In Git, "stashing" refers to the process of temporarily saving changes in your working directory that are not ready to be committed or that you don't want to commit yet. Stashing allows you to switch to a different branch, perform other tasks, or resolve urgent issues without committing half-finished work. Once you've completed the other tasks, you can reapply the stashed changes to your working directory.

Here's how to use Git stash:

**1. Stash Your Changes:**

To stash your changes, open your terminal and navigate to your Git repository. Use the following command:

Replace **"Your stash message"** with a descriptive message to help you identify the stash later. This message is optional.

**2. Check the Status:**

You can run **git status** to ensure that your working directory is clean.

**3. Perform Other Tasks:**

Now, you can switch branches, make other changes, or perform any necessary actions without your unfinished changes interfering.

**4. Apply the Stash:**

Once you're ready to reapply your stashed changes, use one of the following methods:

**Apply the Latest Stash:**

**Apply a Specific Stash:**

Use **git stash list** to see a list of all your stashes along with their identifiers (stash numbers). Then, apply the desired stash by specifying its identifier:

Replace **n** with the stash number you want to apply.

**Apply and Remove the Latest Stash:**

**Apply and Remove a Specific Stash:**

**5. Resolve Conflicts (if any):**

If there are conflicts between your stashed changes and the changes in the branch where you're applying the stash, you'll need to resolve them manually.

**6. Commit the Changes:**

After resolving conflicts (if any) and verifying that everything is as expected, commit the changes as you normally would.

**7. Delete the Stash (Optional):**

To delete a specific stash, use the following command:

Replace **n** with the stash number you want to delete. Be cautious when deleting stashes, as they cannot be recovered once deleted.

Stashing is a handy feature for managing your work and switching between different tasks in Git without the need to create unnecessary intermediate commits. It's particularly useful when working on multiple branches or when you need to switch to a different branch temporarily.