Docker Information

Version 1.1, February 17, 2020

Table of Contents

Chapter 1. Introduction

This docker is designed to enable portability to leverage imx series GPUs. It enables flexibility on top of the yocto image and allows use of libraries available in ubuntu for machine learning which is otherwise difficult. Using docker, user can develop and prototype GPU applications and then ship and run it anywhere using the container.

Chapter 2. Yocto Installation Guide

This chapter provides step by step guide for configuring and building the Linux Yocto Release for *i.MX 8 family* along with the *meta-virtualization* layer. The main configuration changes are:

- Retrieve and add the *meta-virtualization* layer.
- Modify the *local.conf* configuration file or layer files depending on which configuration is needed.

2.1. Software Requirements

1. Host GNU/Linux Ubuntu Distribution `16.04`.
2. Host packages:
   a. The essential Yocto project host packages are:

      Host GNU/Linux PC Terminal

      ```
      $ apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev
      ```

   b. The i.MX layers host packages for the Ubuntu OS host setup are:

      Host GNU/Linux PC Terminal

      ```
      $ apt-get install libsdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils python-pysqlite2 help2man gcc g++ make
      desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc u-boot-tools
      ```

      You need root privileges (sudo) to use `apt-get` package manager.

2.2. Setting Up the Yocto Environment

1. Follow the next instructions to start with *Yocto BSP Releases*:

   Host GNU/Linux PC Terminal

   ```
   $ mkdir ~/bin
   $ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo
   $ PATH=${PATH}:~/bin
   ```

   The above content is only required once.

2. To retrieve the *imx-linux-warrior* branch from `4.19.35` release:

   Host GNU/Linux PC Terminal

   ```
   $ repo init -u git://source.codeaurora.org/external/imx/imx-manifest.git -b imx-linux-warrior -m imx-4.19.35-1.1.0.xml
   $ repo sync
   ```

   This step may take several minutes.

3. Retrieve the *meta-virtualization* layer:

   Host GNU/Linux PC Terminal

```
$ git clone https://git.yoctoproject.org/git/meta-virtualization sources/meta-virtualization -b warrior
```

2.3. Image Building Process

2.3.1. Creating the Build Project

1. To first step you need to do is create the build folder project:

   Host GNU/Linux PC Terminal

   ```
   $ EULA=1 MACHINE=imx8qmmek DISTRO=fsl-imx-xwayland source ./fsl-setup-release.sh -b bld-dir
   ```

2.3.2. Configuring the Yocto Layer

1. Update the *bblayers.conf* file adding the *meta-virtualization* layer:

   Host GNU/Linux PC Terminal

   ```
   $ echo "BBLAYERS += \" \${BSPDIR}/sources/meta-virtualization \"" >> conf/bblayers.conf
   ```

2.3.3. Configuring the Yocto

1. Update the `local.conf` as follows:

   a. Add extra space for images:

      local.conf file

      ```
      IMAGE_ROOTFS_EXTRA_SPACE = " 10000000 "
      ```

      This step requires a 32GB SDCard. You can remove it and resize the SDCard manually after the deploy image process.

   b. Add *docker* to the image and also *connman* to manage the networking:

      local.conf file

      ```
      DISTRO_FEATURES_append = " virtualization "
      IMAGE_INSTALL_append += " docker docker-contrib connman connman-client "
      CORE_IMAGE_EXTRA_INSTALL += " openssh  "
      ```

   c. Add basic development capabilities:

      local.conf file

      ```
      EXTRA_IMAGE_FEATURES = " dev-pkgs debug-tweaks tools-debug ssh-server-openssh "
      ```

   d. Add packages for networking capabilities:

      local.conf file

```
IMAGE_INSTALL_append = " net-tools iputils dhcpcd "
```

e. Add generic tools:

   local.conf file

```
IMAGE_INSTALL_append = " which gzip python python-pip "
IMAGE_INSTALL_append = " wget cmake gtest git zlib patchelf "
IMAGE_INSTALL_append = " nano grep vim tmux swig tar unzip "
IMAGE_INSTALL_append = " parted e2fsprogs e2fsprogs-resize2fs "
```

f. Add *CMake* for SDK's cross-compiler:

   local.conf file

```
TOOLCHAIN_HOST_TASK_append = " nativesdk-cmake nativesdk-make "
```

2.3.4. Building an Full i.MX Image

1. Build the image as follows:

   Host GNU/Linux PC Terminal

```
$ bitbake fsl-image-qt5
```

This step may take several hours.

Chapter 3. Getting Started with Docker

3.1. Docker Information

1. To retrieve a very wide system information about Docker, use the next command:

   i.MX Board Terminal

   ```
   $ docker info
   ```

3.2. Initial Docker Steps

1. Start pulling the basic GNU/Linux Ubuntu image and then install the needed tools onto it:

   i.MX Board Terminal

   ```
   $ docker pull ubuntu:16.04
   ```

2. Launch a basic Docker image:

   i.MX Board Terminal

   ```
   $ docker run -it --privileged=true --net=host ubuntu:16.04
   ```

   The *privileged mode* is required to host exposes the `/dev/galcore` to the docker container.

3.3. Extra Docker Packages

1. Install the following packages:

   Docker Container Terminal

   ```
   $ apt-get update
   $ apt-get install vim git build-essential checkinstall cifs-utils nfs-common software-properties-common strace wget unzip
   ```

2. If not present, create the following folder:

   Docker Container Terminal

   ```
   $ mkdir -p /etc/OpenCL/vendors
   ```

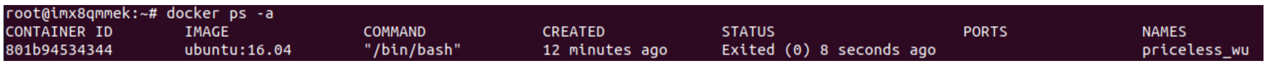
3. Close the Docker container and get its id:

   Docker Container Terminal

   ```
   $ exit
   ```

   i.MX Board Terminal

```
$ docker ps -a
```



```
root@imx8qmmek:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
801b94534344        ubuntu:16.04        "/bin/bash"         12 minutes ago      Exited (0) 8 seconds ago                        priceless_wu
```

Figure 1. Container ID example

4. Save the docker id and commit it:

   i.MX Board Terminal

   ```
   $ docker commit <my_container_id>
   ```

5. Copy the *clinfo* from Host to Docker container:

   i.MX Board Terminal

   ```
   $ docker cp /opt/viv_samples/cl11/UnitTest/clinfo <my_container_id>:/home/
   ```

6. Copy the *Vivante.icd* to Docker container:

   i.MX Board Terminal

   ```
   $ docker cp /etc/OpenCL/vendors/Vivante.icd <my_container_id>:/etc/OpenCL/vendors/
   ```

7. By using the `docker cp` command or `scp` inside the docker container, copy all the following libraries from host */usr/lib* to docker container */lib/aarch64-linux-gnu/*:

   ```
   libCLC.so libNNGPUBinary-xsvx.so libOpenVXU.so libdrm.so.2 libneuralnetworks.so
   libwayland-client.so.0 libEGL.so.1 libNNVXCBinary-evis.so libOvx12VXCBinary-evis.so
   libgbm.so libnnrt.so libGAL.so libOpenCL.so libVSC.so libgbm_viv.so libovxlib.so libwayland-server.so.0
   libGLESv2.so.2 libOpenVX.so libVivanteOpenCL.so
   ```

8. And the following libraries from host */lib* to docker container */lib/aarch64-linux-gnu/* as well:

   ```
   libm.so.6 libm-2.29.so
   ```

9. To start the Container again, run the following commands:

   i.MX Board Terminal

   ```
   $ docker start <my_container_id>
   $ docker attach <my_container_id>
   ```

10. Upgrade the *GCC* package:

    Docker Container Terminal

    ```
    $ add-apt-repository ppa:ubuntu-toolchain-r/test -y
    $ apt-get update
    $ apt-get install gcc-9 -y
    $ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 60
    $ apt-get install libstdc++6
    ```

Chapter 4. Testing the Environment

1. Execute the *clinfo* program:

   Docker Container Terminal

   ```
   $ ./clinfo
   ```

   Use strace to check library dependency in case of error.

2. Further NNAPI execution can be verified by `label_image` application. Once it is established that `clinfo` is executing then the following `tflite` sample application can be executed. From `/usr/bin/tensorflow-lite-1.13.2/examples`, copy the following files from host to docker container:

   i.MX Board Terminal

   ```
   $ docker cp /usr/bin/tensorflow-lite-1.13.2/examples/label_image <my_container_id>:/home/
   $ docker cp /usr/bin/tensorflow-lite-1.13.2/examples/grace_hopper.bmp <my_container_id>:/home/
   ```

3. Inside the docker container, go to `home` directory and download the `tflite` model:

   Docker Container Terminal

   ```
   $ cd /home/
   $ wget download.tensorflow.org/models/mobilenet_v1_224_android_quant_2017_11_08.zip
   ```

4. `Unzip` it:

   Docker Container Terminal

   ```
   $ unzip mobilenet_v1_224_android_quant_2017_11_08.zip
   ```

5. Run the `label_image`:

   Docker Container Terminal

   ```
   $ ./label_image -a 1 -m mobilenet_quant_v1_224.tflite
   ```