# Week 5: Introduction to Using Git

**DATA 510: Data Science Capstone**

Lucas P. Cordova, Ph.D.

2026-06-08

Week 5 companion session: Git for DS3 capstone repos. Deliverable formats (Quarto encouraged, PDF milestones always accepted), solo and team workflows, install on macOS and Windows, clone, commit, push, collaborating on shared files, merges, a brief intro to branches and pull requests, and a hands-on exercise: render the write-up skeleton to PDF, push to GitHub, and submit on Canvas.

## Table of contents

# 1 Learning Objectives

## 1.1 Today's Objectives

## 1.2 What You Will Leave With

By the end of this session, you will be able to:

1. **Explain** why Git is the audit trail for capstone deliverables and weekly DS3 work.
2. **Install and verify** Git on **macOS** or **Windows**.
3. **Clone** your project repo, **commit** changes, and **push** to GitHub.
4. **Choose** a sane solo or team workflow for milestone files in `deliverables/`.
5. **Describe** what merges can and cannot do (text vs PDF), and when branches or pull requests help teams.
6. **Render** the Week 5 write-up skeleton to **PDF**, **commit and push** to your repo, and **upload** the PDF to Canvas.

## 1.3 Before We Touch Git: Deliverable Formats

**Encouraged:** Quarto (`.qmd`) for write-ups, with rendered HTML/PDF committed in `deliverables/` (see the Block 1 lecture today).

**Also permitted:** Google Docs, Microsoft Word, LaTeX, or any editor that exports **PDF**. Drop the **PDF** (and optional source) in the correct milestone folder:

```
deliverables/M1-proposal/
deliverables/M2-data-summary/
deliverables/M3-poster-draft/
deliverables/M4-writeup-draft/
deliverables/M5-final/
```

Git still tracks whatever you commit. I grade the **milestone artifact**, not your toolchain religion.

# 2 Part 1: Why Git for This Course

## 2.1 Git for the Capstone

## 2.2 What Git Gives You

Your DS3 repo is not just file storage. Git is a **time machine** and **evidence log**:

- Peer POs and I can see **what changed** between Iteration Reviews.
- You can recover last Tuesday's notebook when tonight's refactor breaks everything.
- Team members can work in parallel without emailing `final_v2_REAL.pdf`.

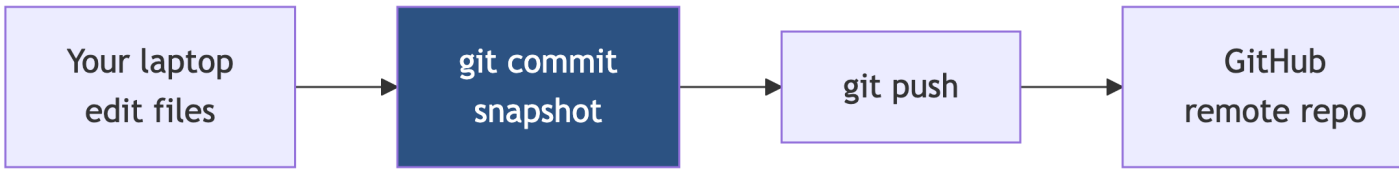
**Course weight:** Git workflow is part of **Project process and practices (5%)** on the syllabus. Commit quality matters weekly, especially in the last five weeks.

### 2.3 GitHub vs Git

| Term | What it is |
|---|---|
| **Git** | Version control on your machine (commits, history, merges) |
| **GitHub** | Hosting + collaboration (remote repo, access control, pull requests) |

You run **Git** locally. You **push** to **GitHub** so collaborators, peer POs, and I can read your repo.

