## **Session 1 – SCM & Git Commands**

## **Activity 1.**

Research what are the differences between **Configuration Management** (CM) and **Version Control**? You may Google it or look at the following web sites <https://en.wikipedia.org/wiki/Software_configuration_management>.

Write some notes on each of the following:

1. For **Configuration Management**, [https://en.wikipedia.org/wiki/Configuration\_item](https://en.wikipedia.org/wiki/Configuration_item%20) , find out what are:
   * the configuration items?
2. Research what is **Software Configuration Management**, look at **Watts Humphrey** comments on:

* Simultaneous Update
* Shared Code
* Common codes
* Version control

1. For **Disciplines, Version Control,** find out what are the **CM and version control tools**?

<http://www.sitepoint.com/version-control-software-2014-what-options/>.

List 3 tools.

Do you see **Git**, **CollabNet – Subversion, Mercurial** there?

## **Activity 2.**

# Visit: http://sourceforge.net/about and http://github.com

# Write some notes on an overview of **Open Source Software development** likeSourceForge or GitHub**.**

1. Who are the main users of this site?

## **Activity 3.**

Write some notes about how do you organise a software development team? From the following site find out what is a **vertical team**, **horizontal team** vs **hybrid team**.

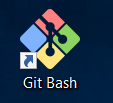
<http://www.databaseskill.com/1308134/>.

## **Activity 4. (Using Git)**

# Check whether your computer has been installed with Git by pressing 🡪 **Control Panel** 🡪 **Programs** 🡪 **Programs and Features**, do you see Git version 2.11.xx installed?

*If not, you should be able to find the Git installation file from L:\its\files\...\git\Git-2.11.0-64-bit.exe and install it into your computer.*

After the installation, you need to customize your Git environment. In Windows 10, the configuration is stored in the C:\Users\yourname**\.gitconfig** file.

On the desktop, you should be able to find the **Git Bash** icon.  **Git Bash** for windows is a BASH emulation to run Git from the command line. It runs similarly in the Linux command line environment.

Click on the icon will open a command prompt for entering the **git bash** commands.

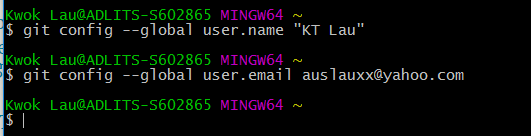
To add the user details, in the **git bash** prompt, type your name and email with the syntax:

$ git config --global user.name “*yourname*”

Bash means Bourne Again Shell in Linux.

$ git config --global user.email [youremail@xxxxxx.com.au](mailto:youremail@xxxxxx.com.au)

*e.g.*

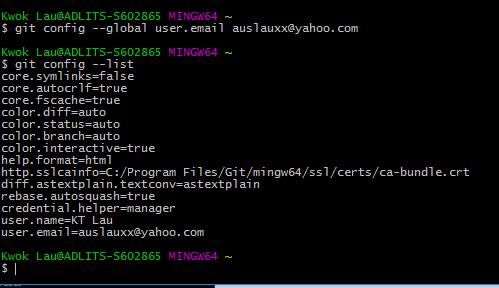


Note: the git bash commands are case sensitive.

Checking your settings

$ git config --list

or $ git config -l



You can also display the Git specific key’s value by typing git config <key> e.g.

$ git config user.name

Two ways to get help while using Git

1st way:

$ git help <verb>

e.g.

$ hit help commit

$ git help config

$ git help checkout

**2nd way:**

$ git <verb> --help

e.g.

$ git commit –help

$ git checkout help

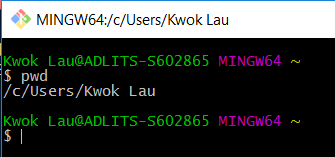
**Exercise 1.**

**Aim:** This exercise is to set up a local repository called **MyRepos** and version control a file called **readme.txt** in it.

After the installation, you should be able to see a Git Bash  icon in the desktop. Click on the Git Bash to start the Git Bash command prompt. Type the following commands:

Display the current directory:

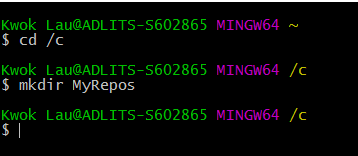
$ pwd



Change directory to c:\ and create a directory called MyRepos

$ cd /c

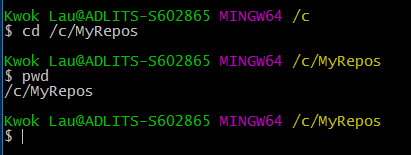
$ mkdir MyRepos



Change the current directory c:\MyRepos and check it make sure that the current directory is c:\MyRepos.

$ cd /c/MyRepos

$ pwd



List all files in the current directory. -l switch means show the details information of each files.

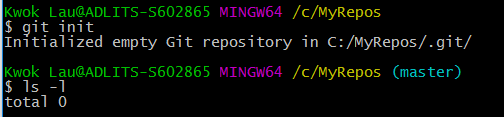
$ ls -l

The directory is empty, you should not see any files or directory yet.

Create a local repository using the git init command on the current directory.

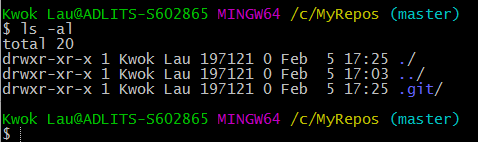
$ git init

$ ls -l



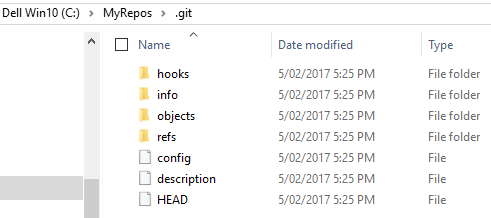
The repository has created but you still not seeing the files. However, there are some hidden files for the repository has been created. If you use the ls -al switch, you should be able to see some files. The -a is to display the hidden files and folders.

$ ls -al

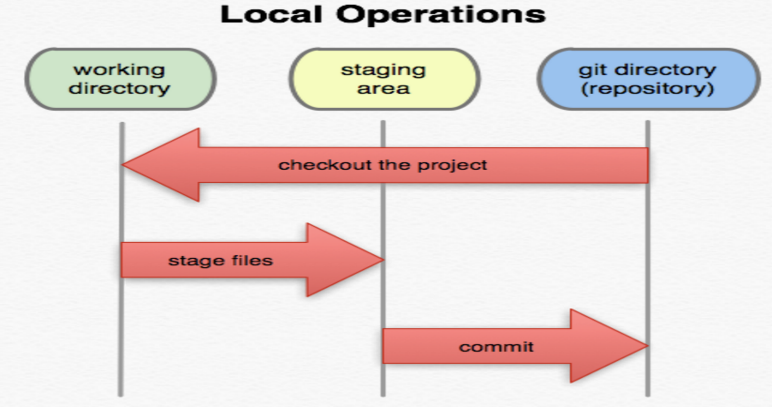


Use the File Explorer to visit the C:\MyRepos folder, you should be able to see:

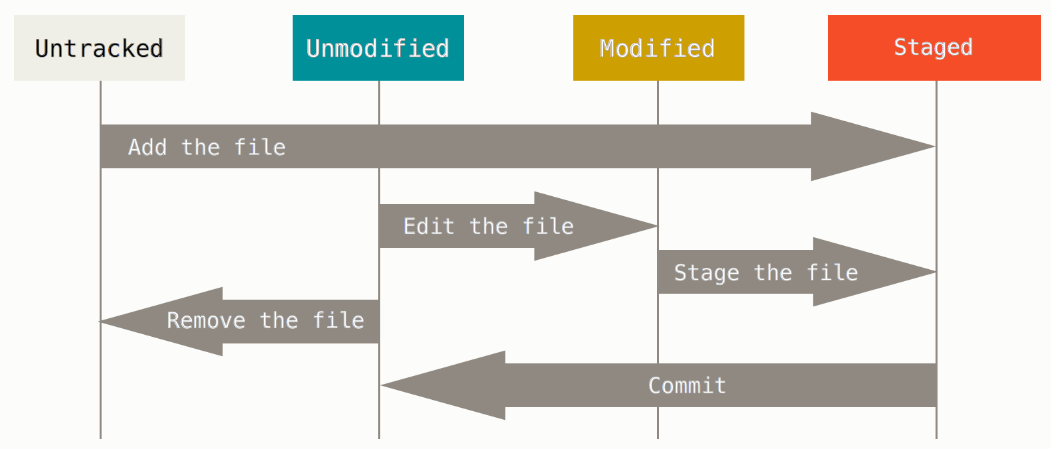
.git folder and the sub folders.



The git repository is a database which stores different versions of the files and folders.

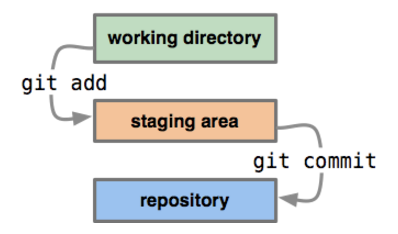


Staging files



The life cycle of the status of the files

**Git add** adds your modified files to the queue *to be committed later*. Files are not committed. Git add adds files to the Git index, which is a staging area for objects prepared to be committed. Add tells git to start tracking a file.  
  
**Git commit**commits the files that have been added and creates a new revision with a log. If you do not add any files, git will not commit anything. You can combine both actions with git commit -a. Git commit commits the files in the index to the repository, git commit -a is a shortcut to add all the modified tracked files to the index first. Commit commits your current changes on your local repository.



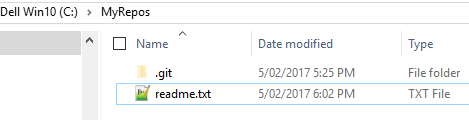
<https://www.youtube.com/watch?v=GGYq3UAIrzM>

Use notepad to create a file **readme.txt** inside the **c:\MyRepos** directory with the following contents:

This is the first line.

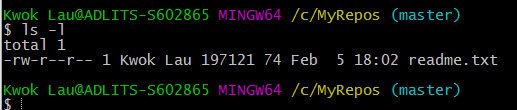
This is the second line.

This is the third line.



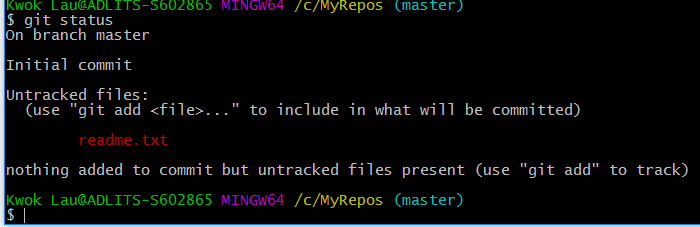
Display the files in the **MyRepos** directory.

$ ls -l



Check the status of your files.

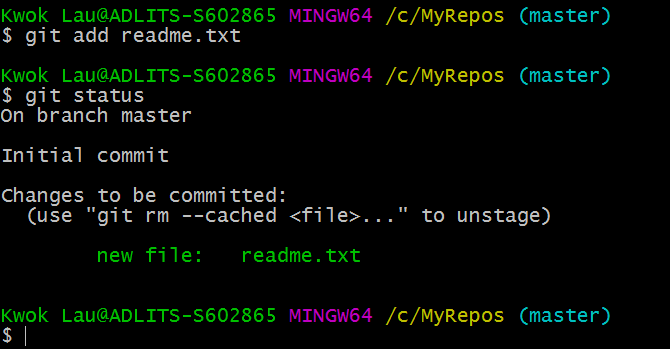
$ git status i.e. display a list of files modified since last commit



At the moment, the **readme.txt** is untracked (it is shown in red). You need to use **git add** to stage for tracking a specific file and use git commit to commit the changes in the repository.

