@mayank ahuja

Some Best Practices

Commit frequently and logically

Make small, focused commits that represent a logical unit of work.

For example, instead of committing multiple unrelated changes, commit each change separately.

Write clear and descriptive commit messages

Use meaningful commit messages that describe the purpose of the commit.

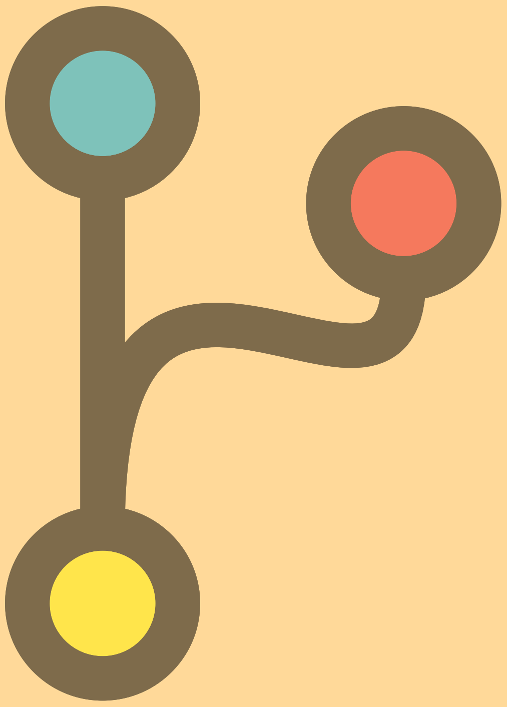
For instance, "Add user authentication feature" is better than "Update code."

@mayank ahuja

Use branches for features and bug fixes

Create separate branches for each feature or bug fix.

For instance, feature/user-authentication or bugfix/issue-123



Regularly update your local repository

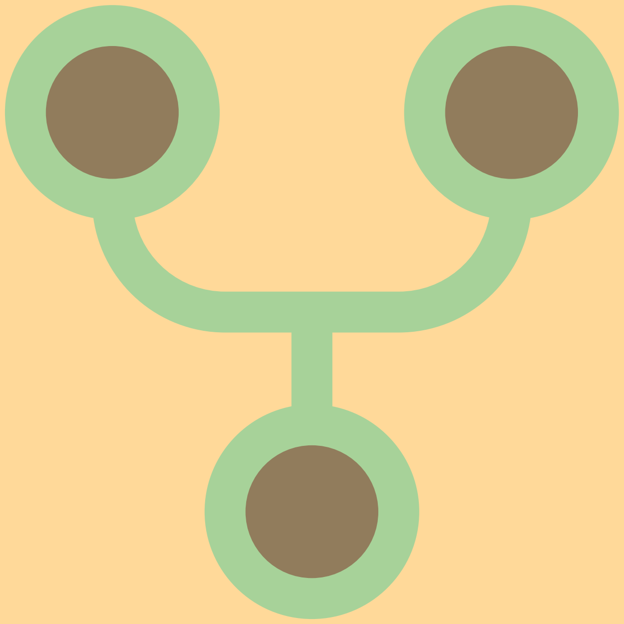
Fetch and pull changes from the remote repository to keep your local copy up to date.

For example, git pull origin master

Rebase instead of merge for cleaner history

Use rebase to integrate changes from one branch into another, preserving a cleaner commit history.

