🌐 Docker Networking + Hands-On Example

🔑 Core Concepts

🛠️ Network Type	🌟 What It Is	📝 Typical Use
🏖️ **Bridge** (default)	Docker creates a private network on your host. Containers get internal IPs.	Isolated app containers on one host.
🏠 **Host**	Container shares the host's network stack. No NAT, no isolation.	High-performance networking, apps that must bind directly to host ports (Linux only).
🌉 **Overlay**	A network spanning **multiple Docker hosts** (requires Swarm). Containers on different machines talk like on same LAN.	Multi-host / cluster networking.

Expose vs Publish

- **EXPOSE** in Dockerfile = hint/documentation of internal port.
- **p hostPort:containerPort** at run time = **publishes** the port to the host/outside world.

🏗️ Why Use a Custom Network?

Default `bridge` works, but:

- No automatic container name DNS unless you create your own bridge.
- Using a **user-defined bridge** lets you connect multiple containers (like app+DB) and refer to each by name.

📝 Mini-Project: Express App + MongoDB

We'll build:

- A simple **Express** app that stores a visit count in **MongoDB**.
- Run them in separate containers on a custom network.
- Publish only the Express app to the outside world.

📂 Project Structure

```
docker-network-demo/
├─ app/
│  ├─ package.json
│  ├─ index.js
│  └─ Dockerfile
└─ docker-compose.yml  (optional)
```

1️⃣ Express App

app/package.json

```
{
  "name": "docker-network-demo",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mongodb": "^6.0.0"
  }
}
```

app/index.js

```javascript
const express = require('express');
const { MongoClient } = require('mongodb');
const app = express();
const port = process.env.PORT || 3000;

// Use container name 'mongo-db' as host inside the network
const mongoUrl = 'mongodb://mongo-db:27017/demo';

app.get('/', async (req, res) => {
  try {
    const client = await MongoClient.connect(mongoUrl);
    const db = client.db();
    const count = await db.collection('visits').countDocuments();
    await db.collection('visits').insertOne({ visitedAt: new Date() });
    await client.close();
    res.send(`👋 Hello! You are visitor number ${count + 1}`);
  } catch (err) {
    res.status(500).send('Error connecting to MongoDB: ' + err.message);
  }
});

app.listen(port, () => console.log(`App listening on port ${port}`));
```

2️⃣ Dockerfile for Express App

app/Dockerfile

```dockerfile
FROM node:18-alpine

WORKDIR /app

COPY package.json ./
RUN npm install --production

COPY . .
```

```
ENV NODE_ENV=production
EXPOSE 3000
CMD ["node", "index.js"]
```

Build:

```
cd app
docker build -t my-express-app .
```

3️⃣ Create a Custom Bridge Network

```
docker network create my_net
```

This is a **user-defined bridge** network:

- Containers can resolve each other by name.
- Traffic stays isolated from the host and other networks.

4️⃣ Run MongoDB on That Network

```
docker run -d \
  --name mongo-db \
  --network my_net \
  mongo:6
```

- MongoDB listens on `27017` internally.
- Not published to the outside.

5️⃣ Run Express App on That Network

```
docker run -d \
  --name express-app \
  --network my_net \
```

```
  -p 3000:3000 \
  my-express-app
```

- Publishes port 3000 on host → container 3000.
- Inside network, app can reach Mongo by `mongo-db:27017`.

6️⃣ Test

Visit http://localhost:3000 and you'll see:

```
👋 Hello! You are visitor number 1
```

Refresh to increment count — shows Express and Mongo are talking.

7️⃣ Same Setup with docker-compose.yml (simpler)

`docker-compose.yml` in project root:

```yaml
version: '3.8'
services:
  mongo-db:
    image: mongo:6
    networks:
      - my_net

  express-app:
    build: ./app
    ports:
      - "3000:3000"
    networks:
      - my_net

networks:
  my_net:
    driver: bridge
```

Run:

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

Compose automatically creates the `my_net` network, starts both containers, and publishes Express's port.

8️⃣ Visual Model 🧠

```
Host machine
  |
  |--- published port 3000 → Express container:3000
  |
Docker network "my_net" (user-defined bridge)
  ├─ express-app container (connects to mongo-db:27017)
  └─ mongo-db container (not exposed to host)
```

Outside world 🌍 → `localhost:3000` → Express → Mongo (internal)

🧠 Recap of Concepts + Example

- 🏖️ **Bridge network** = Docker's private network on one host.
- 🏠 **Host network** = Container uses host's network stack (no mapping needed).
- 🌉 **Overlay network** = Multi-host virtual LAN (Swarm/K8s).
- 🚪 **EXPOSE** = hint only.
- 🔓 **p host:container** = actually publishes the port.
- 🔗 **Custom bridge** = automatic DNS by container name (as shown in our example).

With this single project you've:

- Learned the 3 main network types.
- Seen how **EXPOSE vs -p** differs.

- Connected multiple containers (app + db) using a custom bridge.
- Published only what you need to the outside world.