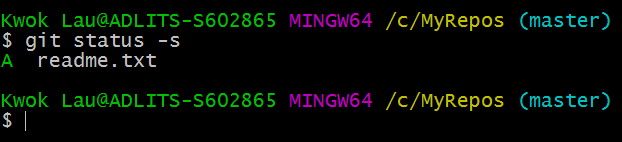
$ git add readme.txt

$ git status i.e. display a list of files modified since last commit



The git status output is very comprehensive, you may want to checking the status and display in a short/simpler form.

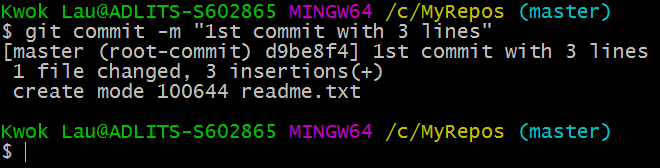
$ git status -s



Now commit the changes into the repository.

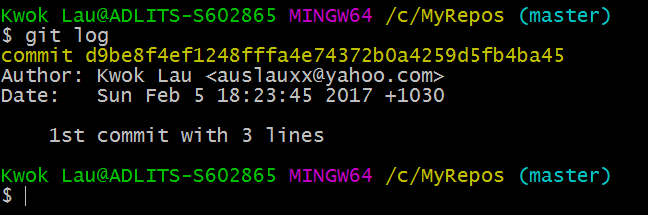
$ git commit -m "1st commit with 3 lines"

The -m switch stands for message.



Display the commit log history.

$ git log

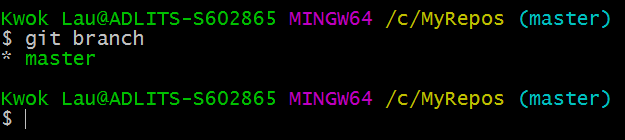


Note: the commit index is a 40-byte hexadecimal number as the commit object name.

It also displays who was the author and the date/time it commits.

Display all the branch names in the repository.

$ git branch



At the moment, there is only one branch called **master**.

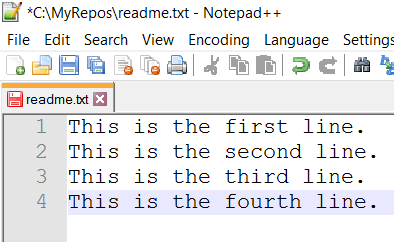
Check out the master for further modifying the **readme.txt** file. Note: The current branch by default is on the **master** branch even there is no other branch. In future, we are going the create other branches in future sessions.

$ git checkout

*Git checkout means updates files in the working tree to match the version in the index or the specified tree. If no paths are given,*git checkout*will also update HEAD to set the specified branch as the current branch. As earlier, the current branch is master.*

Use the notepad to modify the **readme.txt** file by add an extra line.

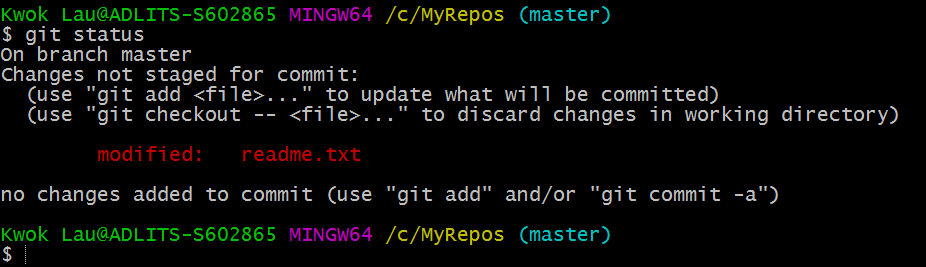
**This is the fourth line.**



Save the file.

Display the status of the working tree (i.e. master). i.e. files/directories changes

$ git status i.e. display a list of files modified since last commit



$ git add -A i.e. stage the file for modification. -A switch means stage all files in the current directory.

Note: sometimes certain files you don’t want to add i.e. by using .gitignore

$ git status i.e. display a list of files modified since last commit of the working tree.

The -A switch means all files modified. You can specify specific file or use wild card characters as file path.

e.g. $ git add readme.txt *add specify file*

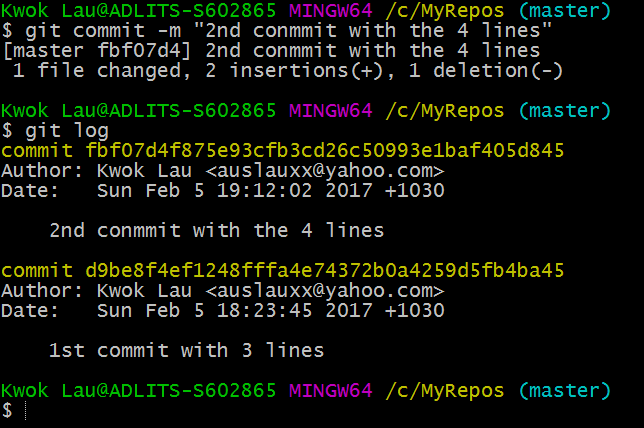
$ git add . . *means all files in the current directory*



Now commit the changes to the repository.

$ git commit -m "2nd commit with the 4 lines" i.e. commit the changes and specify a message

$ git log i.e. show the change log history



You can see that it has committed twice with specific index. Write down your two 40 characters index.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The commit order is in reverse chronological order. i.e. the most recent commits show up first.

Since the readme.txt is a text file, you can display the content by:

$ cat readme.txt i.e. cat is a Linux command to display the content of a text file.

You can display the specific version back to the 1st commit by using the index or hash.

$ git show d9be8 i.e. show the revision of d9be8……….

Use the first 5 characters in your index.5 digits

Just type the first 5 or 7 characters is unique enough

**Exercise 2.**

**Aim:** Create another repository and version control a C# program.

Click on the Git Bash  icon in the desktop to start the **Git Bash** command prompt. Type the following commands:

$ pwd i.e. display the current directory.

The default should be at /c/user/yourname

$ cd /c i.e. change the current directory to c:\

$ mkdir /c/NewRepos i.e. create a new directory c:\NewRepos

$ cd /c/NewRepos i.e. change directory to c:\NewRepos

$ pwd i.e. Display the current directory

$ git init i.e. create a new repository

Use File Explorer to copy the **MyInitial** C# program into the **c:\NewRepos** directory. i.e. Put the Visual Studio project into the repository. Use file Explorer to see what is in the folder. i.e. your working directory and the .git repository

$ git add -A i.e. stage all files for commit. -A means all files and directories.

$ git status i.e. display a list of new/modified files since last commit.

$ git commit -am "Initial loading of the program" i.e. -a switch means all files. -m switch means the commit message.

$ git status i.e. display a list of files modified since last commit.

$ git log i.e. show the commit log history (checksum, author, email, date/time,

commit message). The index is the SHA-1 checksum. (See 1st commit).

$ git log --stat i.e. --stat switch shows more details about the file changes.

Use Visual Studio to modify the **JuiceBarForm** so that it calculates the items with the following prices.

**12 Ounces 5.00**

**16 Ounces 6.00**

**20 Ounces 7.00**

Rebuild, Run & Save the changes.

$ git status i.e. display again a list of files modified since last commit.

$ git add -A i.e. add the modified and new files in the current directory and sub-

directories to the staging area. -A switch means all files/directories.

$ git commit -am "Price modified " i.e. puts your all changes into your local repository.

