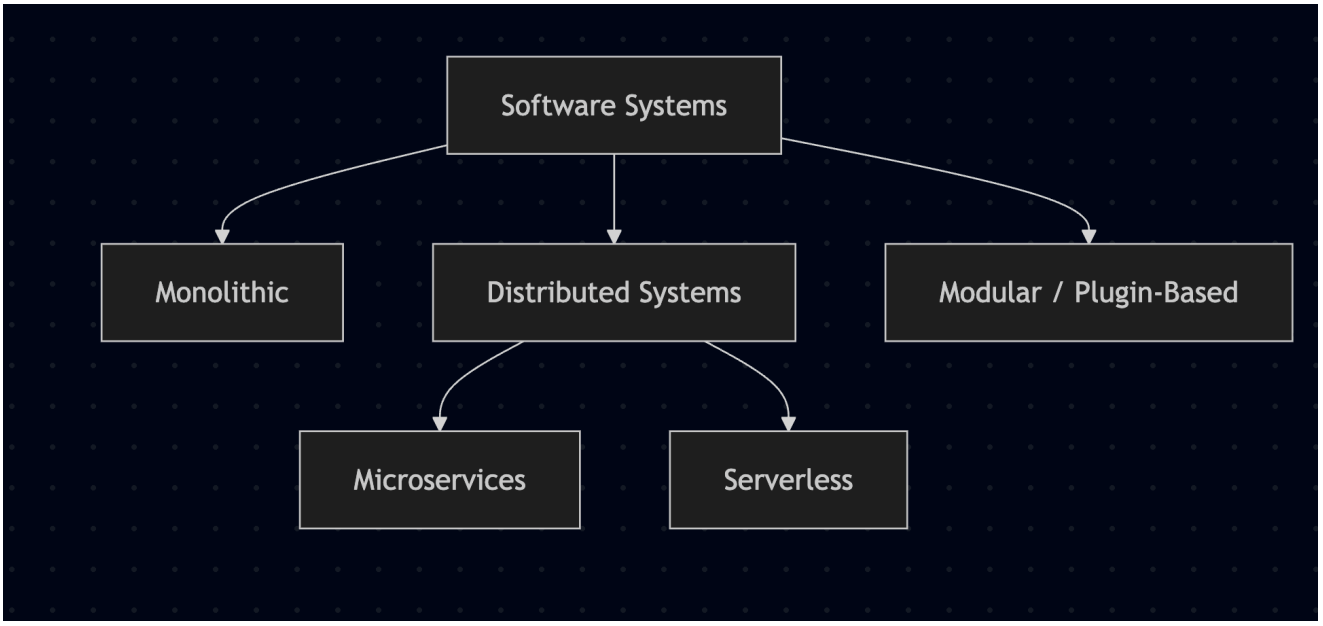
Lesson 1 -> Types of systems

Types of Systems (with ML Context)



1️⃣ Monolithic System

📌 Definition

A **single application** where all features (UI, business logic, data access, ML logic) are bundled and deployed together.

🧱 Characteristics

- One codebase
- One deployment unit
- Shared resources (CPU, memory)
- Tight coupling between components

✅ Pros

- Simple to build and deploy
- Easy debugging
- Suitable for small teams

❌ Cons

- Poor scalability

- One failure impacts entire system
- ML workloads compete with core logic

🌍 Real-world Examples

- Early-stage startups
- Legacy banking systems
- Internal enterprise tools

🤖 ML Mapping

- MVP models
- Batch predictions
- Not suitable for real-time ML at scale

2️⃣ Distributed System

📌 Definition

A system where **multiple machines work together** to appear as a single system.

🧱 Characteristics

- Runs across multiple nodes
- Network communication
- Fault-tolerant by design

✅ Pros

- High scalability
- Fault tolerance
- Parallel processing

❌ Cons

- Complex debugging
- Network failures
- Consistency challenges

🌍 Real-world Examples

- Hadoop
- Spark
- Kafka

- Google Search

🤖 ML Mapping

- Model training
- Feature engineering
- Offline inference

3️⃣ Microservices Architecture

📌 Definition

An application broken into **small, independent services**, each responsible for one business capability.

🧱 Characteristics

- Independent deployment
- Independent scaling
- Service-to-service communication

✅ Pros

- Fault isolation
- Independent scaling
- Faster iteration

❌ Cons

- Operational complexity
- Requires DevOps maturity
- Network overhead

🌍 Real-world Examples

- Netflix
- Amazon
- Uber
- Swiggy

🤖 ML Mapping

- Real-time inference APIs
- Recommendation systems

- Fraud detection services

4️⃣ Serverless Architecture

📌 Definition

Code that runs **on demand** without managing servers. Infrastructure is fully managed by cloud providers.

🧱 Characteristics

- Event-driven
- Auto-scaling
- Pay-per-execution

✅ Pros

- No server management
- Zero idle cost
- Instant scalability

❌ Cons

- Cold start latency
- Execution time limits
- Not ideal for heavy ML inference

🌍 Real-world Examples

- AWS Lambda
- Google Cloud Functions
- Azure Functions

🤖 ML Mapping

- Feature pipelines
- Image processing
- Async ML jobs
- Data validation

5️⃣ Modular / Plugin-Based System

📌 Definition

A core application with **pluggable modules** that extend functionality without changing the core.

🧱 Characteristics

- Extensible design
- Core remains stable
- Plugins can be added/removed

✅ Pros

- High flexibility
- Encourages experimentation
- Easy feature isolation

❌ Cons

- Not ideal for massive scale
- Plugin compatibility issues

🌍 Real-world Examples

- VS Code extensions
- WordPress plugins
- Jupyter extensions
- Obsidian plugins

🤖 ML Mapping

- Model experimentation
- A/B testing frameworks
- Internal ML platforms

🧠 One-Line Summary

- **Monolithic** → Easy to start
- **Distributed** → Scales computation
- **Microservices** → Scales applications & ML inference
- **Serverless** → Event-driven ML workflows
- **Modular** → ML experimentation & extensibility

Distributed System
└── Model Training (Spark, GPUs)

Serverless
└── Feature pipelines & events

Microservices
└── Real-time inference

Modular
└── Model experimentation layer

Comparison Table

Type	Single Deploy	Network Based	Independent Scaling	Best For
Monolithic	✅	❌	❌	Small systems
Distributed	❌	✅	Varies	Large scale
Microservices	❌	✅	✅	Large apps
Serverless	❌	✅	Auto	Event-driven
Modular	✅	❌	❌	Extensible apps