Howto set up Git over Https with Apache on Ubuntu Server 14.04

Welcome,

In this post we will look at setting up Git over https (git-http-backend) with Apache on a Ubuntu Server 14.04 LTS. We will require users to be authenticated with basic auth before accessing the central git-repositories, both when reading from and writing to any repository.

Updated guide for Ubuntu Server 18.04 LTS here: https://github.com/jbilander/HowTos/wiki/Setting-up-Git-over-https-with-Apache-on-Ubuntu-Server-18.04-LTS

Let's start by installing git on our Ubuntu system:

```
root@ubuntu01:~# apt-get upgrade
```

```
root@ubuntu01:~# apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-bzr git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 3 not upgraded.
Need to get 3,346 kB of archives.
After this operation, 21.6 MB of additional disk space will be used.
```

Setting up Git over Https with Apache is basically just enabling a CGI-script that is provided with Git called git-http-backend on the server.

We can find out the location of git-http-backend on our system by searching for it:

```
root@ubuntu01:/etc/apache2/mods-enabled# find / -name git-http-backend
/usr/lib/git-core/git-http-backend
```

Now, in order for the git-http-backend to work properly with Apache we need to enable these modules: *mod_cgi*, *mod_alias*, and *mod_env*. On my system I already have *mod_alias* and *mod_env* up and running so I only need to enable *mod_cgi*:

```
root@ubuntu01:/etc/apache2# a2enmod cgi
Enabling module cgi.
```

For security purposes, it is generally a good practice to execute CGI-scripts as a different user than the web server user, hence we create the unprivileged user and group called git, we will also install and make use of the apache2 suexec packages:

First, you create a git group:

```
root@ubuntu01:/opt# groupadd git
```

You can easily restrict the git user to only doing Git activities with a limited shell tool called git-shell that comes with Git. If you set this as your git user's login shell, then the git user can't have normal shell access to your server. To use this, specify git-shell instead of bash or csh for your user's login shell. To do so, you must first add git-shell to /etc/shells if it's not already there:

```
root@ubuntu01:/opt# more /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/tmux
/usr/bin/screen
```

So git-shell is not enabled, let's enable it, first we need to find out the path:

```
root@ubuntu01:/opt# find / -name git-shell
/usr/bin/git-shell
/usr/lib/git-core/git-shell
```

Okay, so let's add the /usr/bin/git-shell to /etc/shells

```
root@ubuntu01:/opt# vi /etc/shells

root@ubuntu01:/opt# more /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/tmux
/usr/bin/screen
/usr/bin/git-shell
```

Now create a home directory for the git user at /opt/git

```
root@ubuntu01:/opt# mkdir git
```

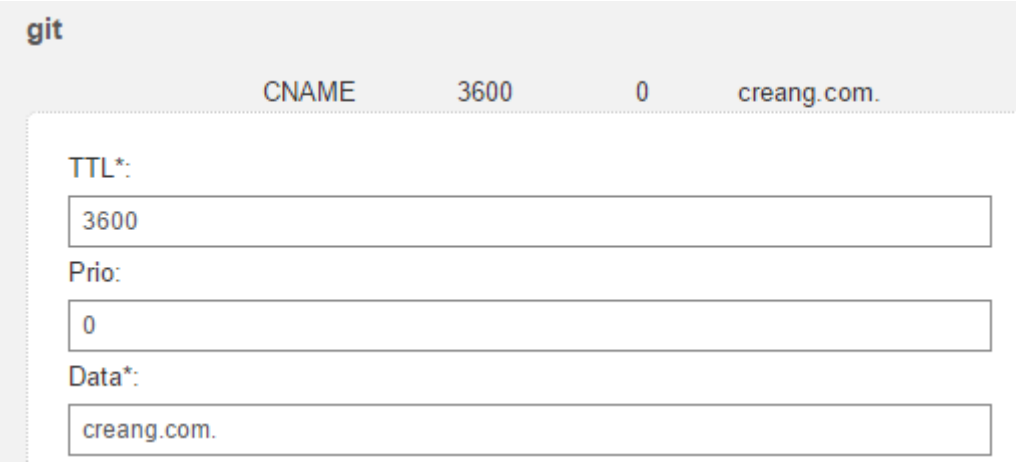
Now create a git user. We'll make this user a member of the git group, with a home directory of /opt/git, and with a shell of /usr/bin/git-shell

```
root@ubuntu01:/opt# useradd -s /usr/bin/git-shell -g git -d /opt/git git
```

Make the git user and group the owner of the /opt/git folder:

```
root@ubuntu01:/opt# chown git:git git/
```

Now, I've decided to use a subdomain called git with my domain so that the url will look similar to this: https://git.example.com For this to work I need to add a subdomain record to my DNS-configuration. I will use a CNAME-record for this.



