Docker 安装与 GPU 支持（Windows/Ubuntu）

1. 实验目的

- 在 Windows（WSL2 + Docker Desktop）与 Ubuntu 环境快速完成 Docker 安装与验证。
- 可选：启用 NVIDIA GPU 容器支持（`docker run --gpus all ...`）。
- 可选：配置国内镜像加速，提高拉取镜像速度。
- 可选：配置 `buildx` + QEMU，支持多架构构建环境。

2. 实验要求

2.1 Windows 端（WSL2 + Docker Desktop）

- 软件要求：Windows 10/11（建议 22H2+）；已开启 CPU 虚拟化；WSL2；Docker Desktop（WSL2 后端）。
- GPU 要求（可选）：NVIDIA 显卡 + 支持 WSL 的 NVIDIA Windows 驱动（用于容器 `--gpus all`）。
- 网络要求：可访问 Docker Hub/镜像源（或按步骤配置镜像加速）。

2.2 Ubuntu 端（脚本安装）

- 软件要求：Ubuntu 20.04/22.04；具备 `sudo` 权限；可联网（apt/curl）。
- GPU 要求（可选）：已安装可用的 NVIDIA 驱动（宿主机/系统内 `nvidia-smi` 可正常运行）。
- 说明：脚本会写入/修改系统 Docker 配置（`/etc/docker/daemon.json`）并可能重启 Docker 服务。

3. 实验地址

例程目录：[安装目录]\RflySimAPIs\2.RflySimUsage\0.ApiExps\e11_docker

文件说明：

- `dockerInstall.sh`：按 Docker 官方源安装 Docker CE（含 buildx/compose 插件）。
- `dockerAcc.sh`：写入国内镜像加速（`/etc/docker/daemon.json`），并尝试重启/验证。
- `dockerGPU.sh`：安装 NVIDIA Container Toolkit、QEMU/binfmt，并验证 `--gpus all`，创建 buildx builder。

4. 实验内容或步骤

4.1 Windows：安装 WSL2

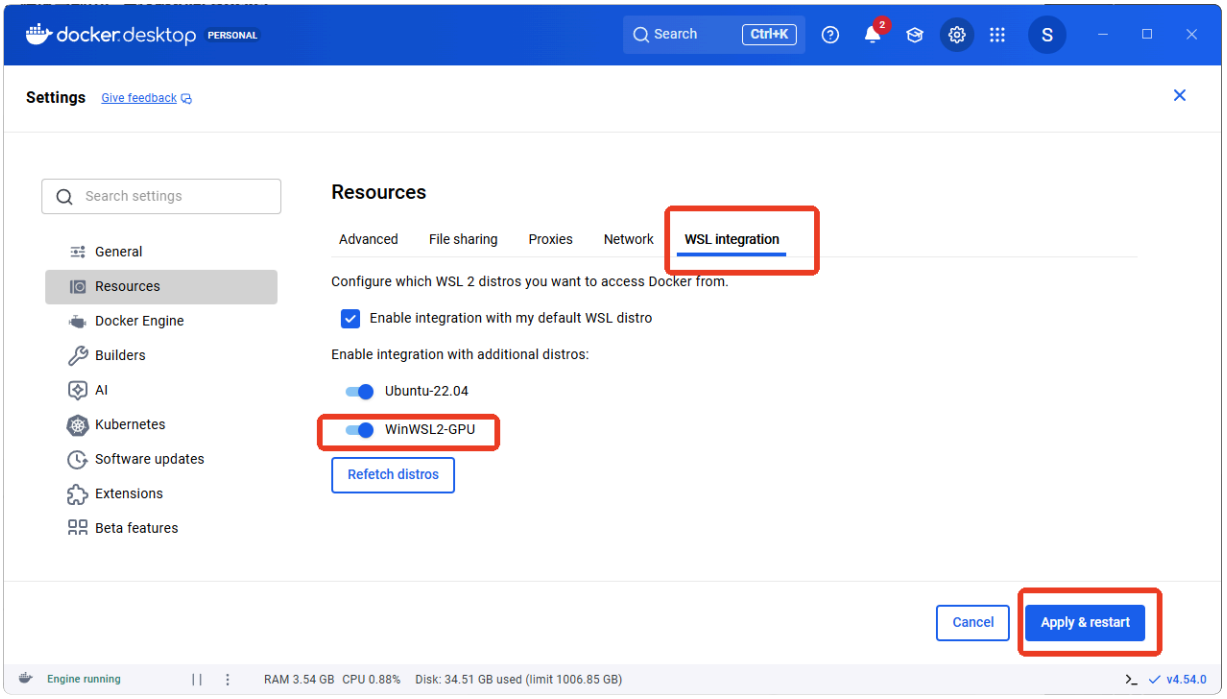
参考如下例程完成WSL2 安装：
/1.RflySimIntro/2.AdvExps/e13.WinWSL2-GPU/Readme.pdf

