September 2025
“By Your Command” And “git + Github”

Spoken interface to robots
git + github - zero to hero in 60 min

Agenda

- DPRG News - RoboColumbus
- Visitors introduction
- Questions?
- “By Your Command” demo and further explanation
- git + Github
- Lunch
- Show’n’tell
- University collaboration

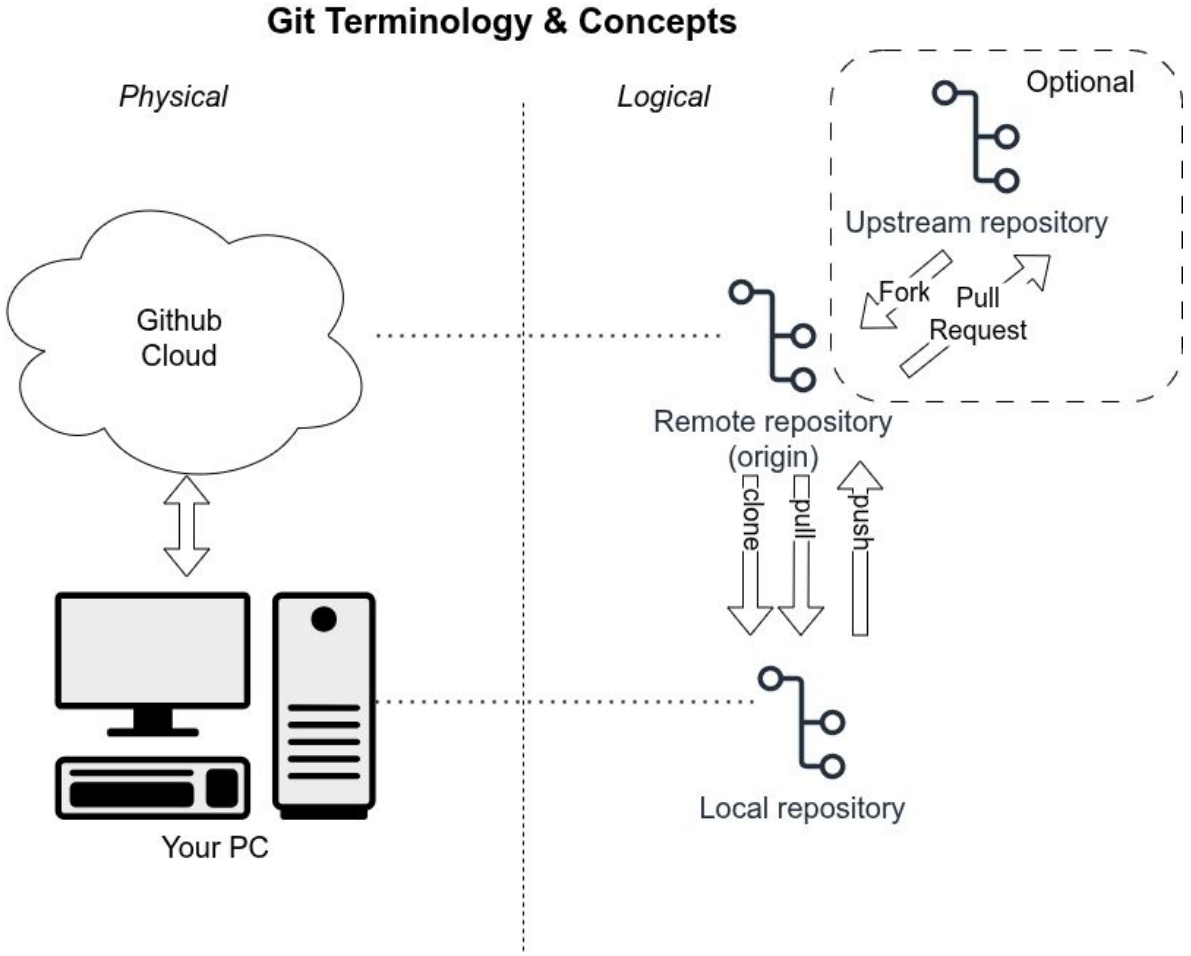

By Your Comand

git + Github

Basics + some advanced techniques

What is git + Github

- Github keeps a copy of your repository in the cloud
- git manipulates the local repository version history
    - Provides for syncing it with the cloud version (push/pull/clone commands)
- Github provides for creating new cloud repos, and interacting with optional upstream repos (fork, pull request)
- Git tracks diffs between versions
    - Diffs of text files keep commit sizes small
    - binary files supported (schematics, 3D models)



Why use git + Github? A ton of reasons...

