WebDAV: Share

Part 11: Allowing several people to work on a document increases overall quality and reduces your workload – **Neil Bothwick** shows you how.

Our expert

Neil Bothwick
Neil has a computer in every room, but for security's sake won't disclose the location of his central server.

When Sir Tim Berners-Lee conceived of the World Wide Web, he foresaw more than a passive medium where one browsed and read the work of others. He saw the web as a read/write medium where geographically separate people could collaborate on the same documents. While the use of forums and comment sections enables others to add to the content of a site, that's more read/add than read/write. To make the web really work we need to be able to alter its content. That's what really differentiates it from the magazine you're holding now.

There are two very different technologies that we'll look at adding to our basic *Apache* server in order to enable content modification. The second – the wiki – is something you're almost certainly familiar with, even if you've only read content on one. But first we'll look at WebDAV, an extension to HTTP that enables the editing of files after upload, and more.

"Sir Tim Berners-Lee saw the web as a read/write medium."

WebDAV

We like our acronyms and abbreviations, but when something has a name like "Web-based distributed authoring and versioning" this becomes a necessity more than a choice, so WebDAV it will be from now on. At this point you may be thinking, "Hang on, haven't we already done version control with *Subversion*?" Well yes we have, but while there are similarities, there are also significant differences (though it's possible to interface with *Subversion* using WebDAV, but that's a story for another day).

Apache is generally set up with WebDAV enabled. Unless you're a *Gentoo*-using control freak, in which case you'll need to install *Apache* with the **DAV USE** flag and add **-D DAV** to **APACHE2_OPTS** in **/etc/conf.d/apache2**. Now you need to edit your *Apache* configuration file in **/etc/apache2** and add:

```
DAVLockDB /var/lock/apache2/var/DAVLock
<Directory /var/www/localhost/htdocs>
Order Allow,Deny
Allow from all
DAV On
AuthType Basic
AuthName DAV
AuthUserFile /var/www/localhost/.htpasswd
<LimitExcept GET OPTIONS>
Require user admin
</LimitExcept>
</Directory>
```

If you think back to our coverage of *Subversion* in **LXF118**, one of the situations we dealt with was concurrency – where a user starts working on a file that someone else is already editing. Without some form of control, there's a risk that one user would save their changes only for a second person to save a new version, overwriting the work of the first. WebDAV handles this with file locking, enabling a file to be locked when someone opens it for read/write operations, preventing anyone else doing the same (although they can still read it). The **DAVLockDB** directive sets the file to use for control locking and must be set globally. It may already be set for you in one of the files included by **httpd.conf**. Check for this with:

```
grep -r DAVLock /etc/apache2
```

If it's not present, add it to **httpd.conf**. The filename isn't important; just put it in a logical location (to you). The rest can either be put in the global configuration or a virtual host. We covered virtual hosts in the first part of this series, but as a refresher, put:

```
NameVirtualHost *:80
```

In your **httpd.conf** and enclose the part from **<Directory>** to **</Directory>** within:

```
<VirtualHost your.host.name:80>
</VirtualHost>
```

» **Last month** We set up a web proxy to save bandwidth and filter websites.

