Software and Programming II

An Introduction to Version Control with Git

Department of Computer Science and Information Systems
Birkbeck, University of London

September 30, 2014

What is *Version Control*?

- Allows you to keep a history of every change within a project
- Allows multiple people to collaborate on the same project, without disaster occuring!
- All files (and historical versions) are backed-up automatically

What is Git?

- Created by Linus Torvalds (yes, Mr Linux)
- Runs locally with no server needed
- Works offline
- Has a command line, GUI, or IDE interface
- Available from `http://git-scm.com`

Configuring Git

- Configure your name and email address

```
git config --global user.name = 'fred'
git config --global user.email = 'fred@host.place'
```

- Using `--global` makes these the default value for all projects

Starting a new *Git-Managed* project

1. Create directory for project
2. `cd` into project directory
3. Type `git init` to initialise Git for this project

Terminology

Repository — where the current and historial data is stored

Working copy — the local copy of files from a repository, as a specific time or revision

Commit — copy files from your working copy to the repository. These changes are stored together as an individual revision

Basic Git Workflow

1. Add, edit, delete files in/to your project in the normal way (i.e., using your favourite editor or IDE)
2. Tell Git which file(s) are to be saved into the new *commit* using `git add`
3. Commit files to the repository using `git commit`

Demo

Git commands

`git status` — view status of working copy

`git add` — add un-staged changes

- `git add filename filename ...`
- `git add .`

`git commit` — commit changes to the repository

best used as `git commit -m "{message}"`

More Git Commands

`git diff` — shows the differences betweej the working copy and the last revision

`got log` — shows the history log

`git log filename` — restrict log to file(s)

`git blame filename` — shows when/who made changes to a file

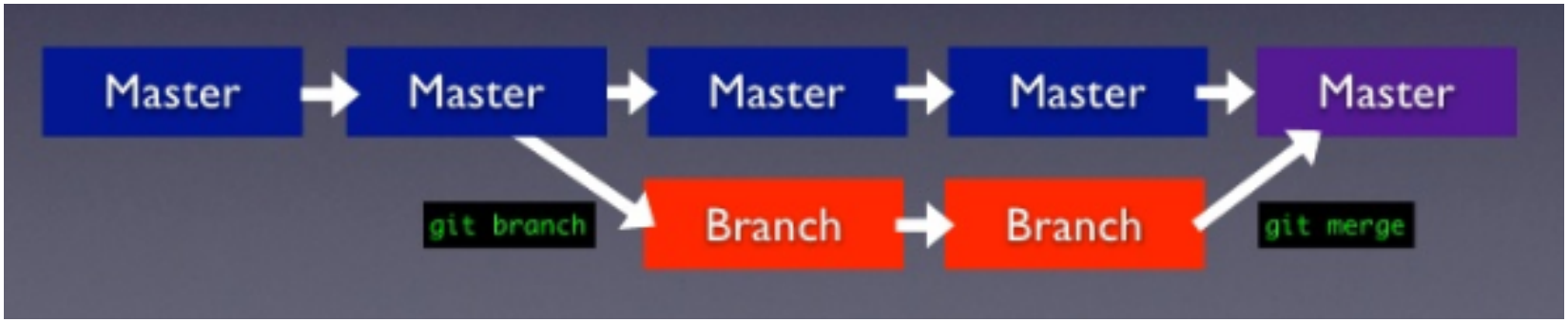
GUI tools

- Various tools exist that integrate with IDEs or are standalone
- They all give you ability to add files and create commits without resorting to the command line!
- See `http://git-scm.com/downloads` for more information

Advanced (optional) concepts

Branching

- Allows experimental features to be developed separately, without affecting the stable codebase
- Branches can be easily created and later merged together



Working with branches

`git branch` — view branches

`git branch {branchname}` — create a new branch

`git checkout {branchname}` — switch to another branch

`git checkout -b {branchname}` — create a new branch and immediately switch to it

Merging branches

Merge `otherbranch` into `mainbranch`:

1. `git checkout {mainbranch}`
2. `git merge {otherbranch}`
3. `git merge {otherbranch -m "commit msg"`

Conflict Resolution I

Demo

Conflict Resolution II

1. merge manually
2. commit the merge

Collaborating with others

Clone an existing repository:

- `git clone {anotherpath} {directory}`
- `git clone {url} {directory}`

Manage connections to remote repositories:

- `git remote`
- `git remote add {name} {url}`

Pushing and Pulling

Pull (download) from another repository:

- `git pull {remotename} {remotebranch}`
- `git pull`

Push (upload) to another repository:

- `git push {remotename} {remotebranch}`
- `git push`

Some Git hosts

GitHub — `http://www.github.com`

Bitbucket — `https://bitbucket.org`

Gitorious — `http://www.gitorious.org`

Unfuddle — `http://www.unfuddle.com`

...

GitHub. . .

Demo

Questions