1. Jelaskan teorema CAP dan BASE dan keterkaitan keduanya. Jelaskan menggunakan contoh yang pernah anda gunakan:
   - CAP (Consistence — Availability — Partition tolerance)

   a. Consistency (C): Semua node melihat data yang sama pada waktu yang sama — operasi baca setelah penulisan mengembalikan nilai terbaru.
   b. Availability (A): Setiap permintaan non-failing ke node menghasilkan respon (bukan error), walaupun mungkin bukan nilai paling baru.
   c. Partition tolerance (P): Sistem tetap beroperasi walau ada pemisahan (network partition) antara subset node.

   Teorema: Untuk sistem terdistribusi yang mengalami partisi jaringan (P), Anda hanya bisa memilih antara konsistensi atau ketersediaan jadi praktis saat P terjadi, harus memilih C atau A (bukan keduanya). Jika tidak ada partisi, Anda bisa mendapatkan C dan A, tetapi partisi jaringan selalu mungkin; karenanya pilihan itu nyata.

   Contoh nyata (yang sering digunakan):

   a. Sistem Bank (transfer antar rekening): memilih C over A. Jika ada pemisahan jaringan, sistem bisa menolak transaksi (unavailable) demi menjaga konsistensi (mis. mencegah double-spend).
   b. Sosial Media (feed posting, likes): sering memilih A over C. Saat ada partisi, sistem masih menerima posting/like dan mereplikasi lazimnya (eventual consistency).Pengguna masih bisa berinteraksi walaupun data mungkin sedikit tidak sinkron sesaat.

   - BASE (Basically Available, Soft state, Eventual consistency

   - BASE adalah filosofi desain untuk sistem yang memilih Availability (A) dan Partition tolerance (P) — yaitu tidak menuntut konsistensi kuat pada setiap saat.
   - Basically Available: sistem menjamin ketersediaan minimal (respon terhadap request)
   - Soft state: keadaan internal sistem bisa berubah tanpa input baru (mis. due to replikasi yang sedang berlangsung).
   - Eventual consistency: jika tidak ada mutasi lebih lanjut, semua replica akan konvergen menjadi keadaan yang sama pada akhirnya.

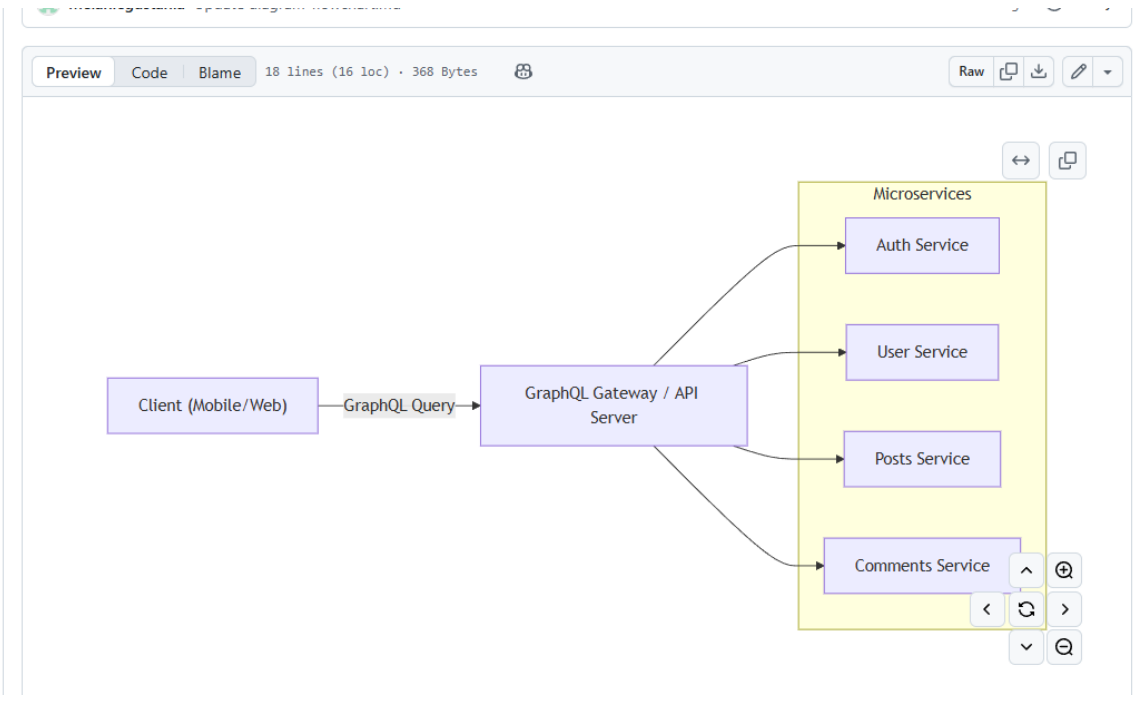
   Kaitan CAP ↔ BASE:

   a. BASE adalah praktik/pendekatan untuk sistem yang memilih A dalam trade-off CAP (terutama ketika P ada). Jadi:
   b. CAP menyatakan trade-off teoritis (C/A/P).
   c. BASE menjelaskan bagaimana merancang sistem yang memprioritaskan Availability dan masih masuk akal: menerima sementara inkonsistensi, lalu mengandalkan replikasi / resolusi konflik untuk mencapai consistency akhirnya (eventual).

2. Jelaskan keterkaitan antara GraphQL dengan komunikasi antar proses pada sistem terdistribusi. Buat diagramnya.
   - GraphQL adalah query language untuk API yang memberi klien kemampuan meminta persis data yang dibutuhkan.
   - Pada lapisan arsitektur, GraphQL sering dipasang sebagai gateway / API layer di atas banyak layanan mikro (microservices).
   - Dalam sistem terdistribusi, GraphQL melakukan orchestration: menerima satu query dari klien -> mem-breakdown menjadi banyak panggilan antar-proses (HTTP/GRPC/DB calls) ke microservices -> mengagregasi hasil -> mengembalikan response tunggal kepada klien.
   - Jadi GraphQL bukan protokol IPC sendiri, tetapi memfasilitasi komunikasi antar-proses (IPC) melalui:
     a. Resolver yang memanggil service backend (via HTTP, gRPC, message bus, dsb).
     b. Batching dan DataLoader untuk mengurangi round-trip dan mencegah N+1 problem (efisiensi IPC).
     c. Schema stitching / federation untuk menggabungkan banyak layanan menjadi satu graph.

     Digramnya:

     https://github.com/Melaniegustania/UTS_245410081_Melanie-Gustania-Nur-Hapsari/tree/main



3. Dengan menggunakan Docker / Docker Compose, buatlah streaming replication di PostgreSQL yang bisa menjelaskan sinkronisasi.
   - Membuat file
   - Kode program

docker-compose - Notepad

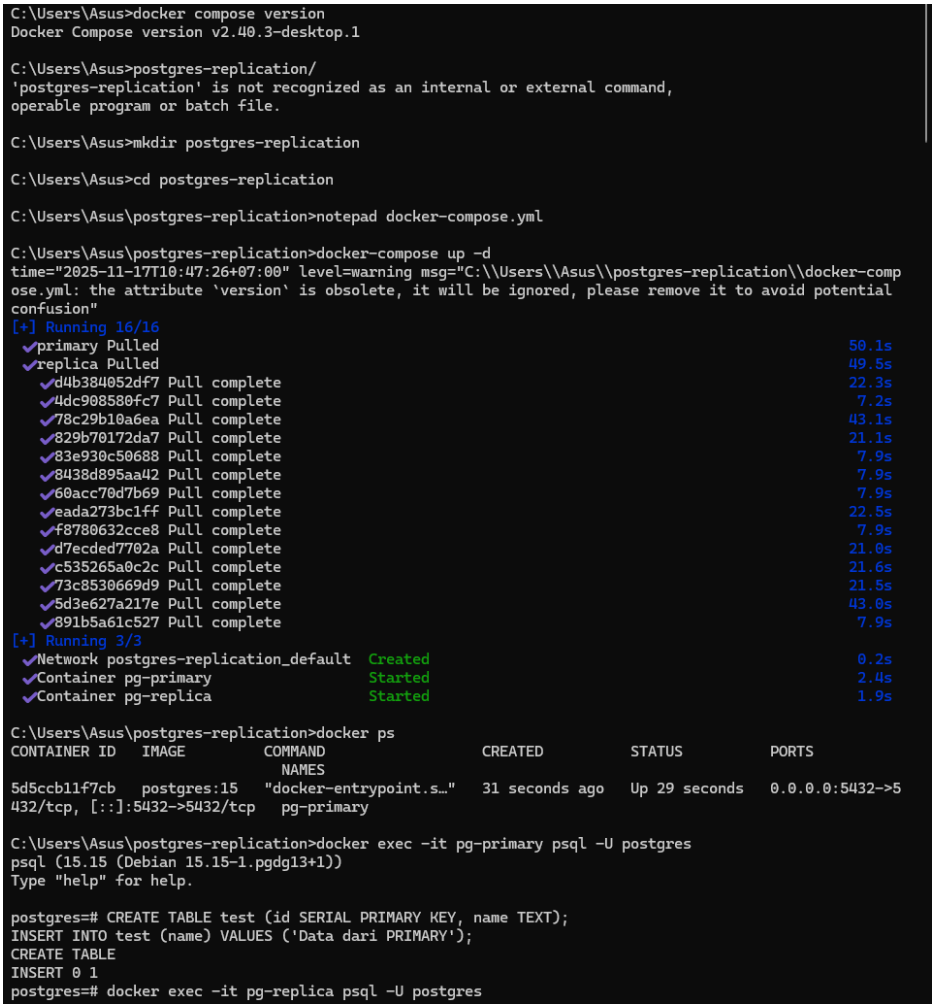
File Edit Format View Help

```
version: "3.9"

services:
  primary:
    image: postgres:15
    container_name: pg-primary
    environment:
      POSTGRES_PASSWORD: postgres
    ports:
      - "5433:5432"
    volumes:
      - ./primary:/var/lib/postgresql/data
    networks:
      - pgnet

  replica:
    image: postgres:15
    container_name: pg-replica
    environment:
      POSTGRES_PASSWORD: postgres
    depends_on:
      - primary
    ports:
      - "5434:5432"
    volumes:
      - ./replica:/var/lib/postgresql/data
    networks:
      - pgnet

networks:
  pgnet:
```

- Membuat postgress-repl dan menjalankan docker-compose

Penjelasannya:

Gambar pada CMD tersebut menunjukkan bahwa kamu sedang membuat folder project bernama *postgres-repl1*, kemudian membuat folder *primary* dan *replica* sebagai tempat penyimpanan data untuk database utama dan database replika. Setelah itu, kamu mencoba membuat file *docker-compose.yml* memakai perintah notepad, tetapi ternyata nama file yang kamu buat salah ketik menjadi **docker-coompose.yml**. Karena kesalahan nama file itu, saat kamu menjalankan perintah docker compose up -d, Docker tidak bisa menemukan file konfigurasi tersebut dan menampilkan pesan error bahwa file compose tidak ditemukan. Jadi, gambar CMD tersebut sebenarnya memperlihatkan langkah kamu menyiapkan file konfigurasi Docker Compose, tetapi program tidak berjalan karena nama file-nya salah. Setelah file dinamai dengan benar menjadi **docker-compose.yml**, barulah Docker bisa menjalankan replikasi PostgreSQL seperti yang seharusnya.



Penjelasan:

proses replikasi PostgreSQL dengan Docker Compose akhirnya berhasil dijalankan. Kamu memulai dengan membuat folder project bernama *postgres-replication*, lalu membuat file konfigurasi **docker-compose.yml** menggunakan Notepad. Setelah file tersebut selesai dibuat dengan benar, kamu menjalankan perintah docker-compose up -d, dan Docker mulai menarik (pull) semua image yang dibutuhkan untuk membuat dua container: **pg-primary** sebagai database utama dan **pg-replica** sebagai database replica. Ketika proses selesai, kedua container berhasil dijalankan, ditandai dengan status *Started*. Setelah itu, kamu masuk ke container **pg-primary** menggunakan perintah docker exec dan membuat tabel serta menyisipkan data baru. Langkah berikutnya, kamu masuk ke container **pg-replica** untuk mengecek apakah data yang dibuat di primary otomatis tersalin ke replika. Semua proses ini menunjukkan bahwa sinkronisasi dan streaming replication PostgreSQL berjalan dengan baik melalui Docker Compose.

- Membuat table pada compose dockernya

```
postgres=# CREATE TABLE test (id SERIAL PRIMARY KEY, name TEXT);
INSERT INTO test (name) VALUES ('Data dari PRIMARY');
CREATE TABLE
INSERT 0 1
postgres=# docker exec -it pg-replica psql -U postgres
postgres-# SELECT * FROM test;
```

```
C:\Users\Asus\postgres-replication>docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS                     PORTS
                                NAMES
0cff7b51414b   postgres:15   "docker-entrypoint.s…"   5 minutes ago   Exited (1) 5 minutes ago
                                pg-replica
5d5ccb11f7cb   postgres:15   "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes               0.0.0
.0:5432->5432/tcp, [::]:5432->5432/tcp   pg-primary
```

```
C:\Users\Asus\postgres-replication>docker compose up -d
time="2025-11-17T10:58:36+07:00" level=warning msg="C:\\Users\\Asus\\postgres-replication\\docker-comp
ose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
confusion"
[+] Running 2/2
 ✔Container pg-primary  Running                                                                  0.0s
 ✔Container pg-replica  Started                                                                  1.0s

C:\Users\Asus\postgres-replication>docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS
                                NAMES
5d5ccb11f7cb   postgres:15   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:5432->5
432/tcp, [::]:5432->5432/tcp   pg-primary

C:\Users\Asus\postgres-replication>docker exec -it pg-primary psql -U postgres
psql (15.15 (Debian 15.15-1.pgdg13+1))
Type "help" for help.

postgres=# CREATE TABLE mahasiswa (
    id serial PRIMARY KEY,
    nama text
);
CREATE TABLE
postgres=# INSERT INTO mahasiswa (nama) VALUES ('Budi'), ('Siti');
INSERT 0 2
```

```
);
CREATE TABLE
postgres=# INSERT INTO mahasiswa (nama) VALUES ('Budi'), ('Siti');
INSERT 0 2
postgres=# SELECT * FROM mahasiswa;
 id | nama
----+------
  1 | Budi
  2 | Siti
(2 rows)
```

Penjelasan:

replikasi PostgreSQL bekerja dengan baik. Kamu masuk ke container **pg-primary**, lalu membuat tabel bernama *test* dan memasukkan satu baris data berisi teks “Data dari PRIMARY”. Perintah tersebut berhasil dijalankan, ditunjukkan oleh pesan *INSERT 0 1*. Setelah itu, kamu masuk ke container **pg-replica** menggunakan perintah docker exec dan menjalankan query SELECT * FROM test; untuk mengecek apakah data yang dibuat di primary sudah tersalin ke replika. Proses ini merupakan tahap verifikasi bahwa mekanisme streaming replication berjalan dengan benar. Jika data muncul di replica tanpa kamu memasukkan apa pun di sana, berarti sinkronisasi antar database sudah berhasil.

- Tampilan pada docker nya