# Git

Eine kurze Einführung

TU Chemnitz
Professur Softwaretechnik

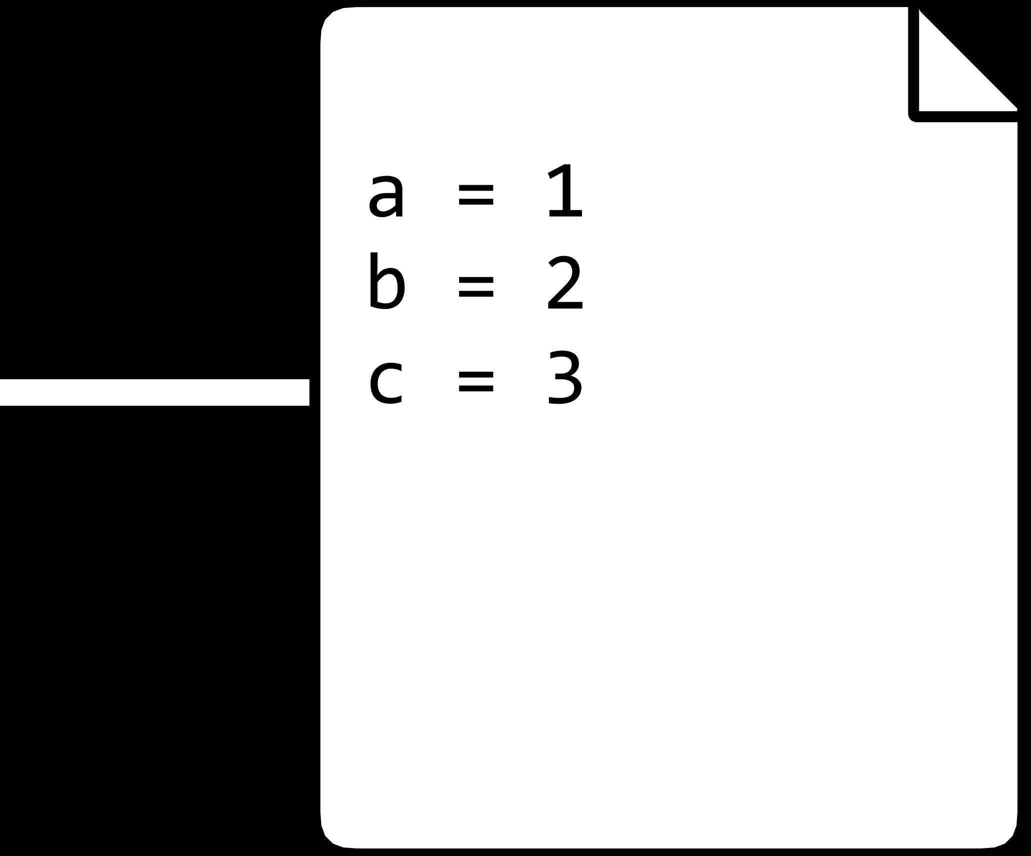
Dominik Gorgosch

## 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.”

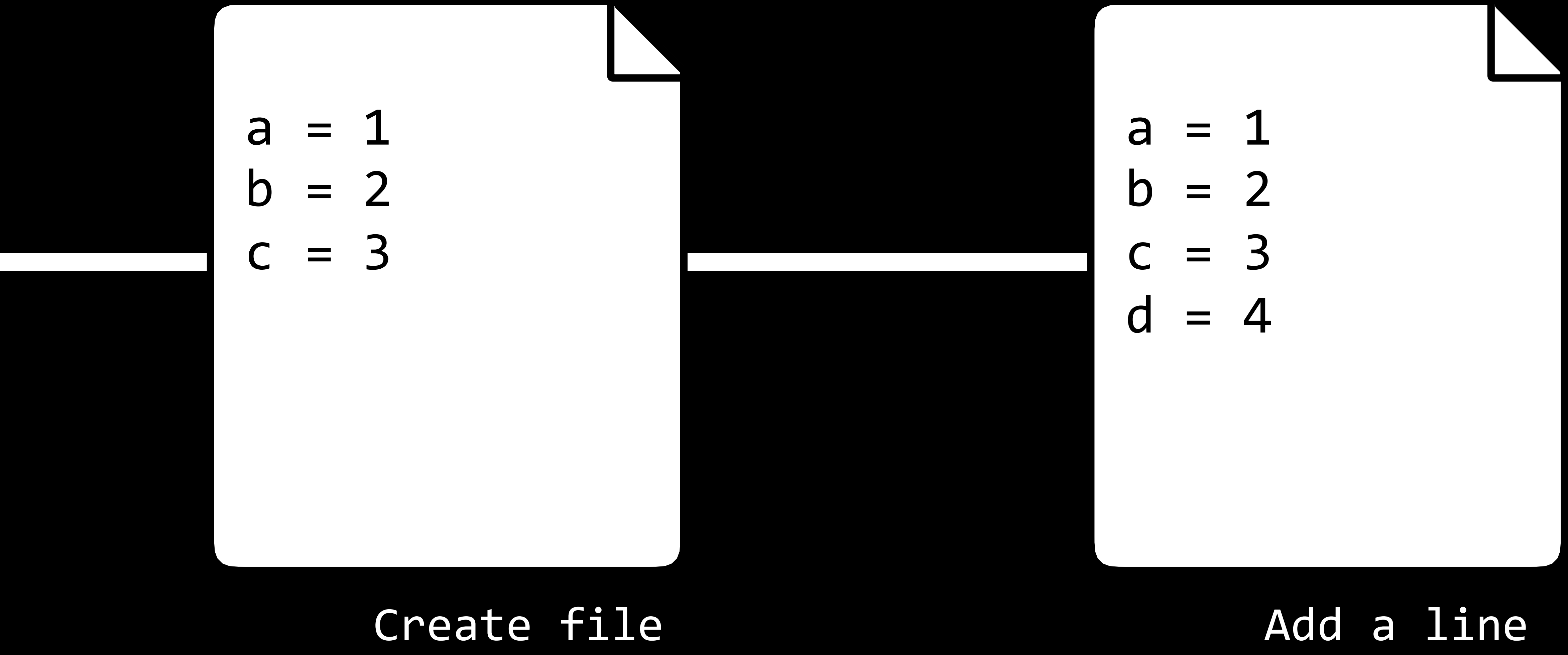
## Keep track of changes to code.



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

Create file

## Keep track of changes to code.

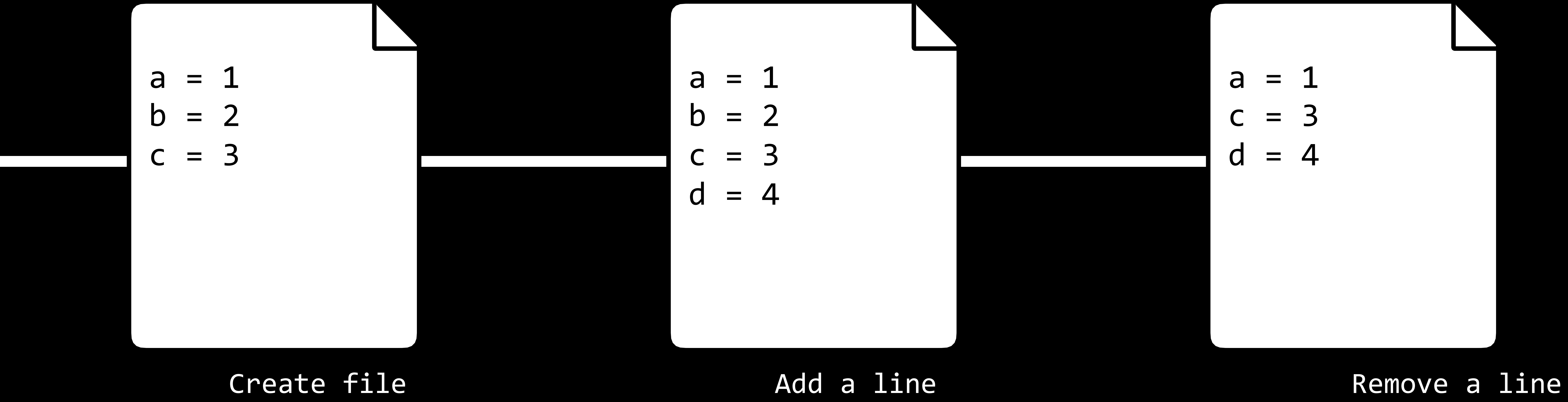


```
a = 1
b = 2
c = 3
```

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

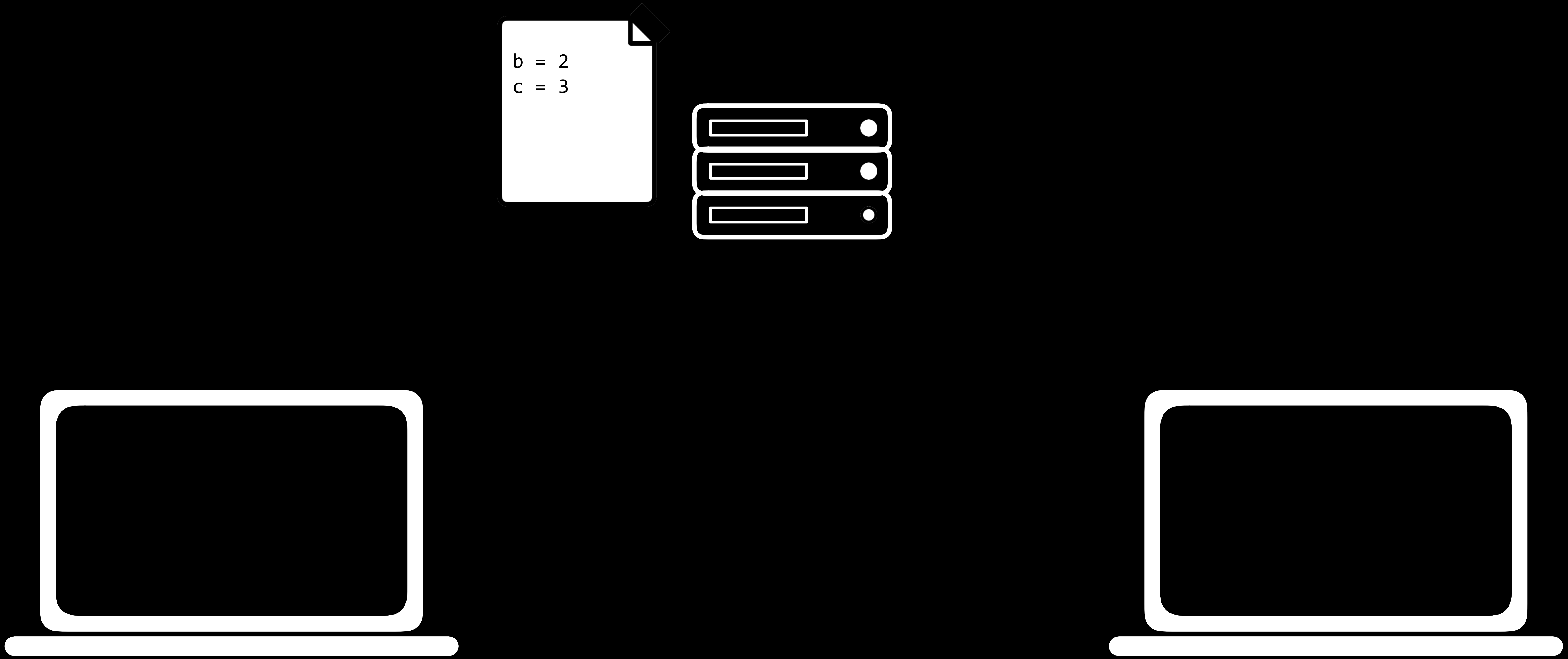
[1]

## Keep track of changes to code.

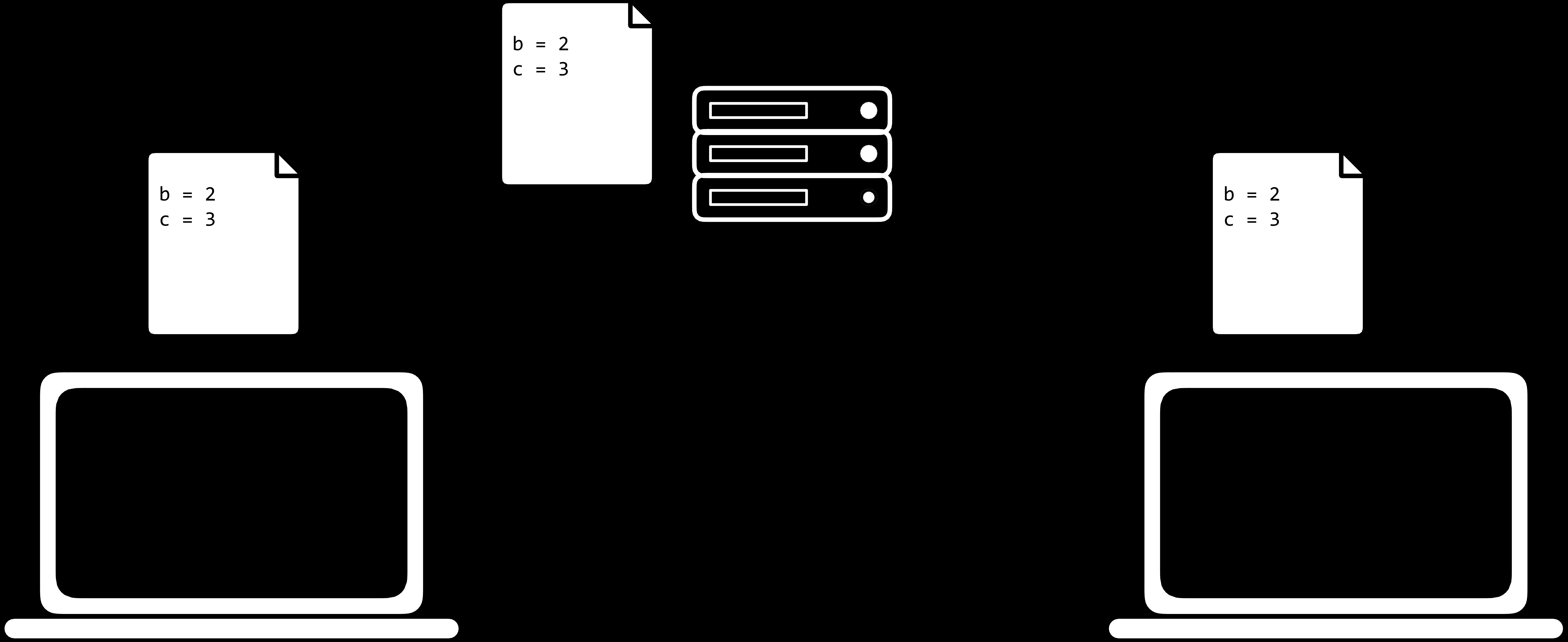


[1]

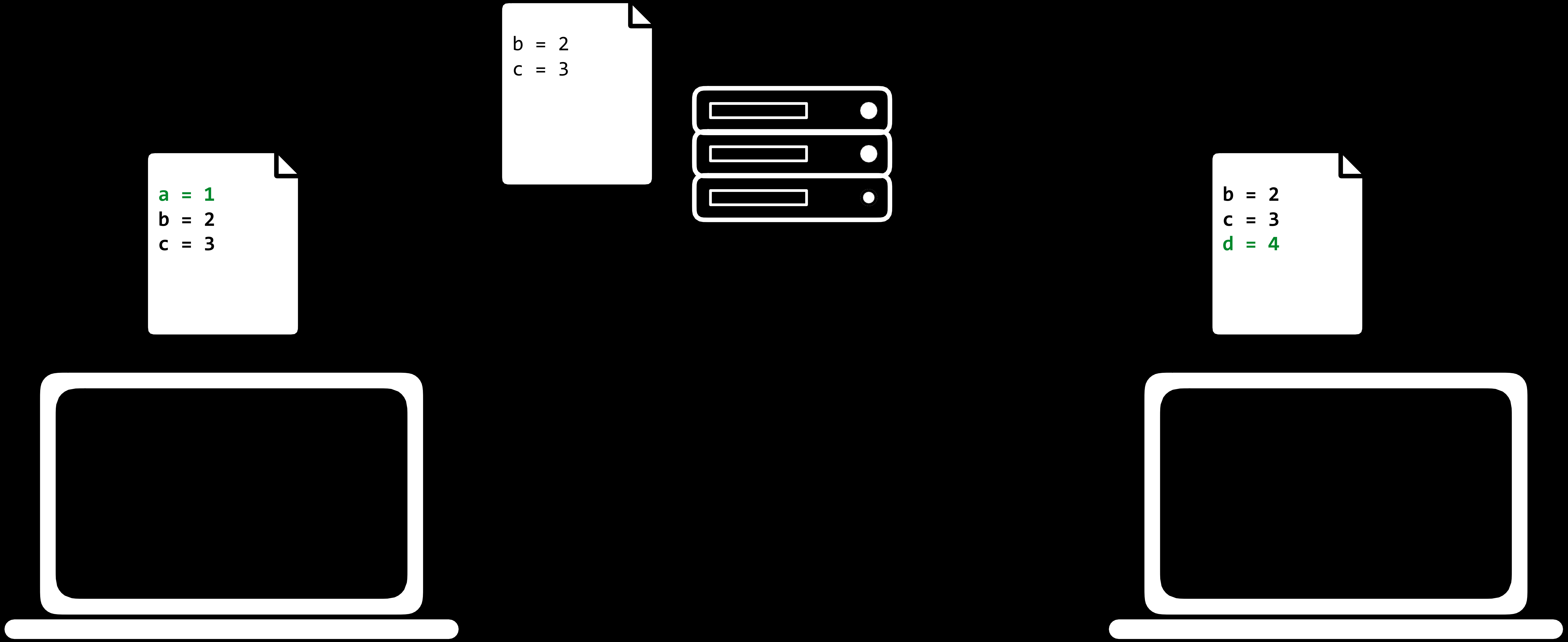
## Synchronizes code between different people.



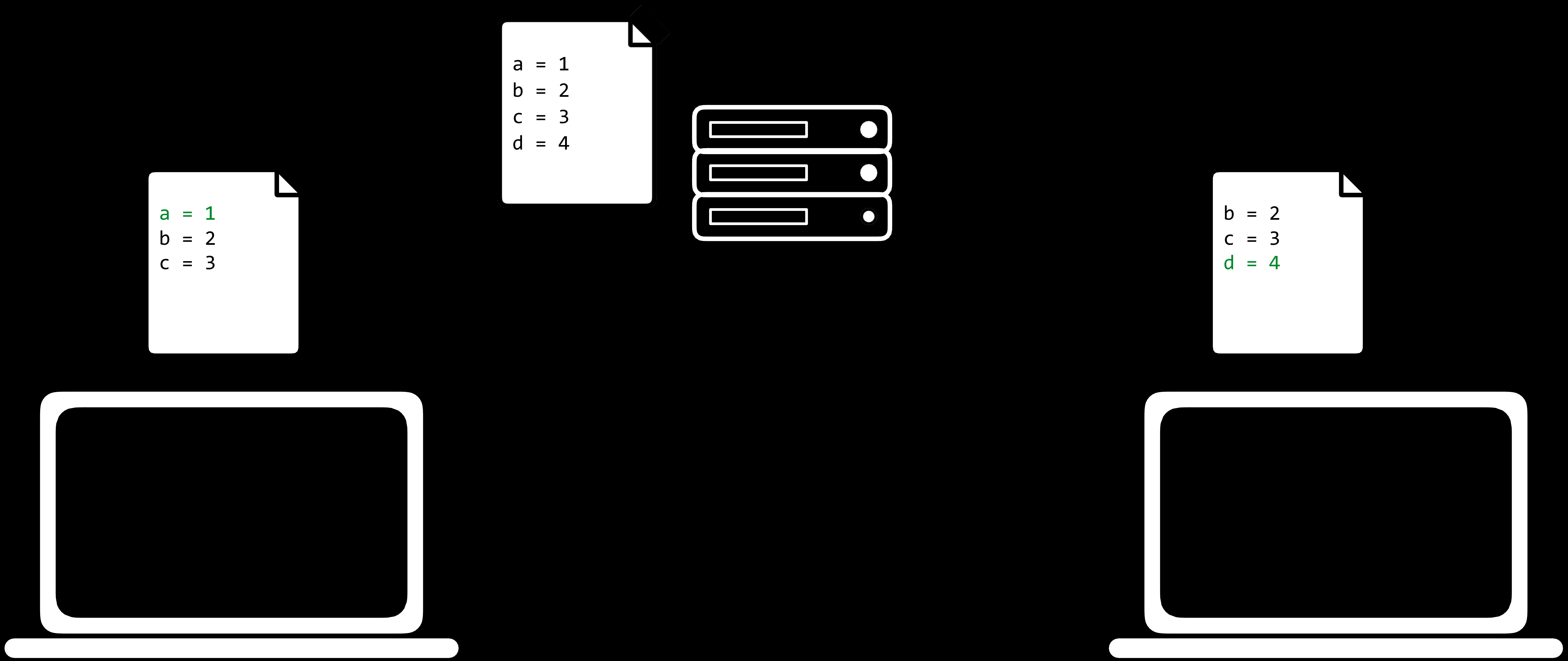
## Synchronizes code between different people.



