

## SE 2XB3 Group 4 Report 1

Huang, Kehao  
400235182  
huangk53@mcmaster.ca  
L01

Jiao, Anhao  
400251837  
jaoa3@mcmaster.ca  
L01

Ye, Xunzhou  
400268576  
yex33@mcmaster.ca  
L01

22 January 2021

## Team Contract

- We will primarily be using Discord to communicate.
- All members must respond to messages which they are mentioned in within 2 hours.
- During 3:30pm–5:30pm on Mondays, all members must present in a Discord meeting to prepare for the lab on Tuesday.
- During the assigned lab period on Tuesdays 8:30am–11:20am, all members must present in a Discord meeting to work on the lab.

## Signatures



Anhao Jiao



Kehao Huang



Xunzhou Ye

# 1 Version Control

## 1.1 Experiment 1: Push and Pull

All group members successfully pushed their new files to the remote repository and pulled the files created by other group members from the remote repository. Screenshots can be found in the appendices.

## 1.2 Experiment 2: Revert and Reset

The revert command is a special type of commit which stage and save the inverse of the changes done by the commit specified in the command argument. This reverting changes commit is placed at the end of the commit history chain.

The reset command provides a way to manipulate the `HEAD` pointer. The commonly used reset modes are mixed (default mode when mode flags are not specified), soft, and hard. The reset command invoked in the former two modes moves the `HEAD` pointer to the specified commit, while not changing the files in the current working directory at all, while in the latter mode, local files are reset to exactly like the specified commit.

Reverting should be used when one wants to apply the inverse of a commit from the project history. Resetting should be used when one wants to alter the project history, either compressing multiple commits into one or change part of the commit chain entirely.

## 1.3 Experiment 3: Implementing `are_valid_groups`

Kehao created the `code.py` file and pushed to the repo, all of us pulled the same file and started to work locally. Once we were done coding, Kehao added committed and pushed and file without any conflict. While Xunzhou trying to push the modified file to the repo, conflicts showed up. With the fact that Kehao updated `code.py` before Xunzhou, Xunzhou need to merge his file with the Kehao's version. Xunzhou chose to keep both of their codes and added comments to specify their own versions. Xunzhou resolved the conflict manually, then he added, committed and pushed the merged version. Finally, Anhao encountered the same conflict as Xunzhou did, he also kept all of the codes remained and added comments for his version. After resolving the conflict, Anhao updated the `code.py` and pushed it to the repo. In addition, all of us pulled the newest version to our local machine and discussed the pros and cons about our code. We decided to use Xunzhou's code as our final version, also updated it to the repository. Finally we had the same final version of the `code.py` locally.

## 1.4 Experiment 4: Changing `are_valid_groups`

First of all, the player pulled the remote repo to stay up to date and opened the `code.py` file on his local machine to modify to according to the specification. Meanwhile, the adversaries tried to make some trouble by commenting all the code in the file. Then the adversaries pushed their commit before the player did. When the Player finished the function in `code.py`, he successfully added and committed the file, but encountered a conflict when he tried to push it. Also, the auto-merge does not solve the conflict. To solve the conflict, the player

pulled, and opened the `code.py` file in his local repository. There are a few lines of auto-generated message and symbols indicating the conflict. The player then deleted all irrelevant lines and kept his original version of function. Finally, he added, committed and pushed his local repo and conflicts are resolved.

One of the lessons we learned was that manually solving conflicts by examining the codes and `diff` outputs is tedious. To avoid conflicts, one of the common practices is to always perform a `git pull` before working on any file. Alternatively, each member in a group can work on the project in different branches. It would only require a one-time conflict fix to merge the branches.

