LHCb Dockerized Build Environment

M Clemencic¹, M Belin², J Closier¹ and B Couturier¹

¹ CERN, CH-1211 Geneva 23, Switzerland
² ISIMA, 1 rue de la Chebarde, 63178 Aubiere CEDEX, France
E-mail: marco.clemencic@cern.ch

Abstract. Used as lightweight virtual machines or as enhanced chroot environments, Linux containers, and in particular the Docker abstraction over them, are more and more popular in the virtualization communities.

The LHCb Core Software team decided to investigate how to use Docker containers to provide stable and reliable build environments for the different supported platforms, including the obsolete ones which cannot be installed on modern hardware, to be used in integration builds, releases and by any developer.

We present here the techniques and procedures set up to define and maintain the Docker images and how these images can be used to develop on modern Linux distributions for platforms otherwise not accessible.

1. Introduction

LHCb started the investigation on Docker containers[1] in an attempt to optimize the load on the build machines used in the LHCb Nightly Builds System[2]. The problem was that we are building our software stack in several configurations on different reference platforms, and the load on each platform could not be estimated, making it impossible to correctly choose the number of build machines to install with which operating system.

The idea was that we could use a uniform pool of build machines and wrap builds and tests of our software projects in Docker containers mimicking the reference platforms.

Very early in the development of this solution we realized that the Docker images we were preparing could be extended to provide a ready to use development environment that any LHCb developer could use to work on the various reference platforms from a unique machine.

2. Requirements

To support the use case of our Nightly Builds System we needed one Docker image per supported platform, with all external libraries preinstalled, that had access to some special paths of the host filesystem, like the mounted CernVM-FS[3] volumes and the scratch space mount point used for the build.

In addition to the requirements for the Nightly Builds System, to work as a portable development environment our Docker images had to integrate with the user system. In particular the running containers must

- allow persistent user settings
- provide access to the user development workspace on the host
• use the same UNIX user id as the host user, to prevent creation of files not accessible by the host user

3. Docker Wrapper Script
The command `docker run`, used to start a Docker container from an image, allows many customizations of the container, like the host directories that should be visible inside the container, but it becomes soon unpractical to use if there are a few options that must be always given, like, in our case, the request to share with the container the host directories where LHCb software is deployed.

To circumvent the problem, we relieved the user from the need to call directly `docker run` with all its options by preparing a wrapper script, `lb-docker-run`, that takes care of preparing the arguments to `docker run`. For example, `lb-docker-run` always tries to share the special mount point `/cvmfs/lhcb.cern.ch` with the container, reporting a helpful error message if that is not possible. It also takes care, if requested, to set up a persistent home directory for the user account inside the container, mapped to a conventional location in the home directory of the host user.

The use of a wrapper around the low level command `docker run` provides a way to endlessly tune the behavior of the containers, allowing us to develop multipurpose images.

4. Custom Docker Images
Docker allows stacking of images, so that a base image (e.g. a minimal system) can be shared among several specialized images.

Our custom Docker images are built on top of the existing base images provided by CERN at https://hub.docker.com/u/cern/, with the addition of a layer where we install all system packages required to build the LHCb software (e.g. development libraries).

Although the added system packages would be enough to build our software, we add another layer to inject custom scripts needed to fine tune the behavior of the container after it is started (see next section).

5. Docker Container Start Script
The user account (name and id) owning the main process of a Docker container is defined in the Docker image, but we need that the process is owned by a user with the same name and id of the host user that started the container. The only way to achieve this behavior is to use a starter script, executed as root in the container, that creates the required user and switches to it.

Because of the isolation guaranteed by the containers, it’s not possible for a process running in the container to gather information about the user of the host, so we leveraged on our Docker wrapper script to pass the needed details from the host to the container via some special environment variables.

6. Conclusions
We have been using our Docker images in the Nightly Builds System for several months to provide native builds for all the reference platforms using a uniform pool of CentOS 7 build machines.

The possibility of using Docker containers as portable development environments proved very useful in the context of the Hackathons for the LHCb Upgrade Software, where the uniform, ready to use and tested environment made it possible for developers to be ready to work on the assigned tasks in minutes.

Since the Docker containers are used not only to build, but also to test the LHCb software projects in the Nightly Builds System, they can be used to run any LHCb reconstruction or
analysis application, so we are investigating the possibility of using these containers in the context of Analysis Preservation studies.

References
[1] Docker web page URL https://www.docker.com/
[2] Clemencic M and Couturier B 2014 *J.Phys.Conf.Ser.* 513 052007
[3] Blomer J, Aguado Sanchez C, Buncic P and Harutyunyan A 2011 *J.Phys.Conf.Ser.* 331 042003