git version control revisited

Work smarter not harder.

Joe Hamman

Hydro-group Computing Seminar

February, 25 2014

Best Practices for Scientific Computing

D.A. Aruliah [*], C. Titus Brown [†], Neil P. Chue Hong [‡], Matt Davis [§], Richard T. Guy [¶], Steven H.D. Haddock [‖], Katy Huff [**], Ian M. Mitchell [††], Mark D. Plumbley [‡‡], Ben Waugh [§§], Ethan P. White [¶¶], Greg Wilson [***], Paul Wilson [†††]

1. Write programs for people, not computers.
2. Automate repetitive tasks.
3. Use the computer to record history.
4. Make incremental changes.
5. Use version control.
6. Don’t repeat yourself (or others).
7. Plan for mistakes.
8. Optimize software only after it works correctly.
9. Document design and purpose, not mechanics.
10. Conduct code reviews.

Today…

1. Example from one of my projects.
2. Quick refresher on the git basics
3. Tutorial on using github (or any other “remotes”)
4. Using git for your projects

Notes:

- I don’t have a lot planned so I would like this to be a discussion on how best to use git as a tool for your research.

Using Git to record history

Recent Example from RVIC

Version Control Systems

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Git is an **open source**, distributed version control system designed for speed and efficiency

Git is an open source,
distributed version control
system designed for speed
and efficiency

(almost) everything is local

No Network Needed

Performing a diff

Viewing file history

Committing changes

Merging branches

Obtaining any other revision of a file

Switching branches

which means

everything is fast

every clone is a backup

work offline

Git is an open source, distributed version control system **designed for speed and efficiency**

Immutable

(almost) never removes data

Snapshots, not Patches

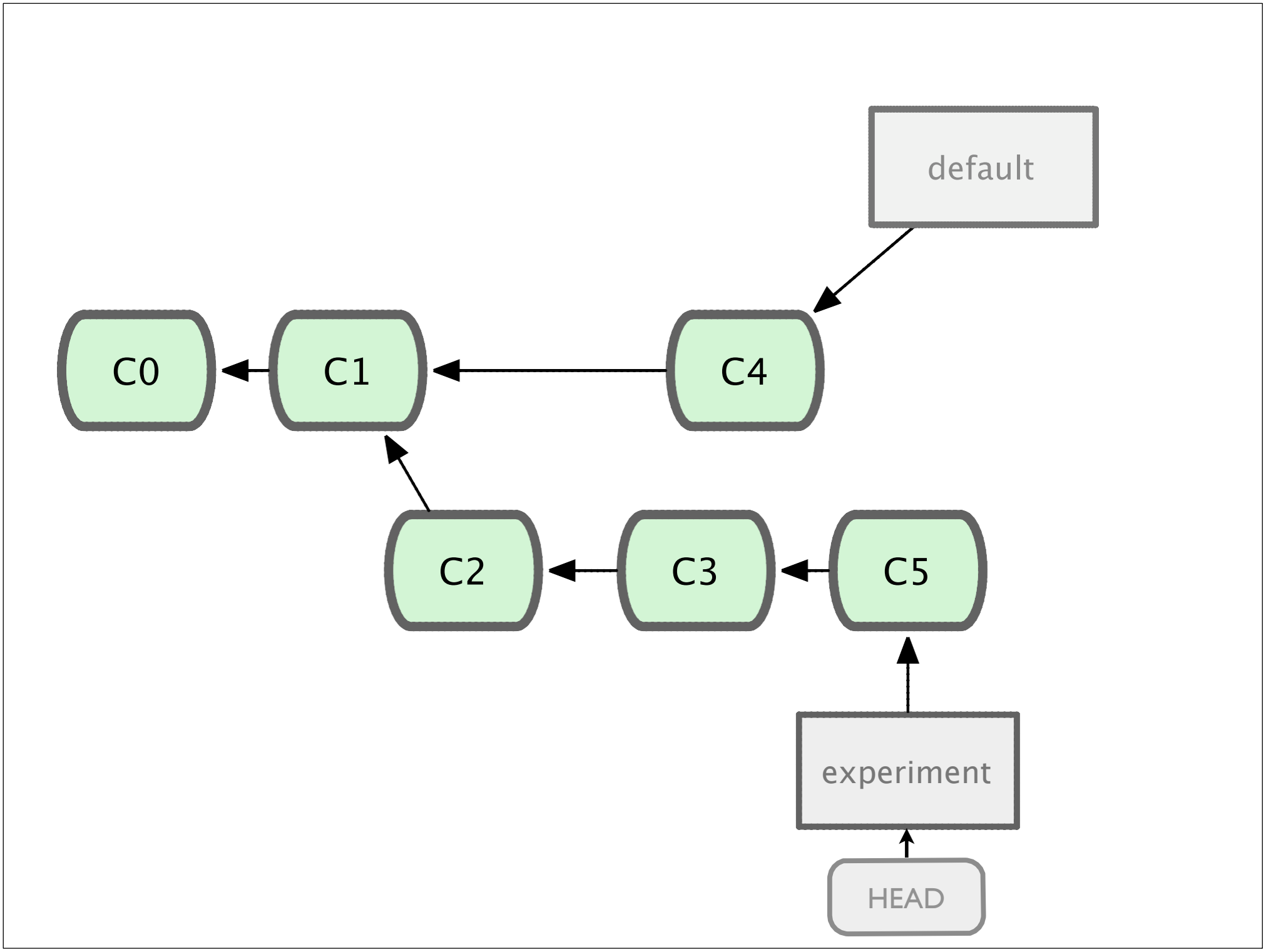
A Basic Workflow

Edit files	vim / emacs / etc
Stage the changes	git add (file)
Review your changes	git status
Commit the changes	git commit
Push your changes	git push

Branching and Merging

git merge

apply edits on one branch to another branch


default
C0
C1
C4
C2
C3
C5
experiment
HEAD

Remotes

Distributed Workflow
&
Parallel development

try out an idea

isolate work units

long running topics

pain free
context switching

Example

VIC Glacier Work

Collaboration and Issue Tracking

VIC Repo Example

- Manage separate branches from multiple developers
- Follow and discuss issues, bugs, and new features
- Documentation (history and wiki)