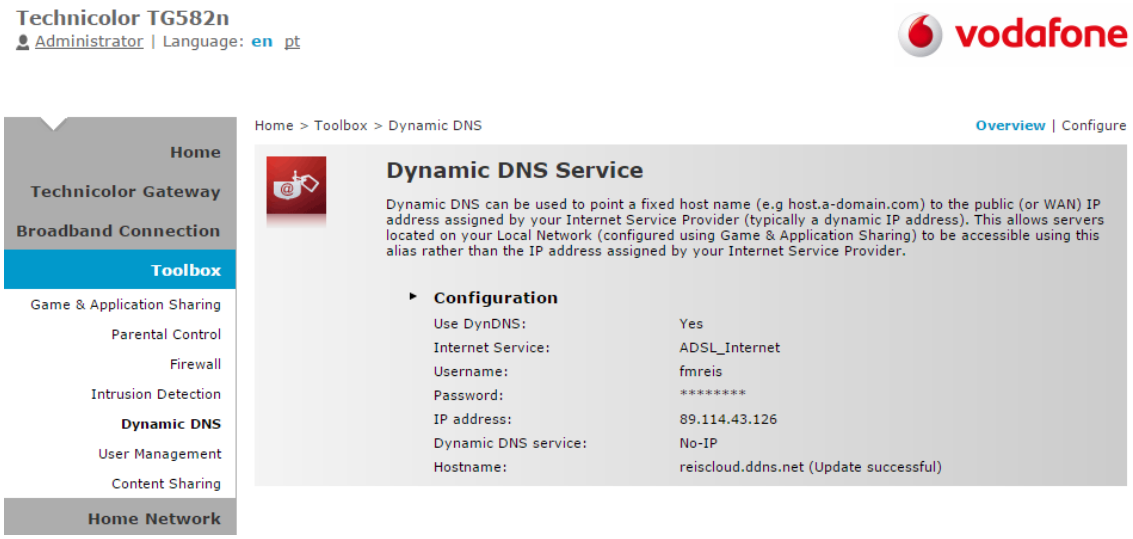


Figura 41 - Configuração do serviço Dynamic DNS

Fonte: Fonte própria

A3 – Implementação do servidor OpenVPN

Instalação dos pacotes:

- *apt-get install openvpn easy-rsa*

Configuração do OpenVPN:

- *nano /etc/openvpn/server.conf*

```
#porta para escutar
port 1194

#protocolo de transporte
proto udp

#tipo de interface
dev tun

#chaves de encriptaçao
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt
key /etc/openvpn/easy-rsa/keys/server.key  # This file should be kept
secret

#Parametros do metodo de criptografia Diffie hellman
 dh /etc/openvpn/easy-rsa/keys/dh2048.pem

#congiguraçao do tunnel
topology subnet
server 10.8.0.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
push "redirect-gateway def"

#verificar quem esta conectado
ifconfig-pool-persist ipp.txt

#servidor dns
push "dhcp-option DNS 8.8.8.8"

#keepalive pings de 10 em 10 segundos e vai a ligaçao a baixo se não
houver resposta durante 120
keepalive 10 120

#compressao no tunnel
comp-lzo

#maximo de clientes ligados
max-clients 10

#manter privilegios
persist-key
persist-tun

#estado das ligaçoes
status openvpn-status.log

#verbose moderado
verb 3
```

Criação das chaves e certificado do servidor:

- *cp -r /usr/share/easy-rsa/ /etc/openvpn*
- *mkdir /etc/openvpn/easy-rsa/keys*
- *nano /etc/openvpn/easy-rsa/vars*

  export KEY_COUNTRY="<valor>"
  export KEY_PROVINCE="<valor>"
  export KEY_CITY="<valor>"
  export KEY_ORG="<valor>"
  export KEY_EMAIL="<valor>"
  export KEY_OU="<valor>"
  export KEY_NAME="server"

- *openssl dhparam -out /etc/openvpn/dh2048.pem 2048*
- *cd /etc/openvpn/easy-rsa*
- *./clean-all*
- *./build-ca*
- *./build-key-server server*
- *cp /etc/openvpn/easy-rsa/keys/{server.crt,server.key,ca.crt} /etc/openvpn*

Criação das chaves para os clientes:

- *cd /etc/openvpn/easy-rsa*
- *./build-key cliente1*

Ficheiro de configuração do cliente, exemplo cliente.ovpn:

```
#especificaçao de cliente
client
#tipo de interface
dev tun
#protocolo de transporte
proto udp
#endereço e porta do servidor
remote reiscloud.ddns.net 1194
#fazer continuamente resolve do hostname do servidor / util em ligaçoes
wireless
resolv-retry infinite
#sem porta local vinculada
nobind
#manter o mesmo estado do cliente apos perda de ligaçao
persist-key
persist-tun
#chaves do cliente e certificado do servidor
ca ca.crt
cert cliente1.crt
key cliente1.key
#verificar o certificado do servidor
remote-cert-tls server
#ativar compressao
comp-lzo
#verbose moderado
verb 3
```

Anexo B

Configurações de componentes OpenStack

B1 – Configuração de servidor WSGI

Instalação dos pacotes no Controlador:

- *apt-get install keystone python-openstackclient apache2 libapache2-mod-wsgi memcached python-memcache*

Criação ficheiro de configuração para o servidor WSGI:

- *nano /etc/apache2/sites-available/wsgi-keystone.conf*

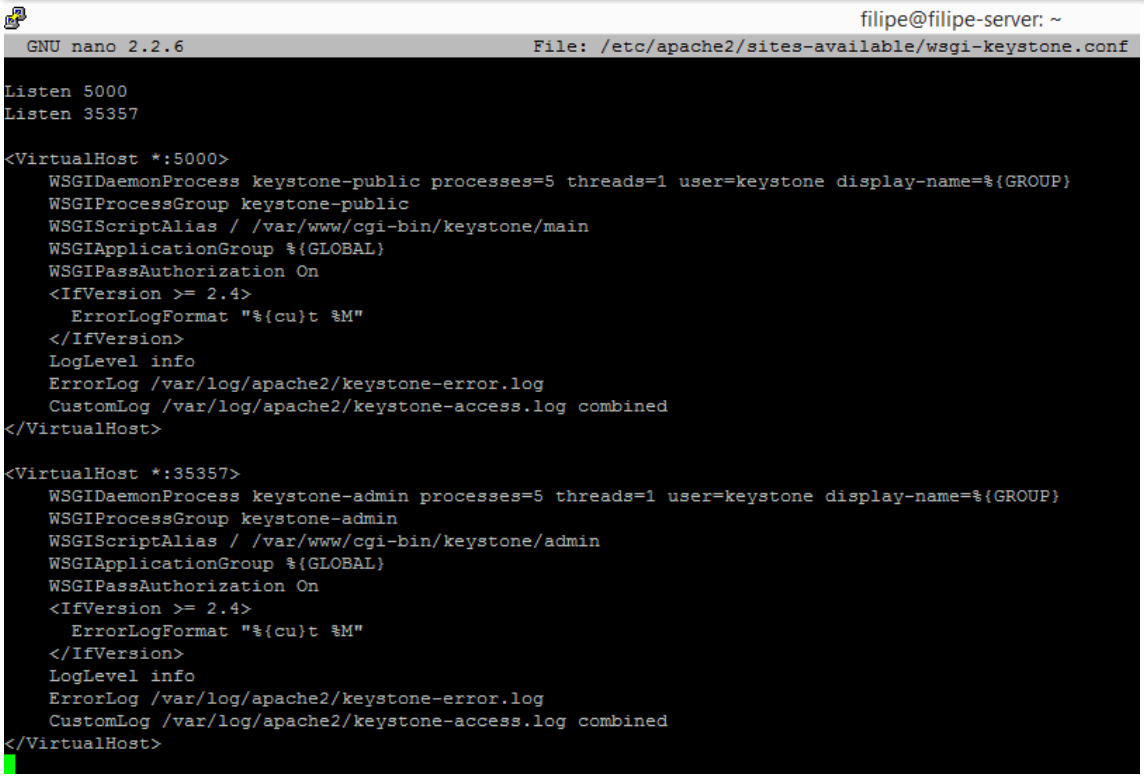


Figura 42 - Configuração do servidor WSGI

Fonte: Fonte própria

Ativação de virtual hosts para escuta nas portas configuradas:

- *ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled*

Criação do diretório para os componentes WSGI e cópia destes desde um repositório:

- *mkdir -p /var/www/cgi-bin/keystone*
- *curl http://git.openstack.org/cgit/openstack/keystone/plain/httpd/keystone.py?h=stable/kilo | tee /var/www/cgi-bin/keystone/main /var/www/cgi-bin/keystone/admin*

Atribuição de privilégios e propriedade do Keystone:

- *chown -R keystone:keystone /var/www/cgi-bin/keystone*
- *chmod 755 /var/www/cgi-bin/keystone/**

B2 – Configuração do serviço Glance

Instalação dos pacotes no Controlador:

- *apt-get install glance python-glanceclient*

Após a instalação dos pacotes, os ficheiros de configuração já vêm com algumas opções por defeito, as seguintes linhas foram adicionadas nos ficheiros glance-api.conf e glance-registry.conf.

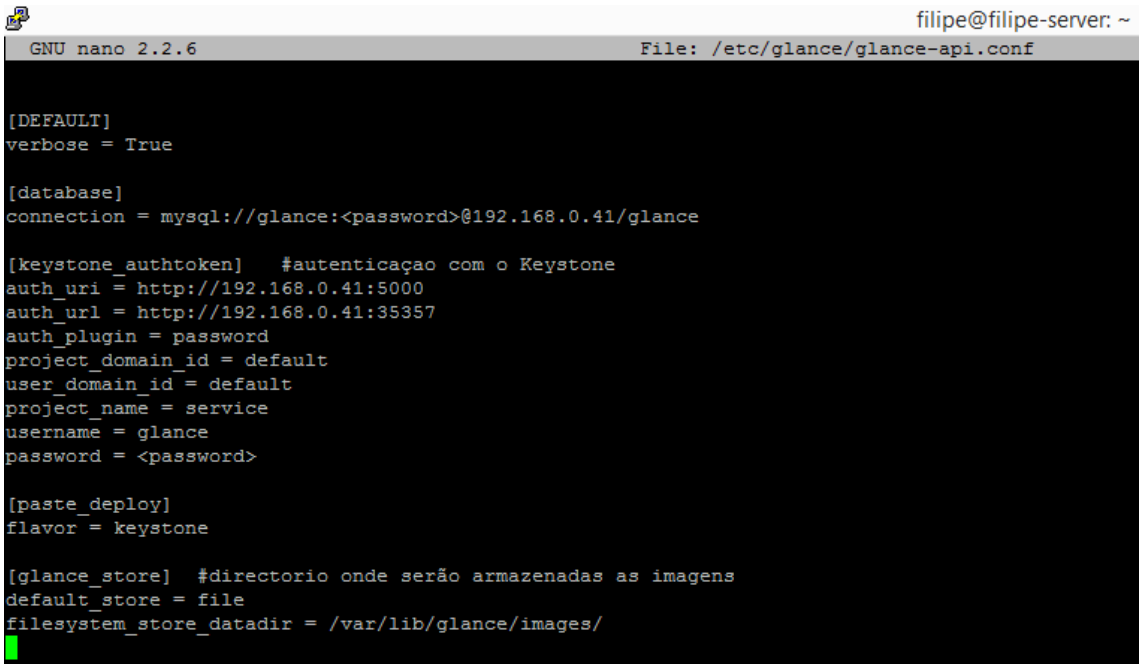


Figura 43 - Configuração do Glance

Fonte: Fonte própria

A configuração do ficheiro glance-registry.conf é semelhante à glance-api.conf com a excepção da secção [glance_store].

