Tunisian republic
Ministry of Higher Education and Scientific Research
University of Sfax
National Engineering School of Sfax

Technician Internship Report

High availability Architecture for E-commerce

College year : 2019-2020

Produced by :
Sanei Nessrine
Yousfi Mohamed Ali

Internship supervisor :
Rostom Mhiri

Pedagogical supervisor :
Zalila Bechir

Gratitude

We express our gratitude to everyone who has contributed in one way or another to the realization of this work and the development of this document. We are very grateful to all the people who directly or indirectly support us and encourage us to go forward.

We would like to thank especially the Internet who contributed to the success of our project which accompanied and helped us during building the architecture, configuring, and the report writing.

Abstract

This document is a synthesis of the technician internship project carried out within Digital Click enterprise. The objective to be achieved is to make the web sites offered by any company more available.

In fact, due to the massive requests on a server, the latter may experiences problems that incapable it from responding to the client requests. This problem happens because each server has a maximum limit for serving clients.

To solve this problem, while this project, we built a high-availabile architecture based on HAproxy , Mysql servers and the Prestashop Content Management System.

Acronyms List

HTTP: Hyper Text Transfer Protocol

HTTPS: Hyper Text Transfer Protocol Secure

DNS: Domain Name System

FTP: File Transfer Protocol

CMS: Content Management System

WAN: Wide area network

SSL: Transport Layer Security

HA: High availability

Haproxy: High availability proxy

NFS: Network File System

GlusterFS: Scalable Network File System

Table of Contents

Table of Figures

Introduction

Communicating between computers and sharing data between them is usually done over a network. Whether it's a local network or an external network (generally the internet).

Because every business is highly dependent on the Internet, every minute counts and achieving business continuity is a primary concern for modern organizations. Downtime can cause significant financial impact and, in some cases, irrecoverable data loss. That is why the companies computers and servers must stay operational at all times.

> “But the more technology develops, the more it is sophisticated to manage it.”

As the traffic within a company's network or website increases, the strain on data center servers grows. Each request to access applications or information from a server adds to the overall processing capacity that it is able to handle. This increase in user access continues to add up until, ultimately, the server cannot handle any more traffic, and makes the effectiveness of a website server decrease. This is because one server bears the burden (weight) of the number of visitors who come so that it makes the server down or decreases its performance. From these problems it can also lead to a lack of practical data management. [1]

Apart from soaring traffic problems, another problem that arises is the availability of a strong and reliable technology infrastructure in managing big data every day. “Managing the big data is done by the server”. The solution to this problem is to add more servers and share the load between them, and besides that, the use of replication on each database server will make the server load lighter. When HTTP traffic increases on an organization's website, load balancing helps distribute the flow to the appropriate web servers.[2] So with the above solutions, each server has its own load according to the capacity of each server, and also data management can be done on two servers without having to enter the same data one by one to all servers. With this method, HTTP requests received by the server will be handled by the server in the best condition . So the load of a server will be divided according to the capacity of each server.

In implementing load balancing and replication, we use an open source called HAproxy for server load sharing and an open source SQL called MySQL for storing a database. And also done a replication, namely to backup data / share the same data between servers.

So how can we implement a reliable high availability architecture that will meet the needs of companies?

To answer the prospecting problem and to achieve our objectives, our work will be structured as follows: in the first chapter, we present the host company, the second chapter we will define the basic theories and present the state of the art, in the other chapter, we will constitute our technical study and the last chapter will deal with our achievements and tests. This report will close with a conclusion.

Chapter 1:Company’s Presentation

Introduction

In this chapter, we will present the host company and describe the areas in which it produces and intervenes. Besides, we will describe its human employment structure.

I.Description

Digital Click is a system and computer network integrator. This company is located in Sfax-Tunisia. It is a flexible, reactive structure that offers tailor-made solutions in the field of new technologies and information systems.

It intervenes in all areas related to IT infrastructures:

* servers: consolidation, clusters, server virtualization (VMWare, Citrix etc),

* applications: databases and corporate messaging,

* access: company directories and access strategies for information systems,

* networks and security: Lan/Wan/Man, VPN, ToIP, wireless network, public key infrastructure, network security strategy

* clients: thin clients, workstations/tablet/smartphone, operating systems

* Computer data storage: SAN, NAS, backup, remote backup,

* Supervision and management of information systems. [2]

II.Members of the team

Digital Click bases its development on its human capital, in fact the women and men of Digital Click constitute the main wealth of the company. They consider excellence in customer service as the key of success . Digital Click has thus implemented a state-of-the-art talent management policy and resources while respecting the values. Employees there are 80% engineers, 100% young and 50% of them are women. [2]

Conclusion

In this chapter, we have presented the company that has welcomed us throughout the period of our internship. We have also detailed the framework and subject of our project.

The following chapter will be devoted to a detailed presentation of the developed project.

Chapter 2: State of The Art

Introduction

To understand the subject well , we need to understand the basic theories such as load balancing,web server, Haproxy, database server,cache... Than, we will present an existent solution for the high-availability web applications on the Cloud provided by RightScale campany.

I.Basic Theory

1.Load-balancing

Load balancing is defined as the methodical and efficient distribution of network or application traffic across multiple servers in a server farm. Each load balancer sits between client devices and backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them.[4] It improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions.

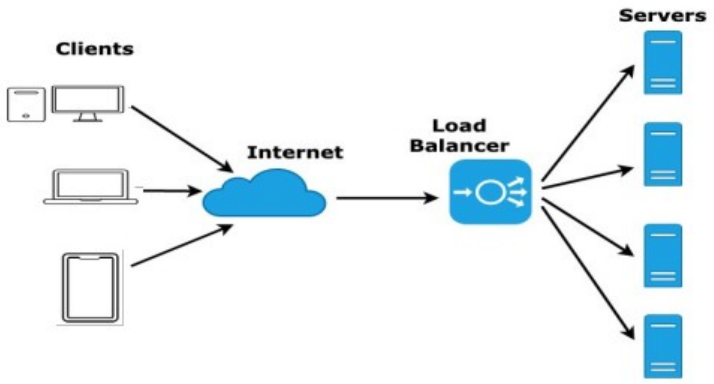


Figure 1: Load-balancer

Load balancers are generally grouped into two categories: Layer 4 and Layer 7. Layer 4 load balancers act upon data found in network and transport layer protocols (IP, TCP, FTP, UDP). Layer 7 load balancers distribute requests based upon data found in application layer protocols such as HTTP. [5]

Requests are received by both types of load balancers and they are distributed to a particular server based on a configured algorithm. Some industry standard algorithms are: [5]

- Round Robin

- Weighted Round Robin
- Least Connections
- Least Response Time
- Least Bandwidth
- …

2.Web Server

A web server is software and hardware that uses HTTP and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing and delivering web pages to users. Besides HTTP, web servers also support SMTP and FTP, used for email, file transfer and storage.

Web server hardware is connected to the internet and allows data to be exchanged with other connected devices, while web server software controls how a user accesses hosted files. The web server process is an example of the client/server model. All computers that host websites must have web server software.[6]

3.HAProxy

HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers quite a number of the world's most visited ones. Over the years it has become the de-facto standard opensource load balancer, is now shipped with most mainstream Linux distributions, and is often deployed by default in cloud platforms.[7]

Its most common use is to improve the performance and reliability of a server environment by distributing the workload across multiple servers (e.g. web, application, database). It is used in many high-profile environments, including: GitHub, Imgur, Instagram, and Twitter.

There are two types of proxy: forward proxy and reverse proxy.

A forward proxy, often called proxy server, or web proxy, is a server that sits in front of a group of client machines. When those computers make requests to sites and services on the Internet, the

proxy server intercepts those requests and then communicates with web servers on behalf of those clients, like a middleman.[8]

A reverse proxy is a server that sits in front of web servers and forwards client (e.g. web browser) requests to those web servers. Reverse proxies are typically implemented to help increase security, performance, and reliability. [8]

The following figure describes the position of the proxy server in both forward and reverse-proxy.

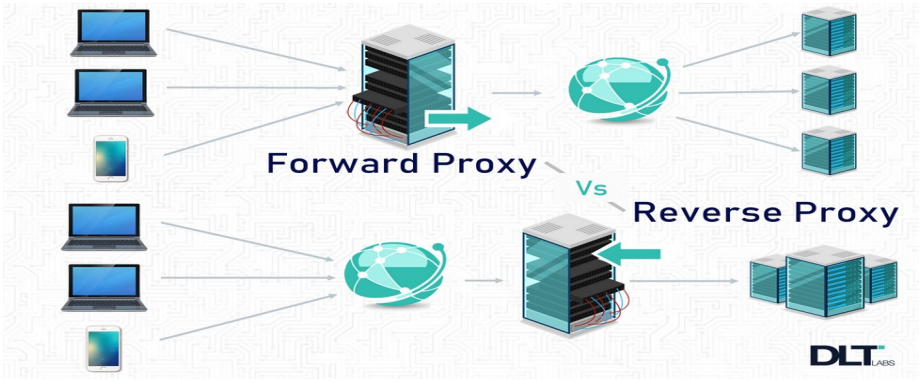


Figure 2: Forward proxy vs reverse-proxy

4.Database Server

Database Server is a service provider application for manage databases using the client / server model. The database management system is usually available various database server functions. To be able to access database, some other very database management system relies on a client-server architectural model, for example such as MySQL and Microsoft SQL Server.

5.Mysql Server

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). It is most noted for its quick processing, proven reliability, ease and flexibility of use. MySQL is an essential part of almost every open source PHP application. Good examples for PHP & MySQL-based scripts are WordPress, Joomla!, and Drupal. [9]

6.Cache

A cache is a reserved storage location that collects temporary data to help websites, browsers, and apps load faster. Whether it's a computer, laptop or phone, web browser or app, you'll find some variety of a cache. A cache makes it easy to quickly retrieve data, which in turn helps devices run faster. It acts like a memory bank, making it easy to access data locally instead of re-downloading it every time you visit a website or open an app. [10]

7.Content manager system

A content management system (CMS) is a computer software used to manage the creation and modification of digital content. CMSs are typically used for enterprise content management (ECM) and web content management (WCM). ECM typically supports multiple users in a collaborative environment by integrating document management, digital asset management and record retention. Alternatively, WCM is the collaborative authoring for websites and may include text and embed graphics, photos, video, audio, maps and program code that display content and interact with the user. ECM typically includes a WCM function.[11]

There are various CMSs such as WordPress, Shopify, Joomla and Prestashop. In our architecture we will be using the latter CMS.

II.High availability architecture by RightScale

Anyone with a good idea can develop a small application, purchase a domain name, and set up a few PC-based servers to handle incoming traffic. The initial investment is small, so the start-up risk is minimal. But a successful low-cost infrastructure can become a serious problem quickly. A single server that handles all the incoming requests may not have the capacity to handle high traffic volumes once the business becomes popular. In such a situations companies often start to *scale up*: they upgrade the existing infrastructure by buying a larger box with more processors or add more memory to run the applications.

In a widely used server load balancing architecture, the incoming request is directed to a dedicated server load balancer that is transparent to the client. Based on parameters such as availability or current server load, the load balancer decides which server should handle the request and forwards it to the selected server. To provide the load balancing algorithm with the required input data, the

load balancer also retrieves information about the servers' health and load to verify that they can respond to traffic. [12]

The most basic architecture that will provide a high-availability is the following 3-tier setup. It is a load-balanced 6-server setup with fault tolerance and database backups that are stored as **EBS** "Elastic Block Storage (Amazon Web Services)" Snapshots. As shown in the figure, this classic architecture of load balancing.

