

Latex & Markdown

Tópicos de Informática para Automação

Mário Antunes

December 8, 2025

Exercises

This guide will take you from setting up your environment to write using latex and markdown to building a web service that automatically converts Markdown files to PDF. No prior coding experience is required—just follow the steps and copy the code blocks exactly.

Step 0: System Setup

First, we need to install the “Compiler” tools.

1. Open your **Terminal**.
2. Copy and paste the following command (you will need your password):

```
sudo apt update
sudo apt install wget texlive-luatex texlive-latex-recommended \
texlive-latex-extra texlive-fonts-recommended texlive-fonts-extra \
fonts-lmodern fonts-noto fonts-noto-cjk fonts-noto-color-emoji pandoc -y
```

3. Create a folder named ex12 to place the exercises.
4. Run this in your terminal inside the ex12 folder first:

```
wget -O profile.png "https://placeholder.co/400x400/png"
wget -O graph.png "https://placeholder.co/600x400/png"
```

Note: `texlive-full` is the complete version of the `texlive` tool, but it is large (several GBs). It contains every package you might need so you don't encounter missing errors later. In case of error consider installing it.

Step 1: Write a CV

We will create a CV using both methods. Create a folder named `ex12` on your Desktop to keep things organized.

Option A: The LaTeX Way (Professional)

LaTeX has a famous class called `moderncv`. It looks great out of the box.

1. Create a file named `my_cv.tex` inside your `ex12` folder.
2. Paste this code into it:

```
\documentclass[11pt,a4paper,sans]{moderncv}
\moderncvstyle{classic} % Options: casual, classic, banking, oldstyle, fancy
\moderncvcolor{blue}      % Options: blue, orange, green, red, purple, grey
\usepackage{graphicx} % Required for images

% Personal Data
\name{John}{Doe}
\title{Computer Science Student}
\address{123 University Ave}{Aveiro, Portugal}
\phone[mobile]{+351 912 345 678}
```

```

\email{john.doe@student.pt}

% Profile picture (height: 64pt, border: 0.4pt)
\photo[64pt][0.4pt]{profile.png}

\begin{document}
\makecvtitle

\section{Education}
\cventry{2023--2026}{BSc in Computer Science}{University of Aveiro}{Aveiro}{}{Focus on Algorithms and Web Development.}

\section{Skills}
\cvitem{Languages}{Python, C, Java, SQL}
\cvitem{Tools}{Git, Docker, LaTeX, Linux}

\section{Experience}
\cventry{2022}{Summer Intern}{Tech Corp}{Lisbon}{}{Helped fix bugs in the frontend application.}

\end{document}

```

3. **Compile it:** Run this command in your terminal inside the folder:

```
pdflatex my_cv.tex
```

4. Check the folder. You now have `my_cv.pdf`.

Option B: The Markdown Way (Fast)

1. Create a file named `my_cv.md`.
2. Paste this code:

```

# John Doe
![Profile Picture](https://placehold.co/150x150/png)

**Computer Science Student**
*Aveiro, Portugal | +351 912 345 678 | john.doe@student.pt*

---

## Education
**BSc in Computer Science** | University of Aveiro | *2023--2026*
* Focus on Algorithms and Web Development.

## Skills
* **Languages:** Python, C, Java, SQL
* **Tools:** Git, Docker, LaTeX, Linux

## Experience
**Summer Intern** | Tech Corp | *2022*
* Helped fix bugs in the frontend application.

3. Compile it:
pandoc my_cv.md -o my_cv_markdown.pdf

```

Step 2: Write a Project Report

Scenario: You are writing a report on “The Efficiency of Coffee on Student Coding Speed”.

Option A: The LaTeX Way

1. Create a file named `report.tex`.
2. Paste this code:

```
\documentclass[12pt]{report}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{hyperref}
\usepackage{booktabs}

\title{The Efficiency of Coffee on
Student Coding Speed}
\author{Group 12: John Doe \& Jane Smith}
\date{\today}

\begin{document}

\maketitle
\tableofcontents

\chapter{Introduction}
This study analyzes whether caffeine intake
correlates with lines of code written per hour.

\begin{figure}[!htb]
    \centering
    \includegraphics[width=0.8\textwidth]{graph.png}
    \caption{Projected Coding Speed vs. Caffeine}
    \label{fig:coffee_graph}
\end{figure}

\chapter{Methodology}
We observed 10 students over 4 hours.

\begin{table}[!htb]
    \centering
    \caption{Participant Group Settings}
    \label{tab:groups}
    % Notice the @{} to remove side spacing
    \begin{tabular}{@{}llrr@{}}
        \toprule
        Group & Drink Type & Dosage (mg) \\
        & Participants \\
        \midrule
        A & Water & 0 & 5 \\
        B & Espresso & 120 & 5 \\
        C & Energy Drink & 160 & 5 \\
        \bottomrule
    \end{tabular}
\end{table}

\chapter{Results}
Group B wrote code 20\% faster but introduced
10\% more bugs.

\chapter{Conclusion}
Coffee increases speed but decreases accuracy.

\end{document}
```

