



Build and Run Embedded Apps Faster from QtCreator with Docker

July 8, 2020

Building Qt SW for Embedded Targets

- Based on Yocto Embedded Linux recipes
- Created embedded toolchains can be configured as Qt Creator kits
- Qt Debug Bridge helps makes debugging and deployment rather straightforward
- Boot 2 Qt - Plenty of ready-made images for variety of development boards
- How to improve the workflow with containers?

Ready-made B2Qt stack

Sample app

Qt

Operating system
and middleware

Hardware drivers
and BSP

DevKit HW



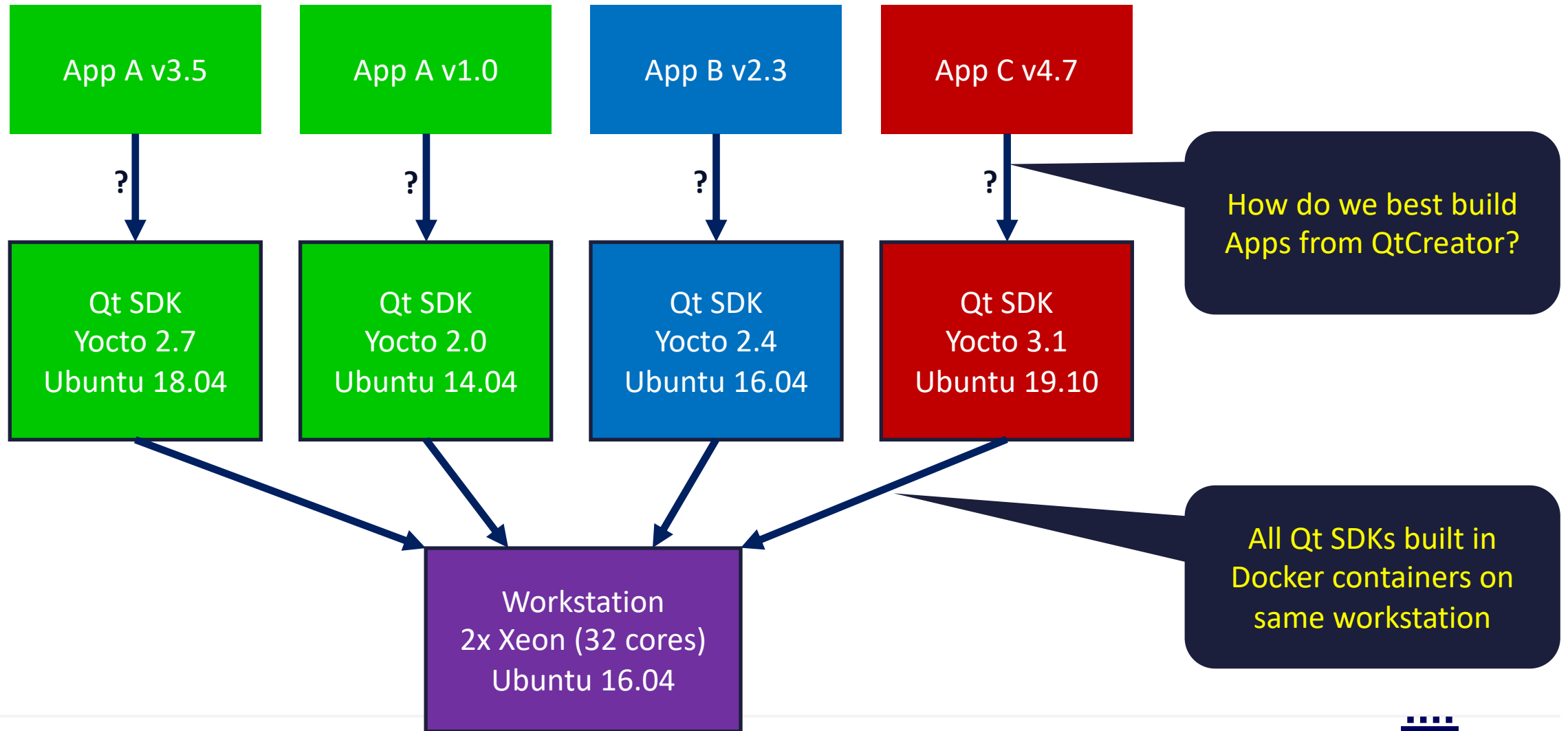


Build and Run Embedded Apps Faster from QtCreator with Docker