### 2.4 Where Git Fits in DS3



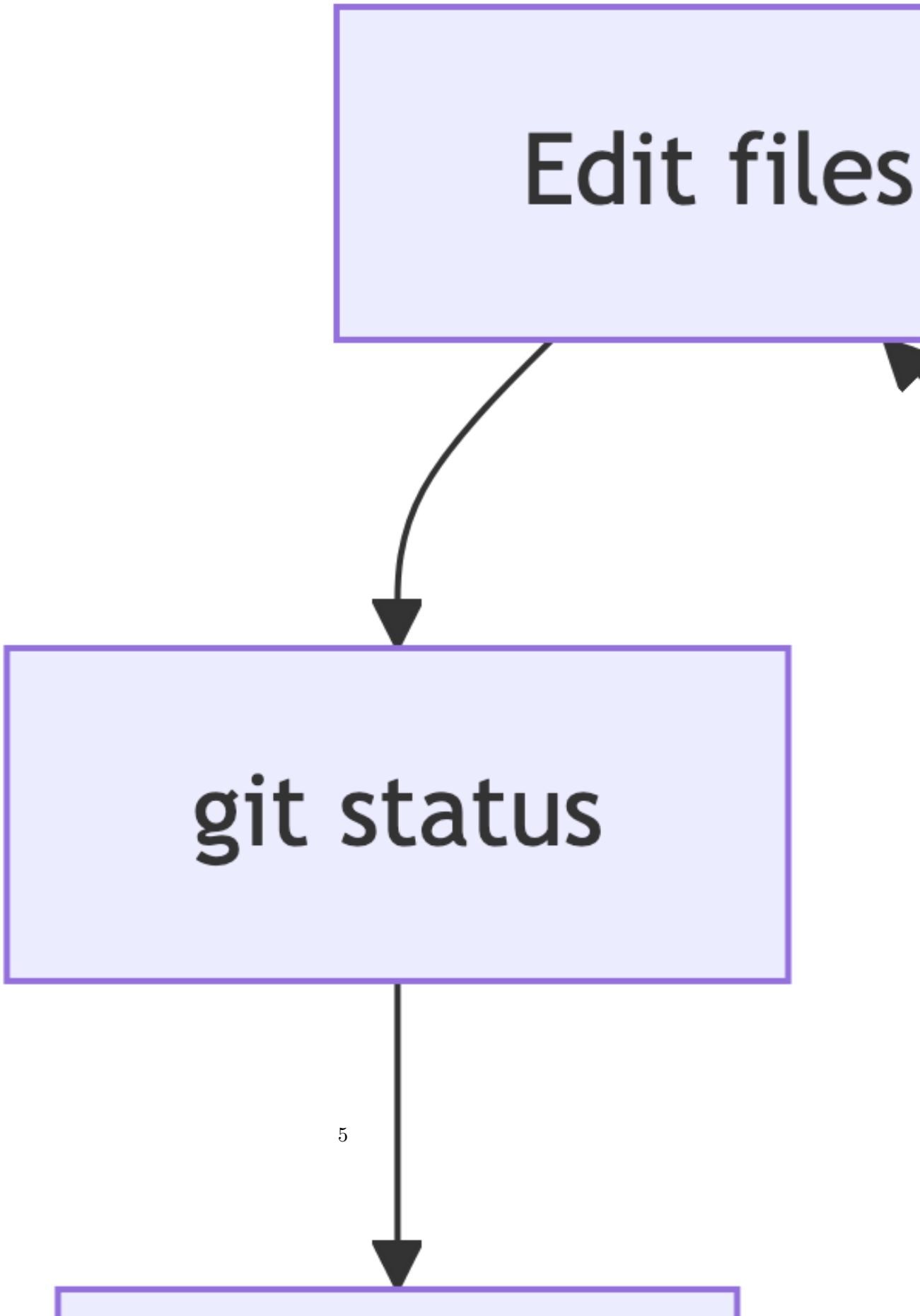

Charter, backlog, code, notebooks, and `deliverables/` all live in one repo. **Commit often** so the story is visible.

# 3 Part 2: Solo Workflow

## 3.1 Solo Git Workflow

## 3.2 Solo: One Person, One Repo

You are the only author. Workflow is linear:


Edit files
git status

**Rule:** If it took more than 30 minutes and you might need it back, **commit**. Small commits beat heroic Sunday dumps.

