Creating docker image of Kria-App

The Kria-Ubuntu requires docker based application to run with DPU and other hardware components like VCU. While creating docker based application, it is easy to run inside Ubuntu and we can setup all the necessary drivers and dependencies inside docker itself.
This document is on how to create “Kria-App” based on docker as like the SmartCam and other.

Get the docker files for creating new custom image by cloning `kria-docker` repository available at: https://github.com/Xilinx/kria-docker

Now create a new docker file with the following content inside kria-docker/dockerfiles

```
# Copyright (C) 2022, Advanced Micro Devices, Inc. All rights reserved.
# SPDX-License-Identifier: MIT

FROM xilinx/kria-developer:2022.1 as build

RUN \
    pip install meson ; \
    apt install --yes build-essential ninja-build;\
    git clone https://github.com/sanamshakya/v4l-utils.git;\
    cd v4l-utils;\
    meson setup build/ ; \
    ninja -C build/ ; \
    ninja -C build/ install ;



FROM xilinx/kria-developer:2022.1 as build
WORKDIR /workspace


RUN \
    apt install --yes libusb-1.0-0-dev ; \
    apt install  --yes libopencv-dev libboost-all-dev libusb-1.0-0-dev
libprotobuf-dev protobuf-compiler ; \
    apt update --fix-missing; \
    apt install  --yes  libhdf5-dev hdf5-tools libglew-dev libglfw3-dev
libcanberra-gtk-module ffmpeg  ; \
    apt update --fix-missing; \
    apt install --yes libhdf5-dev; \
    apt install --yes git python3 python3-pip python3-setuptools
python3-wheel ninja-build ;\
    pip3 install meson ;\
    git clone https://github.com/sanamshakya/openeb.git; \
    cd openeb ;\
    mkdir -p build;  \
    cd build; \
    cmake .. -DBUILD_TESTING=OFF -DCOMPILE_PYTHON3_BINDINGS=OFF; \
    cmake --build . --config Release -- -j4 ;

FROM xilinx/kria-runtime:2022.1
RUN \
    apt install --yes graphviz ; \
    pip install pydot ;\
    pip install meson ; \
    apt install --yes build-essential ninja-build;

COPY --from=build /workspace/openeb/build/bin /usr/local/bin
COPY --from=build /workspace/openeb/build/lib /usr/local/lib

#ENV PATH="${PATH}:/opt/xilinx/kv260-smartcam/bin"
ENV LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:/usr/local/lib"
```

Build docker image using psee-openeb-v4l2 dockerfile

```
docker build -f <docker-file-name> . -t <docker-image-name>
```

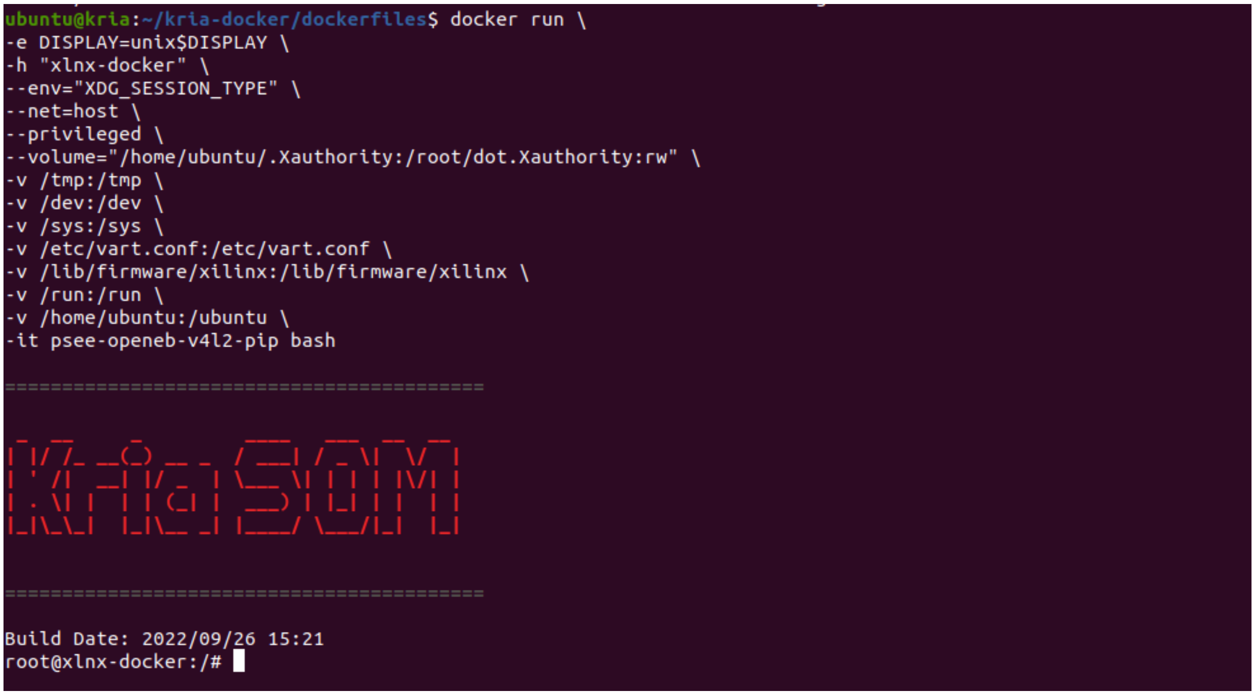
This will create docker image with `docker-image-name`, which can be listed using `docker image` command

