Git - Core Ideas

Perfect Drawing Room

Imagine a **perfect drawing room** — a magical space where you can **undo any stroke**, **repaint without fear**, and **return to any moment in time**.

What If You Could

- Experiment freely
- Keep every version of your artwork
- Revisit past drafts
- Collaborate with others without losing your own changes

That’s what **Git** does — but for **code instead of paint**.

Drawing Room and Git Equivalent

Concept	Drawing Room	Git Equivalent
Canvas	Working Canvas	Workspace
Decide to Keep	Pin board (decide what to preserve)	Staging area (Index)
Save	Commit to your collection	Commit (Snapshot in history/repository)
Rooms	Different rooms in the same gallery	Branch
Time machine	Versions of paintings	Checkout
Collaboration	Shared gallery	Remote repository (GitHub, etc.)

The Rhythm of Git

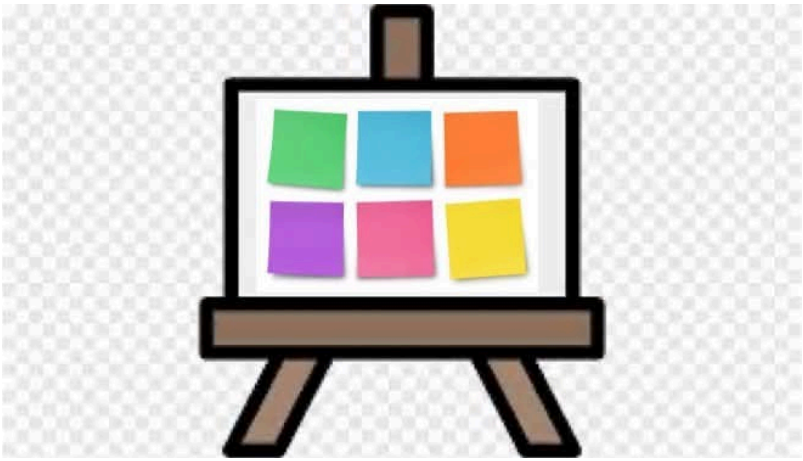
> Making a PostIt note → Pin what you like → Save forever.

1. **Workspace (Canvas)**
    - Where you create and modify your work.
    - This is your **Working Directory** — the active space for changes.

2. **Index / Staging Area (Pin Board)**
    - When you like something, you **pin** it — marking it for saving.
    - This step expresses the **decision to keep** selected changes.
3. **History / Repository (Gallery Collection)**
    - When you **commit**, you take a **snapshot** and store it forever.
    - Each commit becomes part of your project's **permanent timeline**.

Magic Gallery

In this **Gallery**, we create our masterpieces by sketching ideas on **Post-it notes** in many **Rooms**.



Working in the Gallery

- The **Canvas** is our **Working Directory (WD)** — where we draw, erase, and experiment freely.
- Each **Post-it note** represents a **file** we're working on.
- When we decide certain notes are worth keeping, we **track** them — telling Git, "Pay attention to these ones!"

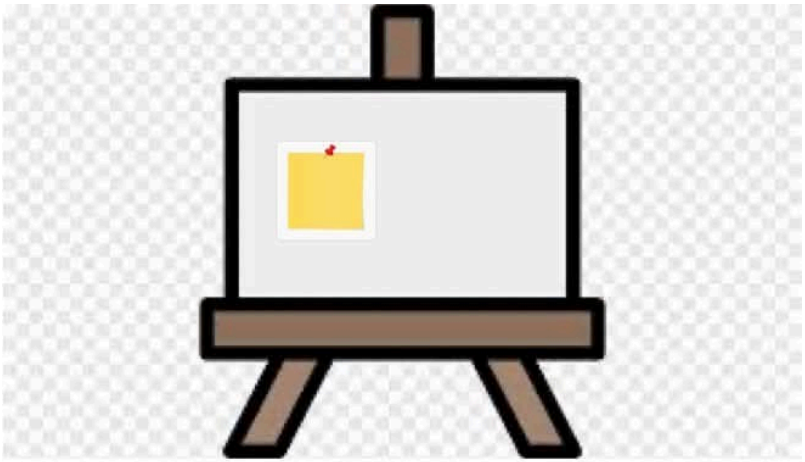
Let's Choose a Room and Start Drawing

In our **Gallery**, we first choose a **room** to draw in; Let’s pick the room called **“main.”**

- The **Room** is called a **Branch** in Git.
- Every project starts in the **main branch** — your default creative space.

Pin board (Staging Area/Index)

When you're OK with some Post-it notes, you **pin** them to your board so they don't get lost.

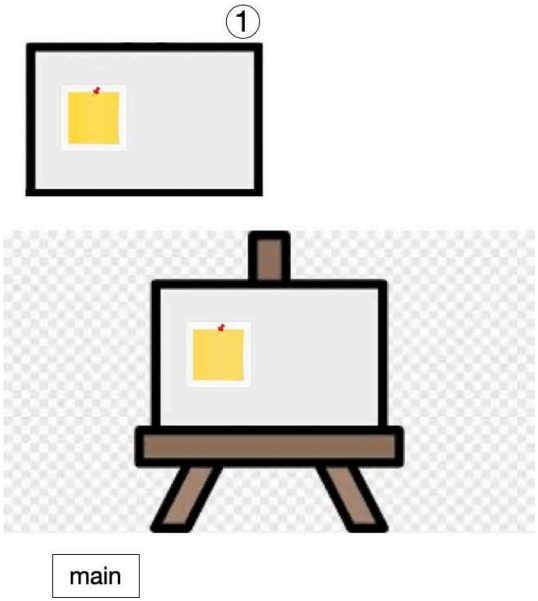


main

- This **Pin Board** represents Git's **Staging Area (Index)**.
- You're telling Git, "I'm ready to keep these in my next saved (snapshot) version."

Commit to the Collection

When you're happy with your notes and think they're worth keeping, you **commit** them to your collection — so they're **saved forever** nd can be **retrieved anytime**..



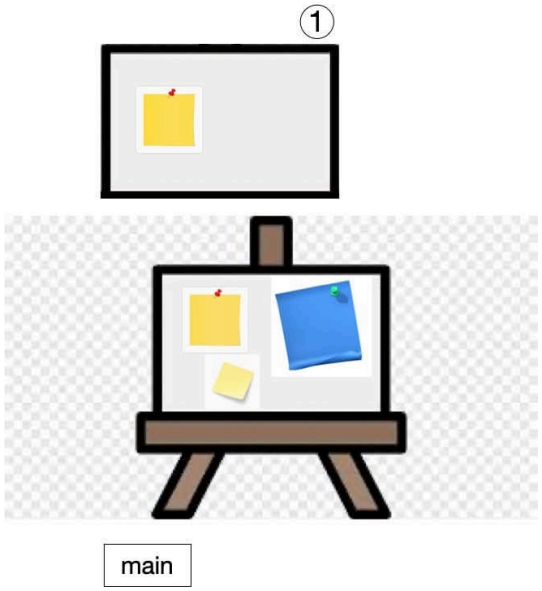
- Notice only the pinned notes (staged files) are stored in the snapshot (history).

- This action is called **taking a snapshot** or **storing it in history** in Git.
- Each **commit** is like numbering your finished canvas — a permanent record in your creative timeline.

> **Key Idea:**
> Every commit captures a moment in time — your project's memory, one snapshot at a time.

Changes and Add to the Index

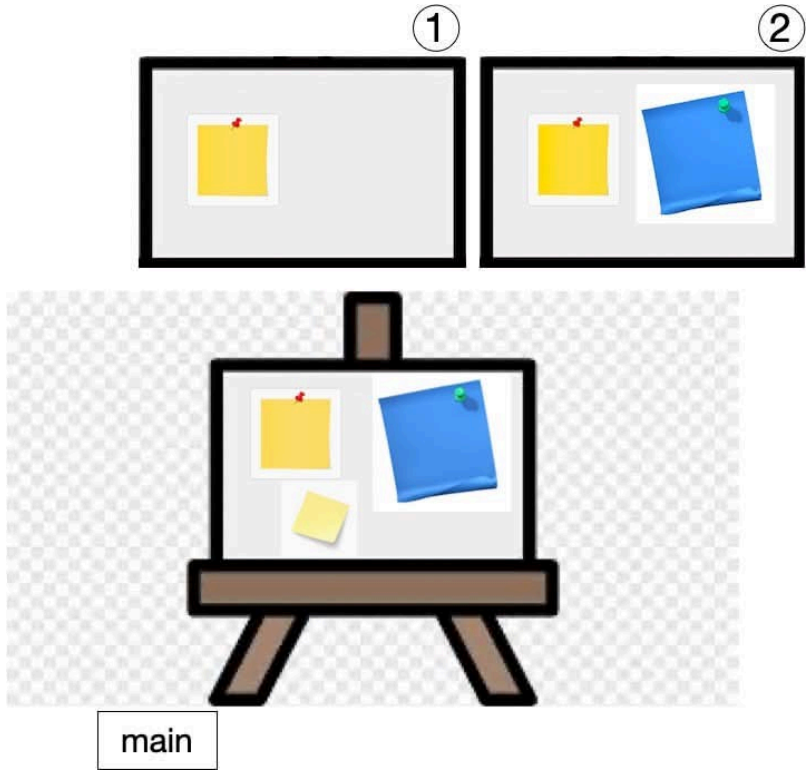
After saving our first version, we can continue to **make more changes** — adding a **blue** and a **yellow** Post-it note.



