

Publishing a note

NSDF-Data student guide 4 — for every computer

Version 11 · 20 October 2026

Tip

This is guide 4 of 5. It is about getting a finished note onto the site. Guide 3 covers writing the note; guide 5 covers doing this with an AI assistant.

What happens when you publish

Your note goes through four stages. You do the first two, the automatic check does the third, and your lead does the fourth:

1. **You** write the note as a `.md` file in the `notes/src/` folder, on your own branch.
2. **You** open a **pull request**: a request to add your work to the project.
3. **The check** runs at once. It lists anything wrong with the note.
4. **Your lead** reviews the note, gives it a note number (the next free one, for example 8), and merges it. After the merge the site builds your page. This takes a few minutes.

Nothing reaches the public site until the merge.

Before you start

- **You have write access.** Your lead adds you to the project on GitHub. You need this to push a branch.
- **Your note is written and checked** against guide 3, as a Markdown file (the VS Code route) or a Word or Google Docs file, and the notebook it uses is committed to your branch.
- **You know your branch name.** It is `student-yourname`, from the setup guide.

Step 1. Get your note into `notes/src/`

Your note has to end up as a Markdown file in the `notes/src/` folder, with its pictures in `notes/src/img/`. Follow the part that matches how you wrote it.

If you wrote Markdown in VS Code

The note is already in place. Check that:

- the note is `notes/src/your-note-name.md`, started from `_template.md`, with the header kept;
- every picture is a file directly in `notes/src/img/` (not in a subfolder), named with your note's name at the start, and linked as `img/your-note-name-fig1.png`;

- the template's line that points to `img/example.png` is deleted or replaced by one of your own pictures. The check reports a missing picture otherwise.

If you wrote in Word or Google Docs

A small program in the project, `tools/word_to_note.py`, turns the document into a note and saves its pictures in the right place under the right names.

a. Export the document.

- In **Word**: **File**, then **Save As**, and choose **Word Document (.docx)**.
- In **Google Docs**: **File**, then **Download**, then **Microsoft Word (.docx)**.

Save it in your Research folder, next to NSDF-Data, as `my-note.docx`.

b. Install pandoc once. In the terminal, with your environment active:

```
conda activate darkmatter_cli_env
conda install -c conda-forge pandoc -y
```

On macOS you can also use `brew install pandoc`. On Windows, the [pandoc website](#) has an installer.

c. Convert. In the terminal, go to the NSDF-Data folder and run (change `my-note.docx` and choose a note name made of lowercase letters and hyphens):

```
python tools/word_to_note.py ../my-note.docx your-note-name
```

It writes `notes/src/your-note-name.md`. It saves each picture in `notes/src/img/` as `your-note-name-fig1.png`, then `-fig2.png`, and so on, in the order the pictures appear. It fixes the links and tells you what it did.

Checkpoint

The program printed `Wrote notes/src/your-note-name.md` and the number of pictures you expect. Open the note in VS Code or any text editor. Every section title you styled as a heading should start with `#` or `##`, and every picture should be a line starting `!` [with a caption line under it. If the program listed leftover formatting (`<u>`, `<sup>`, `<sub>`, `<span>`), delete those tags and keep the words, or the check will reject the note. If you see no `#` lines at all, your titles were not Heading-styled: fix the document and run the command again.

d. Fill in the header at the top of the new note: title, author, date, keywords and the one-sentence description (guide 3, *The header*). Leave `id: "TBD"`.

Tip

Test the conversion before you rely on it. Convert a short two-section document with one picture first. If the headings and the picture both come through, convert the real note.

Your lead will replace `id: "TBD"` with a number.

Step 2. Check your note, then commit to your branch

In the terminal, from the NSDF-Data folder, with your environment active, run the checks on your note (change the file name to yours):

```
python tools/check_note.py notes/src/your-note-name.md
```

This runs, in one go, the same checks that GitHub runs on your pull request, and shows only what concerns your note:

- **Structure:** the header, the status, the pictures. A problem fails the check.
- **Pinned links:** every link to your code is pinned to a commit that exists. A problem fails the check. Push your notebook commit before you open the pull request: GitHub only sees pushed commits.
- **Numbers:** the numbers in your text that are not in your notebook's saved output. This is advice only (guide 3 explains what to do with each line), and it does not fail the check.

Read the last line. *ready for review* means the structure and the links are fine. If it says to fix the problems, fix them (Step 4 has a table of the common messages) and run it again.

