Cloud enabled media database: Installation and Deployment

By Payton Young, Joseph Cort, Christopher Ayling, and Ethan Bridgehouse

Table of Contents

1 Introduction

1.1 Introduction

Cloud enabled media database - a service hosted online for any person to be able to gather new information and share their opinions or thoughts freely on vast types of media. This is the service that we set out to engineer and construct for our project. Creating an entire system that can store all types of media sorted through different categories and genres all on one persistent database is, from what we learned through this process, not a simple and straightforward task. We, like most inexperienced persons when taking on a project such as this, began our collaborative effort to conceptualize what our end product will be and how it will operate. Essentially, we were only able to visualize the front end of the service and what our end users would be using our site for and how they would be able to interact with it and navigate through it. Although the front end operations are crucial for our project and for any service that is hosted on the cloud. What we could not expect, due to sheer lack of experience and knowledge in this field, is all of the different problems and technical difficulties that could arise and eventually would.

1.2 Overview

Shifting out of the planning phase of the process we moved forward with the idea that we will build a cloud based service that will host different types of media for limited users to interact with and register to create accounts with. All of these services and those affiliated with our planned product will be containerized through the use of Google's Kubernetes and be configured such that each service will be deployed through its own container and will be hosted and run on its own machine. Kubernetes gives us the tools to schedule and run our containerized applications on either virtual or physical machines, meaning that we will be able to orchestrate all of the containers to work together with each other to deliver our final product of a cloud enabled media database.

1.3 Scope

Our group and experiment was able to achieve almost everything that we set out to accomplish in the end. We were mostly able to implement our three service components into their own containers that could be launched and managed by Google's Kubernetes. The size of the project could only be scaled so large for reasons that we will touch on later. Since the main goal of this project was for us to learn and work through all the different types of failure that comes along with cloud computing and building your own service, the scope was always being forced to change and move around so that we could stay on track and keep up with the assigned deliverables.

1.4 Target Audience

Once again, due to our initial lack of experience in the field we had assumed that the scope of our project and our end service could not spread to that many different users. We knew there would be issues that we could eventually work through and bugs that would need to be ironed out. There are other errors that we will touch on later in this document that seemed to

be out of our control. Since this project was for educational purposes and not much else, the cloud system that we were using to host our containerized applications was that of Cloud Lab. Since Cloud Lab's experiments only run for 18 hours at a time and can only provide us with so many resources for our respective nodes and services, we could not expand the scope of our audience very far past our smaller circle. Since this project is intended for educational growth and experience, the audience will most likely not deviate far from professors and future employers.

1.5 Objectives

As stated previously, our mission was to build a cloud enabled media database for our own educational growth. Therefore our main objective was to learn and sharpen our skills as computer science majors here at West Chester University. We needed to do extensive research and spend countless hours staring at failing code and broken pipes and learn to cope with all the growing pains that come with having to learn how to essentially become a cloud computing engineer. Throughout the project our main objectives were as follows: allow users to register and log in with accounts, allow users to add new information about a selected media, allow users to add new media to be reviewed, allow users to interact with other users' feedback and storing all of this information in an easy to reference database. Due to time and resource constraints we had to be flexible with ourselves and our own plan and veer off course a little to keep the scope of our project in line. Overall, since this was a learning experience for everyone involved in the project, we feel satisfied with our final project and enjoyed the process of learning how to work through all of these problems on our own.

2 Summary

2.1 Technical Plan

When we were first choosing which project we wanted to work on as a group we ruled out the benchmarking test fairly early on because we wanted to try and build something new and push ourselves to learn new skills and marketable abilities. With that in mind we decided to build a container based, fully deployable service that would host a webserver with an apache2 based website in its own container. From there we would also have another container for an LDAP (Lightweight Directory Access Protocol) authentication service. Finally we would have a postgres database that can be accessible through the apache2 based website. We planned on launching all of the different containers on their own node through the services provided to us by

CloudLab and we would have one head node that controlled all the rest. Please refer to figure 2.1 below for further detail.

