Spring Boot

TECHNOLOGY

Microservices

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

Learning Objectives

By the end of this lesson, you will be able to:

- Examine the need for microservices and the concept of microservices architecture
- Identify the features of microservices architecture
- Differentiate between microservices and API
- Differentiate between MSA and SOA

Learning Objectives

By the end of this lesson, you will be able to:

- Analyze the monolithic architecture
- Analyze the architectural frameworks
- Classify the concept of orchestration and differentiate it from automation
- Identify the best practices for microservices security

A Day in the Life of a Full Stack Developer

You are hired in an organization as a senior developer. You have been asked to develop an enterprise application for the organization. The organization wants a self-contained service to run applications and maintain them.

You decide to make use of microservices, which is a collection of small services designed to be self-contained and run applications.

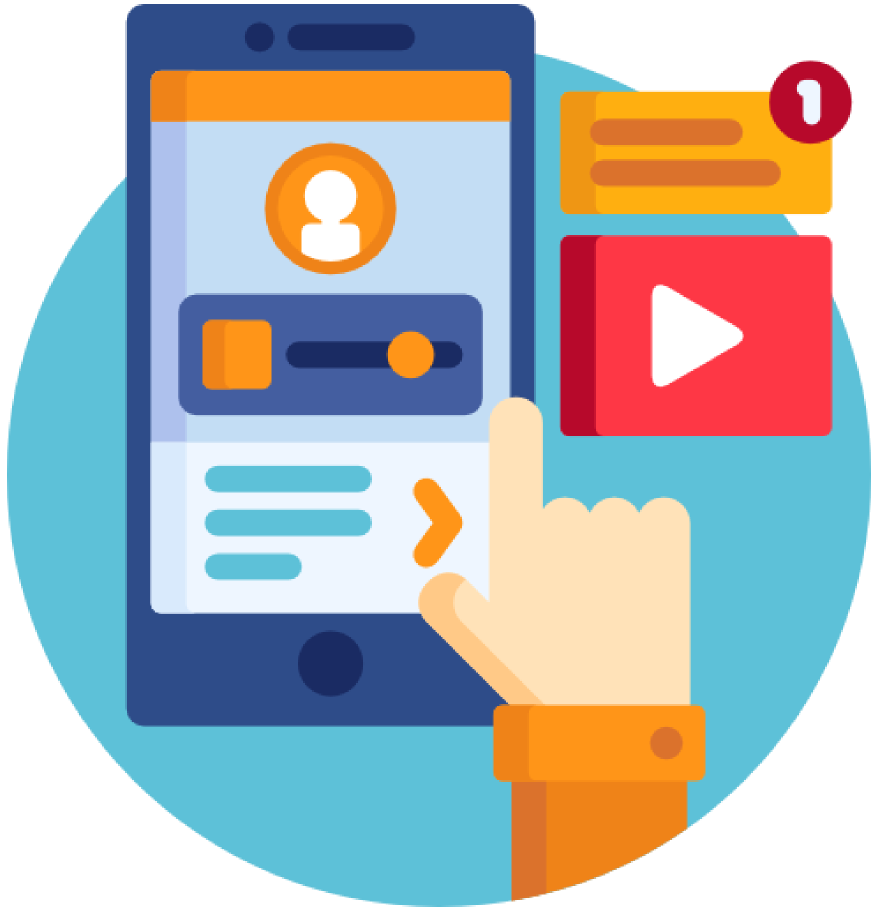
To do so, you must explore more about microservices, their features, monolithic architecture, and best practices.

TECHNOLOGY

Microservices: Overview

Microservices: Overview

Microservices is a collection of small services designed to be self-contained and run applications.



Microservices: Overview

In monolithic architecture, the application's various components are combined to create a large application.

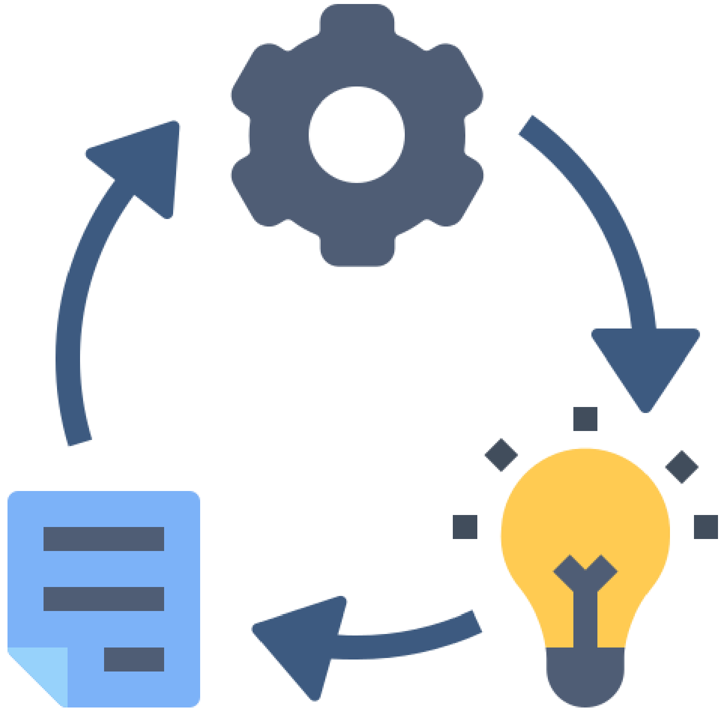
If the monolithic application is down, the entire application will be down.

