Installation in Home Assistant

Beschreibung:

Ein Videoüberwachungssystem , Installation in Home Assistant direkt.

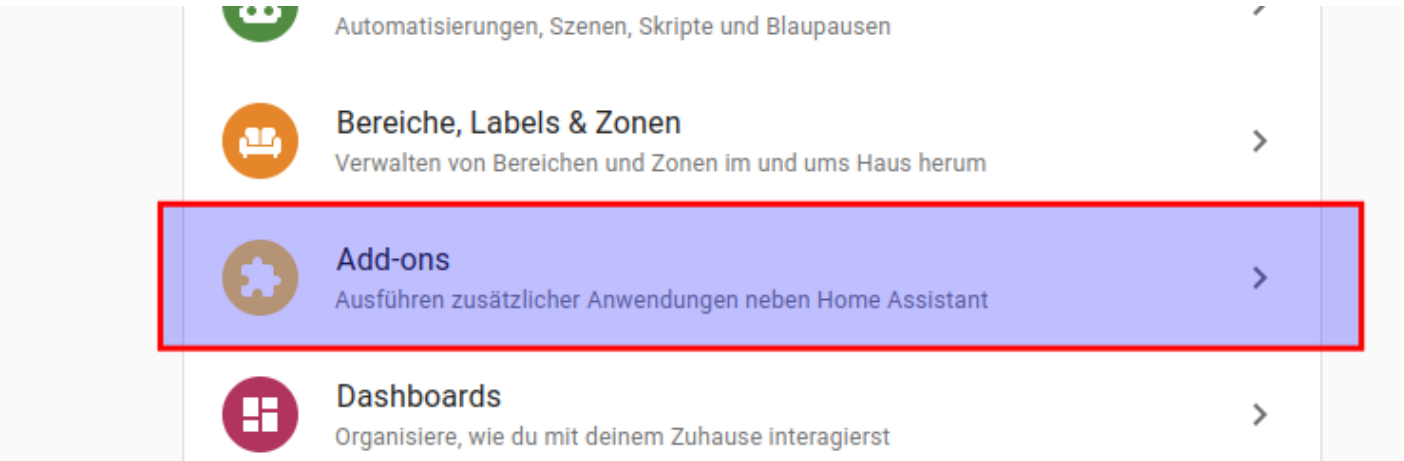
Vorraussetzungen:

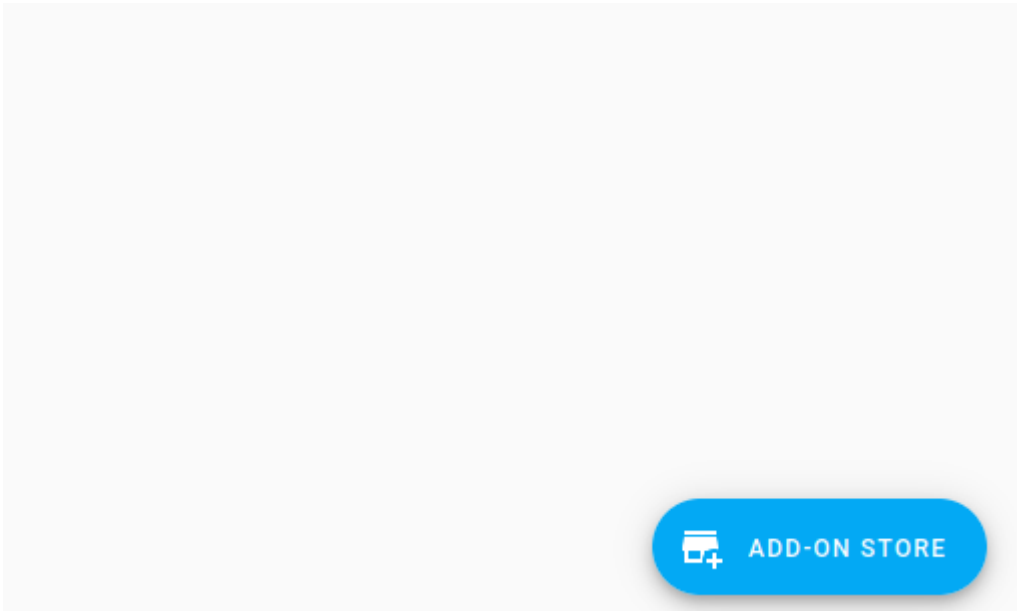
Ein MQTT Broker
Das Fileeditor Addon

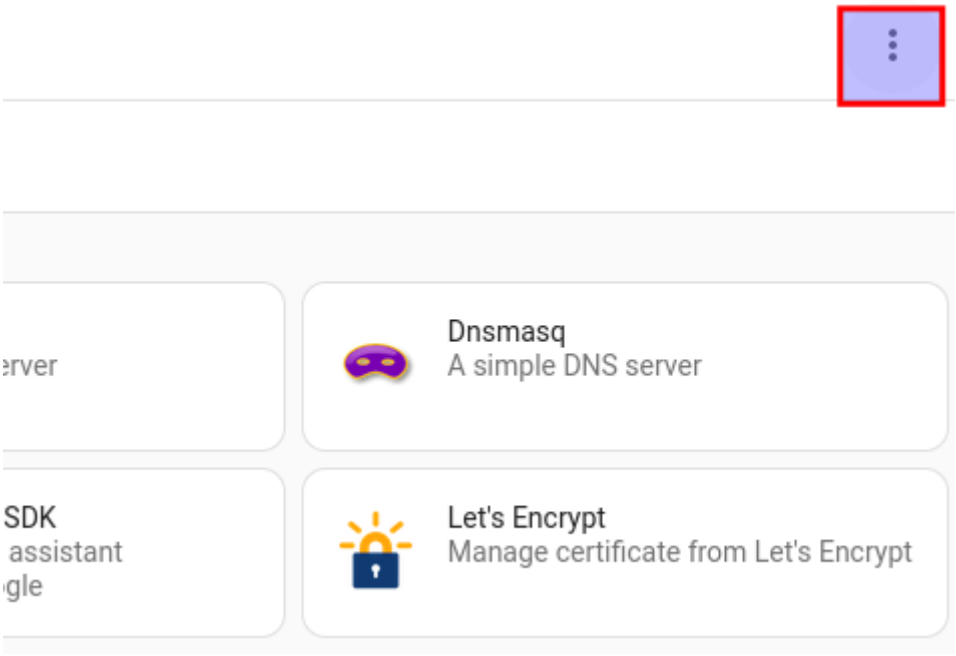
Installation aus dem Addon Store:

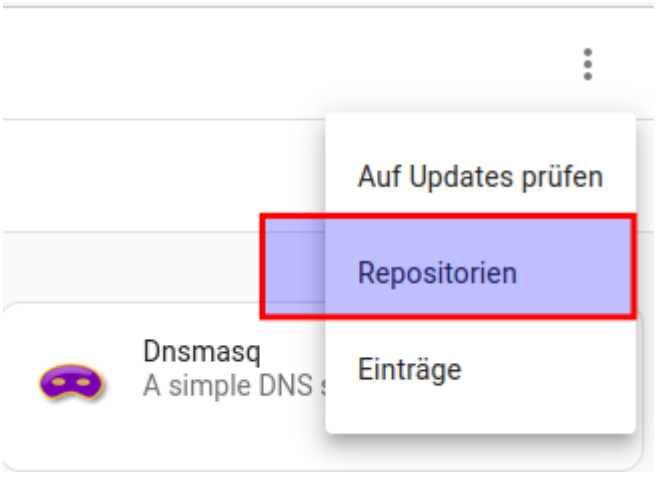
Dazu auf
Einstellungen -> Addons -> Addon Store

Dort Repository hinzufügen (3 Punkte oben rechts)





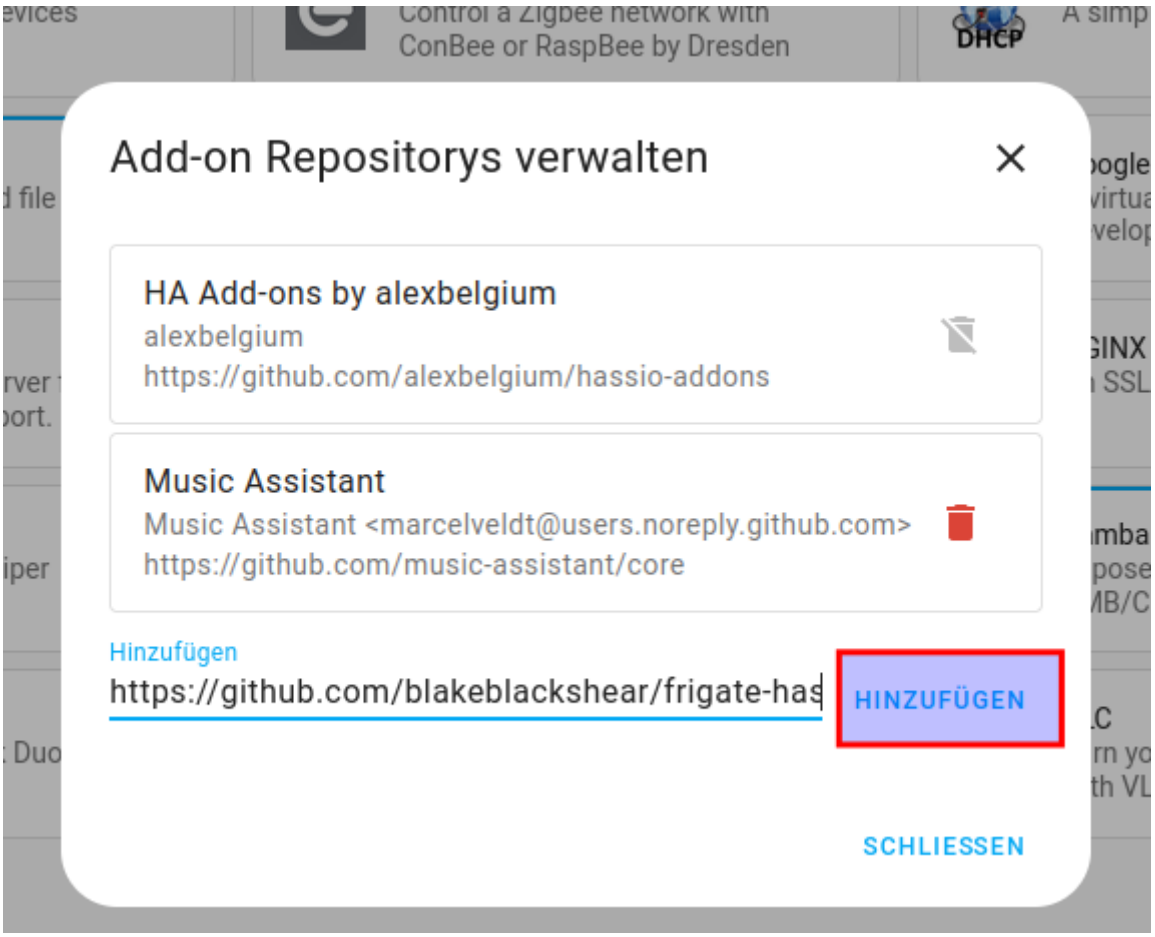




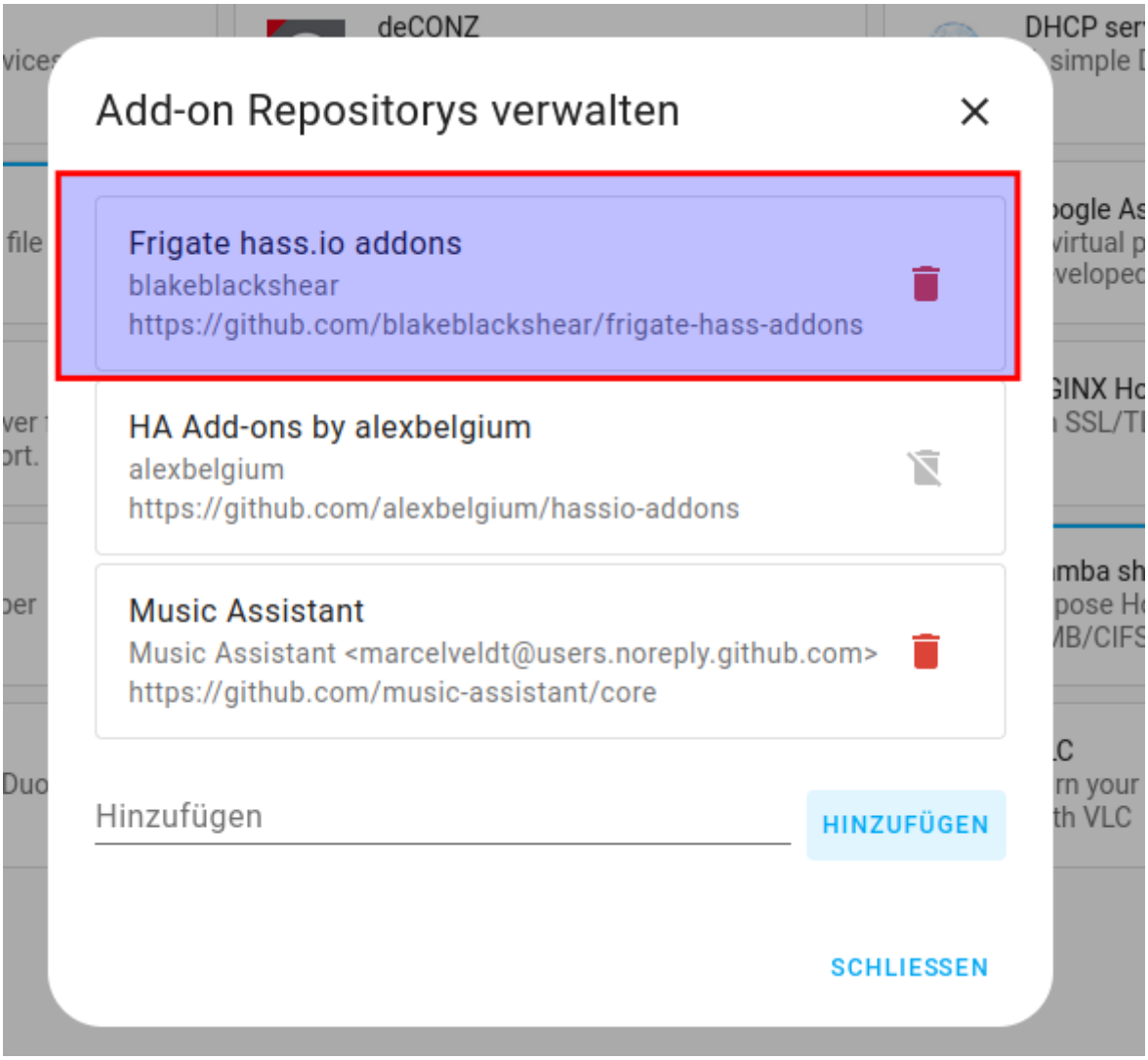
Dort folgende Repo hinzufühgen

