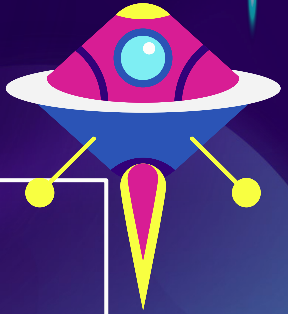
CH4

Branches

merge & rebase

Oh, do you really develop everything on main?
That’s a problem.

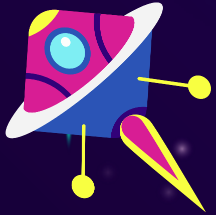


Branches

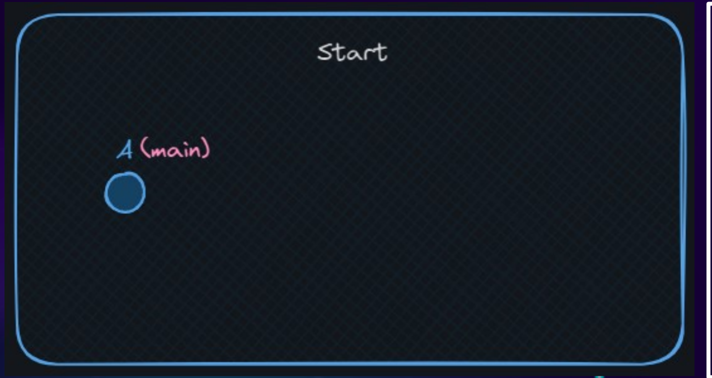
We all know what is a branch but nobody asks why is a branch.

“Branching means you diverge from the main line of development and continue to do work without messing with that main line, to be safe and keep the main safe from you.”

Mission 0



Desired State



git-branches

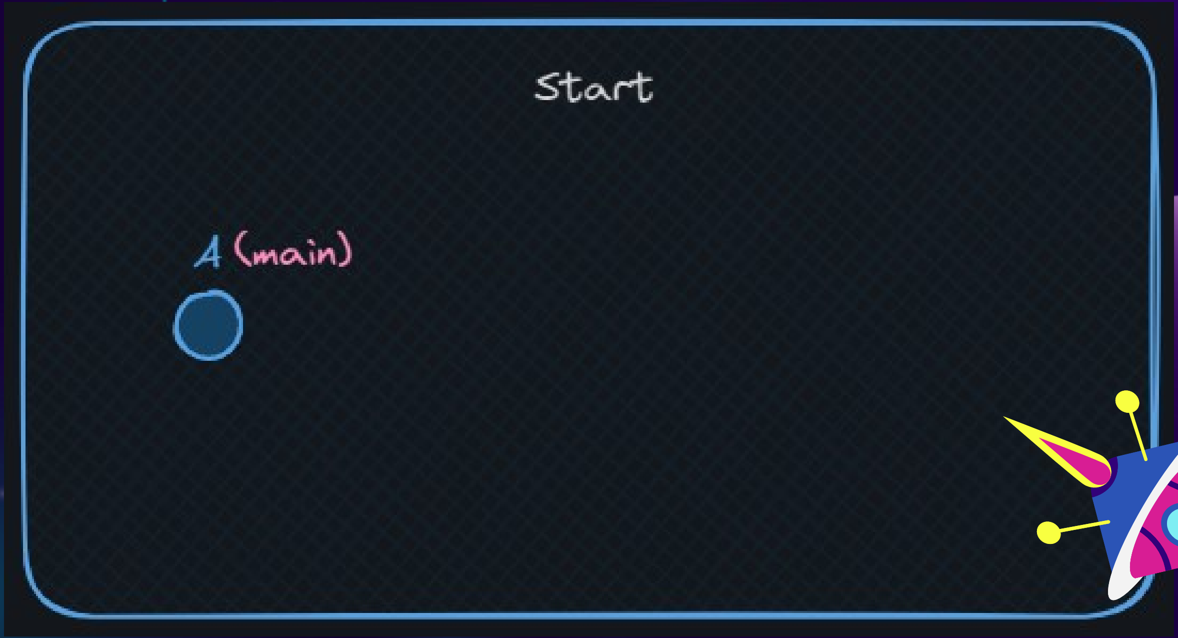
Code your way to succeed

Prepare your repo:

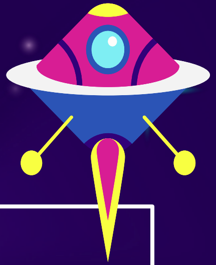
- Make a new dir called `git-branches`
- Make a repo in this dir
- Make a README.md with “A” in it and commit with message “A”'

Mission 0

Desired State



Mission 0



Great Job!

