yewdocs

Release 0.2.0

Paul Wolf

Mar 30, 2021

CONTENTS:

Yewdocs is a personal document manager that makes creating and editing text documents from the command line easier than using only an editor and filesystem commands.

Yewdocs is for *text* documents: plain text, restructuredText, markdown, conf, etc. It offers these features:

- Filesystem transparency: the user doesn't need to know where files are stored.
- Entirely keyboard driven.
- Tags: define tags and organise your documents with them.
- Familiar commands: yewdocs has commands like 'ls', 'head', 'tail', that are familiar to most shell users.
- Optional cloud storage for synchronising to multiple devices/workstations.
- Document conversions: it will generate any format that pandoc is able to convert to/from. There is special support for generation of html and the use of templates.
- Integration with other command line utilities: as just another shell utility, you can do normal shell piping in and out, grep, etc.

Think of it as a command line note taking application.

The target users are those who prefer to work on the command line without the overhead of switching back and forth between a shell and the host OS GUI. When working regularly on the command line, it is a considerable annoyance to have to break out to use the host OS file management app to find files and use the mouse. Yewdocs lets the user seamlessly browse and operate on her collection of text files. These can be snippets, larger documents, notes, etc. Exporting to other formats is easy and natural.

A major design goal is to reduce the mental overhead of finding files. Once a file is managed by Yewdocs, it is easy to perform operations like editing it without needing to remember the exact name or location. Documents can be managed by Yewdocs either within its own repository or in-place as linked documents.

You are not forced to choose between your favourite non-text editor and shell editor. You can just as well use Sublime, Atom or other non-console interfaces for editing Yewdocs documents.

It's possible to maintain text documents on a server and sync to any local device that supports Python (>= 3.6) and one of the common *nix shells.

You can edit and manage any kind of text file, txt, rst, md, conf, etc. Yewdocs does have a slight prejudice towards Markdown for newly created documents but you can easily specify any format you wish or convert a file to another format after creating it.

CHAPTER

ONE

INSTALLATION

Yewdocs works with Python >= 3.8

Make sure you have Python3 installed. Make sure you have pip3 working.

You can user yewdocs without Pandoc, but some nice features will be missing like converting documents to other types, pdf, html, docx, etc. Pandoc must be installed following the instructions specific to your operating system.

MacOS:

```
brew install pandoc
```

Ubuntu:

```
sudo apt-get install pandoc
```

Windows:

```
http://pandoc.org/installing.html
```

Git clone the repo:

```
git clone git@github.com:paul-wolf/yewdoc-client.git
```

cd into the resulting directory and execute the install command:

```
cd yewdoc-client
pip3 install --editable .
```

That should be all that is required. We will later make a PyPi module available. Now type:

```
yd info
```

You should see output about settings.

For PDF exports, you'll need to also install pdflatext in addition to pandoc:

On macos:

```
brew cask install mactex
```

CHAPTER TWO

USAGE

Create a new document and start editing:

```
yd create foo
```

Yewdocs uses the EDITOR environment to determine what editor to launch. You can also have Yewdocs launch an editor from the host OS and let it decide which application handles that file type. This might be Atom, Sublime Text or whatever editor you choose to associate with the file type depending on your OS.

Edit the file we just created:

```
yd edit foo
```

Edit the file with Sublime or whatever you have specified in your host OS to handle this type of file:

```
yd edit -o foo
```

You don't have to provide the whole title of the document. If the fragment, in this case "foo", matches case-insensitively to a document, it will be loaded in the editor. Otherwise, the user is presented with a choice of all matching files.

```
yd show foo
```

will dump the contents of "foo" to stdout. Create a new document now from stdin:

```
yd read bar
```

Type some content and enter end of file (ctrl-d usually).

Copy a document to a new one:

```
yd show foo | yd read --create bar
```

A copy of the contents of foo will appear in a new document, bar.

List all your documents:

```
yd ls
```

Sync your documents to/from a Yewdocs server:

```
yd sync
```

You must have previously registered with the Yewdocs server. This is entirely optional. It will also sync tags and document/tag associations. See below under Configuration.

