Ollama GPU Setup with Docker Compose

This guide provides step-by-step instructions for setting up Ollama with NVIDIA GPU support using Docker Compose on Ubuntu.

Step 1: Install NVIDIA Drivers

First, check if NVIDIA drivers are already installed:

nvidia-smi

After installation, reboot your system:

sudo reboot

```
nvidia-smi
```

You should see output showing your GPU information.

Step 2: Install Docker

```
If Docker is not already installed:
```

```
# Update package index
```

sudo systemctl start docker sudo systemctl enable docker

Add your user to docker group (requires logout/login or newgrp docker)

sudo usermod -aG docker $USER

Step 3: Install NVIDIA Container Toolkit

Add NVIDIA Container Toolkit Repository

Get distribution information

```
curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.li
```

Install the Toolkit

Update package list

sudo apt update

Install nvidia-container-toolkit

sudo apt install -y nvidia-container-toolkit

```
### Configure Docker Runtime

‘‘‘bash
# Configure Docker to use NVIDIA runtime
sudo nvidia-ctk runtime configure --runtime=docker

# Restart Docker daemon
sudo systemctl restart docker
```

Verify Installation

```
# Check if NVIDIA runtime is configured
cat /etc/docker/daemon.json
```

You should see:

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

Step 4: Create Docker Compose Configuration

Create a `docker-compose.yml` file:

```
version: '3.8'

services:
  ollama:
    image: ollama/ollama:latest
    hostname: ollama
    ports:
      - "11434:11434"
    volumes:
      - ./models:/root/.ollama/models
    networks:
```

```
      - genai-network
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
    runtime: nvidia
    restart: always

networks:
  genai-network:
    driver: bridge
    name: genai-network

# GPU-enabled Ollama setup without open-webui
```

Step 5: Start Ollama

```
# Start the container in detached mode
sudo docker compose up -d

# Check container status
sudo docker compose ps

# View logs to verify GPU is detected
sudo docker compose logs ollama
```

Look for these indicators in the logs: `- Device 0: NVIDIA [Your GPU Model], compute capability X.X - loaded CUDA backend from /usr/lib/ollama/libggml-cuda.so - CUDA0 model buffer size = XXX.XX MiB`

Step 6: Verify Installation

Test API Connection

```
curl http://localhost:11434/api/version
```

Expected output:

```
{"version":"0.x.x"}
```

Pull and Test a Model

```
# Enter the container
sudo docker exec -it testgpu-ollama-1 bash
```

```
# Inside the container, pull a small model
ollama pull qwen2:0.5b

# Test the model
ollama run qwen2:0.5b "Hello, how are you?"

# Exit the container
exit
```

Test API with curl

```
curl -X POST http://localhost:11434/api/generate \
  -H "Content-Type: application/json" \
  -d '{
    "model": "qwen2:0.5b",
    "prompt": "Why is the sky blue?",
    "stream": false
  }'
```

Step 7: Monitor GPU Usage

Install and use `nvtop` to monitor GPU usage:

```
sudo apt install nvtop
nvtop
```

Project Structure

```
testgpu/
```

Troubleshooting

Common Issues

1. **"NVIDIA driver not found"**
   - Ensure NVIDIA drivers are installed: `nvidia-smi`
   - Reboot after driver installation
2. **"nvidia-container-runtime not found"**
   - Ensure nvidia-container-toolkit is installed
   - Check Docker daemon configuration: `cat /etc/docker/daemon.json`
   - Restart Docker: `sudo systemctl restart docker`
3. **"Permission denied" for Docker commands**
   - Add user to docker group: `sudo usermod -aG docker $USER`
   - Log out and log back in, or run: `newgrp docker`
4. **Container starts but no GPU detected**
   - Check if `runtime: nvidia` is in docker-compose.yml

- Verify GPU device reservation in compose file
- Check container logs: `sudo docker compose logs ollama`

Debug Commands

```
# Check Docker daemon status
sudo systemctl status docker

# Check NVIDIA Container Toolkit
nvidia-ctk --version

# Test NVIDIA Docker integration
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi

# View detailed container logs
sudo docker compose logs -f ollama
```

Performance Notes

- **GeForce MX230**: Can handle small models (0.5B-2B parameters)
- **Memory**: Monitor GPU memory usage with `nvtop`
- **Models**: Start with smaller models and scale up based on your GPU memory

Available Models

Recommended models for different GPU memory sizes:

- **2GB GPU**: qwen2:0.5b, phi3:mini
- **4GB GPU**: llama3.2:3b, qwen2:1.5b
- **8GB+ GPU**: llama3.1:8b, qwen2:7b

API Usage Examples

Generate Text

```
curl -X POST http://localhost:11434/api/generate \
  -H "Content-Type: application/json" \
  -d '{
    "model": "qwen2:0.5b",
    "prompt": "Explain quantum computing",
    "stream": false
  }'
```

List Models

```
curl http://localhost:11434/api/tags
```

Pull Model via API

```
curl -X POST http://localhost:11434/api/pull \
  -H "Content-Type: application/json" \
  -d '{"name": "qwen2:0.5b"}'
```

Security Considerations

- Ollama runs on localhost by default (port 11434)
- For production, consider adding authentication
- Use firewall rules to restrict access if needed

Updates

To update Ollama:

```
# Pull latest image
sudo docker compose pull

# Restart with new image
sudo docker compose up -d
```

Cleanup

To remove everything:

```
# Stop and remove containers
sudo docker compose down

# Remove images (optional)
sudo docker rmi ollama/ollama:latest

# Remove volumes (optional - this deletes downloaded models)
sudo docker volume prune
```

Note: This setup was tested on Ubuntu 22.04 with NVIDIA GeForce MX230. Adjust GPU-specific settings based on your hardware.