nvidia-docker 설치 가이드 (Ubuntu 22.04 기준)

이 문서는 Ubuntu 22.04 환경에서 NVIDIA GPU를 Docker 컨테이너 내에서 사용할 수 있도록 해주는 **nvidia-docker**(NVIDIA Container Toolkit)를 설치하는 방법을 단계별로 설명합니다.

사전 요구사항

- **Ubuntu 22.04 LTS**가 설치되어 있어야 합니다.
- NVIDIA GPU 드라이버(>= 450)가 호스트에 설치되어 있어야 합니다.
- 기본 Docker(>= 20.10)가 설치되어 있어야 합니다. 설치되지 않은 경우 아래 **1. Docker 설치** 과정을 참고하세요.

1. Docker 설치 (옵션)

Docker가 설치되어 있지 않다면, 다음 명령어로 설치합니다.

```
#!/bin/bash
# 패키지 목록 업데이트 및 필수 패키지 설치
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release

# Docker 공식 GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Docker 저장소 추가
echo \
  "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker 엔진 설치
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

설치 후, `docker --version` 명령으로 버전을 확인하세요.

```
#!/bin/bash
docker --version
# 예시 출력: Docker version 2x..xx, build xxxx
```

2. Docker Compose 설치

```
#!/bin/bash
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

3. NVIDIA Container Toolkit 저장소 추가

NVIDIA Container Toolkit을 설치하기 위해 NVIDIA의 패키지 저장소를 추가합니다.

```
#!/bin/bash
# 배포판 변수 설정
distribution=$(. /etc/os-release; echo $ID$VERSION_ID)

# NVIDIA GPG 키 추가
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg -
-dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-
container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-
container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

# 패키지 목록 업데이트
sudo apt-get update
```

3. nvidia-container-toolkit 설치

아래 명령어로 `nvidia-container-toolkit` 패키지를 설치합니다.

```
#!/bin/bash
export NVIDIA_CONTAINER_TOOLKIT_VERSION=1.17.8-1
  sudo apt-get install -y \
      nvidia-container-toolkit=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      nvidia-container-toolkit-base=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      libnvidia-container-tools=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      libnvidia-container1=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
```

컨테이너 런타임을 구성합니다.

```
#!/bin/bash
sudo nvidia-ctk runtime configure --runtime=docker
```

4. Docker 데몬 재시작

설치된 후 nvidia 런타임을 추가하고,Docker 데몬을 재시작하여 NVIDIA 런타임이 적용되도록 합니다.

```
#!/bin/bash
sudo vim /etc/docker/daemon.json
```

파일에 있는 내용을 아래와 같이 수정해 줍니다. 입력 모드 전환 `i`

```
# Vim
{
    "default-runtime": "nvidia",
    "runtimes": {
        "nvidia": {
            "args": [],
            "path": "nvidia-container-runtime"
        }
    }
}
```

`esc` + `:wq` 명령어를 통해 vim 종료

```
#!/bin/bash
sudo usermod -aG docker $USER
newgrp docker
sudo systemctl restart docker
```

5. 설치 확인 및 테스트

정상적으로 설치되었는지 확인하기 위해 다음 명령어를 실행합니다.

```
#!/bin/bash
# Ubuntu 20.04 CUDA version 11.0
sudo docker run --rm --gpus all \
  nvidia/cuda:11.0.3-base-ubuntu20.04 \
  nvidia-smi

# Ubuntu 22.04 CUDA version 11.8
sudo docker run --rm --gpus all \
  nvidia/cuda:11.8.0-base-ubuntu22.04 \
  nvidia-smi

# Ubuntu 24.04 CUDA version 12.9.1
sudo docker run --rm --gpus all \
  nvidia/cuda:12.9.1-base-ubuntu24.04 \
  nvidia-smi
```

성공 시, `nvidia-smi` 출력과 함께 GPU 정보가 표시됩니다.

6. 사용 예시

컨테이너 실행 시 `--gpus` 플래그를 사용하여 GPU를 할당할 수 있습니다.

```
#!/bin/bash
# 모든 GPU 사용
sudo docker run --rm --gpus all ubuntu:22.04 nvidia-smi

# 특정 GPU만 사용 (예: GPU 0)
sudo docker run --rm --gpus "device=0" ubuntu:22.04 nvidia-smi
```

7. 여름학교 환경 실행

7.1 Dockerhub를 이용한 빌드

2025 인공지능 여름학교 환경을 사전구성하여 docker 환경을 구축된 환경을 이용할 수 있습니다.

```
#!/bin/bash
docker pull ailabsummerschool2025/summerschool-2025:public
```

Docker 컨테이너의 GUI를 호스트 머신에 접근가능하도록 X서버를 설정합니다.

```
#!/bin/bash
xhost +local:docker
XAUTH_FILE=/tmp/.docker.xauth       # 에러시 sudo 를 앞에 붙여서 실행
touch $XAUTH_FILE
xauth nlist $DISPLAY | sed -e 's/^..../ffff/' | xauth -f $XAUTH_FILE nmerge
-
chmod 644 $XAUTH_FILE
```

Pull 받은 이미지를 사용하여 컨테이너를 실행합니다.

원활한 환경 구성을 위해 docker run이 종료되면 컨테이너를 없애는 `--rm` 플래그가 들어있습니다. 환경 내에서 코드를 수정, 코드가 삭제되길 원치 않는 경우 `--rm` 을 제거 하시고 실행 container가 유지됩니다.

```
#!/bin/bash

docker run -it --rm \
    --name isaac-lab-container \
    --gpus all \
    --privileged \
    -e DISPLAY=$DISPLAY \
    -e XAUTHORITY=/tmp/.docker.xauth \
    -v /tmp/.X11-unix:/tmp/.X11-unix:rw \
```

```
    -v $XAUTH_FILE:/tmp/.docker.xauth:rw \
    -v $(pwd):/workspace/host \
    --network=host \
    --shm-size=8g \
    ailabsummerschool2025/summerschool-2025:public
```

위 내용이 포함된 온라인 수강생을 위한 쉘 스크립트는 아래 명령어를 통해 실행가능합니다.

```
#!/bin/bash
sudo bash online_run.sh
```

오프라인 수강생분들께서는 아래 쉘 스크립트를 통해 도커 환경을 이용할 수 있습니다.

```
#!/bin/bash
sudo zsh /home/user/run_env.sh
```

`--rm` 플래그를 제거 하시어 기존에 container에 다시 접근하기를 원하실 때에는

```
#!/bin/bash
docker start -ai isaac-lab-container
docker exec -it isaac-lab-container /bin/bash
```

커맨드를 통해 다시 접근이 가능합니다.

추가 참고 자료

- NVIDIA Container Toolkit 공식 문서: https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/overview.html
- Docker 공식 문서: https://docs.docker.com/
