Version Control Cheatsheet

A comprehensive cheat sheet covering essential version control concepts, commands, and workflows with Git.

Git Basics

Core Concepts

Repository: A directory containing project files and a `.git` subdirectory that stores the repository's metadata, object database, and configuration.

Working Directory: The directory on your file system containing the actual files you are working on. This is where you make changes.

Staging Area (Index): A file that stores information about the changes you want to include in your next commit. Use `git add` to stage changes.

Commit: A snapshot of your repository at a specific point in time. Commits have a unique ID (SHA-1 hash).

Branch: A movable pointer to a commit. Branches allow you to work on different features or fixes without affecting the main codebase.

Remote: A repository hosted on another computer or server. Used for collaboration and backups.

Basic Commands

`git init`	Initializes a new Git repository in the current directory.
`git clone <repository_url>`	Clones a remote repository to your local machine.
`git status`	Displays the status of the working directory and staging area.
`git add <file>`	Adds a file to the staging area.
`git commit -m "<message>"`	Commits the staged changes with a descriptive message.
`git log`	Shows the commit history of the repository.

Branching and Merging

Branch Management

`git branch`	Lists all local branches. The current branch is marked with an asterisk (*).
`git branch <branch_name>`	Creates a new branch with the specified name.
`git checkout <branch_name>`	Switches to the specified branch.
`git checkout -b <branch_name>`	Creates a new branch and switches to it.
`git branch -d <branch_name>`	Deletes the specified branch (if it has been merged).
`git branch -D <branch_name>`	Force deletes the specified branch (even if it hasn't been merged).

Merging Branches

`git merge <branch_name>`	Merges the specified branch into the current branch.
`git mergetool`	Opens a merge tool to resolve conflicts manually.
`git commit`	After resolving conflicts, commit the merge.
`git merge --abort`	Aborts the merge process and returns to the state before the merge.

Rebasing

Rebasing is an alternative to merging that integrates changes from one branch into another by moving or combining a sequence of commits to a new base commit.

`git rebase <branch_name>` - Rebase the current branch onto `<branch_name>`.

After resolving conflicts, use `git rebase --continue` to proceed.

Use `git rebase --abort` to stop rebasing process.

Remote Repositories

Working with Remotes

`git remote add <name> <url>`	Adds a remote repository with a specified name and URL.
`git remote -v`	Lists all remote repositories with their URLs.
`git remote remove <name>`	Removes the remote repository with the specified name.
`git fetch <remote>`	Fetches the latest changes from the remote repository without merging them.
`git pull <remote> <branch>`	Fetches and merges changes from a remote branch into the current branch.
`git push <remote> <branch>`	Pushes local commits to the remote branch.

Collaboration Workflow

1. **Clone** the repository: `git clone <repository_url>`
2. **Create a branch** for your changes: `git checkout -b <feature_branch>`
3. **Make changes** and commit them locally: `git add .`, `git commit -m "<descriptive_message>"`
4. **Push your branch** to the remote repository: `git push origin <feature_branch>`
5. **Create a pull request** on the remote repository.
6. After review and approval, your changes will be merged.

Advanced Git

Stashing

`git stash`	Temporarily saves changes that you don't want to commit immediately.
`git stash list`	Lists all stashed changesets.
`git stash apply`	Applies the most recent stashed changes.
`git stash apply stash@{2}`	Applies a specific stashed changeset (e.g., the third one).
`git stash drop`	Removes the most recent stashed changes.
`git stash pop`	Applies and removes the most recent stashed changes.

Rewriting History

Warning: Rewriting history can cause issues if you're collaborating with others. Use with caution.
`git commit --amend` - Modify the last commit (e.g., to fix the commit message or add staged changes).
`git rebase -i <commit>` - Interactive rebase, allows you to edit, reorder, or squash commits.

Ignoring Files

`.gitignore`	A file that specifies intentionally untracked files that Git should ignore.
Example	`*.log` - Ignores all files with the `.log` extension. `/temp/` - Ignores the `temp` directory at the root of the repository.