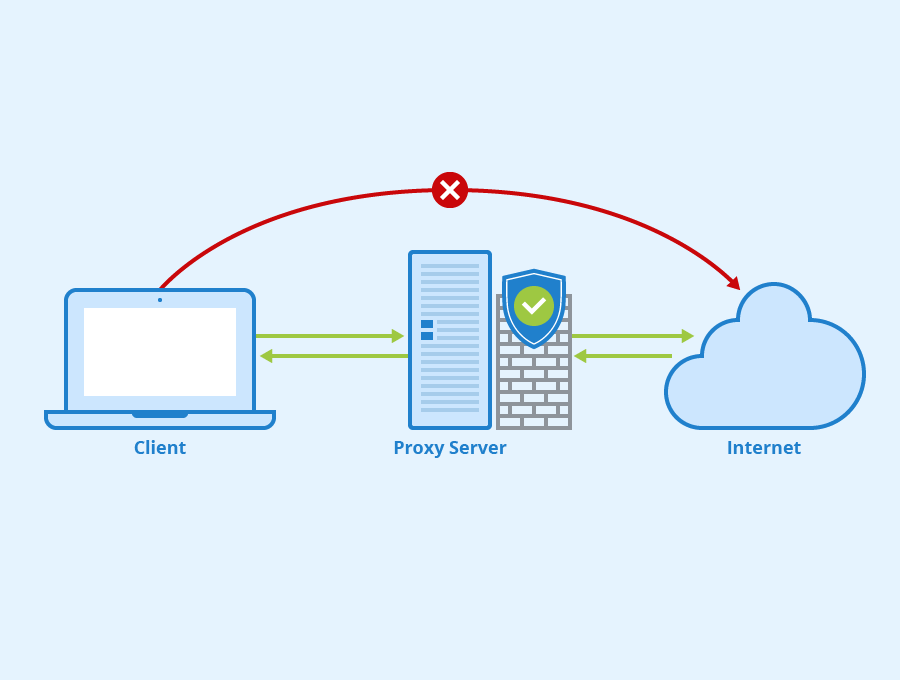
# What is a Proxy Server?

A proxy server acts as an intermediary between clients and other servers. It sits between a client device, such as a phone or laptop, and an origin server that holds the requested resources.

Instead of connecting directly to origin servers, clients connect to the proxy server first. The proxy server then makes requests to the origin on behalf of the client.



This indirection brings several benefits:

- \*\*Security\*\* - The proxy hides client IP addresses from origin servers

- \*\*Caching\*\* - The proxy can cache resources and serve them directly for better performance

- \*\*Filtering\*\* - The proxy can block malicious or unauthorized requests

- \*\*Load balancing\*\* - The proxy can distribute load across multiple origin servers

- \*\*Logging\*\* - The proxy can log all requests for analytics or auditing

In summary, a proxy server is an intermediary that sits between clients and origin servers. It acts on behalf of clients to get resources from servers and can provide security, performance and administrative benefits. Proxies are commonly used to control and monitor access to internal networks in businesses.

### References

https://www.cloudflare.com/learning/ddos/glossary/proxy-server/

<https://www.varonis.com/blog/proxy-server>

<https://vhost.vn/huong-dan-ket-noi-proxy-server-cua-vhost-de-tang-toc-do-truy-cap-internet-quoc-te/>

# Report on Proxy Server Code

## Table of Contents

- Allows handling of methods: GET, POST, HEAD

- Disallow: PUT, PATCH, DELETE, OPTIONS...

- Response to 403 code and 403 page content in 5.1, 5.3, 5.4

- Cache images

- Whitelisting

- Limited access from time to time

- Concurrent request handling

- Config files

- Reference link

## Allows handling of methods: GET, POST, HEAD

The proxy server code allows handling of the GET, POST and HEAD methods by checking the request method in the connect() function:

A screen shot of a computer code

Description automatically generated

If the method is GET, POST or HEAD, it will proceed with processing the request. Otherwise, it will send a 403 error response and close the connection.

This allows the proxy server to support common read-only methods like GET and HEAD, as well as POST for submitting forms. Other methods like PUT, PATCH and DELETE for modifying resources on the server are restricted.

## Disallow: PUT, PATCH, DELETE, OPTIONS...

As shown above, the connect() function explicitly checks for GET, POST and HEAD methods. Any other methods will trigger a 403 error response. This disallows methods like:

- PUT - For uploading/replacing resources

- PATCH - For partial modifications to resources

- DELETE - For deleting resources

- OPTIONS - For querying supported HTTP methods

Disallowing these methods improves security by making the proxy read-only. The proxy will reject any requests trying to modify resources on the origin servers.

