Wikiprint Book

Title: Trac with FastCGI

Subject: TechTIDE-Wiki - TracFastCgi

Version: 1

Date: 11/25/25 04:57:33

Table of Contents

Frac with FastCGI	3
Apache configuration	3
Set up with mod_fastcgi	3
Set up with mod_fcgid	3
Alternative environment setup	4
Cherokee Configuration	4
Lighttpd Configuration	4
LiteSpeed Configuration	7
Nginx Configuration	8

Trac with FastCGI

<u>FastCGI</u> interface allows Trac to remain resident much like with <u>mod_python</u> or <u>mod_wsgi</u>. It is faster than external CGI interfaces which must start a new process for each request. Additionally, it is supported by a much wider variety of web servers.

Note that unlike mod_python, FastCGI supports <u>Apache SuEXEC</u>, ie run with different permissions than the web server runs with. mod_wsgi supports the WSGIDaemonProcess with user / group parameters to achieve the same effect.

Note for Windows: Trac's FastCGI does not run under Windows, as Windows does not implement Socket.fromfd, which is used by _fcgi.py. If you want to connect to IIS, you may want to try _AJP/_ISAPI.

Apache configuration

There are two FastCGI modules commonly available for Apache: mod_fastcgi and mod_fcgid (preferred). The latter is more up-to-date.

The following sections focus on the FCGI specific setup, see also TracModWSGI for configuring the authentication in Apache.

Regardless of which cgi module is used, be sure the web server has executable permissions on the cgi-bin folder. While FastCGI will throw specific permissions errors, mod_fcgid will throw an ambiguous error if this has not been done: Connection reset by peer: mod_fcgid: error reading data from FastCGI server.

Set up with mod_fastcgi

mod_fastcgi uses FastCgiIpcDir and FastCgiConfig directives that should be added to an appropriate Apache configuration file:

```
# Enable fastcgi for .fcgi files
# (If you're using a distro package for mod_fcgi, something like
# this is probably already present)
<IfModule mod_fastcgi.c>
  AddHandler fastcgi-script .fcgi
  FastCgiIpcDir /var/lib/apache2/fastcgi
</IfModule>
LoadModule fastcgi_module /usr/lib/apache2/modules/mod_fastcgi.so
```

Setting FastCgilpcDir is optional if the default is suitable. Note that the LoadModule line must be after the IfModule group.

Configure ScriptAlias or similar options as described in TracCg; but calling trac.fcgi instead of trac.cgi.

Add the following to the Apache configuration file (below the FastCgilpcDir line) if you intend to set up the TRAC_ENV as an overall default:

```
FastCgiConfig -initial-env TRAC_ENV=/path/to/env/trac
```

Alternatively, you can serve multiple Trac projects in a directory by adding this:

```
FastCgiConfig -initial-env TRAC_ENV_PARENT_DIR=/parent/dir/of/projects
```

You can also specify the PYTHON_EGG_CACHE environment variable using a second -initial-env directive:

```
FastCgiConfig -initial-env TRAC_ENV=/var/lib/trac \
-initial-env PYTHON_EGG_CACHE=/var/lib/trac/plugin-cache
```

Set up with mod_fcgid

Configure ScriptAlias (see TracCgi for details), but call trac.fcgi instead of trac.cgi:

```
ScriptAlias /trac /path/to/www/trac/cgi-bin/trac.fcgi/
```

Note the slash at the end.

To set up Trac environment for mod_fcgid it is necessary to use DefaultInitEnv directive. It cannot be used in Directory or Location context, so if you need to support multiple projects, try the alternative environment setup below:

```
DefaultInitEnv TRAC_ENV /path/to/env/trac/
```

Alternative environment setup

A better method to specify the path to the Trac environment is to embed the path into trac.fcgi script itself. That doesn't require configuration of the server environment variables, works for both FastCgi modules as well as for lighttpd and CGI:

```
import os
os.environ['TRAC_ENV'] = "/path/to/projectenv"

or:
import os
os.environ['TRAC_ENV_PARENT_DIR'] = "/path/to/project/parent/dir"
```

With this method different projects can be supported by using different .fcgi scripts with different ScriptAliases.

See Ethis fcgid example config which uses a ScriptAlias directive with trac.fcgi with a trailing / like this:

```
ScriptAlias / /srv/tracsite/cgi-bin/trac.fcgi/
```

Cherokee Configuration

Configuring Cherokee with Trac is straightforward, if you spawn Trac as an SCGI process. You can either start it manually, or better yet, automatically by letting Cherokee spawn the server whenever it is down.

First set up an information source in cherokee-admin with a local interpreter:

```
Host:
localhost:4433

Interpreter:
/usr/bin/tracd -single-env -daemonize -protocol=scgi -hostname=localhost -port=4433 /path/to/project/
```

If the port was not reachable, the interpreter command would be launched. Note that, in the definition of the information source, you will have to manually launch the spawner if you use a *Remote host* as *Information source* instead of a *Local interpreter*.

After doing this, we will just have to create a new rule managed by the SCGI handler to access Trac. It can be created in a new virtual server, trac.example.net for instance, and will only need two rules. The **default** one will use the SCGI handler associated to the previously created information source. The second rule will be there to serve the few static files needed to correctly display the Trac interface. Create it as *Directory rule* for */common* and just set it to the *Static files* handler and with a *Document root* that points to the appropriate files: \$TRAC_LOCAL/htdocs/ (where \$TRAC_LOCAL is a directory defined by the user or the system administrator to place local Trac resources).

Note: If the tracd process fails to start up, and Cherokee displays a 503 error page, you might be missing the <u>python-flup</u> package (<u>#9903</u>). Python-flup is a dependency which provides Trac with SCGI capability. You can install it on Debian based systems with:

```
sudo apt-get install python-flup
```

Lighttpd Configuration

The FastCGI front-end was developed primarily for use with alternative webservers, such as Lighttpd.

Lighttpd is a secure, fast, compliant and very flexible web-server that has been optimized for high-performance environments. It has a very low memory footprint compared to other web servers and takes care of CPU load.

For using trac.fcgi(prior to 0.11) / fcgi_frontend.py (0.11) with Lighttpd add the following to your lighttpd.conf:

Note that you will need to add a new entry to fastegi.server for each separate Trac instance that you wish to run. Alternatively, you may use the TRAC_ENV_PARENT_DIR variable instead of TRAC_ENV as described above, and you may set one of the two in trac.fcgi instead of in lighttpd.conf using bin-environment, as in the section above on Apache configuration.

Note that Lighttpd has a bug related to 'SCRIPT_NAME' and 'PATH_INFO' when the uri of fastcgi.server is '/' instead of '/trac' in this example (see #2418). This is fixed in Lighttpd 1.5, and under Lighttpd 1.4.23 or later the workaround is to add "fix-root-scriptname" => "enable" as a parameter of fastcgi.server.

For using two projects with lighttpd add the following to your lighttpd.conf:

Note that the field values are different. If you prefer setting the environment variables in the .fcgi scripts, then copy/rename trac.fcgi, eg to first.fcgi and second.fcgi, and reference them in the above settings. Note that the above will result in different processes in any event, even if both are running from the same trac.fcgi script.

Note: The order in which the server.modules are loaded is very important: if mod_auth is not loaded before mod_fastcgi, then the server will fail to authorize to the user.

For authentication you should enable mod_auth in lighttpd.conf 'server.modules', select auth.backend and auth rules:

```
server.modules = (
...
   "mod_auth",
...
)

auth.backend = "htpasswd"

# Separated password files for each project
# See "Conditional Configuration" in
# http://trac.lighttpd.net/trac/file/branches/lighttpd-merge-1.4.x/doc/configuration.txt

$HTTP["url"] =~ "^/first/" {
   auth.backend.htpasswd.userfile = "/path/to/projenv-first/htpasswd.htaccess"
}
```

Note that Lighttpd (v1.4.3) stops if the password file doesn't exist.

Note that Lighttpd doesn't support 'valid-user' in versions prior to 1.3.16.

Conditional configuration is also useful for mapping static resources, ie serving out images and CSS directly instead of through FastCGI:

```
# Aliasing functionality is needed
server.modules += ("mod_alias")
# Set up an alias for the static resources
alias.url = ("/trac/chrome/common" => "/usr/share/trac/htdocs")
# Use negative lookahead, matching all requests that ask for any resource under /trac, EXCEPT in
# /trac/chrome/common, and use FastCGI for those
$HTTP["url"] =~ "^/trac(?!/chrome/common)" {
# Even if you have other fastcgi.server declarations for applications other than Trac, do NOT use += here
fastcgi.server = ("/trac" =>
                  ("trac" =>
                    ("socket" => "/tmp/trac-fastcgi.sock",
                     "bin-path" => fcgi_binary,
                     "check-local" => "disable",
                     "bin-environment" =>
                       ("TRAC_ENV" => "/path/to/projenv")
                  )
```

The technique can be easily adapted for use with multiple projects by creating aliases for each of them, and wrapping the fastcgi.server declarations inside conditional configuration blocks.

Also there is another way to handle multiple projects and it uses TRAC_ENV_PARENT_DIR instead of TRAC_ENV as well as global authentication:

Changing date/time format also supported by lighttpd over environment variable LC_TIME:

For details about languages specification see <a>TracFaq question 2.13.

Other important information like the $\underline{\text{mapping static resources advices}}$ are useful for non-fastcgi specific installation aspects.

Relaunch Lighttpd and browse to http://yourhost.example.org/trac to access Trac.

Note about running Lighttpd with reduced permissions: If nothing else helps and trac.fcgi doesn't start with Lighttpd settings server.username = "www-data", server.groupname = "www-data", then in the bin-environment section set PYTHON_EGG_CACHE to the home directory of www-data or some other directory accessible to this account for writing.

LiteSpeed Configuration

The FastCGI front-end was developed primarily for use with alternative webservers, such as LiteSpeed.

LiteSpeed web server is an event-driven asynchronous Apache replacement designed from the ground-up to be secure, scalable, and operate with minimal resources. LiteSpeed can operate directly from an Apache config file and is targeted for business-critical environments.

- i. Please make sure you have a working install of a Trac project. Test install with "tracd" first.
- ii. Create a Virtual Host for this setup. From now on we will refer to this vhost as TracVhost. For this tutorial we will be assuming that your Trac project will be accessible via:

```
http://yourdomain.com/trac/
```

iii. Go "TracVhost \rightarrow External Apps" tab and create a new "External Application":

```
Name: MyTracFCGI
Address: uds://tmp/lshttpd/mytracfcgi.sock
```

```
Max Connections: 10
Environment: TRAC_ENV=/fullpathto/mytracproject/ <--- path to root folder of trac project
Initial Request Timeout (secs): 30
Retry Timeout (secs): 0
Persistent Connection Yes
Connection Keepalive Timeout: 30
Response Bufferring: No
Auto Start: Yes
Command: /usr/share/trac/cgi-bin/trac.fcgi <--- path to trac.fcgi
Back Log: 50
Instances: 10
```

iv. Optional: If you need to use htpasswd based authentication. Go to "TracVhost → Security" tab and create a new security Realm:

```
DB Type: Password File

Realm Name: MyTracUserDB <--- any name you wish and referenced later

User DB Location: /fullpathto/htpasswd <--- path to your htpasswd file
```

If you don't have a htpasswd file or don't know how to create the entries within one, go to http://sherylcanter.com/encrypt.php, to generate the user:password combos.

v. Go to "PythonVhost \rightarrow Contexts" and create a new FCGI Context:

Modify /fullpathto/mytracproject/conf/trac.ini:

```
#find/set base_rul, url, and link variables
base_url = http://yourdomain.com/trac/ <--- base url to generate correct links to
url = http://yourdomain.com/trac/ <--- link of project
link = http://yourdomain.com/trac/ <--- link of graphic logo</pre>
```

vii. Restart LiteSpeed: lswsctrl restart, and access your new Trac project at http://yourdomain.com/trac/.

Nginx Configuration

Nginx is able to communicate with FastCGI processes, but can not spawn them. So you need to start FastCGI server for Trac separately.

i. Nginx configuration with basic authentication handled by Nginx - confirmed to work on 0.6.32

```
# and remove the auth entries below if you want Trac to enforce
         # authorization where appropriate instead of needing to authenticate
         # for accessing the whole site.
         # (Or ``~ location /some/prefix(/.*)``.)
         location ~ (/.*) {
            auth_basic
                                 "trac realm";
            auth_basic_user_file /home/trac/htpasswd;
            # socket address
            fastcgi_pass unix:/home/trac/run/instance.sock;
            # python - wsgi specific
            fastcgi_param HTTPS on;
            ## WSGI REQUIRED VARIABLES
            # WSGI application name - trac instance prefix.
            # (Or ``fastcgi_param SCRIPT_NAME /some/prefix``.)
            fastcgi_param SCRIPT_NAME
                                          "";
            fastcgi_param PATH_INFO
                                            $1;
            \ensuremath{\#\#} WSGI NEEDED VARIABLES - trac warns about them
            fastcgi_param REQUEST_METHOD $request_method;
            fastcgi_param SERVER_NAME
                                            $server_name;
            fastcgi_param SERVER_PORT
                                            $server_port;
            fastcgi_param SERVER_PROTOCOL $server_protocol;
            fastcgi_param QUERY_STRING
                                           $query_string;
            # For Nginx authentication to work - do not forget to comment these
            # lines if not using Nginx for authentication
            fastcgi_param REMOTE_USER
                                            $remote_user;
            # for ip to work
            fastcgi_param REMOTE_ADDR
                                            $remote_addr;
            \# For attchments to work
            fastcgi_param CONTENT_TYPE
                                             $content_type;
            fastcgi_param CONTENT_LENGTH $content_length;
     }
ii. Modified trac.fcgi:
  #!/usr/bin/env python
  import os
  sockaddr = '/home/trac/run/instance.sock'
  os.environ['TRAC_ENV'] = '/home/trac/instance'
  trv:
      from trac.web.main import dispatch_request
      import trac.web._fcgi
      fcgiserv = trac.web._fcgi.WSGIServer(dispatch_request,
          bindAddress = sockaddr, umask = 7)
      fcgiserv.run()
  except SystemExit:
     raise
  except Exception, e:
     print 'Content-Type: text/plain \r\n',
     print 'Oops...'
```

You can copy this whole location to ``location [/some/prefix](/login)``

```
print
print 'Trac detected an internal error:'
print
print e
print
import traceback
import StringIO
tb = StringIO.StringIO()
traceback.print_exc(file=tb)
print tb.getvalue()
```

iii. Reload nginx and launch trac.fcgi:

```
trac@trac.example ~ $ ./trac-standalone-fcgi.py
```

The above assumes that:

- There is a user named 'trac' for running Trac instances and keeping Trac environments in its home directory
- /home/trac/instance contains a Trac environment
- /home/trac/htpasswd contains authentication information

/home/trac/run is owned by the same group the Nginx runs under

- and if your system is Linux the /home/trac/run has setgid bit set (chmod g+s run)
- and patch from <u>■#7239</u> is applied, or you'll have to fix the socket file permissions every time

Unfortunately Nginx does not support variable expansion in fastcgi_pass directive. Thus it is not possible to serve multiple Trac instances from one server block.

If you worry enough about security, run Trac instances under separate users.

Another way to run Trac as a FCGI external application is offered in <u>■#6224</u>.

See also: <u>TracGuide</u>, <u>TracInstall</u>, <u>ModWSGI</u>, <u>CGI</u>, <u>ModPython</u>, <u>■TracNginxRecipe</u>