Super Quick Intro To Git

Allan McRae

allan@archlinux.org

http://allanmcrae.com/files/intro-to-git.pdf

What is Version Control

- (a.k.a. Revision Control – Wikipedia...)
- A system to keep track of changes to your code
- Organisational tool

Why Use Version Control

- Multiple versions of a project
  - Developmental (many...)
  - Release
- Easily track and revert breakages
- Integrate changes provided by others
- Have a record of changes you made and why
- Backup (there are better ways...)

Why Use Git?

- It is the best! (entirely not subjective at all...)
- Fast
- Can be used off-line
- Space efficient
- Cheap branching
- Easy merging

Getting Started

- **Set-up git...**

```
$ git config --global user.name 'Allan
McRae'

$ git config --global user.email
'allan@archlinux.org'

$ git config --global core.editor "vim"
```

Getting Started

- **Create a repository**

```
$ mkdir foo

$ cd foo

$ git init
Initialized empty Git repository in
/home/allan/foo/.git/
```

Getting Started

- **Add and commit a file**

```
$ touch README

$ git add README

$ git commit
```

Commit Messages

- When committing a file, your editor will be opened for you to add a commit message
- These are very important!
- Provide enough detail so that you can go back and look at the change after a long time and understand what you did and why

Commit Message Format

- Short summary line
- Blank Line
- Longer description

- e.g.

```
$ git commit
Initial commit to repo

This is a long description of what I
just did. But since it was only
committing a blank README file, it is
rather useless.
```

Changing Files

- **Just edit, add, commit**

```
$ vim README

$ git commit -a -m “Updated file”
[master 9d70563] Updated file
 1 file changed, 1 insertion(+)
```

Browse History

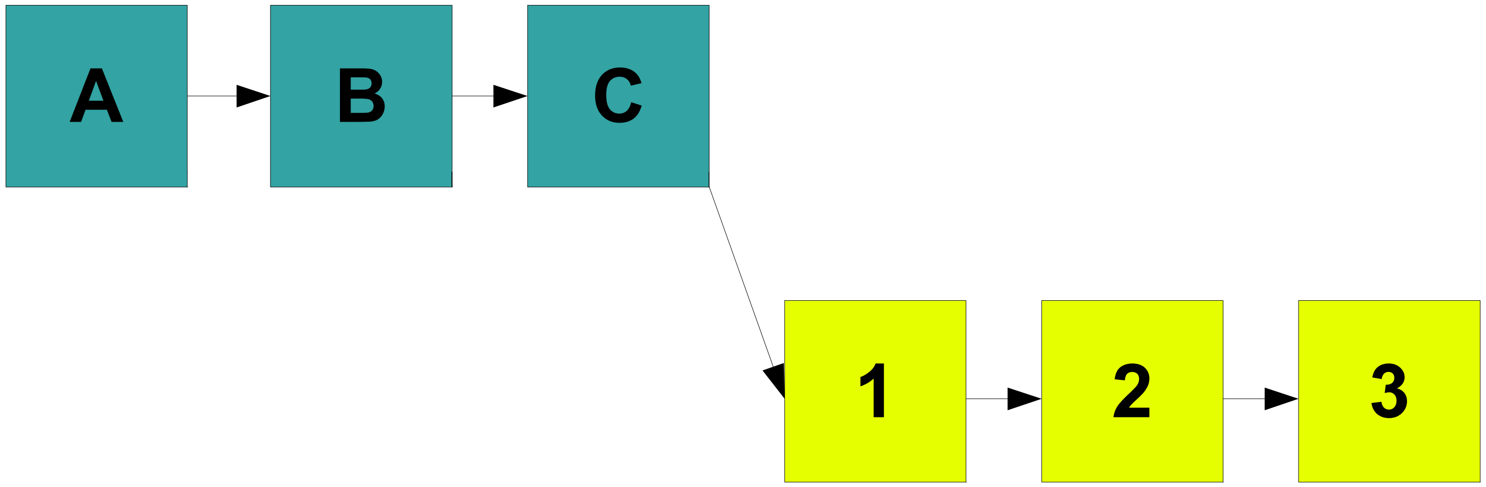
- **Also graphical interfaces (gitk, gitx, ...)**

```
$ git log
commit 9d70563c10d993ffc0a96d37631b5...
Author: Allan McRae
<allan@archlinux.org>
Date:   Wed Mar 13 17:01:34 2013 +1000

    Updated file

commit 100c93e2352fcc4fa38709128c363...
...
```

Using Branches

- Developmental/Release versions
- Try out new features
- Cheap → use them frequently

Using Branches


$$\begin{tikzcd}
A \arrow[r] & B \arrow[r] & C \arrow[dr] & & & \\
& & & 1 \arrow[r] & 2 \arrow[r] & 3
\end{tikzcd}$$

Using Branches

- **Create and switch to a branch**

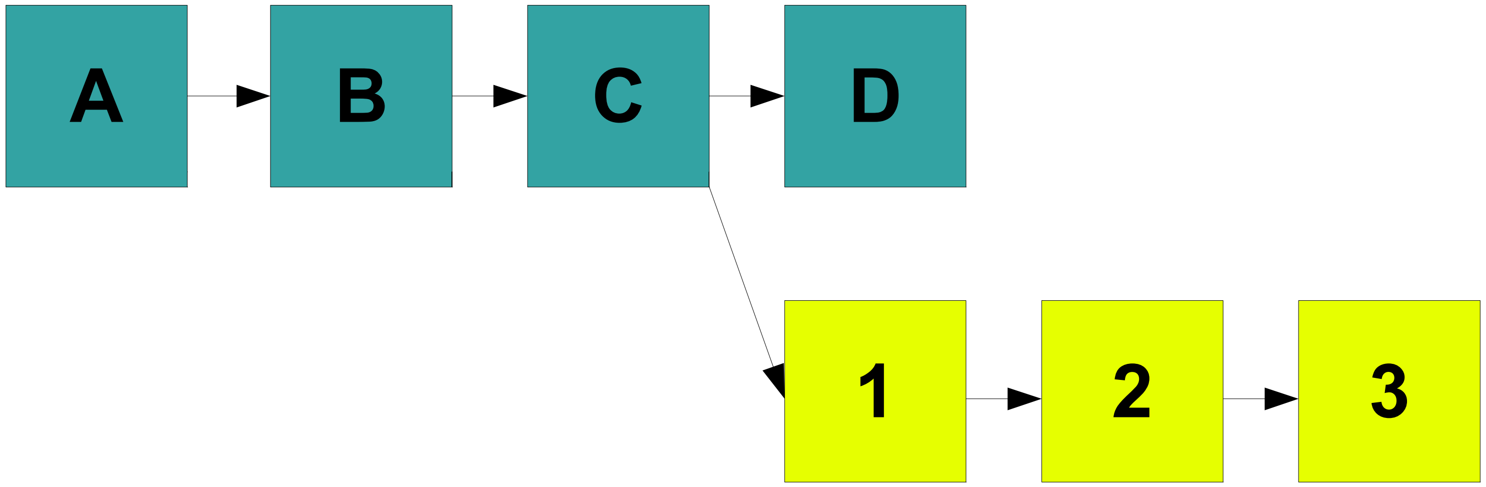
```
$ git branch working

$ git checkout working


$ git branch
  master
* working
```

Using Branches

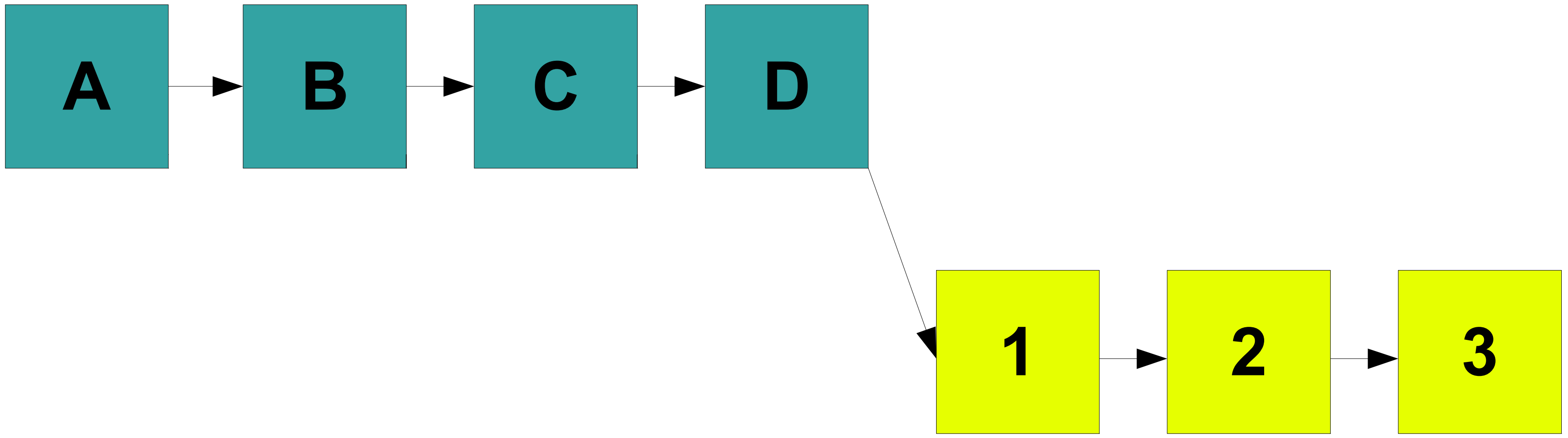
- **Integrate changes back to master**

```
$ git checkout master

$ git merge working
```

Using Branches



Using Branches



Using Branches

- **Rebase branch onto master**

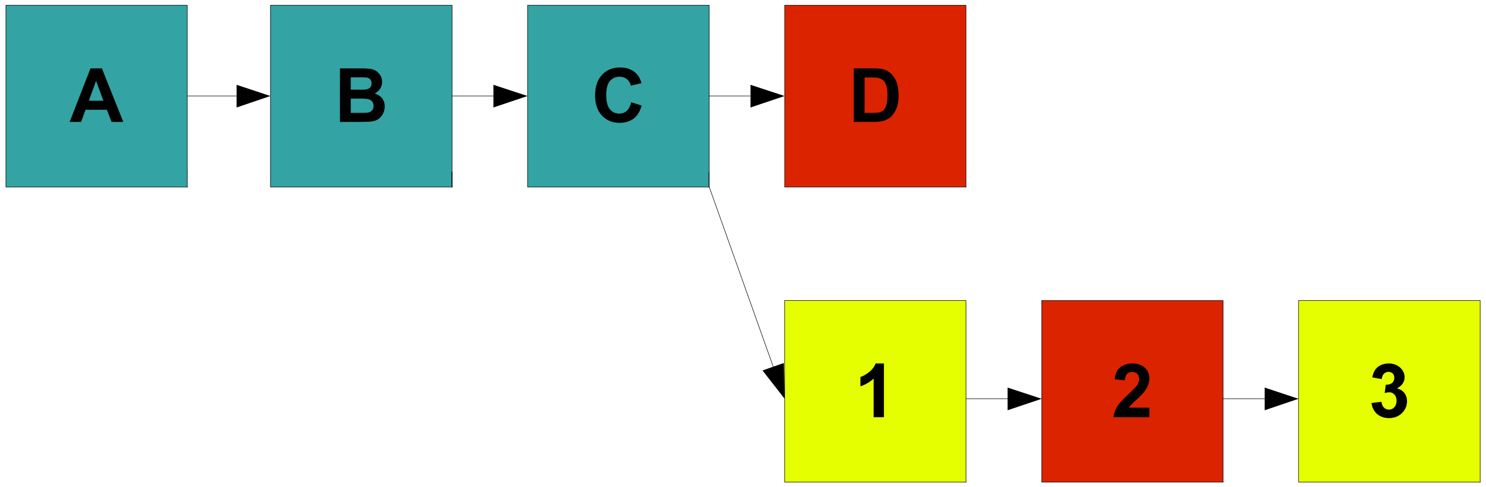
```
$ git checkout working

$ git rebase master
```

Using Branches

- Rebasing is a powerful tool
- Change commit order
- Merge/split commits
- Make fixes in earlier commits
  DO NOT DO ON PUBLIC MASTER BRANCH!!!

```
$ git rebase -i master
```

Using Branches



Using Branches

- **Dealing with conflicts – git conflict markers**

```
$ cat foo.c
<<<<<<< HEAD
current content
=======
branch content
>>>>>>> newbranch

$ vim foo.c
$ git add foo.c
$ git rebase --continue
```

Working With Others

- **Get a copy of the main repository**

```
$ git clone git://example.org/foo.git
$ cd foo
...

$ git fetch
$ git pull
```

Working With Others

- **Adding other peoples working repos**

```
$ git remote add allan
http://allanmcrae.com/foo.git

$ git checkout -b allan-working
allan/working



$ git remote update -p
```

Other Useful Commands

- See current status (files changed, new files)

```
$ git status
```

- Look at current changes

```
$ git diff
```

Other Useful Commands

- Temporarily store current changes

```
$ git stash
```

- Restore stored changes

```
$ git stash pop
```

Other Useful Commands

- **Locate a broken commit**

```
$ git bisect start

$ git bisect bad HEAD

$ git bisect good <commit>

$ git bisect <good|bad>
```

Other Useful Commands

- **Revert a commit**

```
$ git revert <commit>
```

- **Pull single commit (e.g. from a branch)**

```
$ git cherry-pick <commit>
```

Hiding Files From Git

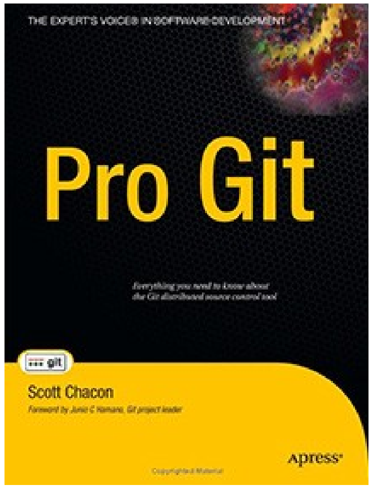
- Do not want generated objects in the repo
- Add a “.gitignore” file in root directory
- Add list of files to ignore (wildcards allowed)

For More Info

- **http://gitref.org/**

> **“quick reference for learning and remembering the most important and commonly used Git commands”**

- **http://progit.org**



License