https://github.com/blakeblackshear/frigate-hass-addons

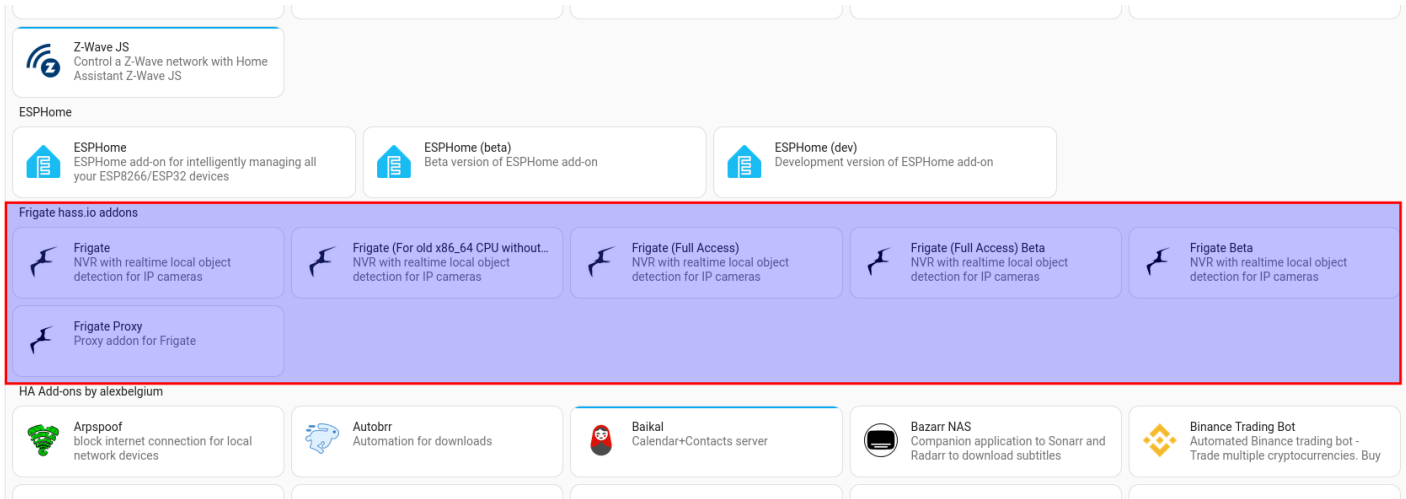
und auf hinzufügen klciken



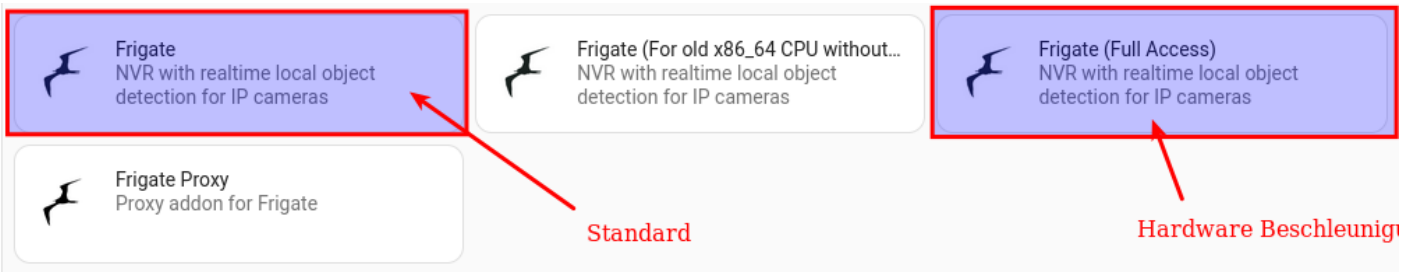
Nun ist die Repo drin und auf schließen klicken, danach den browser mit F5 refresh durchführen



