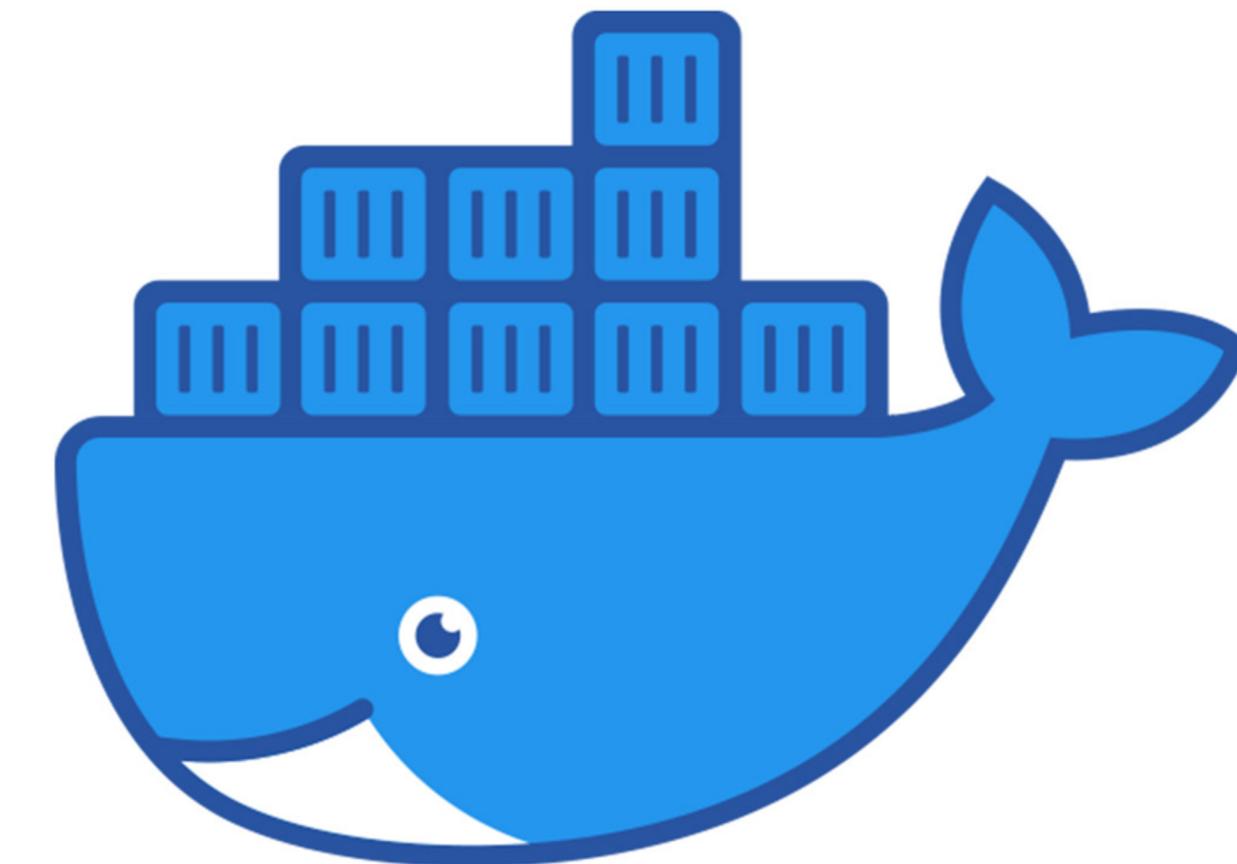


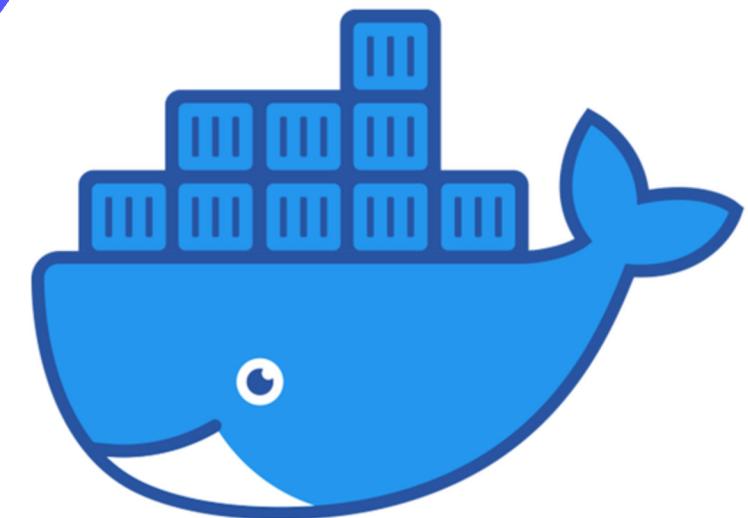
DOCKER MOUNT & LOG



Docker Storage là hệ thống quản lý dữ liệu trong Docker, cho phép lưu trữ dữ liệu bền vững (persistent) và chia sẻ dữ liệu giữa các container.

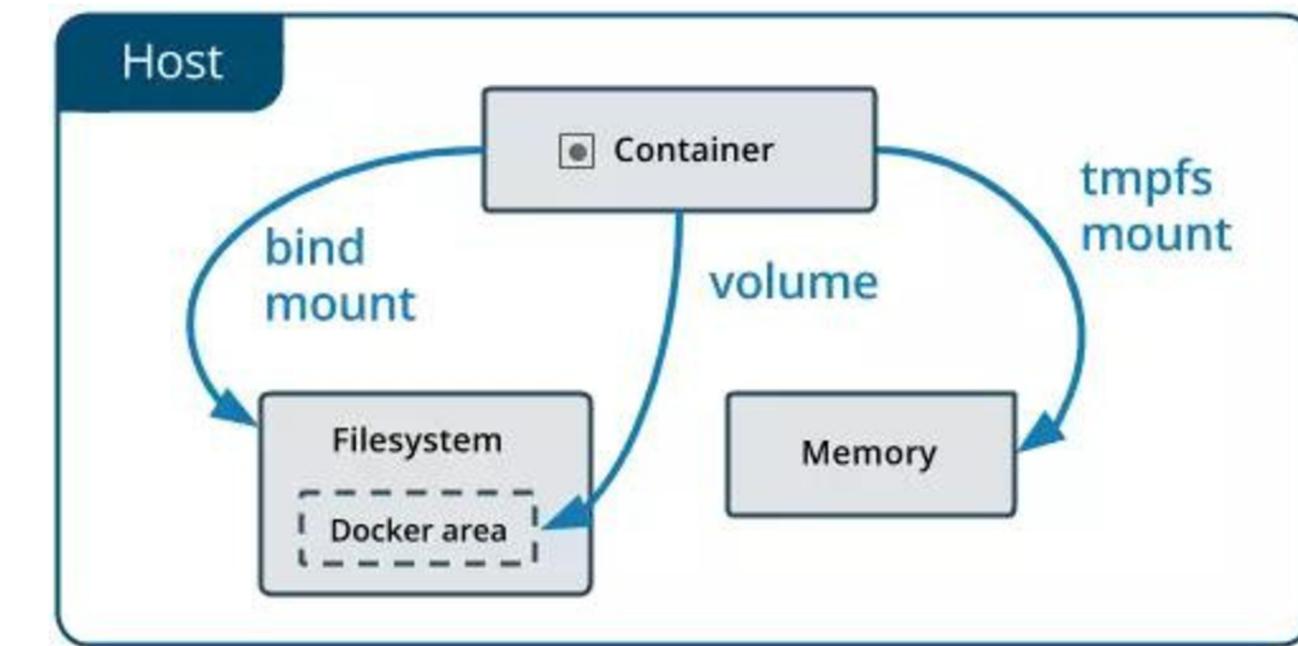
Bao gồm các tùy chọn:

- **Volumes:** Là phương pháp lưu trữ phổ biến nhất, được quản lý bởi Docker và lưu trữ dữ liệu trên hệ thống tệp của host. Volumes được thiết kế để bền vững và hiệu quả, hỗ trợ các container chia sẻ dữ liệu một cách dễ dàng.
- **Bind Mounts:** Gắn một thư mục hoặc tệp trên host vào container. Phương pháp này đơn giản nhưng không được quản lý bởi Docker, dễ gây lỗi nếu không kiểm soát tốt.
- **Tmpfs:** Lưu trữ dữ liệu trong bộ nhớ RAM, rất nhanh nhưng không bền vững, thường dùng cho dữ liệu tạm thời.