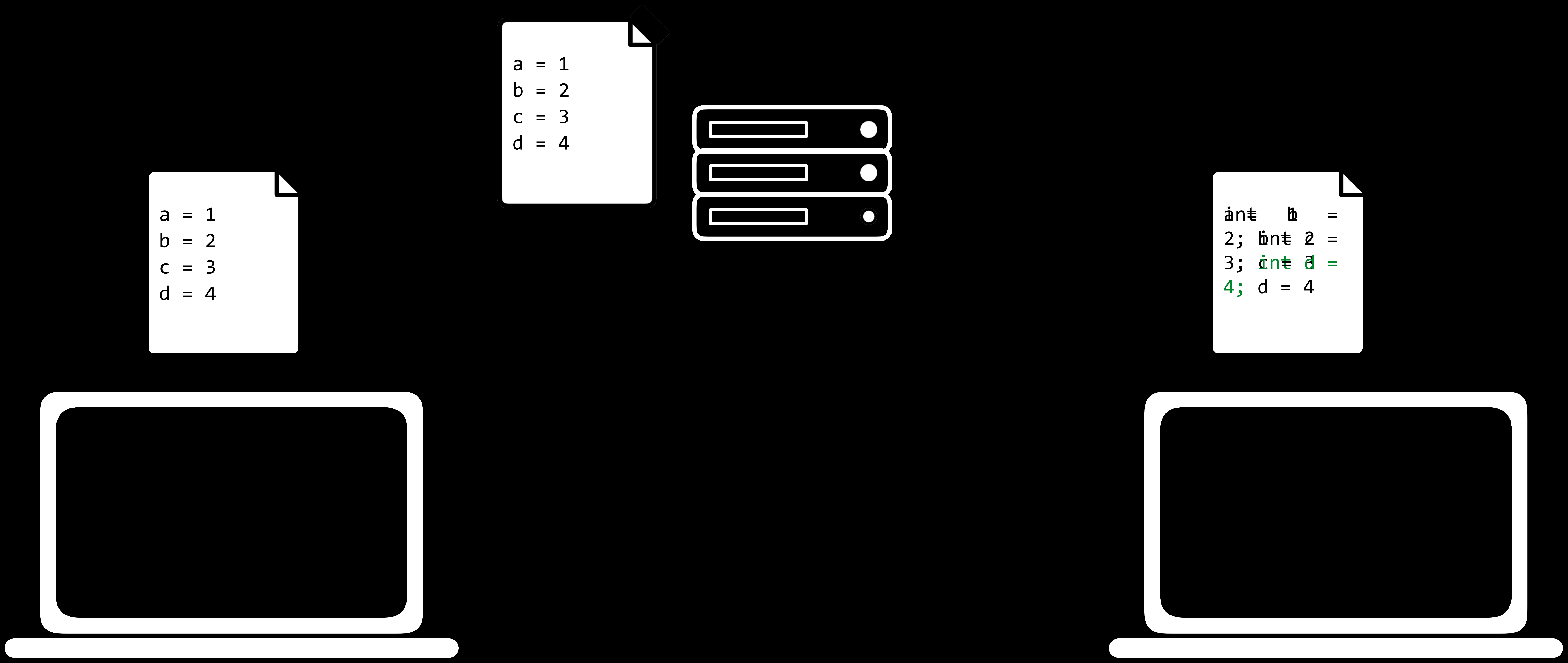
## Synchronizes code between different people.



## Synchronizes code between different people.



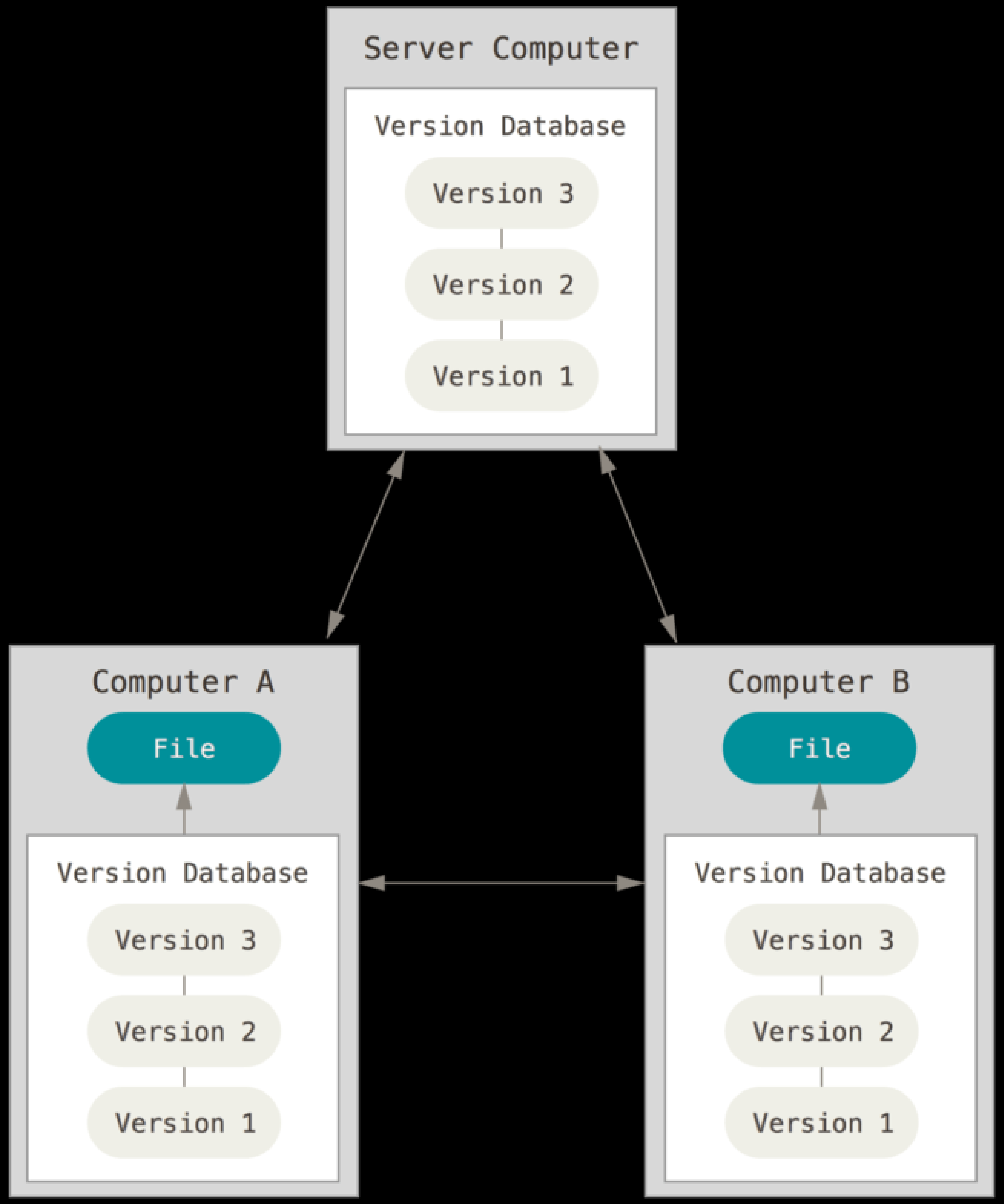
## Synchronizes code between different people.



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

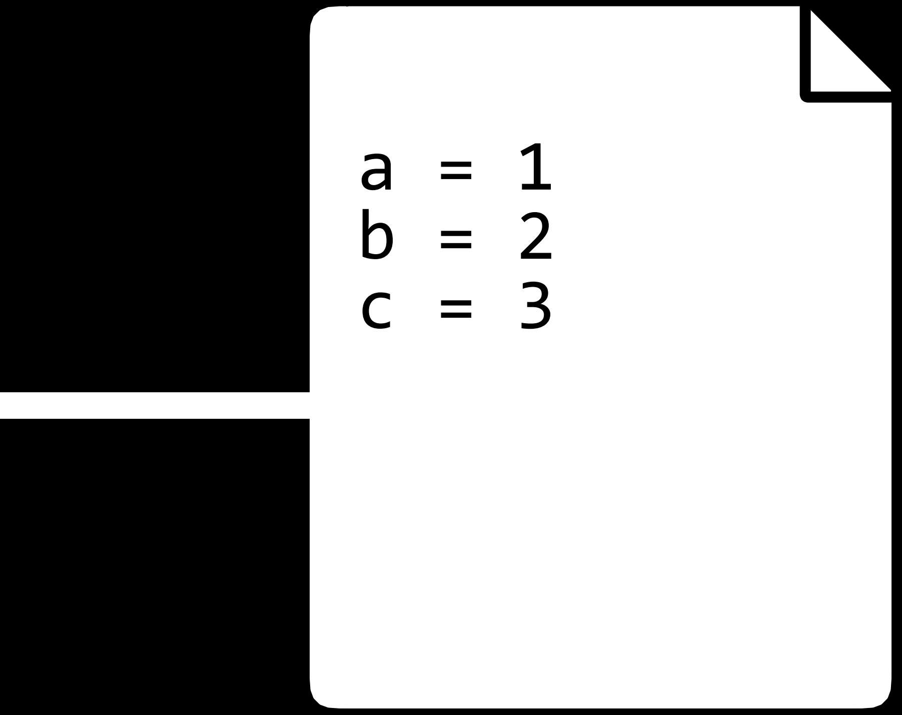
```
a = 1
b = 2
c = 3
d = 4
```

```
int  b  =
2; int 2 =
3; int d =
4; d = 4
```


Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

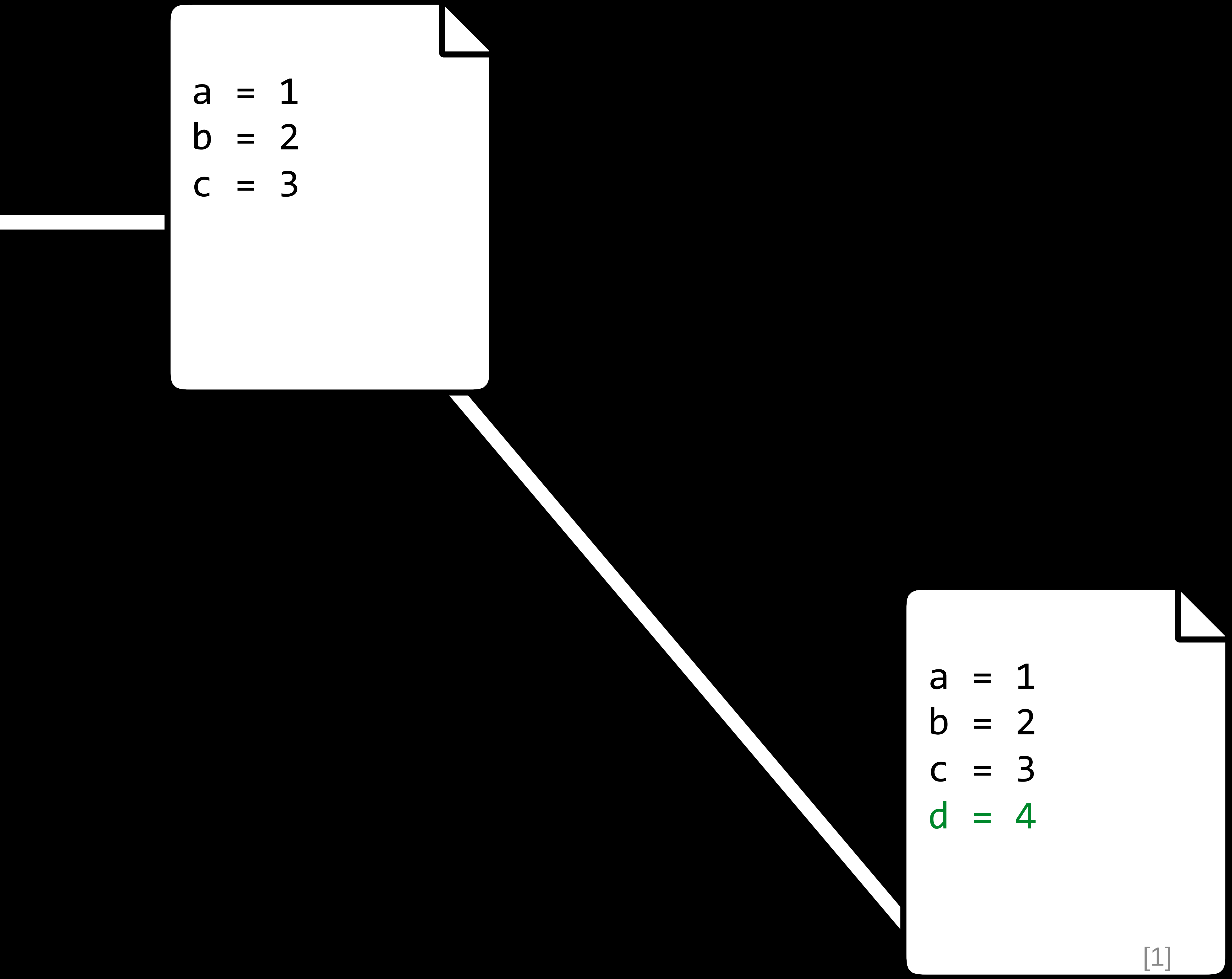
[3]

## Test changes to code without losing the original.

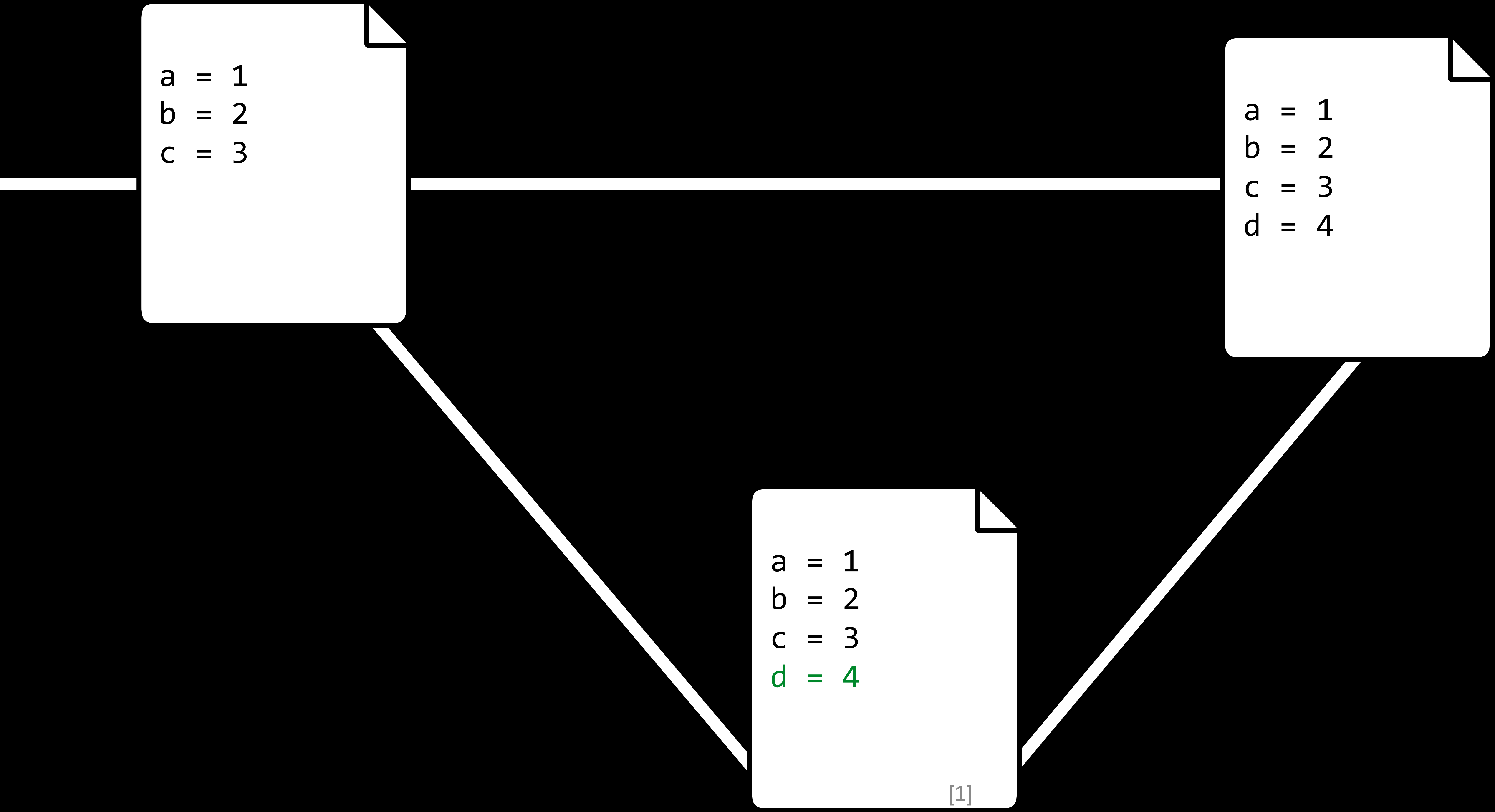


```
a = 1
b = 2
c = 3
```

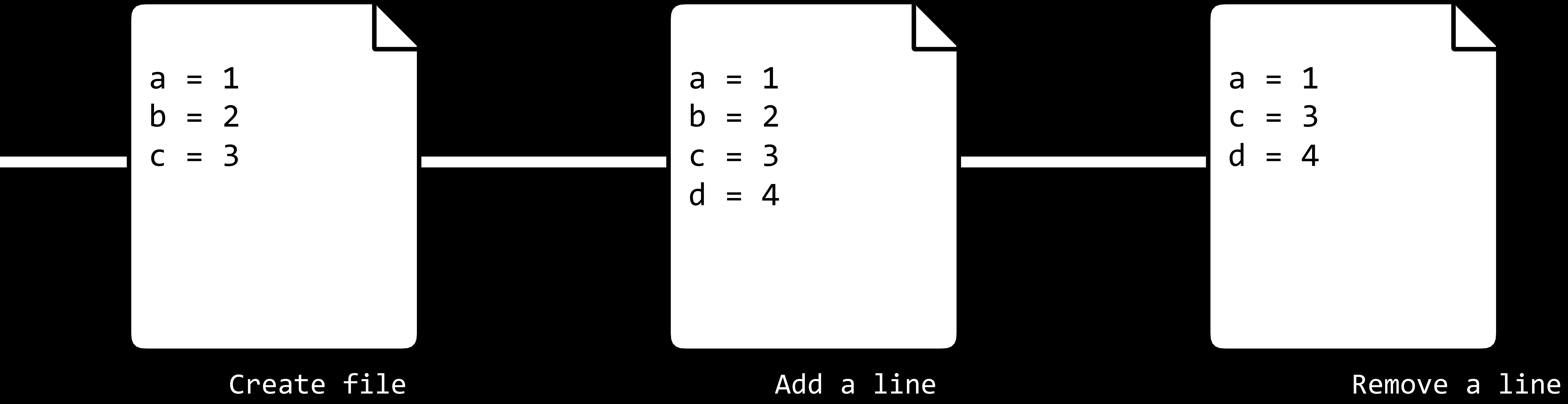
## Test changes to code without losing the original.



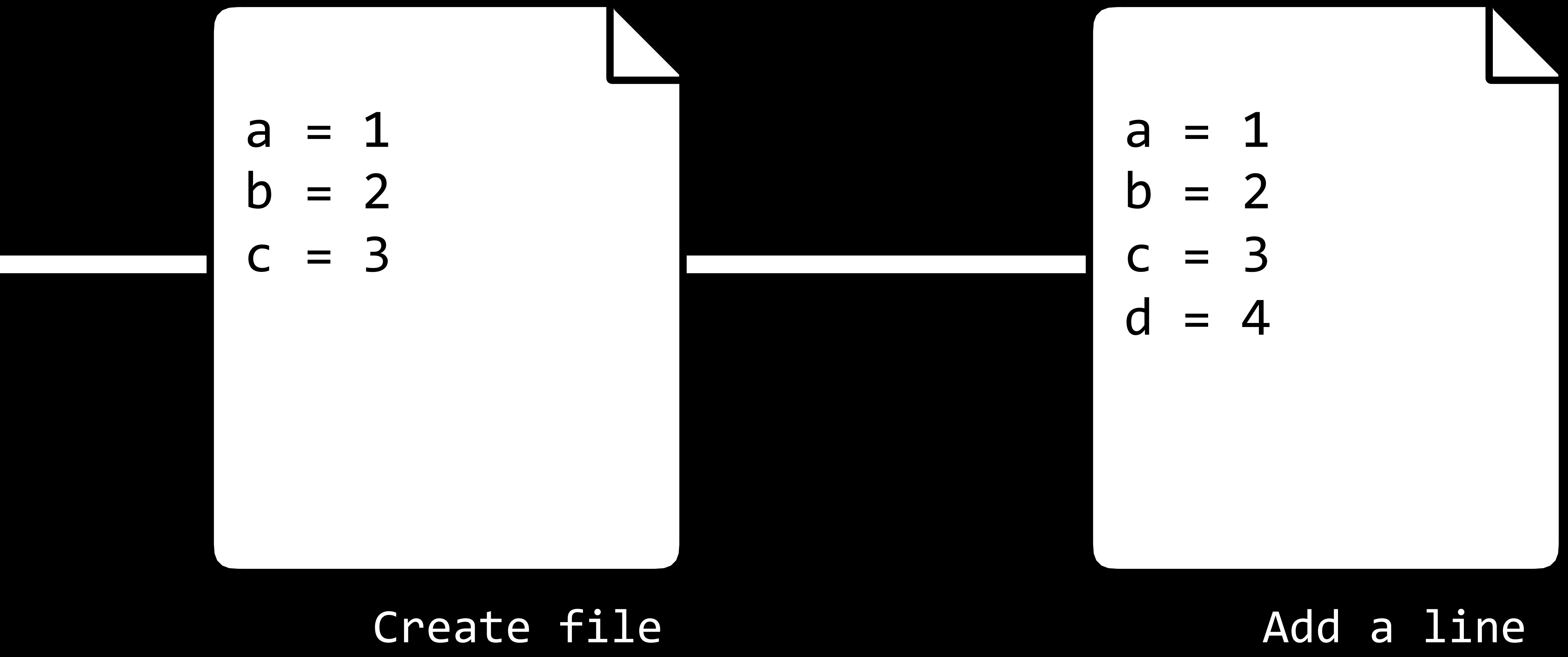
## Test changes to code without losing the original.



