MODUL PEMBELAJARAN

TEKNOLOGI KOMPUTASI AWAN

DOCKER

Modul 3 Docker

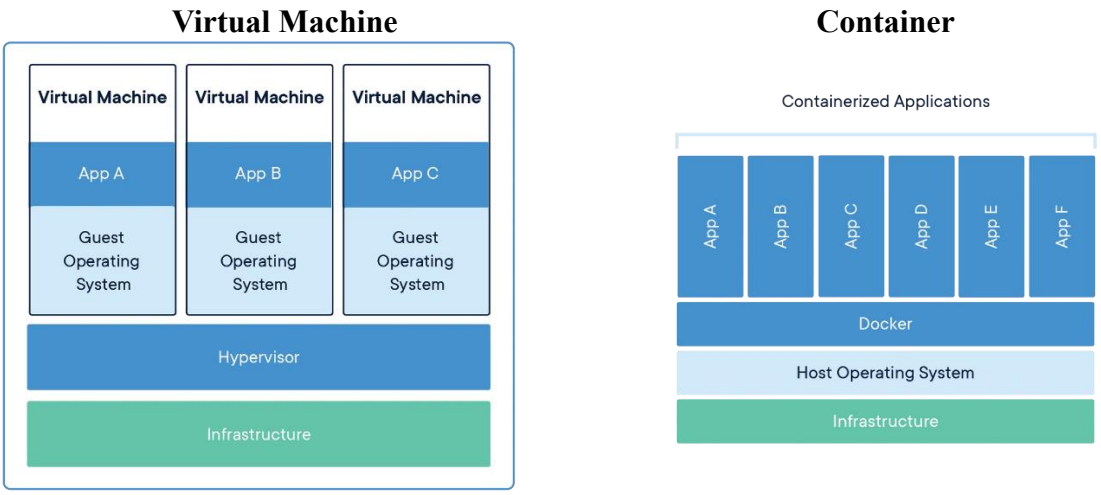
Dosen pengampu :
Henning Titi Ciptaningtyas, S.Kom, M.Kom.

Disusun oleh :
Ilham Muhammad Sakti
Naufal Dhiya Ulhaq
Sharira Saniane

Daftar isi

Dasar Teori

A. Virtual Machine vs Container



A.1 Virtual Machine

Virtual Machine (VM) merupakan abstraksi perangkat keras fisik yang mengubah satu server menjadi banyak server. Hypervisor atau virtual machine manager memungkinkan beberapa VM dapat berjalan pada satu mesin. Setiap membuat sebuah VM, diharuskan menginstall sistem operasinya juga sehingga ukuran aplikasi menjadi besar karena tidak hanya aplikasi tetapi juga sistem operasinya. Kerugian lain VM adalah apabila membuat aplikasi yang membutuhkan banyak VM, maka akan memakan banyak resource (VM dapat berukuran lebih dari 1 GB).

A.2 Container

Container adalah abstraksi pada lapisan aplikasi yang mengemas kode dan dependencies secara bersamaan. Beberapa container dapat berjalan di mesin yang sama dan berbagi kernel OS dengan container lain, masing-masing berjalan sebagai proses yang terisolasi di ruang pengguna dengan menggunakan container manager. Container menggunakan lebih sedikit ruang daripada VM (container images biasanya berukuran puluhan MB).

