## ****Create a Fixup Commit for**** "Added file1.txt"

### ****1. Find the Commit Hash of**** "Added file1.txt"

sh

CopyEdit

git log --oneline

This lists recent commits in a short format. Look for the commit with the message "Added file1.txt" and note its **hash** (e.g., abc1234).

### ****2. Create a Fixup Commit****

sh

CopyEdit

git commit --fixup=abc1234

### ****Explanation:****

* git commit --fixup=abc1234 creates a **fixup commit**, which is a special commit meant to amend a previous commit (abc1234 in this case).
* A fixup commit doesn’t modify the commit message but is designed to be **automatically squashed (merged) with the target commit** in the next rebase.

## ****Step 4: Perform an Interactive Rebase to Merge the Fixup Commit****

sh

CopyEdit

git rebase -i --autosquash abc1234^

### ****Explanation:****

* git rebase -i → Starts an **interactive rebase**.
* --autosquash → Automatically merges (squashes) the fixup commit with the original commit (abc1234).
* abc1234^ → Tells Git to start the rebase from **one commit before** abc1234.

### ****What Happens Next?****

1. Git opens an interactive editor showing a list of commits.
2. The fixup commit will be **automatically marked as squash** (you don’t need to change anything).
3. Simply **save and exit** the editor.
4. Git rewrites history, merging the fixup commit into abc1234.

## ****Final Step: Verify the Commit History****

sh

CopyEdit

git log --oneline

Summary Fixup commits are a useful tool for making small changes to previous commits without disrupting the work of other developers. By creating a fixup commit, you can make the necessary changes and have them automatically merged with the original commit during the next rebase.