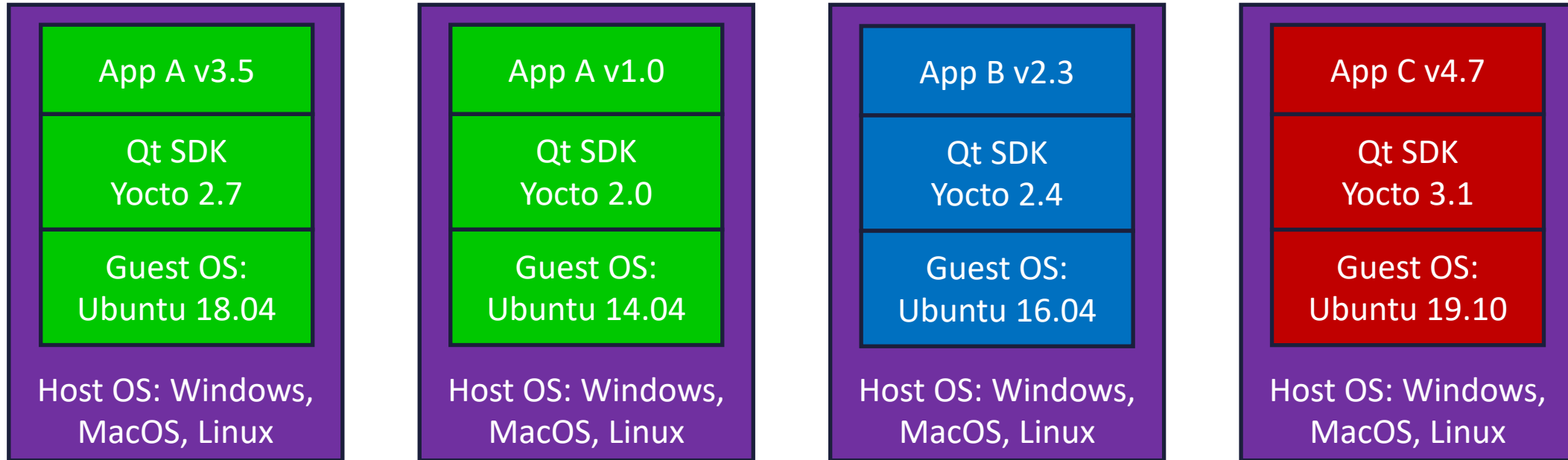
Burkhard Stubert

As a solo consultant, I help teams succeed with Qt embedded systems

The Problem



Solution 1 (Currently Used): Virtual Machines



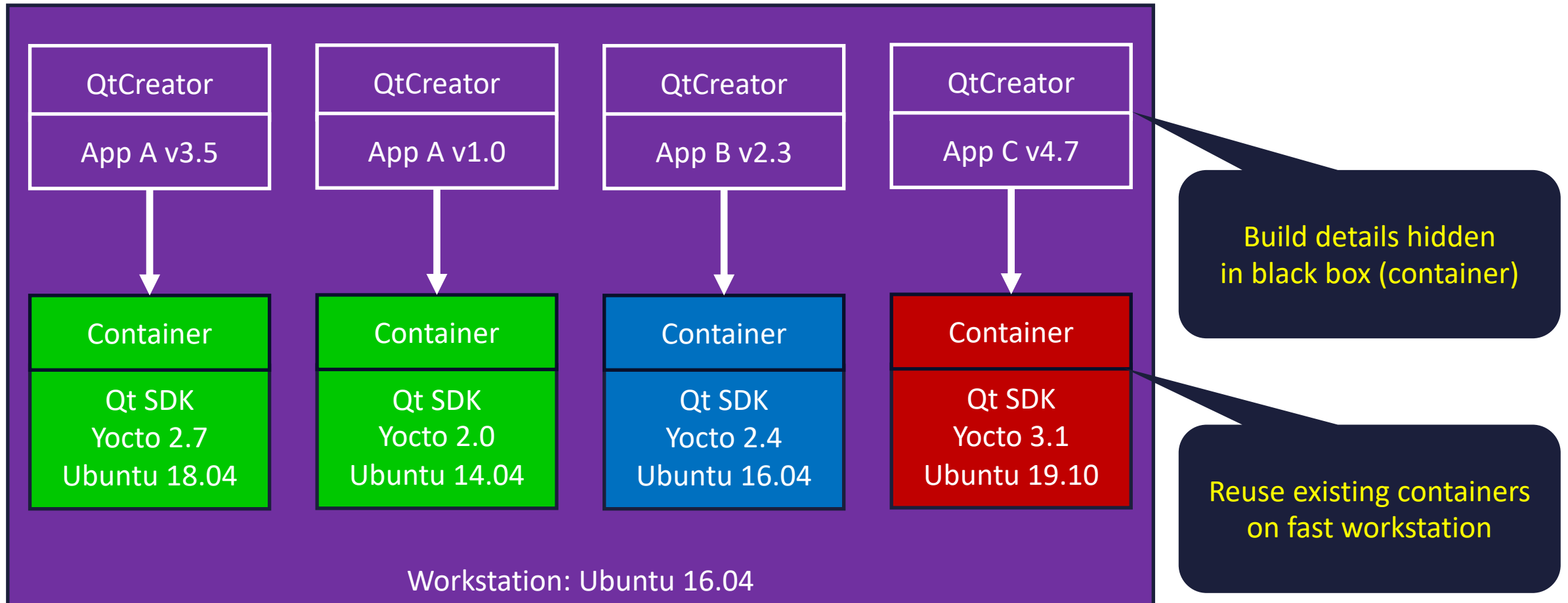
2-3 VMs fit on same dev PC

VMs slower than containers

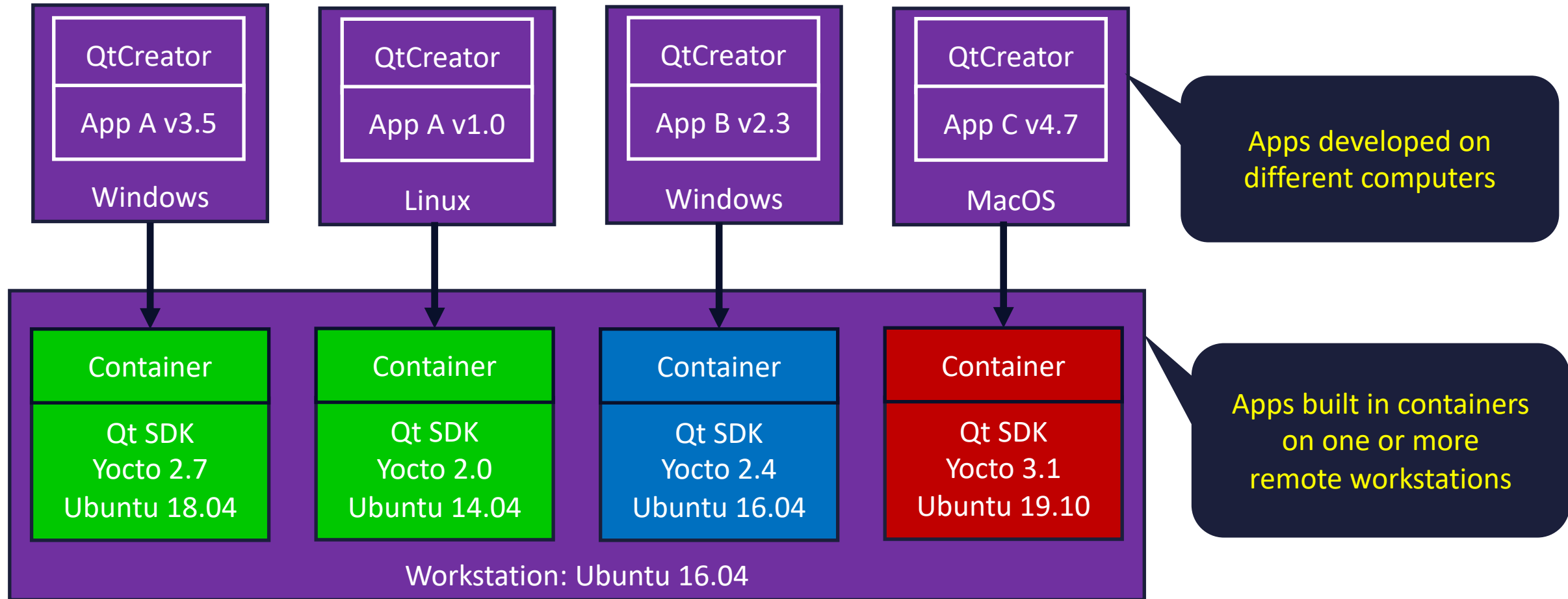
Fast workstation and
existing containers
not used for app builds



Solution 2 (This Talk): Containers on Local Workstation



Solution 3 (Future): Containers on Remote Workstation



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

How QtCreator Calls CMake (Native Build)

