Deployment Guide

Load Balancing

12/14/2025

Load Balancing

Contents

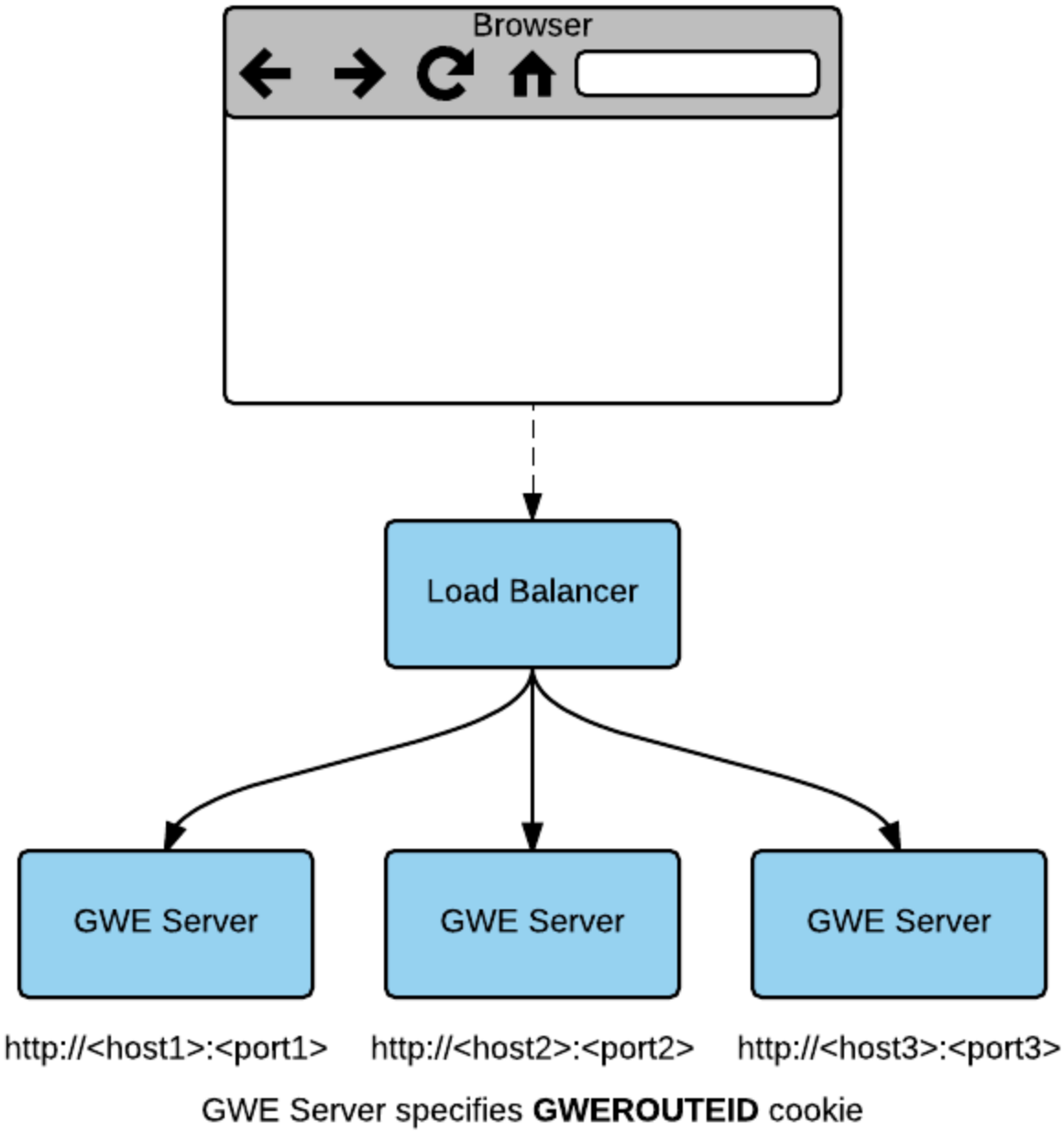
Genesys Web Engagement supports any third-party load balancer as long as the load balancing features include cookie support.

