CSE 816 : Software Production Engineering

Final Project

Automated creation and deployment of a Mail System

AMEDEKA Tsevi Christian: MS2024502

Instructor: Prof B. Thangaraju

May 2025

List of Figures

1 Introduction

Email services are essential in personal, academic, and enterprise communication. Hosting a self-managed email server provides full control over data, privacy, and server configurations. The goal of this project is to develop and deploy a secure, scalable, and automated mail server infrastructure using open-source technologies and DevOps practices. The solution includes Postfix as the Mail Transfer Agent (MTA), Dovecot as the Mail Delivery Agent (MDA), Roundcube as the Mail User Agent (MUA), MySQL for database support, and a PHP-based web administration tool. The deployment and configuration process is fully automated using Docker, Docker Compose, Ansible, and Jenkins.

2 System Architecture.

The architecture consists of several modular components :

- Postfix as MTA (Mail Transfer Agent). So Postfix is configured to send and receive email messages using the SMTP protocol. It handles mail routing and relaying, including authentication and TLS encryption for secure delivery.
- Dovecot as MDA (Mail Delivery Agent). Dovecot delivers mail to users' mailboxes and allows them to retrieve mail using IMAP and POP3. It also provides authentication services for Postfix, including integration with MySQL for virtual users.
- MySQL database : The database stores virtual user information, including email addresses, hashed passwords, and domain associations. It also provides authentication support for both Postfix and Dovecot. We added phpmyadmin for GUI access in case needed.
- A web administration tool: A lightweight web interface built with PHP allows administrators to create, modify, and delete user accounts. It communicates directly with the MySQL database.

All components are containerized using Docker and orchestrated using Docker Compose. Each service runs in a separate container with its own configuration, network, and volume bindings.

