Master in Free Libre Open Source Software

Academic Course 2014/2015

Master Thesis

Implementation of a high availability solution based on Free Libre Open Source Software tools for Netnovation's Email and Collaboration System

Author: DANIEL H. GÁMEZ V.
Tutor: DR. GREGORIO ROBLES

Abstract

This is the Abstract...
Key words: Cluster, Corosync, CRM, DRBD, FOSS, FℓOSS, High Availability, OCF, Pacemaker, Zimbra

Acknowledgements

A huge recognition to the Free Libre Open Source Software and its wonderful community, thanks to the many opportunities and satisfactions it has given to me.

To the Universidad Rey Juan Carlos and its prestigious team of professors and academics, in particular to the Grupo de Sistemas y Comunicaciones (GSyC) and Libresoft who promote the movement of Free Libre Open Source Software with such passion and excellent quality.

Also to the Netnovation team and especially Eduardo Vítols, who has supported me for a long time. This project has been possible thanks to their innovation spirit.

To my beautiful country Venezuela, and also to this graceful land in which I reside, Spain. To my family and friends, who encourage me everyday to go forward.

Terminology

API: Application Programming Interface

CLI: Command Line Interface

cLVM: clustered Logical Volume Manager

CPU: Central Process Unit

CRM: Cluster Resource Manager / Customer Relationship Management

CTDB: Clustered Trivial Database

DRBD: Distributed Replicated Block Device

FℓOSS: Free Libre Open Source Software

GNU: GNU is Not Unix

GPL: General Public License

GUI: Graphic User Interface

HA: High Availability

HAWK: HA Web Konsole

HDD: Hard Disk Drive

IPv4: Internet Protocol version 4

ISO: International Organization for Standardization

IT: Information Technologies

KVM: Kernel-based Virtual Machine

LGPL: Lesser General Public License

LSB: Linux Standard Base

OBS: openSUSE Build Service

OCF: Open Cluster Framework

OCFS: Oracle Cluster File System

OS: Operating System

PBX: Private Branch Exchange

RHEL: Ret Hat Enterprise Linux

SAN: Storage Area Network

SCSI: Small Computer System Interface

SLA: Service Level Agreement

SME: Small and Medium Enterprises

VPN: Virtual Private Network

VPS: Virtual Private Server

WAN: Wide Area Network

YaST: Yet another Setup Tool

ZCS: Zimbra Collaboration System

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growing use of cloud-oriented systems and the need for this information to be always available, online systems play an increasingly important role in our society nowadays. The technologies that support such schemes have evolved fleetingly and today there are numerous ways to get this kind of solutions, from proprietary software implementations, tools oriented to corporate environments that can be even based on FℓOSS products and a variety of business models, to FℓOSS standalone tools representing a robust solution to meet today's demands.

Since 2004 Netnovation™ is a Venezuelan SME formed by a team of professionals in the areas of IT and telecommunications who adopted a business model based on consulting around FℓOSS, providing system integration, timely development and Software as a Service (SaaS) cloud services.

Due to a number of reasons that will be discussed throughout this dissertation, cloud services require a set of components to ensure the security, availability and reliability of data that is stored in data center facilities remotely accessible from internet. The study conducted here is focused specifically on the availability of the data that must be accessible to their applicants at all times.

This study was undertaken in the one hand to test some aspects of business practices in the current technology market and the use of FℓOSS as a key factor, and in the other one, to fulfill the demands of a business organization showing that business models around FℓOSS are a fact.

The way this dissertation is going to be organized is the following:

Chapter 2 with the problem statement, setting why there is an issue in the current situation, what are the justification and motivation of this dissertation, also defining the objectives and the proper scope.

Chapter 3 describing the most relevant related technologies around the possible solutions.

Chapter 4 establishing the used methodology, supported by guidelines proposed by Carlo Daffara and the Lazy User model, as well as analysis of concrete metrics.

Chapter 5 showing the architecture of the company and its infrastructure, available hardware for operations, the network scheme, and an overview of the software supporting the current situation, all of this in order to understand how to adapt an actual solution.

Chapter 6 describing the implemented technologies to achieve the solution for the problem stated.

Chapter 7 will provide a detailed description of the actual implementation for the proposed solution.

Chapter 8 with the results and discussion around the subject.

Finally in Chapter 9 the conclusions and future work will be raised.

Chapter 2

Problem statement

Business continuity in the field of information technology is supported in a large extent by the uninterrupted operation of the systems used in productivity tasks [6]. These systems must be fault tolerant, so that operations have the least possible impact in the event that an unexpected incident occurs.

Nowadays there are increasingly more people and organizations using centralized remote systems that allow online access to resources and everyday services, this scheme is called cloud computing [12]. Through this type of services, end users whether individuals or corporations, are abstracted to support the infrastructure that this entails, giving responsibility to intermediary companies providing cloud services. So these intermediaries are the ones who must ensure the proper availability of the services, as well as factors such as communications security and redundancy of stored data, among many others.

In particular Netnovation is a SME in the field of information technologies, which provides private cloud services from data storage to hosting virtual private servers (VPS), including e-mail and collaboration servers. The latter is precisely one of the mainstays for the operations of the company, which employs mainly FℓOSS to its internal systems, specifically using the FℓOSS e-mail and collaboration suite Zimbra™[1]. One of the main problems that Netnovation faces is to ensure the communication and workflow continuity that is carried through this collaboration tool, as well as meet the SLAs offered to its customers over this software.

There are various software solutions offering high availability of services such as those provided by Zimbra, each one with its own legal implications, associated costs and implementation difficulty[2]. A valid alternative is the integration of multiple tools in the field of FℓOSS providing a framework to ensure continuity of systems operation or business continuity. By doing it this way it is possible to adapt the different requirements and use different technologies to provide the most consistent solution to what is desired.

One key technology for this purpose is a cluster, in particular a high availability cluster, basically a a group of computers interconnected that work together trying to maintain a service up and running all the time.

[1] http://www.zimbra.com

[2] Some of these solutions will be addressed in Chapter 3

On previous occasions, Netnovation has managed to successfully consolidate most of its operations infrastructure adapting FℓOSS, making it a wonderful idea to keep this scheme working. To achieve this, it is necessary to evaluate the state of the art in the field of systems that provide high availability, with the aim of offering an effective solution, all of this in accordance to the guidelines that have been proposed by the company.

2.1 Justification / Motivation

The factors that motivate this work are on one hand, give proper credit to business models based on FℓOSS as those used by technology companies nowadays [3], and on the other hand, show that private enterprise can be benefited by FℓOSS through a set of toolsets and mechanisms which allow obtaining robust solutions in accordance to technology needs.

2.2 Objectives

The overall objectives are:

- To frame the FℓOSS business model used by Netnovation
- To show various current alternatives provided by FℓOSS at the corporate level
- To adapt the proposed solution to the guidelines established by Netnovation
- To establish an initial point reference for implementing high availability private cloud services offered by Netnovation

The specific objectives are:

- To implement a high availability solution based on FℓOSS for the e-mail and collaboration system Zimbra used by Netnovation
- To describe the methodology used for the selection of the solution to be implemented
- To describe the process undertaken to implement the selected solution
- To perform tests in a controlled laboratory environment and validate the correct operation of the solution in order to promote it to a production environment

2.3 Scope

The solution to be implemented consists of FℓOSS tools that allow its adaptation to the current infrastructure of Netnovation, they are not intended to replace the elements of the existing operations platform.

The methodology used for the selection of FℓOSS tools that make the proposed solution is not intended to provide an exhaustive process that considers all possibilities in the area, but

a flexible way that allows classify them qualitatively, justifying their choice through concrete metrics.

Having successfully implemented a high availability solution on the e-mail and collaboration system used by Netnovation, this will serve as a reference for providing high availability to other enterprise systems, but these other configurations are not covered in this exercise.

Chapter 3

Related technologies

A possible way to categorize high-availability technologies are Enterprise Solutions and FℓOSS based tools, considering that usually the first ones have associated support plans over the whole provided solution, whereas the second ones provide support over its own tools, but not necessarily over the whole cluster implementation.

3.1 Commercial Enterprise Cluster Software

Some of these high-availability implementations have been proprietary since the beginning, either way at some point commercial companies realized that these technologies could be integrated into open systems thanks to licenses such as GNU/LGPL. Since then, as a business model strategy, companies such as Hewlett Packard, Red Hat Inc. and SUSE (a Novell company) provide support plans over open system platforms, considering their entire solution and charging for business licenses, generally on annual basis. In the following sections there is a description for each of them.

3.1.1 HP Serviceguard

Hewlett Packard (HP) claims the credit for the development of the first high availability solution for UNIX systems since 1990[1]. MC/ServiceGuard is a high-availability cluster software released for HP-UX and later for GNU/Linux systems. Since the first development of the software, HP has partnered with with companies such as Oracle or SAP to deploy high-availability in enterprise environments, as Figure 3.1 (took from `http://www.hpintelco.net`) shows.

With the appearance of Linux in 1994 and its increasing popularity together with GNU, in 1999 HP released a Linux port called SG/LX, allowing high-availability features on it.

Since 2001, companies Intel, Red Hat and HP joined efforts to produce the Red Hat Open Source Solutions Initiative[2] (OSSI), with the aim to reduce partner's sale cycle by delivering enterprise reliable solutions to their customers. To some extent, realizing that their technologies could be strengthened by FℓOSS.

[1] `http://www.hpintelco.net/sglx/service.html`
[2] `http://www.hpintelco.net/hp-intel-redhat.htm`

Figure 3.1: High-availability race in enterprise environments

Some technical specifications of this product are the following:

- Proprietary Licensing Model[3]
- Active/active, active/standby, and rotating standby cluster types
- Quorum Server support
- 32 Fibre Channel nodes, 2 single-path SCSI nodes, and 32 multipath SCSI nodes
- Operating Systems RHEL and SLES
- Hardware HP ProLiant ML, DL, and BL G7, Gen8 and Gen9 servers
- HP 3PAR, EVA, StoreSure, EMC VMAX, VNX storage
- ext3, ext4, NFS, XFS, VxFS, VxVM and btrfs filesystems
- Logical Volume Manager

The general overview of an HP Serviceguard architecture is shown in Figure 3.2 (took from data sheet HP Serviceguard Solutions for Linux[4]).

