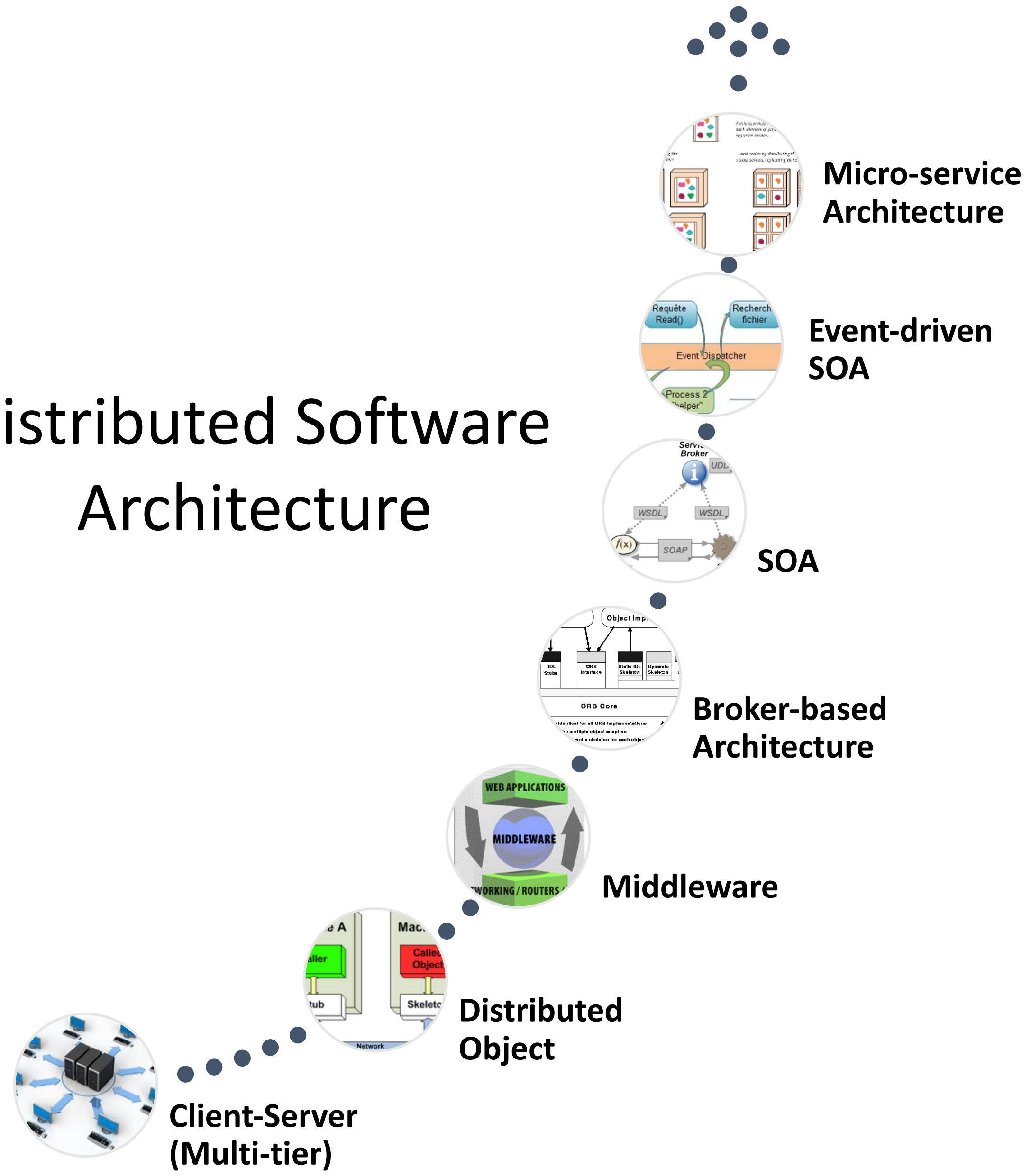
Microservice Tutorial

Slides are adopted from the Internet

Distributed Software Architecture

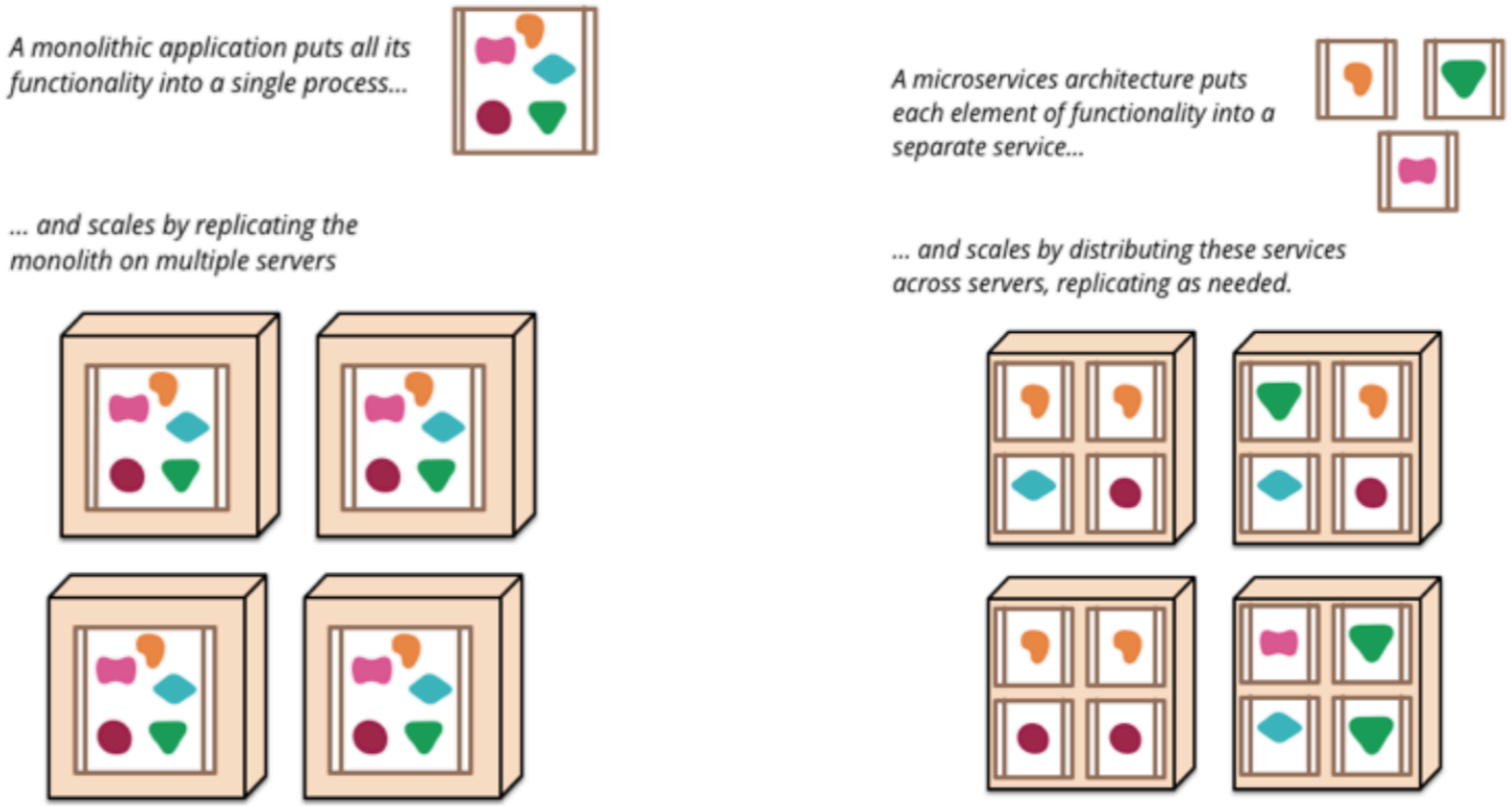

Micro-service Architecture
Event-driven SOA
SOA
Broker-based Architecture
Middleware
Distributed Object
Client-Server (Multi-tier)

Component ✚ **Connector**

- Component patterns
  - Distributed process
  - Distributed object
  - Service
  - Microservice
- Connector patterns
  - Remote procedure call
  - Stub/Skeleton of Distributed object
  - Middleware
  - Broker-based
  - Messaging
  - Event-driven
  - Service-oriented

The ultimate goal: to deliver better software faster.

From Monolithic Application to Microservices



A monolithic application puts all its functionality into a single process...

