ARPIT JAIN | VIPIN DHONKARIA

What are Microservices?

You must be thinking why Rocks and Pebbles?

Lets start with **Monolithic Architecture**

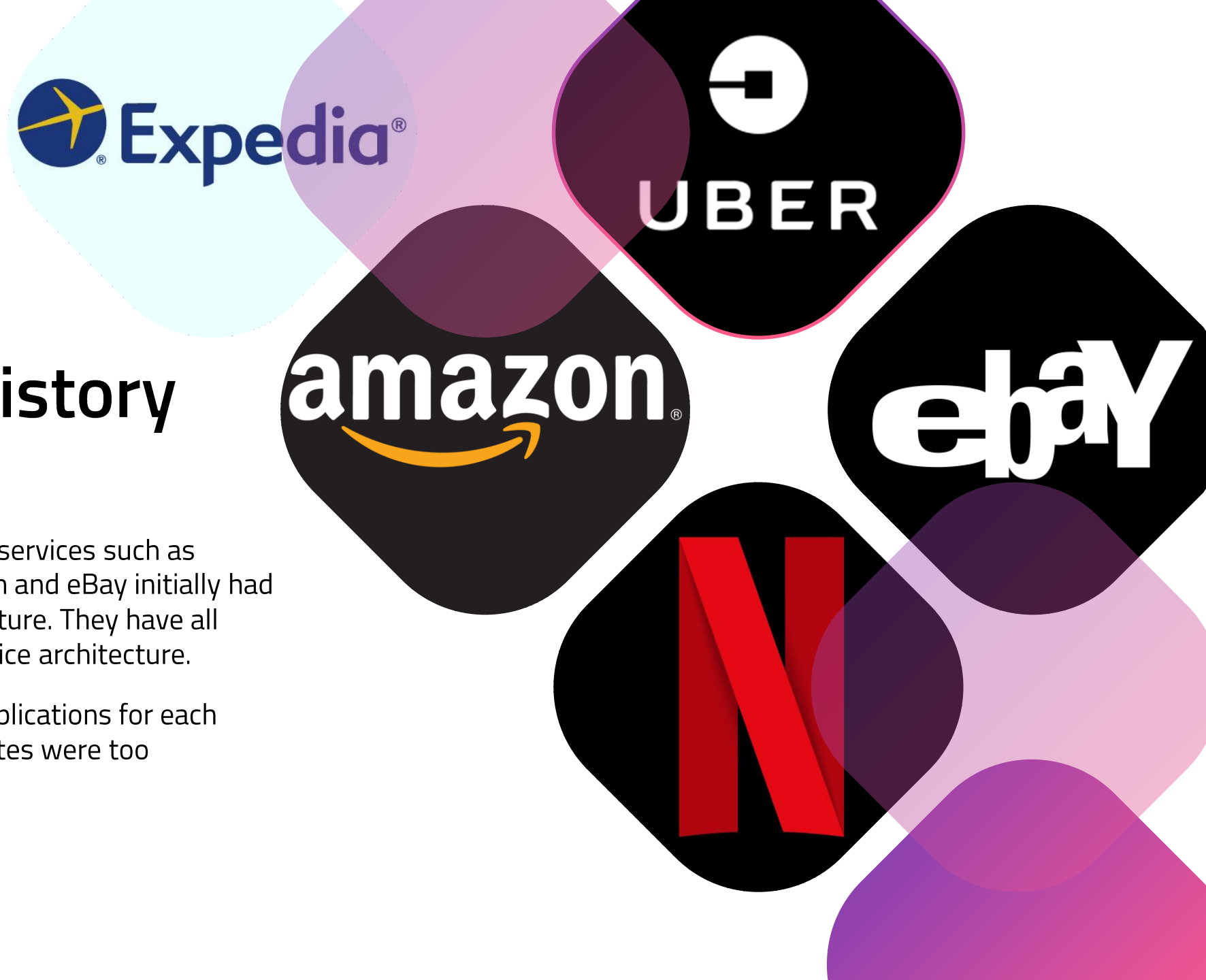
Monolithic architecture is the traditional unified model for the design of a software program.

Monolithic software is designed to be self-contained.

Components of the program are inter-connected and inter-dependent rather than loosely coupled.

In a tightly-coupled or non-modular architecture, each component and its associated components must be present in order for code to be executed or compiled.

