Containerizing an eTextbook Infrastructure

Alexander Hicks
Virginia Tech
Blacksburg, United States
alexhicks@vt.edu

Clifford A. Shaffer
Virginia Tech
Blacksburg, United States
shaffer@vt.edu

ABSTRACT

The CS Education community has developed many educational tools in recent years, such as interactive exercises. Often the developer makes them freely available for use, hosted on their own server, and usually they are directly accessible within the instructor's LMS through the LTI protocol. As convenient as this can be, instructors using these third-party tools for their courses can experience issues related to data access and privacy concerns. The tools typically collect clickstream data on student use. But they might not make it easy for the instructor to access these data, and the institution might be concerned about privacy violations. While the developers might allow and even support local installation of the tool, this can be a difficult process unless the tool carefully designed for third-party installation. And integration of small tools within larger frameworks (like a type of interactive exercise within an eTextbook framework) is also difficult without proper design.

This paper describes an ongoing containerization effort for the OpenDSA eTextbook project. Our goal is both to serve our needs by creating an easier-to-manage decomposition of the many tools and sub-servers required by this complex system, and also to provide an easily installable production environment that instructors can run locally. This new system provides better access to developer-level data analysis tools and potentially removes many FERPA-related privacy concerns. We also describe our efforts to integrate Caliper Analytics into OpenDSA to expand the data collection and analysis services. We hope that our containerization architecture can help provide a roadmap for similar projects to follow.

Keywords

Containerization, Caliper Analytics, Learning Tools Interoperability

1. INTRODUCTION

Currently, the OpenDSA eTextbook project gives instructors and students access to a hosted version of the application for free. But this is not sufficient for all institutional users, or even for our own installation as demand and subsequent computational load grows. In principle, OpenDSA (being an open-source project) has supported users or universities seeking to deploy their own copy. But this could only be done by following a complex set of instructions.

This paper describes our work to simplify the installation process. We have many motivations for this. At first we wanted to allow instructors to host this tool on their own, both to reduce our server load and to avoid complications when their University requires strong privacy restrictions. Outsourcing in this way also lets them gain access to additional student data that is being collected. So, this work began as an effort to create a new development environment for OpenDSA that would work on any platform. Once we saw the benefits of this new development environment, and realized the benefits of keeping our development and production environments as similar as possible, we started creating the production environment described below. In particular, we have now made it easy for the many students who work on the project to easily spin up a complete development environment for any part of the system, a great savings in effort for both those students and the project managers.

The OpenDSA system has two major parts that needed to be containerized: a front-end content delivery server and a back-end LTI and data collection server [1, 11]. In this paper, the front-end server will be referred to as OpenDSA and the back-end server is called OpenDSA-LTI or LTI. These two parts work together to serve content using the LTI protocol [2]. These two systems work together to serve content to students. OpenDSA compiles the book instances from RST files into html files (using Sphinx [5]) that can be displayed in a user's browser. Then OpenDSA-LTI communicates with the learning management system (LMS) to serve content files located on the server to the student.

2. BACKGROUND

Containers are lightweight and portable packages of software than can run anywhere their runtime is supported [8]. There are several different container providers, but we selected Docker as our container platform due to its familiarity and its current status as an industry standard for containerization [10].

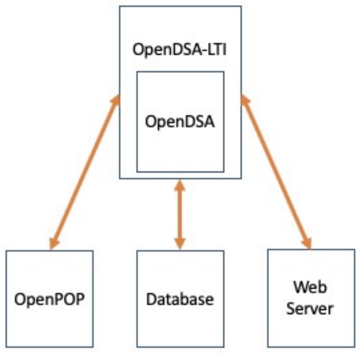

Figure 1: Previous OpenDSA System Architecture

A containerized application has several benefits over traditionally packaged applications including portability and ease of setup. With Docker, the only program a user will need to install on their server is Docker, and everything else will be installed and run within Docker containers using built-in orchestration through Docker-Compose. This makes it particularly easy for new developers to start working on copies of the system.

3. ARCHITECTURE

In order to containerize the existing OpenDSA architecture, we first had to investigate how the technologies that make up the current stack could be split and containerized. The current stack consists primarily of a Ruby on Rails application (OpenDSA-LTI), a MySQL database, and an Nginx web server along with several other tools and requirements.

Since OpenDSA and OpenDSA-LTI work by serving content files to the LMS, these containers, described below in Section 3.1, need to share a filesystem. Originally, these two processes ran on the same host natively, so there were no issues with where files were located as long as both components were installed in the correct location [11]. As shown in figure 1, the previous OpenDSA system architecture had a copy of OpenDSA (the content part) within OpenDSA-LTI (the server), and maintained external connections on the host machine with the database and the web server. Additionally, a supporting visualization tool named OpenPOP ran as a separate server on the same machine, but not in a dedicated environment. Under the new organization shown in figure 2, all of the systems are running in Docker on the same host, meaning we no longer have to worry about compatibility between OpenDSA and OpenPOP's dependencies. All of the networking is handled by Docker rather than an Nginx configuration file. In the first attempt at containerizing this system, these two processes were combined into only one Docker container, which was functional but slow because the container required three language runtimes and thus ran many different processes. In order to split this container into two, we created a REST API around the OpenDSA functionality that handles book compilation. This API is accessed by the LTI container using HTTP requests and places the compiled books in the Docker shared directory that LTI can access.

