**GitHub Guide: Commands, Scenarios, and Solutions**

**Basic Operations**

1. **Undo the Last Commit but Keep Changes in the Working Directory**
   * Use this command if you mistakenly committed something but want to edit the commit or add more changes.

git reset --soft HEAD~1

* + **Explanation:** This resets the last commit while keeping changes staged (soft reset). You can make adjustments and re-commit.

1. **Discard Changes in a File**
   * Useful when you want to revert a file to the last committed version.

git checkout -- <file>

* + **Explanation:** This discards local changes in the file and resets it to its last committed state.

1. **Stage All Changes**
   * Quickly stage all modified files in the repository.

git add .

* + **Explanation:** The . adds all changes (new, modified, deleted files) to the staging area.

1. **Delete a File and Stage the Deletion**
   * Deletes a file and stages the change in one step.

git rm <file>

* + **Explanation:** Ensures Git knows the file was removed and includes it in the next commit.

1. **Check the Current Branch**
   * Shows the branch you’re currently working on.

git branch

* + **Explanation:** The current branch will be highlighted with an asterisk (\*).

**Branching**

1. **Rename a Branch**
   * Changes the branch name without affecting history.

git branch -m <new-branch-name>

* + **Explanation:** This renames the current branch without creating a new branch.

1. **Delete a Remote Branch**
   * Use this when a branch is no longer needed remotely.

git push origin --delete <branch-name>

* + **Explanation:** This removes the branch from the remote repository.

1. **Create and Switch to a New Branch**
   * Create a new branch and start working on it immediately.

git checkout -b <branch-name>

* + **Explanation:** Combines branch creation and checkout into one step.

1. **List All Branches (Local and Remote)**

git branch -a

* + **Explanation:** This displays all branches, including remote-tracking branches.

1. **Set Up Tracking for a New Remote Branch**
   * Links a local branch to its remote counterpart for easy pushing and pulling.

git branch --set-upstream-to=origin/<branch-name> <branch-name>

* + **Explanation:** Enables commands like git pull to work without needing explicit arguments.

**Merging and Rebase**

1. **Resolve Merge Conflicts**
   * After a conflict during a merge:
   * git add <file>
   * git commit
   * **Explanation:** Edit the conflicting file(s), stage the resolved changes, and commit them to complete the merge.
2. **Abort a Merge**
   * Cancel an ongoing merge and return to the pre-merge state.

git merge --abort

* + **Explanation:** Useful if you want to discard the merge and start over.

1. **Rebase a Branch**
   * Incorporate the latest changes from another branch into your current branch.

git rebase <branch-name>

* + **Explanation:** Keeps a clean linear history by replaying your commits on top of the other branch.

1. **Continue a Rebase After Conflict Resolution**

git rebase --continue

* + **Explanation:** After resolving conflicts during a rebase, this command resumes the process.

1. **Squash Commits During a Rebase**
   * Combine multiple commits into one.

git rebase -i HEAD~<number-of-commits>

* + **Explanation:** Use an interactive rebase to mark commits as "squash."

**Remote Repositories**

1. **Add a Remote Repository**

git remote add origin <repository-url>

* + **Explanation:** Links your local repository to a remote repository.

1. **View Remote Repositories**

git remote -v

* + **Explanation:** Displays the URLs of all remote repositories.

1. **Change the URL of a Remote Repository**

git remote set-url origin <new-repository-url>

* + **Explanation:** Updates the remote repository URL without removing it.

1. **Fetch Changes from a Remote Repository**

git fetch origin

* + **Explanation:** Downloads changes from the remote repository without integrating them.

1. **Delete a Remote Repository Reference**

git remote rm origin

* + **Explanation:** Removes the connection to a remote repository.

**Stashing**

1. **Stash Changes**

git stash

* + **Explanation:** Temporarily saves changes to your working directory without committing them.

1. **List Stashes**

git stash list

* + **Explanation:** Shows all stashed changes with IDs.