- We **pin** the new blue note to the board
  → this means we've **added it to the Index (Staging Area)**.
- The yellow one is still **untracked** — not yet pinned.

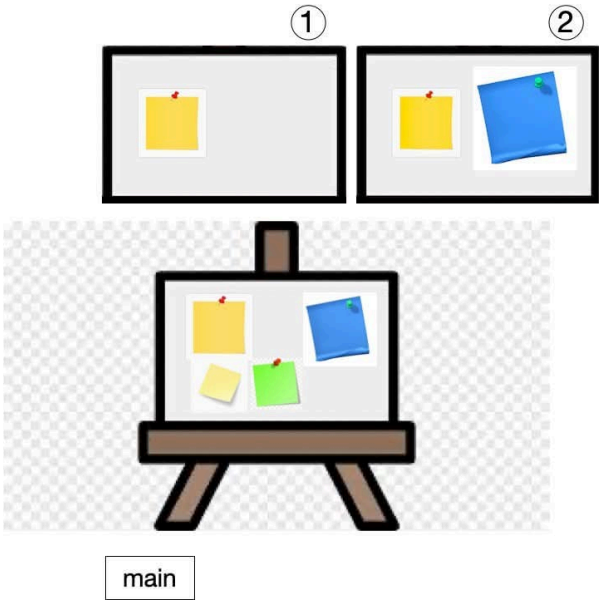
Commit Again

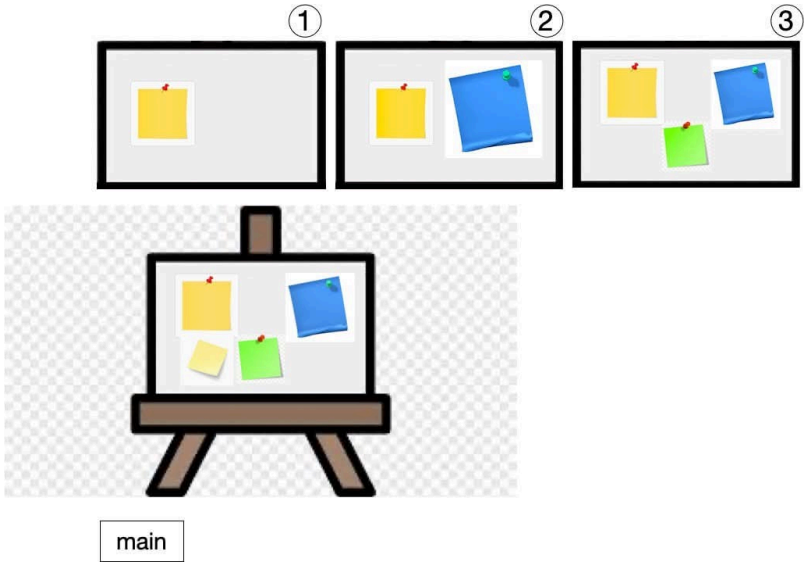
When we're satisfied again, we **commit** again the pinned notes — creating **another snapshot** in our project's history.



Keep adding & committing

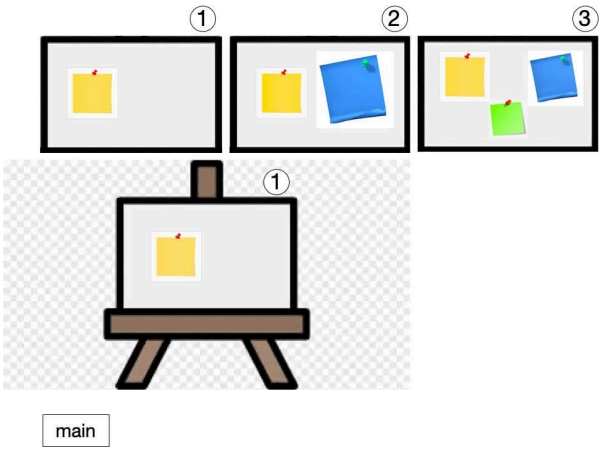
We can keep adding (green postIt) to the index (Staging Area) and committing to make a snapshot in history.





Checkout (Time Machine)

We can **travel back in time** to any previous moment using Git's **checkout** command.

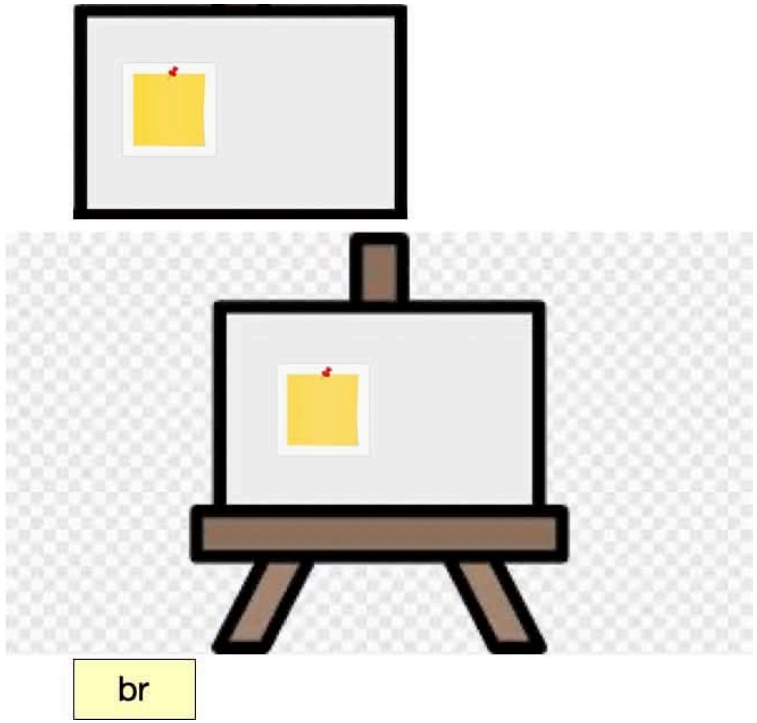


- Each **commit** is a *snapshot in history*.
- By checking out an earlier one, we return to that **exact version of the canvas**.
- In this example, we can **checkout snapshot #1** to revisit our original drawing.

Branching (a New Room)

When we **checkout** snapshot #1, we can start adding new ideas **without overwriting the old version** — by creating a **new branch**.

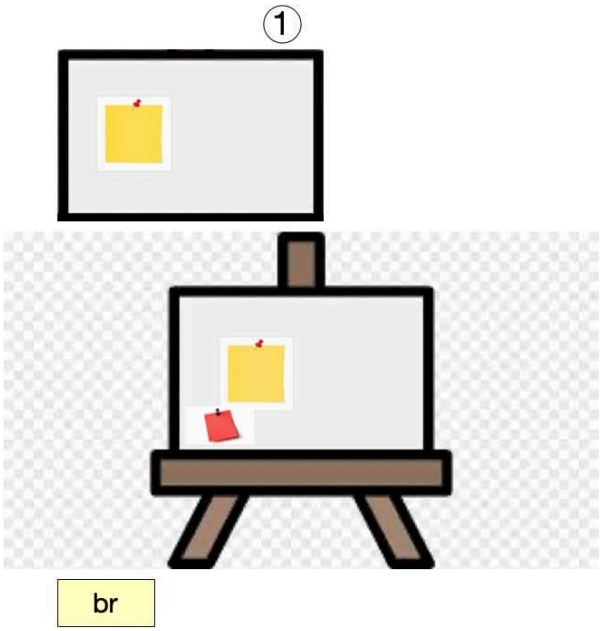
- This is like opening a **new room in the same gallery**, where we can continue our work independently.

