What is version control?

- Keeps track of changes to your code.
- You don't have to worry about accidentally losing or deleting code.
- You can experiment and reset to a known good state.
- Makes collaborating with others easier.

What is version control?

- Keeps track of changes to your code.
- You don't have to worry about accidentally losing or deleting code.
- You can experiment and reset to a known good state.
- Makes collaborating with others easier.

What is git?

- 'the stupid content tracker'
- Distributed - everything is kept on your local machine.
- 'Repository' - a collection of code and history.
- 'Commit' - a chunk of saved changes.

Getting Started

- `git init` Makes a new empty git repository.
- `git add FILE` Adds changes in `FILE` to the next commit.
- `git status` Shows the status of the repository.

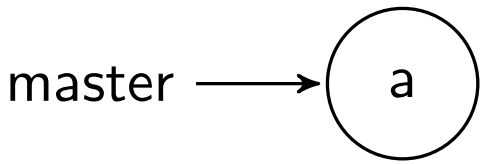
Getting Started

- `git init` Makes a new empty git repository.
- `git add FILE` Adds changes in `FILE` to the next commit.
- `git status` Shows the status of the repository.

- `git config --global user.name NAME`
- `git config --global user.email EMAIL`

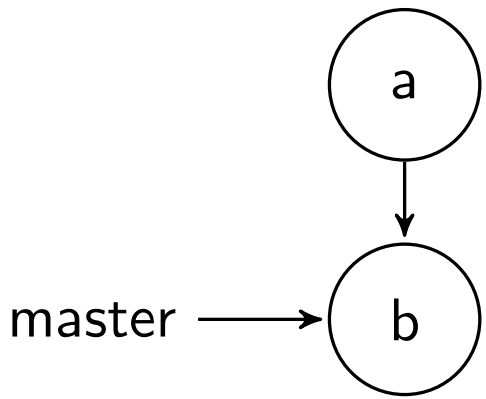
Committing

```
git commit -m ‘‘a’’
```



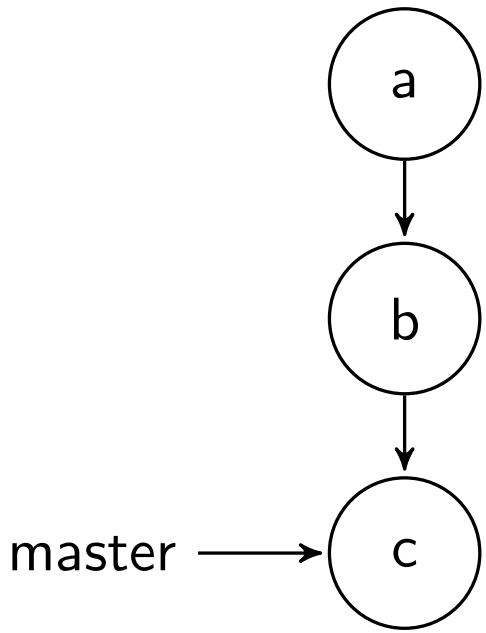
Committing

```
git commit -m ‘‘b’’
```



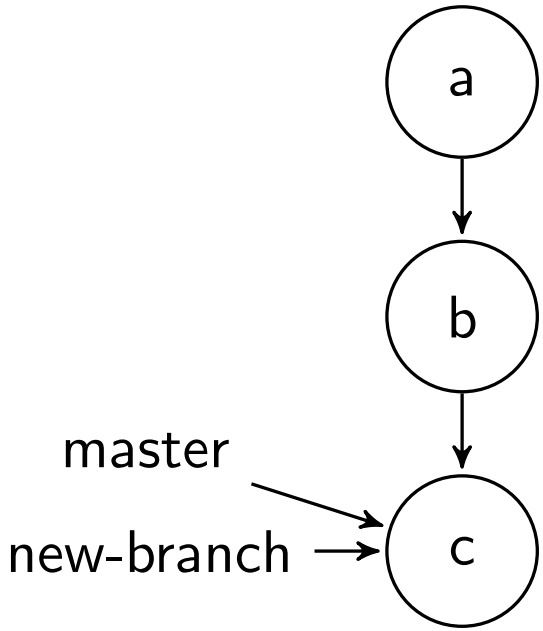
Committing

```
git commit -m ‘‘c’’
```



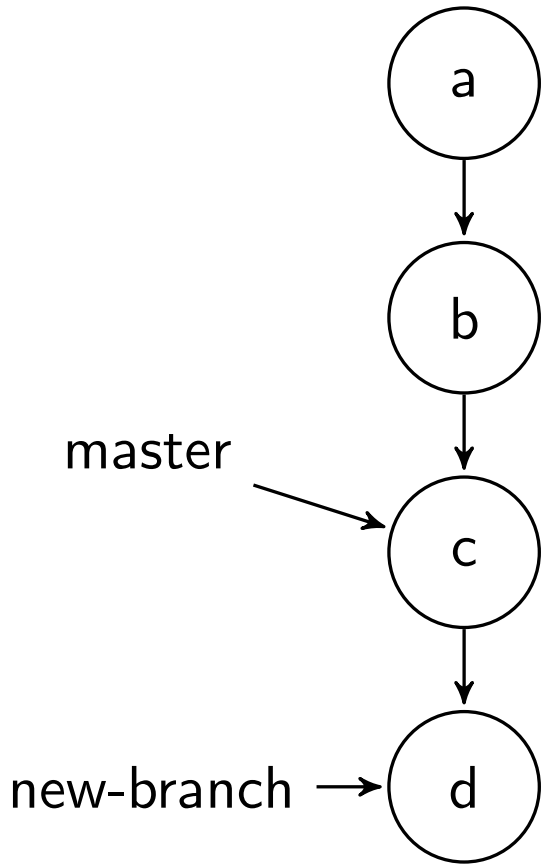
Branching

```
git checkout -b new-branch
```

