Git Merge, Resets and Branches

For this lab you will need a copy of these instructions as well as a git repository. It is ideal to have that repository in a central/remote location such as Github or BItBucket. These instructions assume an editor of Visual Studio Code but can be used with any editor.

Steps to Setup Application

1. Open Visual Studio Code
2. Create a dotnet core console application  
   dotnet new console
3. Create the .gitignore file for dotnet core & favorite editor  
   Note: sometimes editors will have special files that you can ignore…
4. Create git repository in directory  
   git init
5. Commit application with code as it is  
   git add --all  
   git commit -m “<<message>>”
6. Push the code to the cloud or remote server  
   git push origin master
7. Pull code into a separate directory (referred to as DIR2 within)  
   git clone <<repo url>> DIR2
8. Modify code in DIR2
9. Add and commit code to repository  
   git add --all  
   git commit -m “<<message>>”
10. Push code to remote repository  
    git push origin master
11. Return to directory 1 where you initially create the repository

YOU ARE NOW SETUP TO PLAY WITH GIT MERGE, RESET and BRANCHES as done in presentation

Git Merge

1. Make changes to code within Directory 1
2. Add and commit code to repository  
   git add --all  
   git commit -m “<<message>>”
3. Attempt to push code   
   git push origin master

Note: This will fail since directory 1 doesn’t have the change that DIR2 committed. We need to merge the two together

1. Pull down the changes from the repository  
   git pull origin master

Note: pulling will not do anything to your saved (committed changes) it will just fetch the changes on the remote and then try and merge them with your code

1. Resolve any conflicts in code
2. Add and commit code to repository (if no conflicts skip this step)  
   git add --all  
   git commit -m “<<message>>”

Note: Now you should have 2 commits more than the remote 1) your original changes 2) the merge commit combining your changes with remote

Git Reset

1. Make changes to application
2. Add and commit code to repository  
   git add --all  
   git commit -m “<<message>>”
3. Reset changes to staged  
   git reset --soft HEAD~1

Note: HEAD~1 means 1 commit before the HEAD

1. Reset changes to working directory  
   git reset --mixed
2. Reset changes to undone  
   git reset --hard

Git Branch

1. Create a new branch  
   git checkout -b feat1

Note: git checkout in this case with the -b will be used to create the branch and subsequently check it out.

1. Make change to the repository
2. Add and commit code to repository  
   git add --all  
   git commit -m “<<message>>”
3. Git checkout master

Note: the change committed on feat1 is not present on master

1. Merge the change from feat1 to master  
   git merge feat1
2. Delete feat1  
   git branch -d feat1