Mastering Git and GitHub: A Comprehensive Guide Part 4

1. What is the purpose of using `git rebase` and

when should it be employed?

Answer:

`git rebase` is used to integrate changes from one branch into another by moving or combining a sequence of commits. It is often used to maintain a linear project history. This can be beneficial in situations where a feature branch needs to be updated with changes from the main branch. The command `git rebase <base>` is typically used for this purpose, where `<base>` is the branch you want to rebase onto.

2. How do you fork a repository on GitHub, and what

advantages does forking provide?

Answer:

Forking a repository on GitHub involves creating a personal copy of someone else's project. This can be done by clicking the "Fork" button on the GitHub repository page. Forking allows you to freely experiment with changes without affecting the original project. Once you've made your modifications, you can submit a pull request to propose changes back to the original repository.

3. What steps should be taken to resolve merge conflicts in Git?

Answer:

Merge conflicts occur when Git is unable to automatically reconcile changes in different branches. To resolve conflicts, follow these steps:

Use `git pull` to fetch the changes from the remote repository.
Open the files with conflicts and manually resolve the differences.
Use `git add` to stage the resolved files.
Complete the merge with `git merge`.
Finally, commit the changes with `git commit`.

4. How can GitHub organizations enhance collaborative development?

Answer:

GitHub organizations provide a way to manage and structure collaborative development. Features include centralized access control, team management, and the ability to create

repositories under the organization. This facilitates better organization, collaboration, and access control within a group of developers working on a project.

5. What is a GitHub webhook, and how can it be used in development workflows?

Answer:

GitHub webhooks are automated notifications that inform external services about repository events. They can be used to trigger actions such as CI/CD pipelines, automated testing, or deployment scripts. Setting up a webhook involves configuring a payload URL and specifying the events that should trigger the webhook.

6. How does the `git stash` command work, and when is it useful?

Answer:

The `git stash` command is used to save changes that are not ready to be committed but need to be set aside temporarily. When a stash is created using `git stash save "message"`, the changes are saved, and the working directory is reverted to the last commit. The stash can later be reapplied using `git stash apply` or removed using `git stash drop`. The `git stash list` command displays all stashes.

7. How can you inspect the contents of a specific stash in Git?

Answer:

The command `git stash show stash@{0}` can be used to inspect the contents of a specific stash. It shows the changes introduced in the stash, providing a summary of modified files and the lines changed within each file.

8. What is the difference between

`git stash apply` and `git stash pop`?

Answer:

Both commands are used to apply stashed changes, but `git stash apply` leaves the stash in the stash list, while `git stash pop` not only applies the stash but also removes it from the stash list.

9. How can you undo a specific stash in Git?

Answer:

To apply and drop a specific stash in one command, you can use `git stash pop stash@{0}`. If you want to keep the stash after applying, you can use `git stash apply stash@{0}`. To simply remove a stash without applying, use `git stash drop stash@{0}`.

10. What is the purpose of `git restore` and how does it differ from `git reset`?

Answer:

`git restore` is used to restore working tree files from either the index or a specific commit. To unstage changes, you can use `git restore --staged <file>`. In contrast, `git clean -i` is used to interactively remove untracked files. `git reset` is more powerful and can be used to reset the staging area and working directory.

11. How can you selectively unstage changes before committing?

Answer:

To unstage specific changes before committing, you can use `git restore --staged <file>`. This command unstages the changes in the specified file, leaving the working directory unchanged.

12. Explain the purpose of `git cherry-pick` and

when it is commonly used.

Answer:

`git cherry-pick` is used to apply a specific commit from one branch to another. This is useful when you want to include a specific change without merging the entire branch. The command is `git cherry-pick <commit-id>`.

13. What is the difference between `git revert` and `git reset`?

Answer:

`git revert` is used to create a new commit that undoes the changes made by a previous commit, maintaining a clear history. On the other hand, `git reset` is more powerful and can be used to reset the branch to a specific commit, either preserving or discarding changes.

14. How can GitHub issues enhance collaboration in a development project?

Answer:

GitHub issues provide a platform for tracking and discussing tasks, enhancements, bugs, and other aspects of a project. Developers can create, assign, and comment on issues, providing a centralized way to manage and prioritize work.

15. How can you close a GitHub issue using a commit message?

Answer:

To close a GitHub issue using a commit message, include the phrase "closes #<issue-number>" or "close #<issue-number>" in the commit message. For example, `git commit -m "Fixing a bug closes #123"` will automatically close issue number 123 upon merging the commit into the main branch.

Check_Out_Detailed_Blog:-https://medium.com/@srivastavayushmaan1347/mastering-git-and-github-a-comprehensive-guide-part-4-193e3280c094