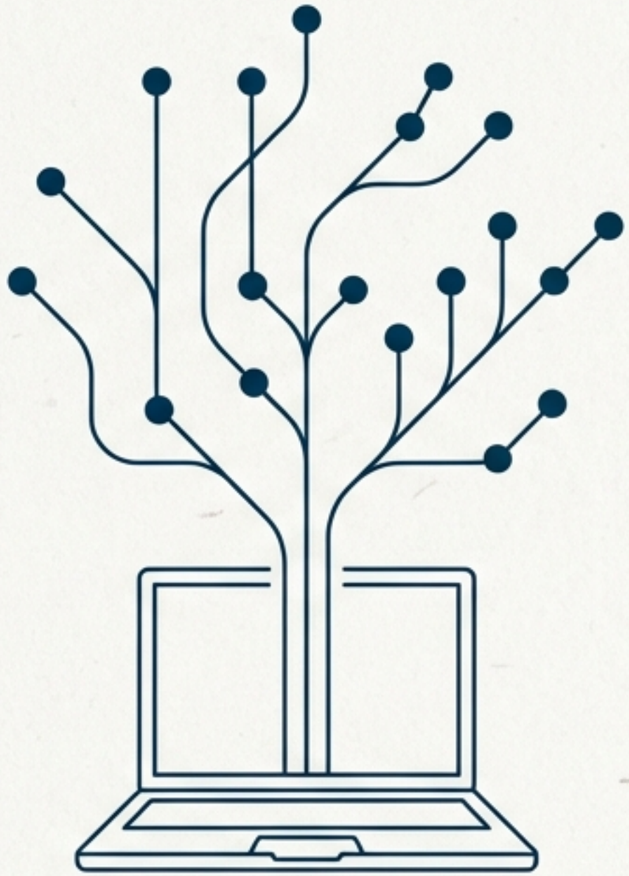
Week 1: The Infrastructure of Science

Git & Development Tools

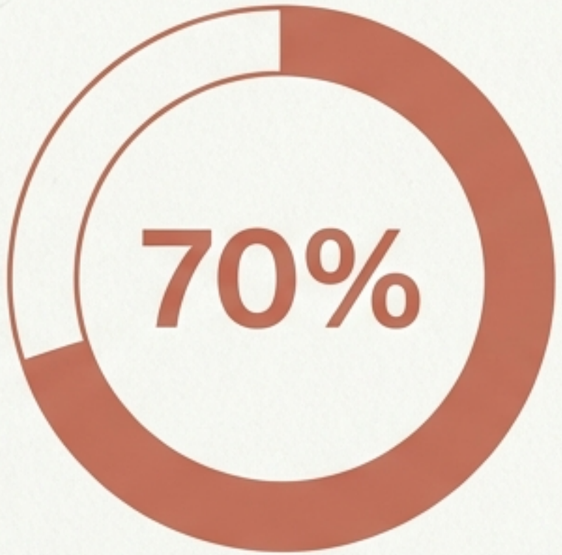


How do we prevent the 70% failure rate in scientific reproducibility?

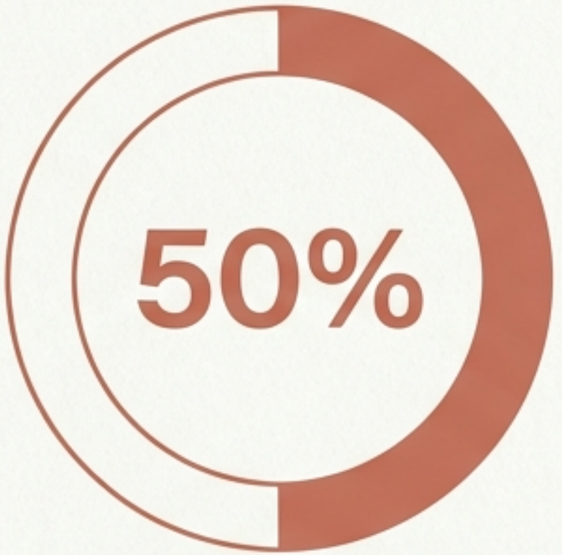
Numerical Scientific Computing
Aalborg University

The Reproducibility Crisis

Source: Nature Survey of 1,576 Researchers (2016)



Failed to reproduce ANOTHER scientist's experiments.