Tip

Behind the project? If you have not updated your branch for a while, do the *Updating your setup* guide first, so you have the newest checks. A few commits behind is fine.

Then commit and push, from the NSDF-Data folder:

```
git checkout student-yourname
git add notes/src/your-note-name.md notes/src/img
git status
git commit -m "Add note: your title"
git push -u origin student-yourname
```

Checkpoint

Look at the list that `git status` prints before the commit. It should contain only your note and its pictures. If it shows `.bin` files, anything from the `idx` folder, or changes to other people's notes, stop and ask your lead.

Step 3. Open the pull request

Tip

What is a pull request? Nothing scary. Your branch is your own copy of the project. Nothing you do on it changes anyone else's work or the public site. A *pull request* is a button on github.com that means "I'm done, please have a look." Your lead reads it, may ask for changes, and only then adds your work to the project. Until then you can push as often as you like, and you cannot break anything. If something looks wrong, stop, type `git status`, and send your lead what it prints. Guide 8, *Git in five minutes*, explains the words and the few commands you need.

On github.com, open the repository. GitHub shows a yellow banner saying your branch had recent pushes. Click **Compare & pull request**. Check that:

- the **base** is `develop` and the **compare** branch is `student-yourname`;
- the title says what the note is about;
- the description says, in one sentence, what question the note answers;
- if your note replaces an earlier note, the description says so (for example *Supersedes Note*

3).

Click **Create pull request**.

Careful

The description box is already filled in, with a checklist written for the project's developers. Do not be put off by it, and do not delete it. For a note, the *Appendix* at the end of this guide gives you the exact text to paste into the box and says where to change it. The same form appears if you start the pull request from VS Code.

Careful

If your note replaces an earlier note, that note's status must change and it must link to yours, and this has to travel in the same pull request. Guide 3 (*If your note replaces an earlier note*) says what goes in your note. For a hand-written note, your lead adds the change as a commit on your branch, so you do not edit docs/. For another student note, include its changed file in your pull request.

Step 4. Read the check

The check starts automatically and takes less than a minute. It is the structure check of Step 2, and for the notes your pull request changes it also lists, as warnings, the numbers that are not in the notebook output. When it finishes, click **Details** on the check, and read its messages. Each message names the file and the problem. Here are the common ones and what to do:

Message says	What to do
header field 'X' is missing or empty	Fill in that line of the header.
status must be one of ...	Use exactly <i>In progress</i> , <i>Complete</i> or <i>Superseded (reference/link)</i> .
id must be a note number such as 8 ... or TBD	Leave id: "TBD".
raw HTML tag <u> (or <sup>, <sub>,)	Remove the formatting and keep the words.
notebook link pinned to 'master'	Replace master with the commit hash you committed the notebook at.
placeholder COMMIT is still in the note	Put the real commit hash in place of COMMIT.
picture not found	Check the file name and that the picture is in notes/src/img/.
picture '...' must be directly in notes/src/img/	Move the picture out of the subfolder into notes/src/img/ and link it as img/name.png. A note made with word_to_note.py already does this.

To fix a problem: edit the file on your branch and commit again. The pull request updates itself, and

the check runs again.

Tip

If the check passes, the structure is right. It cannot tell whether your numbers are right or your reading is fair. Check those yourself, and ask your lead if you are unsure.

Step 5. Wait for review

Your lead reads the note, may comment on the pull request, and may ask you for changes. Answer each comment in the pull request. When the note is ready, your lead gives it a number and merges it.

After the merge

- The site builds your page automatically, usually within two minutes. Check the front page: your note appears in the **Analysis Notes** table, in order of its note number.
- The page is at `notes/student-<name>.html` on the site.

Changing a note later

You can change a published note either way you wrote it. Start from the latest version of your branch. Your lead gave your note its number on the pull request, and your own copy of the file may still say TBD:

```
git switch student-yourname
git pull origin develop
```

Open `notes/src/your-note-name.md` and look at the header. It should say your note's number, for example `id: "8"`. **Keep that number.** If it says TBD, stop and ask your lead: the check does not mind TBD, but the site cannot publish a note with it, and nothing is published until it is fixed.

Then choose **one** of the two ways and keep to it, because the second one replaces the text of the note:

Way 1. Edit the Markdown. Open `notes/src/your-note-name.md` in VS Code (or any text editor), change the text, and replace a picture by saving the new PNG under the same name in `notes/src/img/`.

