GIT

CHEAT SHEET

Git Basics

1. **git init**
   Initializes a new Git repository in your project directory.
2. **git clone**
   Creates a copy of an existing repository from a remote server.
3. **git status**
   Displays the state of the working directory and staging area.
4. **git add**
   Adds changes in the working directory to the staging area.
5. **git commit**
   Records the changes in the staging area as a new commit in the repository.
6. **git branch**
   Lists, creates, or deletes branches in the repository.
7. **git checkout**
   Switches between branches or restores files from a specific commit.
8. **git switch**
   An alternative to checkout for switching branches.
9. **git merge**
   Combines changes from one branch into the current branch.
10. **git rebase**
    Re-applies commits on top of another base commit, streamlining the commit history.

Remote Repository Commands

11. **git remote**
    Manages the connections to remote repositories.
12. **git fetch**
    Downloads changes from a remote repository without merging them into your branch.
13. **git pull**
    Fetches changes from a remote repository and merges them into your current branch.
14. **git push**
    Uploads your local branch's commits to a remote repository.
15. **git remote add**
    Adds a new remote repository connection to your project.
16. **git remote remove**
    Removes a connection to a remote repository.
17. **git remote rename**
    Renames an existing remote connection.
18. **git remote show**
    Displays detailed information about a remote repository.
19. **git pull --rebase**
    Fetches changes and rebases local commits on top of the latest changes from the remote repository.
20. **git push --force**
    Forcibly updates the remote branch, overwriting changes.

Inspecting and Comparing

21. **git log**
    Shows the commit history of the repository.
22. **git diff**
    Displays the differences between various states of the repository.
23. **git show**
    Shows the details of a specific commit.
24. **git blame**
    Displays line-by-line history for a file to identify changes and contributors.
25. **git shortlog**
    Summarizes the commit history by author.
26. **git reflog**
    Records changes to the repository's reference logs.
27. **git tag**
    Lists or creates tags to mark specific points in the commit history.
28. **git log --graph**
    Displays a graphical representation of the commit history.
29. **git diff --staged**
    Shows the changes staged for the next commit.
30. **git show-branch**
    Displays the branches and their commit history.

Stashing and Cleaning

31. **git stash**
    Temporarily stores uncommitted changes to work on a clean branch.

32. **git stash pop**
    Applies and removes the most recent stash.

33. **git stash apply**
    Applies the most recent stash without removing it.

34. **git stash list**
    Lists all the stashes in the repository.

35. **git stash drop**
    Deletes a specific stash.

36. **git clean**
    Removes untracked files from the working directory.

Working with Submodules

37. **git submodule add**
    Adds another repository as a submodule in your project.

38. **git submodule update**
    Updates submodules to match the latest commit in their respective repositories.

39. **git submodule init**
    Initializes the submodule configuration in a cloned repository.

40. **git submodule deinit**
    Uninitializes a submodule to detach it from your main repository.

Resetting and Rewriting History

41. **git reset**
    Resets the current branch to a specific state without removing changes.

42. **git reset --soft**
    Moves the HEAD pointer but leaves changes staged.

43. **git reset --hard**
    Resets the working directory, staging area, and HEAD to a specified state.

44. **git revert**
    Creates a new commit that undoes changes from a previous commit.

45. **git cherry-pick**
    Applies specific commits from one branch onto another branch.

Miscellaneous

46. **git archive**
    Creates a tarball or zip file of the repository or a specific branch.

47. **git bisect**
    Uses binary search to find the commit that introduced a bug.

48. **git gc**
    Cleans up unnecessary files and optimizes the repository.

49. **git config**
    Configures user information and settings for the repository.

50. **git help**
    Displays documentation for Git commands.

From SAURABH NAMDEO

Happy Learning