The following points are important for you to consider when setting up load balancing:

- Due to Safari's strict cookie policy, Genesys recommends that your load balancer is hosted under the same domain as the website (or its subdomain). Otherwise, chat "stickiness" cookies might be rejected as "third-party", and the solution will not work (users won't be able to start chat).
- Apache does not support WebSockets load balancing by default. If you want to enable this option, you must use the mod_proxy_wstunnel module. **Note:** This module requires Apache version 2.4+ and is only available for Linux.
- If your load balancer does not support WebSockets, make sure that you disable WebSockets on the client side. See Chat Application - disableWebSockets and Tracker Application - disableWebSockets for details. You can also control the usage of WebSockets in CometD on the server side. See the transports option for details.

Architecture

The following diagram shows how you can implement a load balancing configuration for your Web Engagement servers.



Sample of Deployment for Load Balancing

In the above example, the load balancer implements sticky sessions to GWE Servers based on the **GWEROUTEID** cookie. The GWE Server is responsible for specifying this cookie.

In the Web Engagement 8.5 architecture, the load balancer can be associated with the Web Engagement cluster application. Usually, this means that the host and the default port specified in the cluster application should correspond to the host and port of the load balancer. For more information about the cluster application see the section on Creating the Cluster Application in Installing the Genesys Web Engagement Server.

Sticky Sessions

The load balancer implements sticky sessions to GWE Servers based on the **GWEROUTID** cookie specified by the GWE Server instance. Therefore, it must support the following feature:

- Cookie-based stickiness to enable engagement.

> **Important**
> The GWEROUTEID cookies are created by the Genesys Web Engagement servers.

GWE requires sticky sessions not only for performance reasons, but also to switch over ongoing transactions if a node fails.

Cookie Name	Cookie Value
GWEROUTEID	"." + *<GWE Server application name in Genesys Administrator>*

Sample Configurations

Genesys provides sample load balancing configurations for two common load balancers: Apache and Nginx. For details, select a tab below:

Apache

The following procedures provide a sample load balancing configuration for Apache. Before you begin, make sure you have completed the following prerequisites:

- You already deployed your Web Engagement application into a production (or production-like) environment and have at least two nodes configured to work in the cluster (see Installing the Genesys Web Engagement Server for details).
- For this configuration example, you installed and configured an instance of Apache, version 2.2.

Configuring the Apache Load Balancer

Start

1. Confirm that the following modules are present in your Apache load balancer:
   - mod_proxy.so
   - mod_proxy_balancer.so

- mod_proxy_connect.so
- mod_proxy_http.so
- mod_headers.so

2. Edit the **./conf/httpd.conf** file and confirm that the modules are loaded:
   - LoadModule proxy_module modules/mod_proxy.so
   - LoadModule proxy_module modules/mod_proxy_balancer.so
   - LoadModule proxy_module modules/mod_proxy_connect.so
   - LoadModule proxy_module modules/mod_proxy_http.so
   - LoadModule proxy_module modules/mod_headers.so
3. Add the following configuration script to the end of the **httpd.conf** file:

```
<VirtualHost *:80>
ProxyRequests Off
<Proxy *>
order allow,deny
Allow from All
</Proxy>
ProxyPass /server http://<GWE_cluster_app_host_IP_or_FQDN>:<GWE_cluster_app_port>/server
ProxyPassReverse /server
http://<GWE_cluster_app_host_IP_or_FQDN>:<GWE_cluster_app_port>/server
</VirtualHost>
Listen <GWE_cluster_app_port>
<VirtualHost *:<GWE_cluster_app_port>>
<Proxy balancer://cluster>
   #BalancerMember route parameter is the same as name of GWE Server node application in
the Genesys Configuration Layer, for example GWE_Node_1
   BalancerMember http://<GWE_node1_app_host_IP_or_FQDN>:<GWE_node1_app_port>/server
route=GWE_Node_1
   BalancerMember http://<GWE_nodeN_app_host_IP_or_FQDN>:<GWE_nodeM_app_port>/server
route=GWE_Node_N
ProxySet stickysession=GWEROUTEID
</Proxy>
ProxyPass /server balancer://cluster
<Location /balancer-manager>
SetHandler balancer-manager
Order Deny,Allow
Allow from all
</Location>
</VirtualHost>
```

