Deployment Guide

Configuring a Load Balancer for Co-browse Cluster

12/16/2025

Contents

Configuring a Load Balancer for Co-browse Cluster

Load Balancer Requirements

When configuring a load balancer, note the following requirements:

- You must use a third-party HTTP load balancer. Genesys cannot provide or validate a third-party load balancer.
- The load balancer must support health check monitoring of each node. A failed Co-browse node cannot recover gracefully. The load balancer must detect node failure to notify the client and allow a manual restart of the session.
- The load balancer must also support cookie based session stickiness. Genesys components add the `gcbSessionServer` cookie to HTTP requests and you should configure the load balancer to distribute requests to the appropriate Co-browse node based on the cookie value.
- We highly recommend WebSocket support. See also, Sizing.
- If WebSocket support is enabled, the load balancer must be able to balance HTTP requests and WebSocket connections at the same time to properly handle scenarios where the end user's browser or your infrastructure does not support WebSockets.
- SSL decryption—Co-browse relies on application-generated cookie headers and the load balancer must have access to HTTP headers. If incoming traffic uses HTTPS, the load balancer must be able to decrypt HTTPS traffic and access the cookie headers used for session stickiness. The resulting traffic going from the load balancer to the Co-browse server can be re-encrypted (SSL Bridging/Re-encryption) or remain in HTTP. Keeping the traffic in HTTP reduces the load on the Co-browse server (SSL Offloading).

WebSocket Support

To achieve the best performance with Co-browse, Genesys highly recommends you configure WebSocket support for your load balancer. WebSockets improve performance and considerably reduce the request throughput rate of each session. If WebSockets are unavailable, Co-browse still functions but uses other transports that perform significantly slower. In some cases, WebSockets become mandatory to ensure proper order of DOM change events as Co-browse server does not provide native ordering of DOM change events.

If your load balancer does not support WebSockets and you do not want to wait for Co-browse to automatically switch to another transport, you can use the `disableWebSockets` options for the customer side and the agent side. For more information, see JavaScript Configuration API and disablewebsockets configuration.

Load Balancer Configuration

Your load balancer configuration will depend upon which load balancer you implement. See the examples below for sample configurations of Nginx and Apache.

All proposed examples assume cookie-based stickiness. If you use URL-based stickiness and the actual nodes are not publicly accessible, you may want to add logic to route publicly accessible URLs of Co-browse nodes (such as `http://<load-balancer>?co-browse-node-id={node-id}`) to the actual nodes. However, such configuration is beyond the scope of this Guide. For the details about cookie-based and URL-based stickiness supported by Co-browse Solution, see Stickiness.

> ### Important
> Due to browsers' strict cookie policies, Genesys highly recommends that you host the Load Balancer on the same domain as the website or on one of its sub-domains. Otherwise, chat and Co-browse stickiness cookies may be rejected as being from third parties and the solution will not work. Users will not be able to start chat nor begin co-browsing.

High Availability and Health Checks

Currently, there is no fail-over support for Co-browse sessions. If a Co-browse Server node becomes inaccessible, Co-browse live sessions hosted by this server terminate. To notify clients (agent desktop and end user's browser application) that a session has ended you must implement healthchecks/ fallback functionality in your Load Balancer. You must configure your LB to route all requests that go to a failed node to another node. This node will detect it does not *own* the Co-browse sessions and terminate them, sending notifications to the clients. After sessions are terminated, agents and customers can manually establish new Co-browse sessions.

> ### Important
> Built-in chat *does* support fail-over. If you use built-in chat, chat session stay active and agents join new Co-browse sessions automatically.

> ### Important
> You can use the /cobrowse/health HTTP resource for health checks.

Nginx Configuration Samples

Below are two sample configurations for load balancing with Nginx:

- The first sample keeps connections secure with HTTPS both **from browsers to load balancer** and **from load balancer to servers**.
- The second sample uses the SSL Acceleration technique, where HTTPS is used only from the browsers to the load balancers; plain HTTP is used from the load balancer to the Co-browse servers.

> **Important**
>
> These configurations are intended to be examples and might not represent best practices for Nginx configuration.

