

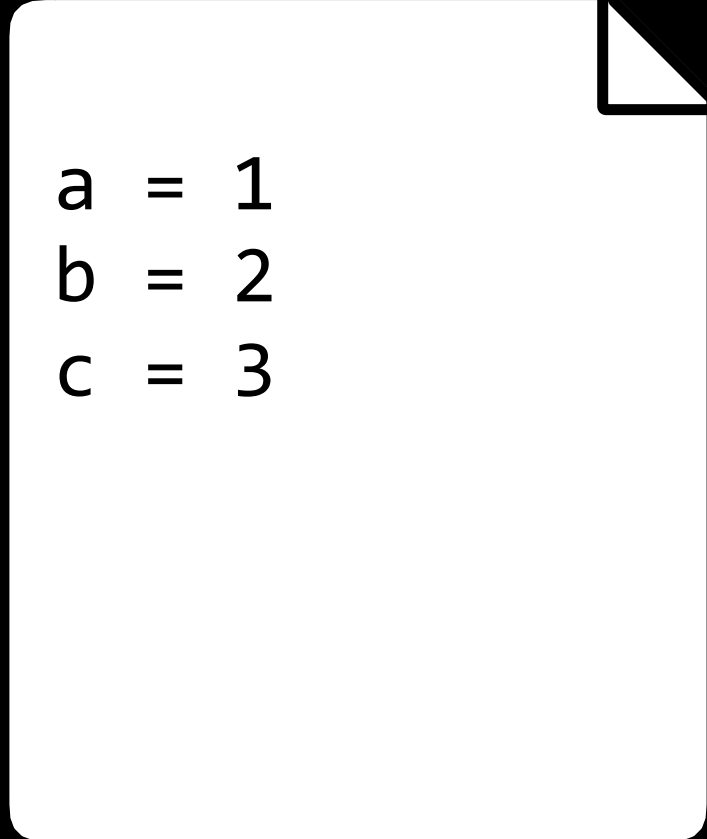
# Git

# git

git (engl. Slang für Blödmann)

“The joke ‘I name all my projects for myself, first Linux, then git’ was just too good to pass up. But it is also short, easy-to-say, and type on a standard keyboard. And reasonably unique and not any standard command, which is unusual.”

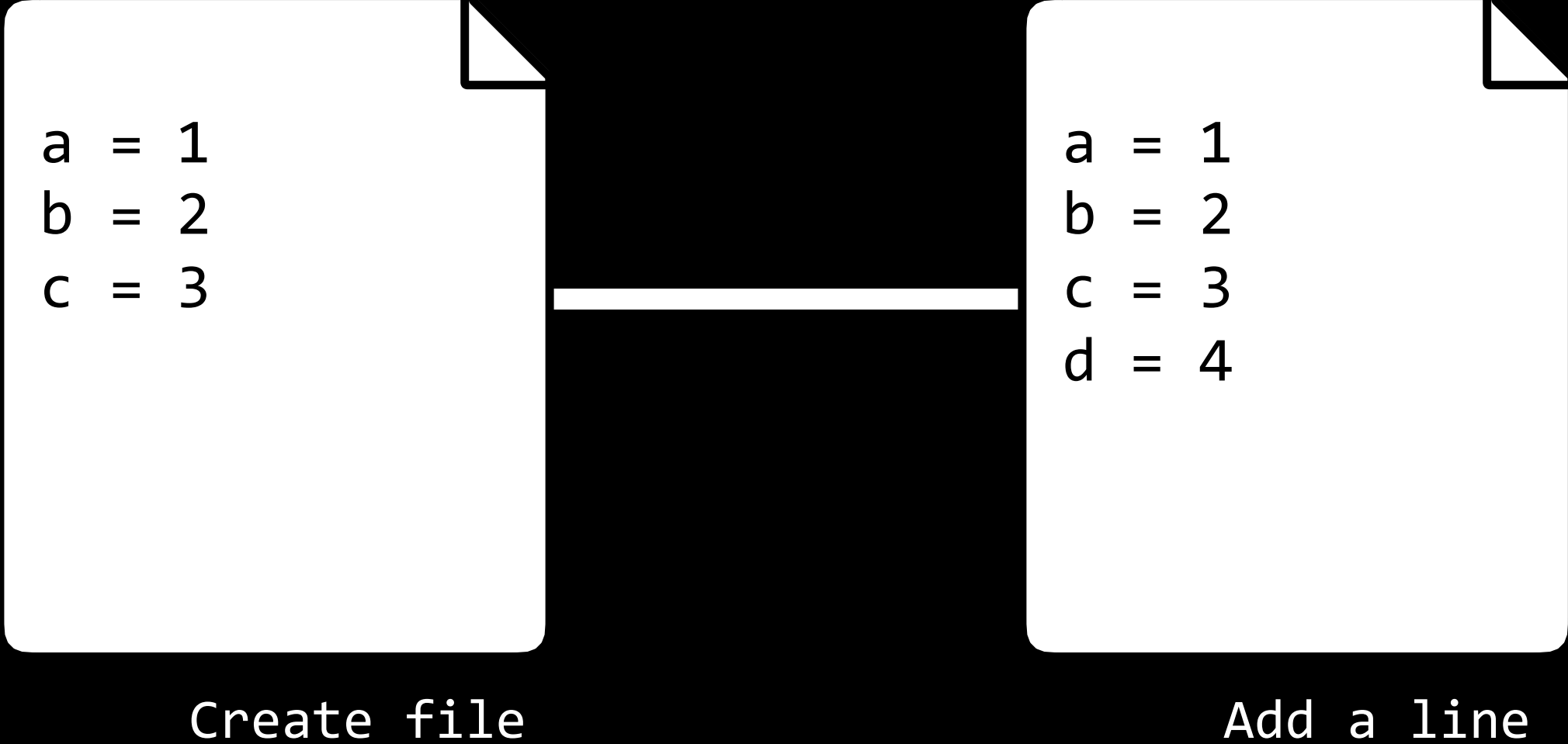
# Keep track of changes to code.



```
a = 1  
b = 2  
c = 3
```

Create file

# Keep track of changes to code.



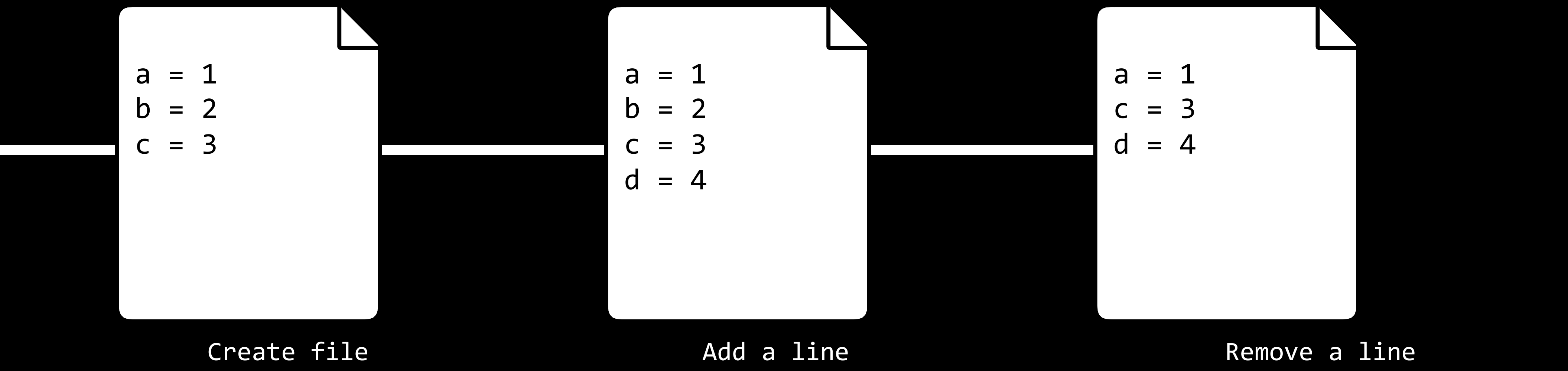
```
a = 1  
b = 2  
c = 3
```

Create file

```
a = 1  
b = 2  
c = 3  
d = 4
```

Add a line

# Keep track of changes to code.



```
a = 1  
b = 2  
c = 3
```

The diagram illustrates a sequence of three code files connected by horizontal arrows. The first file, labeled 'Create file', contains the code 'a = 1', 'b = 2', and 'c = 3'. The second file, labeled 'Add a line', contains the same code as the first but with an additional line 'd = 4'. The third file, labeled 'Remove a line', contains the code 'a = 1', 'c = 3', and 'd = 4', with the line 'b = 2' removed. Each file is represented as a white document icon with a folded top-right corner.

