GIT Cheatsheet

Common Required Commands for Assignment

After GIT is initialized, for this assignment you will only need a few basic commands:

1) Create a new branch:
   ```
   git branch branch_name
   ```

2) Checkout a branch:
   ```
   git checkout branch_name
   ```

3) Add new files (and deletions):
   ```
   git add -A
   ```

4) Commit code:
   ```
   git commit -am 'Message'
   ```

5) Merge a branch:
   ```
   git merge branch_name
   ```

6) Delete a local branch:
   ```
   git branch -d branch_name
   ```

7) Pull code from HEAD:
   ```
   git pull origin branch_name
   ```

8) Puch code to HEAD:
   ```
   git push origin branch_name
   ```

Typical Process

For the most part your GIT commands will follow a common cycle. Let's pretend you are adding a new function to generate random password for the Shared Functions group. Your GIT process would look like this:

1) If you don't have a branch named after your group then create one:
   ```
   git branch shared_functions
   ```

2) Checkout the group's branch:
   ```
   git checkout shared_functions
   ```

3) Pull the most recent stable version from your group:

```
git pull origin shared_functions
```

4) Next you have a task to complete so you will create a branch for that task:

```
git branch random_password
```

5) You will then checkout that branch:

```
git checkout random_password
```

6) Then you start coding. When the task you are working on is complete and tested you will add all new files and deletions if there are any:

```
git add -A
```

7) Commit the changes with a message:

```
git commit -am 'Completed a random password generating function'
```

Or use the text editor to add a longer message:

```
git commit -a
```

Note: After writing your message push CTRL X, Y and ENTER to save and exit the text editor.

8) Then you will switch back to your group's branch:

```
git checkout shared_functions
```

9) You will then merge the completed branch with your master:

```
git merge random_password
```

10) If you are completely done and tested that branch you can delete it:

```
git branch -d random_password
```

11) And finally you can push your code to your group's remote branch:

```
git push origin shared_functions
```

As a group you will be responsible to push your code to the master branch if your code is tested and stable. That process will include the following:

1) If you don't have a branch named after your group then create one:

```
git branch shared_functions
```

2) Checkout the group's branch:

```
git checkout shared_functions
```

3) Pull the most recent stable version from your group:

```
git pull origin shared_functions
```

4) Checkout the master branch:

```
git checkout master
```

5) Make sure your master has the most recent code:

```
git pull origin master
```

6) Merge with your group's code:

```
git merge shared_functions
```

7) If there are any conflicts, correct them and then commit:

```
git commit -a
```

Note: After committing a merge GIT will write a message for you and then you can push CTRL X to save and exit the text editor.

8) Push code to the master:

```
git push origin master
```

9) Then update your groups banch:

```
git checkout shared_functions
git pull origin master
git push origin sharted_functions
```

Useful Commands

Here are a handful of other useful commands that may be needed:

1) index.loc issues

If at one point you receive an error regarding the "index.loc" file this means that GIT did not complete an early er process. Just log in to your Webmin, go to Ohter/File Manager and delete the "index.loc" file in the .git directory of your project. Then re-attempt your GIT command.

2) Status

It can be helpful to get your current status at any given time. This command will let you know what files have been changed, deleted, added and the current branch you are on.

```
git status
```

3) Reference Error
   If you are receiving an error regarding the GIT reference here are two possible solutions:

   This command will reset the current branch to the status of the remote repository:

   ```
   git reset --hard HEAD
   ```

   If this does not work you can always delete your branch and start the branch over:
   ```
   git branch -d branch_name
   ```

4) Cache User Credentials
   You may find it tedious to be entering your username and password every time you want to pull or push. This command will cache your credentials for five minutes:

   ```
   git config credential.helper 'cache --timeout=300'
   ```