Stage	Working Directory	CMake Command
Configure project	<code>/tmp/QtCreator-Yqhjyl/qtc-cmake-MDHAJjOH => <work-dir-1></code>	<code>cmake '-GUnix Makefiles' -C <work-dir-1>/qtcsettings.cmake /public/Work/cuteradio-apps</code>
Generate build files	<code>/public/Work/build-cuteradio-apps-Desktop_Qt_5_14_2_GCC_64bit-Debug => <work-dir-2></code>	<code>cmake '-GUnix Makefiles' -C <work-dir-2>/qtcsettings.cmake /public/Work/cuteradio-app</code>
Compile	<code><work-dir-2></code>	<code>cmake --build . --target all -- -j4</code>
Install	<code><work-dir-2></code>	<code>cmake --build . --target install</code>



Call Docker wrapper
instead of cmake

Docker Wrapper for CMake

```
#!/bin/bash

args=$(echo $@ | sed -e "s|-GCodeBlocks - Unix Makefiles|'-GCodeBlocks - Unix Makefiles'|g")

docker run --rm -v /public/Work:/public/Work -v /tmp:/tmp \
  -w $(pwd) dr-yocto:sdk-18.04 cmake $args
```

Save script in
\$HOME/bin/dr-cmake

/public/Work and /tmp
visible both on host PC
and in container

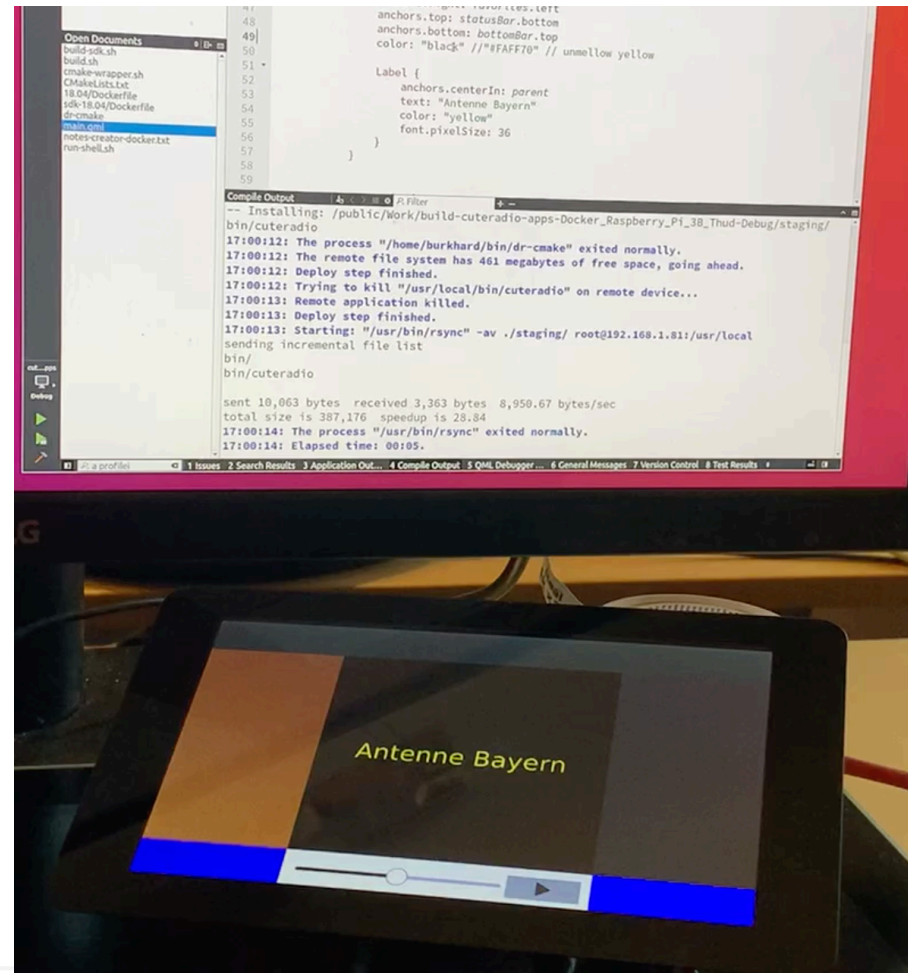
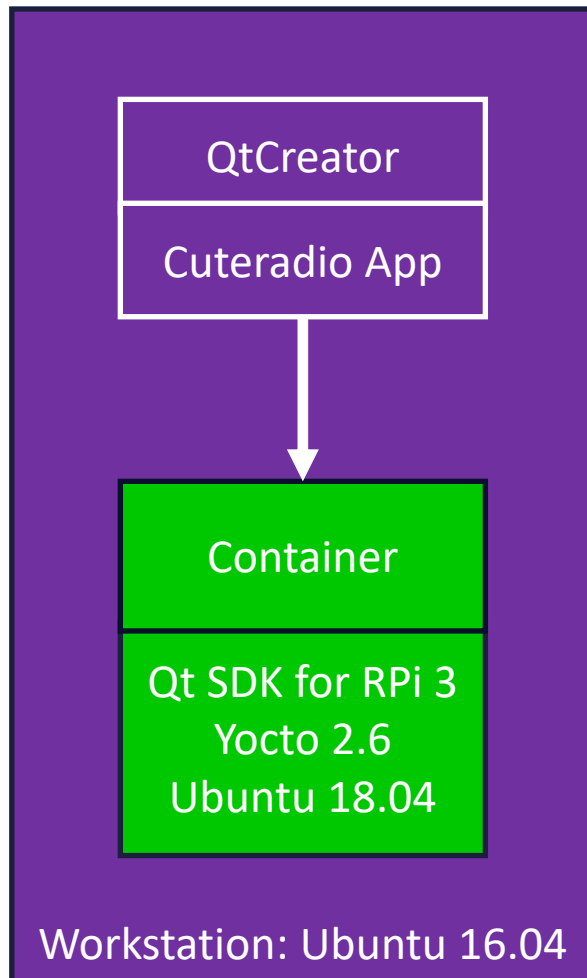
Resurrect single quotes
removed by shell



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

My Setup



Only a proof of concept!

