**Git Command Guide**

**1. Repository Management**

**Description:** These commands are used to create, clone, manage, and configure Git repositories.

* **Create a new repository:** Initializes a new Git repository in your current directory.

git init

* **Clone an existing repository:** Copies a remote repository to your local system.

git clone <repository-URL>

* **Show current repository details:** Displays the remote URLs associated with the repository.

git remote -v

* **Add a remote repository:** Links your local repository to a remote repository.

git remote add origin <repository-URL>

* **Rename a remote repository:** Changes the alias name of the remote repository.

git remote rename origin new-origin-name

* **Remove a remote repository:** Deletes the reference to a remote repository from your local configuration.

git remote remove origin

**2. File Management**

**Description:** Commands to manage files in a repository, stage them, or revert changes.

* **Add a file to staging:** Prepares a specific file for the next commit.

git add <file-name>

* **Add all files to staging:** Stages all modified and new files.

git add .

* **Remove a tracked file:** Deletes a file from the repository and stages the removal.

git rm <file-name>

* **Undo file changes:** Reverts changes to a file in the working directory to the last committed state.

git checkout -- <file-name>

* **Unstage a file:**

git reset <file-name>

**3. Branch Management**

**Description:** Commands for creating, switching, managing, and deleting branches.

* **Create a new branch:** Creates a new branch to work on a specific feature or fix.

git branch <branch-name>

* **Switch to an existing branch:** Changes the working branch to the specified branch.

git checkout <branch-name>

* **Create and switch to a new branch:** Combines branch creation and switching into one command.

git checkout -b <branch-name>

* **Show all branches:** Lists local and remote branches.

git branch -a

* **Rename a branch:** Updates the name of a branch.

git branch -m <new-branch-name>

* **Delete a branch:** Removes a branch locally.

git branch -d <branch-name>

* **Delete a remote branch:** Deletes a branch from the remote repository.

git push origin --delete <branch-name>

**4. Commit Management**

**Description:** Commands for creating and managing commits, which capture changes in a repository.

* **Create a commit:** Saves changes in the staging area to the repository.

git commit -m "Commit message"

* **Amend the last commit:** Updates the most recent commit with new changes.

git commit --amend

* **Show commit history:** Displays a list of past commits.

git log

* **Show a specific commit:** Displays details of a specified commit.

git show <commit-hash>

* **Undo the last commit (keep changes unstaged):**

git reset HEAD~

* **Revert a specific commit:** Creates a new commit that undoes a specific commit.

git revert <commit-hash>

* **View a summary of commits:**

git log --oneline

**5. Synchronization**

**Description:** Commands for synchronizing changes between local and remote repositories.

* **Push changes:** Uploads changes from your local branch to the remote repository.

git push origin <branch-name>

* **Fetch changes:** Downloads updates from the remote repository but does not merge them.

git fetch

* **Pull changes:** Combines fetching and merging remote changes into the current branch.

git pull origin <branch-name>

**6. Merging and Rebasing**

**Description:** Commands for integrating changes from one branch into another.

* **Merge a branch:** Combines changes from a specified branch into the current branch.

git merge <branch-name>

* **Resolve merge conflicts:** Manually resolves conflicting changes after a merge.
* git add <file-name>

git commit

* **Rebase a branch:** Reapplies commits from one branch on top of another, creating a cleaner commit history.

git rebase <branch-name>

**7. Stashing**

**Description:** Temporarily stores changes not ready for commit, allowing you to work on other tasks.

* **Stash changes:** Saves changes without committing them.

git stash

* **Apply stashed changes:** Restores saved changes without removing them from the stash.

git stash apply

* **Show stashed changes:** Lists all stashed entries.

git stash list

* **Delete a specific stash:** Removes a specific stash entry.

git stash drop <stash-id>

* **Retrieve and apply a specific stash:**

git stash apply <stash-id>

**8. Comparison and Inspection**

**Description:** Commands to inspect and compare changes in a repository.

* **Show differences between commits:** Displays changes between two commits.

git diff <commit1> <commit2>

* **Compare local and remote branches:** Shows differences between a local branch and its remote counterpart.

git diff <branch-name> origin/<branch-name>

* **Show who changed each line:** Displays the author and commit associated with each line in a file.

git blame <file-name>

**9. Git Configuration**

**Description:** Commands to configure global or local settings in Git.

* **Set global username:** Configures the global username for commits.

git config --global user.name "Your Name"

* **Set global email:** Sets the email used for commits.

git config --global user.email "your.email@example.com"

* **View current configuration:** Lists all current Git configuration settings.

git config --list

* **Edit Git configuration:**

git config --global --edit

**10. Practical Use Cases**

**Description:** Real-world scenarios for utilizing Git commands effectively.

* **Switch to a previous commit:** Temporarily checks out a specific commit.

git checkout <commit-hash>

* **Remove a file from the last commit:** Unstages a file from the most recent commit.

git reset HEAD~ <file-name>

* **Create a tag:** Marks a specific commit for reference.

git tag <tag-name>

* **Push tags to remote:** Uploads local tags to the remote repository.

git push origin --tags

* **List all tags:**

git tag

**11. How to Practice**

**Description:** Steps for practicing Git commands effectively.

* **Create a demo project:**
  + Initialize a repository (git init).
  + Create multiple branches and work on features (git branch).
  + Commit changes (git add, git commit).
* **Experiment with remote repositories:**
  + Clone a repository from GitHub (git clone).
  + Push changes and observe their effect (git push).
* **Handle scenarios:**
  + Simulate merge conflicts and resolve them.
  + Practice rebasing (git rebase).
  + Use stash commands when switching tasks.