For instance, git rebase main



@mayank ahuja

Use .gitignore to exclude unnecessary files

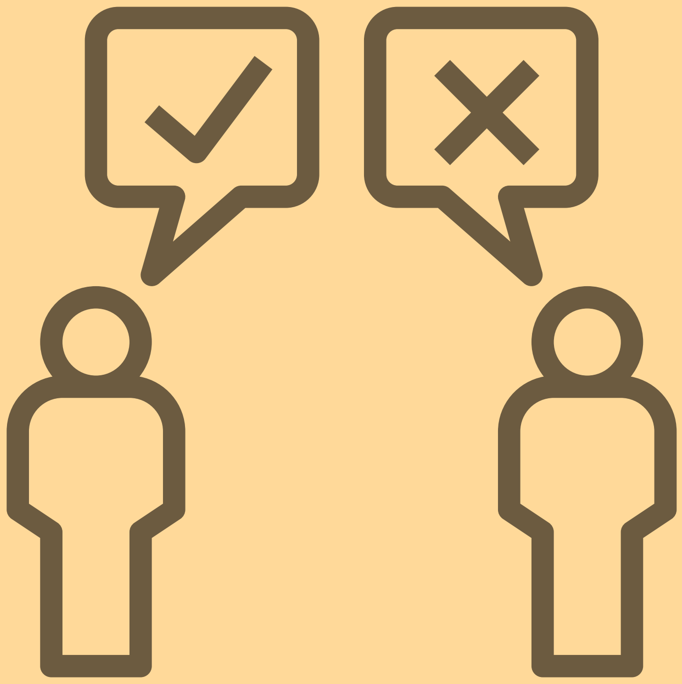
Specify files and directories to ignore in the .gitignore file, such as temporary files, build artifacts, or sensitive information.

@mayank ahuja

Review changes before committing

Use 'git diff' to review changes before committing them.

This allows you to ensure that only the intended changes are included in your commits.



Use interactive staging

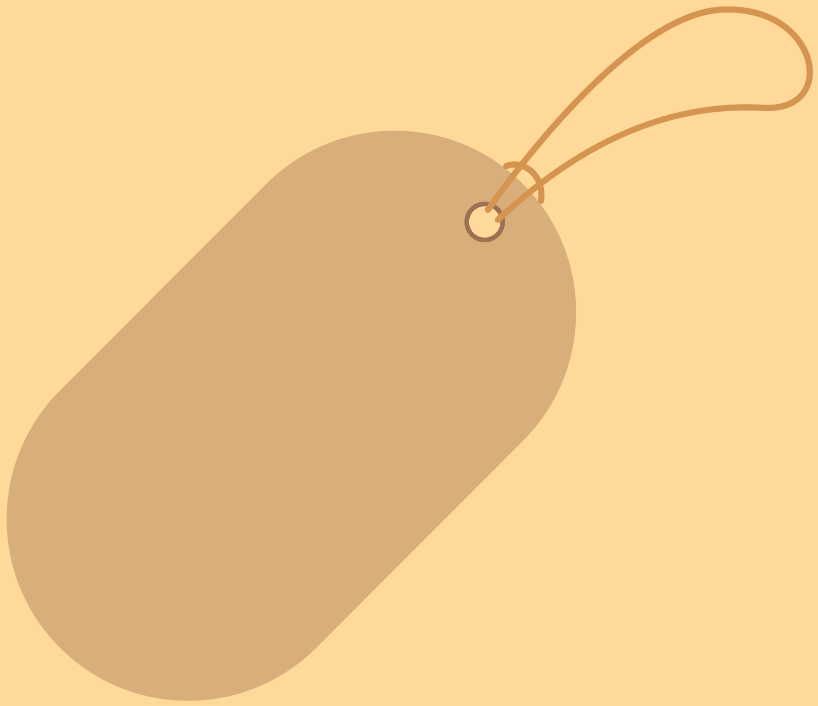
Use 'git add -p' to interactively stage changes and selectively include specific changes within a file.

@mayank ahuja

Use tags for versioning

Use tags to mark important points in your project's history, such as releases or significant milestones.

For example, git tag v1.0.0



Use git stash for temporary work

Use 'git stash' to save your changes temporarily when you need to switch to a different branch or work on something else.

For example, git stash save "Work in progress"

@mayank ahuja

Use git cherry-pick for specific commits

Selectively apply specific commits from one branch to another using 'git cherry-pick'.

This can be useful when you want to apply specific changes without merging entire branches.