DOCUMENT VERSION CONTROL WITH GIT

BEFORE WE START...

WHAT IS VERSION CONTROL?

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- Has existed for almost as long as writing has existed (ex. document version)
- Today, the most capable (as well as complex) revision control systems are those used in software development.

WHY?

- Revert files back to a previous state
- "Freeze" important versions of a document
- Compare changes over time
- Track progress of a project
- See who modified something, and when

MODERN VERSION CONTROL SYSTEMS

- Remote backup of files
- Powerful tool for collaboration

GIT

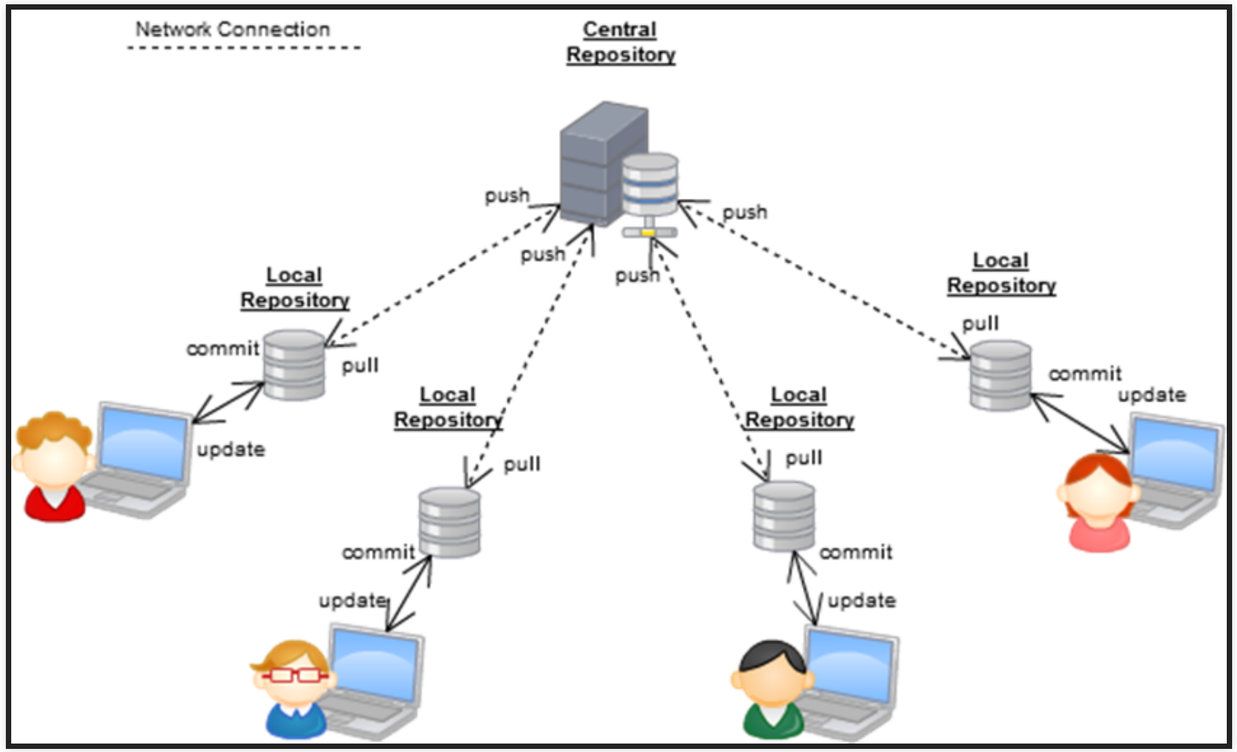
- Developed by Linus Torvalds in 2005
- The linux Kernel:
  - ~63000 files
  - Roughly 15,600 developers from more than 1,400 companies

CHARACTERISTICS

- Free and open source
- Distributed
- Powerful and flexible
- Learning curve can be steep


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.

HOW DOES IT WORK?



INSTALLATION

https://git-scm.com/download/win

Package managers are heavily recommended!

