Software Architecture 1

Software Engineering
Prof. Maged Elaasar

Categories of Architecture Patterns

1

Application Landscape

2

Application Structure

3

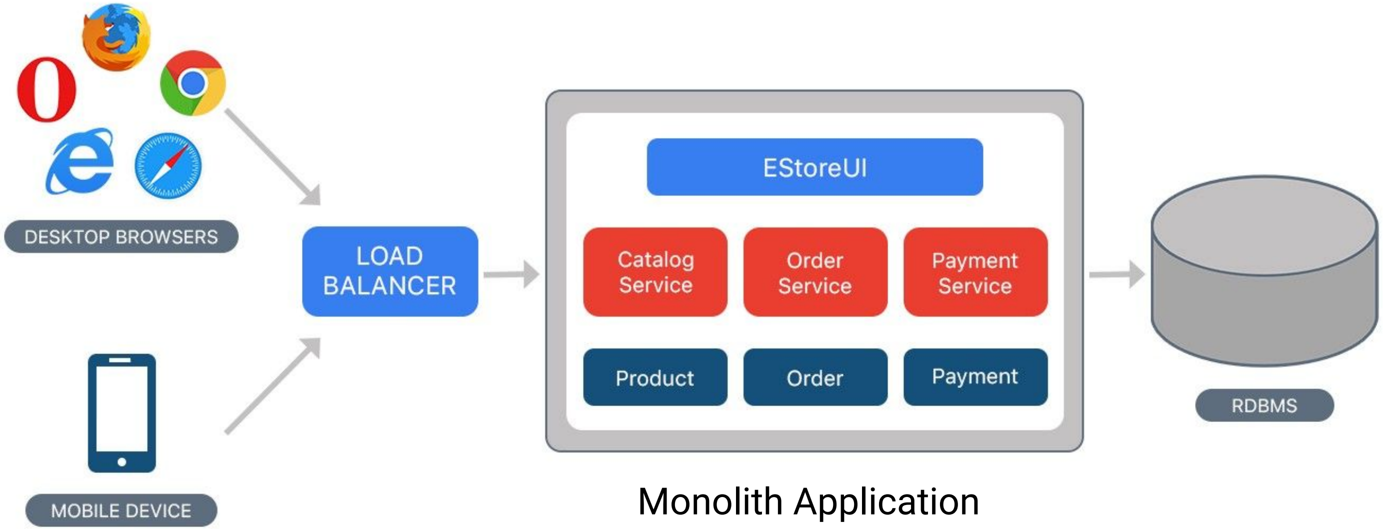
User Interface

1. Application Landscape Patterns

- Monolith
- N-tier
- Service-oriented
- Microservices
- Serverless
- Peer-to-peer

Monolith

- A single application
- All functionalities are included



Monolith Application

Monolith Examples

1. **Facebook** had its first backend developed as a monolithic server in PHP
   a. As it grew in popularity, they opted to go through continuous scaling instead of starting over and losing valuable time.
   b. Eventually they migrated to microservices

2. **Reddit** has a simple monolith with some limited media presentation features.
   a. All different components of the application are tightly integrated and deployed together.
   b. Includes web frontend, backend servers, database, and search engine all in one package.
   c. This architecture simplifies the deployment and scaling of the application,
   d. But it makes it more difficult to make changes or add new features

Monolith

1

Advantages

- Easy to understand/implement/test
- Easy to deploy
- Ideal to start with

2

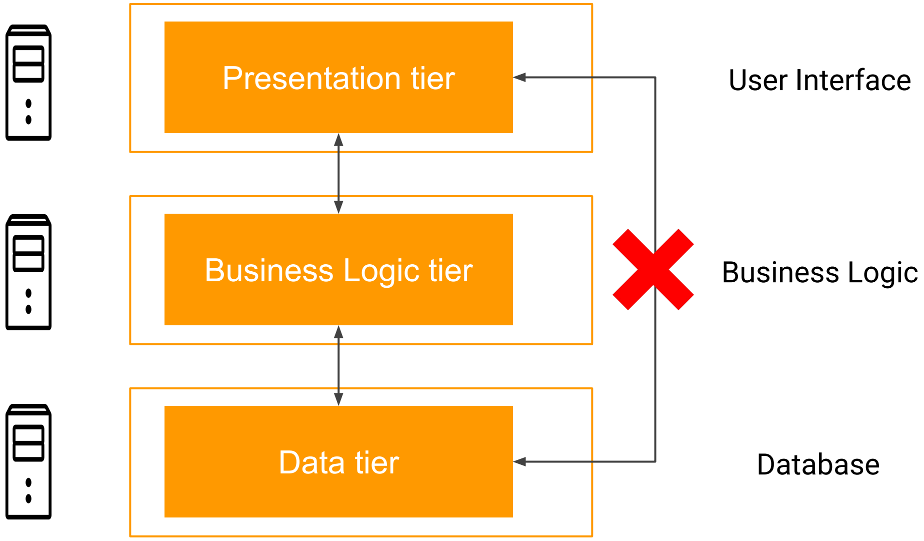
Disadvantages

- Tight coupling between functions
- Easily leads to complex code

A monolith is NOT necessarily
synonymous with badly structured code

N-Tier

- Multiple tiers
- Tier performs specific task
- Tiers can be physically separated
- Different from layers
- Split across technical boundaries