3. **Compile it:**

```
pdflatex report.tex
```

```
pdflatex report.tex
```

(Note: We run it twice so LaTeX can generate the Table of Contents numbers correctly).

Option B: The Markdown Way

1. Create a file named `report.md`.
2. Paste this code:

```
---
```

title: The Efficiency of Coffee on Student Coding Speed
author: Group 12 - John Doe & Jane Smith
date: 2023-12-05

```
--
```

Introduction

This study analyzes whether caffeine intake correlates with lines of code written per hour.

{ width=80% }

Methodology

We observed 10 students over 4 hours.

Table: Participant Group Settings

Group	Drink Type	Dosage (mg)	Participants
A	Water	0	5
B	Espresso	120	5
C	Energy Drink	160	5

Results

Group B wrote code 20% faster but introduced 10% more bugs.

Conclusion

Coffee increases speed but decreases accuracy.

3. **Compile it** (Using the `--number-sections` flag to make it look like a report):

```
pandoc report.md -o report_markdown.pdf --number-sections
```

Step 3: Write Presentation Slides

Now you need to present the findings from Step 2.

Option A: The LaTeX Way (Beamer)

1. Create a file named `slides.tex`.
2. Paste this code:

```
\documentclass{beamer}  
\usepackage{Madrid}  
\usepackage{booktabs} % Required for the table  
  
\title{Coffee vs. Code}  
\author{Group 12}  
\date{\today}  
  
\begin{document}  
  
\frame{\titlepage}
```

```

\begin{frame}
\frametitle{Introduction}
\begin{itemize}
    \item Does caffeine help us code?
    \item We tested 10 students.
\end{itemize}
\begin{center}
    \includegraphics[width=0.6\textwidth]{graph.png}
\end{center}
\end{frame}

\begin{frame}
\frametitle{Methodology}
\begin{table}
    \centering
    \begin{tabular}{@{}llrr@{}}
        \toprule
        Group & Drink & Dosage & N \\
        \midrule
        A & Water & 0mg & 5 \\
        B & Espresso & 120mg & 5 \\
        \bottomrule
    \end{tabular}
\end{table}
\end{frame}

\begin{frame}
\frametitle{Results}
\alert{Result:} Speed increased by 20\%.
\end{frame}

\end{document}

```

3. Compile it:

`pdflatex slides.tex`

Option B: The Markdown Way (Beamer via Pandoc)

1. Create a file named `slides.md`.
2. Paste this code (Use --- to separate slides):

```

---
title: Coffee vs. Code
author: Group 12
date: today
---

# Introduction
* Does caffeine help us code?
* We tested 10 students.



---

# Methodology

| Group | Drink | Dosage | N |
| :--- | :--- | ---: | ---: |
| A | Water | 0mg | 5 |
| B | Espresso | 120mg | 5 |

```

```
# Results
**Result:** Speed increased by 20%.
3. Compile it (We tell Pandoc to output type beamer):
pandoc slides.md -t beamer -o slides_markdown.pdf
```

Step 4: The Automated Converter (Docker)

Now, we will build a real web application. A user uploads a Markdown file on a website, the server converts it to PDF using Pandoc, and sends it back.

4.1 Folder Structure

Create a new folder named converter_app. Inside it, create two folders: backend and frontend. Your structure must look exactly like this:

```
converter_app/
├── compose.yml
└── backend/
    ├── Dockerfile
    ├── main.py
    └── requirements.txt
└── frontend/
    ├── Dockerfile
    ├── index.html
    └── nginx.conf
```

4.2 The Files

1. converter_app/compose.yml (This tells Docker to run both the Python API and the Web Server).

```
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    volumes:
      - ./backend:/app

  frontend:
    build: ./frontend
    ports:
      - "8080:80"
```

