NAME

git-http-backend − Server side implementation of Git over HTTP

SYNOPSIS

git http−backend

DESCRIPTION

A simple CGI program to serve the contents of a Git repository to Git clients accessing the repository over http:// and https:// protocols. The program supports clients fetching using both the smart HTTP protocol and the backwards−compatible dumb HTTP protocol, as well as clients pushing using the smart HTTP protocol. It also supports Git's more−efficient "v2" protocol if properly configured; see the discussion of **GIT_PROTOCOL** in the ENVIRONMENT section below.

It verifies that the directory has the magic file "git−daemon−export−ok", and it will refuse to export any Git directory that hasn't explicitly been marked for export this way (unless the **GIT_HTTP_EXPORT_ALL** environmental variable is set).

By default, only the **upload−pack** service is enabled, which serves *git fetch−pack* and *git ls−remote* clients, which are invoked from *git fetch*, *git pull*, and *git clone*. If the client is authenticated, the **receive−pack** service is enabled, which serves *git send−pack* clients, which is invoked from *git push*.

SERVICES

These services can be enabled/disabled using the per−repository configuration file:

http.getanyfile
: This serves Git clients older than version 1.6.6 that are unable to use the upload pack service. When enabled, clients are able to read any file within the repository, including objects that are no longer reachable from a branch but are still present. It is enabled by default, but a repository can disable it by setting this configuration item to **false**.

http.uploadpack
: This serves *git fetch−pack* and *git ls−remote* clients. It is enabled by default, but a repository can disable it by setting this configuration item to **false**.

http.receivepack
: This serves *git send−pack* clients, allowing push. It is disabled by default for anonymous users, and enabled by default for users authenticated by the web server. It can be disabled by setting this item to **false**, or enabled for all users, including anonymous users, by setting it to **true**.

URL TRANSLATION

To determine the location of the repository on disk, *git http−backend* concatenates the environment variables PATH_INFO, which is set automatically by the web server, and GIT_PROJECT_ROOT, which must be set manually in the web server configuration. If GIT_PROJECT_ROOT is not set, *git http−backend* reads PATH_TRANSLATED, which is also set automatically by the web server.

EXAMPLES

All of the following examples map **http://$hostname/git/foo/bar.git** to **/var/www/git/foo/bar.git**.

Apache 2.x
: Ensure mod_cgi, mod_alias, and mod_env are enabled, set GIT_PROJECT_ROOT (or DocumentRoot) appropriately, and create a ScriptAlias to the CGI:

```
SetEnv GIT_PROJECT_ROOT /var/www/git
SetEnv GIT_HTTP_EXPORT_ALL
ScriptAlias /git/ /usr/libexec/git−core/git−http−backend/

# This is not strictly necessary using Apache and a modern version of
# git−http−backend, as the webserver will pass along the header in the
# environment as HTTP_GIT_PROTOCOL, and http−backend will copy that into
```

```
# GIT_PROTOCOL. But you may need this line (or something similar if you
# are using a different webserver), or if you want to support older Git
# versions that did not do that copying.
#
# Having the webserver set up GIT_PROTOCOL is perfectly fine even with
# modern versions (and will take precedence over HTTP_GIT_PROTOCOL,
# which means it can be used to override the client's request).
SetEnvIf Git−Protocol ".*" GIT_PROTOCOL=$0
```

To enable anonymous read access but authenticated write access, require authorization for both the initial ref advertisement (which we detect as a push via the service parameter in the query string), and the receive−pack invocation itself:

```
RewriteCond %{QUERY_STRING} service=git−receive−pack [OR]
RewriteCond %{REQUEST_URI} /git−receive−pack$
RewriteRule ^/git/ − [E=AUTHREQUIRED:yes]

<LocationMatch "^/git/">
    Order Deny,Allow
    Deny from env=AUTHREQUIRED

    AuthType Basic
    AuthName "Git Access"
    Require group committers
    Satisfy Any
    ...
</LocationMatch>
```

If you do not have **mod_rewrite** available to match against the query string, it is sufficient to just protect **git−receive−pack** itself, like:

```
<LocationMatch "^/git/.*/git−receive−pack$">
    AuthType Basic
    AuthName "Git Access"
    Require group committers
    ...
</LocationMatch>
```

In this mode, the server will not request authentication until the client actually starts the object negotiation phase of the push, rather than during the initial contact. For this reason, you must also enable the **http.receivepack** config option in any repositories that should accept a push. The default behavior, if **http.receivepack** is not set, is to reject any pushes by unauthenticated users; the initial request will therefore report **403 Forbidden** to the client, without even giving an opportunity for authentication.