3-Tier Architecture

N-Tier

1

Advantages

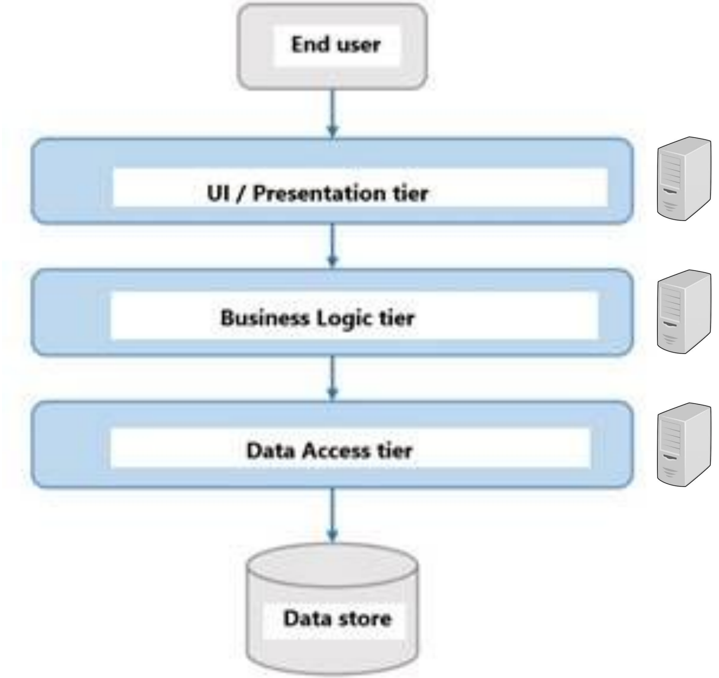
- Independent development on layers
- Scalability of each layer

