From Email Chaos to Seamless Collaboration: How Git Could Transform Research Teamwork

Supervisor: prof. dr hab. inż. Zbigniew Galias,
Assistant supervisor: dr inż. Bartłomiej Garda

Karol Bednarz

April 2025

The Research Collaboration Challenge

- Multiple versions of documents scattered across emails
- `Final_v3_REAL_FINAL.docx` syndrome
- Lost work due to accidental deletions or overwrites
- Difficulty tracking who changed what and when
- Challenges merging contributions from multiple researchers
- No clear history of project evolution

What is Git?

Git is a distributed version control system that:

- Tracks changes in files over time
- Enables seamless collaboration between multiple contributors
- Maintains complete history of project evolution
- Works with any file type (code, documents, data, images)
- Operates both locally and remotely

A sophisticated ”track changes” system for your entire research project

Why Git Could Matters for Researchers

Core Benefits

- **Reproducibility:** Complete record of how research evolved
- **Collaboration:** Multiple researchers can work simultaneously without conflicts
- **Backup:** Distributed nature means your work is safely stored in multiple locations
- **Experimentation:** Try new approaches without fear of losing previous work
- **Transparency:** Clear attribution of contributions and changes

Key Git Concepts

Repository (Repo)

Your project folder with complete version history

- `git init` - Create a new repository
- `git clone https://github.com/user/project.git` - Copy existing repository

Commit

A snapshot of your project at a specific point in time

- `git add <filepath/.>` - Stage files for commit
- `git commit -m "Add analysis results"` - Save snapshot with message
- `git log` - View commit history

Branch

Parallel version of your project for testing new ideas

- `git branch experiment` - Create new branch
- `git checkout experiment` - Switch to branch
- `git branch -a` - List all branches

Merge

Combining changes from different branches or contributors

- `git checkout main` - Switch to main branch
- `git merge experiment` - Merge experiment branch into main

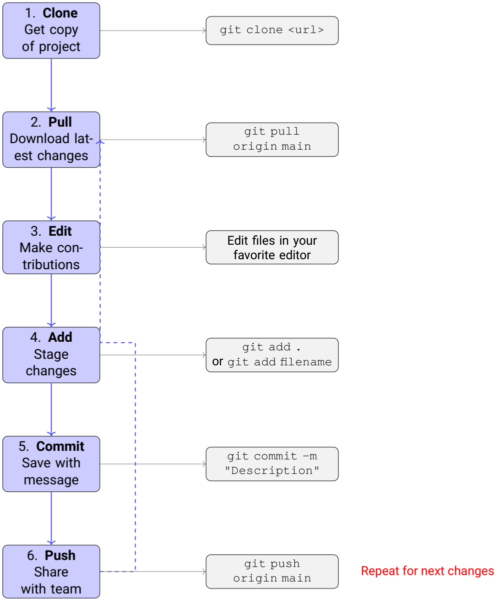
Remote

Online copy of your repository (e.g., GitHub, GitLab)

- `git remote add origin https://github.com/user/repo.git` - Add remote
- `git push origin main` - Upload changes to remote
- `git pull origin main` - Download changes from remote

Git Workflow for Research Teams

Git Collaboration Workflow



1. **Clone** Get copy of project — `git clone <url>`

2. **Pull** Download latest changes — `git pull origin main`

3. **Edit** Make contributions — Edit files in your favorite editor

4. **Add** Stage changes — `git add .` or `git add filename`

5. **Commit** Save with message — `git commit -m "Description"`

6. **Push** Share with team — `git push origin main`

Repeat for next changes

Key Points

- Start with **Clone** only once per project
- Always **Pull** before making changes
- The cycle **Pull** → **Edit** → **Add** → **Commit** → **Push** repeats
- Use descriptive commit messages for better collaboration

Aplications and Possible Scenarios

Aplications

Manuscript Writing:

- Track revisions and reviewer responses
- Collaborate on papers with multiple authors
- Maintain different versions for different journals

Data Analysis:

- Version control for analysis scripts
- Track parameter changes and results
- Share reproducible analysis pipelines

Research Scenarios

Scenario 1: Multi-author Paper

- Each author works on different sections
- Git merges contributions automatically
- Complete history of who wrote what

Scenario 2: Code development

- Collaboratively develop analysis tools
- Test new features without breaking working code
- Easy rollback if something goes wrong

Git online platforms

As a distributed version control system, Git does not require a separate server application. However, there are services that extend the original software, primarily by adding access control, support for managing multiple repositories, and a web interface.

GitHub

- Most popular platform
- Excellent for open science
- Free private repositories for research

GitLab

- Strong institutional options
- Built-in CI/CD for automated testing
- Self-hosted options available

Common concerns

”It’s too technical”

- Modern GUI tools make Git accessible
- Learn gradually, starting with basic operations
- Focus on concepts, not commands

”We don’t write code”

- Git works with any file type
- Particularly valuable for text-based work
- Many researchers use it for papers and documentation

”Our files are too large”

- Git LFS (Large File Storage) handles big files
- Consider data management strategies
- Use selective repository organization

File Type Recommendations

Recommended

Text-based files:

- Source code (.py, .r, .m, .cpp)
- LaTeX documents (.tex, .bib)
- Markdown files (.md, .txt)
- Configuration files (.yaml, .json, .xml)
- Scripts and makefiles
- CSV data files (small-medium)
- Documentation files

Not Recommended (could be excluded using `.gigignore` file)

Binary/Large files:

- Images/figures (.jpg, .png, .pdf)
- Large datasets (>100MB)
- Video/audio files
- Compiled executables
- Office documents (.docx, .xlsx)
- Temporary/cache files
- Log files

Learning resources

Courses:

- GitHub Skills url: *https://skills.github.com/*
- Git-it Tutorial url: *https://jlord.computer/git-it/*
- Codecademy Git Course url: *https://www.codecademy.com/learn/learn-git*
- freeCodeCamp Git Tutorial url: *https://www.youtube.com/watch?v=zTjRZNkhiEU*
- GIT CHEAT SHEET url: *https://education.github.com/git-cheat-sheet-education.pdf*

Research-Specific:

- Version Control for Scientists url: *https://www.youtube.com/watch?v=ohTW4FJdmeQ*
- Git for scientists - Neurath's boat url: *https://neurathsboat.blog/post/git-intro/*

Thank You for attention.