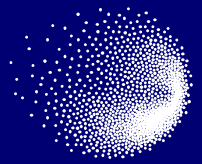


PSI Center for Scientific Computing, Theory and Data

Introduction to GIT

Understanding your timeline

Basil Bruhn
12/05/2026

Agenda

- Looking back in time
- Configuring your workspace
- Understanding your repository
- Daily GIT workflows
- Working with remotes
- Branches and collaboration
- Merge conflicts
- Taking a glance into the future

Looking back in time

- History of changes
  - HEAD and commits
- Local and Remote
  - GIT and Gitea / GitHub
- Authentication with SSH / HTTPS
  - Creation of SSH key / PAT
- Creating a Repo and pushing
  - init
  - add
  - status
  - commit
  - push

Configuring your workspace

PSI

- Identity
  - `git config --global user.name`
  - `git config --global user.email`

- Default editor
  - `git config --global core.editor vim`

- Default branch
  - `git config --global init.defaultBranch main`

- Credential helper

Configuring your workspace

PSI

- ~/.gitignore
  - `*.log`
    `.env`
    `__pycache__/`
    `.idea/`

- GIT tracks changes.

  `.gitignore` tells Git which files should not be tracked

Understanding the Repository

A normal folder versus a Git repository

```
➜  ~ mkdir research_project

➜  ~ cd research_project

➜  research_project touch notes.txt

➜  research_project ls -la
drwxr-xr-x@   3 bruhn_b  staff    96 May 11 12:42 .
drwxr-x---+ 108 bruhn_b  staff  3456 May 11 12:42 ..
-rw-r--r--@   1 bruhn_b  staff     0 May 11 12:42 notes.txt
```

Understanding the Repository

A normal folder versus a Git repository

```
➜  research_project git init
Initialized empty Git repository in /Users/bruhn_b/research_project/.git/

➜  research_project git:(main) ✗ ls -la
total 0
drwxr-xr-x@   4 bruhn_b  staff   128 May 11 12:43 .
drwxr-x---+ 108 bruhn_b  staff  3456 May 11 12:43 ..
drwxr-xr-x@   9 bruhn_b  staff   288 May 11 12:43 .git
-rw-r--r--@   1 bruhn_b  staff     0 May 11 12:42 notes.txt
```

Understanding the Repository
What is inside `.git`?

```
➜  research_project git:(main) ✗ ls -la .git
total 24
drwxr-xr-x@  9 bruhn_b  staff  288 May 11 12:43 .
drwxr-xr-x@  4 bruhn_b  staff  128 May 11 12:43 ..
-rw-r--r--@  1 bruhn_b  staff  137 May 11 12:43 config
-rw-r--r--@  1 bruhn_b  staff   73 May 11 12:43 description
-rw-r--r--@  1 bruhn_b  staff   21 May 11 12:43 HEAD
drwxr-xr-x@ 16 bruhn_b  staff  512 May 11 12:43 hooks
drwxr-xr-x@  3 bruhn_b  staff   96 May 11 12:43 info
drwxr-xr-x@  4 bruhn_b  staff  128 May 11 12:43 objects
drwxr-xr-x@  4 bruhn_b  staff  128 May 11 12:43 refs
```

Understanding the Repository
create

```
➜  research_project git:(main) ✗ echo "experiment 1" > notes.txt

➜  research_project git:(main) ✗ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        notes.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Understanding the Repository
stage

```
➜  research_project git:(main) ✗ git add notes.txt

➜  research_project git:(main) ✗ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   notes.txt
```

Understanding the Repository
commit

```
➜  research_project git:(main) ✗ git commit -m "add experiment notes"
[main (root-commit) 842a938] add experiment notes
 1 file changed, 1 insertion(+)
 create mode 100644 notes.txt

➜  research_project git:(main) git log –oneline

842a938 add experiment notes
```

Understanding the Repository
Why staging exists

```
➜  research_project git:(main) echo "important result" >> notes.txt

➜  research_project git:(main) ✗ echo "temporary debug" > debug.log

➜  research_project git:(main) ✗ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   notes.txt

no changes added to commit (use "git add" and/or "git commit -a")
➜  research_project git:(main) ✗ git add notes.txt
```

Daily Git workflows
the golden loop

change -> review -> stage -> commit -> push

- `git status`
- `git diff`
- `git add`
- `git commit -m "message"`
- `git push`

Daily Git workflows
review before committing

