**Lab Exercise 1- Working with Git Revert**

**Lab Exercise: Git Revert**

This exercise will guide you through reverting changes in Git. The git revert command is used to create a new commit that undoes the changes introduced by a previous commit without modifying the history.

**Objective**

* Learn how to use git revert to undo changes from specific commits.
* Practice handling merge conflicts during a revert.

**Prerequisites**

1. Install Git on your system.
2. Set up a Git repository:

git init git-revert-lab

cd git-revert-lab

**Steps**

**1. Set Up the Repository**

1. Create a file:

echo "Line 1" > file.txt

1. Stage and commit the file:

git add file.txt

git commit -m "Initial commit: Add file.txt with Line 1"

1. Add more changes:

echo "Line 2" >> file.txt

git commit -am "Add Line 2 to file.txt"

1. Add another change:

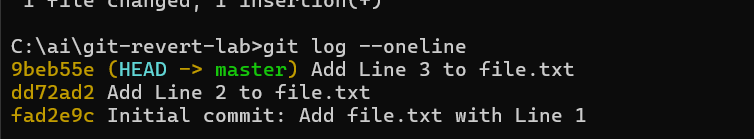
echo "Line 3" >> file.txt

git commit -am "Add Line 3 to file.txt"

1. Verify the commit history:

git log --oneline

Example output:

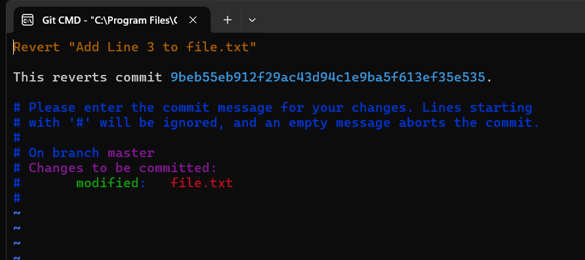


**2. Revert the Last Commit**

1. Revert the most recent commit:

git revert HEAD

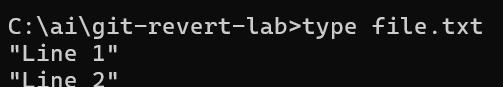
1. Git will open a text editor for the commit message. Save and close the editor to complete the revert.



1. Verify the contents of file.txt:

cat file.txt

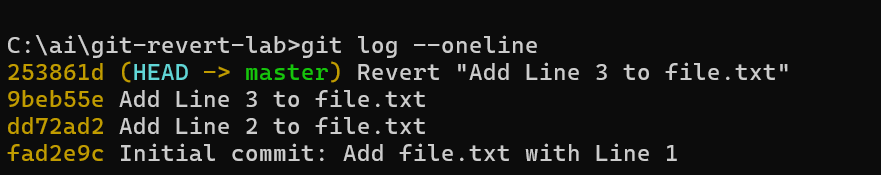
Output:



1. Check the commit history:

git log --oneline

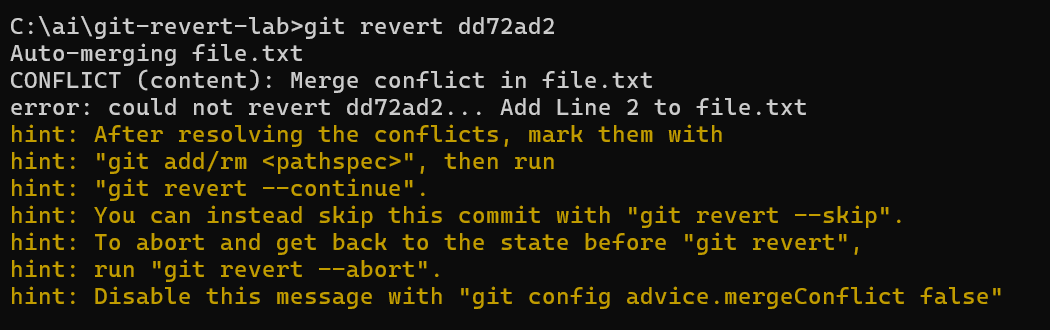
Example output:



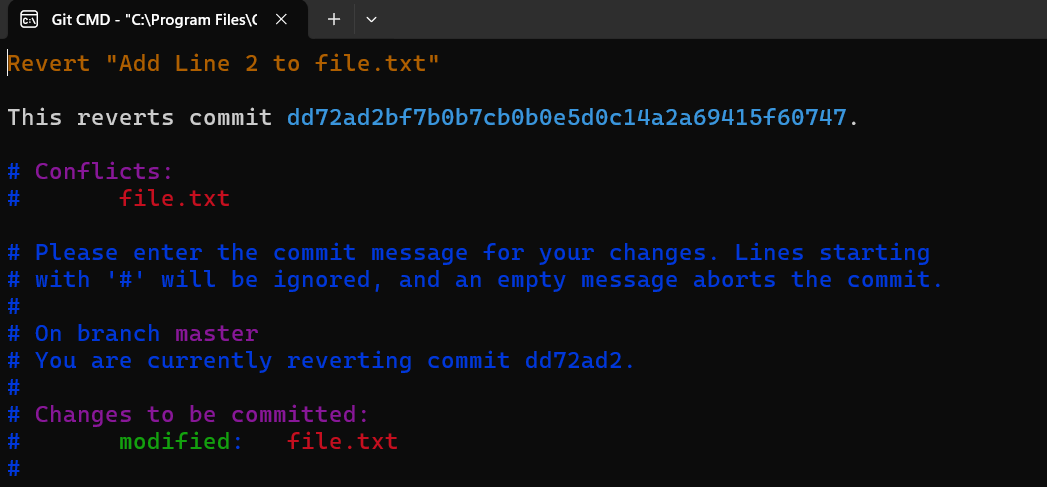
**3. Handle Merge Conflicts During Revert**

1. Modify file.txt:

echo "Conflict Line" >> file.txt

git commit -am "Add Conflict Line"

1. Revert the second commit (8b2a1c1 again) to trigger a conflict:

git revert 8b2a1c1

1. Git will indicate a conflict. Resolve it:
   * Open file.txt and remove conflict markers.
   * Keep the desired lines.
2. Stage the resolved file:

git add file.txt

1. Complete the revert:

git revert --continue

