**GIT Commands**

* git cat-file -t <hash>
* git cat-file -p <hash>
* git cat-file -p <treehash>
* git cat-file -p <blobhash>
* **config**
  + git config -l
  + git config --global --replace-all user.email 'parvezmisarwala@gmail.com'
  + git config - -global core.autocrlf true ( in case of windows)
  + git config - -global core.autocrlf input ( in case of mac and linux)
  + git config - -global core.autocrlf false ( if only windows)
  + git config --global core.editor vim
  + git config core-sparsecheckout true (to checkout selected directory). Details below
  + git config - -global merge.tool kdiff3 (brew install kdiff3)
    - * + git mergetool -t kdiff3
  + git config - -global diff.tool kdiff3
  + git config --global core.whitespace -trailing-space,-space-before-tab - This will disable whitespace warnings
  + git config - -global clean.requireForce false
* **add, commit, amend**
  + - git add .
    - git add -u - Stages only modified files and ignores untracked files
    - git add -i ( for interactive staging)
    - git add file1.txt file2.txt file3.txt
    - git add \*.txt
    - git add p
    - git rm —cached - To unstage
    - git reset HEAD <filename> - remove from staging
    - git commit
    - git commit -m <commit message>
    - git commit -am <commit message>
    - git commit - -amend
    - Exercise
      * 1. Create 4 files and commit these files in all separate commits
* Status
  + git status -u - Shows untracked files
  + git status -sb - Gives output in short format of your branch
* **.gitignore**
  + - .gitignore - to share the ignore files
    - exclude - for local ignore
    - git check-ignore -v \* - To list ignored files
* **status**
  + - git status
    - git status -u : shows untracked files
    - git status -sb : gives output in short format of your branch
* **log**
  + - git shortlog -s -n : To get the number of commits
    - git log -p
    - git log - -oneline
    - git log - -graph
    - git log --oneline --abbrev-commit --all --graph
    - git log - -stat: Show statistics for files modified in each commit
    - git log - -shortstat: Display only the changed/insertions/deletions line from the --stat command.git
    - git log - -name-only: Show the list of files modified after the commit information.
    - git log - -name-status: Show the list of files affected with added/modified/deleted information as well.
    - git log --pretty=oneline
    - git log --pretty=format:"%h - %an, %ar : %s”

%H-Commit hash, %h-Abbreviated Commit hash

%T- Tree hash, %t-Abbreviated tree hash

%P-Parent hashes, %p-Abb.. parent hashes

%an-Author name, %ae-Author email, %ad-Author date, %ar-Author date relative

%cn-committer name, %ce-Committer email, %cd-Committer date, %cr-Relatives

%s: Subject (commit message)

* **Alias**
  + git config —global [alias.co](http://alias.co) commit
  + git config —globbal alias.last ‘log -1 HEAD’
  + git config --global alias.unstage 'reset HEAD —'
* **diff** - Show changes between commits, commit and working tree, etc
  + - git diff --staged - Shows difference between staged and last commit
    - git diff: Show differences between your working directory and the index.
    - git diff --cached: Show differences between the index and the most recent commit.
    - git diff HEAD: Show the differences between your working directory and the most recent commit.
    - git diff --color-words HEAD
    - git diff --name-status master..branchName
    - git diff --ignore-space-at-eol -b -w —ignore-blank-lines
      * + -b is synonymous to --ignore-space-change
        + -w is synonymous to --ignore-all-space
* **rm**
  + - git rm - Remove files from the working tree and from the index
    - -q, --quiet do not list removed files
    - --cached only remove from the index (to untrack)
    - -n, --dry-run dry run
    - -f, --force override the up-to-date check
    - -r allow recursive removal
    - --ignore-unmatch exit with a zero status even if nothing matched
* **mv**
* manually renaming a file
* **clean** - Remove untracked file from working directory
  + - git clean -f -n - Show what will be deleted with the -n option:
    - git clean -f -d - Also removes directories
    - git clean -f -X - Also removes ignored files
    - git clean -f -x - Removes both ignored and non-ignored files
    - git clean -i
* Revert - To undo a committed snapshop
* Reset - Reset Current head to specified state
  + - * Git reset --hard

Clean directory, no modified files

Modified files, not staged yet

Staged file, not committed yet

* + - * Git Reset --soft - Does not touch the index file or the working tree at all

Clean directory, no modified files

Modified files, not staged yet

Staged file, not committed yet

* + - * Git Reset - -mixed -Resets the index but not the working tree

Clean directory, no modified files

Modified files, not staged yet

Staged file, not committed yet

* branch, merge, rebase, rebase -i, conflict, mergetool
  + - * View conflicted files

