Mail Server Project

CSC 651 System Administration (Spring 2019)

Project Summary

We created a mail server that allows users to send emails to each other (within the same domain). User accounts on the mail server can be managed by administrators through a web-based interface, and users are able to send and receive mail through a browser-based email client.

This was accomplished using technologies such as Nginx, Let's Encrypt, Postfix, Dovecot, PostfixAdmin, Roundcube, and MySQL.

Although mail is unable to be sent outside of the server (e.g. someone with an account on our mail server cannot send mail to a Gmail or Yahoo email address), local mail is fully functional and users are able to do everything that can be expected when it comes to emails (forwarding, replying, attaching images, flagging, etc).

The server is currently online (management at https://test-mail.online/postfixadmin, mail client at https://test-mail.online/roundcubemail). The only method of obtaining login credentials for the mail client is to be manually added by someone with an administrator account.

Group Members

Regarding specific roles, we never stuck to any strict positions over the course of working on the server and each handled multiple parts of setting it up, depending on group member availability and whatever work needed to be finished. As a result, every group member has had at least some amount of exposure to each of the components that make up the project, but nobody worked on getting a single component of the server up and running on their own.

Project Design

The server that we chose to use for hosting our application is an Amazon Web Services EC2 instance (specifically t2.micro) running Ubuntu 16.04 LTS. The specific type of instance was eligible for the AWS free tier due to its specs, meaning that we were not required to spend any money to set it up. We decided to select Ubuntu because all members of the group were more familiar with it than Windows Server, and the tutorials that we found regarding setting up a mail server used Ubuntu as the server's operating system.

We needed to get a domain name in order to set up the mail server (as the email addresses for users needed to be associated with some domain). We ended up purchasing a domain name (test-mail.online) through GoDaddy. As some domain extensions are cheaper than others, we decided to purchase a domain with the extension ".online" because of its low cost of $0.99 for the first year (although it's unlikely that we will keep this domain beyond the first year because it was purchased specifically for this project). Through GoDaddy's DNS configuration, this hostname was pointed towards our EC2 instance's IP address.

Nginx, a web server, was installed in order to access the web-based components of our project that we would install later on. Although we only planned on using this server personally for this project and had no intent to open it up for other users, we thought that it would be worth the effort to obtain an SSL certificate for our domain in order to enable. We obtained the SSL certificate files through Let's Encrypt, a certificate authority that provides people with the certificates required to enable HTTPS on their websites. We were able to receive certificate files from Let's Encrypt through a tool for Ubuntu called Certbot, which was created by the Electronic Frontier Foundation for the purpose of deploying Let's Encrypt certificates. Nginx was configured to use these files and HTTPS was enabled for our site.

A MySQL database was used to store information about users, such as their usernames, passwords, and creation dates. Alongside user information, the database also holds information about domains such as the maximum amount of mailboxes that each is allowed to contain and their mailbox quotas. Postfix Admin, a web-based interface that can be used to configure Postfix-based email servers, was used to manage user mailboxes by using the previously described MySQL database.

The back-end of our mail server is comprised of two major parts, Postfix and Dovecot. Postfix, a service that is used to send and receive emails, was set up to use virtual domains and mailboxes (i.e. domains and mailboxes limited to our domain). Postfix was configured to access the same MySQL database that Postfix Admin was able to modify, connecting the two and allowing changes made by Postfix Admin to be reflected by Postfix. Dovecot, which allows mail to be accessed through IMAP (Internet Message Access Protocol) and handles authenticating users, was configured to use the same database and mail directories as Postfix Admin and Postfix. With the two parts working together, mail is sent between users through Postfix, and mail is accessed by users through Dovecot.

For the front-end of our mail server we used an open source webmail software called Roundcube. Roundcube is a browser-based multilingual IMAP client with an application-like user interface. IMAP allows you to access your email messages wherever you are; much of the time, it is accessed via the Internet. Basically, email messages are stored on servers, so they can be accessed from any device that you connect with. It provides full functionality you expect from an email client, including MIME support, address book, folder manipulation, message searching and spell checking. MIME is a standard for formatting files of different types, such as text,

graphics, or audio, so they can be sent over the Internet and seen or played by a web browser or email application.

