

DANIEL LUÍS SILVA MARTINS
IMPLEMENTAÇÃO DE *CLOUD* PRIVADA
MESTRADO DE ENGENHARIA INFORMÁTICA
DISSERTAÇÃO/ESTÁGIO
RELATÓRIO FINAL

ORIENTADORES
MÁRIO BERNARDES | FERNANDO BOAVIDA
30 DE JUNHO DE 2014



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Daniel Luís Silva Martins

WEBSITE:

<http://student.dei.uc.pt/~dmart>

E-MAIL:

dmart@student.dei.uc.pt

RESUMO

A computação em *cloud* tem vindo a assumir, nos últimos anos, uma importância crescente no panorama das tecnologias da informação no interior das organizações.

Com o objectivo de fornecer elasticidade e aprovisionamento dinâmico, por forma a proporcionar uma maior e melhor racionalização de recursos computacionais, obtendo assim ganhos ao nível económico e operacional, a *cloud* é, actualmente, um tema incontornável.

Neste trabalho, pretende-se estudar a viabilidade da adopção deste paradigma computacional, por parte do *Serviço de Gestão de Sistemas e Infra-estruturas de Informação e Comunicação (GSIIC)*.

Numa primeira fase, será feito um levantamento das soluções existentes (com a identificação das respectivas vantagens e desvantagens), um estudo comparativo entre as mesmas (tendo como principal critério de avaliação a capacidade de integração com a infra-estrutura e com os serviços actualmente em utilização no *GSIIC*), bem como um plano de implementação da solução escolhida.

Numa segunda fase, proceder-se-á à implementação de um sistema experimental, que se pretende que possibilite uma melhor avaliação do potencial e das limitações deste tipo de sistemas.

ABSTRACT

Cloud computing has assumed, in the last couple of years, a rising importance on the information technology landscape inside organizations.

With the purpose of supplying elasticity and dynamic provisioning, to achieve greater and better computing resources rationalization, leading to operational and financial gains, the cloud is, at the moment, an unavoidable subject.

In this work, we intend to study the adoption viability of this paradigm by the *University of Coimbra Information and Communication System and Infrastructure Administration Service (GSIIC)*.

In a first phase, we will perform a survey of existing solutions (identifying their respective advantages and disadvantages), a comparative study between them (using the integration capability with the in-use *GSIIC* systems as the primary criterion), as well as an implementation plan of the chosen solution.

At a second stage, we will proceed with an experimental system deployment, with the goal of better assessing the potential and limitations of this kind of systems.

AGRADECIMENTOS

A toda a equipa do *GSIIIC*. Em particular ao Prof. Mário Bernardes, ao Prof. Fernando Boavida e ao Eng. Alexandre Santos, pelo apoio, motivação e orientação.

Aos meus pais, irmã e restante família (demasiado numerosa para particularizar), por vinte e três anos de apoio constante.

À Ana, por nunca me ter deixado desistir, e à D. Goretti por me tratar como a um filho.

ÍNDICE

1	INTRODUÇÃO	1
1.1	Enquadramento	1
1.2	Objectivos	1
1.3	Organização do trabalho	2
1.3.1	Contexto	2
1.3.2	Calendarização	2
1.4	Estrutura do relatório	3
2	SISTEMAS DE OPERAÇÃO EM CLOUD	5
2.1	Conceitos	5
2.2	Características de um sistema de operação em <i>cloud</i>	8
2.3	Soluções comerciais	9
2.3.1	<i>VMware vCloud Suite</i>	10
2.3.2	<i>fluidOps eCloudManager</i>	10
2.3.3	<i>Flexiant Cloud Orchestrator</i>	11
2.3.4	<i>OnApp Cloud</i>	11
2.3.5	<i>Virtustream xStream</i>	11
2.3.6	<i>CA AppLogic</i>	11
2.4	Soluções <i>open-source</i>	12
2.4.1	<i>OpenNebula</i>	12
2.4.2	<i>OpenStack</i>	12
2.4.3	<i>Apache CloudStack</i>	13
2.4.4	<i>Eucalyptus</i>	14
2.5	Comparação de soluções	14
2.5.1	Informação Geral	14
2.5.2	Compatibilidade com <i>Application Programming Interfaces (APIs)</i>	15
2.5.3	<i>Hypervisors</i> Suportados	16
2.5.4	Rede	16
2.5.5	Armazenamento	16
2.6	Solução a adoptar	18
3	ESTUDO DO <i>openstack</i>	21
3.1	Descrição do <i>OpenStack</i>	21
3.2	Projectos <i>OpenStack</i>	21
3.2.1	<i>Horizon</i>	23
3.2.2	<i>Keystone</i>	23
3.2.3	<i>Nova</i>	24
3.2.4	<i>Glance</i>	26
3.2.5	<i>Swift</i>	26
3.2.6	<i>Cinder</i>	27
3.2.7	<i>Neutron</i>	29
3.2.8	<i>Heat</i>	29
3.2.9	<i>Ceilometer</i>	29
3.3	Topologia de rede	29
3.3.1	<i>FlatManager</i>	30
3.3.2	<i>FlatDHCPManager</i>	30
3.3.3	<i>VlanManager</i>	32

3.3.4	<i>Floating IPs</i>	32
3.4	Instalação de um sistema de demonstração	33
3.4.1	Instalação	33
3.4.2	Ensaaios	34
3.5	Avaliação de <i>hypervisors</i>	34
3.5.1	O que é um <i>hypervisor</i>	34
3.5.2	KVM	34
3.5.3	XEN	35
3.5.4	VMware ESXi	36
4	IMPLEMENTAÇÃO DE CLOUD PRIVADA COM OPENSTACK	39
4.1	Planeamento	39
4.1.1	Sistemas de Aprovisionamento e <i>Deployment</i>	39
4.1.2	<i>Hardware</i>	41
4.1.3	Organização dos serviços <i>OpenStack</i>	42
4.1.4	Base de dados	42
4.1.5	<i>Messaging Queue</i>	43
4.1.6	<i>Load Balancer</i>	44
4.1.7	<i>Backend(s)</i> de armazenamento	46
4.1.8	Autenticação e Autorização	48
4.1.9	<i>Caching</i>	49
4.1.10	Arquitectura de rede	50
4.2	Descrição do ambiente a instalar	51
4.3	Instalação	51
4.3.1	Preparação da estrutura de rede	53
4.3.2	Aprovisionamento do <i>hardware</i>	53
4.3.3	<i>HAProxy & keepalived</i>	53
4.3.4	<i>Percona XtraDB Cluster</i>	54
4.3.5	<i>RabbitMQ</i>	54
4.3.6	<i>memcached</i>	54
4.3.7	<i>Ceph</i>	54
4.3.8	Serviços de gestão <i>OpenStack</i>	55
4.3.9	<i>nova-compute & nova-network</i>	56
4.3.10	<i>Horizon</i>	56
4.4	Integração com serviços existentes	56
4.4.1	Serviço de directório	56
4.4.2	Serviço de monitorização	58
5	VALIDAÇÃO E ANÁLISE DO COMPORTAMENTO DO SISTEMA	59
5.1	Validação do <i>deployment</i>	59
5.2	Comportamento em situações de falha	60
5.2.1	<i>Keepalived</i>	61
5.2.2	<i>HAProxy</i>	61
5.2.3	<i>Percona XtraDB Cluster</i>	63
5.2.4	<i>RabbitMQ</i>	65
5.2.5	<i>memcached</i>	67
5.2.6	<i>Ceph</i>	67
5.2.7	Serviços de gestão <i>OpenStack</i>	69
5.2.8	<i>nova-compute</i>	70
5.2.9	<i>nova-network</i>	71
5.3	Recuperação de situações de falha	71
5.3.1	<i>HAProxy</i>	72
5.3.2	<i>Percona XtraDB Cluster</i>	72
5.3.3	<i>RabbitMQ</i>	72

5.3.4	<i>memcached</i>	73
5.3.5	<i>Ceph</i>	74
5.3.6	Serviços de gestão <i>OpenStack</i>	74
5.3.7	<i>nova-compute</i>	74
5.3.8	<i>nova-network</i>	75
5.4	Considerações sobre a escalabilidade	75
5.4.1	<i>Ceph</i>	75
5.4.2	<i>OpenStack</i>	75
5.5	Actualização do <i>OpenStack</i>	76
6	CONCLUSÃO	77
6.1	Trabalho Futuro	77
	BIBLIOGRAFIA	79
	Anexos	
A	INSTALAÇÃO DE SISTEMA EXPERIMENTAL	
B	CONFIGURAÇÃO DO <i>switch</i>	
C	INSTALAÇÃO DO <i>dnsmasq</i>	
D	CONFIGURAÇÃO DO <i>nat</i>	
E	INSTALAÇÃO DO <i>openvpn</i>	
F	INSTALAÇÃO DO <i>ntpd</i>	
G	INSTALAÇÃO DO <i>puppet</i> E <i>foreman</i>	
H	INSTALAÇÃO DO <i>percona xtradb cluster</i>	
I	INSTALAÇÃO DO <i>haproxy</i> E DO <i>keepalived</i>	
J	INSTALAÇÃO DO <i>rabbitmq</i>	
K	INSTALAÇÃO DO <i>memcached</i>	
L	INSTALAÇÃO DO <i>ceph</i>	
M	INSTALAÇÃO DO <i>keystone</i>	
N	INSTALAÇÃO DO <i>glance</i>	
O	INSTALAÇÃO DO <i>cinder</i>	
P	INSTALAÇÃO DO <i>nova controller</i>	
Q	INSTALAÇÃO DO <i>horizon</i>	
R	INSTALAÇÃO DO <i>kvm</i>	
S	INSTALAÇÃO DO <i>nova-compute</i> E <i>nova-network</i>	
T	INSTALAÇÃO DO <i>openldap</i>	
U	INSTALAÇÃO E CONFIGURAÇÃO DO <i>zabbix</i>	

LISTA DE FIGURAS

Figura 1	Calendarização do projecto.	3
Figura 2	Os três modelos de serviço de computação em <i>cloud</i>	6
Figura 3	Arquitectura conceptual de um sistema de operação em <i>cloud</i>	9
Figura 4	Arquitectura conceptual do <i>OpenStack</i> . (Imagem sob a licença Apache 2.0 extraída de [16])	22
Figura 5	Arquitectura conceptual do <i>Keystone</i>	23
Figura 6	Arquitectura conceptual do <i>Nova</i>	25
Figura 7	Arquitectura conceptual do <i>Glance</i>	27
Figura 8	Relação entre <i>account</i> , <i>container</i> e objecto	28
Figura 9	Arquitectura lógica do <i>Swift</i>	28
Figura 10	Arquitectura conceptual do <i>Cinder</i>	28
Figura 11	<i>nova-network</i> - <i>FlatManager</i>	30
Figura 12	<i>nova-network</i> - <i>FlatDHCPManager</i> (<i>single-host</i>)	31
Figura 13	<i>nova-network</i> - <i>FlatDHCPManager</i> (<i>multi-host</i>)	31
Figura 14	<i>nova-network</i> - <i>VlanManager</i>	32
Figura 15	Arquitectura física do sistema de demonstração.	33
Figura 16	Arquitectura conceptual de um <i>hypervisor</i>	35
Figura 17	<i>KVM</i> - Caso particular de um <i>hypervisor</i> do tipo 1.	35
Figura 18	Integração de <i>KVM</i> com <i>OpenStack</i>	35
Figura 19	Integração de <i>XEN</i> com <i>OpenStack</i>	35
Figura 20	Integração de <i>ESXi</i> com <i>OpenStack</i> (modo <i>stand-alone</i>).	36
Figura 21	Integração de <i>ESXi</i> com <i>OpenStack</i> (modo <i>clustered</i>).	37
Figura 22	Arquitectura conceptual do <i>Percona XtraDB Cluster</i>	43
Figura 23	<i>RabbitMQ cluster</i> com <i>mirrored queues</i> - Publicação de uma mensagem.	44
Figura 24	Arquitectura conceptual do <i>HAProxy</i> e <i>Keepalived</i>	46
Figura 25	<i>CPEH</i> - Arquitectura conceptual.	47
Figura 26	<i>CPEH</i> - <i>Placement Groups</i> (com factor de replicação 3) e algoritmo <i>CRUSH</i>	48
Figura 27	<i>memcached</i> - Fluxograma.	49
Figura 28	Diagrama da arquitectura de rede.	51
Figura 29	Visão geral da arquitectura física a implementar.	52
Figura 30	<i>HAProxy</i> - Comportamento do <i>cluster</i> durante falha do nó <i>MASTER</i>	62
Figura 31	<i>Percona XtraDB Cluster</i> - Comportamento do <i>cluster</i>	64
Figura 32	<i>RabbitMQ</i> - Comportamento do <i>cluster</i>	66
Figura 33	<i>memcached</i> - Comportamento após falha de um nó.	67
Figura 34	<i>Keystone</i> - Comportamento após falha de um nó.	70

LISTA DE TABELAS

Tabela 1	Matriz Serviço/Implementação de exemplos de soluções <i>cloud</i>	7
Tabela 2	Comparação de soluções <i>Open-Source</i> - Informação geral	15
Tabela 3	Comparação de soluções <i>Open-Source</i> - Compatibilidade com <i>APIs</i>	15
Tabela 4	Comparação de soluções <i>Open-Source</i> - <i>Hypervisors</i> suportados	16
Tabela 5	Comparação de soluções <i>Open-Source</i> - Funcionalidades de rede suportadas	17
Tabela 6	Comparação de soluções <i>Open-Source</i> - Serviços de armazenamento disponibilizados	18
Tabela 7	Comparação de soluções <i>Open-Source</i> - Vantagens e desvantagens	19
Tabela 8	<i>HAProxy</i> - Comportamento em situação de falha. . .	62
Tabela 9	<i>Percona XtraDB Cluster</i> - Comportamento em situação de falha.	63
Tabela 10	<i>RabbitMQ</i> - Comportamento em situação de falha. . .	65
Tabela 11	<i>Ceph</i> - Comportamento do <i>cluster MON</i> em situação de falha.	68
Tabela 12	<i>Ceph</i> - Comportamento do <i>cluster MDS</i> em situação de falha.	69
Tabela 13	<i>Ceph</i> - Comportamento do <i>cluster OSD</i> em situação de falha.	69
Tabela 14	<i>Percona XtraDB Cluster</i> - Recuperação de falhas. . . .	72
Tabela 15	<i>RabbitMQ</i> - Recuperação de falhas.	73
Tabela 16	<i>Ceph</i> - Recuperação de falhas no <i>cluster MON</i>	74

LISTA DE ACRÓNIMOS

AOE	ATA over Ethernet
API	Application Programming Interface
AMQP	Advanced Message Queuing Protocol
ARP	Address Resolution Protocol
CDN	Content Distribution Network
CERN	Conseil Européen pour la Recherche Nucléaire
CIFS	Common Internet File System
CLI	Command-Line Interface
DHCP	Dynamic Host Configuration Protocol

DNS Domain Name System

EBS Elastic Block Store

EC2 Elastic Compute Cloud

ENC External Node Classifier

EPEL Extra Packages for Enterprise Linux

ESA European Space Agency

FCP Fiber Channel Protocol

FP7 European Union Seventh Framework Programme

GPL GNU Public Licence

GSII Gestão de Sistemas e Infra-estruturas de Informação e Comunicação

HA High Availability

HTTP Hypertext Transfer Protocol

IAAS Infrastructure-as-a-Service

IP Internet Protocol

IRC Internet Relay Chat

iSCSI Internet Small Computer System Interface

JSON JavaScript Object Notation

KVS Key-Value Store

LAAS Landscape-as-a-Service

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LVM Logical Volume Manager

MAAS Metal-as-a-Service

MOM Message Oriented Middleware

NAS Network Attached Storage

NASA United States National Aeronautics and Space Administration

NAT Network Address Translation

NFS Network File System

NTP Network Time Protocol

occi Open Cloud Computing Interface

OGF Open Grid Forum

OSI Open Source Initiative

PAAS Platform-as-a-Service

PKI Public Key Infrastructure
PXE Preboot eXecution Environment
RAID Redundant Array of Inexpensive Disks
RAM Random Access Memory
REST Representational State Transfer
S3 Simple Storage Service
SAAS Software-as-a-Service
SAN Storage Area Network
SDN Software-Defined Networking
SMB Server Message Block
SNMP Simple Network Management Protocol
SO Sistema Operativo
SPICE Simple Protocol for Independent Computing Environments
SPOF Single Point of Failure
SQL Structured Query Language
SSH Secure Shell
TCP Transmission Control Protocol
TFTP Trivial File Transfer Protocol
UUID Universally Unique Identifier
VIP Virtual IP
VLAN Virtual LAN
VM Virtual Machine
VMFS Virtual Machine File System
VNC Virtual Network Computing
VNIC Virtual NIC
VPDC Virtual Private Data Center
VRRP Virtual Router Redundancy Protocol
WAN Wide Area Network
WMI Windows Management Instrumentation
WOL Wake-on-LAN
WSGI Web Server Gateway Interface

1 | INTRODUÇÃO

Este relatório visa documentar o trabalho realizado, no âmbito do projecto de estágio da cadeira de *Dissertação/Estágio do Mestrado em Engenharia Informática* do Departamento de Engenharia Informática da Universidade de Coimbra.

1.1 ENQUADRAMENTO

O papel cada vez mais central que as tecnologias da informação têm vindo a assumir no seio das organizações tem feito emergir a gestão das infra-estruturas informáticas como uma peça crítica do plano de operações, necessária para assegurar a continuidade da actividade.

Esta dependência, que cria uma constante pressão para aumentar a qualidade de serviço e agilizar processos, aliada a um clima económico instável que faz com que o orçamento de muitos departamentos informáticos seja reduzido, faz com que muitas organizações relembram para segundo plano investimentos em infra-estruturas e procurem soluções que optimizem os recursos existentes.

Anunciando capacidades de alta disponibilidade, tolerância a falhas, provisionamento automático de recursos e escalabilidade flexível que potenciam a redução de custos, devido a um melhor aproveitamento de recursos informáticos e humanos, os serviços de *cloud* e as *clouds* privadas ou híbridas são cada vez mais encaradas pelas organizações como possíveis soluções para estes problemas.

1.2 OBJECTIVOS

O objectivo deste projecto passa por estudar a viabilidade da adopção, por parte do GSIIC¹, de um sistema de operação em *cloud* que permita agilizar e tornar mais eficiente e eficaz o processo de administração da sua infra-estrutura informática.

Pretende-se obter familiaridade com este tipo de sistemas, podendo assim, de forma objectiva, analisar a sua adequação a uma organização como o GSIIC e às necessidades impostas pelas especificidades da sua natureza e dos serviços e departamentos que serve.

Para além desta visão global sobre os sistemas de operação em *cloud*, interessa principalmente obter conhecimento sobre uma solução em particular (que deverá ser *open-source*), que contribua para o esforço, que o GSIIC tem vindo a desenvolver, de migrar as suas ferramentas para soluções abertas e de utilização gratuita.

Utilizando esta solução, pretende-se ainda implementar uma infra-estrutura experimental, dotada de características de alta disponibilidade e de tolerância a falhas, que sirva de prova de conceito e, eventualmente, de plataforma inicial para essa migração.

¹ <http://www.uc.pt/ciuc>

1.3 ORGANIZAÇÃO DO TRABALHO

No primeiro semestre, o plano de trabalhos consistiu no estudo dos diversos modelos de *cloud*, tendo sido analisada diversa literatura subordinada à temática.

Fez-se um levantamento das soluções de operação em *cloud* disponíveis, bem como das suas características, por forma a escolher a mais adequada às necessidades do *GSIIC*.

Seguiu-se a implementação de um sistema experimental, que visava, logo numa fase inicial do projecto, identificar as principais potencialidades e eventuais limitações da solução escolhida que não tivessem sido evidenciadas durante a fase de análise.

De seguida foi feita uma avaliação da capacidade de integração, com alguns sistemas de virtualização com especial interesse para o *GSIIC* (devido à sua familiaridade com os mesmos, ou por possuírem características de relevo para a operação do seu *datacenter*).

No segundo semestre, começou-se por fazer o planeamento de um ambiente, com características muito semelhantes às que existiriam num ambiente de produção, utilizado como prova de conceito da viabilidade da adopção da solução escolhida por parte do *GSIIC*.

Seguiu-se a implementação do referido sistema e consequente análise das suas capacidades de escalabilidade e do seu comportamento em situações de stress e de falha.

1.3.1 Contexto

O *GSIIC* é o serviço da Administração da Universidade de Coimbra responsável pela área das tecnologias de informação. As suas principais actividades são desenvolvidas nos domínios da concepção e planeamento das aplicações informáticas, das comunicações, da infra-estrutura de rede, de servidores e de bases de dados.

Para além das tarefas de auditoria e suporte, uma das principais ocupações do *GSIIC* é a gestão e operação da infra-estrutura tecnológica da Universidade, sendo precisamente nesta vertente de operação que este projecto se desenvolve.

Uma parte considerável dessa infra-estrutura está localizada no *datacenter* composto por um número considerável de servidores e equipamentos de rede e de armazenamento, de diversos fabricantes, que o *GSIIC* possui nas suas próprias instalações.

1.3.2 Calendarização

As datas relevantes para o estabelecimento da calendarização para o projecto são:

16 DE SETEMBRO DE 2013 Início oficial do estágio

28 DE JANEIRO DE 2014 Entrega do relatório intermédio de estágio

5 DE FEVEREIRO DE 2014 Defesa pública da primeira parte do estágio

27 DE JUNHO DE 2014 Término oficial do estágio

1 DE JULHO DE 2014 Entrega do relatório final de estágio

10 DE JULHO DE 2014 Defesa pública do estágio

A calendarização, ilustrada na Figura 1, teve em linha de conta estas datas, tendo o trabalho sido organizado por forma a ser possível terminar os objectivos propostos atempadamente.

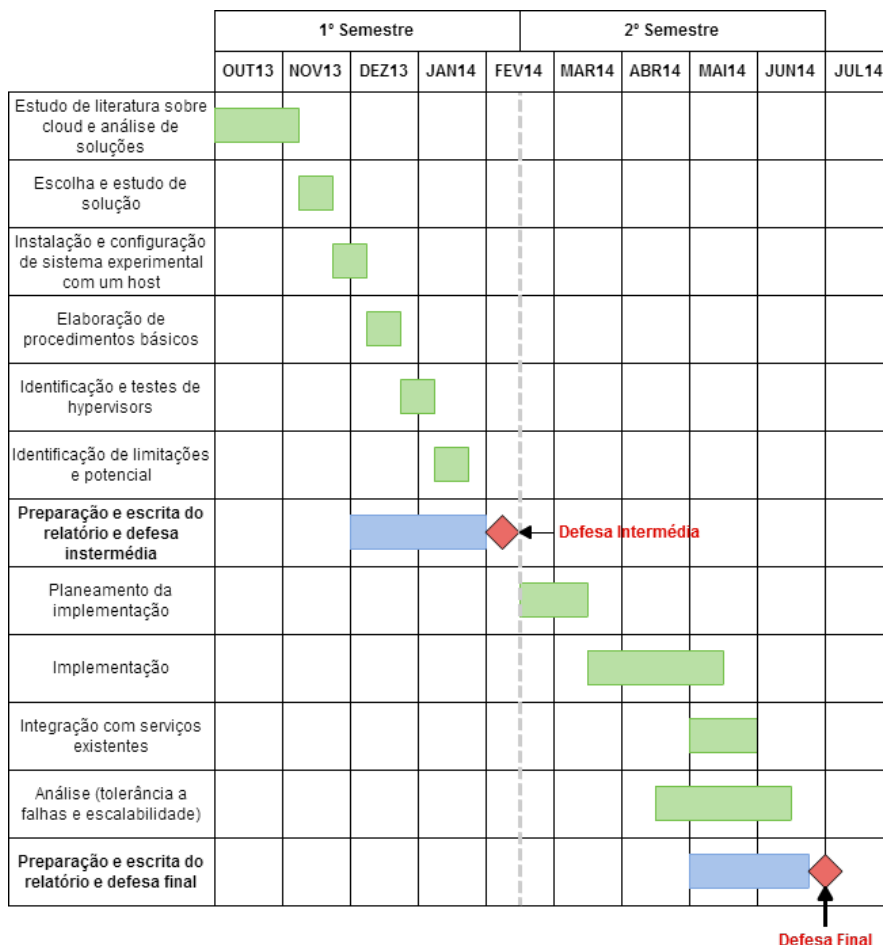


Figura 1: Calendarização do projecto.

1.4 ESTRUTURA DO RELATÓRIO

Neste Capítulo 1, fez-se uma apresentação do enquadramento do projecto que motivou este relatório, dos seus objectivos e do âmbito em que foi realizado. Explicaram-se as motivações do GSIIC para abordar os sistemas de operação em *cloud* e apresentou-se o planeamento do trabalho e respectiva calendarização.

No Capítulo 2, será feita uma introdução aos sistemas de operação em *cloud*. Serão abordados os principais conceitos subjacentes e far-se-á uma análise da proposta de valor deste tipo de sistemas relativamente aos sistemas de operação convencionais. Segue-se a identificação das principais soluções actualmente disponíveis (comerciais e *open-source*) e uma análise comparativa entre as mesmas. Por fim, é apresentada a solução a adoptar (*OpenStack*) neste projecto, bem como o racional utilizado na sua escolha.

O Capítulo 3 é dedicado à análise do *OpenStack*. É feita uma descrição detalhada do mesmo, apresentando-se a sua arquitectura conceptual e os seus diversos componentes e projectos constituintes. No final, é feito um relato da instalação de um sistema experimental e avaliação da capacidade de integração com diversos *hypervisors*².

No Capítulo 4, é feita uma descrição detalhada da prova de conceito a implementar e do seu planeamento. São apresentadas as decisões técnicas que foi necessário tomar e as razões subjacentes à sua escolha. Segue-se o relato da instalação do sistema e da integração com alguns serviços existentes no *GSII*C.

O Capítulo 5 descreve toda a fase de análise e avaliação do sistema. É descrita a validação funcional do sistema, a análise da escalabilidade, e os testes ao seu comportamento, em situações de falhas, e consequente recuperação.

Por fim, no Capítulo 6, são apresentadas as conclusões resultantes do trabalho realizado e são identificados possíveis desenvolvimentos.

² Um *hypervisor* é uma solução de virtualização baseada em *software*. Na Secção 3.5.1 é feita uma abordagem mais detalhada a este tipo de sistemas.

2

SISTEMAS DE OPERAÇÃO EM CLOUD

O termo *computação em cloud* é utilizado para descrever um vasto conjunto de conceitos e tecnologias em permanente evolução e bastante díspares entre si. Foi portanto necessário, primeiro de tudo, obter uma visão global destes conceitos e das suas relações para se conseguir perceber as suas vantagens e enquadrar as necessidades do GSIIC num dos segmentos do vasto panorama da computação em *cloud*.

2.1 CONCEITOS

A computação em *cloud* é, fundamentalmente, um novo paradigma de computação que fornece uma abstracção entre os recursos computacionais (computação, armazenamento, aplicações, etc. . .) e a infra-estrutura subjacente (servidores, *hardware* de armazenamento, dispositivos de rede, . . .), possibilitando o acesso conveniente e sob demanda a uma *pool* de recursos elásticos que podem rapidamente ser aprovisionados e libertados com um mínimo de esforço e intervenção (uma espécie de *self-service*) [1] [2] [3] [4]. Os modelos de computação em *cloud* podem ser organizados em três grandes categorias, de acordo com o nível de abstracção dos serviços e o nível de controlo sobre os recursos que disponibilizam aos seus utilizadores, tal como ilustrado na Figura 2:

Software-as-a-Service (SaaS): Neste tipo de modelo, o serviço disponibilizado ao utilizador é a possibilidade de aceder e utilizar remotamente aplicações informáticas, abstraindo completamente o utilizador das camadas inferiores da plataforma e da infra-estrutura. A adopção deste tipo de aplicações requer, habitualmente, pouco ou nenhum investimento por parte das organizações, uma vez que eliminam os custos com recursos computacionais para serem corridas e os custos com recursos humanos especializados para o seu desenvolvimento, instalação, configuração e manutenção. Geralmente, é apenas cobrado ao utilizador uma taxa de subscrição que lhe permite aceder ao serviço por um determinado período de tempo. Este é, por isso, um modelo especialmente atractivo para as empresas de pequena e média dimensão.

Platform-as-a-Service (PaaS): Num modelo deste tipo, é posta à disposição do utilizador uma plataforma completa (recursos computacionais, ferramentas, bibliotecas, etc. . .), onde este pode desenvolver e implementar as suas próprias soluções de *software*. Existem diversos serviços deste tipo, cada um deles oferecendo diferentes opções de escalabilidade e de manutenção, no entanto, o denominador comum a todos eles é a oferta de um ambiente de instalação e alojamento de aplicações e a possibilidade de integração com vários serviços (armazenamento, *logging*, monitorização, etc. . .). Desta forma, as organizações podem beneficiar de todas as vantagens de soluções de *software* desenhadas à medida libertando-se, no entanto, do esforço e dos custos ne-

cessários para manter e administrar toda a infra-estrutura subjacente. Para além destes serviços, têm surgido ferramentas que permitem às organizações criar as suas próprias soluções de *PaaS*, utilizando a sua infra-estrutura ou a de terceiros. Possibilitando-lhes assim criar uma plataforma própria, de desenvolvimento e alojamento, transversal a todas as suas aplicações.

Infrastructure-as-a-Service (IaaS): Neste modelo, as organizações têm um controlo mais granular e de baixo nível sobre a infra-estrutura. Ao utilizador são disponibilizados recursos computacionais fundamentais (computação, armazenamento, redes de comunicação) virtualizados, que lhe permitem construir sistemas distribuídos complexos, eliminando, no entanto, a necessidade de gerir um *datacenter* físico. Enquanto serviço, este modelo permite às organizações evitarem os custos de aquisição, manutenção e administração de *hardware*, ou tirarem partido destes serviços para complementarem a sua infra-estrutura física ajudando a suprir ocasionais picos de utilização. Enquanto produto, põe à disposição das organizações ferramentas e uma nova filosofia de gestão da infra-estrutura que lhes permite otimizar os seus recursos e tornar os seus processos mais eficientes.

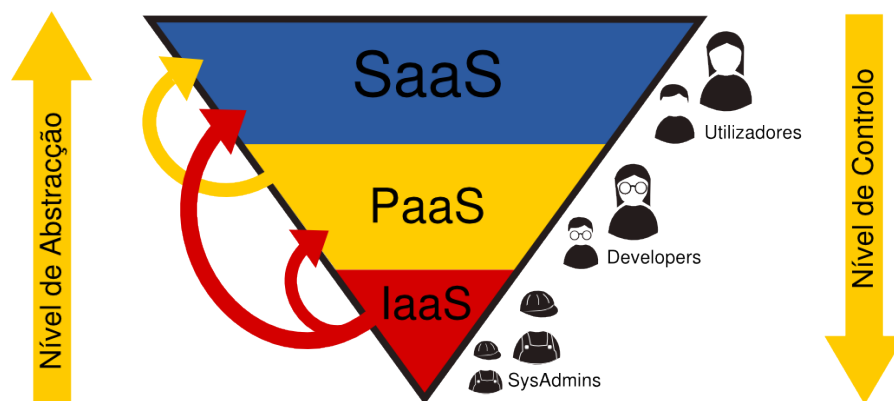


Figura 2: Os três modelos de serviço de computação em *cloud*.

Os serviços de computação em *cloud* podem ainda ser categorizados de acordo com o seu modelo de implementação:

cloud PRIVADA: A solução é disponibilizada para utilização exclusiva de uma única organização com múltiplos utilizadores. Pode ser detida, administrada e operada pela organização, por terceiros, ou uma combinação das duas, e pode estar localizada dentro ou fora das instalações.

cloud COMUNITÁRIA: A solução é disponibilizada para utilização exclusiva de uma comunidade específica de utilizadores pertencentes a um conjunto de organizações que partilham interesses comuns. Pode ser detida, administrada e operada por uma ou mais das organizações pertencentes à comunidade, por terceiros, ou uma combinação das duas, e pode estar localizada dentro ou fora das suas instalações. Pode, de certa forma, ser visto como um caso particular do modelo privado, divergindo, no entanto, no conceito de organização (organização enquanto conjunto de organizações).

cloud PÚBLICA: A solução é disponibilizada para utilização aberta pelo público em geral. Pode ser detida, administrada e operada por uma organização comercial, académica, ou governamental, ou uma combinação delas, e regra geral está localizada nas instalações do fornecedor do serviço.

cloud HÍBRIDA: A solução é uma combinação de duas ou mais infra-estruturas das anteriores (Privada, Comunitária ou Pública) que permanecem entidades únicas, mas que estão interligadas por tecnologia *standard* ou proprietária que permite a portabilidade de dados e aplicações (ex.: partilha da carga para balanceamento de carga entre *clouds*, ou externalização da carga quando a capacidade máxima da *cloud* é atingida).

Implementação / Serviço	<i>SaaS</i>	<i>PaaS</i>	<i>IaaS</i>
Privada / Comunitária	Não faz sentido!	<i>Apprenda</i> ¹ (produto comercial); <i>OpenShift</i> ² (projecto <i>open-source</i>)	<i>OpenStack</i> ³ (projecto <i>open-source</i>); <i>VMware vCloud Suite</i> ⁴ (produto comercial)
Híbrida	Não faz sentido!	<i>Pivotal Cloud Foundry</i> ⁵ (projecto <i>open-source</i> e serviço comercial)	<i>Eucalyptus</i> ⁶ (projecto <i>open-source</i> e serviço comercial)
Pública	<i>Salesforce</i> ⁷ (serviço comercial); <i>Basecamp</i> ⁸ (serviço comercial)	<i>Heroku</i> ⁹ (serviço comercial); <i>Google App Engine</i> ¹⁰ (serviço comercial)	<i>Amazon Web Services</i> ¹¹ (serviço comercial); <i>Rackspace Cloud</i> ¹² (serviço comercial)

Tabela 1: Matriz Serviço/Implementação de exemplos de soluções *cloud*.

Na Tabela 1 são apresentados alguns exemplos de soluções de diferentes modelos de serviço e de implementação. Não existem exemplos de soluções *SaaS* privadas/comunitárias ou híbridas por, por definição, não fazerem sentido.

Feita uma primeira abordagem ao conceito de computação em *cloud* e revendo as particularidades e os objectivos do *GSII*C, facilmente se percebe

¹ <http://apprenda.com>

² <https://www.openshift.com/>

³ <http://www.openstack.org>

⁴ <http://www.vmware.com/products/vcloud-suite>

⁵ <https://www.gopivotal.com/platform-as-a-service/pivotal-cf>

⁶ <https://www.eucalyptus.com/eucalyptus-cloud/iaas>

⁷ <https://www.salesforce.com>

⁸ <https://basecamp.com/>

⁹ <https://www.heroku.com>

¹⁰ <https://cloud.google.com/products/app-engine>

¹¹ <https://aws.amazon.com>

¹² <http://www.rackspace.com/cloud>

que as suas necessidades se alinham com as características de um modelo privado de *IaaS*, uma vez que este modelo se direcciona para organizações que possuem um *datacenter* próprio e que procuram novas formas de otimizar a sua operação.

2.2 CARACTERÍSTICAS DE UM SISTEMA DE OPERAÇÃO EM *cloud*

Os sistemas de operação em *cloud* constituem um novo paradigma e um conjunto de novas metodologias e processos de gestão de infra-estruturas informáticas.

A tendência de consolidação de recursos e normalização que começou com o advento das tecnologias de virtualização, vê na computação em *cloud* o estágio seguinte da sua evolução.

Enquanto num sistema tradicional os recursos são aprovisionados e configurados manualmente por pessoal qualificado, os sistemas de operação em *cloud* disponibilizam um ambiente *multi-tenant*¹³ onde os recursos são aprovisionados automaticamente (de forma *self-service*) a partir de uma *pool* de recursos de forma fácil e intuitiva (geralmente através de aplicações *web*) ficando imediatamente disponíveis para utilização.

A adopção de um sistema deste tipo acarreta não só mudanças ao nível da operação da infra-estrutura, mas também uma mudança da filosofia utilizada no seu desenho.

Actualmente, uma grande parte das infra-estruturas são desenhadas tendo em vista a robustez. Os *datacenters* são compostos por componentes redundantes (múltiplos sistemas de alimentação energética, múltiplos sistemas de climatização, etc...), servidores dispendiosos com múltiplas fontes de alimentação e soluções de armazenamento proprietárias. Tudo isto para que a camada de aplicação nunca tenha de lidar com uma falha da camada da infra-estrutura subjacente.

A filosofia da computação em *cloud*, por outro lado, aceita a inevitabilidade das falhas da infra-estrutura devendo a arquitectura das aplicações ser concebida tendo esse facto em consideração. Normalmente é utilizado *hardware* comum e menos dispendioso que permite à infra-estrutura escalar horizontalmente e não verticalmente, como acontece num sistema tradicional.

Uma analogia que ilustra, de forma muito interessante, esta diferença de paradigmas foi apresentada por *Randy Bias* (e atribuída a *Bill Baker*) no evento *CloudConnect*¹⁴ [5]. Num ambiente de operação convencional os servidores, tipicamente *Virtual Machines (VMs)*¹⁵, são como animais de estimação: recebem um nome (ex.: *gatodasbotas.gsiic.uc.pt*), são administrados manualmente e, quando há algum problema, são tratados até voltarem a ficar saudáveis. Num ambiente de operação em *cloud* os servidores são como gado: recebem um identificador genérico (ex.: *nig16677438.gsiic.uc.pt*) e quando adoecem são abatidos e substituídos por uma nova instância.

Existem diversas soluções (comerciais e *open-source*) que possibilitam a implementação de sistemas deste tipo. Embora difiram amplamente a nível

¹³ Princípio segundo o qual uma instância de um determinado *software* serve diversos grupos isolados de utilizadores/organizações (*tenants*).

¹⁴ <http://www.cloudconnectevent.com>

¹⁵ Uma máquina virtual (*VM*) é a implementação, em software, de uma máquina capaz de simular as funcionalidades de uma máquina física.

de funcionalidades disponibilizadas, é possível definir um conjunto de características comum a todas elas, que permita estabelecer uma arquitectura genérica de alto nível a que este tipo de solução obedece.

Um sistema de operação em *cloud* deve fornecer:

- Gestão de acessos e políticas de segurança
- Capacidades de monitorização
- Interface de administração do sistema
- Interface *self-service* de aprovisionamento e consumo dos recursos
- Aprovisionamento automático de recursos
- Orquestração de recursos
- Escalonamento de recursos
- Gestão de imagens (*templates* de VMs)

Muitas outras funcionalidades poderiam ser aqui inseridas, mas estas são suficientes para se descrever uma grande parte das capacidades de um sistema de operação em *cloud*.

Assim, na Figura 3 é possível ver uma possível arquitectura conceptual comum a este tipo de sistemas.

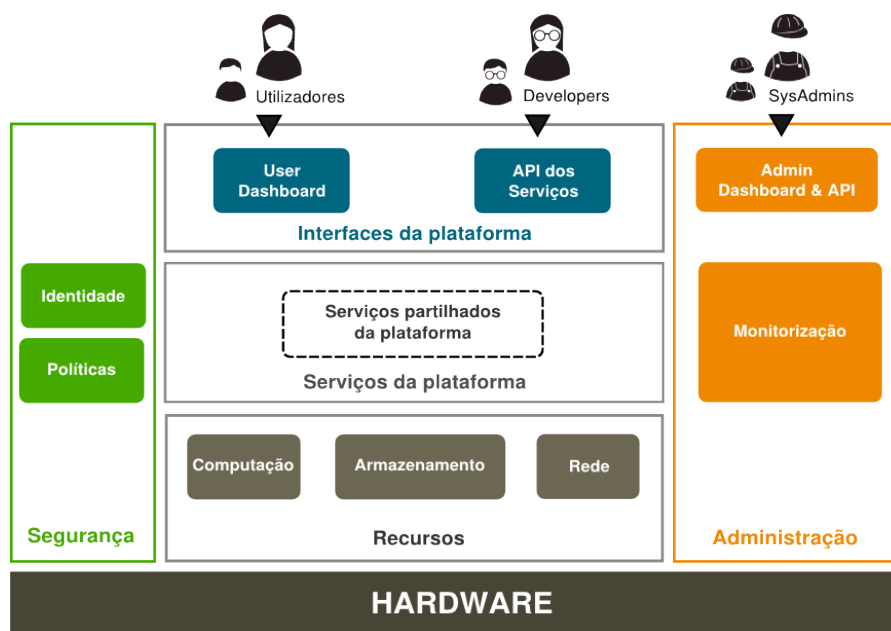


Figura 3: Arquitectura conceptual de um sistema de operação em *cloud*.