LXF124.tut_network 90 28/8/09 5:31:26 pm

over the web

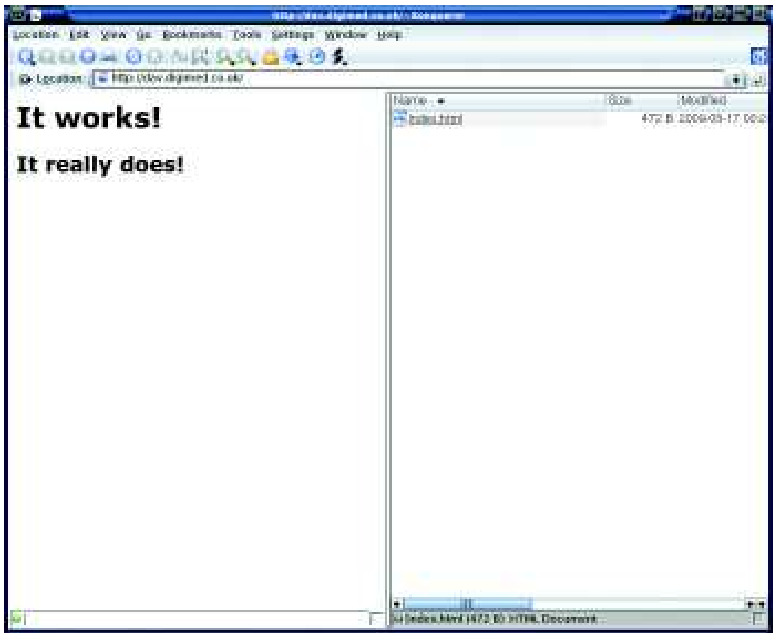
LINUX FORMAT On the DVD
» MediaWiki 1.15.1

The first two directives in the **Directory** stanza are standard fare while the next, unsurprisingly, turns on **DAV** for this location. The three **Auth** directives are to limit who can access this directory, but we've put the **Require** directive inside a **LimitExcept** clause. This applies the **Require** except for **GET** and **OPTIONS** requests, so this directory can be read freely but requires a login for any form of editing. There's also a **Limit** option, which has the opposite effect of only requiring authentication for specified operations, but the "everything but" method works best here.

As you will be writing to the directory you specified, and this writing will be done by the web server, you need to make sure the directory is writable by the user running *Apache*. Usually that user is called *Apache*, but check the **httpd.conf** file for the **User** directive to make sure. Restart *Apache* to apply your changes, but before you do that check the syntax of your configuration by running the following:

```
apache2 -t
```

If that shows no errors, put some contents in your **htdocs** directory – a minimal **index.html** will do for now – restart *Apache* and you're ready to test it.


It works!

It really does!

› *Konqueror* **illustrates the two faces of** ***Apache*** **with WebDAV. On the left you have the standard HTTP view while the right gives read/write access to the files.**

Now you see it...

Konqueror is a good candidate for this, because it's both a browser and file manager with WebDAV support. Open **http://yourhostname** and you will see your HTML page, as expected. Now change the **http:** to **webDAV:** and instead of *Apache* automatically displaying the index file when you give it a directory, it will show a file listing. Right-click on your **index.html** and open it with *KWrite* or *Kate*, then change the text and save it. Switch back to the **http** URL, and there's your modified page.

You don't have to use *Konqueror*: as many other programs work with WebDAV, although KDE's kio slaves make it relatively easy. Fire up *Nautilus* if you're a Gnome user, select File > Connect To Server and your WebDAV server address, then open the file with *Gedit* or *OpenOffice.org*.

This makes remote editing of web pages extremely easy. All you have to do as the caretaker is add whoever you want to edit the pages to the list of permitted users. Because WebDAV is an extension to HTTP it uses the standard HTTP authentication methods, so you can set different permissions for various directories. You can also create groups of users and assign access to groups rather than individual users.

While we've looked at editing HTML, WebDAV is far more flexible. You can read all sorts of files over HTTP, plus watch videos, download software and so on. A WebDAV directory makes a really useful remote file store, and some of the commercial off-site storage services use them for exactly this purpose. Those with write access can use any WebDAV-capable file manager to upload files, while others can connect using a standard HTTP browser session to download them. You aren't limited to Linux software either, as there's plenty of suitable software on all platforms (*Windows Explorer* and *Microsoft Office* both support WebDAV).

As you're permitting read/write access via the internet, there are some sensible precautions to take: most importantly, don't do this if you don't need to. If this is for intranet use, don't grant internet access. Make sure that your authentication controls are in place and tested before opening up your web server to the internet, then test again after doing so. A laptop with a mobile broadband modem is probably the most convenient way to test remote access without leaving your desk.

"There are precautions: most important, don't do this if you don't need to."

If your *Apache* installation is already serving pages to the internet and you don't want to interrupt this, you can either run a second instance of *Apache* on a different port or run your testing system on a virtual machine using *VirtualBox* or one of the VMware products. Virtualisation is ideal for this type of work as it provides an easy way to isolate the virtual computer from the rest of your network.

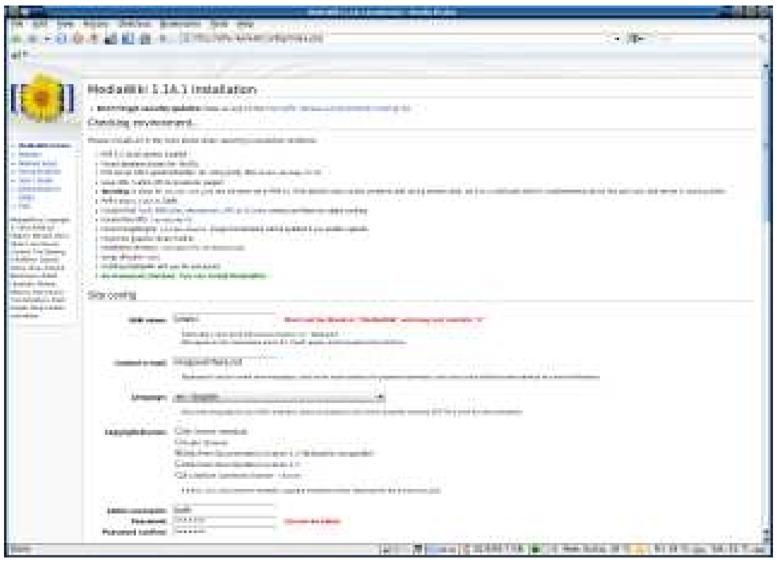
Keeping it private

We've already mentioned that the directory you use must be writable by the user running the *Apache* server. New files are also owned by this user, rather than the logged-in user that »

Quick tip

Use a laptop with a 3G mobile broadband dongle to test access from outside of your network without physically leaving.

» **If you missed last issue** Call 0870 837 4773 or +44 1858 438795.


MediaWiki 1.14.1 installation

Checking environment...

Site config

› *MediaWiki*'s configuration is done through a web form. Make sure outside access is blocked until you complete this.

created them. To reduce the risk of you falling victim to any exploits that may appear, you should limit the abilities of this user – for example their login shell should be set to **/bin/false**. The WebDAV repository is considered private to *Apache*, and modifying files outside of *Apache*, by direct filesystem access for example, should not be allowed. Similarly, if you run an FTP server you shouldn't give it write access to the WebDAV directories.

If you're providing access to private files over the internet, access control may not be enough. You probably want the contents to be secure as they traverse the internet, which means you need an encrypted connection using SSL (Secure Sockets Layer). As we're running over HTTP and this has a secure option – the HTTPS that you use to max out your credit card at Amazon and ThinkGeek – all we need to do it set up a secure virtual machine to give WebDAV this security.

"Allowing the world and his dog to edit your wiki may seem like a bad idea."

You may need to enable loading of the SSL modules when *Apache* starts up. For example in Ubuntu you'll need to input the following:

```
sudo ln -s ../mods-available/ssl.load /etc/apache2/mods-enabled/
sudo ln -s ../mods-available/ssl.conf /etc/apache2/mods-enabled/
```

You also need to create a separate virtual host to handle SSL connections; your distro has almost certainly provided a default one, but you may need to activate it. Ubuntu users do this with:

```
sudo ln -s ../sites-available/default-ssl /etc/apache2/sites-enabled/000-default-ssl
```

You can either use the default SSL virtual host or create your own in the usual way. The main differences are that you replace port 80 with port 443 in the **VirtualHost** directive and include the directives to turn on SSL and specify the location of the certificate, as so:

```
<VirtualHost your.host.name:443>
 SSLEngine on
 SSLCertificateFile /etc/apache2/ssl/server.crt
 SSLCertificateKeyFile /etc/apache2/ssl/server.key
 Rest of config goes here
</VirtualHost>
```

The easiest way to do this is to make a copy of the **default-ssl** virtual host file and edit the names and paths to suit. As usual, you will have to restart the server to apply these changes, which you can do from the services manager or by running:

```
apache2ctl restart
```

The Wiki way

When you mention collaborative web documents, many people immediately think of Wikis – and Wikipedia in particular. The idea of having a website where content can be added, edited and even deleted by anyone seems a formula for chaos at first, but actually works extremely well. Of course you don't have to make a Wiki that's open for all; most Wiki software enables you to create access controls that give groups of users restricted editing rights, or no rights at all beyond reading the pages.

Wikis are often written in PHP or a similar server-side language, and they use an SQL database to store the content and are installed into an *Apache* server, a classic example of LAMP software. As Wikipedia is the best-known and most commonly used example of a wiki, you could do a lot worse than following the project's choice of software, *MediaWiki* (**www.mediawiki.org**).

This has a number of dependencies, so the best way to install it is through your package manager. Once it's installed you need to set up its database, which you can do through its web interface. Because you haven't yet set up any access controls, don't expose your *MediaWiki* directory or virtual host to anyone else until you've completed its configuration. The process is similar for most distros, although some details may vary. On Ubuntu you need to edit **/etc/apache2/conf.d/mediawiki.conf** and uncomment the third line (the one that sets an alias). This is because Ubuntu doesn't recommend using a virtual host for this but an alias instead, although other distros run *MediaWiki* on a virtual host with no problems. If you want the URL to be something other than **http://your.host/mediawiki**, change the first path in the **alias** comment, for example to a more generic:

```
Alias /wiki /var/lib/mediawiki
```

Now point your browser to **http://localhost/wiki** and click on the Setup link. If you see a message saying "Can't write config file, aborting" then you need to set the permissions of the **config** directory so that the *Apache* user can write to it with:

```
chmod a+w /path/to/wiki/config
```

This is only necessary while you're running the initial configuration, so set it back after you have done this with:

```
chmod go-w /path/to/wiki/config
```

The configuration script asks you to set up an admin user and also needs the root password to create a database. If you don't have the root password, you'll have to either ask the admin to create the database for you or use an existing one. The installer will create the necessary tables within the database. The password is for the root user of *MySQL* rather than the computer as a whole – an important distinction,

» **Never miss another issue** Subscribe to the #1 source for Linux on p102.

especially for Ubuntu users. If your package manager installed *MySQL* as a dependency of *MediaWiki*, it will have asked you for a root password; otherwise it will have a password already.

The various options are documented on the configuration page, read through them until you reach the install button. Provided the installer found no problems with your settings it will tell you to move the **LocalSettings.php** file and set its permissions so only the user running the database (usually *Apache*) can read it. Do this with:

```
chown apache: LocalSettings.php
chmod 600 LocalSettings.php
```

Then reset the permissions on the **config** directory as above. Now click on the link in the 'Installation successful!' message to start using your wiki. You could just hit the Edit button and start typing away, but it's better to have some idea of the structure you want and set up the framework for it before you start adding contents. An even better idea, since a wiki is all about collaboration, is to set up the framework and then invite others to add the content.

Access control

Allowing the world and his dog to edit your wiki may seem like a bad idea, and sometimes it is. Access control can be performed at a number of levels, the simplest of which is to prevent users who aren't logged in, or who don't have administrator privileges, from editing a page. While logged in with the administrator account you set up during installation, each page has a Protect tab that can be used to control editing rights to that page.

Anything more requires some delving into the **LocalSettings.php** file. This file is written in PHP, so it helps if you have at least a minimal understanding of the language, but this isn't essential, as it consists mainly of variable assignments. For example, the line:

```
$wgEmailConfirmToEdit=true;
```

means that a user can only edit pages after they've supplied a valid email address and responded to a confirmation sent to that address, which serves to keep many spammers out, although in reality it won't stop someone who's really determined to deface your site.

The **LocalSettings.php** file is fairly short as master configuration files go because the defaults are set in **includes/DefaultSettings.php**. You mustn't edit this file however, as it will be overwritten during an upgrade, wiping out your carefully crafted settings. Instead, copy the settings you want to change from this file to **LocalSettings.php** and make the changes there. **LocalSettings.php** has priority over the defaults.

Hone your security

More detailed controls can be set with **$wgGroup Permissions**. This is a complex array that can look quite scary, but individual settings are easy enough to understand. For instance, this line in **DefaultSettings**.**php** enables registered users to edit pages:

```
$wgGroupPermissions['user']['edit'] =true;
```

Copy the line to **LocalSettings**.**php** and change the **true** to **false** to remove that permission. This refers to the group called **User**, which contains all registered (and confirmed, if required) users. There are other groups defined, such as **Sysop** and **Bureaucrat**, that have greater powers. You can change the permissions for each of these groups, as well as controlling which users belong to the more powerful groups. You can also create new groups, to tie in with members of particular projects perhaps. All of this is documented on your wiki; go to **http://your. hostname/wiki/Manual: Configuration_settings** for a full list of configuration options.

Now that you have either a DAV server, a wiki, or both running, you have no excuse for not getting your organisation's documentation into tip top shape, letting others do most of the work in the process. LXF

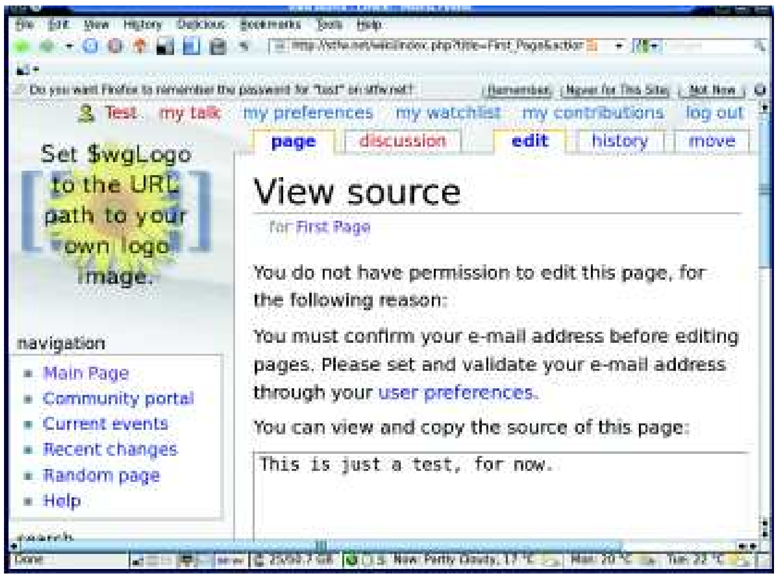
SSL certificates

If you want to provide secure access over HTTPS, you need to install a certificate on your site. Most distros include a package with a self-signed certificate, which is enough for testing but not suitable for real-world use. HTTPS provides two types of security; first it encrypts the data so that when you send your credit card details to your favourite online store, no-one snooping on the connection can read them.

Second, it verifies that you're connected to who you think you are, and not sending your credit card information to some dodgy site on the other side of the world. A self-signed certificate takes care of the first part, but not the second. So if you just want to keep your edits away from prying eyes it will do the job but for real security you need to make sure you buy a properly authenticated certificate.

Quick tip

If *MediaWiki* seems rather heavyweight and you just want to be able to work on a few documents over a LAN, *DokuWiki* (**http://wiki. splitbrain.org/ wiki:dokuwiki**) is a much simpler alternative, without many of the features or complexity of *MediaWiki*.


File Edit View History Delicious Bookmarks Tools Help
Test my talk my preferences my watchlist my contributions log out
page discussion edit history move
Set $wgLogo to the URL path to your own logo image.
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
Done

View source

for First Page

You do not have permission to edit this page, for the following reason:

You must confirm your e-mail address before editing pages. Please set and validate your e-mail address through your user preferences.

You can view and copy the source of this page:

```
This is just a test, for now.
```

› Blocking unverified users is a simple way to restrict editing of your Wiki to only those who can be identified.

» **Next month** Keep your data safe with an automated backup server.