Introduction & Setup

Session 1

Mickaël CANOUIL, Ph.D.[1*]

Saturday, the 7th of February, 2026

Affiliations

[1] Independent Contractor, Lille, France

ORCID

Mickaël CANOUIL, Ph.D.: 0000-0002-3396-4549

[*]Corresponding author: pro@mickael.canouil.dev

Table of Contents

1. Introduction

1.1. Session 1: Introduction & Setup

- Welcome & Overview
  - Workshop objectives and participant goals.
  - Quarto overview: purpose, strengths, and reproducibility benefits.
  - What Quarto is and how it works with Pandoc.
  - Brief look at the Quarto Guide.
- Installation & Environment Setup
  - Installing Quarto (installer, Homebrew, Chocolatey).
  - Verify installation with `quarto check`.
  - Quarto as a command-line interface.
- Quarto Projects
  - Project types: default, website, blog, book, manuscript.
  - Creating projects with `quarto create project`.
  - Writing with various editors (VS Code, RStudio, Jupyter, etc.).

1.2. Workshop Objectives

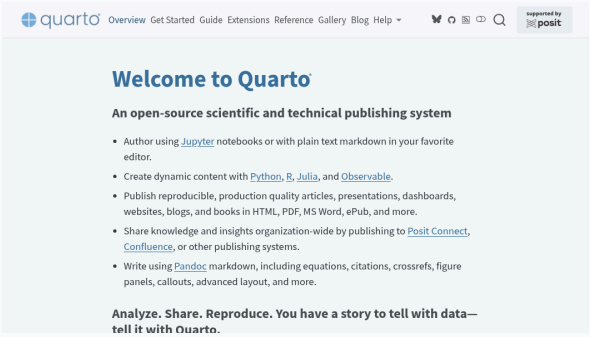
- Familiarise participants with the Quarto ecosystem.
- Empower attendees to create reproducible documents and interactive publications.
- Guide learners through embedding code, adjusting outputs, and publishing projects.
- Encourage hands-on practice with interactive sessions.

1.3. Learning Objectives

By the end of this session, participants will be able to:

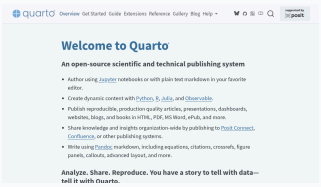
- Explain what Quarto is and how it relates to Pandoc.
- Install Quarto and verify the installation.
- Create and render different Quarto project types.
- Set up an IDE for Quarto development.

2. What is Quarto?



Quarto is an open-source scientific and technical publishing system built on Pandoc:

- Multi-language platform supporting Python, R, Julia, and Observable.js.
- Creates dynamic, reproducible documents from code and narrative text.
- Single-source authoring for articles, websites, books, presentations, and dashboards.



Quarto is an open-source scientific and technical publishing system built on Pandoc:

- Multi-language platform supporting Python, R, Julia, and Observable.js.
- Creates dynamic, reproducible documents from code and narrative text.
- Single-source authoring for articles, websites, books, presentations, and dashboards.

2.1. Key Strengths

Multi-format output

: Generate HTML, PDF, Word, ePub, and more from one source.

Advanced markdown

: Cross-references, equations, citations, callouts, and complex layouts.

Tool flexibility

: Works with VS Code, RStudio, JupyterLab, or any text editor.

Project system

: Render document collections with shared configurations.

Interactive elements

: Embed widgets, Shiny apps, and Observable JS for dynamic content.

2.2. Reproducibility Benefits

Embedded computation

: Code and results stay together, eliminating copy-paste errors.

Version control friendly

: Plain text format works seamlessly with Git.

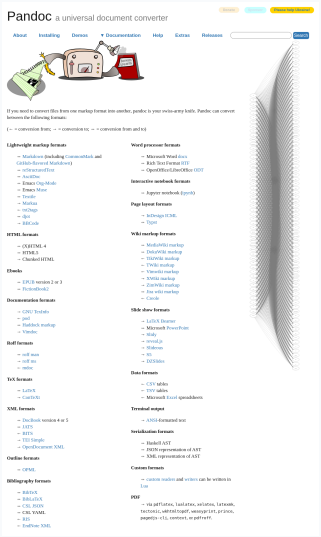
Transparent workflows

: Complete audit trail from data to final output.

Collaborative research

: Shared environments and dependencies ensure consistent results.

