INF5750

Version control

University of Oslo
Department of Informatics

Background

- Version control system essential when collaborating on code
- Use version control system both for individual assignments and group projects
- Git will be used in this course

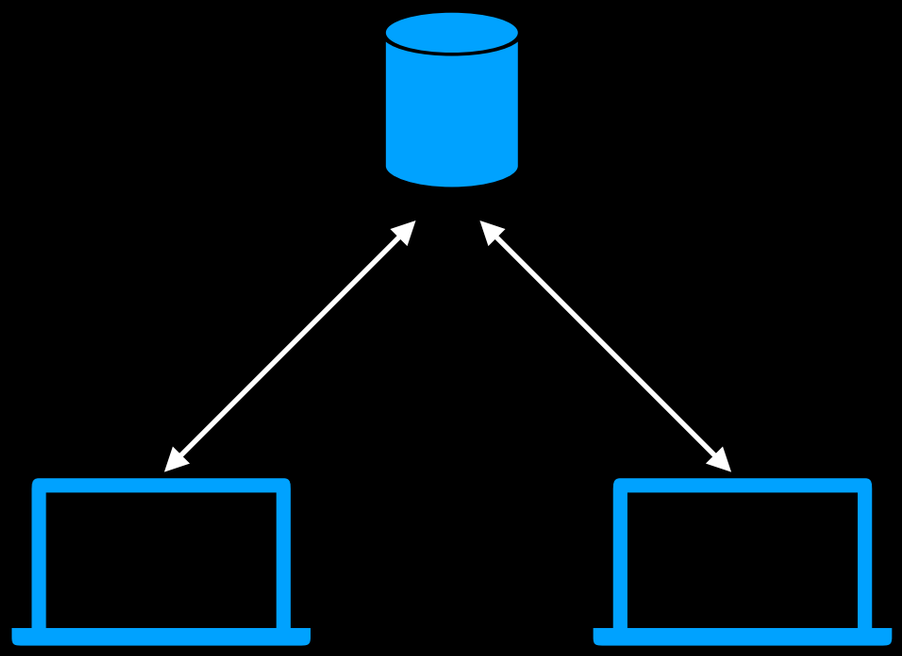
Repository

- Where everything is stored
- Stores every change to any file that has been *commited*
- Administrative information is kept in hidden folder (for example .git)

Centralized vs distributed

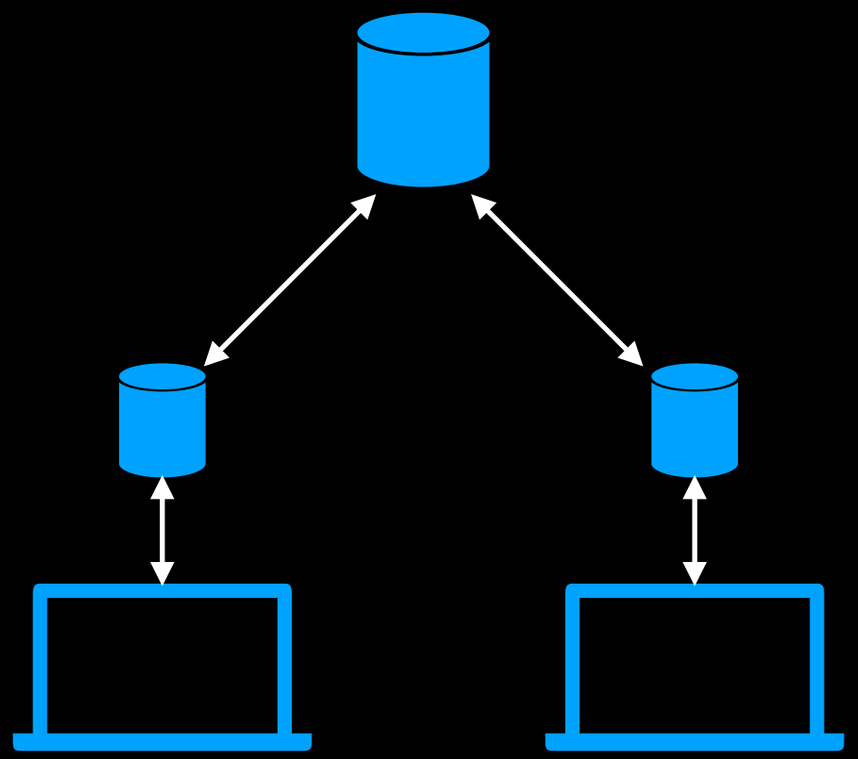
Centralized

Each developer works
against one central repository



Distributed

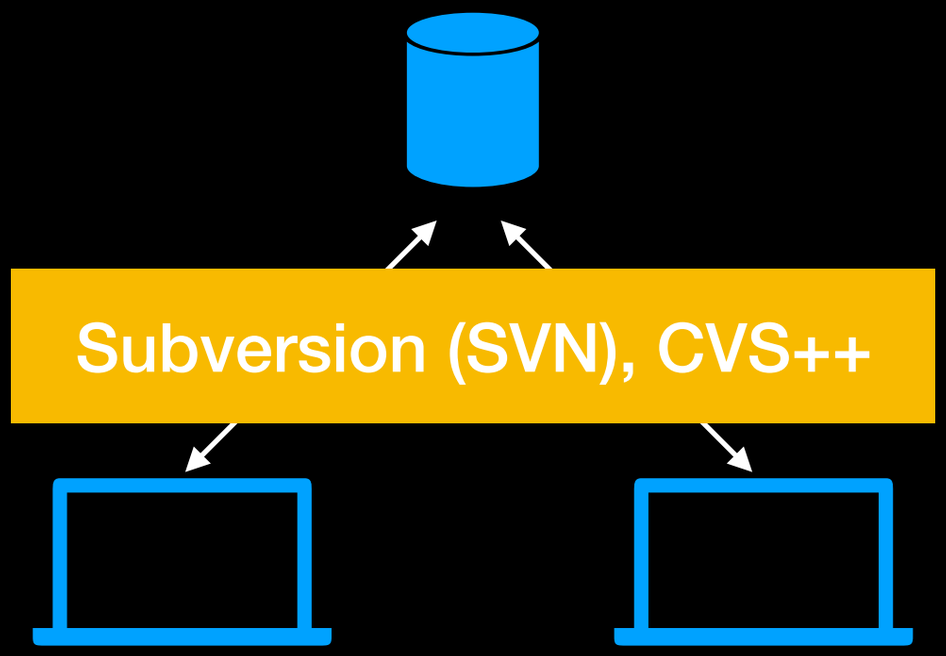
Each developer has a full local copy of the repository, and pushes changes from the local to the central repository



Centralized vs distributed

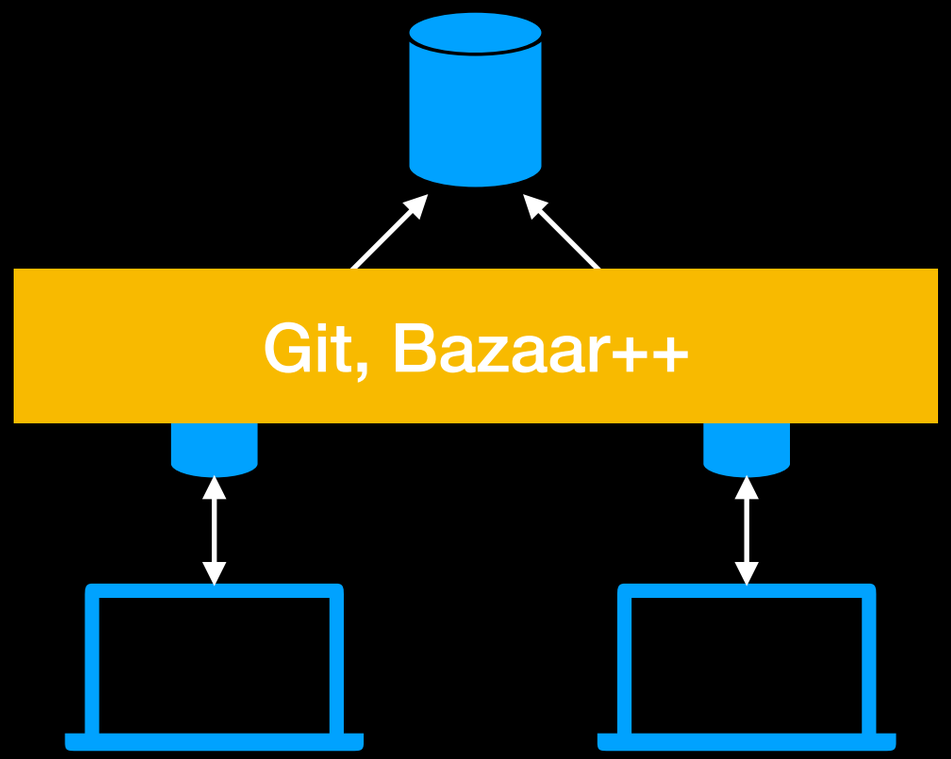
Centralized

Each developer works
against one central repository



Distributed

Each developer has a full local copy of the repository, and pushes changes from the local to the central repository