Sincronização do Glance à base de dados:

- *su -s /bin/sh -c "glance-manage db_sync" glance*

B3 – Configuração do serviço Nova

Instalação dos pacotes no Controlador:

- *apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy \ nova-scheduler python-novaclient nova-network sysfsutils*

Configuração do Nova no Controlador:

```
[DEFAULT]
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = messagingv2
verbose = True
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
log_dir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
force_dhcp_release=True
libvirt_use_virtio_for_bridges=True
ec2_private_dns_show_ip=True
api_paste_config=/etc/nova/api-paste.ini
enabled_apis=ec2,osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.0.41
vnc_enabled = True
vncserver_listen = 192.168.0.41
vncserver_proxyclient_address = 192.168.0.41
novncproxy_base_url = http://192.168.0.41:6080/vnc_auto.html
xvpvncproxy_base_url = http://192.168.0.41:6081/console
keystone_ec2_url = http://192.168.0.41:5000/v2.0/ec2tokens
ec2_dmz_host = 192.168.0.41
network_api_class = nova.network.api.API
security_group_api = nova
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
allow_resize_to_same_host=True
allow_migrate_to_same_host=True
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
force_dhcp_release = True
flat_network_bridge = br100
flat_interface = eth0
public_interface = br100
vlan_interface = eth0

[database]
connection = mysql://nova:<password>@192.168.0.41/nova

[glance]
host = 192.168.0.41

[keystone_authtoken]
auth_uri = http://192.168.0.41:5000
auth_url = http://192.168.0.41:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = <password>
```

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

[oslo_messaging_rabbit]
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>

[libvirt]
virt_type = kvm
```

Sincronização do Nova à base de dados:

```
su -s /bin/sh -c "nova-manage db_sync" nova
```

Instalação dos pacotes no Nova *node*:

- *apt-get install nova-compute nova-network sysfutils*

Configuração do Nova *node*:

```
[DEFAULT]
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = messagingv2
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
force_dhcp_release=True
libvirt_use_virtio_for_bridges=False
verbose=True
ec2_private_dns_show_ip=True
api_paste_config=/etc/nova/api-paste.ini
enabled_apis=ec2,osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.0.40
vnc_enabled = True
vncserver_listen = 192.168.0.40
vncserver_proxyclient_address = 192.168.0.40
novncproxy_base_url = http://192.168.0.41:6080/vnc_auto.html
xvpvncproxy_base_url = http://192.168.0.41:6081/console
xvpvncproxy_host = 0.0.0.0
novncproxy_host = 0.0.0.0
network_api_class = nova.network.api.API
security_group_api = nova
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
allow_resize_to_same_host=true
allow_migrate_to_same_host=true
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
```

```
flat_network_bridge = br100
flat_interface = eth0
public_interface = br100
vlan_interface = eth0
enabled_apis = ec2, osapi_compute, metadata
ec2_dmz_host = 192.168.0.41

[glance]
host = 192.168.0.41
api_servers = 192.168.0.41:9292

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

[oslo_messaging_rabbit]
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>

[keystone_authtoken]
auth_uri = http://192.168.0.41:5000
auth_url = http://192.168.0.41:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = <password>

[libvirt]
virt_type = qemu
```

B4 – Configuração do serviço Cinder

Instalação dos pacotes no Controlador:

- *apt-get install cinder-api cinder-backup cinder-volume cinder-scheduler *
- *python-cinderclient lvm2*

Criação do volume físico e associação deste volume ao grupo cinder-volumes no Controlador:

- *pvcreate /dev/sda6*
- *vgcreate cinder-volumes /dev/sda6*

Instalação dos pacotes no Nova *node*:

- *apt-get install cinder-backup cinder-volume python-mysqldb lvm2*

Criação do volume físico e associação deste volume ao grupo cinder-volumes no Nova *node*:

- *pvcreate /dev/sda5*
- *vgcreate cinder-volumes /dev/sda5*

A configuração é semelhante para os dois *nodes* alterando apenas o endereço IP do *node* em questão no ficheiro cinder.conf:

```
[DEFAULT]
Verbose = True
backup_driver = cinder.backup.drivers.swift
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
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 = 192.168.0.41
control_exchange = cinder
notification_driver = messagingv2
enabled_backends = lvm
glance_host = 192.168.0.41

[database]
connection = mysql://cinder:<password>@192.168.0.41/cinder

[keystone_authtoken]
auth_uri = http://192.168.0.41:5000
auth_url = http://192.168.0.41:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = <password>

[oslo_concurrency]
lock_path = /var/lock/cinder

[lvm]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = tgtadm

[oslo_messaging_rabbit]
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>
```

Sincronização do Cinder à base de dados:

- *su -s /bin/sh -c "cinder-manage db_sync" cinder*

B5 – Configuração do serviço Swift

Ao contrário dos outros serviços, o Swift não utilizada uma base de dados SQL no Controlador, em vez disso utiliza bases de dados SQLite distribuídas por cada *node* de armazenamento.

Instalação dos pacotes no Controlador:

- *apt-get install swift swift-proxy python-swiftclient python-keystoneclient \ python-keystonemiddleware memcached*

Download do ficheiro de configuração proxy-server e adição das seguintes linhas:

- *curl -o /etc/swift/proxy-server.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/proxy-server.conf-sample?h=stable/kilo*

```
[DEFAULT]
bind_port = 8080
user = swift
swift_dir = /etc/swift

[app:proxy-server]
account_autocreate = true

[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
auth_uri = http://192.168.0.41:5000
auth_url = http://192.168.0.41:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = swift
password = <password>
delay_auth_decision = true

[filter:cache]
memcache_servers = 127.0.0.1:11211

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = admin,user

[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache
container_sync bulk ratelimit authtoken keystoneauth container-quotas
account-quotas slo dlo proxy-logging proxy-server
```

Instalação dos pacotes nos *nodes* de armazenamento:

- *apt-get install xfsprogs rsync swift swift-account swift-container swift-object*

Configuração do rsync:

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = <ip_do_node>

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

Download dos ficheiros de configuração dos serviços Swift:

- *curl –o /etc/swift/account-server.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/account-server.conf-sample?h=stable/kilo*
- *curl -o /etc/swift/container-server.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/container-server.conf-sample?h=stable/kilo*
- *curl -o /etc/swift/object-server.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/object-server.conf-sample?h=stable/kilo*
- *curl -o /etc/swift/container-reconciler.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/container-reconciler.conf-sample?h=stable/kilo*
- *curl -o /etc/swift/object-expirer.conf \ https://git.openstack.org/cgit/openstack/swift/plain/etc/object-expirer.conf-sample?h=stable/kilo*

Adição das seguintes linhas ao ficheiro account-server.conf:

```
[DEFAULT]
bind_ip = <ip_do_node>
bind_port = 6002
user = swift
swift_dir = /etc/swift
devices = /srv/node

[pipeline:main]
pipeline = healthcheck recon account-server

[filter:recon]
recon_cache_path = /var/cache/swift
```

Adição das seguintes linhas ao ficheiro container-server.conf:

```
[DEFAULT]
bind_ip = <ip_do_node>
bind_port = 6001
user = swift
swift_dir = /etc/swift
devices = /srv/node

[pipeline:main]
pipeline = healthcheck recon container-server

[filter:recon]
recon_cache_path = /var/cache/swift
```

Adição das seguintes linhas ao ficheiro object-server.conf:

```
[DEFAULT]
bind_ip = <ip_do_node>
bind_port = 6000
user = swift
swift_dir = /etc/swift
devices = /srv/node

[pipeline:main]
pipeline = healthcheck recon object-server

[filter:recon]
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

Atribuir privilégios ao Swift sobre os dispositivos montados e criação de diretório para *cache*:

- *chown -R swift:swift /srv/node*
- *mkdir -p /var/cache/swift*
- *chown -R swift:swift /var/cache/swift*

B6 – Configuração do serviço Horizon

Instalação dos pacotes no Controlador:

- *apt-get install openstack-dashboard*

Adição das seguintes linhas ao ficheiro de configuração do Horizon:

```
OPENSTACK_HOST = "192.168.0.41"
ALLOWED_HOSTS = ['*', ]
CACHES = {
    'default': {
        'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
TIME_ZONE = "UTC"
```

B7 – Configuração do serviço Ceilometer

Instalação dos pacotes no Controlador:

- *apt-get install mongodb-server mongodb-clients python-pymongo ceilometer-api \ ceilometer-collector ceilometer-agent-central ceilometer-agent-notification \ ceilometer-alarm-evaluator ceilometer-alarm-notifier python-ceilometerclient*

Adição das seguintes linhas ao ficheiro de configuração do Ceilometer:

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
verbose = True

[database]
connection                                                    =
mongodb://ceilometer:<password>@192.168.0.41:27017/ceilometer

[keystone_authtoken]
auth_uri = http://192.168.0.41:5000/v2.0
identity_uri = http://192.168.0.41:35357
admin_tenant_name = service
admin_user = ceilometer
admin_password = <password>

[publisher]
telemetry_secret = <secret>

[service_credentials]
os_auth_url = http://192.168.0.41:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = <password>
os_endpoint_type = internalURL
os_region_name = RegionOne
```

```
[oslo_messaging_rabbit]
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>
```

Instação do ceilometer-agent nos *nodes* que fazem computação:

- *apt-get install ceilometer-agent-compute*

Adição das seguintes linhas ao ficheiro de configuração do Nova:

```
[DEFAULT]
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = messagingv2
```

Adição das seguintes linhas aos ficheiros de configuração do Glance:

```
[DEFAULT]
notification_driver = messagingv2
rpc_backend = rabbit
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>
```

Adição das seguintes linhas ao ficheiro de configuração do Cinder:

```
[DEFAULT]
control_exchange = cinder
notification_driver = messagingv2
```

Adição das seguintes linhas ao ficheiro de configuração do Proxy-server do Swift:

```
[filter:ceilometer]
paste.filter_factory = ceilometermiddleware.swift:filter_factory
control_exchange = swift
url = rabbit://openstack:<password>@192.168.0.41:5672/
driver = messagingv2
topic = notifications
log_level = WARN

[filter:keystoneauth]
operator_roles = admin,user,ResellerAdmin

[pipeline:main]
pipeline = authtoken cache healthcheck keystoneauth proxy-logging
ceilometer proxy-server
```

B8 – Configuração do serviço Heat

Instalação dos pacotes no Controlador:

- *apt-get install heat-api heat-api-cfn heat-engine python-heatclient*

Configuração do Heat:

```
[DEFAULT]
Verbose = True
rpc_backend = rabbit
heat_metadata_server_url = http://192.168.0.41:8000
heat_waitcondition_server_url                                                    =
http://192.168.0.41:8000/v1/waitcondition
stack_domain_admin = heat_domain_admin
stack_domain_admin_password = <password>
stack_user_domain_name = heat_user_domain

[database]
connection = mysql://heat:<password>@192.168.0.41/heat

[ec2authtoken]
auth_uri = http://192.168.0.41:5000/v2.0

[keystone_authtoken]
auth_uri = http://192.168.0.41:5000/v2.0
identity_uri = http://192.168.0.41:35357
admin_tenant_name = service
admin_user = heat
admin_password = <password>

[oslo_messaging_rabbit]
rabbit_host = 192.168.0.41
rabbit_userid = openstack
rabbit_password = <password>
```

Exemplo de template que cria um stack com duas máquinas virtuais:

```
heat_template_version: 2013-05-23

description: Criar duas VMs

parameters:
  imagem_1:
    type: string
    label: Image Name
    description: Imagem do SO da VM1
    default: cirrOS
  imagem_2:
    type: string
    label: Image Name
    description: Imagem do SO da VM2
    default: ubuntu
  network_id:
    type: string
    label: Network ID
    description: Rede das VMs para o Stack
    default: v-net
```

```
resources:
  servidor1:
    type: OS::Nova::Server
    properties:
      image: { get_param: imagem_1 }
      flavor: micro
      networks:
        - network : { get_param : network_id }
  servidor2:
    type: OS::Nova::Server
    properties:
      image: { get_param: imagem_2 }
      flavor: pequeno
      networks:
        - network : { get_param : network_id }
```

Anexo C

Criação de Scripts

C1 – Scripts com credenciais dos utilizadores *admin* e *filipe* para autenticação com o Keystone

O URL de autenticação é diferente porque o admin utiliza o *endpoint* de admin e o utilizador filipe utiliza o URL público/interno.

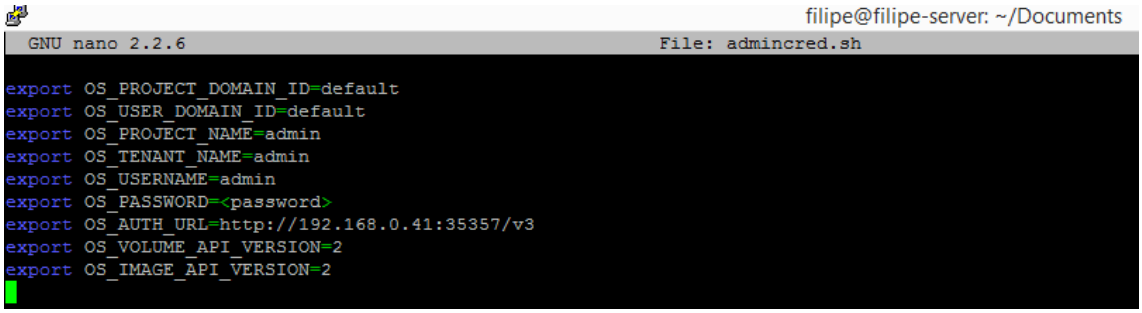


Figura 44 – Autenticação com credenciais de admin

Fonte: Fonte própria

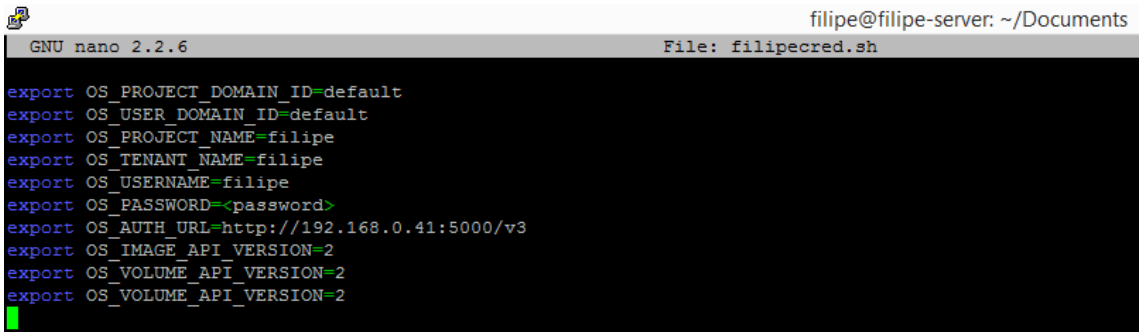


Figura 45 - Autenticação com credenciais de utilizador

Fonte: Fonte própria

C2 – Script com regras da *firewall* para o servidor OpenVPN