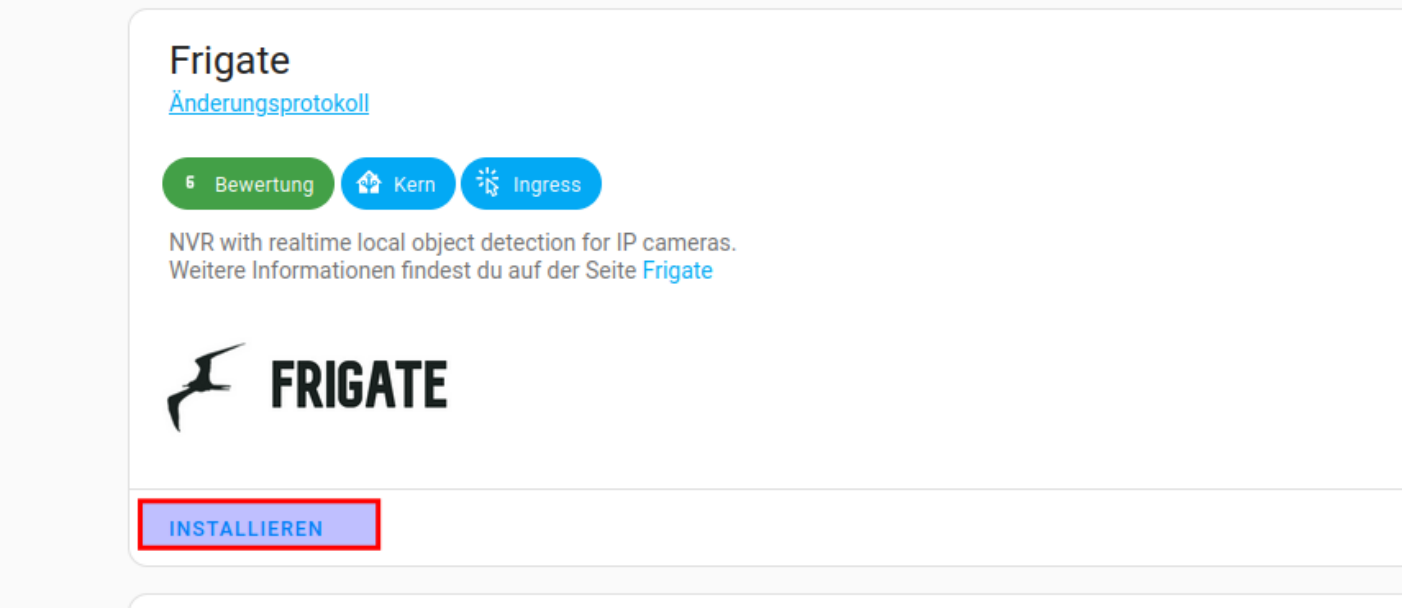
Nun haben wir in unserem Store die frigate Varianten zur Verfügung



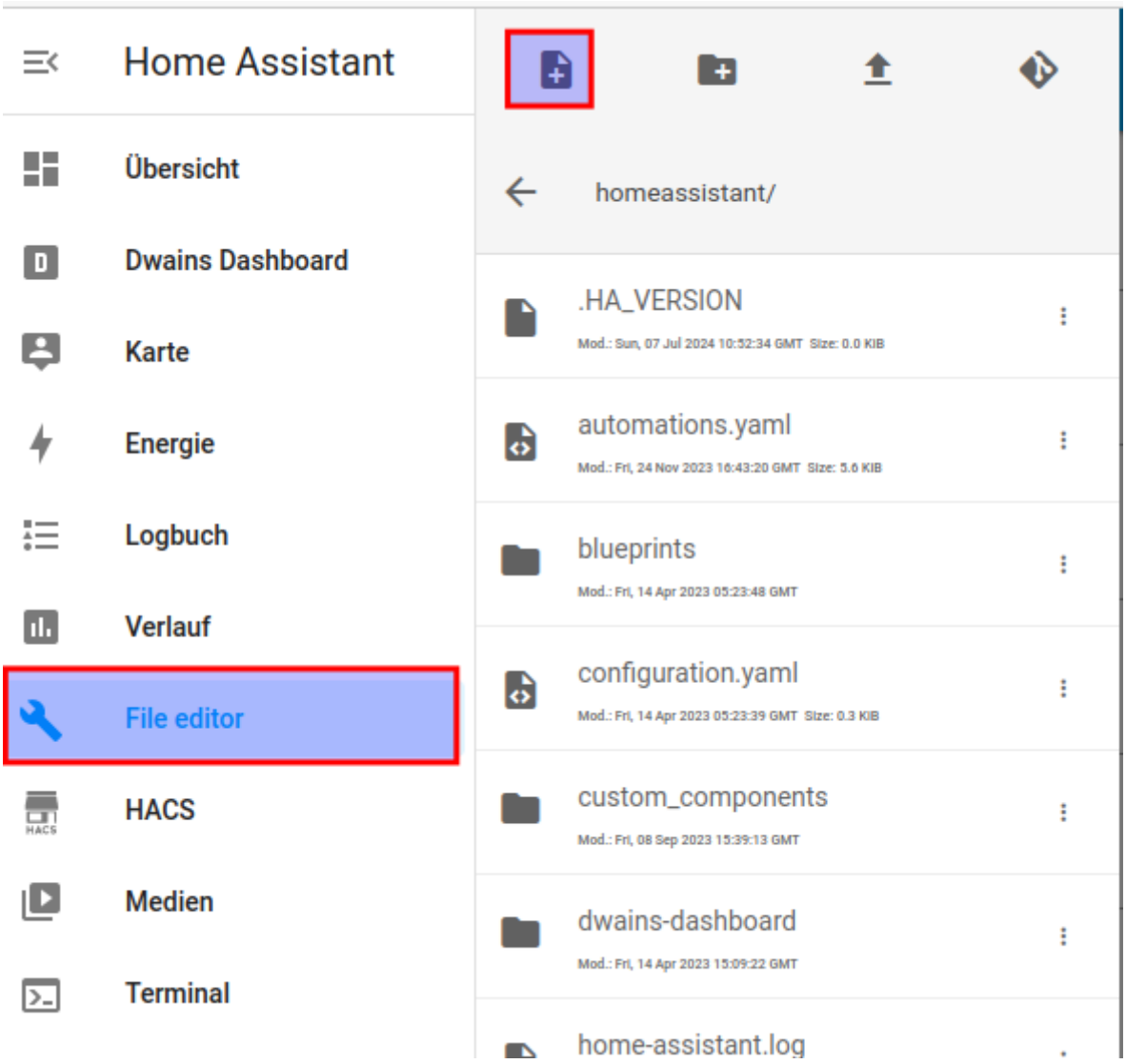
Wir wäheln die Frigate Standard Variante, Full Access wäre für Kameras mit Hardware Beschleunigung.
Habt Ihr Kameras mit Hardware Beschleunigung nehmt Ihr Full Access



Nun auf installieren klicken, danach das Addon noch **NICHT!!** starten



Nun im File Editor auf Browse / Durchsuchen gehen gehen und eine neue Datei anlegen mit dem namen
frigate.yml



