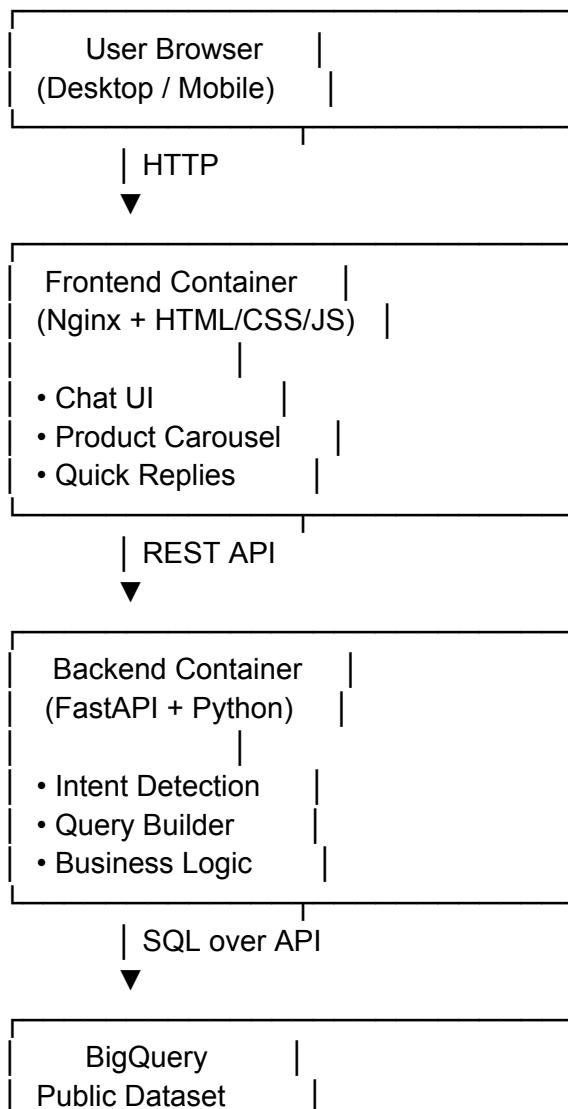


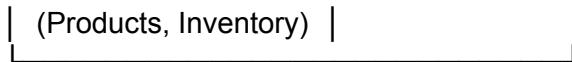
# Deployment Diagram — E-commerce Chatbot

## Overview

This diagram shows how the E-commerce Chatbot is deployed locally and how it would scale in a production environment. The system is containerized using Docker and follows a clean separation of frontend, backend, and data services.

## Local Deployment (Docker Compose)





## Container Responsibilities

### Frontend (chatbot-frontend)

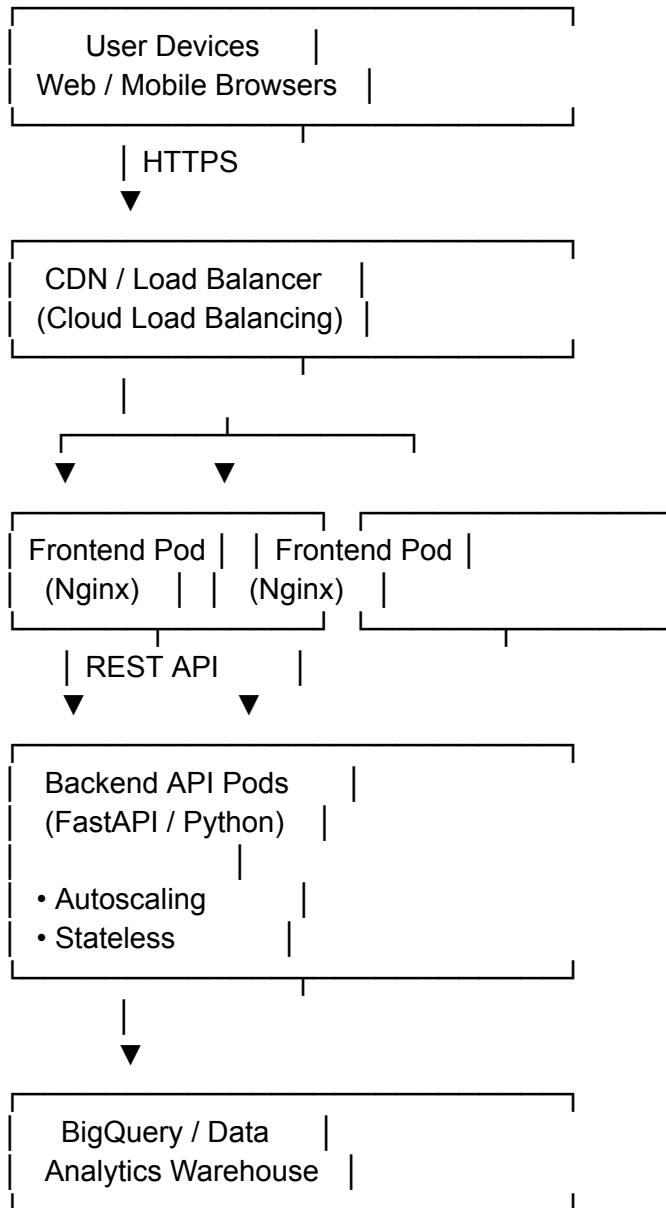
- Runs on **Nginx**
- Serves static assets:
  - `index.html`
  - `app.js`
  - `app.css`
- Handles:
  - Chat rendering
  - User input
  - Product cards
  - Quick replies

### Backend (chatbot-backend)

- Runs **FastAPI** with Uvicorn
- Responsibilities:
  - Natural language parsing
  - Constraint extraction
  - SQL query generation
  - Response formatting

- Stateless design (scales horizontally)

## Production Deployment (Recommended)



## Infrastructure Notes

- Docker Compose for local development

- **Kubernetes (GKE/EKS)** recommended for production
- **Cloud Load Balancer** for traffic routing
- **Service Account** for BigQuery access
- **Secrets Manager** for credentials
- **Horizontal Pod Autoscaler** for backend scaling

## Security & Reliability

- No credentials exposed to frontend
- Backend uses Google Application Default Credentials
- Parameterized SQL queries prevent injection
- Stateless backend enables zero-downtime deploys

## Why This Deployment Design

- Simple local setup
- Production-ready scaling model
- Clear separation of concerns
- Cost-efficient analytics using BigQuery
- Easily extensible to LLM or recommendation services