

Fundamentals of Git and GitHub

Git is a version control system for tracking code changes. GitHub is a platform for hosting and collaborating on Git repositories. Together, they enable efficient software development and collaboration.



**by David
Ouma**

Getting Started with Git

1

Install Git

Download and install Git on your system.

2

Configure Git

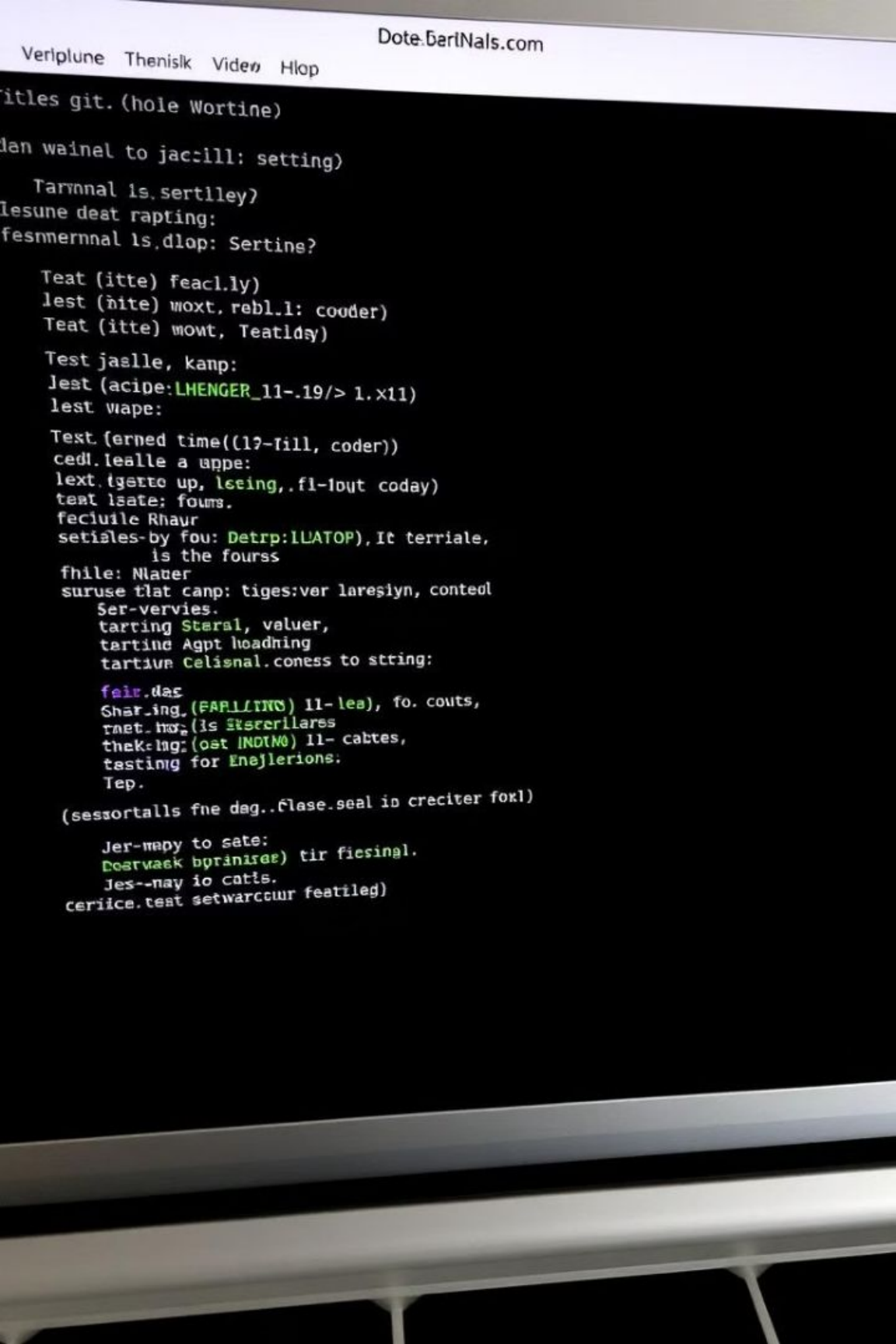
Set up your username and email using git config commands.

3

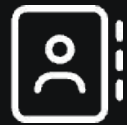
Initialize Repository

Create a new Git repository with git init.





Basic Git Commands



git add

Add changes to the staging area.



