Apache Module mod\_proxy\_balancer

|  |  |
| --- | --- |
| [**Description:**](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOJC%2BZFVPTXaCF0YV9hM%3D&b=1#Description) | [mod\_proxy](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1) extension for load balancing |
| [**Status:**](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOJC%2BZFVPTXaCF0YV9hM%3D&b=1#Status) | Extension |
| [**Module Identifier:**](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOJC%2BZFVPTXaCF0YV9hM%3D&b=1#ModuleIdentifier) | proxy\_balancer\_module |
| [**Source File:**](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOJC%2BZFVPTXaCF0YV9hM%3D&b=1#SourceFile) | mod\_proxy\_balancer.c |
| [**Compatibility:**](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOJC%2BZFVPTXaCF0YV9hM%3D&b=1#Compatibility) | Available in version 2.1 and later |

Summary

This module *requires* the service of [mod\_proxy](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1) and it provides load balancing for all the supported protocols. The most important ones are:

* HTTP, using mod\_proxy\_http
* FTP, using mod\_proxy\_ftp
* AJP13, using mod\_proxy\_ajp
* WebSocket, using mod\_proxy\_wstunnel

The Load balancing scheduler algorithm is not provided by this module but from other ones such as:

* mod\_lbmethod\_byrequests
* mod\_lbmethod\_bytraffic
* mod\_lbmethod\_bybusyness
* mod\_lbmethod\_heartbeat

Thus, in order to get the ability of load balancing, [mod\_proxy](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1), [mod\_proxy\_balancer](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXUqUWEIA9Rw50m218pER&b=1) and at least one of load balancing scheduler algorithm modules have to be present in the server.

Warning

Do not enable proxying until you have [secured your server](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#access). Open proxy servers are dangerous both to your network and to the Internet at large.Load balancer scheduler algorithm

At present, there are 4 load balancer scheduler algorithms available for use: Request Counting (mod\_lbmethod\_byrequests), Weighted Traffic Counting (mod\_lbmethod\_bytraffic), Pending Request Counting (mod\_lbmethod\_bybusyness) and Heartbeat Traffic Counting (mod\_lbmethod\_heartbeat). These are controlled via the lbmethod value of the Balancer definition. See the [ProxyPass](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxypass) directive for more information, especially regarding how to configure the Balancer and BalancerMembers.

Load balancer stickyness

The balancer supports stickyness. When a request is proxied to some back-end, then all following requests from the same user should be proxied to the same back-end. Many load balancers implement this feature via a table that maps client IP addresses to back-ends. This approach is transparent to clients and back-ends, but suffers from some problems: unequal load distribution if clients are themselves hidden behind proxies, stickyness errors when a client uses a dynamic IP address that changes during a session and loss of stickyness, if the mapping table overflows.

The module [mod\_proxy\_balancer](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXUqUWEIA9Rw50m218pER&b=1) implements stickyness on top of two alternative means: cookies and URL encoding. Providing the cookie can be either done by the back-end or by the Apache web server itself. The URL encoding is usually done on the back-end.

Examples of a balancer configuration

Before we dive into the technical details, here's an example of how you might use [mod\_proxy\_balancer](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXUqUWEIA9Rw50m218pER&b=1) to provide load balancing between two back-end servers:

<**Proxy** "balancer://mycluster">

**BalancerMember** "[http://192.168.1.50:80](http://192.168.1.50/)"

**BalancerMember** "[http://192.168.1.51:80](http://192.168.1.51/)"

</**Proxy**>

**ProxyPass** "/test" "balancer://mycluster"

**ProxyPassReverse** "/test" "balancer://mycluster"

Another example of how to provide load balancing with stickyness using [mod\_headers](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLq6ZBlPQWeFF0YV9hM%3D&b=1), even if the back-end server does not set a suitable session cookie:

**Header** add Set-Cookie "ROUTEID=.%{BALANCER\_WORKER\_ROUTE}e;

path=/" env=BALANCER\_ROUTE\_CHANGED

<**Proxy** "balancer://mycluster">

**BalancerMember** "[http://192.168.1.50:80](http://192.168.1.50/)" route=1

**BalancerMember** "[http://192.168.1.51:80](http://192.168.1.51/)" route=2

**ProxySet** stickysession=ROUTEID

</**Proxy**>

**ProxyPass** "/test" "balancer://mycluster"

**ProxyPassReverse** "/test" "balancer://mycluster"

Exported Environment Variables

At present there are 6 environment variables exported:

**BALANCER\_SESSION\_STICKY**

This is assigned the stickysession value used for the current request. It is the name of the cookie or request parameter used for sticky sessions

**BALANCER\_SESSION\_ROUTE**

This is assigned the route parsed from the current request.

