Git / GitHub

Basic repo making commands:

```
echo "# Repo-Name" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/User-Name/Repo-Name.
gitgit push -u origin main
```

All Git/GitHub Commands

List of Git Commands with Explanation

No.	Command	Syntax	Function
1	`git init`	`git init`	Initializes a new Git repository in the current directory.
2	`git clone <repository-url>`	`git clone <repository-url>`	Clones an existing repository from a URL to your local machine.
3	`git status`	`git status`	Shows the current state of the working directory and staging area.
4	`git add <file>`	`git add <file>`	Stages a specific file (or files) for commit.

5	`git add .`	`git add .`	Stages all modified or new files in the current directory for commit.
6	`git commit -m "<message>"`	`git commit -m "<message>"`	Commits the staged changes with a descriptive message.
7	`git commit --amend`	`git commit --amend`	Modifies the most recent commit, allowing changes to the message or files.
8	`git log`	`git log`	Displays the commit history in reverse chronological order.
9	`git branch`	`git branch`	Lists all branches in the repository, with the current branch highlighted.
10	`git branch <branch-name>`	`git branch <branch-name>`	Creates a new branch with the given name.
11	`git checkout <branch-name>`	`git checkout <branch-name>`	Switches to the specified branch in the repository.
12	`git checkout -b <branch-name>`	`git checkout -b <branch-name>`	Creates and switches to a new branch in one command.
13	`git merge <branch-name>`	`git merge <branch-name>`	Merges the changes from the specified branch into the current branch.
14	`git pull`	`git pull`	Fetches changes from a remote repository and merges them into the current branch.
15	`git push`	`git push`	Pushes committed changes from the local repository to the remote repository.
16	`git push origin <branch-name>`	`git push origin <branch-name>`	Pushes a specific branch to the remote repository.
17	`git remote add origin <url>`	`git remote add origin <url>`	Adds a new remote repository with the given

			URL (usually named `origin`).
18	`git remote set-url origin <url>`	`git remote set-url origin <url>`	Changes the URL of the existing `origin` remote repository.
19	`git fetch`	`git fetch`	Downloads changes from the remote repository but doesn't merge them.
20	`git reset`	`git reset <commit-hash>`	Resets the repository to a previous commit, affecting the staging area and working directory.
21	`git rm <file>`	`git rm <file>`	Removes a file from both the staging area and the working directory.
22	`git stash`	`git stash`	Temporarily saves changes that aren't ready to be committed.
23	`git stash pop`	`git stash pop`	Applies the most recent stashed changes and removes them from the stash.
24	`git stash list`	`git stash list`	Lists all stashed changes.
25	`git stash drop`	`git stash drop`	Removes a specific stash from the list of stashes.
26	`git tag <tag-name>`	`git tag <tag-name>`	Creates a tag with the given name pointing to the current commit.
27	`git show <tag-name>`	`git show <tag-name>`	Displays information about a specific tag (commit, date, etc.).
28	`git diff`	`git diff`	Shows the differences between the working directory and the index (staging area).

29	`git diff <commit-hash>`	`git diff <commit-hash>`	Shows differences between a commit and the current working directory.
30	`git remote -v`	`git remote -v`	Displays the URLs of all remotes configured in the repository.
31	`git config --global user.name "<name>"`	`git config --global user.name "<name>"`	Sets the global username for commits across all repositories.
32	`git config --global user.email "<email>"`	`git config --global user.email "<email>"`	Sets the global email for commits across all repositories.
33	`git cherry-pick <commit-hash>`	`git cherry-pick <commit-hash>`	Applies the changes from a specific commit onto the current branch.
34	`git clean -f`	`git clean -f`	Removes untracked files from the working directory.

Git Command Breakdown

1. `git init`

Syntax: `git init`

Breakdown:

- `git`: The Git command-line tool.
- `init`: Initializes a new repository in the current directory.

Function:

Initializes a new Git repository in the current directory.

---

2. `git clone <repository-url>`

Syntax: `git clone <repository-url>`

Breakdown:

- `git`: The Git command-line tool.
- `clone`: Copies an existing remote repository to the local machine.
- `<repository-url>`: The URL of the repository to be cloned.

Function:

Clones an existing repository from a URL to your local machine.

---

3. `git status`

Syntax: `git status`

Breakdown:

- `git`: The Git command-line tool.
- `status`: Shows the status of changes in the working directory and staging area.

Function:

Displays the current state of the working directory and staging area.