2.3 SOLUÇÕES COMERCIAIS

Um dos requisitos deste projecto consiste na utilização de soluções *open-source* que assentem em standards abertos.

Assim, devido à sua natureza, as soluções comerciais são incompatíveis com os objectivos deste projecto. Para além dos elevados custos de licenciamento tipicamente associados a este tipo de soluções, estes produtos fazem

uso de interfaces e *standards* proprietários que os tornam incompatíveis com os restantes produtos existentes no mercado. Isto cria para os utilizadores uma enorme dependência no fornecedor (*vendor lock-in*) uma vez que uma eventual migração se torna num processo extremamente complexo e com custos bastante elevados. No entanto, optou-se por inclui-las nesta análise, para efeitos de referência.

Não foram, no entanto, incluídas as soluções comerciais baseadas em soluções *open-source* (HP Cloud OS¹⁶, Citrix CloudPlatform¹⁷, etc...), uma vez que este tipo de produtos é composto por soluções *open-source* às quais foram acrescentadas ferramentas e pacotes proprietários que fornecem funcionalidades adicionais. Por existir uma grande (em alguns casos quase total) sobreposição de funcionalidades, optou-se pela sua exclusão.

2.3.1 VMware vCloud Suite

A solução, para a *cloud*, da VMware é uma poderosa plataforma, que agrega a extensa gama de tecnologias de virtualização (*hypervisors*, armazenamento, redes definidas por software, etc...) que esta comercializa.

Algumas das suas funcionalidades são: integração com ESXi¹⁸, balanceamento automático de carga (migração automática de VMs e volumes de armazenamento entre servidores para otimizar a utilização de recursos), migração em tempo real de VMs e de volumes de armazenamento, Virtual Machine File System (VMFS)¹⁹ e capacidades avançadas de High Availability (HA).

Embora, como se pode ver, esta seja uma solução dotada de funcionalidades poderosas, estas são alcançadas através da utilização de um grande número de tecnologias proprietárias da VMware, existindo um suporte muito diminuto, ou inexistente, a produtos e tecnologias exteriores ao ecossistema, o que cria uma enorme dependência no fabricante.

2.3.2 fluidOps eCloudManager²⁰

O eCloudManager da fluidOps é uma solução completa para administração de ambientes *cloud*. Suporta um grande número de *hypervisors* (vSphere, Hyper-V²¹, XEN²² e KVM²³) e pode inclusive controlar instalações OpenStack.

Uma característica muito peculiar é a introdução do conceito de *Landscape-as-a-Service* (LaaS), que permite o *deployment* automático de complexos ambientes aplicativos. O que lhe permite, por exemplo, fazer *deployment* de uma solução SAP completa de forma automática.

Tirando partido de um conjunto de parcerias com grandes vendedores de soluções de armazenamento, virtualização, rede e aplicações (EMC, NetApp, Microsoft, SAP, Fujitsu, VMware, etc...) o eCloudManager constitui uma solução unificada que permite administrar todas as camadas (desde a de *hardware* até à de *software*) de um *datacenter*.

¹⁶ <http://www.hpcloud.com/>

¹⁷ <https://www.citrix.com/products/cloudplatform/overview.html>

¹⁸ <http://www.vmware.com/products/esxi-and-esx/overview.html>. Analisado na Secção 3.5.4

¹⁹ Sistema de ficheiros proprietário, da VMware, de elevada performance especificamente criado para utilização em ambientes de virtualização.

²⁰ <http://www.fluidops.com/eccloudmanager>

²¹ <http://microsoft.com/hyper-v> - Hypervisor da Microsoft, distribuído com o Windows Server

²² <http://www.xenproject.org>. Analisado na Secção 3.5.3

²³ <http://www.linux-kvm.org>. Analisado na Secção 3.5.2

2.3.3 Flexiant Cloud Orchestrator²⁴

O *Cloud Orchestrator* é uma solução principalmente orientada para organizações que desejem fornecer serviços comerciais de *cloud*.

Com suporte a múltiplos *hypervisors* (KVM, XEN, VMware e Hyper-V) e *back-ends* de armazenamento, funcionalidades de HA, *Software-Defined Networking* (SDN) e orquestração aplicacional integradas, no entanto, as suas características mais distintivas são aquelas específicas do seu público-alvo: funcionalidades avançadas de medição de utilização, facturação, cobrança (e até de revenda) e integração com ferramentas externas de facturação.

2.3.4 OnApp Cloud²⁵

À semelhança do *eCloudManager*, o *OnApp Cloud* é uma solução orientada para serviços comerciais de *cloud*.

Com suporte a KVM, XEN e VMware, uma *Storage Area Network* (SAN) distribuída, funcionalidades embutidas de escalonamento automático e redundância a falhas, integração com a *OnApp Content Distribution Network* (CDN), um catálogo de milhares de *templates* de VMs prontas a utilizar e ferramentas integradas de gestão de planos de facturação (temporais ou com base no nível de utilização), o *OnApp Cloud* é uma ferramenta poderosa utilizada por um grande número de fornecedores de serviços de *cloud*.

2.3.5 Virtustream xStream²⁶

A proposta da *Virtustream*, pretende ser uma solução “tudo-em-um”. Permitindo criar *clouds* públicas, privadas e híbridas (existe mesmo uma “bolsa” onde as organizações podem vender e comprar capacidade de computação).

Suportando KVM, XEN e ESXi, o *xStream* apresenta como alguma das suas principais características: capacidade de monitorização avançada e alertas integrados, suporte à API da Amazon, um grande foco na segurança (inclui ferramentas de encriptação por *hardware*), uma solução integrada de *backup* e suporte a *back-ends* de armazenamento de um grande número de fabricantes.

2.3.6 CA AppLogic²⁷

A plataforma para *cloud* da CA Technologies, *AppLogic*, anteriormente comercializada pela 3tera (que foi adquirida pela CA em 2010), é uma solução com um propósito mais ambicioso do que as restantes soluções comerciais aqui enumeradas. Esta plataforma, para além de permitir criar soluções de *IaaS*, também permite criar soluções de *PaaS*, *SaaS* e *Virtual Private Data Center* (VPDC)²⁸, sendo utilizada, por um grande número de fornecedores, como base para os seus serviços comerciais de *cloud*.

²⁴ <http://www.flexiant.com/flexiant-cloud-orchestrator>

²⁵ <http://onapp.com/cloud>

²⁶ <http://www.virtustream.com/cloud-software/xstream>

²⁷ <http://www.ca.com/us/products/detail/ca-applogic.aspx>

²⁸ Um VPDC pode ser visto como um conjunto de serviços *cloud*, fornecidos como uma única solução, formando um *datacenter* virtual.

2.4 SOLUÇÕES *open-source*

No que diz respeito às soluções *open-source*, optou-se por analisar apenas as quatro mais proeminentes. Não se considerou relevante a análise de outras soluções (*Nimbus*²⁹, *Ganeti*³⁰, *Contrail*³¹, etc...) por serem direccionadas para âmbitos específicos (utilização científica, *clusters* de computação de alta performance, etc...) e/ou por terem uma base de utilizadores muito diminuta e documentação exígua.

2.4.1 *OpenNebula*³²

O *OpenNebula* nasceu de um projecto de investigação iniciado em 2005 por Ignacio M. Llorent e Rubén S. Montero, tendo a sua primeira versão sido lançada apenas em Março de 2008.

Sendo parte integrante de um conjunto de projectos de investigação internacionais (ex: *RESERVOIR*, *BonFIRE*, *StratusLab*) relacionados com computação em *cloud*, recebeu por diversas vezes apoio financeiro da *União Europeia* ao abrigo do programa *European Union Seventh Framework Programme (FP7)* [6].

Distribuído sob a *Apache Licence*, o *OpenNebula* é desenvolvido numa grande quantidade de linguagens: C++, C, Ruby, Java, Shell script, lex e yacc.

Um dos principais *deployments* de *OpenNebula* era, até há pouco tempo, o do *Conseil Européen pour la Recherche Nucléaire (CERN)*, que entretanto migrou para *OpenStack* precisamente por considerar arriscado utilizar uma solução da qual era o maior utilizador, sendo constantemente confrontado com situações limítrofe que nenhuma outra organização partilhava [7].

Algumas das principais funcionalidades do *OpenNebula* são:

- Suporte a KVM, XEN e VMware.
- Compatibilidade com múltiplas APIs: *Elastic Compute Cloud (EC2)* e *Elastic Block Store (EBS)*, *Open Grid Forum (OGF) Open Cloud Computing Interface (OCCI)*³³ e *vCloud*.
- Suporte nativo a *clouds* híbridas.
- Uma arquitectura direccionada para a tolerância a falhas e alta disponibilidade.
- Interfaces *web* distintas para utilizadores e administradores e uma *Command-Line Interface (CLI)*.

Alguns dos utilizadores do *OpenNebula* são a *European Space Agency (ESA)*, a *Telefonica* e a *Blackberry* [8].

2.4.2 *OpenStack*

Nascido no final de 2010 de uma parceria entre a *United States National Aeronautics and Space Administration (NASA)* e a *Rackspace* [9], o *OpenStack* é uma das mais proeminentes soluções para criação de plataformas *IaaS*.

²⁹ <http://www.nimbusproject.org>

³⁰ <https://code.google.com/p/ganeti>

³¹ <http://contrail-project.eu>

³² <http://opennebula.org>

³³ A OCCI consiste num conjunto de especificações desenvolvidas pela OGF, que têm como objectivo a criação de uma *API standard* e aberta para serviços de computação em *cloud*.

Actualmente coordenado pela *OpenStack Foundation*, o projecto *OpenStack* é patrocinado por um enorme número de organizações (mais de duzentas), entre as quais constam vários gigantes tecnológicos (*Red Hat*, *Canonical*, *HP*, *IBM*, *Cisco*, *Intel*, *Yahoo*, *VMware*, *EMC*, *Oracle*, etc. . .) [10] e conta com mais 900 contribuidores.

Desenvolvido essencialmente em *Python* sob a licença *Apache License*, o *OpenStack* consiste num conjunto de projectos interrelacionados responsáveis por gerir os diferentes recursos da infra-estrutura (computação, rede, armazenamento, etc. . .).

Algumas das principais funcionalidades do *OpenStack* são:

- Suporte a um grande número de *hypervisors* (*KVM*, *XEN*, *VMware ESXi*, *LXC*³⁴, *Hyper-V* e *Docker*³⁵).
- Suporte a um grande número de *back-ends* de armazenamento *block-based* (*Ceph*, *EMC*, *GlusterFS*, *IBM*, *Hitachi*, *Nexenta*, . . .).
- Suporte a diversos modos de rede, incluindo um sofisticado módulo de *SDN*.
- Possui uma plataforma distribuída e redundante de armazenamento de objectos (semelhante ao *Amazon Simple Storage Service (S3)*).

Alguns dos seus *deployments* mais notáveis são os da *Sony*, *Paypal*, *CERN*, *Wikipedia*, *Intel* e *NASA* [11].

2.4.3 *Apache CloudStack*³⁶

O *CloudStack* foi inicialmente desenvolvido pela *Cloud.com* (que nasceu em 2008, na altura com o nome *VMOps*) e disponibilizado sob a licença *GNU Public Licence (GPL) v3* em 2010.

Após a aquisição da *Cloud.com* pela *Citrix* (o que ditou o final do envolvimento, desta última, com o projecto *OpenStack*) em Julho de 2011, o desenvolvimento do *CloudStack* passou a ser coordenado pela *Apache Foundation* e a sua licença foi alterada para a *Apache Licence* [12].

Desenvolvido fundamentalmente em *Java*, o *CloudStack* apresenta como principais características:

- Suporte a *Xen*, *KVM* e *VMware ESXi* (usando *vCenter*).
- Uma interface web para administração e utilização da *cloud*.
- Uma *API* nativa ou alternativamente uma *API* compatível com a *Amazon EC2/S3*.
- Amplas capacidades de orquestração de redes (*SDN* e *Virtual LANs (vLANs)*³⁷).
- *Multi-tenancy* e *multi-role*, permitindo integração com *Lightweight Directory Access Protocol (LDAP)*.

³⁴ <https://linuxcontainers.org>

³⁵ <http://www.docker.com>

³⁶ <https://cloudstack.apache.org>

³⁷ O standard *IEEE 802.1Q* define um mecanismo através do qual é possível particionar uma rede, em diversas redes isoladas, com domínios de *broadcast* diferentes. Essas partições chamam-se *VLANs*. Este particionamento pode ser alcançado ao nível dos portos do *switch/router* (cada porto transporta tráfego de uma *vLAN* diferente), ou através de *vLAN Tagging* (adição de uma *tag* aos pacotes *Ethernet*), o que permite que tráfego de *vLANs* diferentes seja transportado num meio físico comum.

- Um assiente de *deployment* com interface gráfico.

Alguns *deployments* utilizando *ClouStack* são o da *GoDaddy*, *Tata Communications*, *Zynga*, *Spotify* e *Datapipe* (actualmente o de maior dimensão, agregando seis datacenters geograficamente distribuídos) [13].

2.4.4 *Eucalyptus*

Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems (*Eucalyptus*) é um sistema de operação em *cloud* que tem como principal objectivo disponibilizar uma interface compatível com a API dos diversos serviços da *Amazon*.

Desenvolvido pela *Eucalyptus Systems* utilizando principalmente *Java* e *C*, o *Eucalyptus* era, até há bem pouco tempo, distribuído num modelo comumente apelidado de *open-core* ou *open-source* comercial (existia uma versão *open-source* e uma versão comercial, sendo que algumas funcionalidades apenas estavam disponíveis na versão comercial).

O *Eucalyptus* foi utilizado inicialmente pela *NASA* no seu projecto *Nebula* mas, devido a preocupações com este modelo de licenciamento, foi abandonado em detrimento de uma solução construída de raiz (projecto que se viria a tornar uma das bases do *OpenStack*) [14].

O *Eucalyptus* tem como principais funcionalidades:

- Suporte a *KVM* e *VMware ESXi*.
- Estreita compatibilidade com a API da *Amazon EC2/S3/EBS*.
- Quatro modos de rede que permitem diversos níveis de isolamento de tráfego.
- Suporte a diversos *back-ends* para armazenamento de volumes.
- Uma solução distribuída de armazenamento baseado em objectos (*Walrus*).

2.5 COMPARAÇÃO DE SOLUÇÕES

Identificadas as principais soluções existentes, realizou-se uma análise comparativa e detalhada entre elas, possibilitando assim a criação de uma melhor perspectiva sobre as suas funcionalidades. O que permitiu decidir, de forma mais objectiva, a solução que melhor se adapta às necessidades deste projecto.

2.5.1 Informação Geral

Na Tabela 2 podem ver-se algumas características gerais das soluções analisadas. Nomeadamente a licença em que são disponibilizadas, as linguagens utilizadas no desenvolvimento e o modelo governativo da comunidade.

A licença é um ponto importante porque, desde o início do projecto, se estabeleceu a necessidade de a solução ser *open-source*. Neste caso, isso não é um problema uma vez todas são disponibilizadas sob licenças reconhecidas, pela *Open Source Initiative (OSI)*³⁸, como *open-source*.

³⁸ <http://opensource.org/>

A linguagem de desenvolvimento é uma outra característica que importa ter em consideração, porque se pretende que seja uma com a qual o *GSII* tenha experiência, de forma a possibilitar eventuais desenvolvimentos. Nesta rubrica, o claro favorito é o *OpenStack*, por ser fundamentalmente desenvolvido em Python/Web Server Gateway Interface (WSGI), a linguagem mais utilizada pelas equipas de desenvolvimento do *GSII*.

Por fim, o modelo governativo é de especial importância uma vez que é passível de determinar o futuro da solução (cuja adopção implica um compromisso a longo prazo por parte das organizações). Embora existam exemplos de projectos que funcionam muito bem sob o modelo de *Ditadura Benevolente*³⁹, existe a preferência por uma solução de modelo meritocrático, por a sua estrutura ser mais democrática e transparente.

	Licença	Linguagem	Modelo Governativo
<i>OpenNebula</i>	<i>Apache Licence</i>	<i>C++, C, Ruby, Java, Shell script, lex, yacc</i>	Ditadura Benevolente
<i>OpenStack</i>	<i>Apache Licence</i>	<i>Python, Shell script</i>	Meritocracia Técnica
<i>CloudStack</i>	<i>Apache Licence</i>	<i>Java, C</i>	Meritocracia Técnica
<i>Eucalyptus</i>	<i>GPL</i>	<i>Java, C</i>	Ditadura Benevolente

Tabela 2: Comparação de soluções *Open-Source* - Informação geral

2.5.2 Compatibilidade com APIs

A nível de interfaces utilizados pelas soluções de computação em *cloud*, existem actualmente duas grandes APIs com as quais estas fornecem compatibilidade. A da *Amazon* que, pela sua antiguidade e por ser amplamente utilizada, pode ser quase considerada um standard da indústria e a *OGF OCCI* que pretende ser o standard aberto e global para soluções de computação em *cloud* (*IaaS*, *PaaS* e *SaaS*). Algumas soluções procuram ainda oferecer compatibilidade com a interface da *VMware vCloud*. Na Tabela 3 podem ver-se as APIs suportadas pelas diversas soluções.

	Nativa	<i>Amazon EC2/S3/EBS</i>	<i>OGF OCCI</i>	<i>vCloud</i>
<i>OpenNebula</i>	Sim	Sim	Sim	Sim
<i>OpenStack</i>	Sim	Parcial	Sim	Não
<i>CloudStack</i>	Sim	Sim	?	Não
<i>Eucalyptus</i>	Não	Sim	Não	Não

Tabela 3: Comparação de soluções *Open-Source* - Compatibilidade com APIs

³⁹ "Ditadura Benevolente" é o termo utilizado para descrever o modelo governativo de projectos open-source onde um indivíduo, frequentemente o fundador do projecto, retém o poder de decisão final sobre qualquer discussão ou disputa. Alguns exemplos são o de *Linus Torvalds*, criador do *kernel Linux*, e *Guido van Rossum*, criador da linguagem *Python*. [15]

2.5.3 Hypervisors Suportados

Os *hypervisors* têm um papel central num sistema de operação em *cloud*, condicionando muitas vezes, as escolhas relativas aos sistemas de armazenamento ou de rede. Por este motivo, é importante que exista suporte ao maior número possível de *hypervisors*.

Para além disto, é desejável (mas não um factor eliminatório) que a solução a adoptar suporte simultaneamente um *hypervisor open-source* e *ESXi*. Desta forma, caso se revele necessário, será possível ao *GSII* combinar a solução de virtualização que já possui (*ESXi*) com uma nova, *open-source*.

Como se pode ver na Tabela 4, todas as soluções cumprem o desejo acima enunciado. Sendo que, no entanto, algumas requerem a utilização adicional do *vCenter*.

O *OpenStack* suporta ainda *Hyper-V*, *LXC* e *Docker*. Sendo que, estes dois últimos, são *hypervisors* em forte crescimento.

	<i>KVM</i>	<i>XEN</i>	<i>ESXi</i>	<i>Hyper-V</i>	<i>LXC</i>	<i>Docker</i>
<i>OpenNebula</i>	Sim	Sim	Sim (com ou sem <i>vCenter</i>)	Não	Não	Não
<i>OpenStack</i>	Sim	Sim	Sim (com ou sem <i>vCenter</i>)	Sim	Sim	Sim
<i>CloudStack</i>	Sim	Sim	Sim (só com <i>vCenter</i>)	Não	Não	Não
<i>Eucalyptus</i>	Sim	Não	Sim	Não	Não	Não

Tabela 4: Comparação de soluções *Open-Source* - *Hypervisors* suportados

2.5.4 Rede

No que diz respeito às funcionalidades de rede, foi analisado o suporte a diferentes modos de rede (*SDN*, *vLANs* e simples) bem como o suporte a *IPs* elásticos (endereços *Internet Protocol (IP)* públicos que podem ser dinamicamente atribuídos a uma *VM*) e à definição de grupos de segurança (conjuntos de regras de *firewall*).

As soluções revelaram-se muito semelhantes neste aspecto, como se pode ver na Tabela 5. Sendo que, com excepção do *Eucalyptus* que não possui uma solução de *SDN*, todas possuem as funcionalidades analisadas.

2.5.5 Armazenamento

Uma outra característica muito importante de um sistema de operação em *cloud* é o armazenamento. Foram identificados três tipos diferentes de serviços de armazenamento disponibilizados aos utilizadores:

block-based: permite aos utilizadores criar volumes de armazenamento que podem ser conectados às *VMs* para armazenamento permanente.

	SDN	vLANs	Simples	IPs Elásticos	Grupos de segu- rança
<i>OpenNebula</i>	Sim	Sim	Sim	Sim (Elastic IPs)	Sim
<i>OpenStack</i>	Sim (Neutron)	Sim (VlanMa- nager)	Sim (Flat- Manager e FlatDHCP- Manager)	Sim (Floating IPs)	Sim
<i>CloudStack</i>	Sim	Sim	Sim	Sim (Elastic IPs)	Sim
<i>Eucalyptus</i>	Não	Sim (Managed Mode)	Sim (Static Mode)	Sim (em Managed Mode)	Sim (em Managed Mode)

Tabela 5: Comparação de soluções *Open-Source* - Funcionalidades de rede suportadas

Este tipo de serviço é geralmente implementado recorrendo aos dispositivos de armazenamento dos próprios servidores usando *Logical Volume Manager (LVM)*⁴⁰ ou a uma *SAN*⁴¹. A informação é tratada como blocos dentro de sectores e faixas. O acesso aos volumes é feito com protocolos como o *Internet Small Computer System Interface (iSCSI)*⁴², *Fiber Channel Protocol (FCP)*⁴³ ou *ATA over Ethernet (AoE)*⁴⁴.

object-based: fornece aos utilizadores uma solução distribuída de armazenamento de informação não estruturada, sobre a forma de objectos (semelhante ao *Amazon S3*). Cada objecto é composto pelos dados, meta-dados e um identificador único. A interface deste tipo de serviços é, tipicamente, uma *API Representational State Transfer (REST)*. A existência de uma solução deste tipo é uma grande mais valia, uma vez que fornece aos utilizadores um repositório de dados central, ao qual podem aceder facilmente a partir de qualquer local (de forma algo semelhante ao que é possível com sistemas como a *Dropbox*⁴⁵).

file-based: possibilita aos utilizadores a criação de directórios de rede partilhados e a montagem automática destas partilhas nas VMs. A informação é organizada como uma hierarquia de ficheiros. Este tipo

⁴⁰ LVM é o gestor de volumes lógicos do *Linux*. Volumes lógicos podem ser vistos como uma tecnologia, baseada em software, análoga ao *Redundant Array of Inexpensive Disks (RAID)*, agregando diversos volumes físicos num único volume lógico.

⁴¹ Uma *SAN* é uma rede dedicada, que permite às organizações centralizarem o armazenamento dos seus servidores em *datacenters*. Os servidores acedem remotamente a volumes de armazenamento remotos, como se estes se tratassem de dispositivos locais.

⁴² O *iSCSI* é um protocolo *SAN*. Permite o acesso remoto, sobre infra-estruturas de rede tradicionais (*Local Area Network (LAN)*, *Wide Area Network (WAN)*, etc...), a dispositivos de armazenamento.

⁴³ O *FCP* é um protocolo em tudo semelhante ao *iSCSI*, no entanto requer cablagem especial.

⁴⁴ O *AoE* é um outro protocolo de *SAN*. A sua grande característica diferenciadora é o facto de se tratar de um protocolo de *layer 2* (não corre sobre *TCP/IP*). Esta singularidade faz com que o *aoe* seja non-routable, o que limita o escopo da sua aplicabilidade.

⁴⁵ <https://www.dropbox.com/>

de serviços é implementado com base numa *Network Attached Storage (NAS)*⁴⁶, onde o acesso aos ficheiros é disponibilizado através de protocolos comuns como o *Network File System (NFS)*⁴⁷ ou o *Common Internet File System (CIFS)*⁴⁸. Esta é uma funcionalidade, actualmente muito utilizada na Universidade, com directórios partilhados na rede a serem utilizados para partilha interdepartamental de documentos. É importante não confundir este tipo de serviço disponibilizado ao utilizador, com soluções de NAS que muitas das plataformas utilizam internamente para suporte de funcionalidades como migração em tempo real de VMs.

Infelizmente, como se pode ver na Tabela 6, nenhuma solução inclui um serviço de armazenamento *file-based* e apenas o *OpenStack* e *Eucalyptus* possuem soluções baseadas em objectos.

	<i>Block-based (SAN)</i>	<i>Object-based</i>	<i>File-based (NAS)</i>
<i>OpenNebula</i>	Não	Não	Não
<i>OpenStack</i>	Sim (<i>Cinder</i>)	Sim (<i>Swift</i>)	Não (projecto <i>Manilla</i> ainda em desenvolvimento)
<i>CloudStack</i>	Sim	Não (mas suporta integração com <i>Swift</i>)	Não
<i>Eucalyptus</i>	Sim	Sim (<i>Walrus</i>)	Não

Tabela 6: Comparação de soluções *Open-Source* - Serviços de armazenamento disponibilizados

2.6 SOLUÇÃO A ADOPTAR

Feita esta breve análise comparativa das funcionalidades disponibilizadas pelas diversas soluções e identificação das respectivas vantagens e desvantagens, que podem ser vistas na Tabela 7, a solução que emergiu como mais apropriada foi o *OpenStack*.

O facto de ser desenvolvido em *Python/WSGI* (com o qual as equipas de desenvolvimento do GSIIC têm ampla experiência), ter uma gigantesca comunidade, ter suporte a inúmeros *back-ends* de armazenamento e de rede de diversos fabricantes (que possibilita a utilização do variado equipamento que o GSIIC já possui) e possuir uma solução integrada de armazenamento baseado em objectos foram factores preponderantes na sua escolha.

O *CloudStack* revelou ser também uma solução muito sólida. No entanto, o facto de ter um conjunto limitado de apoiantes dentro dos principais

⁴⁶ Uma NAS é um dispositivo de armazenamento que permite o acesso remoto ao seu *file-system*. Ou seja, um servidor de ficheiros.

⁴⁷ O NFS é um protocolo *open standard* utilizado para aceder a uma NAS.

⁴⁸ O CIFS (anteriormente *Server Message Block (SMB)*) é um outro protocolo, desenvolvido pela *Microsoft*, para aceder a sistemas de ficheiros partilhados na rede. A sua utilização em *UNIX/Linux* é possível, através do *SAMBA*.

players da indústria (*Citrix* é o grande impulsionador), levanta algumas dúvidas quanto ao futuro do projecto.

	Vantagens	Desvantagens
<i>OpenNebula</i>	Utilizado em diversos projectos europeus	Suporte limitado a <i>back-ends</i> de armazenamento
<i>OpenStack</i>	Enorme comunidade; Desenvolvimento em <i>Python/WSGI</i> , já utilizado internamente no <i>GSII</i> ; Arquitectura altamente modular; Possui solução de armazenamento baseado em objectos	<i>Deployment</i> complexo, uma vez que (ainda) não inclui uma ferramenta para o efeito; Curva de aprendizagem elevada (devido à numerosa quantidade de projectos e terminologia associada)
<i>CloudStack</i>	Excelente interface; Possui assistente gráfico de <i>deployment</i> ; Suporta migração em tempo real de VMs através da interface web	<i>Citrix</i> é o principal impulsionador do desenvolvimento
<i>Eucalyptus</i>	Possui solução de armazenamento baseado em objectos; Facilidade de criação de modelos híbridos de <i>cloud</i>	Modelo de licenciamento ainda levanta algumas preocupações; Não possui solução de <i>SDN</i>

Tabela 7: Comparação de soluções *Open-Source* - Vantagens e desvantagens

3 | ESTUDO DO *OPENSTACK*

O *OpenStack* surgiu da união de esforços do *NASA's Ames Research Center* e da *Rackspace* (uma empresa de alojamento com sede no Texas).

Em 2010 ambas as entidades procuravam desenvolver uma solução de *IaaS*. A *NASA* no âmbito do projecto *Nebula*, tinha iniciado o desenvolvimento de uma ferramenta, chamada *Nova*, para computação em *cloud* e disponibilizou-a utilizando a licença *Apache*.

A *Rackspace*, que tinha já desenvolvido o *Cloud Files* (uma plataforma de armazenamento na *cloud* semelhante ao *Amazon S3*, que viria mais tarde a ser a base do *Swift*), identificou imediatamente o potencial do *Nova* e a parceria nasceu sob o nome *OpenStack*.

Agora sob a alçada da *OpenStack Foundation*, o *OpenStack* conta com uma enorme comunidade e com o apoio de inúmeros gigantes da indústria.

3.1 DESCRIÇÃO DO *openstack*

O *OpenStack* é um sistema de operação em *cloud* que possibilita a gestão de um largo conjunto de recursos de um *datacenter* através de um *dashboard web*, possibilitando aos seus utilizadores auto aprovisionarem recursos através desse mesmo interface. O *OpenStack* tinha inicialmente uma arquitectura monolítica, com grande parte das funcionalidades a serem servidas por um único módulo. No entanto, ao longo do tempo essas funcionalidades foram sendo divididas em projectos, dedicados a uma área em particular, e o *OpenStack* tem agora uma arquitectura altamente modular e distribuída, como se pode ver na Figura 4. Através da utilização de *plugins* e *drivers*, permite a adição de novas funcionalidades e a integração com um grande número de tecnologias, tornando-se assim altamente configurável e extensível.

3.2 PROJECTOS *openstack*

O *OpenStack* consiste actualmente (edição *Havana*) em nove projectos inter-relacionados e dedicados a implementar cada um dos serviços do sistema da plataforma. Existem diversos outros projectos (*Manilla*¹, *Ironic*², *TripleO*³, etc...), que deverão ser incluídos nas próximas versões, mas que ainda se encontram em fase embrionária ou experimental e que não foram, portanto, incluídos na análise abaixo.

¹ *Manilla* é a solução de armazenamento *file-based* do *OpenStack*. Ainda se encontra em fase de incubação.

² *Ironic* é o projecto que visa possibilitar o aprovisionamento de servidores físicos (*bare-metal*).

³ Tirando partido das capacidades disponibilizadas pelo *Ironic*, o *TripleO* (*OpenStack-on-OpenStack*) pretende tornar-se na ferramenta *standard* de *deployment* de *clusters OpenStack*.

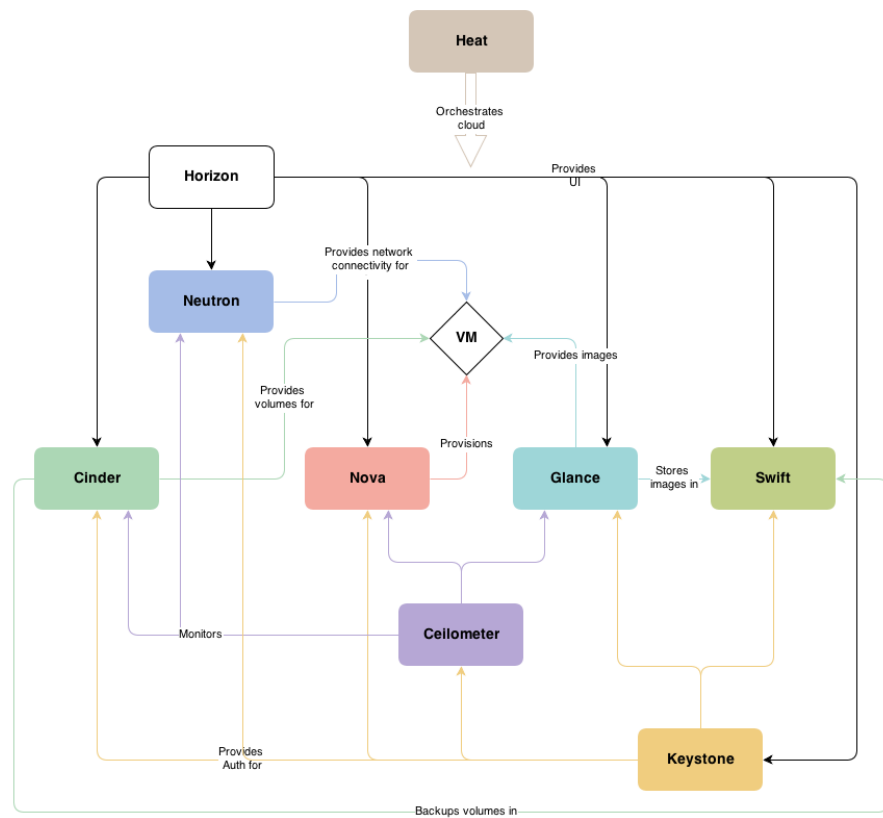


Figura 4: Arquitetura conceitual do *OpenStack*. (Imagem sob a licença Apache 2.0 extraída de [16])

3.2.1 Horizon

O *Horizon* (*OpenStack Dashboard*) é o *dashboard* do *OpenStack*. Trata-se que uma aplicação *web*, desenvolvida em *Django*, que disponibiliza aos administradores e utilizadores uma interface gráfica onde podem facilmente interagir com os diversos serviços do *OpenStack*.

Isto permite aos utilizadores aprovisionarem os seus próprios recursos, de um modo *self-service*, dentro dos limites e políticas de utilização definidos pelos administradores.

3.2.2 Keystone

Implementando a *Identity API*, o *Keystone* (*OpenStack Identity Service*) fornece os serviços de gestão de identidades, autenticação, gestão de políticas e de catálogo de serviços (listagem de serviços disponíveis) a todos os outros componentes do *OpenStack*.

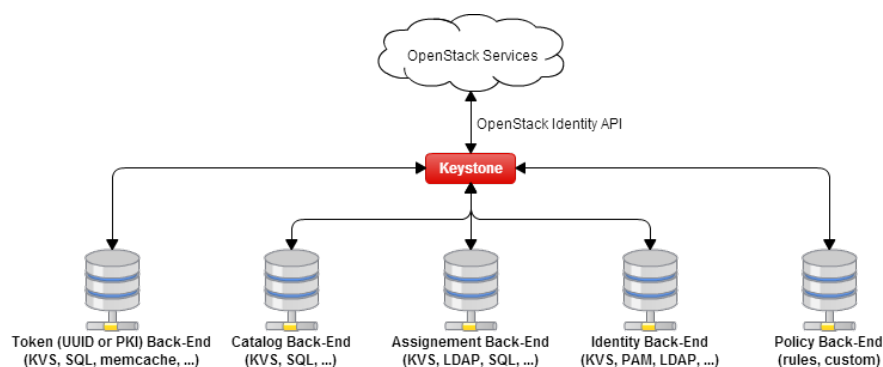


Figura 5: Arquitectura conceptual do *Keystone*

O *Keystone* faz a gestão dos utilizadores, *projectos* (ou *tenants*) e *roles*. Os *projectos* são unidades organizacionais aos quais podem ser assignados utilizadores, sendo que os utilizadores podem ser membros de um ou mais *projectos*. Os *roles* são atribuídos a pares “utilizador-projecto” e definem quais as permissões que um determinado utilizador possui, no âmbito de um determinado *projecto*.

Esta organização permite uma grande flexibilidade na gestão de permissões. É possível definir, por exemplo, a quota de armazenamento de um determinado *projecto* ou atribuir a um determinado utilizador permissão para utilização de *Floating IPs* num determinado *projecto*, mas não nos restantes a que está associado.

O *Keystone* suporta diferentes *plugins* e *backend*, para gestão e armazenamento dos vários tipos de informação que disponibiliza.

Como a Figura 5 ilustra, para tokens existe um plugin de *Universally Unique Identifier* (UUID) e outro de *Public Key Infrastructure* (PKI) e para identificação existe suporte a LDAP, Kerberos e X.509. Os *backends* suportados são *Structured Query Language* (SQL), *Key-Value Store* (KVS), LDAP, *memcached* e *rule files*.

3.2.3 Nova

O *Nova* é o componente mais complexo do *OpenStack*, tendo como algumas das suas principais responsabilidades:

- Iniciar, redimensionar e parar VMs.
- Atribuir e remover *Floating IPs*.
- Conectar e desconectar volumes.
- Adicionar, alterar e remover grupos de segurança.
- Criar *snapshots* de VMs.

A arquitectura conceptual do *Nova*, visível na Figura 6, é composta por um grande número de serviços, que a seguir se identificam:

nova-api: implementa as diferentes *APIs* do *Nova*.

nova-compute: utilizando as *APIs* dos diversos *hypervisors* (*libvirt*, *XenAPI*, etc. . .) cria e termina as VMs. Existe um *driver* especial que permite o aprovisionamento de servidores físicos.

nova-scheduler: determina, com base em diversos algoritmos de escalonamento (que podem ser configurados) e métricas de utilização, em que nó de computação uma determinada VM deve ser criada.

nova-network: disponibiliza recursos de gestão de redes que podem ser utilizados em *deployments* onde não se pretenda tirar partido dos recursos mais complexos e avançados de SDN disponibilizados pelo *Neutron*.

nova-console: fornece uma *proxy*, do tipo *Virtual Network Computing* (VNC) ou *Simple Protocol for Independent Computing Environments* (SPICE), para acesso às consolas das VMs.

QUEUE: Uma *queue Advanced Message Queuing Protocol* (AMQP) é utilizada pelos diferentes serviços do *Nova* para comunicarem entre si.

BASE DE DADOS: Uma base de dados SQL é utilizada para manter o estado da *cloud* (VMs em execução, redes disponíveis, etc. . .).

O processo de criação de uma nova instância pode ser descrito nos seguintes passos:

1. O utilizador requer a criação de uma nova instância, de um determinado *flavor*⁴, utilizando uma determinada imagem e, opcionalmente, associando-lhe um volume de armazenamento persistente.
2. O *nova-scheduler* decide em que nó de computação a instância deve ser inicializada.
3. O *nova-compute* obtém do *Glance* a imagem seleccionada pelo utilizador e copia-a para o nó de computação.
4. Utilizando o *driver* apropriado, o *nova-compute* solicita ao *hypervisor* a criação de uma nova VM com as características definidas no *flavor* seleccionado pelo utilizador.

⁴ Um *flavor* define um conjunto de características padrão (quantidade de *Random Access Memory* (RAM), número de processadores, etc. . .) que as instâncias podem possuir.

