> Git Initialization, Add, Commit, and Push Workflow

1. **Initialize a New Repository**
   ```
   git init
   ```
   Use: Initializes a new Git repository in the current directory.
2. **Add Files to Staging Area**
   ```
   git add <file_name>
   ```
   OR to add all files:
   ```
   git add .
   ```
   Use: Adds the specified files or all changes to the staging area.
3. **Commit Changes**
   ```
   git commit -m "Commit message"
   ```
   Use: Records the staged changes with a message describing the changes.
4. **Add a Remote Repository**
   ```
   git remote add origin <repository_url>
   ```
   Use: Links the local repository to a remote repository.
5. **Push Changes to Remote**
   ```
   git push origin <branch_name>
   ```
   Use: Pushes committed changes from the specified branch to the remote repository.
6. **Clone an Existing Repository**
   ```
   git clone <repository_url>
   ```
   Use: Creates a local copy of a remote repository.
7. **Check Repository Status**
   ```
   git status
   ```
   Use: Displays the status of the working directory and staging area.

> Git Branch Commands and Their Uses

Basic Branch Commands

1. **Create a New Branch**
   `git branch <branch_name>`
   Use: Creates a new branch from the current branch without switching to it.
2. **List All Branches**
   `git branch`
   Use: Lists all local branches. The active branch is highlighted with an asterisk (*).
3. **Switch to a Branch**
   `git checkout <branch_name>`
   OR (modern method):
   `git switch <branch_name>`
   Use: Switches to an existing branch.
4. **Create and Switch to a New Branch**
   `git checkout -b <branch_name>`
   OR (modern method):
   `git switch -c <branch_name>`
   Use: Creates a new branch and switches to it immediately.
5. **Delete a Branch**
   `git branch -d <branch_name>`
   Use: Deletes a branch if it has been merged.
   For **force deletion**:
   `git branch -D <branch_name>`

Remote Branch Commands

6. **List Remote Branches**
   `git branch -r`
   Use: Lists all branches available on the remote repository.
7. **List Local and Remote Branches**
   `git branch -a`
   Use: Displays both local and remote branches.

8. **Push a Branch to Remote**
   `git push origin <branch_name>`
   Use: Pushes the local branch to the remote repository.
9. **Delete a Remote Branch**
   `git push origin --delete <branch_name>`
   Use: Deletes a branch from the remote repository.
10. **Track a Remote Branch Locally**
    `git checkout --track origin/<branch_name>`
    Use: Creates a local branch that tracks the specified remote branch.

Merging and Rebase Commands

11. **Merge a Branch**
    `git merge <branch_name>`
    Use: Merges the specified branch into the current branch.
12. **Rebase a Branch**
    `git rebase <branch_name>`
    Use: Reapplies commits from the current branch on top of the specified branch.

Branch Cleanup and Information

13. **Rename a Branch**
    `git branch -m <new_branch_name>`
    Use: Renames the current branch.
    To rename another branch:
    `git branch -m <old_branch_name> <new_branch_name>`
14. **Show the Branch You're On**
    `git branch --show-current`
    Use: Displays the name of the current branch.
15. **Find Out Which Branches Have Been Merged**
    `git branch --merged`
    Use: Shows branches that have been merged into the current branch.
16. **Find Out Which Branches Are Not Merged**
    `git branch --no-merged`
    Use: Shows branches that haven't been merged into the current branch.

17. **Prune Remote-Tracking Branches**
    ```
    git fetch --prune
    ```
    Use: Cleans up stale references to remote branches.

Visualizing Branches

18. **Show a Graph of Branches**
    ```
    git log --oneline --graph --all
    ```
    Use: Displays a graphical view of the commit history across branches.
19. **Show Branch Tracking Information**
    ```
    git remote show origin
    ```
    Use: Shows detailed information about remote branches and their tracking status.

> Some Additional Topics

1. **Visualizing Changes with Git Diff and GUI Tools**

   `git diff`

   Use: Compares file changes between commits or working directories. GUI tools like GitKraken or SourceTree provide visual diff representation.
2. **Stashing and Temporary Changes**

   `git stash`

   Use: Temporarily saves uncommitted changes without committing them to the branch.
3. **Exploring Old Commits and Undoing Changes**

   `git log` and `git checkout <commit_hash>`

   Use: Views commit history or reverts to previous commits for debugging or undoing changes.
4. **Introduction to GitHub**

   Use GitHub to host repositories and collaborate on code development.
5. **GitHub Contributions and Pull Requests**

   Use: Create pull requests to propose changes and collaborate with contributors.
6. **Managing Organizations on GitHub**

   Use: Handle permissions and repositories for teams or organizations.
7. **Reviewing Commit History**

   `git log --stat`

   Use: Reviews detailed commit history with file change statistics.
8. **Interactive Staging and Selective Changes**

   `git add -p`

   Use: Stages selected portions of file changes interactively.
9. **Merging and Resolving Conflicts**

   `git merge <branch_name>` and manual resolution

   Use: Combines changes from different branches and resolves conflicts.

10. **Rebasing and Managing Changes**

    `git rebase <branch_name>`

    Use: Rewrites commit history for cleaner change integration.

11. **Introduction to Version Control and Git, GitHub**

    Learn version control basics and how Git and GitHub streamline collaboration.

12. **The Command Line and Initial Setup**

    `git config`

    Use: Configures user details, aliases, and preferences.

13. **Git Repositories and Tracking Changes**

    `git init` and `git add`

    Use: Initializes repositories and tracks files for version control.

14. **Commit History and Viewing Changes**

    `git log` and `git diff`

    Use: Tracks commit history and compares changes.

15. **Branch Management**

    Manage branches using commands like `git branch` and `git checkout`.

16. **Working with Remote Repositories**

    `git push` and `git pull`

    Use: Synchronizes local changes with remote repositories.

17. **Rebasing and Advanced Commands**

    Advanced git workflows using `rebase` and other commands.

18. **Ignoring Files and Git Aliases**

    `.gitignore` and `git config`

    Use: Excludes unnecessary files from version control and creates custom shortcuts.

19. **Merging and Resolving Conflicts**

    `git merge` and conflict resolution techniques for collaboration.
