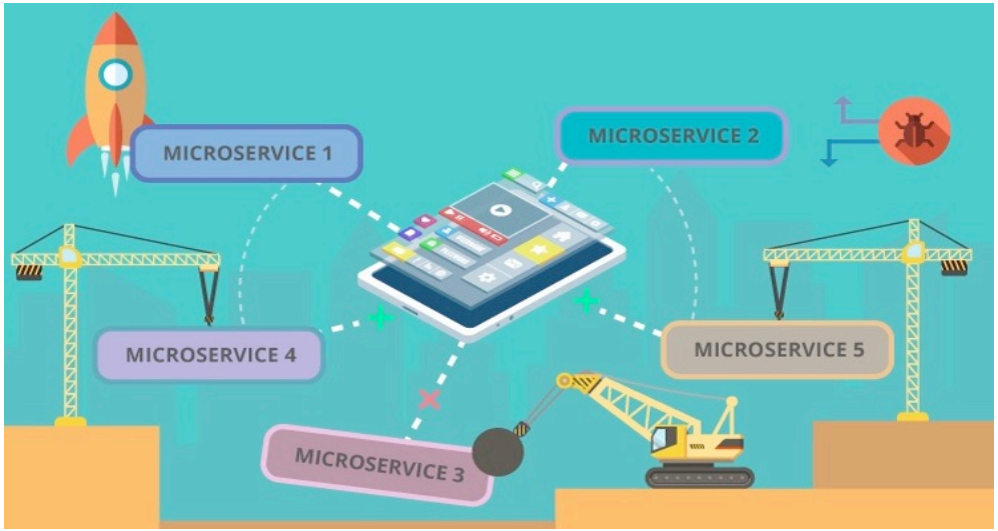
UFCFX3-15-3 Advanced Topics in Web Development II 2021/22

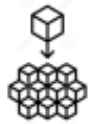


Lecture 9: Understanding Microservices

abstract

Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

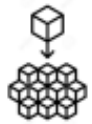
A microservice is not a layer within a monolithic application (example, the web controller, or the backend-for-frontend). Rather it is a self-contained piece of business functionality with clear interfaces, and may, through its own internal components, implement a layered architecture. From a strategy perspective, microservices architecture essentially follows the Unix philosophy of "Do one thing and do it well".



definition

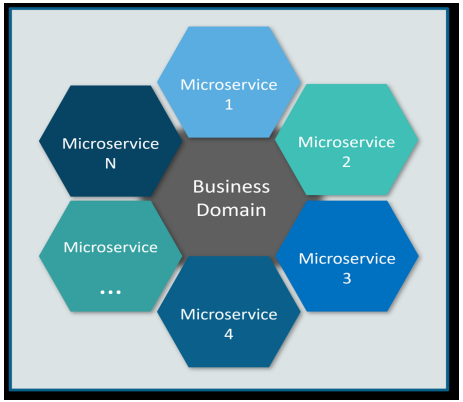
There is no single definition for microservices. A consensus view has evolved over time in the industry. Defining characteristics of microservices include include:

- Per Martin Fowler and other experts, services in a microservice architecture (MSA) are often processes that communicate over a network to fulfill a goal using technology-agnostic protocols such as HTTP.
- Services in a microservice architecture are independently deployable.
- Services are organized around business capabilities.
- Services can be implemented using different programming languages, databases, hardware and software environment, depending on what fits best.
- Services are small in size, messaging-enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes.

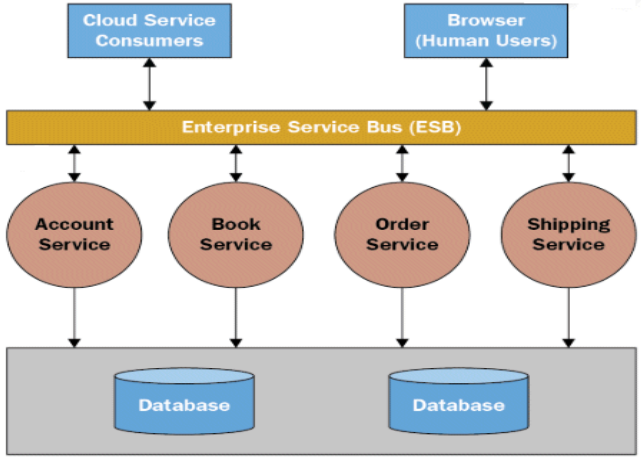


microservices is a soa *architectural style*

- In a microservices application architectural style, an application is composed of many discrete, network-connected components, termed microservices.