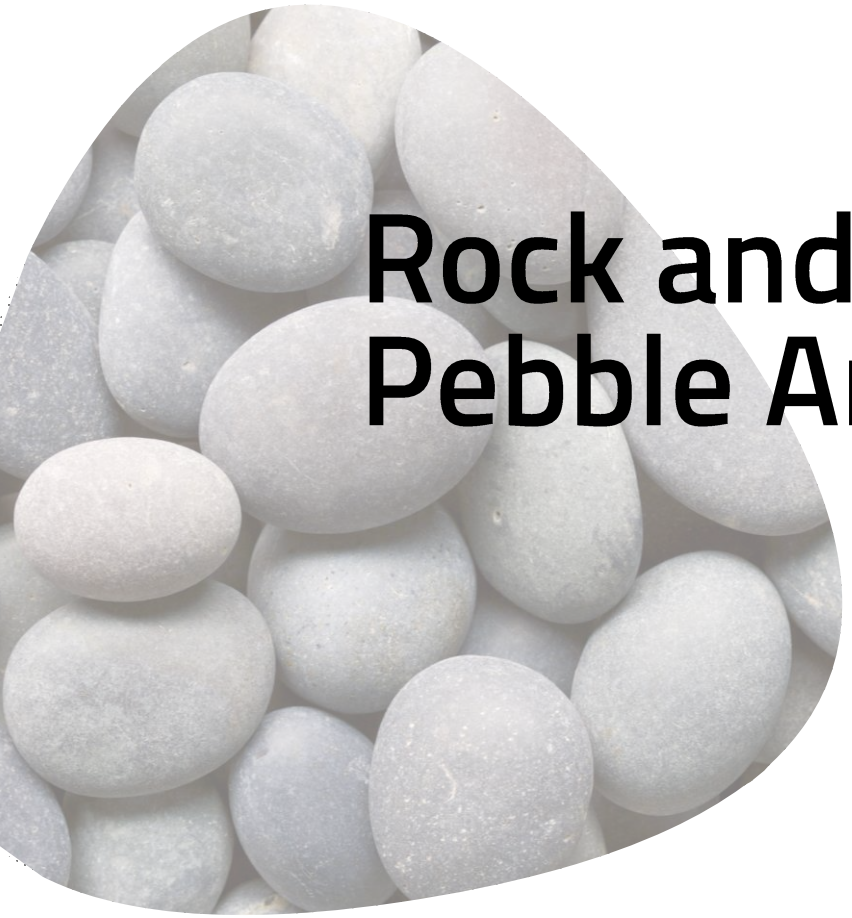
Monolithic Application is like a house. Once built, it can't be moved!



A Little **History**

Some Facts

- Well known internet services such as Netflix, Uber, Amazon and eBay initially had a monolithic architecture. They have all evolved to Microservice architecture.
- Back in 2005, the applications for each of Expedia's travel sites were too difficult to modify.



Rock and Pebble Analogy

Pebble

The starting phase of any application.

Time and Effort

(Size of Team and Application Increases)

Microservice Territory

1. This is where the decision is made to convert the application to Microservice Architecture.
2. If application doesn't have the potential to grow larger, then monolithic architecture is sufficient.

Monolith

A large, heavy unit.

Service Oriented Architecture **(SOA)**

A Service-oriented architecture is essentially a collection of services.

These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity.

Some means of connecting services to each other is needed.

What are **Microservices?**

Microservices accelerate delivery by minimizing communication and coordination between people while reducing the scope of change.

Microservices are

An Engineering Approach
Focused on Decomposing applications
Into single function modules
And well defined interfaces
Which are independently deployed
And operated by small teams
Who own the entire lifecycle of service.

Microservices can be developed in any language and they communicate with each other using Language Neutral APIs Like REST.

Microservices Characteristics

- Small and Focused.
- Loosely Coupled.
- Bounded Context.
- Autonomous (Independent)
- Developed on its own timetable
- Technology heterogeneity (supports many languages and platforms)
- Manages its own data
- Easy deployment
- Scales and fails independently

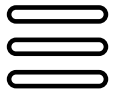
Now, What is Microservice architecture?

Microservice Architecture is a special design pattern of Service-oriented Architecture(SOA). In this type of service architecture, all the processes will communicate with each other with the smallest granularity to implement a big system or service.

It deals with system of services that communicate over a network.

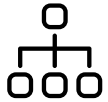
SOA vs Microservices

CAB Booking System (Software Units)



