ENGR 301 *Project Management*

Lecture 11 — git IV

James Quilty

School of Engineering and Computer Science
Victoria University of Wellington

Introduction

Today's lecture concludes our discussion of basic usage of git with branches. References you may find useful are:

- git documentation `https://git-scm.com/docs`
- Pro Git book
  `https://git-scm.com/book/en/v2`
- Git: Mastering Version Control (O'reilly)
  `https://bit.ly/4accAng`
- Version Control (Git)
  `https://missing.csail.mit.edu/2020/version-control/`

Cherry-picking

Sometimes you'll want to move a commit, or series of commits, from one branch to another.

- On the target branch use `git cherry-pick <sha>`
- Use `git cherry-pick --no-commit <sha>` if you just want the files without the commit
- Perform an interactive rebase on the source branch to drop moved commits.

The local vs. remote rules apply: local is *safe[-ish]* but working with commits pushed to remote should be approached with caution.

Merging Branches: Resolving Merge Conflicts

Merging branches is simple, we use GitLab's Merge Request interface. What happens when there's a merge conflict and how do we resolve it *safely*?

```
Risky live-coding demonstration here!
```

Note that merge conflicts can also occur when performing a rebase.

A git GUI like GitKraken is helpful when resolving merge conflicts!

Pruning Branches

Keeping your local copy tidy sometimes requires some manual intervention, particularly regarding branches.

```
git remote prune origin
git branch --all
git branch --delete <branch>
```

will help remove merged and deleted branches from your local copy.

Again, reference to a git GUI like GitKraken will be helpful.

History

The commit history shows information about what was changed:

- Use `git log` to see the commit history
- Plenty of flags, e.g. `git log --oneline` and `git log --numstat`
- Obtain the History of a specific file with:
  ```
  git log --all --full-history -- <file>
  ```
- Can restrict to a range of commits:
  ```
  git log --numstat 082bb416..cdaa956f -- device
  ```

Often easier to use a git GUI like GitKraken for readability.

Restoring Files

There's more than one way to restore a specific file to a previous version:

```
git checkout <commit_hash_id> -- <file_path>
git restore --source <commit_hash_id> <file_path>
```

The contemporary way is to use **git restore** (`https://git-scm.com/docs/git-restore`) although note the warning from the documentation:

THIS COMMAND IS EXPERIMENTAL. THE BEHAVIOUR MAY CHANGE.

Pitfall: if the file has been moved to a different path in the tree subsequent to `<commit_hash_id>` then the checkout/restore will put the file at the *old* path.