Configuration Management
Team-3 and Team-4

1.Set up

Create an account in github

2. Install git bash on the system

3. Create username and email

Use the following commands to create username and email:

```
git config --global user.name "Ajay"
git config --global user.email "Ajay"
```

4. Create a git repository

Create a local git repository using the following command

```
git init
```

5. Connect the local git repository to upstream

git remote add origin <<Enter the url to which the repository should be connected>>

For example: git remote add origin https://github.com/akkineni10009/Practicum

https://github.com/TejuGupta/ModelTraceability

6. Configure the proxy settings

Run the following command:

git config --global http.proxy http://proxyuser:Ssn1!Som2@Sase3#@proxy.ssn.net:8080

The following commands are used for unsetting the proxy settings:

```
git config --global --unset http.proxy
git config --global --unset https.proxy
```

7. Check the status of connection

Use the following command the check whether remote origin has been added or not

```
git remote -v
```

8. Modify the url/ Remove remote origin(Necessary only if step.5 is incorrect)

This step is necessary if the Url that is being entered in incorrect in step.5 . The url can be modified using the following command:

git remote set-url origin <<Enter url>>

If you want to remove the origin itself:

git remote rm origin

9. From here on, you could use the command line or eclipse GUI. Eclipse GUI would be easy to use but i would suggest you to know the command line approach which would let you understand the working of git.

10. Make changes to the files

11. Add files to local git

Add the files that are changed to git with the following command

git add .

This command adds all the files that are changed. If you want to add specific files then use the following command:

git add <<filename>>

12. Commit files to local git

Step.11 only marks the files that needs to be committed. The following command actually commits the files that are marked.

git commit -m "<<Enter any message for the commit”

You can add and commit in one step using the following command:

git commit -a -m "<<Enter any message for commit>>"

13. Track status and log

At any point of time, you can view the status and log using the following commands

git status

git log --oneline

14. Push the commit to origin

Use the below command. This would update the changes to the origin hosted in github. You can check the github url to see the modifications once this command is executed successfully.

15. Alternate flow using Eclipse GUI

- Step 11-14 won't be needed if you are planning to use Eclipse GUI.
- Default Git perspective would be there in eclipse. Go to git perspective.
- Within the git perspective, there would be "git staging". Here you would find
- +"unstaged files" and "staged files".
- All the files that are modified would be in "unstaged files". Whatever files you want to commit, drag and drop them in staged files.
- Once the files are dragged and dropped, enter the message and click on "commit and push"
- If this step is successful, go to github origin to see the changes.
- In the git perspective explore other action that can be performed. You can see "git history" which would show the entire history of events happened.

16. Git branches in github

There are 2 branches in github, one is the development branch and the other is the master branch. All the commits would happen to the development branch. From there a new pull request has to be created to push the changes into the master branch. We have set up a code review mechanism for pushing the code to master branch.

17. Committing to the feature branch

Create the branch on your local machine and switch in this branch :

```
git checkout -b [name_of_your_new_branch]
```

Push the branch on github :

```
git push origin [name_of_your_new_branch]
```

When you want to commit something in your branch, be sure to be in your branch. You can see all branches created by using :

```
git branch
```

Add a new remote for your branch :

```
git remote add [name_of_your_remote]
```

Push changes from your commit into your branch :

```
git push [name_of_your_new_remote] [name_of_your_branch]
```

Update your branch when the original branch from official repository has been updated :