Robotics centrally includes software/firmware. You need to use proper techniques in its development. (There's some overhead in managing the local, and optional remote repos.) Git enables good techniques, Here's how:

- Cloud backup of your code saves you if you lose your code folder on your PC.
  - It's a lot harder to delete a github repo than a directory.
- Save intermediate "working states" of your code, and easily get back to them for testing
- Switch between multiple configs - e.g. between two robot configs, or two sensor configs
- Avoid "multiple copies of code directories" syndrome.
  - Easily make changes to test something without changing the known-good version (difficult without multiple copies if not using git)
- Fearless refactoring or trying out new libraries/features.
  - Git encourages bold changes. Start a major refactoring on a branch and abandon it if it doesn't work out. Safety net gives you confidence to try big changes.

Why use git + Github (cont.)?

- Document changes (vs. documenting lines of code) with comments in the commit message.
  - Supports long-term project archeology - months or years later you can see not just what you changed but why.
- Sync your code directory between multiple machines. Get warnings about conflicting changes between them.
  - Scenarios: same codebase on PC and on Raspberry Pi used for different purposes
- Maintain a "Release" version that's always ready to run, while you develop on a branch.
  - Helps you quickly isolate whether a problem that arises in development is due to the SW change or if HW broke
- Maintain version control of HW files (3D cad models, PCB designs etc), as well as software & firmware.

Tools, Getting Started

Git is multi-platform - Windows, Linux, MacOS, x86 + Arm

- Install git on your platform. Check you have gitk (should come with git)
  - You don't need the github CLI (gh). Use the website instead.
- If you use vs-code, install the git extension
- If you don't have a Github account, create one at github.com
  - You'll have to set up ssh keys and 2FA

Getting Started: Edit .gitconfig in your home directory

The .gitconfig file in your home directory contains global settings.

```
[user]
        email = paul.bouchier@gmail.com
        name = Paul Bouchier
[alias]
    s = status
    co = checkout
[diff]
    tool = meld
[difftool]
    prompt = false
[difftool "meld"]
    cmd = meld "$LOCAL" "$REMOTE"
[merge]
    tool = meld
[mergetool "meld"]
    cmd = meld "$LOCAL" "$MERGED" "$REMOTE" --output "$MERGED"
```

Advanced Git Use

- git remote -v # display where the remote repo is
- .gitignore # configure files & directories that should be ignored by git
- git push -f # forced push makes remote match local repo
- git revert (before push & after push) # Back out a commit
- git cherry-pick <hash> # pull a single commit from one branch to another
- git blame <file> # Find out who last changed each line of a file
- git rebase -i HEAD~<n> # Squash n previous commits into a single commit by rewriting history
- git bisect # Seek to the commit that broke something

Basic Demos

Assumption: You have created an account

- Create & Clone new repo
- Copy some local content into new repo. Edit README.md
- View local repo status
  - git status + vs-code + gitk + git difftool
- Add to staged & commit and push it to github.

