wakarimasen Documentation

Release 1.2

dequis

Apr 04, 2017

Contents

Contents:

CHAPTER 1

Installation

Wakarimasen is still experimental software - use it at your own risk and if you know how to fix stuff when it breaks.

Requirements

- Shell access to the server
- Python >= 2.6, <= 3
- Werkzeug
- SQLAlchemy >= 0.8
- Jinja2
- ImageMagick commandline tools (`convert` and `identify`)
- `file` command

Supported deployment methods

- uWSGI
- FastCGI
- CGI (fallback)

Supported webservers

- Apache: Completely supported
- nginx: Works, but a few features such as bans rely in .htaccess
- lighttpd: Works, but same as nginx.

Development server included (`python wakarimasen.py http`)

Note on root access

Most instructions in here assume that you have at least a virtual private server with root access. However, it's technically possible to install requirements using virtualenv and use wakarimasen over cgi or fastcgi if already configured in a shared server.

Installing dependencies

All dependencies should be available from the package manager of the average linux distro.

If the python dependencies are too old, you could install them with pip instead. If you don't want or can't do system-wide installs of python packages, virtualenv exists and integrates nicely with pip.

If you don't have `convert`, `identify` or `file`, and can't install them with a package manager system-wide, well, hope you don't mind not having images in the imageboard.

Basic installation (CGI)

This section explains the simplest setup, assuming that your webserver already has CGI working. If you need to configure your webserver for cgi or something more efficient than cgi, see *Webserver configuration*

1. First, place the source code somewhere in the docroot. That is, the `wakarimasen.py` file should be where you'd put an index.html file.
2. Copy config.py.example to config.py. Edit it and set `ADMIN_PASS`, `SECRET` and `SQL_ENGINE`. The format for `SQL_ENGINE` is the following:

   ```
   SQL_ENGINE = 'mysql://USERNAME:PASSWORD@HOSTNAME/DATABASE'
   ```

   You can also use sqlite:

   ```
   SQL_ENGINE = 'sqlite:///wakarimasen.sqlite'
   SQL_POOLING = False
   ```

   Note that this will create the database in the current directory - please avoid exposing it to the webserver!
3. Now make sure the shebang line in wakarimasen.py points to the right python interpreter (the default is `#!/usr/bin/python2`, do not use a python 3.x interpreter) and that the file has execute permissions. If you use suexec for cgi, it must be chmod 755, too.

   Visit `http://example.com/wakarimasen.py` - This will check for any configuration errors in your installation, and if everything is okay, it should open the first time setup page. Enter the `ADMIN_PASS` here.
4. To create a new board called /temp/, copy the base_board directory:

   ```
   cp -r base_board temp
   ```

   Edit temp/board_config.py. Important settings are NUKE_PASS, TITLE and SQL_TABLE. Keep in mind most of those options are not supported for now (captcha, load balancing, proxy, etc).
5. Go to `http://example.com/wakarimasen.py?board=temp` - This should rebuild the cache and redirect you to your board.

CHAPTER 2

Webserver configuration

Apache

CGI

TODO (Should be very similar to the first steps of FastCGI setup..)

FastCGI

Add this to your config:

```
DirectoryIndex wakaba.html

<Directory "/path/to/wakarimasen">
    Options +ExecCGI
</Directory>
```

Choose either `mod_fastcgi`:

```
LoadModule fastcgi_module modules/mod_fastcgi.so
<IfModule fastcgi_module>
  AddHandler fastcgi-script .fcgi
</IfModule>
```

Or `mod_fcgid`:

```
LoadModule fcgid_module modules/mod_fcgid.so
<IfModule fcgid_module>
    AddHandler fcgid-script .py .fcgi
</IfModule>
```

Nginx

CGI with fcgiwrap

See this page for fcgiwrap installation details.

Then add this to the server block:

```
index wakaba.html;
include /etc/nginx/fcgiwrap.conf;
```

You should ensure that fcgiwrap.conf includes a location block, and that it matches wakarimasen.py (sometimes it's limited to .cgi files). If it doesn't have a location block, put that include inside one:

```
location /wakarimasen.py {
    include /etc/nginx/fcgiwrap.conf;
}
```

If you don't do this, fcgiwrap might do weird stuff like throwing '502 bad gateway' errors for most files.

