Frontend running on port : 8080

Backend running on port :4000

Mongodb running on port :27017

Kong used ports : 8000,8001,8002,8443,8444

1. Different programming language for every service

2. Database per service concept to be applied, different database product for every service

3. You need to use: Discovery Service, Circuit Breaker, API Gateway, Monitoring and Logging

4. Synch as well as Asynch communication to be used like REST API, gRPC, Topic or Queues with Message Brokers (Active MQ, Kafka),

5. Services to be managed through Kubernetes Pods/Clusters having multiple containers like docker to host services of the application you implement.

6. Necessary Security provisions should be there like TLS, certificates generation etc

Steps:

1: run the node.js server

docker-compose build

docker-compose up

2. run the react fronted

docker run -p 8080:8080 fronted-react-app

3. run the Api gateway

docker-compose build

docker-compose up

4. run the Prometheus container

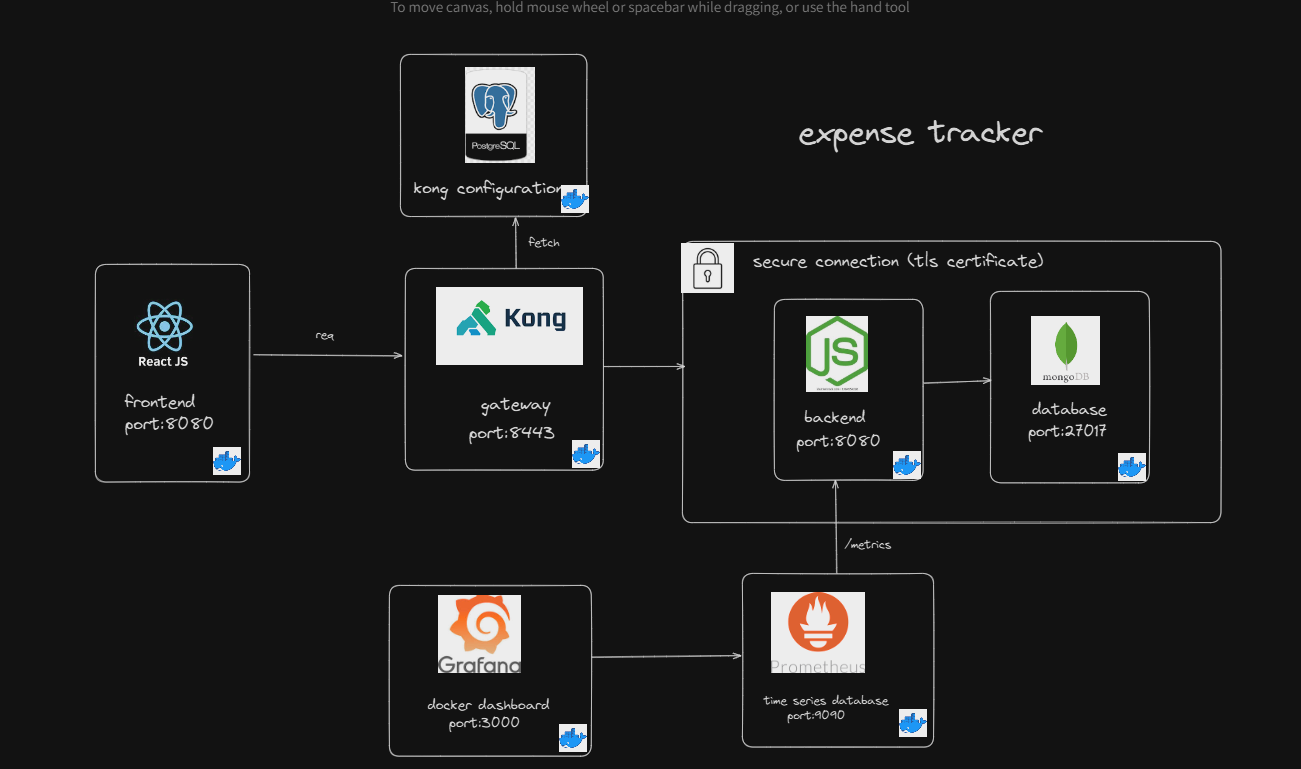
docker-compose build

docker-compose up

5. run the Grafana container

docker-compose build

docker-compose up



**Expense Tracker App - Microservices Architecture Documentation**

This documentation provides an in-depth overview of the Expense Tracker application built with a microservices architecture. The application is designed to manage user expenses efficiently and includes features such as wallet management, transaction logging, monthly summaries, and filtering options. The architecture leverages Kong Gateway for secure API routing, Prometheus and Grafana for monitoring, and MongoDB as the primary data store.

**1. Architecture Overview**

This application is structured as a set of loosely coupled microservices. Each service performs a specific function and can be scaled independently, facilitating a high degree of modularity and maintainability. The services are as follows:

* **Frontend (ReactJS)**: Runs on port 8080 and serves the user interface for managing expenses.
* **Backend (Node.js)**: Runs on port 8080 and provides a REST API for the frontend to interact with the data.
* **Database (MongoDB)**: Stores user data, transactions, and wallet information. It is exposed on port 27017.
* **Kong Gateway**: Serves as a reverse proxy and API gateway, running on port 8443. It routes requests from the frontend to the backend securely.
* **Prometheus**: Monitors the backend, collecting metrics from the /metrics endpoint, and runs on port 9090.
* **Grafana**: Provides a dashboard for visualizing metrics collected by Prometheus, running on port 3000.

**2. Components and Responsibilities**

**Frontend (ReactJS)**

* **Purpose**: Provides the user interface for the expense tracking application.
* **Port**: 8080
* **Main Features**:
  + Wallet management
  + Transaction logging
  + Monthly summary and filters for transactions
* **Docker**: Containerized using a Dockerfile, exposing port 8080.

**Backend (Node.js)**

* **Purpose**: Provides RESTful APIs for handling requests from the frontend.
* **Port**: 8080
* **Endpoints**:
  + /api/transactions: CRUD operations for transaction data
  + /api/wallets: CRUD operations for wallets
  + /metrics: Exposes Prometheus-compatible metrics
* **Docker**: Containerized using a Dockerfile, exposing port 8080.
* **Security**: Communication with the Kong Gateway is secured via TLS.

**Database (MongoDB)**

* **Purpose**: Stores application data, including user information, transactions, and wallet details.
* **Port**: 27017
* **Configuration**:
  + Hosted in a Docker container, exposing port 27017 for secure access from the backend.
  + Stores JSON documents in a flexible schema.

**Kong Gateway**

* **Purpose**: Acts as an API gateway, securing and routing requests to the backend.
* **Port**: 8443
* **Configuration**:
  + Set up with PostgreSQL for configuration storage.
  + Routes requests to the backend based on defined API paths.
  + Adds TLS for secure communications.
  + Load-balances requests as necessary.
* **Docker**: Configured and managed via Docker for ease of deployment.

**Prometheus**

* **Purpose**: Collects metrics from the backend, particularly from the /metrics endpoint.
* **Port**: 9090
* **Configuration**:
  + Set up to scrape the backend metrics endpoint at regular intervals.
  + Stores time-series data for monitoring purposes.
* **Docker**: Configured in Docker to simplify integration with the rest of the architecture.

**Grafana**

* **Purpose**: Visualizes data collected by Prometheus.
* **Port**: 3000
* **Configuration**:
  + Dashboards are configured to display backend performance metrics.
  + Allows for customized alerts and data visualizations.
* **Docker**: Configured in Docker with predefined dashboards for expense tracking metrics.

**3. Service Descriptions**

**Frontend Service (React)**

The frontend provides an interactive user experience. Users can add wallets, record transactions, and view or filter past transactions. It communicates with the backend through the Kong Gateway, which secures and routes API requests.

**Backend Service (Node.js)**

The backend is the central hub of the application. It provides RESTful API endpoints to handle CRUD operations on expenses, wallets, and other core features. It also exposes a /metrics endpoint that Prometheus scrapes to gather performance data.

**Kong Gateway**

Kong is configured as the API gateway for the application. It routes incoming requests from the frontend to the backend and provides security by requiring TLS. PostgreSQL is used to store Kong's configuration, allowing it to handle complex routing and security policies.

**Prometheus**

Prometheus is used for monitoring and collecting metrics from the backend. It scrapes data from the backend’s /metrics endpoint. This data is then visualized in Grafana.

**Grafana**

Grafana provides a dashboard to visualize and monitor the performance of the backend service. It pulls data from Prometheus and allows for the creation of custom dashboards and alerts based on collected metrics.

**4. Inter-Service Communication**

* **Frontend to Backend**: Requests from the frontend go through Kong Gateway on port 8443, which routes them to the backend.
* **Backend to MongoDB**: The backend connects directly to MongoDB on port 27017 to perform database operations.
* **Prometheus to Backend**: Prometheus scrapes the backend’s /metrics endpoint periodically to collect data.
* **Grafana to Prometheus**: Grafana pulls metrics data from Prometheus to generate dashboards.

**5. Docker Setup**

Each service is containerized. Below are sample Dockerfiles and docker-compose configuration.

**Dockerfile for React Frontend**

dockerfile

Copy code

# Dockerfile for React

FROM node:14-alpine

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD ["npm", "start"]

**Dockerfile for Node.js Backend**

dockerfile

Copy code

# Dockerfile for Node.js backend

FROM node:14-alpine

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD ["node", "server.js"]

**docker-compose.yml**

yaml

Copy code

version: '3.8'

services:

frontend:

build:

context: ./frontend

ports:

- "8080:8080"

backend:

build:

context: ./backend

ports:

- "8080:8080"

environment:

MONGODB\_URI: "mongodb://database:27017/expenses"

database:

image: mongo

ports:

- "27017:27017"

kong:

image: kong

ports:

- "8443:8443"

prometheus:

image: prom/prometheus

ports:

- "9090:9090"

grafana:

image: grafana/grafana

ports:

- "3000:3000"

**6. Kong Gateway Configuration**

To configure Kong Gateway, you’ll need to set up routes, services, and consumers. The following commands illustrate the basic setup.

bash

Copy code

# Configure Kong service

curl -i -X POST http://localhost:8001/services/ \

--data "name=backend" \

--data "url=http://backend:8080"

# Configure route for service

curl -i -X POST http://localhost:8001/services/backend/routes \

--data "paths[]=/api" \

--data "strip\_path=true"

**7. Prometheus and Grafana Setup**

* **Prometheus Configuration**: Update the Prometheus config file to scrape metrics from the backend.
* **Grafana Setup**: Add Prometheus as a data source and import a dashboard for visualizing backend metrics.

**8. Security Configuration**

Enable TLS in Kong Gateway to secure communication between frontend and backend.

**9. Environment Variables and Configuration**

Include environment variables in a .env file for ease of configuration.