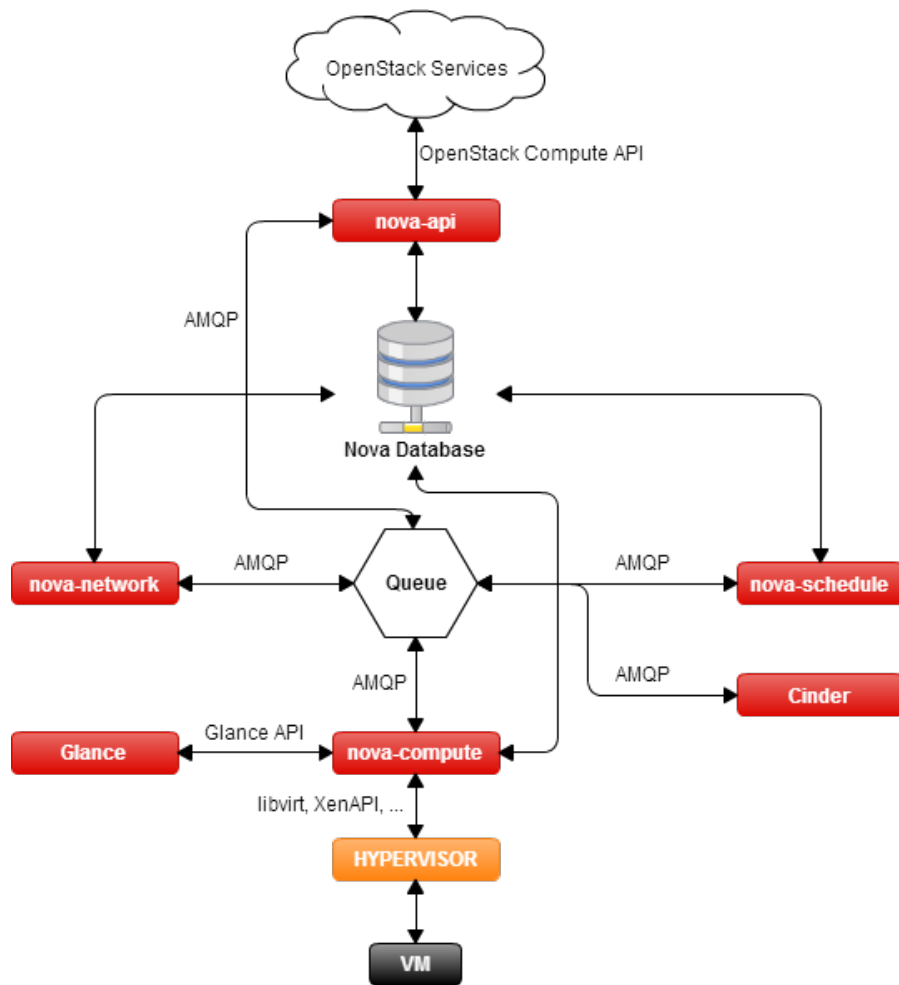


Figura 6: Arquitectura conceptual do *Nova*

5. O *nova-compute* associa à instância a imagem que será utilizada no seu arranque.
6. O *nova-compute* cria um segundo disco de armazenamento efêmero, que pode ser utilizado pela VM para armazenamento mas que será destruído aquando esta.
7. O *nova-compute* cria um terceiro disco que é mapeado via *iSCSI* para o volume opcionalmente seleccionado pelo utilizador e cuja localização é obtida por consulta ao *Cinder*.
8. O *nova-compute* solicita ao *hypervisor* que inicialize a VM.

3.2.4 Glance

O *Glance*, também conhecido como *OpenStack Image Service*, é o responsável por fornecer os serviços de descoberta, registo e distribuição de imagens a serem utilizadas pelas VMs.

Estas imagens são *templates* de *file systems* (incluindo Sistema Operativo (SO), *software* básico, etc...) a serem utilizadas como base de operação de uma VM.

O *OpenStack* permite que se crie um *snapshot* de uma instância em execução e, com base neste, se crie automaticamente uma imagem no *Glance*. Esta imagem pode então ser utilizada para criar novas máquinas virtuais, em tudo semelhantes à original.

A arquitectura conceptual do *Glance*, que pode ser vista na Figura 7, é essencialmente composta pelos seguintes serviços:

GLANCE-API: Implementa a *OpenStack Image API*, recebendo pedidos de pesquisa, armazenamento e obtenção de imagens.

GLANCE-REGISTRY: Obtém e processa a informação sobre as imagens (tamanho, tipo, etc...).

BASE DE DADOS: Uma base de dados *SQL* é utilizada para armazenar a informação produzida pelo *glance-registry*.

back-end DE ARMAZENAMENTO: Através da utilização de um *driver*, o *Glance* persiste as imagens num determinado *back-end* de armazenamento. Pode ser, inclusivamente, utilizado o *Swift*.

3.2.5 Swift

O *Swift* (*OpenStack Object Storage*) é uma solução distribuída, redundante, escalável e altamente disponível de armazenamento baseado em objectos, à semelhança do *Amazon S3*.

Utilizando hardware comum, o *Swift* permite criar um *cluster* de armazenamento, que pode ser utilizado para armazenar quantidades maciças de dados não estruturados. A sua arquitectura lógica pode ser vista na Figura 9.

O *Swift* é composto pelos seguintes componentes:

SERVIDOR proxy: Implementa a *OpenStack Object API*. Trata de todos os pedidos feitos ao *Swift*.

OBJECTO: A estrutura que representa os próprios dados.

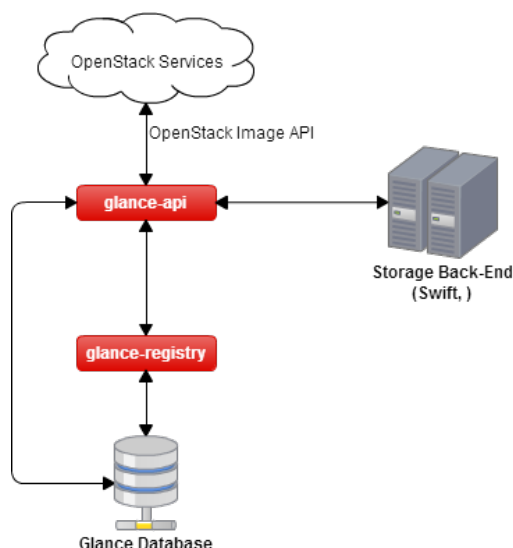


Figura 7: Arquitectura conceptual do Glance

accounts E containers: As *Accounts* e os *Containers* são as unidades estruturais do *Swift*. Cada *account* e cada *container* são uma base de dados SQL isolada. Um *container* contém uma lista de objectos contidos nesse mesmo *container*. Uma *account* contém um grupo de *containers*. A relação entre estes três conceitos pode ser vista na Figura 8.

PARTIÇÃO: Uma partição agrega um conjunto de objectos, *accounts* e *containers* que são armazenados em disco conjuntamente.

swift ring: O algoritmo responsável por fazer o mapeamento de partições para determinadas localizações em disco.

ZONAS: As zonas são a forma como o *Swift* alcança as capacidades de alta disponibilidade e redundância. Cada zona é isolada das restantes e possui uma cópia de cada objecto existente no *cluster*.

3.2.6 Cinder

O *Cinder* (*OpenStack Block Storage*) é a solução de armazenamento *block-based* do *OpenStack*. Este é o serviço responsável por disponibilizar os dispositivos de armazenamento (volumes) utilizados pelas máquinas virtuais para armazenamento persistente.

A implementação, por defeito, do *Cinder* é uma solução que utiliza *iSCSI* e *LVM* em *Linux*.

Para além desta implementação, existem ainda drivers que permitem a utilização de vários outros dispositivos de armazenamento, de diversos fabricantes (*EMC*, *NetApp*, *Dell*, *VMware*, *HP*, *IBM*, etc...), como *back-end*.

A sua arquitectura pode ser vista na Figura 10, onde se pode perceber melhor o processo através do qual uma *VM* obtém um volume.

O *Cinder* não é uma solução de armazenamento partilhado como uma *SAN* ou como volumes *NFS*. Com o *Cinder*, um volume não pode ser conectado a mais do que uma *VM* simultaneamente.

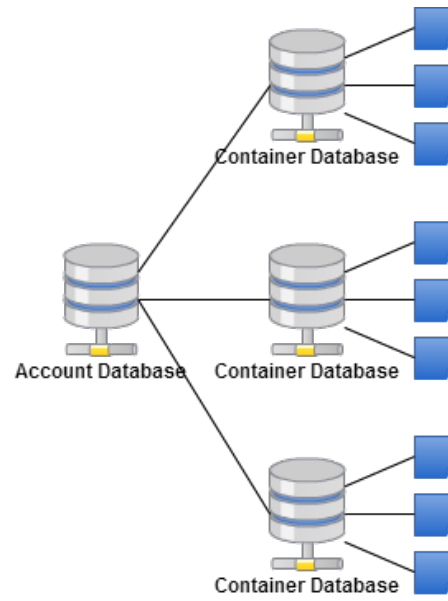


Figura 8: Relação entre *account*, *container* e *object*

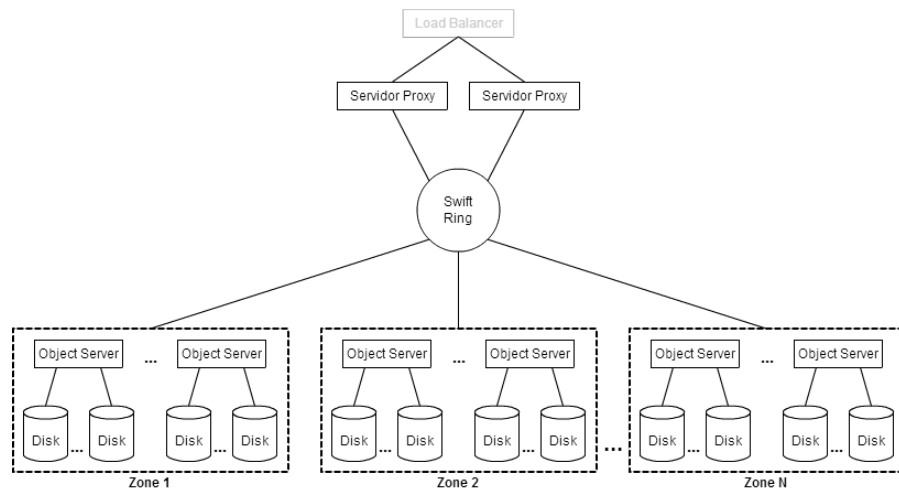


Figura 9: Arquitectura lógica do *Swift*

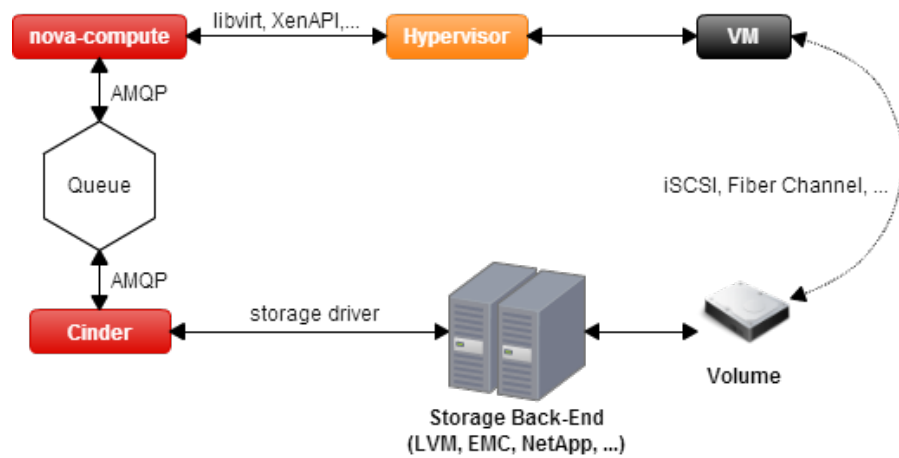


Figura 10: Arquitectura conceptual do *Cinder*

3.2.7 Neutron

O *Neutron* (*OpenStack Networking*) é o serviço de rede (SDN) do *OpenStack*. Recorrendo à utilização de *Open vSwitch*⁵ o *Neutron* disponibiliza capacidades de *Networking-as-a-Service* (*Virtual NICs (vNICs)*,...) que abstraem a camada de controlo (onde é decidido para onde o tráfego deve ser direccionado) da camada de dados (sistemas de mais baixo nível que direccionam o tráfego para o destino seleccionado).

Esta abstracção possibilita uma enorme flexibilidade, útil para fazer face às necessidades de complexos ambientes *multi-tenant*.

Uma limitação actualmente conhecida da utilização do *Neutron* é que, devido a não ser possível utilizar mais do que um *plugin* (que implementa o suporte à tecnologia de rede de um determinado fabricante) em simultâneo, todos os dispositivos de rede devem ser do mesmo fabricante. Existe já, pelo menos uma, solução para este problema, embora ainda em estado experimental.

3.2.8 Heat

O *Heat* (*OpenStack Orchestration*), disponível a partir da versão *Havana*, é um motor de orquestração aplicacional. Permite, através da utilização de *templates* (compatíveis com o formato dos *templates* da *Amazon CloudFormation*), fazer o *deployment* automático de aplicações distribuídas sobre o *OpenStack*.

Um *template*, sob a forma de um ficheiro de texto, descreve a infra-estrutura (servidores, *floating IPs*, volumes,...) de uma aplicação distribuída. Utilizando estes *templates*, o *Heat* invoca as *APIs* dos serviços necessários, de forma a criar automaticamente esta infra-estrutura.

O *Heat* permite ainda a integração com ferramentas de gestão de configurações (*Chef*, *Salt*,...), o que possibilita não só o aprovisionamento dos recursos, como também a sua configuração.

3.2.9 Ceilometer

O *Ceilometer* ou *OpenStack Telemetry*, disponível a partir da versão *Havana* e ainda numa fase algo inicial do seu desenvolvimento, tem como objectivo centralizar a monitorização e métrica de recursos de todos os serviços *OpenStack*.

Embora suporte outros, o *back-end* para armazenamento recomendado é o *MongoDB*.

3.3 TOPOLOGIA DE REDE

Depois de analisado em detalhe o *OpenStack* decidiu-se fazer uma análise mais exaustiva às capacidades de rede disponibilizadas pelo *nova-network*.

Como não se pretende, pelo menos nesta fase, utilizar as capacidades mais avançadas do *Neutron* e como o *nova-network* dispõe de vários modos de rede que influenciam directamente a topologia da mesma, achou-se relevante esta análise.

Os modos de rede suportados pelo *nova-network* (implementados por *drivers* de igual nome) são: *FlatManager*, *FlatDHCPManager* e *VlanManager*.

⁵ <http://openvswitch.org/> - O *Open vSwitch* é um projecto *open-source* de virtualização de redes.

Quer o *FlatManager* quer o *FlatDHCPManager* assentam no conceito de *bridged networking*.

A principal ideia por detrás do destes dois modos de rede é ter uma única *pool* de endereços definida em todo o *cluster* (daí o *flat*), que é partilhada por todas as *VMs*, independentemente do *tenant* a que pertencem.

Esta abordagem sofre de uma limitação conhecida da utilização de *bridges* em Linux: uma *bridge* pode apenas ser associada a uma única interface física. Isto leva à não existência de isolamento de tráfego *Layer 2* entre os servidores, uma vez que todos eles partilham o mesmo domínio de *broadcast Address Resolution Protocol (ARP)*.

3.3.1 *FlatManager*

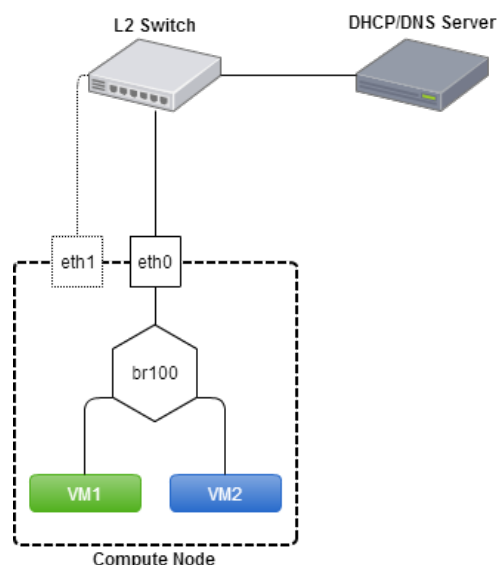


Figura 11: *nova-network - FlatManager*

O *FlatManager*, de todos os *drivers*, é aquele que fornece o conjunto de operações mais rudimentar. O seu papel, tal como pode ser visto na Figura 11, limita-se a fazer a conexão das *VMs* à *bridge*, não fazendo, por defeito, qualquer configuração de *IPs* nas *VMs*. Esta tarefa é deixada para o administrador do sistema, que o pode fazer utilizando um servidor *Dynamic Host Configuration Protocol (DHCP)* externo ao *cluster OpenStack*, ou outros meios.

3.3.2 *FlatDHCPManager*

O *FlatDHCPManager* pode ser visto, essencialmente, como uma extensão do *FlatManager*. Para além de conectar as *VMs* à *bridge*, o *FlatDHCPManager* também disponibiliza um servidor *DHCP* que estas podem utilizar.

O método de operação, por defeito, consiste numa única instância do serviço *nova-network* e do *dnsmasq*⁶ a correrem no nó controlador.

Embora esta abordagem tenha como benefício o facto de não ser necessário instalar e configurar uma instância do *nova-network* e do *dnsmasq* em cada um dos nós de computação, tem também a desvantagem de o nó con-

⁶ <http://www.thekelleys.org.uk/dnsmasq/doc.html> - O *dnsmasq* é um pequeno e leve servidor de *Domain Name System (DNS)*, *DHCP* e *Trivial File Transfer Protocol (TFTP)*.

trolador agir como *default gateway* de todas as VMs, criando assim um *Single Point of Failure (SPOF)*.

Na Figura 12, pode ver-se que as quatro VMs, pertencentes a três *tenants* diferentes (verde, azul e laranja), partilham um único espaço de endereçamento e que todo o seu tráfego é gerido por uma única instância do *nova-network* e do *dnsmasq* a correrem no *Controller Node*, no centro da figura.

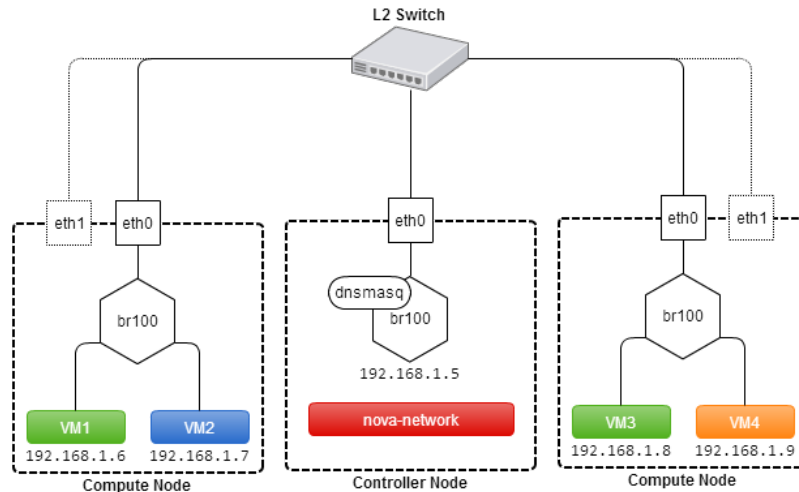


Figura 12: *nova-network - FlatDHCPManager (single-host)*

Este problema pode ser adereçado, utilizando uma abordagem *multi-host* (ilustrada na Figura 13). Cada nó de computação corre os seus próprios serviços *nova-network* e *dnsmasq*, sendo atribuída a cada VM um *default gateway* que depende do nó de computação em que é criada.

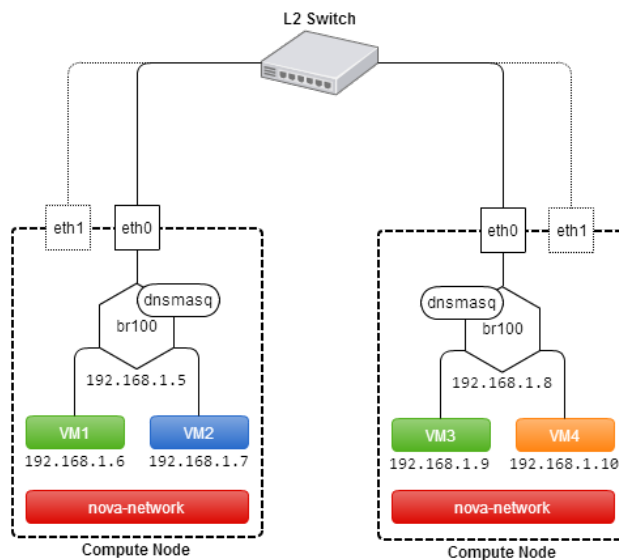


Figura 13: *nova-network - FlatDHCPManager (multi-host)*

No entanto, todas as VMs num destes modos de rede conseguem "ver" o tráfego umas das outras, independentemente do *tenant* a que pertencem.

Estes drivers são, portanto, uma boa opção para soluções cujas necessidades possam ser satisfeitas com um único espaço de endereçamento IP

e onde a gestão de endereçamento possa ser feita por um servidor *DHCP* externo ao *cluster* ou então, de forma simples, pelo *dnsmasq*.

Por outro lado, soluções implementadas com estes drivers, podem ainda sofrer de problemas de escalabilidade, uma vez que todas as *VMs* partilham o mesmo domínio de *broadcast ARP*.

Estes problemas (*multi-tenancy* e escalabilidade) são, de certa forma, adequados pelo *VlanManager*.

3.3.3 VlanManager

Enquanto os drivers *flat* são uma boa opção para *deployments* simples e/ou de pequena dimensão, o *VlanManager* é a opção ideal para *deployments* de larga escala.

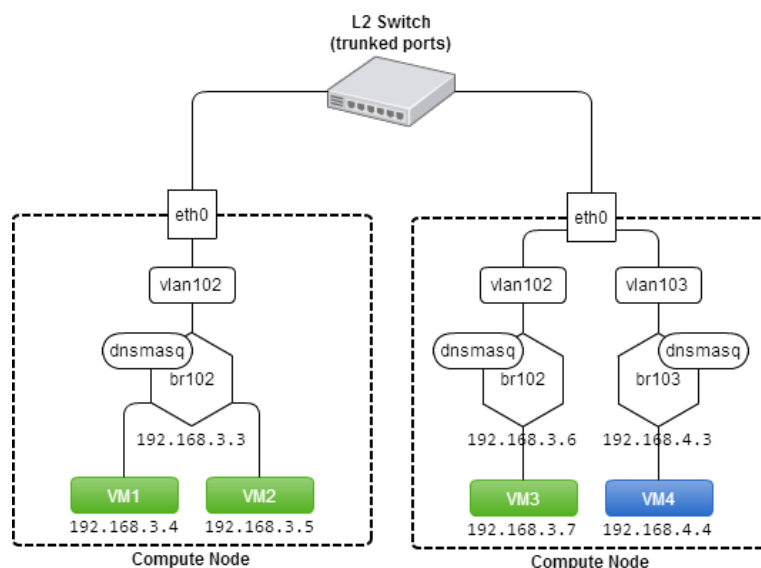


Figura 14: *nova-network* - *VlanManager*

O *VlanManager* recorre à utilização de *vLANs* para particionar uma rede física em domínios de *broadcasting* diferentes (desta forma, dispositivos em *vLANs* diferentes não se conseguem "ver").

Com este *driver*, cada *tenant* possui um espaço de endereçamento dedicado (uma *bridge* e uma *vLAN* diferentes). Para suportar a comunicação entre *VMs* de um mesmo *tenant* em nós de computação diferentes é utilizado *vLAN Tagging*, pelo que o *switch* a utilizar deve ser configurado apropriadamente (portas do *switch* em *trunk mode* em vez de *access mode*).

Na Figura 14 podem ver-se quatro *VMs*, pertencentes a dois *tenants* diferentes (verde e azul). Pode ver-se que cada um dos *tenants* tem uma *bridge* e uma *vLAN* dedicada para as suas *VMs*, atingindo-se assim o isolamento de tráfego entre *tenants*.

3.3.4 Floating IPs

Há ainda um outro conceito relacionado com o modo como o *OpenStack* faz a gestão de rede que importa perceber.

Um *Floating IP* é um endereço *IP*, tipicamente público (acessível do exterior do *cluster OpenStack*), que pode ser dinamicamente atribuído a uma

VM em execução, em adição ao IP privado que lhe foi automaticamente assignado no momento da sua criação.

Isto permite que determinadas VMs possam ser acedidas a partir do exterior do *cluster*, o que é necessário para que, por exemplo, a VM possa agir como um servidor *Hypertext Transfer Protocol (HTTP)*.

Esta funcionalidade é fornecida com recurso a *Network Address Translation (NAT)* e *IP Forwarding* e, caso seja utilizada, deve-se garantir que a comunicação entre VMs é feita utilizando o endereço IP privado e não o *Floating IP*.

3.4 INSTALAÇÃO DE UM SISTEMA DE DEMONSTRAÇÃO

Após a fase de estudo do *OpenStack*, dos seus componentes e arquitectura, passou-se à instalação de um pequeno sistema de demonstração *single-host* (todos os serviços a correr num único servidor), com o objectivo de obter algum conhecimento prático sobre o processo de *deployment* e operação do *OpenStack*. A versão inicialmente utilizada foi a *Grizzly*, tendo sido posteriormente feita a migração para a versão *Havana*.

3.4.1 Instalação

Para o processo de *deployment* optou-se pela utilização do *DevStack*⁷, que consiste num conjunto de *scripts* que possibilitam a rápida criação de ambientes de desenvolvimento *OpenStack*. Embora o seu principal objectivo seja criar ambientes de desenvolvimento, tem também sido frequentemente utilizado na criação de ambientes de teste e provas de conceito simples.

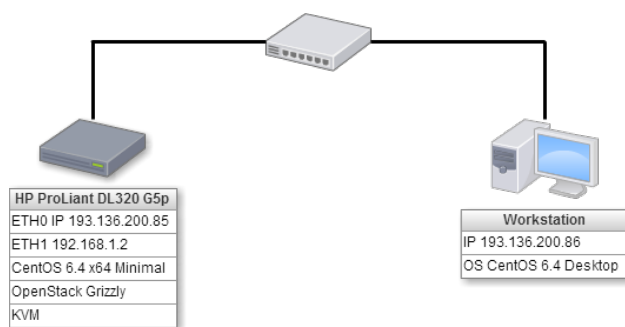


Figura 15: Arquitectura física do sistema de demonstração.

Para o sistema de demonstração foi utilizado um servidor *HP ProLiant DL320 G5p*, uma *workstation* simples e um *switch Cisco Catalyst 3550*, configurados conforme a Figura 15.

O sistema operativo utilizado foi o *CentOS 6.4, x64 Minimal* no servidor e *x86 Desktop* na *workstation*. O *hypervisor* utilizado foi o *KVM*.

Após instalado o sistema operativo e configuradas as interfaces de rede, foram seguidos os passos descritos no Anexo A para a instalação do *OpenStack*, utilizando o *DevStack*.

⁷ <http://devstack.org/>

3.4.2 Ensaios

A implementação deste sistema permitiu o ensaio de alguns procedimentos básicos. Nomeadamente:

- Criação de projectos, utilizadores e roles.
- Criação de instâncias.
- Criação de um volume e atribuição a uma instância.
- Criação de um *snapshot* de uma instância.

3.5 AVALIAÇÃO DE *hypervisors*

Depois de implementado o sistema experimental de *OpenStack* com *KVM*, e realizados alguns ensaios preliminares, o passo seguinte consistiu em testar as capacidades de integração com outros *hypervisors*.

Para além do *KVM*, que foi utilizado inicialmente, decidiu-se testar também a integração com o *XEN* e com o *ESXi*. Interessa primeiro, no entanto, perceber o conceito de *hypervisor* e conhecer os tipos que existem para que se possa compreender as diferenças entre aqueles que foram testados.

3.5.1 O que é um *hypervisor*

Um *hypervisor* é uma plataforma de virtualização que possibilita a utilização simultânea de diversos sistemas operativos num único computador.

O *hypervisor* “particiona” os recursos físicos disponíveis e aloca-os a VMs diferentes.

Existem essencialmente dois tipos de *hypervisor*, cujas diferenças estão ilustradas na Figura 16.

- Os de tipo 1 (também chamados de *bare-metal*), correm directamente sobre o hardware e dispensam, regra geral, a existência de um sistema operativo anfitrião. Um caso muito particular deste tipo de *hypervisor* é o *KVM*. Embora seja um *hypervisor bare-metal*, este é frequentemente mal classificado, uma vez que, como se pode ver na Figura 17, a sua instalação tem de ser, necessariamente, precedida da de um sistema operativo anfitrião. No entanto, o *KVM* não corre sobre o sistema operativo, sendo na verdade um módulo do *kernel* do *Linux*. Alguns exemplos de *hypervisors* do tipo 1 são: *KVM*, *VMware ESXi*, *XEN* e *Microsoft Hyper-V*.
- Os *hypervisors* de tipo 2 (também denominados de *hosted*) correm dentro do ambiente de um sistema operativo anfitrião, tratando-se fundamentalmente de uma segunda camada de software sobre o hardware. Alguns exemplos de *hypervisors* deste tipo são: *QEMU*, *Oracle Virtual-Box* e *VMware Workstation*.

3.5.2 *KVM*

O *KVM* (*Kernel-based Virtual Machine*) é uma solução *open-source* de virtualização para *Linux* em arquitecturas *x86*, consistindo em dois módulos do

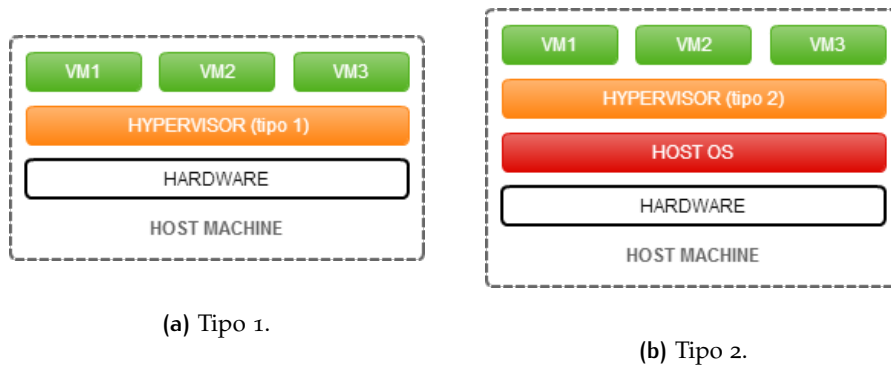


Figura 16: Arquitectura conceptual de um *hypervisor*.

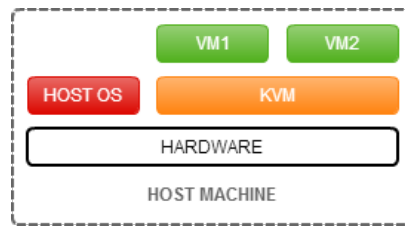


Figura 17: KVM - Caso particular de um *hypervisor* do tipo 1.

kernel (um que fornece a infra-estrutura nuclear de virtualização e um dependente da arquitectura do processador – *Intel* ou *AMD*). O KVM é desenvolvido pela *Open Virtualization Alliance*.

Para além das suas capacidades técnicas, as principais razões para a escolha do KVM foram a sua enorme base de utilizadores (em ambientes *OpenStack* é muito provavelmente o *hypervisor* mais utilizado), a vasta documentação disponível e a grande capacidade de integração com *OpenStack* (visível na Figura 18). Sendo aquele com maior número de funcionalidades oficialmente suportadas [17].

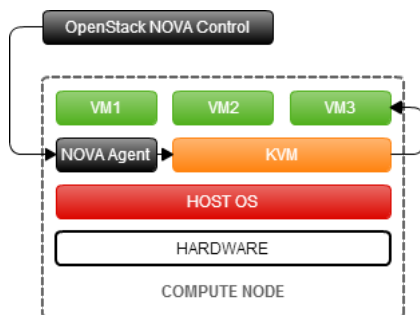


Figura 18: Integração de KVM com *OpenStack*.

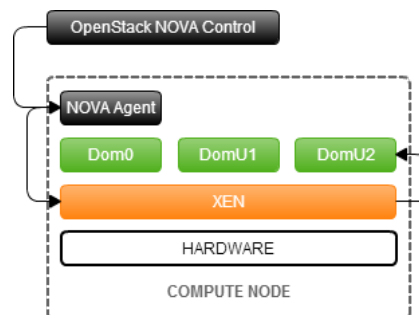


Figura 19: Integração de XEN com *OpenStack*.

3.5.3 XEN

O XEN é um *hypervisor bare-metal open-source* inicialmente desenvolvido pela Universidade de Cambridge e hoje um projecto sob a alçada da *Linux Foundation*.

O *XEN* corre directamente sobre o *hardware*, no entanto, para ser usável requer que exista um sistema operativo (análogo ao *SO* anfitrião no *KVM*) instalado numa máquina virtual especial denominada *Domo* (*domain zero*).

Esta *VM*, com privilégios especiais, é iniciada automaticamente durante o arranque do *XEN*, e é responsável por correr as suas ferramentas de administração.

À semelhança do *KVM*, o *XEN* foi escolhido por se tratar de um *hypervisor* *open-source* dotado de um vasto leque de funcionalidades e com uma boa compatibilidade com *OpenStack*. Um diagrama da sua integração pode ser visto na Figura 19.

Como curiosidade, o *XEN* foi o *hypervisor* inicialmente utilizado pela *Rackspace* e pela *NASA* no projecto que viria a dar origem ao *OpenStack*.

3.5.4 VMware ESXi

O *ESXi* é um *hypervisor* proprietário do tipo 1 comercializado pela *VMware*.

Embora seja uma ferramenta bastante poderosa e com um grande número de funcionalidades, esta é uma solução comercial que acarreta elevados custos de licenciamento.

A principal razão pela qual foi escolhido deve-se ao facto de, actualmente, ser extensivamente utilizado pelo *GSIIC* no seu *datacenter*.

O *ESXi* é suportado pelo *OpenStack* em dois modos de funcionamento, recorrendo a dois drivers distintos:

- O modo *stand-alone*, visível na Figura 20, em que o *nova-compute* comunica directamente com o *ESXi* usando o *VMwareESXDriver*. Para além de não suportar funcionalidades avançadas da *VMware*, este *driver* tem a enorme desvantagem de requerer uma instância do *nova-compute* por cada *host ESXi*. Uma vez que não podem ser ambos executados num mesmo servidor físico, este cenário torna-se rapidamente insustentável.
- O modo *clustered*, ilustrado na Figura 21, usando o *VMwareVCDriver*. Neste caso, o *nova-compute* comunica com uma instância do *vCenter* que gere um *cluster* de *hosts ESXi*, existindo apenas uma instância do *nova-compute* a correr por cada *cluster*. Com este *driver*, e havendo acesso a armazenamento partilhado, existe compatibilidade com as funcionalidades de *HA*, *acdrs* e *vMotion* da *VMware*.

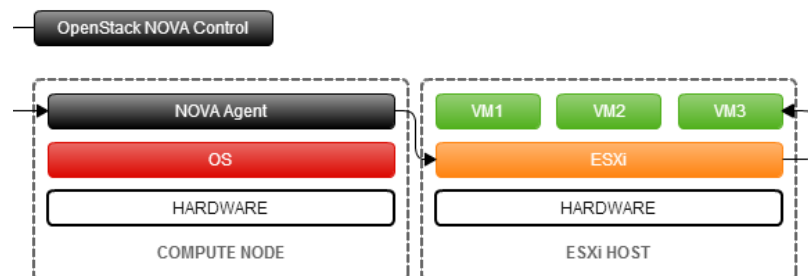


Figura 20: Integração de *ESXi* com *OpenStack* (modo *stand-alone*).

Os testes respeitantes ao *ESXi* foram, em grande parte, inconsequentes. A escolha inicial recaiu sobre o modo *stand-alone* e só na fase de testes nos apercebemos das suas limitações.

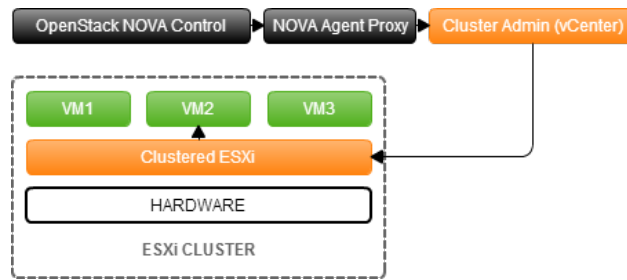


Figura 21: Integração de *ESXi* com *OpenStack* (modo *clustered*).

Os testes, em modo *clustered*, usando *vCenter*, não foram realizados por não existir uma licença livre disponível para utilização. A integração com *ESXi* não foi, no entanto, posta de parte, sendo necessário avaliar se as vantagens da sua utilização justificam os custos de licenciamento adicionais.

4

IMPLEMENTAÇÃO DE CLOUD PRIVADA COM OPENSTACK

Depois da análise possibilitada pelo pequeno sistema experimental, implementado na fase anterior, seguiu-se o planeamento e implementação de uma prova de conceito.

Importa lembrar que o objectivo da implementação deste sistema consistia na criação de um ambiente (o mais próximo possível daquilo que seria um ambiente de produção) que permitisse estudar a viabilidade da adopção do *OpenStack* por parte do *GSII*.

4.1 PLANEAMENTO

Na fase de planeamento foi necessário tomar um grande número de decisões técnicas. Foi necessário decidir a infra-estrutura física a utilizar, o método a utilizar no *deployment*, o software a utilizar para fazer face aos requisitos do *OpenStack* (base de dados, *message queues*, etc...), a organização dos serviços tendo em conta o *hardware* disponível, os diversos *back-ends* a utilizar, etc...

As decisões foram sempre tomadas tendo em conta os requisitos identificados no início do projecto: utilização de opções *open-source* sempre que possível e construção de uma arquitectura tolerante a falhas e altamente disponível.

4.1.1 Sistemas de Aprovisionamento e *Deployment*

A primeira decisão tomada disse respeito ao método de *deployment* a utilizar. Uma vez que o *OpenStack* tem uma arquitectura altamente complexa e requer a instalação e configuração de um grande número de módulos e serviços, decidiu-se começar por realizar uma avaliação das ferramentas *open-source* de *deployment* actualmente existentes [18]:

devstack: O *DevStack* é uma ferramenta sobretudo direccionada para a construção de ambientes de desenvolvimento *OpenStack*. Consiste basicamente num conjunto de *shell scripts*. Embora possa ser personalizada, é uma ferramenta bastante limitada no que toca a criação de ambientes mais elaborados.

mirantis fuel: Desenvolvida pela *Mirantis*, o *Fuel*¹ é uma ferramenta muito poderosa e de utilização bastante simples. Possui um cliente *web* e um *CLI*, que permitem, de forma rápida, aprovisionar servidores físicos via *Preboot eXecution Environment (PXE)*² e criar clusters *OpenStack*. Permite ainda fazer *deployment* de clusters *Ceph* e integrá-los com o *OpenStack*.

¹ <http://software.mirantis.com/>

² O *PXE* é um protocolo que permite o arranque de computadores a partir de uma interface de rede, utilizando um SO disponível remotamente. Não confundir com o *Wake-on-LAN (WOL)*, que permite apenas que se ligue um computador remotamente.

dell crowbar: O *Crowbar*³ é uma ferramenta especificamente desenhada para o *deployment* de *clusters OpenStack*. Combinando a utilização de *Puppet* com ferramentas de aprovisionamento *bare metal*, aparenta ser uma ferramenta extremamente versátil. Infelizmente, a documentação existente é quase nula, limitando-se quase em exclusivo ao repositório *GitHub* do projecto.

canonical juju: O *JuJu*⁴ é uma ferramenta, extremamente versátil, de automatização de *deployments*. Embora não seja especificamente desenhada para o efeito, é capaz de criar *clusters OpenStack* de elevada complexidade de forma fácil e rápida. Tem a grande desvantagem de só suportar, de momento, *Ubuntu*. Pode ser combinada com o *Canonical Metal-as-a-Service (MaaS)* para aprovisionar servidores físicos.

puppet: O *Puppet*⁵ é uma reputada ferramenta de gestão de configurações. Através da utilização de *puppet classes*⁶, é possível implementar, e gerir, qualquer tipo de *cluster*. Existe um repositório (*Puppet Forge*) que disponibiliza uma grande quantidade de *puppet modules* (conjunto de *puppet classes*) pré-definidos, para uma grande variedade de software, o que permite agilizar ainda mais o processo de *deployment*.

the foreman: O *Foreman*⁷ é um *Puppet External Node Classifier (ENC)*. Quer isto dizer que, o *Foreman* fornece um mecanismo (um dashboard web) através do qual se podem atribuir *puppet classes* a nós (servidores). Para além disto, o *Foreman* fornece ainda capacidades de *bare-metal provisioning*.

triple-o: O *OpenStack-on-OpenStack (Triple-O)* tem um conceito algo surpreendente: propõe que se automatize o *deployment* de *clusters OpenStack* utilizando... *OpenStack*! Tirando partido da existência de um *driver* do *Nova* que possibilita o aprovisionamento de servidores físicos e das capacidades de orquestração do *Heat*, a ideia consiste em criar uma primeira instalação *OpenStack*, muito simples (pode ser corrida num único servidor), que será depois utilizada para fazer o *deployment* do *cluster OpenStack* em si.

