Apurva Patel

Jenny George

IS601-851

Mini-Project # 1

Git is a distributed version control system (DVCS). This is very commonly used for open source and commercial digital development. This system essentially allows every user to get access to every files and branches that have been made in the repository. It gives a user full access to all the changes that were made to the system and give him/her access to all history of the repository. In this system developers can work from basically anywhere in the world on a repository if they are the collaborators for that repository to make changes in it. This basically eliminates slower work environment by reducing unnecessary tasks. Docker is essentially a tool to design create, deploy and run applications. This helps many companies to maintain and store files in container. So, it not too difficult for a software developer to deploy an application. Developers can could store their files in the libraries to be used as in future reference.

Automated testing is also a very important tool in the industry. This helps many firms across the globe for a faster and efficient testing operation for its software. There are less chances of human error when testing for a specific software is done automatically.

**Git Commands & Terminology**

1. **Repository:** A repository is the most basic element of GitHub. It is basically your folder for the entire project. This folder will contain all you document, project file and each of its revision history. The repository can be public, so multiple collaborators can work on the same project under one repository.



Simple representation of a repository where the initial commit is **C0** and the following commit is **C1**



Visual representation of a project repository with 3 commits.

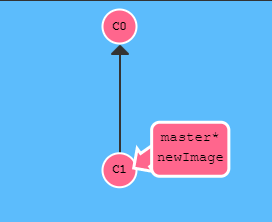
1. **Clone:** A clone is a copy of the repository that is basically saved on your computer instead of on a website’s server. In this copy of your current repository you can edit files in your preferred editor and use Git to keep track of the changes made to the repository without having to being online.



Once the command, “**git clone**” is implemented, it creates a remote repository of a working project

Original git repository with two commits. The initial commit is **C0** and the following commit is **C1**

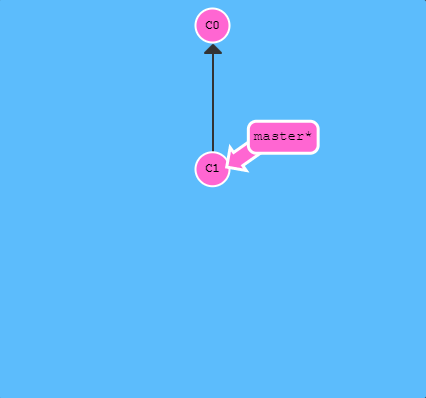
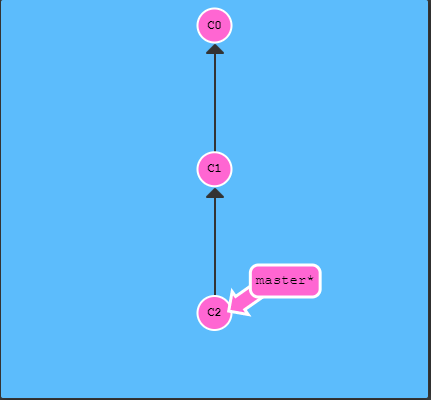
1. **Fork:** A fork is a personal copy of another user’s repository that is basically in your account. This allows you to easily make changes without affecting the original repository. Additionally, this is also attached to the original repository. So, you can essentially submit a pull request to the author/owner of the repository to make update with your new implemented changes.
2. **Branch:** A branch is essentially a parallel version of your repository. This is embedded within the repository but does not affect the primary master branch for the repository. When the changes that needs to be done or updated are completed on the branch, you can merge that branch into the master branch. This will basically publish all the changes to the master branch.



Once the command, “**git branch newImage**” is implemented, a new branch is created which now refers to the commit C1

Original git repository with two commits. The initial commit is **C0** and the following commit is **C1**

1. **Commit:** A commit is an individual change to a file. Every time you save in Git, it essentially created a unique ID for the file. It essentially records a snapshot of all the files in your directory. This allows the user/owner of the repository, to keep track of what changes were made in the file when and by who. (Please see listed example below)

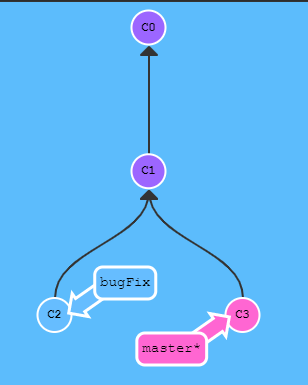
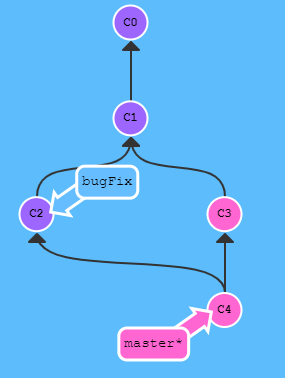


Once the command, **“git commit”** is implemented, it makes changes to the repository. It has a new commit **C2** to the repository and saves its.

Original git repository with two commits. The initial commit is **C0** and the following commit is **C1**

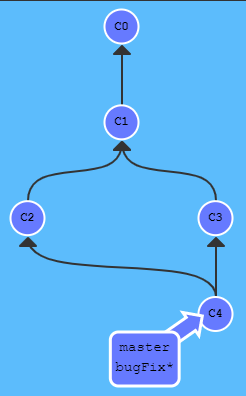
1. **Merge:** Merging essentially takes changes from one branch within the repository and applies them into another. This often takes place a pull request within the repository. This pull request can be done with the GitHub web interface if there are not conflicting changes on the branches.

