Project: Installation and Deployment

Teammates: Tyler Heller, Cameron Brown, Daniel Jankowski, David Kilgannon
Github: https://github.com/DavidKilgannon/cloud-computing-project-3

Bold and Italicized Lines are Section Headers
Bold Lines are Subsection Headers

Detailed Descriptions about Project Progress

Project Intention:

When the project began, the intent was that the 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. These services were originally chosen due to the familiarity of the team's members with those services, as it was originally designed solely to be for academic use, with no thought put towards possible business applications, though they can be easily repurposed for business use. Postfix allows for communication between employees, the mySQL database, while presently basic, can be modified for any sort of database (and is presently utilized by the mail system), and the Apache web server allows team members to gain access to these services through a web browser if necessary.

Problems Encountered:

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, after consulting with the professor, UFW was struck from the system. This is satisfactory in an academic system, with limited security threat, however it is insufficient for a business environment. Furthermore, Postfix is no longer the mail server provided, due to major problems with insecure subservices.

New Design:

The present system consists of an Apache server, a mySQL database, and Mailu. Mailu is a mail system designed for use with Docker, and will be expanded upon later.

Systems, in depth

Docker is a system facilitating virtualization of containers: it was chosen for use in this project due to team experience, ease of use, and abundant documentation. The system was installed with relative ease, and automation has been relatively simple to set up, though execution of said automation remains a difficulty. A number of docker-compose files presently exist, and the major remaining challenge is collision between the executing commands and the setup of environments on subsystems. It should be noted, however, that Docker does have some security flaws, which, if the project was longer in duration, would be contended with.

Apache:
Apache is a popular, free, open-source Web-Server software, and was selected for use in this project based on ease of use and good documentation. In the project, Apache is automatically installed inside of a Docker container, which contains the basic commands to install apache, using an ubuntu 18.04 OS, and exposes port 80 of the docker container. The Docker container also copies html and other files of use into the container. To run the Docker container manually, specify the proper port, presumably in Cloudlab. *Ex: "sudo docker run -p 8080:80 -d apache".*
mySQL: (commands are followed by newline for reader's clarity)
mySQL is a popular open source database for web-based applications. In the project, a mySQL database is created from a custom image file generated by Dockerfile and setup.sql. Within the sql script file, there is some basic php code that creates a 2 by 2 table. The Dockerfile downloads the latest mySQL image from dockerhub, sets environment variables for the database and links the database script (setup.php) to the docker-entrypoint-initdb.d variable. With these two files it is possible to run the command *docker build -t team/mysql01 /local/repository/mysql* to create the image with the name team/mysql01.
After the image is created, run the command *docker run --name db2 -p3306:3306 -d team/mysql01* to turn the image into a container with the name db2 and map its default port to 3306.
With the image created the last thing to run is *docker exec -it db2 /bin/bash* to direct run commands to the database via shell. This process can take some time to create but when it is finished the container and database is complete and ready to accept commands. To access the database, 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.
Mailu
Mailu is a mail server based off of Docker images, with features including 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 is incredibly user-friendly and should be utilized 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.

A Statement from the team regarding the removal of Squirrelmail in later iterations

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.

Assessments from Second Deliverable, maintained for comparison
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

The team is 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.

Assessments for Third Deliverable

All components for the webmail server have been automatically installed; if the user checks the version for Apache, MySQL, and PHP, they will see all have been installed at instantiation.

An issue may arise from the docker compose with Mailu, and may have to be executed manually. Attempting to set a variable from and IP address within a shell script has proven challenging. This leads to issues with writing over the docker-compose.yml file to the cloud computer’s IP address when necessary. If the user can follow our github page, they should be able to access the commands necessary to start up the mail server in install_email_components.sh.

Though complete automation has not been attained, this is not from lack of effort: consultation of the included repository, especially a timeline of the commits, should show hard work and diligence.