Microservices: Overview

Microservices break down an application into smaller sections. It helps to understand where a problem occurs.

Microservices: Overview

Microservices help design simple applications and maintain them.



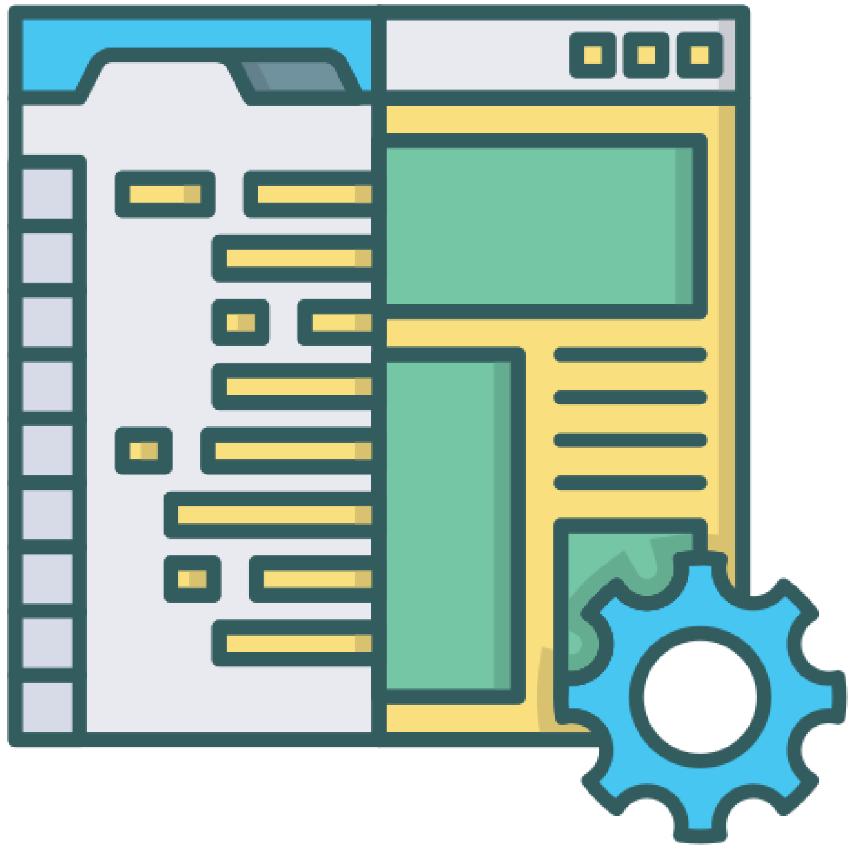
Microservices: Overview

Modules communicate with each other through simple, universally accessible APIs.

Monolithic Architecture

Monolithic Architecture

It describes a single-tiered software application where different components are combined into a single program.



Components of Monolithic Architecture

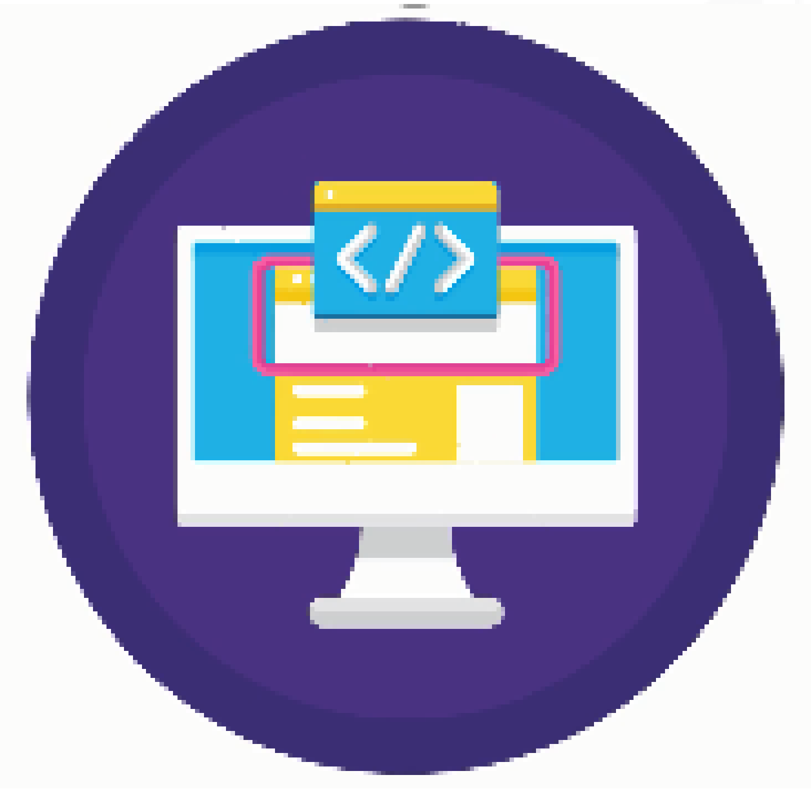
- Authorization
- Presentation
- Business logic
- Database layer
- Application integration
- Notification module