CREATING A REMOTE REPOSITORY

- register at the remote git server
  - https://git.webhosting.rug.nl/
- create repository
- add participants *ssh public keys*
- clone the repository in your machine

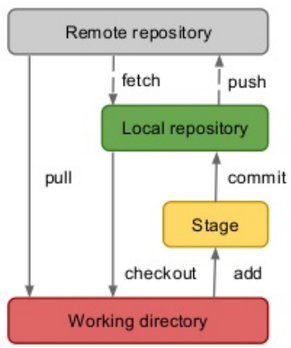
README AND .GITIGNORE

Every repository should have these 2 files:

- **README**: project description and useful information
- **.gitignore**: special file indicating GIT which files are **not** to be tracked

WORKFLOW

Understanding of workflow



- Obtain a repository
  - *git init* or *git clone*
- Make some changes
- Stage your changes
  - *git add*
- Commit changes to the local repository
  - *git commit -m "My message"*
- Push changes to remote
  - *git push remotename remotebranch*

COPYING REMOTE REPOSITORY: CLONE

- git clone *repository*
- Clones the remote repository into the local one

STAGING CHANGES (LOCAL)

- git add *files*
- Adds the changes into the local staging area

SAVING CHANGES: COMMIT (LOCAL)

- **git commit "message"**
- Saves the changes in the staging area into the repository
- Creates a "snapshot" of the current state of one or more files
- A message describing the changes must be provided

HISTORY AND REVERT (LOCAL)

- **git log *files***
- returns a history of the file modifications
- **git revert *commit***
- removes one or more commits from the local files, changes must be committed after

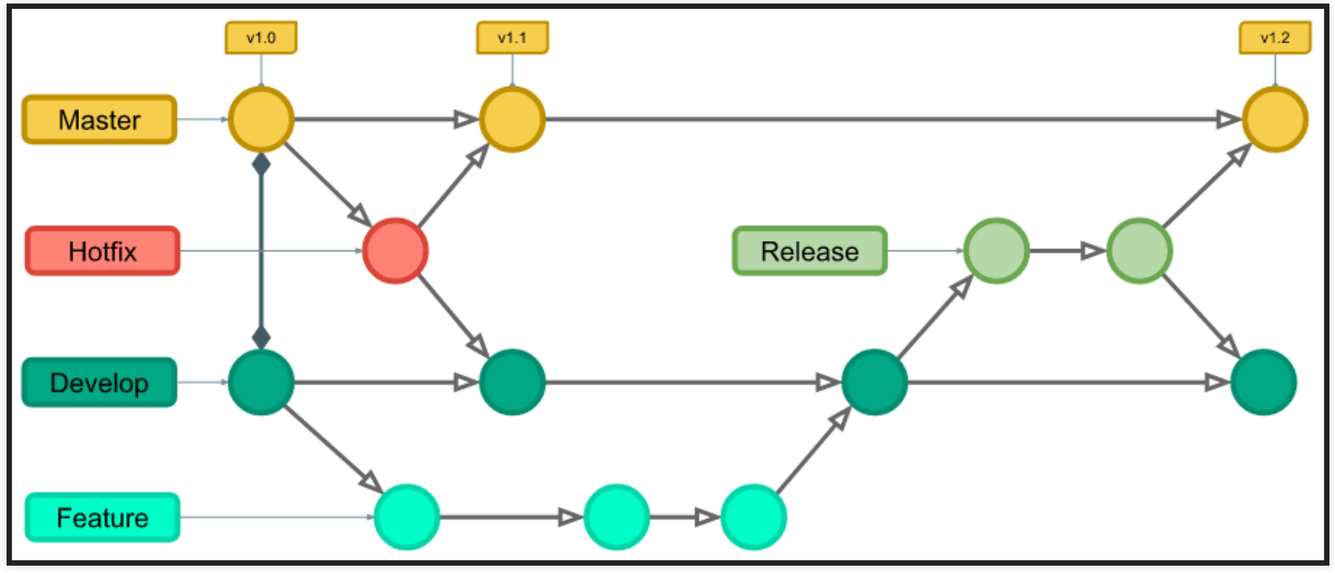
UPLOAD TO REMOTE REPOSITORY: PUSH

- git push
- Uploads the state of the local repository to the remote one

DOWNLOAD FROM REMOTE REPOSITORY: PULL

- **git pull**
- Fetch and merges the documents in the remote repository into the local one
- Merging files can generate conflicts, git will ask us to fix them and commit the changes

BRANCHING



OTHER (ADVANCED) STUFF

- tags
- partial reverts
- change history
- …

DOCS AS CODE

- Software is a small part of the documents a project must handle
- Still, version control and remote collaboration are needed for all the documents
- In the last years there is a big push of treating documents the same way as programming files
  - https://www.writethedocs.org/guide/docs-as-code/

ADVANTAGES

- Working in plain text files (rather than binary file formats like Word)
- Collaborating using version control such as git and GitHub
- Storing docs in the same repositories as the programming code itself
- Versioning docs through git tags/releases (rather than duplicating all the files to archive each release)
- Generate other formats or websites without modifying the document

JUST A LITTLE PROBLEM…

- The most common document formats: word, pdf… are binary files
- git (text based) doesn’t work with them

SOLUTIONS?