Service Oriented Architecture

1. GetPayments and DriverInformation and MappingDataAPI
2. AuthenticateUsers and DriversAPI



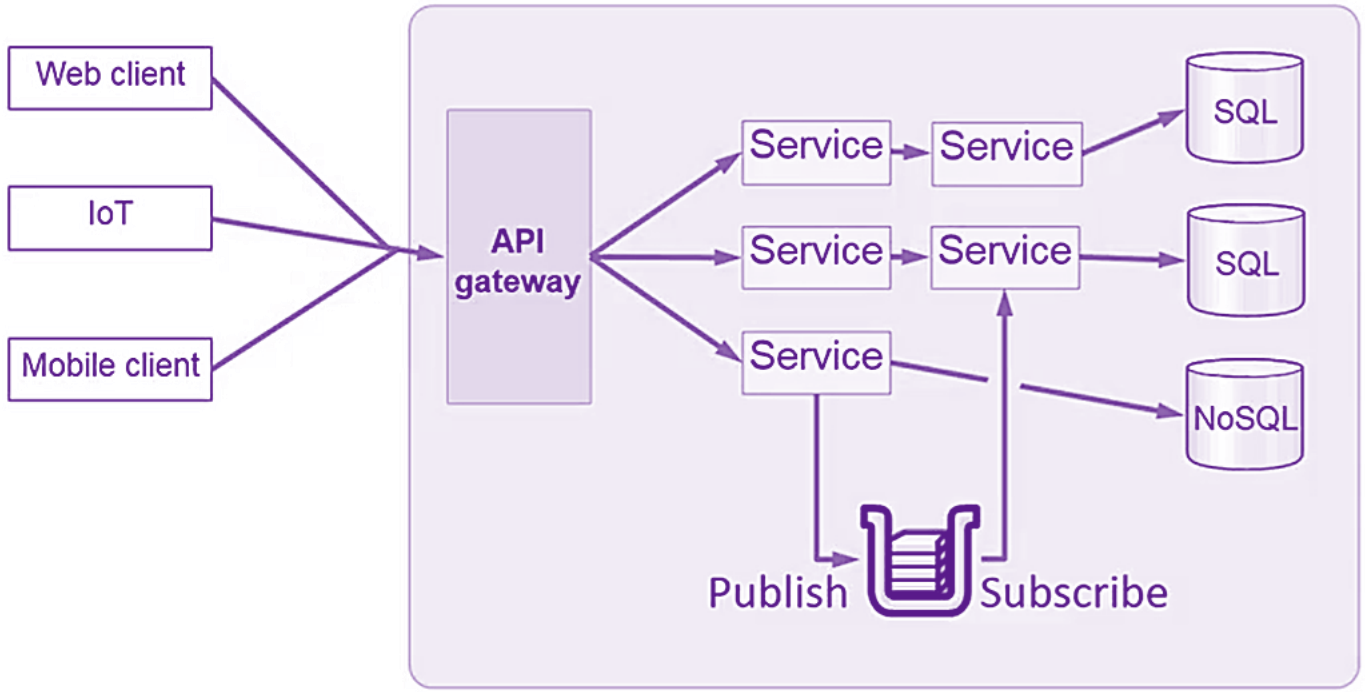
Microservice Architecture

1. SubmitPaymentService
2. GetDriverInfoService
3. GetMappingDataService
4. AuthenticateUserService
5. AuthenticateDriverService

API Gateway

The API gateway sits between the clients and the microservices, and provides APIs that are made for the clients.

It provides a simplified interface to the clients, making the service easier to use, understand and test.



Size of Microservices?

There is no strict rule regarding the optimal size, but some practices are there. Several techniques that can be used alone or in combination:

- Too many responsibilities: A service that serve too many functions needs to been broken up.
- Service Type: A microservice should do one thing, for example-
  - Handle Authentication
  - UI
  - Serve several web pages

- **Bounded context separation:** It represents parts of the system that are self-sufficient.

Microservice Architecture Example

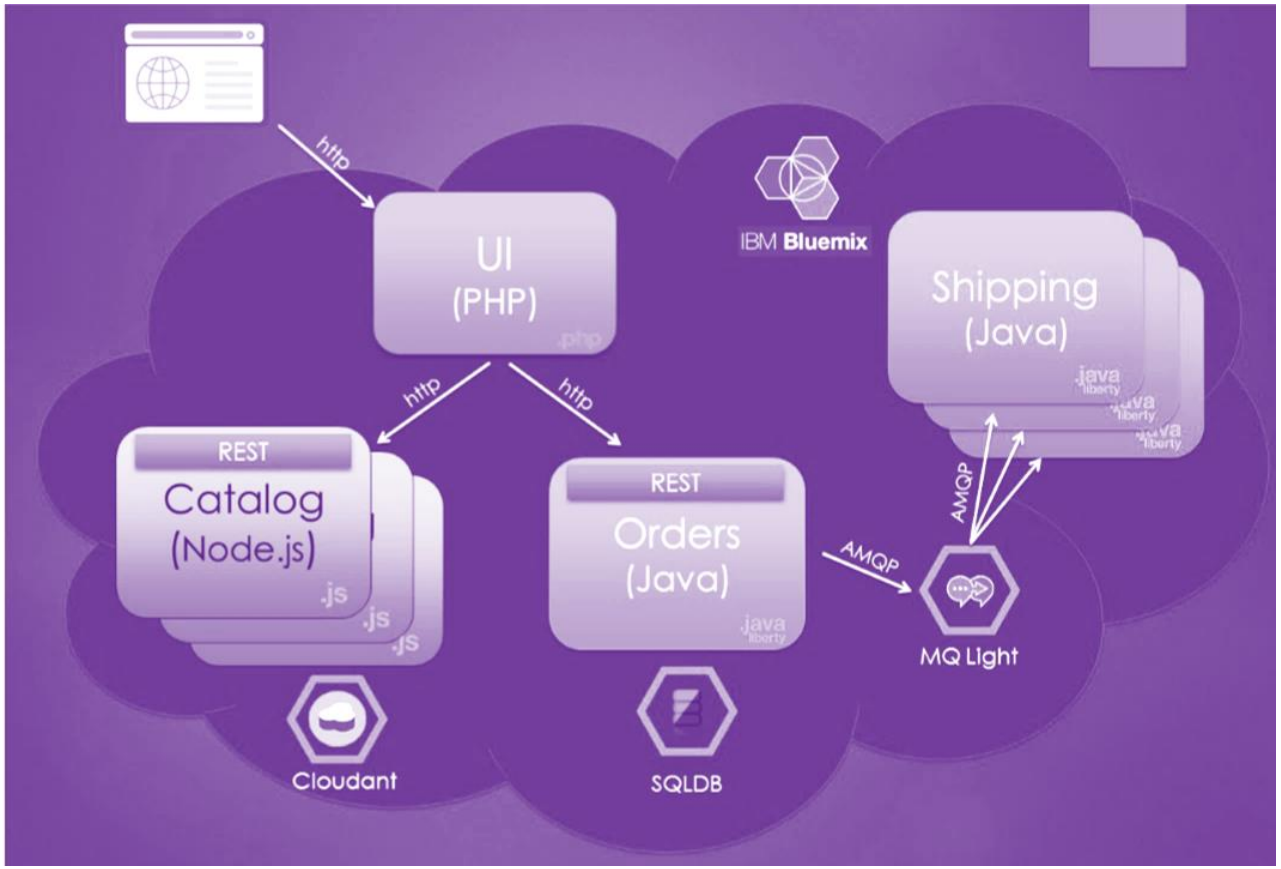
An online store that serves different functions:

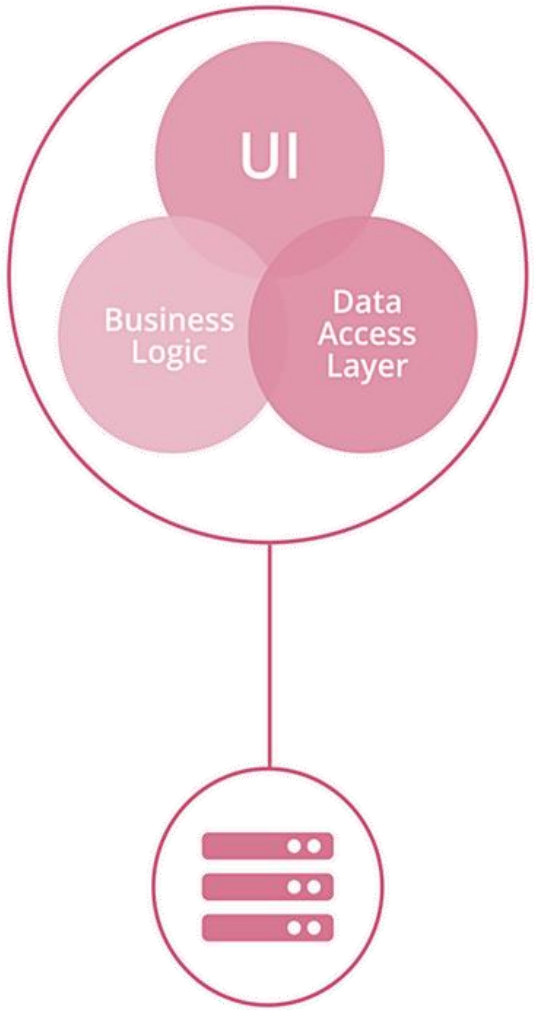
Catalog: Responsible for keeping track of all the items for sale in the store.

Orders: Process new customer orders, and provides details of past orders.

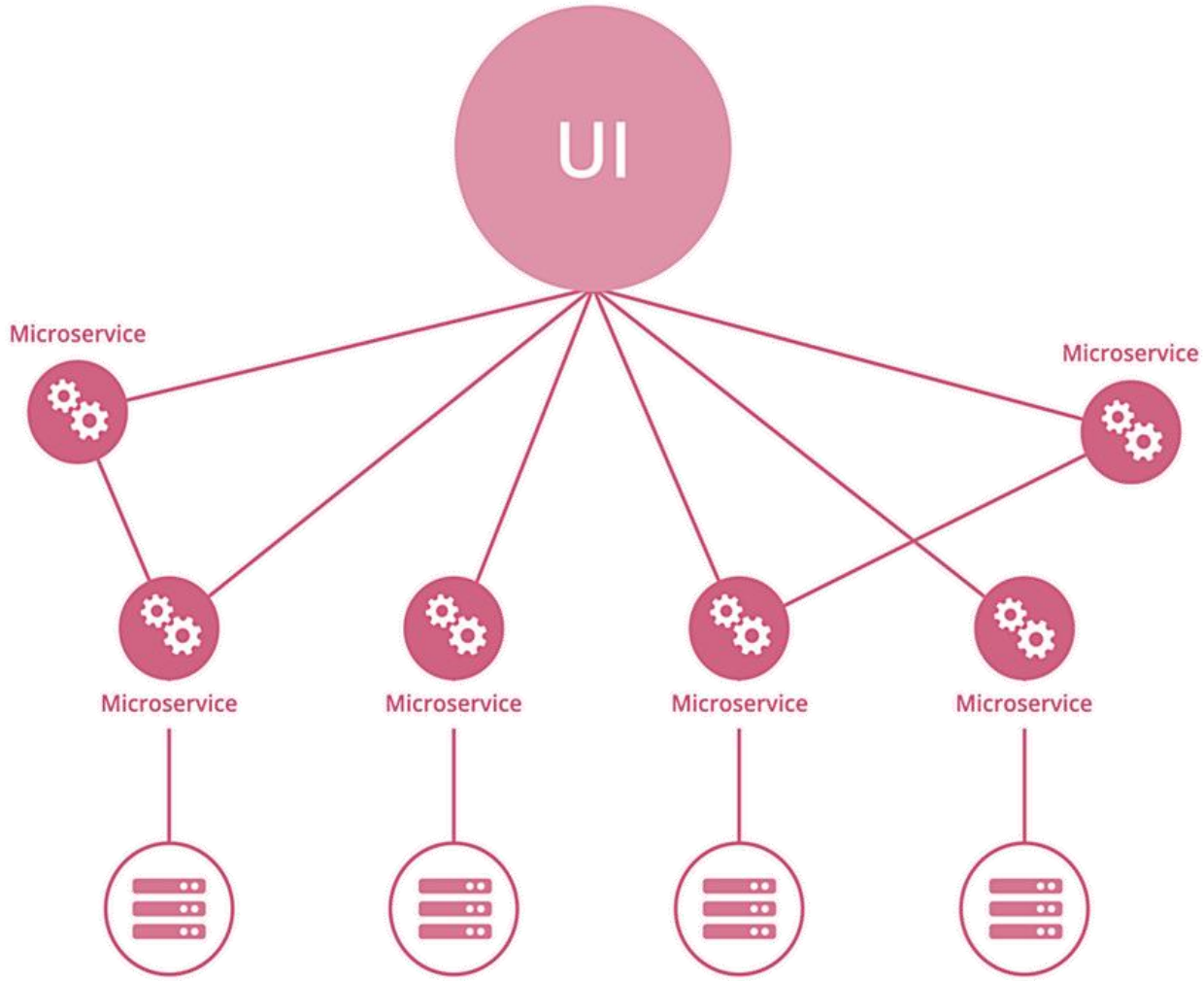
Shipping: After order is made, responsible for shipping related processes.

UI: A web front end store, which enables shoppers to browse the catalog and place the order.

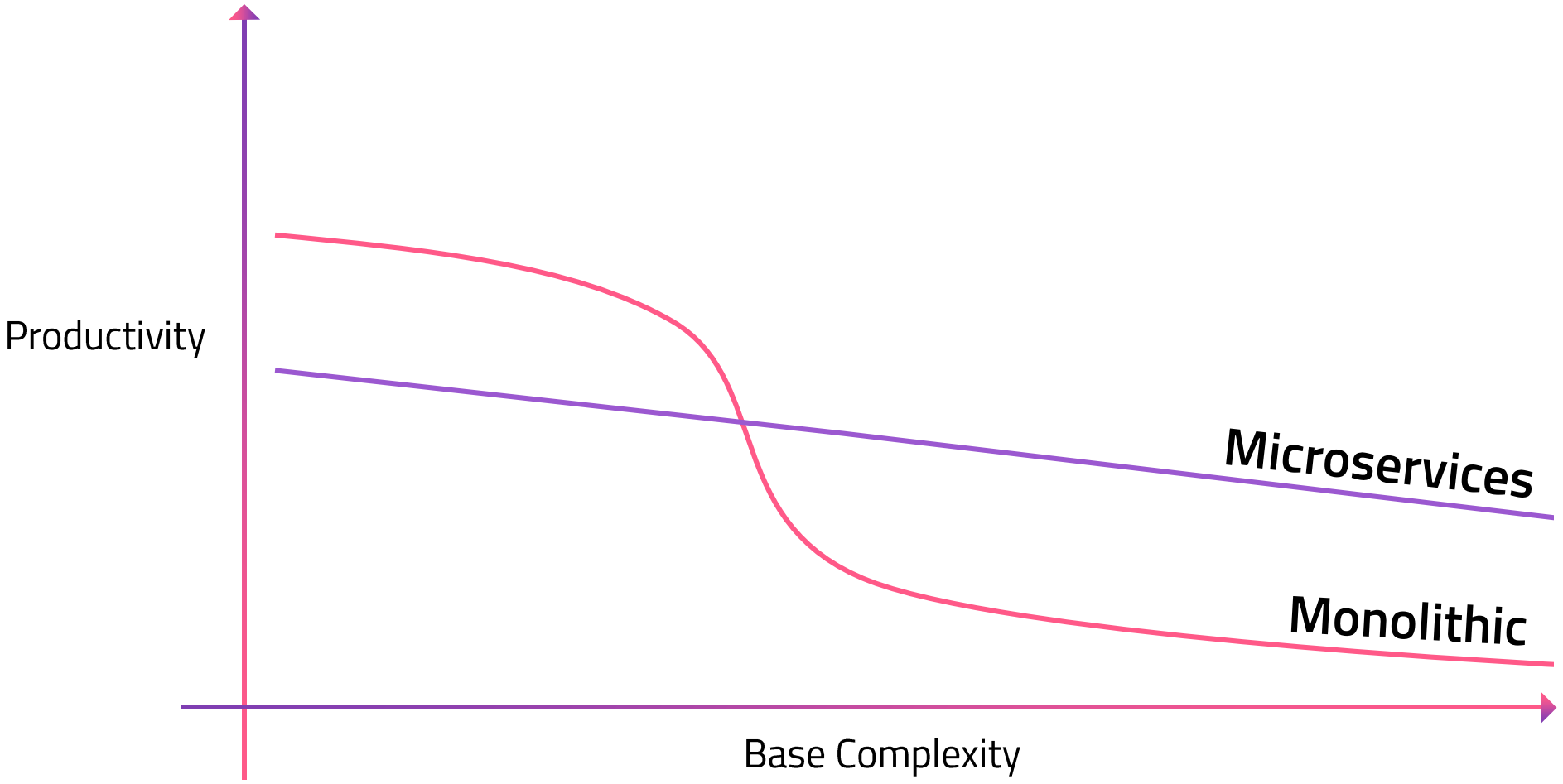




Monolithic Architecture



Microservice Architecture


Productivity
Microservices
Monolithic
Base Complexity

Three major disadvantages of **Monolithic Arch.**

This is why Microservice architecture gained stature for large applications.

