```toml
# FILE: pyproject.toml

[tool.poetry]
name = "bwga-nexus-backend"
version = "0.1.0"
description = "The AI and Data Engine for the BWGA Nexus 7.0 Platform."
authors = ["Brayden Walls <brayden@bwglobaladvis.info>"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.11"  # We are explicitly telling Render to use Python 3.11

# Core Web Framework
fastapi = "^0.110.0"
uvicorn = {extras = ["standard"], version = "^0.27.1"}

# Data Science & AI Libraries
pandas = "^2.2.1"
scikit-learn = "^1.4.1.post1"
spacy = "^3.7.4"

# Database Connector
psycopg2-binary = "^2.9.9"

# API Requests (for pulling data)
httpx = "^0.27.0"

# Add any other libraries you need here

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

content_copydownload

Use code with caution.Toml

Step 2: Create the `Dockerfile`

Now we will create the `Dockerfile` from my previous answer. This is the instruction manual that tells Render *how* to use your `pyproject.toml` file to build the application inside a container.

Action: In the root directory of your project, create a file named `Dockerfile` and paste this code into it.

Generated dockerfile

```
# FILE: Dockerfile


# Use an official Python runtime as a parent image
FROM python:3.11-slim-bullseye


# Set the working directory inside the container
WORKDIR /app


# Set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1


# Install system dependencies
RUN apt-get update && apt-get install -y build-essential libpq-dev && rm -rf
/var/lib/apt/lists/*


# Install poetry
RUN pip install poetry


# Copy only the dependency files to leverage Docker layer caching
COPY poetry.lock pyproject.toml /app/


# Install project dependencies into the virtual environment
RUN poetry config virtualenvs.create false && poetry install --no-root --no-dev


# Copy your application's source code into the container
COPY . .


# Expose the port that FastAPI will run on
EXPOSE 8000


# The command to run your application
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

content_copydownload

Use code with caution.Dockerfile

(Note: I have made a small correction to the `poetry install` command to work more reliably inside Docker.)

Step 3: Tell Render How to Build Your Project

Now, you need to go into your Render dashboard for this service and change one setting.

Action:

1. Go to the "Settings" page for your `nexus-7.0` service on Render.
2. Find the **"Build & Deploy"** section.
3. Change the **"Build Command"** from `pip install -r requirements.txt` to be **completely empty**. Delete the text.
4. Ensure the **"Start Command"** is also empty.
5. Scroll down to the **"Environment"** section and ensure it says **"Docker."**

Why this works: When Render detects a `Dockerfile` in your repository, it ignores the build and start commands. The `Dockerfile` itself becomes the *only* set of instructions it needs. It will automatically follow the steps you've laid out: install Poetry, use `pyproject.toml` to install your libraries, and then run the `CMD` instruction to start your server.

Your Final Action Plan

1. **Add the two files (`pyproject.toml` and `Dockerfile`)** to the root of your `nexus-7.0` GitHub repository.
2. **Commit and push** these changes to GitHub.

Generated bash

```bash
git add .
git commit -m "feat: Add pyproject.toml and Dockerfile for Render deployment"
git push
```