**A Git best practice, the happy scenario ☺**

1. Clone an existing repo

git clone <repo\_URL>

this will create a folder for the repo

1. Switch to the folder with repo name

cd <folder\_name>

1. Check status

git status

1. List current branches

git branch

1. Create a branch

git checkout –b <new\_branch\_name>

1. List branches. Notice your new branch in the list and which branch has the ‘\*’ (current branch)

git branch

1. If your branch is not the current branch, switch to the new branch

git checkout <new\_branch\_name>

1. List branches. Now your new branch should be the current branch (marked with an ‘\*’)

git branch

1. Edit some files or add a new one.
2. Check the status

git status

you will see the files you changed listed as modified. These files are in the working tree.

1. Stage your changes. This will stage all your modified untracked files. This will update the index

git add .

.

1. Check the status. You should see your modified files are ready to commit.

git status

1. If you are ready to commit your changes build succeed and test pass), go ahead and commit your changes (this will update the index)

git commit –m “my test commit”

1. Check the status.

git status

1. Your new branch is still local to your machine. Assuming you do not want to push your branch to the server and that you want to merge your changes to another branch that exists on the server, where others contribute to. Let’s say the you want to merge to the master. First switch to the master branch (the to-be-destination of your merge). You can check the branch list by:

git branch

switch to the merge destination branch

git branch checkout master

1. Make sure you get latest changes from the server.

git pull

1. It is a best practice to first merge the latest into your new branch and make sure everything works

git checkout <new\_branch\_name>

1. Merge the changes you pulled to master into your new branch

git merge master

1. After verifying that everything still works. Switch back to the master branch

git checkout master

1. Now you are ready to merge your changes into master BUT let’s check and make sure nothing was pushed to server repo since your last pull

git pull

if new changes were pushed, you need to repeat steps 17 to 20.

if nothing was pushed since your last pull, do the merge

git merge <new\_branch\_name>

1. Now your changes are merged with latest code on master and ready to be pushed. First get the status

git status

you will see some modified files (due to the merge), stage your changes (add them to the index)

git add .

Do another status. Your changes should be staged and ready to commit

git status

1. Commit your changes (adding them to the local repo)

git commit –m “some comment that describe the changes you made”

1. Check the status. Your changes should be ready to push

git status

1. Now you push your contribution to the server. Yay!

git push

1. Check Bitbucket. You should see your changes.
2. You will delete your new branch and create a new one (if you are switching to a new feature or task). You can continue working on your branch if you are still working toward the same task. To delete a branch use:

git branch –d <name\_of\_branch\_to\_delete>

1. If your new branch was intended to be a “feature” branch that others would contribute to, you would push your new branch to the server before you edit files in step 9. If you push your branch as a feature branch, you may/should create another branch to work on then merge to the feature branch (not to master) and push to server. This allow you to work with the team on the feature branch and merge to master when you are ready.

Please make sure changes on master are pulled into your feature branch on a regular basis. Smaller and frequent merges are recommended as this minimize the need of any huge merge (merge nightmares aka merge hell).

1. You can push your branch to server by:

git push origin <name\_of\_your\_feature\_branch>

1. This was a Git best practice. It may seem much but we really used 5 or 6 commands! Congratulation ☺