NORTHWEST NAZARENE UNIVERSITY

Using Docker in an Educational Environment

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Corbin Hunter Whitton

2021

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE (or ARTS)

Corbin Hunter Whitton

2021

Using Docker in an Educational Environment

Author: Corbin Whitton

Corbin Hunter Whitton

Approved: Kevin S McCarty

Kevin McCarty, Ph.D., Department of Mathematics and Computer Science,
Faculty Advisor

Approved: Christopher Patterson

Christopher Patterson, Information Technology, Northwest Nazarene University

Second Reader

Approved: Barry Myers

Barry L. Myers, Ph.D., Chair,

Department of Mathematics & Computer Science

ABSTRACT

Using Docker in an Educational Environment to Deploy Services for Computer Scientists and Students.

WHITTON, CORBIN (Department of Mathematics and Computer Science), MYERS, DR. BARRY (Department of Mathematics and Computer Science).

Northwestern Nazarene University Computer Science seeks to educate students in the latest tools and technologies used in today's computing environments. As these environments are varied, the computer lab requires several different virtual systems and services to cover a wide range of student/professor needs, from Big Data to Cyber Security. My project intends to extend the university's current educational capacity via the addition of several new flexible but highly specific and robust Linux-based environments that utilize Docker to manage and deploy various services.

The goal of this project was to create an environment for dynamic learning. To do so, services such as databases, web servers, web IDE's, git services, and more need to be researched and made available for quick and easy deployment. First, knowing about Linux and Docker is a must, then the project implementation can begin; once it is done, a future roadmap is to be devised.

ACKNOWLEDGEMENTS

I would like to thank my family and friends who have stuck with me through the years, I will always be indebted to them. To Dale Hamilton Ph.D., Barry Myers Ph.D, and Kevin McCarty Ph.D., I want to express my thanks for your insight, wisdom, and council through college and my career. I would also like to thank the Open Source Community for your software, solutions, and support, I would not have been able to do my project without them.

TABLE OF CONTENTS

Intro 1
Background 1
History 1
What Are Containers 2
What is Docker 2
What Are Swarms (Clusters) 2
Networking 3
Managing Storage (Volumes) 3
Questions 4
Why not Kubernetes? 4
Why not Python’s venv? 4
Why not use a GUI on the Linux distribution? 4
Implementation 4
Results 6
Figures 6
Server Management Dashboard 6
Portainer Dashboard 6
Portainer Templates 6
File Server 7
Conclusion 7
Future Work 8
Ticketing System 8
LDAP Authentication 8
Domain Controller 8
The Ability for External Users to Authenticate 8
Mail Server Connection (cs@nnu.edu) 8
References 9

Intro

As a computer science student, and IT professional, I know how important it is to have quick access to services and technology. Not every student is lucky enough to be able to quickly deploy a server on their Homelab or a server they have access to experiment with. This reason is why I decided to help NNU students and faculty speedily and efficiently deploy software services that can be used across campus. I wanted to create a secure environment in which students can learn new technology that will benefit them in their careers; I also wanted to create an easily manageable system.

The biggest struggle I see in the workforce is a lack of real-world experience. Higher learning equips students with the knowledge to accomplish great things but rarely prepares them for the professional environment. Any professional working environment is built from services deployed internally or externally that employees, clients, and management use to ensure the best product is being produced. The ideal goal of secondary education is to prepare students for the workforce, and there is no better way than to expose them to tools they will use daily, such as virtualization, containerization, Linux, and Docker.

Background

History

Ten years ago, deploying services quickly and efficiently would have been a distant dream. It would have required either multiple physical machines (which can be quite expensive to buy, let alone run), or utilize our physical hardware more efficiently using a technology first postulated in the 1960s, virtualization (*1.1.1. brief history of virtualization,* 2012).

