

# *VERSION CONTROL WITH GIT*

*Wanqian Yang*  
*Slides by Maria Zlatkova*  
*CS50 for MBAs*

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

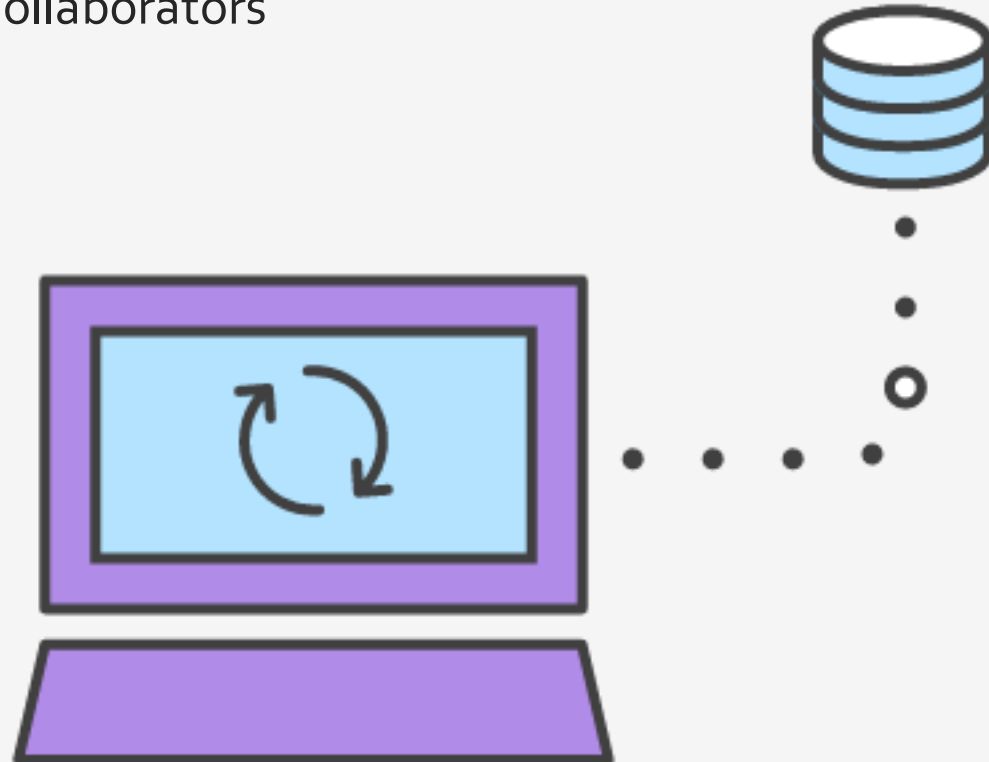
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.



# *Why Git?*

- Collaboration
- Distributed version control model
- Local repository
- Remote repository
- Collaborators



# *Most common use cases*

---

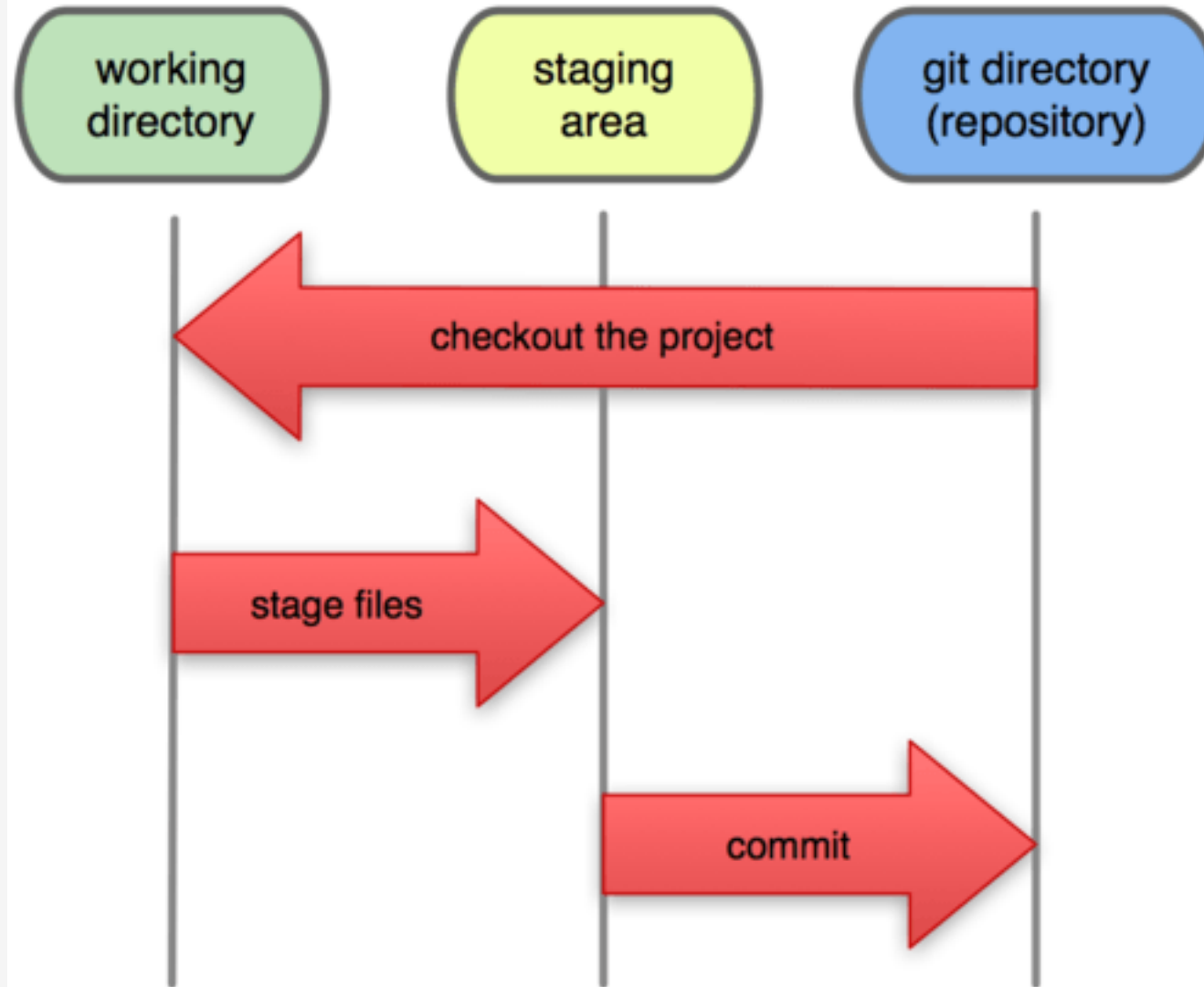
- adding/modifying a new file
- creating and merging a branch with and without merge conflicts
- Viewing the history/changelog
- Performing a rollback to a certain commit
- Sharing/synching your code to a remote/central repository



# *Terminology*


- **repository**, or “**repo**” for short, a digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves.
  - **master** - the repository’s main branch. Depending on the work flow it is the one people work on or the one where the integration happens
  - **remote** - these are “remote” locations of your repository, normally on some central server.
  - **clone** - copies an existing git repository, normally from some remote location to your local environment.
  - **commit** - submitting modified files to the repository (the local one)
  - **fetch or pull** - like updating or getting latest version. The difference between fetch and pull is that pull combines both, fetching the latest code from a remote repo as well as performs the merging.
  - **push** - is used to submit the code to a remote repository
  - **head** - is a reference to the node to which our working space of the repository currently points.
  - **branch** - a particular label on a given node, a snapshot/copy in time of a particular repository, further modified with its own changes
  - **SHA** - every commit or node in the Git tree is identified by a unique SHA key. You can use them in various commands in order to manipulate a specific node.
-

# Local Operations

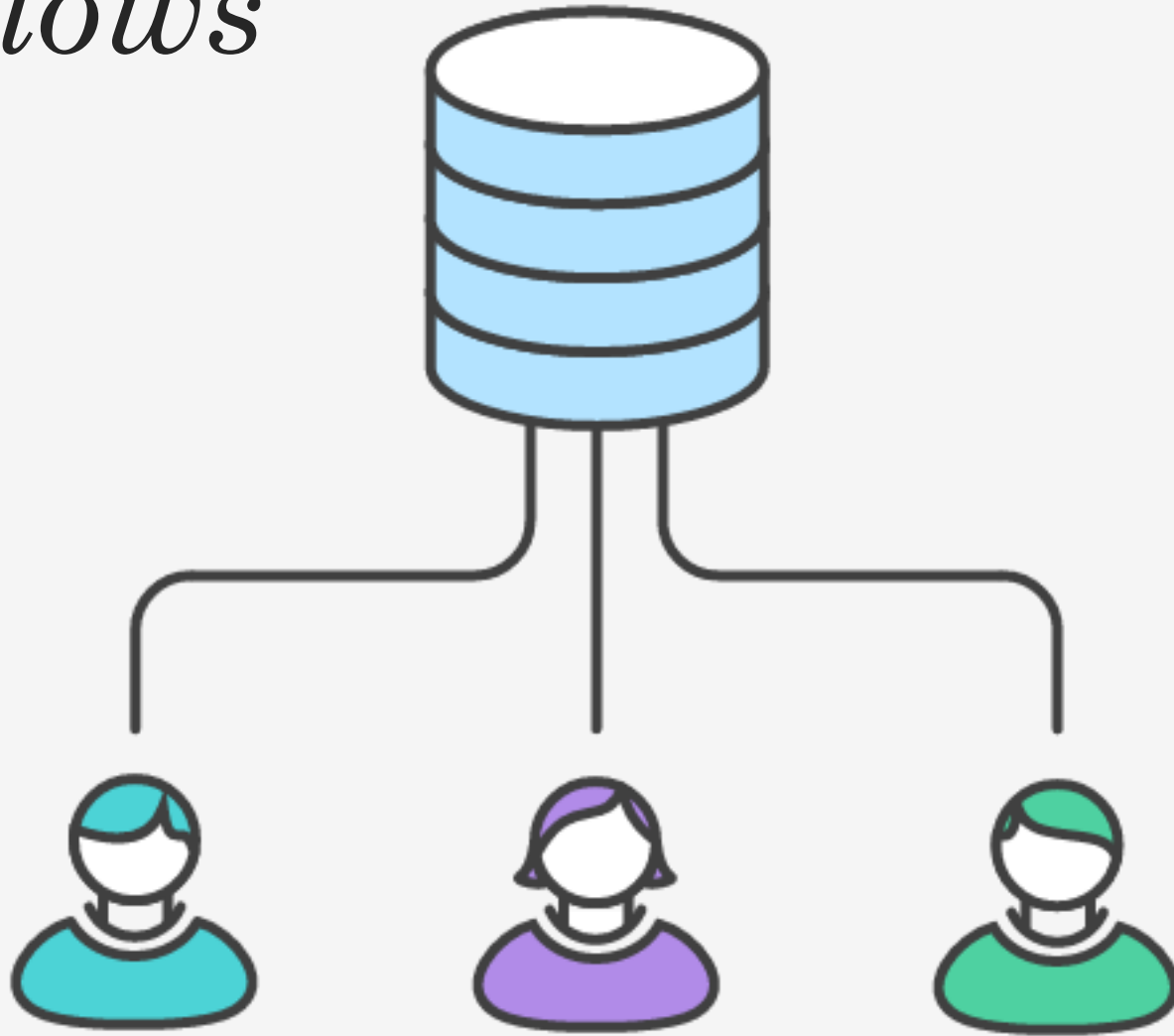


# *Basic Git Workflow*

---

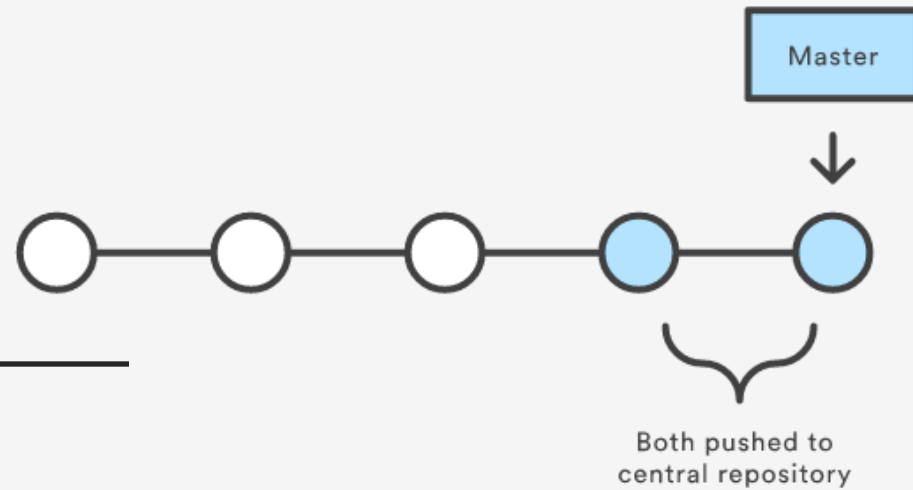
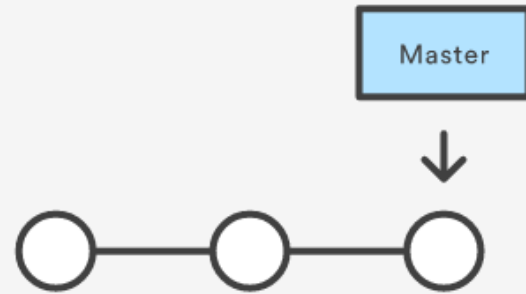
- You modify files in your working directory.
  - You stage the files, adding snapshots of them to your staging area.
  - You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.
- 

# *Workflows*



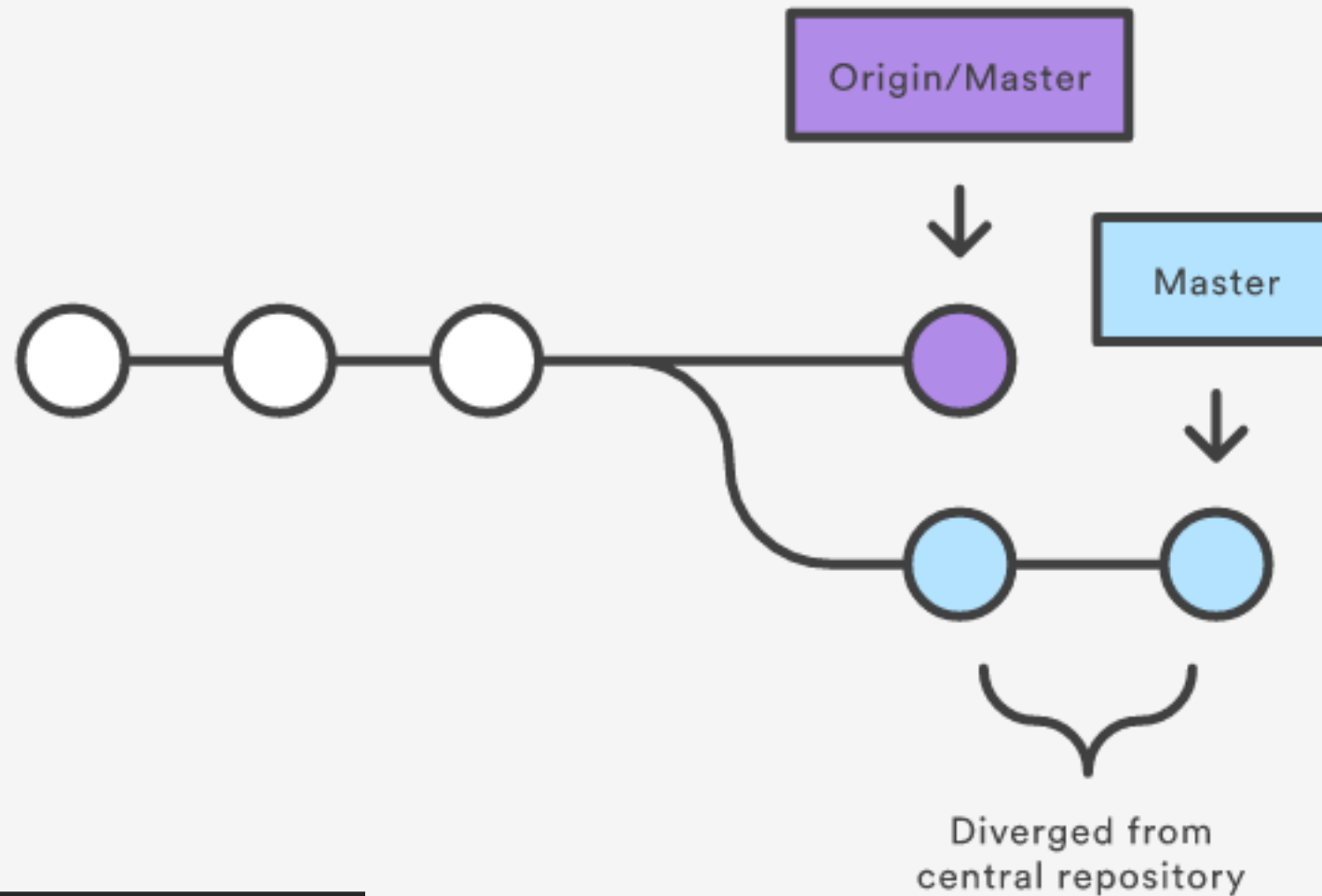


# *Centralized*



# *Centralized*

Local Repository



# *Example*

```
git config --global user.name "Maria Zlatkova"
```

```
git config --global user.email "zlatkova@college.harvard.edu"
