Introduction to code versioning

damien.francois@uclouvain.be, juan.cabrera@uclouvain.be
November 2016

http://www.cism.ucl.ac.be/training
Introduction to code versioning

- Notions of code versioning
  - Local Data Model
  - Client-Server Model
  - Distributed Model
- Working with Git and Mercurial
- Workflows
Notions of code versioning

Versions have existed for almost as long as writing has existed
Notions of code versioning

OpenOffice Documents

Google Docs

Microsoft Office documents

Undo-Redo
Synonyms

- Version control
- Source control
- Revision control
- Source code management
Notions of code versioning

- Track the history and evolution of the project
- Who, what, when, why
- Benefits:
  - team work
  - tracking bugs
  - recovering from mistakes
Evolution of technology

- Local model
- Client-server model
- Distributed model
Local Version Control System

- **Source Code Control System (SCCS)**
- **Revision Control System (RCS)**
  - Repository is in a shared folder; everyone works there
  - Operates on individual files, checked out and in
  - Delta files are in a RCS sub-directory (repository or store)

**Diagram**

Repository

working-dir

Centralized Version Control System

- Concurrent Versions System (CVS)
- Subversion (SVN)

Repository is shared folder in remote machine accessed with ssh, http+web_dav or cvs/svn server

Developers work locally on their computer

http://tortoisesvn.net/support.html http://www.tortoise cvs.org
Centralized Version Control System

Repository/:

- repos
  - conf
    - authz
    - passwd
    - svnserve.conf
  - db
    - current
    - ...
    - revprops
    - revs
    - ...
  - format
  - hooks
    - post-commit.tmpl
    - ...
    - start-commit.tmpl
  - locks
    - db.lock
    - db-logs.lock
    - README.txt

working-dir:

- project1/
  - branches
    - rel1-0
      - hello_c.c
      - hello_c.h
      - Makefile
      - README
      - .svn
  - .svn
    - entries
    - format
    - pristine
    - tmp
    - wc.db
  - tags
  - trunk
    - hello_c.c
    - hello_c.h
    - Makefile
    - README
Centralized Version Control System

- Copy-modify-merge solution

Two users copy the same file
- Repository
- Harry
- Sally

They both begin to edit their copies
- Repository
- Harry
- Sally

Harry compares the latest version to his own
- Repository
- Harry
- Sally

A new merged version is created
- Repository
- Harry
- Sally

Sally publishes her version first
- Repository
- Harry
- Sally

Harry gets an "out-of-date" error
- Repository
- Harry
- Sally

The merged version is published
- Repository
- Harry
- Sally

Now both users have each others' changes
- Repository
- Harry
- Sally

check-out
check-out
check-in
check-in
merge
update
update

cvs commit: Up-to-date check failed for A
Distributed Version Control System

- **Mercurial**
- **Git**

Full copy of the repository

- Repository are cloned on multiple machines:
- Shared folder in remote machine with: ssh, http protocol
- No technical difference between repositories only policy says which one is the reference

+ You can work locally and make draft copies
+ Do not care about network for commits
+ Less dependent on backups every clone is a backup

https://www.mercurial-scm.org/wiki/MercurialBook
Introduction to code versioning

- Notions of code versioning
  - Local Data Model
  - Client-Server Model
  - Distributed model
- Working with Git and Mercurial
- Workflows
Global configuration

Before the use of hg or git set configuration

Edit ~/.hgrc file

```bash
[ui]
username = John Doe <johndoe@example.com>
```

Set global options for git

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

You can set other parameters as editor, merge tool ...

Ignore files

Use .gitignore or .hgignore files to select files and folders you do not want to track

```bash
# Backup files left behind by the Emacs and vim editor.
*~

# Temporary files used by the vim editor.
.*.swp

# compiled objects
.*.pyc

# directory fileter example (case sensitive)
# ignore log dir
Logs/
```
Single project single user

$ hg init
$ git init

working-dir

Repository
Single project single user

$ vim test.c
$ vim test.h
$ hg status
? test.c
? test.h

$ git status
On branch master

Initial commit

Untracked files:
(use "git add <file>..." to include in what will be committed)

    test.c
test.h

nothing added to commit but untracked files present (use "git add" to track)
Single project single user

$ hg add *
$ hg st
A test.c
A test.h

hg add is used to start tracking files

git add is used to stage files i.e. mark them for next commit

$ git add test.c
$ git status
On branch master

Initial commit

Changes to be committed:
(use "git rm --cached <file>..." to unstage)

    new file: test.c

Untracked files:
(use "git add <file>..." to include in what will be committed)

    test.h
$ hg commit test.c -m'Add test.c'
$ hg st
A test.h

$ git commit -m'Add test.c'
[master (root-commit) 46ef322] Add test.c
  1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.c
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  test.h

nothing added to commit but untracked files present (use "git add" to track)
Single project single user

$ hg commit test.h -m 'Add test.h'
$ hg log
changeset: 1:bb105d00ed3a
tag: tip
user: Damien Francois <damien.francois@uclouvain.be>
date: Tue Oct 25 10:57:35 2016 +0200
summary: Add test.h

changeset: 0:97d3b6e78985
user: Damien Francois <damien.francois@uclouvain.be>
date: Tue Oct 25 10:55:44 2016 +0200
summary: Add test.c
Single project single user

$ git add test.h
$ git commit -m 'Add test.h'
[master 56bdae9] Add test.h
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 test.h
$ git log
commit 56bdae9399a8d9d74232ee69c2a535d460acf02f
Author: Damien François <damien.francois@uclouvain.be>
Date:   Tue Oct 25 10:58:17 2016 +0200

  Add test.h

commit 46ef3222fad9d4a50fe6e5ba2f9bfa6345ae719
Author: Damien François <damien.francois@uclouvain.be>
Date:   Tue Oct 25 10:56:20 2016 +0200

  Add test.c
Single project single user

```
$ vim test.c
$ hg diff
diff -r 79fa9e722a59 test.c
--- a/test.c    Wed Nov 09 14:46:51 2016 +0100
+++ b/test.c    Wed Nov 09 14:48:08 2016 +0100
@@ -1,4 +1,4 @@
 int main()
 {  
-    int a=5;
+    int a=6;

}$
```
Single project single user

$ vim test.c
$ git diff
diff --git a/test.c b/test.c
index 0197793..0c7f097 100644
--- a/test.c
+++ b/test.c
@@ -1,4 +1,4 @@
int main()
{
-    int a=5;
+    int a=6;
} int a=6;
$
$ ssh hall "mkdir -p bcktestgit && cd bcktestgit && git init --bare"
Initialized empty Git repository in /home/pan/dfr/bcktestgit/
$ ssh hall "mkdir bcktesthg && hg init"
Single project single user + backup

$ echo -e 'paths\nhall=ssh://hall/bcktesthg' > .hg/hgrc
$ hg path
hall = ssh://hall/bcktesthg
$ hg push hall
pushing to ssh://hall/bcktesthg
searching for changes
remote: adding changesets
remote: adding manifests
remote: adding file changes
remote: added 2 changesets with 2 changes to 2 files
$ git remote add hall ssh://hall/bcktestgit
$ git remote -v
hall  ssh://hall/bcktestgit (fetch)
hall  ssh://hall/bcktestgit (push)
$ git push hall --all
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 439 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To ssh://hall/~/bcktestgit
  * [new branch]      master -> master
Multiple users central server

- Cloning

```bash
$ hg clone ssh://hall/bcktesthg
$ git clone ssh://hall/~/bcktestgit
```
Multiple users central server

- Working + committing

$ hg commit ...

$ git commit

Repository

working-dir

Repository

working-dir
Multiple users central server

- Pushing

```
$ hg push
$ git push
```
Multiple users central server

- Conflict resolution

```
$ hg push hall
pushing to ssh://hall/bcktesthg
searching for changes
remote has heads on branch 'default' that are not known locally: 7aaa042c0373
_abort: push creates new remote head 5a984c5a3005!
(pull and merge or see "hg help push" for details about pushing new heads)
$ hg pull
pulling from ssh://hall/bcktesthg
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 1 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
$ hg merge
merging test.c
warning: conflicts while merging test.c! (edit, then use 'hg resolve --mark')
0 files updated, 0 files merged, 0 files removed, 1 files unresolved
use 'hg resolve' to retry unresolved file merges or 'hg update -C .' to abandon
```
Multiple users central server

