Here are some commonly used Git commands:

- **git init**: Initializes a new Git repository in the current directory.
- **git clone [repository_url]**: Creates a local copy of a remote repository.
- **git add [file_name]**: Adds a file or changes to the staging area.
- **git commit -m "[commit_message]"**: Commits the changes in the staging area with a descriptive message.
- **git status**: Shows the current status of the repository, including modified files and files in the staging area.
- **git push**: Pushes the committed changes to a remote repository.
- **git pull**: Fetches the latest changes from a remote repository and merges them into the current branch.
- **git branch**: Lists all branches in the repository.
- **git branch [branch_name]**: Creates a new branch with the specified name.
- **git checkout [branch_name]**: Switches to the specified branch.
- **git merge [branch_name]**: Merges changes from the specified branch into the current branch.
- **git remote add [remote_name] [remote_url]**: Adds a remote repository with a specified name and URL.
- **git log**: Displays the commit history of the repository.
- **git diff**: Shows the differences between the working directory and the staging area.
- **git reset [file_name]**: Removes a file from the staging area.
- **git reset --hard [commit]**: Resets the repository to a specific commit, discarding all changes after that commit.

- **git stash**: Temporarily saves changes that are not ready to be committed.
- **git tag [tag_name]**: Creates a new tag for a specific commit.
- **git checkout [commit]**: Checks out a specific commit, detaching the HEAD.
- **git cherry-pick [commit]**: Applies the changes of a specific commit to the current branch.
- **git rebase [branch_name]**: Reapplies commits on top of another branch.
- **git remote remove [remote_name]**: Removes a remote repository from the local configuration.
- **git blame [file_name]**: Shows who last modified each line of a file.
- **git show [commit]**: Displays information about a specific commit.
- **git clean -n**: Shows a list of untracked files that will be deleted.
- **git config --global user.name "[name]"**: Sets the global username.
- **git config --global user.email "[email]"**: Sets the global email address.
- **git config --global alias.[alias_name] "[git_command]"**: Creates a shortcut (alias) for a Git command.

- **git reset --soft HEAD~[number]** : Reset or discared the commits and **"number"** is used for removing any number of commits. **Ex –** git reset --soft HEAD~4, it will remove the 4 commits.

Getting & Creating Projects

Command	Description
git init	Initialize a local Git repository
git clone ssh://git@github.com/[username]/[repository-name].git	Create a local copy of a remote repository

Basic Snapshotting

Command	Description
git status	Check status
git add [file-name.txt]	Add a file to the staging area
git add -A	Add all new and changed files to the staging area
git commit -m "[commit message]"	Commit changes
git rm -r [file-name.txt]	Remove a file (or folder)

Branching & Merging

Command	Description
git branch	List branches (the asterisk denotes the current branch)
git branch -a	List all branches (local and remote)
git branch [branch name]	Create a new branch
git branch -d [branch name]	Delete a branch
git push origin --delete [branch name]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git branch -m [old branch name] [new branch name]	Rename a local branch
git checkout [branch name]	Switch to a branch
git checkout -	Switch to the branch last checked out
git checkout -- [file-name.txt]	Discard changes to a file
git merge [branch name]	Merge a branch into the active branch
git merge [source branch] [target branch]	Merge a branch into a target branch
git stash	Stash changes in a dirty working directory
git stash clear	Remove all stashed entries

Sharing & Updating Projects

Command	Description
git push origin [branch name]	Push a branch to your remote repository
git push -u origin [branch name]	Push changes to remote repository (and remember the branch)
git push	Push changes to remote repository (remembered branch)
git push origin --delete [branch name]	Delete a remote branch
git pull	Update local repository to the newest commit
git pull origin [branch name]	Pull changes from remote repository
git remote add origin ssh://git@github.com/[username]/[repository-name].git	Add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	Set a repository's origin branch to SSH

Inspection & Comparison

Command	Description
git log	View changes
git log --summary	View changes (detailed)
git log --oneline	View changes (briefly)
git diff [source branch] [target branch]	Preview changes before merging

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

```
git config --global user.name "[firstname lastname]"
```
set a name that is identifiable for credit when review version history

```
git config --global user.email "[valid-email]"
```
set an email address that will be associated with each history marker

```
git config --global color.ui auto
```
set automatic command line coloring for Git for easy reviewing

SETUP & INIT

Configuring user information, initializing and cloning repositories

```
git init
```
initialize an existing directory as a Git repository

```
git clone [url]
```
retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

```
git status
```
show modified files in working directory, staged for your next commit

```
git add [file]
```
add a file as it looks now to your next commit (stage)

```
git reset [file]
```
unstage a file while retaining the changes in working directory

```
git diff
```
diff of what is changed but not staged

```
git diff --staged
```
diff of what is staged but not yet committed

```
git commit -m "[descriptive message]"
```
commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

```
git branch
```
list your branches. a * will appear next to the currently active branch

```
git branch [branch-name]
```
create a new branch at the current commit

```
git checkout
```
switch to another branch and check it out into your working directory

```
git merge [branch]
```
merge the specified branch's history into the current one

```
git log
```
show all commits in the current branch's history