**BALANCER\_NAME**

This is assigned the name of the balancer used for the current request. The value is something like balancer://foo.

**BALANCER\_WORKER\_NAME**

This is assigned the name of the worker used for the current request. The value is something like [http://hostA:1234](http://hosta:1234/).

**BALANCER\_WORKER\_ROUTE**

This is assigned the route of the worker that will be used for the current request.

**BALANCER\_ROUTE\_CHANGED**

This is set to 1 if the session route does not match the worker route (BALANCER\_SESSION\_ROUTE != BALANCER\_WORKER\_ROUTE) or the session does not yet have an established route. This can be used to determine when/if the client needs to be sent an updated route when sticky sessions are used.

Enabling Balancer Manager Support

This module *requires* the service of [mod\_status](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqhdRlfUWbYUVoM9w%3D%3D&b=1). Balancer manager enables dynamic update of balancer members. You can use balancer manager to change the balance factor of a particular member, or put it in the off line mode.

Thus, in order to get the ability of load balancer management, [mod\_status](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqhdRlfUWbYUVoM9w%3D%3D&b=1) and [mod\_proxy\_balancer](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXUqUWEIA9Rw50m218pER&b=1) have to be present in the server.

To enable load balancer management for browsers from the example.com domain add this code to your httpd.conf configuration file

<**Location** "/balancer-manager">

**SetHandler** balancer-manager

**Require** host example.com

</**Location**>

You can now access load balancer manager by using a Web browser to access the page <http://your.server.name/balancer-manager>. Please note that only Balancers defined outside of <Location ...> containers can be dynamically controlled by the Manager.

Details on load balancer stickyness

When using cookie based stickyness, you need to configure the name of the cookie that contains the information about which back-end to use. This is done via the stickysession attribute added to either [ProxyPass](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxypass) or [ProxySet](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxyset). The name of the cookie is case-sensitive. The balancer extracts the value of the cookie and looks for a member worker with route equal to that value. The route must also be set in either [ProxyPass](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxypass) or [ProxySet](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxyset). The cookie can either be set by the back-end, or as shown in the above [example](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXUqUWEIA9Rw50m218pER&b=1&f=norefer) by the Apache web server itself.

Some back-ends use a slightly different form of stickyness cookie, for instance Apache Tomcat. Tomcat adds the name of the Tomcat instance to the end of its session id cookie, separated with a dot (.) from the session id. Thus if the Apache web server finds a dot in the value of the stickyness cookie, it only uses the part behind the dot to search for the route. In order to let Tomcat know about its instance name, you need to set the attribute jvmRoute inside the Tomcat configuration file conf/server.xml to the value of the route of the worker that connects to the respective Tomcat. The name of the session cookie used by Tomcat (and more generally by Java web applications based on servlets) is JSESSIONID (upper case) but can be configured to something else.

The second way of implementing stickyness is URL encoding. The web server searches for a query parameter in the URL of the request. The name of the parameter is specified again using stickysession. The value of the parameter is used to lookup a member worker with route equal to that value. Since it is not easy to extract and manipulate all URL links contained in responses, generally the work of adding the parameters to each link is done by the back-end generating the content. In some cases it might be feasible doing this via the web server using [mod\_substitute](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqhdBpYUHyCTFoEtRcozS8%3D&b=1) or [mod\_sed](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqhZBwFTGGbVQ%3D%3D&b=1). This can have negative impact on performance though.

The Java standards implement URL encoding slightly different. They use a path info appended to the URL using a semicolon (;) as the separator and add the session id behind. As in the cookie case, Apache Tomcat can include the configured jvmRoute in this path info. To let Apache find this sort of path info, you neet to set scolonpathdelim to On in [ProxyPass](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxypass) or [ProxySet](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLqicxdTXTueTUMN&b=1#proxyset).

