



Nama: **Arsyadana Estu Aziz (121140068)**
Mata Kuliah: **Pervasive Computing (IF4025)**

Tugas Ke: **2**
Tanggal: 5 November 2024

1 Introduction

Quick Question, *Kubernetes itu netesin apa?* Perkenalkan nama Saya Arsyadana Estu Aziz, dan saya akan menjelaskan tentang apa itu Docker dan membuat apps dan menguploadnya di *Docker*. Pengerjaan laporan dilakukan dalam LaTeX

2 Docker

Apa itu Docker? Docker adalah sebuah platform yang digunakan untuk membuat, mendeploy dan menjalankan apps dengan menggunakan container.

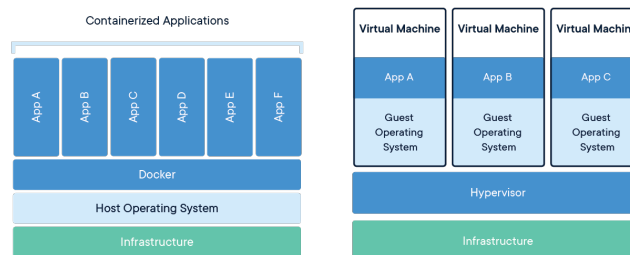


Figure 1: Docker Overview

Docker sendiri beda seperti OS pada umumnya, dimana di dalam docker kita menggunakan Images hanya untuk proses build, deployment dan run aplikasi. Berbeda dengan OS / Virtual Machine dimana kamu bisa berinteraksi dengan OS tersebut.

Pada kesempatan kali ini, saya akan melakukan deploy aplikasi CPU-Monitoring ke Docker dan Mungkin mengelolanya dengan Kubernetes. Docker sendiri menjalankan aplikasinya dalam bentuk Container, dan Kubernetes mengelola Container itu secara otomatis sehingga developer tidak perlu pusing dalam proses setup di mesin lain.

Hal pertama yang dilakukan adalah membuat environment conda dan melakukan instalasi dalam file **requirements.txt**

```
1 Flask==2.0.1
2 Werkzeug==2.0.1
3 firebase-admin==5.0.0
4 psutil
```

Setelah itu Kita membuat sebuah file **Dockerfile** (Sebuah perintah / script) yang secara sederhana instruksi untuk membangun Docker Image, yang digunakan untuk menjalankan Docker container.

```
1 FROM python:3.10
2
```

```
3 WORKDIR /app
4
5 COPY requirements.txt .
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 COPY . .
10
11 CMD ["python", "app.py", "--host=0.0.0.0", "--port=5000"]
```

3 Implementation

Bagian ini masih mirip seperti dengan Hands-on sebelumnya namun saya menambahkan beberapa 1 Halaman Landing Page dan mengubah routing dari App.py sebagai berikut.

```
1 from flask import Flask, render_template, jsonify
2 import firebase_admin
3 from firebase_admin import credentials, db
4 import os
5
6 app = Flask(__name__)
7
8 # Inisialisasi Firebase Admin SDK
9 cred = credentials.Certificate("pc-cpu-monitoring-firebase-adminsdk-2yfd5-cf147c47f5.json")
10 firebase_admin.initialize_app(cred, {
11     'databaseURL': 'https://pc-cpu-monitoring-default-rtdb.asia-southeast1.firebaseio.com/'
12 })
13
14 # Referensi ke node 'cpu_usage' di Firebase
15 ref = db.reference('cpu_usage')
16
17 @app.route('/')
18 def home():
19     return render_template('home.html')
20
21 @app.route('/graph')
22 def index():
23     return render_template('index.html')
24
25 ## Getting the last currently updated snapshot data
26 ## And put it in the List Comprehension before sending the Json data to App.py
27 @app.route('/data')
28 def get_data():
29     snapshot = ref.order_by_key().limit_to_last(10).get()
30     data = [{'cpu': value['cpu'], "disk": value["disk"], "net": value["net"], 'timestamp': value['timestamp']} for _, value in snapshot.items()]
31     return jsonify(data)
32
33 if __name__ == '__main__':
34     app.run(host='0.0.0.0', port=5000)
```

Kode 1: Python App.py

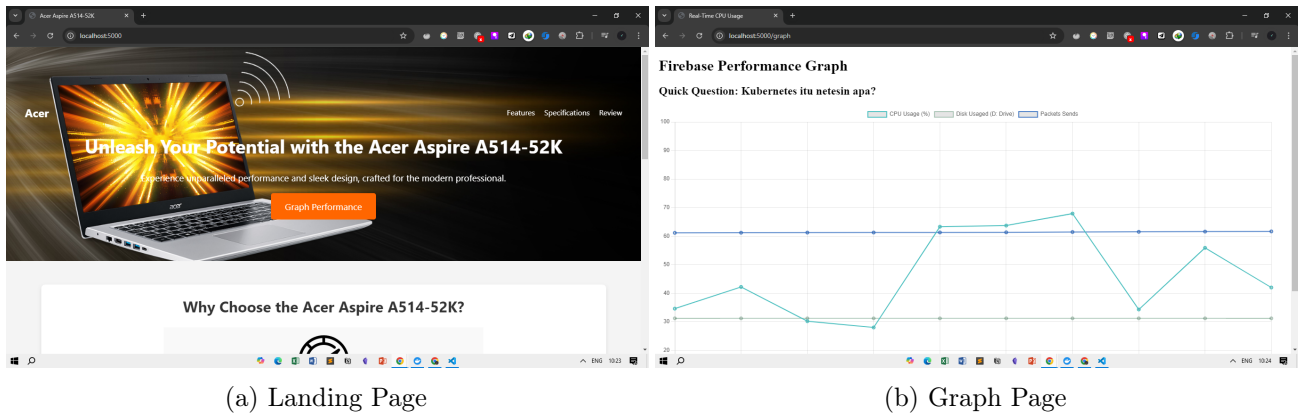


Figure 2: CPU Monitoring App

3.1 Running the Docker Container On Local

Pada bagian ini, kita dapat melakukan proses running pada aplikasi ini secara local dengan beberapa langkah.

1. Melakukan proses build. Setelah kita membuat aplikasi, kita dapat melakukan proses build dengan command berikut:

```
1 docker build -t performance-monitoring .
2
```

flag **-t** sendiri merupakan tag dari images yang akan dibuat dan **.** merujuk pada root directory.

2. Menjalankan Docker secara Lokal Setelah membuat images di Docker Desktop, kita dapat melakukan proses running images tersebut di dalam sebuah kontainer dengan command berikut:

```
1 docker run -p 5000:5000 performance-monitoring:latest
2
```

flag **-p** disini merujuk pada port 5000 ketika dijalankan dan **latest** merujuk pada images versi terbaru. Sekarang cukup buka **localhost:5000** dan kalian bisa menggunakan aplikasi yang berjalan di Image docker. (Bukan di mesin kalian).

4 Docker Hub

Kita sudah bisa melakukan proses build and run di lokal dengan Docker Desktop, sudah saatnya kita melakukan proses share images ini ke Internet sehingga orang lain bisa menggunakan images / app ini.

Hal pertama yang perlu dilakukan adalah dengan membuat akun di [Docker Hub](#) dan membuat repository (sebagai tempat untuk mengupload images dari Docker Desktop kita)

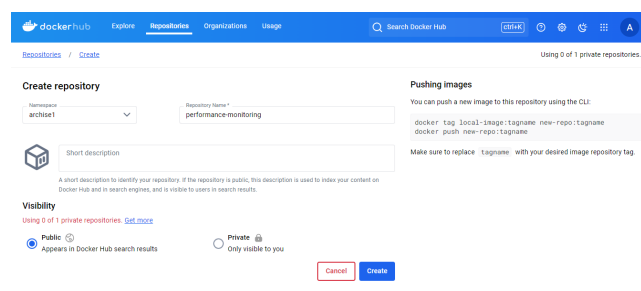
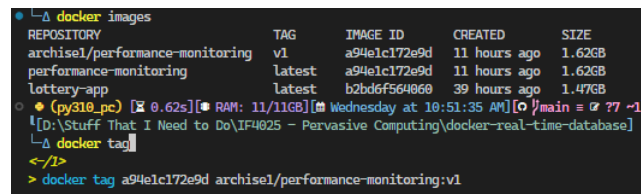


Figure 3: Docker Hub

It takes a while, dan ketika selesai kalian bisa mengunggah Docker Images ke Docker Hub dengan beberapa Steps berikut (Untuk saat ini terdapat bug pada Docker 4.34 sehingga tidak bisa langsung upload ke Docker Hub melalui Docker Desktop, Referensi. [Issue](#)).

1. Hal pertama yang perlu dilakukan adalah membuat tag pada image untuk proses upload ke Docker hub, disini kita menggunakan Image ID untuk membuat Image (performance-monitoring) dengan nama username **Docker/image-name**



```

❯ docker images
REPOSITORY              TAG          IMAGE ID       CREATED        SIZE
archisel/performance-monitoring v1          a94e1c172e9d  11 hours ago  1.62GB
performance-monitoring latest      a94e1c172e9d  11 hours ago  1.62GB
lottery-app             latest      b2bd6f564060  39 hours ago  1.47GB
❯ docker tag a94e1c172e9d archisel/performance-monitoring:v1

```

Figure 4: Docker Tag

```

1 docker images
2 docker tag a94e1c172e9d archisel/performance-monitoring:v1

```

Selanjutnya adalah melakukan proses upload ke Docker Hub dengan command berikut:

```

1 docker push archisel/performance-monitoring:v1

```

bagian **v1** adalah merupakan tag dari Docker images ini.

