LAPORAN PROYEK KUBERNETES

ECOTRACK – SMART WASTE MONITORING SYSTEM BERBASIS KUBERNETES

Disusun untuk Memenuhi Tugas Mata Kuliah Komputasi Awan

Disusun oleh

15-2023-142 - Irfan Nur Iqbal

Prodi Informatika
Fakultas Teknologi Industri
Institut Teknologi Nasional
Bandung

2026

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi cloud computing dan containerization saat ini memberikan perubahan besar terhadap proses deployment dan pengelolaan aplikasi modern. Salah satu teknologi yang banyak digunakan dalam industri adalah Kubernetes, yaitu platform orkestrasi container yang memungkinkan aplikasi dapat berjalan secara scalable, reliable, serta mudah dikelola dalam berbagai environment.

Dalam era digital saat ini, permasalahan lingkungan terutama pengelolaan sampah menjadi salah satu tantangan yang cukup serius di berbagai daerah. Banyak laporan mengenai sampah yang tidak tertangani dengan cepat sehingga menyebabkan pencemaran lingkungan dan menurunkan kualitas kebersihan kota. Oleh karena itu, diperlukan sebuah sistem monitoring yang mampu membantu proses pelaporan dan pengelolaan sampah secara lebih efektif dan terorganisir.

Berdasarkan permasalahan tersebut, dibuatlah aplikasi bernama EcoTrack – Smart Waste Monitoring System, yaitu aplikasi berbasis web yang digunakan untuk melakukan monitoring laporan sampah secara digital. Aplikasi ini memungkinkan pengguna untuk menambahkan laporan sampah, mengubah status penanganan, serta melakukan monitoring data secara realtime.

Selain itu, proyek ini juga memiliki keterkaitan dengan konsep Sustainable Development Goals (SDGs), khususnya:

1. **SDG 11 – Sustainable Cities and Communities**
   Mendukung terciptanya lingkungan perkotaan yang lebih bersih dan terorganisir.
2. **SDG 12 – Responsible Consumption and Production**
   Mendukung pengelolaan sampah dan monitoring limbah secara lebih baik.

Implementasi sistem dilakukan menggunakan Kubernetes berbasis local cluster dengan bantuan Minikube pada VMware sesuai alternatif platform yang disediakan dalam praktikum. Sistem dibangun menggunakan arsitektur cloud-native dengan implementasi deployment, service, ingress, persistent volume claim, configmap, secret, dan horizontal pod autoscaler.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah pada proyek ini adalah:

1. Bagaimana cara melakukan deployment aplikasi berbasis container menggunakan Kubernetes?
2. Bagaimana cara menghubungkan frontend, backend, dan database dalam Kubernetes cluster?
3. Bagaimana cara menerapkan konsep scalability menggunakan Horizontal Pod Autoscaler?
4. Bagaimana implementasi konsep SDGs pada aplikasi monitoring sampah?
5. Bagaimana penerapan deployment cloud-native menggunakan Kubernetes pada local cluster?

1.3 Tujuan

Tujuan dari proyek ini adalah:

1. Mengimplementasikan aplikasi berbasis Kubernetes.
2. Melakukan deployment frontend, backend, dan database menggunakan container.
3. Menghubungkan service antar container menggunakan Kubernetes Service.

4. Mengimplementasikan Horizontal Pod Autoscaler (HPA).
5. Mengimplementasikan Persistent Volume Claim (PVC) untuk penyimpanan database.
6. Mengembangkan aplikasi yang mendukung Sustainable Development Goals (SDGs).

1.4 Manfaat

Manfaat dari proyek ini adalah:

1. Membantu proses monitoring dan pengelolaan laporan sampah.
2. Memberikan pemahaman mengenai implementasi Kubernetes.
3. Memberikan pengalaman deployment aplikasi berbasis cloud-native.
4. Meningkatkan pemahaman mengenai container orchestration menggunakan Kubernetes.

BAB II

LANDASAN TEORI

2.1 Kubernetes

Kubernetes merupakan platform open-source yang digunakan untuk mengelola container secara otomatis. Kubernetes menyediakan fitur deployment, scaling, self-healing, dan service discovery sehingga aplikasi dapat berjalan dengan lebih stabil dan scalable.

Dalam proyek ini Kubernetes digunakan untuk menjalankan aplikasi frontend, backend, dan database dalam bentuk pod yang saling terhubung melalui service.

2.2 Docker

Docker merupakan platform containerization yang digunakan untuk membangun image aplikasi. Dengan Docker, aplikasi dapat dijalankan secara konsisten pada berbagai environment tanpa perlu melakukan konfigurasi ulang.

Pada proyek ini Docker digunakan untuk membangun image frontend React dan backend Node.js.

