Fundamentals of ML Applications

Microservices and Containerization

Eloi Pereira, PhD

CE290I - Control and Information Management, UC Berkeley
November, 3 2025

About me

- Head of Data Science / Machine Identity & Intelligence Services at Car IQ Inc.
- Former:
  - Head of Metrology Lab, Portuguese Air Force
  - Professor and Researcher,
    - Portuguese Air Force Academy,
    - ISEC - Coimbra Institute of Engineering,
    - and UC Berkeley
- Education:
  - PhD in Systems Eng., UC Berkeley
  - Masters in Robotics, U Porto
  - Electrical and Computer Eng., Portuguese Air Force / U Lisboa
- Research interests:
  - Mobile Robotics, Drones, Domain-Specific Languages, Models of Computation

Email: eloi@berkeley.edu / eloi@cariqpay.com

Webpage: www.eloipereira.com

LinkedIn: https://www.linkedin.com/in/eloipereira/

Agenda

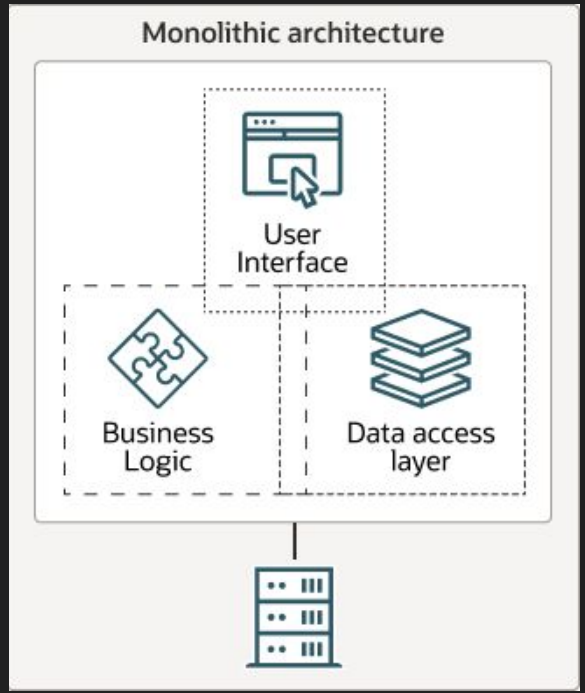
Monolithic architecture vs Microservices architecture

Why Microservices for ML applications

Containers & Microservices

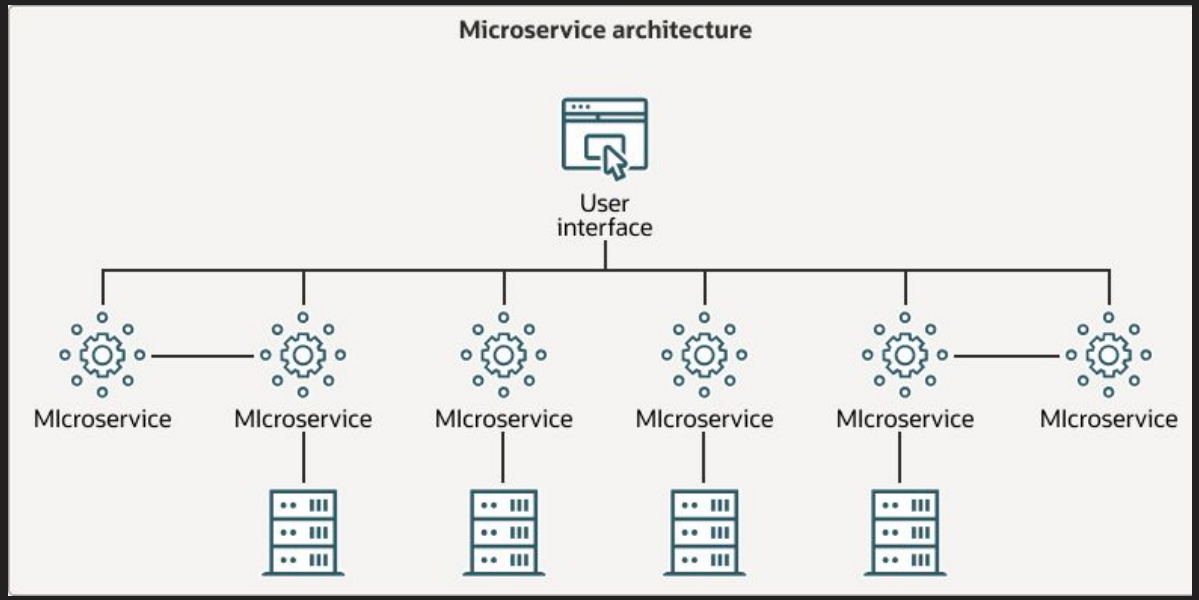
Code example: https://github.com/eloipereira/ce290i-microservices/tree/main

Monolithic architecture



Source: https://docs.oracle.com/en/solutions/learn-architect-microservice

Microservice architecture



Source: https://docs.oracle.com/en/solutions/learn-architect-microservice

Monolithic vs Microservices

Characteristic	Microservices	Monolithic
Design	Loosely coupled, single-purpose services	Single, unified codebase
Reuse	APIs enable easy reuse across apps	Limited reuse outside the app
Communication	Service-to-service via REST/HTTP	Internal function calls
Tech Stack	Each service can use its own tech	Single language and framework
Data	Decentralized, per-service databases	Centralized database

Monolithic vs Microservices

Characteristic	Microservices	Monolithic
Deployment	Independent deployments	Entire app redeployed for any change
Maintenance	Easier to update and evolve	Becomes complex as app grows
Resilience	One service failing doesn’t stop others	One failure can bring down the app
Scalability	Scale services independently	Must scale whole system

Why Microservices for ML applications?

Modern ML systems demand modularity, scalability, and flexibility.

Microservices deliver these by design.

Key Advantages:

- 🧩 **Modular Components** – Separate data ingestion, feature engineering, training, inference, and monitoring into independent services.
- 🚀 **Scalable Workloads** – Scale model inference or data pipelines independently to handle variable demand.
- 🔄 **Continuous Delivery** – Deploy and update models or pipelines without redeploying the entire system.
- 🧠 **Technology Freedom** – Use the best tools for each task (e.g., Python for modeling, Go for APIs).
- 🔍 **Observability & Resilience** – Easier monitoring, fault isolation, and rollback for specific components.
- ☁️ **Cloud-Native Alignment** – Works seamlessly with Kubernetes, CI/CD, and MLOps platforms.

Containers & Microservices

🧱 What Are Containers

Lightweight, portable units that package code, dependencies, and runtime together.

Run consistently across environments — from a laptop to production servers.

Start quickly and use fewer resources than virtual machines.

⚙️ Why Containers Matter for Microservices

Isolation: Each microservice runs in its own container — independent, predictable, and secure.

Portability: “Runs anywhere” — ideal for distributed, multi-environment systems.

Scalability: Easy to replicate containers for load balancing or autoscaling.

Consistency: Eliminates “it works on my machine” issues across teams.

Rapid Deployment: Enables CI/CD pipelines for continuous delivery of microservices.

Integration: Works seamlessly with Kubernetes for orchestration and resilience.

💡 In short: Containers make microservices practical — ensuring every service runs the same way, everywhere, at any scale.

Let us jump into code…

Use case:

Containerize an simple microservices architecture comprised of a data ingestor, a REST API backend, and a Frontend application

Stack:

- Containers framework: Docker
- Web framework: FastAPI
- Frontend: Streamlit