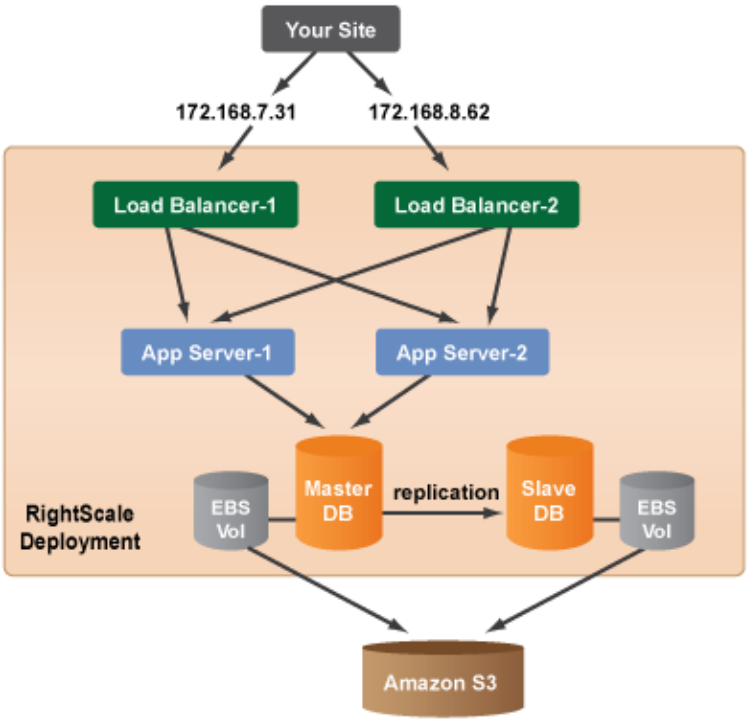


Figure 3: Classic load balancer architecture

RightScale has found HAProxy to be the highest performance, most reliable and dependable software load balancer, especially when you need to scale very quickly. When HAProxy is used in conjunction with Apache, it allows you to handle SSL connections and filter incoming requests to serve static content locally or direct traffic to one or more server arrays. This additional use of Apache adds a lot of flexibility with only a negligible amount of latency. Our default load balancers are configured to run in this manner.[13]

By default, clients round-robin through the load balancers by a DNS A-name record. In the unlikely event that a load balancer goes down, clients will simply retry the next address returned after a browser-specific timeout. You can also configure a hot-swappable standby, which monitors your load balancer's and assumes the A-record or elastic IP of a downed machine.[13]

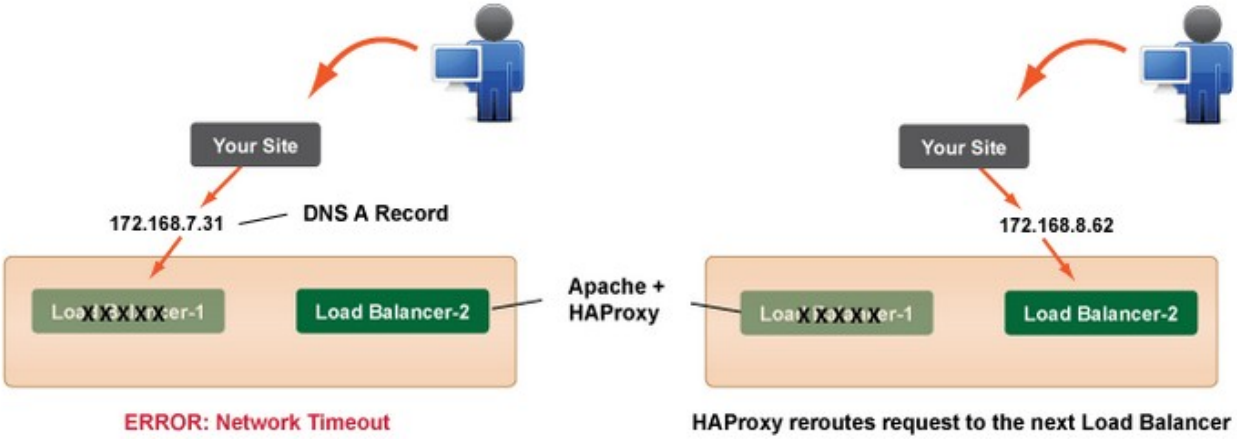


Figure 4: Case load balancer goes down

For the majority of web applications, it is absolutely essential to run a reliable database. RightScale provides the necessary master/slave Server-Templates that are already pre-configured for automatic replication and regular backups. They are also designed to easily handle a variety of fail-over scenarios, such as a failure of the Master-DB.[13]

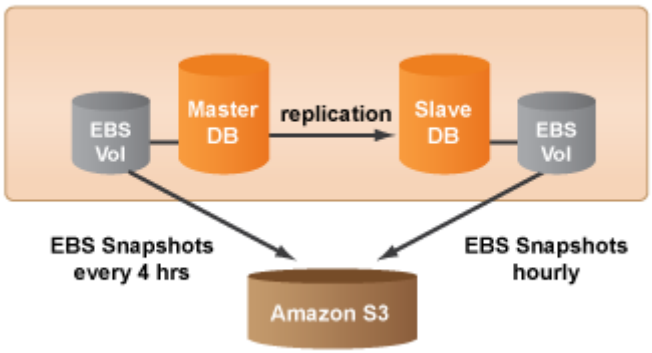


Figure 5: EBS Snapshot backups

By default, the database ServerTemplates perform asynchronous replication so that your database is always present on two live servers (i.e. Master-DB, Slave-DB). If you are using one of the MySQL-EBS setups, backups of your database will be stored as EBS Snapshots. Any database restore will use one of the previously saved EBS Snapshots to restore the data. [13]

Notes:

✔ Under normal conditions, there is no need to scale the number of load balancers because two of these servers can handle a very large amount of traffic.

✔ However, if you do need to automatically scale your load balancers, this can be done by creating two scripts. One that registers and deregisters A records with your DNS provider when your servers are launched and when they are terminated. The second script loads an HAProxy config when it is launched

✔ Besides the EBS snapshots backup, there is also a promote to master operational script on all RightScale database templates, so that any slave server can assume the master role simply by running one script on the server. This script makes it easy to recover from a failure of the master database server.

Conclusion

High availability, basically, is redundancy in the system: if one server fails, the others take over the failed server's load transparently. The failure of an individual server is invisible to the client.

HAProxy (High Availability Proxy) is a common choice for load balancing, as it can handle load balancing at multiple layers, and for different kinds of servers, including database servers.

Chapter 3: Proof of Concept -Technical Guide

Introduction

The following figure describes the main components of our high availability architecture and the interaction between them ,which we are going to explain them by details in this chapter and detail the sequential configurations.