2.3 Minikube

Minikube merupakan local Kubernetes cluster yang digunakan untuk simulasi deployment Kubernetes pada lingkungan lokal. Minikube mempermudah proses pembelajaran Kubernetes tanpa harus menggunakan cloud provider secara langsung.

Implementasi proyek dilakukan menggunakan Minikube pada virtual machine VMware.

2.4 Sustainable Development Goals (SDGs)

Sustainable Development Goals (SDGs) merupakan program pembangunan berkelanjutan yang dibuat oleh United Nations.

Aplikasi EcoTrack mendukung:

- SDG 11 (Sustainable Cities and Communities)
- SDG 12 (Responsible Consumption and Production)

melalui sistem monitoring dan pengelolaan laporan sampah berbasis digital.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Deskripsi Sistem

EcoTrack merupakan aplikasi monitoring sampah berbasis web yang digunakan untuk mencatat dan memonitor laporan sampah dari pengguna.

Fitur utama aplikasi:

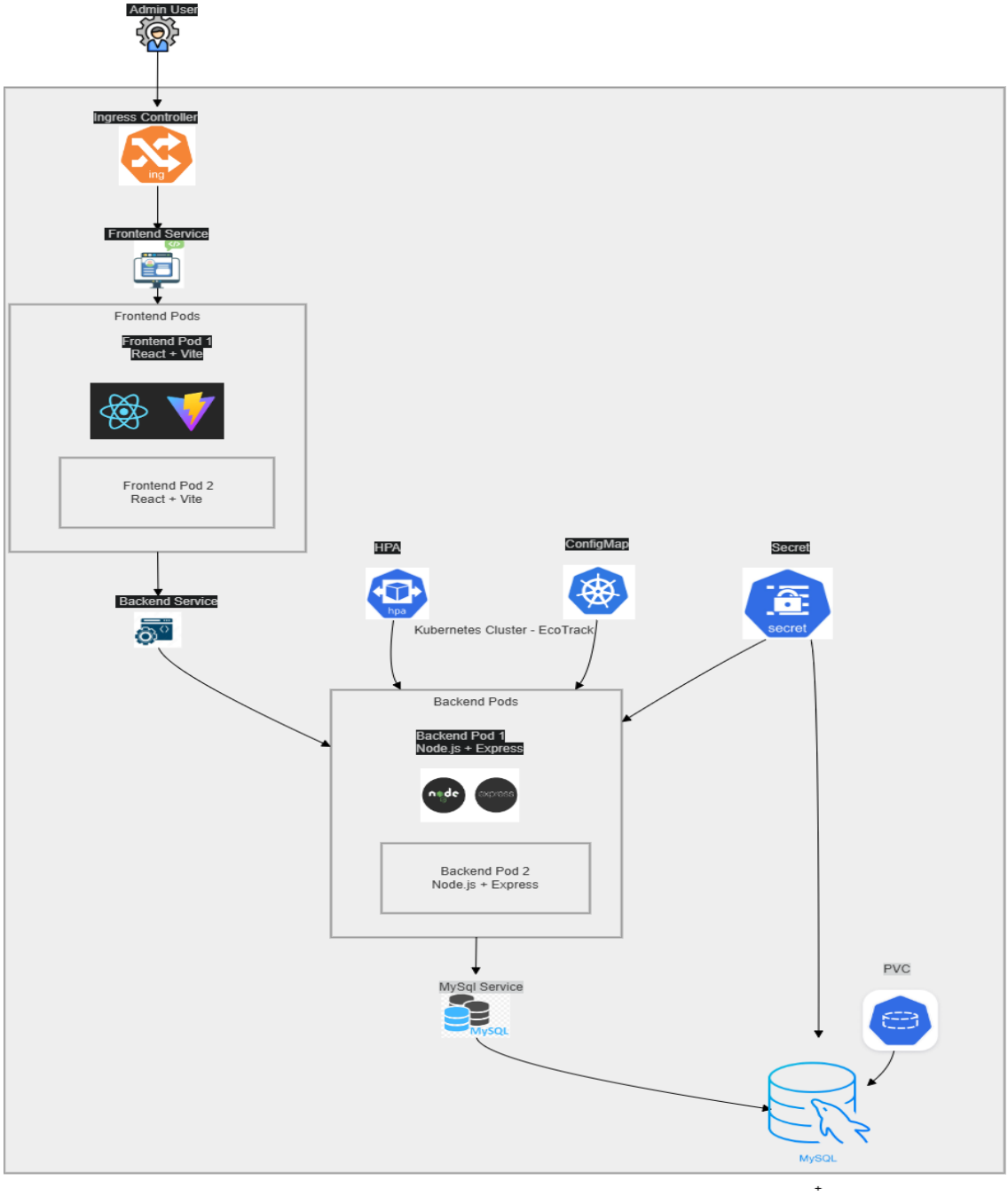
1. Menambahkan laporan sampah
2. Mengubah status laporan
3. Menghapus laporan
4. Monitoring data laporan