## Revert back to old versions of code.



## Revert back to old versions of code.


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

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

[1]

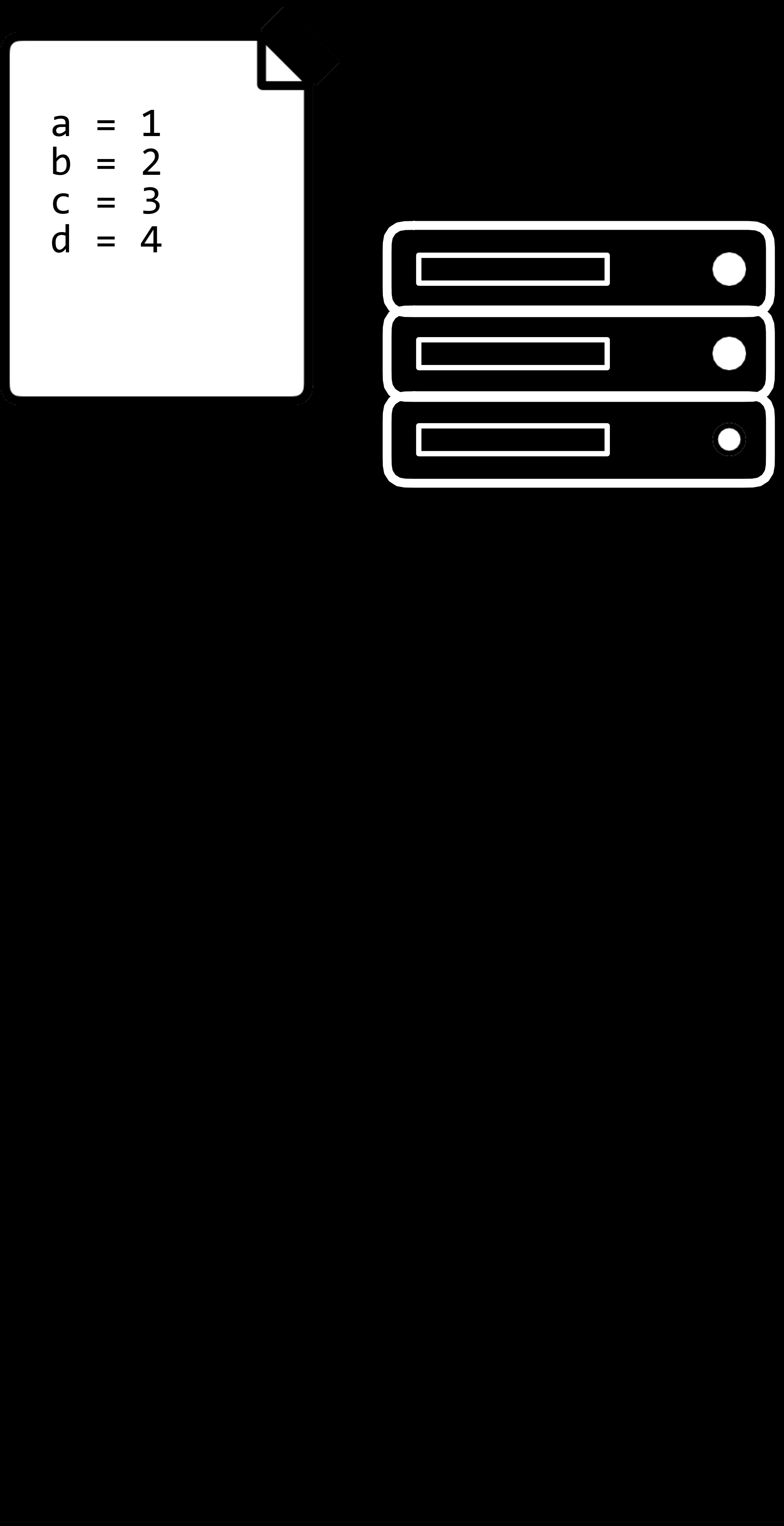
## GitHub

[1]

## `git clone`

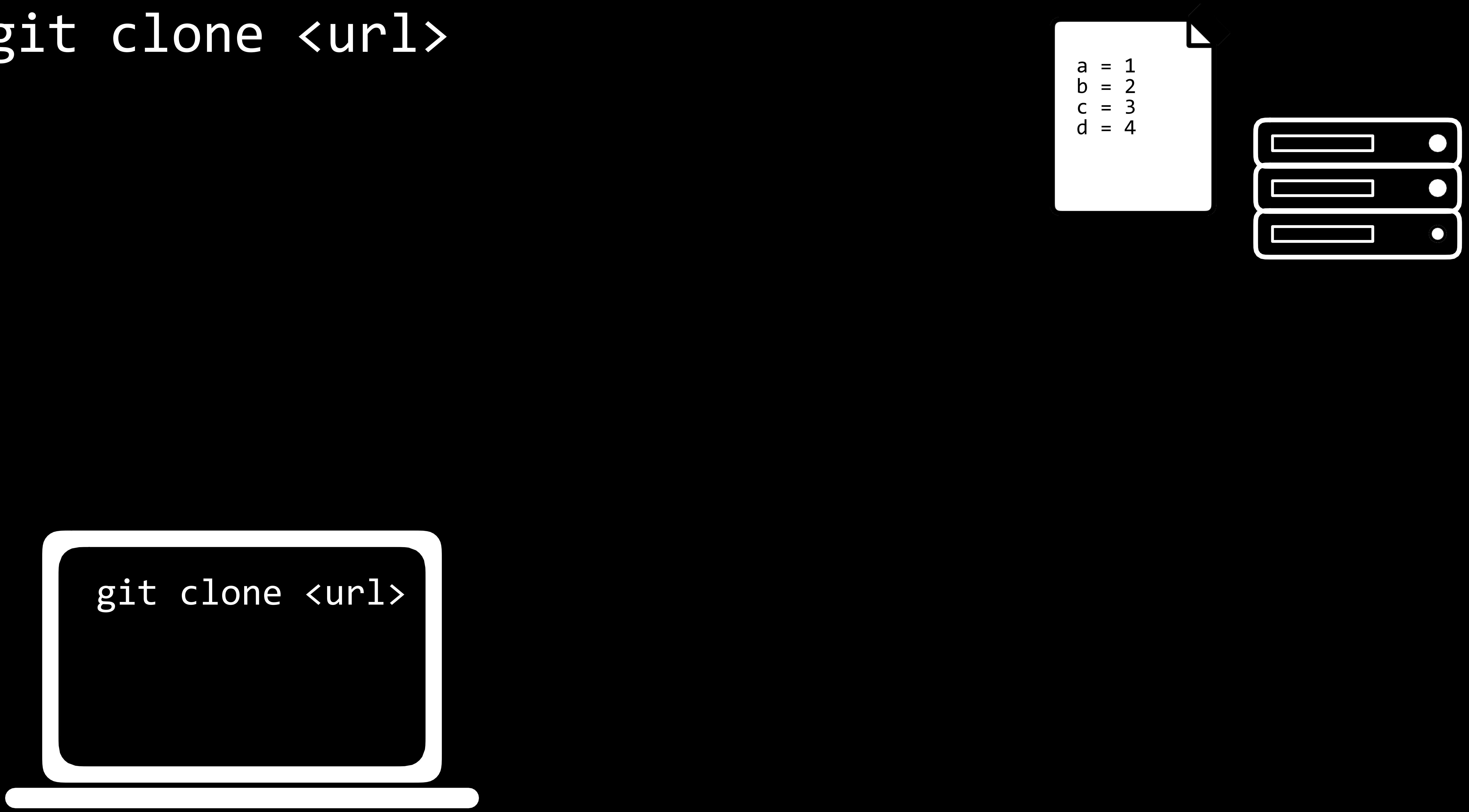
[1]

```
git clone <url>
```



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

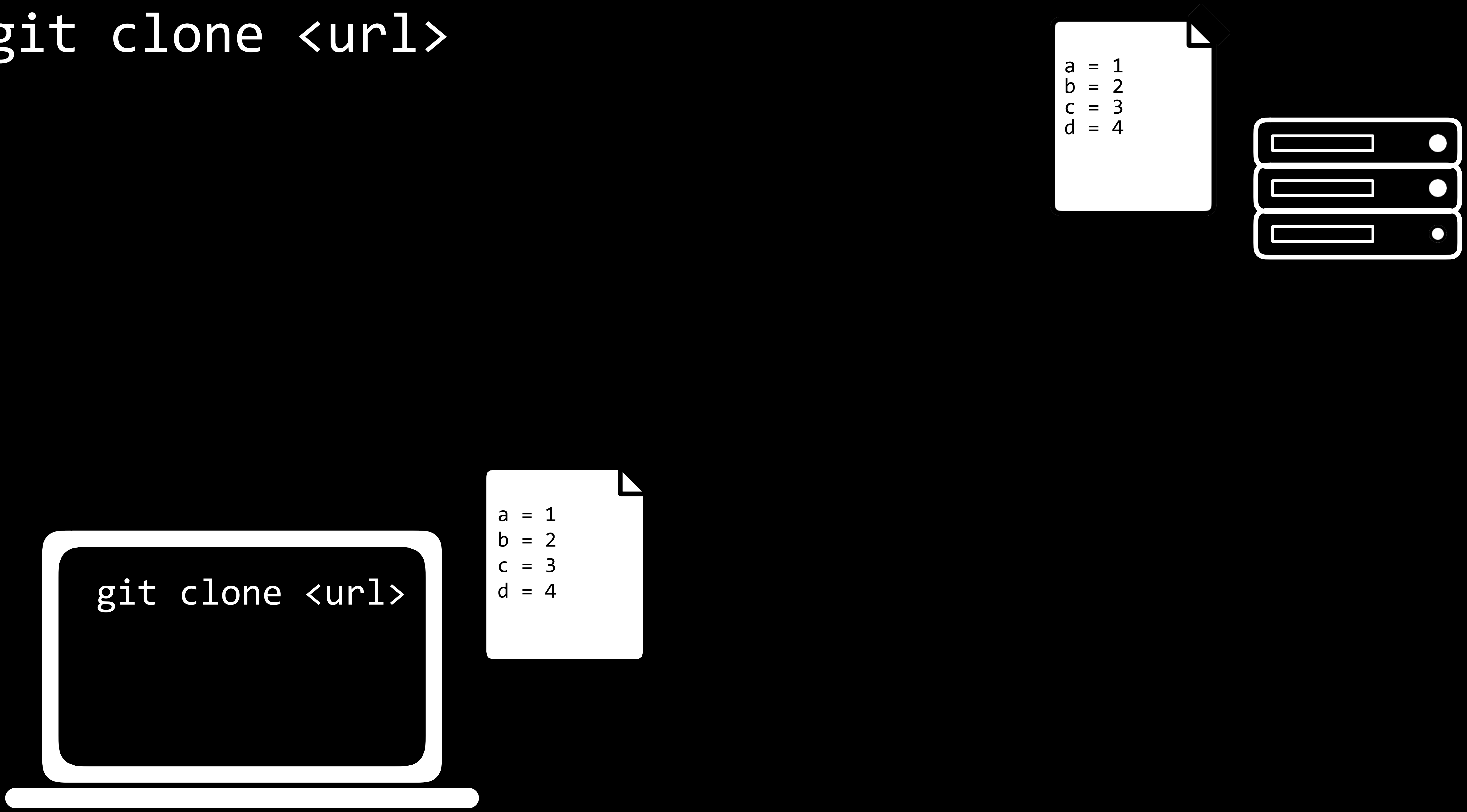
# git clone <url>



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

```
git clone <url>
```

## git clone <url>



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

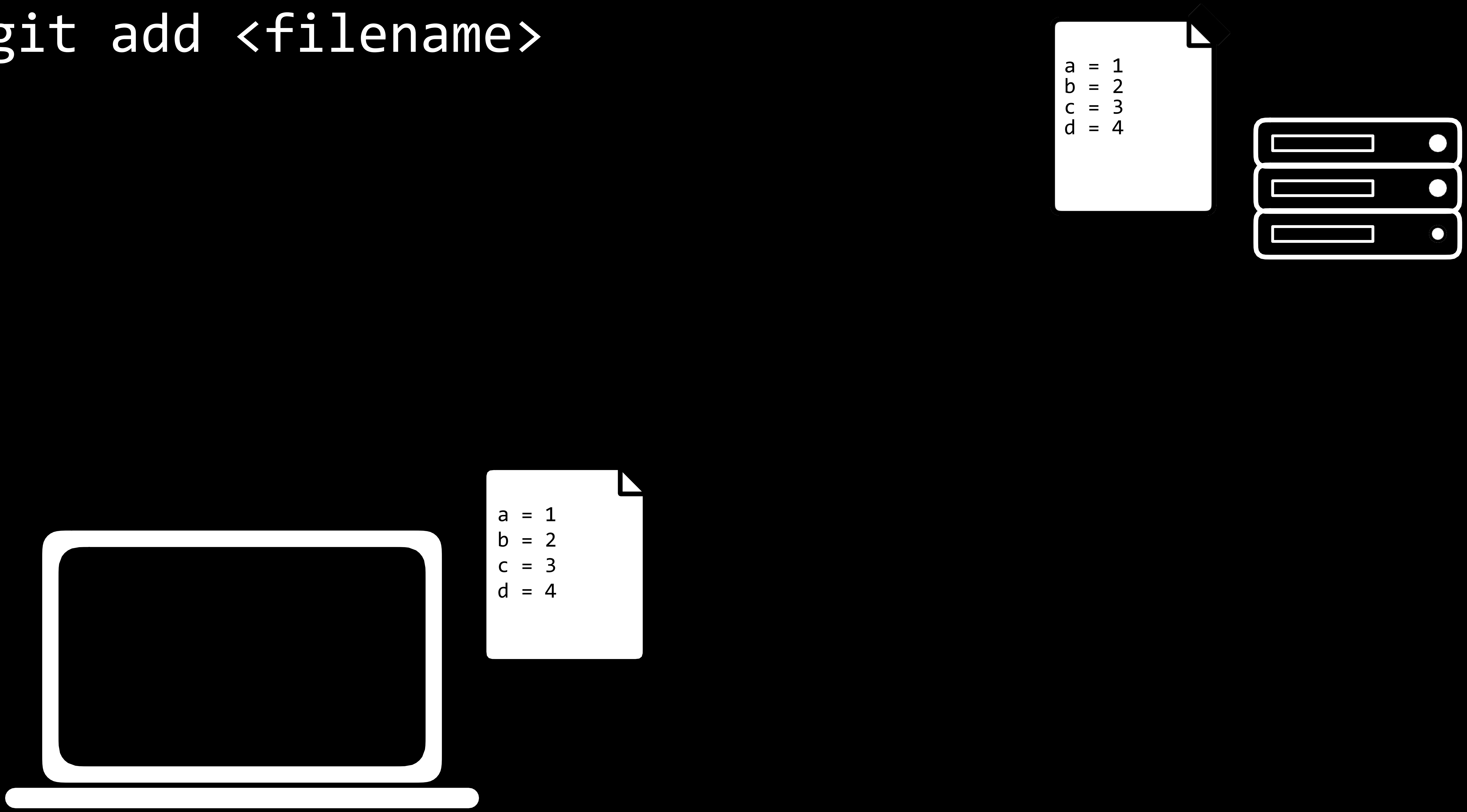
```
git clone <url>
```

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

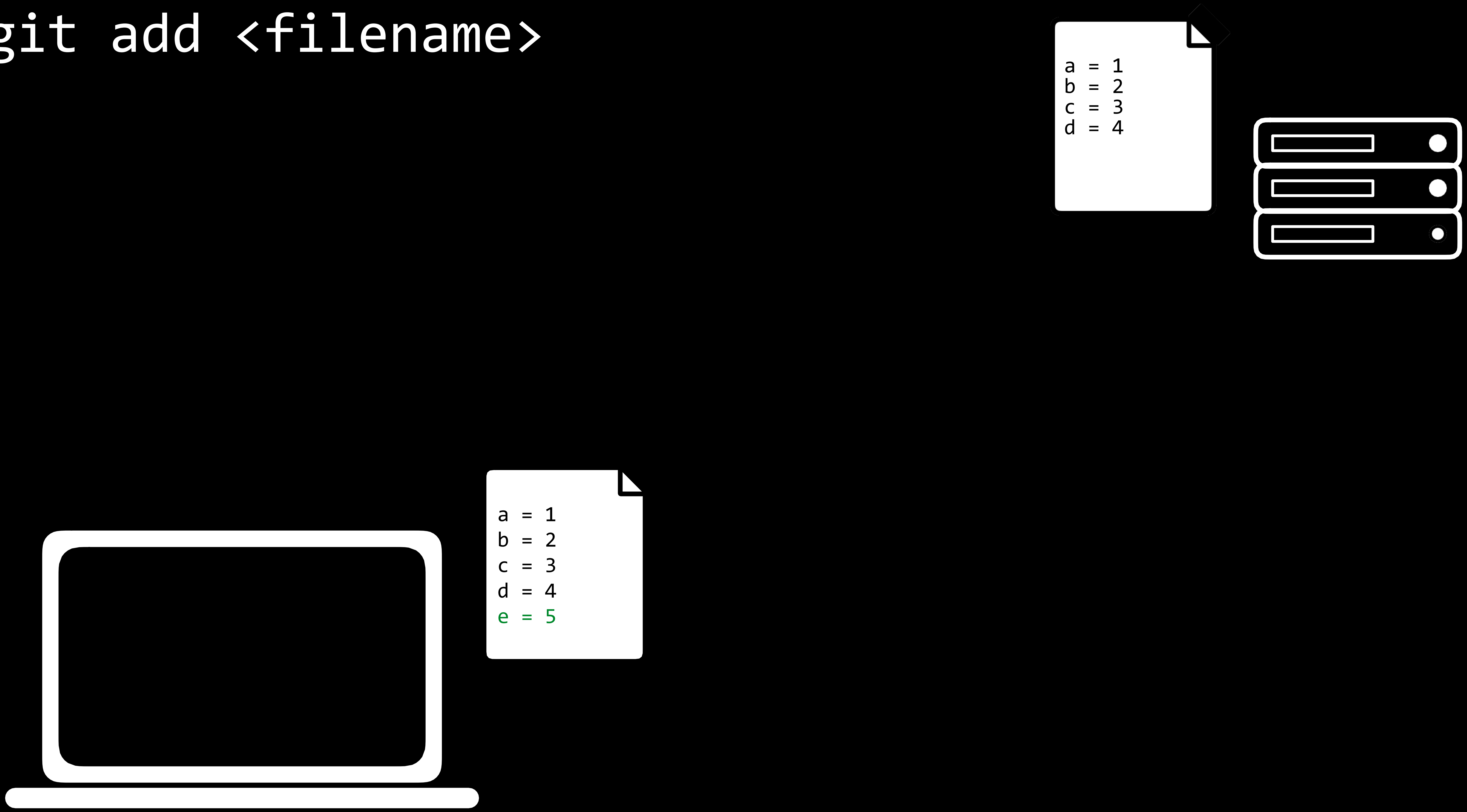
## git add

[1]

