An introduction to git and conda

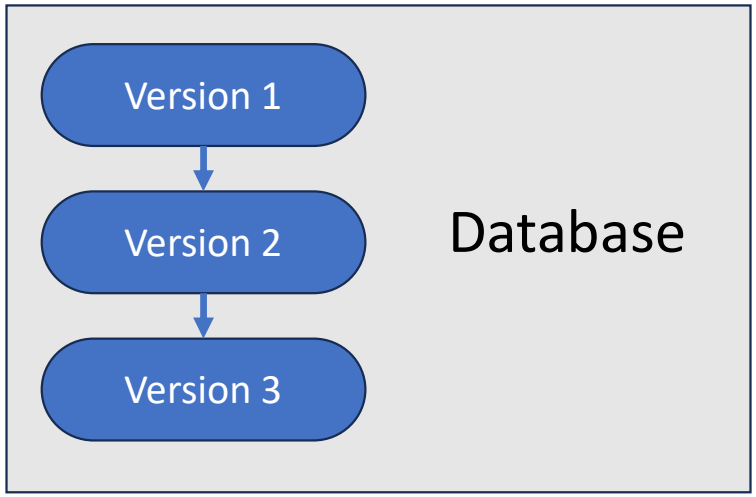
Daniel J. Bridges, 2023/10/14

Git
A brief history

- Created in 2005 by Linus Torvalds the creator of Linux
- Designed to do version control on Linux kernel
- Goals of Git:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently
- Name?
  - A "git" is a cranky old man. Linus meant himself

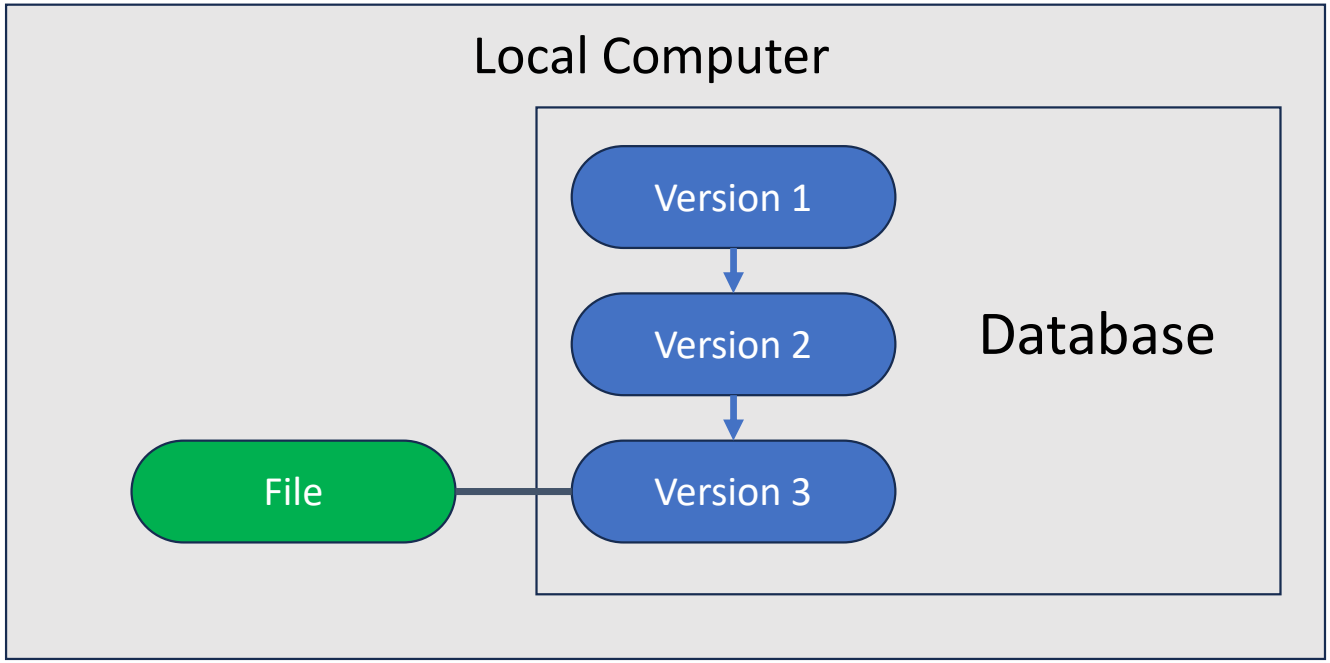
Version Control System
The basics

- A system that records changes to a file or set of files over time so that you can recall specific versions later
- Can be applied to essentially any file(s) type(s)
- Making a snapshot in the Vbox is versioning the state of all files in the guest machine (albeit less formally for recall)
- A database (repository) can be used to record the changes between file versions