- Markup languages:
- Markup languages are ways of annotating an electronic document.
- Usually markup will either specify how something should be displayed or what something means.
  - html, xml, latex…

MARKUP LANGUAGES

- Documents are written in plain text, then a program convert them into the final document
- The same document can be used to generate files in other formats: latex, word, pdf or even slides
- Formating is done by the computer, output is always consistent
- Fast and light
- Can be used in version control systems

MARKUP LANGUAGES: ADVANCED FEATURES

- Automatic generation of documents
- Inline comments (not rendered in the final document)
- Split one the document into several. Ex: main document, chapters and bibliography
- Code executed and plots rendered *in* the document

LATEX

- Extensively used for technical papers
- Beautiful generated documents
- Very powerful…
- …and very heavy
- Setup and document customization are complex

LATEX: EXAMPLE

```
\documentclass{article}
\usepackage{graphicx}

\begin{document}

\title{Introduction to \LaTeX{}}
\author{Author's Name}

\maketitle

\begin{abstract}
The abstract text goes here.
\end{abstract}

\section{Introduction}
```

LATEX: EXAMPLE II

```
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}

\section*{Notes for My Paper}

Don't forget to include examples of topicalization.
They look like this:

{\small
\enumsentence{Topicalization from sentential subject:\\
\shortex{7}{a John$_i$ [a & kltukl & [el &
  {\bf l-}oltoir & er & ngii$_i$ & a Mary]]}
{ & {\bf R-}clear & {\sc comp} &
```

LATEX ALTERNATIVE: LYX

- WYSIWYG latex editor
- Documents are generated in .lyx, a subset of latex
- Can be used together with version control
- Provides, by default, templates for many of the biggest scientific journals

LYX: EXAMPLE

LyX: ~/Research/papers/ZFU/foo.lyx

Standard

Appendix: A Model of ZFU* + AC* + N + S

Let ZF^+ be ZF + "There is an inaccessible cardinal". To reflect more faithfully the restricted version of choice **AC*** that we have argued is most appropriate for ZFU*, let us formulate a correspondingly restricted notion of "accessible" choice for ZF^+. Let κ^* be the smallest inaccessible:

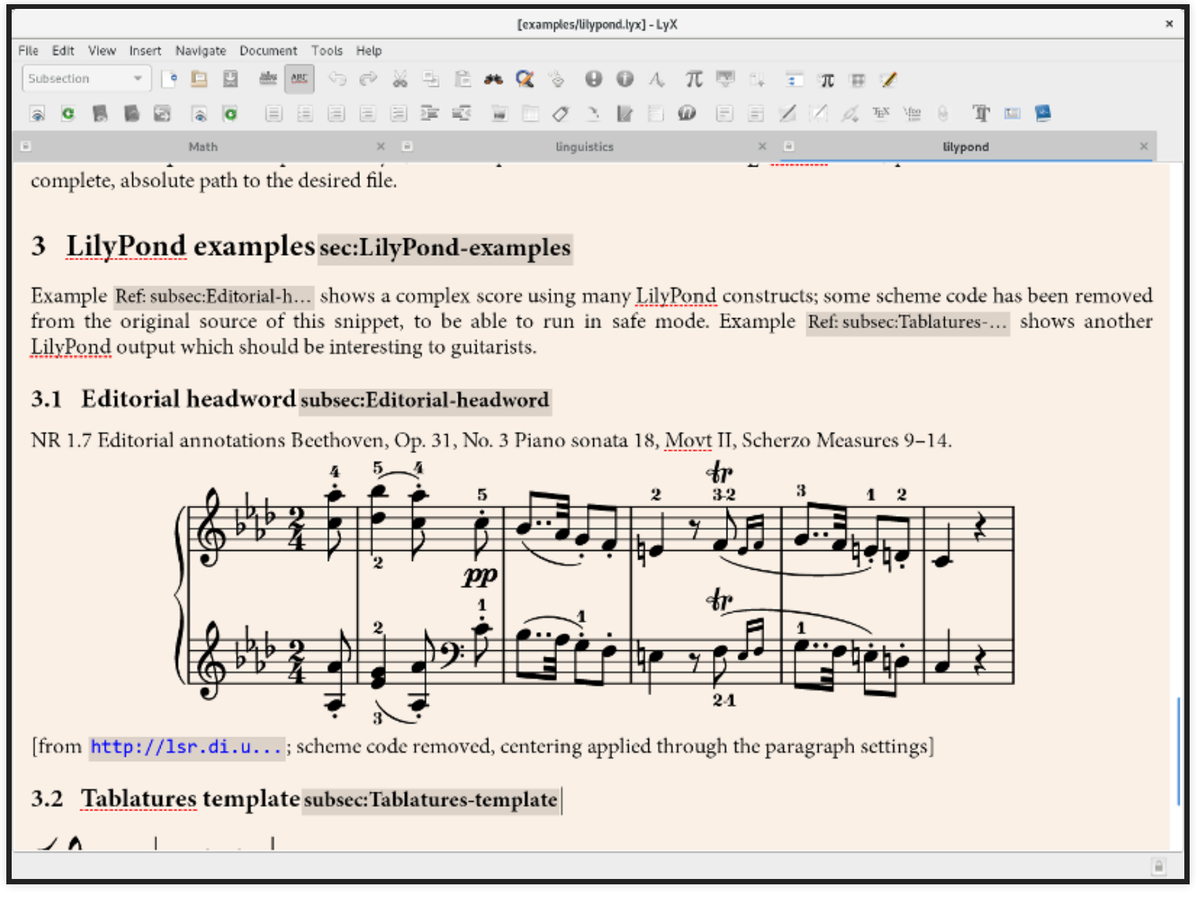
$$\mathbf{AC}_{<\kappa^*} \quad (A \neq \varnothing \wedge |A| < \kappa^* \wedge \forall x(x \in A \rightarrow \exists y\, y \in x)) \rightarrow \exists f \forall x(x \in A \rightarrow f(x) \in x)$$

Now for the model, defined in $ZF^+ + \mathbf{AC}_{<\kappa^*}$: Let $A^* = \{\langle \kappa^*, \alpha \rangle : \alpha < \kappa^*\}$. For $\alpha < \kappa^*$ and limit ordinals $\lambda \leq \kappa^*$, let:

$$\begin{aligned} U_0 &= A^* \\ U_{\alpha+1} &= U_\alpha \cup \{A \in \wp(U_\alpha) : |A| \leq \kappa^*\} \\ U_\lambda &= \bigcup_{\alpha < \lambda} U_\alpha \end{aligned}$$

Font: Default

LYX: EXAMPLE II



LIGHTWEIGHT MARKUP LANGUAGES

- Also called Plain Text Markup or humane markup language
- Provide a way of formating the document, while still being readable
- Widely used on websites and code documentation

