Git branching commands:

מקרי שימוש(הנושאים צריכים לכלול שימושים מתקדמים.

חסרונות ויתרונות.

דוגמאות קידוד.

אלטרנטיביות.

צילומי מסך.

**Git rebase:**

**Use cases:**

The git rebase command allows you to easily modify a series of obligations efficiently and linearly, it allows changes from the main branch to your branch to be combined before the merger back and thus keeps the attribute branches up to date, keeps the history of the repository clean and clear and the possibility of organizing and editing commitments, resolving conflicts that occur when integrating changes from different industries to ensure a smoother integration process, During a rebase, several commitments that are not necessary to create a concise and organized history can be squashed.

**Advantages and disadvantages:**

advantages:

* Keeping a clean commit history - rebasing allows for a linear commit history by reapplying commits in a different branch, making the history easier to read and understand.
* Smoother integration – By integrating changes from one branch to another more efficiently, rebasing can result in fewer conflicts between mergers and a smoother integration process.
* The ability to delete a commit - Rebasing gives you the option of compressing multiple commits into one commit, and helps keep the commit history concise and focused.
* Keeps feature branches up to date - Using a rebase to pull changes from the master branch helps keep feature branches up to date with the latest codebase changes.

Disadvantages:

* Not suitable for shared workflows - when collaborating with other team members in a shared branch, using a rebase can introduce complexities and disrupt the workflow. Since rebasing requires history, this can create confusion for other contributors.
* Rewrites the history - one of the main disadvantages of rebase is that it rewrites the commit history, this can make it challenging to track changes over time and can cause confusion for collaborators.
* Complexity - Interactive rebase and conflict handling during rebase can add complexity to the process.

**alternative:**

Merge workflow - Instead of rewriting the commit history, the merge workflow creates a new commit that combines changes from two branches, preserving the original commit history.

How to use it - Use the -git merge command to merge changes from one branch to another, creating a merge commit that will reflect the merged history.

**Coding examples:**

**1.**

git checkout feature-branch

git rebase main

git push origin feature-branch

Explain: Switch to the feature branch, Update the feature branch with changes from the main branch, push the rebased feature branch.

**2.**

git checkout main

git pull

git checkout feature-branch

git rebase main

Explain: Update the main branch with the latest changes, Switch to the feature branch and rebase it onto the updated main branch.