Advice:

- Commit when you reach a point that you consider “a significant accomplishment”, and to which you might want to return
- Push whenever you commit

Create repo

Browse to github.com/<your_username>

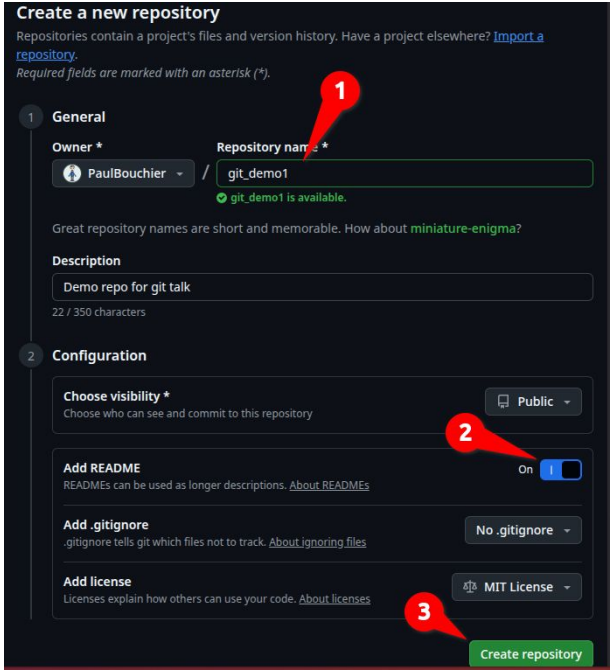
1. Click “Repositories” tab
2. Click “New”

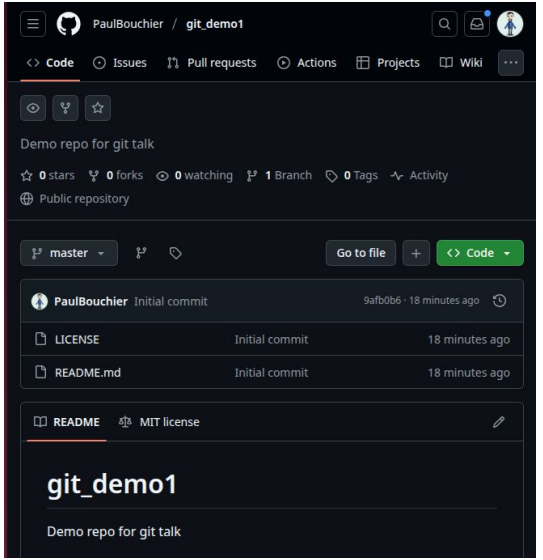
In the “Create New” page

1. Enter a name for your repo
2. Click “Add REAME” on
3. Click “Create repository

The page changes to show the contents of your new repo with some default files.