Desta análise, foi possível concluir que a utilização de uma ferramenta deste tipo tem claras vantagens, tanto ao nível do tempo despendido na implementação como à reprodutibilidade dos processos de *deployment*.

Devido às suas características, o *DevStack* (por ser orientado para ambientes de desenvolvimento), o *JuJu* (por ser exclusivo para *Ubuntu*), o *Puppet* (por não aprovisionar servidores físicos), o *Crowbar* (pela inexistente documentação) e o *Triple-O* (por ainda se encontrar num estágio muito precoce de desenvolvimento) foram descartados.

Após alguns testes feitos com o *Fuel*, percebeu-se que este assenta sobretudo na utilização de um conjunto de arquitecturas de referência (que foram amplamente testadas e se sabe funcionarem bem), não tendo ficado claro, no entanto, se este possui a flexibilidade necessária para suportar algumas personalizações que eventualmente serão necessárias (não foi possível perceber, por exemplo, se é de todo possível configurar a integração com *LDAP*).

³ <https://crowbar.github.io/home.html>

⁴ <https://juju.ubuntu.com/>

⁵ <http://puppetlabs.com/puppet/puppet-open-source>

⁶ Classes escritas em *puppet language* que definem, de forma declarativa, um conjunto de *software* a instalar e respectiva configuração.

⁷ <http://theforeman.org/>

Já o *Foreman*, devido à sua utilização do *Puppet*, é completamente personalizável. De tal forma que, a sua adopção, não faz só sentido no âmbito do *deployment OpenStack*, como também noutras áreas da operação do *GSII*. Podendo ser, por exemplo, utilizado na instalação e configuração de *software* em todo o parque informático que o *GSII* suporta (sendo que este não dispõe, actualmente, de nenhuma ferramenta para o efeito).

No entanto, e embora existam alguns pacotes base disponíveis, a criação de classes para toda a infraestrutura que se pretende montar, não só é extremamente difícil, tendo em conta o tempo disponível, como extravasa largamente o âmbito deste projecto.

Por este motivo, e uma vez que o principal objectivo do projecto era obter o maior conhecimento possível sobre o processo de operação e manutenção do *cluster OpenStack*, decidiu-se realizar o *deployment* manualmente, utilizando o *Foreman* apenas como ferramenta de aprovisionamento, possibilitando assim o rápido arranque, instalação e configuração básica do diverso equipamento via *PXE*.

4.1.2 Hardware

O hardware que, ao longo deste projecto, foi possível disponibilizar, por parte do *GSII*, foi o seguinte:

1 X SWITCH CISCO CATALYST 3550 com 24 portas FastEthernet

4 X SERVIDOR HP PROLIANT DL320 G5

1. 2 X Processador Intel Xeon 2.33GHz Quad-Core; 8GB RAM; 2 X 146GB SAS 10K RPM
2. Processador Intel Xeon 2.33GHz Quad-Core; 4GB RAM; 2 X 72GB SAS 10K RPM
3. Processador Intel Xeon 2.33GHz Dual-Core; 6GB RAM; 2 X 72GB SAS 10K RPM
4. Processador Intel Xeon 1.60GHz Quad-Core; 8GB RAM; 72GB SAS 10K RPM

1 X SERVIDOR HP PROLIANT DL320 G5P Processador Intel Xeon 2.66GHz Dual-Core; 2GB RAM; 2 X 72GB SAS 10K RPM

1 X SERVIDOR HP PROLIANT DL380 G5 2 X Processador Intel Xeon 2.33GHz Quad-Core; 8GB RAM; 4 X 146GB SAS 10K RPM

1 X SERVIDOR DELL POWEREDGE 1950 Processador Intel Xeon 1.60GHz Dual-Core; 2GB RAM; 2 X 73GB SAS 10K RPM

2 X WORKSTATION HP PRODESK Processador Intel Core i5 Quad-Core; 4GB RAM

1 X WORKSTATION HP COMPAQ Processador Intel Pentium 4 1.6GHz; 1GB RAM

Devido à quantidade de recursos relativamente limitada, decidiu-se que, caso se revelasse necessário, se criaria uma camada adicional de virtualização (usando *KVM*) entre algum do hardware e o software a instalar, promovendo assim uma melhor racionalização do hardware disponível.

Assim, decidiu-se desde logo reservar as duas *Workstations HP ProDesk* para a instalação de *software* auxiliar ao *cluster* (ferramenta de aprovisionamento, servidor *Network Time Protocol (NTP)*, servidor *DNS*, etc...) e a *Workstation HP Compaq*, devido às suas características muito limitadas, para realizar tarefas de gestão do sistema (acesso aos *dashboards*, conexão remota ao restante equipamento, etc...), tendo-se deixado a atribuição do restante *hardware* para uma fase mais avançada do planeamento.

4.1.3 Organização dos serviços *OpenStack*

No que diz respeito aos projectos *OpenStack*, decidiu-se não utilizar o *Neutron* (solução de *SDN*) nem o *Ceilometer* (solução de *metering*). O *Neutron* porque se considerou que é demasiado complexo e que a sua utilização não trás mais valias significativas em relação à utilização do *nova-network* (que tem uma arquitectura muito mais simples e mais facilmente configurável). E o *Ceilometer* por ainda se encontrar num estado algo precoce (ainda nem dispõe de um *dashboard*) e por ainda não ser claro se a sua utilização fará sentido no contexto de operação do *GSIIIC*, uma vez que as métricas que recolhe são fortemente orientadas para sistemas de cobrança ao utilizador.

Decidiu-se ainda instalar o *nova-network*, a par do *nova-compute*, nos nós de computação, porque, tal como se viu na análise das topologias de rede, esta é a forma de obter uma maior tolerância a falhas.

Relativamente aos restantes projectos (*Keystone*, *Glance*, *Cinder*, *Swift*, *Nova* e *Horizon*), tendo em vistas os objectivos de alta disponibilidade e tolerância a falhas, optou-se pela criação de *clusters*. Isto é, a instalação de várias instâncias de cada serviço, em servidores diferentes, e a utilização de um *load balancer*.

4.1.4 Base de dados

Depois de decidido o método de *deployment* e decididos os projectos a utilizar, o passo seguinte consistiu em escolher o software para fazer face aos requisitos do *OpenStack*.

A nível de base de dados, as características que se pretendia que a solução escolhida possuísse eram: ser *open-source*, ter elevada performance e possuir replicação de dados (redundância) e tolerância a falhas.

A escolha do motor de base de dados recaiu imediatamente sobre o *MySQL*, por ser uma tecnologia *open-source* com a qual o *GSIIIC* tem vasta experiência.

No entanto, para fazer face aos restantes requisitos, acima enunciados, optou-se pela utilização do *Percona XtraDB Cluster*⁸. Agregando num único pacote, *MySQL*, a biblioteca *Galera* e o *Percona XtraBackup*, o *Percona XtraDB Cluster* permite a criação de *clusters MySQL* dotados de alta disponibilidade (em configuração *Activa/Activa*) e replicação *Master/Master*.

Quer isto dizer que, a qualquer momento, todos os nós do cluster se encontram disponíveis para receber pedidos dos clientes, sendo que, no entanto, um pedido de escrita só é finalizado depois de a operação ter sido replicada por todo o *cluster*.

Desta forma, e como se pode ver na Figura 22, é possível ter várias instâncias *MySQL* simultaneamente activas e que se mantêm permanentemente sincronizadas entre si. A cada momento, a decisão sobre qual a versão cor-

⁸ <http://www.percona.com/software/percona-xtradb-cluster>

recta dos dados é tomada por *quorum* (daí a replicação ser *Master/Master* e não *Master/Slave*), devendo por isso o cluster ser composto por, pelo menos, três nós, por forma a evitar condições *split-brain*.

Uma condição *split-brain* acontece quando, após uma partição do *cluster* em dois ou mais fragmentos, com igual número de nós, não se consegue alcançar um consenso. Só é possível acontecer uma condição *split-brain* num *cluster* com um número par de nós.

Decidiu-se portanto pela criação de um *cluster* com três nós (para evitar condições *split-brain*), a serem instalados em três dos servidores *HP ProLiant DL320 G5*. Decidiu-se ainda usar estes servidores para instalar os *clusters* de serviços de gestão *OpenStack*, mencionados na secção anterior.

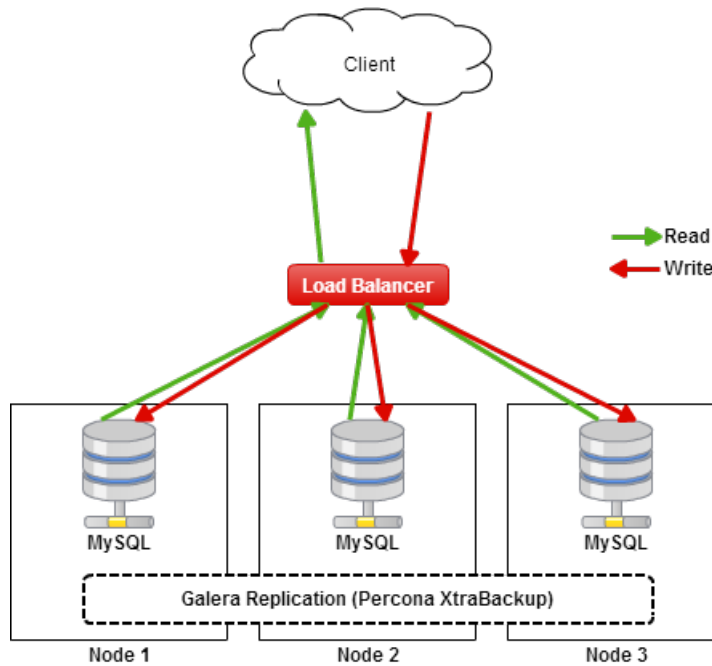


Figura 22: Arquitectura conceptual do *Percona XtraDB Cluster*.

4.1.5 Messaging Queue

Para possibilitar a comunicação entre diversos serviços, o *OpenStack*, que tem uma arquitectura *loosely coupled*, utiliza um *Message Oriented Middleware* (MOM) com base no protocolo AMQP. Embora existam diversas implementações do protocolo (*Apache Qpid*⁹, *Apache ActiveMQ*¹⁰, etc...), a solução imediatamente escolhida foi o *RabbitMQ*¹¹, por ser a opção por defeito do *OpenStack* e devido à sua arquitectura altamente vocacionada para HA.

Para fazer face às exigências de uma arquitectura altamente disponível, o *RabbitMQ* permite que várias instâncias sejam agregadas num *cluster* do tipo Activo/Activo e com replicação *Master/Master*. No entanto, embora seja feita a replicação dos *exchanges* (pontos de entrada de mensagens, que depois as encaminham para as *queues* apropriadas) e *bindings* (associação, a um *exchange*, de uma ou mais *queues*), por defeito, o *RabbitMQ* não faz replicação das próprias *queues*.

⁹ <https://qpid.apache.org/>

¹⁰ <https://activemq.apache.org/>

¹¹ <http://www.rabbitmq.com/>

Para tal, é necessário configurar o *mirroring*, que consiste numa estratégia de replicação *Master/Slave*. A cada momento existe apenas uma *queue* activa (*Master*), sendo que cada mensagem publicada é replicada para todas as outras (*Slaves*). Em caso de falha da *queue Master*, uma das restantes entra em actividade.

É possível ver uma ilustração destes conceitos na Figura 23, que ilustra o processo de publicação de uma mensagem.

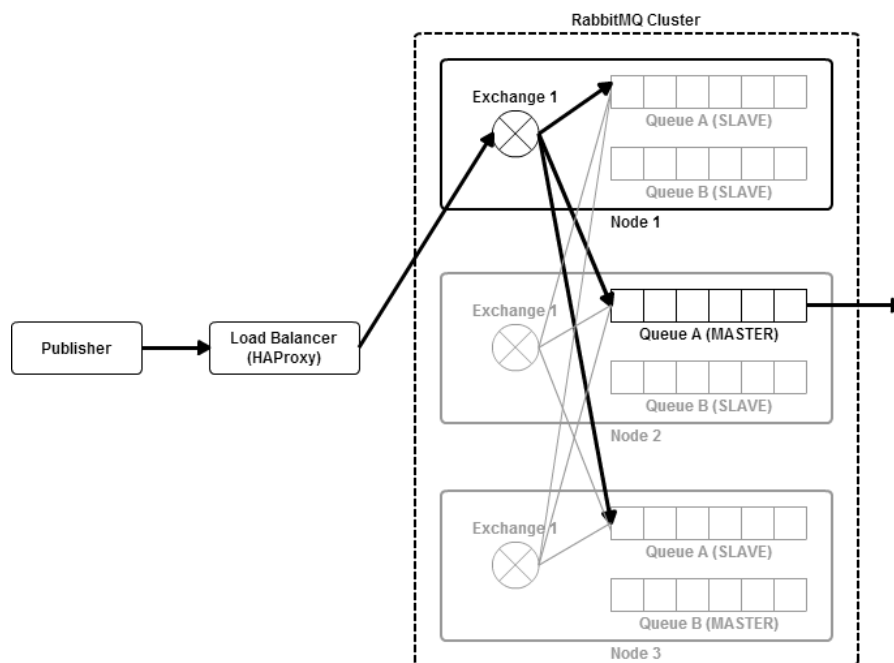


Figura 23: RabbitMQ cluster com mirrored queues - Publicação de uma mensagem.

Por uma questão de simplicidade, decidiu-se instalar o RabbitMQ nos mesmos servidores do cluster de base de dados e dos serviços de controle do OpenStack. Sendo que, portanto, se optou por criar um cluster com três nós.

4.1.6 Load Balancer

De forma a distribuir o tráfego pelos vários nós dos diversos clusters (base de dados, messaging queue, APIs do OpenStack, etc...), era necessária a utilização de um mecanismo de load balancing.

Uma vez que o GSIIC não dispunha de nenhum load balancer baseado em hardware, foi necessário procurar uma solução baseada em software. A escolha, por ser uma ferramenta com a qual já havia trabalhado anteriormente, recaiu sobre o HAProxy¹².

O HAProxy é uma ferramenta open-source de load balancing, proxying e HA, para serviços Transmission Control Protocol (TCP) (Layer 4) e HTTP (Layer 7).

Através de uma monitorização activa dos serviços, o HAProxy detecta possíveis falhas e direcciona automaticamente o tráfego para os restantes nós activos. Sendo portanto uma mais valia para alcançarmos alta disponibilidade e tolerância a falhas no nosso deployment.

O HAProxy suporta nove algoritmos de balanceamento diferentes, adequados a propósitos distintos:

¹² <http://haproxy.1wt.eu/>

ROUNDROBIN: Utiliza todos os servidores, alternadamente, de acordo com os seus pesos (sendo que o peso deve representar a capacidade de um servidor em relação aos restantes). É um algoritmo dinâmico, uma vez que a alteração do peso de um servidor tem repercussão imediata.

STATIC-RR: Semelhante ao anterior, mas é estático. Ou seja, a alteração do peso de um servidor não altera o escalonamento.

LEASTCONN: O pedido é encaminhado para o servidor que tiver um menor número de conexões activas. Em caso de empate, é utilizado *round robin*. Esta é a política mais indicada para conexões longas, como *LDAP* e *SQL*.

FIRST: O pedido é encaminhado para o primeiro servidor disponível. De notar que com este algoritmo, todos os pedidos serão encaminhados para o mesmo servidor, até que este atinja o número máximo de conexões.

SOURCE: É calculado o *hash* do *IP* de origem do pedido e dividido pelo número de servidores disponíveis. Este algoritmo garante (excepto em situações de falha) que todos os pedidos, de determinado cliente, serão sempre encaminhados para o mesmo servidor, sendo ideal para situações em que a mudança de servidor implique custos (por exemplo, conexões *statefull*).

URI: Semelhante ao anterior, sendo que o *hash* é calculado com base no *URI* do pedido e não no *IP*. Este algoritmo é frequentemente utilizado em *caches*, para maximizar a *hit rate*.

URL_PARAM : Utilizado apenas para serviços *HTTP*, este algoritmo utiliza um determinado parâmetro, presente no *URL*, para determinar o servidor de destino.

HDR: Semelhante ao anterior, mas em vez de um parâmetro do *URL*, utiliza um determinado *header* do pedido *HTTP*. Com o *header user-agent*, pode ser utilizado para encaminhar pedidos provenientes de *browsers* diferentes para servidores distintos.

RDP-COOKIE: Calculando o *hash* de determinado *cookie*, permite, por exemplo, garantir que todos os pedidos com o mesmo *session ID* são encaminhados para o mesmo servidor.

Analizadas todas estas políticas, decidiu-se utilizar a *leastconn* para o *cluster* de base dados e de *messaging queues* (devido a serem serviços com conexões longas) e o *source* para todos os restantes serviços (na sua maioria *APIs HTTP* que beneficiam do facto de os pedidos de determinado cliente serem sempre recebidos pelo mesmo servidor).

A utilização de *HAProxy* (ou qualquer outro *load balancer*) tem, no entanto, uma contrariedade. Uma vez que todo o tráfego, destinado aos serviços, deve ser canalizado através de si, este torna-se num *SPOF*.

Para superar esta contrariedade é necessário que o próprio *HAProxy* seja dotado de características de *HA* e tolerância a falhas. Ou seja, é necessário que exista um *cluster* de *HAProxy*.

Para isso, decidiu-se utilizar o *Keepalived*¹³. À semelhança do *HAProxy*, o *Keepalived* também é uma ferramenta *open-source* de *HA* e *load balancing*.

¹³ <http://www.keepalived.org/>

Usando *Virtual Router Redundancy Protocol (VRRP)*¹⁴, o *Keepalived* permite que se crie um *cluster* do tipo *Activo/Passivo* com duas instâncias de *HAProxy*. Um *Virtual IP (VIP)*¹⁵ é atribuído à instância *master* e, em caso de falha, o *Keepalived* migra automaticamente o *VIP* para a outra instância. Esta interação pode ser melhor percebida, através do diagrama da Figura 24.

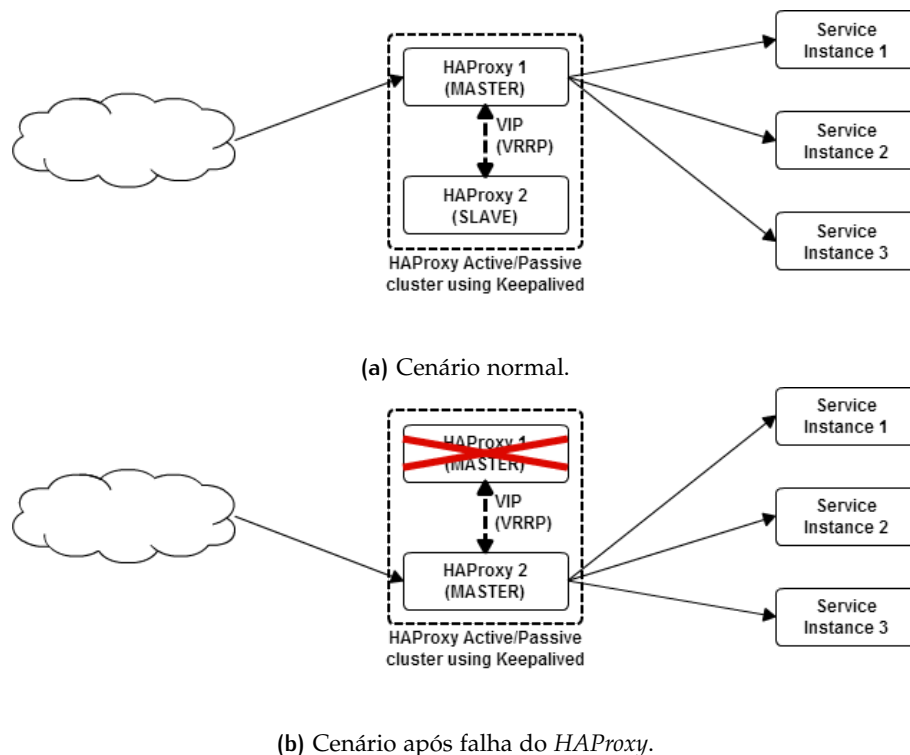


Figura 24: Arquitectura conceptual do HAProxy e Keepalived.

4.1.7 Backend(s) de armazenamento

Como já foi dito anteriormente, o *OpenStack* suporta um grande número de backends de armazenamento (para o *Cinder* e para o *Glance*), possuindo, inclusive, uma solução integrada de armazenamento baseado em objectos (*Swift*).

Durante o processo de tomada de decisão sobre o(s) *backend(s)* a utilizar (o *GSIIC* possui soluções de armazenamento *enterprise grade* de diversos fabricantes), foi-nos dado a conhecer [19] [20] uma solução *open-source*, relativamente recente, de armazenamento distribuído: o *Ceph*.

O *Ceph* é uma plataforma *open-source* de armazenamento distribuído, tolerante a falhas e altamente disponível. Utilizando hardware comum, o *Ceph* pretende ser uma solução unificada de armazenamento, disponibilizando, sobre uma camada base comum de armazenamento baseado em objectos (*RADOS*), quatro interfaces distintas:

librados: Uma interface de baixo nível para acesso directo ao *RADOS*.

radosgw: Uma *API HTTP* compatível com as do *Amazon S3* e *OpenStack Swift*.

¹⁴ O protocolo *VRRP* fornece *failover* dinâmico de endereços *IP*, de um *virtual router* para outro, em caso de falha.

¹⁵ Um *VIP* é um endereço *IP* que pode "flutuar" entre duas, ou mais interfaces de rede.

rbd: Interface que disponibiliza dispositivos de armazenamento block-based, que podem ser utilizados, por exemplo, para o armazenamento de VMs.

cephfs: Um sistema de ficheiros distribuído *POSIX-compliant*.

A sua arquitectura, que pode ser vista na Figura 25, consiste em três tipos de serviços, todos eles redundantes:

osd: O *Ceph Object Storage Daemon*, é um pequeno *daemon* responsável por gerir o armazenamento e disponibilização de objectos num disco físico. Deve existir uma instância por cada disco do *cluster*, podendo existir mais do que um em cada nó (servidor).

mon: O *Ceph Monitor Daemon* é o serviço responsável por manter e gerir a configuração e estado de todo o *cluster RADOS*.

mds: O *Ceph Metadata Server Daemon* é o *daemon* responsável pela manutenção do sistema de ficheiros do *CephFS*, armazenando a hierarquia de *directórios* e os meta-dados dos ficheiros.

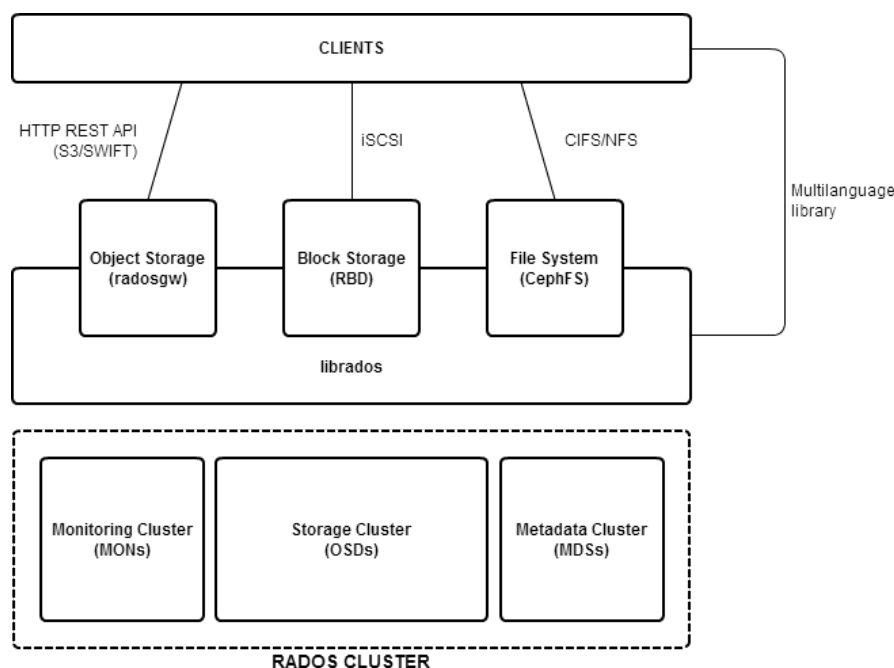


Figura 25: CPEH - Arquitectura conceptual.

O *Ceph* introduz ainda o conceito de *placement groups*, ilustrado na Figura 26, como forma de dissociar a organização dos objectos da organização dos discos físicos. Quando um objecto é adicionado ao *Ceph*, o algoritmo de posicionamento (*CRUSH*) atribui-lhe um *placement group*, que é depois mapeado para vários *OSDs* (tantos quanto o factor de replicação definido). Desta forma, o *Ceph* consegue de forma dinâmica reequilibrar automática e rapidamente o *cluster* em caso de falha de um nó.

Por ser uma ferramenta que visa substituir (ou pelo menos ser uma alternativa) as soluções de armazenamento comerciais, altamente dispendiosas, por uma solução que, utilizando hardware comum, permite construir *clusters* de armazenamento distribuídos, decidiu-se que seria muito interessante utilizar o *Ceph* como *backend* de armazenamento. Não só por ter uma

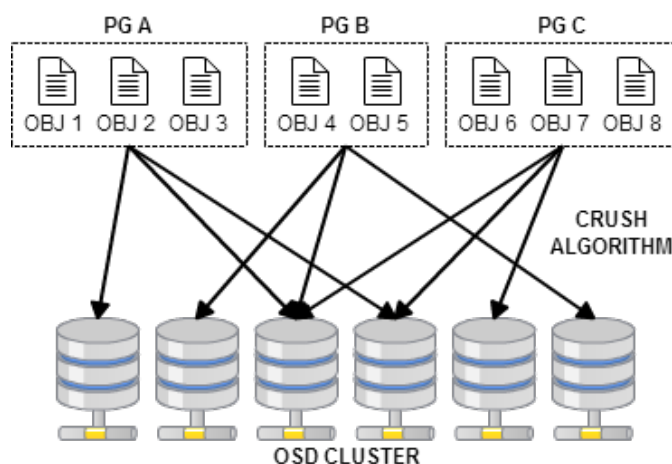


Figura 26: CPEH - Placement Groups (com factor de replicação 3) e algoritmo CRUSH.

grande capacidade de integração com o *OpenStack*, mas também por ser uma boa oportunidade de experimentar esta ferramenta, que poderá ter interesse para o GSIIC até mesmo fora deste contexto.

Uma vez que o *Ceph* é uma solução unificada de armazenamento, esta decisão tem a vantagem adicional de evitar que tenhamos de configurar e integrar com o *OpenStack* três *backends* distintos (para o *Glance*, *Cinder* e, no futuro, *Manila*). Facilitando assim o processo de *deployment* e centralizando a administração de todos os recursos de armazenamento.

4.1.8 Autenticação e Autorização

No capítulo da autenticação e autorização foi necessário sobretudo escolher os diversos *back-ends* e *plugins* a utilizar pelo *Keystone*.

Tal como foi dito na Secção 3.2.2, o *Keystone* actua como repositório e fornecedor de cinco tipos diferentes de informação: *tokens* (utilizados para evitar que o utilizador tenha de ser autenticado, utilizando as suas credenciais, repetidamente a cada pedido), *catalog* (catálogo de todos os serviços e *endpoints* *OpenStack* disponíveis), *assignment* (identifica os *roles* que cada utilizador possui em cada *tenant*), *identity* (identificação e autorização dos utilizadores e dos próprios serviços do *OpenStack*) e *policy* (regras passíveis de serem atribuídas a cada *role*).

Assim, em primeiro lugar, decidiu-se que era vantajoso utilizar a funcionalidade de *tokens* (que é opcional), uma vez que proporciona um processo de autenticação mais eficiente. Para a sua geração e disponibilização, decidiu-se utilizar o *plugin* para PKI, por se tratar da opção que oferece um mais elevado nível de segurança (uma vez que os *tokens* são assinados digitalmente) e o *backend memcached* (uma vez que armazena os *tokens* em memória, tem uma performance muito mais elevada do que um *backend* que os persista em disco) para o seu armazenamento.

No que diz respeito ao *catalog*, *assignment* e *identity*, optou-se pela utilização do *MySQL* enquanto *back-end*. Sendo que se definiu desde logo que, numa fase posterior, se iria migrar este último para um *back-end* LDAP.

Por fim, para *policy*, decidiu-se utilizar *rule files*¹⁶ (a opção por defeito do *Keystone*), uma vez que não se prevê, pelo menos para já, a necessidade de criação de regras adicionais às *standard*.

4.1.9 Caching

De forma a tentar otimizar o tempo de resposta de alguns serviços, optou-se pela utilização de um mecanismo de *caching*. Esta decisão revelou-se especialmente útil a evitar que determinados conteúdos web do *Horizon* (*dashboard*) tivessem de ser transferidos repetidamente e em particular no *Keystone*, onde a sua utilização para armazenamento de *tokens* permitiu tornar o processo de autenticação mais rápido e eficiente.

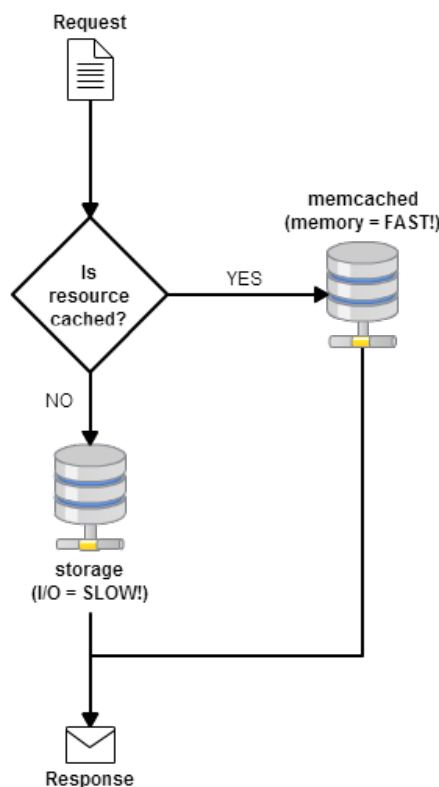


Figura 27: *memcached* - Fluxograma.

Para isto, decidiu-se utilizar o *memcached*¹⁷, uma vez que se trata de uma ferramenta *open-source* muito reputada [21] (alguns dos seus utilizadores são o *Twitter*, *Wikipedia* e *Youtube*) e que é amplamente suportada pelo *OpenStack*.

A principal razão para o ganho de performance, obtido com a utilização do *memcached*, prende-se com o facto de este utilizar apenas a memória para armazenar os objectos da *cache*. Assim, como se pode ver na Figura 27, a sua recuperação é muito mais rápida uma vez que evita que tenha de ser feito o pedido ao serviço de destino.

Uma vez que o *memcached* não tem nenhum mecanismo que lhe permita replicar os dados através de diversos nós, não fazia sentido utilizar um *cluster* Activo/Activo, uma vez que cada instância iria ter um conjunto de

¹⁶ Ficheiros que descrevem regras personalizadas em formato *JavaScript Object Notation* (JSON).

¹⁷ <http://memcached.org/>

objectos diferentes na *cache*, levando, por sua vez, à existência de um baixo *hit rate*.

Assim, optou-se por criar um *cluster* em configuração Activo/Passivo, com apenas um nó *memcached* em funcionamento e dois outros de *backup*.

4.1.10 Arquitectura de rede

No seguimento da análise, realizada na Secção 3.3, às topologias de rede suportadas pelo *nova-network*, optou-se pela utilização do *driver VlanManager*. O facto de realizar segmentação de rede utilizando *vLANs*, abordagem fortemente utilizada pelo *GSIIIC*, fez com que a considerássemos a alternativa mais apropriada.

Decidiu-se ainda optar pelo modo de operação *multi-host*, que descentraliza a gestão da rede por diversas instâncias do *nova-network*, de forma a optimizarmos a tolerância a falhas.

De seguida, fez-se uma análise dos segmentos em que se deveria dividir a rede (utilizando como critério o tipo e o volume de tráfego) e chegou-se à seguinte estrutura, cujo diagrama pode ser visto na Figura 28:

vLAN 701 - ADMINISTRAÇÃO: A utilizar no processo de *deployment* e para operação e manutenção do sistema. Este será um segmento com pouca actividade, mas que, devido à sua natureza, faz sentido estar isolado.

vLAN 702 - CONTROLO: *vLAN* dedicada à comunicação entre os vários componentes dos *OpenStack* responsáveis pela orquestração do *cluster*, o que inclui o tráfego respeitante às bases de dados e às *queues AMQP*.

vLAN 703 - ARMAZENAMENTO: *vLAN* a utilizar em todas as tarefas que envolvam o acesso ao(s) *backend(s)* de armazenamento. De todos, será talvez aquele com tráfego mais avultado.

vLAN 704 - PÚBLICA/floating ips: *vLAN* onde estará alocada toda a gama de endereços públicos (acessíveis a partir do exterior) do *cluster*. Nomeadamente os *Floating IPs* a atribuir dinamicamente às *VMs* e os endereços públicos a utilizar pelos serviços do *OpenStack* expostos ao exterior (dashboard, etc...).

vLANs [705-707] - PRIVADA: Gama de *vLANs* utilizar na comunicação entre *VMs* e no acesso ao exterior (utilizando o *IP* privado e *NAT*). Cada *tenant* utilizará uma *vLAN* dedicada por forma a existir isolamento de tráfego entre *tenants*.

Devido às características do segmento de rede a utilizar na instalação (uma sub-rede isolada do resto da rede do *GSIIIC*), achou-se oportuno instalar localmente alguns serviços auxiliares. Nomeadamente, decidiu-se instalar um servidor *DNS* (*dnsmasq*), por forma a poder identificar as máquinas pelos seus *hostnames*, um servidor *NTP* (*ntpd*¹⁸), de forma a manter os relógios das máquinas sincronizados, e um *software NAT router*, para que se pudesse racionalizar a utilização dos poucos endereços *IPs*, com conectividade ao exterior, disponíveis.

Decidiu-se ainda instalar um servidor *VPN* (*OpenVPN*¹⁹), para que se pudesse aceder remotamente à rede do projecto, evitando o inconveniente de trabalhar no mesmo espaço físico que o equipamento.

¹⁸ <http://doc.ntp.org/4.1.0/ntpd.htm>

¹⁹ <https://openvpn.net/>

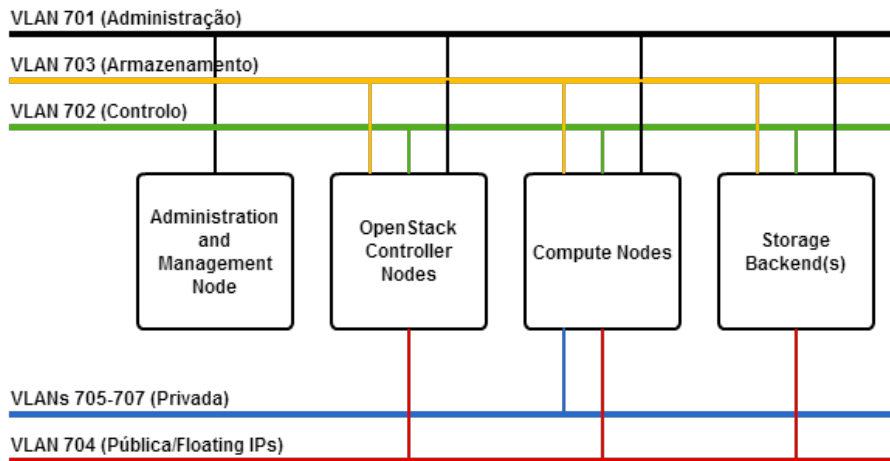


Figura 28: Diagrama da arquitectura de rede.

4.2 DESCRICHÃO DO AMBIENTE A INSTALAR

Depois de tomadas as decisões acima descritas, foi possível então definir a arquitectura global do ambiente a instalar.

Nesta arquitectura, que pode ser vista na Figura 29, tentou-se eliminar todos os *SPOF*, de forma a termos um sistema o mais robusto possível.

Assim, temos um *cluster* Activo/Activo de serviços OpenStack, utilizando *clusters* de base de dados e de *messaging queues* Activo/Activo com replicação *Master/Master* para a persistência de dados e intercomunicação. É ainda tirado partido de um mecanismo de *caching* implementado por um *cluster* Activo/Passivo de *memcached*. A carga destes *clusters* é balanceada por uma instalação de *HAProxy*, ela própria redundante (utilizando *Keepalived*).

Como *backend* de armazenamento para o *Glance* e *Cinder* é utilizado o *Ceph*, um sistema distribuído e altamente disponível de armazenamento.

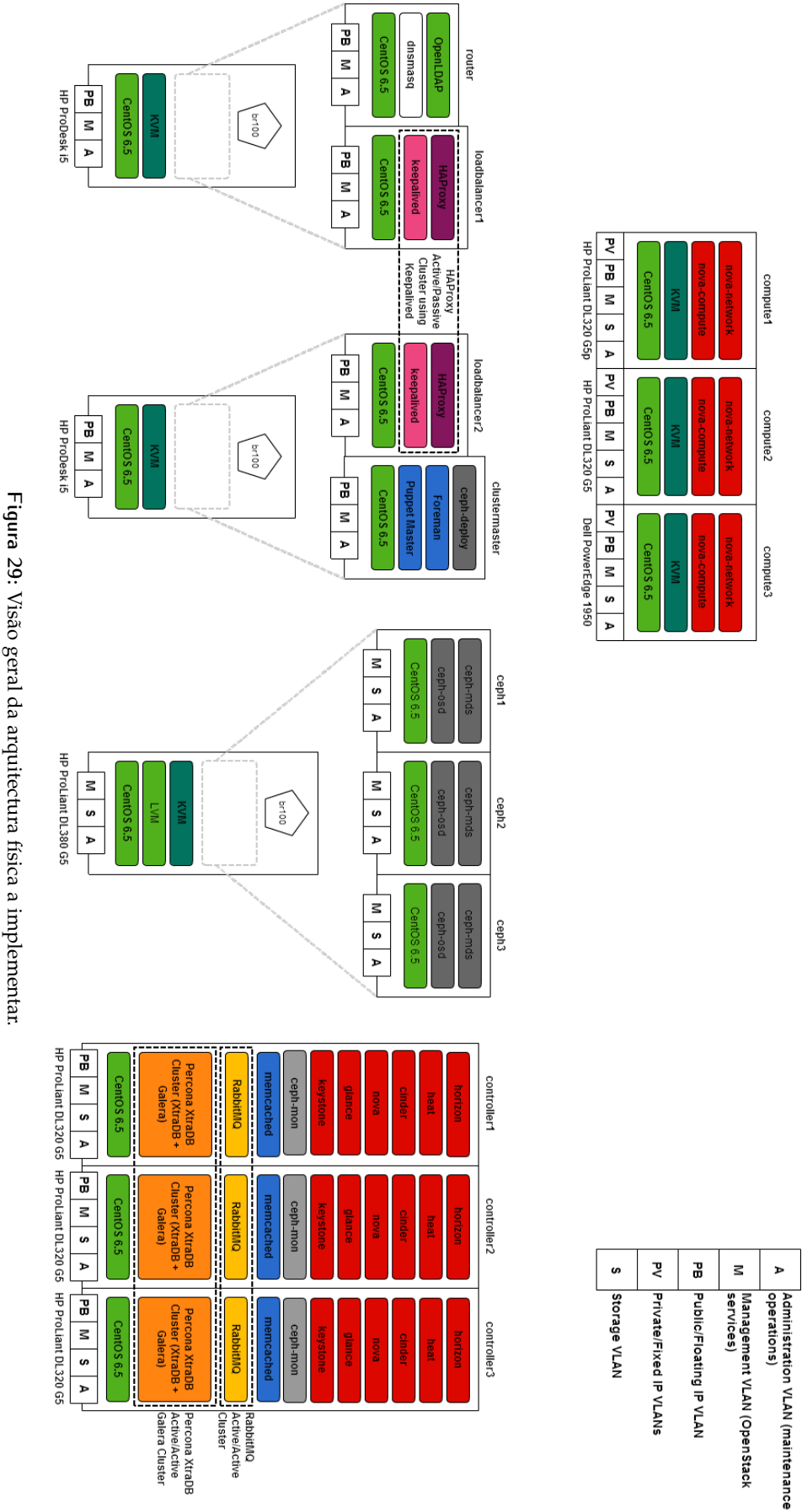
Temos ainda três nós de computação, que suportam um total de cerca de dez *VMs*, com serviço de rede dedicado (*nova-network multi-host*) que evita que a falha de um nó afecte a conectividade dos restantes. O *hypervisor* utilizado é *KVM*, sendo a sua integração com o OpenStack feita via *libvirt*.

