Part 1

Ask about RED/GREEN stickies

```
set -C   # set -o noclobber
set -o   # just to check
         # find a line that says 'noclobber  on'
```

```
cd ~/Desktop
git init git-adv
cd git-adv
```

```
cat <<EOF  > README.md
## Advanced Git lesson

**Where**: ITPro Forum

**When**: $(date)

EOF
```

```
git status
git diff
git add --intent-to-add README.md
git status
git diff
```

```
git add -u
git diff --staged # too long
git config --global alias.staged 'diff --staged'
git staged
```

```
git config --list
git config --list --show-origin
```

```
git commit -m "Beginning of the 'Advanced Git' lesson"
```

- create a file called `00-git-goodies.md`
- write to `00-git-goodies.md` :

```
echo "add -N: tell Git that we're going to add this file later" >> 00-git-goodies.m
```

```
git status
git add -u
```

nothing happens. Why?

```
git add `00-git-goodies.md`
```

write to `00-git-goodies.md` :

```
echo "add -u" >> 00-git-goodies.md
echo "config --global alias.<name> 'command'"
```

```
git staged
git diff
git commit -a -m "Add notes about the 'add' command and more"
```

write to `00-git-goodies.md` :

```
echo "commit -a = (add -u) + commit" >> 00-git-goodies.md
```

```
git add -u
git commit --amend
git log --oneline
```

```
echo "commit --amend: ammend last commit" >> 00-git-goodies.md
```

```
git log --oneline -p
```

```
echo "" >> 00-git-goodies.md
echo "commit -a" >> 00-git-goodies.md
echo "commit --amend" >> 00-git-goodies.md
```

Part 2

Now, we'd like to work on a side-project. However, we don't want it to affect our main one for various reasons (in-development, special feture for customers, etc)

```
git branch devel
git checkout devel
```

notice the message:

> M 00-git-goodies.md

what is that?

```
git status -s
git branch
```

```
echo "git branch <name>: creates a new branch" >> 01-branches.md
echo "git checkout <name>: checks out files from the named branch" >> 01-branches.m
```

```
git add 01-branches.md
git commit --author="Bob Dylan <rob.allen@zimmer.man>" --date="2005-02-03T23:12:00
Close encounter of Git branches

notes about `branch` and `checkout` commands
<Ctrl+D>
```

```
git log --oneline
git log --oneline --all
```

```
git config --global alias.lg 'log --oneline --graph --decorate --all'
git lg
git lg -p
```

```
git checkout master
ls # no `01-branches.md` file
```

We have just created a new branch, checked it out, made some changes in that branch and then switched back to our master branch.

Let's add a note about `git checkout <branch-name>` to `00-git-goodies.md`

```
echo "checkout <name>: switch to a branch called <name>" >> 00-git-goodies.md
```

Let's also add some comments about Git:

```
# add text "Git rulezz" to the top of `00-git-goodies.md`
git commit -am "Adding notes about checking out branches"
```

After thinking for a little bit while, we decide that we need to don't what these two changes to be in the same commit, so we want to split this commit into two.

```
git reset --soft HEAD~1
git reset -p HEAD README.md
git commit -m "Notes 1"
git commit -am "Git RULEZZ"
```

Hmmmm. The problem is that the last commit stands out and is, perhaps, not ready for the main branch. Let's move it to another branch!!!

```
git checkout another/branch
git reset --hard HEAD~1
```

That's it!

Hey, but we forgot about the `devel` branch. Let's merge the changes that we made there to our main branch!

```
git checkout master
git merge devel
git log --oneline --graph --all
git branch --list --merged
git branch -d devel
```

You decide to give it a try and merge that "Git Rulezz" `git cherry-pick <commit of Git Rulezz>`

FREESTYLE