Week 1: Foundations of Computing, Version control with Git

MSc/MRes CMEE 2014-15

Samraat Pawar

Imperial College
London

October 10, 2014

YESTERDAY'S WEE UNIX QUESTION?

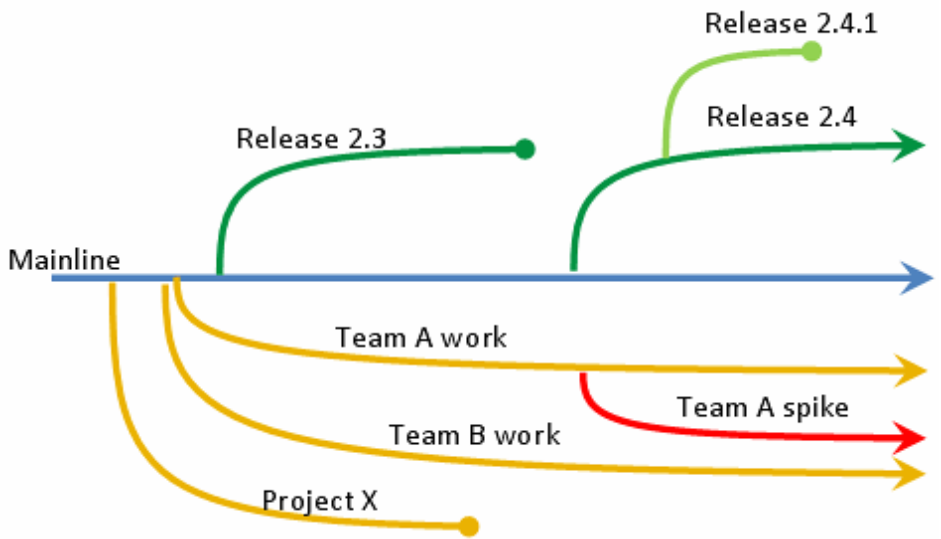
```
$ find . -type f -exec ls -s {} \; | sort -n | head -10
```

Today's Overview

- Version control with Git
- LaTeX
- Shell scripting
- Practical

What's Version Control?

- Record of all changes made to a set of files and directories, so that you can access any previous version of the files
- Branch (and merge) new projects



- "roll back" data, code, documents (not formats like *.docx though!)
- But version control is embedded in various word processors and spreadsheets, e.g., Google Docs and Sheets

WHY VERSION CONTROL?

maktoons.blogspot.com/2009/06/if-dont-use-version-control-system.html

GIT

- We will use git, developed by Linus Torvalds, the Linu in Linux
- In git each user stores a complete local copy of project, including the history and all versions
- So you do not rely as much on a centralized (remote) server
- We will use bitbucket.org – it gives you unlimited free private repositories if you register with an academic email!

GIT

- We will use git, developed by Linus Torvalds, the Linu in Linux
- In git each user stores a complete local copy of project, including the history and all versions
- So you do not rely as much on a centralized (remote) server
- We will use bitbucket.org – it gives you unlimited free private repositories if you register with an academic email!
- Install and configure `git`:

```
$ sudo apt-get install git
$ git config --global user.name "Your Name"
$ git config --global user.email "Your.Name@imperial.ac.uk"
$ git config --list
```

YOUR FIRST REPOSITORY

- Time to bring your `CMEECourseWork` under version control:

```
$ cd CMEECourseWork
$ git init
$ echo "My CMEE 2014-14 Coursework Repository" > README.txt
$ git config --list
$ ls -all
$ git add README.txt #Staging
$ git status
$ git commit -m "Added README file." #you can use -am too
$ git status #what does it say now?
$ git add -A
$ git status
```

YOUR FIRST REPOSITORY

- Time to bring your `CMEECourseWork` under version control:

```
$ cd CMEECourseWork
$ git init
$ echo "My CMEE 2014-14 Coursework Repository" > README.txt
$ git config --list
$ ls -all
$ git add README.txt #Staging
$ git status
$ git commit -m "Added README file." #you can use -am too
$ git status #what does it say now?
$ git add -A
$ git status
```

- Nothing has been sent to a remote server yet!

YOUR FIRST REPOSITORY

- Time to bring your `CMEECourseWork` under version control:

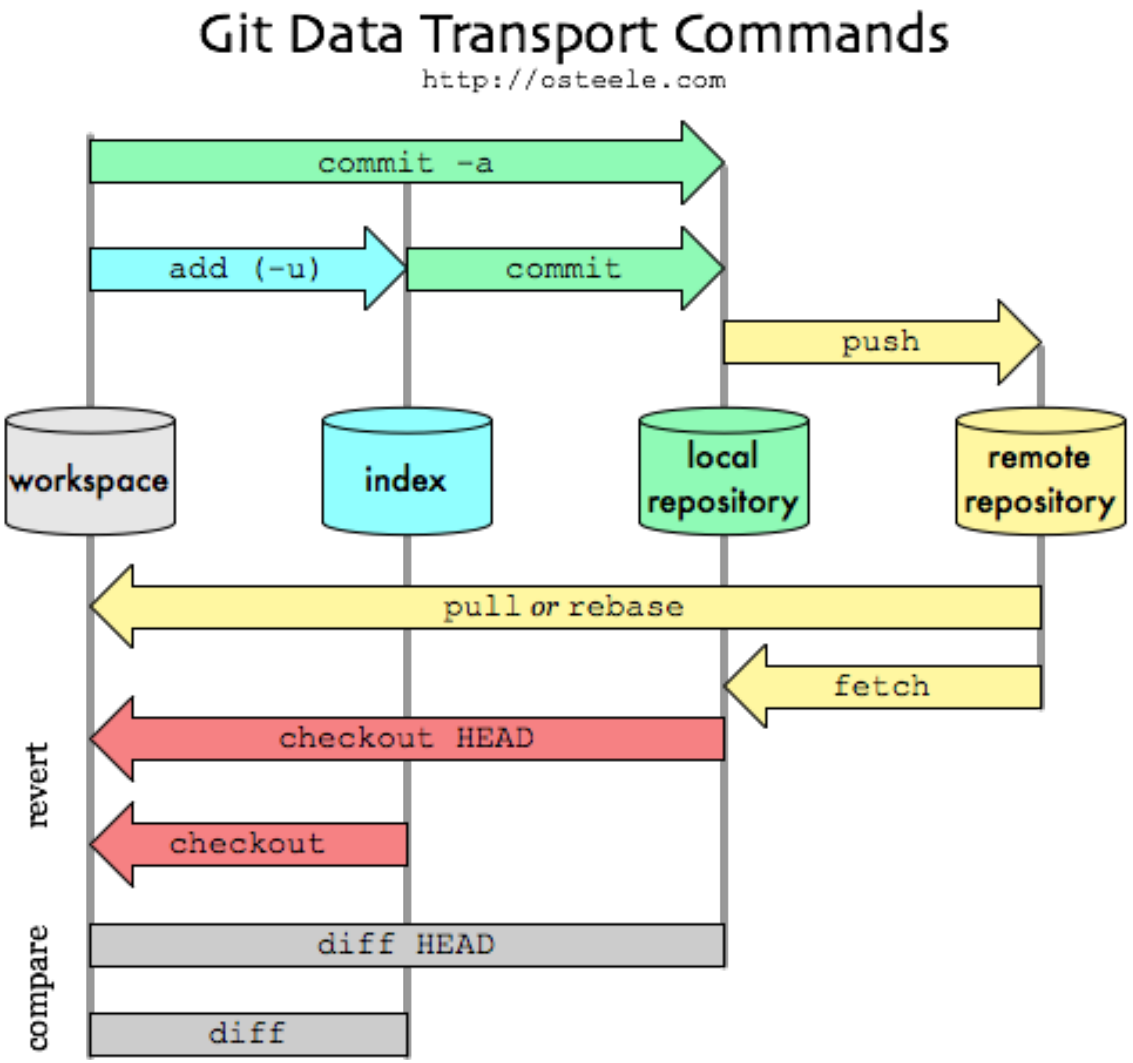
```
$ cd CMEECourseWork
$ git init
$ echo "My CMEE 2014-14 Coursework Repository" > README.txt
$ git config --list
$ ls -all
$ git add README.txt #Staging
$ git status
$ git commit -m "Added README file." #you can use -am too
$ git status #what does it say now?
$ git add -A
$ git status
```

- Nothing has been sent to a remote server yet!
- Now let's go to bitbucket

COMMIT OFTEN, COMMENT ALWAYS!

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

GIT COMMAND STRUCTURE I



Basic git commands I

git init	Initialize a new repository.
git clone	Download a repository from a remote server.
git status	Show the current status.
git diff	Show differences between commits.
git blame	Who is to be blamed for the changes?
git log	Show commit history.
git commit	Commit changes to current branch
git branch	Show branches.
git branch name	Create new branch.
git checkout name	Switch to a different commit/branch.
git pull	Upload from remote repository.
git push	Send changes to remote repository.

Back to your repository

- So nothing has been sent to a remote server yet!

Back to your repository

- So nothing has been sent to a remote server yet!
- So let's go to bitbucket.org and setup...
- Login to your account
- Set up your `ssh` based access to bitbucket – `https://confluence.atlassian.com/pages/viewpage.action?pageId=270827678`
- Then create repository there with name `CMEECourseWork`
- Then follow bitbucket instructions... let's do it now!

Ignoring Files

- You will have some files you don't want to track (log files, temporary files, executables, etc)
- You can ignore entire classes of files with `.gitignore` (be in your `CMEECourseWork`!):

```
$ echo -e "*~ \n*.tmp" > .gitignore

$ cat .gitignore
*~
*.tmp

$ git add .gitignore

$ touch temporary.tmp

$ git add *
The following paths are ignored by one of your .gitignore
files:
temporary.tmp
Use -f if you really want to add them.
fatal: no files added
```

Removing files

- To remove a file use `git rm`:

```
$ echo "Text in a file to remove" > FileToRem.txt

$ git add FileToRem.txt

$ git commit -am "added a new file that we'll remove later"
master 5df9e96 added a new file that we'll remove later
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 FileToRem.txt

$ git rm FileToRem.txt
rm 'FileToRem.txt'

$ git commit -am "removed the file"
master b9f0b1a removed the file
 1 files changed, 0 insertions(+), 1 deletions(-)
 delete mode 100644 FileToRem.txt
```

Removing files

- To remove a file use `git rm`:

```
$ echo "Text in a file to remove" > FileToRem.txt

$ git add FileToRem.txt

$ git commit -am "added a new file that we'll remove later"
master 5df9e96 added a new file that we'll remove later
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 FileToRem.txt

$ git rm FileToRem.txt
rm 'FileToRem.txt'

$ git commit -am "removed the file"
master b9f0b1a removed the file
 1 files changed, 0 insertions(+), 1 deletions(-)
 delete mode 100644 FileToRem.txt
```

- I typically just do all my stuff and then just use `git add -A`

ACCESSING HISTORY OF THE REPOSITORY

- To see particular changes introduced, read the repo's log :

```
$ git log
commit 08b5c1c78c8181d4606d37594681fdcfca3149ec
Author: Your Name <your.name@imperial.ac.uk>
Date:   Wed Oct 8 16:41:51 2014 -0500

    removed the file

commit 13f701775bce71998abe4dd1c48a4df8ed76c08b
Author: Your Name <your.name@imperial.ac.uk>
Date:   Wed Sep 5 16:41:16 2014 -0500

    added a new file that we'll remove later

commit a228dd3d5b1921ef18c5efd926ef11ca47306ed5
Author: Your Name <your.name@imperial.ac.uk>
Date:   Wed Sep 5 10:03:40 2014 -0500

    Added README file
```

- For a more detailed version, add `-p` at the end

Reverting to a previous version I

- If things horribly wrong with new changes, you can revert to the previous, “pristine” state:

```
$ git reset --hard
$ git commit -am "returned to previous state" #Note I used -am here
```

- If instead you want to move back in time (temporarily), first find the “hash” for the commit you want to revert to, and then check-out:

Reverting to a previous version II

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

$ git log
commit c797824c9acbc59767a3931473aa3c53b6834aae
Author: Your Name <your.name@imperial.ac.uk>
Date:   Wed Aug 22 16:59:02 2014 -0500
.
.
.

$ git checkout c79782
```

- Now you can look around
- However, if you commit changes, you create a “branch” (git plays safe!)

Reverting to a previous version III

- To go back to the future, type `git checkout master`

Branching I

- Imagine you want to try something out, but you're not sure it will work well
- E.g., you want to rewrite the Introduction of your paper, using a different angle, or you want to see whether switching to a library for a piece of code improves speed

BRANCHING II

- What you then need is branching, which creates an “alternate reality” in which you can experiment:

```
$ git branch anexperiment

$ git branch
  anexperiment
* master

$ git checkout anexperiment
Switched to branch 'anexperiment'

$ git branch
* anexperiment
  master

$ echo "Do I like this better?" >> README.txt

$ git commit -am "Testing experimental branch"
[anexperiment 9f17dc1] Testing experimental branch
 1 files changed, 2 insertions(+), 0 deletions(-)
```

BRANCHING III

- If you decide to merge the new branch after modifying it:

```
$ git checkout master

$ git merge anexperiment
Updating 08b5c1c..9f17dc1
Fast-forward
 README.txt |    2 ++
 1 files changed, 2 insertions(+), 0 deletions(-)

$ cat README.txt
My CMEE 2014-14 Coursework Repository
Do I like this better?
```

- If there are no conflicts (i.e., some files that you changed also changed in the master in the meantime), you are done, and you can delete the branch:

```
$ git branch -d anexperiment
Deleted branch anexperiment (was 9f17dc1).
```

Branching IV

- If instead you are not satisfied with the result, and you want to abandon branch:

```
$ git branch -D anexperiment
```

- When you want to test something out, always branch!
- Reverting changes, especially in code, is typically painful

Wrapping up

- You can do all of this using a GUI (your choice!) – some good Ubuntu git GUIs out there
- Invite me (s.pawar@imperial.ac.uk) to your `CMEECourseWork` repository
- I will invite you to a read only `CMEE2014MasteRepo`

WRAPPING UP

- You can do all of this using a GUI (your choice!) – some good Ubuntu git GUIs out there
- Invite me (s.pawar@imperial.ac.uk) to your `CMEECourseWork` repository
- I will invite you to a read only `CMEE2014MasteRepo`
- You will then `git checkout` the `CMEE2014MasteRepo`
- `CMEE2014MasteRepo` will contain data and code files for upcoming practicals
- You will `git pull` inside `CMEE2014MasteRepo` thereafter
- You will `cp` files from `CMEE2014MasteRepo` to your `CMEECourseWork` as and when needed

READINGS

- Excellent book on Git: `http://git-scm.com/book`
- Also, `https://www.atlassian.com/git/`, including Bitbucket 101