# Viseron NVR

- Install and Use Viseron

# Install and Use Viseron

https://www.youtube.com/embed/wbTQC660vUk

Viseron is fairly new to the open source options for NVRs as of the writing of this article.  It's got a good set of features available, but definitely still young.

## Pros

- Fairly simple to get running in docker.
- Works with RTSP based cameras and MJpeg streams
- Motion Detection
- Object Detection
- Facial Recognition (untested in this article, but claimed on their website)

## Cons

- Uses a yaml configuration file to set up everything (No simple web user interface / form)
- No ONVIF options for camera discovery or setup.
- Lack of feedback on motion / object detection for tuning visually

Viseron offers a fairly simple, clean interface. It doesn't have a lot of bells and whistles when it comes to setup and configuration of cameras, motion detection, or object detection. All of the configuration is done through a yaml configuration file. They do support a "secrets" file for storing your camera iPs, usernames, and passwords separately from the main configuration file, but it is still done via a config file.

Viseron does make the main configuration file available to be edited through a file editor in the web browser, which saves you having to open it in your terminal or a separate text editor.

## The Experience Overall

Fortunately, the configuration file is pretty simple to work with, and they give you a good boiler plate to start from.  The documentation on options available to be used in the configuration file, but not already shown in the boiler plate is a bit light.  Getting my cameras setup wasn't too hard once I understood what values needed to go where.  I was able to use a secrets.yaml file placed in the same folder as the main configuration.yml file to put in my camera IPs, usernames, and passwords, vs using the main configuration file for all of this sensitive detail.

The cameras were generally quickly detected, and showed up in the 'Camera's interface quickly. The system doesn't provide a true live conglomeration view (multiple cameras in a grid or layout). The main cameras view did appear to update about every 7 to 10 seconds, but no more often than

that during my usage and testing.

You can move to a live view per camera, as well as a view for any recordings a camera makes, but you have to dig down a bit to get to "all" recordings.

Overall, Viseron is a great little option for someone looking to monitor a few cameras around their home or business, and get recordings based on object motion, and not just motion in general.

# Installing Viseron

First, you'll want to have docker and docker-compose installed on your system. If you don't already have those follow the section below, and use my simple script designed to help you get them installed with ease.  If you already have them installed, feel free to skip down to the next section on installing Viseron itself.

# Installation via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github.  Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

## Installing Viseron in Docker

Now that you have docker and docker-compose installed, you'll want to make a folder to install Viseron in.  I like to put all of my docker based applications / services inside of a parent docker folder, with the application or service setup as a folder of it's own. To do this in Linux just do

`mkdir -p docker/viseron`

The command above says to create the "docker" folder if it doesn't already exist, but if it does, then just use the one that's there.  Next, it does the same check for the "viseron" folder, and does the same thing.

Now, move into the "viseron" folder with the command:

`cd docker/viseron`

We'll create a new file inside this folder called "docker-compose.yml".  This is a yaml (pronounced "yammel") file that we'll use to define how to get the Viseron application, where to store video recordings, what ports to use, and what the application should be called in docker.

`nano docker-compose.yml`

Copy the code block below, and paste it into the file.

```
version: "2.4"

services:
  viseron:
    image: roflcoopter/viseron:latest
    container_name: viseron
    volumes:
      - {recordings path}:/recordings
      - {config path}:/config
      - /etc/localtime:/etc/localtime:ro
    ports:
      - 8888:8888
    environment:
      - PUID=1000
      - PGID=1000
```

The above compose content comes directly from the Viseron documentation as of the writing of this article.

> NOTE: You should always check the documentation for any updated installation procedures, as it is more likely to be up to date and accurate as time goes by.

In the above content, please make sure to replace only mapped items (volumes, ports) on the left side of the colon ':'. The right side of the colon is what the docker container uses for making the application run, and you should never change those values.  The left side, however, is what your host machine (the machine on which docker is running) uses.  So if you want to reach the Viseron web interface on port 9612, you'll change the port mapping to look like

`- 9612:8888`

Unless you have a need to change the default port mapping, then just let it stay as 8`888:8888`, and know that you'll access the web user interface on port 8888 when it's time.

For the volume mappings, I like to change my volumes to all be inside the current directory. In Linux we do this with the `./` before the path to the location.

So, in my compose file, I changed `{config path}` to be `./data/config`.

For the `{recordings path}`, I would normally also put this in the current folder, but since recordings can grow, and get to be rather large, you may want to use an external NAS or some other mapped network drive. So, here, make sure you enter the proper path to where you want the recordings stored.  Generally a path for a NAS will be found in `/mnt/some/path...` but it's not always the case, so it is important for you to know where your network storage is mounted.

Once you've updated those few details,save the file with CTRL + O, then confirm by pressing Enter, and exit the nano editor with CTRL + X.

Now, you'll want to bring up your Viseron system.  You can do this with the command:

`docker compose up -d` (if you're running an older version of docker-compose, you may need to do `docker-compose up -d` instead.

Give the application time to download, and get started. When you see a 'done' message, you can open your favorite web browser, and go to your host machines IP address and the port you mapped on the left side of the colon.  If you are using the same machine you are sitting in front of, then you can use `localhost` instead of the machine IP if you prefer

I went to `http://192.168.10.60:8541` as I mapped port 8541 on my host to the container port 8888 in my docker-compose.yml file.

You'll see the Viseron app load, and a spinner. The spinner will just continue to spin, because you haven't added any cameras to the configuration file yet.  That's where we'll go next.

## The Viseron Configuration Files

As stated previously, Viseron using yaml configuration files to setup cameras and options for the system.  You'll need two files setup to be more secure, but by default the main configuration file is setup for you.  You can access, edit, and save this file right from the web user interface.  You can even restart the server after making changes and saving them in order to have those changes take

effect.

The main configuration won't look exactly like my example below, but it will be close to what I'm going to show.  Again, on the Viseron documentation, they provide an example configuration file with more detail than what I will have here.  My file is just one to help you get started.

Click the "hamburger" icon on the upper left of the Viseron screen to reveal the navigation panel. Here you can click on the "Configuration" option to go to your default configuration.yml file right in the browser.

It should look something like:

```
ffmpeg:
  camera:
    viseron_camera:
      name: Driveway
      host: 195.196.36.242
      path: /mjpg/video.mjpg
      port: 80
      stream_format: mjpeg
      fps: 25

    viseron_camera2:
      name: Front Door
      host: !secret cam2_ip
      path: /mjpg/video.mjpg
      stream_format: mjpeg
      port: !secret cam2_port
      fps: 15

    viseron_camera3:
      name: Back Yard
      host: !secret cam3_ip
      path: /h265Preview01_sub
      port: !secret cam3_port
      username: !secret cam3_username
      password: !secret cam3_password
      stream_format: rtsp
      fps: 15

darknet:
```

```
  object_detector:
    cameras:
      viseron_camera:
        fps: 5
        scan_on_motion_only: false
        labels:
          - label: person
            confidence: 0.8
            trigger_recorder: true
      viseron_camera2:
        fps: 5
        scan_on_motion_only: true
        labels:
          - label: person
            confidence: 0.65
            trigger_recorder: true
      viseron_camera3:
        fps: 5
        scan_on_motion_only: true
        labels:
          - label: person
            confidence: 0.75
            trigger_recorder: true

mog2:
  motion_detector:
    cameras:
      viseron_camera:
        fps: 5
      viseron_camera2:
        fps: 5

nvr:
  viseron_camera:
  viseron_camera2:
  viseron_camera3:
```

I've left the above configuration setup in a mixed format. I have left the camera 1 connection information setup directly in the configuration file, and you are more than able and welcome to do this if you want to. Camera 2 and Camera 3, however, I have setup to use a second yaml file called "secrets.yaml". The name of this file, and it's location are very important. It's useful, however, as

you can easily share this config across machines, or with others, and not worry about your sensitive camera information getting out.

The "secrets.yaml" file is also fairly easy to setup.  We'll simply create a new file inside the same folder where the main configuration.yml file is located. In this case, it's in our `docker/viseron/data/config` folder.

Once in that folder use the following command to create the new file:

`nano secrets.yaml`

Once open in the text editor start adding the value you want to use in the configuration file, followed by a colon ':', then the actual value.  Put one value set on each line.

Example below:

```
cam2_ip: 192.168.101.15
cam2_port: 443
cam3_username: sammie
cam3_password: e243mlfis85Eals89
cam3_ip: 192.168.101.27
cam3_port: 554
```

Once you've setup your file, save it with CTRL + O, then press Enter to confirm, and use CTRL + X to exit the nano text editor.

Now, in your configuration file, anywhere you want to use a secrets.yaml variable, just put

`!secret <variablename>`

in place of the actual value.  Save your changes, then once saved, click the 'Restart Server' button to check and make sure your settings are working as desired.

There you have it, you now have an NVR system up and running, with optional motion and object detection.  The use of TPM devices like the Google Coral can definitely help with removing the CPU burden from object detection, especially as you add more cameras, but with my testing on 2 cameras, it seemed to do fin with very little additional CPU usage.

## Final Thoughts

One drawback of the Viseron system, is not having a say to setup the automatic removal of old recordings.  This is an overlooked feature in my book, as it removes the concern of running out of storage space at some point, and particularly as you begin to add more cameras to the system.

I think Viseron has a great start, and I am excited to see it move forward with more capabilities and features as time goes on. I truly hope they will be successful, and add another amazing open

source option to the NVR landscape.

## Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:
https://www.patreon.com/awesomeopensource