Foundation Project Guidelines

Overview: Building an end-to-end application on a self-managed server is an excellent way to prepare for the programme. The foundation project is a simplified version of the weekly projects, and involves building, containerizing, and deploying an MNIST digit classifier.

Project Brief: Build, containerize, and deploy a simple digit recognizer trained on the MNIST dataset.

1. Get Comfortable with the Basics

a. Python & PyTorch: Learn Python programming, how to set up virtual environments, and the basics of PyTorch. You'll use PyTorch to load the MNIST dataset, build a simple model, train it, and save the weights. For detailed tutorials, visit:

PyTorch Tutorials

b. Streamlit for Web Apps: Streamlit enables rapid development of interactive web apps in Python. It will be used to create the front-end where users can draw digits or upload images to get predictions. Check out the documentation at:

Streamlit Docs

2. Train Your MNIST Model Locally

Develop a Python script or Jupyter Notebook that performs the following: loading the MNIST dataset (using torchvision), building and training a simple model using PyTorch, and saving the trained model weights. You can experiment with the model training in a Jupyter Notebook for immediate feedback and visualization.

3. Build the Streamlit Front-End

Create a web interface using Streamlit. This interface should include a canvas or image upload area for digit input, display the model's prediction and its confidence score, and allow users to provide the correct label for feedback.

4. Logging with PostgreSQL

Store each prediction made by the model, along with the user-provided true label and a timestamp, in a PostgreSQL database. This step involves setting up the database, creating the necessary tables, and writing Python scripts to handle the logging. For more information on PostgreSQL, visit:

PostgreSQL Docs

5. Containerization with Docker

Containerize your application by setting up Docker containers for the PyTorch model/service, the Streamlit web app, and the PostgreSQL database. Write Dockerfiles for each service and use Docker Compose to orchestrate them. Further details can be found at:

Docker Docs

6. Deployment

Deploy your containerized application on a self-managed server (e.g., a Hetzner basic instance, DigitalOcean, or AWS Lightsail). Ensure Docker and Docker Compose are installed, open the necessary ports, and run your containers. Access the application through a public IP or domain.

7. Managing Project Files and Tools

A key part of this project is organizing your code and files efficiently. Here?s a recommended structure and tool usage:

Project Folder Structure:

```
mnist_digit_classifier/
??? docs/              # Documentation and guidelines
??? notebooks/         # Jupyter notebooks for experimentation
??? src/               # Source code organized as follows:
?   ??? model/         # PyTorch model training and inference scripts
?   ??? webapp/        # Streamlit application code
?   ??? database/      # Database interaction and logging scripts
```

```
??? docker/             # Dockerfiles and docker-compose.yml
??? requirements.txt    # Python dependencies
??? README.md           # Project overview and instructions
```

Tools to Use:

Visual Studio Code: Use this IDE to develop and manage your main project files. Write and organize your Python scripts, Docker configurations, and documentation here. Visual Studio Code supports working with multiple files and integrated version control via Git.

Jupyter Notebook: Use notebooks for prototyping and experimenting, especially when training the MNIST model. They provide interactive code execution and visualization, which is ideal for testing different model parameters.

GitHub: Use Git for version control and push your project to GitHub to keep track of changes, collaborate if needed, and back up your work. Integrate with Visual Studio Code through Git extensions for seamless version control.

8. Add Project to GitHub

Once your project structure is set up and your code is written, commit your changes and push your project to a GitHub repository. Include all project files (code, Dockerfiles, notebooks, and documentation) in the repository, and update your README.md with instructions on how to run and deploy your application.

Summary

This guide outlines the process of building an end-to-end MNIST digit recognizer. It covers training a PyTorch model, creating an interactive Streamlit web app, logging predictions in PostgreSQL, containerizing the application with Docker, and finally deploying it on a self-managed server. Furthermore, it provides guidance on how to manage your project files using Visual Studio Code for main development, Jupyter Notebook for prototyping, and GitHub for version control. Follow these detailed steps to build and deploy your project successfully.