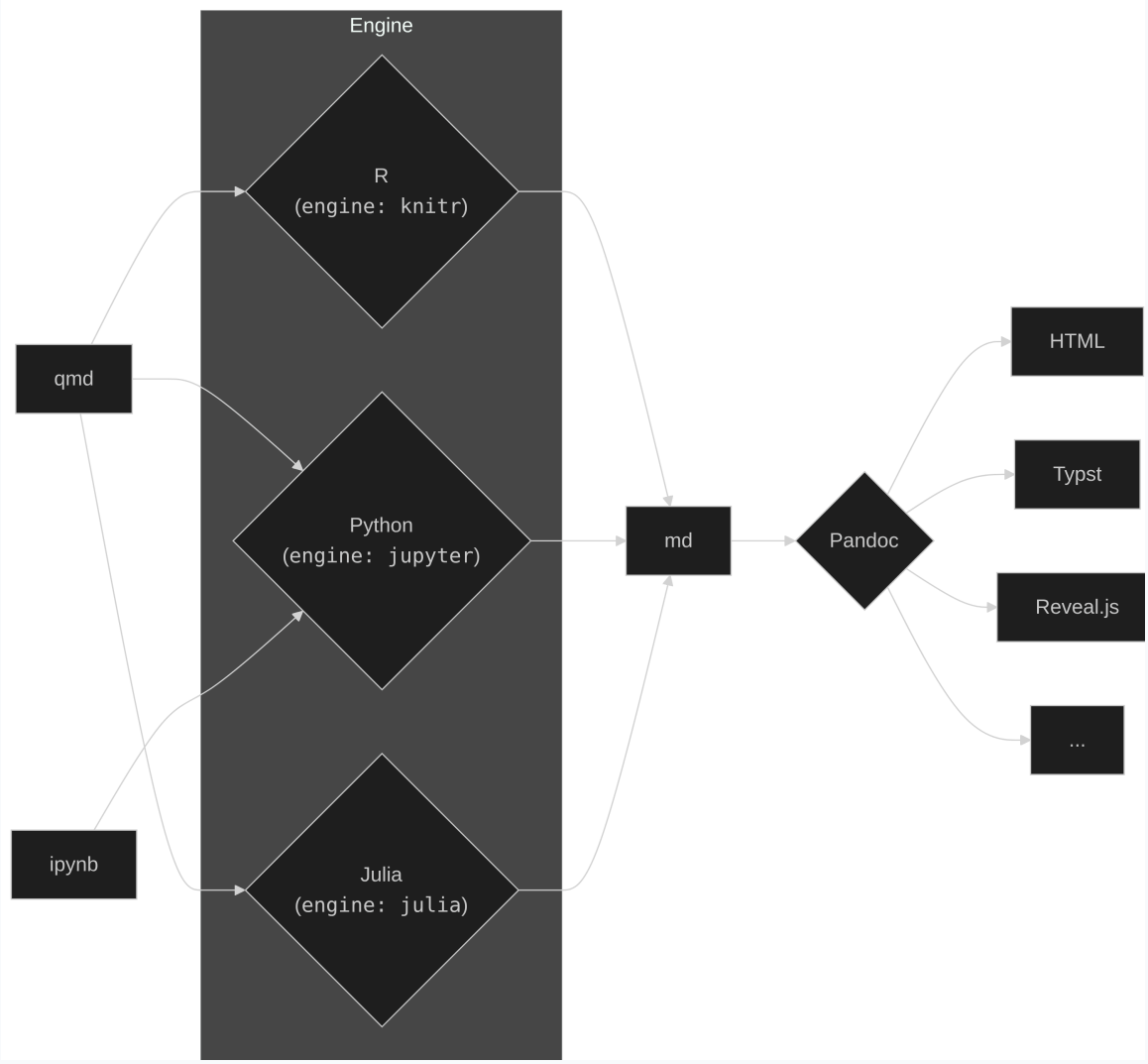
2.3. What is Pandoc?



Pandoc (pandoc.org):

- is a free software, released under the GPL by John MacFarlane.
- is a command line interface (CLI).
- allows to convert to and between multiple formats.
- understands Markdown syntax.
- understands LaTeX math and macros.
- handles citations and bibliographies.

2.4. How does Quarto work?



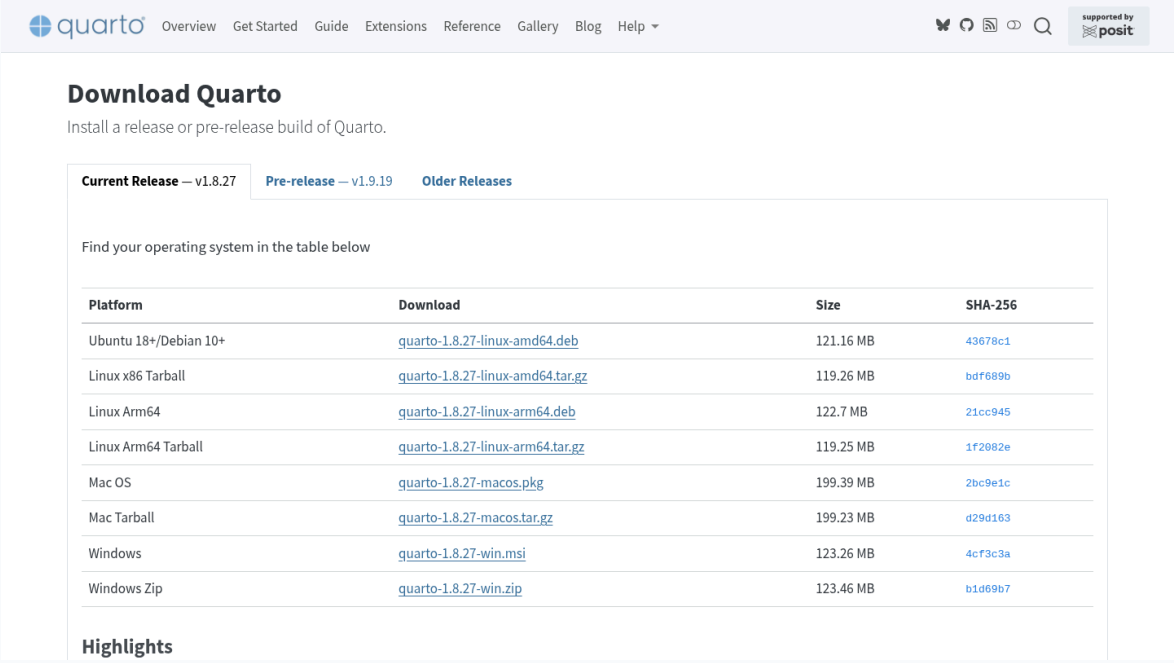
2.5. A Guide to Quarto Versions

Quarto follows a versioning scheme similar to Semantic Versioning with three release types: Pre-release, Release Candidate (RC), and Release.