3 System Diagram

Next, we have a picture of the system diagram in a starred topology network

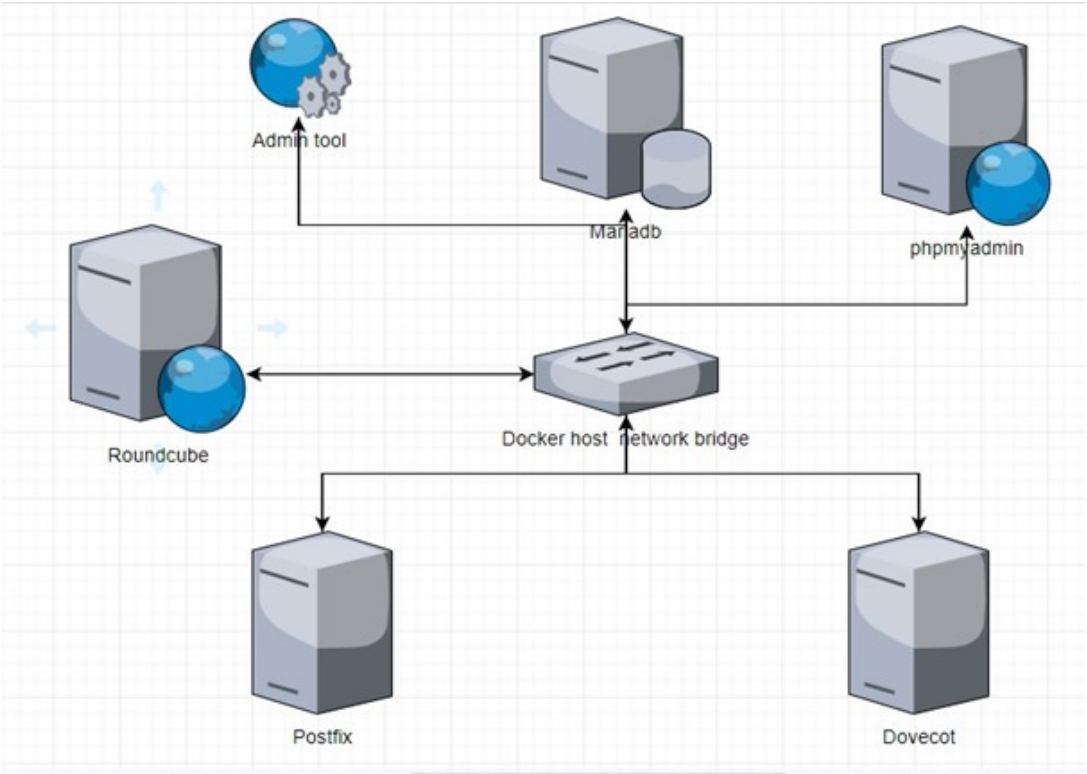


Figure 1: System Architecture Diagram

4 Devops tool chain de deployment strategy

To Streamline, the configuration, the continous delivery as well as the deployment, we used the following DevOps tools:

- Git: we used it to control the different versions of the system, created Github hook to trigger automatically the build of the project everytime we push code to Github.
- Docker: We used docker containers to encapsulate each service (Postfix, Dovecot, Roundube, Mysql and the web admin tool). We built the containers using Dockerfiles, ubuntu as base image and we included our own configuration files.
- Docker Compose: We used docker compose to define the services, their interdependencies, the network as well as volumes. This allow a seamless startup using a single command.
- Ansible: We automated the task such as the deployment of the system as well as some configurations
- Jenkins: We used jenkins to create a CI/CD pipeline that pulls the code from Github. I this pipeline, we built docker images, pushed them to docker hub, run the containers and deploy them. To do so, we creaded a Jenkinsfile which defined stages including the checkout, the build, the test and the deployment

Here is a visual of the DevOps tool chain workflow used

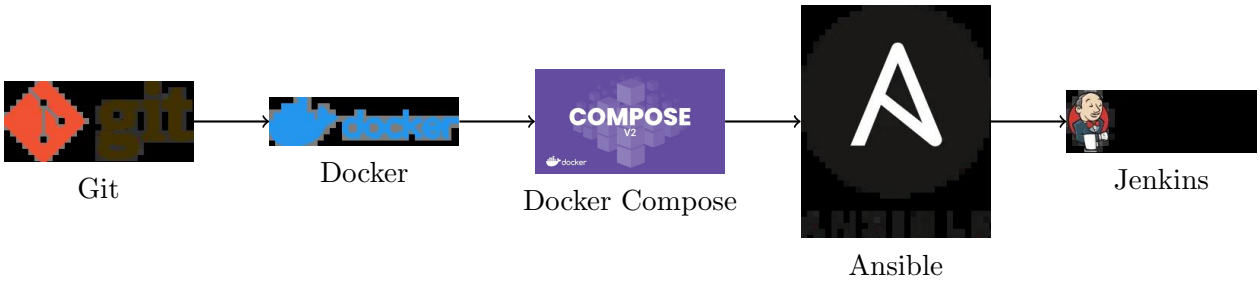


Figure 2: DevOps work flow.

We planned to use Kubernetes to scale the deployment but it was not neccessary. Moreover, the configurations of postfix and dovecot needed static IPV4 adress to communicate between dovecot, postfix and the database. Since kubernetes does not allow us to create our own network and provide the static ipv4 adress, we preferred using docker compose instead.

5 Implementation

This section details the steps taken to implement the system. It includes:

- Setting up the initial project structure
- Configuration of Postfix
- Configuration of Dovecot
- Creation of the Web admin tool
- integration with mysql and Roundcube using docker compose
- Pipeline creation using Jenkins
- deployment using Ansible

5.1 Setting up the initial project structure

at this level, what we did is to create a folder for the project. We named this folder, Project. in this folder, we initiated a git repository. We assembled the files related to each system in a sub folder. all the docker related things are in the docker folder, ansible, jenkins same. here is a picture of how it is organized