Hier mal ein Inhalt den man als Vorlage nehmen und erweitern kann:

```
mqtt:
  enabled: True
  host: 172.16.0.14
  user: smarthome
  password: smarthome
  topic_prefix: frigate
```

```
  # Optional: client id (default: shown below)
  # WARNING: must be unique if you are running multiple instances
  client_id: frigate
  # Optional: interval in seconds for publishing stats (default: shown below)
  stats_interval: 15

#immer die frigatre datenbank da hinlegen wo auch die medien liegen, denn sollte mal das
netzlaufwerkabbrechen,
#haben wir keine inkonsistente Datenbank zu den videos
database:
  path: /media/frigate/frigate.db

# Optional: Detectors configuration. Defaults to a single CPU detector
detectors:
  # Required: name of the detector
  detector_name:
    # Required: type of the detector
    # Frigate provided types include 'cpu', 'edgetpu', and 'openvino' (default: shown below)
    # Additional detector types can also be plugged in.
    # Detectors may require additional configuration.
    # Refer to the Detectors configuration page for more information.
    type: cpu
#Hardware Beschleunigung aktivieren, wenn eine GPU vorhanden für hw Beschleunigung dann den Kommentar
bei hwaccel_args entfernen
ffmpeg:
  #hwaccel_args: -c:v h264_qsv
  hwaccel_args: preset-intel-vaapi
  #hwaccel_args: preset-intel-qsv-h264
  #hwaccel_args: preset-rpi-64-h264
  #hwaccel_args: preset-rpi-64-h265

birdseye:
  enabled: True
  restream: True
  mode: objects

objects:
  track:
    - person
    - car
```

```
#ich hab den Filter rausgenommen, aber wer möchte das die Objekterkennung nur über einen bestimmten Bereich geht, kann diesen Definieren.
#Allerdings wäre es dafür einfacher Zones für die Kameras in der UI zu erstellen.
  #filters:
   # person:
     # min_area: 5000
     # max_area: 100000



go2rtc:
  streams:
    Treppenhaus:
      - rtsp://smarthome:Smarthome&More@192.168.101.185:554/h264Preview_01_main#video=copy
    Treppenhaus_sub:
      - rtsp://smarthome:Smarthome&More@192.168.101.185:554/h264Preview_01_sub#video=copy
  #enable webrtc für flüssigeres bild in der lovlace Karte
  webrtc:
    candidates:
      - 127.0.0.1:8555
      - stun:8555

cameras:
  Treppenhaus_CAM:
    ffmpeg:
      inputs:
        - path: rtsp://127.0.0.1:8554/Treppenhaus
          input_args: preset-rtsp-restream
          roles:
            - record
        - path: rtsp://127.0.0.1:8554/Treppenhaus_sub
          input_args: preset-rtsp-restream
          roles:
            - detect
    detect:
      width: 1280 #Hier unbedingt die reale Auslösung des Streams wählen
      height: 720 #Hier unbedingt die reale Auslösung des Streams wählen
      fps: 5
    snapshots:
      enabled: True
    record:
```

```
  enabled: True
  retain:
    mode: motion #nimmt nur auf wenn Bewegung erkannt wurde und nicht komplett durchgehend,
          #dies ist Lösung für die Cach Warning wenn zu wenig RAM vorhanden ist.
          #Und warum soll man die Festplatte unnötig volllaufen lassen, auch wenn nur für 1 Tag wäre
    days: 1 #ich möchte ja nur die clips haben wo auch objekte erkannt werden.
           #Es wird aber in frigate tatsächlich die ganze zeit aufgenommen.
           #Wir brauchen keine Komplettaufnahmen länger als ein Tag, also noch einem Tag wegschmeißen.
           #Denn den Record brauchen wir leider, damit frigate die Clips daraus schneiden kann.
  detections:
    pre_capture: 30        # Sekunden vor dem Ereignis mitschneiden
    post_capture: 60       # Sekunden nach dem Ereignis mitschneiden
    retain:
      default: 14
```

Die Config Werte:

MQTT:

Wir ändern den Username und kennwort bei MQTT auf unseren MQTT Benutzer und Kennwort ab.
Den Host ändern wir auf die IP Adresse unseren Home Assistanten ab.

Den rest von MQTT lassen wir so.

Detectors:

Hier type: cpu nehmen oder wer ein google corel ai Stick hat stellt folgendes ein:
Birdeye view

FFMPEG:

> **Hinweis:** Darauf achten das die Kamera Streams den gleichen Codec haben den die HW Beschleunigung auch kann!!!

- `#hwaccel_args: -c:v h264_qsv`
  Quick Sync direkt erzwingen (H.264). Das ist ein **Roh-ffmpeg-Flag** und allein meist **nicht ausreichend** (eigentlich bräuchtest du zusätzlich `-hwaccel qsv`, Device usw.). In Frigate lieber die QSV-Preset-Zeile unten nutzen. Nur für Intel-iGPU, nur H.264.
- `#hwaccel_args: preset-intel-vaapi`
  Intel iGPU über VAAPI (Linux). Solide Allround-Variante, funktioniert für **H.264 und oft auch H.265** je nach iGPU.
  Voraussetzungen: Linux, `/dev/dri` in den Container durchreichen. Gut, wenn du H.265 im

Main-Stream hast, aber trotzdem HW-Decode willst.

- `#hwaccel_args: preset-intel-qsv-h264`
  Intel Quick Sync (QSV) speziell für H.264. Sehr effizient, geringe CPU-Last. Voraussetzungen: Intel-iGPU + `/dev/dri`. **Nur verwenden, wenn der zu decodierende Stream H.264 ist** (für H.265 gibt es ein anderes Preset, nicht dieses).
- `#hwaccel_args: preset-rpi-64-h264` und bei 265 genau umgekehrt, kein h264
  Raspberry Pi (64-bit) via V4L2 M2M, H.264-Decode. Für Pi 4/5 mit 64-bit OS. **Kein H.265-HW-Decode**, also Sub-Stream in der Kamera auf H.264 stellen. Container braucht Zugriff auf die Video-Devices (bei HA-Addon i. d. R. schon so).
- `#hwaccel_args: preset-rpi-64-h265`
  Raspberry Pi (64-bit) via V4L2 M2M, H.265-Decode. Für Pi 4/5 mit 64-bit OS. **Kein H.264-HW-Decode**, also Sub-Stream in der Kamera auf H.265 stellen. Container braucht Zugriff auf die Video-Devices (bei HA-Addon i. d. R. schon so).
-

Object:

Wir wollen Personen und Autos erkennen.

go2rtc:

hier wird eine Verbindung zur Kamera aufgebaut. Wenn eine Kamera auch einen Substream hat, für hohe und niedrige auflösung wählen.
Main ist immer hochauslösend, sub als Previe.
Kann die Cam nur ein Stream.
Diesen bei beie eintargen.

Cameras:

Dort den Namen der Camere eintragen wie er auch wirklcih erscheinen soll.
Hier erstellen wir ein neuen RTSP Stream auf localhost.
Dieser verlinkt auf den go2rtc. Vorteil eure Kamera die in go2rtc angelegt ist, unterstützt nur einen Stream gleichzeitg zum abruf, nun könnt Ihr den input von cameras auch für externe Anwendungen nutzen. Denn in der externen Anwedung gebt ihr den Camera input an. Und dann greift Home Assistant auf die Cam im go2rtc zu.

record:

In Frigate gibt’s im `record:`-Block mehrere Untersektionen, die unterschiedliche Arten von Videoaufzeichnungen betreffen.
Die Namen **`alerts`**, **`detections`** und **`events`** tauchen dort auf, um gezielt **unterschiedliche Aufnahmetypen** separat zu konfigurieren.

