Docling Testing

Using Git on Forgejo Server

The Forgejo Server

Create project

http://192.168.1.210:3000/stephen/137_Docing-Test

this creates an empty repository

On remote site MacBook

Open terminal on 01_Projects in the terminal

```
stephenlohning@Scotty 01_Projects % git clone
http://192.168.1.210:3000/stephen/137_Docing-Test
Cloning into '137_Docing-Test'...
warning: You appear to have cloned an empty repository.
```

This creates a directory called "137_Docing-Test" the same as the repository. There only one file in the directory

```
stephenlohning@Scotty 137_Docing-Test % ls -la
total 0
drwxr-xr-x    3 stephenlohning  staff    96 May 29 19:42 .
drwxr-xr-x@ 158 stephenlohning  staff  5056 May 29 19:42 ..
drwxr-xr-x    9 stephenlohning  staff   288 May 29 19:42 .git
stephenlohning@Scotty 137_Docing-Test % git remote -v
origin  http://192.168.1.210:3000/stephen/137_Docing-Test (fetch)
origin  http://192.168.1.210:3000/stephen/137_Docing-Test (push)
```

```
stephenlohning@Scotty 137_Docing-Test % git branch -M main
stephenlohning@Scotty 137_Docing-Test % touch .gitignore
stephenlohning@Scotty 137_Docing-Test % git config --global push.followTags
true
stephenlohning@Scotty 137_Docing-Test %
```

Create structure you want, template

```
stephenlohning@Scotty 137_Docing-Test % tree
.
```

```
├── Readme.md
└── doc
    ├── Notes.md
    ├── Notes.pdf
    └── images

3 directories, 3 files
```

Check uv version and upgrade

- This I did in VS Code terminal

```
stephenlohning@Scotty 137_Docing-Test % uv --version
uv 0.5.20 (1c17662b3 2025-01-15)
stephenlohning@Scotty 137_Docing-Test % uv self update
info: Checking for updates...
success: Upgraded uv from v0.5.20 to v0.11.17! https://github.com/astral-
sh/uv/releases/tag/0.11.17
stephenlohning@Scotty 137_Docing-Test %
```

setting up uv

```
stephenlohning@Scotty 137_Docing-Test % uv init
Initialized project `137-docing-test`
stephenlohning@Scotty 137_Docing-Test %
```

check what version of python are avainable

```
tephenlohning@Scotty 137_Docing-Test % uv python list
cpython-3.15.0b1-macos-aarch64-none                 <download available>
cpython-3.15.0b1+freethreaded-macos-aarch64-none    <download available>
cpython-3.14.5-macos-aarch64-none
/opt/homebrew/bin/python3.14 -> ../Cellar/python@3.14/3.14.5/bin/python3.14
cpython-3.14.5-macos-aarch64-none
/opt/homebrew/bin/python3 -> ../Cellar/python@3.14/3.14.5/bin/python3
cpython-3.14.5-macos-aarch64-none
/Users/stephenlohning/.pyenv/shims/python3.14
cpython-3.14.5-macos-aarch64-none
/Users/stephenlohning/.pyenv/shims/python3
cpython-3.14.5-macos-aarch64-none                   <download available>
cpython-3.14.5+freethreaded-macos-aarch64-none      <download available>
cpython-3.13.13-macos-aarch64-none                  <download available>
cpython-3.13.13+freethreaded-macos-aarch64-none     <download available>
cpython-3.13.11-macos-aarch64-none
/opt/homebrew/bin/python3.1
```

```
more here
```

create virtual environment

- uv venv --python 3.14

This creates .venv directory, a main.py and a pyproject.toml

```
stephenlohning@Scotty 137_Docing-Test % uv venv --python 3.14
Using CPython 3.14.2 interpreter at:
/Users/stephenlohning/.pyenv/versions/3.14.2/bin/python3.14
Creating virtual environment at: .venv
Activate with: source .venv/bin/activate
stephenlohning@Scotty 137_Docing-Test % ls -la
total 448
drwxr-xr-x   11 stephenlohning  staff     352 May 29 20:30 .
drwxr-xr-x@ 158 stephenlohning  staff    5056 May 29 19:42 ..
drwxr-xr-x   12 stephenlohning  staff     384 May 29 20:13 .git
-rw-r--r--    1 stephenlohning  staff       0 May 29 19:48 .gitignore
-rw-r--r--    1 stephenlohning  staff       5 May 29 20:28 .python-version
drwxr-xr-x    7 stephenlohning  staff     224 May 29 20:30 .venv
-rw-r--r--    1 stephenlohning  staff    1989 May 29 20:12 Readme.md
-rw-r--r--    1 stephenlohning  staff  212930 May 29 20:12 Readme.pdf
drwxr-xr-x    5 stephenlohning  staff     160 May 29 20:06 doc
-rw-r--r--    1 stephenlohning  staff      93 May 29 20:21 main.py
-rw-r--r--    1 stephenlohning  staff     161 May 29 20:21 pyproject.toml
```

