## Contents

1 User’s Guide  
1.1 Foreword ......................................................... 3  
1.2 Installation .................................................... 4  
1.3 Quick Start ...................................................... 5  
1.4 Configuration .................................................. 8  
1.5 Post Syntax ...................................................... 9  
1.6 Themes .......................................................... 10  
1.7 Commands ....................................................... 10  
1.8 Blog’s Structure ............................................... 12  
1.9 The Makefile helper ........................................... 13  
1.10 Use Cases ...................................................... 13  
1.11 Contribute ..................................................... 14
Release v0.3.8. *(Installation)*

Lilac is a MIT Licensed static blog generator, written in Python. It’s fast, simple enough and easy to use.

This is the document’s English version.

The codes are on github.com: hit9/lilac.git
1.1 Foreword

Read this before you get started with lilac. This introduces purpose and goals of this project.

1.1.1 About the name ‘lilac’

‘lilac’ is ‘’ in Chinese, they are everywhere in my school (HIT). They are beautiful!

1.1.2 Why another ‘wheel’?

There are so many similar tools already, why do you create a new one?

- I need one written in Python
- I need only tags, not categories.
- I like clean themes.
- It should be easy to configure.

And, a really big reason is:

- TOML fits to be its configuration language.

1.1.3 Features

Here is lilac’s features, it’s worth to take a look.

- TOML and GFM based.
- 100% in Python (any linux distribution comes with python)
- Built-in tags & feed & theme & codes highlighting support
- We use Jinja2 to render templates.
- No categories, only tags. (It’s A GOOD FEATURE!)
- Minimal configuration.
1.1.4 Libraries behind the back

You may want to check out this file to get the answer: requirements.pip

- **misaka**: the python wrapper for **sundown** (Sundown is a markdown parser in C that runs really fast)
- **Jinja2**: the famous templating engine for Python, comes from **pocoo**.
- **Pygments**: wide used python syntax highlighter.
- **blinker**: an excellent signal library for Python.
- **docopt**: a pythonic command line arguments parser, really an awesome library, it uses readable doc string to parse command arguments.
- **toml.py**: a TOML language parser for Python, written by me.

1.1.5 Demo Site

What does a blog generated by lilac look like?

Here is a demo website generated by lilac: Make Difference, and you may want to see the source codes of the demo site: Demo site on github.com

Ready to start? See **Installation**.

1.2 Installation

There are several ways to install lilac, but the recommended one is to install lilac with **Virtualenv**.

1.2.1 OS Support

Lilac supports only *nix: Mac OS X, Linux, BSD (Only linux tested).

1.2.2 Virtualenv

If you are a pythoner, you should install lilac via **virtualenv**.

If virtualenv is not installed on your OS, install it:

```bash
$ sudo pip install virtualenv
```

Once you have virtualenv installed, open a shell and create a new environment for your blog:

```bash
$ mkdir MyBlog
$ cd MyBlog
$ virtualenv venv
New python executable in venv/bin/python
Installing setuptools...............done.
Installing pip....................done.
```

Now, whenever you want to write blog, you have to activate the corresponding environment:

```bash
$ . venv/bin/activate
```

Then, install lilac in your virtualenv:
$ pip install lilac

1.2.3 System-Wide Installation

Just run pip with root privileges:

$ sudo pip install lilac

1.2.4 Get the Code

Lilac is hosted on GitHub, where you can get the latest code.
You can clone the repository:

$ git clone git://github.com/hit9/lilac.git

Or, download the tarball.

Once you get the code, you can install it into your site-packages:

$ [sudo] python setup.py install

1.2.5 Upgrade your Lilac

Whenever you want to upgrade your lilac to latest version:

$ [sudo] pip install lilac --upgrade

You may want to see the Quick Start part now.

1.2.6 Installation trouble?

If you meet problems similar to this: Cann't find Python.h, try to install Python development libraries.
If you’re using Ubuntu, install it via:

$ [sudo] apt-get install python-dev

There’s no this issue on OS X.

1.3 Quick Start

Eager to get started? Now this section assumes that you have lilac installed.

1.3.1 Create a New Blog

You need to create a new directory for your new blog, and then run lilac deploy in the empty directory:

$ mkdir MyBlog
$ cd MyBlog
$ lilac deploy
You may want to run `ls` to see what has happened:

```
$ ls
classic  config.toml  Makefile  src  venv
```

After this step, lilac generates you a sample config file `config.toml`, a sample post file `src/post/sample.md`, and the default theme `classic`.

And, lilac generates a `Makefile` to manage blog.

Now, what to do next is configuring our fresh blog.

### 1.3.2 Configure your Blog

The configuration is dead minimal, let’s take a look at the `config.toml`:

```
root_path = ""

[blog]
name = "Make Difference"
description = "Here goes your blog’s description"
url = "http://your-blog.org"
theme = "classic"

[author]
name = "me"
email = "me@some.com"

[disqus]
shortname = "your-disqus-short-name"

[theme.vars]
github = "your-github-username"
```

The configuration is in TOML, but you really needn’t to learn this language, it is obvious and simple.

