How To Set Up a Reverse Proxy (Step-By-Steps for Nginx and Apache)

KINSTA

A reverse proxy sits in front of a web server and receives all the requests before they reach the origin server. It works similarly to a forward proxy, except in this case it's the web server using the proxy rather than the user or client. Reverse proxies are typically used to enhance performance, security, and reliability of the web server.

For example, you can have a non-WordPress site hosted at **`example.com`** domain on Server A and have its blog running on WordPress at **`example.com/blog`** URL hosted on Server B. You can achieve this by adding a reverse proxy for the server hosting your primary site. You can configure the reverse proxy to redirect requests to the blog to a different server (e.g. a managed WordPress host like Kinsta).

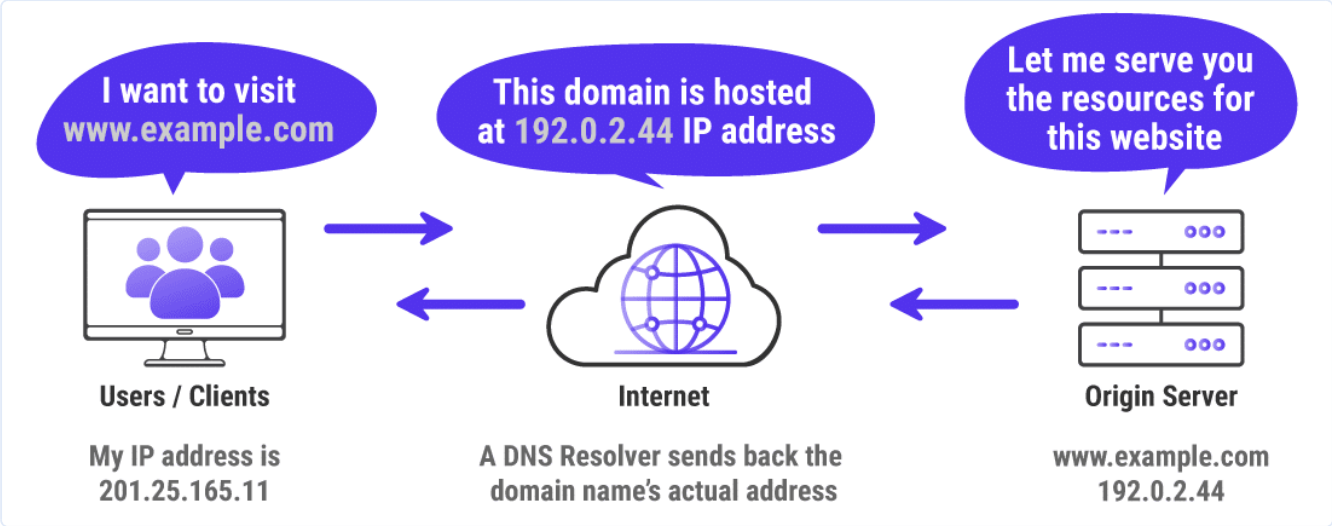
In this article, you'll learn the basics of reverse proxy servers, how they work, what their major benefits are, and how you can use them to speed up and secure your WordPress site.

Excited? Let's start!

What Is a Reverse Proxy?

To understand what a reverse proxy server is, you need first to know its role and get familiar with all its related terms.

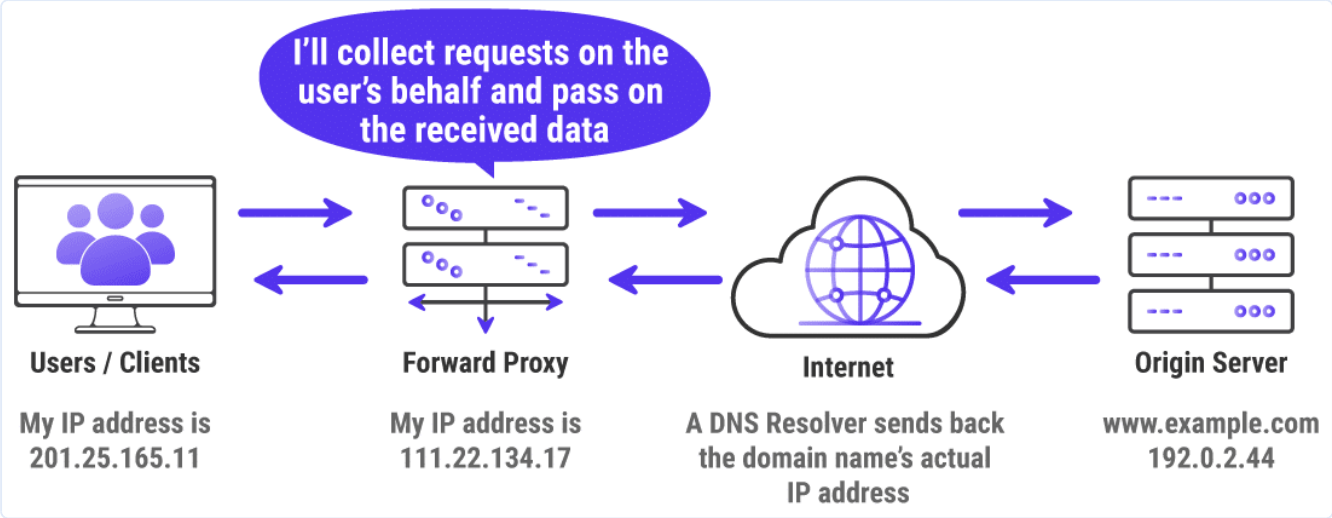
When you browse the web normally by entering a domain name or clicking a link, your browser/device connects to the website's server directly and starts downloading its resources.



— How browsing on the internet works usually

If you want to anonymize your IP address from the websites you visit, then you can use a proxy server to send all your requests to it first. It'll forward your requests to the DNS resolver and then download the website's resources from its origin server.

Afterward, it'll pass on those resources to your device. This is called a forward proxy.



— How a forward proxy server works

You're completely hidden from the website as it thinks your request is originating from the forward proxy.

