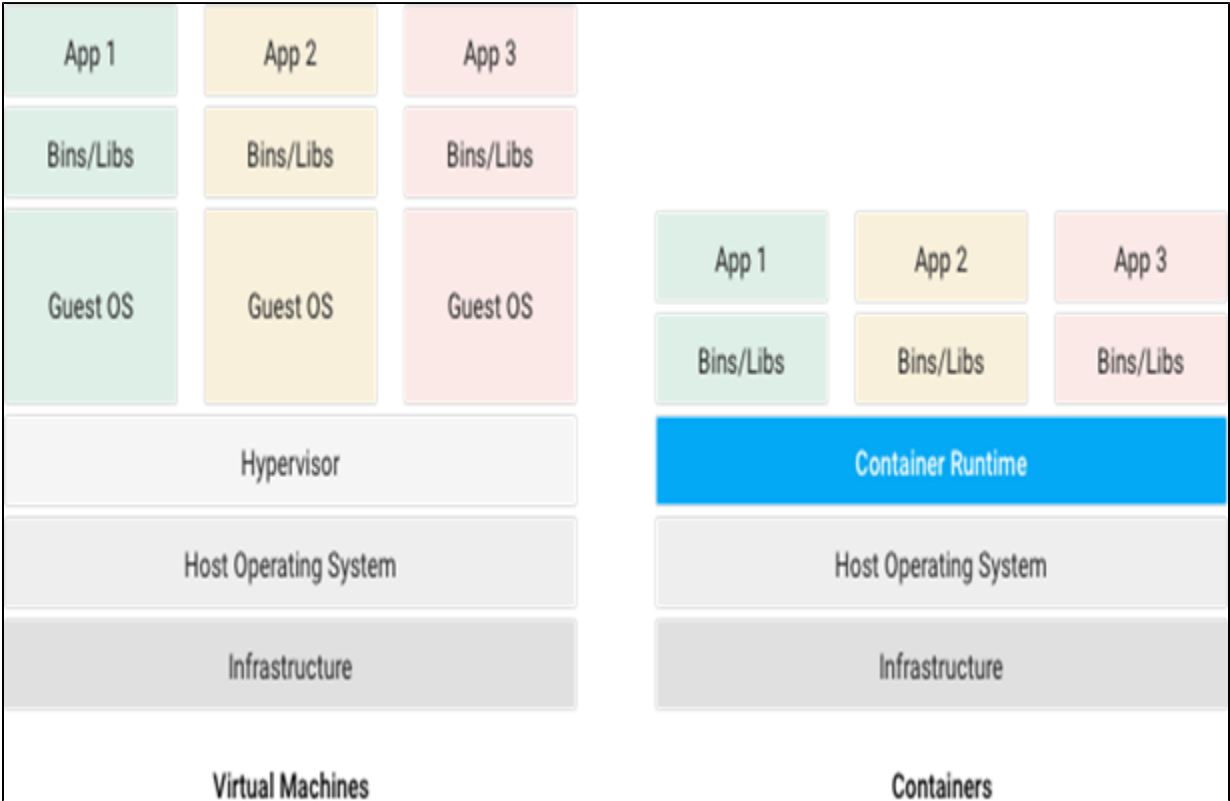


RDK-V RPI containers User Manual

- [Introduction](#)
 - [Why Containers?](#)
 - [Advantages of Containers](#)
- [Implementation details](#)
 - [Containers layer - meta-rdk-containers:](#)
 - [Emulator layer - meta-rdk-bsp-emulator:](#)
 - [New Container generation process:](#)
 - [XML and conf files:](#)
 - [Service files:](#)
- [Implemented containers](#)
 - [Platformcontrol](#)
 - [Rmfstreamer](#)
 - [Rdkbrowser2](#)
- [Building procedure](#)
- [Container verification](#)
- [DEBUG Logs](#)
- [Test cases](#)
- [References](#)

Introduction

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they run. Containers are often compared with virtual machines (VMs). a guest operating system such as Linux or Windows runs on top of a host operating system with virtual access to the underlying hardware. Like virtual machines, containers allow to package your application together with libraries and other dependencies, providing isolated environments for running your software services.



Why Containers?