Create file

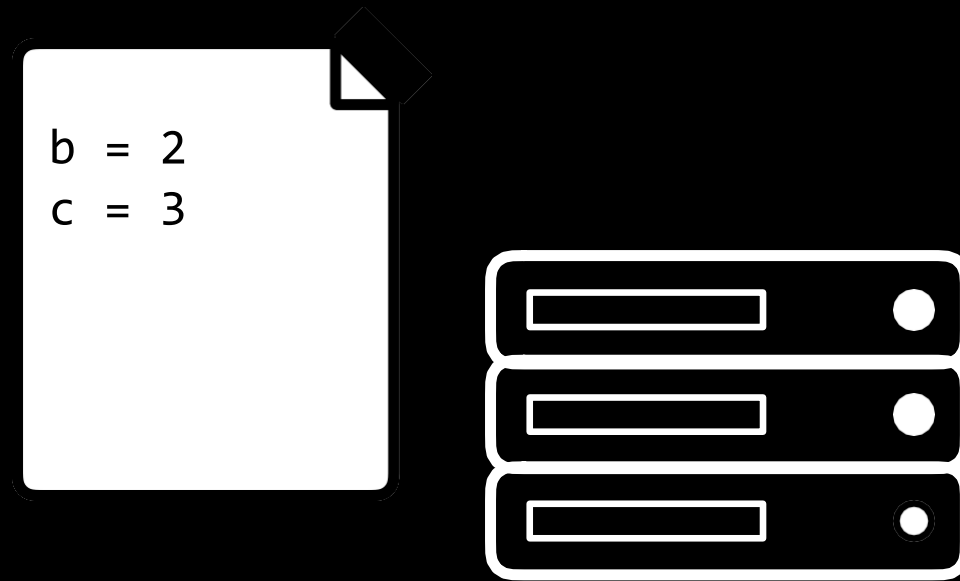
```
a = 1  
b = 2  
c = 3  
d = 4
```

Add a line

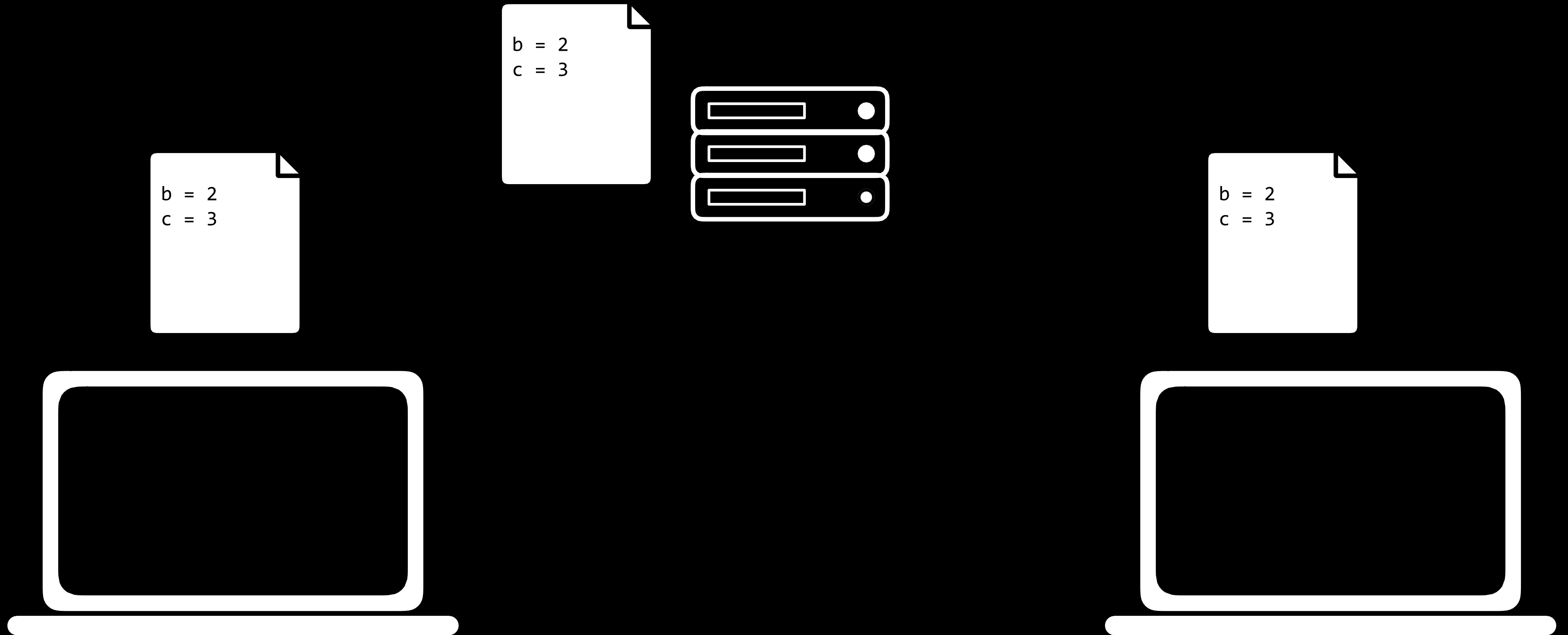
```
a = 1  
c = 3  
d = 4
```

Remove a line

# Synchronizes code between different people.

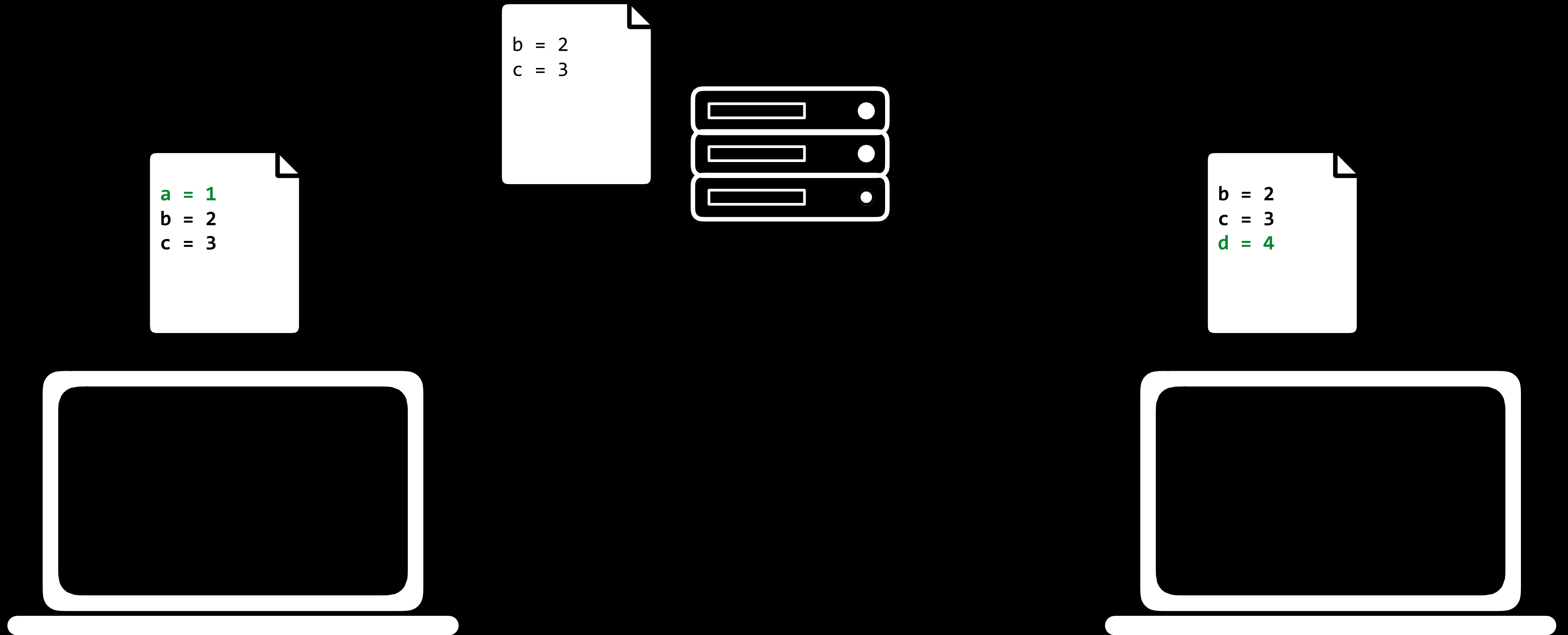


# Synchronizes code between different people.

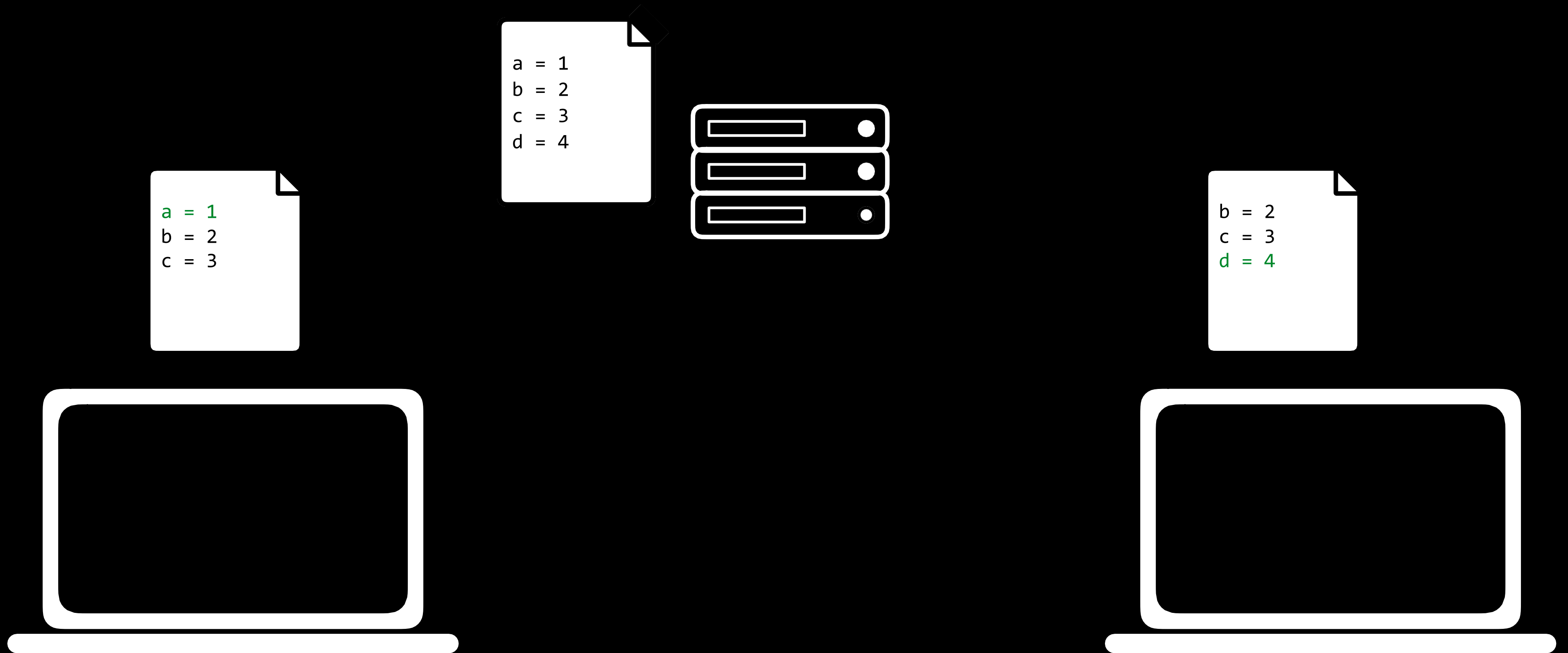




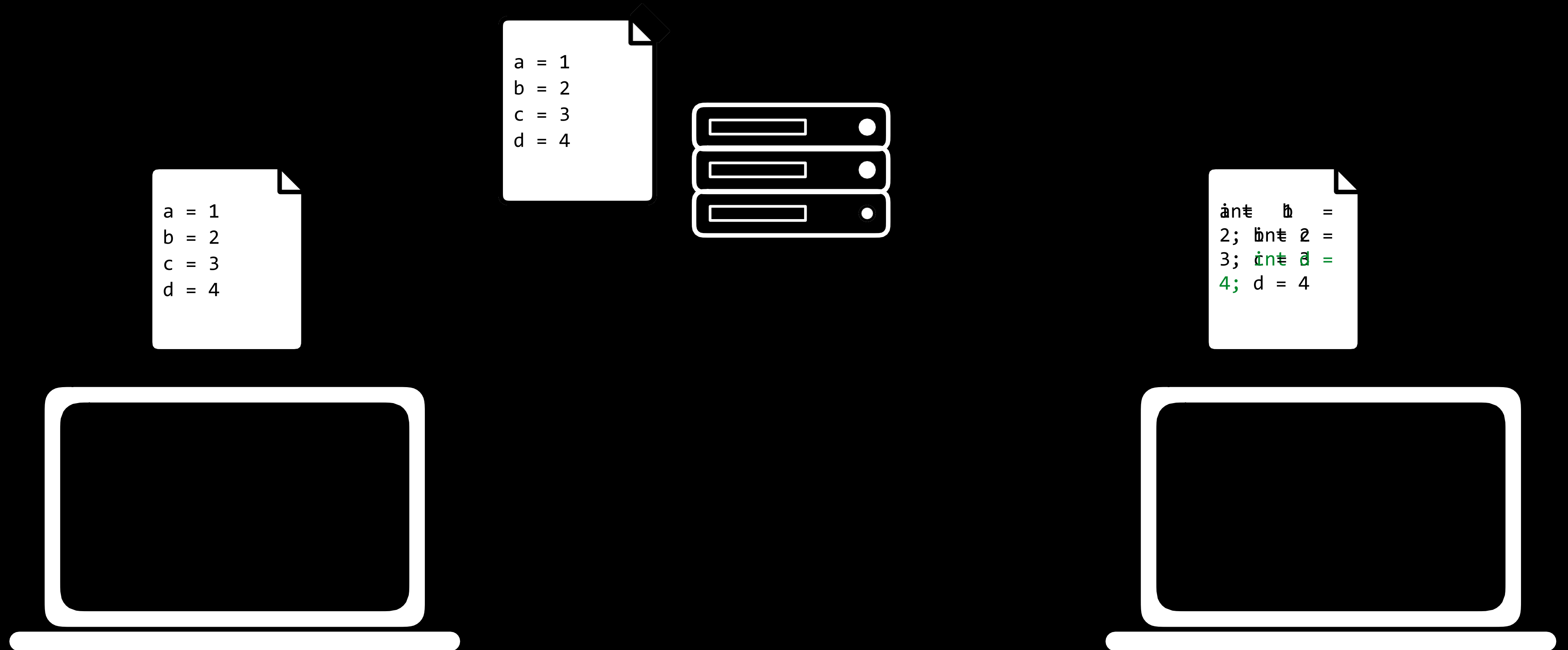
# Synchronizes code between different people.

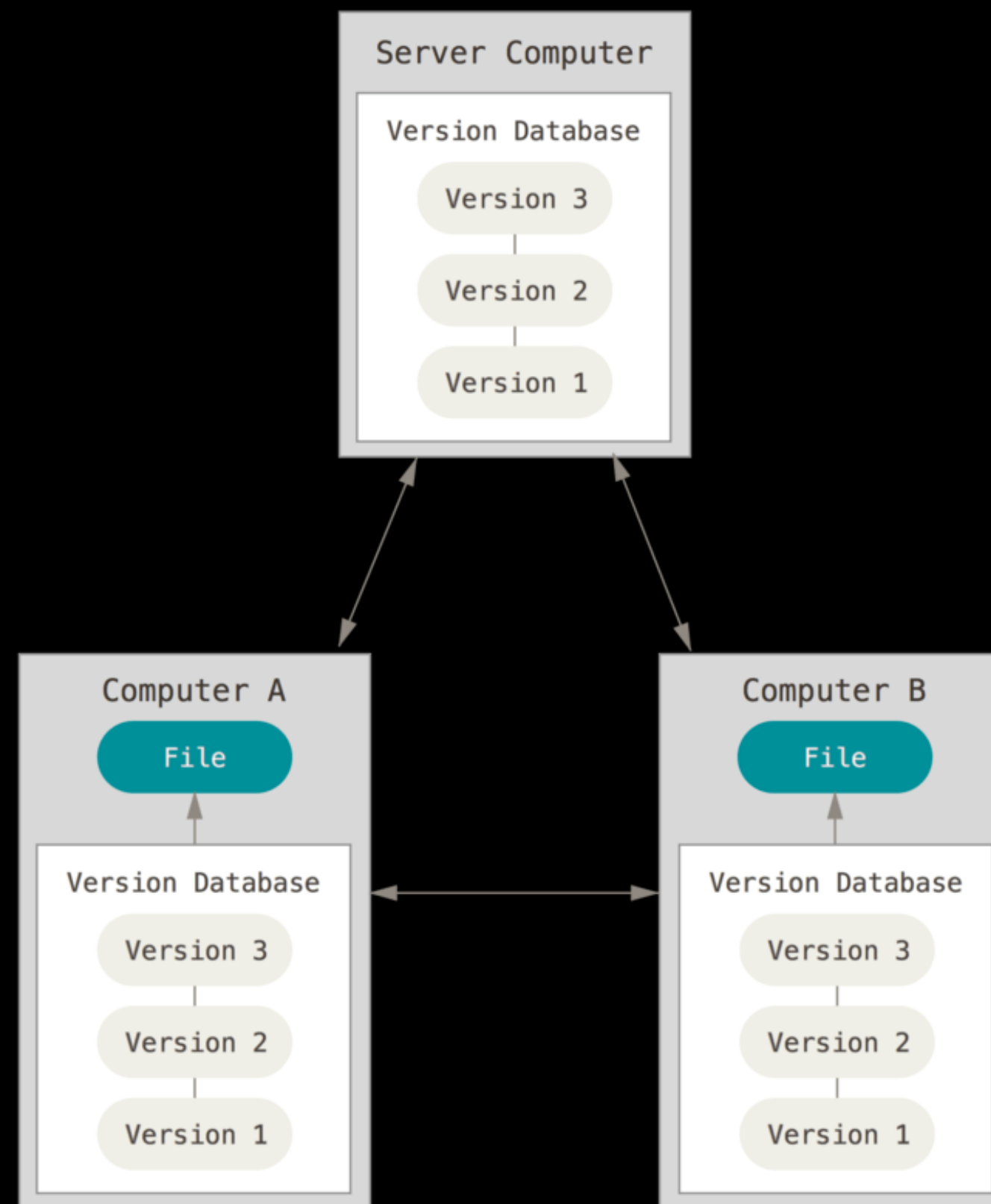


# Synchronizes code between different people.

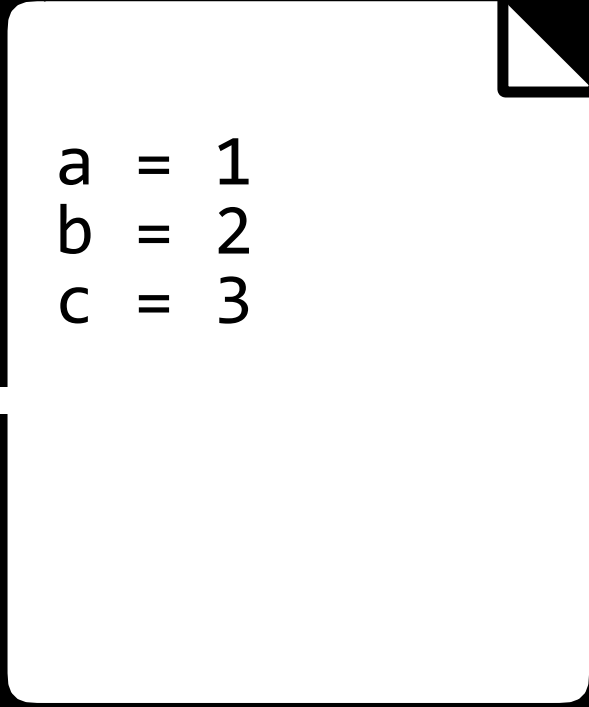


# Synchronizes code between different people.



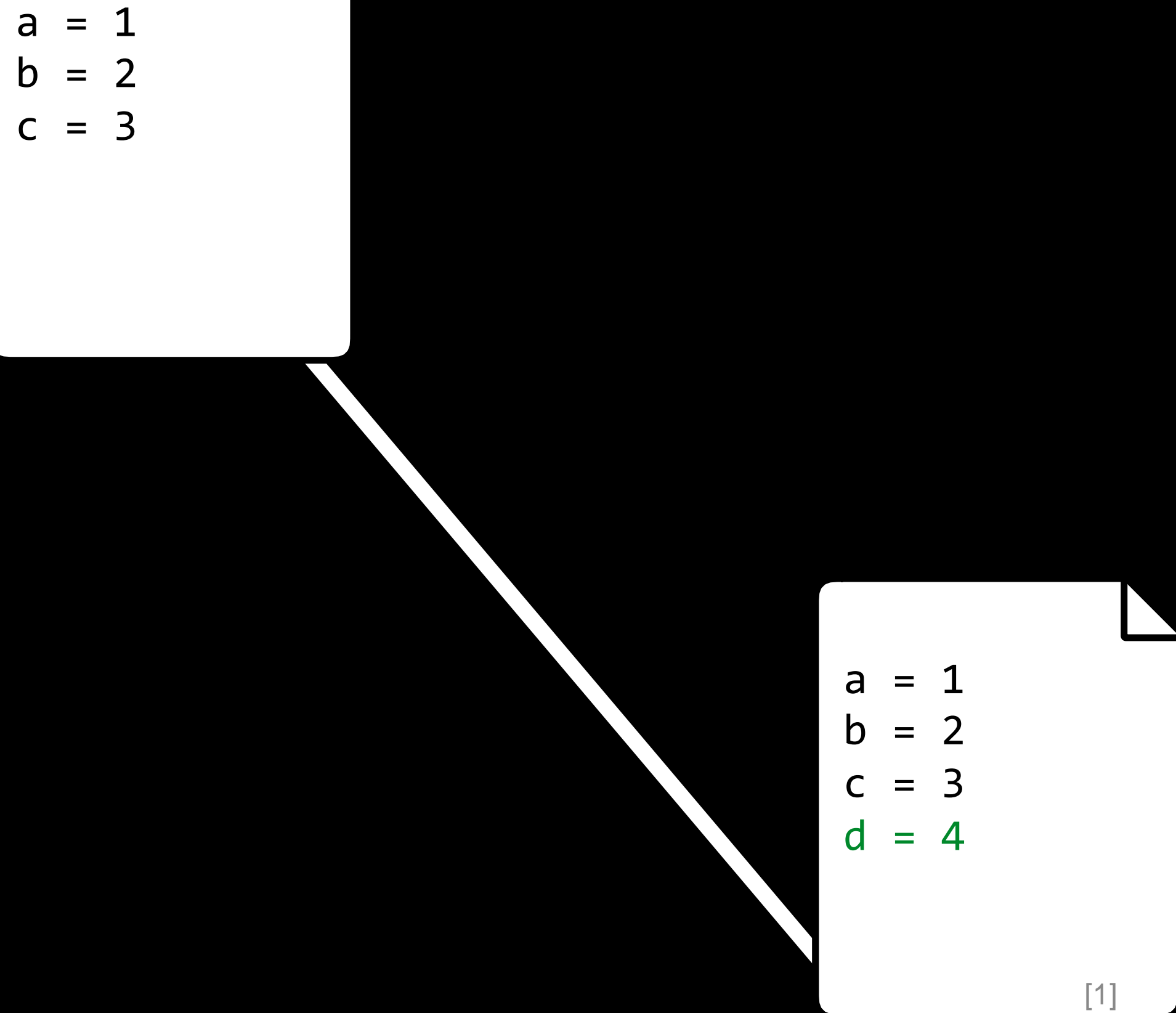


# Test changes to code without losing the original.



```
a = 1  
b = 2  
c = 3
```

# Test changes to code without losing the original.



```
a = 1  
b = 2  
c = 3
```

The diagram illustrates a workflow for testing code changes. It starts with a code block containing three lines of Python code: `a = 1`, `b = 2`, and `c = 3`. A horizontal line extends from the left side of this block. A diagonal line then connects the bottom-right corner of the first block to the top-left corner of a second block. The second block contains the same three lines of code, but with an additional line, `d = 4`, highlighted in green. At the bottom right of the second block, there is a small label `[1]`.

```
a = 1  
b = 2  
c = 3  
d = 4
```

[1]

# Test changes to code without losing the original.

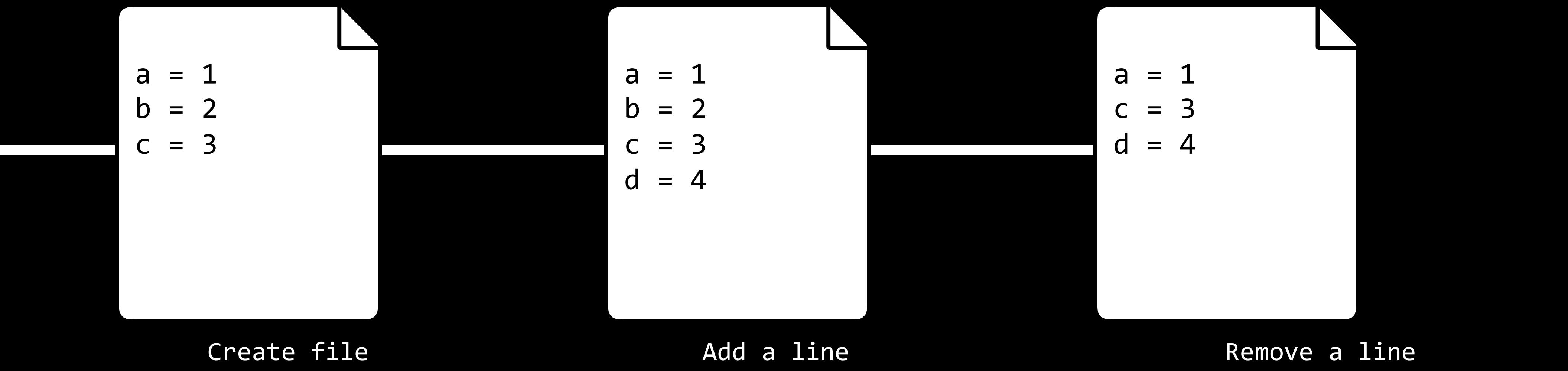
```
a = 1  
b = 2  
c = 3
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

[1]

# Revert back to old versions of code.