---

4. `git add <file>`

Syntax: `git add <file>`

Breakdown:

- `git`: The Git command-line tool.
- `add`: Adds changes in a file to the staging area.
- `<file>`: The file or files to be staged for commit.

Function:

Stages a specific file for commit.

---

5. `git add .`

Syntax: `git add .`

Breakdown:

- `git`: The Git command-line tool.
- `add`: Adds changes in the working directory to the staging area.
- `.`: Stages all files in the current directory (including subdirectories).

Function:

Stages all modified or new files in the current directory for commit.

---

6. `git commit -m "<message>"`

Syntax: `git commit -m "<message>"`

Breakdown:

- `git`: The Git command-line tool.
- `commit`: Commits the staged changes.
- `m`: Option to include a commit message inline.
- `"<message>"`: The commit message describing the changes.

Function:

Commits the staged changes with a message.

---

7. `git commit --amend`

Syntax: `git commit --amend`

Breakdown:

- `git`: The Git command-line tool.

- `commit`: Commits the staged changes.
- `-amend`: Modifies the most recent commit, allowing changes to the commit message or staged files.

Function:

Modifies the most recent commit (message or files).

---

8. `git log`

Syntax: `git log`

Breakdown:

- `git`: The Git command-line tool.
- `log`: Displays the commit history of the repository.

Function:

Shows the commit history in reverse chronological order.

---

9. `git branch`

Syntax: `git branch`

Breakdown:

- `git`: The Git command-line tool.
- `branch`: Lists all the branches in the repository.

Function:

Lists all branches in the repository.

---

10. `git branch <branch-name>`

Syntax: `git branch <branch-name>`

Breakdown:

- `git`: The Git command-line tool.
- `branch`: Creates a new branch.
- `<branch-name>`: The name of the new branch.

Function:

Creates a new branch with the specified name.

---

11. `git checkout <branch-name>`

Syntax: `git checkout <branch-name>`

Breakdown:

- `git`: The Git command-line tool.
- `checkout`: Switches to the specified branch.
- `<branch-name>`: The name of the branch to switch to.

Function:

Switches to the specified branch in the repository.

---

12. `git checkout -b <branch-name>`

Syntax: `git checkout -b <branch-name>`

Breakdown:

- `git`: The Git command-line tool.
- `checkout`: Switches to the specified branch.
- `b`: Option that creates the branch and switches to it.
- `<branch-name>`: The name of the new branch.

Function:

Creates and switches to a new branch in one command.

---

13. `git merge <branch-name>`

Syntax: `git merge <branch-name>`

Breakdown:

- `git`: The Git command-line tool.
- `merge`: Combines the changes from another branch into the current branch.
- `<branch-name>`: The name of the branch to merge into the current branch.

Function:

Merges changes from a specified branch into the current branch.

---

14. `git pull`

Syntax: `git pull`

Breakdown:

- `git`: The Git command-line tool.
- `pull`: Fetches and merges changes from a remote repository.

Function:

Fetches and merges changes from the remote repository into the current branch.

---

15. `git push`

Syntax: `git push`

Breakdown:

- `git`: The Git command-line tool.
- `push`: Pushes local commits to a remote repository.

Function:

Pushes committed changes from the local repository to the remote repository.

---

16. `git push origin <branch-name>`

Syntax: `git push origin <branch-name>`

Breakdown:

- `git`: The Git command-line tool.
- `push`: Pushes commits to a remote repository.
- `origin`: The default name of the remote repository.
- `<branch-name>`: The name of the branch to push.

Function:

Pushes a specific branch to the remote repository.

---

17. `git remote add origin <url>`

Syntax: `git remote add origin <url>`

Breakdown:

- `git`: The Git command-line tool.
- `remote`: Manages remote repositories.
- `add`: Adds a new remote.
- `origin`: The default name for the remote repository.
- `<url>`: The URL of the remote repository.

Function:

Adds a new remote repository with the given URL.

---

18. `git remote set-url origin <url>`

Syntax: `git remote set-url origin <url>`

Breakdown:

- `git`: The Git command-line tool.
- `remote`: Manages remote repositories.
- `set-url`: Changes the URL of an existing remote.
- `origin`: The default name of the remote repository.
- `<url>`: The new URL for the remote repository.

Function:

Changes the URL of the `origin` remote repository.

---

19. `git fetch`

Syntax: `git fetch`

Breakdown:

- `git`: The Git command-line tool.
- `fetch`: Downloads changes from the remote repository without merging them.

Function:

Fetches changes from the remote repository without merging them into the current branch.

---

20. `git reset`

Syntax: `git reset <commit-hash>`

Breakdown:

- `git`: The Git command-line tool.
- `reset`: Resets the current branch to a specific commit.
- `<commit-hash>`: The hash of the commit to reset to.

Function:

Resets the repository to a previous commit, affecting the staging area and working directory.

---

21. `git rm <file>`

Syntax: `git rm <file>`

Breakdown:

- `git`: The Git command-line tool.
- `rm`: Removes a file.
- `<file>`: The file to be removed from both the working directory and the staging area.

Function:

Removes a file from both the staging area and the working directory.

---

22. `git stash`

Syntax: `git stash`

Breakdown:

- `git`: The Git command-line tool.
- `stash`: Temporarily saves changes that aren't committed yet.

Function:

Temporarily saves changes that aren't ready to be committed, allowing you to work on something else.

---

23. `git stash pop`

Syntax: `git stash pop`

Breakdown:

- `git`: The Git command-line tool.
- `stash`: Refers to the temporary saved changes.
- `pop`: Applies the most recent stash and removes it from the stash list.

Function:

Applies the most recent stash and removes it from the stash list.

---

24. `git stash list`

Syntax: `git stash list`

Breakdown:

- `git`: The Git command-line tool.
- `stash`: Refers to the stashes.
- `list`: Lists all the stashes stored in the repository.

Function:

Lists all stashed changes in the repository.

---

25. `git stash drop`

Syntax: `git stash drop`

Breakdown:

- `git`: The Git command-line tool.
- `stash`: Refers to the saved changes.
- `drop`: Removes a specific stash from the stash list.

Function:

Removes a specific stash from the list.

---

26. `git tag <tag-name>`

Syntax: `git tag <tag-name>`

Breakdown:

- `git`: The Git command-line tool.
- `tag`: Creates a tag.
- `<tag-name>`: The name of the tag.

Function:

Creates a tag at the current commit.

---

27. `git show <tag-name>`

Syntax: `git show <tag-name>`

Breakdown:

- `git`: The Git command-line tool.
- `show`: Displays information about the tag.
- `<tag-name>`: The name of the tag to display information about.

Function:

Displays detailed information about a specific tag.

---

28. `git diff`

Syntax: `git diff`

Breakdown:

- `git`: The Git command-line tool.
- `diff`: Shows changes between commits, the working directory, and the staging area.

Function:

Displays the differences between files that have been modified, but not yet committed.

---

29. `git diff <commit1> <commit2>`

Syntax: `git diff <commit1> <commit2>`

Breakdown:

- `git`: The Git command-line tool.
- `diff`: Shows the differences between two commits.
- `<commit1>` and `<commit2>`: The commits to compare.

Function:

Shows the differences between two specific commits.

---

30. `git config --global user.name "<name>"`

Syntax: `git config --global user.name "<name>"`

Breakdown:

- `git`: The Git command-line tool.
- `config`: Modifies Git configuration.
- `-global`: Applies the configuration globally (for all repositories).
- `user.name`: The configuration key for setting the user's name.
- `"<name>"`: The user's name to associate with commits.

Function:

Sets the global username for your Git commits.

---

31. `git config --global user.email "<email>"`

Syntax: `git config --global user.email "<email>"`

Breakdown:

- `git`: The Git command-line tool.
- `config`: Modifies Git configuration.
- `-global`: Applies the configuration globally (for all repositories).
- `user.email`: The configuration key for setting the user's email.
- `"<email>"`: The user's email address to associate with commits.

Function:

Sets the global email address for your Git commits.

---

32. `git remote -v`

Syntax: `git remote -v`

Breakdown:

- `git`: The Git command-line tool.
- `remote`: Manages remote repositories.
- `v`: Displays the URL of each remote repository.

Function:

Shows the URL of the remote repositories associated with the current local repository.

---

33. `git remote rm <name>`

Syntax: `git remote rm <name>`

Breakdown:

- `git`: The Git command-line tool.
- `remote`: Manages remote repositories.
- `rm`: Removes a remote repository.

- `<name>`: The name of the remote repository (e.g., `origin`).

Function:

Removes the specified remote repository from the configuration.

---

34. `git reflog`

Syntax: `git reflog`

Breakdown:

- `git`: The Git command-line tool.
- `reflog`: Shows a log of all the references in the repository, including changes to the HEAD.

Function:

Displays a history of all the changes to the HEAD, useful for recovering lost commits.