Acquisition, Storage and Sharing of Data with Clients through Containers & Secure Channels

Aaron Rajesh Anthony
Msc.Information System with Computing
Dublin Business School

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acquisition, Storage and Sharing of Data with Clients through Containers & Secure Channels

***Abstract*—The purpose of this project is to highlight the versatility of Containers, Docker, Cloud Technologies, File Sharing Systems, Authentication Protocols etc. This project showcases how data flows from the user all the way to the host and its clients through a complex, reliable and secure system. The in-depth explanation of all the technologies and its integration has been described below.**

I. Introduction

Data, A group or collection of information that can be stored & manipulated in order to be utilised for further processing, gaining insights into trends developing over time, etc. Data these days are massive in volume and most of the time useful & garbage data are all mixed up and sent from one client to another through vulnerable/exposed channels. The first step in order to tackle these issues is to divide the problem into parts and solve them. The first part in this case would be data acquisition, the Second part would be data storage, and the Third part would be data sharing. The standard procedure would be to implement all three of these together on a single server application. But, when dealing with vast amounts of data we need to see to it that none of these three parts ever fail and even if they do so, we should be able to troubleshoot them easily. This is not possible on a single server application. Hence, We use something called as Containers, which can run complete applications by running each part in a separate container/sandbox and then synchronize or join them together to run smoothly. This solves the Data Acquisition & Storage part. The Data sharing is handled by various other security protocols, directory protocols etc. for example, Active Directory, LDAP(Light Weight Directory Access Protocol), Kerberos Authentication, Network File Sharing System etc. In this system i am implementing a basic web application which will take in information from the user and send it to the database for storage. The web application & Database will run on two separate containers.Then, the data stored in the database container will be saved onto the virtual machine(1)'s shared directory which then can be accessed by another virtual machine(2) through a secure network.

II. Problem Definition

The **Aim** of this project is to showcase the versatility & pragmatics of utilising Containers with the help of Docker. It also aims to showcase the file/network sharing capabilities between a host server and its clients through a secure channel. The **Scope** of the project is to deliver a system that will successfully accept the data from the user and send it to the database, forthwith, the accumulated data is saved on the host server from where the clients can access this data. We must keep in mind that the containers must be running and in sync with each other. Meanwhile, the file sharing system must be secure such that no form of break-in or loss of data occurs during the client to host session. We shall be considering that both client and host are Linux based systems and hence appropriate technologies have been implemented for the same.

III. Proposed Solution

For this system I shall be using the following technologies:

A. Virtual Machines

A virtual machine is a computer file, typically called an image, that behaves like an actual computer. In other words, creating a computer within a computer. It runs in a window, much like any other program, giving the end user the same experience on a virtual machine as they would have on the host operating system itself.(Azure,2019) The purpose of using virtual machine on a cloud platform is to simulate a host, multiple clients, servers, directories and applications by utilising minimal resources and the choice of operating system which we would like to use to run our system on. I am using Virtual machines hosted by Azure Cloud Platform. where both my Client and Host are Linux based servers.

B. Containers

Containers are an executable unit of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud.(IBM Cloud Education, 2019). Containers are used to separate long monotonous, dependable processes into its own standalone process which is later integrated with its co-processes to run the main application. They use Linux names-paces an integral part wherein every process is assigned a set of resources and these resources are then distributed to multiple processes with the same names-pace. This has multiple advantages,

- A container does not need its own OS and shares the kernel of the bare-metal or host machine. Hence, It can run processes and applications by itself and utilise only that much amount of resources needed.
- By using a Container for separate processes we can troubleshoot any issues pertaining to that process very easily without the need to shut the whole system down or encounter some other problem elsewhere. We can also develop that part of the application and upload it without any hassle.

- Containers do not run on your hardware machines and hence anything related to it is all cloud based. Making it more secure, dependable and less prone to failing.

Hence, I shall be using the container tool **Docker** which makes containerization process much simpler, structured and effective. There will be two containers with MySQL and Apache Server along with PHP running. An image of each of them will be pulled from the Docker Hub along with all the necessary codes, packages etc. I may even create images and compose the Yaml file to create distinct applications. Both these containers will run on its own and data will be inserted by the USER into the Web-Application hosted on Apache server and it shall be inserted into the database through PHP. The stored data from the MySQL container shall be saved/copied onto the shared file on the Virtual Machine which can be accessed by the Clients from remote locations with the right authorization.

C. Network File System

A network file system is a network abstraction over a file system that allows a remote client to access it over a network in a similar way to a local file system.(M.Jones, 2010). This system is a Client/Server based application where in the Client can access files on a remote server as if it was on their local machine. This is done in Linux by mounting the directory to be shared on to a server. From where, the Clients can access a part or complete file depending upon the privileges assigned to them. NFS uses Remote Procedure Calls(RPC) to route requests between Client and Server. Another great feature is that NFS in order to maintain the same language between the host and its various clients and requests, it has an interoperability layer, called as External Data Representation(XDR), when a different architecture requests a data type representation, different from that of the host architecture, the XDR converts this to a common representation which can be then shared and used by all architectures and systems.(M.Jones, 2010) Hence, I shall be using NFS to access the data on my host Virtual Machine from the Client Virtual Machine. I shall also be adding some authentication procedures discussed in the next point.

D. Kerberos Authentication

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography.(MIT, 2019) Th internet is not a secure place for data transfer to take place. Many password, packet sniffers or bugs are always on watch and hence, in order to prevent this issue, I shall be using Kerberos Authentication which uses an intermediate server called as Key Distribution Center(KDC) which assigns an encrypted key to each request from the Client, Authentication Server and the Ticket Granting Service(TGS). When the client wants to access any information or data from the server, it issues a TGT(Ticket Granting Service) to the KDC. The KDC, verifies the initial credentials and sends an encrypted TGT back to the client, who in turn stores the TGT and once it expires the session manager requests another TGT. Now, the current TGT is sent to the TGS along with the SPN(Service Principal Name) of the resource it wants to access on the server. The TGS then sends a session key which then the client uses to access the server data.(Jeff Peters, 2018). Thus in this way I intend to implement Kerberos Authentication along with NFS to keep my data sharing with clients safe and secure.

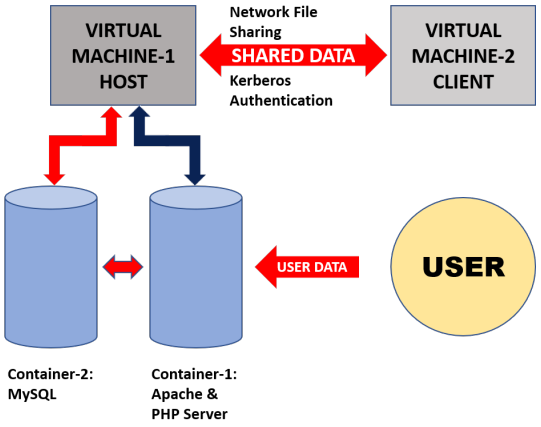

Fig. 1. Block Diagram

IV. Research Methodology

During the Initial phase of the project, I had planned to use MySQL, Apache and PHP separately on three individual containers. Upon further research, it was found that it is very difficult to implement them individually, especially when you need the containers to talk with each other. Hence, I used the alpine images which provides a back bone for these three common applications. It allowed smooth operation and communication between these containers. Also Volumes as a major addition to the project since, it allowed a copy of the containers directory to be mapped to the host directory, such that despite multiple compose ups/downs, data being stored in these containers would always have a common point of save. Preventing data loss. Next part was implementing the communication with the Host and its Clients. Network file system is one of the best forms of file sharing in Linux. It supports various types of encryption for its channels and has a good encryption of its own. Hence, Upon researching more, I decided to use Kerberos Authentication along with NFS.

V. Implementation & Analysis

A. Implementation of Virtual Machines & Containers

Azure Cloud Platform offers us various web services which can be used to deploy massive projects as well for prototyping of systems. We shall create two virtual machines (Host & Client) as shown in Fig(1), with a Disk space of 2GB and running an Ubuntu Server 18.04 LTS. *Refer Appendix A: Fig.1* We shall name them as VM-1, VM-2, Host and Client respectively.Before setting up the containers the first task is to create directories called as Apache, php, public-html and a

docker-compose.yml file in order to keep the files organised. *Refer Appendix A: Fig.2* Now, within Apache, we will create a **Dockerfile** which will contain the image of Apache Web-Server (httpd) along with Alpine Server which takes up just 130MB of disk space. As you can see from *Appendix A: Fig.3* the Command **FROM** is used to pull images from Docker-Hub to initialise the container. The **COPY** Command is used to copy the demo.apache.conf file into the conf folder since this will be the proxy to decouple the two containers from PHP. Thus, smoothly displaying the web page. In, php folder we have a Dockerfile which pulls the php 7.2.7 server image along with Alpine Server 3.7 and also initializes the mysql extension for PHP.*Appendix A: Fig 4* The next step is to make the PHP script within public-html in order for our web-application to be created. We create two PHP scripts, index.php and insert.php, where index.php has the html code along with the routing to insert.php, which has the database configuration and insertion into database script. *Appendix A: Fig.5*. Finally, we implement the Docker-compose.yml file, *Appendix A: Fig.7*, which is basically a collection of all the images, packages together for the container to run smoothly. We shall compose up the whole system and the web page appears on port 8080 of the Virtual Machine, which can be viewed on our browser at 'VM-IP:8080'. Now there are three containers running on one PID(1) simultaneosuly that is, MySQL, PHP and Apache containers. *Refer Appendix A: Fig. 8*. We design a basic database with a simple table 'info' in the database 'info_user'*Refer Appendix A: Fig.9*. The web-page has just two inputs 'name ' and 'email' with an auto-incremented ID. As you can see from the figure, data has been taken from the Apache container and using PHP extensions and scripts it pushes it into the net container of MySQL. The data stored by the user is now all in the second container that is, MySQL. We shall use one of the most important features of Docker that is **Volumes**. A Docker volume is a directory (or files) that exists on the host file system (outside the Union File System).(Jack Wallen, 2017). We create a folder called as container data and we build the Yaml file along with this volume directory. Thus, Whatever data that is stored on the container directory is copied onto the host directory irrespective of the number of times the docker is composed up/down.

B. Implementation of Network File System & Kerberos Authentication

The first step is to install NFS server package on both the Host & Client which allows it to share its directories. *Refer Appendix A: Fig.10* We shall be making the container-data available to the clients. We change the /etc/exports file and add the File to be shared with the client IP of the client server, along with some important commands like, rw - giving both host and client access to read and write to the file, sync - makes nfs to write changes to the disk before replying, etc. (Melissa Anderson, 2016) We then adjust the firewall settings such that the NFS traffic is allowed. We then create mount point directories on the client server as well. The directories can be mounted onto the client by using the mount function which makes available the directories from the host on the client as if it was available on its local directory. Now this file sharing needs to be encrypted with something very strong and reliable, hence, we shall be using **Kerberos Authentication**, Where, we shall install yum repos initially and then create a KDC(Key Distribution Center) server. After that we create the Kerberos Database, where all the confidential credentials are kept stored. We shall then start Kadmin and krb5kdc simultaneously such that they restart with the new settings.Next we adjust the firewall settings and then launch kadmin. we shall setup the client by downloading the authconfig-atk package. and then create a user and fill up its credentials. Were then ready to implement the system and KDC is ready to give out TGT's.

VI. Conclusions and Recommendations

In Conclusion, When we have a massive system application to be run, it is essential to utilise and implement applications/Systems using minimal resources that are efficient and pragmatic. From this project, I have ran a whole Data Acquisition System , Stored it and Shared it with clients using minimal resources and efficient technologies. It can also be inferred that its more organised and resourceful to implement the applications in part. The containers running MySQL and Apache & PHP used less than 300 MB totally to implement a massive application which can take multiple user input values. Meanwhile, the data in the containers are easily handled and stored permanently on the directories of the host machine, which is then mounted(placed) onto the NFS server and clients can access these files through a secure channel with the help of Kerberos Authentication. In case of any glitch, bug, problems we can decouple the containers and troubleshoot them. In case of software updates or scaling up of the system, we can use back-up containers and quickly deploy the new software and keep the running of the system very smooth. There are various methods to implement MySQL,Apache &PHP together like LAMP-STACK, ALPINE etc. PHP scripts can be replaced with Java Scripts and the Web-application can be made more robust and secure. Since, PHP is not so secure.

References

[1] Azure, *Virtual Machines*.https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/. Accessed on: Nov 2, 2019

[2] IBM Cloud Education, *Containers* https://www.ibm.com/cloud/learn/containers. Accessed on: Nov 16, 2019.

[3] M.Jones, Network File Systems and Linux *IBM Developer*, https://developer.ibm.com/tutorials/l-network-filesystems/. Accessed on: Nov 16, 2019.