```



# *Example*

- Maria and David work on a project together
- One of them initializes the central repository

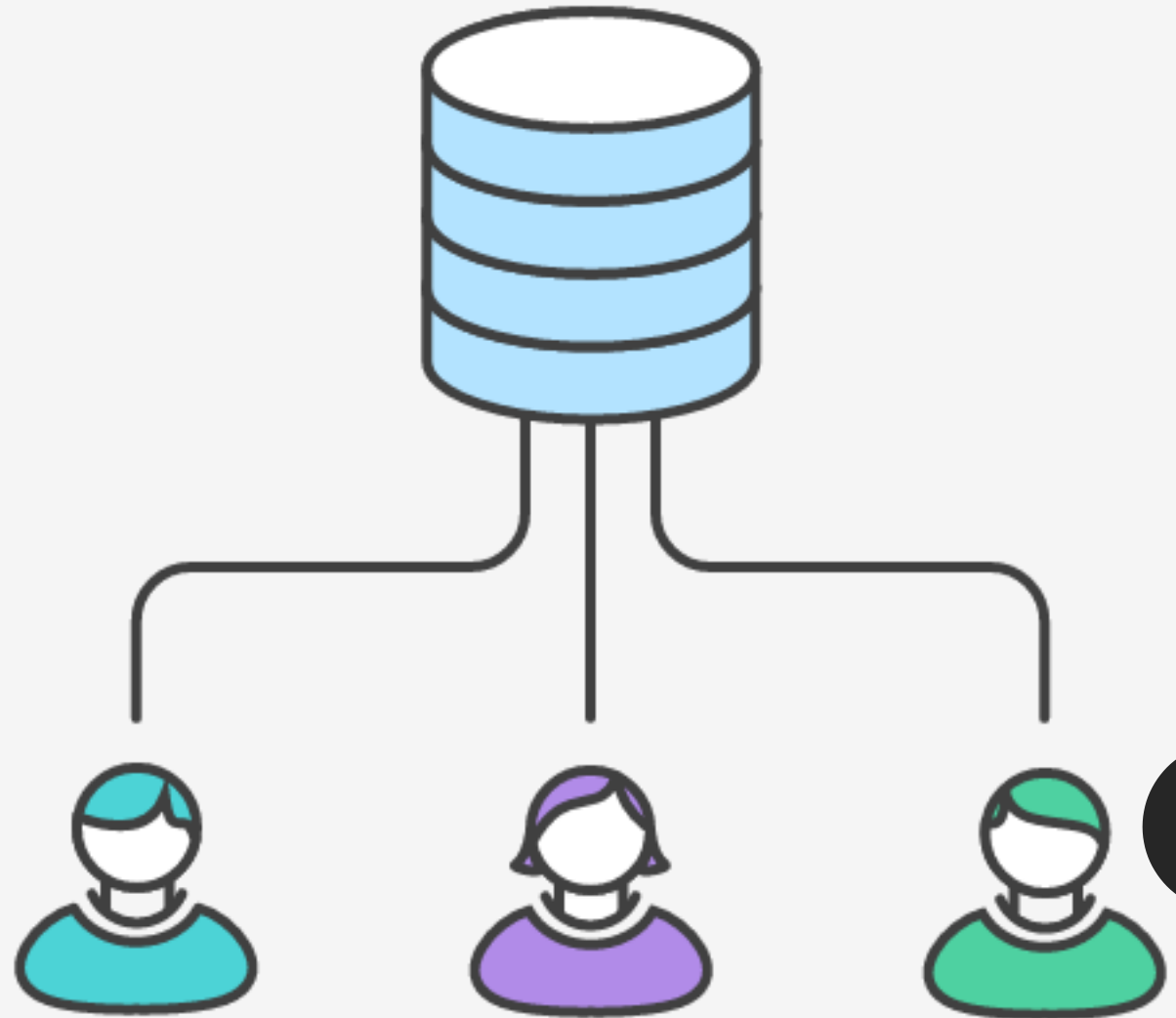
ssh user@host **git init** --bare /path/to/repo.git



*Everybody  
clones the  
central  
repository*

---

```
git clone ssh://user@host/path/to/repo.git
```



*David  
works on  
his feature*

---

git status # View the state of the repo

git add <some-file> # Stage a file

git commit # Commit a file</some-file>



*Maria  
works on  
her own  
feature*

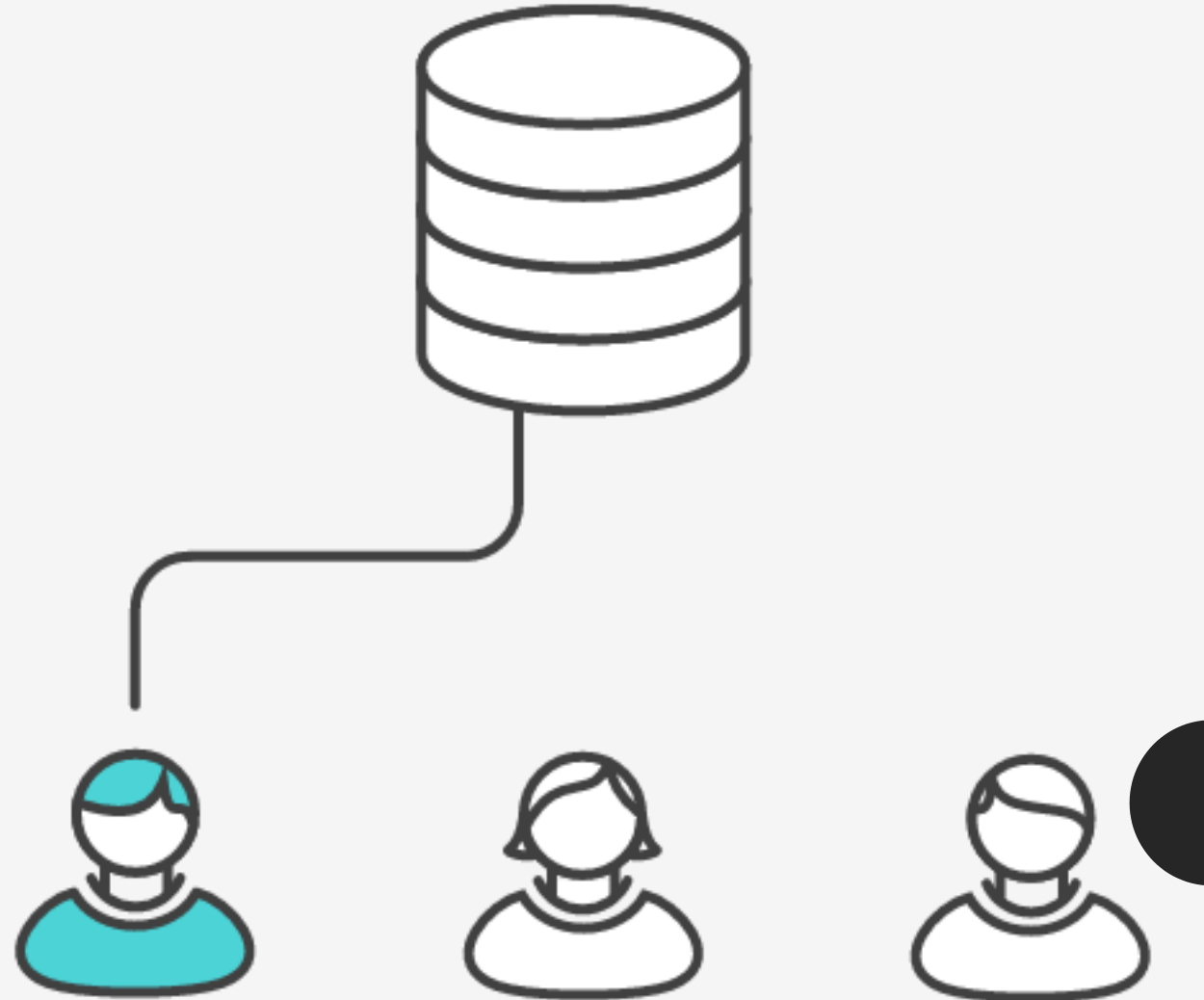
---

git status # View the state of the repo  
git add <some-file> # Stage a file  
git commit # Commit a file</some-file>



*David  
publishes  
his feature*

git push origin master





# *Maria tries to publish her feature*

`git push origin master`

error: failed to push some refs to '/path/to/repo.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')

hint: before pushing again.

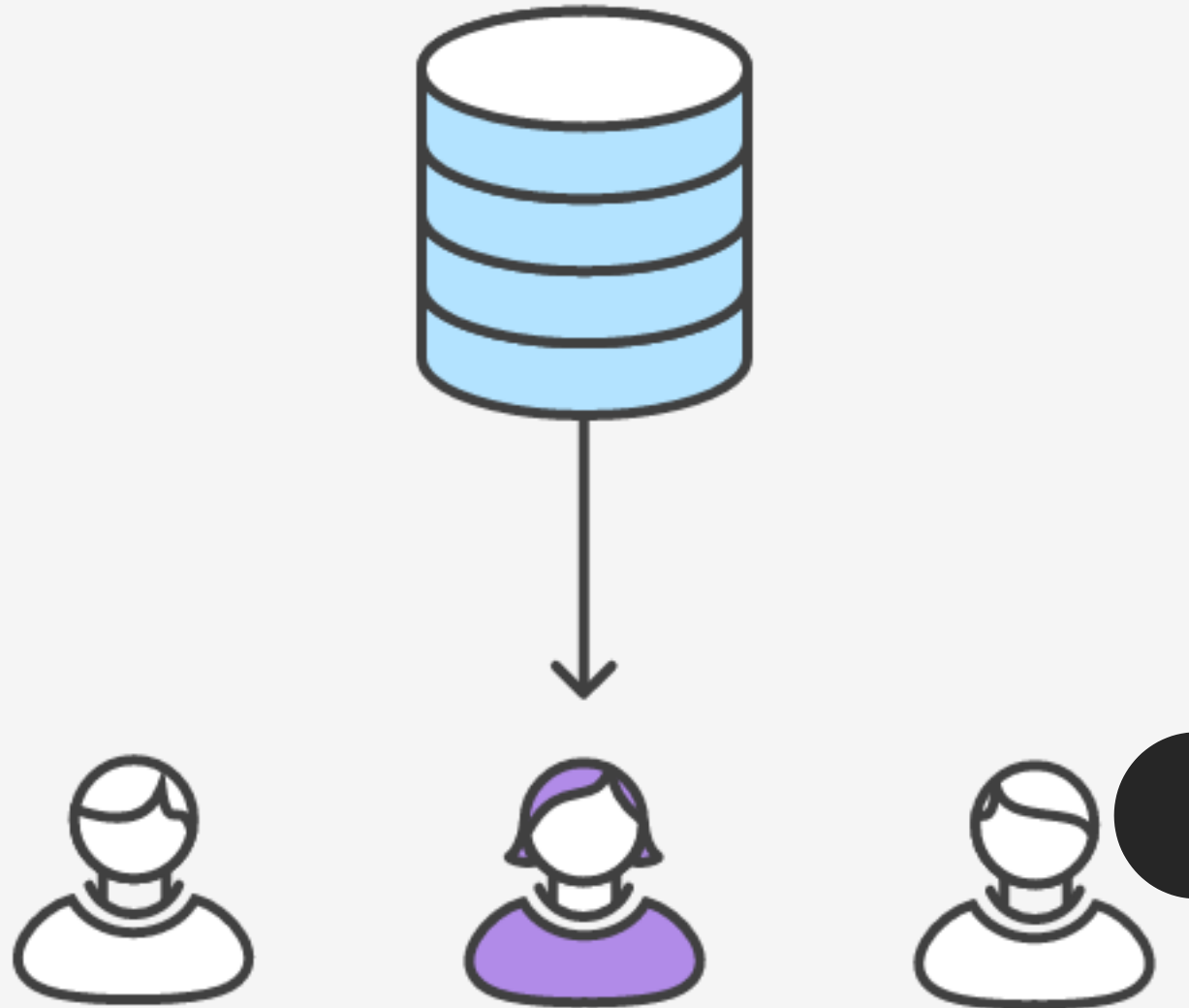
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

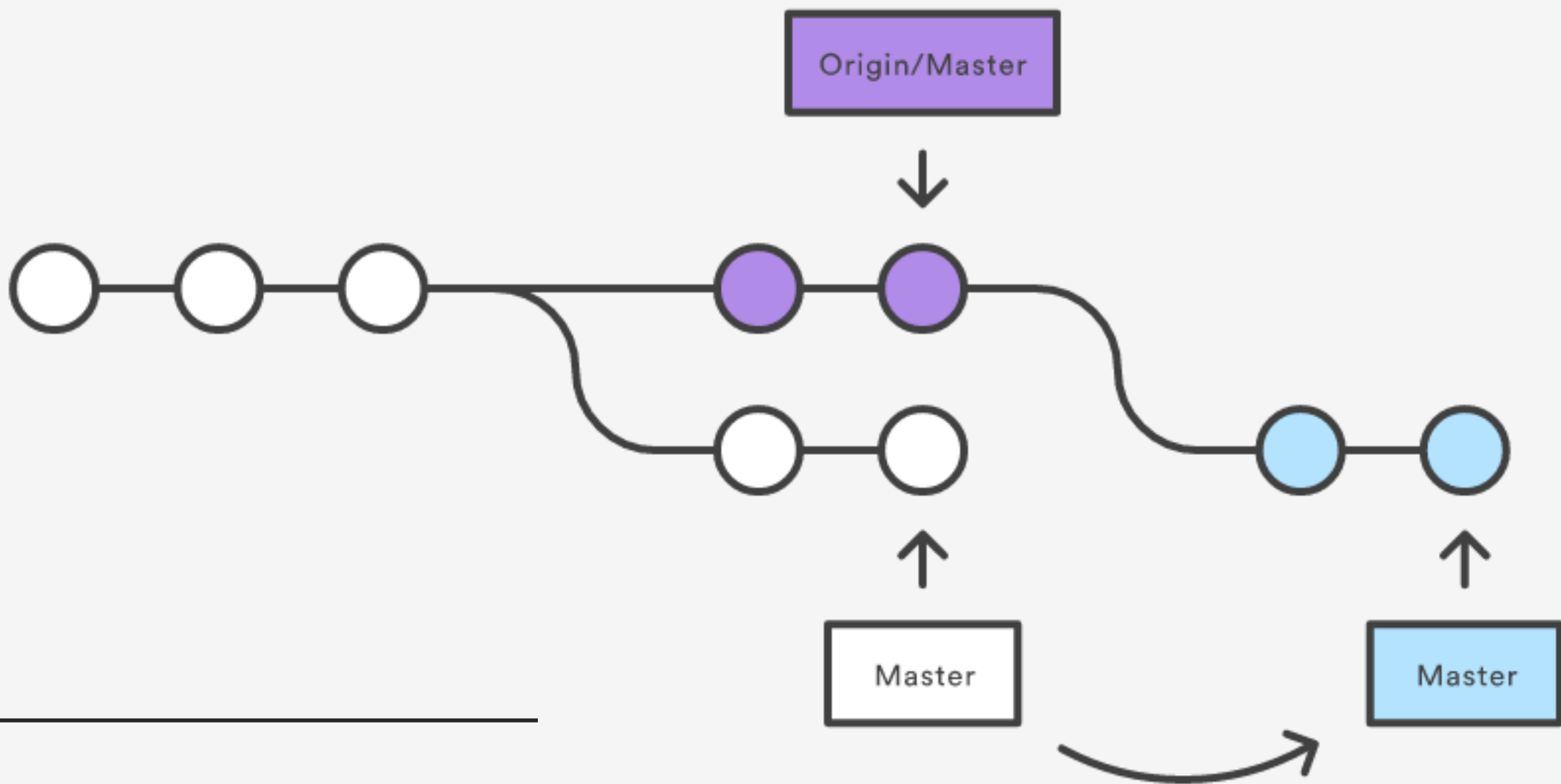


*Maria  
rebases on  
top of  
David's  
commits*

---

git pull --rebase origin master





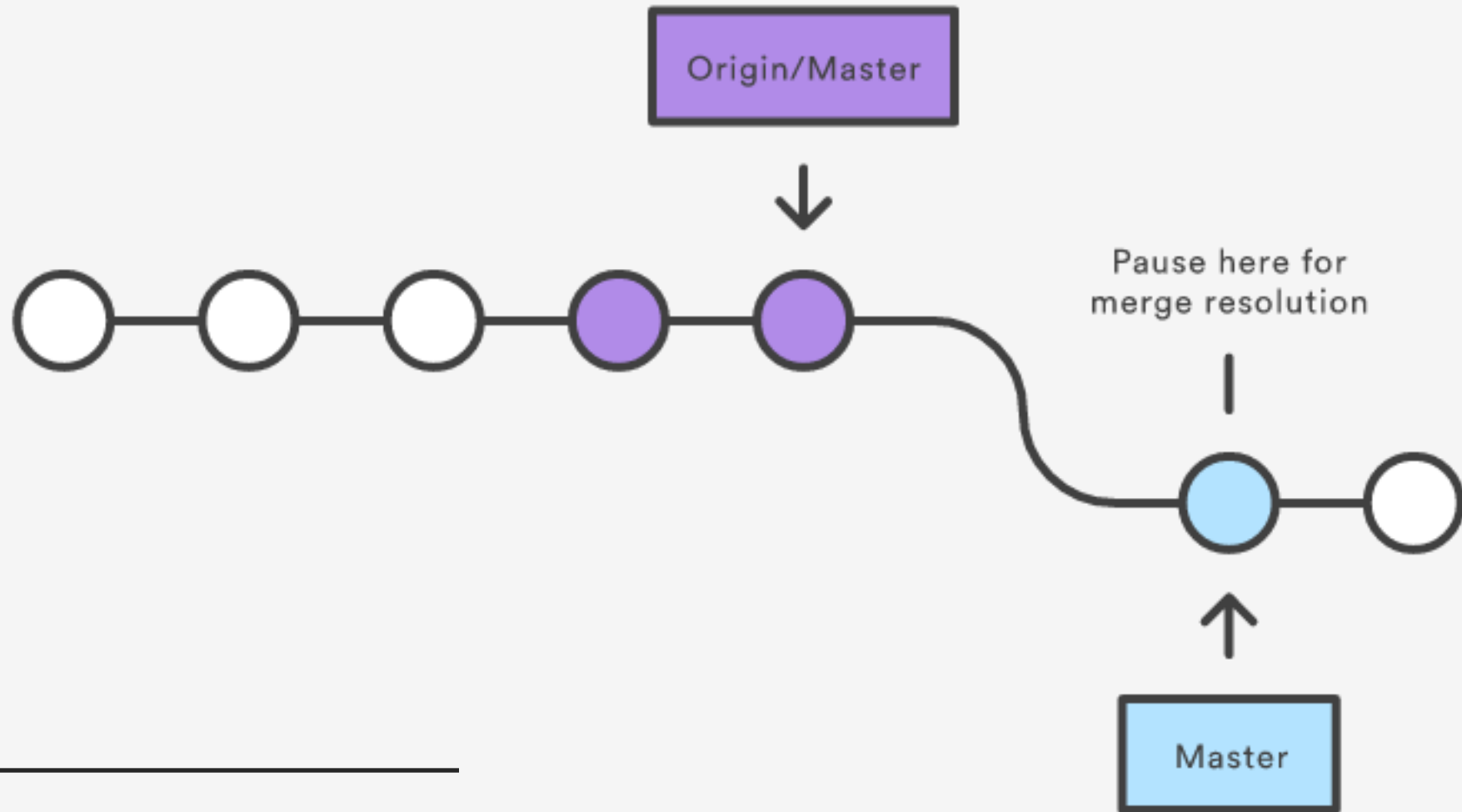
# *Maria resolves a merge conflict*

---

CONFLICT (content): Merge conflict in <some-file>



CONFLICT (content): Merge conflict in <some-file>



# Unmerged paths:  
# (use "git reset HEAD <some-file>..." to unstage)  
# (use "git add/rm <some-file>..." as appropriate to mark resolution)  
#  
# both modified: <some-file>

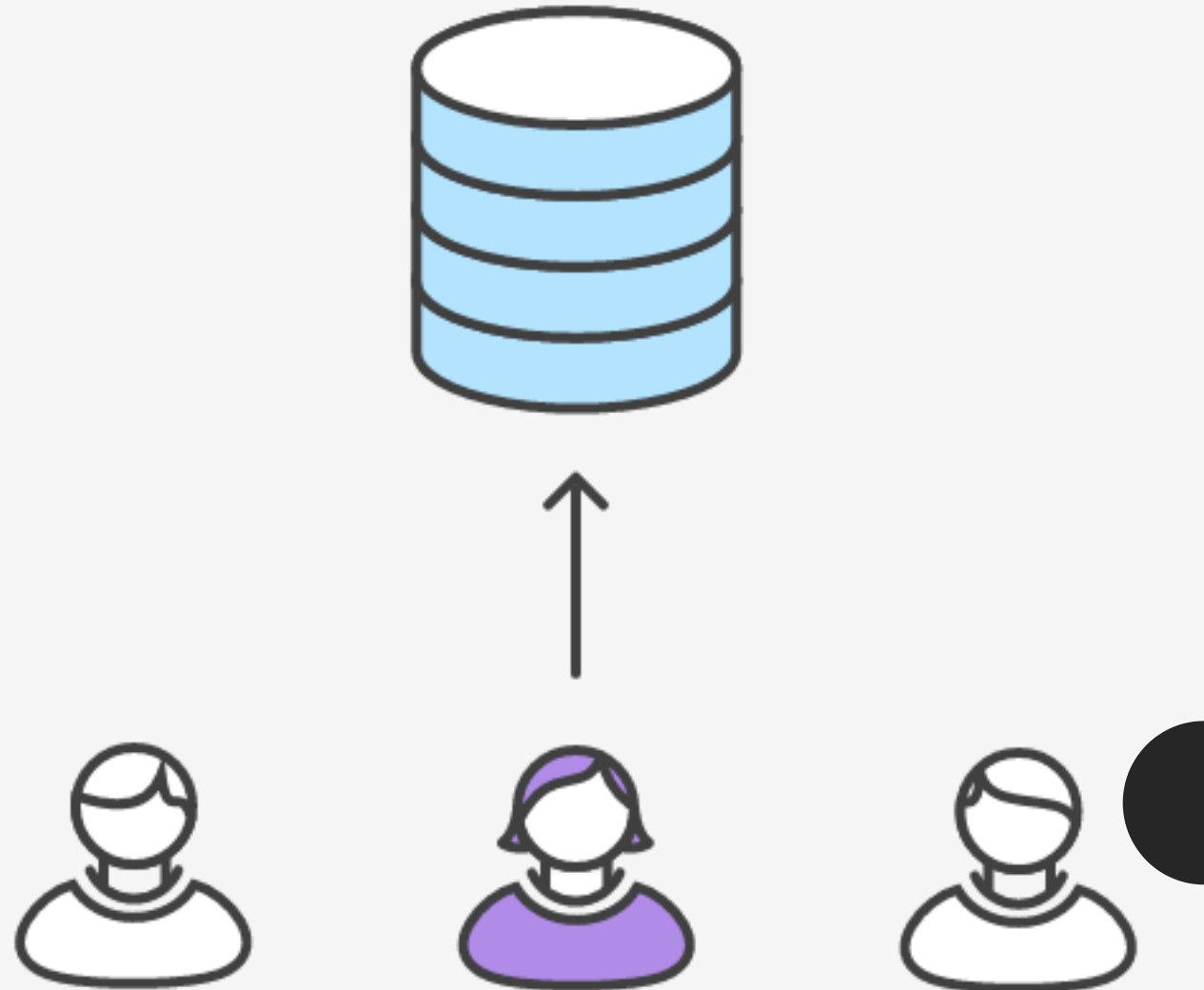
git add <some-file>  
git rebase --continue



*Maria  
successfully  
publishes  
her feature*

---

git push origin master





	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.



*Questions?*