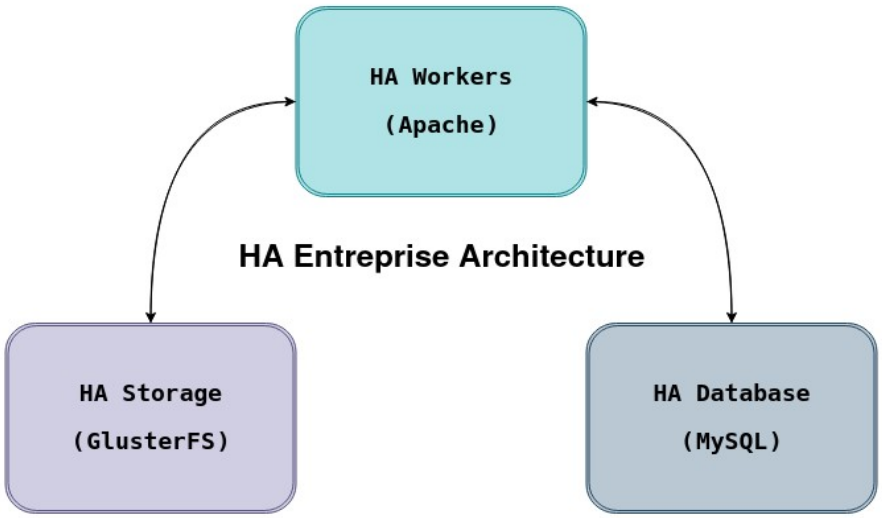


Figure 6: Illustration of the High Availibility Blocks and theirGlobal Interactions

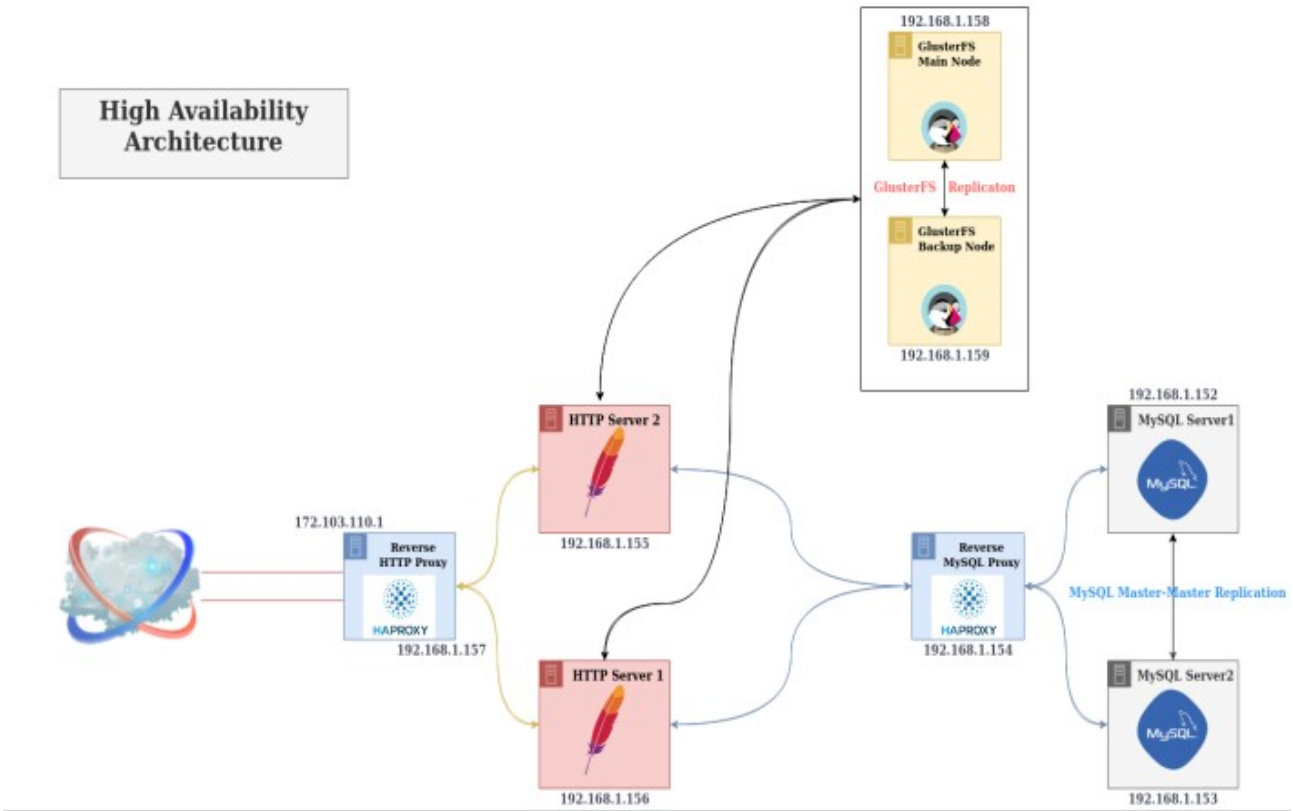


Figure 7: Adopted Architecture Components View and their Interactions

I.Before You Begin

1. This guide uses a total of nine nodes, all nodes running CentOS 7 with SELinux enabled, and all within the same data center. You can create them all in the beginning, or as you follow along.

2. You should also be familiar with securing your server strategy , and follow best security practices as you create your servers. Do not create firewall rules yet, as we'll be handling that step in our guide.

3. The nodes we create in this guide will use the following hostname conventions:

    - File system nodes - gfs1, gfs2.
    - Database nodes - cn1, cn2.
    - Application nodes - wa1, wa2.
    - HA Reverse Proxy nodes - haproxy_mysql, haproxy_http.

    You can call your nodes anything you like, but try to keep the naming consistent for organizational purposes. When you see one of the above names, be sure to substitute the hostname you configured for the corresponding node.

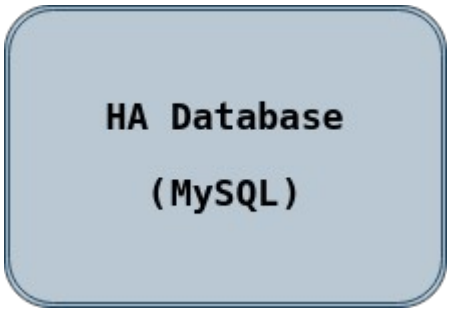
```
sudo yum update
```

4. To create a private network among your nodes, you'll need a private IP address for each.

Remember to update each node's package repositories before attempting to install software:

Note
Most steps in this guide require root privileges. Be sure you're entering the commands as root, or using sudo if you're using a limited user account.

