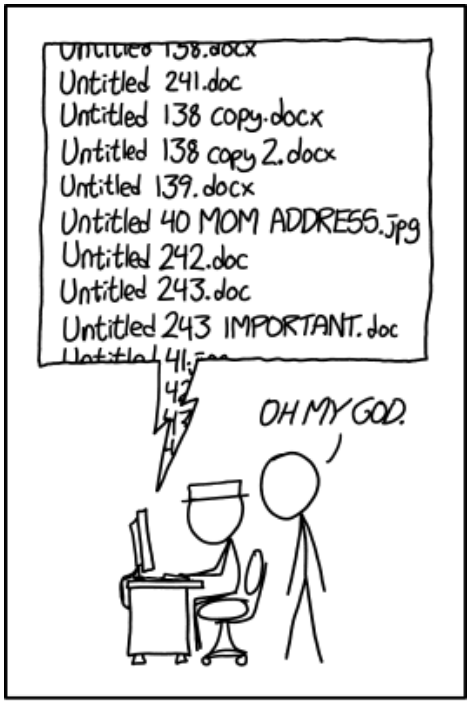
# Programming Techniques for Scientific Simulations I

Version Control

# One person working on a file

- How do you keep track of your changes?




https://xkcd.com/1459

# Two people working on the same file

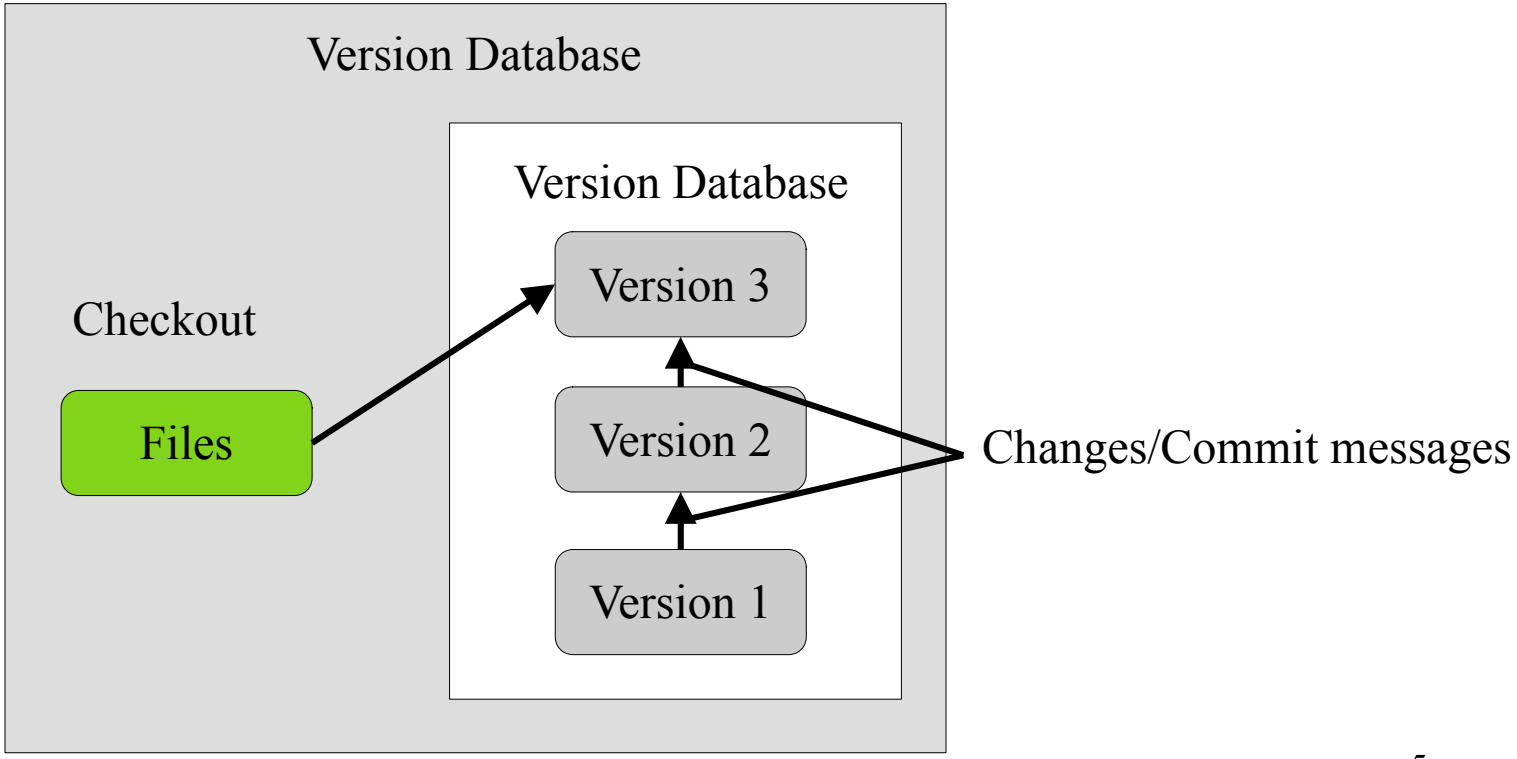
- What do we do if two people work on the same file?


https://xkcd.com/1597/

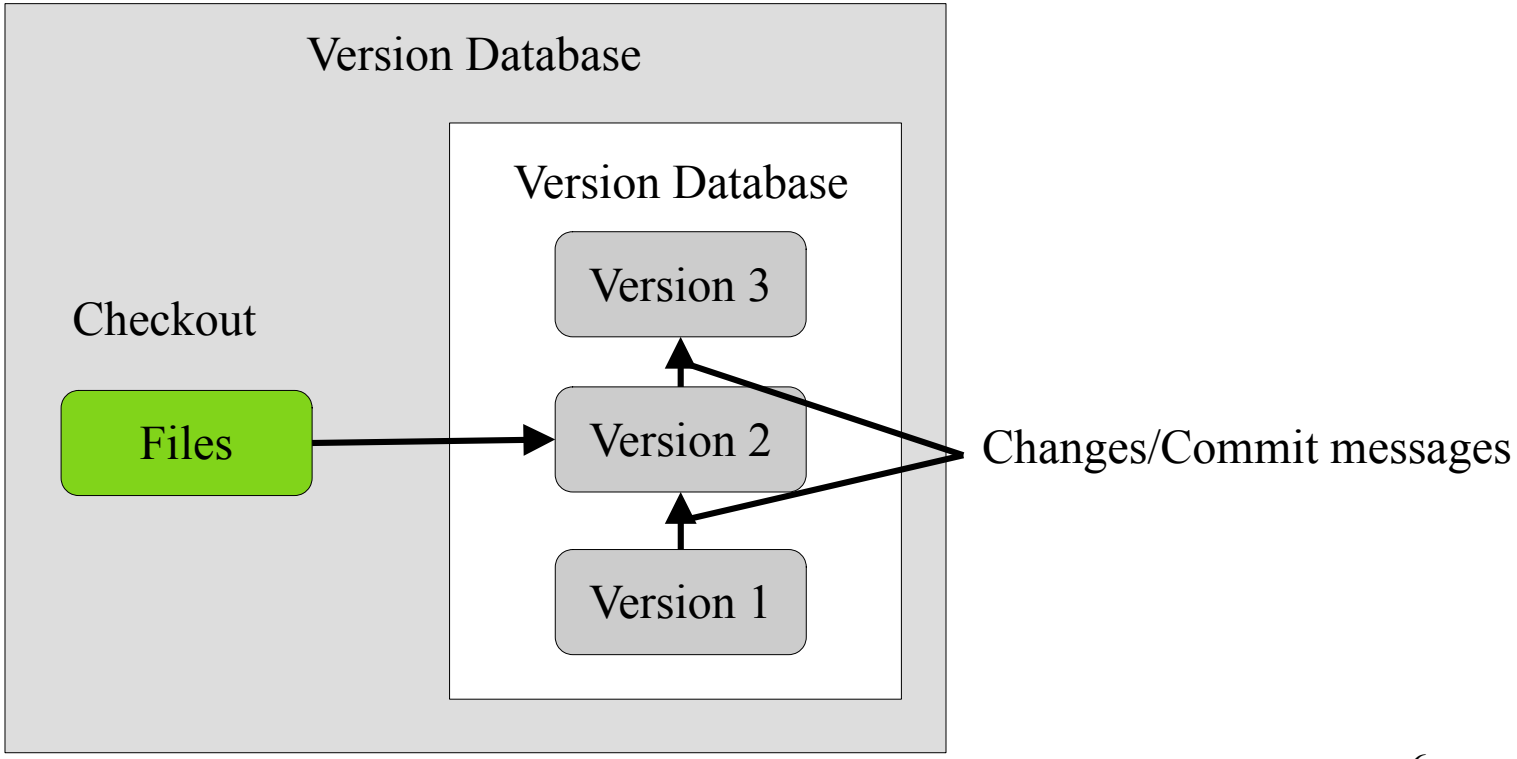
# Version Control Systems (VCS)

- There are many different version control systems, some are
  - Local, one repository on one computer
    - Revision Control System (RCS) (~1980s)
  - Centralized, one repository on a server
    - Concurrent Versioning System (CVS) (~mid 1980s)
    - SubVersioN (SVN) (~2000s)
  - Decentralized, many local repositories that can be synced
    - Git, used in this course (~2005)
    - Mercurial (Hg) (~2005)
- They can be used
  - Via command line tools
  - Via graphical user interfaces, e.g. SmartSVN and SmartGIT
  - Editor plugins (vim, emacs, VSC, ...)

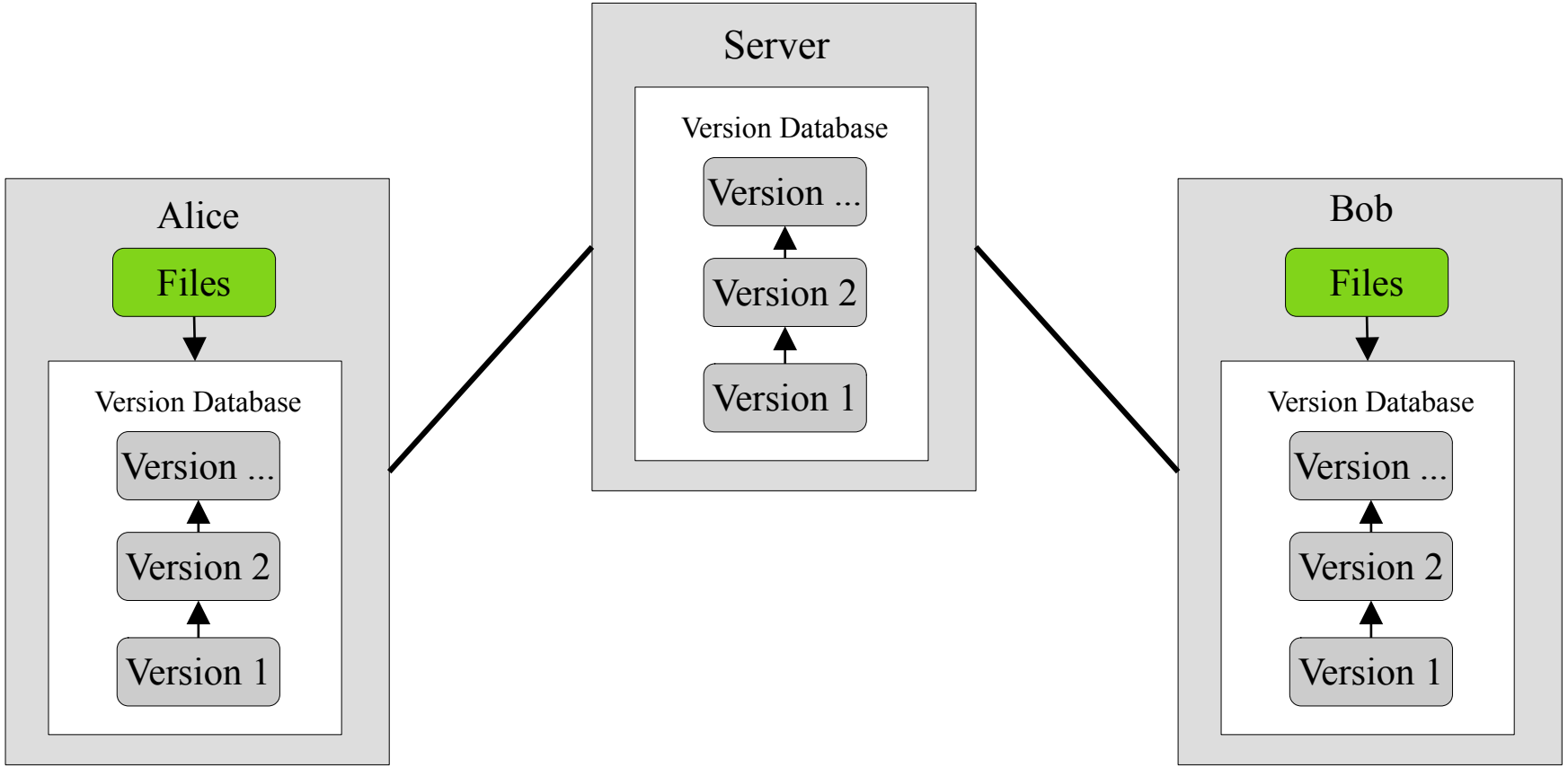
# Version Control System



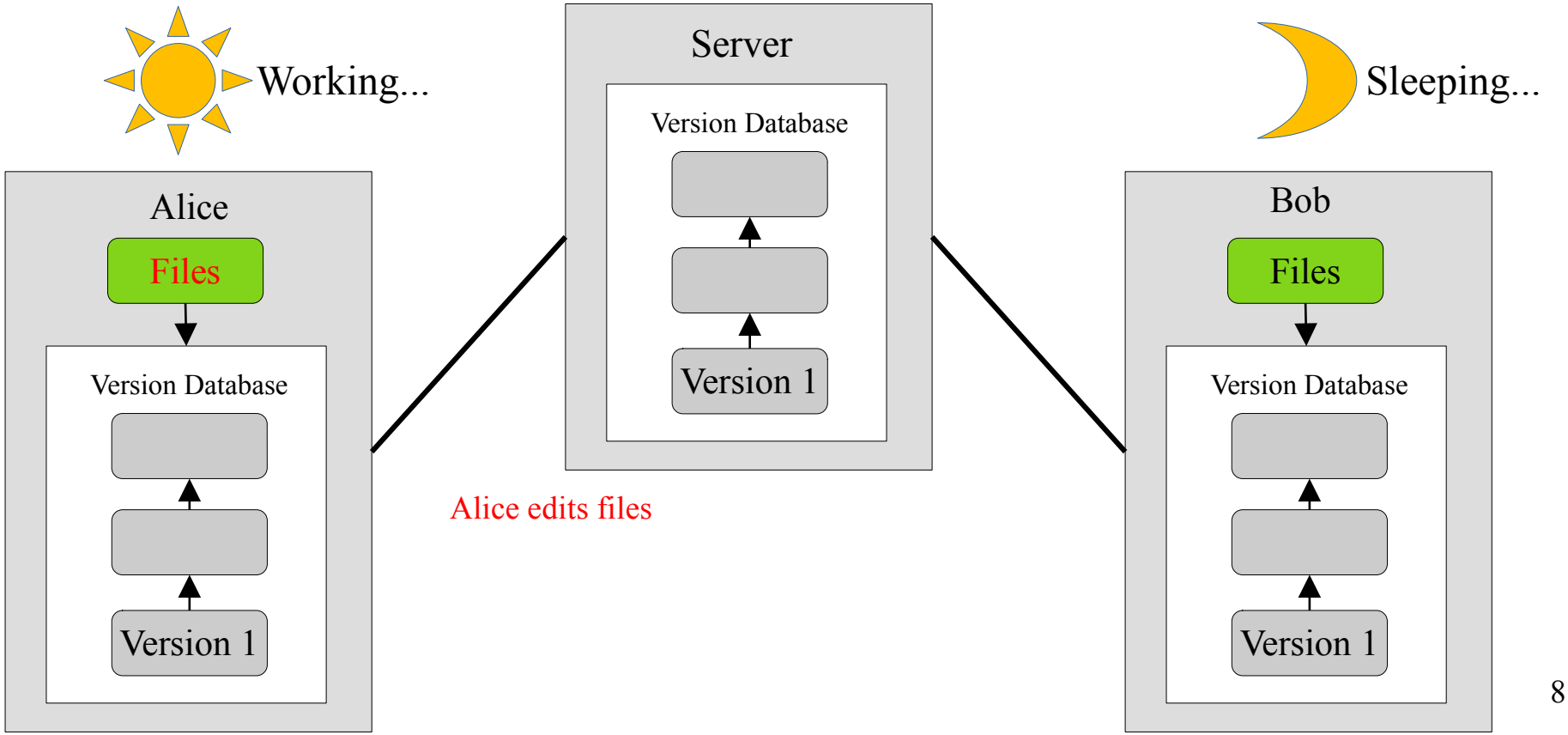
# Version Control System



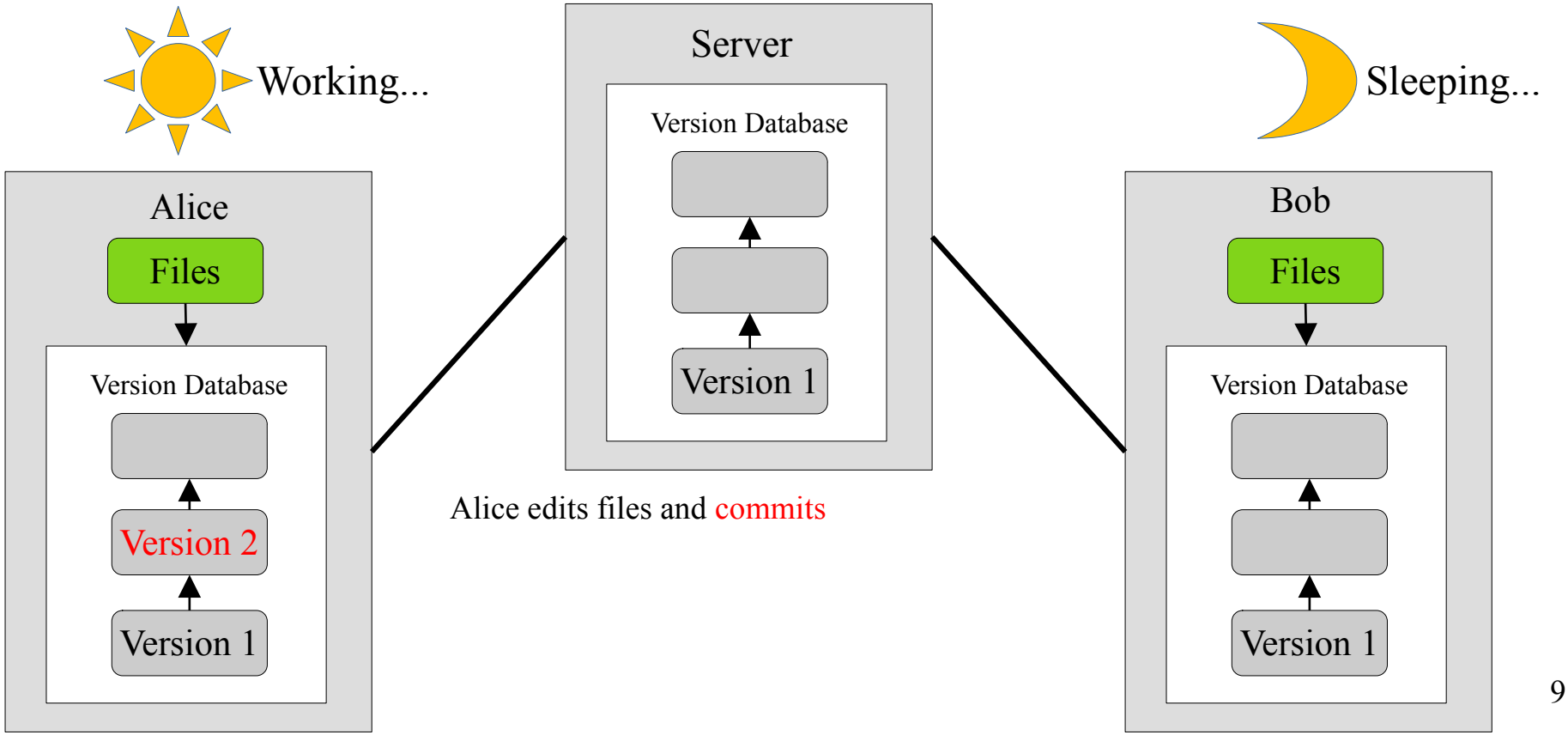
# Version Control System



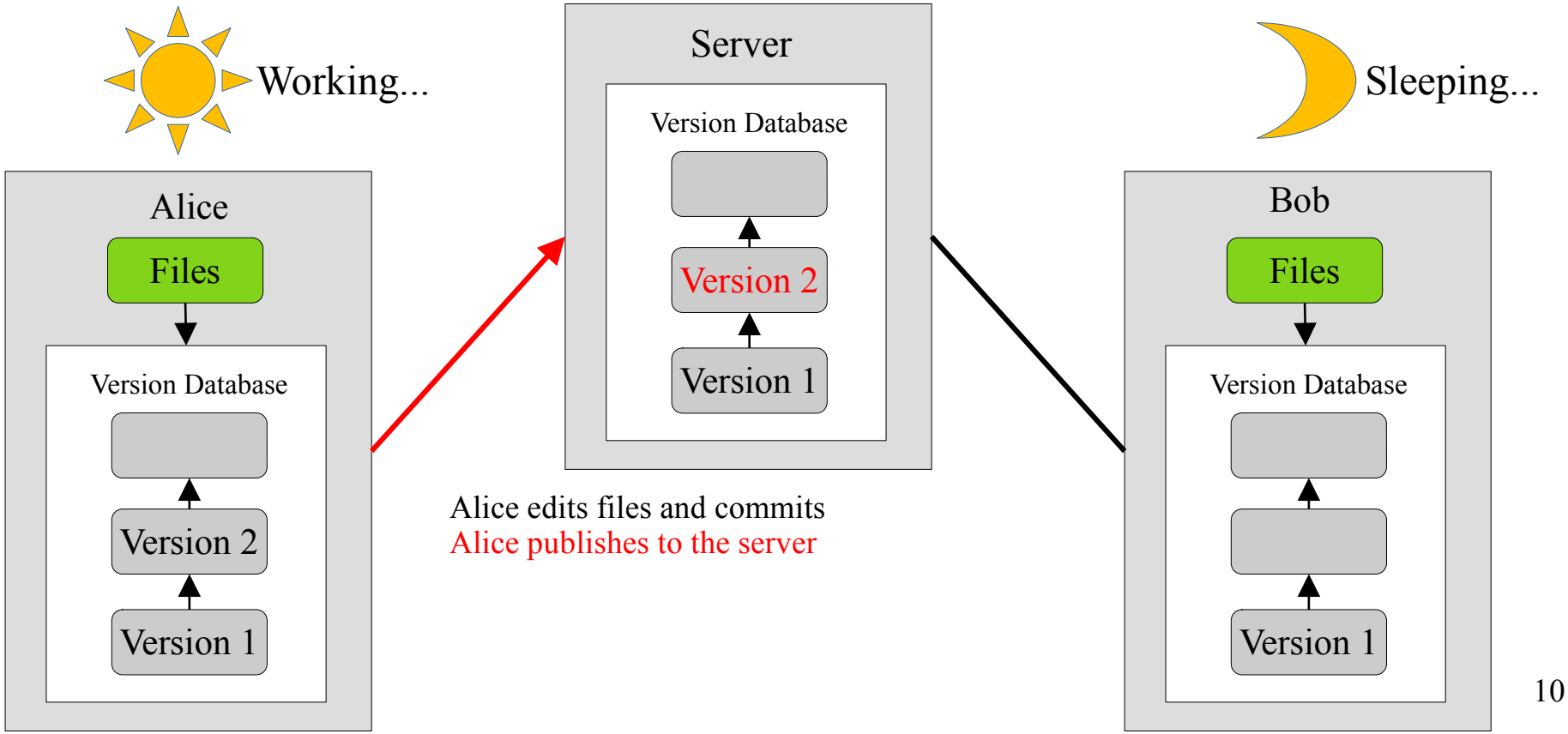
# Version Control System



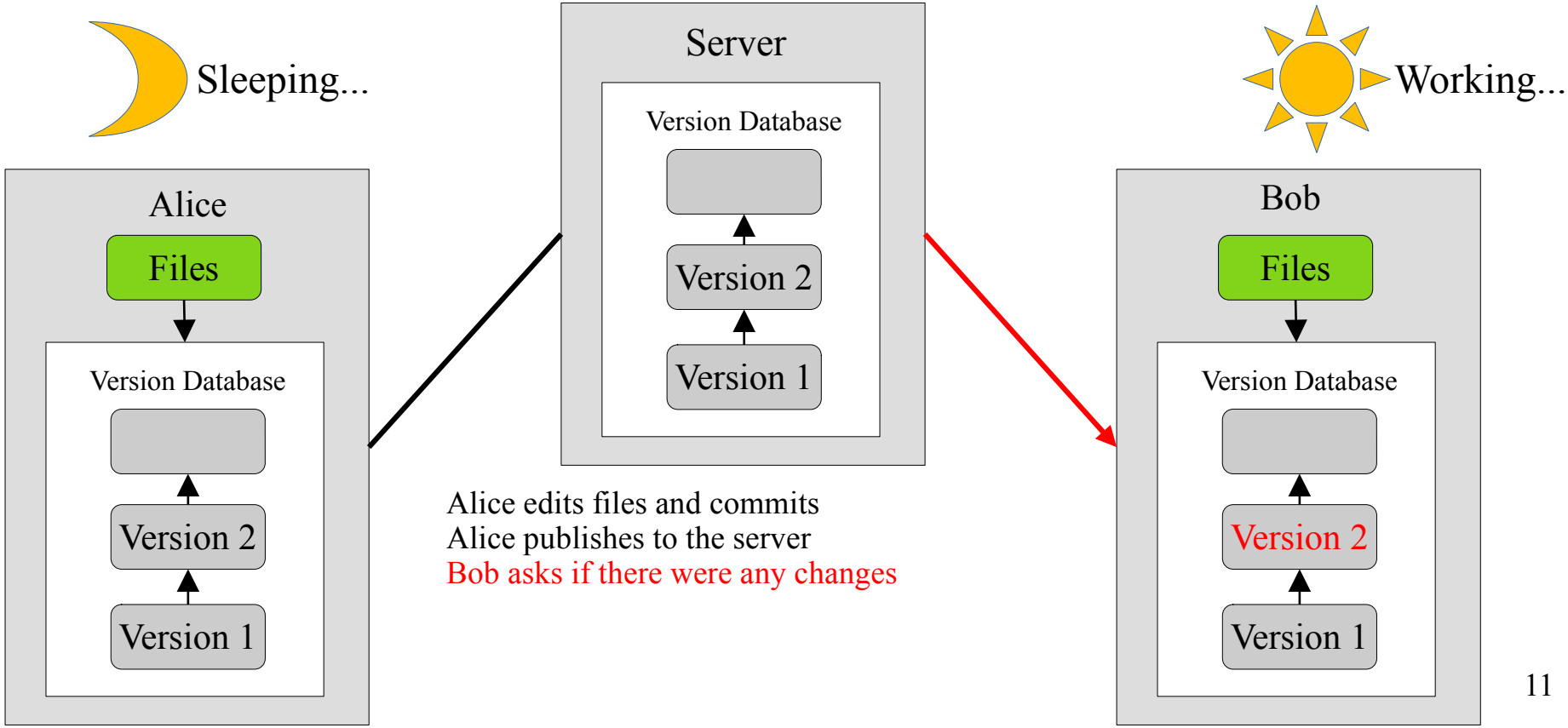
# Version Control System



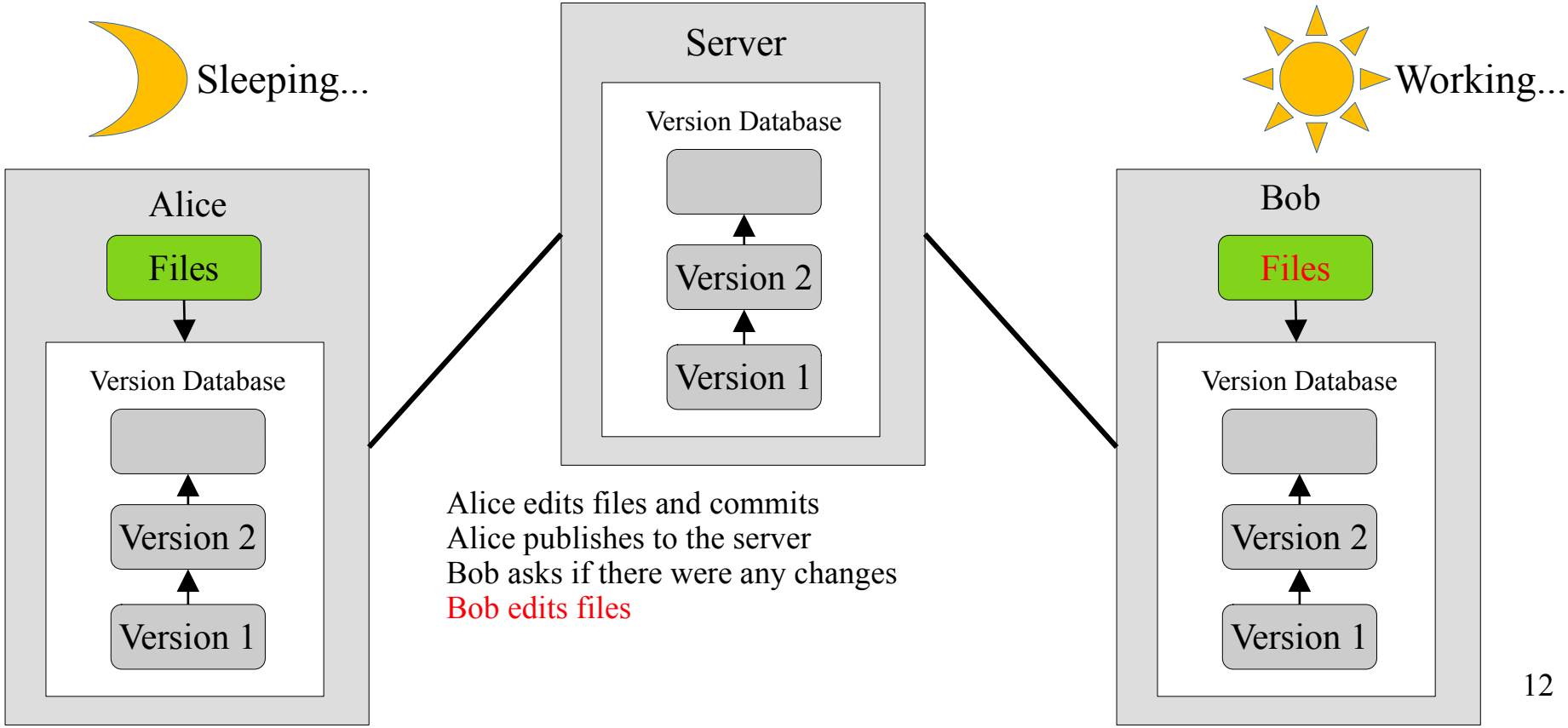
# Version Control System



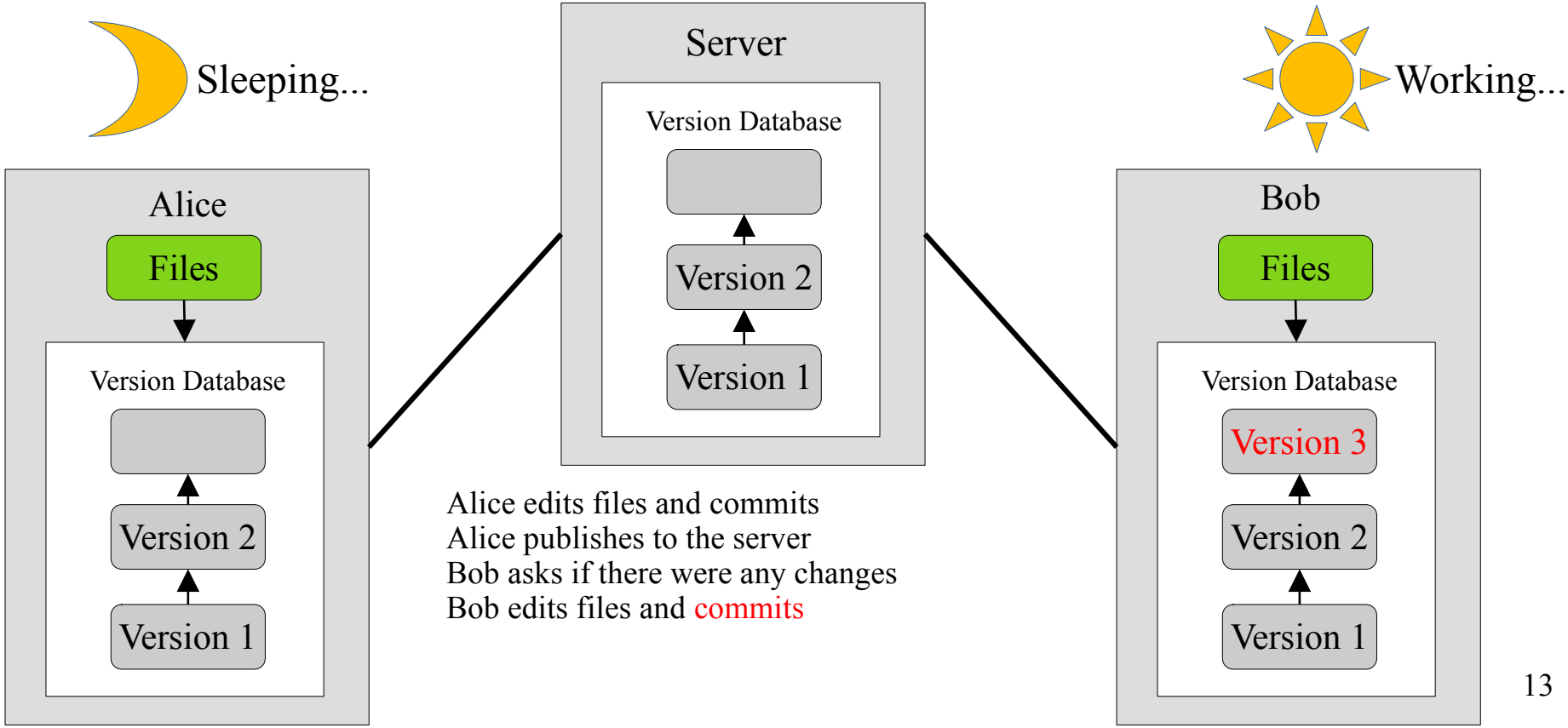
# Version Control System



# Version Control System

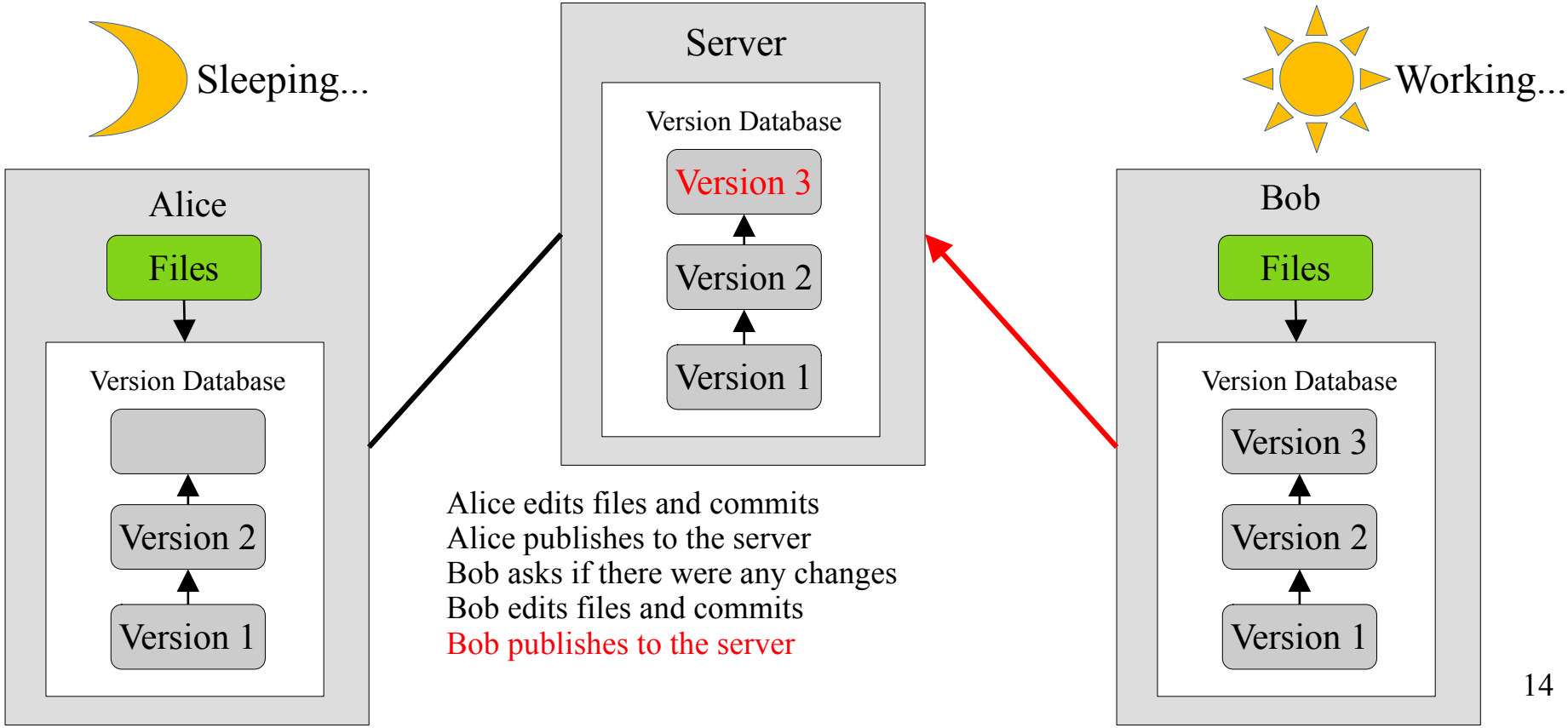


# Version Control System



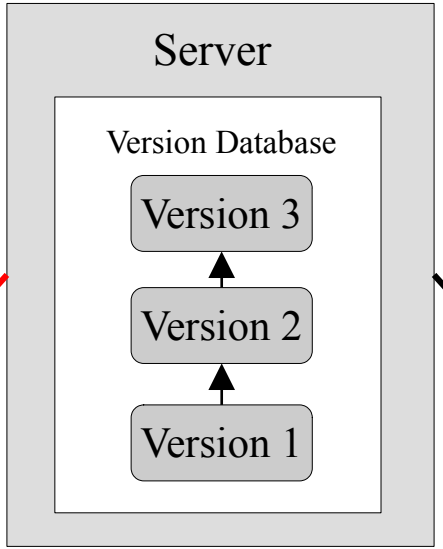

Alice edits files and commits
Alice publishes to the server
Bob asks if there were any changes
Bob edits files and commits

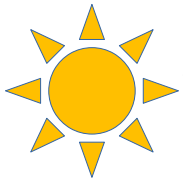
# Version Control System



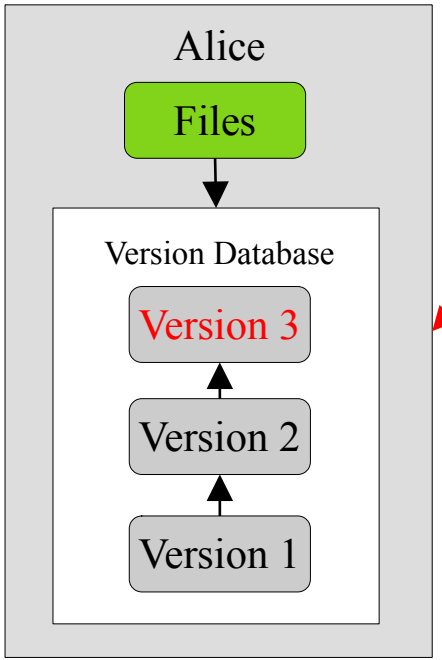
# Version Control System

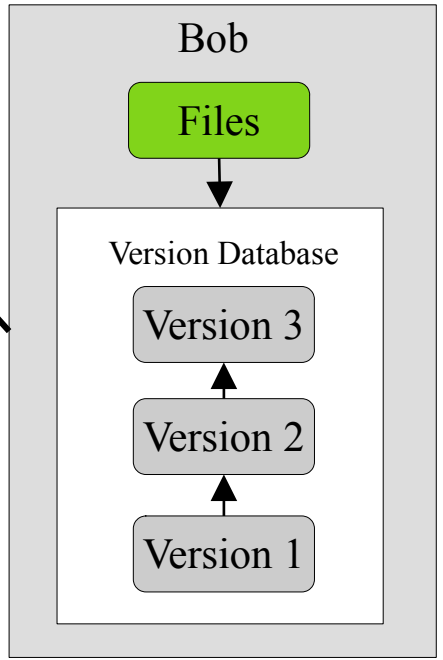
Sleeping...



