Serve App Installation Guide

- Install Unifi Controller On Proxmox Container Ubuntu 24.04.3
- Installing CyberPanel on Proxmox Container Ubuntu 22.04
- Install Uptime Kuma on Ubuntu with Node.js and Nginx
- Install Cacti on Proxmox Container Ubuntu 24.04
- Installing Odoo 19 using Packaged installers on Ubuntu 24.04
- Install Transmission with Webui & Create Directory Indexing Ubuntu 24.04
- Install Moodle In Ubuntu 24.04 & Access from local ip (http,https) and domain
- Install ERPNext 15 on Ubuntu 24.04 Proxmox CT
- FFmpeg + NGINX HTTPS HLS Streaming Server Ubuntu 24.04
- Enable HTTPS LibreNMS Ubuntu 24.04

Install Unifi Controller On Proxmox Container Ubuntu 24.04.3

1. Make sure the ca-certificates package is installed

```
apt-get update; apt-get install ca-certificates curl -y
```

2. Download and execute the script! *(change it to your wanted version)*

```
curl -sO https://get.glennr.nl/unifi/install/unifi-9.4.17.sh && bash unifi-9.4.17.sh
```

3. Once the installation is completed, browse to your server IP address

```
https://localhost:8443
```

Installing CyberPanel on Proxmox Container Ubuntu 22.04

The CyberPanel installation process is straightforward and can be completed by following the provided instructions.

- **Step 1**: **Connect to the server via SSH client.** (**Putty**, **Bitvise SSH client**, etc)
  First login to your server through SSH client as a Root user (Sudo will not work). You can get the Login details from your web host.
- **Step 2**: **Update server Packages.**
  Update your server OS first (Because it updated overall services because it provides much better compatibility). Run this command

1. Update and upgrade ubuntu server first.

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

2. Run the Installation Script**.** Execute the provided command to initiate the automated installation script. This script will guide you through several decisions regarding the LiteSpeed version and additional add-ons you wish to install.

```
sh <(curl https://cyberpanel.net/install.sh || wget -O - https://cyberpanel.net/install.sh)
```

3. Upon completion of the installation process, you will encounter a screen displaying important information regarding your configuration. It is recommended to select and securely copy this information to a safe location for future reference.

Install Uptime Kuma on Ubuntu with Node.js and Nginx

One of the first steps is to install Node.js on Ubuntu but before you do so make sure your server is updated. Please run the x2 commands below before following the tutorial.

```
sudo apt update && apt-get upgrade -y
```

Once your OS is updated, you can run the below command to install Node.js:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash - && sudo apt install -y nodejs
```

Once Node.js is installed, you will need to clone the Git Repo of Uptime Kuma, however before doing that make sure you have Git installed with the below command:

```
sudo apt update && sudo apt install -y git
```

To clone the Uptime Kuma GitHub repository locally, please run the below command in your terminal which will clone the latest release:

```
git clone https://github.com/louislam/uptime-kuma.git
```

Make a note of the directory where you cloned the repository e.g. /home and navigate to it.

```
cd /home/uptime-kuma
```

Once you are in the relevant directory you need to run the setup script along with the PM2 process manager which will keep Uptime Kuma running:

```
npm run setup
npm install pm2 -g
```

We also recommend setting up log rotation with the below command as logs will help you troubleshoot if any issues arise:

```
pm2 install pm2-logrotate
```

Once you have the above setup, it is now time to start Uptime Kuma with a simple command below and to ensure that if you ever reboot your server, Uptime Kuma starts back automatically:

```
pm2 start server/server.js --name uptime-kuma
pm2 startup
```

To make sure Uptime Kuma can be accessed via the browser and a domain name, Nginx or Apache needs to be setup as a reverse proxy, in this case we will be using Nginx:

```
apt install nginx -y
```

Once Nginx has been installed, make sure it is installed and running the latest version with the below commands:

```
nginx -v
systemctl status nginx
```

Now you have to create a Nginx configuration file, also known as a “conf” file with the below contents that you can copy and paste, you can either use Vim or Nano editor:

```
nano /etc/nginx/conf.d/uptime-kuma.conf
```

Configuration file contents with improved security added along with performance enhancements, replace “uptime-kuma.yourdomainname.com” with your own domain name if you have one:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.com;

    location / {
        proxy_pass         http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection "upgrade";
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key $http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version $http_sec_websocket_version;
```

```
        proxy_set_header   Sec-WebSocket-Extensions $http_sec_websocket_extensions;

        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }

    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Once you have saved the above configuration file, just restart Nginx and you will now have Uptime Kuma running on your Linux server:

```
systemctl restart nginx
```

You should be able to access the Uptime Kuma dashboard now at your domain name, our example used in this guide was “uptime-kuma.yourdomainname.com”.

Install Cacti on Proxmox Container Ubuntu 24.04

Here’s a clean step-by-step guide for Cacti installation on Ubuntu 24.04 inside your Proxmox CT

Update the System

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

Install Required Packages, Cacti needs a web server, PHP, SNMP, RRDtool, and MySQL/MariaDB.

```
sudo apt install -y apache2 mariadb-server mariadb-client \
php php-mysql php-snmp php-xml php-ldap php-gd php-mbstring php-gmp \
rrdtool snmp snmpd librrds-perl unzip git
```

Enable required Apache modules:

```
sudo a2enmod php* rewrite
sudo systemctl restart apache2
```

Secure MariaDB:

```
sudo mysql_secure_installation
```

- Set root password
- Remove anonymous users
- Disallow remote root login
- Remove test DB
- Reload privilege tables

Create Cacti database and user:

```
mysql -u root -p
CREATE DATABASE cacti CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'cactiuser'@'localhost' IDENTIFIED BY 'yourpassword';
GRANT ALL PRIVILEGES ON cacti.* TO 'cactiuser'@'localhost';
GRANT SELECT ON mysql.time_zone_name TO 'cactiuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Load time zone data:

```
mysql_tzinfo_to_sql /usr/share/zoneinfo | sudo mysql -u root -p mysql
```

Download Cacti official GitHub: https://github.com/Cacti/cacti

```
cd /var/www/html
sudo git clone -b master https://github.com/Cacti/cacti.git
sudo chown -R www-data:www-data cacti
```

Import Default Database

```
mysql -u cactiuser -p cacti < /var/www/html/cacti/cacti.sql
```

Configure Cacti Settings,Edit config file:

```
sudo nano /var/www/html/cacti/include/config.php
```

Set DB info:

```
$database_type     = "mysql";
$database_default  = "cacti";
$database_hostname = "localhost";
$database_username = "cactiuser";
$database_password = "yourpassword";
$database_port     = "3306";
```

Configure Apache, for cacti

```
sudo nano /etc/apache2/sites-available/cacti.conf
```

Paste

```
<VirtualHost *:80>
    DocumentRoot /var/www/html/cacti
    ServerName cacti.local

    <Directory /var/www/html/cacti>
        Options +FollowSymLinks
        AllowOverride All
        Require all granted
```

```
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/cacti_error.log
    CustomLog ${APACHE_LOG_DIR}/cacti_access.log combined
</VirtualHost>
```

Enable site:

```
sudo a2ensite cacti.conf
sudo systemctl reload apache2
```

Configure Cron for Poller:

```
sudo nano /etc/cron.d/cacti
```

Add:

```
*/5 * * * * www-data php /var/www/html/cacti/poller.php > /dev/null 2>&1
```

Configure SNMP, Edit SNMP daemon:

```
sudo nano /etc/snmp/snmpd.conf
```

Example (simple):

```
rocommunity public
sysLocation ServerRoom
sysContact admin@example.com
```

Restart SNMP:

```
sudo systemctl restart snmpd
```

Access Cacti Web Installer

```
http://<your-server-ip>/cacti
```

Now you have Cacti latest version running on Ubuntu 24.04

Installing Odoo 19 using Packaged installers on Ubuntu 24.04

Odoo provides packaged installers for Debian-based Linux distributions (Debian, Ubuntu, etc.), RPM-based Linux distributions (Fedora, CentOS, RHEL, etc.), and Windows for the Community and Enterprise editions.

