GIT

"Unlocking Git's Power: A Deep Dive into Essential Commands and Best Practices"

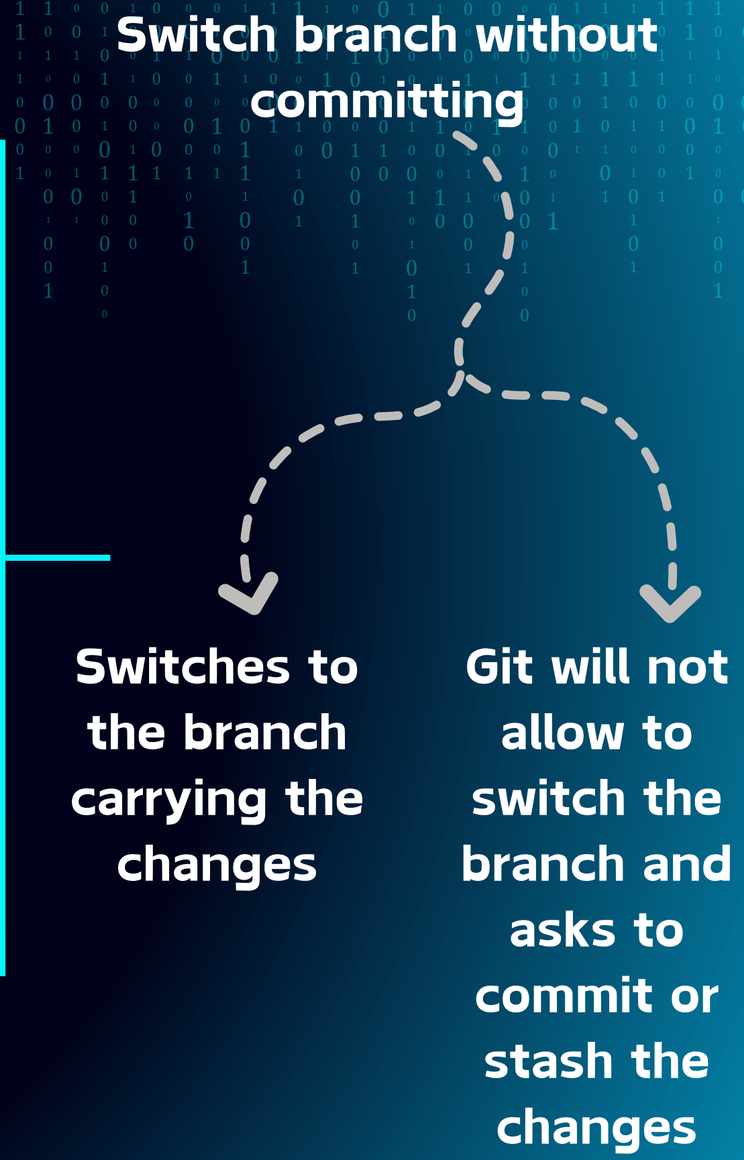
GITHUB REPOSITORY LINK: https://github.com/Ananya-m0140/Git_KOSS

CONTENTS

1. GIT STASH
2. GIT BISECT
3. GIT REFLOG
4. GIT DIFF
5. GIT SWITCH
6. GIT REBASE
7. GIT CHERRY-PICK

1 GIT STASH

- Git stash is a command used to temporarily store changes in your working directory without committing them to the repository.
- Features:
  - Provides a way to save work-in-progress changes without creating a commit.
  - Allows you to switch branches or perform other operations without committing incomplete changes.
  - Supports multiple stashes, enabling you to store different sets of changes separately.



GIT STASH

git stash
Save changes in the working directory to stash

git stash apply
Apply stashed changes to the working directory

git stash pop
Remove the topmost stashed changes from the stash list

git stash list
List all stashes with descriptions

git stash apply stash@{<stash-index>}
Apply a specific stash from the stash list

git stash drop stash@{<stash-index>}
Drop a specific stash from the stash list

2 GIT BISECT

Git bisect is used for binary searching through commits to find a specific bug or issue.