Even above file builds updated v4l-utils package, it is unable to update image libraries. So it has to be updated manually running following command inside docker environment:

First run the docker container to start the docker container shell:
Docker run command:

```
docker run \
-e DISPLAY=unix$DISPLAY \
-h "xlnx-docker" \
--env="XDG_SESSION_TYPE" \
--net=host \
--privileged \
--volume="/home/ubuntu/.Xauthority:/root/dot.Xauthority:rw" \
-v /tmp:/tmp \
-v /dev:/dev \
-v /sys:/sys \
-v /etc/vart.conf:/etc/vart.conf \
-v /lib/firmware/xilinx:/lib/firmware/xilinx \
-v /run:/run \
-v /home/ubuntu:/ubuntu \
-it <psee-openeb-ml-vitis:latest> bash
```

Here is console after running above command:



```
ubuntu@kria:~/kria-docker/dockerfiles$ docker run \
-e DISPLAY=unix$DISPLAY \
-h "xlnx-docker" \
--env="XDG_SESSION_TYPE" \
--net=host \
--privileged \
--volume="/home/ubuntu/.Xauthority:/root/dot.Xauthority:rw" \
-v /tmp:/tmp \
-v /dev:/dev \
-v /sys:/sys \
-v /etc/vart.conf:/etc/vart.conf \
-v /lib/firmware/xilinx:/lib/firmware/xilinx \
-v /run:/run \
-v /home/ubuntu:/ubuntu \
-it psee-openeb-v4l2-pip bash

==========================================

 _  __     _       ____   ___  __  __
| |/ /_ __(_) __ _/ ___| / _ \|  \/  |
| ' /| '__| |/ _` \___ \| | | | |\/| |
| . \| |  | | (_| |___) | |_| | |  | |
|_|\_\_|  |_|\__,_|____/ \___/|_|  |_|

==========================================

Build Date: 2022/09/26 15:21
root@xlnx-docker:/#
```

Run following commands inside docker container console:

```
apt-get update --fix-missing
apt install git
git clone https://github.com/sanamshakya/v4l-utils.git
cd v4l-utils
meson setup build/
ninja -C build/
ninja -C build/ install
```

Next install necessary package to run the ML application:

```
pip install numpy
pip install "opencv-python==4.5.5.64"
pip install "h5py==3.7.0"
```

Installing Vitis AI 3.5:

Get the Vitis AI 3.5 build packages:

```
wget -O vai3.5_kr260.zip
https://www.xilinx.com/bin/public/openDownload?filename=vai3.5_kr260.zip
unzip vai3.5_kr260.zip
cd vai3.5_kr260/target/runtime_deb/
bash setup.sh
cd ..
tar -xzf lack_lib.tar.gz
cp -r lack_lib/* /usr/lib
Cd ..
cp ./xbutil_tool/xbutil2 /usr/bin/unwrapped/
```

Also need to copy `/usr/lib/aarch64-linux-gnu/libdfx.so.1.0` from kria ubuntu filesystem to `/usr/lib` directory.
It can be copied using docker copy command or copying the file in `/home/ubuntu` directory, which is shared between docker container and ubuntu user space.

Saving Docker container with updated installation:

After above installation, updated image is created with above running docker container ID. It will make the image with updated installation. Using `docker ps` command get the container ID corresponding to above docker run. Then use docker commit command to create the image of the container:

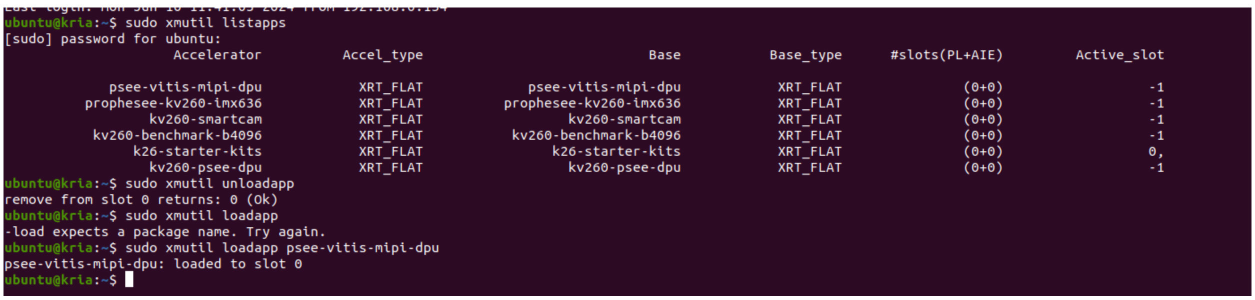
```
docker commit <container ID> <docker image name>
```

One can verify the creation of docker image by running `docker images` command.

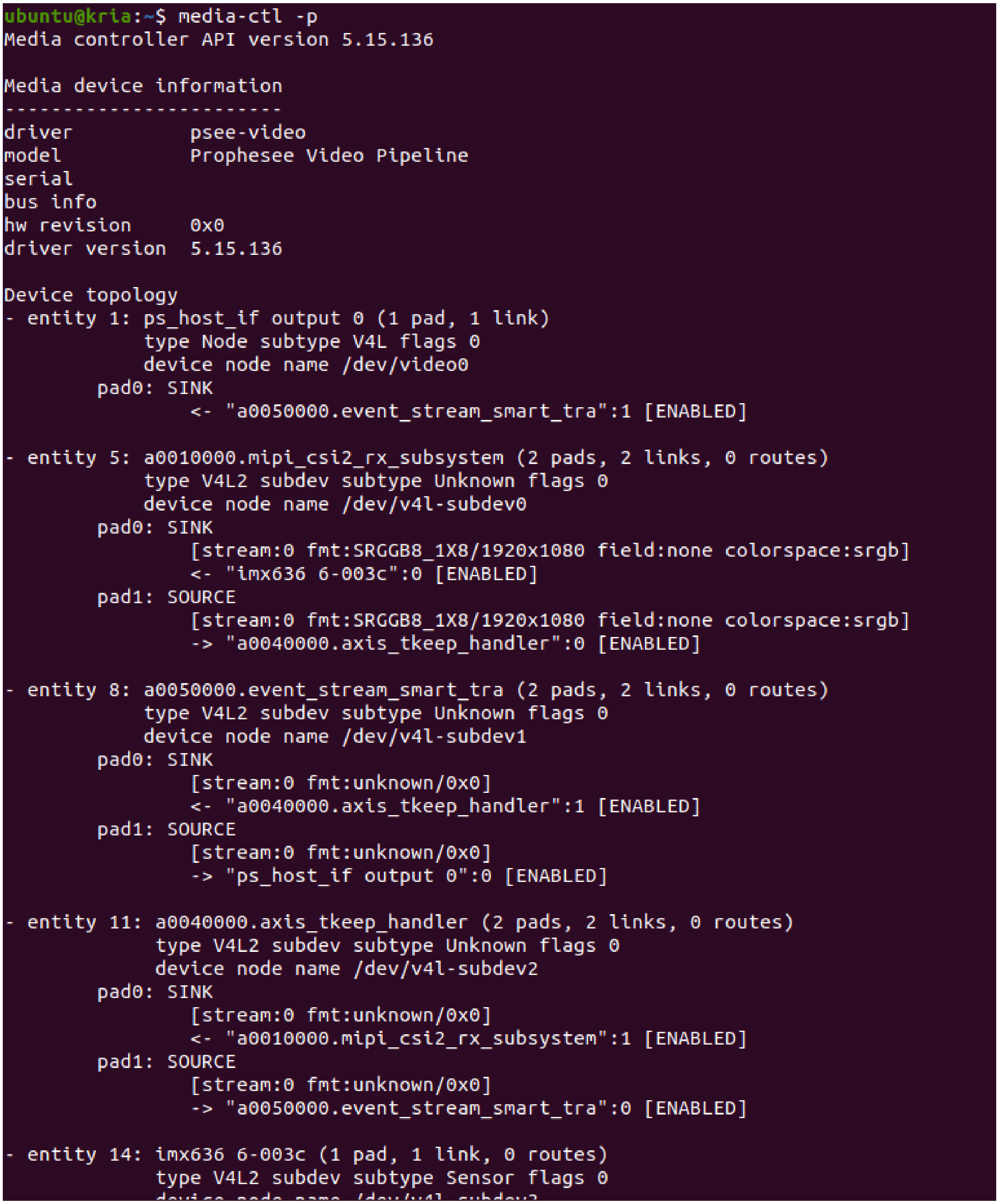
Running Prophesee Docker image:

Load the hardware overlay using xmutil :

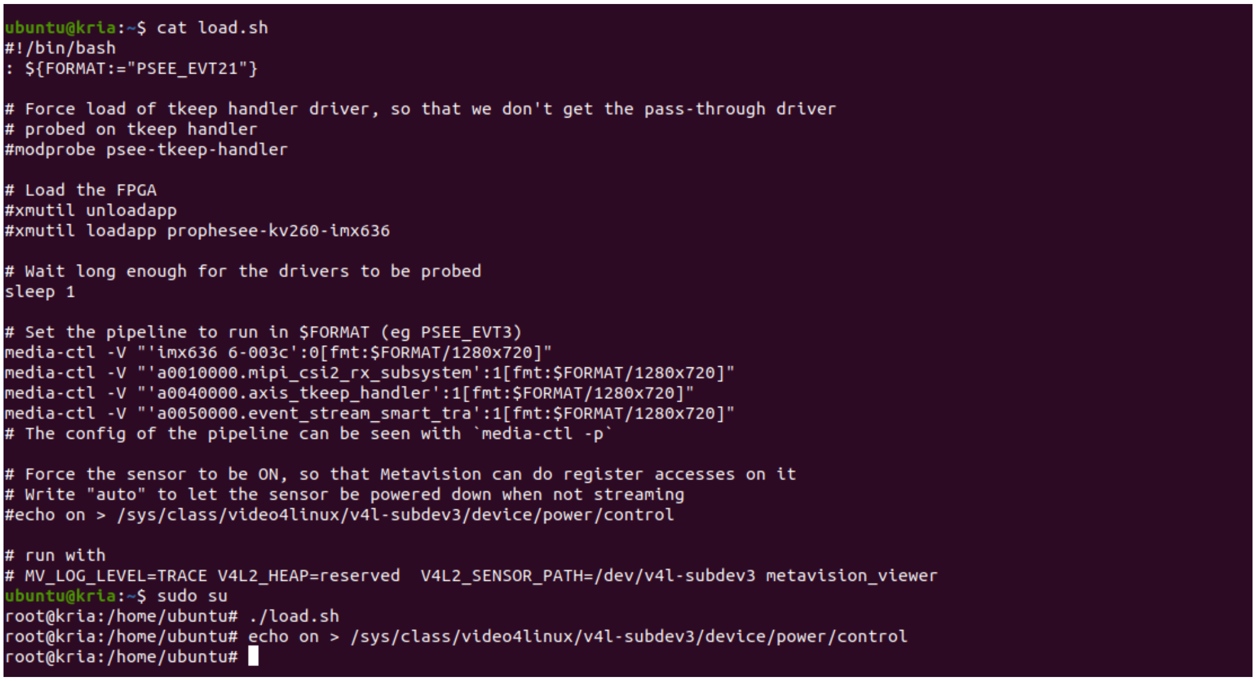
```
Sudo xmutil unloadapp
Sudo xmutil loadapp <hardware overlay>
```



```
Last login: Mon Jun 10 11:41:05 2024 from 192.168.0.154
ubuntu@kria:~$ sudo xmutil listapps
[sudo] password for ubuntu:
                     Accelerator           Accel_type                            Base           Base_type         #slots(PL+AIE)          Active_slot

             psee-vitis-mipi-dpu             XRT_FLAT             psee-vitis-mipi-dpu            XRT_FLAT                  (0+0)                   -1
          prophesee-kv260-imx636             XRT_FLAT          prophesee-kv260-imx636            XRT_FLAT                  (0+0)                   -1
                 kv260-smartcam             XRT_FLAT                  kv260-smartcam            XRT_FLAT                  (0+0)                   -1
            kv260-benchmark-b4096             XRT_FLAT            kv260-benchmark-b4096            XRT_FLAT                  (0+0)                   -1
               k26-starter-kits             XRT_FLAT                k26-starter-kits            XRT_FLAT                  (0+0)                   0,
                 kv260-psee-dpu             XRT_FLAT                  kv260-psee-dpu            XRT_FLAT                  (0+0)                   -1
ubuntu@kria:~$ sudo xmutil unloadapp
remove from slot 0 returns: 0 (Ok)
ubuntu@kria:~$ sudo xmutil loadapp
-load expects a package name. Try again.
ubuntu@kria:~$ sudo xmutil loadapp psee-vitis-mipi-dpu
psee-vitis-mipi-dpu: loaded to slot 0
ubuntu@kria:~$ 
```



```
ubuntu@kria:~$ media-ctl -p
Media controller API version 5.15.136

Media device information
------------------------
driver          psee-video
model           Prophesee Video Pipeline
serial
bus info
hw revision     0x0
driver version  5.15.136

Device topology
- entity 1: ps_host_if output 0 (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: SINK
                <- "a0050000.event_stream_smart_tra":1 [ENABLED]

- entity 5: a0010000.mipi_csi2_rx_subsystem (2 pads, 2 links, 0 routes)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev0
        pad0: SINK
                [stream:0 fmt:SRGGB8_1X8/1920x1080 field:none colorspace:srgb]
                <- "imx636 6-003c":0 [ENABLED]
        pad1: SOURCE
                [stream:0 fmt:SRGGB8_1X8/1920x1080 field:none colorspace:srgb]
                -> "a0040000.axis_tkeep_handler":0 [ENABLED]

- entity 8: a0050000.event_stream_smart_tra (2 pads, 2 links, 0 routes)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev1
        pad0: SINK
                [stream:0 fmt:unknown/0x0]
                <- "a0040000.axis_tkeep_handler":1 [ENABLED]
        pad1: SOURCE
                [stream:0 fmt:unknown/0x0]
                -> "ps_host_if output 0":0 [ENABLED]

- entity 11: a0040000.axis_tkeep_handler (2 pads, 2 links, 0 routes)
             type V4L2 subdev subtype Unknown flags 0
             device node name /dev/v4l-subdev2
        pad0: SINK
                [stream:0 fmt:unknown/0x0]
                <- "a0010000.mipi_csi2_rx_subsystem":1 [ENABLED]
        pad1: SOURCE
                [stream:0 fmt:unknown/0x0]
                -> "a0050000.event_stream_smart_tra":0 [ENABLED]

- entity 14: imx636 6-003c (1 pad, 1 link, 0 routes)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev3
```



```
ubuntu@kria:~$ cat load.sh
#!/bin/bash
: ${FORMAT:="PSEE_EVT21"}

# Force load of tkeep handler driver, so that we don't get the pass-through driver
# probed on tkeep handler
#modprobe psee-tkeep-handler

# Load the FPGA
#xmutil unloadapp
#xmutil loadapp prophesee-kv260-imx636

# Wait long enough for the drivers to be probed
sleep 1

# Set the pipeline to run in $FORMAT (eg PSEE_EVT3)
media-ctl -V "'imx636 6-003c':0[fmt:$FORMAT/1280x720]"
media-ctl -V "'a0010000.mipi_csi2_rx_subsystem':1[fmt:$FORMAT/1280x720]"
media-ctl -V "'a0040000.axis_tkeep_handler':1[fmt:$FORMAT/1280x720]"
media-ctl -V "'a0050000.event_stream_smart_tra':1[fmt:$FORMAT/1280x720]"
# The config of the pipeline can be seen with `media-ctl -p`

# Force the sensor to be ON, so that Metavision can do register accesses on it
# Write "auto" to let the sensor be powered down when not streaming
#echo on > /sys/class/video4linux/v4l-subdev3/device/power/control

# run with
# MV_LOG_LEVEL=TRACE V4L2_HEAP=reserved  V4L2_SENSOR_PATH=/dev/v4l-subdev3 metavision_viewer
ubuntu@kria:~$ sudo su
root@kria:/home/ubuntu# ./load.sh
root@kria:/home/ubuntu# echo on > /sys/class/video4linux/v4l-subdev3/device/power/control
root@kria:/home/ubuntu# █
```

For GUI to get the output from application running in docker container, run following command:

```
xhost local:docker
```

Then run the docker run command: