Github Project: Git Commands Documentation Template

Programming for Data Science Nanodegree Program

The git commands were used to complete all tasks on local and remote git repository for this project.

1. Fork project and clone to local by ssh
   a. **git clone git@github.com:rabbithunter0502/pdsnd_github.git**
2. Copy bikeshare.py and csv file to local repo
3. Add files to stage, commit and push code to master branch on remote
   a. Check file status: **git status**
   b. Add all change to stage: **git add** .
   c. Commit change: **git commit -m "feat: Init code for bikeshare project"**
   d. Push to master branch on remote: **git push**
4. Checkout new branch: documentation
   a. **git checkout -b documentation**
5. Update README file and push to remote
   a. Edit README.md
   b. Check file status: **git status**
   c. Add all changes to stage: **git add .**
   d. Commit change: **git commit -m "docs: Update the README.md file to be consistent with the workflow in the Bikeshare project."**
   e. Edit README.md
   f. Check file status: **git status**
   g. Add all changes to stage: **git add .**
   h. Commit change: **git commit -m "docs: Add pandas and numpy docs link to README"**
   i. Push to master branch on remote: **git push --set-upstream origin documentation**
6. Checkout to master branch on local

a.   **git checkout master**
7.   Checkout new branch: refactoring
   a.   **git checkout -b refactoring**
8.   Update code in bikeshare.py and push to remote
   a.   Edit bikeshare.py
   b.   Check file status: **git status**
   c.   Add all changes to stage: **git add .**
   d.   Commit change: **git commit -m "refactor: Refactor get_filters function"**
   e.   Edit bikeshare.py
   f.   Check file status: **git status**
   g.   Add all changes to stage: **git add .**
   h.   Commit change: **git commit -m "docs: Add pandas and numpy docs link to README"**
   i.   Push to master branch on remote: **git push --set-upstream origin refactoring**
9.   Checkout to master branch
   a.   **git checkout master**
10. Fetch all changes from remote
   a.   **git fetch --all**
11. Merge documentation branch to master
   a.   **git merge documentation**
12. Merge refactoring branch to master
   a.   **git merge refactoring**
13. Push local change in master branch to remote
   a.   **git push**
14. Verify all changes on github repo

1. Set Up Your Repository

The following are the steps you will take to create your git repository, add your python code, and post your files on GitHub.

Step 1. Create a GitHub profile (if you don't already have one).
Step 2. Fork a repository from Udacity's GitHub Project repository and provide a link to your forked GitHub repository here:

GitHub Repository Link
<Add a link to your forked GitHub project repository>

Step 3. Complete the tasks outlined in the table below and copy and paste your git commands into the "Git Commands" column. The first git command is partially filled out for you.

	Tasks	Git Commands
A.	Clone the GitHub repository to your local repository.	git clone <insert the repository link you provided above>
B.	Move your bikeshare.py and data files into your local repository.	**No git command needed (you can use cp or a GUI)**
C.	Create a .gitignore file containing the name of your data file.	**No git command needed (you can use touch or a GUI)**
D.	List the file names associated with the data files you added to your .gitignore	**No git command needed (add the file names into your .gitignore file)**
E.	Check the status of your files to make sure your files are not being tracked	
F.	Stage your changes.	
G.	Commit your changes with a descriptive message.	
H.	Push your commit to your remote repository.	

2. Improve Documentation

Now you will be working in your local repository, on the BikeShare python file and the README.md file. You should repeat steps C through E three times to make at least three commits as you work on your documentation improvements.

	Tasks	Git Commands
A.	Create a branch named *documentation* on your local repository.	
B.	Switch to the *documentation* branch.	
C.	Update your README.md file.	**No git command needed (edit the text in your README.md file)**
D.	Stage your changes.	
E.	Commit your work with a descriptive message.	
F.	Push your commit to your remote repository branch.	
G.	Switch back to the master branch.	

3. Additional Changes to Documentation

In a real world situation, you or other members of your team would likely be making other changes to documentation on the documentation branch. To simulate this follow the tasks below.

	Tasks	Git Commands
A.	Switch to the *documentation* branch.	
B.	Make at least 2 additional changes to the documentation - this might be additional changes to the README or changes to the document strings and line comments of the bikeshare file.	
C.	After each change, stage and commit your changes. When you commit your work, you should use a descriptive message of the changes made. Your changes should be small and aligned with your commit message.	
D.	Push your changes to the remote repository branch.	
E.	Switch back to the *master* branch.	
F.	Check the local repository log to see how *all the branches* have changed.	
G.	Go to Github. Notice that you now have two branches available for your project, and when you change branches the README changes.	**No git command needed**

4. Refactor Code

Now you will be working in your local repository, on the code in your BikeShare python file to make improvements to its efficiency and readability. You should repeat steps C through E three times to make at least three commits as you refactor.

	Tasks	Git Commands
A.	Create a branch named *refactoring* on your local repository.	
B.	Switch to the *refactoring* branch.	
C.	Similar to the process you used in making the documentation changes, make 2 or more changes in refactoring your code.	**No git command needed (edit the code in your python file)**
D.	*For each change,* stage and commit your work with a descriptive message of the changes made.	
E.	Push your commits to your remote repository branch.	
F.	Switch back to the *master* branch.	
G.	Check the local repository log to see how *all the branches* have changed.	
H.	Go to GitHub. Notice that you now have 3 branches. Notice how the files change as you move through the branches.	**No git command needed**

5. Merge Branches

	Tasks	Git Commands
A.	Switch to the *master* branch.	
B.	Pull the changes you and your coworkers might have made in the passing days (in this case, you won't have any updates, but pulling changes is often the first thing you do each day).	
C.	Since your changes are all ready to go, merge all the branches into the master. Address any merge conflicts. If you split up your work among your branches correctly, you should have no merge conflicts.	
D.	You should see a message that shows the changes to the files, insertions, and deletions.	**No git command needed**
E.	Push the repository to your remote repository.	
F.	Go to GitHub. Notice that your master branch has all of the changes.	**No git command needed**

Submission:

This concludes the project.
- Please review this document to make sure you entered all the required response fields in all four sections.
- Download this document as a PDF file.
- Submit the PDF file on the Project Submission page within the Udacity Classroom.