pyproject.toml

```
stephenlohning@Scotty 137_Docing-Test % cat pyproject.toml
[project]
name = "137-docing-test"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.13"
dependencies = []
stephenlohning@Scotty 137_Docing-Test %
```

uv add docling

The command uv add docling does two distinct things simultaneously: it executes an installation via uv, and it targets a specific, popular open-source AI package called docling.

Running this command executes the following actions:

1. What uv does behind the scenes

Because you are using uv's modern project management tool, this single command handles the entire dependency lifecycle:

- Resolves Dependencies: It looks up docling on PyPI, checks all the sub-packages it requires, and resolves any version conflicts in milliseconds.
- Updates your pyproject.toml: It automatically writes docling into your project's configuration file under the dependencies list.
- Updates your uv.lock file: It locks down the exact cryptographic versions of docling and its sub-dependencies to ensure reproducibility across different machines.
- Installs into .venv: It downloads the wheels and installs them directly into your project's local virtual environment (creating the environment first if it doesn't already exist).

Example

```
stephenlohning@Scotty 137_Docing-Test % uv add docling
Resolved 122 packages in 2.09s
      Built pylatexenc==2.10
      Built antlr4-python3-runtime==4.9.3
Prepared 99 packages in 4.59s
Installed 99 packages in 551ms
 + accelerate==1.13.0
 + annotated-doc==0.0.4
 + annotated-types==0.7.0
 + antlr4-python3-runtime==4.9.3
 + anyio==4.13.0
 + attrs==26.1.0
 + beautifulsoup4==4.14.3
 + certifi==2026.5.20
 + charset-normalizer==3.4.7
 + click==8.4.1
 + colorlog==6.10.1
 + defusedxml==0.7.1
 + dill==0.4.1
 + docling==2.96.0
 + docling-core==2.77.1
 + docling-ibm-models==3.13.2
 + docling-parse==6.2.0
 + docling-slim==2.96.0
 + et-xmlfile==2.0.0
 + faker==40.19.1
 + filelock==3.29.0
 + filetype==1.2.0
 + fsspec==2026.4.0
 + h11==0.16.0
 + hf-xet==1.5.0
 + httpcore==1.0.9
 + httpx==0.28.1
```

```
+ huggingface-hub==1.17.0
+ idna==3.17
+ jinja2==3.1.6
+ jsonlines==4.0.0
+ jsonref==1.1.0
+ jsonschema==4.26.0
+ jsonschema-specifications==2025.9.1
+ latex2mathml==3.81.0
+ lxml==6.1.1
+ markdown-it-py==4.2.0
+ marko==2.2.3
+ markupsafe==3.0.3
+ mdurl==0.1.2
+ mpire==2.10.2
+ mpmath==1.3.0
+ multiprocess==0.70.19
+ networkx==3.6.1
+ numpy==2.4.6
+ omegaconf==2.3.0
+ opencv-python==4.13.0.92
+ openpyxl==3.1.5
+ packaging==26.2
+ pandas==3.0.3
+ pillow==12.2.0
+ pluggy==1.6.0
+ polyfactory==3.3.0
+ psutil==7.2.2
+ pyclipper==1.4.0
+ pydantic==2.13.4
+ pydantic-core==2.46.4
+ pydantic-settings==2.14.1
+ pygments==2.20.0
+ pylatexenc==2.10
+ pypdfium2==5.8.0
+ python-dateutil==2.9.0.post0
+ python-docx==1.2.0
+ python-dotenv==1.2.2
+ python-pptx==1.0.2
+ pyyaml==6.0.3
+ rapidocr==3.8.1
+ referencing==0.37.0
+ regex==2026.5.9
+ requests==2.34.2
+ rich==15.0.0
+ rpds-py==2026.5.1
+ rtree==1.4.1
+ safetensors==0.7.0
+ scipy==1.17.1
+ semchunk==3.2.5
+ setuptools==81.0.0
+ shapely==2.1.2
+ shellingham==1.5.4
+ six==1.17.0
+ soupsieve==2.8.4
```

```
 + sympy==1.14.0
 + tabulate==0.10.0
 + tokenizers==0.22.2
 + torch==2.12.0
 + torchvision==0.27.0
 + tqdm==4.67.3
 + transformers==5.9.0
 + tree-sitter==0.25.2
 + tree-sitter-c==0.24.2
 + tree-sitter-javascript==0.25.0
 + tree-sitter-python==0.25.0
 + tree-sitter-typescript==0.23.2
 + typer==0.21.2
 + typing-extensions==4.15.0
 + typing-inspection==0.4.2
 + urllib3==2.7.0
 + websockets==16.0
 + xlsxwriter==3.2.9
stephenlohning@Scotty 137_Docing-Test %
```

