Last modified April 2019

Introduction to The Git Version Control System

Ziv Yaniv, Ph.D.

GitHub CheatSheet: https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
Git reference: https://git-scm.com/docs
"Pro Git", Chacon and Straub: https://git-scm.com/book/en/v2

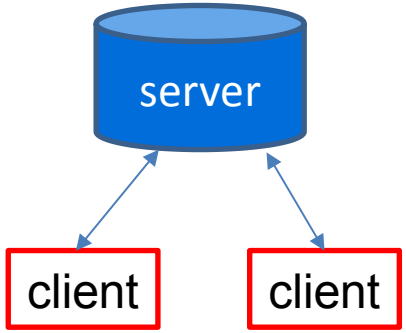
NIH National Institute of Allergy and Infectious Diseases

MSC medical science & computing

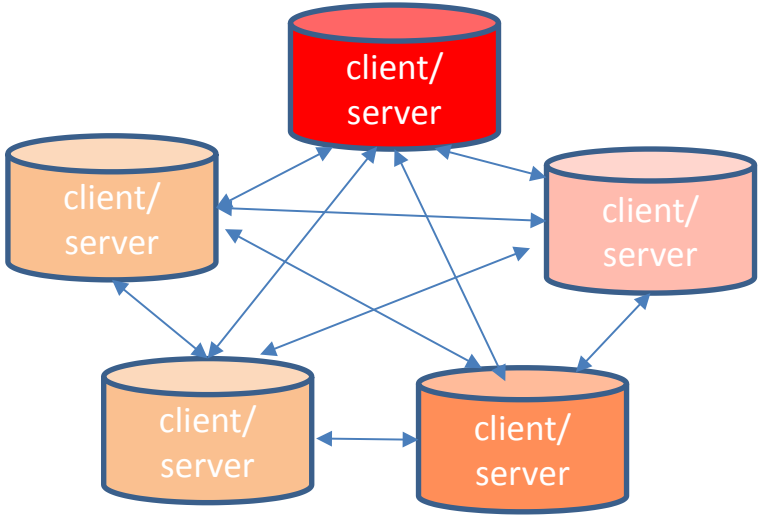
Version Control System

"A system that records changes to a file or set of files over time so that you can recall specific versions later."*

centralized (e.g. svn):



distributed (e.g. mercurial):

