Short Introduction to SCM and Git

Daniel Rodriguez

Computer Science Department
University of Alcala

Table of Contents

Software Configuration Management (SCM)
Version Control

Version control systems keep track of changes to source code.
Allows multiple people to edit a project in a predictable manner.

Software configuration Management (SCM)

Source Configuration Management

Software configuration management is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.
(https://en.wikipedia.org/wiki/Software_configuration_management)
Main open source software configuration management systems

- 1982 RCS
- 1990 CVS
- 2000 Subversion
- 2005 Git/Mercurial

There are many proprietary ones.
Git is now the most popular one.
All software should be under a version control system, if not, it ain't software!

Git
What is Git?

Git is an open source distributed version control system, created by Linus Torvald.
`https://git-scm.com/`
It is easier to star with free hosting sites instead of maintaining your own server.

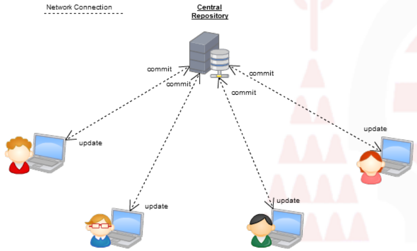
- Github: public repositories (as many as you want), but private ones are not free.
- Bitbucket: allow us to keep private repositories limiting the number of collaborators.

It is typically used as central repository:

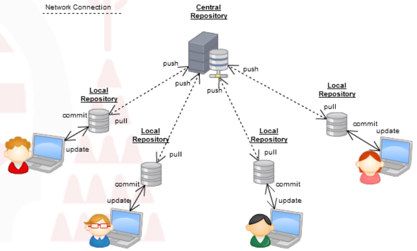
- from which everyone pulls other people's changes
- to which everyone pushes changes they have made

Git
Git vs. SVN (I)

Centralized (SVN)



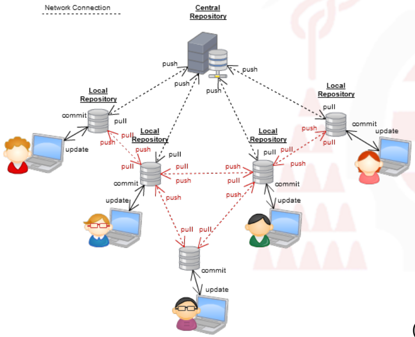
Distributed (Git)



(Source)

Git
Git vs. SVN (II)

Fully distributed (Git)



(Source)

Git concepts to know

- Commit, update
- Push, pull
- Origin, remote

Using Git
Repository initialization and clonning:Initialization

Initialization:

```
mkdir /path/to/your/project
cd /path/to/your/project
git init
git remote add origin https://<where>/<path>/<project.git>
```

Using Git
Repository initialization and clonning: Cloning

To work with someone else's repository, we first need to clone it to get a local copy.

```
git clone <repo>
```

E.g.:

```
git clone https://github.com/danrodgar/gitSlides.git
```

Note: once cloned, you can edit the repository as much as you want. No changes make their way back to the 'central' repository until you explicitly do so.

Using Git
Basic commands: Commit

Then we can start tracking files. To do so, we need to add commit, and push the file(s) that we want to track.

```
echo "A new file..." >> Readme.md
git add Readme.md
git commit -m 'Initial commit'
git push -u origin master
```

Using Git
Basic commands: Pulling

To integrate all changes other people have made since you cloned/pulled:
`git pull .`

- If you have made local changes you have to `git stash` before pulling, then `git stash pop` afterwards
- You can see which files you've modified with `git status`
- You can permanently remove your local changes by: `git checkout <file>`

Using Git
Basic commands: Pushing

`git add <file>` makes git track the file <file>
Or to record all changes into a commit (notice the '.'):
`git commit .`
`git push origin master` This pushes all new commits to the repository.

Using Git
Merge and conflicts

If two people both modify the same file, the first to push *wins*. The second person will have to pull and merge before pushing.

- Changes in different parts of a file are automatically merged
- Changes in the same part of a file cause conflicts (between «< === »>) and require the user to manually resolve them. Can select either HEAD (your changes) or remote, or a mix of the two
- Two merging cases: have / haven't committed

Using Git
Merge and conflicts: diff

`diff -u <old file> <new file>`

This command shows what changes you would need to apply to old file to change it into new file.