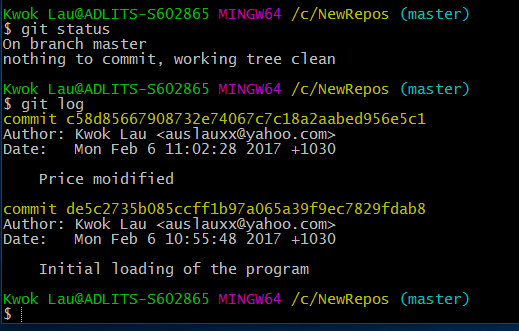
$ git status i.e. display again a list of files modified since the last commit.

$ git log i.e. show the commit log history (See 1st & 2nd commit)

Write down your git hash/indexes in the reverse chronological order.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Yours hash/index value would be different.

Now use Visual Studio to modify the **JuiceBarForm** so that it calculates the items with the following prices.

**12 Ounces 10.00**

**16 Ounces 12.00**

**20 Ounces 14.00**

Rebuild, Run & Save the changes.

$ git status i.e. shows what are the changes in the working directory.

**How to discard changes if files have not been staged?**

What happen if you want to discard this changes? **Note: these files have not been add/staged for modification yet.** You may use the following command to discard the changes.

$ git checkout -- . i.e. the -- means no option. The . means the current directory.

*The git checkout updates files in the working tree to match the version in the index or the specified tree. If no paths are given,*git checkout*will also update HEAD to set the specified branch as the current branch. Another word, in here, it updates the files back to the HEAD (the latest revision).*

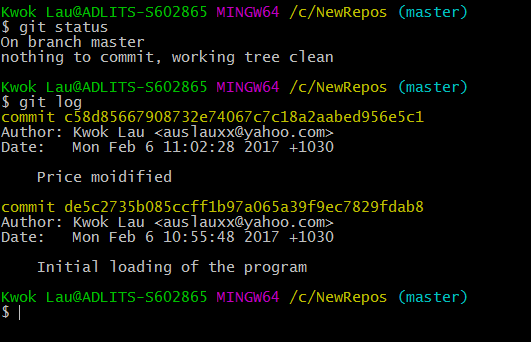
$ git status Note: The files went back to “Price modified” i.e.

12 Ounces 5.00

16 Ounces 6.00

20 Ounces 7.00

$ git log i.e. see the log history



How do you know the changes have been discarded?

This time instead of using the Visual Studio to open the solution files, use the **Notepad++** to check whether the **JuiceBarForm.cs** file had the5.00m, 6.00m & 7.00m in the codes. Note: do not change the file at all in here. We just want to confirm the content of the file.

Or instead of using Notepad++, you may use the Linux **cat** command to display the content of the file.

$ cat /c/NewRepos/MyInitial/JuiceBarForm.cs

**How to discard changes if the files have been staged?**

What about if we want to discard the modified files if they have been added or stages? We will need to use the **git revert** command instead.

Now use Visual Studio to modify again on the **JuiceBarForm** so that it calculates the items with the following prices as you have done it before.

**12 Ounces 10.00**

**16 Ounces 12.00**

**20 Ounces 14.00**

Rebuild, Run & Save the changes.

$ git status i.e. check the status

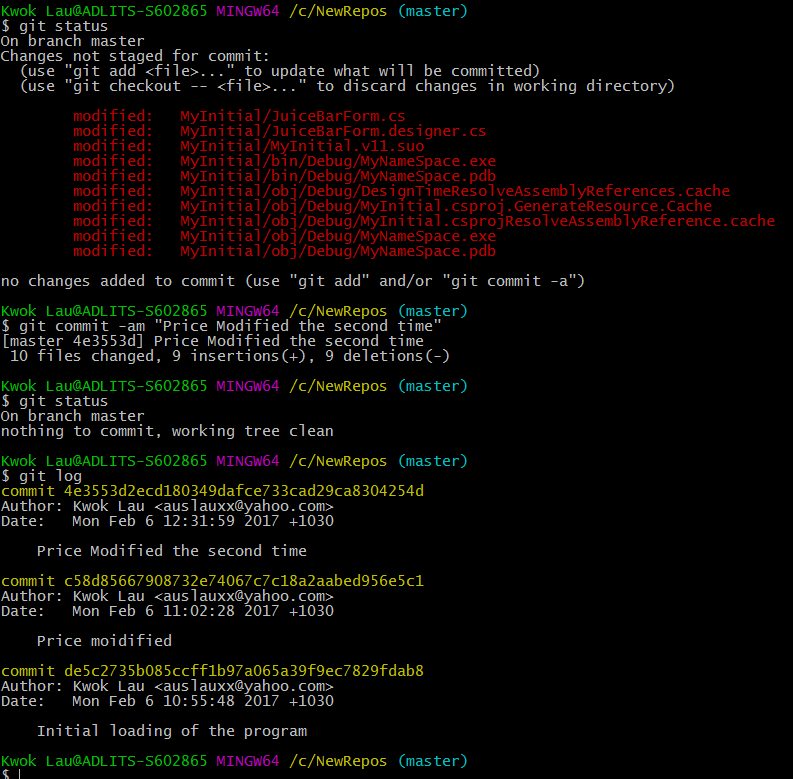
This time we will stage the files for modification and also will commit the changes to the repository.