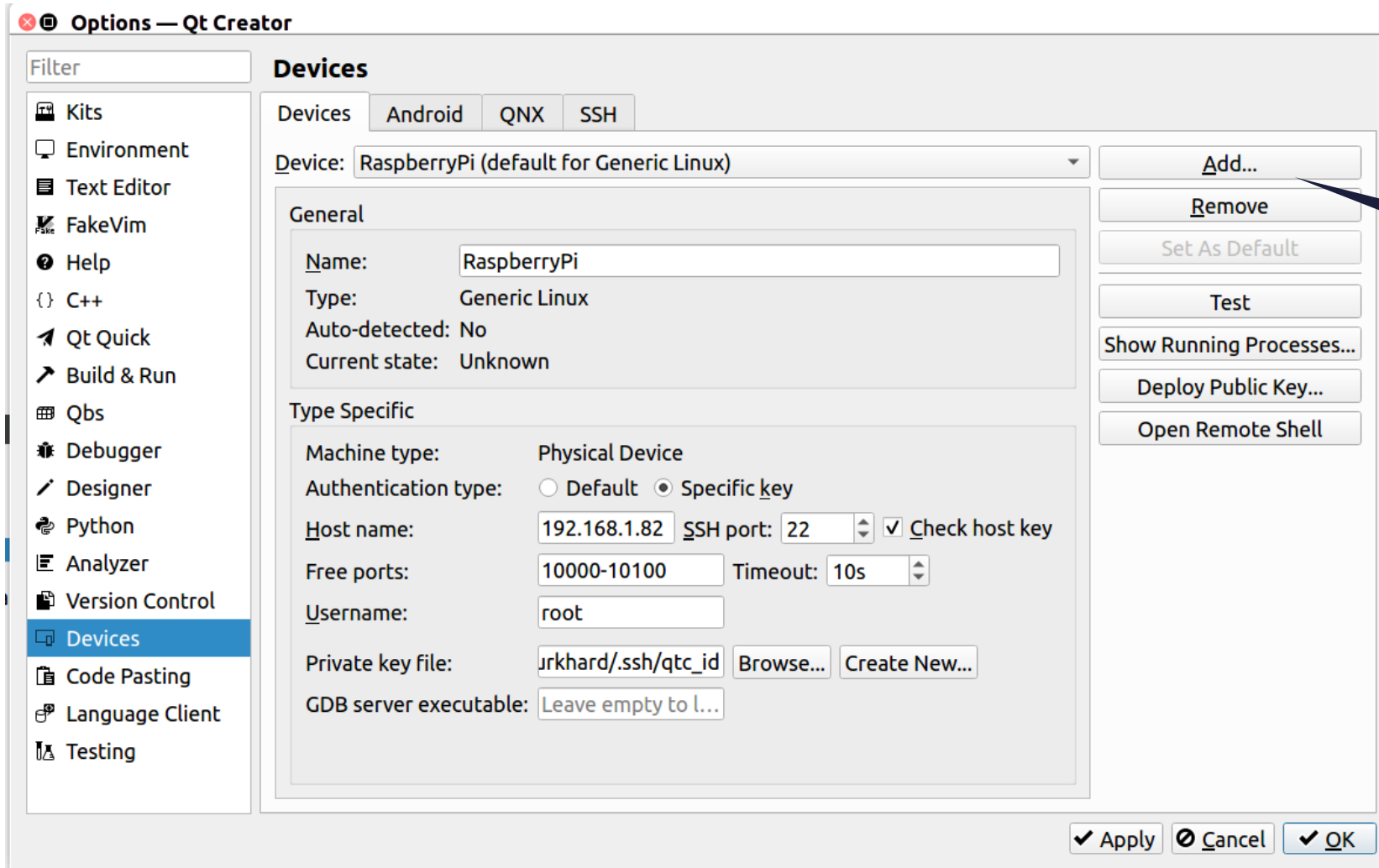
Prerequisites: General

- Install Docker on workstation (see [2])
- Create a Docker container (sdk-18.04) for Yocto builds (see [3])
- Build a Linux image with the Docker container (see [3])
 - Or: Use a pre-built Boot2Qt image from The Qt Company (see [5] and [6])
- Build Qt SDK with the Docker container (see [4])
 - Or: Use a pre-built Boot2Qt SDK from The Qt Company (see [5] and [6])
- Establish SSH connection between workstation and device (see [1] and [4])

QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

Prerequisites: SSH Access to Device



See [1] and [4]
how to add device



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

Installing Qt SDK in Container

On Workstation:

```
$ cd /public/Work
$ docker run -t -rm -v /public/Work:/public/Work -v /tmp:/tmp \
  -w $(pwd) dr-yocto:sdk-18.04
```

In Docker container:

```
# cd cuteradio-thud/build-rpi3/tmp/deploy/sdk/
# ./poky-glibc-x86_64-meta-toolchain-qt5-cortexa7t2hf-neon-vfpv4-
toolchain-2.6.4.sh
Poky (Yocto Project Reference Distro) SDK installer version 2.6.4
=====
Enter target directory for SDK (default: /opt/poky/2.6.4):
/public/Work/qt-sdk-thud
...
```



Setting Up Build Environment in Dockerfile

```
source environment-setup-cortexa7t2hf-neon-vfpv4-poky-linux-gnueabi
```

```
export OECORE_NATIVE_SYSROOT="/public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux"  
export OECORE_TARGET_SYSROOT="${SDKTARGETSYSROOT}"  
export OECORE_BASELIB="lib"  
export OECORE_TARGET_ARCH="arm"  
export OECORE_TARGET_OS="linux-gnueabi"  
...
```

Change all environment variables

Dockerfile

```
ENV OECORE_NATIVE_SYSROOT="/public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux"  
ENV OECORE_TARGET_SYSROOT="${SDKTARGETSYSROOT}"  
ENV OECORE_BASELIB="lib"  
ENV OECORE_TARGET_ARCH="arm"  
ENV OECORE_TARGET_OS="linux-gnueabi"  
...
```



Fixing Environment Variables for QtCreator

```
export CC="arm-poky-linux-gnueabi-gcc -march=armv7ve -mthumb -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=${SDKTARGETSYSROOT}"  
export CFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types "
```

```
export CXX="arm-poky-linux-gnueabi-g++ -march=armv7ve -mthumb -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=${SDKTARGETSYSROOT}"  
export CXXFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types "
```

Move options from CC to CFLAGS
and from CXX to CXXFLAGS

Do same with LD/LDFLAGS
and CPP/PPFLAGS

```
ENV CC="arm-poky-linux-gnueabi-gcc"  
ENV CFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types -march=armv7ve -mthumb -  
mfloat-abi=hard -mcpu=cortex-a7 --sysroot=${SDKTARGETSYSROOT}"
```