Virtualized environments are preferred than installing the software directly on the server hardware, as virtual environments allow for near-total isolation from the host machine so if the virtualized system is corrupted or malware-infested, only the virtual machine (VM) is compromised and not the entire physical system. To install a service, IT professionals would create the VM, allocate resources, install the operating system (OS), remote into the virtualized system, install the prerequisites, and eventually install the program, and allow it through the firewall on the VM and the host.

VMs can also be easily backed up to a safe location, so in cases of a server-room fire, or a dissatisfied employee, the data is safe and secure. VM templates can also be created, allowing a single operating system image to be deployed with minimal configuration and setup. Unfortunately, all software on a virtual machine (VM) needs to be virtualized, including the operating system. Another shortcoming of a VM is unless the VM is being executed under a type 1 hypervisor, the performance of using the bare hardware cannot be utilized; instead, memory, storage, GPUs and CPUs all have to be emulated to a degree, which plays a significant part in the performance of a service.

Modern-day Hypervisors, the software that allows for virtualized systems, are starting to enable VMs to access the hardware directly; this greatly increases the performance of the virtualized systems, but, in most cases, the hardware can only be allocated once, and, thus, used, by one VM at a time.

The optimal service deployment environment would need to allow for complete isolation of each piece of software, allow bare-metal access, have minimal operating system images or a small shared read-only operating system image, and can quickly deploy services.

What Are Containers

Containers are similar to virtual machines; however, they are much lighter weight to run. Containerized applications run closer to the machine as the CPU and hard drive does not need to be virtualized. They run alongside the host operating system and only require a lightweight translation layer. They allow for near-total isolation of services and run off a lightweight operating system image, usually a Linux-based OS.

What is Docker

Docker is a piece of software used by developers, system admins, and IT professionals to run applications in a secure “containerized” environment. It is like virtualization software, as it allows for complete separation of services and separate environments with complete control over the allocation of resources, access to files, and networking. Docker applications can be quickly pulled from the cloud or compiled locally and ran; they are ideal for a CI/CD environment (*Docker overview,* 2021).

What Are Swarms (Clusters)

Swarms are a network of Docker daemons. Each swarm must have a leader, at least one manager, and can have many worker nodes. It is ideal for there to be at least two manager nodes for redundancy's sake. Swarms have an overlay network that allows them to communicate with each other and share data. Applications deployed to a swarm are called services. Services can have two scheduling modes, global, in which case a container is deployed to every node that runs the same container, and replicated, in which only *x* number of containers can be deployed.

When a leader node goes down, the manager nodes elect a new leader from the pool of managers to run the swarm, which is why it is important to have more than one manager in a Docker swarm.

A place where this can be used is for load balancing websites. The website would be run as a globally scheduled service, every node in the swarm will run the website. Then, a reverse proxy and load balancer will be duplicated once across the swarm so that the website(s) can be accessed from the same IP address.

Networking

Using Docker, it is possible to completely virtualize networking, allowing for secure communication between the containers, or to the outside world. There are 3 main types of networking drivers that are used when working with Docker (*Container networking,* 2021).

Bridges

Bridges allow containers to access the host network virtually, like virtual networks in standard virtualization software.

Overlays

Overlays are networks that are shared between nodes in a docker swarm. They are used to communicate between services and containers as if they are on the same network.

Host

The host network adapter allows for containers to access the host's network adapter directly.

Managing Storage (Volumes)

When a Docker container is started, it will not have any access to files from the host OS unless given to it. Docker allows for files/directories to be mounted for the container to use allowing for data to persist even if the container is destroyed.

Docker can create and manage volumes, or directories that can be given to containers to store data. These directories are useful to store config files, data directories, or other persistent data. This way, when containers are stopped and deleted, they can be recreated without having to reconfigure each application from scratch.

Questions

Why not Kubernetes?

As an IT professional, I have been exposed to Kubernetes (K8), and I have found it difficult to learn and implement. K8 is a very powerful platform that is much more powerful than Docker, however, since it is difficult to learn, students furthering and using my project may have a hard time learning the system. My goal was to make a system that was easy to use, and K8, as advanced as it is, is difficult to use, which is why I did not select it for this project.

Why not Python's venv?

I have not been greatly exposed to Python's virtual environment much, but as it is an uncommon containerization platform, there isn't much documentation or useful tools to deploy whatever service is needed. An academic institution needs to expose students to commonly used tools, and Python's venv isn't used as often (*Virtual Environments and Packages,* 2021).

Why not use a GUI on the Linux distribution?

Docker is mainly a command line application, and since the system wasn't designed to support everyday users, and was supposed to exclusively run services, there was no need to have the overhead of an entire GUI. Headless installations are known to easily outperform GUI-based operating systems.

Implementation

Before the project began, three main goals came to mind. The goal of this project was to create an environment that was easy to learn, easy to maintain, and easy to use. While I have experience deploying servers with complicated firewalls, strange networking configurations, and complex software, I wanted students to be able to just jump in and go.

I originally wanted to deploy a Kubernetes cluster, as it is used more often in highly professional environments, but since it is complicated to configure and install, I decided to go with Docker. I also wanted a flavor of Linux that is easy to use and utilizes minimum resources. Since most of the server management would be done through SSH, I decided to go with a headless installation of OpenSUSE since I'm most familiar with that distribution.

After deploying a virtual machine and installing the operating system on the computer science department's VMWare cluster, I changed the network settings of the VM to match our network settings (changing the default gateway and assigning a static IP-address). I then created a template and made several more virtual machines across four different nodes, configured them, and joined them together in a swarm as manager nodes.

Using the Docker stack utility, I deployed a Docker management interface service called Portainer, and a Portainer Agent across the swarm. This allows Docker to be controlled through an easily usable web portal, instead of having to SSH into the server, and deploy and manage services from the command-line.

Once Portainer was installed and configured, dozens of templates had to be created such that users can deploy services without having to mess with *docker-compose.yml* files, or manually creating a container from scratch.

Once templates were created, services could start being deployed, a MariaDB, File Server, Git Server, Password Manager, and a Mongo DB were at the top of the list. Thanks to templates, each one could be deployed in minutes instead of hours of research & planning, and another hour of implementing.

A Hadoop (Big Data Analytics) system was also requested as the project was reaching its final form and could also be easily deployed using scripts and *docker-compose.yml* files provided by the open-source-community.

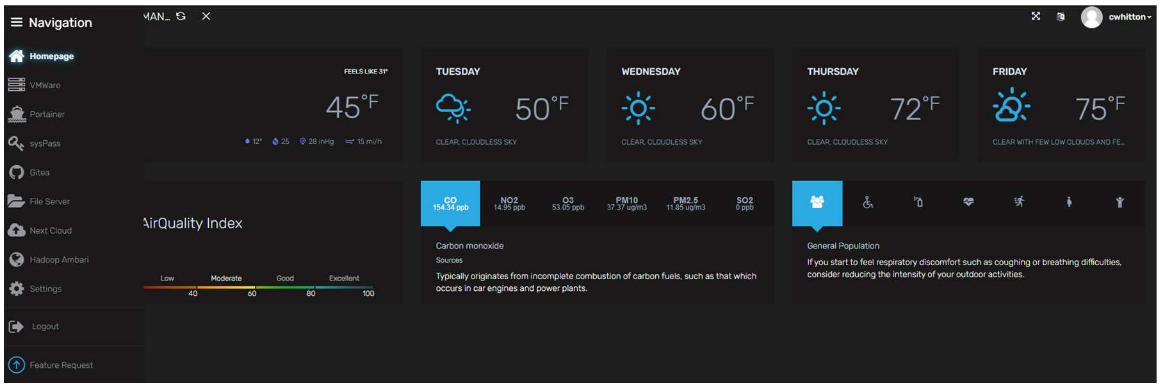
Once the project was completed in the Fall of 2020, lab administrators were briefed and given the ability to manage the docker environment themselves. While more configuration and improvements will always need to be done to each service, plenty of documentation is available for lab admins and students to learn each service and make it perform its best for the NNU community.

Results

Figures

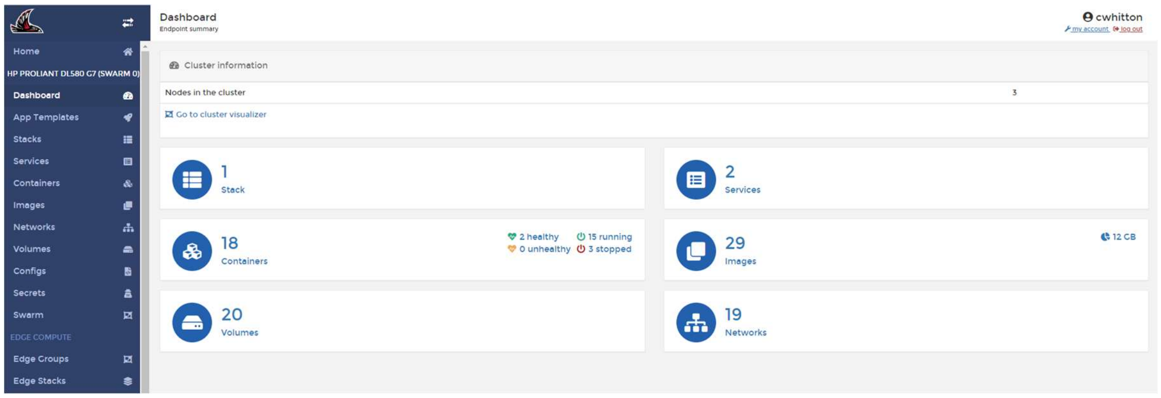
Server Management Dashboard

Allows authenticated users to quickly launch commonly accessed utilities and services.



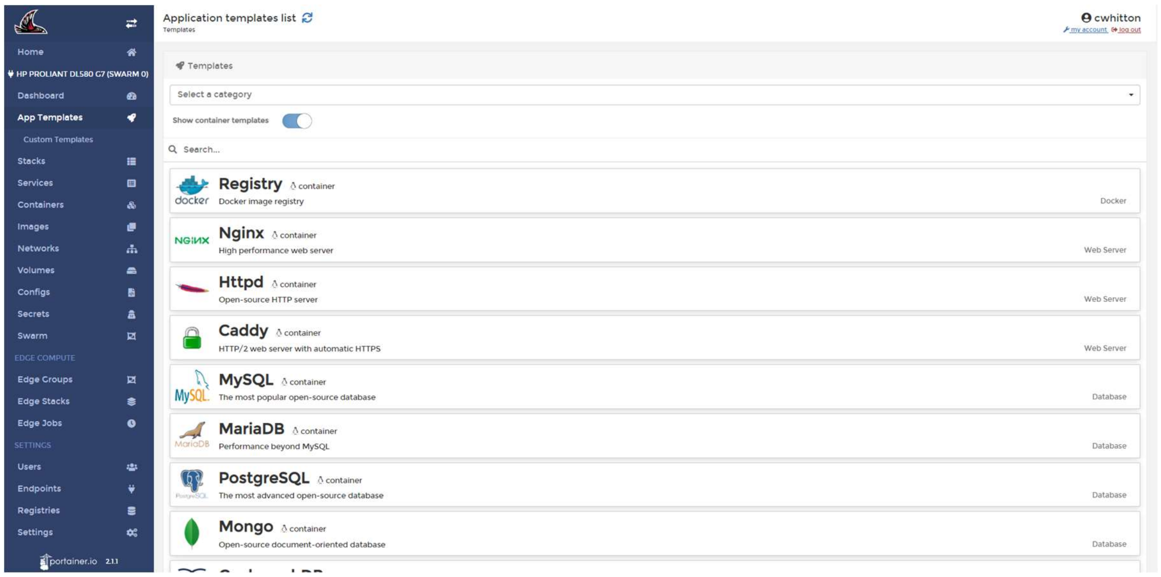
Portainer Dashboard

The dashboard that allows authenticated lab administrators to create/manage/update/remove Docker containers.



Portainer Templates

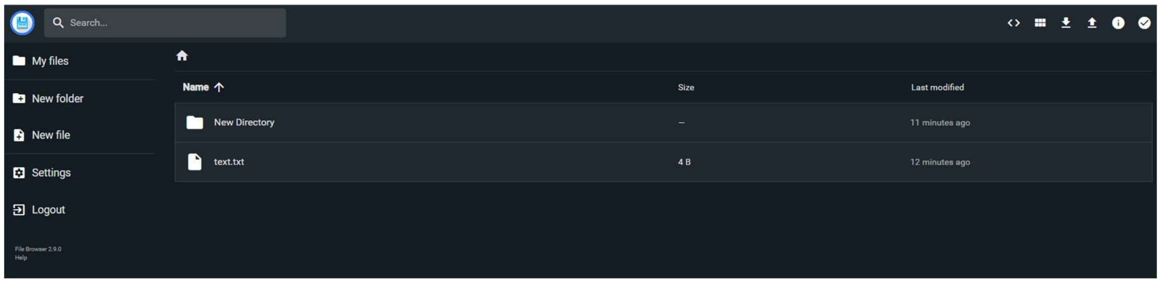
A list of containers and services that can quickly be deployed within the Docker ecosystem. Each one requires minimal configuration to get it up and running.



Registry container	Docker image registry	Docker
Nginx container	High performance web server	Web Server
Httpd container	Open-source HTTP server	Web Server
Caddy container	HTTP/2 web server with automatic HTTPS	Web Server
MySQL container	The most popular open-source database	Database
MariaDB container	Performance beyond MySQL	Database
PostgreSQL container	The most advanced open-source database	Database
Mongo container	Open-source document-oriented database	Database

File Server

A very responsive file server that allows authenticated users to upload/create/edit/delete files and directories.



Name	Size	Last modified
New Directory	–	11 minutes ago
text.txt	4 B	12 minutes ago

Conclusion

After countless hours, and frustrating moments, services, such as web servers, can easily be deployed by NNU faculty, students, or lab administrators with minimal effort. This will allow for bigger, and more ambitious senior and research projects to happen in the future.

Future Work

Ticketing System

Every major IT infrastructure needs to have the ability to handle and track many requests efficiently. Ticketing systems are a way for IT professionals to help users they support get their issues resolved quickly and effectively. Users can then see, track, and respond to progress being done on their request.

LDAP Authentication

Managing user accounts is a hassle when working with multiple services, luckily, Active Directory authentication can greatly aid lab administrators grant and revoke access of students with great ease. This will allow users to log into any of the services the Computer Science Department offer with the same username and password used to authenticate into NNU's network.

Domain Controller

A domain controller will greatly aid both the Computer Science Department, and NNU IT by assigning easy to remember domain names to servers instead of having to remember the IP address of each server.

The Ability for External Users to Authenticate

Eventually, some of the computer science services could be exposed externally, allowing students, faculty, and collaborators to access applications that would normally only be available inside NNU's network.

Mail Server Connection (cs@nnu.edu)

This email would be an administrator type email, where students requesting services could email requests, where corporate and governmental agencies could inquire about our work, and where official computer science department emails could be sent from.

References

1.1.1. brief history of virtualization. (2012). Retrieved April 28, 2021, from https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html

Docker overview. (2021, April 23). Retrieved April 28, 2021, from https://docs.docker.com/get-started/overview/

Container networking. (2021, April 28). Retrieved April 28, 2021, from https://docs.docker.com/config/containers/container-networking/

Virtual Environments and Packages. (2021). Retrieved April 28, 2021, from https://docs.python.org/3/tutorial/venv.html