# LoggerHelper Class and Docker Commands Explanation

## Introduction

This document explains the LoggerHelper class implementation, its usage in the FastAPI project, and the steps to build and run the application using Docker. Additionally, it describes how to access logs generated by the application.

## LoggerHelper Class Implementation

The LoggerHelper class is a centralized logging utility that standardizes the logging setup across the entire application. It provides a method to configure logging and retrieve named loggers for use in different modules.

### Code Implementation

class LoggerHelper:  
 import logging  
 from logging.handlers import RotatingFileHandler  
 import os  
  
 LOG\_DIR = "logs"  
 os.makedirs(LOG\_DIR, exist\_ok=True)  
 LOG\_FILE = os.path.join(LOG\_DIR, "app.log")  
  
 @staticmethod  
 def setup\_logging():  
 # Create a rotating file handler  
 file\_handler = RotatingFileHandler(LoggerHelper.LOG\_FILE, maxBytes=5 \* 1024 \* 1024, backupCount=3)  
 file\_handler.setLevel(logging.INFO)  
  
 # Create a stream handler for console logs  
 console\_handler = logging.StreamHandler()  
 console\_handler.setLevel(logging.INFO)  
  
 # Configure the logging module  
 logging.basicConfig(  
 level=logging.INFO,  
 format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",  
 handlers=[file\_handler, console\_handler],  
 )  
  
 @staticmethod  
 def get\_logger(name: str):  
 return logging.getLogger(name)

### LoggerHelper Class Explanation

1. setup\_logging(): This static method initializes logging by configuring two handlers:  
 - A RotatingFileHandler writes logs to a file (logs/app.log), rotating the file when it reaches 5MB.  
 - A StreamHandler outputs logs to the console.  
 The logs include timestamps, logger names, log levels, and messages.  
2. get\_logger(name: str): This static method returns a logger instance with the specified name. It allows modules to use a named logger for better traceability.

## Usage in Project Files

The LoggerHelper class is used in various modules to log messages. Below is an example of its usage in each file.

### Example: main.py

from app.helpers import LoggerHelper  
from fastapi import FastAPI  
  
# Setup logging  
LoggerHelper.setup\_logging()  
  
app = FastAPI()  
  
@app.get("/")  
def read\_root():  
 logger = LoggerHelper.get\_logger("MainApp")  
 logger.info("Root endpoint called")  
 return {"message": "Hello, World!"}

### Example: user\_controller.py

from app.helpers import LoggerHelper  
  
logger = LoggerHelper.get\_logger("UserController")  
  
# Use logger in endpoints  
logger.info("Controller log example")

## Docker Commands Explanation

To build and run the FastAPI application in a Docker container, the following commands are used:

### 1. docker build -t app:latest .

This command builds a Docker image from the Dockerfile in the current directory. The `-t app:latest` option tags the image with the name 'app' and version 'latest'. Make sure your Dockerfile and project files are in the correct structure before running this command.

### 2. docker run -d -p 5000:5000 app:latest

This command runs the Docker container in detached mode (`-d`) and maps port 5000 of the container to port 5000 on the host machine (`-p 5000:5000`). The container runs the image tagged 'app:latest'.

### 3. docker logs <container\_id>

This command displays the logs of the running container. Replace `<container\_id>` with the actual container ID obtained from the `docker ps` command. Logs include application-level logs and any errors or information printed by the application.