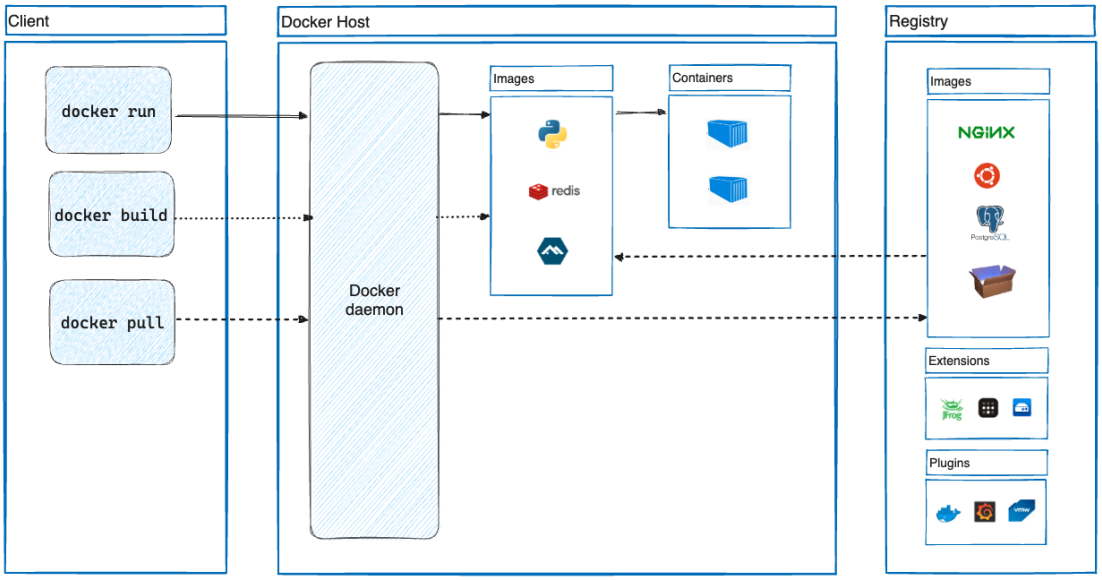
B. Docker

B.1 Pengertian Docker

Docker adalah platform open source untuk mengembangkan, mengirim, dan menjalankan aplikasi. Docker memungkinkan untuk memisahkan aplikasi dari infrastruktur menggunakan container sehingga dapat mengirimkan software lebih cepat. Container berfungsi sebagai lingkungan eksekusi terisolasi untuk menjalankan aplikasi, termasuk source code, runtime, dan dependencies yang diperlukan.

Docker memungkinkan developer dapat membuat container portabel yang konsisten yang dapat berjalan di berbagai lingkungan komputasi, termasuk local machine, server cloud, atau lingkungan pengembangan dan produksi lainnya secara bersamaan. Docker membantu mengisolasi aplikasi dan dependencies, sehingga aplikasi dapat berjalan secara konsisten di seluruh lingkungan tanpa mengganggu sistem operasi host atau aplikasi lainnya.

B.2 Arsitektur Docker



Docker menggunakan arsitektur client-server. Docker client menerima perintah dan mengirimkannya ke Docker daemon untuk menjalan perintah tersebut seperti building, running, dan distributing Docker container. Docker client dan daemon dapat berjalan pada sistem yang sama, atau Anda dapat menyambungkan Docker client ke remote Docker daemon.

Docker client dan daemon berkomunikasi menggunakan REST API, melalui soket UNIX atau network interface. Docker client lainnya adalah Docker Compose, yang memungkinkan bekerja dengan aplikasi yang terdiri atas sekumpulan container.

Docker Daemon

Docker Daemon (dockerd) mendengarkan Docker API requests dan mengelola objek Docker seperti image, container, jaringan, dan volume. Sebuah Docker daemon juga dapat berkomunikasi dengan daemon lain untuk mengelola layanan docker.

Docker Client

Docker Client (docker) adalah interface pengguna untuk berinteraksi dengan Docker. Saat menggunakan perintah seperti docker run, klien mengirim perintah ini ke dockerd, yang menjalankannya. Perintah Docker menggunakan Docker API. Docker Client dapat berkomunikasi dengan lebih dari satu daemon.

Docker Desktop

Docker Desktop adalah aplikasi Docker yang dipasang pada Mac, Windows, atau Linux environment yang memungkinkan untuk membangun dan berbagi aplikasi dan layanan mikro yang terkontainerisasi. Docker desktop mencakup Docker daemon (dockerd), Docker client (docker), Docker compose, Docker content trust, kubernetes, dan credential helper.