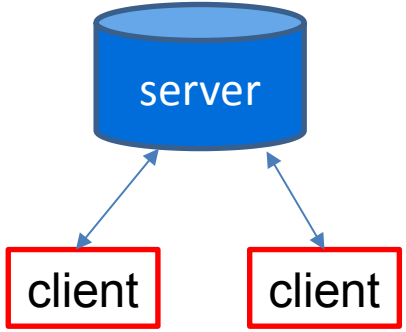


*Pro Git

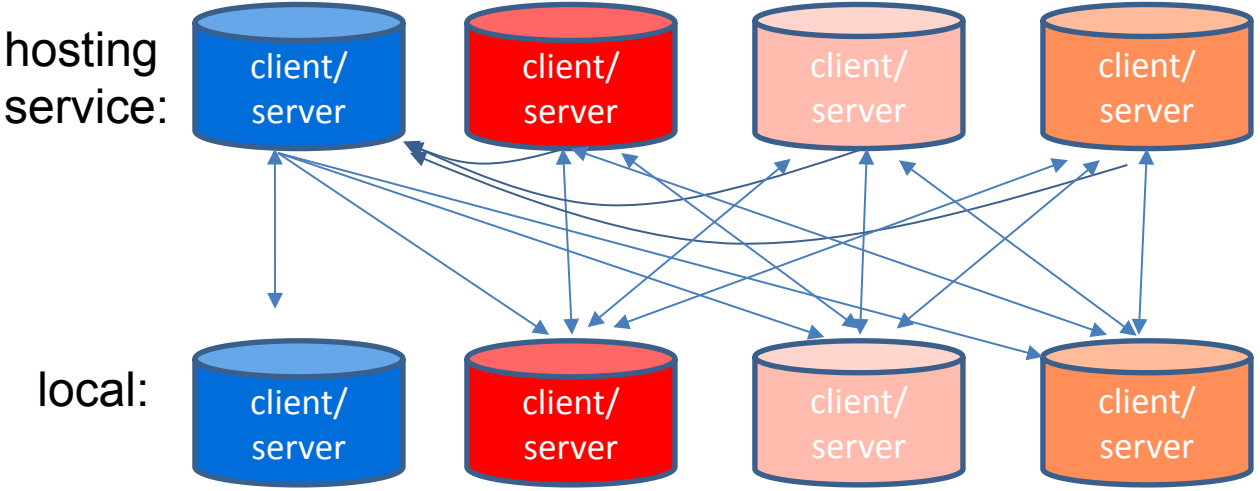
Version Control System

"A system that records changes to a file or set of files over time so that you can recall specific versions later."*

centralized (e.g. svn):



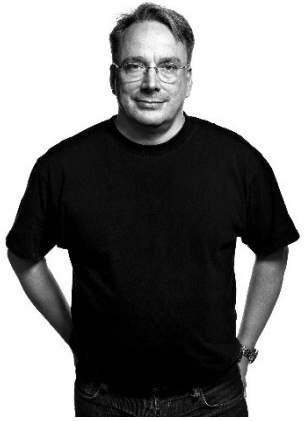
distributed with forking workflow (e.g. git):



*Pro Git

Git

- “A distributed version-control system for tracking changes in source code during software development” [wikipedia].
- Free Open Source Software (FOSS) under GNU General Public License version 2.



Linus Torvalds – Git Creator

Junio Hamano – Git Maintainer

And 1200+ contributors

“Faces of Open Source” project, Peter Adams http://www.facesofopensource.com/

Installation and Interaction

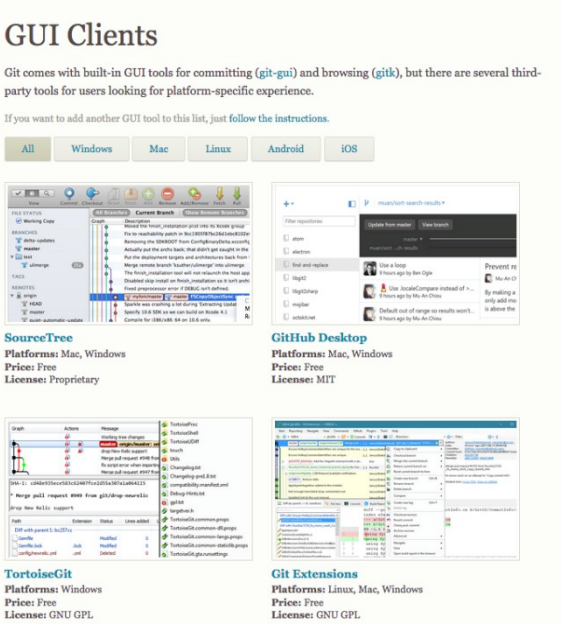
1. Install Git (https://git-scm.com/downloads)

2. Use git:
   i. Command line tool (old school)

```
[(master) ~/toolkits/simpleITK/SimpleITK-Notebooks]
```

   ii. Dedicated GUI front end

   iii. Part of your IDE (Rstudio, Visual Studio…)



https://git-scm.com/downloads/guis/

General Setup (One Time)

- Hierarchical configuration (each layer overrides the preceding one):
  1. All users on the system: /usr/local/git/etc/gitconfig
  2. All of your repositories: ~/.gitconfig
  3. A specific repository: my_repository_directory/.git/config

```
git config --list --show-origin

git config --global user.name "Ziv Yaniv"
git config --global user.email "zivyaniv@nih.gov"
git config --global core.editor emacs
```

List of configurable settings: https://git-scm.com/docs/git-config

General Setup (One Time)

- If you work on the command line, configure your prompt to display the branch (in your .bashrc on unix):

```
source /usr/share/git-core/contrib/completion/git-prompt.sh
PS1='[\[\[\e[1;32m\]$(__git_ps1 "(%s)")\[\e[m\] \[\e[0;31m\]\w\[\e[m\]]'
```

To configure prompt on your specific system “google”:

display git branch in prompt your_operating_system

Words of Caution

- Do not commit large binary files into a git repository, use Git Large File Storage (Git-LFS).
- NEVER commit files that contain critical/private information (passwords, AWS access keys).
- You will make mistakes – fix them locally before you share with the world (push to shared repository).

