Assembled Documentation Content

This file contains the combined body content from all HTML files in your Jekyll _site directory.

---

assembled_docs_body.html

Assembled Documentation Content

This file contains the combined body content from all HTML files in your Jekyll _site directory.

---

assembled_docs_body.html

Assembled Documentation Content

This file contains the combined body content from all HTML files in your Jekyll _site directory.

---

index.html

Documentation

- Overview
- Jekyll system

Welcome to the DevEx Starter Template docs

This project has three main sections:

- **Jekyll-based documentation content** — Markdown -> Jekyll -> HTML/JS/CSS -> GitHub Pages
- **Python tools** — Validate the Markdown, links, and assets of the documentation content
- **CI/CD flow** — GitHub Action workflows to run the Python tools before rebuilding and updating the Docker image, and before pushing doc changes to GitHub Pages.

GitHub repository

Since this repository is a **DevEx Starter Template**, the first step is to create your own independent copy by forking the repository.

1. Create a server-side fork (web)
    1. Navigate to the DevEx Starter Template GitHub page.
    2. Click the **Fork** button to create a copy of the repository under your personal account.
2. Clone your fork (command line)

    After the fork is complete, clone **your new copy** to your local machine. This is the sole remote (`origin`) for your project going forward.

```
# Replace YOUR_USERNAME and YOUR_REPO with your new fork details
git clone https://github.com/YOUR_USERNAME/YOUR_REPO.git
cd YOUR_REPO
```

Your project is now initialized. All changes, pushes, and branches are managed within your new repository.

Jekyll-based doc content

Jekyll is a static site generator that transforms your Markdown content into a complete website. In this project, Jekyll configuration files are located in the `/docs` directory.

This section covers the Jekyll-based documentation content, including Markdown, Jekyll, HTML/JS/CSS, and GitHub Pages.

Markdown

Use Markdown to create the content for your documentation. It is a lightweight markup language with plain text formatting syntax. In this project, Markdown files are located in the `/docs/topics` directory.

You should pick a style guide for your Markdown content to ensure consistency. This project follows the Google Developer Documentation Style Guide style guide for technical documentation.

Jekyll

Jekyll processes the Markdown files and applies templates to generate a static website. The Jekyll configuration file (`_config.yml`) and layout files are located in the `/docs` directory.

You need to install Jekyll and its dependencies to build the site locally. Follow the Jekyll installation guide for your operating system.

For full details on how the Jekyll system is set up in this project, see Jekyll system.

HTML/JS/CSS

Format your static website pages with JavaScript and CSS to enhance the user experience of your documentation site.

These files are in the `/docs/assets/css` and `/docs/assets/js` directories.

GitHub Pages

GitHub Pages hosts your Jekyll-generated static site directly from your GitHub repository. The site is automatically built and deployed whenever you push changes to the repository.

To enable GitHub Pages for your repository, go to the repository **Settings**, navigate to the **Pages** section, and select the source branch and folder (usually `main` branch and `/docs` folder).

Optionally, you can configure a custom domain for your GitHub Pages site in the same **Pages** section of the repository **Settings**.

Python tools

This project includes Python tools to validate the Markdown, links, and assets of the documentation content.

FastAPI and Swagger docs

The FastAPI application provides automatic API documentation using Swagger and ReDoc.

The FastAPI app is located in the `/src/api.py` file.

You can see an example of Swagger docs for the deployed instance of this project at this Lightsail container.

Typer CLI

The Typer command-line interface (CLI) allows you to run the validation tools from the command line.

The Typer CLI is located in the `/src/cli.py` file.

To run the CLI, navigate to the project root directory and use the following command:

```
python -m src.cli <command> [options]
```

CI/CD flow

When you push changes to the repository, GitHub Actions workflows are triggered to run the Python validation tools before rebuilding and updating the Docker image and before updating GitHub Pages with doc changes.

GitHub actions

GitHub Actions workflows are defined in the `.github/workflows` directory.

The **DevEx Starter Template** includes the following workflows:

- **Check Links** — This workflow runs `htmlproofer` to check for broken links in the generated site whenever changes are pushed to the `main` branch.
- **Generate PDF** — This workflow generates a PDF version of the documentation site using `wkhtmltopdf` whenever changes are pushed to the `main` branch.
- Others coming soon...

