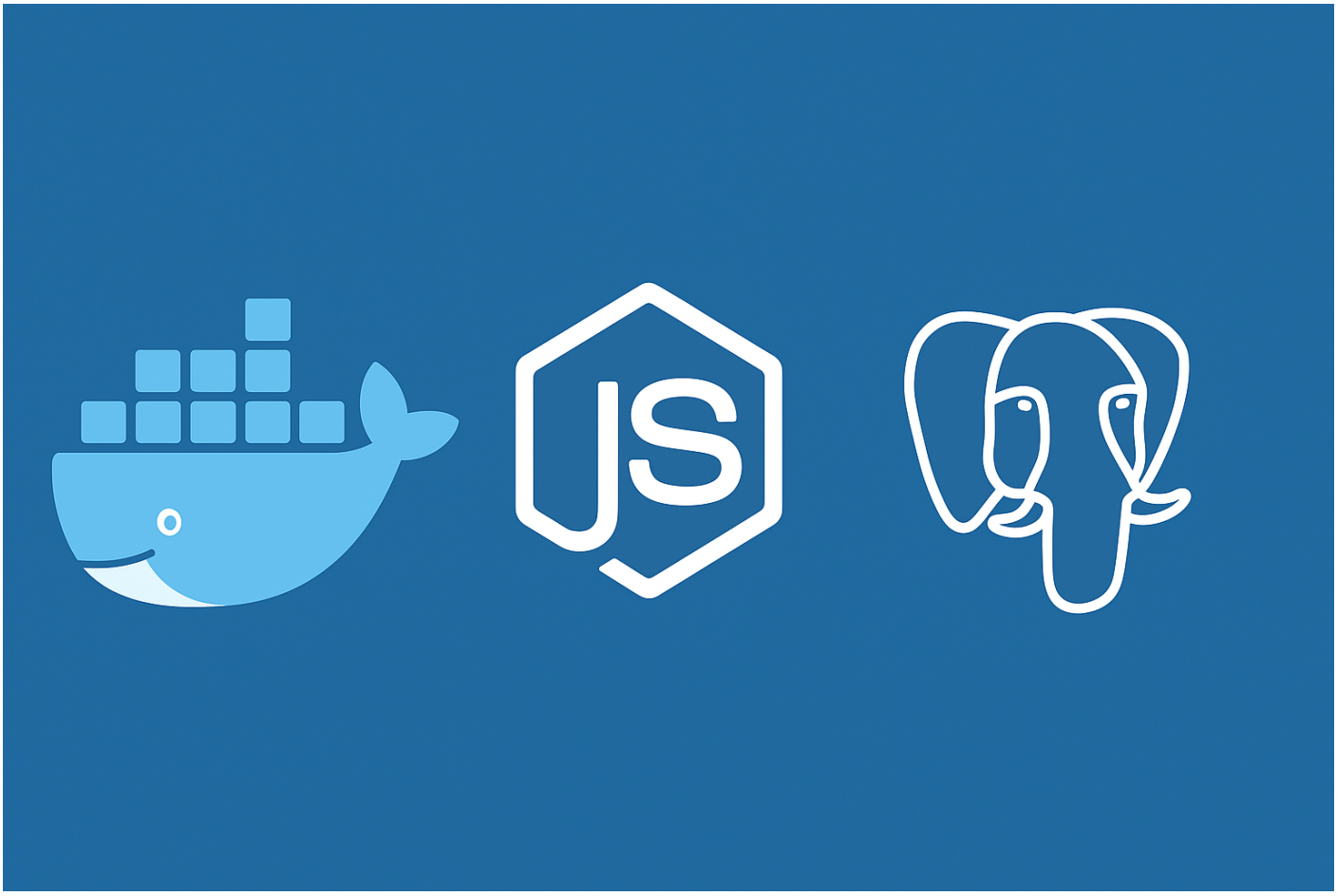
4. Membuat Aplikasi CRUD Sederhana dengan Docker

CRUD Web + Database Full Docker



Docker adalah platform containerization yang memungkinkan kamu membungkus aplikasi beserta dependensinya dalam satu wadah (container). Hal ini mempermudah proses deployment, testing, dan distribusi aplikasi tanpa khawatir perbedaan sistem operasi, package, atau versi runtime.

Tutorial ini cocok untuk:

- Pemula yang ingin belajar perintah dasar Docker
- DevOps / backend developer yang ingin praktik langsung

- Siapa pun yang mau membuat **aplikasi CRUD web + database** secara full menggunakan Docker

---

Persiapan Awal

Syarat:

- Sudah terinstall **Docker** dan **Docker Compose**
- OS: Ubuntu 22.04 LTS
- Editor: bebas (vim, vi, nano, Visual Studio Code, notepad)

Untuk memastikan Docker sudah terinstall:

```
docker -v
docker compose version
```

Jika belum, kunjungi: https://www.32inside.com/books/docker/page/instal-docker

Selanjutnya, kita akan membuat aplikasi web CRUD yang terdiri dari:

- Backend API (Node.js + Express)
- Database (PostgreSQL)
- Frontend GUI sederhana (HTML)
- Semua dijalankan melalui Docker Compose
- Data disimpan secara persisten (volume Docker)

Struktur Folder

```
crud-app/
├── backend
│   ├── Dockerfile
│   ├── index.js
│   ├── package.json
│   └── public
│       └── index.html
└── docker-compose.yml
```

1. Buat Struktur Direktori Proyek

```
mkdir -p crud-app/backend/public
cd crud-app
```

2. Buat File `docker-compose.yml`

```
version: "3.9"

services:
  web:
    build: ./backend
    ports:
      - "3000:3000"
    depends_on:
      - db
    volumes:
      - web-data:/app
    restart: unless-stopped

  db:
    image: postgres:15
    environment:
      POSTGRES_DB: cruddb
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    volumes:
      - pgdata:/var/lib/postgresql/data
    restart: unless-stopped

volumes:
  pgdata:
  web-data:
```

3. File `package.json` untuk Backend

```
cd backend
vim package.json
```

```
{
  "name": "crud-app",
  "version": "1.0.0",
  "main": "index.js",
  "dependencies": {
    "express": "^4.18.2",
```

```
    "pg": "^8.11.0",
    "body-parser": "^1.20.2"
  }
}
```

4. File `index.js` (Backend Logic)

```
vim index.js
```

```
const express = require('express');
const bodyParser = require('body-parser');
const { Pool } = require('pg');
const path = require('path');

const app = express();
const port = 3000;

const pool = new Pool({
  user: 'postgres',
  host: 'db',
  database: 'cruddb',
  password: 'postgres',
  port: 5432,
});

app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));

pool.query(`
  CREATE TABLE IF NOT EXISTS items (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL
  )
`);

app.get('/items', async (req, res) => {
  const result = await pool.query('SELECT * FROM items');
  res.json(result.rows);
});
```

```
app.post('/items', async (req, res) => {
  const { name } = req.body;
  const result = await pool.query('INSERT INTO items(name) VALUES($1) RETURNING *', [name]);
  res.json(result.rows[0]);
});

app.put('/items/:id', async (req, res) => {
  const { id } = req.params;
  const { name } = req.body;
  await pool.query('UPDATE items SET name = $1 WHERE id = $2', [name, id]);
  res.sendStatus(200);
});

app.delete('/items/:id', async (req, res) => {
  const { id } = req.params;
  await pool.query('DELETE FROM items WHERE id = $1', [id]);
  res.sendStatus(200);
});

app.listen(port, () => {
  console.log(`Server running on http://localhost:${port}`);
});
```

5. File `Dockerfile` (Build Backend)

```
vim Dockerfile
```

```
FROM node:18

WORKDIR /app
COPY package.json .
RUN npm install
COPY . .

EXPOSE 3000
CMD ["node", "index.js"]
```

6. File `index.html` (Frontend GUI)

```
cd public
vim index.html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CRUD Docker App</title>
  <style>
    body { font-family: sans-serif; margin: 2rem; }
    input, button { margin: 0.5rem; padding: 0.5rem; }
    table { border-collapse: collapse; margin-top: 1rem; width: 100%; }
    td, th { border: 1px solid #ccc; padding: 0.5rem; text-align: left; }
    tr:nth-child(even) { background-color: #f9f9f9; }
  </style>
</head>
<body>

<h1>CRUD Docker App (Express + PostgreSQL)</h1>

<input type="text" id="newItemName" placeholder="Item Name">
<button onclick="addItem()">Add Item</button>

<table>
  <thead>
    <tr>
      <th>ID</th><th>Name</th><th>Action</th>
    </tr>
  </thead>
  <tbody id="itemsTable"></tbody>
</table>

<script>
const apiBase = "/items";

async function fetchItems() {
  const res = await fetch(apiBase);
  const items = await res.json();
  const table = document.getElementById("itemsTable");
  table.innerHTML = "";
```

```
  items.forEach(item => {
    const row = document.createElement("tr");
    row.innerHTML = `
      <td>${item.id}</td>
      <td contenteditable onblur="updateItem(${item.id}, this.innerText)">${item.name}</td>
      <td><button onclick="deleteItem(${item.id})">Delete</button></td>
    `;
    table.appendChild(row);
  });
}

async function addItem() {
  const name = document.getElementById("newItemName").value;
  if (!name) return alert("Please enter a name.");
  await fetch(apiBase, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name })
  });
  document.getElementById("newItemName").value = "";
  fetchItems();
}

async function updateItem(id, name) {
  await fetch(`${apiBase}/${id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name })
  });
}

async function deleteItem(id) {
  await fetch(`${apiBase}/${id}`, { method: "DELETE" });
  fetchItems();
}

fetchItems();
</script>

</body>
```

```
</html>
```

7. Menjalankan Aplikasi

Setelah semua file selesai dibuat dan diisi, kembali ke root folder:

```
cd ../../
```

8. Build dan jalankan seluruh stack:

```
docker compose up --build -d
```

Buka di browser:

`http://<IPADDRESS>:3000`

Revision #4
Created 26 July 2025 20:11:24 by Kapak Maut
Updated 3 April 2026 02:53:54 by Kapak Maut