AUTOMATICALLY IDENTIFIES A FAULTY COMMIT BY PERFORMING A BINARY SEARCH.

HELPS IN ISOLATING THE COMMIT WHERE AN ISSUE WAS INTRODUCED.

git bisect start

Start the bisect session

1

git bisect good <commit>

Mark a known good commit

2

git bisect bad

Mark the current commit as bad

3

GIT REFLOG

Git reflog (reference log) is a tool for viewing the history of Git references (such as commits, branches, and tags) in a repository.

USES OF GIT REFLOG

git reflog

View the reflog history

git reset --hard HEAD@{<reflog-index>}

Restore to a previous state using reflog

git reflog <branch-name>

Show reflog history for a specific branch

4 GIT DIFF

- Displays changes made to files, lines added, modified, or deleted.
- Useful for reviewing code changes before committing or merging.

git diff branch1..branch2

Compare changes between two branches

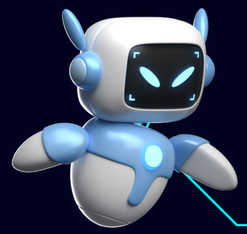
git diff -- myfile.txt

Show changes for a specific file

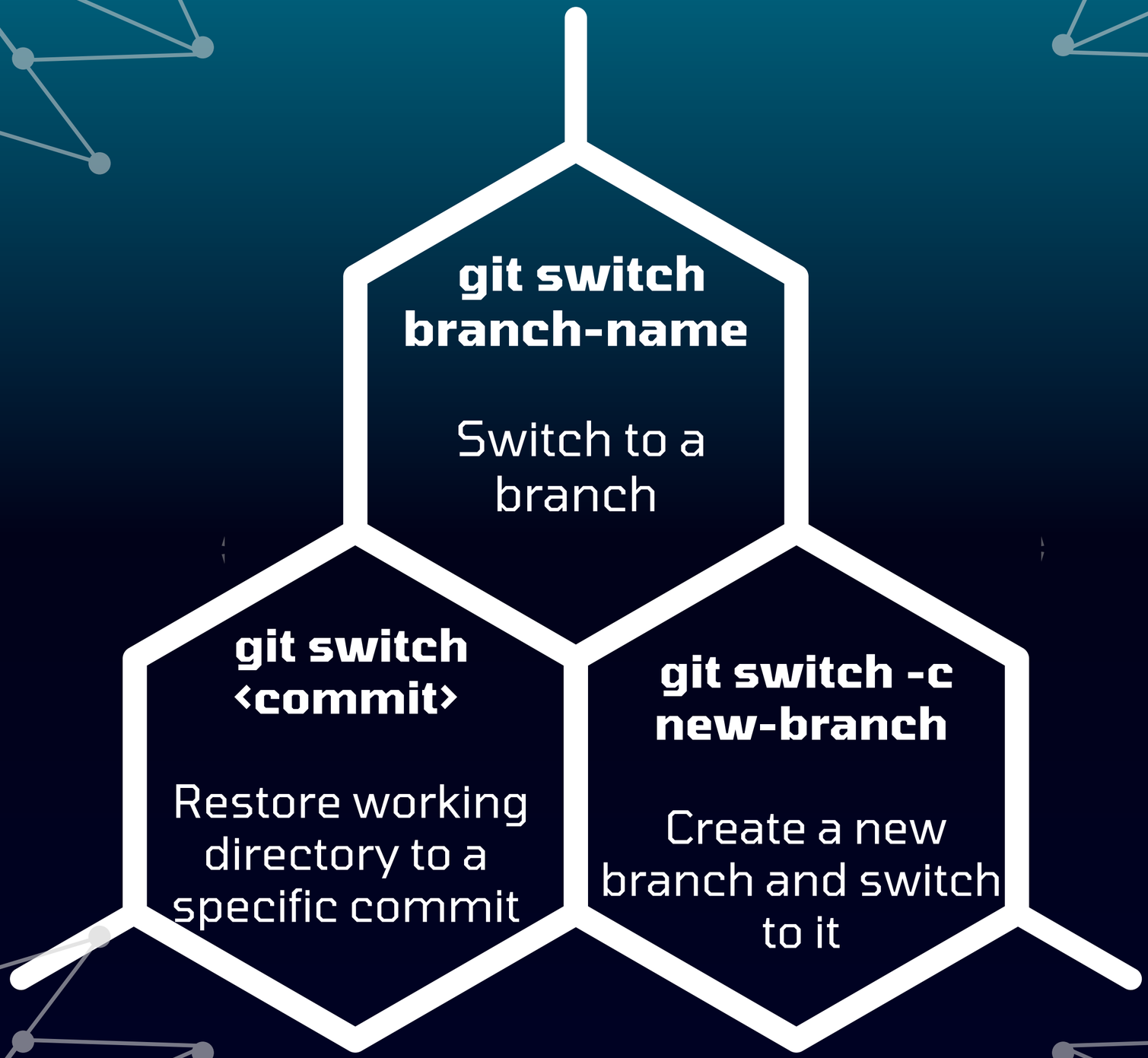
5 GIT SWITCH

Git switch is used to switch between branches or restore working directory states.

Quickly switch to a different branch without creating a new one.

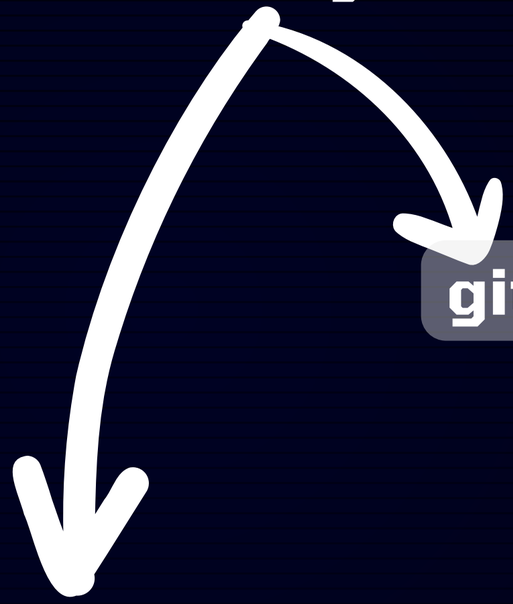


Restore the working directory to a specific commit or branch state.


git switch branch-name
Switch to a branch
git switch <commit>
Restore working directory to a specific commit
git switch -c new-branch
Create a new branch and switch to it

DIFFERENCE BETWEEN GIT SWITCH AND GIT CHECKOUT

The key difference between git switch and git checkout is that git switch is specifically designed for moving between branches, while git checkout can handle a broader range of operations, including moving between commits or checking out specific files.



git switch branch-name

```
git checkout commit-hash  # Move to a specific commit
git checkout branch-name  # Switch to a different branch
git checkout -- filename  # Check out a specific file from a commit or branch
```

6 GIT REBASE

Helps maintain a linear project history by avoiding merge commits.

Git rebase is used to integrate changes from one branch into another by reapplying commits on top of another branch.

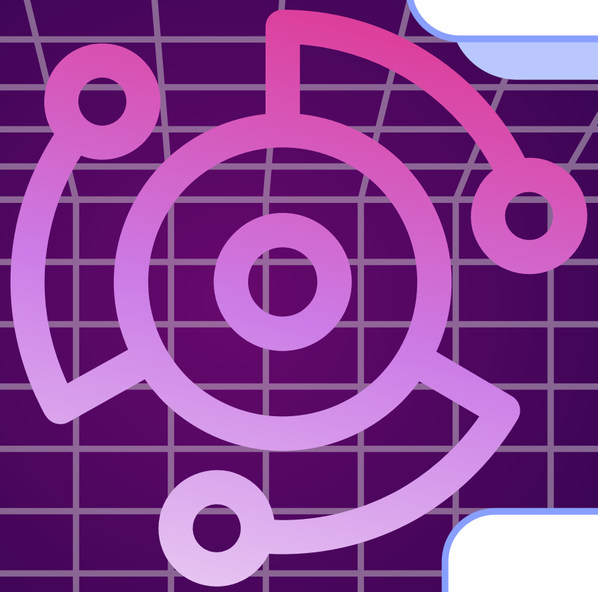
Useful for incorporating changes from a feature branch into the main branch.

git rebase --skip

Complete the rebase after resolving conflicts

git rebase main feature-branch

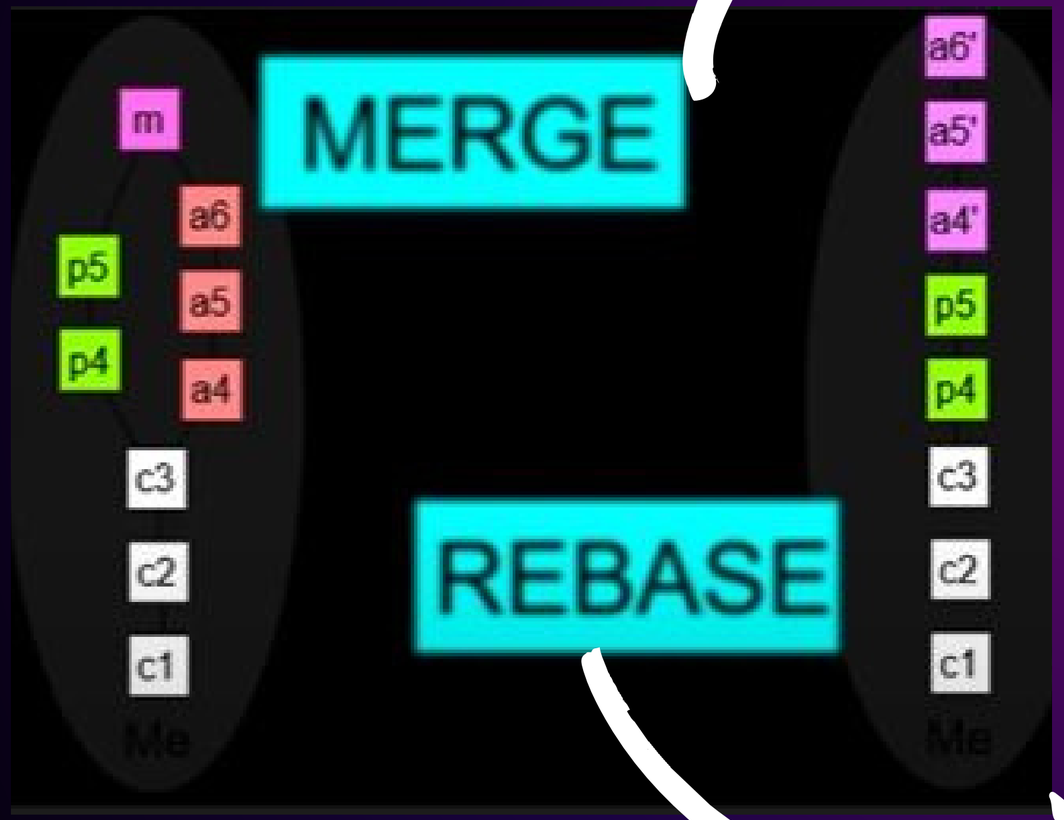
Start a rebase to integrate feature branch into main branch