### 3.3 Solo: What to Commit Each Week

Minimum rhythm I expect:

- `README.md` Iteration Review before class
- Progress on `src/`, `notebooks/`, or `data/processed/` as appropriate
- Milestone PDFs (and `.qmd` source if you use Quarto) under `deliverables/`
- `CHARTER.md` / `BACKLOG.md` updates when they change

**Do not commit:** secrets, huge raw data (check `.gitignore`), `.quarto/` cache, personal notes outside the repo structure.

### 3.4 Solo: Typical Session

```
cd ~/path/to/your-ds3-repo
git status
git add deliverables/M2-data-summary/data-summary.pdf
git add README.md
git commit -m "M2: data summary PDF and week 6 iteration review"
git push
```

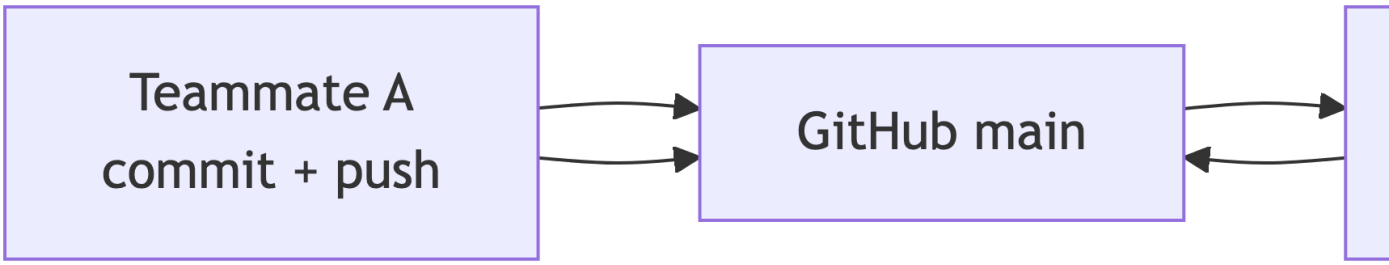
Write commit messages a peer could understand in one line. `fixed stuff` is not a message.

## 4 Part 3: Team Workflow

### 4.1 Team Git Workflow

### 4.2 Teams of 2 or 3: Same Repo, Shared History

One GitHub repo per project. Everyone clones the **same** remote. You coordinate so two people do not silently overwrite each other's work on `main`.




**Before you edit:** `git pull`. **After you finish a logical chunk:** `git add`, `git commit`, `git push`.

### 4.3 Splitting Work Without Collisions

| Strategy | When to use |
|---|---|
| **Different files** | Teo owns `src/ingest/`, Ben owns `notebooks/03_viz.ipynb` |
| **Quarto includes** | One `writeup.qmd`; each person edits `_sections/methods.qmd`, `_sections/results.qmd` |
| **Different milestone folders** | One person leads M2 data doc, another M3 poster (still review together) |
| **Take turns on `main`** | Pull, edit, push quickly; communicate in `#standup` |

If two people edit the **same line** of the **same file** without pulling, Git will ask you to **merge**.

### 4.4 Quarto Teams: Prefer Includes

From the Quarto lecture: split large write-ups with include shortcodes (each teammate owns a section file):

Parent `writeup.qmd` stays stable; merges happen in **smaller files**. Fewer conflicts than one 3,000-line document.

### 4.5 Word / Google Docs Teams

If you write in Word or export from Google Docs:

- One person **owns** integration each week (downloads PDF, places in `deliverables/`, commits).

- Do not commit `.docx` with the same filename from two laptops without pulling first.
- **PDF is the submission artifact**; source docs can live in Drive if your team prefers, but the repo PDF must be current for graders.

