# 🦾 AI - Setting Up a Debian 12 Docker Environment with NVIDIA RTX 4070 GPU for Ollama

This step-by-step guide demonstrates how to configure a Debian 12 system running a Docker environment that utilizes the NVIDIA RTX 4070 GPU to run the Ollama AI model. The guide covers the installation of the NVIDIA driver, CUDA Toolkit, NVIDIA Container Toolkit, Docker Compose configuration, and testing and troubleshooting steps.

---

## 🏁 Prerequisites

- Debian 12 system.
- NVIDIA RTX 4070 graphics card.
- Docker and Docker Compose installed.
- Internet connection for downloading packages.
- Administrative (sudo) privileges.

---

## 🛠️ Physical Installation

1. Power off the machine and unplug the power cable.
2. Remove the previous graphics card (if applicable).
3. Install the RTX 4070 into the PCIe x16 slot.
4. Connect the power cable (1× 8-pin or 2× 6+2-pin, depending on your power supply).
5. Restart the server.

---

## 1. System Update

Keeping the system up to date ensures you use the latest packages and dependencies.

**Commands:**

```
sudo apt update
sudo apt upgrade -y
```

**Explanation:**

- `apt update` refreshes the package list, and `apt upgrade` installs the latest package versions.
- This helps avoid compatibility issues during the installation of the NVIDIA driver and other tools.

---

## 2. NVIDIA Driver and Firmware Installation

The NVIDIA driver is required for the system to recognize and utilize the RTX 4070 GPU.

**Commands:**

```
sudo apt install -y nvidia-driver firmware-misc-nonfree nvidia-utils
```

**Explanation:**

- `nvidia-driver` provides the driver necessary for GPU operation.
- `firmware-misc-nonfree` includes non-free firmware that may be required for full GPU functionality.
- `nvidia-utils` includes tools like `nvidia-smi` for monitoring GPU status.

**Verification:**

Reboot the system to load the driver:

```
sudo reboot
```

Check GPU functionality:

```
nvidia-smi
```

**Expected Output:**

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.247.01             Driver Version: 535.247.01   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 4070       On  | 00000000:01:00.0 Off |                  N/A |
|  0%   45C    P8             9W / 200W |      4MiB / 12282MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```

If you see the above output, the NVIDIA driver is successfully installed, and the GPU is recognized.

---

## 3. CUDA Toolkit Installation

The CUDA Toolkit enables the GPU's computational capabilities for Ollama.

**Commands:**

```
sudo apt install -y nvidia-cuda-toolkit
```

**Explanation:**

- `nvidia-cuda-toolkit` provides CUDA libraries and tools required for Ollama's GPU support.
- The RTX 4070 supports CUDA 11.x and 12.x versions. The CUDA version in Debian 12's default repositories is generally compatible.

**Verification:**

Check the CUDA version:

```
nvcc --version
```

**Expected Output (example):**

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
```

If the CUDA version is displayed, the installation is successful.

---

## 4. NVIDIA Container Toolkit Installation

The NVIDIA Container Toolkit allows Docker containers to access the GPU.

**Commands:**

```
# Add the NVIDIA Container Toolkit repository
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

# Update the package list and install the toolkit
sudo apt update
sudo apt install -y nvidia-container-toolkit

# Configure Docker to use the NVIDIA runtime
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

**Explanation:**

- The first two commands add the NVIDIA Container Toolkit repository to the Debian package manager and ensure authentication with the GPG key.
- `nvidia-container-toolkit` enables GPU access for Docker.
- `nvidia-ctk runtime configure` sets up the Docker runtime to use the NVIDIA GPU, and the restart activates the changes.

**Verification:**

Test if Docker can access the GPU:

```
docker run --rm --gpus all nvidia/cuda:12.0.0-base-ubuntu20.04 nvidia-smi
```

**Expected Output:**

The command should display the RTX 4070 GPU, similar to the `nvidia-smi` output on the host. If you get an error (e.g., "no GPU devices available"), verify the NVIDIA driver and toolkit installation.