Docker

The Dockerfile and Docker Hub repository are used to build and store the Docker image for your project. The Dockerfile is located in the root of the repository, and the Docker Hub repository should be created under your Docker Hub account.

Deploy image

You can deploy your image to a cloud service like AWS Lightsail Container Service, AWS ECS, or any other container orchestration platform.

---

jekyll-system.html

Documentation

- Overview
- Jekyll system

Jekyll system

```
/docs
    ├── _config.yml          # Jekyll configuration file
    ├── _data                # Data files for Jekyll
    │     └── tree.yml       # Site structure for sidebar navigation
    ├── _layouts             # Jekyll layout templates
    │     └── page.html      # Layout for documentation pages
    ├── _includes            # Jekyll includes (reusable components)
    ├── _site                # Generated site (output)
    ├── assets               # CSS, JS, images
```

```
│     ├── css
│     │    └── main.css   # Main stylesheet
│     ├── js
│     │    └── main.js    # Main JavaScript file
│     └── images          # Image assets
├── topics                # Markdown documentation content
│     └── <your-docs>.md  # Your documentation files
└── index.md              # Home page content
```

- `_config.yml`: This file contains configuration settings for the Jekyll site, and is also the place to define external resources such as plugins and themes.

  The current configuration defines the minimum settings: where the source files are located and the destination for the generated site.

- `_data/tree.yml`: This file defines the structure of the documentation site for generating the sidebar navigation. It uses a simple YAML format to list the titles and URLs of the pages in the documentation.

- `_layouts`: This directory holds layout templates that define the structure of your documentation pages. The `page.html` layout is the default for rendering individual documentation pages.

- `_includes`: This directory contains reusable components that can be included in multiple pages, such as headers, footers, navigation elements, and boilerplate text. From your content, you include these components with Liquid tags, such as `{% include header.html %}`.

- `_site`: This is the output directory where Jekyll generates the static site.

- `assets`: This directory contains static assets like CSS, JavaScript, and images used in the documentation.

- `topics`: This directory contains the Markdown files for your documentation content.

- `index.md`: This file serves as the home page for your documentation site.

A note on underscores: In Jekyll, directories and files that start with an underscore (_) are not generated in the static site, but Jekyll uses them for templating and configuration. You can also create your own directories and files that start with an underscore for work in progress or topics you want to remove from the released documentation but keep for future work, or simply store internal notes in them.

For more information on Jekyll's structure and conventions, refer to the Jekyll documentation.

Generating and previewing the site

To run a local Jekyll server for previewing your documentation site, use the following command in your terminal:

```
jekyll server
```

This command starts a local server and watches for changes in your files, allowing you to see updates in real-time as you edit your documentation. By default, the server is accessible at `http://localhost:4000`.

You can also build the site without starting a server using:

```
jekyll build
```

This command generates the static site in the `_site` directory without serving it. You can then deploy the contents of the `_site` directory to your web server or hosting service, or as in this DevEx Starter Template examples, doing a `git push` automatically starts a GitHub Action that updates the GitHub Pages site with the updated content.

- Overview
- Jekyll system

Welcome to the DevEx Starter Template docs

This project has three main sections:

- **Jekyll-based documentation content** — Markdown -> Jekyll -> HTML/JS/CSS -> GitHub Pages
- **Python tools** — Validate the Markdown, links, and assets of the documentation content
- **CI/CD flow** — GitHub Action workflows to run the Python tools before rebuilding and updating the Docker image, and before pushing doc changes to GitHub Pages.

GitHub repository

Since this repository is a **DevEx Starter Template**, the first step is to create your own independent copy by forking the repository.

1. Create a server-side fork (web)
    1. Navigate to the DevEx Starter Template GitHub page.
    2. Click the **Fork** button to create a copy of the repository under your personal account.
2. Clone your fork (command line)

    After the fork is complete, clone **your new copy** to your local machine. This is the sole remote (`origin`) for your project going forward.

    ```
    # Replace YOUR_USERNAME and YOUR_REPO with your new fork details
    git clone https://github.com/YOUR_USERNAME/YOUR_REPO.git
    cd YOUR_REPO
    ```

    Your project is now initialized. All changes, pushes, and branches are managed within your new repository.

Jekyll-based doc content

Jekyll is a static site generator that transforms your Markdown content into a complete website. In this project, Jekyll configuration files are located in the `/docs` directory.

This section covers the Jekyll-based documentation content, including Markdown, Jekyll, HTML/JS/CSS, and GitHub Pages.

Markdown

Use Markdown to create the content for your documentation. It is a lightweight markup language with plain text formatting syntax. In this project, Markdown files are located in the `/docs/topics` directory.

You should pick a style guide for your Markdown content to ensure consistency. This project follows the Google Developer Documentation Style Guide style guide for technical documentation.

Jekyll

Jekyll processes the Markdown files and applies templates to generate a static website. The Jekyll configuration file (`_config.yml`) and layout files are located in the `/docs` directory.

You need to install Jekyll and its dependencies to build the site locally. Follow the Jekyll installation guide for your operating system.

For full details on how the Jekyll system is set up in this project, see Jekyll system.

HTML/JS/CSS

Format your static website pages with JavaScript and CSS to enhance the user experience of your documentation site.

These files are in the `/docs/assets/css` and `/docs/assets/js` directories.

GitHub Pages

GitHub Pages hosts your Jekyll-generated static site directly from your GitHub repository. The site is automatically built and deployed whenever you push changes to the repository.

To enable GitHub Pages for your repository, go to the repository **Settings**, navigate to the **Pages** section, and select the source branch and folder (usually `main` branch and `/docs` folder).

Optionally, you can configure a custom domain for your GitHub Pages site in the same **Pages** section of the repository **Settings**.

Python tools

This project includes Python tools to validate the Markdown, links, and assets of the documentation content.

FastAPI and Swagger docs

The FastAPI application provides automatic API documentation using Swagger and ReDoc.

The FastAPI app is located in the `/src/api.py` file.

You can see an example of Swagger docs for the deployed instance of this project at this Lightsail container.

Typer CLI

The Typer command-line interface (CLI) allows you to run the validation tools from the command line.

The Typer CLI is located in the `/src/cli.py` file.

To run the CLI, navigate to the project root directory and use the following command:

```
python -m src.cli <command> [options]
```

CI/CD flow

When you push changes to the repository, GitHub Actions workflows are triggered to run the Python validation tools before rebuilding and updating the Docker image and before updating GitHub Pages with doc changes.

GitHub actions

GitHub Actions workflows are defined in the `.github/workflows` directory.

The **DevEx Starter Template** includes the following workflows:

- **Check Links** — This workflow runs `htmlproofer` to check for broken links in the generated site whenever changes are pushed to the `main` branch.
- **Generate PDF** — This workflow generates a PDF version of the documentation site using `wkhtmltopdf` whenever changes are pushed to the `main` branch.
- Others coming soon…

Docker

The Dockerfile and Docker Hub repository are used to build and store the Docker image for your project. The

Dockerfile is located in the root of the repository, and the Docker Hub repository should be created under your Docker Hub account.

Deploy image

You can deploy your image to a cloud service like AWS Lightsail Container Service, AWS ECS, or any other container orchestration platform.

Documentation

- Overview
- Jekyll system

Jekyll system

```
/docs
    ├── _config.yml           # Jekyll configuration file
    ├── _data                 # Data files for Jekyll
    │     └── tree.yml        # Site structure for sidebar navigation
    ├── _layouts              # Jekyll layout templates
    │     └── page.html       # Layout for documentation pages
    ├── _includes             # Jekyll includes (reusable components)
    ├── _site                 # Generated site (output)
    ├── assets                # CSS, JS, images
    │     ├── css
    │     │    └── main.css   # Main stylesheet
    │     ├── js
    │     │    └── main.js    # Main JavaScript file
    │     └── images          # Image assets
    ├── topics                # Markdown documentation content
    │     └── <your-docs>.md  # Your documentation files
    └── index.md              # Home page content
```

- `_config.yml`: This file contains configuration settings for the Jekyll site, and is also the place to define external resources such as plugins and themes.

  The current configuration defines the minimum settings: where the source files are located and the destination for the generated site.

- `_data/tree.yml`: This file defines the structure of the documentation site for generating the sidebar navigation. It uses a simple YAML format to list the titles and URLs of the pages in the documentation.