Info

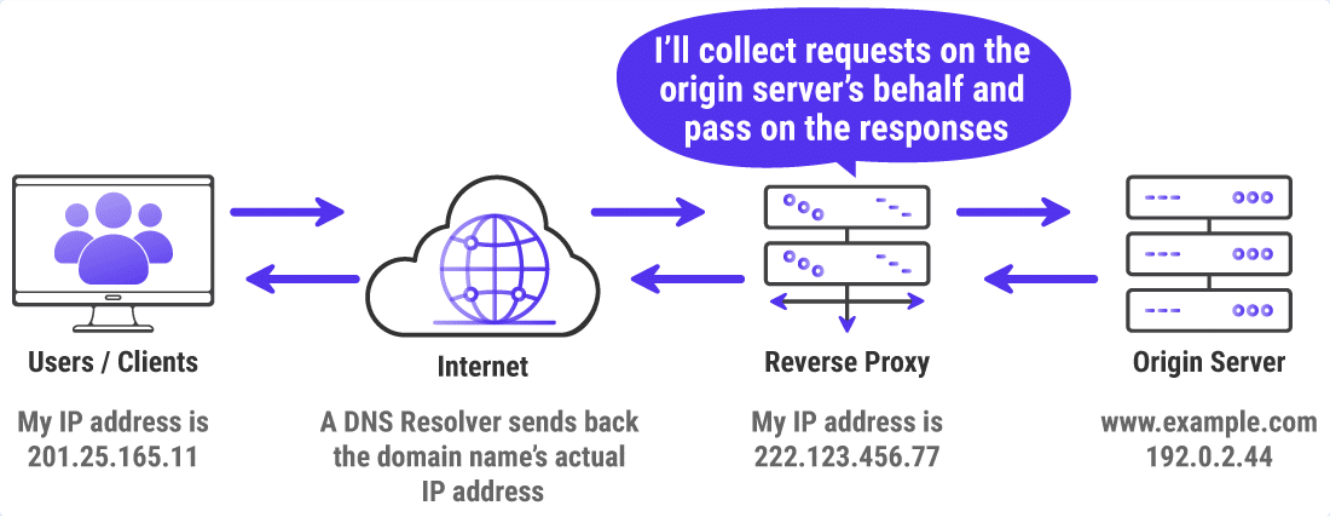
Due to the way some hosting providers like Kinsta count site visits, they require users to set a header to notify their real IP address to the origin server. Hence, the privacy benefits don't apply in specific cases such as these.

Apart from enhancing user privacy, a forward proxy is mainly used to bypass geographical content restrictions. For instance, if you want to watch a video that's blocked in your region, you can use a forward proxy with an IP address on which the video is available to view.

A forward proxy works almost the same way as a Virtual Private Network (VPN), but they're distinct technologies with unique use cases (they can sometimes overlap though).

Reverse Proxy Server vs Forward Proxy Server

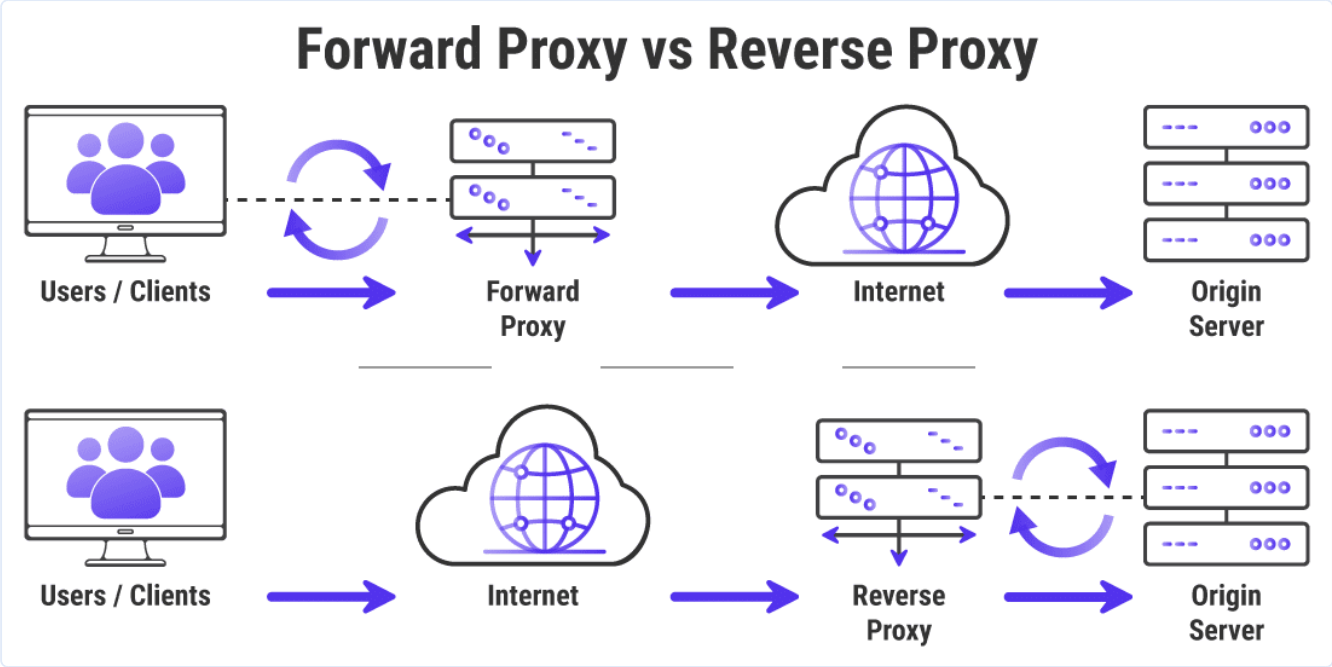
A reverse proxy server acts as a front for the origin server to maintain anonymity and enhance security, just like how a user/client can use a forward proxy to achieve the same. It ensures that no user or client communicates directly with the origin server.



— How a reverse proxy server works

The difference between a forward proxy vs a reverse proxy is minor, but they work differently.

Both can work together as there's no overlap between their functioning. Typically, users/clients use a forward proxy, while origin servers use a reverse proxy.



— Forward Proxy vs Reverse Proxy servers

Since a server admin can control how the reverse proxy works, you can use it to enable many useful features.

We'll list all its benefits later in this post.

Why Use a Reverse Proxy?