2.2 The Process

To kick off the project we had to get an operational web server so that we could begin building all of the backend services that we would need. We knew that we needed to get a dockerfile built which would create an apache server that could support LDAP. After getting everything working with kubernetes for deployment we had landed on using an apache server that with the help of php, we could work toward implementing mySQL and LDAP alike. Since so many services are open source and have many different versions and methods of integration for different use cases, it took us a while to find out exactly what type of each service we would

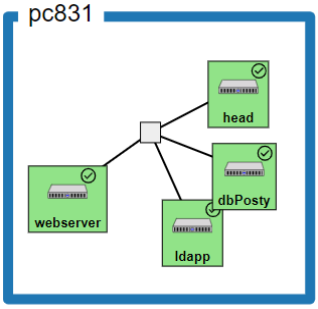

Figure 2.1: Network Diagram

need and how many of the extra bells and whistles we could shave off to get our project running smoothly and have all the nodes interact and communicate with each other nicely in final deployment. We knew early on that persistent data storage would be a problem with CloudLab since our experiments can only run for about 18 hours. Since it would shut down after the allotted time we knew we would have issues down the road with keeping our database populated. At this point in the project though, it was not at the top of our list of problems that we needed to be addressing. One of our big breakthroughs was finalizing our containerized website with a logon form that we could then implement our LDAP service through which allowed us to move on to both the implementation of LDAP and the beginning of learning and working through the basic documentation of mySQL.

After having multiple internal conversations on our group's discord channel and talking with Dr. Ngo, we decided to drop mySQL and move forward with postgreSQL instead due to its long history of support and updates and because of recommendations from Dr. Ngo himself because postgres is much more established and is a professional service to use. Coming up to the end of the road for the semester we were able to solidify and finalize our apache webserver with the LDAP plugins and the php login form that would communicate with LDAP for the final deployment. All that was left was to work with postgreSQL, finally deal with our problem of the persistent database and configure everything through kubernetes for when we deploy our project onto CloudLab.

2.2 Technical Difficulties

When we started digging into the different processes that we would need to construct we realized that this project was a bit more complicated than we initially thought it to be when we were in our planning phase. We had to find out how to containerize everything that we needed and that task turned out to not be an easy one. Many of the different services were challenging to get working on their own and then containerizing them would normally become a problem in and of itself. After spending a decent amount of time working with LDAP we learned that it would be much easier to implement this service onto our web server if we used openLDAP just because of its ease of integration. There were similar issues like this throughout the entire process where we just essentially wasted our time on one type of service only to learn of a different easier service that we would then pivot to and have to backtrack all of our previous work to get this to integrate on its own. By the time of our last tech demo for deliverable 2 we were able to launch with an operational LDAP authentication server and a basic database that we were not able to perfect as much as we would like. At the time of deployment our LDAP server was operational but we were not able to actually reach our LDAP admin page. We currently are able to display our web page on the server but we are not able to say for sure that the LDAP service is communicating properly with the rest of our nodes. When you navigate to the basic forgot password page, it all functions normally. When you attempt authentication the page will try to connect and eventually the LDAP bind will fail. At the moment we are not able to access the LDAP admin page either. When we run the service and containers locally on a single machine we are able to access it but not as a cluster due to not being able to use both the clusterIP and the nodePort service in the yaml file.

3 Services

3.1 Docker

Docker is an open source project that was developed by Dotlab. Docker is part of a set of Platform as a Service products that use OS level virtualization to develop, distribute and run software in what's called a container. Essentially docker allows developers to package up their application and all of the required resources that it will need such as libraries and all other dependencies and ship it or deploy it all as one package. Docker is the main overarching tool that allowed us to move on with this experiment and get it into a functional state. Since our project needed to be hosted on the cloud by the use of CloudLab we needed an easy way for each node to have everything it needed all within one package that could be easily implemented into our groups github repository, which you can find the link to at the bottom of our title page.

3.2 Web server: Apache and PHP

In the simplest terms, Apache is the web server that processes requests and then will serve out the different assets and functionalities by means of HTTP. PHP is essentially the coding language that Apache will use. So when the server is launched it will parse through our github repository and look for any php files that it can run. Which in our case it would only find one php file. The Apache server will also display our webpage which is written in the HTML language so that users will have an actual interface to interact with.

3.3 LDAP Single Sign On Service

Lightweight Directory Access Protocol is a software protocol which has become the industry standard for accessing and maintaining access to a distributed directory information over an IP network. LDAP can be used as a central place to store all usernames and passwords, which is what we planned on doing so that those interacting with our product could make their own account so that when they were to leave reviews and do anything else on the service it could all stay tied to their account. We specifically wanted to use LDAP for a single sign on service. A single sign-on service allows users to log in only once and then be able to have access to all services being offered without ever having to re-authenticate their account information. LDAP is what allows our group to store the user information so that we can achieve a single sign-on service through our web server.

3.4 PostgreSQL Database

PostgreSQL is a database management system that we used to manage our service's database. Postgres has taken over the industry as the new standard from mySQL. We at first were attempting to integrate mySQL into our service as the database management tool but through our own research and the help of Dr. Ngo we learned that we should move onto the new standard and use Postgres. The difference between the two is that mySQL is strictly a regional database while Postgres is an object-relational database which means that it can include more functional features and it also adheres much more closely to the SQL standards than mySQL. With over 20 years of community driven development, postgreSQL is now used as the primary data store or data warehouse for all different types of web, mobile, geospatial, and analytics applications.

3.5 Kubernetes

Tying our whole project together is Kubernetes. Kubernetes is an open-source container management system that was originally developed by Google but then had the reins passed onto the Cloud Native Computing Foundation. Kubernetes allows us to take our individual nodes and turn them into their own elastic web server to host our application. It will essentially be the conductor for all of our different containers since we are deploying each container on its own node which is being hosted through CloudLab. Kubernetes essentially turns all of our nodes into a cluster where all of the worker nodes become the workhorses for the application we developed and the head node becomes the maestro.

4 Implementation

4.1 Summary of Changelog

Like previously stated throughout this document, the project was a slow burn at the start with a lot of back and forth on what we need to really get this off the ground and build the momentum for the rest of the project. We started simply with the specifications of our project on

paper and a network diagram that eventually evolved to what you can see in figure 2.1 above. Moving forward we altered the profile.py of our project to correctly account for the amount of nodes that we would need so that we can match the network diagram that we designed. We then moved forward to try and create our web server because that is step one to getting everything off of the ground. We initially wanted to use NGINX because we thought that it would be ideal to get everything running off of it. We quickly realized that in our current state NGINX does everything we need, but due to its “freemium” nature we might have to change it in the future to something else. We moved ahead with NGINX for the time being and containerized our server so that we could display some sort of HTML file and attempt to expose it to the outside. We eventually learned that NGINX would not be able to provide all of the functionality we needed from it without paying a premium for it. Therefore, we changed course and we got a container running from the apache:php-7 image. After we got the new server ready we pushed the updated files from Payton’s WSL branch which contained updated HTML and CSS files for the apache web server. From there we moved on to the next services that we needed to implement into the web server. We first changed the specification of the database to simple text because we were predicting the database to become an issue for us with time winding down. After that we made a strong push to get the LDAP server off the ground and operate with the web server. Multiple pre-made LDAP server images were looked over before unique Docker Images were published to both Payton and Joe’s personal DockerHub accounts. This was essential for the implementation of our very simple LDAP server we set out to create. After many attempts to get the LDAP server off the ground with php we met with Dr. Ngo to learn that we should probably switch to a different CentOS image to assure that we can get everything functioning within the given time constraints. While running out of time in the semester we finally wrote and pushed the redkube Kubernetes file along with a CentOS image built for apache. Although there are still some errors being thrown, we implemented our Kubernetes cluster onto CloudLab.

4.2 Initial Setup

For the project to get anywhere off the ground we first had to set up and write a profile for CloudLab to run scripts and correctly install both Docker and Kubernetes onto each of the nodes within our cluster. Furthermore we had to set up additional scripts on our GitHub for CloudLab to be able to go through and differentiate between the different nodes and again, make sure that Docker and Kubernetes is correctly installed on each node or we would not be able to form the cluster for our nodes in our later scripts. These steps are necessary so that we can easily distinguish the head node from the worker nodes, this is essential for the facilitation of our cluster. All of these steps will run on their own once the profile is instantiated on CloudLab and a cluster is chosen to finalize the experiment before launching it. Even after CloudLab says the experiment is ready these scripts will usually take a few more minutes to load and run before we can move onto the next steps.

4.3 User Required Setup

Once the experiment is finally fully launched and users are able to successfully secure shell into the experiment with the given key, they are able to begin launching the experiment. We first must launch the network by executing the launch_network.sh file within our GitHub.

This script will provide multiple configuration options that help all of the nodes communicate with each other. Finally after all of those commands are deployed and functioning we finally run the command “kubectl apply” which sets up the kubernetes cluster and deploys all of the pods and services. The service and our final product is now live for the users to interact with.

4.4 Final Product

With the service fully launched and deployed users are now able to access the web server and the attached web page that comes with it. In its current state the LDAP bind is not fully configured or implemented due to difficulties that we ran into with the NODEPORT assignment. This was a problem that we were not able to solve due to time constraints and other external factors. For example, we were not able to implement the cloud-provider based load balancing methods since they are not yet supported by CloudLab. We know that our LDAP server is functioning in some manner because the web server should not be able to maintain its stability without the connection to it. Additionally, we attempted to have a postgreSQL service running within its own container that would manage the database of media for our server. Unfortunately, we also ran into issues with it running.These issues were similar to the issues we had observed with LDAP. We could tell that postgreSQL was running, but it could not properly display the application.

5 Conclusion

5.1 Final Thoughts

As was previously stated throughout the entire report, this was a learning experience for everyone involved in the project. We knew that we were objectively choosing the more challenging and time consuming project by taking on an entire service that we would have to educate ourselves on and then proceed to showcase that knowledge with quite a quick turn around. That being said, although we did not accomplish everything that we set out to do with this service and project, we all believe that we have grown as students and computer scientists by taking part in this. We learned the ups and downs of working with group members remotely on a scheduled time frame, we learned how the back end of all cloud based services operate and the hours of coding and teamwork that go into setting them up and keeping them in a stable build while trying to add on to the backend. This project showed us that not everything is going to go your way easily and there will not be a specific tutorial or walk through when you are building something unique. If you are building and creating something that has a walkthrough and a tutorial to do it then that means it has already been done and there was really no point in spending your time and money on doing it.

Some recommendations that we can make from our experience with this project is to spend more time in the planning and research phase and really plan out all that you will need to do on the backend as well as the front end. We were eager to get started with the technical work which made us have to backtrack on multiple occasions due to lack of research to assure that we were using the best service or tool for the job. One obvious example for this is creating an NGINX web server and then later realizing that we should be using apache. Another lesson

we learned from this project is that things will FAIL. Pipes will break, code will not compile, services will not communicate, nothing will ever play nice with each other on the first go around. That is true in the field of computer science and just in the world in general. This project showed us that failing and having technical difficulties is both normal and should be encouraged at this point in our professional careers. The best way to further your education and your technical skills is to learn from your own mistakes.

We are disappointed that we could not deliver everything that we initially had planned. The project began to pile up quickly and required more and more time and resources from us. As the semester began to wind down we all as a group found ourselves having less and less time to really put into the project as we wanted to. We learned tons of useful skills, techniques, languages, how to use different services like GitHub and DockerHub, and the list goes on. We as a group had almost no experience in any of the tasks we were required to do for this project to get off the ground. Only one group member even had prior experience in HTML. With all of that being said, we are disappointed that the project does not fully function, but we are insanely proud of all that we accomplished with this project while also having to balance all other school work, jobs, and home lives, and a global pandemic at the same time.