Modifying content on this site

You have two options for editing content: directly in your browser using GitHub, or using a Git-based workflow from your local work environment.

Notes on content additions

Sections: Generally, content additions should fit somewhere within the existing top-level and second-level sections. Try not to introduce a new top-level or second-level section.

Screenshots: If you are contributing a change that contains screenshots from Cleura Cloud Management Panel, they should use a resolution of 1920×1080 pixels (1080p). If your screen uses a larger resolution, use Firefox Responsive Design Mode or Chrome/Chromium Device Mode to configure your browser with 1080p. Screenshots should be added to a directory named `assets`, located in the same directory as the Markdown file you are adding or editing.

CLI screen dumps: If you are contributing a change that contains a screen dump from the `openstack` command-line client, please limit its width to 100 characters. You can do this by setting the following environment variable in your terminal, before you start work on your change.

```
export CLIFF_MAX_TERM_WIDTH=100
```

Modifying content from your browser

Every page on this site has an Edit button ✏️. If you click it, it'll take you straight to the corresponding source page in GitHub. Then, you can follow GitHub's documentation on how to propose changes to another user's repository.

Modifying content using Git

The Git repository for this site lives at https://github.com/citynetwork/docs. You can fork that repository, make the proposed changes in your fork, and then send us a standard GitHub pull request.

For this purpose, use `git` in combination with either GitHub's web interface, or the `gh` command-line interface (CLI).

First, create a fork of the documentation repository:

`git` client and web browser **`gh` client**

Open https://github.com/citynetwork/docs and click the *Fork* button. When you create your new fork, it's fine to leave the *Copy the `main` branch only* option enabled.

Then, proceed to create a new local checkout of your fork:

```
git clone git@github.com:<yourusername>/<your-repo-fork> cleura-docs
cd cleura-docs
```

and make your modifications:

```
git checkout -b <your-topic-branch-name>
# edit your files
git add <files-to-add>
git commit
cd cleura-docs
```

Please see our notes on commit messages.

Finally, create a pull request (PR) from your changes:

`git` client and web browser **`gh` client**

Run the following `git` command (assuming `origin` is the remote that points to your fork):

```
git push origin <your-topic-branch-name>
```

Then, open your browser to the URL suggested by

the `git push` command, and proceed to create a pull request.

```
gh pr create --fill
```

Monitoring changes as you edit

ges as you work on them, you can use tox. Having created a topic branch with your modifications, run:

```
cd cleura-docs
git checkout <your-topic-branch-name>
tox -e serve
```

A local copy of the documentation will then run on your local machine and be accessible from http://localhost:8000 in your browser.

When you are planning to make several changes in rapid succession, you may want to speed up rendering the site after each change. You may do so by disabling a plugin that checks all links (including external links) for accessibility:

```
cd cleura-docs
export DOCS_ENABLE_HTMLPROOFER=false
tox -e serve
```

Commit messages

When you submit a change, you will need to provide a commit message, which is very nearly as important as the change itself. Excellent guides on what constitutes a good commit message are available from Tim Pope and Colleen Murphy.

In addition, we have adopted the Conventional Commits style for commit message subjects. Please make sure that your commit message starts with one of the following prefixes:

- `feat:` denotes a content addition, such as adding documentation for some Cleura Cloud functionality that was not included in the documentation before.
- `fix:` denotes a content correction, such as fixing a documentation bug.
- `build:` denotes a change to the build process, such as an improvement to a CI check.
- `chore:` denotes a minor change that is neither a feature, nor a fix, nor a build improvement, such as when you edit the `.mailmap` file.

- `docs:` denotes a change to the documention *for* this site, such as an update to the `README.md` file.

Last update: 2022-11-08

Created: 2022-10-25

Authors: Florian Haas