Mission 0 status: done

git-branches

```
mkdir git-branches
cd git-branches
git init
echo "A" >> README.md
git add .
git commit -m "A"
```

Creating branches

- To create a new branch:

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

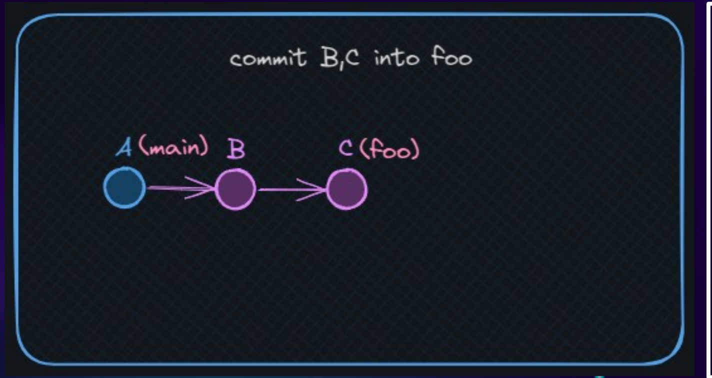
- To switch to a branch:

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

⚠️ Attention: after you create a branch git doesn't automatically switch to it

Mission 1

Desired State



git-branches

Code your way to succeed

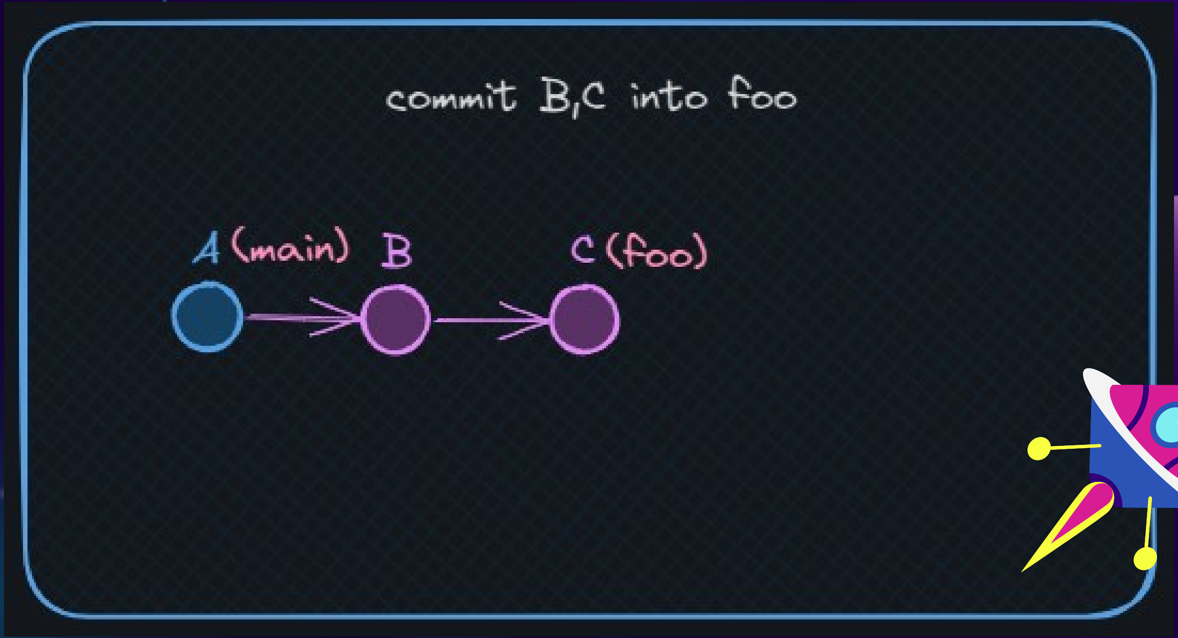
Your first branch:

- Create a branch “foo” and switch to it.
- find foo in .git
- Commit “B” and “C” into foo

Note: make the change the same as the commit message

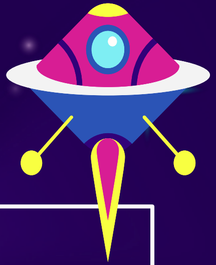
Mission 1

Desired State



$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \end{tikzcd}$$

Mission 1



Great Job! Mission 1 status: done

git-branches

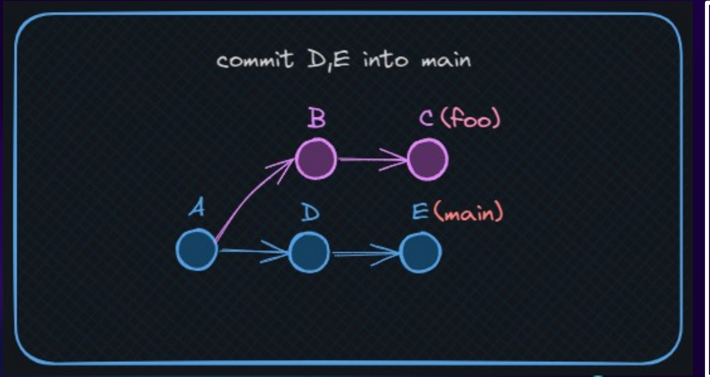
```
git branch foo
git checkout foo
find .git

# refs/heads/foo
```

```
echo "B" >> README.md
git add .
git commit -m "B“
echo "C" >> README.md
git add .
git commit -m "C"
```

