The WebSocket protocol provides a way of creating web applications that support real-time bidirectional communication between clients and servers. Part of HTML5, WebSocket makes it much easier to develop these types of applications than the methods previously available.

Reverse Proxy Websockets with Nginx

For enterprise production use, where multiple WebSocket servers are needed for performance and high availability, a load balancing layer that understands the WebSocket protocol is required, and NGINX has supported WebSocket since version 1.3 and can act as a reverse proxy and do load balancing of WebSocket applications.

The WebSocket protocol is different from the HTTP protocol, but the WebSocket handshake is compatible with HTTP, using the HTTP Upgrade facility to upgrade the connection from HTTP to WebSocket. This allows WebSocket applications to more easily fit into existing infrastructures. For example, WebSocket applications can use the standard HTTP ports 80 and 443, thus allowing the use of existing firewall rules.

A WebSocket application keeps a long-running connection open between the client and the server, facilitating the **development of real-time applications**.

The HTTP Upgrade mechanism used to upgrade the connection from HTTP to WebSocket **uses the Upgrade and Connection headers**.

There are some challenges that a reverse proxy server faces in supporting WebSocket. One is that WebSocket is a hop-by-hop protocol, so when a proxy server intercepts an Upgrade request from a client it needs to send its own Upgrade request to the backend server, including the appropriate headers. Also, since WebSocket connections are long lived, as opposed to the typical short-lived connections used by HTTP, the reverse proxy needs to allow these connections to remain open, rather than closing them because they seem to be idle.

NGINX supports WebSocket by allowing a tunnel to be set up between a client and a backend server. For NGINX to send the Upgrade request from the client to the backend server, the Upgrade and Connection headers must be set explicitly, as in this example:

```
server {
        listen 8020;
        location / {
            proxy_pass http://websocket;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;
            proxy_set_header Host $host;
        }
    }
```

Reverse Proxy Websockets with Apache

WebSockets were introduced to open two-way interactive communication sessions, between a client and a server. This paved the way for event-driven responses, such as notifying a user of new content without refreshing the page.

In order to enable WebSocket reverse proxying, the Apache modules for handling such requests must be enabled.

● Proxy	:- a2enmod proxy
● Proxy_http	:- a2enmod proxy_http
● Proxy_wstunnel	:- a2enmod proxy_wstunnel

Note :- these commands are used to enable the respective modules.

- a2enmod proxy
- a2enmod proxy_http
- a2enmod proxy_wstunnel

```
<VirtualHost *:443>
  ServerName ws.serverlab.ca

  RewriteEngine on
  RewriteCond ${HTTP:Upgrade} websocket [NC]
  RewriteCond ${HTTP:Connection} upgrade [NC]
  RewriteRule .* "wss:/localhost:3000/$1" [P,L]
  <Proxy balancer://backend-cluster>
    BalancerMember http://server01:3000
    BalancerMember http://server02:3000
    BalancerMember http://server03:3000
  </Proxy>

  ProxyPass / balancer://backend-cluster/
  ProxyPassReverse / balancer://backend-cluster/
  ProxyRequests off
</VirtualHost>
```

ServerName ws.serverlab.ca
The hostname of the virtual web host that will handle the WebSocket connections.

RewriteEngine on
Used to set the status of the RewriteEngine to either on or off. To support WebSockets it must be turned on.

RewriteCond ${HTTP:Upgrade} websocket [NC]
A condition that must be matched in order for a request to be processed by the RewriteRule.

RewriteCond ${HTTP:Connection} upgrade [NC]
To something

RewriteRule .* "wss:/ws-backend%{REQUEST_URI}" [P]
Rewrite all incoming requests to use the wss protocol, and replace the destination hostname to that of a backend service.

]

```
<VirtualHost *:443>
      ServerName apache.fantasywl.in

         ProxyPass "/" "http://localhost:8080/"
         ProxyPassReverse "/" "http://localhost:8080/"

         ProxyPass "/" "ws://localhost:8080/"
         ProxyPassReverse "/" "ws://localhost:8080/"


         RewriteCond ${HTTP:Upgrade} websocket [NC]
         RewriteCond ${HTTP:Connection} upgrade [NC]
         RewriteRule .* "ws:/localhost:8080/$1" [P,L]

         ProxyPass / http://localhost:8080/
         ProxyPassReverse / http://localhost:8080/
         ProxyRequests on


       ErrorLog ${APACHE_LOG_DIR}/demosocket_error.log
       CustomLog ${APACHE_LOG_DIR}/demosocket_access.log combined


Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateFile /etc/letsencrypt/live/apache.fantasywl.in/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/apache.fantasywl.in/privkey.pem
</VirtualHost>
</IfModule>
```

Final configuration Apache:-

```
<VirtualHost *:80>
            ServerName apache.fantasywl.in

       RewriteEngine on
       RewriteCond ${HTTP:Upgrade} websocket [NC]
       RewriteCond ${HTTP:Connection} upgrade [NC]
       RewriteRule .* "ws:/localhost:8080/$1" [P,L]

       ProxyPass / http://localhost:443/
       ProxyPassReverse / http://localhost:443/
       #ProxyRequests on


         RewriteCond %{HTTP:Upgrade} =websocket [NC]
       RewriteRule /(.*)           ws://localhost:8080/$1 [P,L]
       RewriteCond %{HTTP:Upgrade} !=websocket [NC]
       RewriteRule /(.*)           http://localhost:8080/$1 [P,L]


       ProxyPass "/" "http://localhost:8080/"
       ProxyPassReverse "/" "http://localhost:8080/"

       ProxyPass "/" "ws://localhost:8080/"
       ProxyPassReverse "/" "ws://localhost:8080/"


       RewriteCond ${HTTP:Upgrade} websocket [NC]
       RewriteCond ${HTTP:Connection} upgrade [NC]
       RewriteRule .* "ws:/localhost:8080/$1" [P,L]

       ErrorLog ${APACHE_LOG_DIR}/demosocket_error.log
       CustomLog ${APACHE_LOG_DIR}/demosocket_access.log combined


</VirtualHost>
```

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
        ServerName apache.fantasywl.in

        RewriteEngine on

        RewriteCond %{HTTP:Upgrade} =websocket [NC]
        RewriteRule /(.*)           ws://localhost:8080/$1 [P,L]
        RewriteCond %{HTTP:Upgrade} !=websocket [NC]
        RewriteRule /(.*)           http://localhost:8080/$1 [P,L]


        ProxyPass "/" "http://localhost:8080/"
        ProxyPassReverse "/" "http://localhost:8080/"

        ProxyPass "/" "ws://localhost:8080/"
        ProxyPassReverse "/" "ws://localhost:8080/"


        RewriteCond ${HTTP:Upgrade} websocket [NC]
        RewriteCond ${HTTP:Connection} upgrade [NC]
        RewriteRule .* "ws:/localhost:8080/$1" [P,L]

        ProxyPass / http://localhost:8080/
        ProxyPassReverse / http://localhost:8080/
        ProxyRequests on


       ErrorLog ${APACHE_LOG_DIR}/demosocket_error.log
       CustomLog ${APACHE_LOG_DIR}/demosocket_access.log combined

Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateFile /etc/letsencrypt/live/apache.fantasywl.in/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/apache.fantasywl.in/privkey.pem
</VirtualHost>
</IfModule>
```