- `_layouts`: This directory holds layout templates that define the structure of your documentation pages. The `page.html` layout is the default for rendering individual documentation pages.

- `_includes`: This directory contains reusable components that can be included in multiple pages, such as headers, footers, navigation elements, and boilerplate text. From your content, you include these components with Liquid tags, such as `{% include header.html %}`.

- `_site`: This is the output directory where Jekyll generates the static site.

- `assets`: This directory contains static assets like CSS, JavaScript, and images used in the documentation.

- `topics`: This directory contains the Markdown files for your documentation content.

- `index.md`: This file serves as the home page for your documentation site.

A note on underscores: In Jekyll, directories and files that start with an underscore (_) are not generated in the static site, but Jekyll uses them for templating and configuration. You can also create your own directories and files that start with an underscore for work in progress or topics you want to remove from the released documentation but keep for future work, or simply store internal notes in them.

For more information on Jekyll's structure and conventions, refer to the Jekyll documentation.

Generating and previewing the site

To run a local Jekyll server for previewing your documentation site, use the following command in your terminal:

```
jekyll server
```

This command starts a local server and watches for changes in your files, allowing you to see updates in real-time as you edit your documentation. By default, the server is accessible at `http://localhost:4000`.

You can also build the site without starting a server using:

```
jekyll build
```

This command generates the static site in the `_site` directory without serving it. You can then deploy the contents of the `_site` directory to your web server or hosting service, or as in this DevEx Starter Template examples, doing a `git push` automatically starts a GitHub Action that updates the GitHub Pages site with the updated content.

---

index.html

Documentation

- Overview
- Jekyll system

Welcome to the DevEx Starter Template docs

This project has three main sections:

- **Jekyll-based documentation content** — Markdown -> Jekyll -> HTML/JS/CSS -> GitHub Pages
- **Python tools** — Validate the Markdown, links, and assets of the documentation content
- **CI/CD flow** — GitHub Action workflows to run the Python tools before rebuilding and updating the Docker image, and before pushing doc changes to GitHub Pages.

GitHub repository

Since this repository is a **DevEx Starter Template**, the first step is to create your own independent copy by forking the repository.

1. Create a server-side fork (web)
    1. Navigate to the DevEx Starter Template GitHub page.
    2. Click the **Fork** button to create a copy of the repository under your personal account.
2. Clone your fork (command line)

After the fork is complete, clone **your new copy** to your local machine. This is the sole remote (`origin`) for your project going forward.

```
# Replace YOUR_USERNAME and YOUR_REPO with your new fork details
git clone https://github.com/YOUR_USERNAME/YOUR_REPO.git
cd YOUR_REPO
```

Your project is now initialized. All changes, pushes, and branches are managed within your new repository.

Jekyll-based doc content

Jekyll is a static site generator that transforms your Markdown content into a complete website. In this project, Jekyll configuration files are located in the `/docs` directory.

This section covers the Jekyll-based documentation content, including Markdown, Jekyll, HTML/JS/CSS, and GitHub Pages.

Markdown

Use Markdown to create the content for your documentation. It is a lightweight markup language with plain text formatting syntax. In this project, Markdown files are located in the `/docs/topics` directory.

You should pick a style guide for your Markdown content to ensure consistency. This project follows the Google Developer Documentation Style Guide style guide for technical documentation.

Jekyll

Jekyll processes the Markdown files and applies templates to generate a static website. The Jekyll configuration file (`_config.yml`) and layout files are located in the `/docs` directory.

You need to install Jekyll and its dependencies to build the site locally. Follow the Jekyll installation guide for your operating system.

For full details on how the Jekyll system is set up in this project, see Jekyll system.

HTML/JS/CSS

Format your static website pages with JavaScript and CSS to enhance the user experience of your documentation site.

These files are in the `/docs/assets/css` and `/docs/assets/js` directories.

GitHub Pages

GitHub Pages hosts your Jekyll-generated static site directly from your GitHub repository. The site is automatically built and deployed whenever you push changes to the repository.

To enable GitHub Pages for your repository, go to the repository **Settings**, navigate to the **Pages** section, and select the source branch and folder (usually `main` branch and `/docs` folder).

