Using Git

Ciaran has a set of SAR tools he is maintaining using Github; [LINK] Here’s how you can get that code for your own use, make any changes you feel are needed and share it with anyone else who would require it.

Installing Git

If you’re on Windows, then you can install Git for Windows via the University’s Program Installer. If you’re on an Ubuntu machine, then Git is usually pre-installed. If not, then you can install it with

Sudo apt-get install git-all

on the command line.

[CALLOUT: From here out, it’s assuming that you are using a standard Bash terminal for navigation. If you are on windows, a good enough emulation comes with Git for Windows; just type ‘Git Bash’ into the search bar.]

Getting the code

First, navigate to a convenient directory using cd and mkdir, as appropriate. Once there:

$git clone [LINK]

This is now the **root** of your local repository – as far as Git is concerned, everything is relative to this point.

What is happening here

This will copy all of Ciaran’s code from Github to your machine into a **local repository**, complete with the history and branches [see later] of his project right to the beginning – you can view this with the command ‘git log’.

Why do this?

As well as getting the entire project’s codebase, when you cloned the repository with [git clone], you also cloned its history and configuration. This means that you can:

* Keep it up to date with any revisions Ciaran might make in the future
* Review any changes he might have made in the past, and understand why he made those changes.
* If you make a huge, unfixable mistake, you can roll the entire project back to a point where
* Fix any bugs or glitches you might find
* Add any further features that you think the code might need, and share those features with anyone else.

Keeping the code up-to-date

Whenever Ciaran or anyone else makes a change or adds a new feature, you can easily update his code on your own machine with this command:

$git pull origin master

You should now see a short summary of what has been changed; you can get more information using [git diff], or see the later section on using graphical tools.

What is happening here

TODO more here

[CALLOUT: When you first cloned Ciaran’s repository, your local copy of Git stored a reference to it under the name ‘origin’. The repository is set up such that the default branch is ‘master’ - more on branches in the next section.]

Editing code of your own

Let’s say you wanted to add a function to Ciaran’s codebase. You’ve written the script – let’s call it [newthing] – in a file of its own (newthing.py), and now you want to add it to Ciaran’s repository so everyone else can use it. This process has a few steps.

1. Put newthing.py in a sensible place in your **local repository** using a file manager. You can check it’s there with

$git status

As you might gather from the status message, Git does not know that you want to add newthing.py to the repository.

1. Use the command

$git add FILEPATH

where FILEPATH is the path from the **root of your local repository**. If in doubt, use the message that git status gives you.

This adds newthing.py to your **staging area.** This is where any changes you make are stored until you **commit** them to your local repository in the next step. Once a file is added to your staging area, it will stay there until deleted or explicitly removed; you do not need to keep [add]ing it.

1. Commit the changes to your **local repository** with the command

$git commit –m “some informative message here”

The message you write in here is what will appear when anyone runs “git log”, so make sure it’s informative and clean(ish)

[CALLOUT: What to do if you get trapped in Vim here]

[CALLOUT; WHEN TO COMMIT]

[CALLOUT: Writing good commit messages]

-Making your own changes, the staging area and local commit

--getting stuck in Vim

-Branching

-Merging

-Rolling back

-Quick reference at end