Version Control and Forges
UMONS’s Forge, Github, Bitbucket and more

Alexis Moinet
alexis.moinet@umons.ac.be

Signal Processing Department, numediart Institute
University of Mons, Belgium

July 2, 2015
Outline

1. Version Control
2. Forge
3. Demos
4. Conclusions
Outline

1. Version Control
2. Forge
3. Demos
4. Conclusions
Version Control Systems (VCS)

In short, VCSs are an answer to this dreadful observation:
Version Control Systems (VCS)

In short, VCSs are an answer to this dreadful observation:\footnote{Secondary effects may include preventing known cases of myfile.m, myfile2.m, myfile-working.m, myfile-reallyworkingthisetime.m, myfileMateiPasTouche.m, myfileNico.m, myfile.backup.m, myfile12345.m, ...}

« I didn’t change nothing but it ain’t working anymore!? »
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
  - when?
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
  - when?
  - why?
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
  - when?
  - why?

- Travel across the history of changes
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
  - when?
  - why?

- Travel across the history of changes

- Revert changes
Version Control Systems (VCS)

A Version Control System (VCS) is used to:

- Record the history of changes to a set of files:
  - who?
  - what?
  - when?
  - why?
- Travel across the history of changes
- Revert changes
- Backup. **Not!** (but ...)
Centralized VCS

- One repository to rule ... (DB + service on a server).
  - network required
  - no local commits
  - no local access to history

- Users send and receive changes (commits) to/from the server

- Examples: CVS, Subversion (svn)
Decentralized VCS

- Each user has its own “copy” (clone) of the repository locally:
  - stand-alone, no strings attached, no background process
  - local commits
  - full access to known history locally

- Users can share changes directly and/or through a server

- Branches, merges, pull requests

- Examples: Git, Mercurial (hg)
Workflows

Centralized

Decentralized
Timelines

Centralized (linear):

Decentralized (tree):
Clients

- Tortoise{Svn|Hg|Git}
- SourceTree (Mac, Win)
- Github (Mac, Win, Android)
- IDE integration (Eclipse, Netbeans, VS, Xcode, vim, emacs, ...)
- qgit, gitg, git gui, gitk, tig, ...
- The Command Line™
- ...

...
Clients

- Tortoise{Svn|Hg|Git}
- SourceTree (Mac, Win)
- Github (Mac, Win, Android)
- IDE integration (Eclipse, Netbeans, VS, Xcode, vim, emacs, ...)
- qgit, gitg, git gui, gitk, tig, ...
- The Command Line™ (no kidding!)
- ...

[The Command Line™ is trademarked by the Foundation for Open Source Software Development.]
Git commands

- `git help`
- `git init / clone`
- `git add / commit`
- `git log / status / diff`
- `git fetch / merge / pull`
- `git push / remote`
- `git branch / checkout`
- ...
- Seriously: `git help!!!`
git init (1. Create a git repository)

```
git init A && cd A
A
git add file.txt readme.txt
A
git commit -m 'First commit'
A
...
A
```
git init (2. Create a remote and push to it)

```
A
```

```
git init R --bare
R
```

```
cd A
git remote add origin /path/to/R
git push
R
```

```
R
```

A. Moinet  numediart

Version Control and Forges

2015-07-02  13 / 31
git clone

R

A

B

A

B

git clone R A

git clone R B
git add/commit

```
A
git add file.txt
A
A
B
git commit -m 'change file'
A

B
git add readme.txt
B
B
```
git push

B: git push
git fetch

A: git push → error
A: git fetch

A: git merge
or git pull
A: git rebase
git merge/rebase

A: git merge

A: git rebase

A: git push
Outline

1. Version Control
2. Forge
3. Demos
4. Conclusions
Software Forge

From Wikipedia:
Software Forge

From Wikipedia:

« A forge is a web-based collaborative software platform for both sharing computer applications and developing them. »
Software Forge

From Wikipedia:

« A forge is a web-based collaborative software platform for both sharing computer applications and developing them. »

- web-based
- collaborative
- (software) project management
Software Forge - services

A forge usually provides tools such as:

- Source code repository, code review
- Issue tracker (tickets, bugs)
- Target, milestones, calendar
- Wiki, website, documentation
- Mailing-lists, discussions, meetings
- Automatic build, test units
Software Forge - examples

- **Github**: Mage, MotionMachine
- **Bitbucket** (git, hg): mediacycle (a long long time ago)
- **Allura** (git, hg, cvs, svn, ...): SourceForge
- **Gitlab**: PREDATTOR
- **Trac** (git, hg, svn, ...): mediacycle (server died)
- **Redmine** (git, hg, svn, ...): UMONS’s Forge ➔ mediacycle (finally ...)
- . . .
https://forge.umons.ac.be
Outline

1. Version Control
2. Forge
3. Demos
4. Conclusions
Demo Forge

- Starting a new project:
  1. Create a project `foo` with the Forge
  2. Clone it:
     - `git clone https://forge.umons.ac.be/git/foo`
Demo Forge

- **Starting a new project:**
  1. Create a project `foo` with the Forge
  2. Clone it:
     - `git clone https://forge.umons.ac.be/git/foo`

- **Add an existing git project:**
  1. Create a project `bar` with the Forge
  2. Push the local repository to the Forge:
     - `git remote add origin https://forge.umons.ac.be/git/bar`
     - `git push` or `git push -u origin master` (the first time)
Demo Github

1. Create a project with Github
   - `git clone` *(new project)*
   - or
   - `git remote ... and git push ...` *(pre-existing git)*

2. Fork it!

3. Pull request!
Pointers

- From Git:
  - https://git-scm.com/doc

- From Github:
  - https://try.github.io

- From Atlassian (Bitbucket):
  - https://www.atlassian.com/pt/git/tutorial
  - https://www.atlassian.com/pt/git/workflows

- Branching:
  - http://nvie.com/posts/a-successful-git-branching-model
Outline

1. Version Control
2. Forge
3. Demos
4. Conclusions
Market shares

Source: https://www.openhub.net/repositories/compare
DVCS

Diagram showing version control interactions.
DVCS

FORGE

https://forge.umons.ac.be
DVCS

FORGE

https://forge.umons.ac.be
Things I didn’t talk about . . .

... and should have.

- branches!!!
- tags + GPG signing
- configuration options (.git/config)
- ignore files (.gitignore)
- reverts, rewrite history
- hooks (.git/hooks/)
- bisect
- cherry-pick
- . . .