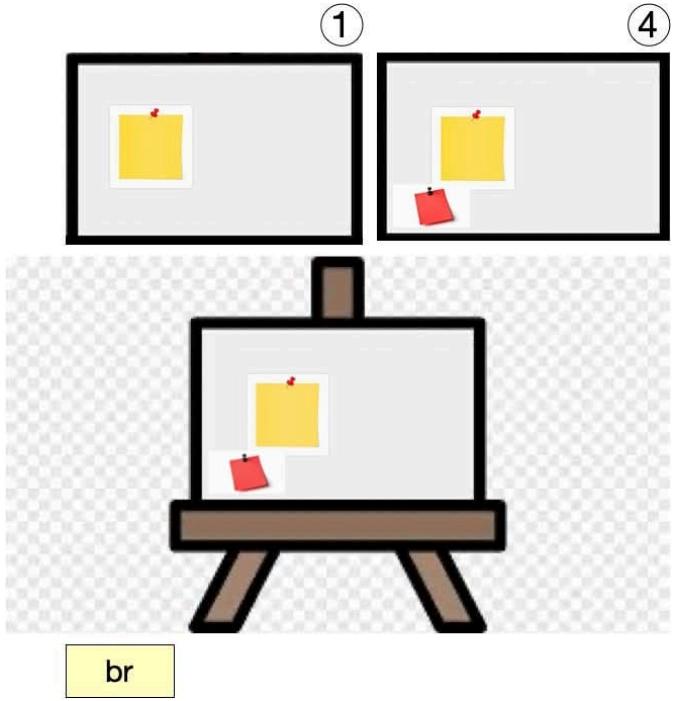


- The previous room (main) remains unchanged, preserving its original artwork.
- I can start working in a new room (branch) named "br".

Keep Working on a new Branch (a New Room)

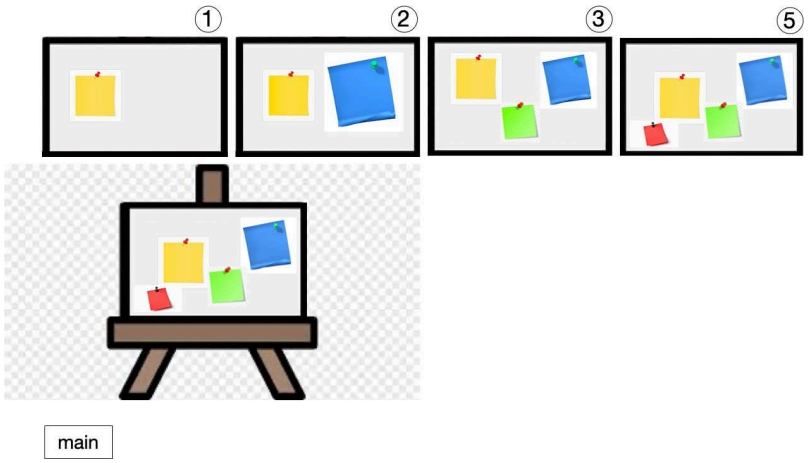
Now that we're in our **new room (branch)**, we can continue our creative journey — adding new **Post-it notes** and saving them as new snapshots.





Merging Two Rooms

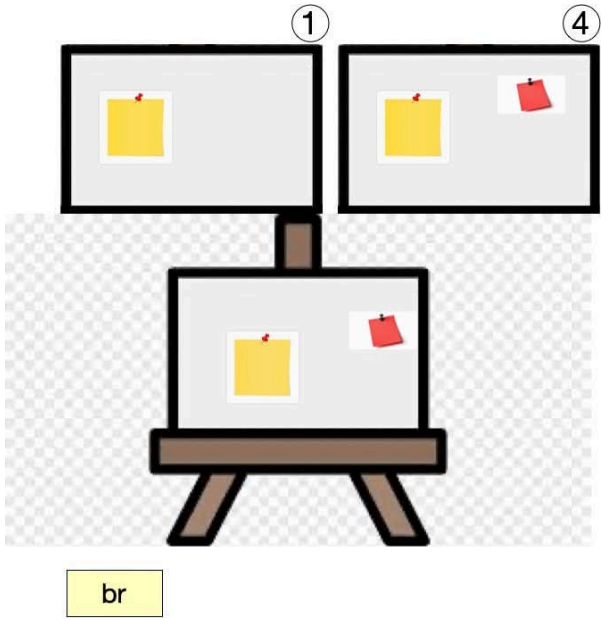
We can **merge the two rooms** whenever necessary: in this example, we bring the updates from the **“br” room** into the **“main” room**.