- Instead of virtualizing the hardware stack, containers virtualize at the operating system level, with multiple containers running atop the OS kernel directly which means containers are more lightweight: they share the OS kernel, start much faster, and use a fraction of the memory compared to booting an entire OS.

- Container consists of an entire run-time environment: an application, plus all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.

Advantages of Containers

- A container may be only tens of megabytes in size, whereas a virtual machine with its own entire operating system may be several gigabytes in size.
- Containerization allows for greater modularity. Rather than run an entire complex application inside a single container, the application can be split in to modules.

Implementation details

Containers layer - meta-rdk-containers:

- Consists of main container image(rdk-generic-hybrid-lxc-image).
- Latest "lxc-container-generator" has been added for container generation at do_rootfs stage.
- Distro feature and latest lxc version updated in qemu86hybsecure.conf.

Emulator layer - meta-rdk-bsp-emulator:

- Added emulator specific package groups and plugins to the container image.

New Container generation process:

This subsection describes how the new container generation process is replacing the earlier process.

- In this process containers will be generated using "lxc-container-generator" recipe, which will use corresponding .xml files to generate containers.
- All dependencies(such as required binaries,libraries,script files) will be provided in each container .XML file.
- For permissions of files "add-users-groups-file-owners-and-permissions.inc" file has been added.
- At rootfs stage containers will be generated in /container path of rootfs.
- Each container will consists of corresponding script (.sh) file for launching that particular container.
- Every process will be launched from corresponding component service file. Single (or) multiple processes can be launched/attached to container.

XML and conf files:

- All required XML and configuration files are placed along with lxc-container-generator recipe in meta-rdk-bsp-emulator layer.

Service files:

- **In platformcontrol container:**

Three service files used for launching corresponding processes inside container (sysmgr.service, irmgr.service and dsmgr.service) .

- **In rmfstreamer container:**

rmfstreamer.service file has been used.

Already these two containers are present in the existing system. we have implemented one more container (rdkbrowser2) which is the replacement of wpelauncher container.

- **In rdkbrowser2 container:**

rdkbrowser2.service file has been used.

Implemented containers

Platformcontrol

- runs sysmgr,irmgr and dsmgr processes.
- sysmgr will be launched in new container using lxc-execute.

- irmgr and dsMgr processes has been attached to same container using lxc-attach.

Rmfstreamer

- runs rmfstreamer.
- rmfstreamer will be launched in new container using lxc-execute.

Rdkbrowser2

- runs rdkbrowser2 browser application inside container.
- westeros will be launched in new container using lxc-execute.
- rdkbrowser2 will be attached to the same container using lxc-attach.

Note: As we are in the plan of bringing APPmanager as default application we are not running rdkbrowser2 service file on boot-up.

Building procedure

- repo init -u <https://code.rdkcentral.com/r/manifests> -b rdk-next -m rdkv-asp-extsrc.xml
- repo sync --no-tags
- source meta-cmf-raspberrypi/setup-environment
- select meta-cmf-raspberrypi/conf/machine/raspberrypirdkhyblxc.conf
- bitbake rdk-generic-hybrid-wpe-lxc-image

Container verification

- pstree can be used to track the list of containers running as below.

```
root@raspberrypirdkhyblxc:/lib/systemd/system# pstree
systemd+-IARMDaemonMain
|-RFCbase.sh---sleep
|-audiocapturemgr
|-dbus-daemon
|-deepSleepMgrMai
|-diskMgrMain
|-dnsmasq
|-dropbear---sh---pstree
|-dropbear---sh
|-dumpLogs.sh
|-klogd
|-lighttpd
|-lircd
|-2*[login---sh]
|-lxc-attach---irmgrMain
|-lxc-attach---dsMgrMain
|-lxc-execute---init.lxc.static---sysMgrMain
|-mfrMgrMain
|-netsrvmgr
|-nlmon
|-ntpd
|-pwrMgrMain
|-rdkbrowser2.sh---lxc-execute---init.lxc.static---rdkbrowser2.sh---rdkbrowser2+-WPENetworkProce
|-rebootSTB.sh---sleep
|-rfStatisticsChe---sleep
|-rmfstreamer.sh---lxc-execute---init.lxc.static---rmfStreamr
|-runAppManager.s+-Spark
|   |-launch_rdkbrows---rdkbrowser2+-WPENetworkProce
|   |   |-WPEWebProcess
|   |-runServiceManag---servicemanagera
|-snmpd
|-syslogd
|-systemd-journal
|-systemd-logind
|-systemd-resolve
|-systemd-timesyn
|-systemd-udev
|-udhcpc
|-westeros
|-wpa_supplicant
|-xdiscovey
```