```
➜  research_project git:(main) ✗ echo "new experiment" >> notes.txt

➜  research_project git:(main) ✗ git diff
diff --git a/notes.txt b/notes.txt
index 107b070..4a49b71 100644
--- a/notes.txt
+++ b/notes.txt
@@ -1,2 +1,3 @@
 experiment 1
 important result
+new experiment
(END)
```

Daily Git workflows
selective staging

```
➜  research_project git:(main) ✗ touch debug_script.py

➜  research_project git:(main) ✗ git add notes.txt

➜  research_project git:(main) ✗ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   notes.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        debug_script.py
```

Daily Git workflows
selective staging

Undo a change in a file

```
git restore notes.txt
```

Undo all unstaged local files (panic button)

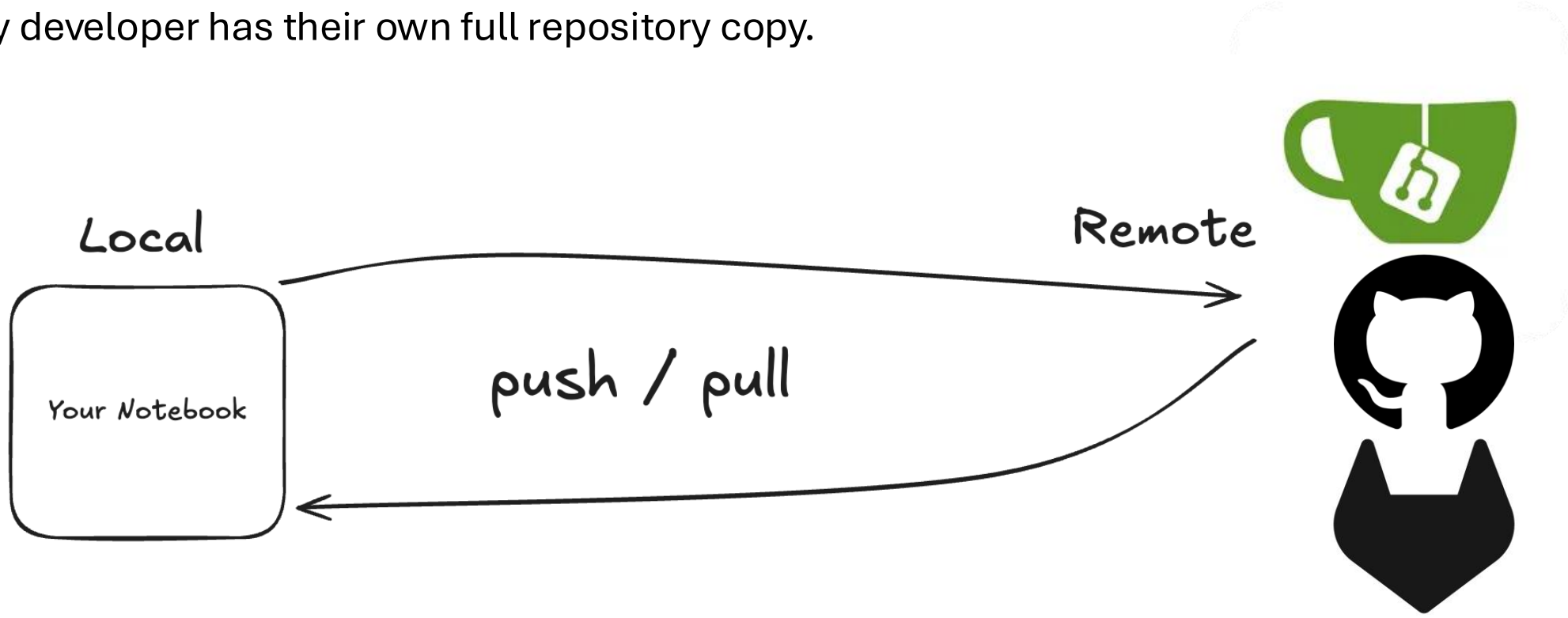
```
git restore .
```

Remove a file from staging