4. Save your changes. The load balancer is now configured.
5. Your cluster is healthy if the load balancer receives a successful response on requests to
   - http(s)://*Web Engagement Server Host*:*Web Engagement Server Port (secured port for https)*/server/about

   or

   - http(s)://*Web Engagement Server Host*:*Web Engagement Server Port (secured port for https)*/server/isAlive

End

Nginx

The following procedures provide a sample load balancing configuration for Nginx. Before you begin, make sure you have completed the following prerequisites:

- You already deployed your Web Engagement application into a production (or production-like) environment and have at least two nodes configured to work in the cluster (see Installing the Genesys Web Engagement Server for details).
- For this configuration sample, you installed and configured an instance of Nginx.

To configure your Nginx load balancer, edit the **./conf/nginx.conf** file and modify the configuration according to the samples provided below. For details about the configuration, consult the Nginx documentation.

Configuration Sample: Nginx Load Balancer

```
events {
            worker_connections 1024;
}
http {
        include mime.types;
         default_type application/octet-stream;
        map_hash_bucket_size 64;

        log_format main 'balancing_cookie: $cookie_GWEROUTEID --> $remote_addr - $remote_user
[$time_local] "$request" ';

        access_log logs/nginx_access.log main;
        error_log logs/nginx_error.log debug;

# Select node on top of existing cookie GWEROUTEID
map $cookie_GWEROUTEID $http_sticky {
        .GWE_Node_1 192.168.1.1:9081; # GWE_Node_1 is running on 192.168.1.1:9081
        .GWE_Node_2 192.168.1.2:9081; # GWE_Node_2 is running on 192.168.1.2:9081
 }

# Select node (round-robin) if cookie GQWEROUTEID is absent
upstream http_cluster {
        server 192.168.1.1:9081 fail_timeout=30s; # GWE_Node_1 is running on 192.168.1.1:9081
        server 192.168.1.2:9081 fail_timeout=30s; # GWE_Node_2 is running on 192.168.1.2:9081
 }

 map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
 }
 server {
        listen <GWE_cluster_app_port>;

        location @fallback {
               proxy_pass http://http_cluster;
        }

        location /server {
               # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
```

```
                proxy_http_version 1.1;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $connection_upgrade;
                # Increase buffer sizes to find room for DOM and CSS messages
                proxy_buffers 8 2m;
                proxy_buffer_size 10m;
                proxy_busy_buffers_size 10m;
                proxy_connect_timeout 5s;
                # Fall back if server responds incorrectly
                error_page 502 = @fallback;
                # or if doesn't respond at all.
                error_page 504 = @fallback;
                # Create a map of choices
                # see https://gist.github.com/jrom/1760790
                if ($scheme = ‘http’) {
                    set $ftest HTTP;
                 }
                if ($http_sticky) {
                    #echo 'HTTP-STICKY scheme';
                    set $ftest "${ftest}-STICKY";
                }
                if ($ftest = HTTP-STICKY) {
                    #echo 'Pass to stickyness ';
                    proxy_pass http://$http_sticky$uri?$args;
                    break;
                }
                if ($ftest = HTTP) {
                    proxy_pass http://http_cluster;
                    break;
                }
                return 500 "Misconfiguration";
        }
    }
}
```