Jabberwocky Software Design Document

Ian Orzel
Dylan McDougall

October 5, 2022

Contents

Chapter 1

Introduction

1.1 Purpose

This document provides the software design description for the Virtual Development Environment in a Box project. This document will describe the design and architecture of the project in relation to the project's goals.

1.2 Intended Audience

This document is intended to be read by faculty who plan to, have an interest in, or currently are incorporating this project into one or more of their courses, students who are taking or plan to take courses taught by professors who incorporate or are planning to incorporate this project into their courses, project sponsors, project clients, and project developers.

1.3 Scope

The goal of the Virtual Development Environment in a Box project is to create a platform that provides a virtual environment for students to run software that requires specific system setups (i.e., hardware, architecture, software stack) that is convenient to the user. This platform should be simple to use for both students and faculty, should be scalable, and should support multiple architectures. Specifically, this platform should have support for the Compiler Theory class, allowing for students to easily test their compilers on the platform. Our motivation is to eliminate some of the difficulty that comes with setting up certain development tools that, for instance, require exotic architectures, hardware, or software configurations for use in several classes at Florida Tech, such as Compiler Theory and Operating Systems Concepts. This creates many inconveniences for both students and faculty alike, as students have to figure out how to get the systems properly set up and faculty have to assist students with this setup process.

1.4 Overview

This document provides an overall system architecture diagram, a mock-up of the command line interface, an architecture diagram for the container sharing functionality, and descriptions of various non-self-explanatory functions.

1.5 Definitions

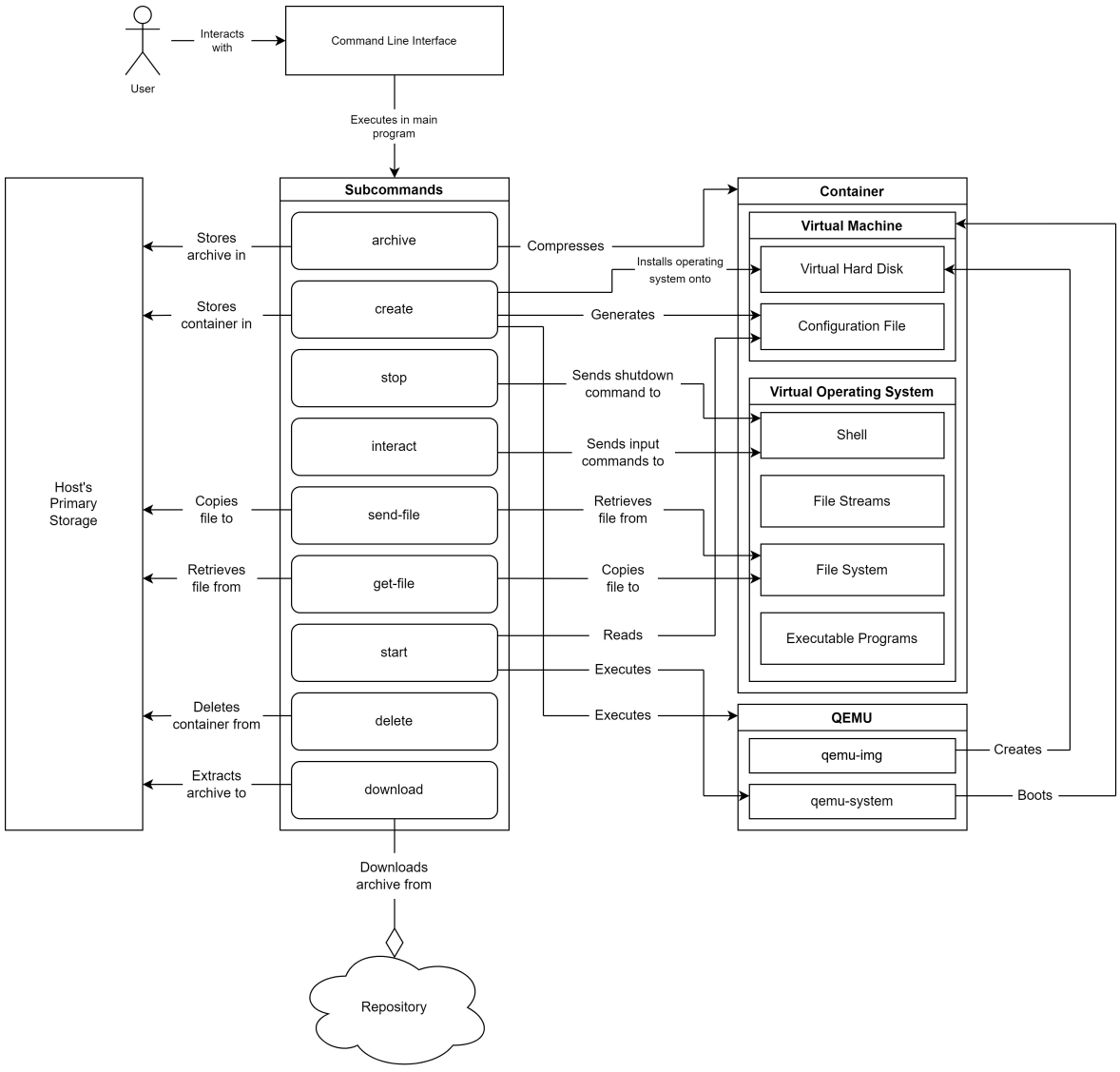
- A host refers to the system which a container or containers is stored on and is dependant on the computing resources of.
- QEMU is an open source machine emulator and virtualizer.

