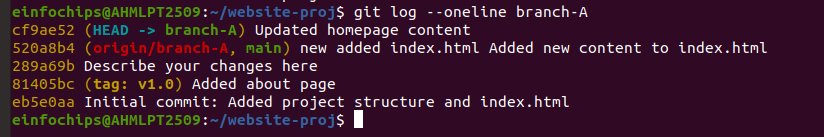
#### **Git Cherry Pick**

#### **Scenario:**

* You have two branches: branch-A and branch-B.
* You made a bug fix commit on branch-A that you now want to apply to branch-B without merging all changes from branch-A into branch-B.

#### **Steps:**

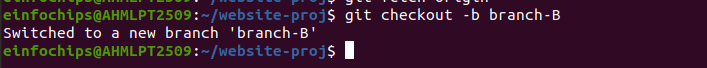
**Identify the Commit**:  
First, find the commit hash of the bug fix commit on branch-A:  
  
git log --oneline branch-A



1. Suppose the commit hash is abcdef1234567890.

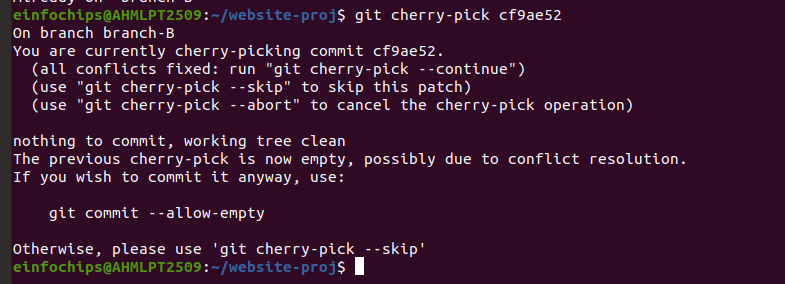
**Switch to branch-B**:  
Ensure you are on branch-B where you want to apply the bug fix:  
  
git checkout branch-B

git checkout -b branch-B



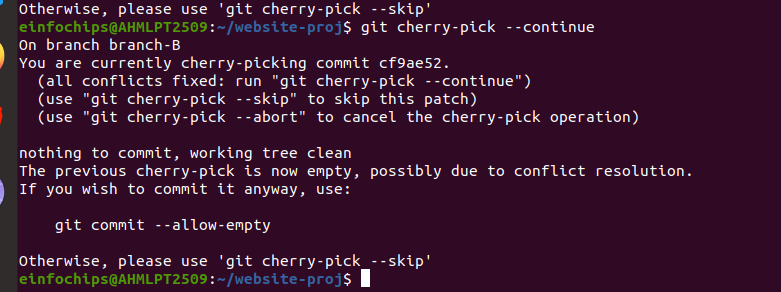
**Cherry-pick the Commit**:  
Apply the bug fix commit from branch-A to branch-B:  
  
git cherry-pick abcdef1234567890

git cherry-pick cf9ae52

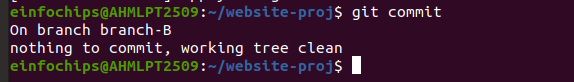


1. This command applies the changes introduced by the commit abcdef1234567890 onto branch-B.

**Resolve Conflicts (if any)**:  
  
git cherry-pick –continue



**Commit the Cherry-picked Changes**:  
After resolving conflicts (if any), commit the cherry-picked changes on branch-B:  
  
git commit



git commit --allow-empty -m "Apply changes from commit cf9ae52"



1. This creates a new commit on branch-B that includes the changes from branch-A's selected commit.

**Git Stash**

#### **Step 1: Initialize a Git Repository**

First, create a new directory for your project and initialize a Git repository:

mkdir git-stash-example



cd git-stash-example



git init

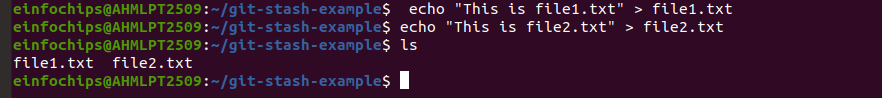


#### **Step 2: Add and Commit Files**

Create some files and add content to them:

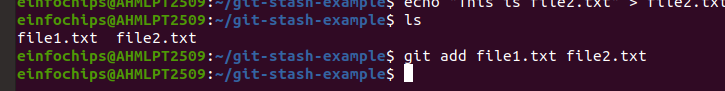
echo "This is file1.txt" > file1.txt

echo "This is file2.txt" > file2.txt

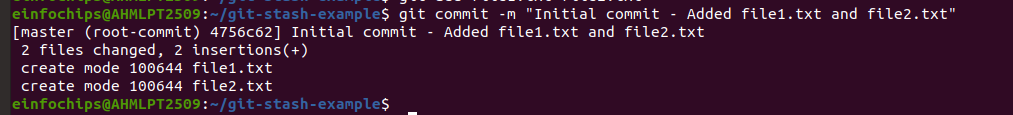


Add these files to the staging area and commit them:

git add file1.txt file2.txt



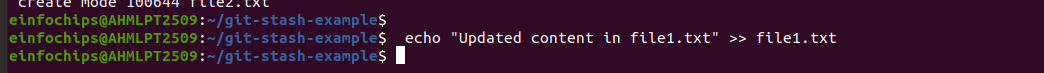
git commit -m "Initial commit - Added file1.txt and file2.txt"



#### **Step 3: Modify Files**

Make some changes to file1.txt:

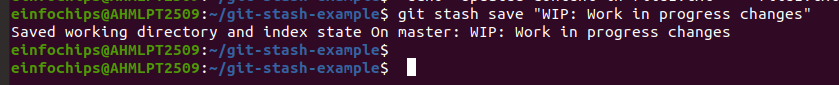
echo "Updated content in file1.txt" >> file1.txt



#### **Step 4: Use git stash**

Now, let's use git stash to temporarily store the changes in file1.txt without committing them:

git stash save "WIP: Work in progress changes"

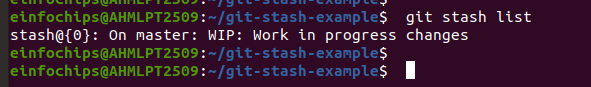


This command saves your changes (in this case, the update to file1.txt) to a stash with a message "WIP: Work in progress changes".

#### **Step 5: Verify Stash**

You can verify the stash list using:

git stash list



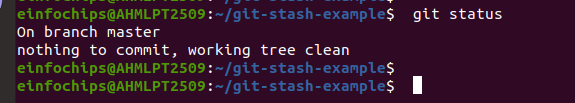
It should show something like:

stash@{0}: On master: WIP: Work in progress changes

#### **Step 6: Check Working Directory Status**

Check the status of your working directory:

git status

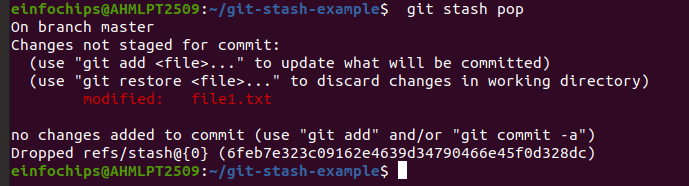


It should indicate that your working directory is clean (no changes).

#### **Step 7: Apply Stashed Changes**

Let's apply the stashed changes back into your working directory:

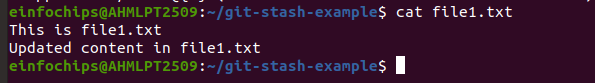
git stash pop



#### **Step 8: Verify Changes**

Check the changes in file1.txt:

cat file1.txt



#### **Step 9: Commit Stashed Changes**

If you are satisfied with the changes, commit them:

git add file1.txt



git commit -m "Updated file1.txt with stashed changes"