Failed to reproduce THEIR OWN experiments.

The Consensus

90% of researchers agree there is a significant crisis in science.

The Computational Cause

- Code 'available upon request' (lost)
- Missing dependencies
- Unlinked versions

The Reality

Reproducing the original chart from the Nature survey, explicitly showing the breakdown of successful vs unsuccessful reproduction attempts across disciplines.

The Cost of Bad Infrastructure

Real-world consequences of manual version control.

The Horror Stories

The Retracted Paper

Dr. Sarah Chen overwrote working code. Results lost. Paper retracted.

The Collaboration Disaster

PhD group had 47 versions of `analysis.py`. Missed conference deadline.

The Wasted Years

Published method unusable due to messy ZIP file. 18 months of research lost.

The Manual Trap

```
analysis_final.py
analysis_final_v2.py
analysis_final_fixed_SARAH.py
analysis_REAL_FINAL.py
analysis_OLD_DO_NOT_USE.py
```

“Final_v2.py” is not a strategy. It is a liability.

What is Version Control?

Scientific Infrastructure, not just software.

Time Machine

Revert to any previous state instantly.
Undo mistakes.

Lab Notebook

Automatically records who changed what and why.

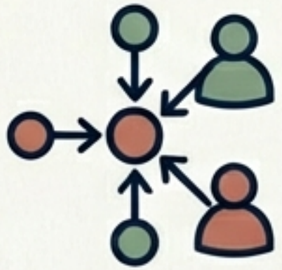
Git

Safety Net

Experiment in ‘branches’ without breaking main code.

Collaboration

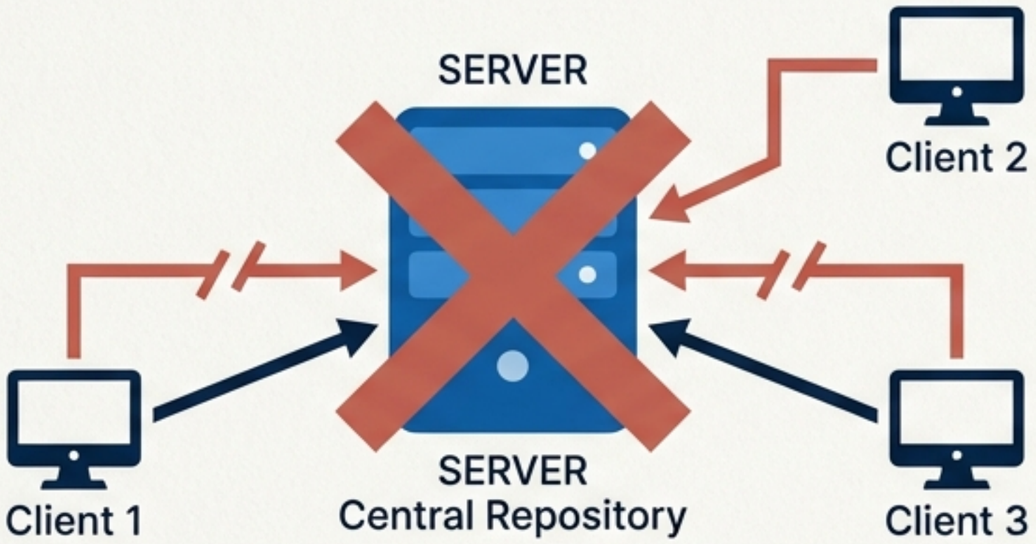
Merge work from multiple people without overwriting.



