# A Whitepaper on

# How to Scale HCL Compass

By
Ganesh Kamath
(QA Manager)

## Table of Contents:

## A Brief Introduction

Single Compass server instance might not be able to provide high concurrent user loads and are prone to failures, downtime, and security attacks. To address these issues, multiple instances of Compass servers are clustered together, and a load balancer is placed in front of them.

A load balancer can enhance the stability, efficiency, security and high availability of Compass application. Additionally, a load balancer can:

- Reduce server workload
- Increase performance
- Reduce Single Point of Failure through redundancy and rebalancing workloads when a server fails
- Improve scalability by the addition of new servers
- Prevents from Security Attacks such as DDOS and other attacks.

## Audience

This white paper has been designed to provide a high level overview for Administrators, Application Support, Developers, Analysts or anyone eager to learn and quickly set up Load Balancer to enable HCL Compass Product for high user concurrency, fault tolerance, fail-over mechanism and security.

## Overview of Apache HTTP Server

The Apache HTTP Server is a free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache HTTP Server either can be used as a Web Server for hosting static and dynamic contents and as well be used as a Reverse Proxy for Load Balance mechanism.

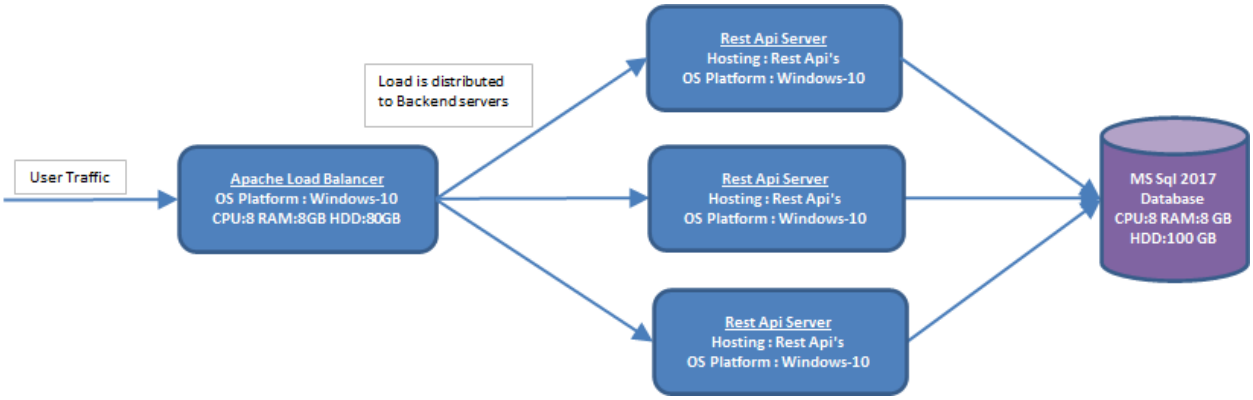
One of the biggest Advantages of Apache is in its ability to handle large volume of traffic with minimal configuration, making it light weight and efficient.

## Pre-requisites

Following server configurations are been used in setting up Apache server

- **Storage**: 100 GB
- **CPU**: 8
- **Memory**: 16 GB RAM
- **OS Platform**: Linux RHEL-8.0
- **Load Balancer**: Apache 2.4.37
- **Cryptography tool**: OpenSSL-1.1.1
- **License Server**: Local FlexNet License Server
- **Server Instance**: HCL Compass product hosting – Rest Api’s and Modern UI
- **Database**: MS Sql Server 2017

## Logical Architecture




**Note :**

1. The above diagram illustrates only a high-level view of setting up HCL Compass Rest Api server product with Apache Load Balancer.
2. Application owners need to pay attention and due diligence on setting up a secure network like DMZ zone and firewalls at your premises. How to set up a secure network is not in scope of this document.
3. Apache Load Balancer and Database are single point of failures. For this exercise, planning for redundancy of these servers is not in scope for now.
4. You need a License server for HCL Compass to function properly. Cloud based license servers are generally prone for latencies. Hence, a local license server is most preferred.

## Environment Set up

Create VM's as per the Logical Architecture as shown in above diagram. Here we are setting up 3 Compass Rest Api server and placing a load balancer in front of them. This document assumes that you have already installed HCL Compass Rest server on 3 Linux VM machines and each Compass VM are configured to point to same database. You can as well deploy Docker Compass on each VM.

The below steps are used for setting up Apache HTTP server.

