Version Control with Git

Md Tariqul Islam
RA 228009

Key Elements: Computational Reproducibility

- Data
- Code
- Documentation
- Workflow
- Distribution

Key Elements: Computational Reproducibility

- Data
- Code :
  - Track record all updates done in codes
  - Give access to code with adequate documentation
    - Version control
      - git
    - Documentation
      - readme
- Documentation
- Workflow
- Distribution

Version Control

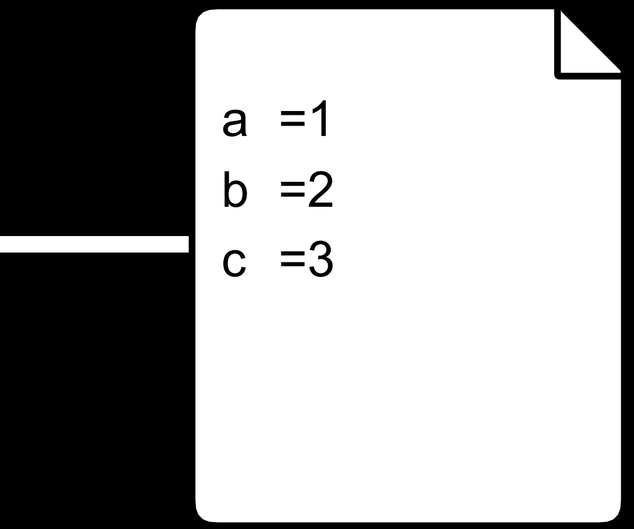
“… is the management of changes to documents, programs, and other information stored as computer files.”

- Wikipedia

Git

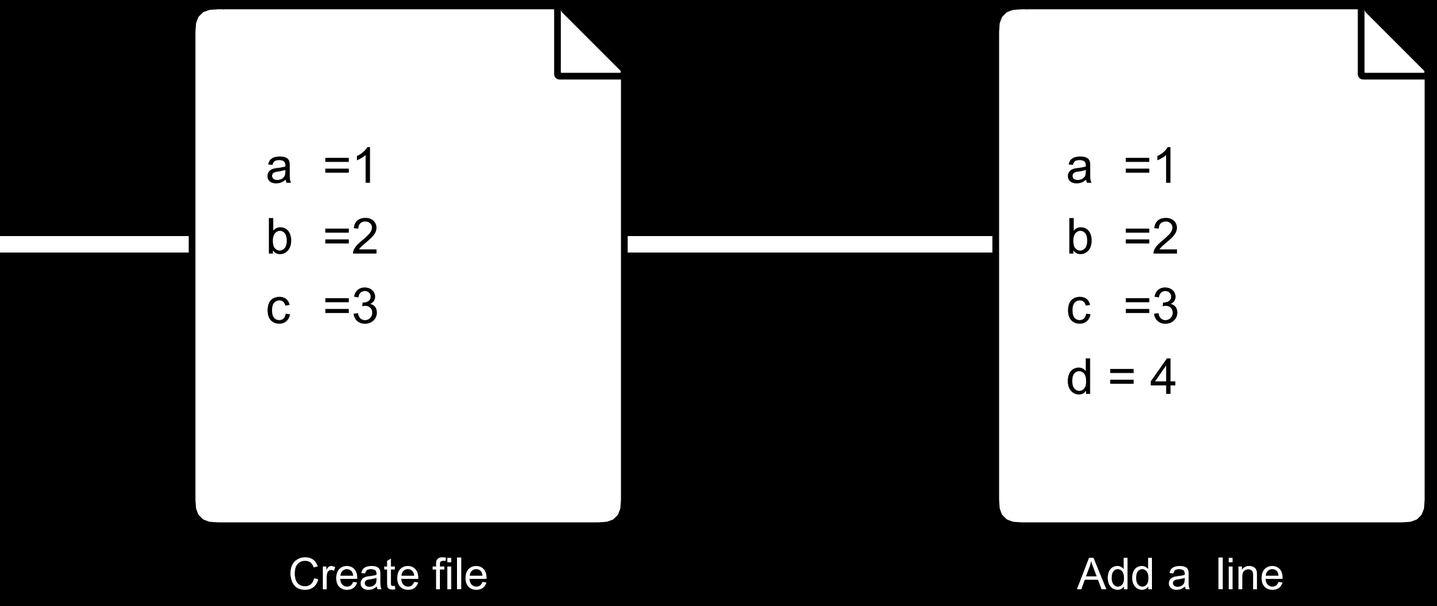
Most popular a piece of software work as version control tool

Keep track of changes to code.

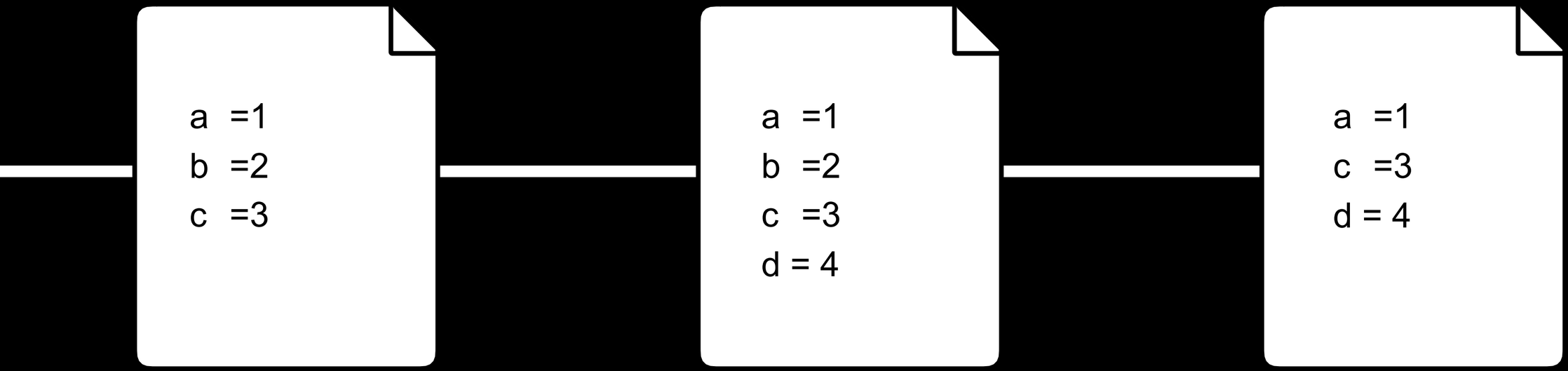


Create file

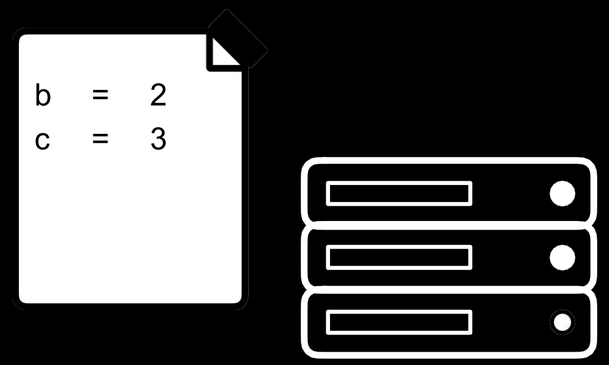
Keep track of changes to code.

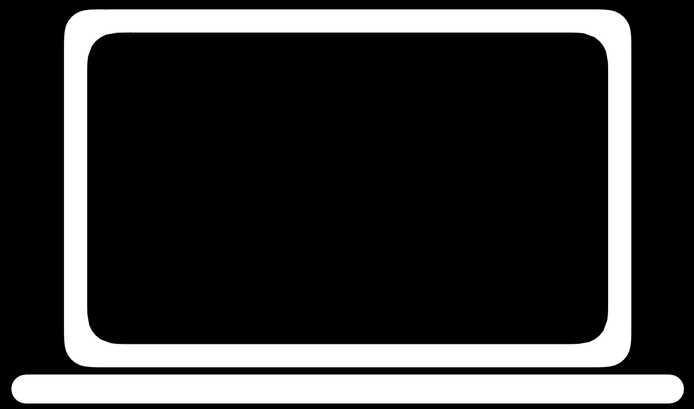


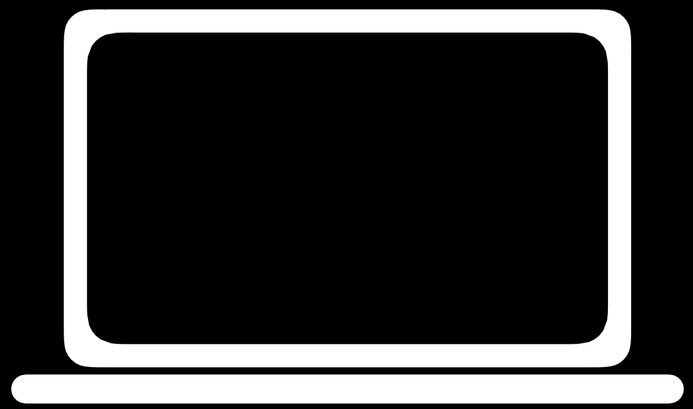
Keep track of changes to code.



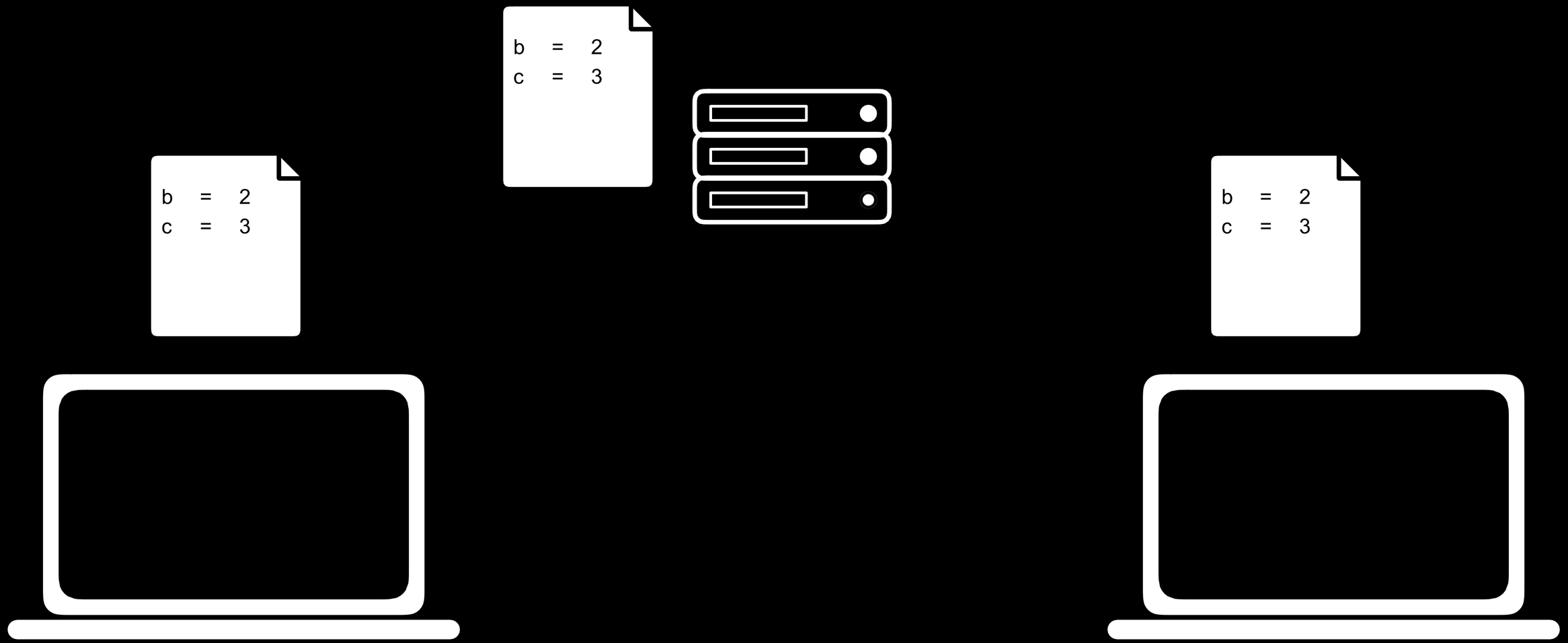
Synchronizes code between different people.