Finally you can support cookies and URL encoding at the same time, by configuring the name of the cookie and the name of the URL parameter separated by a vertical bar (|) as in the following example:

**ProxyPass** "/test" "balancer://mycluster" stickysession=JSESSIONID|jsessionid scolonpathdelim=On

<**Proxy** "balancer://mycluster">

**BalancerMember** "[http://192.168.1.50:80](http://192.168.1.50/)" route=node1

**BalancerMember** "[http://192.168.1.51:80](http://192.168.1.51/)" route=node2

</**Proxy**>

If the cookie and the request parameter both provide routing information for the same request, the information from the request parameter is used.

Troubleshooting load balancer stickyness

If you experience stickyness errors, e.g. users lose their application sessions and need to login again, you first want to check whether this is because the back-ends are sometimes unavailable or whether your configuration is wrong. To find out about possible stability problems with the back-ends, check your Apache error log for proxy error messages.

To verify your configuration, first check, whether the stickyness is based on a cookie or on URL encoding. Next step would be logging the appropriate data in the access log by using an enhanced [LogFormat](https://eu0.proxysite.com/process.php?d=B1tRzFe3NmbztMLZBIj4Cfw9wXCegmK3rpQvlteeaRno2c7iOLq%2Bbh90R3qYX0cGtRcozS8%3D&b=1#logformat). The following fields are useful:

**%{MYCOOKIE}C**

The value contained in the cookie with name MYCOOKIE. The name should be the same given in the stickysession attribute.

**%{Set-Cookie}o**

This logs any cookie set by the back-end. You can track, whether the back-end sets the session cookie you expect, and to which value it is set.

**%{BALANCER\_SESSION\_STICKY}e**

The name of the cookie or request parameter used to lookup the routing information.

**%{BALANCER\_SESSION\_ROUTE}e**

The route information found in the request.

**%{BALANCER\_WORKER\_ROUTE}e**

The route of the worker chosen.

**%{BALANCER\_ROUTE\_CHANGED}e**

Set to 1 if the route in the request is different from the route of the worker, i.e. the request couldn't be handled sticky.

Common reasons for loss of session are session timeouts, which are usually configurable on the back-end server.

The balancer also logs detailed information about handling stickyness to the error log, if the log level is set to debug or higher. This is an easy way to troubleshoot stickyness problems, but the log volume might be to high for production servers under high load.

# Load Balancer Setup with Apache HTTP

Here you’ll find a quick setup to configure a proxy with load balancer to distribute requests between multiple instances of Server each one running in a different Tomcat process.

**Warning**

The following steps are OS specific as they involve creating scripts and setting environmental variables to run Tomcat. As such we will differentiate instructions between different OS.

## Windows

Into the provided package you’ll find the **apache\_start.bat** script which sets specific environment variables and starts the httpd server as standalone program. Closing the window will shutdown the service.

The **apache\_start.bat** script is used to setup the APACHE HTTPD settings and to startup the server:

set TRAINING\_ROOT=%~dp0

set HTTPD\_HOME=%TRAINING\_ROOT%\Apache2.2

set HTTPD\_PORT=88call %ROOT%\setenv.bat

set EXECUTABLE=%HTTPD\_HOME%\bin\httpd.exe

call %EXECUTABLE%

In the above script we setup some useful environment variables which will be used into the other configurations:

1. the HTTPD\_HOME (set to %TRAINING\_ROOT%/Apache2.2)
2. the HTTPD\_PORT (set to 88)

**Note**

The default Apache port is 80. We are using port 88 to avoid any overlap.

Using the above script you may access to the Apache HTTPD server using a browser pointing to the URL:

<http://localhost:88>

To setup the Load Balancer we still have to configure httpd to proxy the Server tomcat instances.

### HTTPD setup

The main configuration file is located here:

%HTTPD\_HOME%/conf/httpd.conf

Edit that file to add global changes to the httpd server.