```
# Permitir que o trafego da VPN aceda a LAN
    iptables -I FORWARD -i tun0 -o eth0 \
         -s 10.8.0.0/24  \
         -m conntrack --ctstate NEW -j ACCEPT

# Permitir que o trafego estabelecido seja aceite em ambas as direcoes
    iptables -I FORWARD -m conntrack --ctstate RELATED,ESTABLISHED \
         -j ACCEPT

#Permitir que o trafego da VPN aceda a Internet pela interface eth0
    iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
```

C3 – Script para criação de máquinas virtuais

```
#!/bin/bash
#autenticaçao do utilizador
read -p "Introduza o nome do utilizador : " utilizadorvm
echo "Introduza a password"
stty -echo
read utipassvm
stty echo

#autenticacao admin para criacao da flavor
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<password>
export OS_AUTH_URL=http://192.168.0.41:35357/v3
export OS_VOLUME_API_VERSION=2
export OS_IMAGE_API_VERSION=2


#criacao de um numero unico
horas=`date +%H`
minutos=`date +%M`
segundos=`date +%S`
randomstring="$(($horas+$minutos)*($segundos))"

#colocar um numero unico no flavorname para nao existirem flavors
repitidas
flavorname=$utilizadorvm$randomstring
flavorid=$randomstring

#leitura dos dados de computacao que o utilizador necessita
echo "-> Recursos Computacionais <-"
read -p "Quantos vCPUs?(1-2) - " vcpuflavor
read -p "Quanta RAM?(0-2500 MB) - " ramflavor
read -p "Quanto espaço em disco?(0-80 GB) - " discoflavor

#criacao da flavor
nova flavor-create $flavorname $flavorid $ramflavor $discoflavor
$vcpuflavor

#escolha do SO
read -p "Escolha o SO (1 - Ubuntu | 2 - CentOS | 3 - Fedora | 4 -
openSUSE | 5 - Debian | 6 - cirrOS) - " valorso
if [ $valorso -eq "1" ]; then
    echo; soescolhido='ubuntu'
    echo Escolheu o sistema operativo - $soescolhido
elif [ $valorso -eq "2" ]; then
    echo; soescolhido='CentOS'
    echo Escolheu o sistema operativo - $soescolhido
elif [ $valorso -eq "3" ]; then
    echo; soescolhido='Fedora'
    echo Escolheu o sistema operativo - $soescolhido
elif [ $valorso -eq "4" ]; then
    echo; soescolhido='openSUSE'
    echo Escolheu o sistema operativo - $soescolhido
elif [ $valorso -eq "5" ]; then
    echo; soescolhido='Debian'
```

```
    echo Escolheu o sistema operativo - $soescolhido
elif [ $valorso -eq "6" ]; then
    echo; soescolhido='cirrOS'
    echo Escolheu o sistema operativo - $soescolhido
fi

#escolha do nome para o servidor
read -p "Escolha um nome para o seu servidor - " nomecloudserver

#autenticaçao do utilizador
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=$utilizadorvm
export OS_TENANT_NAME=$utilizadorvm
export OS_USERNAME=$utilizadorvm
export OS_PASSWORD=$utipassvm
export OS_AUTH_URL=http://192.168.0.41:5000/v3
export OS_VOLUME_API_VERSION=2
export OS_IMAGE_API_VERSION=2

#criaçao da chave publica e privada para acesso ao servidor
read -p "Escolha o nome da chave de autenticaçao - " chave
ssh-keygen -t rsa -f $chave
aux=".pub"

nova keypair-add --pub-key /var/www/html/reiscloud/scripts/$chave$aux
$chave


#criacao do servidor
openstack server create $nomecloudserver --image $soescolhido --flavor
$flavorname --availability-zone Controlador --key-name $chave
```

C4 – Script para atribuição de quotas/limites de armazenamento

```
#!/bin/bash
#autenticaçao do utilizador
read -p "Introduza o nome do utilizador : " utilizadorswift
echo "Introduza a password"
stty -echo
read swiftpass
stty echo

#leitura do espaço para armazenamento
echo "-> Espaço em disco <-"
read -p "Quanto espaço de armazenamento (MB) - " armazenaswift

#autenticaçao do utilizador
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=$utilizadorswift
export OS_TENANT_NAME=$utilizadorswift
export OS_USERNAME=$utilizadorswift
export OS_PASSWORD=$swiftpass
export OS_AUTH_URL=http://192.168.0.41:5000/v3
export OS_VOLUME_API_VERSION=2
export OS_IMAGE_API_VERSION=2
```

```
#limitaçao do armazenamento
swift -V 3 post -m quota-bytes:$armazenaswift

#criaçao do container do utilizador, com ficheiro de boas vindas
swift -V 3 upload $utilizadorswift bemvindo.txt
```

C5 – Script para definir permissões seguras para o ownCloud

```
#!/bin/bash
ocpath='/var/www/html/owncloud'
htuser='www-data'
htgroup='www-data'

find ${ocpath}/ -type f -print0 | xargs -0 chmod 0640
find ${ocpath}/ -type d -print0 | xargs -0 chmod 0750

chown -R root:${htuser} ${ocpath}/
chown -R ${htuser}:${htgroup} ${ocpath}/apps/
chown -R ${htuser}:${htgroup} ${ocpath}/config/
chown -R ${htuser}:${htgroup} ${ocpath}/data/
chown -R ${htuser}:${htgroup} ${ocpath}/themes/

chown root:${htuser} ${ocpath}/.htaccess
chown root:${htuser} ${ocpath}/data/.htaccess

chmod 0644 ${ocpath}/.htaccess
chmod 0644 ${ocpath}/data/.htaccess
```

C6 – Script para criação de utilizador OpenStack através de interação com página PHP

```
#!/bin/bash
#autenticaçao para admin na API
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<password>
export OS_AUTH_URL=http://192.168.0.41:35357/v3
export OS_VOLUME_API_VERSION=2
export OS_IMAGE_API_VERSION=2

utilizador=$1
password=$2

openstack project create --description "Projeto do $utilizador"
$utilizador
openstack user create --password $password
openstack role add --project $utilizador --user $utilizador user
```

Página PHP:

```
<!DOCTYPE html>
<html>
<head>
<title>Registo de utilizador</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<img src="../images/reiscloud.png" alt="reiscloud">
<div id="main">
<h1>Registo de utilizador</h1>
<div id="login">
<h2>Registo</h2>
<form action="" method="post">
<label>Utilizador :</label>
<input id="utilizador" name="utilizador" placeholder="Nome de
utilizador" type="text">
<label>Password :</label>
<input id="password" name="password" placeholder="**********"
type="password">
<input name="submit" type="submit" value=" Registar ">
<span><?php echo $error; ?></span>
</form>
</div>
</div>
</body>
</html>
<?php
if (isset($_POST['submit'])){
if (empty($_POST['utilizador']) || empty($_POST['password'])){
$error = "Utilizador ou Password estao errados";
}
else
{
$utilizador=$_POST['utilizador'];
$password=$_POST['password'];
$output = exec("./aut.sh $utilizador $password");
}
}
?>
```

Figura 46 - Página web para criação de utilizadores OpenStack

Fonte: Fonte própria

Anexo D

Implementação do ownCloud

Instalação dos pacotes e dependências do PHP e MySQL no Swift *node*:

- *add-apt-repository ppa:ondrej/php5*
- *apt-get update*
- *apt-get install apache2 mysql-server libapache2-mod-php5*
- *apt-get install php5 php5-gd php5-json php5-mysql php5-curl*
- *apt-get install php5-intl php5-mcrypt php5-imagick*

Criação da base de dados e atribuição de privilégios:

- *mysql –u root –p*
- *CREATE DATABASE owncloud;*
- *GRANT ALL ON owncloud.* to 'owncloud'@'localhost' IDENTIFIED BY 'database_password';*

Download da chave associada ao ownCloud:

- *cd /tmp*
- *wget http://download.opensuse.org/repositories/isv:ownCloud:community/xUbuntu_14.04/Release.key*
  apt-key add - < Release.key

Adição do repositório ownCloud e instalação dos pacotes:

- *sh -c "echo 'deb http://download.opensuse.org/repositories/isv:/ownCloud:/community/xUbuntu_14.04/ /' >> /etc/apt/sources.list.d/owncloud.list"*
- *apt-get update*
- *apt-get install ownCloud*

Definir permissões seguras ao ownCloud:

- *sh ownCloud.sh* (Anexo C5)

Anexo E

Implementação das ferramentas de monitorização

Instalação do SNMP, SNMPD em todos os *nodes* e configuração da Community:

- *apt-get install snmp snmpd*
- *nano /etc/snmp/snmpd.conf*

Adição das seguintes linhas:

```
rocommunity <community>
syslocation "<Localidade>"
syscontact "<Contacto>"
```

Implementação do Nagios no Gestor *node*

Criação de um utilizador e grupo para operar o Nagios e privilégios sobre o Apache:

- *useradd nagios*
- *groupadd nagcmd*
- *usermod –a –G ngcmd nagios*
- *usermod –a –G nagcmd www-data*

Instalação dos pacotes das dependências:

- *apt-get install build-essential apache2 apache2-utils php5-gd libgd2-xpm-dev libapache2-mod-php*

Download e instalação do Nagios Core e Plugins:

- *cd /tmp*
- *wget http://switch.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz*
- *wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz*
- *tar -xvf nagios-4.0.8.tar.gz*
- *tar -xvf nagios-plugins-2.0.3.tar.gz*
- *cd /tmp/nagios-4.0.8*
- *sudo make all*
- *sudo make install*
- *sudo make install-init*
- *sudo make install-config*
- *sudo make install-commandmode*
- *sudo cp -R contrib/eventhandlers/ /usr/local/nagios/libexec/*
- *sudo chown -R nagios:nagios /usr/local/nagios/libexec/eventhandlers*
- *sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg*
- *sudo ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios*
- *cd /tmp/nagios-plugins-2.0.3/*

- *sudo ./configure --with-nagios-user=nagios --with-nagios-group=nagios --enable-perl-modules --enable-extra-opts*
- *sudo make*
- *sudo make install*

Criar o ficheiro de configuração do Nagios no Apache:

- nano /etc/apache2/sites-enabled/nagios.conf

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">
   Options ExecCGI
   AllowOverride None
   Order allow,deny
   Allow from all
   AuthName "Restricted Area"
   AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>

Alias /nagios "/usr/local/nagios/share"

<Directory "/usr/local/nagios/share">
   Options None
   AllowOverride None
   Order allow,deny
   Allow from all
   AuthName "Restricted Area"
   AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>
```

Ativar os módulos *rewrite* e *cgi* do Apache:

- *a2enmod cgi*
- *a2enmod rewrite*

Adicionar utilizador "nagiosadmin" à *interface web*:

- *sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin*

Instalação do Nagios NRPE no Gestor *node*:

- *curl -L -O http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz*
- *tar xvf nrpe-*.tar.gz*
- *cd nrpe-**
- *./configure --enable-command-args --with-nagios-user=nagios --with-nagios-group=nagios --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib/x86_64-linux-gnu*
- *make all*
- *make install*
- *make install-xinetd*
- *make install-daemon-config*

Configuração do NRPE e adicionado o endereço do Gestor *node* para que apenas este possa requisitar métricas dos recursos:

- *nano /etc/xinetd.d/nrpe*

```
only_from = 127.0.0.1, 192.168.0.43
```

Instalação do Nagios NRPE e Plugins nos restantes *nodes:*

- *apt-get install nagios-plugins nagios-nrpe-server*
- *nano /etc/nagios/nrpe.cfg*

```
allowed_hosts=127.0.0.1, 192.168.0.43
nrpe_user=nagios
nrpe_group=nagios
```

