**Introduction to git commands:**

Go to Github, and create an account

* Fork <https://github.com/AbdulAli/gitdemo.git> into your account
* git clone https://github.com/{$YourAccountName$}/gitdemo.git
* Create a readme file from command line:
  + echo “# gitdemo” >> readme.md
  + vim readme.md
  + git add -A or git add - -all 🡪 adds (stages) files to be used by the commit
    - git add . 🡪 juts adds the added files, and does not remove the deleted files)
  + git commit -am “readme file added.” 🡪 -a option, keeps track of all files (including modified and removed ones)
  + git push origin master
* Hint: You can also create a repo from terminal.
* Getting help:
  + man git-commit
* Mention git config:
  + git config --list
  + git config —global user.name "Name Family"
  + git config —global user.email $Your\_github\_email$
* Remove files you created:
  + Before adding:
    - echo “test” >> temp.txt
    - Do one of the followings:
    - Rm temp.txt
    - Or
    - Use git clean to do it automatically (especially if there are several files):
      * git clean -nd 🡪 -n is for dry run. Just tells you what would be deleted in real run!
      * git clean -fd 🡪 -f is for force. -d is for directories.
  + After adding:
    - echo “test” >> temp.txt
    - git add -A
    - Now, one of the followings:
      * git reset temp.txt
      * git reset 🡪 deletes all the added files.
* Show list of commits:
  + Open another terminal window and try using git log (keep it open for later usage):
    - git log
    - git log - - graph
    - gitk
    - gitk –all 🡪 Great visualization tool!
* Do two more pushes (for example, add test1.txt and test2.txt).
* Make a conflict (between online and local pushes) and show how to resolve it:
  + Create a file, like test5.txt in browser and commit.
  + Create a file with the same name in terminal, and commit.
  + Try to push
    - Git push origin master 🡪 [Rejected … the remote contains work that you do not have locally …]
  + We should pull and then push
  + git pull
    - CONFLICT in test5.txt
  + To resolve the conflict, edit the test5.txt, save it, and do:
    - git add -A
    - git commit -am “conflict resolved”
    - git push origin master
  + Now check the online repo, also do:
    - git log
  + CONCLUSION:
    - Always do “git pull”, then start working, then add, commit and push.
    - Do this more frequently.
* Ignoring some files/folders from being tracked:
  + mkdir data
  + cd data
  + touch a.txt
  + touch .gitignore
  + vim .gitignore
  + data 🡪 you can add “bin”, etc.
  + Now, add it, commit and push. Then check the online repo.
* Branches:
  + Test the commit history (and see why we need branches?):
    - ls
    - git checkout <sha\_of\_an\_older\_commit> 🡪 4 letters are enough
    - ls 🡪 HEAD is detached; do not change the code in this stage. Instead, we’ll use the braches.
    - git checkout master
    - ls
  + Commits: snapshots of the project. Switching between them is so easy and fast.
  + Branches:
    - Branch early, and branch often
    - a branch essentially says "I want to include the work of this commit and all parent commits."
    - HEAD is the symbolic name for the currently checked out commit (always points to the most recent commit which)
    - HEAD can be thought of as a variable pointing to a specific commit. It can change and isn’t related to a branch.
    - master is the common name for the default branch. It doesn’t need to exist, but it often does.
  + Remotes are [local] aliases that store the url of repositories. It helps us not to type the full remote url when pushing.
    - git remote -v 🡪 shows what url belongs to each remote
    - Issuing new commits changes HEAD, checking anything out changes HEAD.
    - Origin is the default alias for your remote repo
  + git branch 🡪 shows all the branches in your repo
  + git branch -a 🡪 shows “remote” branches
  + git remote -v 🡪 shows what url belongs to each remote
  + git branch
  + git remote
  + git branch <name\_of\_branch> 🡪 creates a new branch (does not check it out).
  + git checkout database
  + Create a new file called “db.txt”, add it and commit. Then push;
    - git push origin database
  + Check the created file online, and the branch.
  + Create a new file called “db2.txt”, add it and commit. Then push;
    - git push origin database
  + Check the created file online, and the branch.
  + git branch
  + Show how this affects master:
    - git checkout master
    - Check the terminal, and, the “File manager”
  + git checkout master
  + add a file, functionality1.txt, and then a commit and push in master.
  + add a file, functionality2.txt, and then a commit and push in master.
  + gitk --all
  + Diverging again:
    - git checkout database
    - add a file, abc1.txt, and then a commit and push in database branch.
    - gitk – all
    - See how the branch is diverging again!
  + git checkout master
  + git merge database
    - edit the commit