```
a = 1  
b = 2  
c = 3
```

Create file

```
a = 1  
b = 2  
c = 3  
d = 4
```

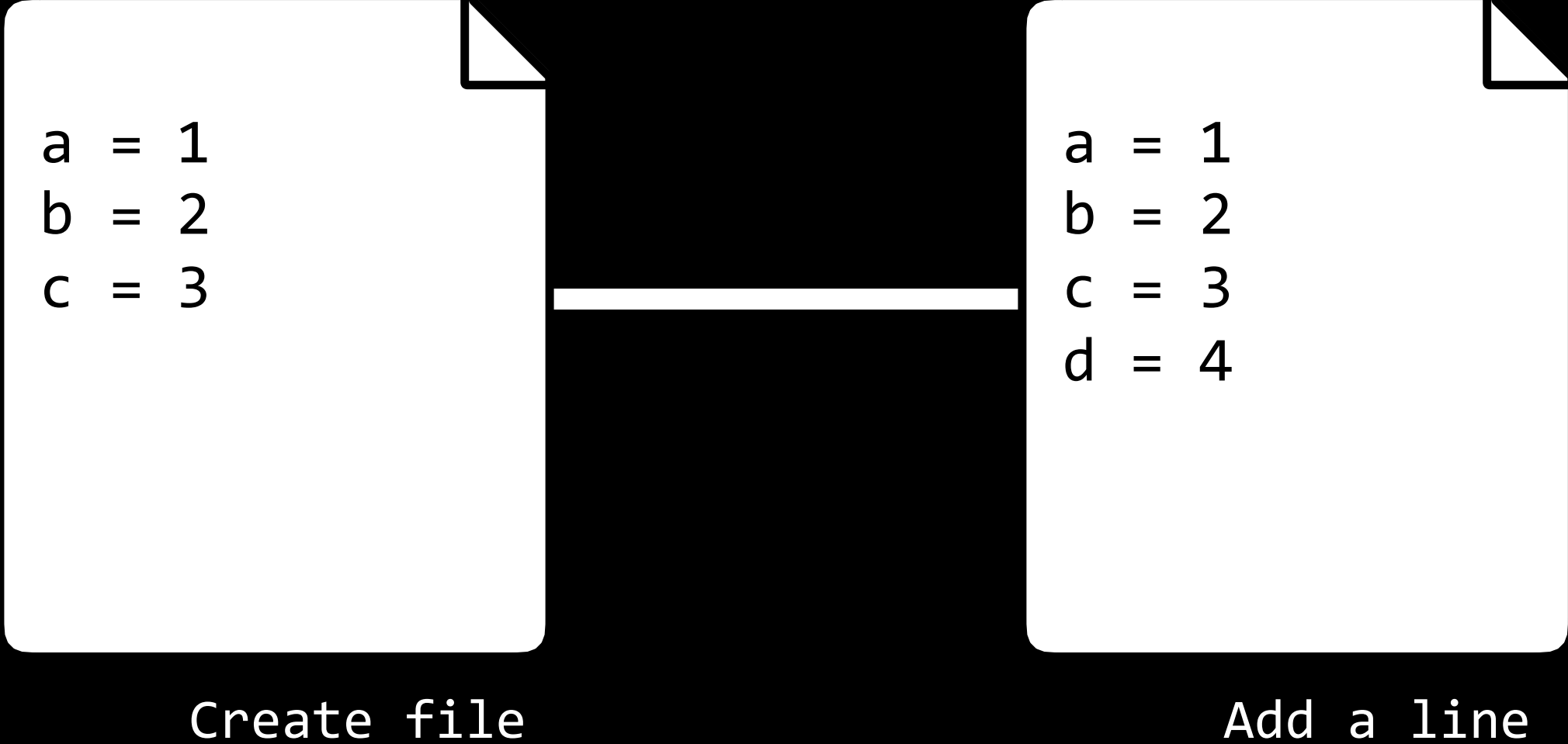
Add a line

```
a = 1  
c = 3  
d = 4
```

Remove a line



# Revert back to old versions of code.



```
a = 1  
b = 2  
c = 3
```

Create file

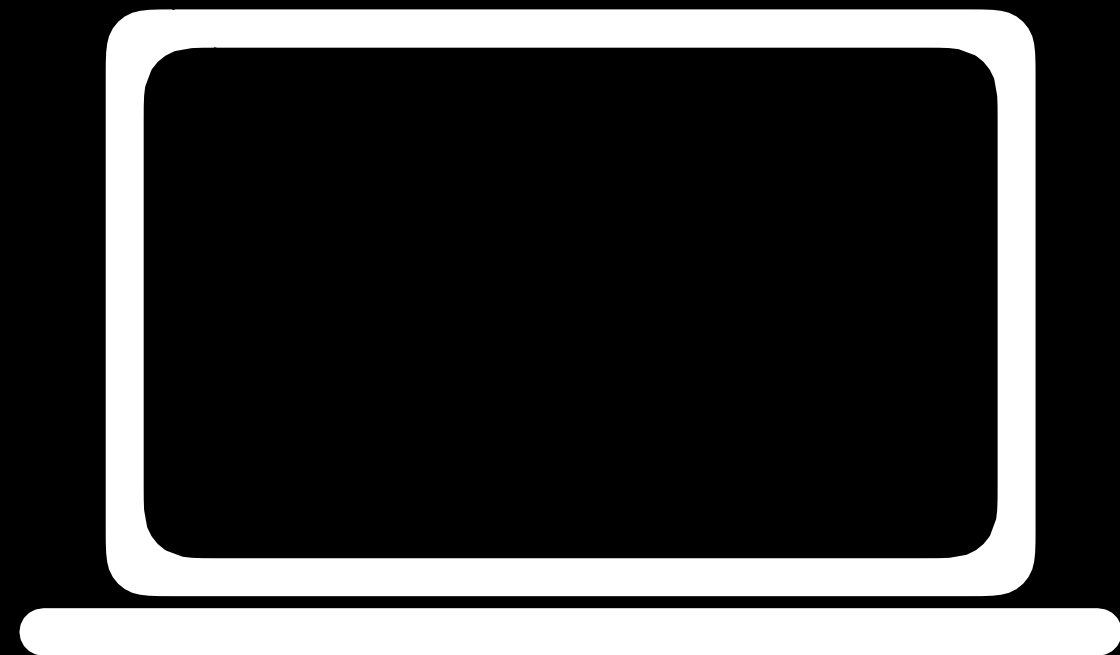
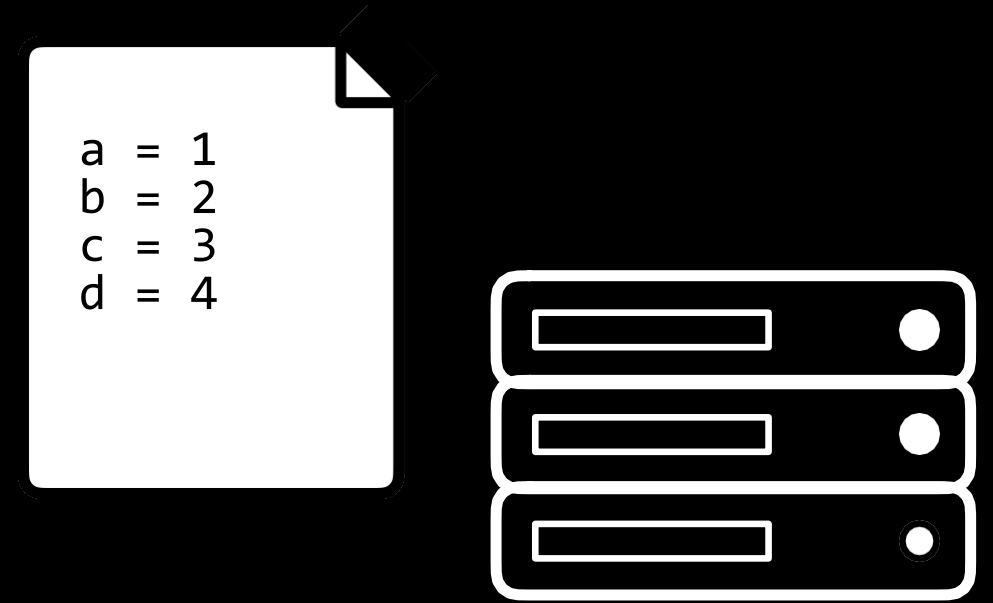
```
a = 1  
b = 2  
c = 3  
d = 4
```

Add a line

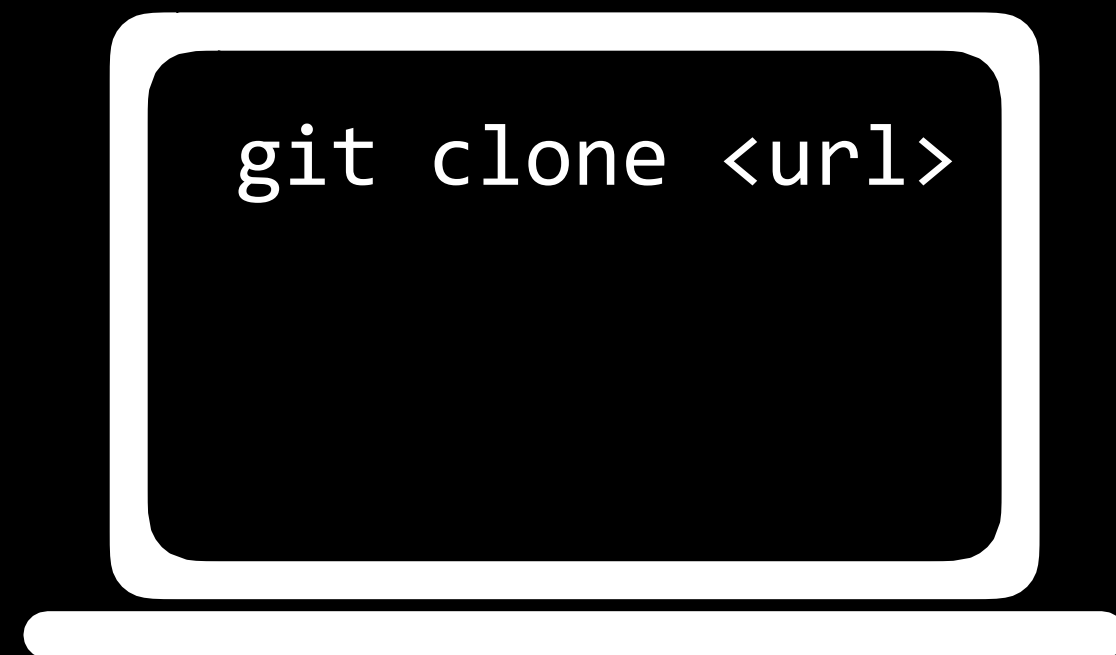
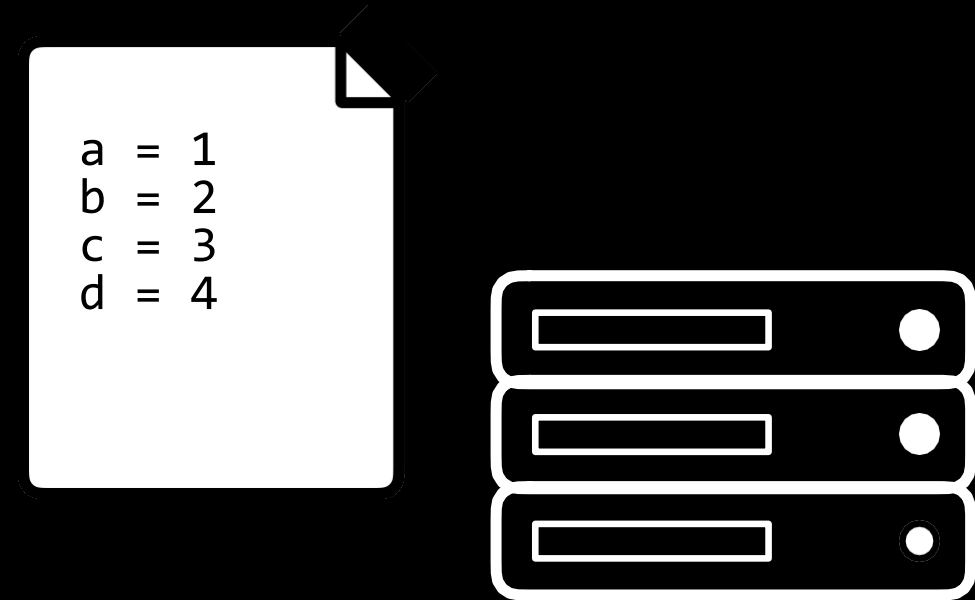
# GitHub

git clone

```
git clone <url>
```

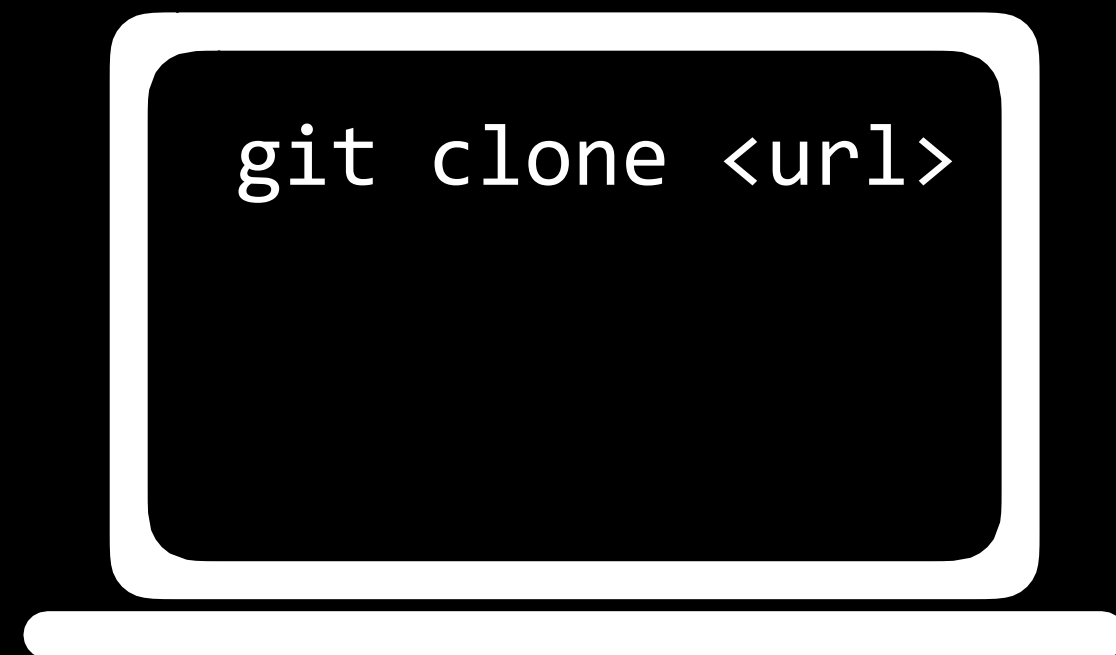
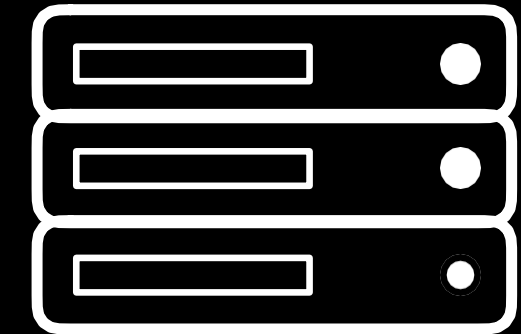