# 5 Part 4: Install Git

## 5.1 Install Git

## 5.2 macOS Installation

**Option A (recommended):** Install Xcode Command Line Tools (includes Git):

```
xcode-select --install
```

**Option B:** Install Homebrew then:

```
brew install git
```

Verify:

```
git --version
```

## 5.3 Windows Installation

1. Download **Git for Windows** from git-scm.com/download/win
2. Run the installer (defaults are fine for this course; **Git Bash** and **Git from the command line** are useful)
3. Open **Git Bash** or PowerShell:

```
git --version
```

Optional: GitHub Desktop for a GUI. I still want you comfortable with `commit` / `push` in the terminal for README reproducibility.

## 5.4 First-Time Git Identity

Set once per machine (use your Willamette email):

```
git config --global user.name "Your Name"
git config --global user.email "you@willamette.edu"
```

### 5.5 VS Code Integration

Install the **Git** built-in features (and **GitHub Pull Requests** extension if you use PRs). Source Control panel shows changed files; integrated terminal runs the same commands below.

## 6 Part 5: Clone, Commit, Push

### 6.1 Git Essentials

### 6.2 Clone Your Project Repo

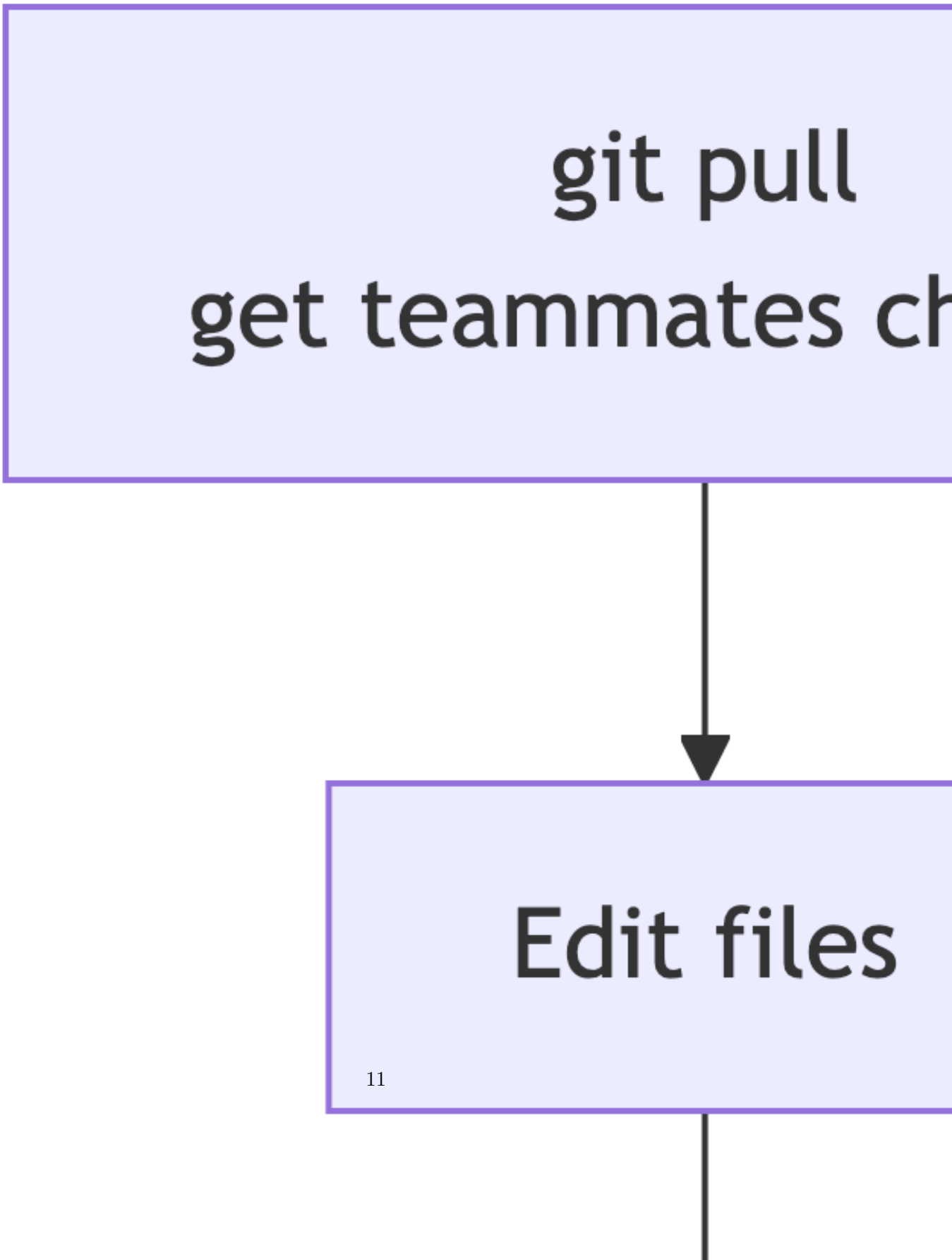
After you create a repo from the DS3 template on GitHub:

```
cd ~/Projects   # or wherever you keep repos
git clone https://github.com/YOUR-ORG/YOUR-REPO.git
cd YOUR-REPO
```

