Monolithic
SOA
Microservices

Software Architecture Evolution

Table of Contents

What is Software Architecture Pattern?

Software Architecture Pattern

Software architecture pattern help define the basic characteristics and behavior of an application.

The success of any application or system depends on the architecture pattern you use. Architecture patterns not only guide designers and developers on how to design and structure the software components, but also determine the ways in which those components should interact.

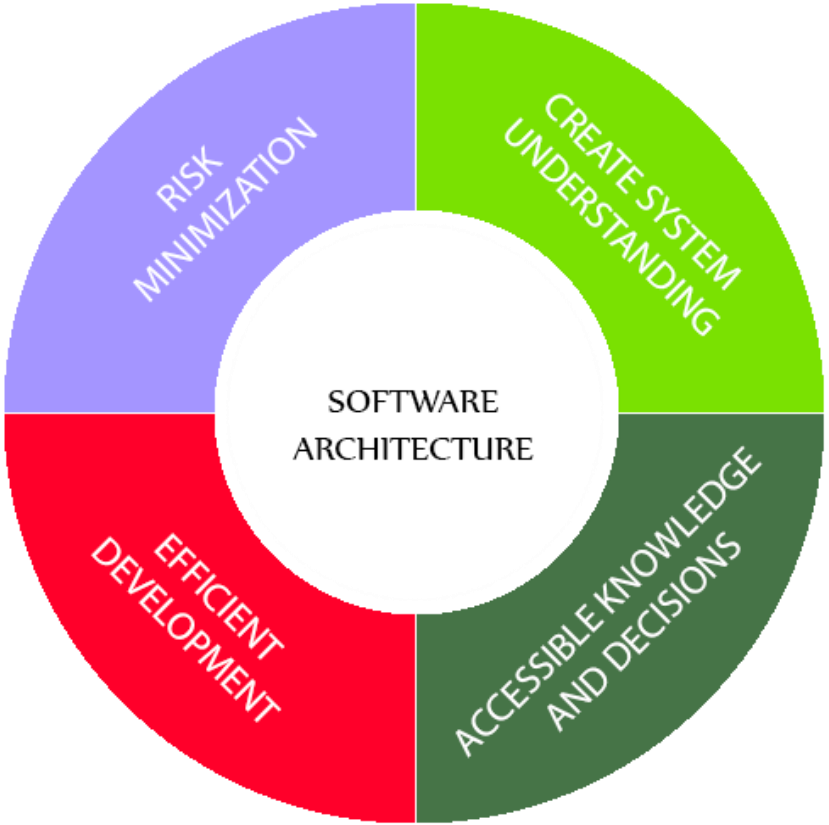
The software architecture represents the earliest software design decisions. These design decisions are the most critical to get right and the most difficult to change downstream in the system development life cycle.

Why Software Architecture Patterns are required?

Software Architectures Patterns are required for the success of any software application.

Applications lacking a formal architecture are generally tightly coupled, brittle, difficult to change, and without a clear vision or direction.

As a result, it is very difficult to determine the architectural characteristics of the application without fully understanding the inner-workings of every component and module in the system.



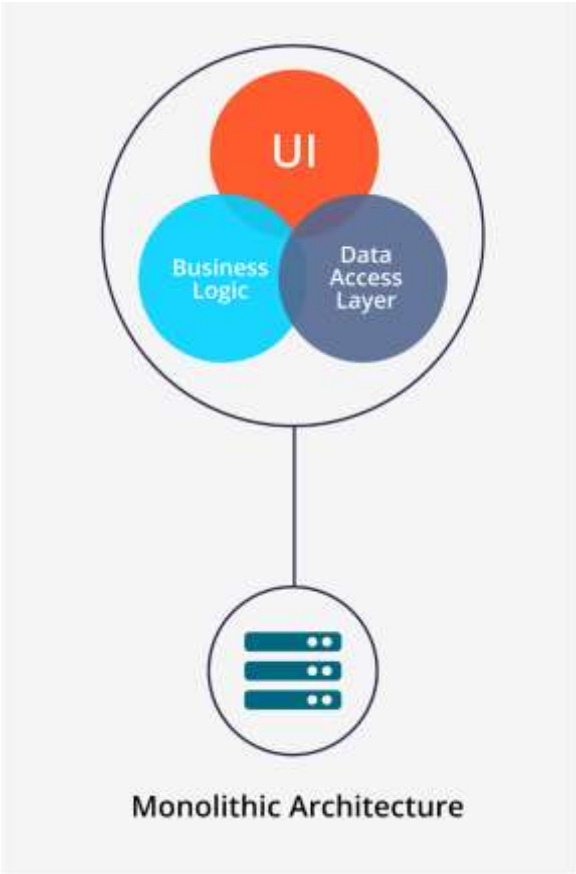
Forces Behind Choosing Best Architecture Pattern

