 **GIt** **Key Terms**

**Version Control System (VCS)** or **Source Code Manager (SCM): A VCS allows you to: revert ﬁles back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modiﬁed something that might be causing a problem, who introduced an issue and when, and more.**

**Commit (snapshot): Git thinks of its data like a set of snapshots of a mini ﬁle system. Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your ﬁles look like at that moment and stores a reference to that snapshot.**

**Repository (repo): A directory that contains your project work, as well as a few ﬁles (hidden by default in Mac OS X) which are used to communicate with Git. Repositories can exist either locally on your computer or as a remote copy on another computer.**

**Working Directory: The ﬁles that you see in your computer's ﬁle system. When you open your project ﬁles up on a code editor, you're working with ﬁles in the Working Directory.**

**This is in contrast to the ﬁles that have been saved (in commits!) in the repository.**

**When working with Git, the Working Directory is also diﬀerent from the command line's concept of the current working directory which is the directory that your shell is "looking at" right now.**

**Checkout: When content in the repository has been copied to the Working Directory. It is possible to checkout many things from a repository; a ﬁle, a commit, a branch, etc.**

**Staging Area** or **Staging Index** or **Index: A ﬁle in the Git directory that stores information about what will go into your next commit. You can think of the staging area as a prep table where Git will take the next commit. Files on the Staging Index are poised to be added to the repository.**

**SHA: A SHA is basically an ID number for each commit. It is a 40-character string composed of characters (0–9 and a–f) and calculated based on the contents of a ﬁle or directory structure in Git. "SHA" is shorthand for "SHA hash". A SHA might look like this:**

**e2adf8ae3e2e4ed40add75cc44cf9d0a869afeb6**

**Branch: A branch is when a new line of development is created that diverges from the main line of development. This alternative line of development can continue without altering the main line.**

**Going back to the example of save point in a game, you can think of a branch as where you make a save point in your game and then decide to try out a risky move in the game. If the risky move doesn't pan out, then you can just go back to the save point. The key thing that makes branches incredibly powerful is that you can make save points on one branch, and then switch to a diﬀerent branch and make save points there, too.**

**Take Your Own Notes**

## git log --stat - a flag that can be used to display the files that have been changed in the commit, as well as the number of lines that have been added or deleted. The flag is --stat ("stat" is short for "statistics"):

## git log -p: a flag that can be used to display the actual changes made to a file. The flag is --patch which can be shortened to just -p

1. git log -p -w: git log -p --stat
2. git log -p --stat: git log -p --stat
3. git log or git show: display a specific commit's details without worrying about all of the others in the repo?.There are actually two ways to do this!

* providing the SHA of the commit you want to see to git log
* use a new command git show

1. git log -p fdf5493: By supplying a SHA, the git log -p command will *start at that commit*! No need to scroll through everything! Keep in mind that it will *also* show all of the commits that were made *prior* to the supplied SHA.
2. git show command will show *only one commit*. So don't get alarmed when you can't find any other commits - it only shows one. The output of the git show command is exactly the same as the git log -p command

## git diff-The git diff command can be used to see changes that have been made but haven't been committed, yet.