Now you need to edit the sample configuration generated by lilac.

Note: **You need to set `root_path` only if you are deploying your blog to sub_directory**

To learn more about configuration, see also `configuration`.

### 1.3.3 Choose a Theme

Don’t like the classic theme? Go to the theme list to choose one.

Just clone the theme repo down, for example, the theme `less`:

```
$ git clone git://github.com/hit9/lilac-theme-less.git less
```

and then set “theme” to “less” in config.toml:

```
theme = "less"
```

See `themes` to learn more.
1.3.4 Create a New Post

Each time you start with a new post, you need to touch a new file under directory `src/post`:

```
$ vim src/post/Hello-World.md
```

Here is a sample post:

```markdown
title = "Hello World!"
datetime = "2013-06-05 17:12"
-------
# Hello World!
```

You can go to *Post’s Syntax* to learn more.

1.3.5 Build Site

You can fire up another shell to run lilac’s server:

```
$ make serve
```

This command will start a web server here and tell lilac to start watch file changes - once you update source files, lilac will auto rebuild the blog.

To see build results in action, point your browser at http://localhost:8888

For more about building server, see command `serve`

1.3.6 Writing Steps

Each time writing a new post, you just do these steps:

1. activate the environment if you install lilac in virtualenv.
2. run lilac’s server.
3. touch a new file under `src/post/` and then start writing.
4. see results in browser after doing a save.

But, from v0.3.7, we have built-in tool *ililac*, and writing becomes more easy:

1.3.7 Use ililac

*ililac* runs lilac’s server and rebuilder in the background, each time we start writing, just:

```
$ cd myblog
$ ililac start
```

remember to stop the daemon when finish writing:

```
$ ililac stop
```

We just need at most one shell session to blog.

1.3.8 What’s Next?

I think reading these will be very helpful: *Post Syntax, Commands, Configuration.*
1.4 Configuration

1.4.1 Overview

There are only a few items to configure. A minimal configuration is:

```yaml
root_path = ""

[blog]
name = "Make Difference"
description = "Here goes your blog’s description"
url = "http://your-blog.org"
theme = "classic"

[author]
name = "me"
email = "me@some.com"

[disqus]
shortname = "your-disqus-short-name"
```

1.4.2 Root Path

If you are deploying your blog right under the server’s root, you needn’t to set this item, leave it blank:

```yaml
root_path = ""
```

Else, if you are deploying your blog to some sub_directory under the server, you need to configure this item. For instance, your blog are deployed here:

```
http://my-server.com/myblog/
```

you should set `root_path` to “/myblog”:

```yaml
root_path = "/myblog"
```

But note that: when in localhost, your site will run regardless of `root_path`, so you must run `lilac build` before deploying this site to remote server.

1.4.3 Blog & Author

It’s easy enough to configure these by yourself.

Notes you need to know:

- the item `url` in section `blog` is only used in feed generation.
- better to set `author`’s `email` to your Gravatar’s email.

1.4.4 Disqus

Lilac uses Disqus to manage blog’s comments.

Follow this link to [register your blog to disqus.com](http://disqus.com), and then set item `shortname` in the section `disqus`:
[disqus]
shortname = "your-short-name-from-disqus.com"

### 1.4.5 Theme Vars

This section configure your theme. We configure theme’s variables in config.toml instead of your_theme/theme.toml so that we can use theme as a standalone repo(or submodule).

What to configure depends on your theme.

### 1.5 Post Syntax

#### 1.5.1 Overview

A post is made up of two parts: header and body.

The header is in TOML and body is in Github Flavored Markdown, the two parts are separated with a ‘—’ like separator.

A sample post is:

```toml
title = "Hello World"
datetime = "2013-06-05 19:38"
tags = ["sample", "some-tag"]
----------
# Here is markdown content
```

#### 1.5.2 Post’s Filename Extension

Post’s filename extension should be ".md”, for example, “my-post.md”.

#### 1.5.3 Post’s Header

The post’s header is in TOML.

The header part contains post’s information: title, tags, created time, etc.

The required items are *title* and *datetime*, others like *tags*, *summary* are optional.

- the *datetime* is a “%Y-%m-%d %H:%M” formatted string, it’s the post’s created time.
- the *tags* is an array of tags.
- the *summary* is your post’s summary(default: the post’s first certain count characters).

Other variables in header can be got in template:

```toml
{{post.myvar}}
```

#### 1.5.4 Post’s Body

The body is in Markdown. I recommend your this link to learn markdown in minutes: [Markdown CheatSheet](#).
1.5.5 The separator

Post’s header and body are separated with a separator:

----

Or longer:

-------------

It’s at least 3 `-` long.

1.6 Themes

There isn’t a process called “installation” for lilac’s themes, just tell lilac where the target theme is by editing item `theme` in `config.toml`:

```
[blog]
theme = "the-theme-path"
```

Generally, the themes are under the root of your blog’s directory.

1.6.1 Use Theme

You really should manage your theme in a standalone git repository, and use it as a submodule of your blog’s submodule if your blog is under git versioning too.

