## Command Lines:-

1. **Git Basics:**

- **git init <directory>**

( create empty git repo in specified directory . Run with no arguments to initialize the current directory as a git repository)

- **git clone <repo>**
  (Clone repo located at <repo> onto local machine . Original repo can be located on the local file system or on a remote machine via HTTP or SSH.)
- **git config user.name <name>**
  (Define author name to be used for all commits in current repo. Devs community use --global flag to set config options for current user. )
- **git add <directory>**
  (Stage all changes in <directory> for the next commit . Replace <directory> with a <file> to change a specific file. )
- **git commit -m "<message>"**
  (Commit the staged snapshot , but instead of launching a text editor, use <message> as the commit message. )
- **git status**
  (List which files are staged , unstated and untracked)
- **git log**
  (Display the entire commit history using the default format. For customization see additional options. )
- **git diff**
  (Show unstaged changes between you index working directory. )

2. **Rewriting Git History:-**

- **git commit --amend**
  (Replace the last commit with the staged changes and last commit combined use with nothing staged to edit the last commit's message. )
- **git rebase <base>**
  (Rebase the current branch onto <base>, <base> can be a commit ID, branch name , a tag or a relative reference to HEAD.)
- **git reflog**
  (Show a log of changes to the local repository's HEAD.
  Add --relative-date flag to show date info or --all to show all refs.)

3. **Git Branches :-**
   - **git branch**
     (List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.)
   - **git checkout -b <branch>**
     (Create and checkout a new branch named <branch>.

Drop the -b flag to checkout an existing branch. )

- **git merge <branch>**
  (Merge <branch> into the current branch. )

4. **Remote Repositories:-**
   - **git remote add <name> <url>**
     (Create a new connection to a remote repo . After adding a remote, you can use <name> as a shortcut for <url> in other commands. )
   - **git fetch <remote> <branch>**
     (Fetches a specific <branch> ,from the repo . Leave off <branch> to fetch all all remote refs.)
   - **git pull <remote>**
     (Fetch the specified remote's copy of current branch and immediately merge it into the local copy.)
   - **git push <remote> <branch>**
     ( Push the branch to remote, along with necessary commits and objects. Create named branch in the remote repo if it doesn't exist. )

5. **Git Config :-**
   - **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>**
     (Define the author email to be used for all commits by the current user.)
   - **git config --global alias. <alias-name> <git-command>**
     (Create shortcut for a git command. E.g. alias.glog "log--graph--oneline" will set "gitglog" equivalent to "git log --graph--oneline .)
   - **git config --system core.editor <editor>**
     (Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor .)
   - **git config --global** --**edit**
     (Open the global configuration file in a text editor for manual editing. )

6. **Git log :-**
   - **git log -<limit>**
     (Limit number of commits by <limit> )
   - **git log --oneline**
     (Condense each commit to a single line. )
   - **git log -p**
     (Display the full diff of each commit. )
   - **git log --stat**
     (Include which files were altered and the relative number of lines that were added or deleted from each of them . )
   - **git log --author ="<pattern>"**
     (Search for commits with a commit message that matches <pattern> )

- **git log <since>..<until>**
  (Show commits that occur between <since> and <until> . Args can be a commit ID, branch name , HEAD or any other kind of revision reference. )
- **git log -- <file>**
  (Only display commits that have the specified file. )
- **git log --graph --decorate**
  (--graph flag draws a next based graph of commits on left side of commit msg. --decorate adds names of branches or tags of commits shown. )

**7. Git DIFF :-**

- **git diff HEAD**
  (show difference between working directory and last commit. )
- **git diff --cached**
  (Show difference between staged changes and last commit. )

**8. Git RESET :-**

- **git reset**
  (Reset staging area to match most recent commit, but leave the working directory unchanged. )
- **git reset --hard**
  (Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory. )
- **git reset <commit>**
  (Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone. )
- **git reset --hard <commit>**
  (Same as previous,  but resets both the staging area and working directory to match.  Delete uncommitted changes , and all commits after <commit. )

**9. Git Rebase :-**

- **git rebase -i <base>**
  (Interactively rebase current branch onto <branch>. Launches editor to enter commands for how each commit will be transferred to the new base. )

10**. Git Pull :-**

- **git pull --rebase <remote>**
  **(**Fetch the remote's copy of current branch and Rebase it into local copy. Use git Rebase instead of merge to integrate the branches. )

11. **Git Push :-**

- **git push <remote> --force**
  (Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you are absolutely sure you know what you're doing. )
- **git push <remote> --all**

(Push all of your local branches to the specified remote. )

- **git push <remote> --tag**
  ( tags are not automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo. )

**12. Undoing Changes:-**

- **git revert <commit>**
  (Create new commit that undoes all the changes made in <commit> , then apply it to the current branch. )
- **git reset <file>**
  (Remove <file> from the staging area , but leave the working directory unchanged. This unstages a file without overwriting any changes. )
- **git clean -n**
  (Shows which files would be removed from working directory. Use the -f flag in a place of the -n flag to execute the clean .)