

Reverse proxy



- [Security of subpath](#)
- [NGINX](#)
 - [Basic HTTP](#)
 - [HTTP with a subpath](#)
 - [HTTP with Unix socket](#)
 - [FastCGI with Unix socket](#)
 - [HTTPS](#)
 - [HTTPS with Certbot](#)
 - [HTTPS with manually installed certificates](#)
- [Apache](#)
 - [Basic HTTP](#)
 - [HTTP with a subpath](#)
 - [HTTPS](#)
 - [HTTPS with Certbot](#)
 - [HTTPS with manually installed certificates](#)
- [HAProxy](#)
 - [Basic HTTP](#)
 - [HTTPS](#)
 - [Redirection to SSL](#)
 - [SSL frontend](#)
 - [SSL backend](#)
 - [HTTPS with UNIX Socket](#)
 - [chroot](#)
 - [SSL Backend](#)
 - [HAProxy with UNIX socket using Puppet](#)
- [Caddy](#)
 - [HTTPS](#)
 - [HTTPS with a subpath](#)

- [Proxy Authentication](#)
 - [Docker / Container Registry](#)
-

Forgejo can live standalone, or behind a [reverse proxy](#).

You may want this for scenarios like:

- Subpath mapping.
If you want Forgejo at something like `https://example.com/code/` or `https://example.com/repositories/` instead of the default `https://example.com`.
- Port mapping.
If you want to run Forgejo on the standard port, and that port is already taken by another web server.
I.e. as `https://example.com` instead of as `https://example.com:3000`.
- Proxy authentication.
Using an external login service.
Forgejo usually does not need a proxy for this, as it can be configured to talk to many login services directly.
- rate limiting.
Fail2ban allows to rate-limit TCP connections, but with a load-balancer you can inspect the headers, perform User-Agent detection, match the information provided by an ACL
- advanced security settings.
Using a load balancer you can apply [Content Security policies](#), tweak your SSL ciphers, and configure a [Web Application Firewall](#).
- caching and resilience.
load-balancers offer both caching and robustness. For instance, Haproxy can handle millions of simultaneous connections, and caching alleviates the load on the application.

Forgejo does not need the help of a proxy to do HTTPS, it can do it directly.

Set in `SERVER` section of the configuration `PROTOCOL=https` and either set `CERT_FILE` and `KEY_FILE` or let Forgejo manage the certificates with `ENABLE_ACME=true`

Security of subpath

The security model of browsers assumes that interactions on the same-origin are safe.^[^1] By using a subpath, this assumption of the browser no longer holds, which can make using Forgejo insecure if that origin hosts content entered by a user under different sub-paths. Forgejo offers **no protection** to mitigate this scenario. For example, malicious content on the same origin could host CSRF attacks against Forgejo.

^[^1]: A good read about this security model is <https://educatedguesswork.org/posts/web-security-model-origin>.

NGINX

Basic HTTP

To set up a basic HTTP reverse proxy in nginx, create a file `forgejo.conf` in `/etc/nginx/conf.d` and add the following configuration:

```
server {  
    listen 80; # Listen on IPv4 port 80  
    listen [::]:80; # Listen on IPv6 port 80  
  
    server_name git.example.com; # Change this to the server domain name.  
    merge_slashes off; # Required to handle URL-encoded slashes  
  
    location / {  
        proxy_pass http://127.0.0.1:3000; # Port 3000 is the default Forgejo port  
  
        proxy_set_header Connection $http_connection;  
        proxy_set_header Upgrade $http_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;  
  
        client_max_body_size 512M;  
    }  
}
```

Make sure to reload/restart nginx after changing the configuration.

HTTP with a subpath

If you want to serve Forgejo on a subpath, e.g. on `http://example.com/code`, use the following configuration:

```
server {  
    listen 80; # Listen on IPv4 port 80  
    listen [::]:80; # Listen on IPv6 port 80
```

```

server_name example.com; # Change this to the server domain name.
merge_slashes off; # Required to handle URL-encoded slashes

location /code/ { # Replace /code here with your subpath
    proxy_pass http://127.0.0.1:3000/;

    proxy_set_header Connection $http_connection;
    proxy_set_header Upgrade $http_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;

    client_max_body_size 512M;
}
}

```

Make sure to set the Forgejo `ROOT_URL` configuration key to the URL *with* the subpath, otherwise links generated by Forgejo will be broken.