Is responsible for authorizing a user

Components of Monolithic Architecture

- Authorization
- Presentation
- Business logic
- Database layer
- Application integration
- Notification module

Is responsible for handling the HTTP requests and responding with either HTML or JSON/XML



Components of Monolithic Architecture

- Authorization
- Presentation
- **Business logic**
- Database layer
- Application integration
- Notification module

Provides the application’s business logic

Components of Monolithic Architecture

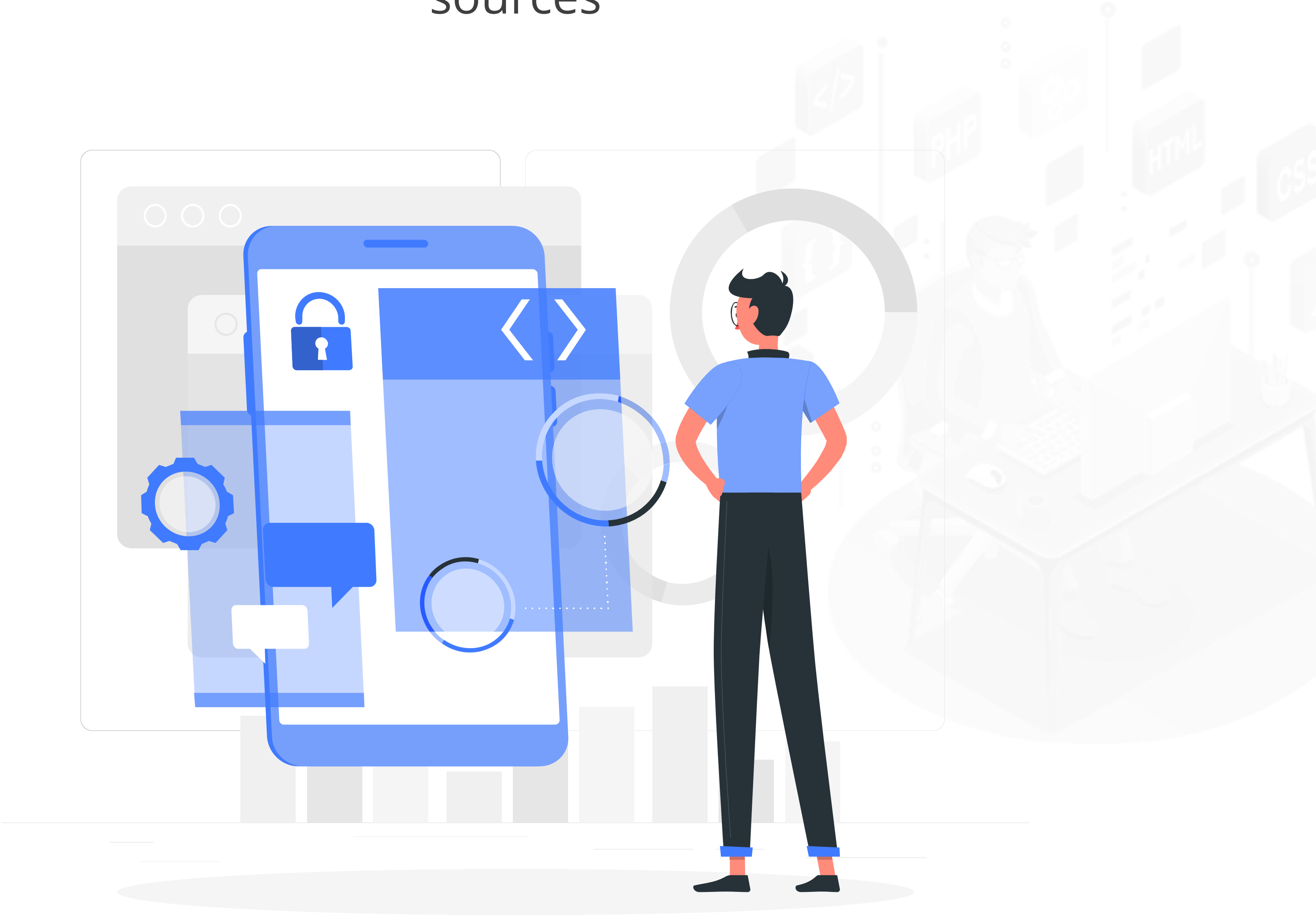
- Authorization
- Presentation
- Business logic
- Database layer
- Application integration
- Notification module

Data access objects are responsible for accessing the database.

Components of Monolithic Architecture

- Authorization
- Presentation
- Business logic
- Database layer
- Application integration
- Notification module

Provides integration with the other services or data sources



Components of Monolithic Architecture

- Authorization
- Presentation
- Business logic
- Database layer
- Application integration
- **Notification module**

Is responsible for sending email notifications

Monolithic Architecture: Advantages

It is simple to test.

It is horizontally scalable by running multiple copies.

It is easy to develop and work with.

It is easy to deploy.

Monolithic Architecture: Limitations

Monolithic applications can be challenging to scale when the different modules have conflicting resource requirements.

If the application is too large and complex to understand, making the changes quickly and correctly is challenging.

TECHNOLOGY

Microservices Architecture

Microservices Architecture

The architectural framework encapsulates the minimum set of practices and requirements for the artifacts.

Microservices Architecture

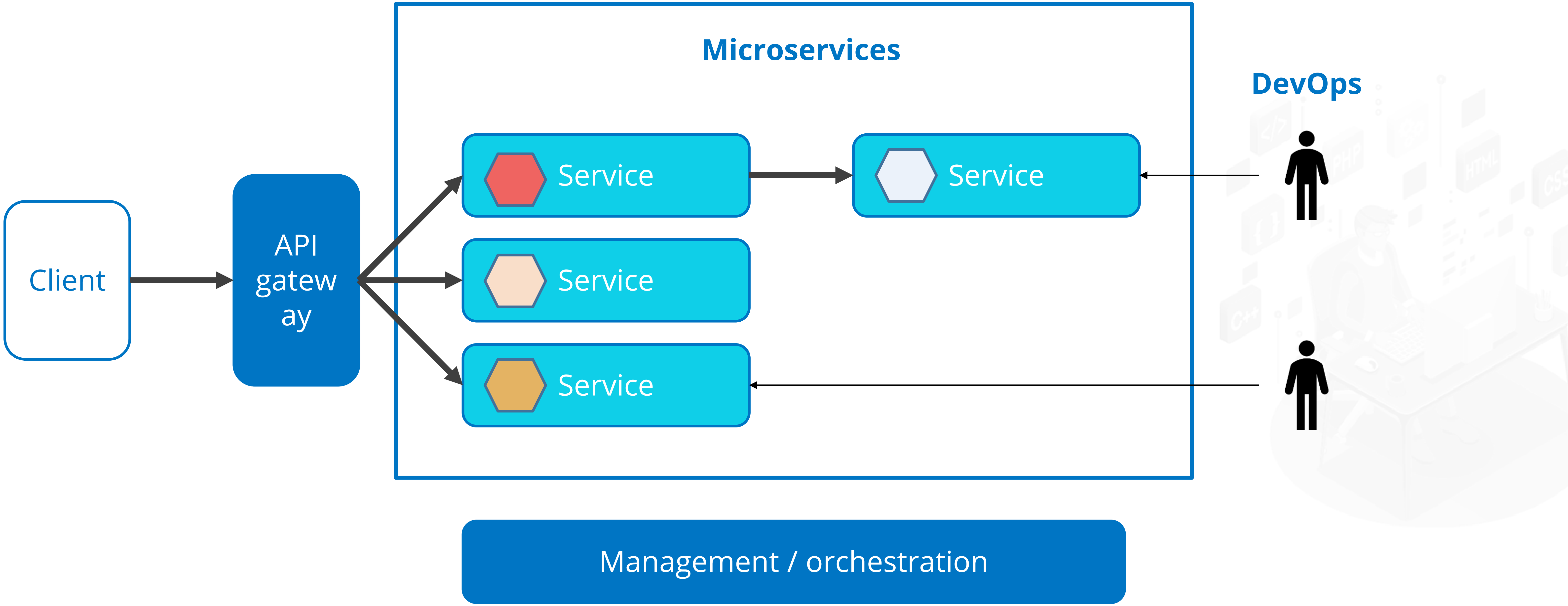
The foundation for developers and integrators create the design and implement architectures and views.

Microservices Architecture

Microservices should break the large service into many small independent services.

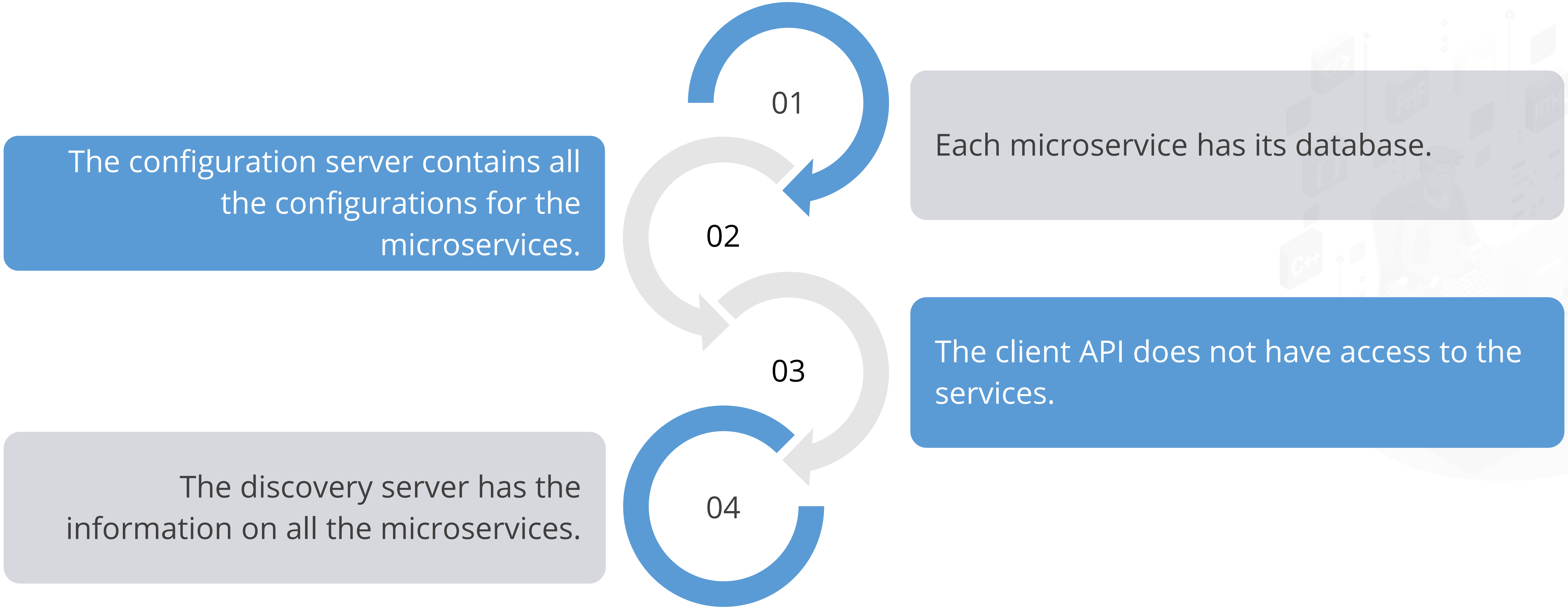
Microservices Architecture

Below is the Microservices architecture:



Microservices Architecture

Below are the characteristics of microservices architecture:



01

Each microservice has its database.

02

The configuration server contains all the configurations for the microservices.

03

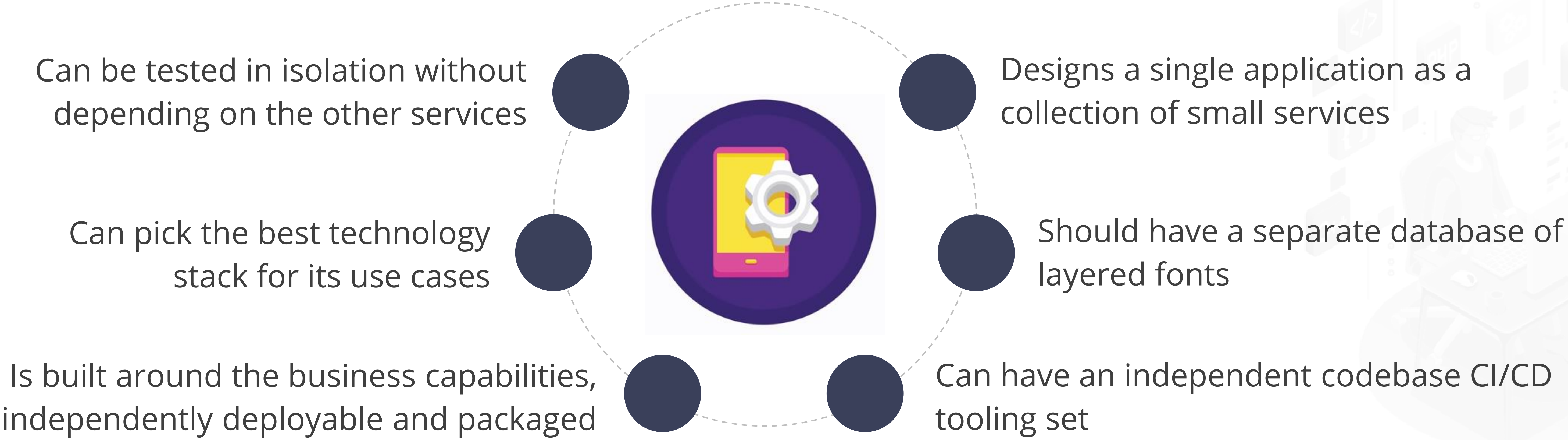
The client API does not have access to the services.

04

The discovery server has the information on all the microservices.

Microservices Architecture

Below are the characteristics of Microservices architecture. It:



Can be tested in isolation without depending on the other services

Designs a single application as a collection of small services

Can pick the best technology stack for its use cases

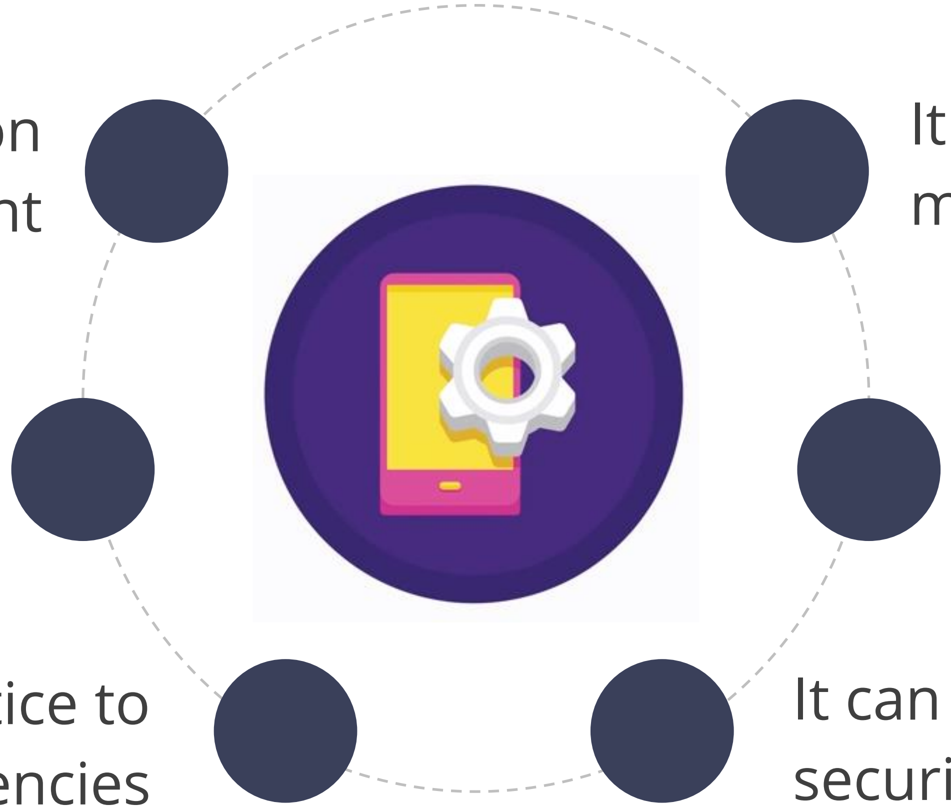
Should have a separate database of layered fonts

Is built around the business capabilities, independently deployable and packaged

Can have an independent codebase CI/CD tooling set

Microservices Architecture

Below are the characteristics of Microservices architecture. It:



It is well-understood distribution transaction management

It should have the operation team's monitoring and troubleshooting

It can use HTTP (Rest) or messaging for communication

It can implement retry functionality in case of network failure

It should employ the practice to eliminate any dependencies

It can implement an independent security mechanism

Microservices Architecture

Below are the characteristics of Microservices architecture. It:

- Has clear separation of the stateless and the stateful services
- Can run without waiting for the other service to go online
- Can use different languages, frameworks, and technologies
- Can explicitly check for the rules and the constraints

Microservices Architecture

Below are the characteristics of Microservices architecture. It:

- Does not share the libraries or the SDKs
- Can implement self-registration and the discovery mechanism
- Has an environment built to maintain compatibility with other services

TECHNOLOGY

Microservices vs. API

Microservices vs. API

Microservices are:

Small, single services offered by the company

Derived from the distributed computing architecture that connects many small services

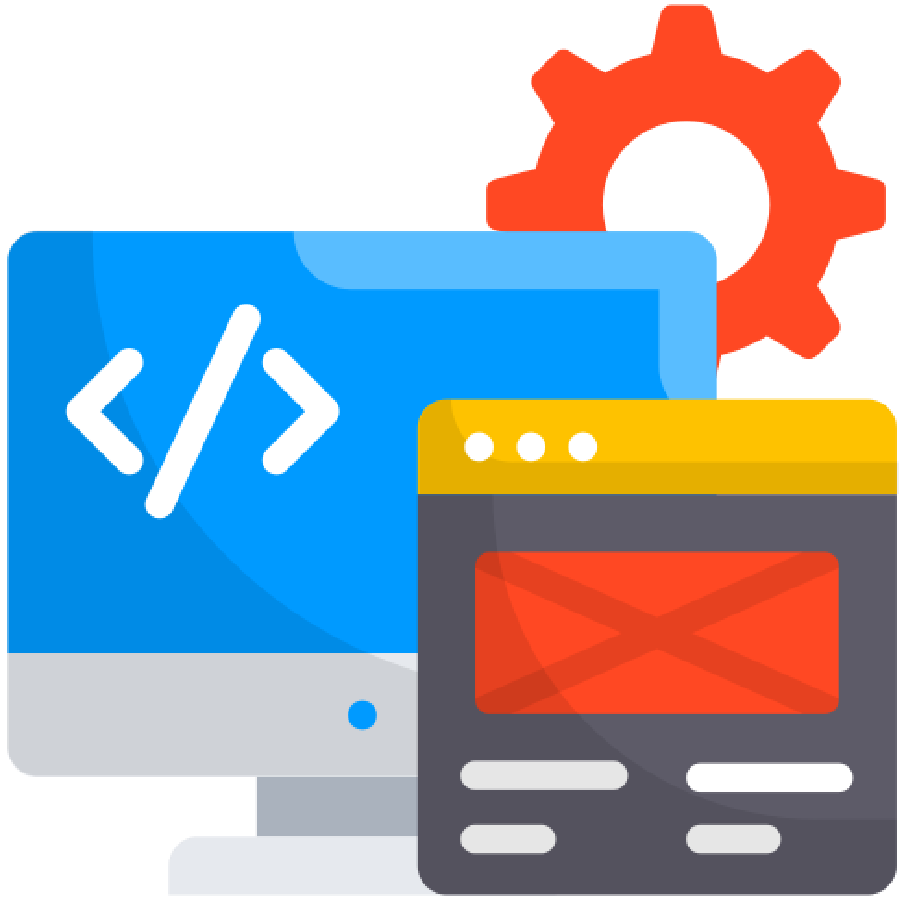
Delivered through an application programming interface (API)

Microservices vs. API

API is a method of communication between the requester and the host, most often accessed through an IP address.

Microservices vs. API

API can communicate multiple types of information to the users, that includes:



Data to be shared

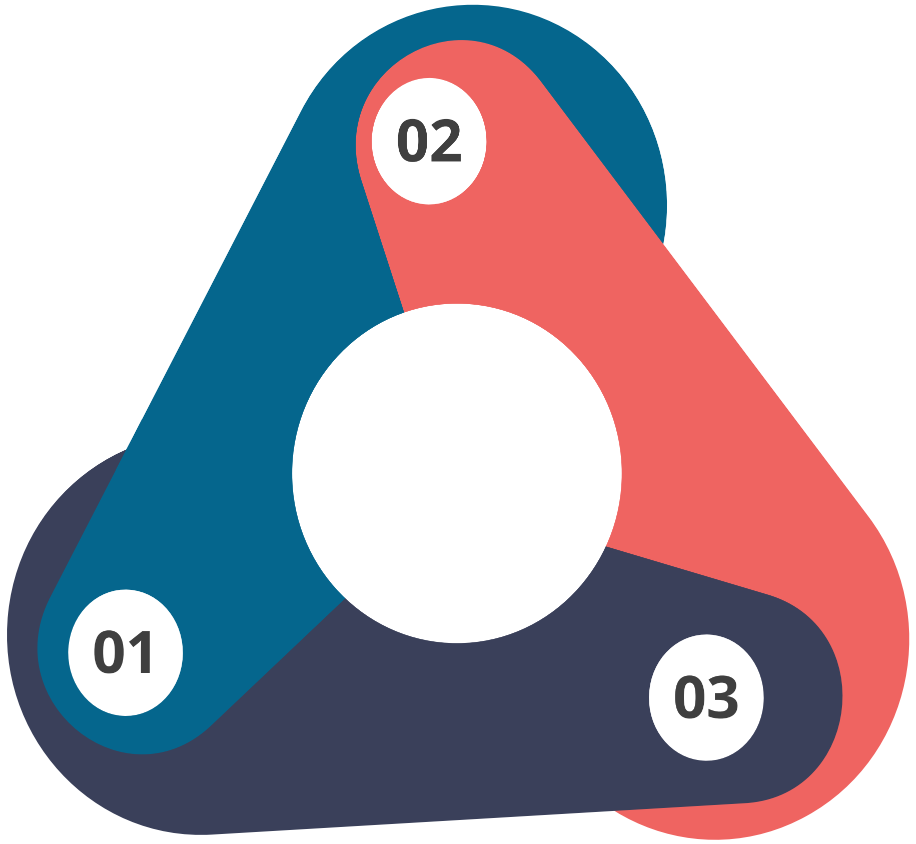
Function that needs to be provided

TECHNOLOGY

MSA vs. SOA

MSA vs. SOA

Microservice-Based Architecture (MSA):