Caution
During this guide, we assume that you have prior knowledge of Linux commands, Linux Administration policies, SELinux , Sudo, Package Manager, Logs, Networking and Troubleshooting.



II.High-Availability MySQL cluster with load balancing using HAProxy

This section describes how to set up highly available MySQL cluster, using the following components: MySQL cluster from two master nodes, HAProxy.

1.Preparing

First we'll need to choose few subnets for the MySQL replications and for the HAProxy, better separate them and if your server have a few network interface put these subnets on different interfaces as well.

To make things simpler, we've choosed the same network for HAProxy and DB traffic.

192.168.1.1/24 - network for DB traffic and HAProxy.

HOSTNAME	IP	OS	DESCRIPTION
cn1	192.168.1.152	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	MySQL cluster node 1 (Master 1)
cn2	192.168.1.153	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	MySQL cluster node 1 (Master 2)
hapeoxy_mysql	192.168.1.154	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	HAProxy MySQL Load Balancer

2. Configuring MySQL servers with master - master replication

First we need to install MySQL on both of ours servers:

MySQL must be installed from the community repository.

- Download and add the repository, then update.

```
Wget http://repo.mysql.com/mysql-community-release-el6-5.noarch.rpm
sudo rpm -ivh mysql-community-release-el6-5.noarch.rpm
yum update
```

- *Install MySQL as usual and start the service. During installation, you will be asked if you want to accept the results from the .rpm file's GPG verification. If no error or mismatch occurs, enter **y**.*

```
sudo yum install mysql-server
sudo systemctl start mysqld
```

- *Configure MySQL to start automatically on reboot:*

```
sudo chkconfig mysqld on
chkconfig --list mysqld
```

- *Run the **mysql_secure_installation** script to address several security concerns in a default MySQL installation.*

```
sudo mysql_secure_installation
```

- *Edit the **/etc/mysql/my.cnf** on first and second node, to enable replication between MySQL servers and make them use IPS from 192.168.0.0/24 subnet:*

- *Server1 configuration :*

```
[mysqld]
bind-address    = 192.168.1.152
server_id        = 1
log_bin          = /var/log/mysql/mysql-bin.log
log_bin_index      = /var/log/mysql/mysql-bin.log.index
relay_log         = /var/log/mysql/mysql-relay-bin
relay_log_index    = /var/log/mysql/mysql-relay-bin.index
expire_logs_days    = 10
max_binlog_size    = 100M
log_slave_updates   = 1
sync_binlog = 1
sync-relay-log = 1
sync-relay-log-info = 1
sync-master-info = 1
auto-increment-increment = 1
Auto-increment-offset = 1
connect_timeout = 300
max_allowed_packet = 128M
wait_timeout = 28800
interactive_timeout = 28800
innodb_log_file_size = 128 MB
```

- *Server2 configuration :*

```
[mysqld]
bind-address    = 192.168.1.153
server_id          = 2
log_bin            = /var/log/mysql/mysql-bin.log
log_bin_index      = /var/log/mysql/mysql-bin.log.index
relay_log          = /var/log/mysql/mysql-relay-bin
relay_log_index     = /var/log/mysql/mysql-relay-bin.index
expire_logs_days    = 10
max_binlog_size     = 100M
log_slave_updates   = 1
sync_binlog = 1
sync-relay-log = 1
sync-relay-log-info = 1
sync-master-info = 1
auto-increment-increment = 1
Auto-increment-offset = 1
connect_timeout = 300
max_allowed_packet = 128M
wait_timeout = 28800
interactive_timeout = 28800
innodb_log_file_size = 128 MB
```

- *Then restart them both and make sure MySQL leaf on specified IP:*

```
Server1# systemctl restart mysql
server1# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
    PID/Program name
Tcp        0      0 192.168.1.152:3306      0.0.0.0:*               LISTEN      9057/
    mysqld

server2# systemctl restart mysql
server2# netstat -ntlpActive Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
    PID/Program name
tcp        0      0 192.168.1.153:3306      0.0.0.0:*               LISTEN      8740/
    mysqld
```

- *Now will create a user for replication between databases, you can use **pwgen** utility to generate strong enough password. Connect to each MySQL servers and create this user with IP from opposite server:*

```
server1# mysql -u root -p
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replicauser'@'192.168.1.153'
    IDENTIFIED BY 'somestrongpassword';

server2# mysql -u root -p
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replicauser'@'192.168.1.152'
    IDENTIFIED BY 'somestrongpassword';
```

- *Check that **replicauser** have access for each MySQL server :*

```
server1# mysql -u replicauser -p -h 192.168.1.153
Enter password: somestrongpassword
Welcome to the ...

server2# mysql -u replicauser -p -h 192.168.1.152
Enter password: somestrongpassword
Welcome to the ...
```

- *Fine, now we can continue with configuring replication between MySQL servers. From that time better to have opened two consoles from both of MySQL servers, as we need to input commands, based on output from another server.*
  Get the MySQL master status on server1:

```
server1# mysql -u root -p

mysql> SHOW MASTER STATUS;
+--------------------------+------------+--------------------+--------------------------+
| File                     | Position | Binlog_Do_DB | Binlog_Ignore_DB  |
+--------------------------+------------+--------------------+--------------------------+
| mysql-bin.000002 |     120      |                          |
    |
+--------------------------+------------+--------------------+--------------------------+
1 row in set (0.00 sec)
```

- *We'll need the **File** and **Position** information, from this output. Open the MySQL console on server2 and configure the slave relation with first server.*

```
server2# mysql -u root –p
mysql> STOP SLAVE;
mysql> CHANGE MASTER TO master_host='192.168.1.153',
    master_port=3306, master_user='replicauser',
    master_password='somestrongpassword', master_log_file='mysql-
    bin.000002', master_log_pos=120;
mysql> START SLAVE;
```

- *Now query the master status from server2 and configure slave relation for MySQL on first server.*

```
server2# mysql -u root -p

mysql> SHOW MASTER STATUS;
+--------------------------+------------+--------------------+--------------------------+
| File                     | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------------+------------+--------------------+--------------------------+
| mysql-bin.000002 |    120     |                    |                                |
+--------------------------+------------+--------------------+--------------------------+
1 row in set (0.00 sec)



server1# mysql -u root –p
mysql> STOP SLAVE;
mysql> CHANGE MASTER TO master_host='192.168.1.152',
    master_port=3306, master_user='replicauser',
    master_password='somestrongpassword', master_log_file='mysql-
    bin.000002', master_log_pos=120;
mysql> START SLAVE;
```