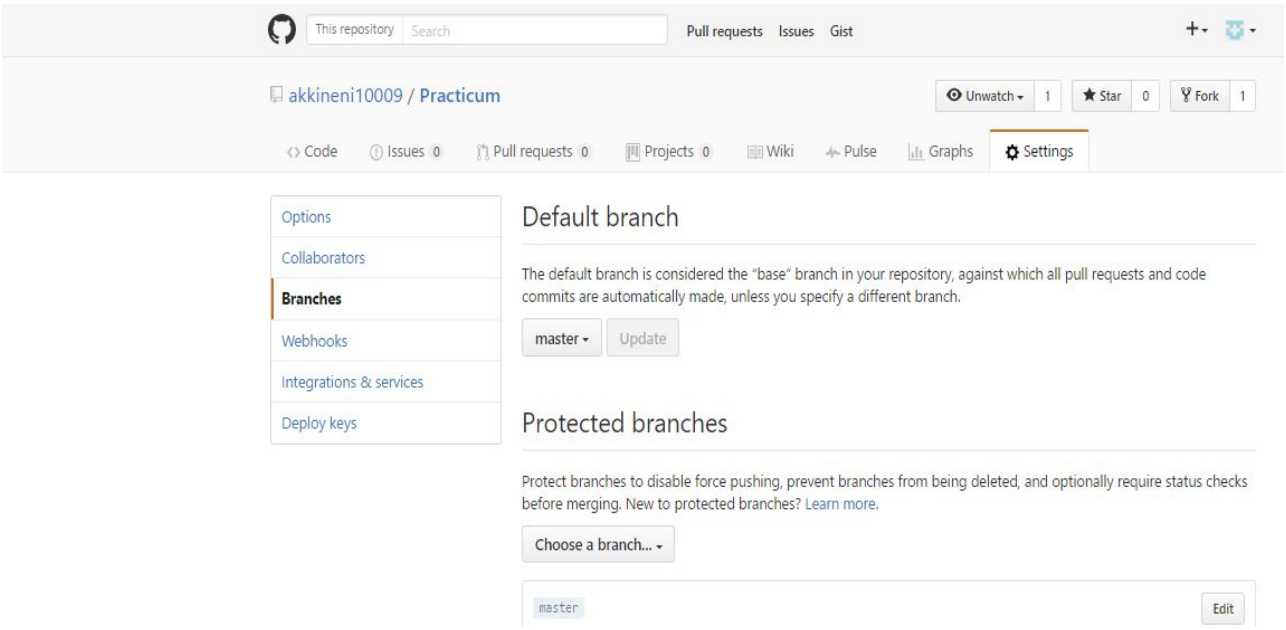
```
git fetch [name_of_your_remote]
```

18. Create a branch in github

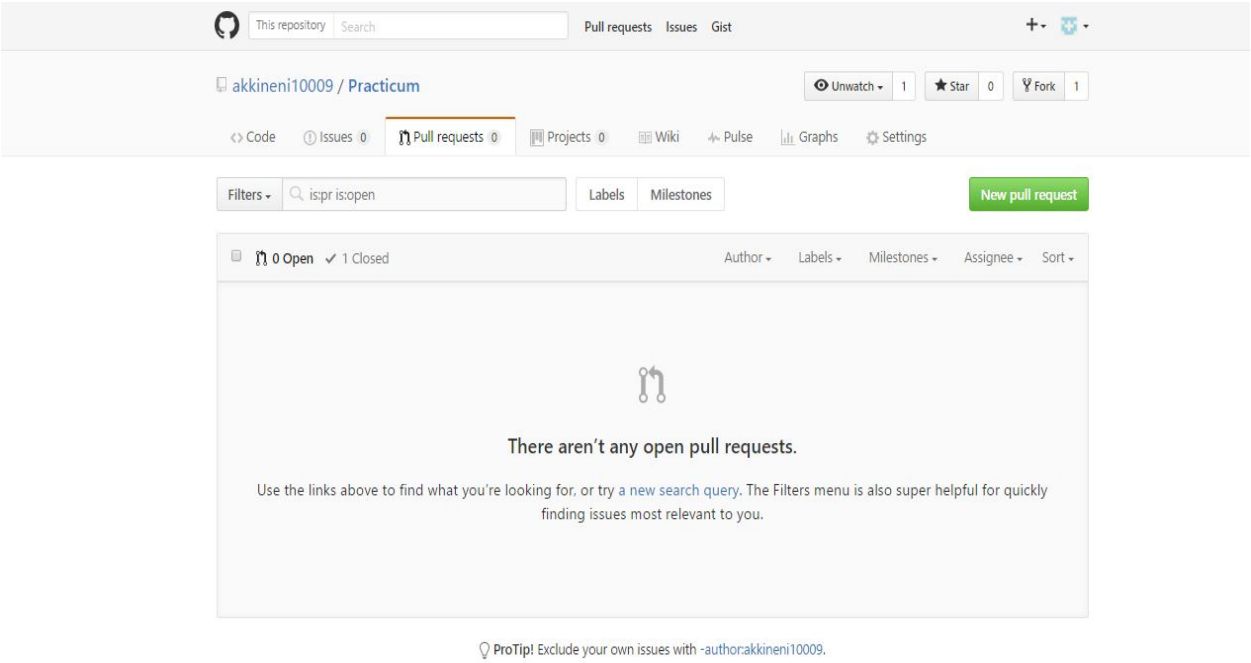
Go to the repository in the github. There would be a button called “Branches”. If you click on the button, you will get a an option to create a new branch. Give a name to the branch. This way you can easily create a branch.

19. Setting up code review

Once the new branch is created, there would be two branches in the repository. One is the master branch and the other is the new feature branch that is created. The master branch would be the production branch and the feature branch would be the development branch. In order to push the code from the development branch to master branch, we have set up code review process. Code review process would be set up. Go to the location shown in the image:



In protected branches, choose the “master” branch. If you want to merge the changes from feature branch into the development branch, you have to create a pull request. The image below shows where to go and create the pull request.