FastCGI servers

Recent versions of wakarimasen have TCP and unix socket based standalone fastcgi servers. To use them, start wakarimasen.py like this:

```
# start a tcp fcgi server with the default settings, in 127.0.0.1:9000
python wakarimasen.py fcgi_tcp

# bind tcp fcgi server to a certain ethernet interface, port 9001
python wakarimasen.py fcgi_tcp 192.168.0.1 9001

# start a unix socket fcgi server in /tmp/derp
python wakarimasen.py fcgi_unix /tmp/derp
```

In the nginx config:

```
index wakaba.html;
location /wakarimasen.py {
    include /etc/nginx/fastcgi.conf;
    fastcgi_pass unix:/tmp/derp;

    # or: fastcgi_pass 127.0.0.1:9001;
}
```

When using unix sockets, check that the file is readable by the nginx user.

Nginx doesn't have a fastcgi process spawner. You'll have to write a init script, a systemd unit, or use something like supervisor.

Or just leave the thing running in a tmux/screen session, only to find a few weeks later that your wakarimasen has been offline for a long time because your server mysteriously rebooted.

Lighttpd

CGI

Just add this to the config:

```
server.modules += ("mod_cgi")
cgi.assign = (".py"  => "/usr/bin/python2")
index-file.names += ("wakaba.html")
```

As an nginx fanboy I'm slightly annoyed at how easy this was.

FastCGI

TODO

uWSGI

uWSGI is probably the best deployment setup. It can also be the most complex to setup. This document is not going to cover the details, but you can check the uWSGI docs. In particular:

- The quickstart gives a rough outline of the process.
  - Note: wakarimasen can't run directly with the uwsgi http server for now, you need to put it behind a real webserver.
  - Note: The uwsgi "network installer" is awesome, try it out.
- Using the emperor can raise the enterpriseness of your setup significantly.
- The web server integration page gives several alternatives for each server.
  - There are a few modules for apache. You have to grab them from the uwsgi git repo and run the specified `apxs` command to compile and install.
  - Nginx has built-in support of uwsgi. That page describes how to use it.

More detailed instructions soon™

CHAPTER 3

Commandline actions

Wakarimasen includes a few administrative commands that can be used from the commandline.

To use them, do:

```
python wakarimasen.py <command> [parameters]
```

To see usage info of an individual command, do:

```
python wakarimasen.py help <command>
```

Servers

- fcgi_tcp *[host [port]]*

  Starts a standalone FastCGI server over tcp. Defaults to listening on 127.0.0.1, port 9000

- fcgi_unix <*path*>

  Starts a standalone FastCGI over unix socket. The path is required, and you should ensure the permissions allow the webserver to connect.

- http *[host [port]]*

  Starts a http server for development/testing purposes. Do not use in production, even cgi is better than this.

Admin actions

- delete_by_ip <*ip*> <*boards*>

  <*boards*> is a comma separated list of board names.

- rebuild_cache <*board*>

- rebuild_global_cache

Admin actions require some knowledge about the webserver environment. For this reason, you need to pass the following environment variables

- `DOCUMENT_ROOT`: full filesystem path to html files. Example: `/srv/http/imageboard.example.com/`
- `SCRIPT_NAME`: url to wakarimasen.py without host part. Example: `/wakarimasen.py`
- `SERVER_NAME`: hostname of the webserver. Example: `imageboard.example.com`
- `SERVER_PORT`: port of the webserver *(optional)*. Example: `80`

If these values are wrong, it will probably result in a bunch of broken links in the generated pages. Try rebuilding cache from the real web interface.

Complete example usage:

```
DOCUMENT_ROOT=$PWD SCRIPT_NAME=/wakarimasen.py SERVER_NAME=0.0.0.0 \
    python wakarimasen.py rebuild_global_cache
```

You could also have a script that sets this for you.