1. Create a Virtual Machine or a Docker environment with Linux RHEL 8.0.
2. Configure same subnet mask across all VM's.
3. Create apache user and group on Apache VM.
4. Install VNC, SSH server for accessing virtual machines from remote.
5. Use Sudo or Root user access for installation of software's.
6. Always protect other users from accessing binary and configuration folders.

Now let us start installing Apache httpd software on Linux platform:

Login into Load Balancer VM. By default, Linux RHEL-8.0 comes with pre-installed Apache httpd software with version 2.4.37.

How to check if Apache is installed on your machine. Open a terminal console and run below command:

First change to root user :

```
$ Su -
```

```
$ httpd -v
```

The above command will display version number of Apache httpd if already installed. If not found, please follow below step using yum repository for installing the software.

```
$ yum install -y httpd
```

The above command will install Apache httpd latest supported and available version of the software.

Now start the Apache web server with following command –

```
$ service httpd start
```

To enable httpd to always start at boot-up time :

```
$ systemctl enable httpd
```

Check status of running httpd service

```
$ service httpd status
```

Configure firewall on the same Apache VM to open https port for accessing Apache server. This can be done using below commands

```
$sudo firewall-cmd --permanent --zone=public --add-service=https
```

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

You can test Apache is properly running by opening a browser and entering below url –

http://localhost

On the browser you should see the Apache server Test Page.With above steps you have successfully installed Apache web server.

By default, the Linux system firewall selinux blocks the HTTP connections to upstream backend servers. So, to let communication flow between Apache LB and backend servers, configure the selinux firewall network parameter.

```
$setsebool -P httpd_can_network_connect=1
```

## Create SSL Certificates

HCL Compass Rest Api's are enabled to access only on SSL (https protocol), you need to have https SSL enabled at http clients like browsers and Apache Load Balancer server as well. This means that you need to have SSL certificate installed in your browsers and Rest Api client's. Our Apache http server routes all incoming traffic over SSL to upstream backend server such as HCL Compass.

### Generate self-signed SSL certificate

For this exercise we are going to generate our own SSL certificate called as self-signed certificate. But for your production environment, you need to get SSL certificates from **trusted certificate authority**.

TLS/SSL works by using a combination of a public certificate and a private key. The SSL key is kept secret on the Apache Load Balancer server. It is used to encrypt content sent to clients. The SSL certificate is publicly shared with anyone requesting the content. It can be used to decrypt the content signed by the associated SSL key.

Create SSL folder under /etc folder :

```
$ sudo mkdir /etc/ssl
```

For enabling SSL in Apache load balancer, you need to install apache SSL module. Install it like this :

```
$ sudo yum install -y mod_ssl
```

You need to disable default configuration files under /etc/httpd/conf.d directory by adding ".temp" like below :

```
$ sudo mv ssl.conf ssl.conf.temp
```

```
$ sudo mv userdir.conf userdir.conf.temp
```

```
$ sudo mv welcome.conf welcome.conf.temp
```

```
$ sudo mv autoindex.conf autoindex.conf.temp
```

Now we can create a self-signed key and certificate pair with OpenSSL in a single command:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/apache-self-signed.key -out /etc/ssl/apache-selfsigned.crt
```

As we stated above, these options will create both a key and certificate files. We will be asked few questions about our server to embed the information correctly in the certificate.

Fill out the prompts appropriately. The most important line is the one that requests the Common Name (e.g. server FQDN or YOUR name). You need to enter the domain name associated with your Apache VM server or, more likely, your Apache server's public IP address.

The entirety of the prompts will look something like this:

```
Country Name (2 letter code) [AU]:US
```

State or Province Name (full name) [Some-State]:New York

Locality Name (eg, city) []:New York City

Organization Name (eg, company) [Internet Widgits Pty Ltd]: Test Company.

Organizational Unit Name (eg, section) []:Some Org Unit

Common Name (e.g. server FQDN or YOUR name) []:**LB_Server_IP_Address**

Email Address []:admin@your_domain.com

### Configure Apache for accessing Compass Server

Please note that SSL certificates are only valid for a specific IP address or domain name where you installed Apache server. While creating an SSL certificate ensure Common Name parameter to have either IP address or domain name.

On the Load Balancer machine, navigate to /etc/httpd/conf.d directory. Create a file "lb1-ssl.com.conf" and paste below configuration script.

