Mkdocs Tutorial and Template

Making writing documentations easier!

Frinze Erin Lapuz

Table of contents

1. Welcome to System Health Lab MkDocs Tutorial and Template

This is a tutorial and template based from Mkdocs Frinze Template. This is a template that contains extensions that are very nice to have when you just want a standard documentation for anything!

For full documentation visit:

- mkdocs.org for the generic MkDocs
- PyMdown Extensions for the different extensions that are installed
- MkDocs Material for the customisation of the web server documentation.

1.1 Examples of Other Documentations that uses this template

- IndEAA (Industrial Engineers Australia Assessment) Web Application used for streamlining course accreditation review
- ASER (Asset Equipment Registry) Web Application used for accessible equipment registry
- Living Lab UWA Documentation used for technical documentation on accelerated life testing

1.2 Why documentation?

Part of the success of every project is its maintainability, and that means that ability to pass on the knowledge and technical details to the people that will carry on the work.

1.3 What is Markdown and what is Mkdocs?

Markdown is a simplistic markup language that is used to write documentations with a file that ends with `.md`. The greatest thing about markdown is its simplicity, this allows it to be rendered in many formats - `.docx`, `.pdf`, `.tex`, and with the case of Mkdocs, to render websites. Mkdocs is simply a renderer for markdown that generates files essential for websites (HTML, CSS, JS). These files allows the possiblity of deploying markdown documents into your own websites (in servers or external providers such as github pages).

There are a lot of places to learn how to write markdown, and due to its simplistic design, it is relatively easy to learn. Below the summary of a guide made by Github.

> **Alternative**
>
> There are a lot of alternatives with MkDocs in the realms of markdown-based documentation such as gitbook, confluence, github wiki, and docusaurus.
>
> In the side of other ways for documentation:
>
> ```
> - Onenote
> - Word Documents in OneDrive/Google Drive
> ```

1.4 What do I hope to achieve with this tutorial and template?

This tutorial and template has 2 main purpose:

1. Make the documentation setup easier and accessible for everyone (template)
2. Teach Markdown (tutorial)

1.5 How easy is this to deploy?

1. Clone This Repo or press the big green button "Use this template"
2. Follow the installation
3. Delete the markdown files here and replace it with your own
4. Change a couple of things in the `mkdocs.yml` file (there are comments around it to make it easier)
5. Modify the `nav` in the `mkdocs.yml` file or delete it (Mkdocs will sort you documentation files to display)
6. Deploy somewhere ! (easiest way is with Github Pages see here)

> **Branch Name**
>
> When you press "Use This template", the new repository will have "template" in its name. Change that in the Github >> Settings >> Branches >> Default Branch >> "Click the pencil icon".

> **Private Repositories Github Pages**
>
> When you create a private repository, by default, your website will be flagged as "ready to be published". To publish the website, you have to go to Github >> Settings >> Pages >> "Change the Source Branch to `gh-pages`" >> Press Save

1.6 Installation

> **Prerequisite**
>
> You need to have Python installed to be able to use `pip`. There are a few ways of installing Python. You can use a package distributor like Anaconda Or you can just install Python.

Once you have installed Python, install mkdocs requirements by opening a terminal and typing:

```
pip install -r requirements.txt
```

> **Python Environments (Optional)**
>
> however, it is good practice to use different environments for different purposes, in which case, for Anaconda, you would open a terminal and type:
>
> ```
> conda create -n mkdocstutorial python
> conda activate mkdocstutorial
> ```
>
> then enter:
>
> ```
> pip install -r requirements.txt
> ```

1.7 Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server. Very helpful when you want to take a look at the docs before deploying.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.
- `mkdocs gh-deploy` - Deploy in github pages

1.8 Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

1.9 Extending this template

This template is made to be simple such that it gives you a brief overview of how you would be writing your documentation with a few configuration. This is the type of documentation that you just build on top of.

If in the scenario that you feel that I missed that is essential to be in the template, please see Contributions Section. However, if you feel that you would like to extend this template much more, I would highly recommend to visit the original Mkdocs Material Documentation.

