Git Bash + Docker I/O Success Pattern Summary

Goal:

Enable input/output operations between a Docker container and the Windows host using Git Bash as the primary shell interface.

Key Notes:

- Avoid using: docker run -v C:/...:/... -> Git Bash breaks the path at the colon :
- Use: docker-compose.yml instead -> YAML treats strings as-is, bypassing Bash parsing
- Avoid: /mnt/c/... or $PWD -> Git Bash may misinterpret or Docker may not resolve them
- Git Bash can run docker compose safely
- Use fully-qualified Windows paths in volumes: like "C:/Users/..."

Example Project Structure:

```
project-root/
|-- docker-compose.yml
|-- Dockerfile
|-- write_image.py (optional)
|-- output/   (host folder)
```

Dockerfile:

```
FROM python:3.11-slim
WORKDIR /app
```

docker-compose.yml:

```
version: "3.9"

services:
  writer:
    build: .
    volumes:
      - "C:/Users/myname/Documents/project-root/output:/app/output"
    command: sh -c "mkdir -p /app/output && echo hello > /app/output/hello.txt"
```

Execution (in Git Bash):

```
cd /c/Users/myname/Documents/project-root
docker compose up --build
```

Output Check:

```
cat output/hello.txt
# Should print: hello
```

Summary:

- Avoid docker run -v from Git Bash
- Use docker-compose.yml for stability
- Use Windows-style absolute paths in volumes
- Confirm output via bound host folder