

Docker 鏡像優化

資工二 S1254013 楊敦傑

1. 引言

在手動部署 Kubernetes (K8s) 集群時，我遇到最多的問題是 Dockerfile 建構容器時，容器總是會自動關閉，這時 AI 總會告訴你容器內的建構有錯，但我始終無法找到有效的解決方案。這些問題不僅妨礙了應用程式的順利部署，還影響了後續的開發與測試。經過一段時間的調查與反思，我意識到，即使 Dockerfile 的內容看似正確，容器的性能和穩定性仍可能受到多種因素的影響。因此，我決定深入研究並優化 Dockerfile，目的是提升容器的穩定性、縮短建構時間，並提高容器運行的效率。

2. 問題

```
FailDockerfile X
k8s-multi-node-2 > FailDockerfile > ...
1  # 使用 Ubuntu 作為基礎映像
2  FROM ubuntu:latest
3
4  # 安裝所需的軟體包
5  RUN apt-get update -y && \
6      apt-get install -y systemd nano firewalld && \
7      apt-get clean && \
8      rm -rf /var/lib/apt/lists/*
9
10 # 啟用 systemd 支援
11 ENV container=docker
12 |
13 # 設置停止信號
14 STOPSIGNAL SIGRTMIN+3
15
16 # 添加一個 systemd 服務來保持容器運行
17 RUN echo "[Unit]\nDescription=Sleep Infinity\n\n[Service]\nExecStart=/"
18
19 # 啟用 sleep 服務
20 RUN systemctl enable sleep.service
21
22 # 設置 systemd 為主進程
23 CMD ["/lib/systemd/systemd"]
24
```

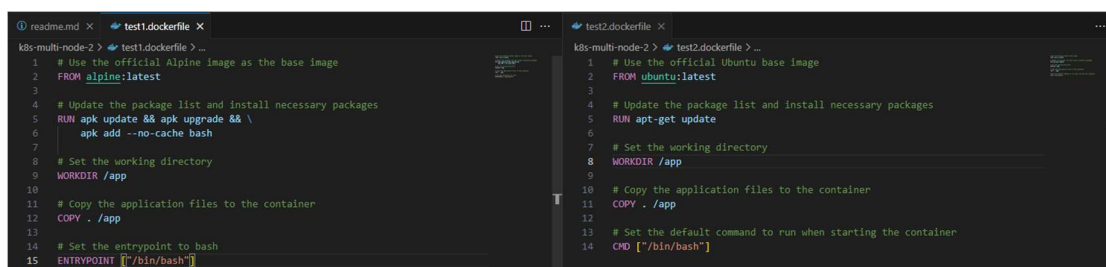
```
alpine 3.16.3 b95359c25051 2 years ago 9.61MB
grason@LAPTOP-30V3PCQM:~/developer/k8s/k8s-multi-node-2$ docker run -d -it test3
42b2475c8fb90675ee18d0eafdcf801f619053387f5d66905010edb249b24ba4
grason@LAPTOP-30V3PCQM:~/developer/k8s/k8s-multi-node-2$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
grason@LAPTOP-30V3PCQM:~/developer/k8s/k8s-multi-node-2$ docker run -d -it test1
f1aef9ea84398f9c005776f81c951ad38a13afa88db78f4bcd07e59797a8e328
grason@LAPTOP-30V3PCQM:~/developer/k8s/k8s-multi-node-2$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
f1aef9ea8439   test1     "/bin/bash"   4 seconds ago   Up 4 seconds   focused_newton
grason@LAPTOP-30V3PCQM:~/developer/k8s/k8s-multi-node-2$
```

這是我構建的 Dockerfile，乍看之下似乎沒有問題，但容器卻無法正確執行。問題出在過度使用了多個指令進行構建。Dockerfile 中的 FROM、RUN、ENV、CMD 等指令會按照順序依次執行，每條指令都會創建一個新的鏡像層，並且這些層會逐層疊加，並不是平行處理。例如，每執行一次 RUN，Docker 都會為該層建立一個新的檔案系統層，這樣設計導致容器在運行時無法正確啟動或出現錯誤。