F.e. we have added the **HTTPD\_HOME** prefix in front all the relative paths and set the server root and port as following:

ServerRoot "${HTTPD\_HOME}"

#

# Listen: Allows you to bind Apache to specific IP addresses and/or

# ports, instead of the default. See also the <VirtualHost>

# directive.

#

# Change this to Listen on specific IP addresses as shown below to

# prevent Apache from glomming onto all bound IP addresses.

##Listen 12.34.56.78:80

Listen 0.0.0.0:${HTTPD\_PORT}

Be also sure to uncomment the extra configuration inclusions:

*# Virtual hosts*Include conf/extra/httpd-vhosts.conf

**Note**

In the distributed package you’ll find this file under %HTTPD\_HOME%/conf/extra/http-vhosts.conf

Which may result as following:

<VirtualHost \*:${HTTPD\_PORT}>

LoadModule proxy\_ajp\_module modules/mod\_proxy\_ajp.so

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule proxy\_balancer\_module modules/mod\_proxy\_balancer.so

LoadModule proxy\_connect\_module modules/mod\_proxy\_connect.so

LoadModule proxy\_ftp\_module modules/mod\_proxy\_ftp.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

LoadModule reqtimeout\_module modules/mod\_reqtimeout.so

<IfModule mod\_proxy\_ajp.c>

ProxyRequests Off

ProxyTimeout 300

ProxyPreserveHost On

ProxyVia On

<Proxy balancer://cluster>

BalancerMember ajp://localhost:8009 route=route1

BalancerMember ajp://localhost:8010 route=route2

ProxySet lbmethod=bybusyness

</Proxy>

<Location /server>

Order allow,deny

Allow from all

ProxyPass balancer://cluster/server stickysession=JSESSIONID

</Location>

</IfModule>

</VirtualHost>

**Note**

in the above configuration we consider the HTTPD\_PORT environment variable set to an valid integer port number.

The used proxy parameters are:

* 1. [ProxyRequests](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxyrequests)
     1. This allows or prevents Apache from functioning as a forward proxy server. (Setting ProxyRequests to Off does not disable use of the ProxyPass directive.)
  2. [ProxyVia](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxyvia)
     1. This directive controls the use of the Via: HTTP header by the proxy. We don’t need it here, but it does not hurt either.
  3. [ProxyTimeout](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxytimeout)
     1. This directive allows a user to specify a timeout on proxy requests. This is useful when you have a slow/buggy appserver which hangs, and you would rather just return a timeout and fail gracefully instead of waiting however long it takes the server to return.
  4. [ProxyPreserveHost](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypreservehost)
     1. When enabled, this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the ProxyPass line.
  5. [ProxySet](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxyset)
     1. This directive is used as an alternate method of setting any of the parameters available to Proxy balancers and workers normally done via the ProxyPass directive

The Proxy node contains all the BalancerMember of the cluster, in this configuration we configured 2 different instances of tomcat using the ajp connectors (on ports 8009, 8010). It contains some other parameters:

1. [Order](http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html#order)
   1. The Order directive, along with the Allow and Deny directives, controls a three-pass access control system. The first pass processes either all Allow or all Deny directives, as specified by the Order directive. The second pass parses the rest of the directives (Deny or Allow). The third pass applies to all requests which do not match either of the first two.
2. [Allow](http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html#allow)
   1. The Allow directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.
3. [ProxyPass](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypass)
   1. This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. The local server is often called a reverse proxy or gateway. The path is the name of a local virtual path; url is a partial URL for the remote server and cannot include a query string.
4. [ProxyPassReverse](http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypassreverse)
   1. This directive lets Apache adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy (or gateway) to avoid by-passing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy
5. [stickysession](https://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html#stickyness)
   1. This directive is required for the user to be able to interact with Server’s UI. When a session is created for the user (e.g. as a result of the user logging into Server’s UI) all subsequent requests from the user will be routed through the same route instead of being balanced to both instances. This is required to properly interact with the UI.

**Warning**

When a session is created (e.g. you used the Server GUI or the integrated GWC GUI) a cookie is set in your browser to transport the Session ID (it uses the key JSESSIONID). To observe the behaviour where incoming requests are being balanced to all Server instances you need to either clear your browser cookies or open an [incognito mode](https://en.wikipedia.org/wiki/Privacy_mode) window in the browser. Normally users going through the OGC services will not have a session id.

[For the tomcat connectors configuration click here](https://geoserver.geo-solutions.it/edu/en/clustering/load_balancing/tomcat.html#tomcat).

Each BalancerMember points to a tomcat instance using the ajp protocol (through its route: see below):

<Proxy balancer://cluster>

BalancerMember ajp://localhost:8009 route=route1

BalancerMember ajp://localhost:8010 route=route2

ProxySet lbmethod=bybusyness

</Proxy>

**Note**

The traffic is routed bybusyness but you may chose a different algorithm (see Proxy doc).

If you want to use the http connector you may change the proxy configuration pointing to those connectors:

<Proxy balancer://cluster>

BalancerMember <http://localhost:8083>

BalancerMember <http://localhost:8084>

ProxySet lbmethod=bybusyness

</Proxy>

The incoming requests is redirect from the proxy to the load balancer by:

<Location /server>

Order allow,deny

Allow **from** **all**

ProxyPass balancer://cluster/server

stickysession=JSESSIONID

</Location>

Each request to:

[http://localhost:88/server](http://localhost:88/geoserver)

is now redirected to one of your tomcat instances depending on the incoming traffic and accordingly to the chosen lbmethod. You are able to check the status of each Balancer Member using your browser (protected by authentication).

## Linux

**Warning**

The following instructions are distribution-dependant and they have been tested on Ubuntu 12.04 that is the OS of the linux training environment. Although they should be valid on all the Debian-based distributions some changes may are required on other Debian or Ubuntu versions. For the Linux Red Hat-based distributions the apache package is called **httpd** instead of apache2.

Install the **apache webserver** on the linux training environment running the command:

sudo apt-get install apache2

The apt-get installation process will install Apache 2.2.22 on your system as a system service.

start the apache2 daemon:

sudo service apache2 start

Stop you apache2 daemon:

sudo service apache2 stop

Reload the configuration files without stopping the service:

sudo service apache2 reload

**Note**

For some configurations a reload is not enough and a restart is required, see the [official apache documentation](http://httpd.apache.org/docs/2.2/) for more info

The main apache modules configuration directory is /etc/apache2/mods-enabled/

To configure the apache2 proxy you have to configure the tomcat connector (see the previous chapter) and the webserver configuration.

First load all the apache2 needed modules that are not enabled by default:

sudo cp /etc/apache2/mods-available/proxy.load /etc/apache2/mods-enabled

sudo cp /etc/apache2/mods-available/proxy\_ajp.load /etc/apache2/mods-enabled

sudo cp /etc/apache2/mods-available/proxy\_balancer.load /etc/apache2/mods-enabled

Then create a new configuration file under the apache modules configuration directory:

sudo nano /etc/apache2/mods-enabled/proxy\_balancer.conf

Edit it as follows:

ProxyRequests Off

ProxyTimeout 300

ProxyPreserveHost On

ProxyVia On

<Proxy balancer://cluster>

BalancerMember ajp://localhost:8009 route=route1

BalancerMember ajp://localhost:8010 route=route2

ProxySet lbmethod=bybusyness

</Proxy>

<Location /server>

Order allow,deny

Allow **from** **all**

ProxyPass balancer://cluster/server

stickysession=JSESSIONID

</Location>

Save the file (ctrl-o using nano as suggested) and restart apache as seen before (a configuration reload in not enough for this step).

Open the browser and access to the url:

[http://localhost/server](http://localhost/geoserver)

The server is now accessed through the apache2 webserver and the requests are balanced between the 2 server instances available.

## Further optional configurations

The following section is optional (not configured in the Linux/Windows packages). This is useful to check the configuration using your browser:

<Location /balancer-manager> SetHandler balancer-manager AuthType basic AuthName "My\_auth\_name" AuthUserFile "/etc/httpd/passwd/passwords" *# Anonymous \** Require valid-user</Location>

**Note**

Here we also have set a basic authentication to access the basic auth

Point your browser to <http://localhost:88/balancer-manager> to see the manager.

To configure a basic authentication account on Linux proceed as following.

Create a password file:

mkdir /etc/httpd/passwdtouch /etc/httpd/passwd/passwords

Create the user:

htpasswd -c /etc/httpd/passwd/passwords USERPassword: PASSWORD