도커 이미지 경량화 및 빌드 최적화 기법

Docker & CI/CD 스터디 특강 (2차)

튜터 송영훈

01

도커 이미지에 대한 이해 높이기

빌드 최적화

이미지 경량화

Dockerfile

01 도커 이미지에 대한 이해 높이기

도커 이미지를 최적화 했을 때 얻을 수 있는 이점

- CI/CD 시간 단축
- 저장소 유지 비용 절감
- 컨테이너가 조금 더 빠르게 동작 할 수 있는 가능성 상승

01 도커 이미지에 대한 이해 높이기

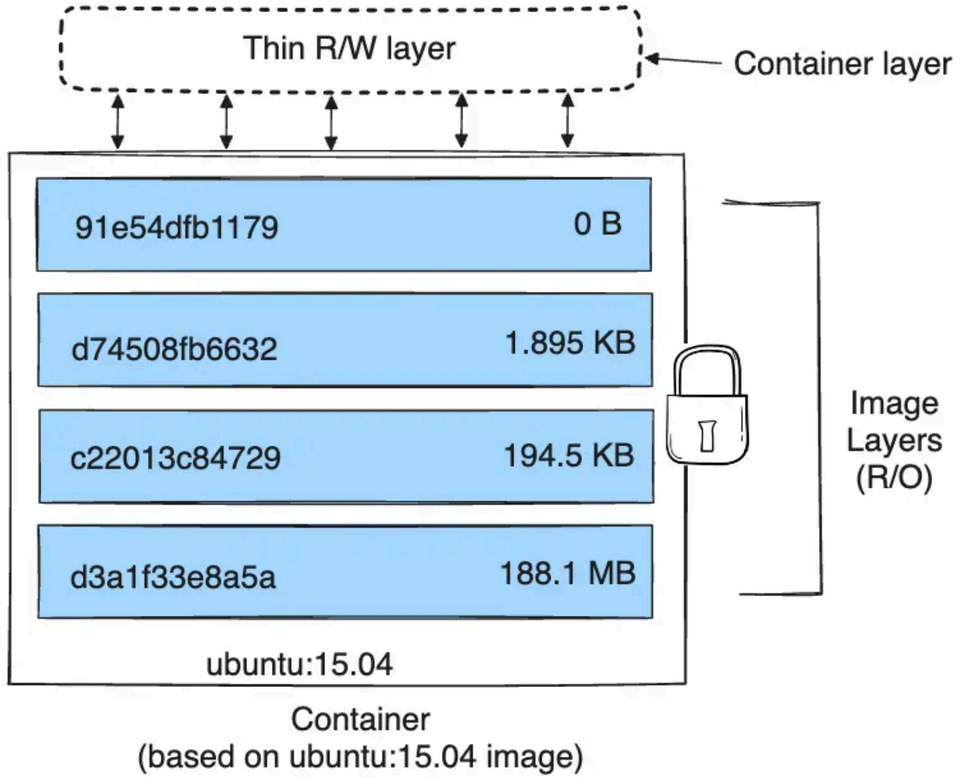
```dockerfile
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

CMD ["python", "app.py"]
```

01

도커 이미지에 대한 이해 높이기

컨테이너 이미지 레이어란?

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
EXPOSE 8080
CMD /app/launch
```



01 도커 이미지에 대한 이해 높이기

왜 레이어 개념이 필요한가요?

- 이미지의 재사용
- 빌드 시간 단축
- 효율적인 배포 (이미지 경량화에 도움)

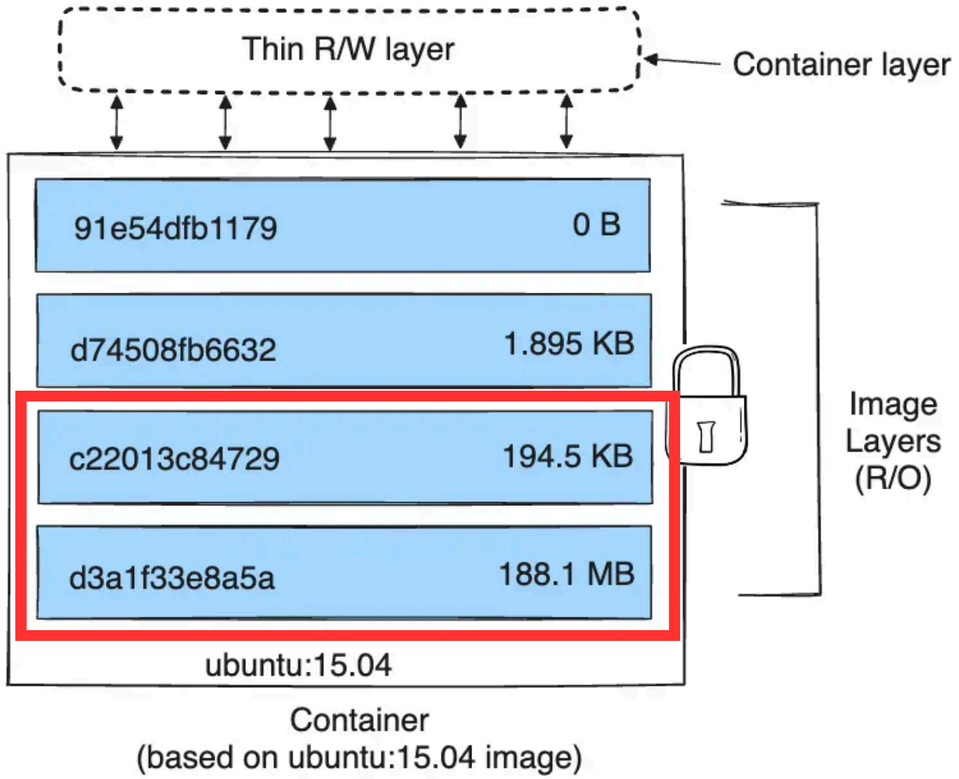
02

도커 이미지 빌드 캐시에 대해 이해하기

캐싱(CACHING)이란?

자주 사용하는 데이터나 결과를 임시 저장하여,
필요할 때 빠르게 다시 접근할 수 있도록하는 기법.

02 도커 이미지 빌드 캐시에 대해 이해하기



02 도커 이미지 빌드 캐시에 대해 이해하기

새로운 레이어가 추가되는 명령어

FROM	COPY	RUN
ADD	ENV	WORKDIR
EXPOSE / LABEL	USER	VOLUME

02

도커 이미지 빌드 캐시에 대해 이해하기

캐시를 활용하지 못하는 사례

```
FROM ubuntu:15.04
ENV BUILD_MODE=BLUE
COPY . /app
RUN make /app
CMD /app/launch
```

```
FROM ubuntu:15.04
ENV BUILD_MODE=RED
COPY . /app
RUN make /app
CMD /app/launch
```

02 도커 이미지 빌드 캐시에 대해 이해하기

일반적으로 생각 할 수 있는 원칙

1. ENV는 사용되기 직전 타이밍에 입력한다.

```
FROM ubuntu:15.04
ENV BUILD_MODE=RED
COPY . /app
RUN make /app
CMD /app/launch
```

→

```
FROM ubuntu:15.04
COPY . /app
ENV BUILD_MODE=RED
RUN make /app
CMD /app/launch
```

2. 의존 패키지는 미리 따로 설치한다.

```
FROM python:3.12
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

→

```
FROM python:3.12
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

3. 가능하면 ADD보다 COPY를 쓰자.

03 도커 이미지 경량화

도커 이미지를 경량화 하기 위해 고려해야 할 것들

1. 베이스 이미지를 잘 선택하기

2. 불필요한 파일을 이미지에 포함 시키지 않기

03 도커 이미지 경량화

경량화 이미지 선택

python:3.12 MULTI-PLATFORM

LANGUAGES & FRAMEWORKS

INDEX DIGEST sha256:b52e97a85736bc3a6bfb081ec3f2a5b828233120b2138080af1761e504b06529

OS/ARCH

linux/amd64

COMPRESSED SIZE	LAST PUSHED	TYPE	VULNERABILITIES	MANIFEST DIGEST
363.36 MB	3 days ago by doijanky	Image	0 1 2 113 2	sha256:1b7f52f4a...

python:3.12-slim MULTI-PLATFORM

LANGUAGES & FRAMEWORKS

INDEX DIGEST sha256:032c52613401895aa3d418a4c563d2d05f993bc3ecc065c8f4e2280978acd249

OS/ARCH

linux/amd64

COMPRESSED SIZE	LAST PUSHED	TYPE	VULNERABILITIES	MANIFEST DIGEST
44.12 MB	4 days ago by doijanky	Image	0 0 0 26 0	sha256:311c6c70...

03

도커 이미지 경량화

DOCKERIGNORE 파일을 사용하여 불필요한 파일을 포함 시키지 않기

```
# 파일명: dockerignore

__pycache__/
*.py[cod]
*.pyo
*.pyd

venv/
env/

tests/
.coverage
htmlcov/
```

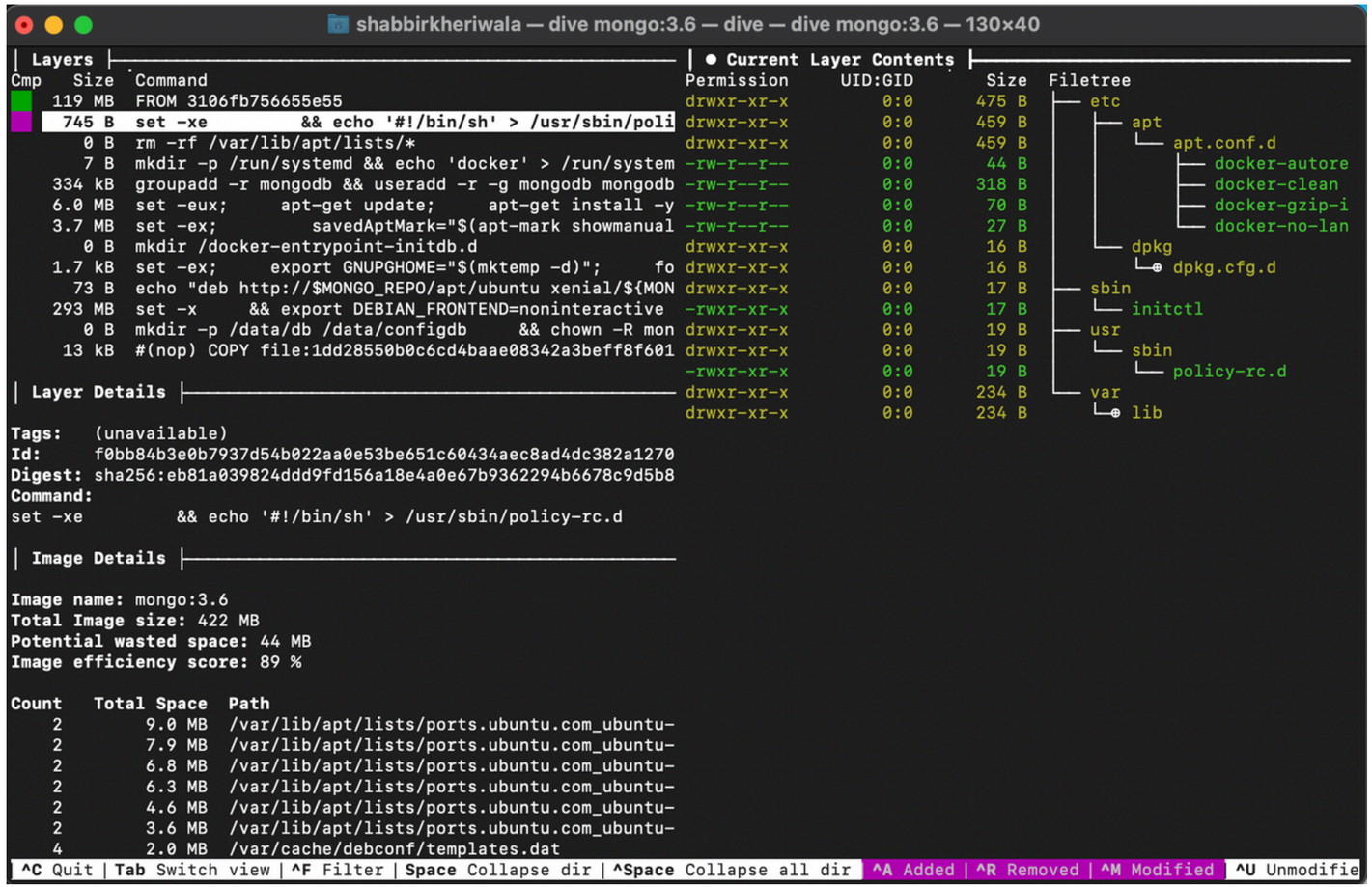
03 도커 이미지 경량화

멀티 스테이지 빌드 활용

```
# 빌드 스테이지
FROM maven:3.8.6-eclipse-temurin-17 AS builder
WORKDIR /app
COPY pom.xml .
RUN mvn dependency:go-offline
COPY src ./src
RUN mvn package -DskipTests

# 실행 스테이지
FROM eclipse-temurin:17-jre-alpine
WORKDIR /app
COPY --from=builder /app/target/*.jar app.jar

CMD ["java", "-jar", "/app/app.jar"]
```

04 도커 이미지 분석 도구: dive

https://github.com/wagoodman/dive



Github에서 Gitlab처럼 CI/CD 파이프라인을 구축하는 방법

https://docs.github.com/ko/actions

```
name: Build and Deploy to ECS

on:
  push:
    branches:
      - main

jobs:
  build_and_push_to_ecr:
    name: Build Docker image and push to ECR
    runs-on: ubuntu-latest

    steps:
      # Checkout the repository
      - name: Checkout code
        uses: actions/checkout@v3

      # Set up Python environment
      - name: Set up Python 3.9
        uses: actions/setup-python@v4
        with:
          python-version: '3.9'

      # Install Python dependencies (if any)
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi

      # Set up AWS credentials
      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v2
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: <your-aws-region>

      # Log in to Amazon ECR
      - name: Log in to Amazon ECR
        run: |
          aws ecr get-login-password --region <your-aws-region> | docker login --username AWS --password-stdin <aws-account-id>.dkr.ecr.<your-aws-region>.amazonaws.com

      # Build the Docker image using make build
      - name: Build Docker image
        run: make build

      # Tag and push the Docker image to ECR
      - name: Tag and Push Docker image to ECR
        run: |
          IMAGE_TAG=${{ github.sha }}
          docker tag <your-docker-image>:latest <aws-account-id>.dkr.ecr.<your-aws-region>.amazonaws.com/<your-ecr-repository>:${IMAGE_TAG}
          docker push <aws-account-id>.dkr.ecr.<your-aws-region>.amazonaws.com/<your-ecr-repository>:${IMAGE_TAG}

      # Deploy the image (optional step depending on your setup)
      # This step assumes that you have some deployment mechanism such as ECS, EKS, or Lambda
      - name: Deploy to ECS
        run: |
          aws ecs update-service --cluster <your-ecs-cluster> --service <your-ecs-service> --force-new-deployment
```

05

Q&A

Q&A