ONE PAGE SUMMARY

IT & Software › Git & GitHub › Conflict Resolution

Introduction to Conflict Resolution

Mastering Conflict Resolution in Git and GitHub

DEFINITION

> Conflict resolution in Git occurs when changes from different sources clash. It involves identifying and resolving these discrepancies to maintain a smooth workflow.

KEY CONCEPTS

Merge Conflicts

Merge conflicts happen when two branches have competing changes that Git cannot automatically reconcile.

Conflict Markers

Conflict markers are special symbols added by Git in files to indicate where conflicts occur.

Manual Resolution

Manual resolution requires users to edit conflicting files directly and choose which changes to keep.

Staging Changes

After resolving conflicts, changes must be staged before committing them to the repository.

Rebasing

Rebasing is a method to integrate changes from one branch into another, often used to avoid conflicts.

EXAMPLES

→ Two developers edit the same line in a file, causing a merge conflict.

→ A developer resolves a conflict by choosing one version of the code over another.

→ Using 'git status' to identify files with merge conflicts.

MEMORY TIPS

★ Remember 'C' for Conflict and 'R' for Resolution.

★ Use the phrase 'Merge, Edit, Stage, Commit' to recall the conflict resolution steps.

★ Think of 'Conflict Markers' as road signs indicating where to stop and resolve.

COMMON MISTAKES

✗ Ignoring conflict markers and committing without resolving conflicts.

✗ Failing to test the code after resolving conflicts.

✗ Not communicating with team members about changes made during conflict resolution.

QUICK RECAP

Conflict resolution in Git is essential for collaborative coding. It involves identifying, editing, and staging changes to resolve discrepancies between branches. Always communicate with your team and test your code after resolving conflicts.