- *If all was done right, we must have a working replication between MySQL masters. You can create some test DB and check this.*

```
server1# mysql -u root -p
mysql> CREATE DATABASE TESTDB;
mysql> CREATE TABLE TESTDB.REPLICA (`id` varchar(40));
```

- *Then check this database was appeared on second server as well:*

```
server2# mysql -u root -p
mysql> SHOW TABLES IN TESTDB;
+----------------------------+
| Tables_in_TESTDB   |
+----------------------------+
| REPLICA                    |
+----------------------------+
1 row in set (0.00 sec)
```

*And as you can see, the **TESTDB** base was successfully replicated to server2. We just completed a first stage of creating our failover cluster.*

3. Configuring HAProxy on both servers

In second stage we'll install and configure HAProxy on a third server, for balancing a incoming requests between MySQL servers.

First we need to add additional user on our MySQL servers (user must be created without any password), this user will be used by HAProxy for checking a health status of MySQL servers.

```
server1# mysql -u root -p
mysql> CREATE USER 'haproxy_check'@'%';
mysql> FLUSH PRIVILEGES;
```

You can create this user on any of our MySQL servers, as we have a replication configured between them. Check that user was added, using this command:

```
server1# mysql -u root -p -e "SELECT User, Host FROM mysql.user"
Enter password:
+--------------------+---------------------------+
| User     | Host                |
+--------------------+---------------------------+
| haproxy_check       | %                   |
| replicauser    | 192.168.1.153   |
| root     | localhost           |
+--------------------+---------------------------+
server2# mysql -u root -p -e "SELECT User, Host FROM mysql.user"
Enter password:
+--------------------+---------------------------+
| User     | Host                |
+--------------------+---------------------------+
| haproxy_check       | %                   |
| replicauser    | 192.168.1.152   |
| root     | localhost           |
+--------------------+---------------------------+
```

Also let’s create a user with root privileges, for making some test requests later:

```
server1# mysql -u root -p
mysql> CREATE USER 'haproxy_root'@'%' IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'haproxy_root'@'%';
```

Now it’s time for HAProxy installation:

```
server3# yum install haproxy -y
```

Save original config and create new one:

```
server3# mv /etc/haproxy/haproxy.cfg{,.back}
server3# vi /etc/haproxy/haproxy.cfg
```

Next add this configuration:

```
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
        user haproxy
        group haproxy
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
        mode http
        log global
        retries 3
        timeout connect 1m
        timeout server 480m
        timeout client 480m
#---------------------------------------------------------------------
#HAProxy Monitoring Config
#---------------------------------------------------------------------
listen stats
        bind 0.0.0.0:9000
        stats enable
        stats hide-version
        stats uri /stats
        stats auth haproxy:godmode
#---------------------------------------------------------------------
# FrontEnd Configuration
#---------------------------------------------------------------------
Frontend main
        bind 0.0.0.0:3306
        default_backend app-main
#---------------------------------------------------------------------
# BackEnd roundrobin as balance algorithm
#---------------------------------------------------------------------
backend app-main
        mode tcp
        option mysql-check user haproxy_check
        balance roundrobin
        server cn1 192.168.1.152:3306 check weight 256 inter 2s
    downinter 5s rise 3 fall 2
        server cn2 192.168.1.153:3306 check weight  1    inter 2s
    downinter 5s rise 3 fall 2
```

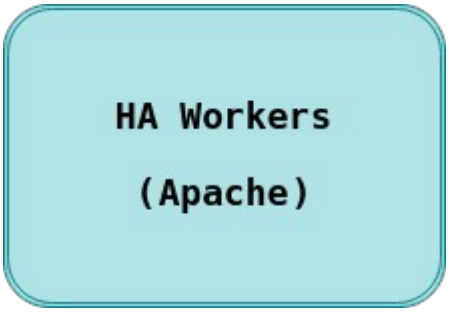
Note

Given that this architecture is designed to work with a website created with PrestaShop CMS, which interfer with the database settings and imposes that both of the variables ***auto_increment_increment*** and ***auto_increment_offset*** are set to 1. With this settings, the database cluster nodes can easily fall into saving records with the same primary key, and fall into duplicate key error.
Note also if you are not configuring this architecture for PrestaShop CMS, you can set the variables as below :

variables	mysql_node1	mysql_node2
auto_increment_increment	1	2
auto_increment_offset	2	2

We proposed a fast solution to prevent this kind of behaviour, by keeping the same master-2-master replication and making use of the HAProxy weight variable which acts as a priority by which HAProxy decides which nodes will process the given request. Thus we set the `cn1` to the highest weight (priority) and `cn2` to the lowest weight.

```
❯ ls
AbstractInstall.php  Database.php  LanguageList.php  Simplexml.php  System.php   XmlLoader.php
DatabaseDump.php     Install.php   Language.php      SqlLoader.php  Upgrade.php
❯ grep -i 'if (!Db::checkAutoIncrement' -A 2 Database.php
                    if (!Db::checkAutoIncrement($server, $login, $password)) {
                        $errors[] = $this->translator->trans('The values of auto_increment increment and offset must
be set to 1', array(), 'Install');
                    }
```



III.Setting up Apache Web Servers with load balancing using HAProxy

1.Preparing

As a primary step we need to choose subnets for the apache web servers and for the HAProxy, better separate them and if your server have a few network interface put these subnets on different interfaces as well.

To make things simpler, we've choosed the same network for HAProxy and Apache traffic.

HOSTNAME	IP	OS	DESCRIPTION
wa1	192.168.1.155	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	Apache Web Server 1
wa2	192.168.1.156	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	Apache Web Server 2
hapeoxy_http	192.168.1.157	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	HAProxy HTTP Load Balancer

2.Installing Apache on CentOS7

Step 1: Update Software Versions List

Ensure you are using the latest versions of the software. In a terminal window, input the command:

```
sudo yum update
```

The system should reach out to the software repositories and refresh the list to the latest versions.

Step 2: Install Apache

To install Apache on your CentOS server, use the following command:

```
sudo yum install httpd
```

The system should download and install the Apache software packages.

