Repositories and Git

Week 2 ELEC 376

Why do we need Repositories

- The need comes from having **more than one person** working on a single project:
  - Each coder needs to be able to **get the latest, tested version** of another coder's source code files at any time, **from any place**.
  - Any coder needs to know **why changes have been made** to a source code file.
  - Any coder might need to **go back to an older version** of the source code in case the newer version is causing more problems than the older one…
  - (Source code is a company asset - the company needs to know where it is located, so that it controls access to the code.)
- (Or a single coder just wants a backup for his code…)

Functional Requirement of a Repository

- A **single location** that contains the **latest version** of all source code files.
- A single location that stores **previous versions** of all source code files so that you can revert, if needed.
- A system that **keeps track of version** numbers automatically and can produce a history of changes including who made the changes, when and for what reason.
- A system that can be **accessed from anywhere** and preferably is still useable if off-line.
- A system that can protect some files from being changed and can limit what some users can tamper with.
- A system that helps to **resolve conflicts**.
- A system that allows you to **link file commits to issues** for integration.
- A system that can **work with a variety of client tools**, including IDEs.

Centralized Repositories

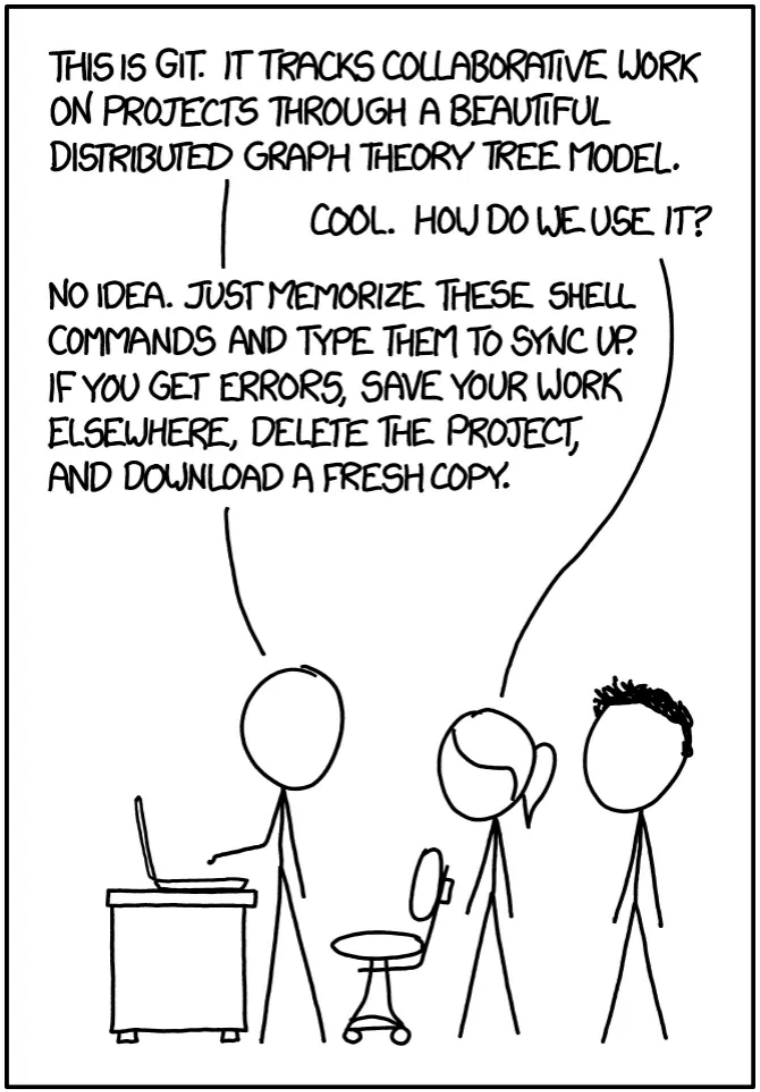
- Centralized
  - Commercial ($$): Perforce: http://www.perforce.com
  - Older: CVS: http://www.cvshome.org
  - Replaced CVS: Subversion (or “SVN”): http://subversion.apache.org/
  - CVS: “Concurrent Versions System”
  - SVN: “Apache Subversion” or just “Subversion”

Git – the best!

