Git Exchange 7.11.2013 - Rebase, Rewrite, Amend

- Sometimes rebasing doesn't work / stops in the middle
  - How to do it in multiple steps?
    1. Run a normal rebase to get e.g. most up-to-date master
    2. Reorder commits
    3. Squash some commits

- What is rebase ?
  - Rebase VS. Interactive rebase (--interactive or -i)
  - Rebase
    - is by default against tracking branch
    - Rebasing means:
      - 1. Rewind to last known common reference (e.g. in master)
      - 2. Fast-forward to latest current (local) master
      - 3. **REAPPLY** your branch's commits **ON TOP** of master
    - e.g. from dev branch, git rebase master
  - Interactive rebase
    - Just using the rebase command to **REWRITE** your past 5 commits
      - or just to reword another local commit back in the history
    - e.g. git rebase -i HEAD~5
    - is very handy for non-shared branches, as a normal rebase won't do anything
    - **needs guide for git interactive rebase**
      - e.g. reword doesn't happen in the interactive editor
      - first choose your commands (pick, edit, reword, squash)
        - deleting a line drops this commit, changes will not be reapplied!
      - then save & quit the editor
        - goes to next editor for rewording
        - (or back to edit position in the history)

- **git commit --amend**
  - same as git commit, except that it appends changes to the last commit!

- Workflow to keep dev branch in sync with master
  - Merge master into dev regularly (don't rebase to avoid spoiling history)
  - There are merge commits in your dev branch
  - Then when you want to merge back:
    1. pull latest master or .x you want to rebase your branch on top of
    2. rebase
    3. merge into master --ff-only
       - note that merge commits that were on your branch are magically gone on master!

- git pull = git fetch + merge
  - git fetch: gets all new commit refs from origin (new tags, newly shared branches, new commits)
  - git pull --rebase = git fetch + rebase

- people not using console
  - how to configure the editor of your choice
  - e.g. export EDITOR=vi
  - or in .gitconfig
  - when using IDE plugins, it's not always obvious what is really going on

- bugfix on branch, merge back into both master + .x branch ?
  - cherry-pick is safest option
    - we can cherry-pick multiple commit refs
  - if bugfix was derived from master, merging to 4.5.x branch will never happen smoothly
    - it might try to reapply anything back to common ancestor of 4.5.x and master!
  - squash first

- is it too late if a commit is already pushed?
  - yes.
- also just to amend message of already pushed commit?
  - yes.

- videos?

- Show the files involved in one commit
    - git show <commitref> --name-status
        - where <commitref> can be
            - HEAD (last commit)
            - HEAD^ (commit before last)
            - HEAD~2 (two commits before HEAD)
            - cafebab (one specific commit hash as visible in git log)
    - adds a list of added (A) modified (M) and deleted (D) file names in that commit