rimelek/httpd24

Ákos Takács

Mar 28, 2023

CONTENTS:

Have you heard about Docker (Container engine)? Well, if you ever had problem with installing dependencies on the same machine for different projects, you should. If you need an Apache HTTPD web server, you can run it in seconds if you have Docker installed on your machine.

Unfortunately, you almost always need to customize it and create a new image with your own configuration. Just think of the need of PHP FPM or other modules disabled by default. This documentation is about an enhanced version of the HTTPD 2.4 image inherited from the official version. It provides a way to customize the configuration by environment variables. PHP FPM configuration was the first reason of creating this Docker image but now it is just an option among many.

CHAPTER

ONE

FEATURES

- Change some default server settings like ServerName, ServerAdmin, DocumentRoot and LogLevel
- Configure HTTPD to connect PHP FPM and increase timeout if you need.
- Turn SSL on or off easily. You can even generate a self-signed certificate to test locally and optionally redirect all HTTP requests to HTTPS.
- Do you use Reverse proxy front of the web server? You may want to log the original client IP address in the HTTPD container instead of the proxy's IP. You are free to use ProxyProtocol or set RemoteIPInternalProxy.
- Turn HTTP Basic authentication on or off. You can mount your own htpasswd file or generate it automatically from environment variable.
- Use HTTPD as reverse proxy to proxy one url to another.
- Mount your own configuration or copy that into your own image and enable or disable it.
- Run any custom command when httpd is ready to run but before it actually runs, so you can change anything before that.
- Enable/Disable modules or custom and built-in configurations without changing httpd.conf manually

CHAPTER

TWO

ENVIRONMENT VARIABLES

Some variables have boolean values. Those values can be set various ways.

Valid values for **true**: “true”, “1”, “on”, “yes”
Valid values for **false**: “false”, “0”, “off”, “no”

Each of them is case insensitive.

2.1 Variables

SRV_ADMIN
(default: “you@example.com”) “ServerAdmin” directive’s value.

SRV_ALLOW_OVERRIDE
(default: “false”) Set it to “true” to enable overriding some configuration from .htaccess. See SRV_ENABLE_MODULE to enable rewrite module.

SRV_AUTH
(default: “false”) Set this to “true” if you need HTTP Basic authentication using an htpasswd file. Without SRV_AUTH_USERS you need to create the htpasswd file manually.

SRV_AUTH_NAME
(default: “Private Area”) This is the value of AuthName directive.

SRV_AUTH_USERS

(default: “”) Set the users and their passwords line by line where the user and the password are separated by only one space.

Example:

SRV_AUTH_USERS=”admin1 secretpassword\
admin2 secretpassword2”

SRV_DOCROOT
(default: “/usr/local/apache2/htdocs”) The value after DocumentRoot directive. This variable helps you change the document root and it’s Directory block.

SRV_ENABLE_CONFIG
(default: “”) Pass the name of configurations you want to enable separated by space. There are 3 type of configuration.

- Official configuration files in “conf/extra” directory. The name of them starts with the name of configuration and ends with “.conf”. For example: httpd-default.conf. SRV_ENABLE_CONFIG=”httpd-default httpd-ssl” will enable httpd-default.conf and httpd-ssl.conf. You will rarely need it.
- Custom configurations of rimelek/httpd24. These are in “conf/custom-extra” directory. To enable these configurations you would need to prefix them with “@”. Example: SRV_ENABLE_CONFIG=”@php” These are controlled by environment variables so you don’t need to enable them this way.
- Your custom configuration can be saved to “conf/custom-extra/user” directory. If you want to enable them, prefix the configuration names with “@user/”. Example: SRV_ENABLE_CONFIG=”@user/my-conditional-redirect”. Of course, the filename must ends with “.conf”. See SRV_DISABLE_CONFIG to disable configurations.

SRV_ENABLE_MODULE

(default: “”) Just like SRV_ENABLE_CONFIG, but this is to enable modules like rewrite.

Example: SRV_ENABLE_MODULE=”rewrite dav ssl”

To disable modules enabled by default use SRV_DISABLE_MODULE.

SRV_DISABLE_CONFIG

(default: “”) Disable configurations enabled by default. It will comment out the configuration’s include.

Example: SRV_DISABLE_CONFIG=”proxy-html”

SRV_DISABLE_MODULE:

(default: “”) If you want to disable every module you don’t really need, this variable is for you.

Example: SRV_DISABLE_MODULE=”autoindex”

SRV_LOGLEVEL
(default: “warn”) The server’s log level

SRV_NAME
(default: “localhost.localdomain”) To change ServerName directive’s value.

SRV_PHP
(default: “false”) If it is “true”, requested PHP scripts will be sent to PHP FPM. PHP FPM’s hostname is “php” by default. In the same docker network you can name the PHP container as “php” or use “php” as the name of Docker Compose service. You can customize the hostname by SRV_PHP_HOST.

SRV_PHP_DISABLE_REUSE
(default: “true”) When you upgrade the PHP container, the container’s IP can change. To make sure HTTPD use the new IP to connect, you need to set “disablereuse=on” for FCGI proxy. To turn it off, set this variable’s value to “true”. You can use “on” and “off” too.

SRV_PHP_HOST
(default: “php”) The hostname of your PHP FPM service.

SRV_PHP_PORT
(default: 9000) The port number of PHP FPM service.

SRV_PHP_TIMEOUT
(default: “60”) PHP FPM proxy timeout in seconds. When you want to use XDebug or other debug tools, or your PHP scripts running too long, you may want to increase the timeout.

SRV_PROXY_FORWARD_FROM
(default: “/”) You can use HTTPD as a reverse proxy to proxy some requests to another website. By default, all requests will be forwarded. You can change it, if you want: SRV_PROXY_FORWARD_FROM=”/admin/”

SRV_PROXY_FORWARD_TO:
(default: “”) If it is not an empty string, httpd will forward all requests from SRV_PROXY_FORWARD_FROM to the given URL.

SRV_PROXY_PROTOCOL
(default: “false”) Behind a reverse proxy HTTPD will log the proxy’s IP address and your PHP scripts need to check X-Forwarded-For or X-Client-Ip headers to determine the client IP. The Proxy Protocol helps you avoid it.

SRV_REVERSE_PROXY_CLIENT_IP_HEADER
(default: “X-Forwarded-For”) In case of reverse proxy front of the HTTPD, this is the HTTP header in which the proxy stores the real client IP. When SRV_REVERSE_PROXY_DOMAIN is set, you can see the real IP in server log.

SRV_REVERSE_PROXY_DOMAIN
(default: “”) If it is not empty, it can be an alternative to the proxy protocol. If you cannot use proxy protocol or you don’t want to, you can tell httpd the domain, IP or IP range of the reverse proxy server. Example: SRV_REVERSE_PROXY_DOMAIN=”haproxy” or SRV_REVERSE_PROXY_DOMAIN=”10.42.0.0/16”

SRV_SSL
(default: “false”) Whether you need SSL or not. You can set it to “true” and httpd will look for the SSL private key and certificate. To tell HTTPD where those files are, see SRV_SSL_CERT and SRV_SSL_NAME.

SRV_SSL_AUTO
(default: “false”) In case of “true”, the container will generate a self-signed certificate for you before starting the server. Note that it can take some minutes and won’t be valid. Use it only for testing purposes.

SRV_SSL_CERT
(default: “/usr/local/apache2/ssl/${CERT_NAME}.crt”) The path of the ssl certificate inside the container. If you set SRV_SSL_LETSENCRYPT to “true” it will change the value of SRV_SSL_CERT to “/etc/letsencrypt/live/${CERT_NAME}/fullchain.pem”.

SRV_SSL_KEY
(default: “/usr/local/apache2/ssl/${CERT_NAME}.key”) The path of the SSL private key file inside the container. If you set SRV_SSL_LETSENCRYPT to “true” it will change the value of SRV_SSL_CERT to “/etc/letsencrypt/live/${CERT_NAME}/privkey.pem”.

SRV_SSL_LETSENCRYPT
(default: “false”) In case of “true”, it changes the the value of SRV_SSL_CERT and SRV_SSL_KEY to be compatible with Let’s Encrypt.

SRV_SSL_NAME
(default: “ssl”) The default SSL certificate is /usr/local/apache2/ssl/ssl.crt and the private key file is /usr/local/apache2/ssl/ssl.crt. You can change the file name by setting changing the value of this variable:

Example: SRV_SSL_NAME=”custom”

Then the file names will be changed to custom.crt and custom.key

There are some other variables too just to be compatible with other docker containers like Nginx Proxy.

VIRTUAL_HOST
Nginx proxy and Hosts gen use this to determine the domain name of the containers. If you set this variable and SRV_SSL_NAME, CERT_NAME, SRV_NAME are not set, it can be used to set the name of ssl certificate. See SRV_SSL_CERT and SRV_SSL_KEY.

CERT_NAME
Nginx Proxy use this variable to choose the SSL certificate for the backend container if the certificate’s name is not the same as the domain.

CHAPTER

THREE

EXAMPLES

3.1 Using SSL certificate

3.1.1 With Let's Encrypt without custom certificate name

```
docker run -d \
   --env SRV_SSL="true" \
   --env SRV_LETSENCRYPT="true" \
   --env SRV_NAME=YOURDOMAIN \
   -v /etc/letsencrypt:/etc/letsencrypt \
   -p 443:443 \
   rimelek/httpd24:2.0
```

3.1.2 With Let's Encrypt and custom certificate name

```
docker run -d \
   --env SRV_SSL="true" \
   --env SRV_LETSENCRYPT="true" \
   --env SRV_NAME=YOURDOMAIN \
   --env CERT_NAME=CUSTOMCERTNAME \
   -v /etc/letsencrypt:/etc/letsencrypt \
   -p 443:443 \
   rimelek/httpd24:2.0
```

3.1.3 You can mount custom certificate two ways

```
docker run -d \
   --env SRV_SSL="true" \
   -v /path/to/custom.key:/usr/local/apache2/ssl.key \
   -v /path/to/custom.crt:/usr/local/apache2/ssl.crt \
   -p 443:443 \
   rimelek/httpd24:2.0
```

or

```
docker run -d \
   --env SRV_SSL="true" \
   --env SRV_CERT=/ssl.crt \
   --env SRV_CERT_KEY=/ssl.key \
   -v /path/to/custom.key:/ssl.key \
   -v /path/to/custom.crt:/ssl.crt \
   -p 443:443 \
   rimelek/httpd24:2.0
```

3.2 HTTP Authentication

```
docker run --rm -i rimelek/httpd24 /bin/bash -c "htpasswd -nb YOURUSERNAME YOURPASSWORD"␣
↪>> .htpasswd
docker run -d \
   -v `pwd`/.htpasswd:/usr/local/apache2/.htpasswd \
   --env SRV_AUTH="true" \
   -p 80:80 \
   rimelek/httpd24:2.0
```

or just generate htpasswd inside the container

```
docker run -d \
   --env SRV_AUTH="true" \
   --env SRV_AUTH_USERS="admin1 password1\
 admin2 password2"
   -p 80:80 \
   rimelek/httpd24:2.0
```

3.3 Simplest way to use PHP-FPM

3.3.1 Legacy way

```
mkdir -p src
echo "<?php phpinfo(); " > src/index.php
docker run -d \
   -v $PWD/src:/usr/local/apache2/htdocs \
   --name php \
   php:7.1-fpm
docker run -d \
   --volumes-from php \
   --env SRV_PHP="true" \
   -p "80:80" \
   --link php \
   rimelek/httpd24:2.0
```

3.3.2 Recommended way

```
mkdir -p src
echo "<?php phpinfo(); " > src/index.php
docker network create phptest
docker run -d \
   -v $PWD/src:/usr/local/apache2/htdocs \
   --name php \
   --network phptest \
   php:7.1-fpm
docker run -d \
   --volumes-from php \
   --env SRV_PHP=1 \
   -p "80:80" \
   --network phptest \
   rimelek/httpd24:2.0
```

3.3.3 Reusing the network of the HTTPD container

```
mkdir -p src
echo "<?php phpinfo(); " > src/index.php
docker run -d \
   -v $PWD/src:/usr/local/apache2/htdocs \
   --name php \
   -p "80:80" \
   php:7.1-fpm
docker run -d \
   --volumes-from php \
   --env SRV_PHP=1 \
   --env SRV_PHP_HOST=localhost \
   --network container:php \
   rimelek/httpd24:2.0
```

3.4 Use the rewrite engine

```
docker run -d \
    -v $PWD/src:/usr/local/apache2/htdocs \
    -p 80:80 \
    --env SRV_ENABLE_MODULES="rewrite" \
    --env SRV_ALLOW_OVERRIDE="true" \
    rimelek/httpd24:2.0
```

3.5 Forward the admin page to another site:

```
docker run -d \
    -v $PWD/src:/usr/local/apache2/htdocs \
    -p 80:80 \
    --env SRV_PROXY_FORWARD_FROM="/admin/" \
    --env SRV_PROXY_FORWARD_TO="http://admin.mysite.tld/" \
    rimelek/httpd24:2.0
```

3.6 Get real client IP behind reverse proxy

3.6.1 Proxy protocol

```
docker run -d \
    -v $PWD/src:/usr/local/apache2/htdocs \
    -p 80:80 \
    --env SRV_PROXY_PROTOCOL="true" \
    rimelek/httpd24:2.0
```

3.6.2 Client IP header

```
docker run -d \
    -v $PWD/src:/usr/local/apache2/htdocs \
    -p 80:80 \
    --env SRV_REVERSE_PROXY_CLIENT_IP_HEADER="X-Forwarded-For" \
    --env SRV_REVERSE_PROXY_DOMAIN="haproxy" \
    rimelek/httpd24:2.0
```