Education

Git Cheat Sheet

Welcome to GitHub Education's Git cheat sheet! This handy guide is perfect for newcomers and experienced developers alike. It includes the most commonly used Git commands to make your coding journey smoother and more efficient. Ready to get started? Let's Git to it!

1. Git Confirmation

Setting up Git for your local repositories.

```
git config --global user.name "<name>" // Sets your name for all local repositories
git config --global user.email "<email>" // Sets your email for all local repositories
git config --list // Lists all settings
```

2. Git Repository Initialization

Git Repository Initialization

```
git init // Initializes a new Git repository
git clone <repo> // Downloads a project and its entire version history
```

3. Changes and Staging

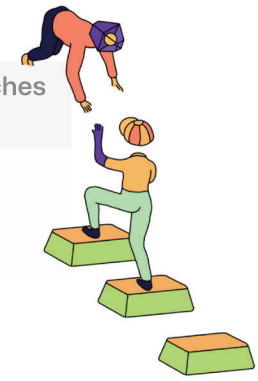
Propose changes to your project and prepare them for commit.

```
git status // Lists all new or modified files to be committed
git add <file> // Adds a snapshot of the file to staging
git add . // Adds all current changes to the next commit
git add -p // Adds changes interactively by hunk from files that have previously been checked in
git reset <file> // Unstages the file, but preserves its contents
git commit -m "<message>" // Records changes to the repository with a descriptive message
git diff // Shows file differences not yet staged
git diff —staged // Shows file differences between staging and the last file version
```

4. Branching and Merging

Develop features isolated from each other with branches and combine lines of development.

```
git branch // Lists all local branches in the current repository.
git branch <branch> // Creates a new branch
git switch <branch> // Switches to the specified branch
git switch —c <branch> // Creates a new branch and switches to it
```



5. Rebasing and Managing Commit History

Move or combine a sequence of commits to a new base commit. Manipulate your commit history, change commit messages, and alter commits and trees.

```
git rebase <branch> // Moves or combines a sequence of commits to a new base commit
git rebase --onto <branch> // Rebase all commits on the current branch onto another branch
git commit --amend // Change the last commit (Don't amend published commits!)
git rebase -i [commit] // Interactively rebase your current HEAD onto [commit]
git rebase -i --root // Interactively rebase your entire history
```

6. Undoing Changes

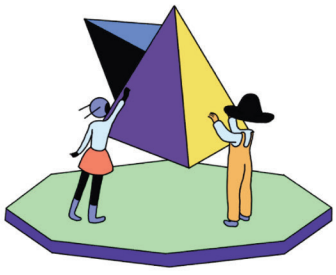
Unstage a file or discard changes in the working directory.

```
git restore <file> // Discards changes in the working directory
git restore --staged <file> // Unstage changes
```

7. Inspecting and Comparing Changes

Inspect changes or compare two different commits.

```
git log // Shows commit history
git log branchB..branchA // Show the commits on branchA that are not on branchB
git log —follow [file] // Show the commits that changed file, even across renames
git diff branchB...branchA // Show the difference of what is in branchA that is not in branchB
git show [SHA] // Show any object in Git in human-readable format
```

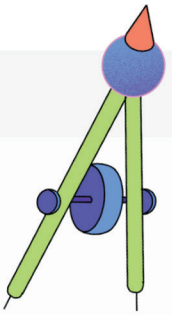


8. Working with Remote Repositories

Connect your repository with others and synchronize data.

git remote // Shows all remote repositories
git remote -v // Shows all remote repositories verbose]
git remote add [alias] [url] // Adds a git URL as an alias
git fetch [alias] // Fetch down all the branches from that Git remote
git merge [alias]/[branch] // Merge a remote branch into your current branch to bring it up to date
git push [alias] [branch] // Transmit local branch commits to the remote repository branch
git pull // Fetch and merge any commits from the tracking remote branch
git push origin --delete <branch_name> // Delete a remote branch
git checkout -b [branch] [remotename]/[branch] // Check out a remote branch

9. Stashing and Cleaning



Save changes that you want to put away for later or clean your working directory.

git stash // Save modified and staged changes
git stash list // List stack-order of stashed file changes
git stash pop // Write working from top of stash stack
git stash drop // Discard the changes from top of stash stack
git clean -n // Show what would be removed
git clean -f // Force remove untracked files
git clean -fd // Force remove untracked directories

10. Tracking and Ignoring Files

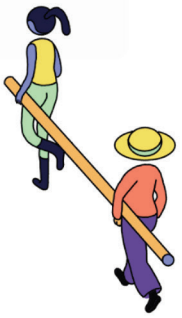
Control which files Git can track and ignore.

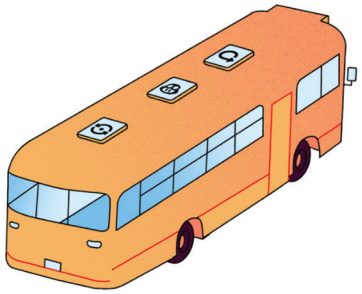
git ls-files --other —ignored --exclude-standard // List all ignored files in this project
touch .gitignore // Create a .gitignore file
.gitignore // Open the .gitignore file and add rules

11. Finding and Fixing Bugs

Identify the commit that introduced a bug by using a binary search.

git bisect start // Start bisecting
git bisect good [commit] // Mark [commit] as known-good
git bisect bad [commit] // Mark [commit] as known-bad
git bisect reset // End bisect session





12. Cherry Picking

Apply changes introduced by some existing commits.

git cherry-pick [commit] // Apply changes introduced by an existing commit
git cherry-pick —abort // Cancel the operation
git cherry-pick —continue // Continue the operation

13. Viewing Authorship

View who changed what and when in [file].

git blame [file] // Show what revision and author last modified each line of a file

14. Moving Files

Move files from one directory to another.

git mv [file-original-path] [file-new-path] // Move file from the original path to the new path

Explore GitHub Desktop and GitHub Mobile Apps!

Enhance your Git experience with GitHub Desktop and GitHub Mobile apps. Whether you're working on your local machine or on the go, these apps provide seamless integration with your GitHub repositories. Download now to enjoy a user-friendly interface, real-time collaboration, and more!

GitHub Desktop: https://gh.io/desktop

GitHub Mobile: https://gh.io/mobile

Remember, this is just the beginning of your Git journey.
With time and practice, these commands will become second nature. Now, go forth and code!

github.com/education
Ready for more coding fun? Discover tutorials, free resources, and connect with our community on our website. See you there!