```
#======Start of Apache Configuration==============
#lb1-ssl.com.conf
Listen 443 https
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn
Timeout 300
ProxyTimeout 300
ProxyBadHeader Ignore

# This configuration requires mod_ssl, mod_socache_shmcb, mod_rewrite, and mod_headers.
# You can redirect any traffic coming on port 80. But for this exercise you can disable it for time
# being.
# <VirtualHost *:80>
#  RewriteEngine On
#  RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
# </VirtualHost>
```

```
<VirtualHost < IP Addr-01>:443>
  SSLEngine on
  SSLCertificateFile  /etc/ssl/apache-self-signed.crt
  SSLCertificateKeyFile /etc/ssl/apache-self-signed.key

# we are using cookie session for routing same user's subsequent request to same backend
# server where login was accepted.
    Header always set Strict-Transport-Security "max-age=63072000"
    <Proxy "balancer://rest-api-cluster">
    Header add Set-Cookie "REST-API-ROUTE-ID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
        BalancerMember "https://<IP Addr-02>:<port>" route=Node-1 retry=60 timeout=300
        BalancerMember "https://<IP Addr-03>:<port>" route=Node-2 retry=60 timeout=300
        BalancerMember "https://<IP Addr-04>:<port>" route=Node-3 retry=60 timeout=300
        ProxySet stickysession=REST-API-ROUTE-ID
    </Proxy>

    <Proxy "balancer://legacy-ui-cluster">
    Header add Set-Cookie "LEGACY-UI-ROUTE-ID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
        BalancerMember "http://<IP_Addr_Compass_WAS_server-01>:12080" route=Node-1 retry=60 timeout=2000
        BalancerMember "http:// <IP_Addr_Compass_WAS_server-02>:12080" route=Node-2 retry=60 timeout=2000
        BalancerMember "http:// <IP_Addr_Compass_WAS_server-03>:12080" route=Node-3 retry=60 timeout=2000
        ProxySet stickysession=LEGACY-UI-ROUTE-ID
    </Proxy>

    <Proxy "balancer://modern-ui-cluster">
    Header add Set-Cookie "MODERN-UI-ROUTE-ID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
```

```
        BalancerMember "https:// <IP Addr-02>:8190" route=Node-1 retry=60 timeout=2000
        BalancerMember "https:// <IP Addr-02>:8190" route=Node-2 retry=60 timeout=2000
        BalancerMember "https:// <IP Addr-02>:8190" route=Node-3 retry=60 timeout=2000
        ProxySet stickysession=MODERN-UI-ROUTE-ID
    </Proxy>
    ProxyPass /ccmweb/rest balancer://rest-api-cluster/ccmweb/rest
    ProxyPassReverse /ccmweb/rest balancer://rest-api-cluster/ccmweb/rest
    ProxyPass /cqweb balancer://legacy-ui-cluster/cqweb/
    ProxyPassReverse /cqweb balancer://legacy-ui-cluster/cqweb/
    ProxyPass / balancer://modern-ui-cluster/
    ProxyPassReverse / balancer://modern-ui-cluster/
</VirtualHost>

#Global SSL configurations
SSLProxyEngine on
SSLSessionCache         shmcb:/run/httpd/sslcache(512000)
SSLSessionCacheTimeout  300

# Since we are using a self-signed SSL certificate, hence disable SSL verifications
# For your production environment, please turn-on below SSL directive for additional security.
SSLProxyVerify none
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off