Por fim, ao nível da segurança, temos uma arquitectura de rede particionada, que permite o isolamento de tráfego entre *tenants*, *APIs* dos serviços com *SSL*, e *tokens* assinados digitalmente via *PKI*.

4.3 INSTALAÇÃO

Terminada fase de planeamento, iniciou-se a implementação da arquitectura desenhada. Utilizou-se uma abordagem *bottom-up*, tendo-se começado por preparar toda a estrutura física e serviços auxiliares, evoluindo depois para a instalação do OpenStack em si.

Não é, de todo, a intenção desta secção fazer uma descrição técnica detalhada dos passos utilizados na instalação dos diversos serviços (essa informação pode ser vista nos anexos), mas sim uma breve descrição de alto nível dos procedimentos realizados e dos problemas encontrados e respectivas soluções.



controller1

horizon

heat

cinder

nova

glance

keystone

ceph-mon

memcached

RabbitMQ

Percona Xtradb Cluster (Xtradb + Galera)

CentOS 6.5

PB M S A

controller2

horizon

heat

cinder

nova

glance

keystone

ceph-mon

memcached

RabbitMQ

Percona Xtradb Cluster (Xtradb + Galera)

CentOS 6.5

PB M S A

controller3

horizon

heat

cinder

nova

glance

keystone

ceph-mon

memcached

RabbitMQ

Percona Xtradb Cluster (Xtradb + Galera)

CentOS 6.5

PB M S A

RabbitMQ

Active/Active Cluster

Percona Xtradb

Active/Active Galera Cluster

A	Administration VLAN (maintenance operations)
M	Management VLAN (OpenStack services)
PB	Public/Floating IP VLAN
PV	Private/Fixed IP VLANs
S	Storage VLAN

Figura 29: Visão geral da arquitectura física a implementar.

4.3.1 Preparação da estrutura de rede

O primeiro passo do processo de instalação consistiu na configuração do equipamento de rede.

Primeiro de tudo, foi necessário reservar um conjunto de endereços *IP* e a gama de *vLANs* identificadas na fase de planeamento. De seguida foi necessário configurar o *switch*, por forma a que as portas a utilizar transportassem as *vLANs* necessárias em modo *trunk* e que a *vLAN* de gestão fosse definida como *default*. A configuração completa pode ser vista no anexo B.

Configurado o *switch*, foi necessário instalar manualmente o *CentOS*, numa das *workstations HP ProDesk*, seguido do *KVM* por forma a poder criar uma *VM* a utilizar na instalação do diverso software de rede auxiliar.

Nessa *VM*, foi então instalado o *dnsmasq* (Anexo C), configurada a *firewall* e habilitado o *IP Forwarding* (Anexo D), por forma a que todas as máquinas pudessem ser identificadas pelo seu *hostname* e para que pudessem aceder ao exterior (para instalação de software, etc...), ainda que encontrando-se num segmento de rede isolado.

Foi ainda necessário criar um servidor local de *NTP* (*ntpd*), cuja instalação pode ser vista no Anexo E, para que todas as máquinas pudessem manter os seus relógios sincronizados.

4.3.2 Aprovisionamento do hardware

O passo seguinte consistiu em instalar o sistema operativo e *KVM* na outra *workstation HP ProDesk*, para que se pudesse criar uma *VM* a utilizar pelo software de aprovisionamento e deployment.

Assim, nesta *VM*, foi instalado o *Puppet* e o *Foreman* (Anexo G), de forma a permitir aprovisionar automaticamente todas as outras máquinas (físicas e virtuais) do ambiente.

A instalação destas duas ferramentas foi bastante simples, tendo-se optado por utilizar os repositórios binários do *CentOS*.

Para auxiliar no processo de configuração da funcionalidade de aprovisionamento do *Foreman* foi utilizado o *Foreman Setup Plugin*.

Aquando do aprovisionamento das primeiras máquinas, fomos confrontados com um problema, durante a obtenção dos pacotes para a instalação do SO. Este erro era provocado por o *script* de instalação do *CentOS* não conseguir validar o certificado *SSL* dos repositórios, devido à hora do sistema estar errada. Para contornar este problema, foi necessário alterar o *template* de aprovisionamento do *CentOS* no *Foreman*, por forma a que a sincronização do relógio ocorresse logo no início do processo de pré-instalação.

4.3.3 HAProxy & keepalived

De seguida, foram criadas duas novas *VMs*, uma em cada uma das *workstations HP ProDesk*, já utilizando o *Foreman* para o processo de aprovisionamento, a utilizar para a instalação do *cluster HAProxy*.

Primeiro, foi necessário instalar o *Keepalived* e configurar os *virtual routers* e o *VIP* a serem utilizados. De seguida-se procedeu-se à instalação do *HAProxy* e configuração do *dashboard* e dos diversos *clusters* a serem geridos por este.

De notar que, grande parte dos *clusters* foram sendo configurados progressivamente, à medida que os serviços correspondentes foram sendo instalados, e não nesta fase.

Os procedimentos, detalhados, de instalação e configuração destes dois sistemas, podem ser vistos no Anexo I.

4.3.4 Percona XtraDB Cluster

Para o *Percona XtraDB Cluster*, provisionaram-se três dos servidores *HP ProLiant DL320 G5* via *Foreman*, tendo-se, de seguida, instalado e configurado o *Percona*, utilizando o repositório binário oficial, de acordo com as instruções descritas no Anexo H.

Criou-se também a estrutura de tabelas utilizadas internamente pelo *MySQL*, os diversos utilizadores e respectivas permissões e tornou-se a instalação mais segura, removendo os utilizadores e tabelas de teste.

Por fim, foi ainda necessário criar um *script*, usando *xinetd*²⁰, de monitorização do estado do nó em relação ao *cluster*, a ser utilizado pelo *HAProxy*.

4.3.5 RabbitMQ

A instalação do *RabbitMQ* foi um processo bastante simples, como se pode ver no Anexo J, tendo-se instalado uma instância em cada um dos servidores onde já havia antes sido instalado o *Percona*.

A configuração, no entanto, foi uma tarefa muito mais complicada. Após o arranque das três instâncias, constatou-se que estas não conseguiam comunicar entre si, de forma a estabelecerem o *cluster*. Depois de algum *debug* infrutífero ao nível da camada de rede, chegou-se à conclusão que o problema se devia ao facto de se ter descurado um dos passos da configuração.

Assim, foi necessário criar e sincronizar *erlang magic cookies*²¹ entre todos os nós, por forma a que as várias instâncias pudessem comunicar entre si.

Estabelecido o *cluster*, foi ainda necessário definir a política de *HA* para utilizar *mirrored queues* e configurá-las para sincronizar automaticamente em caso de reinício do nó.

Achou-se ainda apropriada a configuração do *RabbitMQ Management Plugin* (*dashboard* web de gestão), o que exigiu a criação de um utilizador de administração e definição das permissões adequadas, por forma a poder facilmente monitorizar o *cluster*.

4.3.6 memcached

O *memcached* foi, de todos, o serviço mais fácil de instalar. Foi apenas necessário instalar os pacotes adequados (a partir do repositório *Extra Packages for Enterprise Linux (EPEL)*), não se tendo procedido a qualquer configuração, uma vez que a existente por defeito (tamanho da *cache* e número máximo de conexões) se afigurou adequada e adicionar o *cluster* (com um nó activo e dois de *backup*) ao *HAProxy*.

4.3.7 Ceph

Já o *Ceph*, devido ao seu grande número de componentes, foi talvez o mais complexo.

²⁰ O *xinetd* (*extended internet daemon*) é um *daemon* existente em sistemas *UNIX*, que monitoriza pedidos *TCP* recebidos em determinadas portas e inicia os serviços correctos para os receberem.

²¹ Um *magic cookie* é um mecanismo de baixo nível utilizado pelo *erlang* (linguagem de programação orientada para sistemas em tempo real distribuídos utilizada pelo *RabbitMQ*) para gerir as permissões da comunicação entre processos.

Primeiro de tudo, foi necessário instalar a ferramenta *ceph-deploy*, para auxiliar o processo de instalação.

Depois, utilizando esta ferramenta foram criados o *cluster* de nós de monitorização (*MON*), nos três servidores *HP ProLiant DL320 G5*, tendo sido necessário, primeiro, configurar em todos eles o acesso por *Secure Shell (SSH)* sem password.

Para os *clusters* de armazenamento (*OSD*) e meta-dados (*MDS*), aprovionou-se o servidor *HP ProLiant DL380 G5* usando o *Foreman* e configuraram-se os seus discos de forma a dispormos de nove volumes lógicos (para além daquele onde foi instalado o *SO*). De seguida, instalou-se o *KVM* e criaram-se três *VMs*, aprovionada via *Foreman*, cada uma delas com três dos volumes configurados anteriormente. Um deles destinado ao *SO* e os dois restantes a serem utilizados pelo *Ceph* para armazenamento.

Mais uma vez utilizando o *ceph-deploy*, criou-se os *cluster OSD* e *MDS* nestas três máquinas virtuais, tendo sido criadas três instâncias *MDS* (uma por cada *VM*) e seis *OSD* (uma por cada disco das três máquinas).

Foi ainda necessário alterar algumas configurações, nomeadamente o factor de replicação, que foi alterado para três (devem existir três réplicas de cada objecto), e o número de *placement groups* (utilizando uma fórmula recomendada na documentação do *Ceph*).

O procedimento de instalação detalhado pode ser encontrado no Anexo [L](#).

4.3.8 Serviços de gestão OpenStack

Em relação aos serviços de gestão do *OpenStack*, estes foram instalados nos nós de controlo (os mesmos da base de dados e *queues*), utilizando o repositório binário disponibilizado pela *Red Hat*.

Antes da instalação, foi necessário criar e inicializar as base de dados a utilizar por cada um dos serviços.

O primeiro foi o *Keystone* (Anexo [M](#)), tendo sido configurado para utilizar *SQL* e *memcached* como *backends* e o driver de *PKI* nos *tokens* (para o qual foram gerados certificados digitais). Depois de instalado, foi necessário criar no *Keystone* utilizadores aplicativos para os restantes serviços e adicioná-los, bem como aos seus endpoints, ao catálogo.

O serviço seguinte foi o *Glance*, tendo-se instalado o *glance-api* e o *glance-registry* (responsável por manter o catálogo de imagens). Foi configurado o *Ceph* como *backend*, tendo, para tal, sido necessário criar uma *pool*, dedicada ao armazenamento das imagens, e um utilizador no *CephX* (serviço de autenticação do *Ceph*) para o *Glance*.

Seguiram-se os serviços de controle do *Nova*, tendo sido instalados o *nova-api* (implementa várias *APIs* suportadas), o *nova-cert* (utilizado pela *API* compatível com a da *Amazon EC2*), o *nova-console-auth* (responsável pela autenticação nos acessos às *VMs* via *NoVNC* ou *SPICE*), o *nova-scheduler* (responsável pelo processo de agendamento dos nós de computação), o *nova-conductor* (que permite que os nós de computação possam aceder à base de dados de forma *decoupled*) e o *nova-novncproxy* (que fornece uma *proxy* para acesso via *NoVNC* às *VMs*). A sua instalação, cujas instruções podem ser vistas no Anexo [P](#), foi bastante simples.

Por fim, o *Cinder*. Foram instalados, como se pode ver com mais detalhe no Anexo [O](#), o *cinder-volume*, o *cinder-backup* (permite efectuar *backups* de volumes), o *cinder-scheduler* (utilizado em ambientes *multi-backend*) e o *cinder-api*. O *Ceph* foi configurado como *backend* quer de volumes quer de *backups*,

tendo sido necessário criar uma *pool* e um utilizador no *CephX* para cada um deles.

4.3.9 *nova-compute* & *nova-network*

Relativamente ao *nova-compute* e *nova-network*, uma vez que estes serviços correm nos nós de computação, a primeira tarefa foi aprovisionar os servidores (HP Proliant DL320 G5, HP Proliant DL320 G5p e Dell PowerEdge) a utilizar.

Seguiu-se a instalação do *KVM* e *libvirt*, e a substituição do *qemu-img* (componente do *libvirt*) por uma versão compatível com *Ceph*.

Foi então instalado o *nova-compute*, tendo a sua configuração exigido a criação de chaves a utilizar pelo *libvirt* no acesso aos volumes armazenados no *Ceph*.

Por fim, foi feita a instalação e configuração do *nova-network*, tendo-se criado as *bridges* a utilizar pelas VMs e configurado o *driver VlanManager*.

4.3.10 *Horizon*

Para a instalação do *Horizon*, uma vez que este se trata de uma aplicação web, foi primeiro necessário instalar o servidor *HTTP Apache*.

Seguiu-se a instalação dos pacotes do *Horizon*, bem como o módulo de integração com *memcached*.

Relativamente à configuração, foi apenas necessário alterar o caminho *URL* da página de *login*, que por defeito vinha incorrectamente configurado e alterar as configurações do *Apache*, de forma a que o endereço de acesso ao *Horizon* fosse <http://c0/> e não <http://c0/horizon>.

As instruções detalhadas podem ser vistas no Anexo [Q](#).

4.4 INTEGRAÇÃO COM SERVIÇOS EXISTENTES

Terminada a implementação do sistema, passou-se à fase de integração com alguns dos serviços existentes no *GSIIC*. Optou-se por fazer a integração com o serviço de directório, de forma a possibilitar a utilização, no *OpenStack*, das mesmas credenciais que os utilizadores utilizam nos restantes serviços, e com o serviço de monitorização, de forma a possibilitar a monitorização de toda a infraestrutura criada (serviços *OpenStack*, *cluster MySQL*, *Ceph*, etc...) conjuntamente com a restante infraestrutura do *GSIIC*.

A integração com as soluções de armazenamento comerciais existentes no *GSIIC*, embora estivesse inicialmente planeada, não foi executada, por ter sido considerada uma operação demasiado sensível para a maturidade do projecto.

4.4.1 Serviço de directório

O *Lightweight Directory Access Protocol (LDAP)* é um protocolo *open standard* para acesso a serviços de directório.

No caso do *GSIIC*, este serviço de directório é implementado pelo *OpenLDAP*²² e centraliza os serviços de identificação e autenticação de um grande

²² <http://www.openldap.org/>

número de sistemas da Universidade (servidores de *email*, *Inforestudiante*, rede *wireless*, etc. ...).

Fazia, portanto, todo o sentido que o novo serviço de computação em *cloud* também tirasse partido deste directório, para que os utilizadores lhe pudessem aceder utilizando as mesmas credenciais que utilizam para os restantes serviços.

O *Keystone* contempla a utilização do *LDAP*, enquanto *back-end*, para duas funções distintas: autenticação (verificação das credenciais de um utilizador) e autorização (validação do *role* e respectivas permissões de um utilizador em determinado *tenant*).

Decidiu-se utilizar a integração com *LDAP* apenas na vertente de autenticação e em modo *read-only*, uma vez que, para autorização, a criação e atribuição de *tenants* e de *roles* teria de ser realizada via *LDAP*, "poluindo" assim o directório com entradas úteis apenas no contexto do *OpenStack*. Para tal foram adicionados/alterados os seguintes parâmetros no ficheiro de configuração do *Keystone*:

```
[ldap]
url = ldap://router
user =
password = None
suffix = dc=ci,dc=uc,dc=pt
use_dumb_member = False
allow_subtree_delete = False
page_size = 0
query_scope = sub
user_tree_dn = ou=rh,ou=users,dc=ci,dc=uc,dc=pt
user_objectclass = person
user_id_attribute = uid
user_name_attribute = uid
user_mail_attribute = mailLocalAddress
user_pass_attribute = userPassword
user_enabled_attribute = sn
user_enabled_default = True
user_allow_create = False
user_allow_update = False
user_allow_delete = False
```

Uma vez que o *OpenStack*, pelo menos na edição actual, não suporta *back-ends* de autenticação distintos para as contas de utilizador e para as contas aplicacionais, era necessário criar contas para diversos serviços (*Glance*, *Cinder*, *Nova*, etc. ...) no directório. Por este motivo, e por questões de segurança, optou-se pela instalação do *OpenLDAP*, na máquina onde já haviam sido instalados o *dnsmasq* e o *OpenVPN*, e pela criação de um *mirror* local do directório.

Com a integração com *LDAP*, a utilização de *tokens* e respectivo *caching* tornou-se ainda mais importante, uma vez que o processo de validação de credenciais passou a ter um custo ainda mais elevado.

Devido à quantidade maciça de utilizadores existentes no directório da Universidade e por questões de performance, foi ainda necessário implementar o seguinte filtro na configuração do *Keystone*, de forma a que fossem "visíveis" apenas os utilizadores da equipa do *GSIIC*:

```
user_filter = (memberof=cn=gestor_ciuc,ou=groups,dc=ci,dc=uc,dc=pt)
```

A instalação pormenorizada do *OpenLDAP* e criação do *mirror* pode ser vista no Anexo T.

Adicionalmente, aproveitou-se ainda para configurar o *Foreman* de forma a utilizar também o *LDAP* como mecanismo de autenticação. A sua configuração foi trivial tendo sido realizada via *Foreman dashboard*.

4.4.2 Serviço de monitorização

Actualmente, o *GSII* possui dois sistemas de monitorização em funcionamento. O *Icinga*²³ e o *Zabbix*²⁴, sendo que se encontra em fase de transição de toda a infraestrutura para este último. Fazia portanto todo o sentido que a integração fosse feita com a nova plataforma.

O *Zabbix* é uma ferramenta de monitorização de classe empresarial. Permite monitorizar a disponibilidade de dispositivos (via *Simple Network Management Protocol (SNMP)*, *SSH* e *Windows Management Instrumentation (WMI)*) e serviços (via *HTTP*, *SSH*, etc. . .), recolha de métricas de utilização de recursos (utilização de *CPU*, memória, disco, etc. . .) e gestão de eventos (alertas) baseados em regras (*triggers*).

À semelhança do que aconteceu com o *LDAP*, por questões de segurança e uma vez que o servidor *Zabbix*, em produção no *GSII*, desempenha um papel crítico nas operações diárias, optou-se por criar um servidor local. Os procedimentos detalhados da sua instalação e configuração podem ser vistos no Anexo U.

A integração com o *Zabbix* foi feita a vários níveis. Primeiro, ao nível do hardware, tendo-se configurado o *zabbix-agent* para recolher métricas de utilização de recursos. Foi ainda configurada, via *SNMP*, a monitorização do *switch*.

Depois ao nível dos serviços auxiliares, tendo sido configurada a monitorização do *cluster* de base de dados (utilizando um *template* para *Zabbix* disponibilizado pela *Percona*), do *Ceph* e do *cluster RabbitMQ* (tendo sido utilizados *templates* desenvolvidos pela comunidade). O *memcached* e *HAProxy* não são directamente monitorizados, uma vez que não se conseguiu encontrar nenhum *template* compatível com versão do *Zabbix Server* em utilização.

Por fim, ao nível dos serviços do *OpenStack*, tentou fazer-se a instalação de um *template* ainda em fase experimental, mas não se conseguiu que este funcionasse apropriadamente.

Assim, embora não se tenha conseguido encontrar *templates* para todos os serviços, conseguiu-se abranger uma quantidade razoável de pontos de monitorização.

A instalação e configuração detalhada do *Zabbix* pode ser encontrada no Anexo U.

²³ <https://www.icinga.org/>

²⁴ <http://www.zabbix.com/>

5

VALIDAÇÃO E ANÁLISE DO COMPORTAMENTO DO SISTEMA

Terminada toda a implementação e configuração do ambiente, a fase seguinte consistiu na validação da instalação *OpenStack* e na análise do comportamento dos diversos sub-sistemas.

Para a validação da instalação *OpenStack* fizeram-se testes de integração, utilizando uma ferramenta especificamente desenhada para o efeito. Seguiram-se testes à resiliência do sistema, tendo-se analisado o comportamento dos sub-sistemas em situações de falhas e a sua capacidade de recuperação. Por fim, foram avaliadas, de forma funcional, as capacidades de escalabilidade.

Dada a rápida evolução do *OpenStack* (são feitas *releases* a cada seis meses), achou-se ainda pertinente testar a dificuldade do processo de *upgrade*.

5.1 VALIDAÇÃO DO *deployment*

O primeiro passo da fase de testes, consistiu em validar o *deployment*. Isto é, realizar testes de integração à implementação, de forma a garantir que todos os seus componentes foram correctamente instalados e configurados e que se encontram em perfeito funcionamento.

Para tal, decidiu-se utilizar a ferramenta *Tempest* (*The OpenStack Integration Test Suite*)¹. Parte integrante do projecto *OpenStack*, o *Tempest* consiste num conjunto de *scripts* que permitem validar rapidamente *clusters OpenStack*, de qualquer dimensão.

O processo de instalação e configuração pode ser visto no Anexo V.

Após a instalação, o *Tempest* foi utilizado para correr uma enorme bateria de testes contra os serviços instalados (o *Tempest* suporta muitos serviços que ainda não fazem parte da distribuição *OpenStack*)

A primeira sessão de testes, resultou num grande número de erros, em grande parte relacionados com o *Keystone*. Depois de corrigidos vários deles (sobretudo relativos à integração com *LDAP*) e várias repetições dos testes, um número considerável de erros, aparentemente intermitentes, persistiu.

Depois de um longo processo de *debug* e ajuda de alguns membros do canal *#openstack* do *Internet Relay Chat (IRC)*, o problema foi identificado: a funcionalidade de *tokens PKI* do *Keystone* havia sido mal configurada. Os certificados utilizados pelo *Keystone*, para a assinatura dos *tokens*, foram configurados independentemente em cada um dos nós, fazendo com que um *token* gerado por uma das instâncias do *Keystone* não fosse reconhecido como válido pelas restantes. A intermitência dos erros é explicada pelo facto de utilizarmos *load balancing*: umas vezes a geração do *token* e a sua validação acontecem num mesmo nó (o que é infrequente), e outras vezes acontece em nós diferentes (produzindo o erro mencionado). Para contornar este problema, optou-se por mudar os *tokens* para *UUID*, que embora mais

¹ <http://docs.openstack.org/developer/tempest/>

inseguros são muito mais fáceis de configurar. Tendo-se deixado o retorno aos *tokens PKI* para uma fase posterior do projecto (em que se deverá utilizar os certificados reais do GSIIC).

Ultrapassado este problema, os testes foram todos concluídos com sucesso.

No final, fizeram-se ainda manualmente alguns *smoke tests* através do *dashboard* (criação de VMs, imagens, volumes, etc...). Estes testes, embora de natureza algo limitada, permitiram descobrir um outro problema: quando se tentava criar um novo projecto, o *dashboard* "congelava", obrigando a um *refresh* da página. O erro, mais uma vez, só foi descoberto após vários pedidos de ajuda no canal *#openstack*: o elevado número de utilizadores no directório *LDAP* (todos os utilizadores da Universidade) provocavam um *timeout* no código *Javascript* responsável por criar a lista de utilizadores passíveis de serem adicionados ao projecto, que se estava a criar. Tal como referido na Secção 4.4.1, o problema foi resolvido filtrando os utilizadores "visíveis" pelo *Keystone*.

5.2 COMPORTAMENTO EM SITUAÇÕES DE FALHA

O passo seguinte consistiu na realização de um conjunto de testes, que se pretendia permitirem analisar o comportamento, dos diversos sistemas, em situações de falha. Os testes tinham como objectivo avaliar a robustez destes sistemas especialmente nos aspectos de disponibilidade e consistência de dados. É necessário que se perceba que, de acordo com o teorema *CAP*², não é possível a um sistema distribuído, em caso de partição, manter-se simultaneamente estas duas características.

A metodologia utilizada é muito simples: provocou-se a falha intencional de parte de um determinado *cluster* e observou-se o seu comportamento.

O teste foi considerado bem sucedido se se tiver observado, cumulativamente, os seguintes comportamentos:

1. O *cluster* manteve a consistência dos dados.
2. O serviço, disponibilizado pelo *cluster*, continuou disponível de forma ininterrupta (ou, pelo menos, com uma interrupção temporária muito breve), ainda que possivelmente, a custo da degradação (que a acontecer, deve ser proporcional à gravidade da falha) da qualidade do serviço.

De notar que, como já vimos, há situações em que é impossível que os dois comportamentos se verifiquem, tendo-se, nesses casos, privilegiado a consistência.

Para avaliar essas situações (quando aplicáveis), foi ainda testado o comportamento do sistema em caso de partição (devido a uma falha de rede, o *cluster* é dividido em pelo menos dois grupos de nós que não conseguem contactar entre si). Para provocar este tipo de falhas, foram utilizadas regras de *firewall* para bloquear a comunicação entre os nós.

Para avaliar a continuidade de operação e capacidade de resposta dos serviços, durante cada um dos testes, foram criados pequenos *bash scripts*,

² O teorema *CAP* (*Consistency, Availability, Partition tolerance*) é o teorema, enunciado por *Eric Brewer*, segundo o qual não é possível a um sistema computacional distribuído garantir mais do que duas destas características simultaneamente. É, no entanto, comumente aceite que as partições, num sistema distribuído, são inevitáveis (embora raras) [22] e que o que não é possível garantir é a consistência e a disponibilidade em caso de partição.

tipicamente utilizando *curl*³, que realizam pedidos *HTTP* ininterruptos (um a cada segundo, independentemente de o anterior ter já terminado ou não) a um determinado serviço (cuja resposta permita inferir a saúde do *cluster* que se está a testar) e guardam o resultado e o tempo de execução num ficheiro de output.

É de salientar ainda que, durante os testes foi necessário alterar a política de balanceamento do *load balancer* para *round-robin*. Isto porque, com as políticas que haviam sido definidas e uma vez que os pedidos, utilizados para verificar a resposta dos serviços, eram provenientes de uma única máquina, estes iriam ser encaminhados para um único servidor.

Para cada sistema, foram testados diversos cenários, de diferente gravidade. Isto é, para um *cluster* com *N* nós, foi testada a falha de 1, 2, ... e *N*-1. O cenário de falha de *N* nós não foi testado, por razões óbvias.

5.2.1 Keepalived

O comportamento do *Keepalived* não foi testado, uma vez que a sua falha não acarreta qualquer impacto na disponibilidade dos restantes serviços. A falha de uma das instâncias do *Keepalived* simplesmente pode impedir que o nó *SLAVE* do *HAProxy* assuma automaticamente o papel de *MASTER*, numa eventual situação de falha.

5.2.2 HAProxy

O *cluster HAProxy* tem a particularidade de ser constituído apenas por dois nós e de ser um *cluster Master/Slave*, pelo que os cenários utilizados divergem dos utilizados nos restantes *clusters*.

O *script* utilizado, para avaliar a resposta do serviço, faz pedidos *HTTP* ao *dashboard OpenStack* (que precisam de atravessar o *HAProxy* para lá chegar):

```
#!/bin/bash
counter=0
while sleep 1
do
    curl -sL -w "$counter: %{http_code} %{time_total}\n" "
        http://co" -o /dev/null | tee -a curl_times.out &
    counter=$((counter+1))
done
```

Como se pode ver na Tabela 8, foram testados dois cenários: a falha do nó *MASTER* e a falha do nó *SLAVE*.

Na Figura 30 pode ver-se o comportamento do *cluster* durante a falha do nó *MASTER*. Aproximadamente aos 8 segundos do teste, o nó *MASTER* falhou, originando uma indisponibilidade de cerca de 1 segundo (o tempo que o *Keepalived* demorou a detectar a falha e a promover o nó *SLAVE*). O *cluster* retomou depois a sua normal actividade.

O segundo cenário (falha do nó *SLAVE*) como seria de esperar, não teve qualquer impacto na operação do *cluster*.

³ O *curl* é uma ferramenta *CLI* que permite fazer o download de informação via *HTTP* (entre muitos outros protocolos).

Cenário	Comportamento	Disponível
Falha do nó MASTER	Nó SLAVE assumiu o papel de MASTER (com breve interrupção total do serviço)	✓
Falha do nó SLAVE	Nó MASTER manteve o seu funcionamento (sem qualquer interrupção)	✓

Tabela 8: HAProxy - Comportamento em situação de falha.

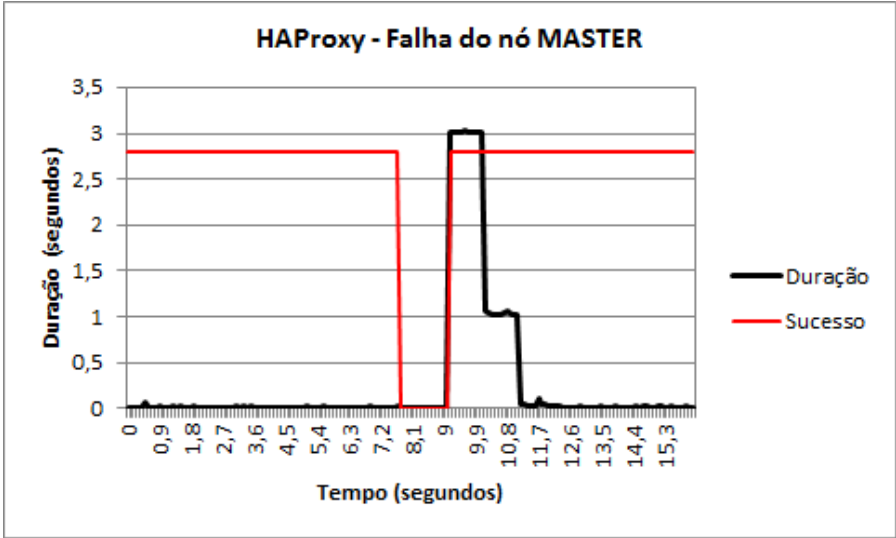


Figura 30: HAProxy - Comportamento do cluster durante falha do nó MASTER.

5.2.3 Percona XtraDB Cluster

Relativamente aos testes do cluster de base de dados, foram utilizados quatro cenários, como se pode ver na Tabela 9: a falha de dois nós (de forma sequencial) e partição do *cluster* (primeiro em dois nós + um, seguida de nova partição um + um). Para avaliar a disponibilidade do serviço, foram feitas *queries* ininterruptas a uma tabela de testes, utilizando o seguinte *script*:

```
#!/bin/bash
counter=0
declare -A start_times=()
while sleep 1
do
    start_times[$counter]=$ (date +%s%N)
    mysql -u root -h c0 -D test_database -pDreams936603? -e "
        select count(*) from test_table;" | awk -v c=$counter
        -v t=$ (date +%s%N) -${start_times[$counter]} 'FNR
        == 2 {print c ">" t }' | tee -a test &
    counter=$((counter+1))
done
```

Cenário	Comportamento	Disp.	Consis.
Falha de um nó	Serviço manteve o seu funcionamento (com breve interrupção parcial e sem degradação perceptível).	✓	✓
Falha de um segundo nó	Serviço interrompeu totalmente o seu funcionamento após breve interrupção parcial.	✗	✓
Partição do <i>cluster</i> (em 2+1)	Serviço manteve o seu funcionamento (com breve interrupção parcial e sem degradação perceptível).	✓	✓
Segunda partição do <i>cluster</i> (em 1+1)	Serviço interrompeu totalmente o seu funcionamento após breve interrupção parcial.	✗	✓

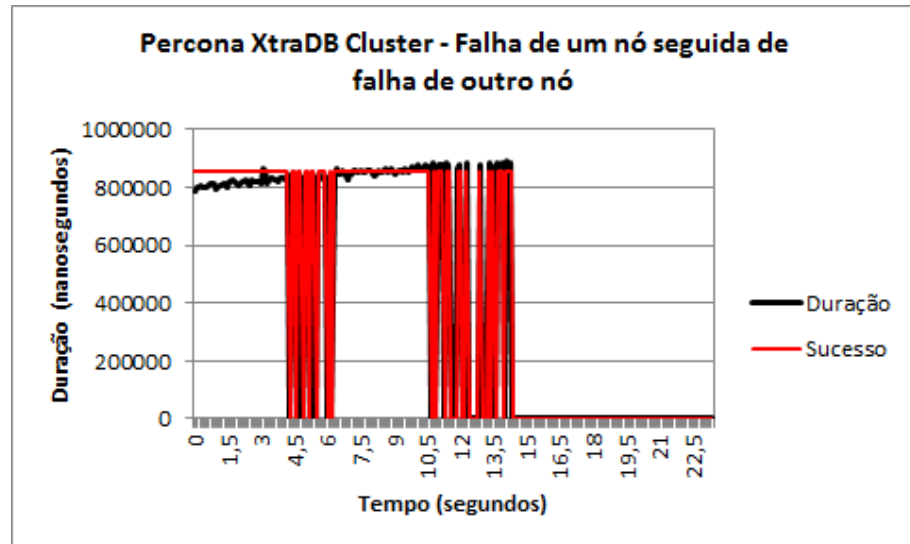
Tabela 9: Percona XtraDB Cluster - Comportamento em situação de falha.

Os resultados, corresponderam àquilo que era o esperado.

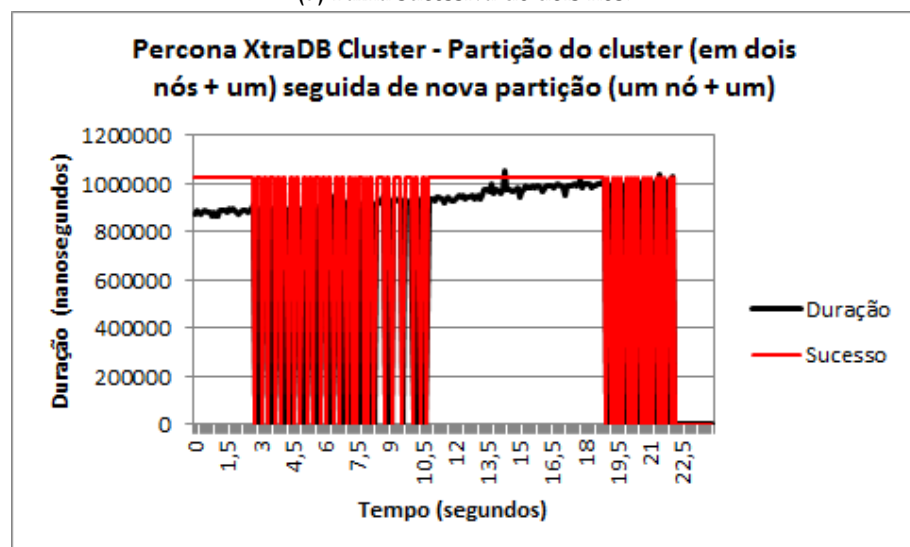
Com a falha de um nó, os dois nós restantes são capazes de proporcionar a continuidade do serviço. A falha parcial temporária, que se pode ver na Figura 31a entre os 4,5 e os 6 segundos, corresponde ao tempo que o *HAProxy* demorou a detectar a falha do nó e a deixar de lhe encaminhar pedidos.

A falha de um segundo nó, originou também uma breve quebra parcial do serviço, após a qual o *cluster* interrompeu totalmente o serviço. Esta paragem aconteceu porque o nó restante não consegue perceber se se encontra na presença de uma situação de falha ou de partição de rede e interrompe o serviço preventivamente.

Na Figura 31b pode ver-se o comportamento do *cluster* nos outros dois cenários. Após se isolar um dos nós (criando uma partição com um nó e



(a) Falha sucessiva de dois nós.



(b) Partição sucessiva do cluster.

Figura 31: Percona XtraDB Cluster - Comportamento do cluster

outra com dois), iniciou-se o processo de obtenção de *quorum*, originando uma quebra parcial do serviço. Após obtenção do *quorum* por parte da partição de dois nós, o *cluster* retomou a normal actividade.

Após nova partição do *cluster*, nenhuma das duas partições conseguiu obter o *quorum*, pelo que o *cluster* ficou indisponível para proteger a consistência dos dados.

5.2.4 RabbitMQ

Os testes seguintes foram ao *cluster RabbitMQ*. Os cenários testados, como se pode ver na Tabela 10, foram os mesmos quatro da secção anterior.

Para registar o comportamento, foi utilizado o *script* abaixo, que tem como propósito estabelecer repetidamente ligações *TCP* na porta do *RabbitMQ* e injectar lixo, de forma a obter uma resposta (a resposta, a qualquer pedido que o *RabbitMQ* não compreenda, é "AMQP").

```
#!/bin/bash
counter=0
declare -A start_times=()
while sleep 1
do
    start_times[$counter]=$ (date +%s%N)
    echo "abcdefghijklmnopqrstuvwxyzo123456789" | socat - TCP:
    c0:5672 | awk -v c=$counter -v t=$ (date +%s%N) -${
    start_times[$counter]} 'FNR == 2 {print $1 ":" c "→
    " t }' | tee -a test &
    counter=$((counter+1))
done
```

Cenário	Comportamento	Disp.	Consis.
Falha de um nó	Serviço manteve o seu funcionamento (com breve interrupção parcial e sem degradação perceptível).	✓	✓
Falha de um segundo nó	Serviço interrompeu totalmente o seu funcionamento, após breve interrupção parcial.	✗	✓
Partição do <i>cluster</i> (em 2+1)	Serviço manteve o seu funcionamento (com breve interrupção parcial que se manteve por aproximadamente vinte segundos).	✓	✓
Segunda partição do <i>cluster</i> (em 1+1)	Serviço interrompeu completamente o seu funcionamento após interrupção parcial.	✗	✓

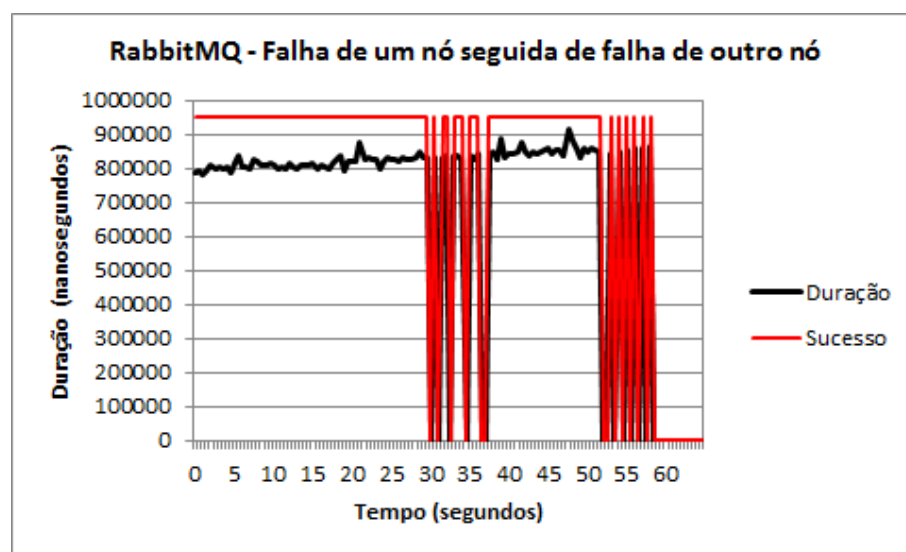
Tabela 10: *RabbitMQ* - Comportamento em situação de falha.