Criação de ficheiros de monitorização dos *nodes*, exemplo *controlador.cfg*:

```
#nome do host
define host{
        use                     linux-server            ; Name of host
template to use
                                ; This host definition will inherit all
variables that are defined
                                ; in (or inherited by) the linux-server host
template definition.
        host_name               Controlador
        alias                   Controlador
        address                 192.168.0.41
        }
```

```
#a que grupo de hosts pertence
define hostgroup{
        hostgroup_name  linux-servers ; The name of the hostgroup
        alias           Linux Servers ; Long name of the group
        members           Gestor,Controlador, Nova, Swift      ; Comma
separated list of hosts that belong to this group
        }



#verificar o estado da máquina (up/down)
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             PING
    check_command           check_ping!100.0,20%!500.0,60%
        }


#verificar o espaço em disco, envia alerta se tiver menos de 20%, e
alerta critico se menos de 10%
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             Root Partition
    check_command           check_local_disk!20%!10%!/
        }



# verificar o nº de utilizadores, alerta com mais de 10 e critico se
tiver mais de 20
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             Current Users
    check_command           check_local_users!10!20
        }


# verificar o nº de processos, alerta com mais de 250 e critico com mais
de 400
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             Total Processes
    check_command           check_local_procs!250!400!RSZDT
        }
```

```
# verificar a carga (load average/ocupação do CPU) no host,  alerta caso
passe os valores (5.0,4.0,3.0) e critico (10.0,6.0,4.0)
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             Current Load
    check_command           check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
        }


# verificar a swap, alerta caso passe 20% critico se passar os 10%
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             Swap Usage
    check_command           check_local_swap!20!10
        }


# verificar ssh, sem notificações ativadas
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             SSH
    check_command           check_ssh
    notifications_enabled       0
        }


# verificar http, sem notificaçoes ativadas
define service{
        use                             local-service         ; Name of
service template to use
        host_name                       Controlador
        service_description             HTTP
    check_command           check_http
    notifications_enabled       0
        }
```

Adição da localização deste *node* ao ficheiro de configuração do Nagios (/usr/local/nagios/etc/nagios.cfg):

- echo “cfg_file=<caminho>/controlador.cfg” >> /usr/local/nagios/etc/nagios.cfg

Implementação do Cacti no Gestor *node*

Instalação dos pacotes do RRDtools, Cacti e Spine:

- *apt-get install rrdtool cacti cacti-spine*

As restantes configurações são feitas graficamente:

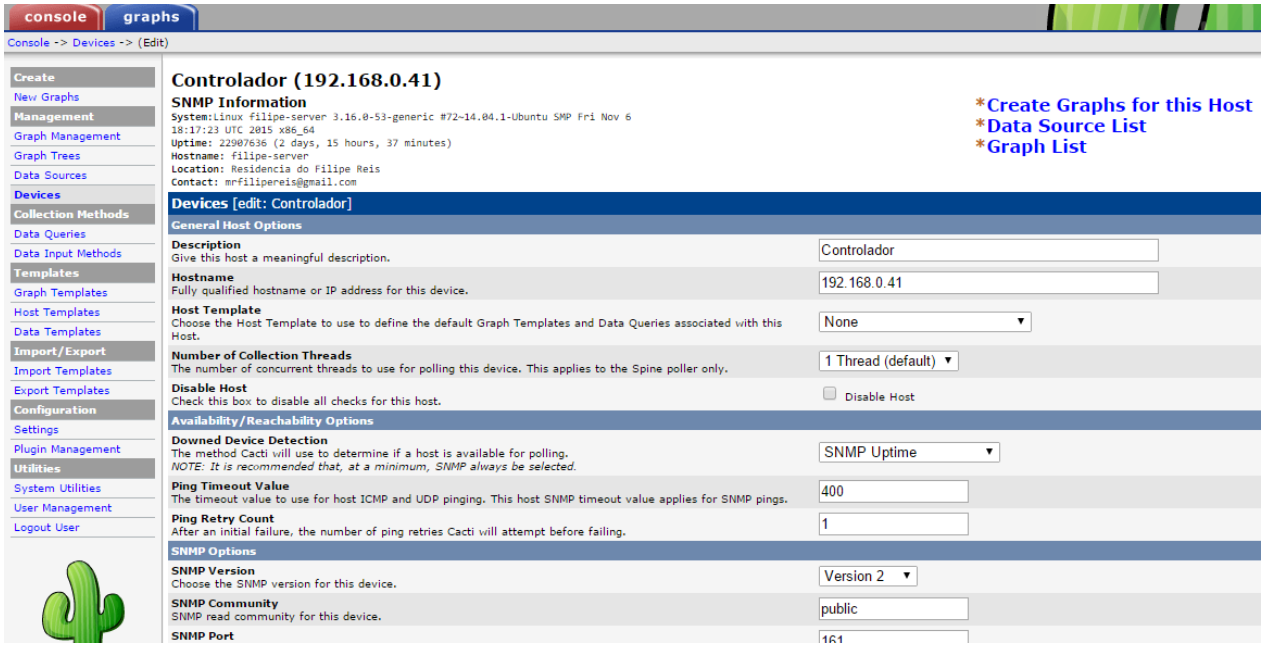


Figura 47 - Configuração dos nodes para monitorização no Cacti

Fonte: Fonte própria

Implementação do Ajenti

Adição do repositório e instalação dos pacotes:

- *echo "deb http://repo.ajenti.org/debian main main" >> /etc/apt/sources.list*
- *wget http://repo.ajenti.org/debian/key -O- | sudo apt-key add -*
- *sudo apt-get update*
- *sudo apt-get install ajent*

Anexo F

Implementação do ELK Stack

Configuração do servidor Rsyslog para receber os *logs* dos *nodes*:

- nano /etc/rsyslog.d/servidor.conf

```
#Ativar UDP
$ModLoad imudp
#Escutar no endereço ip do servidor e porta 514
$UDPServerAddress <ip_servidor>
$UDPServerRun 514

#Templates dos serviços
$template Nova,"/var/log/rsyslog/%HOSTNAME%/nova.log"
$template Glance,"/var/log/rsyslog/%HOSTNAME%/glance.log"
$template Cinder,"/var/log/rsyslog/%HOSTNAME%/cinder.log"
$template Keystone,"/var/log/rsyslog/%HOSTNAME%/keystone.log"

#Passar tudo o resto para o syslog.log
$template DynFile,"/var/log/rsyslog/%HOSTNAME%/syslog.log"
*.* ?DynFile

#Passar os logs dos componentes openstack para ficheiros individuais
local0.* ?Nova
local1.* ?Glance
local2.* ?Cinder
local3.* ?Keystone
& ~
```

Adição das seguintes linhas aos ficheiros de configuração dos serviços OpenStack:

```
use_syslog = True
syslog_log_facility=<Log_Local do Serviço como especificado no servidor>
#Exemplo se for o serviço Cinder syslog_log_facility=LOG_LOCAL2
```

Adição da seguinte linha aos *nodes*, para enviarem os *logs* para o servidor de *logs:*

- *echo "*.* @<ip_servidor>" >> /etc/rsyslog.d/cliente.conf*

Instalação das dependências do ELK Stack:

- *sudo apt-get install python-software-properties software-properties-common apache2 -y*
- *sudo add-apt-repository ppa:webupd8team/java*
- *sudo apt-get update*
- *echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections*
- *echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections*
- *sudo apt-get -q -y install oracle-java7-installer*
- *sudo bash -c "echo JAVA_HOME=/usr/lib/jvm/java-7-oracle/ >> /etc/environment"*

Instalação do Elastic Search:

- *wget -O - http://packages.elasticsearch.org/GPG-KEY-elasticsearch | sudo apt-key add -*
- *echo 'deb http://packages.elasticsearch.org/elasticsearch/1.1/debian stable main' | sudo tee /etc/apt/sources.list.d/elasticsearch.list*
- *sudo apt-get update*
- *sudo apt-get install -y elasticsearch=1.1.1*
- *sudo service elasticsearch start*
- *sudo update-rc.d elasticsearch defaults 95 10*

Instalação do Kibana:

- *cd ~; wget http://download.elasticsearch.org/kibana/kibana/kibana-latest.zip*
- *unzip kibana-latest.zip*
- *sudo mkdir -p /var/www/kibana*
- *sudo cp -R ~/kibana-latest/* /var/www/kibana/*
- *sudo cat > /etc/apache2/conf-enabled/kibana.conf <<EOF*
  Alias /kibana /var/www/kibana
  <Directory /var/www/kibana>
  Order allow,deny
  Allow from all
  </Directory>
  EOF
- *sudo service apache2 restart*

Instalação do Logstash:

- *echo 'deb http://packages.elasticsearch.org/logstash/1.4/debian stable main' | sudo tee /etc/apt/sources.list.d/logstash.list*
- *sudo apt-get update*
- *sudo apt-get install -y logstash=1.4.1-1-bd507eb*

Configuração do Logstash:

- *sudo nano /etc/logstash/conf.d/10-syslog.conf*

```
input {
  tcp {
    port => 9000
    type => syslog
  }
  udp {
    port => 9000
    type => syslog
  }
}
```

```
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss"
]
    }
  }
}
output {
  elasticsearch { host => localhost }
  stdout { codec => rubydebug }
```

Adição da seguinte linha para enviar os *logs* do rsyslog para o Logstash:

- *echo "*.* @@localhost:9000" >> /etc/rsyslog.d/servidor.conf*

Anexo G

Funcionamento da Dashboard OpenStack

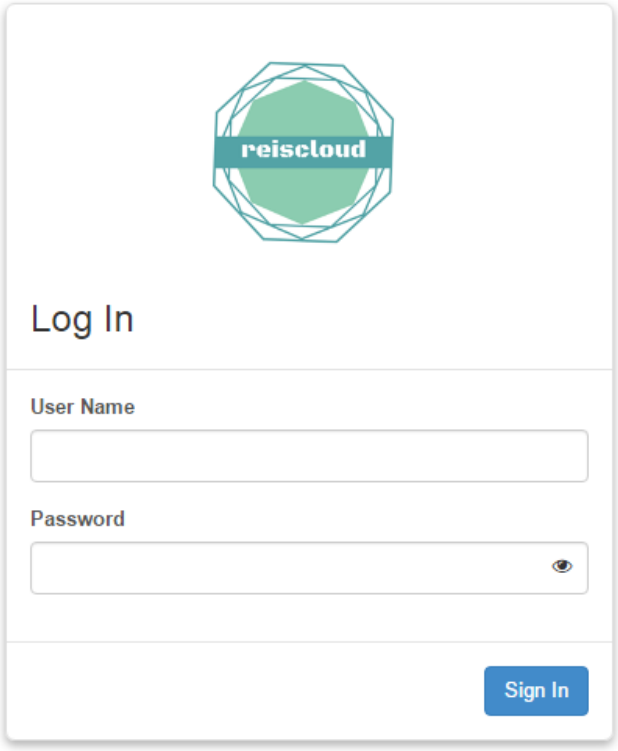


Figura 48 - Login OpenStack Horizon

Fonte: Fonte própria

Interface de administrador

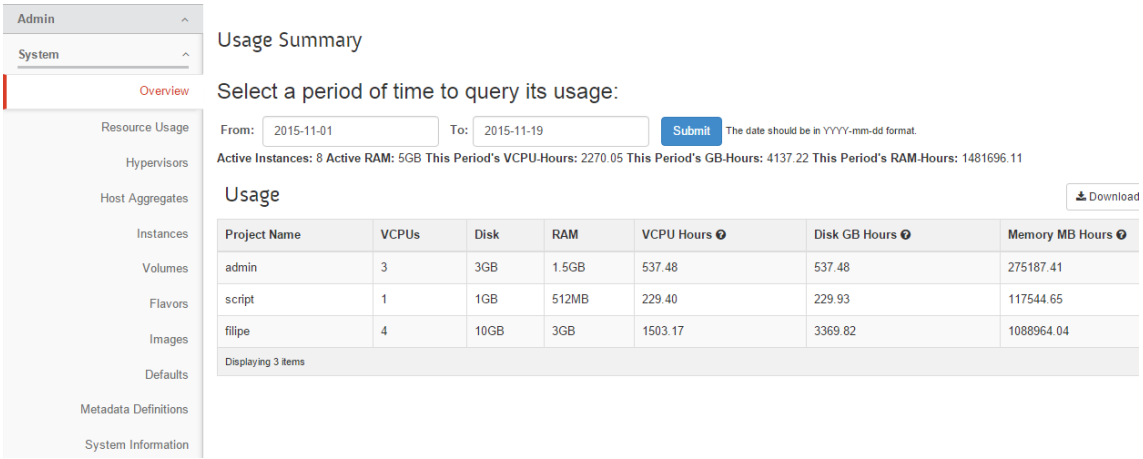


Figura 49 - Visualização dos recursos utilizados por vários projetos

Fonte: Fonte própria

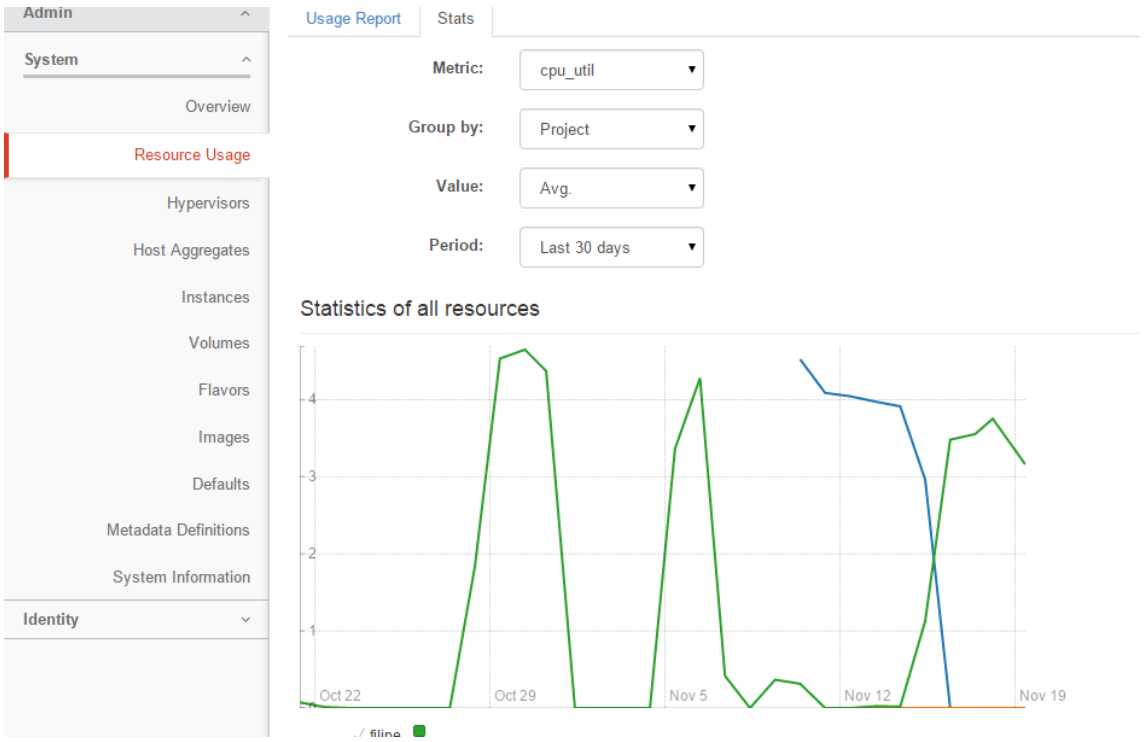


Figura 50 - Estatística gerada pelo Ceilometer sobre a utilização média de CPU

Fonte: Fonte própria

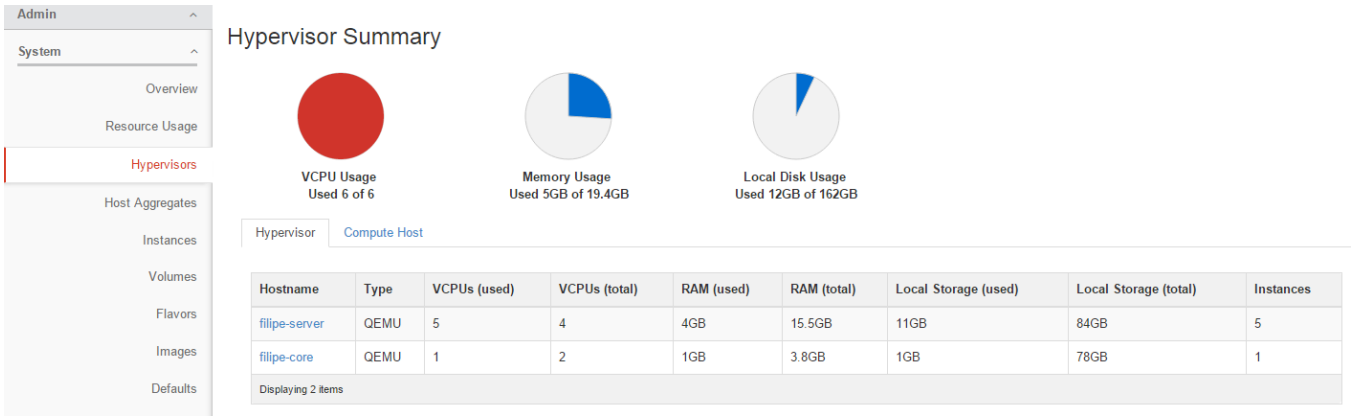


Figura 51 - Visualização dos recursos utilizados em cada node

Fonte: Fonte própria

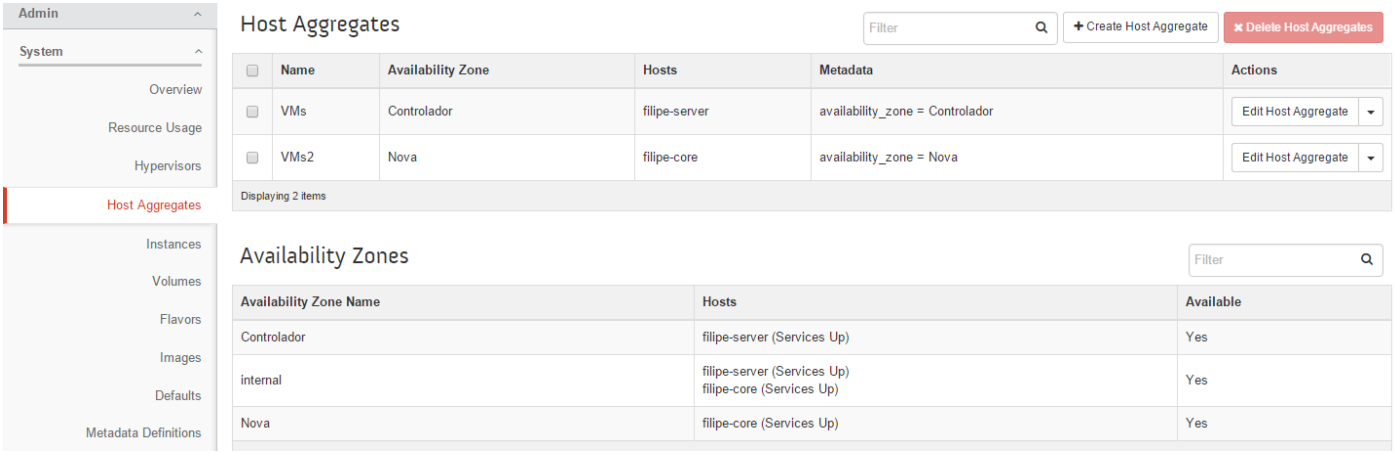


Figura 52 - Atribuição de hosts a zonas e visualização do estado destes

Fonte: Fonte própria

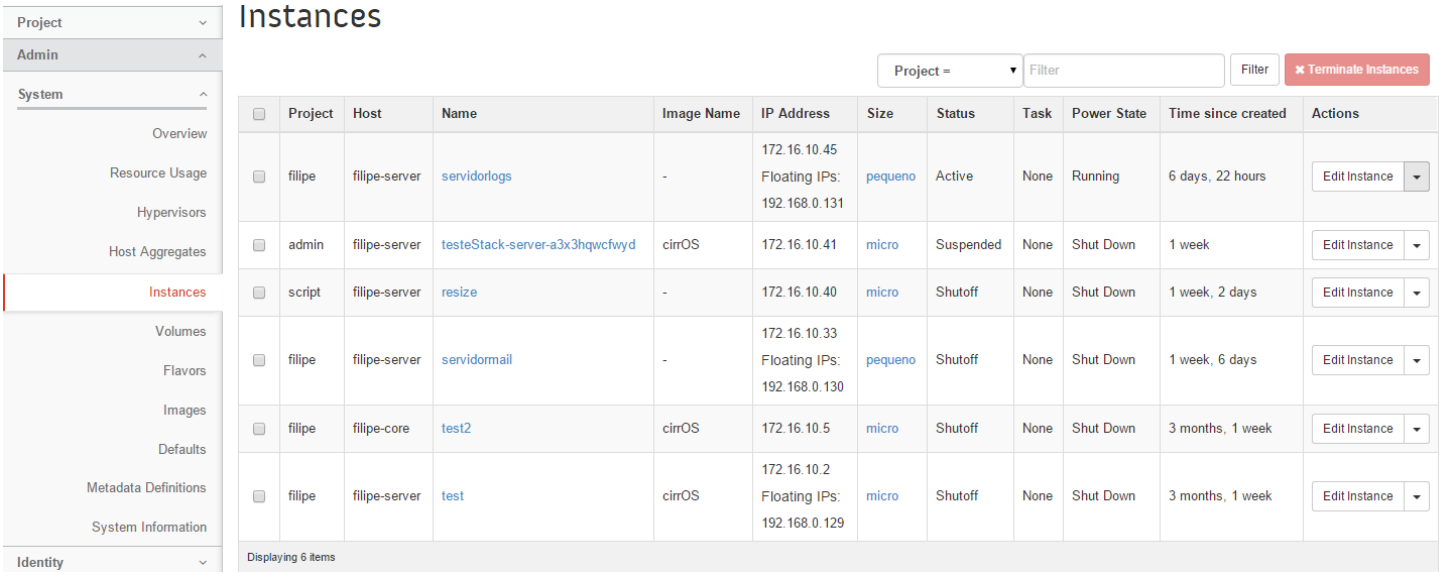


Figura 53- Visualização das máquinas virtuais do ambiente

Fonte: Fonte própria

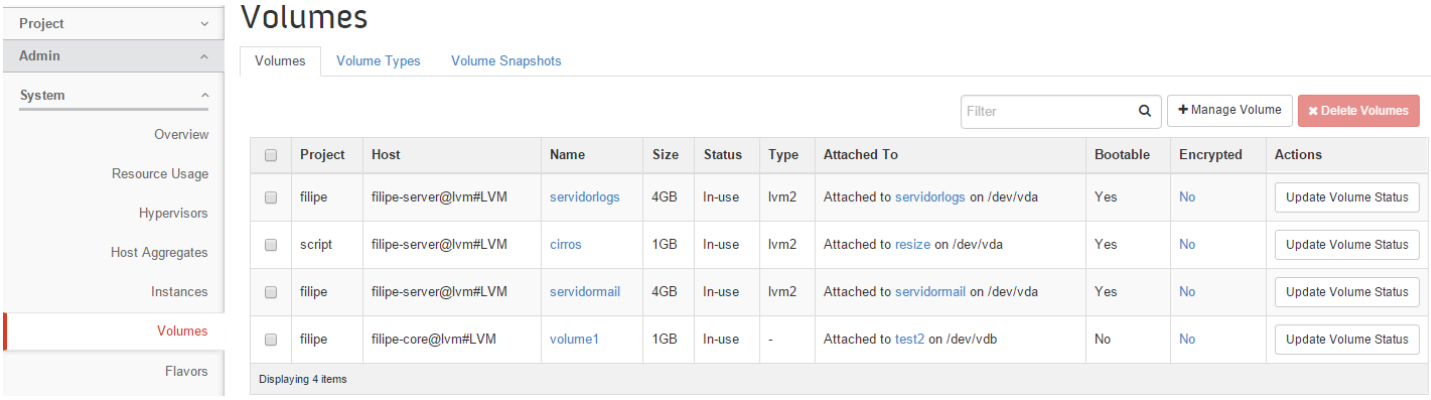


Figura 54 - Visualização dos volumes do ambiente

Fonte: Fonte própria

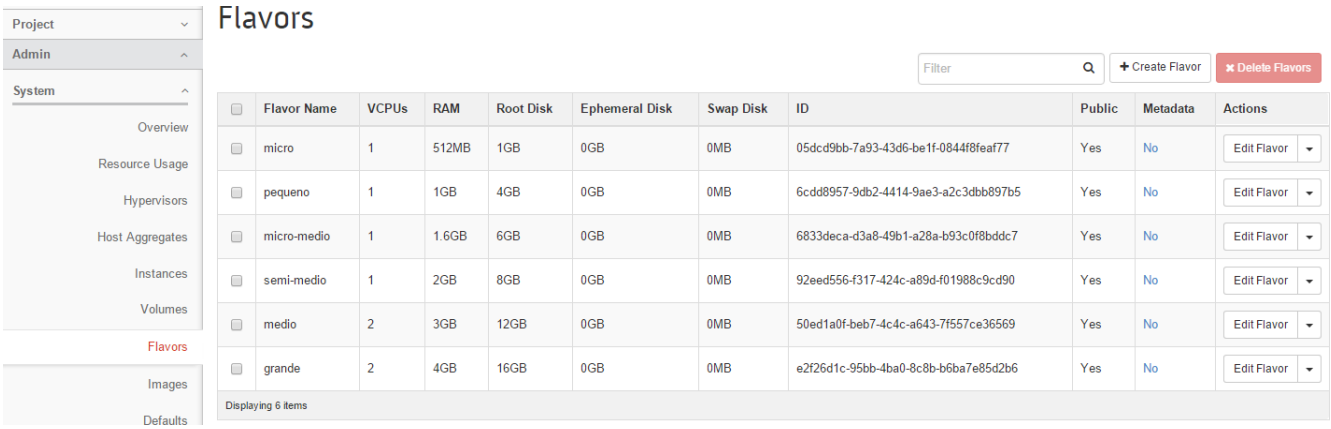


Figura 55 - Visualização e criação de modelos pré definidos de recursos computacionais para as máquinas virtuais

