Introduction to Git and GitHub

by: Alvar Almstedt

What is Git?

- Version Control System (VCS)
- Backups
- Branching
- Mutiple users on the same project
- Good error messages
- Free & Open Source
- Developed by Linus Torvalds

What is GitHub?

- One of many Git repository hosting services (others: Bitbucket, GitLab, Codebase etc.)

- Public repositories: Free
- Private repositories: Paid

- Hosts many popular bioinformatics software: samtools, BioPython, PacBio software etc.

- Nice and easy to understand interface

Creating a repository

Locally:

- Creata a new dir and move into it
- `git init`
- Start working

Note:
A good place to start is to create a readme file for the repo in markdown language.

Remotely:

- `git clone <html>/repository.git`
- add, commit and push something

or

- Just follow the instrictions from GitHub:

```
echo "# testrepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:alvaralmstedt/testrepo.git
git push -u origin master
```

add

- New files are not tracked and thus ignored by git
- `git add <filename>`

If confused, write "`git status`" often

```
Alvars-MBP:testrepo alvaralmstedt$ touch testfile.txt
Alvars-MBP:testrepo alvaralmstedt$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        testfile.txt

nothing added to commit but untracked files present (use "git add" to track)
Alvars-MBP:testrepo alvaralmstedt$
```

git is now aware of `testfile.txt`
This type of add only needs to be done once per file

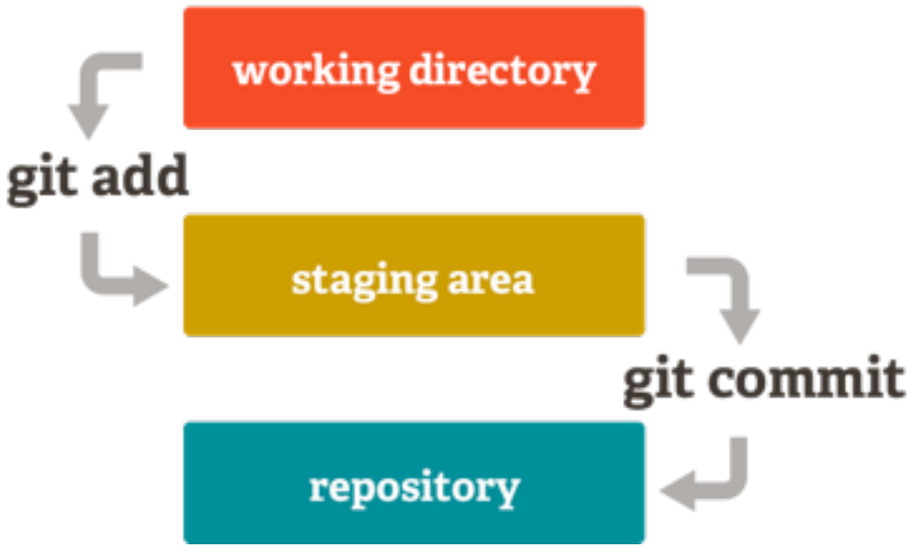
Files get added to staging area/index

```
Alvars-MBP:testrepo alvaralmstedt$ git add testfile.txt
Alvars-MBP:testrepo alvaralmstedt$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   testfile.txt

Alvars-MBP:testrepo alvaralmstedt$
```

commit



https://git-scm.com/

<u>Every time changes to a file are to be committed, they have to be added to the staging area/index</u>

(unless `git commit -a` is used)

To review commit history: `git log`

```
Alvars-MBP:testrepo alvaralmstedt$ git commit -m "added testfile.txt to repository"
[master d3ef6cd] added testfile.txt to repository
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testfile.txt
Alvars-MBP:testrepo alvaralmstedt$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
Alvars-MBP:testrepo alvaralmstedt$
```

SHA-1 short ID

Add a useful commit message if possible :)

xkcd.com

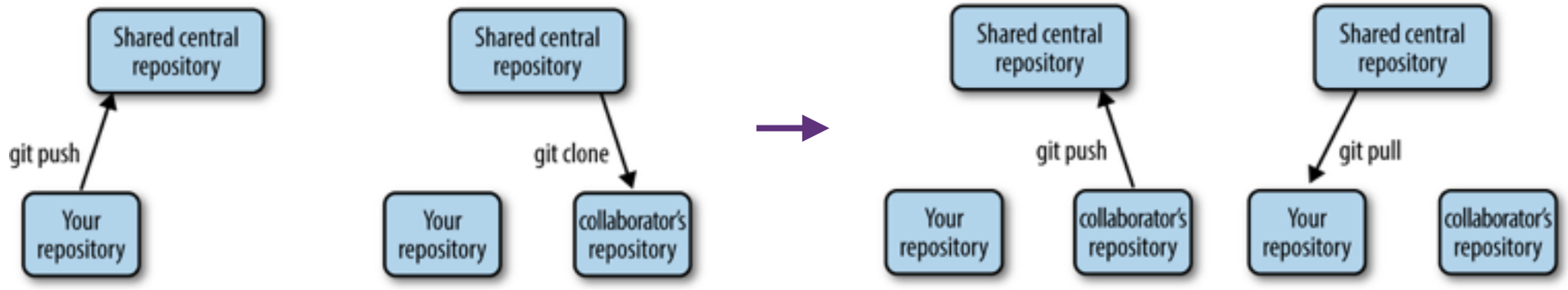
Remotes: push & pull

pull:
"pulls" the remote repo state (GitHub) into your local repository

push:
"pushes" the local repo state onto your remote repository

```
Alvars-MBP:testrepo alvaralmstedt$ git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 228 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:alvaralmstedt/testrepo.git
 * [new branch]      master -> master
Alvars-MBP:testrepo alvaralmstedt$
```

