Apache Reverse Proxy

***Introduction:**

Apache reverse proxy is a server that is configured near to the actual host server(eg., www.facebook.com). Reverse proxy server acts as an intercept between the client and the original host server. Client simply requests for a site or application then apache reverse proxy server takes in the requests and sends the request on behalf on the client to the original server. When the response is received from the original server, the reverse proxy server receives the response and forwards the response to the client.

In this case, the client will never know who was the actual server (server's IP), it will only know the proxy. The server will or won't know the actual client, depending on the configurations of the reverse proxy.

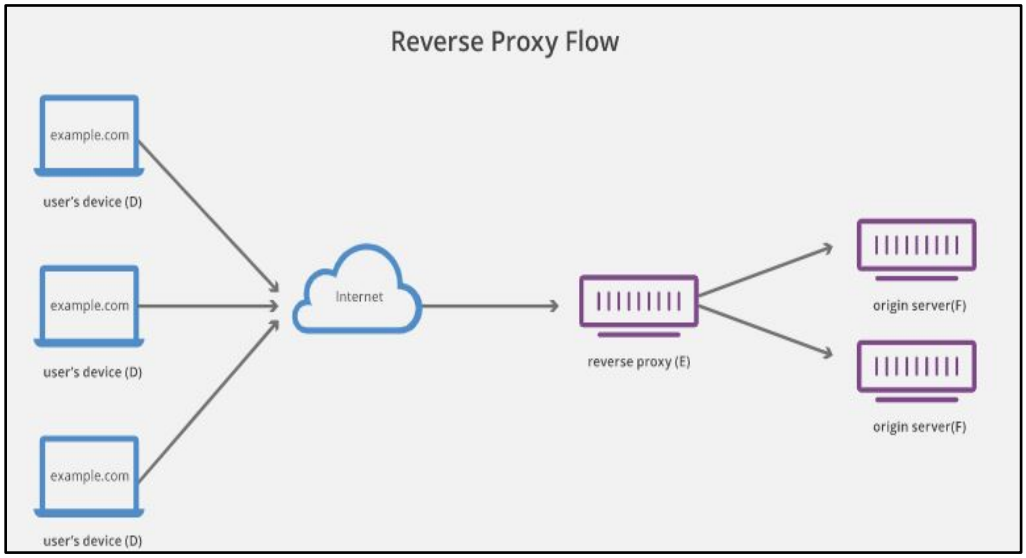

Fig 1., Reverse Proxy Flow

***Benefits :**

1) Security
2) Caching
3) Logging

***Prequsites needed before installing and configuring apache reverse proxy server**

1) Ubuntu 18.04LTS/Ubuntu 16.04 LTS

2) Apache web server

- **Installations:**

Following are the steps/command for installing Apache Reverse Proxy Server. Assuming you have a fresh ubuntu(X) VM/instance of cloud.

1) Updating the Operating system(Ubuntu 18.04LTS)

#sudo apt-get update

2) Upgrading Ubuntu. This is done in case any of the library hasn't been updated

#sudo apt-get upgrade -y

3) Installing apache2 on ubuntu.

#sudo apt-get install apache2 -y

4) To verify whether apache is installed or not type in the following command.

 # apache2 -version

You can find the directory in /etc/apache2/

5) Allow the firewall rules

#sudo ufw allow 80

#sudo ufw allow 443

#sudo ufw enable

Press y to enable the firewall.

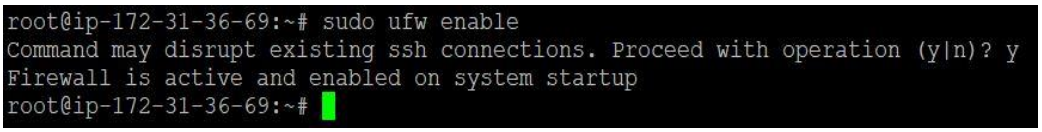

Fig 2., Enabling Firewall Rules

You can verify whether the rules are enabled or not. By the foll. Command.

#netstat –lntp as in Fig 3.,

```
root@ip-172-31-36-69:~# netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      704/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      977/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      16261/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      977/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      16261/apache2
```

Fig 3., netstat-lntp

6) To check whether the server is running properly, on the browser type

http://localhost/

It will direct you to the apache default page that is in the /var/www/html directory.

The default port is 80. We can change the port in ports.conf file in /etc/apache2 directory

- **Configuration of virtual host file.**

For implementing reverse proxy we need to make certain configurations of the apache web server and in the virtualhost file which is /etc/apache2/sites-available/000-default.conf

First, we need to enable modules that are necessary for configuring reverse proxy

By using a2enmod we will configure the proxy modules.

Following are the modules which we need for reverse proxy.

a2enmod proxy
a2enmod proxy_http
a2enmod proxy_ajp
a2enmod rewrite
a2enmod deflate
a2enmod headers
a2enmod proxy_balancer
a2enmod proxy_connect
a2enmod proxy_html

Now we will modify the default configurations in 000-default.conf.

***VirtualHost File(000-default.conf):**

1) **Virtual Host ➔**

Syntax: <VirtualHost> …… </VirtualHost>

this directive defines many other directives and configurations which the reverse proxy server uses to serve the requests from client and to serve the response from actual server to the client. Its default port value is set to port 80.

Eg., <VirtualHost _default_:80>

ServerAdmin ➔ it is useful when an error occurs. It is basically an error message which provide the admin email address e.g., contact@ xxxxx@gmail.com.

ServerName ➔ It is used to identify a virtual host. It sets the hostname. If no ServerName is specified the server asks for the hostname from the operating system

ServerAlias ➔ It just sets alternate names for the hosts. It goes in order of defininig alternate names in the configuration to check whether the name matches with the vhost.

ProxyPreserveHost ➔ The ProxyPreserveHost directive is used to instruct Apache mod_proxy, when acting as a reverse proxy, to preserve and retain the original Host: header from the client browser when constructing the proxied request to send to the target server.

ProxyPass ➔ This directives maps the actual server to the apache reverse proxy server.

e.g., Assume we have Apache as the Reverse Proxy Server and actual server as http://facebook.com/

I want to go to login page (/login). so i won't directly jump on to actual server with http://facebook.com/login . ProxyPass will write the request Url as http://--ip--/login/

ProxyPassReverse ➔ Before apache send the response to the browser, ProxyPassReverse changes/modifies the header that is received to apache from the server so that it won't redirect it to the other location. Same as the configuration of ProxyPass.

Below is the detailed example:

```
<VirtualHost _default_:80>
        ServerAdmin webmaster@facebook
        ServerName www.facebook.com
```

```
        ServerAlias facebook.com
        ProxyPreserveHost on
        ProxyPass / https://www.facebook.com/login
        ProxyPassReverse / https://www.facebook.com/login
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

After the configurations have been made we need to reload apache.

#sudo /etc/init.d/apache2 reload

OR

#service apache2 restart

After the configurations have been made on browser we only need to hit the IP i.e., http://127.0.0.1 and we will be directed to the facebook.com application as in Fig 4.,

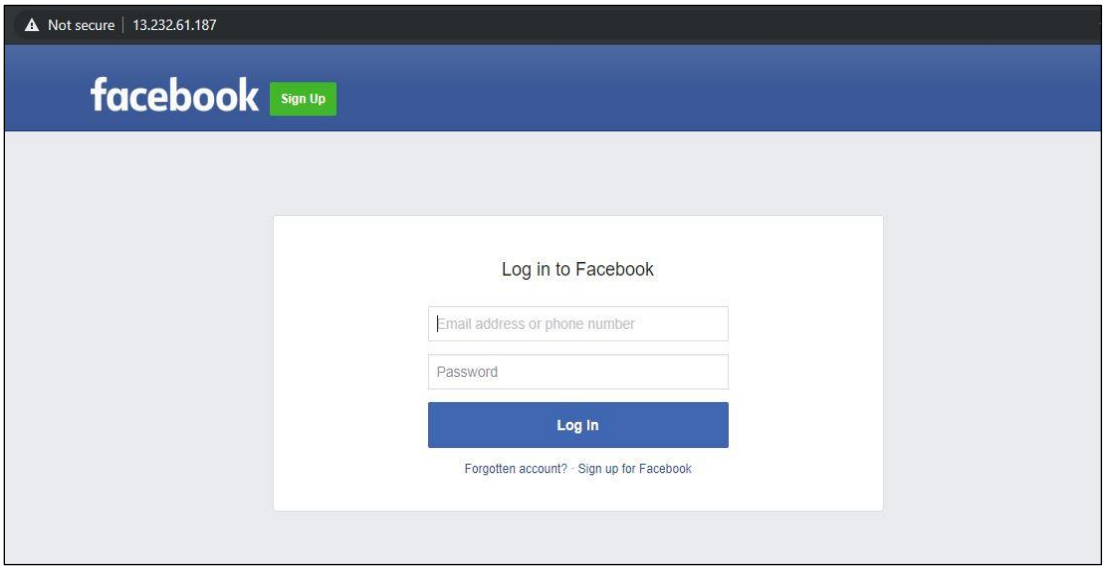

Fig 4., Application

Hence, we have successfully configured Reverse Proxy Server.

ReverseProxy::FormFiller

Going on ahead we will now configure Reverse Proxy FormFiller.

***Introduction:**

ReverseProxy::FormFiller makes an Apache server, positioned as a reverse-proxy, which fills and submits html forms in place of users.
It is mainly used for authentication purpose.

This module is based on perl module. Hence, first we need to enable mod_perl.

***Installation and Configuration to enable mod_perl.**

1) install the Reverse proxy by using
#cpanm ReverseProxy::FormFiller

2) download ReverseProxy File from
https://cpan.metacpan.org/authors/id/D/DE/DELTOMBE/ReverseProxy-FormFiller-0.5.tar.gz
Using wget and extract it using tar –xvzf to a directory ., lets say /opt/ReverseProxy

3) Get inside the folder and execute the following commands
#make
#make test
#make install

4) Reload the apache server.
#service apache2 reload

***Modification in virtualhost 000-default.conf**

Next we will use the application that is on the sever side. In this case we will use facebok.com

Next, we need to modify our virtual host file.
We need to add two important Modules that are :

1) PerlModule ReverseProxy::FormFiller → This directive ıt simply loads the Perl Modules using the package name
2) PerlSetVar FormFillerParamFile → It passes the configuration from defined location to the Perl modules using variable "FormFillerParamFile"

Our virtualhost file will look like follows:

```
<VirtualHost *>
  ServerName facebook.com

  PerlModule ReverseProxy::FormFiller
  PerlSetVar FormFillerParamFile "/etc/apache2/FormFiller/example"

  ProxyPass        / http://facebook.com/
  ProxyPassReverse / http://facebook.com/

  <Location /login/>
    RequestHeader unset Accept-Encoding
    Header        unset Content-Length
    PerlOutputFilterHandler ReverseProxy::FormFiller::output
  </Location>

  <Location /login/>
    PerlInputFilterHandler  ReverseProxy::FormFiller::input
  </Location>
</VirtualHost>
```

Next, we need make a file i.e., example. In this file we will store all the credentials of the form we will be filling. In this case we have example file that is in /etc/apache2/FormFiller/example.

It will look as follows:

```
jQueryUrl => 'http://ajax.googleapis.com/ajax/libs/jquery/1.3.1/jquery.min.js',
submit => 'true',
publicFormData => {
   email => ‘“XXXXXXX”’,
   pass  => '"XXXXXX"',
},
secretFormData => {
   pass => '"secret"',
},
```

After the configurations have been made we need to restart our apache server and hit http://--ip--
The form will be filled automatically.