Como se pode ver na Tabela 10 e na Figura 32, o comportamento do sistema foi muito semelhante ao do *Percona XtraDB Cluster*. No caso de falha

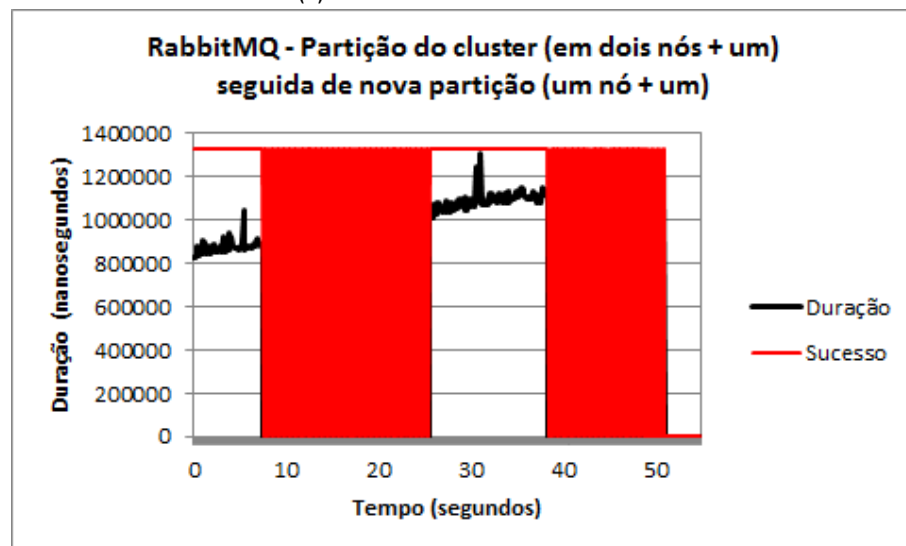
de um nó, o sistema sofreu uma breve interrupção parcial do serviço, após a qual retomou a normal actividade. Já na falha de um segundo nó, o sistema interrompeu totalmente a actividade, após um período de interrupção parcial.

A nível de partições, com uma primeira partição, o sistema foi capaz de se manter em funcionamento, tendo, no entanto, sofrido uma interrupção parcial algo prolongada. Após uma segunda partição, o sistema ficou inactivo, sendo este o comportamento esperado.

À semelhança do *Percona*, pode ver-se que, quer nos cenários de falha, quer nos de partição, o comportamento é muito semelhante. Daqui se percebe que, na prática, a diferença entre a falha de um nó e uma partição é muito diminuta.



(a) Falha sucessiva de dois nós.



(b) Partição sucessiva do *cluster*.

Figura 32: *RabbitMQ* - Comportamento do *cluster*

5.2.5 memcached

O *memcached* é um *cluster* muito peculiar, sendo que a sua configuração Activa/Passiva, faz com que a cada momento apenas exista um nó activo.

Por este motivo, o problema da consistência dos dados não se coloca, sendo que também não fazia sentido avaliarmos o comportamento em situações de partição, porque a verdade é que não existe qualquer tipo comunicação que possa ser interrompida.

A nível de disponibilidade, o comportamento do *cluster* é bastante previsível: quando o nó que se encontra em actividade falha, um dos restantes passa a ser o nó principal.

Utilizou-se, então, o seguinte script para registar o comportamento do *cluster*:

```
#!/bin/bash
counter=0
declare -A start_times=()
while sleep 1
do
    start_times[$counter]=$(date +%s%N)
    echo "stats" | socat - TCP:c0:11211 | awk -v c=$counter -
    v t=${$(date +%s%N)-${start_times[$counter]}} 'FNR ==
    2 {print $1 ":" c "->" t }' | tee -a test &
    counter=$((counter+1))
done
```

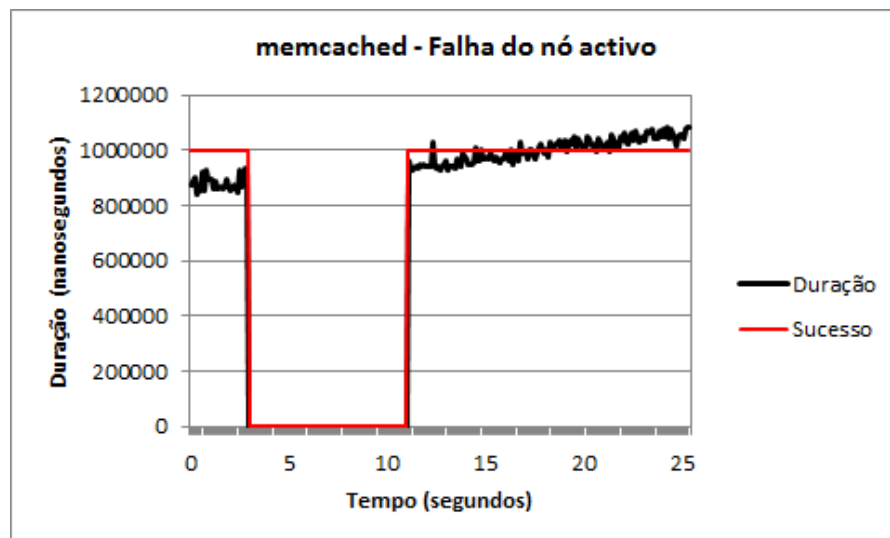


Figura 33: *memcached* - Comportamento após falha de um nó.

Como se pode ver na Figura 33, o comportamento foi exactamente o esperado: após falha do nó *Master*, houve um breve período de total inactividade, correspondente ao tempo que o *HAProxy* levou na sua detecção, após o qual o tráfego passou a ser direccionado para um dos nós *Slave*.

5.2.6 Ceph

Os testes relativos ao *Ceph* revelaram-se bastante complexos. Primeiro de tudo, por não se tratar de um único *cluster*, mas sim de três. E depois,

porque, devido às suas características, o *cluster* de dados (*OSD*) não tem apenas como estados possíveis "disponível" e "indisponível".

A qualquer momento, os *placement groups* do *Ceph* podem estar indisponíveis, disponíveis para leitura ou disponíveis para leitura e escrita. Sendo que este conjunto de estados simplificados corresponde apenas aos estados percebidos pelo utilizador, já que, na verdade, o *Ceph* tem um conjunto de estados internos muito maior (dezoito), podendo inclusivamente, estar em mais do que um simultaneamente. Para complicar ainda mais a situação, os diversos *placement groups* existentes podem estar (e, em situação de falha, geralmente estão) em estados diferentes.

Assim, decidiu-se utilizar uma metodologia diferente, daquela utilizada nos restantes sistemas, para avaliar o comportamento. Após a criação da situação de falha, foi utilizado o *ceph-client* para monitorizar o estado do *cluster* com o seguinte comando:

```
$ ceph -w
```

O que este comando faz é mostrar, em tempo real, o número de nós em actividade, em cada *cluster*, e o estado de todos os *placement groups*.

O primeiro *cluster* testado foi o *MON*, sendo que o comportamento observado, como pode ser visto na Tabela 11, foi, sem surpresa, muito semelhante ao dos restantes sistemas com funcionalidade de *quorum* (*Percona XtraDB Cluster* e *RabbitMQ*). Após a falha de um dos nós, os restantes dois reuniram o *quorum* e mantiveram o *cluster* em funcionamento e, após falha de um segundo, o sistema ficou imediatamente indisponível (indisponibilizando também, como consequência, os dois restantes *clusters*). De igual forma, com uma partição de rede (2 nós + 1), o sistema manteve-se em funcionamento, tendo ficado indisponível com a ocorrência de uma segunda.

Cenário	Comportamento	Disp.	Consis.
Falha de um nó	Serviço manteve o seu funcionamento (após reunião de <i>quorum</i>).	✓	✓
Falha de um segundo nó	Serviço interrompeu totalmente o seu funcionamento.	✗	✓
Partição do <i>cluster</i> (em 2+1)	Serviço manteve o seu funcionamento (após reunião de <i>quorum</i>).	✓	✓
Segunda partição do <i>cluster</i> (em 1+1)	Serviço interrompeu completamente o seu funcionamento.	✗	✓

Tabela 11: *Ceph* - Comportamento do *cluster MON* em situação de falha.

O *cluster* seguinte, *MDS*, devido à sua configuração Activo/Passivo, apresentou um comportamento ainda mais previsível, como se pode observar na Tabela 12. A falha do nó *Master*, gerou uma pequena indisponibilidade temporária do serviço *CephFS*, após a qual um dos nós de *backup* entrou em funcionamento. A falha de um dos nós de *backup*, como seria de esperar, não teve qualquer impacto.

Por fim, o *cluster* mais crítico, o *OSD*. Neste sistema, em vez de ser feita a monitorização binária da disponibilidade do serviço, fez-se a monitóri-

Cenário	Comportamento	Disponível
Falha do nó <i>MASTER</i>	Nó <i>SLAVE</i> assumiu o papel de <i>MASTER</i> .	✓
Falha do nó <i>SLAVE</i>	Nó <i>MASTER</i> manteve o seu funcionamento (sem qualquer interrupção)	✓

Tabela 12: *Ceph* - Comportamento do *cluster MDS* em situação de falha.

zação, de mais baixo nível, do estado dos *placement groups*, pelas razões já mencionadas.

Como se pode ver na Tabela 13, o comportamento observado não correspondeu exactamente ao que era esperado, uma vez que se esperava que após a falha de um dos *OSDs*, o *Ceph* redistribuísse automaticamente as réplicas dos *PGs*, pelos restantes, de forma a manter constantemente três cópias activas de cada objecto.

No entanto, após a falha de um único *OSD*, quase metade dos *PGs* ficaram indisponíveis para leitura e após falha de um segundo, a totalidade deixou de estar disponível. Este estado manteve-se durante a falha consecutiva de mais três *OSDs*, sendo que a falha de um quinto (deixando o *cluster* apenas com um activo) fez com 471 desses *PGs* passassem a estar indisponíveis de todo (quer para leitura, quer para escrita).

Depois de uma leitura mais exaustiva da documentação oficial do *Ceph*, percebeu-se que este comportamento não constitui uma anomalia, visto ser o comportamento por defeito. Isto deve-se ao facto de o *Ceph* definir como unidade mínima de replicação (*leaf type*) o *host* e não o *OSD*. Embora fosse possível alterar o *leaf type* para *OSD*, optou-se por manter a configuração por defeito, uma vez que a sua alteração acarretava consequências mais gravosas (possibilidade de o *Ceph* não conseguir lidar com a falha de todos os *OSDs* num *host*, podendo levar à perda de dados).

Cenário	Comportamento
Cenário inicial (antes de qualquer falha)	960 <i>PGs</i> , todos no estado <i>active+clean</i> (disponíveis).
Falha de um <i>OSD</i>	489 <i>PGs active+clean</i> e 471 <i>active+degraded</i> (disponíveis apenas para leitura).
Falha de dois <i>OSDs</i>	960 <i>active+degraded</i> .
Falha de três <i>OSDs</i>	960 <i>active+degraded</i> .
Falha quatro <i>OSDs</i>	960 <i>active+degraded</i> .
Falha de cinco <i>OSDs</i>	489 <i>PGs active+degraded</i> e 471 <i>active+degraded+stale</i> (indisponíveis).

Tabela 13: *Ceph* - Comportamento do *cluster OSD* em situação de falha.

5.2.7 Serviços de gestão *OpenStack*

A etapa seguinte diz respeito aos *clusters* de serviços de gestão do *OpenStack*. Nesta categoria enquadram-se todos os serviços expostos ao utilizador por

intermédio do *load balancer* (*Keystone*, *Glance*, *Horizon*, *Cinder*, etc...). Os passos da execução de um pedido a um destes serviços são os seguintes:

1. Pedido do cliente chega ao *HAProxy*.
2. *HAProxy* encaminha o pedido para um dos nós disponíveis.
3. Serviço recebe o pedido e analisa-o.
4. Serviço consulta informação de um dos *backends* (quando aplicável).
5. Serviço processa o pedido.
6. Serviço escreve informação num dos *backends* (quando aplicável).
7. Serviço envia resposta ao pedido.
8. Serviço termina a sua execução.

Analisando este fluxo de execução, facilmente se percebe que todos estes serviços têm a particularidade de serem *stateless*, uma vez que utilizam exclusivamente outros serviços (base de dados, *messaging queues*, etc...) para persistirem os dados. Por este motivo, e à semelhança do que acontece com o *memcached*, a consistência não é uma preocupação.

De igual forma, se percebe que, uma vez que os nós do *cluster* operam de forma independente, a falha de um deles tem como único impacto, uma percebida falha parcial temporária do sistema (correspondente aos pedidos que o *HAProxy* continua a encaminhar para esse nó até que se aperceba da sua falha). Para o comprovar, na Figura 34, pode-se observar o comportamento de um desses serviços (*Keystone*).

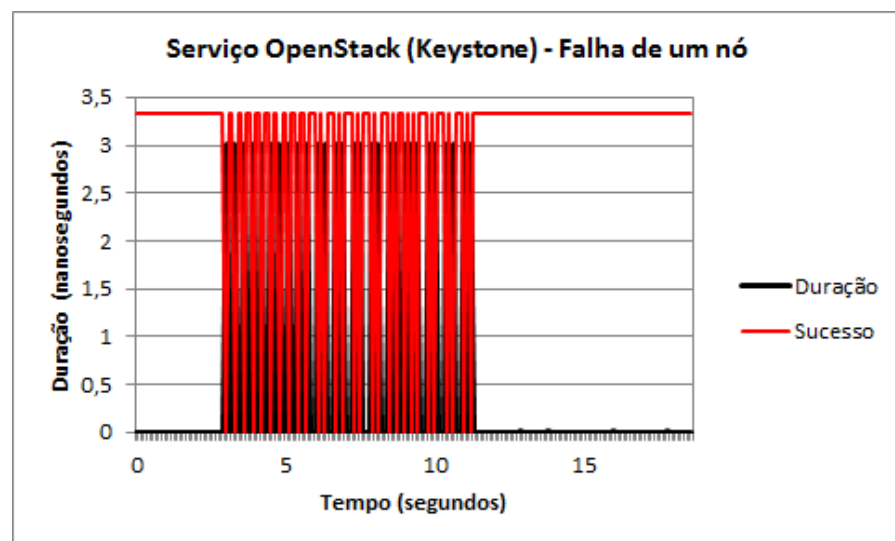


Figura 34: *Keystone* - Comportamento após falha de um nó.

5.2.8 nova-compute

Ao contrário dos restantes serviços, o *nova-compute* não pode ser propriamente considerado um *cluster*. Embora existam várias instâncias (uma em cada nó de computação), a verdade é que estas operam de forma totalmente independente.

É, no entanto, de extrema importância perceber o que acontece em caso de falha de um destes serviços, sendo que o comportamento esperado era que todas as VMs, a correr nesse nó, fossem automaticamente migradas para um outro.

Para testar esta situação, foi criada uma VM num dos três nós de computação, tendo-se de seguida morto o processo do *nova-compute*. Ao contrário do que era esperado, a VM ficou indisponível, mas não ocorreu o processo de migração.

Depois de alguma pesquisa, percebeu-se que, de facto, este é o comportamento expectável e que, pelo menos para já, o *OpenStack* não suporta evacuação (termo aplicado ao processo de migrar VMs de um nó de computação que sofreu uma falha) automática.

Na realidade, a adição desta funcionalidade é um tema controverso na comunidade, uma vez que tem um grande número de implicações. Por exemplo, é necessário que exista um mecanismo confiável que permita verificar que, de facto, o nó de computação falhou. Caso contrário, existe a possibilidade de este retomar o funcionamento (no caso de se ter tratado de uma falha transitória) durante o processo de evacuação, levando à ocorrência de comportamentos inesperados e aleatórios (duplicação de VMs, desaparecimento de VMs, etc...).

Assim, é de esperar que, neste tipo de situações, exista a intervenção manual de um administrador. Foi, portanto, necessário executar o seguinte comando, por forma a migrar todas as VMs (neste caso só uma) para um novo nó de computação:

```
$ nova host-evacuate --target_host cpu2 --on-shared-storage cpu1
```

5.2.9 nova-network

À semelhança do *nova-compute*, o *nova-network* também não pode ser considerado um *cluster*, uma vez que cada instância funciona de modo independente das restantes.

Após a falha deste serviço, nenhuma das VMs sofreu qualquer tipo de impacto, uma vez que o *nova-network* é apenas utilizado durante a configuração inicial da rede em cada uma delas. Ou seja, a sua falha impacta apenas as VMs que se criem ou iniciem posteriormente.

Esta situação persiste até que seja resolvida manualmente, uma vez que o *nova-network* não contempla nenhum mecanismo de redundância nem de recuperação de falhas. No entanto, devido à nossa utilização do *nova-network* em configuração *multi-host*, o impacto da falha foi restringido a apenas um dos nós de computação.

5.3 RECUPERAÇÃO DE SITUAÇÕES DE FALHA

Analizado o comportamento do sistema em situações de falha, o passo seguinte consistiu em analisar o comportamento do sistema após essas situações se deixarem de verificar, de forma a perceber as capacidade de recuperação destes sistemas.

Os cenários utilizados são, em grande parte, análogos aos da secção anterior.

5.3.1 *HAProxy*

No caso do *HAProxy*, foi testado o comportamento do *cluster* após a recuperação do nó *MASTER*. De recordar que, aquando da falha do nó *MASTER*, o nó *SLAVE* havia assumido a responsabilidade da operação do *cluster*.

Após reinício do nó, o que se observou, foi que o *Keepalived* procedeu ao *failback* deste nó. Ou seja, este voltou novamente a assumir o papel de *MASTER*, tendo o outro nó sido demovido para o papel de *SLAVE*.

Ao contrário do que aconteceu aquando da falha, não se observou qualquer tipo de interrupção do serviço durante o processo de *failback*.

5.3.2 *Percona XtraDB Cluster*

Em relação ao *Percona XtraDB Cluster*, haviam sido testados cenários de falha e partição de rede de diferentes gravidades. Como se pode ver na Tabela 14, após o reinício do serviço ou eliminação da regra de *firewall* (utilizada para produzir a partição de rede), os nós afectados foram reintroduzidos no *cluster*, tendo este, nas situações em que havia ficado inactivo, retomado imediatamente a actividade.

Cenário	Comportamento
Recuperação de um nó após falha de um nó	O nó foi imediatamente reintegrado no <i>cluster</i> , não causando qualquer tipo de indisponibilidade.
Recuperação de um nó após falha de dois nós	Serviço retomou o seu funcionamento (que havia sido interrompido).
Restauração da conectividade entre duas partições (2+1)	As duas partições foram unidas, não tendo sido causada indisponibilidade.
Restauração da conectividade entre duas partições (1+1)	Após um intervalo de alguns segundos, as duas partições não primárias (sem <i>quorum</i>) foram unidas. O serviço foi retomado.

Tabela 14: *Percona XtraDB Cluster* - Recuperação de falhas.

5.3.3 *RabbitMQ*

Devido à semelhança de comportamento com o *Percona XtraDB Cluster*, observada durante os testes de falhas, esperava-se, de igual forma, um comportamento semelhante nos testes de recuperação. No entanto, como se pode ver na Tabela 15, esse não foi o caso.

Embora o *RabbitMQ* tenha sido capaz de recuperar facilmente das situações em que os nós tinham falhado, tal não se verificou nas situações de partição de rede, tendo o *cluster* continuado dividido e, num dos casos, indisponível.

Este comportamento deve-se ao facto de, ao contrário do *Percona*, o *RabbitMQ* possuir um mecanismo que lhe permite identificar inequivocamente as situações em que ocorreu uma partição de rede: durante uma partição, cada instância mantém uma lista dos nós que não conseguem contactar. Após a conectividade ser retomada, é estabelecida uma "conversação" e, se

Cenário	Comportamento
Recuperação de um nó após falha de um nó	O nó foi imediatamente reintegrado no <i>cluster</i> , não causando qualquer tipo de indisponibilidade.
Recuperação de um nó após falha de dois nós	Serviço retomou o seu funcionamento (que havia sido interrompido).
Restauração da conectividade entre duas partições (2+1)	Foi emitido um alerta de partição de rede (via <i>dashboard</i>). Após reinício manual do nó da partição minoritária, este foi adicionado ao <i>cluster</i> . Durante todo o processo não se verificou nenhuma indisponibilidade do serviço.
Restauração da conectividade entre duas partições (1+1)	O serviço continuou indisponível, tendo sido reiniciar um dos nós, para que a actividade fosse retomada.

Tabela 15: *RabbitMQ* - Recuperação de falhas.

dois nós constarem na lista um do outro, estes chegam à conclusão que sofreram uma partição. Nestes casos, por precaução, é necessária a intervenção manual do administrador, que deve decidir qual o nó com a informação correcta (se eventualmente forem diferentes) e reiniciar o restante. Após este processo o *cluster* retoma a sua normal actividade.

5.3.4 *memcached*

Como foi dito na secção anterior, a configuração Activa/Passiva do *memcached*, fez com que, devido a uma falha do nó em actividade, se tivesse observado um breve período de inactividade (correspondente ao tempo que o *HAProxy* levou a detectar a falha), após o qual um novo nó entrou em actividade.

Assim, o comportamento que se esperava observar, após reiniciar o nó que havia falhado, era este ser adicionado como nó de backup ao *cluster*, não afectando o funcionamento do sistema. No entanto, este não foi o comportamento que se observou.

Embora não se tenha verificado nenhuma interrupção do serviço, após consulta ao *dashboard* do *HAProxy*, verificou-se que o nó tinha voltado a assumir o papel de nó *MASTER*. Isto, é explicado pelo facto de o *HAProxy* utilizar um mecanismo de *failback* em todos os *clusters* do tipo Activo/Passivo.

Este comportamento, aparentemente inofensivo, tem, no entanto, uma grande desvantagem quando aplicado ao *memcached*: aquando da falha no nó principal, a *cache* do sistema perdeu-se, tendo de ser, depois, progressivamente reconstruída no novo nó (à custa de um grande número de *cache misses*). Ao voltarmos a alterar o nó *Master* (desnecessariamente), estamos a fazer com que este processo se repita novamente.

Uma vez que o *HAProxy* não permite desabilitar o mecanismo de *failback*, este comportamento ainda persiste, sendo necessário, caso se torne realmente problemático, encontrar uma alternativa.

5.3.5 *Ceph*

À semelhança do que foi feito na análise do comportamento em situação de falha, optou-se também aqui por fazer uma análise *cluster* a *cluster* do *Ceph*.

No caso do *MON*, o comportamento, visível na Tabela 16, foi bastante linear. Tendo, a eliminação das condições de erro, provocado o retorno do *cluster* à normalidade (e à actividade quando esta havia sido interrompida).

Cenário	Comportamento
Recuperação de um nó após falha de um nó	O nó foi imediatamente reintegrado no <i>cluster</i> .
Recuperação de um nó após falha de dois nós	Serviço retomou o seu funcionamento (que havia sido interrompido).
Restauração da conectividade entre duas partições (2+1)	As duas partições foram unidas.
Restauração da conectividade entre duas partições (1+1)	O serviço voltou a ficar disponível, após união das duas partições.

Tabela 16: *Ceph* - Recuperação de falhas no *cluster MON*.

No caso do *MDS*, o comportamento foi ainda mais trivial, sendo que a recuperação do nó (que anteriormente era o *Master*) fez apenas com que este fosse reinserido no *cluster*. Foi possível ainda verificar que não ocorreu *failback*, tendo o nó sido adicionado como *Slave*.

Por fim, no caso do *ODS*, embora não tenha sido perceptível nos testes realizados, é de esperar uma breve indisponibilidade do serviço, ou pelo menos alguma degradação, uma vez que, após recuperação de um nó, o *Ceph* tem de proceder ao reequilíbrio do *cluster*. No nosso caso, como já visto anteriormente, isto não se verificou. Uma vez que o *Ceph* não havia rebalanceado o *cluster*, na situação de falha, a recuperação foi instantânea, não tendo originado qualquer tipo de indisponibilidade.

5.3.6 Serviços de gestão *OpenStack*

Devido à sua natureza *stateless*, a recuperação de falhas dos serviços de controle do *OpenStack* é bastante trivial. Após reinício de um nó, este volta imediatamente a ser adicionado ao *cluster*, não causando qualquer impacto na disponibilidade do sistema.

5.3.7 *nova-compute*

Em relação ao *nova-compute*, e como vimos nos testes de falha, este não possui nenhum mecanismo automático de evacuação das *VMs*. Portanto, como é natural, também não tem nenhum mecanismo para a situação inversa.

Após reinício ou resolução do problema que *afectava* o nó de computação, este é simplesmente adicionado à *pool* de nós *activos*, ficando disponível para alojar novas *VMs*.

Caso se pretenda, pode-se facilmente migrar de volta as *VMs* que este alojava, utilizando o mesmo mecanismo de anteriormente (*nova evacuate*).

5.3.8 nova-network

Por fim, também a recuperação do *nova-network* apresenta um comportamento bastante simples. Reiniciado o serviço, é restaurada a capacidade de novas VMs serem criadas ou iniciadas com configurações de rede válidas, no nó de computação afectado.

5.4 CONSIDERAÇÕES SOBRE A ESCALABILIDADE

No que diz respeito à escalabilidade do sistema implementado, os constrangimentos de tempo e de equipamento disponível, não permitiram que se fizesse o *benchmarking* necessário à sua análise objectiva.

Assim, em alternativa, optou-se por fazer uma análise funcional, tentando perceber de que forma é possível escalar o sistema, como um todo, e quais os impedimentos que, eventualmente, existam.

5.4.1 Ceph

Uma vez que o *Ceph* se trata de um sistema autónomo e exterior ao *OpenStack*, faz sentido que seja também analisado de forma independente. Assim, observando a sua arquitectura, facilmente se percebe que este se trata de um sistema que pode, de forma fácil, ser escalado horizontalmente. A adição de nós de armazenamento permite aumentar, de forma linear, a capacidade do *cluster* e, simultaneamente, o nível de tolerância a falhas (uma vez que passa a existir um maior número de nós para onde o *Ceph* pode mover réplicas em caso de necessidade).

De igual forma, o *cluster* de monitorização, sendo um sistema com baixos requisitos de performance, pode facilmente ser incrementado ao nível da disponibilidade, através da adição de novos nós.

No *MDS*, no entanto, devido à sua configuração Activo/Passivo, o aumento de número de *hosts* produz apenas efeito ao nível da tolerância a falhas, exigindo, em caso de necessidade, o aumento dos recursos de computação (escalabilidade vertical) do *cluster*, por forma a incrementar a sua performance.

5.4.2 OpenStack

Ao nível dos serviços de controle e computação, o *OpenStack* possibilita que estes sejam escalados horizontalmente de forma indeterminada (existem *deployments* com milhares de *hosts*), incrementando a quantidade de recursos computacionais disponíveis, a tolerância e a disponibilidade do sistema e a performance (sobretudo dos serviços de controle).

Esta capacidade, aparentemente ilimitada, é, no entanto, fortemente comprometida pelos serviços auxiliares utilizados. Nomeadamente o sistema de base de dados e o *messaging middleware*.

Estes sistemas, embora permitam a realização de pedidos a qualquer um dos seus nós, obrigam a que as operações de escrita sejam primeiro replicadas por todo o *cluster* antes de serem confirmadas. Isto, faz com que a performance do *cluster* seja tão elevada quanto a performance do seu elemento mais fraco. Ou seja, a existência de um nó com sub-performance prejudica gravemente a rapidez de todo o *cluster*. Assim, a escalabilidade

horizontal destes sistemas permite apenas obter ganhos ao nível da disponibilidade, sendo necessário aumentar os recursos computacionais de todos os nós por forma a incrementar a sua capacidade e performance.

Este, é um problema de difícil resolução (se é que possível), em sistemas que privilegiem a disponibilidade e a consistência dos dados, existindo no entanto medidas que podem ser adoptadas para o mitigar. Pode-se, por exemplo, segmentar as bases de dados, criando um *clusters* para cada uma delas. Sendo que, esta solução tem, no entanto, um forte impacto ao nível da complexidade arquitectural, e consequentemente da dificuldade da administração e manutenção do sistema.

Um outro ponto de estrangulamento é o *load balancer*. Embora não constitua uma preocupação ao nível da disponibilidade (pois, como se viu, foi instalada uma solução redundante), apresenta fortes constrangimentos ao nível da capacidade e performance, pois é responsável por processar todo o tráfego do sistema.

Mais uma vez, este é uma problema complexo, cuja resolução deverá passar por escalar verticalmente o sistema (pouco prático e dispendioso). Há poucas possibilidades para mitigar esta situação, sendo que, por exemplo, a criação de diversos pontos de entrada no ambiente (vários *load balancers*, cada um com um endereço *IP* distinto) é complexa e nem sempre factível.

5.5 ACTUALIZAÇÃO DO OPENSTACK

Uma vez que se tem assistido a uma muito rápida evolução do projecto *OpenStack*, com novas *releases* a acontecerem periodicamente de seis em seis meses, achou-se que seria relevante avaliar a dificuldade e complexidade de actualização para uma versão mais recente.

Assim, uma vez que durante este projecto utilizámos a versão *Havana* e uma vez que a *Icehouse* já se encontra disponível, optamos por migrar o nosso ambiente para esta versão.

Antes de mais, importa esclarecer que apenas se procedeu à actualização dos projectos que haviam já sido instalados, não se tendo adoptado o novo projecto que foi introduzido nesta nova versão (*Trove* é o novo projecto *OpenStack*, sendo uma solução de *Database-as-a-Service*).

Para o processo de actualização foram seguidas as instruções disponibilizadas pela *Red Hat* [23]. Este processo, muito simples, consistiu basicamente em substituir o repositório da edição *Havana* pelo da *Icehouse* e reinstalar o *OpenStack* componente a componente, começando pelos serviços de controle e terminando no *nova-compute* e *nova-network*.

As instruções seguidas foram desenhadas para sistemas em ambiente de produção, pelo que não implicaram a indisponibilização do serviço.

A execução deste projecto, permitiu concluir que a adopção de um sistema de computação em *cloud* possibilitará ao GSIIC um melhor aproveitamento dos seus recursos tecnológicos e humanos (que poderão deixar de estar afectos à realização de tarefas repetitivas e que não acrescentam qualquer valor à actividade da organização).

Mais especificamente, foi possível concluir que o *OpenStack* é uma plataforma madura, apta a ser utilizada desde já pelo GSIIC, e com um grande potencial de evolução, existindo diversos projectos, de interesse, que se esperam serem lançados já nas próximas *releases*.

A implementação da prova de conceito permitiu ainda a obtenção de um conhecimento mais concreto sobre o *OpenStack*, nomeadamente nos aspectos de instalação, configuração e operação. O sistema implementado é dotado de capacidades de alta disponibilidade e tolerância a falhas, sendo que estes eram dois dos principais requisitos do projecto.

Esta prova de conceito, permitiu ainda analisar o comportamento do *OpenStack* em condições adversas, tendo-se concluído que o seu desempenho correspondeu àquilo que era de esperar de um sistema deste tipo.

As ferramentas utilizadas no aprovisionamento do sistema revelaram-se extremamente versáteis, tendo o GSIIC identificado outras áreas da sua actividade onde poderá beneficiar da sua utilização.

Foi ainda possível obter um primeiro contacto prático com o *Ceph*. Este sistema, que acabou por despertar um interesse quase tão grande como o próprio *OpenStack*, possui um conjunto de características que o tornam uma alternativa viável e eficiente, aos sistemas de armazenamento comerciais, altamente dispendiosos e de difícil escalabilidade (devido à sua arquitectura algo monolítica).

Finalmente, a nível pessoal, a realização deste projecto permitiu-me ter contacto com um vasto número de áreas (administração de sistemas *UNIX*, administração de redes informáticas, sistemas de virtualização, computação em *cloud*, tolerância a falhas e alta disponibilidade em sistemas distribuídos, etc. ...) em ambiente real, e obter conhecimento sobre diversas ferramentas que, com certeza, se revelarão mais valias no meu percurso profissional.

6.1 TRABALHO FUTURO

A evolução lógica deste projecto, consistirá no *deployment* do *OpenStack* em ambiente de produção. Este ambiente permitirá, gradualmente, ganhar confiança na solução e obter a desenvoltura necessária à sua operação no dia-a-dia.

No entanto, existem ainda um conjunto de aspectos que podem, e devem, ser adicionados e/ou melhorados na prova de conceito implementada.

Será muito importante, desde logo, aprofundar o conhecimento sobre o *Puppet*, para que, em conjunto com o conhecimento obtido neste projecto, se possa criar automatismos para o processo de *deployment*, diminuindo as-

sim o tempo dispendido e eliminando os erros de configuração, inevitáveis numa implementação manual.

Depois, devido à não criticidade da solução desenvolvida no âmbito deste projecto, houve um compromisso entre a robustez dos mecanismos de segurança e a facilidade da sua configuração (por exemplo, a utilização de *PKI* foi abandonada). O aspecto da segurança deverá agora ser revisto de forma mais minuciosa e cuidada, ainda que se preveja que num futuro próximo esta continue a ser uma solução de utilização apenas interna.

Existem também algumas preocupações relativamente ao futuro do *nova-network*, sendo que já por diversas vezes se ouviram rumores sobre a sua possível descontinuação. Este facto, aliado ao previsível aumento da complexidade dos serviços que utilizam a plataforma e que irão exigir capacidades cada vez mais avançadas ao nível da rede, faz com que a migração para o *Neutron* seja uma possibilidade a considerar e avaliar. A premência desta avaliação é ainda maior, se considerarmos que, uma possível migração, se tornará um processo extremamente complexo se realizado num ambiente de produção, já com um número considerável de utilizadores.

Será ainda relevante e necessário aprofundar o conhecimento sobre o *Ceph* e realizar testes mais detalhados, de forma a confirmar que esta se trata de uma opção viável para, eventualmente, substituir a longo prazo as soluções actualmente em uso.

Por fim e num outro contexto, será também útil realizar testes mais elaborados com o *Foreman*, nomeadamente em ambiente *Windows*, de forma a validar a utilidade desta ferramenta enquanto solução de aprovisionamento e gestão de configurações genérica, a utilizar pelo *GSIIC*.

BIBLIOGRAFIA

- [1] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres e Maik Lindner. A break in the clouds: towards a cloud definition. *Acm sigcomm computer communication review*, 39(1):50–55, 2008.
- [2] Lutz Schubert, Keith G Jeffery e Burkard Neidecker-Lutz. *The future of cloud computing: opportunities for european cloud computing beyond 2010:—expert group report*. European Commission, Information Society e Media, 2010.
- [3] Peter Mell e Tim Grance. The NIST definition of cloud computing. *National institute of standards and technology*, 53(6):50, 2009.
- [4] T Sridhar. Cloud computing—a primer part 1: models and technologies. *The internet protocol journal*, 12(3):2–19, 2009.
- [5] Randy Bias. Architectures for open and scalable clouds. Cloud Connect. 2012. URL: <http://pt.slideshare.net/randybias/architectures-for-open-and-scalable-clouds>.
- [6] OpenNebula Project. About the OpenNebula project. OpenNebula Project. URL: <http://openebula.org/about/project/> (acedido em 14/06/2014).
- [7] David Meyer. Here’s why CERN ditched OpenNebula for OpenStack. *Gigaom*, 31 de maio de 2013. URL: <http://gigaom.com/2013/05/31/heres-why-cern-ditched-openebula-for-openstack/> (acedido em 14/06/2014).
- [8] OpenNebula Project. Featured users. OpenNebula Project. URL: <http://openebula.org/users/featuredusers/> (acedido em 14/06/2014).
- [9] Cade Metz. The secret history of OpenStack, the free cloud software that’s changing everything. *Wired*, 4 de fevereiro de 2012. URL: <http://www.wired.com/2012/04/openstack-3/> (acedido em 14/06/2014).
- [10] OpenStack Foundation. Companies supporting The OpenStack Foundation. OpenStack Foundation. URL: <http://www.openstack.org/foundation/companies/> (acedido em 14/06/2014).
- [11] OpenStack Foundation. User stories. OpenStack users by industry. OpenStack Foundation. URL: <http://www.openstack.org/user-stories/> (acedido em 14/06/2014).
- [12] Caleb Garling. Citrix splits with OpenStack, takes cloud to Apache. *Wired*, 4 de abril de 2012. URL: <http://www.wired.com/2012/04/citrix-cloudstack/> (acedido em 17/07/2012).
- [13] Brandon Butler. Datapipe launches largest CloudStack deployment. Citrix-backed CloudStack is competing in the open source cloud world and now has a globally based IaaS offering. *Networkworld*, 20 de julho de 2010. URL: <http://www.networkworld.com/article/2190750/cloud-computing/datapipe-launches-largest-cloudstack-deployment.html> (acedido em 17/07/2012).

- [14] Cade Metz. NASA drops Ubuntu's Koala food for (real) open source. Open core is not open source: a cautionary tale. *The register*, 20 de julho de 2010. URL: http://www.theregister.co.uk/2010/07/20/why_nasa_is_dropping_eucalyptus_from_its_nebula_cloud/ (acedido em 14/06/2014).
- [15] Henrik Ingo. *Open life: the philosophy of open source*. Lulu. com, 2006.
- [16] OpenStack Foundation. OpenStack installation guide for Red Hat Enterprise Linux, CentOS, and Fedora - Havana. OpenStack Foundation. 2013. URL: http://docs.openstack.org/havana/install-guide/install/yum/content/ch_overview.html (acedido em 14/06/2014).
- [17] OpenStack Foundation. Hypervisor support matrix. OpenStack Foundation. URL: <https://wiki.openstack.org/wiki/HypervisorSupportMatrix> (acedido em 14/06/2014).
- [18] Brian Chong e Shane Gibson. An evaluation of OpenStack deployment frameworks. OpenStack Summit - Hong Kong 2013. 2013. URL: <https://www.youtube.com/watch?v=45GCWEVfWok> (acedido em 14/06/2014).
- [19] Benjamin Matthew, Casey Bodley, Adam Emerson e Marcus Watts. A saga of smart storage devices. An overview of object storage. *Login*, 39(1):6–10, 2014.
- [20] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long e Carlos Maltzahn. Ceph: a scalable, high-performance distributed file system. Em *Proceedings of the 7th symposium on operating systems design and implementation*. Em OSDI '06. USENIX Association, Seattle, Washington, 2006, páginas 307–320. ISBN: 1-931971-47-1.
- [21] Brad Fitzpatrick. Distributed caching with memcached. *Linux journal*, 2004(124):5, 2004.
- [22] Coda Hale. You can't sacrifice partition tolerance. 2010. URL: <http://codahale.com/you-cant-sacrifice-partition-tolerance> (acedido em 14/06/2014).
- [23] Red Hat. Upgrading rdo to icehouse. Red Hat. 2014. URL: http://openstack.redhat.com/Upgrading_RD0_To_Icehouse (acedido em 14/06/2014).

ANEXOS

A

INSTALAÇÃO DE SISTEMA
EXPERIMENTAL

1. Adicionar um novo utilizador (com permissões sudo) que será usado para instalar e correr o DevStack:

```
$ adduser stack
$ yum install &ndash;y sudo
$ echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

2. Mudar sessão para o utilizador criado:

```
$ su stack
```

3. Download do Devstack:

```
# sudo yum install &ndash;Y git
# git clone git://github.com/openstack-dev/devstack.git &ndash;b stable/havana devstack/
# cd devstack
```

4. Configuração do Devstack:

```
# sudo vi localrc
```

```
# Credentials
ADMIN_PASSWORD=devstack
MYSQL_PASSWORD=devstack
RABBIT_PASSWORD=devstack
SERVICE_PASSWORD=devstack
SERVICE_TOKEN=token

# Branches
NOVA_BRANCH=stable/havana
CINDER_BRANCH=stable/havana
GLANCE_BRANCH=stable/havana
HORIZON_BRANCH=stable/havana
KEYSTONE_BRANCH=stable/havana
NEUTRON_BRANCH=stable/havana
SWIFT_BRANCH=stable/havana

# Heat
enable_service heat h-api h-api-cfn h-api-cw h-eng
# Ceilometer
enable_service ceilometer-acompute ceilometer-acentral ceilometer-collector ceilometer-api
CEILOMETER_BACKEND=mongo
# Swift
enable_service s-proxy s-object s-container s-account

# Output
LOGFILE=/opt/stack/logs/stack.sh.log
VERBOSE=True
LOG_COLOR=False
SCREEN_LOGDIR=/opt/stack/logs
```

5. Iniciar o DevStack:

```
# ./stack.sh
```

B | CONFIGURAÇÃO DO SWITCH

1. Aceder à interface de gestão do switch.
2. Proceder à configuração do mesmo de acordo como mostrado abaixo. As portas 1 a 22, devem ser configuradas para transportar as VLANs 17 e 701–705 em modo `trunk` e com a VLAN 701 por defeito. As portas 23 e 24, a utilizar para conexão WAN, devem ser configuradas para transportar a VLAN 17 em modo `access`.