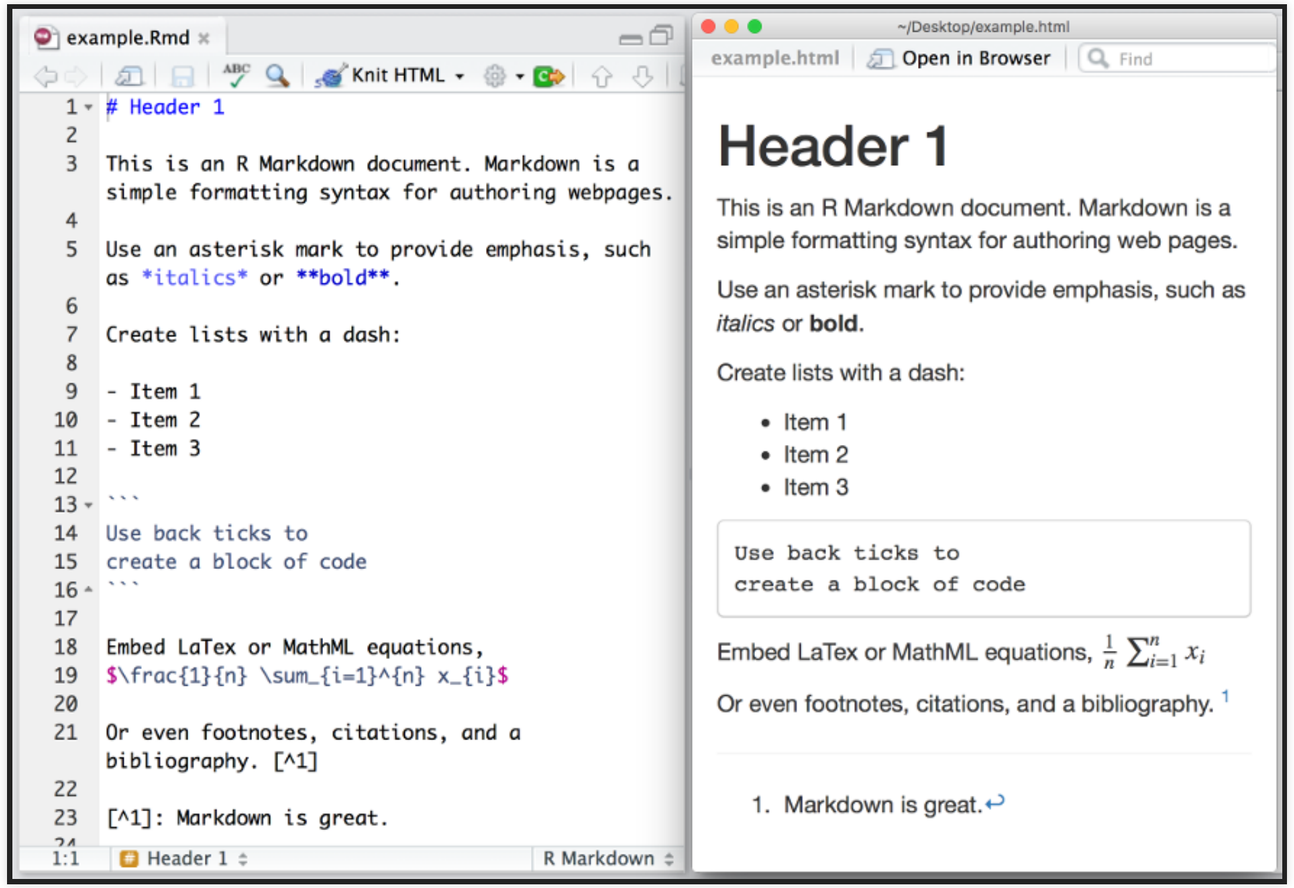
LML: CURRENT OPTIONS

- Markdown
- reStructuredText (rst)
- Asciidoc

MARKDOWN

- Created for minimal formating of web text
- used ***everywhere***: web, jupyter notebooks, r-markdown…
- There is no standard, currently exist many flavours of it (github, commonmark, pandoc)
- Originally not intended for documents, very limited
- Different flavors and tools try to overcome this limitation
  - (+ pandoc)

MARKDOWN: EXAMPLE




```
# Header 1

This is an R Markdown document. Markdown is a
simple formatting syntax for authoring webpages.

Use an asterisk mark to provide emphasis, such
as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

```
Use back ticks to
create a block of code
```

Embed LaTex or MathML equations,
$\frac{1}{n} \sum_{i=1}^{n} x_{i}$

Or even footnotes, citations, and a
bibliography. [^1]

[^1]: Markdown is great.
```

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark to provide emphasis, such as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

```
Use back ticks to
create a block of code
```

Embed LaTex or MathML equations, $\frac{1}{n} \sum_{i=1}^{n} x_i$

Or even footnotes, citations, and a bibliography. [1]

1. Markdown is great.

ASCIIDOC

- Developed for book creation.
- Limited number of users
- Standardized and extensible, great documentation
- Lack of resources makes that bugs or request take time to be fixed

RESTRUCTUREDTEXT

- Originally intended for python documentation
- medium sized but very tech-savvy community
- Syntax is a little different than the other two
- Very powerful and extensible

WHICH ONE TO USE?

- Notetaking:
  - Markdown
  - Asciidoc
  - reStructuredText
- Anything more serious:
  - reStructuredText
  - Latex/Lyx

RESOURCES

https://chocolatey.org

```
choco install git vscode pandoc
```

QUESTIONS?