Tutorial

The first step to setting up an email application is first choosing a server to host your application. For our application we chose an AWS EC2 instance. There are many services to choose from, but EC2 is a commonly used service in industry and offers a free tier service for non-professional users.

The next step is installing and setting up NGINX, PHP, and MySQL:

```
sudo apt install nginx mysql-server php7.0-fpm php7.0-cli php7.0-imap php7.0-json
    php7.0-mysql php7.0-opcache php7.0-mbstring php7.0-readline
```

```
wget -q
    https://downloads.sourceforge.net/project/postfixadmin/postfixadmin/postfixadmin-
    ${VERSION}/postfixadmin-${VERSION}.tar.gz
```

Next, you need to create a new MySQL user and database using the following commands:
CREATE DATABASE postfixadmin;
GRANT ALL ON postfixadmin.* TO 'postfixadmin'@'localhost' IDENTIFIED BY 'P4ssvv0rD';
FLUSH PRIVILEGES;

Next step is to configure the PHP files to be able to access the database we just created and set the mail server values such as quota, number of mailboxes, or aliases. After modifying the files */var/www/postfixadmin/config.local.php* and */var/www/postfixadmin/upgrade.php* , a superadmin user with all privileges would be created and added to the server.

We now need to configure Postfix to use virtual mailboxes and domains. Our goal is to create configuration files to instruct Postfix how to access our MySQL database. The required cf files are: *mysql_virtual_domains_maps.cf, mysql_virtual_alias_maps.cf, mysql_virtual_alias_domain_maps.cf, mysql_virtual_alias_domain_catchall_maps.cf, mysql_virtual_mailbox_maps.cf, mysql_virtual_alias_domain_mailbox_maps.cf.* All of these should be located under */etc/postfix/sql/* and each contains the information needed to connect to the database and one SQL query to retrieve specific information needed to perform a task. Once the SQL configuration files are created, we need to tell Postfix about these files by using the command *postconf.* Also, since we are building a mail server, we need to set the submission port (587) and smtps port (465) inside */etc/postfix/master.cf.*

Lastly, we need to configure Dovecot to match our mail server configuration. The first file we need to modify is */etc/dovecot/dovecot-sql.conf.ext* to instruct Dovecot how to access our databases and find the information about email accounts. Next, the file *conf.d/10-mail.conf* will set values for users and groups in our mail server. We then set authentication up by modifying the files *conf.d/10-auth.conf* and *conf.d/10-master.conf*. To enable SSL/TLS, we set our certification inside *conf.d/10-ssl.conf*. We have to set the imap quota inside *conf.d/20-imap.conf* and the lmtp configuration inside *conf.d/20-lmtp.conf*. By modifying *conf.d/20-lmtp.conf*, we define the default mailboxes.

Additionally, we can set quota limits and run a script that will send an email to the user when this quota is about to be reached by modifying *conf.d/90-quota.conf*. As a last step, we need to tell Dovecot how to access the quota SQL dictionary by editing the file *conf.d/90-quota.conf.ext*.

After restarting the Dovecot service, we now have a fully functional mail system.

References

Tutorials

PostfixAdmin Configuration:
https://linuxize.com/post/set-up-an-email-server-with-postfixadmin/

Nginx Configuration:
https://linuxize.com/post/how-to-install-nginx-on-ubuntu-16-04/

SSL Certificates:
https://linuxize.com/post/secure-nginx-with-let-s-encrypt-on-ubuntu-16-04/

Postfix and Dovecot Configuration:
https://linuxize.com/post/install-and-configure-postfix-and-dovecot/

Roundcube Configuration:
https://linuxize.com/post/install-and-configure-roundcube-webmail/

Documentation

AWS EC2 User Guide:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

GoDaddy Help Center:
https://www.godaddy.com/help

Nginx Wiki:
https://www.nginx.com/resources/wiki/

Let's Encrypt Documentation:
https://letsencrypt.org/docs/

MySQL Documentation:
https://dev.mysql.com/doc/

PostfixAdmin Wiki:
https://sourceforge.net/p/postfixadmin/wiki/Home/
Postfix Documentation:
http://www.postfix.org/documentation.html

Dovecot Wiki:
https://wiki2.dovecot.org/

Roundcube Wiki:
https://github.com/roundcube/roundcubemail/wiki