MicroServices

let’s break a monolithic [Node.js](https://Node.js/) app into microservices and make it container- or cloud-friendly:

### 🧱 **Starting Point – Monolith**

You have one big project directory like this:

/rentify-app

├── app.js

├── routes/

├── models/

├── services/

├── utils/

└── database.js

Everything — authentication, payments, listings, reviews — lives inside this single [Node.js](https://Node.js/) process.

### 🛠️ **Break It Down – Microservice-Friendly Structure**

Split core features into independent services, each with its own API, logic, and database access:

/rentify-services

├── users-service → handles signups, login

├── property-service → manages listings, photos

├── booking-service → handles leases, payments

├── review-service → user ratings and reviews

├── gateway-api → routes external traffic to services

Each of these becomes its own [**Node.js**](https://Node.js/) app, with its own [package.json](https://package.json/), Dockerfile, and optionally, its own database or schema.

### 🐳 **Containerize with Docker**

Each service can now be built and deployed in a container:

# Example: users-service Dockerfile

FROM node:18

WORKDIR /app

COPY . .

RUN npm install

CMD ["npm", "start"]

You can orchestrate them locally with docker-compose or in the cloud with ECS, Kubernetes, or App Runner.

### 🌐 **API Gateway or Reverse Proxy**

Use Amazon API Gateway or Nginx to expose and route traffic:

* /users/\* → users-service
* /properties/\* → property-service
* /bookings/\* → booking-service

This cleanly separates internal logic while keeping one external-facing endpoint.

### 💬 **Service-to-Service Communication**

* Use REST (HTTP) or messaging systems like SQS, SNS, or EventBridge
* Keep communication asynchronous when possible for better performance

### 🧠 **Benefits You Unlock**

* Independent scaling (e.g., bookings spike, only that service scales)
* Easier updates & deployments
* Better fault isolation
* Cloud native: ready for Lambda, ECS, or EKS

Let me know if you'd like a detailed folder structure for each microservice or a [docker-compose.yml](https://docker-compose.yml/) to wire them together!