# Generally old browsers support older version of SSL protocol, we can accept all incoming
# protocol requests to support them for now. But for production you can enable to accept only
# latest secure protocol like TLSv1.2
#SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
SSLProtocol             all
```

SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384

SSLHonorCipherOrder off

SSLSessionTickets off

#======= End of Apache Configuration ===============

Please note that the above sample script is created as a demo version only for this exercise. You need to fine tune for more security and update the script accordingly for your production environment. Please refer to Apache official documentation to understand more on each directive's.

In the above script, update "SSLCertificateFile" and "SSLCertificateKeyFile" directive parameters to point to the path of SSL certificate and key file where created and ensure each "BalancerMember" directive has URL pointing to 3 compass rest servers.

Save the configuration file and restart httpd with below command:

$ sudo systemctl restart httpd

You can check if our apache has started successfully, run below command :

$ sudo systemctl status httpd

Check the command output if the httpd instance is active and started successfully.

If httpd has not started, you can debug the configuration script using below command :

$ httpd -t

To list down all available modules installed in Apache issue below command :

$ httpd -M

To check if your default website is configured as primary virtual host, use below command :

$ httpd -D DUMP_VHOSTS

Now start Rest Api Server on each Compass Rest Api Linux VM.

cd /opt/hcl/ccm/compass/compass-rest-server-distribution/bin

start.sh

Now its time to check if Rest Api's are working :

Hope you have set up repo on compass server. In order to get a list of repos, issue below curl command on compass VM console terminal :

$ curl -k https://Compass_Server_IP_Address:8190/ccmweb/rest/repos

This will output list of available repositories.

Now check if load balancer is forwarding Rest Api traffic to Compass Rest Api server, issue below command in the terminal :

$ curl -k https://LB_Server_IP_Address/ccmweb/rest/repos

Now access the same above curl command using ip address of Apache Load balancer server without any port. If you see the output from Rest Api's which means our set up is successful.

Please follow SWAGGER documentation for Compass Rest Api specifications.

For Accessing modern UI's from browser client, open a browser and enter below url of load balancer:

https://LB_IP_Address/

For Accessing Legacy UI's from browser client, open a browser and enter below url of load balancer :

http://LB_IP_Address/cqweb

With this you should be able to access Legacy UI, Modern UI's and Rest Api's as well.

## Securing and harden Apache HTTP Server

Here below are few security considerations that are highlighted when setting up Apache http server. For more security set up please visit Apache official website for securing your production server.

1. **Prevent web server version being exposed in the HTTP header:**

   Update httpd.conf file with below:

   ```
   ServerTokens Prod
   ServerSignature Off
   ```

2. **Disable directory listing:**
   Update httpd.conf file with below directives:

   ```
   <Directory /var/httpd/www>
       Options None
   </Directory>
   ```

3. **Disable Etag :**
   Implement as below in httpd.conf file –

   ```
   FileETag None
   ```

4. **Run Apache from a non-privileged account:**

   A default installation runs as nobody or daemon. Using a separate non-privileged user for Apache is always good practice.

   The idea here is to protect other services running in case of any security breach.

Create a user and group called apache

```
$ groupadd apache
```

```
$ useradd –G apache apache
```

Change apache installation directory ownership to a newly created non-privileged user:

```
$ chown –R apache:apache /etc/apache
```

Edit httpd.conf file to have user and group directive to use newly created user and group name.

```
User apache
```

```
Group apache
```

5. **Protect binary and configuration directory permission:**

   By default, permission for binary and configuration is 755. This means any other users can able to view the configuration files.
   In order to prevent others viewing configuration files, Change permission of bin and conf folders as below:

```
chmod –R 750 bin conf
```

6. **System Settings Protection:**

   In a default installation, users can override apache configuration using .htaccess. If you want to stop users from changing your apache server settings, you can add AllowOverride to None as shown below. This must be done at the root level.

   Edit httpd.conf to implement as :

```
<Directory />
   Options -Indexes -Includes
  AllowOverride None
</Directory>
```

   Note: if you have multiple Directory directives in your environment, you should consider doing the same for all.

7. **Limit HTTP Request Methods:**

```
<LimitExcept GET POST HEAD PUT DELETE CONNECT>
   deny from all
</LimitExcept>
```

8. **Prevent Cross Site Scripting attacks:**

   Enable cookie with HttpOnly and Secure flag

```
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

9. **Clickjacking Attack:**

   Add the following directive and save the httpd.conf
   ```
   Header always append X-Frame-Options SAMEORIGIN
   ```

10. **X-XSS Protection:**

    Cross Site Scripting (XSS) protection can be bypassed in many browsers.

    Add following directives in httpd.conf

    Header set X-XSS-Protection "1; mode=block"

11. **Disable HTTP 1.0 Protocol:**

    HTTP 1.0 has security weakness related to session hijacking. We can disable this by using the mod_rewrite module.

    ```
    RewriteEngine On
    RewriteCond %{THE_REQUEST} !HTTP/1.1$
    RewriteRule .* - [F]
    ```

12. **Mod Security:**

    Mod Security is an open-source Web Application Firewall, which you can use with Apache. Please go to the apache website for enabling Mod Security.

    Note: For more security you can add Mod Security to your Apache Web Server, but at the same time you might experience performance impact. So, care must be taken based on the security requirements.

## Summary

Now that we have configured load balancer to route traffic to 3 instances of Compass server, you can test for fault tolerance and high availability features by bringing down one of the Compass servers to check if incoming requests are automatically routed to next available live server. During this transition process, user session is broken, and Compass will present Login page to initiate new session. The configurations are meant to lock a user request to always route to a same Compass server instance after login is validated and sessions are maintained. New user requests are routed to next available live servers so that users are loaded evenly on all 3 Compass servers.

Please go through Apache official website to learn more on traffic routing algorithms, security configurations and hardening of servers for better security.