Replace the URL with **your** repo (HTTPS or SSH if you have keys set up). Open the folder in VS Code: **File > Open Folder**.

**Clone once.** After that, use `git pull` and `git push`, not clone again.

### 6.3 Daily Cycle (Diagram)



### 6.4 Commands You Need Tonight

| Command | Purpose |
|---|---|
| `git status` | See modified and untracked files |
| `git add <file>` | Stage a file for the next commit |
| `git add .` | Stage all changes (use carefully) |
| `git commit -m "message"` | Save a snapshot locally |
| `git push` | Send commits to GitHub |
| `git pull` | Download and merge remote changes |
| `git log --oneline -5` | Recent history (sanity check) |

### 6.5 Commit Messages (Capstone Style)

Good:

- `M1: add proposal PDF and charter alignment note`
- `Ingest: AQS pull script and raw snapshot 2026-06-08`
- `README: week 5 iteration review`

Weak:

- `updates`
- `asdf`
- `fixed merge`

## 7 Part 6: Merges and Limits

### 7.1 Merges and Collaboration

### 7.2 When Git Says "Merge Conflict"

If you and a teammate edit the **same lines** in a **text file** (`.qmd`, `.py`, `.md`, `.R`), the second person to push may need to **merge**:

1. `git pull`
2. Git marks conflicts in the file with `<<<<<<<`, `=======`, `>>>>>>>`
3. Edit the file to the correct combined version
4. `git add` the resolved file
5. `git commit` (merge commit)
6. `git push`

VS Code has a merge editor that color-codes choices. Ask for help the first time; it is a rite of passage.

### 7.3 Binary Files: PDFs Do Not Merge

**PDF, PNG, `.docx`, and most binaries cannot be merged line-by-line.** Git can store them, but if two people commit different versions of `proposal.pdf`, Git picks one history or forces you to **choose a file**, not blend contents.

**Team practice for PDF milestones:**

- One person commits the canonical PDF per milestone submission, **or**
- Use clear filenames with dates (`proposal-draft-2026-06-08.pdf`) until you agree on final, **or**
- Work in **text source** (Quarto, LaTeX) where merges work, then render PDF once

### 7.4 Merge vs Rebase (One Slide)

**Merge:** Preserves branching history; creates a merge commit. Default and fine for this course.

**Rebase:** Rewrites history to look linear. Advanced; optional for solo polish, **avoid on shared `main`** unless your team knows what they are doing.

You do not need rebase to pass DATA 510. You need **pull, resolve, push**.

## 8 Part 7: Branches and Pull Requests (Advanced)

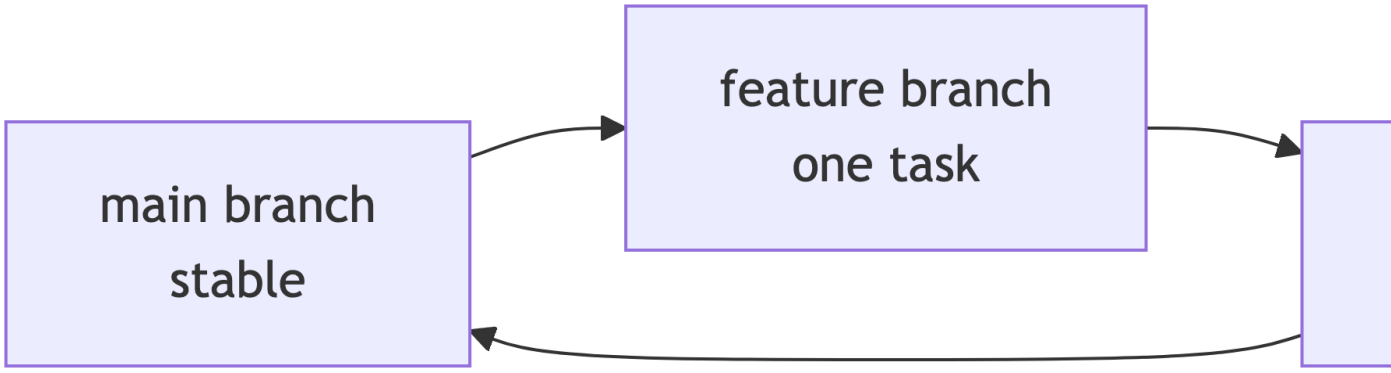
### 8.1 Branches and PRs

### 8.2 Working on Branches (Teams)

For parallel feature work without blocking `main`:

```
git checkout -b feature/aqs-ingest
# ... edit, commit ...
git push -u origin feature/aqs-ingest
```

Open a **Pull Request (PR)** on GitHub: teammate reviews, then **merge** into `main`. Everyone `git pull` on `main` afterward.




**Solo students:** optional. Working directly on `main` with frequent commits is acceptable.

### 8.3 When PRs Help

- Team of 3 with overlapping code areas
- You want a peer PO or teammate to **approve** before merging risky pipeline changes
- You are experimenting and might throw the branch away

**Minimum bar for the course:** visible commits on `main` (or merged PRs) before milestones. Empty repos on due dates hurt everyone.

## 9 Part 8: Milestones and Habits

### 9.1 Milestones on GitHub

### 9.2 `deliverables/` and Git

Each milestone folder should contain:

- **PDF** required for Canvas-style grading paths
- **Source** when you use Quarto (`.qmd`, `refs.bib`, figures)
- Short `README.md` in the folder (optional but kind): status, date submitted, known gaps

Commit **both** source and rendered PDF when using Quarto so peer POs can open HTML/PDF without rendering.

### 9.3 Commit Often (Seriously)

| Bad habit | Better habit |
|---|---|
| One 200-file commit Sunday night | Daily commits per task or PBI |
| "It works locally" but never pushed | Push at end of each work session |
| Email files to teammates | Push to GitHub; link in Discord |

If your laptop dies, **unpushed commits are gone**. GitHub is the backup your team shares.

### 9.4 What I Look For

- Repo URL works; `LucasCordova` and peer POs have access
- Commit history tells a story aligned with your Projects board
- Milestone PDFs exist in the right folders on due dates
- No secrets, no broken `.gitignore` surprises

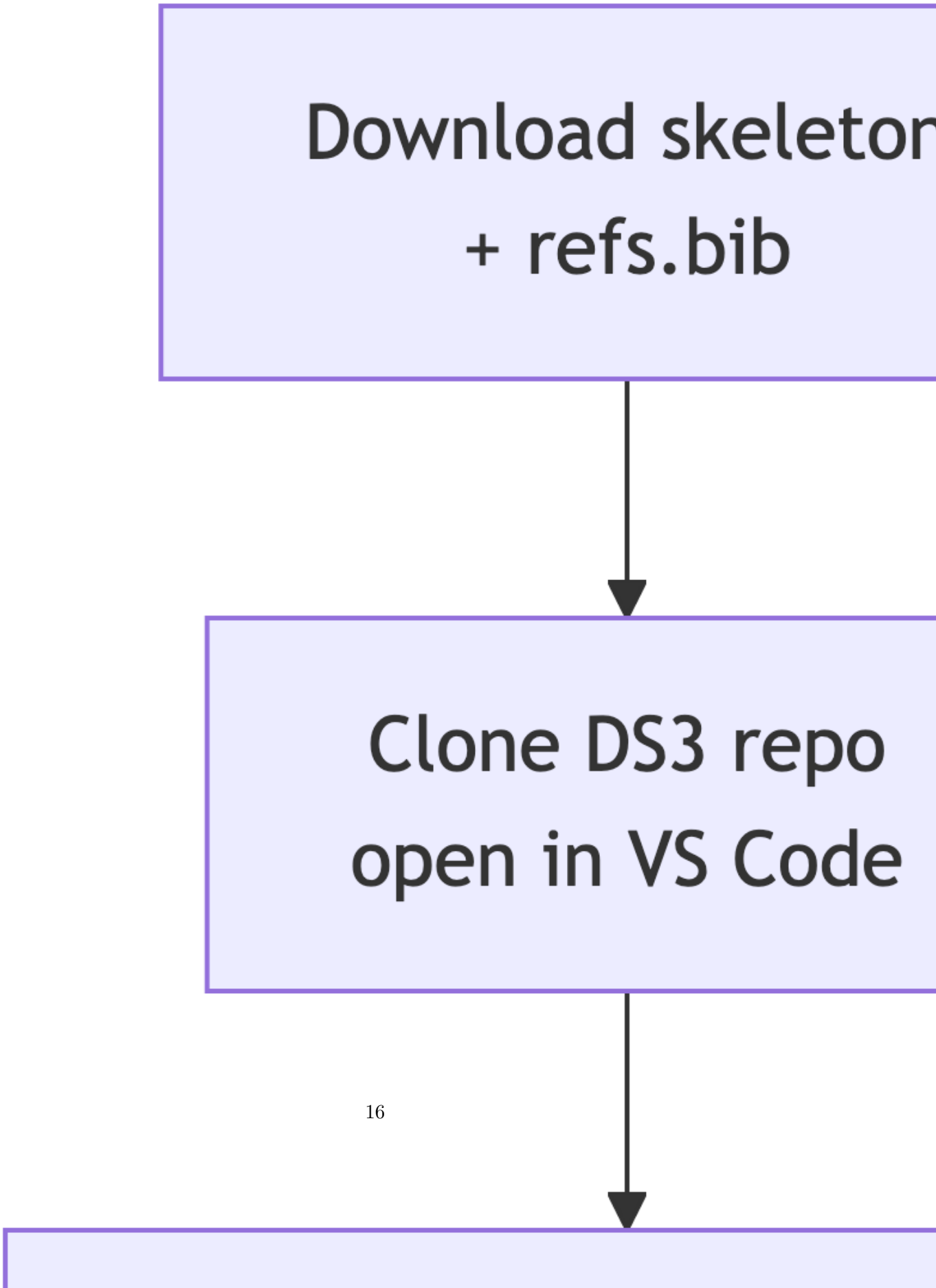
## 10 Part 9: Final Exercise (Skeleton PDF)

### 10.1 Final Exercise

### 10.2 What You Will Do Tonight

This is the hands-on we deferred from the Quarto lecture because many of you have **not cloned your repo yet**. You will:

1. Get the **write-up skeleton** template files
2. **Clone** (or open) your DS3 repo
3. **Render** a PDF from the skeleton
4. **Commit and push** source + PDF to GitHub
5. **Upload** the PDF to the **Canvas assignment** for week 5


Download skeleton
+ refs.bib
Clone DS3 repo
open in VS Code

### 10.3 Step 1: Download Template Files

From the Block 1 Quarto lecture (course site):

- writeup-skeleton.qmd
- refs.bib (required for render; keep next to the `.qmd`)

Save both to your laptop. You will copy them into your repo in Step 3.