Fonte: Fonte própria

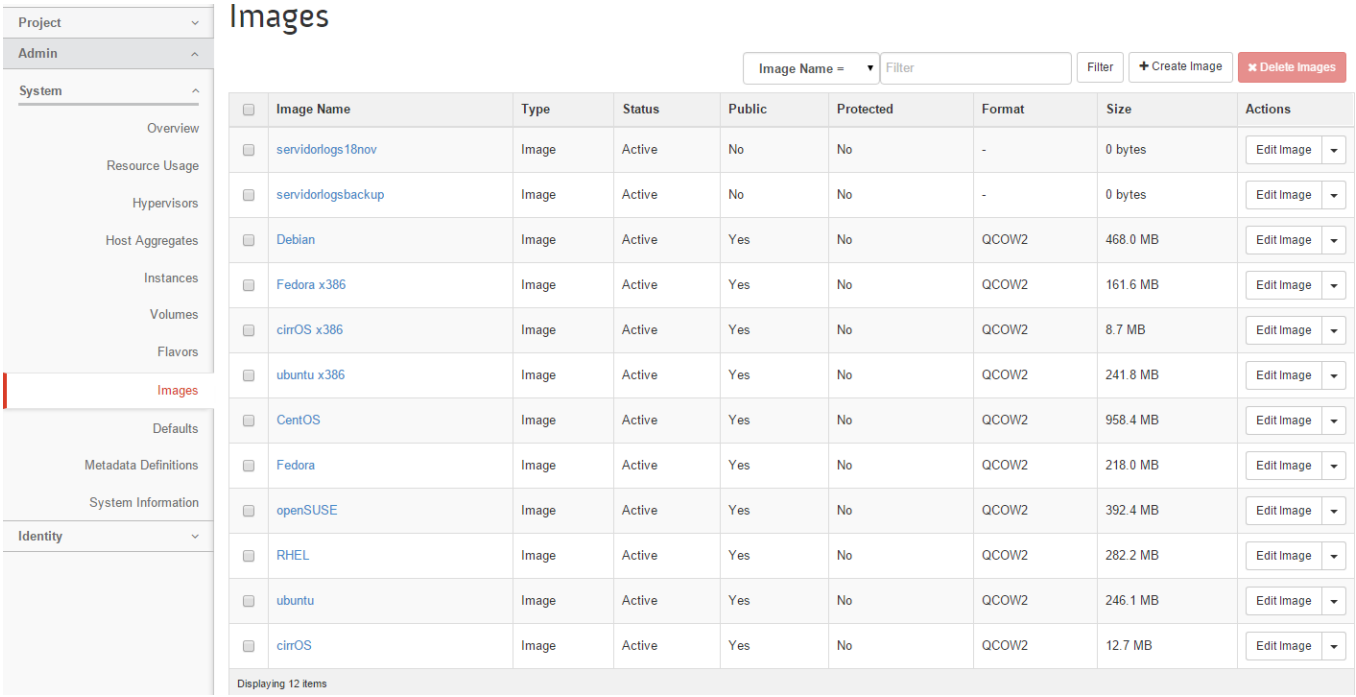


Figura 56 - Visualização e criação de imagens de sistemas operativos para as VMs

Fonte: Fonte própria

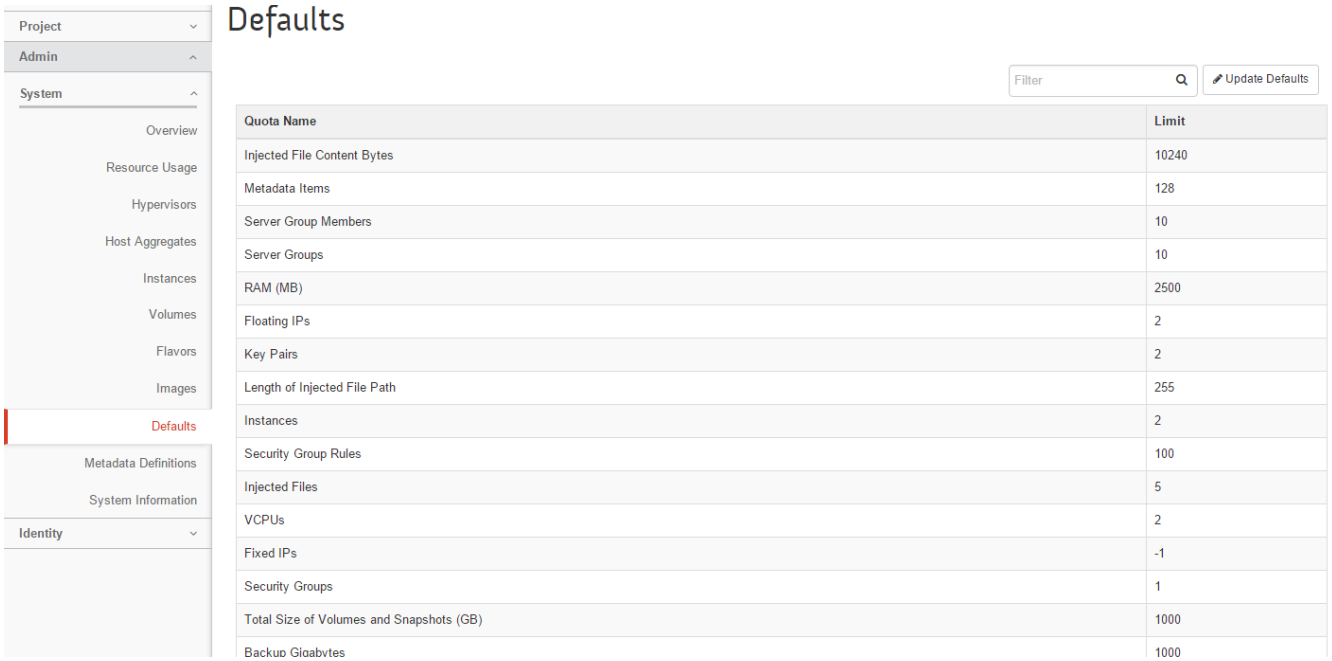


Figura 57 - Valores default de recursos e serviços a atribuir a novos projetos

Fonte: Fonte própria

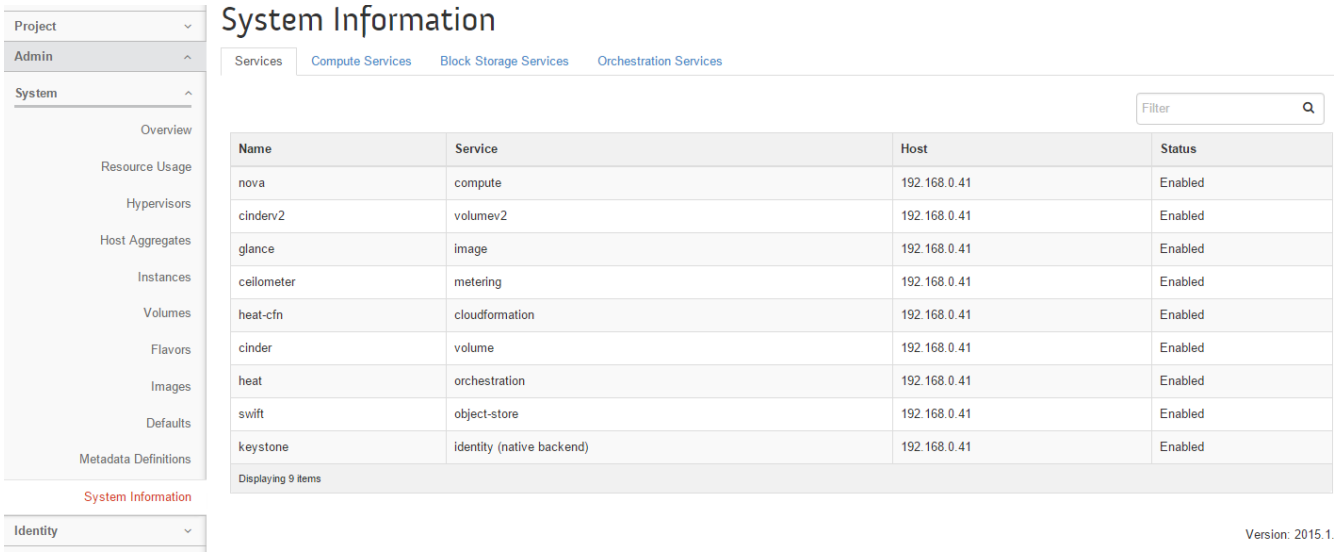


Figura 58 - Visualização do estado dos serviços OpenStack

Fonte: Fonte própria

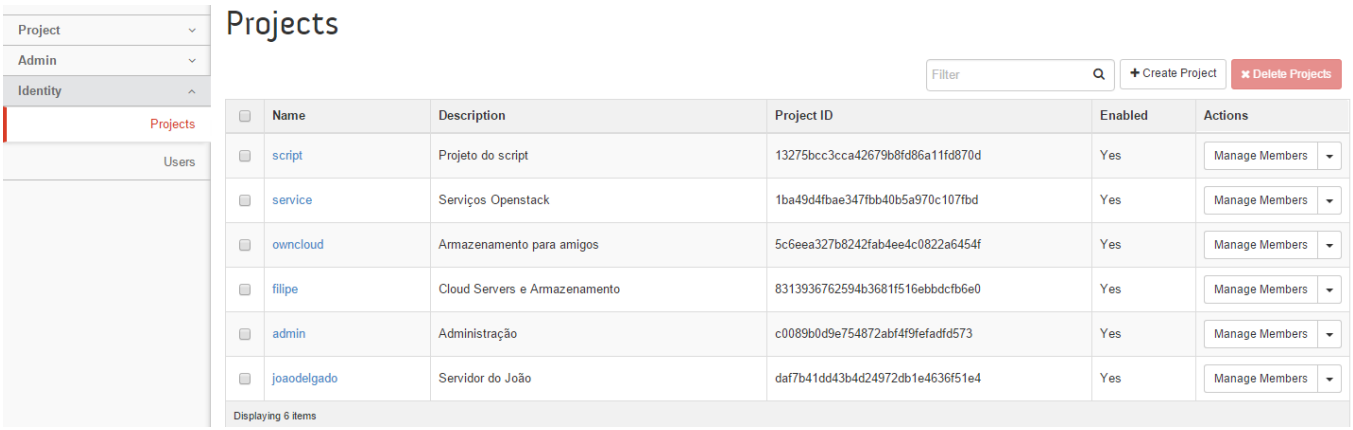


Figura 59 - Visualização e criação de projetos

Fonte: Fonte própria

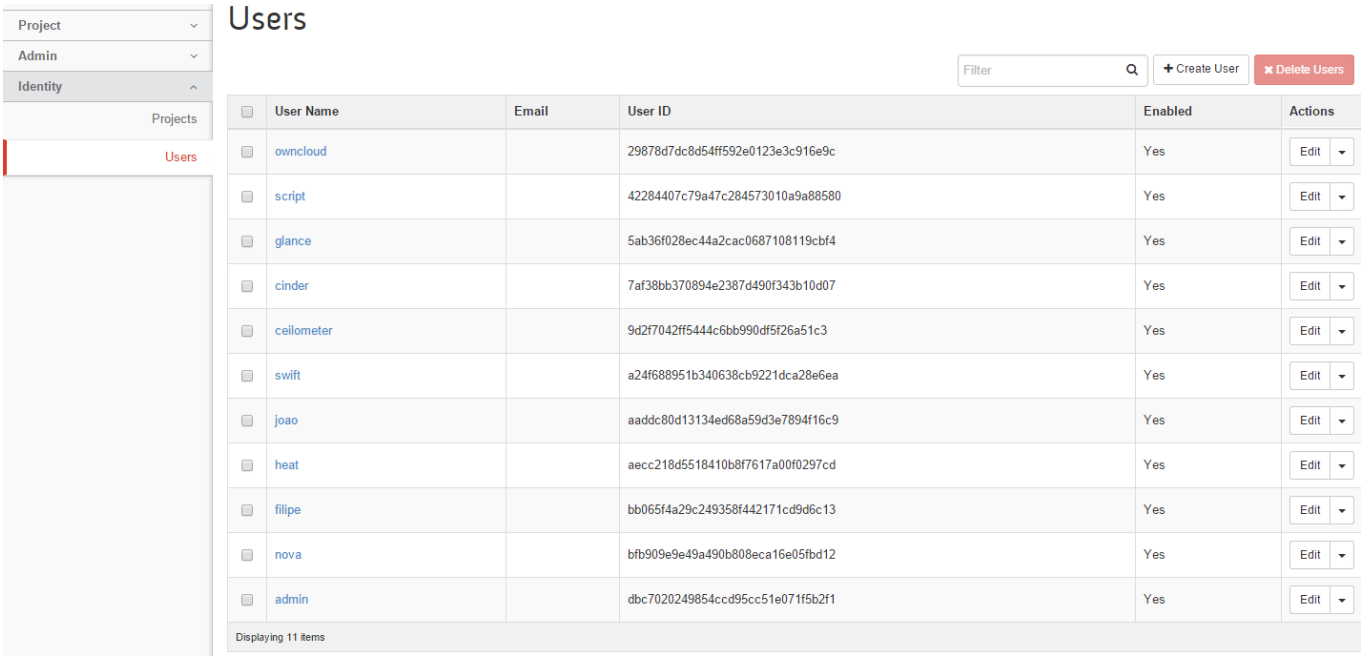
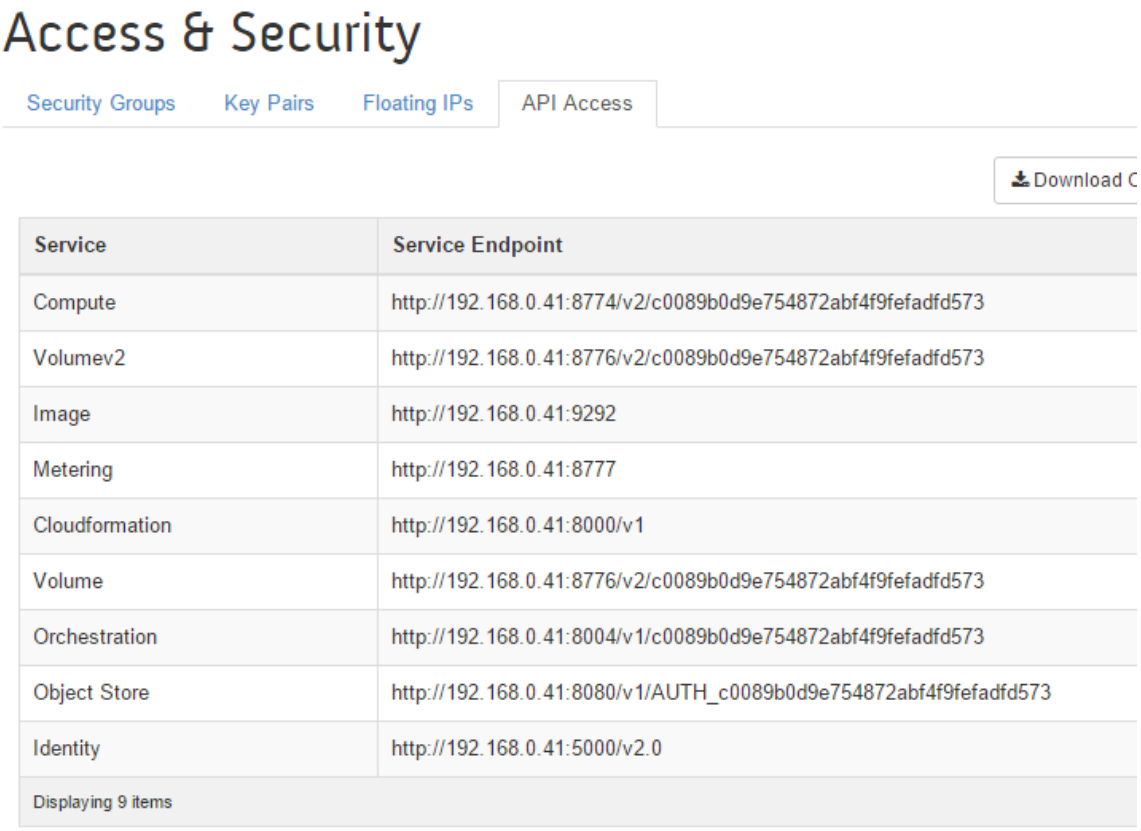


Figura 60 - Visualização e criação de utilizadores

Fonte: Fonte própria

Interface de utilizador



Service	Service Endpoint
Compute	http://192.168.0.41:8774/v2/c0089b0d9e754872abf4f9fefadfd573
Volumev2	http://192.168.0.41:8776/v2/c0089b0d9e754872abf4f9fefadfd573
Image	http://192.168.0.41:9292
Metering	http://192.168.0.41:8777
Cloudformation	http://192.168.0.41:8000/v1
Volume	http://192.168.0.41:8776/v2/c0089b0d9e754872abf4f9fefadfd573
Orchestration	http://192.168.0.41:8004/v1/c0089b0d9e754872abf4f9fefadfd573
Object Store	http://192.168.0.41:8080/v1/AUTH_c0089b0d9e754872abf4f9fefadfd573
Identity	http://192.168.0.41:5000/v2.0
Displaying 9 items	