Centralized vs distributed

- Advantages of distributed version control:
    - Several smaller commits can be made privatly then pushed together as a whole
    - Everything but pushing/pulling to central repository can be done offline
- Disadvantages of distributed version control:
    - Repositories with large files or long history can become big

Working tree

Repository

Staging area

Working tree

checkout

src

index.html

app.js

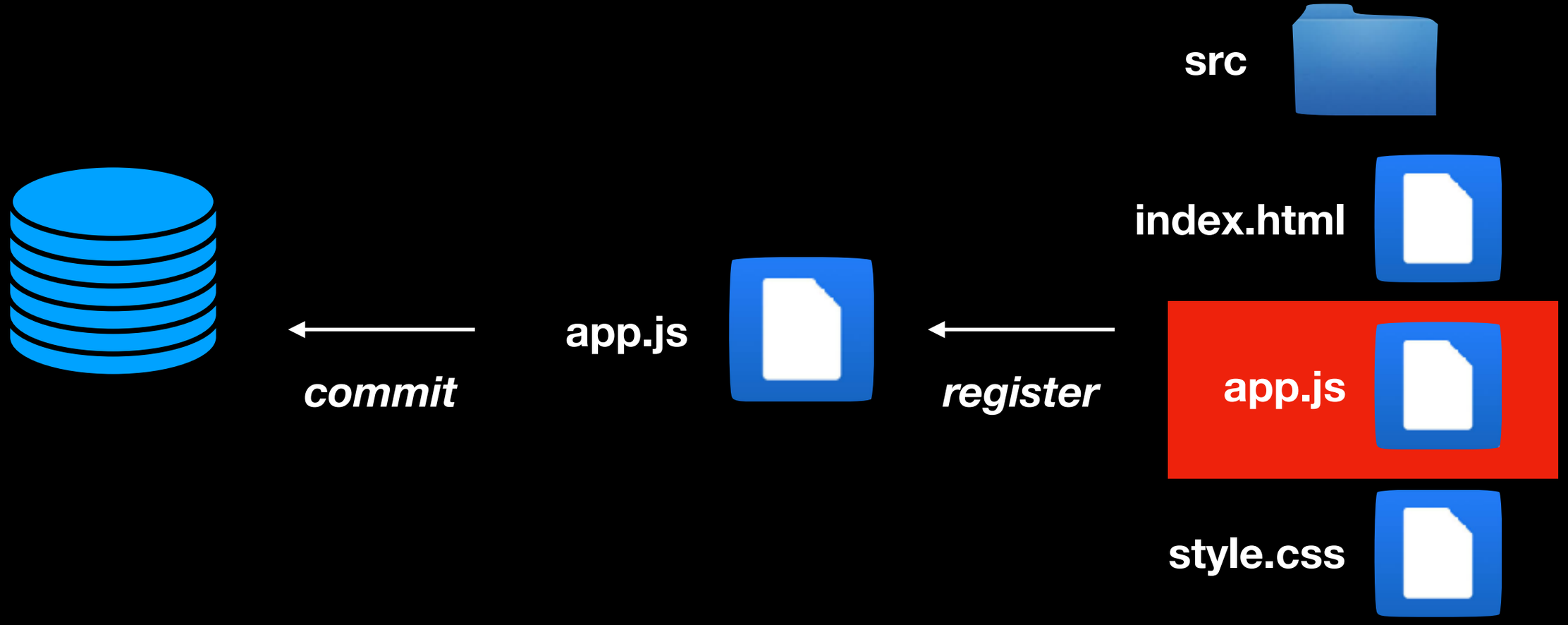
style.css

Working tree

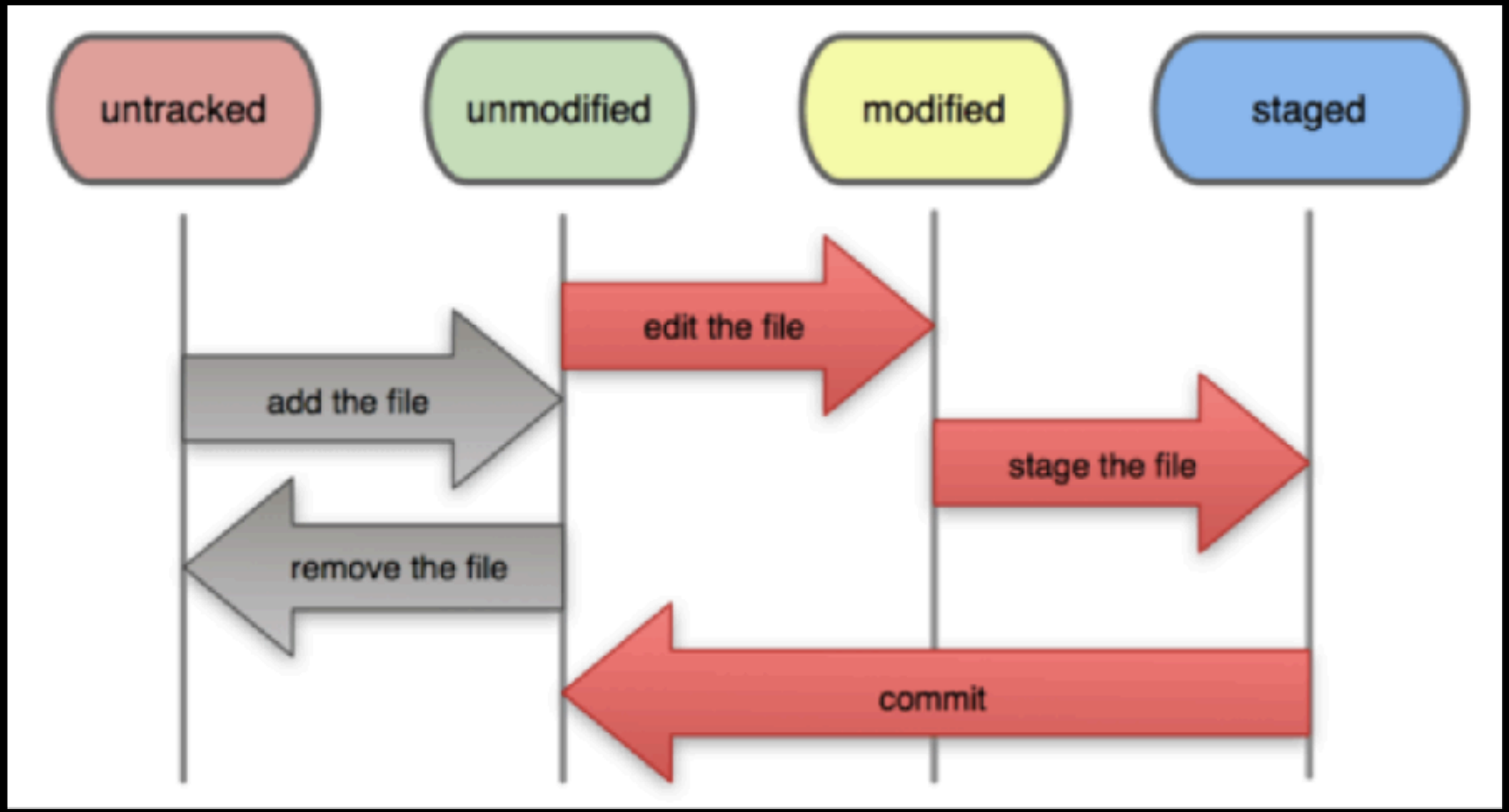
Repository

Staging area/
index

Working tree



File status


untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

Revisions

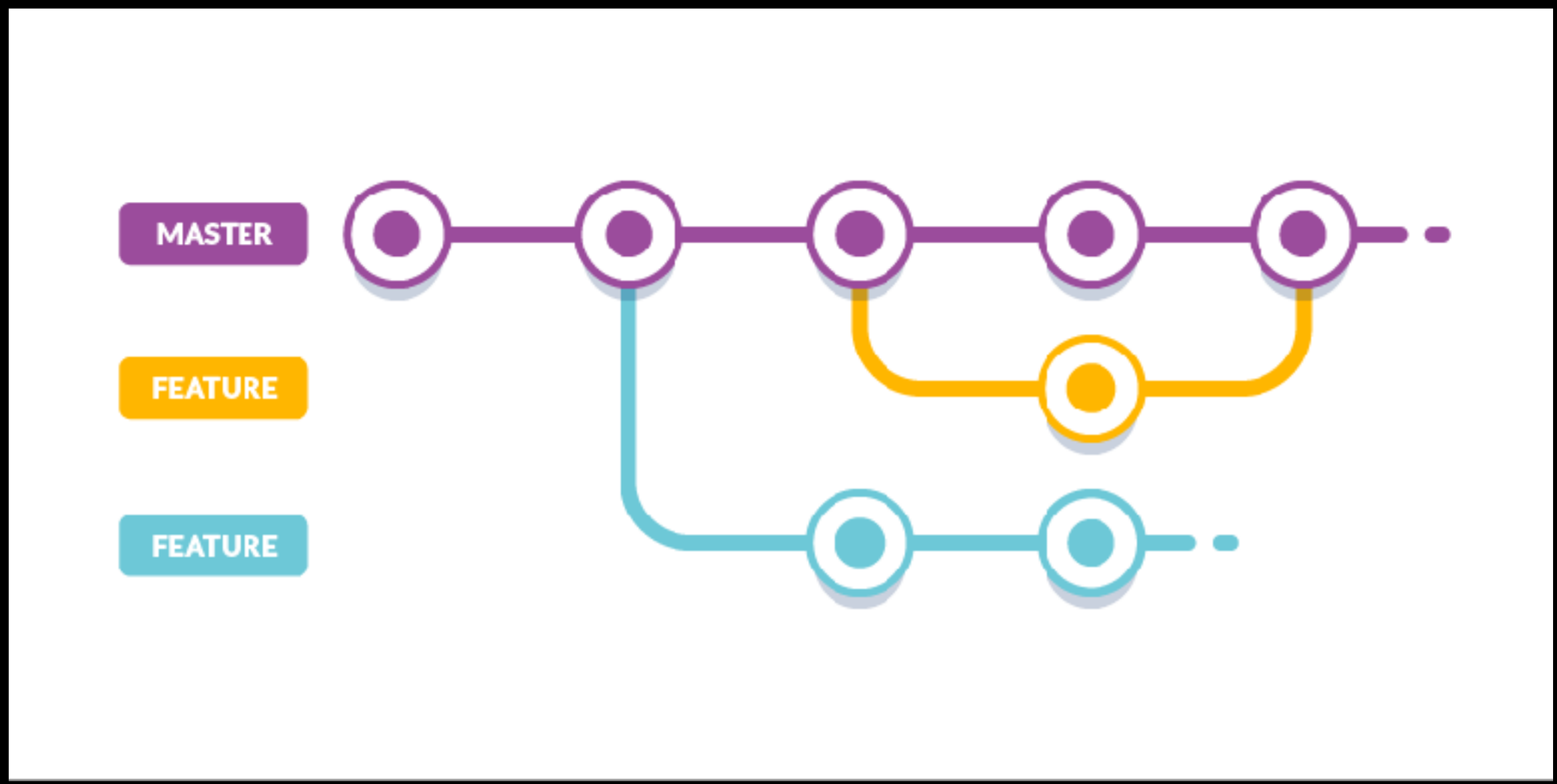
- Each *commit* creates a *revision* with a unique identifier
- Each *revision* refers to a state of a branch
- Revisions can be:
  - checked out - creates a working tree with the state of the branch when it was commited
  - reverted - undoing changes in that commit

Branches

- *master* is the original/main line of development - also a branch
- A *branch* is a copy of master which exists independently and is maintained separately - can later be merged
- Useful in several situations:
  - Large modifications which takes long time and affects other parts of the system (safety, flexibility, transparency)
  - Different versions for production and development
  - Customised versions for different requirements

Master and branches

- Different workflows/philosophies exists - features branches, relsease branhces, development branches



source: https://buddy.works

Conflicts

- A *conflict* occur if more than one developer modifes the same part of a file
- Git tries to auto-resolve conflicts
- Must be resolved manually if auto-resolve fails

Pull Requests

- Useful when working on repositories without permission to commit

- *pull request* - request that the maintainer of the repository pull and merge your changes

- Two models:

    - fork repository - pull requests from private to central repository

    - make branch in repository - pull requests from feature branch to master

Good practice

- Update (fetch changes and merge) and test *before* commiting
- Update before editing
- Commit often - check changes before commiting
- Add source code and configuration files
- Don’t add generated files, libraries, IDE files etc

Resources

Git basics:
http://rogerdudler.github.io/git-guide/

Git and GitHub at UiO:
http://www.uio.no/tjenester/it/maskin/filer/versjonskontroll/github.html [norwegian]
https://github.uio.no

source: https://xkcd.com