```
git restore --staged notes.txt
```

Working with remotes
Git is decentralized

Every developer has their own full repository copy.



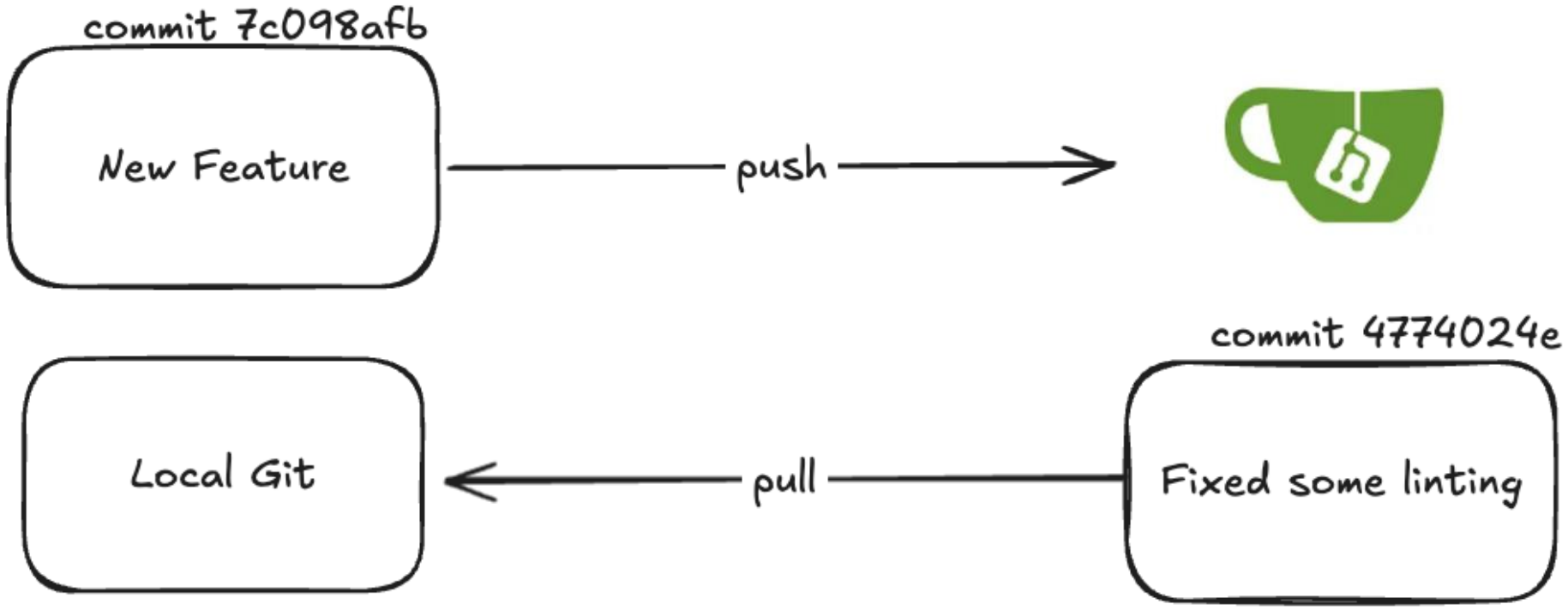
Working with remotes
origin?

```
➜  research_project git:(main) ✗ git remote -v

➜  research_project git:(main) ✗ git remote add origin
git@gitea.psi.ch:bruhn_b/research_project.git