```
git clone <url>
```



```
git clone <url>
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

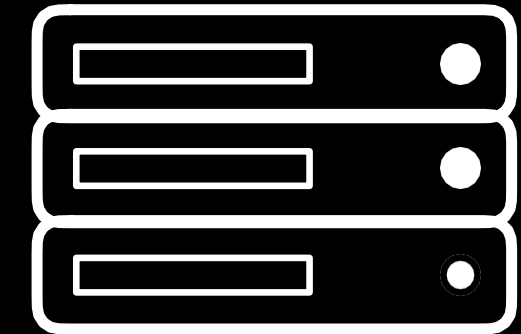


```
a = 1  
b = 2  
c = 3  
d = 4
```

git add

```
git add <filename>
```

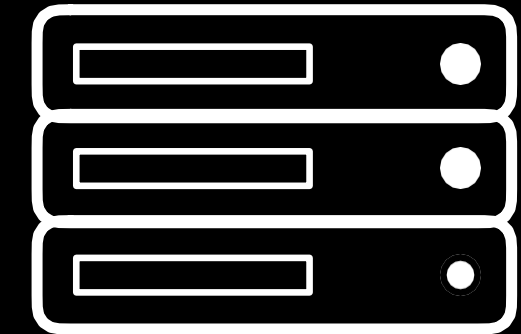
```
a = 1  
b = 2  
c = 3  
d = 4
```





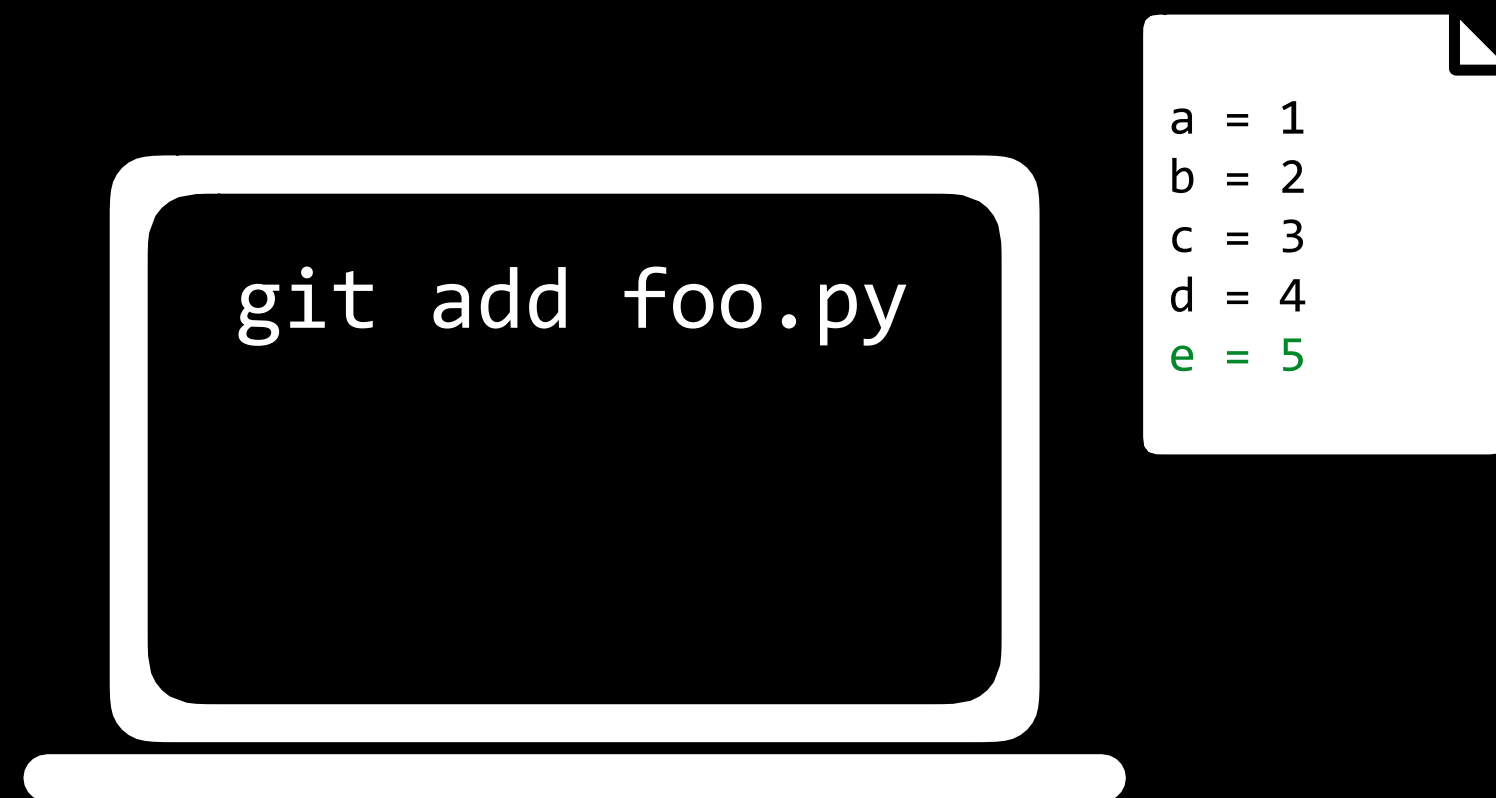
```
git add <filename>
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

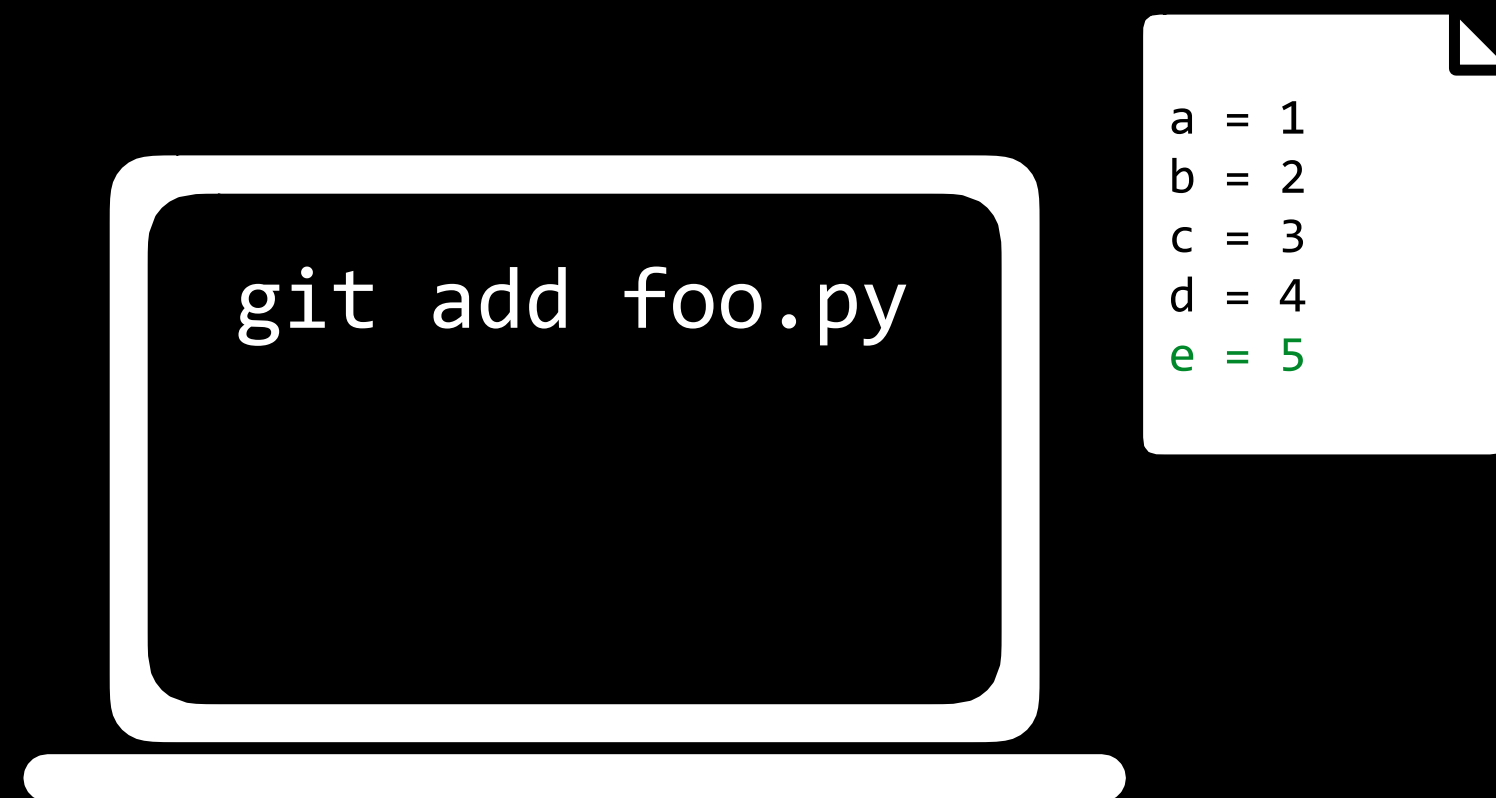
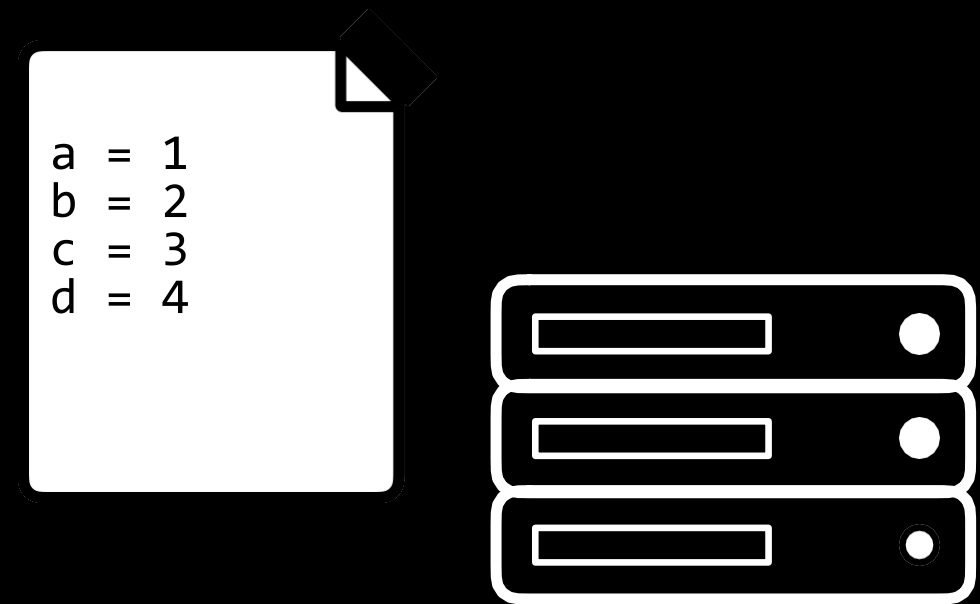


```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

```
git add <filename>
```



```
git add <filename>
```

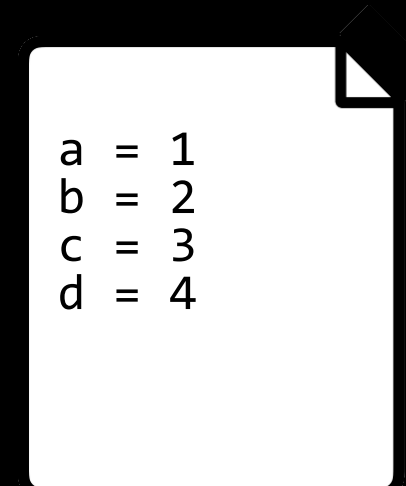


Changes to be committed:

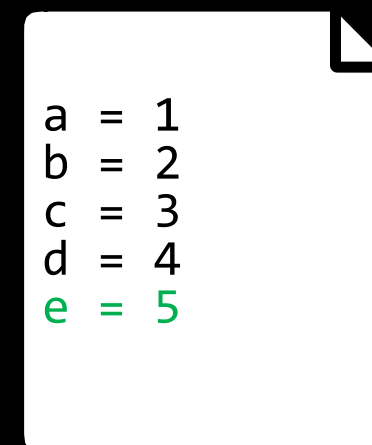
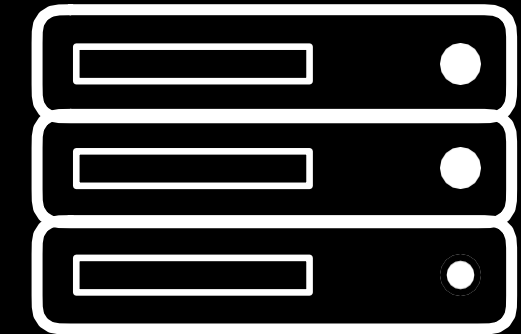
**modified: foo.py**

git commit

```
git commit -m "message"
```



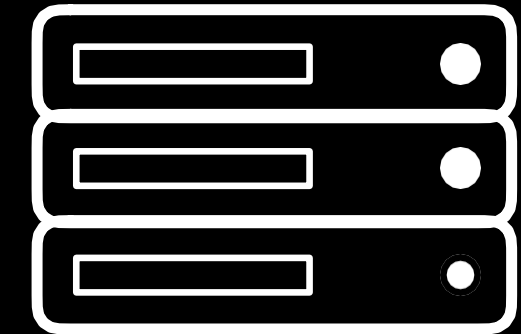
```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

```
git commit -m "message"
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

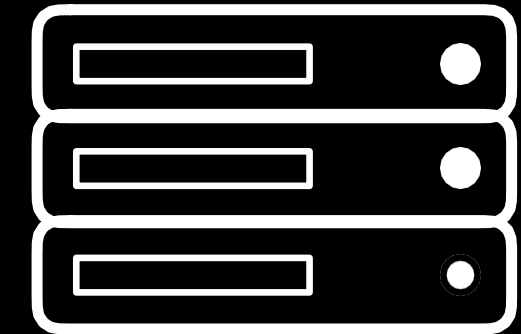


```
git commit -m  
"Add line"
```

```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

```
git commit -m "message"
```

```
a = 1  
b = 2  
c = 3  
d = 4
```



```
git commit -m  
"Add line"
```

```
a = 1  
b = 2  
c = 3  
d = 4
```

```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

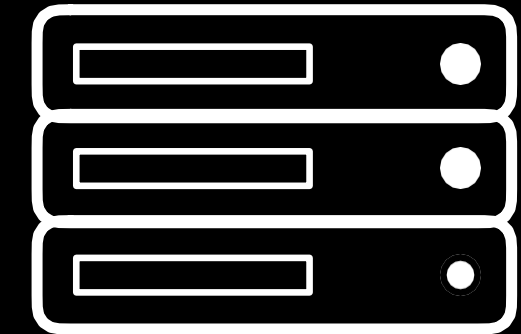
Add line

```
git status
```



# git status

```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4
```

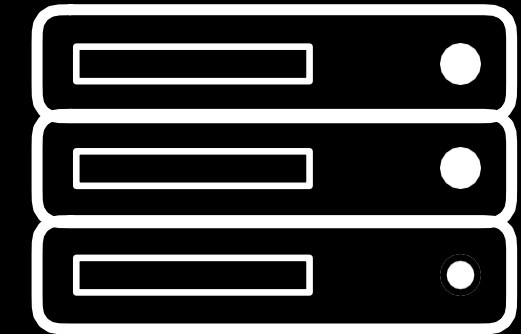
```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

Add line



# git status

```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4
```

```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

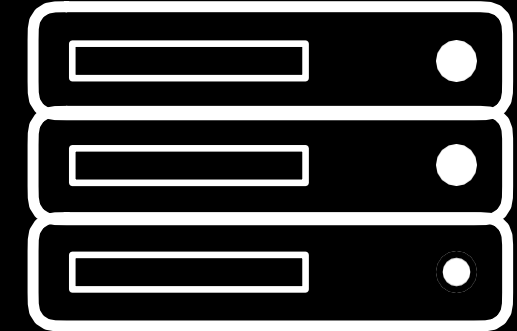
Add line



git status

# git status

```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4
```

```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

Add line

```
git status
```

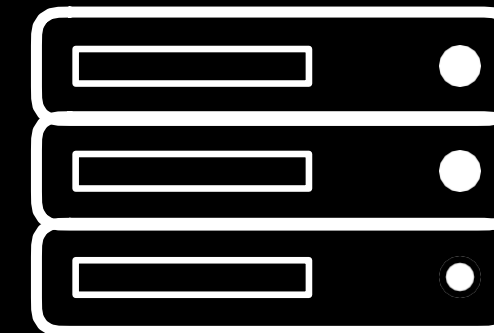
On branch master

Your branch is ahead of 'origin/master' by 1 commit.  
(use "git push" to publish your local commits)

git push

# git push

```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4
```

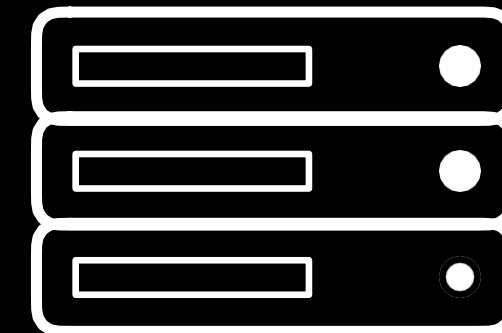


