Docker - Mesh - Environment

1. Install Docker and Verify the Installation

1. **Install Docker** on `https://www.docker.com`
2. **Open** the `.dmg` file and dragging the Docker icon to the Applications folder.
3. **Launch Docker** from your Applications folder. Once Docker is running, you should see a Docker icon in the menu bar at the top of your screen.
4. **Verify Docker** is working by opening a terminal and running this command:

```
docker --version
```

   You should see the Docker version printed out, which means Docker is correctly installed.

2. Create a Dockerfile

This step involves creating a Dockerfile, which is essentially a set of instructions Docker uses to create an environment (image) for running your Python code. In this environment, your packages and dependencies (e.g., `numpy`, `pygalmesh`) will be installed, and the `.tif` file will be processed by your Python script inside a Docker container.

1. **Create a project folder**

   You'll start by creating a directory that will hold everything needed for Docker to run your code.

```
mkdir my_docker_project
cd my_docker_project
```

   This will create a folder called `my_docker_project` and navigate into that folder.

2. **Create a Dockerfile**

   Now, you need to create the Dockerfile. This file tells Docker what environment to create, including installing necessary software and dependencies. In the terminal, you can create a Dockerfile using the `nano` text editor (or `vim`, or any other text editor of your choice):

```
nano Dockerfile
```

   This will open a blank file named Dockerfile in your terminal using the `nano` editor. Now, you'll add the following content to this file:

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /usr/src/app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    libeigen3-dev \
    cmake \
    git \
    build-essential \
    g++ \
```

```
    libcgal-dev \
    libcgal-qt5-dev \
    libopencv-dev \
    python3-opencv \
    && rm -rf /var/lib/apt/lists/*

# Install pygalmesh dependencies
RUN pip install pybind11
RUN pip install numpy meshio tifffile multiprocess imutils meshio==5.3.5 opencv-python

# Clone and install pygalmesh from source
RUN git clone https://github.com/nschloe/pygalmesh.git && \
    cd pygalmesh && \
    pip install .

# Copy the current directory contents into the container
COPY . .

# Run your Python script when the container starts
CMD ["python", "./mesh_voodoo_ABQS_v31.py"]
```

3. **Save and exit the Dockerfile**

   Once you've typed or pasted the content into the nano editor:

   - Press `CTRL + X` to exit.
   - Press `Y` to confirm saving the changes.
   - Press `Enter` to confirm the filename (which is already Dockerfile).

   Your Dockerfile is now created and ready

3. Build the Docker Image

You now need to build the Docker image based on the **Dockerfile** you just created. The Docker image will be the environment in which your code and dependencies are installed.

1. In your terminal, make sure you're still in the `my_docker_project` folder, and then run:

```
docker build -t my_python_image .
```

   - `docker build` is the command to build a Docker image.
   - `-t my_docker_image` gives your Docker image a name (`my_docker_image`).
   - The . at the end tells Docker to look for the Dockerfile in the current directory.

   This command may take a few minutes, as Docker will download the necessary dependencies, install Python packages, and configure the environment according to the **Dockerfile**.

2. Wait for the build process to complete. Docker will output the steps it's executing, and you should see a message like:

```
Successfully built <image-id>
Successfully tagged my_python_image:latest
```

   This means your Docker image was successfully created.

4. Run the python script in the Docker Container

Now that you've built the image, you can run your Python script in the Docker container. In the terminal, run the following command to create and start a container from your image:

```
docker run --rm -v $(pwd):/usr/src/app my_python_image python mesh_voodoo_ABQS_v31.py
```

This will run your Python script inside the Docker container. If the script processes your `.tif` file, it should do so here, and any output will be visible in your terminal.