Click on the “New pull requests” and select the base and compare branches. In the “base” select the master branch. In the “compare”, select the development branch. Once the pull request is raised, a email will be sent to the members who were added as the reviewers. Once the review is done by them the changes would go into the master branch.

20. Pull the latest code

Run the following two commands:

```
git fetch origin
```

The following commands also can be used. It will reset the repository. So all local changes would not be available. Save the local changes before running this command.

```
git reset --hard origin/master
```

If nothing works out try this option:

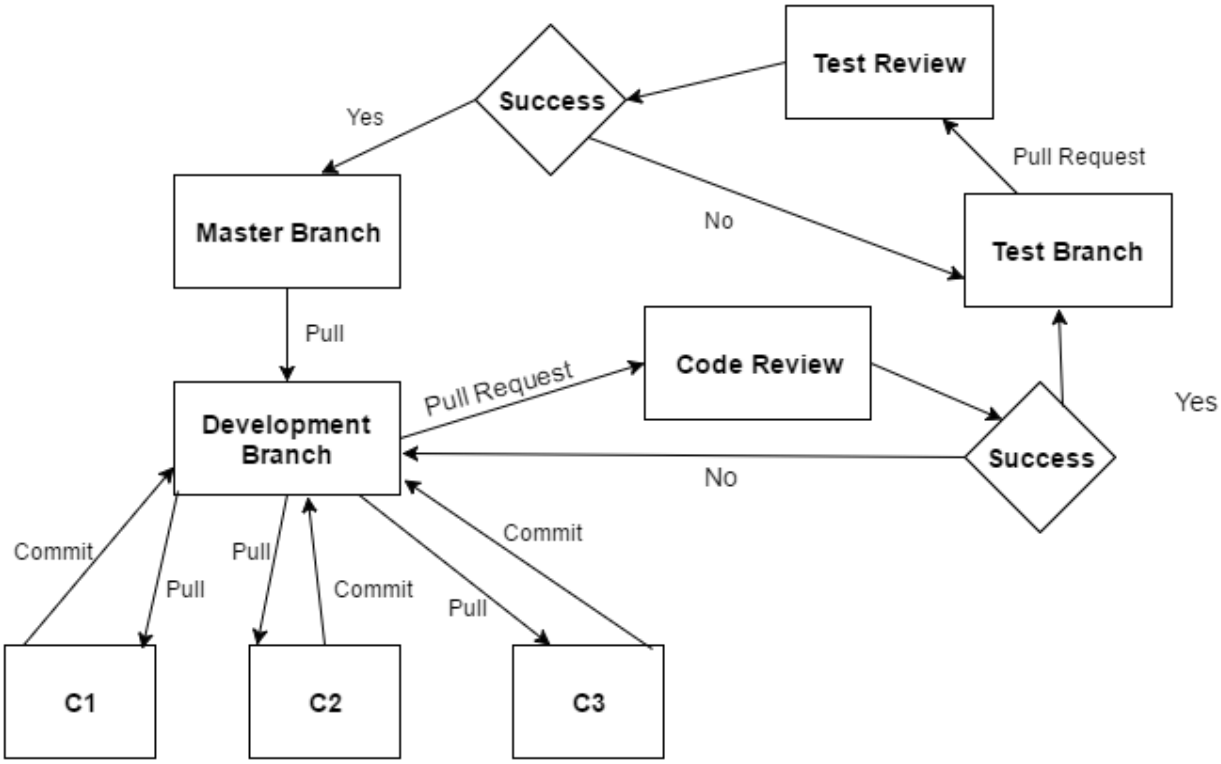
Delete the entire folder and clone again.

21. Merge conflicts and how would you resolve them

Github has documented how to resolve the merge conflicts. We would be using the same approach. The link to the github page is given below:

https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/

22. Configuration Management Model



Sequence:

1. Master branch is the main production branch
2. A copy of the master branch is created and a new branch called Development branch is created. All the developers work on this branch
3. Developers get the latest code by pulling the code from the Development branch
4. Once all the changes are done, the changes are committed to the Development branch
5. From the development branch, the code has to be pushed into the Test branch where the testing would happen
6. In order to push the changes into the Test branch, pull request would be raised. The pull request would have to be reviewed. If the review is successful, the changes would be pushed to the Test branch. Otherwise, the changes would remain in the Development branch.

7. Test branch is used for testing the product. Once the testing is done, the changes have to be pushed to the Master branch, which is the production branch.
8. In order to push the changes into the Master branch, pull request would be raised. The pull request would have to be reviewed. If the review is successful, the changes would be pushed to the Master branch. Otherwise, the changes would remain in the Test branch.