Odoo needs a PostgreSQL server to run properly.

```
sudo apt install postgresql -y
```

Install wkhtmltopdf (with patched Qt), remove any old system package to avoid conflict.

```
sudo apt update
sudo apt remove --purge -y wkhtmltopdf
```

Install common dependencies and fonts (needed for rendering).

```
sudo apt install -y \
  xfonts-75dpi xfonts-base fontconfig libxrender1 libxext6 \
  libssl3 libc6 libstdc++6 wget ca-certificates \
  ttf-mscorefonts-installer
```

update font cache (helps avoid strange missing-font rendering problems).

```
sudo fc-cache -f -v
```

There are packaged builds that include the patched Qt. Use the `packaging` GitHub release binary for amd64 that is known to work with Ubuntu 22.04/24.04.

```
cd /tmp
wget https://github.com/wkhtmltopdf/packaging/releases/download/0.12.6.1-2/wkhtmltox_0.12.6.1-2.jammy_amd64.deb
sudo dpkg -i wkhtmltox_0.12.6.1-2.jammy_amd64.deb
sudo apt --fix-broken install -y
```

Ensure binary is in standard PATH (some packages install to /usr/local/bin).

```
sudo ln -sf /usr/local/bin/wkhtmltopdf /usr/bin/wkhtmltopdf
sudo ln -sf /usr/local/bin/wkhtmltoimage /usr/bin/wkhtmltoimage
rm wkhtmltox_0.12.6.1-2.jammy_amd64.deb
```

Odoo S.A. provides a repository that can be used to install the **Community** edition by executing the following commands:

```
 wget -q -O - https://nightly.odoo.com/odoo.key | sudo gpg --dearmor -o
/usr/share/keyrings/odoo-archive-keyring.gpg
 echo 'deb [signed-by=/usr/share/keyrings/odoo-archive-keyring.gpg]
https://nightly.odoo.com/19.0/nightly/deb/ ./' | sudo tee /etc/apt/sources.list.d/odoo.list
 sudo apt-get update && sudo apt-get install odoo
```

After installation, open odoo using web browser with port 8069.

```
http://server-ip:8069/
```

Install Transmission with Webui & Create Directory Indexing Ubuntu 24.04

```
sudo apt update
sudo apt install -y transmission-daemon transmission-cli transmission-common
```

Stop the service (so we can edit config safely)

```
sudo systemctl stop transmission-daemon
```

Edit Transmission config, open with nano

```
sudo nano /var/lib/transmission-daemon/info/settings.json
```

Change (or confirm) these important options:

```
"rpc-authentication-required": true,
"rpc-username": "yourusername",
"rpc-password": "yourpassword",   // it will be hashed after restart
"rpc-whitelist-enabled": false,   // allow access from any IP (or keep enabled and set IPs)
"rpc-bind-address": "0.0.0.0",
"rpc-port": 9091,
```

Make sure the debian-transmission user owns the folder:

```
sudo chown -R debian-transmission:debian-transmission /var/lib/transmission-daemon/info
```

Start and enable the service

```
sudo systemctl enable transmission-daemon
sudo systemctl start transmission-daemon
```

Open your browser:

```
http://<your-server-ip>:9091
```

Install Moodle In Ubuntu 24.04 & Access from local ip (http,https) and domain

Prepare Your Container, do update & upgrade:

```
apt update && apt upgrade -y
apt install -y unzip git curl software-properties-common
```

Install Dependencies (Nginx only, no Apache):

```
apt install -y nginx mariadb-server mariadb-client php-fpm php-mysql php-xml php-mbstring php-curl php-zip php-gd php-intl php-soap php-ldap php-bcmath unzip git
```

