

# Microservices vs Monoliths

## Architectural Differences



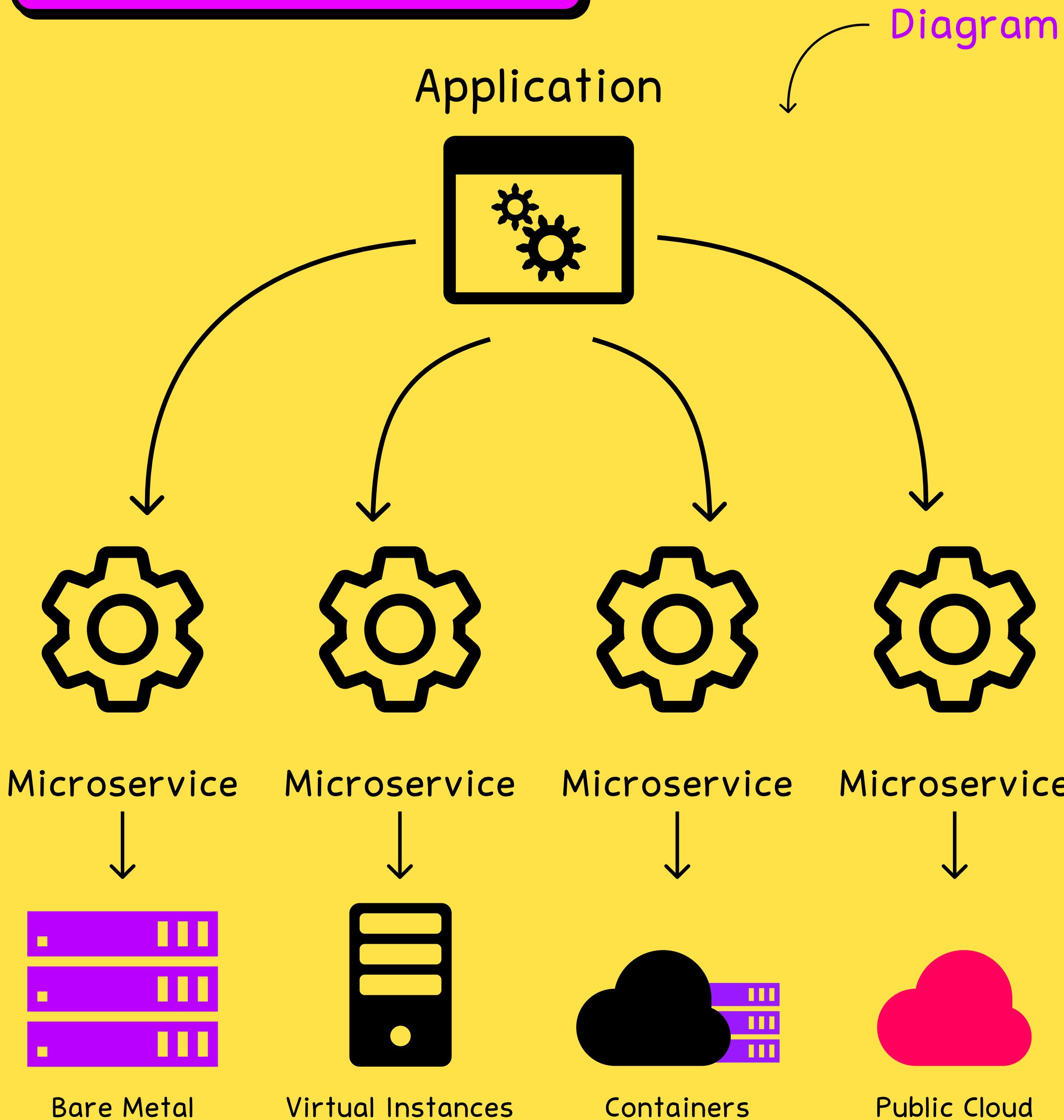
# Microservices

Microservices is a software architecture style in which a large application is built as a suite of small, independent and deployable services.

Each service is designed to perform a specific task and communicates with other services through well-defined protocols such as HTTP, gRPC, AMQP (Advanced Message Queuing Protocol) and Pub/sub.

With Microservices, each service can be deployed independently and scaled up or down as needed, which can reduce costs.

# Microservices



# Microservices

## Pros

### Better Organization

Broken down into **smaller services** and is relatively better organized.

### Increased Agility

Individuals of a team can build and deploy modules **independently**.

## Debugging and Testing Challenges

Microservices can be **challenging to debug and test** because they have more components and interactions.

## Cons

### Complexity

The **increased number** of moving parts and components that need to be maintained and managed makes microservices more complex.