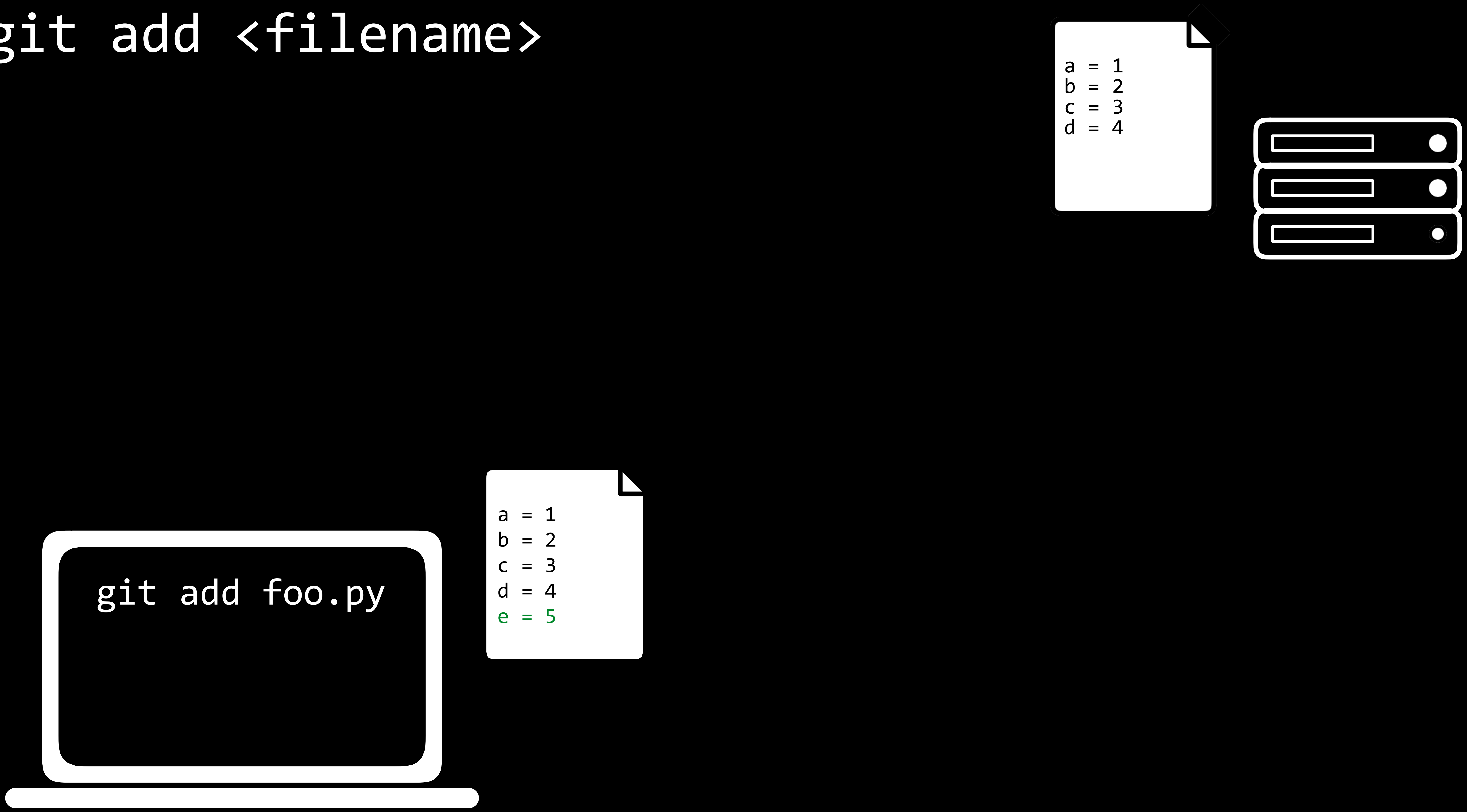
## git add <filename>



## git add <filename>



## git add <filename>

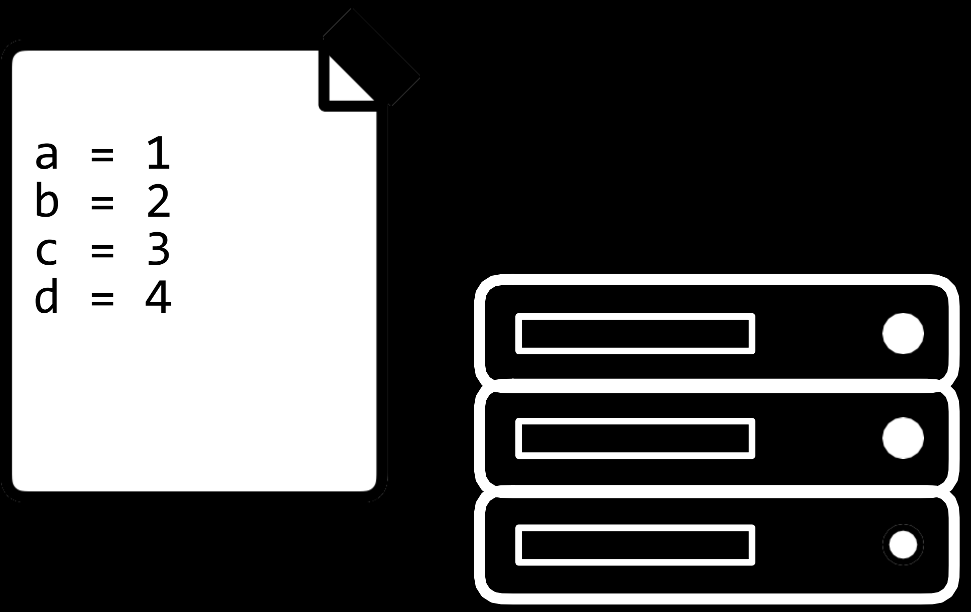


```
a = 1
b = 2
c = 3
d = 4
```

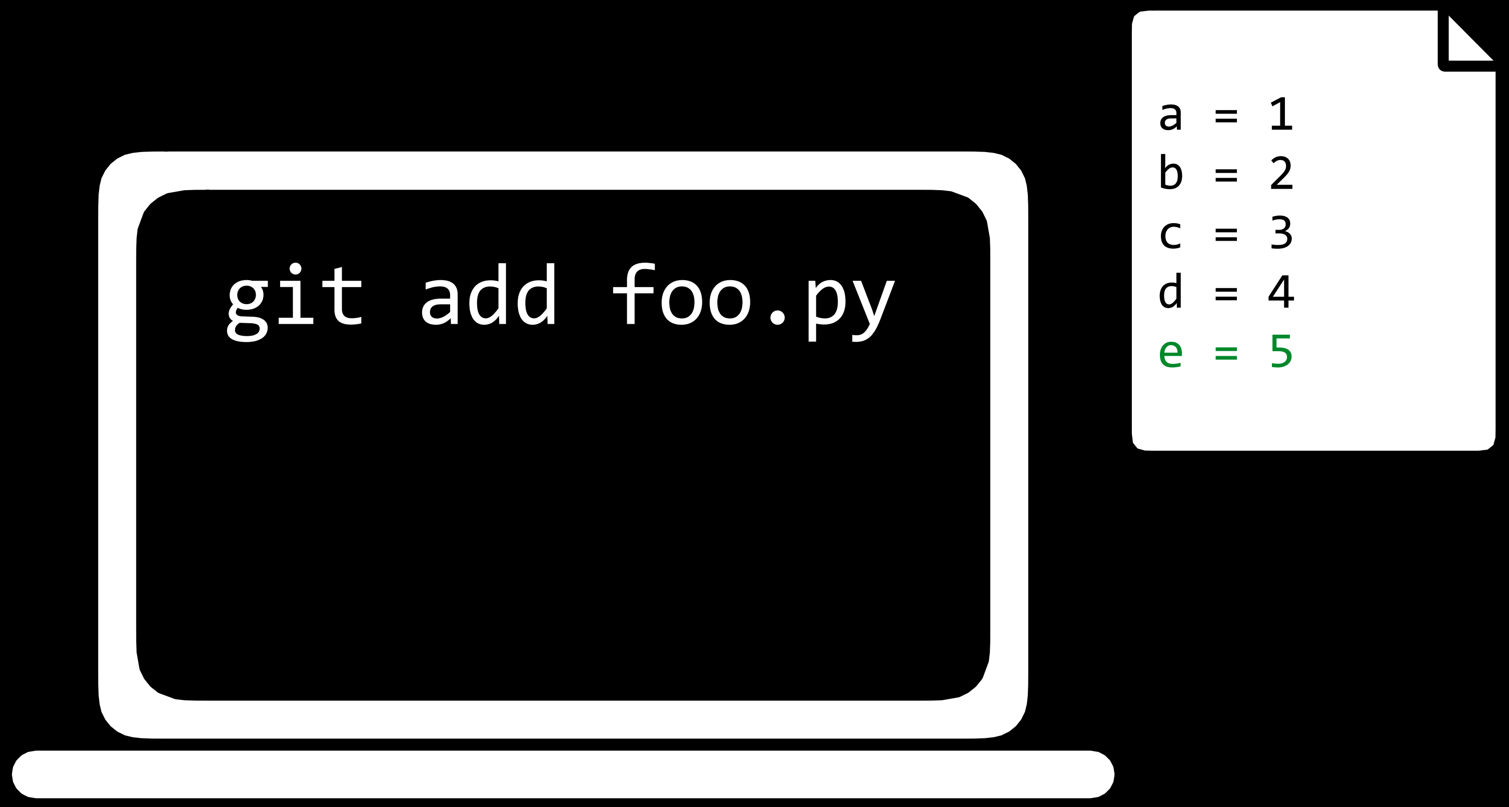
```
git add foo.py
```

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

## git add <filename>



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



```
git add foo.py
```

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

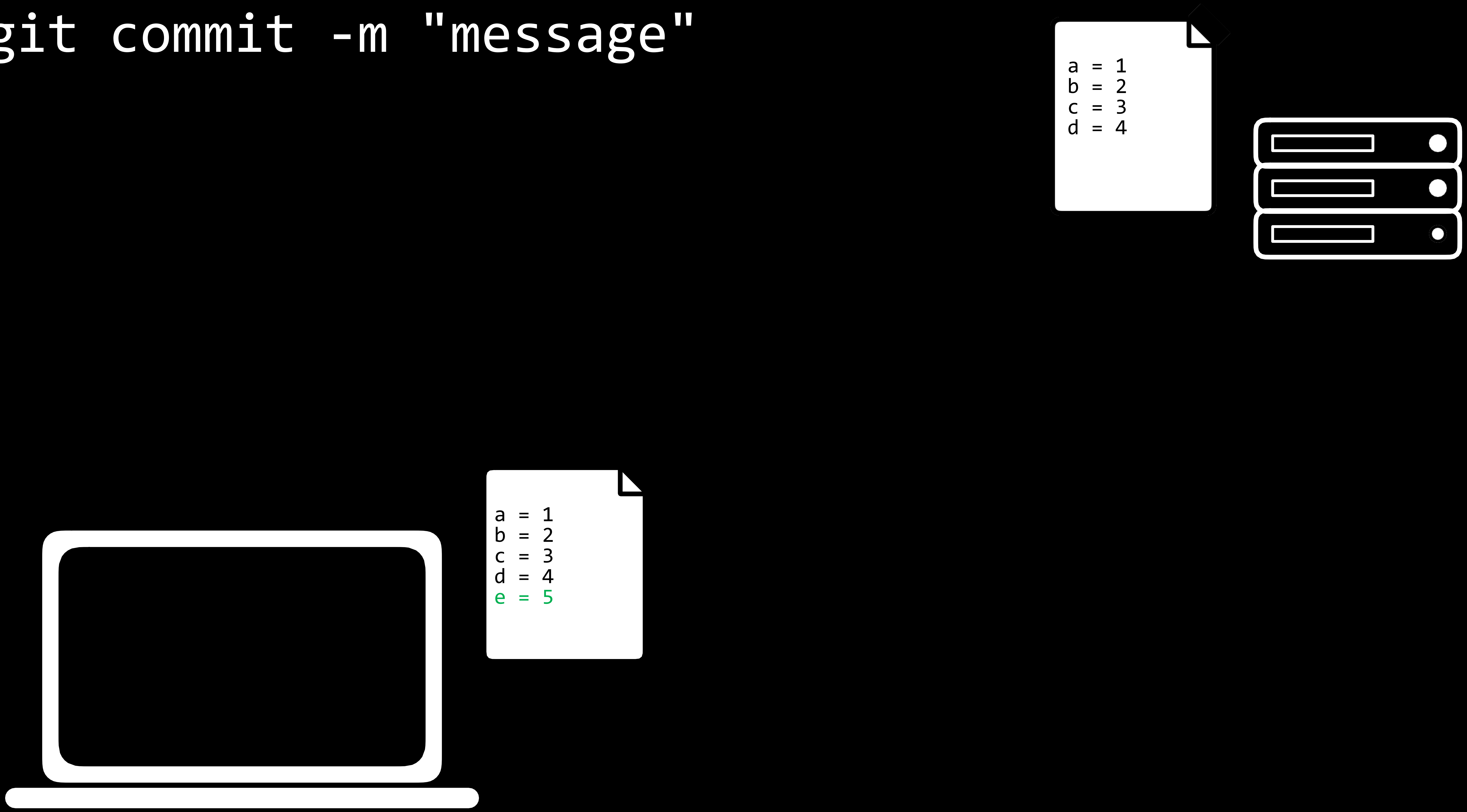
```
Changes to be committed:

        modified: foo.py
```

```
git commit
```

[1]

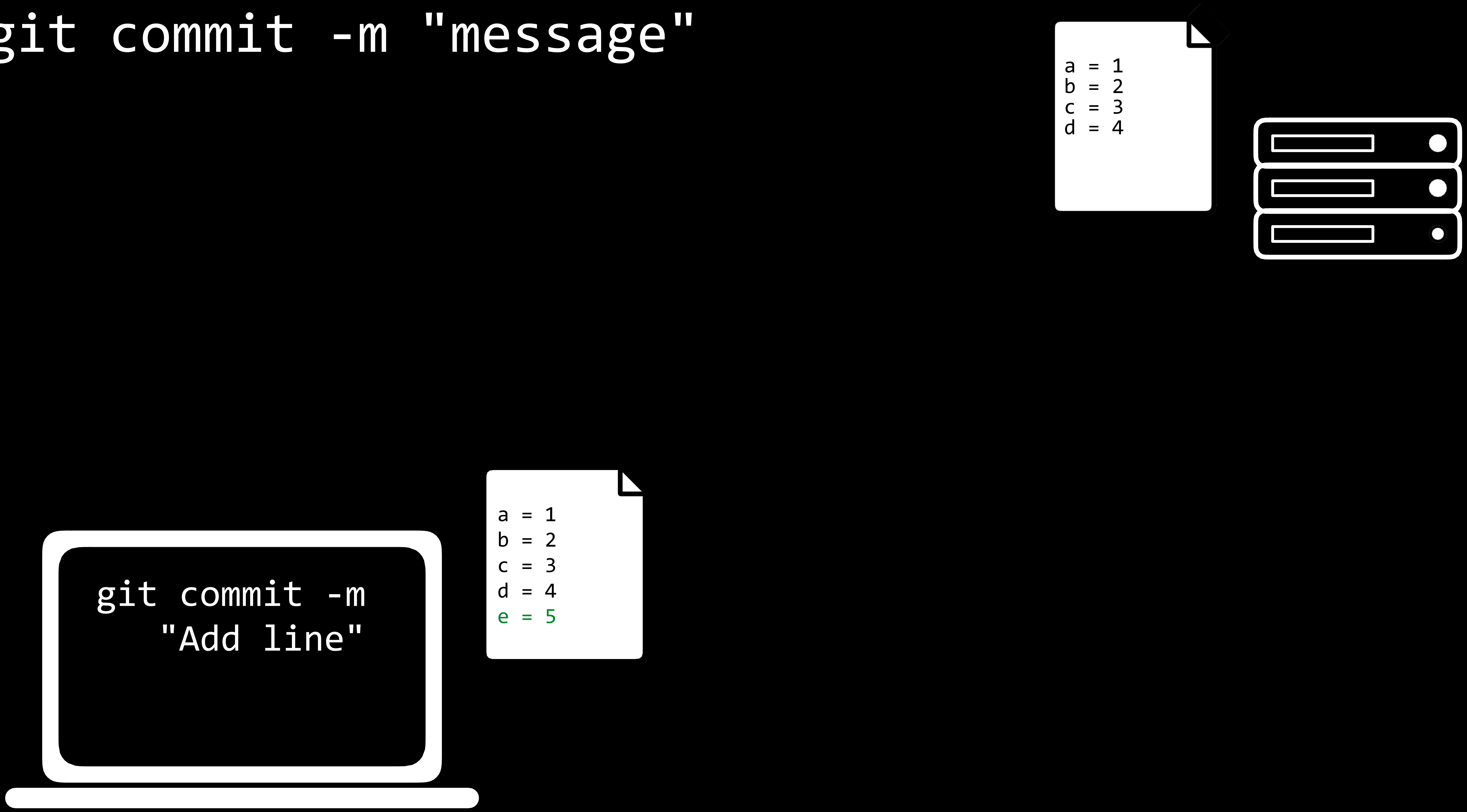
```
git commit -m "message"
```



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

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

## git commit -m "message"



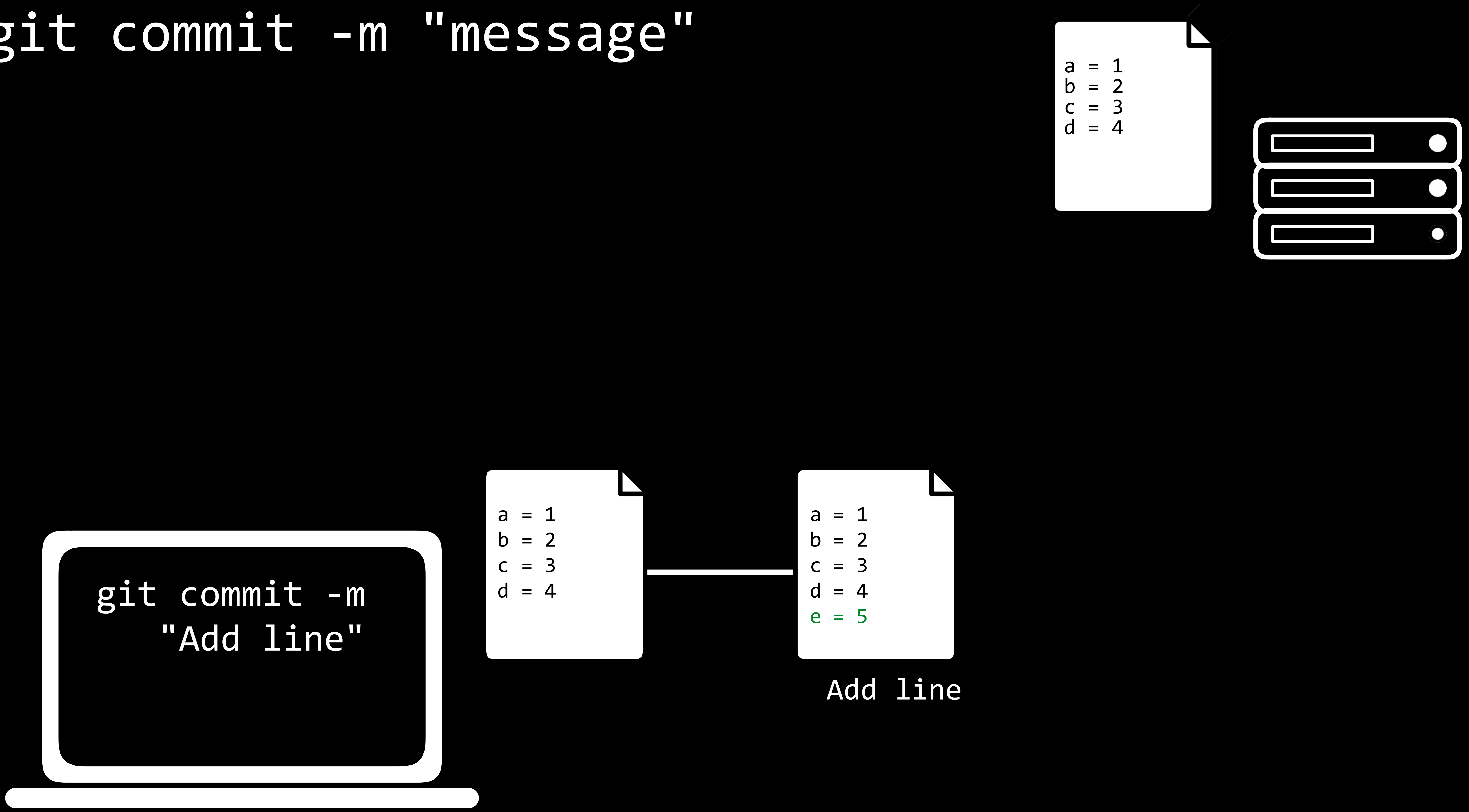
```
a = 1
b = 2
c = 3
d = 4
```

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

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

[1]

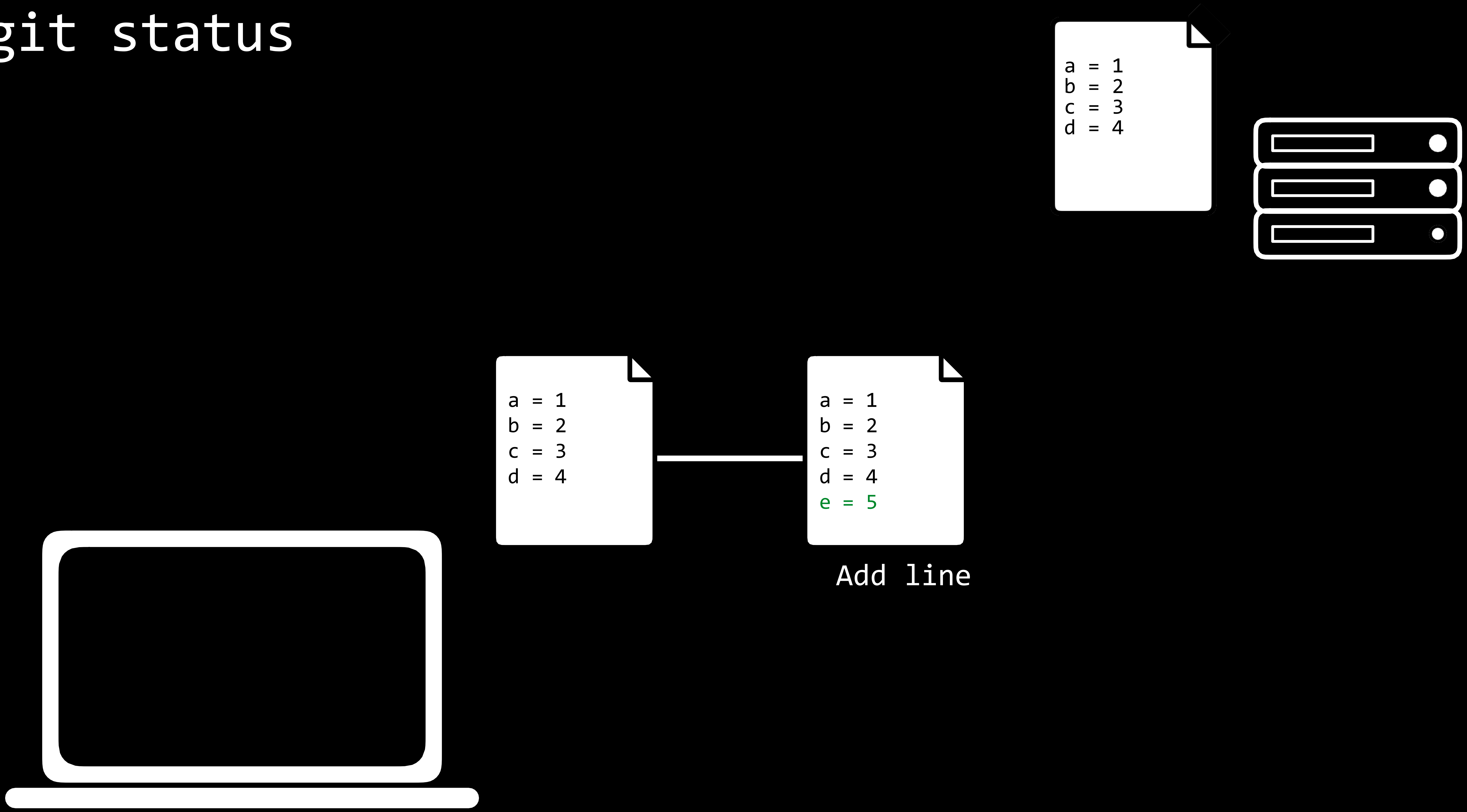
## git commit -m "message"



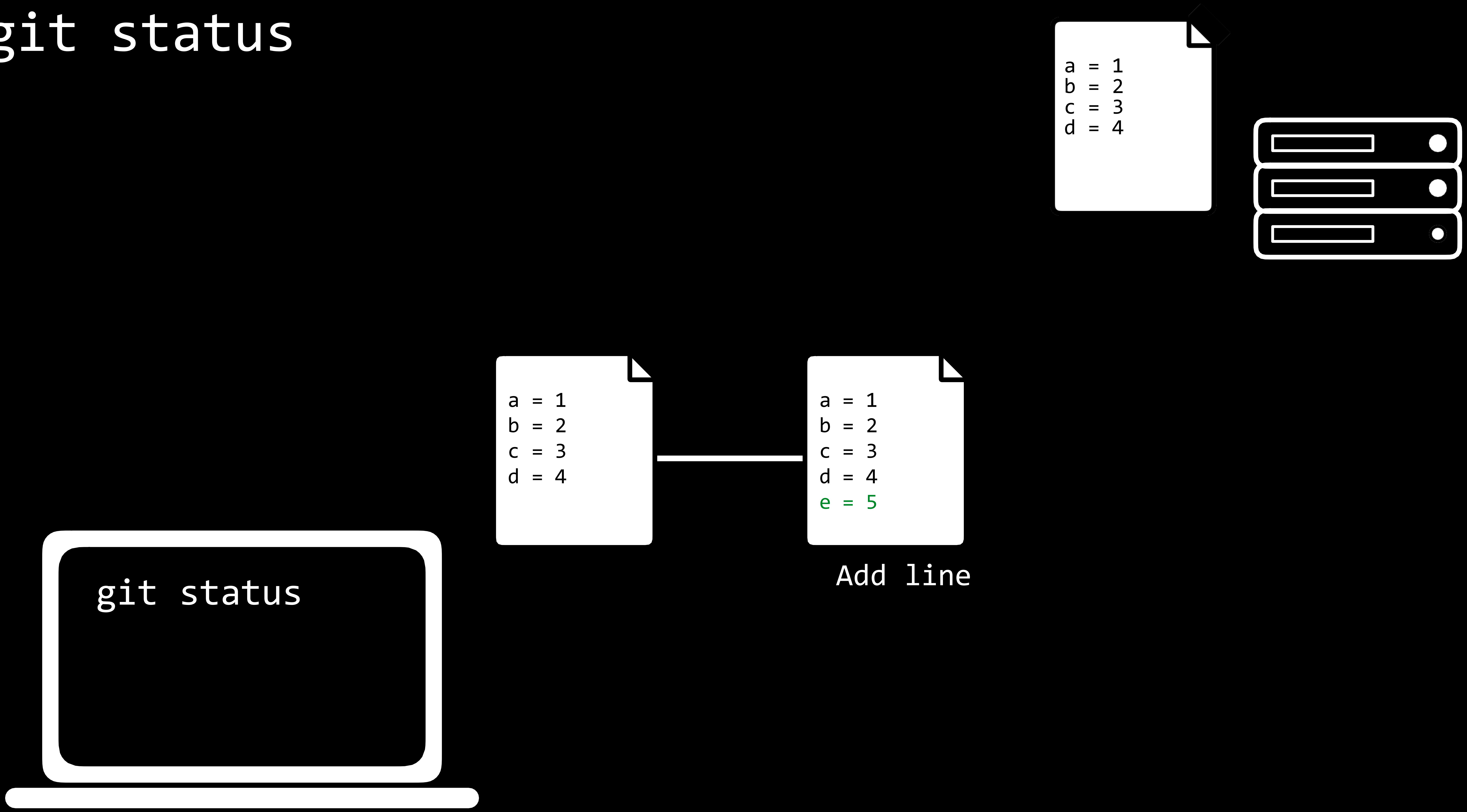
## `git status`

[1]

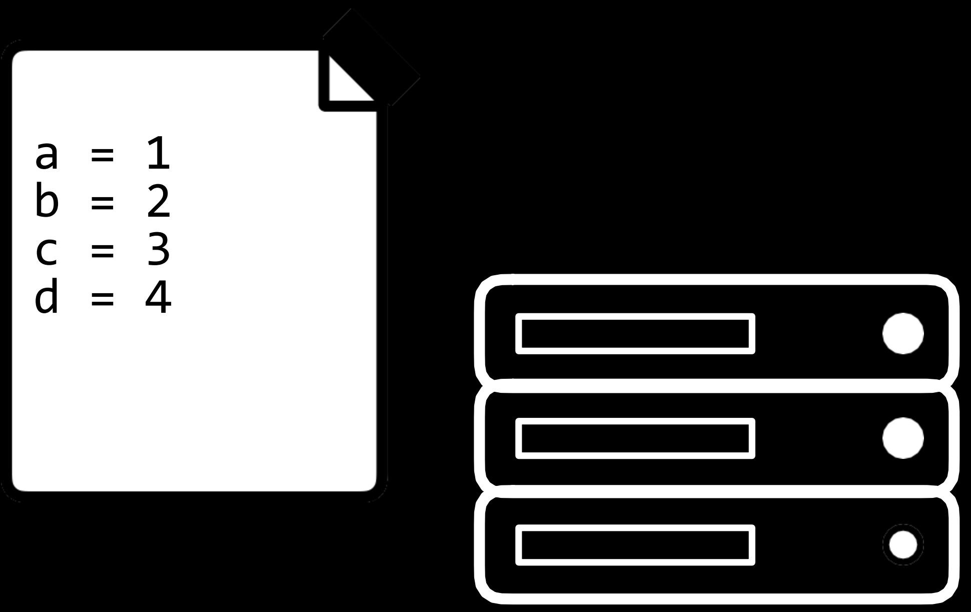
## git status

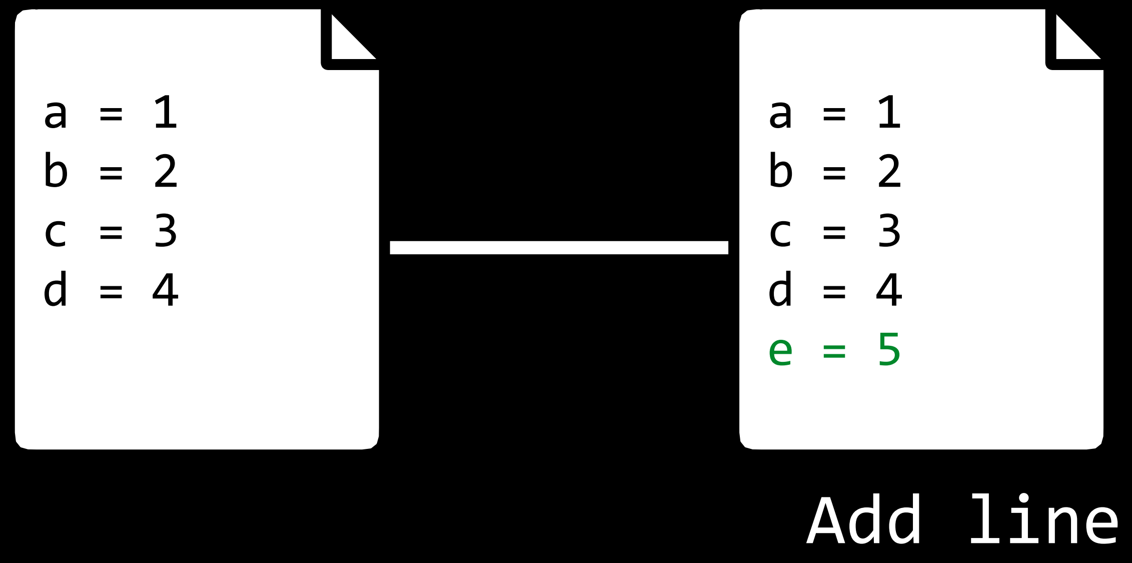


# git status



## git status





```
git status
```

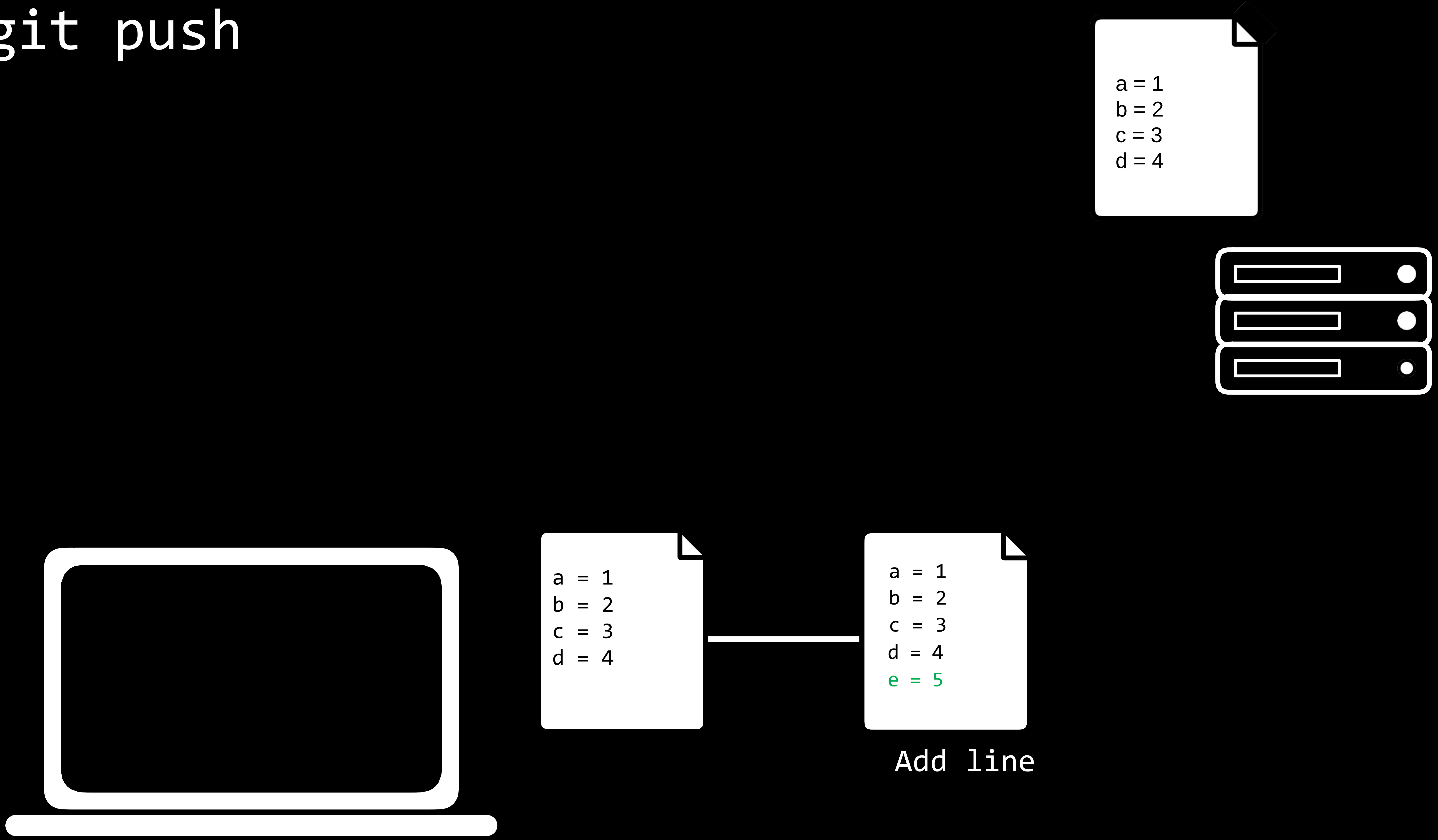
```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

[1]

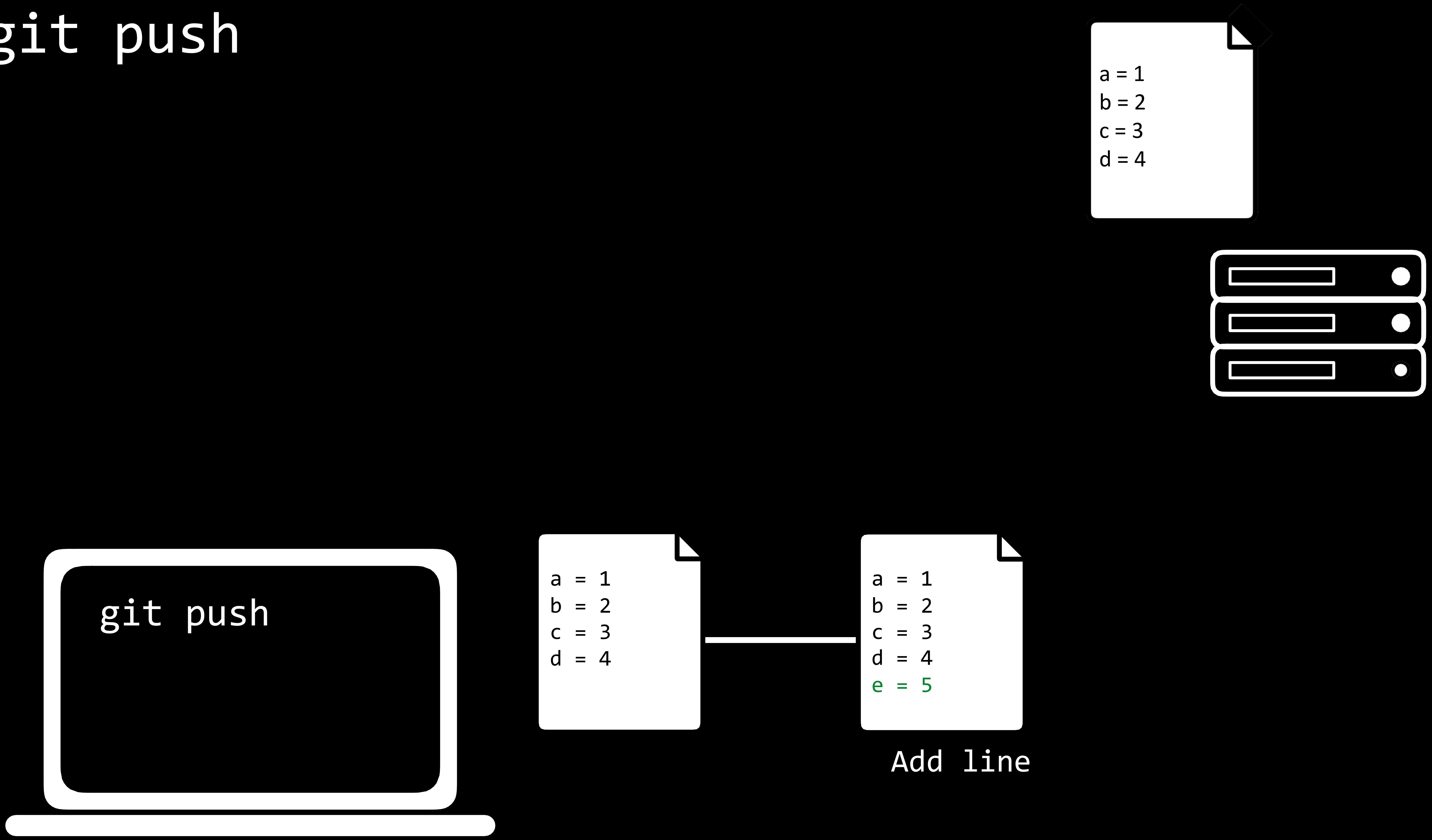
## git push

[1]

## git push

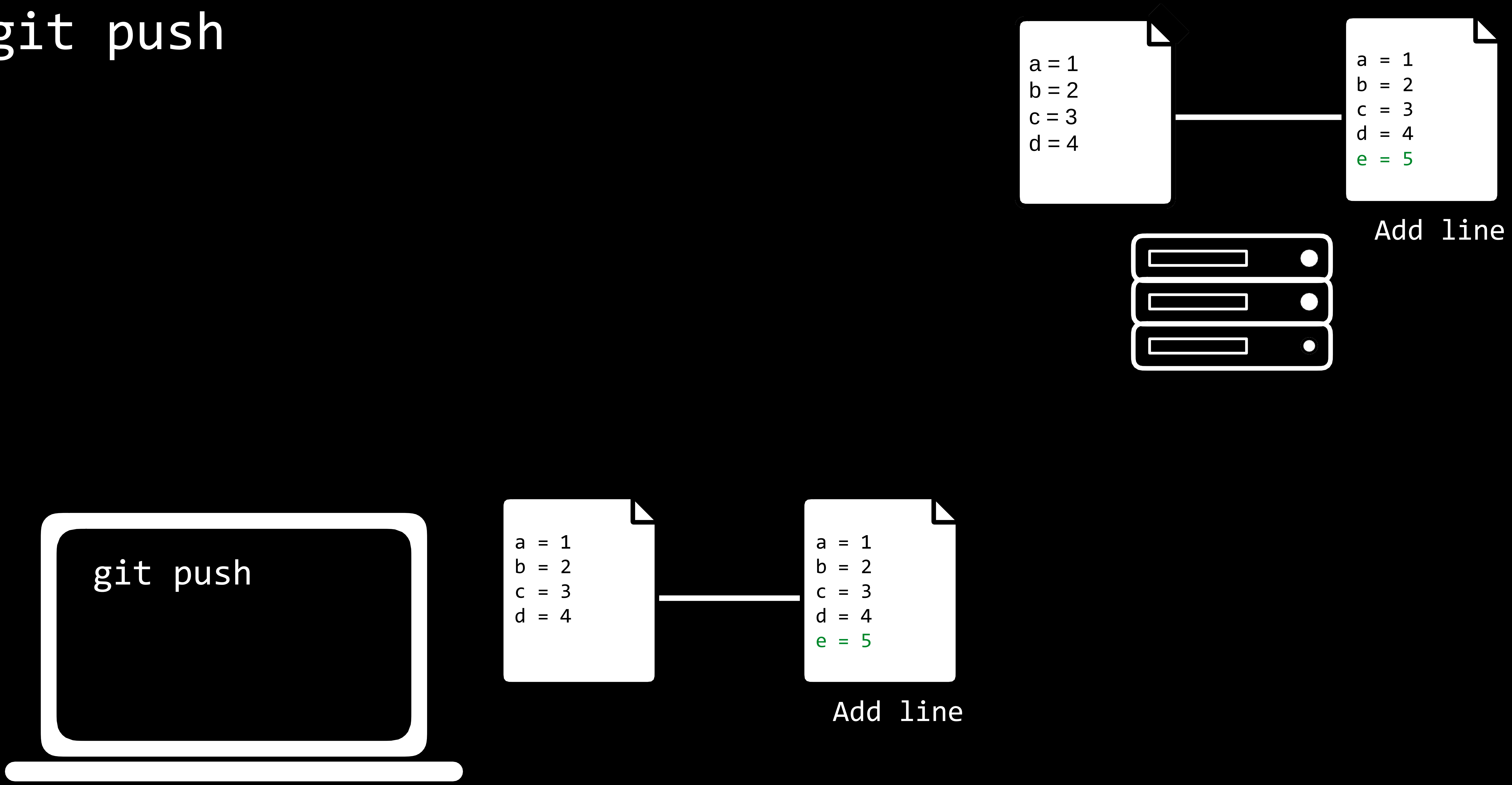


## git push



[1]

## git push

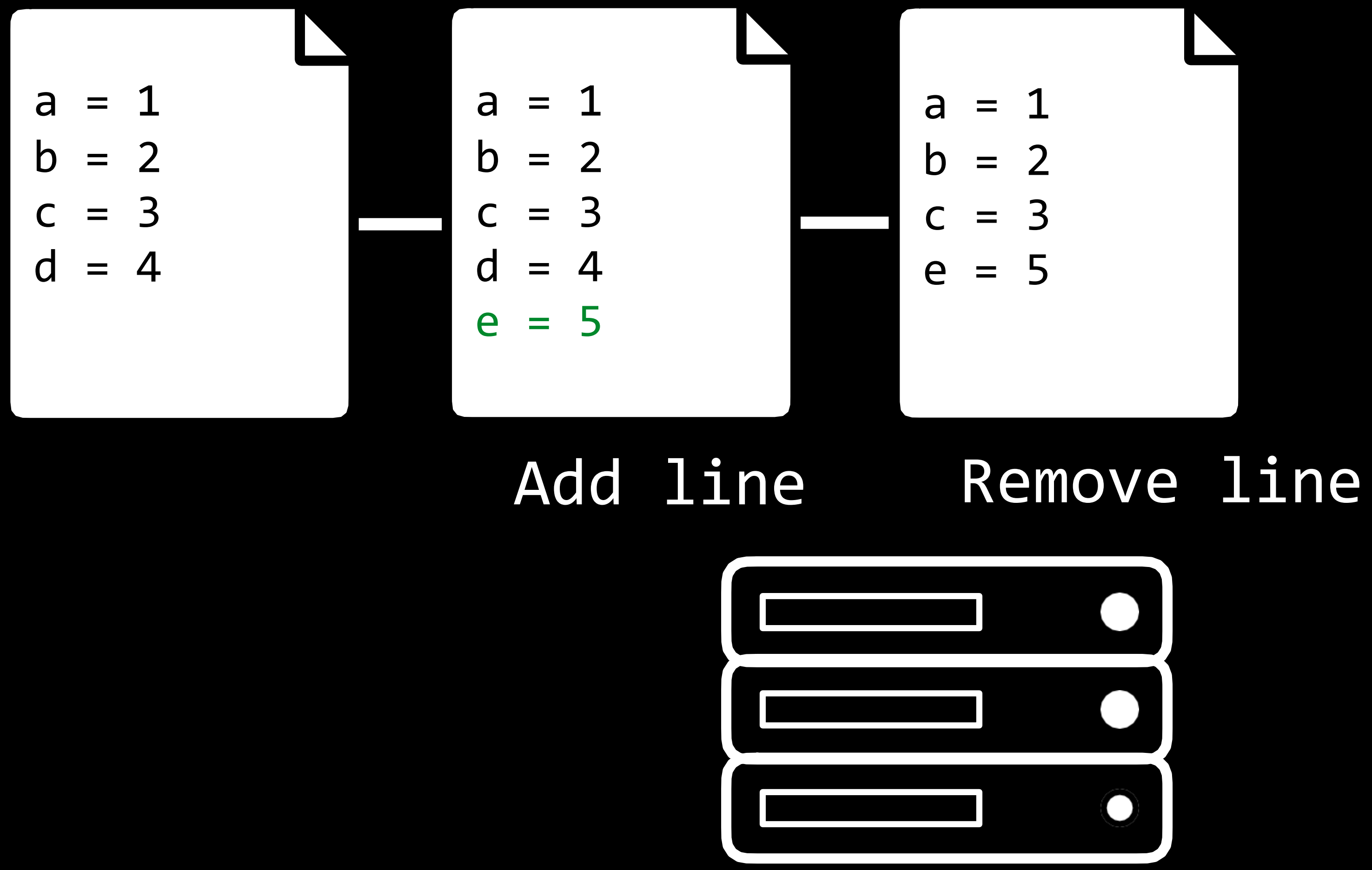

a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line
git push
a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line

[1]

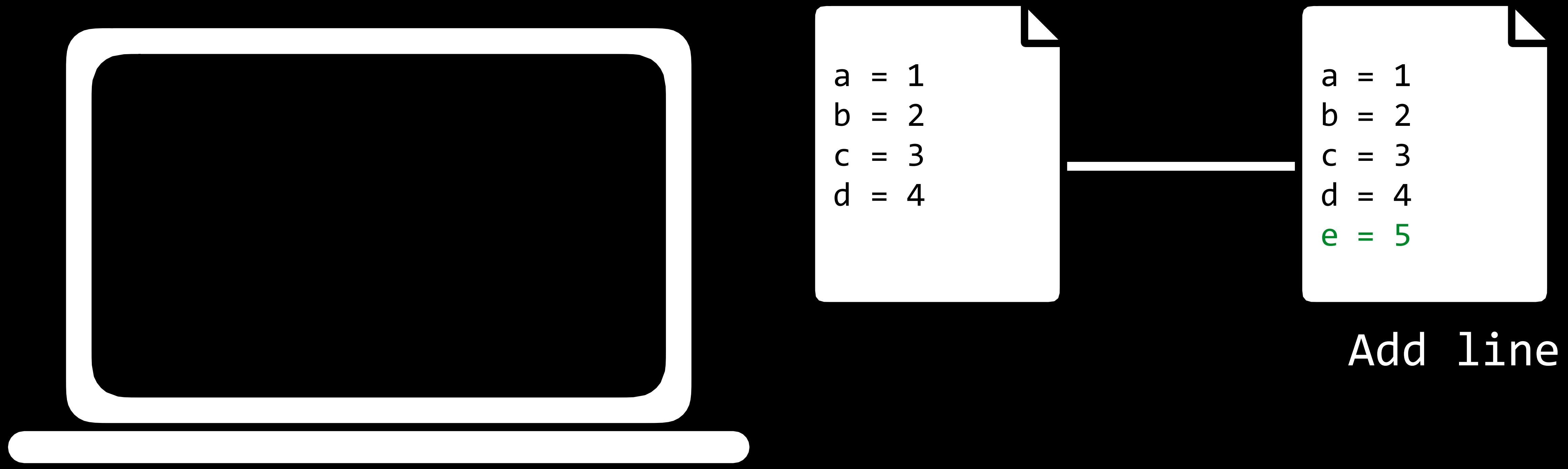
## `git pull`

[1]

# git pull



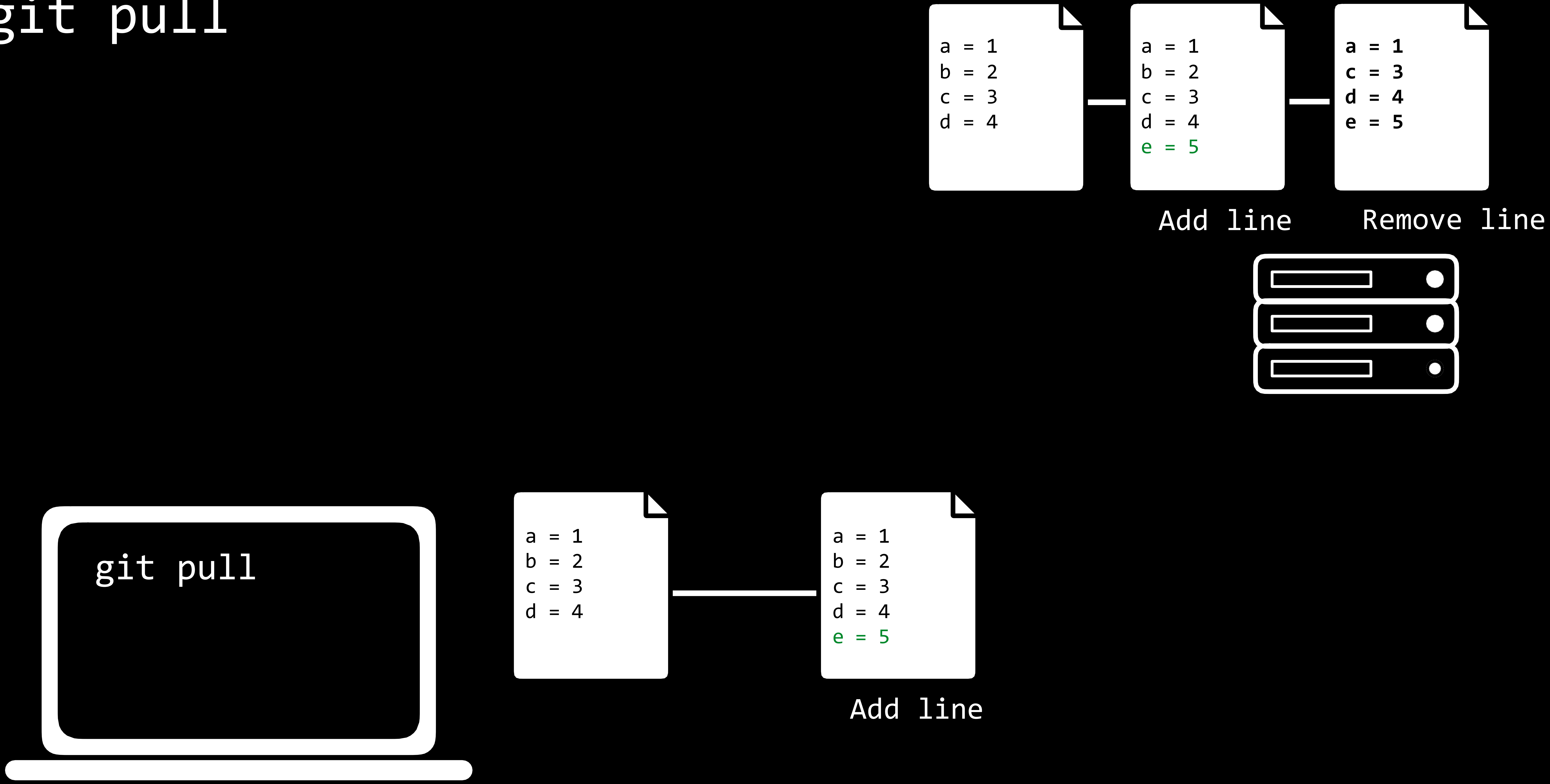
a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
a = 1
b = 2
c = 3
e = 5
Add line
Remove line



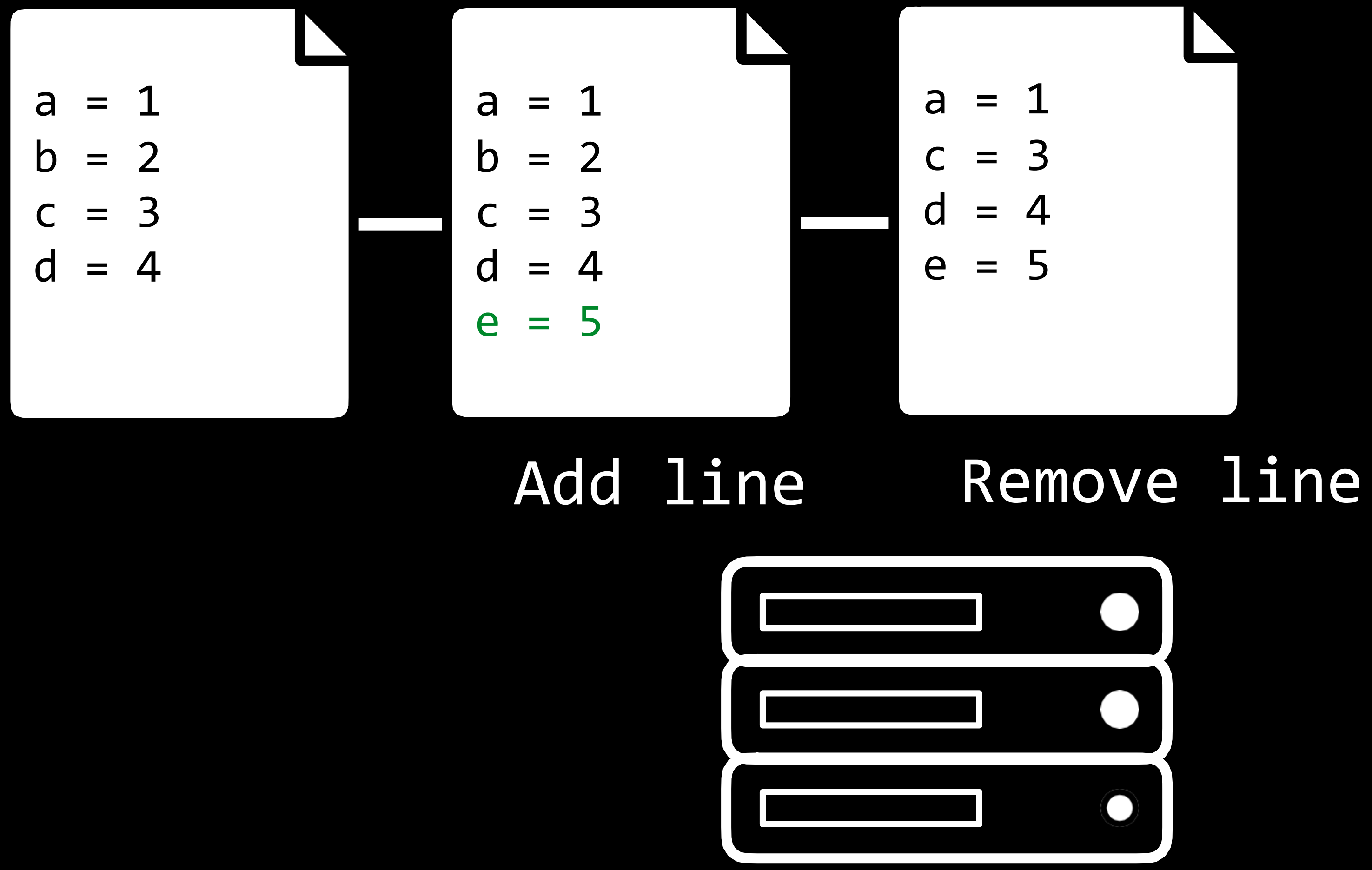
a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line

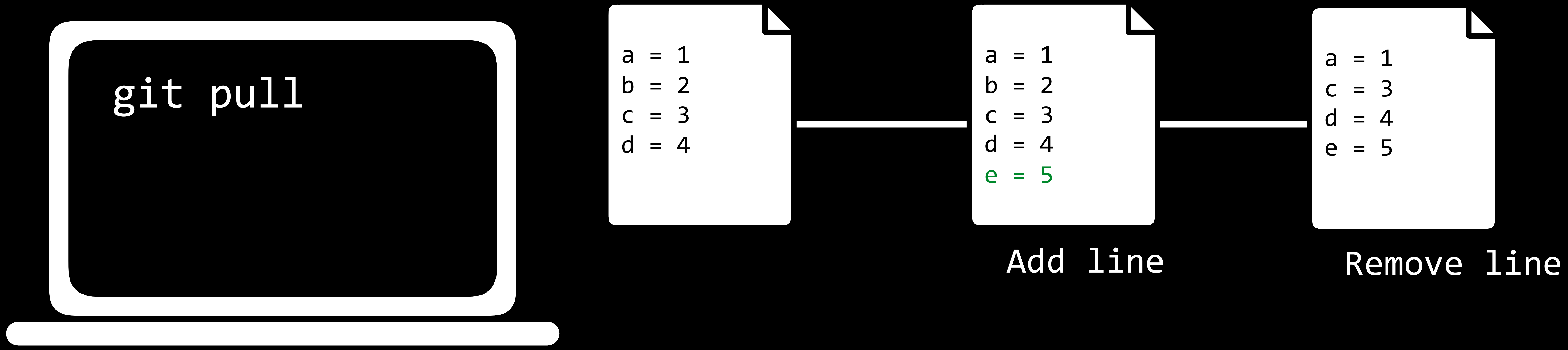
[1]

## git pull


a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line
a = 1
c = 3
d = 4
e = 5
Remove line
git pull
a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line

## git pull


a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line
a = 1
c = 3
d = 4
e = 5
Remove line


git pull
a = 1
b = 2
c = 3
d = 4
a = 1
b = 2
c = 3
d = 4
e = 5
Add line
a = 1
c = 3
d = 4
e = 5
Remove line

[1]

## Merge Conflicts

[1]

## Merge Conflicts

[1]

## Merge Conflicts