2

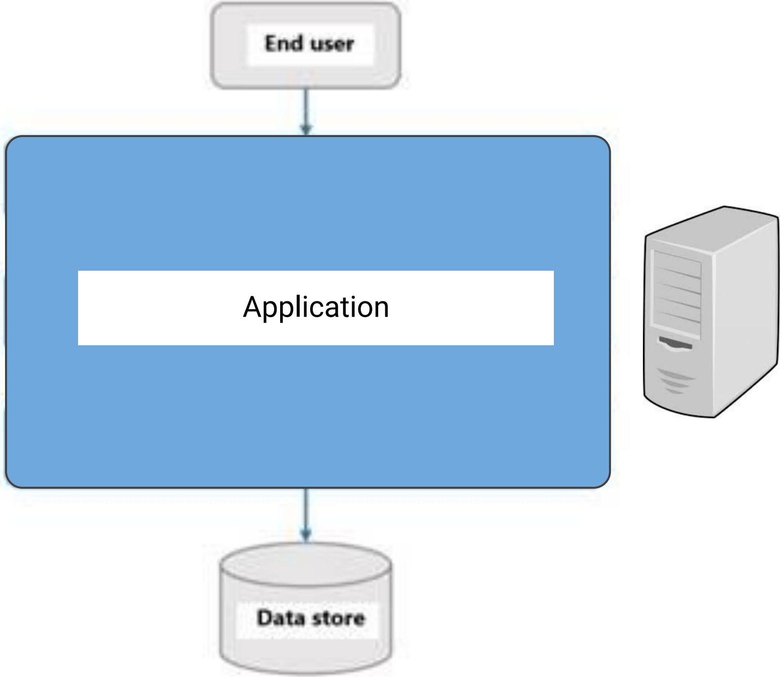
Disadvantages

- Changes ripple through layers

N-Tier vs Monolith



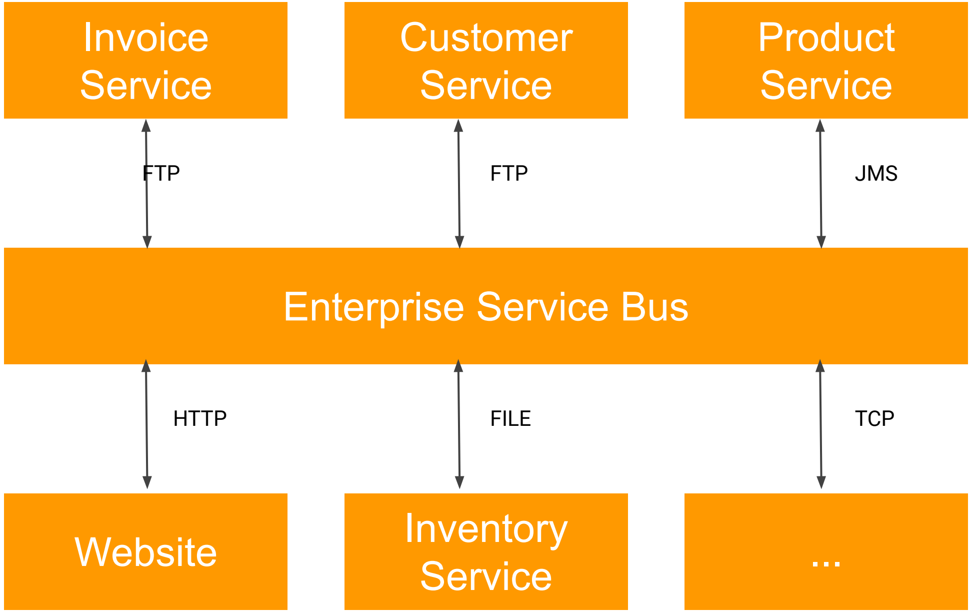
N-Tier



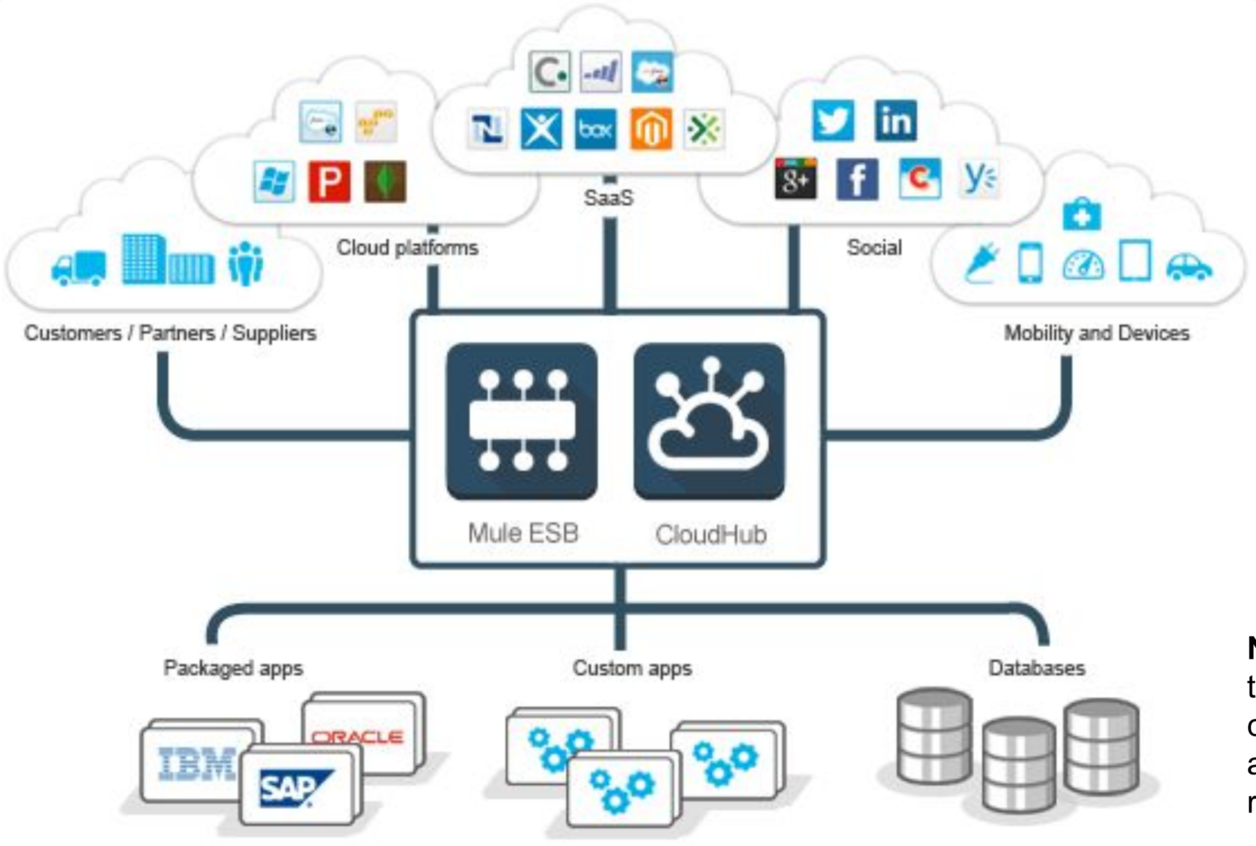
Monolith

Service-Oriented

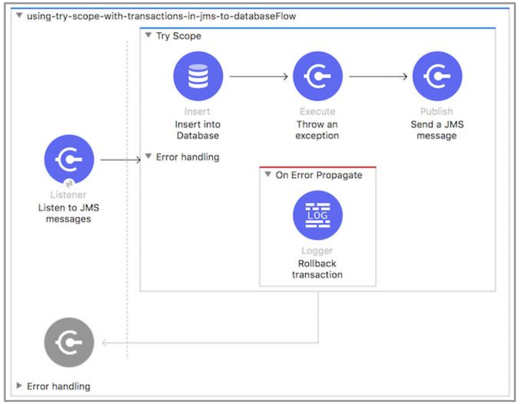
- Multiple services
- Each service is a business activity
- Service composability
- Contract standardization
- Enterprise service bus



Service-Oriented Example: Mule ESB



MuleSoft®



Non-functional requirements: faster time to market, lower costs, better consistency, increased agility, usability, and maintainability, and reduced redundancy.