1.9.1 Contributors

All thanks to the following contributors for maintaining this:

- Frinze Lapuz
- Ben Travaglione
- Melinda Hodkiewicz

1.10 About this tutorial

There are 4 main portion of this tutorial, which are ordered sequentially:

1. Overview and Installation of Mkdocs (the current documentation you are looking at)
2. Writing Markdown
3. Flavoured Markdown
4. Deployment and Automated Deployment

2. Writing Markdown

2.1 Github Guide

Here’s an overview of Markdown syntax that you can use anywhere on GitHub.com or in your own text files.

2.1.1 Headers

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

2.1.2 Emphasis

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

_You **can** combine them_
```

2.1.3 Lists

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
1. Item 2
1. Item 3
   1. Item 3a
   1. Item 3b
```

2.1.4 Images

```
![GitHub Logo](/images/logo.png)
Format: ![Alt Text](url)
```

2.1.5 Links

```
http://github.com - automatic!
[GitHub](http://github.com)
```

2.1.6 Blockquotes

```
As Kanye West said:

> We're living the future so
> the present is our past.
```

2.1.7 Inline code

```
I think you should use an
`<addr>` element here instead.
```

3. Flavoured Markdown

Flavoured markdown is a type of markdown that is customised with different syntax to provide a more stylistic documentationn. As you know, markdown is great for its simplicity, however this also makes it inflexible when making more stylistic documentation such as creating Math formula with Latex and many more feature that you wil see here. With Flavoured Markdown, there are certain syntax to follow on top of the original markdown syntax.

Below is a **very very small** overview to whatMkdocs Material - the base extension . I will just highlight some of them, because those are the only documentation syntax that is commonly use and usually remembered.

3.1 Admonitions

These are kind of those fancy boxes that you usually in cool Science Books that adds extra information.

Note

As you can see this box, is very attractive.

The syntax for this is:

```
!!! note
As you can see this box, is very attractive.
```

What If You want a different Title

The syntax for this is:

```
!!! note "What If You want a different Title"
As you can see this box, is very attractive.
```

3.1.1 Icons

More info here

You can also change these icons by changing the first word after `!!!` or `???`.

`note`, `seealso`

Note

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`abstract`, `summary`, `tldr`

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`info` , `todo`

> **Info**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`tip` , `hint` , `important`

> **Tip**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`success` , `check` , `done`

> **Success**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`question` , `help` , `faq`

> **Question**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`warning` , `caution` , `attention`

> **Warning**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`failure` , `fail` , `missing`

> **Failure**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`danger` , `error`

> **Danger**
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`bug`

Bug

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`example`

Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

`quote`, `cite`

Quote

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

3.1.2 Collapsible Block

More info here

If things are getting a little bit crowded, why not make some of them collapsible?

Example of a More Complex Documentation

Here is the basic idea of bubble sort!

```
def bubble_sort(items):
    for i in range(len(items)):
        for j in range(len(items) - 1 - i):
            if items[j] > items[j + 1]:
                items[j], items[j + 1] = items[j + 1], items[j]
```

The Syntax for the Example Above

````
    ??? Example "Example of a More Complex Documentation"
    Here is the basic idea of bubble sort!
    ```python
    def bubble_sort(items):
        for i in range(len(items)):
            for j in range(len(items) - 1 - i):
                if items[j] > items[j + 1]:
                    items[j], items[j + 1] = items[j + 1], items[j]
    ```
````

3.2 Code Highlight

This is powered by codehilite. Whenever, you need code, this is the one that makes it pretty.

For example:

```
def bubble_sort(items):
    for i in range(len(items)):
        for j in range(len(items) - 1 - i):
            if items[j] > items[j + 1]:
                items[j], items[j + 1] = items[j + 1], items[j]
```

Syntax of the Example Above

```
    ``` python linenums="1"
    def bubble_sort(items):
        for i in range(len(items)):
            for j in range(len(items) - 1 - i):
                if items[j] > items[j + 1]:
                    items[j], items[j + 1] = items[j + 1], items[j]
    ```
```

3.2.1 Highlight Specific Code Lines

What if I want to show some cool lines? I could highlight which specific line number should be highlighted.

```
def bubble_sort(items):
    for i in range(len(items)):
        for j in range(len(items) - 1 - i):
            if items[j] > items[j + 1]:
                items[j], items[j + 1] = items[j + 1], items[j]
```

Syntax of the Example Above

```
    ``` python hl_lines="2 3"
    def bubble_sort(items):
        for i in range(len(items)):
            for j in range(len(items) - 1 - i):
                if items[j] > items[j + 1]:
                    items[j], items[j + 1] = items[j + 1], items[j]
    ```
```

3.3 Latex / Math Symbol Renderer

This is for math nerds that needs some Maths in their documentation. More info on Latex here.

For example, the Pythagoras Theorem $$ a^2 + b^2 = c^2 $$

Syntax of the Example Above

```
$$ a^2 + b^2 = c^2 $$
```

3.3.1 Inline Latex

According to the results with the p-value $p < 0.05$, it means that we will reject the null Hypothesis H_0, and that there is a significant difference in the means.

3.4 Footnotes

Woah woah woah! Getting a little bit nerdy referencer here!

"You can tell that I don't know much about referencing"[1]. If you click this shiny number, it takes you to the bottom of the page where the reference is.

Syntax of the Example Above

```
"You can tell that I don't know much about referencing"[^1]

[^1]:
Book of Wisdom - John Doe
```

3.5 Content Tabs

Very useful for when you need one or the other.

For example, when dealing with multiple programming languages.

C

```c
#include <stdio.h>

int main(void) {
  printf("Hello world!\n");
  return 0;
}
```

C++

```cpp
#include <iostream>

int main(void) {
  std::cout << "Hello world!" << std::endl;
  return 0;
}
```

Syntax of Above

```
    === "C"

        ``` c
        #include <stdio.h>

        int main(void) {
        printf("Hello world!\n");
        return 0;
        }
        ```

    === "C++"

        ``` c++
        #include <iostream>

        int main(void) {
        std::cout << "Hello world!" << std::endl;
        return 0;
        }
        ```
```

3.6 Icons and Emoji

Just worth mentioning, not too sure if you're going to use it.

- – `.icons/material/account-circle.svg`
- – `.icons/fontawesome/regular/laugh-wink.svg`
- :octicons-octoface-16: – `.icons/octicons/octoface-16.svg`

Syntax of Above

```
- :material-account-circle: – `.icons/material/account-circle.svg`
- :fontawesome-regular-laugh-wink: – `.icons/fontawesome/regular/laugh-wink.svg`
- :octicons-octoface-16: – `.icons/octicons/octoface-16.svg`
```

3.7 Images

Can be done with Markdown or HTML.

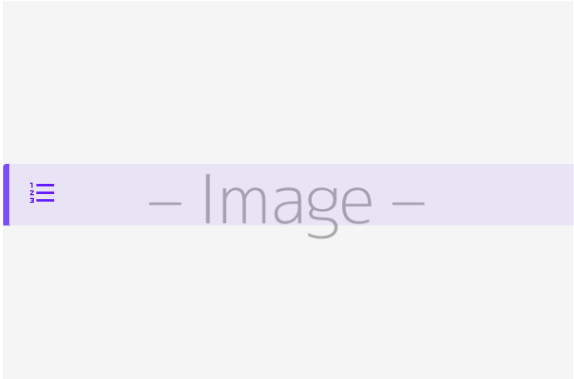
3.7.1 Image Captioning

The Logo that Daphne from Coders for Causes gave me

Syntax of Above

```
<figure>
    <img src="../images/docs.png" width="100" />
    <figcaption>The Logo that Daphne from Coders for Causes gave me</figcaption>
</figure>
```

3.7.2 Image Alignment



This is for when you have paragraphs and some text, but you really wanted those fancy images on the side. You can either say `left` or `right`. Now Let me just fill this with some random words so that the image doesn't look to weird.

Syntax Above

```
![Placeholder](https://dummyimage.com/600x400/f5f5f5/aaaaaa&text=-%20Image%20-){: align=left width=300 }

This is for when you have paragraphs and some text, but you really wanted those fancy images on the side. You can either say `left` or `right`. Now Let me
just fill this with some random words so that the image doesn't look to weird.
```

3.8 Graph In Markdown / Mermaid Markdown

More Information here.

What if you really just want to create some fancy graphs, but you really can't be bothered to:

1. Load some other software
2. Draw this graph that you wanted to show
3. Save this graph that you want to show
4. Upload this graph somewhere
5. Link this image back to this documentation

Like there are just soooo many steps.

Introducing **mermaid markdown**.

graph TD A --> B & C B --> C

Syntax for Above

```
    ```mermaid
    graph TD
        A --> B & C
        B --> C
    ```
```

How about more complex ones? Is this complex enough for your

graph TD A[Hard] -->|Text| B(Round) B --> C{Decision} C -->|One| D[Result 1] C -->|Two| E[Result 2]

Syntax for Above

```
    ```mermaid
    graph TD
        A[Hard] -->|Text| B(Round)
        B --> C{Decision}
        C -->|One| D[Result 1]
        C -->|Two| E[Result 2]
    ```
```

3.8.1 Some Examples of Other Charts

Sequence Diagram

Result

sequenceDiagram participant Alice participant Bob Alice->>John: Hello John, how are you? loop Healthcheck John->>John: Fight against hypochondria end Note right of John: Rational thoughts
prevail! John-->>Alice: Great! John->>Bob: How about you? Bob-->>John: Jolly good!

Syntax

```
    ```mermaid
    sequenceDiagram
    participant Alice
    participant Bob
    Alice->>John: Hello John, how are you?
    loop Healthcheck
        John->>John: Fight against hypochondria
    end
    Note right of John: Rational thoughts <br/>prevail!
    John-->>Alice: Great!
    John->>Bob: How about you?
    Bob-->>John: Jolly good!
    ```
```

Gantt Chart

Result

gantt dateFormat YYYY-MM-DD title Adding GANTT diagram to mermaid excludes weekdays 2014-01-10 section A section Completed task :done, des1, 2014-01-06,2014-01-08 Active task :active, des2, 2014-01-09, 3d Future task : des3, after des2, 5d Future task2 : des4, after des3, 5d

Syntax

```
    ```mermaid
    gantt
    dateFormat  YYYY-MM-DD
    title Adding GANTT diagram to mermaid
    excludes weekdays 2014-01-10

    section A section
    Completed task            :done,    des1, 2014-01-06,2014-01-08
    Active task               :active,  des2, 2014-01-09, 3d
    Future task               :         des3, after des2, 5d
    Future task2               :         des4, after des3, 5d
    ```
```

Class Diagram

Result

classDiagram Class01 <|-- AveryLongClass : Cool Class03 *-- Class04 Class05 o-- Class06 Class07 .. Class08 Class09 --> C2 : Where am i? Class09 --* C3 Class09 --|> Class07 Class07 : equals() Class07 : Object[] elementData Class01 : size() Class01 : int chimp Class01 : int gorilla Class08 <--> C2: Cool label

Syntax

```
    ```mermaid
    classDiagram
    Class01 <|-- AveryLongClass : Cool
    Class03 *-- Class04
    Class05 o-- Class06
    Class07 .. Class08
    Class09 --> C2 : Where am i?
    Class09 --* C3
    Class09 --|> Class07
    Class07 : equals()
    Class07 : Object[] elementData
    Class01 : size()
    Class01 : int chimp
    Class01 : int gorilla
    Class08 <--> C2: Cool label
    ```
```

---

1. Book of Wisdom - John Doe ↵

4. Site Deployment

As said previously, Mkdocs allows conversion of `.md` to `HTML`, `CSS`, and `JS` files in order to create and deploy websites. Below are the possible approaches.

4.1 Site Deployment with Github

Assuming that your repository is in github, when an Mkdocs repository is created, simply type

```
mkdocs gh-deploy
```

Just follow the prompt, and it will automatically deploy your website in github pages. No other complication needed, this command will handle the following steps:

1. Generation of website files
2. Deployment with Github Pages

4.1.1 Automatic Site Deployment with Github Action

This is a configuration which allows your documentation from github to auto-deploy to the github pages. You might not want to run `mkdocs gh-deploy` everytime you have new changes.

> **Why do I need this?**
>
> Let me give you an example, for this documentation it is hosted at https://uwasystemhealth.github.io/shl-mkdocs-tutorial-and-template/. If this thing is configured, then whenever you modify the github repository, it automatically redeploys in github pages.

How do I do this?

If you look closely in the repository, there is a file `.github/workflows/main.yml`. Copy this file over to you repository. The content of it is roughly like below. Note that you have to change 2 lines highlighted to the path of your documentation.

Example of Path that you will have to change

If in the scenario that your repository is just documentation, which means that your `mkdocs.yml` file is in the root, then you don't have to change anything.