```
git pull
```

# Merge Conflicts

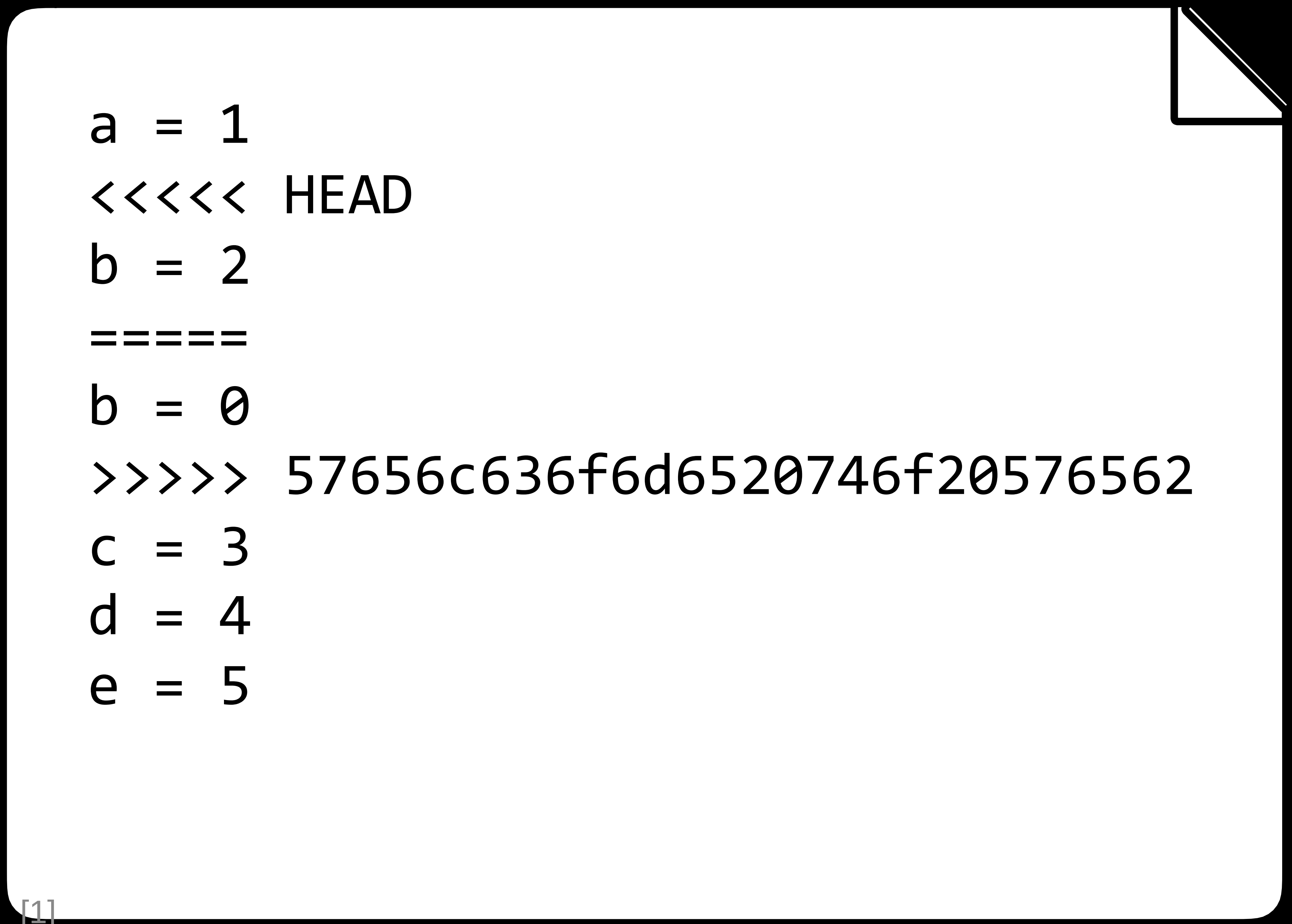
```
git pull
```

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

[1]

# Merge Conflicts

```
git pull
```

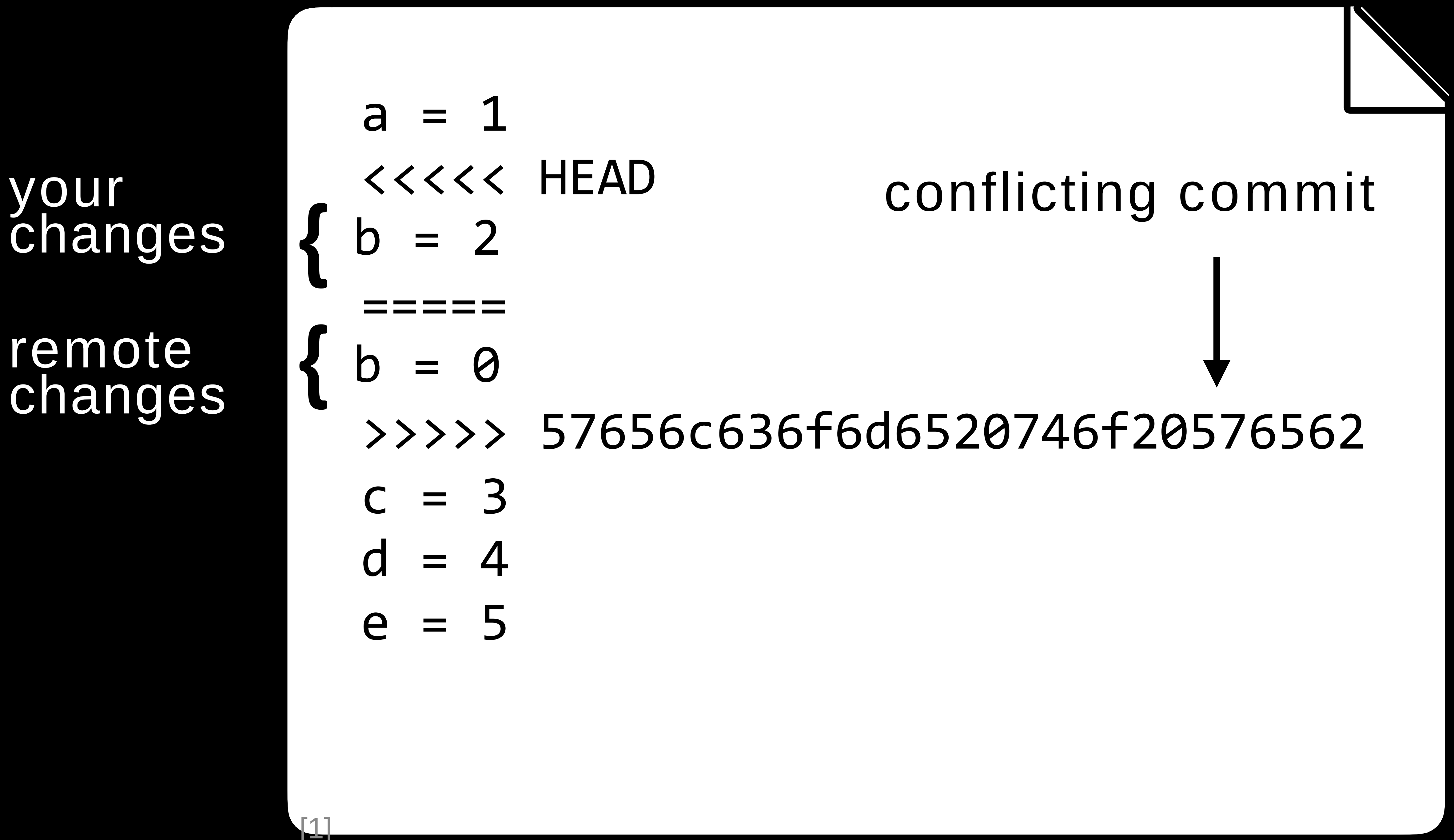


```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
c = 3
d = 4
e = 5
```

[1]

# Merge Conflicts

```
git pull
```



your changes

remote changes

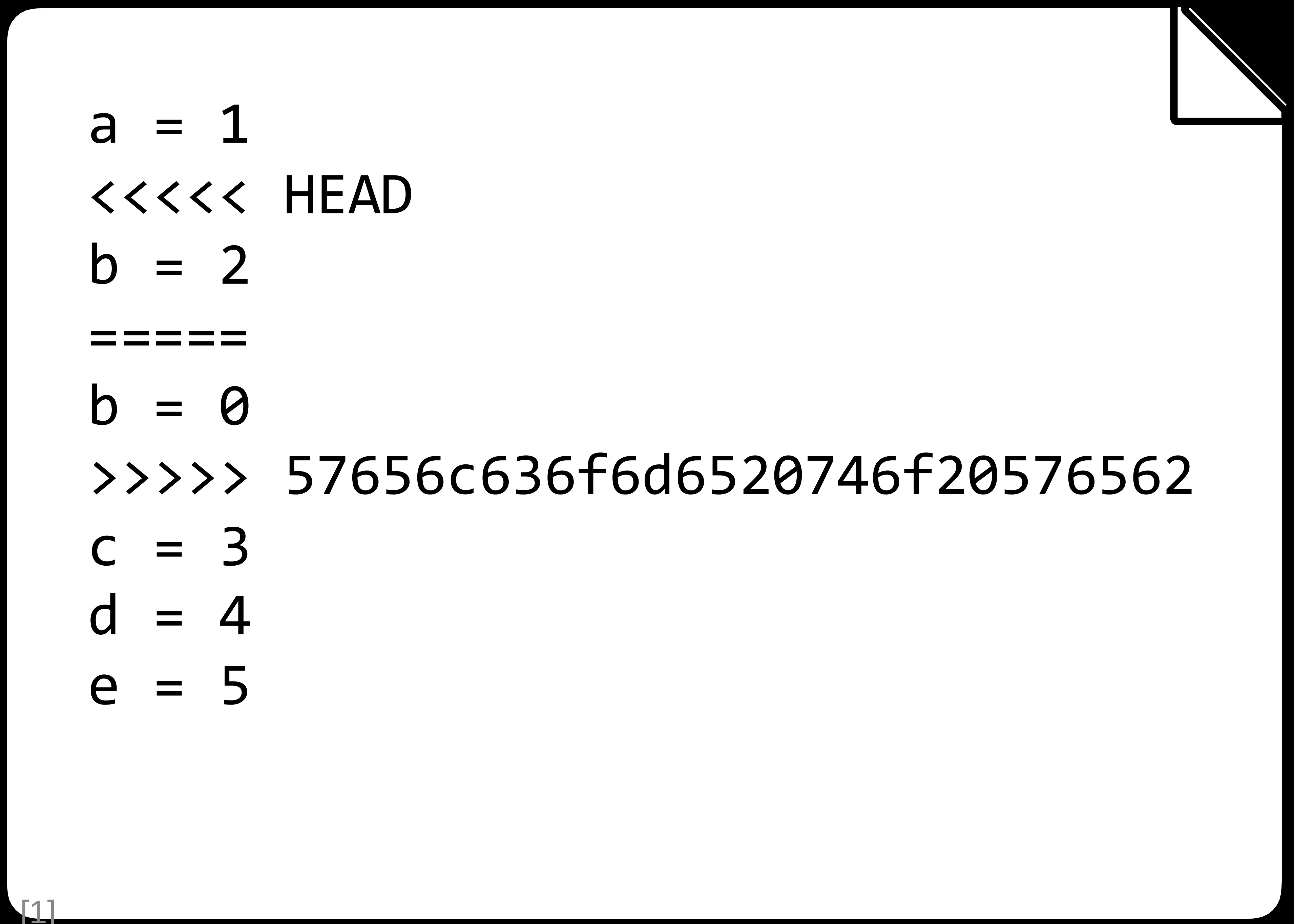
conflicting commit

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

[1]

# Merge Conflicts

```
git pull
```

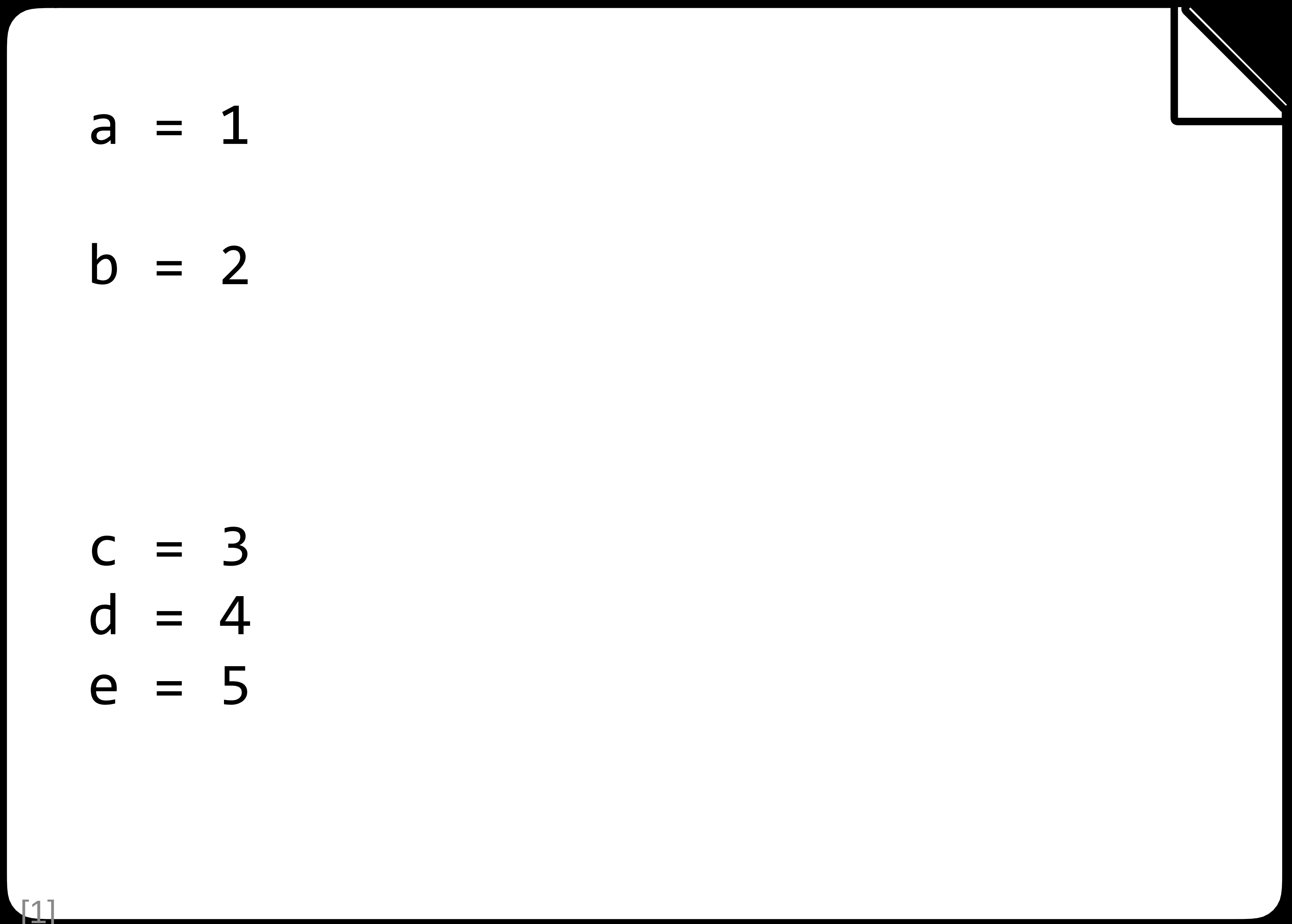


```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
c = 3
d = 4
e = 5
```

[1]

# Merge Conflicts