- `ps -Af | grep lxc` also lists the current running containers.

```
root@raspberrypi4dkhbylxc:~# ps -Af | grep lxc
root      945      1  0 09:37 ?        00:00:00 /usr/bin/lxc-execute -n PLATFORMCONTROL -f /container/PLATFORMCONTROL/conf/lxc.conf --
/usr/bin/sysMgrMain --debugconfig /etc/debug.ini
platform+ 989    945  0 09:37 ?        00:00:00 /init.lxc.static --gid 704 --uid 704 -- /usr/bin/sysMgrMain --debugconfig /etc/debug.i
ni
root      1040   1038  0 09:37 ?        00:00:00 /usr/bin/lxc-execute -n RMFSTREAMER -f /container/RMFSTREAMER/conf/lxc.conf -- /usr/bl
n/rmfStreamer --config /etc/rmfconfig.ini --debugconfig /etc/debug.ini
rmfstre+  1045   1040  0 09:37 ?        00:00:00 /init.lxc.static --gid 705 --uid 705 -- /usr/bin/rmfStreamer --config /etc/rmfconfig.i
ni --debugconfig /etc/debug.ini
root      1116      1  0 09:37 ?        00:00:00 /usr/bin/lxc-attach -n PLATFORMCONTROL -f /container/PLATFORMCONTROL/conf/lxc.conf -u
704 -g 704 -- /usr/bin/lrmMgrMain --debugconfig /etc/debug.ini
root      1146      1  0 09:37 ?        00:00:00 /usr/bin/lxc-attach -n PLATFORMCONTROL -f /container/PLATFORMCONTROL/conf/lxc.conf -u
704 -g 704 -- /usr/bin/dsMgrMain --debugconfig /etc/debug.ini
root      28685  28682  0 10:31 ?        00:00:00 /usr/bin/lxc-execute -n RDKBROWSER2 -f /container/RDKBROWSER2/conf/lxc.conf -- /usr/bl
n/rdkbrowser2.sh
rdkbrow+  28693  28685  0 10:31 ?        00:00:00 /init.lxc.static --gid 706 --uid 706 -- /usr/bin/rdkbrowser2.sh
root      32132  2256  0 10:34 pts/0    00:00:00 grep lxc
root@raspberrypi4dkhbylxc:~#
root@raspberrypi4dkhbylxc:~#
```

DEBUG Logs

- `strace` can give more debug information about containers:

Example:

```
strace -f -o lxc-execute.log /usr/bin/lxc-attach -n PLATFORMCONTROL -f /container/PLATFORMCONTROL/conf/lxc.conf
```

```
-u 704 -g 704 -- /usr/bin/dsMgrMain
```

- `lxc-execute.log` for debugging purpose.

Test cases

- RMFAPP can be used to verify rmfstreamer container.

Example: play http://192.168.2.68:8080/vldms/tuner?ocap_locator=ocap://0x125d

- RDKBROWSER2 can be used to launch any URL.

Example:

```
systemctl start rdkbrowser2.service - user can see the webpage in rdkbrowser2.
```

If user wants to change URL, then we need to enter into this container and need to change `rdkbrowser2.sh` binary as below:

```
systemctl stop rdkbrowser2.service
```

use command:

```
/usr/bin/lxc-execute -n RDKBROWSER2 -f /container/RDKBROWSER2/conf/lxc.conf -- /bin/sh
```

and then change url in `/usr/bin/rdkbrowser2.sh` file inside this container environment.

References

- <https://cloud.google.com/containers/>
- <https://www.cio.com/article/2924995/software/what-are-containers-and-why-do-you-need-them.html>
- <https://linuxcontainers.org/>
- <https://help.ubuntu.com/lts/serverguide/lxc.html>
- <https://linuxcontainers.org/lxc/documentation/>