Figura 61 - Visualização dos endpoints

Fonte: Fonte própria

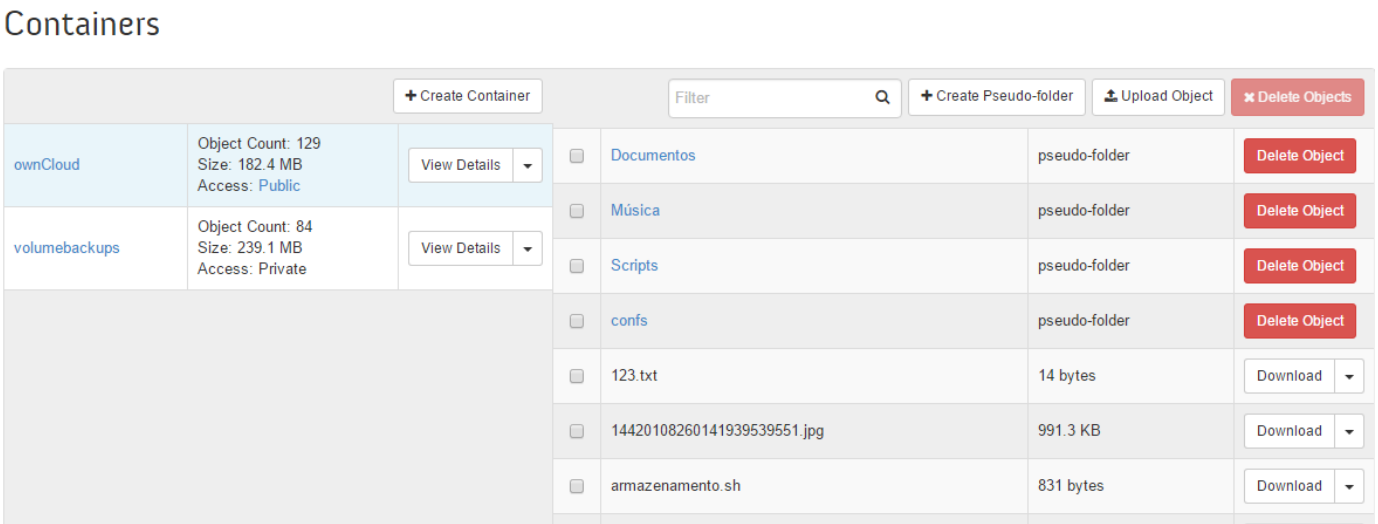


Figura 62 - Visualização e armazenamento de ficheiros

Fonte: Fonte própria

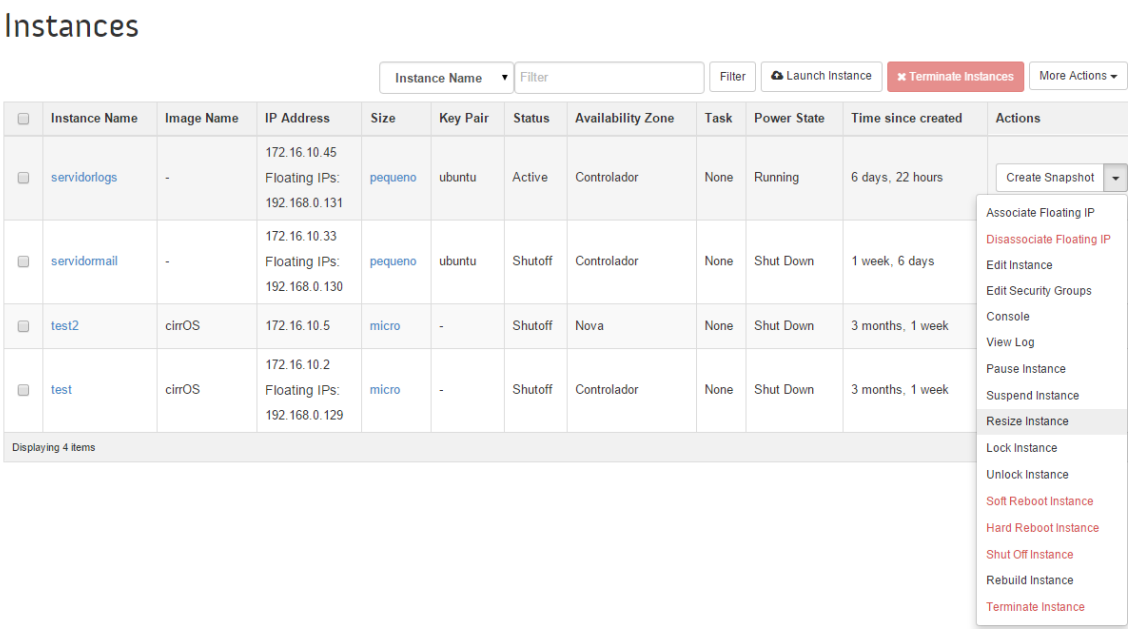


Figura 64 - Aumentar recursos do Servidor de Logs

Fonte: Fonte própria

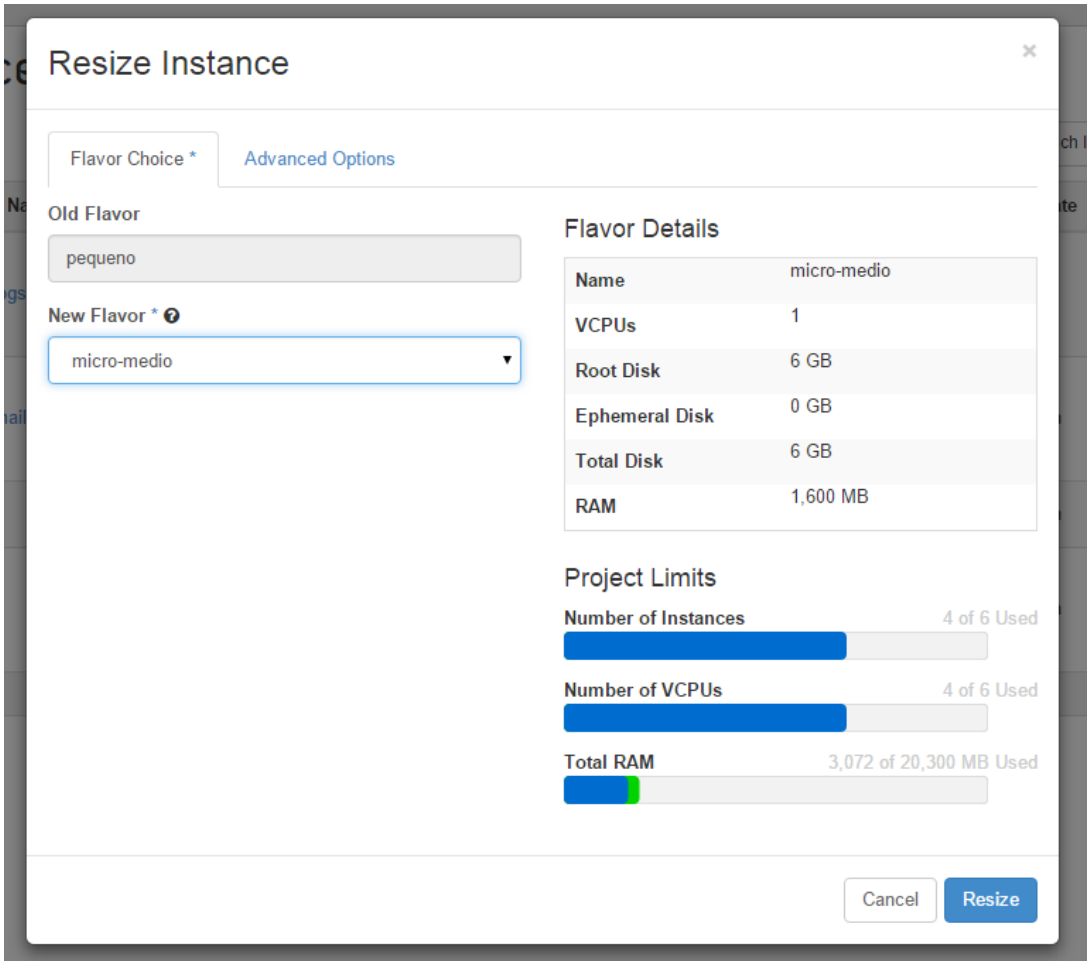


Figura 63 - Aumentar de 1GB de RAM para 1.6 GB

Fonte: Fonte própria

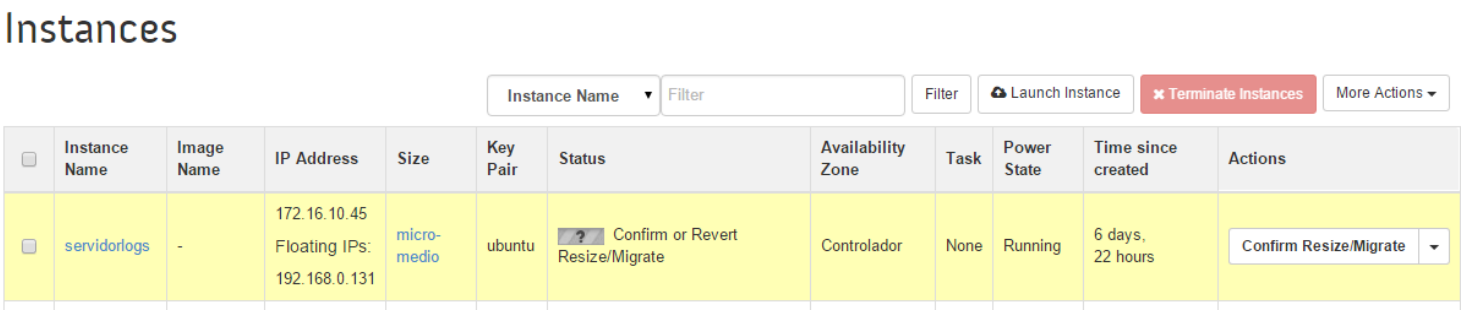


Figura 65 - Confirmar o aumento de recursos

Fonte: Fonte própria

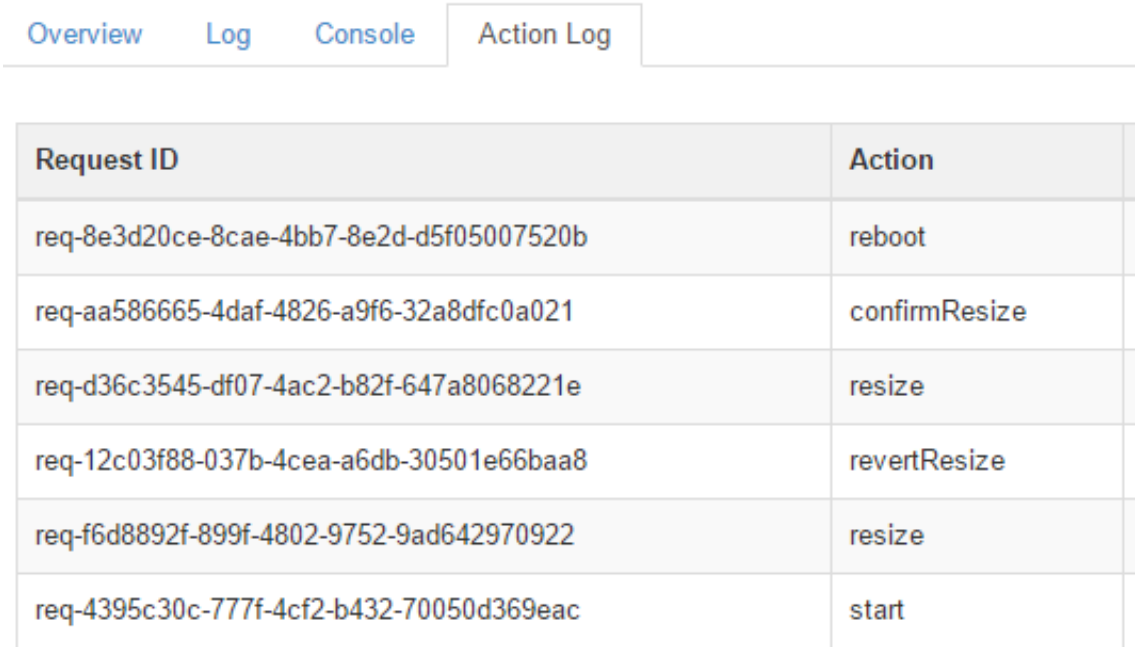


Figura 66 - Log dos estados da máquina virtual

Fonte: Fonte própria

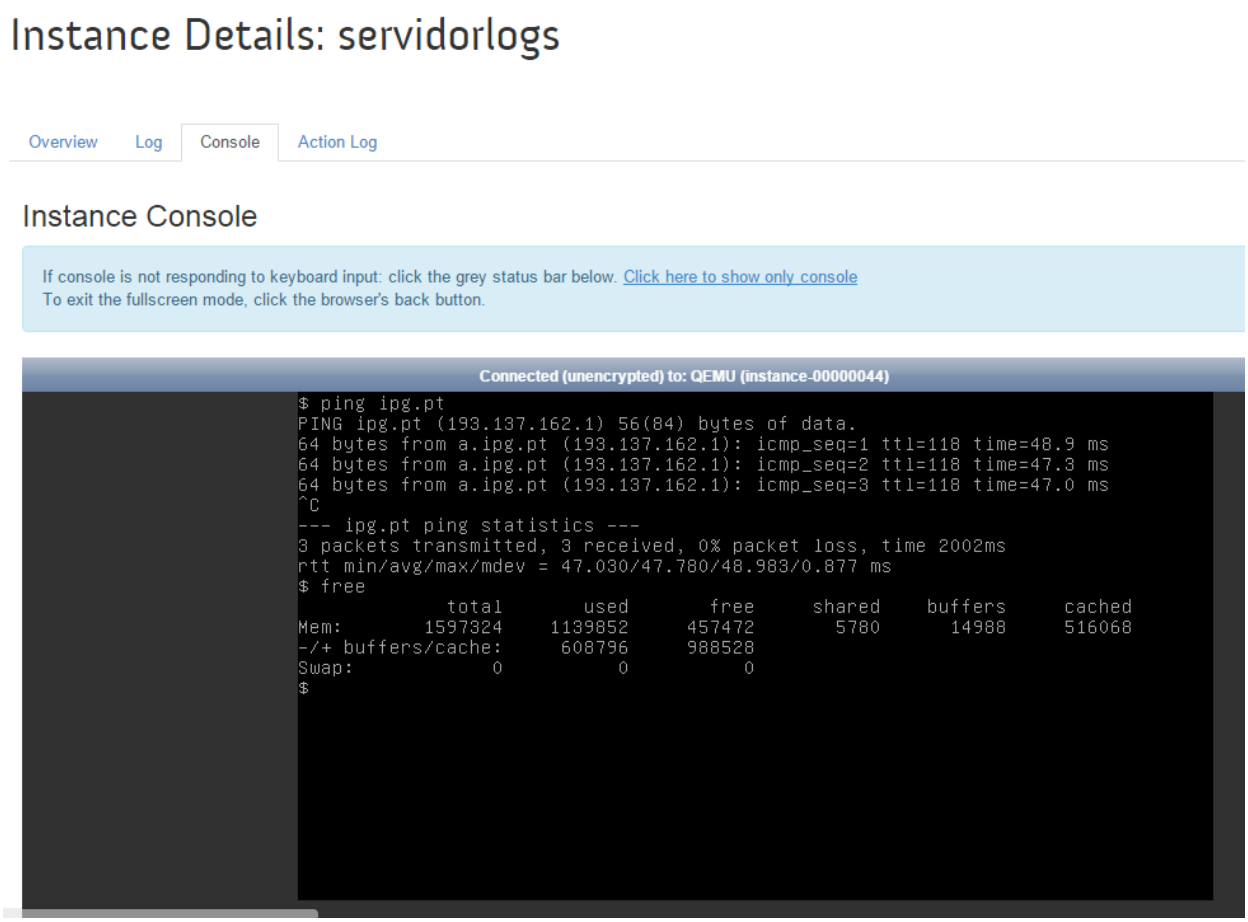


Figura 67 - Verificação do aumento da RAM no Servidor de Logs via VNC

Fonte: Fonte própria

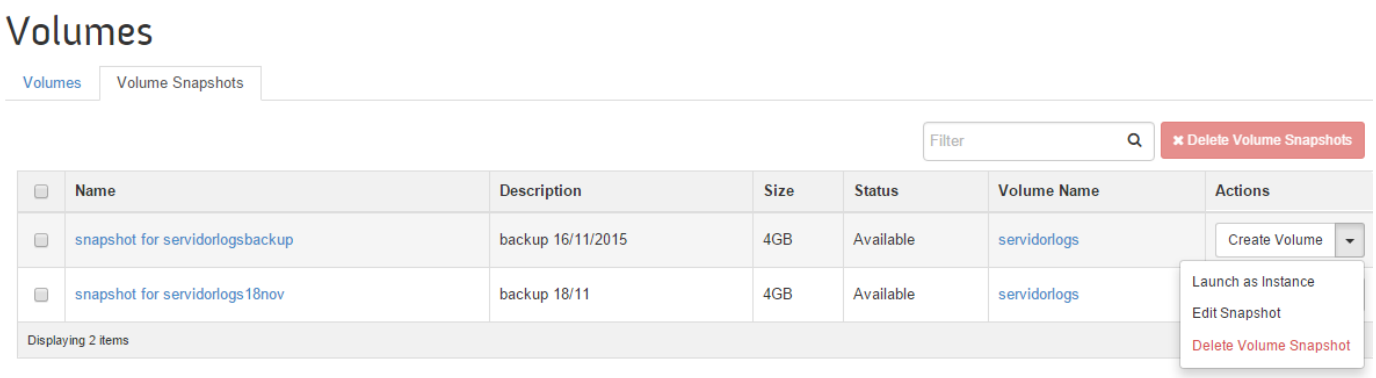


Figura 68 - Visualização e criação de snapshots dos volumes de dados

Fonte: Fonte própria

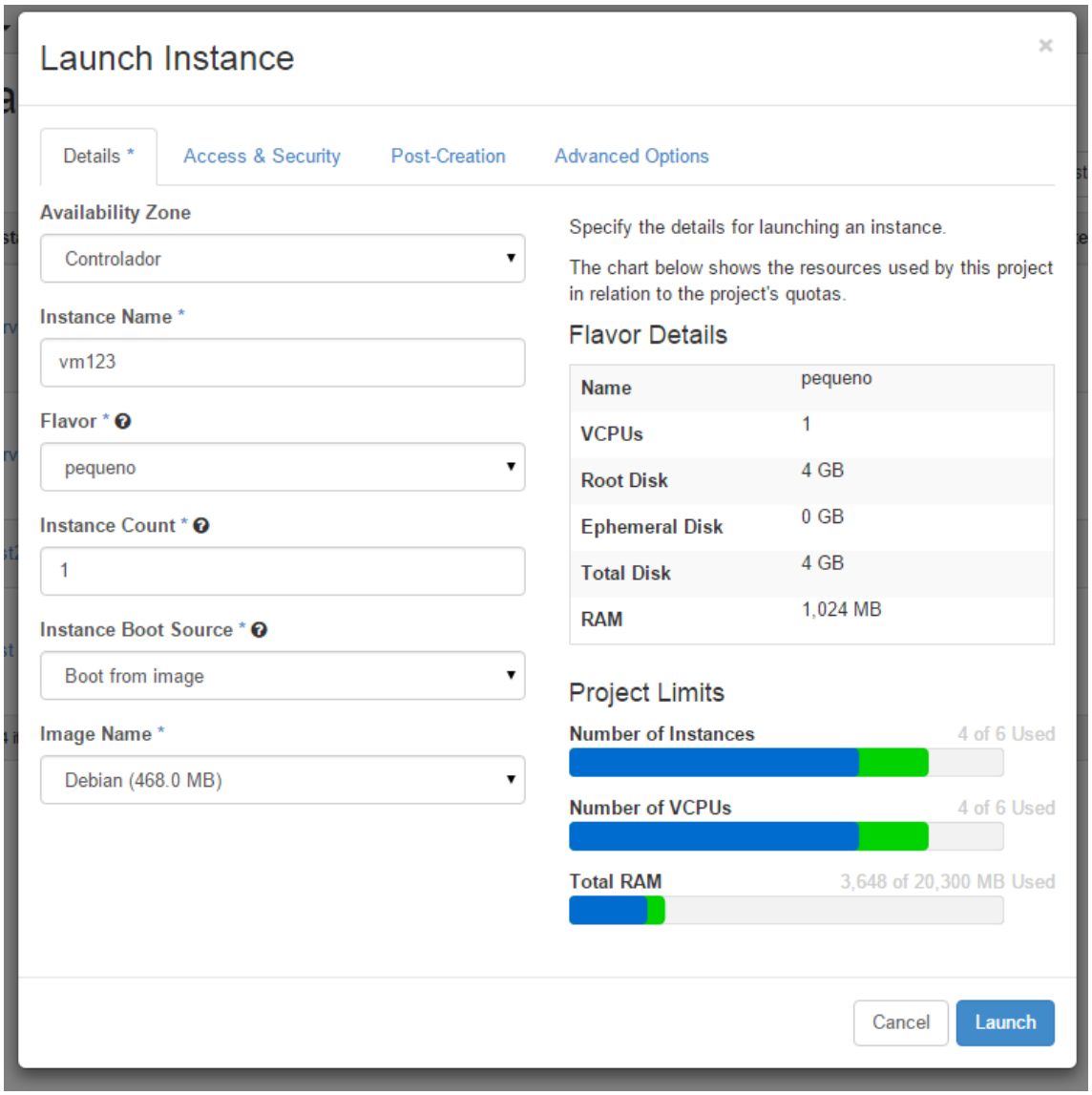


Figura 69 - Criação de máquina virtual

Fonte: Fonte própria