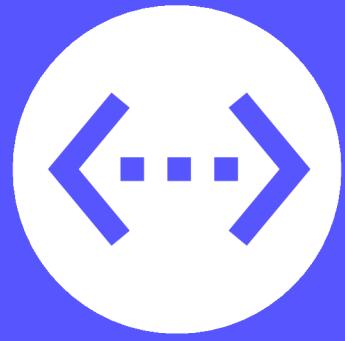
Các kiểu mount

1



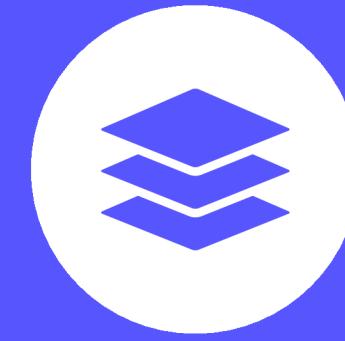
Volume

Mount-point sẽ nằm
/var/lib/docker/volumes/ của Docker
Host và được quản lý bằng Docker.



Bind Mounts

Mount-point có thể nằm ở bất kỳ đâu
Docker Host không được quản lý bởi
Docker.



TMPFS Mounts

Data sẽ được lưu vào memory của
Docker host và sẽ mất đi khi khởi động
lại hoặc stop Container.

Các kiểu mount

1

Volume

Ưu điểm:

- Được Docker quản lý, an toàn và bền vững.
- Dễ backup/restore.
- Hỗ trợ chia sẻ giữa nhiều container.
- Hiệu suất tốt hơn Bind Mount trên các hệ thống tệp phức tạp.

Nhược điểm:

- Không trực quan khi kiểm tra nội dung trực tiếp từ host.

Khi nào dùng:

- Lưu trữ dữ liệu lâu dài như cơ sở dữ liệu, file logs, hoặc cấu hình chia sẻ.

```
docker run -d --name db-container -v db-data:/var/lib/mysql mysql:latest
```

Bind Mounts

Ưu điểm:

- Linh hoạt, dễ gắn tệp hoặc thư mục host vào container.
- Có thể kiểm tra và chỉnh sửa dữ liệu trực tiếp trên host.

Nhược điểm:

- Không được Docker quản lý, có thể dẫn đến lỗi nếu không kiểm soát tốt.
- Không an toàn nếu sử dụng sai cách.

Khi nào dùng:

- Phát triển và kiểm thử khi cần sửa đổi tệp ngay lập tức.

```
docker run -d --name web-container -v /path/to/code:/usr/share/nginx/html nginx:latest
```

TMPFS Mounts

Ưu điểm:

- Tốc độ rất nhanh do lưu trữ trong RAM.
- Dữ liệu tự động xóa khi container dừng.

Nhược điểm:

- Không bền vững, dữ liệu mất khi container dừng.
- Hạn chế bởi dung lượng RAM.

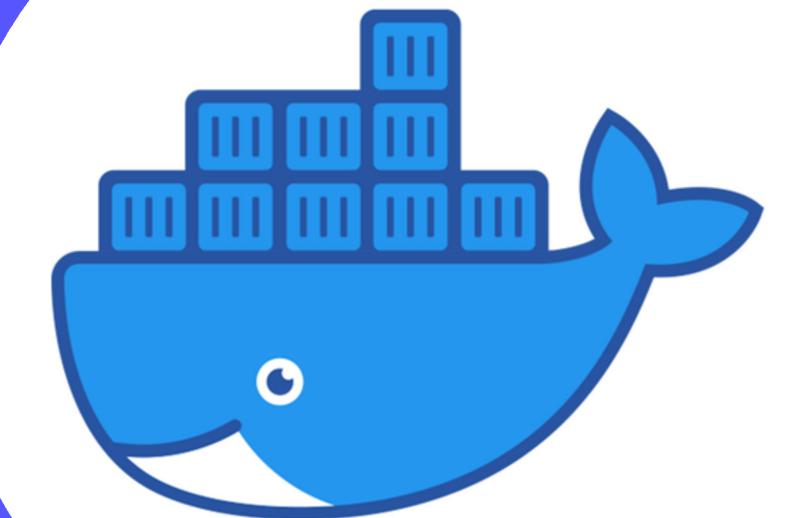
Khi nào dùng:

- Lưu trữ dữ liệu tạm thời như cache, session, hoặc thông tin không cần lưu trữ lâu dài.

```
docker run -d --name cache-container --tmpfs /app/cache:rw,size=64m nginx:latest
```

Docker Logs

2

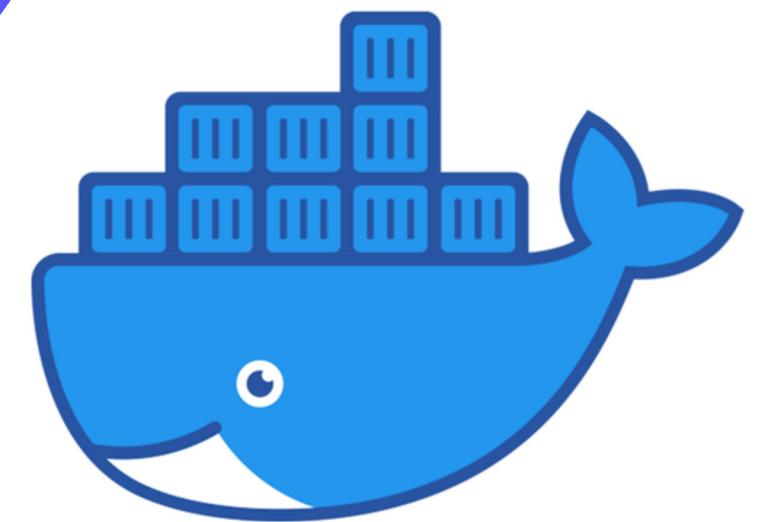


Là gì?

Docker logs là các bản ghi (logs) được tạo ra bởi container hoặc Docker daemon.

Cung cấp thông tin về hoạt động của ứng dụng trong container và trạng thái của Docker daemon

Giúp chúng ta gỡ lỗi và theo dõi hiệu suất hệ thống.



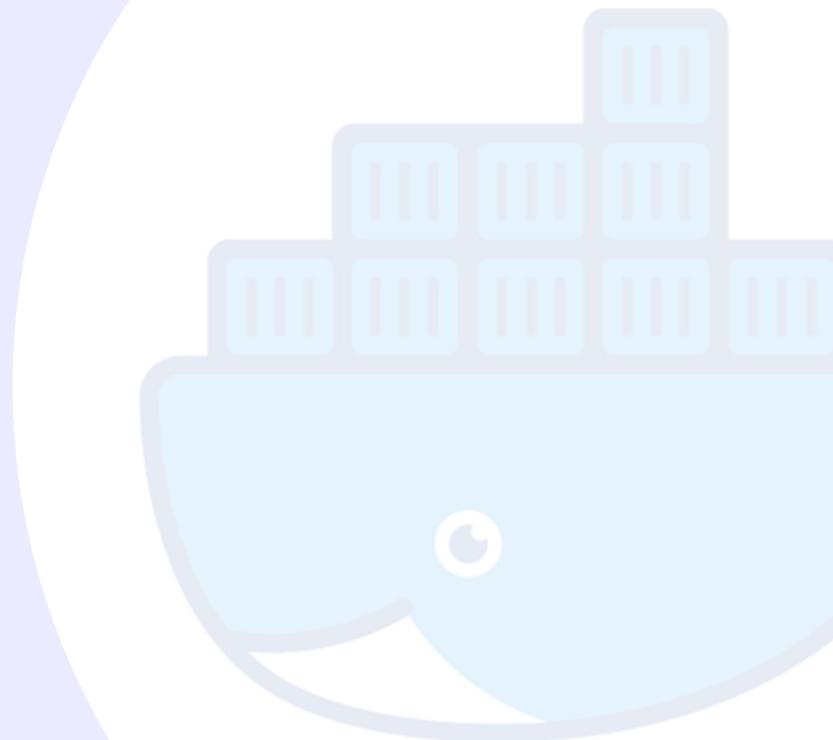
1. Daemon log

Ghi lại trạng thái và hoạt động của chính Docker daemon

ví dụ: khởi động container, lỗi từ các tiến trình nền

2. Container log

Ghi lại mọi thông điệp từ ứng dụng đang chạy trong container.



1. Daemon log

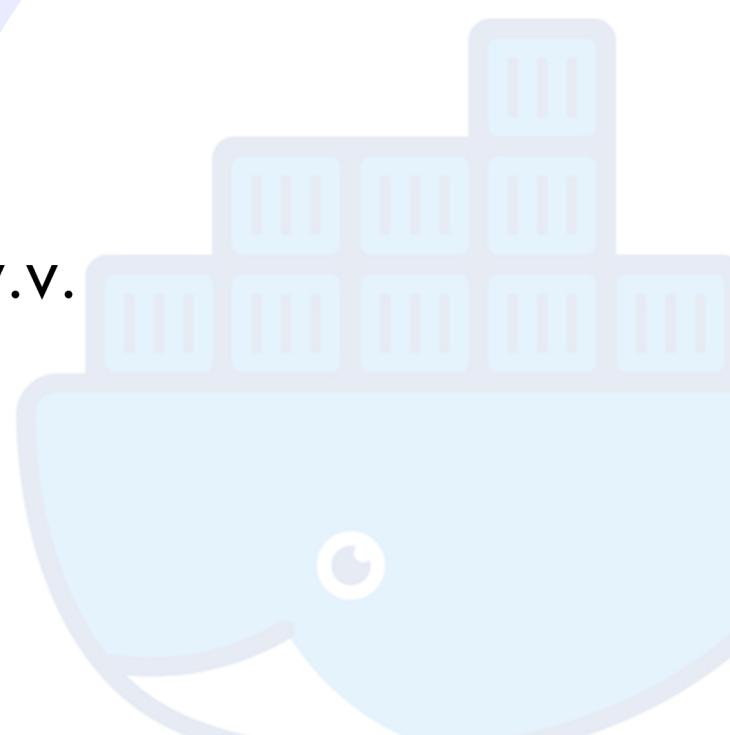
Docker daemon log là các bản ghi (log) được tạo ra bởi Docker daemon – tiến trình chính của Docker (thường là dockerd)

Chịu trách nhiệm quản lý các container, images, network, và volumes.

- Trạng thái của Docker daemon (khởi động, dừng, hoặc lỗi).
- Các sự kiện xảy ra khi Docker daemon thực thi, như khởi tạo container, tải image, lỗi mạng, v.v.
- Cảnh báo hoặc lỗi khi daemon gặp sự cố.

Vị trí lưu log:

- /var/log/docker.log (hoặc /var/log/daemon.log trên một số bản phân phối).
- Sử dụng hệ thống journalctl nếu log được cấu hình với journald.



1. Daemon log

Cách xem log:

- Sử dụng journalctl (khi log driver là journald):

Xem toàn bộ logs:

```
bash  
journalctl -u docker.service
```

Xem logs gần đây:

```
bash  
journalctl -u docker.service --since "1 hour ago"
```

Theo dõi logs theo thời gian thực:

```
bash  
journalctl -u docker.service -f
```

- Xem trực tiếp:

Nếu logs được lưu trong file `/var/log/docker.log`:

```
bash  
cat /var/log/docker.log
```



1. Daemon log

1. Cách cấu hình: file daemon.json

```
{  
  "log-level": "info",  
  "log-driver": "journald"  
}
```

- log-level: Mức độ chi tiết của logs (mặc định là info). Các mức:
 - debug: Chi tiết nhất, dùng khi gỡ lỗi.
 - info: Mức độ thông tin thông thường.
 - warn: Cảnh báo, chỉ log các vấn đề có thể gây lỗi.
 - error: Chỉ log lỗi nghiêm trọng.
- log-driver: Chọn driver cho log (mặc định là json-file hoặc journald).