## Response to 403 code and 403 page content in 5.1, 5.3, 5.4

The proxy server sends a 403 Forbidden response in several cases:

### 5.1. Unsupported HTTP method

If the request uses an unsupported HTTP method like PUT or DELETE, the connect() function will return a 403 error:

A computer screen shot of a computer code

Description automatically generated

The send\_error\_response() function sends a 403 response with custom HTML content loaded from error403.html:

A computer code with text

Description automatically generated with medium confidence

This allows customizing the 403 page shown to the client.

### 5.3. Whitelist violation

If the request domain is not in the whitelist configured in config.txt, the connect() function will return 403:

A computer screen shot of a code

Description automatically generated

Again this sends the custom 403 page.

### 5.4. Time limit violation

If the request falls outside the allowed time range in config.txt, 403 is returned:

A computer screen shot of a code

Description automatically generated

The consistent use of custom 403 responses allows admins to customize and explain the errors to clients.

## Cache images

The proxy server caches image files to improve performance. When the proxy receives an image response, it will save the image body to disk:

A black background with white text

Description automatically generated

The save\_image\_cache() function saves the image to a folder structure based on domain and file path:

A computer screen shot of a computer code

Description automatically generated

On subsequent requests for the same image, the proxy server will check this cache folder and serve the image directly without contacting the origin server. This avoids re-downloading frequent images on each request.

The proxy also embeds image URLs directly into HTML pages. When a HTML page references an image, the proxy will download the image and substitute the URL with a reference to the local cache:

A screenshot of a computer code

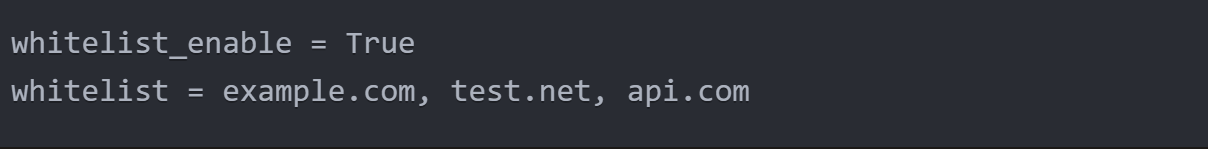
Description automatically generated

This further improves performance for pages with multiple images.

## Whitelisting

The proxy server implements a whitelist defined in config.txt to restrict access to approved domains only.

The whitelist is defined as a comma-separated list of allowed domains:



The connect() function checks each request against the whitelist:

A computer screen shot of a code

Description automatically generated

The is\_in\_white\_list() function checks if the requested domain matches any entry in the whitelist:

A computer screen shot of a code

Description automatically generated

This whitelist restricts the proxy to only approved sites, blocking all other sites. This prevents abuse of the proxy to access arbitrary URLs.

## Limited access from time to time

The proxy server can restrict access to certain time windows defined in config.txt.

The allowed timeframe is specified as a start and end time:

A black rectangular object with a white border

Description automatically generated

The connect() function checks the current time against these limits:

A screen shot of a computer code

Description automatically generated

The is\_in\_time\_limit() function compares the request time to the allowed range:

A computer screen shot of text

Description automatically generated

This allows restricting proxy access to business hours or other schedules. Requests outside the time limit receive a 403 error.

## Concurrent request handling

The proxy server handles multiple simultaneous connections using multithreading.

The main() function listens for incoming connections in a loop:

A screen shot of a computer

Description automatically generated

For each new client connection, it spawns a thread using Python's Thread class:

A computer screen shot of a black background

Description automatically generated

The connect() function is invoked in the new thread to handle the client request. This keeps the main thread free to accept new connections.

Using multithreading allows the proxy to process multiple client connections concurrently. Otherwise, it would only be able to serve one client at a time.

## Config files

The proxy server loads its configuration from a file called config.txt:

A screen shot of a computer

Description automatically generated

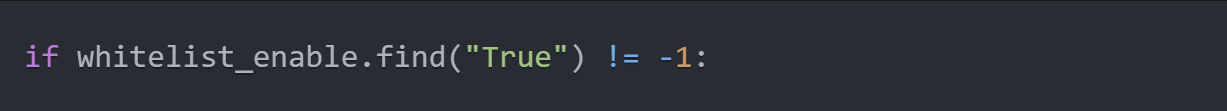
The read\_config\_file() function parses this file:

A screen shot of a computer code

Description automatically generated

It extracts configuration options like cache expiry, max receive size, whitelist domains etc.

The config options are parsed into variables that control the runtime behavior of the proxy. For example:



This allows dynamically configuring the proxy without changing code. The config can be modified by editing the text file.