Мелочи

- Чистим System volume
- ffmpeg
- Восстанавливаем после ошибки Check of pool pve/data failed (status:64)
- Адмнские шары без домена
- Ollama
- Reverse proxy для WebSocket в Nginx

Чистим System volume

DiskShadow
Delete shadows OLDEST c:\

ffmpeg

g729 to wav

```
ffmpeg -f g729 -i "C:\Video\wav\123.g729" -f wav "C:\Video\wav\123.wav"
```

avi to mov с использованием GPU

```
ffmpeg -i video.avi -c:v h264_nvenc -preset p7 -pix_fmt yuv420p -an test.mov
 50\50 CPU и GPU
ffmpeg -hwaccel cuda -c:v h264_cuvid -i input.avi -c:v h264_nvenc -preset p7 -pix_fmt yuv420p -an output.mov
100% GPU
```

mov to mov

```
ffmpeg -hwaccel cuvid -i input.mov -c:v h264_nvenc -bsf:v h264_mp4toannexb -b:v 5M -c:a copy output.mov
```

100%GPU

Восстанавливаем после ошибки Check of pool pve/data failed (status:64)

```
lvconvert --repair pve/data
```

занимет около 5 минут но может и больше

Админские шары без домена

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System]
"LocalAccountTokenFilterPolicy"=dword:00000001

Ollama

1. Prerequisites

- Ubuntu 22.04 server
- NVIDIA GPU with at least 16 GB VRAM
- Root or sudo access
- Stable internet connection

---

2. Install NVIDIA Drivers

Make sure your GPU drivers are installed and working.

```
sudo apt update
sudo apt install -y ubuntu-drivers-common
ubuntu-drivers devices   # check recommended drivers
sudo ubuntu-drivers autoinstall
```

Reboot and confirm GPU availability:

```
sudo reboot
nvidia-smi
```

You should see details of your GPU.

---

3. Install CUDA Toolkit

Ollama uses CUDA for GPU acceleration. Install CUDA 12.x (recommended).

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-ubuntu2204.pin
```

```
sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600

wget https://developer.download.nvidia.com/compute/cuda/12.2.2/local_installers/cuda-repo-ubuntu2204-12-2-local_12.2.2-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2204-12-2-local_12.2.2-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu2204-12-2-local/cuda-*-keyring.gpg /usr/share/keyrings/
sudo apt update
sudo apt install -y cuda
```

Check CUDA installation:

```
nvcc --version
```

3.1

CUDA Toolkit Installer	
Installation Instructions:	

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-ubuntu2204.pin
sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/13.0.2/local_installers/cuda-repo-ubuntu2204-13-0-local_13.0.2-580.95.05-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2204-13-0-local_13.0.2-580.95.05-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu2204-13-0-local/cuda-*-keyring.gpg /usr/share/keyrings/sudo apt-get update
sudo apt-get -y install cuda-toolkit-13-0
```

3.2

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

4. Install Ollama

Download and install Ollama runtime:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Verify service is running:

```
systemctl status ollama
```

Test Ollama

Run a quick model test:

```
ollama run tinyllama
```

Other models available:

```
ollama pull gpt-oss:20b
```

---

5. Install Docker

Open WebUI runs inside Docker. Follow these steps to install Docker and Docker Compose:

```
# Remove any old versions
sudo apt remove -y docker docker-engine docker.io containerd runc

# Install required packages
sudo apt update
sudo apt install -y ca-certificates curl gnupg lsb-release

# Add Docker’s official GPG key
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

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

# Install Docker and Docker Compose
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
plugin
```

Enable and test Docker:

```
sudo systemctl enable --now docker
sudo docker run hello-world
```

Allow your user to run Docker without sudo:

```
sudo usermod -aG docker $USER
newgrp docker
```

Check Docker Compose version:

```
docker compose version
```

---

6. Install Open WebUI

Clone and run Open WebUI:

```
git clone https://github.com/open-webui/open-webui.git
cd open-webui
docker compose up -d
```

Access the interface at:

```
http://<server-ip>:3000
```

---

7. Connect Open WebUI to Ollama

In the WebUI:

- Go to **Settings → Backends → Ollama**
- Set API URL:

```
http://host.docker.internal:11434
```

Or, if accessing remotely:

```
http://<server-ip>:11434
```

---

8. Enable GPU Support in Docker (Optional)

If you want Docker containers (like Open WebUI) to access GPU directly:

```
# Install NVIDIA Container Toolkit
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee
/etc/apt/sources.list.d/nvidia-docker.list
sudo apt update
sudo apt install -y nvidia-container-toolkit
sudo systemctl restart docker
```

Edit `docker-compose.yml` for Open WebUI:

```
services:
  open-webui:
    deploy:
      resources:
        reservations:
```

```
      devices:
        - driver: nvidia
          count: all
          capabilities: [gpu]
```

Restart:

```
docker compose down
docker compose up -d
```

---

9. Verify GPU Usage

Run a model and check GPU utilization:

```
ollama run gpt-oss:20b
```

In another terminal:

```
nvidia-smi
```

You should see `ollama` using GPU memory.

---

10. Start Chatting with any Model You Like

Now you can:

1. Pull models with Ollama:

```
ollama pull gpt-oss
ollama pull deepseek-r1
ollama pull llama3
ollama pull llama4
```

```
ollama pull gemma3
ollama pull phi4
ollama pull codellama
```

2. Select the model in **Open WebUI**.
3. Chat through your browser!

Reverse proxy для WebSocket в Nginx

Минимальные настройки: /etc/nginx/conf.d/include/proxy.conf

```
# WebSocket support
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

отключаем буферизацию

```
proxy_buffering off;
```