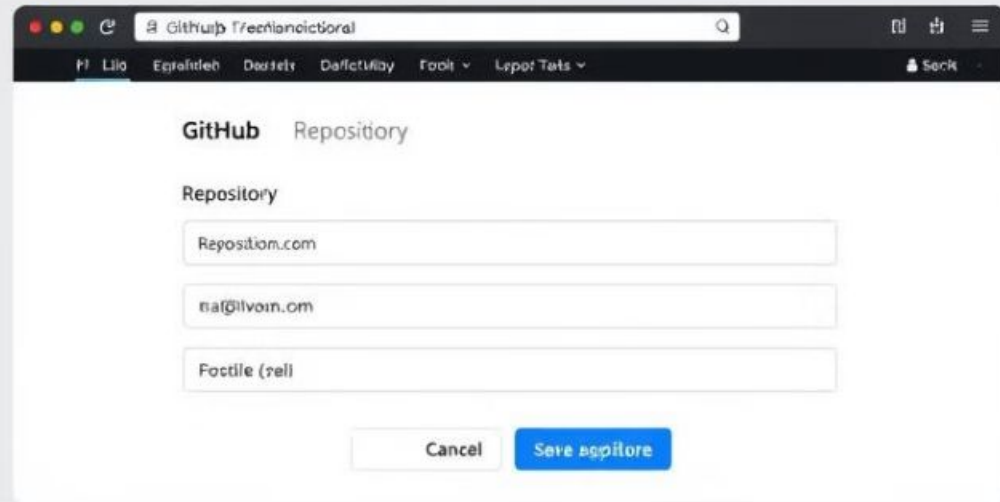
git commit

Save changes in the repository with a message.



git status

Check which files are staged, unstaged, and untracked.



GitHub Essentials

1

Create Repository

Set up a new repository on GitHub.

2

Push Local Repository

Use git remote add and git push to upload local code.

3

Collaborate

Invite others to contribute to your project.

Creation

- The move are different the is take care poly am Gileadice solve here repository of all these.
- A repository is a place to store your code, and it's the best way to keep your code safe and secure.
- To create a repository, you need to have a GitHub account.
- Once you have a GitHub account, you can create a repository by clicking on the '+' icon in the top right corner of the page.
- You can also create a repository by using the command line.
- Once you have created a repository, you can push your code to it using the command line.
- You can also push your code to a repository using a GUI tool like GitHub Desktop.



Branching in Git

1

Create Branch

Use `git branch` to create a new branch.

2

Switch Branch

Use `git checkout` to move between branches.

3

Merge Branches

Combine changes from one branch into another with `git merge`.

Collaborative Features on GitHub

Forking

Create a copy of someone else's repository on your GitHub account.

Pull Requests

Submit changes for review and integration into the main project.

Issues

Track bugs, enhancements, and tasks within a project.

Advanced Git Topics

Rebasing

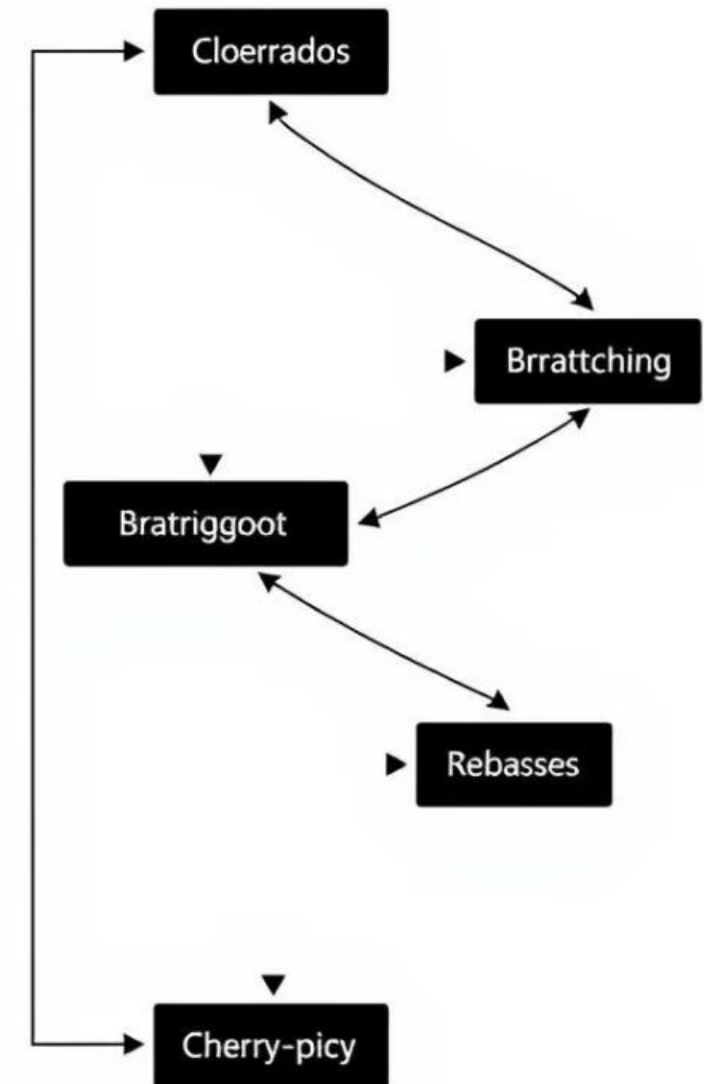
Clean up commit history and integrate changes from one branch to another.