# Appendices

## A Experiment 1 Screenshots

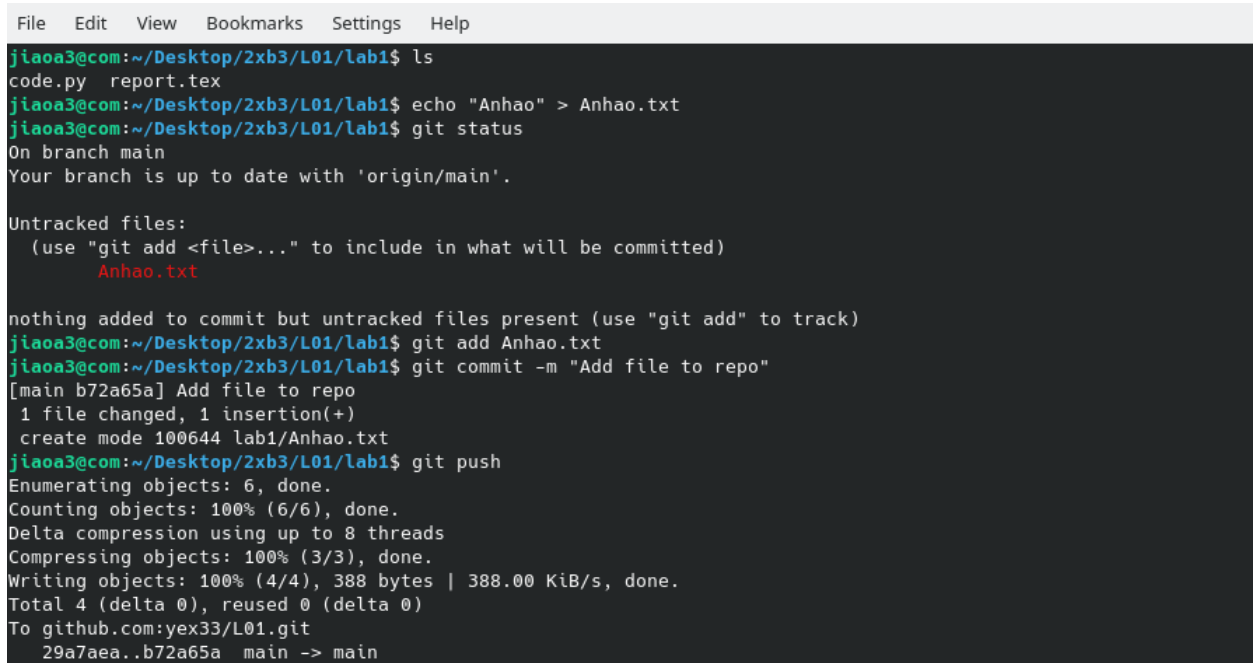
A screenshot of a terminal window with a light gray title bar containing menu items: File, Edit, View, Bookmarks, Settings, and Help. The terminal background is dark gray. The user 'jlaoa3@com' is in the directory '~/Desktop/2xb3/L01/lab1'. They run 'ls' showing 'code.py' and 'report.tex'. Then they run 'echo "Anhao" > Anhao.txt'. Next, they run 'git status', which shows they are on the 'main' branch and that 'Anhao.txt' is an untracked file. They run 'git add Anhao.txt' and then 'git commit -m "Add file to repo"', which shows the file being added to the repository. Finally, they run 'git push', which shows the commit being pushed to the remote repository 'github.com:yex33/L01.git'.

Figure 1: Member 1

```

kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ ls
code.py  report.tex
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ git pull
Warning: Permanently added the RSA host key for IP address '140.82.114.3' to the list of known hosts.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 368 bytes | 368.00 KiB/s, done.
From github.com:yex33/L01
   29a7aea..b72a65a  main      -> origin/main
Updating 29a7aea..b72a65a
Fast-forward
 lab1/Anhao.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 lab1/Anhao.txt
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ ls
Anhao.txt  code.py  report.tex
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ echo "Kehao" > kehao.txt
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
   kehao.txt

nothing added to commit but untracked files present (use "git add" to track)
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ git add *
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ git commit -m "file created"
[main 2313a5f] file created
 1 file changed, 1 insertion(+)
 create mode 100644 lab1/kehao.txt
kehaoh@kehao-kdeneon:~/Desktop/2xb3/L01/lab1$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 421 bytes | 421.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:yex33/L01.git
   b72a65a..2313a5f  main -> main

```

Figure 2: Member 2

```
lab1 : zsh — Konsole
> ls
auto code.py report.aux report.log report.synctex.gz report.tex
> git pull
Enter passphrase for key '/home/joe/.ssh/mac-yex33':
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 1), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), 663 bytes | 663.00 KiB/s, done.
From github.com:yex33/L01
   29a7aea..2313a5f  main      -> origin/main
Updating 29a7aea..2313a5f
Fast-forward
 lab1/Anhao.txt | 1 +
 lab1/kehao.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 lab1/Anhao.txt
 create mode 100644 lab1/kehao.txt
> ls
Anhao.txt auto code.py kehao.txt report.aux report.log report.synctex.gz report.tex
> echo "something" > xunzhou.txt
> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
   xunzhou.txt

nothing added to commit but untracked files present (use "git add" to track)
> git add *
The following paths are ignored by one of your .gitignore files:
lab1/report.aux
lab1/report.log
lab1/report.synctex.gz
Use -f if you really want to add them.
> git commit -m "xunzhou added file"
[main 49e688b] xunzhou added file
 1 file changed, 1 insertion(+)
 create mode 100644 lab1/xunzhou.txt
> git push
Enter passphrase for key '/home/joe/.ssh/mac-yex33':
Enter passphrase for key '/home/joe/.ssh/mac-yex33':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 364 bytes | 364.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:yex33/L01.git
```

Figure 3: Member 3

## B Experiment 2 Screenshot

```
lab1 : zsh — Konsole
> git revert 5f1f9d0
[main f790bdf] Revert "added tex gitignore"
1 file changed, 129 insertions(+), 417 deletions(-)
rewrite .gitignore (62%)
> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    auto/
    report.aux
    report.synctex.gz

nothing added to commit but untracked files present (use "git add" to track)
> git log --oneline
> ls
Anhao.txt  code.py  report.aux  report.synctex.gz  xunzhou.txt
auto      kehao.txt  report.log  report.tex
> git log --oneline
> git log --oneline
> git reset 4e7aabc
Unstaged changes after reset:
M   .gitignore
> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   ../.gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    auto/
    report.aux
    report.synctex.gz

no changes added to commit (use "git add" and/or "git commit -a")
> git log --oneline
> git log --oneline
> git reset 4e7aabc --hard
HEAD is now at 4e7aabc updated anhao version
> git status
On branch main
Your branch is up to date with 'origin/main'.
```