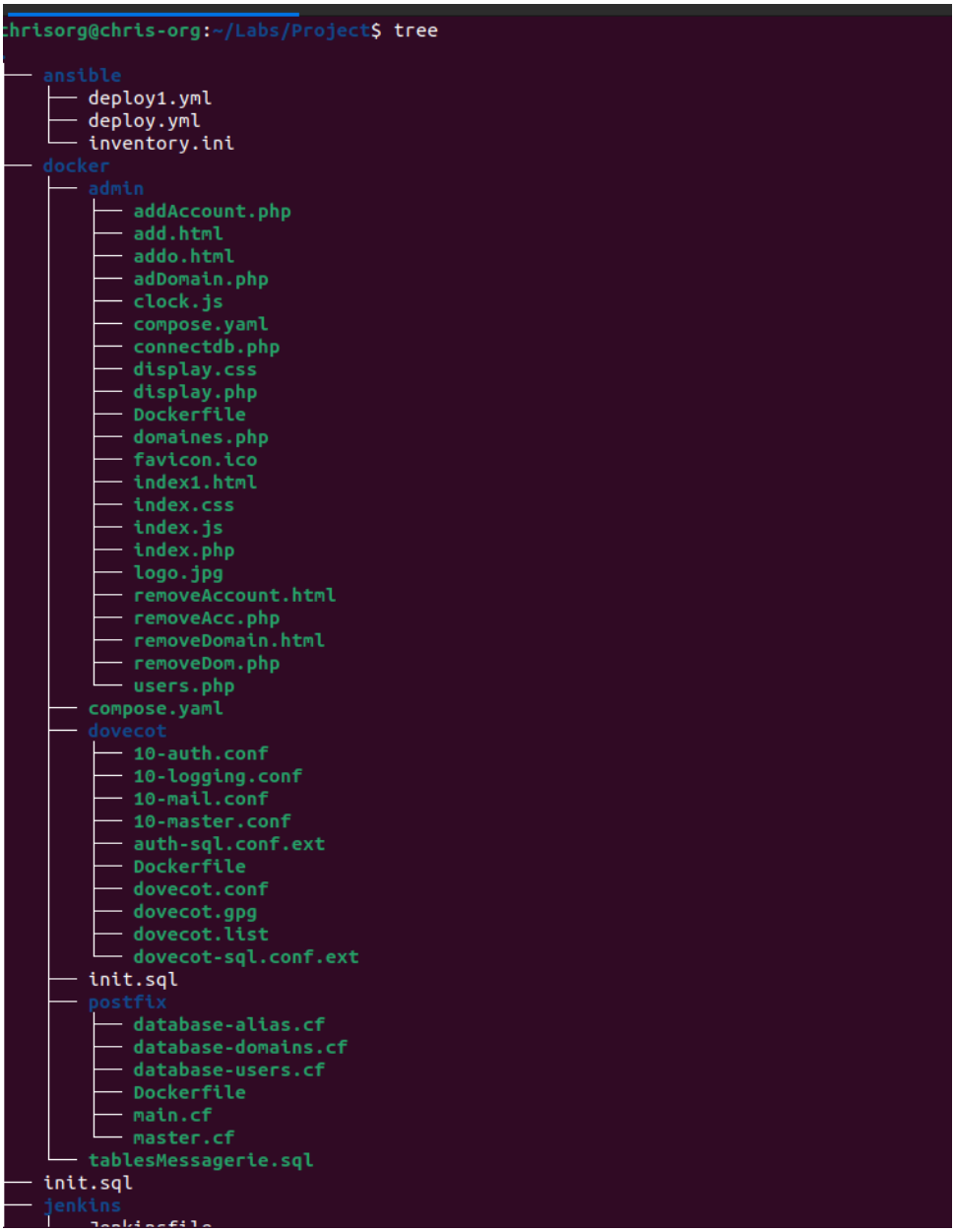


Figure 3: Project folder structure

github repository link: GitHub Repository.