Stashing

Temporarily save changes when switching branches.

Interactive Rebase

Edit, reorder, or squash commits using `git rebase -i`.





Advanced GitHub Actions

GitHub Actions

Automate workflows like testing and deployment

GitHub Pages

Host static websites directly from a repository

Branch Protection

Set rules to restrict direct pushes to critical branches

Advanced Branching Strategies

1

Feature Branch Workflow

Develop each feature in its own branch.

2

Git Flow

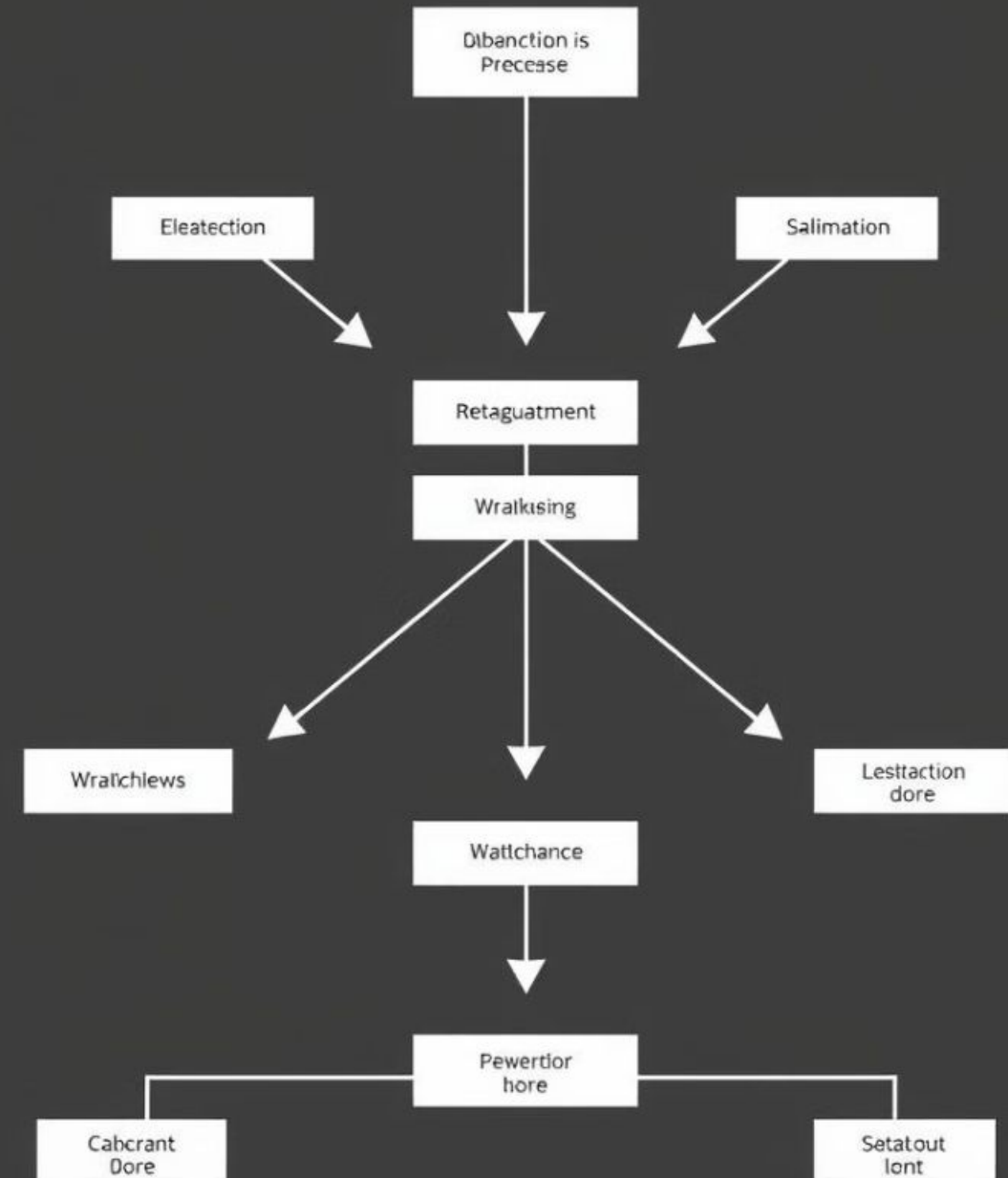
Use multiple branches for different stages of development.

