



## 容器日誌問題解決



```
dockerfile:# 使用 Python 3.9 slim 作為基礎映像
FROM python:3.9-slim

# 更新系統並安裝必要依賴和 MariaDB
RUN apt-get update && apt-get install -y \
    mariadb-server \
    mariadb-client \
    libmariadb-dev \
    gcc \
    libffi-dev \
    libssl-dev \
    && apt-get clean

# 設置 MariaDB root 密碼和數據庫名稱
ENV MARIADB_ROOT_PASSWORD=yourpassword
ENV MARIADB_DATABASE=fastapi_crud

# # 初始化 MariaDB 數據目錄並設置 root 密碼
# RUN mariadb-install-db --user=mysql --datadir=/var/lib/mysql && \
#     mysqld_safe & \
#     sleep 30 && \
#     mysqladmin --user=root password
"${MARIADB_ROOT_PASSWORD}" && \
#     mysql -u root -p"${MARIADB_ROOT_PASSWORD}" -e "FLUSH
PRIVILEGES" && \
#     pkill -f mysqld_safe

# 複製初始化 SQL 文件到容器
COPY init.sql /docker-entrypoint-initdb.d/

# 安裝 FastAPI 和 Python 依賴
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 複製應用程式碼
WORKDIR /app
COPY . /app/

# 暴露服務端口
EXPOSE 3306 8000

# # 啟動 MariaDB 和 FastAPI
# CMD ["sh", "-c", "mysqld_safe & sleep 30 && uvicorn main:app --
host 0.0.0.0 --port 8000"]
```



```
# # 啟動 Bash，允許手動執行 MariaDB 和 FastAPI
# CMD ["/bin/bash"]

# 啟動 Bash，允許手動執行 MariaDB 和 FastAPI
CMD ["/bin/bash", "-c", "tail -f /dev/null"] , container啟用後，argocd ui
裡沒有log
```



在你的 `Dockerfile` 配置中，容器啟動後採用了 `tail -f /dev/null` 的方式保持容器運行，這樣會導致容器本身沒有執行任何應用邏輯或服務，因此 `argocd` 的 UI 中看不到任何 log 輸出。

以下是可能的解決方案：

## 1. 確認容器需要執行的主要服務

如果你希望容器啟動時同時運行 MariaDB 和 FastAPI 應用，應該在 `CMD` 中啟動這些服務，而非使用 `tail -f /dev/null`。

以下是修改範例：

```
# CMD 啟動 MariaDB 和 FastAPI
CMD ["sh", "-c", "mysqld_safe & sleep 10 && uvicorn main:app --host 0.0.0.0 --port 8000"]
```

此設置中：

- `mysqld_safe` 啟動 MariaDB。
- `sleep 10` 確保 MariaDB 有足夠時間啟動。
- `uvicorn` 啟動 FastAPI 應用。

## 2. 確認 `init.sql` 文件是否正確初始化數據庫

你複製了一個 `init.sql` 文件到 `/docker-entrypoint-initdb.d/`，這適用於 MariaDB 官方的 `docker-entrypoint.sh` 腳本。確保該 SQL 文件能正常執行，例如創建數據庫、表等。

## 3. 檢查 `requirements.txt` 是否安裝正確

確保你的 `requirements.txt` 包含 `uvicorn`、`fastapi` 和其他必要的依賴。

## 4. 確保 Argocd 部署環境的容器設定正確

在 `argocd` 部署 YAML 或 Helm chart 中，檢查是否正確配置了容器的 logs 或 stdout 輸出。通常需要確保容器的 `stdout` 或 `stderr` 日誌流向正確的位置。



例如，`kubectl logs` 命令應能顯示輸出的內容：

```
kubectl logs <pod-name>
```

## 5. 測試和排錯

若仍無法正常啟動，進入容器內部排查問題：

```
docker exec -it <container-id> /bin/bash
```

檢查：

- MariaDB 是否正常運行（`ps aux | grep mysqld`）。
- FastAPI 是否啟動並綁定到指定端口。
- 查看具體日誌文件（如 MariaDB 的 `error.log`）。