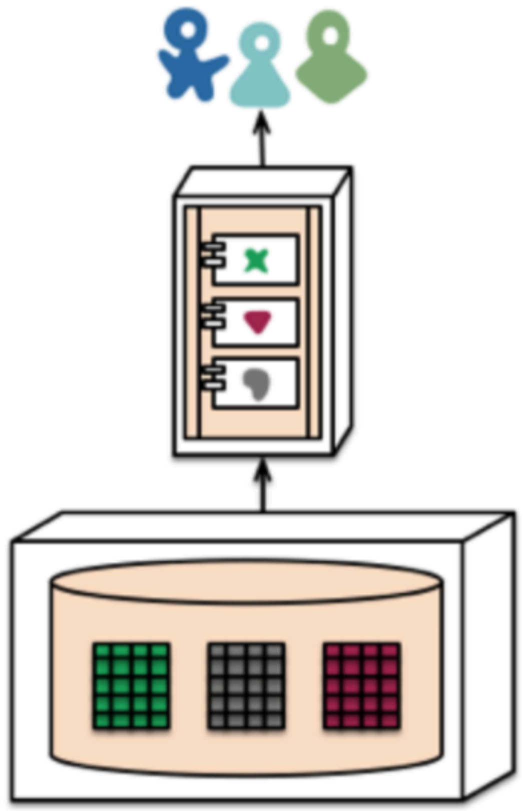
... and scales by replicating the monolith on multiple servers

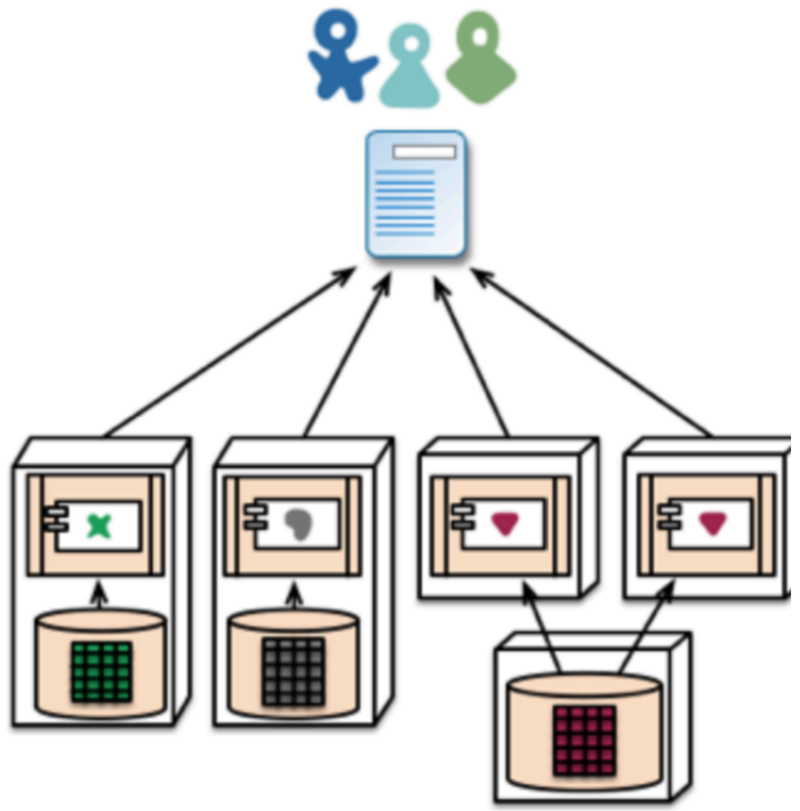
A microservices architecture puts each element of functionality into a separate service...

... and scales by distributing these services across servers, replicating as needed.