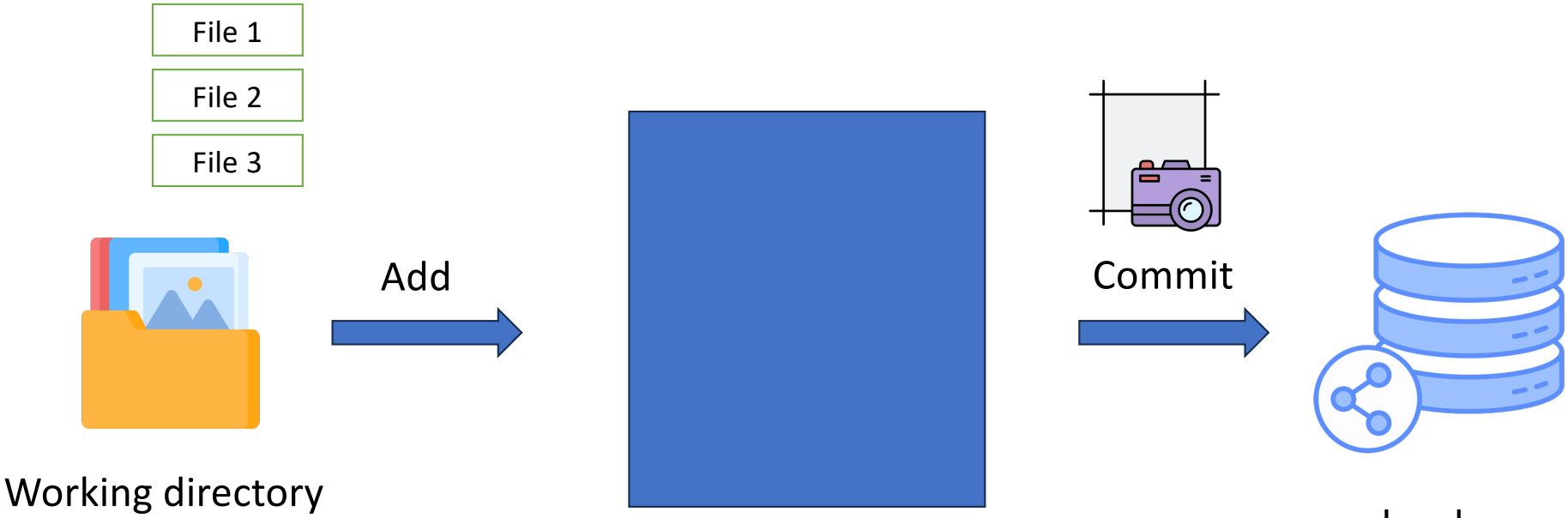
- Git is a distributed version control software/technology that tracks changes in a set of files
- Originally designed by Linus Torvalds in 2005
- De-centralized: Everyone has a repository and all the necessary history and version log information in compressed format
- Much faster and smaller footprint than SVN or CVS. Less required server interaction!

What Git isn’t

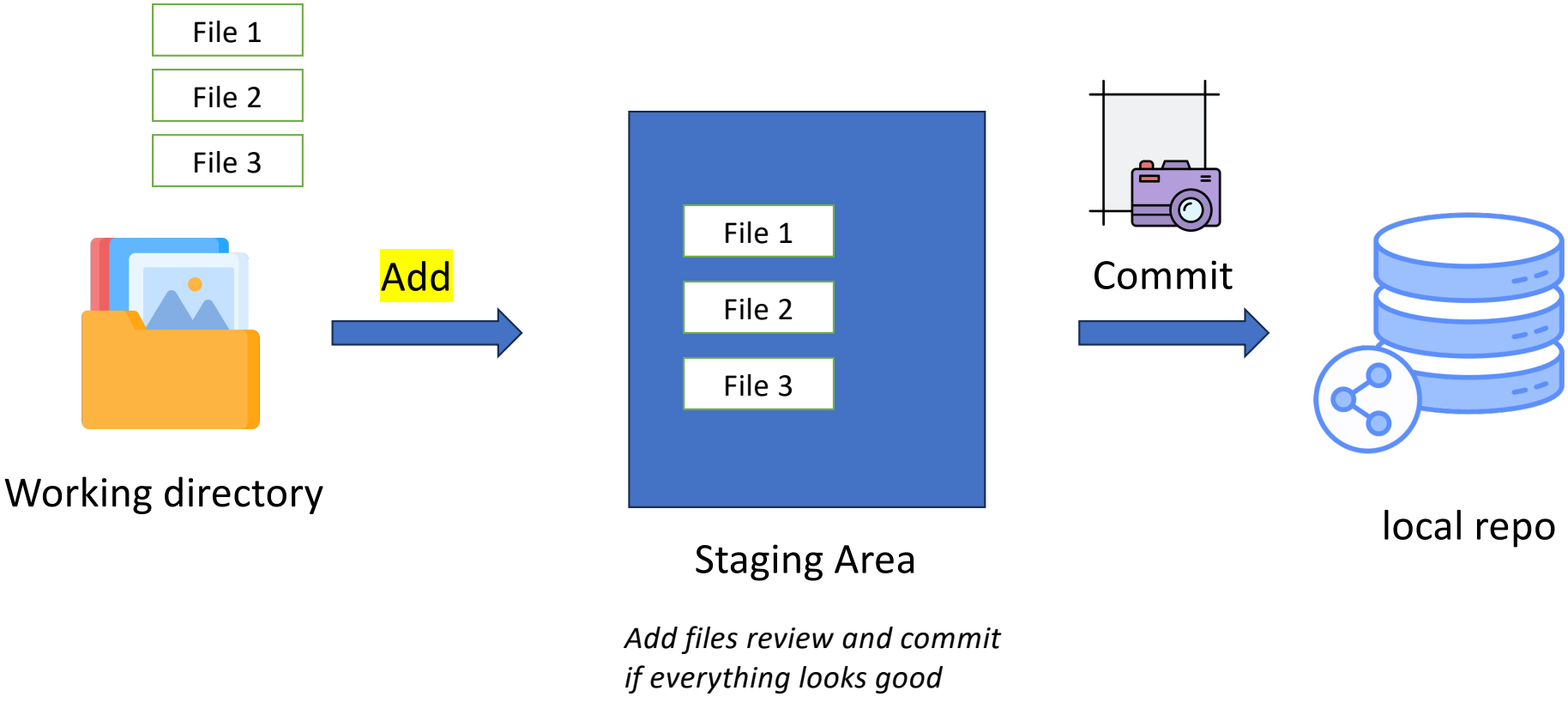
- Git is NOT a website (often mixed up with GitHub)
- Git is NOT an automatic backup of your data / project
- Git is not MAGIC, will not automatically solve your merge problems


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.

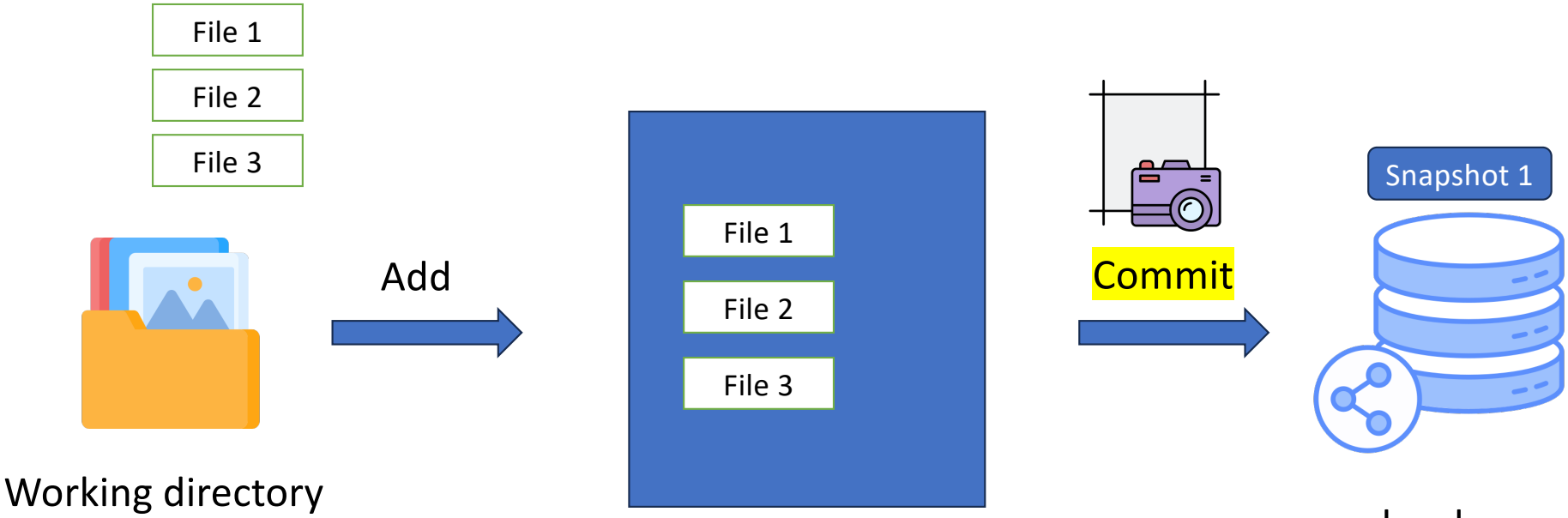
But we do want to have an idea, right?



But we do want to have an idea, right?

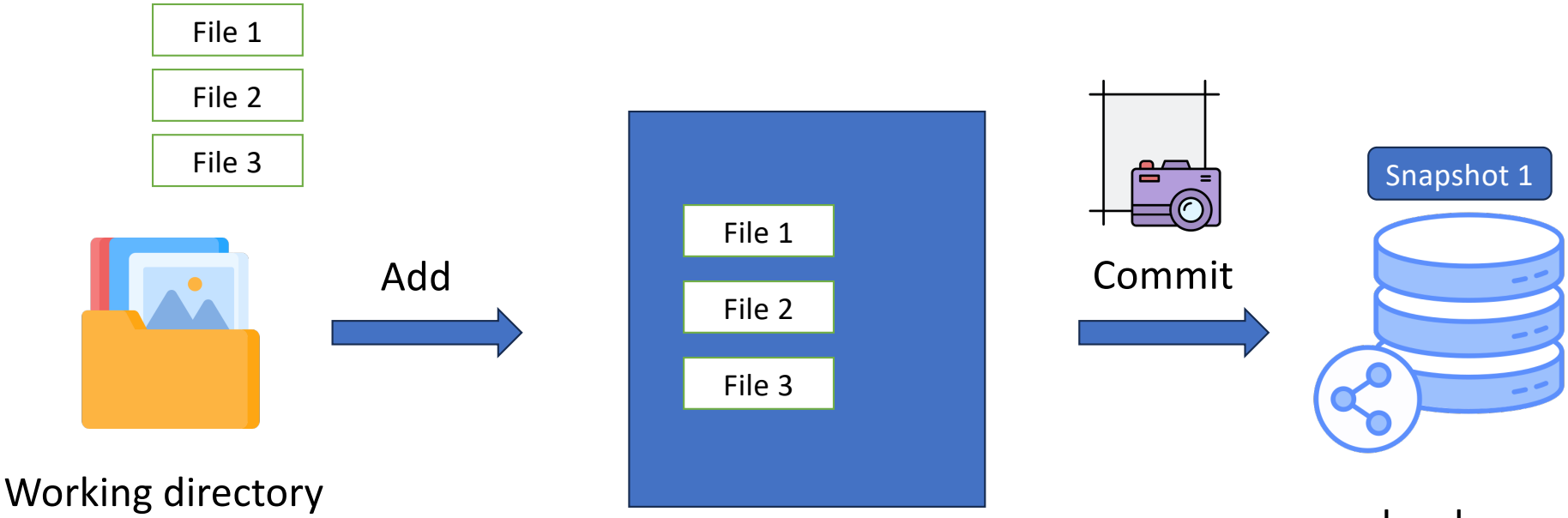


But we do want to have an idea, right?



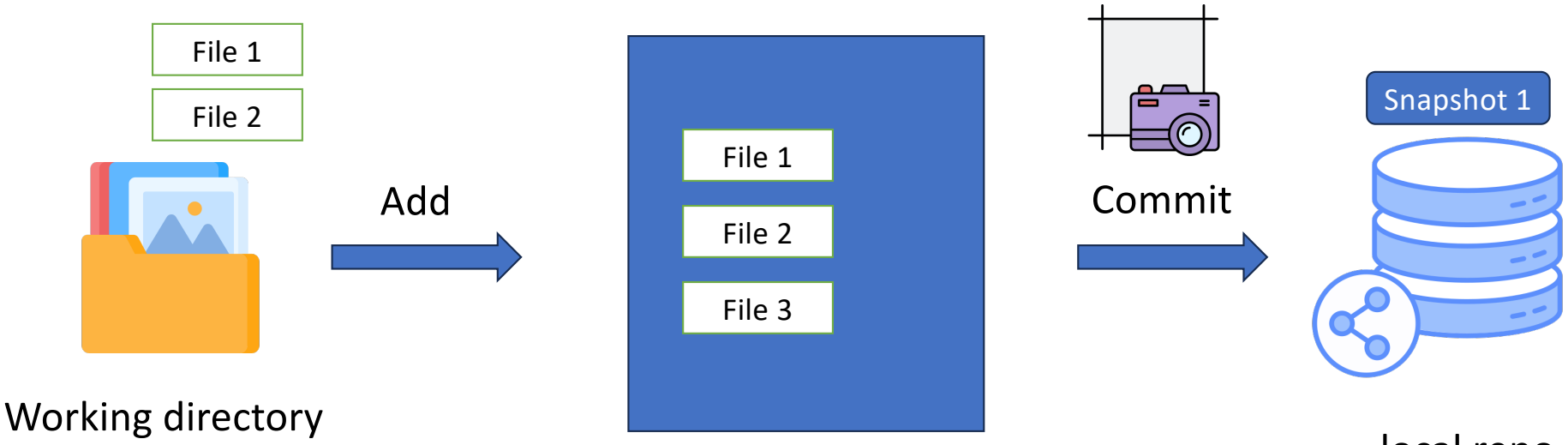
Add files review and commit
if everything looks good

But we do want to have an idea, right?



