WDV341 – PHP

Robert McDonell

GIT Terminology

All information cited using Wikipedia..

1. Version Control Software -  a distributed **version control** system, it can be used as a server out of the box. Dedicated **Git** server**software** helps, amongst other features, to add access**control**, display the contents of a **Git** repository via the web, and help managing multiple repositories.
2. Git - Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
3. Repository - . A **git repository**contains, among other things, the following: A set of commit objects.
4. Stage - Staging is a step before the commit process in **git**. That is, a commit in **git** is performed in two steps: staging and actual commit. As long as a changeset is in the staging area, **git** allows you to edit it as you like (replace staged files with other versions of staged files, remove changes from staging, etc.)
5. Commit - **commit** puts your changes into your local repo, while **git** push sends your changes to the remote location. ... **git** push is used to add **commits** you have done on the local repository to a remote one - together with **git**pull , it allows people to collaborate.
6. Push - **git push** "updates remote refs along with associated objects".
7. Pull - " **git pull** " will fetch and replay the changes from the remote master branch since it diverged from the local master (i.e., E ) until its current commit ( C ) on top of master and record the result in a new commit along with the names of the two parent commits and a log message from the user describing the changes.
8. Revert -  **git revert** is used to record a new commit to reverse the effect of an earlier commit (often a faulty one). If you want to throw away all uncommitted changes in your working directory, you should see git-reset[1], particularly the --hard option.
9. Branching - Branching is a feature available in most modern version control systems. Branching in other VCS's can be an expensive operation in both time and disk space. In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes. This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future's history before merging it into the main branch.
10. Merging -  **git merge** is used to combine two branches. The following examples in this document will focus on this branch **merging** pattern. In these scenarios, **git merge** takes two commit pointers, usually the branch tips, and will find a common base commit between them.
11. change history -  One of the great things about Git is that it allows you to make decisions at the last possible moment. You can decide what files go into which commits right before you commit with the staging area, you can decide that you didn’t mean to be working on something yet with git stash, and you can rewrite commits that already happened so they look like they happened in a different way. This can involve changing the order of the commits, changing messages or modifying files in a commit, squashing together or splitting apart commits, or removing commits entirely — all before you share your work with others.
12. clone or clone(d) -  If a project has already been set up in a central repository, the **git clone** command is the most common way for users to obtain a development copy. Like **git** init , cloning is generally a one-time operation.

Create a Cheat Sheet of Git commands for your own use. Attach a copy of that sheet to this assignment when complete.