- A container is a QEMU virtual machine composed of at least a bootable virtual hard disk and a defined configuration capable of running certain software, with the intention that the virtual machine will be distributed among multiple people so that they may be able to use the same software within the same virtual environment.
- A repository is an HTTP file server consisting of an arbitrary number of container archives and a single index file providing the locations of each archive on the server.

Chapter 2

System Architecture

2.1 System Architecture Diagram



Chapter 3

Command Line Interface

3.1 Mock-Up

The following is a mock-up for the command line interface portion of the project. This mock-up assumes that the name of the binary file which executes the main program is `vde`, which may not necessarily be the case in the final product. The command line interface is organized into a series of subcommands, which when called perform the functions defined in the mock-up.

```
usage:  vde [subcommand] {args}

help
Print this help page

interact [container name]
Open the shell of the container

interact [container name] {args}
Execute provided arguments in the container's shell

start [container name]
Boots a container and allows it to idle in the background

stop (container name)
Powers off a container or all containers if no container name is provided

send-file [container name] [path to source] [path to destination]
Copy a file from the host to a container

get-file [container name] [path to source] [path to destination]
Copy a file from a container to the host

install [path to archive] (new name)
Installs a container archive onto your computer
If a (new name) is provided, rename the container to the new name

download [container name] (new name)
Downloads a container to your computer
If a (new name) is provided, rename the container to the new name

archive [container name] [path to destination]
Sends the container to a distributable archive.
```

```
add-repo [repository URL]
remove-repo [repository URL]
Adds or removes a repository

update-repo (repository URL)
Download index files for a repository
If no (repository URL) is provided, update all repositories

delete [container name]
Deletes a container from the file system

create
Starts the container creation wizard
```

3.1.1 Interact

When called with no arguments other than the container name, the interact subcommand logs the user into the relevant virtual operating system's shell. When called with arguments, it passes those arguments to the relevant virtual operating system's shell in the host's current working directory.

3.1.2 Managing Files

The user may wish to copy files to and from a container. The send-file subcommand provides the ability to copy a file from the host to a container. The get-file subcommand provides the ability to copy a file to the host from a container.

3.1.3 Starting and Stopping

Containers may run idle in the background in order to prevent repeated power cycles within the virtual machine, which may consume a lot of time depending on the nature of the container. The user can manage this manually by using the start and stop subcommands. 'Starting' a container boots its virtual hard disk in a QEMU virtual machine and allows the virtual machine to idle. 'Stopping' a container sends a shutdown command to the virtual machine. This will close any shell sessions and kill any processes the user may be running inside the container.

3.1.4 Installing, Archiving, Deleting, and Downloading

Containers are saved to the user's primary storage device. Users may install containers from a container archive or download them from a repository. Users may delete containers from storage if necessary. Users may archive containers, which will export them into a distributable format that can later be installed.

3.1.5 Managing Repositories

When downloading containers, it will search the known repositories for a container of the provided name and download and install it if it exists. Users may add an arbitrary number of repositories. Once a repository is added or removed, the user should run the `-update-repos` subcommand in order to refresh the cache.

