git cheatsheet for students

Marcel Oliver

(Dated: September 8, 2016)

1. FIRST TIME SETUP

To begin, install git on your machine; google for the best install method for your operating system. Then set a few global variables, at a minimum, you should type

```
git config --global user.name "My Name"
git config --global user.email "xxx@yyy.zzz"
```

(Insert your name and email address!) You might want to set a preferred editor (for me it's emacs) and color coding of command output.

```
git config --global core.editor /usr/bin/emacs
git config --global color.ui "auto"
```

There are other more advanced configuration options not necessary for a quick start.

2. FORK THE COURSE REPOSITORY

The public course repository is hosted at

`https://bitbucket.org/marcel_oliver/acm221/`

Visit this web page. If you do not have a bitbucket account, register for one. Click the "···" menu on the left, select "Fork", and select "This is a private repository". Click "Fork repository" to proceed. Now select the cog icon ("Settings") and go to "Access management". Add me (username `marcel_oliver`) and the Teaching Assistant (username `kpjkpj`) to the list of users with *write access*. Now you are done on the server.

On your local machine, go to the directory where you want to create the local repository and type

```
git clone \
ssh://git@bitbucket.org:username/acm221.git
```

You should now see a new directory named `acm221`; enter this directory. Now type

```
git remote add instructor \
ssh://git@bitbucket.org:marcel_oliver/acm221.git
```

This sets up `instructor` as a name for my public course repository. In particular, it allows you to pull all updates to the class material and assignments by issueing

```
git pull instructor master
```

any time in the future.

3. LOCAL WORK CYCLE

The following describes a standard git work cycle which allows you to keep a record of the changes you make. To later push to the remote repository, you need to do this at least once.

1. Do some work (add, modify, or delete files).
2. Double-check your work:
   `git status` displays which files have changed,
   `git diff [filename]` gives details of changes.
3. `git add filename` or `git add .` will stage some or all files for the commit.
4. `git commit -m "Commit Message"` performs the actual commit.
   `git commit -am "Commit Message"` is a shortcut for the last two steps *only* if no new files need to be added.

4. PUSHING AND PULLING

When you want to share your work (or submit your homework), push it to the server saying

```
git push
```

Vice versa, if the server branch has work that you want to merge into your local branch, say

```
git pull
```

5. BRANCHING

In the standard workflow for this class, you can work on the one branch named `acm221` which you had initially forked from the instructor's repository.

Often, it is useful to work with several branches, e.g. to try out several options before settling for the best solution. One of the easiest ways to create a new branch is to write

```
git checkout -b lisa
```

You are now working on the new branch `lisa`. To push it to the server, write, for the first time

```
git push origin lisa
```

where `origin` is a short-cut referring to the server. Subsequently, you can simply say `git push`. Vice versa, when you want to see Fred's branch `fred`, you write first time

```
git fetch origin fred
git checkout fred
```

Subsequently, you can check out your local version of `fred`, then pull:

```
git checkout fred
git pull
```

You can pull changes from other branches into your branch. E.g., to pull in new changes from local branch `lisa` into your local branch `fred`, use

```
git checkout fred
git pull . lisa
```

If the branch `lisa` you wish to pull from resides on th server instead, replace the last line by

```
git pull origin lisa
```

If you need more fine-grained control, separate the `pull` into a `fetch` followed by a `merge`, see `http://longair.net/blog/2009/04/16/git-fetch-and-merge/`.

To fetch all remote branches *without merging*, use

```
git remote update
```

This is useful for inspecting the branches that others are working on or downloading them for future off-line use.

6. QUERYING GIT

A list of commands for obtaining status information:

- `git log` will show the commit history of the currently check out branch.
- `git status` shows which branch you are on, which files have been changed, and which files are staged for commit.
- `git branch` lists all local branches. Add the `-r` switch for remote, or the `-a` switch for all branches *known to your local repository.*
- `git remote show origin` lists remote branches *by querying the server* and prints information on your local remote branches.
- `git config -l` prints global configuration information.

7. DELETING AND RESETTING

git makes branching easy. So you do experimental work on a separate branch to later merge or cherry-pick into your main branch. To then delete the experimental branch `crazyidea` in your local repository, use

```
git branch -d crazyidea
```

To delete `crazyidea` on the server, use (with great care!)

```
git push origin :crazyidea
```

(The command line interface of git is not known for its consistency. The logic behind this seemingly crazy command is that it pushes an empty branch into the existing branch `crazyidea` in the remote location `origin`.)

To reset your working copy to the last commit, deleting all changes, use

```
git reset --hard HEAD
```

8. A NOTE ON PROGRAM-GENERATED OUTPUT

To keep the repository as clean as possible, do not commit output which is automatically generated by programs such as LaTeX or Python. The repository is configured not to add any such files by default, the exclusion rules are contained in the file `.gitignore` in the root directory. If you have figures or other documents as PDF files from external sources, for example, you should place them into subdirectories named `figures` or `literature`.

In particular, to read any of the `.tex` files in the repository, you need to run them through LaTeX.

9. FURTHER READING

- *Understanding git conceptually*, an excellent general introduction to git by Charles Duan.
  `http://www.sbf5.com/~cduan/technical/git/`
- *Version control with Git: resources*, a blog post specifically aimed at scientists with a number of useful links by Fernando Perez.
  `http://www.fperez.org/py4science/git.html`
- The git project website with comprehensive manuals and download instructions.
  `http://git-scm.com/`
- *Git Reference* is another fairly comprehensive reference site.
  `http://gitref.org/`
- How to *set up an ssh key for bitbucket* to never type in your password again.
  `https://confluence.atlassian.com/bitbucket/set-up-ssh-for-git-728138079.html`