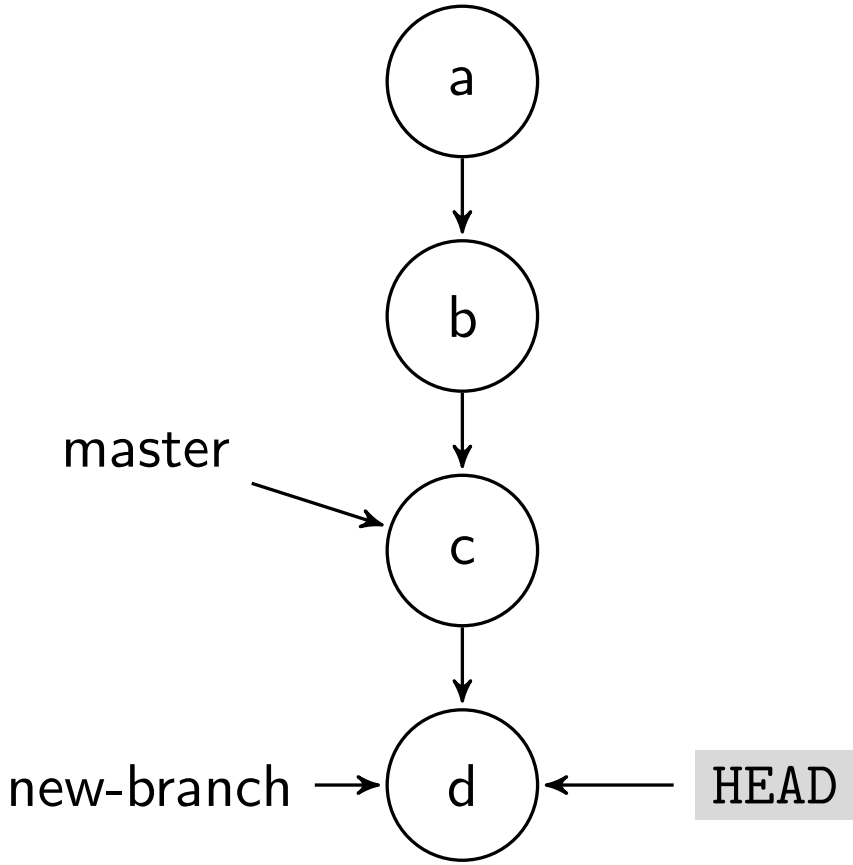


Branching

```
git commit -m ‘‘d’’
```

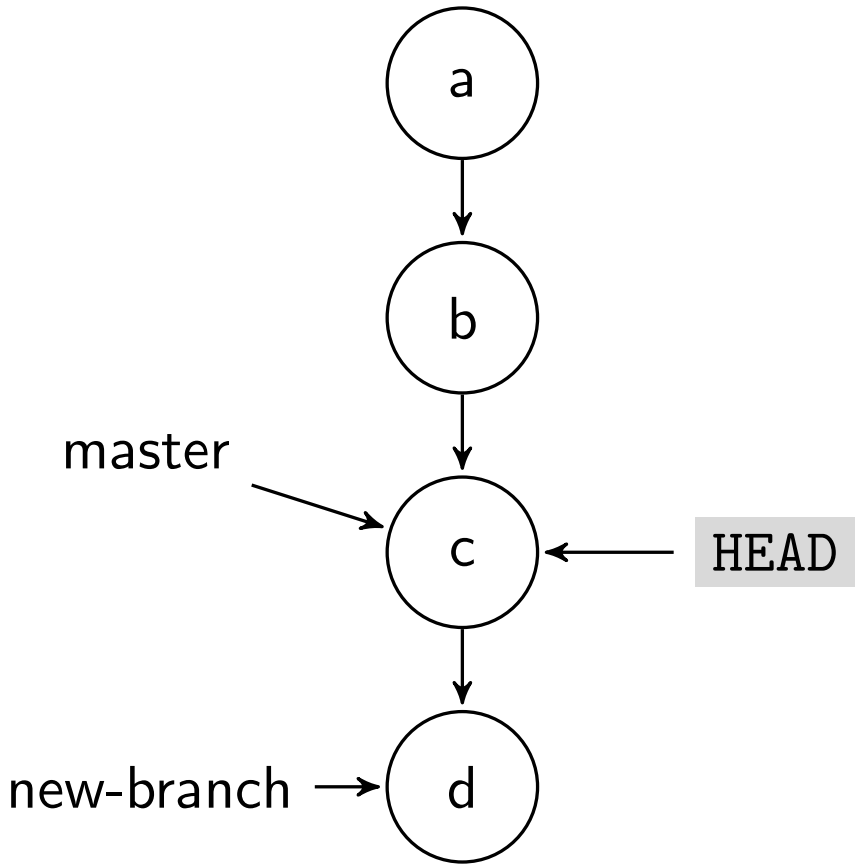


Branching