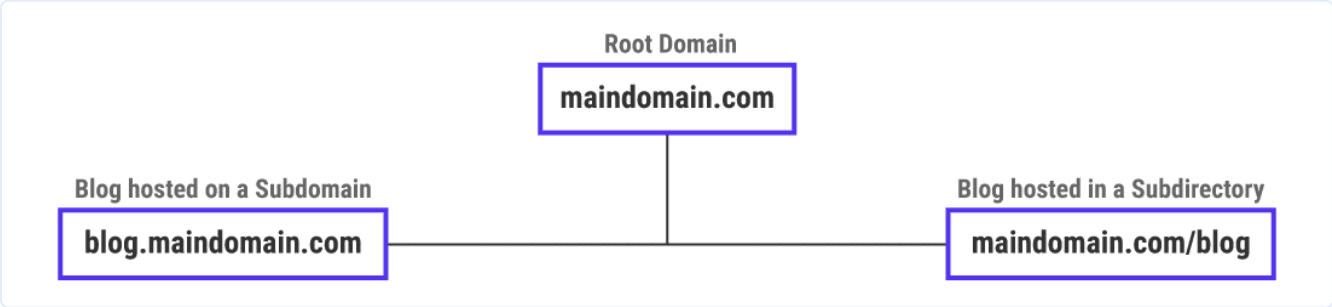
Many businesses, especially large enterprises, use bespoke websites that are tailor-made to their unique needs and aren't running on WordPress. Some examples include bank and insurance websites.

In other cases, a business may host their site on an external service that doesn't allow them to install any external software (e.g. WordPress). Usually, these are small to mid-sized retailers using an ecommerce platform such as Shopify.

Since WordPress has robust CMS features, many businesses, including large enterprises with bespoke websites, may prefer to host their blogs using WordPress.

One way to get around this problem is to install WordPress on the main website's subdomain and structure the navigation menus such that users can switch easily between the main website and the blog.

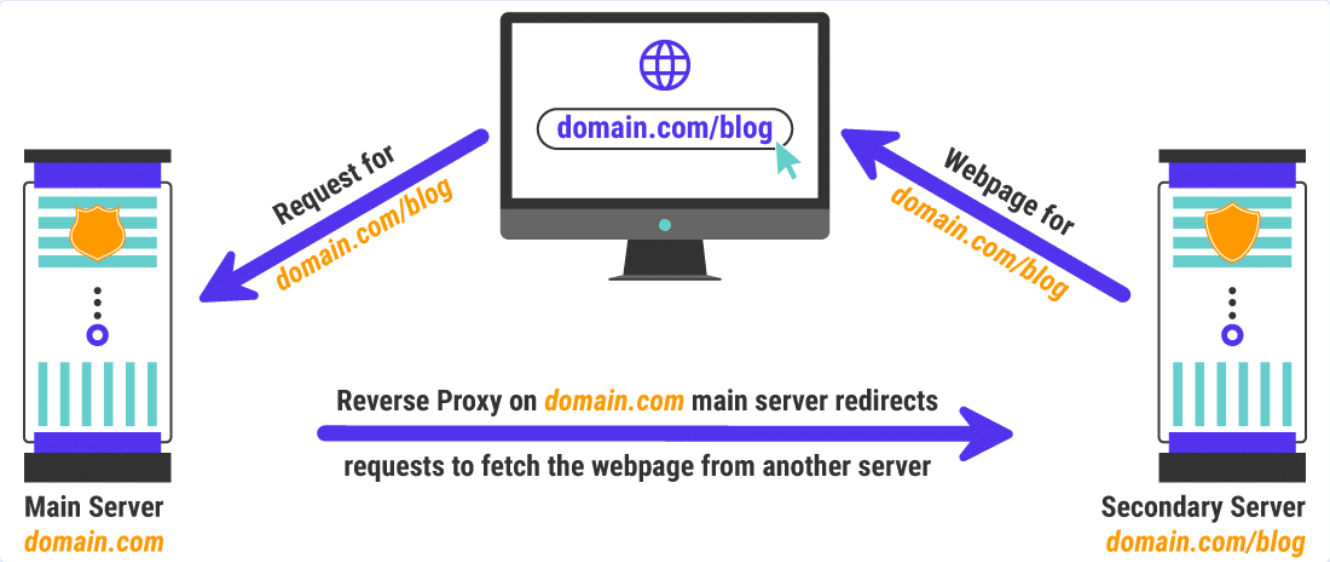
Since subdomains behave as a unique domain, it can affect your site's SEO. Even though Google treats both subdomains and subdirectories equally, it takes more effort to optimize a website for search engine rankings if it's hosted on a subdomain than if it's hosted in a subdirectory.



— Two approaches to hosting blogs on a website

Google has reaffirmed that it treats both subdomains and subdirectories equally, but some SEO experts disagree with it. And even if it doesn't affect the site's SEO, a site hosted in a subdirectory is simply easier to maintain.

That's why you can use a reverse proxy to redirect requests to the site's blog hosted on a separate server. For example, a bank can host their main website on their servers securely, but they can also host their WordPress-powered blog separately on a managed WordPress host like Kinsta.



— An example of a reverse proxy use case

Unifying two different sites under a single domain name is one of the key advantages of using a reverse proxy. It helps brands keep their sites organized, professional, and maintain credibility.

Benefits of Using a Reverse Proxy

Besides the above use case, reverse proxies also grant many other benefits. The section below discusses some of their major advantages.

Load Balancing

A single origin server cannot handle all the incoming traffic for a website with millions of daily unique visitors. In these cases, you can distribute the traffic smartly among a pool of many servers. Usually, all the servers will host the same content to eliminate a single point of failure, making the website more reliable.

A reverse proxy is a great way to set this up as it can receive the incoming traffic before it reaches the origin server. If the origin server is overloaded or fails completely, it can distribute

the traffic to other servers without affecting the site functionality.

Reverse proxies can also direct the incoming requests to several servers, with each server performing a specific function it's optimized for. The reverse proxy can then gather responses from all the servers and deliver them to the client.

Since we use most of the popular reverse proxies primarily for load balancing, they're also referred to as **Load Balancers**.

Global Server Load Balancing (GSLB)

GSLB is an advanced load balancing method for distributing website traffic among many servers placed strategically around the world. It's typically done via anycast routing technique, where the reverse proxy picks the server node based on the fastest travel time between the client and the server.

Not only does GSLB increase the site's reliability and security considerably, it also reduces latency and load times, thereby enhancing user experience. You can use GSLB with other network optimization techniques such as Spoon Feeding to free up the origin servers' computational resources even more.

Though you can set up Global Server Load Balancing manually on your server, it's usually taken care of by dedicated CDNs such as Cloudflare and KeyCDN (which also powers Kinsta CDN). Kinsta serves all the websites hosted with it through a Load Balancer powered by Google Cloud Platform.

Enhanced Security

Reverse proxies can cloak the IP address and other characteristics of origin servers. Thus, your website's origin server can maintain its anonymity better, increasing its security significantly.

Since the reverse proxy will receive all the traffic before it reaches the main server, any attackers or hackers will find it harder to target your website with security threats such as DDoS attacks.

You can use a strict firewall to harden the reverse proxy with tighter security against common cyber attacks. Without a reverse proxy installed, it is difficult to remove malware or start takedowns.

> **Info**
>
> Kinsta uses reverse proxies in its backend architecture and offers free WordPress hack fixes to all the websites it hosts.

A reverse proxy like HAProxy can add basic HTTP access authentication to a web server that doesn't have it enabled. You can also use a reverse proxy to add centralized authentication for various types of requests.

Powerful Caching

You can use a reverse proxy for web acceleration purposes by caching both static and dynamic content. This can reduce the load on the origin server, resulting in a faster website.

For instance, if your origin server is in the USA and a user from Europe visits your website, then you can serve a cached version of your site from a reverse proxy server in Europe. Since the reverse proxy is closer to the user than the origin server, the website will take less time to load, making it perform superbly.

Varnish and Nginx FastCGI are prominent examples of reverse proxies that are used for caching web content. If your site is hosted with Kinsta, you don't have to worry about caching as Kinsta takes care of all the caching legwork for you.

Superior Compression

Server responses use up a lot of bandwidth. Compressing server responses (e.g. with gzip) before sending them to the client can reduce the amount of bandwidth required, speeding up server responses over the network.

A reverse proxy is ideal to compress server responses as it sits in between the origin servers and the client.

Optimized SSL Encryption

Encrypting and decrypting SSL/TLS requests for each client can be highly taxing for the origin server. A reverse proxy can take up this task to free up the origin server's resources for other important tasks, like serving content.

Another advantage of offloading SSL/TSL encryption and decryption is to reduce latency for clients that are geographically distant from the origin server.

You can also opt for a reverse proxy with specialized SSL/TLS acceleration hardware to optimize this task even further. Such a reverse proxy is called an SSL/TLS termination proxy. Some servers like Varnish do not support SSL/TSL protocols, so an SSL/TLS termination reverse proxy can help secure the traffic passing through them.

Better A/B Testing

Most A/B testing tools require you to use external JavaScript libraries to load their functions. However, loading third-party scripts can slow down your page load times and create a choppy experience for users.

Instead, you can use a reverse proxy to create two separate flows at the server level itself. For example, you can use Nginx's **`split_clients`** or **`sticky route`** methods to control traffic redirection.

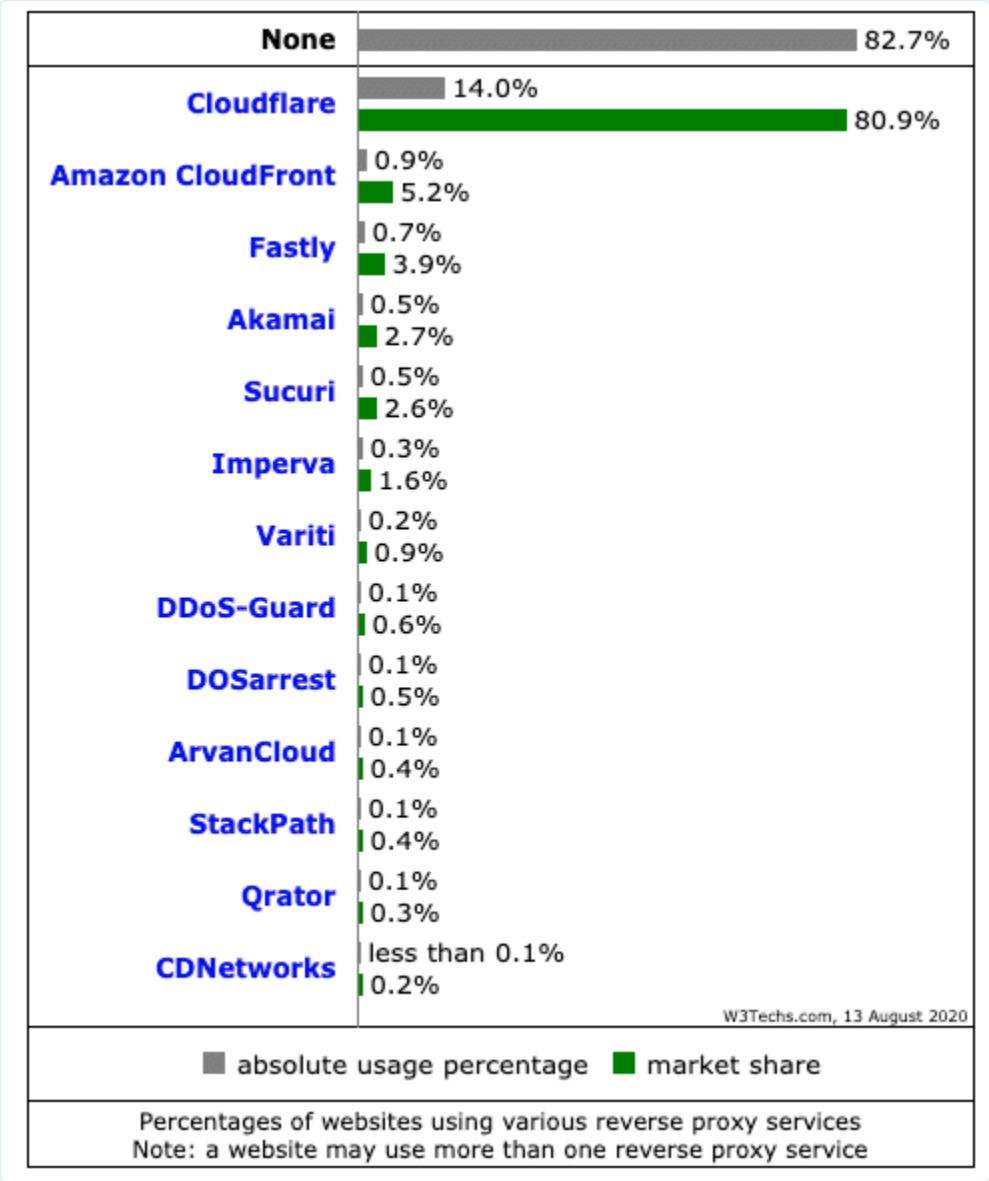
You can refer to tutorials on Nginx and freeCodeCamp to learn more about performing A/B testing with a reverse proxy.

Monitoring and Logging Traffic

A reverse proxy captures any requests that go through it. Hence, you can use them as a central hub to monitor and log traffic. Even if you use multiple web servers to host all your website's components, using a reverse proxy will make it easier to monitor all the incoming and outgoing data from your site.

The Most Popular Reverse Proxies

As per W3Techs, **almost 83%** of the websites use no reverse proxy services that they monitor.



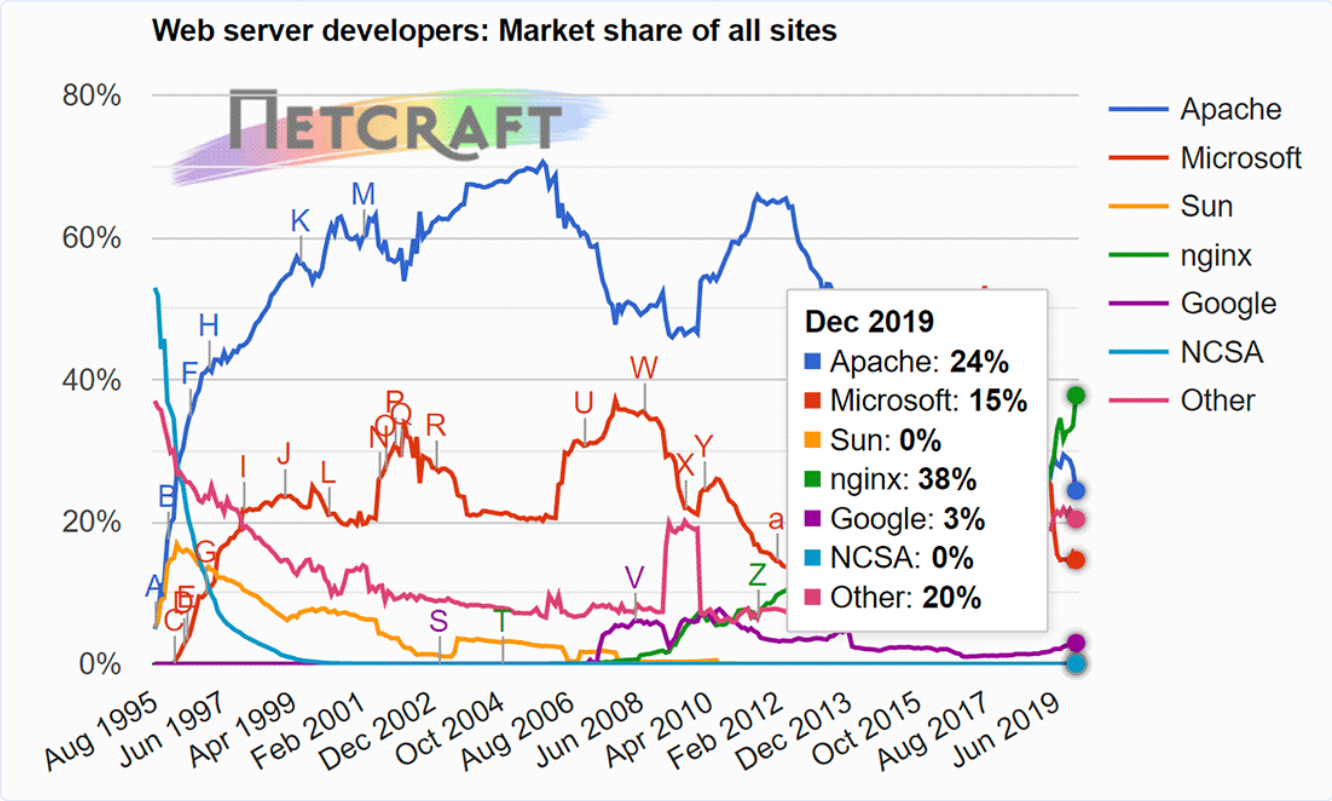
— Statistics of reverse proxies used by websites (Source: W3Techs.com)

Of the **17%** websites that use a reverse proxy (listed above), you'll notice that most of them are CDNs. That's because most reverse proxies hide their existence by default as a safety precaution. Hence, you cannot rely on website monitoring services like W3Techs to find which reverse proxies are the most popular ones.

From our research and experience, the most popular reverse proxies in use today are:

Nginx

Nginx is an open source web server that can also serve as a reverse proxy. Apart from being used to host websites, it's also one of the most widely used reverse proxy and load balancing solutions. As per Netcraft, over **479 million** web servers were using Nginx in December 2019, making it the leader in the web server market share.



— Web server market share of all sites (Source: Netcraft)

Nginx provides all the reverse proxy benefits discussed above, plus more. It improves web performance, security, reliability, and scalability. You can configure Nginx using its configuration file, which is also hot reloadable. At Kinsta, Nginx reverse proxy is one of several premium add ons you can use.

But you can also use Nginx Plus, a commercial offering, to get access to API-based configuration options and other features suitable for large enterprise websites.

Kinsta powers all its websites with Nginx. It has ranked in Review Signal's Top Tier web hosting status in every category it has competed in. Some other major companies that use Nginx are MaxCDN, Cloudflare, and Netflix.

Setting up Nginx as a basic reverse proxy is simple. Nginx also provides you with various directives to customize your server's reverse proxy as per your requirements. We'll discuss how to do this in a later section. If you're a Kinsta customer, you'll also learn how to use a reverse proxy for websites hosted with Kinsta in the same section.

Varnish

Varnish is an open source HTTP reverse proxy with a built-in cache engine. It's designed primarily for high-traffic websites that serve dynamic content. You can also use Varnish as a load balancer, a web app firewall (WAF), and an edge authentication and authorization server.

It works on all modern versions of Linux and FreeBSD, being used mainly as a front for Nginx or Apache web servers. Varnish's powerful and highly flexible Varnish Configuration Language (VCL) lets you define various features such as handling HTTP requests, caching, and connecting to one or more web servers.

For this reason, many CDNs use Varnish as their main foundation for delivering content swiftly.

Varnish also supports Edge Side Includes (ESI), a language that helps you to reuse sections of one web page in other web pages. If your website uses a lot of repeated content in different pages, ESI can help you speed up your site's page load times by caching frequently used sections.

You can extend Varnish with its various modules (VMODs). Head to Varnish's official tutorial to learn how to set up Varnish as a reverse proxy for WordPress.

Apache Traffic Server

Apache Traffic Server is an open source caching proxy server. It's popular for its fast, scalable features. It was a commercial product developed by Yahoo! long ago, but they made

it open source and donated it to the Apache Foundation for maintenance.

Several major content networks and CDNs like Comcast, Akamai, LinkedIn, Yahoo, and Apple use Apache Traffic Server to power their technology.

You can also use Apache HTTP Server (**Apache httpd**), an HTTP server daemon, to set up a reverse proxy on your web server. Apart from acting as a basic web server, it also helps you serve static and dynamic content to users. You'll learn how to set up Apache as a reverse proxy later in this article.

HAProxy

HAProxy is an open source reverse proxy and load balancer. It's designed to integrate with most existing web server architectures, including Linux distributions and cloud platforms. Similar to Nginx, HAProxy uses an event-driven I/O model and supports splitting requests across multiple worker processes.

For HTTP requests, HAProxy performs exceptionally well even under heavy loads. Some of the highest traffic websites on the internet such as Airbnb, Reddit, Instagram, Stack Overflow, Tumblr, GitHub, and Imgur use HAProxy to deliver their websites efficiently.

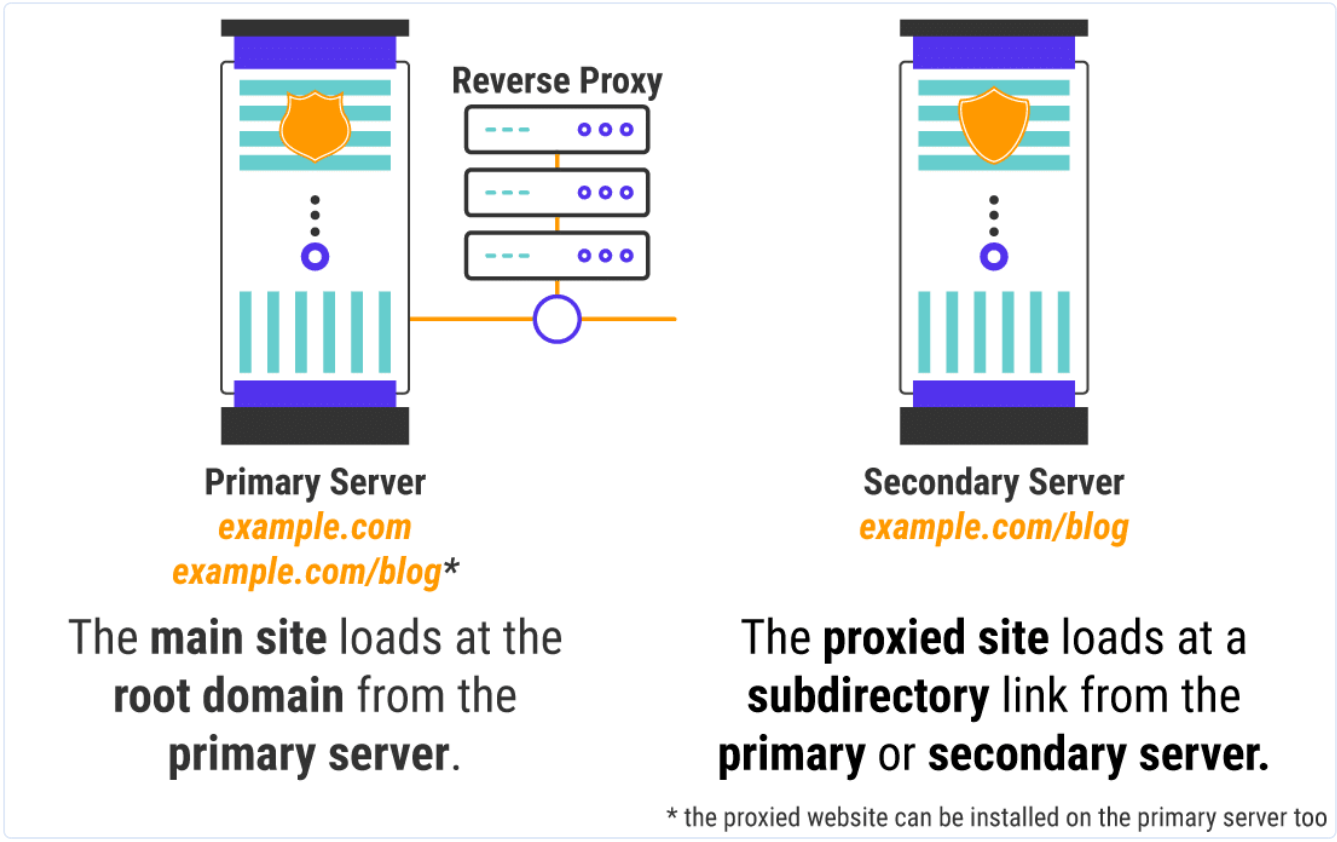
Discussing how to implement HAProxy is beyond the scope of this article, but you can refer to their documentation to understand how it works.

Note: Traefik and Envoy are two other open source alternatives to HAProxy. They're both high-performant reverse proxies and load balancers with many advanced features.

Some other popular reverse proxies are AWS Elastic Load Balancer, GLBC, DigitalOcean Load Balancer, and Google Cloud Load Balancer. For an exhaustive list of the top reverse proxies and load balancers in use today, you can check out Stackshare.io.

Reverse Proxy: Use Cases for WordPress Sites

There are mainly three use cases for employing a reverse proxy for WordPress sites, including sites hosted at Kinsta.



— Loading a ‘Main Site’ vs a ‘Proxied Site’

We’ll only use Nginx for this example, as it’s the most popular reverse proxy used for WordPress sites today. But the same basic principles will apply to other reverse proxies.

Reverse proxies are often challenging to install, configure, and support. For this reason, Kinsta offers a $50 monthly add-on subscription for each reverse proxy that you need help with setting up. You can reach out to Kinsta’s support team for further details.

1. Main and Proxied Sites Hosted on the Same Server

If both the main site and the proxied site are hosted on the same server, the main site can run on a WordPress installation, while a separate WordPress installation powers the proxied site.

As you’ll have access to both the sites and their shared web server, you can set up the reverse proxy rules for the main site, and then configure the proxied site to load from the

reverse proxy.

If you're hosting both these sites at Kinsta, then you can reach out to Kinsta's support team and request them to set up the reverse proxy for you. Here's the procedure you need to follow:

- Make sure that both the main site and the proxied site are hosted at Kinsta. If they're not, then you can migrate both sites to Kinsta's environment, either manually or by submitting a migration request.
- Open a support ticket and provide Kinsta's support team with a clear description of the domain configuration. I'll take approximately one business day to set up the reverse proxy.
- Kinsta will set up the relevant reverse proxy rules on the main site and configure the proxied site to load over the reverse proxy.

Here are the standard Nginx reverse proxy directives used by Kinsta to load a subdirectory site over a reverse proxy:

```
location ^~ /subfolder/ {
proxy_pass http://subfolder.domain.com;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}
```

In the above code, you need to replace the **`/subfolder/`** placeholder with the actual subdirectory name (e.g. **`/blog/`**, **`/shop/`**). Plus, the **`http://subfolder.domain.com`** subdomain should match the URL used to point the reverse proxy towards the proxied site.

The **`location`** directive includes caret and tilde symbols (**`^~`**) to tell Nginx that if it finds the string defined, it should stop searching for further matches and use the directives listed here. Learn more about Nginx's reverse proxy directives in its documentation.
Next, you need to configure the proxied site to load over the reverse proxy. Here are the

standard steps followed by Kinsta to configure the proxied site:

- Create a subdirectory at the path where the proxied site is loaded from. All the proxied website's files are moved to this subdirectory.
- Update the web server's configuration files to define the new subdirectory as the root directory for the proxied site. In addition, you need to add a rewrite rule to remove the subdirectory from the request URI for each incoming request.
- Update all URLs in the proxied site's database to match the live site URLs (e.g. **`example.com/blog`**).
- Edit the proxied site's **`wp-config.php`** file with the **`$_SERVER['HTTP_HOST']`** definition, pointing it to the main site's URL.
- If you're using an SSL certificate, then you need to define strict rules in the **`wp-config.php`** file to avoid redirection loops.

Note: A proxied site cannot create URLs that duplicate the same subdirectory under which the proxied site loads. For instance, a proxied site at **`example.com/blog`** cannot create a page or directory at **`example.com/blog/blog`**.

2. Only the Proxied Site Hosted on Your Server

If you only have access to the proxied site and its web server, then you need to contact the server admin of the main site and ask them to set up the reverse proxy rules for you.

To do that, you must follow the same steps outlined above, except in this case you must configure the rules on two different servers.

To host your proxied site with Kinsta, add a domain to the site which will point to the reverse proxy. Usually, subdomain suits this purpose (e.g. **`blog.example.com`**) to load the proxied site over a subdirectory link (e.g. **`example.com/blog`**).

After setting up your proxied site on Kinsta, you can contact Kinsta support team to configure the proxied site to load over a reverse proxy. At this time, our support team will require the real IP of your server in order to complete the setup process in a way that counts visits correctly. If you are unable to provide a static IP due to dynamic IP restrictions from certain providers (e.g. AWS CloudFront), your plan will be converted to a comparable bandwidth-based plan instead.

Lastly, setting up the reverse proxy on your server falls outside the scope of Kinsta support as only the server admin can take care of it.

3. Only the Main Site Hosted on Your Server

If you only have access to the main site and its web server, then you should set up the reverse proxy and configure its rules to load the proxied site from an external host. Installing and configuring the proxied site to load over the reverse proxy is the responsibility of the secondary server's admin.

Having your main site hosted at Kinsta will grant you access to Kinsta's support team. You can raise a support ticket with them to add the standard reverse proxy rules listed earlier in this article. You can also have any additional customizations added to those rules if needed.

In this scenario, you're fully responsible for configuring the proxied site to load it properly over the reverse proxy.

How to Set Up Nginx as a Reverse Proxy

If Kinsta doesn't host your website and you manage your servers, then you must set up the reverse proxy yourself and configure it to point towards the proxied site.

Depending on your web server's operating system, you can install Nginx differently. For Linux distributions, you can use various Nginx packages based on your Linux distribution's version.

In the example below, we've installed the primary site at **`example.com`** domain name, while the proxied WordPress site is installed at **`blog.domain.com`** subdomain. They're both powered by Apache on a web server running on Ubuntu 18.04. We'll install and configure Nginx as a reverse proxy on the main server.

To begin, access your server's terminal via SSH. Then use the **`apt-get`** command to update your distribution's packages list and install Nginx on your web server.

```
sudo apt update
sudo apt install nginx
```

Next, you need to configure Nginx to proxy requests for domains hosted on Apache. To do that, create a new virtual host file. Here, I'm using the **nano** editor to add the code, but you can use any code editor of your choice.

```
sudo nano /etc/nginx/sites-available/example.com.conf
```

Then set Nginx directives to forward requests to Apache by adding the following **`server {...}`** and **`location`** blocks:

```
server {
listen      80;
server_name example.com www.example.com;
index       index.php;
root        /var/www/example.com/public    # fallback for index.php
location / {
try_files $uri $uri/ /index.php?$query_string;
}location /blog {
proxy_pass http://blog.domain.com;proxy_http_version                 1.1;
proxy_cache_bypass                $http_upgrade;

# Proxy headers
proxy_set_header Upgrade          $http_upgrade;
```

```
    proxy_set_header Connection        "upgrade";
    proxy_set_header Host              $host;
    proxy_set_header X-Real-IP         $remote_addr;
    proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host  $host;
    proxy_set_header X-Forwarded-Port  $server_port;

    # Proxy timeouts
    proxy_connect_timeout              60s;
    proxy_send_timeout                 60s;
    proxy_read_timeout                 60s;
    }
```

In the code above, I'm defining a subdirectory **`example.com/blog`** link that will be served by the Apache server. Ensure that you use your proxied website's public IP address (or URL) in the **`proxy_pass`** directive. In my case, my proxied website is hosted on the **`blog.domain.com`** subdomain.