Service-Oriented

1

Advantages

- Services are loosely coupled
- Better scalability
- No duplication of functionality

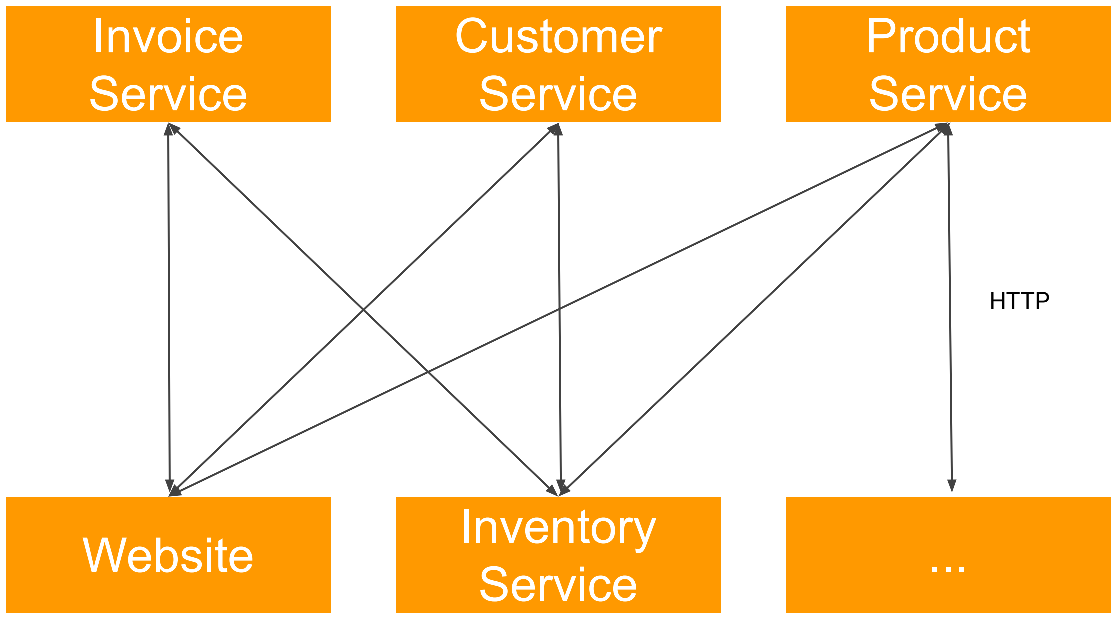
2

Disadvantages

- Increased overhead
- High cost
- Needs high-bandwidth bus

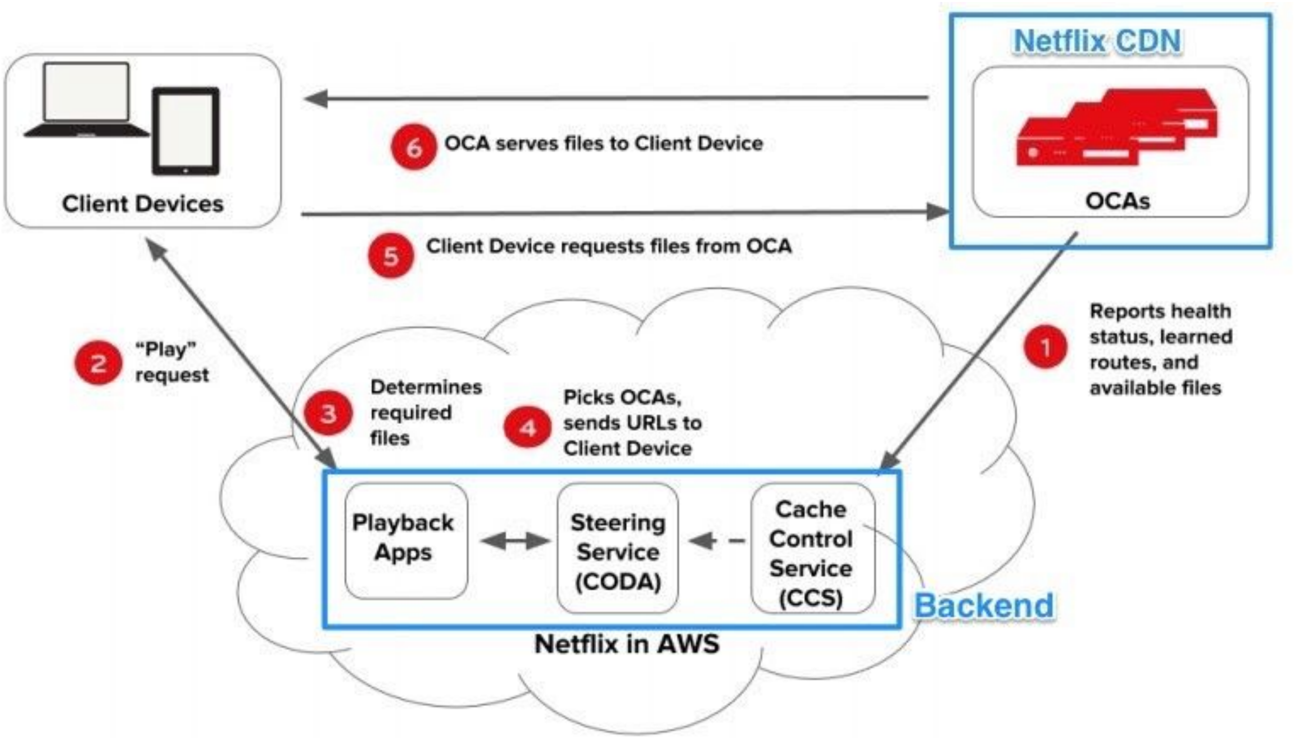
Microservices

- Multiple services
- Each service is a business activity
- Teams run the service
- No logic-heavy enterprise service bus
- Maximum automation



