Dobby

Key Info

Dobby is a management layer above the open-source crun OCI containerization tool, used for managing and running containers.

- Repository: https://github.com/rdkcentral/Dobby
- Language: C++
- License: Apache 2.0

Overview

Dobby is a container management tool, originally developed by Sky and open-sourced to the RDK community. Dobby is a daemon-based program that makes use of the **crun** runtime and provides a more user-friendly experience for starting/stopping containers, handles container lifecycle management and enhancing the base functionality provided by crun.

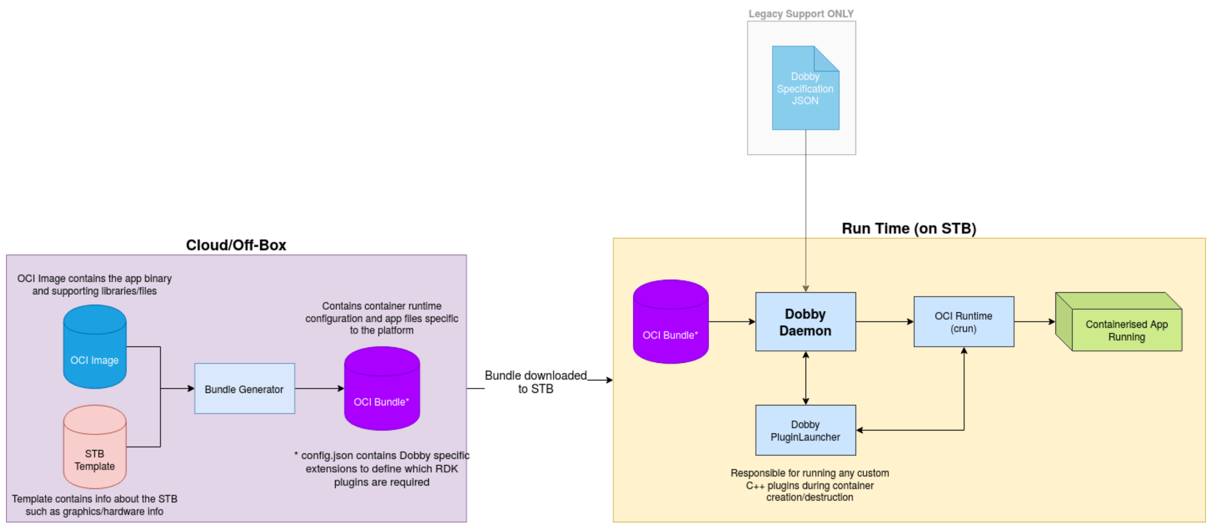
Dobby is designed to be used for starting, stopping and monitoring all containerised applications on an RDK-V device. To start a container with Dobby, provide Dobby with a path to an OCI bundle and a container ID, and Dobby will handle the rest.

Written in C++ and with a plugin-based architecture, Dobby is lightweight, highly expandable and customisable, offering the ability to run custom code at various stages in the container lifecycle to add additional functionality. Out of the box, Dobby provides a number of plugins ready for use to provide commonly needed functionality including:

- Advanced container networking support with NAT and both IPv4 and IPv6 support. Allows for easily adding iptables rules to allow/prevent traffic flow in and out of container
- GPU memory limiting (providing the kernel has the appropriate support)
- Container log management to either files or directly to journald
- Loopback storage mounts to add persistent, isolated storage to containers
- IPC support between containers/host by allowing access to the host dbus inside containers

Plugins are C++ code written against the plugin interface, allowing for operators to easily add additional functionality, or modify existing functionality easily. All the plugins for Dobby can be found in the RDKPlugins directory in the repo here: https://github.com/rdkcentral/Dobby/tree/master/rdkPlugins. Each plugin contains a README file with documentation on its usage. The **TestPlugin** is designed as a minimal reference plugin that can be used as an example for your own development.

Note there is a directory in the Dobby repo called *plugins* which containers legacy plugins required for some platforms. These should **not** be used as a reference for new plugins.



Legacy Support ONLY
Dobby Specification JSON
Run Time (on STB)
Cloud/Off-Box
OCI Image contains the app binary and supporting libraries/files
OCI Image
STB Template
Template contains info about the STB such as graphics/hardware info
Bundle Generator
Contains container runtime configuration and app files specific to the platform
OCI Bundle*
* config.json contains Dobby specific extensions to define which RDK plugins are required
Bundle downloaded to STB
OCI Bundle*
Dobby Daemon
OCI Runtime (crun)
Containerised App Running
Dobby PluginLauncher
Responsible for running any custom C++ plugins during container creation/destruction

Usage

The core component of Dobby is the **DobbyDaemon**. This should be started at STB boot and then listens over dbus for commands. The full dbus API of Dobby can be found here: https://github.com/rdkcentral/Dobby/blob/master/protocol/include/DobbyProtocol.h, although it is not recommended to communicate with Dobby manually over dbus. Instead, Dobby provides abstractions over dbus which are discussed later in this document.

Dobby ships with a systemd unit file called **dobby.service**, which allows Dobby to be started/stopped by systemd.

Command-Line

For debug builds of Dobby, the **DobbyTool** binary is installed on the STB. This is a simple command line app that communicates with Dobby directly over dbus to issue common commands. This is very useful for troubleshooting and testing Dobby.

Example:

```
# Extract OCI bundle to STB
$ tar -xzvf my_app_bundle.tar.gz

# Start container
$ DobbyTool start my_app ./my_app_bundle/
```

Integration with Thunder

OCIContainers Plugin

To control Dobby using a JSON-RPC API, then use the **OCIContainer** Thunder NanoService. This exposes the same functionality as DobbyTool, but in a manner more suited for integration into other code in RDK-V.

On platforms where Thunder is not available, the raw dbus interface can be used instead - OCIContainers only serves as an abstraction over dbus so other components that already communicate with JSON-RPC can easily control Dobby.

Example of OCIContainer to start a container from an OCI bundle:

Request

```
$ curl -X POST http://127.0.0.1:9998/jsonrpc/ -d '{
   "jsonrpc":"2.0",
   "id":3,
   "method":"org.rdk.OCIContainer.1.startContainer",
   "params":{
      "containerId": "testContainer",
      "bundlePath": "[-INSERT BUNDLE PATH-]"
   }
}'
```

Response

```
{
   "jsonrpc":"2.0",
   "id":3,
   "result":{
      "descriptor":257,
      "success":true
   }
}
```

The OCIContainer code contains a README.md file with detailed documentation on its usage - see here: https://github.com/rdkcentral/rdkservices/blob/sprint/2009/OCIContainer/README.md

ProcessContainers

Thunder includes the ability to run plugins in containers as well as the traditional in/out of process modes. Thunder offers different back-ends for running Thunder plugins in containers, one of which is Dobby. This mode will communicate with Dobby to start/stop Thunder plugins inside Dobby containers. This provides the advantage of meaning Dobby is responsible for running all types of container on the STB, from native apps to Thunder plugins.

To enable this, compile Thunder with the following CMake flags:

- -DPROCESS_CONTAINERS=ON
- -DPROCESSCONTAINERS_DOBBY=ON

Then for the plugin(s) you wish to run in a container, set the execution mode to "container" in the plugin configuration file.

Source Code/Development

Dobby source code lives on the RDKCentral GitHub here: https://github.com/rdkcentral/Dobby and is Apache 2.0 licenced.

To develop Dobby, it is recommended to use Ubuntu 16.04. Dobby ships with a Vagrant file to create a development environment. More information is available here: https://github.com/rdkcentral/Dobby/tree/master/develop

If you would like to contribute code to this project you can do so through GitHub by forking the repository and sending a pull request. Before RDK accepts your code into the project you must sign the RDK Contributor License Agreement (CLA).