git diff --name-only —diff-filter=U

git ls-files -u | awk '{print $4}' | sort | uniq

git ls-files -u | cut -f 2 | sort -u

* + - * View if branch is merged or unmarked

git branch --merged master

git branch --no-merged master

* + - * + Rebase

git rebase master

git rebase —onto

git rebase --onto master next topic

git rebase --onto topicA~5 topicA~3 topicA

git rebase -i

* **Patch**
  + - git format-patch master --stdout > fix\_empty\_poster.patch : This will create a new file fix\_empty\_poster.patch with all changes from the current (fix\_empty\_poster) against master.
    - git apply --stat fix\_empty\_poster.patch: This will show commits which is present in path file
    - git am --signoff < fix\_empty\_poster.patch - This will apply the patch
    - git format-patch -10 HEAD --stdout > 0001-last-10-commits.patch: The last 10 patches from head in a single patch file:
    - git format-patch -1 <sha> --stdout > specific\_commit.patch: To generate patch from a specific commit (not the last commit):
    - Apply Patch
      * + Checkout to a new branch: $ git checkout review-new-feature
        + # If you received the patch in a single patch file: $ cat new-feature.patch | git am
        + # If you received multiple patch files: $ cat \*.patch | git am
* cherrypick
  + - To pick up a particular commit
* Stash
* remote
  + git remote
  + git remote -v
  + git remote show origin
  + git remote rename pb paul
  + git remote rm paul
  + git branch --set-upstream-to=upstream/foo foo
* Clone
* Sync Repository with Pull Push fetch merge and rebase
  + - Pull - Updates the working directory
* Pull & Rebase: There are now 3 different levels of configuration for default pull behavior. From most fine grained to most general, they are:
  + - * + branch.<branchname>.rebase

Setting this to true means that that particular branch will always pull from its upstream via rebasing, unless git pull --merge is used explicitly.

* + - * branch.autosetuprebase

Setting this to always means that whenever a tracking branch is created, a configuration entry like the one above will be created for it. For finer grained control, this can also be set to never, local or remote and can be set per repository or globally. See git config --help for further details.

* + - * pull.rebase

Setting this to true means that git pull is always equivalent to git pull --rebase (unless branch.<branchname>.rebase is explicitly set to false). This can also be set per repository or globally.

* Fetch and Merge
* tag
  + - git tag v1.0 -m ‘tag message’
    - git tag -l
    - git checkout v1.0 -b NewBranch
* submodules
  + - A submodule allows you to keep another Git repository in a subdirectory of your repository.
    - Submodule does not automatically upgrade
    - The other repository has its own history, which does not interfere with the history of the current repository. This can be used to have external dependencies such as third party libraries for example.
      * git submodule add <https://github.com/pmisarwala/myrepo1.git>
      * cat .gitmodules
      * Clone a repo with submodule

git submodule update - -init

* **subtree**
  + - Add a subtree: git subtree add --prefix .vim/bundle/fireplace https://github.com/tpope/vim-fireplace.git master —squash
    - To update: git subtree pull --prefix .vim/bundle/fireplace https://github.com/tpope/vim-fireplace.git master --squash
* **Split** repository
  + - * git subtree split --prefix=lib -b split
* **symlinks**
  + - * ln -s originalfile linkedfile
      * git ls-files -s
      * git config - -system core.symlinks true
* **git rev-parse - -verify HEAD**
  + One common usage of git rev-parse is to print the SHA1 hashes given a revision specifier.
  + --verify to verify that the specified object is a valid git object.
  + --git-dir for displaying the abs/relative path of the the .git directory.
  + Checking if you're currently within a repository using --is-inside-git-dir or within a work-tree using --is-inside-work-tree
  + Checking if the repo is a bare using --is-bare-repository
  + Printing SHA1 hashes of branches (--branches), tags (--tags) and the refs can also be filtered based on the remote (using --remote)
  + --parse-opt to normalize arguments in a script (kind of similar to getopt) and print an output string that can be used with eval.
* **Sparse checkout**
  + - git clone - clone the whole repo
    - Activate the feature: git config core.sparsecheckout true
    - Add folders that are needed explicitly, ignoring assets folders: echo src/ › .git/info/sparse-checkout
    - Read the tree as specified: git read-tree -m -u HEAD
* git rev-list - Lists commit objects in reverse chronological order
  + - git rev-list - -ALL
    - git rev-list master
    - git rev-list HEAD ^origin - lists commits that are in HEAD and not in origin.
* grep, ls-tree
* archive, prune, fsck,
* gc, bisect
* **show**
  + - git show <commit ID>: filename - This will show content of file for a particular commit
* help -a
* annotate, blame, br, ci
* citool, describe difftool
* Hooks

**To Amend Last commit**

* One method
  + - git rebase —interactive ‘31f73c0^’
    - choose edit instead of default pick
    - make changes and do git add .
    - git rebase —continue
    - (This will update the current commit, but commit ID will be new)
* Second method (
  + - make changes and do git add .
    - git commit —all —amend
* Third method
  + - git rebase -i <commit ID> (edit)
    - update the contents..
    - git add .
    - git commit —amend
    - git rebase —continue
* Fourth Option
  + - git reset HEAD~
    - git add …
    - git commit -c ORIG\_HEAD

**Visual Studio**

* Git cherry-pick is supported in MS Visual Studio 2015 Update-2
* Submodules are supported in MS VS 2015 Update-2

**Bitbucket Plugin Development**

* Install Atlassian Plugin SDK
* Run command atlas-create-stash-plugin
  + - com.atlassian.stash.plugin.demoplugin
    - demo-plugin
* Run command atlas-create-stash-plugin-module
  + - 8
    - Democlass