Mission 2

Desired State



git-branches

Code your way to succeed

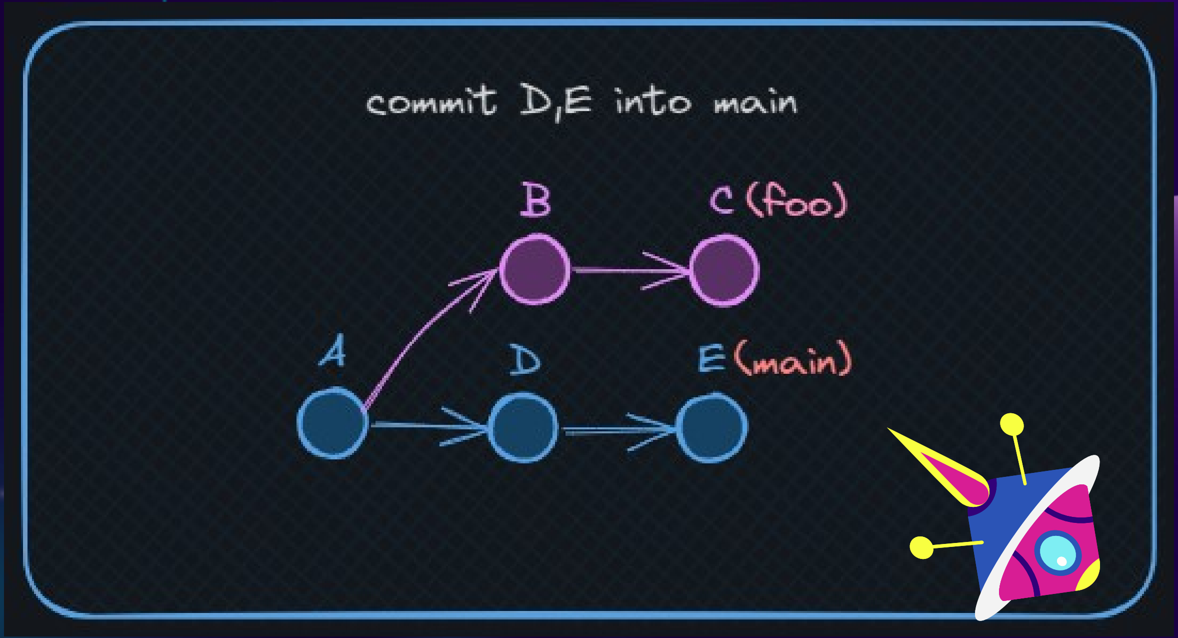
Things happen on main, you know:

- Switch to main
- commit "D" and "E" into a new file second.md

NOTE: make sure to commit in a new file
We don't want conflicts yet 😉

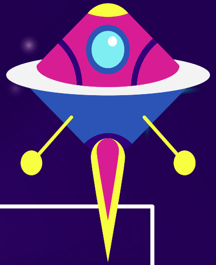
Mission 2

Desired State



$$\begin{tikzcd} & B \arrow[r] & C \\ A \arrow[ur] \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

Mission 2



Great Job!

Mission 2 status: done

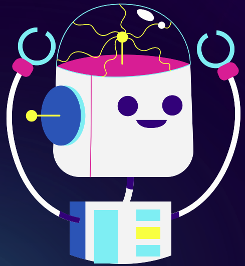
git-branches

```
git checkout main
echo "D" >> second.md
git add .
git commit -m "D"
```

```
echo "E" >> second.md
git add .
git commit -m "E“
git log --graph
--oneline --parents
```

Combining Your Work:

In git, there are 2 main ways to join branches:



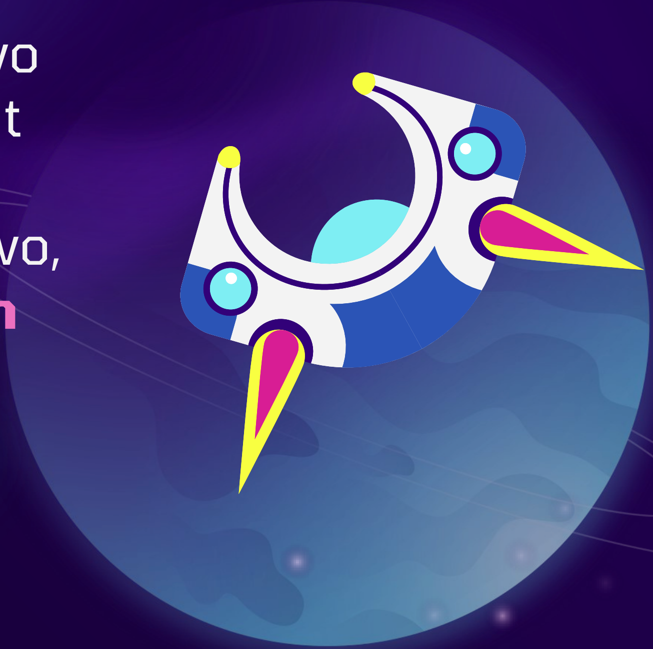
merge

rebase

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**"

- The docs



merge

To merge a branch into the currently checked branch:

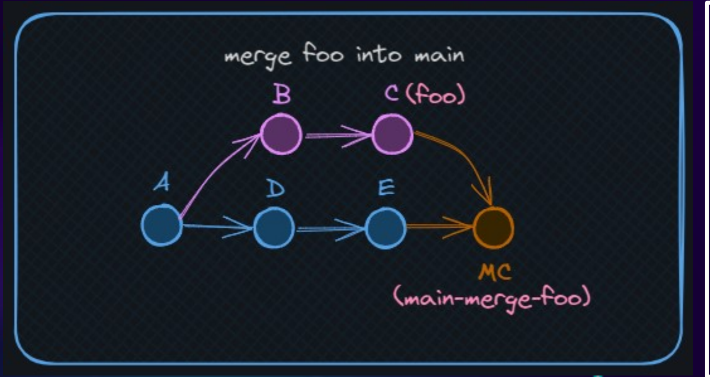
```
git merge <source-name>
```

Merge has 2 outcomes depending on the state, more on that later.

⚠️ NOTE: the target-branch is the currently checked branch

Mission 3

Desired State



git-branches

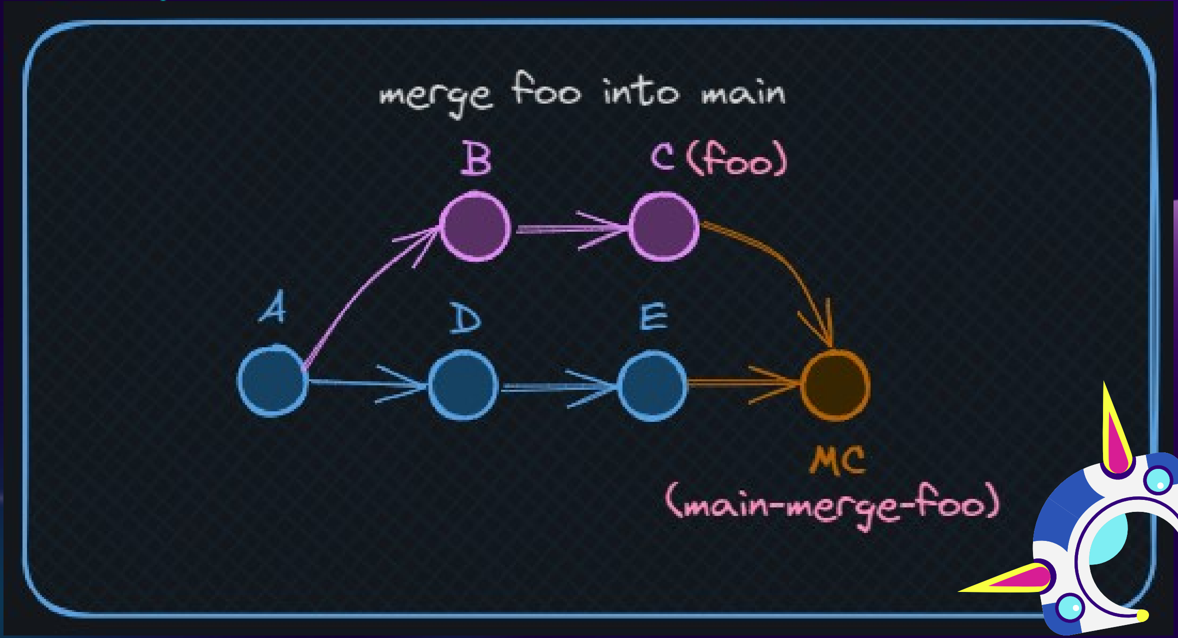
Code your way to succeed

Let’s merge:

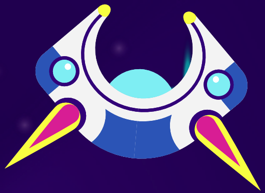
- Create a new branch “main-merge-foo” of main.
- Merge foo
- Look at the graph

Mission 3

Desired State



Mission 3



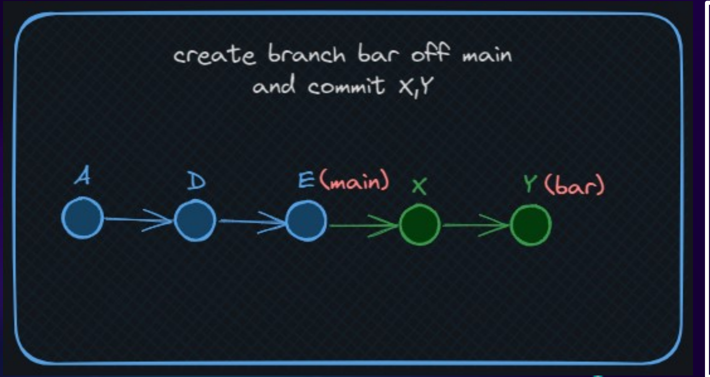
Great Job! Mission 3 status: done

git-branches

```
git checkout main
git checkout -b main-merge-foo
git merge foo
git log --graph --oneline --parents
```