However, there are cases where you have a monorepo - a type of repository that contains multiple files such as for example in a software project, there are the documentation files, and source code files. It is quite common to have a file structure that looks like this:

```
frontend/
    ...
backend/
    ...
mkdocs/
    mkdocs.yml    # The configuration file.
    docs/
        index.md  # The documentation homepage.
        ...       # Other markdown pages, images and other files.
```

This means that you will have to change the one highlighted. From the example above here, the correct lines changes are:

`key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}` to `key: ${{ runner.os }}-pip-${{ hashFiles('**/mkdocs/requirements.txt') }}`

and

`python3 -m pip install -r ./requirements.txt` to `python3 -m pip install -r ./mkdocs/requirements.txt`

```
# Workflow for deploying to github
name: Publish docs via GitHub Pages
on:
  push:
    branches:
      - master
      - main
      - mkdocs-experimental
  workflow_dispatch:

jobs:
  deploy:
    name: Deploy docs
    runs-on: ubuntu-latest
    steps:
      - name: Checkout main
        uses: actions/checkout@v2

      - name: Setup Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.8'

      - name: Upgrade pip
        run: |
          # install pip=>20.1 to use "pip cache dir"
          python3 -m pip install --upgrade pip
      - name: Get pip cache dir
        id: pip-cache
        run: echo "::set-output name=dir::$(pip cache dir)"

      - name: Cache dependencies
        uses: actions/cache@v2
        with:
          path: ${{ steps.pip-cache.outputs.dir }}
          key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}
          restore-keys: |
            ${{ runner.os }}-pip-
      - name: Install dependencies
        run: python3 -m pip install -r ./requirements.txt

      - run: mkdocs build
        env:
          ENABLE_PDF_EXPORT: 1

      - name: Deploy
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./site
```

4.1.2 Custom Domain Name

In the scenario that you like a custom domain name such as https://www.tutorial-mkdocs.systemhealthlab.com , follow this documentation.

Simplified instructions:

1. Go to the domain registar, in my case Cloudflare
2. Register a "CNAME" of the domain/subdomain going towards `<organisation/githubname>.github.io` (eg. `uwasystemhealth.github.io`)
3. Add a `CNAME` file with the name of the domain/subdomain in the `/docs` folder
4. Give the `CNAME` file a content of the subdomain name (eg. `tutorial-mkdocs.systemhealthlab.com`)

4.2 Custom Site Deployment

Let say you don't want to deploy it in github pages. You would like to deploy it elsewhere such as your own server or a VPS. Be aware that this portion is a little bit technical, and may not even be what you do in a regular basis or not necessary if you already deployed it with Github Pages. You have to type this command

```
mkdocs build
```

This will create the `/site` folder which contains your website files. Now you would have to setup a server application that serves static files such as `NGINX` or `Apache` server app. After setting this up, copy the contents of the `/site` folder into the static file folder.

5. Contributions

Hi! I am happy that you thought of contributing! If you have any suggestions or issues, please raise it here. I would be happy if you could provide pull requests, if you know how to do it here. Also add your name in the contributors section.

5.1 Structure

5.1.1 Folder Structure

The structure of this repo is as follows:

```
├── docs                    // Folders for documentation
│   ├── CNAME
│   ├── contributions.md
│   ├── deployment_and_automated_site_deployment.md
│   ├── flavoured_markdown.md
│   ├── images              // Assets
│   │   ├── shl.png
│   │   └── shl_with_text.png
│   ├── index.md
│   └── writing_markdown.md
├── LICENSE
├── mkdocs.yml              // MkDocs Configuration
├── overrides
│   └── partials
│       └── footer.html
├── README.md
└── requirements.txt
```

5.1.2 Branch Structure

You might have noticed that there are a couple of branches that are important:

- `gh-pages` is the branch that contains the GitHub Pages build as per requirement of a website hosted in Github Pages
- `main` is the branch that contains the `main` branch for any development and contribution
- `template` is the branch that is in the front page of the Github Repo. This is the branch where all repositories will inherit the template.

> **Why have `template` and `main` separate?**
>
> One of the things that you could see in `main` is the file called `CNAME` this is configuration for the domain name. Templates that inherit the files should not have this, otherwise there would be conflict in the organisation deployment for domain name.
>
> Furthermore, by separating this, the changes could be version batched before releasing it in `template`.