```
interface FastEthernet0/1
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/2
description maq_de_gestao
switchport access vlan 703
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/3
description maq_de_gestao
switchport access vlan 703
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/4
description maq_de_gestao
switchport access vlan 703
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/5
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/6
description maq_de_gestao
switchport access vlan 704
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/7
description maq_de_gestao
switchport access vlan 703
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/8
description maq_de_gestao
switchport access vlan 704
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
```

```

switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/9
description maq_de_gestao
switchport access vlan 705
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/10
description maq_de_gestao
switchport access vlan 705
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/11
description maq_de_gestao
switchport access vlan 705
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/12
description maq_de_gestao
switchport access vlan 705
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/13
description maq_de_gestao
switchport access vlan 17
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/14
description maq_de_gestao
switchport access vlan 17
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/15
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/16
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!

```

```
interface FastEthernet0/17
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/18
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/19
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/20
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/21
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/22
description maq_de_gestao
switchport trunk encapsulation dot1q
switchport trunk native vlan 701
switchport trunk allowed vlan 17,701-705
switchport mode trunk
no ip address
spanning-tree portfast
!
interface FastEthernet0/23
description rede_inseguros_2-cl
switchport access vlan 17
switchport mode dynamic desirable
no ip address
spanning-tree portfast
!
interface FastEthernet0/24
switchport access vlan 17
switchport mode dynamic desirable
no ip address
spanning-tree portfast
!
```

C | INSTALAÇÃO DO *DNSMASQ*

1. Configurar o repositório EPEL.

```
$ wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ sudo rpm -Uvh epel-release-6*.rpm
```

2. Instalar o dnsmasq.

```
$ yum install dnsmasq
```

3. Editar o ficheiro de configuração do dnsmasq, definindo em que interfaces este deve aceitar pedidos DNS.

```
$ vi /etc/dnsmasq.conf
```

```
##### dnsmasq.conf #####
interface=eth0
##### dnsmasq.conf #####
```

4. O dnsmasq utiliza as entradas disponíveis no ficheiro `/etc/hosts` para construir as respostas aos pedidos DNS. Assim, deve existir uma entrada por cada máquina, no seguinte formato:

Nota: Em caso de posterior adição de entradas, o serviço dnsmasq deve ser reiniciado, para que as alterações surtam efeito.

```
XXX.XXX.XXX.XXX maquinaX.gsiic.uc.pt maquinaX
```

5. Iniciar o dnsmasq.

```
$ service dnsmasq start
```

6. Configurar o dnsmasq para iniciar no arranque do sistema.

```
$ chkconfig dnsmasq on
```

NOTA: Para que uma máquina possa referenciar as outras pelo seu hostname, esta deve ter como seu default gateway, o router (máquina com o dnsmasq).

D | CONFIGURAÇÃO DO NAT

1. Habilitar IP forwarding.

```
$ vi /etc/sysctl.conf
```

```
net.ipv4.conf.default.forwarding=1
```

2. Reiniciar o serviço de rede.

```
$ service network restart
```

3. Adicionar regra de `postrouting` na firewall, de forma a que o tráfego seja encaminhado para o exterior (neste caso através da interface `eth0`). Para todas as máquinas em que pretenda conectividade com o exterior, o `default gateway` deve ser o endereço do router.

```
$ service iptables stop  
$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  
$ service iptables save  
$ service iptables restart  
$ chkconfig iptables on
```

E

INSTALAÇÃO DO *OPENVPN*

1. Configurar o repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o OpenVPN

```
$ yum install openvpn
```

3. Criar o ficheiro de configuração.

```
$ touch /etc/openvpn/server.conf
```

4. Editar o ficheiro de configuração.

```
port 1194 #- port
proto udp #- protocol
dev tun
tun-mtu 1500
tun-mtu-extra 32
mssfix 1450
reneg-sec 0
ca /etc/openvpn/easy-rsa/2.0/keys/ca.crt
cert /etc/openvpn/easy-rsa/2.0/keys/server.crt
key /etc/openvpn/easy-rsa/2.0/keys/server.key
dh /etc/openvpn/easy-rsa/2.0/keys/dh1024.pem
plugin /usr/share/openvpn/plugin/lib/openvpn-auth-pam.so /etc/pam.d/login
client-cert-not-required
username-as-common-name
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
keepalive 5 30
comp-lzo
persist-key
persist-tun
status 1194.log
verb 3
```

5. Copiar os ficheiros easy-rsa para a localização correcta.

```
$ mkdir -p /etc/openvpn/easy-rsa/keys
$ cp -rf /usr/share/openvpn/easy-rsa/2.0/* /etc/openvpn/easy-rsa
```

6. Editar o ficheiro de configuração.

```
$ vi /etc/openvpn/easy-rsa/2.0/vars
```

```
export KEY_CONFIG=/etc/openvpn/easy-rsa/2.0/openssl-1.0.0.cnf
```

7. Criar a Certificate Authority (CA).

```
$ cd /etc/openvpn/easy-rsa
$ source ./vars
$ ./clean-all
$ ./build-ca
```

8. Gerar o certificado para o servidor OpenVPN.

```
$ ./build-key-server server
```

9. Gerar Diffie Hellman.

```
$ ./build-dh
```

10. Gerar certificado para um cliente.

```
$ cd /etc/openvpn/easy-rsa
$ ./build-key client
```

11. Criar regra na Firewall para permitir o tráfego da VPN. **NOTA: Também é necessário que o IP Forwarding esteja habilitado. Não foi aqui incluído, por já ter sido habilitado aquando da configuração do NAT.**

```
$ iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
$ service iptables save
```

12. Iniciar o OpenVPN.

```
$ service openvpn start
```

13. Definir o OpenVPN para iniciar aquando do arranque do sistema.

```
$ chkconfig openvpn on
```

14. Criar ficheiro de configuração para um cliente. Deve ser dada especial atenção aos parâmetros `remote` (que deve apontar para o endereço público da máquina a correr o servidor OpenVPN), `ca` (deve conter o certificado da CA, ou seja o conteúdo do ficheiro `ca.crt`), `cert` (deve conter o certificado do cliente, ou seja o conteúdo do ficheiro `client.crt`) e `key` (deve conter a chave do cliente, ou seja o conteúdo do ficheiro `client.key`).

```
$ vi client.ovpn
```

```
client
dev tun
proto udp
remote 193.136.200.85 1194
resolv-retry infinite
nobind
persist-key
persist-tun
comp-lzo
verb 3
<ca>
Contents of ca.crt
</ca>
<cert>
Contents of client.crt
</cert>
<key>
Contents of client.key
</key>
```

15. Testar a ligação. Numa máquina cliente correr o comando.

```
# sudo openvpn --config client.ovpn
```

F

INSTALAÇÃO DO *NTPD*

1. Instalar o ntpd.

```
$ yum install ntp
```

2. Definir o ntpd para iniciar aquando do arranque do sistema.

```
$ chkconfig ntpd on
```

3. Editar o ficheiro de configuração do ntpd. Adicionar as entradas abaixo, caso não existam.

```
$ vi /etc/ntp.conf
```

```
restrict default ignore  
restrict 193.137.215.1 mask 255.255.255.245 nomodify notrap noquery  
server 193.137.215.1  
restrict 192.168.201.0 mask 255.255.255.0 nomodify notrap
```

4. Nos clientes, repetir todos os passos excepto o 3, que deve ser substituído pelo seguinte.

```
$ vi /etc/ntp.conf
```

```
server 192.168.201.100
```

5. Iniciar o ntpd.

```
$ service ntpd start
```

G

INSTALAÇÃO DO *PUPPET* E
FOREMAN

1. Configurar os repositórios EPEL e Puppet Labs.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ rpm -ivh http://yum.puppetlabs.com/el/6/products/i386/puppetlabs-release-6-7.noarch.rpm
```

2. Instalar o Puppet Server.

```
$ yum install puppet-server
```

3. Configurar o repositório Foreman.

```
$ yum -y install http://yum.theforeman.org/releases/1.4/el6/x86_64/foreman-release.rpm
```

4. Adicionar regras de Firewall, para permitir o acesso às portas utilizadas pelo Foreman e Puppet.

```
$ iptables -I INPUT -p udp --dport 69 -j ACCEPT
$ iptables -I INPUT -p tcp --dport 80 -j ACCEPT
$ iptables -I INPUT -p tcp --dport 443 -j ACCEPT
$ iptables -I INPUT -p tcp --dport 8140 -j ACCEPT
$ iptables -I INPUT -p tcp --dport 8081 -j ACCEPT
$ service iptables save
$ service iptables restart
$ chkconfig iptables on
```

5. Instalar o Foreman Installer.

```
$ yum -y install foreman-installer
```

6. Instalar o Foreman utilizando o Foreman Installer

```
$ foreman-installer
```

7. Instalar o Foreman Setup Plugin (auxilia na configuração do mecanismo de aprovisionamento).

```
$ yum -y install ruby193-rubygem-foreman_setup
```

8. Instalar pacote que corrige incompatibilidades entre o Foreman Setup Plugin e o Foreman v1.4.

```
$ yum -y install ruby193-rubygem-uglifier
```

9. Reiniciar o serviço do Foreman para detectar o novo plugin.

```
$ service foreman restart
```

10. Configurar o mecanismo de aprovisionamento, via dashboard, em Infrastructure -> Provisioning Setup.

11. Substituir o template "Default Kickstart template" em "Hosts" -> "Provisioning templates" pelo template que a seguir se apresenta. Aqui, o código responsável pelo NTP sync (utilizando o servidor definido no host parameter `ntp-server` ou o endereço 193.137.215.1, por defeito) é movido para a fase de pré-instalação do SO, evitando assim problemas ao obter os pacotes de instalação via SSL (devido à data/hora do sistema incorrectamente definidas). **NOTA: Em caso de os problemas durante o aprovisionamento persistirem, a hora do sistema deve ser definida manualmente.**

```
##### START OF Kickstart Default template #####
<%#
kind: provision
name: Community Kickstart
oses:
- CentOS 4
- CentOS 5
- CentOS 6
```

```

- CentOS 7
- Fedora 16
- Fedora 17
- Fedora 18
- Fedora 19
%>
<%
    rhel_compatible = @host.operatingsystem.family == 'Redhat' && @host.operatingsystem.name != 'Fedora'
    os_major = @host.operatingsystem.major.to_i
    # safemode renderer does not support unary negation
    pm_set = @host.puppetmaster.empty? ? false : true
    puppet_enabled = pm_set || @host.params['force-puppet']
%>

install
<%= @mediapath %>
lang en_US.UTF-8
selinux --enforcing
keyboard us
skipx
network --bootproto <%= @static ? "static --ip=#{@host.ip} --netmask=#{@host.subnet.mask} --gateway=#{@host.subnet.gateway} --
nameserver=#{[@host.subnet.dns_primary,@host.subnet.dns_secondary].reject{|n| n.blank?}.join(',')}" : 'dhcp' %> --hostname <%=
@host %>
rootpw --iscrypted <%= root_pass %>
firewall --<%= os_major >= 6 ? 'service=' : '' %>ssh
authconfig --useshadow --passalgo=sha256 --kickstart
timezone <%= @host.params['time-zone'] || 'UTC' %>
<% if rhel_compatible && os_major > 4 -%>
services --disabled
autofs,gpm,sendmail,cups,iptables,ip6tables,auditd,aprtables_jf,xfs,pamcia,iscn,rawdevices,hpoj,bluetooth,openibd,avahi-
daemon,avahi-dnscnf,hidd,hplip,pcsd,restorecond,mcstrans,rhnsd,yum-updatesd
<% end -%>

%pre
#update local time
/usr/sbin/ntpdate -sub <%= @host.params['ntp-server'] || '193.137.215.1' %>
/usr/sbin/hwclock --systohc
%end

<% if @host.operatingsystem.name == 'Fedora' -%>
repo --name=fedora-everything --mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=fedora-<%=
@host.operatingsystem.major %>&arch=<%= @host.architecture %>
<% if puppet_enabled && @host.params['enable-puppetlabs-repo'] && @host.params['enable-puppetlabs-repo'] == 'true' -%>
repo --name=puppetlabs-products --baseurl=http://yum.puppetlabs.com/fedora/f<%= @host.operatingsystem.major %>/products/<%=
@host.architecture %>
repo --name=puppetlabs-deps --baseurl=http://yum.puppetlabs.com/fedora/f<%= @host.operatingsystem.major %>/dependencies/<%=
@host.architecture %>
<% end -%>
<% elsif rhel_compatible && os_major > 4 -%>
repo --name="Extra Packages for Enterprise Linux" --mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-<%=
@host.operatingsystem.major %>&arch=<%= @host.architecture %>
<% if puppet_enabled && @host.params['enable-puppetlabs-repo'] && @host.params['enable-puppetlabs-repo'] == 'true' -%>
repo --name=puppetlabs-products --baseurl=http://yum.puppetlabs.com/el/<%= @host.operatingsystem.major %>/products/<%=
@host.architecture %>
repo --name=puppetlabs-deps --baseurl=http://yum.puppetlabs.com/el/<%= @host.operatingsystem.major %>/dependencies/<%=
@host.architecture %>
<% end -%>
<% end -%>

<% if @host.operatingsystem.name == 'Fedora' and os_major <= 16 -%>
# Bootloader exception for Fedora 16:
bootloader --append="nofb quiet splash=quiet <%=ks_console%>" <%= grub_pass %>
part biosboot --fstype=biosboot --size=1
<% else -%>
bootloader --location=mbr --append="nofb quiet splash=quiet" <%= grub_pass %>
<% end -%>

<% if @dynamic -%>
%include /tmp/diskpart.cfg
<% else -%>
<%= @host.diskLayout %>
<% end -%>

text
reboot

%packages --ignoremissing
yum
dhclient
ntp
wget
@Core
epel-release
<% if puppet_enabled %>

```

```

puppet
<% if @host.params['enable-puppetlabs-repo'] && @host.params['enable-puppetlabs-repo'] == 'true' -%>
puppetlabs-release
<% end -%>
<% end -%>
%end

<% if @dynamic -%>


```

%pre
<%= @host.diskLayout %>
%end
<% end -%>

%post --nochroot
exec < /dev/tty3 > /dev/tty3
#changing to VT 3 so that we can see whats going on....
/usr/bin/chvt 3
(
cp -va /etc/resolv.conf /mnt/sysimage/etc/resolv.conf
/usr/bin/chvt 1
) 2>&1 | tee /mnt/sysimage/root/install.postnochroot.log
%end

%post
logger "Starting anaconda <%= @host %> postinstall"
exec < /dev/tty3 > /dev/tty3
#changing to VT 3 so that we can see whats going on....
/usr/bin/chvt 3
(

update all the base packages from the updates repository
yum -t -y -e 0 update

<% if puppet_enabled %>
echo "Configuring puppet"
cat > /etc/puppet/puppet.conf << EOF
<%= snippet 'puppet.conf' %>
EOF

Setup puppet to run on system reboot
/sbin/chkconfig --level 345 puppet on

/usr/bin/puppet agent --config /etc/puppet/puppet.conf -o --tags no_such_tag <%= @host.puppetmaster.blank? ? '' : "--server
#{@host.puppetmaster}" %> --no-daemonize
<% end -%>

sync

Inform the build system that we are done.
echo "Informing Foreman that we are built"
wget -q -O /dev/null --no-check-certificate <%= foreman_url %>
Sleeping an hour for debug
) 2>&1 | tee /root/install.post.log
exit 0

%end
END OF Kickstart Default template

```


```

12. Instalação das puppet classes do puppetdb. Na consola da máquina do puppet/foreman correr:

```
$ puppet module install puppetlabs-puppetdb
```

13. Via foreman, adicionar as classes puppetdb e puppetdb::master::config ao puppet/foreman host. Fazer override dos parâmetros puppetdb_port (usar a porta 8081) e puppet_service_name (usar httpd, uma vez que o foreman configura o puppetmaster para correr sobre o Apache Passenger).

14. Aplicar as alterações ao puppet/foreman host.

```
$ puppet agent -tv
```




INSTALAÇÃO DO *PERCONA* *XTRADB CLUSTER*

1. Instalar o repositório Percona

```
$ rpm -Uvh http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_64.rpm
```

2. Remover pacotes do MySQL, possivelmente já instalados, passíveis de gerarem conflitos.

```
$ rpm -e --nodeps mysql-libs
```

3. Instalar o Percona XtraDB Cluster

```
$ yum install Percona-XtraDB-Cluster-56
```

4. Criar o ficheiro de configuração my.cnf.

```
$ touch /etc/my.cnf
```

5. Editar o ficheiro de configuração. Deve ser tomada especial atenção aos parâmetros `server_id` (que deve ser único para cada um dos nós do cluster), `wsrep_cluster_address` (que, idealmente, deve ser constituído pelos IPs de todos os nós do cluster) e `wsrep_node_name` (que deve ser um nome único que identifique cada um dos nós do cluster, por exemplo, o FQDN).

```
$ vi /etc/my.cnf
```

```
##### my.cnf START #####
[mysqld]
server_id=3
binlog_format=ROW
log_bin=mysql-bin
wsrep_cluster_address=gcomm://192.168.201.104,192.168.201.106,192.168.201.108
wsrep_provider=/usr/lib64/libgalera_smm.so
datadir=/var/lib/mysql
wsrep_slave_threads=2
wsrep_cluster_name=gsiic_percona_cluster
wsrep_sst_method=xtrabackup
wsrep_node_name=c3.havana.gsiic.uc.pt
wsrep_sst_auth=root:Dreams936603?
log_slave_updates
innodb_autoinc_lock_mode=2
innodb_buffer_pool_size=400M
innodb_log_file_size=64M
##### my.cnf END #####
```

6. Iniciar o serviço do XtraDB Cluster.

```
$ service mysql start
```

NOTA: No caso de ser o primeiro nó do cluster, o comando deve ser substituído pelo seguinte:

```
$ service mysql start --wsrep-cluster-address="gcomm://"
```

7. Verificar que o serviço iniciou correctamente e que o nó foi adicionado ao cluster. O valor `wsrep_cluster_size` do output deve ser igual ao número de nós já adicionados ao cluster.

```
$ mysql -u root -p -e "show global status like 'wsrep%'"
```

8. Definir XtraDB Cluster para arrancar aquando do boot do sistema.

```
$ chkconfig mysql on
```

9. No caso de ser o primeiro nó do cluster, as tabelas internas MySQL devem ser inicializadas.

```
$ mysql_upgrade -u root -p
```

10. Tornar a instalação mais segura (remove utilizadores e tabelas de teste e redefine a password do root). **NOTA: A password do root deve ser a mesma em todos os nós do cluster!**

```
$ mysql_secure_installation
```

11. Instalar o script `clustercheck`, que verifica o estado do cluster e que será utilizado pelo HAProxy.

```
mysql -u root -p -e "grant process on *.* to 'clustercheckuser'@'localhost' identified by 'Dreams936603?';"  
mysql -u root -p -e "flush privileges;"
```

```
vi /usr/bin/clustercheck
```

```
MYSQL_USERNAME="{1-clustercheckuser}"  
MYSQL_PASSWORD="{2-Dreams936603?}"
```

```
yum -y install xinetd
```

```
vi /etc/xinetd.d/mysqlchk
```

```
##### START OF mysqlchk #####  
# default: on  
# description: mysqlchk  
service mysqlchk  
{  
# this is a config for xinetd, place it in /etc/xinetd.d/  
  disable = no  
  flags = REUSE  
  socket_type = stream  
  port = 9200  
  wait = no  
  user = nobody  
  server = /usr/bin/clustercheck  
  log_on_failure += USERID  
  only_from = 0.0.0.0/0  
  # recommended to put the IPs that need  
  # to connect exclusively (security purposes)  
  per_source = UNLIMITED  
}  
##### END OF mysqlchk #####
```

```
vi /etc/services
```

```
mysqlchk 9200/tcp # mysqlchk
```

```
service xinetd restart
```

12. Verificar que o script está a funcionar.

```
$ curl localhost:9200
```

I

INSTALAÇÃO DO *HAPROXY* E DO *KEEPALIVED*

1. Configurar o repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o keepalived.

```
$ yum install keepalived
```

3. Habilitar o binding de endereços IP não locais (VIP). Isto é, de endereços não atribuídos explicitamente a nenhuma interface.

```
$ echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf
$ sysctl -p
```

4. Editar a configuração do keepalived. Deve ser tida especial atenção aos parâmetros `state` (deve ser definido como `MASTER` num dos nós e como `SLAVE` no restante), `virtual_router_id` (identificador único do cluster, deve ser o mesmo em todos os nós do cluster) e `priority` (deve ser 101 no nó `MASTER` e 100 no `SLAVE`).

```
$ vi /etc/keepalived/keepalived.conf
```

```
##### BEGIN OF keepalived.conf #####
vrrp_script chk_haproxy {
    script "killall -0 haproxy" # verify the pid existence
    interval 2                  # check every 2 seconds
    weight 2                    # add 2 points of prio if OK
}

vrrp_instance VI_1 {
    interface eth0              # interface to monitor
    state MASTER
    virtual_router_id 51        # Assign one ID for this route
    priority 101                # 101 on master, 100 on backup
    virtual_ipaddress {
        192.168.201.222        # the virtual IP
    }
    track_script {
        chk_haproxy
    }
}
##### END OF keepalived.conf #####
```

5. Iniciar o keepalived.

```
$ service keepalived start
```

6. Definir o keepalived para iniciar aquando do arranque do sistema.

```
$ chkconfig keepalived on
```

7. Verificar que o bind do VIP foi correctamente efectuado. Correr no nó MASTER o seguinte comando.

```
$ ip a | grep -e inet.*eth0
```

8. Intalar o HAProxy

```
$ yum install haproxy
```

9. Editar o ficheiro de configuração do HAProxy, por forma a configurar todos os cluster necessários.

```
$ vi /etc/haproxy/haproxy.cfg
```

```
##### BEGIN OF haproxy.cf #####
global
    log 127.0.0.1 local0
    log 127.0.0.1 local1 notice
```

```

chroot /var/lib/haproxy
daemon
group haproxy
maxconn 4000
pidfile /var/run/haproxy.pid
user haproxy

defaults
    log global
    option tcplog
    option dontlognull
    maxconn 2000
    option redispatch
    retries 3
    timeout http-request 10s
    timeout queue 1m
    timeout connect 10s
    timeout client 1m
    timeout server 1m
    timeout check 10s
    option redispatch

listen stats :1936
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /
    stats auth admin:Dreams936603?

listen dashboard_cluster
    bind 192.168.201.222:443
    balance source
    option tcpka
    option httpchk
    option tcplog
    server c1 192.168.201.104:443 check inter 2000 rise 2 fall 5
    server c2 192.168.201.106:443 check inter 2000 rise 2 fall 5
    server c3 192.168.201.108:443 check inter 2000 rise 2 fall 5

listen galera_cluster
    bind 192.168.201.222:3306
    balance leastconn
    option httpchk
    server c1 192.168.201.104:3306 check port 9200 inter 2000 rise 2 fall 5
    server c2 192.168.201.106:3306 check port 9200 inter 2000 rise 2 fall 5
    server c3 192.168.201.108:3306 check port 9200 inter 2000 rise 2 fall 5

listen glance_api_cluster
    bind 192.168.201.222:9292
    balance source
    option tcpka
    option httpchk
    option tcplog
    server c1 192.168.201.104:9292 check inter 2000 rise 2 fall 5
    server c2 192.168.201.106:9292 check inter 2000 rise 2 fall 5
    server c3 192.168.201.108:9292 check inter 2000 rise 2 fall 5

listen glance_registry_cluster
    bind 192.168.201.222:9191
    balance source
    option tcpka
    option tcplog
    server c1 192.168.201.104:9191 check inter 2000 rise 2 fall 5
    server c2 192.168.201.106:9191 check inter 2000 rise 2 fall 5
    server c3 192.168.201.108:9191 check inter 2000 rise 2 fall 5

listen keystone_admin_cluster
    bind 192.168.201.222:35357
    balance source
    option tcpka
    option httpchk
    option tcplog
    server c1 192.168.201.104:35357 check inter 2000 rise 2 fall 5
    server c2 192.168.201.106:35357 check inter 2000 rise 2 fall 5
    server c3 192.168.201.108:35357 check inter 2000 rise 2 fall 5

listen keystone_public_internal_cluster
    bind 192.168.201.222:5000
    balance source
    option tcpka
    option httpchk
    option tcplog

```

```

server c1 192.168.201.104:5000 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:5000 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:5000 check inter 2000 rise 2 fall 5

listen nova_ec2_api_cluster
bind 192.168.201.222:8773
balance source
option tcpka
option tcplog
server c1 192.168.201.104:8773 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8773 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8773 check inter 2000 rise 2 fall 5

listen nova_compute_api_cluster
bind 192.168.201.222:8774
balance source
option tcpka
option httpchk
option tcplog
server c1 192.168.201.104:8774 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8774 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8774 check inter 2000 rise 2 fall 5

listen nova_metadata_api_cluster
bind 192.168.201.222:8775
balance source
option tcpka
option tcplog
server c1 192.168.201.104:8775 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8775 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8775 check inter 2000 rise 2 fall 5

listen cinder_api_cluster
bind 192.168.201.222:8776
balance source
option tcpka
option httpchk
option tcplog
server c1 192.168.201.104:8776 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8776 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8776 check inter 2000 rise 2 fall 5

listen ceilometer_api_cluster
bind 192.168.201.222:8777
balance source
option tcpka
option httpchk
option tcplog
server c1 192.168.201.104:8774 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8774 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8774 check inter 2000 rise 2 fall 5

listen spice_cluster
bind 192.168.201.222:6082
balance source
option tcpka
option tcplog
server c1 192.168.201.104:6080 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:6080 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:6080 check inter 2000 rise 2 fall 5

listen neutron_api_cluster
bind 192.168.201.222:9696
balance source
option tcpka
option httpchk
option tcplog
server c1 192.168.201.104:9696 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:9696 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:9696 check inter 2000 rise 2 fall 5

listen swift_proxy_cluster
bind 192.168.201.222:8080
balance source
option tcplog
option tcpka
server c1 192.168.201.104:8080 check inter 2000 rise 2 fall 5
server c2 192.168.201.106:8080 check inter 2000 rise 2 fall 5
server c3 192.168.201.108:8080 check inter 2000 rise 2 fall 5
##### END OF haproxy.cf #####

```

10. Iniciar o HAProxy.

```
$ service haproxy start
```

11. Definir HAProxy para iniciar aquando do arranque do sistema.

```
$ chkconfig haproxy on
```

12. Visitar o dashboar do HAProxy para verificar que tudo se encontra em correcto funcionamento (utilizar o VIP `http://<VIP>:1936`).



INSTALAÇÃO DO *RABBITMQ*

1. Configurar repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar dependências do RabbitMQ

```
$ yum install erlang
```

3. Download do pacote de instalação do RabbitMQ.

```
$ wget http://www.rabbitmq.com/releases/rabbitmq-server/v3.3.0/rabbitmq-server-3.3.0-1.noarch.rpm
```

4. Importação da chave pública do RabbitMQ.

```
$ rpm --import http://www.rabbitmq.com/rabbitmq-signing-key-public.asc
```

5. Instalação do RabbitMQ.

```
$ yum install rabbitmq-server-3.3.0-1.noarch.rpm
```

6. Iniciar o serviço rabbitmq em modo detached (não tenta adicionar-se a um cluster).

```
$ service rabbitmq-server start --detached
```

7. Verificar que o rabbitmq iniciou correctamente.

```
$ rabbitmqctl cluster_status
```

8. Sincronizar os cookies erlang (necessários para a comunicação entre nós do rabbitmq). O cookie deve ser o mesmo em todos os nós do cluster. A forma mais rápida de fazer isto é copiar o cookie de um dos nós para todos os outros.

```
$ scp root@cl:/var/lib/rabbitmq/.erlang.cookie /var/lib/rabbitmq
```

9. Definir a política de HA do RabbitMQ para utilizar sempre mirrored queues, e para as sincronizar automaticamente após reboot.

```
$ rabbitmqctl set_policy HA '.*' '{"ha-mode":"all", "ha-sync-mode": "automatic"}'
```

10. Habilitar o RabbitMQ Management Plugin (web dashboard).

```
$ rabbitmq-plugins enable rabbitmq_management
```

11. Juntar o nó ao cluster. **NOTA: Deve ser executado em todos os nós excepto o primeiro.**

```
$ rabbitmqctl stop_app  
$ rabbitmqctl join_cluster rabbit@cl  
$ rabbitmqctl start_app
```

12. Verificar que o nó foi correctamente adicionado.

```
$ rabbitmqctl cluster_status
```

13. Definir o rabbitmq para iniciar aquando do arranque do sistema.

```
$ chkconfig rabbitmq-server on
```

14. Criar um utilizador de administrador (necessário para aceder o dashboard de gestão). **NOTA: Deve ser executado apenas em um dos nós (a alteração será automaticamente replicada por todo o cluster).**

```
$ rabbitmqctl add_user admin Dreams936603?
```

15. Definir permissões para o utilizador `admin`. **NOTA: Deve ser executado apenas em um dos nós (a alteração será automaticamente replicada por todo o cluster).**

```
$ rabbitmqctl set_permissions admin ".*" ".*" ".*"
```

16. Definir as flags de `admin` e `management` para o utilizador `admin`. **NOTA: Deve ser executado apenas em um dos nós (a alteração será automaticamente replicada por todo o cluster).**

```
$ rabbitmqctl set_user_tags admin management  
$ rabbitmqctl set_user_tags admin administrator
```

K

INSTALAÇÃO DO *MEMCACHED*

1. Configurar o repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o memcached.

```
$ yum install memcached php-pecl-memcache python-memcached
```

3. Editar o ficheiro de configuração do memcached. Deve ser dada especial ao parâmetro OPTIONS (deve ser usado o endereço da interface de Controlo da máquina).

```
$ vi /etc/sysconfig/memcached
```

```
##### memcached BEGIN #####  
PORT="11211"  
USER="memcached"  
MAXCONN="4096"  
CACHE_SIZE="128"  
OPTIONS="-l 192.168.201.106"  
##### memcached END #####
```

4. Iniciar o memcached.

```
$ service memcached start
```

5. Definir o memcached para iniciar no arranque do sistema.

```
$ chkconfig memcached on
```

6. Verificar que o memcached se encontra em correcto funcionamento. O valor `accepting_conns` do output deve ser 1.

```
$ memcached-tool 192.168.201.106:11211 stats
```

L | INSTALAÇÃO DO *CEPH*

1. Configurar o repositório Ceph.

```
$ vi /etc/yum.repos.d/ceph.repo
```

```
[ceph-noarch]
name=Ceph noarch packages
baseurl=http://ceph.com/rpm-emperor/el6/noarch
enabled=1
gpgcheck=1
type=rpm-md
gpgkey=https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc
```

2. Instalar o ceph-deploy.

```
$ yum update && yum install ceph-deploy
```

3. Criar utilizador para o Ceph.

```
$ useradd -d /home/ceph -m ceph
$ passwd ceph
```

4. Configurar sudo sem password

```
$ echo "ceph ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/ceph
$ chmod 0440 /etc/sudoers.d/ceph
$ visudo /etc/sudoers
```

4.1. Comentar a linha Defaults requiretty

5. Alterar para o utilizador ceph.

```
$ su ceph
```

6. Configurar o acesso SSH sem password.

```
$ mkdir /home/ceph/ceph-cluster
$ cd /home/ceph/ceph-cluster
```

```
$ ssh-keygen (save the key on /home/ceph/.ssh/id_rsa)
```

```
$ vi /home/ceph/.ssh/config
```

```
Host c1
  Hostname c1
  User ceph
Host c2
  Hostname c2
  User ceph
Host c3
  Hostname c3
  User ceph
Host osd1
  Hostname osd1
  User ceph
Host osd2
  Hostname osd2
  User ceph
Host osd3
  Hostname osd3
  User ceph
```

```
ssh-copy-id ceph@c1
```

NOTA: Deve ser executado para cada um dos nós que irão correr serviços Ceph.

7. Criar um novo cluster com os nós c1, c2 e c3 como initial monitors.

```
$ ceph-deploy new c1 c2 c3
```

8. Instalar o ceph em todos os nós.

```
$ ceph-deploy install c1 c2 c3 osd1 osd2 osd3
```

9. Criar monitors nos nós definidos como initial monitors (c1, c2 e c3).

```
ceph-deploy mon create-initial
```

10. Limpar os discos a serem usados pelo Ceph para armazenamento de dados. Neste caso, caso nó tem dois discos a serem utilizados (sda e sdb). **NOTA: Este comando vai apagar todo o conteúdo dos discos.**

```
$ ceph-deploy disk zap osd1:sda
$ ceph-deploy disk zap osd1:sdb
$ ceph-deploy disk zap osd2:sda
$ ceph-deploy disk zap osd2:sdb
$ ceph-deploy disk zap osd3:sda
$ ceph-deploy disk zap osd3:sdb
```

11. Preparar os discos a serem utilizados pelo Ceph e respectivos journal. Neste caso iremos guardar o journal de cada disco no próprio disco. Esta configuração tem uma performance ligeiramente inferior, mas é muito mais simples de configurar e permite que os discos sejam hot-swapped.

```
$ ceph-deploy osd prepare osd1:sda
$ ceph-deploy osd prepare osd1:sdb
$ ceph-deploy osd prepare osd2:sda
$ ceph-deploy osd prepare osd2:sdb
$ ceph-deploy osd prepare osd3:sda
$ ceph-deploy osd prepare osd3:sdb
```

12. Activar os OSDs e adicioná-los ao cluster.

```
$ ceph-deploy osd activate osd1:/dev/sda1
$ ceph-deploy osd activate osd1:/dev/sdb1
$ ceph-deploy osd activate osd2:/dev/sda1
$ ceph-deploy osd activate osd2:/dev/sdb1
$ ceph-deploy osd activate osd3:/dev/sda1
$ ceph-deploy osd activate osd3:/dev/sdb1
```

13. Cria um MDS (Metadata Server) em cada um dos nós OSD.

```
$ ceph-deploy mds create osd1 osd2 osd3
```

14. Criar uma pool para armazenamento de imagens a ser utilizada pelo OpenStack (Glance). O valor 256 que se pode ver nos comandos corresponde ao número de placement groups. Uma boa heurística para este valor é obtida utilizando o resultado da fórmula $\text{number_of_osds} \times 100 / \text{replication_factor}$, arredondado à potência de 2 superior.

```
$ ceph osd pool create os_images 256 256
$ ceph osd pool create os_volumes 256 256
$ ceph osd pool create os_backups 256 256

$ ceph osd pool set os_images size 3
$ ceph osd pool set os_volumes size 3
$ ceph osd pool set os_backups size 3
```

15. Copiar o ficheiro de configuração para os restantes nós, de forma a que sejam todos iguais.

```
ssh c2 sudo tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
ssh c3 sudo tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
```

16. Criar utilizadores no Ceph para os vários serviços do OpenStack (cinder, glance e cinder-backup).

```
$ ceph auth get-or-create client.cinder mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=os_volumes, allow rx pool=images'
$ ceph auth get-or-create client.glance mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=os_images'
```



```
$ ceph auth get-or-create client.cinder-backup mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=os_backups'
```

17. Obtém o keyring de acesso ao Ceph e copia-o para os controladores OpenStack. **NOTA: Deve ser executado para cada um dos nós de controlo do OpenStack (c1, c2 e c3).**

```
$ ceph auth get-or-create client.glance | ssh c1 sudo tee /etc/ceph/ceph.client.glance.keyring
$ ssh c1 sudo chown glance:glance /etc/ceph/ceph.client.glance.keyring
$ ceph auth get-or-create client.cinder | ssh c1 sudo tee /etc/ceph/ceph.client.cinder.keyring
$ ssh c1 sudo chown cinder:cinder /etc/ceph/ceph.client.cinder.keyring
$ ceph auth get-or-create client.cinder-backup | ssh c1 sudo tee /etc/ceph/ceph.client.cinder-backup.keyring
$ ssh c1 sudo chown cinder:cinder /etc/ceph/ceph.client.cinder-backup.keyring
```

M

INSTALAÇÃO DO
KEYSTONE

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o Keystone

```
$ yum install openstack-keystone python-keystoneclient
```

3. Editar o ficheiro de configuração do Keystone.

```
$ vi /etc/keystone/keystone.conf
```

```
##### START OF keystone.conf #####
[DEFAULT]
admin_token = d479ab3b20f84e79436b
bind_host = 192.168.201.106
public_port = 5000
admin_port = 35357
compute_port = 8774
policy_file = policy.json
policy_default_rule = admin_required
debug = False
verbose = False
log_file = keystone.log
log_dir = /var/log/keystone
use_syslog = False

[sql]
connection = mysql://keystone:Dreams936603?@c0/keystone
idle_timeout = 30

[identity]
driver = keystone.identity.backends.ldap.Identity

[credential]
# driver = keystone.credential.backends.sql.Credential

[trust]
# driver = keystone.trust.backends.sql.Trust
# enabled = True

[os_inherit]
# enabled = False

[catalog]
driver = keystone.catalog.backends.sql.Catalog

[endpoint filter]
# driver = keystone.contrib.endpoint_filter.backends.sql.EndpointFilter
# return_all_endpoints_if_no_filter = True

[token]
driver = keystone.token.backends.sql.Token
provider = keystone.token.providers.uuid.Provider
expiration = 86400
caching = True

[cache]
enabled = False
# config_prefix = cache.keystone
# expiration_time = 600
backend = dogpile.cache.memcached
# Example format: <argname>:<value>
backend_argument = url:127.0.0.1:11211
# proxies =
# debug_cache_backend = False

[policy]
driver = keystone.policy.backends.rules.Policy

[ec2]
# driver = keystone.contrib.ec2.backends.sql.Ec2

[assignment]
driver = keystone.assignment.backends.sql.Assignment
caching = True
# cache time =
```

```

[oauth1]
# driver = keystone.contrib.oauth1.backends.sql.OAuth1
# request_token_duration = 28800
# access_token_duration = 86400

[ssl]
#enable = True
#certfile = /etc/keystone/pki/certs/ssl_cert.pem
#keyfile = /etc/keystone/pki/private/ssl_key.pem
#ca_certs = /etc/keystone/pki/certs/cacert.pem
#ca_key = /etc/keystone/pki/private/cakey.pem
#key_size = 1024
#valid_days = 3650
#cert_required = False
#cert_subject = /C=US/ST=Unset/L=Unset/O=Unset/CN=localhost

[signing]
#token_format =
#certfile = /etc/keystone/pki/certs/signing_cert.pem
#keyfile = /etc/keystone/pki/private/signing_key.pem
#ca_certs = /etc/keystone/pki/certs/cacert.pem
#ca_key = /etc/keystone/pki/private/cakey.pem
#key_size = 2048
#valid_days = 3650
#cert_subject = /C=US/ST=Unset/L=Unset/O=Unset/CN=www.example.com

[ldap]
url = ldap://router
user =
#cn=Manager,dc=ci,dc=uc,dc=pt
password = None
suffix = dc=ci,dc=uc,dc=pt
use_dumb_member = False
allow_subtree_delete = False
# dumb_member = cn=dumb,dc=example,dc=com
page_size = 0
# alias_dereferencing = default
query_scope = sub

user_tree_dn = ou=rh,ou=users,dc=ci,dc=uc,dc=pt
#user_filter = (memberof=cn=gestor_ciuc,ou=groups,dc=ci,dc=uc,dc=pt)
# user_filter =
user_objectclass = person
user_id_attribute = uid
user_name_attribute = uid
user_mail_attribute = mailLocalAddress
user_pass_attribute = userPassword
user_enabled_attribute = sn
# user_enabled_mask = 0
user_enabled_default = True
# user_attribute_ignore = default_project_id,tenants
# user_default_project_id_attribute =
user_allow_create = False
user_allow_update = False
user_allow_delete = False
# user_enabled_emulation = False
# user_enabled_emulation_dn =

# group_tree_dn =
# group_filter =
# group_objectclass = groupOfNames
# group_id_attribute = cn
# group_name_attribute = ou
# group_member_attribute = member
# group_desc_attribute = desc
# group_attribute_ignore =
# group_allow_create = True
# group_allow_update = True
# group_allow_delete = True

# ldap TLS options
# if both tls_cacertfile and tls_cacertdir are set then
# tls_cacertfile will be used and tls_cacertdir is ignored
# valid options for tls_req_cert are demand, never, and allow
# use_tls = False
# tls_cacertfile =
# tls_cacertdir =
# tls_req_cert = demand

# domain_additional_attribute_mapping =
# group_additional_attribute_mapping =
# role_additional_attribute_mapping =

```

```
# project_additional_attribute_mapping =
# user_additional_attribute_mapping =

[auth]
methods = external,password,token,oauth1
#external = keystone.auth.plugins.external.ExternalDefault
password = keystone.auth.plugins.password.Password
token = keystone.auth.plugins.token.Token
oauth1 = keystone.auth.plugins.oauth1.OAuth

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
# config_file = /usr/share/keystone/keystone-dist-paste.ini
##### END OF keystone.conf #####
```

4. Inicializar a base de dados do Keystone **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ openstack-db --init --service keystone --password Dreams936603?
```

NOTA: O `openstack-db` **irá emitir um warning dizendo que o** `mysqld` **não se encontra em execução (o nome do serviço do Percona XtraDB é apenas** `mysql` **) e perguntar se o deseja iniciar. Deve ser respondido que sim. O arranque do** `mysqld` **vai falhar (uma vez que não existe), mas este é o comportamento esperado, uma vez que o** `openstack-db` **continuará a sua execução de qualquer das formas.**

5. Criar chaves e certificados para PKI. **NOTA: Isto deve ser executado apenas uma vez. Todos os nós devem partilhar a mesma chave/certificado.**

```
$ keystone-manage pki_setup --keystone-user keystone --keystone-group keystone
$ scp -r root@cl:/etc/keystone/ssl /etc/keystone/ssl
```

6. Definir o utilizador `keystone` como owner dos ficheiros do Keystone.