Microservices Example: Netflix

Video streaming system with very high availability and scalability


Netflix CDN
OCAs
Client Devices
6 OCA serves files to Client Device
5 Client Device requests files from OCA
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
1 Reports health status, learned routes, and available files
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Backend
Netflix in AWS

Non functional requirements: availability, scale, and speed

Microservices

1

Advantages

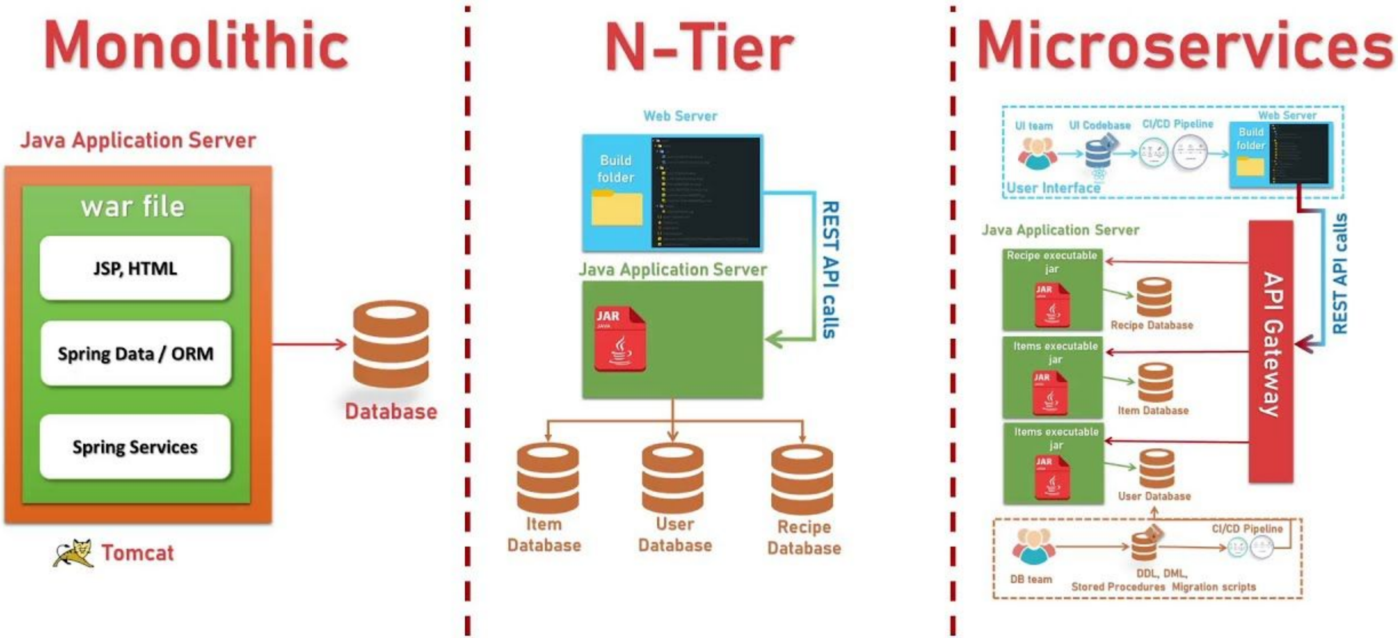
- Services are loosely coupled and easily scalable
- Increased agility
- Reliability
- Designed to handle failure

2

Disadvantages

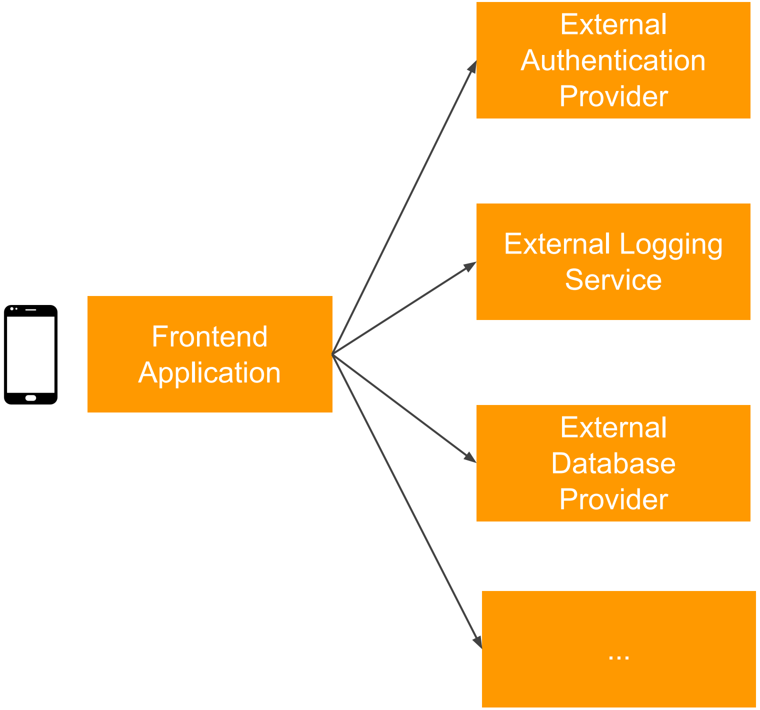
- Boundaries not always clean
- Communication patterns can become complex
- Hard to troubleshoot problems