Docker Registry

Docker Registry adalah tempat untuk menyimpan Docker Image. Dengan menggunakan Docker Registry, kita bisa menyimpan Image yang kita buat, dan bisa digunakan di Docker Daemon dimanapun selama bisa terkoneksi ke Docker Registry.

Saat menggunakan perintah docker pull atau docker run, Docker melakukan pull image yang dibutuhkan dari registry yang dikonfigurasikan. Saat menggunakan perintah docker push, Docker melakukan push image ke registry yang telah dikonfigurasi. Berikut contoh-contoh Docker Registry:

- Docker Hub (default)
- Digital Ocean Container Registry
- Google Cloud Container Registry
- Amazon Elastic Container Registry
- Azure Container Registry

Docker Objects

Docker Object merupakan komponen-komponen yang terdapat pada Docker. Docker Object dapat meliputi image, container, network, volume, dan objek lainnya.

Instalasi Docker

Docker Desktop (Windows, Mac, Linux)

1. Download docker desktop.
2. Buka file installer yang telah diunduh, kemudian ikuti panduan instalasi yang muncul. Ini akan menginstal Docker Desktop ke komputer Anda.
3. Jalankan Docker Desktop.
4. Cek versi docke dengan perintah

```
docker --version
```

5. Selanjutnya, jalankan perintah berikut untuk memeriksa apakah Docker dapat menjalankan kontainer Hello World:

```
docker run hello-world
```

6. Jika semuanya berjalan dengan baik, Anda akan melihat pesan yang mengkonfirmasi bahwa Docker telah diinstal dengan sukses.
7. Linux
   Lakukan instalasi Docker Engine
   Lakukan instalasi sampai post-installation steps for Linux.
   Setelah itu, install Docker Compose

Object Docker

1. Docker Image

Docker Image adalah *read-only* template yang berisi intruksi untuk membuat Docker Container. Image dapat dibuat sendiri menggunakan Dockerfile atau mengambil milik orang lain di Docker Hub atau bahkan memodifikasinya. Di dalam Docker Image berisi aplikasi dan *dependencies.* Sebelum menjalankan aplikasi di Docker, pastikan terlebih dahulu memiliki Docker Image aplikasi tersebut.

Berikut beberapa perintah pada Docker Image.

Perintah	Keterangan
build	Untuk membuat image dari Dockerfile
history	Untuk melihat history image
Import	Untuk import image dari sebuah file
Inspect	Untuk melihat informasi detail dari satu atau lebih image
Load	Untuk memuat image dari sebuah arsip yang telah disimpan
ls	Untuk melihat daftar image yang telah terunduh
prune	Untuk menghapus image yang tidak digunakan
Pull	Untuk mengunduh image dari Registry
push	Untuk mengunggah image ke Regisry
rm	Untuk menghapus satu atau lebih image
save	Untuk menyimpan satu atau lebih image ke dalam sebuah arsip

Contoh

```
# mencari image
docker search [imageName]
```

```
# list image
docker image ls

# download image
docker image pull [imageName]:[tag]
docker pull [imageName]:[tag]
docker pull httpd
```

```
C:\Users\ilham>docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
a803e7c4b030: Already exists
053327351b4a: Pull complete
de42e9dfbbe1: Pull complete
9d28e265584b: Pull complete
```

```
C:\Users\ilham>docker search httpd
NAME                            DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
httpd                           The Apache HTTP Server Project                  4554      [OK]
clearlinux/httpd                httpd HyperText Transfer Protocol (HTTP) ser…   5
paketobuildpacks/httpd                                                          0
vulhub/httpd                                                                    0
```

2. Docker Container

Docker Container adalah instance image yang dijalankan. Container berisi aplikasi dan dependensinya, dimana setiap container terisolasi dengan host maupun container lainnya. Container dapat berjalan pada berbagai environment meskipun memiliki perbedaan infrastruktur dan konfigurasinya. Anda dapat membuat, menjalankan, menghentikan, memindahkan, atau menghapus kontainer menggunakan Docker API atau CLI. Selain itu, juga dapat menyambungkan container ke asdsatu atau beberapa jaringan, melampirkan penyimpanan ke container tersebut. Docker Contaoner mempermudah dalam mengelola dan menjalankan aplikasi lintas environment tanpa harus mengkhawatirkan masalah konfigurasi dan dependensinya.