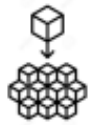
- The microservices architectural style is evolution of the SOA (Services Oriented Architecture) architectural style.



soa offers many (enterprise) advantages

*A **service-oriented architecture** (SOA) is not just an architecture of services from a technology perspective; it includes the policies, practices, and frameworks by which we ensure the right services are provided and consumed. The basic principles of SOA include independence of vendors, products, and technologies.*

- ✓ Moving away from a monolithic and heavily integrated portal applications to a modular, ***service-oriented architecture***
- ✓ Less in-house hardware and data storage, and more distributed hardware and data storage leveraging cloud services
- ✓ More distributed software portal applications
- ✓ Less integrated management of individual portal application, more autonomy for applications



microservices

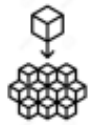
An SOA variation, microservice architecture is an approach to developing a single application as a suite of small services. Each service:

- Runs its own processes
- Can be written in any programming language and is platform agnostic
- Is independently deployable and replaceable
- May use different storage technologies
- Requires a minimum of centralized management

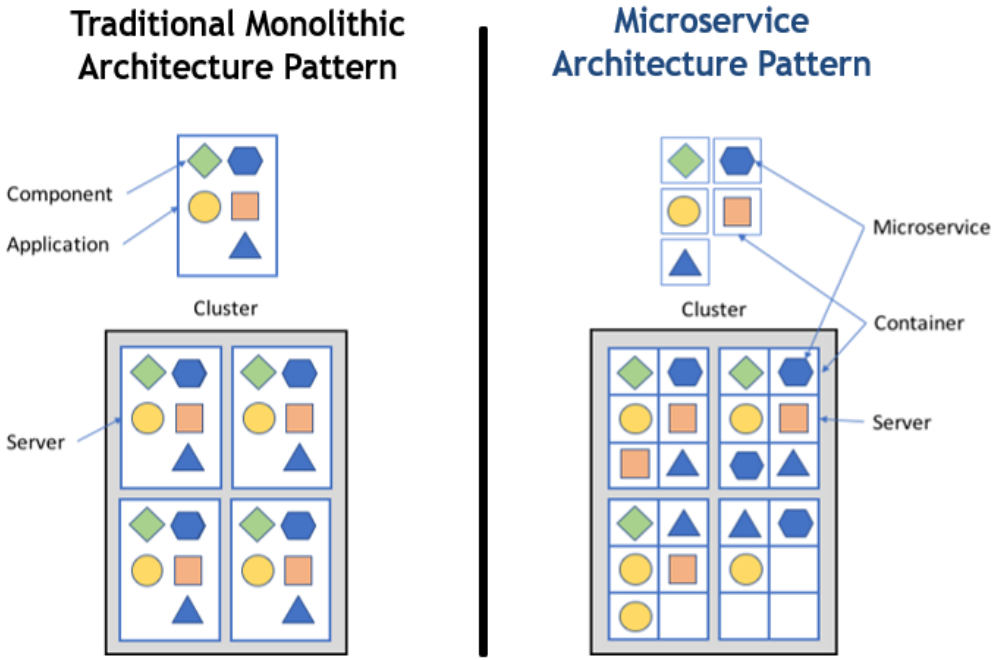
Large companies, such as Amazon, eBay, and Netflix, have already adopted microservice architecture as their main architecture.