> **Tip**
>
> Always use the latest stable release for production work. See the Quarto Changelog for the full release history.

3. Getting started

3.1. Downloading and installing Quarto



https://quarto.org/

Additionally, you can download and install Quarto using:

```bash
brew install quarto
```

```powershell
choco install quarto
```

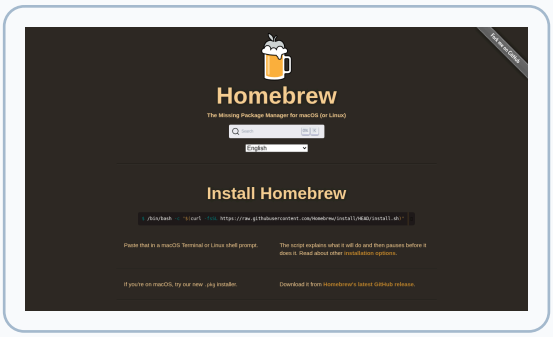


Figure 3.1: https://brew.sh/

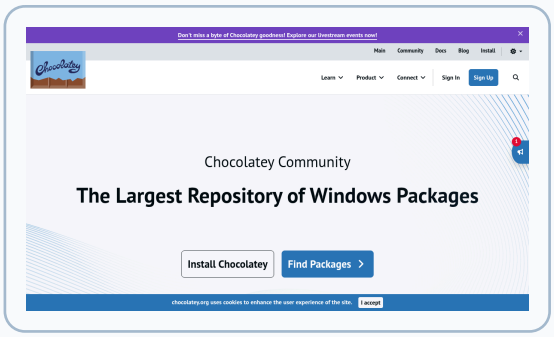


Figure 3.2: https://community.chocolatey.org/

3.2. Using GitHub Codespaces

Open in GitHub Codespaces

A pre-configured development environment is available via GitHub Codespaces with all required tools pre-installed: Quarto, R, Python, Julia, TinyTeX, and supporting utilities.

3.3. Quarto: a command line interface

```bash
quarto --help
```

```
Usage:   quarto
Version: 1.9.18

Description:

  Quarto CLI

Options:

  -h, --help     - Show this help.
  -V, --version  - Show the version number for this program.

Commands:

  render     [input] [args ... ]     - Render files or projects to various
document types.
  preview    [file] [args ... ]      - Render and preview a document or website
project.
  serve      [input]                 - Serve a Shiny interactive
document.
  create     [type] [commands ... ]  - Create a Quarto project or
extension
  use        <type> [target]         - Automate document or project setup
tasks.
  add        <extension>             - Add an extension to this folder or
project
  update     [target ... ]           - Updates an extension or global
dependency.
  remove     [target ... ]           - Removes an
extension.
  convert    <input>                 - Convert documents to alternate
representations.
  pandoc     [args ... ]             - Run the version of Pandoc embedded within
Quarto.
  typst      [args ... ]             - Run the version of Typst embedded within
Quarto.
  run        [script] [args ... ]    - Run a TypeScript, R, Python, or Lua
script.
  list       <type>                  - Lists an extension or global
dependency.
  install    [target ... ]           - Installs a global dependency (TinyTex or
Chromium).
  uninstall  [tool]                  - Removes an
```



```
extension.
  tools                          - Display the status of Quarto installed
dependencies
  publish    [provider] [path]   - Publish a document or project to a
provider.
  check      [target]            - Verify correct functioning of Quarto
installation.
  call                           - Access functions of Quarto subsystems such
as its rendering engines.
  help       [command]           - Show this help or the help of a sub-
command.
```

3.4. Checking Quarto is installed

BASH

```
quarto check
```

SH

```
Quarto 1.8.24
[✓] Checking environment information ...
      Quarto cache location: /home/vscode/.cache/quarto
[✓] Checking versions of quarto binary dependencies ...
      Pandoc version 3.6.3: OK
      Dart Sass version 1.87.0: OK
      Deno version 2.3.1: OK
      Typst version 0.13.0: OK
[✓] Checking versions of quarto dependencies......OK
[✓] Checking Quarto installation......OK
      Version: 1.8.24
      Path: /opt/quarto/bin                                              (1)

[✓] Checking tools.....................OK
      TinyTeX: (external install)
      Chromium: (not installed)                                          (2)

[✓] Checking LaTeX.....................OK
      Using: TinyTex
      Path: /home/vscode/.TinyTeX/bin/x86_64-linux
      Version: 2025                                                      (3)

[✓] Checking Chrome Headless.....................OK
      Using: Chrome found on system
      Path: /usr/bin/google-chrome
      Source: PATH                                                       (4)
```



```
[✓] Checking basic markdown render....OK

[✓] Checking Python 3 installation....OK
      Version: 3.13.7
      Path: /usr/local/python/current/bin/python3
      Jupyter: 5.8.1
      Kernels: julia-1.11, python3                                    (5)

[✓] Checking Jupyter engine render....OK

[✓] Checking R installation...........OK
      Version: 4.5.1
      Path: /opt/R/4.5.1/lib/R
      LibPaths:
        - /home/vscode/R/x86_64-pc-linux-gnu-library/4.5
        - /opt/R/4.5.1/lib/R/library
      knitr: 1.50
      rmarkdown: 2.29                                                 (6)

[✓] Checking Knitr engine render......OK
```

1. Quarto version and dependencies.
2. Tools installed by Quarto (e.g., `quarto install`).
3. LaTeX installation detected by Quarto.
4. Chromium-based browser installation detected by Quarto.
5. Python installation and computing dependencies detected by Quarto.
6. R installation and computing dependencies detected by Quarto.

4. Quarto Projects

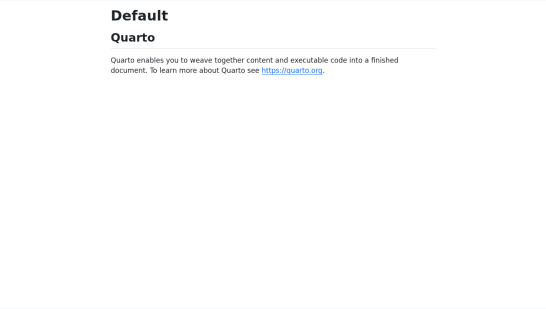
4.1. Creating a new Quarto project

```bash
quarto create project
```

```
? Type
❯ default
  website
  blog
  manuscript
  book
  confluence
```

4.2. Quarto Projects: `default`

```bash
quarto create project default
Default
```




```
_demo/Default
├── Default.qmd
`-- _quarto.yml

0 directories, 2 files
```

- `_quarto.yml` : project configuration file.

```yaml
project:
  title: "Default"
```

- `Default.qmd` : default Quarto document.

```yaml
---
title: "Default"
---

## Quarto
```


Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

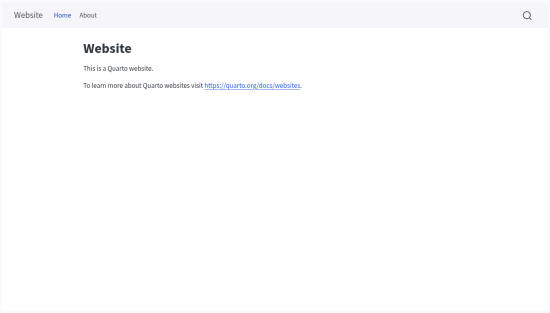
4.3. Quarto Projects: `website`

Bash

```
quarto create project website
Website
```

```
_demo/Website
├── _quarto.yml
├── about.qmd
├── index.qmd
`-- styles.css

0 directories, 4 files
```



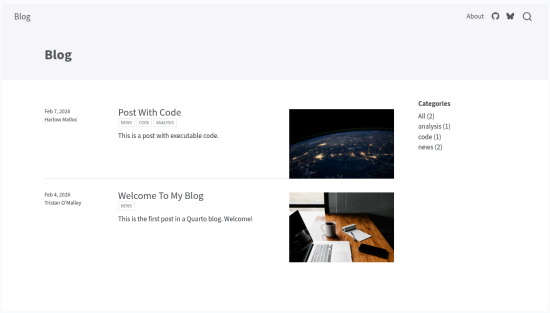
4.4. Quarto Projects: `blog`

Bash

```
quarto create project blog Blog
```

```
_demo/Blog
├── _quarto.yml
├── about.qmd
├── index.qmd
├── posts
│   ├── _metadata.yml
│   ├── post-with-code
│   │   ├── image.jpg
│   │   `-- index.qmd
│   `-- welcome
│       ├── index.qmd
│       `-- thumbnail.jpg
├── profile.jpg
`-- styles.css

3 directories, 10 files
```

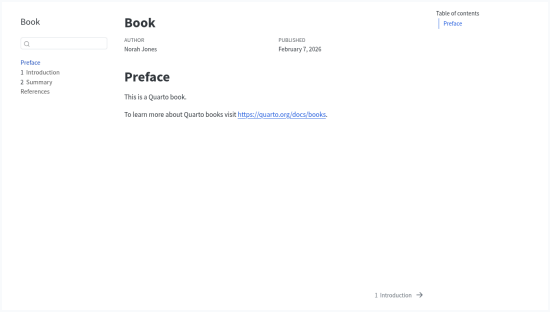


4.5. Quarto Projects: `book`

```bash
quarto create project book Book
```

```
_demo/Book
├── _quarto.yml
├── cover.png
├── index.qmd
├── intro.qmd
├── references.bib
├── references.qmd
`-- summary.qmd

0 directories, 7 files
```