```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

Add line

# git push

```
a = 1  
b = 2  
c = 3  
d = 4
```



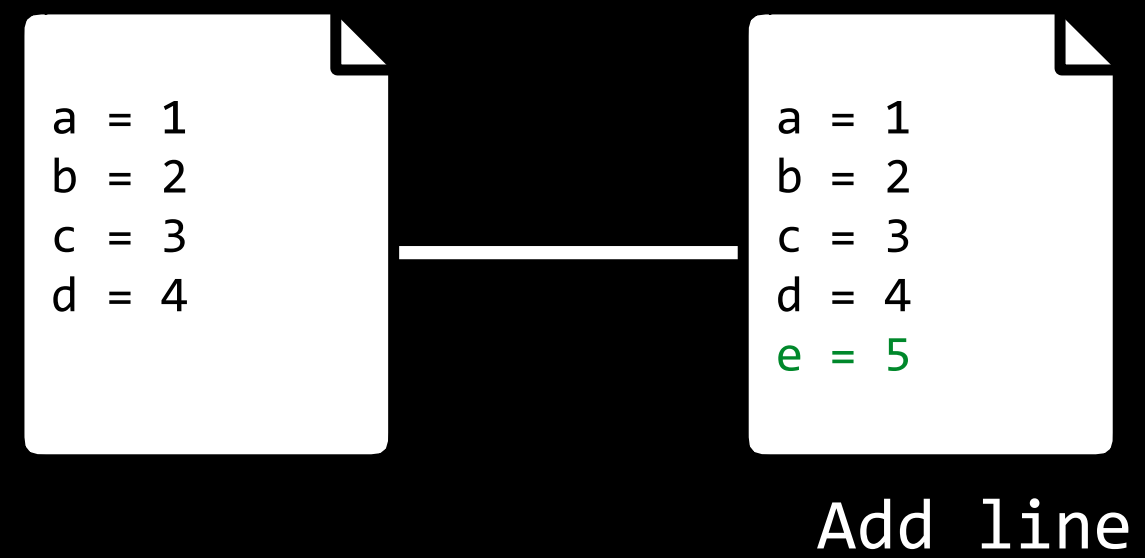
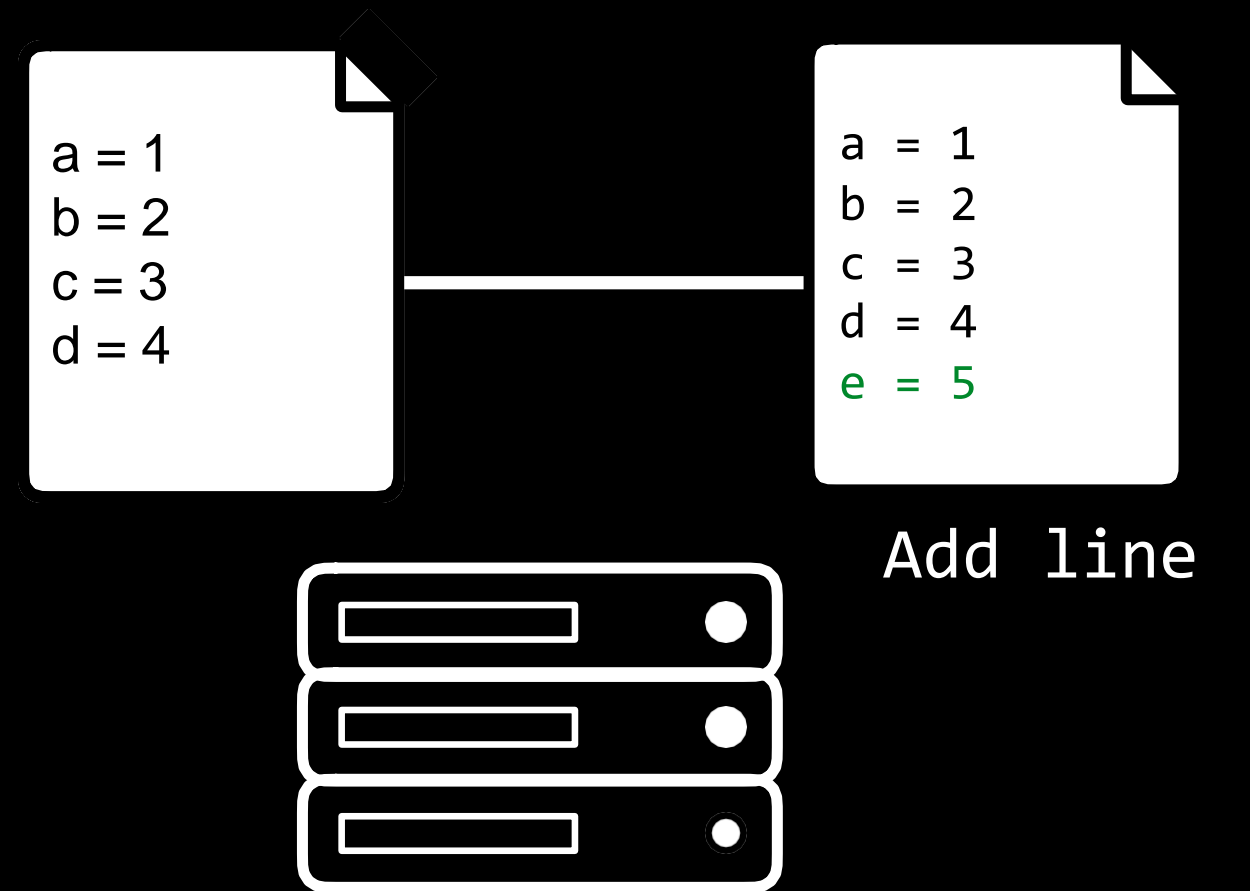
```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

Add line

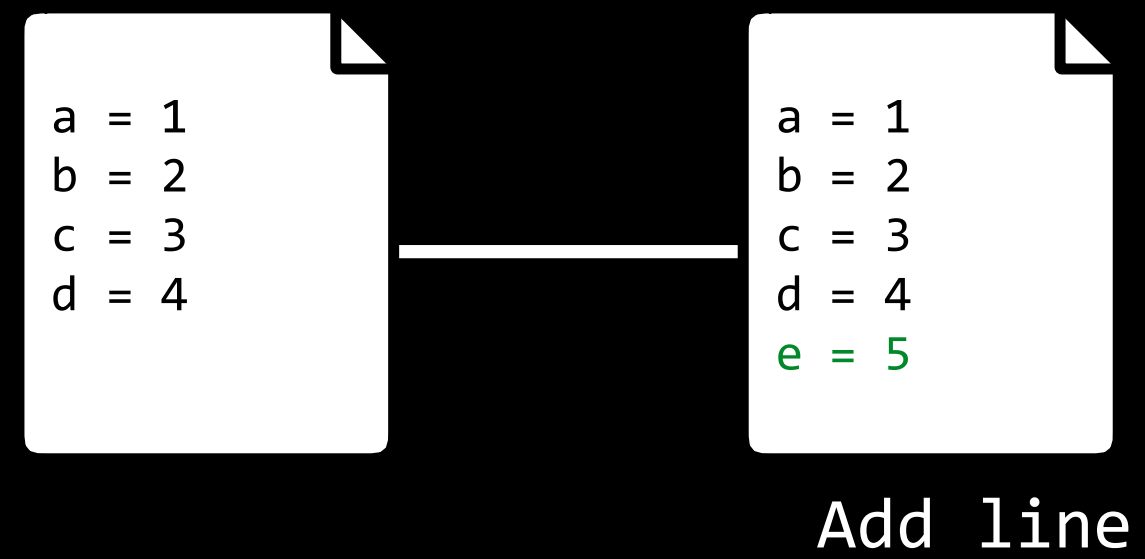
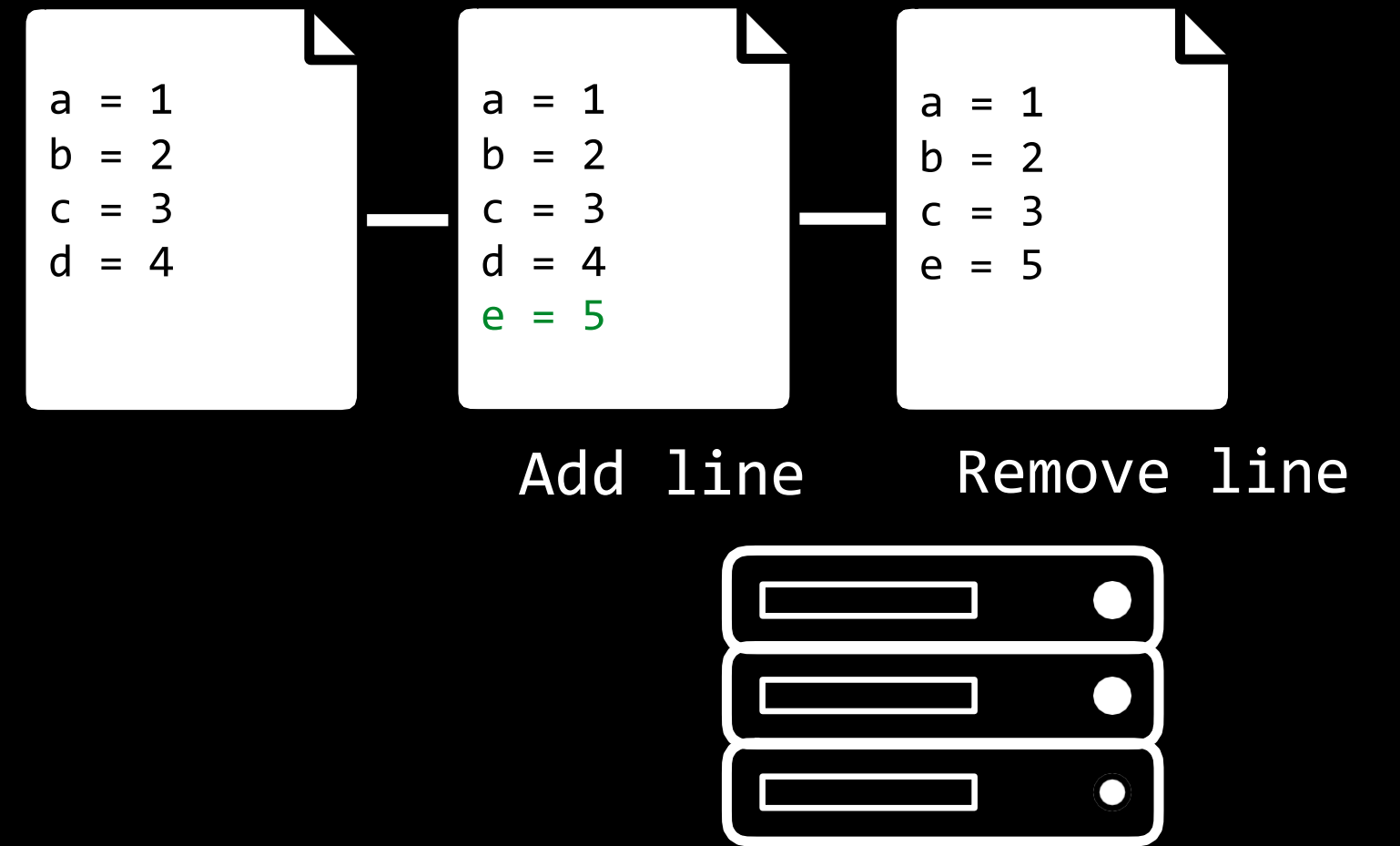
# git push



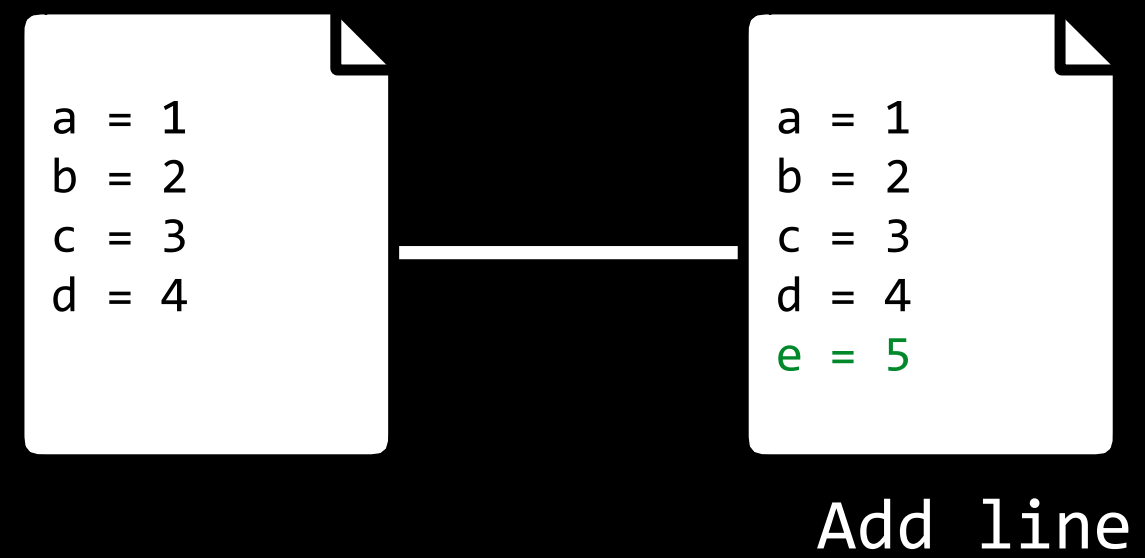
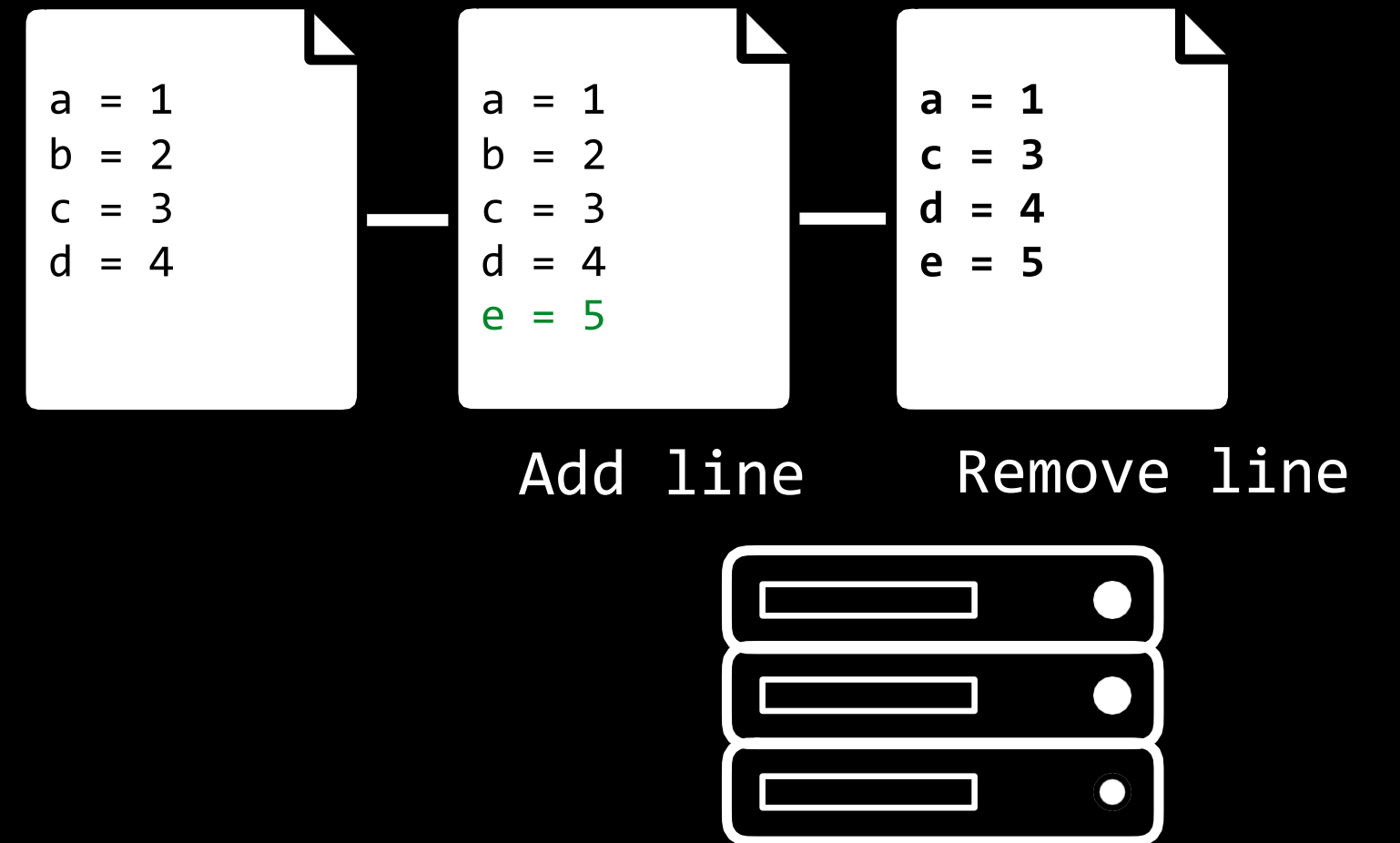
```
git pull
```



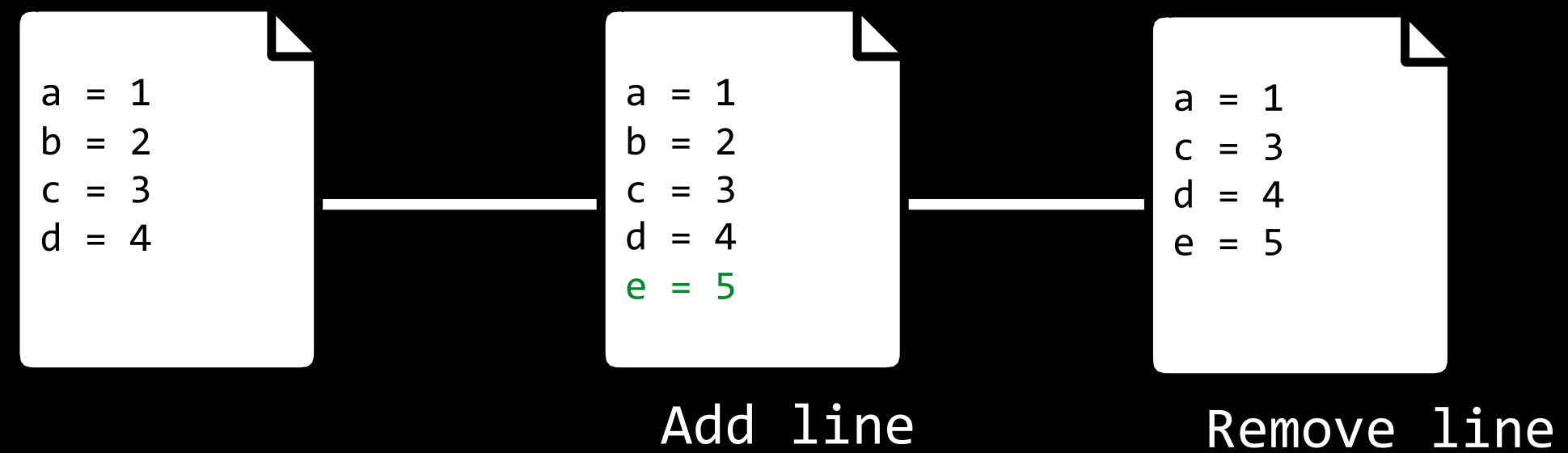
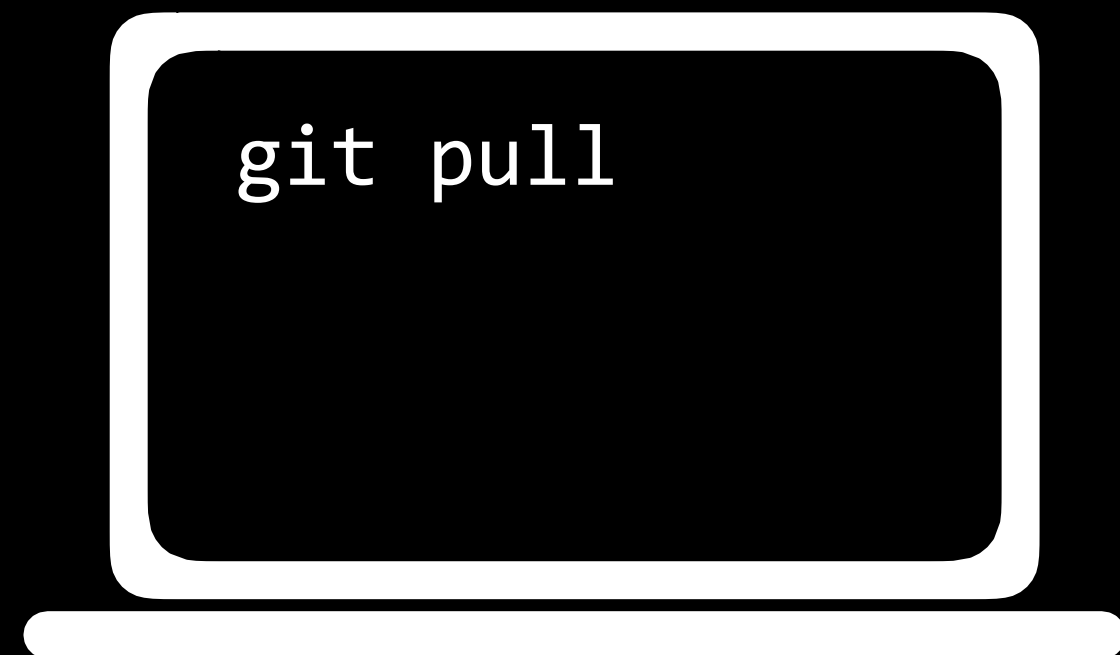
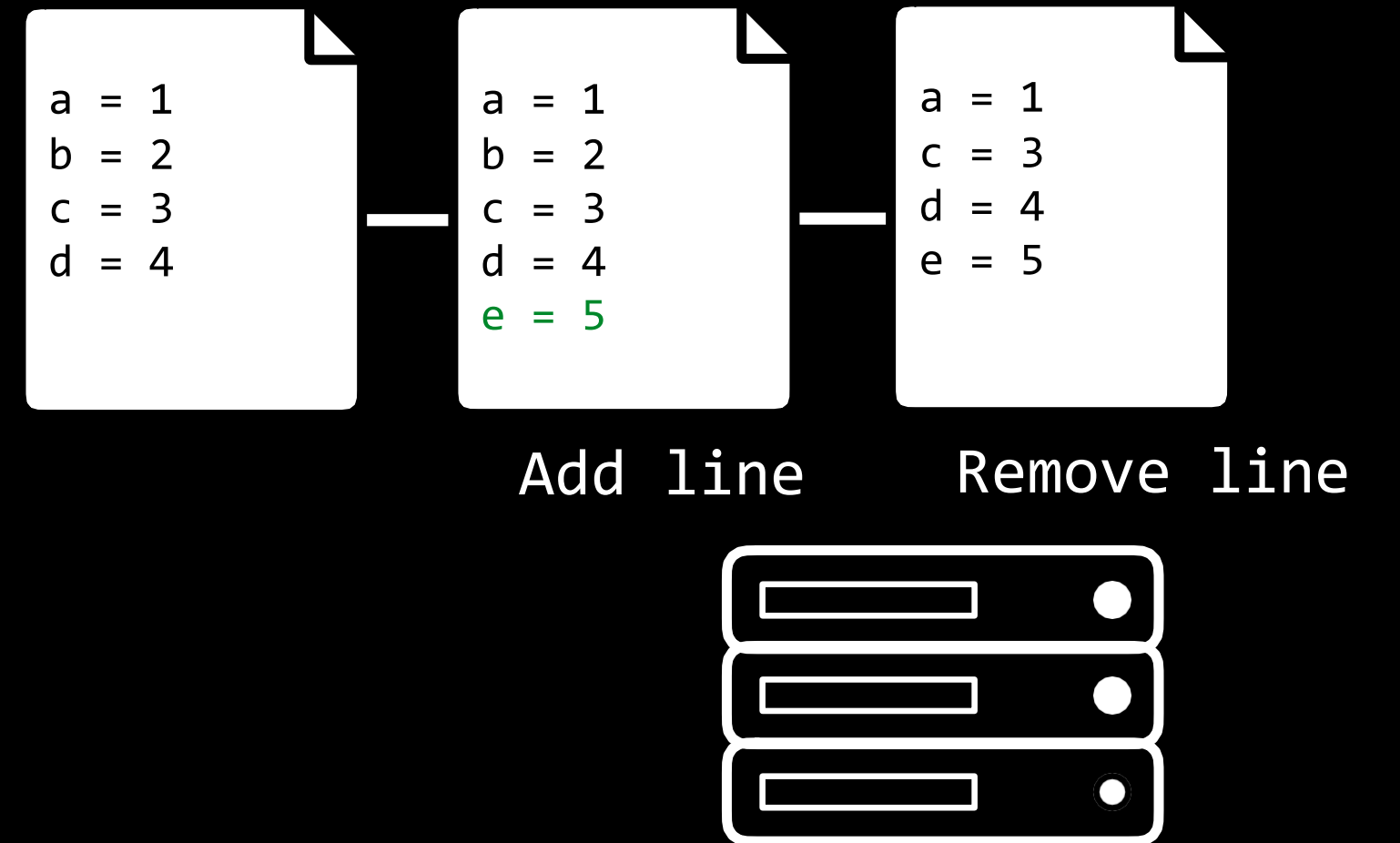
# git pull



# git pull

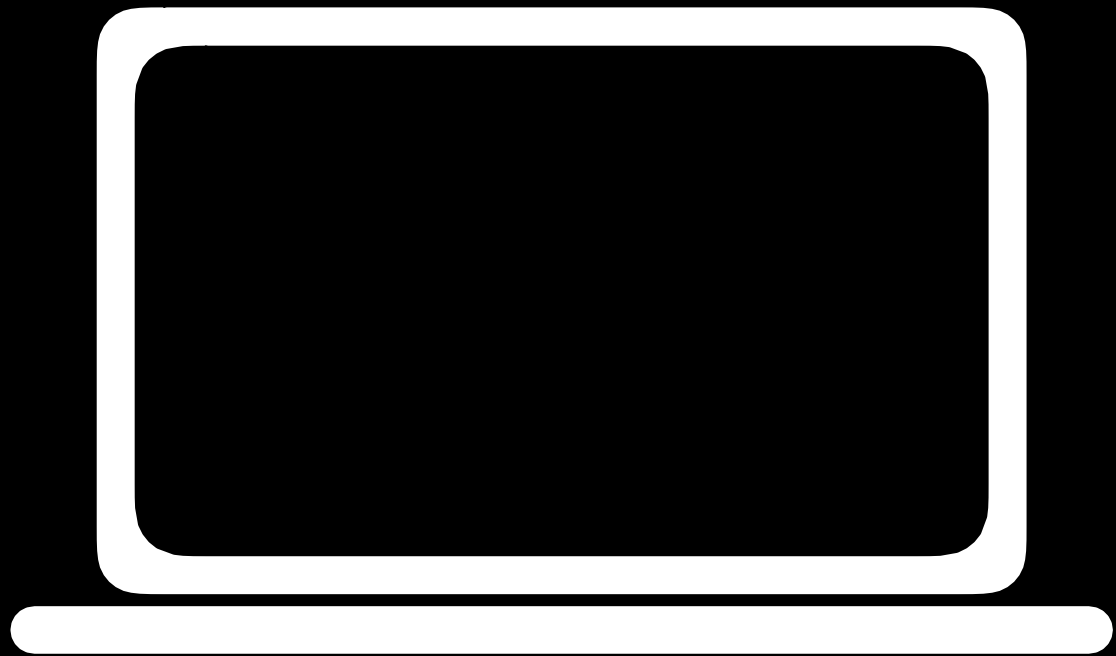


# git pull



# Merge Conflicts

# Merge Conflicts



# Merge Conflicts



