Version Control with Git

Michael S. Bryant | HPC @ LSU

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Overview

- Version control is the process of **maintaining file revisions**
- Examples are:
  - Saving a backup copy: **file.txt** saved as file-**old**.txt
  - Adding a version number or date: file-**v1**.txt or file-**2014-11-12**.txt
  - Using **version control software** such Git, Mercurial, Subversion, etc
- The advantage of version control software is that it keeps track of the **complete file history** (why/what/who/when changed), allows for **shared development**, and supports **complex workflows** (multiple branches).

Git

- Git is a free and open source **distributed version control system**
  - It's **very fast** since most operations are local (no server communication)
  - Any local directory can be a Git repository which means no server to set up
  - Each copy of the repository serves as a complete backup (distributed nature)
  - Website: http://git-scm.com/
- It's not only for source code management:
  - Managing Linux/Mac home directory configuration files (search GitHub for dotfiles)
  - It can be used to edit source code local and push changes to an HPC cluster instead of editing within a command-line interface
    - Git hooks can be used to automatically update cluster copy on push
- Excellent documentation: http://git-scm.com/doc
- Git hosting: GitHub, BitBucket, etc; Do-it-yourself hosting: Gitolite
- Plenty of GUI clients available: http://git-scm.com/downloads/guis
  - Git-cola/Gitg are good for showing a visual commit tree and history

Common Git Commands

- **Init**: create new repository
- **Clone**: downloads a copy of a repository (SVN: checkout)
- **Add**: adds a new file to the working copy
- **Status, log, diff**
- **Push/pull**: uploads/downloads repository changes
- **Commit**: save changes into repository
- Cheat sheet:
  - https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

Configuring Git

- Identify yourself so commit messages are correct:

```
% git config --global user.name "Your Name"
% git config --global user.email user@lsu.edu
```

- Other settings for ~/.gitconfig (adjust as needed):

```
[user]
    name = Your Name
    email = user@lsu.edu
[core]
    editor = vim
    pager = less
[diff]
    renames = true
[alias]
    st = status --short --branch
    di = diff
    di-staged = diff --staged
```

Commit Message Standard

- Use present tense (i.e. Add, not Added)
- Put the commit summary on the first line (subject) and, if needed, add a detailed message after a blank line (body)

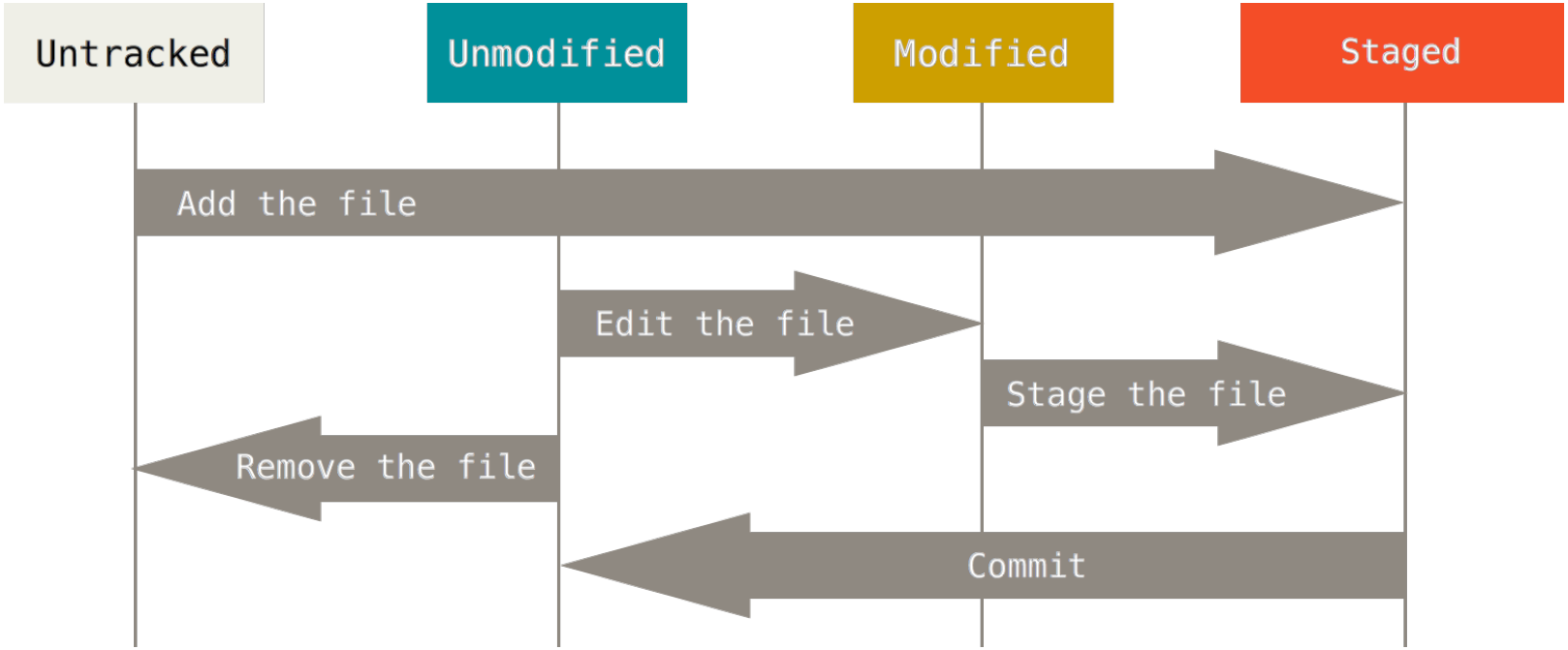
```
Short (50 chars or less) summary of changes

More detailed explanatory text, if necessary.  Wrap it to
about 72 characters or so.  In some contexts, the first
line is treated as the subject of an email and the rest of
the text as the body.  The blank line separating the
summary from the body is critical (unless you omit the body
entirely); tools like rebase can get confused if you run
the two together.

Further paragraphs come after blank lines.

  - Bullet points are okay, too

  - Typically a hyphen or asterisk is used for the bullet,
    preceded by a single space, with blank lines in
    between, but conventions vary here
```