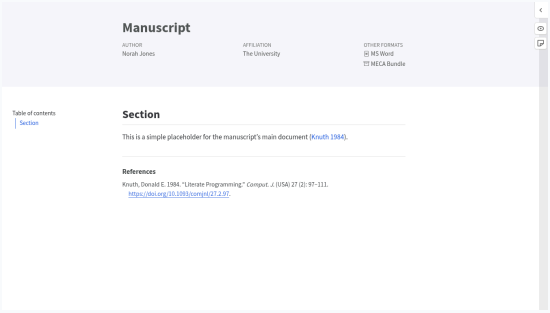


4.6. Quarto Projects: `manuscript`

```bash
quarto create project manuscript
Manuscript
```

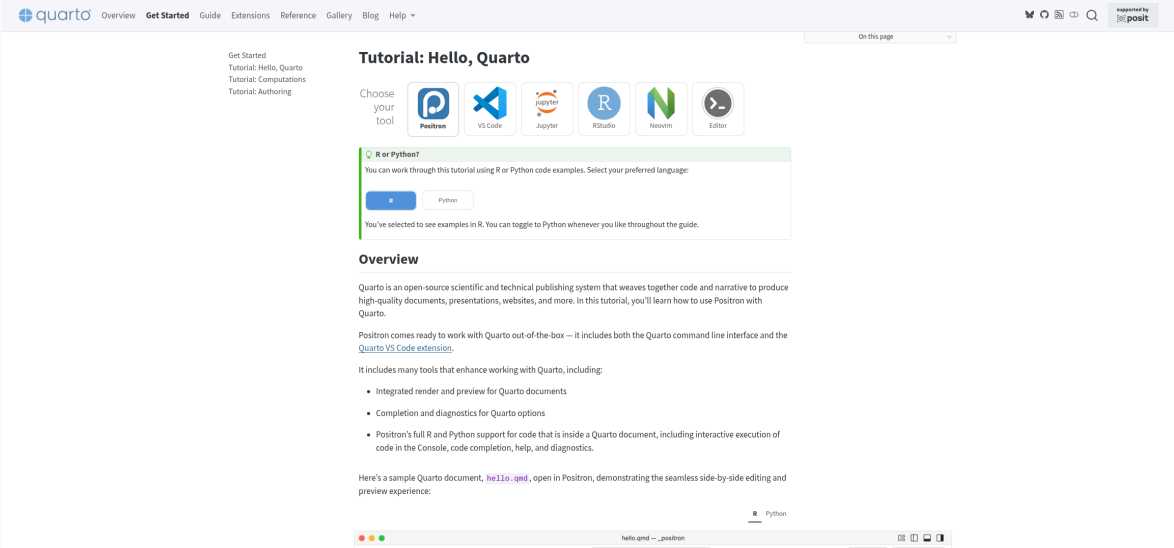
```
_demo/Manuscript
├── _quarto.yml
├── index.qmd
`-- references.bib

0 directories, 3 files
```

