CluebaseVMS

- Ubuntu Server/Desktop 22.04 / 24.04 / 26.04 Pre-Configuration Guide for High-Load VMS
- WebRTC Setup for Cluebase VMS
- Enabling Visual Assistant in Cluebase VMS
- Optimizing PTZ Performance in Cluebase VMS
- Enabling NVIDIA GPU Support for the LPR Module in Cluebase VMS
- Connecting a NAS to Linux via iSCSI

Ubuntu Server/Desktop 22.04 / 24.04 / 26.04 Pre-Configuration Guide for High-Load VMS

This guide describes the recommended initial configuration of Ubuntu systems for stable operation of a high-load video surveillance platform based on **Cluebase VMS** running in Docker containers.

The recommendations apply to:

- Ubuntu Server/Desktop 22.04 LTS
- Ubuntu Server/Desktop 24.04 LTS
- Ubuntu Server/Desktop 26.04 LTS

The system is expected to operate under:

- High network throughput
- Intensive disk I/O
- Large numbers of simultaneous video streams
- NVIDIA GPU acceleration
- Continuous 24/7 workloads

---

1. Fully Update the System

Update package indexes and install all available updates.

```
sudo apt update && sudo apt dist-upgrade -y && sudo apt autoremove -y && sudo apt autoclean && sudo snap refresh
```

Reboot after updates:

```
sudo reboot
```

---

2. Disable Automatic System Updates

Automatic updates may restart services unexpectedly and affect VMS stability.

Disable unattended upgrades:

```
sudo systemctl stop unattended-upgrades && sudo systemctl disable unattended-upgrades
```

Disable periodic APT updates:

```
sudo nano /etc/apt/apt.conf.d/20auto-upgrades
```

Set:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Save the file.

---

3. Install Additional System Packages

Install useful administration and monitoring utilities:

```
sudo apt install -y \
mc \
htop \
curl \
unzip \
```

```
apt-transport-https \
ca-certificates \
gnupg \
lsb-release \
gnupg2 \
iotop \
net-tools \
iftop \
nvme-cli \
smartmontools \
jq
```

Package purpose:

Package	Purpose
mc	File manager
htop	Process monitoring
iotop	Disk I/O monitoring
iftop	Network traffic monitoring
smartmontools	Disk health monitoring
nvme-cli	NVMe SSD management

---

4. Check Secure Boot Status

Secure Boot must be disabled in BIOS/UEFI before installing proprietary NVIDIA drivers.

Check status:

```
mokutil --sb-state
```

Possible output:

```
SecureBoot enabled
```

or

```
SecureBoot disabled
```

If Secure Boot is enabled:

1. Reboot the server
2. Enter BIOS/UEFI
3. Disable:
   - Secure Boot
   - Fast Boot (recommended)
4. Save settings and reboot

---

5. Install NVIDIA Driver and NVIDIA Container Toolkit

Install NVIDIA drivers only if the server has an NVIDIA GPU.

Recommended driver installation:

```
sudo ubuntu-drivers autoinstall
```

Reboot:

```
sudo reboot
```

Verify driver installation:

```
nvidia-smi
```

Expected result:

- GPU model displayed
- Driver version visible
- No errors

Install NVIDIA Container Toolkit only after installing Cluebase VMS

or Docker

Official documentation:

NVIDIA Container Toolkit Install Guide

Configure the production 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 packages list from the repository:

```
sudo apt update
```

Install the NVIDIA Container Toolkit packages:

```
export NVIDIA_CONTAINER_TOOLKIT_VERSION=1.19.0-1
  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}
```

Configure runtime:

```
sudo nvidia-ctk runtime configure --runtime=docker
```

Restart Docker:

```
sudo systemctl restart docker
```

6. Enable TCP BBR Congestion Control

BBR improves network throughput and reduces latency under heavy traffic.

Open sysctl configuration:

```
sudo nano /etc/sysctl.conf
```

Add:

```
net.core.default_qdisc=fq
net.ipv4.tcp_congestion_control=bbr
```

Apply settings:

```
sudo sysctl -p
```

Verify:

```
sysctl net.ipv4.tcp_congestion_control
```

Expected output:

```
net.ipv4.tcp_congestion_control = bbr
```

---

7. Configure Static IP Address

Ubuntu 22.04/24.04/24.06 uses Netplan.

