Basic Git Commands

- Init
- Clone
- Status
- Add/RM
- Commit
- Push
- Pull
- Diff
- Log
- Help
- Branch
- Merge

git init

This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton.

- Start a new git repository for an existing code base
- $ cd /path/to/my/codebase
- $ git init (1)
- $ git add . (2)
- 1. prepare /path/to/my/codebase/.git directory
- 2. add all existing file to the index

git clone

- What if you want to get a copy of an existing Git repository to contribute to it? the command you need is

  $ git clone [url]

- Every version of every file for the history of the project is pulled down when you run git clone.

- The different protocols you can use for the url are:
  - git://
  - http(s)://
  - SSH [user@server:/path.git]

git clone continued

Example using SSH.

```
$ git clone git@github.com:ECSIG/irc-client-framework.git
```

1. That creates a directory named irc-client-framework
2. Initializes a .git directory inside it
3. Pulls down all the data for that repository,
4. Checks out a working copy of the latest version.
5. If you go into the new grit directory, you'll see the project files in there, ready to be worked on or used.

git status

- The main tool you use to determine which files are in which state is the git status command. If you run this command directly after a clone, you should see something like this:

  ```
  $ git status
  ```

- Sample output:

  ```
  # On branch master
  nothing to commit (working directory clean)
  ```

git status continued

- Let's say we change a file:

```
$ vim README
```

- Now let's check the status of our repo again

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   README
nothing added to commit but untracked files present (use "git add" to track)
```

git add

- In order to begin tracking a new file, you use the command git add. To begin tracking the README file, you can run this:

```
$ git add README
```

- Now let's run git status again

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   README
#
```

How do you know if files are untracked?

- Run git status then git add any untracked files.

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   README
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#
#   modified:   benchmarks.rb
#
```

git rm

If you simply remove the file from your working directory, it shows up under the “Changes not staged for commit” (that is, unstaged) area of your git status output:

```
$ rm grit.gemspec
$ git status
# On branch master
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#
#       deleted:    grit.gemspec
```

How do we fix this?...

git rm continued

If you run git rm, it stages the file’s removal:

```
$ git rm grit.gemspec
rm 'grit.gemspec'
$ git status
# On branch master
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    grit.gemspec
#
```

A .gitignore file

- Have files that you don’t want Git to automatically add or even show you as being untracked?
- You can create a file listing patterns to match them named .gitignore. Here is an example .gitignore file:

```
# a comment - this is ignored
*.a       # no .a files
!lib.a    # but do track lib.a, even though you're ignoring .a files above
/TODO     # only ignore the root TODO file, not subdir/TODO
build/    # ignore all files in the build/ directory
doc/*.txt # ignore doc/notes.txt, but not doc/server/arch.txt
```

git commit

- Now that your staging area is set up the way you want it, you can commit your changes.
- Remember that anything that is still unstaged — any files you have created or modified that you haven't run git add on since you edited them — won't go into this commit.
- They will stay as modified files on your disk.
- The simplest way to commit is to type git commit:

```
$ git commit
```

git commit continued

- You can type your commit message inline with the commit command by using the -m flag, like this:

  $ git commit -m 'This is a message.'

- If you want to skip the staging area, providing the -a flag to the git commit command makes Git automatically stage every file that is already tracked before doing the commit, letting you skip the git add part:

  $ git commit -a -m 'This will commit all tracked files with this message.'

git push

- When you have your project at a point that you want to share, you have to push it upstream.

  Syntax: git push [remote-name] [branch-name]

- To push to the master branch to your origin server run:

  $ git push origin master

git pull

- How do you get other peoples work from the remote git server?
- Running git pull generally fetches data from the server you originally cloned from and automatically tries to merge it into the code you're currently working on.
- When in your git repo run:

```
$ git pull origin master
```

git diff

- git-diff - Show changes between commits, commit and working tree, etc

```
$ git diff
diff --git a/benchmarks.rb b/benchmarks.rb
index 3cb747f..da65585 100644
--- a/benchmarks.rb
+++ b/benchmarks.rb
@@ -36,6 +36,10 @@ def main
         @commit.parents[0].parents[0].parents[0]
       end

+      run_code(x, 'commits 1') do
+        git.commits.size
+      end
+
       run_code(x, 'commits 2') do
         log = git.commits('master', 15)
         log.size
```

git log

- How do we look back to see what has happened? The git log command:

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code
```

git help

- How do we find out how to use a command?

  $ git help [command]

Any Questions?

Resources

1. git-scm book:

http://git-scm.com/book/en/Git-Basics

2.http://help.github/

https://help.github.com/articles/ignoring-files