1 **`alerts`**

- Bezieht sich auf Clips, die durch einen **Alarm** ausgelöst werden (z. B. wenn ein Objekt mit einer sehr hohen Erkennungssicherheit erkannt wird).
- Entsteht meist, wenn `review: true` aktiviert ist oder ein **MQTT-/API-Trigger** für einen Alarm kommt.
- Wird eher selten genutzt, außer du hast externe Systeme, die Frigate “Alarm!” rufen lassen.

---

2 **detections**

- Steht für **Videoabschnitte während einer aktiven Objekterkennung**.
- Wenn Frigate etwas erkennt (Person, Auto, Tier) und die `detect:`-Parameter erfüllt sind, speichert dieser Modus die Clips nur für diese Zeit.
- Du kannst hier eigene `pre_capture`- und `post_capture`-Zeiten festlegen, die nur für “Erkennungsclips” gelten.
- Hier stellst du typischerweise deine gewünschten 30 Sekunden vorher und 60 Sekunden nachher ein.

Praxis-Tipp:
Wenn du einfach nur möchtest, dass **jeder erkannte Vorfall** mit Puffer aufgenommen wird, reicht es, nur **`events`** in `record:` zu konfigurieren – die anderen brauchst du nicht zwingend.

Mögliche Parameter in **`alerts`** / **`detections`** / **`events`**

Parameter	Typ	Bedeutung
`pre_capture`	Sekunden (float/int)	Anzahl Sekunden Video, die **vor dem Ereignis** gespeichert werden sollen.
`post_capture`	Sekunden (float/int)	Anzahl Sekunden Video, die **nach dem Ereignis** gespeichert werden sollen.
`retain`	Block	Einstellungen, wie lange diese Aufnahmen behalten werden.
`retain.default`	Tage (int)	Standardanzahl Tage, die diese Aufnahmen behalten werden. Alte syntax days

Parameter	Typ	Bedeutung
retain.objects	Mapping	Abweichende Aufbewahrungsdauer pro Objektklasse (z. B. `"person": 30`). egal was in default steht, wird eine Person erkannt, 30 Tage aufbewahren. Beispiel: retain: default: 14 # Standard: 14 Tage behalten objects: person: 30 # Clips mit Personen: 30 Tage behalten car: 7 # Clips mit Autos: 7 Tage behalten Ein Event ohne Person → 14 Tage behalten Ein Event mit Person → 30 Tage behalten Ein Event mit Auto, aber ohne Person → 7 Tage behalten Ein Event mit Auto und Person → nimmt den höchsten Wert (30 Tage) □□ Das ist praktisch, wenn du z. B. Fahrzeuge oder Tiere kürzer behalten willst, aber Personenaufnahmen länger archivieren möchtest.
enabled	true/false	Aktiviert/deaktiviert die Aufzeichnung für diese Untersektion.
objects	Liste	Beschränkt Aufnahmen auf bestimmte Objekte (z. B. `["person", "car"]`).
required_zones	Liste	Aufnahmen nur, wenn das Objekt in einer dieser definierten Zonen erkannt wird.
min_score	Zahl (0–1)	Mindest-Score für die Erkennung, damit ein Clip gespeichert wird.

Parameter	Typ	Bedeutung
active_objects	Liste	• Wenn ein Objekt erkannt wird, startet ein Event. • Das Event bleibt **solange aktiv**, wie mindestens eines der festgelegten active_objects noch im Bild ist. • Erst wenn kein active_object mehr sichtbar ist **und** der post_capture-Puffer abgelaufen ist, wird der Clip abgeschlossen. • Praktisch, um zu verhindern, dass eine Aufnahme in mehrere kurze Clips gesplittet wird, wenn ein Objekt kurz verschwindet und wiederkommt. record: enabled: true detections: pre_capture: 30 post_capture: 60 active_objects: - person - car retain: default: 14 ➡ In diesem Beispiel: Event startet, wenn Person oder Auto erkannt wird. Solange eines dieser beiden Objekte im Bild ist, bleibt das Event aktiv. Wenn beide weg sind, läuft noch post_capture (hier 60 Sekunden) und der Clip wird dann abgeschlossen. **Wichtig:** • Hat **keinen Einfluss** auf alerts oder detections, sondern nur auf events. • Wenn du Objekte wie "dog" nicht in active_objects aufnimmst, können diese zwar erkannt werden, halten das Event aber **nicht** offen.

Parameter	Typ	Bedeutung
`mode`	String	In Frigate steuert der `mode`-Parameter unter `record:` **wann und wie Aufnahmen gespeichert werden**. Er beeinflusst **die komplette Recorder-Logik** – also, ob Clips ständig oder nur bei Erkennung gespeichert werden. ## Mögliche Werte von `mode` **Wert** / **Bedeutung** / **Wann wird aufgenommen?** `all` / Daueraufnahme / 24/7-Aufzeichnung, unabhängig von Bewegung oder Erkennung. Events werden zusätzlich markiert, aber der komplette Stream wird gespeichert. `motion` / Nur bei Bewegung / Speichert, sobald Bewegung erkannt wird (auch wenn kein Objekt erkannt wird). Spart Speicherplatz, kann aber Ereignisse abschneiden, wenn Bewegung spät erkannt wird. `active_objects` / Nur bei aktiven Objekten / Speichert, wenn ein erkennbares Objekt (aus deiner Objektliste) aktiv im Bild ist. Noch sparsamer, aber Risiko, dass Aufnahmen fehlen, wenn Erkennung mal aussetzt. `never` *(oder einfach weglassen)* / Keine dauerhafte Aufzeichnung / Es gibt nur Event-Clips, aber keine kontinuierliche Speicherung. Beispiel □□ Allerdings ist das hier jetzt reine Bewegung, denn in Events fehlt die Objekt Erkennung nur als info □□ record: enabled: true mode: motion detections: pre_capture: 30 post_capture: 60 retain: default: 14 ➡ In diesem Beispiel: Frigate speichert Clips nur bei Bewegung. Dank pre_capture werden auch 30 Sekunden vor der Bewegung mitgeschnitten. Ereignisse bleiben 14 Tage gespeichert. □□ Praxis-Tipp:

anderen Speicherort festlegen:

frigate stopp

Alle Daten werden im /media/frigate Verzeichnis gespeichert.
Damit die Datenbank aber nicht im Config Verzeichnis liegt:

An die Konfig anfügen.

```
database:
  path: /config/frigate.db

ändern zu bzw hinzufügen wenn noch kein Database: Eintrag vorhanden

database:
  path: /media/frigate/frigate.db
```

Nun einen neuen Netzwerkspeicher anlegen mit dem Namen frigate.
Dieser wird dann mit dem Namen frigate in Verzeichnis /media gemountet.
Somit ist der Pfad dann Media frigate
Möchte man seine alte Daten noch halten dann benennen wir vor dem erstellen des Speichers , das Verzeichnis frigate in frigate_tmp um
Nachdem der Netzwerkwerk speicher angehängt wurde verschieben wir den Inhalt aus frigate_tmp in frigate.
Dann starten wir das Addon wieder.