2. converter_app/backend/Dockerfile (This installs Python + Pandoc + LaTeX inside the container).

```
FROM python:3.12-trixie
```

```
# Install system dependencies (Pandoc and LaTeX)
RUN apt update && apt install -y \
    texlive-luatex texlive-latex-recommended \
    texlive-latex-extra texlive-fonts-recommended \
    texlive-fonts-extra fonts-lmodern fonts-noto \
    fonts-noto-cjk fonts-noto-color-emoji pandoc \
    && rm -rf /var/lib/apt/lists/*
```

```
WORKDIR /app
```

```
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
```

```
COPY . .
```

```
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

```
3. converter_app/backend/requirements.txt
```

```
fastapi
uvicorn
python-multipart
```

```
4. converter_app/backend/main.py (The Python logic: Receive file -> Run Pandoc -> Return PDF).
```

```
import subprocess
import os
from fastapi import FastAPI, UploadFile, File
from fastapi.responses import FileResponse
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Allow the frontend to talk to this backend
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.post("/convert")
async def convert_md_to_pdf(file: UploadFile = File(...)):
    # 1. Save the uploaded markdown file
    input_filename = "input.md"
    output_filename = "output.pdf"

    with open(input_filename, "wb") as f:
        f.write(await file.read())

    # 2. Run Pandoc command inside the container
    # pandoc input.md -o output.pdf
    try:
        subprocess.run(
            ["pandoc", input_filename, "-o", output_filename],
            check=True
        )
    except subprocess.CalledProcessError:
        return {"error": "Conversion failed"}

    # 3. Return the generated PDF
    return FileResponse(output_filename, filename="converted.pdf", media_type='application/pdf')
```

```
5. converter_app/frontend/Dockerfile
```

```
FROM nginx:alpine
COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY index.html /usr/share/nginx/html/index.html
```

```
6. converter_app/frontend/nginx.conf
```

```
server {
    listen 80;
    server_name localhost;
```

```

        location / {
            root    /usr/share/nginx/html;
            index   index.html index.htm;
        }
    }
}

```

7. converter_app/frontend/index.html (The User Interface).

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Markdown to PDF Converter</title>
    <style>
        body { font-family: sans-serif; display:
            flex; justify-content: center; align-items: center;
            height: 100vh; background: #f0f0f0; }
        .card { background: white; padding: 2rem;
            border-radius: 10px; box-shadow: 0 4px 6px rgba(0,0,0,0.1);
            text-align: center; }
        button { background: #007bff; color: white; border: none;
            padding: 10px 20px; border-radius: 5px; cursor: pointer;
            margin-top: 10px; }
        button:hover { background: #0056b3; }
    </style>
</head>
<body>
    <div class="card">
        <h1>MD to PDF Converter</h1>
        <p>Select a Markdown file to convert.</p>
        <input type="file" id="fileInput" accept=".md">
        <br><br>
        <button onclick="uploadAndConvert()">Convert & Download</button>
        <p id="status"></p>
    </div>

    <script>
        async function uploadAndConvert() {
            const fileInput = document.getElementById('fileInput');
            const status = document.getElementById('status');

            if(fileInput.files.length === 0) {
                alert("Please select a file!");
                return;
            }

            const formData = new FormData();
            formData.append("file", fileInput.files[0]);
            status.innerText = "Converting ... Please wait.";

            try {
                // Send file to Backend (FastAPI)
                const response = await fetch('http://localhost:8000/convert', {
                    method: 'POST',
                    body: formData
                });

                if (!response.ok) throw new Error("Conversion failed");

                // Create a blob from the response (the PDF)
                const blob = await response.blob();
            }
        }
    </script>

```

```

        const url = window.URL.createObjectURL(blob);

        // Force download
        const a = document.createElement('a');
        a.href = url;
        a.download = "converted_document.pdf";
        document.body.appendChild(a);
        a.click();
        a.remove();
        status.innerText = "Done!";
    } catch (error) {
        console.error(error);
        status.innerText = "Error converting file.";
    }
}
</script>
</body>
</html>

```

4.3 Running the App

1. Open your terminal inside the converter_app folder.

2. Run the following command:

```
sudo docker compose up --build
```

(This will take a few minutes to download the Python image and install LaTeX inside it. Be patient).

3. Once it stops moving and says “Application startup complete”:

- Open your browser (Firefox/Chrome).
- Go to: <http://localhost:8080>

4. Upload the `my_cv.md` or `report.md` you created earlier.

5. Click **Convert**. The browser should download a PDF version of your Markdown file!