Enable services:

```
systemctl enable --now nginx mariadb php8.3-fpm
```

Secure MariaDB & Create DB:

```
mysql_secure_installation
mysql -u root -p
CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'moodleuser'@'localhost' IDENTIFIED BY 'StrongPass123!';
GRANT ALL PRIVILEGES ON moodle.* TO 'moodleuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Install Moodle:

```
cd /var/www
git clone -b MOODLE_405_STABLE https://github.com/moodle/moodle.git
mkdir /var/moodledata
chown -R www-data:www-data /var/www/moodle /var/moodledata
chmod -R 755 /var/www/moodle
```

Nginx Config (Domain via NPM):

```
server {
        listen 80;
        server_name 172.16.X.X;   # Change to your server IP
        root /var/www/moodle;
        index index.php;
        client_max_body_size 100M;

        location / {
            try_files $uri $uri/ /index.php?$query_string;
        }

        location ~ [^/]\.php(/|$) {
            fastcgi_split_path_info ^(.+?\.php)(/.*)$;
            fastcgi_pass unix:/run/php/php8.3-fpm.sock;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
        }
    }
```

Enable configuration file and delete nginx default file:

```
ln -s /etc/nginx/sites-available/moodle /etc/nginx/sites-enabled/
rm /etc/nginx/sites-enabled/default
nginx -t && systemctl reload nginx
```

Nginx Config (Local IP with Self-Signed SSL), Create Self Signed Certificate:

```
mkdir -p /etc/ssl/moodle
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /etc/ssl/moodle/moodle.key \
    -out /etc/ssl/moodle/moodle.crt \
    -subj "/C=ID/ST=Local/L=Local/O=Moodle/OU=IT/CN=(server-ip)"
```

Create nginx site configuration:

```
nano /etc/nginx/sites-available/moodle-ssl
```

Paste to moodle-ssl configuration:

```
    server {
        listen 443 ssl;
        server_name 172.16.x.x;   # replace with your server IP

        ssl_certificate /etc/ssl/moodle/moodle.crt;
        ssl_certificate_key /etc/ssl/moodle/moodle.key;

        root /var/www/moodle;
        index index.php;
        client_max_body_size 100M;

        location / {
            try_files $uri $uri/ /index.php?$query_string;
        }

        location ~ [^/]\.php(/|$) {
            fastcgi_split_path_info ^(.+?\.php)(/.*)$;
            fastcgi_pass unix:/run/php/php8.3-fpm.sock;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
        }
    }
```

Enable configuration file:

```
ln -s /etc/nginx/sites-available/moodle-local /etc/nginx/sites-enabled/
nginx -t && systemctl reload nginx
```

PHP Tuning for Moodle:

```
nano /etc/php/8.3/fpm/php.ini
```

Change the following config, depending your server spec:

```
max_input_vars = 5000
memory_limit = 512M
upload_max_filesize = 128M
```

```
post_max_size = 128M
max_execution_time = 300
```

Restart php-fpm:

```
systemctl restart php8.3-fpm
```

Now open moodle setup page, please access from https, and follow the instruction:

```
https://<your-server-ip>
```

After finish setup, change config.php file

```
nano /var/www/moodle/config.php
```

Disable or remove defult $CFG->wwwroot config, change it with this:

```
if (isset($_SERVER['HTTP_HOST'])) {
    $host = $_SERVER['HTTP_HOST'];
    $scheme = (!empty($_SERVER['HTTPS']) || (isset($_SERVER['HTTP_X_FORWARDED_PROTO']) &&
$_SERVER['HTTP_X_FORWARDED_PROTO'] === 'https'))
        ? 'https' : 'http';
    $CFG->wwwroot = $scheme . '://' . $host;
} else {
    // Fallback
    $CFG->wwwroot = 'http://localhost';
}
```

You can open moodle using http://<server-ip> or https://<server-ip> and https://yourdomain.com

Install ERPNext 15 on Ubuntu 24.04 Proxmox CT

Prerequisites

The below prerequisites are advised in order to get an optimal functionality of ERPNext on your server.

Software Requirements

- Updated Ubuntu 24.04
- Python 3.12+
- A user with sudo privileges
- pip 20+
- MariaDB 10.3.x
- Node.js 18
- yarn 1.22+

Create Frappe user:

```
sudo apt-get update -y && sudo apt-get upgrade -y
sudo adduser frappe
sudo usermod -aG sudo frappe
su frappe
cd /home/frappe
```

Install Git and Python:

```
sudo apt-get install git
sudo apt-get install python3-dev
sudo apt-get install python3-setuptools python3-pip
sudo apt install python3.12-venv
sudo apt install -y python3-pip python3-dev python3-venv build-essential libffi-dev libssl-dev
```

Install MariaDB:

```
sudo apt-get install software-properties-common
sudo apt install mariadb-server
sudo systemctl status mariadb
```

```
sudo mysql_secure_installation
```

```
Enter current password for root: (Enter your SSH root user password)
-Switch to unix_socket authentication [Y/n]: Y
-Change the root password? [Y/n]: Y
It will ask you to set new MySQL root password at this step. This can be different from the
SSH root user password.

-Remove anonymous users? [Y/n] Y
-Disallow root login remotely? [Y/n]: N
This is set as N because we might want to access the database from a remote server for using
business analytics software like Metabase / PowerBI / Tableau, etc.

-Remove test database and access to it? [Y/n]: Y
-Reload privilege tables now? [Y/n]: Y
```

Add the below code block at the bottom of the file:

```
sudo nano /etc/mysql/my.cnf
```

```
[mysqld]
character-set-client-handshake = FALSE
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

[mysql]
default-character-set = utf8mb4
```

Restart sql server

```
sudo service mysql restart
```

Install Redis

```
sudo apt-get install redis-server
```

Install Node.js 20.X package

```
sudo apt install curl
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
source ~/.profile
```

```
nvm install 20
```

Install Yarn

```
sudo apt-get install npm
sudo npm install -g yarn
```

Install wkhtmltopdf

```
sudo apt-get install xvfb libfontconfig wkhtmltopdf
```

Install frappe-bench

```
sudo -H pip3 install frappe-bench --break-system-packages
bench --version
```

Initilise the frappe bench & install frappe latest version

```
bench init frappe-bench --frappe-branch version-15
cd frappe-bench/
chmod -R o+rx /home/frappe
```

Create New Site

```
bench new-site domain.com
```

Get App ERPNext, Payment and HRMS

```
bench get-app erpnext --branch version-15
bench get-app payments
bench get-app hrms
```

Install App in site

```
bench --site domain.com install-app erpnext
```

Disable maintenance mode and Enable scheduler

```
bench --site domain.com set-maintenance-mode off
bench --site domain.com enable-scheduler
```

Install production dependencies

```
sudo apt update
sudo apt install -y supervisor nginx
```

Install ansible from APT (simplest)

```
sudo apt install -y ansible-core
sudo bench setup production frappe
```

regenerate Supervisor config:

```
cd ~/frappe-bench
bench setup supervisor
sudo ln -s `pwd`/config/supervisor.conf /etc/supervisor/conf.d/frappe-bench.conf
```

Reload Supervisor so it reads the new config:

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl status
```

You should finally see ERPNext processes (web, worker, scheduler, etc.).
If they aren’t running, start them:

```
sudo supervisorctl start all
```

FFmpeg + NGINX HTTPS HLS Streaming Server Ubuntu 24.04

In this tutorial, we build a lightweight streaming server on **Ubuntu 24.04** that takes **RTSP video feeds from IP cameras** and publishes them to the web using **HLS (HTTP Live Streaming)**.

