Muhammad Iqbal Ramadhan
5025221274

Cloud Computing
EF4704

Tugas 3

Repository: github.com/LaptopMerah

Pendahuluan

Tugas ini membahas implementasi pembuatan Docker image custom melalui studi kasus yang telah disediakan pada Case 1 hingga Case 4, serta pengembangan kreasi mandiri pada Case 5. Masing-masing case mendemonstrasikan penggunaan teknologi containerization dengan stack yang berbeda, mulai dari web server Apache dan Nginx, PHP-FPM, hingga desktop Linux yang dapat diakses via browser menggunakan noVNC. Melalui analisis keempat case tersebut, mahasiswa dapat memahami pola-pola umum dalam pembuatan Dockerfile seperti pemilihan base image, instalasi dependencies, konfigurasi service, dan pengelolaan proses menggunakan Supervisor.

Tujuan dari tugas ini adalah untuk mempraktikkan pembuatan Docker image secara mandiri dengan menerapkan konsep-konsep yang telah dipelajari dari keempat case sebelumnya. Mahasiswa diharapkan mampu menganalisis struktur Dockerfile, memahami fungsi setiap instruksi yang digunakan, serta mengembangkan kreasi container sendiri yang menggabungkan berbagai konsep seperti port expose, volume mount untuk persistent data, dan konfigurasi service yang berjalan di dalam container. Pada Case 5, kreasi yang dikembangkan adalah container Jupyter Notebook yang menyediakan environment interaktif untuk pemrograman Python dan dapat diakses langsung melalui browser.

Tugas yang Diberikan

- Menjalankan 4 Case yang disediakan pada repository github. Setiap case, diperlukan dokumentasi yang mencakup cara menjalankan, screenshot hasil, dan penjelasan proses.
- Membuat Case baru (case 5) berbasis pada case sebelumnya. Proyek ini harus dilengkapi dengan:
    - Penjelasan skenario atau kondisi di mana kreasi tersebut penting atau cocok untuk digunakan.
    - Gambar arsitektur sistem yang dibuat.
    - Script pendukung dan screenshot hasil serta penjelasannya.

Case 1:

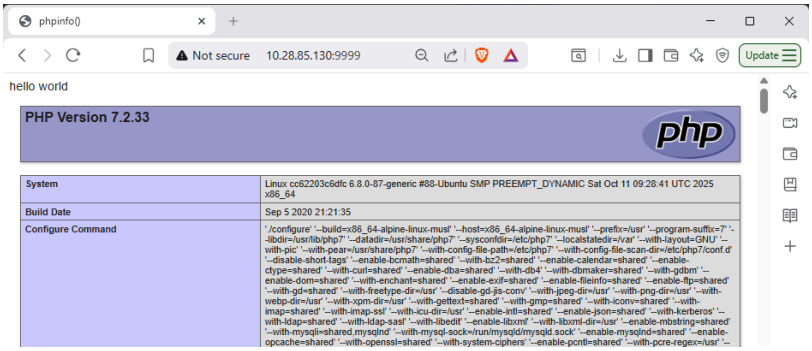
Pada case1 akan membuat image **mywebserver:1.0** menggunakan Dockerfile. Image ini adalah aplikasi PHP yang menampilkan tulisan “Hello World” dan detail dependensi yang digunakan.

```
LaptopMerah@LaptopMerah:case1
> cat platform/Dockerfile
FROM alpine:3.9

RUN apk update && apk add --no-cache apache2 curl supervisor socat php7-intl php7-openssl php7-pecl-redis php7-sqlite3 php7-p
do_mysql php7-mbstring apache2-ssl php7-apache2 php7-intl php7-openssl php7-sqlite3 php7-pdo_mysql php7-mbstring apache2-ssl
php7-apache2 apache2 apache2-ctl php7-pecl-ssh2 php7-bcmath php7-curl php7-json php7-pecl-couchbase php7-zip php7-mysqli mysq
l-client php7-pecl-mcrypt php7-pecl-redis supervisor
RUN apk add composer php7-gmp php7-sodium  php7-xml supervisor
RUN rm -rf /tmp/* /var/cache/apk/*

ADD start.sh /tmp
ADD supervisord.conf /tmp/supervisord.conf

EXPOSE 80
EXPOSE 9999

ENV SESSIONAGE 60
ENV REAUTHENTICATION 60
ENV IPEXPIRE 120
VOLUME ["/var/www/localhost/htdocs"]

ENTRYPOINT ["sh","-C","/tmp/start.sh"]
```

Dockerfile pada folder /platform ini membangun lingkungan server web berbasis Alpine Linux 3.9 yang menjalankan Apache2 dan PHP 7 dengan dukungan ekstensi lengkap (termasuk MySQL, Redis, dan enkripsi). Konfigurasi ini mengintegrasikan supervisor untuk manajemen proses, menyertakan composer untuk manajemen dependensi, serta menyiapkan skrip inisialisasi sistem. Image ini mengekspos port 80 dan 9999, menetapkan variabel lingkungan sesi, dan menyediakan volume persisten pada direktori root

```
LaptopMerah@LaptopMerah:case1
> cat runcontainer/run-mywebserver.sh
docker rm -f mywebserver

docker run \
        -dit \
        --name mywebserver \
        -p 9999:80 \
        -v "$(pwd)/html:/var/www/localhost/htdocs/" \
        mywebserver:1.0
```

run-mywebserver.sh ini berfungsi untuk menjalankan container menggunakan image **mywebserver** yang dibangun dari Dockerfile. container ini memetakan port 9999 di host ke port 80 di container dan melakukan mounting direktori host html ke volume container.



Dengan menjalankan **build.sh** di direktori /platform untuk membangun Docker Image, lalu jalankan **run-mywebserver.sh** yang berada di direktori /runcontainer, maka aplikasi PHP dapat diakses melalui http://10.28.85.130:9999 yang menampilkan tulisan "hello world" diikuti dengan halaman informasi detail mengenai versi dan konfigurasi PHP yang digunakan.

Case 2:

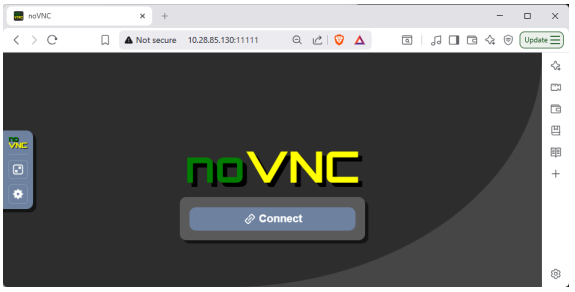
Pada case2 akan membuat image bernama **mylinux:1.0**. Aplikasi Linux akan berjalan pada browser menggunakan VNC.

```
LaptopMerah@LaptopMerah:case2
> cat platform/Dockerfile
FROM alpine:3.18

RUN apk update && apk add xvfb x11vnc openbox ttf-dejavu lxterminal firefox supervisor git bash

RUN git clone --depth 1 https://github.com/novnc/noVNC.git /opts/noVNC && \
    git clone --depth 1 https://github.com/novnc/websockify /opts/noVNC/utils/websockify && \
    rm -rf /opts/noVNC/.git && \
    rm -rf /opts/noVNC/utils/websockify/.git && \
    sed -i -- "s/ps -p/ps -o pid | grep/g" /opts/noVNC/utils/novnc_proxy

RUN rm -rf /tmp/* /var/cache/apk/*
RUN cd /opts/noVNC && cp vnc.html index.html
RUN mkdir -p /home/ && mkdir -p /etc/xdg/openbox

ADD certs/My* /tmp/
ADD start.sh /tmp
ADD supervisord.conf /tmp/supervisord.conf
ADD menu.xml /etc/xdg/openbox/menu.xml

RUN adduser user1 -D ; echo 'user1:user1qwerty' | chpasswd

EXPOSE 5910
EXPOSE 6666
EXPOSE 11111

ENTRYPOINT ["sh","-C","/tmp/start.sh"]
```

Dockerfile ini bertujuan untuk membangun sebuah container berbasis Alpine Linux yang menyediakan lingkungan desktop grafis ringan (menggunakan Openbox dan Xvfb) yang dapat diakses secara remote langsung melalui web browser menggunakan noVNC. Image ini sudah dilengkapi dengan aplikasi Firefox dan terminal, serta dikonfigurasi dengan supervisor untuk manajemen proses, sehingga memungkinkan pengguna untuk menjalankan dan berinteraksi dengan aplikasi GUI Linux di dalam container secara mudah dan terisolasi.

```
LaptopMerah@LaptopMerah:case2
> cat runcontainer/run-mylinux.sh
docker rm -f mylinux

docker run \
        -dit \
        --name mylinux \
        -p 12111:5910 \
        -p 11111:11111  \
        mylinux:1.0
```

run-mylinux.sh ini bertujuan untuk menjalankan container baru dari image **mylinux:1.0** yang dibangun dari Dockerfile, dengan memetakan port host 12111 ke port 5910 (untuk akses noVNC) dan port 11111 ke 11111 di dalam container, sehingga layanan desktop grafis tersebut dapat diakses dari luar.



Dengan menjalankan **build.sh** di direktori /platform untuk membangun Docker Image, lalu jalankan **run-mylinux.sh** yang berada di direktori /runcontainer, maka VNC dapat diakses melalui web browser pada alamat http://10.28.85.130:11111 yang menampilkan antarmuka noVNC untuk memulai koneksi remote desktop

Case 3:

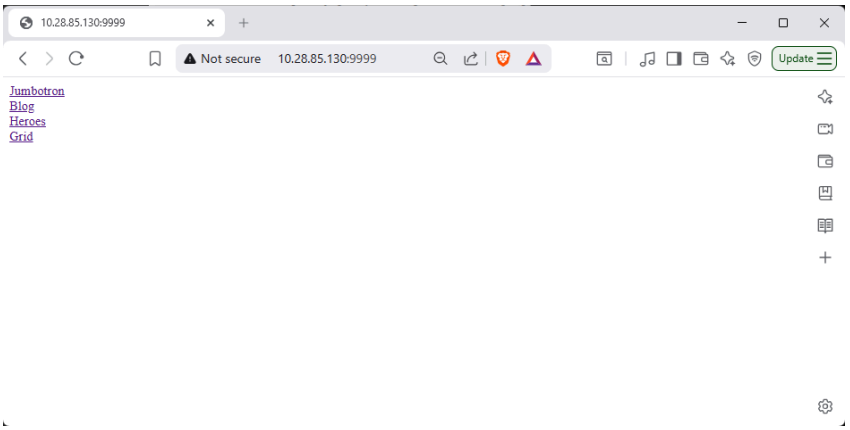
Pada case3 akan membuat image **mywebserver:2.0** yang menampilkan template bootstrap dengan Nginx menggunakan HTTP.

```
LaptopMerah@LaptopMerah:case3
> cat Dockerfile
FROM nginx:1.15.12-alpine

ADD nginx-conf/nginx.conf /etc/nginx/conf.d
COPY nginx-conf/nginx.conf /etc/nginx/conf.d/default.conf
ADD html/ /var/www/html/
```

Dockerfile ini bertujuan untuk membuat container web server Nginx yang ringan (berbasis Alpine) dengan konfigurasi berasal dari Task 2 Compose case1, di mana ia menyalin file konfigurasi nginx.conf lokal untuk menggantikan pengaturan default server dan memuat konten web statis dari direktori html/ host ke dalam container agar siap disajikan.

```
LaptopMerah@LaptopMerah:case3
> cat run-server.sh
docker rm -f webserver2
docker run -dit \
        --name webserver2 \
        -p 9999:80 \
        mywebserver:2.0
```

run-server.sh ini berfungsi untuk menjalankan container baru menggunakan image **mywebserver:2.0** yang dibangun dari Dockerfile, dengan memetakan port 9999 pada host ke port 80 di dalam container agar layanan web server tersebut dapat diakses dari luar melalui port 9999.



Dengan menjalankan **build.sh** untuk membangun Docker Image, lalu jalankan **run-server.sh**, maka hasilnya berupa sebuah layanan yang menyajikan template bootstrap yang tersedia pada http://10.28.85.130:9999

Case 4:

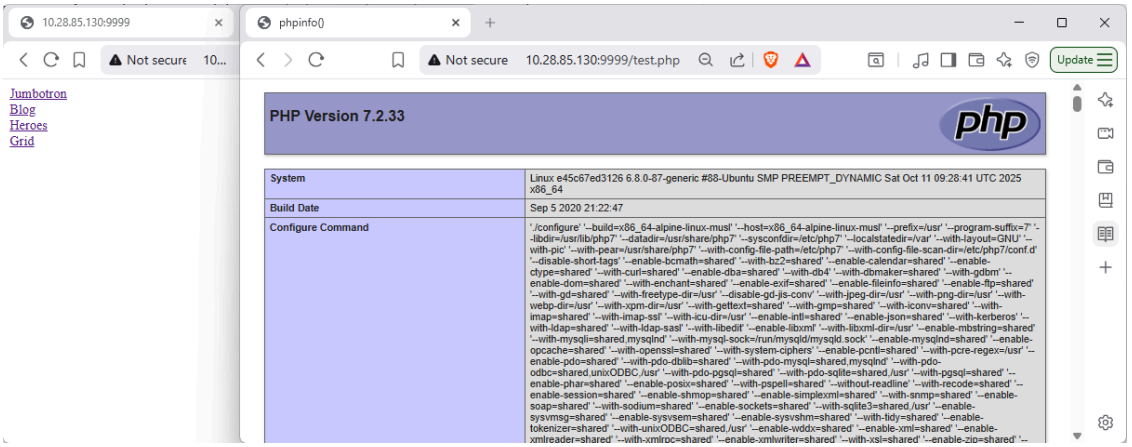
Pada case4 akan membuat image mywebserver:2.1 dan menampilkan berbagai template bootstrap menggunakan Nginx web server, dan terdapat aplikasi php pada path /test.php.

```
LaptopMerah@LaptopMerah:case4
> cat Dockerfile
FROM nginx:1.15.12-alpine

RUN apk update && apk add php7-fpm supervisor git curl

ADD nginx.conf /etc/nginx/conf.d
COPY nginx.conf /etc/nginx/conf.d/default.conf
ADD html/ /var/www/html/
ADD start.sh /
ADD supervisord.conf /tmp


ENTRYPOINT ["/bin/sh","/start.sh"]
```

Dockerfile ini bertujuan untuk membangun container web server Nginx yang mampu menjalankan aplikasi PHP (menggunakan php7-fpm), dilengkapi dengan supervisor untuk mengelola proses Nginx dan PHP secara bersamaan. Image ini menyalin konfigurasi Nginx custom, konten web dari folder html/, serta skrip startup, menjadikannya lingkungan siap pakai untuk menyajikan situs web dinamis berbasis PHP.

```
LaptopMerah@LaptopMerah:case4
> cat run-server.sh
docker rm -f webserver2
docker run -dit \
        --name webserver2 \
        -p 9999:80 \
        mywebserver:2.1
```

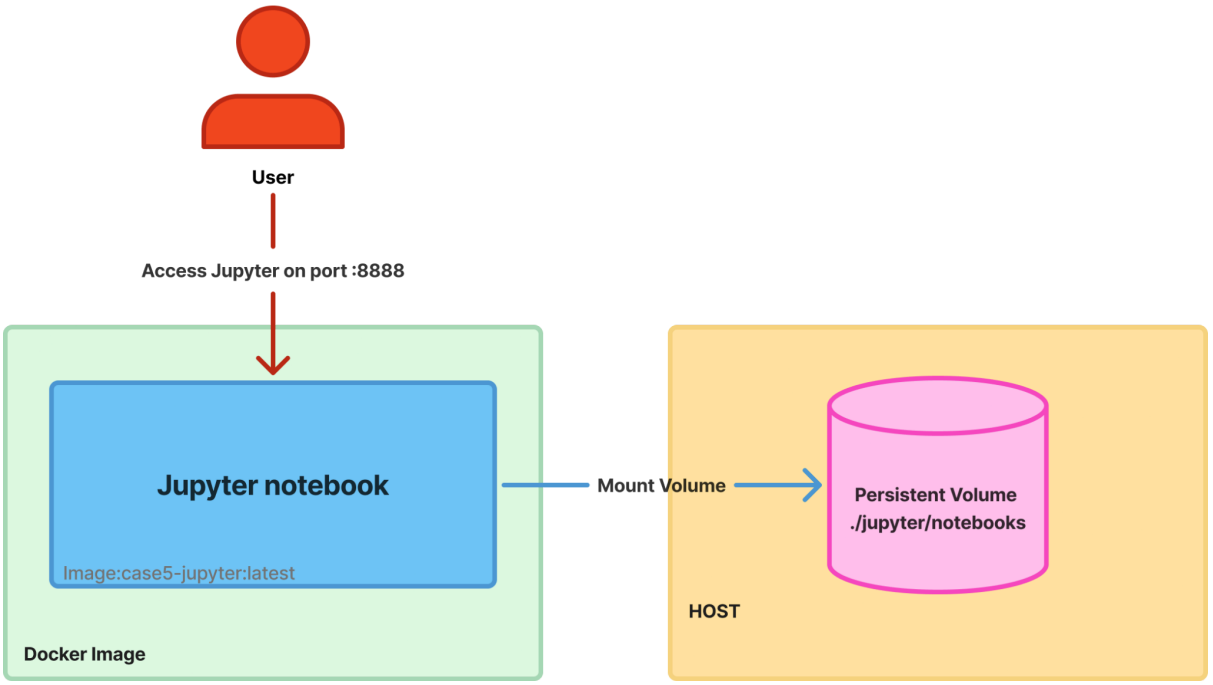
run-server.sh ini berfungsi untuk menjalankan container baru menggunakan image **webserver:2.1** yang dibangun dari Dockerfile, dengan memetakan port 9999 pada komputer host ke port 80 di dalam kontainer agar layanan web server (yang mendukung PHP) tersebut dapat diakses dari luar melalui port 9999.



Dengan menjalankan **build.sh** untuk membangun Docker Image, lalu jalankan **run-server.sh**, maka web akan menampilkan berbagai template bootstrap dan aplikasi PHP dengan protokol HTTP. Untuk melihat hasilnya, maka dapat akses http://10.28.85.130:9999, maka akan ditamplikan template bootstrap. Untuk melihat hasil aplikasi PHP maka dapat diakses pada http://10.28.85.130:9999/test.php.

Case 5:

Spesifikasi VM yang digunakan: 4096 MB Memory, 2 vCPU, 32GB Disk Size, IP address 10.21.85.130. Port yang digunakan pada case ini utamanya port 8888.



Jupyter Notebook ini cocok sebagai materi pembelajaran Docker image karena menggabungkan konsep-konsep penting containerization dalam satu studi kasus: membangun image dari Alpine Linux, menginstall dependencies, menjalankan multiple process dengan Supervisor, membuat user non-root untuk keamanan, mengekspos port untuk akses dari luar, serta menggunakan volume mount untuk persistent data. Dengan output berupa Jupyter yang langsung bisa diakses via browser, mahasiswa dapat membuktikan bahwa container berfungsi sekaligus memahami konsep Dockerfile, build, run, port mapping, dan volume secara terintegrasi.

```
LaptopMerah@LaptopMerah:case5
> cat jupyter/Dockerfile
FROM alpine:3.18

RUN apk update && apk add --no-cache \
    python3 \
    py3-pip \
    supervisor \
    curl \
    bash \
    gcc \
    python3-dev \
    musl-dev \
    linux-headers \
    && rm -rf /var/cache/apk/*

RUN python3 -m venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

COPY requirements.txt /tmp/requirements.txt
RUN pip install --no-cache-dir -r /tmp/requirements.txt

RUN adduser -D -h /home/jupyter jupyter
RUN mkdir -p /home/jupyter/notebooks && chown -R jupyter:jupyter /home/jupyter

COPY start.sh /tmp/start.sh
COPY supervisord.conf /tmp/supervisord.conf
COPY jupyter_config.py /home/jupyter/.jupyter/jupyter_notebook_config.py

RUN chmod +x /tmp/start.sh
RUN chown -R jupyter:jupyter /home/jupyter

EXPOSE 8888

WORKDIR /home/jupyter/notebooks

VOLUME ["/home/jupyter/notebooks"]

ENTRYPOINT ["/bin/sh", "/tmp/start.sh"]
```

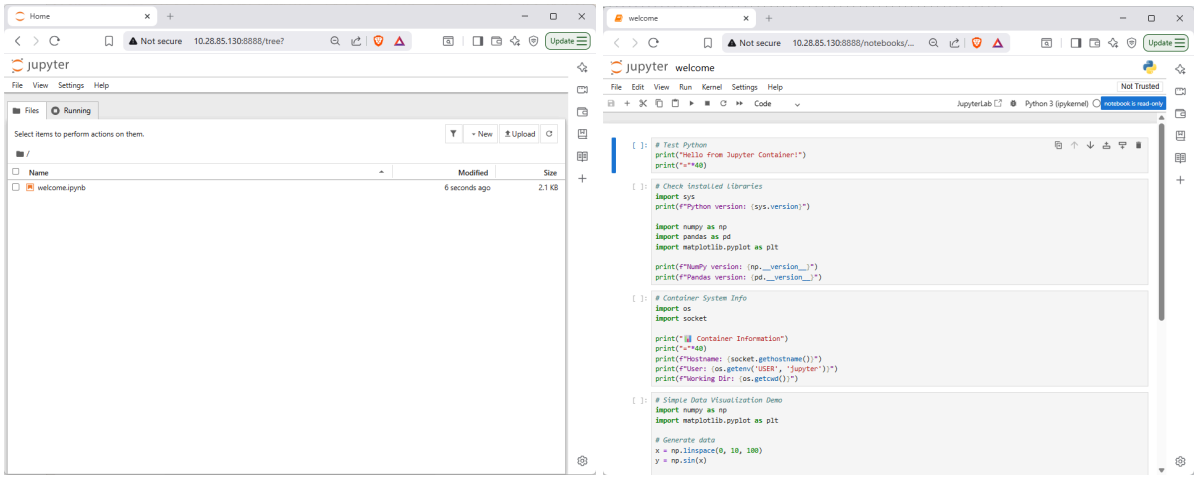
Dockerfile ini bertujuan untuk membangun sebuah container berbasis Alpine Linux versi 3.18 yang menjalankan Jupyter Notebook sebagai platform interaktif untuk pemrograman Python. Pada tahap awal, container menginstall Python 3, pip, supervisor sebagai process manager, serta beberapa build tools seperti gcc, python3-dev, dan musl-dev yang diperlukan untuk mengkompilasi library Python tertentu. Selanjutnya, dibuat sebuah virtual environment di /opt/venv untuk mengisolasi instalasi Python packages, kemudian file requirements.txt disalin ke container dan digunakan untuk menginstall Jupyter beserta library pendukung seperti numpy, pandas, dan matplotlib.

Untuk alasan keamanan, container membuat user non-root bernama "jupyter" dengan home directory di /home/jupyter, dimana folder notebooks akan menjadi tempat penyimpanan file notebook. File konfigurasi seperti start.sh, supervisord.conf, dan jupyter_config.py disalin ke dalam container untuk mengatur bagaimana Jupyter akan dijalankan.

Container mengekspos port 8888 yang merupakan port default Jupyter Notebook, dan mendefinisikan volume pada /home/jupyter/notebooks agar notebook yang dibuat dapat tersimpan secara persisten di host. Terakhir, container akan menjalankan script start.sh sebagai entrypoint yang memulai supervisor untuk mengelola proses Jupyter Notebook.

```
LaptopMerah@LaptopMerah:case5
> cat build-case-5.sh
#!/bin/bash
cd "./jupyter"
docker build -t case5-jupyter:latest .
LaptopMerah@LaptopMerah:case5
> cat run-case-5.sh
#!/bin/bash

docker rm -f case5-jupyter

docker run -d \
    --name case5-jupyter \
    -p 8888:8888 \
    -v "./jupyter/notebooks:/home/jupyter/notebooks" \
    case5-jupyter:latest
```

Pengoperasian cukup mengeksekusi **bash build-case-5.sh** untuk membangun Docker image dari Dockerfile yang berisi konfigurasi Jupyter Notebook beserta seluruh dependencies-nya, dilanjutkan dengan **bash run-case-5.sh** untuk menjalankan container dari image yang sudah dibangun dengan port 8888 dan volume mount untuk persistent data. Proses ini akan secara otomatis menghapus container lama jika ada, kemudian membuat container baru yang dapat diakses melalui browser di http://10.28.85.130:8888.



Hasil dari implementasi arsitektur ini, menampilkan antarmuka Jupyter Notebook yang dapat diakses langsung melalui browser pada http://10.28.85.130:8888. Tampilan ini membuktikan bahwa Image Container berhasil berjalan dengan baik, dimana pengguna dapat langsung membuat, mengedit, dan menjalankan kode Python secara interaktif menggunakan library seperti numpy, pandas, dan matplotlib yang sudah terinstall di dalam container.

Kesimpulan

Dengan tugas ini yang mempelajari empat studi kasus pembuatan Docker image, mulai dari Case 1 (Apache + PHP web server), Case 2 (Desktop Linux via noVNC yang dapat diakses melalui browser), Case 3 (Nginx static web server), hingga Case 4 (Nginx + PHP-FPM), saya memperoleh pemahaman mendalam tentang berbagai konsep containerization seperti pemilihan base image, instalasi dependencies, konfigurasi Supervisor sebagai process manager, pengaturan port expose, dan penggunaan volume untuk persistent data. Pembelajaran ini kemudian saya terapkan untuk membuat kreasi sendiri pada Case 5 berupa container Jupyter Notebook yang menggabungkan konsep-konsep tersebut, yaitu menggunakan Alpine Linux sebagai base image yang ringan, menginstall Python beserta library data science, mengelola proses dengan Supervisor, serta mengimplementasikan volume mount agar notebook tetap tersimpan meskipun container dihapus. Kreasi ini menunjukkan bahwa pemahaman dari keempat case sebelumnya dapat diaplikasikan untuk membangun container sesuai kebutuhan, dalam hal ini menyediakan environment interaktif untuk pemrograman Python yang dapat diakses langsung melalui browser.