How To Create Temporary and Permanent Redirects with Apache

What is an HTTP Redirect?

HTTP redirection, or URL redirection, is a technique of pointing one domain or address to another. There are many uses for redirection, and a few different kinds of redirection to consider.

As you create content and administrate servers, you will often find the need to redirect traffic from one place to another. This guide will discuss the different use-cases for these techniques, and how to accomplish them in Apache and Nginx.

Why Do Servers Use Redirects?

Redirects are used whenever a site needs people requesting one address to be directed to another address. There are many situations where you may find yourself in this position.

Moving to a Different Domain

If you have established a web presence and would like to change your domain, it is best not to just abandon your old domain.

Bookmarks to your site and links to your site located on other pages throughout the internet will break if your content disappears without any instructions to the browser about how to find its new location.

Changing domains without redirecting will cause your site to lose traffic from previous visitors and lose all of the credibility you have worked to establish.

Expanding to Capture Similar Domains

Often, it is helpful to register multiple variations of a name in order to benefit from users typing in addresses similar to your main domain.

For example, if you have a domain called "mymessyspiders.com", you might also buy the domain names for "mymessyspiders.net" and "mymessyspiders.org" and redirect them both to your "mymessyspiders.com" site.

This allows you to catch users who might be trying to get to your site with the wrong address. It can also help prevent another site from using a similar domain and profiting off of your web presence.

Creating a Persistent Experience In Spite of Page Name Changes

Sometimes, it is necessary to change the names of pages that have already been published and received traffic on your site.

Normally, this would lead to a 404 Not Found error or possibly another error depending on your security settings. These can be avoided by leading your visitors to another page that contains the correct content they were trying to access.

Forcing SSL

A simple but common use for redirects is directing all site traffic to use SSL instead of standard HTTP.

Using redirects, it is possible to make all requests for "http://www.mysite.com" be redirected to "https://www.mysite.com".

Note the HTTPS instead of HTTP.

Redirect Methods

There are a few different kinds of URL redirects, each of which mean something different to the client browser.

The two most common types are 302 temporary redirects, and 301 permanent redirects.

Temporary Redirects

Temporary redirects are useful if your web content for a certain URL temporarily needs to be served out of a different location.

For example, if you are performing site maintenance, you may wish to use a temporary redirect of all of the pages for your domain to an explanation page to inform your visitors that you will be back shortly.

Temporary redirects inform that browser that the content is temporarily located at a different location, but that they should continue to attempt to access the original URL.

Permanent Redirects

Permanent redirects are useful when your content has been moved to a new location forever.

This is useful for when you need to change domains or when the URL needs to change for other reasons and the old location will no longer be used.

This redirect informs the browser that it should no longer request the old URL and should update its information to point to the new URL.

How to Redirect in Apache

Apache can redirect using a few different tools. The simplest ways are accomplished with tools from the "mod_alias" module, but more extensive redirects can be created with "mod_rewrite".

Using the Redirect Directive

In Apache, you can accomplish simple, single-page redirects using the "Redirect" directive, which is included in the "mod_alias" module. This directive takes at least two arguments: the old URL and the new URL.

In its simplest form, you can accomplish a redirect with the following lines in your server configuration:

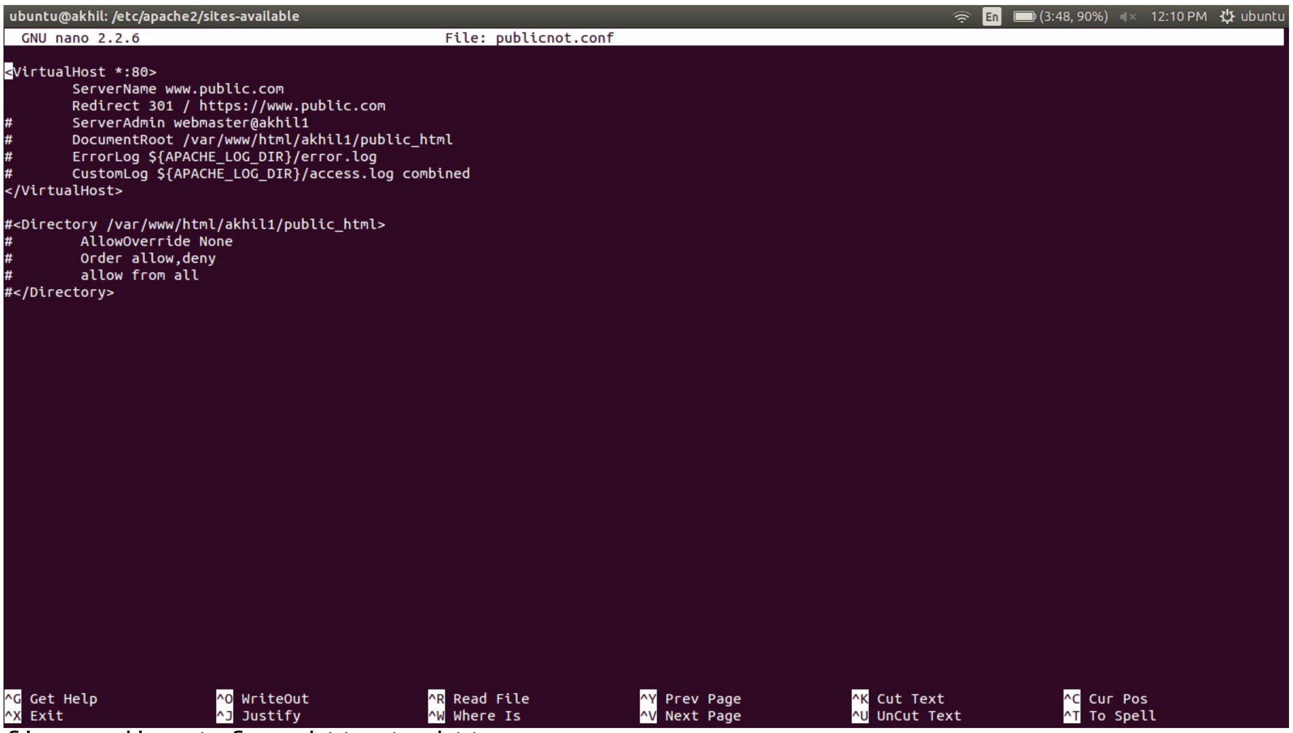
```
<VirtualHost *:80>
        ServerName www.domain1.com
        Redirect / http://www.domain2.com
</VirtualHost>

<VirtualHost *:80>
        ServerName www.domain2.com
        . . .
        . . .
</VirtualHost>
```

This redirect instructs the browser to direct all requests for "www.domain1.com" to "www.domain2.com". This is only for a single page, not for the entire site

By default, the "Redirect" directive establishes a 302, or temporary, redirect.

