RDK-B Dobby Demo

On this Page:

Scope

Dobby is a container management tool, to make it easy for other applications to start/stop/monitor containers. It can be thought of as a "Docker for the embedded world".

This page demonstrates **building a sample Java application in RDK framework and running in a container using Dobby** in a RDK-B reference board (Rpi).

Key Links

- Dobby source code: https://github.com/rdkcentral/Dobby
- DAC Working Group: DAC - Advanced Support Program - RDK Central Wiki

Major Dobby Components in RDK-B

- **DobbyDaemon**
  - This is the main Dobby process, which is launched at bootup by systemd. When started, DobbyDaemon registers itself on dbus. It then idles and waits for commands over dbus to start, stop or inspect containers.
- **DobbyTool**
  - CLI to interact with Dobby for developers, and issue commands such as start, stop or info

Build Dobby in RDK-B

- Getting the RDK-B dunfell Code

```
mkdir < workspace_dir>
cd  <workspace_dir>
repo init -u https://code.rdkcentral.com/r/manifests -b dunfell -m rdkb-extsrc.xml
repo sync -j4 --no-clone-bundle
```

- Add dobby,readline, and crun packages to the package recipe file to consider it during the build

```
Path: ./meta-cmf-raspberrypi/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
```

```
:~/DAC_RDKB/meta-cmf-raspberrypi$ git diff recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.
bbappend
diff --git a/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend b/recipes-core
/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
index 87a66a2..43d0f8f 100644
--- a/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
+++ b/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
@@ -6,6 +6,9 @@ RDEPENDS_packagegroup-rdk-ccsp-broadband_append = "\
     libseshat \
     notify-comp \
     start-parodus \
+    dobby \
+    crun \
+    readline \
     \
 "

:~/DAC_RDKB/meta-cmf-raspberrypi$
```

- Add readline dependencies in the dobby recipe file

```
Path: ./meta-rdk/recipes-containers/dobby/dobby-thunderplugin.bb
```

```
:~/DAC_RDKB/meta-rdk$ git diff recipes-containers/dobby/dobby-thunderplugin.bb
diff --git a/recipes-containers/dobby/dobby-thunderplugin.bb b/recipes-containers/dobby/dobby-
thunderplugin.bb
index 877b7fc..592b2db 100644
--- a/recipes-containers/dobby/dobby-thunderplugin.bb
+++ b/recipes-containers/dobby/dobby-thunderplugin.bb
@@ -5,7 +5,8 @@ LIC_FILES_CHKSUM = "file://${WORKDIR}/git/LICENSE;md5=c466d4ab8a68655eb1edf0bf8c

 include dobby.inc

-DEPENDS = "dobby wpeframework-clientlibraries"
+DEPENDS = "dobby"
+DEPENDS += "readline"

 S = "${WORKDIR}/git/rdkPlugins/Thunder"

:~/DAC_RDKB/meta-rdk$
```

- Add the below kernel options to support containerization

```
Path: ./meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi.inc
```

```
:~/DAC_RDKB/meta-raspberrypi$ git diff recipes-kernel/linux/linux-raspberrypi.inc
diff --git a/recipes-kernel/linux/linux-raspberrypi.inc b/recipes-kernel/linux/linux-raspberrypi.inc
index 3219a23..9262374 100644
--- a/recipes-kernel/linux/linux-raspberrypi.inc
+++ b/recipes-kernel/linux/linux-raspberrypi.inc
@@ -35,6 +35,8 @@ CMDLINE_append = ' ${@oe.utils.conditional("ENABLE_KGDB", "1", "kgdboc=serial0,1
 # Disable rpi logo on boot
 CMDLINE_append += ' ${@oe.utils.conditional("DISABLE_RPI_BOOT_LOGO", "1", "logo.nologo", "", d)}'

+CMDLINE_append += "cgroup_enable=cpuset cgroup_enable=memory cgroup_memory=1"
+
 # You can define CMDLINE_DEBUG as "debug" in your local.conf or distro.conf
 # to enable kernel debugging.
 CMDLINE_DEBUG ?= ""
:~/DAC_RDKB/meta-raspberrypi$
```

Build Java Application in RDK-B

- Download the meta-java layer for Java support in RDKB image for Raspberry Pi

```
cd  <workspace_dir>
git clone git://git.yoctoproject.org/meta-java
cd meta-java
git checkout remotes/origin/dunfell
```