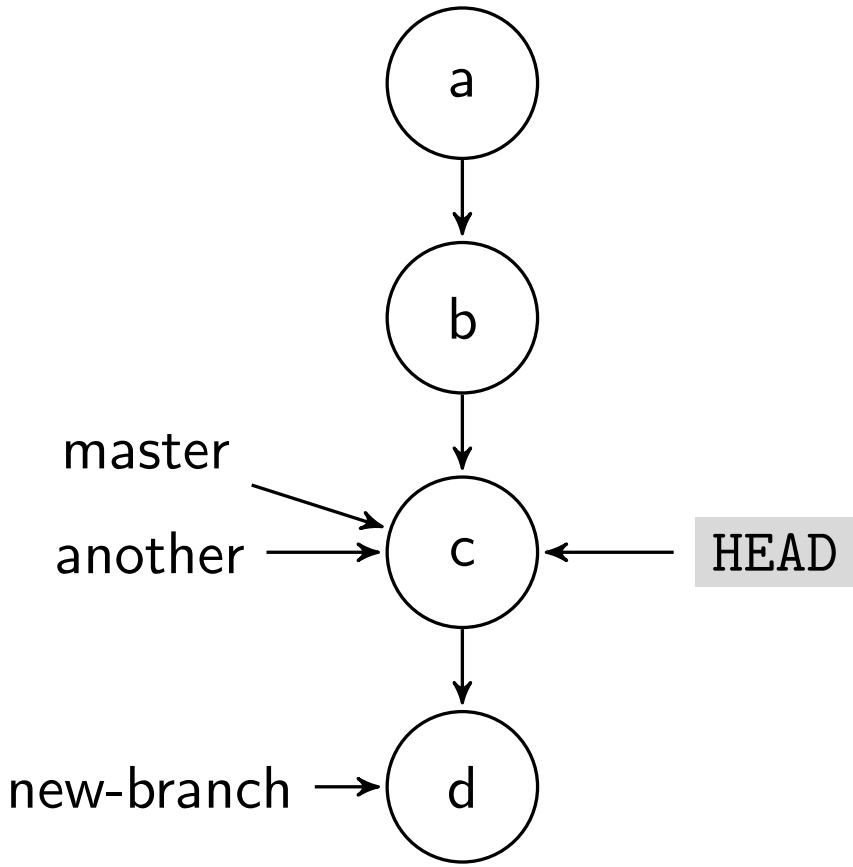
Branching

```
git checkout master
```



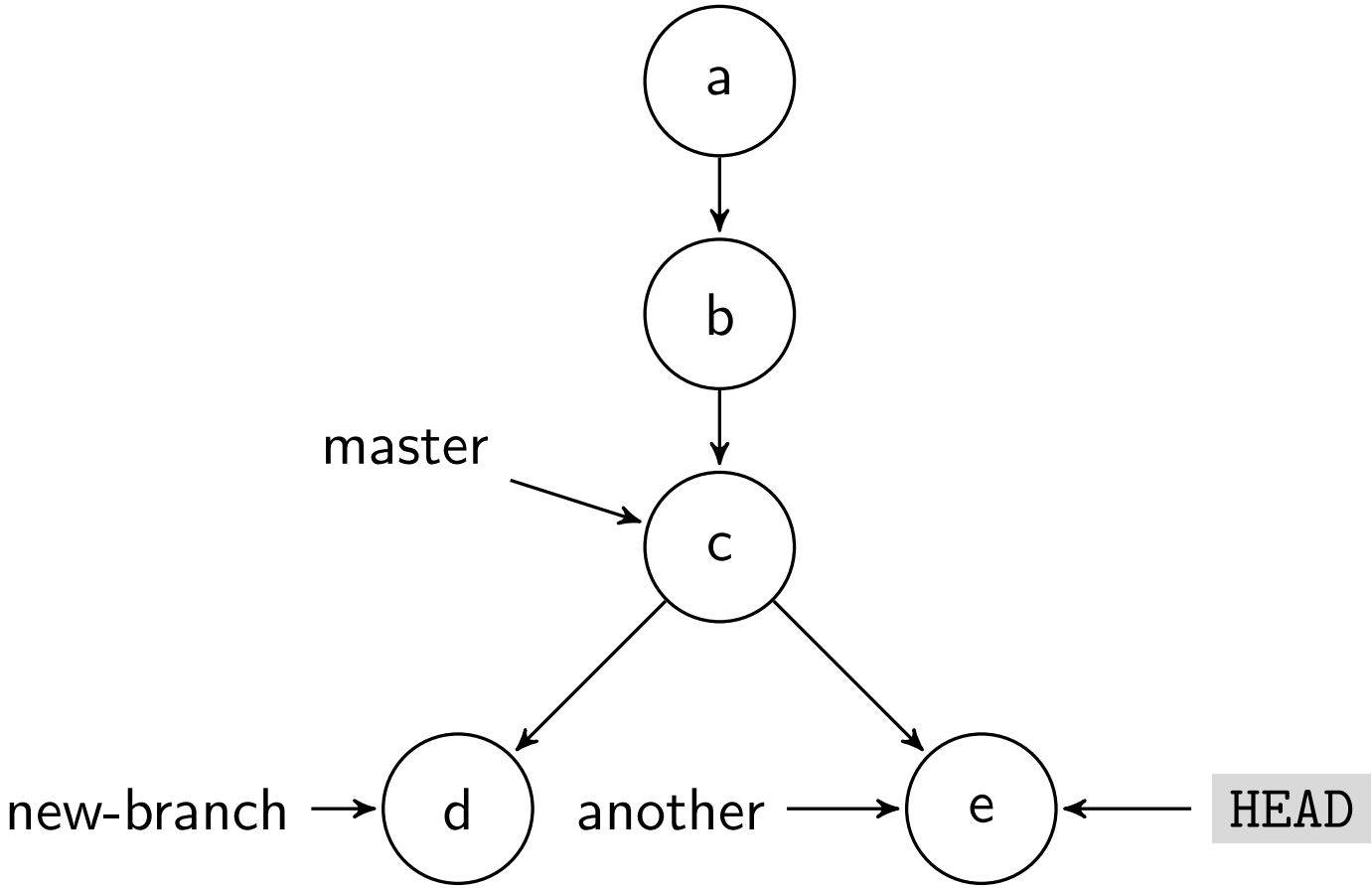
Branching

```
git checkout -b another
```