You can symbolically link a file:

```
yd take ~/.tmux.conf --symlink
```

the file `.tmux.conf` will become a managed document but any operations on the file will modify the file in-place. This can be very convenient because you don't have to remember the exact name or location of the tmux file to edit it:

```
yd edit tmux
```

But, of course, great care must be taken to not forget that you are not working on a copy but the file itself.

We don't sync from remote to a symlinked file. That means, local symlinked files will not be overwritten from remote. This is to prevent unintential changes to local files.

CHAPTER

THREE

SPECIFYING DOCUMENTS

All the commands that operate on one or more documents will take as a name one of the following:

- Document title (or case-insensitive sub-string of the title)
- Document id (a UUID)
- Short document id (first 8 characters of full id)

The document id is a UUID that looks something like this:

```
3ccc3fcc-5acc-11e5-b07d-ac87a33b7daa
```

The short document id is therefore `3ccc3fcc`. I can use the `describe` command to get full information about this document:

```
  yd describe 3ccc3fcc
uid      : 3ccc3fcc-5acc-11e5-b07d-ac87a33b7daa
link     : False
title    : Project Management
location : default
kind     : md
size     : 1563
digest   : 0899f1728b9b653f8477a64b6fa5f750b87bd2a77a716dfe0eeeb91ae90b8fc4
path     : /Users/paul/.yew.d/yewser/default/3ccc3fcc-5acc-11e5-b07d-ac87a33b7daa/doc.
↪md
Last modified: 2015-09-16 19:07:16+00:00
```

I can also use a part of the title:

```
yd describe project
```

If more than one title has the string `project`, it will provide a list to choose from.

Use the id to be very exact:

```
yd describe 3ccc3fcc-5acc-11e5-b07d-ac87a33b7daa
```

or the short id:

```
yd describe 3ccc3fcc
```

CHAPTER FOUR

TAGS

You can create tags and associate them with documents for making file organisation and publishing easier.

Assign the ‘red’ tag to the doc ‘foo’:

```
yd tag red foo
```

List documents with the red tag in long format, humanized:

```
yd ls -lh -t red
```

Dissociate the tag ‘red’ from the document ‘foo’:

```
yd tag -u red foo
```

How many documents with the foo tag:

```
yd ls -t foo | wc
```

List all tags:

```
yd tags
```

CHAPTER FIVE

BACKING UP YOUR DATA

There are many ways to backup data:

- rsync the data directory (~/.yew.d).
- Use an online file replication service, like Dropbox
- Use the RESTful remote that is built in or an alternative remote.
- Use the archive command

The archive command:

```
yd archive
```

will create a tgz file in the local directory with all documents. You can then put this somewhere safe or

CHAPTER

SIX

EXPORTING DOCUMENTS

You can create MS Word, PDF, HTML or other types of documents from your text documents.

```
yd convert foo docx
```

will create a file in the current directory called: 'foo.docx'.

It supports whatever pandoc supports. Type:

```
yd convert
```

To see a list of supported formats.

For PDF, you'll need to also install pdflatext in addition to pandoc:

https://www.latex-project.org/get/

On macos:

```
brew cask install mactex
```

CHAPTER

SEVEN

DOCUMENTS IN BROWSER

You can view a set of documents associated with a tag or tags in a web browser:

```
yd browse -t blog
```

This will try to convert all documents tagged with 'blog' to html and load them in the default browser. The default

```
yd browse my_template.j2 -t blog
```

where `my_template.j2` is a Jinja template (http://jinja.pocoo.org/). Without this the default template is used. This has a left sidebar for navigating documents.

CHAPTER

EIGHT

USERS AND CONFIGURATION

Yewdocs implements its own users. These are not the same as either the local system user nor the remote Yewdocs user. In `~/.yew.d/` you'll find one or more user names. You can setup new users any time with the `--user` parameter that comes right after `yd`:

```
yd --user paul ls -l
```

Use this immediately after `yd` and the command will use that user context. Yewdocs tries to get the user via several means:

- user the current operating system user
- check for `--user`
- check for environment: YEWDOC_USER
- check for a config file called `~/.yew` that has this:

  [Yewdoc] username = yewser

where `yewser` is the desired username.