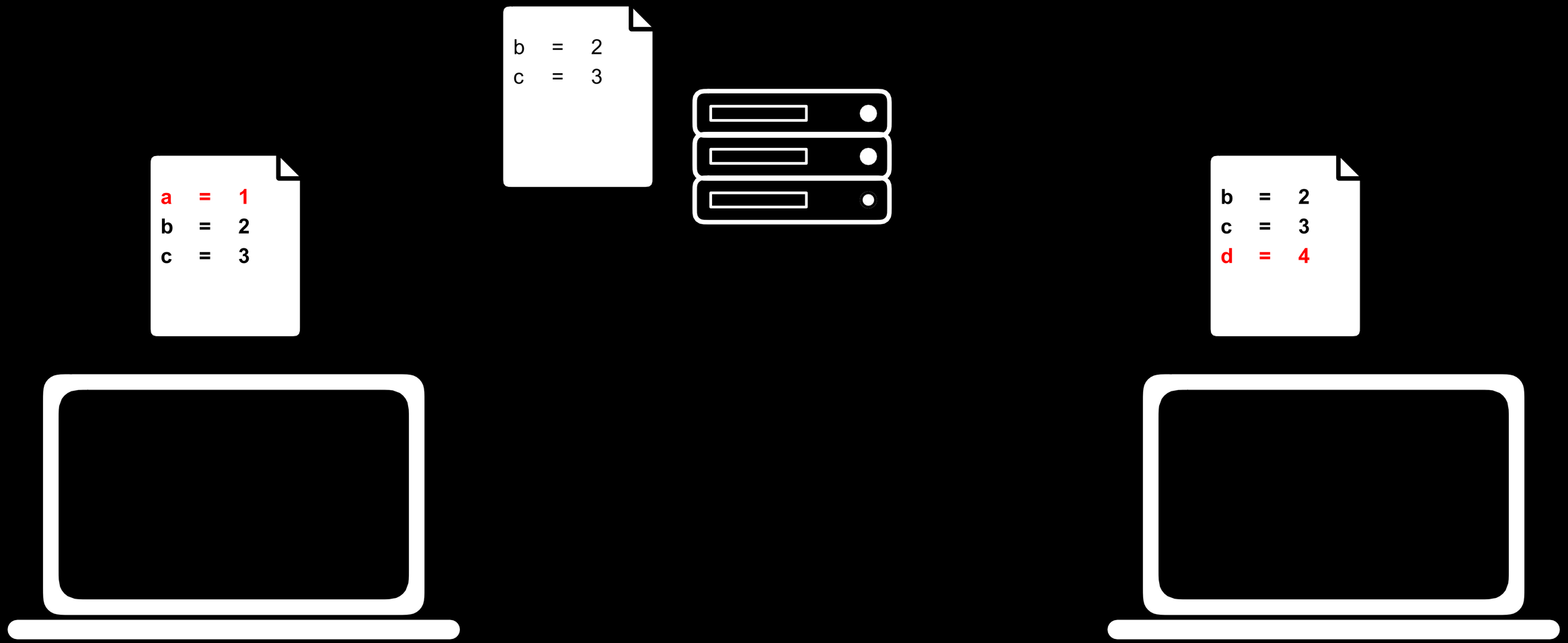




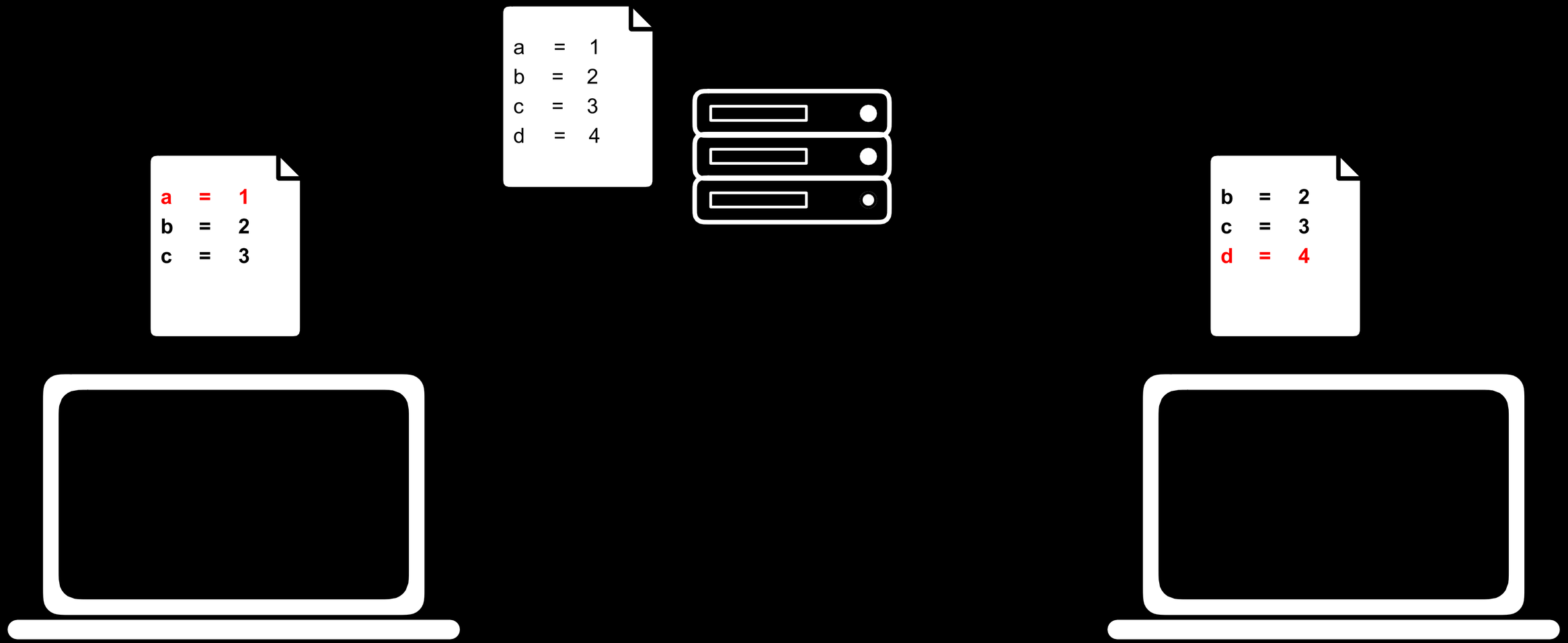
Synchronizes code between different people.



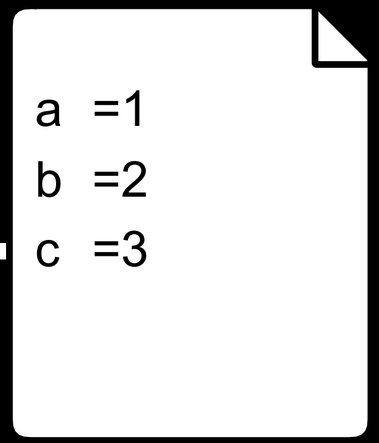
Synchronizes code between different people.



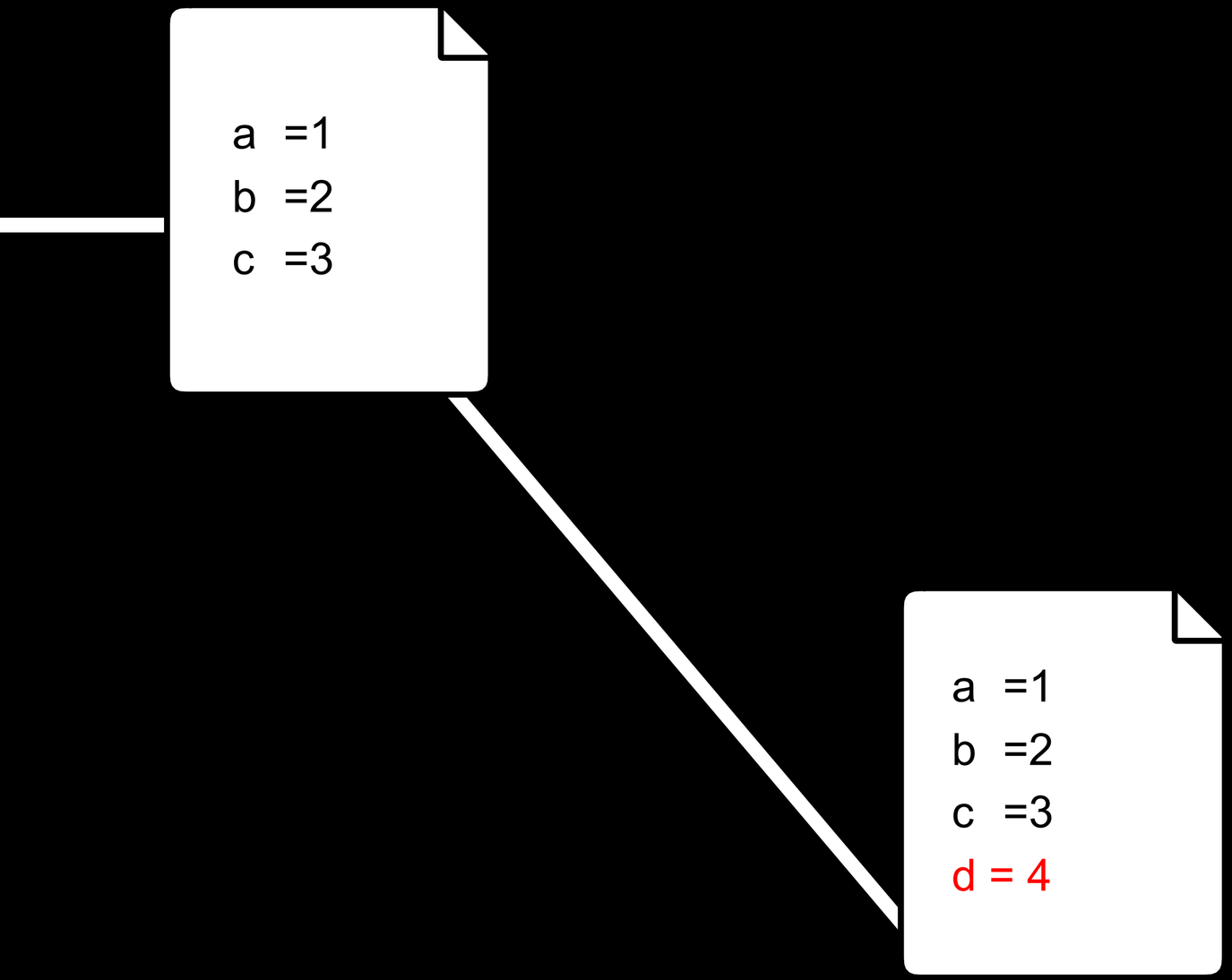
Synchronizes code between different people.



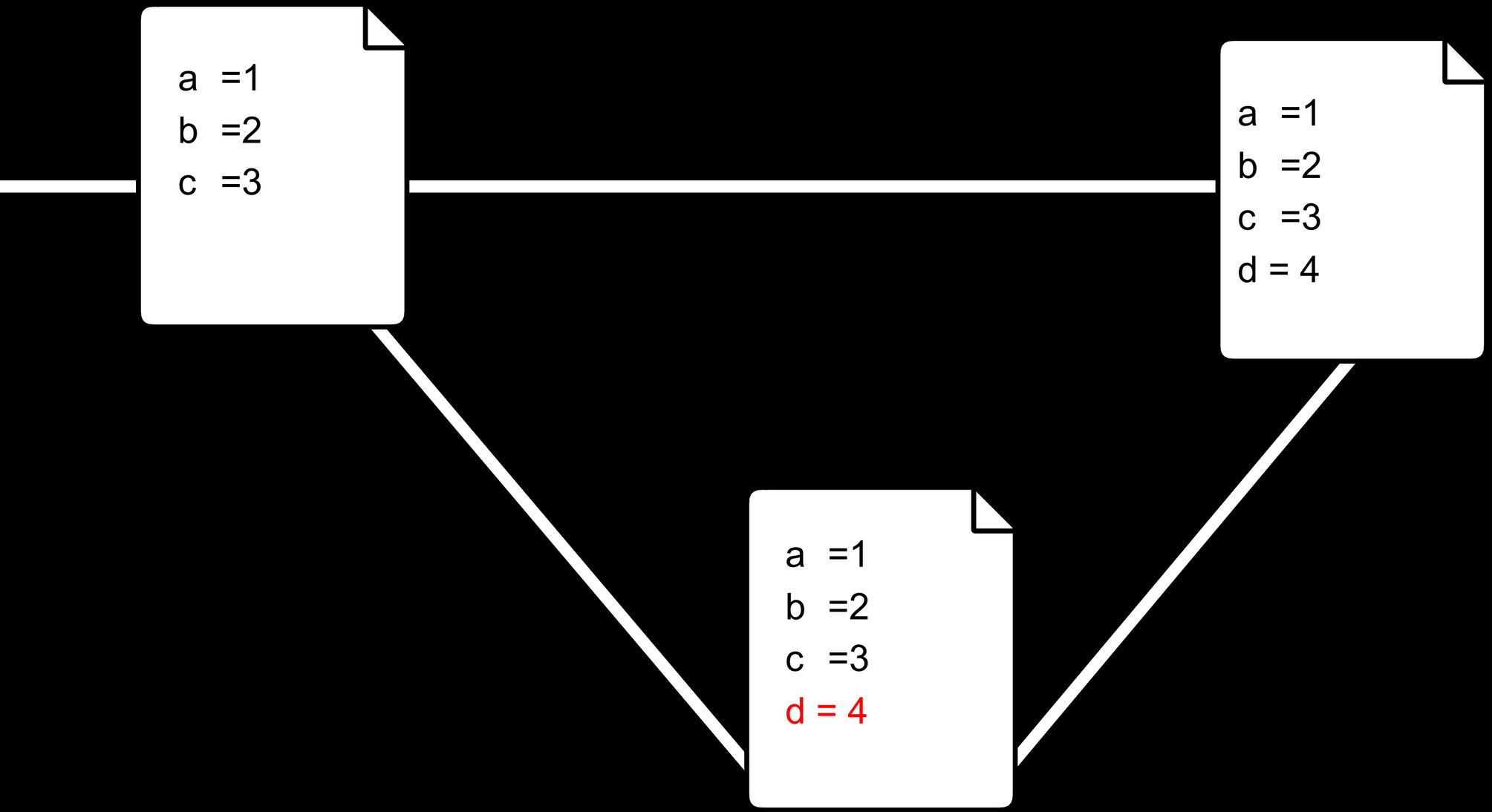
Test changes to code without losing the original.



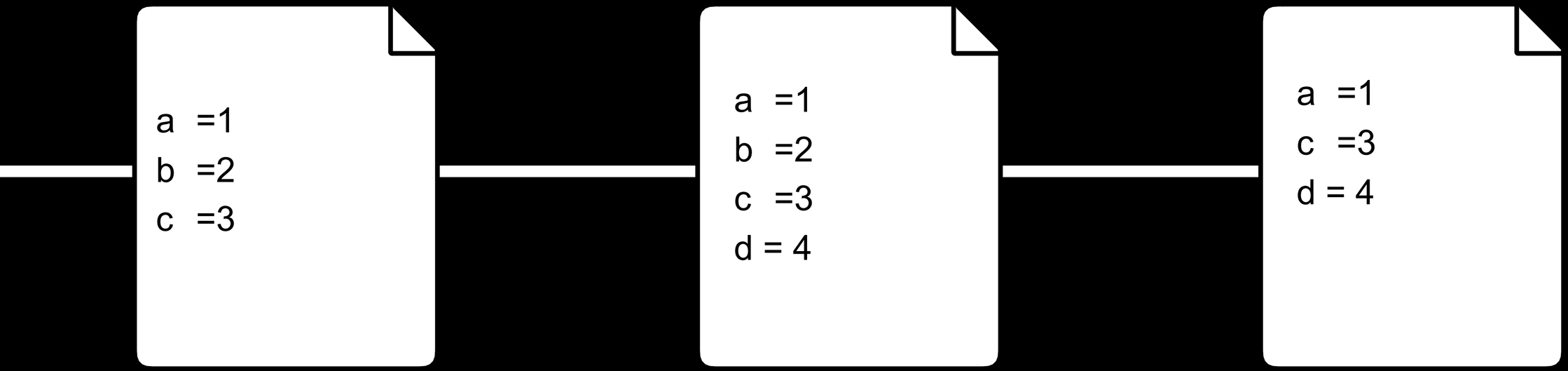
Test changes to code without losing the original.



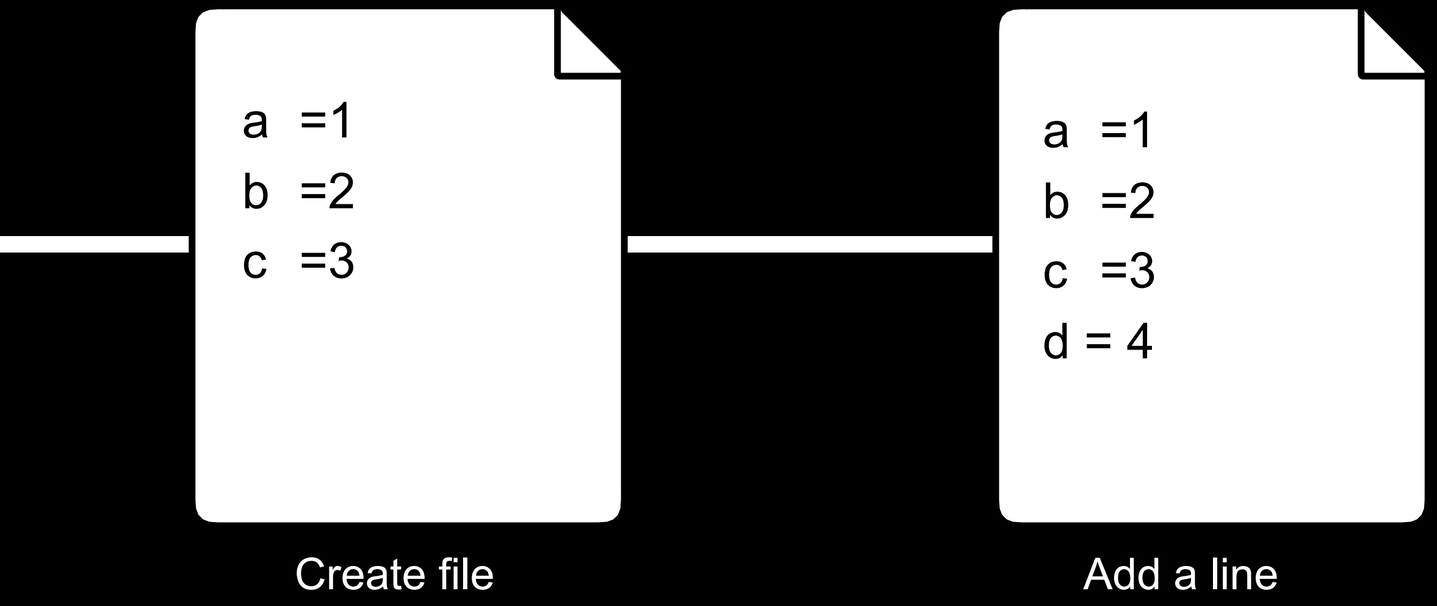
Test changes to code without losing the original.



Revert back to old versions of code.



Revert back to old versions of code.



A Git repository is the .git/ folder inside a project. This repository tracks all changes made to files in your project, building a history over time

GitHub

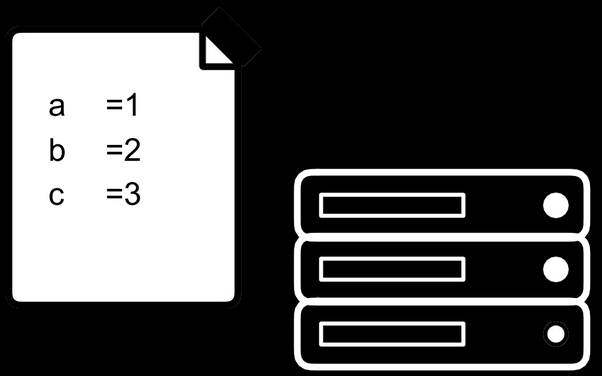
A website to store Git
Repositories on the internet

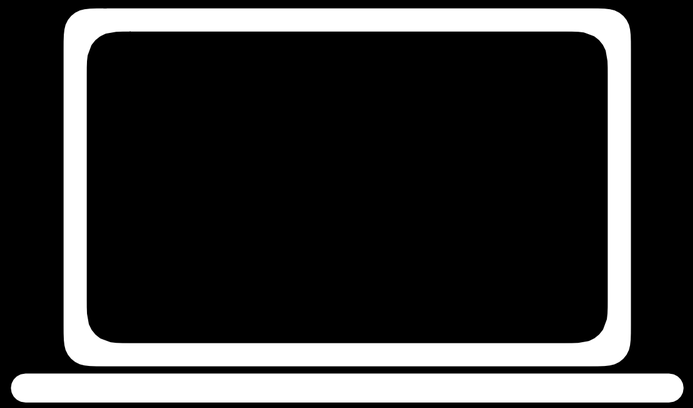
Give access people all around the world to Git Repositories

- use them
- contribute to them
- push them to Github

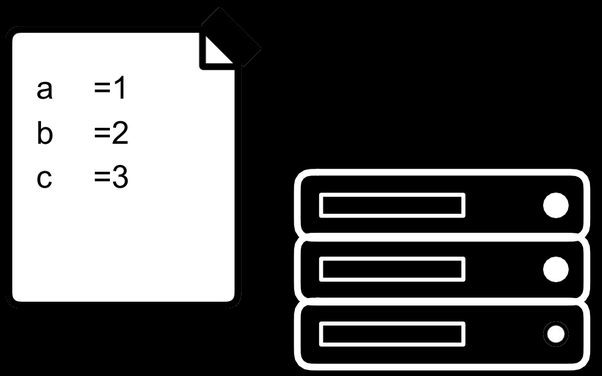
git clone

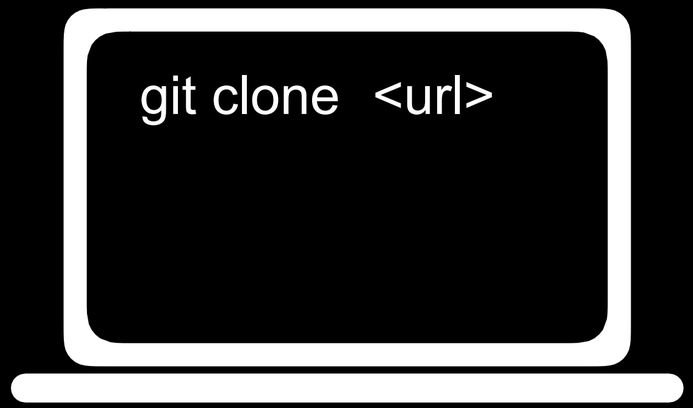
git clone <url>





git clone <url>



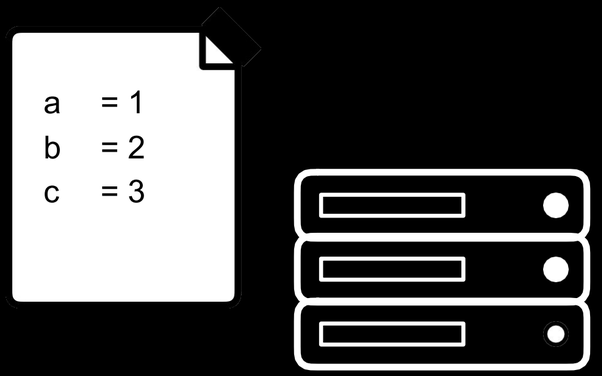


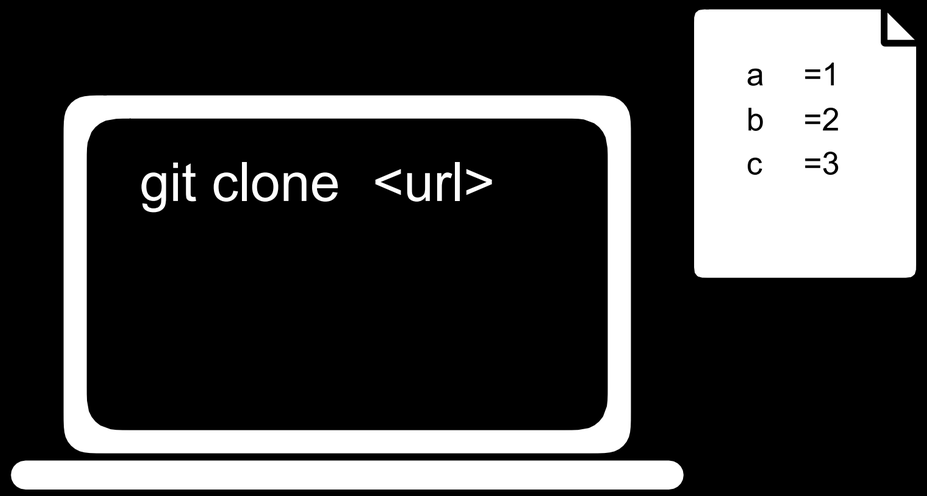
git install

```
sudo apt install git-all
```