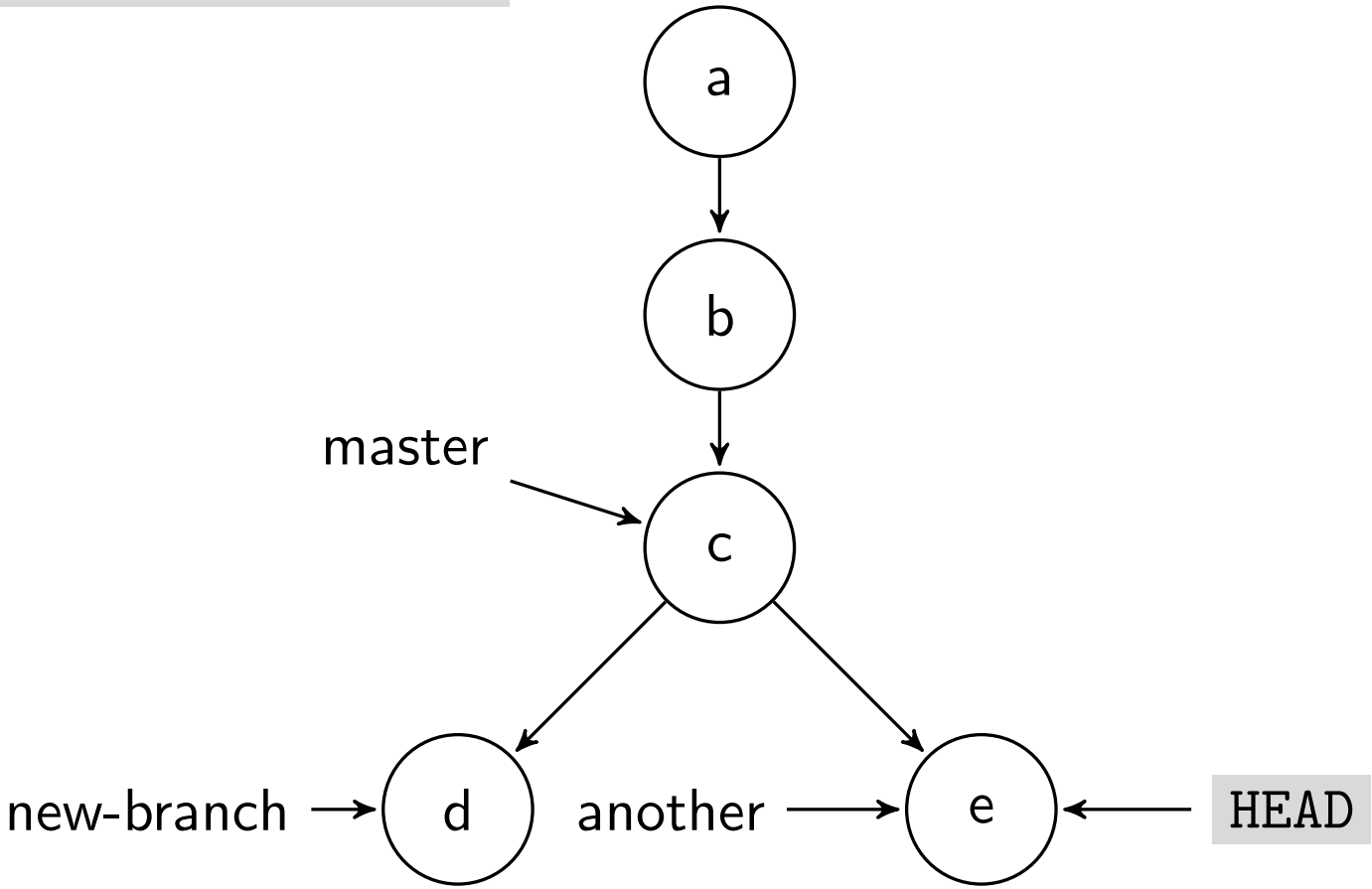
Branching

```
git commit -m ‘‘e’’
```



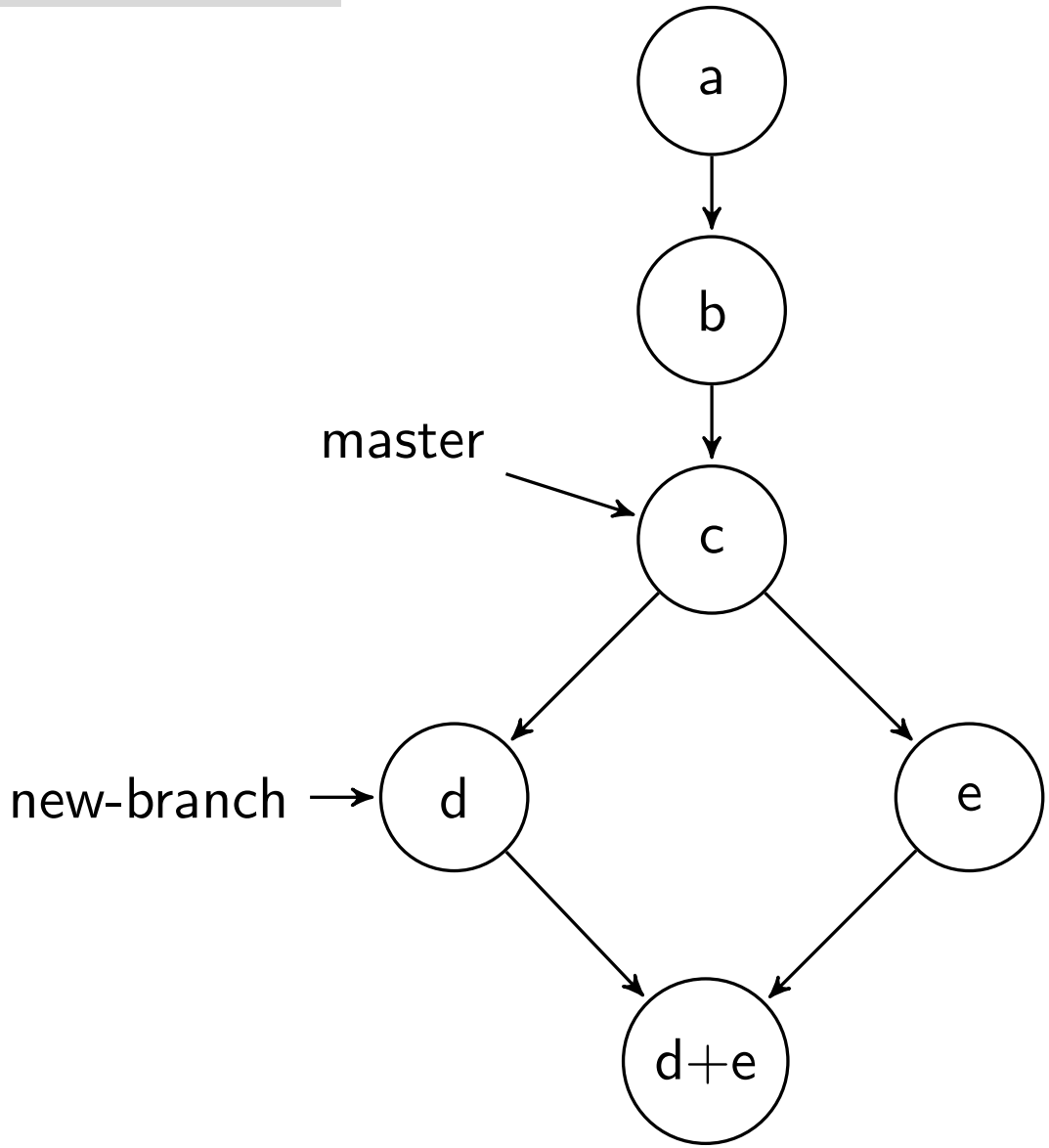