[3]http://h20564.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c02199685
[4]http://h20195.www2.hp.com/v2/GetDocument.aspx?docname=4AA4-1792ENW

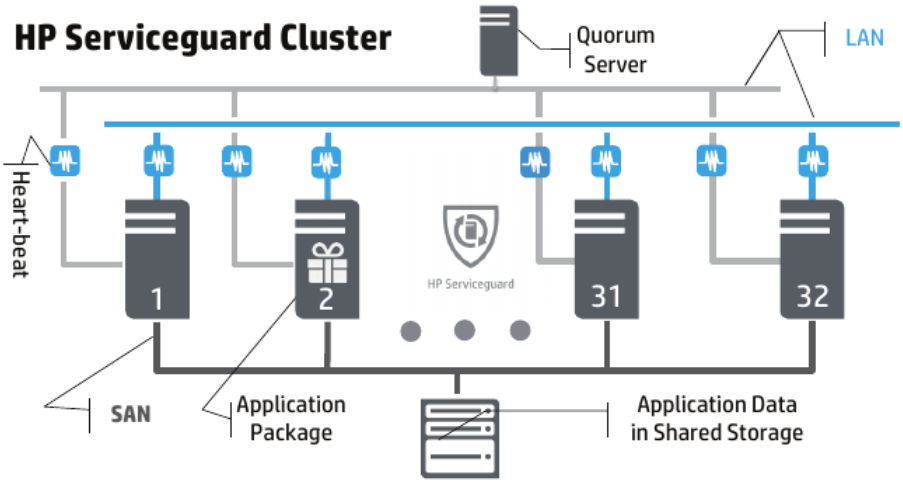

Figure 3.2: Basic HP Serviceguard for Linux cluster

3.1.2 Red Hat Cluster Suite

At the end of the 90's Red Hat Inc. introduced its Enterprise Linux Advanced Server, designed specifically for use in enterprise environments to deliver superior application support, performance, availability and scalability. It included a high-availability clustering feature as part of the base product. Since then, the product has evolved to the current Red Hat Cluster Suite (RHCS), provided as a separately licensed product, also on top of Red Hat's Linux Enterprise Server.

The RHCS has three major features, one of them is the Cluster Manager or **cman**, which adds functionality to, and is dependent upon other cluster stacks such as Corosync or OpenAIS. Is noteworthy that this is an adaptation of the Linux-HA project. Another key feature is the Resource Group Manager or **rgmanager**, a fully functional replacement for Pacemaker intended to work exclusively with RHCS. And also the IP Load Balancing (originally called Piranha), was originally developed by researchers at Oak Ridge National Laboratory, basically a text mining technology that Red Hat adapted in order to allow transparent load balancing and failover between servers[5].

The Cluster Manager, the Resource Group Manager and the IP Load Balancing are complementary high-availability technologies that can be used separately or in combination, depending on application requirements. Some of these technologies come from previous FℓOSS projects and have been properly integrated into RHCS.

Some technical details of this product are the following:

- Support for up to 128 nodes (16 nodes on Red Hat Enterprise Linux 3, 4, 5, and 6)
- NFS, CIFS, GFS share and cluster filesystem managers
- File system and services failover support

[5]http://www.ornl.gov/connect-with-ornl/for-industry/partnerships/technology-licensing/licensing-opportunity-announcements/piranha

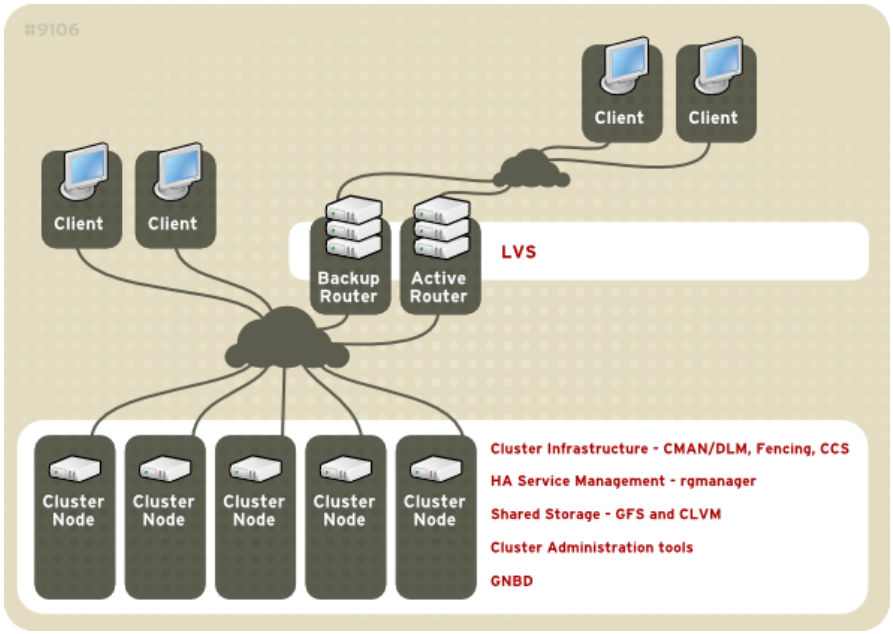

Figure 3.3: RHCS Basic Infrastructure

- Fully shared storage subsystem
- Comprehensive data integrity
- SCSI and fiber channel support
- OCF and LSB resource agents

A general RHCS infrastructure is shown in Figure 3.3 (took from RHCS online documentation[6]).

3.1.3 SUSE Linux Enterprise High Availability Extension

This is an integrated suite of clustering technologies that enables the implementation of high availability over physical and virtual Linux clusters [14]. It allows monitoring, messaging, and cluster resource management, handling failover and load balancing of resources.

As Red Hat did, other companies such as Novell took advantage of the high-availability demand over the enterprise sector and also adapted their own HA solution. This product is available as a paid add-on to SUSE Linux Enterprise Server GNU/Linux distribution, although in OpenSUSE many of these tools are included into the base system for free (without charge), with available repositories on OBS to provide newer versions of the packages for various GNU/Linux distributions[7].

Among the main product features of this product are the following:

[6] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/images/9106.png

[7] https://en.opensuse.org/openSUSE:High_Availability

- Multiple clustering scenarios, as active/active and active/passive configurations, as well as hybrid physical and virtual clusters
- Supports mixed clustering, physical and virtual Linux servers, based on Xen and KVM hypervisors
- Corosync messaging and membership layer, also Pacemaker cluster resource manager
- Storage and data replication supporting Fibre Channel or iSCSI SAN
- Cluster-aware file systems with GFS and OCFS, and cLVM as volume manager
- Supports replication through DRBD
- Samba clustering with CTDB
- Provides resource agent manager OCF
- GUI and CLI administration tools, such as YaST, HAWK and CRM

A SUSE HA cluster architecture is depicted in Figure 3.4 (took from SUSE Linux High Availability Extension online documentation[8]).

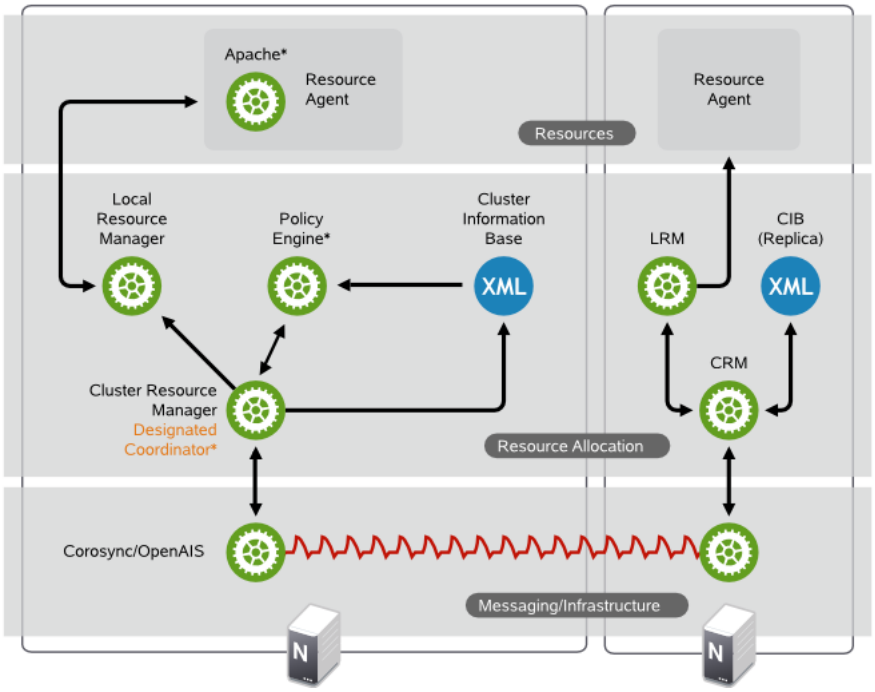

Figure 3.4: SUSE HA Cluster Components

[8]https://www.suse.com/documentation/sle_ha/book_sleha/graphics/ha_cluster_components_arch.png

3.2 High availability FℓOSS based tools

Some of these tools are possibly part of the enterprise products mentioned before, but they do not necessarily have an infrastructure providing corporate-oriented services. Either way, some of these FℓOSS technologies are backed by professional support over their standalone software, giving rise to business models such as product specialists [2].

To understand the comprehensive solution that these tools are capable to provide, it is appropriate to understand briefly how an HA cluster works and how its components have evolved since the appearance of this concept.

An HA cluster could be defined as a group of computers supporting server applications, ensuring its accessibility with a minimum of down-time. They operate by using specialized software that leverage the redundancy of computers and avoid single points of failure by implementing a cluster architecture [13].

Then, as long as there is a service that must be maintained up and running and accessible to the users, computers require a way to communicate to one another and coordinate between each other to provide this service. Here computers are running GNU/Linux operating system and rely on TCP/IP network protocol, as well as network interconnected hardware to achieve the communication between nodes in the cluster.

One of the earliest tools who managed this task was Heartbeat, as a daemon installed on each node, able to *talk* to the other ones and share cluster related information. This task was referred to as the Cluster Messaging Layer or Group Communication System.

Another important branch evolved, focused on the service or group of resources that the cluster is supposed to provide, for which Heartbeat incorporated a Cluster Resource Manager (CRM). A process to manage software resources, making use of scripts known as *Resource Agents* responsible to perform actions depending on the status of each node.

With these two distinct groups, one mostly concerned with the cluster messaging, and the other concerned with cluster resources, begins an important race of technologies seeking specialization on each area.

Heartbeat, part of the Linux-HA project, was first released in 1999 under GNU GPL and GNU LGPL license, maintaining a set of building blocks for high availability cluster systems, including a cluster messaging layer, a bunch of resource agents for a variety of applications, a plumbing library and an error reporting toolkit. Around 2007 this project evolved to Pacemaker, integrating or allowing interaction with multiple FℓOSS cluster stacks such as Corosync and OCF.

Nowadays Pacemaker is a resource manager responsible for starting and stopping cluster services in a proper way. Combined with other tools is able to detect service-level failures and move resources between cluster nodes as needed, to ensure the smooth operation of the services.

On the other hand Corosync is responsible for cluster membership, message passing and quorum, using the totem protocol for heartbeat, monitoring other node's health.

Technologies such as OpenAIS (a software API) also compete into the HA race. It is a FℓOSS implementation of the Application Interface Specification released under the terms of the Artistic License[9], used to define how HA applications work together, trying to mask hardware, operating system, middleware and application-level failures.

As in enterprise HA solutions, Cluster Managers are an important element within HA implementations, providing backend GUI or CLI software that runs on one or all cluster nodes, responsible for managing and controlling clustered services. Currently there are several FℓOSS alternatives, including the ones listed in Table 3.1.

Apache Mesos	
Company:	
License:	Apache License Version 2.0
Website:	http://mesos.apache.org/
Description:	
Commercial support:	
Keepalived	
Company:	
License:	GNU GPL v2+
Website:	http://keepalived.sourceforge.net/
Description:	
Commercial support:	
Linux Cluster Manager	
Company:	
License:	GNU GPL
Website:	http://linuxcm.sourceforge.net/
Description:	
Commercial support:	
oneSIS	
Company:	
License:	GNU GPL v2
Website:	http://onesis.org/
Description:	
Commercial support:	
Rocks Cluster Distribution	
Company:	
License:	BSD 3-Clause like
Website:	http://www.rocksclusters.org/
Description:	
Commercial support:	

[9] http://opensource.org/licenses/Artistic-2.0

SCMS.pro	
Company:	
License:	Apache 2 License
Website:	http://www.scms.pro/
Description:	
Commercial support:	
Ultra Monkey	
Company:	
License:	LGPL v2
Website:	http://www.ultramonkey.org/
Description:	
Commercial support:	
xCAT	
Company:	
License:	Eclipse Public License
Website:	http://xcat.org/
Description:	
Commercial support:	

Table 3.1: FℓOSS Cluster Managers

The cluster oriented filesystems are another key element regarding HA implementations. They provide data replication and fault tolerance, allowing operations continuity against incidents. There are several FℓOSS alternatives to provide clustered storage, including the tools listed in Table 3.2. One of the most consolidated clustered storage technologies is DRBD, first released in 1999, it will be aborded in Chapter 6.

Apache Hadoop	
Company:	
License:	Apache License 2
Website:	http://hadoop.apache.org/
Description:	
Commercial support:	
GFS	
Company:	
License:	GPL
Website:	http://sourceware.org/cluster/gfs/
Description:	
Commercial support:	

GlusterFS	
Company:	
License:	GPL v3
Website:	http://www.gluster.org/
Description:	
Commercial support:	
Lustre FS	
Company:	
License:	GPL
Website:	http://lustre.org/
Description:	
Commercial support:	
MooseFS	
Company:	
License:	GPL v2
Website:	https://moosefs.com/
Description:	
Commercial support:	
OCFS2	
Company:	
License:	GPL
Website:	https://oss.oracle.com/projects/ocfs2/
Description:	Oracle Cluster File System
Commercial support:	
XtreemFS	
Company:	
License:	BSD License
Website:	http://www.xtreemfs.org/
Description:	
Commercial support:	

Table 3.2: FℓOSS Cluster Storage Technologies

Chapter 4

Methodology

The followed roadmap to achieve the objectives outlined is a set of guidelines and suggestions for the adoption of FℓOSS within SMEs [2], in the sense that a methodology is not an exact formula but a set of practices. At using this model, companies find a supporting guide from the initial selection and adoption of FℓOSS within the IT infrastructure and even to the consolidation of business models around open source.

On the one hand, the guidelines proposed by Daffara suggest a research method by collecting and read as much information related to the project is available and select the appropriate solution from a matching set that fulfill the requirements. On the other hand, following a practical formal approach, applying a model of technology acceptance such as the Lazy User Model (LUM) [9] is possible to frame the process by which are chosen the technological tools that will make up the solution that meets user requirements, which in this case is being represented by the company Netnovation. This model focuses on user needs and the demanded effort when selecting a solution to a problem from a set of possible solutions, "According to the lazy user model, a user is likely to choose the solution that requires the least effort. The user examines this cost in terms of time, energy and money when considering how to use a new solution." [10] The LUM proposes that technology acceptance is impacted by this principle.

Additionally at using MetricsGrimoire [8], a flexible toolset which allow to obtain data from repositories related to software development, is possible to analyze databases that can later be mined for specific patterns or summaries of activity, allowing to establish objective comparisons in relation to the projects analyzed.

Chapter 5

Architecture

In order to provide IT services to customers, considering software services hosted in an on-line remote location, Netnovation requires a proper software and hardware infrastructure to operate. Some of these services are from VPS, Data Storage Systems, Customer Relationship Management Systems (CRM), Email and Collaboration Systems, to Voice over IP PBX systems. Both parties, software infrastructure and product services offered are based on FℓOSS.

In particular the Zimbra server to which a high availability schema is been configured, resides into this architecture, and it is consistent with the company's principles and business model, which is why it is useful to understand the environment to which it belongs.

5.1 Company Infrastructure

Currently the services are offered from two DataCenters (DC1 and DC2) geographically distributed with the aim of guarantee data redundancy. Assuming that communication with the main DC is lost, it has been defined a procedure that allows the restoration of services in the other DC, with the disadvantage that it is a manual procedure that requires administrators intervention.

5.2 Existent Hardware

Each DC has an average of seven Dell PowerEdge™ Racked Servers with different capacities, interconnected via communication devices that provide various services analogously.

There is a Dell PowerEdge 2850/2950 server serving as firewall and main router on each DC. On the one hand it has a WAN 1000Mbps interface, which is connected through UTP Cat-6 wired to 24 PoE ports Switches Netgear FS728TPv1 Gigabit. Physical servers are installed in 20U rack cabinets. These servers range from models Dell PowerEdge 1950, R510 to R710, have Intel Xeon CPUs within 24 and 64 cores, count with 8 to 64GB of RAM, and also have SCSI HDD with capacities between 100GB and 2,5TB.

5.3 Company Network Scheme

The housing services leased by the providers offer a pool of public IPv4 addresses that are handled by the main router on each DC facility. DC1 and DC2 are interconnected by a VPN through WAN, each of them associated to a different private Class B network internally. To the LAN Ethernet ports of the switches are connected the physical servers of the private network with transfer speed rates of 100/1000Mbps. The overall interconnection scheme can be appreciated in Figure 5.1.

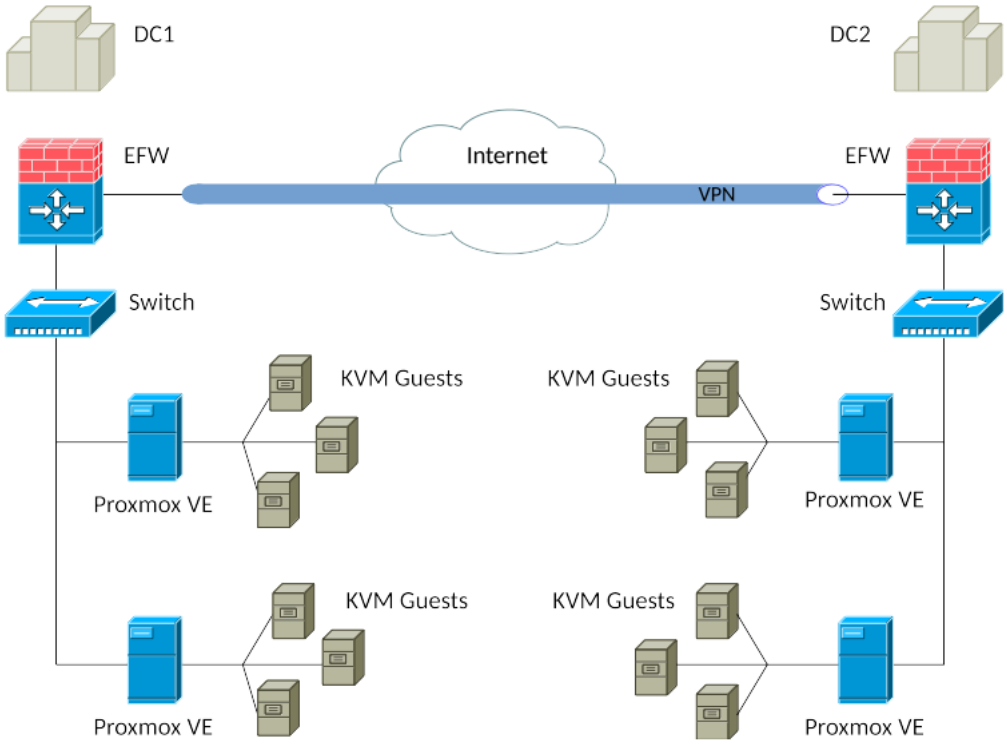

Figure 5.1: Network interconnection scheme

5.4 Software Supporting the Infrastructure

In Table 5.1 there are some software solutions currently used by Netnovation that are related with the required architecture to provide IT cloud-oriented services, with a brief description and legal licensing information for each one of them.

UTM Endian Firewall	
Company:	Endian S.r.l.
Industry:	Unified Threat Management
License:	GNU GPL
Website:	endian.com
Description:	A linux security distribution with full featured Unified Threat Management functionality. Include a stateful packet inspection firewall, application-level proxies for various protocols, antivirus support, virus and spam-filtering for email traffic, content filtering of Web traffic, also an OpenVPN solution. Distribution based on Red Hat.
Supported Platforms:	GNU/Linux
Commercial support:	annual subscription
Proxmox VE	
Company:	Proxmox Server Solutions GmbH
Industry:	Server Virtualization
License:	GNU Affero and GPLv3
Website:	pve.proxmox.com
Description:	Virtualization management solution for servers, based on KVM and containers Server Virtualization Platform, provides KVM and OpenVZ hypervisors. Distribution based on Debian.
Supported Platforms:	GNU/Linux
Commercial support:	annual subscription
FreeNAS	
Company:	iXsystems, Inc.
Industry:	Computer Storage
License:	BSD 2-Clause
Website:	freenas.org
Description:	Network-attached storage server, supporting many network and storage protocols such as Samba and NFS. Also supports ZFS. Distribution based on FreeBSD.
Supported Platforms:	BSD Unix
Commercial support:	custom quotes and support tickets
Zabbix	
Company:	Zabbix SIA
Industry:	IT Monitoring
License:	GNU GPLv2
Website:	zabbix.com
Description:	Solution for monitoring of networks, applications and databases.
Supported Platforms:	GNU/Linux
Commercial support:	custom quotes and support tickets

Table 5.1: Software Supporting the Infrastructure

Chapter 6

Technological background

The following software tools represent the key elements on which it has been possible to implement a comprehensive high availability solution, some of them mentioned in Section 3.2.