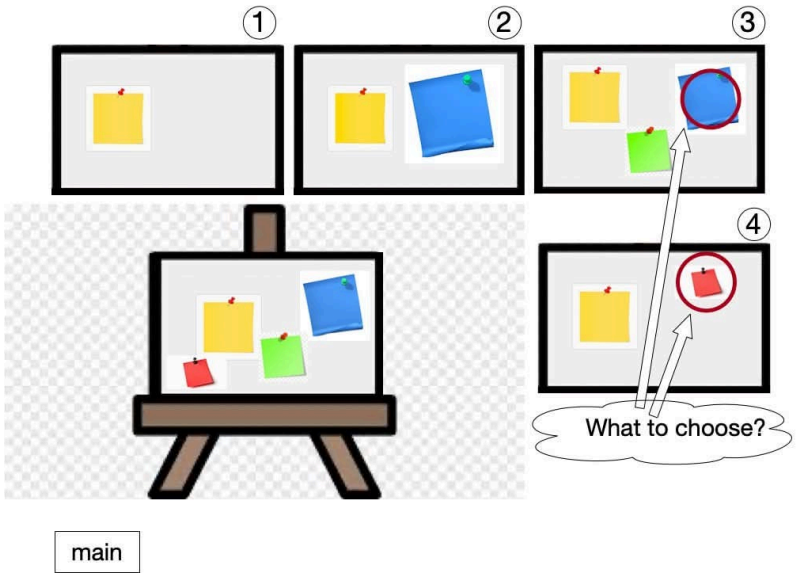


- The **magic** is that Git can **merge automatically** when there are **no conflicts** between the changes.

The Conflict

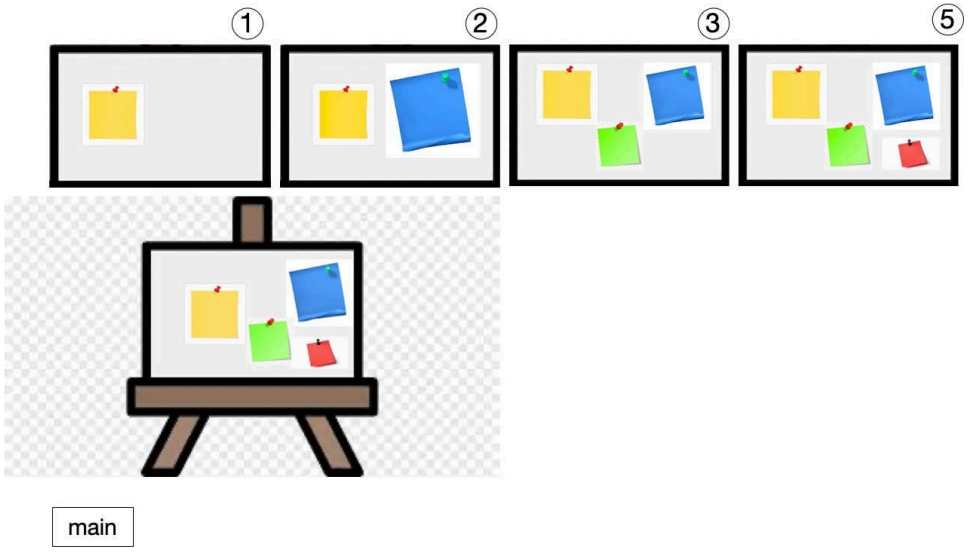
What if we changed the **same spot on the same canvas** (same file & same lines) in both `br` and `main`, then tried to merge?





Manual Resolution

In this case, the only solution is to resolve the conflicts manually.



The Git Life Cycle

This is how we use Git.

- We start with the main branch "branch".
- We keep adding/deleting/updating files & directories to the Index/Staging Area.
- We make a snapshot in history/respository.
- We make a branch when we need to do something different.
- We merge two branches and resolve the conflicts.