Modular Space System Analogy

Rather than using a large, complex, single satellite, agencies are migrating to a suite of connected, independent, low-cost, and replaceable small satellites, each with its own function (e.g., communication, imagery, sensors). This decomposition and migration to autonomous modules offers these agencies ease of functionality, ease of replaceability, and ease of modernization with newer modules.



from monolithic to microservices architecture (1)



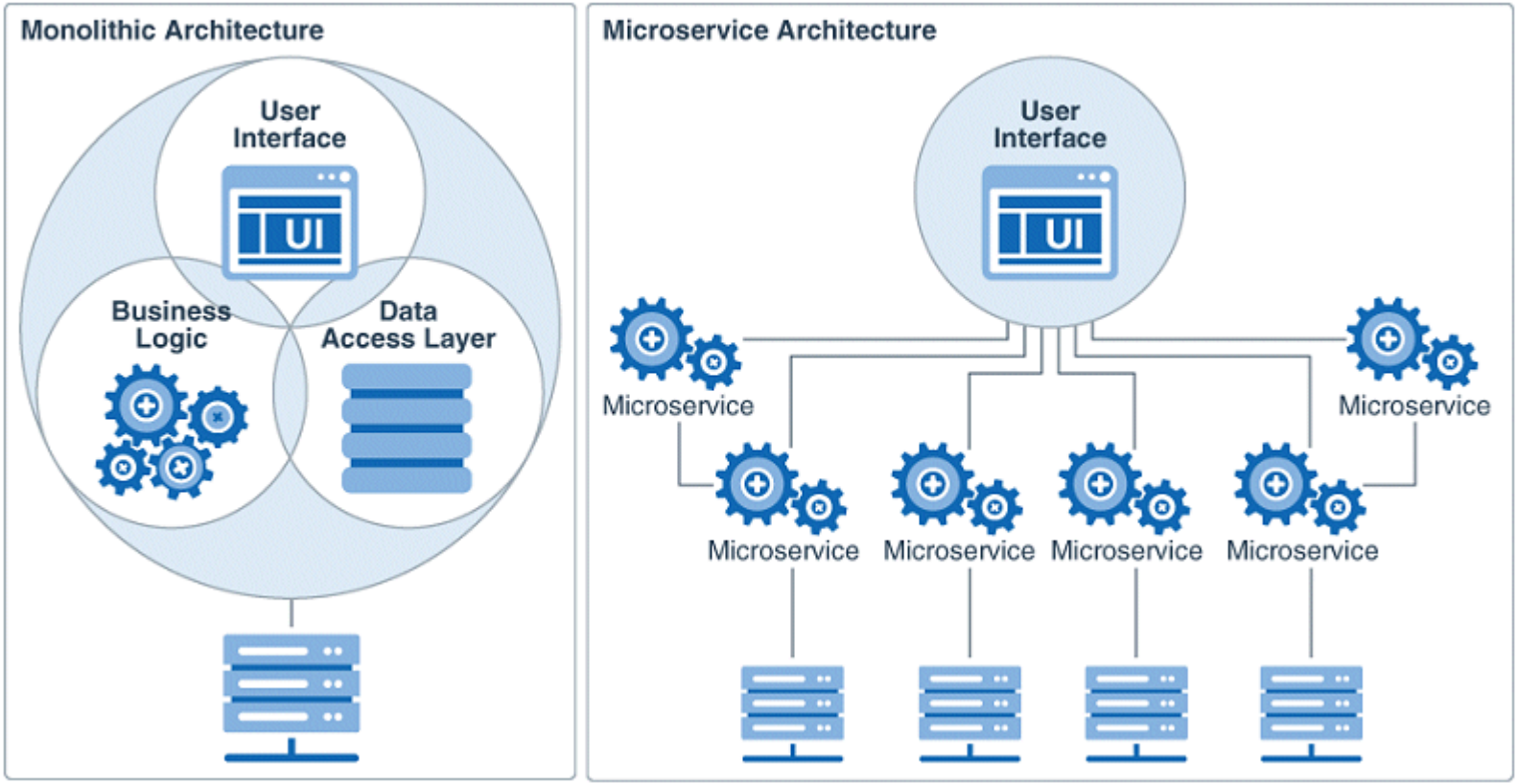
"Monoliths and microservices are not a simple binary choice.