INSPECT & COMPARE

Examining logs, diffs and object information

```
git log
```

show the commit history for the currently active branch

```
git log branchB..branchA
```

show the commits on branchA that are not on branchB

```
git log --follow [file]
```

show the commits that changed file, even across renames

```
git diff branchB...branchA
```

show the diff of what is in branchA that is not in branchB

```
git show [SHA]
```

show any object in Git in human-readable format

TRACKING PATH CHANGES

Versioning file removes and path changes

```
git rm [file]
```

delete the file from 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

IGNORING PATTERNS

Preventing unintentional staging or commiting of files

```
logs/
*.notes
pattern*/
```

Save a file with desired patterns as .gitignore with either direct string matches or wildcard globs.

```
git config --global core.excludesfile [file]
```

system wide ignore pattern for all local repositories

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

```
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 merge [alias]/[branch]
```

merge a remote branch into your current branch to bring it up to date

```
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

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

```
git rebase [branch]
```

apply any commits of current branch ahead of specified one

```
git reset --hard [commit]
```

clear staging area, rewrite working tree from specified commit

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

```
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

1.git help
Take help from github help section for different commands and other errors.

2.git config

To set the basic configurations on github like your name and email.

3.git config –-global user.name “Ashish Madaan”

Sets configuration values for your user name on git.

4.git config –-global user.email ashishmadaan6@gmail.com

Sets configuration values for your user email on git.

5.git config –-global color.ui true

To see different colours on command line for different outputs.

6.mkdir store

Create a directory if not created initially.

7.cd store

To go inside the directory and work upon its contents.

8.git init

To create a local git repository for us in our store folder.This will help to manage the git commands for that particular repository.

9.git status

To see whats changed since last commit.It shows all the files that have been added and modified and ready to be committed and files which are untracked.

10.git add Readme.txt

To add a file Readme.txt to the staging area to track its changes.

11.git commit -m “Created a Readme.txt”
To commit our changes(taking a snapshot) and providing a message to remember for future reference

12.git log
To check the history of commits for our reference.

Different ways to use add command:

13.git add

To add a specific list of files to staging area.

14.git add --all

To add all files of current directory to staging area.

15.git add *.txt

To add all text files of the current directory to staging area.

16.git add docs/*.txt

To add all text files of a particular directory(docs) to staging area.

17.git add docs/

To add all files in a particular directory(docs) to staging area.

18.git add “*.txt”

To add text files of entire project to staging area.

More Commands:

19.git diff

To figure out what changes you made since last commit.

20.git reset head license

To undo staging of the file that was added in the staging area.

21.git checkout –license

To Blow away all changes since the last commit of the file.

22.git commit -a -m “Readme.md”

To add any of our tracked files to staging area and commit them by providing a message to remember.

23.git reset –soft HEAD^

To undo last commit and bring file to staging area.

24.git reset –hard HEAD^

To undo last commit and remove file from the staging area as well(In case we went horribly wrong).

25.git reset –hard HEAD^^

To undo last 2 commits and all changes.

26.git remote add origin https://github.com/madaan123/MyAlgorithms.git

This commands make a bookmark which signifies that this particular remote refers to this URL. This remote will be used to pull any content from the directory and push our local content to the global server.

27.git remote add <address>

To add new remotes to our local repository for a particular git address.

28.git remove rm

To remove a remote from our local repository.

29.git push -u origin master

To push all the contents of our local repository that belong to master branch to the server(Global repository).

30.git clone https://github.com/madaan123/MyAlgorithms.git

To clone or make a local copy of the global repository in your system (git clone command downloads the repository and creates a remote named as origin which can be checked by command – git remote -v).

31.git branch Testing

To create a new branch named as Testing.

32.git branch

To see all the branches present and current branch that we are working on.

33.git checkout Testing

To switch to branch Testing from master branch.

34.ls

To see directories and files in the current directory.

35.ls -la

To see hidden directories and files with in the current directory.

36.git merge Testing

To merge Testing branch with master branch.

37.git branch -d Testing

To delete Testing branch.

38.git checkout -b admin

To create a new branch admin and set it as current branch.

39.git branch -r

To look at all the remote branches.

40.git branch -D Testing

To forcefully delete a branch without making commits.

41.git tag

To see the list of available tags.

42.git checkout v0.0.1

To set the current tag to v0.0.1.

43.git tag -a v0.0.3 -m “version 0.0.3”

To create a new tag.

44.git push –tags

To push the tags to remote repository.

45.git fetch

To fetch down any changes from global repository to current repository .

46.git stash

To move staged files to stash area which are present in staging area.

47.git stash pop

To get back the files which are present in stash area.

48.git stash clear

To clear the stash folder.

49.git rebase