Installing Git

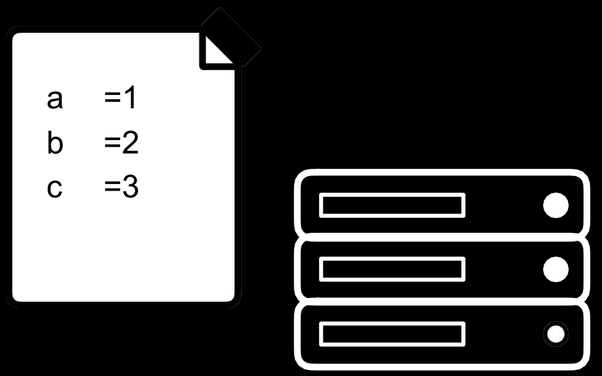
git clone <url>

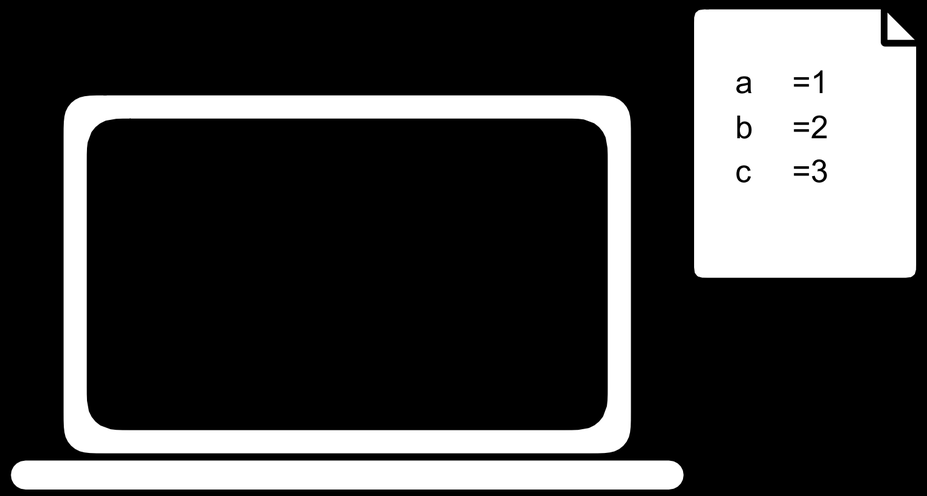




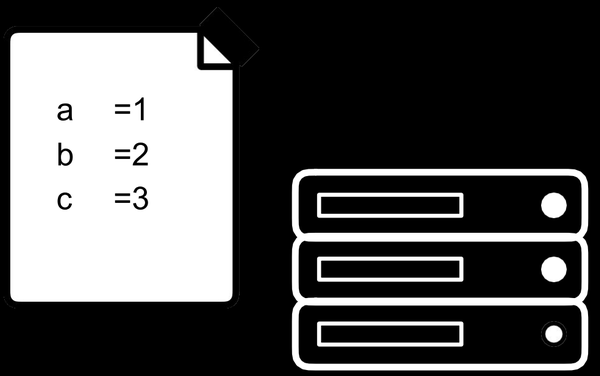
git add

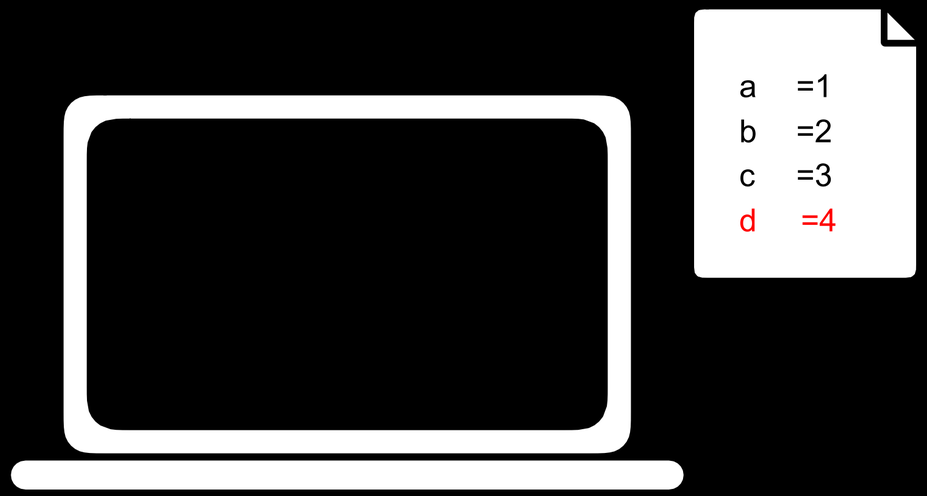
git add <filename>



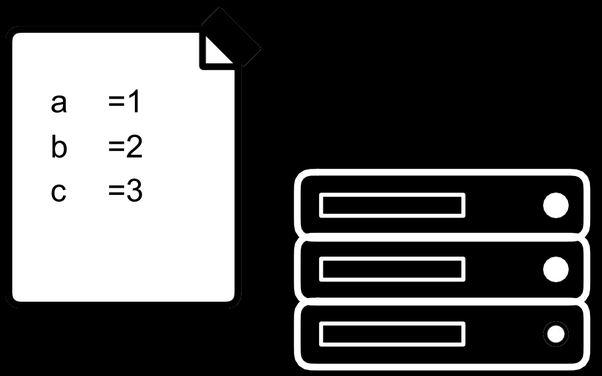


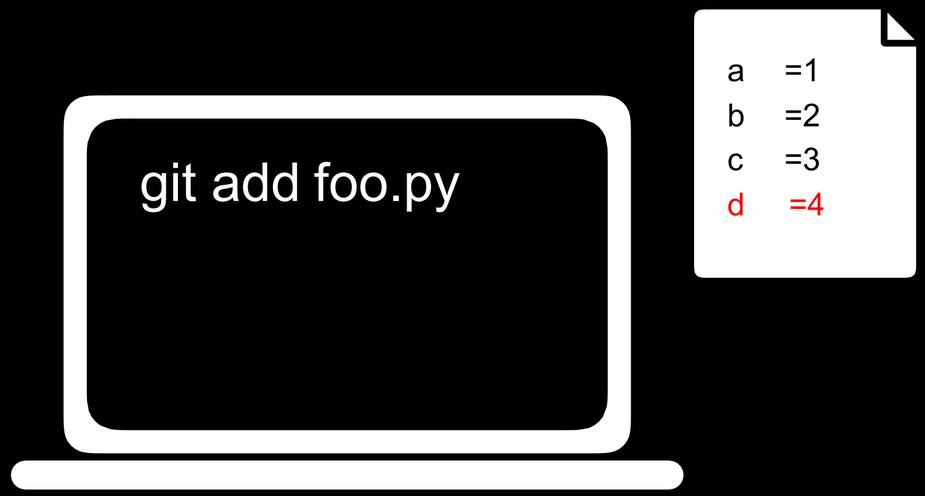
git add <filename>



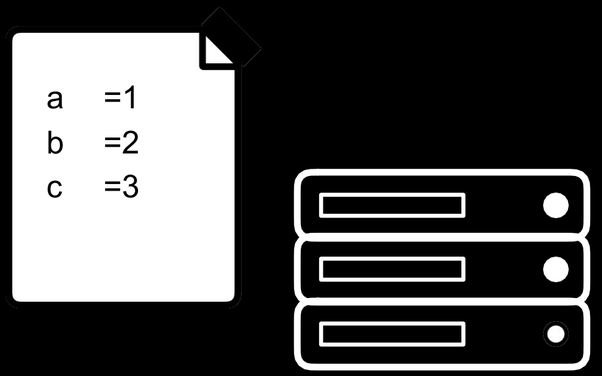


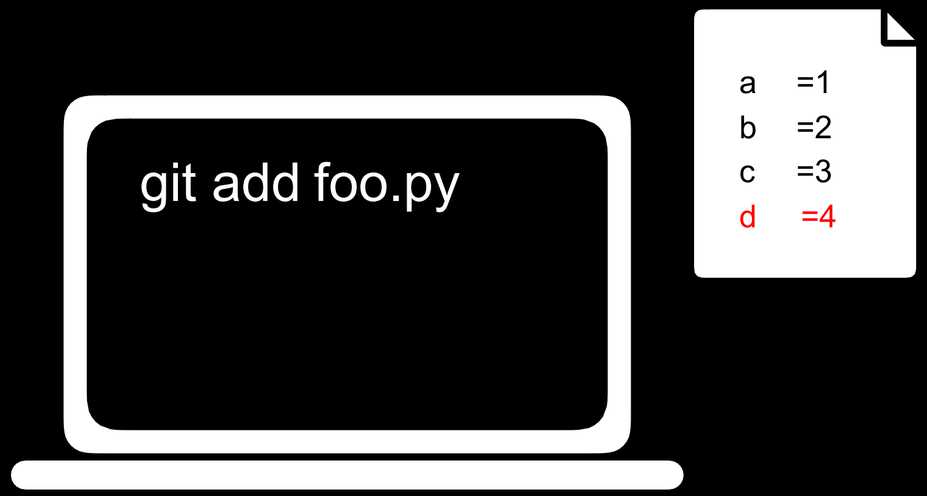
git add <filename>





git add <filename>



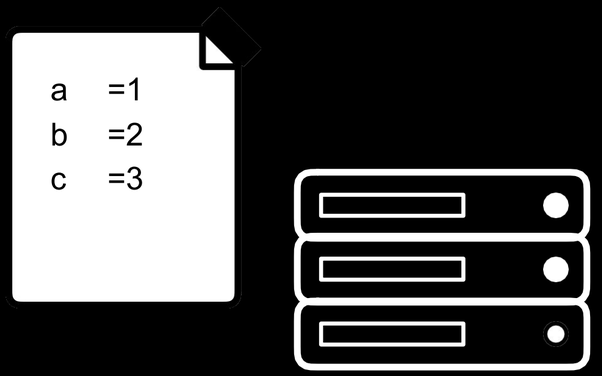


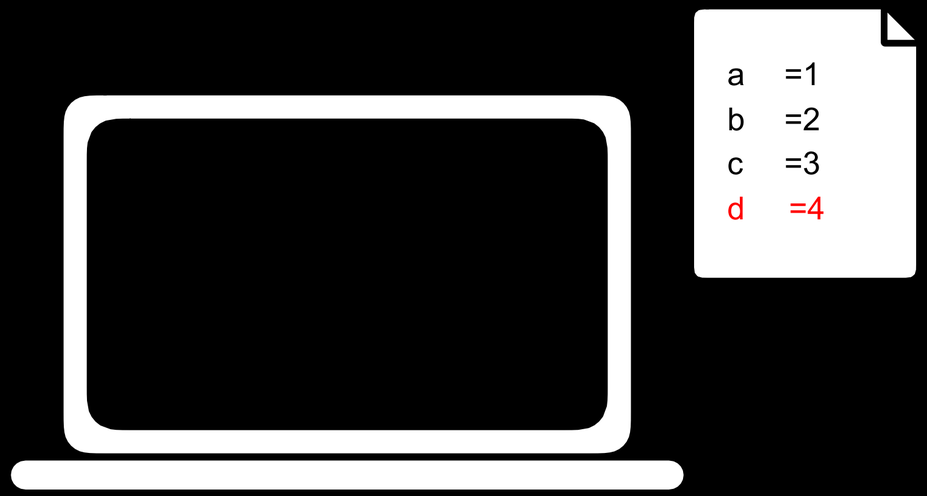
Changes to be committed:

modified: foo.py

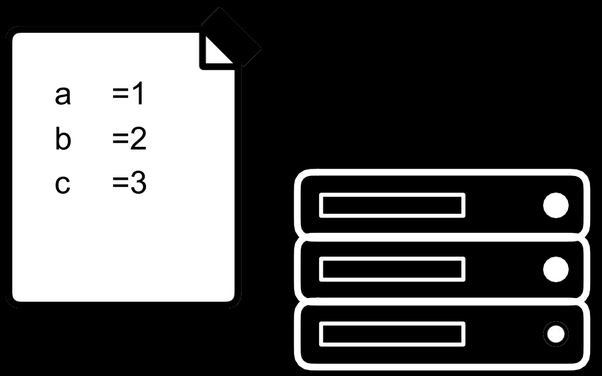
git commit

git commit -m "message"

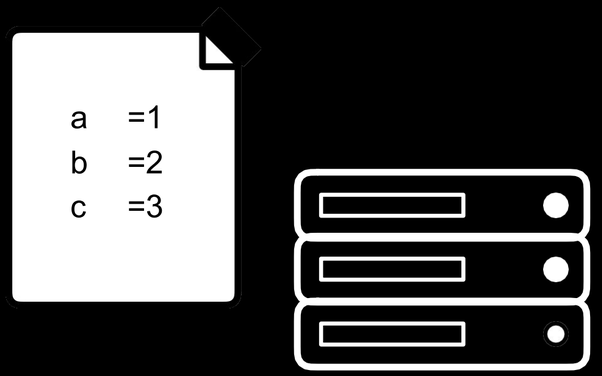


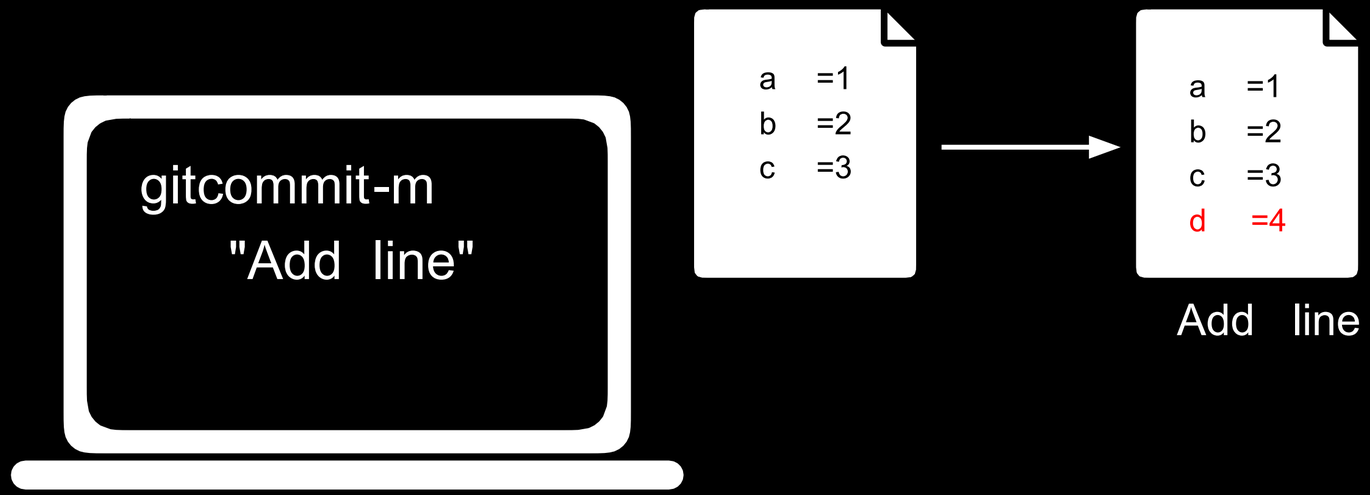


git commit -m "message"



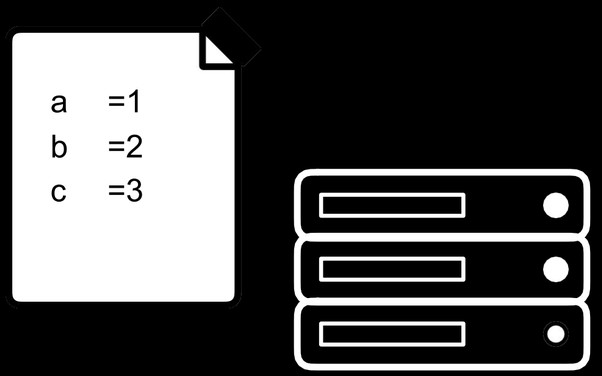
git commit -m "message"

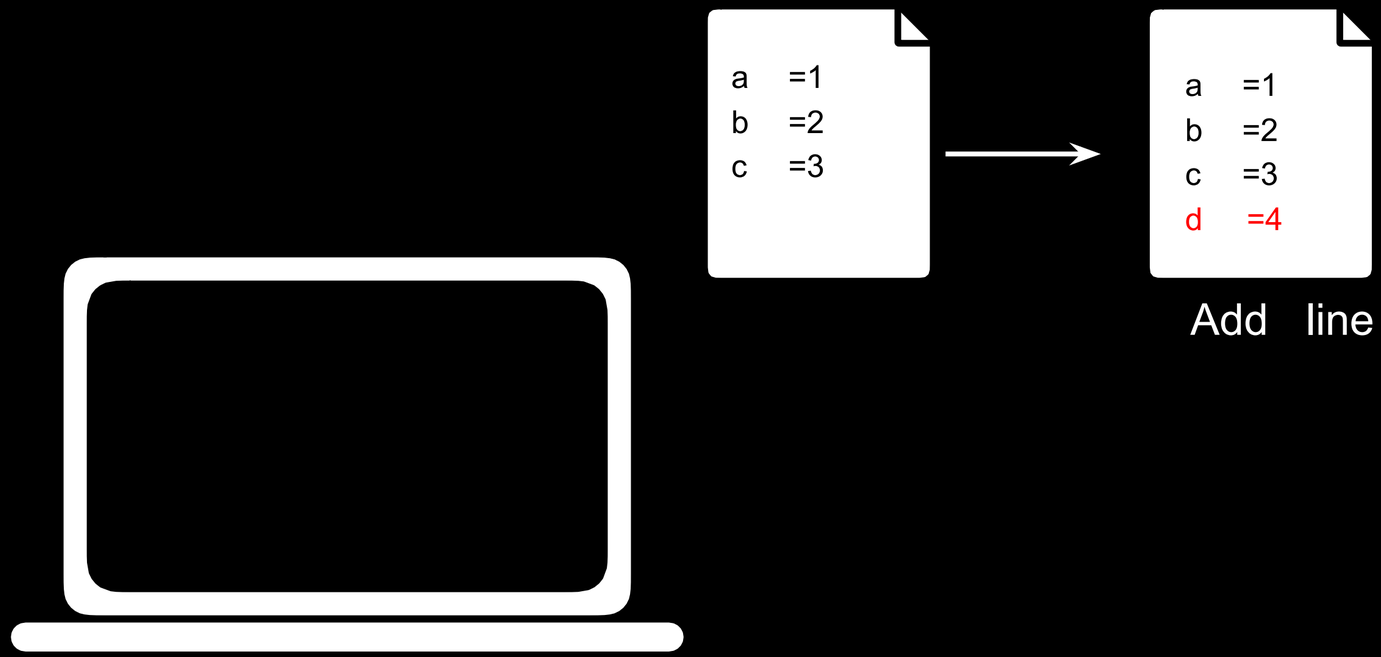




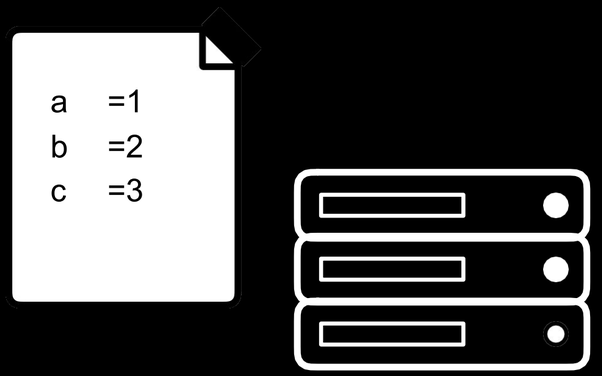
git status

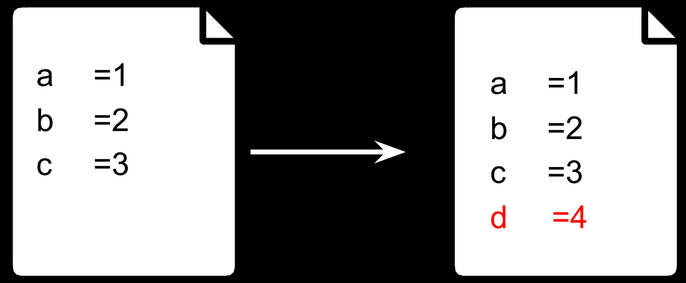
git status



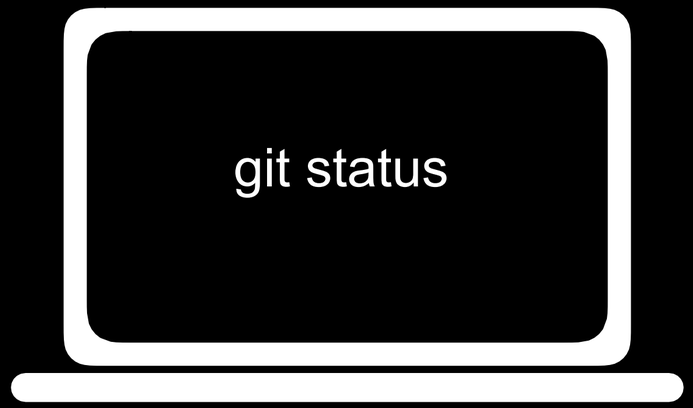


git status

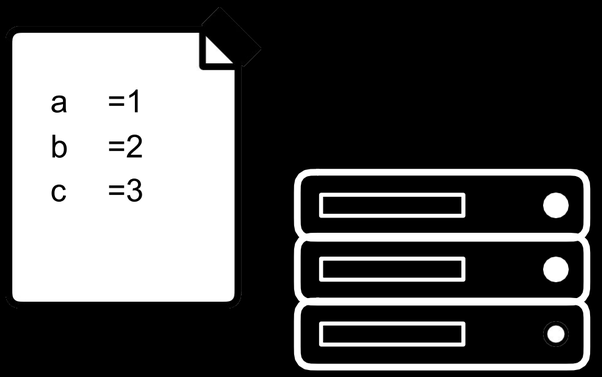


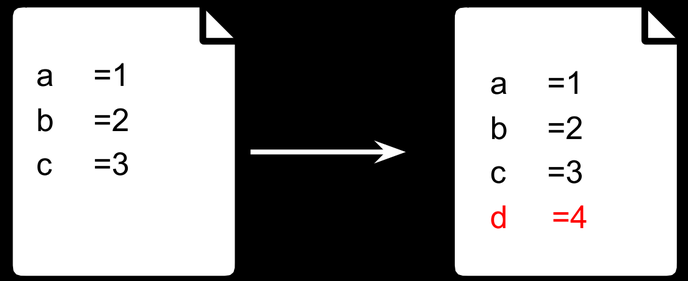


Add line

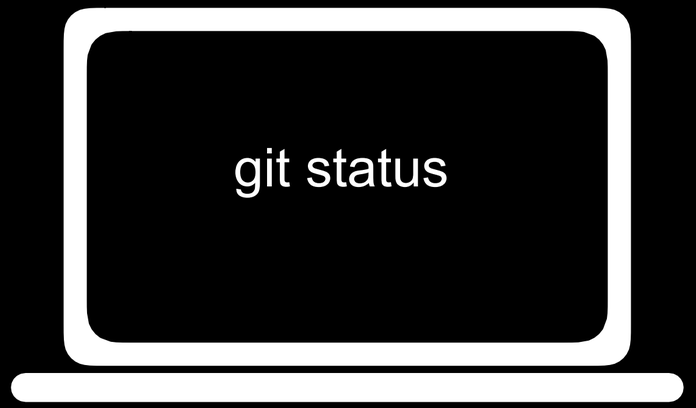


git status





Add line

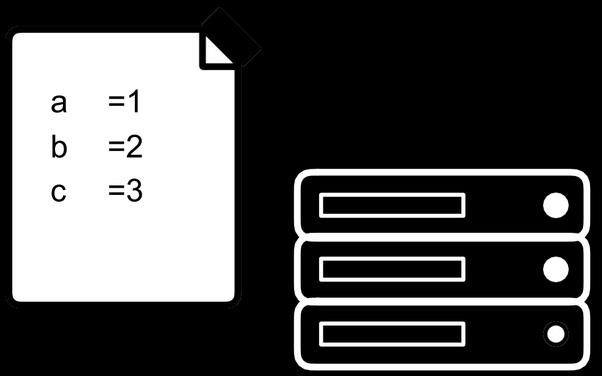


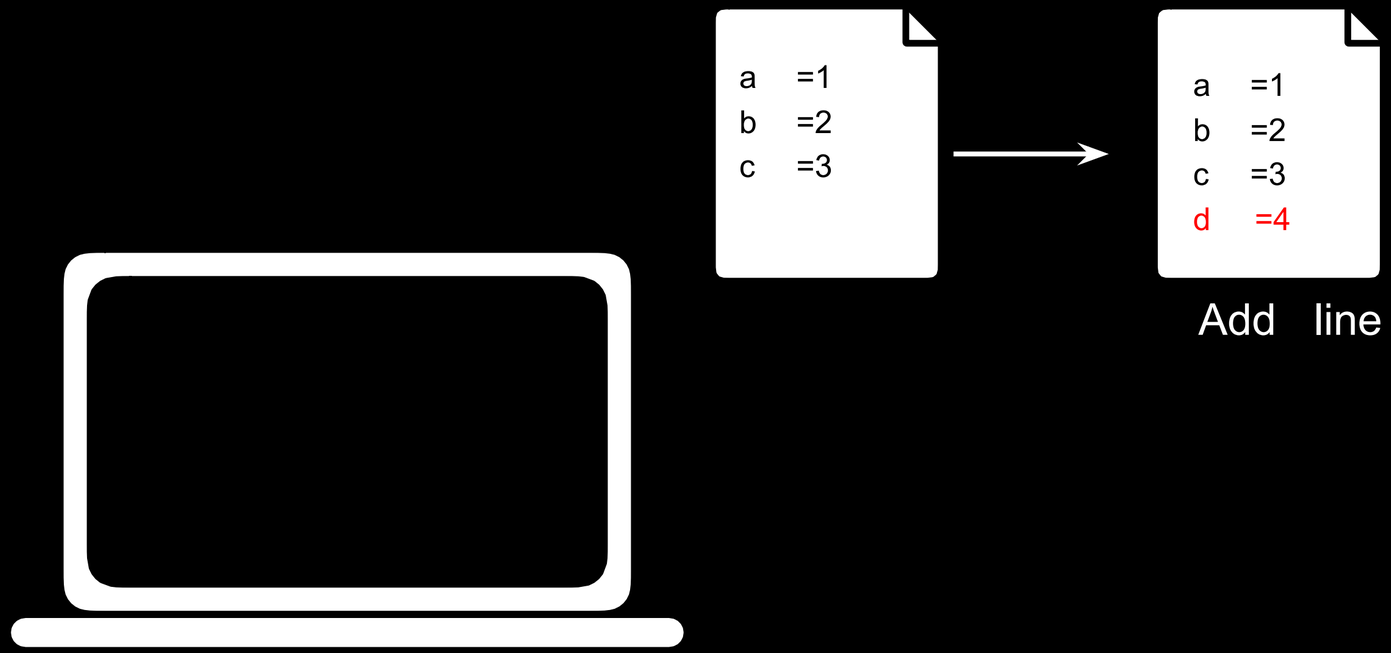
On branch master

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

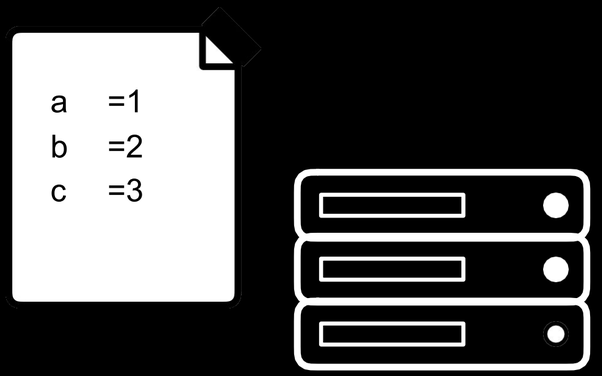
git push

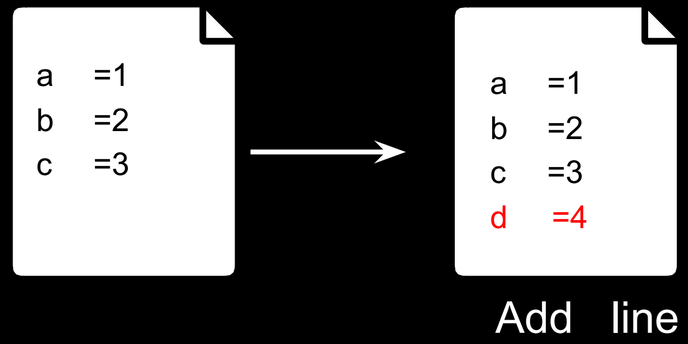
git push

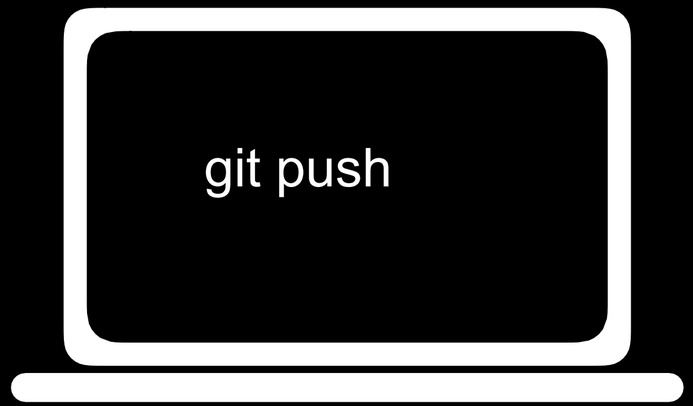




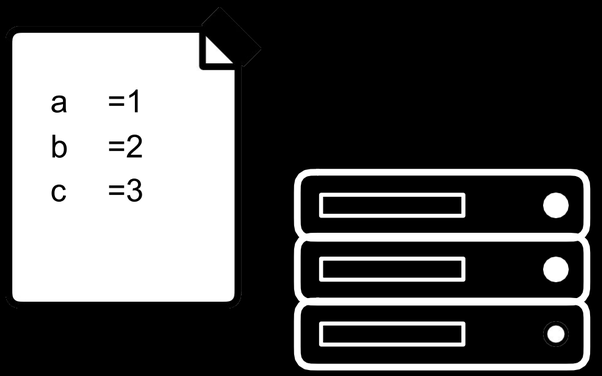
git push

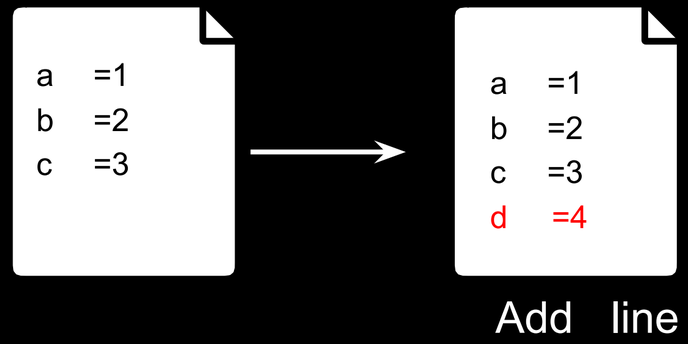


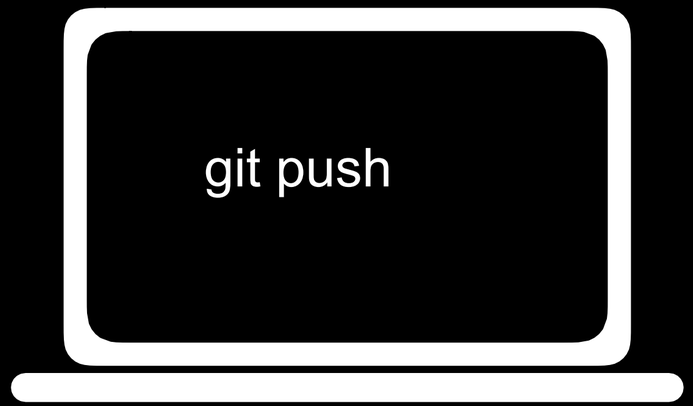




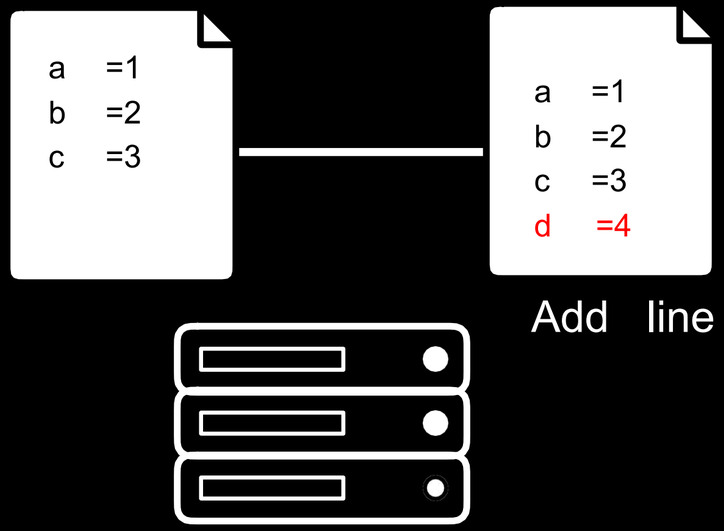