- Red Hat Enterprise Linux Server

GNU/Linux enterprise-oriented distribution providing a very stable base system, vast documentation and proper support from manufacturer, released as FℓOSS mainly under the terms of the GNU Lesser General Public License 2.1, except for some optional components. In order to be specific in this exercise, the Linux kernel 2.6.32-431.el6.x86_64 that is included by the RHEL version 6.5 was used.

- Zimbra Collaboration System (ZCS)

Server and client collaboration software, supporting e-mail, contacts, calendar, documents, push synchronization, and many other enterprise features related to groupware. The software is FℓOSS released under the terms of the Common Public Attribution License version 1 and the GNU General Public License version 2 (GPLv2). The exact version implemented was ZCS FOSS 8.0.7_GA_6021.RHEL6_64.

- Distributed Replicated Block Device (DRBD)

A distributed replicated storage system for Linux, implemented as several userspace management applications and shell scripts, used to provide data redundancy. Works on top of block devices, such as hard disk partitions or LVM logical volumes, mirroring each data block that it is written to disk to the peer node.

- Corosync

It is released as FℓOSS under the 3-clause BSD License. This software provides features based on C programming language implementing high availability within applications, through virtual synchrony for replicated state machines, simple availability handling responsible for applications restart when fail, it keeps configuration and statistics in a memory database providing the ability to set, retrieve, and receive change notifications of information, and a quorum system that notifies applications when it is achieved or lost.

- Pacemaker

A FℓOSS high availability resource manager software released under GNU GPLv2. This software was part of the Linux-HA project until 2007, then was split out to be its own project. It implements APIs for resources control, including the Open Cluster Framework (OCF). It is used on computer clusters since 2004.

- Cluster Resource Manager Shell (CRMsh[1]) and Pacemaker Configuration System (PCS[2])

Initially, the CRMsh was distributed as part of the Pacemaker project, but it was split into its own separate project in 2011. Also as CRMsh, PCS is a command-line interface to the Pacemaker cluster resource management stack.

- Cluster Configuration System (CCS)

Manages the cluster configuration and provides information to other cluster components. Runs in each cluster node and makes sure that the cluster configuration file in each cluster node is up to date. In Figure 6.1 (took from `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Cluster_Suite_Overview/images/`) is represented a CCS overview.

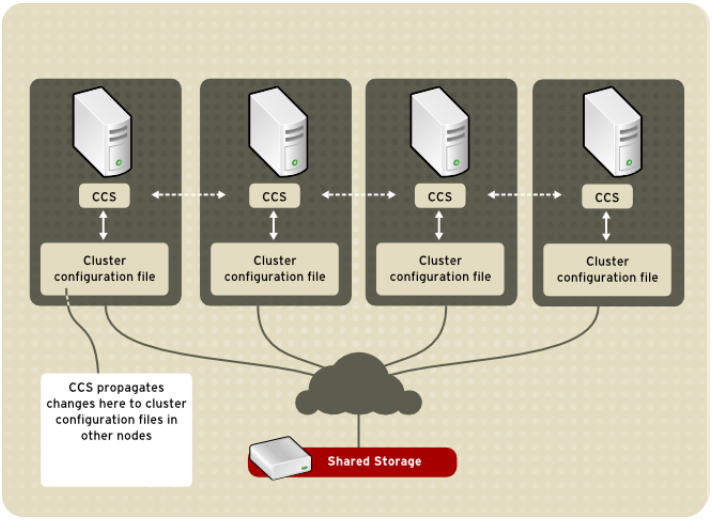

Figure 6.1: CCS overview

- Cluster Manager (CMAN[3])

A set of kernel patches and a userspace program, formed by a Connection Manager (cnxman) and a Service Manager (sm). The first one handles membership, messaging, quorum, event notification and transitions, and the second one is responsible for instances of external systems. It combines some functionalities provided by CRMsh, PCS and CCS.

[1] http://crmsh.github.io
[2] https://github.com/feist/pcs
[3] https://www.sourceware.org/cluster/cman/

Chapter 7

Implementation

This section is intended to provide technical documentation in the process of implementing high availability in a FℓOSS Zimbra Collaboration System (ZCS). The scope of this implementation is framed by the following software components and versions:

- Red Hat Enterprise Linux Server release 6.5 (Santiago)
- GNU/Linux 2.6.32-431.el6.x86_64
- zcs 8.0.7_GA_6021.RHEL6_64 FOSS edition
- drbd 8.4.3-33
- corosync 1.4.5-2.2
- pacemaker 1.1.10-14
- pcs 0.9.90-2
- crmsh 1.2.5-0
- ccs 0.16.2-69
- cman 3.0.12.1-59

The defined cluster consists of two nodes which will be referenced as Astapor and Braavos in the domain got.com (as in the novel Game of Thrones). These nodes are virtual machines hosted on two Proxmox Virtual Environment servers based on KVM virtualization, which are installed on separate physical machines in the same LAN to avoid single point of failure. The proposed scheme is conceptually similar to the observed in Figure 7.1.

7.1 Operating system considerations

The configuration must be similar in both nodes. In Table 7.1 is shown the configuration selected for the current implementation. It should be take into consideration that it is not necessary to format partitions for devices vdb1 or vdc1 during OS install.

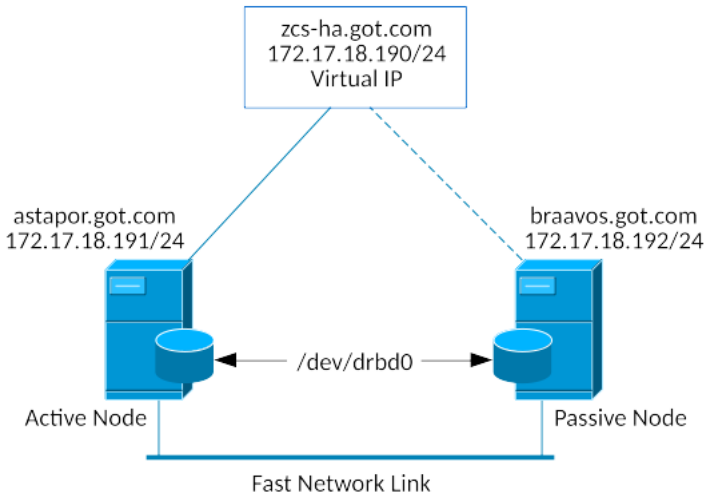

Figure 7.1: Two nodes HA cluster

RHEL 6.5 x86_64			
Disk Partitions:	/	10 Gb	
	/boot	100 Mb	
	/opt/zimbra	8 Gb	(/dev/vdb1)
	drbd meta-data	150 Mb	(/dev/vdc1)
CPU:	1		
RAM:	2 Gb		

Table 7.1: Operating system configuration

7.1.1 FQDN hostnames and IP addresses

Table 7.2 shows the current configuration for the virtual IP address shared by the two nodes, and for the primary IP address on each node.

Split DNS IP:	172.17.18.190	zcs-ha.got.com
Astapor:	172.17.18.191	astapor.got.com
Braavos:	172.17.18.192	braavos.got.com

Table 7.2: FQDN hostnames and IP addresses

On both nodes, /etc/hosts file should contain at least the entries described in Table 7.3:

127.0.0.1	localhost.localdomain	localhost
127.0.0.1	zcs-ha.got.com	zcs-ha
172.17.18.190	astapor.got.com	astapor
172.17.18.191	braavos.got.com	braavos

Table 7.3: /etc/hosts file

A useful command to handle hostname changes in RHEL:

service hostname restart

7.1.2 Network

- Internet Protocol version 4 (IPv4)

Set the proper network parameters in /etc/sysconfig/network-scripts/ifcfg-eth0 file on each server, as described in Table 7.4.

Astapor	Braavos
DEVICE=eth0	DEVICE=eth0
HWADDR=26:34:99:65:d7:77	HWADDR=26:34:99:65:d7:78
TYPE=Ethernet	TYPE=Ethernet
ONBOOT=yes	ONBOOT=yes
NM_CONTROLLED=no	NM_CONTROLLED=no
BOOTPROTO=none	BOOTPROTO=none
IPADDR=172.17.18.191	IPADDR=172.17.18.192
NETMASK=255.255.255.0	NETMASK=255.255.255.0
GATEWAY=172.17.18.1	GATEWAY=172.17.18.1
DNS1=127.0.0.1	DNS1=127.0.0.1
IPV6INIT=no	IPV6INIT=no
USERCTL=no	USERCTL=no

Table 7.4: /etc/sysconfig/network-scripts/ifcfg-eth0 file

Set the correct Netmask and Gateway, so servers are able to reach internet addresses, also disable the firewall or allow the http and ftp outgoing rules on it. The primary DNS server will be configured later to be the localhost, with forwarding to external DNS servers.

Some useful commands to manipulate and consult the network service on RHEL:

service network restart
/etc/init.d/network restart
ifconfig eth0 down; ifconfig eth0 up
ifdown eth0; ifup eth0
ifconfig
ip addr show

- NTP

Required RPM packages to sincronize cluster nodes through network time protocol: ntp, ntpdate.

Set the proper NTP parameters in /etc/ntp.conf file on each server, so both nodes share the same date and time, as shown in Table 7.5.

driftfile	DEVICE=eth0/var/lib/ntp/drift
restrict	default kod nomodify notrap nopeer noquery
restrict	127.0.0.1
server	172.17.18.1
includefile	/etc/ntp/crypto/pw
keys	/etc/ntp/keys

Table 7.5: /etc/ntp.conf file

Some useful commands to manipulate and consult NTP service on RHEL are:

service ntpd restart
ntpstat
ntpq -pn
date

- BIND

Required RPM packages for domain name resolution: bind, bind-utils.

A primary DNS server configured on each server is crucial, or alternatively a remote centralized DNS server on the LAN with the whole configuration. Here is considered the first option. Table 7.6 shows the content of /etc/named.conf file.

```
zone “got.com.” IN {
type master;
file “got.com.db”;
};
```

Table 7.6: /etc/named.conf file

Astapor node holds /var/named/got.com.db file, with the content described in Table 7.7. Dot characters at the end of hostnames are not a typo, they should be included so that the configuration is correct, and must be absent in the case of IP addresses.

	IN	1H	NS	zcs-ha.got.com.
	IN	1H	MX 5	zcs-ha.got.com.
zcs-ha	IN	1H	A	172.17.18.190
astapor	IN	1H	A	172.17.18.191
astapor.got.com	IN		CNAME	zcs-ha.got.com.

Table 7.7: /etc/named.conf file

A similar got.com.db file must be set on braavos node replacing the corresponding hostname and IP address. Leaving zcs-ha entries without changes in both nodes.

Some useful commands to handle and request BIND service on RHEL are:

named-checkconf -z
service named restart
service named status
dig -t ANY got.com
nslookup astapor.got.com

7.1.3 ZCS dependencies

As requirement for ZCS, the following RPM packages must be installed in the OS:

- nc
- sudo
- libidn
- gmp
- libaio

Some other suggested RPM packages are:

- perl-5.10.1
- sysstat
- sqlite

The postfix daemon must be turned off and excluded from boot start-up:
service postfix stop
chkconfig postfix off

7.2 DRBD

The Distributed Replicated Block Device (DRBD) provides a mirrored storage required for the HA environment.

7.2.1 Initial configuration

The following actions must be performed in parallel on both nodes, except in those cases where otherwise specified.

- Ensure to adapt hostname to 'astapor' on the primary node and 'braavos' on the secondary node.
- Install RPM packages:
drbd-kmdl-2.6.32-431.el6-8.4.3-33.el6.x86_64
drbd-8.4.3-33.el6.x86_64
- Leave /etc/drbd.conf and /etc/drbd.d/global_common.conf files by default.
- Add /etc/drbd.d/optzimbra.res file with the content described in Table 7.8:

zcs-ha	IN	1H	A	172.17.18.190
astapor	IN	1H	A	172.17.18.191

Table 7.8: /etc/drbd.d/optzimbra.res file

- Remove from /etc/fstab file any reference to /dev/vdb1 or /dev/vdc1 devices, as drbd is going to handle its mounting.

- Initialize data and metadata disks:
  dd if=/dev/zero of=/dev/vdb1 bs=1K count=100
  dd if=/dev/zero of=/dev/vdc1 bs=1K count=100
- Start DRBD module:
  modprobe drbd
- Create resource:
  drbdadm create-md optzimbra
- Execute first DRBD synchronisation on astapor:
  drbdadm up optzimbra
  drbdadm primary –force optzimbra
  drbdadm –discard-my-data connect optzimbra
- It is possible to check synchronisation status with:
  watch cat /proc/drbd
- Final output will show:
  ds:UpToDate/UpToDate
- Verify current roles:
  drbdadm role optzimbra
  It will show 'Primary/Secondary' on astapor
  and 'Secondary/Primary' on braavos node.
- Now make the filesystem on astapor:
  mkfs.ext4 /dev/drbd0
- Then demote node to secondary, by executing only on astapor:
  drbdadm secondary optzimbra
- Promote node to primary, by executing only on braavos:
  drbdadm primary optzimbra
- Make the filesystem on braavos:
  mkfs.ext4 /dev/drbd0

Now it is necessary to revert the roles back, making braavos the secondary node and astapor the primary one.

7.2.2 DRBD Split Brain Recovery

Assuming that the primary node is still consistent, and the secondary node has an inconsistent state, it would be necessary to recover data loss. The following actions will allow to recover the data corrupted in secondary node.

- In both nodes:
  drbdadm disconnect optzimbra
- In the secondary node:
  drbdadm secondary optzimbra
  drbdadm connect - -discard-my-data optzimbra

- In the primary node:
  drbdadm connect optzimbra

- Finally it is possible to check the sync status, running the command “cat /proc/drbd”, which is going to show a message similar to this:
  cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r- - - -

7.3 ZCS

Here will be fully installed ZCS on astapor but just a dummy installation on braavos, since DRBD will replicate the data to the other node. Download and place ZCS installation file in astapor and braavos filesystems. It can be found at `http://www.zimbra.com/downloads/os-downloads.html`. In order to complete a full install on a single server, the following resource will be useful:
`http://files.zimbra.com/website/docs/8.5/Zimbra_OS_Quick_Start_8.5.0.pdf`

7.3.1 ZCS full install on primary node

The following actions must be performed sequentially on ***astapor***.

- Create directory for ZCS:
  mkdir /opt/zimbra

- Mount DRBD device on ZCS mount point:
  mount /dev/drbd0 /opt/zimbra

- Check mounted device:
  df | grep zimbra
  mount | grep zimbra

- Set manual virtual link configuration temporally:
  ifconfig eth0:1 inet 172.17.18.190 netmask 255.255.255.0

- Set split DNS hostname temporally:
  hostname zcs-ha.got.com
  It is also recommendable to change /etc/sysconfig/network file.

- Unpack ZCS installer and proceed with full installation:
  ./install.sh

- Leave all packages to install by default, and follow the process.

- When prompted for domain name change, select “Yes” and then provide: got.com

- On “Main Menu” section, set admin user password by browsing through option 3 and then 4:
  “Password for admin@zcs-ha.got.com (min 6 characters)”

- Apply configuration and advance until ZCS setup process is completed:
  “Configuration complete - press return to exit”

- Check ZCS status:
  service zimbra status

- Stop ZCS:
  service zimbra stop

- Umount DRBD device:
  umount /opt/zimbra

- Set original DNS hostname:
  hostname astapor.got.com
  Revert change in /etc/sysconfig/network file if needed.

- Delete temporal virtual link configuration:
  ifconfig eth0:1 down

- Demote astapor to secondary DRBD, and continue with section 7.3.2:
  drbdadm secondary optzimbra

7.3.2 ZCS dummy install on secondary node

The following actions must be performed sequentially on braavos.

- Promote braavos to primary DRBD:
  drbdadm primary optzimbra

- Create directory for ZCS:
  mkdir /opt/zimbra

- Mount DRBD device on ZCS mount point:
  mount /dev/drbd0 /opt/zimbra

- Check mounted device:
  df | grep zimbra
  mount | grep zimbra

- Unpack ZCS installer and proceed with a dummy installation:
  ./install.sh -s

- Stop ZCS:
  service zimbra stop

- Umount DRBD device:
  umount /opt/zimbra

- Demote braavos back to secondary DRBD:
  drbdadm secondary optzimbra

- Promote astapor back to primary DRBD, executing from astapor node:
  drbdadm primary optzimbra

  At this point DRBD has to synchronize data from primary node, so check the status until it is done:
  watch cat /proc/drbd

7.4 OCF

Open Cluster Framework, standard scripts to control services such as ZCS. Following actions must be performed in both nodes.

- Add btactic zimbra script to /usr/lib/ocf/resource.d/btactic/zimbra. The source code of this script is included in the Appendix A.

- Also create the following symbolic link:
  ln -s /usr/lib/ocf/resource.d/btactic/zimbra /usr/lib/ocf/resource.d/heartbeat/

In section 7.5 this file will be referenced.

7.5 Pacemaker

Resource manager, starts and stops services orderly.

- Install the required RPM packages:
  pacemaker-cluster-libs-1.1.10-14.el6.x86_64
  pacemaker-libs-1.1.10-14.el6.x86_64
  pacemaker-cli-1.1.10-14.el6.x86_64
  pacemaker-1.1.10-14.el6.x86_64
  cman-3.0.12.1-59.el6.x86_64
  crmsh-1.2.5-0.el6.x86_64
  ccs-0.16.2-69.el6.x86_64
  resource-agents-3.9.2-40.el6_5.7.x86_64

Usually it is difficult to obtain the required RPM's for RHEL, so an alternative is to add CentOS repository by editing /etc/yum.repo.d/centos.repo file with the content described in Table 7.9.

[centos-6-base]		
name	=	CentOS-$releasever - Base
mirrorlist	=	http://mirrorlist.centos.org/?release=6.5&arch=x86_64&repo=os
enabled	=	0
gpgcheck	=	0
baseurl	=	http://mirror.centos.org/centos/6.5/os/x86_64/

Table 7.9: /etc/yum.repo.d/centos.repo file

- Then update and install the packages:
  yum install −−enablerepo=centos-6-base pacemaker pcs.noarch cman \
  ccs resource-agents crmsh

There are two ways to interact with Pacemaker configuration. The first one is using the crmsh interpreter, starting the crm shell with "crm" command, and then providing configuration sentences. For instance:

```
[root@astapor ~]# crm
crm(live)# help
```

crm(live)# quit

Another way would be through ***pcs*** and ***ccs*** instructions directly from a linux tty in a bash session. Following is going to be used this way to configure the cluster, executing the commands only on the primary node.

- Create the cluster:
  ccs −−file /etc/cluster/cluster.conf −−createcluster zcsCluster

- Add the nodes:
  ccs −−file /etc/cluster/cluster.conf −−addnode astapor.got.com
  ccs −−file /etc/cluster/cluster.conf −−addnode astapor.got.com

- Set fencing to defer to Pacemaker:
  ccs −−file /etc/cluster/cluster.conf −−addfencedev pcmk agent=fence_pcmk
  ccs −−file /etc/cluster/cluster.conf −−addmethod pcmk-redirect astapor.got.com
  ccs −−file /etc/cluster/cluster.conf −−addmethod pcmk-redirect braavos.got.com
  ccs −−file /etc/cluster/cluster.conf −−addfenceinst pcmk astapor.got.com \
  pcmk-redirect port=astapor.got.com
  ccs −−file /etc/cluster/cluster.conf −−addfenceinst pcmk braavos.got.com \
  pcmk-redirect port=braavos.got.com

- Disable CMAN quorum:
  This will let the cluster function if only one node is up, and it is necessary to be performed in both nodes.
  echo “CMAN_QUORUM_TIMEOUT=0” >> /etc/sysconfig/cman

- Start Pacemaker Cluster:
  pcs cluster start −−all
  Also equivalent to execute on each node,
  “service pacemaker start” or “pcs cluster start”

- Copy cluster file to secondary node:
  scp -p /etc/cluster/cluster.conf braavos:/etc/cluster/

- Check Pacemaker cluster status:
  pcs status
  crm_mon -1

- Show current cluster config:
  pcs config
  pcs property
  crm configure show

- Check configuration validity:
  crm_verify -L -V

- Disable STONITH (a type of fencing):
  pcs property set stonith-enabled=false

- Ignore Quorum Policy:
  pcs property set no-quorum-policy=ignore

- Set reconnect attempt:
  pcs property set migration-threshold=1 −−force

- Set stickiness:
  pcs property set resource-stickiness=100 −−force

Now, it is going to be used the crmsh interpreter, starting it with the following command:
crm configure

- Add floating IP address resource (Virtual IP - VIP):
  pcs resource create VIP1 IPaddr2 ip=172.17.18.190 broadcast=172.17.18.255 \
  nic=eth0 cidr_netmask=24 iflabel=VIP1 op monitor interval=30s timeout=30s

- Define DRBD cluster resource:
  configure primitive drbd ocf:linbit:drbd params \
  drbd_resource=optzimbra \
  op monitor role=Master interval=60s \
  op monitor role=Slave interval=50s \
  op start role=Master interval=60s timeout=240s \
  op start role=Slave interval=0s timeout=240s \
  op stop role=Master interval=60s timeout=100s \
  op stop role=Slave interval=0s timeout=100s

- Define DRBD Zimbra data clone:
  configure ms drbd_ms drbd \
  meta master-max=1 master-node-max=1 \
  clone-max=2 clone-node-max=1 notify=true

- Define Zimbra service resource:
  configure primitive zcs_service ocf:btactic:zimbra \
  op monitor interval=2min timeout=”40s” \
  op start interval=”0” timeout=”360s” \
  op stop interval=”0” timeout=”360s”

- Define Zimbra cluster filesystem resource:
  configure primitive zcs_fs ocf:heartbeat:Filesystem params \
  device=”/dev/drbd0” directory=”/opt/zimbra” fstype=ext4 \
  op start interval=0 timeout=60s \
  op stop interval=”0” timeout=”60”

- Group all resources in the same host:
  group zcsgroup zcs_fs zcs_service \
  configure colocation VIP1-with-drbd_ms-Master inf: drbd_ms:Master VIP1
  configure colocation drbd_ms-Master-with-zcs_fs inf: zcs_fs drbd_ms:Master
  configure colocation zcs_fs-with-zcs_service inf: zcs_service zcs_fs

- Order resources:
  configure order drbd_ms-promote-on-VIP1 inf: VIP1:start drbd_ms:promote
  configure order zcs_fs-on-dbrb_ms-promote inf: dbrb_ms:promote zcs_fs:start
  configure order zcs_service-on-zcs_fs inf: zcs_fs:start zcs_service:start

- Commit configuration changes and quit:
  commit
  quit

On both nodes make sure chkconfig is off on every service but DRBD. This means the service will not start up on when the server starts up.

chkconfig corosync off
chkconfig cman off
chkconfig ricci off
chkconfig pacemaker off
chkconfig drbd on

7.6 Control and check services

- Check Pacemaker cluster status:
  crm_mon -1
  pcs status

- Check resources status:
  crm resource status RESOURCE

- Check configuration validity:
  crm_verify -L -V

- Edit values already configured:
  crm configure edit
  After save changes through the preferred text editor, exit and execute:
  cibadmin −−replace

- Delete existent resource:
  pcs resource delete RESOURCE

- Clean resource history errors (check configuration health):
  crm_resource -P

- List available classes and resources:
  crm ra classes
  crm ra list ocf btactic
  crm ra list lsb

- Delete cluster configuration (WARNING):
  pcs cluster destroy

7.7 Testing failover

- On primary node:
  crm node standby
  Or stop pacemaker:
  service pacemaker stop

- Now “crm_mon” or “pcs status” will show:
  Node astapor.got.com: standby
  Online: [braavos.got.com]
- It is going to take a while before secondary node takes control. So it is possible to check logs and “crm_mon” status during the process.
  crm_mon
  tail -F /var/log/zimbra.log
  tail -F /var/log/messages
- Also it is possible to check with “crm_standby” command. A value of true|on indicates that the node is not able to host any resources and a value of false|off indicates it does.
  crm_standby −−get-value
- At any moment it will be displayed a message like the depicted in Table 7.10.

Master/Slave Set:	drbd_ms [drbd]
Masters:	[braavos.got.com]
Slaves:	[astapor.got.com]
Resource Group:	zcsgroup
zcs_fs (ocf::heartbeat:Filesystem):	Started braavos.got.com
zcs_service (ocf::btactic:zimbra):	Started braavos.got.com
VIP1 (ocf::heartbeat:IPaddr2):	Started braavos.got.com

Table 7.10: DRBD failover test

- Now the secondary node has control of the cluster resources, while the primary node is in standby or unreachable state. If primary node is back online, secondary node will keep the control of resources, until an explicit node move is done.
- Set back online the primary node:
  crm node online
  Or start over pacemaker service:
  service pacemaker start
- To give the control back to primary node, execute on secondary node:
  crm node standby
  Then resources will be transferred back to primary node.
- Finally “crm_mon” or “pcs status” on each node will display a similar to the one showed in Table 7.11

Online:	[astapor.got.com braavos.got.com]
Master/Slave Set:	drbd_ms [drbd]
Masters:	[astapor.got.com]
Slaves:	[braavos.got.com]
Resource Group:	zcsgroup
zcs_fs (ocf::heartbeat:Filesystem):	Started astapor.got.com
zcs_service (ocf::btactic:zimbra):	Started astapor.got.com
VIP1 (ocf::heartbeat:IPaddr2):	Started astapor.got.com

Table 7.11: DRBD synced status

Chapter 8

Results and discussion

In the initial research over internet, main FℓOSS source code repositories were consulted (listed bellow alphabetically) in order to review the state of the art regarding tools that could fit the requirements. Most of the related projects were currently hosted in Github. It is worth noting important work in this area such as the "Study of available tools" [5] by the FLOSSMetrics Consortium.

- BerliOS (berlios.de)
- BountySource (bountysource.com)
- FLOSSMetrics (flossmetrics.org)
- FLOSSmole (ossmole.sourceforge.net)
- GitHub (github.com)
- Gitorious (gitorious.org)
- GNU Savannah (savannah.gnu.org)
- Launchpad (launchpad.net)
- SourceForge (sourceforge.net)

Since the desired solution was corporate oriented and focused on a GNU/Linux distribution, official websites and documentation were browsed for Hewlett Packard, Redhat Enterprise Linux and SuSE Linux Enterprise Server.

- "HP Serviceguard sglx for Linux Deployment Guide". Hewlett Packard Co. DOI=http://www.hp.com/go/sglx/info
- "Red Hat Enterprise Linux 6 Cluster Administration". Red Hat Inc. et al, 2014. DOI=https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/ \ Cluster_Administration
- Roth T. and Schraitle T. "SUSE Linux Enterprise High Availability Extension". Novell Inc, 2014. https://www.suse.com/documentation https://en.opensuse.org/openSUSE:High_Availability

Also documentation from each individual FℓOSS product:

- DRBD: Haas F., Reisner P., Ellenberg L. et al. “The DRBD User’s Guide”. LINBIT Information Technologies GmbH and LINBIT HA Solutions GmbH. 2011.
  DOU=https://drbd.linbit.com/users-guide

- Linux-HA: Haas, Florian. “The Linux-HA User’s Guide”. LINBIT HA-Solutions GmbH, The Linux-HA Project. 2010.
  DOI=http://www.linux-ha.org/wiki/MainPage

- Pacemaker: “A scalable High Availability cluster resource manager”. ClusterLabs.
  DOI=http://clusterlabs.org/wiki/MainPage

- OCF: Haas, Florian. “The OCF Resource Agent Developer’s Guide”. LINBIT HA-Solutions GmbH, Novell, Inc., SUSE Linux GmbH, hastexo Professional Services GmbH. 2011.
  DOI=http://www.linux-ha.org/doc/dev-guides/ra-dev-guide.html

Also specific related work was found, such as:

- Gibanel Lopez, Adrian. “Zimbra 8 High Availability on Ubuntu 12.04”. Universitat de Lleida. 2013. DOI=http://repositori.udl.cat/bitstream/handle/10459.1/46685/agibanell.pdf

- OCF zimbra script: https://github.com/adrian15/hazimbra-thesis/blob/master/ocf/zimbra

- Vidal Lopez M. and Castro Jose L. “Creacion de un cluster de alta disponibilidad con software libre”. Novatica Journal. Nro 210. 2011.
  DOI=http://www.ati.es/novatica/2011/209/Nv209-75.pdf

Once enough information on the subject of interest have been collected, is crucial to abstract the essence, for which a model as LUM [9] is ideal. This model focuses on user needs and the demanded effort when selecting a solution to a problem from a set of possible solutions, identifying significant results from patterns and focus on the primary research collected.

A particular type of data that has been obtained is the source code repository for some of the main projects involved in the final solution. They refers to collection of source code used to build a particular software system, supporting versioning through revision control systems, on multi-developer projects to handle various code versions and providing aid in resolving conflicts that arise from developers submitting conflicting modifications.

- Corosync: https://github.com/corosync/corosync

- DRBD: git://git.drbd.org/drbd-8.4.git

- Pacemaker: https://github.com/ClusterLabs/pacemaker

These repositories aside from offering the source code of the software, basically provide data about the project’s development behavior and details of the community that makes it possible. This allows for instance, to find out how long has been developing the software and to know if the project is still active, among many other interesting analysis. To achieve this is very useful to use tools like Metrics Grimoire [8], obtaining data from project’s source code repositories and retrieving information about commits, ticket management, communication in mailing lists, etc. The data is organized and stored into SQL databases that can later be mined for specific patterns or summaries of activity.

In particular Repository Handler and CVSanalY are the tools that have been used to extract the required data. A detailed description for setup and use of these tools can be found in the document “Analysing

Libre Software communities" [4] published by GSyC LibreSoft from Universidad Rey Juan Carlos and Bitergia[1] team. Part of the analysis consisted into make the following SQL requests:

- Number of commits in the last 6 months, shown in Figure 8.1 for Corosync, Figure 8.2 for DRBD and Figure 8.3 for Pacemaker:

```
MariaDB [cvsanaly_corosync]> SELECT COUNT(s.id) FROM scmlog s, people p WHERE s.committer_id=p.i
d AND YEAR(s.date)=2014 AND MONTH(s.date) IN (7,8,9,10,11,12);
+-------------+
| COUNT(s.id) |
+-------------+
|          82 |
+-------------+
```

Figure 8.1: Corosync - number of commits in the last 6 months

```
MariaDB [cvsanaly_drbd]> SELECT COUNT(s.id) FROM scmlog s, people p WHERE s.committer_id=p.id AN
D YEAR(s.date)=2014 AND MONTH(s.date) IN (7,8,9,10,11,12);
+-------------+
| COUNT(s.id) |
+-------------+
|          46 |
+-------------+
```

Figure 8.2: DRBD - number of commits in the last 6 months

```
MariaDB [cvsanaly_pacemaker]> SELECT COUNT(s.id) FROM scmlog s, people p WHERE s.committer_id=p.
id AND YEAR(s.date)=2014 AND MONTH(s.date) IN (7,8,9,10,11,12);
+-------------+
| COUNT(s.id) |
+-------------+
|         367 |
+-------------+
```

Figure 8.3: Pacemaker - number of commits in the last 6 months

The values returned on these queries indicate that indeed each of the projects keep some activity to the date that they were executed (December 2014), in particular Pacemaker with 367 commits, followed by Corosync with 82 commits and finally DRBD with 46 commits during the last six months.

A low amount of commits does not necessarily imply a lack of developer's participation, it can also be related to the maturity of the project and its robust structure, although also to a lack of bug posting from the users community. It could be observed in the following queries that these projects particularly have a small but consistent group of committers.

- Evolution in the number of commits per year since the begining of the project, shown in Figure 8.4 for Corosync, Figure 8.5 for DRBD and Figure 8.6 for Pacemaker:

  For this purpose was employed a GNU GPL python script INCLUDE INTO APPENDIX published by Daniel Izquierdo from GSyC/LibreSoft which uses the following query:
  SELECT YEAR(date), MONTH(date), DAY(date), COUNT(*) FROM scmlog GROUP BY YEAR(date), MONTH(date), DAY(date);

[1] http://bitergia.com

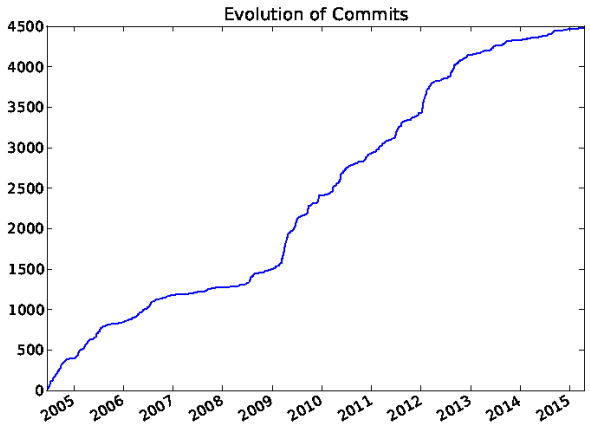

Figure 8.4: Corosync - evolution in the number of commits per year

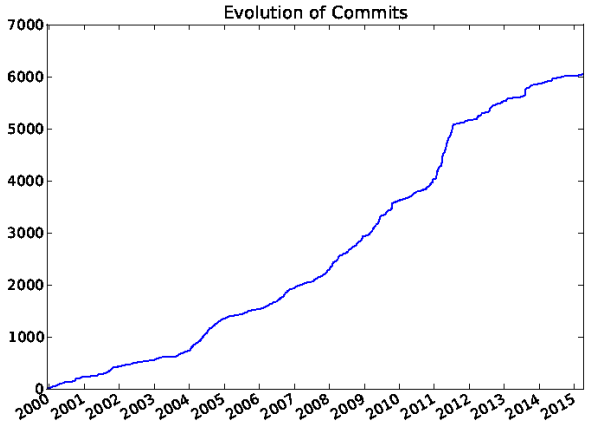

Figure 8.5: DRBD - evolution in the number of commits per year

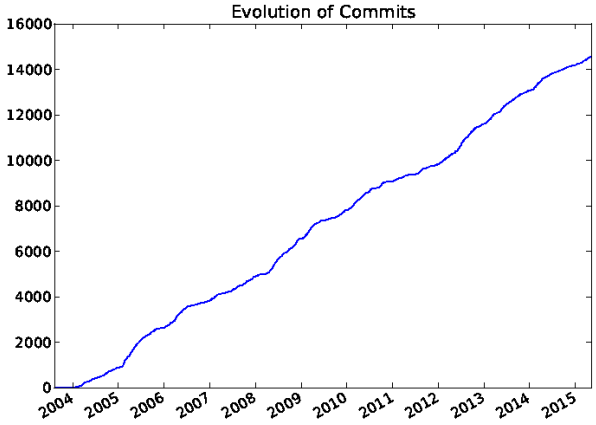

Figure 8.6: Pacemaker - evolution in the number of commits per year

The incremental progress is evident in the number of commits from the start of each project, being Pacemaker the one with a larger number of commits in a relatively similar period of time. DRBD meanwhile began about four years earlier and presents a consistent growth but below the 1000 commits. It was not until the emergence of projects such as Pacemaker and Corosync that its development is driven,

due to the benefits that these tools are capable to offer working together to provide a common goal - high availability.

- Evolution in the number of active committers since the begining of the project, shown in Figure 8.7 for Corosync, Figure 8.8 for DRBD and Figure 8.9 for Pacemaker:

  To produce these bar charts was used a simple python script INCLUDE INTO APPENDIX with the following queries:
  SELECT year(s.date) FROM scmlog s GROUP BY year(s.date);
  SELECT COUNT(DISTINCT(p.id)) FROM scmlog s, people p WHERE s.committer_id=p.id AND YEAR(s.date)=XXXX;

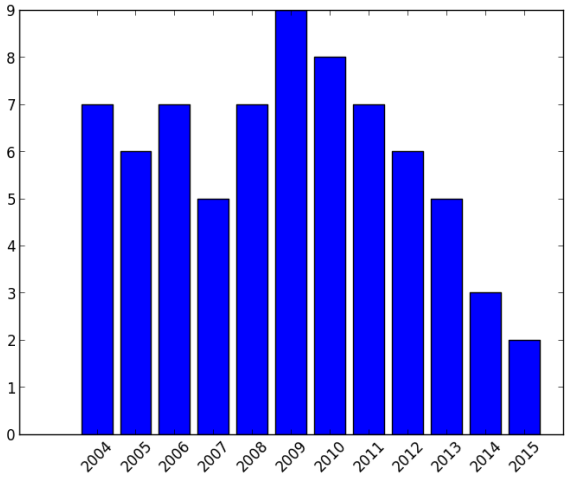

Figure 8.7: Corosync - evolution in the number of active committers per year

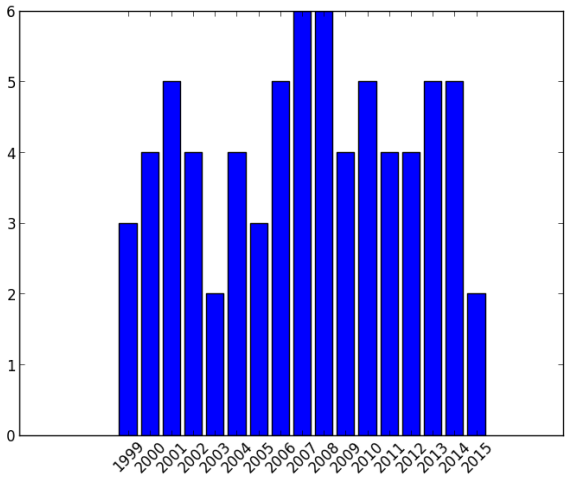

Figure 8.8: DRBD - evolution in the number of active committers per year

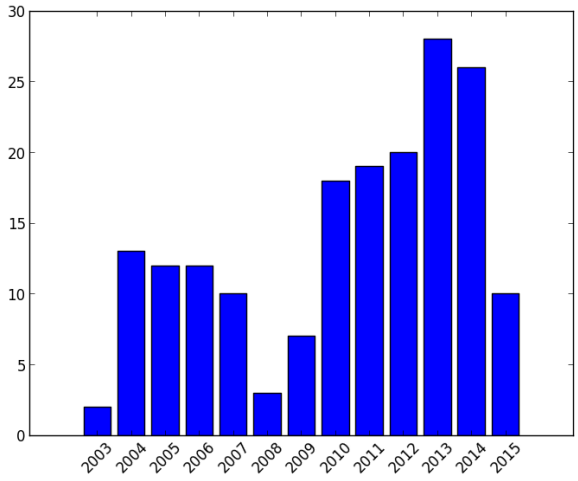

Figure 8.9: Pacemaker - evolution in the number of active committers per year

As histograms show, annually the number of active developers related to Corosync (5 on average) and DRBD (4 on average) is low in comparison with Pacemaker (13 on average). It can be said that DRBD and Pacemaker have tended to keep their average number of active committers, while Corosync has decreased it with the passing of the years.

Increases in the number of committers as of 2009 for Corosync, can be related to the fact that the project was formally announced as independent in 2008 [1], the source code of OpenAIS was refactored and its core infrastructure components were adopted into Corosync, possibly turning it more attractive to developers.

Peaks as the one around 2007 for DRBD is related to the discussions to include the project into the Linux kernel mainline [7], it is likely that this event captured the attention of developers in the community at the moment, and in fact this actually happened. The last value of 2 active developers in 2005 occurs because it still has not been calculated the activity for the entire year.

In 2008 there is a considerable decrease in the number of active committers for Pacemaker, caused by the splitting of the Linux-HA project [2], but then the community have reacted positively and got involved in the project in a notorious way.

Finally it can be said that the analysis suggested by Carlo Daffara [2] through his guidelines for the adoption of FℓOSS within SMEs has proved tremendously useful in the process of selecting and evaluate new tools among the vast amount of existent alternatives in the market.

The proposed hypothesis of a search method by collecting and read as much information related to the project is available, results on the one hand, a task very well supported with internet approachability and lots of information available for both official and unofficial. Though on the other hand, this amount of information is not always available in an organized or easy way to digest, leading to an extensive analysis on many occasions that the companies are not willing to invest.

To deal with the latter fact, the Lazy User Model leverages the process by which are elected the technological tools that represent the appropriate solution and fulfill the user requirements.

[2] http://clusterlabs.org/wiki/Pacemaker#Project_History

Chapter 9

Conclusions and future work

Text..

As future work: - OpenBRR [11], QSoS, QualOSS,

Bibliography

[1] Dake S. Caulfield C. and Beekhof A. The corosync cluster engine. Proceedings of the Linux Symposium. Corosync Cluster Engine Team, 2008.

[2] Daffara Carlo. The sme guide to open source software. FLOSSMETRICS EU, 2009.

[3] Daffara Carlo. The economic value of open source software. Conecta Research. Italy, 2012.

[4] Izquierdo D. and Romera T. Practical approach: analysing libre software communities. Master on Free Software at URJC, 2009.

[5] Robles G. Izquierdo D. et al. Study of available tools. FLOSSMetrics Consortium, 2008.

[6] International Organization for Standardization Technical Committee 223. Iso/pas 22399:2007. International Organization for Standardization, 2007.

[7] Ellenberg Lars. Drbd wants to go mainline. linux-kernel Mailing list, 2007.

[8] GSyC LibreSoft. Metricsgrimoire. Universidad Rey Juan Carlos - Bitergia, 2012. DOI=http://metricsgrimoire.github.io.

[9] Collan M. and Tetard F. Lazy user theory of solution selection. Proceedings of the CELDA 2007 Conference. Portugal, 2007.

[10] Collan M. and Tetard F. Lazy user theory: A dynamic model to understand user selection of products and services. 42nd Hawaii International Conference on System Sciences, 2009.

[11] Wasserman A. Pal M. and Chan C. The business readiness rating model: an evaluation framework for open source. Proceedings of the EFOSS: OpenBRR Workshop. Italy, 2006.

[12] Mell Peter and Grance Timothy. The nist definition of cloud computing. National Institute of Standards and Technology, 2011.

[13] Weygant Peter. Clusters for high availability: A primer of hp solutions. Prentice Hall, 2001.

[14] Tanja R. and Schraitle T. High availability guide. SUSE LLC and contributors, 2014.

Appendix A

btactic zimbra script

```
#!/bin/sh
#
# Resource script for Zimbra
#
# Description: Manages Zimbra as an OCF resource in an high-availability setup.
#
# Author: Adrian Gibanel
# <adrian.gibanel@btactic.com>: Original Author
#
# License: GNU General Public License (GPL)
# Note: Aimed at an active/passive cluster originally
# Inspired from postfix OCF script
# Inspired from Ubuntu LSB script.
# Not sure it will work for other distros without modifying
#
# usage: $0 {start|stop|reload|status|monitor|validate-all|meta-data}
#
# The “start” arg starts Zimbra
# The “stop” arg stops it.
#
# OCF parameters:
# OCF_RESKEY_binary
# OCF_RESKEY_config_dir
# OCF_RESKEY_parameters
#
######################

# Initialization:

: ${OCF_FUNCTIONS_DIR=${OCF_ROOT}/lib/heartbeat}
. ${OCF_FUNCTIONS_DIR}/ocf-shellfuncs
: ${OCF_RESKEY_binary=“zmcontrol”}
: ${OCF_RESKEY_zimbra_dir=“/opt/zimbra”}
: ${OCF_RESKEY_zimbra_user=“zimbra”}
: ${OCF_RESKEY_zimbra_group=“zimbra”}
USAGE=“Usage: $0 {start|stop|reload|status|monitor|validate-all|meta-data}”;
```

```
################################

usage() {
    echo $USAGE >&2
}

meta_data() {

    cat <<END

    <?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="zimbra">
<version>0.1</version>
<longdesc lang="en">
This script manages Zimbra as an OCF resource in a high-availability setup.
</longdesc>
<shortdesc lang="en">
Manages a highly available Zimbra mail server instance
</shortdesc>

<parameters>

<parameter name="binary" unique="0" required="0">
<longdesc lang="en">
Short name to the Zimbra control script.
For example, "zmcontrol".
</longdesc>
<shortdesc lang="en">
Short name to the Zimbra control script</shortdesc>
<content type="string" default="zmcontrol" />
</parameter>

<parameter name="zimbra_dir" unique="1" required="0">
<longdesc lang="en">
Full path to Zimbra directory.
For example, "/opt/zimbra".
</longdesc>
<shortdesc lang="en">
Full path to Zimbra directory</shortdesc>
<content type="string" default="/opt/zimbra" />
</parameter>

<parameter name="zimbra_user" unique="1" required="0">
<longdesc lang="en">
Zimbra username.
For example, "zimbra".
</longdesc>
```

<shortdesc lang="en">Zimbra username</shortdesc>
<content type="string" default="zimbra" />
</parameter>

<parameter name="zimbra_group" unique="1" required="0">
<longdesc lang="en">
Zimbra group.
For example, "zimbra".
</longdesc>
<shortdesc lang="en">Zimbra group</shortdesc>
<content type="string" default="zimbra" />
</parameter>

</parameters>

<actions>
<action name="start" timeout="360s" />
<action name="stop" timeout="360s" />
<action name="restart" timeout="360s" />
<action name="monitor" depth="0" timeout="40s" interval="60s" />
<action name="validate-all" timeout="360s" />
<action name="meta-data" timeout="5s" />
</actions>
</resource-agent>

END
}

command()
{
    if [-f ${zimbra_dir}/redolog/redo.log]; then
        chown -f ${zimbra_user}:${zimbra_group} ${zimbra_dir}/redolog/redo.log
    fi
    su - ${zimbra_user} -c "${binary} $1 </dev/null"
}

running() {
    # run Zimbra status
    command status
}

zimbra_status()
{
    running
}

zimbra_start()
{
    # if Zimbra is running return success

```
    if zimbra_status; then
        ocf_log info “Zimbra is already running.”
        return $OCF_SUCCESS
    fi

    # start Zimbra
    command start
    ret=$?
    if [ -d /var/lock/subsys -a $ret -eq 0 ]; then
        touch /var/lock/subsys/zimbra
    fi

    if [ $ret -ne 0 ]; then
        ocf_log err “Zimbra returned an error.” $ret
        return $OCF_ERR_GENERIC
    fi

    # grant some time for startup/forking the sub processes
    sleep 2

    # initial monitoring action
    running
    ret=$?
    if [ $ret -ne $OCF_SUCCESS ]; then
        ocf_log err “Zimbra failed initial monitor action.” $ret
        return $OCF_ERR_GENERIC
    fi

    ocf_log info “Zimbra started.”
    return $OCF_SUCCESS
}

zimbra_stop()
{
    # if Zimbra is not running return success
    if ! zimbra_status; then
        ocf_log info “Zimbra already stopped.”
        return $OCF_SUCCESS
    fi

    # stop Zimbra
    command stop
    ret=$?

    if [ -d /var/lock/subsys -a $ret -eq 0 ]; then
        rm -f /var/lock/subsys/zimbra
    fi

    if [ $ret -ne 0 ]; then
```

```
        ocf_log err “Zimbra returned an error while stopping.” $ret
        return $OCF_ERR_GENERIC
    fi

    # grant some time for shutdown and recheck 5 times
    for i in 1 2 3 4 5; do
        if zimbra_status; then
            sleep 1
        fi
    done

    # escalate to abort if we did not stop by now
    if zimbra_status; then
        ocf_log err “Zimbra failed to stop. Escalating to ‘abort’.”

        ORPHANED=`ps -u ${zimbra_user} -o “pid=”‘ && kill -9 $ORPHANED 2>&1
        ret=$?
        sleep 10

        # zimbra abort did not succeed
        if zimbra_status; then
            ocf_log err “Zimbra failed to abort.”
            return $OCF_ERR_GENERIC
        fi
    fi

    ocf_log info “Zimbra stopped.”
    return $OCF_SUCCESS
}

zimbra_restart()
{
    if zimbra_status; then
        ocf_log info “Reloading Zimbra.”
        command restart
    fi
}

zimbra_monitor()
{
    if zimbra_status; then
        return $OCF_SUCCESS
    fi
    return $OCF_NOT_RUNNING
}

zimbra_validate_all()
{
    # check zimbra_dir parameter
```

```
    if [ ! -d "$zimbra_dir" ]; then
        ocf_log err "Zimbra directory '$config_dir' does not exist." $ret
        return $OCF_ERR_INSTALLED
    fi
    # check that the Zimbra binaries exist and can be executed
    if ! have_binary "${zimbra_dir}/bin/${binary}" ; then
        return $OCF_ERR_INSTALLED
    fi

    # check permissions
    user=${zimbra_user}
    zimbra_writable_dirs="${zimbra_dir}/conf"
    for dir in "$zimbra_writable_dirs"; do
        if ! su -s /bin/sh - $user -c "test -w $dir"; then
            ocf_log err "Directory '$dir' is not writable by user '$user'."
            exit $OCF_ERR_PERM;
        fi
    done

    return $OCF_SUCCESS
}

#
# Main
#

if [ $# -ne 1 ]; then
    usage
    exit $OCF_ERR_ARGS
fi

binary=$OCF_RESKEY_binary
zimbra_dir=$OCF_RESKEY_zimbra_dir
zimbra_user=$OCF_RESKEY_zimbra_user
zimbra_group=$OCF_RESKEY_zimbra_group
parameters=$OCF_RESKEY_parameters

# build Zimbra options string *outside* to access from each method
OPTIONS=''
OPTION_CONFIG_DIR=''

# check if the Zimbra config_dir exist
if [ "x$config_dir" != "x" ]; then
    # check for postconf binary
    #check_binary "${zimbra_dir}/bin/${binary}"

    # remove all trailing slashes
    zimbra_dir=`echo $zimbra_dir | sed 's/*$//'`
fi
```

```
case $1 in
    meta-data) meta_data
               exit $OCF_SUCCESS
               ;;

    usage|help) usage
               exit $OCF_SUCCESS
               ;;
esac

zimbra_validate_all
ret=$?

LSB_STATUS_STOPPED=3
if [ $ret -ne $OCF_SUCCESS ]; then
    case $1 in
        stop) exit $OCF_SUCCESS ;;
        monitor) exit $OCF_NOT_RUNNING;;
        status) exit $LSB_STATUS_STOPPED;;
        *) exit $ret;;
    esac
fi

case $1 in
    monitor) zimbra_monitor
        exit $?
        ;;
    start) zimbra_start
        exit $?
        ;;

    stop) zimbra_stop
        exit $?
        ;;

    restart) zimbra_restart
        exit $?
        ;;

    status) if zimbra_status; then
               ocf_log info “Zimbra is running.”
               exit $OCF_SUCCESS
            else
               ocf_log info “Zimbra is stopped.”
               exit $OCF_NOT_RUNNING
            fi
            ;;
```

```
    validate-all) exit $OCF_SUCCESS;;

    *) usage
        exit $OCF_ERR_UNIMPLEMENTED;;
esac
```