Docker container log 관리

^Docker Container의 log를 관리하는 방법을 가이드 합니다.

개요

Docker는 기본적으로 로그파일에 크기 제한을 두지 않고 따로 log lotation 설정도 없기에 시간이 지남에 따라 쌓여서 디스크 공간을 많이 차지할 수 있으므로 관리가 필요합니다.

관리 방법

logrotate를 활용하는 방법

logrotate를 사용하여 docker data directory에 containers에 저장된 로그파일을 직접 관리하는 방법입니다. 세부적으로 원하는 대로 설정 할 수 있는 장점이 있습니다.

아래 코드는 로그 파일의 사이즈가 1G가 이상일 경우 로테이션을 생성하여 압축파일로 저장하고 빈 log파일을 새로 생성합니다.

아래 명령어로 docker root directory 확인

```
docker info | grep "Docker Root Dir"
```

logrotate 설정

```
vi /etc/logrotate.d/docker

/data/cicd/docker/containers/*/*.log {
  rotate 3
  size 1G
  daily
  compress
  missingok
  copytruncate
}
```

그 밖에 고려해봐야 할 옵션 정리

옵션	설명	예시
rotate [숫자]	로그파일 개수가 선택한 숫자 이상이면 오래된 로그 파일 삭제	rotate 3
daily	로테이트 실행 주기 옵션 yearly(연단위), monthly(월단위), weekly(주단위), daily(일 단위)	
size [숫자 K,M,G]	로그 파일이 크기가 설정보다 커지면 로테이트 실행	size 10G
create [권한] [유저] [그룹]	로테이트 될때 생성되는 로그파일 권한 및 소유자 지정	create 644 root root
notifempty	로그 내용이 없으면 로테이트를 실행하지 않음	
ifempty	로그 내용이 없어도 로테이트를 진행	

compress	로테이트로 생성되는 로그파일 gzip으로 압축생성	
nocompress	로테이트로 생성되는 로그파일을 압축하지 않고 생성	
missingok	로그파일을 발견하지 못해도 에러처리 하지 않음	
dateext	로테이트 파일의 이름에 날짜가 들어가도록 생성	
copytruncate	복사본 저장 후 원본 로그파일을 빈 파일로 생성	

logrotate를 적용하고 하루를 기다리지 않고 아래 동작들로 미리 확인을 해볼 수 있습니다.

logrotate를 미리 실행해 보기(dry-run)

실제 실행은 되지 않고 시뮬레이션만 수행하여 결과 출력

```
/usr/sbin/logrotate -vd /etc/logrotate.d/docker
```

logrotate를 강제로 실행하기

로테이션 조건을 만족하지 않아도 강제로 실행

```
/usr/sbin/logrotate -vf /etc/logrotate.d/docker
```

logrotate를 바로 실행하기

로그로테이션 조건을 만족하면 실행됨

```
/usr/sbin/logrotate -v /etc/logrotate.d/docker
```

logrotate 실행 시간 변경하기

daily조건이 설정된 경우 24시간 이전에 실행해도 이미 로테이션 되었다고 수행되지 않는다. 그럴 때 아래처럼 status 파일에서 날짜를 수정해주면 로테이션을 수행시킬 수 있습니다.

```
vi /var/lib/logrotate/logrotate.status

"/rep/cicd/docker/containers/5639y498yf983j98fur40/5639y498yf983j98fur40-json.log" 2023-8-8-1:28:3 //
```

Docker Compose yml파일에 설정하는 방법

Docker Compose로 컨테이너를 구성하였을 경우 docker-compose.yml파일에 아래와 같이 **logging** 설정하여 관리가 가능합니다.

위에 logrotate처럼 세세한 설정은 불가능하지만 컨테이너별로 설정을 할 수 있습니다.

먼저 docker logging driver 설정

```
sudo vi /etc/docker/daemon.json

{
  "log-driver": "json-file"
}
```

그 다음 docker-compose.yml 파일에 logging 영역 추가

```
  gitlab:
    container_name: gitlab
    image: 'gitlab/gitlab-ee:latest'
    restart: always
    hostname: 'gitlab.example.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'https://gitlab.example.com' # gitlab
        # Add any other gitlab.rb configuration here, each on its own line
    ports:
      - "80:80"
      - "443:443"
      - "8022:22"
    volumes:
      # export GITLAB_HOME=/srv/gitlab      OR $GITLAB_HOME
      - './config:/etc/gitlab'
      - './logs:/var/log/gitlab'
      - './data:/var/opt/gitlab'
    shm_size: '256m'
        logging:
      driver: 'json-file'
      options:
        max-size: "10m"
        max-file: "10"
```

Docker log lotation 설정하는 방법

각 Docker Daemon에는 기본 Logging Driver가 있으며, 이 드라이버는 다른 Logging Driver를 사용하도록 구성하지 않는 한 각 Container에서 사용합니다.
기본적으로 Docker는 Container log를 내부적으로 JSON으로 캐시하는 json 파일 Logging Driver를 사용합니다. 아래 가이드에서는 두 가지 옵션을 제공합니다.

간단하게 설정 가능하지만 세부적인 설정도 불가하고 모든 Container에 적용됩니다.

Local Logging Driver 사용

local Logging Driver는 기본적으로 로그 로테이션 기능을 지원하기 때문에 디스크 용량 관리하는 방법으로 유용합니다. 또한 log-opts를 활용하여 로테이션 조정도 가능합니다.

```
//deamon.json
sudo vi /etc/docker/daemon.json
Windows: C:\ProgramData\docker\config\daemon.json

{
  "log-driver": "local",
  "log-opts": {
    "max-size": "15m"
    "max-file": "5"
  }
}
```

json-file Logging Driver 사용 (labels, env 생략가능)

```
//deamon.json
sudo vi /etc/docker/daemon.json
Windows: C:\ProgramData\docker\config\daemon.json

{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "10",
    "labels": "production_status",
    "env": "os,customer"
  }
}

//docker
sudo systemctl restart docker
```

docker daemon.json 편집 이후 생성되는 Container에 대해서만 해당 logging 설정이 적용됩니다. 변경전에 생성된 Container에는 적용되지 않으므로 새로 재생성해야 합니다.

참조링크

- Configure logging drivers | Docker Documentation