Workflow:
- pull
- code/work
- commit
- push



Images from: [Bioinformatics Data Skills (2015), page 85]

.gitignore

When working with many untracked files, git status can become clogged. Add these files to `.gitignore`

```
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        readfile1.fastq
        readfile2.fastq
        readfile3.fastq

nothing added to commit but untracked files present (use "git add" to track)
Alvars-MBP:testrepo alvaralmstedt$ echo "*.fastq" > .gitignore
Alvars-MBP:testrepo alvaralmstedt$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

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

Large data files and intermediate files should be added to `.gitignore`

add and commit `.gitignore` to get a clean directory

```
Alvars-MBP:testrepo alvaralmstedt$ git add .gitignore
Alvars-MBP:testrepo alvaralmstedt$ git commit -m "added .gitignore"
[master 7e91d4c] added .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
Alvars-MBP:testrepo alvaralmstedt$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
Alvars-MBP:testrepo alvaralmstedt$
```

branch & checkout

branches allow work to be done without messing with stable commits

checkout switches between branches or commits (dual function)

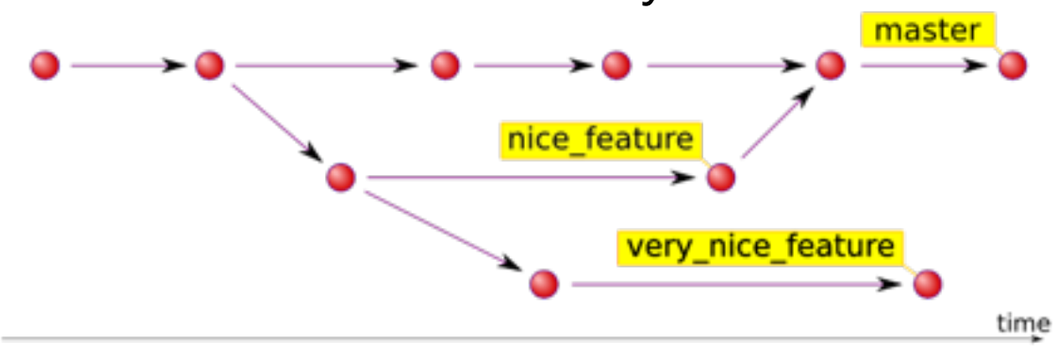
```
Alvars-MBP:testrepo alvaralmstedt$ git branch experimental
Alvars-MBP:testrepo alvaralmstedt$ git branch
  experimental
* master
Alvars-MBP:testrepo alvaralmstedt$ git checkout experimental
Switched to branch 'experimental'
Alvars-MBP:testrepo alvaralmstedt$ git branch
* experimental
  master
Alvars-MBP:testrepo alvaralmstedt$
```

merging allows work from separate branches to be merged together

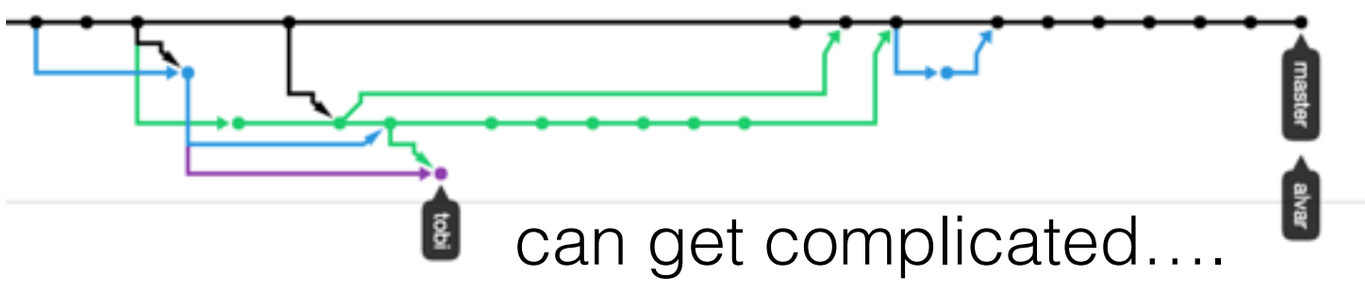
Merge conflict:

```
the number of planets are
<<<<<<< HEAD
nine
=======
eight
>>>>>>> branchname/commitname
```

ideally:



but



can get complicated….

Quick “No-Extras” method

First time:

- Create a remote repository on GitHub.
- `git clone https://github.com/user_name/repo_name.git`
- work
- `git add -A`
- `git commit -a -m “commit message”`
- `git push`

Subsequent times:

- Create a remote repository on GitHub.
- `git pull`
- work
- `git commit -a -m “commit message”`
- `git push`

xkcd.com

other commands (advanced)

`git diff`
lists unstaged file differences

`git mv; git rm`
moves/renames and removes tracked files and stages the changes to be commited.

`git reset`
unstages staged files

`git diff`
lists unstaged file to committed file differences

`git stash`
temporarily stash working changes and restores state to latest commit

`git rebase`
another, more linear way to merge branches. Re-writes history.

`git config <options>`
configure various user-specific settings

A few resources

- Codeacemy Git course: https://www.codecademy.com/learn/learn-git
- TryGit quick tutorial: https://try.github.io
- Pro Git Book: http://git-scm.com/book/en/v2
- Git Reference: gitref.org
- Graphical git branching tutorial: http://pcottle.github.io/learnGitBranching/

Thank you!