```
git pull
```

# Merge Conflicts



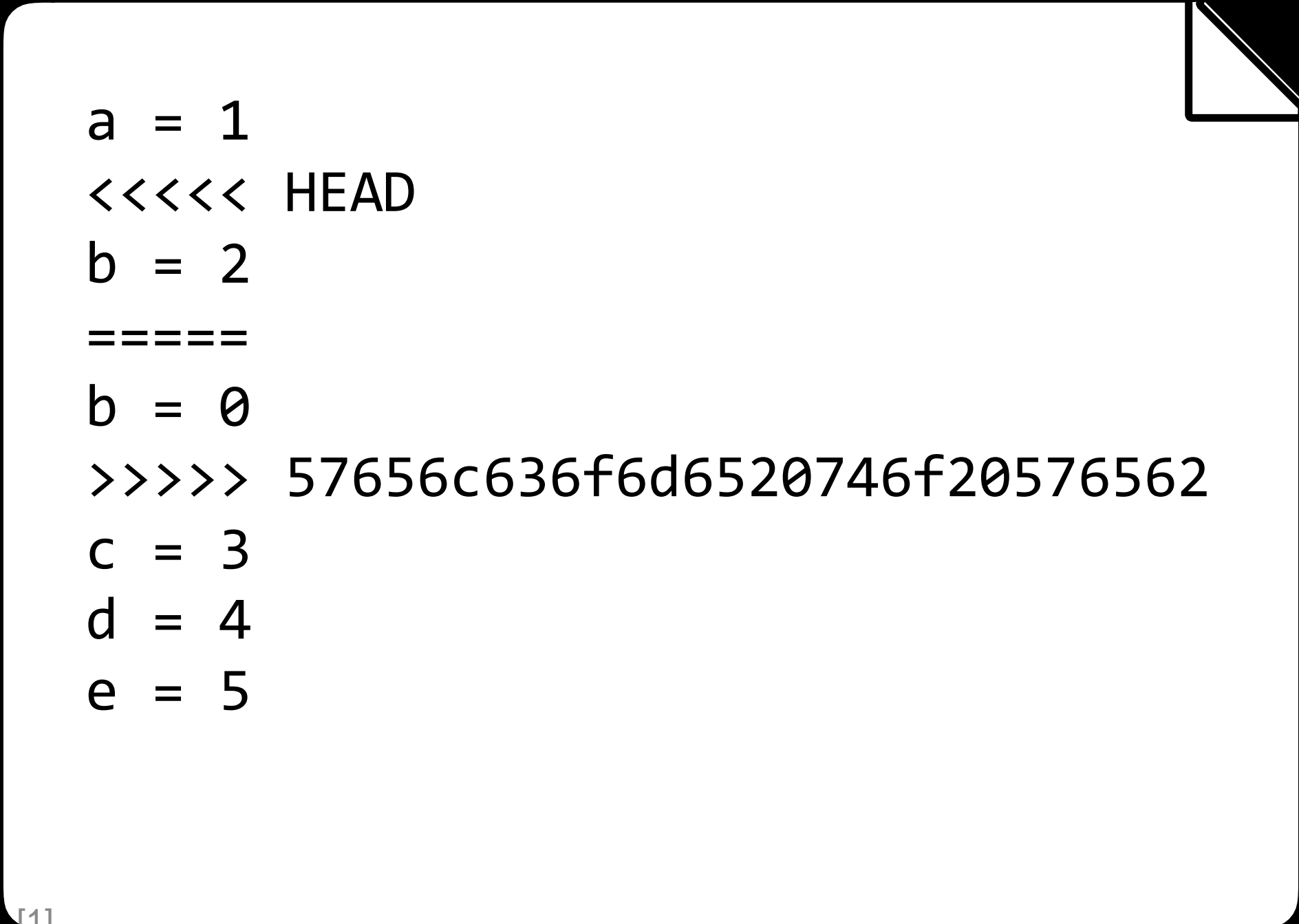
```
git pull
```

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

# Merge Conflicts



```
git pull
```



```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
c = 3
d = 4
e = 5
```



# Merge Conflicts




```
git pull
```

your  
changes

remote  
changes

```
a = 1
<<<<< HEAD
{ b = 2
  =====
  { b = 0
    >>>>> 57656c636f6d6520746f20576562
    c = 3
    d = 4
    e = 5
```

conflicting commit



# Merge Conflicts



```
git pull
```

```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
c = 3
d = 4
e = 5
```

# Merge Conflicts



```
git pull
```

```
a = 1
```

```
b = 2
```

```
c = 3
```

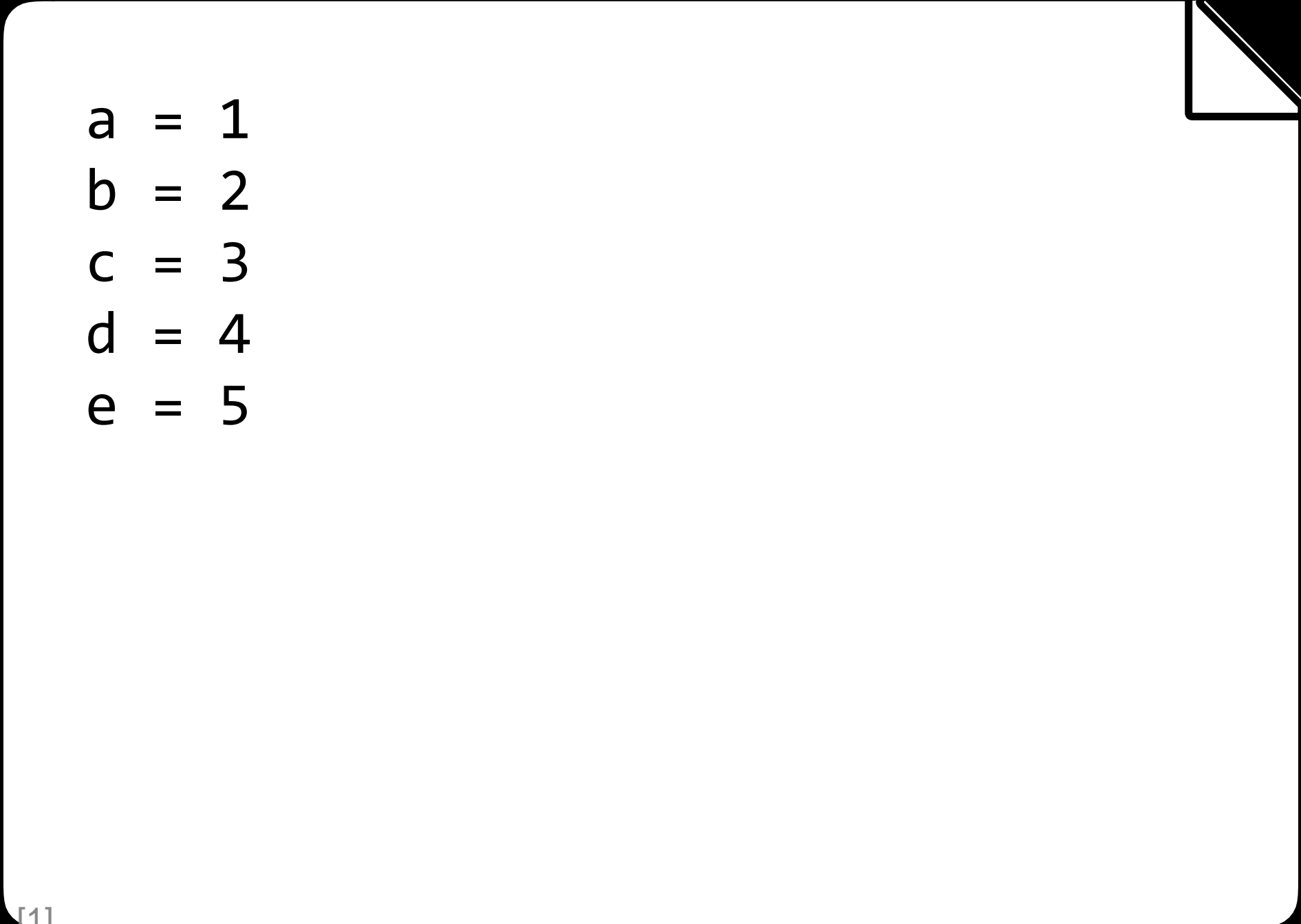
```
d = 4
```

```
e = 5
```

# Merge Conflicts

A white terminal window icon with a thick border and a horizontal base. Inside the terminal, the text "git pull" is written in a monospaced font.

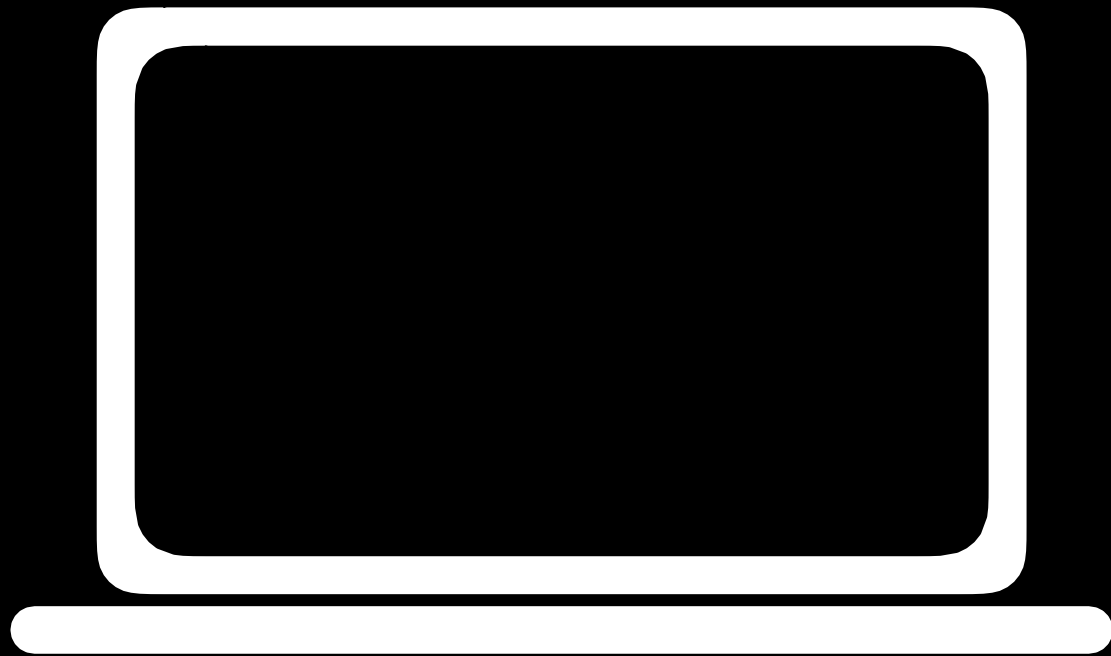
```
git pull
```

A white rectangular code block with rounded corners and a folded top-right corner. It contains five lines of code.