SCALABILITY

To accommodate more customers, the instances of the same application need to be created, but the resources utilized in other services are wasted, since they don't need have a need to scale. This leads to decrease in agility of application.

FAULT TOLERANCE

When one component fails, the entire system goes down.

BOTTLENECKS

Part of the Software that takes larger time, slows down the entire software. No Independence.

WHERE IS THE PROBLEM?

Disadvantages of **Microservice Architecture**

01

DISTRIBUTED SYSTEM

Distributed System is complex to design.

02

COST

Cost of designing and building a distributed system is high.

03

ENTERPRISE READINESS

Converting from monolithic to Microservice Architecture is extremely tough.

When and When not to use Microservice Architecture

When to

NEED FREQUENT SCALING

When any large dynamic application needs frequent scaling.

MAINTAIN INDEPENDENCE

When each module of the application needs to be handled separately. This ensures fault tolerance.

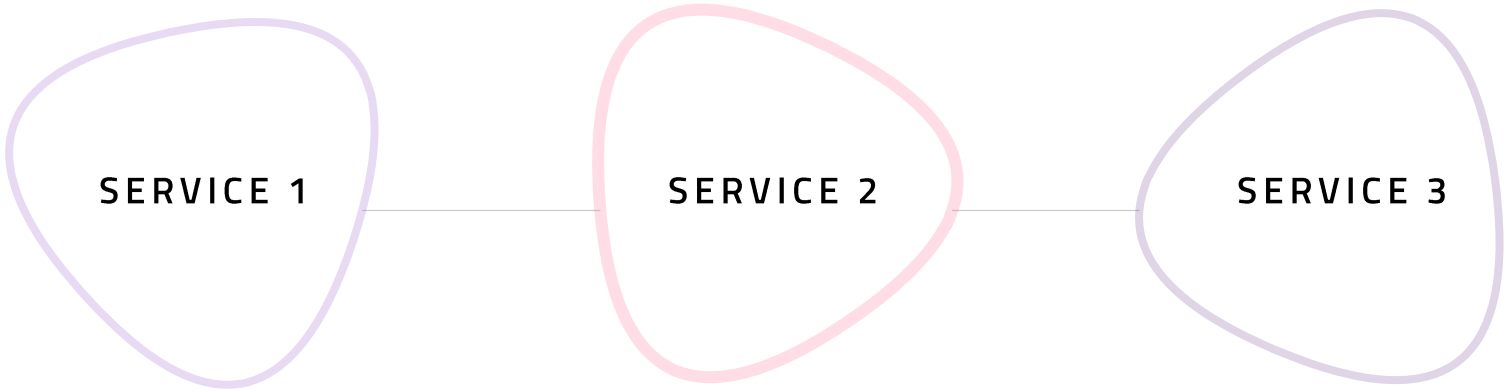
When not to

NO NEED OF EXPANSION

When its clear that the application wont expand further than it currently is.

TECHNOLOGY STACK IS FIXED NO NEED OF FREQUENT DEPLOYMENTS

- When the entire application is built on the same Tech. Stack and it wont change in the near future.
- When there is a constant need to deploy new features frequently.

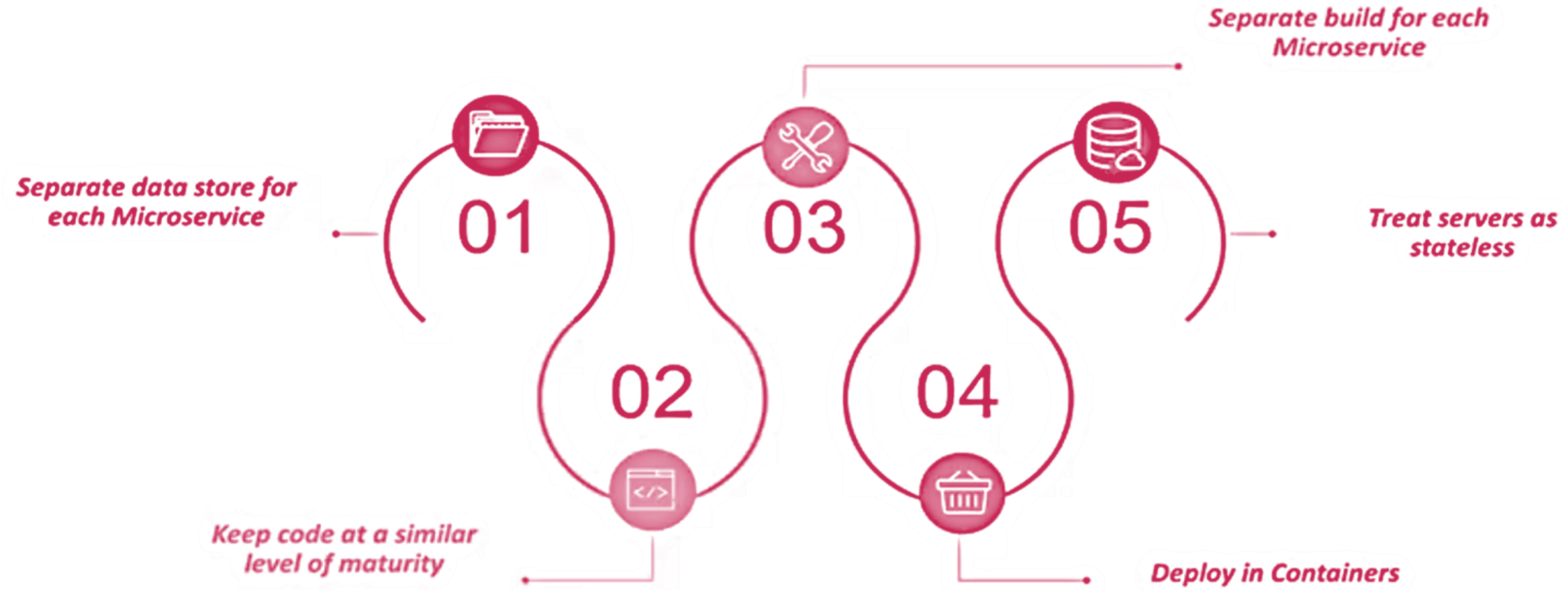


Microservice Communication

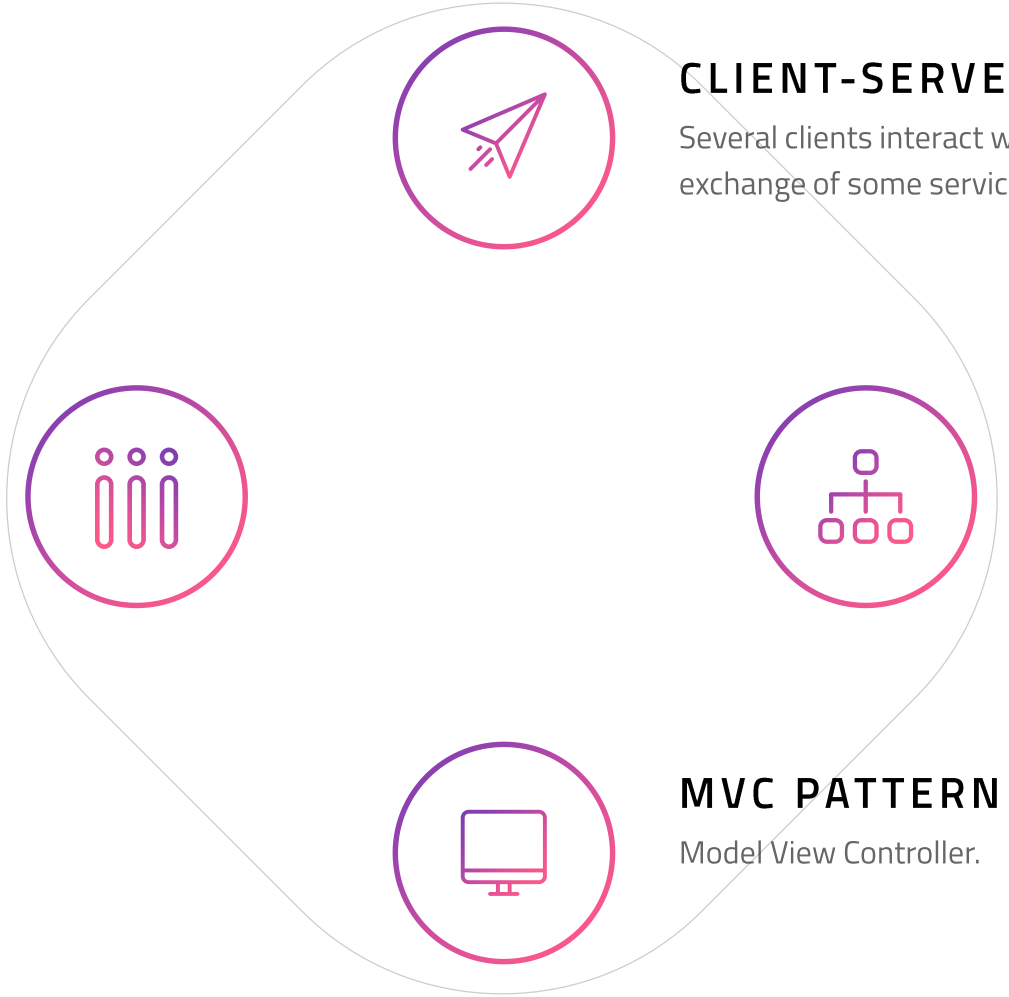
- Lightweight protocols are used such as **JSON or HTTP** for **REST** and **Kafka** for Messaging.
- The goal is to Decouple the application into self functioning modules (microservices). In order to Achieve that, Messaging is established whenever required.

Note : Communication is done in a single language (language neutral) but services can be in different languages.

Microservice Architecture
Best Practices



Other Architectures for **Software Development**



CLIENT-SERVER

Several clients interact with the server in exchange of some service offered by server.

PEER TO PEER

Several services communicate with one-another at the same level.

