Bootcamp DevOps Entry Level
DLA-SS001
Week 2

Tugas Minggu 2
CI/CD + Docker
(Tanpa Docker Compose)

DEBUG LIFE ACADEMIA

> TO BOLDLY GO BEYOND THE CODE

by:
Kresnayana Nanda A.

Daftar Isi

1. Push Docker Image ke DockerHub

a. Siapkan Dockerfile

```
# === Stage 1: Build dependencies ===
FROM python:3.11-alpine AS builder

RUN apk add --no-cache build-base python3-dev

WORKDIR /build

COPY ../requirements.txt .

RUN pip install --no-cache-dir --prefix=/install -r
requirements.txt

# === Stage 2: Final image ===
FROM python:3.11-alpine

WORKDIR /app

COPY ../src .
COPY --from=builder /install /usr/local

EXPOSE 5000

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

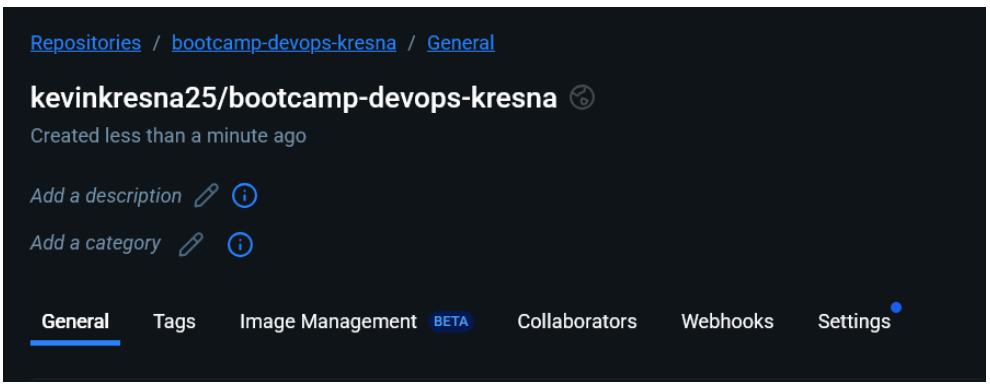
b. Membuat Script Start

Membuat script “start.sh” agar lebih mudah untuk melakukan setupnya.

```
docker build -f docker/Dockerfile -t bootcamp-devops-kresna:v2 .
```

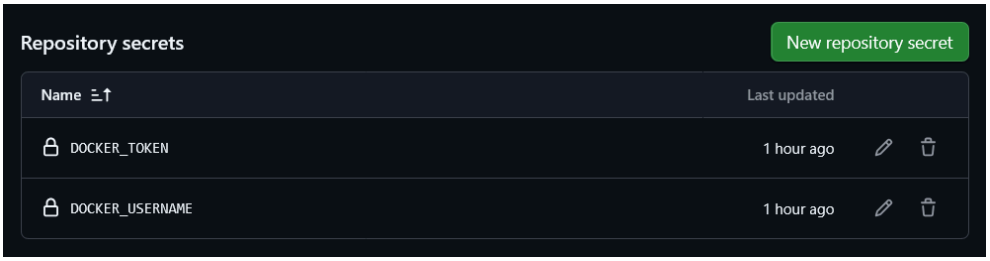
c. Buat Repository di DockerHub

Create Repository dengan nama repo bootcamp-devops-kresna dengan Visibility Public.



d. Buat Secrets di GitHub Repository

Masuk ke repo GitHub dan buka:
Settings → Secrets and variables → Actions → New repository secret
Tambahkan **DOCKER_USERNAME** dan **DOCKER_TOKEN** input sesuai kredensial DockerHub.



e. Membuat Workflow CI/CD

Buat Workflow YAML di ".github/workflows/dockerhub.yml"

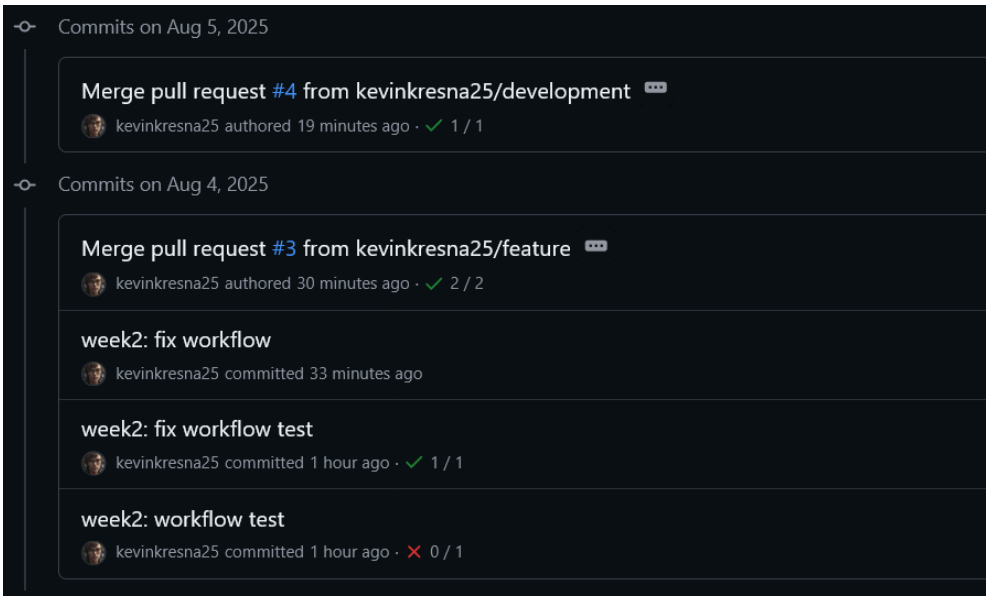
```yaml
name: CI/CD Build and Push

on:
  push:
    branches:
      - development
      - main
  pull_request:
    branches:
      - main

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to DockerHub (only on main branch)
        if: github.ref == 'refs/heads/main'
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_TOKEN }}

      - name: Build & (Push if on main branch)
        uses: docker/build-push-action@v5
        with:
          context: .
```

```
          file: docker/Dockerfile
          push: ${{ github.ref == 'refs/heads/main' }}
          tags: ${{ secrets.DOCKER_USERNAME
}}/bootcamp-devops-kresna:v2
          platforms: linux/amd64
```

f. Push ke GitHub

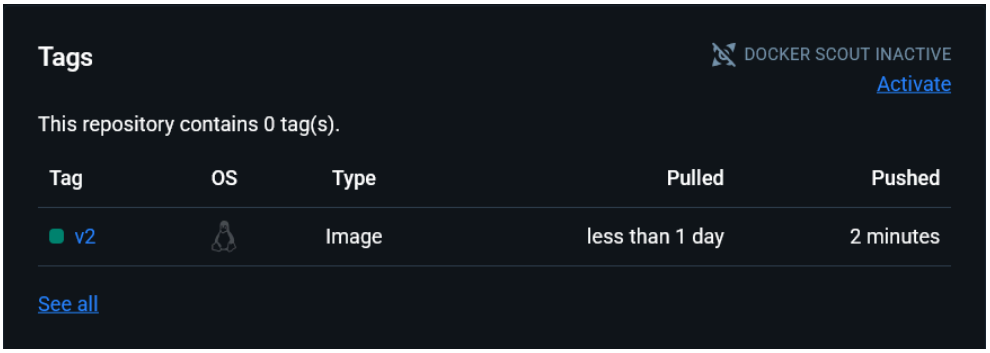
Setelah melakukan commit di branch “feature”, lakukan push ke GitHub dan buat pull request (PR) ke branch “development”. Lakukan review dan perbaikan jika masih ada bug atau kesalahan. Jika program sudah benar-benar fix, commit dan push perbaikannya ke GitHub. Selanjutnya, buat PR dari branch “development” ke “main”.

Ketika PR ke “main” dibuat, GitHub Actions akan berjalan untuk melakukan build image. Jika proses build berhasil, lakukan *merge* PR tersebut. Setelah PR berhasil digabung ke branch “main”, workflow GitHub Actions akan berjalan kembali untuk melakukan push image ke DockerHub.



g. Lakukan Verifikasi ke DockerHub

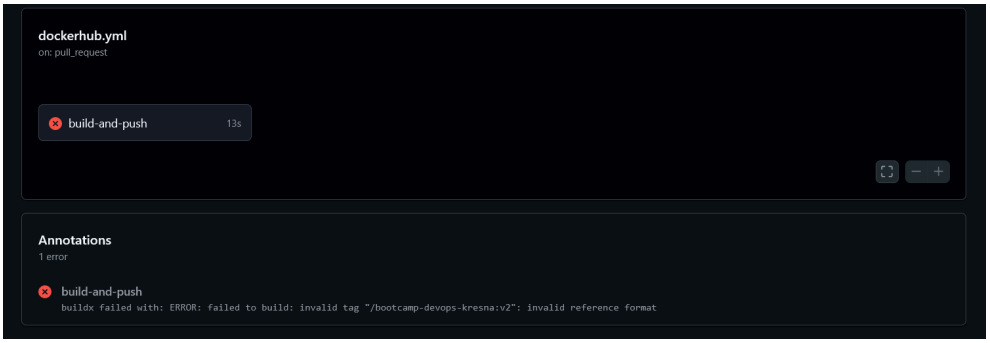
Pada DockerHub harus ada image bootcamp-devops-kresna:v2



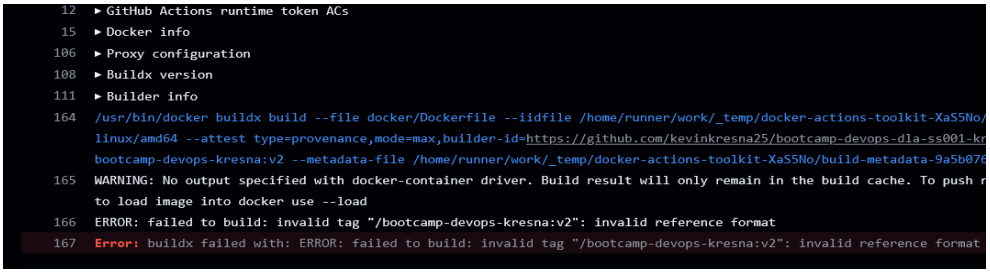
2. GitHub Actions untuk PR

a. Workflow GitHub Actions yang gagal berjalan saat ada PR

Ketika PR dan trigger dari GitHub Actions berjalan lalu gagal, tampilannya akan seperti berikut.

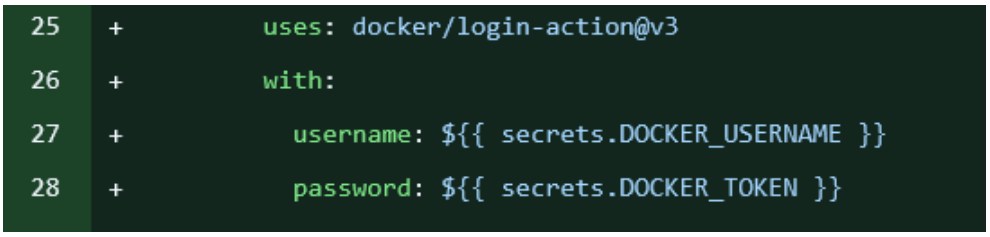


ketika ditelusuri ternyata letak kesalahannya pada nama variabel secret yang sudah diset di awal tadi.

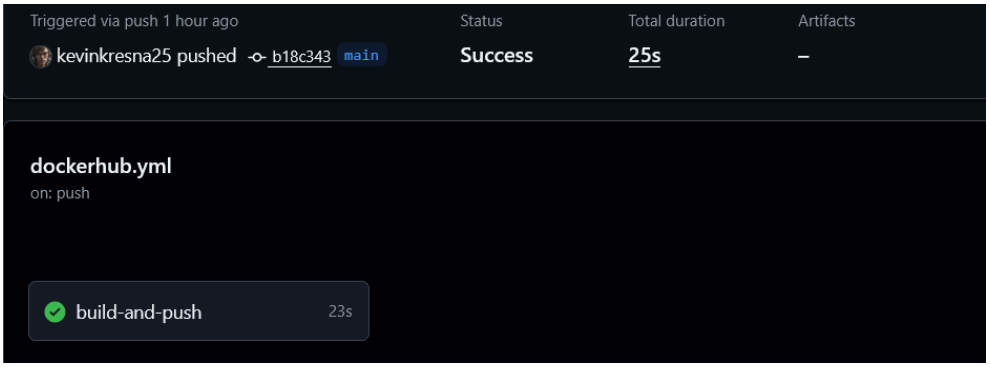


b. Rekomendasi Mitigasi

Ubah nama variabel secret sesuai yang sudah didefinisikan dari awal agar Workflow GitHub Actions menjadi sukses. berikut adalah nama variabel secret yang benar.



c. Workflow GitHub Actions yang sukses berjalan saat ada PR



3. Multi-Stage Docker Build dan Alpine Docker Image

a. Dockerfile Backend

```
# === Stage 1: Build dependencies ===
FROM python:3.11-alpine AS builder

RUN apk add --no-cache build-base python3-dev

WORKDIR /build

COPY requirements.txt .

RUN pip install --no-cache-dir --prefix=/install -r requirements.txt

# === Stage 2: Final image ===
FROM python:3.11-alpine

WORKDIR /app

COPY app.py .
COPY --from=builder /install /usr/local

EXPOSE 5000

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

b. Dockerfile Frontend

```
# === Stage 1: Build dependencies ===
FROM python:3.11-alpine AS builder

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir --prefix=/install -r requirements.txt

# === Stage 2: Final image ===
FROM python:3.11-alpine

WORKDIR /app

COPY . .

COPY --from=builder /install /usr/local

EXPOSE 3000

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

4. Simulasi Microservices: FE & BE

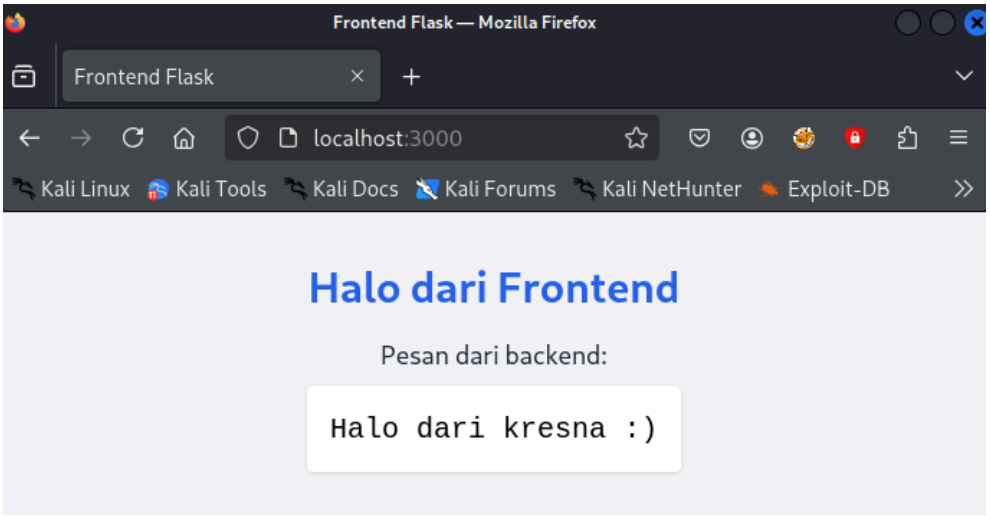
a. Script Menjalankan Microservices

Agar backend dan frontend dapat saling berkomunikasi melalui jaringan internal Docker, terlebih dahulu perlu dibuat sebuah Docker network. Kedua container **frontend** dan **backend** harus dijalankan **di network yang sama**. Pada sisi frontend, untuk mengakses backend cukup menggunakan nama container backend sebagai hostname, diikuti oleh port yang digunakan oleh backend. Setelah itu, lakukan build image untuk masing-masing service backend dan frontend, lalu jalankan container-nya.

```
#!/bin/bash
set -e

# === Clean up containers and network ===
docker rm -f frontend backend 2>/dev/null || true
docker network rm kresna-net 2>/dev/null || true

# === Create new network ===
docker network create kresna-net

# === Build backend image ===
echo "🚧 Building backend image..."
docker build --rm -t bootcamp-devops-kresna-backend:v2 \
  -f project/backend/docker/Dockerfile project/backend

# === Build frontend image ===
echo "🚧 Building frontend image..."
docker build --rm -t bootcamp-devops-kresna-frontend:v2 \
  -f project/frontend/docker/Dockerfile project/frontend

# === Run backend container ===
echo "🚀 Running backend container..."
docker run -d \
  --name backend \
  --network kresna-net \
  -e FLASK_DEBUG=false \
  bootcamp-devops-kresna-backend:v2

# === Run frontend container ===
echo "🚀 Running frontend container..."
docker run -d \
  --name frontend \
  --network kresna-net \
  -p 3000:3000 \
  -e BACKEND_URL=http://backend:5000/hello \
  bootcamp-devops-kresna-frontend:v2

echo "✅ Microservices Running Successfully"
```

b. Microservices Berhasil Berjalan

Untuk melihat apakah Microservices berhasil dijalankan, coba cek pada localhost di port 3000. Berikut adalah tampilan frontend yang berhasil terhubung dengan backend menggunakan docker internal network.



c. Script Menghentikan Microservices

Script ini akan secara otomatis **menghentikan dan menghapus** container **frontend** dan **backend** jika keduanya masih berjalan, kemudian menghapus jaringan internal Docker **kresna-net** yang sebelumnya digunakan sebagai media komunikasi antar container. Setelah itu, script juga menghapus image Docker frontend dan backend agar image lama tidak digunakan kembali saat proses build berikutnya. Dengan langkah-langkah ini, seluruh lingkungan Docker yang digunakan untuk menjalankan aplikasi backend dan frontend benar-benar dibersihkan.

```
#!/bin/bash
set -e

echo "🛑 Stopping containers..."
docker rm -f frontend backend 2>/dev/null || echo "Containers
already removed."

echo "🧹 Removing Docker network..."
docker network rm kresna-net 2>/dev/null || echo "Network already
removed."

echo "🧽 Removing Docker images..."
docker rmi bootcamp-devops-kresna-frontend:v2
bootcamp-devops-kresna-backend:v2 2>/dev/null || echo "Images
already removed."

echo "✅ All stopped and cleaned up."
```

5. Verifikasi Isi Docker Image

a. Cek Size Docker Image

Untuk melihat ukuran docker image bisa menggunakan perintah “docker images” atau “docker image ls”. Berikut adalah size docker image backend dan frontend. Backend: **59.1MB** dan Frontend: **62.1MB**

```
┌──(root㉿kaliVM)-[/media/sf_Kali/dla/bootcamp-devops-dla-ss001-kresna]
└─# docker images
REPOSITORY                          TAG       IMAGE ID       CREATED              SIZE
bootcamp-devops-kresna-frontend     v2        fd64aee98705   28 minutes ago       62.1MB
bootcamp-devops-kresna-backend      v2        a6d2b17c5099   About an hour ago    59.1MB
```

b. Isi Docker Container

Berikut adalah file dan folder dari **Backend**.

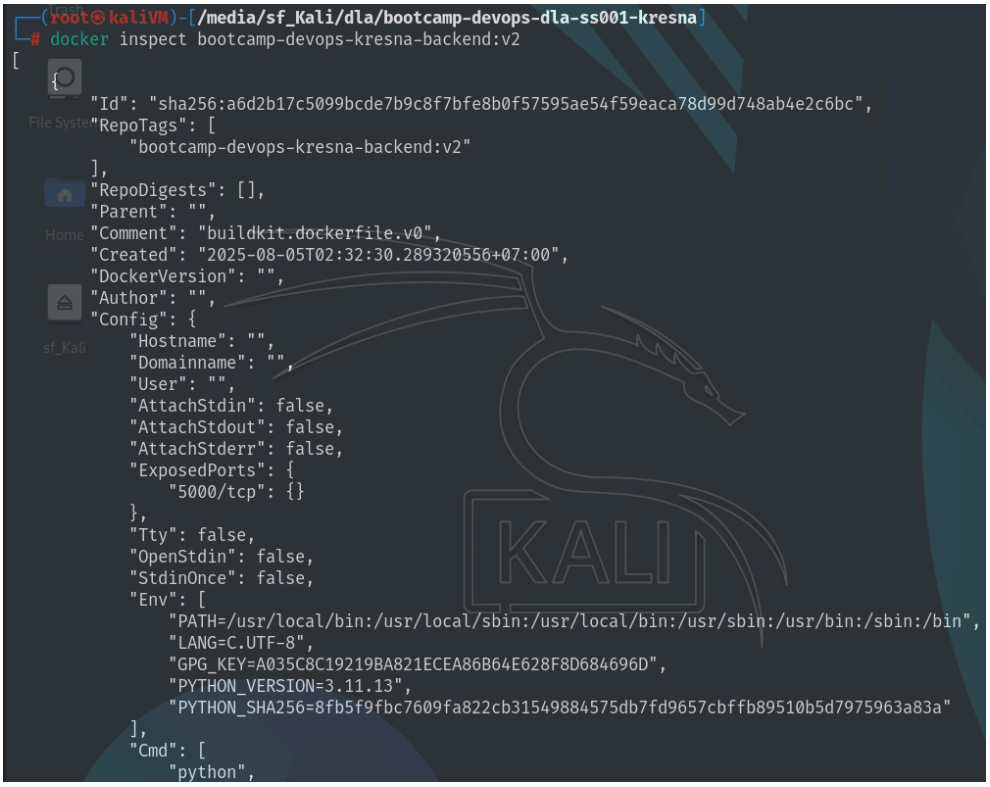
```
└─# docker exec -it backend sh
/app # tree
.
└── app.py
```

Berikut adalah file dan folder dari **Frontend**.

```
└─# docker exec -it frontend sh
/app # tree
.
├── app.py
├── docker
│   └── Dockerfile
├── requirements.txt
├── static
│   └── style.css
└── templates
    └── index.html
```

c. Inspect Docker Image

Berikut adalah hasil inspect image dari **Backend**.



Berikut adalah hasil inspect image dari **Frontend**.