---

## 5. Optional: Modify the Source List

In some cases, you may need to modify the Debian source list to enable the `non-free` and `non-free-firmware` repositories for installing NVIDIA drivers and firmware.

**Commands:**

1. Open the source list for editing:

   ```
   sudo nano /etc/apt/sources.list
   ```

2. Ensure each line ends with:

   ```
   main contrib non-free non-free-firmware
   ```

   Example line:

   ```
   deb http://deb.debian.org/debian bookworm main contrib non-free non-free-firmware
   ```

3. Save the file and update the package list:

   ```
   sudo apt update
   ```

**Explanation:**

- The `non-free` and `non-free-firmware` repositories may be required for proprietary NVIDIA drivers and firmware.

- If the `nvidia-driver` and `firmware-misc-nonfree` packages are already installed, this step is likely unnecessary but worth checking if issues arise.

---

## 6. Optional: Set NVIDIA as Default Docker Runtime

Configuring the default Docker runtime to `nvidia` ensures that all containers use the NVIDIA runtime unless otherwise specified, simplifying GPU support for multiple containers.

**Steps:**

1. Open the `/etc/docker/daemon.json` file for editing:

   ```
   sudo nano /etc/docker/daemon.json
   ```

2. Add or modify the file with the following content:

   ```
   {
     "iptables": true,
     "default-runtime": "nvidia",
     "runtimes": {
       "nvidia": {
         "path": "nvidia-container-runtime",
         "runtimeArgs": []
       }
     }
   }
   ```

3. Save the file and restart the Docker service:

   ```
   sudo systemctl restart docker
   ```

4. Verify that Docker is using the NVIDIA runtime:

   ```
   docker info --format '{{.DefaultRuntime}}'
   ```

   **Expected Output:** `nvidia`

**Explanation:**

- `"iptables": true`: Ensures Docker manages iptables rules, necessary for network configuration.
- `"default-runtime": "nvidia"`: Sets the NVIDIA runtime as the default for all containers.
- `"runtimes": { "nvidia": ... }`: Defines the NVIDIA runtime and its path (`nvidia-container-runtime`).
- This step is optional because your Docker Compose configuration already specifies `runtime: nvidia` for the `ollama` service. However, it simplifies future configurations if you plan to run additional GPU-enabled containers without explicitly defining the runtime.

**Note:**

- The JSON syntax must be valid (e.g., include commas where needed).
- This step is unnecessary if all GPU-using containers explicitly define the NVIDIA runtime, as in your case.

---

## 7. Docker Compose Configuration

Configuring the Ollama container in Docker Compose ensures GPU support and proper network integration with your web application.

**Configuration:**

Create or modify the `docker-compose.yml` file with the following content (only the `ollama` service is shown here; see previous messages for the full file):

```
services:
  ollama:
    image: ollama/ollama:latest
    container_name: ollama
    restart: unless-stopped
    ports:
      - "11434:11434"
    volumes:
      - ./ollama_data:/root/.ollama
    networks:
      - traefik-proxy
    environment:
      - OLLAMA_HOST=0.0.0.0:11434
      - OLLAMA_MAX_LOADED_MODELS=1
      - OLLAMA_KEEP_ALIVE=5m
      - OLLAMA_NUM_PARALLEL=1
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=compute,utility
    runtime: nvidia
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
networks:
  traefik-proxy:
    external: true
```

**Explanation:**

- `image: ollama/ollama:latest`: Uses the latest Ollama Docker image.
- `ports: - "11434:11434"`: Exposes the Ollama API port to the host.
- `volumes: - ./ollama_data:/root/.ollama`: Provides persistent storage for downloaded models.
- `environment`: Environment variables for configuring Ollama and GPU support.
  - `NVIDIA_VISIBLE_DEVICES=all`: Makes all GPUs available to the container.
  - `NVIDIA_DRIVER_CAPABILITIES=compute,utility`: Enables compute and utility functions.
- `runtime: nvidia`: Explicitly uses the NVIDIA runtime.
- `deploy.resources.reservations.devices`: Ensures the container has access to the GPU.

