Crash course in revision control
Or, how to never lose work again

Phillip Whittlesea - DevECS

Electronics and Computer Science

April 24, 2012
What we are going to cover today:

- Why do we all use revision control?
- What is revision control?
- Git
Who here has lost work?

Who here has never lost work?
Why do we use revision control?

Which of the following is the largest?

- A: A Peanut
- B: An Elephant
- C: The Moon
- D: A Kettle

Figure: Not using revision control is always a bad choice.
Why do we use revision control?

Figure: This is what happens when you use Revision Control
The number one rule for revision control?

Use It!
What is revision control?

♦ More commonly known as "Source Control Management" (SCM)
♦ What does a SCM do:
  ♦ Keep track of files creation/deletion, changes over time
  ♦ Backup to offsite store
♦ Two major types:
  ♦ Centralised SCM
    ♦ Server: The Repository
    ♦ Client: Local changes
  ♦ Decentralised SCM
    ♦ Anyone can be the server
    ♦ Complete history
    ♦ Offline usage
Just remember, you have everything locally

**Figure:** Git has several layers allowing for complete control

*Image from http://whygitisbetterthanhx.com/*
Google it!

It can get pretty advanced pretty quickly
Time to rock and roll

Let's get it started...

Time for you guys to join in! Steps to follow:
1. Have a computer with git OR ssh into linuxproj
2. Have a GitHub account
Time to rock and roll

Let's get it started...

Time for you guys to join in! Steps to follow:
1. Have a computer with git OR ssh into linuxproj
2. Have a GitHub account

To GitHub!

Let's start simple and start a project on GitHub
Git init, or, where do baby repositories come from

**In a nutshell**, you use `git init` to make an existing directory of content into a new Git repository. You can do this in any directory at any time, completely locally.

```
$ git config --global user.name 'Phillip Whittlesea'
$ git config --global user.email pw6g08@ecs.soton.ac.uk

$ mkdir learn-git
$ cd learn-git
$ git init
```
Git init, or, where do baby repositories come from

**In a nutshell**, you use `git init` to make an existing directory of content into a new Git repository. You can do this in any directory at any time, completely locally.

```sh
$ git config --global user.name 'Phillip Whittlesea'
$ git config --global user.email pw6g08@ecs.soton.ac.uk

$ mkdir learn-git
$ cd learn-git
$ git init
```

To the command line!

Creating our repository
Git status, or, 'Hey Git what's happening?'

**In a nutshell**, you run git status to see if anything has been modified since your last commit so you can decide if you want to commit.

```
$ touch README
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what...
# README
```
Git status, or, 'Hey Git what's happening?'

In a nutshell, you run git status to see if anything has been modified since your last commit so you can decide if you want to commit.

$ touch README
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
# (use "git add <file>..." to include in what...
#  README
Git add, or, what to do if you like your changes

**In a nutshell**, you run `git add` on a file when you want to include whatever changes you've made to it in your next commit.

```
$ git add README
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
# new file:  README
```
Git add, or, what to do if you like your changes

In a nutshell, you run `git add` on a file when you want to include whatever changes you've made to it in your next commit.

```
$ git add README
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
## new file:  README
#
```

To the command line!

Telling Git what to remember
Stop the press, what is a staging area?! 

In a nutshell, think of the staging area as a loading dock before your truckload of code heads off to Commitsville².

---

²Image from http://whygitisbetterthamex.com/
Git commit, or, lets keep that as its kinda important

**In a nutshell**, you run `git commit` to record the snapshot of your staged content. This snapshot can then be compared, shared and reverted to if you need to.

```
$ git commit -m 'first commit'
[master (root-commit) ddf46d4] first commit
  0 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 README
$ git status
# On branch master
nothing to commit (working directory clean)
```
Commit messages

What I care about:
- Features you have implemented
- Bugs you have fixed
- Purpose of the commit
- Related tickets

What I DON'T care about:
- If you're going to lunch

How frequently do I commit?
Commit little, stable code often!
Git commit

Commit messages

What I care about:
- Features you have implemented
- Bugs you have fixed
- Purpose of the commit
- Related tickets

What I DON'T care about:
- If you're going to lunch

How frequently do I commit?
Commit little, stable code often!

To the command line!
Telling Git that you REALLY want this
Git remote, or, where did I leave that repository?

**In a nutshell**, with git remote you can list remote repositories. Use `git remote add` to add new remotes and `git remote rm` to delete existing ones.

```bash
$ git remote add origin \
  git@github.com:pwhittlesea/learn-git.git
```
Git push, or, sharing is caring!

**In a nutshell**, you run `git push` to update a remote repository with the changes you've made locally.

```bash
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 213 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:pwhittlesea/learngit.git
  * [new branch] master -> master
Branch master set up to track remote branch master from origin.
```
Git push, or, sharing is caring!

**In a nutshell**, you run `git push` to update a remote repository with the changes you've made locally.

```
$ git push -u origin master
Counting objects: 3, **done**.
Writing objects: 100% (3/3), 213 bytes, **done**.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:pwhittlesea/learngit.git
  * [new branch] master -> master
Branch master **set up** to track remote branch master from origin.
```

To the command line!

Deploying local code to a remote location
Git pull / fetch + merge; bring it back now y'all!

**Git fetch + merge**

*Git fetch*, will pull changes from the remote to the local repo.

*Git merge*, will merge a branch into the current context.

**OR**

**Git pull**

*In a nutshell*, you run git pull to take the latest changes from the remote repository.
tl;dr: What we've covered...

Figure: Don’t forget git pull!
Git Branch, or, programming your bit on the side!

In a nutshell, you use `git branch` to list your current branches, create new branches and delete unnecessary or already merged branches.

![Diagram showing branch management]

**Figure:** A single git repository can maintain multiple branches of development.
Git Branch, or, programming your bit on the side!

In a nutshell, you use `git branch` to list your current branches, create new branches and delete unnecessary or already merged branches.

![Diagram of branch structure](image)

Figure: A single git repository can maintain multiple branches of development.

To the command line

Creating a branch for some simple changes
Git ignore
Ignore those files we don't care about

Git clone
Checkout a remote repository into the local folder

Stuff gone wrong?

```
git reset --HARD
```