Both are fuzzy definitions that mean many systems would lie in a blurred boundary area among the two"

-Martin Fowler

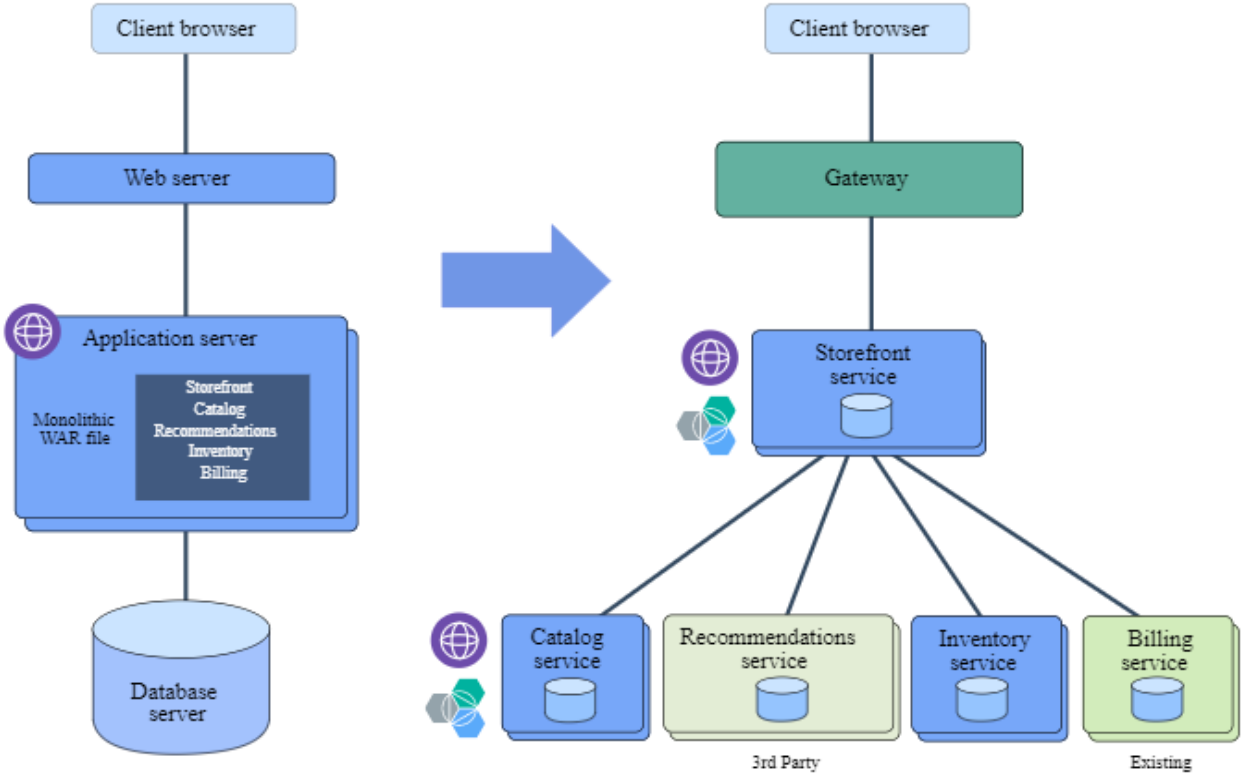
Martinfowler.com

from monolithic to microservices architecture (2)



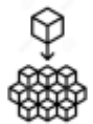
from monolithic to microservices architecture (3)