### 10.4 Step 2: Clone Your DS3 Repo

If you already have the repo locally, open that folder in VS Code and run `git pull`. Otherwise:

```
cd ~/Projects   # or your preferred folder
git clone https://github.com/YOUR-ORG/YOUR-REPO.git
cd YOUR-REPO
code .          # VS Code: File > Open Folder
```

Create the repo from the DS3 template on GitHub first if you have not already.

### 10.5 Step 3: Copy Into `deliverables/M4-writeup-draft/`

```
mkdir -p deliverables/M4-writeup-draft
cp ~/Downloads/writeup-skeleton.qmd deliverables/M4-writeup-draft/writeup.qmd
cp ~/Downloads/refs.bib deliverables/M4-writeup-draft/refs.bib
```

Adjust paths to where you saved the downloads. **Rename** the skeleton to `writeup.qmd` so the output PDF is `writeup.pdf`.

Optional tonight: paste a sentence or two from your M1 proposal into the Introduction stub. Empty stubs are fine for this exercise; I am checking that **render + Git + Canvas** work.

### 10.6 Step 4: Render PDF

From your **repo root** in the VS Code terminal:

```
quarto render deliverables/M4-writeup-draft/writeup.qmd
```

Or Command Palette: **Quarto: Render** with `writeup.qmd` open.

**Success:** `deliverables/M4-writeup-draft/writeup.pdf` exists and opens. If PDF fails, run `quarto install tinytex` once, then try again. HTML output is a bonus; **Canvas wants the PDF**.

### 10.7 Step 5: Commit and Push

```
git status
git add deliverables/M4-writeup-draft/writeup.qmd
git add deliverables/M4-writeup-draft/refs.bib
git add deliverables/M4-writeup-draft/writeup.pdf
git commit -m "Week 5: write-up skeleton PDF in M4 folder"
git push
```

Commit **source and PDF** so peer POs and I can open artifacts without re-rendering.

### 10.8 Step 6: Upload PDF to Canvas

Open the **Week 5: Write-Up Skeleton PDF** assignment on Canvas. Upload **`writeup.pdf`** from `deliverables/M4-writeup-draft/`.

**Solo:** you submit your own PDF.

**Team: one person** uploads for the team; list every member in the Canvas comment or text box as the assignment instructs.

## 11 Wrap-Up

### 11.1 Before You Leave

### 11.2 Tonight's Checklist

- ☐ `git --version` works (Mac Terminal or Windows Git Bash)
- ☐ `user.name` and `user.email` configured
- ☐ DS3 repo cloned and opened in VS Code
- ☐ Skeleton rendered to **`writeup.pdf`** in `deliverables/M4-writeup-draft/`
- ☐ Source, `refs.bib`, and PDF **committed and pushed**
- ☐ **`writeup.pdf` uploaded to Canvas** (Week 5 assignment)
- ☐ You know **PDF binaries do not merge** and teams need a plan

Block 2 continues with data engineering consultations after this exercise. Use Git between conversations, not instead of talking to your team.

### 11.3 Quick Reference Card

```
git pull
# edit files
git status
git add <files>
git commit -m "clear message"
git push
```

Questions: Discord `#general` or `#blockers`, or grab me in studio.

## 12 References

### 12.1 References


1. Chacon, S., & Straub, B. *Pro Git* (free online). https://git-scm.com/book/en/v2
2. GitHub Docs. *Set up Git.* https://docs.github.com/en/get-started/git-basics/set-up-git
3. GitHub Docs. *Resolving a merge conflict.* https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts
4. Cordova, L. P. DATA 510 DS3 template. https://github.com/LucasCordova/ds3template
5. Cordova, L. P. Week 5 Quarto lecture. https://courses.lpcordova.phd/data510/lectures/05-1-techwriting-quarto/