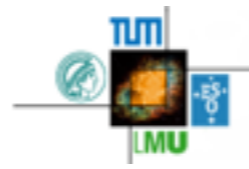




Excellence Cluster Universe

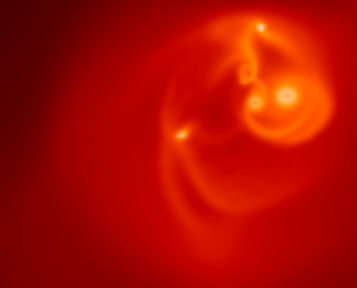


Installation & compilation

Giovanni Rosotti

IoA Cambridge

27th October 2015



Version control system

- To download and update the code, we use the version control system *git*
- Why?
 - easy to maintain several versions
 - easy to go back in time
 - fundamental to collaborate
- Git is one of the most widely used version control systems (e.g. it's used for developing Linux)

What is version control?

- Stores every version of all the files in the repository: provides therefore a *history* of the code
- The updates can be *pushed* to a repository, allowing the code to be *shared* among users
- Different versions (they are called *branches*) can coexist, allowing a feature to be developed before being *merged* in the master branch

Version control is the solution

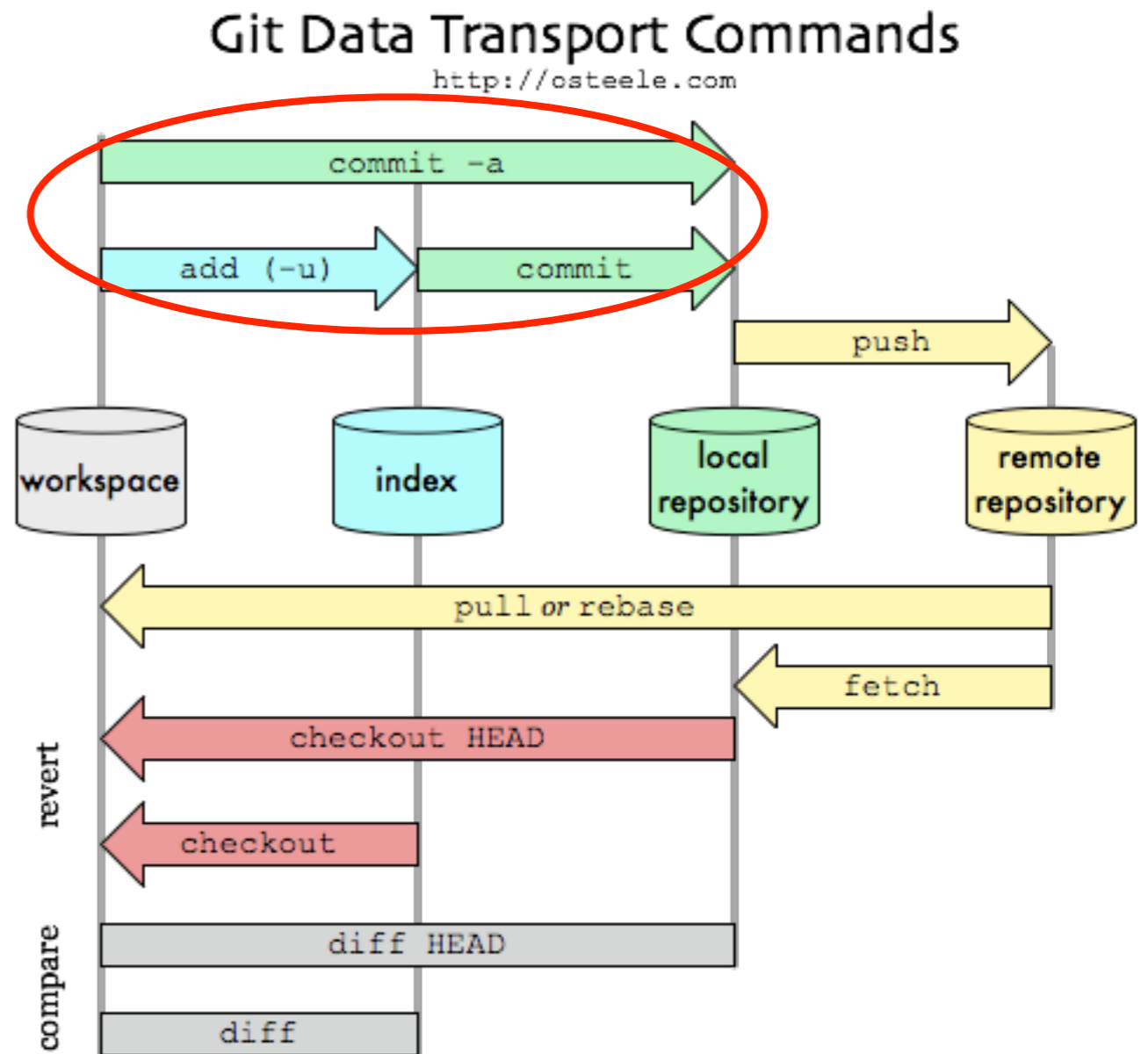
- This was the computer science point of view. In real life, if you ever had these questions (raise your hand if it never happened to you), **version control is what you need**:
 - how do I avoid having 10 folders with different versions of my code?
 - I changed something in the code and now nothing works anymore! How can I go back to a working version?
 - I am working on a code with a collaborator. How can we avoid e-mailing each other 100 times a day with different source files?

Distributed version control

- git is a *distributed* version control
- This means that you don't need to setup a server to start working; all existing copies of a repository have the same importance (as far as git itself is concerned, at least)
- A commit is different from a push (will be clearer later)
- Does NOT mean that you can't have a central repository that everybody will be *pushing* to and *pulling* from

Basic workflow when developing

- edit code & test (yes, please please test your code!)
- add changed files
- commit
- push
- The last two actions are done together in a central version control system (e.g., svn). Means you always have to be on-line to commit



A git primer

- Let's create a new folder and create a git repository

```
git init
```

- Create a file with some content
- Add the file to the “staging area” running

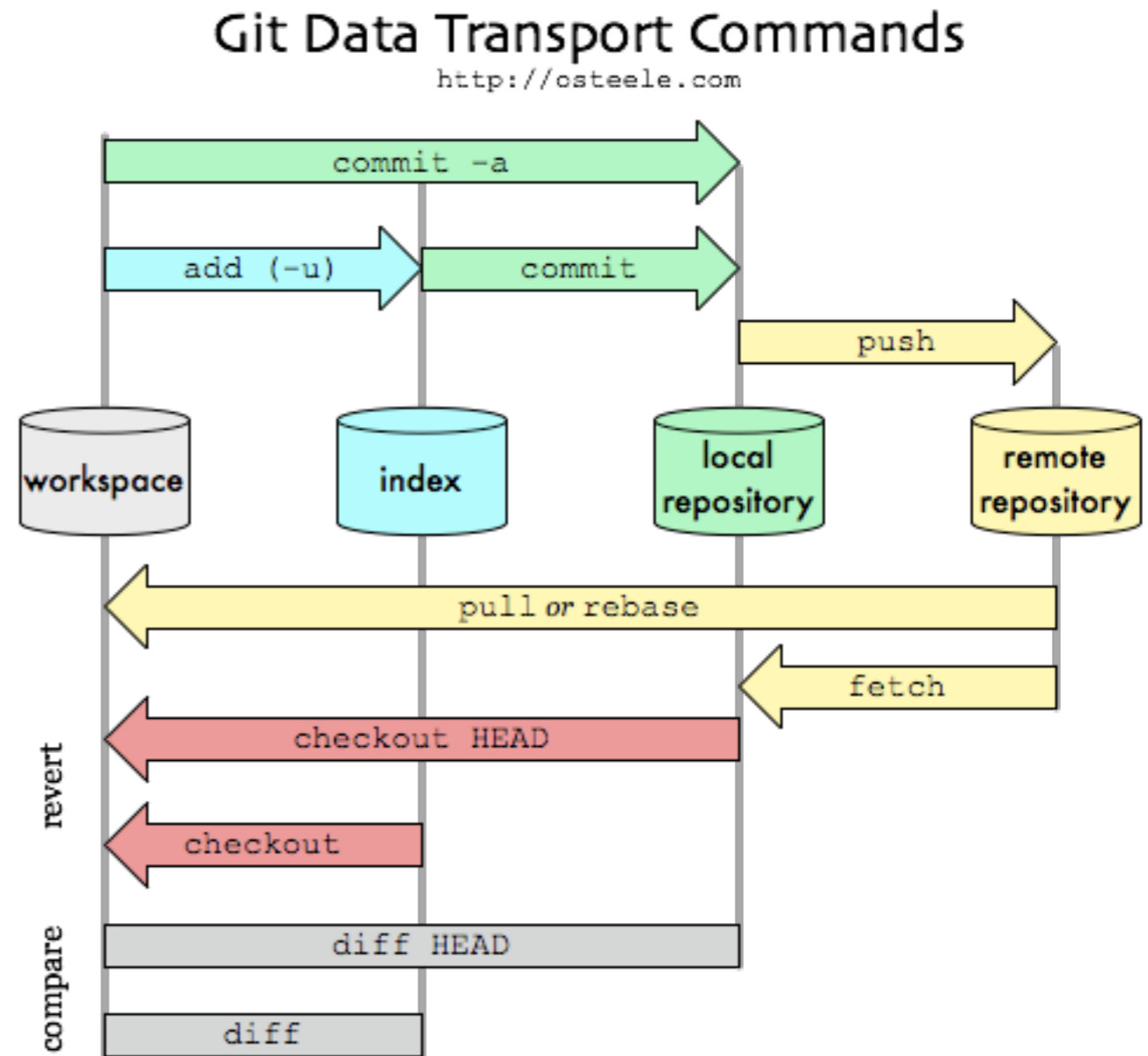
```
git add file.c
```

- Finally commit by running