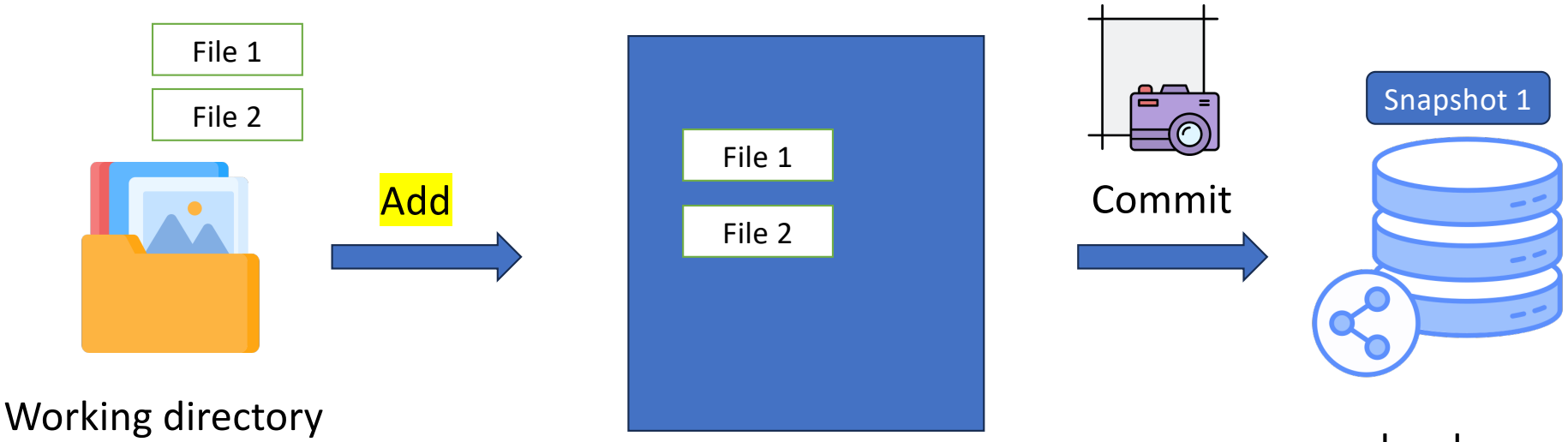
Add files review and commit if everything looks good

We delete file 3



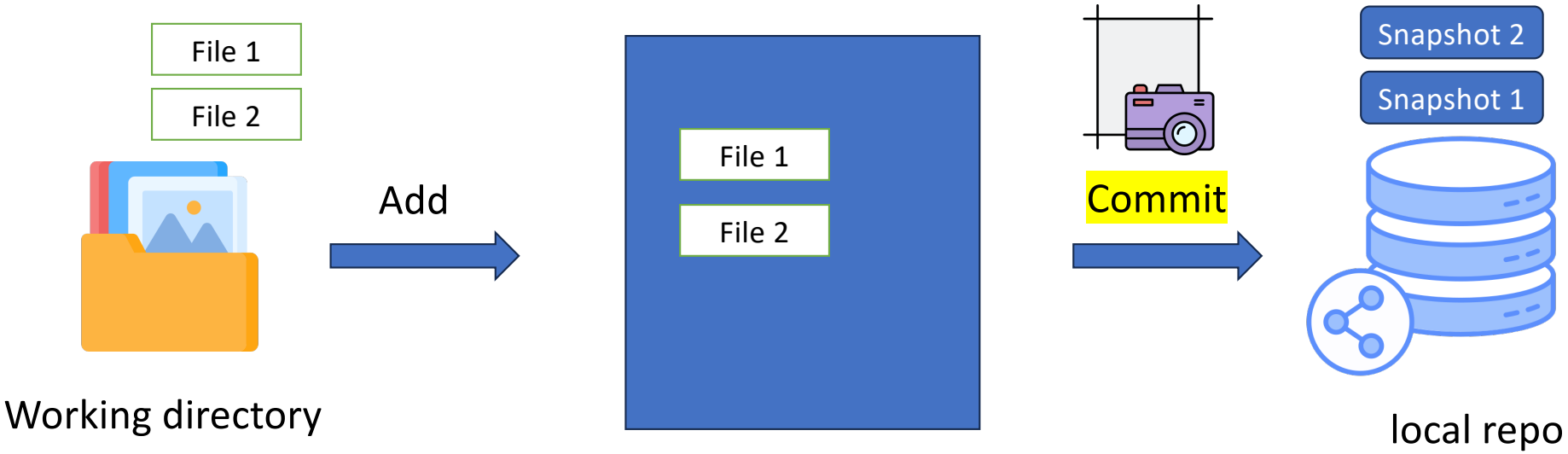
Add files review and commit if everything looks good