We use **FFmpeg** to pull video streams from each camera, remove audio, segment the video into HLS chunks, and keep a rolling archive.
Then, **NGINX with HTTPS** serves those HLS playlists securely so they can be streamed in any modern browser — no plugins required.

This setup is perfect when you want:

- Live CCTV viewing over HTTPS
- Minimal resource usage (video passthrough, no transcoding)
- A clean HLS output that works in browsers + mobile
- Automatic reconnection if the RTSP camera drops
- Simple, scalable directory-based configuration

1 — Update and install dependencies

```
sudo apt update
sudo apt install -y ffmpeg nginx python3-yaml openssl
```

2 — Directory setup

```
sudo mkdir -p /usr/local/ffmpeg-hls
sudo mkdir -p /var/www/html/cctv
sudo chown -R www-data:www-data /var/www/html
```

3 — Camera configuration file

Create file `/usr/local/ffmpeg-hls/cams.yml`

```
- name: cam1
  url: rtsp-url-here
- name: cam2
  url: rtsp-url-here
- name: cam3
  url: rtsp-url-here
```

```
# Add up to cam16 if needed...
```

4 — Auto FFmpeg launcher script

Create file `/usr/local/ffmpeg-hls/start-all-cams.sh`

```
#!/bin/bash
CONFIG_FILE="/usr/local/ffmpeg-hls/cams.yml"
OUTPUT_ROOT="/var/www/html/cctv"
HLS_DURATION=10            # seconds per segment
TOTAL_HISTORY_MIN=10       # total minutes to keep
SEGMENTS_TO_KEEP=$(( (TOTAL_HISTORY_MIN*60) / HLS_DURATION ))

mkdir -p "$OUTPUT_ROOT"

#Generate list of active cams
python3 - <<'PYCODE' > /tmp/camlist.txt
import yaml, sys
data = yaml.safe_load(open("/usr/local/ffmpeg-hls/cams.yml"))
for cam in data:
    print(f"{cam['name']}|{cam['url']}")
PYCODE

#Build array of active cam names
ACTIVE_CAMS=($(awk -F'|' '{print $1}' /tmp/camlist.txt))

#Clean up old folders not in YAML
for dir in "$OUTPUT_ROOT"/*/; do
  [ -d "$dir" ] || continue
  CAM_NAME=$(basename "$dir")
  if [[ ! " ${ACTIVE_CAMS[@]} " =~ " ${CAM_NAME} " ]]; then
    echo "Removing old folder: $dir"
    rm -rf "$dir"
  fi
done

#Start each active camera
while IFS="|" read -r NAME URL; do
    [ -z "$NAME" ] && continue
    mkdir -p "$OUTPUT_ROOT/$NAME"
```

```
    echo "Starting $NAME ..."
    (
      while true; do
        ffmpeg -hide_banner -loglevel warning \
          -rtsp_transport tcp \
          -i "$URL" \
          -fflags +genpts -use_wallclock_as_timestamps 1 \
          -an -c:v copy \
          -f hls \
          -hls_time $HLS_DURATION \
          -hls_list_size $SEGMENTS_TO_KEEP \
          -hls_flags delete_segments+append_list+program_date_time \
          -hls_segment_filename "$OUTPUT_ROOT/$NAME/segment_%05d.ts" \
          "$OUTPUT_ROOT/$NAME/index.m3u8"

        echo "$NAME disconnected, retrying in 5s..."
        sleep 5
      done
    ) &
done < /tmp/camlist.txt

echo "All active camera streams started!"
wait
```

Make it executable:

```
sudo chmod +x /usr/local/ffmpeg-hls/start-all-cams.sh
```

5 — Systemd service

Create file `/etc/systemd/system/cctv-hls.service`

```
[Unit]
Description=FFmpeg HLS Auto Streams (All Cameras)
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/local/ffmpeg-hls/start-all-cams.sh
Restart=always
RestartSec=10
```

```
LimitNOFILE=65535

[Install]
WantedBy=multi-user.target
```

Apply changes

```
sudo nano /usr/local/ffmpeg-hls/start-all-cams.sh
# (paste script above)
sudo chmod +x /usr/local/ffmpeg-hls/start-all-cams.sh
sudo systemctl restart cctv-hls
```

6 — HTTPS setup with NGINX (self-signed)
Generate SSL certificate:

```
sudo mkdir -p /etc/ssl/cctv
sudo openssl req -x509 -nodes -days 3650 -newkey rsa:2048 \
  -keyout /etc/ssl/cctv/cctv.key \
  -out /etc/ssl/cctv/cctv.crt \
  -subj "/CN=cctv.local"
```

Configure NGINX site:
Create file `/etc/nginx/sites-available/cctv`

```
server {
    listen 443 ssl;
    server_name _;

    ssl_certificate     /etc/ssl/cctv/cctv.crt;
    ssl_certificate_key /etc/ssl/cctv/cctv.key;

    root /var/www/html;
    index index.html;

    location /cctv/ {
        add_header Cache-Control no-cache;
        types {
            application/vnd.apple.mpegurl m3u8;
            video/mp2t ts;
        }
        autoindex on;
```

```
        add_header Access-Control-Allow-Origin *;
    }
}

server {
    listen 80;
    server_name _;
    return 301 https://$host$request_uri;
}
```

Enable and reload:

```
sudo ln -s /etc/nginx/sites-available/cctv /etc/nginx/sites-enabled/
sudo rm /etc/nginx/sites-enabled/default
sudo systemctl restart nginx
```

7 — Verify everything

```
sudo systemctl status cctv-hls
```

Then visit:

```
https://<server-ip>/cctv/cam1/index.m3u8
https://<server-ip>/cctv/cam2/index.m3u8
https://<server-ip>/cctv/cam3/index.m3u8
```

Enable HTTPS LibreNMS Ubuntu 24.04

Create self-signed SSL certificate

```
sudo mkdir -p /etc/nginx/ssl
sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 \
  -keyout /etc/nginx/ssl/librenms.key \
  -out /etc/nginx/ssl/librenms.crt
```

Edit you exiting NGINX configuration

```
nano /etc/nginx/conf.d/librenms.conf
```

```
# Redirect all HTTP → HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name librenms.domain.com;
    return 301 https://librenms.domain.com$request_uri;
}

# Main HTTPS server
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    server_name librenms.domain.com;

    ssl_certificate     /etc/nginx/ssl/librenms.crt;
    ssl_certificate_key /etc/nginx/ssl/librenms.key;

    root /opt/librenms/html;
    index index.php;

    access_log /opt/librenms/logs/access_log;
```

```
    error_log  /opt/librenms/logs/error_log;

    charset utf-8;
    gzip on;
    gzip_types text/css application/javascript text/javascript application/x-javascript
image/svg+xml text/plain text/xsd text/xsl text/xml image/x-icon;

    # Main app
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    # API v0
    location /api/v0 {
        try_files $uri $uri/ /api_v0.php?$query_string;
    }

    # PHP-FPM
    location ~ \.php$ {
        include fastcgi.conf;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass unix:/run/php-fpm-librenms.sock;

        # Force HTTPS inside PHP
        fastcgi_param HTTPS on;
        fastcgi_param HTTP_X_FORWARDED_PROTO https;
        fastcgi_param HTTP_X_FORWARDED_PORT 443;
    }

    # Security
    location ~ /\.ht {
        deny all;
    }
}
```

Test & reload NGINX

```
sudo nginx -t
sudo systemctl reload nginx
```

Set LibreNMS to use HTTPS

```
sudo nano /opt/librenms/.env
```

Set APP_URL with your domain name

```
APP_URL=https://librenms.domain.com
```

Clear cache, but login as librenms user

```
su - librenms
php artisan config:clear
php artisan cache:clear
```

Done, Visit LibreNMS over HTTPS