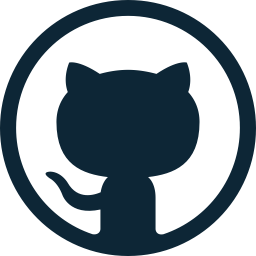
Assignment # 1

Junaid Iqbal Raja

Roll no 153179

BSCS V

Software Engineering



Git vs GitHub

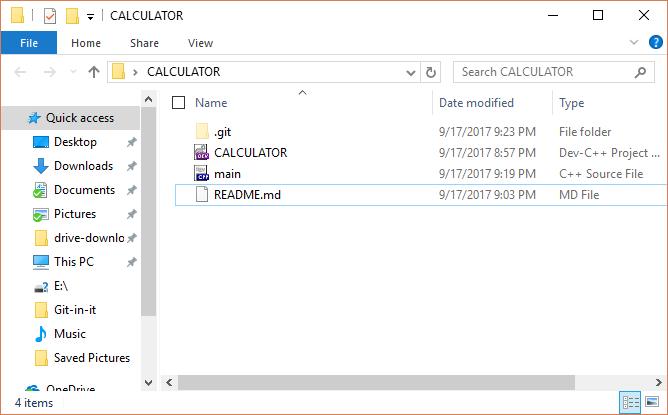
https://github.com/junaidspeaks

Supervisor: Sir Ahmad Mohsin

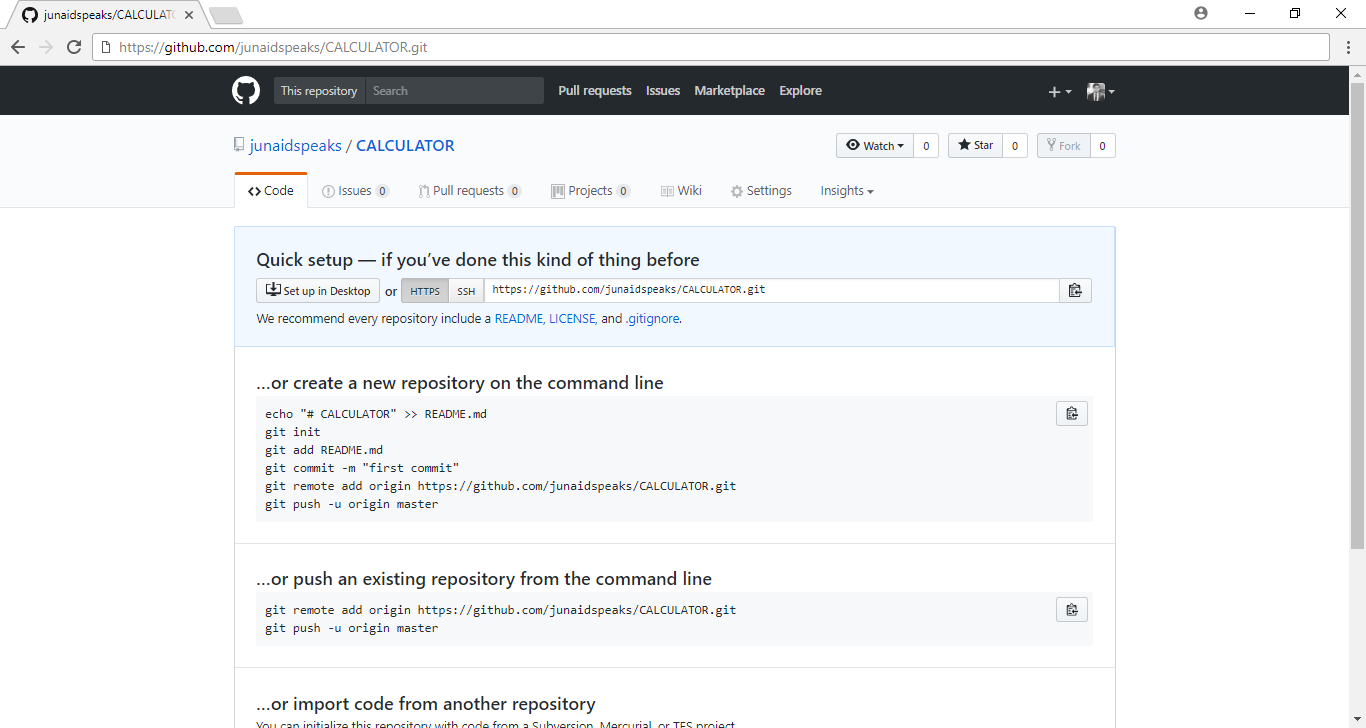
Air University, Multan Cantt

**EXERCISE-0**

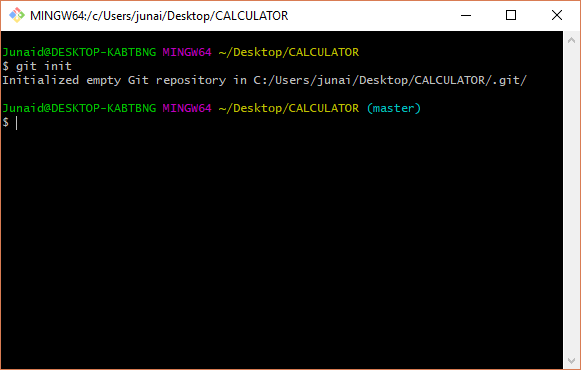
**Step 1**: Create an A C++ PROJECT IN DEV OR CODEBLOCKS. I called mine CALCULATOR.



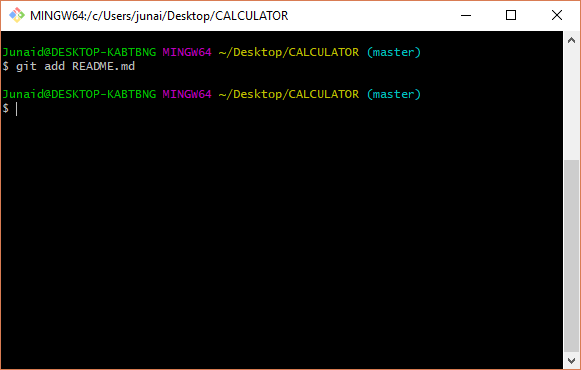
**Step 2**: Create a repository on GitHub of the same name.



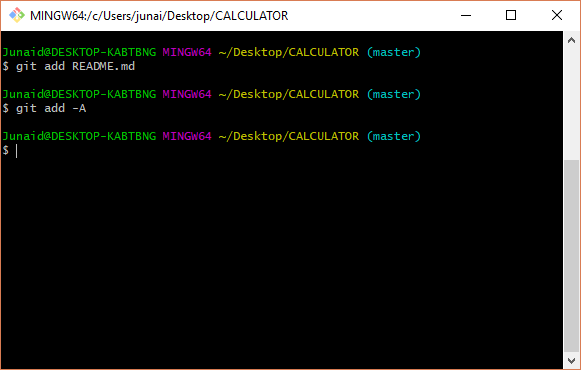
**$ git init**



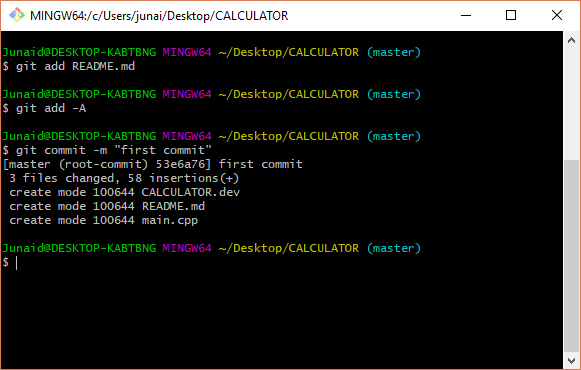
**$ git add README.md**



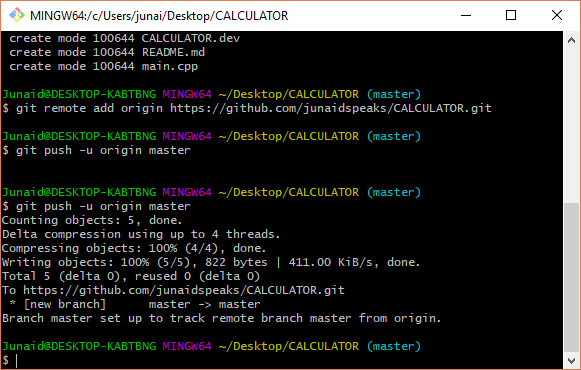
**$ git add –A**



**$ git commit –m “first commit”**

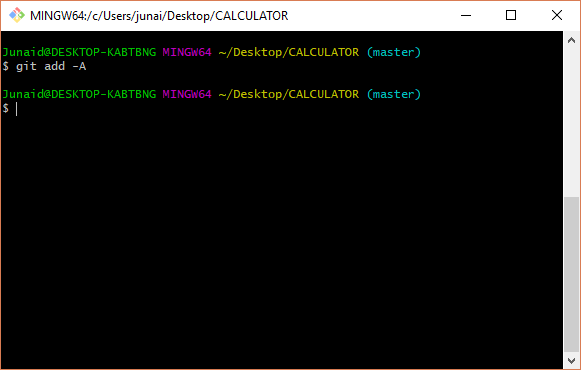


**$ git push –u origin master**

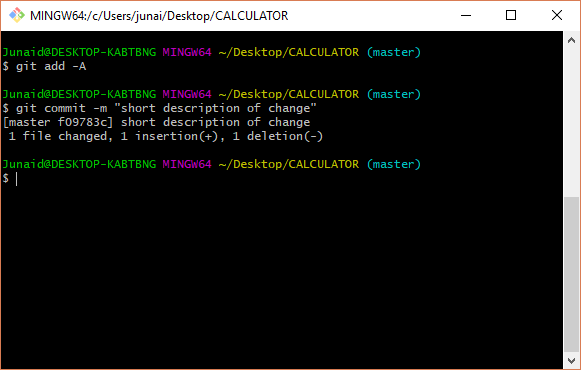


**After a change to your project:**

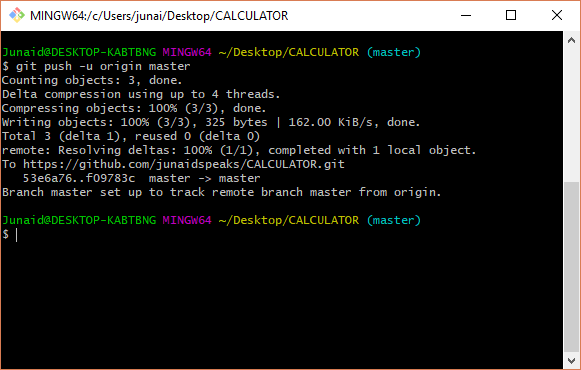
**$ git add –A**

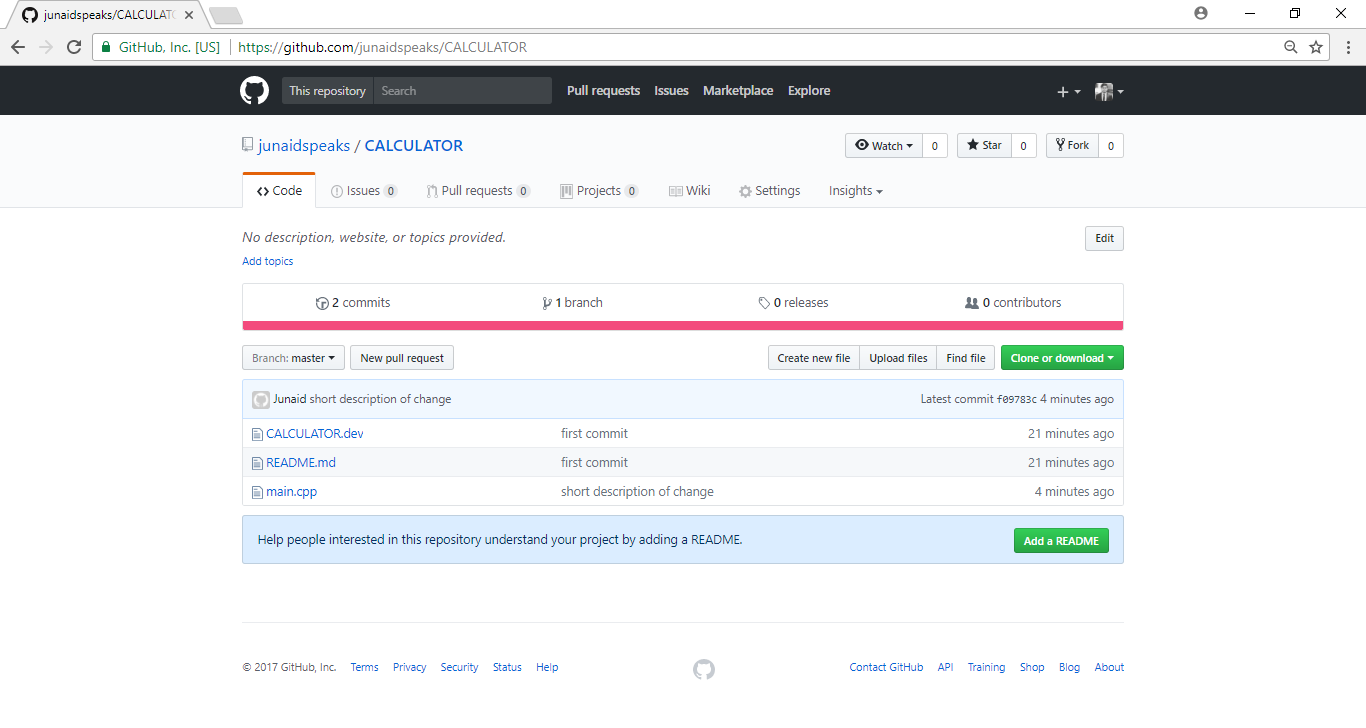


**$ git commit –m “short description”**



**$ git push –u origin master**

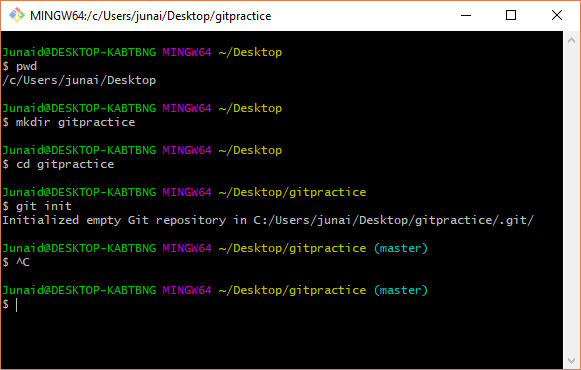




**EXECISE-1**

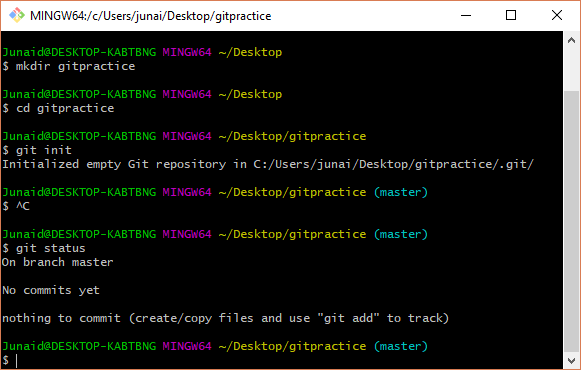
**Start tracking files and subfolders with git:**

**$ git init**



Run git status to see current status of the repository:

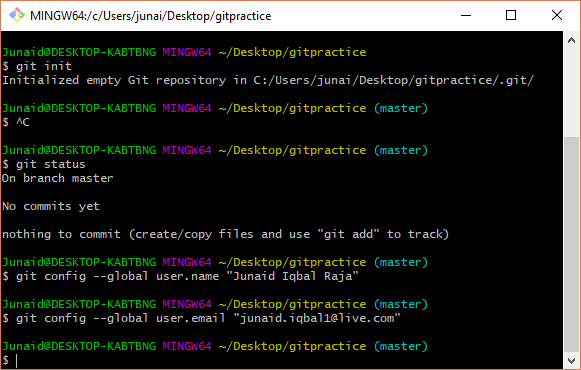
**$ git status**



The first time you use the shell, it is a good idea to set up your identify for committing changes. To set your identity, enter:

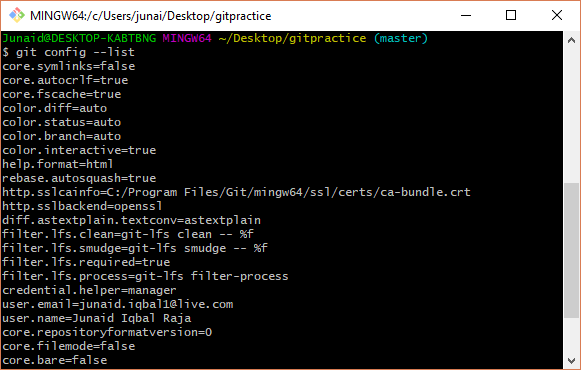
**$ git config --global user.name "Junaid Iqbal Raja"**

**$ git config --global user.email “Junaid.iqbal1@live.com”**



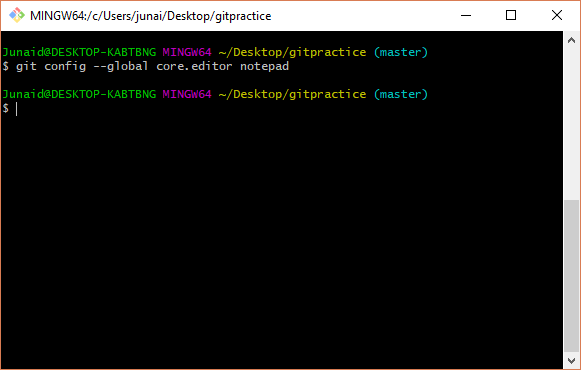
Check values you just set:

**$ git config –list**



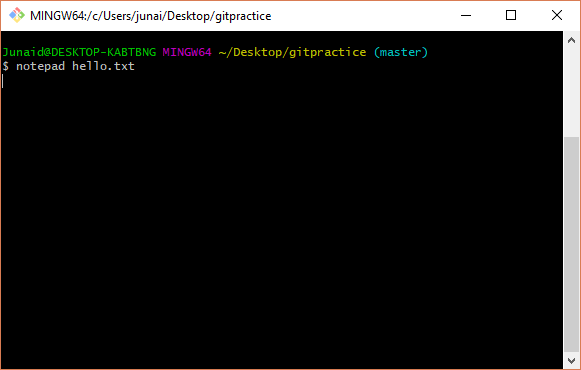
To change the editor to notepad, enter:

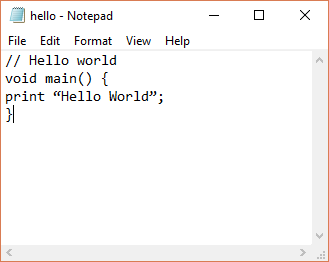
**$ git config --global core.editor notepad**



Add a new file to the directory. For example, use notepad to create a file:

**$ notepad hello.txt**

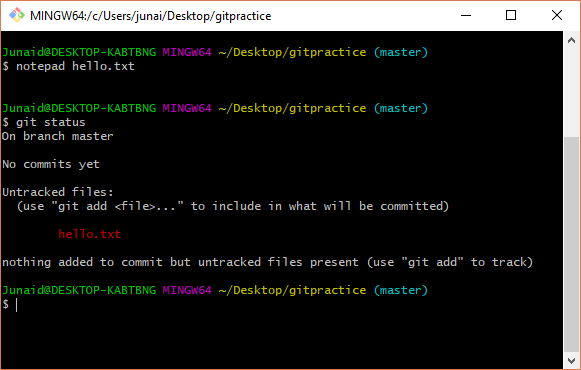




Run git status:

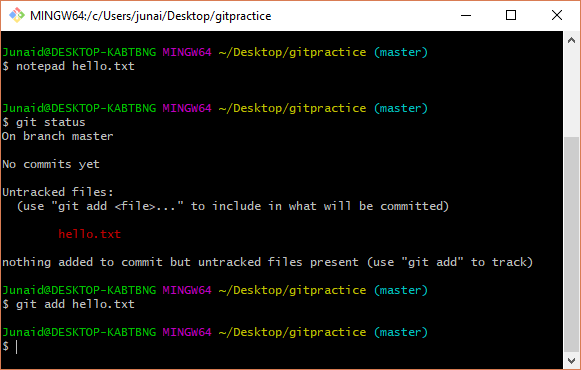
**$ git status**

You should see the file as “untracked”.



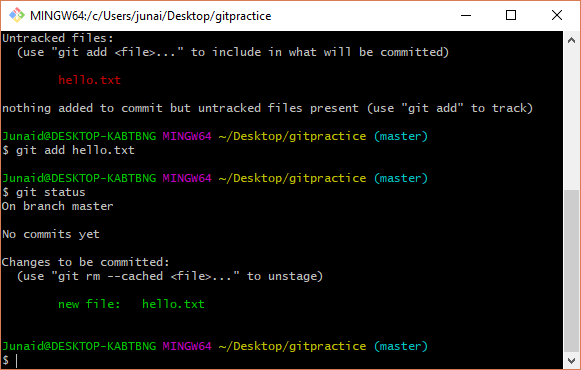
Add the file with:

**$ git add hello.txt**



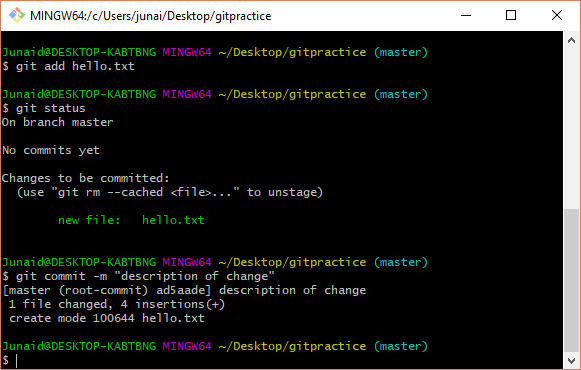
Run git status to verify the file was staged:

**$ git status**



To commit all changes that have been added, enter:

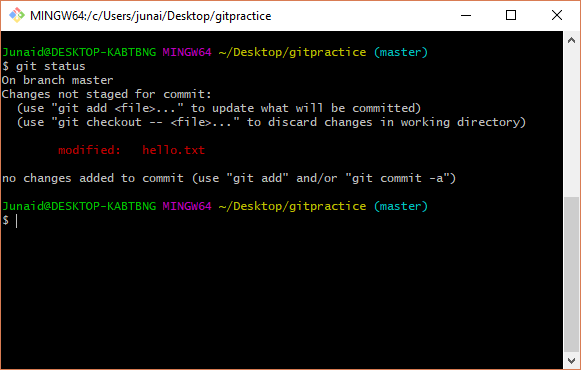
**$ git commit –m ‘description of change’**



To verify the commit was successful, run git status:

**$ git status**

You should see the status of the file as changed but not staged.



Stage it again with:

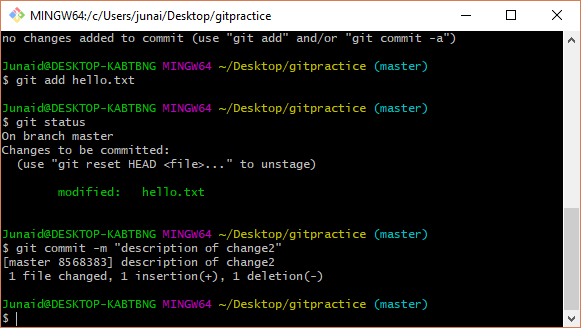
**$ git add hello.txt**

Run git status:

**$ git status**

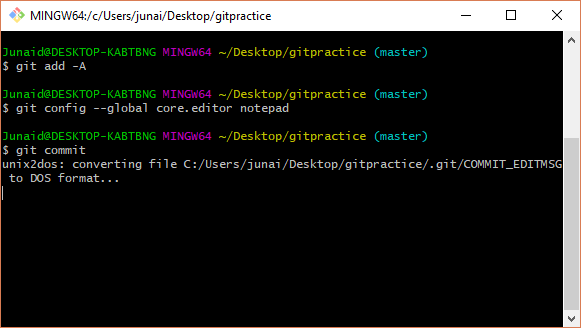
You should see the status of the file as staged but not committed. Now, commit the changes with:

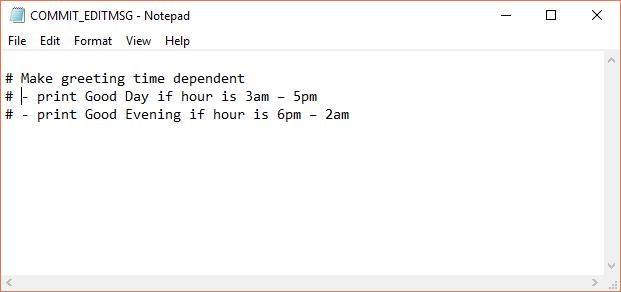
**$ git commit –m ‘description of change2’**



The best way to enter a multi-line comment, is to leave the –m option off the commit. The following will open the default editor for the commit message:

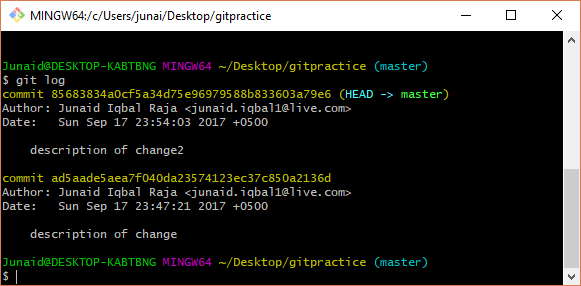
**$ git commit**





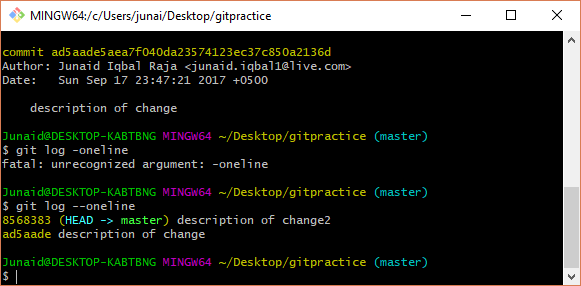
To see a history of changes:

**$ git log**



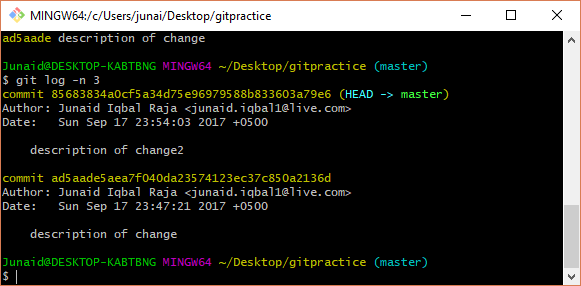
To see an abbreviated history:

**$ git log –oneline**



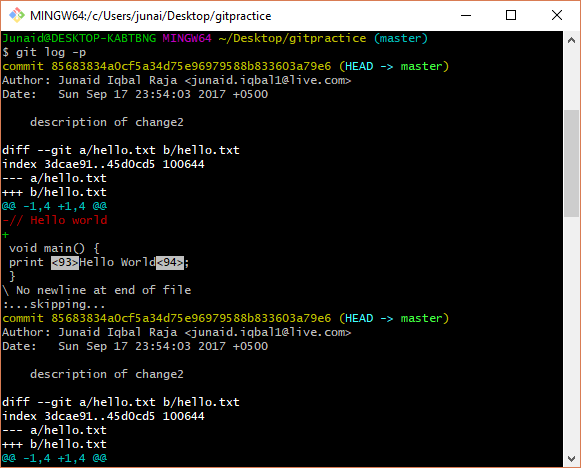
You can also, limit the number of previous commits to show. The following will show the last 3 commits:

**$ git log –n 3**



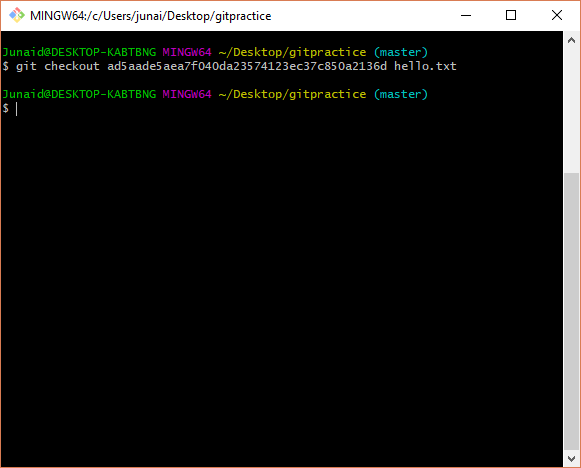
To show the differences introduced in each commit, use the –p option:

**$ git log –p**



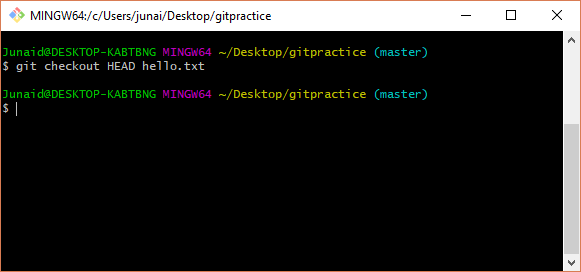
You can also checkout a single file from a previous commit:

**$ git checkout <commit hash> filename**



To get back to the latest version of the file, enter:

**$ git checkout HEAD filename**



A .gitignore file should be committed into your repository (add followed by commit), in order to share the ignore rules with any other users that clone the repository.

Use notepad to create a .gitignore file with the following contents:

// Ignore .class files and everything

// in the bin directory

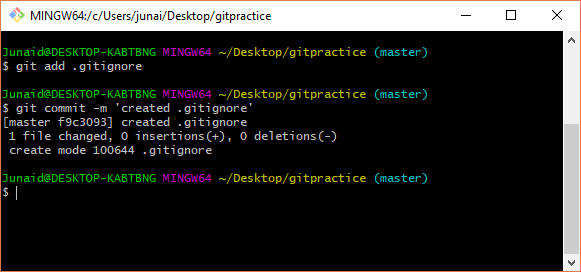
\*.class

bin/

Add and commit the .gitignore file to your project:

**$ git add .gitignore**

**$ git commit –m ‘created .gitignore’**

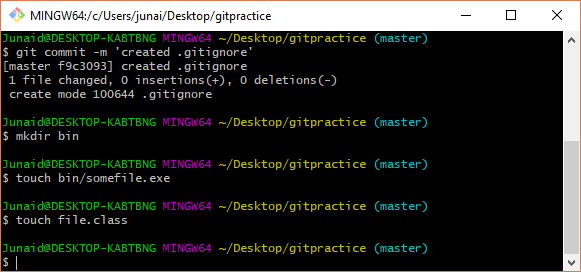


Now, create a bin directory, add a file to it and create a .class file:

**$ mkdir bin**

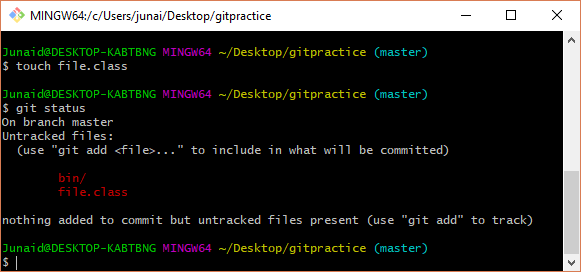
**$ touch bin/somefile.exe**

**$ touch file.class**



Now, check git status and notice these files are ignored (git doesn’t flag these new files as untracked):

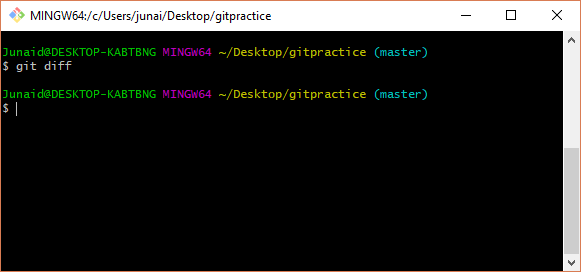
**$ git status**



For more details on what has changed, you need git diff. git diff will tell you (1) what is changed but not staged, and (2) what is staged and about to be committed. Unlike git status, git diff shows exact lines changed/added/removed.

The following shows changed but not added:

**$ git diff**

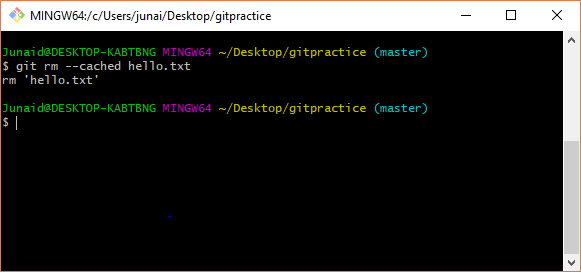


The following shows exactly what will be added in the next commit (exact lines of each file):

**$ git diff –staged**

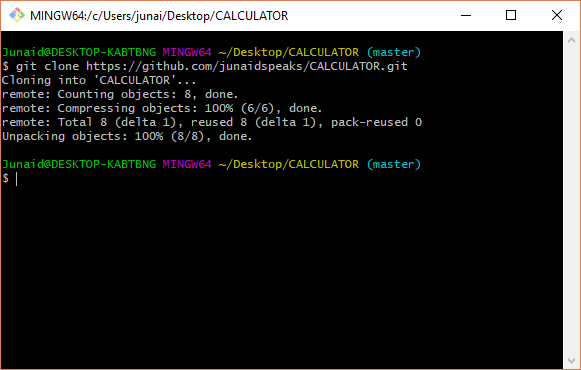
git rm removes the file from the directory and causes git to stop tracking it. If you just want git to stop tracking it, use:

**$ git rm --cached**

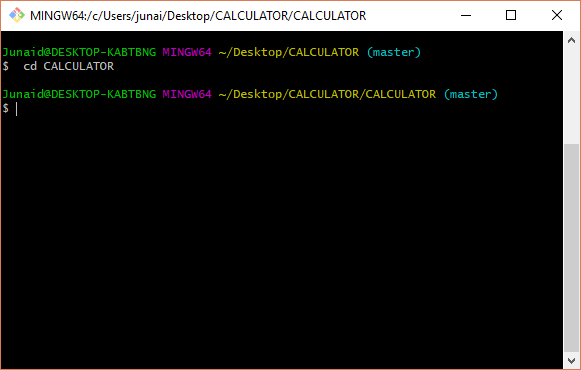


**Exercise -2**

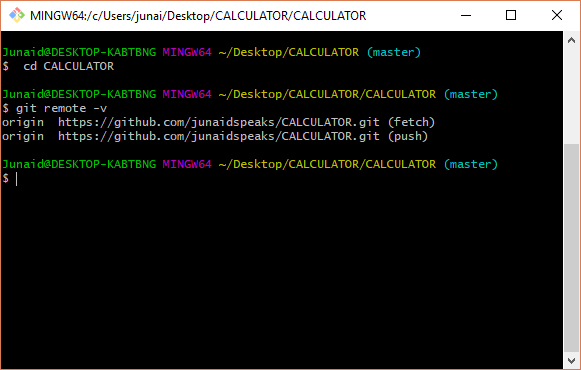
In this exercise, you will clone a repository, make a change to an existing file and upload or push your changes back to the server.



**$ cd gitpractice**

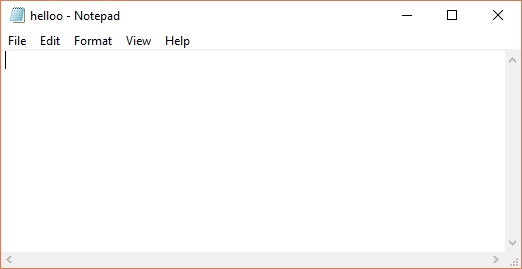
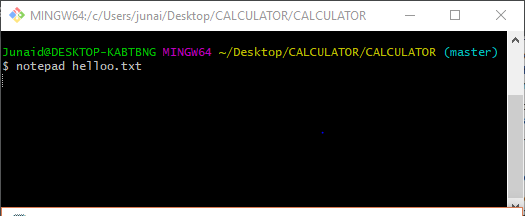


**$ git remote –v**

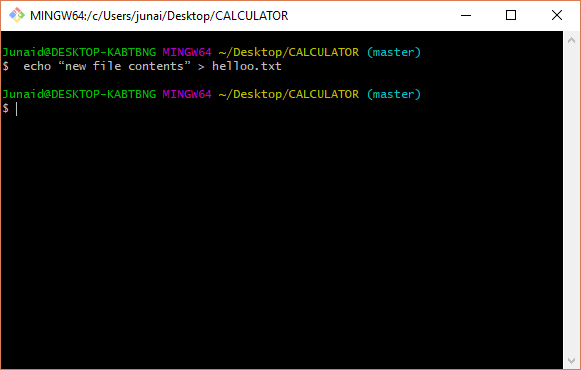


Modify one of the downloaded files or add a new file to the directory:

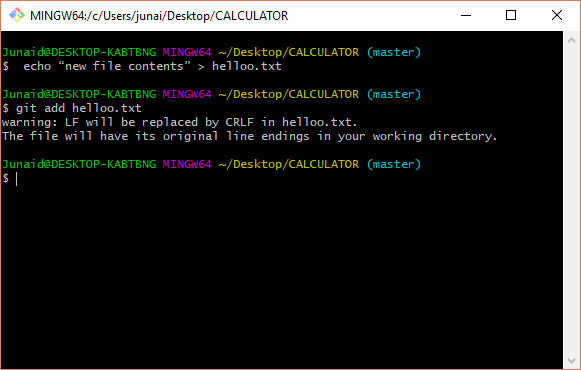
**$ notepad**



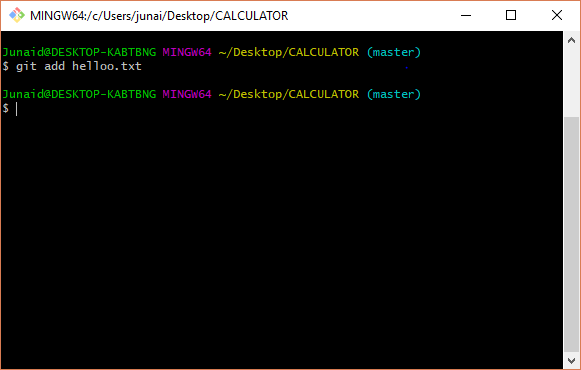
**$ echo “new file contents” > newfile.txt:**



