Important GIT Commands - By Naveen Auto...

Here is a list of some important Git commands with examples:

1. **git init**: This command is used to initialize a new Git repository in the current directory. For example, you can use the following command to initialize a new Git repository in the **my-project** directory:

Plain text ▾ •••

```
$ git init my-project
```

2. **git clone**: This command is used to clone an existing Git repository from a remote source. For example, you can use the following command to clone the **my-project** repository from the https://github.com/user/my-project.git URL:

Plain text ▾ •••

```
$ git clone https://github.com/user/my-project.git
```

3. **git add**: This command is used to add files to the staging area in a Git repository. For example, you can use the following command to add the **file1.txt** and **file2.txt** files to the staging area:

Plain text ▾ •••

```
$ git add file1.txt file2.txt
```

4. **git commit**: This command is used to commit the changes in the staging area to the Git repository. For example, you can use the following command to commit the changes in the staging area with the commit message "Added new files":

Plain text ▾ •••

```
$ git commit -m "Added new files"
```

5. **git push**: This command is used to push the committed changes in the local repository to the remote repository. For example, you can use the following command to push the committed

changes to the **origin** remote branch:

Plain text ▾ •••

```
$ git push origin
```

6. **git pull**: This command is used to pull the changes from the remote repository to the local repository. For example, you can use the following command to pull the changes from the **origin** remote branch:

Plain text ▾ •••

```
$ git pull origin
```

7. **git branch**: This command is used to list, create, or delete branches in a Git repository. For example, you can use the following command to create a new branch called **my-branch**:

Plain text ▾ •••

```
$ git branch my-branch
```

8. **git checkout**: This command is used to switch between branches in a Git repository. For example, you can use the following command to switch to the **my-branch** branch:

Plain text ▾ •••

```
$ git checkout my-branch
```

9. **git log**: This command is used to display the commit history of a Git repository. For example, you can use the following command to view the commit history with the author and date information:

Shell ▾ •••

```
$ git log --pretty=format:"%h %an %ad"
```

10. **git diff**: This command is used to view the differences between two versions of a file in a Git repository. For example, you can use the following command to view the differences between the **file1.txt** file in the **HEAD** and the **my-branch** branch:

Plain text ▾ •••

```
$ git diff HEAD my-branch file1.txt
```

11. **git stash**: This command is used to save the local changes in a Git repository without committing them. For example, you can use the following command to stash the local changes:

Plain text ▾ •••

```
$ git stash
```

12. **git tag**: This command is used to add tags to specific commits in a Git repository. For example, you can use the following command to add a tag called **v1.0** to the latest commit:

Plain text ▾ •••

```
$ git tag v1.0
```

13. **git merge**: This command is used to merge one branch into another branch in a Git repository. For example, you can use the following command to merge the **my-branch** branch into the **master** branch:

Plain text ▾ •••

```
$ git merge my-branch
```

14. **git reset**: This command is used to reset the state of a Git repository to a previous commit. For example, you can use the following command to reset the repository to the **HEAD** commit:

Plain text ▾ •••

```
$ git reset --hard HEAD
```

15. **git config**: This command is used to configure settings for a Git repository. For example, you can use the following command to set the user name and email address for the repository:

Plain text

```
$ git config --global user.name "John Doe"
$ git config --global user.email "john.doe@example.com"
```

16. **git remote**: This command is used to manage the remote repositories for a Git repository. For example, you can use the following command to add a new remote repository called **origin**:

Plain text

```
$ git remote add origin https://github.com/user/my-project.g
```

17. **git fetch**: This command is used to download the objects and references from a remote repository to the local repository. For example, you can use the following command to fetch the objects and references from the **origin** remote repository:

Plain text

```
$ git fetch origin
```

18. **git gc**: This command is used to clean up unnecessary files and optimize the Git repository. For example, you can use the following command to run the garbage collector:

Plain text

```
$ git gc
```

19. **git blame**: This command is used to view the commit history for each line of a file in a Git repository. For example, you can use the following command to view the commit history for the **file1.txt** file:

Plain text

```
$ git blame file1.txt
```

20. **git rev-parse**: This command is used to parse revision names and extract information from the revision names. For example, you can use the following command to get the abbreviated commit hash for the **HEAD** commit:

Plain text

```
$ git rev-parse --short HEAD
```

21. **git show**: This command is used to show the details of a specific commit in a Git repository. For example, you can use the following command to show the details of the **HEAD** commit:

Plain text

```
$ git show HEAD
```

22. **git clean**: This command is used to remove untracked files from a Git repository. For example, you can use the following command to remove all untracked files from the repository:

Plain text

```
$ git clean -df
```

23. **git grep**: This command is used to search for a specific pattern in the files in a Git repository. For example, you can use the following command to search for the **foo** pattern in the **file1.txt** file:

Plain text

```
$ git grep foo file1.txt
```

24. **git submodule**: This command is used to manage submodules in a Git repository. For example, you can use the following command to add a submodule called **my-module** from the **https://github.com/user/my-module.git** URL:

Plain text

```
git submodule add https://github.com/user/my-module.git my-mc
```

25. **git bisect**: This command is used to perform a binary search through the commit history of a Git repository to find a specific commit. For example, you can use the following command to start a bisect search for the **buggy** commit:

Plain text

```
$ git bisect start
$ git bisect bad
$ git bisect good buggy
```

26. **git fsck**: This command is used to verify the integrity of the objects in a Git repository. For example, you can use the following command to verify all the objects in the repository:

Plain text

```
$ git fsck --full
```

27. **git cherry-pick**: This command is used to apply the changes from a specific commit to the current branch in a Git repository. For example, you can use the following command to apply the changes from the **f46f5e5** commit to the current branch:

Plain text

```
$ git cherry-pick f46f5e5
```

28. **git rebase**: This command is used to reapply the commits from a branch on top of another branch in a Git repository. For example, you can use the following command to rebase the **my-branch** branch onto the **master** branch:

Plain text

```
$ git rebase master my-branch
```

29. **git mv**: This command is used to move or rename a file in a Git repository. For example, you can use the following command to rename the **file1.txt** file to **file2.txt**:

Plain text

```
$ git mv file1.txt file2.txt
```

30. **git ls-files**: This command is used to list the files in the index and the working tree of a Git repository. For example, you can use the following command to list all the files in the repository:

Plain text

```
$ git ls-files
```