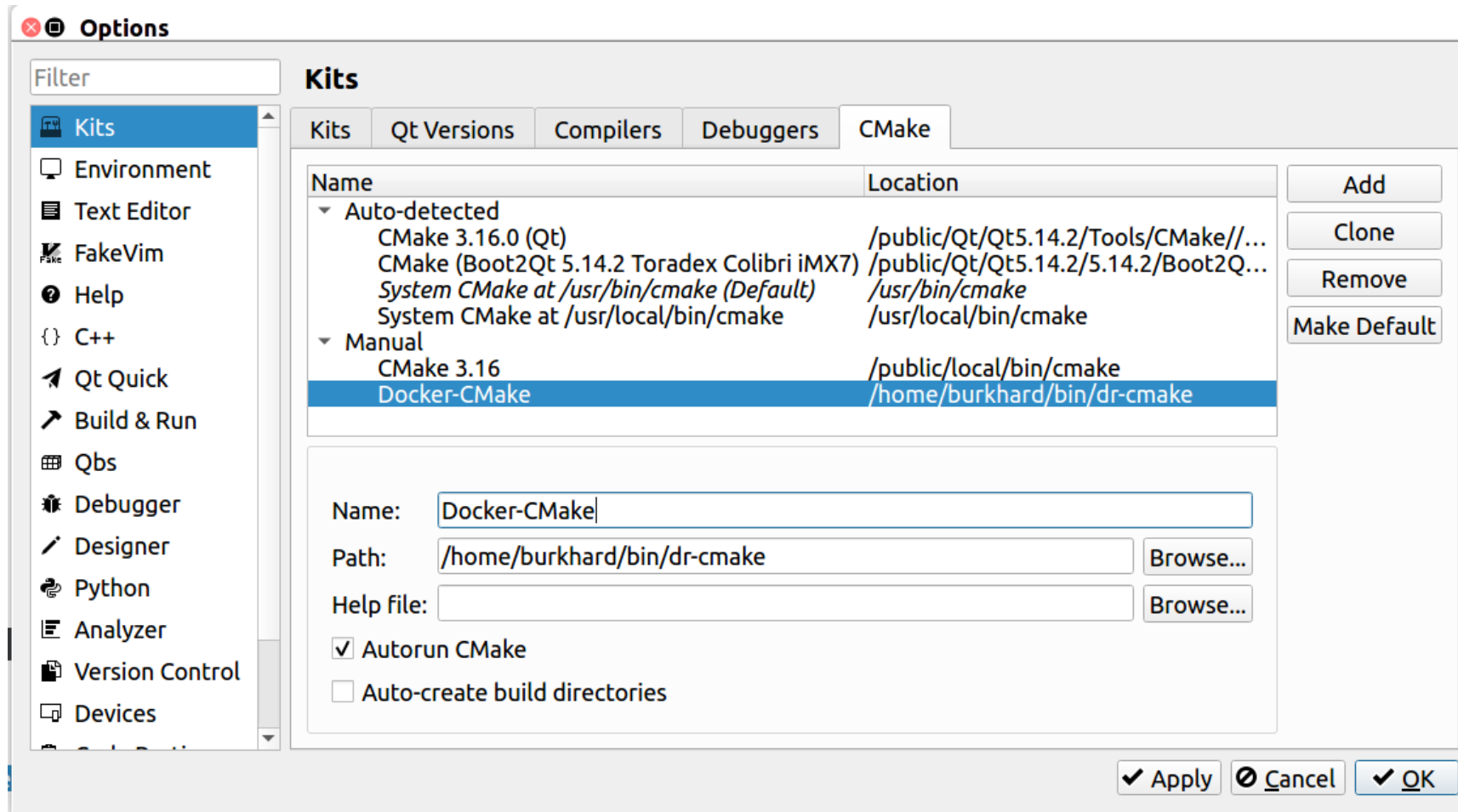
```
ENV CXX="arm-poky-linux-gnueabi-g++"  
ENV CXXFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types -march=armv7ve -mthumb -  
mfloat-abi=hard -mcpu=cortex-a7 --sysroot=${SDKTARGETSYSROOT}"
```



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

Configuring QtCreator: CMake



The screenshot shows the QtCreator Options dialog, specifically the Kits configuration page for CMake. The left sidebar contains a list of configuration categories, with 'Kits' selected. The main area is divided into tabs for 'Kits', 'Qt Versions', 'Compilers', 'Debuggers', and 'CMake'. The 'CMake' tab is active, displaying a table of installed CMake kits.

Name	Location
Auto-detected	
CMake 3.16.0 (Qt)	/public/Qt/Qt5.14.2/Tools/CMake//...
CMake (Boot2Qt 5.14.2 Toradex Colibri iMX7)	/public/Qt/Qt5.14.2/5.14.2/Boot2Q...
System CMake at /usr/bin/cmake (Default)	/usr/bin/cmake
System CMake at /usr/local/bin/cmake	/usr/local/bin/cmake
Manual	
CMake 3.16	/public/local/bin/cmake
Docker-CMake	/home/burkhard/bin/dr-cmake

Below the table, the configuration details for the selected 'Docker-CMake' kit are shown:

- Name:
- Path:
- Help file:
- Autorun CMake
- Auto-create build directories

At the bottom right, there are buttons for 'Apply', 'Cancel', and 'OK'.



Configuring QtCreator: Kit

Options

Filter

Kits

- Environment
- Text Editor
- FakeVim
- Help
- C++
- Qt Quick
- Build & Run
- Qbs
- Debugger
- Designer
- Python
- Analyzer
- Version Control
- Devices
- Code Pasting
- Language Client
- Testing

Kits | Qt Versions | Compilers | Debuggers | CMake

Name

- Auto-detected
 - Boot2Qt 5.14.2 Toradex Colibri iMX7 (default)
 - Desktop Qt 5.14.2 GCC 64bit
 - Qt 5.14.2 WebAssembly
- Manual
 - Docker Raspberry Pi 3B Thud**
 - TwoMonitors (ARM, Qt 5.9)

Add

Clone

Remove

Make Default

Settings Filter...

Default Settings Filter...

Name: Docker Raspberry Pi 3B Thud

File system name:

Device type: Generic Linux Device

Device: RaspberryPi (default for Generic Linux) Manage...

Sysroot: Browse...

Compiler: C: <No compiler> Manage...
C++: <No compiler> Manage...

Environment: No changes to apply. Change...

Debugger: None Manage...

Qt version: None Manage...

Qt mkspec:

Additional Qbs Profile Settings: Change...

CMake Tool: Docker-CMake Manage...

CMake generator: CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none> Change...

CMake Configuration: CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; ... Change...

Apply Cancel OK



Configuring QtCreator: Kit – CMake Generator

The image shows a screenshot of the Qt Creator interface with a dialog box open for configuring the CMake Generator. The dialog box, titled "CMake Generator — Qt Creator", contains the following fields and options:

- Executable: /home/burkhard/bin/dr-cmake
- Generator: Unix Makefiles
- Extra generator: CodeBlocks
- Platform: (empty text field)
- Toolset: (empty text field)

At the bottom of the dialog are "Cancel" and "OK" buttons. The background shows the "Kit" configuration page with the following settings:

- Compiler: (no compiler)
- Environment: (empty)
- Debugger: (empty)
- Qt version: (empty)
- Qt mkspec: (empty)
- Additional Qbs Profile Settings: (empty)
- CMake Tool: (empty)
- CMake generator: CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none>
- CMake Configuration: CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; ...

Buttons for "Manage...", "Change...", and "Apply" are visible throughout the interface.

Configuring QtCreator: Kit – CMake Configuration

Edit CMake Configuration — Qt Creator

```
CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}
CMAKE_C_COMPILER:STRING=%{Compiler:Executable:C}
CMAKE_MAKE_PROGRAM:INTERNAL=/usr/bin/make
CMAKE_PREFIX_PATH:INTERNAL=/public/Work/qt-sdk-thud/sysroots/cortexa7t2hf-neon-vfpv4-poky-linux-gnueabi/usr
CMAKE_TOOLCHAIN_FILE:INTERNAL=/public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/share/cmake/OEToolchainConfig.cmake
OE_QMAKE_PATH_EXTERNAL_HOST_BINS:INTERNAL=/public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/bin
QT_QMAKE_EXECUTABLE:STRING=%{Qt:qmakeExecutable}
```

Reset ✓ Apply ✗ Cancel ✓ OK

CMake Tool: Docker-CMake Manage...

CMake generator: CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none> Change...

CMake Configuration: CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; ... Change...

✓ Apply ✗ Cancel ✓ OK



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

Building the App with dr-cmake: Switching to Project "Docker Raspberry Pi"

The screenshot shows the Qt Creator IDE interface. The title bar reads "sdk-18.04/Dockerfile [ubuntu-18.04] - Qt Creator". The menu bar includes File, Edit, Build, Debug, Analyze, Tools, Window, and Help. The left sidebar contains a "Projects" section with a list of projects: "Boot2Qt 5.14.2 Torad...", "Desktop Qt 5.14.2 GC...", "Docker Raspberry Pi ..." (selected), "Qt 5.14.2 WebAssembly", and "TwoMonitors (ARM, Q...". The "Build & Run" section for the selected project shows "Build" and "Run" options. The "Project Settings" section lists Editor, Code Style, Dependencies, Environment, Clang Code Model, Clang Tools, and Testing. The main area displays the "CMAKE" configuration table with the following key-value pairs:

Key	Value
CMAKE_ASM_FLAGS	-O2 -pipe -g -feliminate-unused-deb...
CMAKE_BUILD_TYPE	Debug
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_LDFLAGS_FLAGS	-O2 -pipe -g -feliminate-unused-deb...
CMAKE_TOOLCHAIN_FILE	/public/Work/qt-sdk-thud/sysroots/...
QT_QMAKE_EXECUTABLE	
Qt5Core_DIR	/public/Work/qt-sdk-thud/sysroots/...
Qt5Gui_DIR	/public/Work/qt-sdk-thud/sysroots/...
Qt5Multimedia_DIR	/public/Work/qt-sdk-thud/sysroots/...

Below the table are buttons for "Add", "Edit", "Unset", and "Reset", along with an "Advanced" checkbox. An "Apply Configuration Changes" button is at the bottom of the table. The "Build Steps" section shows a build step: "Build: dr-cmake --build . --target all". The "Clean Steps" section shows a clean step: "Build: dr-cmake --build . --target clean". The "Build Environment" section shows "Use System Environment".



Building the App with dr-cmake: Output when Switching the Project

```
General Messages  Filter  + -
Running /home/burkhard/bin/dr-cmake '-GCodeBlocks - Unix Makefiles' -C /tmp/QtCreator-YqhjyI/qtc-cmake-
HiDLqZAJ/qtcsettings.cmake /public/Work/cuteradio-apps in /tmp/QtCreator-YqhjyI/qtc-cmake-HiDLqZAJ.
### dr-cmake: '-GCodeBlocks - Unix Makefiles' -C /tmp/QtCreator-YqhjyI/qtc-cmake-HiDLqZAJ/qtcsettings.cmake /
public/Work/cuteradio-apps
loading initial cache file /tmp/QtCreator-YqhjyI/qtc-cmake-HiDLqZAJ/qtcsettings.cmake
-- The C compiler identification is GNU 8.2.0
-- The CXX compiler identification is GNU 8.2.0
-- Check for working C compiler: /public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-
linux-gnueabi/arm-poky-linux-gnueabi-gcc
-- Check for working C compiler: /public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-
linux-gnueabi/arm-poky-linux-gnueabi-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-
linux-gnueabi/arm-poky-linux-gnueabi-g++
-- Check for working CXX compiler: /public/Work/qt-sdk-thud/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-
linux-gnueabi/arm-poky-linux-gnueabi-g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /tmp/QtCreator-YqhjyI/qtc-cmake-HiDLqZAJ
Elapsed time: 46:05.

Find: rsync  Find Previous  Find Next
ch Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Version Control 8 Test Results
```



Building the App with dr-cmake: CMake-Output of Switching to Project

```
15:50:29: Running steps for project cuteradio-apps...
15:50:30: Starting: "/home/burkhard/bin/dr-cmake" --build . --target all
[ 16%] Automatic MOC for target cuteradio
[ 16%] Built target cuteradio_autogen
[ 33%] Automatic RCC for qml.qrc
Scanning dependencies of target cuteradio
[ 50%] Building CXX object
CMakeFiles/cuteradio.dir/cuteradio_autogen/mocs_compilation.cpp.o
[ 66%] Building CXX object CMakeFiles/cuteradio.dir/main.cpp.o
[ 83%] Building CXX object
CMakeFiles/cuteradio.dir/cuteradio_autogen/EWIEGA46WW/qrc_qml.cpp.o
[100%] Linking CXX executable cuteradio
[100%] Built target cuteradio
15:50:33: The process "/home/burkhard/bin/dr-cmake" exited normally.
15:50:33: Elapsed time: 00:05.
```



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

Deployment Settings

The screenshot shows the Qt Creator interface with the 'Run Settings' dialog open. The left sidebar contains navigation options: Welcome, Edit, Design, Debug, Projects, and Help. The 'Projects' section lists several projects, with 'Docker Raspberry Pi ...' selected and its 'Run' option highlighted. The 'Run Settings' dialog is titled 'Run Settings' and has a green play button icon. It is divided into several sections:

- Deployment:** Method is set to 'Deploy to Remote Linux Host'. There are 'Add', 'Remove', and 'Rename...' buttons. Below, 'Files to deploy:' has an unchecked checkbox for 'Override deployment data from build system'. A table with columns 'Local File Path' and 'Remote Directory' is empty, with 'Add' and 'Remove' buttons to its right.
- Build:** The build command is 'dr-cmake --build . --target install'. A 'Details ^' button is to the right.
- Tool arguments:** An empty text input field.
- Targets:** A list of radio buttons: 'Current executable', 'all', 'clean', 'install' (selected), and 'test'.
- Check for free disk space:** A 'Details ^' button is to the right. Below, 'Remote path to check for free space:' is set to '/'. 'Required disk space:' is set to '5MB'.
- Kill current application instance:** A section with no visible options.
- Custom Process Step:** The command is 'rsync -av ./staging/ root@192.168.1.81:/usr/local'. A 'Details ^' button is to the right. Below, 'Command:' is 'rsync', 'Arguments:' is '-av ./staging/ root@192.168.1.81:/usr/local', and 'Working directory:' is '%{buildDir}'. Each has a 'Browse...' button.
- At the bottom, there is an 'Add Deploy Step' button.



Run Settings

Run

Run configuration: Custom Executable (on Raspb) Add... Remove Rename... Clone...

Remote executable:	<input type="text" value="/usr/local/bin/cuteradio"/>
Local executable:	<input type="text" value="%{CurrentProject:BuildPath}/staging/bin/cuteradio"/> Browse...
Command line arguments:	<input type="text" value="-platform eglfs"/> ↕ ▼
Working directory:	<input type="text" value="/home/root"/> Browse... ↶
<input type="checkbox"/> Run in terminal	
<input type="text" value=":0"/>	<input type="checkbox"/> Forward to local display

Environment

Use System Environment	Details ▼
-------------------------------	------------------------



QtCreator-CMake-Docker

- Motivation
- Idea: Docker Wrapper for CMake
- Prerequisites
 - General
 - SSH Access to Device
- Building App with Docker
 - Installing Qt SDK in Container
 - Configuring QtCreator
 - Building App with Docker-CMake
- Running App on Device
 - Deployment and Run Settings
 - Running App

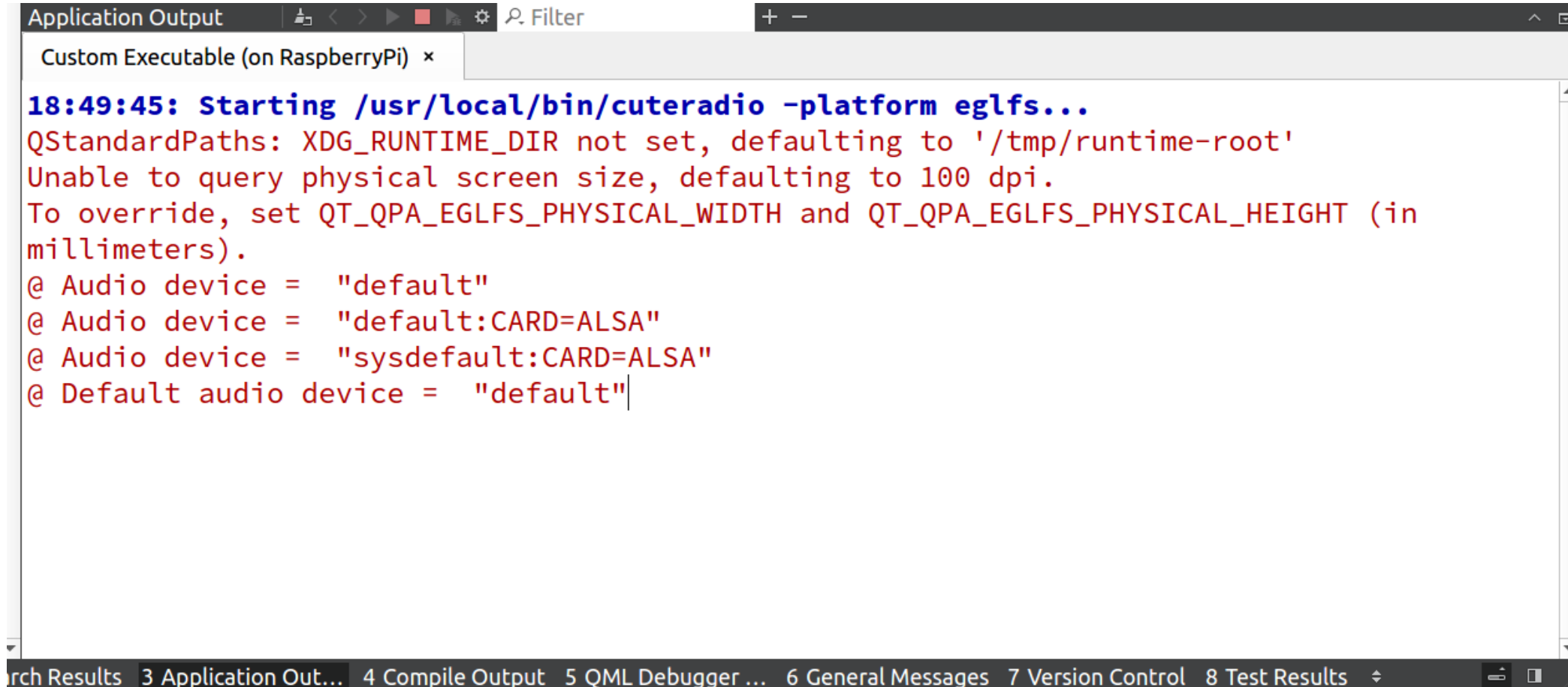
Running App on Device: Compile Output

```
Compile Output  Filter  + -
cmake_install.cmake
-- Install configuration: "Debug"
-- Up-to-date: /public/Work/build-cuteradio-apps-Docker_Raspberry_Pi_3B_Thud-Debug/
staging/bin/cuteradio
18:49:43: The process "/home/burkhard/bin/dr-cmake" exited normally.
18:49:43: The remote file system has 461 megabytes of free space, going ahead.
18:49:43: Deploy step finished.
18:49:43: Trying to kill "/usr/local/bin/cuteradio" on remote device...
18:49:44: Remote application killed.
18:49:44: Deploy step finished.
18:49:44: Starting: "/usr/bin/rsync" -av ./staging/ root@192.168.1.81:/usr/local
sending incremental file list

sent 114 bytes  received 13 bytes  84.67 bytes/sec
total size is 380,304  speedup is 2,994.52
18:49:45: The process "/usr/bin/rsync" exited normally.
18:49:45: Elapsed time: 00:05.
Arch Results 3 Application Out... 4 Compile Output 5 QML Debugger ... 6 General Messages 7 Version Control 8 Test Results
```



Running App on Device: Application Output



```
Application Output | Filter
Custom Executable (on RaspberryPi) x
18:49:45: Starting /usr/local/bin/cuteradio -platform eglfs...
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Unable to query physical screen size, defaulting to 100 dpi.
To override, set QT_QPA_EGLFS_PHYSICAL_WIDTH and QT_QPA_EGLFS_PHYSICAL_HEIGHT (in
millimeters).
@ Audio device = "default"
@ Audio device = "default:CARD=ALSA"
@ Audio device = "sysdefault:CARD=ALSA"
@ Default audio device = "default"
```

Search Results | 3 Application Out... | 4 Compile Output | 5 QML Debugger ... | 6 General Messages | 7 Version Control | 8 Test Results



Running App on Device: Instead of a Video

```
anchors.top: statusBar.left
anchors.bottom: bottomBar.top
color: "black" /*#FAFF70*/ // unmlow yellow

Label {
    anchors.centerIn: parent
    text: "Antenne Bayern"
    color: "yellow"
    font.pixelSize: 36
}
```

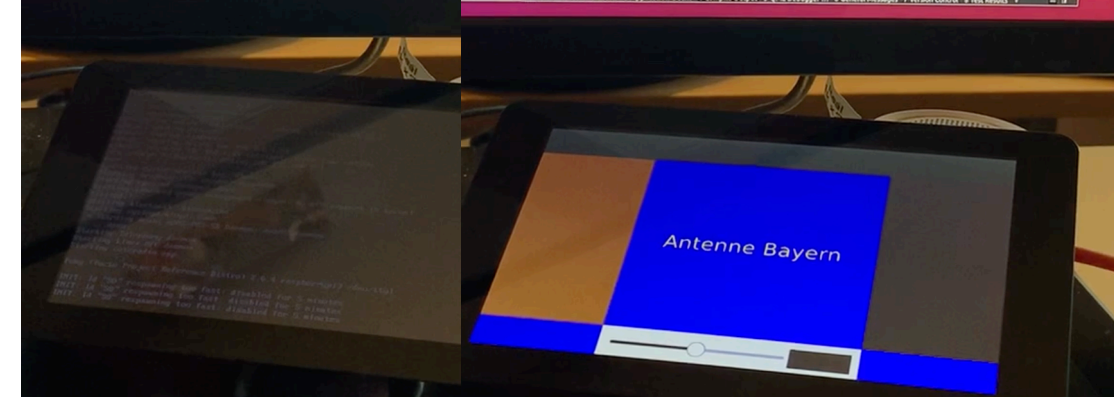
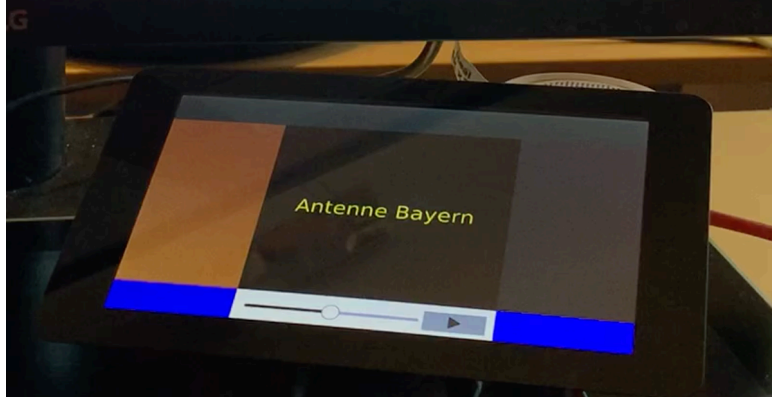
```
17:00:12: The process "/home/burkhard/bin/dr-cmake" exited normally.
17:00:12: Deploy step finished.
17:00:12: Trying to kill "/usr/local/bin/cuteradio" on remote device...
17:00:13: Remote application killed.
17:00:13: Deploy step finished.
17:00:13: Starting: "/usr/bin/rsync" -av ./staging/ root@192.168.1.81:/usr/local
sending incremental file list
bin/
bin/cuteradio
sent 10,063 bytes received 3,363 bytes 8,950.67 bytes/sec
total size is 387,176 speedup is 29.84
17:00:14: The process "/usr/bin/rsync" exited normally.
17:00:14: Elapsed time: 00:05.
```

Ctrl+R

```
anchors.top: statusBar.left
anchors.bottom: bottomBar.top
color: "blue" /*#FAFF70*/ // unmlow yellow

Label {
    anchors.centerIn: parent
    text: "Antenne Bayern"
    color: "yellow"
    font.pixelSize: 36
}
```

```
17:00:42: The process "/home/burkhard/bin/dr-cmake" exited normally.
17:00:42: Starting: "/home/burkhard/bin/dr-cmake" --build
[100] Automatic WOC for target cuteradio/dr-cmake* exists
[100] Built target cuteradio.autogen/dr-cmake* --build
[100] Built target cuteradio.
Install the project...: target cuteradio
-- Install configuration: "Debug"
17:00:44: The process "/home/burkhard/bin/dr-cmake" exited normally.
17:00:44: The remote file system has 461 megabytes of free space, going ahead.
17:00:44: Deploy step finished.
17:00:44: Trying to kill "/usr/local/bin/cuteradio" on remote device...
17:00:45: Remote application killed.
17:00:45: Deploy step finished.
17:00:45: Starting: "/usr/bin/rsync" -av ./staging/ root@192.168.1.81:/usr/local
sending incremental file list
bin/
bin/cuteradio
sent 10,063 bytes received 3,363 bytes 26,852.00 bytes/sec
total size is 387,176 speedup is 29.84
17:00:45: The process "/usr/bin/rsync" exited normally.
17:00:45: Elapsed time: 00:05.
```



References

- [1] [Docker Builds from QtCreator](#). Basis for this talk.
- [2] [Using Docker Containers for Yocto Builds](#). How to install Docker.
- [3] [Qt Embedded Systems – Part 1: Building a Linux Image with Yocto](#). First step to create a Qt SDK.
- [4] [Qt Embedded Systems – Part 2: Building a Qt SDK with Yocto](#). Creates the Qt SDK used in this talk.
- [5] The Qt Company, [Boot to Qt Software Stack](#). Official documentation of Boot2Qt.
- [6] Tino Pyssysalo, [Getting Started with Yocto & eLinux](#). How to install a Qt SDK and start developing an app.



Thank you 😊

Mail: burkhard.stubert@embeddeduse.com

Web: <http://www.embeddeduse.com>



This presentation is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Copyright 2020, Burkhard Stubert