Teammates: Tyler Heller, Cameron Brown, Daniel Jankowski, David Kilgannon

Github: https://github.com/DavidKilgannon/cloud-computing-project-3

Detailed Descriptions about the Project Progress

When we began this project, we decided that our delivered project would be able to be run in a docker container (as per project requirements) and that it would provide the following services: an Apache server, a mySQL database, and Postfix. We set out to ready these services without particular focus on their theoretical applications: we did not think of it as though we were designing software for a company, but, in retrospect, all of the provided services can be argued to be of good use for a company. Postfix allows for communication between employees, the mySQL database, while presently basic, can be modified for any sort of database (we are using it for a mail server), and the Apache web server allows team members to gain access to these services through a web browser if necessary. However, this access is also our major flaw: Docker is a relatively insecure container system, and it is made even more insecure due to the difficulties it has with running UFW (Uncomplicated Firewall), but, based on feedback from the professor, we pressed onwards without UFW. In an academic setting, where our system is only meant to be run for testing and experimentation, this is ok; in any other setting it would be a grievous error. We saw fit, during the course of the project, to choose another service for the mail system, due to substantial difficulties encountered when creating Postfix in our current environment, and have instead opted for the use of Mailu.

Apache is a popular Web-Server software. For our project, we created a docker container that can install apache. The docker container contains the basic commands to install apache, using an ubuntu 18.04 OS, and it exposes port 80 of the docker container, ad it also copies html and other files that we would use into the container.. To run the docker container, you must specify what port of cloudlab, for example “sudo docker run -p 8080:80 -d apache”.

mySQL is the world’s most popular open source database for web-based applications. For our project we are creating a mySQL database from a custom image file generated by Dockerfile and setup.sql. For our sql script file, we have basic php code that creates a 2 by 2 table. For our Dockerfile, its job is to download the latest mysql image from dockerhub, set environment variables for the database and link the database script (setup.php) to the docker-entrypoint-initdb.d variable. With these two files we can run the command docker build -t team/mysql01 /local/repository/mysql to create our image with the name team/mysql01. After our image is created we run the command docker run --name db2 -p3306:3306 -d team/mysql01 to turn our image into a container with the name db2 and map its default port to 3306. With our image created the last thing we need to run is docker exec -it db2 /bin/bash so that we directed run commands to the database via shell. This process can take some time to create but when it is finished our container and database is complete and ready to accept commands. To access the database we first log into it with the username and passwords defined by Dockerfile (mysql -uteam -p123) then run the following commands to populate the table. show database; and select * users;

In deliverable 1, we proposed Squirrelmail as a solution to our final component: a webmail server. We chose it because it was simple, light, and could be installed with a few commands. However, we ran into numerous errors with Squirrelmail. The first of these were the large amount of dependencies. Even after automation of MySQL and Apache were successful, Postfix by itself was very difficult to automate due to the nature of choosing configuration options. Squirrelmail also required a prerequisite dovecot installation; dovecot is a seperate database from MySQL, and so we were under-utilizing the database we had already automated. Furthermore, Squirrelmail itself had configuration options to be chosen in its installation, and so didn't feel reasonable as an automation. It did, however, work nicely with Apache, and could therefor display easily within a browser; but this was all due to manual commands during testing.

The decision to officially stop wasting away further hours of development on Squirrelmail came with the realization that apt-get no longer supported Squirrelmail after Ubuntu 17.04. We had run our setup.sh with docker on Ubuntu 18.04. All things considered, it was impractical to downgrade for such an application. Therefore, we were advised by users from StackOverflow and AskUbuntu whom were experienced with webmail servers to move onto Roundcube-based webmail clients. (These posts will be included in our Resources.) Roundcube was ideal, because it actually utilized a MySQL dependency for its database. However, it still required a separately installed Postfix webmail server which came with configurations to automate. Even more severe of a problem was how the administrator had to manually add users from my the mysql> command line as database inputs rather than from the installed web interface. This would be a nightmare to automate or to work with in any practical sense. Thankfully, we arrived at Mailu, a docker-based mail server that implemented Roundcube as a webmail service.

Mailu is a mail server based off of Docker images. Some of Mailu's features include built-in security, antispam, webmails, administration, and the standard services that come with being an email server. An interesting feature found in utilizing and building Mailu was the automated process for creating the docker-compose and environment files, which was one of the most user-friendly things we have found when working with Docker and the various services. While we are able to access most of the utilities provided by Mailu, and open a browser inside Cloudlab to do so, we are conscious of the difficulties we have encountered with regard to automating the process from Mailu, stemming from the automatic generation of the docker-compose file. An IP address is needed to be given for the configuration of Mailu's Roundcube webmail, which is difficult due to the changing nature of CloudLab instantiation IP addresses. It was suggested to us by our professor that we parse the IP address and nano (edit) it into the docker-compose.yml file. Another solution might be to write by default a 0:0:0:0 (looks for all interfaces) IP address into the docker-compose file, but this can occasionally lead to issues, as told to us by Mailu's documentation. Please see the Mailu.io website included in our Resources file for more details.

Docker was installed easily enough, only really requiring two command lines and then being easy to use. Even automation was relatively simple to set up, which led to us moving onto greater things, with us presently tinkering with getting a docker-compose file set up as well as properly implementing Docker Swarm with an eye towards scaling outwards as needed, though, in our testing environment, we shall not have any need for the power that can be provided. The only real challenges that were encountered with Docker were due to its security and our unfamiliarity with the service. Regarding security, after some research it was discovered that Docker has some noticeable issues using UFW, as mentioned above, which was resolved by ignoring the security concerns.

Self-assessment about whether the project has met the second deliverable

Our second deliverable is the "in-class demonstration of the selected computing services on local machines or on CloudLab (may or may not be automated on CloudLab," with the example selected computing services being "...a computing service consisting of at least two service nodes…" I feel that we have satisfied those conditions, as the services are able to conducted on Cloudlab (with some difficulty). Our services are presently functional, and our main concern at the moment is the proper automation of them, which is the source of our only troubles so far. However, for this stage, the automation is optional, which, considering that we will have satisfied the rest of the conditions by the time of our presentation, is not of immediate, pressing concern.

Self-assessment and projected milestone about whether the project can make the final deliverable

I believe that we are well on schedule to make the final deliverable: the majority of our features are already automated or in the process of being automated. Our system as a whole is simple enough, and I feel that maintaining morale amongst the team for finishing the work will be the largest hurdle, as we are all encountering end of se. At the time of this writing, we are putting the finishing touches on a docker-compose file, which should allow us to execute AND scale the system as needed.

The project will also have to get Apache web server and Mailu communicating so that the project can be demonstrated on a real browser rather than inside of a terminal. The group is also acutely aware that the Mailu's setup is currently under a folder and build titled "postfix" but that this is currently an outdated title. Please visit our group's github for more information.