git push

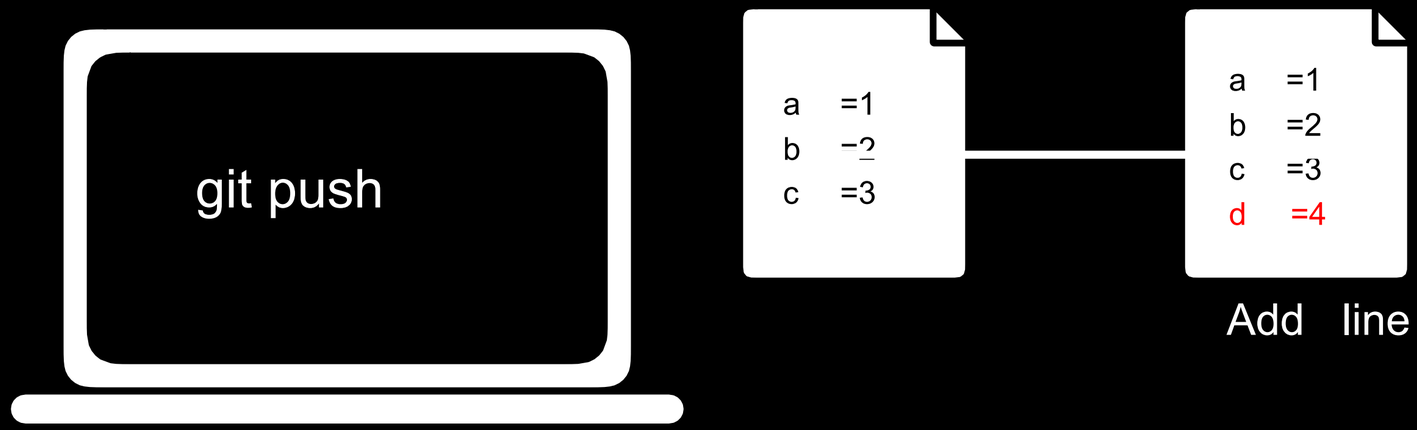






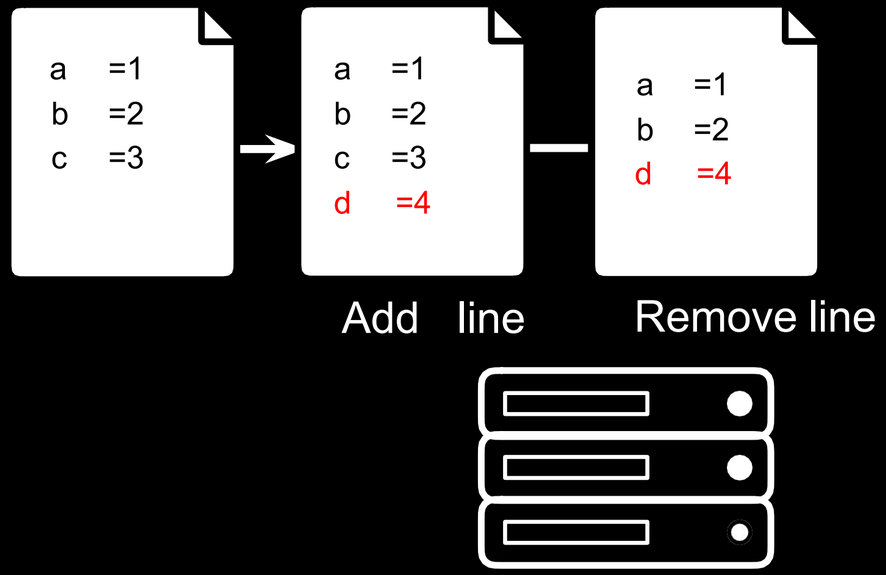
git push

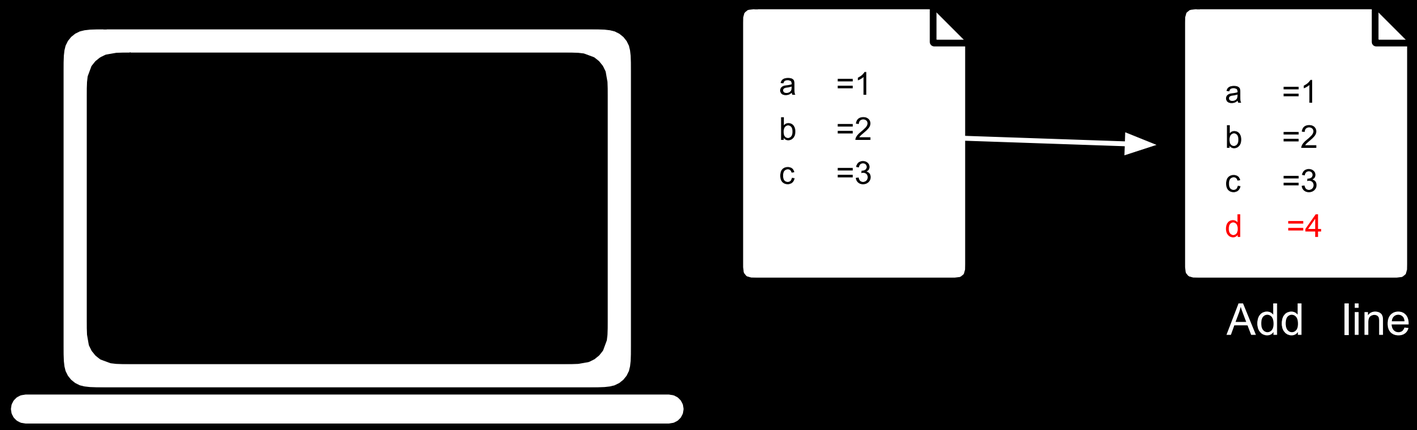




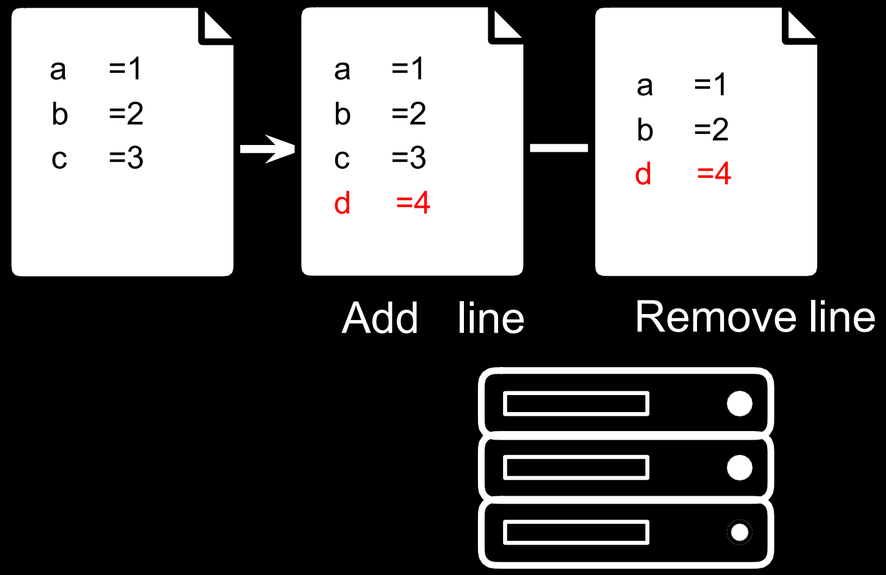
git pull

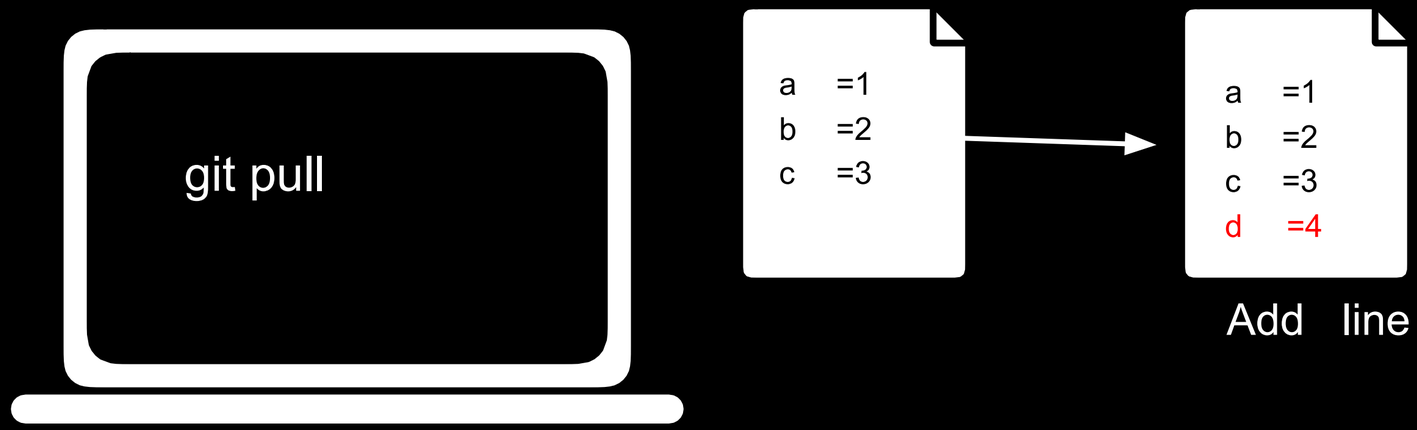
git pull


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

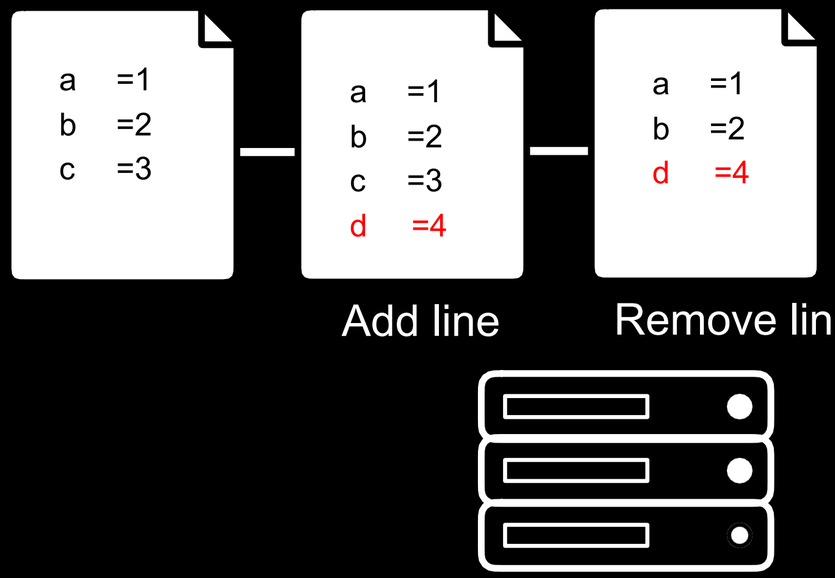

a =1
b =2
c =3
a =1
b =2
c =3
d =4
Add line

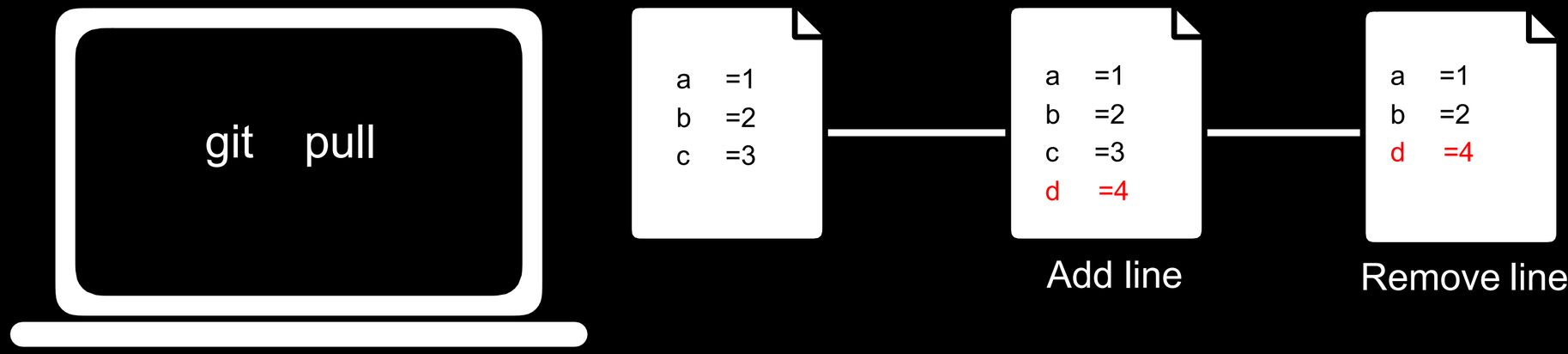
git pull


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


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

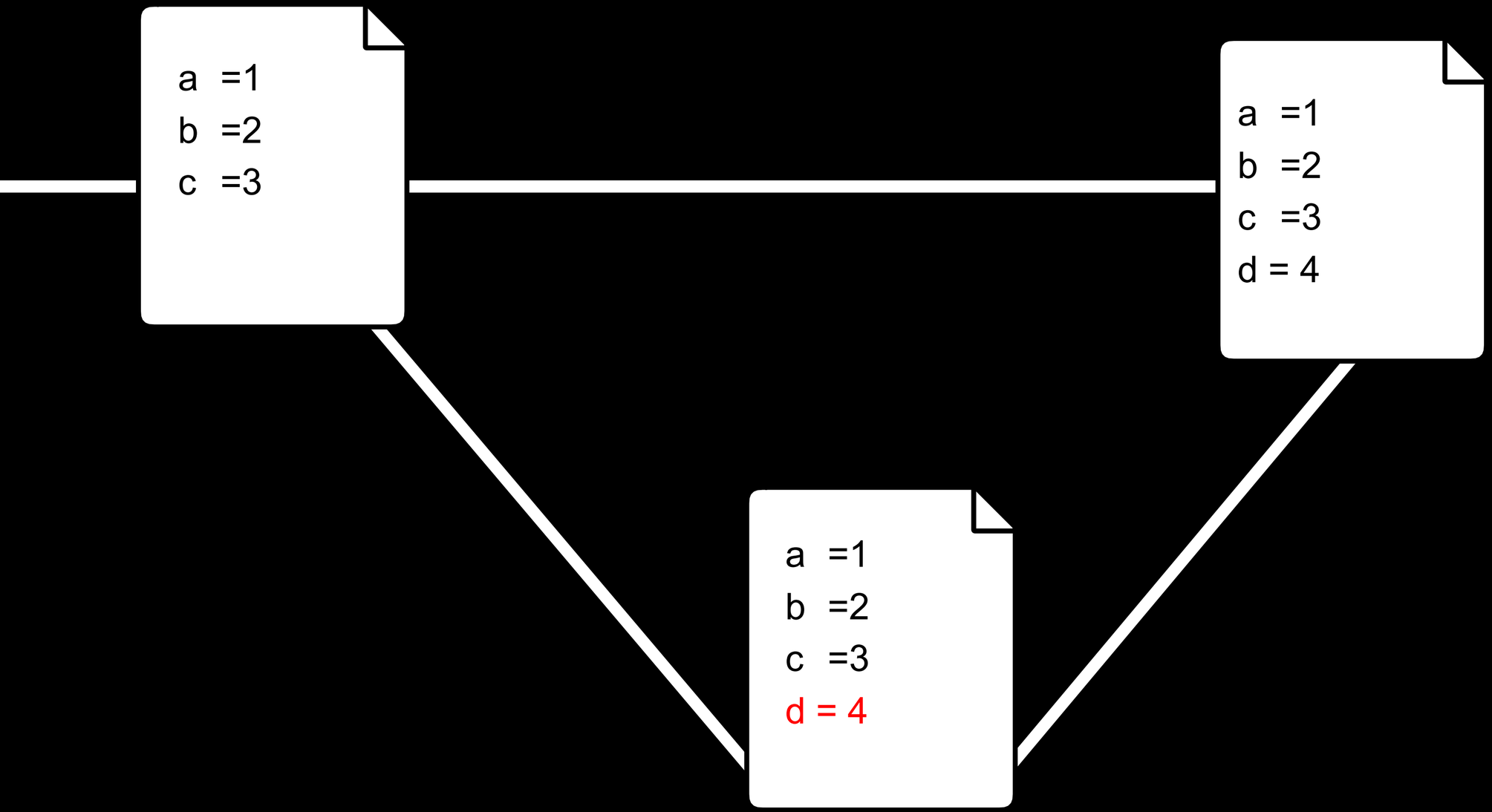
git pull

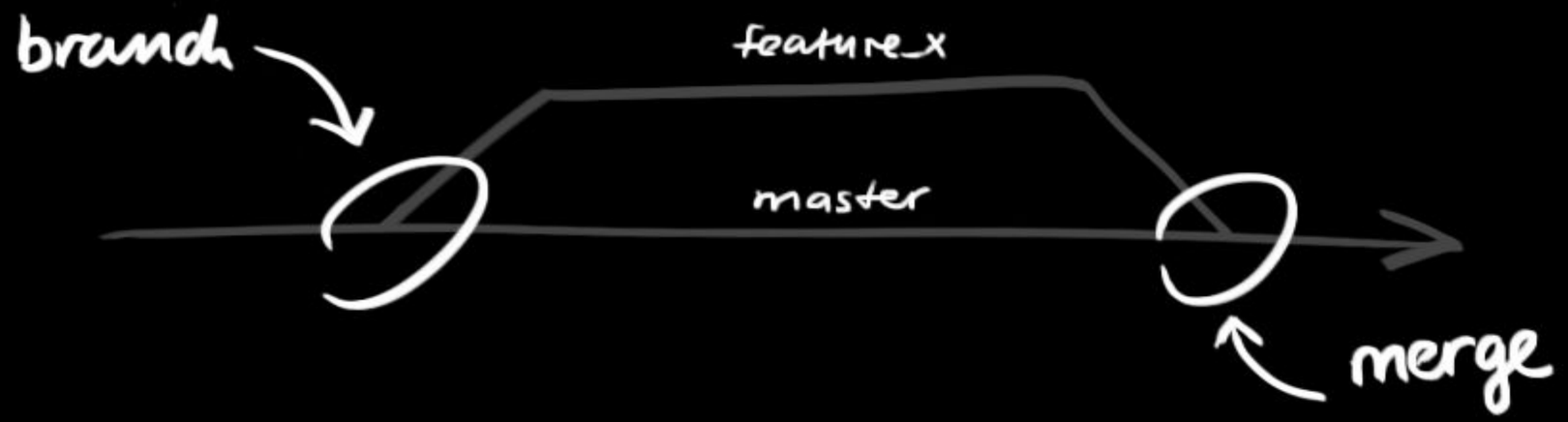

a =1
b =2
c =3
a =1
b =2
c =3
d =4
a =1
b =2
d =4
Add line
Remove line


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

git branching

Test changes to code without losing the original.