5.2 Postfix's configuration

To configure postfix, we created a dockerfile using ubuntu as base image. we included 5 configuration files:

- main.cf : include configurations of smtp relay
- master.cf : define transport configurations
- database-alias.cf : posfix config file to link with the database
- database-user.cf : postfix config file to link the user management with the database

- database-domains.cf : postfix config file to link the domains management with the database
- Dockerfile: the dockerfile to create the docker image of the system

5.3 Dovecot configuration

Like Postfix, we did the same thing for dovecot. including the configuration files as well as the dockerfile

5.4 Creation of the web admin tool

This phase was crucial and Time consuming. Basically, Postfixadmin is the best tool to do it since it is a web based tool build for postfix administration. But in our case, the docker image available wasn't compatible with our system and creating our own image of postfix admin is was not working well. So we decided to create a web admin tool to manage (create, delete) user accounts and domains. It is a web based tool created using php programming language

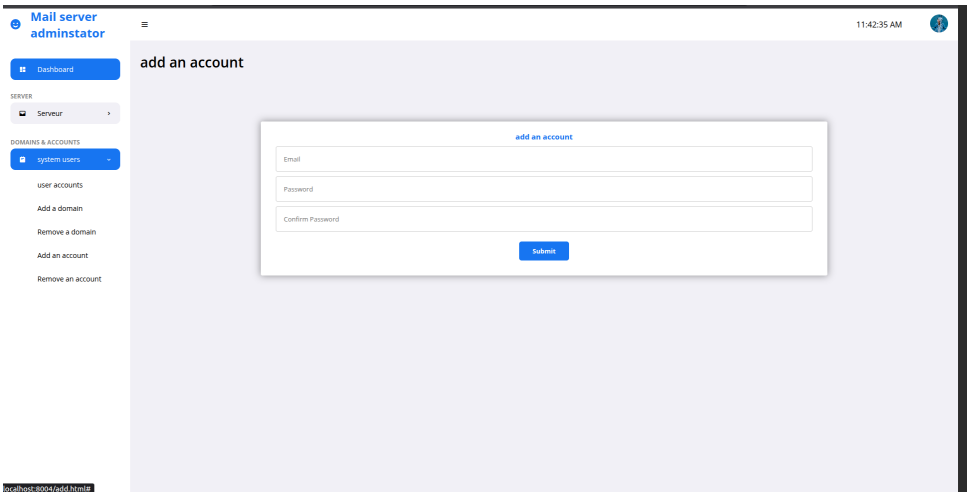
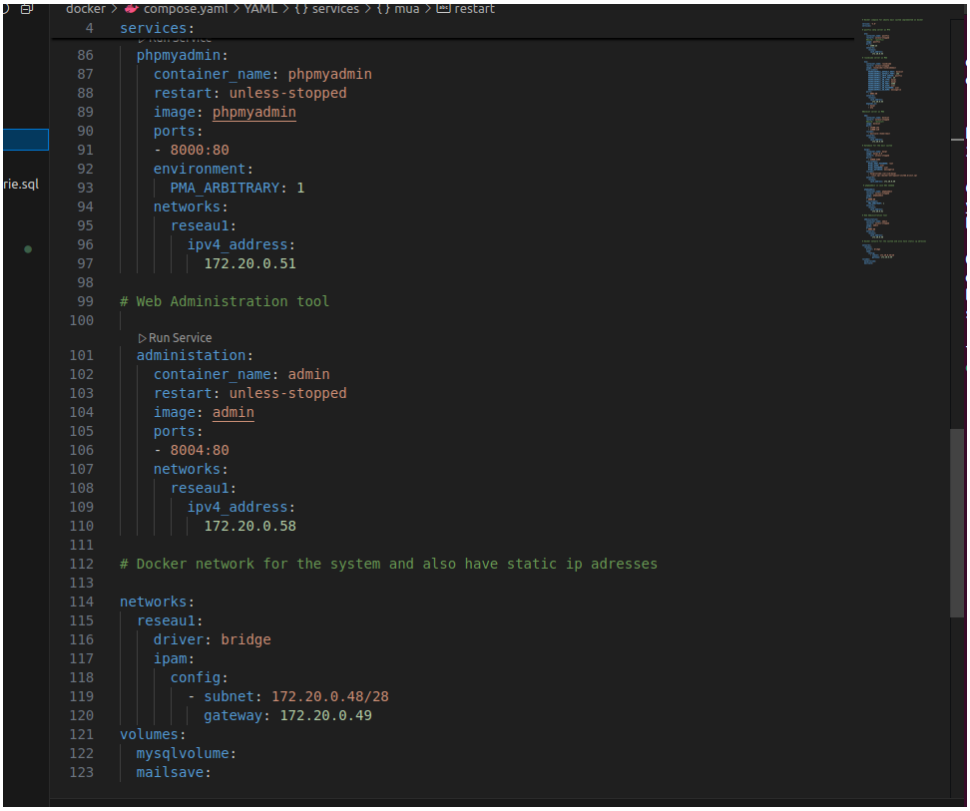