Way 2. Edit the Word or Google Docs file. Change the document, export it again as a `.docx` (Step 1a), and run the same command with the **same note name**:

```
python tools/word_to_note.py ../my-note.docx your-note-name
```

The program replaces the text and the pictures, and **keeps your header as it is**, so the number stays. Pictures are numbered in the order they appear, so the file names can shift if you add one in the middle; the links are rewritten to match, and pictures that are no longer used are removed. Do not edit the Markdown by hand in this way: the next run would overwrite your changes.

After either way, update the **date** line, and the **Status** line if the conclusion has changed, in the header. Look at what changed before you commit:

```
git status
git diff notes/src/your-note-name.md
```

Then commit, push, and open a new pull request (or add to the open one). Let the check run again. If a result was wrong, say so in the note, with the date, as guide 3 explains. Do not delete a published

note to hide a mistake.

Appendix. The pull request form, step by step

When you start a pull request, on github.com or from VS Code, the form has four parts. This is what to do with each one, for a student note.

1. The two branches. At the top, the **base** must be `develop` and the branch you are comparing (the *compare* or *merge* branch) must be `student-yourname`.

- On **github.com** these are two drop-down buttons at the top of the form, labelled *base* and *compare*. Change *base* to `develop` if it says something else.
- In the **VS Code** panel they are the two lines **BASE** and **MERGE**. The branch name next to BASE is a button even though it looks like plain text: click it and choose `develop`. If both lines show the same branch, the panel says there are no commits to compare and the **Create** button stays grey. Changing BASE to `develop` fixes it.

2. The title. One line that says what the note is about, for example the title of your note.

3. The description. GitHub fills this box with a checklist. Click inside the box, select everything in it (Ctrl+A, or Cmd+A on a Mac) and replace it with the text below. Copy it from here, then change the two parts that are in angle brackets, `<like this>`, and nothing else.

```
> **Target branch:** open feature PRs against **`develop`**.  
> Only `release/*` and `hotfix/*` branches target `master`. See `RELEASING.md`.
```

What and why

<One or two sentences: the question your note answers and what you did.
If it replaces an earlier note, say so, for example "Supersedes Note 3".>

What changed

```
- [ ] Library code (`python/`) n/a  
- [x] Notebooks (`data_analysis/`)  
- [ ] Notes site or hand-written notes (`docs/`) n/a  
- [ ] Student guides (`docs/students/src/`) n/a  
- [x] Student note (`notes/src/`)  
- [ ] CI or build files (`.github/`, `tools/`) n/a
```

Checklist

Tick what applies, and say "n/a" for the rest.

```
- [ ] Tests pass (`python -m pytest -q` in `python/`) n/a  
- [ ] New or changed library functions have tests and a `@status` tag n/a  
- [ ] A new or renamed note has its row in `docs/index.html` n/a  
- [ ] Guides: edited the sources, rebuilt, committed sources and PDFs n/a  
- [ ] Guides: bumped the version and date n/a  
- [ ] Changed package pins? Updated `requirements.txt` and the guide n/a  
- [x] Student note: left `id: "TBD"` and ran `check_note.py` on it  
- [ ] Added a line under `[Unreleased]` in `CHANGELOG.md` n/a
```

- [x] No raw data, credentials or tokens committed
- [x] If an AI assistant helped: a `Co-Authored-By:` trailer is in the commits

How I tested it

<Say what you ran and what you saw. For a note: that you ran
`python tools/check\_note.py notes/src/your-note-name.md`, which of the three
checks said OK, and anything it listed that you checked by hand.>

4. Ticks and “n/a”. The square brackets are the box itself: [] is empty and [x] is ticked. To tick a box, put an x between the brackets. To say that a line does not apply, leave the box empty and type n/a at the end of the line, outside the brackets. Do not type inside the brackets: the line would stop being a box. Nothing reads the boxes by machine, so a missed one costs nothing; they tell your lead what you checked.

Checkpoint

Before you click **Create**: the base is develop, the title says what the note is about, and the two parts in angle brackets have been replaced by your own sentences (no <like this> text is left). Then click **Create pull request** (or **Create**) and go on to Step 4.

Session Info

Guide version 11, 20 October 2026. Written for every computer. Please tell the project lead where a step was unclear.

Where to get help

Anthony Villano, project lead: anthony.villano@ucdenver.edu

When you write, include the pull request number, the message the check gave, and the step you are stuck on.

Key links

- Pull requests: <https://github.com/villano-lab/NSDF-Data/pulls>
- Notes site: <https://villano-lab.github.io/NSDF-Data/>
- Repository: <https://github.com/villano-lab/NSDF-Data>