Optimizing Container Deployment with os-shell

Tomasz Olejarczuk

Fontys

21-06-2024

Contents

Optimizing Container Deployment with `os-shell`

1. Summary

This document outlines the integration of the `bitnami/os-shell` container into the Teamium project's Docker Compose environment. The primary purpose of this container is to simplify the management of file and folder permissions within the shared volumes used by different containers in the system. By executing scripts within `bitnami/os-shell` before other containers start, we can ensure that necessary permissions are set correctly, preventing potential errors and access issues during runtime.

2. Introduction

2.1. Background

The Teamium project utilizes Docker Compose to orchestrate a multi-container environment, including the MATLAB Production Server, API wrapper, MinIO storage, PostgreSQL database, and RabbitMQ message broker. These containers often need to share data and files through Docker volumes. However, differences in user IDs (UIDs) and group IDs (GIDs) between containers and the host system can lead to permission problems when accessing files within shared volumes.

For example, if the MATLAB container writes a file to a shared volume with its default UID, the API wrapper might not be able to read that file if its UID is different. This can cause unexpected errors and disruptions in the system's operation.

2.2. Objectives

The main objectives of incorporating `bitnami/os-shell` are:

1. **Permission Management:** Automate the process of setting correct file and folder permissions within shared volumes to ensure consistent access across all containers.
2. **Streamlined Deployment:** Simplify the Docker Compose setup by removing the need for manual permission adjustments after container creation.
3. **Error Prevention:** Prevent runtime errors caused by incorrect permissions in shared volumes.

3. Methodology

3.1. `bitnami/os-shell` Container

The `bitnami/os-shell` container is a lightweight utility container that provides a shell environment within the Docker ecosystem. It allows you to execute scripts and commands during the container build or startup process.

3.2. Docker Compose Integration

In the `docker-compose.yml` file, we've added a new service named `prepare-data` that utilizes the `bitnami/os-shell` image:

```
prepare-data:
    image: bitnami/os-shell:latest
    user: root
    command:
        - /bin/bash
        - -ec
        - |
            chown -R 1001:0 /minio_data
            mkdir -p /data/log
            chmod -R 777 /data
    volumes:
        - miniovol:/minio_data
        - datavol:/data
    networks:
        - matlab-network
```

Explanation:

- **Image:** The `bitnami/os-shell:latest` image is used to provide a simple shell environment for executing setup commands.
- **Volumes:**
  - **`miniovol:/minio_data`:** This line mounts the named Docker volume `miniovol` into the `/minio_data` directory within the `prepare-data` container. This is the directory where MinIO stores its data. By mounting this volume, the `prepare-data` container can access and modify files within MinIO's data directory.
  - **`datavol:/data`:** This line mounts another named Docker volume, `datavol`, into the `/data` directory within the container. This volume is shared with other containers (`matlab-algorithm` and `matlab-wrapper`) and is used for exchanging data and log files.
- **Entrypoint and Command:**

- **`/bin/bash -ec`:** This specifies that the container should start a Bash shell (`/bin/bash`) and execute the commands that follow in a non-interactive mode (`-c`). The `-e` flag ensures that the container stops if any of the commands exit with an error.
- **`chown -R 1001:0 /minio_data`:** This command recursively changes the ownership of all files and directories within the `/minio_data` directory (which corresponds to the `miniovol` volume) to the user with UID 1001 and the group with GID 0. The user with UID 1001 is the default user in the MinIO Docker image, and setting the owner to this user ensures that MinIO can write data to the volume.
- **`mkdir -p /data/log`:** This command creates the `/data/log` directory within the `/data` volume (mounted as `datavol`) if it doesn't already exist. This directory is used for storing MATLAB log files.
- **`chmod -R 777 /data`:** This command recursively sets the permissions of all files and directories within the `/data` directory to `777`. This means that all users (owner, group, and others) have full read, write, and execute permissions. This ensures that both the MATLAB algorithm and the API wrapper can access and modify files within this shared volume without encountering permission errors.

4. Results & Insights

4.1. Key Outcomes

- **Automated Permission Setting:** File and folder permissions within the shared volumes are now set automatically during the Docker Compose startup process.
- **Elimination of Manual Steps:** Developers and system administrators no longer need to manually adjust permissions after container creation.
- **Improved Reliability:** The risk of permission-related errors during runtime is significantly reduced, leading to a more stable and predictable system. ## 4.2. Further Considerations

While `bitnami/os-shell` effectively addresses permission issues in this context, consider these additional points for future optimization:

- **Custom Scripts:** If you have more complex permission requirements, you can create custom scripts and place them in the shared volume. The `command` in the `docker-compose.yml` file can then execute these scripts to tailor permissions to your specific needs.
- **UID/GID Alignment:** If you're using container images from different vendors or with different default users, ensure that the UID/GID used in the `chown` command within `bitnami/os-shell` aligns with the user in the other containers accessing the shared volumes.

5. Conclusion

By incorporating the `bitnami/os-shell` container into our Docker Compose setup, we have significantly simplified the management of file permissions within shared volumes. This improvement streamlines the deployment process, enhances system reliability, and prevents errors caused by permission mismatches.