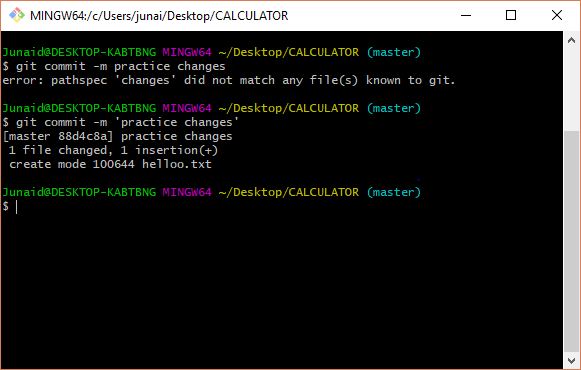
**$ git add <filename> :**



**$ git add newfile.txt:**

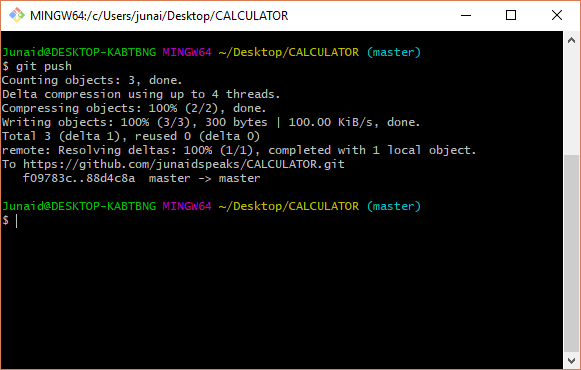


**$ git commit –m ‘practice changes’:**

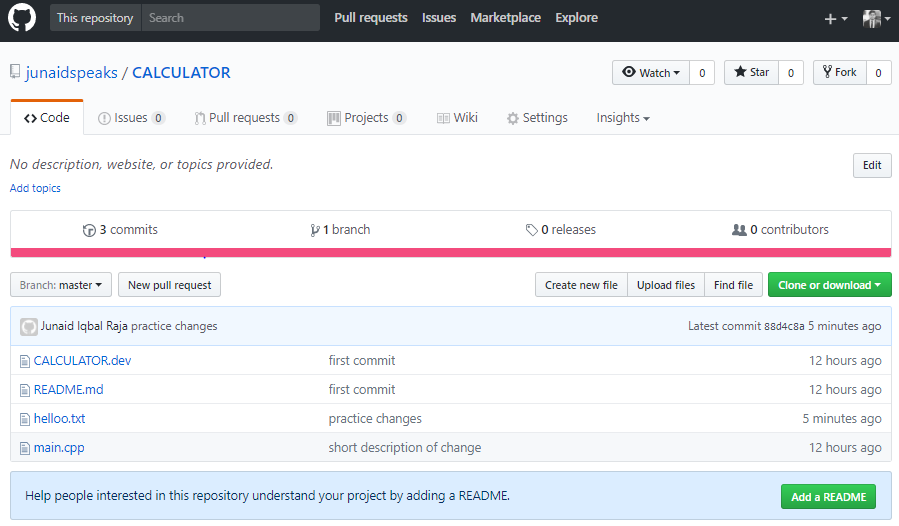


To push your changes to the remote server:

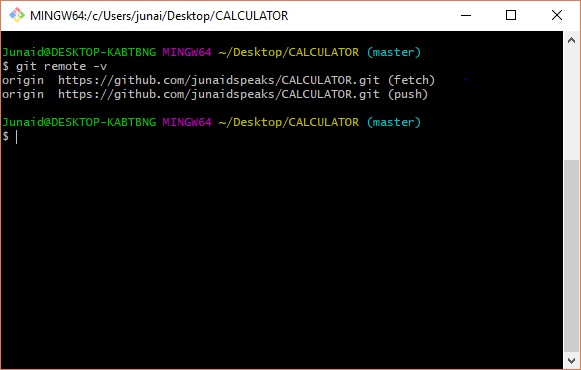
**$ git push:**



**Browse github.com to verify new changes were uploaded correctly:**

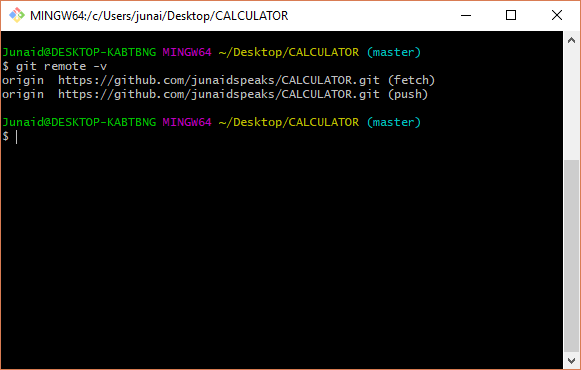


**$ git remote –v:**



Pro tip! As time goes on others may push their own changes to the central repository. On a regular basis you should update your local copy with changes made on the central repository:

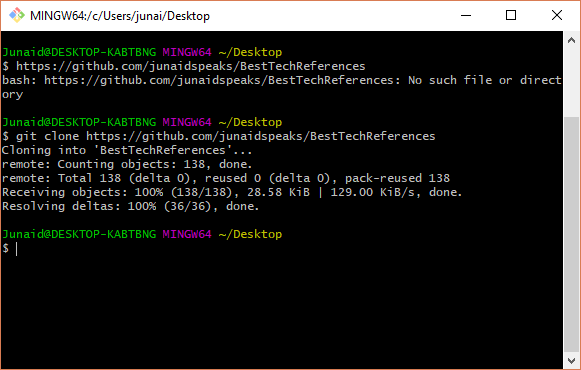
**$ git pull origin master**:



**Exercise 3 – Contributing to an existing github project**

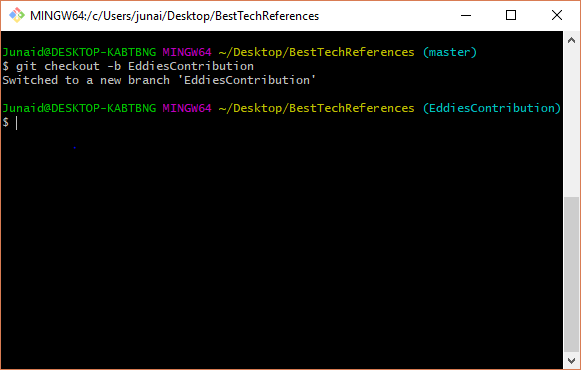
When you fork a github repository, a copy of the repository appears under your account at github. Clone the forked repository to a local computer.

**$ git clone** [**https://github.com/burrisetest/BestTechReferences.git**](https://github.com/burrisetest/BestTechReferences.git)



Create a topic branch for performing work.

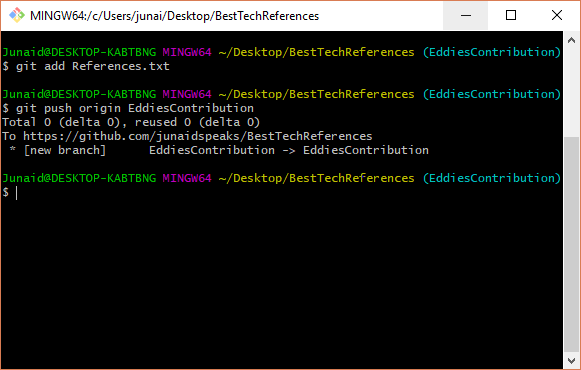
**$ git checkout -b EddiesContribution**



Open the local file References.txt and add a reference. Be sure to include your name so we know who to credit.

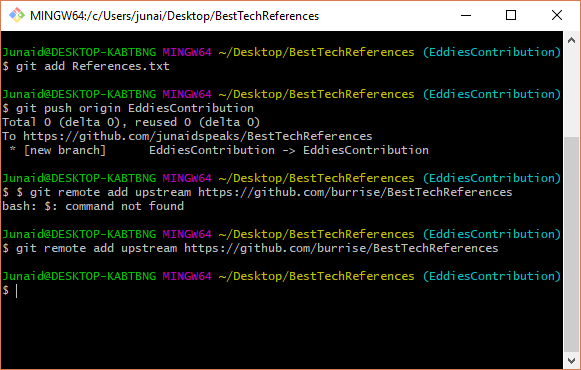
Once you are done with all edits and commits, commit your changes to your topic branch. You could do a git push origin EddiesContribution at this point:

**$ git push origin EddiesContribution**

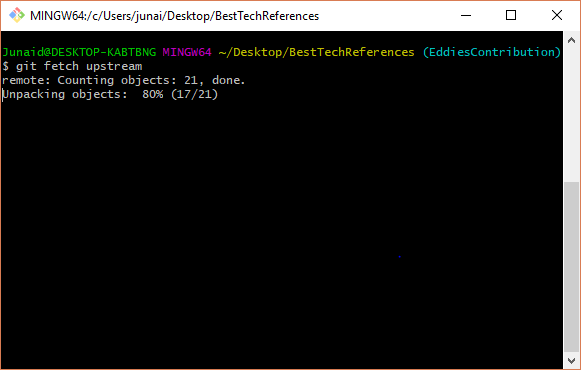


The following command will add an upstream remote:

**$ git remote add upstream** [**https://github.com/burrise/BestTechReferences**](https://github.com/burrise/BestTechReferences)

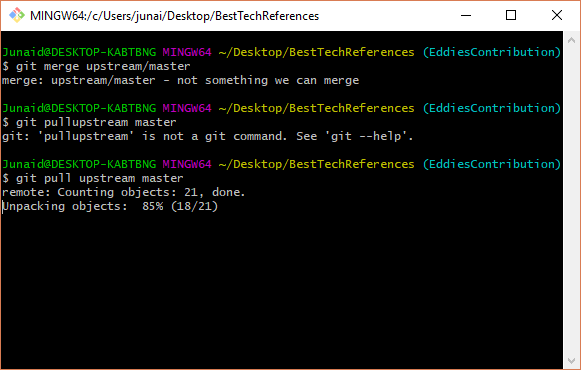


**$ git fetch upstream:**



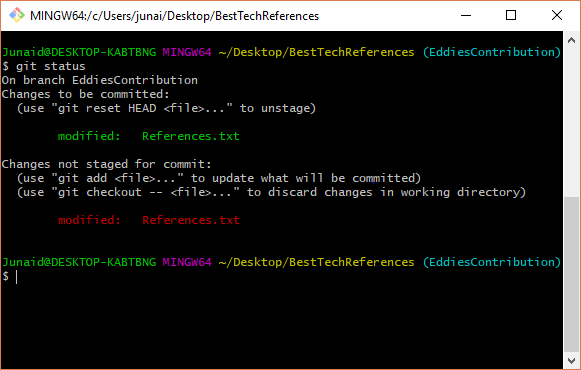
**$ git pull upstream master:**

If there are no upstream changes since you forked, there will be nothing to merge.



