## **7. Git Stash (Temporary Save)**

Sometimes, you are working on a feature but need to switch branches or pull updates without committing your unfinished work. **Git stash** allows you to temporarily save (stash) your changes and apply them later.

### **7.1 Stashing Changes**

To save your current work without committing:

git stash

This stashes your changes and resets your working directory to the last committed state.

**Example:**  
You modified index.html and style.css, but you need to pull the latest changes from main before committing.

git stash

git pull origin main

git stash pop

Now, your local files are updated, and your changes are restored.

### **7.2 Applying Stashed Changes**

To bring back the stashed changes:

git stash pop

This restores the changes and removes them from the stash list.

Alternatively, if you want to keep the changes in the stash while applying them:

git stash apply

This applies the changes but keeps them in the stash.

### **7.3 Viewing Stash List**

To see all stashed changes:

git stash list

**Example Output:**

stash@{0}: WIP on feature-branch: abc1234 Updated homepage layout

stash@{1}: WIP on main: def5678 Fixed navbar bug

### **7.4 Dropping Stashed Changes**

If you no longer need a stash, you can delete it:

git stash drop stash@{0}

Or clear all stashes:

git stash clear

### **Example Workflow**

#### **Scenario:**

You're working on feature-branch, but you need to switch to main to fix a bug.

#### **Steps:**

git stash # Save your unfinished work

git checkout main # Switch to main branch

git pull origin main # Get latest changes

git checkout feature-branch # Switch back

git stash pop # Restore your work

### **Summary**

Would you like a real-world example or demo?