1. **Apply the Latest Stash**

git stash apply

* + **Explanation:** Reapplies the most recent stash without deleting it.

1. **Drop a Stash**

git stash drop

* + **Explanation:** Deletes a specific stash from the list.

1. **Apply and Drop a Stash Simultaneously**

git stash pop

* + **Explanation:** Reapplies the stash and removes it from the stash list.

**Logs and Debugging**

1. **View a Specific Commit**

git show <commit-hash>

* + **Explanation:** Displays details and changes made in a specific commit.

1. **View Changes Made in the Last Commit**

git diff HEAD~1 HEAD

* + **Explanation:** Compares the latest commit with the one before it.

1. **Find Who Changed a Specific Line in a File**

git blame <file>

* + **Explanation:** Shows line-by-line authorship.

1. **Search Commit History for a Keyword**

git log --grep="keyword"

* + **Explanation:** Filters the log for commits containing the keyword in their message.

1. **Reset a File to a Specific Commit**

git checkout <commit-hash> -- <file>

* + **Explanation:** Restores the file to its state at the specified commit.

1. **Cherry-Pick a Commit from Another Branch**

git cherry-pick <commit-hash>

* + **Explanation:** Copies a specific commit from one branch to another.

1. **Amend the Last Commit**

git commit --amend

* + **Explanation:** Allows you to modify the last commit message or files.

1. **Recover a Deleted Branch**

git reflog

git checkout -b <branch-name> <commit-hash>

* + **Explanation:** Restores a branch using its commit history.

1. **Squash All Commits into One**

git reset --soft <initial-commit-hash>

git commit -m "New initial commit"

* + **Explanation:** Combines all changes into a single commit.

1. **Bisect to Find a Bug**

git bisect start

git bisect bad

git bisect good <commit-hash>

* + **Explanation:** Uses binary search to find the commit introducing a bug.

1. **Create a Patch from a Commit**

git format-patch -1 <commit-hash>

* + **Explanation:** Generates a .patch file for sharing or applying changes.

1. **Apply a Patch**

git apply <patch-file>

* + **Explanation:** Applies a patch file to your working directory.

1. **Revert a Commit**

git revert <commit-hash>

* + **Explanation:** Creates a new commit that undoes the changes of the specified commit.

1. **Stage Partial Changes in a File**

git add -p <file>

* + **Explanation:** Lets you select specific changes in a file to stage.

1. **Track an Untracked File**

git add <file>

* + **Explanation:** Adds an untracked file to the staging area.

1. **Remove Ignored Files**

git clean -f -X

* + **Explanation:** Removes files ignored by .gitignore.

1. **Rename a Remote**

git remote rename <old-name> <new-name>

* + **Explanation:** Updates the name of a remote.

1. **Sync a Fork with Upstream**

git fetch upstream

git merge upstream/main

* + **Explanation:** Updates your fork with the original repository.

1. **Force Push to Overwrite Remote History**

git push origin <branch-name> --force

* + **Explanation:** Overwrites the remote branch with local commits.

1. **Create a Lightweight Tag**

git tag <tag-name>

* + **Explanation:** Creates a tag without additional metadata.

1. **Create an Annotated Tag**

git tag -a <tag-name> -m "Tag message"

* + **Explanation:** Includes metadata with the tag.

1. **Push Tags to Remote**

git push origin --tags

* + **Explanation:** Pushes all tags to the remote repository.

1. **List All Tags**

git tag

* + **Explanation:** Displays all tags in the repository.

1. **Remove Local Commits from a Branch**

git reset --hard <commit-hash>

* + **Explanation:** Resets the branch to a specific commit, discarding changes.

1. **List Files Ignored by Git**

git status --ignored

* + **Explanation:** Shows files ignored by .gitignore.