Figure 4: Pipeline

5.5 Integration with Roundcube and Mysql using docker compose

To bring all the services together, we created a docker compose file to bing all the systems. we created a volume to persist the data, and a personalized network on docker to assign static ipv4 adresses to the containers and input those IPV4 adresses in the configuration files of Dovecot and Postfix. We created some sql files to init the database on mysql and input some data to test our configurations



```
docker > compose.yaml > YAML > {} services > {} mua > restart
  4  services:
 86    phpmyadmin:
 87      container_name: phpmyadmin
 88      restart: unless-stopped
 89      image: phpmyadmin
 90      ports:
 91      - 8000:80
 92      environment:
 93        PMA_ARBITRARY: 1
 94      networks:
 95        reseau1:
 96          ipv4_address:
 97            172.20.0.51
 98
 99  # Web Administration tool
100
    ▷ Run Service
101    administation:
102      container_name: admin
103      restart: unless-stopped
104      image: admin
105      ports:
106      - 8004:80
107      networks:
108        reseau1:
109          ipv4_address:
110            172.20.0.58
111
112  # Docker network for the system and also have static ip adresses
113
114  networks:
115    reseau1:
116      driver: bridge
117      ipam:
118        config:
119          - subnet: 172.20.0.48/28
120            gateway: 172.20.0.49
121  volumes:
122    mysqlvolume:
123    mailsave:
```

Figure 5: A picture of one side of the docker compose file

We built the docker image of postfix, dovecot and admin separately before including them into the docker compose

5.6 Pipeline creation using Jenkins

We needed a jenkins pipeline to build automatically the project and ensure the continous integration as well as the continous deployment. we manage the project using git. after that, we created a pipeline in jenkins. The build job is triggered by Github SCM polling. we therefore created a Github hook to support this. Everytime we push the code to github, the build job starts. here is a picture of the pipeline built

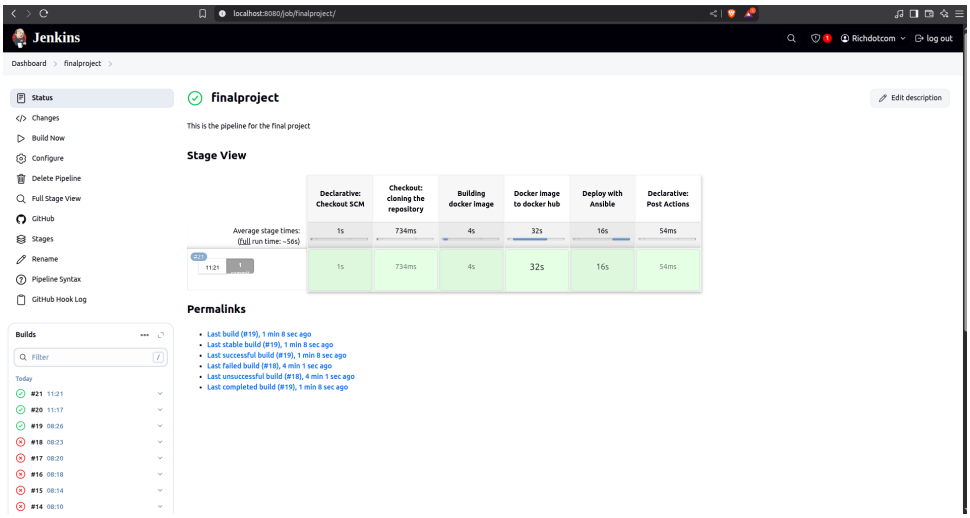


Figure 6: Pipeline

5.7 Deployment with Ansible

To ensure the deployment, we created an ansible playbook in which we check the existence of docker as well as python on the target system. The target system is our local system here since. The playbook and the inventory files are in the ansible folder of the project. Here is a picture of our pipeline being built. we can get from the output of the pipeline stage concerning the ansible deployment

Figure 7: Ansible Deployment

6 test

After building successfully, we can check that our system is deployed using docker ps command.

Figure 8: Check up to see the available containers

After this, we log in with two credentials and test the sending and reception of mails on the system

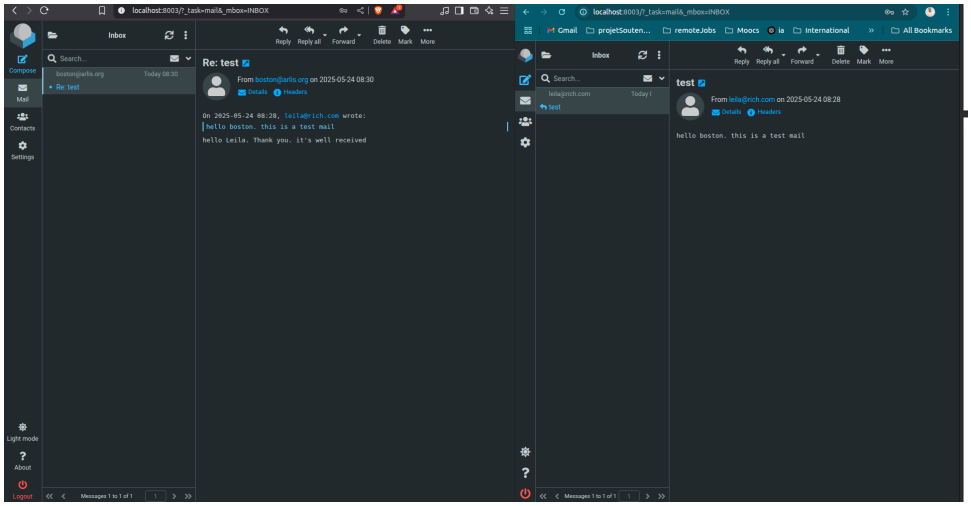


Figure 9: Check up to see the available containers

We can see that there is a mail exchange between two users. leila@rich.com and boston@arlis.org

7 Conclusion

The project delivered a fully functional and secure mail server infrastructure with automated deployment using containerization and DevOps tools. The use of Postfix, Dovecot, and Roundcube provided a reliable core email system. MySQL and the PHP admin interface simplified user management. DevOps automation using Docker, Ansible, and Jenkins ensured repeatability and reduced manual intervention.

8 Future Work for improvement

This is just a basic working version of what the project can be. It can be therefore bettered including many things:

- Include monitoring with either ELK or Grafana/Prometheus
- Include security considerations
- Implement Spam filtering using SpamAssasin and Clamav
- Rebuild the whole network and find a way to use kubernetes