If there are upstream changes that can’t be merged automatically, you will have to manually merge the changes. If there are merge conflicts, you will get a message something to the effect “Automatic Merge Failed”. To see which files need to be manually merged, use the git status command:

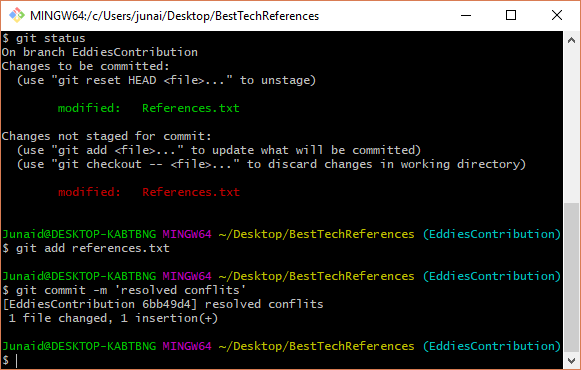
**$ git status**



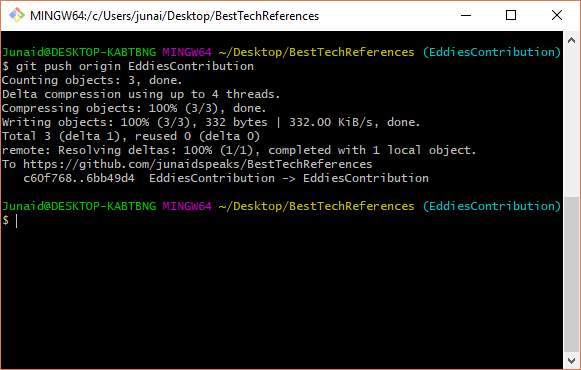
**$ git add filename:**

Staging the file marks it as resolved in Git.

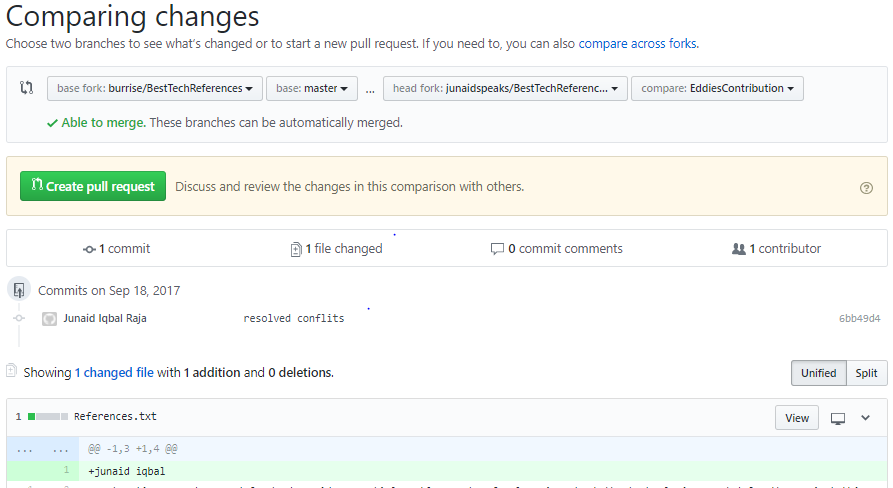
**$ git commit –m ‘resolved conflicts’:**

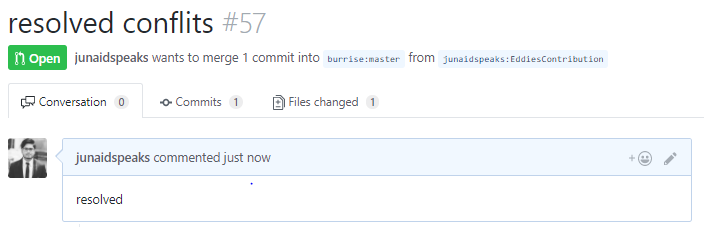


**$ git push origin EddiesContribution:**



Now, send the original author of the repository you forked a pull request.





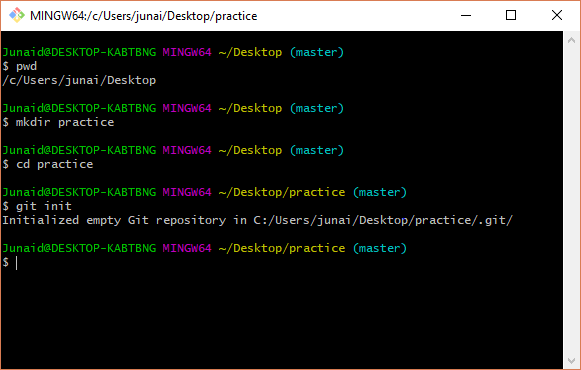
**Exercise – 4**

• Creating a repository on github.com

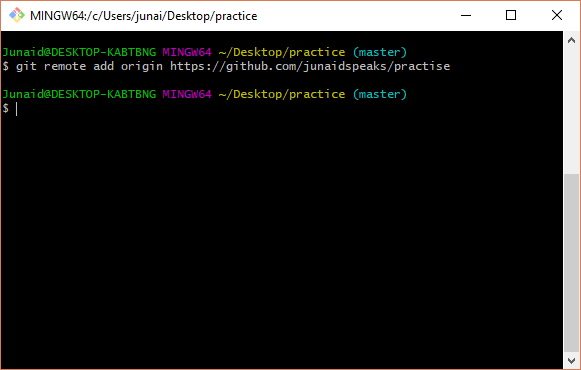
• Committing and syncing (or pushing) changes to github

In this exercise, you will push a local git repository to a remote server. First, create a local git repository (git init). The repository y

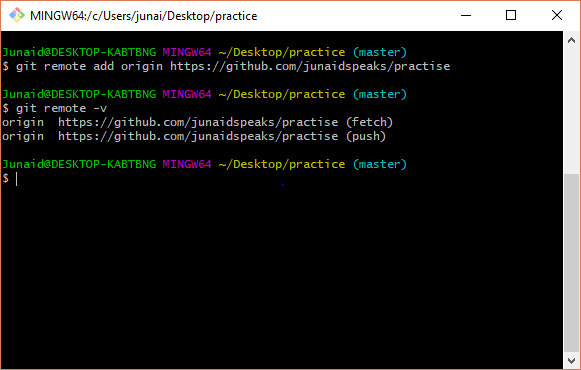
our created in the first exercise will do just fine. Add files and do some commits.



