**GIT COMMANDS:**

**SETUP:**

1. git config --global user.name “[firstname lastname]”

set a name that is identifiable.

1. git config --global user.email “[valid-email]”

set a mail id for identification.

1. git config --global color.ui auto

set automatic command line coloring for Git for easy reviewing

**SETUP and INIT:**

1. git init : Initializing an existing directory as git repository.
2. git clone <url> : retrieve an entire repository from a hosted location via URL
3. git status : show modified files and status of the file.
4. git add <file> : add a file as it looks now to your next commit.
5. git add –all: add all files
6. git reset <file> : unstage a file while retaining the changes in working directory.
7. git reset –hard <commit id>:

--hard : will erase your local work if you have anything stashed.  
--mixed : Unstaging all changes but leave them in the working directory (default).  
-soft : staged and working directory are not altered

1. git revert HEAD:  the commit with a new commit to reverse changes.
2. git diff: diff of what is changed but not staged.
3. git diff –staged: diff of what is staged but not yet commited.
4. git commit -m “[descriptive message]”: commit your staged content as a new commit snapshot.
5. git blame <filename>: Show file line numbers annotated with author information can also see in git log
6. git branch –a: list all the branches
7. git checkout <branch>: checkout or switch to branch
8. git checkout –b <branch>: create new branch and checkout to new branch
9. git checkout <commit id> : checkout to the commit id.
10. git diff e3b43d63 60fdba79 directory/path: show diff between two commit ids
11. git log: Show the Git log messages.
12. git log –oneline: Show the Git log messages in oneline
13. git log --graph --decorate –oneline: Show the Git log messages also displays graph
14. git log --author="name": displays author edited files log message

**Inspect & Compare:**

1. git log: show the commit history for the currently active branch.
2. git log branchB..branchA: show the commits on branchA that are not on branchB.
3. git log --follow [file]: show the commits that changed file, even across renames.
4. git show: **command line utility** that is used to view expanded details.

**Share & Update:**

1. git remote add [alias] [url]: A new connection to remote repository
2. git fetch [alias]: fetch down all the branches from that Git remote
3. git merge [alias]/[branch]: merge a remote branch into your current branch to bring it up to date
4. git merge branch-name: merge branch name to current branch
5. git merge –abort: stop the merge and return everything to its pre-merge state.
6. git mergetool: View file conflicts in a merge tool. Default is vimdiff
7. git push [alias] [branch]: Transmit local branch commits to the remote repository branch
8. git pull: fetch and merge any commits from the tracking remote branch
9. git tag: list all created tags
10. git tag -a April\_Sprint -m "Source used for April sprint demo": create a tag for given commit message
11. git tag -a April\_Sprint -m "Source used for April sprint demo" e3b43d63: ": create a tag for given commit message and for commit id.

**Tracking path changes:**

1. git rm [file]: delete the file from project and stage the removal for commit
2. git rm -r directory/path: delete recursive removal when given a directory name.
3. git mv [existing-path] [new-path]: change an existing file path and stage the move
4. git cp [existing-path] [new-path]: copy file from an existing file path and stage the move
5. git log --stat –M: show all commit logs with indication of any paths that moved
6. git ls-files –stage: list files staged files
7. git ls-files –cached: list cached files
8. git ls-files –modified: list modified files
9. git ls-files –others: list untracked files
10. git ls-files –deleted: list deleted files