IaaS Setup: Dockerized Front-End with Standard Back-End, AI Middleware, and Database

Table of Contents

[Introduction 2](#_Toc177638010)

[1. Front-End (React.js): 3](#_Toc177638011)

[2. Back-End (Node.js): 3](#_Toc177638012)

[3. AI/Chatbot Middleware: 4](#_Toc177638013)

[4. Database: 4](#_Toc177638014)

[Networking and Traffic Management: 5](#_Toc177638015)

[Storage for Docker Containers: 5](#_Toc177638016)

[Operating System (OS) for VMs: 6](#_Toc177638017)

[Resource List for IaaS Providers: 6](#_Toc177638018)

[Front-End (React.js) Resources: 6](#_Toc177638019)

[Back-End (Node.js) Resources: 6](#_Toc177638020)

[AI Middleware Resources: 7](#_Toc177638021)

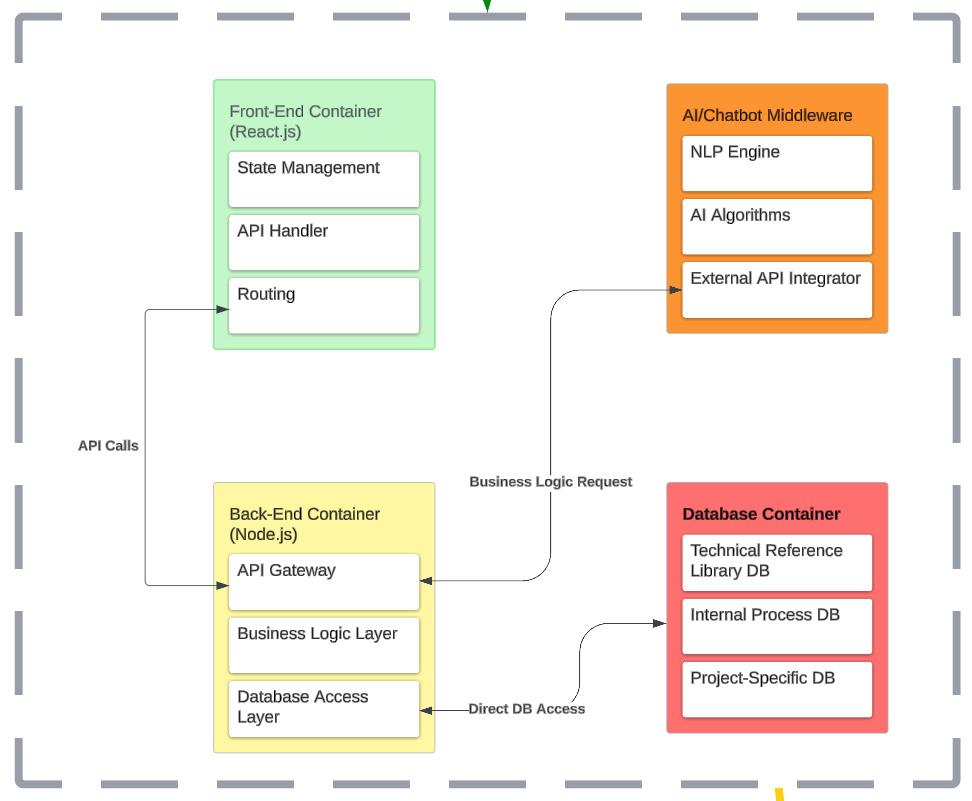
[Database Resources: 7](#_Toc177638022)

[Networking: 7](#_Toc177638023)

[Summary: 7](#_Toc177638024)

# Introduction

This setup leverages **Docker containers** for the **Front-End** and standard OS and resource configurations for the **Back-End**, **AI Middleware**, and Database. The architecture ensures compatibility across major IaaS providers while offering flexibility, scalability, and high availability.



# 1. Front-End (React.js):

* **Containerized Front-End**: The React.js application will run inside a **Docker container**, serving static front-end files (HTML, CSS, JavaScript).
* Service:
  + Use any **Container Orchestration Service** (e.g., Docker Swarm, Kubernetes, or any IaaS-provided container service).
  + Compute:
  + vCPU: 0.25 to 0.5 vCPU
  + Memory: 0.5 to 1 GB RAM
* **Operating System**: Any **Linux-based OS** that supports Docker (e.g., Ubuntu, CentOS, or Debian).
* Storage: Static files will be stored directly inside the Docker container. No external volumes or persistent storage are required unless you have specific requirements for storing larger assets.
* **Auto-Scaling**: Use the container orchestration tool to scale horizontally as traffic increases.

# 2. Back-End (Node.js):

* **Managed Back-End**: The Node.js back-end runs directly on VMs provided by the IaaS provider. It handles business logic, connects with the database, and processes API calls.
* Service:
  + Deployed using **Virtual Machines (VMs)** or **bare metal servers**, depending on your performance needs.
  + Compute:
  + vCPU: 1 to 2 vCPUs
  + Memory: 2 to 4 GB RAM
* **Operating System**: A **Linux-based OS** like Ubuntu, CentOS, or Debian to ensure compatibility with Node.js and related frameworks.
* Storage: Local SSD or HDD storage for logs, session data, or cache, depending on your performance needs. Use high I/O SSDs for fast response times.
* **API Management**:
  + The back-end will manage API traffic between the front-end and AI Middleware.
* **Auto-Scaling**: Use standard auto-scaling mechanisms provided by the IaaS platform (e.g., auto-scaling groups).