Add file 3



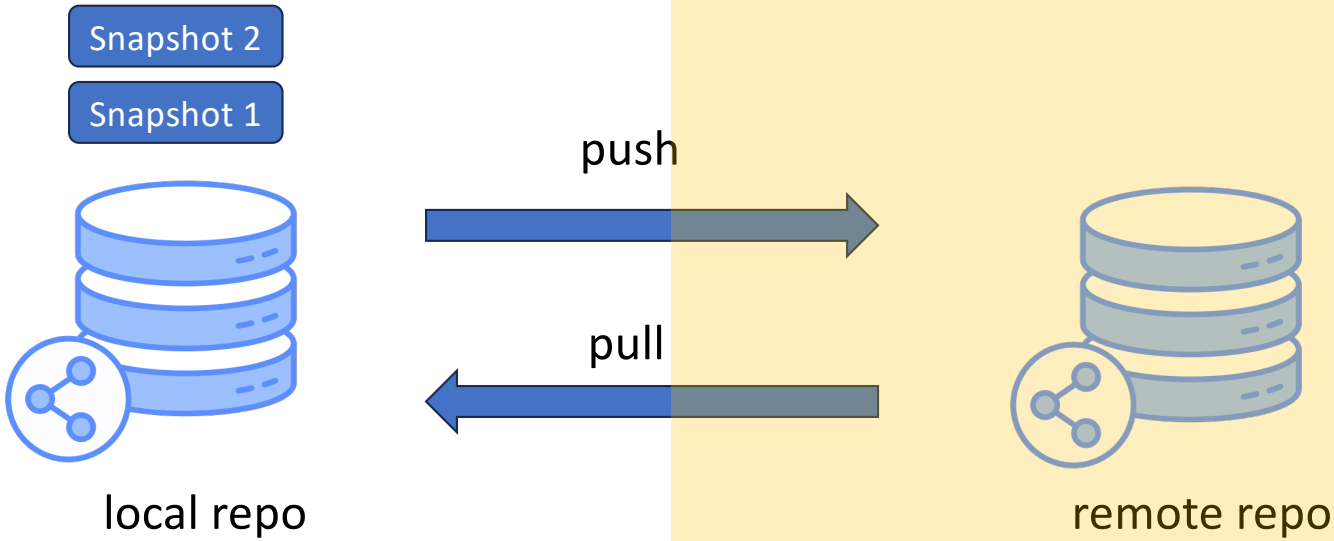
Add files review and commit
if everything looks good

Commit the current state



Add files review and commit
if everything looks good

Push/ pull changes to/ from remote repository

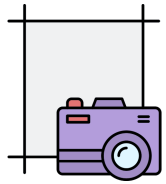


Main commands

You clone to create your repository.

You add to put files in the **staging area**.

When you are ready, you commit files to your local repository.
You have created a snapshot of your repository.



A push sends your snapshot out to the remote repository.
Other users will get your updates “pull”:

Main commands

- A pull carries out a fetch followed by a merge of the Head branch. This will update the files in your working directory to the latest versions.

- If two people have changed the same file before the merge operation you get a merge conflict that has to be resolved.

- Features can be developed in an isolated way by creating a branch. Eventually (at the end of a sprint?) it will need to merge back to the "master" branch.

Initalize a new repository

- Git init
- Git add .
- Git commit –m “created new local repo”
- Git remote add origin
- Git push --set -upstream origin master

Clone repository

In terminal directory you want to be in

```
git clone
```

https://code.smithengineering.queensu.ca/elec376/Elec376_F25_Playground.git

Log in with username and pass (student ID)/ token

```
git status
```

Three step save

- git add
- git commit –m “meaningful comment”
- git push

Other commands

- git commit -a -m (bypass git add)
- git log
  - git log –p
  - git log --oneline

- git branch branchname
- git switch branch
- git merge –m “message” banchname
- git branch –d branchname

Reverting to older commit

- `git reset commitsha`
- `git reset –hard`
- Can also go back to specific commit
- `git reset ––hard 0d1d7fc32 (your specific ID)`

git ignore file

- Not all files should be tracked in the repository
- Build files, object files, images, etc
- The git ignore file can be used to tell git to ignore certain filenames, file types, directories and more.
- Directory structure

Demo