Running with a Proxy

Overview

Posit Workbench runs a fully managed Nginx HTTP server that can be customized for SSL, security and more. It is configured from settings in rserver.conf (e.g. ssl-enabled, www-address) and can be customized with additional Nginx directives. See Customizing default proxy for more info on customizing the built-in proxy server.

Even with a built-in Nginx server, it's common for customers to also use an external proxy server that runs in front of Posit Workbench for security, to bridge firewalls, to leverage an existing IT proxy server, or as a front-end load balancer when using load balancing. This section describes how to configure an external proxy server to proxy traffic to and from RStudio or Posit Workbench.

If you are installing a new Apache or Nginx proxy server, see the complete examples for Nginx configuration and Apache configuration for a quicker start. These step-by-step instructions allow you to customize those examples or update an existing proxy server's configuration.

1. Determine server addresses
2. Add a proxy pass directive
3. Set HTTP Headers
4. Forwarding websockets
5. Check the connection timeout

Determine server addresses

Each Posit Workbench server instance must listening on an ip-address that the proxy server can reach (see `www-address` in rserver.conf). The default `www-port` is 8787 unless `ssl-enabled=1` in which case it is 443, and the protocol the proxy should use for each server address will be https.

If not load balancing, the proxy server may be running on the same node as Posit Workbench. In this case, use localhost as the server hostname, e.g.: `http://localhost:8787`.

If load balancing, the proxy configuration requires a list of Posit Workbench server addresses, e.g. `http://node1:8787`, `http://node2:8787`, etc.

Add a proxy pass directive

The proxy pass directive controls which URLs are forwarded to Workbench. If the proxy server is shared with other applications, choose a path prefix that identifies Workbench requests, e.g., `/rstudio`. Otherwise, for a dedicated proxy server, send all requests to Workbench by specifying a location of `/`. The proxy pass directive requires the host name or ip address of Workbench. In the examples below, replace localhost with Workbench's host name or ip address when the proxy server is not local. The proxy pass directive also requires a protocol in the target URL. Use https when Workbench itself has SSL enabled (with the setting ssl-enabled=1 in rserver.conf), otherwise use http.

When load balancing, the proxy pass refers to a cluster address used when defining the cluster of Posit Workbench servers.

Here's the proxy_pass directive for Nginx with the /rstudio prefix, ssl-enabled=0, and no load balancing:

```
location /rstudio {
    # Removes /rstudio from the URL
    rewrite ^/rstudio(.*)$ /$1 break;
    proxy_pass http://localhost:8787;
    # other directives go here
}
```

or for Nginx with no prefix:

```
location / {
    proxy_pass http://localhost:8787;
    # other directives go here
}
```

For Apache with the /rstudio prefix:

```
RewriteRule /rstudio/(.*)     http://localhost:8787/$1 [P,L]
ProxyPass /rstudio/ http://localhost:8787/
```

or for Apache with no prefix:

```
ProxyPass / http://localhost:8787/
```

Set HTTP headers

To properly generate externally resolvable redirect URLs and cookie domain paths, Workbench needs the external proxy server's host name, port, protocol and optional path prefix. These should be similar to those used in a location redirect for HTTP.

Hostname and port

By default, Workbench obtains the hostname and port from the standard HTTP Host header. This value is set by the browser and forwarded to Workbench but it's important to set it in the proxy configuration as well. Those headers are optional in HTTP and resetting them prevents tampering from the browser, and prevents the proxy server from rewriting the Host header to the internal hostname. It's usually possible to add a `Set-Header` directive to the proxy server configuration that sets Host to the correct external hostname. Add the `:serverport` suffix if you are not using default ports for `http`/`https`.

Server protocol

If the proxy server is configured for SSL and Workbench is not the `X-Forwarded-Proto` header should be set to `https`.

This will prevent Workbench from changing the protocol of generated URLs from `https` to `http` when it performs a redirect. For example, if the header is not set the login page will redirect back to `http://servername` instead of `https://servername`.