To require authentication for both reads and writes, use a Location directive around the repository, or one of its parent directories:

```
<Location /git/private>
    AuthType Basic
    AuthName "Private Git Access"
    Require group committers
    ...
</Location>
```

To serve gitweb at the same url, use a ScriptAliasMatch to only those URLs that *git http−backend* can handle, and forward the rest to gitweb:

```
ScriptAliasMatch \
	"(?x)^/git/(.*/(HEAD | \
			info/refs | \
			objects/(info/[^/]+ | \
				[0−9a−f]{2}/[0−9a−f]{38} | \
				pack/pack−[0−9a−f]{40}\.(pack|idx)) | \
			git−(upload|receive)−pack))$" \
	/usr/libexec/git−core/git−http−backend/$1

ScriptAlias /git/ /var/www/cgi−bin/gitweb.cgi/
```

To serve multiple repositories from different **gitnamespaces**(7) in a single repository:

```
SetEnvIf Request_URI "^/git/([^/]*)" GIT_NAMESPACE=$1
ScriptAliasMatch ^/git/[^/]*(.*) /usr/libexec/git−core/git−http−backend/storage.git$1
```

Accelerated static Apache 2.x

Similar to the above, but Apache can be used to return static files that are stored on disk. On many systems this may be more efficient as Apache can ask the kernel to copy the file contents from the file system directly to the network:

```
SetEnv GIT_PROJECT_ROOT /var/www/git

AliasMatch ^/git/(.*/objects/[0−9a−f]{2}/[0−9a−f]{38})$          /var/www/git/$1
AliasMatch ^/git/(.*/objects/pack/pack−[0−9a−f]{40}.(pack|idx))$ /var/www/git/$1
ScriptAlias /git/ /usr/libexec/git−core/git−http−backend/
```

This can be combined with the gitweb configuration:

```
SetEnv GIT_PROJECT_ROOT /var/www/git

AliasMatch ^/git/(.*/objects/[0−9a−f]{2}/[0−9a−f]{38})$          /var/www/git/$1
AliasMatch ^/git/(.*/objects/pack/pack−[0−9a−f]{40}.(pack|idx))$ /var/www/git/$1
ScriptAliasMatch \
	"(?x)^/git/(.*/(HEAD | \
			info/refs | \
			objects/info/[^/]+ | \
			git−(upload|receive)−pack))$" \
	/usr/libexec/git−core/git−http−backend/$1
ScriptAlias /git/ /var/www/cgi−bin/gitweb.cgi/
```

Lighttpd

Ensure that **mod_cgi**, **mod_alias**, **mod_auth**, **mod_setenv** are loaded, then set **GIT_PROJECT_ROOT** appropriately and redirect all requests to the CGI:

```
alias.url += ( "/git" => "/usr/lib/git−core/git−http−backend" )
$HTTP["url"] =~ "^/git" {
	cgi.assign = ("" => "")
	setenv.add−environment = (
		"GIT_PROJECT_ROOT" => "/var/www/git",
		"GIT_HTTP_EXPORT_ALL" => ""
```

```
        )
}
```

To enable anonymous read access but authenticated write access:

```
$HTTP["querystring"] =~ "service=git−receive−pack" {
        include "git−auth.conf"
}
$HTTP["url"] =~ "^/git/.*/git−receive−pack$" {
        include "git−auth.conf"
}
```

where **git−auth.conf** looks something like:

```
auth.require = (
        "/" => (
                "method" => "basic",
                "realm" => "Git Access",
                "require" => "valid−user"
               )
)
# ...and set up auth.backend here
```

To require authentication for both reads and writes:

```
$HTTP["url"] =~ "^/git/private" {
        include "git−auth.conf"
}
```

ENVIRONMENT

git http−backend relies upon the **CGI** environment variables set by the invoking web server, including:

- PATH_INFO (if GIT_PROJECT_ROOT is set, otherwise PATH_TRANSLATED)
- REMOTE_USER
- REMOTE_ADDR
- CONTENT_TYPE
- QUERY_STRING
- REQUEST_METHOD

The **GIT_HTTP_EXPORT_ALL** environmental variable may be passed to *git−http−backend* to bypass the check for the "git−daemon−export−ok" file in each repository before allowing export of that repository.

The **GIT_HTTP_MAX_REQUEST_BUFFER** environment variable (or the **http.maxRequestBuffer** config variable) may be set to change the largest ref negotiation request that git will handle during a fetch; any fetch requiring a larger buffer will not succeed. This value should not normally need to be changed, but may be helpful if you are fetching from a repository with an extremely large number of refs. The value can be specified with a unit (e.g., **100M** for 100 megabytes). The default is 10 megabytes.

Clients may probe for optional protocol capabilities (like the v2 protocol) using the **Git−Protocol** HTTP header. In order to support these, the contents of that header must appear in the **GIT_PROTOCOL** environment variable. Most webservers will pass this header to the CGI via the **HTTP_GIT_PROTOCOL** variable, and **git−http−backend** will automatically copy that to **GIT_PROTOCOL**. However, some webservers may be more selective about which headers they'll pass, in which case they need to be

configured explicitly (see the mention of **Git−Protocol** in the Apache config from the earlier EXAMPLES section).

The backend process sets GIT_COMMITTER_NAME to *$REMOTE_USER* and GIT_COMMITTER_EMAIL to *${REMOTE_USER}@http.${REMOTE_ADDR}*, ensuring that any reflogs created by *git−receive−pack* contain some identifying information of the remote user who performed the push.

All **CGI** environment variables are available to each of the hooks invoked by the *git−receive−pack*.

GIT

Part of the **git**(1) suite