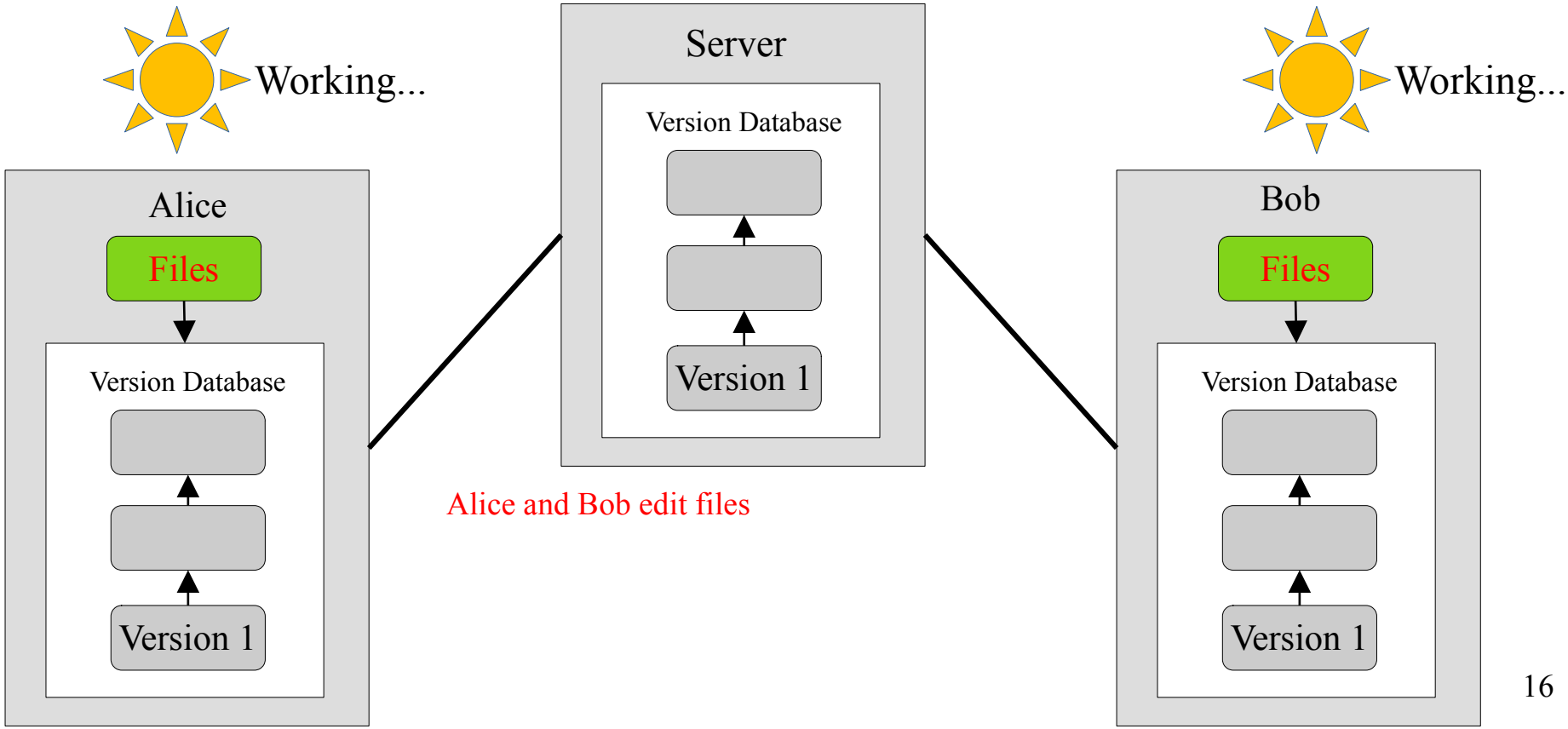
Working...



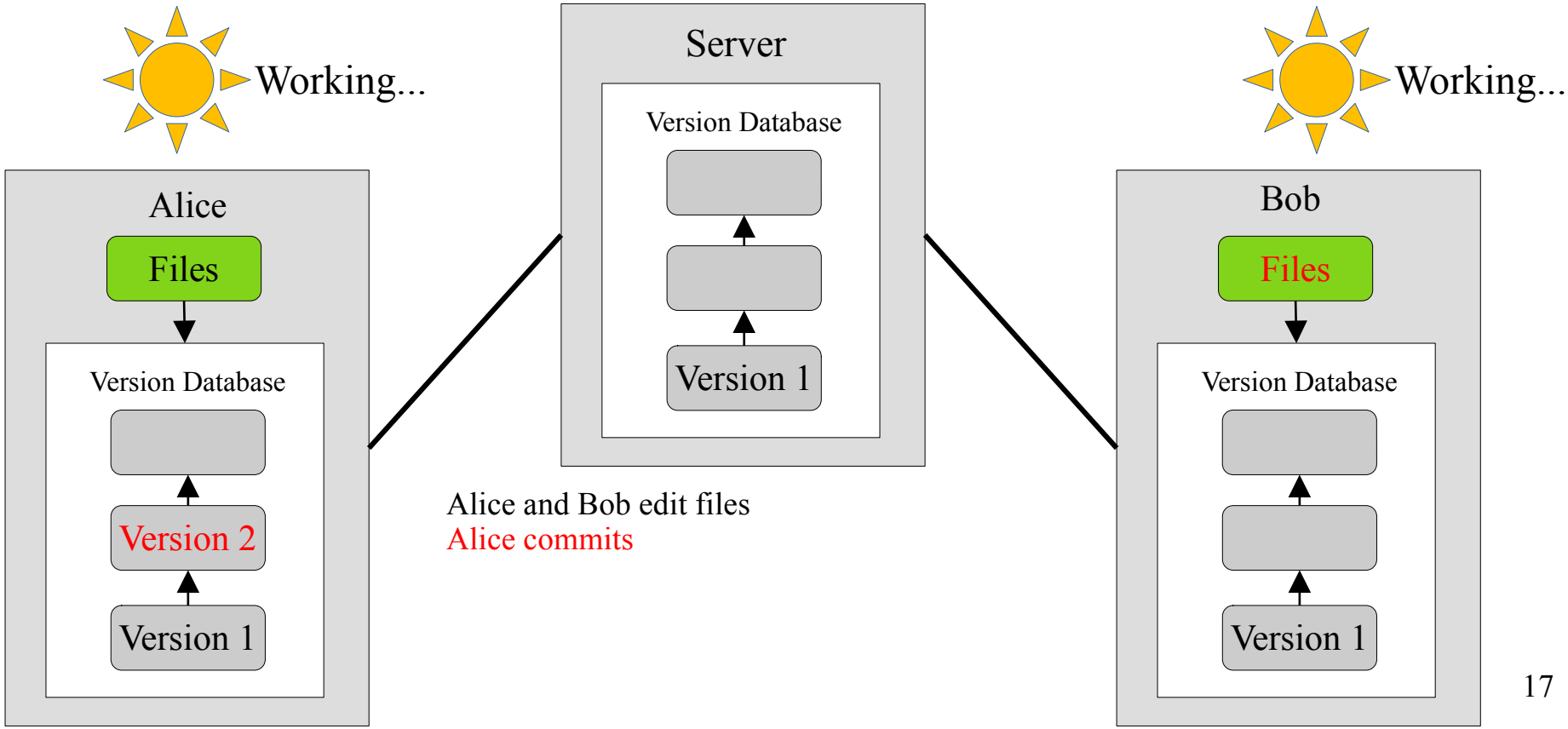




Alice edits files and commits
Alice publishes to the server
Bob asks if there were any changes
Bob edits files and commits
Bob publishes to the server
Alice asks if there were any changes
...

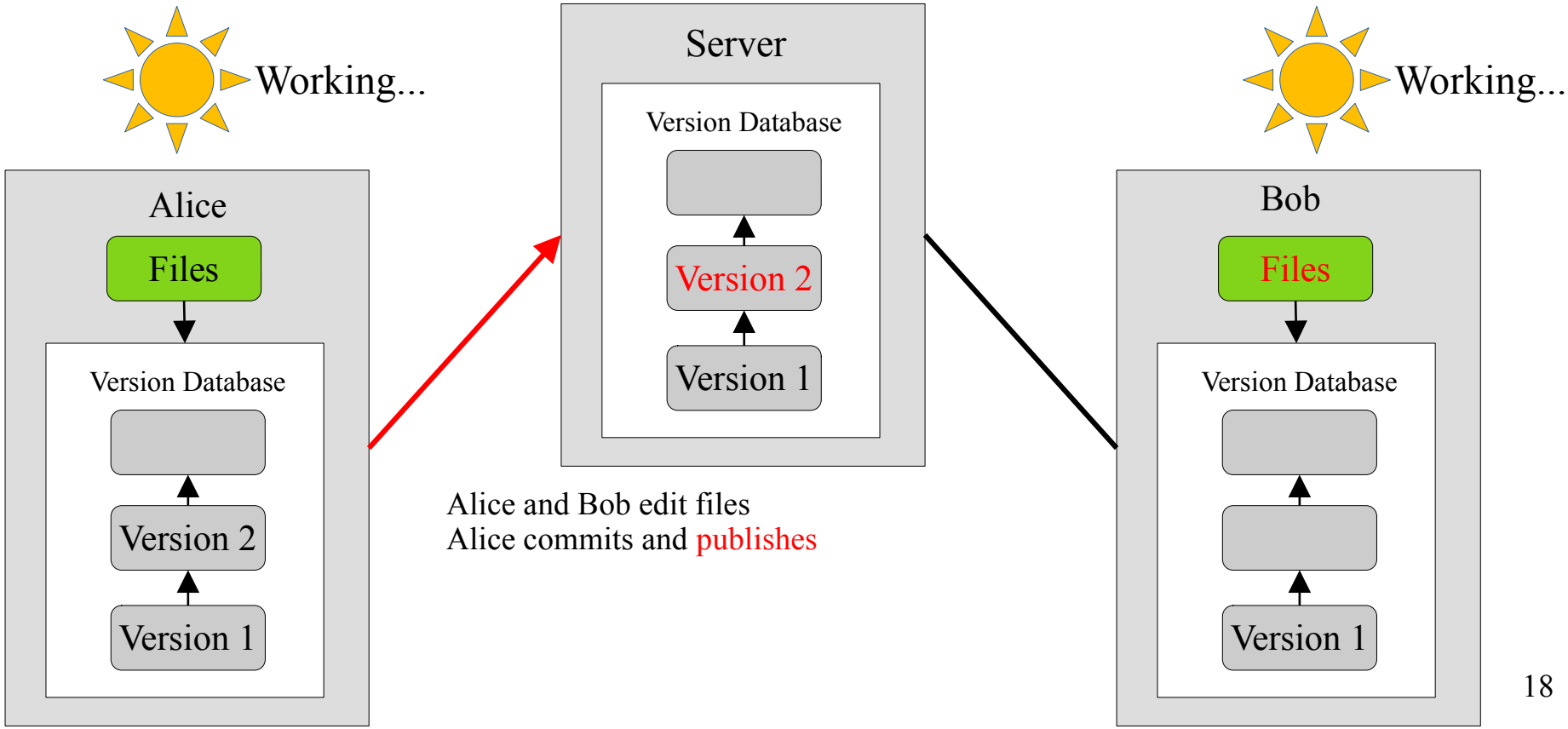
# Version Control System



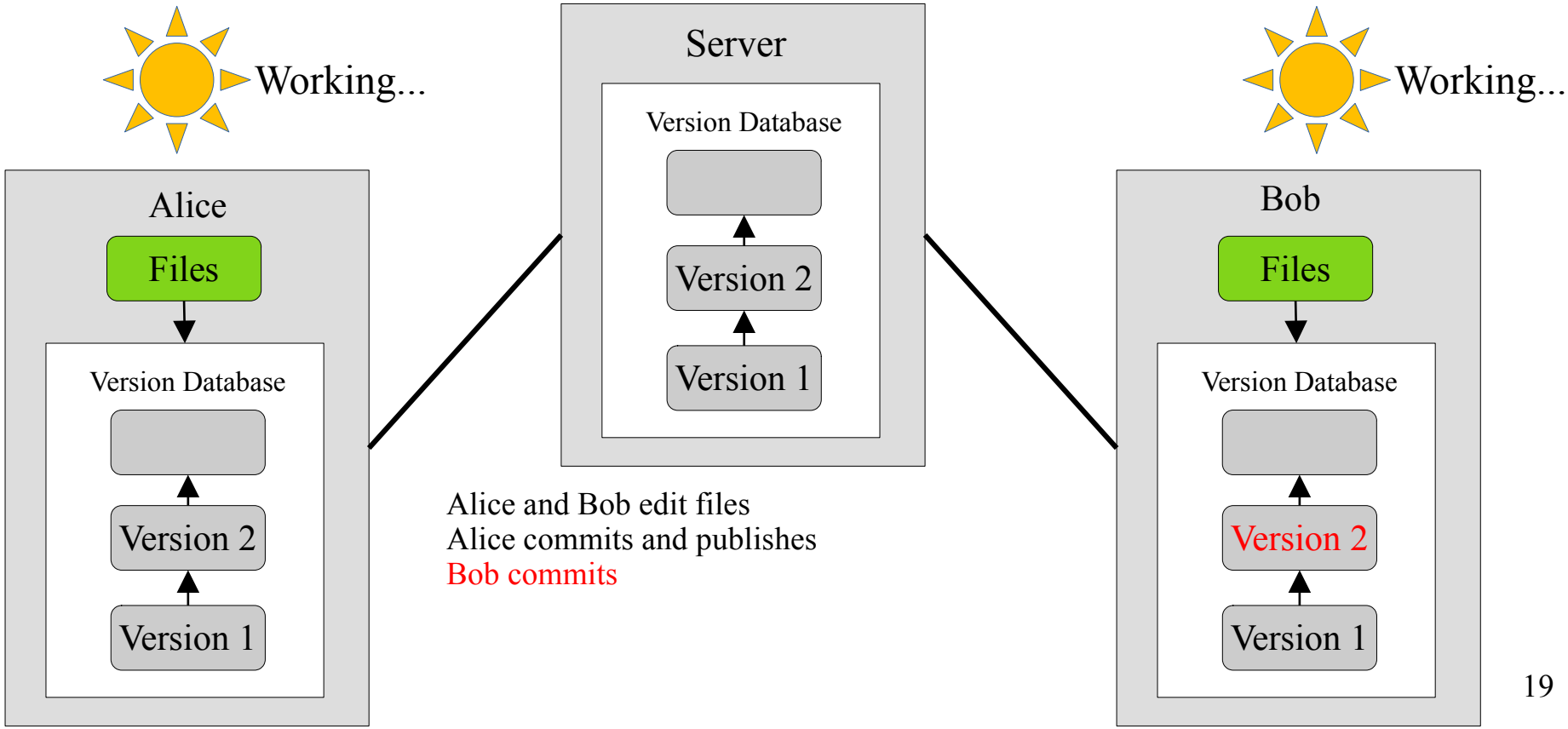
# Version Control System



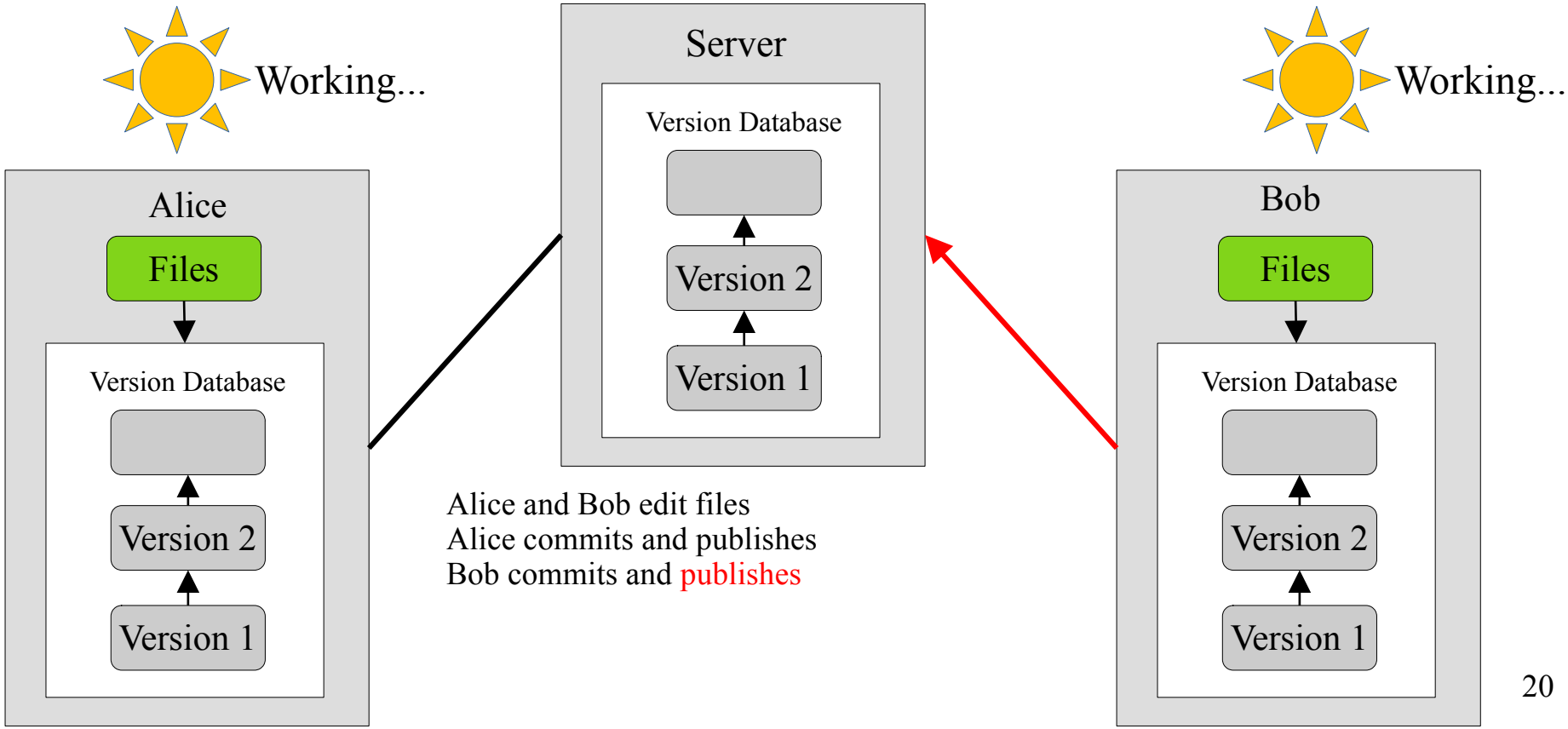
# Version Control System



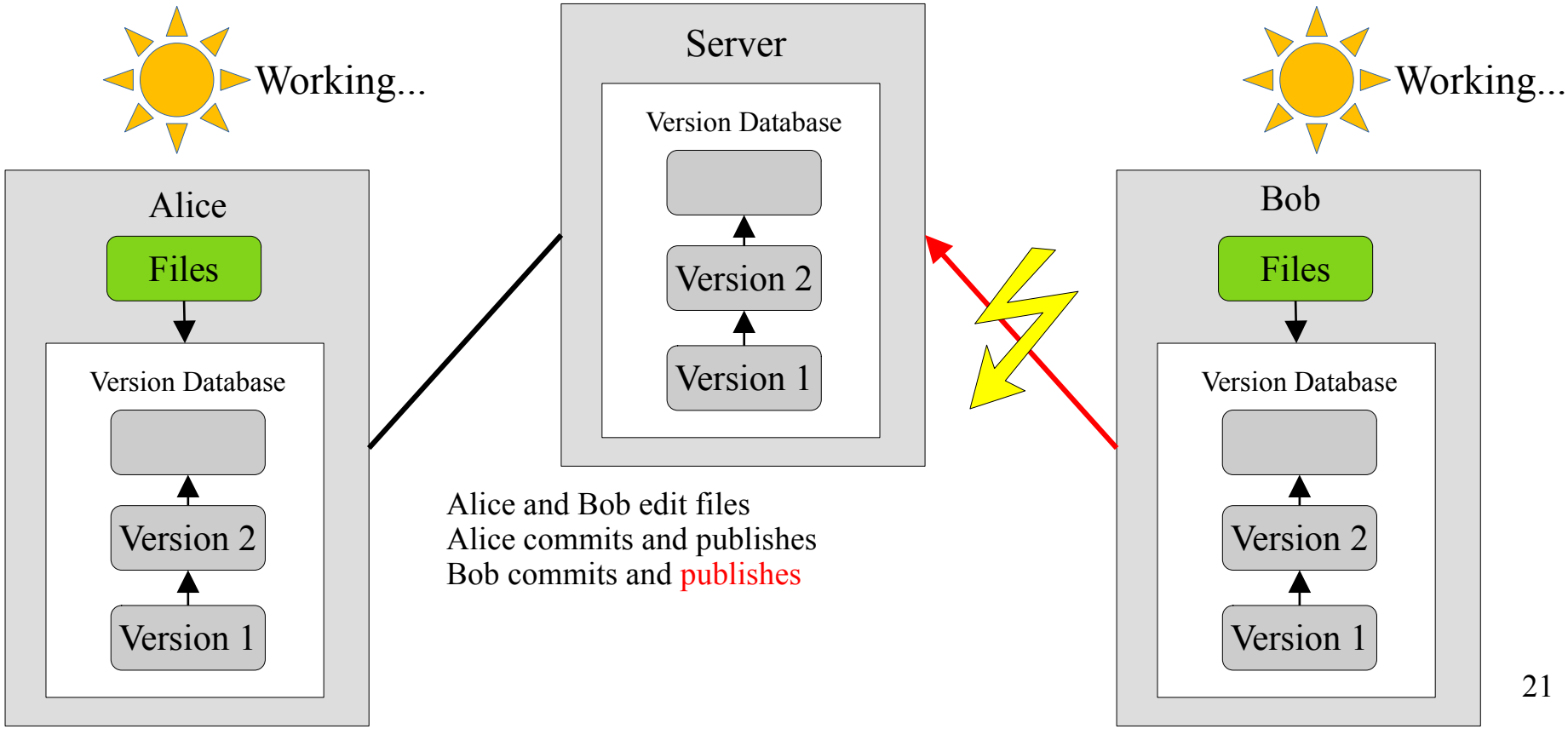
# Version Control System



# Version Control System

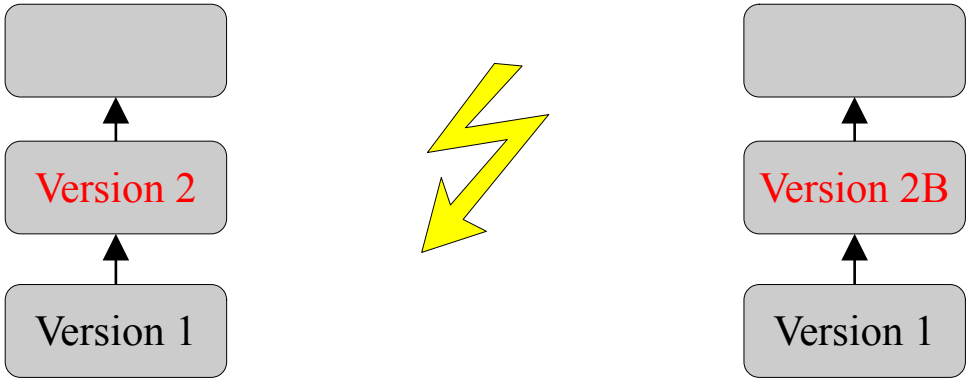


# Version Control System



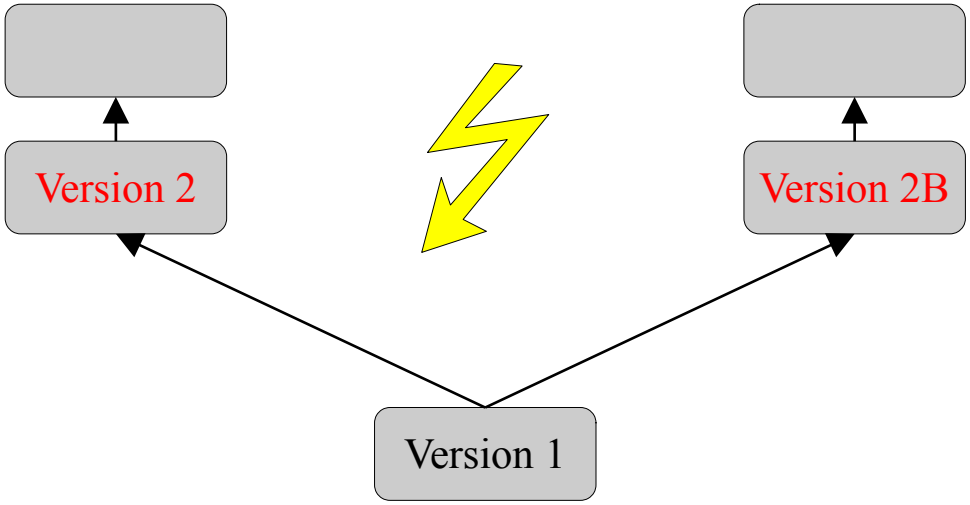
# Version Control System

Conflict!



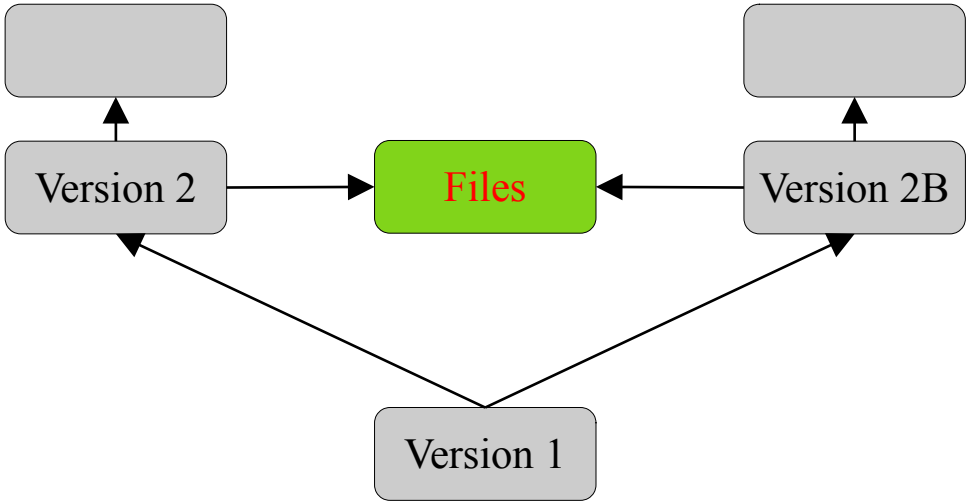
# Version Control System

The histories have diverged!



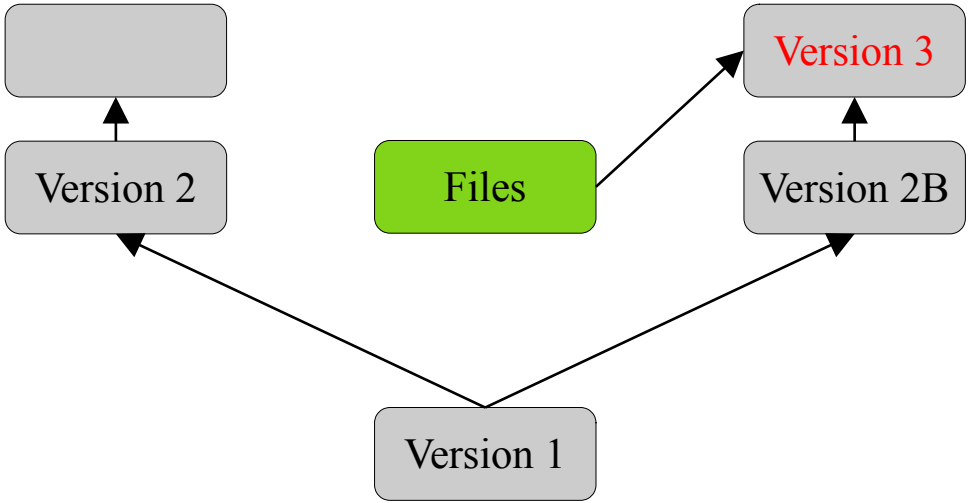
# Version Control System

Bob has to resolve the conflict: merging!

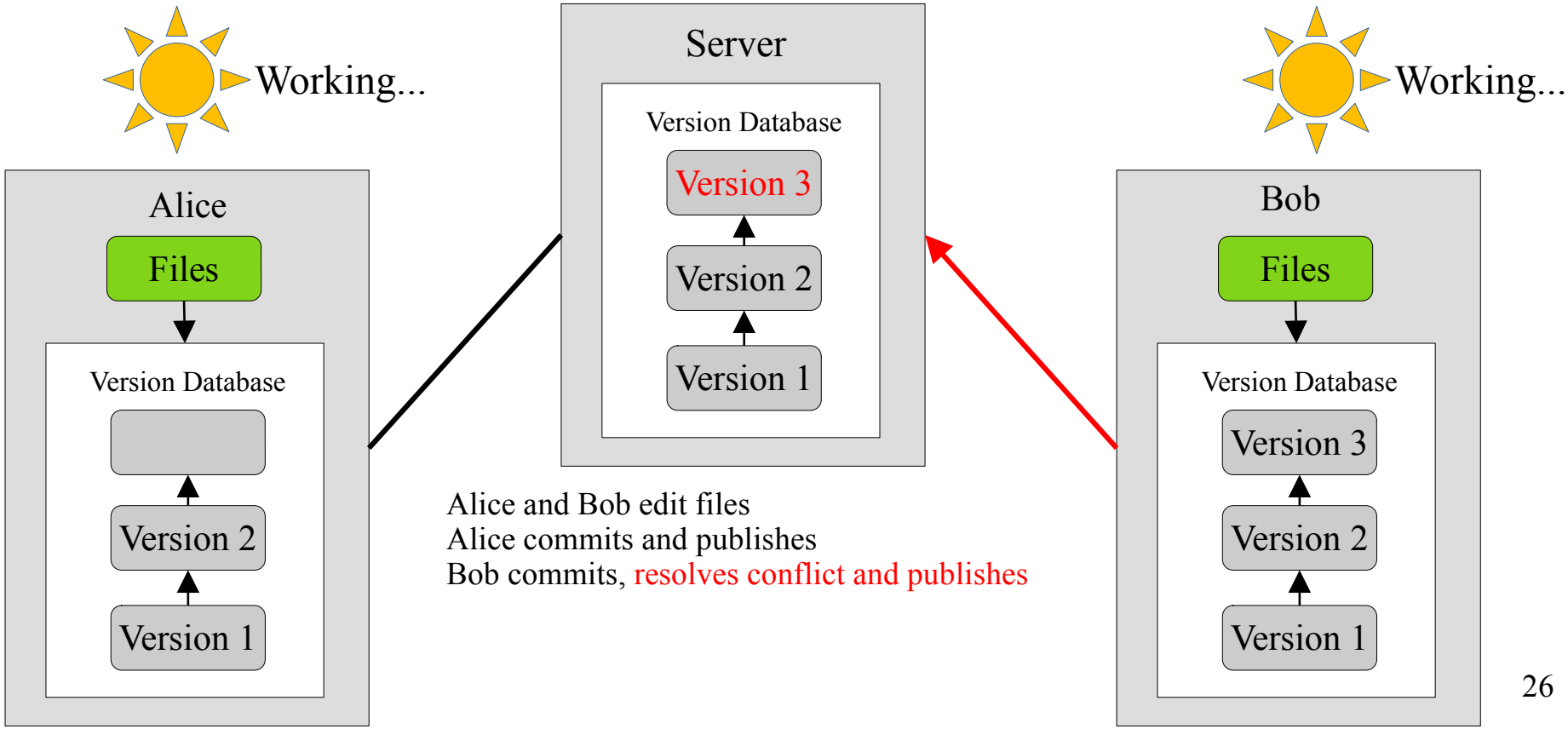


# Version Control System

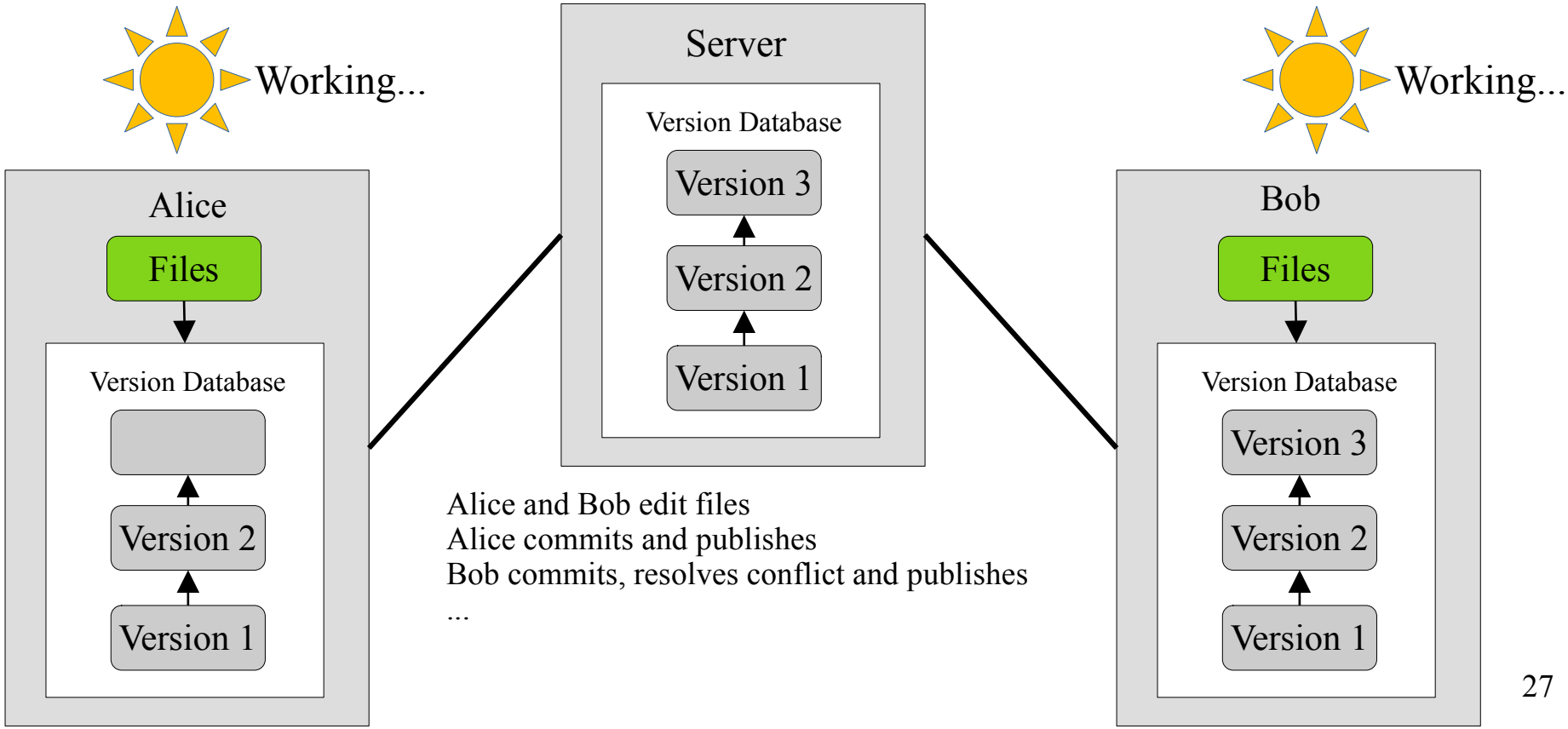
## Merge commit



# Version Control System



# Version Control System



# First-Time Git Setup

- Set your identity (system/**global**/local)

```
$ git config --global user.name "Your name"
$ git config --global user.email userid@ethz.ch
```

- Set you editor (otherwise default editor, e.g. vi)

```
$ git config --global core.editor YOUREDITOR
```

- Checking Your Settings

https://stackoverflow.com/questions/11828270/how-do-i-exit-the-vim-editor

https://xkcd.com/378/

```
$ git config --list
```

- Many more configuration options are available, check the documentation

# Creating/Getting a Git Repository

- Initializing a Repository in an (Existing) Directory
  - E.g. a new directory

```
$ mkdir project
$ cd project
$ git init
```

- Cloning an Existing Repository
  - E.g. cloning the lecture repository

```
$ git clone git@gitlab.ethz.ch:pt1_hs22/lecture.git
$ git clone https://gitlab.ethz.ch/pt1_hs22/lecture.git
```

# Getting started with git

- Check the status

```
$ git status
```

- Check the log (history)

```
$ git log
$ git log --all --graph --decorate
```

- Get help

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

# Add a file and commit it

- Write a file `mars.txt`
- Check the status
- Now let's add it to the "staging area"

```
$ git add mars.txt
```

- Check the status again
- Now let us commit it to the repository, with a sensible comment

```
$ git commit -m "Start plan"
```

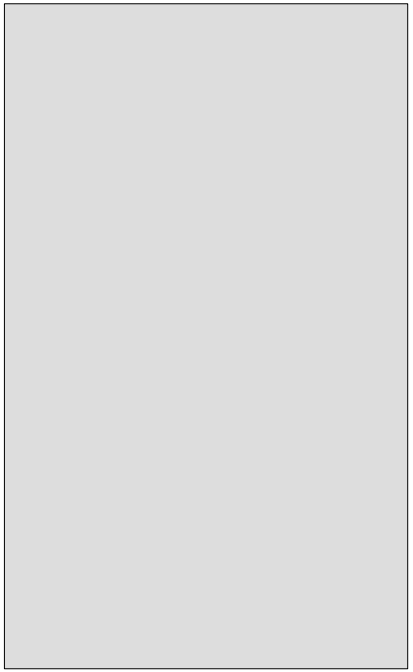
- Check the status and log

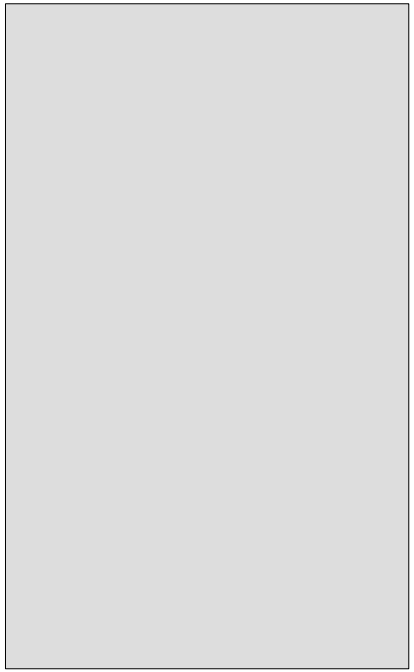
# Local files, staging area and repository

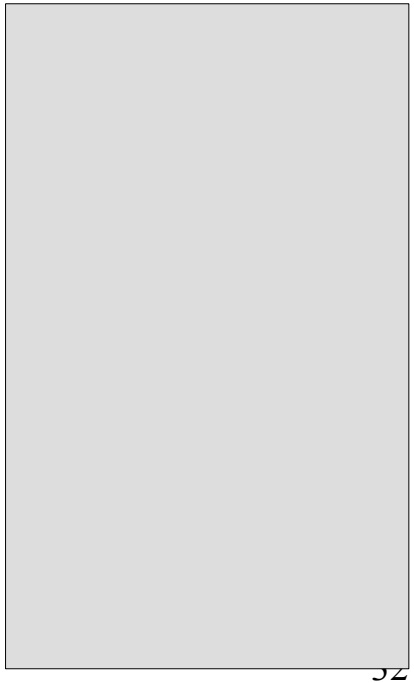
**File system**

**Staging area**

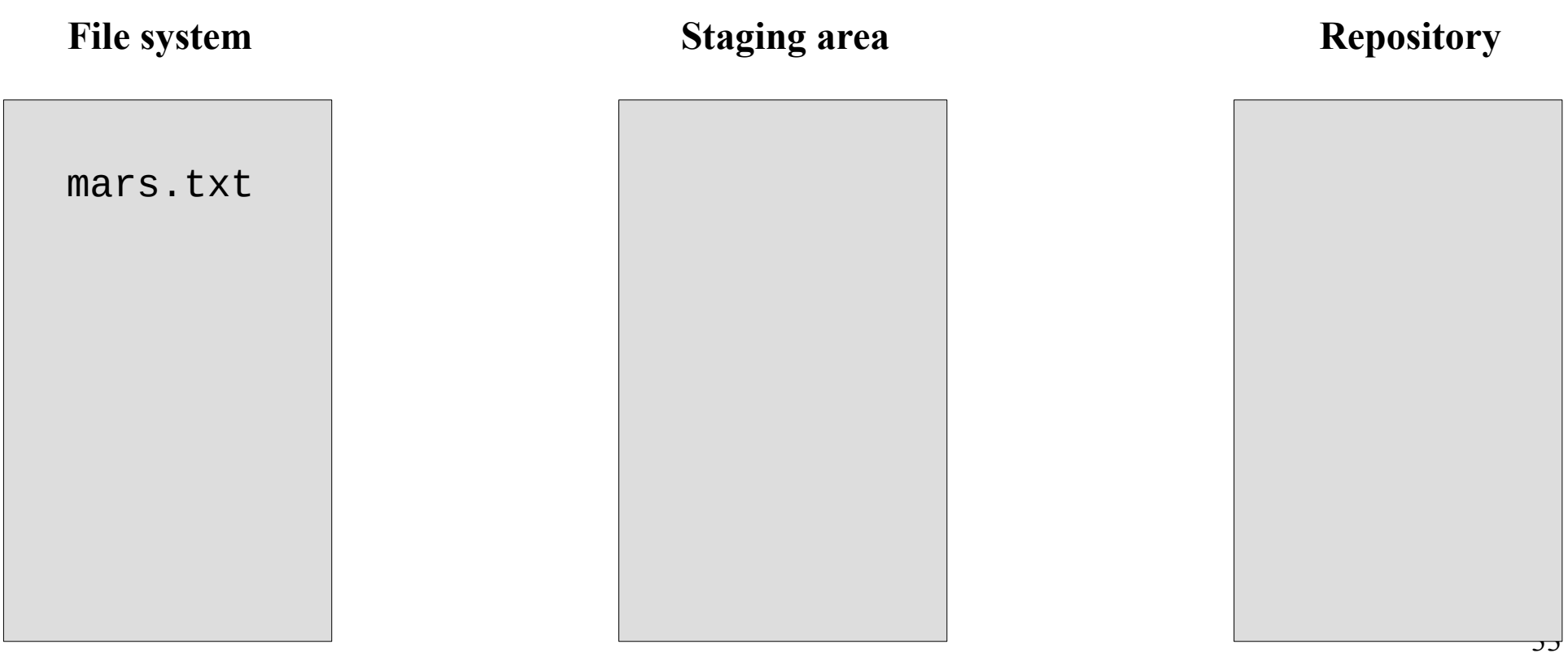
**Repository**



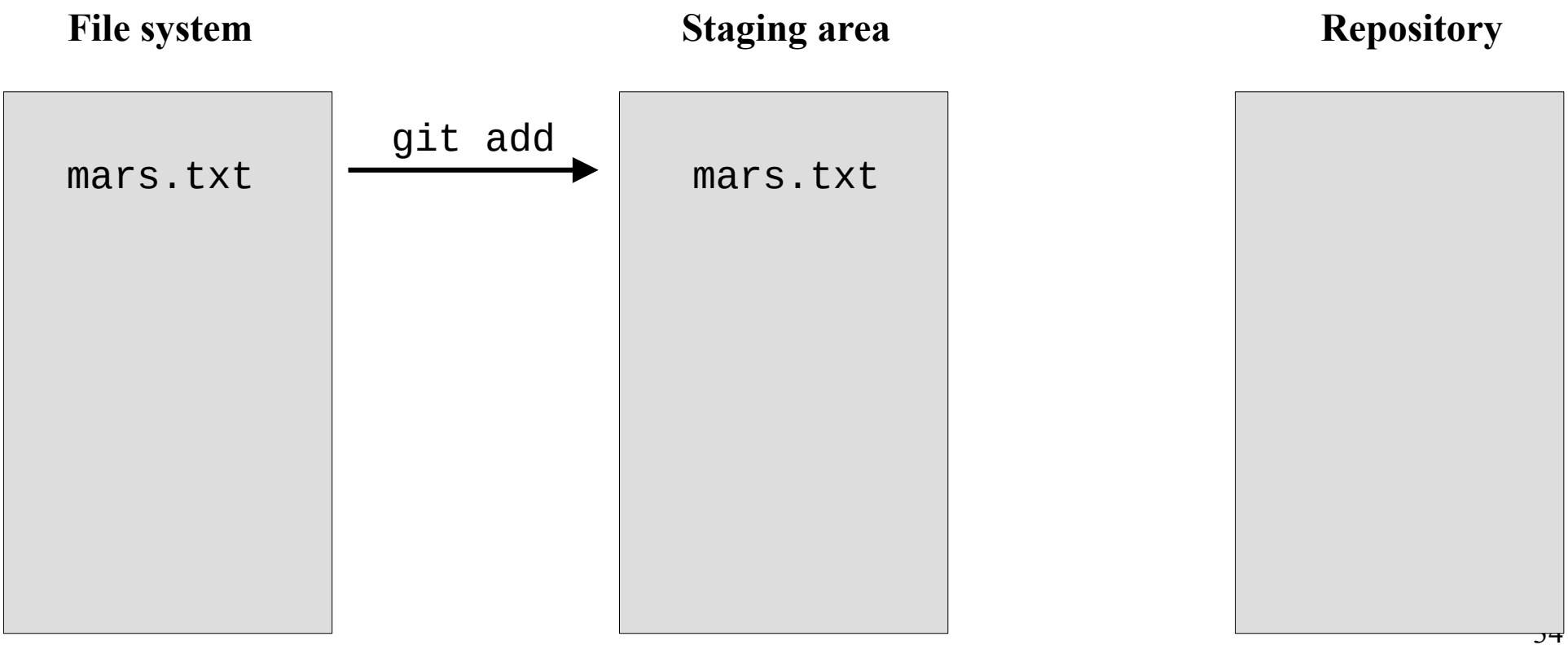




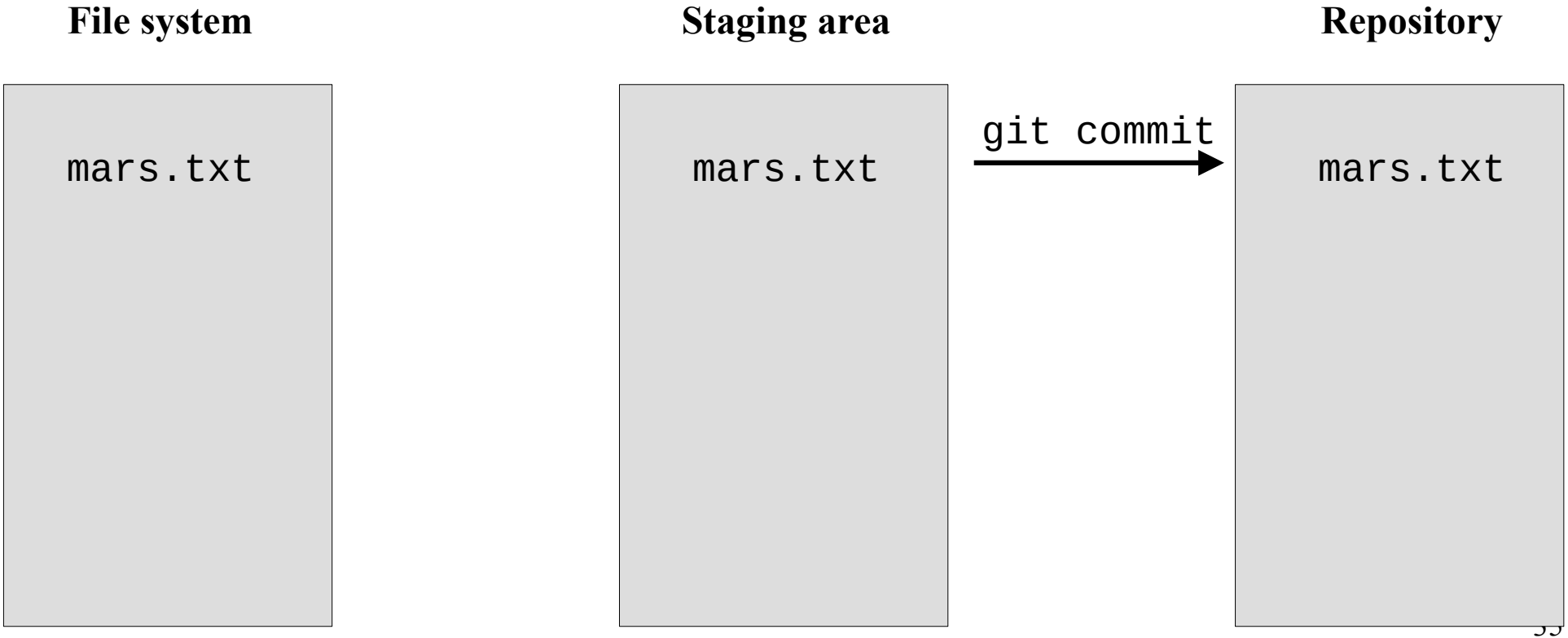
# Local files, staging area and repository



# Local files, staging area and repository



# Local files, staging area and repository



# Edit the file

- Edit the file
- Check the status and the difference

```
$ git status
$ git diff
```

- Add it to the staging area

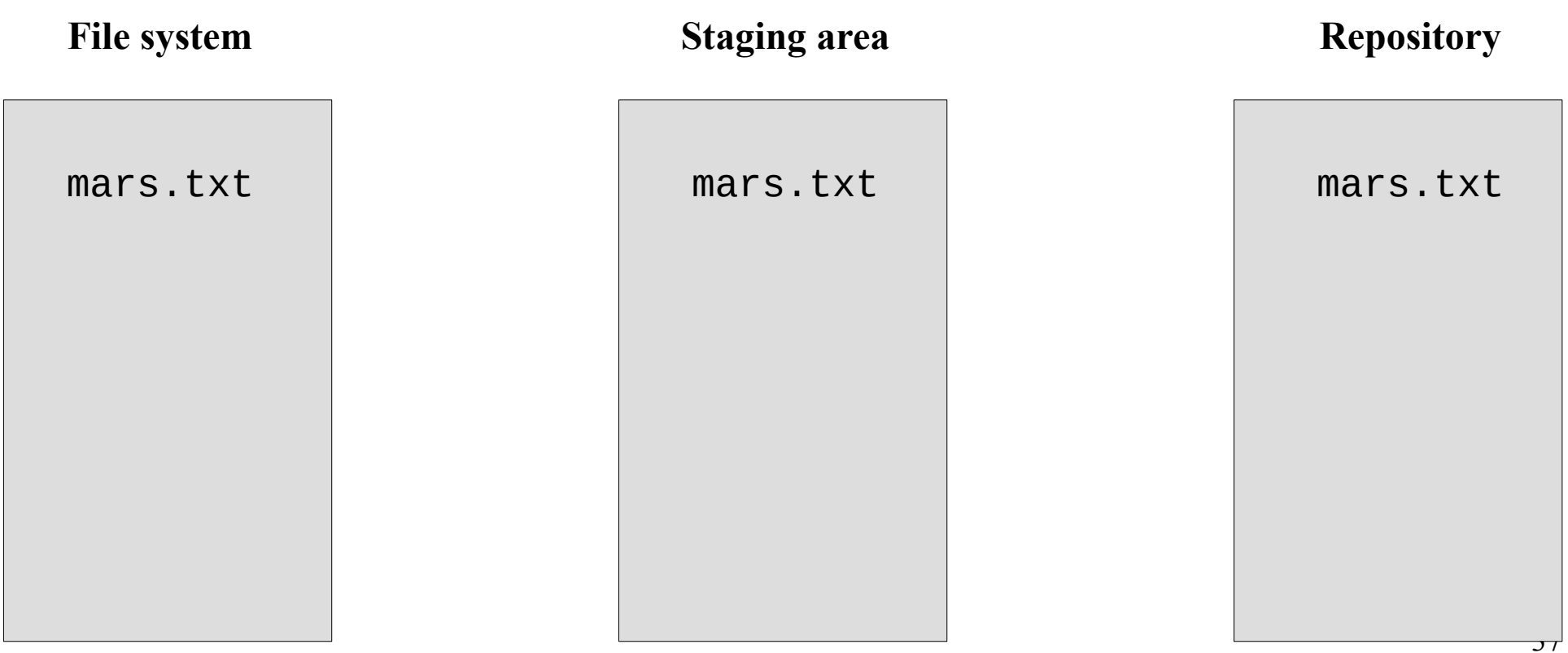
```
$ git add mars.txt
```

- Check the status and diff once more
- Commit the next version

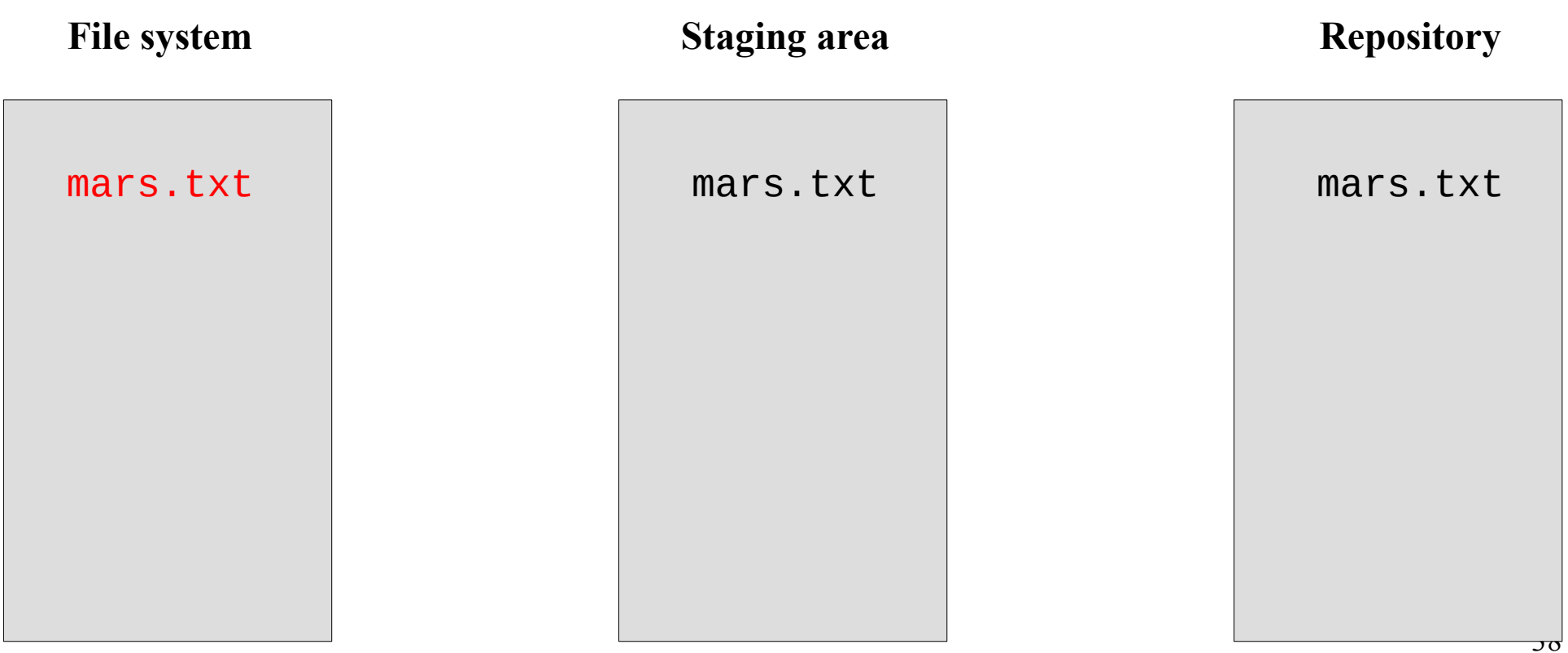
```
$ git commit -m "Elaborate plan”
```

- Check the status and log

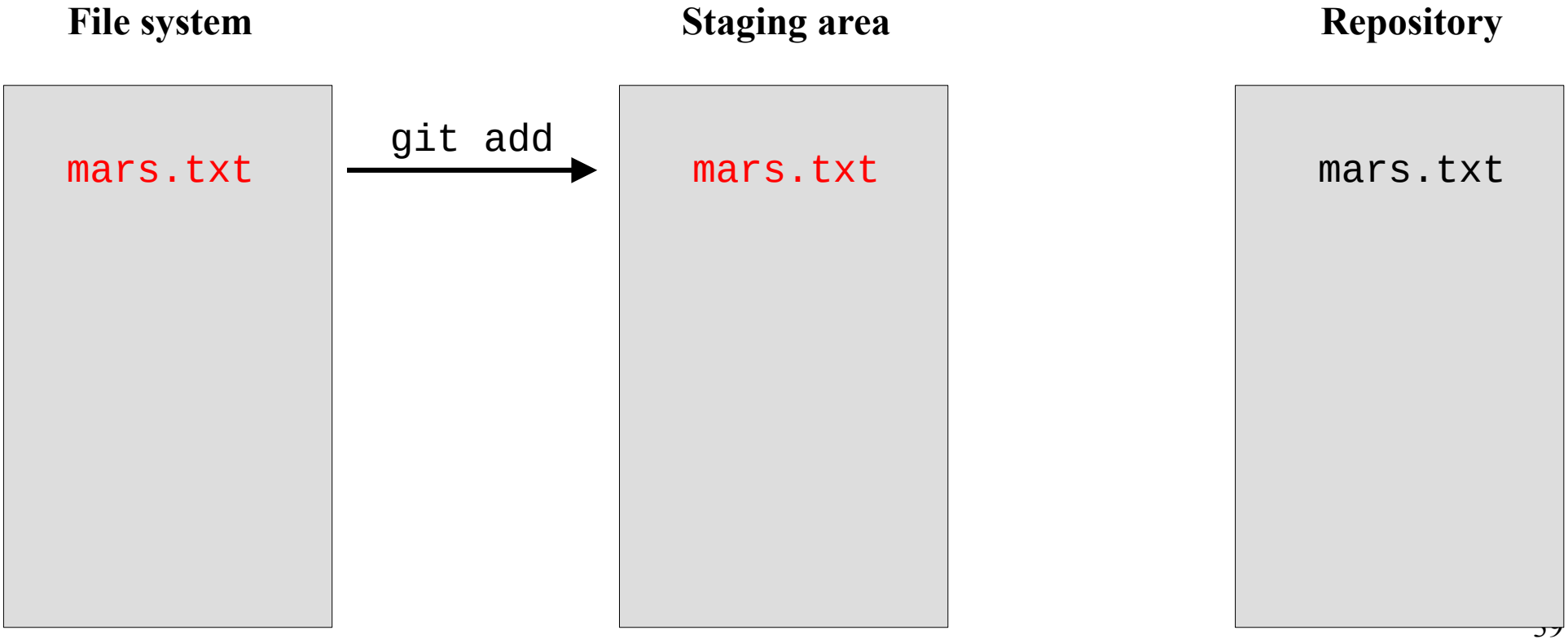
# Local files, staging area and repository



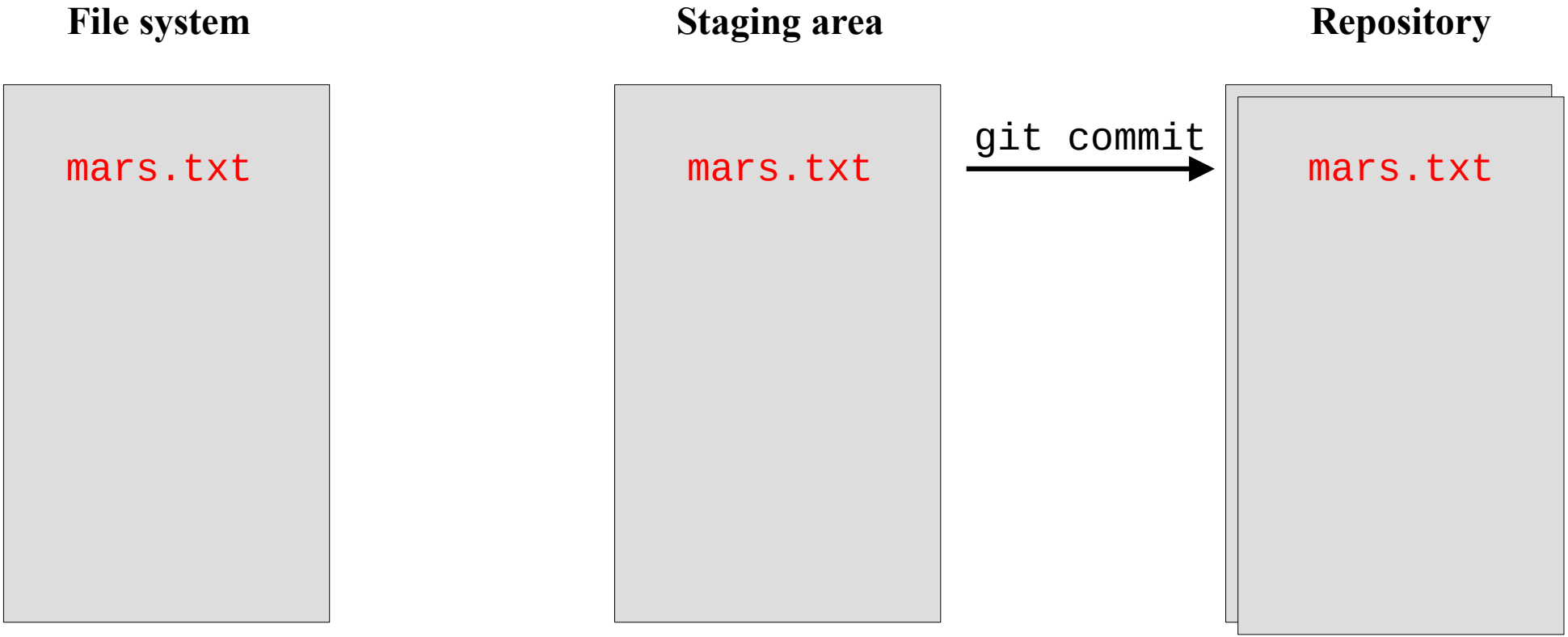
# Local files, staging area and repository



# Local files, staging area and repository



# Local files, staging area and repository



# Collaboration: a common repository

- We want/need to publish our changes somewhere
- Your own server infrastructure (needs work to set up)
- Popular repository hosts
  - https://github.com/
  - https://bitbucket.org
  - ...
- Git-repository managers such as GitLab (https://gitlab.com/)
  - https://gitlab.ethz.ch/
    - Will be used for the lecture material & exercises
    - Lets you control who has read and write access
    - Lets you control who has read and write access

# Setting up a GitLab repository

- Log on to gitlab.ethz.ch and set up your account
- Create an SSH key pair (see here)
- Upload the public key to GitLab
- Create a new project and follow the instructions

# Pushing changes to the repository on the server

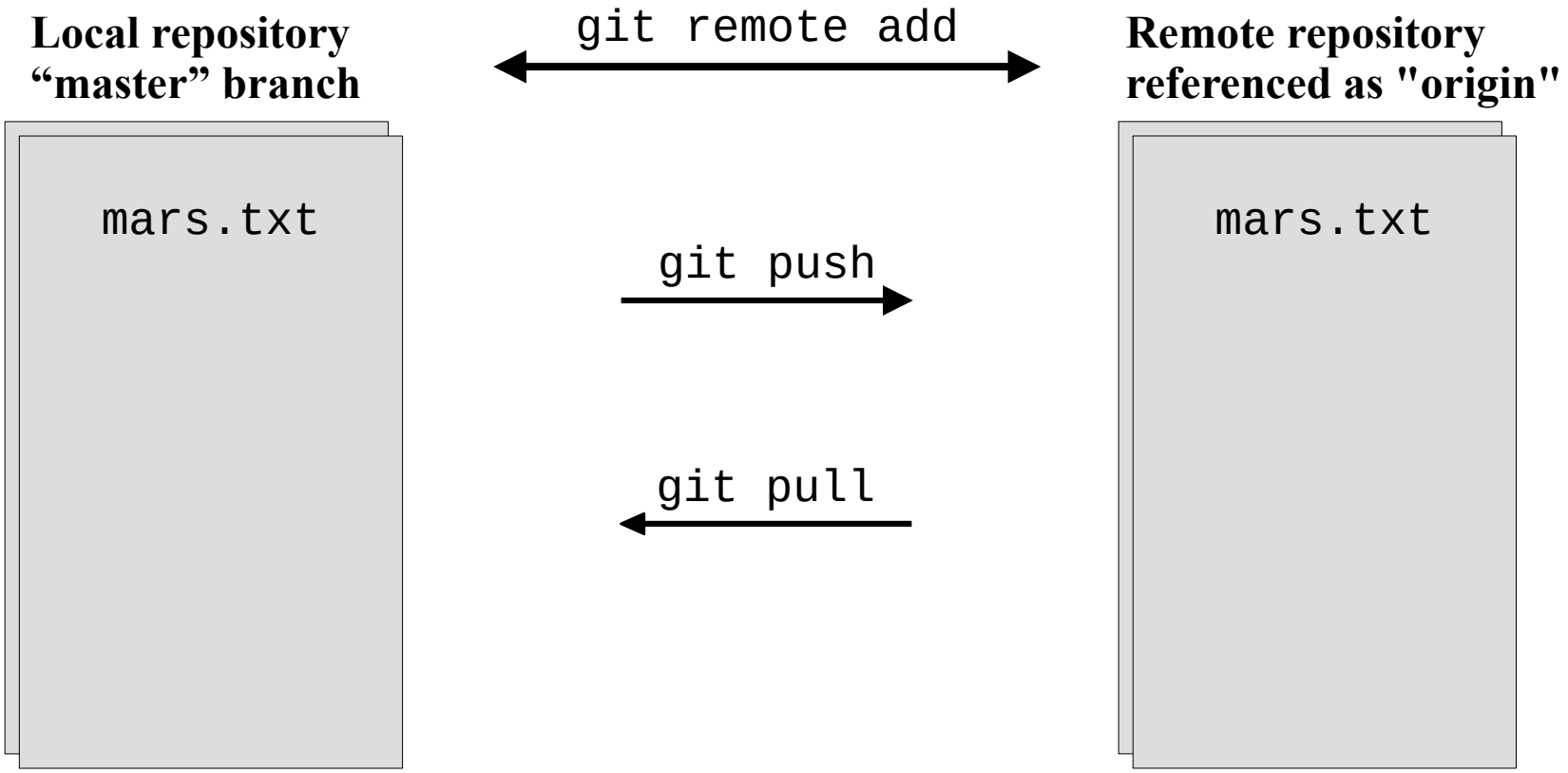
- Follow the instructions given by GitLab
- Connect the local and remote repository

```
$ git remote add origin git@gitlab.ethz.ch:...
```

- And push the changes from the local to the remote repository

```
$ git push –u origin master
```

# Connecting the repositories



# Pulling changes and resolving conflicts

- Someone edits the file and pushes their changes
- We are prevented from pushing our changes before we merged theirs, thus let's pull the changes

```
$ git pull
```

- If there are conflicts they are marked and to be resolved before we can commit and push the files again

```
$ git add mars.txt
$ git commit -m "resolved conflicts"
$ git push
```

# Branching

- Sometimes you want to make some changes but don't mess up the working version
  - Create a new branch

```
$ git branch moons
```

- Now look at the branches

```
$ git branch
```

- Switch to the new branch

```
$ git checkout moons
```

# Merging branches

- Create and add a new file `moons.txt`, then switch back to master

```
$ git add moons.txt
$ git commit –m "Discussing moons"
$ git checkout master
```

- `moons.txt` is gone, since it's only on the moons branch

- We can merge the changes done on moons back to master

```
$ git merge moons
```

- Merging between branches can create conflicts that need to be resolved

# More on working with branches

- Pushing a new branch to the remote repository referred to by “origin”

```
$ git push origin moons
```

- Deleting a local branch

```
$ git branch -d moons
```

- Deleting a remote branch from the remote repository referred to by "origin"

```
$ git push origin :moons
```

# Ignoring Files

- Exclude certain files from being tracked
  - `.gitignore`

- Checkout the collection of templates
  - https://github.com/github/gitignore

# Advice

- Always use version control, for any project (code, thesis, ...)
- Commit often
- Use good commit messages
- Use branches to add new features to a working version
- Commit often
- Submit your solutions by using git, so that the assistants can easily see the code and comment in the files
- Give us "maintainer" access to your repositories

# Advice

- Alw … s, …)
- Com …
- Use …
- Use …
- Com …
- Subm … can easil …
- Give us "maintainer" access to your repositories


https://xkcd.com/1296/

# References

- Documentation and more:
  - Git's man pages

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

- https://git-scm.com/docs (Reference manual)
- https://git-scm.com/book/en/v2 (Pro git book)
- https://swcarpentry.github.io/git-novice/ (Tutorial from Software Carpentry)