You have now created a repo in the cloud. Now you need to clone it onto your PC

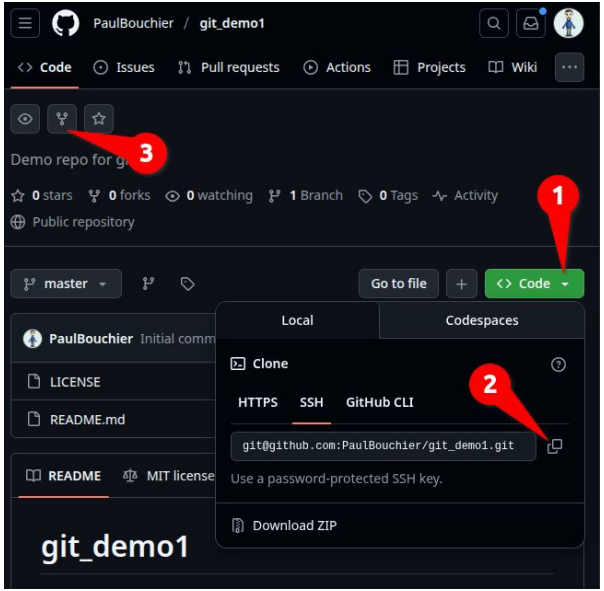




Clone Repo onto your PC

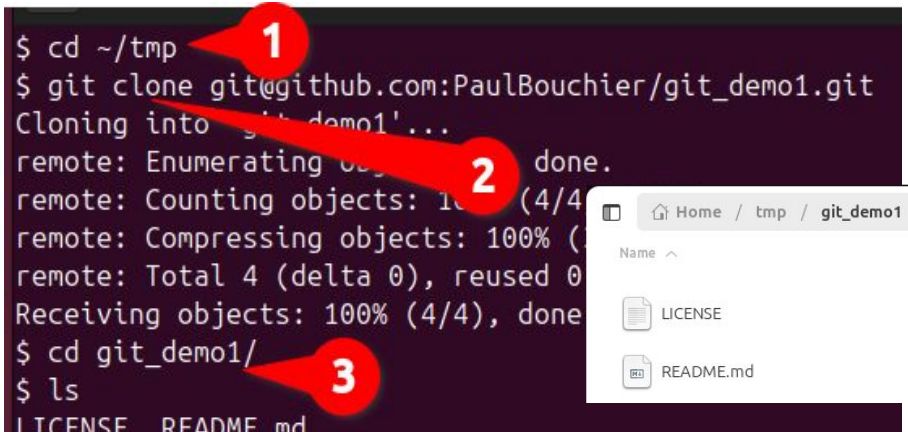
This procedure works for any repo on github, not just one you created

1. On the github repo page, click the arrow next to the “Code” button
2. Click the “copy” icon next to the URL to copy the URL you will clone onto your PC
3. Fork any repo with this button



Open a terminal window

1. cd to the folder you want to clone into (e.g. src)
2. Type “git clone <paste URL>
3. Show contents of <repo_name> folder

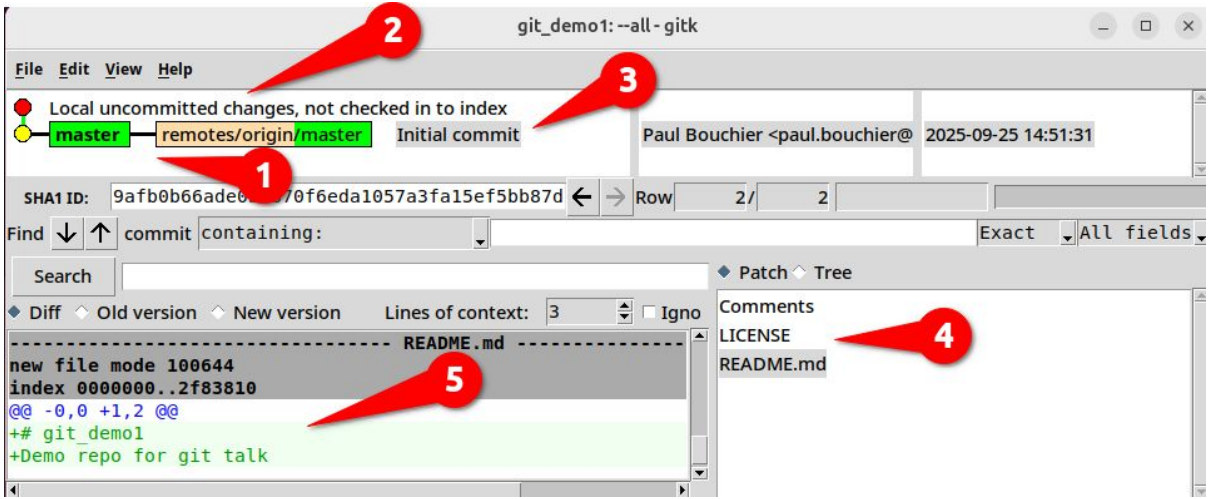


You have now cloned the remote repo into a local copy having the same folder name as your repo

Making and inspecting changes - gitk

Copy some local content into new repo. Edit README.md

Launch gitk



Observe

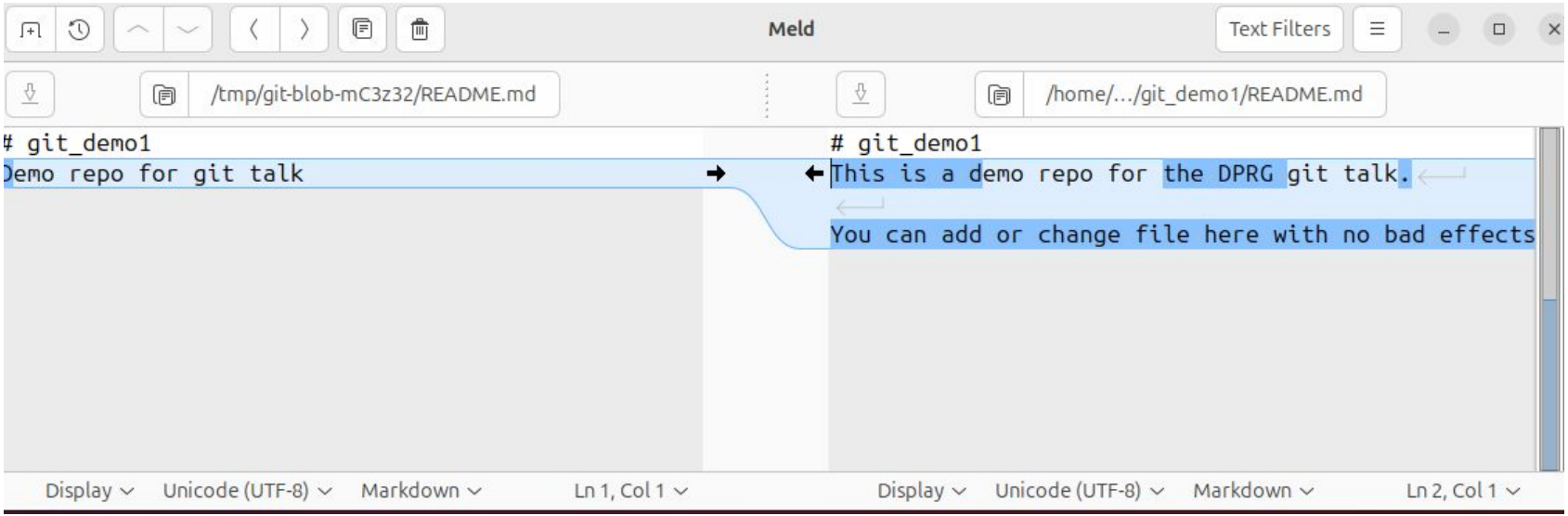
1. Local and remote master are the only commit, and are the same
2. There are uncommitted changes
3. Shaded commit comment shows selected commit. Hash in SHA1 ID
4. Lower right pane shows files changed in selected commit
5. Lower left pane shows changes to each file that was changed

Resize gitk panes, window. Click on other lines in git history. Run gitk in other repos you have downloaded.

Inspecting changes - git difftool

In command window, type “git difftool”

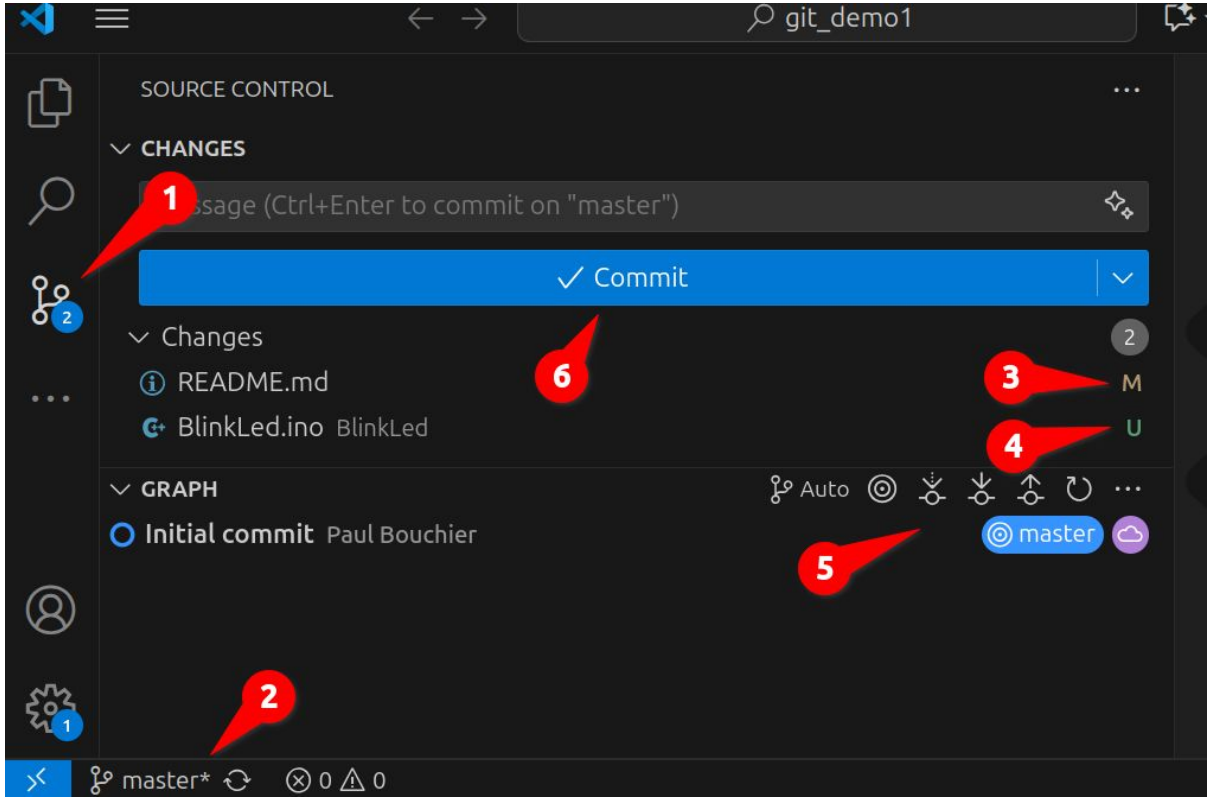
Observe meld shows changes to each modified file



Inspecting changes - vs-code

Launch vs-code in the repo.

1. Select the git extension
2. You are on the master branch and it has been modified
3. One file change has been modified
4. The other file change is uncommitted
5. Actions are available to pull/push
6. You can commit these changes

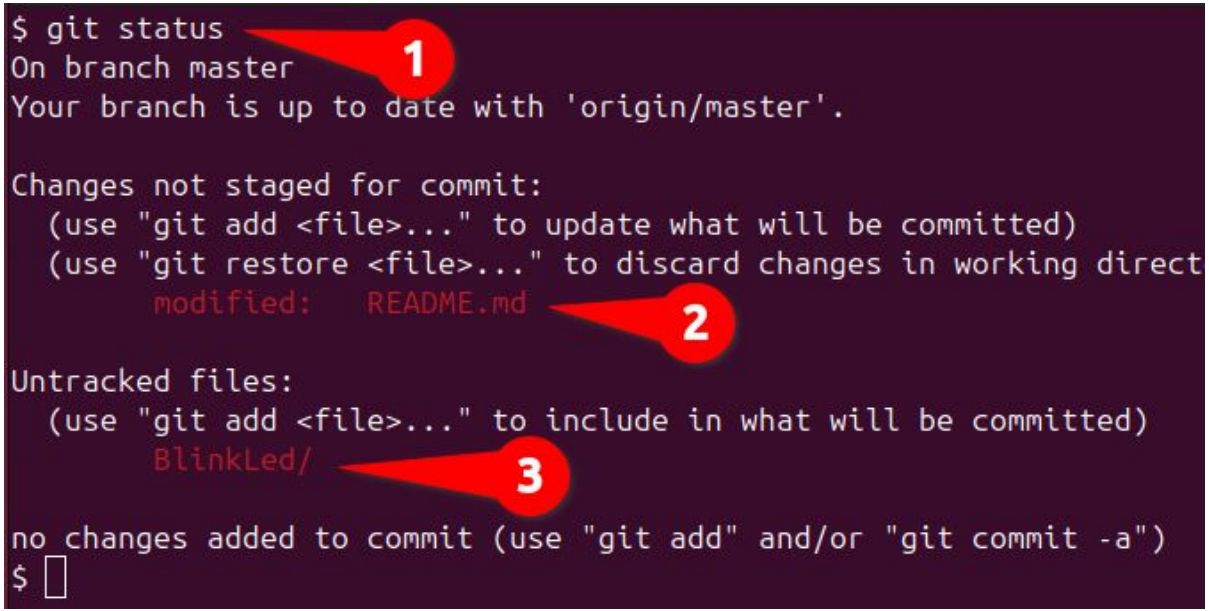


Disclaimer: I don't use vs-code for git, and have no experience with it. I prefer command line + gitk