MASTER SLAVE

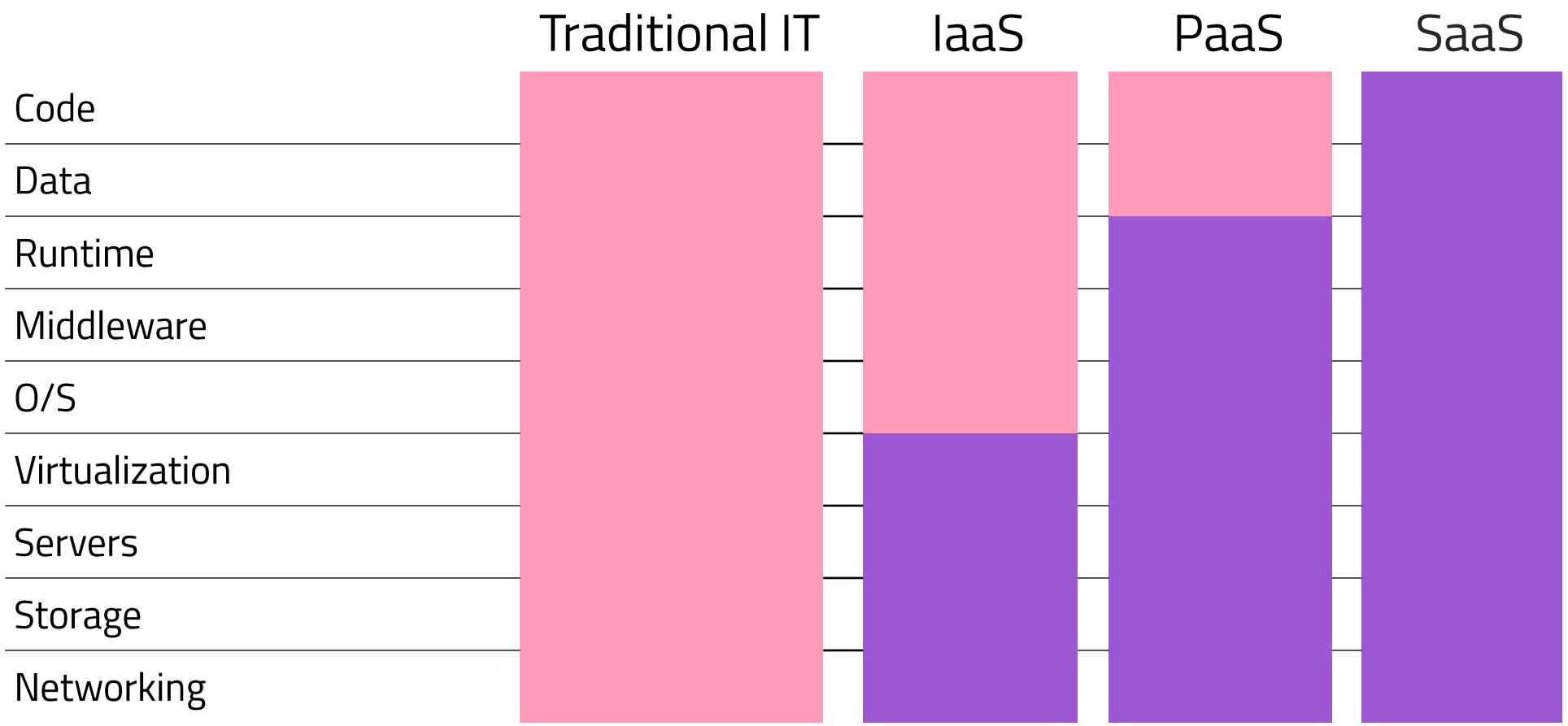
Several sub-services governed by one master controlling service.

MVC PATTERN

Model View Controller.

SERVERLESS ARCHITECTURE

Cloud Solutions



Total Management by Vendor in Cloud

Traditional IT	IaaS	PaaS	SaaS
0 %	55.5 %	77.7 %	98 %

Serverless Architecture

Serverless architectures are application designs that **incorporate third-party "Backend as a Service" (BaaS)** services, and/or that include custom code run in managed, ephemeral containers on a **"Functions as a Service" (FaaS) platform.**

In simple words, Applications that significantly or fully incorporate third-party, cloud-hosted applications and services, to manage server-side logic and state.

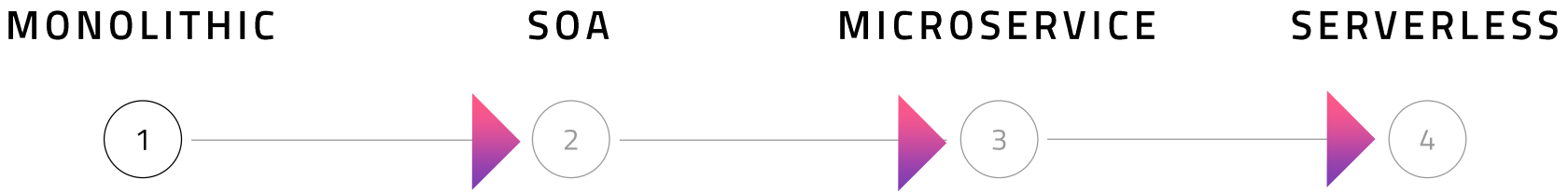
Advantages

- Deployment Simplicity (Reduced time to market)
- Scalability
- Efficiency
- Lower Operations Cost (Due to resource pooling in Cloud)

Dis-advantages

- Increased Complexity due to more flexibility
- Extensive Tooling

What is the **Situation** now?



THANKYOU