LAPORAN

SISTEM TERDISTRIBUSI DAN TERDESENTRALISASI

Disusun Oleh:

215410133

JUNSO SUAT

PROGRAM STUDI INFORMATIKA

PROGRAM SERJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKOLOGI DIGITAL INDONESIA

YOGYAKARTA

2026

TUGAS REMEDIAL

IMPLEMENTASI REPLIKASI DATA MENGGUNAKAN CLOUD FIRESTORE

A. TUJUAN PEMBELAJARAN

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

1. Menjelaskan konsep replikasi data, data consistency, dan replication delay dalam sistem terdistribusi.
2. Menggunakan Cloud Firestore sebagai database terdistribusi untuk replikasi data real-time.
3. Mengimplementasikan dua node pembaca (reader) untuk mengamati propagasi update data.
4. Mengukur dan menganalisis selisih waktu pembaruan data antar node pembaca.

B. KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Laptop/PC dengan koneksi internet.
2. Akun Google aktif untuk akses Firebase.
3. Node.js LTS
4. Firebase CLI (npm install -g firebase-tools)
5. Visual Studio Code / editor lain
6. Browser untuk Firebase Console

C. DASAR TEORI

1. Sistem Terdistribusi Sistem terdistribusi terdiri dari banyak node yang berkomunikasi melalui jaringan. Isu penting di dalamnya meliputi:
   - Replikasi data
   - Konsistensi
   - Latensi
   - Fault tolerance
2. Replikasi Data Replikasi data adalah proses menggandakan data ke beberapa server/region untuk:
   - Mempercepat operasi read
   - Menjaga ketersediaan data (availability)
   - Mengurangi risiko kehilangan data Namun replikasi memunculkan fenomena:
   - Replication delay → waktu propagasi update antar node
   - Perbedaan versi data (inconsistency sementara)
3. Cloud Firestore Cloud Firestore adalah database NoSQL dari Firebase dengan fitur:
   - Replikasi multi-region otomatis
   - Real-time listener melalui onSnapshot()
   - Strong consistency pada pembacaan dokumen dari server

- Eventual consistency ketika menggunakan mode offline atau cache Firestore cocok digunakan untuk simulasi sistem terdistribusi karena menyediakan:
- Update data real-time
- Mekanisme propagasi yang dapat diamati
- Latensi antar node yang nyata

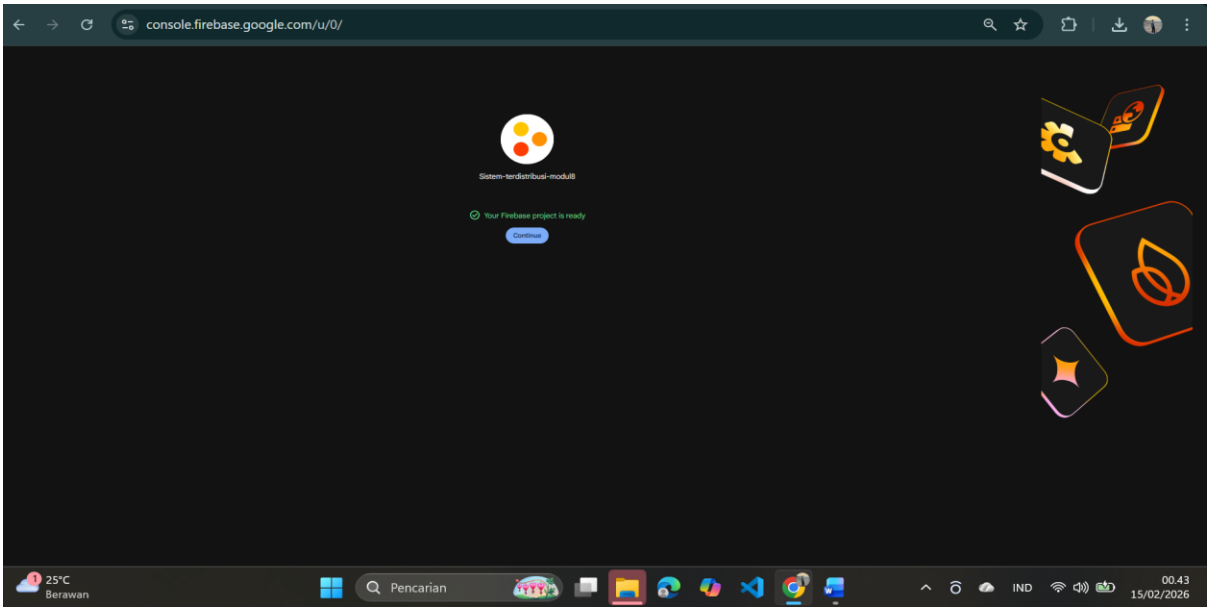
4. Data Consistency Dua jenis konsistensi:
   - Strong consistency: pembacaan mendapatkan nilai paling baru
   - Eventual consistency: pembacaan bisa tertinggal, tapi akhirnya konsisten Firestore memadukan dua model tersebut.
5. CAP Theorem Dalam kondisi normal Firestore cenderung memberikan:
   - C (Consistency) + A (Availability) Saat terjadi partisi jaringan (client offline), Firestore:
   - Tetap memberikan data dari cache
   - Konsistensi tidak terjamin
   - Mengarah pada AP (Availability + Partition)

D. PRAKTIK

1. Persiapan Firebase Project 1) Buka Firebase Console
   - Buka browser (Chrome/Edge/Firefox).
   - Kunjungi alamat: https://console.firebase.google.com/
   - Login menggunakan akun Google Anda.
   - Setelah masuk, Anda akan melihat daftar proyek yang sudah ada.

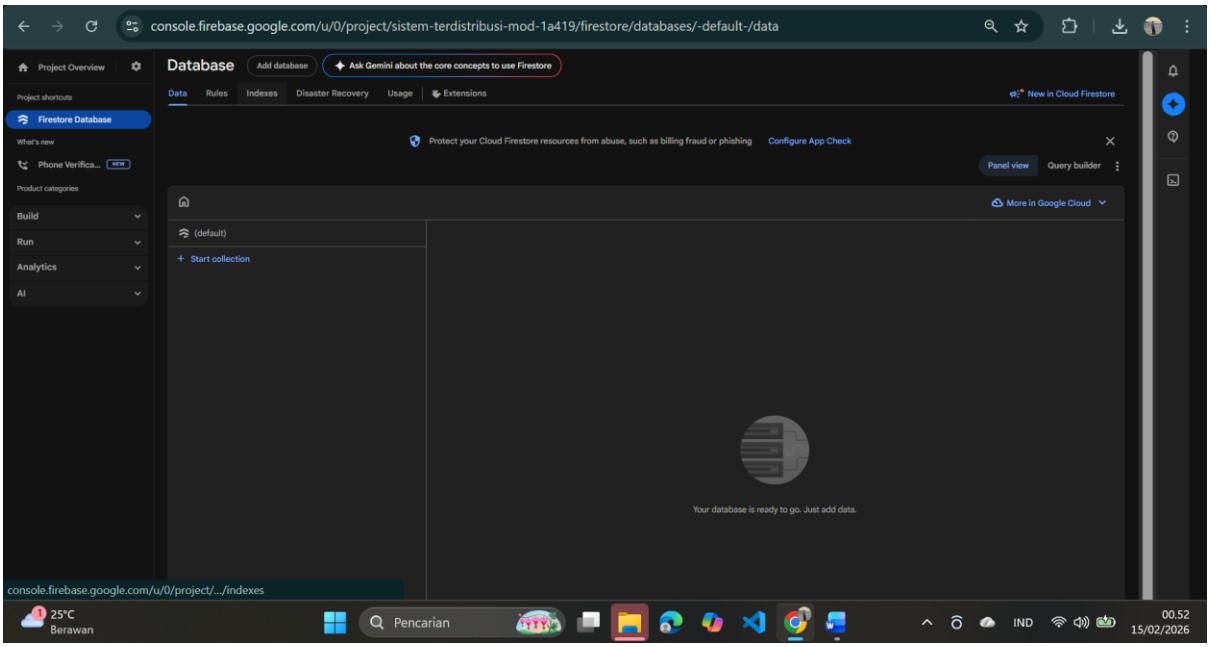
2) Buat Proyek Baru
   - Klik tombol “Add project” atau “+ Create a project”.
   - Masukkan nama proyek, misalnya: Sistem-terdistribusi-modul8
   - Klik Continue.
   - Pada bagian Google Analytics:

- Matikan (recommended untuk praktikum), dengan hilangkan tanda checkbox pada Enable Google Analytics for this project.
- Klik Create project.
   - Tunggu proses inisialisasi sampai muncul pesan: Your project is ready
   - Klik Continue.



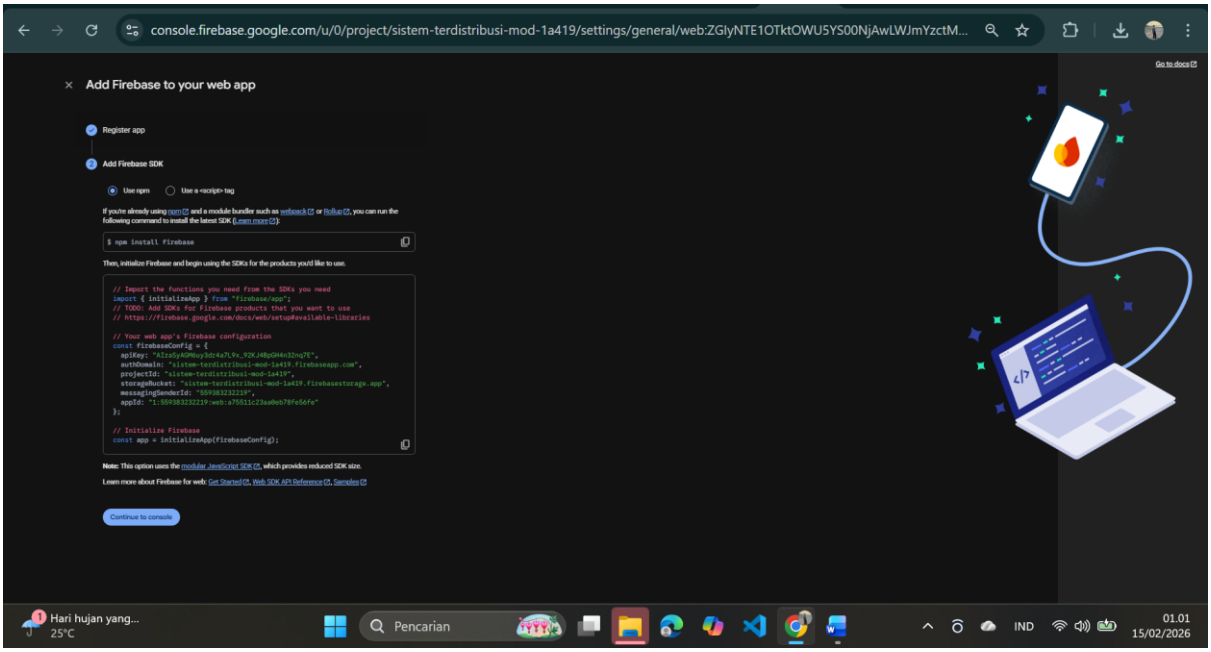
3) Aktifkan Cloud Firestore

- Pada dashboard proyek Firebase, di menu sebelah kiri pilih: Build → Firestore Database
- Klik tombol Create database.
- Muncul pilihan Edition:
  - Pilih Stardart Edition.
- Klik Next.
- Pilih lokasi database: Pilih region default seperti: asia-southeast2 (Jakarta).
- Klik Next.
- Muncul pilihan Configure: Pilih Start in test mode.
- Klik Create.
- Firestore akan membuat instance database (memakan waktu beberapa detik)



4) Tambahkan Firebase App (Untuk Web / Node.js)

- Pada menu Project Overview, pilih submenu Project settings, lalu klik ikon Web () pada bagian bawah, untuk menambahkan aplikasi web baru.
- Beri nama aplikasi pada App nickname, misalnya: modul8-client
- Klik Register app.
- Akan muncul Firebase SDK setup and configuration.



5) Salin Konfigurasi Firebase (firebaseConfig)

- Firebase menampilkan bagian kode seperti:

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyAGM6uy3dr4a7L9x_92KJ4BpGH4n32nq7E",
  authDomain: "sistem-terdistribusi-mod-1a419.firebaseapp.com",
  projectId: "sistem-terdistribusi-mod-1a419",
  storageBucket: "sistem-terdistribusi-mod-1a419.firebasestorage.app",
  messagingSenderId: "559383232219",
  appId: "1:559383232219:web:a75511c23aa0eb78fe56fe"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

6) Klik ikon copy, atau salin manual seluruh objek firebaseConfig.