**Merging bugFix branch into master branch**



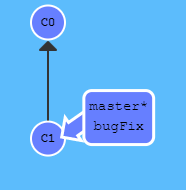
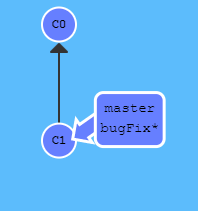
“**git merge bugFix**” command applied

Original repository



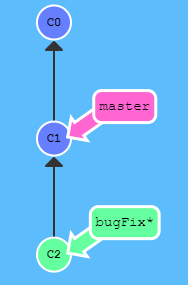
To achieve this step, **“git checkout bugFix”** was applied. This step was performed so, the bugFix branch can be selected. This was than followed by **“git merge master”** command to merge master int bugFix

1. **Checkout:** The checkout command is used when you need to navigate between the created branches in the repository.



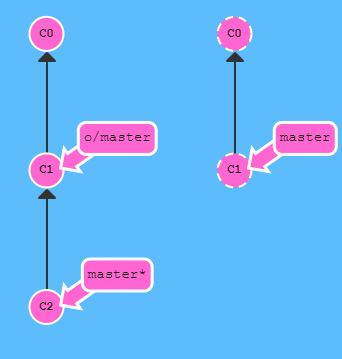
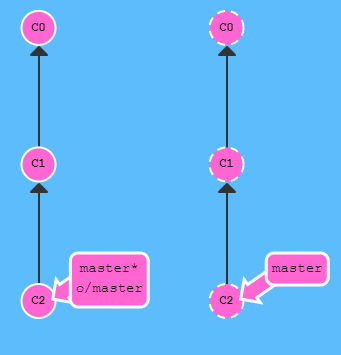
**“git checkout bugFix”** command applied to put the asterisk on bugFix from master.

Original repository with two commits. C0 and C1. When command “git branch bugFix” is inserted, it created a new branch on the C1 commit



**“git commit”** command applied to shift bugFix to C2

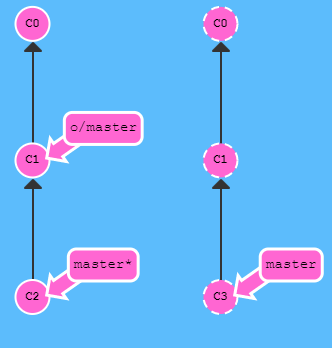
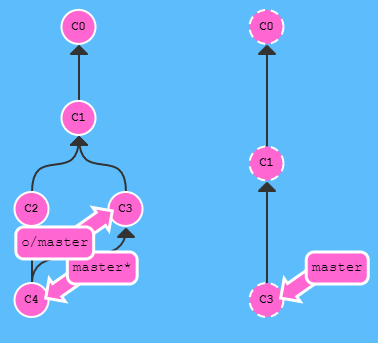
1. **Push:** Push generally refers to sending committed change to a remote repository. In this instance, if an owner/user made the changes to the repository locally, then he/she would need to push to changes. So, other collaborators on the repository can access them.



Once, **“git push”** command is applied, the remote repository received the commit C2. The master branch on the on the remote repository was also updated at C2

In this repository, some changes were made to the original repository which were not present on the remote repository

1. **Pull:** Pull generally refers to when you are fetching changes and merging them. This is usually helpful when someone has edited a file that multiple users are working on. Using pull in this instance will pull those changes to the local copy of the repository.



Once, **“git pull”** command is executed, you essentially fetched C3 from you remote repository to local and then merged it with the o/master. These commands adds new features to the local repository from the remote one.

Current Repository

1. **Status:** This is essentially a type of status check on GitHub. These are used in GitHub to let the user know if the commits that he/she made is meets the condition that are set form that repository.
2. **GitHub Flow:** This flow is essential a branch-based workflow. This flow has 6 steps to it.

* Create a branch
* Add commits
* Open pull request
* Discuss & review code
* Merge
* Deploy

1. **cd:**  Cd stands for “change directory”. Cd switches you into the specific directly of your choosing as well as changing the working directory. The cd command takes a directory name and then switches that to a different directory.



1. **Mkdir**: mkdir is used to make a directory. We use it to create the directories that do not already exist.



1. **Cp:** This command is used to copy files in a directory. We use this command so we will be able to create an exact replica of the file with a different name.

****

1. **Pwd**:  Pwd stands for print working directory. This command prints the working directory. Pwd is an environment which enables us to store the path.



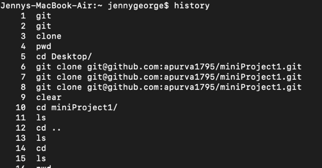
1. **mv.** mv stands for move. This command moves one file from once place to another. We can also use this command to rename a folder.



1. **rm:** rmstands for remove. This command is used to remove files from different directories in Linux.



1. **history**: The history command shows a list all commands that were executed previously since the session had began. We can use this to view those commands and prevent certain commands from being visible.



1. **home directory:** This command is used as a repository for the users personal files and programs. This directory is also called the login directory.
2. **File path:** This is a representation of the location of a file or folder on a computer system.