Proses pengunduhan ini akan memakan waktu yang lumayan lama, dan ketika selesai kalian bisa membuka Docker Hub kalian melihat image nya disana. Kalian bisa mengunduh images tersebut dari

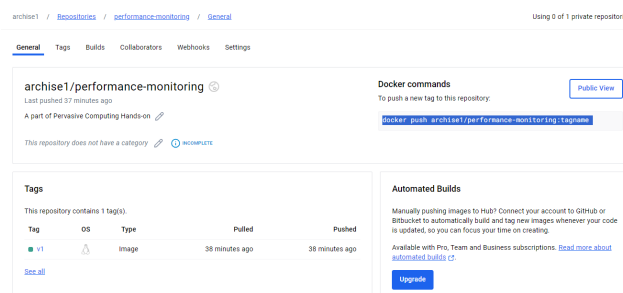


Figure 5: Docker Hub

Docker hub dengan perintah

```

1 docker pull <nama-repository>

```

5 Kubernetes

Quick Question: Kubernetes itu netesin apa? Kubernetes (k8s) adalah sebuah platform untuk mengatur multi container app secara harmonis. Sejujurnya ini terkesan overkill karena kita hanya mempunyai 1-container app untuk menggunakan Kubernetes. Namun terdapat beberapa langkah hal yang bisa dilakukan untuk deploy dengan Kubernetes.

5.1 Setup Kubernetes

Hal pertama yang perlu dilakukan adalah menyalakan Kubernetes dalam Docker Desktop. Untuk yang menggunakan windows dan WSL pastikan melakukan konfigurasi ini [Issue](#)

Setelah itu kita perlu membuat file **deployment.yaml**

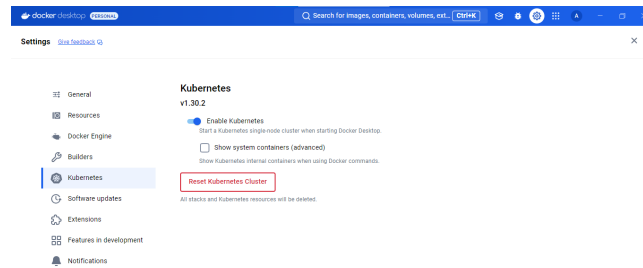


Figure 6: Setup Kubernetes

```

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: performance-monitoring
5 spec:
6   replicas: 3
7   selector:
8     matchLabels:
9       app: performance-monitoring
10  template:
11    metadata:
12      labels:
13        app: performance-monitoring
14    spec:
15      containers:
16        - name: performance-monitoring
17          image: archisel1/performance-monitoring:v1
18          ports:
19            - containerPort: 5000

```

Pastikan disini kita menggunakan image yang sesuai dengan image yang di unggah di Docker Hub, contohnya **archisel1/performance-monitoring:v1** dengan v1 sebagai tag yang akan digunakan. Terakhir kita akan membuat **service.yaml**

```

1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: performance-monitoring
5 spec:
6   selector:
7     app: performance-monitoring
8   ports:
9     - protocol: TCP
10     port: 5000
11     targetPort: 5000
12 type: LoadBalancer

```

Setelah itu, melakukan perintah berikut

```

1 # Untuk melihat status kubernetes sudah berjalan
2 kubectl --cluster-info
3
4 # Apply konfigurasi kubernetes
5 kubectl apply -f deployment.yaml
6 kubectl apply -f service.yaml

```

Terakhir, kita bisa melakukan cek pada **Pods** dan **Service** untuk memastikan semuanya berjalan dengan baik

```

1 kubectl get pods
2 kubectl get service

```

```
kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
(py310_pc) [2.711s] [RAM: 11/11GB] [Wednesday at 1:15:33 PM] [main = 0 77 -1] [docker-desktop]
[D:\Stuff That I Need to Do\IF4025 - Pervasive Computing\docker-real-time-database]
kubectl apply -f deployment.yaml
deployment.apps/performance-monitoring created
(py310_pc) [1.312s] [RAM: 11/11GB] [Wednesday at 1:22:59 PM] [main = 0 77 -1] [docker-desktop]
[D:\Stuff That I Need to Do\IF4025 - Pervasive Computing\docker-real-time-database]
kubectl apply -f service.yaml
service/performance-monitoring created
```

Figure 7: Kubernetes Command

```
kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
performance-monitoring-64f846c6f-ck4tq  1/1     Running   0           2m46s
performance-monitoring-64f846c6f-g8wnn  1/1     Running   0           2m46s
performance-monitoring-64f846c6f-w4rtb  1/1     Running   0           2m46s
(py310_pc) [3.246s] [RAM: 11/11GB] [Wednesday at 1:25:44 PM] [main = 0 77 -1] [docker-desktop]
[D:\Stuff That I Need to Do\IF4025 - Pervasive Computing\docker-real-time-database]
kubectl get services
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes          ClusterIP     10.96.0.1        <none>         443/TCP          12m
performance-monitoring  LoadBalancer 10.106.54.230   localhost     5808:31870/TCP   3m22s
```

Figure 8: Kubernetes Pods Services

Bisa dilihat pada response pods, semua sistem berjalan dengan baik.

Referensi

1. [Sota Docker Helper](#)
2. [Sota Landing Page Maker](#)
3. [Docker Introduction](#)
4. [Flask App + Docker Medium Article](#)

Link of Images in Docker Hub

Here's the Image in the Docker Hub : [Images](#)