- The application must be easy to understand and modify
- You want to practice continuous deployment of the application
- You must run multiple copies of the application on multiple machines in order to satisfy scalability and availability requirements
- You want to take advantage of emerging technologies (frameworks, programming languages, etc)
- New team members must quickly become productive

Monolithic Architecture (Traditional Web Application Architecture)

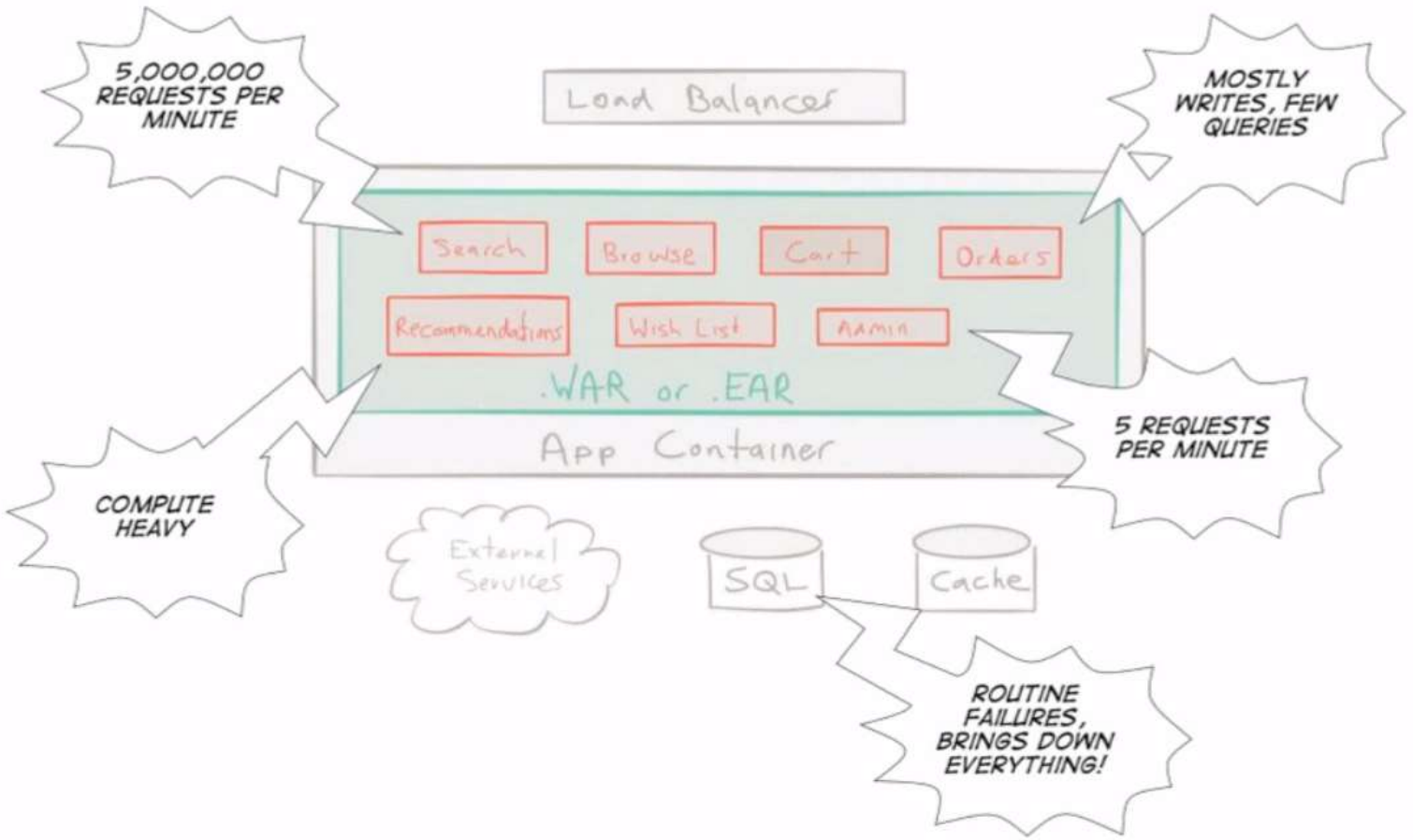
Monolithic architectures are the ones running on a single application layer that tends to bundle together all the functionalities needed by the architecture.

At the architectural level, this is the simplest form of architecture simply because it doesn't involve as many actors as other architectural styles.



Advantages of Monolithic Architecture

- Simple to develop - the goal of current development tools and IDEs is to support the development of monolithic applications
- Simple to test
- Simple to deploy - you simply need to deploy the WAR file (or directory hierarchy) on the appropriate runtime
- Simple to scale - you can scale the application by running multiple copies of the application behind a load balancer
- Can be used for small and midsized applications.


5,000,000 REQUESTS PER MINUTE
Load Balancer
MOSTLY WRITES, FEW QUERIES
Search
Browse
Cart
Orders
Recommendations
Wish List
Admin
.WAR or .EAR
App Container
5 REQUESTS PER MINUTE
COMPUTE HEAVY
External Services
SQL
Cache
ROUTINE FAILURES, BRINGS DOWN EVERYTHING!

Disadvantages of Monolithic Architecture

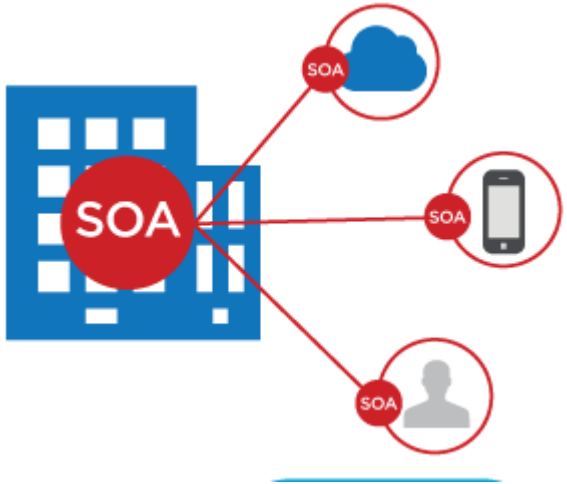
- Difficult to handle when new members introduced in team
- Overload IDE
- Overloaded web container
- Continuous deployment is difficult
- Scaling the application can be difficult
- Require a long term commitment to a technology stack
- Single point of failure

Service Oriented Architecture

A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components (Web clients, Mobile Apps).

A service is a well-defined and self-contained functionality.

Services communicate with each other to perform some activity through a predefined communication protocol over a network.



Service Oriented Architecture

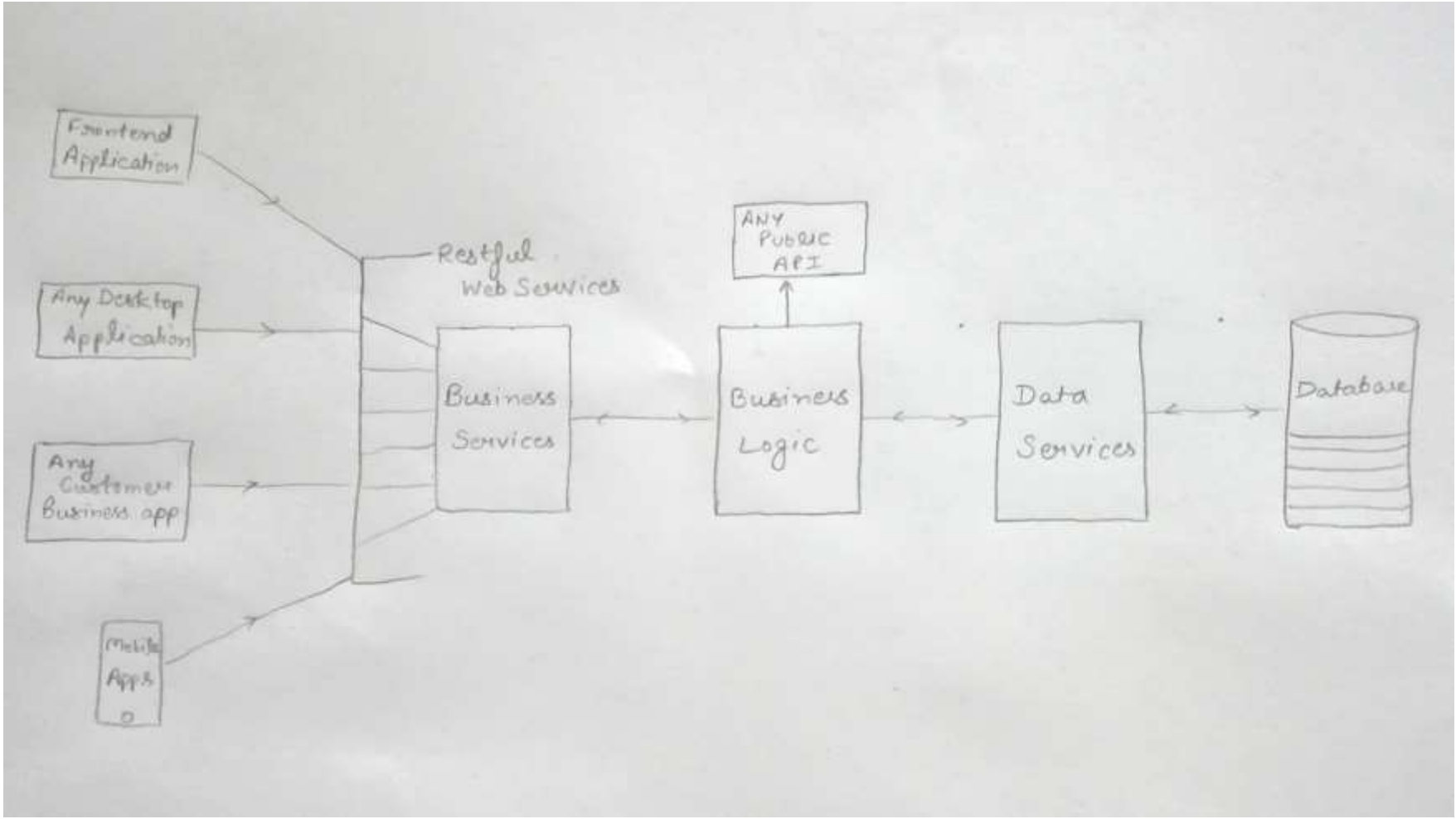
Services are loosely-coupled

i.e., a service need not know the technical details of another service that it is interacting with.

Basically, SOA consists of a

- service consumer
- service provider.

The basic principles of service oriented architecture are independent of vendors, products and technologies. Consumer request for some service, and provide returns the result for the requests.


Frontend Application
Any Desktop Application
Any Customer Business app
Mobile Apps
Restful Web Services
Business Services
ANY PUBLIC API
Business Logic
Data Services
Database

Advantages of SOA Architecture

- Service Reusability
- Easy Maintainability
- Greater Reliability
- Improved Software Quality
- Improved Scalability and Availability
- Increased Productivity

Disadvantages of SOA Architecture

- Requires high availability
- Increased Overhead
- Single point of failure
- High Investment Cost
- Complex Service Management

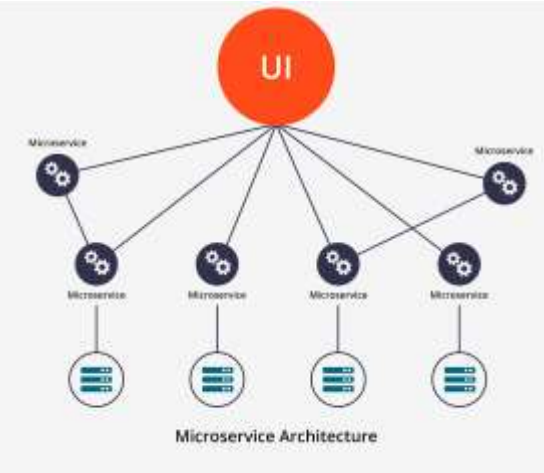
What is a MicroService?

