SMTP Server Deployment Using Docker Compose

November 29, 2024

Introduction

This document explains how the given `docker-compose.yml` file works to deploy an SMTP server securely using Docker secrets for credentials management. The `smtp_user` and `smtp_password` files are generated with the following commands:

```
echo "your_email@gmail.com" > smtp_user
echo "your_secure_password" > smtp_password
```

Docker Compose Explanation

Services

smtp-server

- **Image:** The service uses the `crazymax/msmtpd` image, a lightweight SMTP server designed to forward emails through an external SMTP server.
- **Container Name:** The container is named `smtp-server`.
- **Secrets:** The service uses Docker secrets to manage SMTP credentials securely:
  - `smtp_user`: stores the email address (user) for authentication.
  - `smtp_password`: stores the password associated with the user.
- **Environment Variables:** These variables configure the SMTP server to connect with the external email service:
  - `SMTP_HOST`: Specifies the outgoing mail server, in this case, `smtp.gmail.com`.
  - `SMTP_PORT`: Port `587`, the standard port for secure TLS connections.
  - `SMTP_TLS`: Enables TLS with `on`, ensuring data transmission security.
  - `SMTP_USER_FILE`: Points to the secret file containing the username (`/run/secrets/smtp_user`).
  - `SMTP_PASSWORD_FILE`: Points to the secret file containing the password (`/run/secrets/smtp_password`).
- **Automatic Restart:** The `restart: always` directive ensures that the container restarts automatically in case of failure.

Secrets

Docker secrets are used to securely manage sensitive information:

- `smtp_user`**:** File `smtp_user` contains the user's email address, created with the command:

  ```
  echo "your_email@gmail.com" > smtp_user
  ```

- `smtp_password`**:** File `smtp_password` contains the user's password, created with the command:

  ```
  echo "your_secure_password" > smtp_password
  ```

Data and Backup Management

Data

Credentials are not stored directly in the container or exposed through environment variables. Instead, they are securely handled using Docker secrets, which are mounted at `/run/secrets/` inside the container.

Backups

Since credentials are stored in external files (`smtp_user` and `smtp_password`), it is crucial to back up these files securely, avoiding exposure to external sources. It is recommended to encrypt these files if stored outside the production environment.

Advantages of Using Secrets

- **Enhanced Security:** Credentials are not exposed in the code or as plain environment variables.
- **Integrity:** Docker natively manages secrets, ensuring their secure delivery to authorized containers.

This approach is ideal for environments where security is critical, such as configuring email services requiring secure authentication with external servers.