7) Simpan konfigurasi ini, misalnya ke: Catatan praktikum

8) Klik Continue to console.

2. Buat Folder Proyek Bagian ini bertujuan untuk menyiapkan lingkungan proyek Node.js yang akan digunakan untuk menjalankan writer.js, readerA.js, dan readerB.js.

1) Buka Terminal/Command Prompt

2) Buat folder baru Proyek: mkdir modul8-firestore

3) Masuk ke dalam folder Proyek: cd modul8-firestore

4) Inisialisasi Proyek Node.js Jalankan: npm init

- y Fungsi perintah ini: Membuat file package.json secara otomatis, yang akan muncul pada folder: modul8-firestore
- Menggunakan opsi -y untuk menyetujui semua konfigurasi default
- Menyiapkan struktur untuk mengelola dependensi Node.js

```
C:\Users\User\modul8-firestore>mkdir modul8-firestore

C:\Users\User\modul8-firestore>cd modul8-firestore

C:\Users\User\modul8-firestore\modul8-firestore>npm init -y
Wrote to C:\Users\User\modul8-firestore\modul8-firestore\package.json:

{
  "name": "modul8-firestore",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



C:\Users\User\modul8-firestore\modul8-firestore>
```

5) Instal Library Firebase (Firestore SDK) Jalankan: npm install firebase Perintah ini akan:

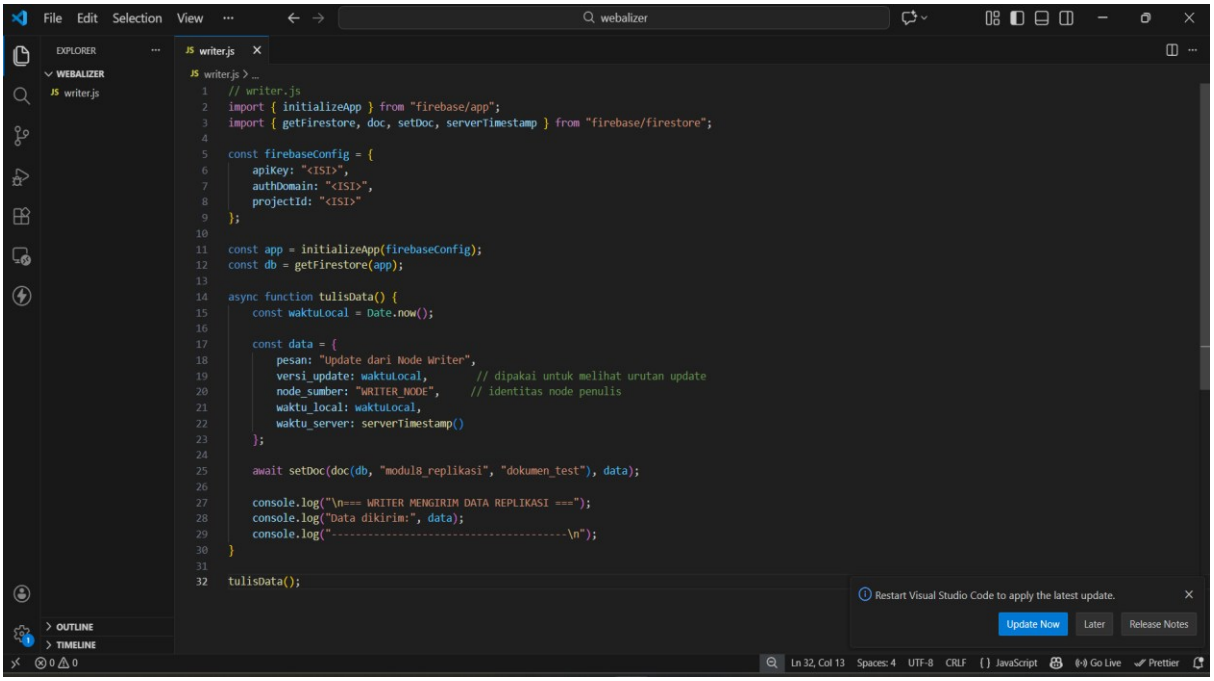
- Mengunduh dan memasang library Firebase JavaScript SDK
- Menambahkan dependensi firebase ke dalam file package.json
- Membuat folder node_modules/ dan file package-lock.json Setelah selesai, struktur folder menjadi:

```
3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\User\modul8-firestore\modul8-firestore>type package.json
{
  "name": "modul8-firestore",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "firebase": "^12.9.0"
  }
}

C:\Users\User\modul8-firestore\modul8-firestore>npm fund
modul8-firestore@1.0.0
+-- https://github.com/sponsors/feross
|   `-- safe-buffer@5.2.1
`-- https://github.com/chalk/wrap-ansi?sponsor=1
  | `-- wrap-ansi@7.0.0
  `-- https://github.com/chalk/ansi-styles?sponsor=1
      `-- ansi-styles@4.3.0


C:\Users\User\modul8-firestore\modul8-firestore>
```

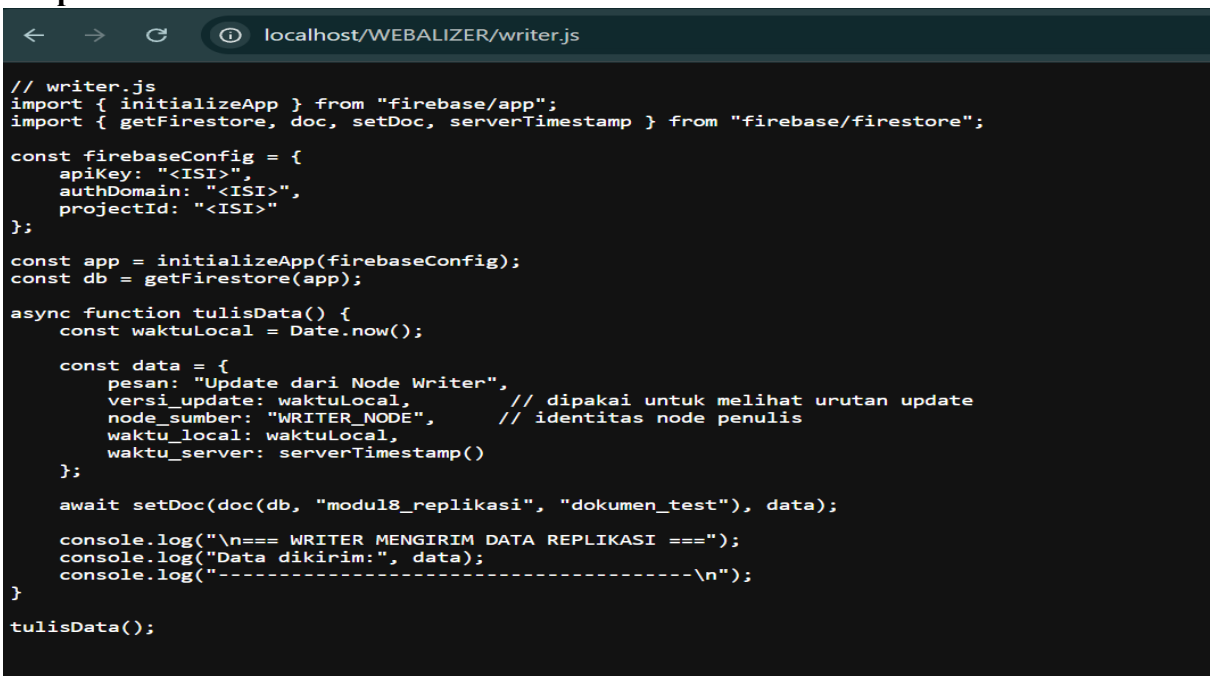
6) Tambahkan tipe modul ES Pada file package.json, tambahkan: "type": "module"
Contoh hasil akhir:

```
C:\Users\User\modul8-firestore\modul8-firestore>type package.json
{
  "name": "modul8-firestore",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "firebase": "^12.9.0"
  }
}
```

2. **File 1: writer.js (Node Penulis / Sumber Replikasi) Node penulis yang mem-publish update.**

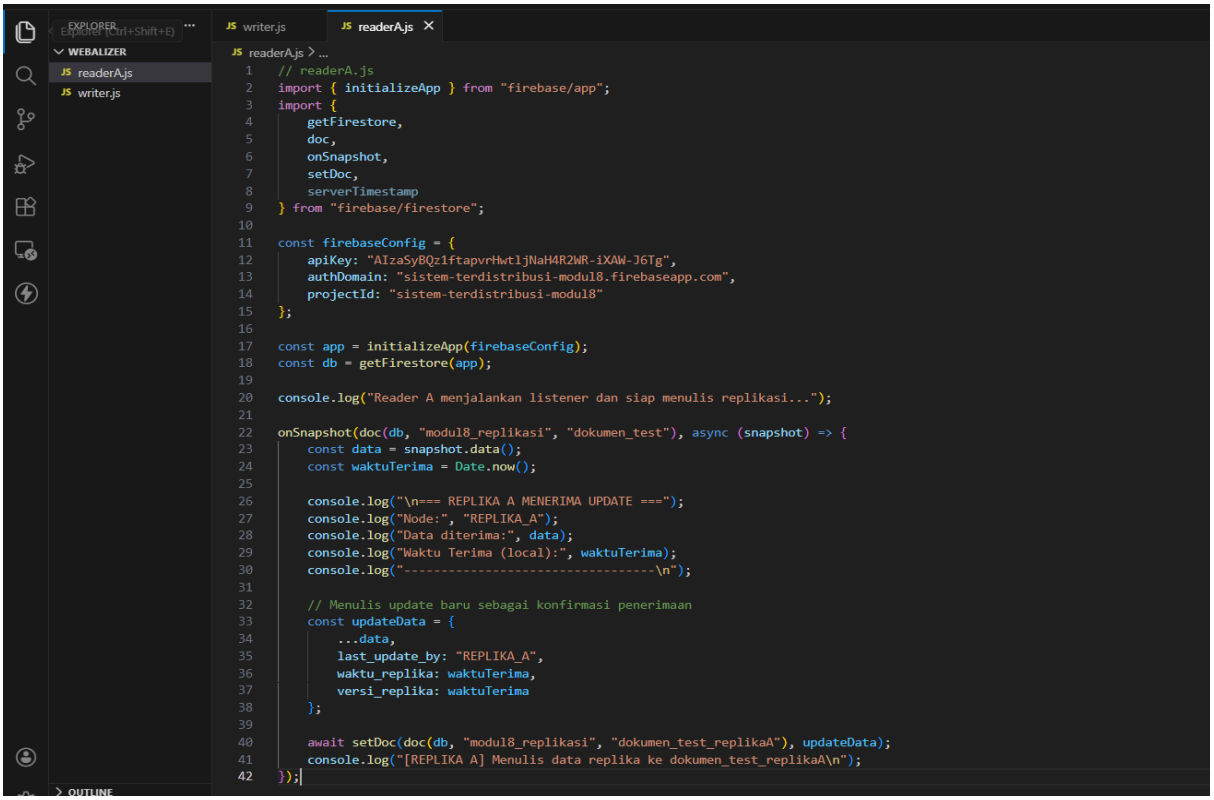


Output:



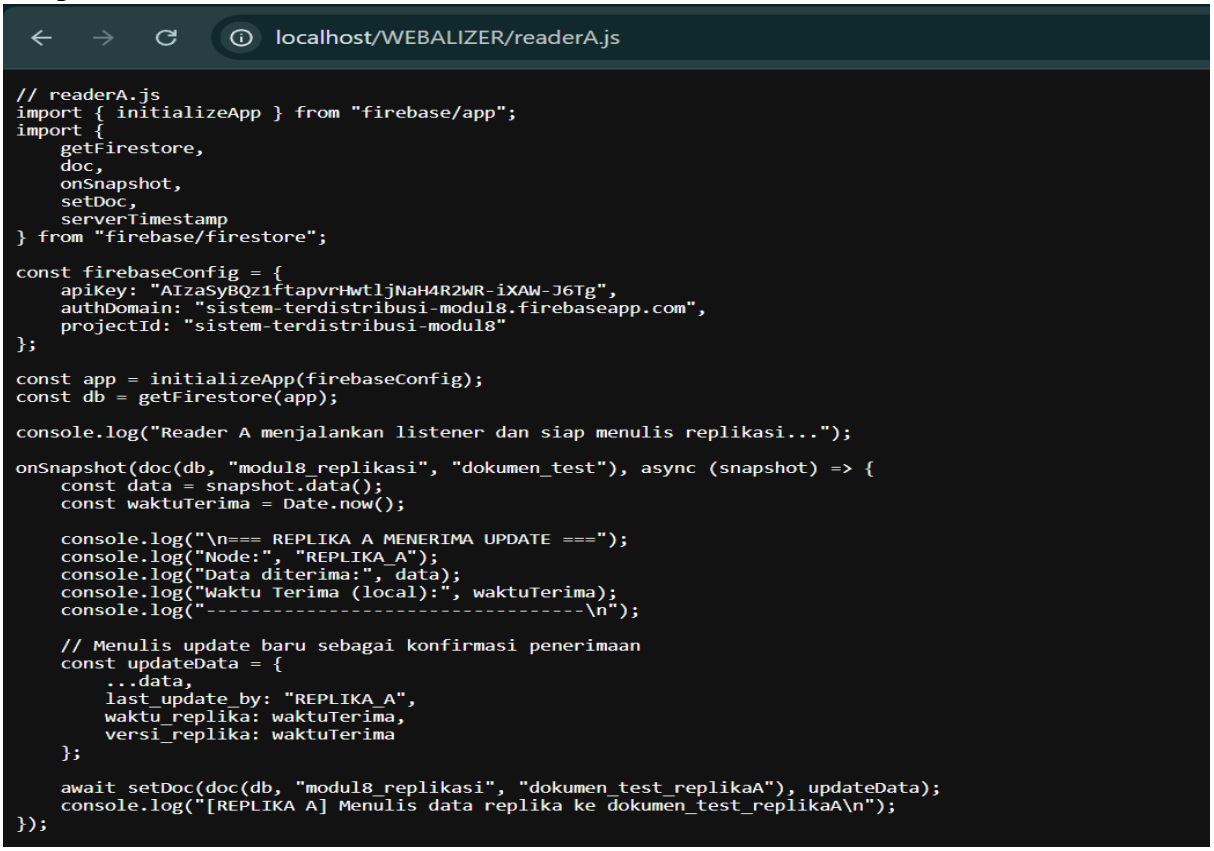
Skrip writer.js ini berfungsi sebagai node penulis yang mengirimkan data ke Cloud Firestore; dimulai dengan inisialisasi aplikasi Firebase dan akses ke Firestore, lalu fungsi tulisData() membuat data berisi pesan, timestamp lokal sebagai versi update, identitas sumber penulis, serta waktu server, kemudian setDoc() menuliskan data tersebut ke dokumen modul8_replikasi/dokumen_test sehingga dapat direplikasi secara real-time ke node pembaca, dan akhirnya menampilkan log bahwa data berhasil dikirim.

3. **File 2: readerA.js (Node Pembaca A / Replica A) Node replika yang menerima update secara real-time.**



```
// readerA.js
import { initializeApp } from "firebase/app";
import {
    getFirestore,
    doc,
    onSnapshot,
    setDoc,
    serverTimestamp
} from "firebase/firestore";

const firebaseConfig = {
    apiKey: "AIzaSyBQz1ftapvrHwtljNaH4R2WR-iXAW-J6Tg",
    authDomain: "sistem-terdistribusi-modul8.firebaseapp.com",
    projectId: "sistem-terdistribusi-modul8"
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

console.log("Reader A menjalankan listener dan siap menulis replikasi...");

onSnapshot(doc(db, "modul8_replikasi", "dokumen_test"), async (snapshot) => {
    const data = snapshot.data();
    const waktuTerima = Date.now();

    console.log("\n=== REPLIKA A MENERIMA UPDATE ===");
    console.log("Node:", "REPLIKA_A");
    console.log("Data diterima:", data);
    console.log("Waktu Terima (local):", waktuTerima);
    console.log("----------------------------------\n");

    // Menulis update baru sebagai konfirmasi penerimaan
    const updateData = {
        ...data,
        last_update_by: "REPLIKA_A",
        waktu_replika: waktuTerima,
        versi_replika: waktuTerima
    };

    await setDoc(doc(db, "modul8_replikasi", "dokumen_test_replikaA"), updateData);
    console.log("[REPLIKA A] Menulis data replika ke dokumen_test_replikaA\n");
});
```

Output:

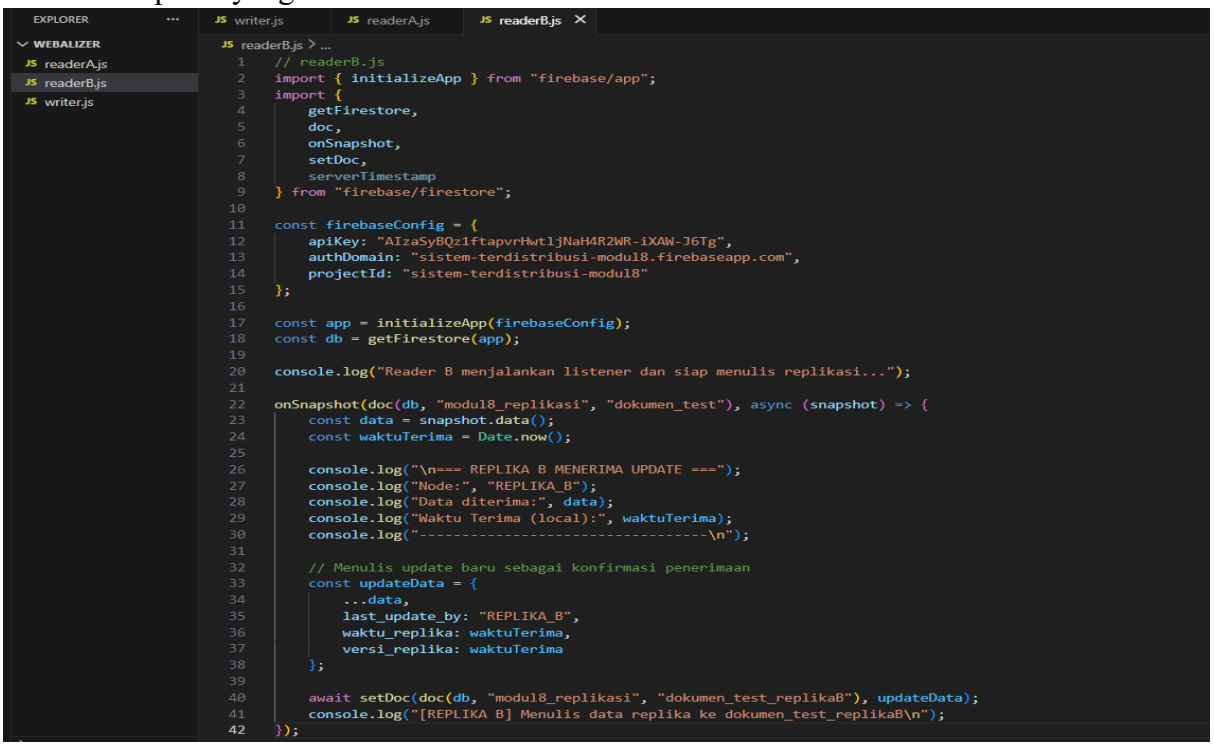


```
localhost/WEBALIZER/readerA.js

// readerA.js
import { initializeApp } from "firebase/app";
import {
    getFirestore,
    doc,
    onSnapshot,
    setDoc,
    serverTimestamp
} from "firebase/firestore";

const firebaseConfig = {
    apiKey: "AIzaSyBQz1ftapvrHwtljNaH4R2WR-iXAW-J6Tg",
    authDomain: "sistem-terdistribusi-modul8.firebaseapp.com",
    projectId: "sistem-terdistribusi-modul8"
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

console.log("Reader A menjalankan listener dan siap menulis replikasi...");

onSnapshot(doc(db, "modul8_replikasi", "dokumen_test"), async (snapshot) => {
    const data = snapshot.data();
    const waktuTerima = Date.now();

    console.log("\n=== REPLIKA A MENERIMA UPDATE ===");
    console.log("Node:", "REPLIKA_A");
    console.log("Data diterima:", data);
    console.log("Waktu Terima (local):", waktuTerima);
    console.log("----------------------------------\n");

    // Menulis update baru sebagai konfirmasi penerimaan
    const updateData = {
        ...data,
        last_update_by: "REPLIKA_A",
        waktu_replika: waktuTerima,
        versi_replika: waktuTerima
    };

    await setDoc(doc(db, "modul8_replikasi", "dokumen_test_replikaA"), updateData);
    console.log("[REPLIKA A] Menulis data replika ke dokumen_test_replikaA\n");
});
```

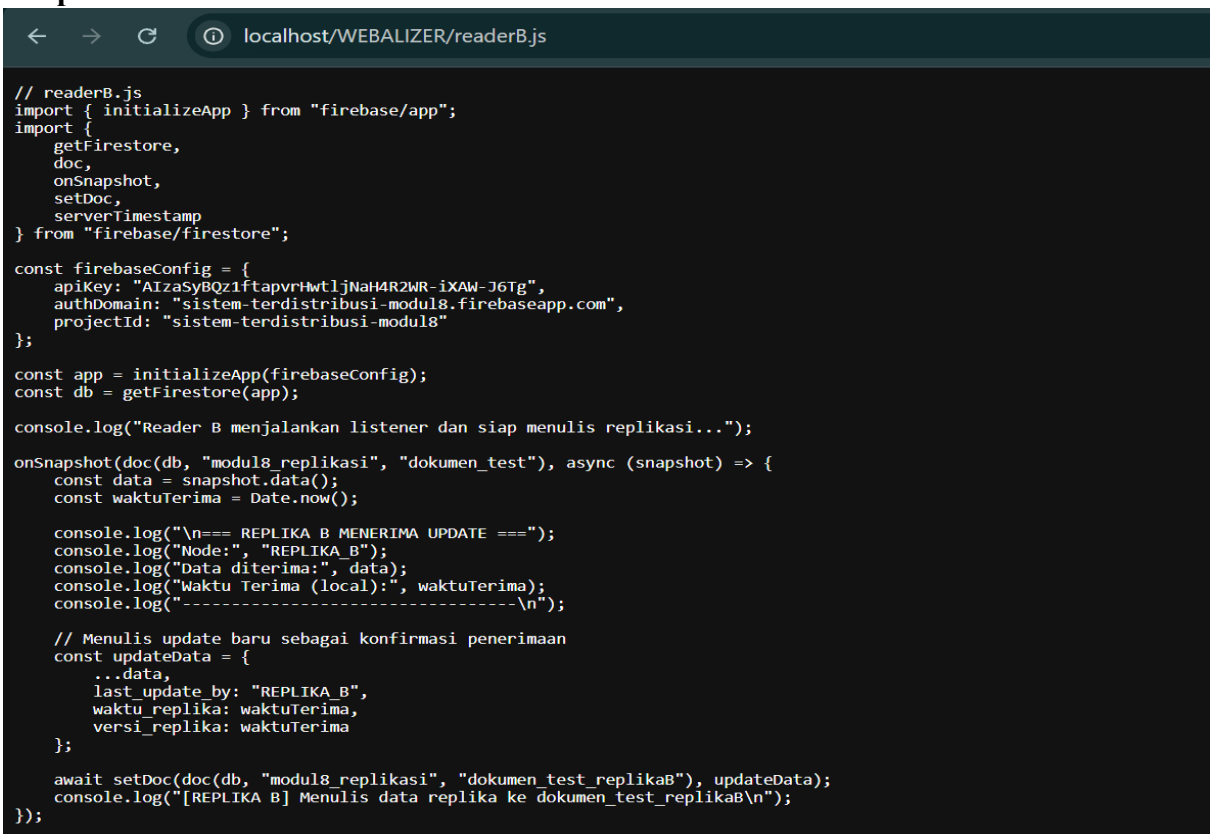
Skrip readerA.js ini berfungsi sebagai node replika yang memantau dokumen dokumen_test di Firestore secara real-time; ketika ada update dari writer, listener onSnapshot() menerima data tersebut, menampilkan informasi update beserta waktu penerimaan, lalu membuat salinan baru dengan menambahkan metadata

replika (last_update_by, waktu_replika, versi_replika) dan menulisnya ke dokumen terpisah dokumen_test_replikaA, sehingga replika ini tidak hanya membaca tapi juga menyimpan data replika di Firestore.

4. **File 3: readerB.js (Node Pembaca B / Replica B)** Node replika kedua yang menerima update yang sama.



Output



Skrip readerB.js berfungsi sebagai node replika yang memantau dokumen dokumen_test di Firestore secara real-time; ketika ada update dari writer, listener onSnapshot() menerima data, menampilkan informasi update beserta waktu penerimaan, lalu membuat salinan baru dengan metadata replika (last_update_by, waktu_replika, versi_replika) dan menulisnya ke dokumen terpisah dokumen_test_replikaB, sehingga replika B tidak hanya membaca tetapi juga menyimpan data replika di Firestore sebagai bukti penerimaan update.

TUGAS 2

Membuat Containerized Application menggunakan Docker dan Podman

BAB I — PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi perangkat lunak menuntut aplikasi dapat dijalankan secara konsisten pada berbagai lingkungan sistem operasi. Salah satu solusi yang digunakan adalah containerization, yaitu teknologi yang memungkinkan aplikasi berjalan dalam lingkungan terisolasi yang disebut container.

Container membantu developer menghindari masalah perbedaan konfigurasi sistem, dependensi, dan lingkungan runtime. Teknologi container yang populer adalah Docker dan Podman. Docker dikembangkan oleh Docker Inc., sedangkan Podman dikembangkan oleh Red Hat.

Pada praktikum ini dilakukan pembuatan containerized application sederhana menggunakan Docker dan Podman.

1.2 Rumusan Masalah

1. Apa itu containerized application?
2. Bagaimana membuat container menggunakan Docker?
3. Bagaimana menjalankan aplikasi menggunakan container?

1.3 Tujuan Praktikum

1. Memahami konsep containerization.
2. Mempelajari penggunaan Docker dan Podman.
3. Membuat aplikasi sederhana berbasis container.
4. Menjalankan aplikasi melalui container.

BAB II — LANDASAN TEORI

2.1 Containerization

Containerization adalah teknologi virtualisasi ringan yang memungkinkan aplikasi berjalan dalam lingkungan terisolasi bersama dependensinya tanpa memerlukan virtual machine penuh.

Keuntungan container:

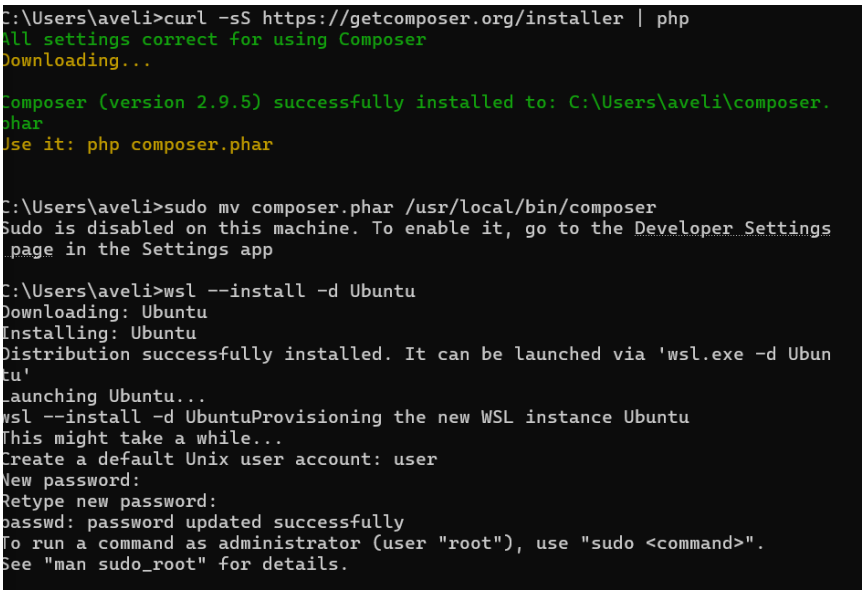
- Portabilitas tinggi

- Konsistensi lingkungan
- Deployment cepat
- Penggunaan resource lebih efisien

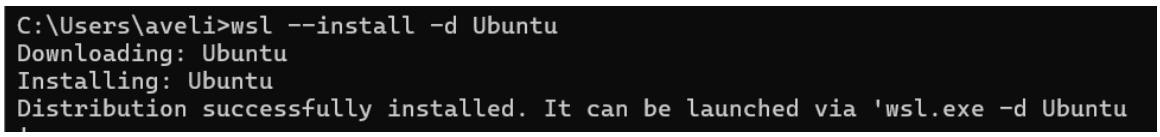
2.2 Docker
Docker adalah platform open-source untuk membuat, menjalankan, dan mengelola container aplikasi. Docker menggunakan Dockerfile untuk mendefinisikan konfigurasi aplikasi.

2.3 Podman
Podman adalah container engine alternatif Docker yang tidak menggunakan daemon dan mendukung rootless container. Podman memiliki command yang hampir sama dengan Docker.
- PHP