e-commerce tiered example

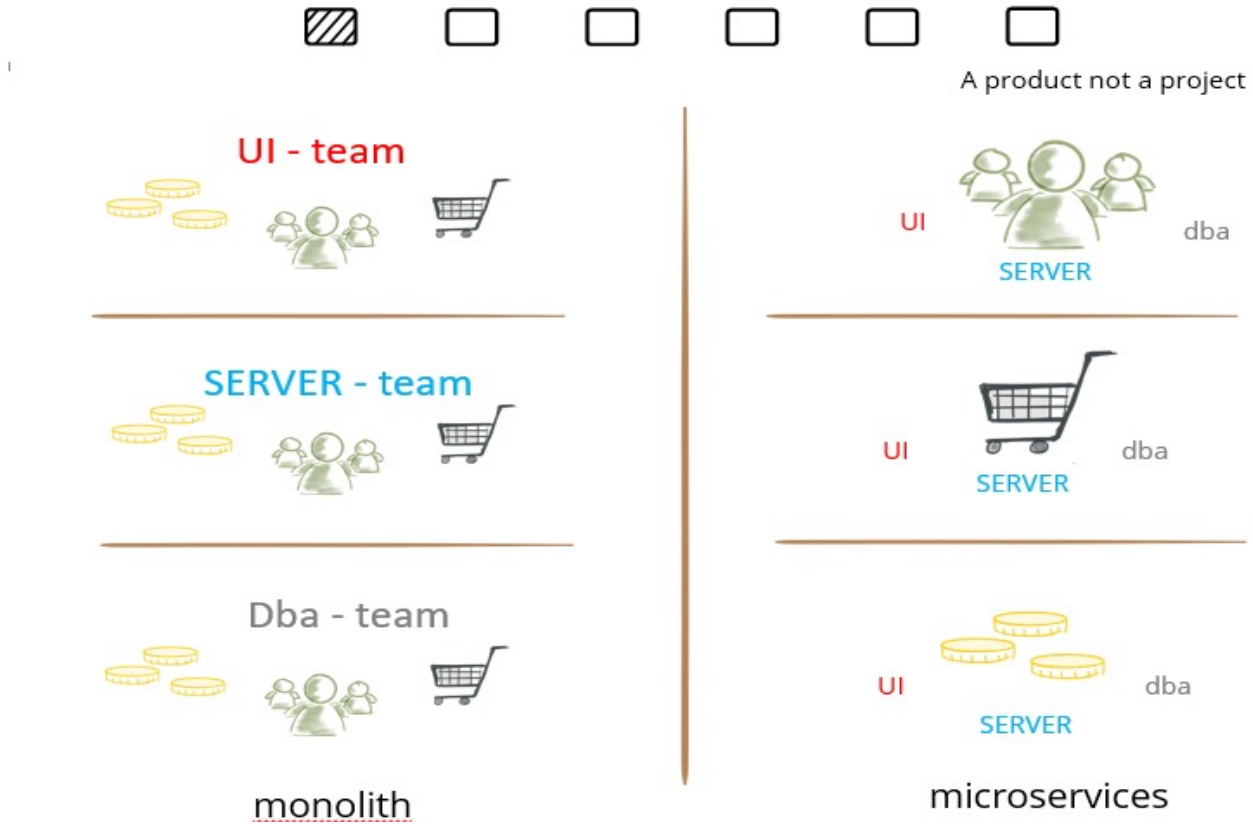


principles of microservices

- modularity
- autonomous
- hide implementation details
- automation
- stateless
- highly observable



principles of microservices: modularity

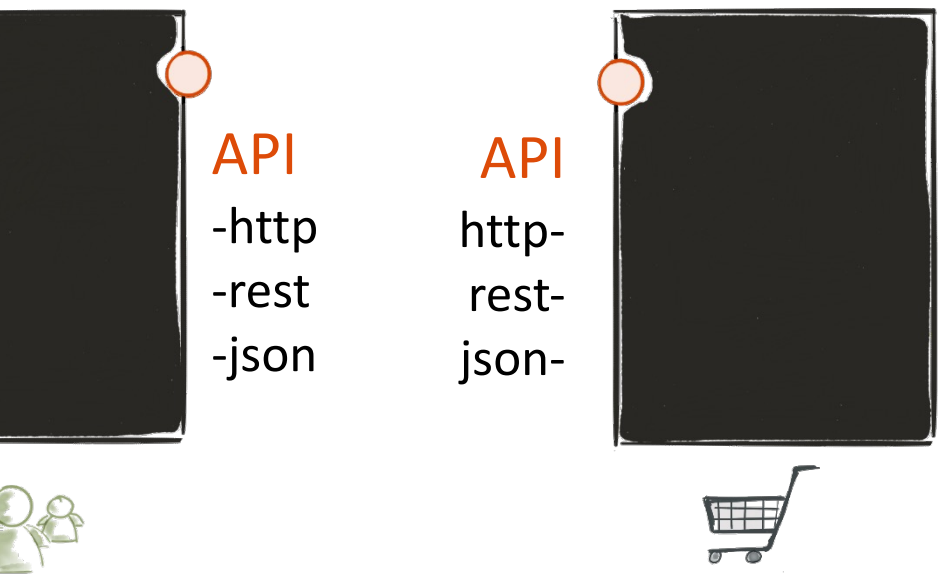


principles of microservices: autonomous

monolith

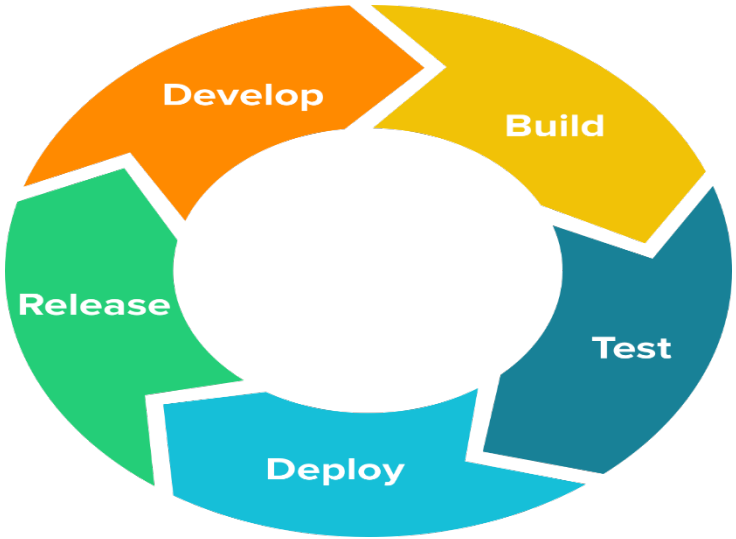
microservices

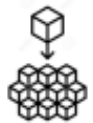
principles of microservices: hide implementation details