> **Important**
>
> These configurations use a five second timeout for High Availability, if a server dies, the load balancer switches the client to another server after five seconds. In production, you can eliminate this timeout by using "health checks" functionality, available in Nginx PLUS or through third-party plug-ins. See the following links for more information:
>
> - http://nginx.com/products/application-health-checks/
> - http://wiki.nginx.org/NginxHttpHealthcheckModule
> - https://github.com/cep21/healthcheck_nginx_upstreams
> - https://github.com/yaoweibin/nginx_upstream_check_module

Sample One for Nginx

```
# Basic configuration for load balancing 2 or more Co-Browse servers.
# All nodes are listed 2 times: in upstream and map directives.
# Co-browse applications are responsible for setting the "gcbSessionServer" cookie
# with one of the values listed in map directive. These values are names of
# applications in config server.
# This (default) variant uses HTTPS (if browser request is HTTPS) for connections
# both from browser to load balancer and from load balancer to Co-Browse servers.
# For another version with HTTPS only from browser to LB, see nginxSSLAccelerated.conf

# IMPORTANT!
# This configuration is not intended for production use!
# It is mere example of how this functionality can be achieved.

events {
    worker_connections  1024;
}
```

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    # to handle longer names of Co-browse server applications
    map_hash_bucket_size 64;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for" "$upstream_addr"';

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

    upstream http_cobrowse_cluster {
        server 192.168.73.210:8700 fail_timeout=5s;
        server 192.168.73.210:8701 fail_timeout=5s;
    }
    upstream https_cobrowse_cluster {
        server 192.168.73.210:8743 fail_timeout=5s;
        server 192.168.73.210:8744 fail_timeout=5s;
    }

    map $cookie_gcbSessionServer $http_sticky_backend {
        default 0;
        .CB_Server_1   192.168.73.210:8700;
        .CB_Server_2   192.168.73.210:8701;
    }
    map $cookie_gcbSessionServer $https_sticky_backend {
        default 0;
        .CB_Server_1   192.168.73.210:8743;
        .CB_Server_2   192.168.73.210:8744;
    }

    map $http_upgrade $connection_upgrade {
        default upgrade;
        ''      close;
    }

    server {
        listen 8080;
        listen 8083 ssl;
        ssl_certificate cobrowse.unsigned.crt;
        ssl_certificate_key cobrowse.unsigned.key;

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

        location /cobrowse {
            # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
            proxy_http_version 1.1;
            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;

            # If Co-browse server doesn't respond in 5 seconds, consider it dead
            # (a 504 will fire and be caught by error_page directive for fallback).
            # This timeout can be eliminated using "health checks" functionality
            # available in Nginx PLUS or via 3rd party plugins. See the following links:
```

```
            # http://nginx.com/products/application-health-checks/
            # http://wiki.nginx.org/NginxHttpHealthcheckModule
            # https://github.com/cep21/healthcheck_nginx_upstreams
            # https://github.com/yaoweibin/nginx_upstream_check_module
            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 $test HTTP;
            }
            if ($scheme = 'https') {
                set $test HTTPS;
            }
            if ($http_sticky_backend) {
                set $test "${test}-STICKY";
            }

            if ($test = HTTP-STICKY) {
                proxy_pass http://$http_sticky_backend$uri?$args;
                break;
            }
            if ($test = HTTPS-STICKY) {
                proxy_pass https://$https_sticky_backend$uri?$args;
                break;
            }
            if ($test = HTTP) {
                proxy_pass http://http_cobrowse_cluster;
                break;
            }
            if ($test = HTTPS) {
                proxy_pass https://https_cobrowse_cluster;
                break;
            }



            return 500 "Misconfiguration";
        }

    }

}
```

Sample Two for Nginx

```
# Basic configuration for load balancing 2 or more Co-browser servers.
# Nodes are listed 2 times: in upstream and map directives.
# Co-browse applications are responsible for setting the "gcbSessionServer" cookie
# with one of the values listed in map directive. These values are names of
# applications in config server.
# Note that this version uses "SSL acceleration" (http://en.wikipedia.org/wiki/
SSL_Acceleration,
# http://en.wikipedia.org/wiki/Load_balancing_(computing)#Load_balancer_features):
# load balancer terminated SSL connections, passing HTTPS requests as HTTP to the servers.

