Implementation Report

Kent Robin Haugen
Marius Wollamo
Cristina Companys-Antunez
Jose Maria Gil-Menendez
Kenneth Børtveit

March 22, 2012

1 Introduction

This report contains a description of the implemented system and changes from the original plan, a step-by-step guide to installing the system from scratch and a description of the routines that should be carried out in order to maintain the system.

In this report we assume that you already have a server up and running based on the installation guide provided by the OS-group. This document can be obtained by sending an email to:

OS-group: tdt4285-os@idi.ntnu.no

or us, the

Web & Wiki -group: tdt4285-web@idi.ntnu.no.

2 System Description

haha

3 Installation Process

This section explains how to set up all of the services which the Web&Wiki-group should deliver. This includes a web server supporting HTTP 1.1, a wiki service and personal homepages. If not told otherwise, all commands should be run in Bash and executed as sudo if you're not logged in as root. Also you need shell access at tdt4285.idi.ntnu.no.

3.1 Installing NGINX, PHP5 and MySQL on Debian Squeeze

This section explains how to install and set up the NGINX web server, installing PHP5 and making it work with the NGINX web server and installing MySQL, all of which are to be installed on the Debian Squeeze running server.

- Installing MySQL 5.0
  To get the wiki to work, we need to install MySQL on the server.

  ```
  $ aptitude install mysql-server mysql-client
  ```

- Installing NGINX and starting up the server

  ```
  $ aptitude install nginx
  $ /etc/init.d/nginx start
  ```

- Installing PHP5 and its dependencies

  ```
  $ aptitude install php5-cgi php5-mysql php5-curl php5-gd php5-idn php-pear p
  ```

  Open /etc/php5/cgi/php.ini

  ```
  $ vi /etc/php5/cgi/php.ini
  ```

  Add this at the end of the line in php.ini-file:

  ```
  cgi.fix_pathinfo = 1
  ```

  There's no standalone FastCGI daemon package for Debian Squeeze, therefore we use the spawn-fcgi program from lighttpd. We install lighttpd as follows:

  ```
  $ aptitude install lighttpd
  ```

  You will probably see an error message saying that lighttpd can't start because port 80 is already in use, and this is because NGINX is already listening on this port. Run the following command so that lighttpd don't start at boot time.

  ```
  $ update-rc.d -f lighttpd remove
  ```

  To start a PHP FastCGI daemon listening on port 9000 on localhost and running as the user and group www-data, we add the following command to /etc/rc.local right before the exit line:

  ```
  $ vi /etc/rc.local
  $ /usr/bin/spawn-fcgi -a 127.0.0.1 -p 9000 -u www-data -g www-data -f /usr/b
  ```

- Setting up NGINX config files
  To get NGINX to work with PHP and setting up the entry point for our webservice we need to edit the config files. If the .conf files don't exist, create them. In the /etc/nginx folder, edit the nginx.conf file to look like this:

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;
    include /etc/nginx/conf.d/default.conf;
    include /etc/nginx/sites-available/mediawiki/mediawiki.conf;
    include /etc/nginx/horde.conf;
}
```

The fastcgi_params file in /etc/nginx/ should look like this:

```
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;
```

```
fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

The default.conf file in /etc/nginx/conf.d should look like this:

```
server {
    listen       80;
    server_name  tdt4285.idi.ntnu.no;

    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;

    location / {
        root   /var/www/;
        index  index.html index.htm index.php;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ \.php$ {
    #    root           html;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  /var/www/$fastcgi_script_name;
        include        /etc/nginx/fastcgi_params;
```

```
        fastcgi_buffers 16 16k;
        fastcgi_buffer_size 32k;
    }

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}

   location /nginx_status {
        stub_status on;
        access_log   off;
        allow all;
   }
}
```

Then restart the NGINX server and add a testfile in /var/www/test.php to check if it works.
The PHP test file:

```
<?php
phpinfo();
?>
\end{lstlisting
```

```
To restart the NGINX server:
\begin{lstlisting}
$ /etc/init.d/nginx restart
```

Then go to http://tdt4285.idi.ntnu.no/info.php to check if it works.

3.2 MediaWiki

MediaWiki is dependant on PHP5 and MySQL4 so this must be working before trying to set up the wiki. You also need the username and password for the database you are going to use with the wiki.

Navigate to /var/www/ and type the following commands in Bash:

```
$ wget http://download.wikimedia.org/mediawiki/1.18/mediawiki-1.18.1.tar.gz
$ tar xzvf mediawiki-1.18.1.tar.gz
$ mv mediawiki-1.18.1 /var/www/mediawiki
```

Then use a browser to navigate to http://tdt4285.idi.ntnu.no/mediawiki and begin the setup process. The MySQL username/password is wiki/appelsin.