```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

git log

```
git log
```



```
git log
```



```
git log
```

# git log

```
commit 436f6d6d6974204d73672048657265
Author: Brian Yu <brian@cs.harvard.edu>
Date:   Tue Jan 14 14:06:28 2020 -0400
```

Remove a line

```
commit 57656c636f6d6520746f20576562
Author: Brian Yu <brian@cs.harvard.edu>
Date:   Tue Jan 14 14:05:28 2020 -0400
```

Add a line



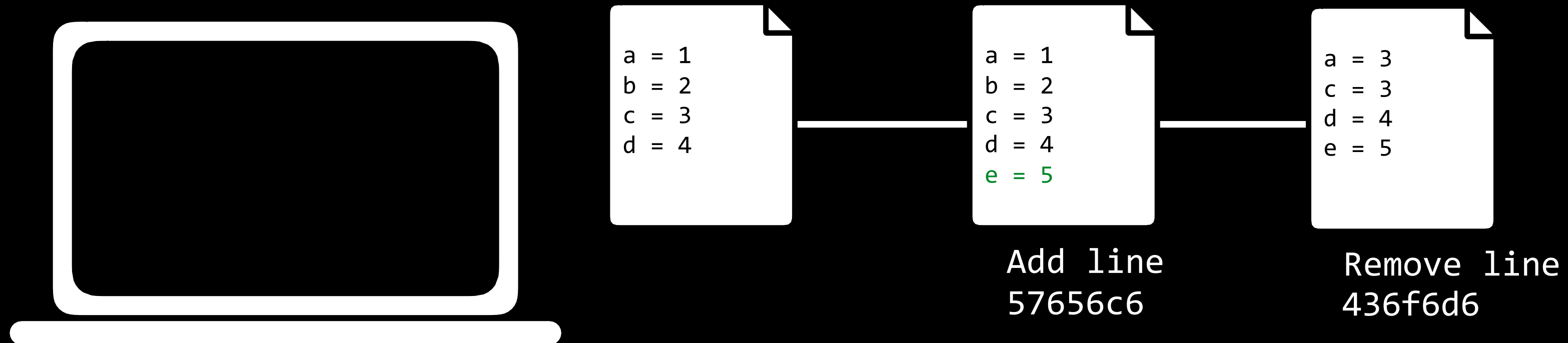
git log



```
git reset
```

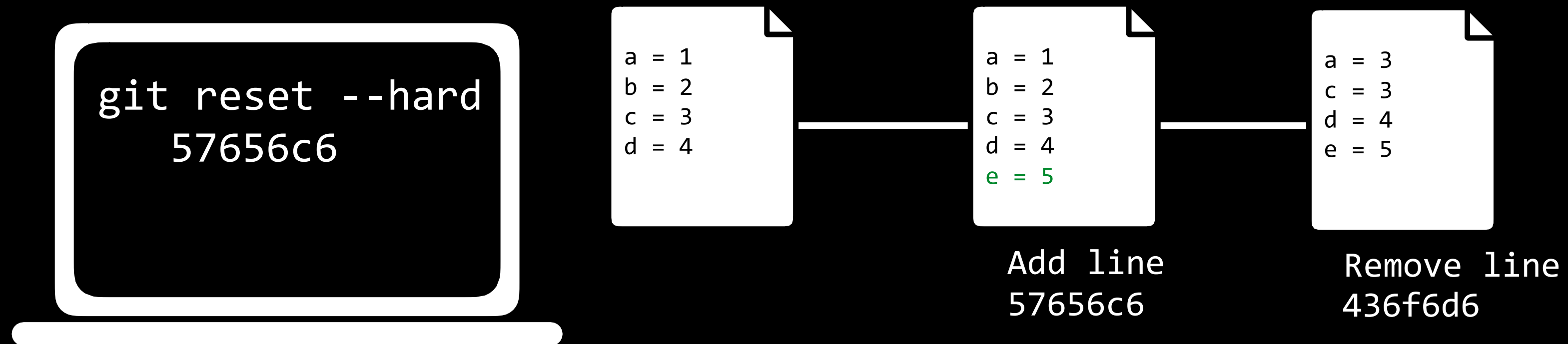
# git reset

- `git reset --hard <commit>`
- `git reset --hard origin/master`



# git reset

- `git reset --hard <commit>`
- `git reset --hard origin/master`

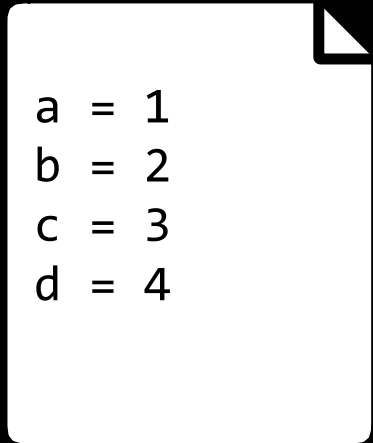


# git reset

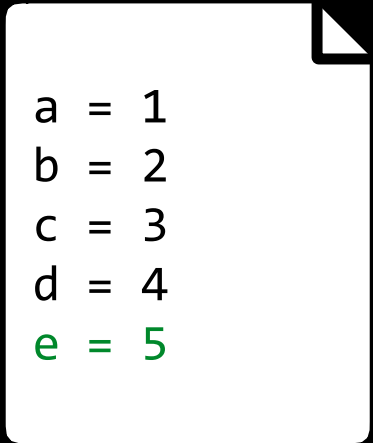
- `git reset --hard <commit>`
- `git reset --hard origin/master`



```
git reset --hard  
57656c6
```



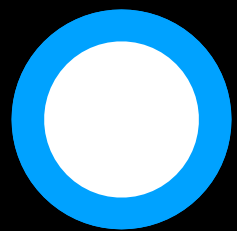
```
a = 1  
b = 2  
c = 3  
d = 4
```



```
a = 1  
b = 2  
c = 3  
d = 4  
e = 5
```