Chapter 4

Containers

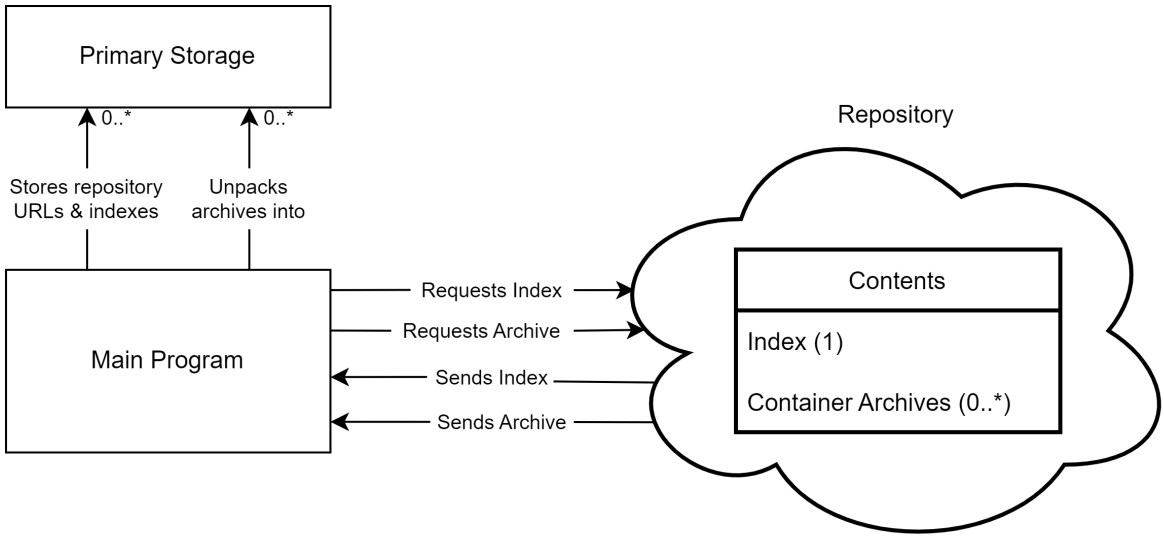
4.1 Definition

Containers are stored on the local machine's primary storage as a virtual hard disk and a configuration file. When starting a container, the main program will read the configuration file, call the appropriate QEMU system, translate the configuration file to a QEMU shell command, and execute the command. A user may have multiple containers installed on their system, each of which can be distinguished by its unique container name. Multiple containers may be allowed to run at the same time, and the user may access multiple containers at a time using different terminals.

4.2 Container Distribution and Repositories

The user may choose to add repositories to download preconfigured containers from. These repositories will be simple HTTP file servers containing compressed archives of containers and an index of the locations of said archives. The use of a repository is not necessary for the main program to function; however, their purpose is to provide a set of convenient central hubs for container distribution.

4.2.1 Architecture Diagram Pertaining To Container Distribution



4.3 Container Creation

Containers can either be created manually or by using the container creation wizard. Manually creating a container is the process of providing a preexisting virtual hard disk image and manually writing

a configuration file for it. This method of creating containers is not recommended for most users. The container creation wizard, which is invoked by executing the `create` subcommand in the main program, will guide the user though creating a preconfigured container.

The container creation wizard will operate as follows:

- The wizard will prompt the user to provide virtual machine specifications. (ie, name, virtual hard disk size, RAM, architecture, etc.)
- The wizard will prompt the user to provide any Debian packages to be installed after the initial installation. These packages must either be available in the standard Debian main, contrib, and non-free repositories or be available on the host's file system.
- The wizard will prompt the user to provide any shell scripts to be executed after all packages are installed.
- The wizard will create a virtual hard disk and generate a configuration file.
- The wizard will install the standard Debian base to the virtual hard disk.
- The wizard will install the desired packages.
- The wizard will execute the desired scripts.
- The wizard will send the virtual hard disk and configuration file to an archive.

4.4 Configuration File

The configuration file will describe to the main program how to execute the container using QEMU. The configuration file will use the JSON format. It will have the following contents:

- A string value `name`.
- A string value `architecture`.
- An object value `flags` where the `flags` object provides the flags to be used in the call to qemu-system. (ie, {`"flag": "value"`} will become `-flag value`.)

4.5 Container Storage

Containers will be stored on the host's primary storage in a directory named `.containers`, the location of which determined at the time of installation. Each container will have its own directory inside of the `.containers` directory which shares the name of the container and will contain of all the files that comprise the container.