CHAPTER

NINE

USING DIFFERENT REMOTES

You can sync your collection of documents to a remote backend to make them available across different devices/workstations.

Two remotes are possible: Web and AWS S3 storage. You can add your own remote backend, see `remote` directory for the source code.

For S3, you need an AWS account and access credentials. In the ~/.yew.d/settings.json you configure access to the remote.

```
{
    "location": {
        "default": {
            "remote_type": "RemoteS3",
            "aws_access_key_id": "XXXXXXXXXXXXXXXXXXXXXXXXXXX",
            "aws_secret_access_key": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",
            "s3_bucket": "my-personal-bucket"
        }
    }
}
```

The remote REST backend is configured something like this:

```
{
    "location": {
        "default": {
            "remote_type": "RemoteREST",
            "url": "https://doc.yew.io",
            "email": "paul.wolf@ripeco.com",
            "username": "paul",
            "password": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
            "first_name": "Blah",
            "last_name": "Wolf",
            "token": "zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz"
        }
    }
}
```

CHAPTER TEN

SOME TECHNICAL DETAILS

The files are all kept in a directory: `~/.yew.d/`. Assume we have a username of `yewser`:

```
~/yew.d/yewser/default/
```

`default` in this example refers to the "location". This is hard-coded at this time, but in future will support different remote settings.

under `default` are all the document directories, one directory per document. Each document directory has the document with the appropriate text extension. There might be a media subdirectory if you have attached files to the document, `yd attach <filepath>`. In addition, there could be a file for holding tags associated with the document `__tags.json`.

in the yewdoc user directory, `~/yew.d/yewser` in our example, there are at least two files:

- index.json: an index of all the documents and tags
- settings.json: user preferences

The index.json is kept up to date whenever the user makes changes to documents, create, edit, tag, delete, etc. If this is corrupted somehow, it can be regenerated:

```
yd generate-index
```

This command can be invoked any time and the index.json will be replaced with a accurate version. settings.json however will need to be created from scratch however if it is deleted or lost. Add things to it with the `user-pref` command:

```
yd user-pref <name> <value>
```

Check preferences:

```
yd user-pref
```

Most of these settings are set via the `yd configure` command. But you can change them via the `user-pref` command more directly:

```
yd user-pref location.default.first_name Paul
```

Will change the first_name to Paul. These are generally only used to configure a remote.

CHAPTER ELEVEN

OVERVIEW OF COMMANDS

For local files the following commands are available:

```
api              Get API of the server.
apply            List documents.
archive          Create tgz archive in the current directory of all...
attach           Take a file and put into media folder.
authenticate     Authenticate with remote and get token.
browse           Convert to html and attempt to load in web browser.
configure        Get configuration information from user.
context          Set or unset a tag context filter for listings.
convert          Convert to destination_format and print to stdout or save...
create           Create a new document.
decrypt          Decrypt a document.
delete           Delete a document.
describe         Show document details.
diff             Compare two documents.
edit             Edit a document.
encrypt          Encrypt a document.
find             Search for spec in contents of docs.
generate-index   Iterate document directory and output index json to...
head             Send start of document to stdout.
info             Provide configuration information.
kind             Change kind of document.
ls               List documents.
path             Show local disk path for document.
ping             Ping server.
purge            Delete documents of zero size.
push             Push all documents to the server.
read             Get input from stdin and either create a new document or...
register         Try to setup a new user account on remote.
rename           Rename a document.
rls              List documents.
show             Send contents of document to stdout.
status           Print info about current setup.
sync             Pushes local docs and pulls docs from remote.
tag              Manage tags.
tags             List all tags.
tail             Send end of document to stdout.
take             Import a file as a document.
user-pref        Show or set user preferences.
verify           Check docs exist.
```

CHAPTER

TWELVE

SETTING UP THE ENVIRONMENT FOR DEVELOPMENT

```
python -m venv .venv && source .venv/bin/activate && pip install --upgrade pip && pip␣
↪install -r requirements.txt
```

CHAPTER

THIRTEEN

INDICES AND TABLES

- genindex
- modindex
- search