# #_ important GIT Operations [ +100 ]

## Basic Local Operations

- `git init` - Initialize a new git repository.
- `git clone [url]` - Clone a repository into a new directory.
- `git add [file]` - Add a file to the staging area.
- `git add .` - Add all new and changed files to the staging area.
- `git commit -m "[commit message]"` - Commit changes with a message.
- `git status` - Check the status of changes as untracked, modified, or staged.
- `git diff` - Show file differences not yet staged.
- `git diff --staged` - Show file differences between staging and the last file version.
- `git reset [file]` - Unstage the file to the last commit.
- `git log` - Display the entire commit history using the default format.
- `git log --oneline` - Display the commit history in a short oneline format.
- `git log --graph` - Display an ASCII graph of the branch and merge history.
- `git checkout [branch-name]` - Switch to the specified branch and update the working directory.
- `git checkout -b [branch-name]` - Create and check out a new branch.
- `git merge [branch]` - Merge the specified branch's history into the current one.
- `git branch` - List all of the branches in your repo.
- `git branch -d [branch-name]` - Delete the specified branch.
- `git push [alias] [branch]` - Transmit local branch commits to the remote repository branch.
- `git pull [alias] [branch]` - Fetch and merge any commits from the tracking remote branch.
- `git stash` - Stash changes in a dirty working directory.
- `git stash apply` - Apply stashed changes back to the working directory.

By: Waleed Mousa

## Branching and Merging

- `git branch [branch-name]` - Create a new branch.
- `git branch -m [old-branch-name] [new-branch-name]` - Rename a local branch.
- `git branch -a` - List all local and remote branches.
- `git branch -d [branch-name]` - Safely delete a branch.
- `git branch -D [branch-name]` - Force delete a branch.
- `git merge [branch]` - Merge another branch into your active branch.
- `git merge --no-ff [branch]` - Merge with a commit even if the merge resolves as a fast-forward.
- `git merge --abort` - Abort the merge.
- `git tag [tag-name]` - Tag a specific commit with a simple, human-readable handle that never moves.

## Sharing & Updating Projects

- `git remote add [alias] [url]` - Add a git URL as an alias.
- `git fetch [alias]` - Fetch down all the branches from that Git remote.
- `git push [alias] [branch]` - Transmit local branch commits to the remote repository branch.
- `git pull` - Fetch and merge any commits from the tracking remote branch.
- `git pull --rebase [alias] [branch]` - Rebase the current branch onto [alias]/[branch] after fetching.
- `git push --tags` - Push all tags to remote repository.

## Inspection & Comparison

- `git log` - Show the commit logs.
- `git log --summary` - Show the commit logs with details about each commit.
- `git log --oneline` - Show the commit logs with each commit on a single line.

By: Waleed Mousa

- `git diff [source-branch] [target-branch]` - Preview changes before merging.

### Tracking Path Changes

- `git rm [file]` - Delete the file from the project and stage the removal for commit.
- `git mv [existing-path] [new-path]` - Change an existing file path and stage the move.
- `git log --stat -M` - Show all commit logs with indication of any paths that moved.

### Rewriting History

- `git rebase [branch]` - Reapply commits on top of another base tip.
- `git rebase --interactive [branch]` - Interactively rebase current branch onto [branch].
- `git commit --amend` - Modify the most recent commit.
- `git rebase -i HEAD~[number]` - Interactively rebase the last [number] commits.

### Temporary Commits

- `git stash` - Save modified and staged changes.
- `git stash list` - List stack-order of stashed file changes.
- `git stash pop` - Write working from top of stash stack.
- `git stash drop` - Discard the changes from top of stash stack.

### Advanced Features

- `git cherry-pick [commit]` - Apply the changes introduced by some existing commits.
- `git rebase --continue` - Continue the rebase after resolving conflicts.
- `git rebase --abort` - Abort a rebase.
- `git bisect start` - Find the commit that introduced a bug by binary search.

By: Waleed Mousa

- `git bisect bad` - Mark the current commit as bad in bisect mode.
- `git bisect good [commit]` - Mark commit as good during bisect search.
- `git bisect reset` - Exit bisect mode.

## Collaborating

- `git remote add [alias] [url]` - Add a remote repository.
- `git fetch [alias]` - Fetch branches from a remote repository.
- `git push [alias] [branch]` - Push a branch to your remote repository.
- `git pull` - Pull changes from a remote repository.
- `git push --all [alias]` - Push all branches to your remote repository.
- `git push --force [alias] [branch]` - Force push a branch to your remote repository, overwriting.
- `git push --set-upstream [alias] [branch]` - Push a branch and set the remote as upstream.
- `git fetch --prune` - Delete local branches that have been removed on the remote.

## Tagging

- `git tag [tag-name]` - Create a tag on the current commit.
- `git tag -a [tag-name] -m [message]` - Create a tagged commit.
- `git push --tags` - Push tags to the remote repository.

## Git Configurations

- `git config --global user.name [name]` - Define the author name to be used for all commits by the current user.
- `git config --global user.email [email address]` - Define the author email to be used for all commits by the current user.
- `git config --global color.ui auto` - Enable helpful colorization of command line output.

## Git Hooks

By: Waleed Mousa

- `pre-commit` - Hook script to run before each commit.
- `post-commit` - Hook script to run after each commit.
- `pre-push` - Hook script to run before each push.
- `post-checkout` - Hook script to run when checking out a new branch or commit.