Merging

```
git merge new-branch
```



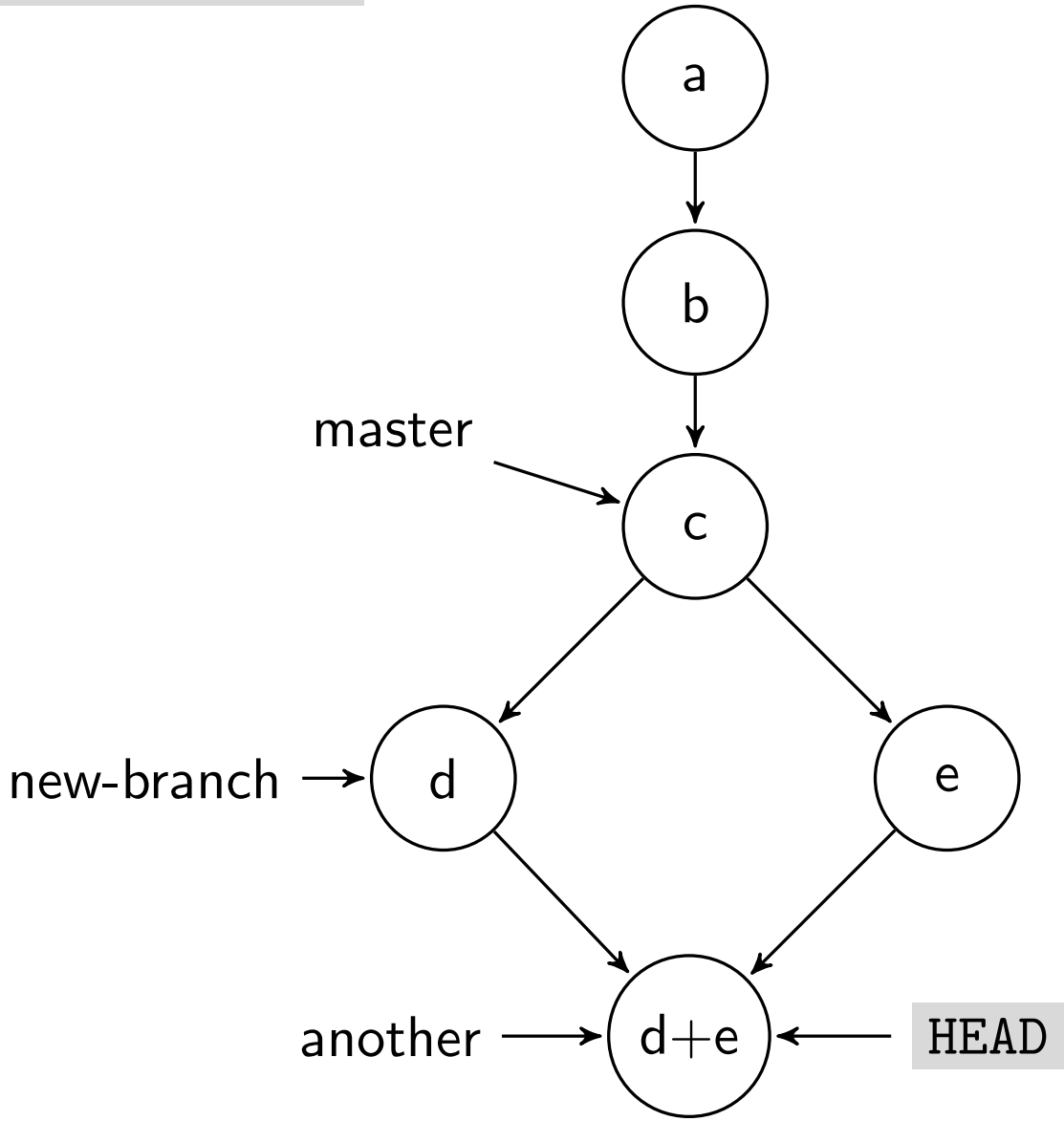
Merging

