GIT PRESENTATION

PRESENTED BY PRASAD YADAV

Git Cheatsheet

This cheatsheet covers essential Git commands, their examples, and usage explanations, along with a diagram illustrating common Git workflows.

Table of Contents

Configuration

Set up your Git environment.

Command	Example	Usage
`git config --global user.name "<name>"`	`git config --global user.name "John Doe"`	Sets the name for commit messages.
`git config --global user.email "<email>"`	`git config --global user.email "john@example.com"`	Sets the email for commit messages.
`git config --list`	`git config --list`	Displays all Git configurations.

Repository Management

Create or clone repositories.

Command	Example	Usage
`git init`	`git init`	Initializes a new Git repository in the current directory.
`git clone <url>`	`git clone https://github.com/user/repo.git`	Clones a remote repository to your local machine.

Working with Files

Manage files in the working directory and staging area.

Command	Example	Usage
`git add <file>`	`git add index.html`	Stages a specific file for the next commit.
`git add .`	`git add .`	Stages all modified and new files in the current directory.
`git status`	`git status`	Shows the status of the working directory and staging area.
`git rm <file>`	`git rm oldfile.txt`	Removes a file from the working directory and stages the deletion.

Committing Changes

Save changes to the repository.

Command	Example	Usage
`git commit -m "<message>"`	`git commit -m "Add login feature"`	Commits staged changes with a descriptive message.
`git commit -a -m "<message>"`	`git commit -a -m "Update styles"`	Stages and commits all modified tracked files in one step.
`git commit --amend`	`git commit --amend`	Modifies the most recent commit (e.g., to change the message or add files).

Branching

Manage multiple lines of development.

Command	Example	Usage
`git branch`	`git branch`	Lists all branches in the repository.
`git branch <branch>`	`git branch feature-x`	Creates a new branch named `feature-x`.
`git checkout <branch>`	`git checkout feature-x`	Switches to the specified branch.
`git checkout -b <branch>`	`git checkout -b feature-x`	Creates and switches to a new branch in one step.
`git branch -d <branch>`	`git branch -d feature-x`	Deletes the specified branch (if merged).

Merging and Rebasing

Integrate changes from different branches.

Command	Example	Usage
`git merge <branch>`	`git merge feature-x`	Merges `feature-x` into the current branch.
`git rebase <branch>`	`git rebase main`	Reapplies commits from the current branch onto `main`.
`git rebase -i <commit>`	`git rebase -i HEAD~3`	Interactively rebases the last 3 commits (e.g., to squash or edit).

Remote Repositories

Work with remote repositories (e.g., GitHub).

Command	Example	Usage
`git remote add <name> <url>`	`git remote add origin https://github.com/user/repo.git`	Adds a remote repository named `origin`.
`git push <remote> <branch>`	`git push origin main`	Pushes the `main` branch to the remote repository.
`git pull <remote> <branch>`	`git pull origin main`	Fetches and merges changes from the remote branch.
`git fetch <remote>`	`git fetch origin`	Downloads objects and refs from the remote without merging.

Inspecting History

View commit history and changes.

Command	Example	Usage
`git log`	`git log`	Shows the commit history.
`git log --oneline`	`git log --oneline`	Displays a compact, one-line commit history.
`git diff`	`git diff`	Shows changes between the working directory and staged files.
`git diff <commit> <commit>`	`git diff abc123 def456`	Shows differences between two commits.

Undoing Changes

Revert or reset changes.

Command	Example	Usage
`git reset <file>`	`git reset index.html`	Unstages a file but preserves its changes.
`git reset --hard <commit>`	`git reset --hard abc123`	Resets the working directory and index to a specific commit.
`git revert <commit>`	`git revert abc123`	Creates a new commit that undoes the specified commit.
`git clean -f`	`git clean -f`	Removes untracked files from the working directory.

Git Workflow Diagram

Below is a textual representation of a typical Git workflow, showing the interaction between the working directory, staging area, local repository, and remote repository.

```plain
[Remote Repository]
    ↑↓ (push/pull)
[Local Repository]
    ↑↓ (commit)
[Staging Area]
    ↑↓ (add/reset)
[Working Directory]
```

Git Cherry-Pick: Explanation and Usage

git cherry-pick is a Git command that allows you to apply a specific commit from one branch to another. It "picks" a single commit (or multiple commits) and applies its changes to the current branch, creating a new commit with the same changes but a different commit hash. This is useful when you want to incorporate specific changes from another branch without merging the entire branch.

Diagram of Cherry-Pick Workflow

```text
[feature]                    [main]
  |                            |
  O---A---B                    O
      |                        |
      |  git cherry-pick B     |
      +------------------->    O---B'
```

- `O`: Initial commit.
- `A`: Feature update commit.
- `B`: Bug fix commit on `feature`.
- `B'`: New commit on `main` with the same changes as `B`.

Thank you very much!

PRESENTED BY PRASAD YADAV