Updated pyproject.toml

```
cat pyproject.toml
[project]
name = "137-docing-test"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.13"
dependencies = [
    "docling>=2.96.0",
]
stephenlohning@Scotty 137_Docing-Test %
```

Update the main.py

```
# main.py
from docling.document_converter import DocumentConverter

def main():
    # Can be a local path (e.g., "docs/invoice.pdf") or a URL
   # source = "https://arxiv.org/pdf/2408.09869"
    source = ./SpanishTextBook.pdf

    print("Initializing Docling pipeline...")
    converter = DocumentConverter()

    print("Processing document locally...")
    result = converter.convert(source)
```

```
    # Export cleanly to structured Markdown
    markdown_output = result.document.export_to_markdown()
    print("\n--- Parsing Successful! Sample Output: ---\n")
    print(markdown_output[:500])

if __name__ == "__main__":
    main()
```

Runing via uv

- uv run main.py

We had multiple errors with Docling

1. The Python Version Compatibility Issue

The Error

Initially, installing Docling failed due to deep dependency conflicts rooted in the Python environment version. The system was running Python 3.14 (an experimental/pre-release version).

```
Plaintext
Configuring project 137_Docing-Test...
  × No solution found when resolving dependencies:
  ╰─▶ Because docling v2.14.0 depends on numpy >=1.26.4 and <2.0.0...
      and no versions of numpy match >=1.26.4,<2.0.0 because python_version
>= "3.14"
```

The Root Cause

Core scientific computing libraries utilized by Docling—specifically numpy (restricted to <2.0.0 by underlying models) and scipy—did not yet have pre-compiled binary wheels available for Python 3.14. Building them from source failed due to strict C-extension requirements that were incompatible with the unreleased Python runtime.

The Fix We utilized the uv package manager to explicitly pin and enforce a stable, production-grade Python runtime environment without altering your global system configuration:

```
uv init --python 3.12
```

This cleanly isolated the project to Python 3.12, instantly unlocking access to fully compiled, highly optimized binary wheels for numpy, scipy, torch, and onnxruntime.

2. The Metal Performance Shaders (MPS) float64 Crash

The Error Once the environment was stable, running the document converter triggered a hard runtime crash during the Stage layout phase when processing the first page:

```
TypeError: Cannot convert a MPS Tensor to float64 dtype as the MPS
framework doesn't support float64. Please use float32 instead.
Stage layout failed for run 1: Cannot convert a MPS Tensor to float64 dtype
as the MPS framework doesn't support float64.
```

The Root Cause

Docling uses IBM's rt_detr_v2 (Real-Time DEtection TRansformer) vision model for document layout and element segmentation.

- By default, PyTorch detected the Apple Silicon M3 GPU hardware and automatically routed tensors to the MPS (Metal Performance Shaders) backend.
- Deep inside the Hugging Face transformers layout code, the model initialized a positional embedding layer using high-precision 64-bit floating-point numbers:

ω= torch.arange(pos_dim,dtype=torch.float64) ---------- pos_dim

- Because Apple's Metal shading language natively operates on 16-bit (half) and 32-bit (float) precisions to maximize unified memory bandwidth, the MPS framework physically lacks hardware support for float64 operations, causing the runtime execution to immediately fault.

The Fix

Standard environment variables (like device="cpu" passing configurations or CUDA_VISIBLE_DEVICES="") failed because the model's internal source code bypassed them to poll the hardware directly.

We resolved this by implementing a pre-emptive monkeypatch at the absolute top of main.py before any heavy machine learning frameworks could execute their initialization loops:

```
import torch

# Intercept and mock PyTorch's hardware discovery backend completely
torch.backends.mps.is_available = lambda: False
torch.backends.mps.is_built = lambda: False

# Force standard CPU allocation fallback
torch.set_default_device("cpu")
```

The Result

By blinding PyTorch to the presence of the local Metal architecture, the layout engine cleanly and safely fell back to the standard CPU processing path. This bypassed the precision limitations completely and

achieved a flawless, structurally complete 9-page Markdown compilation in 36.44 seconds.