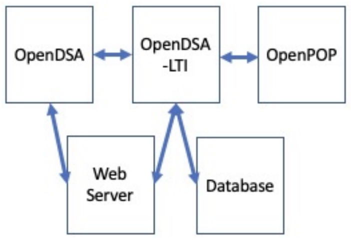

Figure 2: Containerized OpenDSA System Architecture

The OpenDSA production environment currently consists of five containers: OpenDSA, OpenDSA-LTI, OpenPOP, a database, and a web server. All of these images that we created are published and accessible on Docker Hub at `https://hub.docker.com/orgs/opendsa/repositories` and described in section 3.1.

3.1 Component Containers

The OpenDSA container includes the OpenDSA repository (`https://github.com/OpenDSA/OpenDSA`). Python scripts support compiling books to a specific location on the shared filesystem that OpenDSA will serve using the LTI container. This repository also stores configurations for the books to be compiled from. The OpenDSA-LTI container contains the code from `https://github.com/OpenDSA/OpenDSA-LTI` in a Ruby container designed by Bitnami for Rails production deployments [6]. This container consists of the Rails server, and delivers the compiled content as required by an LMS such as Canvas using the LTI protocol.

OpenPOP's (`https://github.com/OpenDSA/OpenPOP`) container is structured in a way similar to the OpenDSA-LTI container. OpenPOP is an external tool that is used by OpenDSA through a different external tool, CodeWorkout, and this work includes OpenPOP in the installation rather than keeping it separate as it was done previously. OpenDSA uses MySQL as the database and provides that in a separate container.

In order to avoid negative impacts from using a database in an ephemeral container, we use a Docker volume to mount the database onto the host filesystem to preserve the data if the container fails, and to allow the container to be restarted separately for routine maintence. Finally, we include two options for a web server container. For users who can acquire their own SSL certificates, this stack will have an option to import those certificates and use them in a Traefik web server [7]. For users that do not have their own SSL certificates, there is an option to use Let's Encrypt to automate their SSL encryption, as long as they provide a domain [9]. Both of these options have minimal overhead required to set up and help keep the overall set up effort for the complete application to a minimum.

4. DISCUSSION AND FUTURE WORK

As the CS Education community grows, there is an increasing availability both for individual tools that need to be integrated into broader systems, and also integrated systems such as OpenDSA. In both cases, containerization can extend their use by other parties. As the number of these systems grow, the CS Education community needs to address how to keep all of these tools interoperable, and more available to instructors. We hope to provide a roadmap for other tools to follow to extend their reach. As issues around data access and privacy become more prevalent, hosting software on premises becomes more attractive to administrators, and the work presented here provides one method for doing so. In addition to the privacy benefits, the roadmap explored in this paper provides a centralized data repository that is closer to the consumers and allows for greater access and additional analysis by the instructors. Containerization also provides a cloud-based install option using AWS or a more robust container orchestration system such as Kubernetes as a possibility.

While the efforts described in section 3 are ongoing, there are several other future enhancements planned for the platform. Currently, OpenDSA uses LTI 1.1, which is due to be deprecated and will be upgraded into LTI 1.3. Along with providing additional security, LTI 1.3 adds a new feature set and an opportunity to connect OpenDSA with other learning tools and platforms such as Caliper Analytics [3, 4]. Integrating Caliper will allow OpenDSA to collect additional, standardized, data on student progress that can be shared when appropriate across platforms that use the LTI system.

Further work would include breaking the Rails application into a more containerizable microservice architecture. Currently, OpenDSA-LTI is a monolith that cannot take advantage of some benefits of containerization, including scalability.

5. REFERENCES

[1] 12. OpenDSA-Introduction — OpenDSA System Documentation. `https://opendsa.readthedocs.io/en/latest/Introduction.html`.

[2] Basic Overview of How LTI works | IMS Global Learning Consortium. `https://www.imsglobal.org/basic-overview-how-lti-works`.

[3] Caliper Analytics | IMS Global Learning Consortium. https://www.imsglobal.org/activity/caliper.

[4] LTI Security Announcement and Deprecation Schedule | IMS Global Learning Consortium. https://www.imsglobal.org/lti-security-announcement-and-deprecation-schedule.

[5] Overview — Sphinx documentation. https://www.sphinx-doc.org/en/master/.

[6] Secure and Optimize a Rails Web Application with Bitnami's Production Containers. https://docs.bitnami.com/tutorials/secure-optimize-rails-application-bitnami-ruby-production/.

[7] Traefik. https://doc.traefik.io/traefik/.

[8] What is a Container? | App Containerization | Docker. https://www.docker.com/resources/what-container.

[9] Linuxserver/docker-swag. LinuxServer.io, May 2021.

[10] Dave Bartoletti and Charlie Dai. The Forrester New Wave™: Enterprise Container Platform Software Suites, Q4 2018, 2020. `https://www.docker.com/resources/report/the-forrester-wave-enterprise-container-platform-software-suites-2018`, accessed on September 15, 2020.

[11] H. L. Shahin. *Design and Implementation of OpenDSA Interoperable Infrastructure*. Thesis, Virginia Tech, Aug. 2017.