## 修改後的完整 `Dockerfile`

以下是根據你的需求和問題提供的完整修訂版本：

```
# 使用 Python 3.9 slim 作為基礎映像
FROM python:3.9-slim

# 更新系統並安裝必要依賴和 MariaDB
RUN apt-get update && apt-get install -y \
    mariadb-server \
    mariadb-client \
    libmariadb-dev \
    gcc \
    libffi-dev \
    libssl-dev \
    && apt-get clean

# 設置 MariaDB root 密碼和數據庫名稱
ENV MARIADB_ROOT_PASSWORD=yourpassword
ENV MARIADB_DATABASE=fastapi_crud

# 複製初始化 SQL 文件到容器
COPY init.sql /docker-entrypoint-initdb.d/

# 安裝 FastAPI 和 Python 依賴
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

# 複製應用程式碼
```



```

WORKDIR /app
COPY . /app/

# 暴露服務端口
EXPOSE 3306 8000

# 啟動 MariaDB 和 FastAPI
CMD ["sh", "-c", "mysqld_safe & sleep 10 && uvicorn main:app --host 0.0.0.0 --port 8000"]

```

這樣可以確保容器內的服務正常啟動，且 `argocd` 可以捕捉到應用的標準輸出日誌。



```

# 使用 Python 3.9 slim 作為基礎映像
FROM python:3.9-slim

# 更新系統並安裝必要依賴和 MariaDB
RUN apt-get update && apt-get install -y \
    mariadb-server \
    mariadb-client \
    libmariadb-dev \
    gcc \
    libffi-dev \
    libssl-dev \
    && apt-get clean

# 設置 MariaDB root 密碼和數據庫名稱
ENV MARIADB_ROOT_PASSWORD=yourpassword
ENV MARIADB_DATABASE=fastapi_crud

# # 初始化 MariaDB 數據目錄並設置 root 密碼
# RUN mariadb-install-db --user=mysql --datadir=/var/lib/mysql && \
#     mysqld_safe & \
#     sleep 30 && \
#     mysqladmin --user=root password \
#     "${MARIADB_ROOT_PASSWORD}" && \
#     mysql -u root -p"${MARIADB_ROOT_PASSWORD}" -e "FLUSH \
#     PRIVILEGES" && \
#     pkill -f mysqld_safe

# 複製初始化 SQL 文件到容器
COPY init.sql /docker-entrypoint-initdb.d/

# 安裝 FastAPI 和 Python 依賴
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 複製應用程式碼
WORKDIR /app

```



```
COPY ./app/

# 暴露服務端口
EXPOSE 3306 8000

# # 啟動 MariaDB 和 FastAPI
# CMD ["sh", "-c", "mysqld_safe & sleep 30 && uvicorn main:app --host 0.0.0.0 --port 8000"]

# # 啟動 Bash，允許手動執行 MariaDB 和 FastAPI
# CMD ["/bin/bash"]

# 啟動 Bash，允許手動執行 MariaDB 和 FastAPI
CMD ["/bin/bash", "-c", "tail -f /dev/null"]，直接改成CMD ["/bin/bash", "-c", "tail -f logs/app.log"]，container啟動後再手動啟用MariaDB 和 FastAPI
```



如果你希望在容器啟動後透過 `tail -f logs/app.log` 持續查看指定的日誌檔案，但 MariaDB 和 FastAPI 需要手動啟動，以下是一些調整和建議：

## 修改 `Dockerfile`

將 `CMD` 語句改為：

```
CMD ["/bin/bash", "-c", "tail -f logs/app.log"]
```

這樣設置後，容器會在啟動後自動監控 `logs/app.log`，同時允許你在容器內部手動啟動 MariaDB 和 FastAPI。

## 操作步驟

### 1. 構建和啟動容器

構建 Docker 映像並啟動容器：

```
docker build -t your-image-name .
docker run -d --name your-container-name your-image-name
```

### 2. 進入容器

進入容器以手動啟動 MariaDB 和 FastAPI：

```
docker exec -it your-container-name /bin/bash
```





### 3. 啟動 MariaDB

在容器內執行以下命令啟動 MariaDB：

```
mysqld_safe &
```

### 4. 啟動 FastAPI

在容器內啟動 FastAPI 應用（假設入口文件為 `main.py` 且應用對應的 ASGI app 是 `app`）：

```
uvicorn main:app --host 0.0.0.0 --port 8000
```

### 5. 檢查日誌

`tail -f logs/app.log` 會持續監控日誌文件的輸出。確保應用的日誌正確寫入到該檔案中。如果沒有日誌，檢查應用是否配置正確。

## 確保日誌寫入 `logs/app.log`

FastAPI 默認情況下將日誌輸出到 `stdout`，需要將日誌配置為輸出到文件。例如，在 FastAPI 應用中配置日誌文件：

```
import logging

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("logs/app.log"), # 設置日誌輸出到文件
        logging.StreamHandler() # 同時輸出到控制台
    ]
)
```

確保 `logs/` 目錄存在，並具有正確的寫入權限。

## 注意事項

#### 1. 初始化 SQL 的問題

如果 `init.sql` 用於 MariaDB 的初始化，但 MariaDB 在啟動後未執行，需手動確認初始化操作。

#### 2. 日誌文件的持久性

如果需要保留日誌，即使容器被刪除，也建議將日誌文件掛載到宿主機：

```
docker run -d -v /path/to/host/logs:/app/logs your-image-name
```



這樣的配置允許你靈活啟動服務並監控日誌檔案，同時確保 `argocd` 等工具可以正常收集日誌輸出。



這修正做法能修復argocd ui沒有log的問題嗎



這樣的修改可能無法直接解決 `ArgoCD UI` 中無法看到 log 的問題，因為 `ArgoCD` 通常只會顯示容器標準輸出（`stdout`）或標準錯誤輸出（`stderr`）的日誌內容。如果你的應用將日誌輸出到文件（如 `logs/app.log`），這些日誌不會自動顯示在 `ArgoCD` 的 UI 中。

## 解決方法

為了讓 `ArgoCD UI` 顯示日誌，需要確保日誌直接輸出到容器的標準輸出或標準錯誤。以下是具體解決方法：

### 1. 修改 FastAPI 的日誌配置

將 FastAPI 的日誌輸出到標準輸出，這樣 `ArgoCD` 就能捕捉到。

#### 修改日誌配置

在 FastAPI 程式中設定日誌：

```
import logging

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    handlers=[
        logging.StreamHandler() # 只輸出到標準輸出
    ]
)
```

### 2. 直接使用標準輸出代替 `tail -f`

不要使用 `tail -f logs/app.log`，而是將應用的日誌輸出直接流向容器的標準輸出。這樣 ArgoCD 可以捕捉到日誌內容。



修改 `CMD` 為：

```
CMD ["sh", "-c", "mysqld_safe & uvicorn main:app --host 0.0.0.0 --port 8000"]
```

此設定啟動 MariaDB 和 FastAPI，並確保 FastAPI 的日誌會流向標準輸出。

---

### 3. 保留 MariaDB 的日誌輸出

MariaDB 的日誌通常寫入到文件。如果你需要這些日誌，也輸出到標準輸出，可以修改 MariaDB 的配置：

#### 方法 1：在 `mysqld\_safe` 中啟用標準輸出

將 MariaDB 配置調整，讓日誌輸出到 `stderr`：

```
CMD ["sh", "-c", "mysqld_safe --log-error=/dev/stderr & uvicorn main:app --host 0.0.0.0 --port 8000"]
```

#### 方法 2：使用符號連結

讓 MariaDB 的日誌文件流向標準輸出：

```
RUN ln -sf /dev/stdout /var/log/mysql/mysql.log && \  
ln -sf /dev/stderr /var/log/mysql/error.log
```

這樣，MariaDB 的所有日誌也會出現在標準輸出。

---

### 4. 測試 ArgoCD 日誌顯示

部署後，檢查 ArgoCD 的 UI 是否可以看到日誌：

1. 使用 **ArgoCD** 界面查看容器的日誌。
2. 或者使用 `kubectl logs` 檢查日誌：

```
kubectl logs <pod-name>
```

### 5. 確認應用部署和日誌策略





如果問題依然存在，檢查以下項目：

1. **ArgoCD** 是否正確配置容器的標準輸出日誌收集。
2. **Kubernetes** 的 `pod` 日誌是否已被收集。
  - 檢查 Pod 的 `stdout`/`stderr`` 是否記錄正確。
3. 應用是否異常退出，導致日誌流被終止。

透過上述修改，應能讓 ArgoCD 的 UI 顯示你的應用日誌。