Path prefix

If you configured the ProxyPass directive with a path prefix, configure the `www-root-path` attribute in `/etc/rstudio/rserver.conf` so that Workbench knows to add this path prefix to any external URLs it generates.

Listing 1 `/etc/rstudio/rserver.conf`

```
www-root-path=/rstudio
```

Alternatively, to configure the path-prefix entirely in the proxy server's configuration, add a proxy set header directive to set the HTTP header `X-RStudio-Root-Path` with the value of your path prefix - e.g. `/rstudio`.

> **Note**
>
> The header `X-RStudio-Root-Path` and the configuration option `www-root-path` serve the same purpose. If either is set Workbench will always return cookies and redirects for the correct path, without requiring rewrite assistance from the proxy. The header value has precedence over the configuration value.

Alternate headers

Workbench supports other HTTP headers as alternative ways to inform it of the hostname, server port, protocol, and path prefix:

To set all of the values with one header, use:

```
X-RStudio-Request  protocol://server-name:server-port/path-prefix
```

For example:

```
X-RStudio-Request  https://www.example.com:8787/rstudio
```

Just as with `X-Forwarded-Proto`, if your proxy server already sets the headers `X-Forwarded-Host` and `X-Forwarded-Port` they can be used to ensure Workbench redirects to the correct hostname and port, respectively.

Similarly Workbench supports the standard `Forwarded` header for retrieving the host= and proto= fields.

> **Note**
>
> When not using a custom path prefix, the hostname used by Workbench will always come from the Host header, not from X-Forwarded-Host, X-RStudio-Request, or Forwarded.

Forwarding websockets

Beyond the normal reverse proxy configuration you'd apply for any HTTP server application, you also need to ensure that websockets are forwarded correctly between the proxy server and Workbench to ensure that all Workbench functions work correctly. In particular, they're needed for Shiny applications and VS Code sessions.

The configuration to forward websockets varies for each proxy server. There are examples for enabling websockets in Nginx configuration and Apache configuration.

Check the connection timeout

It's also important to ensure that your reverse proxy allows long-standing connections or uses a relatively lenient connection timeout; we recommend at least 60 seconds. Several components of Workbench use HTTP Long Polling to push information to the browser; a connection timeout of under 50 seconds will result in HTTP 504 (gateway timeout) errors from the reverse proxy.

Shiny applications will close abruptly if the websocket is timed out by a proxy server.

Starting a VS Code session will show a blank page if the websocket connection is not fowarded.

Nginx configuration

On Debian or Ubuntu a version of Nginx that supports reverse-proxying can be installed using the following command:

```
sudo apt-get install nginx
```

On CentOS or Red Hat you can install Nginx using the following command:

```
sudo yum install nginx
```

To enable an instance of Nginx running on the same server to act as a front-end proxy to Workbench you would add commands like the following to your `nginx.conf` file. These examples represent the customizable chunks but you should insert each chunk under http, server, and location to modify your nginx.conf file as necessary. Note that you must add the configuration below to proxy websockets in order to correctly display Shiny apps and R Markdown Shiny documents in Workbench. Also note that if you are proxying to a server on a different machine you need to replace references to `localhost` with the correct address of the server where you are hosting Workbench.

```
http {

  # Support proxying of web-socket connections
  map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
  }

  server {
```

```
    listen 80;

    location / {
      proxy_pass http://localhost:8787;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;

      # Set the Host header to match how users access your site. Omit :server_port if using t
      # so that this value will match the Origin: header for CORS (cross origin security vali
      proxy_set_header Host $host:$server_port;

      proxy_read_timeout 20d;
    }
  }
}
```