Architecture: Centralized vs. Distributed

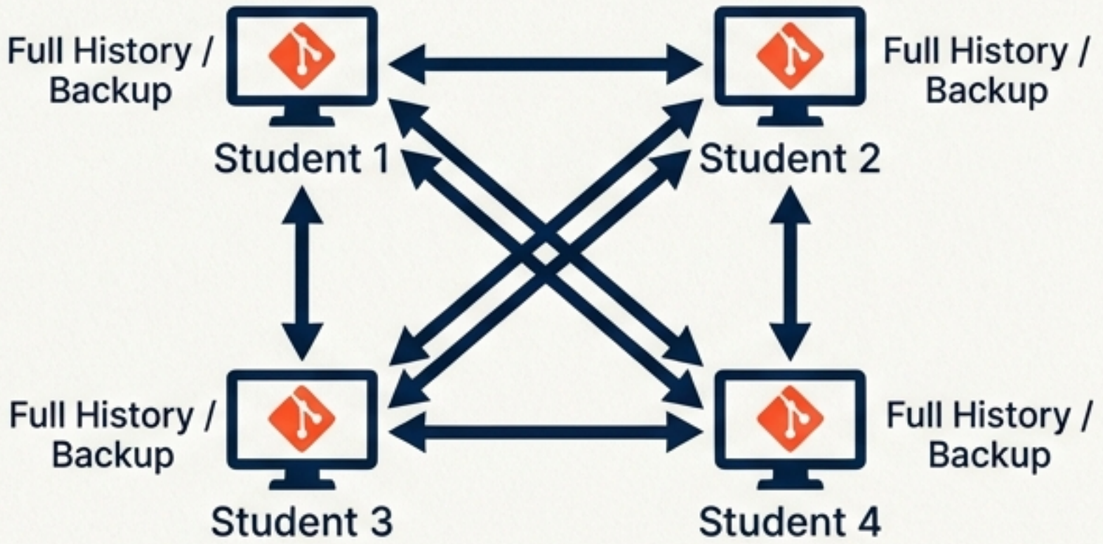
Why Git is resilient.

Centralized (Old/Risky)



Single Point of Failure.
If Server dies, work stops.

Distributed (Git/Resilient)



Every clone is a full backup.
Offline capable. Cryptographically sealed.

In Git, every student has the full history of the project on their own machine.

The Ecosystem

The Tool vs. The Platform

Git (The Tool)

- Local software (CLI)
- Tracks history
- Manages versions
- Works offline

`push` / `pull`

GitHub (The Platform)

- Remote hosting
- Collaboration (Pull Requests)
- Professional Portfolio
- Open Science Hub

The Three-Stage Workflow

The most critical mental model for today.