# 3. AI/Chatbot Middleware:

* **AI Middleware**: The AI Middleware handles tasks like NLP, proposal scoring, and document generation. It can be deployed on general-purpose VMs or GPU-enabled instances for heavier workloads.
* Service:
  + Deployed on **Virtual Machines (VMs)** or **GPU-enabled instances** if your AI tasks require advanced computations.
  + Compute:
  + vCPU: 2 to 4 vCPUs
  + Memory: 4 to 8 GB RAM
  + Optional: **GPU instances** for more complex AI computations.
* **Operating System**: Ubuntu, Debian, or CentOS for general AI tasks, with **CUDA drivers** if using GPU for machine learning models.
* Storage:
  + SSD-backed local storage for temporary files, model artifacts, and AI processing data.
* **External API Integration**:
  + The AI middleware will connect to the back-end API service to request and process data, returning results to the back-end for final output.
* **Auto-Scaling**: Dynamic scaling based on AI task load.

# 4. Database:

* **External Database Services**: The database will be a standalone service using **Virtual Machines (VMs)** with standard OS.
* Service:
  + Deployed on VMs or **bare metal** servers to manage your database.
  + Database:
  + PostgreSQL for relational databases.
  + NoSQL databases like MongoDB or Cassandra if needed for flexible, high-throughput data.
  + Compute:
  + vCPU: 2 vCPUs
  + Memory: 4 to 8 GB RAM
* **Operating System**:
  + Use **Linux-based OS** like Ubuntu, CentOS, or Debian.
* Storage:
  + SSD-backed storage to ensure high performance and fast I/O for database queries and transactions.
  + **Storage Type**: 100 GB SSD storage for database data and backups.
* Backup: Automated backups via the IaaS provider or using custom backup scripts.

# Networking and Traffic Management:

* **API Gateway**: Use a standard **API Gateway** service for managing API requests between the front-end, back-end, and AI Middleware.
  + **Traffic Handling**: Ensure the API Gateway can manage rate-limiting, throttling, and secure API communication.
* **Load Balancer**:
  + Use a **standard load balancer** to distribute traffic across the VMs for the back-end, ensuring redundancy and failover support.
  + **Auto-Scaling**: Load balancers work in conjunction with auto-scaling groups to dynamically add more VMs based on traffic load.

# Storage for Docker Containers:

* **Container Registry**: Use a **container registry** (such as **Docker Hub**, JFrog, or any IaaS provider’s registry) to store and manage Docker images for the front-end. This will allow easy updates and deployments.
  + Storage: Minimal storage for the front-end container images (estimated at 500 MB to 1 GB per image).

# Operating System (OS) for VMs:

* **Standard OS**: Use any **Linux-based OS** compatible with your application stack:
  + Ubuntu
  + Debian
  + CentOS
* Compatibility: These standard OS options work across all major IaaS providers and offer flexibility, security, and stability for application deployment.

# Resource List for IaaS Providers:

## Front-End (React.js) Resources:

* **1 Docker container** for static React.js assets.
* Compute: 0.25 to 0.5 vCPU, 0.5 to 1 GB RAM.
* **Operating System**: Linux-based (Ubuntu, CentOS, Debian).
* **Auto-Scaling**: Horizontal scaling to multiple containers when traffic increases.

## Back-End (Node.js) Resources:

* **2 VMs** for back-end API, business logic, and database access.
* Compute: 1 to 2 vCPUs, 2 to 4 GB RAM per VM.
* **Operating System**: Linux-based (Ubuntu, CentOS, Debian).
* Storage: Local SSD storage for cache and logs.
* **Auto-Scaling**: Based on traffic and API requests.

## AI Middleware Resources:

* **1-2 VMs or GPU instances** for AI processing tasks.
* Compute: 2 to 4 vCPUs, 4 to 8 GB RAM.
* **Optional GPU**: For machine learning tasks.
* **Operating System**: Linux-based (Ubuntu, CentOS, Debian).
* Storage: SSD-backed storage for AI artifacts and processing.

## Database Resources:

* **1 VM** for the database.
* Compute: 2 vCPUs, 4 to 8 GB RAM.
* **Operating System**: Linux-based (Ubuntu, CentOS, Debian).
* Storage: 100 GB SSD-backed storage for data.

## Networking:

* **API Gateway**: Manages API traffic and ensures secure communication between front-end, back-end, and AI middleware.
* **Load Balancer**: Distributes incoming traffic to the appropriate VMs.

# Summary:

This IaaS setup ensures compatibility with any provider while maintaining high scalability and performance. The architecture uses **standard Linux OS** across all services, ensuring flexibility to run on **any cloud provider**. The use of **Docker containers** is limited to the front-end, while VMs handle the back-end, AI middleware, and database tasks with proper load balancing, API management, and auto-scaling in place.