**51. Rewriting Commit History for Multiple Commits**

* **Scenario:** You want to edit commit messages for the last 5 commits.
* git rebase -i HEAD~5
* # Mark commits as "edit" in the editor.
* git commit --amend
* git rebase --continue
  + **Explanation:** Use interactive rebase to modify commit messages, amend each one, and continue rebase until all commits are updated.

**52. Split a Large Commit into Smaller Commits**

* **Scenario:** A single commit contains unrelated changes, and you want to split it into multiple logical commits.
* git reset --soft HEAD~1
* git add <specific-file-or-changes>
* git commit -m "First logical commit"
* git add <other-files-or-changes>
* git commit -m "Second logical commit"
  + **Explanation:** Soft reset the commit, selectively stage changes, and create multiple smaller commits.

**53. Remove Sensitive Information from a Repository**

* **Scenario:** You committed sensitive data (e.g., passwords) and need to completely remove it from history.
* git filter-branch --force --index-filter \
* "git rm --cached --ignore-unmatch <sensitive-file>" \
* --prune-empty --tag-name-filter cat -- --all
* git push origin --force --all
* git push origin --force --tags
  + **Explanation:** Use git filter-branch to rewrite history, remove the sensitive file, and force-push the updated repository.

**54. Recover a Deleted File from a Specific Commit**

* **Scenario:** A file was deleted, and you want to restore it from a past commit.
* git log -- <file>
* git checkout <commit-hash> -- <file>
* git add <file>
* git commit -m "Restore <file> from commit <commit-hash>"
  + **Explanation:** Find the commit where the file existed, restore it using checkout, and commit the restored file.

**55. Rewrite the Author of All Commits**

* **Scenario:** You need to change the author for all commits in the repository.
* git filter-branch --env-filter '
* if [ "$GIT\_AUTHOR\_NAME" = "Old Name" ];
* then
* export GIT\_AUTHOR\_NAME="New Name"
* export GIT\_AUTHOR\_EMAIL="new.email@example.com"
* export GIT\_COMMITTER\_NAME="New Name"
* export GIT\_COMMITTER\_EMAIL="new.email@example.com"
* fi' -- --all
* git push origin --force --all
  + **Explanation:** Rewrites the repository’s history to replace the old author information with the new one.

**56. Resolve Diverged Branches with Rebasing**

* **Scenario:** Your branch has diverged from the remote branch, and you need to reapply your changes on top of the latest remote changes.
* git fetch origin
* git rebase origin/main
* git push origin <branch-name> --force
  + **Explanation:** Rebases your branch onto the updated main branch and force-pushes to reconcile the divergence.

**57. Create and Apply a Custom Git Alias**

* **Scenario:** You frequently type long commands and want to shorten them.
* git config --global alias.lg "log --oneline --graph --decorate --all"
* git lg
  + **Explanation:** Creates an alias for a long command (git log --oneline --graph --decorate --all) and uses it.

**58. Migrate a Subdirectory to a New Repository**

* **Scenario:** You want to extract a folder from a repository and turn it into a standalone repository.
* git subtree split --prefix=<folder-path> -b <new-branch-name>
* git clone <repository-url> <new-repository>
* cd <new-repository>
* git checkout <new-branch-name>
* git push origin main
  + **Explanation:** Use git subtree split to create a new branch with the folder’s history, clone the repo, and push the branch as a new repository.

**59. Resolve "Detached HEAD" State**

* **Scenario:** You checked out a specific commit and are in a detached HEAD state, but now want to create a branch from it.
* git checkout <commit-hash>
* git branch <new-branch-name>
* git checkout <new-branch-name>
  + **Explanation:** Create a branch from the commit and switch to it, returning to a "normal" state.

**60. Revert a Merge Commit**

* **Scenario:** A merge caused issues, and you want to revert it without undoing other changes.
* git log
* git revert -m 1 <merge-commit-hash>
  + **Explanation:** Use -m 1 to specify the parent branch to keep, creating a new commit that undoes the merge.