### Git Attributes

- `.gitattributes` - File to define attributes per path.
- `git check-attr [attributes] [path]` - Check git attributes for a path.

### Git LFS (Large File Storage)

- `git lfs track` - Track large files within your repository.
- `git lfs untrack` - Untrack large files from your repository.

### Submodules

- `git submodule add [url] [path]` - Add a submodule.
- `git submodule init` - Initialize a submodule.
- `git submodule update` - Update the submodules.

### Advanced Merging

- `git mergetool` - Use a GUI merge tool to resolve conflicts.
- `git add [file]` - Mark file as resolved after conflicts are fixed.
- `git merge --strategy-option theirs` - Use their changes for conflicts.

### Patching

- `git format-patch [commit]` - Create a series of patch files.
- `git apply [patch]` - Apply a patch to the current branch.

### Searching

- `git grep [string]` - Search the working directory for a string.

By: Waleed Mousa

- `git log -S[string]` - Search the commit history for changes that introduce or remove a string.

## Rebasing

- `git rebase [branch]` - Reapply commits on top of another base.
- `git rebase -i` - Interactive rebase.

## Working with Remotes

- `git remote show [alias]` - Show information about a remote.
- `git remote prune [alias]` - Delete all stale tracking branches under .
- `git remote rename [old-alias] [new-alias]` - Rename a remote.

## Advanced Inspection

- `git log -p` - Show the patch representing each commit.
- `git show [commit]` - Show the metadata and content changes of the specified commit.
- `git blame [file]` - Show what revision and author last modified each line of a file.

## Working with Tags

- `git tag` - List all tags.
- `git tag -d [tag-name]` - Delete a tag in your local repository.
- `git push [alias] :refs/tags/[tag-name]` - Delete a tag from the remote repository.

## Advanced Branching

- `git branch --merged` - List branches that have been merged into the current branch.
- `git branch --no-merged` - List branches that have not been merged.
- `git branch --track [branch] [remote-branch]` - Set up a local branch that tracks a remote branch.
- `git branch -vv` - List all branches with their upstream branches.

By: Waleed Mousa

## Ignoring Patterns

- `.gitignore` - The file where you define which files and patterns should be ignored by Git.
- `git check-ignore [path]` - Check which `.gitignore` file line matches a given path.
- `git check-ignore -v [path]` - Show verbose output of check-ignore.

## Workflows

- `git flow init` - Initialize a repository with the Git Flow structure.
- `git flow feature start [feature-name]` - Start a new feature.
- `git flow feature finish [feature-name]` - Finish a feature.
- `git flow release start [release-name]` - Start a new release.
- `git flow release finish [release-name]` - Finish a release.
- `git flow hotfix start [hotfix-name]` - Start a new hotfix.
- `git flow hotfix finish [hotfix-name]` - Finish a hotfix.

## Reflog

- `git reflog` - Show a log of changes to the local repository's HEAD.
- `git reflog show [branch]` - Show a log of changes to the specified branch's HEAD.
- `git reflog expire --expire=[time] [ref]` - Purge older reflog entries.

## Housekeeping

- `git gc` - Cleanup unnecessary files and optimize the local repository.
- `git fsck` - Check the file system for errors and inconsistencies.
- `git clean -fd` - Remove untracked files and directories.

## Archive

By: Waleed Mousa

- `git archive --format=zip HEAD > [archive-name].zip` - Archive the HEAD, or any other commit, to a ZIP file.

## Advanced Git

- `git filter-branch` - Rewrite branches for complex history alterations.
- `git replace` - Replace a commit with another one.
- `git cherry` - Find commits yet to be applied to upstream.

## Server-Side Git

- `git daemon` - Start a Git daemon, allowing connections.
- `git update-server-info` - Update info files for dumb servers.
- `git instaweb` - Instantly browse your working repository in gitweb.

## Git Internals

- `git cat-file -p [SHA]` - View the contents of an object by its SHA hash.
- `git ls-tree [branch-name]` - List the contents of a tree object.
- `git show-ref` - List references in a local repository, including all heads and tags.

## Scripting and Automation

- `git bisect run [script]` - Use binary search to run a script on each commit to find the introduction of a bug.
- `git shortlog` - Summarize git log output.
- `git describe --tags` - Give an object a human-readable name based on an available ref.

## Configuration & Setup

- `git config --list` - List all variables set in config file, along with their values.
- `git config [key]` - Get the value of a configuration entry.

By: Waleed Mousa

- `git config --global [key] [value]` - Set a global configuration value.
- `git config --system [key] [value]` - Set a system-wide configuration value.

## Patch Management

- `git am [patch]` - Apply a series of patches from a mailbox.
- `git format-patch -n` - Prepare patches for e-mail submission.

## Bundling

- `git bundle create [file] [branch]` - Package objects and references for transfer.
- `git bundle verify [file]` - Check that a bundle is valid and will apply cleanly to the current repo.
- `git bundle unbundle [file]` - Unpack bundle into the repository.

## Debugging

- `git bisect` - Use binary search to find the commit that introduced a bug.
- `git blame -L [line-range] [file]` - Show what revision and author last modified each line of a file, within a given line range.

## Plumbing Commands

- `git hash-object [file]` - Compute object ID and optionally creates a blob from a file.
- `git merge-base [commit1] [commit2]` - Find a common ancestor of two commits.
- `git rev-parse` - Parse revision (or other objects) specifications.

By: Waleed Mousa