Sistem dibangun menggunakan:

- Frontend: React + Vite
- Backend: Node.js Express
- Database: MySQL
- Containerization: Docker
- Orchestration: Kubernetes

3.2 Arsitektur Sistem

Dibawah ini merupakan gambar Arsistektur sistem:



Penjelasan Gambar:

Pada arsitektur sistem, pengguna mengakses aplikasi melalui frontend service. Frontend kemudian berkomunikasi dengan backend API yang berjalan pada deployment terpisah. Backend melakukan komunikasi dengan database MySQL untuk menyimpan dan mengambil data laporan sampah.

Kubernetes Service digunakan untuk menghubungkan komunikasi antar pod. Sedangkan ingress digunakan untuk mengatur routing trafik pada cluster.

3.3 Struktur Kubernetes

Komponen Kubernetes yang digunakan pada proyek ini:

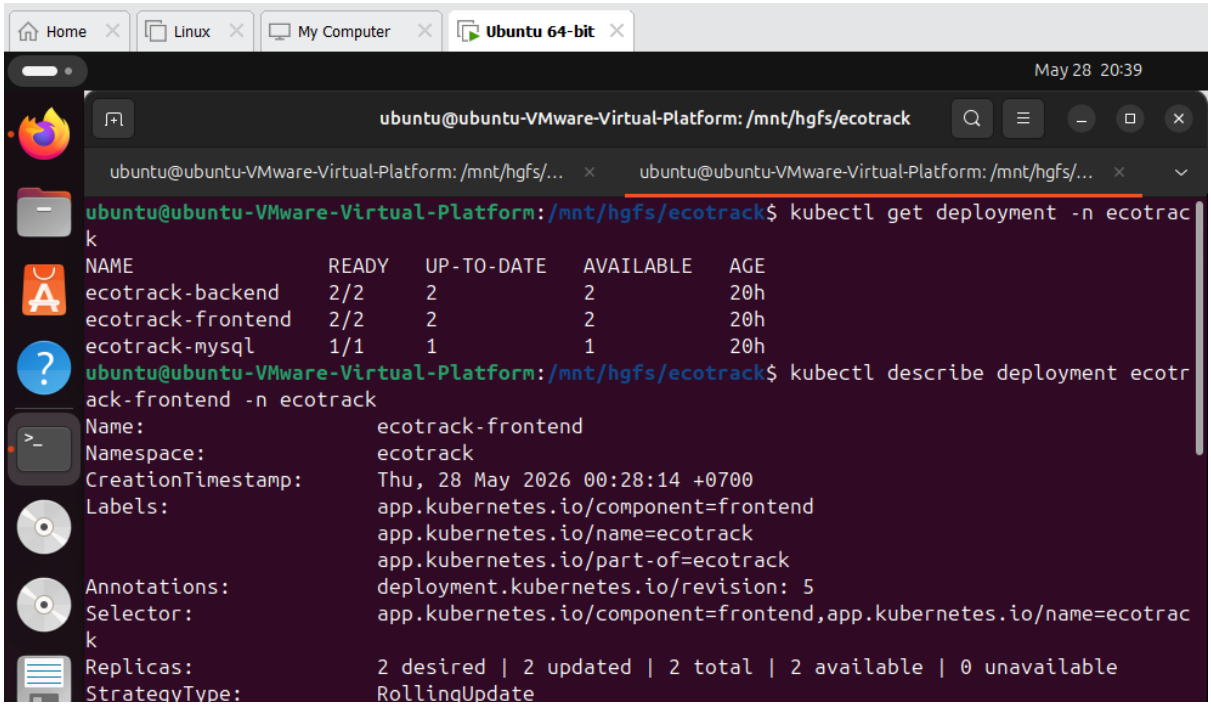
Komponen	Fungsi
Namespace	Memisahkan resource aplikasi EcoTrack di dalam cluster Kubernetes agar lebih terorganisir
Deployment	Mengelola dan menjalankan pod aplikasi secara otomatis
Service	Menghubungkan komunikasi antar pod dan menyediakan akses jaringan internal
Ingress	Mengatur routing trafik dari user menuju service aplikasi
ConfigMap	Menyimpan konfigurasi aplikasi non-sensitif
Secret	Menyimpan data sensitif seperti username dan password database
PVC (Persistent Volume Claim)	Menyediakan penyimpanan persistent untuk database MySQL
HPA (Horizontal Pod Autoscaler)	Melakukan auto scaling jumlah pod berdasarkan penggunaan resource

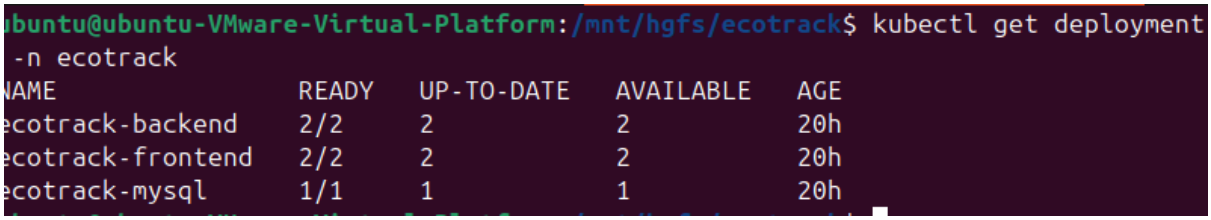
BAB IV

IMPLEMENTASI DAN KONFIGURASI

4.1 Deployment Frontend

Frontend dibangun menggunakan React dan Vite kemudian di-containerize menggunakan Docker.





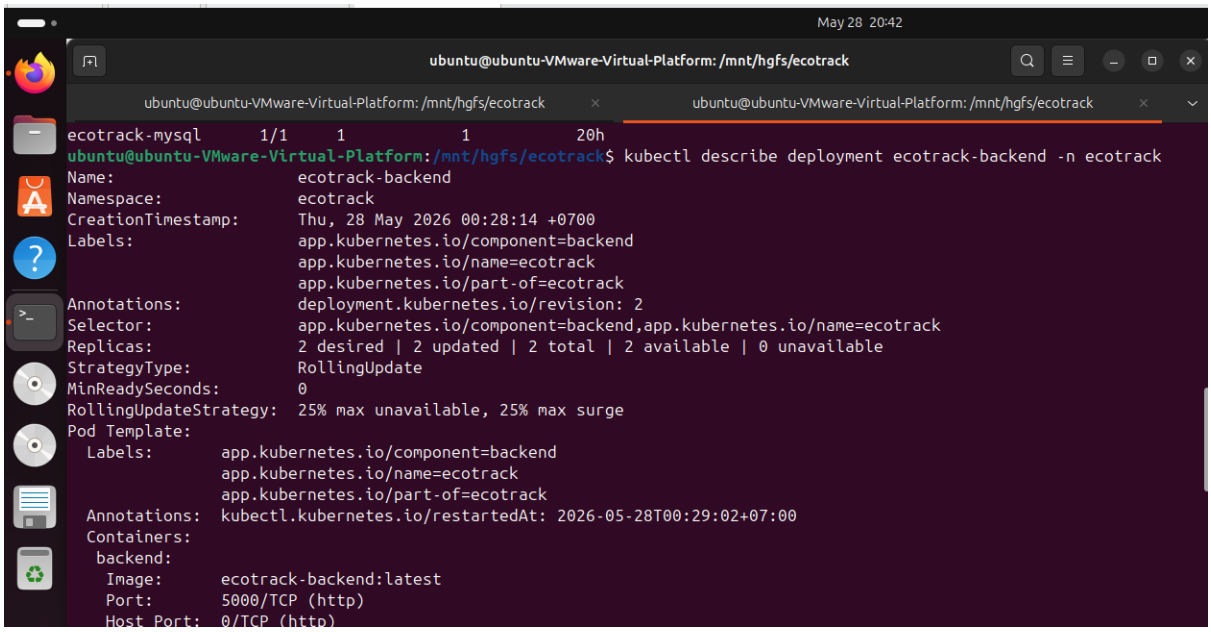
Frontend menggunakan deployment dengan beberapa replica untuk mendukung high availability.

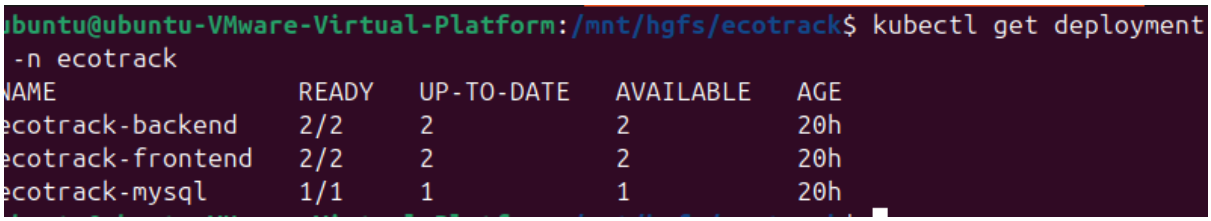
Penjelasan:

- Menggunakan image Docker frontend
- Menggunakan replica deployment
- Port aplikasi berjalan pada port 80
- Menggunakan Kubernetes Service

4.2 Deployment Backend

Backend menggunakan Node.js Express yang bertugas menangani API dan komunikasi database.



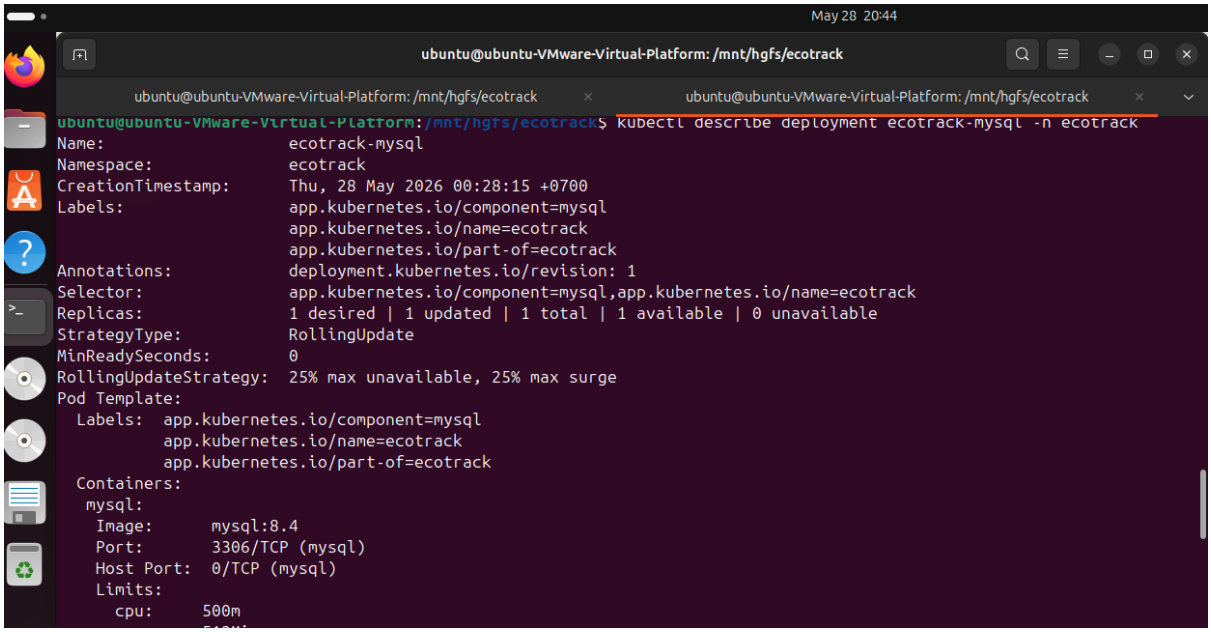


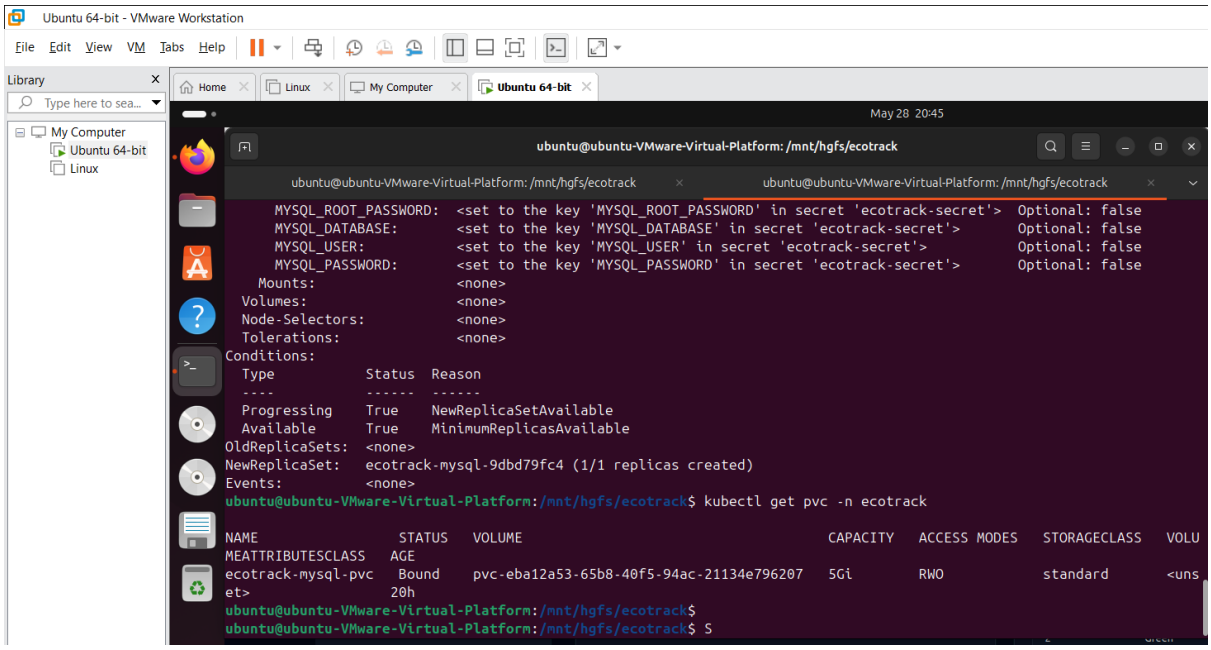
Penjelasan:

- Backend berjalan pada port 5000
- Menggunakan ConfigMap dan Secret
- Memiliki beberapa replica pod
- Menggunakan service internal Kubernetes

4.3 Deployment Database MySQL

MySQL digunakan sebagai database utama aplikasi.





Penjelasan:

- Menggunakan Persistent Volume Claim
- Data tetap tersimpan walaupun pod restart
- Menggunakan service internal Kubernetes

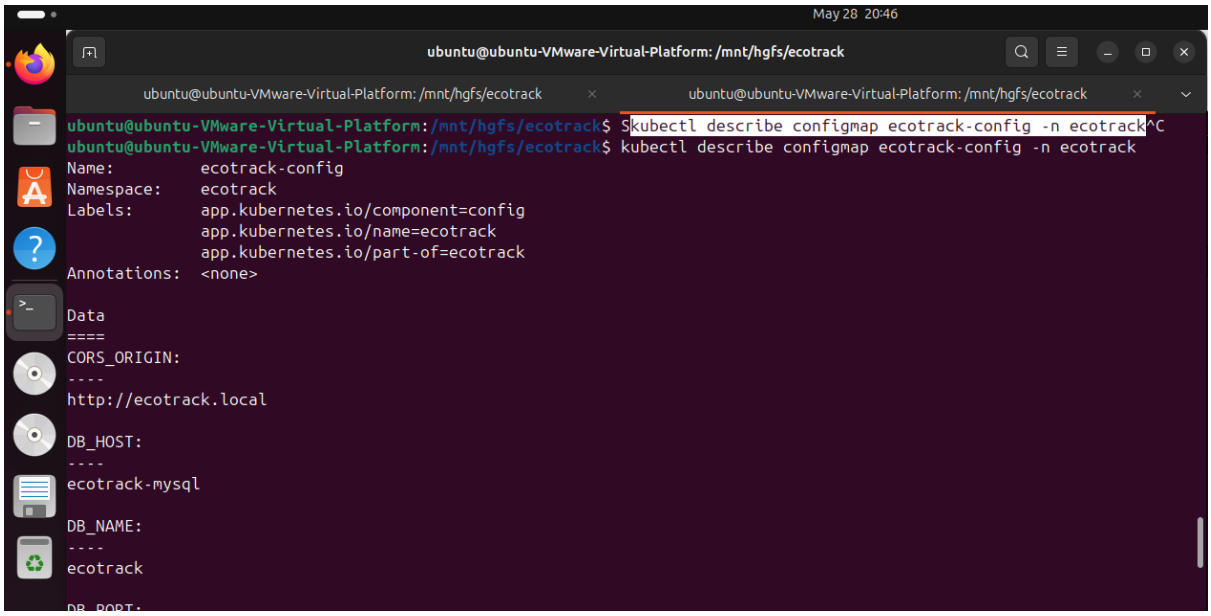
Gambar menunjukkan Persistent Volume Claim berhasil dibuat dan terhubung dengan deployment MySQL untuk menjaga persistensi data aplikasi.

4.4 ConfigMap dan Secret

ConfigMap digunakan untuk menyimpan konfigurasi aplikasi seperti host database dan port aplikasi.

Sedangkan Secret digunakan untuk menyimpan username dan password database.

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get secret -n ecotrack
NAME              TYPE     DATA   AGE
ecotrack-secret   Opaque   6      20h
```




```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get secret -n ecotrack
NAME              TYPE     DATA   AGE
ecotrack-secret   Opaque   6      20h
```


```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl describe secret ecotrack-secret -n ecotrack
Name:         ecotrack-secret
Namespace:    ecotrack
Labels:       app.kubernetes.io/component=secret
              app.kubernetes.io/name=ecotrack
              app.kubernetes.io/part-of=ecotrack
Annotations:  <none>

Type:  Opaque

Data
====
DB_PASSWORD:          17 bytes
DB_USER:              13 bytes
MYSQL_DATABASE:       8 bytes
MYSQL_PASSWORD:       17 bytes
MYSQL_ROOT_PASSWORD:  12 bytes
```

4.5 Ingress

Ingress digunakan untuk mengatur routing trafik dari user menuju service frontend dan backend pada Kubernetes cluster.

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get ingress -n ecotrack
NAME              CLASS   HOSTS             ADDRESS   PORTS   AGE
ecotrack-ingress  nginx   ecotrack.local              80      20h
```

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl describe ingress ecotrack-ingress -n ecotrack
Name:             ecotrack-ingress
Labels:           <none>
Namespace:        ecotrack
Address:
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host           Path  Backends
  ----           ----  --------
  ecotrack.local
                 /api   ecotrack-backend:80 (10.244.0.23:5000,10.244.0.18:5000)
                 /      ecotrack-frontend:80 (10.244.0.19:80,10.244.0.20:80)
Annotations:      <none>
Events:           <none>
```

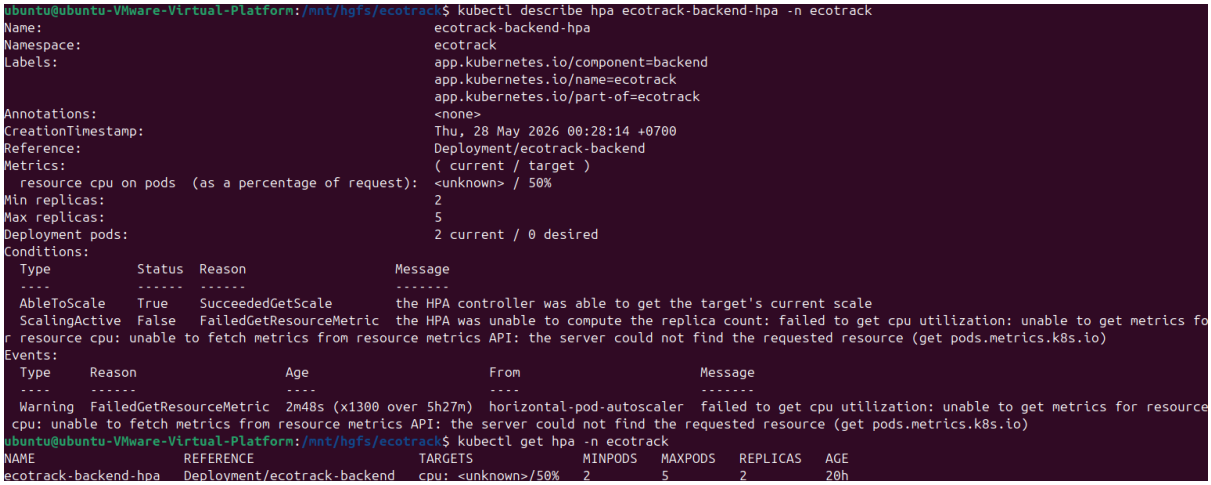
ingress.yaml | .gitignore | backend-deployment.yaml | Quokka | Console Ninja

ecotrack > k8s > ingress.yaml

```yaml
# Ingress routes browser traffic to frontend and backend services.
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ecotrack-ingress
  namespace: ecotrack
spec:
  ingressClassName: nginx
  rules:
    - host: ecotrack.local
      http:
        paths:
          - path: /api
            pathType: Prefix
            backend:
              service:
                name: ecotrack-backend
                port:
                  number: 80
          - path: /
            pathType: Prefix
            backend:
              service:
                name: ecotrack-frontend
                port:
                  number: 80
```

4.6 Horizontal Pod Autoscaler (HPA)

Horizontal Pod Autoscaler digunakan untuk melakukan auto scaling pod backend ketika penggunaan resource meningkat.



Penjelasan:

- Minimum pod: 2
- Maximum pod: 5
- Target CPU utilization: 50%

BAB V

PENGUJIAN SISTEM

5.1 Pengujian Deployment

Pengujian deployment dilakukan menggunakan perintah:

kubectl get pods -n ecotrack

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get pods -n ecotrack
NAME                                READY   STATUS    RESTARTS        AGE
ecotrack-backend-cc5449bf5-msck4    1/1     Running   5 (5h29m ago)   20h
ecotrack-backend-cc5449bf5-nbshj    1/1     Running   3 (5h29m ago)   20h
ecotrack-frontend-57c9999574-5fdn5  1/1     Running   1 (5h29m ago)   19h
ecotrack-frontend-57c9999574-hvlwd  1/1     Running   1 (5h29m ago)   19h
ecotrack-mysql-9dbd79fc4-hbvq4      1/1     Running   1 (5h29m ago)   20h
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$
```

Hasil:

Seluruh pod frontend, backend, dan mysql berhasil berjalan dengan status Running.

5.2 Pengujian Service

Pengujian service dilakukan menggunakan:

kubectl get svc -n ecotrack

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get svc -n ecotrack
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
ecotrack-backend    ClusterIP   10.104.23.25    <none>        80/TCP     20h
ecotrack-frontend   ClusterIP   10.96.115.19    <none>        80/TCP     20h
ecotrack-mysql      ClusterIP   10.98.133.234   <none>        3306/TCP   20h
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$
```

Hasil:

Service frontend, backend, dan mysql berhasil dibuat dan saling terhubung.

5.3 Pengujian HPA

Pengujian HPA dilakukan menggunakan:

kubectl get hpa -n ecotrack

```
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$ kubectl get hpa -n ecotrack
NAME                 REFERENCE                     TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
ecotrack-backend-hpa   Deployment/ecotrack-backend   cpu: <unknown>/50%   2         5         2          20h
ubuntu@ubuntu-VMware-Virtual-Platform:/mnt/hgfs/ecotrack$
```

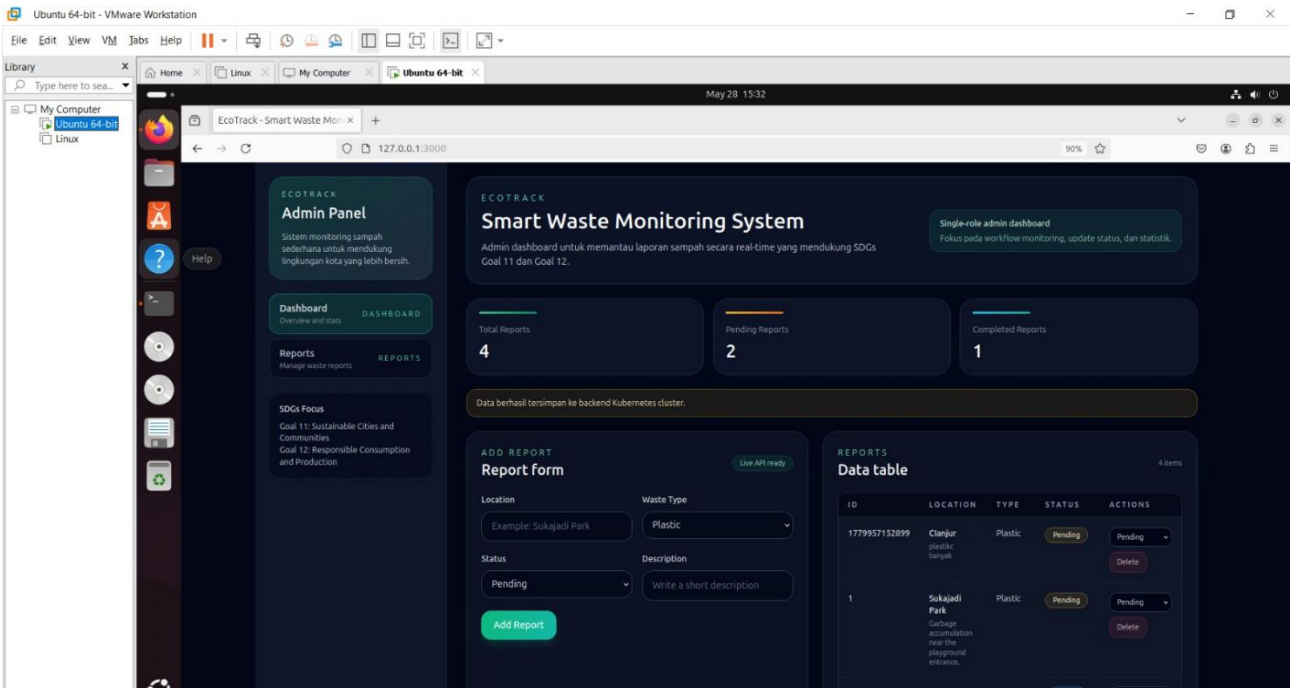
Hasil:

HPA digunakan untuk meningkatkan scalability aplikasi dengan melakukan auto scaling pod berdasarkan penggunaan CPU.

5.4 Pengujian Aplikasi

Pengujian aplikasi dilakukan dengan mencoba:

1. Menambahkan laporan
2. Mengubah status laporan
3. Menghapus laporan



Hasil:

Seluruh fitur CRUD berhasil berjalan dengan baik.

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, aplikasi EcoTrack berhasil di-deploy menggunakan Kubernetes cluster berbasis Minikube.

Sistem berhasil menerapkan:

- Deployment
- Service
- Ingress
- ConfigMap
- Secret
- Persistent Volume Claim
- Horizontal Pod Autoscaler

Aplikasi juga berhasil mendukung Sustainable Development Goals (SDGs), khususnya SDG 11 dan SDG 12 melalui sistem monitoring pengelolaan sampah berbasis digital.

DAFTAR PUSTAKA

1. Kubernetes Documentation – https://kubernetes.io
2. Docker Documentation – https://docs.docker.com
3. Minikube Documentation – https://minikube.sigs.k8s.io/docs/
4. United Nations SDGs – https://sdgs.un.org/goals

Link Github: https://github.com/Irfannuriqbal/ecotrack-kubernetes