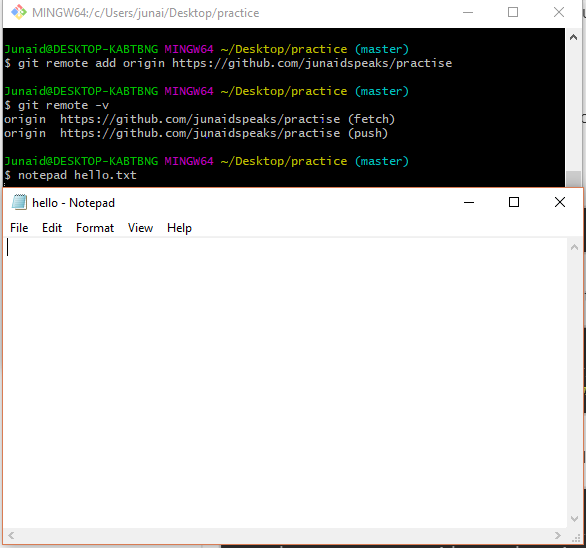
**$ git remote add origin** [**https://github.com/burrise/gitpractice.git**](https://github.com/burrise/gitpractice.git)

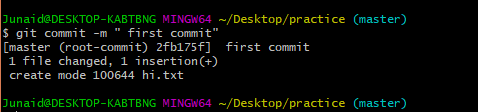


**$ git remote –v:**

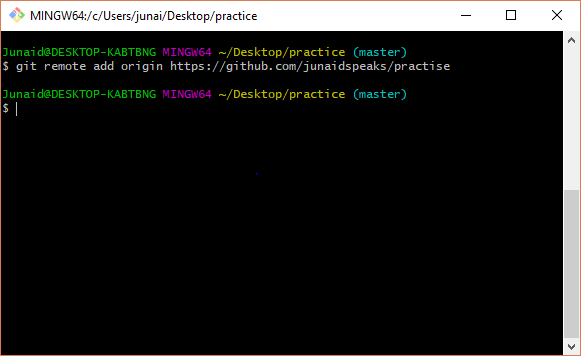


Your local repository is out of sync with the remote that was just created. They have to be in sync before you can push your local changes. The following command will merge the contents of the remote repository (just a readme file) with your existing local repository.

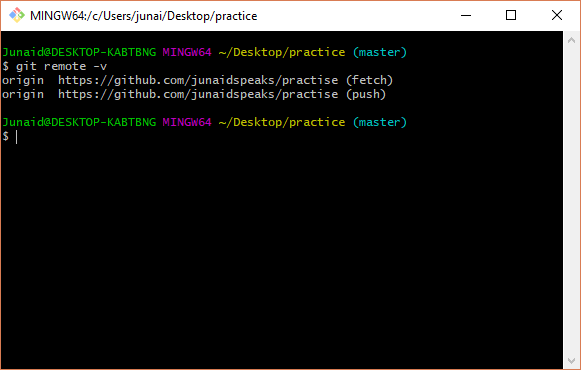




**$ git remote add origin** [**https://github.com/burrise/gitpractice.git**](https://github.com/burrise/gitpractice.git)

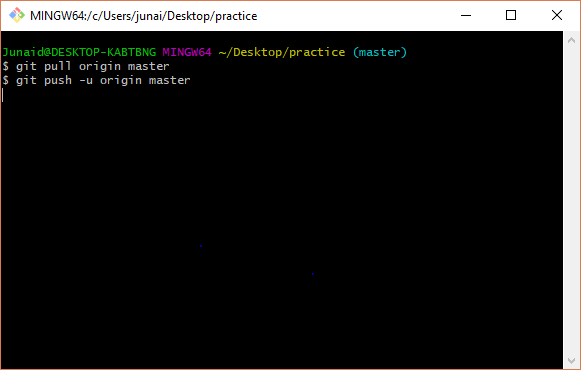


**$ git remote –v:**



**$ git pull origin master :**

**$ git push -u origin master:**



**Exercise 5 – Branching and merging**

This exercise will give you some practice with branching and merging. New feature development and bug fixes are often performed on a branch. Once the work is complete and verified, it is merged back to the main line. This helps ensure there is one branch (usually master or a development branch) that is stable and deployable at all times with the evolution of the product conducted on separate branches.

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ mkdir website

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ cd website

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git init

Initialized empty Git repository in C:/Junaids/junaid/website/.git/

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ notepad index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git add index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git commit -m 'initial checkin'

[master (root-commit) 600b47b] initial checkin

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git branch

\* master

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git branch contactinfo

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git branch

contactinfo

\* master

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git checkout contactinfo

Switched to branch 'contactinfo'

junaid@DESKTOP-KABTBNG MINGW64 ~/website (contactinfo)

$ notepad index.html

junaid@DESKTOP-KABTBNG MINGW64 ~/website (contactinfo)

$ git add index.html

junaid@DESKTOP-KABTBNG MINGW64 ~/website (contactinfo)

$ git commit-m'add phone contact'

git: 'commit-madd phone contact' is not a git command. See 'git --help'.

junaid@DESKTOP-KABTBNG MINGW64 ~/website (contactinfo)

$ git checkout master

Switched to branch 'master'

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ cat index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ sit checkout contactinfo

bash: sit: command not found

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ notepad index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git add index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git commit -m 'add phone contact'

On branch master

nothing to commit, working tree clean

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git checkout master

Already on 'master'

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ notepad index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git add index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git commit -m 'add title'

On branch master

nothing to commit, working tree clean

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git branch

contactinfo

\* master

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git merge contactinfo

Already up-to-date.

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ cat index.html

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git log --oneline --decorate --graph --all

\* 600b47b (HEAD -> master, contactinfo) initial checkin

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

$ git branch -d contactinfo

Deleted branch contactinfo (was 600b47b).

junaid@DESKTOP-KABTBNG MINGW64~/Desktop/practice (master)

**Exercise 6:**

**Branching and Remotes:**

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git remote

origin

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git remote -v

origin https://github.com/Junaidspeaks/practise.git (fetch)

origin https://github.com/Junaidspeaks/practise.git (push)

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git remote add upstream https://github.com/Junaidspeaks/practise

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git branch -r

origin/master

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git checkout origin/feature123

error: pathspec 'origin/feature123' did not match any file(s) known to git.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git branch-r

git: 'branch-r' is not a git command. See 'git --help'.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git branch --r

origin/master

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ is

bash: is: command not found

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ Is

bash: Is: command not found

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git checkout origin/feature123

error: pathspec 'origin/feature123' did not match any file(s) known to git.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ Is

bash: Is: command not found

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git checkout master

Already on 'master'

Your branch is up-to-date with 'origin/master'.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git clone https://github.com/Junaidspeaks/practise.git

Cloning into 'practise'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ cd practise

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git branch -r

origin/HEAD -> origin/master

origin/master

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ cd

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ git clone https://github.com/Junaidspeaks/practise.git practisecopy

Cloning into 'practisecopy'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

junaid@DESKTOP-KABTBNG ~/Desktop/practice (master)

$ cd practise

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git checkout -b fix 123

fatal: '123' is not a commit and a branch 'fix' cannot be created from it

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git push

Everything up-to-date

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git push origin fix 123

error: src refspec fix does not match any.

error: src refspec 123 does not match any.

error: failed to push some refs to 'https://github.com/Junaidspeaks/practise.git'

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git push origin practisecopy

error: src refspec practisecopy does not match any.

error: failed to push some refs to 'https://github.com/Junaidspeaks/practise.git'

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ cd /practisecopy

bash: cd: /practisecopy: No such file or directory

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git branch -r

origin/HEAD -> origin/master

origin/master

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git fetch origin

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git branch -r

origin/HEAD -> origin/master

origin/master

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git checkout -b fix123 origin/fix123

fatal: 'origin/fix123' is not a commit and a branch 'fix123' cannot be created from it

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ notepad file.c

git add

$ git

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git add

Nothing specified, nothing added.

Maybe you wanted to say 'git add .'?

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git commit -m'second developer added a fourth line'

On branch master

Your branch is up-to-date with 'origin/master'.

Untracked files:

file.c

nothing added to commit but untracked files present

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git add

Nothing specified, nothing added.

Maybe you wanted to say 'git add .'?

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git commit -m'second developer added a fourth line'$git

On branch master

Your branch is up-to-date with 'origin/master'.

Untracked files:

file.c

nothing added to commit but untracked files present

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git push

Everything up-to-date

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ cd /practisecopy

bash: cd: /practisecopy: No such file or directory

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git fetch

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git log --oneline

df23618 (HEAD -> master, origin/master, origin/HEAD) Initial commit

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git log -oneline

fatal: unrecognized argument: -oneline

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)

$ git remote show origin

\* remote origin

Fetch URL: https://github.com/Junaidspeaks/practise.git

Push URL: https://github.com/Junaidspeaks/practise.git

HEAD branch: master

Remote branch:

master tracked

Local branch configured for 'git pull':

master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

junaid@DESKTOP-KABTBNG MINGW64 ~/practise (master)