➜  research_project git:(main) ✗ git remote -v
origin git@gitea.psi.ch:bruhn_b/research_project.git (fetch)
origin git@gitea.psi.ch:bruhn_b/research_project.git (push)
```

Working with remotes
push vs pull

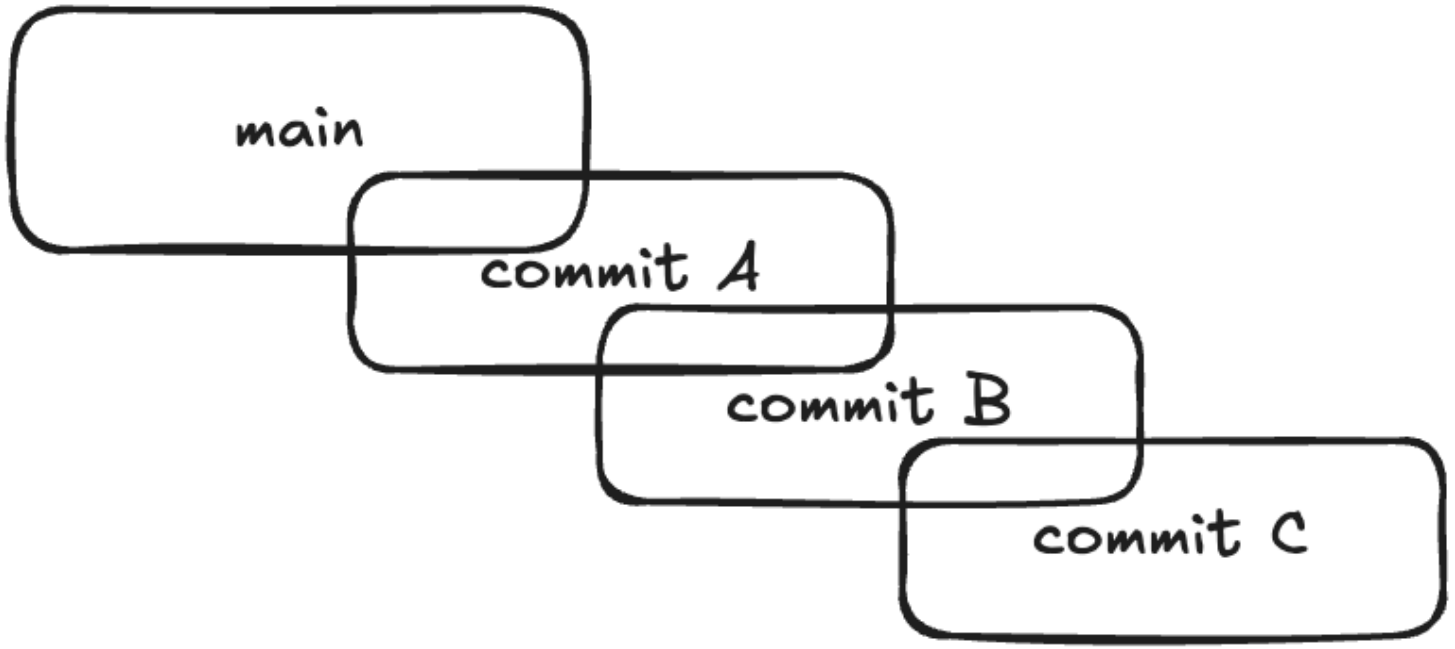


Branches and Collaboration

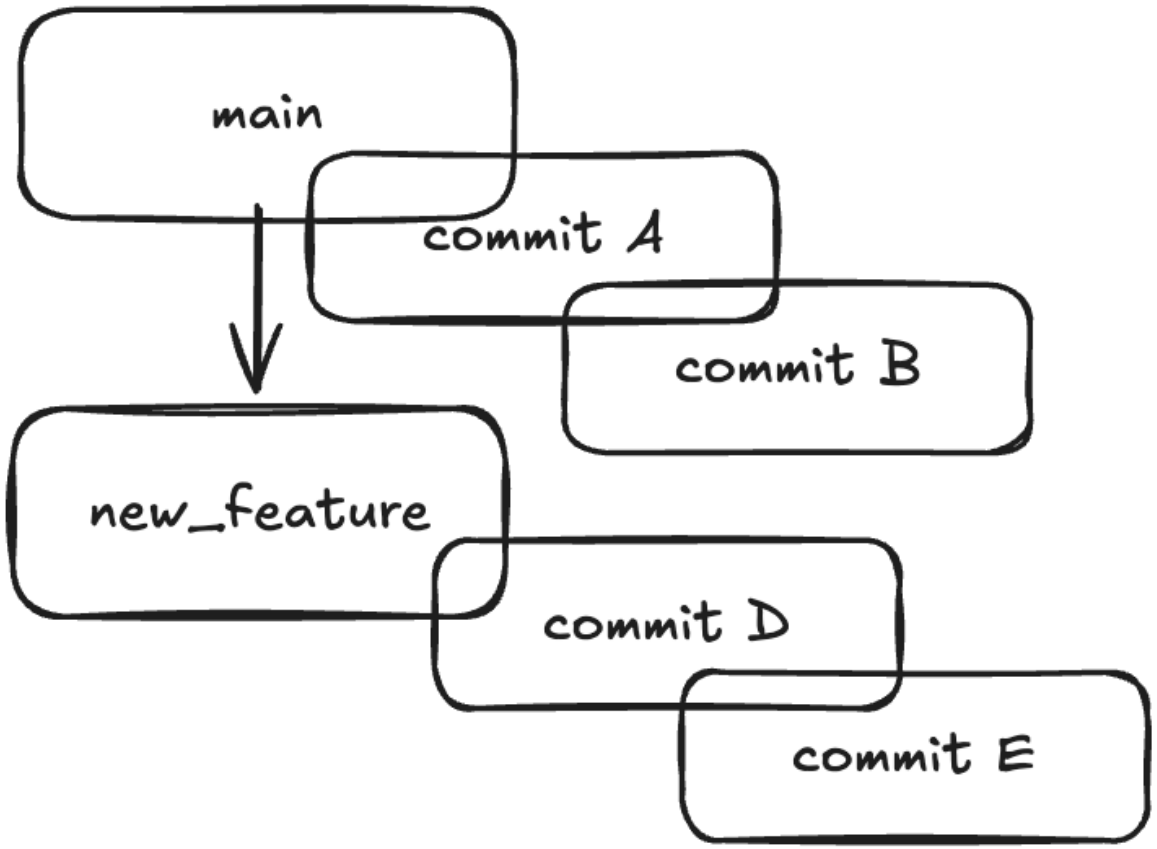
Real life Scenario

- Multiple people push
- Everyone pulls
- History stays synchronized

Branches and Collaboration



Branches and Collaboration



Branches and Collaboration