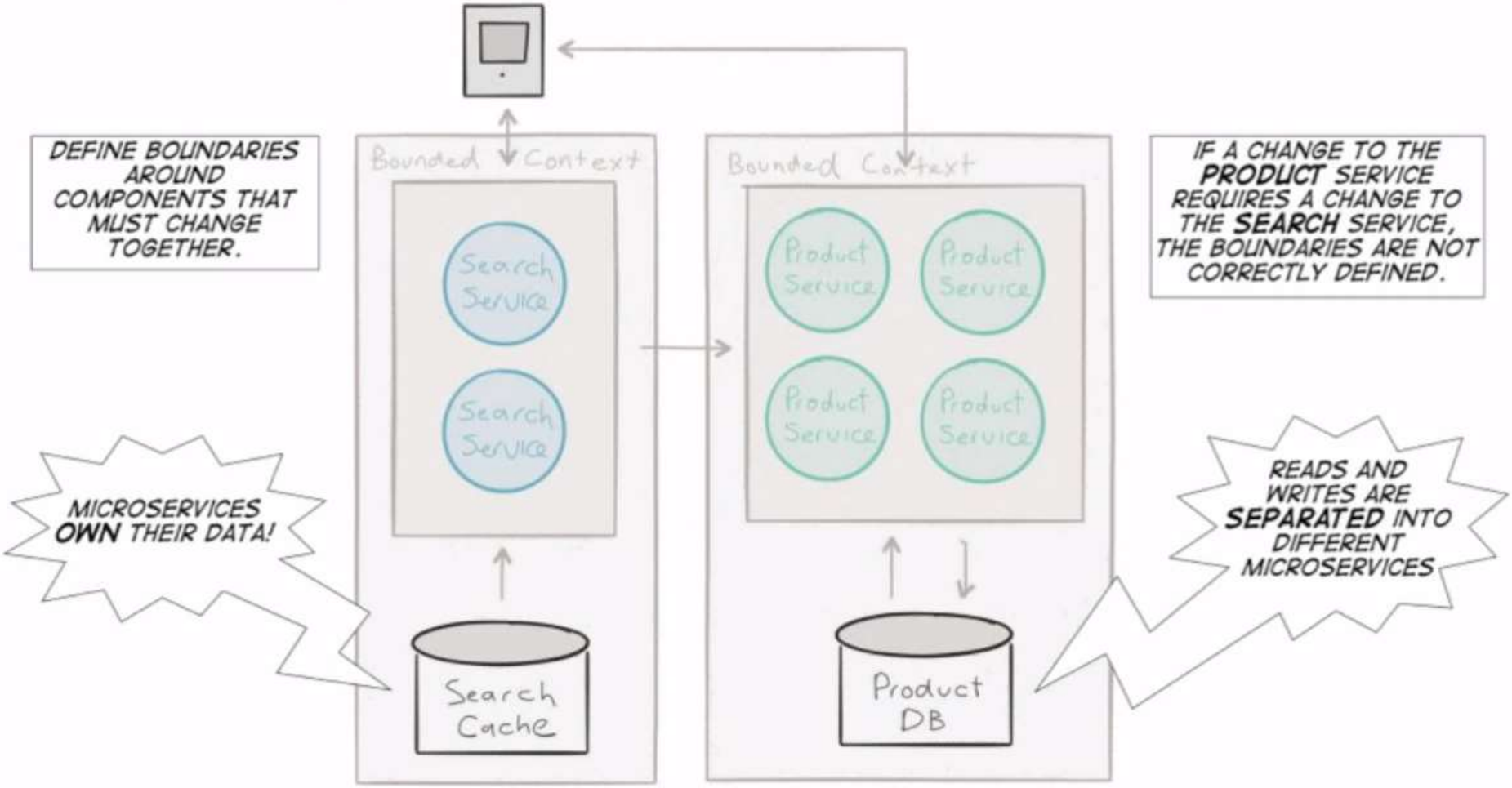
- A approach to developing a single application as a suite of small services which are fully independently deployable units.

- A microservice is an application with a single function, such as routing network traffic, making an online payment or analysing a medical result.

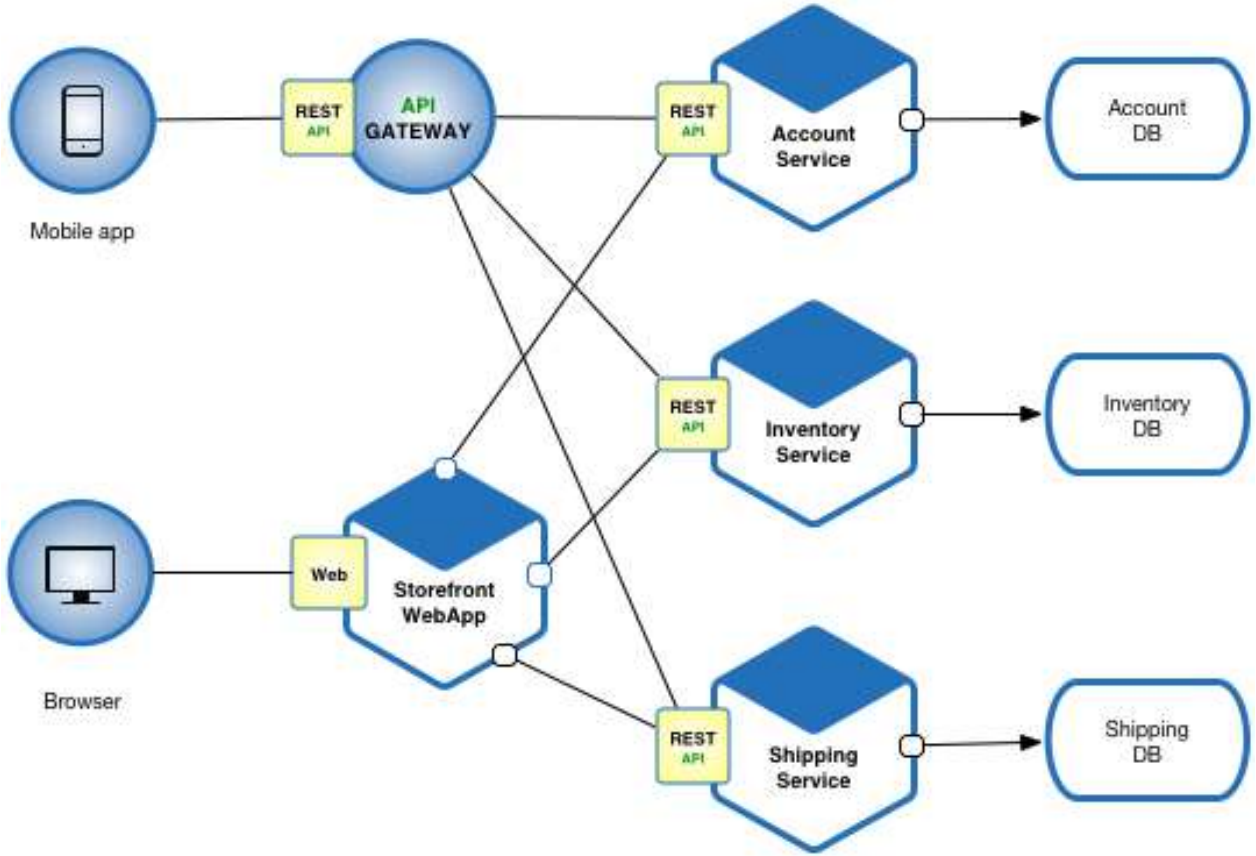
- Right now, the microservices architecture pattern is a rising star in the IT industry. It is the architectural evolution of SOA.

- Containers are a good way to develop and deploy microservices.