Optionally, you can configure a custom domain for your GitHub Pages site in the same **Pages** section of the repository **Settings**.

Python tools

This project includes Python tools to validate the Markdown, links, and assets of the documentation content.

FastAPI and Swagger docs

The FastAPI application provides automatic API documentation using Swagger and ReDoc.

The FastAPI app is located in the `/src/api.py` file.

You can see an example of Swagger docs for the deployed instance of this project at this Lightsail container.

Typer CLI

The Typer command-line interface (CLI) allows you to run the validation tools from the command line.

The Typer CLI is located in the `/src/cli.py` file.

To run the CLI, navigate to the project root directory and use the following command:

```
python -m src.cli <command> [options]
```

CI/CD flow

When you push changes to the repository, GitHub Actions workflows are triggered to run the Python validation tools before rebuilding and updating the Docker image and before updating GitHub Pages with doc changes.

GitHub actions

GitHub Actions workflows are defined in the `.github/workflows` directory.

The **DevEx Starter Template** includes the following workflows:

- **Check Links** — This workflow runs `htmlproofer` to check for broken links in the generated site whenever changes are pushed to the `main` branch.
- **Generate PDF** — This workflow generates a PDF version of the documentation site using `wkhtmltopdf` whenever changes are pushed to the `main` branch.
- Others coming soon...

Docker

The Dockerfile and Docker Hub repository are used to build and store the Docker image for your project. The Dockerfile is located in the root of the repository, and the Docker Hub repository should be created under your Docker Hub account.

Deploy image

You can deploy your image to a cloud service like AWS Lightsail Container Service, AWS ECS, or any other container orchestration platform.

- Overview
- Jekyll system

Jekyll system

```
/docs
    ├── _config.yml          # Jekyll configuration file
    ├── _data                # Data files for Jekyll
    │     └── tree.yml       # Site structure for sidebar navigation
```

```
├── _layouts             # Jekyll layout templates
│   └── page.html        # Layout for documentation pages
├── _includes            # Jekyll includes (reusable components)
├── _site                # Generated site (output)
├── assets               # CSS, JS, images
│   ├── css
│   │   └── main.css     # Main stylesheet
│   ├── js
│   │   └── main.js      # Main JavaScript file
│   └── images           # Image assets
├── topics               # Markdown documentation content
│   └── <your-docs>.md   # Your documentation files
└── index.md             # Home page content
```

- `_config.yml`: This file contains configuration settings for the Jekyll site, and is also the place to define external resources such as plugins and themes.

  The current configuration defines the minimum settings: where the source files are located and the destination for the generated site.

- `_data/tree.yml`: This file defines the structure of the documentation site for generating the sidebar navigation. It uses a simple YAML format to list the titles and URLs of the pages in the documentation.

- `_layouts`: This directory holds layout templates that define the structure of your documentation pages. The `page.html` layout is the default for rendering individual documentation pages.

- `_includes`: This directory contains reusable components that can be included in multiple pages, such as headers, footers, navigation elements, and boilerplate text. From your content, you include these components with Liquid tags, such as `{% include header.html %}`.

- `_site`: This is the output directory where Jekyll generates the static site.

- `assets`: This directory contains static assets like CSS, JavaScript, and images used in the documentation.

- `topics`: This directory contains the Markdown files for your documentation content.

- `index.md`: This file serves as the home page for your documentation site.

A note on underscores: In Jekyll, directories and files that start with an underscore (_) are not generated in the static site, but Jekyll uses them for templating and configuration. You can also create your own directories and files that start with an underscore for work in progress or topics you want to remove from the released documentation but keep for future work, or simply store internal notes in them.

For more information on Jekyll's structure and conventions, refer to the Jekyll documentation.

Generating and previewing the site

To run a local Jekyll server for previewing your documentation site, use the following command in your terminal:

```
jekyll server
```

This command starts a local server and watches for changes in your files, allowing you to see updates in real-time as you edit your documentation. By default, the server is accessible at `http://localhost:4000`.

You can also build the site without starting a server using:

```
jekyll build
```

This command generates the static site in the `_site` directory without serving it. You can then deploy the contents of the `_site` directory to your web server or hosting service, or as in this DevEx Starter Template examples, doing a `git push` automatically starts a GitHub Action that updates the GitHub Pages site with the updated content.