2. Khởi động lại Docker để áp dụng cấu hình `sudo systemctl restart docker`



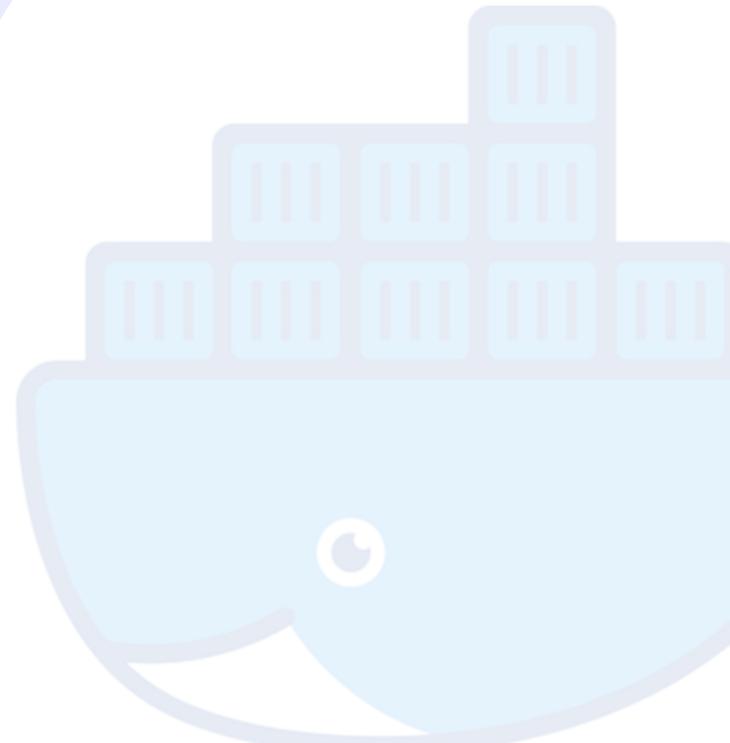
2. Container log

Log container là dữ liệu đầu ra từ container Docker, bao gồm các sự kiện, thông báo lỗi, hoặc dữ liệu ghi nhận hoạt động.

Tại sao cần log container?

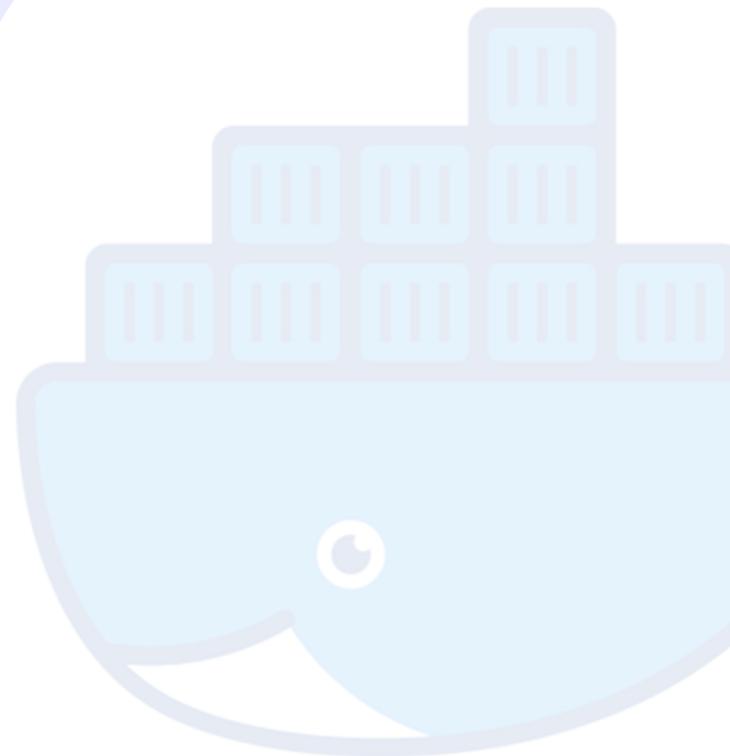
- Giám sát ứng dụng.
- Debug lỗi.
- Phân tích hiệu năng.

Tham khảo thêm: [Link](#)



2. Container log

- Log đầu ra của container:
 - stdout: Log thông tin, hoạt động bình thường.
 - stderr: Log lỗi hoặc thông báo cần chú ý.
- Docker ghi log vào driver json-file theo mặc định:
 - Dữ liệu log được lưu dưới dạng JSON trên host.
 - Vị trí lưu log mặc định trên Linux:
 - `/var/lib/docker/containers/<container-id>/<container-id>-json.log`.



2. Container log

Docker hỗ trợ nhiều driver log, phù hợp với từng mục đích:

Driver	Mô tả
json-file	Driver mặc định, lưu log dưới dạng JSON trên host.
syslog	Gửi log đến syslog daemon trên host hoặc từ xa.
journald	Ghi log vào systemd journal.
fluentd	Gửi log đến Fluentd để phân tích hoặc lưu trữ.
awslogs	Gửi log đến Amazon CloudWatch Logs.
gelf	Gửi log dưới dạng GELF (Graylog Extended Log Format).
none	Tắt log container.



