Continuous Integration
Software Engineering for Scientists

Git reset, restore, revert:

Reset vs. revert: if there are past commits that are erroneous and you want to undo the changes made in those commits you can use both of these. The difference is that “reset” will eliminate the record of them in the log. “Revert” on the other hand will ADD new commits that document the undoing of the changes in the prior commits.

Checkout old commit:

```
git checkout <sha hash>
# once finished you can reattach head with
git checkout <branchname>

# can also use HEAD@{} syntax from reflog
git checkout HEAD@{#}
```

Removing a file from the repo:

```
Git rm –cached filename.py
Git rm -r –cached directory
```

Remove a file from repo AND the directory (working tree):

```
git rm filename.py
```

Had the wrong commit message in last commit—amending commit message

```
git commit –amend -m “new commit message”
```

Forgot a file in *previous* commit:

```
git add forgotten_file.txt
git commit –amend -m “new commit message”

# Or if you don’t want to edit the commit message
git commit –amend –no-edit
```

Reset previous commit(s):

```
Git reset –hard HEAD~1
# try git fetch and you will find that remote is now a commit ahead, so you
have to do:
Git push –force
```

git reset [--hard | –soft] HEAD~#

Git reset resets the head to the specified state

–soft Does not touch the index file or the working tree at all (but resets the head to `<commit>`, just like all modes do). This leaves all your changed files "Changes to be committed", as `git status` would put it.

–mixed Resets the index but not the working tree (i.e., the changed files are preserved but not marked for commit) and reports what has not been updated. This is the default action.

–hard Resets the index and working tree. Any changes to tracked files in the working tree since `<commit>` are discarded.

Revert erroneous commit:

```
git revert <sha hash>  # or HEAD~#
git push
```

Commited to wrong branch—move last commit to correct branch

```
Git reset –mixed HEAD~1
Git checkout correct_branch
Git add/commit
Git checkout wrong_branch
Git push –force origin wrong_branch
# do this to remove the commit in the wrong branch if it had been pushed to
remote
```

Placed tag on wrong commit—move tag to right commit

git reflog

Reference logs, or "reflogs", record when the tips of branches and other references were updated in the local repository. Reflogs are useful in various Git commands, to specify the old value of a reference. For example, `HEAD@{2}` means "where HEAD used to be two moves ago", `master@{one.week.ago}` means "where master used to point to one week ago in this local repository", and so on. See gitrevisions[7] for more details.

https://stackoverflow.com/questions/3689838/whats-the-difference-between-head-working-tree-and-index-in-git