Add line  
57656c6

# Making Changes

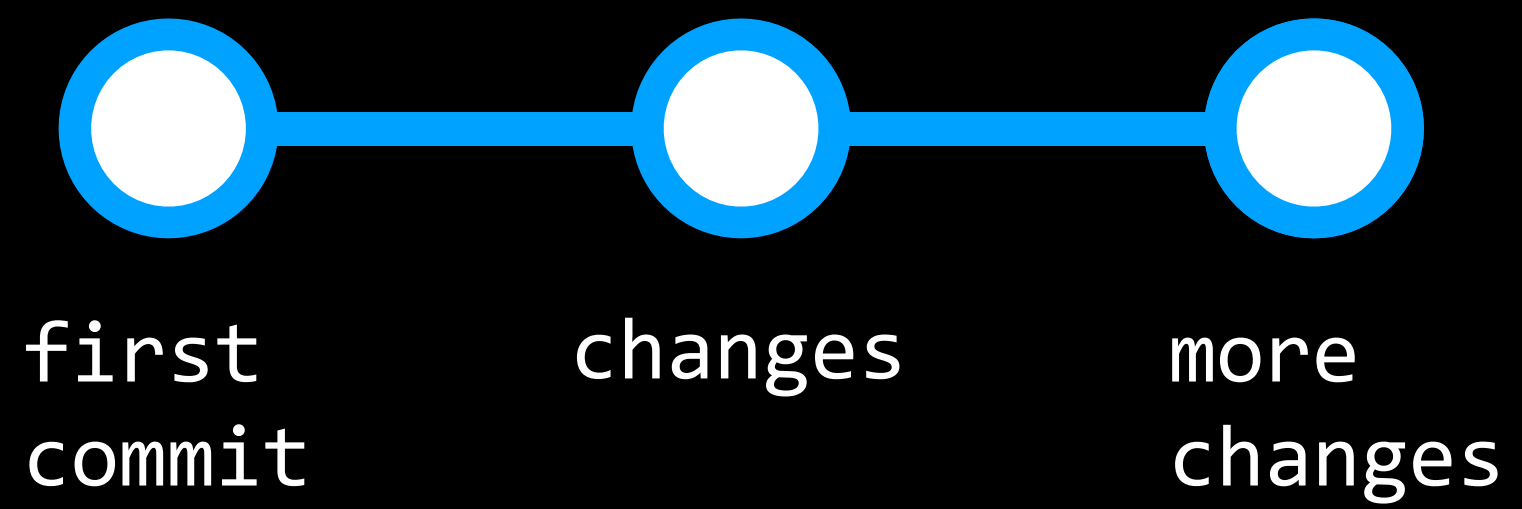


first  
commit

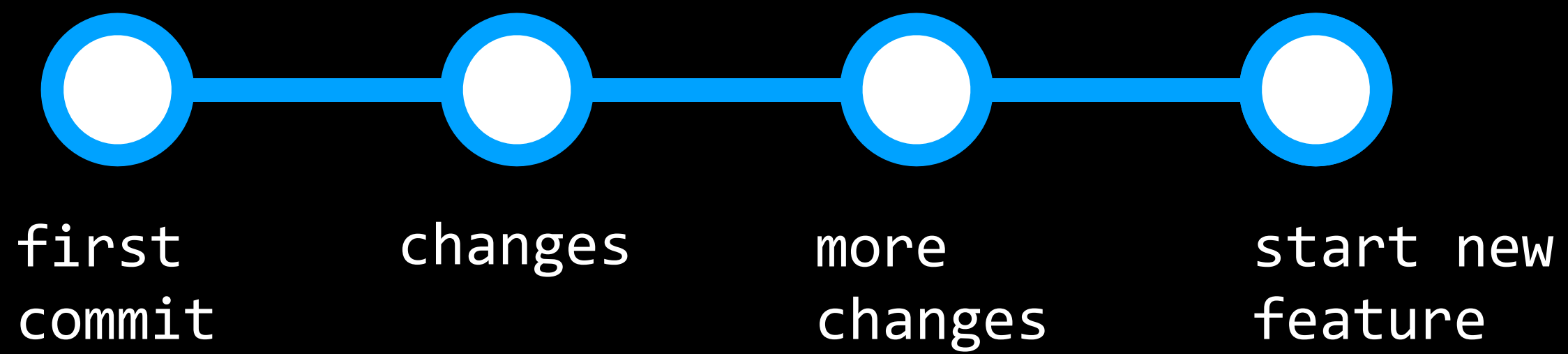


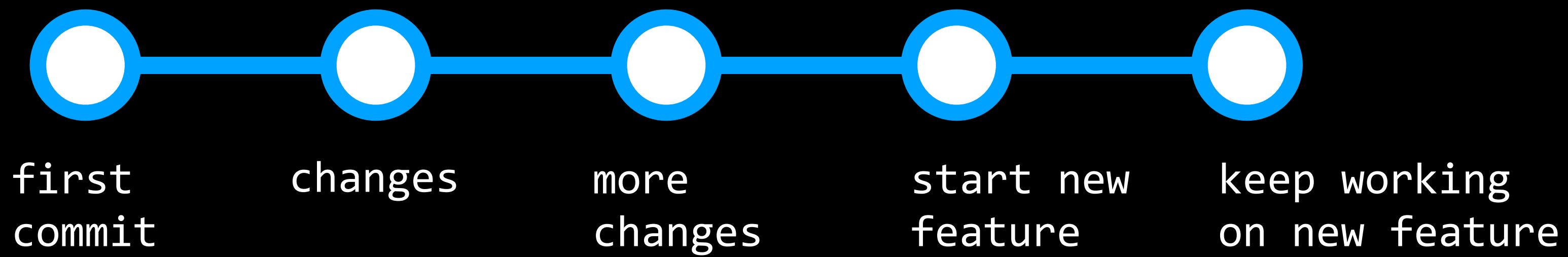
first  
commit

changes

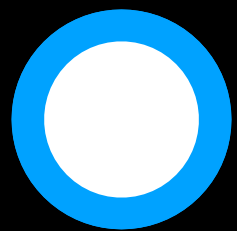








# Branching

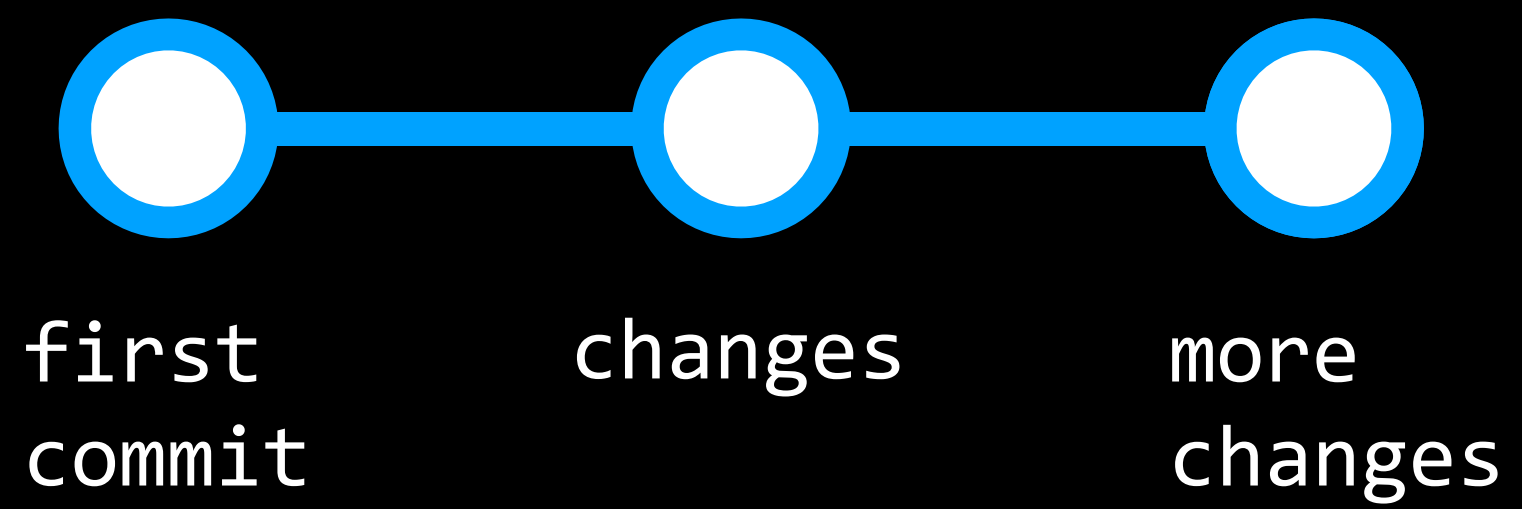


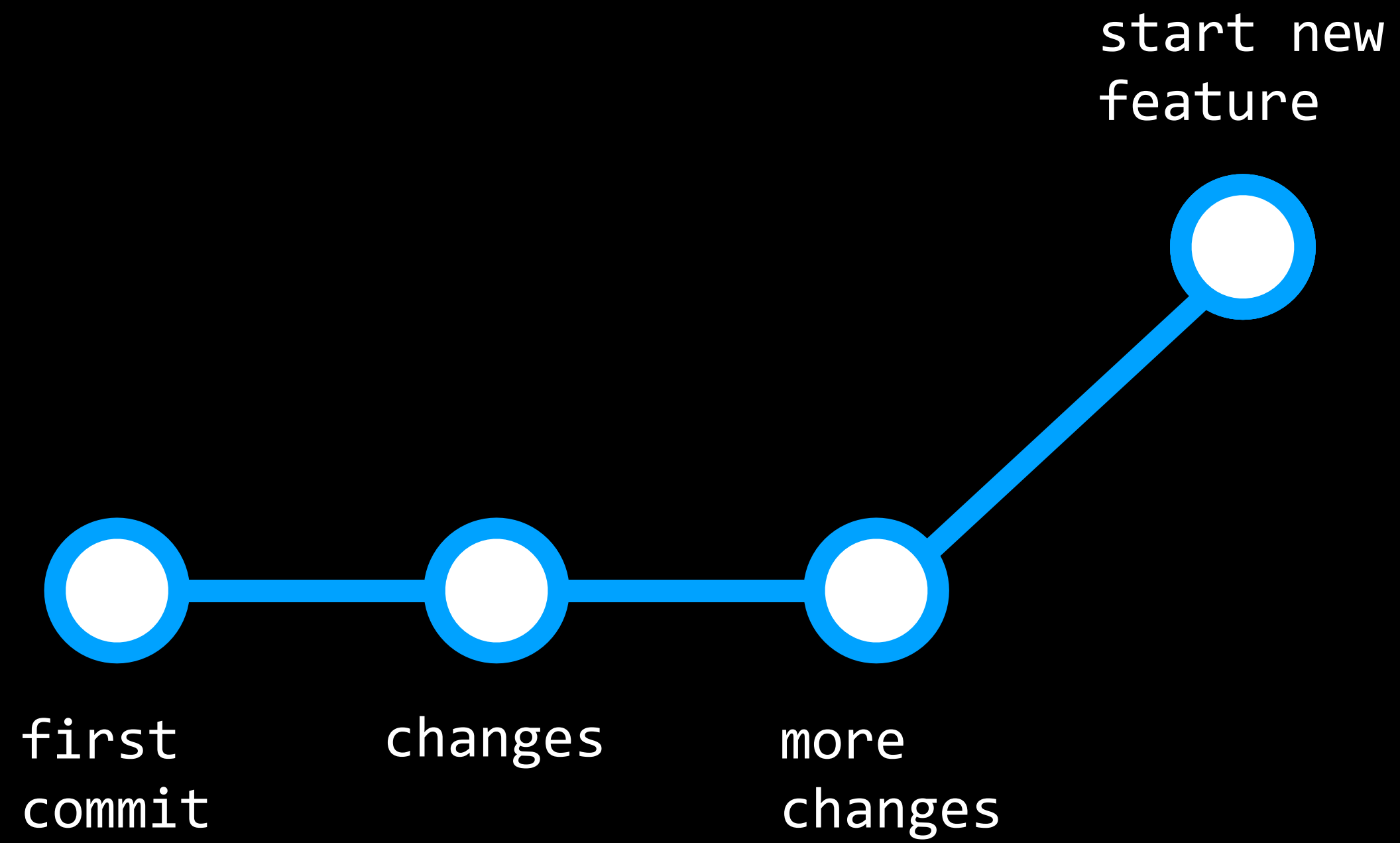
first  
commit

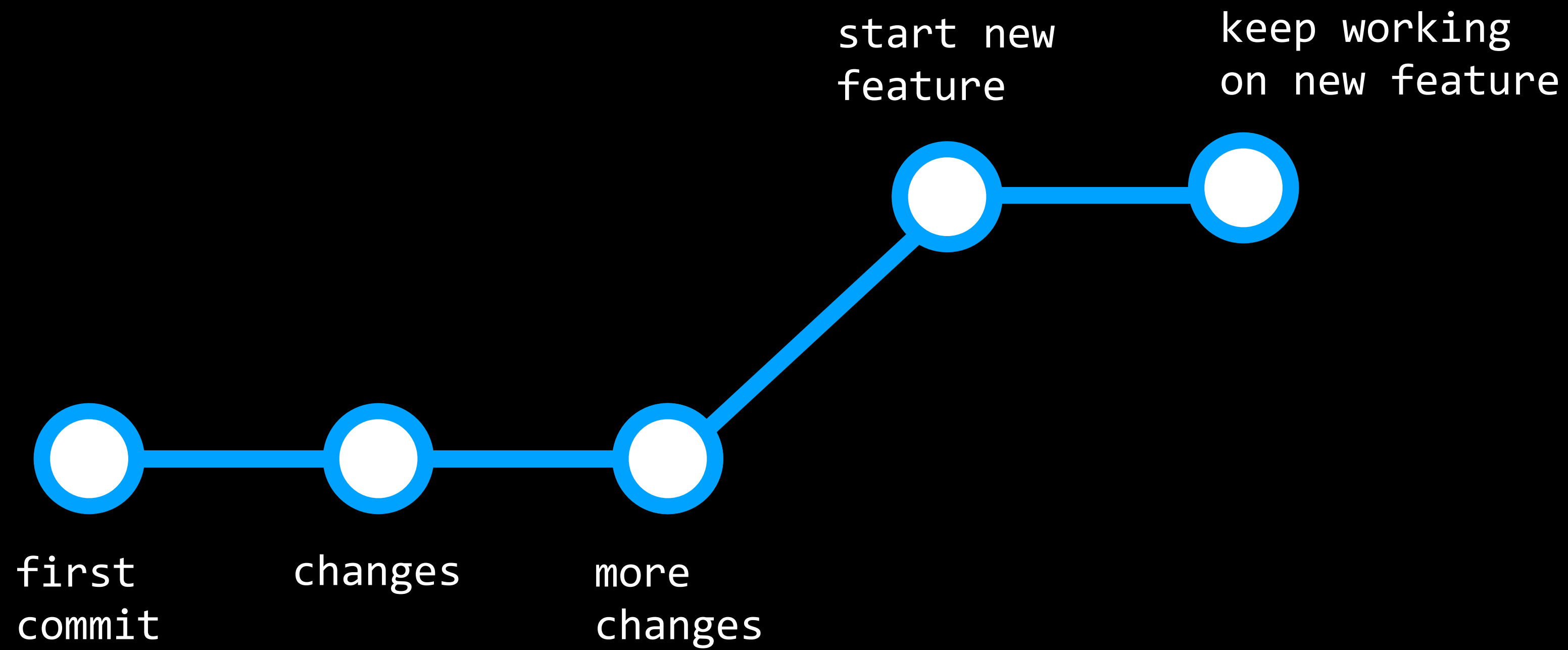


first  
commit

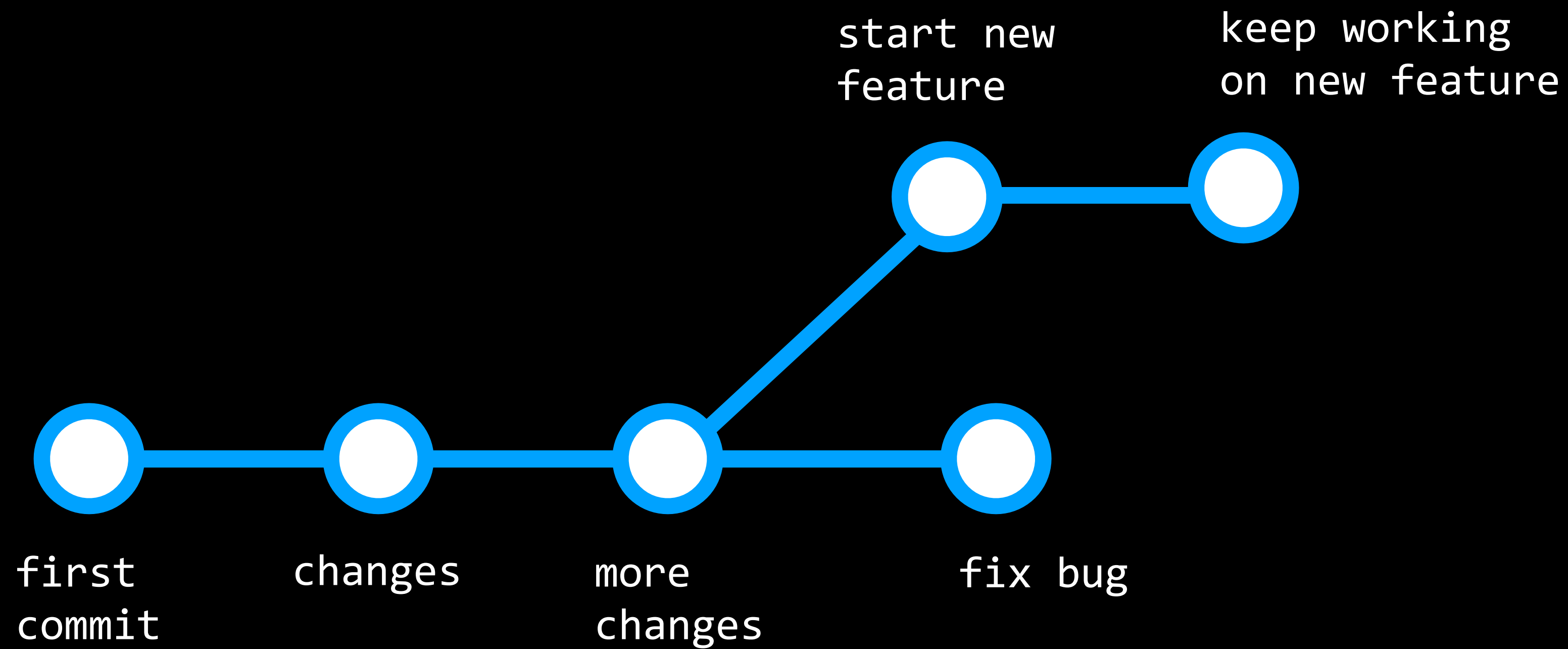
changes

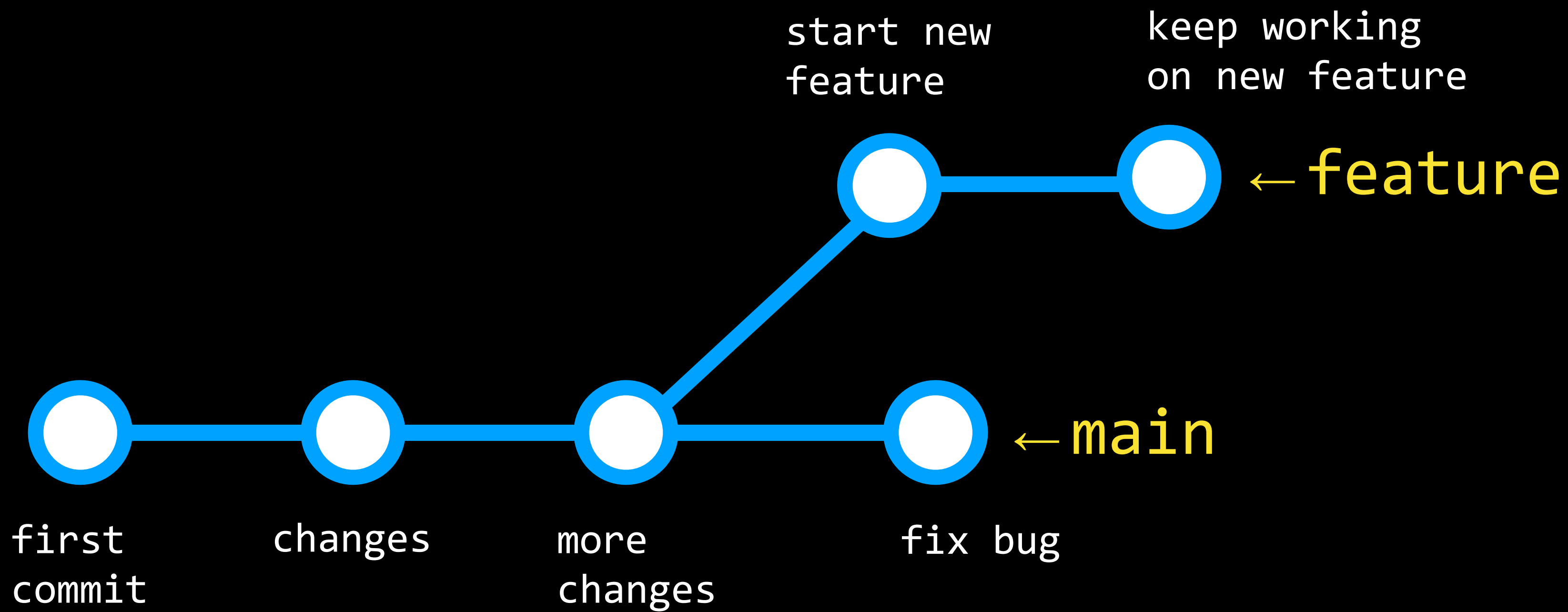




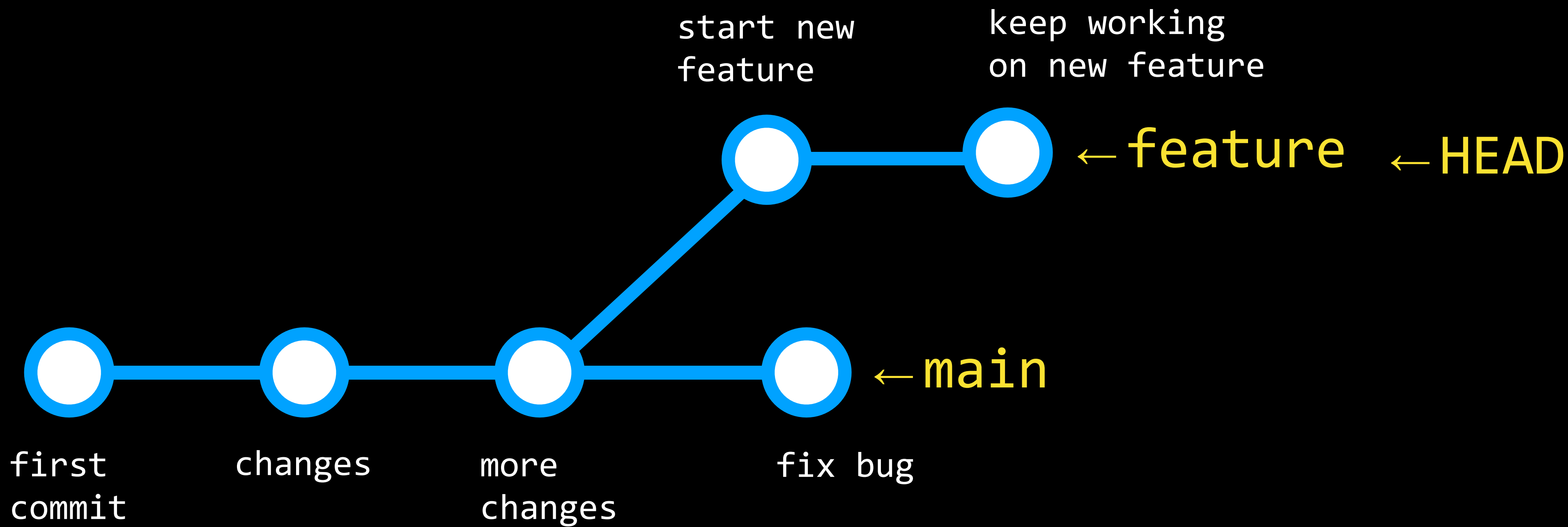


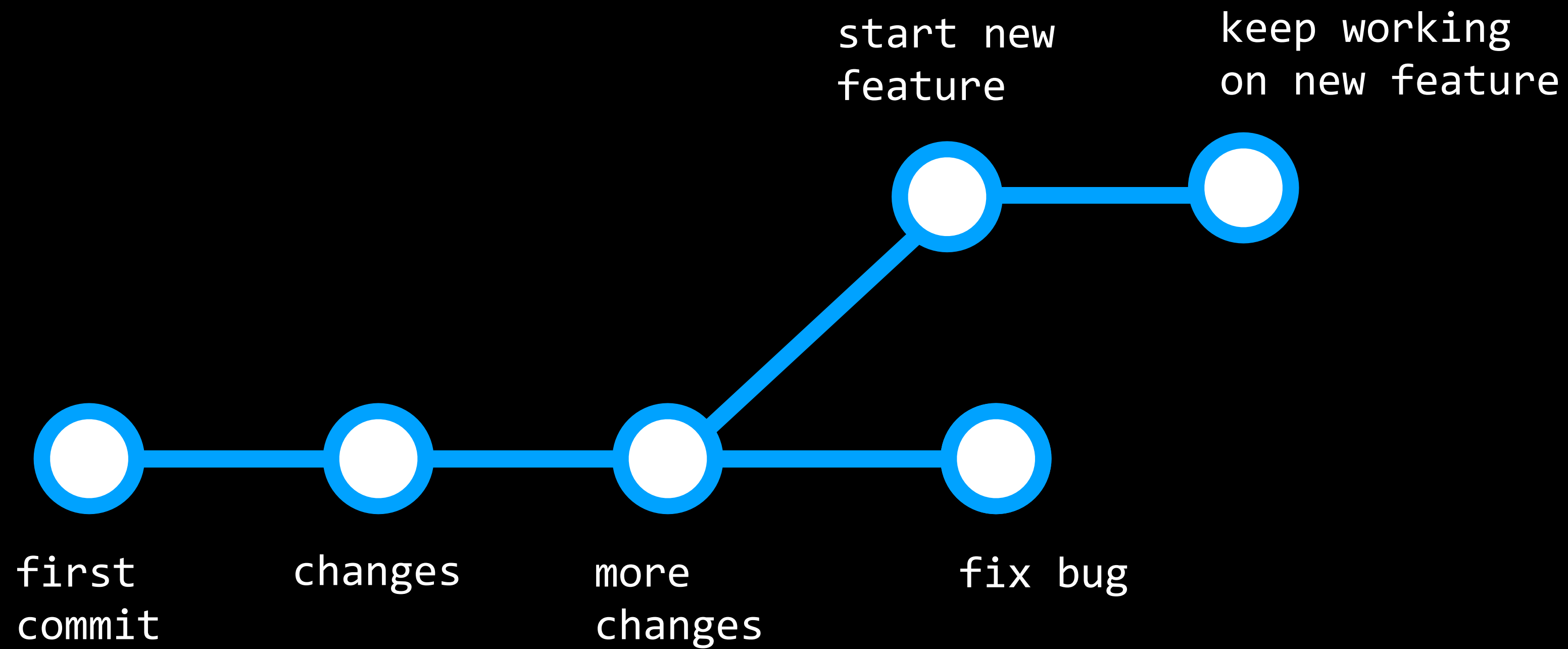


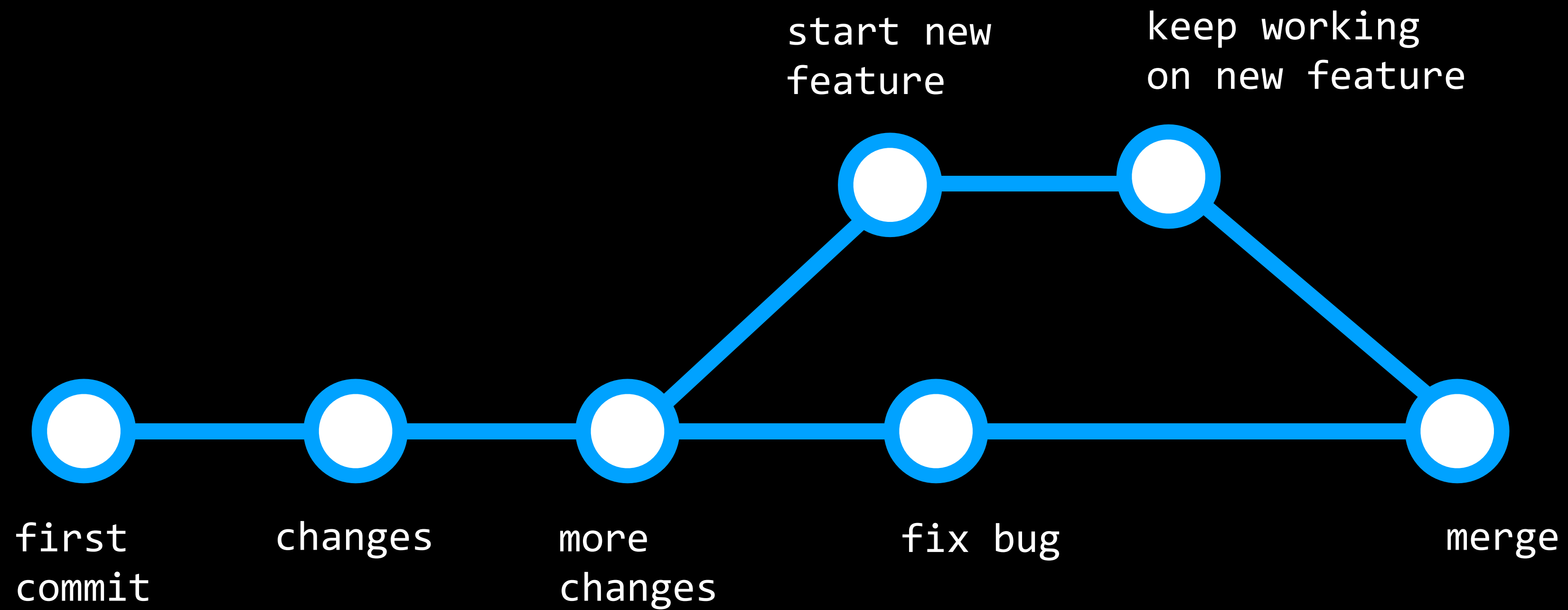












# Branching

- `git branch`
- `git checkout`
- `git merge`

# Lizenz und Quellen

[1] Diese Präsentation ist lizenziert unter einer Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. (CC BY-NC-SA 4.0)[Brian Yu], [2020]. Link zur Lizenz:  
<https://cs50.harvard.edu/web/2020/license/>

[2] <https://www.wired.com/2012/02/github-2/>

[3] <https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F>