Perintah docker container

Berikut beberapa perintah pada Docker Container.

Perintah	Keterangan
attach	Untuk masuk pada terminal container sehingga dapat menjalan perintah pada container yang sedang aktif
commit	Untuk membuat image baru dari perubahan sebuah container
cp	Untuk menyalin file/folder antara file sistem host dan file sistem container
create	Untuk membuat container baru
diff	Untuk memeriksa perubahan pada file atau direktori di file sistem container
exec	Untuk menjalankan perintah dalam container yang sedang berjalan
export	Untuk mengekspor container sebagai arsip tar
inspect	Untuk menampilkan informasi detail pada satu atau beberapa container
kill	Untuk menghentikan satu atau lebih kontainer yang sedang berjalan secara paksa

logs	Untuk mengambil log dari sebuah container
ls	Untuk melihat daftar container yang sedang berjalan
pause	Untuk menghentikan sementara semua proses pada satu container atau lebih
port	Untuk melihat pemetaan port atau pemetaan spesifik pada container
prune	Untuk menghapus semua container yang tidak berjalan
rename	Untuk mengganti nama container
restart	Untuk memulai ulang satu atau beberapa container
rm	Untuk menghapus satu atau beberapa container
run	Untuk membuat dan menjalankan container baru dari sebuah image
start	Untuk memulai satu atau lebih container yang dihentikan
stats	Untuk menampilkan informasi statistik penggunaan sumber daya container
stop	Untuk menghentikan satu atau beberapa container yang sedang berjalan
top	Untuk menampilkan proses yang sedang berjalan dari sebuah container
unpause	Untuk meneruskan proses pada satu container atau lebih yang telah dijeda
update	Untuk memperbarui konfigurasi satu atau beberapa container
wait	Untuk menunggu container selesai menjalankan sebuah perintah sebelum melanjutkannya

Contoh :

```
# list container yang sedang dan tidak berjalan
docker container ls -a
docker ps -a

# memulai/ menjalankkan container
docker container start [containerId/containerName]
docker start [containerId/containerName]

# membuat dan menjalankan container
docker run [options] [image] [command] [arg..]
docker run -itd --name container1 ubuntu:latest

# melihat log secara realtime
docker container logs -f [containerId/containerName]

# untuk mengeksekusi perintah ke dalam container
docker container exec [options] [containerId/containerName] [command]
docker container exec -i -t container1 /bin/bash
docker exec d87283c0e4d7 ls /etc/nginx

# membuat container baru menggunakan base image httpd
docker run --name tka -p 9000:80 -d httpd
# Keterangan:
# --name : untuk memberikan nama pada container
```

```
# -p    : sebagai port forwarding (9000 port host, 80 port container)
# -d    : image container dijalankan sebagai servis
```

```
C:\Users\ilham>docker run --name tka -p 9000:80 -d httpd
fab436671348df33fade7955d7d51f7ea6ba897b5d29ee388fe5a2cb453588f2

C:\Users\ilham>docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED         STATUS         PORTS                  NAMES
fab436671348   httpd          "httpd-foreground"   9 seconds ago   Up 8 seconds   0.0.0.0:9000->80/tcp   tka
8718ba9e973f   ubuntu:latest  "/bin/bash"          3 hours ago     Up 54 minutes                         iniUbuntu3
```

It works!

3. Docker Volume

Docker tidak menyimpan state atau data apapun di dalam container. Sehingga apabila container mati kemudian dinyalakan lagi maka state akan kembali seperti semula. Semua perubahan tidak tersimpan, berbeda dengan virtual mesin yang menyimpan data dan state. Untuk mengatasi hal tersebut, docker memiliki fungsi volume untuk menyimpan perubahan data pada komputer hostnya. Sehingga ketika kontainer mati atau dihapus, data tetap tersimpan di komputer host dan dapat digunakan kembali oleh kontainer yang sama atau berbeda. Untuk mengaktifkan eksternal volume ikuti langkah-langkah berikut:

1. Matikan dan hapus kotnainer yang telah kita buat sebelumnya.

```
docker stop tka
docker rm tka
```

2. Buat folder baru bernama src dan tambahkan file index.html

```
mkdir tka
 echo "Hello kelas Teknologi Komputasi Awan!" > tka/index.html
```

3. Buat dan jalankan container.

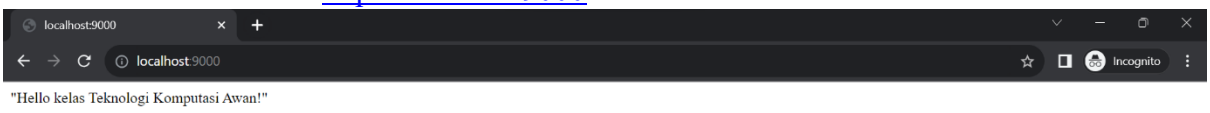
```
# linux
docker run --name tka -v "$PWD"/tka:/usr/local/apache2/htdocs/ -p 9000:80 -d
httpd

# windows
docker run --name tka -v %CD%/tka:/usr/local/apache2/htdocs/ -p 9000:80 -d httpd
```

Keterangan:

- Argument -v untuk menambahkan docker volume.
- perintah "$PWD"/src:/usr/local/apache2/htdocs/, akan menghubungkan folder src ke folder htdocs pada kontainer. Perintah "$PWD" digunakan untuk mendapatkan posisi folder saat ini.

4. Buka browser dan akses http://localhost:9000



Perintah Docker Volume

Perintah	Keterangan
create	Untuk membuat sebuah volume
inspect	Untuk menampilkan detail informasi satu atau lebih volume
ls	Untuk melihat daftar volume
prune	Untuk menghapus volume yang tidak digunakan
rm	Untuk menghapus satu atau lebih volume

Contoh

```
docker volume ls
docker volume create tka
```

Dockerfile

Dockerfile adalah sebuah file teks yang berisi serangkaian instruksi untuk membangun sebuah image Docker. Dockerfile digunakan untuk mendefinisikan bagaimana sebuah image Docker harus dibuat, konfigurasi kontainer, dan apa yang harus terdapat di dalam kontainer tersebut. Ketika Dockerfile dieksekusi dengan perintah docker build, Docker Engine akan membaca instruksi-instruksi dalam Dockerfile dan membangun image Docker yang sesuai.

Perintah Dockerfile

Perintah	Keterangan
FROM	Untuk menentukan base image yang akan digunakan
COPY	Untuk menyalin file atau folder dari host ke dalam image.
ADD	Untuk menyalin file atau folder dari host ke dalam image, bisa juga digunakan untuk men-download file dari URL dan mengekstraknya ke dalam image.
RUN	Untuk menjalankan perintah pada layer yang sedang dibangun dan membuat image baru.
CMD	Untuk menentukan perintah default yang akan dijalankan saat container di-start.
ENTRYPOINT	Untuk menentukan perintah yang akan dijalankan saat container di-start, dapat juga di-overwrite oleh perintah saat container di-run.
ENV	Untuk menentukan environment variable di dalam container.
EXPOSE	Untuk menentukan port yang akan di-expose dari container ke host.

VOLUME	Untuk menentukan direktori yang akan di-mount sebagai volume di dalam container.

Contoh Dockerfile

1. Buatlah folder bernama www

```
mkdir www
```

2. Buatlah file index.php di dalam folder www

```
echo "<?php phpinfo(); ?>" > www/index.php
```

3. Buat Dockerfile dengan isi sebagai berikut:

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y apache2 php7.0 php7.0-fpm libapache2-mod-php && apt-get clean && rm -rf /var/lib/apt/lists/*

COPY www/index.php /var/www/html

WORKDIR /var/www/html

RUN rm index.html

WORKDIR /

CMD ["apachectl", "-D", "FOREGROUND"]

EXPOSE 80
```

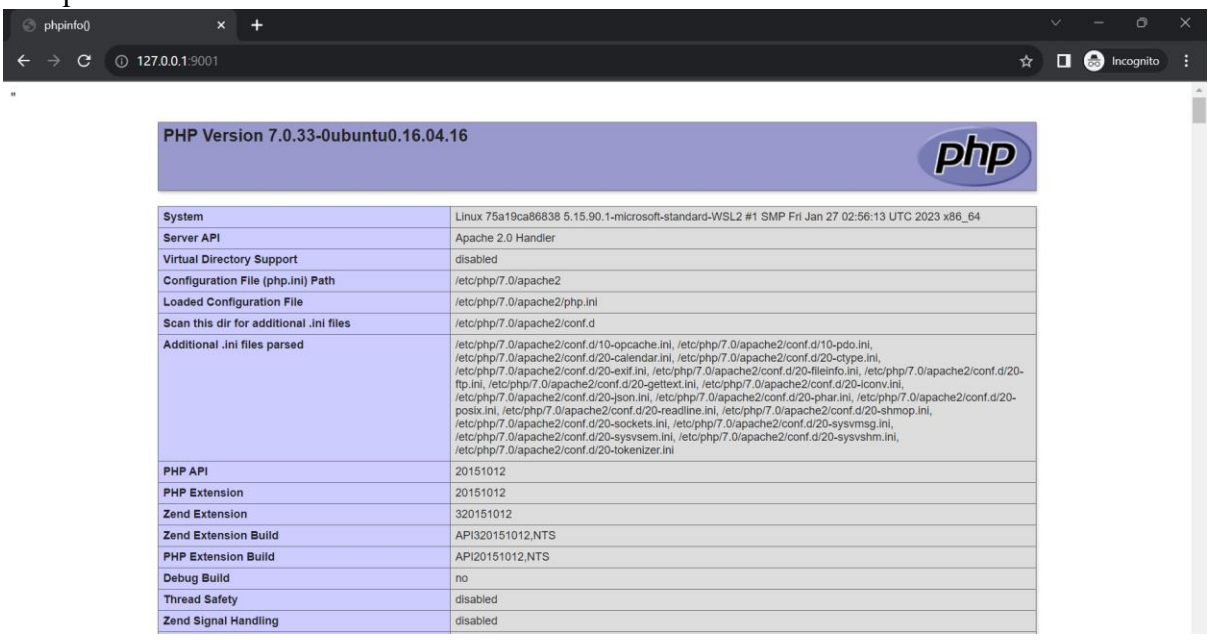
4. Buat image dengan perintah

```
docker build -t ubuntu-tka-image ./
```

5. Buat container baru dengan perintah

```
docker run --name ubuntu-tka-container -p 9001:80 -d ubuntu-tka-image
```

6. Cek pada browser



Docker Compose

Docker Compose merupakan sebuah alat yang digunakan untuk mendefinisikan dan menjalankan aplikasi Docker multi-container menggunakan file konfigurasi YAML. Dengan Docker Compose kita dapat menentukan Docker Image untuk setiap Docker Container, mengatur konfigurasi jaringan, menentukan volume yang dibutuhkan, dan konfigurasi lainnya. Docker Compose berjalan di semua lingkungan; production, staging, development, pengujian, serta CI workflows. Docker Compose juga memiliki perintah untuk mengelola seluruh siklus hidup aplikasi:

- Start, stop, dan rebuild service.
- Melihat status layanan yang sedang berjalan.
- Stream log output dari layanan yang sedang berjalan.
- Menjalankan perintah satu kali pada sebuah layanan.

Contoh Implementasi

Step 1: Define the application dependencies

1. Membuat direktori composetest:

```
mkdir composetest
cd composetest
```

2. Buat file bernama app.pydi direktori proyek dan copy kode berikut:

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

Dalam contoh ini, redisadalah nama host container redis di jaringan aplikasi. Port default Redis adalah 6379.

3. Buat file requirements.txtdi direktori proyek Anda dan tempelkan kode berikut di:

```
flask
redis
```

Step 2: Create a Dockerfile

Buat Docker Image menggunakan Dockerfile.

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

Keterangan :

- Build image dengan base image Python 3.7.
- Set working direktori ke /code.
- Set environment variables yang digunakan oleh perintah flask .
- Install gcc dan dependensi lainnya.
- Copy requirements.txt and install Python dependensi.
- Menambahkan port 5000
- Copy direktori saat ini . ke dalam working direktori image.
- Set default command untuk container untuk flask run.

Step 3: Define services in a Compose file

Buat sebuah file bernama compose.yaml di direktori proyek Anda dan paste kode berikut :

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

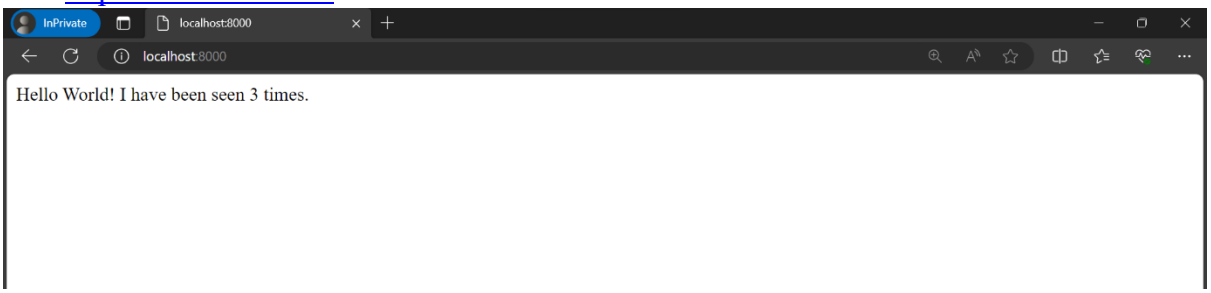
File compose ini berisi dua layanan, yaitu web dan redis.

web services menggunakan Image yang dibuat dari file Dockerfile di direktori saat ini. Lalu, ia bind container dan host machine ke port 8000. Pada contoh ini menggunakan port default untuk server web Flask, 5000.

redis services menggunakan public Redis Image.

Step 4: Build and run your app with Compose

1. Start aplikasi dengan menjalankan perintah.
   ```
   docker compose up
   ```
2. Cek pada browser untuk melihat aplikasi yang sedang berjalan. http://localhost:8000/ atau http://127.0.0.1:8000



Latihan Soal

Buatlah Dockerfile dengan base Image adalah python:3.8 dan install dependensi -no-cache-dir. Atur direktori aplikasi di /usr/src/app serta definisikan port 5000. Setelah itu, jalankan perintah python ./app.py.

Soal Praktikum Modul 3

Docker

1. Waktu pengerjaan praktikum sesuai dengan timeline yang diberikan (Selasa jam 10.00 WIB sampai Sabtu jam 22.00 WIB).
2. Hasil pengerjaan praktikum dalam bentuk video demonstrasi diupload ke Youtube dengan status unlisted dan format judul TKA_[Kelompok]_[Judul Modul] (Contoh: TKA_E03_Docker)
3. Praktikan tidak diperbolehkan menanyakan jawaban dari soal yang diberikan kepada asisten maupun praktikan dari kelompok lainnya.
4. Pengerjaan soal sesuai dengan modul yang telah diajarkan.
5. Jika ditemukan indikasi kecurangan dalam bentuk apapun di pengerjaan soal shift, maka nilai dianggap 0.
6. Pengumpulan video demonstrasi akan dikumpulkan melalui Google Form pada link yang akan dibagikan 1 jam sebelum deadline pengerjaan soal
7. Harap mengumpulkan tepat waktu karena apabila lebih dari waktu yang ditentukan akan mendapat pengurangan poin.

Reference

Reference documentation | Docker Docs

cloud-2018/docker at master · fathoniadi/cloud-2018 (github.com)