- Add a new java application

1. Create a new recipe under meta-rdk-ext/
2. New recipe should have the java application, corresponding .bb and license files

   Example: sample application (HelloWorld.java) under meta-rdk-ext/recipes-java/
3. Add the new recipe to the package group to consider it during the build

   Example : java-helloworld added in meta-cmf-raspberrypi/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend

```
:~/DAC_RDKB/meta-cmf-raspberrypi$ git diff recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.
bbappend
diff --git a/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend b/recipes-core/packagegroups
/packagegroup-rdk-ccsp-broadband.bbappend
index 87a66a2..87f70b8 100644
--- a/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
+++ b/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend
@@ -6,6 +6,11 @@ RDEPENDS_packagegroup-rdk-ccsp-broadband_append = "\
     libseshat \
     notify-comp \
     start-parodus \
+    dobby \
+    crun \
+    readline \
+    openjdk-8 \
+    java-helloworld \
     \
 "

:~/DAC_RDKB/meta-cmf-raspberrypi$
```

- Creating Runtime spec to Containerize the java application

1. Create a configuration file that is config.json for the java application and place it under meta-rdk-ext/recipes-java/<application_folder>

```
:~/DAC_RDKB$ cd meta-rdk-ext/recipes-java/
:~/DAC_RDKB/meta-rdk-ext/recipes-java$ ls
java-helloworld
:~/DAC_RDKB/meta-rdk-ext/recipes-java$ cd java-helloworld/
:~/DAC_RDKB/meta-rdk-ext/recipes-java/java-helloworld$ ls
java-helloworld-1.0  java-helloworld_1.0.bb
:~/DAC_RDKB/meta-rdk-ext/recipes-java/java-helloworld$ cd java-helloworld-1.0
:~/DAC_RDKB/meta-rdk-ext/recipes-java/java-helloworld/java-helloworld-1.0$ ls
config.json  HelloWorld.java  LICENSE
:~/DAC_RDKB/meta-rdk-ext/recipes-java/java-helloworld/java-helloworld-1.0$
```

2. Install the config.json through the Application recipe file

```
SRC_URI += "file://config.json"
```

```
:~/DAC_RDKB/meta-rdk-ext/recipes-java/java-helloworld$ cat java-helloworld_1.0.bb
DESCRIPTION = "Simple Java hello world application"

LICENSE = "Apache-2.0"
LIC_FILES_CHKSUM = "file://LICENSE;md5=e3fc50a88d0a364313df4b21ef20c29e"

RDEPENDS_${PN} = "openjdk-8"

SRC_URI = "file://HelloWorld.java"
SRC_URI += "file://LICENSE"
SRC_URI += "file://config.json"

S = "${WORKDIR}"

inherit java-library

do_compile() {
    mkdir -p build
    javac -d build `find . -name "*.java"`
    fastjar cf ${JARFILENAME} -C build .
}

do_install() {
    install -d ${D}/usr/bin
    install ${S}/build/* ${D}/usr/bin/
    install ${S}/build/HelloWorld.class ${D}/usr/bin/HelloWorld
}

do_install_append() {
    install -d ${D}${sysconfdir}/java_container
    install -d ${D}${sysconfdir}/java_container/rootfs
    install -m 0777 ${WORKDIR}/config.json ${D}${sysconfdir}/java_container
    install -m 0777 ${WORKDIR}/config.json ${D}${sysconfdir}/java_container/rootfs
}


BBCLASSEXTEND = "native"
FILES_${PN} += "/usr/bin/*"
FILES_${PN} += "${sysconfdir}/java_container"
FILES_${PN} += "${sysconfdir}/java_container/rootfs"
FILES_${PN} += "${sysconfdir}/java_container/config.json"
```

Running Dobby Container in Rpi

This provides details on running containerized java applications using dobby in RDK-B Platform (Raspberry Pi)

Steps:

1. Build the source code

```
$ MACHINE=raspberrypi-rdk-broadband source meta-cmf-raspberrypi/setup-environment
$ bitbake rdk-generic-broadband-image
```

2. Flash the image to RPI and copy the container folder to /tmp

```
cp -r /etc/java_container/ /tmp/
chmod +x /tmp/java_container/
chmod -R 744 /tmp/java_container/
```

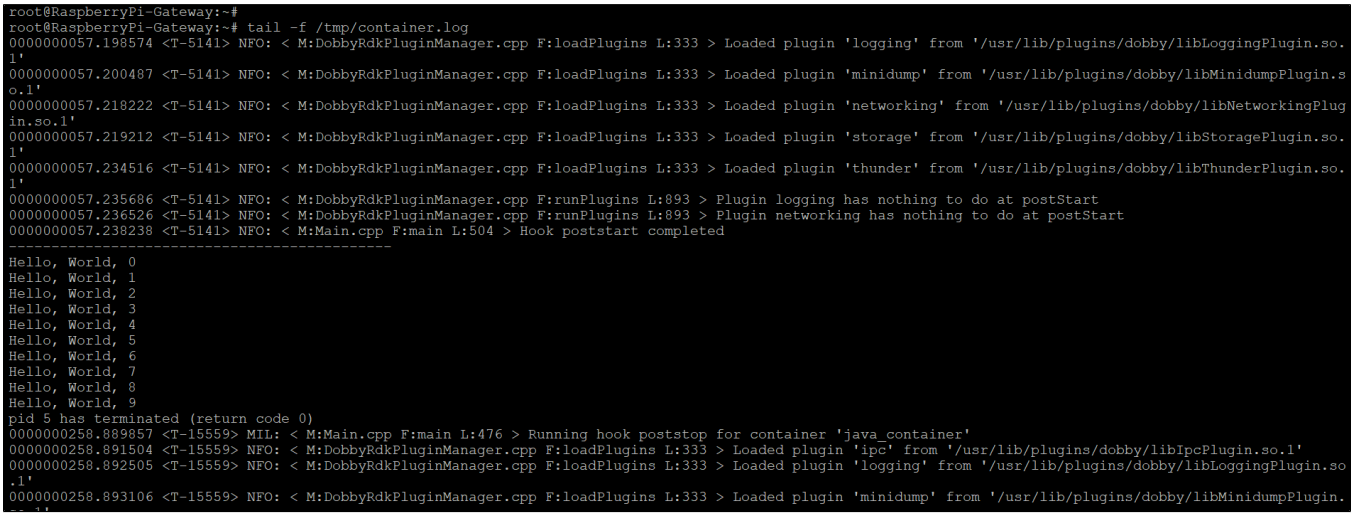
3. Start the container using DobbyTool command

```
/usr/bin/DobbyTool start java_container /tmp/java_container
```

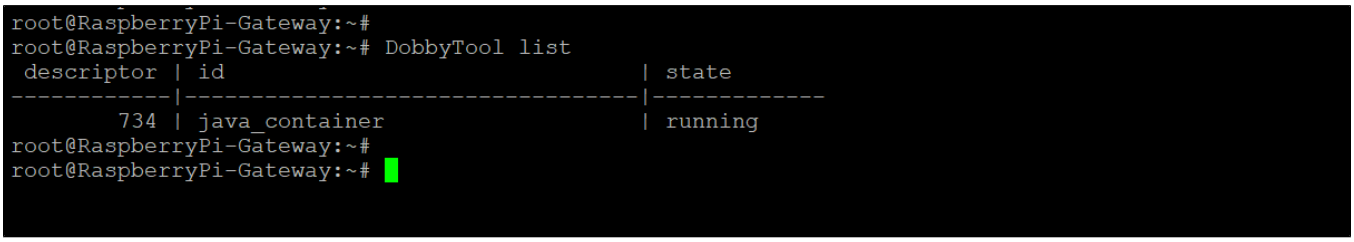
Validation:

- Once the container starts, it will launch the java application. this can be verified through a log

  Log path: /tmp/container.log



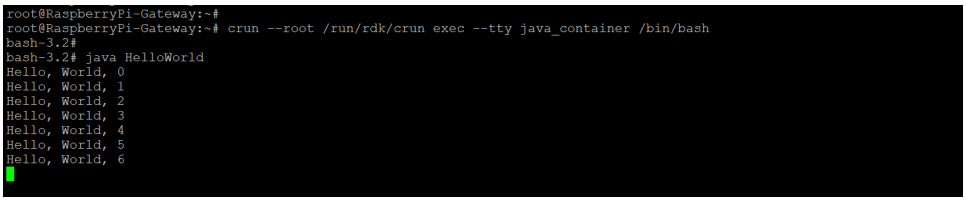
- Container lists can be verified by the command : **DobbyTool list**



- Container access : Run the below command to enter the container console

```
crun --root /run/rdk/crun exec --tty java_container /bin/bash
```

The HelloWorld java application can also be run manually using java command .



Sample Code

Attaching the config,json and Helloworld Java Application recipe file.

recipes-java.zip