When prompted about authentication you choose to only allow admins to create new users and that all registered users can edit the wiki. The wiki itself is supposed to be readable by everybody.
When you are finished you will be asked to download the LocalSettings.php file. Do so, and copy it to /var/www/mediawiki. If that LocalSettings.php file does not work, replace the code in it with this following code, and it is supposed to work:

```
<?php
# This file was automatically generated by the MediaWiki 1.18.1
# installer. If you make manual changes, please keep track in case you
# need to recreate them later.
#
# See includes/DefaultSettings.php for all configurable settings
# and their default values, but don't forget to make changes in _this_
# file, not there.
#
# Further documentation for configuration settings may be found at:
# http://www.mediawiki.org/wiki/Manual:Configuration_settings

# Protect against web entry
if ( !defined( 'MEDIAWIKI' ) ) {
        exit;
}

## Uncomment this to disable output compression
# $wgDisableOutputCompression = true;

$wgSitename       = "TDT4285 - Wiki";

## The URL base path to the directory containing the wiki;
## defaults for all runtime URL paths are based off of this.
## For more information on customizing the URLs please see:
## http://www.mediawiki.org/wiki/Manual:Short_URL
$wgScriptPath       = "/mediawiki";
$wgScriptExtension  = ".php";

## The protocol and server name to use in fully-qualified URLs
$wgServer           = "http://tdt4285.idi.ntnu.no";

## The relative URL path to the skins directory
$wgStylePath        = "$wgScriptPath/skins";

## The relative URL path to the logo.  Make sure you change this from the defaul
## or else you'll overwrite your logo when you upgrade!
$wgLogo             = "/mediawiki/logo.jpg";
```

```
## UPO means: this is also a user preference option

$wgEnableEmail      = true;
$wgEnableUserEmail  = true; # UPO

$wgEmergencyContact = "kentrobi@stud.ntnu.no";
$wgPasswordSender   = "mediawiki@tdt4285.idi.ntnu.no";

$wgEnotifUserTalk       = false; # UPO
$wgEnotifWatchlist      = false; # UPO
$wgEmailAuthentication = true;

## Database settings
$wgDBtype           = "mysql";
$wgDBserver         = "localhost";
$wgDBname           = "tdt4285_wiki";
$wgDBuser           = "wiki";
$wgDBpassword       = "appelsin";

# MySQL specific settings
$wgDBprefix         = "";

# MySQL table options to use during installation or update
$wgDBTableOptions   = "ENGINE=InnoDB, DEFAULT CHARSET=binary";

# Experimental charset support for MySQL 4.1/5.0.
$wgDBmysql5 = false;

## Shared memory settings
$wgMainCacheType    = CACHE_NONE;
$wgMemCachedServers = array();

## To enable image uploads, make sure the 'images' directory
## is writable, then set this to true:
$wgEnableUploads  = true;
#$wgUseImageMagick = true;
#$wgImageMagickConvertCommand = "/usr/bin/convert";

# InstantCommons allows wiki to use images from http://commons.wikimedia.org
$wgUseInstantCommons  = false;

## If you use ImageMagick (or any other shell command) on a
## Linux server, this will need to be set to the name of an
## available UTF-8 locale
$wgShellLocale = "en_US.utf8";
```

```
## If you want to use image uploads under safe mode,
## create the directories images/archive, images/thumb and
## images/temp, and make them all writable. Then uncomment
## this, if it's not already uncommented:
#$wgHashedUploadDirectory = false;

## Set $wgCacheDirectory to a writable directory on the web server
## to make your wiki go slightly faster. The directory should not
## be publically accessible from the web.
#$wgCacheDirectory = "$IP/cache";

# Site language code, should be one of the list in ./languages/Names.php
$wgLanguageCode = "en";

$wgSecretKey = "0ddbedc2af041dc5bffb42d174658336c5a17a12fe42c97b3bf8f1054799ac8c

# Site upgrade key. Must be set to a string (default provided) to turn on the
# web installer while LocalSettings.php is in place
$wgUpgradeKey = "7a4cbf4849a8bbd6";

## Default skin: you can change the default skin. Use the internal symbolic
## names, ie 'standard', 'nostalgia', 'cologneblue', 'monobook', 'vector':
$wgDefaultSkin = "vector";

## For attaching licensing metadata to pages, and displaying an
## appropriate copyright notice / icon. GNU Free Documentation
## License and Creative Commons licenses are supported so far.
$wgRightsPage = ""; # Set to the title of a wiki page that describes your licens
$wgRightsUrl  = "";
$wgRightsText = "";
$wgRightsIcon = "";
# $wgRightsCode = ""; # Not yet used

# Path to the GNU diff3 utility. Used for conflict resolution.
$wgDiff3 = "/usr/bin/diff3";

# Query string length limit for ResourceLoader. You should only set this if
# your web server has a query string length limit (then set it to that limit),
# or if you have suhosin.get.max_value_length set in php.ini (then set it to
# that value)
$wgResourceLoaderMaxQueryLength = 512;

# The following permissions were set based on your choice in the installer
$wgGroupPermissions['*']['createaccount'] = false;
$wgGroupPermissions['*']['edit'] = false;
```

```
# End of automatically generated settings.
# Add more configuration options below.
```

You also need to set up the MediaWiki configuration for NGINX. Create a file called mediawiki.conf in /etc/nginx/sites-available/mediawiki/

```
$ vi /etc/nginx/sites-available/mediawiki/mediawiki.conf
```

And include the following code:

```
server {
   listen 80;
   server_name localhost;

   root /var/www/mediawiki;

   location /mediawiki {
     index index.php5;
     error_page 404 = @mediawiki;
   }

   location @mediawiki {
     rewrite ^/([^?]*)(?:\?(.*))? /index.php5?title=$1&$2 last;
   }

   location ~ \.php5?$ {
     include /etc/nginx/fastcgi_params;
     fastcgi_pass  127.0.0.1:8888;
     fastcgi_index index.php5;
     fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
   }
 }
```

3.3 Create Personal Websites

In order to create personal websites for all the registered users on the server, just run this python script. Put all the usernames in the users array and run as sudo. It will create a link from the users own folder in /home/[username] called public_html and set permissions and owner in the folders in /var/www/[username]. The URL for the personal websites are http://tdt4285.idi.ntnu.no/[username]

```
#create_user_spaces.py
import os

users = ["user1", "user2", "user3"] #include the usernames from /home/
```

```
for u in users:
    if not os.path.exists("/var/www/" + u):
        os.makedirs("/var/www/" + u)
        os.system("chmod 755 /var/www/" + u)
        os.system("chown " + u + " /var/www/" + u)
        os.system("ln -s /var/www/" + u + " /home/" + u + "/public_html")
```

4 Routines

Haha

5 Summary

haha