```
git merge new-branch
```



Merging

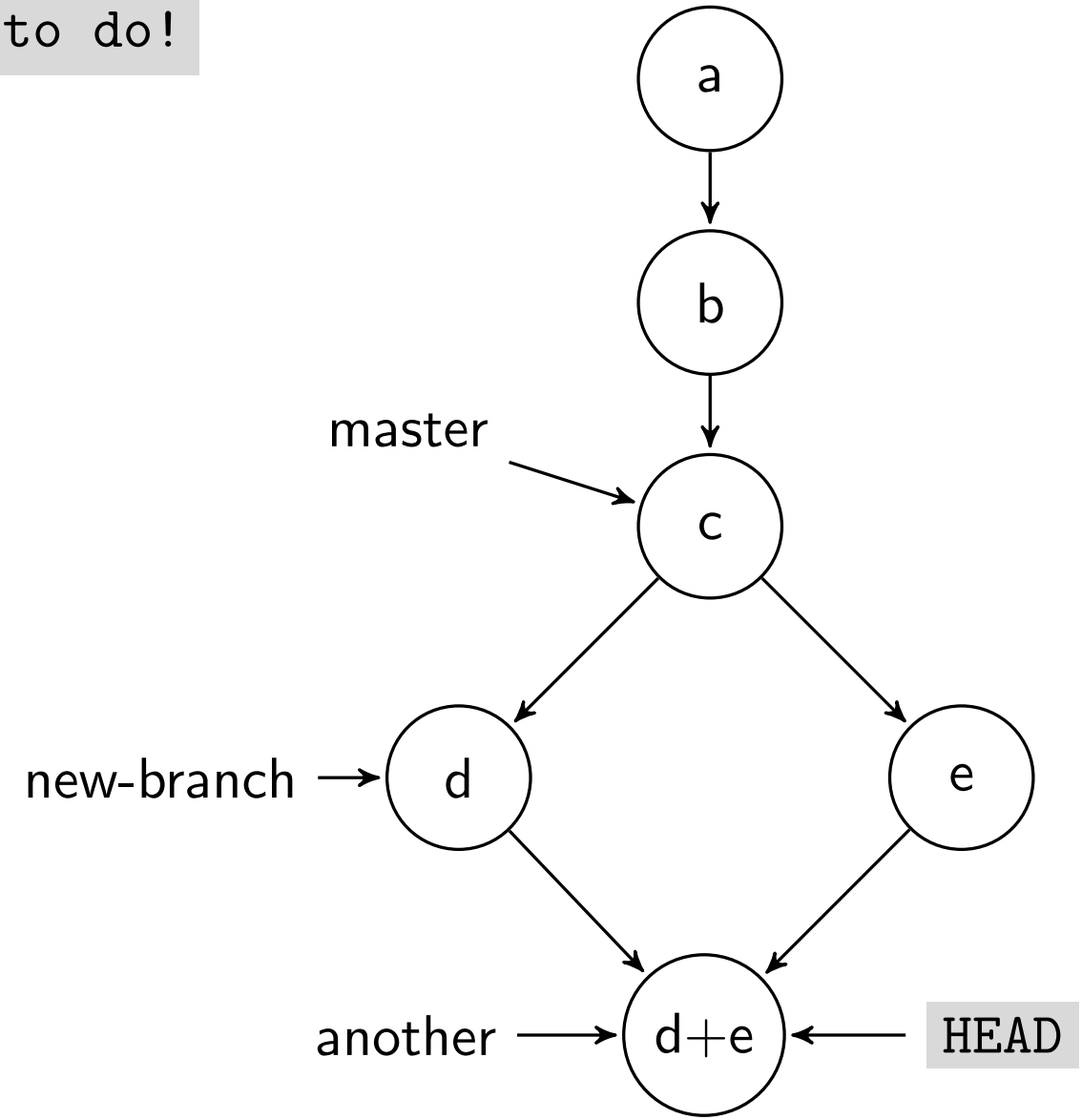
```
git merge new-branch