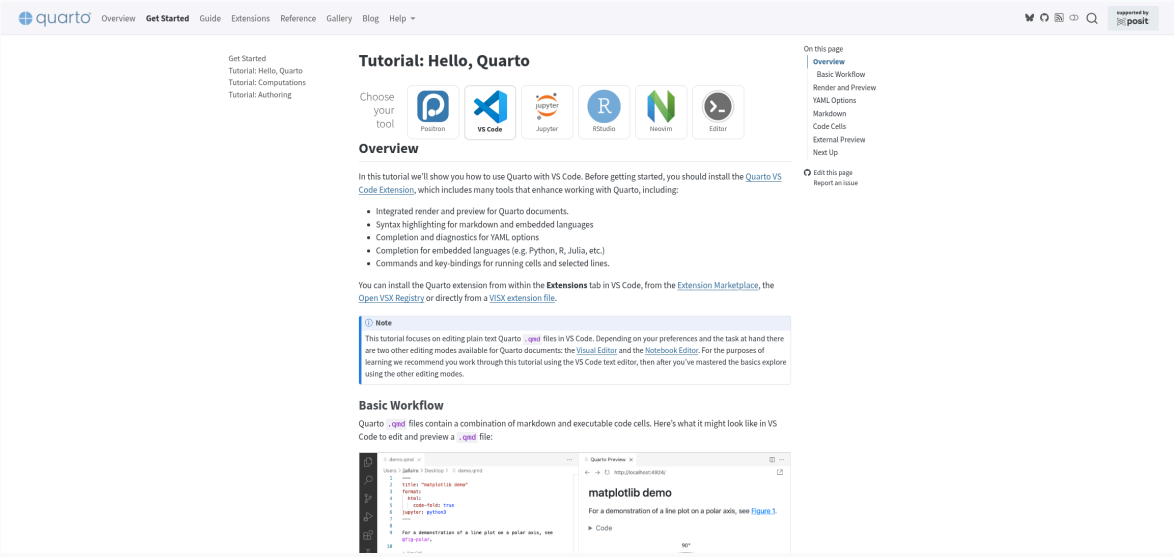


4.7. Writing with your favourite editor

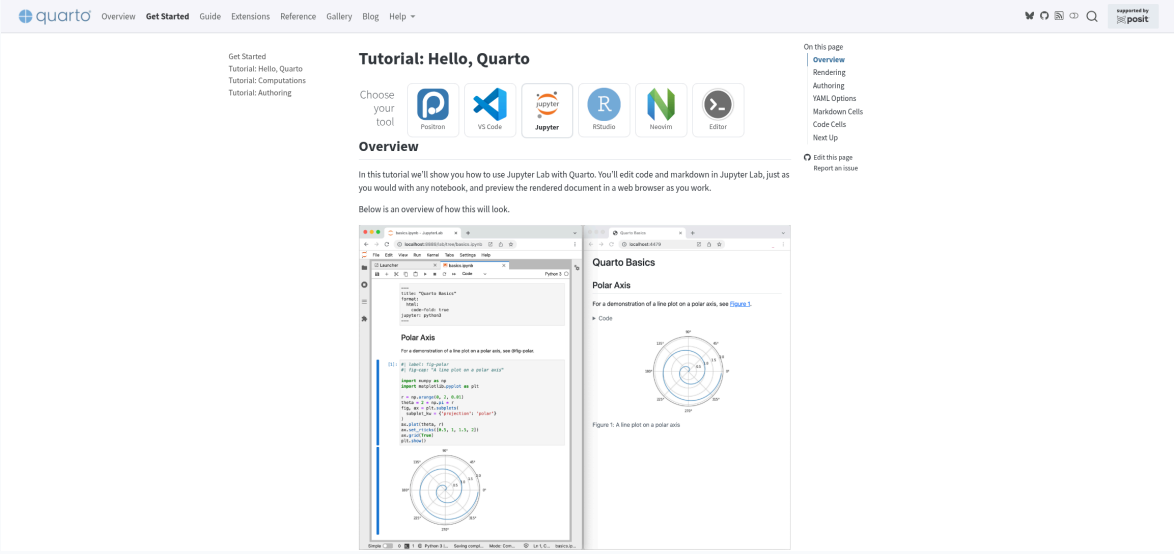
4.7.1. POSITRON



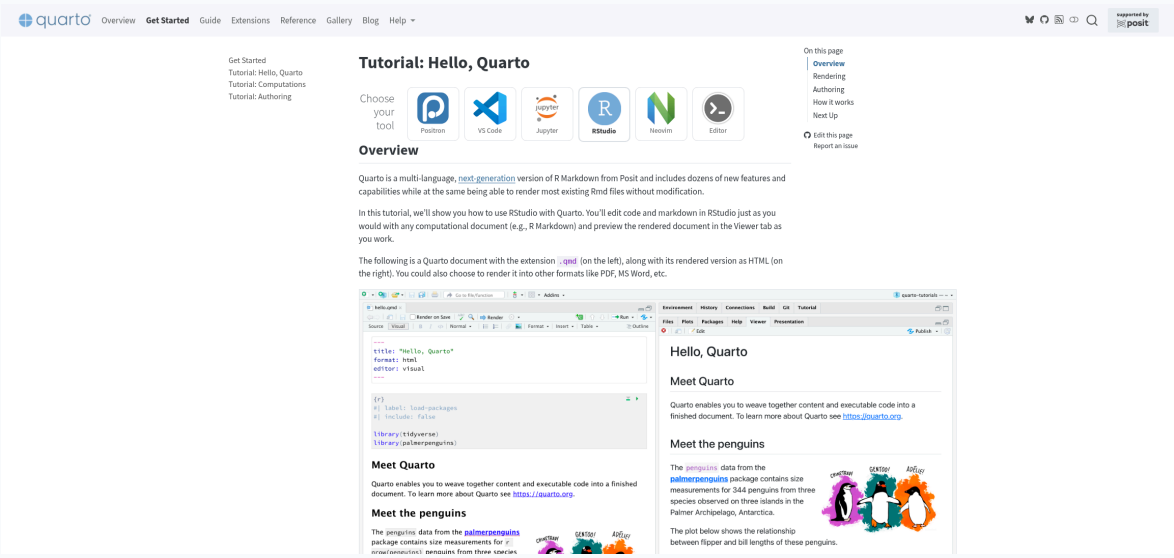
4.7.2. VSCODE



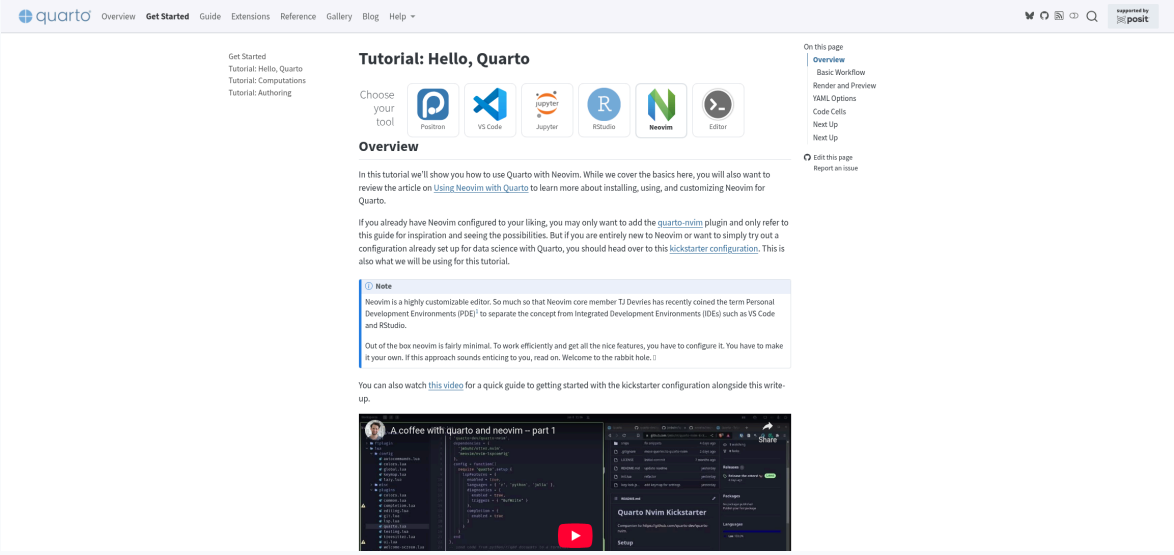
4.7.3. JUPYTER



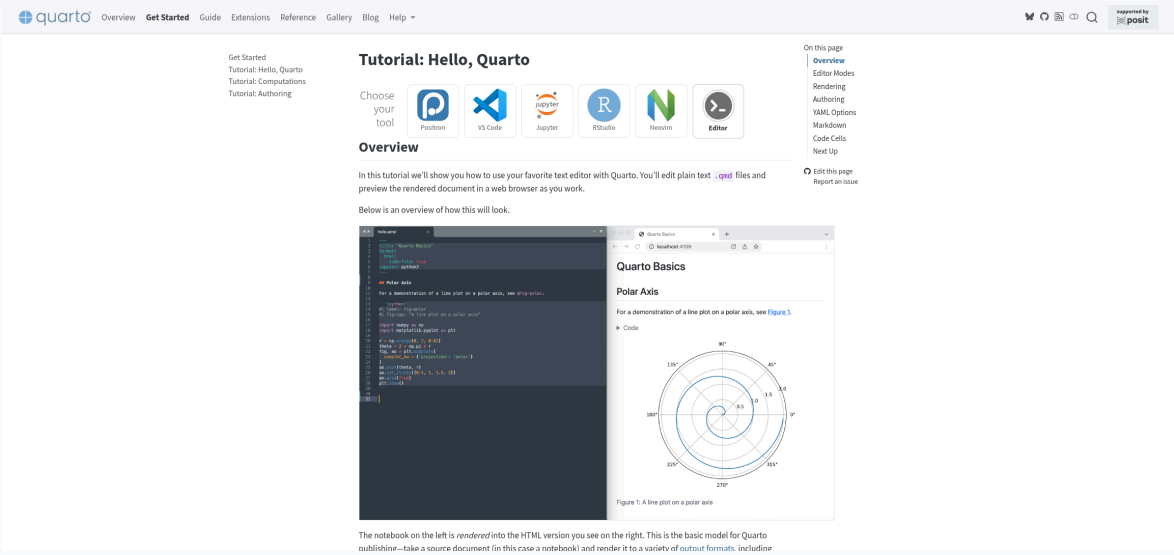
4.7.4. RSTUDIO



4.7.5. NEOVIM



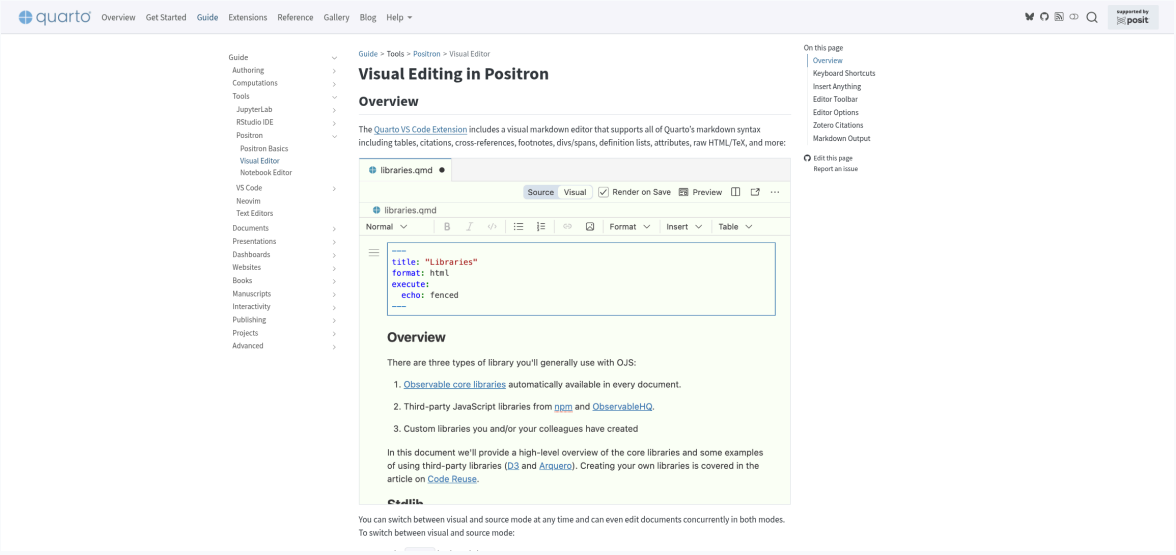
4.7.6. TEXT-EDITOR



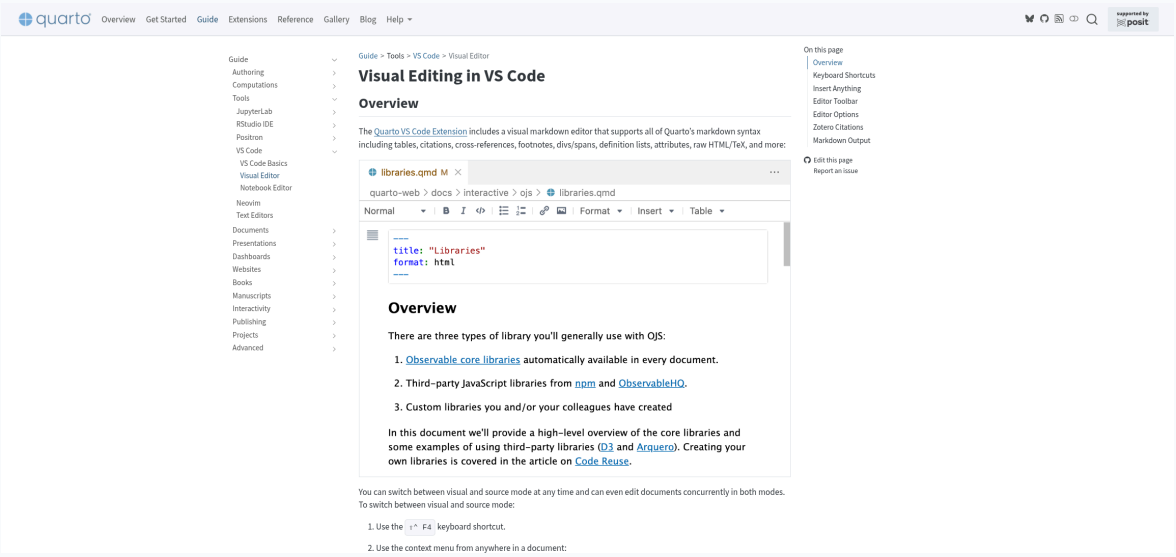
https://quarto.org/

4.8. Writing using the Visual Editor

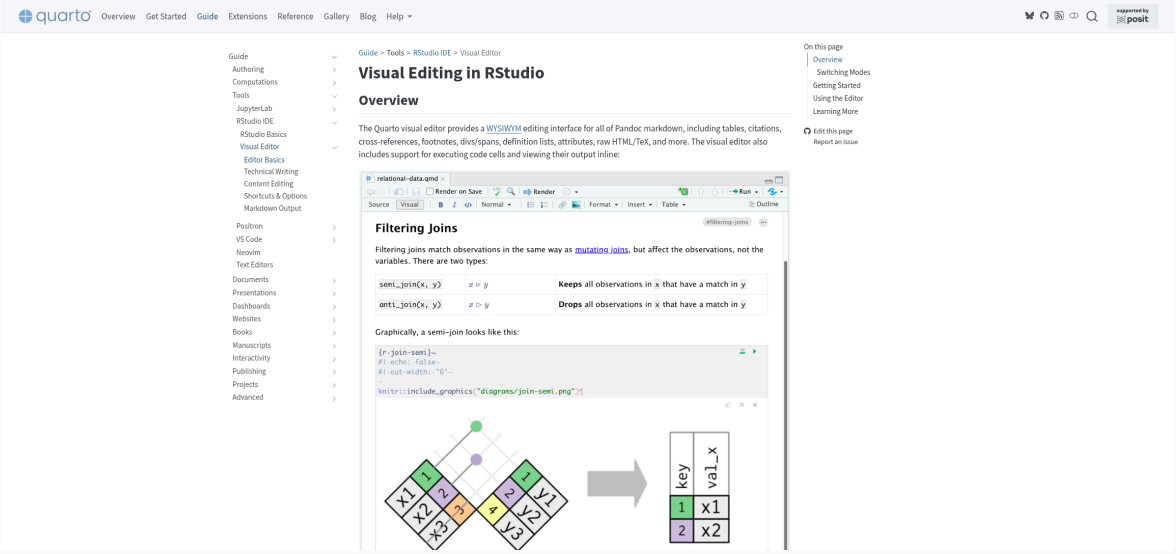
4.9. POSITRON



4.10. VSCODE



4.11. RSTUDIO



https://quarto.org/

5. Hands-On Exercise: Creating Your First Quarto Project

5.1. Exercise Overview

Objective: Create a personal Quarto document that demonstrates basic features, project setup skills, and IDE integration

Example Code: Exercises

```bash
tar -xzf "01-exercises.tar.gz" -C "01-introduction-setup"
```

IDE Configuration

Before starting, ensure your development environment is ready:

- VS Code: Install the Quarto extension (`Ctrl/Cmd + Shift + X`, search "Quarto").