Lines beginning with:

- – or +++ tell you the old / new filenames
- `@@` points to where within the file you are looking (i.e. a space) are lines that are unchanged
- – is a deleted line
- + is a newly added line

Using Git
Merge and conflicts: Applying `diff` changes (patch command)

After the `patch.diff` is created as:

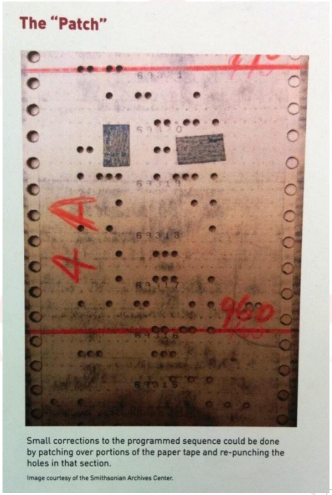
```
diff -u <old file> <new file> > file.patch
```

We can apply it with the `patch` command:

```
patch < file.patch
```

Note that the `file.patch` knows the name of the file to be patched.

Using Git
Merge and conflicts: Original Patch!



The "Patch"

Small corrections to the programmed sequence could be done by patching over portions of the paper tape and re-punching the holes in that section.

Image courtesy of the Smithsonian Archives Center.

Using Git
Commits

- Merge commits record where parallel development unified
- How does Git keep track of things when parallel development happens?
- Every commit has an ID (its hash), which is a 40 character SHA-1 hash based on the commit's content. Not guaranteed to be unique; but it probably is

Using Git
Branches

Branches are used extensively (e.g. some like feature branches).

- A repository (local and remote) can have explicit branches
- The default branch is called master
- `git branch <name>` creates branches
- `git checkout <branch name>` switch branches
- To merge branch X into Y, checkout Y and run git merge X (i.e. you say “I want to merge another branch into me”)

Using Git
Advanced Git: Getting an old commit

Sometimes you need to get an old file or discard some changes. With

- `git log`
- `git log - oneline`

we can check previous commits and select one with `checkout`, e.g.:

- `git checkout c71d008`

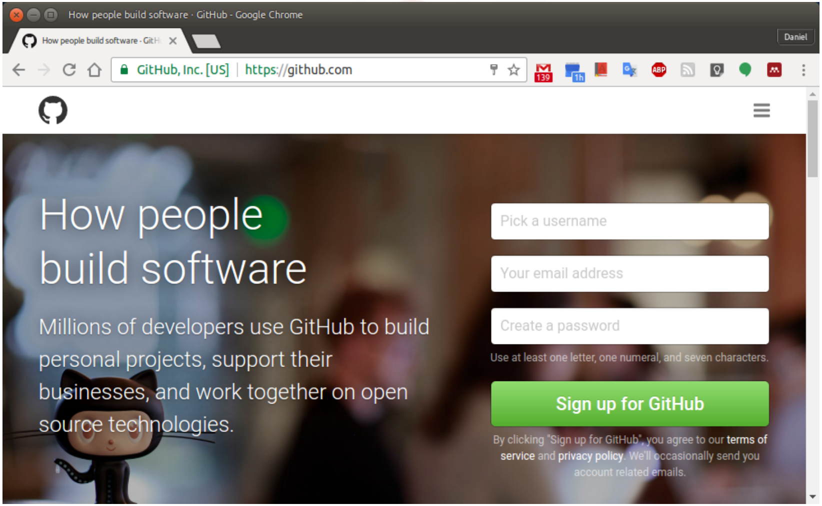
Using Git
Advanced Git: Good practices

Tipically changes are checked by someone other than their author before being merged into master. This kind of **code review** is is naturally captured by pull requests in Git.

Learn on the job: the best way to learn it is by using it. However:

- Best practice: regularly push and pull (at least daily, in general).
- Don't push half-baked changes or pull if you're in the middle of a task.

GitHub
GitHub


How people build software · GitHub - Google Chrome

GitHub, Inc. [US] | https://github.com

How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

Pick a username

Your email address

Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.