**Testing mood for second branch--009**

**1. git init**

👉 **ဘာလုပ်တာလဲ**

* Git repository အသစ်တစ်ခု ဖန်တီးပေးတယ်။
* ဆိုလိုသည်မှာ, Git ကို "ဒီ folder ထဲမှာ commit တွေ track လုပ်တော့မယ်" လို့ပြောတာ။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* မရှိသေးတဲ့ project folder တစ်ခုကို Git နဲ့ စဖမ်းချင်ရင် သုံးတယ်။
* အဲဒီအချိန်မှာ .git/ folder တစ်ခုဖန်တီးပြီး commit, branch, log တွေကိုထည့်သွားမယ်။

-------------------------------------------------

**2. git clone**

👉 **ဘာလုပ်တာလဲ**

* Remote repository (GitHub, GitLab, etc.) ကို **local computer** သို့ copy လုပ်ပေးတယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* GitHub repo တစ်ခုကို download လုပ်ပြီး ကိုယ့် machine ပေါ်မှာ code edit လုပ်ချင်တဲ့အချိန် သုံးတယ်။

👉 **Example**

git clone https://github.com/your-username/movie-app.git

👉 Result: movie-app folder ထဲမှာ repo အတိအကျ copy ရလာမယ်။

## 3. git add

👉 **ဘာလုပ်တာလဲ**

* File တစ်ခု (သို့) တချို့ file တွေကို **staging area** (index) ထဲ ထည့်တယ်။
* staging ဆိုတာ "အခု commit လုပ်မယ့် ဖိုင်တွေ" ဆိုတဲ့ နေရာ။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* ပြင်ဆင်ထားတဲ့ file တိကို commit လုပ်မယ့်အတွက် stage လုပ်ရမယ်။

👉 **Example**

git add frontend/auth.js

👉 Result: frontend/auth.js ကို commit-ready ဖြစ်သွားမယ်။

## 4. git add \*

👉 **ဘာလုပ်တာလဲ**

* Project folder အတွင်းရှိ **အားလုံး file** တွေကို staging area ထဲထည့်တယ် (new/modified files).

👉 **ဘာကြောင့်သုံးရတာလဲ**

* Project ရဲ့ ပြောင်းလဲမှုတွေ အားလုံးကို commit လုပ်ချင်တဲ့အချိန် သုံးတယ်။

👉 **Example**

git add \*

👉 Result: အဲဒီ folder ထဲမှာ ပြောင်းလဲထားတဲ့ file အားလုံး stage လုပ်မယ်။

⚠️ **အကြံပြုချက်** – တခါတလေ unwanted files (log, temp, build/) လည်း stage သွားနိုင်တယ်,  
ဒါကြောင့် .gitignore သေချာထားသင့်တယ်။

## 5. git reset

👉 **ဘာလုပ်တာလဲ**

* Already staged (git add လုပ်ပြီးသား) file တွေကို staging ထဲမှ ထုတ်ပစ်တယ်။
* သို့မဟုတ်, commit တစ်ခုကို ပြန်ပြောင်းချင်တဲ့အချိန်သုံးတယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* Accidentally git add \* လုပ်သွားတဲ့အချိန် reset လုပ်နိုင်မယ်။
* သေချာ မပြီးသေးတဲ့ file တွေကို commit မထည့်ချင်ရင် unstaged လုပ်နိုင်မယ်။

👉 **Example (unstage files only)**

git reset

👉 Result: အစောဆုံး git add လုပ်ထားတဲ့ file တွေ commit-ready မဖြစ်တော့ဘဲ unstaged ပြန်ဖြစ်သွားမယ်။

👉 **Example (unstage specific file)**

git reset frontend/auth.js

👉 **Example (reset to a specific commit)**

git reset --hard abc123

👉 Result: HEAD pointer ကို commit abc123 သို့ ပြန်ဆွဲပြီး working directory ကိုပါ အဲဒီအခြေအနေအတိုင်း ပြန်ပြောင်းမယ်။

⚠️ **သတိ** – --hard သုံးမယ်ဆိုရင် unsaved changes တွေ လုံးဝပျက်သွားနိုင်တယ်။

✅ **အချုပ်**

* git init → Repo အသစ်ဖန်တီး
* git clone → Remote repo ကို local copy
* git add → File တစ်ခုကို stage
* git add \* → File အားလုံး stage
* git reset → Staged files / commit ကို ပြန်ဖြုတ်

## 1. git status

👉 **ဘာလုပ်တာလဲ**

* Repo ရဲ့ လက်ရှိအခြေအနေကိုပြတယ်။
* ဘယ် files တွေ **staged**, **unstaged**, **untracked** ဖြစ်နေသလဲ ဆိုတာပြပေးတယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* Commit လုပ်မယ့်အချိန် file တွေ့မှားသွားမလား စစ်နိုင်မယ်။

👉 **Example**

git status

👉 Result (ဥပမာ):

On branch fix-login-bug

Changes to be committed:

modified: frontend/auth.js

Untracked files:

frontend/auth.test.js

## 2. git commit -m "message"

👉 **ဘာလုပ်တာလဲ**

* Staged files တွေကို commit လုပ်ပြီး log ထဲသိမ်းတယ်။
* -m option က commit message ကို command ထဲမှာပဲရေးချင်တဲ့အချိန် သုံးတယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* File changes တွေကို history အဖြစ် သိမ်းထားပြီး future မှာပြန်ကြည့်နိုင်မယ်။

👉 **Example**

git commit -m "fix(auth): trim email input in resetPassword"

## 3. git commit -a -m "message"

👉 **ဘာလုပ်တာလဲ**

* Modified/Deleted files တွေကို auto-stage လုပ်ပြီး commit တစ်ခါတည်းလုပ်တယ်။
* ❌ New (untracked) files မဖြစ်ပါဘူး။ (အဲဒီအတွက် git add လိုအပ်တယ်။)

👉 **ဘာကြောင့်သုံးရတာလဲ**

* တဆင့်ချင်း git add လုပ်စရာမလိုဘဲ modified files တွေကို တိုက်ရိုက် commit လုပ်ချင်တဲ့အချိန်။

👉 **Example**

git commit -a -m "refactor: update login UI colors"

## 4. git push

👉 **ဘာလုပ်တာလဲ**

* Local branch commits တွေကို Remote (GitHub, GitLab, etc.) သို့ upload လုပ်တယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* ကိုယ့် machine ထဲမှာသာမက GitHub/GitLab ပေါ်မှာလည်း changes တွေ share လုပ်နိုင်မယ်။

👉 **Example**

git push origin fix-login-bug

👉 Result: fix-login-bug branch ကို GitHub repo ပေါ်တင်မယ်။

## 5. git pull

👉 **ဘာလုပ်တာလဲ**

* Remote repo ထဲမှာရှိတဲ့ commits တွေကို local repo ထဲသို့ update လုပ်တယ်။
* git fetch (သစ် commit တွေ ယူချီ) + git merge (local branch ထဲထည့်) ဆိုတဲ့အဓိကအလုပ် ၂ခုကို တစ်ပြိုင်နက် လုပ်ပေးတယ်။

👉 **ဘာကြောင့်သုံးရတာလဲ**

* တူညီတဲ့ project ကို အများစု code changes လုပ်တဲ့အချိန် Local repo ကို up-to-date ထားဖို့။

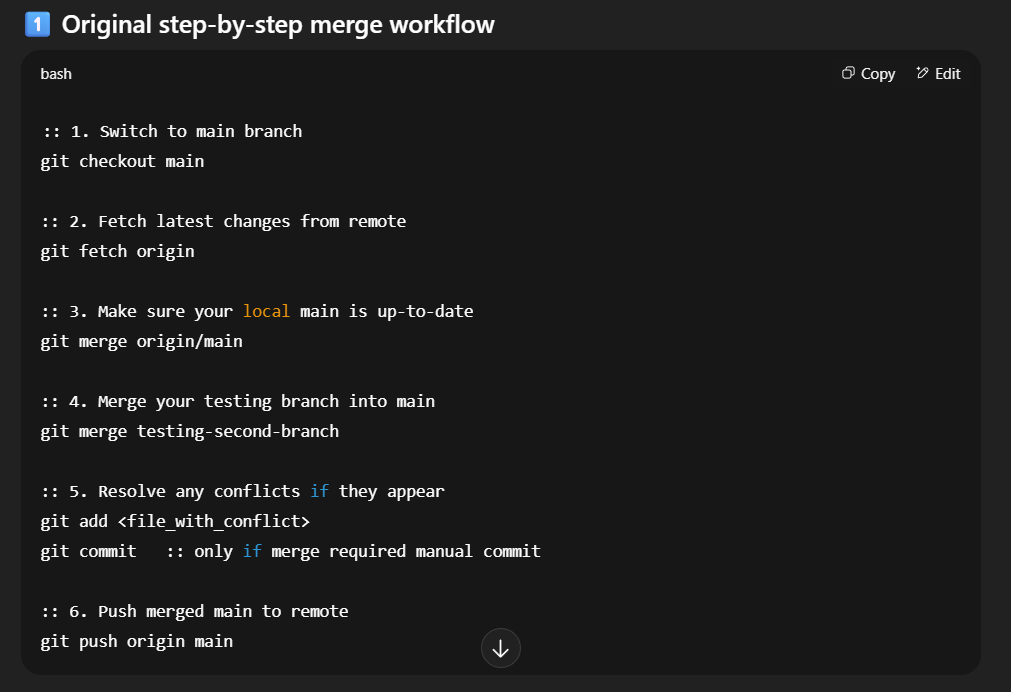
👉 **Example**

git pull origin main

👉 Result: GitHub ရဲ့ main branch commits တွေ local main branch ထဲသို့ update ဖြစ်မယ်။

✅ **အချုပ် (Easy Map)**

* git status → ဘာပြောင်းလဲမယ် စစ်မယ်
* git commit -m → Staged files commit + message
* git commit -a -m → Modified files auto-stage + commit
* git push → Local commits ကို GitHub သို့ upload
* git pull → GitHub changes ကို local သို့ update



 git checkout main → main branch ကို သွားမယ်။

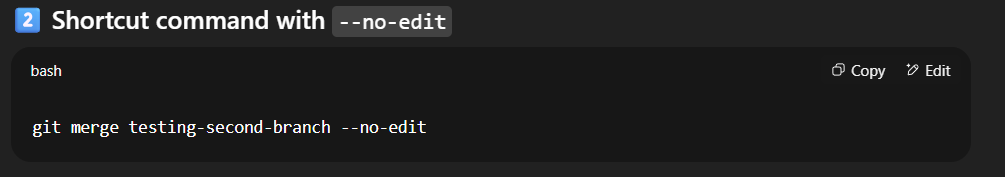
 git fetch origin → Remote repository က အခုဆုံး updates တွေကို ရယူမယ်။

 git merge origin/main → Local main ကို remote ရဲ့ latest version နဲ့ update လုပ်မယ်။

 git merge testing-second-branch → testing-second-branch ကို local main နဲ့ ပေါင်းမယ်။

 git add <file\_with\_conflict> & git commit → Conflict ရင် အဲဒီ file ကို ပြင်ပြီး commit လုပ်မယ်။

 git push origin main → Merged main ကို remote repository ကို push လုပ်မယ်။



### **Explanation (Myanmar / မြန်မာဘာသာ)**

* testing-second-branch ကို လက်ရှိ branch (main ဖြစ်တတ်) နဲ့ merge လုပ်မယ်။
* --no-edit → Git ကို Vim editor ကိုမဖွင့်ဘဲ default merge commit message ကို အလိုအလျောက်သုံးစေတယ်။
* Conflict ရင်တော့ လက်ဖြင့် ပြင်ရမယ်။