Mission 4

Desired State



git-branches

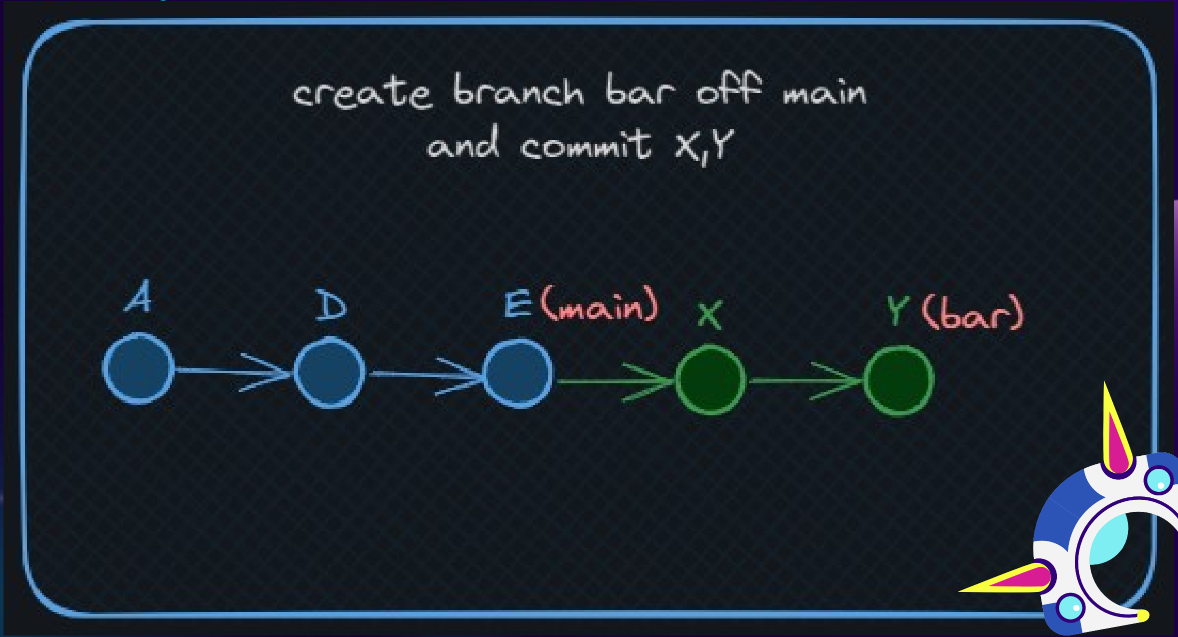
Code your way to succeed

Let's Fast-Forward merge:

- Make a new branch "bar" off main.
- Commit "X" and "Y" in bar into bar.md
- Merge bar into main

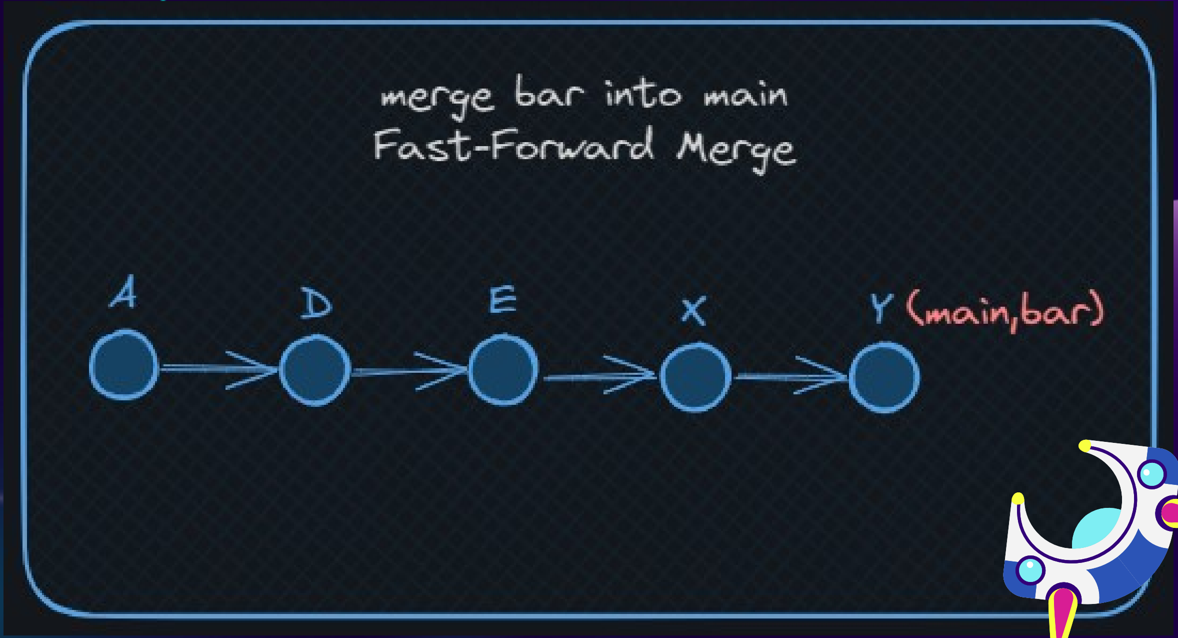
Mission 4

Desired State



Mission 4

Desired State



$$\begin{tikzcd} A \arrow[r] & D \arrow[r] & E \arrow[r] & X \arrow[r] & Y \end{tikzcd}$$

Mission 4

Great Job!