```
git pull
```



```
a = 1

b = 2




c = 3
d = 4
e = 5
```

[1]

## Merge Conflicts

```
git pull
```

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

[1]

# `git log`

[1]

## git log

[1]

## git log

```
git log
```

[1]

## 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
```

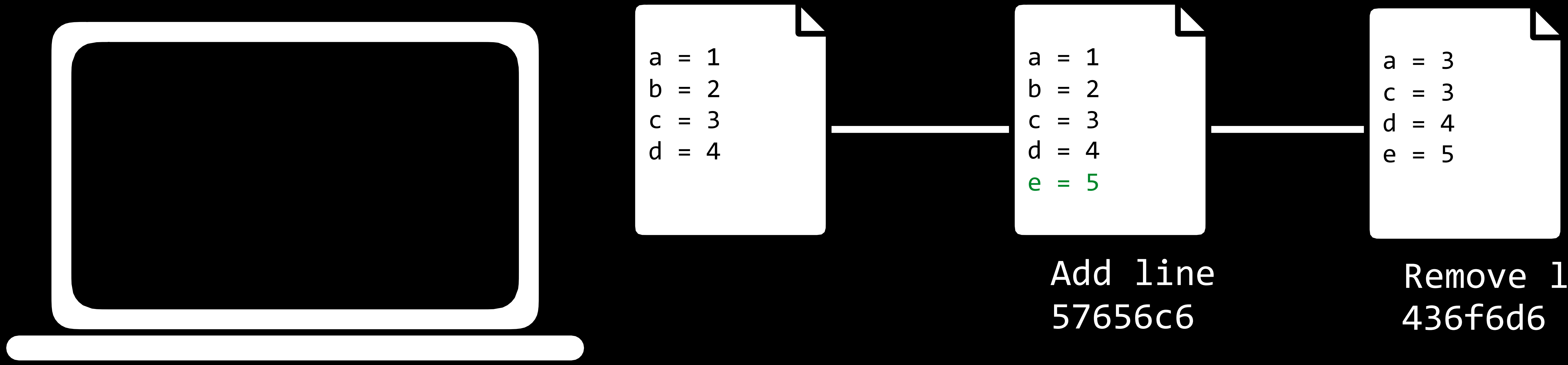
[1]

## git reset

[1]

## git reset

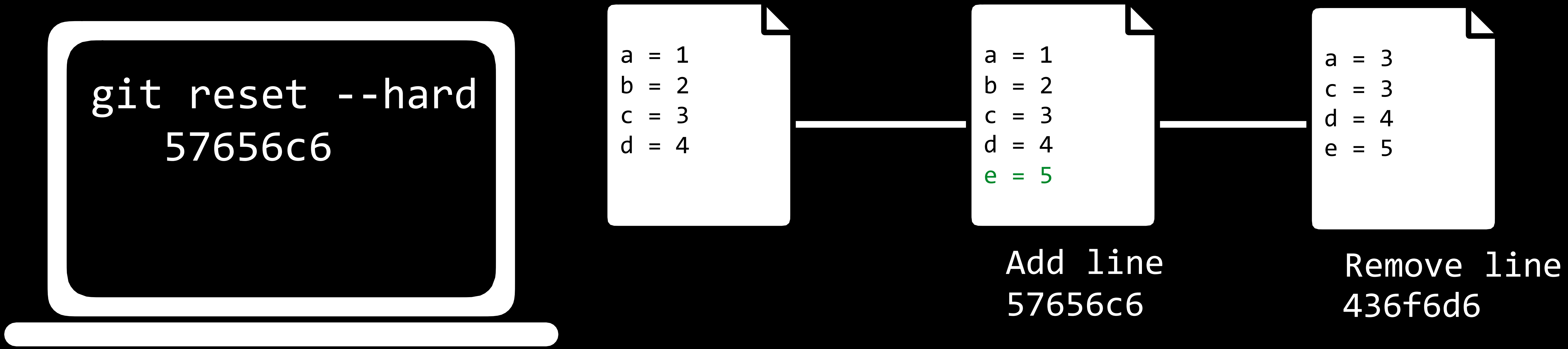
- `git reset --hard <commit>`
- `git reset --hard origin/master`



[1]

## git reset

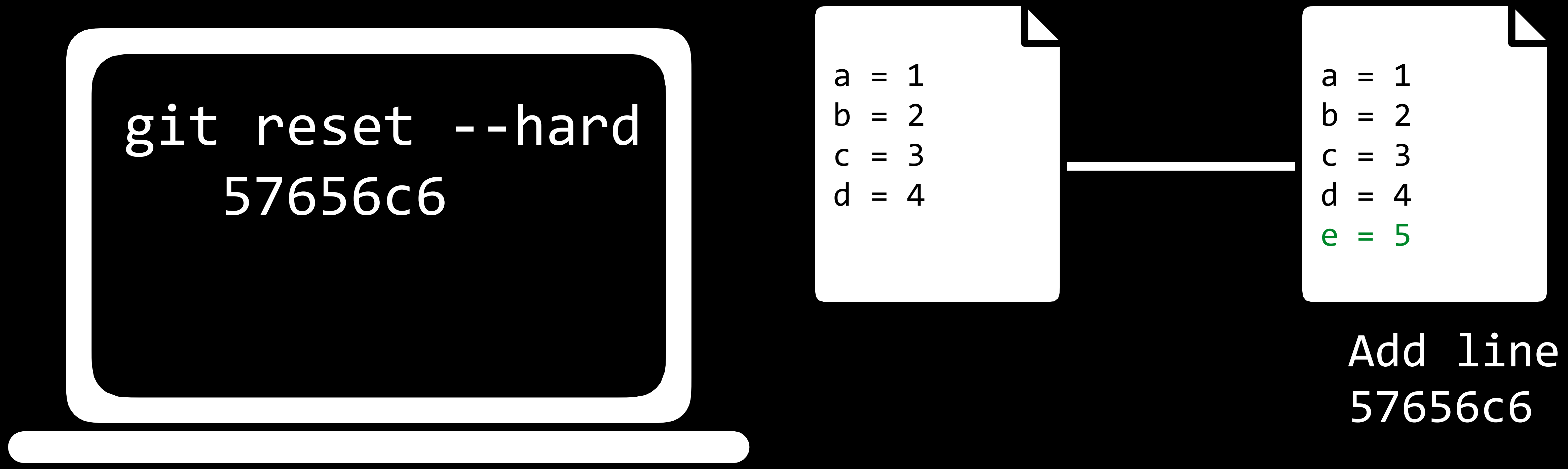
- `git reset --hard <commit>`
- `git reset --hard origin/master`



[1]

## git reset

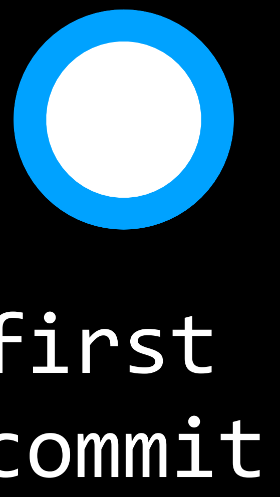
- git reset --hard <commit>
- git reset --hard origin/master

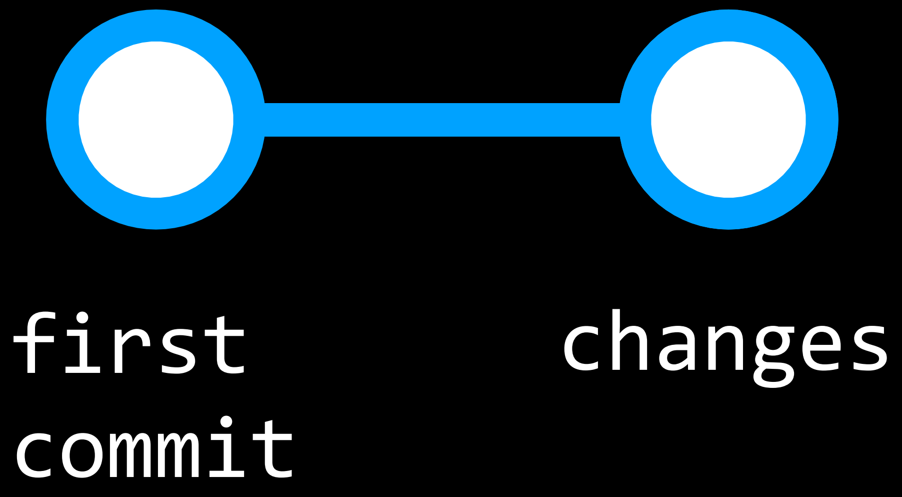


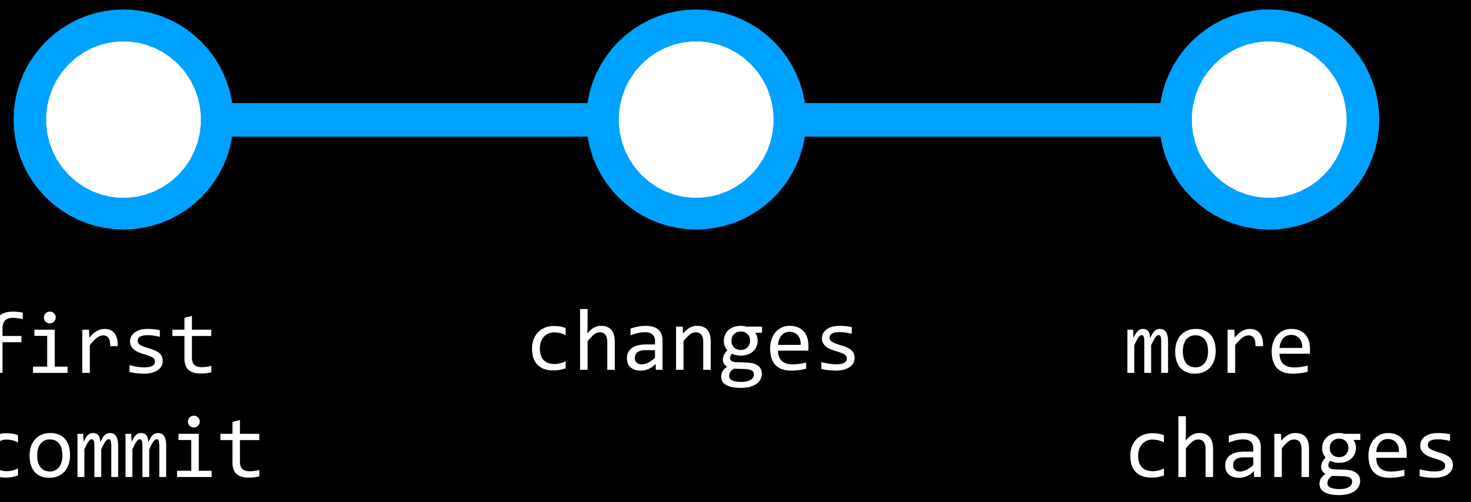
[1]

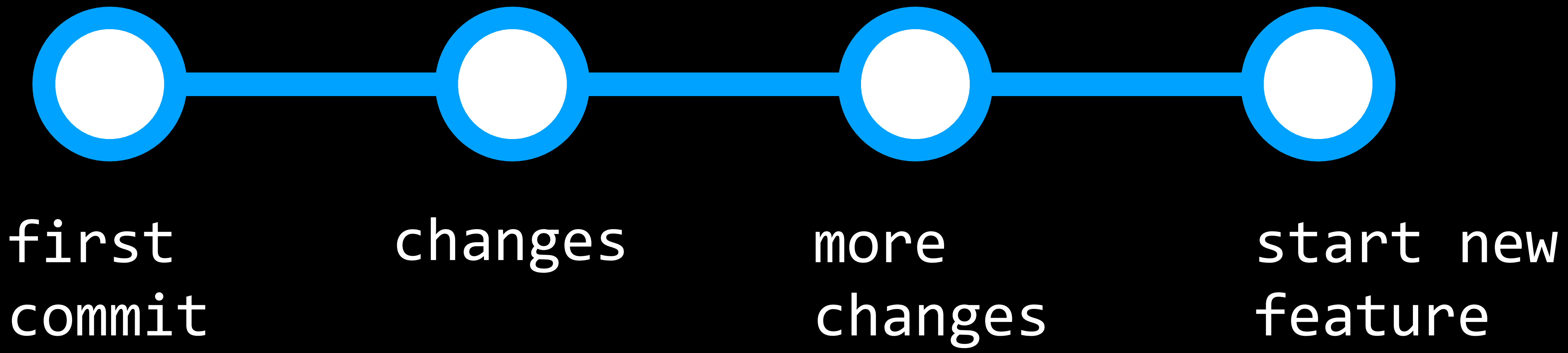
## Making Changes

[1]

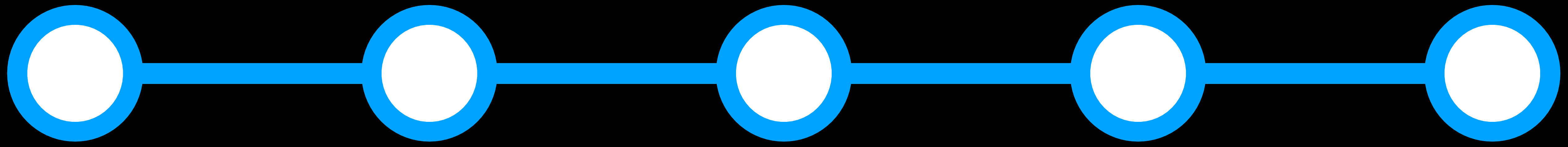

first
commit


first commit
changes


first commit
changes
more changes

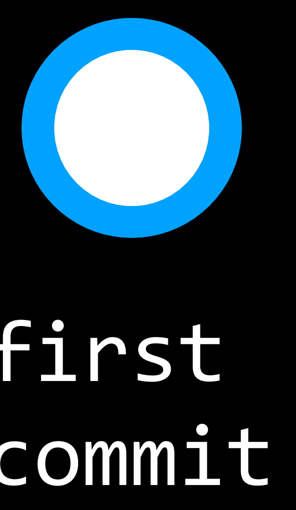


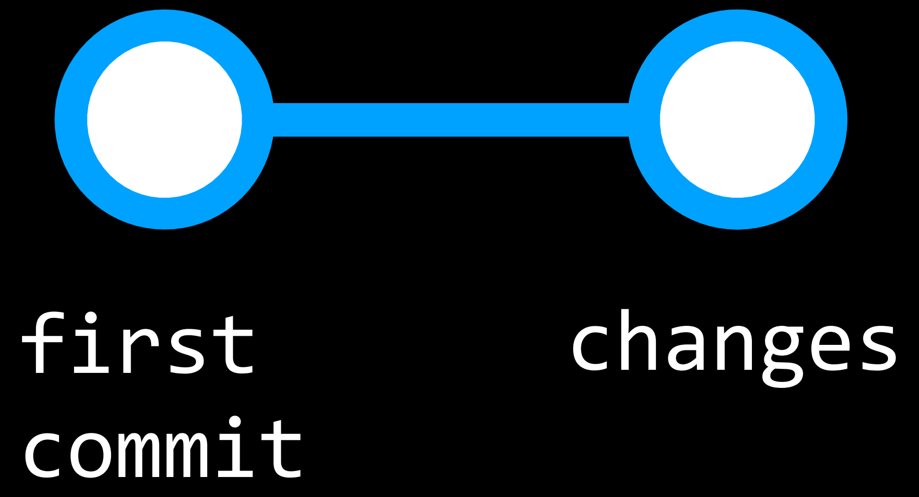
[1]

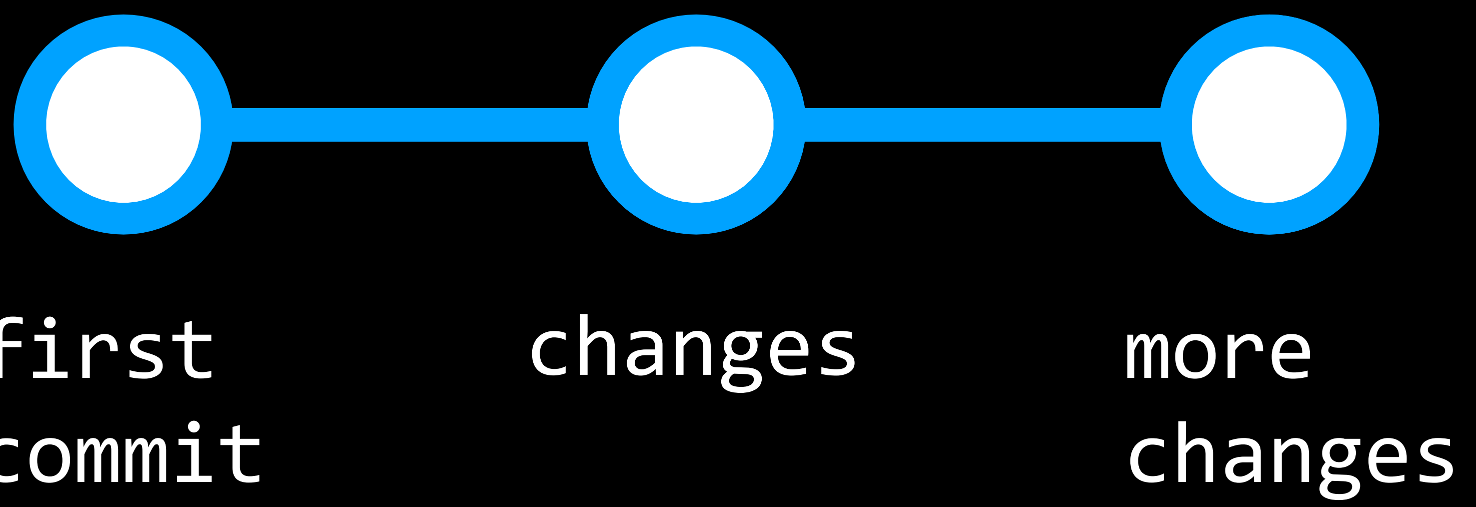

first commit
changes
more changes
start new feature
keep working on new feature

## Branching

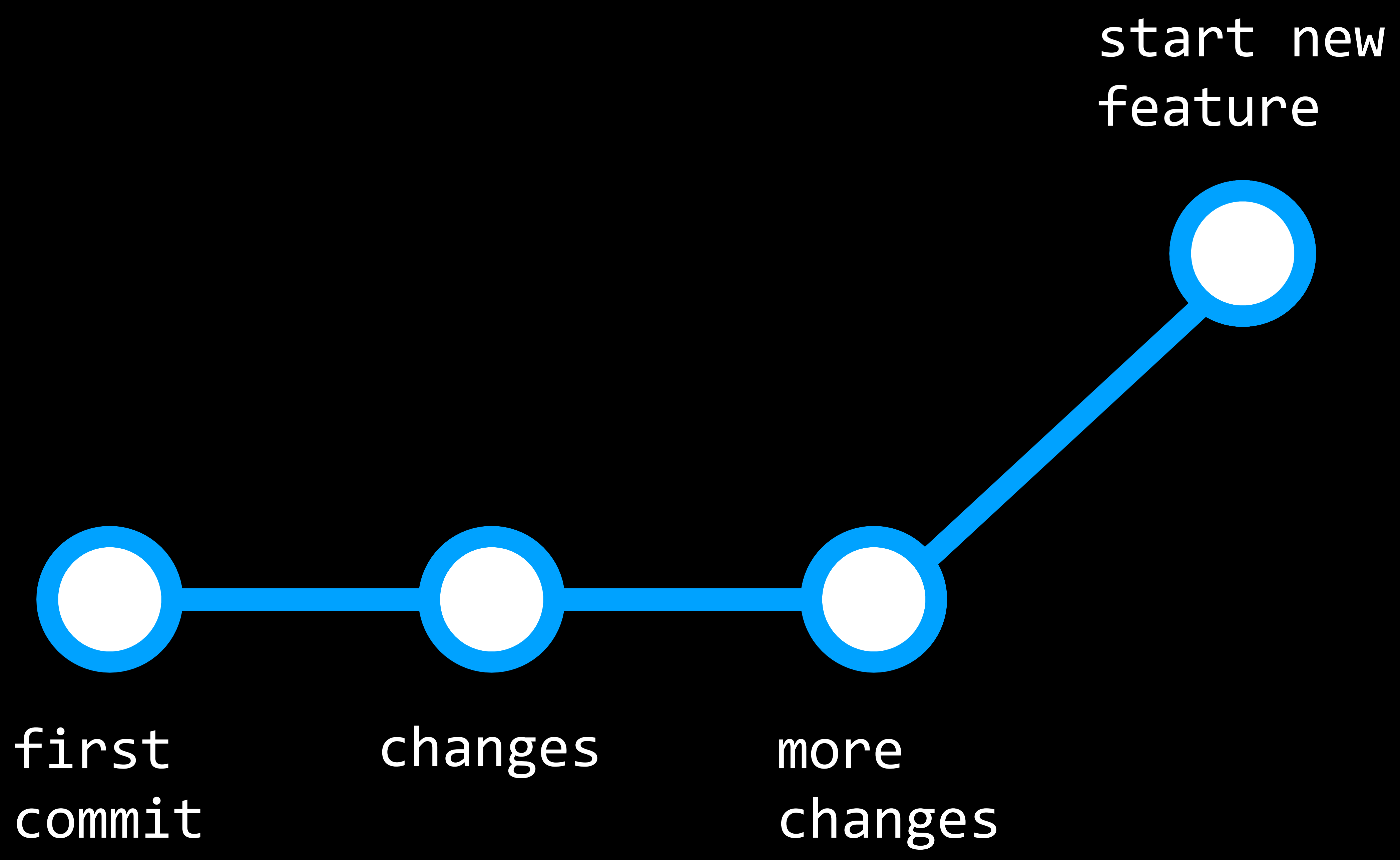
[1]

