Scenario: You have made changes to a file and committed it, but later realized that the commit message was incorrect. How would you amend the commit message?

Answer: You can amend the commit message using git commit --amend. This command allows you to modify the most recent commit, including changing the commit message.

Scenario: You accidentally committed a large file that should not be in the repository. How would you remove it from all commits?

* git filter-branch --force --index-filter 'git rm --cached --ignore-unmatch path/to/largefile' --prune-empty --tag-name-filter cat -- --all
* git push origin --force --all
* git push origin --force –tags

Scenario: You are working on a feature branch and need to incorporate the latest changes from the main branch into your feature branch. How would you do this?

Answer: You can incorporate the latest changes using either git merge or git rebase:

* For merging: git checkout feature-branch && git merge main
* For rebasing: git checkout feature-branch && git rebase main

Hands-On: Demonstrate how to create a new branch, make a change, and push the branch to GitHub.

* git checkout -b new-branch
* echo "Some changes" >> file.txt
* git add file.txt
* git commit -m "Added some changes to file.txt"
* git push origin new-branch

Hands-On: Show how to resolve a merge conflict between two branches.

* git checkout feature-branch
* git merge main
* # If there are conflicts, edit the conflicting files to resolve them
* git add resolved-file.txt
* git commit -m "Resolved merge conflict between feature-branch and main"

Hands-On: How do you revert to a previous commit without losing the changes in the current working directory?

Answer: You can use git reset --soft to revert to a previous commit while keeping the changes in your working directory:

git reset --soft <commit-hash>

Scenario: You need to temporarily switch to another branch to fix a critical bug but don't want to lose your current uncommitted changes. How would you proceed?

Answer: You can use git stash to save your uncommitted changes:

* git stash
* git checkout bugfix-branch
* # Fix the bug and commit the changes
* git checkout your-original-branch
* git stash pop

Scenario: Your colleague has force-pushed changes to a shared branch, causing your local branch to diverge. How do you resolve this?

Answer: You can resolve this by resetting your local branch to match the remote branch:

* git fetch origin
* git reset --hard origin/shared-branch

Hands-On: Demonstrate how to squash multiple commits into a single commit.

* git rebase -i HEAD~n # Replace n with the number of commits to squash
* # In the interactive rebase editor, change 'pick' to 'squash' (or 's') for the commits you want to squash
* # Save and close the editor
* # Git will prompt you to edit the commit message

Hands-On: How do you rename a branch both locally and on GitHub?

* git branch -m old-branch-name new-branch-name # Rename locally
* git push origin :old-branch-name # Delete the old branch on GitHub
* git push origin -u new-branch-name # Push the new branch to GitHub and set the upstream

Hands-On: Explain and demonstrate how to cherry-pick a commit from one branch to another.

* git checkout target-branch
* git cherry-pick <commit-hash>

Hands-On: Demonstrate how to use git reflog to recover a commit that was accidentally deleted.

* git reflog # Find the commit hash of the deleted commit
* git checkout <commit-hash> # Recover the commit

Scenario: You cloned a repository but forgot to include the submodules. How do you initialize and update the submodules?

* git submodule init
* git submodule update

Scenario: You are asked to squash all commits in your feature branch into a single commit before merging into the main branch. How would you do this?

* git checkout feature-branch
* git rebase -i main
* # In the interactive rebase editor, mark all commits except the first as 'squash'
* # Save and close the editor, then edit the commit message if necessary

Hands-On: Show how to set up a global .gitignore file and explain its use.

* touch ~/.gitignore\_global
* git config --global core.excludesfile ~/.gitignore\_global
* # Add patterns to ~/.gitignore\_global that you want to ignore in all repositories

Hands-On: Explain and demonstrate how to create and apply a Git stash, then list all stashes.

* git stash save "Stash message"
* # To apply the latest stash
* git stash apply
* # To list all stashes
* git stash list
* # To apply a specific stash from the list
* git stash apply stash@{index}

Hands-On: Show how to check the difference between the working directory and the last commit.

Answer: git diff HEAD

Hands-On: Show how to use git blame to find the author of a specific line in a file.

Answer: git blame file.txt -L <start-line>,<end-line>

Hands-On: Demonstrate how to set up and use Git aliases for common commands.

Answer:

* git config --global alias.co checkout
* git config --global alias.br branch
* git config --global alias.ci commit
* git config --global alias.st status

Hands-On: Explain how to perform a soft reset and a hard reset in Git, and provide examples for both.

Answer:

Soft reset (keeps changes in the working directory): git reset --soft <commit-hash>

Hard reset (discards all changes and resets the working directory): git reset --hard <commit-hash>