# Git Workflow: main vs origin/main

## Step 1 — Clone the repo

git clone https://github.com/user/repo.git

**Result:**

(main) A---B---C  
(origin/main) A---B---C

Both point to commit C (latest on remote).

Git stores these pointers inside .git/:

.git/  
 refs/  
 heads/  
 main ← your local branch pointer  
 remotes/  
 origin/  
 main ← remote tracking branch

## Step 2 — Make changes locally

# edit files  
git add file1 file2  
git commit -m "My changes"

**Result:**

(main) A---B---C---D ← your new commit  
(origin/main) A---B---C

main is ahead of origin/main by 1 commit.

## Step 3 — Push your changes

git push origin main

**Result:**

(main) A---B---C---D  
(origin/main) A---B---C---D

Both are now in sync.

## Step 4 — Someone else updates remote

Remote after another developer pushes:

Remote main: A---B---C---D---E

Before fetching:

(main) A---B---C---D  
(origin/main) A---B---C---D

## Step 5 — Fetch changes

git fetch

**What happens:**

* Git contacts the remote repository.
* Updates origin/main to match the remote branch.
* Your main branch does **not** change.

**Result:**

(main) A---B---C---D  
(origin/main) A---B---C---D---E

✅ git fetch = *sync remote → origin/main only*.

## Step 6 — Pull changes

git pull origin main

**What happens:**

1. Git first does a fetch (remote → origin/main).
2. Then merges (or rebases) origin/main → main.

**Result:**

(main) A---B---C---D---E  
(origin/main) A---B---C---D---E

✅ git pull = *fetch + merge*.

## Key Recap

* **Clone** → both pointers are same.
* **Commit locally** → main moves forward, origin/main stays put.
* **Push** → remote updates, origin/main matches main.
* **Fetch** → updates only origin/main.
* **Pull** → fetches into origin/main, then merges into main.