Table of Contents

Installing and Deploying

Recommended Approach

The recommended approach for using **al-folio** is to first create your own site using the template with as few changes as possible, and only when it is up and running customize it however you like. This way it is easier to pinpoint what causes a potential issue in case of a bug. The minimum steps required to create your own site are (video tutorial here):

1. Create a new repository using this template. For this, click on Use this template -> Create a new repository above the file list. If you plan to upload your site to `<your-github-username>.github.io`, note that the name of your repository ⚠️ **MUST BE** ⚠️ `<your-github-username>.github.io` or `<your-github-orgname>.github.io`, as stated in the GitHub pages docs.
2. In this new repository, go to `Settings -> Actions -> General -> Workflow permissions` and give `Read and write permissions` to GitHub Actions.
3. Open file `_config.yml`, set `url` to `https://<your-github-username>.github.io` and leave `baseurl` **empty** (do NOT delete it), as `baseurl:`.
4. Wait until the GitHub action with subtitle `Deploy site` finishes (check your repository **Actions** tab), which takes ~4 min. Now, in addition to the `main` branch, your repository has a newly built `gh-pages` branch.
5. Finally, in the repository page go to `Settings -> Pages -> Build and deployment`, make sure that `Source` is set to `Deploy from a branch` and set the branch to `gh-pages` (NOT to main).
6. Wait until the GitHub action `pages-build-deployment` finishes (check your repository **Actions** tab), which takes ~45s, then simply navigate to `https://<your-github-username>.github.io` in your browser. At this point you should see a copy of the theme's demo website. After everything is set up,

you can download the repository to your machine and start customizing it. To do so, run the following commands:

```
$ git clone git@github.com:<your-username>/<your-repo-name>.git
```

Starting version v0.3.5, **al-folio** will automatically re-deploy your webpage each time you push new changes to your repository! ✨

Local setup on Windows

If you are using Windows, it is **highly recommended** to use Windows Subsystem for Linux (WSL), which is a compatibility layer for running Linux on top of Windows. You can follow these instructions to install WSL and Ubuntu on your machine. You only need to go up to the step 4 of the tutorial (you don't have to enable the optional `systemd` nor the graphical applications), and then you can follow the instructions below to install docker. You can install docker natively on Windows as well, but it has been having some issues as can be seen in #1540, #2007.

Local setup using Docker (Recommended)

Using Docker to install Jekyll and Ruby dependencies is the easiest way.

You need to take the following steps to get `al-folio` up and running on your local machine:

- First, install docker and docker-compose.
- Finally, run the following command that will pull the latest pre-built image from DockerHub and will run your website.

```
$ docker compose pull
$ docker compose up
```

Note that when you run it for the first time, it will download a docker image of size 400MB or so. To see the template running, open your browser and go to `http://localhost:8080`. You should see a copy of the theme's demo website.

Now, feel free to customize the theme however you like (don't forget to change the name!). Also, your changes should be automatically rendered in real-time (or maybe after a few seconds).

> Beta: You can also use the slimmed docker image with a size below 100MBs and exact same functionality. Just use `docker compose -f docker-compose-slim.yml up`

Build your own docker image

> Note: this approach is only necessary if you would like to build an older or very custom version of al-folio.

Build and run a new docker image using:

```
$ docker compose up --build
```

> If you want to update jekyll, install new ruby packages, etc., all you have to do is build the image again using `--force-recreate` argument at the end of the previous command! It will download Ruby and Jekyll and install all Ruby packages again from scratch.

If you want to use a specific docker version, you can do so by changing `latest` tag to `your_version` in `docker-compose.yaml`. For example, you might have created your website on `v0.10.0` and you want to stick with that.

Local Setup with Development Containers

`al-folio` supports Development Containers. For example, when you open the repository with Visual Studio Code (VSCode), it prompts you to install the necessary extension and automatically install everything necessary.

Local Setup (Legacy, no longer supported)

For a hands-on walkthrough of running al-folio locally without using Docker, check out this cool blog post by one of the community members!

Assuming you have Ruby and Bundler installed on your system (*hint: for ease of managing ruby gems, consider using rbenv*), and also Python and pip (*hint: for ease of managing python packages, consider using a virtual environment, like venv or conda*).

```
$ bundle install
# assuming pip is your Python package manager
$ pip install jupyter
$ bundle exec jekyll serve
```

To see the template running, open your browser and go to `http://localhost:4000`. You should see a copy of the theme's demo website. Now, feel free to customize the theme however you like. After you are done, remember to **commit** your final changes.

Deployment

Deploying your website to GitHub Pages is the most popular option. Starting version v0.3.5, **al-folio** will automatically re-deploy your webpage each time you push new changes to your repository **main branch**! ✨

For personal and organization webpages

1. The name of your repository **MUST BE** `<your-github-username>.github.io` or `<your-github-orgname>.github.io`.
2. In `_config.yml`, set `url` to `https://<your-github-username>.github.io` and leave `baseurl` empty.
3. Set up automatic deployment of your webpage (see instructions below).