Working Directory

Your Sandbox

`git add`

Staging Area

The Loading Dock

`git commit`

Repository

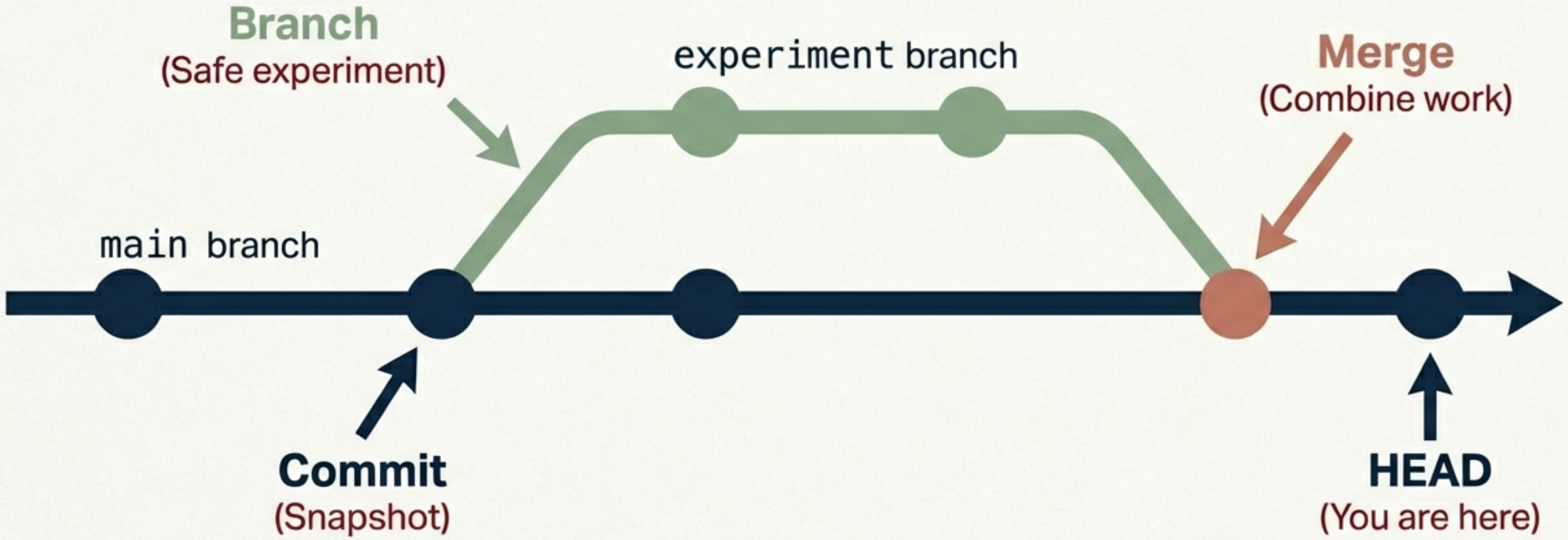
Permanent History

A sealed snapshot.

You must ADD files to the staging area before you can COMMIT them to history.

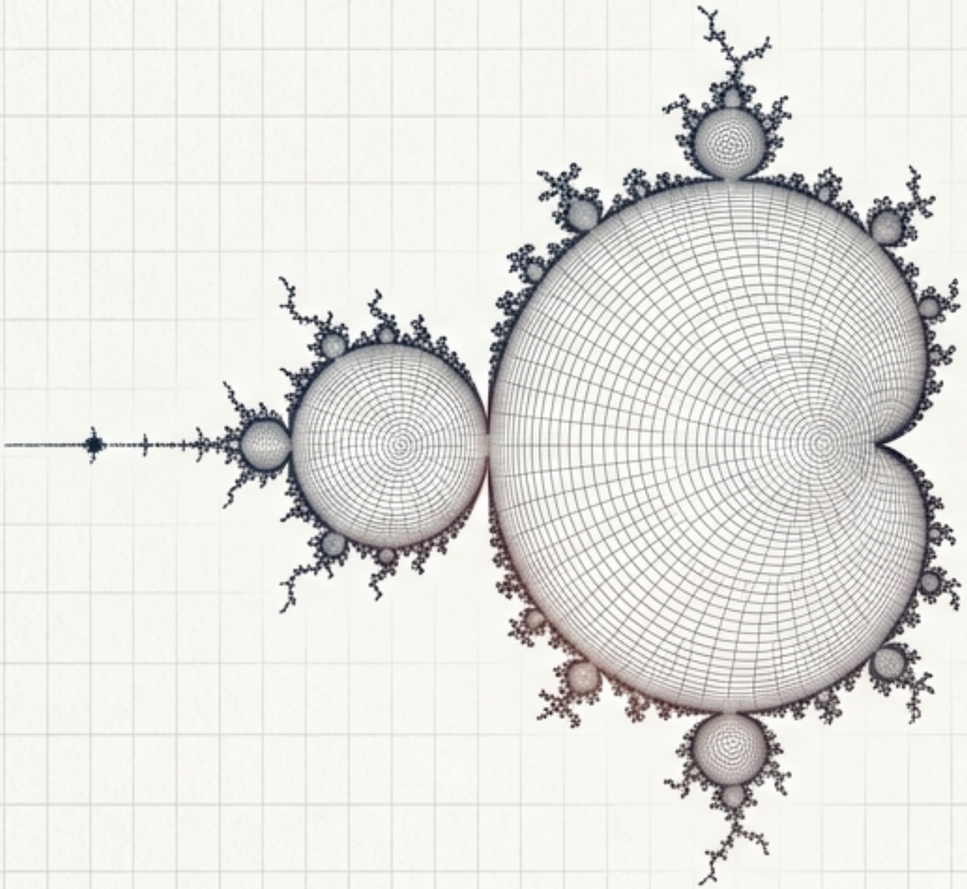
Visualizing History

Commits, Branches, and Merges



Today’s Mission: The Mandelbrot Set

Establishing the Baseline



Goal: Calculate the baseline performance for our semester-long optimization project.

The Steps:

1. Initialize a Git repository.
2. Implement **Naive Python** solution.
3. Measure execution time (Target: **~45 seconds**).
4. Commit the baseline.

The Journey:

Week 1: `45.00s` **(Baseline)**
Week 9: `0.02s` **(GPU)**

Git tracks every step of this 2000× speedup.

Cheat Sheet: Setup & Creation

One-time commands for starting a project.

Create

```
git init
```

Turn current folder into a repo.

Check

```
git status
```

Always check this first. Shows tracked/untracked files.

Download

```
git clone [url]
```

Download an existing repo to your machine.

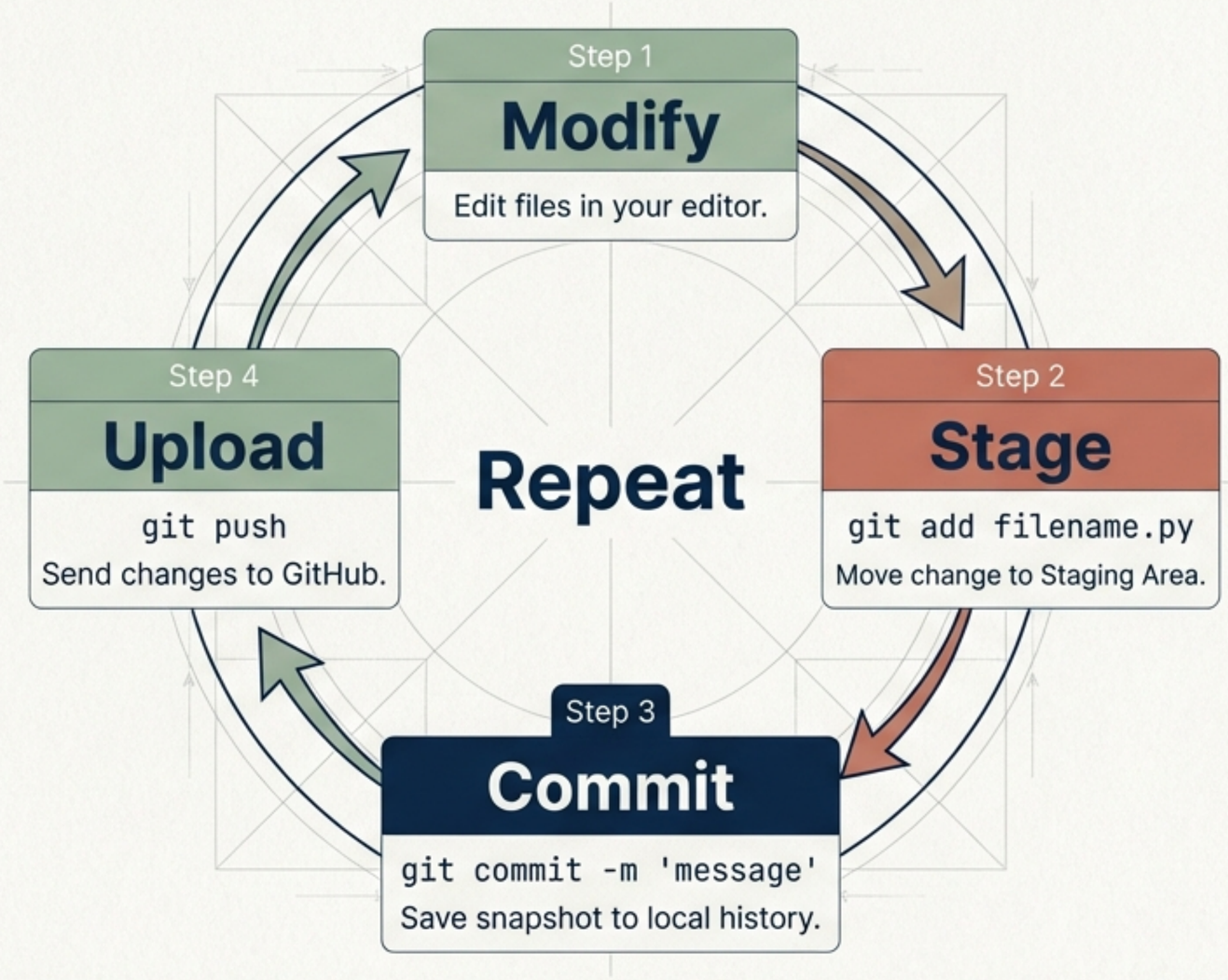
Connect

```
git remote add origin [url]
git push -u origin main
```

Link local repo to GitHub and upload first version.

Cheat Sheet: The Daily Cycle

The loop you will perform hundreds of times.



Best Practices: Best Practices

Rules of the Road for Professional Science

DO

- **Commit Often:** Save small, logical units of work.
- **Write Meaningful Messages:** Explain *why* you made the change.
- **Use** `.gitignore`: Block `__pycache__`, binaries, and datasets.

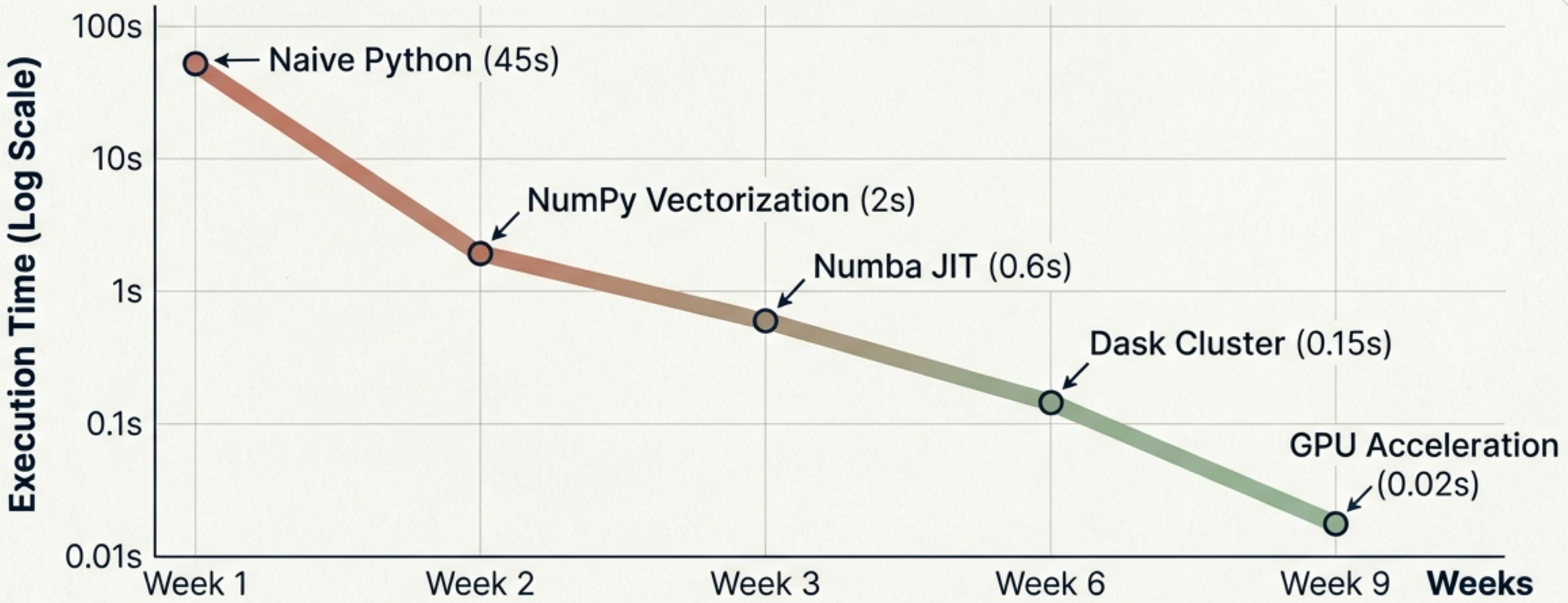
DON’T

- **Don’t commit broken code** to the main branch.
- **Don’t commit secrets** (passwords/API keys).
- **Don’t force push** (`git push -f`) unless absolutely necessary.

Treat your commit log like a lab notebook. It is evidence of your work.

The Optimization Journey

Why we are building this infrastructure.



Git provides the evidence for this portfolio.
Peer reviews will check your commit logs to verify authenticity.

Summary & Action

Ready for Studio.

Key Takeaways

1. **Crisis:** Lack of version control leads to lost science (70% failure rate).
2. **Infrastructure:** Git is your digital lab notebook.
3. **Workflow:** Modify → `add` → `commit` → `push`.
4. **Career:** Your GitHub profile is your professional portfolio.

Open Terminal.
Initialize Repository.
Start the Mandelbrot Challenge.

Next Week: Computer Architecture & Memory Hierarchy