Create a Git Repository

- Run `git init` to create a local repository in any new or existing directory:

```
% mkdir project1; cd project1
% git init
Initialized empty Git repository in /home/user/
project1/.git
% git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git
add" to track)
```

Git Operations



Source: http://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Add a file to project1

```
% echo test > f1
% git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       f1
nothing added to commit but untracked files present (use "git add" to track)
% git add f1  # Stage the change
```

Commit the new file

```
% git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#     new file:   f1
#
% git commit -m "Add f1"
[master (root-commit) a4c3bfb] Add f1
 1 file changed, 1 insertion(+)
 create mode 100644 f1
```

Modify the file and check changes

```
% echo test1 > f1
% git diff
[...]
-test
+test1
% git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
directory)
#
#       modified:   f1
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Add another file before committing

```
% echo test2 > f2
% git add f2
% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   f2
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
directory)
#
#       modified:   f1
#
% git commit -a -m "Include number in f1”  # Didn’t add f1, used -a
```

Show the commit log (history)

```
% git log
commit 9a450643c53074b0df45099e48788a3e4677ac4a
Author: Your Name <user@lsu.edu>
Date:   Wed Nov 12 09:32:02 2014 -0600

    Include number in f1

commit fa61c4a26638088987a83561ac738e24b5608dd0
Author: Your Name <user@lsu.edu>
Date:   Wed Nov 12 09:30:29 2014 -0600

    Add f1
% git log f2  # Why does f2's log only mention f1?
commit 9a450643c53074b0df45099e48788a3e4677ac4a
Author: Your Name <user@lsu.edu>
Date:   Wed Nov 12 09:32:02 2014 -0600

    Include number in f1
```

Check staged changes and unstage

```
% echo test11 > f1; echo test22 > f2
% git add .
% git diff f1  # Must use --staged here
% git diff --staged f1
-test1
+test11
% git status --short
M  f1
M  f2
% git reset HEAD f2  # Unstage f2
```

Commit individual files

```
% git status --short
M  f1
 M f2
% git commit -m "Two numbers in f1"
% git add f2
% git commit -m "Two numbers in f2"
% git log --oneline f2
```

Set up a bare repo for sharing

```
% mkdir project2.git
% cd project2.git
% git init --bare; cd ..
% git clone project2.git p2a
% cd p2a; echo test > f1
% git add f1
% git commit -m "Add f1"
% git push origin master
# Always stay updated, esp. before pushing
% git pull
```

Version Control with Git

Michael S. Bryant | HPC @ LSU

LSU CENTER FOR COMPUTATION & TECHNOLOGY