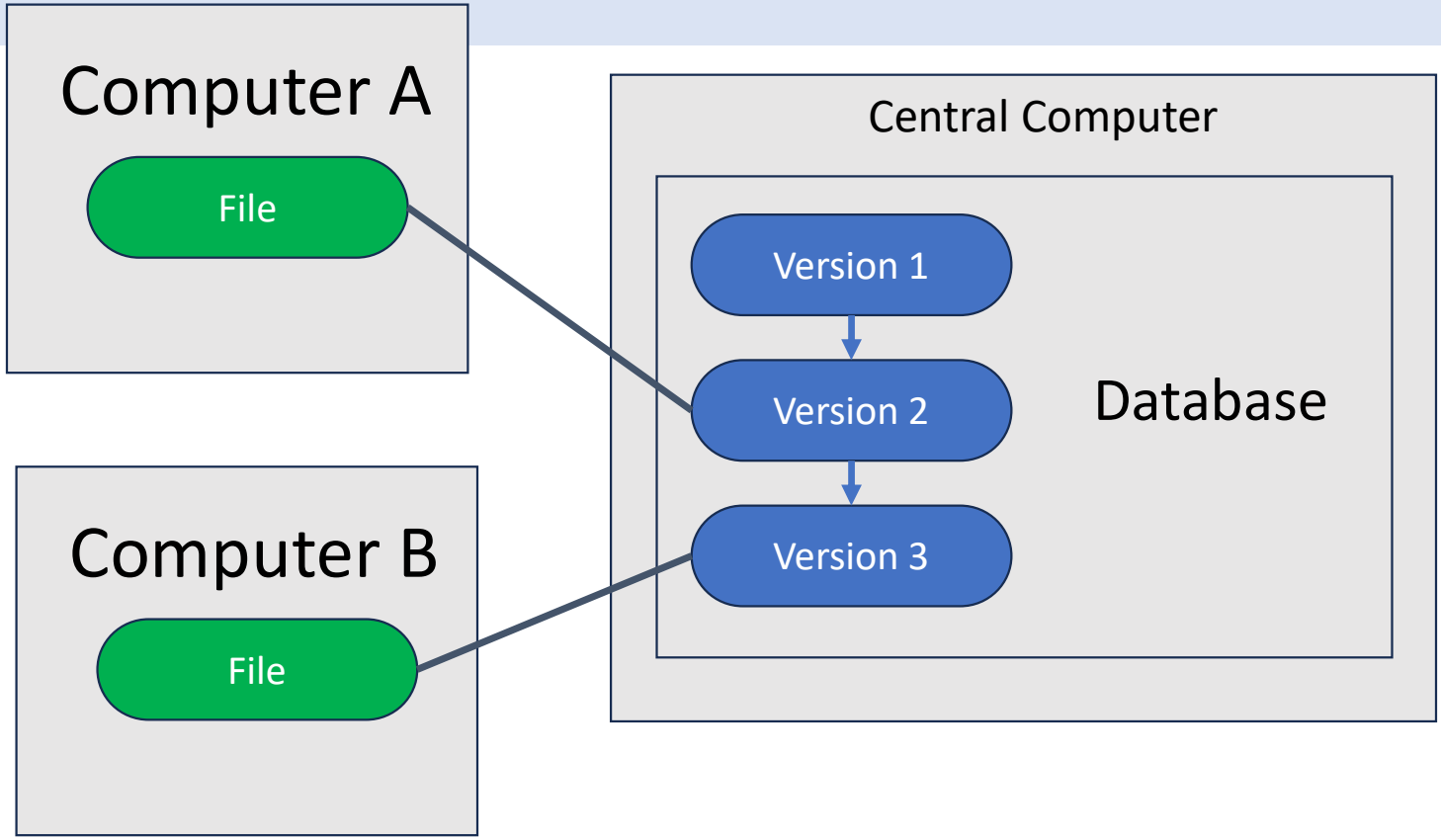
Version Control System Types

- Local



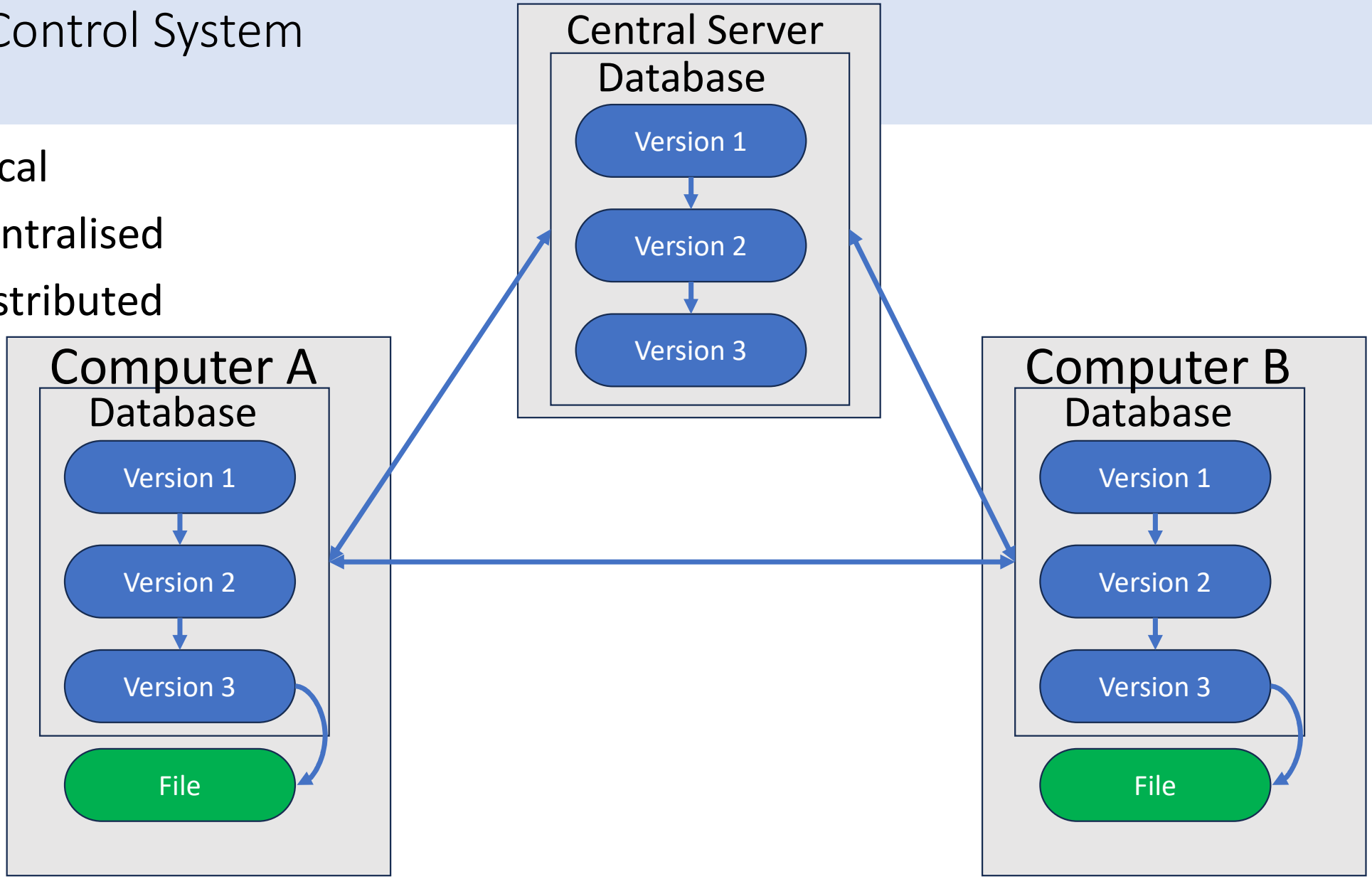
Version Control System Types

- Local
- Centralised



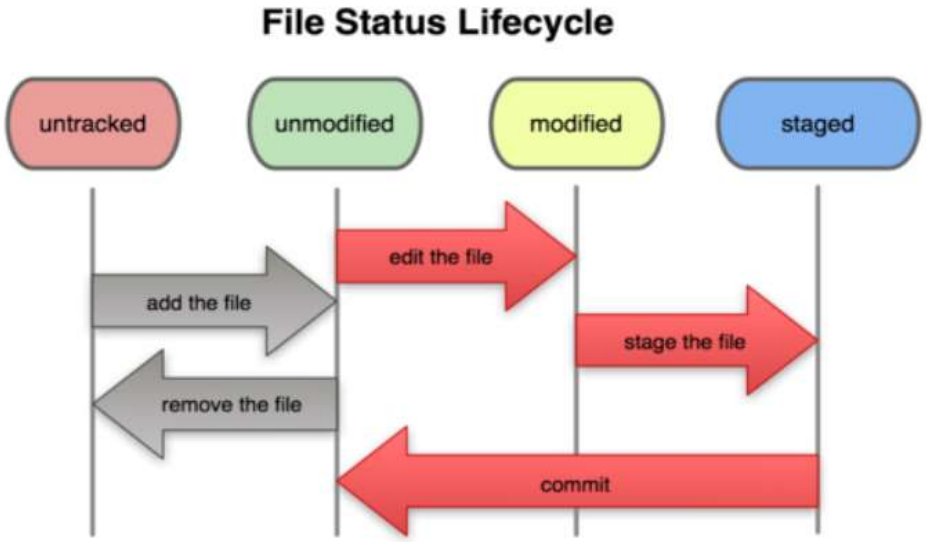
Version Control System Types

- Local
- Centralised
- Distributed



git

- Your repo (wherever it is) is a complete copy of everything
- General Process:
  - Clone or init a repo
  - Edit files as needed
  - Stage files that are ready to have a snapshot made
  - Commit – store snapshot permanently in your git directory



Common commands

command	description
`git clone` ***`url [dir]`***	copy a Git repository so you can add to it
`git add` ***`file`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Can get very complicated – a whole language and process

Github

- The pre-eminent online git storage platform
- Free version with limited private repos
- Description page to outline repo usage, application etc:
  - https://github.com/JasonAHendry/nomadic3

conda

- Package, dependency and environment management for any language – python, ruby C++ etc
- Allows you to easily switch between different software set-ups without interfering with each other
- Why?
  - Software is constantly being updated, which may or may not be retro-compatible
  - Code designed with a particular package in mind
  - Install the exact version of software required

Conda usage

- Install conda
  - Recommend miniforge: https://github.com/conda-forge/miniforge
- In general expect to create a conda environment per repo (if required). Normally defined in an environment.yml file
- Create the environment
- Activate it
- Use the packages as required

Conda example

```
(base) dan@argon:~$ artic -h
artic: command not found
(base) dan@argon:~$ conda activate artic-ncov2019
(artic-ncov2019) dan@argon:~$ artic -h
usage: artic [-h] [-v]
             {extract,basecaller,demultiplex,minion,gather,guppyplex,filter,rampart,export,run}
             ...

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         Installed Artic version

[sub-commands]:
  {extract,basecaller,demultiplex,minion,gather,guppyplex,filter,rampart,export,run}
    extract             Create an empty poredb database
    basecaller          Display basecallers in files
    demultiplex         Run demultiplex
    minion              Run the alignment/variant-call/consensus pipeline
    gather              Gather up demultiplexed files
    guppyplex           Aggregate pre-demultiplexed reads from MinKNOW/Guppy
    filter              Filter FASTQ files by length
    rampart             Interactive prompts to start RAMPART
    export              Export reads and fAST5 into a neat archive
    run                 Process an entire run folder interactively
(artic-ncov2019) dan@argon:~$
```