**Step 1: Installing Docker Desktop and Setting Up**

1. **Download Docker Desktop:**
   * Go to Docker Desktop.
   * Choose the version for your operating system (Windows, macOS).
   * Download and run the installer.
2. **Install Docker Desktop:**
   * Follow the installation instructions based on your OS:
     + **Windows**: The installer will prompt you to install WSL2 (Windows Subsystem for Linux 2) if it's not already installed.
     + **macOS**: Drag the Docker app to the Applications folder.
3. **Set Up Docker Desktop:**
   * Sign in to Docker Hub or create a new account (optional).
   * Docker Desktop should be running now. Check the system tray or taskbar to confirm.
4. **Verify Installation:**
   * Open a terminal/command prompt and run:
   * *docker –version*

**Step 2: Installing Docker and Dev Container Extensions in VS Code**

1. **Install Docker Extension in VS Code:**
   * Open VS Code.
   * Press Ctrl + Shift + X to open the Extensions view.
   * Search for **Docker** and click **Install** on the official Docker extension by Microsoft.
2. **Install Dev Containers Extension in VS Code:**
   * In the Extensions view, search for **Dev Containers** by Microsoft and click **Install**.
3. **Set Up GitHub Authentication (Optional for GitHub projects):**
   * If you're working with GitHub repositories, install the **GitHub Pull Requests and Issues** extension.
   * Sign in to GitHub to seamlessly manage pull requests and issues from within VS Code.

**Step 3: Creating the Dockerfile**

1. **Start with the Python Base Image:**
   * Add this line to use an official Python runtime:

*FROM python:3.12*

1. **Set the Working Directory:**
   * This sets /app as the directory inside the container where all commands will run:

*WORKDIR /app*

1. **Install System Dependencies:**
   * Install dependencies like git, vim, nano, etc. (Add more if needed):

*RUN apt-get update && apt-get install -y \*

*libgl1-mesa-glx \*

*wget \*

*git \*

*vim \*

*nano \*

*&& apt-get clean \*

*&& rm -rf /var/lib/apt/lists/\**

1. **Copy the requirements.txt File:**

COPY *docker/requirements.txt .* (it’s from the docker folder, period at end important)

1. **Install Python Dependencies:**
   * Run pip to install dependencies from requirements.txt:

*RUN pip3 install --no-cache-dir -r requirements.txt*

1. **Copy the Current Directory Contents into the Container:**
   * Copy everything from the current directory on your local machine into the /app directory in the container:

*COPY . .*

1. **Expose Jupyter Port (Optional):**
   * Expose port 8888 for Jupyter notebooks (useful for notebook-based work):
   * (not sure if completely necessary)

*EXPOSE 8888*

1. **Set the Command to Run Jupyter Notebook (Optional):**
   * This sets the default command to run a Jupyter notebook server when the container starts:

*CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8888", "--no-browser", "--allow-root"]*

**Step 4: Building and Running the Docker Container**

**(**Able to do this in VS code or in Docker Desktop)

1. **Build the Docker Image:**
   * Navigate to your project directory where the Dockerfile is located.
   * Run the following command to build the Docker image:

*docker build -t my-python-app .*

* + This will create an image called my-python-app based on the Dockerfile.

1. **Run the Docker Container:**
   * After the image is built, you can run the container using this command:

*docker run -p 8888:8888 -v $(pwd):/app my-python-app*

* + - -p 8888:8888 exposes port 8888 for Jupyter.
    - -v $(pwd):/app mounts the current directory on your host machine to /app inside the container.

**Step 5: Setting Up VS Code to Work Inside the Container**

1. **Open the Docker Container in VS Code:**
   * After your Docker container is running, you can directly open and work in it from VS Code by attaching to the container.
   * **Right-click** on the running Docker container in **Docker Desktop** or use the **Docker extension** in VS Code:
     + In **VS Code**, press Ctrl + Shift + P (or Cmd + Shift + P on macOS) to open the **Command Palette**.
     + Type and select **Remote-Containers: Attach to Running Container**.
     + Choose your running container (e.g., my-python-app), and VS Code will attach to the container.
2. **VS Code Terminal and File Access:**
   * Once attached, you will be able to:
     + Open the **integrated terminal** inside VS Code (Ctrl + ) to run commands directly within the container.
     + Edit files inside the container using the VS Code editor.
     + Any changes you make to files in VS Code will be reflected inside the container (and vice versa), especially if you're using volume mounting (e.g., -v $(pwd):/app).

**Step 6: Install Extensions Inside the Docker Container**

1. **Install Jupyter and Python Extensions**:
   * To install the Python extension inside VS Code:
     + Open the **Extensions** panel (Ctrl + Shift + X).
     + Search for **Python** and click **Install** (make sure it is the extension by Microsoft).
   * To install the **Jupyter extension**:
     + In the **Extensions** panel, search for **Jupyter** and click **Install** (also from Microsoft).
2. **Install Additional Extensions (Optional)**:
   * You can install other useful extensions depending on your workflow:
     + **Pylance** (for Python IntelliSense).
     + **GitLens** (for enhanced Git support).
     + **Docker** (for managing Docker containers within VS Code).
     + **Markdown All in One** (for Markdown editing).
   * Just go to the **Extensions** panel, search for the extension by name, and click **Install**.
3. **Choose a Kernel for Jupyter Notebooks**:
   * After installing the Python and Jupyter extensions, open a .ipynb file in VS Code.
   * In the top right corner of the notebook, click on **Select Kernel** to choose the Python environment or kernel.
   * Select the appropriate kernel corresponding to the Python environment inside your Docker container (it should appear after installation).
4. **Git Integration Inside the Container**:
   * If you are signed into **GitHub** within VS Code and have Git installed in your container, version control (commit, push, pull) should work seamlessly inside the container.
   * Ensure that:
     + Git is installed inside the Docker container (RUN apt-get install -y git in the Dockerfile).
     + You are signed in to GitHub within VS Code (check via the **Source Control** panel or sign in directly).