A Gentle Introduction to Git at CAEN

Andrew Caird

January 29, 2013

Contents

1 Getting Started

```
git clone mistakes-and-lies
```

- What is Git
  - Git is a free distributed version control system.
  - Git manages collections of text files (known as repositories) so many people can work on them and they can be merged together again if needed, and all changes are tracked and can be seen at any time.
  - The distributed nature of Git means that if you get a copy of a Git project (also known as a copy, or a clone, or a pull), you:
    - get the entirety of the project, including all of the history
    - you become a "master repository", since every copy is a "master repository"
- One-time commands
  There are a few commands you'll run only once per system on which you use Git[1]. One is setting your name and email address:

[1] http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

You can also have per-repository names and email address by leaving off `--global` and running those commands inside of a repository.

- Making your own repository
  In any directory you can type `git init` which creates a `.git` directory for you.

  After that, you can create files or use files that exist in that directory to populate your git tree by using the commands `git add <filename>` and `git commit`

  You can then use `git clone`, `git pull`, and `git push` to copy, update from, and update to your repository.

- Migrating an existing SVN Repository
  Use `svn2git` from: https://github.com/nirvdrum/svn2git to create a new git repository using the URL of an existing SVN repo. The README file includes examples of how to convert SVN repos with non-standard layouts or to exclude certain files from the migration.

  Then create a new empty repo in github, and set the origin and push using:

  ```
  git remote add origin <you>@github.com:caen/REPO_NAME.git
  git push origin master
  ```

  More information and some best practices can be found at: https://help.github.com/articles/importing-from-subversion

- Seeing the CAEN GitHub Repositories
  The CAEN GitHub Repositories are listed at: https://github.com/CAEN/

  and looking at the list of repositories on the right-hand side of the page. You will only see those repositories to which you have been granted access.

- Checking out a CAEN GitHub Repository
  To copy a repository from the CAEN GitHub project, you first need permissions. As soon as you can see it in the list at https://github.com/CAEN/ you can copy it.

  The command to copy a repository is:

  ```
  git clone https://github.com/CAEN/<repository_name>
  ```

  After that you'll have a complete copy of everything in that repository.

  There are no "master" repositories in Git, so you can clone the repository again from your copy, or allow others to clone your copy, and merge them back in later or not. In most cases, you have files in Box or AFS so sharing isn't very practical.

- Working with CAEN Repositories
  There are two recommended ways to work with CAEN Git Repositories.

  1. Do all your work locally in branches, and push the `master` branch to GitHub when you're done.
  2. Make a "user branch" for yourself, push the branch to GitHub, do your work in that branch or branchs of that branch, merging in changes from the `master` branch from time to time, and when you'd like your branch merged into the `master` branch, ask the owner of the `master` branch to do the merge for you, so she or he can make sure your changes are appropriate.

  Both of these methods are described in more detail below.

2 Basic Branching and Merging

- Basic Branching and Merging References
  Basic Branching and Merging is well described here: http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

- Creating a Branch
  The command

  ```
  $ git checkout -b iss53
  Switched to a new branch "iss53"
  ```

  creates a branch and switches to it. This is the same as

  ```
  git branch iss53
  git checkout iss53
  ```

- Using Branches
  The command `git branch` lists the branches; the one with the `*` by it is the active branch.

  ```
  [acaird@Andrews-Mac researchcomputing (master)]$ git branch
    acaird
    agenda
  * master
    paul
    storage
    webcontent
  ```

  The command `git checkout <branchname>` switches to another branch.

  The command `git diff <branchname>` shows the differences between the current branch and `<branchname>`.

- Merging Branches
  To merge a branch with the current branch, type: `git merge <branchname>`
- An example workflow
  A common workflow is to do:
  - `git pull`
  - `git checkout -b mybranch`
  - edit files on `mybranch`
  - commit changes on `mybranch` with the `git commit` command
  - do more edits and commits on `mybranch`
  - switch back to the master branch with the command `git checkout master`
  - update the master branch with `git pull`
  - check the differences between the master branch and `mybranch` with the command `git diff mybranch`
  - if the differences look OK, merge `mybranch` into the master branch with the command `git merge mybranch`
  - push your changes back to the origin with the command `git push`
- Switching branches without committing
  To switch branches from a "dirty" branch without commiting the changes, simply type `git stash` [2], which moves your changes off to the side, thus making your current branch clean so you can switch away from it.
- Using `git stash`
  - `git stash list` lists the things you've stashed
  - `git stash apply` applies the most recent stash to the current branch
  - `git stash drop` deletes the most recent stash
  - `gitstash pop` is the same as `git stash apply ; git stash drop`
  - You can apply other stashes by naming them with their `stash@{#}` name
  - You can turn stashed changes into a branch with the command `git stash branch <branchname>` if you want to split it from the branch it was in.

[2] http://git-scm.com/book/en/Git-Tools-Stashing

3 Pushing and Pulling

- Branch Management
  Branch management in repositories is well described here: http://goo.gl/95003

- Pushing a Branch
  Using the name `plugin` for our example branch (`git checkout -b plugin`) the command:

  ```
  git push -u origin plugin
  ```

  tells git to push changes from your `plugin` branch to the `plugin` branch on the origin repository.

  - If origin does not have a `plugin` branch, it is created on the fly.
  - The `-u` tells git that you want to be able to easily push and pull changes to that branch in the future.
  - `-u` is the same as `--set-upstream` and adds a remote reference so the commands `git push` and `git pull` while in that branch locally will push to and pull from that branch remotely).

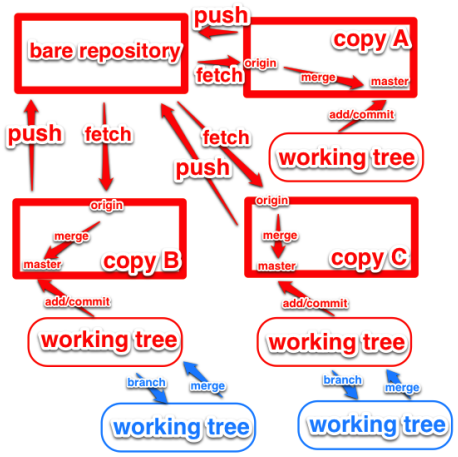
- Pulling a Branch

  ```
  git fetch origin
  git checkout --track origin/plugin
  ```

  The first command updates your repository with the changes from the remote repository.

  The second command creates a local branch named `plugin` that matches the `origin/plugin` branch and tells git that you want to be able to easily push and pull from the branch called `plugin` on GitHub.

- What does all that look like



4 Resources and Tips

- More Resources and Tips
  Git has a large community, so Google is your friend, but there are a few other things that are worth pointing out.

- `bash` command prompt
  Git maintains a lot of state, but to see it you have to ask by running `git status`

  Two of the most used pieces of state information are:

  - the name of the branch you are on
  - whether that branch is "dirty" or not.

  Using advice from http://en.newinstance.it/2010/05/23/git-autocompletion-and-enhanced-bash-prompt/ or the included (in some distributions) `git-completion.bash` you can change your shell prompt when you are in a directory with a `.git/` directory to look like:

- `bash` command prompt

  ```
  [acaird@Andrews-Mac CAEN-Testing (acaird *)]$
  ```

  In this case:

  - I am in the `CAEN-Testing` directory, which is a clone of the `CAEN-Testing` git repository
  - I am on the `acaird` branch
  - The branch is dirty, as shown by the `*`

The optional autocompletion feature is also a time saver, and can complete git commands, branch names, etc.

- Mac OS X Users - installing git through XCode
  git is not installed by default with OS X, but is included in the free download of XCode in the Mac App Store. After installing XCode, you then install the command line tools using the Downloads section in XCode's preferences.
- Mac OS X Users - Getting the git prompt
  To install the autocomplete and git prompt features, you can then:

```
curl -o ~/.git-completion.sh https://raw.github.com/git/git/master/contrib/completion/git-completion.bash
curl -o ~/.git-prompt.sh https://raw.github.com/git/git/master/contrib/completion/git-prompt.sh
```

  Then, add the following lines to your `~/.profile`, creating the file if necessary:

```
source ~/.git-completion.sh
source ~/.git-prompt.sh
GIT_PS1_SHOWDIRTYSTATE=true
```

```
PS1='\[\033[32m\]\u@\h\[\033[00m\]:\w\[\033[31m\]$ (__git_ps1)\[\033[00m\]\$ '
```

  To load the changes into the active terminal session, type:

```
source ~/.profile
```

- Enabling colors in the command line
  Many of the git commands can use color to make reading output more comfortable in the terminal, but not all installations have this enabled by default.

  To enable color:

```
git config --global color.ui true
```

- Abandoning Changes

  – you can delete a whole branch with the `-D` option to `git branch` like:

```
$ git branch
* acaird
master
$ git checkout master
$ git branch -D acaird
```

  – you can revert a file in a modifited branch with the command

```
$ git checkout -- MyFileName
```

- Git Books
  There are many books on Git, and several floating around CAEN if you want to look at them.

  I like **Pro Git** by Scott Chacon, in part because it is free in electronic forms (PDF, Mobi, and ePub), can be ordered from Amazon for about $20, and is online in HTML. All of this is at http://git-scm.com/book

- CAEN Staff

  - Thanks to Dan Maletta, Phil Trieb, and Tom Knox for their help with these slides.
  - Also, I know for a fact that the Linux Systems and HPC Groups use `git`, so you can ask them for help, too.
  - Good email address:
    - `caen-git-users@umich.edu`
    - to get added to that, email `caen-git-admin@umich.edu`