**Execution:**

Navigate to the directory containing the `docker-compose.yml` file and start the container:

```
cd /mnt/raid/docker/ai
docker-compose up -d ollama
```

---

## 8. Verify Ollama GPU Support

Verify that Ollama is using the GPU.

**Commands:**

1. Check the Ollama logs:

```
docker logs ollama
```

**Expected Output:** Look for a message like "NVIDIA GPU detected" to confirm GPU recognition.

2. Enter the Ollama container:

```
docker exec -it ollama bash
```

3. Download a model (e.g., Llama 3 8B, compatible with 12 GB VRAM):

```
ollama pull llama3:8b
```

4. Run a test prompt:

```
ollama run llama3
```

Enter a simple prompt, e.g., `Is GPU enabled?`

5. Check environment variables in the container:

```
env | grep NVIDIA
```

**Expected Output:**

```
NVIDIA_VISIBLE_DEVICES=all
NVIDIA_DRIVER_CAPABILITIES=compute,utility
```

6. Monitor GPU usage on the host:

```
nvidia-smi
```

While Ollama is running, the `nvidia-smi` output should show Ollama processes (e.g., `llama_server`) and GPU memory usage.

**Explanation:**

- The `llama3:8b` is a quantized model that uses less memory, making it suitable for the RTX 4070's 12 GB VRAM.
- If the logs do not indicate GPU usage or `nvidia-smi` shows no activity, GPU support is not functioning correctly.

---

## 9. Test Web Application and Ollama Communication

Your web application (e.g., `app-ai` service) uses the Ollama API. Verify that communication works.

**Command:**

Test the Ollama API directly:

```
curl http://localhost:11434/api/generate -d '{"model": "llama3", "prompt": "Test GPU", "stream": false}'
```

**Explanation:**

- This command sends a simple request to the Ollama API and checks if a response is received.
- If the response is successful, the Ollama API is functioning, and your web application should be able to communicate with it.

---

## 10. Troubleshooting Tips

If GPU support is not working or other issues arise, try the following steps:

1. **Check Docker Version:**

```
docker --version
```

Ensure Docker 19.03 or later is installed, as it is required for GPU support.

2. **Update the Ollama Image:**

```
docker pull ollama/ollama:latest
```

Ensure you are using the latest Ollama image.

3. **Analyze Logs:**

Check container logs for errors:

```
docker logs ollama
docker logs app-ai
```

4. **Model Compatibility:**

If using a large model (e.g., Llama 3 70B), it may not fit in 12 GB VRAM, causing Ollama to fall back to CPU. Use a smaller, quantized model:

```
docker exec -it ollama ollama pull llama3:8b
```

5. **SELinux/AppArmor Issues:**

If SELinux or AppArmor is running on Debian, they may block GPU access. Temporarily disable them for testing:

```
sudo setenforce 0
```

6. **Restart Containers:**

If you modified the `docker-compose.yml` file, restart the containers:

```
cd /mnt/raid/docker/ai
docker-compose down
docker-compose up -d
```

---

## 11. Resources and Further Reading

- **NVIDIA Container Toolkit Installation Guide**: NVIDIA Container Toolkit
- **Ollama GPU Support Documentation**: Ollama GPU Docs
- **Docker Compose Documentation**: Docker Compose

---

## Closing Thoughts

This guide provides a detailed walkthrough for setting up a Debian 12 Docker environment with NVIDIA RTX 4070 GPU support for running the Ollama AI model. By following these steps, Ollama successfully utilizes the GPU, significantly improving AI model inference speed. If further issues arise, check the logs and use the troubleshooting tips. The configuration is flexible and easily integrates with other web applications (e.g., Traefik and Redis).