```



Merging

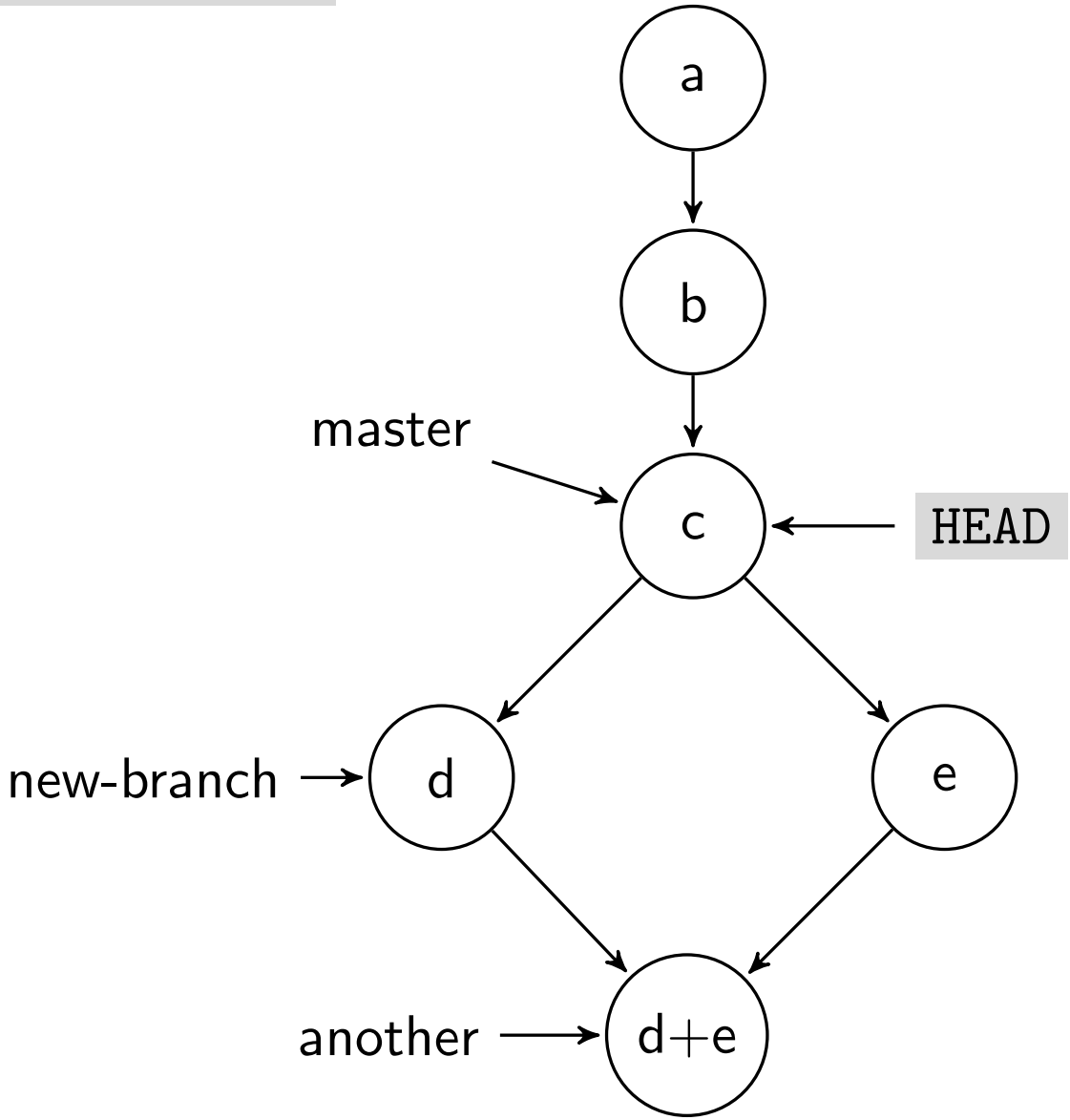
```
git merge master
```

```
Nothing to do!
```



