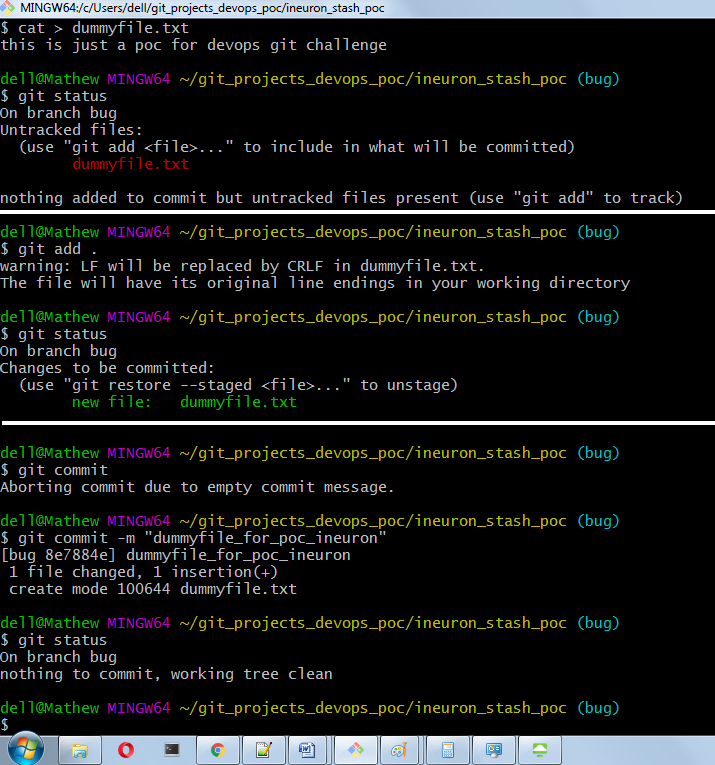
**GIT Assignment 2**

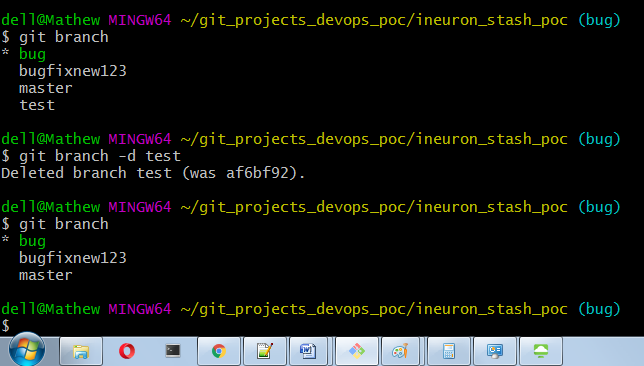
1. What does the command 'git status' do?

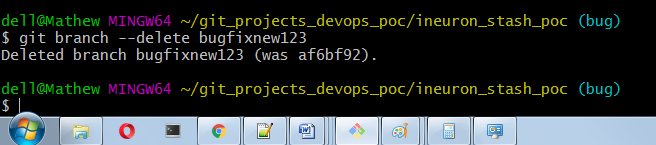
* **git status** shows the status we are in which state currently out of three states(like working state, staging, commit) As usual below is the reference poc screen shot :)
* the red colour dummyfile.txt shows in working area, next green colour staging, last is commit



1. How to delete a Git local branch?

* Command to delete branch is below two ways
* **git branch -d branch\_name**
* **git branch --delete branch\_name**
* just replace the branch\_name with the branch you want to delete .
* As usual below is reference poc :)





1. How can I add a project to Git that already exists?

* Below are the steps/command ( use any one as per requirement in step5a/5b)

step 1 : change directory to root directory of your project

step 2 : Now initialize local directory **git init** or eg. **git init main**

step 3 : add to staging : **git add .** or **git add <file\_name>**

step 4 : now commit : **git commit -m "Type your message/comments"**

**step 5a** : In git bash add the url for the remote repository where your local repository will be pushed **git remote add origin <remote-url>** (this is used to add a new remote)

**step 5b**: **git remote set-url origin <remote-url>** ( this is used to change url of existing remote repository)

step 6 : Verifies the new remote url **git remote -v**

step 7 : **git push origin main**

1. What exactly is a Git fork? What are the distinctions between a fork, a branch, and a clone?

* Let us understand fork. **Fork** is a copy of repository. It simply means you are making a copy of entire repository
* **git fork** is mainly used when you want to propose changes to someone else project or to use an existing project as the starting point.
* Forking a repository will create a copy of the original repository and the repository remains on your GitHub account.
* While when you **clone** a repository, the repository is copied on to your local machine with the help of Git.
* So we can say that **Forking** is done on the GitHub Account and **Cloning** is done using Git.
* **Branch** in Git is just an isolated environment where we can add, modify or delete some portion of the code without messing with the main code/(master) of the project. It's just like creating a sperate envinornmet inside master branch and in that seperate env. when we are done with testing and if it works fine then we can merge them with main/master branch.