Mission 4 status: done

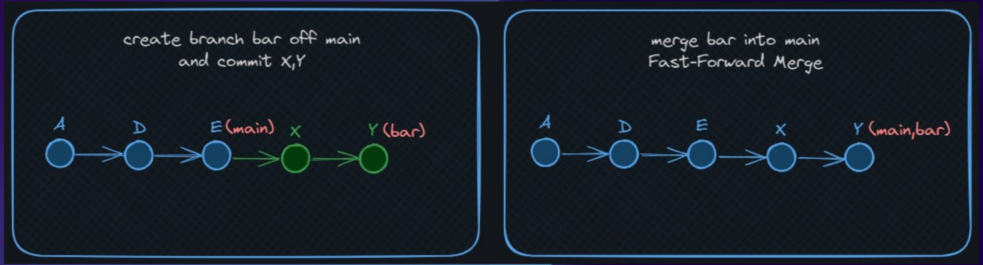
git-branches

```
git checkout main
git checkout -b bar
echo "X" >> bar.md
git add .
git commit -m "X“
```

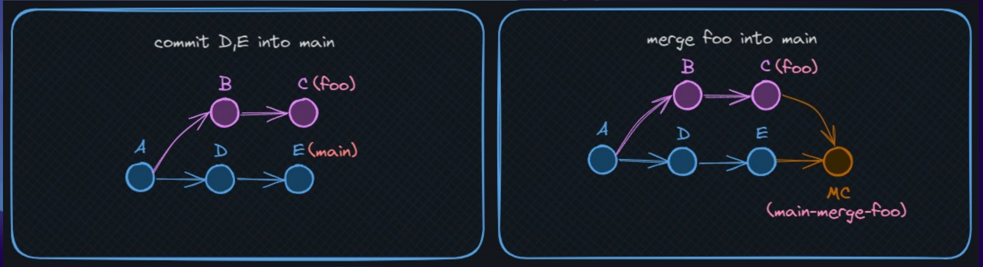
```
echo "Y" >> bar.md
git add .
git commit -m "Y“
git checkout main
git merge bar
```

Merge Outcomes:

- **Fast Forward Merge:**
  just update the pointer/reference (no merge commits)



- **Divergence Merge:**
  create a merge commit to combine 2 commits/histories have 2 parents



rebase

"git-rebase - `**Reapply**` commits on top of another base tip"
- "the docs"

- **To rebase a branch:**

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

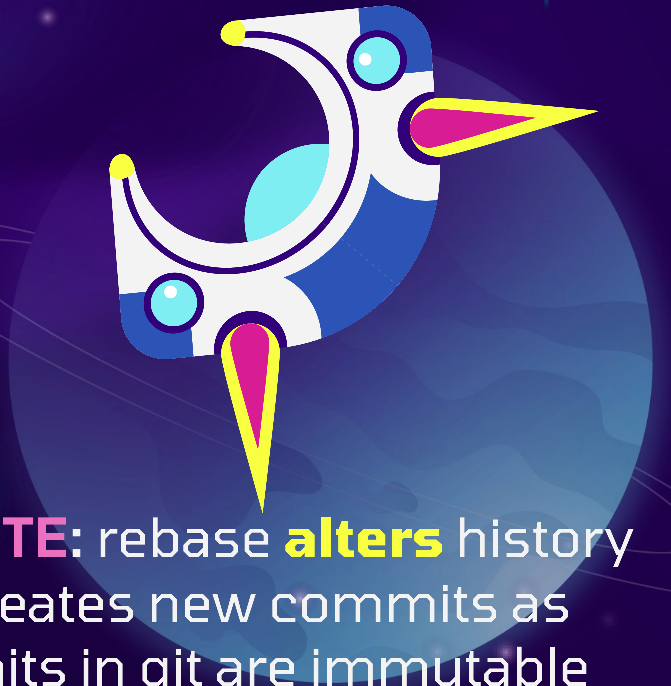
⚠️ **NOTE:** rebase is often used at your private feature branch, so target-branch is often **"main"**

rebase

- **How rebase works:**

1. checkout the latest commit at `<target-branch>`
2. replay one commit at a time of the `<source-branch>`
3. update source branch ref to the latest commit made.

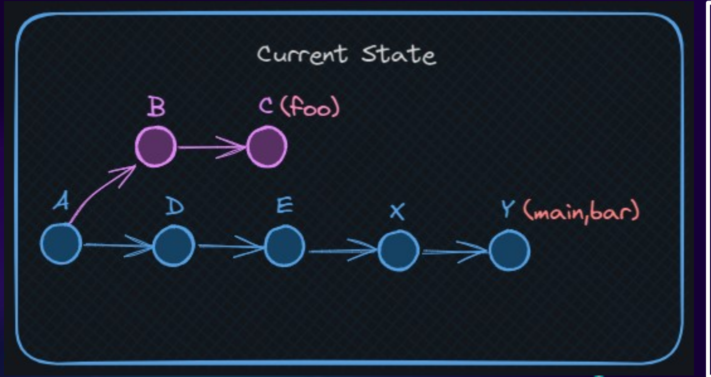
Rebase doesn't create merge-commits.