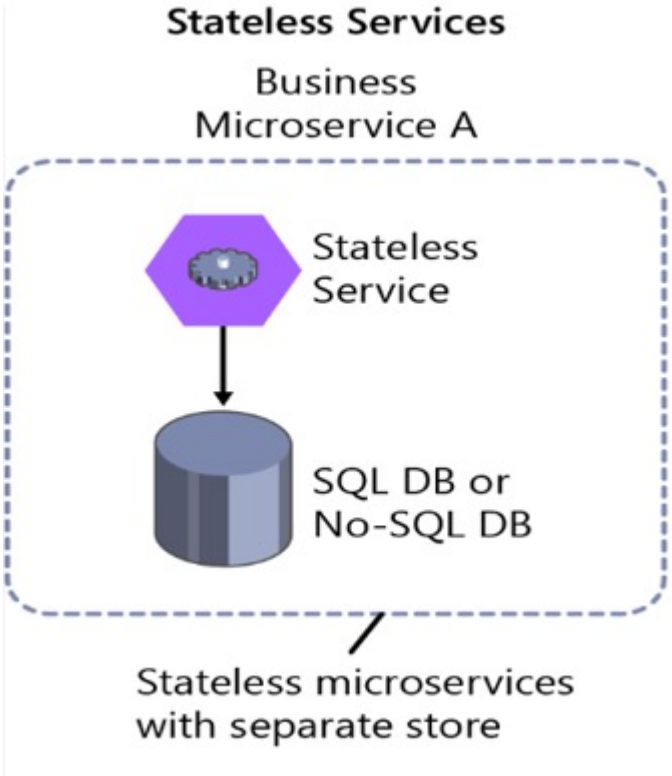
principles of microservices: automation

- continuous integration
- continuous deployment





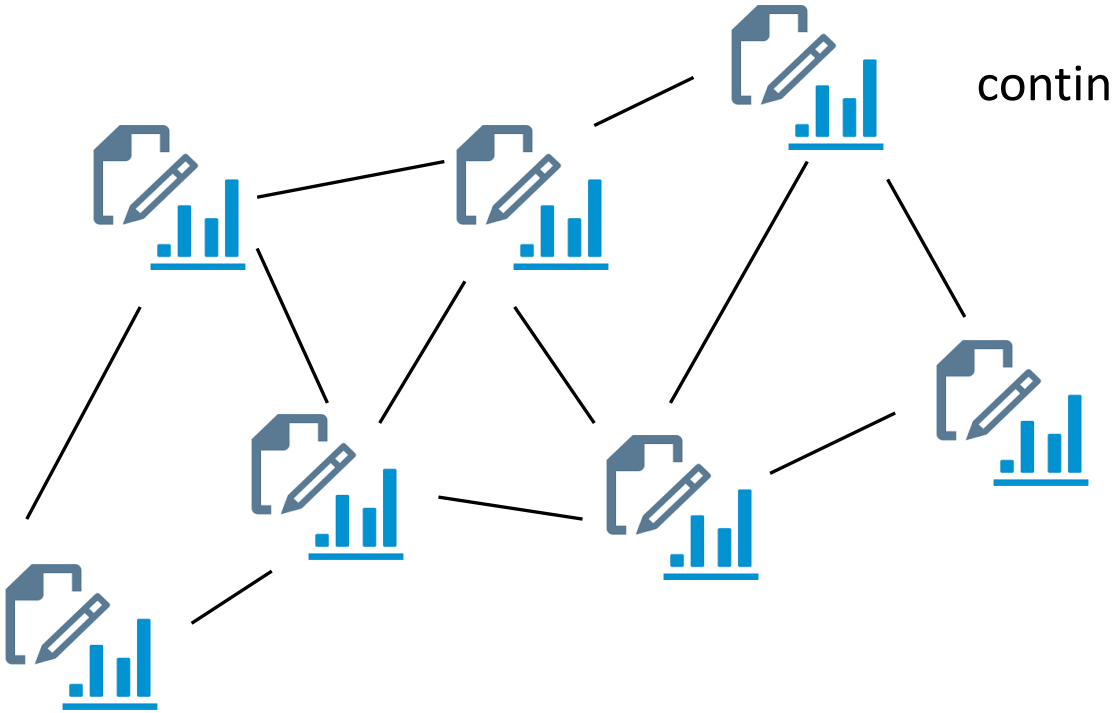
principles of microservices: stateless



HATEOAS
(Hypertext As The Engine Of Application State)

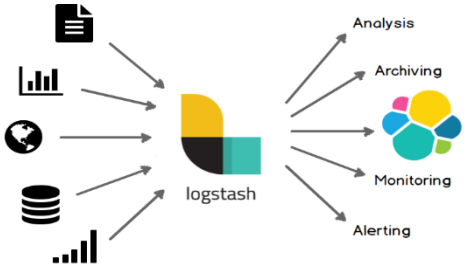
(see Richardson Maturity Model (EXAM!)

principles of microservices: highly observable

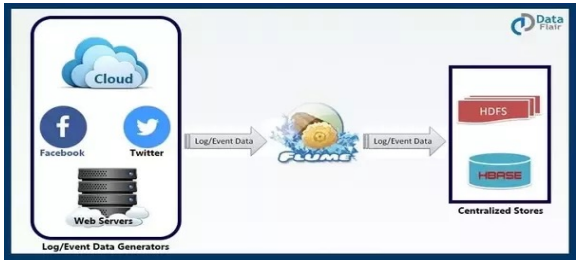


continuous logging

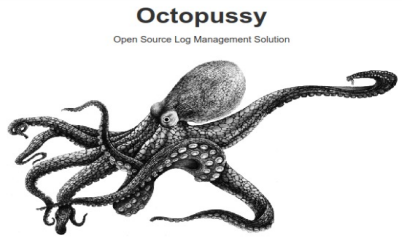
principles of microservices: highly observable



logstash

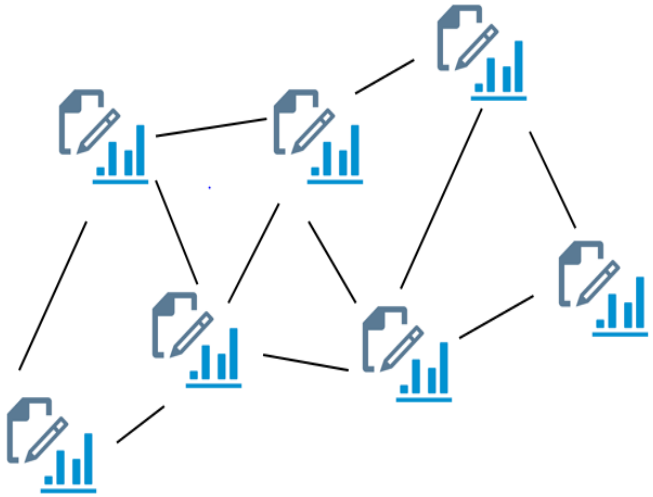


apache flume



octopussy

centralized logging

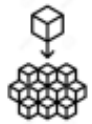


characteristics of microservices vs. monolithic architecture (1)

Characteristic	Microservices Architecture	Monolithic Architecture
Unit design	The application consists of loosely coupled services. Each service supports a single business task.	The entire application is designed, developed, and deployed as a single unit.
Functionality reuse	Microservices define APIs that expose their functionality to any client. The clients could even be other applications.	The opportunity for reusing functionality across applications is limited.
Communication within the application	To communicate with each other, the microservices of an application use the request-response communication model. The typical implementation uses REST API calls based on the HTTP protocol.	Internal procedures (function calls) facilitate communication between the components of the application. There is no need to limit the number of internal procedure calls.
Technological flexibility	Each microservice can be developed using a programming language and framework that best suits the problem that the microservice is designed to solve.	Usually, the entire application is written in a single programming language.
Data management	Decentralized: Each microservice may use its own database.	Centralized: The entire application uses one or more databases.

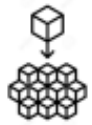
characteristics of microservices vs. monolithic architecture (2)

Characteristic	Microservices Architecture	Monolithic Architecture
Deployment	Each microservice is deployed independently, without affecting the other microservices in the application.	Any change, however small, requires redeploying and restarting the entire application.
Maintainability	Microservices are simple, focused, and independent. So the application is easier to maintain.	As the application scope increases, maintaining the code becomes more complex.
Resiliency	The application functionality is distributed across multiple services. If a microservice fails, the functionality offered by the other microservices continues to be available.	A failure in any component could affect the availability of the entire application.
Scalability	Each microservice can be scaled independently of the other services.	The entire application must be scaled, even when the business requirement is for scaling only certain parts of the application.



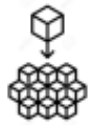
advantages of microservices (1)

- Software built as microservices can be broken down into multiple component services, so that each of these services can be deployed and then redeployed independently without compromising the integrity of an application. That means that microservice architecture gives developers the freedom to independently develop and deploy services.
- Better fault isolation; if one microservice fails, the others will continue to work.
- Code for different services can be written in different languages.
- Easy integration and automatic deployment; using open-source continuous integration tools such as Jenkins, etc.
- The microservice architecture enables continuous delivery.
- Easy to understand since they represent a small piece of functionality, and easy to modify for developers, thus they can help a new team member become productive quickly.



advantages of microservices (2)

- The code is organized around business capabilities.
- Scalability and reusability, as well as efficiency. Easy to scale and integrate with third-party services.
- Components can be spread across multiple servers or even multiple data centres.
- Work very well with containers, such as Docker.
- Complement cloud activities.
- Microservices simplify security monitoring because the various parts of an app are isolated. A security problem could happen in one section without affecting other areas of the project.
- Increase the autonomy of individual development teams within an organization, as ideas can be implemented and deployed without having to coordinate with a wider IT delivery function.

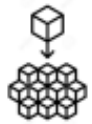


microservices & the cloud (scaling & elasticity)

- Greater efficiencies as they are typically paired with auto-scalers and load balancers
- Easily scalable in three ways:
  - by decomposition into simple functions (microservices)
  - by cloning microservices on demand
  - by partitioning data across a cluster
- Services can be cloned or decommissioned based on user requests in real-time

Auto-scaling - A cloud computing service feature that automatically adds or removes computing resources based on actual demand

Load balancing -The process of distributing workload evenly across computing resources in a cloud computing environment



cloud based component / service integration