If you want to serve Workbench from a custom path (e.g. /rstudio) update your **nginx.conf** file with the directives shown below. Replace all occurrences of /rstudio with your chosen custom path prefix.

```
http {

  # Support proxying of web-socket connections
  map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
  }

  server {
    listen 80;

    location /rstudio/ {
      # Needed only for a custom path prefix of /rstudio
      rewrite ^/rstudio(.*)$ /$1 break;

      # Use http here when ssl-enabled=0 is set in rserver.conf
      proxy_pass http://localhost:8787;

      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
```

```
      proxy_read_timeout 20d;

      # Not needed if www-root-path is set in rserver.conf
      proxy_set_header X-RStudio-Root-Path /rstudio;

      # Set the Host header to match how users access your site. Omit :server_port if using t
      # so that this value will match the Origin: header for CORS (cross origin security vali
      proxy_set_header Host $host:$server_port;
    }
  }
}
```

If the Nginx proxy is using SSL and Workbench has ssl-enabled=0, use the directives from this example:

```
http {
    # Support proxying of web-socket connections
    map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
    }

    server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name localhost;

        # Change to point to your certs
        ssl_certificate /etc/ssl/certs/localhost.crt;
        ssl_certificate_key /etc/ssl/private/localhost.key;

        ssl_protocols TLSv1.2 TLSv1.1 TLSv1;

        location /rstudio {
           # Needed only for prefix of /rstudio
           rewrite ^/rstudio(.*)$ /$1 break;

           # Use http here when ssl-enabled=0 is set in rserver.conf
           proxy_pass http://localhost:8787;
           proxy_http_version 1.1;
```

```
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;

            # Set the Host header to match how users access your site. Omit :server_port if us
            # so that this value will match the Origin: header for CORS (cross origin securit
            proxy_set_header Host $host:$server_port;

            # Not needed if there's no prefix or www-root-path is set in rserver.conf
            proxy_set_header X-RStudio-Root-Path /rstudio;

            # Tells Workbench to send redirect URLs back to https
            proxy_set_header X-Forwarded-Proto https;

            proxy_read_timeout 20d;
        }
    }
}
```

For a load balanced server with SSL enabled and where rserver.conf has `ssl-enabled=1` as well:

```
events {}

http {
   # Log format to include the proxy server address and request processing time
   log_format proxy_info '$remote_addr - $remote_user [$time_local] '
                         '"$request" $status $body_bytes_sent '
                         '$request_time $upstream_addr '
                         '"$http_referer" "$http_user_agent"';

   # Upgrade websockets
   map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
   }

   # Define Posit Workbench cluster - change node1/node2 to your server's ip address or hostn
   upstream rservercluster {
      # other options: hash <var> consistent or omit any mechanism for round-robin
      ip_hash;
      # Using 443 here because of rserver.conf when `ssl-enabled=1` otherwise 8787 or `www-po
      server node1:443;
```

```
    server node2:443;
}
server {
    access_log /var/log/rstudio/rstudio-server/nginx-proxy-access.log proxy_info;
    error_log /var/log/rstudio/rstudio-server/nginx-proxy-error.log warn;
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  example.com
    root         /usr/share/nginx/html;

    # change to paths to your cert/key
    ssl_certificate "/etc/rstudio/rserver.crt";
    ssl_certificate_key "/etc/rstudio/rserver.key";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    rewrite ^$ $scheme://$http_host/ permanent;

    location / {
        rewrite ^/(.*)$ /$1 break;
        proxy_pass https://rservercluster;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;

        # When not using the default port, include :$server_port below to match what you wa
        # users to see for the hostname in the browser.
        proxy_set_header Host $host;

        proxy_read_timeout 20d;
        # Older, more coomplicated ways to configure forwarding of headers
        #proxy_set_header X-RStudio-Request $scheme://$host:$server_port$request_uri;
        #proxy_set_header X-RStudio-Root-Path /rstudio;
        # OR let the proxy rewrite the root path
        #proxy_redirect http://localhost:8787/ $scheme://$host/rstudio/;
        #proxy_cookie_path / /rstudio;
        # OR existing X-Forwarded headers
        #proxy_set_header X-Forwarded-Host $host;
        #proxy_set_header X-Forwarded-Proto $scheme;
```

```
        # OR alternatively the Forwarded header (just an example)
        #proxy_set_header Forwarded "host=$host:$server_port;proto=$scheme;";
      }
   }
}
```