⚠️ **NOTE**: rebase **alters** history and creates new commits as commits in git are immutable

Mission 5: last mission in this chapter

Current State



git-branches

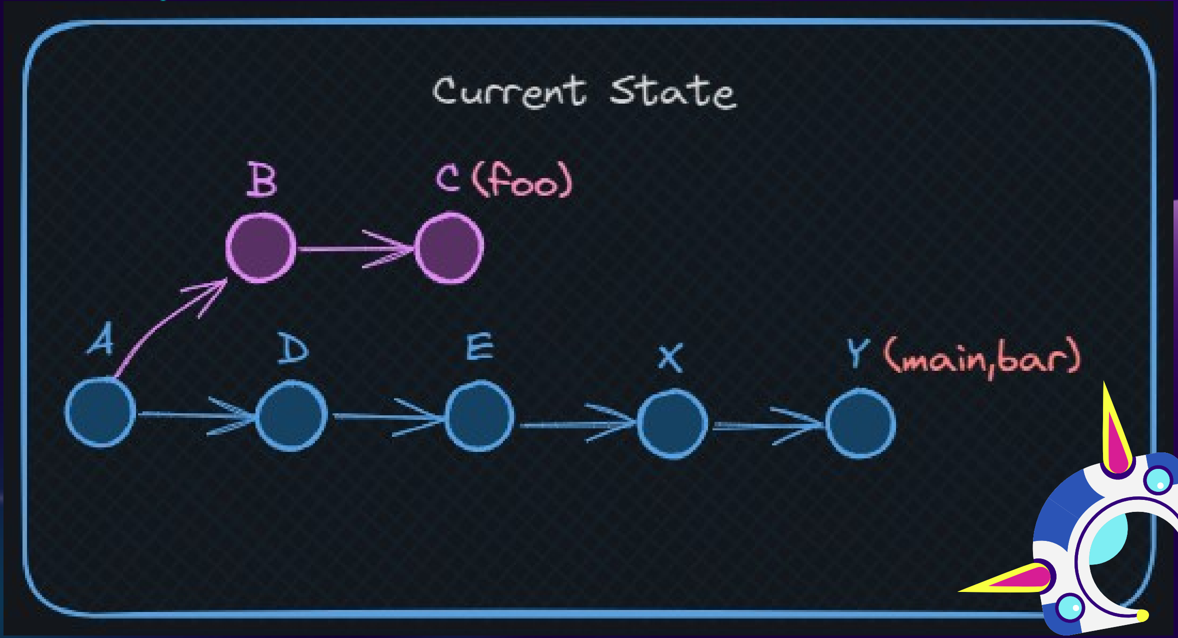
Code your way to succeed

Rebase Your feature branch:

- Checkout foo
- Rebase off main

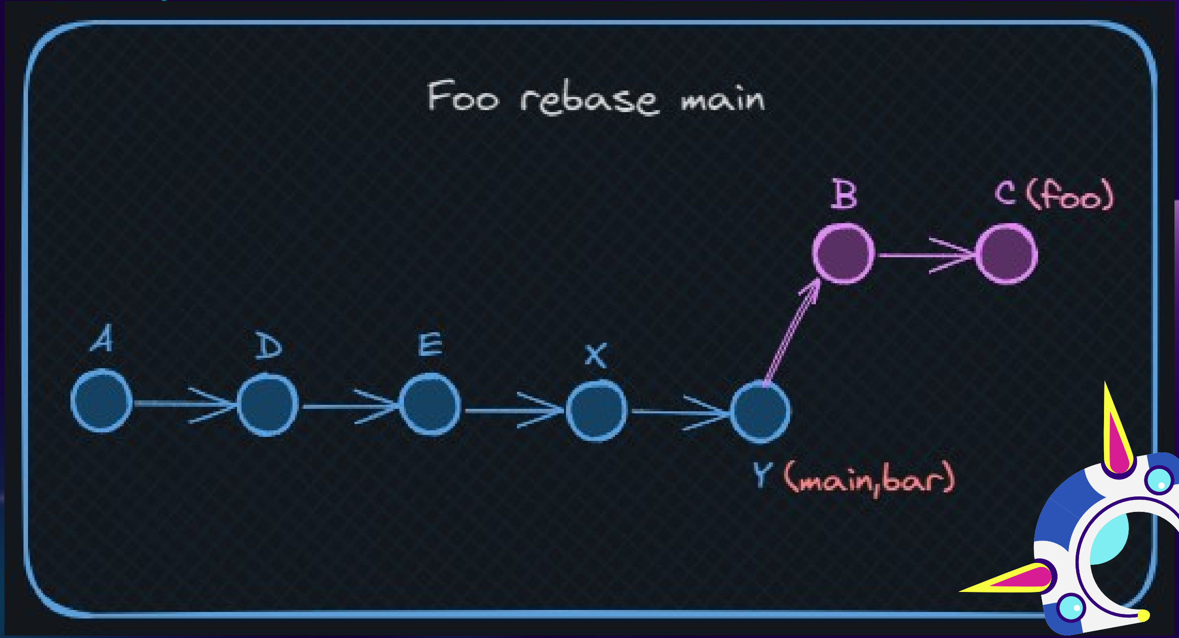
Mission 5

Current State



Mission 5

Desired State



Mission 5: last mission in this chapter

Great Job!