Starting - One Time Setup

- Create a new repository:

```
git init
```

- Create a local copy of an existing repository:

```
git clone https://github.com/zivy/SimpleITK-Notebooks.git
git clone git@github.com:zivy/SimpleITK-Notebooks.git
```

https or ssh:

firewall issues may dictate which one you end up using.

https requires username-password (config-don't-type, either store or cache):

```
git config credential.helper store
```

ssh setup:

instructions from bitbucket: https://confluence.atlassian.com/bitbucket/set-up-an-ssh-key-728138079.html

instructions from github:
https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent#generating-a-new-ssh-key

Starting – Looking Around

`git status` : show working tree status, used all the time.

`git log` : show all commits (and merges).

`git remote -v` : show all remote repositories, used once in a while.

`git branch -a -vv` : show all branches, used once in a while.

When using a GUI, most often, this information is readily visible.

Starting – Git is all about branches

```
[(master) ~/toolkits/simpleITK/src]git branch -a -vv
* master                                    8243c5f2 [origin/master] Merge topic 'WrapCannySegmentationLevelSetImageFilter'
  updateTutorials                           4ab0554c Adding link to EMBC tutorial/workshop.
  remotes/origin/HEAD                       -> origin/master
  remotes/origin/dashboard                  310ec654 Updating common simpleitk test driver with upstream ITK changes.
  remotes/origin/master                     8243c5f2 Merge topic 'WrapCannySegmentationLevelSetImageFilter'
  remotes/origin/next                       42ad84b9 Merge branch 'master' into next
  remotes/origin/release                    925ab7e6 Merge branch 'FixDoxygenCommandsAndEventSnippets' into release
  remotes/origin/rtdUpdateTutorials         0747b5c8 Moved SPIE course from upcoming to past section.
  remotes/origin/updateRInstallInstructions 81f2231e DOC: updating documenation for R installation.
  remotes/origin/updateReadme               1ee8a03a Upadated readme file to match our current state (rc1.2).
  remotes/origin/updateTutorials            4ab0554c Adding link to EMBC tutorial/workshop.
  remotes/upstream/dashboard                3745ac4f Exclude unstable tests from CI
  remotes/upstream/master                   8243c5f2 Merge topic 'WrapCannySegmentationLevelSetImageFilter'
  remotes/upstream/next                     6416fe25 Merge topic 'AddAzureBadges' into next
  remotes/upstream/release                  7c84168f Merge branch 'FixRManPath' into release
```

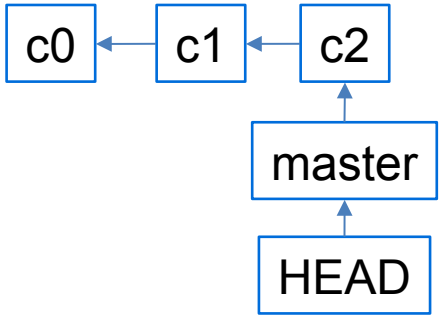
Branch types:

1. Local, non tracking: local repo branch (e.g. updateTutorials)
2. Local, tracking: local repo branch tracking another branch (e.g. master)
3. Remote branch: remote repo branch
4. Remote tracking branch: local copy of remote repo branch (remotes/origin/updateReadme)

Adding File(s) Into The Current Branch

- A two step process?, repeat till done:
  1. Stage files for commit: `git add f1.py f2.py…`
  2. Check the status before commiting: `git status`
  3. Commit: `git commit`

A single branch repo:

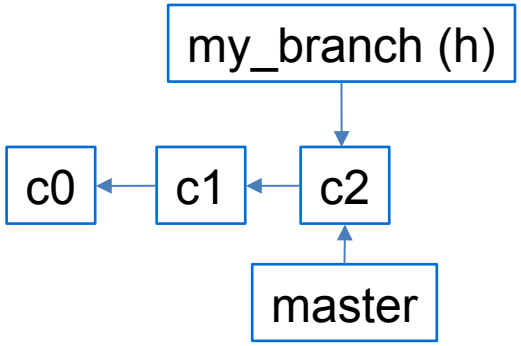


master – default branch name.
HEAD – pointer to last commit on currently checked out branch.

Working with Branches

- Create a new branch and check it out:

```
git checkout -b my_branch
```

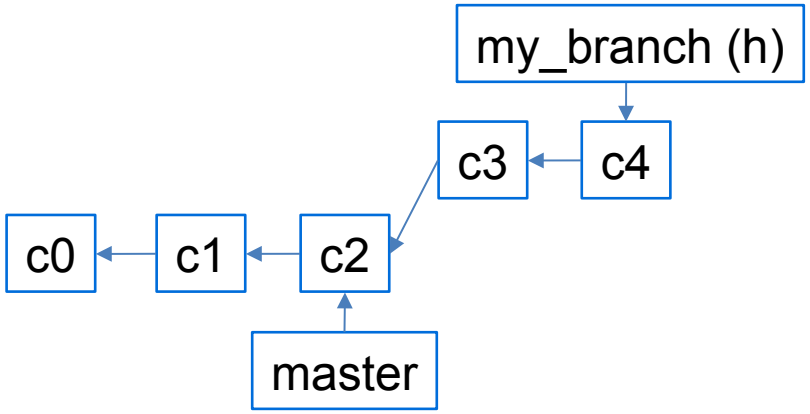


Working with Branches

- Create a new branch and check it out:

```
git checkout -b my_branch
```

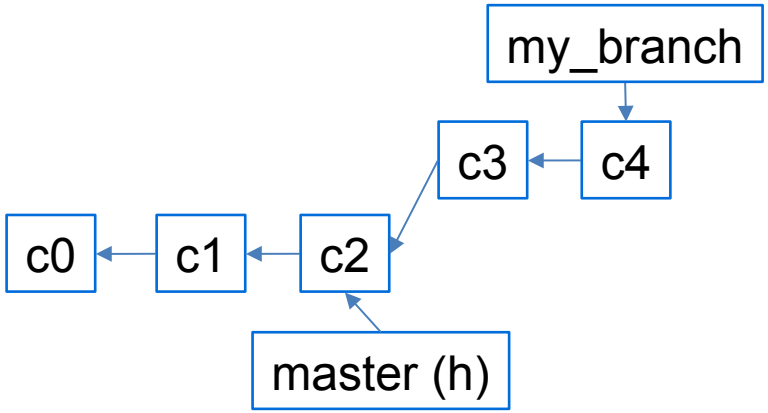
- Add commits.



Working with Branches

- Checkout the master branch:

```
git checkout master
```

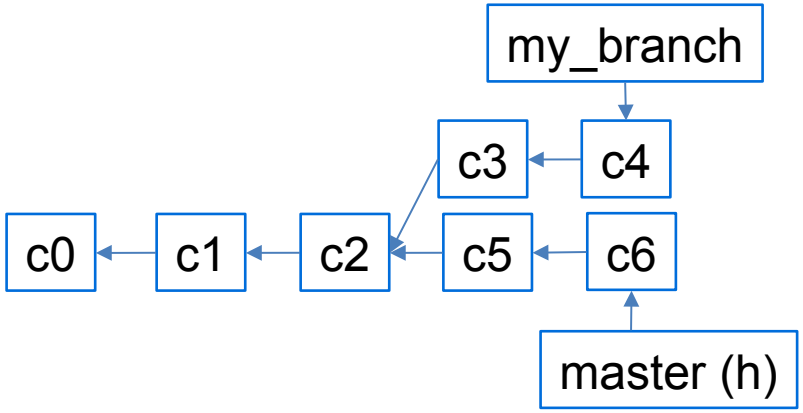


Working with Branches

- Checkout the master branch:

```
git checkout master
```

- Add commits.



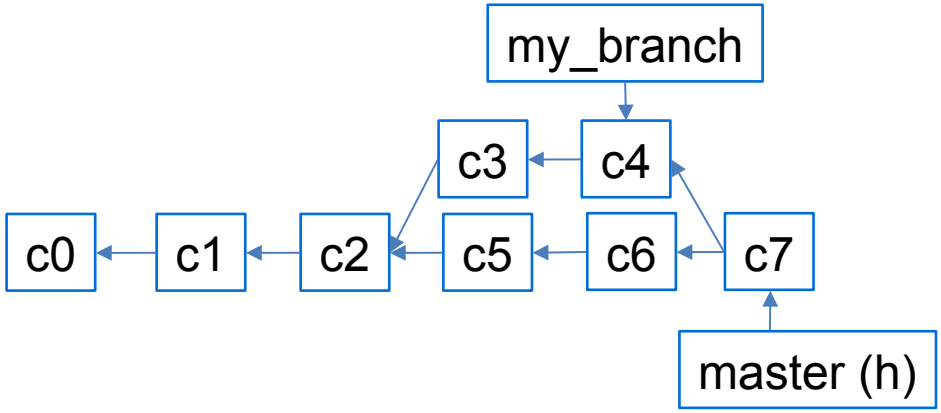
Working with Branches

- Checkout the master branch:

```
git checkout master
```

- Add commits.
- Merge my_branch into master branch (v1):

```
git merge my_branch
```



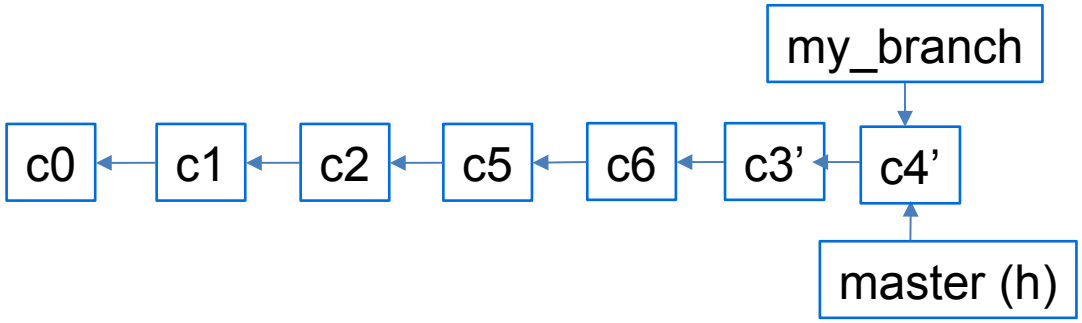
Working with Branches

- Checkout the master branch:

```
git checkout master
```

- Add commits.
- Merge my_branch into master branch (v2):

```
git checkout my_branch
git rebase master
git checkout master
git merge my_branch
```



To Err is Human

Undo staging:

```
git reset file_name
```

Correct an error in the last commit (rewriting history):

```
edit file
git add file_name
git commit --amend
```

Undo commit(s):

```
git reset HEAD^ [--hard to get rid of the changes]
git reset HEAD~n [n commits back]
```

No Repository is an Island

Adding a remote repository:

```
git remote add remote_name remote_url
```

Updating your checked out local tracking branch:

```
git pull
```

equivalent to

```
git fetch  remote_tracking_branch
git merge remote_tracking_branch
```

Updating a remote repository:

```
git push remote_repo_name branch_name
```

`-u`: add tracking reference if you want the current local non tracking branch to track the remote one you are creating.

`-f`: force the push, rewriting history on the remote repo.

Forking Workflow

One time setup:

1. Fork repository of interest on git hosting service (GitHub/Bitbucket/…), let's call it 'repo'.

2. Locally clone:
   ```
   git clone git@github.com:your_username/repo.git
   ```

3. Add the original repository as a remote, usually named 'upstream':
   ```
   git remote add upstream git@github.com:repo_username/repo.git
   ```

4. Check that we have two remote repositories (origin, upstream):
   ```
   git remote -v
   ```

Forking Workflow

1. If not synched with upstream repository, synch before beginning to work:
   ```
   git fetch upstream
   git checkout master
   git merge upstream/master
   ```
2. Create branch on which you will work:
   ```
   git checkout -b my-branch-name
   ```
3. Write code+tests and commit locally:
   ```
   git add my_file
   git commit my_file
   ```
4. Push branch to your repository on hosting service: `git push origin my-branch-name`
5. On the hosting service site, open a pull request from my-branch-name to the upstream repository (possibly triggering CI testing).
6. Owner of the upstream repository will do a code review.
7. Repeat steps 3,4,6 until the branch is merged into the upstream repository.
8. Delete the branch on your remote and local repositories:
   ```
   git push origin --delete my-branch-name
   git branch -d my-branch-name
   ```

Thank You

Open Source Software: don’t just use it, contribute to it:

https://github.com/SimpleITK