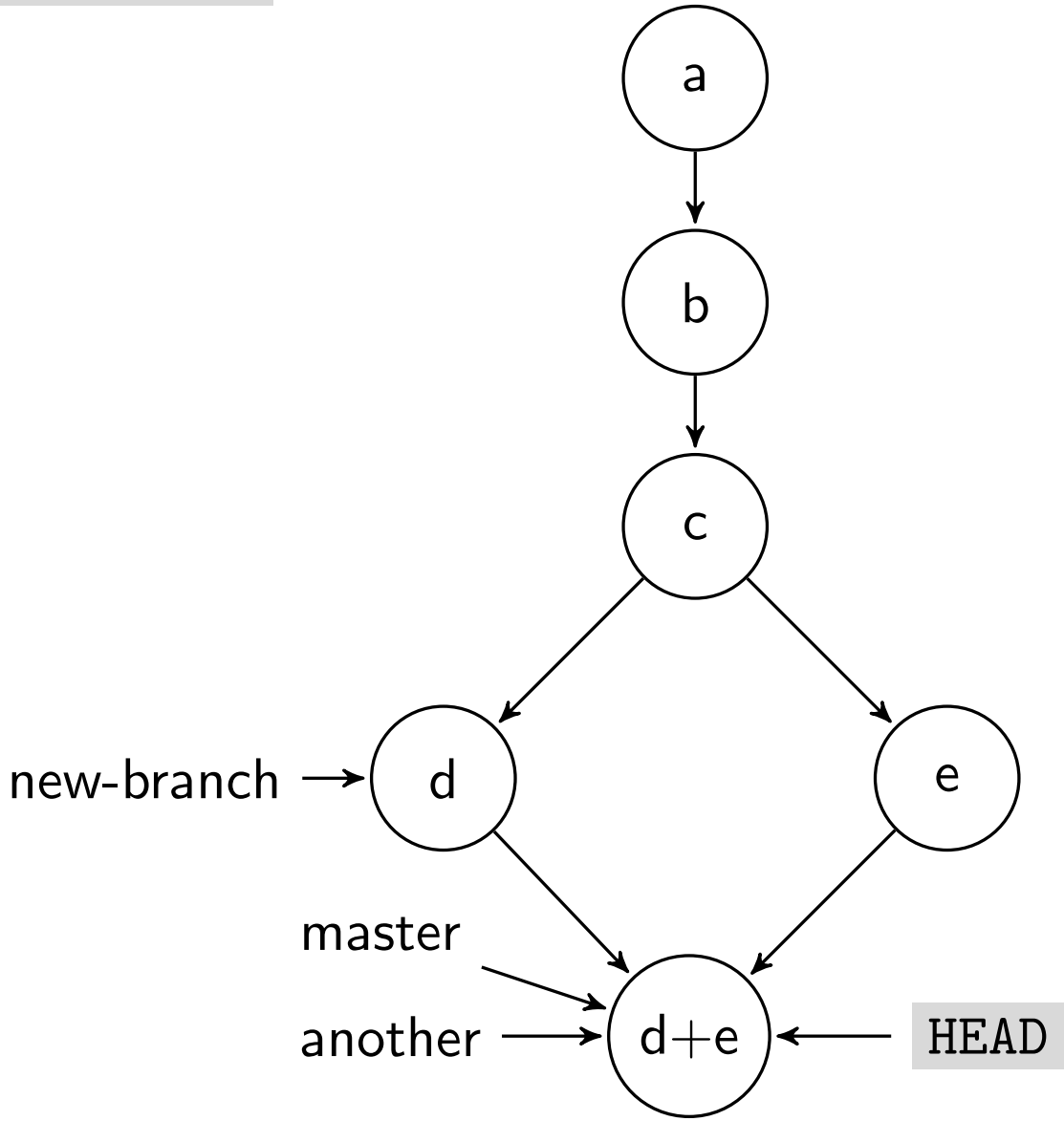
Merging

```
git checkout master
```



Merging

```
git merge another
```



Merge conflicts

```
CONFLICT (content):  Merge conflict in the-file.txt
Automatic merge failed; fix conflicts and then commit
the result.
```

Merge conflicts

```
CONFLICT (content): Merge conflict in the-file.txt
Automatic merge failed; fix conflicts and then commit
the result.
```

In `the-file.txt`:

```
<<<<<<< HEAD
The current branch's contents
=======
Stuff from the branch you're merging
>>>>>>> new-branch
```

Merge conflicts

```
CONFLICT (content): Merge conflict in the-file.txt
Automatic merge failed; fix conflicts and then commit
the result.
```

In `the-file.txt`:

```
<<<<<<< HEAD
The current branch's contents
=======
Stuff from the branch you're merging
>>>>>>> new-branch
```

`git add the-file.txt` and `git commit`

Looking at stuff

- `git log` Show a log of commits
  - `--graph` Neat ASCII graph
  - `-p` Show what changed in each commit

Looking at stuff

- `git log` Show a log of commits
  - `--graph` Neat ASCII graph
  - `-p` Show what changed in each commit
- `git diff` Show uncommitted changes

Looking at stuff

- `git log` Show a log of commits
  - `--graph` Neat ASCII graph
  - `-p` Show what changed in each commit
- `git diff` Show uncommitted changes
- `gitk` Graphical log
- `git gui` Graphical tool for committing

Working with remotes

- `git clone REPO_LOCATION` makes a copy of a repository.

Working with remotes

- `git clone REPO_LOCATION` makes a copy of a repository.
- `git push` Pushes changes from your current branch to the remote branch it tracks.
- `git pull` Pulls changes from the remote branch and merges them into your current branch.

Working with remotes

- `git clone REPO_LOCATION` makes a copy of a repository.
- `git push` Pushes changes from your current branch to the remote branch it tracks.
- `git pull` Pulls changes from the remote branch and merges them into your current branch.
- `git remote add REMOTE_NAME REPO_LOCATION` adds a remote to an existing repository.

Git Tips

- Make your commit messages descriptive!

Git Tips

- Make your commit messages descriptive!
- Don't add generated files (like `a.out`) to your repo.
- You can ignore certain files by putting their names in a `.gitignore` file in your repo.

Git Tips

- Make your commit messages descriptive!
- Don't add generated files (like `a.out`) to your repo.
- You can ignore certain files by putting their names in a `.gitignore` file in your repo.
- When collaborating, work on separate branches and merge as you go along.