Credit: James Lewis and Martin Fowler, Microservices

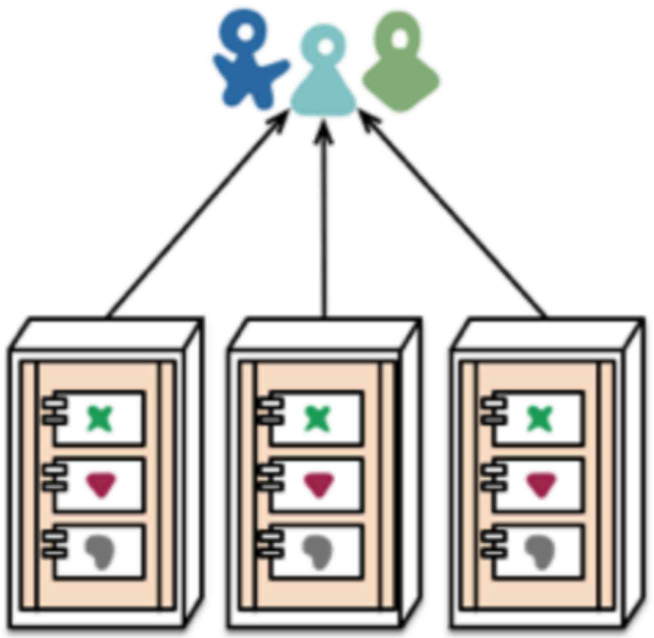
Database Deployment

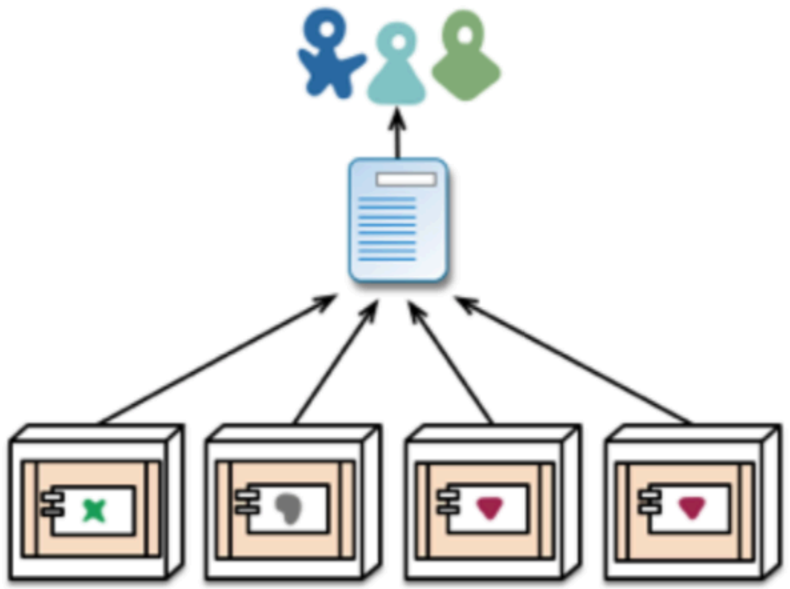

monolith - single database


microservices - application databases

Credit: James Lewis and Martin Fowler, Microservices

Module Deployement


monolith - multiple modules in the same process


microservices - modules running in different processes

Credit: James Lewis and Martin Fowler, Microservices

Four generations of microservice architecture:

(a) Container orchestration.

(b) Service discovery and fault tolerance.

(c) Sidecar and service mesh.

(d) Serverless architecture.