```
➜  research_project git:(main) ✗ git switch -c feature-x

➜  research_project git:(feature-x) ✗ echo "new feature" >> notes.txt

➜  research_project git:(feature-x) ✗ git add notes.txt

➜  research_project git:(feature-x) ✗ git commit -m "add feature"
[feature-x a28a03a] add feature
 1 file changed, 1 insertion(+)
```

Branches and Collaboration

```
➜  research_project git:(feature-x) git switch main
Switched to branch 'main’

➜  research_project git:(main) cat notes.txt
experiment 1
important result
```

Branches and Collaboration

```
➜  research_project git:(main) git merge feature-x
Updating 842a938..a28a03a
Fast-forward
notes.txt       | 1 +
 1 file changed, 1 insertion(+)

➜  research_project git:(main) cat notes.txt
experiment 1
important result
new feature
```

Merge conflicts

```
➜  research_project git:(main) git switch -c feature_a
Switched to a new branch 'feature_a’

➜  research_project git:(feature_a) echo "hello universe" > code.txt

➜  research_project git:(feature_a) ✗ git add code.txt

➜  research_project git:(feature_a) ✗ git commit -m ”new feature"
[feature_a f7eb4e4] new feature
 1 file changed, 1 insertion(+)
 create mode 100644 code.txt
```

Merge conflicts

```
➜  research_project git:(main) ✗ git switch -c feature_b
Switched to a new branch 'feature_b’

➜  research_project git:(feature_b) echo "hello world" > code.txt

➜  research_project git:(feature_b) ✗ git add code.txt

➜  research_project git:(feature_b) ✗ git commit -m ”another feature”
[feature_b 2c89e14] another feature
 1 file changed, 1 insertion(+)
 create mode 100644 code.txt
```

Merge conflicts

```
➜  research_project git:(main) git merge feature_a
Updating a28a03a..f7eb4e4
Fast-forward
 code.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 code.txt

➜  research_project git:(main) git merge feature_b
Auto-merging code.txt
CONFLICT (add/add): Merge conflict in code.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflicts

```
➜  research_project git:(main) ✗ nvim code.txt

<<<<<<< HEAD
hello universe
=======
hello world
>>>>>>> feature_b

➜  research_project git:(main) ✗ git add code.txt

➜  research_project git:(main) git commit -m "using more refined feature"
[main db35705] using more refined feature
```

Taking a glance into the future

- rebasing
- cherry-picking
- recovering lost commits
- Stashing / Worktree
- fixing history
- Fetch forked repos / bring branches from forks to our own repo
- git reset
- Record the course

Cheat Sheet

- git init
- git status
- git add .
- git commit -m "message"
- git log
- git remote add origin <url>
- git push
- git pull
- git remote –v
- git restore file.txt
- git restore --staged file.txt
- git diff
- git switch –c branch
- git merge branch
- git branch

Questions!