Branches, merge & rebase

branches

> What is a branch and why?
> Branching means you diverge from the main line of development and continue to do work without messing with that main line.

create a branch

```
git branch <branch-name>
```

> ⚠ Attention
> After you make a new branch, git doesn't checkout(move to) the new branch automatically.

switch to a branch

```
git checkout <targetbranch>
```

💡 You can use the `-b` option with checkout to create a branch and switch to it.

```
git checkout -b <branch-name>
```

> ⓘ Info
> Branches are virtually `free` just a SHA to a commit/tree.

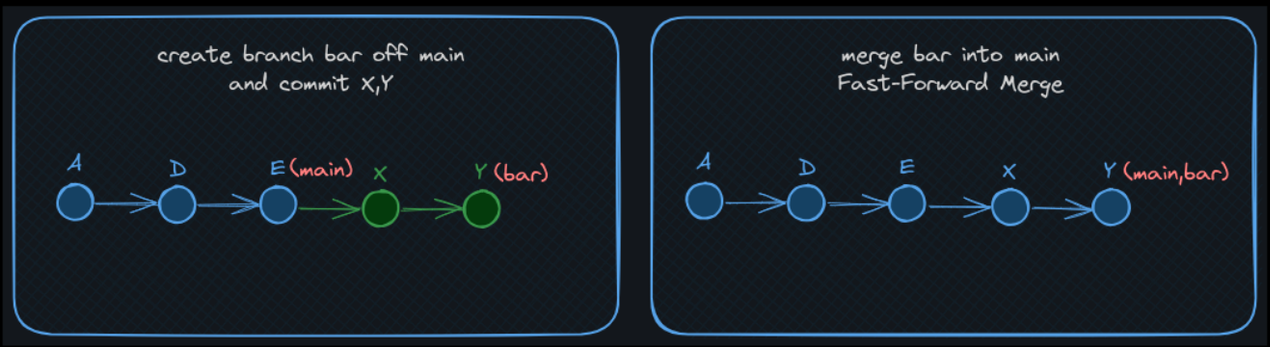
merge

> what is a merge
> A merge is attempting to `combine` two histories together that have diverged at some point in the past. There is a common commit point between the two, this is referred to as the `best common ancestor`

merge have 2 different outcomes

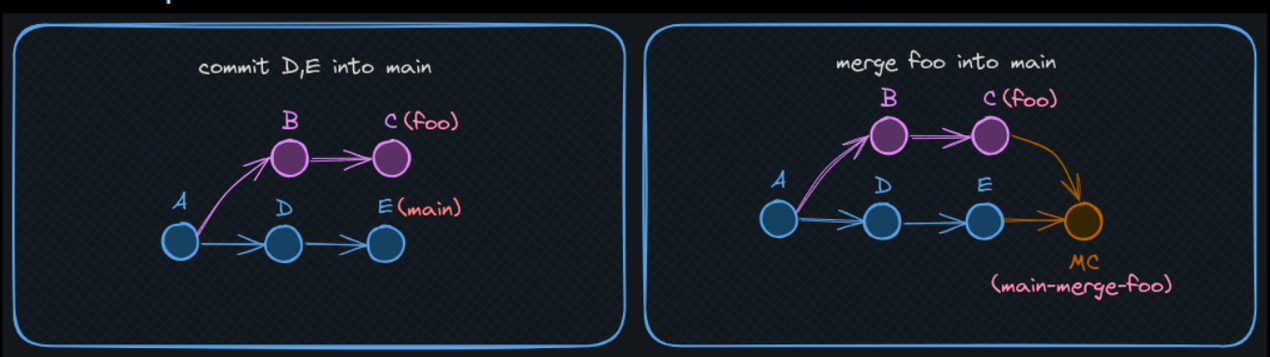
1. Fast Forward:

- just update the pointer/reference (no merge commits)



2. Divergence merge:

- create a merge commit to combine 2 commits/histories have 2 parents



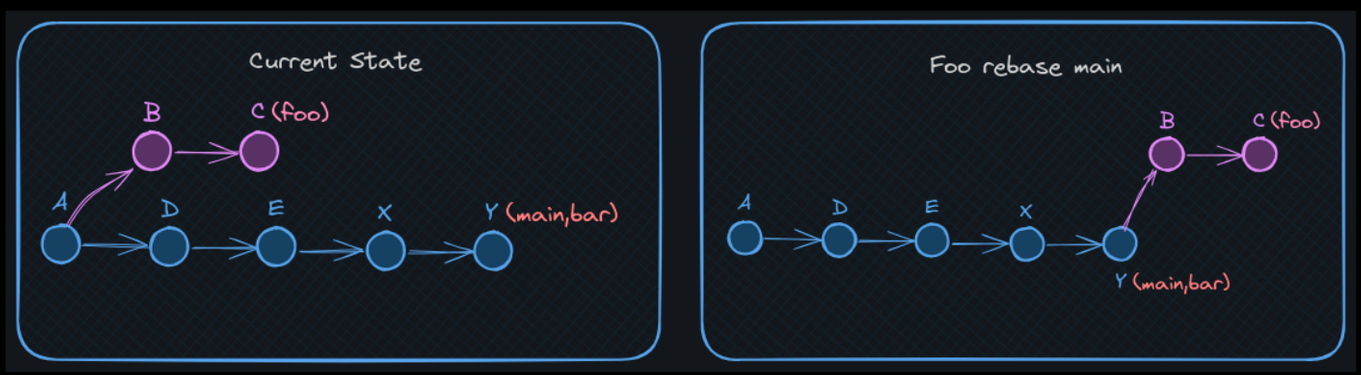
how to merge

```
git merge <source-branch>
```

- target-branch is the currently checked-out branch
- conflicts are discussed on the going remote section.

rebase

> what is a rebase?
>
> git-rebase - `Reapply` commits on top of another base tip
>
> - *"the docs"*



how to rebase

```
git rebase <target-branch>
```

> NOTE
>
> Note the different perspective of merge's one.
> rebase alters your branch to be at the tip of
> `<targetbranch>`

How rebase actually works

1. checkout the latest commit at `<target-branch>`
   - think of `<target-branch>` as `main`
2. replay one commit at a time of the `<source-branch>`
   - `<srouce-branch>` is often the feature branch.
3. update source branch ref to the latest commit made.

> Be careful
>
> Rebase alters history (notice: replay), so don't ever
> rebase main or any other public shared branch

---

merge vs. rebase

- Merge:
  - doesn't alter history
  - doesn't require `push force`
  - works with private and public branches without problems
  - makes annoying merge commits
- Rebase:
  - alters history
  - requires `push force`
  - works best with private branches only
  - no annoying merge commits
  - linear history which is easier to search

---

workflows war

Merge flow

- just merge always
  - merge back into main
  - merge commits happen but we live with it

Rebase flow

- (rebase in private branches - FF merge in public branches)
  - rebase main into your feature branch first
  - then fast-forward merge into main

> **Important**
> In simple cases the difference between the 2 options in practice is what option you use in the following list. but people seem to be really opinionated about this matter.
> anyway `rebase flow` is the best.