With the host-command I can now verify that the new record does resolve in dns:

```
root@ubuntu01:/opt/git# host -t CNAME git.creang.com
git.creang.com is an alias for creang.com.
```

Now, let's set up a VirtualHost in Apache for this subdomain:

```
root@ubuntu01:/opt/git# vi /etc/apache2/sites-enabled/vhosts-default.conf
```

```
        <VirtualHost *:443>
                ServerName git.creang.com
                DocumentRoot /opt/git
                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined
                <Directory /opt/git>
                        Options ExecCGI Indexes FollowSymLinks
                        AllowOverride All
                        Require all granted
                </Directory>
                SSLEngine on
                SSLCertificateFile /etc/apache2/ssl-stuff/myCert.crt
                SSLCertificateKeyFile /etc/apache2/ssl-stuff/myKey.key
                SSLCertificateChainFile /etc/apache2/ssl-stuff/myCA.crt
                <Location />
                        AuthType Basic
                        AuthName "Private Git Access"
                        AuthUserFile /opt/git/.htpasswd
                        Require valid-user
                </Location>
                SuexecUserGroup git git
                ScriptAlias /git /var/www/sbin/git-http-backend-wrapper
        </VirtualHost>
```

Now install apache2-suexec:

```
root@ubuntu01:/etc/apache2# apt-get install apache2-suexec
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-suexec-pristine
The following NEW packages will be installed:
  apache2-suexec apache2-suexec-pristine
```

Enable suEXEC Support so that the git user and group can be used when running the CGI-Script:

```
root@ubuntu01:/etc/apache2# a2enmod suexec
Enabling module suexec.
```

To work with the SuExec security model a wrapper script needs to be create that configures the environment when SuExec executes the script. The script simply sets the correct environment variable and calls git-http-backend.

```
root@ubuntu01:/var/www# mkdir sbin
```

```
root@ubuntu01:/var/www# vi ./sbin/git-http-backend-wrapper

#!/bin/bash
PATH_INFO=$SCRIPT_URL
GIT_PROJECT_ROOT=/opt/git
REMOTE_USER=$REDIRECT_REMOTE_USER
export GIT_HTTP_EXPORT_ALL=true
/usr/lib/git-core/git-http-backend
```

Change owner to git user and group on this folder and script and make it executable:

```
root@ubuntu01:/var/www# chown -R git:git sbin/

root@ubuntu01:/var/www# chmod 755 ./sbin/git-http-backend-wrapper
```

Now create the htpasswd-file. This will require the apache-utils package, install if not installed already:

```
root@ubuntu01:/etc/apache2# apt-get install apache2-utils
```

Create the file, replace with your user:

```
root@ubuntu01:/etc/apache2# htpasswd -c /opt/git/.htpasswd jbilander
New password:
Re-type new password:
Adding password for user jbilander
```

Make the git user and group owner of this file:

```
root@ubuntu01:/etc/apache2# chown git:git /opt/git/.htpasswd
```

Restart Apache:

```
root@ubuntu01:/etc/apache2# service apache2 restart
```

Now create a repository in /opt/git

```
root@ubuntu01:/opt/git# git init --bare --shared=group projectA.git
Initialized empty shared Git repository in /opt/git/projectA.git/
```

Set the git user and group as the owner, recursively, of this repo:

```
root@ubuntu01:/opt/git# chown -R git.git projectA.git/
```

Set the repo to http.receivepack true:

```
root@ubuntu01:/opt/git# cd projectA.git/

root@ubuntu01:/opt/git/projectA.git# git config --file config
http.receivepack true
```

The config file will now look like this:

```
root@ubuntu01:/opt/git/projectA.git# more config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = true
        sharedrepository = 1
[receive]
        denyNonFastforwards = true
[http]
        receivepack = true
```

Now lets access and clone this repo from a client over https. I will do this from the command line just to show how, you may prefer to use a gui client here like SmartGit:

```
C:\Projects>git.exe clone https://git.creang.com/git/projectA.git
myProjectA.git
Cloning into 'myProjectA.git'...
Username for 'https://git.creang.com': jbilander
Password for 'https://jbilander@git.creang.com':
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

I used another name here for the repository folder just for instructional purposes. You can leave that out if you want the same name on the client side as on the server

side. Please note, if you are using a self-signed-certificate you can ignore any warning with this configuration on the client side:

```
git config --global http.sslVerify false
```

Maybe even better you can add your certificate to your trust store. I will not show how to do that here though.

Let's try to add a new file and commit and push to the server repo:

Create a new file:

```
C:\Projects\myProjectA.git>notepad test.txt
```

```
C:\Projects\myProjectA.git>git.exe add test.txt
```

```
C:\Projects\myProjectA.git>git.exe commit -m "a first commit"

[master (root-commit) da37104] a first commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

Push to the remote repository on the server:

```
C:\Projects\myProjectA.git>git.exe push origin master

Username for 'https://git.creang.com': jbilander
Password for 'https://jbilander@git.creang.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 217 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://git.creang.com/git/projectA.git
 * [new branch]      master -> master
```

All done! and it works 🙂 Now time to grab some coffee...

2015-07-14 jbilander HowToForge

3 thoughts on "Howto set up Git over Https with Apache on Ubuntu Server 14.04"

anne

2015-12-17 at 04:00

Thank you !!!

Simon Huggins

2017-04-04 at 21:20

Really excellent blog post – thanks!
Saved me hours of work, and gave me some good starting points for further investigations too!

John

2018-10-22 at 05:31

Thanks so much