Das umbenennen kann man über das Addon Terminal in der Sidebar gemacht werden oder über ein Filebrowser Addon in der Sidebar
Fertig

Zeitstempel:

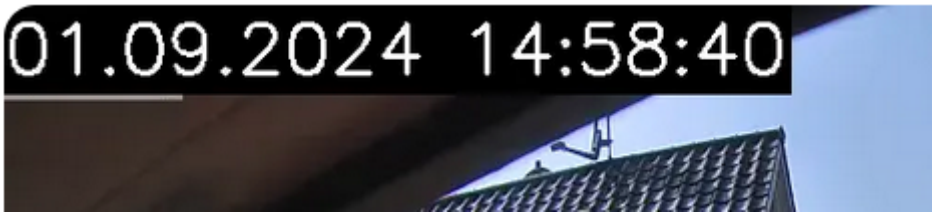
In Frigate hat man nur die Möglichkeit Zeitstempel in die Bilder (Snapshots) einzubetten.
Man kan das auch in die Recording über ffmpeg machen.
Das ist aber sehr Ressourcen hungrig, weil das Video dazu neu encodiert wird.
Nut mit Video Accelrator zu empfehelen.
Mein Vorschlag wäre, ob die Kamera selbst die Möglichkeit hat einen Zeitstempel hinzuzufügen.

Für Unifi Kameras, kann man den Zeitstempel in der GUI config.

Reicht einem die Zeit im Snapshot, dann kann man einen neuen Abschnitt in der Konfig (ich habe diesen unter go2rtc erstellt), dieses hinzufügen:
Hier ändere Ich das Format von US auf 01.12.2024 13:15:45 in der Farbe weiß oben links

```
...
timestamp_style:
  # Optional: Position of the timestamp (default: shown below)
  #           "tl" (top left), "tr" (top right), "bl" (bottom left), "br" (bottom right)
  position: "tl"
  # Optional: Format specifier conform to the Python package "datetime" (default: shown below)
  #           Additional Examples:
  #             german: "%d.%m.%Y %H:%M:%S"
  format: "%d.%m.%Y %H:%M:%S"
  # Optional: Color of font
  color:
    # All Required when color is specified (default: shown below)
    red: 255
    green: 255
    blue: 255
  # Optional: Line thickness of font (default: shown below)
  thickness: 2
  # Optional: Effect of lettering (default: shown below)
  #           None (No effect),
  #           "solid" (solid background in inverse color of font)
  #           "shadow" (shadow for font)
  effect: "solid"
...
```

So sähe das dann aus, aber leider nur im Bild (Snapshots) und nicht in den Auffnahmen



Schlechte Internetverbindung:

Wenn die Internetverbindung einfach zu schlecht ist 128 KB pro Sekunde (1000 Kbit / 1 MBit (DSL 1000 Leitung) Up/download

Sollte die Internetverbindung für ein Flüssiges wiedergeben von Aufnahmen nicht funktionieren.
Weil 4K über 1 Mbit gar nicht geht.
Dann kann man die Gleiche Kamera nochmal anlegen nur zum Beispiel mit den Namen Mobil und dort bei record den Substream mit einer niedrigen Auflösung wählen.
Nun werden zwei Aufnahmen erstellt einmal in 4K bzw höchste Auflösung was die Kamera Kann und einmal in der niedrichsten.
Wazu das ganze?
Ist man im Urlaub möchte man trotzdem auf die Aufnahmen zugreifen können, außerdem schont es zusätzlich das Datenvolumen.
Braucht man zum Beispiel wegen eines Einbruchs dann das Video in 4K kann man dieses in 4K herunterladen.
Vorher wäre es aber interessant sich überhaupt bequem das Einbruchvideo anzuschauen ohne es runter zu laden. Dafür ist dann die Aufnahme mit dem substream in niedriger Auflösung völlig ausreichend

Beispiel Konfig:

```
#Im Go to rtc Teil werden die Streams definiert.
#Die Stremas sind die URLs zu den Jeweiligen Streamd der Kamera(s).
#Wir haben 3 Stück.
#Innenhof = 4K
#Innenhof_Medium = 720p
#Innenhof_sub = DVD qualität
go2rtc:
  streams:
    Innenhof:
      - rtsp://user:pass@192.168.178.68:554/s0#video=copy
    Innenhof_medium:
      - rtsp://user:pass@192.168.178.68:554/s2#video=copy
    Innenhof_sub:
      - rtsp://user:pass@192.168.178.68:554/s1#video=copy
#hat jetzt nichts mit den cams zu tun, aber ich hab den webrtc mal mit reingepackt
  webrtc:
    candidates:
      - 127.0.0.1:8555
      - stun:8555

#Un in dem Camera Teil, die Cams mit detect record und livebild deklariert.
#Einfach doppelt mit anderen namen und dem entsprechendendes Streams.
```

Der detect Stream bleibt bei allen gleich, denn da braucht man ja kein Internet für

```
cameras:
  Innenhof_CAM:
    ffmpeg:
      inputs:
        - path: rtsp://127.0.0.1:8554/Innenhof
          roles:
            - record
        - path: rtsp://127.0.0.1:8554/Innenhof_medium
          roles:
            - detect
    live:
      stream_name: Innenhof_medium
      quality: 8
    detect:
      width: 1280  #Hier unbedingt die reale Auslösung des Streams wählen
      height: 720  #Hier unbedingt die reale Auslösung des Streams wählen
      fps: 5 # in der Regel reichen 5 Frames
    snapshots:
      enabled: true
    record:
      enabled: true
      retain:
        default: 1
        mode: motion
    detections:
      retain:
        default: 30
        mode: active_objects
  Innenhof_CAM_Mobil:
    ffmpeg:
      inputs:
        - path: rtsp://127.0.0.1:8554/Innenhof_medium
          input_args: preset-rtsp-generic
          roles:
            - record
        - path: rtsp://127.0.0.1:8554/Innenhof_medium
          input_args: preset-rtsp-generic
```

```
      roles:
        - detect
  live:
    stream_name: Innenhof_medium
    quality: 8
  detect:
    width: 1280 #Hier unbedingt die reale Auslösung des Streams wählen
    height: 720 #Hier unbedingt die reale Auslösung des Streams wählen
    fps: 5
  snapshots:
    enabled: true
  record:
    enabled: true
    retain:
      default: 1
      mode: motion
    detections:
      retain:
        default: 30
        mode: active_objects
```

Full Reference Konfig:

Will man alle möglichen Optionen wissen, dann unter

https://docs.frigate.video/configuration/reference/

nachschauen

Beispiel konfig small:

Noch ne kleine Erinnerung an die HW Beschleunigung:

FFMPEG:

- `#hwaccel_args: -c:v h264_qsv`
  Quick Sync direkt erzwingen (H.264). Das ist ein **Roh-ffmpeg-Flag** und allein meist **nicht ausreichend** (eigentlich bräuchtest du zusätzlich `-hwaccel qsv`, Device usw.). In Frigate lieber die QSV-Preset-Zeile unten nutzen. Nur für Intel-iGPU, nur H.264.
- `#hwaccel_args: preset-intel-vaapi`
  Intel iGPU über VAAPI (Linux). Solide Allround-Variante, funktioniert für **H.264 und oft auch H.265** je nach iGPU.
  Voraussetzungen: Linux, `/dev/dri` in den Container durchreichen. Gut, wenn du H.265 im Main-Stream hast, aber trotzdem HW-Decode willst.
- `#hwaccel_args: preset-intel-qsv-h264`
  Intel Quick Sync (QSV) speziell für H.264. Sehr effizient, geringe CPU-Last.
  Voraussetzungen: Intel-iGPU + `/dev/dri`. **Nur verwenden, wenn der zu decodierende Stream H.264 ist** (für H.265 gibt es ein anderes Preset, nicht dieses).
- `#hwaccel_args: preset-rpi-64-h264`
  Raspberry Pi (64-bit) via V4L2 M2M, H.264-Decode. Für Pi 4/5 mit 64-bit OS. **Kein H.265-HW-Decode**, also Sub-Stream in der Kamera auf H.264 stellen. Container braucht Zugriff auf die Video-Devices (bei HA-Addon i. d. R. schon so).
- `#hwaccel_args: preset-rpi-64-h265`
  Raspberry Pi (64-bit) via V4L2 M2M, H.265-Decode. Für Pi 4/5 mit 64-bit OS. **Kein H.264-HW-Decode**, also Sub-Stream in der Kamera auf H.265 stellen. Container braucht Zugriff auf die Video-Devices (bei HA-Addon i. d. R. schon so).

Config:

```
#mqtt


mqtt:
  enabled: true
  host: 192.168.178.xx
  user: username
  password: password
  topic_prefix: frigate
  # Optional: client id (default: shown below)
  # WARNING: must be unique if you are running multiple instances
  client_id: frigate
  # Optional: interval in seconds for publishing stats (default: shown below)
  stats_interval: 15



  # Optional: Detectors configuration. Defaults to a single CPU detector
detectors:
```

```
  # Required: name of the detector
  coral1:
    # Required: type of the detector
    # Frigate provided types include 'cpu', 'edgetpu', and 'openvino' (default: shown below)
    # Additional detector types can also be plugged in.
    # Detectors may require additional configuration.
    # Refer to the Detectors configuration page for more information.
    type: edgetpu



    #Hardware Beschleunigung aktivieren, wenn eine GPU vorhanden für hw Beschleunigung dann den
Kommentar bei hwaccel_args entfernen
ffmpeg:
  #hwaccel_args: -c:v h264_qsv
  #hwaccel_args: preset-intel-vaapi
  #hwaccel_args: preset-intel-qsv-h264
  #hwaccel_args: preset-rpi-64-h264
  #hwaccel_args: preset-rpi-64-h264

birdseye:
  enabled: true
  restream: true
  mode: objects

objects:
  track:
    - person
    - car
    - dog
    - cat

#video=copy ist wichtig damit das Video nicht neu transkodiert wird
go2rtc:
  streams:
    parkplatz:
      - rtsp://admin:password@192.168.178.xx/0#video=copy
    parkplatz_sub:
      - rtsp://admin:password@192.168.178.xx/1#video=copy

  #in dem Streams abschnitt kann man über die RTSP Eigenschaft den Timeout und ein Buffer einstellen.
```

```
  #Ich habe diesen nicht benutzt, kann man aber aktivieren.
  #rtsp:
   # mediatimeout: 10s  # Erhöht die Timeout-Zeit für den RTSP-Stream
  #buffer:
   # size: 8388608 # Puffergröße in Bytes, hier etwa 8 MB
  #enable webrtc für flüssigeres bild in der lovlace Karte
  webrtc:
    candidates:
      - 127.0.0.1:8555
      - stun:8555

cameras:
  parkplatz_CAM:
    ffmpeg:
      inputs:
        #full resolution for recording
        - path: rtsp://127.0.0.1:8554/parkplatz
          input_args: preset-rtsp-generic
          roles:
            - record
        #low resoultion for detect
        - path: rtsp://127.0.0.1:8554/parkplatz_sub
          input_args: preset-rtsp-generic
          roles:
            - detect
    live:
      stream_name: parkplatz
    detect:
      width: 640 #Hier unbedingt die reale Auslösung des Streams wählen
      height: 480 #Hier unbedingt die reale Auslösung des Streams wählen
      fps: 5
    snapshots:
      enabled: true
    record:
      enabled: true
      retain:
        mode: motion #nimmt nur auf wenn Bewegung erkannt wurde und nicht komplett durchgehend, dies ist Lösung für die Cach Warning wenn zu wenig RAM vorhanden ist
        default: 1 #ich möchte ja nur die clips haben wo auch objekte erkannt werden. Es wird aber in frigate tatsächlich die ganze zeit aufgenommen.
```

```
      #das brauch ich aber nicht, also noch einem Tag wegschmeißen. Denn den Record brauchen wir leider, damit frigate die Clips daraus schneiden kann.
    alerts:
      retain:
        default: 14 #Tage wo alerts Aufbewart werden sollen
    detections:
      pre_capture: 30        # Sekunden vor dem Ereignis mitschneiden
      post_capture: 60       # Sekunden nach dem Ereignis mitschneiden
      retain:
        default: 14 #Tage Events Aufbewart werden sollen

version: 0.15-1
```

Version #32
Erstellt: 10 Juli 2024 18:53:33 von Admin
Zuletzt aktualisiert: 15 August 2025 07:17:02 von Admin