# IMPORTANT!
```

```
# This configuration is not intended for production use!
# It is mere example of how this functionality can be achieved.

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

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

    upstream http_cobrowse_cluster {
        server 192.168.73.210:8700 fail_timeout=5s;
        server 192.168.73.210:8701 fail_timeout=5s;
    }

    map $cookie_gcbSessionServer $sticky_backend {
        default 0;
        .CB_Server_1   192.168.73.210:8700;
        .CB_Server_2   192.168.73.210:8701;
    }

    map $http_upgrade $connection_upgrade {
        default upgrade;
        ''      close;
    }

    server {
        listen 8080;
        listen 8083 ssl;
        ssl_certificate cobrowse.unsigned.crt;
        ssl_certificate_key cobrowse.unsigned.key;

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

        location /cobrowse {
            # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
            proxy_http_version 1.1;
            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;

            # If Co-browse server doesn't respond in 5 seconds, consider it dead
            # (a 504 will fire and be caught by error_page directive for fallback)
            # This timeout can be eliminated using "health checks" functionality
            # available in Nginx PLUS or via 3rd party plugins. See the following links:
            # http://nginx.com/products/application-health-checks/
            # http://wiki.nginx.org/NginxHttpHealthcheckModule
            # https://github.com/cep21/healthcheck_nginx_upstreams
            # https://github.com/yaoweibin/nginx_upstream_check_module
```

```
            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;

            if ($sticky_backend) {
                proxy_pass http://$sticky_backend$uri?$args;
            }
            proxy_pass http://http_cobrowse_cluster;
        }

    }

}
```

Apache Configuration Samples

Below are two sample configurations for load balancing with Apache (both without WebSockets support):

- The first sample uses an insecure connection with HTTP both from browsers to the load balancer and from the load balancer to the servers.
- The second sample uses the SSL Acceleration technique, where HTTPS is used only from the browsers to the load balancer. Plain HTTP is used from the load balancer to the Co-browse servers.

> **Important**
>
> These configurations are intended to be examples and might not represent best practices for Apache configuration.

Prerequisites for both samples

- If you are using a proxy to inject the instrumentation snippet into your site, you must exclude the load balancer host from proxying. Otherwise Apache configuration will work incorrectly in some cases such as when IE9 is a customer browser.
- Disable WebSockets for the customer side and the agent side. For more information, see JavaScript Configuration API#disableWebSockets and disableWebSockets configuration.

Sample One for Apache

```
Listen APACHE_PORT_1

#Load Balancer of Co-browse Servers
<VirtualHost *:APACHE_PORT_1>
    ProxyRequests Off
    <Proxy balancer://CLUSTER_NAME>
```

```
        BalancerMember http://<HOST_1>:<PORT_1>/cobrowse route=<CO-BROWSE_SERVER_1_APP_NAME>
        BalancerMember http://<HOST_2>:<PORT_2>/cobrowse route=<CO-BROWSE_SERVER_2_APP_NAME>
        BalancerMember http://<HOST_3>:<PORT_3>/cobrowse route=<CO-BROWSE_SERVER_3_APP_NAME>
        ProxySet stickysession=gcbSessionServer
    </Proxy>
    ProxyPass /cobrowse balancer://CLUSTER_NAME
</VirtualHost>
```

Sample Two for Apache

```
Listen 8090
Listen 8093

#Load Balancer of Co-browse Servers
ProxyRequests Off
<Proxy balancer://cluster_cobrowse>
    BalancerMember http://co-browse_host_1:8700/cobrowse route=CB_Server_1
    BalancerMember http://co-browse_host_2:8700/cobrowse route=CB_Server_2
    BalancerMember http://co-browse_host_3:8700/cobrowse route=CB_Server_3
    ProxySet stickysession=gcbSessionServer
</Proxy>
ProxyPass /cobrowse balancer://cluster_cobrowse

NameVirtualHost *:8090
NameVirtualHost *:8093

<VirtualHost *:8090>
    ServerName apache_server_name
</VirtualHost>

<VirtualHost *:8093>
    ServerName apache_server_name

    SSLEngine on
    SSLCertificateFile "cert.crt"
    SSLCertificateKeyFile "priv_key_pkcs8.pem"
    SSLCACertificateFile "CA.crt"
</VirtualHost>
```