branch
feature_x
master
merge

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.

- Create an new branch named “feature_x” and switch to it using

```
git checkout -b  feature_x
```

- Switch back to master

```
git checkout master
```

- Merge new feature from “feature_x” branch to master

```
git merge feature_x
```

- Delete the branch again

```
git branch -d  feature_x
```

- Push the branch to remote repository to get available to others

```
git push origin <branch>
```

git log

git log

git log

git log

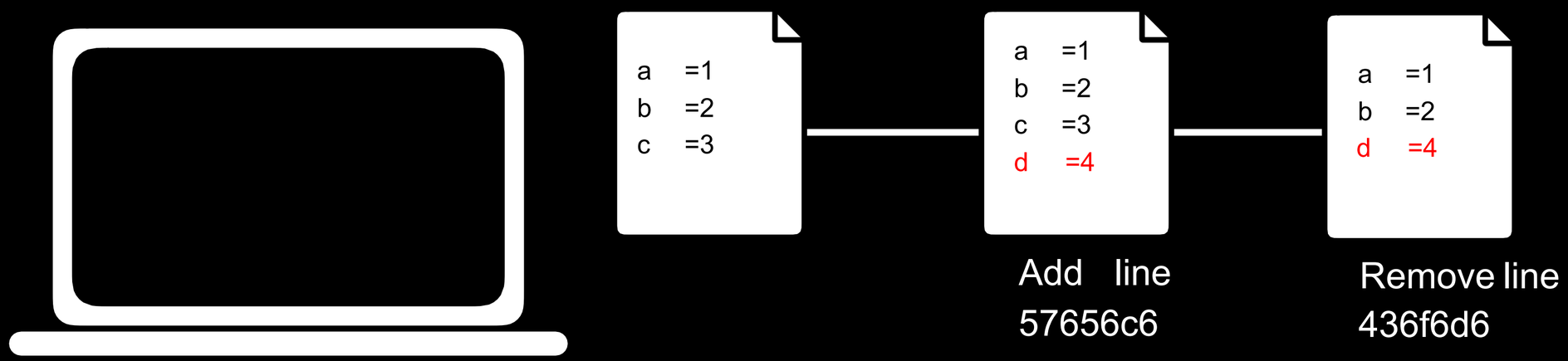

```
commit 436f6d6d6974204d73672048657265
Author: Brian Yu <brian@cs.harvard.edu>
Date: Mon Jan 22 14:06:28 2018 -0400

    Remove a line

commit57656c636f6d6520746f20576562
Author:Brian Yu <brian@cs.harvard.edu>
Date: Mon Jan 22 14:05:28 2018 -0400

    Add a line
```

git reset

git reset

git reset --hard <commit>

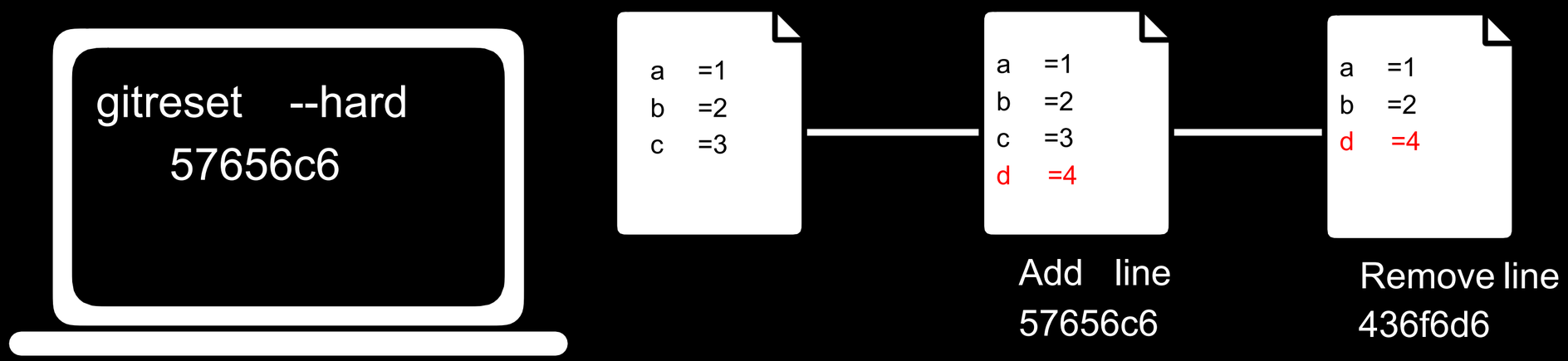
git reset --hard origin/master



git reset

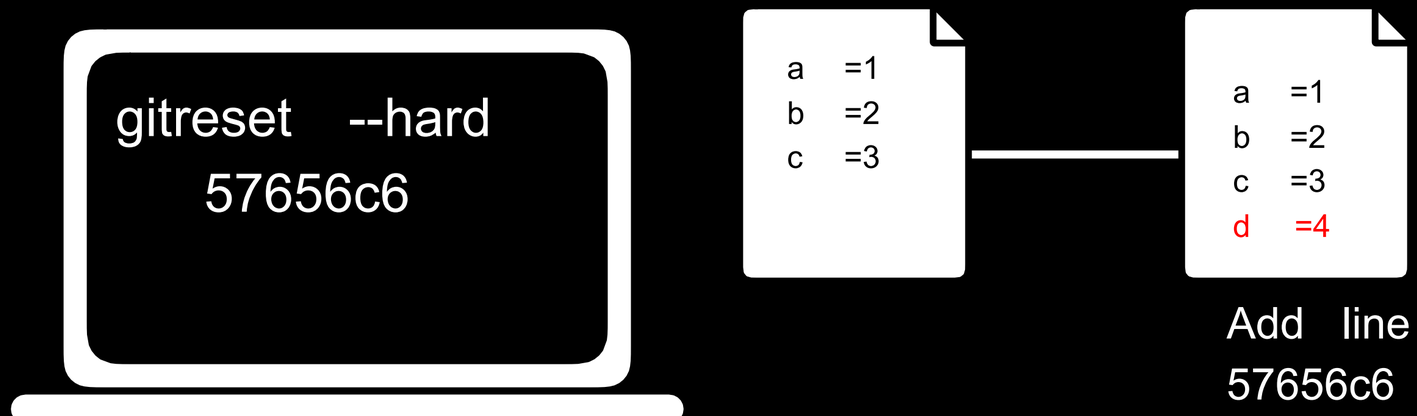
git reset --hard <commit>

git reset --hard origin/master



git reset

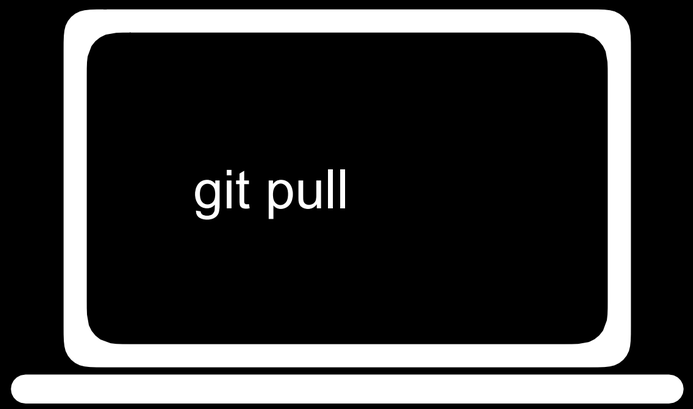
- git reset --hard <commit>
- git reset --hard origin/master



Merge Conflicts

Merge Conflicts

Merge Conflicts



Merge Conflicts


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

Merge Conflicts


```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
```

Merge Conflicts

your change

remote change

```
a = 1
<<<<< HEAD
b = 2
=====
b = 0
>>>>> 57656c636f6d6520746f20576562
```

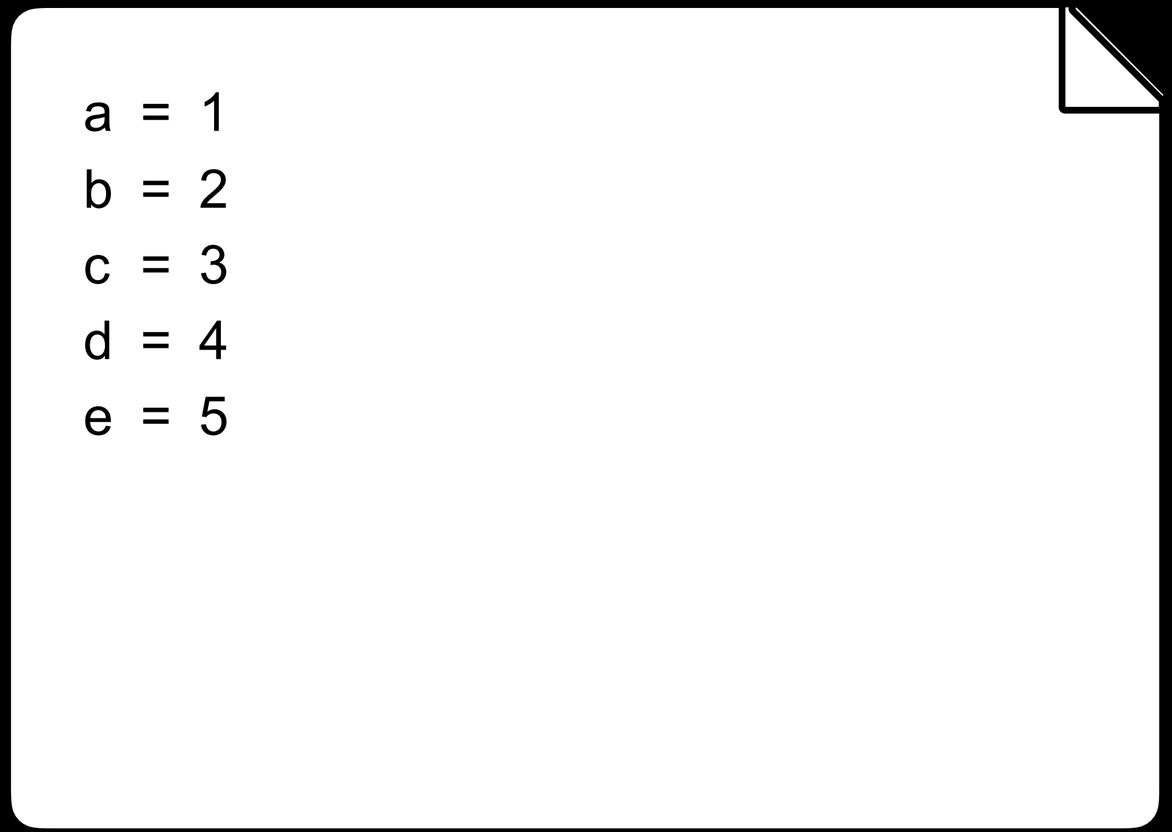
conflicHng commit

Merge Conflicts

a = 1

b = 2

Merge Conflicts

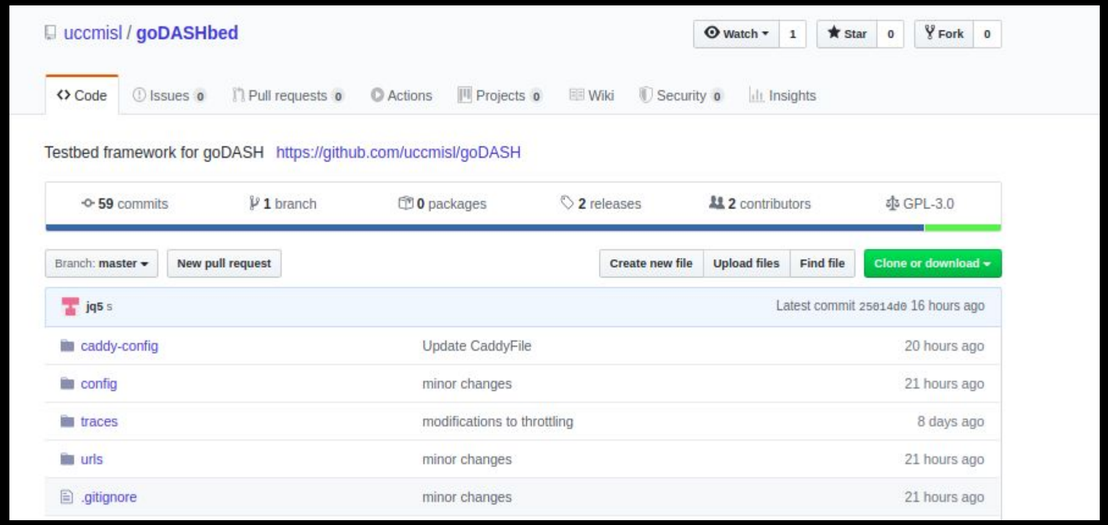


GitHub vs GitLab

Features	GitHub	GitLab
Released	April 2008	September 2011
Free plans	Free for public and private repositories	Unlimited public and private repositories
Code review	Yes	Yes
Bug and Issue tracking	Yes	Yes
Pulling changes	Pull request	Merge Request
Export/Import project	No Contains Only git repository including issues, contribution metrics, etc	Yes Project is a container including all Git repositories, discussion, project specific settings, and much more

README as Shadow Documentation

You can add a README file to your repository to tell other people why your project is useful, what they can do with your project, and how they can use it.



README.md

goDashbed Application

Current release version : 1.1

We kindly ask that should you mention godash or goDASHbed, or use our code, in your publication, that you would reference the following paper:

D. Raca, M. Manifacier, and J.J. Quinlan. goDASH - GO accelerated HAS framework for rapid prototyping. 12th International Conference on Quality of Multimedia Experience (QoMEX), Athlone, Ireland. 26th to 28th May, 2020 CORA (To Appear)

General Description

GoDASHbed is a highly customizable framework for realistic large scale experimentation with two different types of supported traffic:

```
-- Video: HTTP Adaptive Streaming traffic with support for two transportation modes TCP and QUIC
-- VoIP: Realistic VoIP traffic generation (through D-ITG traffic generator[1])
```

Requirements:

How Code in GitHub Helps to Reproduce Work

goDASH - GO accelerated HAS framework for rapid prototyping

Darijo Raca
Faculty of Electrical Engineering
University of Sarajevo, Sarajevo, BiH
draca@etf.unsa.ba

Maëlle Manifacier
Université Clermont Auvergne
Aubière Cedex, France
maelle.manifacier@etu.uca.fr

Jason J. Quinlan
Department of Computer Science
University College Cork, Ireland
j.quinlan@cs.ucc.ie

goDASH also comes equipped with it's own testbed framework, known as *goDASHbed* (https://github.com/uccmisl/goDASHbed.git). This framework utilises Mininet for network emulation and permits streaming

uccmisl / goDASHbed

Watch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security 0 | Insights

Testbed framework for goDASH https://github.com/uccmisl/goDASH

59 commits | 1 branch | 0 packages | 2 releases | 2 contributors | GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jq5 s — Latest commit 25014d0 16 hours ago

caddy-config	Update CaddyFile	20 hours ago
config	minor changes	21 hours ago
traces	modifications to throttling	8 days ago
urls	minor changes	21 hours ago
.gitignore	minor changes	21 hours ago

Install Steps

The easiest way to install goDASHbed is to use the install script available at the UCC Mobile and Internet System Lab MISL

After goDASHbed has been installed, follow these steps for all required dependencies

download dash content to `<content_folder>`, using the `get_your_movies.sh` bash script. Then move the content to your system web folder

```
sudo mv <content_folder> /var/www/html
```

Then change the user permissions on this web folder

```
sudo adduser $USER www-data
sudo chown $USER:www-data -R /var/www
sudo chmod u=rwX,g=srX,o=rX -R /var/www
```

update the url lists in `urls/mpdURL.py` to reflect the content downloaded

```
/var/www/html/<folder> -> "http://www.godashbed.org/<folder>"
```

You may also need to update the `configure.json` file in goDASHbed/config, and change url to point to the content downloaded

README.md

goDashbed Application

Current release version : 1.1

We kindly ask that should you mention godash or goDASHbed, or use our code, in your publication, that you would reference the following paper:

D. Raca, M. Manifacier, and J.J. Quinlan. goDASH - GO accelerated HAS framework for rapid prototyping. 12th International Conference on Quality of Multimedia Experience (QoMEX), Athlone, Ireland. 26th to 28th May, 2020 CORA (To Appear)

General Description

GoDASHbed is a highly customizable framework for realistic large scale experimentation with two different types of supported traffic:

```
-- Video: HTTP Adaptive Streaming traffic with support for two transportation modes TCP and QUIC
-- VoIP: Realistic VoIP traffic generation (through D-ITG traffic generator[1])
```

Requirements:

Examples to launch the app :

run godashbed on a 10Mbit link with 3 video clients for 40 seconds, with 1 VOIP client, with no debug or terminal print outs, once for each trace in the 'traces' folder, using TCP as the transport mode

```
sudo python3 ./goDashBed.py -b 10 --videoclients 3 --duration 40 --voipclients 1 --debug "off" --numruns 1 --t
```

run godashbed on a 10Mbit link with 3 video clients for 40 seconds, with 1 VOIP client, with debug or terminal print outs, once for each trace in the 'traces' folder, using QUIC as the transport mode

```
sudo python3 ./goDashBed.py -b 10 --videoclients 3 --duration 40 --voipclients 1 --debug "on" --numruns 1 --tm
```

References

- Version control
  https://en.wikipedia.org/wiki/Version_control
- Official git site and tutorial
  https://git-scm.com
- GitHub guides
  https://guides.github.com/
- Basic Tutorial
  https://realpython.com/python-git-github-intro/
- Advanced Tutorial
  https://realpython.com/advanced-git-for-pythonistas/
- GitHub vs GitLab
  https://about.gitlab.com/blog/2017/09/11/comparing-confusing-terms-in-github-bitbucket-and-gitlab/
- Slide Template and contents
  https://www.edx.org/course/cs50s-web-programming-with-python-and-javascript

Thank You

Questions?