Inspecting changes - CLI

In the command window:

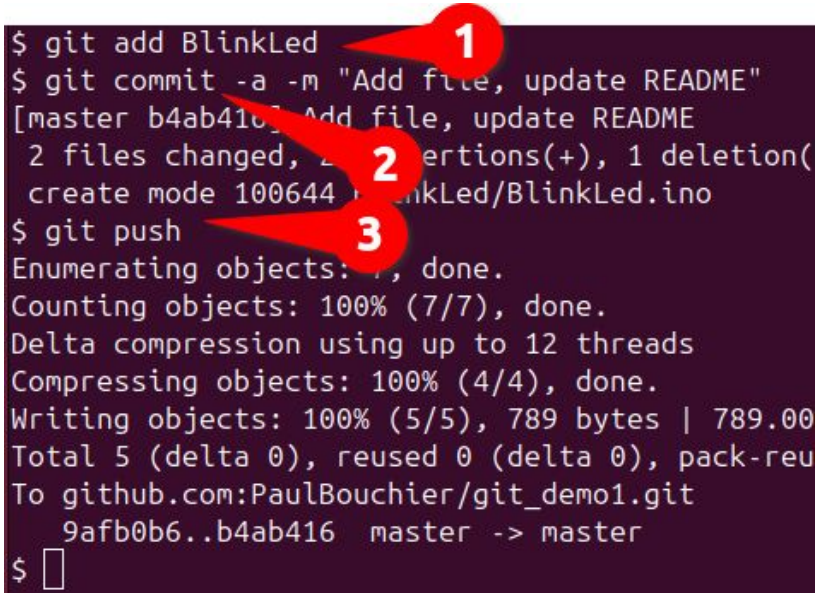
1. Type “git status”
2. Observe 1 file changed, not staged for commit, with instructions for unstaging
3. Observe an untracked folder (which contains our new file) with instructions for adding

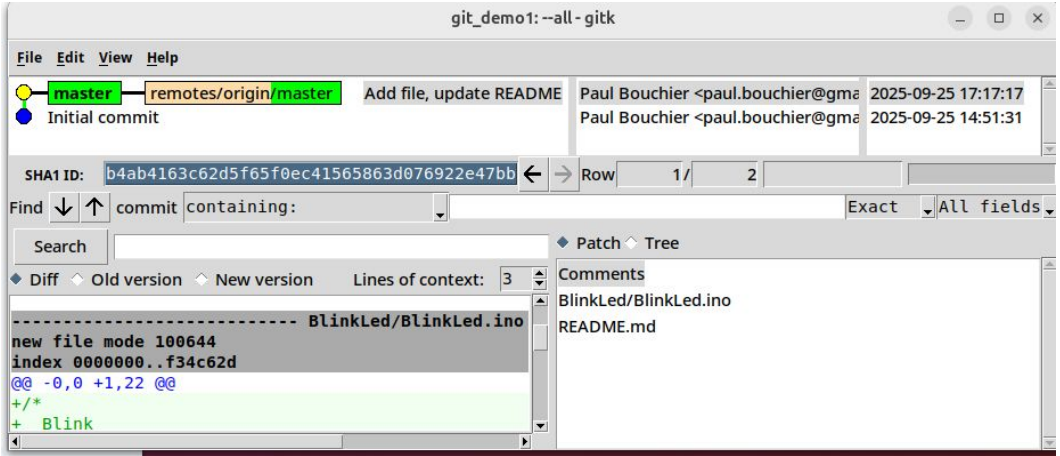


Stage, commit & push

In the CLI, type:

1. git add <new files/folders>. Update gitk and check: File ↠ Reload
2. git commit -a -m “<comment>”. Update gitk and check
3. Git push. Update gitk and check





References

Repo with this talk's slides: https://github.com/dprg/2025_git_talk

Show’n’tell