Note: Ensure that the proxied website is installed and ready to be served before you make any changes.

You can learn more about all the reverse proxy directives used here in Nginx's detailed index of directives.

Save the virtual host file. Then activate the new virtual host by creating a symlink for the files named **`example.com.conf`** in both the **`/etc/nginx/sites-available`** and the **`/etc/nginx/sites-enabled`** directories.

```
    sudo ln -s /etc/nginx/sites-available/example.com.conf /etc/nginx/sites-e
```

After that, test Nginx for any configuration errors.

```
sudo nginx -t
```

If there are no errors, reload Nginx to enforce the changes.

```
sudo systemctl reload nginx
```

You've successfully set up Nginx to work as a reverse proxy now. To confirm this, you can use the phpinfo() function to check the PHP variables loaded when you visit your proxied site.

Under the **SERVER_SOFTWARE** and **DOCUMENT_ROOT** PHP variables, you'll see that Apache serves this domain on the backend. But **HTTP_X_REAL_IP** and **HTTP_X_FORWARDED_FOR** PHP variables confirm that Nginx was used as a reverse proxy to forward the requests.

You can speed up serving your WordPress site over Nginx by using the fastcgi_cache and ngx_cache_purge modules. While the first module will cache your site, the second module will automatically purge the cache based on specific events (e.g. publishing or editing a WordPress post/page).

You can use the Nginx Cache Controller WordPress plugin to control Nginx's proxy server cache directly from your WordPress admin dashboard. If you're using a WordPress Multisite installation, then you can use the Nginx Helper plugin to do the same.

Check out Nginx's main documentation and Nginx WordPress setup guide for a detailed overview of how to work with Nginx and WordPress.

How To Set up Apache as a Reverse Proxy

Before you begin, make sure you have two websites up and running at **`example.com`** and **`blog.domain.com`**. The first website may or may not be a WordPress site, but the second one should be a WordPress site as it's used primarily to load the root domain's blog at **`example.com/blog`** subdirectory link.

Start configuring Apache by opening your server's terminal via SSH and enabling Apache's proxy module.

```
sudo a2enmod proxy proxy_http ssl
```

Running the above command will most likely restart Apache to reload the newly defined directives.

Next, edit your main server's virtual hosts file to create a reverse proxy. Here's the code you need to add:

```
<VirtualHost *>
DocumentRoot /var/www/app/public
SSLProxyEngine On    ProxyRequests off
ProxyPass /blog http://blog.domain.com
ProxyPassReverse /blog  http://blog.domain.com
</VirtualHost>
```

The ProxyPass directive will create a reverse proxy for the paths specified, while the ProxyPassReverse directive will intercept the HTTP response headers sent through this reverse proxy and rewrite them to match the Apache server.

After saving the file, you need to edit your **wp-config.php** file by adding the following code just before the line that asks you to stop editing.

```
# ProxyPass Settings
# overrides the variables below to ensure that any
# request to /blog/* subdirectory is taken care of properly
$_SERVER['REQUEST_URI'] = '/blog' . $_SERVER['REQUEST_URI'];
$_SERVER['SCRIPT_NAME'] = '/blog' . $_SERVER['SCRIPT_NAME'];
$_SERVER['PHP_SELF'] = '/blog' . $_SERVER['PHP_SELF'];
```

Finally, you need to update your WordPress site's database to add the configuration values for the **/blog** subdirectory link. You can do that by running the following SQL query:

```
UPDATE wp_options SET option_value = 'https://www.example.com/blog' WHERE
```

You should now be able to visit **https://www.example.com/blog** URL and have your WordPress site hosted at **http://blog.domain.com** subdomain load without changing its URL. You can continue using WordPress as usual to browse, write, edit, and manage your site.

Limitations of a Reverse Proxy

- A reverse proxy poses a significant security risk as it can read and change all the traffic passing through it. If you're passing HTTPS traffic through the reverse proxy, then it needs to decrypt and re-encrypt the passing data. This means that it must possess the private keys of the SSL/TLS certificate. Thus, if any malicious party can compromise

your reverse proxy, they can log passwords and inject malware into your websites.
- If you or your users can't access your main server directly, then using a reverse proxy can lead to a single point of failure. For example, if you're using a reverse proxy as a front to serve multiple domains, then its outage can lead to all the domains going offline simultaneously.
- If you're relying on a third-party reverse proxy (e.g. Cloudflare), then you're handing over your site's sensitive information to them. While they're trusted, you cannot predict what it may lead to.
- Restoring backups or pushing staging sites live on websites that load over a reverse proxy can cause the proxied site to stop loading properly.

Choosing Between a CDN and a Reverse Proxy

CDNs are an advanced form of reverse proxy with most of the configuration and maintenance taken care of by a third-party. They can provide amazing performance benefits to your WordPress site with minor effort from your end.

Not only do CDNs cache content and serve it swiftly to users, but they also reduce load on your origin servers, lower bandwidth costs, provide an additional layer of security, boost your site's SEO, and help you scale your website better.

> **Info**
>
> Kinsta CDN has very low TTFB and enhances your site's performance significantly. Every Kinsta account comes with a free tier of Kinsta CDN and you can set it up easily in seconds. For more information, you can refer to Kinsta CDN's comparison with a traditional CDN.

You'll notice that most of the benefits provided by CDNs are the same as those provided by reverse proxies. So, should you choose a CDN over a reverse proxy, or vice versa?

There’s no reason you must settle with just one. If you already have a reverse proxy installed, you’ll still see speed and performance gains from using a CDN. Both their caches layer well, and if you have any unique request handling needs (e.g. dynamic content, ecommerce), then you can configure it easily with some custom headers passed on by the CDN or the reverse proxy.

Summary

WordPress is highly flexible. You can use it as a blog, an ecommerce site, or even a Learning Management System. In most cases, you can customize WordPress to suit your unique requirements.

However, sometimes you may have to use a separate domain or a secondary server to host an additional site. As discussed earlier, it may be because of using different technology stacks for a big enterprise site or launching a WordPress blog for a pre-existing non-WordPress site.

A reverse proxy can help in both these cases, helping you get the most out of WordPress without giving up the main website and starting over.