4.2 Windows：安装并配置 Docker Desktop（WSL2 后端）

1、安装 Docker Desktop（官方安装包）。

2、在 Docker Desktop 设置中确认：

- `General`：勾选 `Use the WSL 2 based engine`
- `Resources -> WSL integration`：对你的 Ubuntu 发行版勾选 `Enable integration`



3、在 WSL中验证 Docker：
双击 WinWSL2.bat，在打开的终端中执行如下命令

```
docker version
docker run --rm hello-world
```

4.3 Windows：GPU 支持验证（可选）

前置：安装支持 WSL 的 NVIDIA Windows 驱动并重启系统。

在 WSL（Ubuntu）中执行（可按需将 `ubuntu22.04` 改为 `ubuntu20.04`）：

```
docker run --rm --gpus all nvidia/cuda:12.6.0-base-ubuntu22.04 nvidia-smi
```

看到显卡信息即表示 GPU 容器可用。

> 说明：Windows 下不建议在 WSL 内再单独安装 Linux 版 Docker Engine（避免与 Docker Desktop 冲突）。

4.4 Windows：配置国内镜像加速（可选，Docker Desktop）

打开 Docker Desktop：`Settings -> Docker Engine`，在 JSON 中加入 `registry-mirrors`（示例）后 `Apply & Restart`：

```
{
  "registry-mirrors": [
    "https://docker.m.daocloud.io",
    "https://mirror.iscas.ac.cn",
    "https://ccr.ccs.tencentyun.com",
    "https://hub-mirror.c.163.com"
  ],
  "dns": ["8.8.8.8", "1.1.1.1"]
}
```

4.5 Ubuntu：使用脚本安装 Docker

1、进入本目录：

```
cd .../e11_docker
```

2、执行安装脚本：

```
bash dockerInstall.sh
```

3、按脚本提示重新登录终端（或执行 `newgrp docker`）后验证：

```
docker run --rm hello-world
```

> 若脚本末尾出现 `permission denied while trying to connect to the Docker daemon socket`，通常是 `docker` 组权限尚未在当前会话生效：重开终端后再试即可。

4.6 Ubuntu：配置国内镜像加速（可选）

执行：

```
bash dockerAcc.sh
```

脚本会备份并覆写 `/etc/docker/daemon.json`，写入 `registry-mirrors` /DNS，并尝试重启 Docker，最后用 `hello-world` 验证。

4.7 Ubuntu：GPU 支持 + buildx 多架构（可选）

前置：系统已安装可用 NVIDIA 驱动（`nvidia-smi` 正常输出）。

执行：

```
bash dockerGPU.sh
```

脚本会安装 `nvidia-container-toolkit`、QEMU/binfmt，并用 CUDA 镜像执行 `nvidia-smi` 验证，同时创建 `buildx` builder：`multiarch-builder`。

验证（可选）：

```
docker run --rm --gpus all nvidia/cuda:12.6.0-base-ubuntu22.04 nvidia-smi
docker buildx ls
```

5. 关键知识点

- Docker Desktop vs. Docker Engine：Windows 推荐使用 Docker Desktop（WSL2 后端）统一提供 daemon；Ubuntu 原生通常安装 `docker-ce`。
- `docker` 组权限：加入 `docker` 组后需重新登录/新开终端，才能免 `sudo` 访问 `/var/run/docker.sock`。
- NVIDIA GPU 容器：需要宿主机显卡驱动支持；Ubuntu 侧通常需安装 `nvidia-container-toolkit`；Windows 侧依赖 Docker Desktop + WSL GPU。
- 镜像加速：Ubuntu 侧通过 `/etc/docker/daemon.json` 配置 `registry-mirrors`；Docker Desktop 在其 Engine 配置中设置。

扩展内容

介绍 Docker 多架构构建

Docker 多架构构建（Multi-Architecture Builds）允许您在单一 Dockerfile 的基础上，为多个 CPU 架构（如 x86_64/amd64 和 ARM64/aarch64）创建兼容的 Docker 镜像。这特别有用在开发环境中，例如从 Windows/WSL2（通常 x86）主机构建镜像，然后部署到 NVIDIA Jetson Orin NX（ARM64 架构）等嵌入式设备上，而无需在目标设备上重新构建。推荐此方案是因为它利用 Docker Buildx（BuildKit 的扩展）来实现交叉编译，减少移植时间，支持自动化 CI/CD，并确保镜像在不同架构上的二进制兼容性。对于 CUDA GPU 加速

的视觉算法，这意味着您可以构建包含 ARM64 特定 CUDA 库的镜像，直接在 Jetson 上运行而无需额外编译。

多架构构建的核心优势：

- **可移植性**：一个镜像标签可以支持多个平台，用户拉取时 Docker 会自动选择正确的变体。
- **效率**：在 x86 主机上使用 QEMU 模拟 ARM64 构建，避免在资源有限的 Jetson 上编译大型依赖（如 OpenCV with CUDA）。
- **CUDA 支持**：结合 NVIDIA 的 L4T（Linux for Tegra）基础镜像，确保 GPU 加速在 Jetson 上无缝工作。
- **局限性**：构建时间可能更长（由于模拟），且某些架构特定优化（如 TensorRT 模型转换）仍需在目标设备上执行。

前提条件

在开始前，确保满足以下要求：

- **Docker 版本**：Docker 19.03 或更高（推荐最新版，如 27.x）。对于 WSL2，使用 Docker Desktop 并启用 WSL2 后端。
- **主机架构**：x86_64（amd64），以构建到 ARM64。
- **QEMU 支持**：用于模拟其他架构。在 Linux/WSL2 上，安装 `qemu-user-static` 和 `binfmt-support`：

  ```
  sudo apt update
  sudo apt install qemu-user-static binfmt-support
  ```

  Docker Desktop（Windows/macOS）已内置 QEMU。
- **NVIDIA 特定**：对于 Jetson 目标，确保主机有 NVIDIA 驱动（可选，但有助于测试）。目标 Jetson 需要 JetPack SDK 安装（e.g., JetPack 6.0+ for CUDA 12.2）。
- **实验功能**：在新版 Docker 中默认启用；旧版需在 `/etc/docker/daemon.json` 中添加 `"experimental": true` 并重启 Docker。

安装和设置 Docker Buildx

Docker Buildx 是 Docker CLI 的插件，用于高级构建。如果未安装：

1. **检查安装**：

   ```
   docker buildx version
   ```

   如果未找到，安装：

- Linux/WSL2：

```
docker buildx install
```

- Docker Desktop：已内置，无需额外步骤。

2. **创建构建器（Builder）**：
   Buildx 使用“构建器”来管理构建环境，支持多节点和模拟。

```
docker buildx create --name multiarch-builder --driver docker-container --use
docker buildx inspect --bootstrap
```

- `--driver docker-container`：使用容器化构建，支持 QEMU。
- `--bootstrap`：启动构建器并注册 QEMU 以支持 ARM64 等架构。
  如果 QEMU 未注册，运行：

```
docker run --rm --privileged multiarch/qemu-user-static --reset -p yes
```

3. **验证**：

```
docker buildx ls
```

您应该看到 `multiarch-builder` 支持多个平台，如 `linux/amd64, linux/arm64`。

构建多架构镜像的步骤

1. **准备 Dockerfile**：
   使用多阶段构建（Multi-Stage Builds）来优化。针对 Jetson Orin NX 的 CUDA 视觉算法，基于 NVIDIA L4T 镜像（ARM64 兼容）。
   示例 Dockerfile（假设您的代码在当前目录）：

```
# 第一阶段：构建依赖（交叉编译友好）
FROM --platform=linux/arm64 nvcr.io/nvidia/l4t-base:r36.2.0 AS builder
RUN apt-get update && apt-get install -y \
    cuda-toolkit-12-2 \
    libopencv-dev \
    python3-pip
RUN pip3 install torch torchvision  # 使用 Jetson 特定 wheel，如果需要
COPY . /app
WORKDIR /app
RUN nvcc -o my_cuda_app main.cu  # 编译您的 CUDA 代码
# 如有视觉算法，添加构建步骤，如 cmake for custom libs

# 第二阶段：运行时镜像（最小化）
FROM --platform=linux/arm64 nvcr.io/nvidia/l4t-base:r36.2.0
COPY --from=builder /app /app
WORKDIR /app
CMD ["./my_cuda_app"]
```

- `--platform=linux/arm64`：指定目标架构。
- 使用 NVIDIA 的镜像仓库（nvcr.io）以获取预编译的 ARM64 CUDA 支持。
- 对于 PyTorch 等 ML 框架，从 NVIDIA 下载 ARM64 wheel（如 `torch-2.3.0-cp310-cp310-linux_aarch64.whl`）并在 Dockerfile 中安装。