Find interface name:

```
ip a
```

Example interface:

```
ens18
```

Edit Netplan configuration:

```
sudo nano /etc/netplan/01-netcfg.yaml
```

Example configuration:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens18:
      dhcp4: false
      addresses:
        - 192.168.1.100/24
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses:
          - 1.1.1.1
          - 8.8.8.8
```

Apply configuration:

```
sudo netplan apply
```

Verify:

```
ip a
ip route
```

---

8. Configure Additional fs.aio-max-nr Parameter

Increase asynchronous I/O limits for high-load applications.

Edit sysctl configuration:

```
sudo nano /etc/sysctl.d/99-aio.conf
```

Add:

```
fs.aio-max-nr = 1048576
```

Apply:

```
sudo sysctl --system
```

Verify:

```
sysctl fs.aio-max-nr
```

Additional Recommended Optimizations for High-Load Cluebase VMS Systems

Increase File Descriptor Limits

Large video systems require high open-file limits.

Edit:

```
sudo nano /etc/security/limits.conf
```

Add:

```
* soft nofile 1048576
* hard nofile 1048576
root soft nofile 1048576
root hard nofile 1048576
```

Also configure systemd:

```
sudo nano /etc/systemd/system.conf
```

Add or modify:

```
DefaultLimitNOFILE=1048576
```

Edit:

```
sudo nano /etc/systemd/user.conf
```

Add:

```
DefaultLimitNOFILE=1048576
```

Reload:

```
sudo systemctl daemon-reexec
```

Increase Network Buffers

Edit:

```
sudo nano /etc/sysctl.conf
```

Add:

```
net.core.rmem_max = 67108864
net.core.wmem_max = 67108864
net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.ipv4.tcp_rmem = 4096 87380 67108864
net.ipv4.tcp_wmem = 4096 65536 67108864
net.core.netdev_max_backlog = 250000
```

Apply:

```
sudo sysctl -p
```

Set Performance CPU Governor

Recommended for dedicated VMS servers.

Install tools:

```
sudo apt install -y linux-tools-common linux-tools-generic
```

Set performance mode:

```
sudo cpupower frequency-set -g performance
```

Verify:

```
cpupower frequency-info
```

Enable TRIM for SSD/NVMe

```
sudo systemctl enable fstrim.timer
sudo systemctl start fstrim.timer
```

Verify:

```
systemctl status fstrim.timer
```

Recommended BIOS Settings

For stable 24/7 operation:

Disable:

- Secure Boot
- Fast Boot
- CPU C-States (optional for latency-sensitive systems)
- ASPM power saving (optional)

Enable:

- Above 4G Decoding (for large GPUs)
- Resizable BAR (if supported)
- Performance power profile

Monitoring Recommendations

Useful commands:

CPU:

```
htop
```

Disk I/O:

```
iotop
```

GPU:

```
watch -n 1 nvidia-smi
```

Network:

```
iftop
```

Disk health:

```
sudo smartctl -a /dev/sda
```

Final Recommendations

For production Cluebase VMS servers:

- Use Ubuntu Server instead of Desktop whenever possible
- Use enterprise-grade SSD/NVMe drives
- Use RAID with battery-backed cache for archive storage
- Use dedicated NICs for camera traffic
- Prefer 10G networking for large installations
- Keep OS and archive storage separated
- Avoid installing unnecessary GUI applications and services
- Reboot only during maintenance windows
- Use UPS power protection
- Monitor disk temperatures and SMART status regularly

WebRTC Setup for Cluebase VMS

Local Network Usage

WebRTC works inside a local network by default.
No additional configuration is required.

A TURN server is only needed if remote access to Cluebase VMS is required over the Internet.

---

1. Install a TURN Server in Docker on a VPS

Create a `compose.yaml` file with the following content:

```
services:
  coturn:
    image: coturn/coturn
    container_name: coturn
    restart: always
    network_mode: 'host'
    command:
      - -n
      - --log-file=stdout
      - --listening-ip=0.0.0.0
      - --relay-ip=local_ip
      - --external-ip=public_ip
      - --min-port=40000
      - --max-port=59000
      - --no-auth
```

Replace the following values:

- `local_ip` — local IP address of the VPS
- `public_ip` — public IP address of the VPS

Start the TURN server:

```
docker compose up -d
```

---

2. Using the Same Server for Cluebase VMS and TURN

If the server where Cluebase VMS is installed has a public IP address, the TURN server can be installed on the same server.

In this case:

- Install the TURN server on the same machine as Cluebase VMS
- Configure port forwarding on the router

Ports that need to be forwarded in the router:

- `3478` TCP and UDP
- `40000-59000` UDP

If a firewall is enabled, make sure these ports are allowed in the firewall rules.

---

3. Configure Cluebase VMS

Edit the `.env` file of Cluebase VMS.

Change:

```
ENABLE_STUN_TURN=0
```

to:

```
ENABLE_STUN_TURN=1
```

Add the following parameters:

```
STUN_SERVER_HOST=turn_server_ip
STUN_SERVER_PORT=3478
```

Replace `turn_server_ip` with the IP address of your TURN server.

Enabling Visual Assistant in Cluebase VMS

1. Install and Run Ollama with the LLaVA Model

Create a `compose.yaml` file with the following content:

```
volumes:
  ollama:

services:
  ollama:
    image: ollama/ollama
    container_name: ollama
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
    environment:
      - OLLAMA_SCHED_SPREAD=1
    ports:
      - "11434:11434"
    volumes:
      - ollama:/root/.ollama
    restart: always
```

Start the Ollama container:

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

After the container is running, download the `llava` model:

```
docker exec -it ollama ollama pull llava
```

---

2. Enable Ollama Support in the MySQL Database

Connect to the MySQL database container:

```
docker exec -it vms-db mysql -uroot -p
```

When prompted, enter the MySQL root password.

Select the `vcloud` database:

```
USE vcloud;
```

Enable Ollama integration by executing the following query:

```
UPDATE GeneralSettings SET ollamaStatus="ACTIVE";
```

Exit the MySQL console:

```
exit
```

Result

After completing these steps:

- Ollama will be running in Docker
- The `llava` visual model will be installed
- Visual Assistant support will be activated in Cluebase VMS

Optimizing PTZ Performance in Cluebase VMS

To achieve the **lowest possible latency** and the most responsive PTZ (Pan-Tilt-Zoom) control, please ensure your system is configured with the following settings:

1. **Set Camera Codec to H.264.** Configure your camera's primary stream to use the **H.264** video codec. While H.265 is efficient for storage, H.264 typically offers better compatibility and lower processing overhead for real-time control.
2. **Use WebRTC Protocol.** Ensure that **WebRTC** is selected as the streaming protocol in the Cluebase VMS interface. WebRTC is specifically designed for real-time communication and significantly reduces the delay compared to traditional protocols like WS or HLS.

   image.png

Enabling NVIDIA GPU Support for the LPR Module in Cluebase VMS

Prerequisites

NVIDIA GPU support requires the NVIDIA drivers and NVIDIA Container Toolkit to be installed on the host system.

Install and configure NVIDIA Container Toolkit before proceeding:

NVIDIA Container Toolkit Documentation

You can verify that Docker has access to the NVIDIA runtime using:

```
docker info | grep -i nvidia
```

You can also verify GPU availability on the host system with:

```
nvidia-smi
```

1. Load the Docker image

Copy the `roadar_lpr_websocket.tar.gz` archive containing the Docker image to the server.

Load the Docker image using the following command:

```
sudo docker load -i roadar_lpr_websocket.tar.gz
```

2. Stop the existing `vms-lpr` container

Navigate to the directory where Cluebase VMS is installed and stop the `vms-lpr` container if it is currently running:

```
sudo docker compose down vms-lpr
```

3. Update the `docker-compose.yml` configuration

Open the `docker-compose.yml` file in a text editor.

Replace the following configuration block:

```
vms-lpr:
  image: vcloudaiorg/vcloudai-vms-lpr:latest
  restart: always
  container_name: vms-lpr
  network_mode: host
  extra_hosts:
    - host.docker.internal:host-gateway
  depends_on:
    - vms-server
  volumes:
    - ./static/lpr:/data
  environment:
    WS_HOST: host.docker.internal:${WS_SERVER_PORT}
    API_HOST: 127.0.0.1
    API_PORT: ${ROADAR_PORT}
    STATE_FILE: /data/state.json
    LICENSE_SERVICE_HOST: 127.0.0.1
    LICENSE_SERVICE_PORT: 32433
```

with the following GPU-enabled configuration:

```
vms-lpr:
  image: roadar_lpr_websocket
  restart: always
  container_name: vms-lpr
  network_mode: host
  extra_hosts:
    - host.docker.internal:host-gateway
  depends_on:
    - vms-server
  runtime: nvidia
  deploy:
    resources:
      reservations:
        devices:
          - driver: nvidia
            count: all
            capabilities: [gpu, compute, video]
  volumes:
    - ./static/lpr:/data
  environment:
    - WS_HOST=host.docker.internal:${WS_SERVER_PORT}
    - API_HOST=127.0.0.1
    - API_PORT=${ROADAR_PORT}
    - STATE_FILE=/data/state.json
    - LICENSE_SERVICE_HOST=127.0.0.1
    - LICENSE_SERVICE_PORT=32433
    - TRIAL_TIME=0
    - NVIDIA_VISIBLE_DEVICES=all
    - NVIDIA_DRIVER_CAPABILITIES=compute,video,utility
```

Save the file and exit the text editor.

4. Start the `vms-lpr` container

Run the following command to start the container with NVIDIA GPU support enabled:

```
sudo docker compose up -d vms-lpr
```

5. Verify NVIDIA GPU support inside the container

To verify that the container has access to the NVIDIA GPU, run:

```
sudo docker exec -it vms-lpr nvidia-smi
```

If GPU support is configured correctly, the command will display information about the installed NVIDIA GPU(s).

Connecting a NAS to Linux via iSCSI

If your **vCloud server** and **NAS storage** are on separate machines, the most reliable way to connect them is **iSCSI**.

iSCSI allows the NAS to present a disk over the network, and Linux sees it as a **local block device** (like `/dev/sdb`). This is ideal for video storage because it usually performs better than SMB/NFS for large sequential writes.

Architecture

```
+-------------------+        iSCSI        +-------------------+
|   vCloud Server   | <-----------------> |        NAS        |
| Ubuntu Server     |                     | Synology/TrueNAS  |
| 192.168.1.10      |                     | 192.168.1.20      |
+-------------------+                     +-------------------+
```

Step 1 — Prepare the NAS

You need to configure the NAS and create an iSCSI LUN. To do this, refer to the user manual for your NAS server.

Step 2 — Install iSCSI Client on Ubuntu

On the **vCloud server**:

```
sudo apt update
sudo apt install open-iscsi -y
```

Enable the service:

```
sudo systemctl enable --now iscsid
```

Step 3 — Discover the NAS Target

Replace `192.168.1.20` with your NAS IP.

```
sudo iscsiadm -m discovery -t sendtargets -p 192.168.1.20
```

You should see something like:

```
192.168.1.20:3260,1 iqn.2026-07.local.synology:vcloud-storage
```

Step 4 — Configure Authentication (if CHAP is enabled)

Edit the node configuration:

```
sudo nano /etc/iscsi/iscsid.conf
```

Find and set:

```
node.session.auth.authmethod = CHAP
node.session.auth.username = vcloud
node.session.auth.password = StrongPassword123
```

Save the file.

Restart the service:

```
sudo systemctl restart iscsid
```

Step 5 — Log In to the iSCSI Target

```
sudo iscsiadm -m node --login
```

Expected output:

```
Login to [iface: default, target: iqn.2026-07.local.synology:vcloud-storage, portal:
192.168.1.20,3260] successful.
```

Step 6 — Verify the New Disk

List disks:

```
lsblk
```

Example:

```
sda    1.8T
├─sda1
└─sda2
sdb   10.0T
```

The iSCSI disk is usually `sdb`.

Step 7 — Create a Filesystem

⚠ **This will erase the iSCSI disk.**

```
sudo mkfs.ext4 /dev/sdb
```

Step 8 — Create a Mount Point

```
sudo mkdir -p /mnt/vcloud-storage
```

Step 9 — Mount the Disk

```
sudo mount /dev/sdb /mnt/vcloud-storage
```

Check:

```
df -h
```

Example:

```
/dev/sdb        9.8T   24K  9.3T   1% /mnt/vcloud-storage
```

Step 10 — Make the Mount Persistent

Get the UUID:

```
sudo blkid /dev/sdb
```

Example:

```
/dev/sdb: UUID="a1b2c3d4-e5f6-7890-abcd-1234567890ef"
```

Edit `/etc/fstab`:

```
sudo nano /etc/fstab
```

Add:

```
UUID=a1b2c3d4-e5f6-7890-abcd-1234567890ef /mnt/vcloud-storage ext4 _netdev,nofail 0 2
```

Test:

```
sudo umount /mnt/vcloud-storage
sudo mount -a
```

If there are no errors, the configuration is correct.

Step 11 — Ensure iSCSI Reconnects After Reboot

Enable automatic login:

```
sudo iscsiadm -m node --op update -n node.startup -v automatic
```

Check:

```
sudo iscsiadm -m node
```

Useful Commands

Check iSCSI sessions

```
sudo iscsiadm -m session
```

Log out

```
sudo iscsiadm -m node --logout
```

Rediscover targets

```
sudo iscsiadm -m discovery -t sendtargets -p 192.168.1.20
```