```
git commit -m “A message”
```

commit & push

- A commit saves the changes *only in your local repository*
- If you want the changes to propagate somewhere else, you need to push them



pull

- In the same way as push updates the remote side, pull updates your local version
- Sometimes this is straightforward (in git language: fast-forward); more complicated if the histories have diverged
- Pulling can lead to *conflicts*, i.e. things that git cannot solve by itself
- In contrast, no conflicts can happen when pushing. Git will simply prevent you from pushing

Branches

- As many versions of the code can coexist in different branches

- create a branch

```
git branch new-branch
```

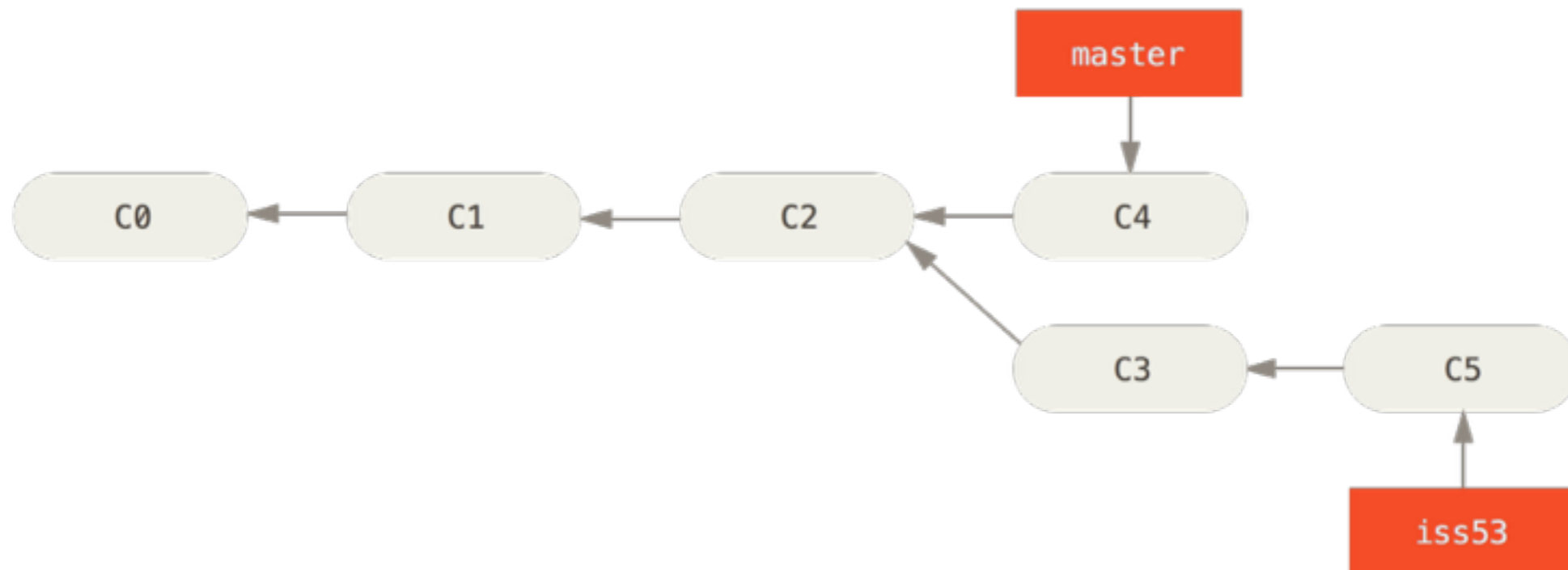
- “switch” the code to another version

```
git checkout new-branch
```


- merge changes from other branch

```
git merge other-branch
```

- In this way you can have as many versions as you want!



History



Modified Boss-Bodenheimer test to be sensible	4b92f7d	Giovanni Rosotti...	18 Oct 2015 22:50
Merge branch 'master' of https://github.com/gandalfcode/gandalf	68c412e	David Hubber <d...	06 Oct 2015 23:18
Fixed bug writing in single precision unformatted file	b2b3b67	Giovanni Rosotti...	06 Oct 2015 17:51
Solved bug with formatted output in single precision	32b92c8	Giovanni Rosotti...	06 Oct 2015 17:04
1. Split off 'On the spot' physics into separate class for modularisation (easier to add additional p...	ff00cf2	David Hubber <d...	06 Oct 2015 23:18
Added optional library and include variables for external libraries in the Makefile	ccd8ac5	David Hubber <d...	06 Oct 2015 15:06
Merge branch 'master' of https://github.com/gandalfcode/gandalf	74a51d1	David Hubber <d...	05 Oct 2015 18:55
Removed references to fortran compiler and f2py from the user guide	8537abd	Giovanni Rosotti...	05 Oct 2015 15:20
Plotting functions in facade now also return the data; added functions get_data and get_render_data	0b95741	Giovanni Rosotti...	02 Oct 2015 17:32
1. Converted Nbody routines to use standard (FLOAT) precision rather than double precision to all...	9633068	David Hubber <d...	05 Oct 2015 18:55
1. Fixed horrible bug calculating particle pointers from index (reached max. integer value). 2. Cha...	c140517	David Hubber <d...	29 Sep 2015 19:25
Merged changes	4b850fa	David Hubber <d...	28 Sep 2015 15:29
Small changes to IC.cpp	43859e4	David Hubber <d...	28 Sep 2015 15:27
1. Small changes (mainly asserts and deallocations)	30bbd61	David Hubber <d...	28 Sep 2015 14:32
Local changes	1d837df	David Hubber <d...	28 Sep 2015 15:28
Adapted RiemannSolver class to output exact solution for Shocktube analytical solution	9d570c8	David Hubber <d...	27 Sep 2015 20:34
1. Moved creation of EOS object to SPH constructor in order to fix problem with invalid pointers b...	7bf5f6c	David Hubber <d...	25 Sep 2015 13:08

- This was just a primer; if you need to know more there's a lot of material on the web!
- Personally I recommend <http://git-scm.com/docs/gittutorial>

Installing GANDALF

- So we have downloaded GANDALF. And now?
- Open the makefile and customise for your system
- Make sure you have the right dependencies:
 - C++ compiler
 - Python with scientific packages
 - SWIG

Linux

- You probably have already a c++ compiler
- Depending on your distribution, you need to use different commands for installing python:
 - red hat (fedora): yum
 - debian (e.g. ubuntu): apt-get
 - if you use another distro, chances are you know what I am saying better than me...

Macs

- The easiest thing is to use a package manager:
 - homebrew
 - fink
 - macports
- DO NOT use apple provided version of python
- apple provided CLANG is fine, but it does NOT support openMP. It should not be a problem as presumably you will not run simulations on your laptop anyway... But if you are developing on your laptop and need to test the code, use gcc
- Confusingly, gcc/g++ on a mac actually invokes clang (yes, I hate apple too)

Python - other possibilities

- You can also consider using the entought/anaconda python distributions
- A simple package that installs everything (?) you need
- In my experience, more difficult to upgrade (and sometimes you are stuck with old versions) - but a lot of people find these packages convenient
- If you are already using a package manager, my suggestion is to stick with that

Makefile

- GANDALF makefile looks like this:

```
CPP          = g++
PYTHON       = python
COMPILER_MODE = FAST
PRECISION    = DOUBLE
OPENMP       = 0
OUTPUT_LEVEL = 1
DEBUG_LEVEL  = 0

# FFTW library flags and paths.
#-----
FFTW         = 0
FFTW_INCLUDE =
FFTW_LIBRARY =

# GNU Scientific library flags and paths.
#-----
GSL          = 0
GSL_INCLUDE =
GSL_LIBRARY =
```

Substitute with the name of the compiler
and python on your system
For MPI on most system you need to use
the compiler mpic++

Test with and without openMP

Put here paths of FFTW and GSL

Makefile 2

- If you are using a non standard compiler, need to set manually the flags in src/Makefile

```
ifeq ($(findstring g++,$(CPP)),g++)
ifeq ($(COMPILER_MODE),FAST)
OPT += -O3 -ffast-math -fPIC -fno-exceptions -fno-rtti
else ifeq ($(COMPILER_MODE),STANDARD)
OPT += -O3 -fPIC -fno-exceptions -fno-rtti
else ifeq ($(COMPILER_MODE),PROFILE)
OPT += -O3 -fPIC -fno-exceptions -fno-rtti -pg
else ifeq ($(COMPILER_MODE),DEBUG)
OPT += -O3 -g -Wall -Wno-unknown-pragmas -Wno-reorder -fbounds-
check -fPIC #-f$
endif
ifeq ($(OPENMP),1)
OPT += -fopenmp
endif
endif
```

Set the options
needed for your
specific compiler

Time to get your hands dirty!