```
$ chown -R keystone:keystone /etc/keystone/* /var/log/keystone/keystone.log
```

7. Iniciar o Keystone.

```
$ service openstack-keystone start
```

8. Definir o Keystone para iniciar aquando do arranque do sistema.

```
$ chkconfig openstack-keystone on
```

9. Definir variáveis de ambiente necessárias para aceder manualmente ao Keystone. **NOTA: Isto é necessário apenas em um dos nós, que será utilizado para criar e configurar os users, tenants, roles, services, endpoints, etc...**

```
$ export OS_SERVICE_TOKEN=d479ab3b20f84e79436b
$ export OS_SERVICE_ENDPOINT=http://cl:35357/v2.0
```

10. Criar os tenants `admin` e `service`.

```
$ keystone tenant-create --name=admin --description="Admin Tenant"
$ keystone tenant-create --name=service --description="Service Tenant"
```

11. Criar o role `admin`.

```
$ keystone role-create --name=admin
```

12. Adicionar o role `admin` a algum utilizador, que no tenant `admin` quer no `service`.

```
$ keystone user-role-add --user=uc2008108818 --tenant=admin --role=admin
$ keystone user-role-add --user=uc2008108818 --tenant=service --role=admin
```

13. Registrar o serviço Keystone, no próprio Keystone. **NOTA: Isto deve ser executado**

apenas no primeiro nó.

```
$ keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"
```

14. Registrar o endpoint do Keystone. **NOTA: O `service-id` é o id devolvido pelo comando anterior. NOTA: Isto deve ser executado apenas no primeiro nó.**

```
$ keystone endpoint-create --service-id=c0a9c35895084223a4389096f1f41213 --publicurl=http://c0:5000/v2.0 --internalurl=http://c0:5000/v2.0 --adminurl=http://c0:35357/v2.0
```

15. Verificar que tudo se encontra em correcto funcionamento. **NOTA: Isto deve ser executado em cada um dos nós, para garantir que todos estão a funcionar correctamente. The `auth-url` deve ser alterado em conformidade (deve ser configurado para apontar para o endereço do nó e não para o VIP). NOTA: O comando `user-list` garante que o role `admin` foi atribuído correctamente (uma vez que o `user-list` é uma tarefa restrita apenas a administradores).**

```
$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
$ keystone --os-username=uc2008108818 --os-password=Emelindo9 --os-auth-url=http://c1:35357/v2.0 token-get
$ keystone --os-username=uc2008108818 --os-password=Emelindo9 --os-tenant-name=admin --os-auth-url=http://c2:35357/v2.0 token-get
$ keystone --os-username=uc2008108818 --os-password=Emelindo9 --os-tenant-name=service --os-auth-url=http://c2:35357/v2.0 token-get
$ keystone --os-username=uc2008108818 --os-password=Emelindo9 --os-tenant-name=admin --os-auth-url=http://c2:35357/v2.0 user-list
```

16. Definir variáveis de ambiente que serão utilizadas durante a configuração dos restantes serviços OpenStack. **NOTA: Por razões de segurança estas variáveis devem ser removidas após a conclusão do deployment.**

```
$ export OS_USERNAME=uc2008108818
$ export OS_PASSWORD=Emelindo9
$ export OS_TENANT_NAME=admin
$ export OS_AUTH_URL=http://c0:35357/v2.0
```

N

INSTALAÇÃO DO *GLANCE*

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o Glance.

```
$ yum install openstack-glance
```

3. Editar o ficheiro de configuração do glance-api. Deve ser tomada especial atenção com os parâmetros `bind_host` (que deve corresponder ao IP da interface de Controlo da máquina), `registry_host` (que deve ser o endereço do cluster `glance-registry`), `rabbit_password` (password do utilizador `admin` do RabbitMQ), `rabbit_hosts` (os endereços de todos os nós do cluster RabbitMQ), `rbd_store_user` (utilizar do Glance no Ceph), `rbd_store_pool` (pool do Glance no Ceph), `rbd_store_ceph_conf` (ficheiro keychain do utilizador do Glance no Ceph), `connection` (connection string do cluster de base de dados), `auth_host` (endereço do cluster Keystone), `admin_user` e `admin_password` (utilizador aplicacional do Glance no Keystone e respectiva password).

```
$ vi /etc/glance/glance-api.conf
```

```
##### START OF glance-api.conf #####
[DEFAULT]
bind_host = 192.168.201.108
debug = False
verbose = False
registry_host = c0
show_image_direct_url=True
scrubber_datadir = /var/lib/glance/scrubber
image_cache_dir = /var/lib/glance/image-cache/
log_file = /var/log/glance/api.log

# == RABBIT ==
notification_driver=rabbit
rabbit_port=5672
rabbit_virtual_host=/
rabbit_userid=admin
rabbit_password=Dreams936603?
rabbit_ha_queues=True
rabbit_hosts=c1:5672,c2:5672,c3:5672

# == RBD ==
default_store=rbd
rbd_store_user=glance
rbd_store_pool=os_images
rbd_store_ceph_conf=/etc/ceph/ceph.conf

[database]
backend=sqlalchemy
connection=mysql://glance:Dreams936603?@c0/glance
idle_timeout=30

[keystone_authtoken]
auth_host = c0
auth_port = 35357
admin_tenant_name = service
admin_user = glance
admin_password = Dreams936603?
auth_protocol = http

[paste_deploy]
flavor=keystone+cachemanagement
config_file = /etc/glance/glance-api-paste.ini
##### END OF glance-api.conf #####
```

4. Editar o ficheiro de configuração do glance-registry. Deve ser dada especial atenção aos parâmetros `bind_host` (deve corresponder ao IP da interface de Controlo da máquina), `connection` (connection string do cluster de base de dados), `auth_host` (endereço do cluster Keystone), `admin_user` e `admin_password` (utilizador aplicacional do Glance no Keystone e respectiva password).

```
$ vi /etc/glance/glance-registry.conf
```



```
##### START OF glance-registry.conf #####
[DEFAULT]
debug = False
verbose = True
bind_host = 192.168.201.108
log_file = /var/log/glance/registry.log

[database]
backend=sqlalchemy
connection = mysql://glance:Dreams936603?@c0/glance
idle_timeout = 30

[keystone_authtoken]
admin_tenant_name = service
admin_user = glance
admin_password = Dreams936603?
auth_host = c0
auth_port = 35357
auth_protocol = http

[paste_deploy]
config_file = /etc/glance/glance-registry-paste.ini
flavor=keystone
##### END OF glance-registry.conf #####
```

5. Inicializar a base de dados `glance`. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ openstack-db --init --service glance --password Dreams936603?
```

NOTA: O `openstack-db` irá emitir um warning dizendo que o `mysqld` não se encontra em execução (o nome do serviço do Percona XtraDB é apenas `mysql`) e perguntar se o deseja iniciar. Deve ser respondido que sim. O arranque do `mysqld` vai falhar (uma vez que não existe), mas este é o comportamento esperado, uma vez que o `openstack-db` continuará a sua execução de qualquer das formas. NOTA: Em caso de erro ao realizar a migração de v5 para v6 (`InvalidVersionError: 5 is not 6`) correr o comando: `glance -s /bin/sh -c "glance-manage db_sync"`

6. Criar o utilizador `glance` no directório LDAP. **NOTA: Esta acção deve ser executada apenas uma vez.**
7. Adicionar o role `admin` ao utilizador `glance` no tenant `service`. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ keystone user-role-add --user=glance --tenant=service --role=admin
```

8. Copiar os "paste files" para a localização correcta (apenas necessário em CentOS).

```
$ cp /usr/share/glance/glance-api-dist-paste.ini /etc/glance/glance-api-paste.ini
$ cp /usr/share/glance/glance-registry-dist-paste.ini /etc/glance/glance-registry-paste.ini
```

9. Definir o utilizador `glance` como owner dos ficheiros do Glance.

```
$ chown -R glance:glance /etc/glance/* /var/log/glance/*
```

10. Registar o serviço `glance` no Keystone, para que outros serviços o possam encontrar. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ keystone service-create --name=glance --type=image --description="Glance Image Service"
```

11. Registar o endpoint `glance` no Keystone. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados). NOTA: O `service-id` é o id devolvido no passo anterior.**

```
$ keystone endpoint-create --service-id=abd0417d4f8f4373833534ad9ffc15db \
--publicurl=http://c0:9292 --internalurl=http://c0:9292 --adminurl=http://c0:9292
```

12. Iniciar os serviços glance-api e glance-registry.

```
$ service openstack-glance-api start  
$ service openstack-glance-registry start
```

13. Definir ambos os serviços para iniciarem aquando do arranque do sistema.

```
$ chkconfig openstack-glance-api on  
$ chkconfig openstack-glance-registry on
```

14. Fazer o upload de uma imagem de teste para o Glance.

```
$ wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img  
$ glance image-create --name="Cirros 0.3.1" --disk-format=qcow2 --container-format=bare \  
--is-public=true < cirros-0.3.1-x86_64-disk.img
```

15. Confirmar que o upload da imagem foi concluído com sucesso.

```
$ glance index  
$ glance image-list
```

O

INSTALAÇÃO DO *CINDER*

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o Cinder.

```
$ yum install openstack-cinder
```

3. Editar o ficheiro paste.ini do Cinder.

```
$ vi /etc/cinder/api-paste.ini
```

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
service_protocol = http
service_host = c0
service_port = 5000
auth_host = c0
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = Dreams936603?
# signing_dir is configurable, but the default behavior of the authtoken
# middleware should be sufficient. It will create a temporary directory
# in the home directory for the user the cinder process is running as.
#signing_dir = /var/lib/cinder/keystone-signing
```

4. Editar o ficheiro de configuração do Cinder.

```
$ vi /etc/cinder/cinder.conf
```

```
[DEFAULT]
sql_idle_timeout=30
rpc_backend=cinder.openstack.common.rpc.impl_kombu
rabbit_ha_queues=True
rabbit_hosts=c1:5672,c2:5672,c3:5672
rabbit_userid=admin
rabbit_password=Dreams936603?
sql_connection = mysql://cinder:Dreams936603?@c0/cinder
osapi_volume_listen = 192.168.201.108
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_config = /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
volume_driver=cinder.volume.drivers.rbd.RBDDriver
rbd_pool=os_volumes
rbd_ceph_conf=/etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot=false
rbd_max_clone_depth=5
glance_api_version=2
rbd_user=cinder
#rbd_secret_uuid=457eb676-33da-42ec-9a8c-9293d545c337
backup_driver=cinder.backup.drivers.ceph
backup_ceph_conf=/etc/ceph/ceph.conf
backup_ceph_user=cinder-backup
backup_ceph_chunk_size=134217728
backup_ceph_pool=os_backups
backup_ceph_stripe_unit=0
backup_ceph_stripe_count=0
restore_discard_excess_bytes=true
```

5. Criar e inicializar base de dados.

```
$ mysql -u root -p -e "CREATE DATABASE cinder;"
$ mysql -u root -p -e "GRANT ALL ON cinder.* TO 'cinder'@ '%' IDENTIFIED BY 'Dreams936603?';"
$ mysql -u root -p -e "GRANT ALL ON cinder.* TO 'cinder'@ 'localhost' IDENTIFIED BY 'Dreams936603?';"
$ cinder-manage db sync
```

6. Atribuir o role de admin ao utilizador cinder nos tenants admin e service.

```
$ keystone user-role-add --user=cinder --tenant=service --role=admin
```

7. Registrar o serviço e endpoint do Cinder no Keystone.

```
$ keystone service-create --name=cinder --type=volume --description="OpenStack Block Storage"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ volume / {print $2}') \
  --publicurl=http://c0:8776/v1/%(tenant_id)s \
  --internalurl=http://c0:8776/v1/%(tenant_id)s \
  --adminurl=http://c0:8776/v1/%(tenant_id)s

$ keystone service-create --name=cinderv2 --type=volumev2 --description="OpenStack Block Storage v2"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ volumev2 / {print $2}') \
  --publicurl=http://c0:8776/v2/%(tenant_id)s \
  --internalurl=http://c0:8776/v2/%(tenant_id)s \
  --adminurl=http://c0:8776/v2/%(tenant_id)s
```

8. Iniciar os serviços do Cinder.

```
$ service openstack-cinder-scheduler start
$ service openstack-cinder-volume start
$ service openstack-cinder-backup start
$ service openstack-cinder-api start
```

9. Definir os serviços do Cinder para iniciarem aquando do arranque do sistema.

```
$ chkconfig openstack-cinder-volume on
$ chkconfig openstack-cinder-backup on
$ chkconfig openstack-cinder-scheduler on
$ chkconfig openstack-cinder-api on
```

10. Criar um volume de testes para garantir que a configuração foi concluída com sucesso.

```
$ cinder create --display-name debugVolume3 1
$ cinder list
```

P

INSTALAÇÃO DO NOVA
CONTROLLER

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar os pacotes dos serviços do Nova.

```
$ yum install openstack-nova-api openstack-nova-conductor openstack-nova-scheduler openstack-nova-novncproxy openstack-nova-cert openstack-nova-console python-novaclient
```

NOTA: O pacote `openstack-nova-console` só deve ser instalado em CentOS. O repositório RDO junta o `nova-console` e o `nova-consoleauth` no mesmo pacote, embora sejam serviços muito distintos. Quando possível só deve ser instalado o `nova-consoleauth`.

3. Editar o ficheiro `paste.ini` do `nova-api`.

```
$ vi /etc/nova/api-paste.ini
```

```
##### START OF api-paste.ini #####
#####
# Metadata #
#####
[composite:metadata]
use = egg:Paste#urlmap
/: meta

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#####
# EC2 #
#####

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory

[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

#####
# Openstack #
#####
```

```

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2
/v3: openstack_compute_api_v3

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_compute_app_v2
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit osapi_compute_app_v2
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext osapi_compute_app_v2

[composite:openstack_compute_api_v3]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth_v3 ratelimit_v3 osapi_compute_app_v3
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit_v3 osapi_compute_app_v3
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext osapi_compute_app_v3

[filter:faultwrap]
paste.filter_factory = nova.api.openstack.FaultWrapper.factory

[filter:noauth]
paste.filter_factory = nova.api.openstack.auth.NoAuthMiddleware.factory

[filter:noauth_v3]
paste.filter_factory = nova.api.openstack.auth.NoAuthMiddlewareV3.factory

[filter:ratelimit]
paste.filter_factory = nova.api.openstack.compute.limits.RateLimitingMiddleware.factory

[filter:ratelimit_v3]
paste.filter_factory = nova.api.openstack.compute.plugins.v3.limits.RateLimitingMiddleware.factory

[filter:sizelimit]
paste.filter_factory = nova.api.sizelimit.RequestBodySizeLimiter.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute.APIRouter.factory

[app:osapi_compute_app_v3]
paste.app_factory = nova.api.openstack.compute.APIRouterV3.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions.Versions.factory

#####
# Shared #
#####

[filter:keystonecontext]
paste.filter_factory = nova.api.auth.NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = c0
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = Dreams936603?
auth_version = v2.0
##### END OF api-paste.ini #####

```

4. Editar o ficheiro de configuração do Nova.

```
$ vi /etc/nova/nova.conf
```

```

##### START OF nova.conf #####
[DEFAULT]
log_dir = /var/log/nova
state_path = /var/lib/nova
lock_path = /var/lib/nova/tmp
dhcpbridge = /usr/bin/nova-dhcpbridge
dhcpbridge_flagfile = /etc/nova/nova.conf
force_dhcp_release = True
injected_network_template = /usr/share/nova/interfaces.template
libvirt_nonblocking = True

```



```

libvirt_inject_partition = -1
libvirt_use_virtio_for_bridges = True
network_manager = nova.network.manager.VlanManager
compute_driver = libvirt.LibvirtDriver
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
rootwrap_config = /etc/nova/rootwrap.conf
iscsi_helper = tgtadm
ec2_private_dns_show_ip = True
api_paste_config = /etc/nova/api-paste.ini
image_service = nova.image.glance.GlanceImageService
glance_api_servers = c0:9292
service_down_time = 60
ec2_listen=192.168.201.104
enabled_apis=ec2,osapi_compute,metadata
osapi_compute_listen=192.168.201.104
volume_api_class=nova.volume.cinder.API
auth_strategy=keystone
novncproxy_port=6080
novncproxy_base_url=http://c0:6080/vnc_auto.html
novncproxy_host=192.168.201.104

rpc_backend=nova.rpc.impl_kombu
rabbit_ha_queues=True
rabbit_hosts=c1:5672,c2:5672,c3:5672
rabbit_virtual_host=/
rabbit_userid=admin
rabbit_password=Dreams936603?

[database]
connection = mysql://nova:Dreams936603?@c0/nova
max_retries = -1
idle_timeout = 30
##### END OF nova.conf #####

```

5. Definir o utilizador `nova` como owner dos ficheiros do Nova.

```
$ chown -R nova:nova /etc/nova/* /var/log/nova/*
```

6. Criar a base de dados do Nova. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```

$ mysql -u root -p -e "CREATE DATABASE nova;"
$ mysql -u root -p -e "GRANT ALL ON nova.* TO 'nova'@ '%' IDENTIFIED BY 'Dreams936603?';"
$ mysql -u root -p -e "GRANT ALL ON nova.* TO 'nova'@ 'localhost' IDENTIFIED BY 'Dreams936603?';"

```

7. Criar o utilizador `nova` no LDAP.
8. Adicionar o role `admin` ao utilizador `nova` no tenant `service`. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ keystone user-role-add --user=nova --tenant=service --role=admin
```

9. Registar o serviço do Nova no Keystone. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ keystone service-create --name=nova --type=compute --description="Nova Compute service"
```

10. Registar o endpoint do Nova no Keystone. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```

$ keystone endpoint-create --service-id=65d4aa160f064cb4979b7d099b9e432b --publicurl=http://c0:8774/v2/%(tenant_id)s --
internalurl=http://c0:8774/v2/%(tenant_id)s --adminurl=http://c0:8774/v2/%(tenant_id)s

```

11. Atualizar o schema da base de dados do Nova para a versão mais recente. **NOTA: Este comando deve ser executado em apenas um dos nós (as alterações serão automaticamente replicadas por todo o cluster de base de dados).**

```
$ nova-manage db sync
```

NOTA: Pode ter de ser executado várias vezes até que termine sem erros.

12. Iniciar os serviços.

```
$ service openstack-nova-api start
$ service openstack-nova-cert start
$ service openstack-nova-consoleauth start
$ service openstack-nova-scheduler start
$ service openstack-nova-conductor start
$ service openstack-nova-novncproxy start
```

13. Definir os serviços para iniciarem aquando do arranque do sistema.

```
$ chkconfig openstack-nova-api on
$ chkconfig openstack-nova-cert on
$ chkconfig openstack-nova-consoleauth on
$ chkconfig openstack-nova-scheduler on
$ chkconfig openstack-nova-conductor on
$ chkconfig openstack-nova-novncproxy on
```

14. Verificar que todos os serviços se encontram em funcionamento.

```
$ nova-manage service list
```

NOTA: Este comando pode mostrar o nova-console como não estando em execução. Este é o comportamento esperado.

```
$ nova image-list
```

Q

INSTALAÇÃO DO *HORIZON*

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm  
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar o Horizon

```
$ yum install python-memcached mod_wsgi openstack-dashboard
```

3. Editar o ficheiro de configuração do Horizon e configurar os parâmetros seguintes (alguns deles já existem, pelo que devem ser alterados). Deve ser dada especial atenção aos parâmetros `CACHES` (o valor de `LOCATION` deve ser o endereço do cluster memcached) e `OPENSTACK_HOST` (que deve ser o endereço do cluster Keystone).

```
$ vi /etc/openstack-dashboard/local_settings
```

```
CACHES = {  
    'default': {  
        'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',  
        'LOCATION' : 'c0:11211'  
    }  
}  
  
TIME_ZONE = "GMT"  
ALLOWED_HOSTS = ['*']  
OPENSTACK_HOST = "c0"  
LOGIN_REDIRECT_URL = '/'  
LOGIN_URL = '/auth/login/'  
LOGOUT_URL = '/auth/logout/'  
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "_member_"
```

4. Mudar o Horizon para o root vhost do Apache.

```
$ vi /etc/httpd/conf.d/openstack-dashboard.conf
```

```
WSGIScriptAlias / /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi
```

5. Iniciar o serviço `apache`.

```
$ service httpd start
```

6. Definir o serviço para iniciar aquando do arranque do sistema.

```
$ chkconfig httpd on
```

R

INSTALAÇÃO DO *KVM*

1. Habilitar a tecnologia de virtualização (Intel VT ou AMD AMD-V) na BIOS. A forma de o fazer varia de servidor para servidor.
2. Configurar o repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

3. Instalar o KVM e o libvirt.

```
$ yum install kvm kmod-kvm python-virtinst libvirt libvirt-python libguestfs-tools bridge-utils
```

4. Habilitar o módulo do Kernel correspondente ao modelo do processador (`kvm-intel` ou `kvm-amd`).

```
$ modprobe kvm-intel
```

5. Verificar que o módulo foi habilitado.

```
$ /sbin/lsmmod | grep kvm
```

6. Iniciar o libvirt.

```
$ service libvirtd start
```

7. Definir o libvirtd para iniciar aquando do arranque do sistema.

```
$ chkconfig libvirtd on
```

S

INSTALAÇÃO DO
NOVA-COMPUTE E
NOVA-NETWORK

1. Configurar os repositórios EPEL e RDO (Red Hat OpenStack).

```
$ yum install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Verificar que a virtualização por hardware se encontra habilitada na BIOS.

3. Instalar o KVM e o libvirt.

```
$ yum install kvm kmod-kvm python-virtinst libvirt libvirt-python libguestfs-tools bridge-utils
```

4. Habilitar o módulo do kernel do KVM.

```
$ modprobe kvm-intel
```

5. Verificar que o módulo foi habilitado com sucesso.

```
$ /sbin/lsmmod | grep kvm
```

6. Criar um bridge e tornar a interface de Gestão slave dessa mesma bridge.

7. Configurar repositórios do Ceph.

```
$ vi /etc/yum.repos.d/ceph-extras-source.repo
```

```
[ceph-extras-source]
name=Ceph Extra Packages and Backports Sources
baseurl=http://ceph.com/packages/ceph-extras/rpm/centos6/SRPMS
enabled=1
gpgcheck=1
type=rpm-md
gpgkey=https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc
```

```
$ vi /etc/yum.repos.d/ceph-extras.repo
```

```
[ceph-extras]
name=Ceph Extra Packages and Backports $basearch
baseurl=http://ceph.com/packages/ceph-extras/rpm/centos6/$basearch
enabled=1
gpgcheck=1
type=rpm-md
gpgkey=https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc
```

```
$ vi /etc/yum.repos.d/ceph-extras-noarch.repo
```

```
[ceph-extras-noarch]
name=Ceph Extra Packages and Backports noarch
baseurl=http://ceph.com/packages/ceph-extras/rpm/centos6/noarch
enabled=1
gpgcheck=1
type=rpm-md
gpgkey=https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc
```

8. Uma vez que o qemu-img 0.12.1 (versão disponível no repositório EPEL) não suporta Ceph, é necessário reinstalá-lo a partir dos repositórios do Ceph.

```
$ yum remove qemu-img

$ yum --disablerepo=* --enablerepo=ceph-extras install -y qemu-img
$ yum --disablerepo=* --enablerepo=ceph-extras install -y qemu-kvm
$ yum --disablerepo=* --enablerepo=ceph-extras install -y qemu-guest-agent
$ yum --disablerepo=* --enablerepo=ceph-extras install -y qemu-kvm-tools
```

9. Verificar que o libvirt não foi desinstalado devido ao qemu-img.

```
$ yum install libvirt
```

10. Definir o libvirtd para iniciar aquando do arranque do sistema.


```
$ chkconfig libvirtd on
```

11. Iniciar o libvirtd.

```
$ service libvirtd start
```

12. Instalar o nova-compute.

```
$ yum install -y openstack-nova-compute
```

13. Editar o ficheiro de configuração do nova-compute.

```
$ vi /etc/nova/nova-compute.conf
```

```
[DEFAULT]
libvirt_type=kvm
compute_driver=libvirt.LibvirtDriver
```

14. Editar o ficheiro de configuração do Nova.

```
$ vi /etc/nova/nova.conf
```

```
[DEFAULT]
logdir = /var/log/nova
state_path = /var/lib/nova
lock_path = /var/lib/nova/tmp
volumes_dir = /etc/nova/volumes
libvirt_nonblocking = True
libvirt_inject_partition = -1
libvirt_use_virtio_for_bridges=True
api_paste_config=/etc/nova/api-paste.ini
iscsi_helper = tgtadm
sql_connection = mysql://nova:Dreams936603?@c0/nova
#compute_driver = libvirt.LibvirtDriver
memcached_servers=c0:11211
#rpc_backend = nova.openstack.common.rpc.impl_kombu
rpc_backend=rabbit
auth_strategy=keystone
rootwrap_config = /etc/nova/rootwrap.conf
rabbit_ha_queues=True
rabbit_hosts=c1:5672,c2:5672,c3:5672
rabbit_userid=admin
rabbit_password=Dreams936603?
rabbit_virtual_host=/
glance_api_servers=c0:9292
volume_api_class=nova.volume.cinder.API
vnc_enabled=False
#vncserver_listen=192.168.201.121
#vncserver_proxyclient_address=192.168.201.121
#novncproxy_base_url=http://c0:6080/vnc_auto.html

libvirt_images_type=rbd
libvirt_images_rbd_pool=volumes
libvirt_images_rbd_ceph_conf=/etc/ceph/ceph.conf
rbd_user=cinder
rbd_secret_uuid=837f2b09-9211-4254-9949-79e70e99ac9e

dhcpbridge = /usr/bin/nova-dhcpbridge
dhcpbridge_flagfile = /etc/nova/nova.conf
force_dhcp_release = True
injected_network_template = /usr/share/nova/interfaces.template
network_manager = nova.network.manager.VlanManager
vlan_interface=eth0
fixed_range=192.169.0.0/12
network_size=256
vlan_start=704
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver

libvirt_type=kvm
compute_driver=libvirt.LibvirtDriver
connection_type=libvirt

[Spice]
agent_enabled = True
html5proxy_base_url = http://c0:6080/spice_auto.html
server_listen = 0.0.0.0
```

```
server_proxyclient_address = 192.168.201.121
keymap = en-us

#[keystone_authtoken]
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%
#auth_host = 127.0.0.1
#auth_port = 35357
#auth_protocol = http
#signing_dir = /tmp/keystone-signing-nova
```

15. Obter a chave do utilizador cinder no Ceph (de forma a que o libvirt possa aceder aos volumes).

```
$ ceph auth get-key client.cinder | tee client.cinder.key
```

16. Gerar um UUID que identificará inequivocamente a chave. **NOTA: Este UUID deve ser usado em todos os nós de compute. NOTA: Este UUID é também aquele que deve ser utilizado no parâmetro rbd_secret_uuid do ficheiro de configuração nova.conf.** \$ uuidgen

17. Criar o secret.xml a utilizar pelo libvirt para importar a chave. **NOTA: A tag UUID deve conter o valor devolvido pelo comando anterior.** \$ cat > secret.xml <<EOF
client.cinder secret EOF

18. Importar a chave para o libvirt.

```
$ sudo virsh secret-define --file secret.xml
$ sudo virsh secret-set-value --secret <UUID anterior> --base64 $(cat client.cinder.key) && xm client.cinder.key secret.xml
```

19. Iniciar o nova-compute.

```
$ service openstack-nova-compute start
```

20. Definir o nova-compute para iniciar aquando do arranque do sistema.

```
$ chkconfig openstack-nova-compute on
```

21. Instalar o nova-network.

```
$ yum install openstack-nova-network
```

22. Habilitar o módulo do Kernel de suporte a VLANs.

```
$ modprobe 8021q
```

23. Adicionar o módulo a /etc/modules.

```
$ vi /etc/modules
```

```
8021q
```

24. Definir o nova-network para iniciar aquando o arranque do sistema.

```
$ chkconfig openstack-nova-network on
```

25. Iniciar o nova-network.

```
$ service openstack-nova-network start
```

26. Verificar que o nova-network se encontra em funcionamento, criando uma rede de teste.

```
nova-manage network create --fixed_range_v4=192.169.4.0/24 --vlan=704 --project_id="5c9db86fa2114e13ba293614b91d55e3"
```

T

INSTALAÇÃO DO *OPENLDAP*

1. Instalar o OpenLDAP

```
$ yum install openldap, openldap-servers, and openldap-clients
```

2. Editar o ficheiro de configuração do LDAP.

```
$ vi /etc/openldap/slapd.conf
```

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include      /etc/openldap/schema/corba.schema
include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/duaconf.schema
include      /etc/openldap/schema/dyngroup.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/java.schema
include      /etc/openldap/schema/misc.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/openldap.schema
include      /etc/openldap/schema/pmi.schema
include      /etc/openldap/schema/ppolicy.schema
include      /etc/openldap/schema/samba.schema
include      /etc/openldap/schema/collective.schema
include      /etc/openldap/schema/vacation.schema
include      /etc/openldap/schema/ciuc-mail.schema
include      /etc/openldap/schema/ciuc-priv.schema
include      /etc/openldap/schema/ciuc-samba.schema
include      /etc/openldap/schema/ciuc.schema
include      /etc/openldap/schema/ciuc-software.schema
include      /etc/openldap/schema/ciuc-www.schema

sizelimit    unlimited
allow bind_v2

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral     ldap://root.openldap.org

pidfile      /var/run/ldap/slapd.pid
argsfile     /var/run/ldap/slapd.args
logfile      /var/log/ldap/ldap_error.log

# Load dynamic backend modules:
# modulepath  /usr/local/ldap14/libexec/openldap
# moduleload  back_bdb.la
# moduleload  back_hdb.la
# moduleload  back_ldap.la

# Sample security restrictions
#   Require integrity protection (prevent hijacking)
#   Require 112-bit (3DES or better) encryption for updates
#   Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#   Root DSE: allow anyone to read it
#   Subschema (sub)entry DSE: allow anyone to read it
#   Other DSEs:
#       Allow self write access
#       Allow authenticated users read access
#       Allow anonymous users to authenticate
#   Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#   by self write
#   by users read
#   by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn. (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!
#
```

```
#####
# Certificats
#####
TLS cipher suite HIGH:MEDIUM:+TLSv1:!SSLv2:+SSLv3
TLSCertificateFile /etc/openldap/certs/cacert.pem
TLSCertificateKeyFile /etc/openldap/certs/serverkey.pem
TLSCertificateFile /etc/openldap/certs/servercert.pem
#TLSVerifyClient demand
TLSVerifyClient never

# SASL
sasl-host jazz
sasl-realm ipa.ci.uc.pt
sasl-secpops minssf=128

password-hash {CLEARTEXT}

#####
# BDB database definitions
#####
database bdb
suffix "dc=ci,dc=uc,dc=pt"
checkpoint 128 15
rootdn "cn=Manager,dc=ci,dc=uc,dc=pt"
# Cleartext passwords, especially for the rootdn, should
# be avoided. See slapd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw teste
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory /home/ldap14/
cachesize 1000000
idlcachesize 1000000

# Indices to maintain
index uid eq,sub
index none eq,sub
index o eq
index uidNumber eq
index mailgidNumber eq
index wwwgidNumber eq
index privgidNumber eq
index cn pres,sub,eq
index objectClass eq
index maillocaladdress eq
index mailroutingaddress eq
index ucid eq
index mailhost eq
index host eq
index sambadomainname eq
index sambaSID eq
index sambaPrimaryGroupSID eq
index default sub
#LDAP10 indexes
index sn eq,sub
index ou eq
index title eq
index gidNumber eq
index telephoneNumber eq
index employeeNumber eq
index employeeType eq
index mobile eq
index givenName eq,sub
index serialNumber eq
index binumber eq
index member eq
index backupMail eq
# O atributo memberUID NAO pode ser indexado, pois conduz &aacute; exaust&atilde;o da mem&oacute;ria
#index memberUid eq
index entryCSN eq
index entryUUID eq
index spamrules eq
index UCTelephoneNumber eq
index UCMobile eq
index deficiente eq
index ncc eq
index ncuc eq
index situacao eq
index UCAnoInicio eq
index UCAnoFim eq
index UCAnoFrequencia eq
```

```
index UCUltimoAnoLectivo eq
index UCPINEstudante eq
index UCIDMifare eq

# Let the replica DN have limitless searches
limits dn.exact="cn=Manager,dc=ci,dc=uc,dc=pt" time.soft=unlimited time.hard=unlimited size.soft=unlimited size.hard=unlimited

#access to attrs=shadowLastChange,sambaPwMustChange,sambaLMPassword,sambaPwLastSet,sambaNTPassword
# by dn="cn=Manager,dc=ci,dc=uc,dc=pt" write
# by self write
# by * none
```

3. Definir o OpenLDAP para iniciar aquando do arranque do sistema.

```
$ chkconfig slapd on
```

4. Exportar o directório para formato ldif

```
$ ldapsearch -Wx -D "cn=Manager,dc=ci,dc=uc,dc=pt" -b "dc=users,dc=ci,dc=uc,dc=pt" -H ldap://ldap-ext.ci.uc.pt -LLL >
ldap_gsiic_dump.ldif
```

5. Importar a cópia do repositório remoto para o repositório local.

```
$ ldapadd -Wx -D "cn=Manager,dc=ci,dc=uc,dc=pt" -H ldap://localhost -f ldap_gsiic_dump.ldif
```

6. Iniciar o OpenLDAP

```
$ service slapd start
```



INSTALAÇÃO E CONFIGURAÇÃO DO *ZABBIX*

1. Configurar repositório EPEL.

```
$ rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

2. Instalar os pacotes necessários.

```
$ yum install zabbix zabbix-agent zabbix-web
```

3. Criar e inicializar, no servidor MySQL, uma base de dados e um utilizador, com as permissões adequadas para o Zabbix.

```
$ mysql -u root -p -e "create database zabbix character set utf8;"
$ mysql -u root -p -e "grant all privileges on zabbix.* to 'zabbix'@'localhost' identified by 'Dreams936603?';"
$ mysql -u root -p -e "flush privileges;"

$ mysql -u zabbix -p zabbix < /usr/share/doc/zabbix-server-mysql-2.0.6/create/schema.sql
$ mysql -u zabbix -p zabbix < /usr/share/doc/zabbix-server-mysql-2.0.6/create/images.sql
$ mysql -u zabbix -p zabbix < /usr/share/doc/zabbix-server-mysql-2.0.6/create/data.sql
```

4. Alterar o fuso horário no ficheiro de configuração do Zabbix, no Apache.

```
$ vi /etc/httpd/conf.d/zabbix
```

```
php_value date.timezone Europe/Lisbon
```

5. Editar o ficheiro de configuração de Zabbix e configurar as credenciais de acesso à base de dados.

```
$ vi /etc/zabbix/zabbix_server.conf
```

6. Iniciar o Zabbix Server e defini-lo para iniciar automaticamente aquando do arranque do sistema.

```
$ service zabbix-server start
$ chkconfig zabbix-server on
```

7. Visitar o dashboard, em <http://192.168.201.100/zabbix>, e seguir as instruções para finalizar a instalação.

8. Instalar, nos nós de base de dados, o agente do Zabbix e o plugin de monitorização do PerconaXtraDB Cluster.

```
$ yum install zabbix-agent
$ yum install percona-zabbix-templates
```

9. Copiar o ficheiro de configuração do plugin, para a localização correcta.

```
$ mkdir -p /etc/zabbix_agentd.conf.d/
$ cp /var/lib/zabbix/percona/templates/userparameter_percona_mysql.conf
/etc/zabbix_agentd.conf.d/userparameter_percona_mysql.conf
```

10. Adicionar ao plugin as credenciais de acesso ao cluster Percona.

```
$ vi /var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php.cnf
```

```
$mysql_user = 'root';
$mysql_pass = 's3cret';
```

11. Fazer o download do plugin de monitorização para o Percona XtraDB Cluster, disponível em <http://www.percona.com/downloads/percona-monitoring-plugins/>, e importar o template para o Zabbix Server via dashboard.

12. Nos nós do Ceph, fazer o clone do repositório git do plugin para monitorização do Ceph.

```
$ git clone https://github.com/thelan/ceph-zabbix.git
```


13. Copiar os ficheiros de configuração para a localização correcta.

```
$ cd ceph-zabbix
$ cp zabbix_agent_ceph_plugin.conf /etc/zabbix_agentd.conf.d/zabbix_agent_ceph_plugin.conf
$ cp ceph-status.sh /opt/ceph-status.sh
```

14. Importar para o Zabbix Server, via dashboard, o ficheiro XML do template.

15. Nos nós do RabbitMQ, clonar o repositório git do plugin de monitorização para RabbitMQ.

```
$ git clone https://github.com/jasonmcintosh/rabbitmq-zabbix.git
```

16. Copiar os ficheiros de configuração para as localizações adequadas.

```
$ cd rabbitmq-zabbix
$ cp rabbitmq-zabbix/zabbix-rabbitmq.conf /etc/zabbix_agentd.conf.d/zabbix-rabbitmq.conf
$ cp -r script /etc/zabbix/
```

17. Criar um novo ficheiro de configuração com as credencias necessárias para aceder ao RabbitMQ.

```
$ vi /etc/zabbix/scripts/.rab.auth
```

```
USERNAME=admin
PASSWORD=Dreams936603?
CONF=/etc/zabbix/zabbix_agent.conf
```

18. Importar para o Zabbix Server, via dashboard, o ficheiro XML do template.



INSTALAÇÃO DO *TEMPEST*

1. Clonar o repositório do Rally (ferramenta que auxilia na instalação e configuração do Tempest).

```
$ git clone https://github.com/stackforge/rally.git
```

2. Instalar o Rally.

```
$ cd rally  
$ ./install_rally.sh
```

3. Definir as variáveis de ambiente, a utilizar na autenticação junto do Keystone.

```
$ export OS_USERNAME=uc2008108818  
$ export OS_PASSWORD=Ermelindo9  
$ export OS_TENANT_NAME=admin  
$ export OS_AUTH_URL=http://c0:35357/v2.0
```

4. Adicionar o cluster OpenStack ao Rally.

```
$ rally deployment create --fromenv --name GSIIC-CLOUD
```

5. Verificar que o cluster foi adicionado com sucesso.

```
$ rally deployment list  
$ rally deployment check
```

6. Instalar o tempest, usando o Rally.

```
$ rally-manage tempest install
```

7. Executar testes.

```
$ rally verify start
```

8. Listar resultado dos testes.

```
$ rally verify list
```