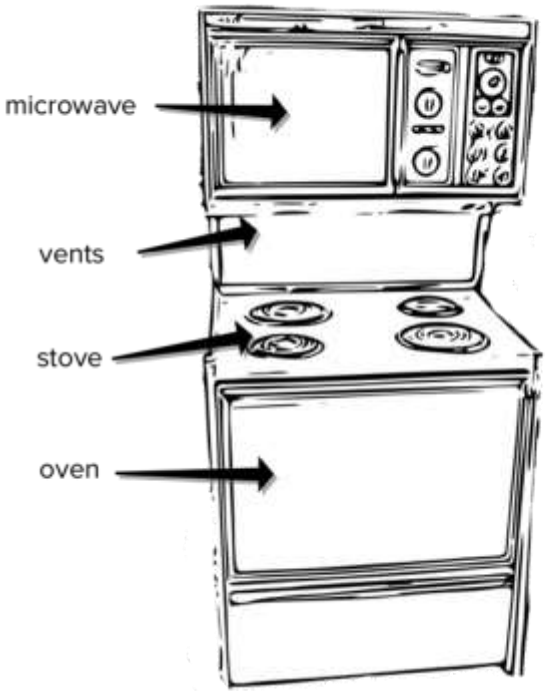



DEFINE BOUNDARIES AROUND COMPONENTS THAT MUST CHANGE TOGETHER.
Bounded Context
Search Service
Search Service
Search Cache
Bounded Context
Product Service
Product Service
Product Service
Product Service
Product DB
IF A CHANGE TO THE PRODUCT SERVICE REQUIRES A CHANGE TO THE SEARCH SERVICE, THE BOUNDARIES ARE NOT CORRECTLY DEFINED.
MICROSERVICES OWN THEIR DATA!
READS AND WRITES ARE SEPARATED INTO DIFFERENT MICROSERVICES

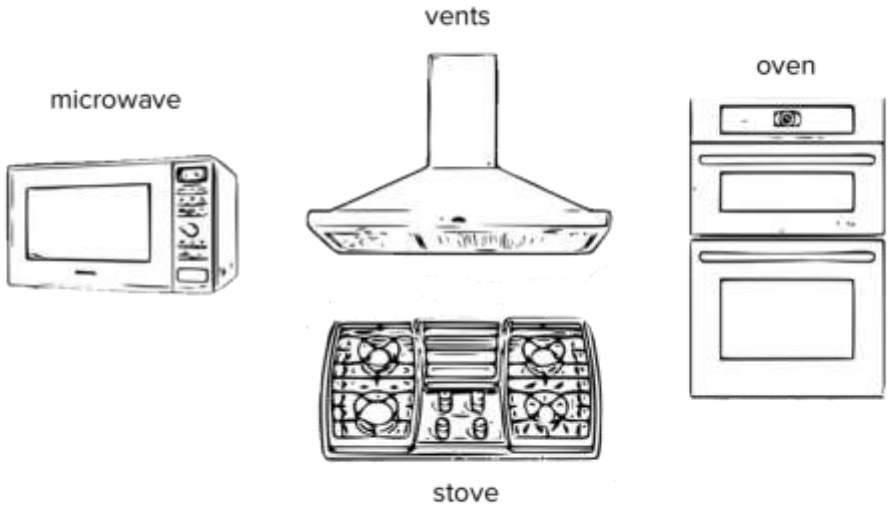
A e-commerce application example that takes orders from customers, verifies inventory and available credit, and ships them.