2. Container log

Ví dụ chạy container với driver log mặc định (json-file)

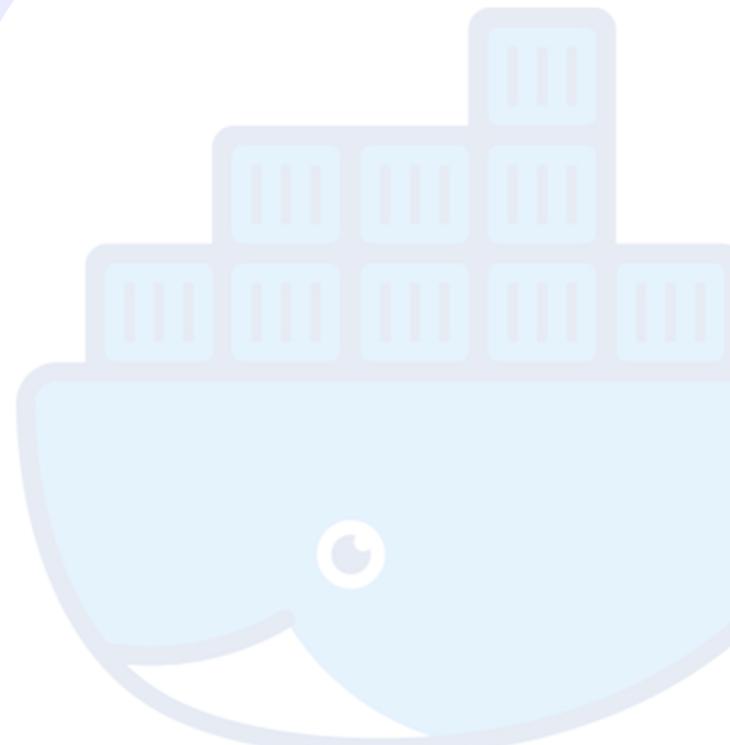
```
#Run docker container  
docker run --name my_nginx -d nginx
```

```
#kiểm tra log  
docker logs my_nginx
```

```
#live log  
docker logs -f my_nginx
```

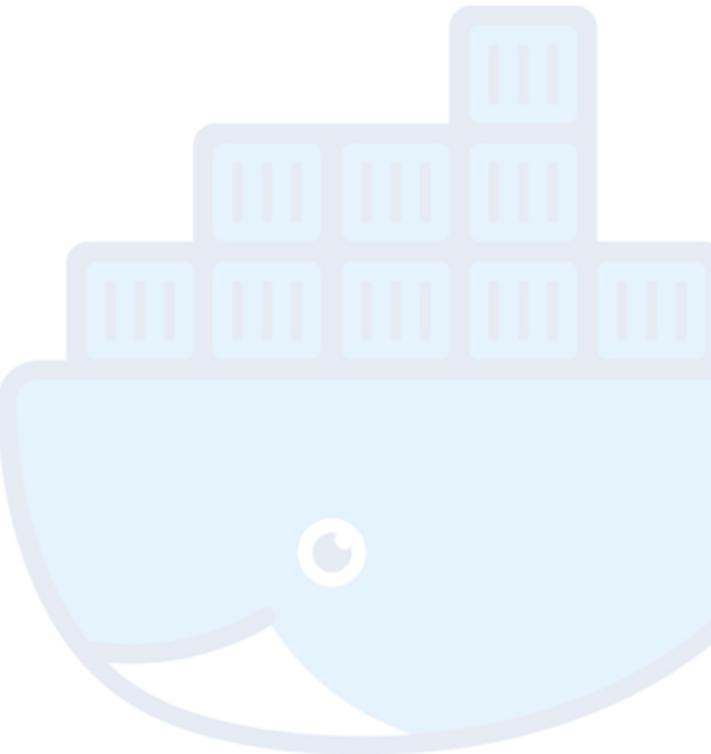
```
#giới hạn số dòng  
docker logs --tail 50 my_nginx
```

```
## Cấu hình limit log  
docker run --name my_nginx_limited -d \  
--log-opt max-size=10m \  
--log-opt max-file=3 \  
nginx
```



Câu hỏi?

Làm thế nào để xem log của nginx container trực tiếp mà vẫn muốn lưu log vào file trên máy host để xem lại khi cần thiết?



Backend Team

THANK
YOU