Step 3: Activate Apache

To activate Apache, start its service first.

1. Enter the following command in a terminal window:

```
sudo systemctl start httpd
```

This will start the Apache service.

2. Next, set the Apache service to start when the system boots:

```
sudo systemctl enable httpd
```

Step 4: Verify Apache Service

Display information about Apache, and verify it’s currently running with:

```
sudo systemctl status httpd
```

Figure 8: Verify Apache Service

Step 5: Configure firewalld to Allow Apache Traffic

In a standard installation, CentOS 7 is set to prevent traffic to Apache.

Normal web traffic uses the **http** protocol on Port 80, while encrypted web traffic uses the **https** protocol, on Port 443.

1. Modify your firewall to allow connections on these ports using the following commands:

```
sudo firewall-cmd —permanent —add-port=80/tcp
sudo firewall-cmd —permanent —add-port=443/tcp
```

2. Once these complete successfully, reload the firewall to apply the changes with the command:

```
sudo firewall-cmd —reload
```

Step 6: Install the necessary PHP modules

Note
PrestaShop 1.7.0 or newer requires php version >=php5.6, we will proceed with php5.6

To install PHP 5.6 on CentOS 7, you need to install remi repository.

```
yum install -y epel-release
wget https://rpms.remirepo.net/enterprise/remi-release-7.rpm
rpm -ivh remi-release-7.rpm
yum update
```

If you have older version of PHP, uninstall it with

```
yum remove php-pdo php-odbc php-xml php-soap php-cli php-devel php-pspell php php-intl php-xmlrpc php-process php-common php-gd php-mysqlnd php-mbstring
```

Now you can install PHP 5.6 with

```
yum install php56 php56-php php56-php-mysqlnd php56-php-gd php56-php-mcrypt php56-php-mbstring php56-php-xml php56-php-cli php56-php-intl
```

Now it’s time for HAProxy installation:

```
server6# yum install haproxy -y
```

Save original config and create new one:

```
server3# mv /etc/haproxy/haproxy.cfg{,.back}
server3# vi /etc/haproxy/haproxy.cfg
```

Note
1.If your application makes use of **SSL certificates**, then some decisions need to be made about how to use them with a load balancer, for further reading check this link.
2.**Performing System Tuning for HAProxy and Sizing recommendations**, It is recommanded to check this link and the official HAProxy Documentation to scale your servers for better performance.

We will add the following configuration :

```
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
        user haproxy
        group haproxy
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
        mode http
        log global
        retries 3
        timeout connect 5s
        timeout server 480m
        timeout client 480m
#---------------------------------------------------------------------
#HAProxy Monitoring Config
#---------------------------------------------------------------------
listen stats
        bind 0.0.0.0:9000
        stats enable
        stats hide-version
        stats uri /stats
        stats auth haproxy:godmode
#---------------------------------------------------------------------
# FrontEnd Configuration
#---------------------------------------------------------------------
frontend hapoxy
        mode http
        maxconn 20000
        bind *:80 name http
        default_backend webservers
#---------------------------------------------------------------------
# BackEnd roundrobin as balance algorithm
#---------------------------------------------------------------------
backend webservers
        mode http
        balance roundrobin
        timeout connect 10s
        timeout server 1m
        server wa1 192.168.1.155:80 check
        server wa2 192.168.1.156:80 check
```

IV.Setting up Highly Availiable Storage Cluster GlusterFS

1.Preparing

The first step towards creating a high-availability setup is to install and configure a file system using GlusterFS. In this section, we'll be using two 2GB nodes named `gluster1`, and `gluster2`.

Edit the `/etc/hosts` file on each node to match the following, substituting your own private IP addresses, fully qualified domain names, and host names:

```
192.168.1.158 gfs1   gluster1
192.168.1.159 gfs2   gluster2
```

HOSTNAME	IP	LINUX	DESCRIPTION
gfs1	192.168.1.158	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	GlusterFS main FileServer
gfs2	192.168.1.159	Centos7 (Linux 3.10.0+1127.18.2.el7.x86_64)	GlusterFS backup FileServer

2.Install GlusterFS

These steps should be run on each file system node in your cluster.

1. Add the `centos-release-gluster` repository, which will allow you to install the GlusterFS server edition package:

```
yum install epel-release
yum install centos-release-gluster
yum install glusterfs-server
```

2. Start the GlusterFS daemon:

```
systemctl start glusterd
```

3.Add Firewall Rules

Run the following commands on each node in your GlusterFS cluster.

1. Add firewall rules that allow the GlusterFS service to communicate between your trusted servers. Replace the IP addresses below with the private IP addresses of your hosts:

```
firewall-cmd --zone=internal --add-service=glusterfs --permanent
firewall-cmd --zone=internal --add-source=192.168.1.158/24 --permanent
firewall-cmd --zone=internal –add-source=192.168.1.159/24 --permanent
```

2. Reload your firewall configuration:

```
firewall-cmd --reload
```

3. Enable the `firewalld` and `glusterd` services to have them start automatically on boot:

```
systemctl enable firewalld glusterd
```

4.Configure GlusterFS

1. Create a *trusted storage pool*. A storage pool is a trusted network of file servers that will communicate to share data. You only need to run this command on one of your nodes. We use `gluster1` in this example, probing each of the other nodes we want to add to our storage pool:

```
gluster peer probe gluster2
```

2. On each node, create a directory to store the files to be replicated. We'll use `/exports/cms-vol`, but you can create this directory wherever you like, and with a name of your choosing:

```
mkdir -p /exports/cms-vol
```

3. Create a distributed replicated volume. This step needs to be done on only one of the nodes in your pool. In this example, we create the volume `cms-vol`, which sets `/exports/cms-vol` as the brick storage directory for each node in the network. Replace the volume name and the hostname values with your own:

```
gluster volume create cms-vol replica 2 gluster1:/exports/cms-vol gluster2:/
   exports/cms-vol   force
```

4. Start the volume to enable replication among servers in your pool. Replace `cms-vol` with your volume name:

```
gluster volume start cms-vol
```

Check the configuration by running `gluster volume info`. If everything is working correctly, the output should resemble the following. Check that each brick is listed here:

```
gluster volume info cms-vol
```

```
root@glusterfs1:~ # gluster volume info prestashop

Volume Name: prestashop
Type: Replicate
Volume ID: 604cabaf-1d27-471a-b34f-202e534eaa83
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: gfs1:/exports/html
Brick2: gfs2:/exports/html
Options Reconfigured:
performance.client-io-threads: off
nfs.disable: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
root@glusterfs1:~ #
```

V.PrestaShop (Optional)

If you're installing PrestaShop to manage your new highly available website, we'll explain how to do so here. If you're using your cluster to serve a custom application, website, or for another purpose, you may skip this section.

1.Mount the Gluster Filesystem

Next, mount the Gluster volume on the application servers. The steps in this section should be performed on each Apache server node.

1. Install `glusterfs-fuse`:

```
yum install glusterfs-fuse -y
```

2. Add the following line to `/etc/fstab`, substituting your own GlusterFS hostnames for `gluster1`, and `gluster2`, and your volume name for *`cms-vol`* if appropriate:

```
gluster1:/cms-vol  /srv/www  glusterfs defaults,_netdev,backup-volfile-servers=gluster2 0 0
```

3. Create the `/srv/www/` directory and mount the volume to it:

```
mkdir /srv/www
mount /srv/www
```

2.Test Replication (Optional)

This section explains how to test file replication between servers in your pool. Testing in this manner should not be done in a live production environment.

1. Mount the volume on one of your hosts. This example uses `gluster1`, but you can use any file system node:

```
mount -t glusterfs gluster1:/cms-vol /mnt
```

2. Create an empty file called `test` within `/mnt`, where we mounted the volume. Do not write directly to `/exports/cms-vol` or its subdirectories.

```
touch /mnt/test
```

3. From your other file system nodes, check the contents of your volume:

```
ls /exports/cms-vol
```

   If replication is working properly, the `test` file you created on the mounted volume should now show up on your other hosts.

4. 4.Delete the test file from `/mnt` (on the same host used to create it) and unmount the volume before using GlusterFS in production:

```
rm /mnt/test
umount /mnt
```

Note
In order for the Apache Web Server to work properly, we need to change ownership on all of your application servers of the directory (mounted throught glusterFS) to the Apache user:

```
chown apache:apache -R /srv/wwwl/prestashop/
```

3.Download and unpack PrestaShop

Download the latest stable release of PrestaShop to your server. Currently, it is version 1.7.4.2

```
wget
https://download.prestashop.com/download/releases/prestashop_1.7.4.2.zip
```

Unpack the archive to the document root directory of your server

```
unzip prestashop_1.7.4.2.zip -d /srv/www
unzip prestashop_1.7.4.2.zip
unzip prestashop.zip -d /srv/wwwl/prestashop/
```

and set the correct permissions to the PrestaShop directory

```
chown apache:apache -R /srv/wwwl/prestashop/
```

4.Create Apache Virtual Host

At this point, you should be able to access your PrestaShop website using your server's IP address. We will create a new Apache virtual host, so we can be able to access it with a domain name. Create a new virtual host directive with the following content. **vi /etc/httpd/conf.d/prestashop.conf**:

```
ServerAdmin admin@your-domain.com
DocumentRoot /srv/www/prestashop/
ServerName your-domain.com
ServerAlias www.your-domain.com
Options +FollowSymlinks
AllowOverride All
ErrorLog /var/log/httpd/prestashop-error_log
CustomLog /var/log/httpd/prestashop-access_log common
```

Save the file and restart the web server for the changes to take effect

```
systemctl restart httpd
```

5.Access and install PrestaShop

Now, since we already installed and configured all necessary services and packages, and downloaded PrestaShop, you should be able to access your PrestaShop e-commerce website at http://your-domain.com and follow the on-screen instructions to complete the installation from your favorite web browser. Once everything is properly installed, for security purposes you should remove the installation directory

```
rm -rf /srv/www/prestashop/install/
```

Conclusion

You can always try to setup such architecture on other Linux distributions and even on Windows Server(not recommended for security & stability reasons).You will find various documentations and tutorials that will guide you from setting-up to troubleshooting.

Conclusion

High availability is an important subset of reliability engineering, focused towards assuring that a system or component has a high level of operational performance in a given period of time. At a first glance, its implementation might seem quite complex; however, it can bring tremendous benefits for systems that require increased reliability.

HA is one of the biggest challenges for cloud computing and system engineers, which represents an increasing research field for the industry.

The proposed solution is generic, that can serve a majority of services (including Web Applications, AI calculations, IoT ... etc) with the cost of modifying the block's configurations. Besides , the nodes can be substituted with Docker containers orchestrated by Kubernetes to obtain an optimized, powerful and HA architectures.

Note that our architecture can be rated as weak in availability but optimized in terms of balancing since we have used only one reverse proxy load balancer in front of each block which is considered as a single point of failure. The failure of one of the proxies lead to the failure of the hole system which results to the non-availability. Imagine if the single load balancer above were to crash or fail, the entire system would be unreachable from all user requests, rendering the system completely unavailable. As a solution, we can use a virtual shared IP address for two or more HAProxies so we avoid the SPF . But due to the luck of resources ,we hardly get the chance to implement the proposed architecture.

Bibliography

[1] D. Ardian, A. Fatchur Rochim, and E. Didik Widianto, “Analisis Perbandingan Unjuk Kerja Sistem Penyeimbang Beban Web Server dengan HAProxy dan Pound Links,” J. Teknol. dan Sist. Komput.p. 28, 2013

[2] https://www.radware.com/resources/server_load_balancing.aspx

[3] https://www.digital-click.tn/

[4] https://www.citrix.com/glossary/load-balancing.html

[5] https://www.f5.com/services/resources/glossary/load-balancer

[6] https://whatis.techtarget.com/definition/Web-server

[7] http://www.haproxy.org/

[8] https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/

[9] https://www.siteground.com/tutorials/php-mysql/mysql/#:~:text=MySQL%20is%20a%20freely%20available,ease%20and%20flexibility%20of%20use.

[10] https://www.businessinsider.com/what-is-cache

[11] https://en.wikipedia.org/wiki/Content_management_system

[12] https://www.infoworld.com/article/2077921/server-load-balancing-architectures--part-1--transport-level-load-balancing.html

[13] https://docs.rightscale.com/cm/designers_guide/cm-designing-and-deploying-high-availability-websites.html