Mission 4 status: done

git-branches

```
git checkout foo
git rebase main
git log --graph --oneline
```

Merge VS. Rebase

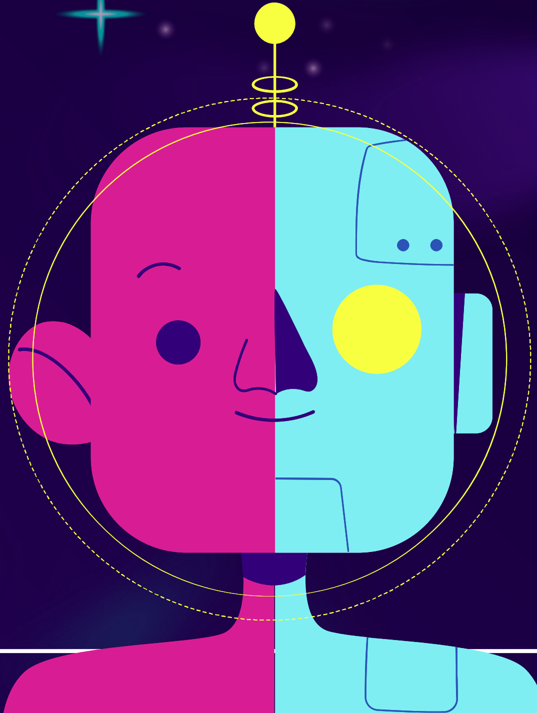
Merge

doesn't **alter** history 👍

doesn't require **push force** 👍

works with **private** and 👍 ‘**public** branches

makes annoying merge 👎 commits



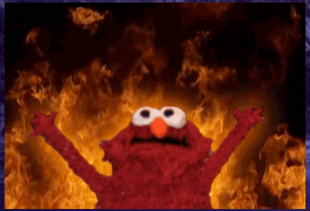
Rebase

👍 no annoying merge commits

👍 linear history which is easier to search

👎 alters history

👎 requires `push force`



git me
PROGR
ES
VS
AMPLE
Squash and merge
You can also open
Merge pull request
You can also open

Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

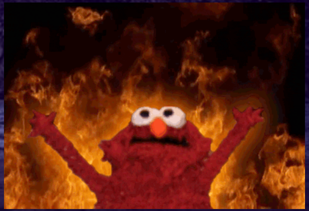
✓ Squash and merge
The 1 commit from this branch will be added to the base branch.

Rebase and merge
The 1 commit from this branch will be rebased and added to the base branch.
Not enabled for this repository

✓ Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge
The 1 commit from this branch will be added to the base branch.

Rebase and merge
The 1 commit from this branch will be rebased and added to the base branch.
Not enabled for this repository



Workflows Wars

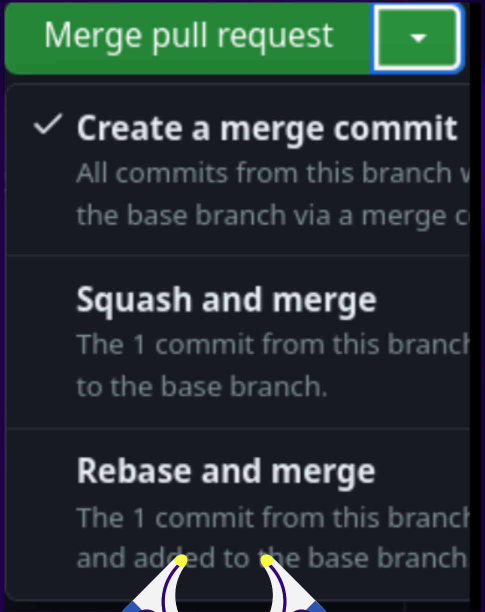
Workflow wars

Merge Workflows
(just merge)

ℹ️ merge back into main

👎 annoying merge commits

⚠️ **NOTE1:** people are always so **opinionated** about which workflow is better, but **rebase** is diffidently the **best** 😉



✓ Create a merge commit
All commits from this branch w
the base branch via a merge c

Squash and merge
The 1 commit from this branch
to the base branch.

Rebase and merge
The 1 commit from this branch
and added to the base branch

Rebase flow
(rebase in private - FF merge in public)

ℹ️ rebase main into your branch first, then fast forward merge into main

⚠️ **NOTE2:** the difference between the 2 flows in simple form compiles to which option you choose in GitHub pull request.