# A brief introduction to Git

Michael Hull

University of Edinburgh

*mikehulluk@googlemail.com*

August 15, 2012

# Overview

## This talk

- Overview of Git (20 mins)
- Practical Session (30 mins)
- Further Concepts (10 mins)

# How do you ...

## ... keep your Laptop & Desktop in-sync

- Dropbox?
- E-mail? (Google Drive)
- USB stick
- (Unison/rsync/ssh)
- ??

# How do you ...

## ... keep your Laptop & Desktop in-sync

- Dropbox?
- E-mail? (Google Drive)
- USB stick
- (Unison/rsync/ssh)
- ??

## ... backup your code

- *(Methods above)*
- Time Machine
- Copy to university-server
- ??

# How do you ...

## ... keep old copies of your code

- Create 'zip' files of the directories with the date.
- ??

# How do you ...



## ... keep old copies of your code

- Create 'zip' files of the directories with the date.
- ??

*Version Control systems solve these problems (& more)*

# Version Control

## History

- Allow multiple developers to work on the same code (1970's)
- Designed by programmers for programmers - work well with text-files
- Commandline & graphical tools

# Version Control

## Common Concepts

- Save *snapshots* of a directory at a point in time, which is associated with a version number. Each snapshot is called a **commit**.
- Allow you to see the changes made between particular snapshots.
- Allow you to jump in time to different snapshots - **check-out**.
- Allow you to **branch** and **merge**.

# Centralised vs Distributed

Centralised - *SVN (subversion)* The project is stored on a central server, users communicate with the server to make snapshots, jump backwards to a previous commit, etc.

Distributed - *git (mercurial,...)* Each developer has an entire copy of the project on thier local machine. Changes are 'pushed' and 'pulled' between copies of the project. (Often a central repository is used) (Backups)

# Git



- Modern Distributed Version Control system (2005)
- Written by Linus Torvald to manage the development of the Linux-Kernel. (Last week 1 commit every 2.5 minutes.)
- Steep learning curve (there are GUI tools)
- (git can 'talk-to' SVN repositories)
- (Windows/Mac/Linux/etc)
- Although it is designed for collaborative work - we will just look at using it for a single user locally with no server.

# Git Commands

## Initial Setup

- `git init` - used once to start a new project
- `git clone <location>` - clone an existing project

## Status

- `git status` - show the status of the files in the repository

# Git Commands

## Making Commits

- `git add <filename>` - select files to add to the next commit (snapshot)
- `git commit -m '<message>'` - make a commit

## Syncronising Commits

- `git pull <location>` - update the local repository with commits made in a remote repository.
- `git push <location>` - update a remote repository with commits made in this local repository.

# Git Tools



## Graphical Tools

- gitg
- git cola

## Initial Config - tell git who you are

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

## Starting a new git repository

```
% mkdir myproject
% cd myproject
% git init

Initialized empty Git repository in /home/mhtest/
    myproject/.git/.

% ls -a

.   ..   .git
```

## Look at initial status

```
% git status

# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git␣add"
    to track)
```

## Create a file

```
% gedit myfile.txt
% cat myfile.txt

Mike's new file
Its not very interesting yet.
```

## Check the status again

```
% git status

# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be
     committed)
#
# myfile.txt
nothing added to commit but untracked files present (
    use "git␣add" to track)
```

# Make a commit

```
% git add myfile.txt
% git status

# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
# new file:   myfile.txt

% git commit -m 'my first commit'

[master (root-commit) 4fe00ad] my first commit
 1 files changed, 3 insertions(+), 0 deletions(-)
 create mode 100644 myfile.txt
```

## Look at status

```
% git status

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

## Make some changes, and a new file

```
% gedit myfile.txt
% git status (output not shown)
% gedit anewfile.txt
% git status
% git add anewfile.txt
% git commit -a -m 'my second commit'

 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 anewfile.txt
```

## Make a few more commits

```
% gedit / git add / git commit / ...
```

## Browsing history using GUI tools

```
% git log
% gitg
```

## Jumping back to an old commit

```
% git log
% git checkout <first-chars-of-sha1-hash>
% git checkout 4fe00ad2b950
% ls

myfile.txt
```

# Brief discussion of concept of branching

## Brief discussion of concept of syncing multiple computers

Pushing and Pulling

## Online git book

`http://git-scm.com/book/en/Git-Basics`

# Any Questions