Git Challenge Step 8

1 Background

At any stage, Git maintains three important stores of information about your files: the working files you are editing and changing, the commit history giving snapshots of all the changes you made, and the "index" or "staging area" that records changes that you want to add to the next commit.

The `git add` command updates the index based on the state of your working directory. The `git commit` command creates a new commit from the current index. The `git reset` command can be used to unstage files.

When switching between branches, modified and new files in your working directory would be lost, as the switch sets the files to the snapshot at the tip of the target branch. For this reason, git will warn you if you try to switch branches with modified files in place.

Sometimes you have made changes that are not quite ready to commit, but you want to switch branches and do some other work. For this purpose, git allows you to "stash" your current work and restore it later, either on original branch, or even elsewhere.

Here, we will see how to use `git stash` for that purpose.

2 Tasks

- Switch to the `master` branch on your local repository and on the Github repository you created in the previous step.
- Search the list of tags for a tag whose name starts with "gimme". (If using the `-l` option to `git tag`, you can use the pattern `"gimme*"` with the quotes.)
- Check out the **single file** `Src/showme` from the commit pointed to by this tag. That is, use

  ```
  git checkout TAGNAME -- Src/showme
  ```

so only the file `Src/showme` is changed to the version from that tag.

- Use `git status` to look at the status of your repository. You will see the `Src/showme` file has changed but is not yet staged. That should be the **only** change listed.
- Try to switch to the `mars` branch. Git should balk.
- Type `git stash` to stash your work. Enter a descriptive message.
- Type `git stash list` and you will see a "stack" of stashed changes.

  Move to the `mars` branch and back to `master`.
- Look at `git status`, then type `git stash apply` to redo the changes. Look at `git status` again. Your work is restored.

  Add the new file and commit all changes with a message.
- Do `git stash list` and notice that the stashed information is still there. What good is that?
- Try to switch to the branch `vulcan`. Git will refuse, even though `git branch -r` will show that both `origin` and `supplement` have a `vulcan` branch.

  The problem is that Git doesn't know *which* branch you want: the one on `origin`, or the one on `supplement`? We have to tell it which we want. We'll do this with

  ```
  git checkout -b vulcan supplement/vulcan
  ```

  This creates a branch called `vulcan` in *your* repository that matches the one in `supplement`.

  Now type `cat Src/showme` at the shell command prompt to look at the contents of the file.

  Now try `git stash apply` and look at the contents of the file and `git status`. Notice how the *changes* you made in the other branch were simply applied here on top of the existing file.
- Restore the state of `vulcan` by typing `git checkout HEAD Src/showme` and do `git status` to confirm that no changes are reflected.
- Move back to the `master` branch.

- Eliminate the stashed information by typing `git stash drop`. The top of the stack is now deleted, which you can confirm with `git stash list`.
- Now create the file `NEXTSTEP.pdf` by typing

  ```
  sh Src/showme
  ```

  at the command prompt, which will show you the password. Stage the changes and commit. Push these changes to your github copy of the repository.

  Now open `NEXTSTEP.pdf` with the password that was given.

3 Resources

- Stashing basics: `https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning`
- Undoing changes (git stash section): `https://www.atlassian.com/git/tutorials/saving-changes`
- Documentation (type `git help COMMAND` or follow the links below) for
  - `git stash` to move between branches: `https://git-scm.com/docs/git-stash`
  - `git checkout` to move between branches: `https://git-scm.com/docs/git-checkout`