Credit: Jamshidi et al., Microservices-–
The Journey So Far and Challenges Ahead

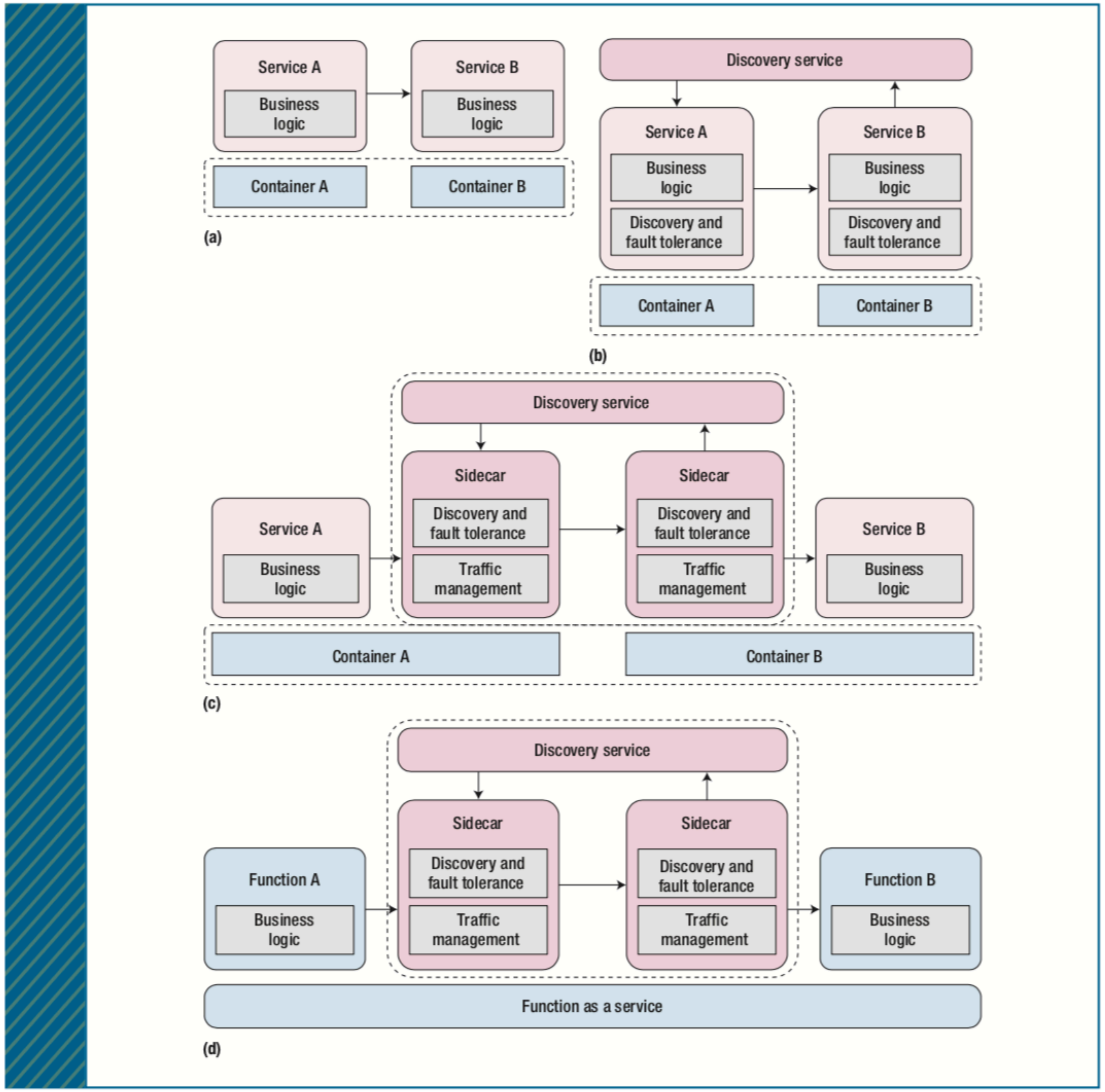
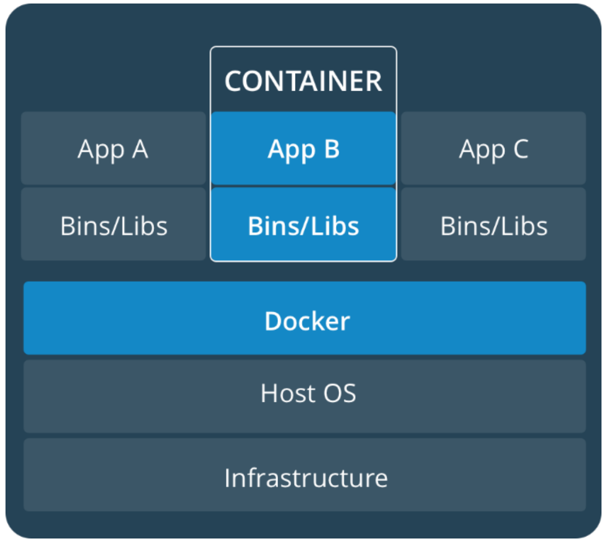
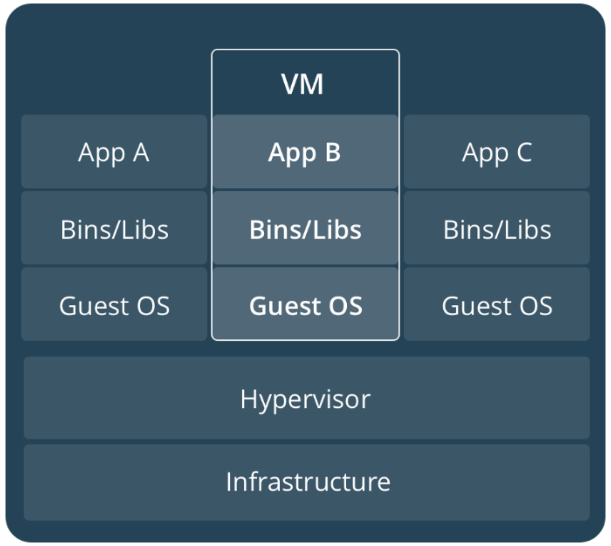


FIGURE 2. Four generations of microservice architecture. (a) Container orchestration. (b) Service discovery and fault tolerance. (c) Sidecar and service mesh. (d) Serverless architecture.

Container vs. Virtual Machine

- Containers provide a way to package software in a format that can *run* ISOLATED on a SHARED operating system.
  - Libraries and settings required to make the software work
  - Lightweight, self-contained, standard, secured systems
  - Guarantees that software will always run the same





Container vs. VM

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware, containers are more portable and efficient.