3. 鏡像優化

1. 使用基礎鏡像

- 如圖所示，我寫了兩個 Dockerfile，分別為 test1.dockerfile 和 test2.dockerfile，並要求這兩者都能使用 Bash，其中一個選用了 Alpine 作為基礎鏡像，另一個則選用了常見的 Ubuntu。從圖中的鏡像大小可以看出，兩者達到了相同的功能效果，但鏡像的大小和建構時間卻有所不同。



```
test1.dockerfile
1 # Use the official Alpine image as the base image
2 FROM alpine:latest
3
4 # Update the package list and install necessary packages
5 RUN apk update && apk upgrade && \
6     apk add --no-cache bash
7
8 # Set the working directory
9 WORKDIR /app
10
11 # Copy the application files to the container
12 COPY . /app
13
14 # Set the entrypoint to bash
15 ENTRYPOINT ["/bin/bash"]

test2.dockerfile
1 # Use the official Ubuntu base image
2 FROM ubuntu:latest
3
4 # Update the package list and install necessary packages
5 RUN apt-get update
6
7 # Set the working directory
8 WORKDIR /app
9
10 # Copy the application files to the container
11 COPY . /app
12
13 # Set the default command to run when starting the container
14 CMD ["/bin/bash"]
```

```
grason@LAPTOP-3OV3PCQM:~/developer/k8s/k8s-multi-node-2$ docker images
```

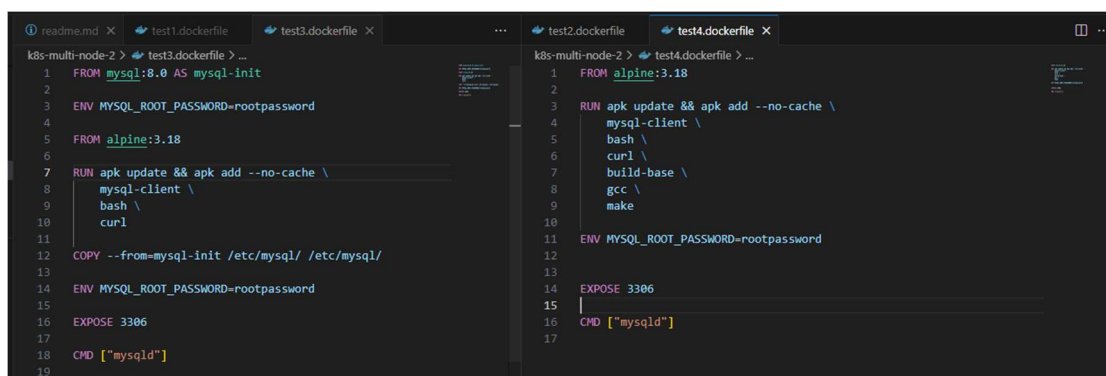
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
test2	latest	56ce51ba4698	4 seconds ago	191MB
test1	latest	4fb2089e1680	10 seconds ago	28.1MB

```
grason@LAPTOP-3OV3PCQM:~/developer/k8s/k8s-multi-node-2$ docker build -t test1 -f test1.dockerfile .
[+] Building 1.9s (9/9) FINISHED
```

```
grason@LAPTOP-3OV3PCQM:~/developer/k8s/k8s-multi-node-2$ docker build -t test2 -f test2.dockerfile .
[+] Building 2.2s (9/9) FINISHED
```

2. 多階段構建

- 如圖所示，我寫了兩個 Dockerfile，分別為 test3.dockerfile 和 test4.dockerfile，實現 alpine 中安裝 mysql，前者使用多階段構建，後者單階段構建，兩個鏡像輸出大小不同，主要是因為 Docker 只會保留不同的階段中最終需要的文件和依賴項。



```
test3.dockerfile
1 FROM mysql:8.0 AS mysql-init
2
3 ENV MYSQL_ROOT_PASSWORD=rootpassword
4
5 FROM alpine:3.18
6
7 RUN apk update && apk add --no-cache \
8     mysql-client \
9     bash \
10    curl
11
12 COPY --from=mysql-init /etc/mysql/ /etc/mysql/
13
14 ENV MYSQL_ROOT_PASSWORD=rootpassword
15
16 EXPOSE 3306
17
18 CMD ["mysqld"]

test4.dockerfile
1 FROM alpine:3.18
2
3 RUN apk update && apk add --no-cache \
4     mysql-client \
5     bash \
6     curl \
7     build-base \
8     gcc \
9     make
10
11 ENV MYSQL_ROOT_PASSWORD=rootpassword
12
13
14 EXPOSE 3306
15
16 CMD ["mysqld"]
```

```
grason@LAPTOP-3OV3PCQM:~/developer/k8s/k8s-multi-node-2$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
test44	latest	dbc367b97c22	42 seconds ago	453MB
test33	latest	6543bd50ace4	About a minute ago	126MB

3. 組合、合併容器內的圖層

- 因為每執行一次 RUN 都會建立新的檔案系統層，未知的錯誤會隱藏其中，同時也會造成創建時，速度較慢。

```
RUN apt-get update
RUN apt-get install -y curl vim
RUN apt-get clean
```

-

```
RUN apt-get update && apt-get install -y curl vim \
| && apt-get clean
```

-