Installing Trac as CGI

Please note that using Trac via CGI is the slowest deployment method available. It is slower than <u>mod python</u>, <u>FastCGI</u> and even <u>IIS/AJP</u> on Windows.

CGI script is the entrypoint that web-server calls when a web-request to an application is made. To generate the trac.cgi script run:

trac-admin /path/to/env deploy /path/to/www/trac

trac.cgi will be in the cgi-bin folder inside the given path. Make sure it is executable by your web server. This command also copies static resource files to a htdocs directory of a given destination.

Apache web-server configuration

In <u>Apache</u> there are two ways to run Trac as CGI:

- 1. Use a ScriptAlias directive that maps an URL to the trac.cgi script (recommended)
- 2. Copy the trac.cgi file into the directory for CGI executables used by your web server (commonly named cgi-bin). You can also create a symbolic link, but in that case make sure that the FollowSymLinks option is enabled for the cgi-bin directory.

To make Trac available at http://yourhost.example.org/trac add ScriptAlias directive to Apache configuration file, changing trac.cgi path to match your installation:

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

Note that this directive requires enabled mod_alias module.

If you're using Trac with a single project you need to set its location using the TRAC_ENV environment variable:

```
<Location "/trac">
SetEnv TRAC_ENV "/path/to/projectenv"
</Location>
```

Or to use multiple projects you can specify their common parent directory using the TRAC_ENV_PARENT_DIR variable:

```
<Location "/trac">
SetEnv TRAC_ENV_PARENT_DIR "/path/to/project/parent/dir"
</Location>
```

Note that the SetEnv directive requires enabled mod_env module. It is also possible to set TRAC_ENV in trac.cgi. Just add the following code between "try:" and "from trac.web ...":

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

Or for TRAC_ENV_PARENT_DIR:

import os

Installing Trac as CGI

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

If you are using the <u>Apache suEXEC</u> feature please see <u>http://trac.edgewall.org/wiki/ApacheSuexec</u>.

On some systems, you *may* need to edit the shebang line in the trac.cgi file to point to your real Python installation path. On a Windows system you may need to configure Windows to know how to execute a .cgi file (Explorer -> Tools -> Folder Options -> File Types -> CGI).

Mapping Static Resources

Out of the box, Trac will pass static resources such as style sheets or images through itself. For a CGI setup this is **highly undesirable**, because this way CGI script is invoked for documents that could be much more efficiently served directly by web server.

Web servers such as <u>Apache</u> allow you to create ?Aliases? to resources, giving them a virtual URL that doesn't necessarily reflect the layout of the servers file system. We already used this capability by defining a ScriptAlias for the CGI script. We also can map requests for static resources directly to the directory on the file system, avoiding processing these requests by CGI script.

There are two primary URL paths for static resources - /chrome/common and /chrome/site. Plugins can add their own resources usually accessible by /chrome/plugin path, so its important to override only known paths and not try to make universal /chrome alias for everything.

Add the following snippet to Apache configuration **before** the ScriptAlias for the CGI script, changing paths to match your deployment:

```
Alias /trac/chrome/common /path/to/trac/htdocs
<Directory "/path/to/www/trac/htdocs">
    Order allow,deny
    Allow from all
</Directory>
```

Note that we mapped /trac part of the URL to the trac.cgi script, and the path /chrome/common is the path you have to append to that location to intercept requests to the static resources.

For example, if Trac is mapped to /cgi-bin/trac.cgi on your server, the URL of the Alias should be /cgi-bin/trac.cgi/chrome/common.

Similarly, if you have static resources in a project's htdocs directory (which is referenced by /chrome/site URL in themes), you can configure Apache to serve those resources (again, put this **before** the ScriptAlias for the CGI script, and adjust names and locations to match your installation):

```
Alias /trac/chrome/site /path/to/projectenv/htdocs
<Directory "/path/to/projectenv/htdocs">
    Order allow,deny
    Allow from all
</Directory>
```

Alternatively to hacking /trac/chrome/site, you can directly specify path to static resources using htdocs_location configuration option in trac.ini:

[trac]

Apache web-server configuration

htdocs_location = http://yourhost.example.org/trac-htdocs

Trac will then use this URL when embedding static resources into HTML pages. Of course, you still need to make the Trac htdocs directory available through the web server at the specified URL, for example by copying (or linking) the directory into the document root of the web server:

\$ ln -s /path/to/www/trac/htdocs /var/www/yourhost.example.org/trac-htdocs

Note that in order to get this htdocs directory, you need first to extract the relevant Trac resources using the deploy command of <u>TracAdmin</u>:

```
deploy <directory>
```

Extract static resources from Trac and all plugins

Adding Authentication

The simplest way to enable authentication with Apache is to create a password file. Use the htpasswd program to create the password file:

```
$ htpasswd -c /somewhere/trac.htpasswd admin
New password: <type password>
Re-type new password: <type password again>
Adding password for user admin
```

After the first user, you dont need the "-c" option anymore:

```
$ htpasswd /somewhere/trac.htpasswd john
New password: <type password>
Re-type new password: <type password again>
Adding password for user john
```

See the man page for htpasswd for full documentation.

After you've created the users, you can set their permissions using TracPermissions.

Now, you'll need to enable authentication against the password file in the Apache configuration:

```
<Location "/trac/login">
AuthType Basic
AuthName "Trac"
AuthUserFile /somewhere/trac.htpasswd
Require valid-user
</Location>
```

If you're hosting multiple projects you can use the same password file for all of them:

```
<LocationMatch "/trac/[^/]+/login">
AuthType Basic
AuthName "Trac"
AuthUserFile /somewhere/trac.htpasswd
Require valid-user
</LocationMatch>
```

Mapping Static Resources

For better security, it is recommended that you either enable SSL or at least use the ?digest? authentication scheme instead of ?Basic?. Please read the <u>Apache HTTPD documentation</u> to find out more. For example, on a Debian 4.0r1 (etch) system the relevant section in apache configuration can look like this:

```
<Location "/trac/login">
LoadModule auth_digest_module /usr/lib/apache2/modules/mod_auth_digest.so
AuthType Digest
AuthName "trac"
AuthDigestDomain /trac
AuthUserFile /somewhere/trac.htpasswd
Require valid-user
</Location>
```

and you'll have to create your .htpasswd file with htdigest instead of htpasswd as follows:

htdigest /somewhere/trac.htpasswd trac admin

where the "trac" parameter above is the same as AuthName above ("Realm" in apache-docs).

See also: <u>TracGuide</u>, <u>TracInstall</u>, <u>TracModWSGI</u>, <u>TracFastCgi</u>, <u>TracModPython</u>