How To Use Apache as a Reverse Proxy with mod_proxy on CentOS 7

A reverse proxy is a type of proxy server that takes HTTP(S) requests and transparently distributes them to one or more backend servers. Reverse proxies are useful because many modern web applications process incoming HTTP requests using backend application servers which aren't meant to be accessed by users directly and often only support rudimentary HTTP features.

You can use a reverse proxy to prevent these underlying application servers from being directly accessed. They can also be used to distribute the load from incoming requests to several different application servers, increasing performance at scale and providing fail-safeness. They can fill in the gaps with features the application servers don't offer, such as caching, compression, or SSL encryption too.

In this tutorial, you'll set up Apache as a basic reverse proxy using the mod_proxy extension to redirect incoming connections to one or several backend servers running on the same network. This tutorial uses a simple backend written with the with Flask web framework, but you can use any backend server you prefer.

Prerequisites

To follow this tutorial, you will need:

- One CentOS 7 server set up with this initial server setup tutorial, including a sudo non-root user.
- Apache 2 installed on your server by following Step 1 of How To Install Linux, Apache, MySQL, PHP (LAMP) Stack on CentOS 7.
- Optionally, the nano text editor installed with yum install nano. CentOS comes with the vi text editor by default, but nano can be more user friendly.

Step 1 - Introducing Necessary Apache Modules

Step 2 - Creating Backend Test Servers (Optional)

Step 3 - Modifying the Default Configuration to Enable Reverse Proxy

Example 1 — Reverse Proxying a Single Backend Server

Paste the following contents into the default-site.conf file, so your configuration file looks like this:

```
<VirtualHost *:80>
    ProxyPreserveHost On

    ProxyPass / http://127.0.0.1:8080/
    ProxyPassReverse / http://127.0.0.1:8080/
</VirtualHost>
```

…

Example 2 — Load Balancing Across Multiple Backend Servers

If you have multiple backend servers, a good way to distribute the traffic across them when proxying is to use load balancing features of mod_proxy.

Replace all the contents within the VirtualHost block with the following, so your configuration file looks like this:

```

<VirtualHost *:80>
<Proxy balancer://mycluster>
    BalancerMember http://127.0.0.1:8080
    BalancerMember http://127.0.0.1:8081
</Proxy>

    ProxyPreserveHost On

    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/
</VirtualHost>
```

…

---

https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod_proxy-on-centos-7

Last update: **2017/10/24 12:52**