Lessons with version control

Objectives

- Understand why version control is useful even for teaching material
- Understand how version control managed lessons can be modified.
- Understand how the CodeRefinery lesson template is used to create new lessons

Instructor note

- Discussion: 25 min
- Exercises or demos: 20 min

Why version control?

- If you are in CodeRefinery TTT, you probably know what version control is and why it is important.
- The benefits of version control also extend to lessons:
  - Change history
  - Others can submit contributions
  - Others can make derived versions and sync up later
  - Same workflow as everything else
  - Write it like documentation: probably more reading after than watching it as a presentation.
- Disadvantages
  - “What you see is what you get” editing is hard
  - Requires knowing version control

Accepting the smallest contribution

Question: if someone wants to make a tiny fix to your material, can they?

Tour of lesson templates options

There are different ways to make lessons with git. Some dedicated to teaching:

- CodeRefinery
  - Example: This lesson itself
  - Based on the Sphinx documentation generator

    - sphinx-lesson is *very* minimal extra functionality
- Carpentries
    - Example: https://carpentries.github.io/lesson-example/
    - Based on R and Rmarkdown

Our philosophy is that anything works: it doesn't have to be just designed for lessons

- Jupyter Book
    - Example: https://jupyterbook.org/
    - Note: is based on sphinx, many extensions here are used in CR lessons
- Various ways to make slides out of Markdown
- Cicero: GitHub-hosted Markdown to slides easily
    - Demo: Asking for Help with Supercomputers The source
- Whatever your existing documentation is.

We like the CodeRefinery format, but think that you should use whatever fits your needs the most.

Sphinx

- We build all our lesson material with Sphinx
- Generate HTML/PDF/LaTeX from RST and Markdown.
- Many Python projects use Sphinx for documentation but **Sphinx is not limited to Python**.
- Read the docs hosts public Sphinx documentation for free!
- Also hostable anywhere else, like Github pages, like our lesson material
- For code a selling point for Sphinx is that also API documentation is possible.

Sphinx is a doc generator, not HTML generator. It can:

- Markdown, Jupyter, and ReST (and more...) inputs. Executable inputs.
    - jupyter-book is Sphinx, so anything it can do we can do. This was one of the inspirations for using Sphinx
- Good support for inline code. Much more than static code display, if you want to look at extensions.
- Generate different output formats (html, single-page html, pdf, epub, etc.)
- Strong cross-referencing within and between projects

CodeRefinery lesson template

It is "just a normal Sphinx project" - with extensions:

- Sphinx lesson extension
    - adds is various directives (boxes) tuned to lesson purposes
    - provides a sample organization and template repo you can use so that lessons look

consistent
- Sphinx gives us other nice features for free
  - Tabs: they are very important for us and allow us to customize in-place instead of copying everything and fragmenting lessons
  - Emphasize lines: they make it easier to spot what has changed in longer code snippets
  - Various input formats
    - Markdown (via the MyST-parser), ReStructured text, Jupyter Notebooks.
  - Many other features designed for presenting and interacting with code
- It's fine if you use some other static site generator or git-based lesson method.

👀 Instructors go through the building and contributing process

Depending on the course, instructors will demo what is roughly exercise 4 below. Or a course might go straight to exercises.

- Instructors decide what change they would want to make
- Instructors clone the repository
- **Instructors make the change**
- **Instructors set up the build environment**
- **Instructors build and preview**
- Instructors command and send upstream

Exercises

Some exercises have prerequisites (Git or Github accounts). Most instances of this course will have you do **1 and 2** below.

✍️ Lesson-VCS-1: Present and discuss your own lesson formats

We don't want to push everyone towards one format, but as long as you use Git, it's easy to share and reuse.

- Discuss what formats you do use
- Within your team, show examples of the lessons formats you use now. Discuss what is good and to-be-improved about them.
- Look at how they source is managed and how easy it might be to edit.

✍️ Lesson-VCS-2: Tour a CodeRefinery or Carpentries lesson on Github

- Look at either a CodeRefinery or Carpentries lesson
  - CodeRefinery Git-Intro: Lesson, Github repo
  - Carpentries Linux shell: Lesson, Github repo
- Can you find
  - Where is the content of the lessons?

- What recent change propsals (pull requests) have been made?
- What are the open issues?
- How you would contribute something?
- How would you use this yourself?

✍️ Lesson-VCS-3: Modify a CodeRefinery example lesson on Github

In this, you will make a change to a lesson on Github, using only the Github interface. (Github account required, and we assume some knowledge of Github. Ask for help in your team if it is new to you!)

- Navigate to the example lesson we have set up: repo, web
- Go to some page and follow the link to go to the respective page on Github. (Alternatively, you can find the page from the Github repo directly).
- Follow the Github flow to make a change, and open a Pull Request with the change proposal:
  - Click on the pencil icon
  - Make a change
  - Follow the flow to make a commit and change. You'll fork the repository to make your own copy, add a message, and open a pull request.

We will look at these together and accept some changes.

✍️ Lesson-VCS-4: Clone and build a CodeRefinery lesson locally

In this exercise, you will use Git to clone one a CodeRefinery lesson and try to preview it locally. It assumes installation and knowledge of Git.

- Use this sample repository: git-intro (or whatever else you would like)
- Clone the repository to your own computer
- Create a virtual environment using the `requirements.txt` contained within the repository.
- Build the lesson.
  - Most people will probably run: `sphinx-build content/ _build/`
  - If you have `make` installed you can `make html`
  - Look in the `_build` directory for the built HTML files.

Overall, this is pretty easy and straightforward, if you have a Python environment set up. The CodeRefinery documentation lesson teaches this for every operating system.

This same tool can be used to build documentation for other software projects and is pretty standard.

✍️ Lesson-VCS-5: (advanced) Create your own lesson using the CodeRefinery format

In this lesson, you'll copy the CodeRefinery template and do basic modifications for your own lesson.

- Clone the lesson template: https://github.com/coderefinery/sphinx-lesson-template
- Attempt to build it as it is (see the previous exercise)
- How can you do tabs?
- How can you highlight lines in code boxes?
- How can you change color, logo and fonts?
- What directives are available?

Summary

❗ Keypoints

- Version control takes teaching materials to the next level: shareable and easy to contribute
- There are different formats that use version control, but we like Sphinx with a sphinx-lesson extension.