

Gemini Chat

i have a github repo
my friend comitted to the repo
and i was working on a previous version

what to do

- ✦ This is a very common situation in Git! The solution is to **pull** your friend's changes from GitHub and combine them with your local work.



You have two main ways to do this: using **rebase** (which is often preferred for a cleaner history) or **merge** (which is the default and a bit simpler).

1. Before You Do Anything: Commit Your Work

First, you *must* save your own work. If you have any changes you were working on, commit them locally.

1. Check what files you've changed:

Bash



```
git status
```

2. Add and commit your changes:

Bash



```
git add .  
git commit -m "My work in progress"
```

Note: If `git status` says "nothing to commit, working tree clean," you can skip this step and go straight to step 2.

2. Choose Your Method to Get Updates

Now, you'll pull your friend's changes.

Option A: The "Cleaner" Way (Recommended)

Use `git pull --rebase`. This will:

1. Fetch your friend's commits.
2. Temporarily "lift" your new commits off.
3. Place your friend's commits down first.
4. Re-apply *your* commits one-by-one **on top** of your friend's work.

This creates a clean, linear history, as if you had done your work *after* your friend, even though you did it at the same time.

Command:

Bash



```
git pull --rebase
```

Option B: The "Simpler" Way (Default)

Use `git pull`. This will:

1. Fetch your friend's commits.
2. Automatically create a new "**merge commit**" that ties your work and your friend's work together.

This is simple, but it adds an extra merge commit to your project's history, which can make it look a bit cluttered.

Command:

Bash



```
git pull
```

3. What if You Get a "Merge Conflict"?

Don't panic! This is normal. It just means you and your friend edited the **exact same lines** in a file, and Git doesn't know whose change to keep.

Git will stop and tell you which file(s) have a conflict.

1. **Open the file** in your code editor.
2. You will see markers like this:

```
<<<<<< HEAD
This is your change (what you wrote).
=====
This is your friend's change.
>>>>>> [commit hash]
```

3. **Fix the file:** Edit the text to be how you want it. **Delete all the** `<<<<<`, `=====`, **and** `>>>>>` **markers.**
4. **Save the file.**
5. **Tell Git you're done:**
 - **If you used** `rebase` :

Bash



```
git add <file_name_you_fixed>
git rebase --continue
```

- **If you used** `merge` **(default pull):**

Bash



```
git add <file_name_you_fixed>
git commit
```

(A commit message will pop up; you can just save and close it.)

Summary: Your Step-by-Step Plan

Here is what I recommend you do:

1. Save your work:

Bash



```
git add .
git commit -m "My local changes"
```

2. Pull using rebase:

Bash



```
git pull --rebase
```

3. Fix conflicts (if any) by editing the files, then run:

Bash



```
git add <fixed-file-name>
git rebase --continue
```

4. Push your work:

Once everything is combined and working, you can push your commits (and your friend's, which are now part of your history) back to GitHub:

Bash



```
git push
```

```
(base) jkb@jkb-HP-ProBook-450-15-6-inch-G10-
Notebook-
PC:~/Documents/sem3/osn/finalproject/course-project-
codeshark$ git pull
git: 'credential-
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' is not
a git command. See 'git --help'.
git: 'credential-
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' is not
a git command. See 'git --help'.
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 9 (delta 5), reused 9 (delta 5), pack-
reused 0 (from 0)
Unpacking objects: 100% (9/9), 1.34 KiB | 171.00 KiB/s,
done.
From https://github.com/CS3-OSN-Monsoon-
2025/course-project-codeshark
   391b1e7..71cc7f1  main    -> origin/main
Updating 391b1e7..71cc7f1
error: Your local changes to the following files would be
overwritten by merge:
   name_server/nm.c
Please commit your changes or stash them before you
merge.
Aborting
```



Ah, this is the exact situation we talked about!



That error message is Git's way of protecting you: `error: Your local changes to the following files would be overwritten by merge: name_server/nm.c`

It means:

1. You have *uncommitted* changes in the file `name_server/nm.c`.
2. The `git pull` you ran fetched new work from your friend, and that new work *also* includes changes to `name_server/nm.c`.
3. Git has stopped, because it refuses to just wipe out your uncommitted changes.

As the message says, you have two choices: **commit** your changes or **stash** them.

The Easiest Fix: `git stash`

This is the best option if you're not ready to make a permanent commit. `git stash` temporarily "stashes" (saves) your local changes, cleans your working directory, lets you pull, and then lets you re-apply your changes.

Step 1: Stash your changes Run this command. It will take your changes to `nm.c` and save them in a temporary holding area.

Bash



```
git stash
```

Your `git status` will now show your directory is "clean."

Step 2: Pull the new code (with `rebase`) Now that your directory is clean, you can safely pull. I still recommend using `-rebase` for a clean history.

Bash



```
git pull --rebase
```

This will fetch your friend's work and put it down first.

Step 3: Re-apply your stashed changes Now, bring your own work back from the holding area.

Bash



```
git stash pop
```

This re-applies the changes you made to `nm.c` on top of the new code.

What if you get a conflict NOW?

After `git stash pop`, you might *still* get a merge conflict. This is normal. It just means you and your friend edited the *exact same lines* in `nm.c`.

If this happens:

1. Open `name_server/nm.c` in your editor.
2. You'll see the `<<<<<`, `====`, `>>>>>` markers.
3. Edit the file to fix it, deleting the markers.
4. Save the file.
5. Run `git add name_server/nm.c` to mark it as resolved.

(The `git: 'credential-ghp_...'` error is a separate configuration issue, but it didn't stop you from pulling, so you can likely ignore it for now.)

```
(base) jkb@jkb-HP-ProBook-450-15-6-inch-G10-  
Notebook-  
PC:~/Documents/sem3/osn/finalproject/course-project-  
codeshark$ git status
```

On branch main

Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.

(use "git pull" to update your local branch)

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: name_server/nm.c

no changes added to commit (use "git add" and/or "git commit -a")

(base) jkb@jkb-HP-ProBook-450-15-6-inch-G10-Notebook-

PC:~/Documents/sem3/osn/finalproject/course-project-codeshark\$ git add .

git commit -m "My work in progress"

[main 7726e91] My work in progress

1 file changed, 64 insertions(+), 3 deletions(-)

(base) jkb@jkb-HP-ProBook-450-15-6-inch-G10-Notebook-

PC:~/Documents/sem3/osn/finalproject/course-project-codeshark\$ git pull

git: 'credential-

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' is not a git command. See 'git --help'.

git: 'credential-

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' is not a git command. See 'git --help'.

hint: You have divergent branches and need to specify how to reconcile them.

hint: You can do so by running one of the following commands sometime before


```
hint: your next pull:
hint:
hint: git config pull.rebase false # merge
hint: git config pull.rebase true  # rebase
hint: git config pull.ff only      # fast-forward only
hint:
hint: You can replace "git config" with "git config --
global" to set a default
hint: preference for all repositories. You can also pass -
-rebase, --no-rebase,
hint: or --ff-only on the command line to override the
configured default per
hint: invocation.
fatal: Need to specify how to reconcile divergent
branches.
```



You're on the right track! You successfully committed your changes.



This new message (`fatal: Need to specify how to reconcile divergent branches`) is happening for a clear reason:

- You created a new commit locally ("My work in progress").
- Your friend created a new commit on GitHub.

Your project's history has "diverged" — it has split into two different paths.

Git is now simply asking you how you want to combine these two paths. You just have to tell it which strategy to use: **rebase** or **merge**.

Your Two Options (Same as Before)

You just need to run `git pull` again, but this time, tell it *how* to combine the histories.

Option 1: Rebase (Recommended)

This is the cleanest option. It will take your commit ("My work in progress") and place it *on top* of your friend's changes.

Run this command:

Bash



```
git pull --rebase
```

Option 2: Merge (The alternative)

This will create a new "merge commit" that ties your work and your friend's work together. It's fine, but adds a bit of clutter to the history.

Run this command:

Bash



```
git pull --no-rebase
```

(or `git pull --merge`)

What to do now

Just pick one of those commands. I **recommend** `git pull --rebase` .

After you run it, you *might* get a merge conflict (if you both edited the same lines). If you do, just follow the steps from my first answer to fix it, then run `git rebase --continue` .

```
(base) jkb@jkb-HP-ProBook-450-15-6-inch-G10-
Notebook-
PC:~/Documents/sem3/osn/finalproject/course-
project-codeshark$ git pull --rebase
git: 'credential-
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'
is not a git command. See 'git --help'.
git: 'credential-
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'
is not a git command. See 'git --help'.
Successfully rebased and updated
refs/heads/main.
```



Excellent! That's perfect.



The message `Successfully rebased and updated refs/heads/main` means it worked exactly as planned.

Here's what just happened:

1. Git fetched your friend's commit.
2. It put your friend's commit down first.
3. It then re-applied your commit ("My work in progress") **on top** of your friend's work.

Your local `main` branch now has your friend's changes *plus* your changes, all in a nice, clean, single line of history.

The Final Step: Push Your Changes

Your local repository is now ahead of the remote repository on GitHub. The last step is to push your combined history back to GitHub so your friend can see your work.

Just run:

Bash



```
git push
```

(Don't worry about those `git: 'credential-...'` errors. They are just noise from a slight misconfiguration but clearly aren't stopping you.)

You're now using Gemini 3 Pro

We've upgraded you from the previous model to 3 Pro, our newest model.