4. Make changes to your main branch, commit, and push!
5. After deployment, the webpage will become available at `<your-github-username>.github.io`.

For project pages

1. In `_config.yml`, set `url` to `https://<your-github-username>.github.io` and `baseurl` to `/<your-repository-name>/`.
2. Set up automatic deployment of your webpage (see instructions below).
3. Make changes to your main branch, commit, and push!
4. After deployment, the webpage will become available at `<your-github-username>.github.io/<your-repository-name>/`.

Enabling automatic deployment

1. Click on **Actions** tab and **Enable GitHub Actions**; do not worry about creating any workflows as everything has already been set for you.
2. Go to `Settings -> Actions -> General -> Workflow permissions`, and give `Read and write permissions` to GitHub Actions
3. Make any other changes to your webpage, commit, and push to your main branch. This will automatically trigger the **Deploy** action.
4. Wait for a few minutes and let the action complete. You can see the progress in the **Actions** tab. If completed successfully, in addition to the `main` branch, your repository should now have a newly built `gh-pages` branch. **Do NOT touch this branch!**
5. Finally, in the **Settings** of your repository, in the Pages section, set the branch to `gh-pages` (**NOT** to `main`). For more details, see Configuring a publishing source for your GitHub Pages site.

If you keep your site on another branch, open `.github/workflows/deploy.yml` **on the branch you keep your website on** and change `on->push->branches` and `on->pull\_request->branches` to the branch you keep your website on. This will trigger the action on pulls/pushes on that branch. The action will then deploy the website on the branch it was triggered from.

Manual deployment to GitHub Pages

If you need to manually re-deploy your website to GitHub pages, go to Actions, click "Deploy" in the left sidebar, then "Run workflow."

Deployment to another hosting server (non GitHub Pages)

If you decide to not use GitHub Pages and host your page elsewhere, simply run:

```
$ bundle exec jekyll build
```

which will (re-)generate the static webpage in the `_site/` folder. Then simply copy the contents of the `_site/` directory to your hosting server.

If you also want to remove unused css classes from your file, run:

```
$ purgecss -c purgecss.config.js
```

which will replace the css files in the `_site/assets/css/` folder with the purged css files.

Note: Make sure to correctly set the `url` and `baseurl` fields in `_config.yml` before building the webpage. If you are deploying your webpage to `your-domain.com/your-project/`, you must set `url: your-domain.com` and `baseurl: /your-project/`. If you are deploying directly to `your-domain.com`, leave `baseurl` blank, **do not delete it**.

Deployment to a separate repository (advanced users only)

Note: Do not try using this method unless you know what you are doing (make sure you are familiar with publishing sources). This approach allows to have the website's source code in one repository and the deployment version in a different repository.

Let's assume that your website's publishing source is a `publishing-source` subdirectory of a git-versioned repository cloned under `$HOME/repo/`. For a user site this could well be something like `$HOME/<user>.github.io`.

Firstly, from the deployment repo dir, checkout the git branch hosting your publishing source.

Then from the website sources dir (commonly your al-folio fork's clone):

```
$ bundle exec jekyll build --destination $HOME/repo/publishing-source
```

This will instruct jekyll to deploy the website under `$HOME/repo/publishing-source`.

Note: Jekyll will clean `$HOME/repo/publishing-source` before building!

The quote below is taken directly from the jekyll configuration docs:

> Destination folders are cleaned on site builds
>
> The contents of `<destination>` are automatically cleaned, by default, when the site is built. Files or folders that are not created by your site will be removed. Some files could be retained by specifying them within the `<keep_files>` configuration directive.
>
> Do not use an important location for `<destination>`; instead, use it as a staging area and copy files from there to your web server.

If `$HOME/repo/publishing-source` contains files that you want jekyll to leave untouched, specify them under `keep_files` in `_config.yml`. In its default configuration, al-folio will copy the top-level `README.md` to the publishing source. If you want to change this behavior, add `README.md` under `exclude` in `_config.yml`.

Note: Do *not* run `jekyll clean` on your publishing source repo as this will result in the entire directory getting deleted, irrespective of the content of `keep_files` in `_config.yml`.

Upgrading from a previous version

If you installed **al-folio** as described above, you can manually update your code by following the steps below:

```
# Assuming the current directory is <your-repo-name>
$ git remote add upstream https://github.com/alshedivat/al-folio.git
$ git fetch upstream
$ git rebase v0.11.0
```

If you have extensively customized a previous version, it might be trickier to upgrade. You can still follow the steps above, but `git rebase` may result in merge conflicts that must be resolved. See git rebase manual and how to resolve conflicts for more information. If rebasing is too complicated, we recommend re-installing the new version of the theme from scratch and port over your content and changes from the previous version manually. You can use tools like meld or winmerge to help in this process.