For instance, add theme `less` a submodule of your blog’s repo:

```
$ git submodule add git://github.com/hit9/lilac-theme-less.git less
```

If you want to modify a theme created by someone else, just fork his(or her) repo, and then modify it.

But it’s 100% ok to use themes not in the submodule way.

1.6.2 Theme List

Here I maintain a lilac’s themes list:

- `less` - a clean theme for lilac, default theme. by hit9
- `classic` - the old default theme for lilac. by hit9
- `pure` - by kshiftlv

The built in theme is classic, also the demo site’s theme.

Have you made one? Please send a pull request on lilac’s Repo, append yours to this list.

1.7 Commands

You’re going to see how simple its command line interface is.
1.7.1 Overview

All command line usage:

Usage:
  lilac [-h|-v]
  lilac build
  lilac deploy
  lilac clean
  lilac serve [<port>] [--watch]

Options:
  -h --help       show this help message
  -v --version    show version
  --watch         watch source files for changes
  <port>          which port for server to use (default: 8888)

Commands:
  deploy          deploy blog in current directory
  build           build source files to htmls
  clean           remove files built by lilac
  serve           start a web server, as a option, start watching

Tools:
  ililac           run lilac’s server and rebuilder as a daemon running in the background

1.7.2 Options

Show help:

$ lilac --help

Show version:

$ lilac --version

1.7.3 Deploy

To deploy a new blog in new-created directory:

$ lilac deploy

1.7.4 Build

To build site from source to htmls, lilac will be honest to config.toml:

$ lilac build

1.7.5 Clean

To remove all htmls(include the feed.atom) lilac built:
$ lilac clean

This command is equivalent to:
$ rm -rf post page tag 404.html about.html archives.html feed.atom index.html tags.html

### 1.7.6 Serve

To start a simple HTTP server:
$ lilac serve

You can tell lilac which port to use (the default port is 8888):
$ lilac serve 8080

To watch source changes the same time when the cute web server running:
$ lilac serve --watch

When you save your writings, lilac can detect the changes and start rebuilding.

Note: the preview serve will run the site from root the regardless of config root_path.

### 1.7.7 ililac

ililac is a tool to run lilac’s server and rebuilder in the background.

$ cd myblog
$ ililac start

to stop the daemon:
$ ililac stop

With this tool, we can write blog with at most one shell session.

Note: ililac is included into lilac in version 0.3.7

### 1.8 Blog’s Structure

The blog’s structure (without the built htmls):

MyBlog/
  -- theme/
  -- src/
  |   -- post/
  |   |   -- sample.md
  |   |   -- my-post.md
  |   -- about.md
  -- config.toml

You can see the whole files structure: [Demo site on GitHub](https://github.com/lilac-lilac/lilac)
1.9 The Makefile helper

Lilac generate you a Makefile to help your managing the blog, let’s take an quick overview on its usage:

Start blogging(start server and auto rebuild):

$ make serve

Build blog:

$ make build

Clean htmls:

$ make clean

1.10 Use Cases

1.10.1 Github Pages

Thank GitHub, we have a great place to host static blogs.

If you want to learn about Github Pages, head over to GitHub Pages.

If you create a repository named username.github.com (now it’s username.github.io), its master branch will be served on GitHub Pages.

If you create a branch named gh-pages on any repository, this branch will be served on GitHub Pages.

GitHub Pages is a static files HTTP server, free to use and no bandwidth limit.

Here’s a short tutorial to use lilac on github pages.

Create a repo for blog on GitHub, and then open a shell:

$ mkdir you.github.com && cd you.github.com
$ git init
$ git remote add origin git@github.com:you/you.github.com.git

It’s a good habit to ignore trash files:

$ vim .gitignore

We should add this to .gitignore:

.*.swp
.*.swo
venv/

Now make the first commit:

$ git add .gitignore
$ git commit -m ‘init commit’
$ git push origin master

It’s time to deploy lilac:

$ lilac deploy
I recommend you to add theme as git submodule(need to remove the auto generated theme directory):

```
$ rm classic -rf
$ git submodule add git://github.com/hit9/lilac-theme-classic.git classic
```

You may take a look at *Quick Start* for customization.

And start writing:

```
$ make serve
```

Let’s deploy the site to GitHub’s Pages:

```
$ git add .
$ git commit -m ‘deploy my blog’
$ git push origin master
```

GitHub will send you an email once your blog is ready.

You can set up a custom domain with github pages, just define a file *CNAME* for your site, and write your domain in it (see here: *CNAME*), after this, create a CNAME record pointing to `your-github-username.github.io`. on the domain panel, more information is on github pages help.

### 1.10.2 Host Blog in Sub Directory

If your blog must run from a sub-directory, the main scenario being if your blog will be hosted on a GitHub Project Page, take a look at this configuration: `root_path`.

### 1.11 Contribute

Found a bug? Have a good idea for improving Lilac? You can fork lilac’s repo on GitHub and then send a feature pull request, or you can visit lilac’s issues tracker on GitHub to report bugs, that will help all users. Welcome for your feedback.