Monolith → N-Tier →Microservices

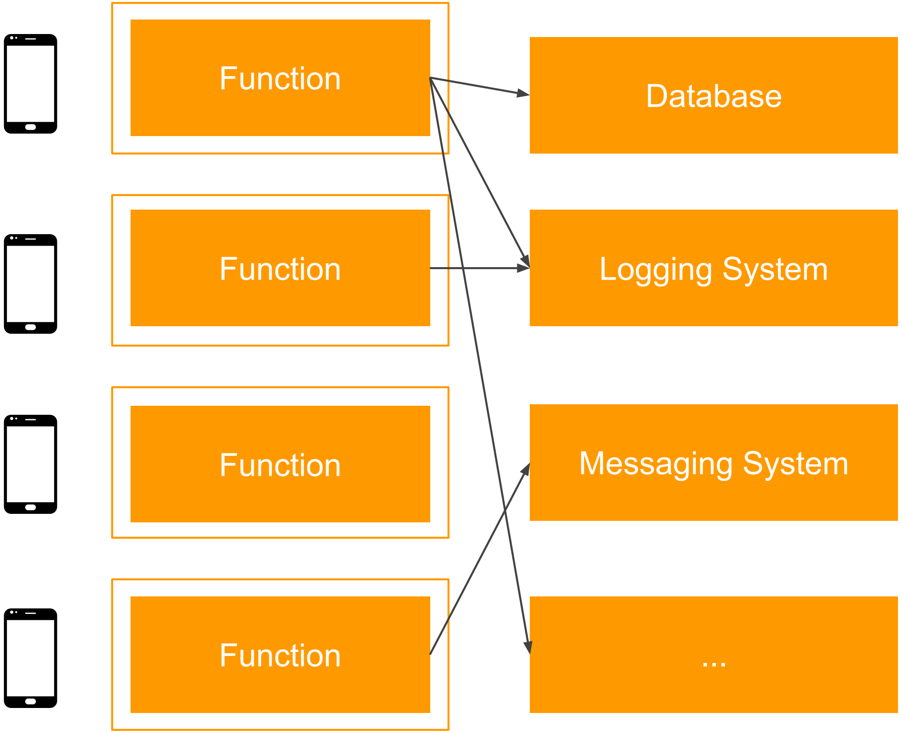


Serverless

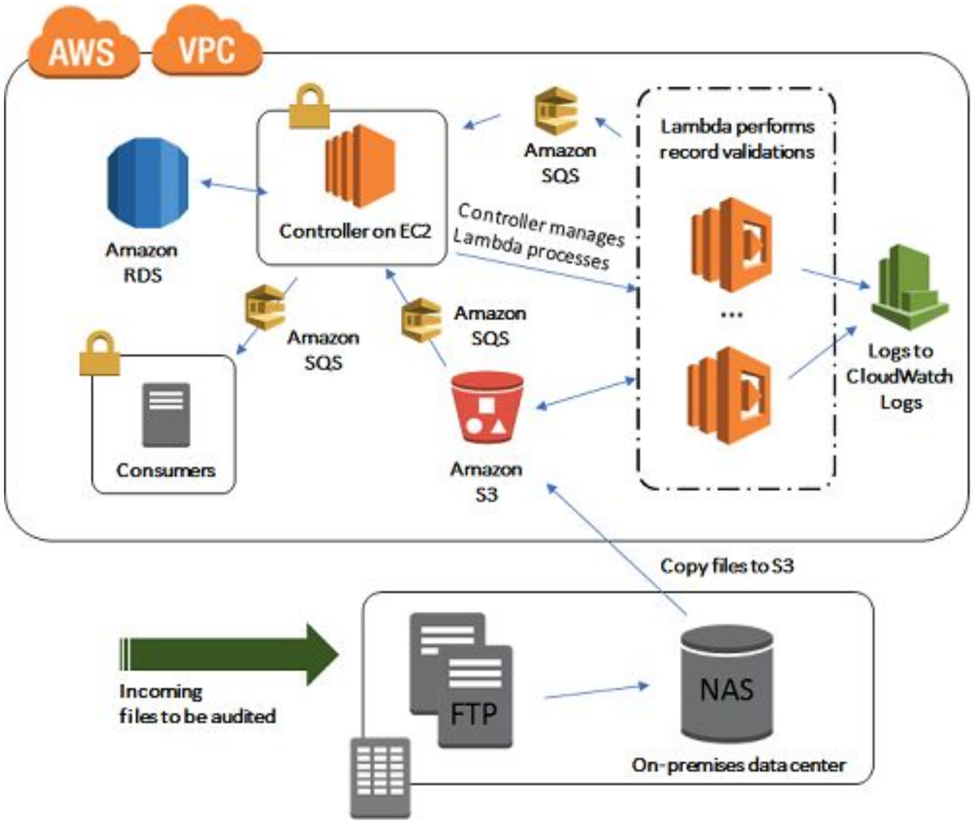
- Backend as a service



- Function as a service



Serverless Example: FINRA



FINRA®

An application to ensure market integrity through effective and efficient regulation of broker-dealers

Non functional requirements: scalability, data partitioning, monitoring, performance, cost, and maintenance requirements

Serverless

1

Advantages

- Easily scalable
- Low cost
- Easy to experiment with new ideas
- Designed to handle failure

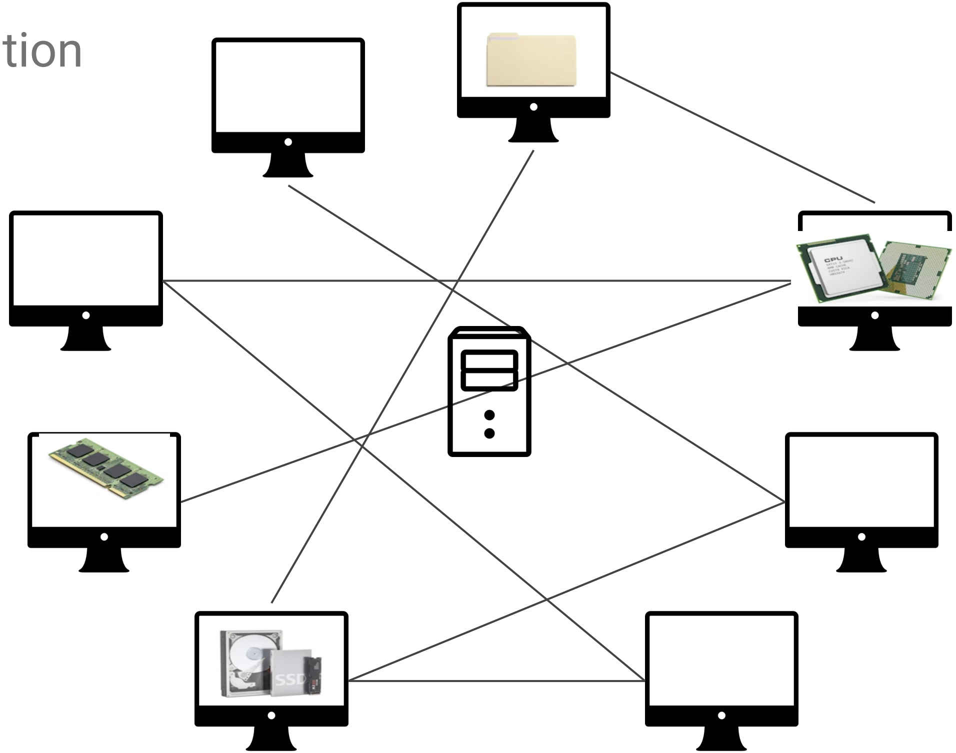
2

Disadvantages

- Vendor constraints
- Vendor lock-in
- Tricky to maintain state in memory
- Cold start of VM

Peer-to-Peer (P2P)

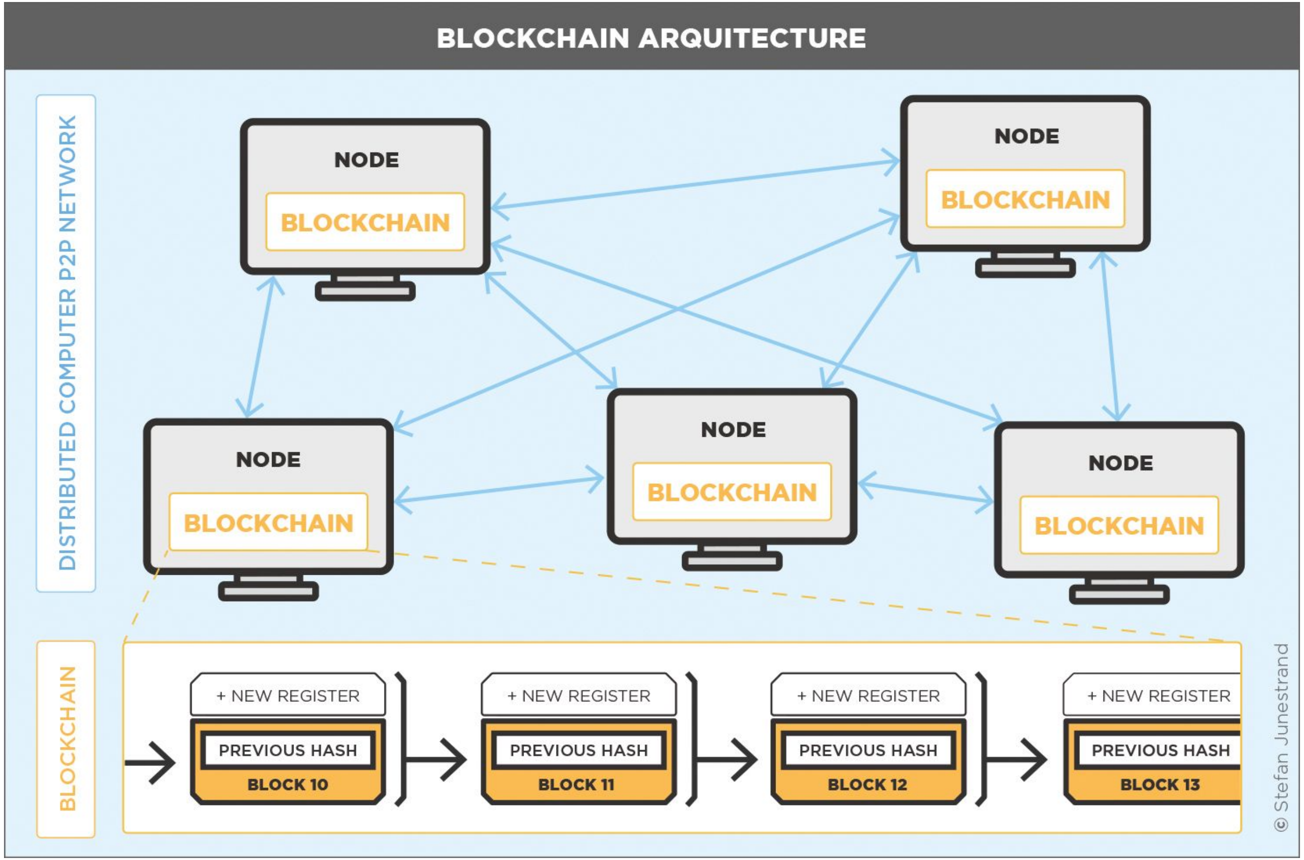
- No central server for communication
- Dynamically discoverable
  - Through a central server
- No constant connection