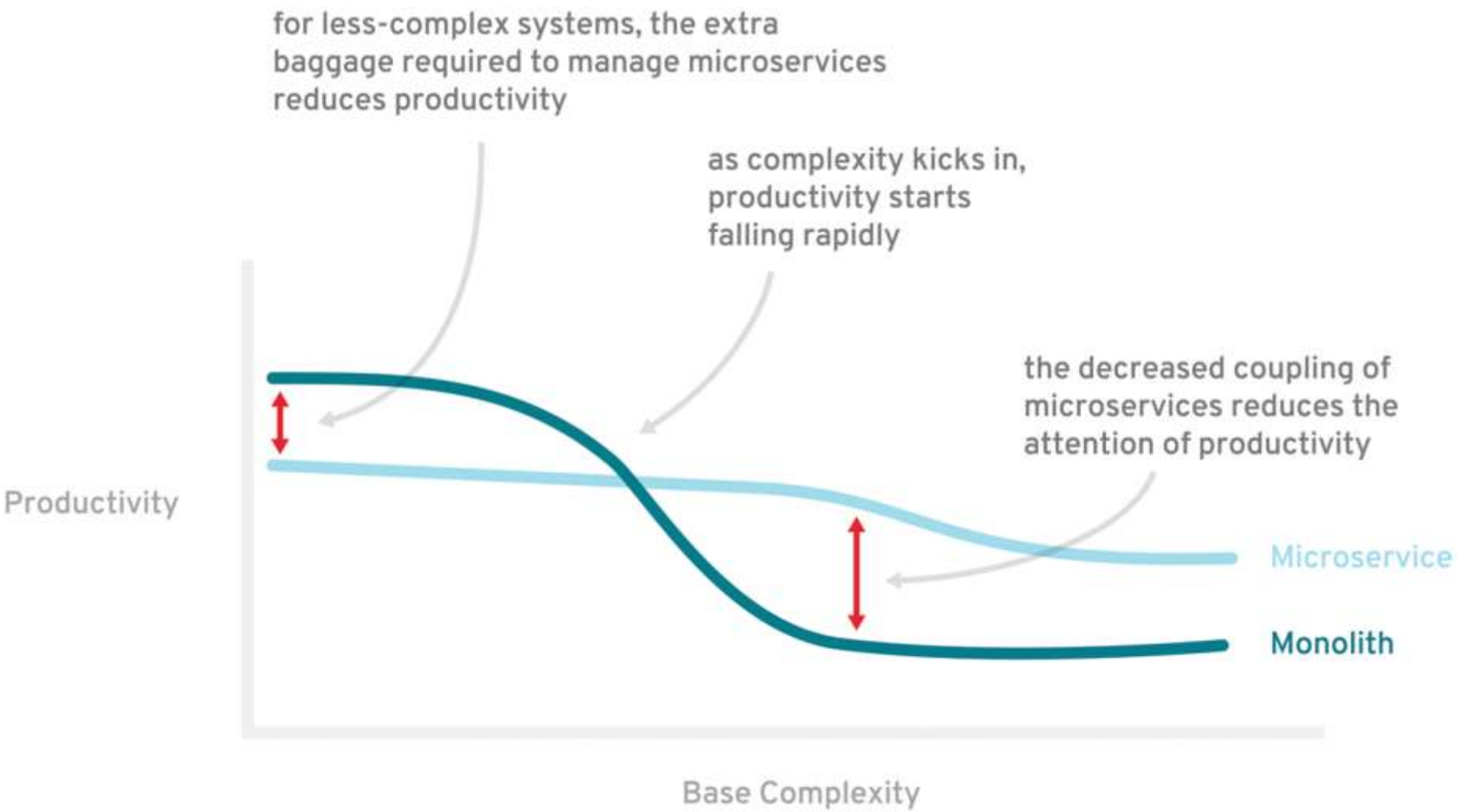


Monolithic Kitchen



Microservices Kitchen




for less-complex systems, the extra baggage required to manage microservices reduces productivity
as complexity kicks in, productivity starts falling rapidly
the decreased coupling of microservices reduces the attention of productivity
Productivity
Microservice
Monolith
Base Complexity

Characteristics of MicroServices

- Each microservice is loosely coupled, relatively small.
- Easier for a developer to understand.
- Each service has a bounded context means isolated from any other service.
- The IDE is faster making developers more productive.
- The web container starts faster, which makes developers more productive, and speeds up deployments.
- Each service can be developed and deployed independently.
- Eliminates any long-term commitment to a technology stack.

Summary

- To choose a best architecture, It is very important to understand the scope and requirements of your application.
- Never introduce the architectural level complexity unnecessarily.
- Both architectures(MSA and SOA) are focused on breaking up large monolithic applications into collections of smaller independent services, and both come with the promise of simplifying development.
- Always look for the scope of improvements into your application.

If you will try more… you will gain more…

Question please?

Good luck!

I Hope this will help you to increase your knowledge about architectural patterns for software design and You will definitely try to explore it more.

References

- http://odino.org/on-monoliths-service-oriented-architectures-and-microservices/
- https://thetechsolo.wordpress.com/2015/07/05/from-monolith-three-tiers-architectures-to-soa-vs-microservices/
- http://microservices.io/patterns/microservices.html
- https://blog.codeship.com/monolithic-core-vs-fully-microservice-architecture/
- http://www.slideshare.net/altoros/microservices-vs-monolithic-architectures-pros-and-cons
- http://martinfowler.com/microservices/