1. What is the difference between HEAD, working tree, and index in GIT?

* **HEAD**: HEAD is a pointer to the branch or commit that you last checked out and which will be the parent of a new commit if you make it.
* **Working trees**: They are nothing but the files that you are currently working on and here you can try making any changes before commiting.
* **Index**: The git index is where you place files you want commit to the git repository. The index is a staging area where the new commit is prepared.

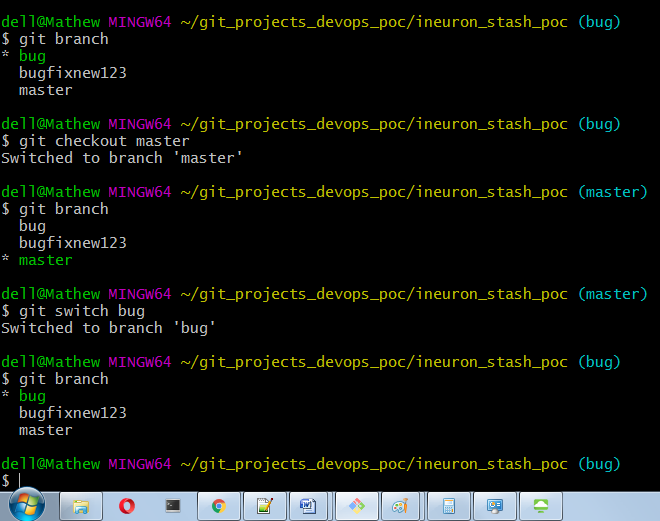
basically contents of the index are what will go into the new commit

1. Which GIT command is used to change branches?

**git checkout branch\_name** -> this will switch/change branch form master branch or anyother branch in which you are currently in to bugfix

**git switch branch\_name** -> after version 2.23 onwards we can use switch keyword to switch/change the branch, else you are used to checkout then it's fine

As usual below is sample poc.. just ignore the filepath since i do lot of poc's :)



1. What is the difference between GitHub and Git?

* **Git**: Git is a distributed version control system for keeping a track/history of all changes in your source code during software development.
* It keeps a track of what changes where done by whom/author at what time.
* It is a command line tool/software
* IT is open sources licensed
* it has no user management options
* **GitHub**: GitHub is just a online web based Git repository hosting service
* It offers all of the distributed revision control and source code management functionality of Git
* It also has some extra features on its own.
* It is a GUI interface
* It has free as well as paid services
* it has built in user management options

1. What are some of the advantages of using the Variation Control System? Which programming language is used in Git?

* I think it is a typo in above question.. i guess that will be version and not variation. let me answer assuming that question is for version control system.
* Few of the Advantages of Version control system are as follows:

1) It keeps track or history of changes done

2) It keeps track in detail like which author did what change at what time with commit id.  
3) It keeps backup meaning whenever we want we can go back to some older version

4) We can easily compare the earlier version of source code(s)

5) Branching is much easier for eg. we have main branch as master and we have 10 commit under that now we got some prod issue then we can sub branch which will be separate from master, we can do some apply fix, test, change code and when that code is working fine then we can merge the sub branch into master or main branch.

* Git is mainly written in C, but we can use multiple languages in git.

It has bindings for many programming languages like [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), Python, and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) etc..

1. List out some Git repository features.

* IT keeps Tracks or history of the source code who changed, when and what in source/code
* It saves time by allowing to work offline once we download from the centralised system or in other words once we get the mirror copy into local
* GIT is Free and open source
* IT is Distributed development
* Undo changes - we can go back and easily track the changes done
* Don't Mix things up - for e.g. there are 10 developers and we don't know who is doing what there will be confusion but with GIT we can check that info and whoever has done generally will test in their local system before commit.
* Make useful commits - if you are making 10 changes then do 10 commits for each change. Be atomic and mention all small changes and do commit.
* It Creates backups
* GIT is Scalable depending upon the app/business requirements
* Supports collaboration
* Branching is much easier for e.g. we have main branch as master and we have 10 commit under that now we got some prod issue then we can sub branch which will be separate from master, we can do some apply fix, test, change code and when that code is working fine then we can merge the sub branch into master or main branch.