# Monolithic

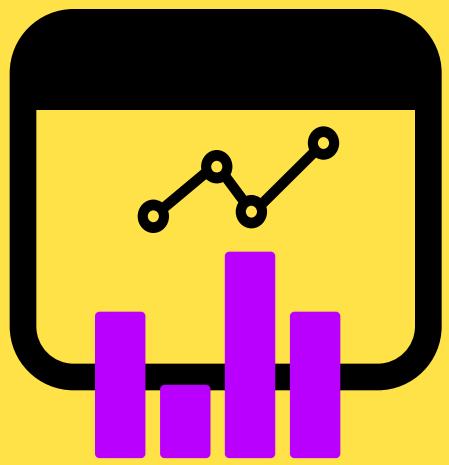
Conventional method of application development which is developed as a single package.

A monolith consists of a three-tier architecture, namely, a database, a user interface, and a server-side application.

Modules are tightly coupled with each other. The application and the business logic is encapsulated in a single deployable binary called a monolith.

# Monolithic

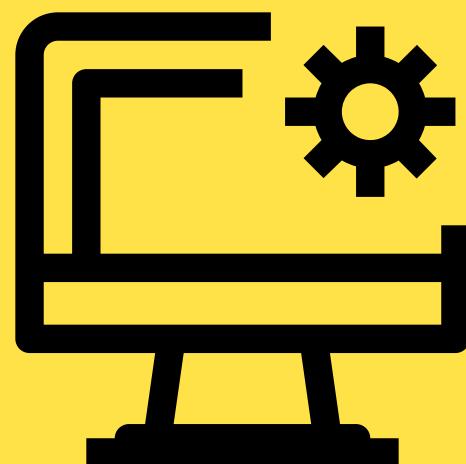
Business Logic



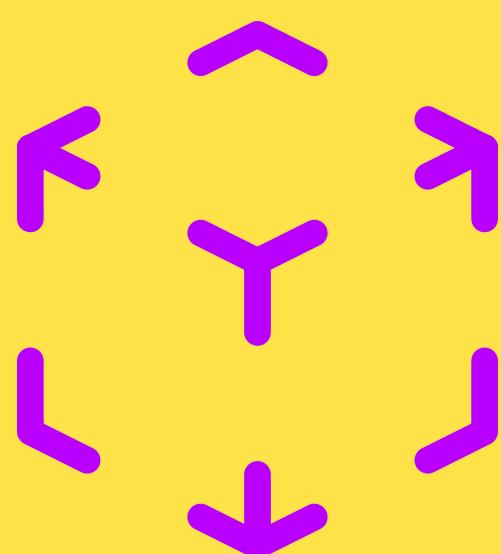
Data Layer



User Interface



Monolith Application



Diagram

swipe >

# Monolithic

## Pros

Simplicity of Debugging and Testing

The debugging and testing process is **simple** because all code is located in one place.

## Easy Development

Developers just perform **single chunk** of deployable code.

## Slow Build and Release

Code base is **enormous**. This **retards** the velocity of the development cycle.

## Cons

## Tight Coupling

Business logic is **tightly entangled**. Makes it difficult to isolate the application.

# Differences

These are the key differences between **Microservice Architecture** and **Monolithic Architecture**.

- Language
- Codebase
- Understandability
- Application Scaling
- Development and Deployment
- Service Startup
- Data Model
- Consistency and Availability

Let us now look these differences in detail

# Language

## Monolithic

## Microservice

Every service can be developed **separately** in a different programming language.

Completely developed using a **single** programming language.

## Monolithic

It has only a **single codebase**. Components are interdependent.

# Codebase

## Microservice

Every service has a **separate codebase** for them.

# Understandability

Monolithic

Microservice

Has high understandability and is very easy to maintain.

It is very difficult to understand, and it is confusing.

Monolithic

# Application Scaling

Application scaling is very difficult as the entire application should be scaled.

Microservice

Very easy as each service can be scaled separately without scaling of the entire application.

# Development and Deployment

Monolithic

Continuous development and deployment are very complicated.

Continuous development and deployment are possible.

Microservice

Monolithic

Time-taking service startup.

# Service Startup

Microservice

Quick service startup.

## Data Model

Monolithic

It has a centralized data model.

## Microservice

It has a federated data model, allowing each service to adopt its own data model.

## Monolithic

Comparatively less consistent and available

## Consistency and Availability

## Microservice

Highly consistent and readily available.

# Conclusion

Monolithic

Monolithic architecture is preferred for small teams, simple applications, and shorter deadlines.

Microservice

Microservice architecture is good for the development of complex applications. It is known for its change adaptability and scalability.