[4] J.MIT, Kerberos Authentication, https://web.mit.edu/kerberos/. Accessed on Nov 6, 2019.

[5] ,Jeff Peters, Data Security: Kerberos Authentication, https://www.varonis.com/blog/kerberos-authentication-explained/. Accessed on Nov 2, 2019.

[6] ,Jack Wallen, Tech-Republic, *How to share data between Docker host and Container*, https://www.techrepublic.com/article/how-to-share-data-between-a-docker-container-and-host/. Accessed on Nov 11, 2019.

[7] ,Melissa Anderson,Digital Ocean Community , *How to mount NFS on ubuntu*, https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-16-04. Accessed on Nov 11, 2019.

APPENDIX A
SCREENSHOTS, CODE SNIPPETS BLOCK DIAGRAMS

Fig.1:

Fig.2:

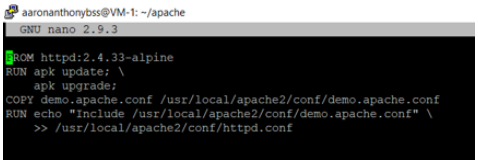

Fig.3:

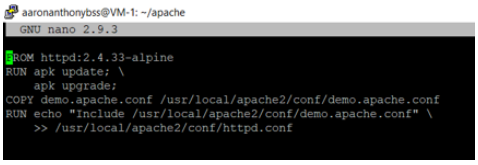

Fig.4:

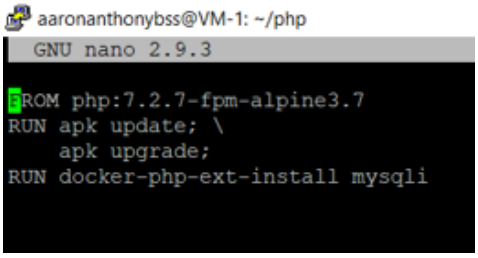

Fig.5:

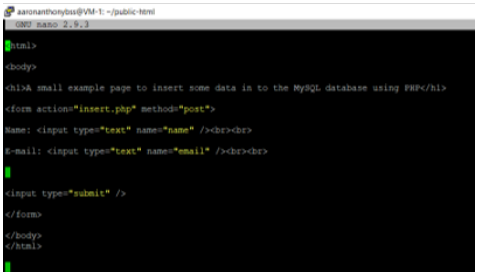

Fig.6:

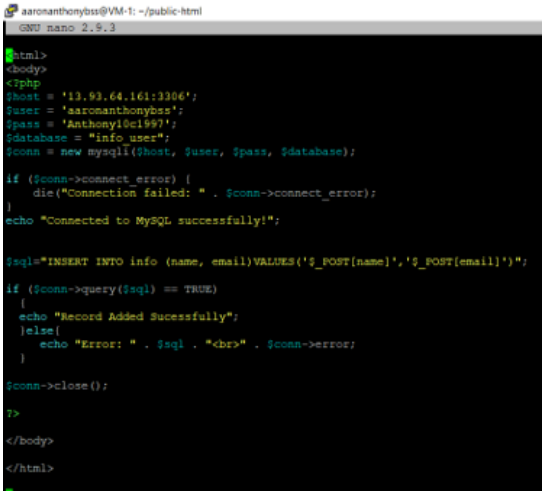

Fig.7:

Fig.8:

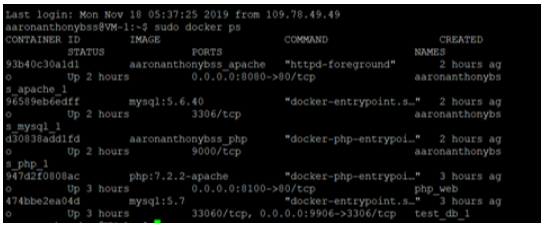

Fig.9:

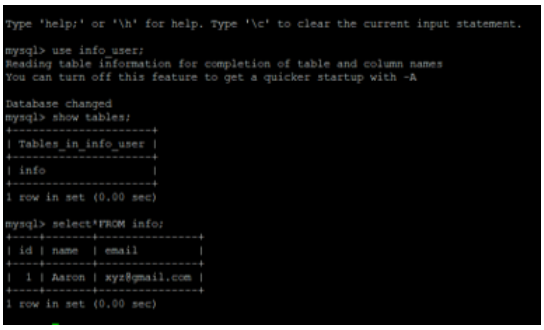

Fig.10: