

4.6 Git on the Server - Smart HTTP

Smart HTTP

We now have authenticated access through SSH and unauthenticated access through `git://`, but there is also a protocol that can do both at the same time. Setting up Smart HTTP is basically just enabling a CGI script that is provided with Git called `git-http-backend` on the server. This CGI will read the path and headers sent by a `git fetch` or `git push` to an HTTP URL and determine if the client can communicate over HTTP (which is true for any client since version 1.6.6). If the CGI sees that the client is smart, it will communicate smartly with it; otherwise it will fall back to the dumb behavior (so it is backward compatible for reads with older clients).

Let's walk through a very basic setup. We'll set this up with Apache as the CGI server. If you don't have Apache setup, you can do so on a Linux box with something like this:

```
$ sudo apt-get install apache2 apache2-utils
$ a2enmod cgi alias env
```

This also enables the `mod_cgi`, `mod_alias`, and `mod_env` modules, which are all needed for this to work properly.

You'll also need to set the Unix user group of the `/srv/git` directories to `www-data` so your web server can read- and write-access the repositories, because the Apache instance running the CGI script will (by default) be running as that user:

```
$ chgrp -R www-data /srv/git
```

Next we need to add some things to the Apache configuration to run the `git-http-backend` as the handler for anything coming into the `/git` path of your web server.

```
SetEnv GIT_PROJECT_ROOT /srv/git
SetEnv GIT_HTTP_EXPORT_ALL
ScriptAlias /git/ /usr/lib/git-core/git-http-backend/
```

If you leave out `GIT_HTTP_EXPORT_ALL` environment variable, then Git will only serve to unauthenticated clients the repositories with the `git-daemon-export-ok` file in them, just like the Git daemon did.

Finally you'll want to tell Apache to allow requests to `git-http-backend` and make writes be authenticated somehow, possibly with an `Auth` block like this:

```
<Files "git-http-backend">
  AuthType Basic
  AuthName "Git Access"
  AuthUserFile /srv/git/.htpasswd
  Require expr !(%{QUERY_STRING} -strmatch '*service=git-receive-pack*' || %{REQUEST_URI} =~ m#/git-receive-pack$#)
  Require valid-user
</Files>
```

That will require you to create a `.htpasswd` file containing the passwords of all the valid users. Here is an example of adding a "schacon" user to the file:

```
$ htpasswd -c /srv/git/.htpasswd schacon
```

There are tons of ways to have Apache authenticate users, you'll have to choose and implement one of them. This is just the simplest example we could come up with. You'll also almost certainly want to set this up over SSL so all this data is encrypted.

We don't want to go too far down the rabbit hole of Apache configuration specifics, since you could well be using a different server or have different authentication needs. The idea is that Git comes with a CGI called `git-http-backend` that when invoked will do all the negotiation to send and receive data over HTTP. It does not implement any authentication itself, but that can easily be controlled at the layer of the web server that invokes it. You can do this with nearly any CGI-capable web server, so go with the one that you know best.

Note For more information on configuring authentication in Apache, check out the Apache docs here: <https://httpd.apache.org/docs/current/howto/auth.html>

[prev](#) | [next](#)