• Conflict resolution

```bash
$ cat test.c
<<<<<<<<< local
line you wanted to push
=======
current version of the line on the server
>>>>>>> other
$ vim test.c
$ hg resolve --mark
(no more unresolved files)
$ hg commit -mmerge
$ hg push
pushing to ssh://hall/bcktesthg
searching for changes
remote: adding changesets
remote: adding manifests
remote: adding file changes
remote: added 2 changesets with 2 changes to 1 files
```
Multiple users central server

- Conflict resolution

```
$ git push origin master
To ssh://hall/~/bcktestgit
  ! [rejected] master -> master (fetch first)
error: failed to push some refs to 'ssh://hall/~/bcktestgit'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From ssh://hall/~/bcktestgit
  a547735..7f32455 master     -> origin/master
Auto-merging test.c
CONFLICT (content): Merge conflict in test.c
Automatic merge failed; fix conflicts and then commit the result.
```
Multiple users central server

- Conflict resolution

```bash
$ cat test.c
<<<<<<< HEAD
line you wanted to push
=======
current version of the line on the server
>>>>>>> 7f32455dbe6bea745bc94efd6b3d5f473446d581
$ vim test.c
$ git add .
$ git commit -m merge
[master 6b884f0] merge
$ git push origin master
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 676 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To ssh://hall/~/bcktestgit
  7f32455..6b884f0  master -> master
```
Multiple users central server

- To avoid trouble when merging, pull often
- These commands merge automatically if possible

```bash
$ hg pull --update
$ git pull --rebase
```
Introduction to code versioning

- Notions of code versioning
  - Local Data Model
  - Client-Server Model
  - Distributed model
- Working with Git and Mercurial
- Workflows
One-off patch submission

• You do not have write access to the repository

Clone, then edit, commit, and extract a patch to send by email for instance:

$ hg clone https://...
$ git clone https://...

$ hg export tip > patch.diff
$ git format-patch -1 > patch.diff
Feature Branch Workflow

- Develop features in separate 'branches'
  
  $ hg branch new-feature-name ; git commit -m 'Create new branch'
  $ git checkout -b new-feature-name master

- You can list branches and choose one with:
  
  $ hg branches ; git update branch-name
  $ git branches ; git checkout branch-name

- Merge the main branch into your as often as possible
  
  $ hg up new-feature-name ; hg merge default ; hg commit -m 'Merge default'
  $ git checkout new-feature-name ; git merge master; git commit

- When new-feature is finished, merge in the opposite direction
  
  $ hg up default ; hg merge new-feature-name ; hg commit -m 'Merge new-feature'
  $ git checkout master; git merge new-feature-name
Feature Branch Workflow

https://www.atlassian.com/git/tutorials/comparing-workflows/
Gitflow Workflow
Github flow

One main branch ('trunk') that is always deployable (protect your new feature with a switch if the feature is not ready for production yet) and issue a pull request very early.
Forking Workflow

- Forked Repository
- Official Repository
- Forked Repository
- Repository
- Repository
- working-dir

Pull: Forked Repository -> Official Repository
Pull: Official Repository -> Forked Repository
Push: Forked Repository -> Repository
Push: Repository -> Forked Repository
Pull: Repository -> working-dir
Pull: working-dir -> Repository
Working with several clusters

Repository

Hmem

Lemaitre2

NIC4

working-dir

pull

push

working-dir

working-dir

working-dir

working-dir

Repository
Working with several clusters

- **Hmem**: working-dir -> pull
- **Lemaitre2**: working-dir -> pull
- **NIC4**: working-dir

**Repository**: working-dir -> push -> **Main repository**
Final messages

• Use Git or Mercurial to manage your code
• Have a remote copy of the repository

• Version everything! (data.csv, code.c, thesis.tex, .bashrc, submit.sh, todo.txt, etc.)

• Learn enough Mercurial to be able to manage a repository
• Learn enough Git to be able to collaborate with others
Final messages

Mercurial

- very easy to handle
- cannot do everything

Git

- a solution for every situation
- more changes to hurt yourself