After adding these entries you'll then need to restart Nginx so that the proxy settings take effect:

```
sudo /etc/init.d/nginx restart
```

or

```
sudo systemctl restart nginx
```

In some cases, such as when streaming job statuses from the launcher, the default response buffering in nginx can be too slow for delivering real-time updates, especially when configured to use SSL. If job output streams are not working properly from the home page, we recommend disabling response buffering by adding the following line under the `server` directive:

```
server {
  # ... follows previous configuration
  proxy_buffering off;
}
```

Apache configuration

The steps to configure an apache proxy configuration are as follows:

1. Enable the proxy module
2. Handle the Host header
3. Increase max connections
4. Define load balanced cluster (when using Posit Workbench with load balancing)
5. Proxy to RStudio/Posit Workbench

Enable the proxy module

To enable an instance of Apache running on the same server to act as a front-end proxy to Workbench use the `mod_proxy` and `mod_proxy_wstunnel` modules. The steps for enabling this module vary across operating systems so you should consult your distribution's Apache documentation for details. Apache as reverse proxy already includes `X-Forwarded-Host` (with port) and `X-Forwarded-Proto` by default.

On Debian and Ubuntu systems Apache can be installed with `mod_proxy` using the following commands:

```
sudo apt-get install apache2
sudo apt-get install libapache2-mod-proxy-html
sudo apt-get install libxml2-dev
```

On Ubuntu/Debian environments, update the Apache configuration files to activate `mod_proxy` by running these commands:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod proxy_wstunnel
sudo a2enmod headers
sudo a2enmod rewrite
```

To enable support for SSL run:

```
sudo a2enmod ssl
```

On CentOS and RedHat systems Apache can be installed with `mod_proxy` and `mod_proxy_wstunnel` by following the instructions here:

http://httpd.apache.org/docs/2.4/platform/rpm.html

By default with Apache 2.4, `mod_proxy` and `mod_proxy_wstunnel` should be enabled. You can check this by opening the file `/etc/httpd/conf.modules.d/00-proxy.conf` and making sure the following lines are included and not commented out:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

Once you have enabled `mod_proxy` and `mod_proxy_wstunnel` in your Apache installation you need to add the required proxy commands to your `VirtualHost` definition. Note that you will also need to include code to correctly proxy websockets in order to correctly proxy Shiny apps and R Markdown documents within Workbench. Also note that if you are proxying to a server on a different machine you need to replace references to `localhost` with the correct address of the server where you are hosting Workbench.

Handle the Host header

For Apache, if you are not setting the Host header with the `RequestHeader set` directive, enable the option:

```
ProxyPreserveHost On
```

Workbench relies on the Host header matching what the browser sees. If ProxyPreserveHost is Off, the server and port are changed in the Host header and URLs will point to the wrong server and port.

Increase max connections

Increase the number of allowed proxy connections by adding the following settings:

```
<IfModule mpm_event_module>
    ServerLimit              50
    StartServers             10
    MinSpareThreads          75
    MaxSpareThreads          250
    ThreadLimit              64
    ThreadsPerChild          64
    MaxRequestWorkers        3200
    MaxConnectionsPerChild   10000
</IfModule>
```

Define load balanced cluster

When using load balancing with Apache, Posit Workbench requires two tag definitions, one for http and the other for websockets. Each Proxy tag contains a BalancerMember directive for each node in the cluster. The http Proxy tag also defines a strategy for balancing http requests across the cluster. Posit Workbench performs best with sticky session load balancing, where the example adds a cookie `PWB_ROUTEID` set to the value of the `route` attribute (`node1` or `node2`). After the first request, the cookie is set. The second request will try that node first.

Here's an example of the definitions for a cluster with two nodes:

```
Header add Set-Cookie "PWB_ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHA

<Proxy balancer://rserver-http>
   BalancerMember http://node1:8787 route=node1
```

```
   BalancerMember http://node2:8787 route=node2
   ProxySet stickysession=PWB_ROUTEID
</Proxy>

<Proxy balancer://rserver-ws>
   BalancerMember ws://node1:8787 route=node1
   BalancerMember ws://node2:8787 route=node2
   ProxySet stickysession=PWB_ROUTEID
</Proxy>
```

This replaces the tag in the basic examples to enable load balancing.

Proxy to RStudio/Posit Workbench

To configure Apache to proxy requests to RStudio/Workbench:

1. Define a VirtualHost that's listening on 80 or 443.
2. Add a definition (that includes a cluster definition if load balancing)
3. Add Rewrite rules for websockets,
4. Add ProxyPass and ProxyPassReverse directives to point to either an individual Posit Workbench node or a load balanced cluster.
5. Increase request timeout for long-lived connections that stream data.

Config examples

In the examples below, change `example.com:80` to the `hostname:port` users will use to access your site.

If not load balancing: 1. Change `localhost:8787` to the hostname and port of Posit Workbench as defined in `rserver.conf`. 2. Change `http` to `https` and `ws` to `wss` if `ssl-enabled=1` is set in rserver.conf

If load balancing, follow these steps and see SSL and load balancing: 1. Update the BalancerMember directives to use a protocol that matches Posit Workbench config in `rserver.conf`: http/ws by default or https if `ssl-enabled=1`. 2. Change `ws://localhost:8787` to `balancer://rserver-ws` and `http://localhost:8787` to `balancer://rserver-http`. 3. Change the tag as described in Define load balanced cluster.

Basic example

Here's a basic example proxy configuration without SSL, without path prefix and without load balancing:

```
ProxyPreserveHost On
<IfModule mpm_event_module>
    ServerLimit              50
    StartServers             10
    MinSpareThreads          75
    MaxSpareThreads          250
    ThreadLimit              64
    ThreadsPerChild          64
    MaxRequestWorkers        3200
    MaxConnectionsPerChild   10000
</IfModule>

<VirtualHost *:80>

  <Proxy *>
    Allow from localhost
  </Proxy>

  RewriteEngine on
  RewriteCond %{HTTP:Upgrade} websocket [NC]
  RewriteCond %{HTTP:Connection} upgrade [NC]
  RewriteRule ^/?(.*) "ws://localhost:8787/$1" [P,L]

  ProxyPass / http://localhost:8787/
  ProxyPassReverse / http://localhost:8787/
  ProxyRequests Off

  # Optional - workbench uses the Host header by default in redirects. With 'ProxyPreserveHos
  # the Host as set by the browser but this it may be slightly more secure to set it explici
  # will be used for redirect URLs.
  # RequestHeader set Host example.com:80

  # For websocket/long polling connections to avoid timeouts
  Timeout 86400

</VirtualHost>
```

Basic config using prefix

If you want to serve Workbench from a custom path (e.g. /rstudio) use this example. Replace all occurrences of /rstudio with your chosen custom path prefix.

```
ProxyPreserveHost On
<IfModule mpm_event_module>
    ServerLimit              50
    StartServers             10
    MinSpareThreads          75
    MaxSpareThreads          250
    ThreadLimit              64
    ThreadsPerChild          64
    MaxRequestWorkers        3200
    MaxConnectionsPerChild   10000
</IfModule>

<VirtualHost *:80>
   <Proxy *>
     Allow from localhost
   </Proxy>
   ProxyPass /rstudio/ http://localhost:8787/
   # RequestHeader set Host "example.com:80"
   RequestHeader set X-RStudio-Root-Path "/rstudio"
   ProxyRequests Off
</VirtualHost>
```

SSL and load balancing

This example configures Apache to use SSL but where rserver.conf does not have ssl-enabled=1 so it's using http, not https.

For Apache to serve SSL, it requires enabling the mod_ssl module.

```
ProxyPreserveHost On
<IfModule mpm_event_module>
    ServerLimit              50
    StartServers             10
    MinSpareThreads          75
    MaxSpareThreads          250
    ThreadLimit              64
    ThreadsPerChild          64
    MaxRequestWorkers        3200
    MaxConnectionsPerChild   10000
</IfModule>
```

```
<VirtualHost *:443>
   ServerName example.com

   SSLProxyEngine on
   SSLEngine on
   SSLCertificateFile /etc/rstudio/rserver.crt
   SSLCertificateKeyFile /etc/rstudio/rserver.key

   #SSLCertificateChainFile /path/to/DigiCertCA.crt

   # Need this if Apache is using https and rserver is using http as the protocol is not in h
   RequestHeader add X-Forwarded-Proto "https"

   # Cookie used by Apache to do sticky session balancing
   Header add Set-Cookie "PWB_ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_

   <Proxy balancer://rserver-http>
      BalancerMember http://node1:8787 route=node1
      BalancerMember http://node2:8787 route=node2
      ProxySet stickysession=PWB_ROUTEID
   </Proxy>

   <Proxy balancer://rserver-ws>
      BalancerMember ws://node1:8787 route=node1
      BalancerMember ws://node2:8787 route=node2
      ProxySet stickysession=PWB_ROUTEID
   </Proxy>

   # For websockets - for update/connection requests rewrite the address to do ws specific pr
   # for load balanced setups, we map it to the ws balancer
   RewriteEngine on
   RewriteCond %{HTTP:Upgrade} websocket [NC]
   RewriteCond %{HTTP:Connection} upgrade [NC]
   RewriteRule ^/?(.*) "balancer://rserver-ws/$1" [P,L]

   ProxyPass / balancer://rserver-http/
   ProxyPassReverse / balancer://rserver-http/
   ProxyRequests Off
   # 1d in seconds (also sets default for ProxyTimeout)
   Timeout 86400
</VirtualHost>
```

Restarting Apache

After you've completed all of the above steps you'll then need to restart Apache so that the proxy settings take effect:

```
sudo /etc/init.d/apache2 restart
```

or

```
sudo systemctl restart httpd
```

Workbench configuration

For improved security when not load balancing, if your Workbench and proxy server are running on the same machine you can also change the port Workbench listens on from 0.0.0.0 (all remote clients) to 127.0.0.1 (only the localhost). This ensures that the only way to connect to Workbench is through the proxy server. You can do this by adding the `www-address` entry to the **`/etc/rstudio/rserver.conf`** file as follows:

```
www-address=127.0.0.1
```

Note that you may need to create this config file if it doesn't already exist.

Customizing default proxy

Workbench exposes itself over TCP by means of an nginx proxy instance that runs as the `rserver-http` process on the local machine. In some cases, this proxy instance may need to be customized.

In order to customize it, you can create any of the following three files. Each file modifies the nginx configuration at `/var/lib/rstudio-server/conf/rserver-http.conf` in the following way:

- `/etc/rstudio/nginx.http.conf` - allows you to add additional nginx directives under the root `http` node, and should be used for altering basic HTTP settings
- `/etc/rstudio/nginx.server.conf` - allows you to add additional nginx directives under the `server` node, and should be used for altering basic server settings
- `/etc/rstudio/nginx.site.conf` - allows you to add additional nginx directives under the `location /` node, and should be used for altering responses sent from Workbench

Simply add the desired nginx configuration in the files above to modify the desired section - the contents of each file is copied into the `rserver-http.conf` template verbatim. Then, restart `rstudio-server` for the changes to take effect.

In most cases, you should not need to create these files and modify the nginx template that is provided.