Peer-to-Peer Example: Blockchain

A distributed ledger with smart contracts

Non-functional requirements: Enhanced security of data, resistant to denial of service (DoS) or censorship



© Stefan Junestrand

Peer-to-Peer

1

Advantages

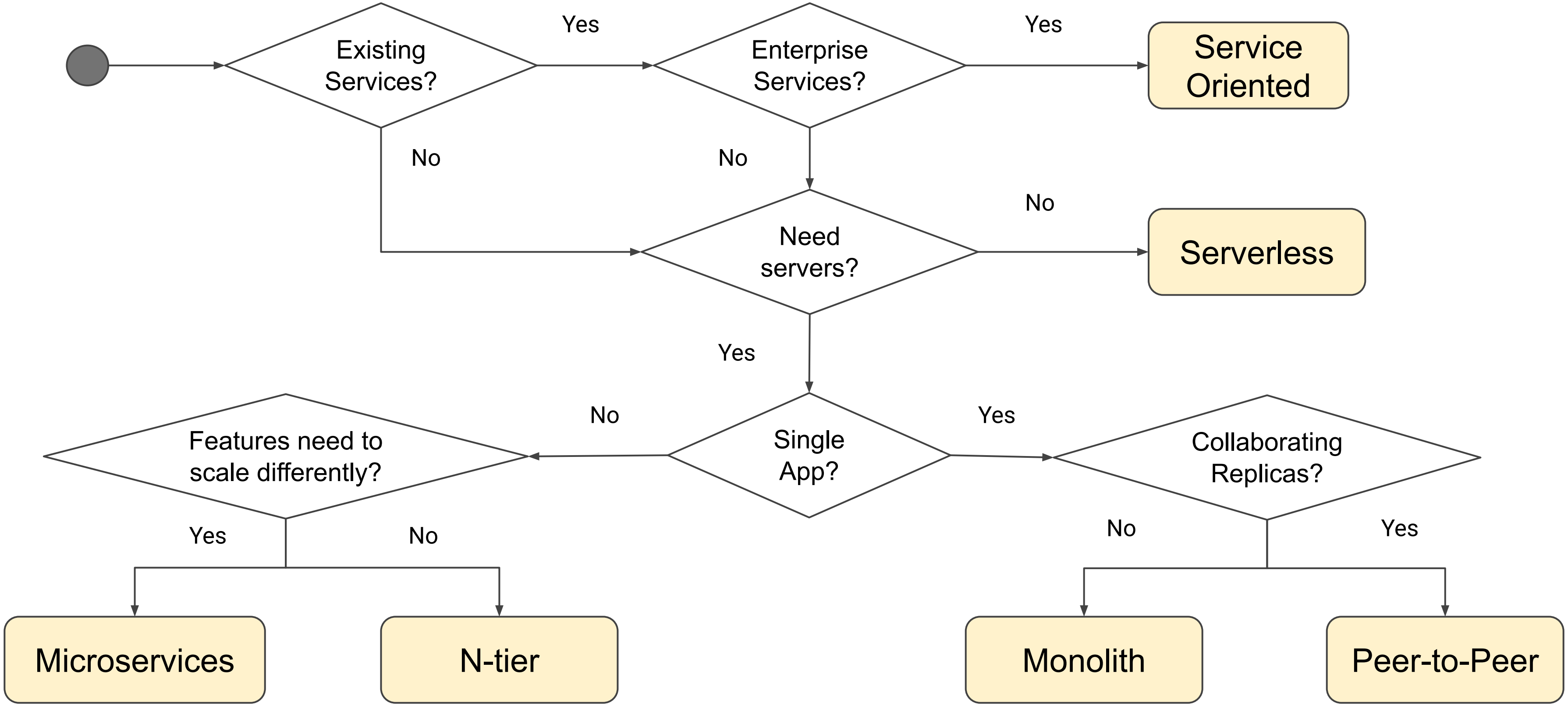
- Easy to share resources
- Cost-effective
- Easily scalable

2

Disadvantages

- Security concerns for machines
- Communication pattern is complex

Landscape Architecture Pattern Guide



Application Landscape Patterns Quiz

References

- Software Architectures: Patterns for Developers by Peter Morlion
- Amazon Web Services Essentials by Jeff Winesett
- Fundamentals of Software Architecture: An Engineering Approach by Marc Richards and Neal Ford
- Netflix Architecture by Cao Nguyen
- FINRA Architecture by Tim Griesbach
- Eclipse Architecture by Kim Moir
- Microservices vs Monolith: Choosing the Best for Your Business App