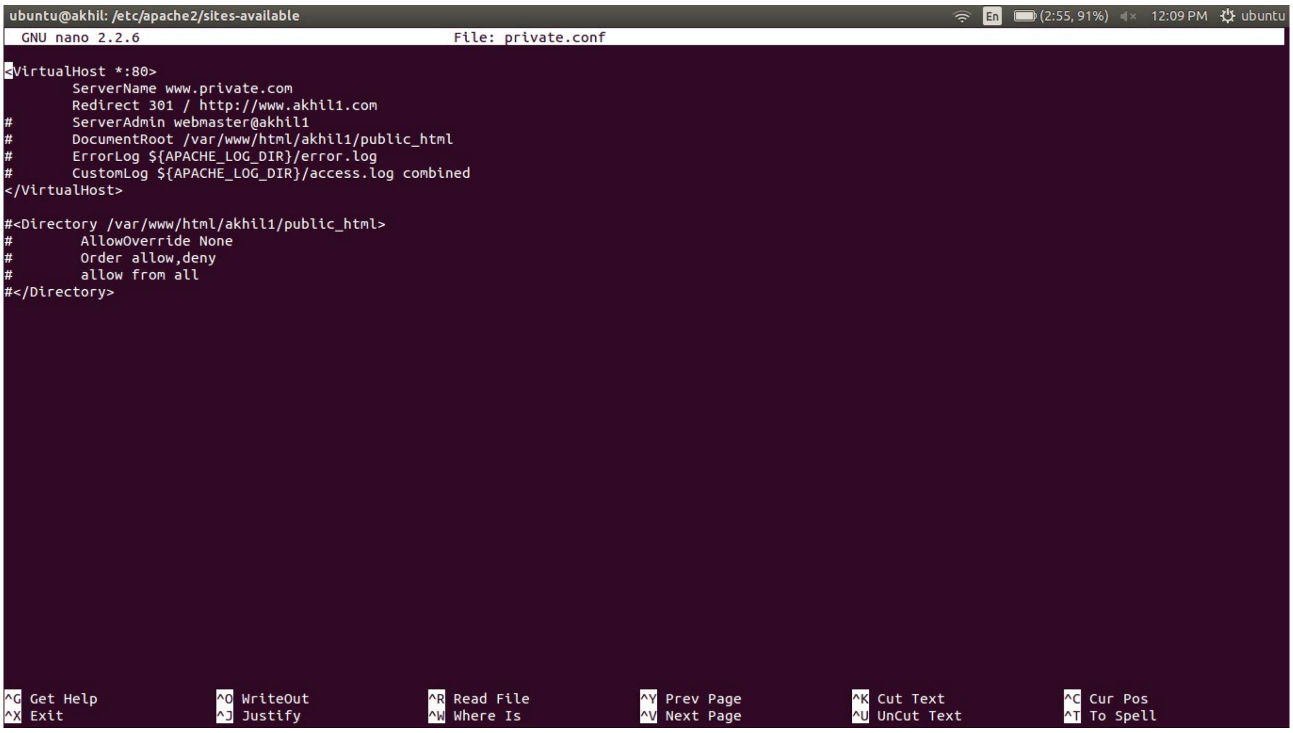
If you would like to create a permanent redirect, you can do so in either of the following two ways:

```
Redirect 301 /oldlocation http://www.domain2.com/newlocation
Redirect permanent /oldlocation http://www.domain2.com/newlocation
```



```
ubuntu@akhil: /etc/apache2/sites-available
  GNU nano 2.2.6                         File: publicnot.conf

<VirtualHost *:80>
        ServerName www.public.com
        Redirect 301 / https://www.public.com
#       ServerAdmin webmaster@akhil1
#       DocumentRoot /var/www/html/akhil1/public_html
#       ErrorLog ${APACHE_LOG_DIR}/error.log
#       CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

#<Directory /var/www/html/akhil1/public_html>
#        AllowOverride None
#        Order allow,deny
#        allow from all
#</Directory>
```

fig: redirect from http to https



```
ubuntu@akhil: /etc/apache2/sites-available                    (2:55, 91%)  12:09 PM  ubuntu
  GNU nano 2.2.6                        File: private.conf

<VirtualHost *:80>
        ServerName www.private.com
        Redirect 301 / http://www.akhil1.com
#       ServerAdmin webmaster@akhil1
#       DocumentRoot /var/www/html/akhil1/public_html
#       ErrorLog ${APACHE_LOG_DIR}/error.log
#       CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

#<Directory /var/www/html/akhil1/public_html>
#        AllowOverride None
#        Order allow,deny
#        allow from all
#</Directory>

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

fig: one host to another

Using the RedirectMatch Directive

To redirect more than a single page, you can use the "RedirectMatch" directive, which allows you to specify directory matching patterns using regular expressions.

This will allow you to redirect entire directories instead of just single files.

RedirectMatch matches patterns in parenthesis and then references the matched text in the redirect using "$1", where 1 is the first group of text. Subsequent groups are given numbers sequentially.

For example, if we wanted to match every request for something within the "/images" directory to a subdomain named "images.example.com", we could use the following:

```
RedirectMatch ^/images/(.*)$ http://images.example.com/$1
```

As with the "Redirect" directive, you can specify the type of redirect by adding the redirect code before the URL location rules.

Using mod_rewrite to Redirect

The most flexible, but complicated way to create redirect rules is with the module called "mod_rewrite".

This is outside of the scope of this article, but you can learn how to use mod_rewrite in this article