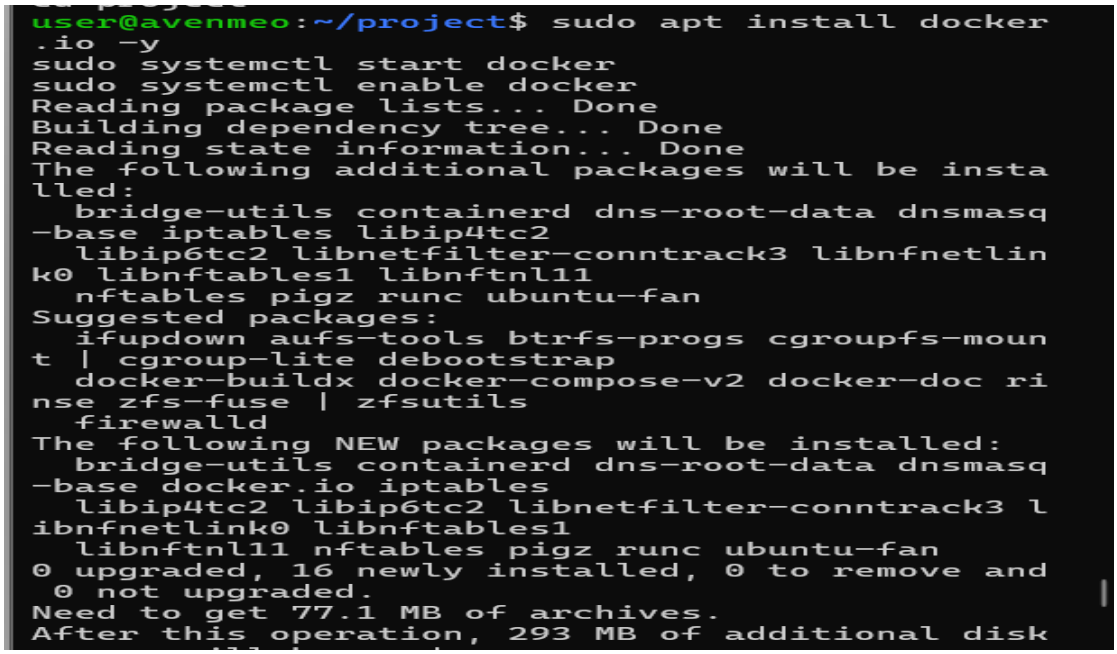
BAB III — LANGKAH PRAKTIKUM



3.1 Persiapan Lingkungan
- Sistem operasi: Windows dengan WSL Ubuntu

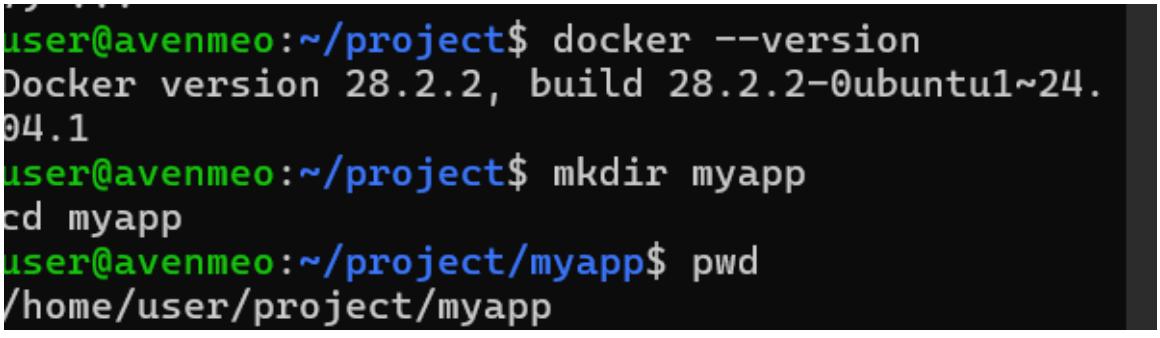


- Docker

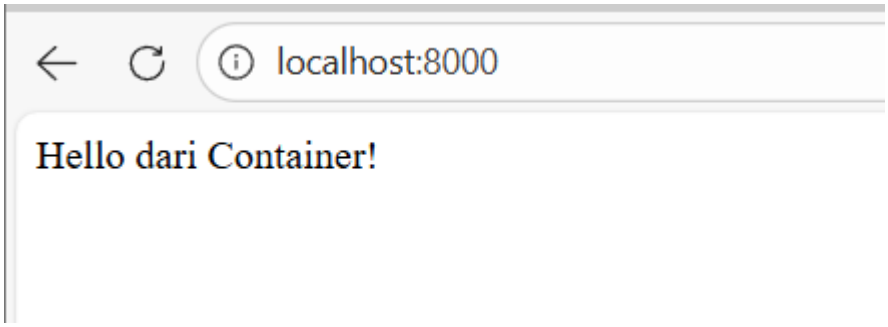


- Podman

3.2 Membuat Aplikasi Sederhana
Membuat folder project:
mkdir myapp
cd myapp



Membuat file index.php:
<?php
echo "Hello dari Container!";



3.3 Membuat Dockerfile
FROM php:8.3-cli

```
Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM php:8.3-cli
8.3-cli: Pulling from library/php
0c8d55a45c0d: Pulling fs layer
```

WORKDIR /app
COPY . .
CMD ["php", "-S", "0.0.0.0:8000"]

```
Step 2/4 : WORKDIR /app
 ---> Running in e59d1641397d
 ---> Removed intermediate container e59d1641397
d
 ---> 99b0ff68af5a
Step 3/4 : COPY . .
 ---> 58250830cd5f
Step 4/4 : CMD ["php", "-S", "0.0.0.0:8000"]
 ---> Running in 323d2139c6bf
 ---> Removed intermediate container 323d2139c6b
f
```

3.4 Build Image Docker
docker build -t myphpapp .

```
user@avenmeo:~/project/myapp$ docker build -t myphpapp .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

time="2026-02-15T01:40:05+07:00" level=error msg="Can't add file /home/user/project/myapp
/Dockerfile to tar: io: read/write on closed pipe"
time="2026-02-15T01:40:05+07:00" level=error msg="Can't close tar writer: io: read/write
on closed pipe"
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/
```

3.5 Menjalankan Container Docker
docker run -p 8000:8000 myphpapp

```
user@avenmeo:~/project/myapp$ sudo docker run -p 8000:8000 myphpapp
[Sat Feb 14 17:58:49 2026] PHP 8.3.30 Development Server (http://0.0.0.0:8000)
 started
[Sat Feb 14 17:59:11 2026] 172.17.0.1:43674 Accepted
[Sat Feb 14 17:59:11 2026] 172.17.0.1:43674 [200]: GET /
[Sat Feb 14 17:59:11 2026] 172.17.0.1:43674 Closing
[Sat Feb 14 17:59:11 2026] 172.17.0.1:43676 Accepted
[Sat Feb 14 17:59:12 2026] 172.17.0.1:43676 [404]: GET /favicon.ico - No such
file or directory
[Sat Feb 14 17:59:12 2026] 172.17.0.1:43676 Closing
```

Aplikasi dapat diakses melalui browser:
http://localhost:8000

```
[Sat Feb 14 17:58:49 2026] PHP 8.3.30 Development Server (http://0.0.0.0:8000)
```

3.6 Menjalankan dengan Podman
podman build -t myphpapp .

```
user@avenmeo:~/project/myapp$ docker build -t myphpapp .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/
```

podman run -p 8000:8000 myphpapp

```
user@avenmeo:~/project/myapp$ podman run -p 8000:8000 myphpapp
WARN[0001] "/" is not a shared mount, this could cause issues or missing mount
s with rootless containers
Error: short-name "myphpapp" did not resolve to an alias and no unqualified-se
arch registries are defined in "/etc/containers/registries.conf"
user@avenmeo:~/project/myapp$
```

BAB V — KESIMPULAN
1. Containerization memungkinkan aplikasi berjalan dalam lingkungan terisolasi.
2. Docker dan Podman dapat digunakan untuk membuat dan menjalankan container.
3. Aplikasi sederhana berhasil dijalankan menggunakan container.
4. Container memberikan kemudahan dalam deployment dan portabilitas aplikasi.