HTTP with Unix socket

To use a Unix socket instead of a TCP port, set the `PROTOCOL` option of your `[server]` block to `http+unix`, and `HTTP_ADDR` to the file name of the Unix socket to be used by Forgejo, `/run/forgejo.sock` for example. The `proxy_pass` from the [Basic HTTP](#) section should then be adapted to something like `proxy_pass http://unix:/run/forgejo.socket:/;`.

FastCGI with Unix socket

The venerable FastCGI protocol may be used as the protocol instead of HTTP, which improves security by being immune to the *desync attacks* (request smuggling) and *untrusted-header confusion* classes of security issues^[^1]. To use it, set the `PROTOCOL` option of your `[server]` block to `fcgi+unix`, and `HTTP_ADDR` to the file name of the Unix socket to be used by Forgejo, `/run/forgejo.sock` for example. You should then be able to use simply in your server block:

```

location / {
    include fastcgi_params;
    fastcgi_pass unix:/run/forgejo.sock;
}

```

[^1]: For a longer explanation, read

https://www.agwa.name/blog/post/fastcgi_is_the_better_protocol_for_reverse_proxies

HTTPS

When using a reverse proxy, it's usually easier to let the proxy handle HTTPS. It's easy to set up HTTPS on nginx.

HTTPS with Certbot

To set up HTTPS with Certbot, first set up an HTTP reverse proxy with the configuration above and ensure that it works as expected. To use HTTPS you need to have a domain name.

Then, install [certbot](#). When running certbot, select the domain name that your Forgejo instance is hosted under, and choose automatic installation. This should automatically set up HTTPS on port 443 and a redirect on the old port 80.

You may wish to change the `ROOT_URL` configuration key to the HTTPS protocol so links generated by Forgejo automatically use HTTPS.

HTTPS with manually installed certificates

If you have obtained certificates from elsewhere or have chosen not to let Certbot automatically install them, make the following changes to the configuration file:

Change the listening ports

Change the lines

```
listen 80;  
listen [::]:80;
```

to

```
listen 443 ssl;  
listen [::]:443 ssl;  
http2 on;
```

Add the SSL certificate information

Generate an SSL configuration at [mozilla](#), and add the SSL parameters to your configuration file. Make sure to replace the paths in the example with paths to your certificate files.

Add a redirect from HTTP

Outside the server block, add this redirection block:

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    location / {  
        return 301 https://$host$request_uri;  
    }  
}
```

This will redirect anyone visiting the HTTP site to the HTTPS site.

Apache

Basic HTTP

To set up a basic HTTP proxy in Apache, create a file `100-forgejo.conf` in `/etc/apache2/sites-available` and add the following configuration:

```
<VirtualHost *:80>  
    ServerName git.example.com  
  
    ProxyPreserveHost On  
    ProxyRequests off  
    AllowEncodedSlashes NoDecode  
    ProxyPass / http://127.0.0.1:3000/ nocanon  
</VirtualHost>
```

Next, enable the site with `a2ensite 100-forgejo.conf` and enable the proxy modules with `a2enmod proxy proxy_http`. Finally, restart the Apache server.

HTTP with a subpath

If you want to serve Forgejo on a subpath, e.g. on `http://example.com/code`, use the following configuration:

```
<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyRequests off
    AllowEncodedSlashes NoDecode
    ProxyPass /code http://127.0.0.1:3000/ nocanon # Change /code here to your desired
subpath.
</VirtualHost>
```

Make sure to set the Forgejo `ROOT_URL` configuration key to the URL *with* the subpath, otherwise links generated by Forgejo will be broken.

HTTPS

When using a reverse proxy, it's usually easier to let the proxy handle HTTPS. It's easy to set up HTTPS on Apache.

HTTPS with Certbot

To set up HTTPS with Certbot, first set up an HTTP reverse proxy with the configuration above and ensure that it works as expected. To use HTTPS you need to have a domain name.

Then, install [certbot](#). When running certbot, select the domain name that your Forgejo instance is hosted under, and choose automatic installation. This should automatically set up HTTPS on port 443 and a redirect on the old port 80.

You may wish to change the `ROOT_URL` configuration key to the HTTPS protocol so links generated by Forgejo automatically use HTTPS.

HTTPS with manually installed certificates

If you have obtained certificates from elsewhere or have chosen not to let certbot automatically install them, make the following changes to the configuration file:

Change the listening ports

Change `<VirtualHost *:80>` to `<VirtualHost *:443>`.

Add the SSL certificate information

Generate an SSL configuration at [mozilla](#), and add the SSL parameters to your configuration file. Make sure to replace the paths in the example with paths to your certificate files.

Add a redirect from HTTP

Outside the `VirtualHost *:443`, add this configuration:

```
<VirtualHost *:80>
    ServerName git.example.com

    RewriteEngine on
    RewriteCond %{SERVER_NAME} =git.example.com
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

This will redirect anyone visiting the HTTP site to the HTTPS site.

HAProxy

Basic HTTP

To setup HAProxy on port 80, without a virtualhost, you can add the following stanza to your `haproxy.cfg`:

```
listen forgejo_80
    bind :::80 v4v6
    mode http
    timeout connect 10s
    timeout client 30s
    timeout server 30s
    server forgejo 127.0.0.1:3000
```

HTTPS

To setup basic HTTPS proxying with HAProxy, you can add these blocks to your haproxy configuration:

Redirection to SSL

```
listen forgejo_80
    bind :::80 v4v6
```

```
mode http
timeout connect 10s
timeout client 30s
timeout server 30s
redirect scheme https code 301
```

SSL frontend

```
frontend forgejo_443
  bind :::443 v4v6 ssl crt /etc/haproxy_certs/forgejo.example.org.pem
  mode http
  option httplog
  option forwardfor
  timeout client 1m
  use_backend forgejo_443 if { ssl_fc_sni forgejo.example.org }
```

SSL backend

```
backend forgejo_443
  mode http
  timeout connect 10s
  timeout server 30s
  server forgejo 127.0.0.1:3000
```

HTTPS with UNIX Socket

A Unix socket has lower latency compared to TCP. When combined with HAProxy, it provides a highly responsive and excellent user interface experience.

We assume that:

- Redirection to SSL and the SSL frontend configuration remain unchanged from the TCP setup.
- you are running Forgejo as `git` user and HAProxy as `haproxy` user.
- the chroot environment is set to the directory `/var/lib/haproxy`.
- you have included the following settings in the `server` stanza of Forgejo:

```
[server]
PROTOCOL = http+unix
HTTP_ADDR = /var/lib/haproxy/forgejo/forgejo.sock
```

```
UNIX_SOCKET_PERMISSION = 660
```

```
...
```

Now you need to create a directory which can be accessed either by the chroot environment used by HAProxy, and by Forgejo.

```
install -o git -g haproxy -m 0770 -d /var/lib/haproxy/forgejo
```

Finally, you can add these blocks into your `haproxy.cfg`.

chroot

you include these lines in the `global` section of your haproxy configuration

```
global
  chroot  /var/lib/haproxy
  user    haproxy
  group   haproxy
  ...
```

SSL Backend

The backend configuration will be as follows:

```
backend forgejo_443
  mode http
  timeout connect 10s
  timeout server 30s
  server forgejo /forgejo/forgejo.sock tfo
```

note: *The Unix socket path is relative to the path of the chroot environment*

HAProxy with UNIX socket using Puppet

This configuration relies on [Puppetlabs HAProxy](#) module. This code sample is a compromise for the sake of the conciseness.

The Forgejo backend will be available only at the second execution of Puppet, unless you add a statement to create a user in advance, and you don't need to set the dependency for the 2 directories against the HAProxy class. You also need to push the SSL certificate, but all this goes far beyond the scope of this documentation.

```

file {
  default:
    notify => Service['haproxy'],
    require => [Class['haproxy'], User['git']];
  '/etc/haproxy_certs':
    ensure => directory,
    purge  => true,
    mode   => '0700',
    owner  => haproxy,
    group  => haproxy,
    recurse => true;
  '/var/lib/haproxy/forgejo':
    ensure => directory,
    mode   => '0770',
    owner  => git,
    group  => haproxy;
}

class { 'haproxy':
  package_ensure => $haproxy_version,
  global_options => {
    'log'           => "/dev/log local0\n log /dev/log local1 notice",
    'chroot'        => '/var/lib/haproxy',
    'maxconn'       => '150000',
    'user'          => 'haproxy',
    'group'         => 'haproxy',
    'stats'         => 'socket /var/run/haproxy.sock user root group sensu mode
660 level admin',
    'tune.ssl.default-dh-param' => '2048',
  },
  defaults_options => {
    'default-server' => 'init-addr libc,none',
    'log'            => 'global',
    'retries'        => '5',
    'option'         => ['redispatch', 'http-server-close', 'logasap'],
    'timeout'        => ['http-request 7s', 'connect 5s', 'check 9s'],
    'maxconn'        => '5000',
  };
}