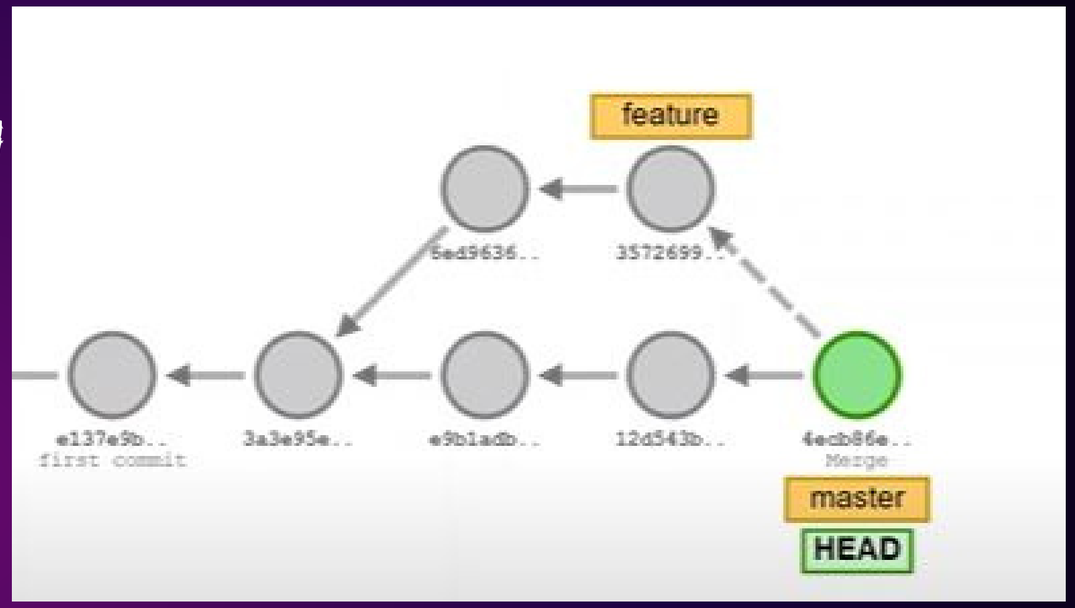


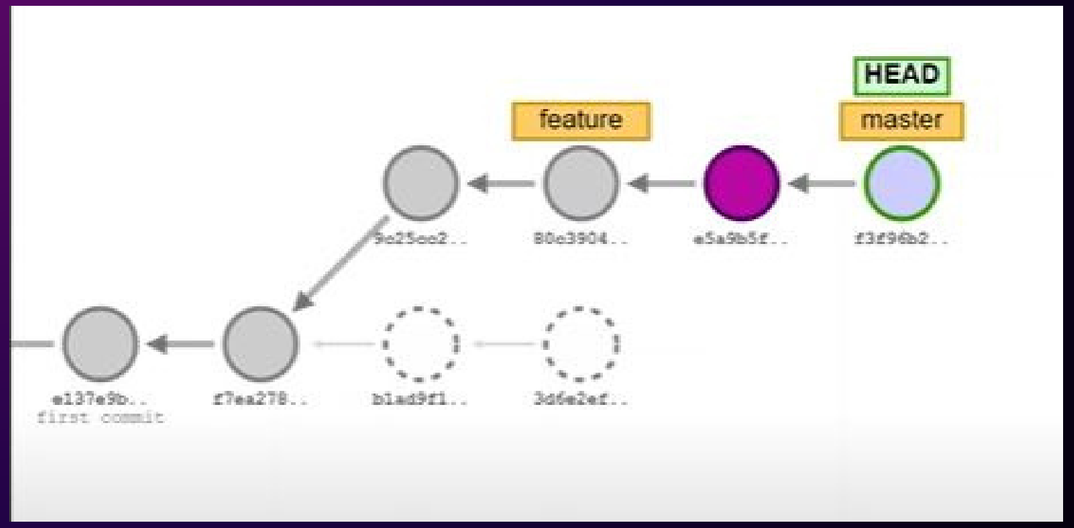
git rebase --continue

Resolve any conflicts during the rebase process

DIFFERENCE BETWEEN REBASE AND MERGE







7 GIT CHERRY-PICK

Git cherry-pick is used to apply specific commits from one branch to another.

- Features:
  - Selectively apply commits without merging entire branches.
  - Useful for porting bug fixes or features to different branches.

git cherry-pick <commit-hash>

Cherry-pick a commit from another branch

git cherry-pick <commit1> <commit2>

Apply multiple commits using cherry-pick

THANK YOU