3

Trunk-Based Development

Work on a single main branch with short-lived feature branches.

Farechning

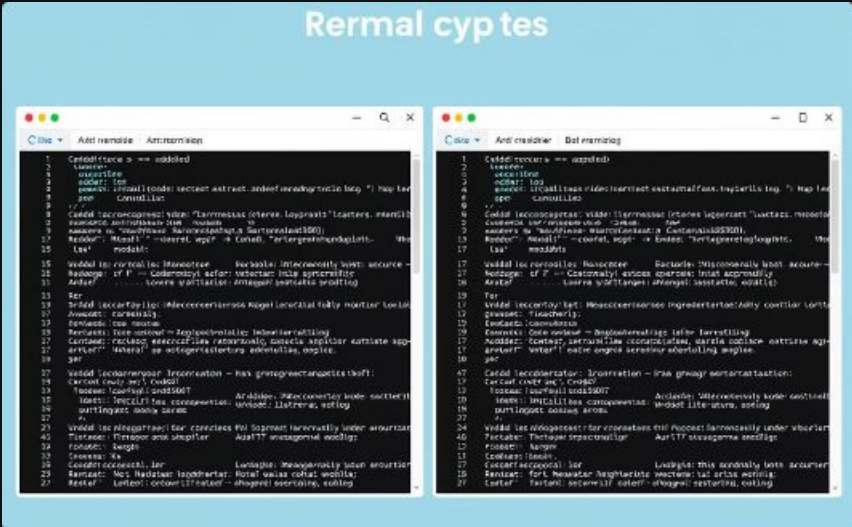


Resolving Conflicts



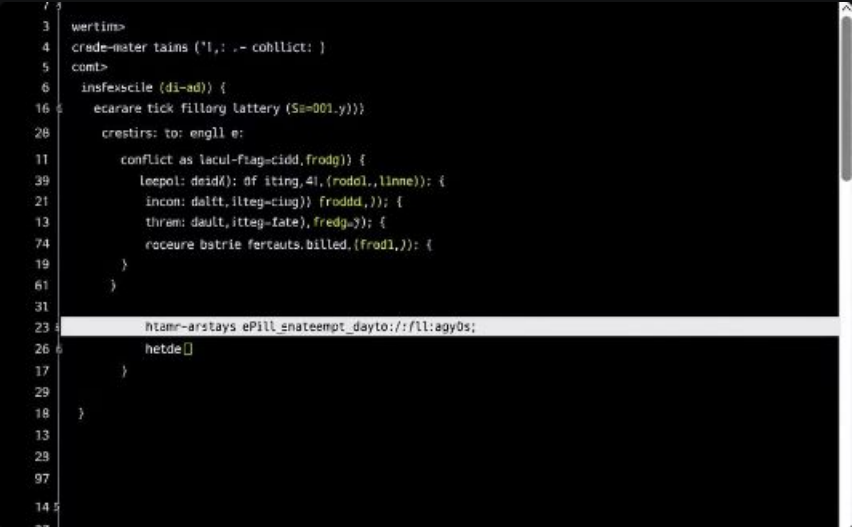
Merge Conflicts

Resolve conflicts when merging branches with conflicting changes.



Merge Tools

Use visual tools like kdiff3 or meld to resolve conflicts.



Manual Resolution

Manually edit files to resolve conflicts and remove conflict markers.