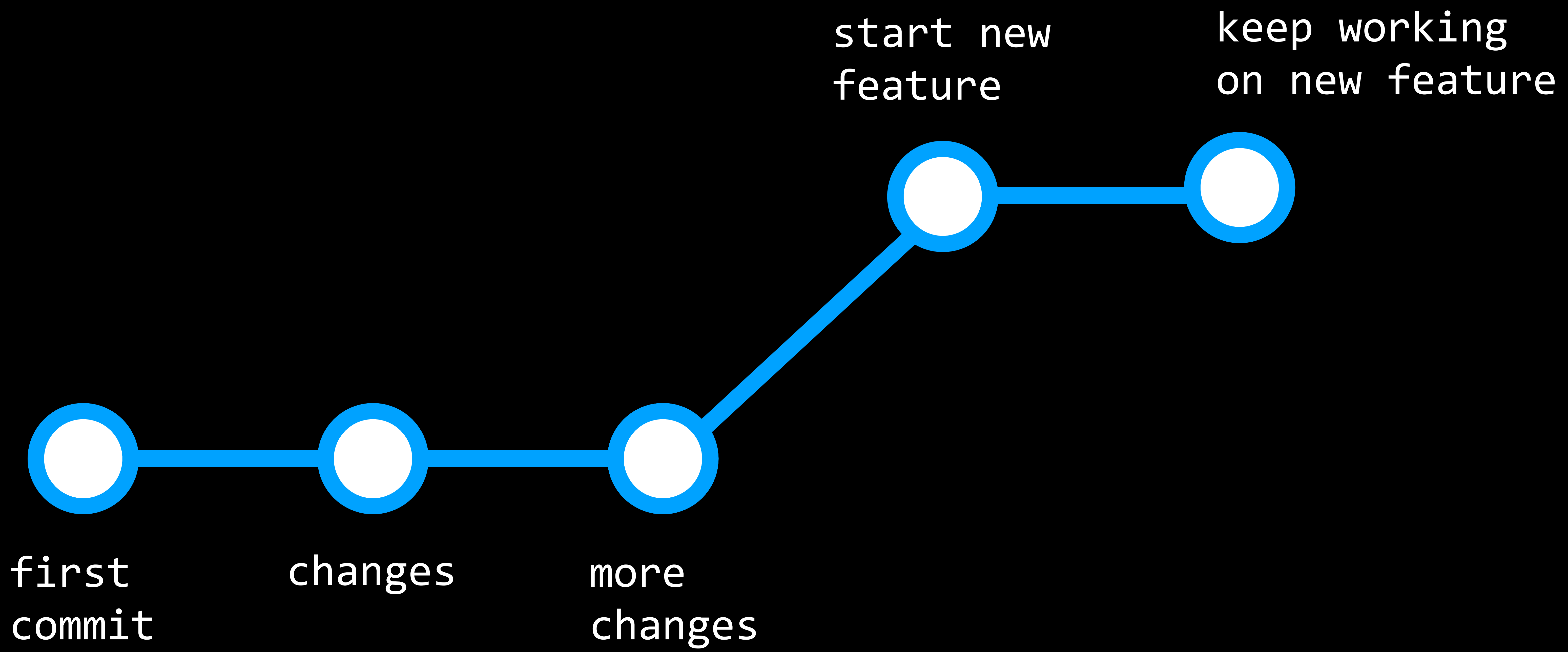

first
commit

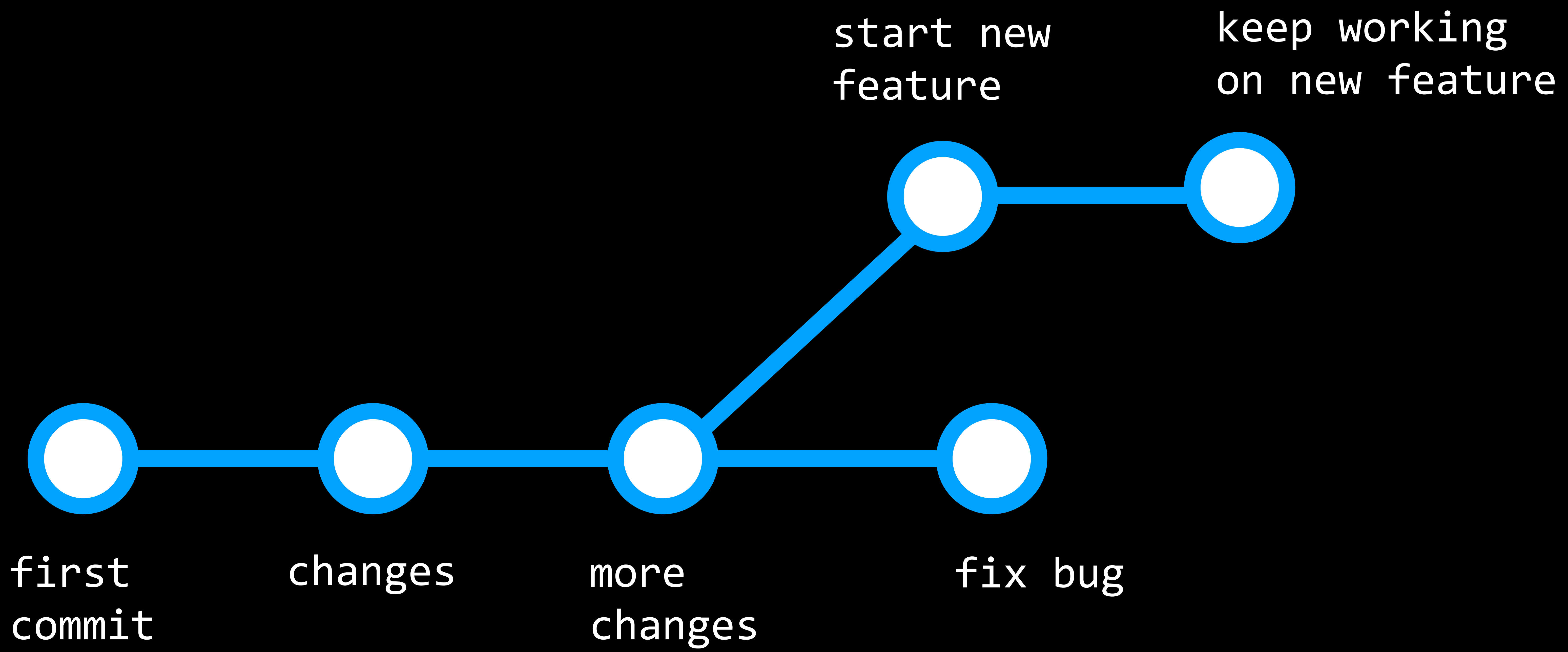

first commit
changes



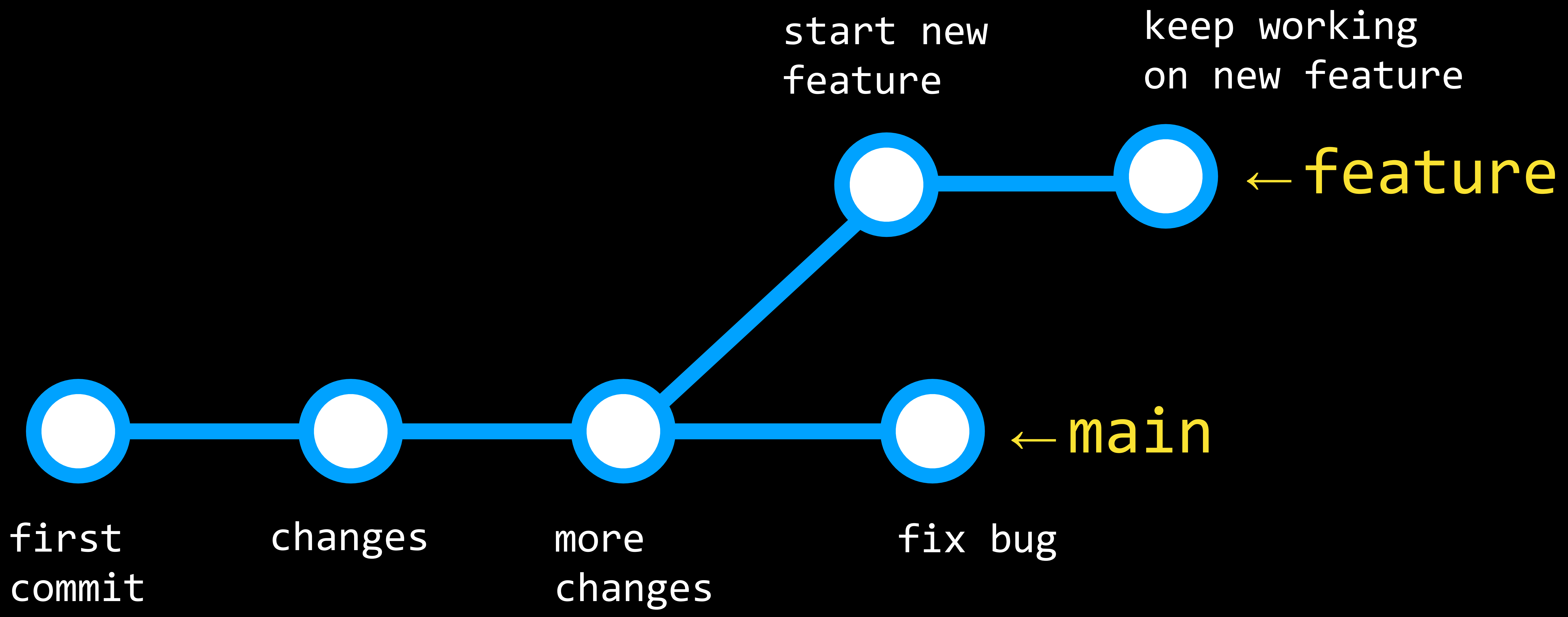
[1]


start new feature
first commit
changes
more changes

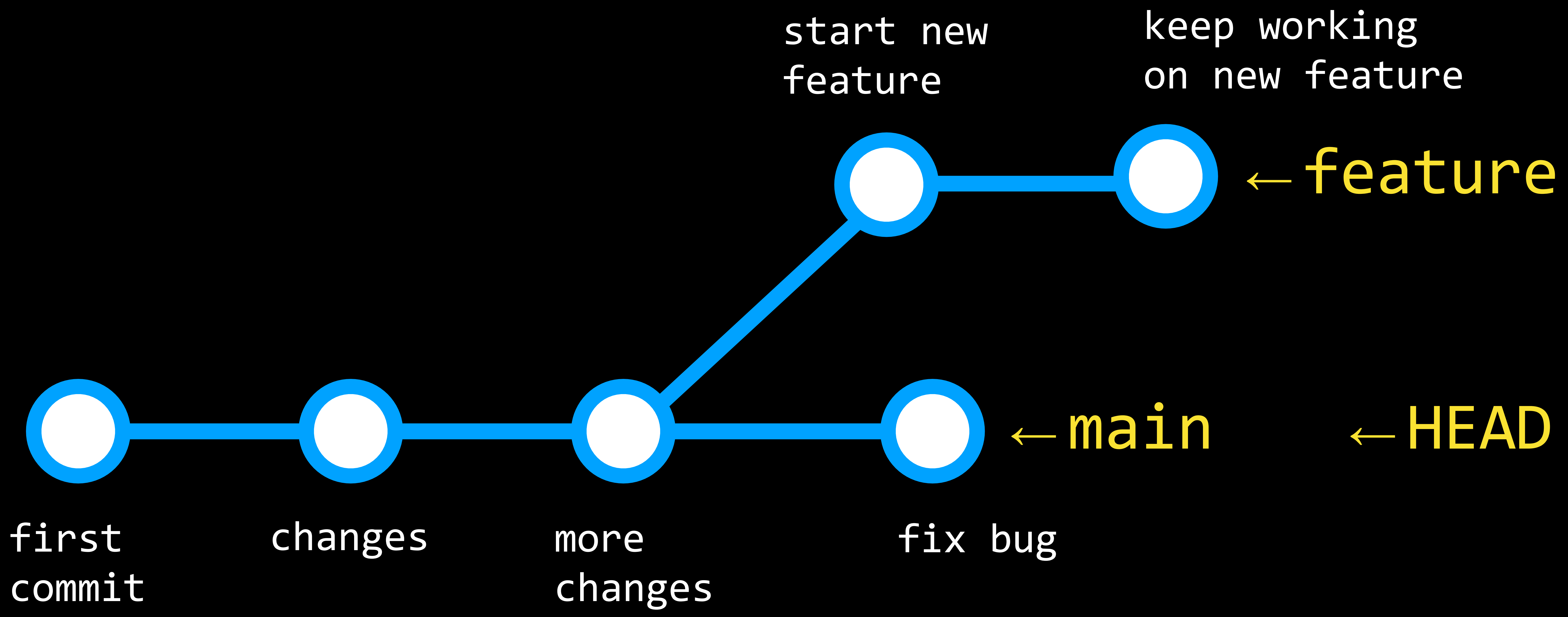

start new feature
keep working on new feature
first commit
changes
more changes


start new feature
keep working on new feature
first commit
changes
more changes
fix bug

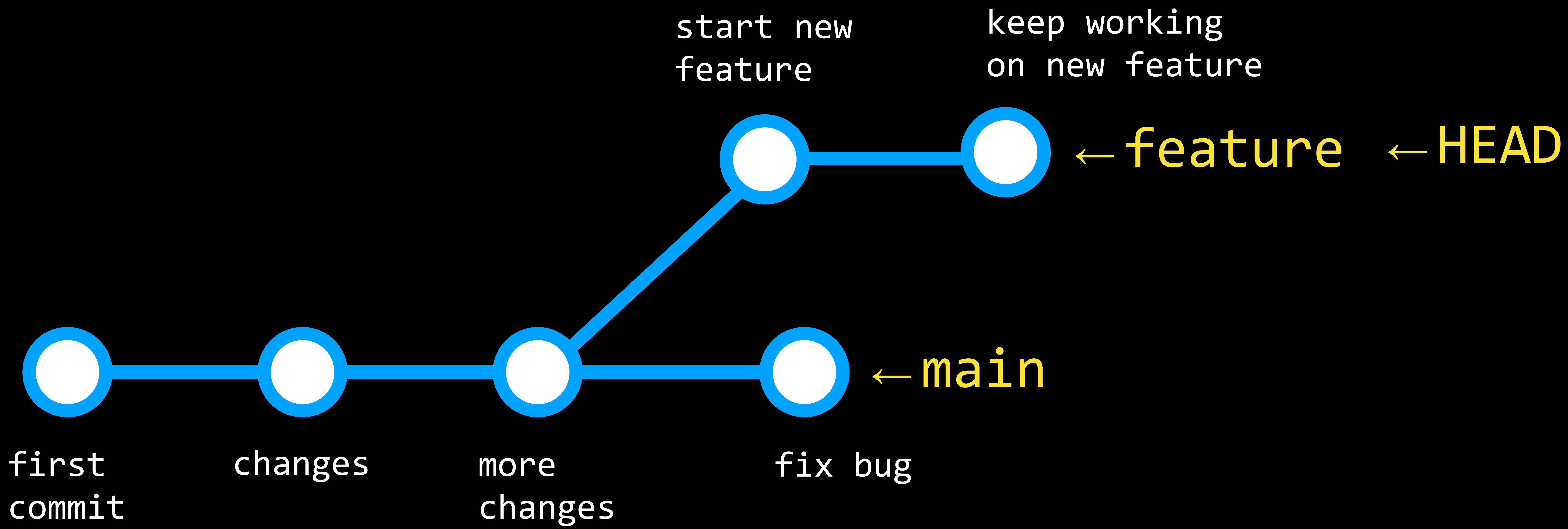
[1]


start new feature
keep working on new feature
← feature
← main
first commit
changes
more changes
fix bug

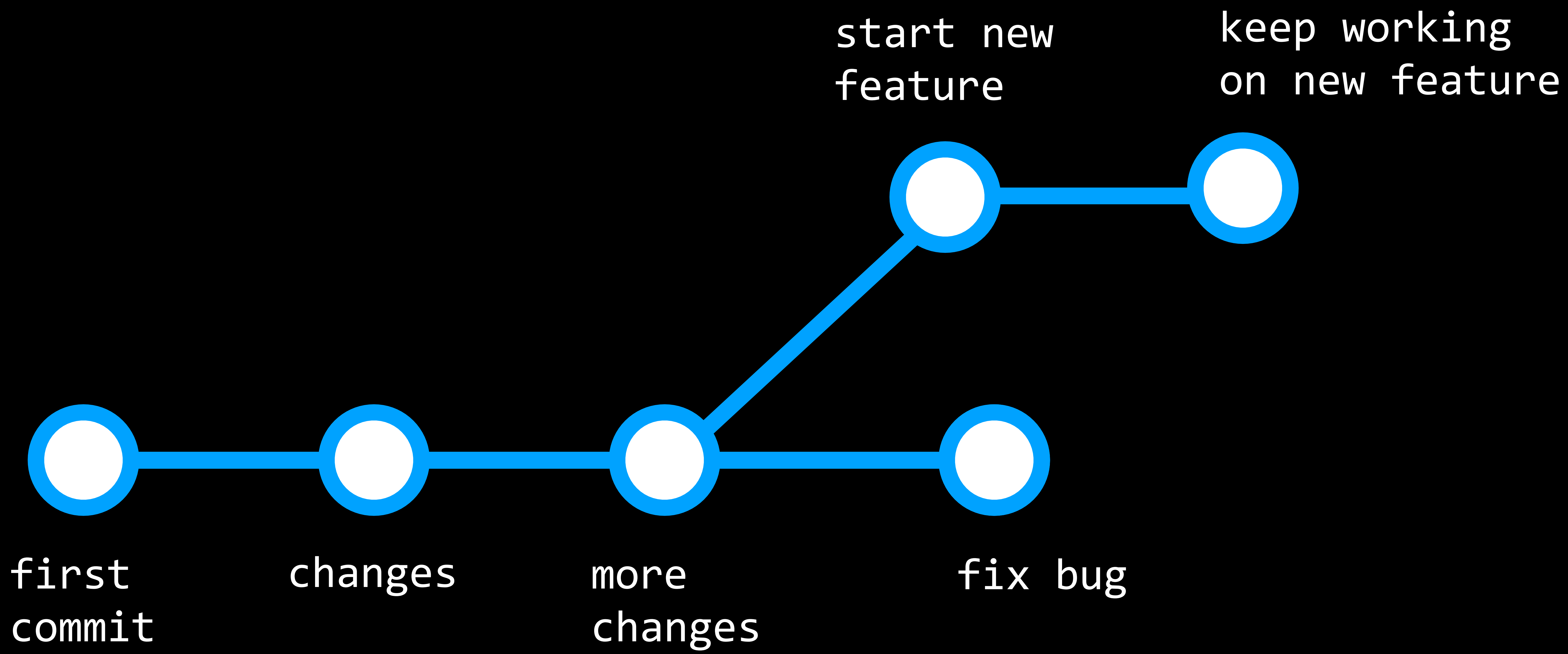
[1]


start new feature
keep working on new feature
← feature
← main
← HEAD
first commit
changes
more changes
fix bug

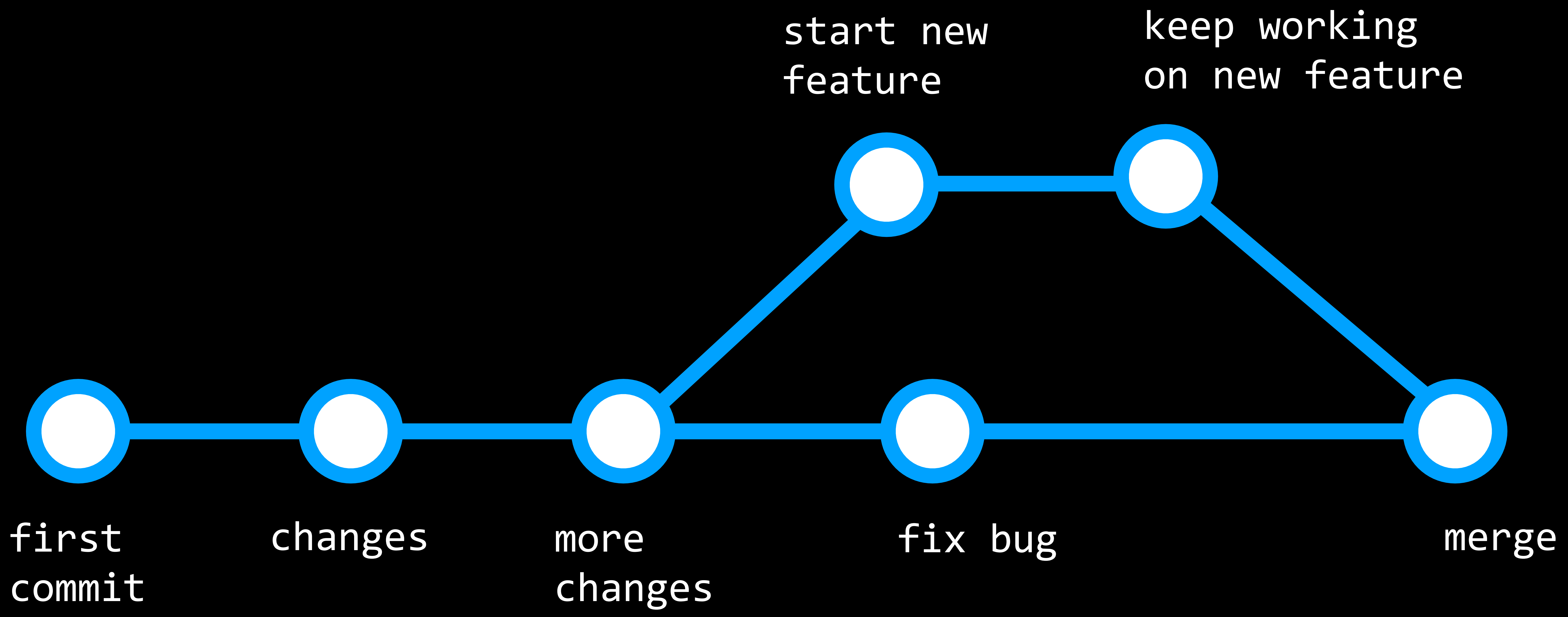
[1]


start new feature
keep working on new feature
← feature
← HEAD
← main
first commit
changes
more changes
fix bug

[1]


start new feature
keep working on new feature
first commit
changes
more changes
fix bug

[1]


start new feature
keep working on new feature
first commit
changes
more changes
fix bug
merge

[1]

## Branching

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

# Beispiel Git-Repository

https://bit.ly/3tMom87

# Weitere Informationen

https://cs50.harvard.edu/web/2020/weeks/1/

[1]

## Lizenz und Quellen

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

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

[1]