- Positron: Built-in Quarto support is included (no additional extensions needed).
- Open the Command Palette (`Ctrl/Cmd + Shift + P`) and type "Quarto" to verify available commands.

5.2. Part 1: Project Creation

1. Choose Your Adventure: Create a project using the CLI or your IDE.
2. Explore the project structure in the IDE's file explorer panel.

5.3. Part 2: IDE Features Exploration

Discover your IDE's Quarto capabilities:

- File recognition: Notice `.qmd` files have distinctive icons and syntax highlighting.
- Document outline: Open a `.qmd` file and check the Outline/Structure panel (shows headings and sections).
- Quarto assist panel: Look for a Quarto-specific panel that provides contextual help.
- Command palette power: `Ctrl/Cmd + Shift + P` ⟶ type "Quarto" to see all available commands.
- Integrated features: Visual editor mode, preview panels, terminal integration.
- Smart editing: Try auto-completion in YAML headers, fenced divs, and code blocks.

5.4. Part 3: Document Customisation

Modify the main document (`index.qmd`) to include:

1. Update the YAML header with your information:


```yaml
---
title: "My Quarto Journey"
author: "Your Name"
date: today
editor: visual  # Enables rich editing features
---
```

2. Add content to the document body.

5.5. Part 4: Rendering

Try different rendering approaches:

1. Command Line (classic method):

```bash
quarto render
```

2. IDE Integration (modern workflow):
   - Command Palette (`Ctrl/Cmd + Shift + P`) - search for "Quarto Preview" and/or "Quarto Render" commands.
   - Use keyboard shortcut if you configured one
   - Notice IDE-specific feedback and error handling

5.6. Success Criteria

☑ You've successfully completed the exercise if you can:

- Configure your IDE for optimal Quarto development.
- Create a new Quarto project using the CLI or your IDE.
- Navigate project files efficiently in your IDE.
- Utilise IDE features like syntax highlighting and command palette.
- Modify documents with IDE assistance (auto-completion, outline view).
- Render documents using both CLI and IDE methods.