2. **构建镜像**：
   在包含 Dockerfile 的目录中运行：

```
docker buildx build --platform linux/arm64 -t yourusername/my-vision-app:arm64 --push .
```

   - `--platform linux/arm64`：指定 ARM64 构建（从 x86 主机交叉编译）。
   - `-t`：镜像标签。
   - `--push`：直接推送至 Docker Hub（需先 `docker login`）。
   - 对于多平台（amd64 和 arm64）：

```
docker buildx build --platform linux/amd64,linux/arm64 -t yourusername/my-vision-app:multi --push .
```

     这会创建一个支持多架构的镜像清单（manifest）。

3. **加载和测试（可选）**：
   - 如果不推送，使用 `--load` 加载到本地（仅单平台）：

```
docker buildx build --platform linux/arm64 -t my-vision-app:arm64 --load .
```

   - 测试：`docker run --gpus all my-vision-app:arm64`（但在 x86 主机上运行 ARM64 镜像需 QEMU，会慢）。

4. **在 Jetson Orin NX 上部署**：
   - 在 Jetson 上安装 Docker 和 NVIDIA Container Toolkit（参考 JetPack 文档）。
   - 拉取镜像：`docker pull yourusername/my-vision-app:arm64`。
   - 运行：`docker run --gpus all --network host my-vision-app:arm64`。
   - 确保 Jetson 的 CUDA 版本匹配 Dockerfile 中的（e.g., 12.2）。

最佳实践和常见问题

- **缓存优化**：使用 `--cache-to=type=inline` 来内联缓存，或外部缓存（如 registry）以加速重复构建。
- **安全**：避免在 Dockerfile 中硬编码凭证；使用 Buildx 的 secrets 支持。
- **性能**：交叉构建 ARM64 时，QEMU 模拟可能慢；对于大型项目，考虑在 ARM64 主机（如另一 Jetson）上构建，或使用 CI/CD（如 GitHub Actions with ARM runners）。
- **常见错误**：

    - "exec /bin/sh: exec format error"：架构不匹配；确保 `--platform` 正确。
    - QEMU 未工作：检查 `docker buildx inspect` 是否列出 arm64。
    - CUDA 不兼容：使用 Jetson 特定镜像和库；测试 `nvidia-smi` 在容器内。
- **扩展**：对于 CI/CD，集成到 GitHub Actions：

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: docker/setup-qemu-action@v3
      - uses: docker/setup-buildx-action@v3
      - run: docker buildx build --platform linux/arm64 --push -t ${{
secrets.DOCKER_USERNAME }}/my-app:arm64 .
```

此方案简化了从 WSL2 Docker 到 Jetson 的移植，确保您的 CUDA GPU 加速视觉算法（如使用 OpenCV 或 TensorRT）高效运行。如果需要特定库的示例，参考 NVIDIA Developer 论坛或 Docker 文档。

6. 参考资料

1. Docker Desktop + WSL2： https://docs.docker.com/desktop/wsl/
2. Docker Engine 安装（Ubuntu）： https://docs.docker.com/engine/install/ubuntu/
3. NVIDIA Container Toolkit：
   https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/
4. CUDA on WSL（NVIDIA）：
   https://docs.nvidia.com/cuda/wsl-user-guide/index.html

7. 常见问题

Q1：执行 `docker run ...` 提示 `Permission denied` / `Cannot connect to the Docker daemon`？

A1：

- Ubuntu：确认 Docker 服务已启动（`sudo systemctl start docker`），并确保已加入 `docker` 组且重新登录终端后生效。

- Windows（Docker Desktop）：确认 Docker Desktop 正在运行，且已开启 WSL integration。

Q2：拉取镜像很慢、经常超时?

A2：

- Ubuntu：执行 `bash dockerAcc.sh` 配置国内镜像加速。
- Windows：在 Docker Desktop 的 `Settings -> Docker Engine` 中配置 `registry-mirrors` 并重启。

Q3：运行 `docker run --gpus all ...` 报错 `could not select device driver "nvidia"`?

A3：

- Windows：更新 NVIDIA Windows 驱动（支持 WSL）与 Docker Desktop，确认 GPU 能在 WSL 环境中正常使用。
- Ubuntu：确认 NVIDIA 驱动可用，并执行 `bash dockerGPU.sh` 安装 `nvidia-container-toolkit`，必要时重启 Docker 服务后重试。