Git Command Line Workflow Cheatsheet

This guide covers essential commands and concepts for synchronizing changes between your local machine and GitHub, especially when mixing CLI and Web UI work.

🚀 The Reliable Synchronization Script (git_sync.sh)

This script is your solution for two-way synchronization. It packages your local work, pulls any web changes, merges them, and pushes the final state back to GitHub.

Script Content:

```
#!/bin/bash

# --- 1. Get Commit Message ---
DEFAULT_MESSAGE="Cleaning up files/sync from local to web"
echo "Enter commit message (or press Enter to use default):"
read -r USER_MESSAGE

if [ -z "$USER_MESSAGE" ]; then
    COMMIT_MESSAGE="$DEFAULT_MESSAGE"
else
    COMMIT_MESSAGE="$USER_MESSAGE"
fi

echo "--- Using commit message: \"$COMMIT_MESSAGE\" ---"

# --- 2. Git Operations ---

# Stage all changes (additions, modifications, deletions)
echo "Staging all changes..."
git add -A

# Commit staged changes
echo "Committing staged changes..."
git commit -m "$COMMIT_MESSAGE"

# Check if the commit was successful before proceeding
if [ $? -ne 0 ]; then
    echo "Warning: No new changes to commit. Proceeding with pull/push sync."
fi
```

```
# PULL: Fetch and merge remote changes (Web -> Local)
echo "Pulling remote changes from origin main..."
git pull origin main

# PUSH: Send combined changes to the remote branch (Local -> Web)
echo "Pushing local changes to origin main..."
git push origin main
```

How to Use:

1. Save the content above into a file named git_sync.sh (if you haven't already).
2. Make it executable: chmod +x git_sync.sh
3. Run the script: ./Scripts/git_sync.sh (or wherever you placed it).

🧠 Essential Git Status & Workflow Checks

git status is the most important command—always run it to know the state of your work.

Status Message	Meaning	Action When Working
nothing to commit, working tree clean	Everything is saved and ready for the next task.	Start new work.
Your branch is ahead...	You have local commits not on GitHub.	Your sync script will fix this.
Your branch is behind...	GitHub has changes you don't have locally.	Your sync script will fix this.
Untracked files:	You created a file that Git is ignoring.	Your sync script's git add -A will pick it up.
Automatic merge failed; fix conflicts...	**CRITICAL STOP.** You have a merge conflict.	See the section on Merge Conflicts below.

🗑️ Cleanup Commands: Tracking vs. Deleting

Remember the difference between telling Git to ignore a file and physically deleting it.

Command	Effect on Git History	Effect on Local Disk	Purpose
git rm --cached <file>	**Deletes from history** (next commit)	**Kept** (becomes an untracked file)	Mass cleanup / Stop tracking specific files.
git rm <file>	**Deletes from history** (next commit)	**Deleted** permanently.	Standard way to delete a tracked file.
rm <file>	**No change to Git** (deletion noted by git add)	**Deleted** permanently.	Purely a file system command.

💥 Troubleshooting: Merge Conflicts

A merge conflict occurs when you and the remote repository change the **same lines in the same file** between synchronizations. The git pull step will halt and require manual intervention.

Fix Steps:

1. **Stop:** Do not run any more Git commands until the conflict is resolved.
2. **Identify:** Run git status to see the "unmerged paths."
3. **Edit:** Open the conflicted file(s). You will see markers:
   <<<<<<< HEAD
   // YOUR local code version
   =======
   // The code version from the Web/GitHub
   >>>>>>> main

4. **Resolve:** Manually choose the correct version of the code, delete the conflict markers (<<<<<<<, =======, >>>>>>>), and save the file.
5. **Finalize:** Tell Git the file is fixed, then finish the merge.
   git add <conflicted-file>
   git commit -m "Resolved merge conflict"
   git push origin main