$ git add -A

$ git commit -am “Price modified the second time”

$ git status i.e. check the status again

$ git log i.e. see the log history



Note: the last commit index was 4e3553d2ecd180349dafce733cad29ca8304254d

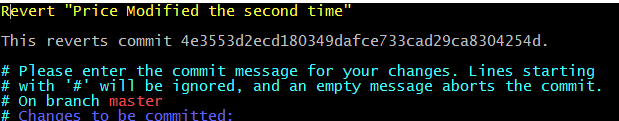
$ git log -1 i.e. To display the status of the last commit

$ git show 4e355 i.e. display the latest committed changes

Use your last git commit index here. The first 5 characters will do.

Type **q** to quit from the show info.

$ git revert 4e355 i.e. discard the changes on 4e355

:

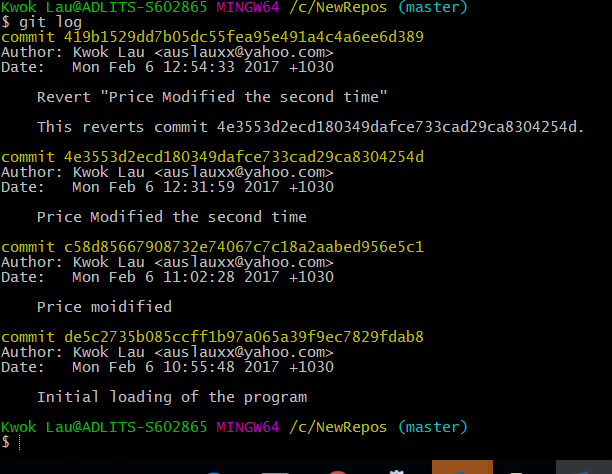
You may take the default comments. It includes the comments of the revert by default. You can add more comments if you want.

Type **:wq** to save and quit from the editor. i.e. write and quit.

Note: It also committed the revert.

$ git log

You should be able to see another commit which the revert is regarded as another commit.



It says “Reverted the 4e355” and has a new commit index.

Use the **Notepad++** to check whether the **JuiceBarForm.cs** file had the5.00m, 6.00m & 7.00m in the codes. Note: do not change the file at all in here. We just want to confirm the content of the file.

Or instead of using Notepad++, you may use the linux **cat** command to display the content of the file.

$ cat /c/NewRepos/MyInitial/JuiceBarForm.cs

You should confirm whether it is reverted or not.

Finally, you can reset the current HEAD to the specified state.

e.g.

$ git reset --hard c58d856 i.e. change the head to a particular index.

Note: this is an irreversible process. By doing this it discards all previous changes after C58d856 in the log history.

**Other Resources:**

1. Git - Introduction

<https://www.youtube.com/watch?v=Kp5BSBoOw8k>

1. Git Reverting / Reseting

<https://www.youtube.com/watch?v=u-kAeG4jkMA>

1. Git – interactive learning tools

<http://try.github.io>