```

```

haproxy::listen { 'forgejo_80':
    bind      => { ':::80' => ['v4v6'] },
    mode      => 'http',
    options => [
        { 'timeout' => 'connect 10s' },
        { 'timeout' => 'client 1m' },
        { 'timeout' => 'server 1m' },
        { 'redirect' => 'scheme https code 301' },
    ];
}

haproxy::frontend { 'forgejo_socket':
    bind      => { ':::443' => ['v4v6', 'ssl', 'crt',
'/etc/haproxy_certs/forgejo.example.com.pem'] },
    options => [
        {
            mode      => 'http',
            option => ['httplog', 'forwardfor'],
        },
        { 'timeout' => 'client 1m' },
        { 'use_backend' => "forgejo if { ssl_fc_sni forgejo.example.com }" },
    ];
}

haproxy::backend { 'forgejo_socket':
    options => [
        { 'mode' => 'http' },
        { 'timeout' => 'connect 10s' },
        { 'timeout' => 'server 1m' },
        { 'server' => 'forgejo /forgejo/forgejo.sock tfo' },
    ];
}

```

Caddy

HTTPS

To set up basic HTTPS proxy in Caddy with Caddyfile create a file `forgejo` in `/etc/caddy/conf.d` and add the following configuration:

```
git.example.com {  
    reverse_proxy 127.0.0.1:3000  
}
```

Caddy will automatically get certificates for the domain.

HTTPS with a subpath

If you want to serve Forgejo on a subpath, e.g. on `https://example.com/code`, use the following configuration:

```
example.com {  
    route /code/* {  
        uri strip_prefix /code  
        reverse_proxy 127.0.0.1:3000  
    }  
}
```

Make sure to set the Forgejo `ROOT_URL` configuration key to the URL with the subpath, otherwise links generated by Forgejo will be broken.

Proxy Authentication

Forgejo supports Reverse Proxy Header authentication, it will read headers as a trusted login user name or user email address. This hasn't been enabled by default, you can enable it with

```
[service]  
ENABLE_REVERSE_PROXY_AUTHENTICATION = true
```

The default login user name is in the `X-WEBAUTH-USER` header, you can change it via changing `[security].REVERSE_PROXY_AUTHENTICATION_USER` in `app.ini`. If the user doesn't exist, you can enable automatic registration with `ENABLE_REVERSE_PROXY_AUTO_REGISTRATION=true`.

The default login user email is `X-WEBAUTH-EMAIL`, you can change it via changing `[security].REVERSE_PROXY_AUTHENTICATION_EMAIL` in `app.ini`, this could also be disabled with `ENABLE_REVERSE_PROXY_EMAIL`

If set `ENABLE_REVERSE_PROXY_FULL_NAME=true`, a user full name expected in `X-WEBAUTH-FULLNAME` will be assigned to the user when auto creating the user. You can also change the header name with `[security].REVERSE_PROXY_AUTHENTICATION_FULL_NAME`.

You can also limit the reverse proxy's IP address range with `[security].REVERSE_PROXY_TRUSTED_PROXIES` which default value is `127.0.0.0/8,::1/128`. By `[security].REVERSE_PROXY_LIMIT`, you can limit trusted proxies level.

Notice: Reverse Proxy Auth doesn't support the API. You still need an access token or basic auth to make API requests.

Docker / Container Registry

The container registry uses a fixed sub-path `/v2` which can't be changed. Even if you deploy Forgejo with a different sub-path, `/v2` will be used by the `docker` client. Therefore you may need to add an additional route to your reverse proxy configuration.

📄 #2

📄 📄 📄 2026-10-31 13:34:12 UTC
📄 📄 📄 2026-10-31 13:34:13 UTC