Is better for fault tolerance

Uses lightweight protocols, such as REST and HTTP

Focuses on decoupling and follows the architectural approach

MSA vs. SOA

MSA

- Uses a simple messaging system for communication
- Has an independent database
- Uses modern relational databases
- Is better suited for the smaller and well-portioned web-based system
- Minimizes sharing through bounded context

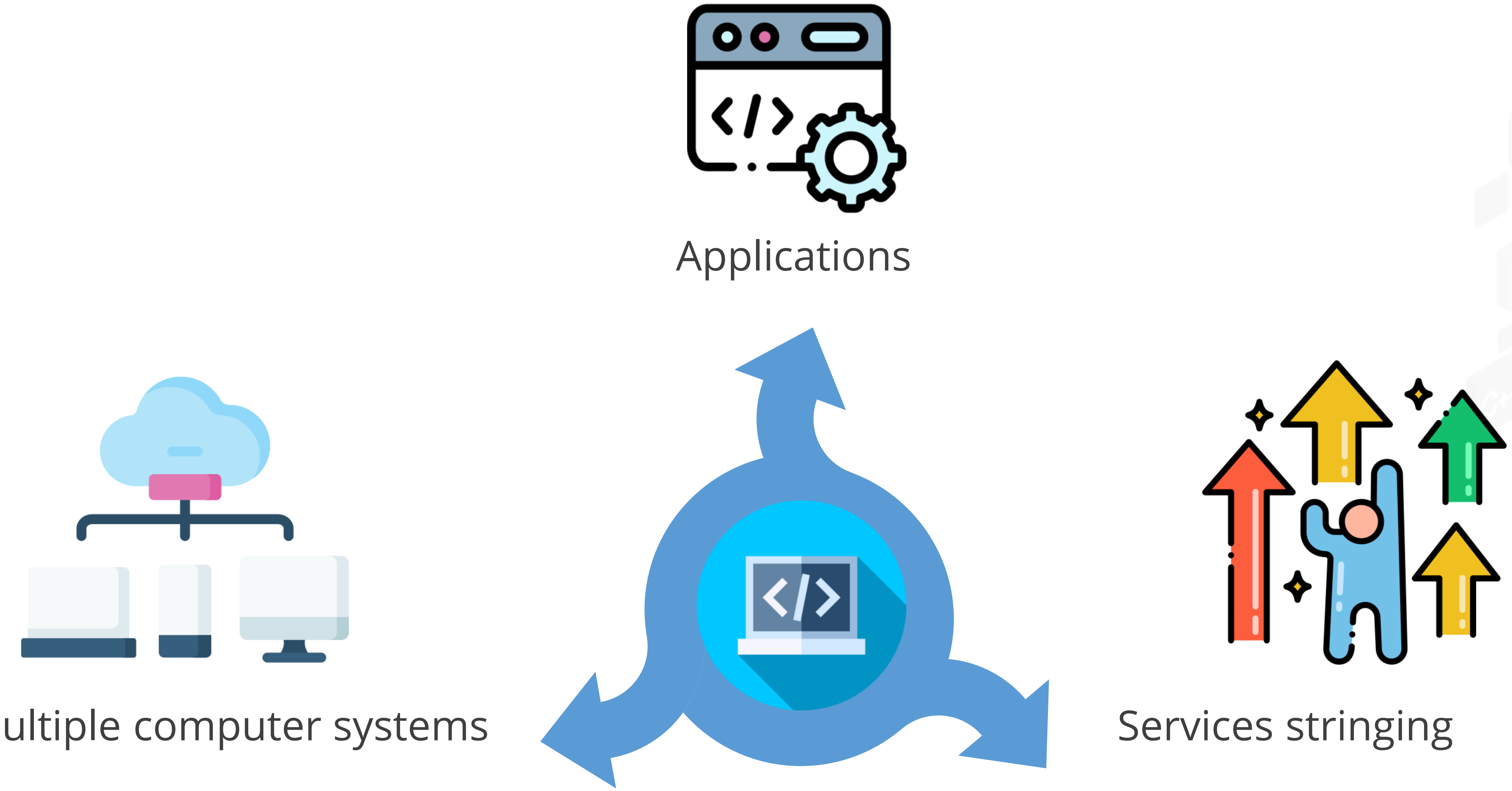
SOA

- Uses the Enterprise Service Bus (ESB) for communication
- Shares the whole data storage
- Uses traditional relational databases
- Is better for large and complex business application environments
- Enhances component sharing

Orchestration

Orchestration

Orchestration refers to the coordination and management of:



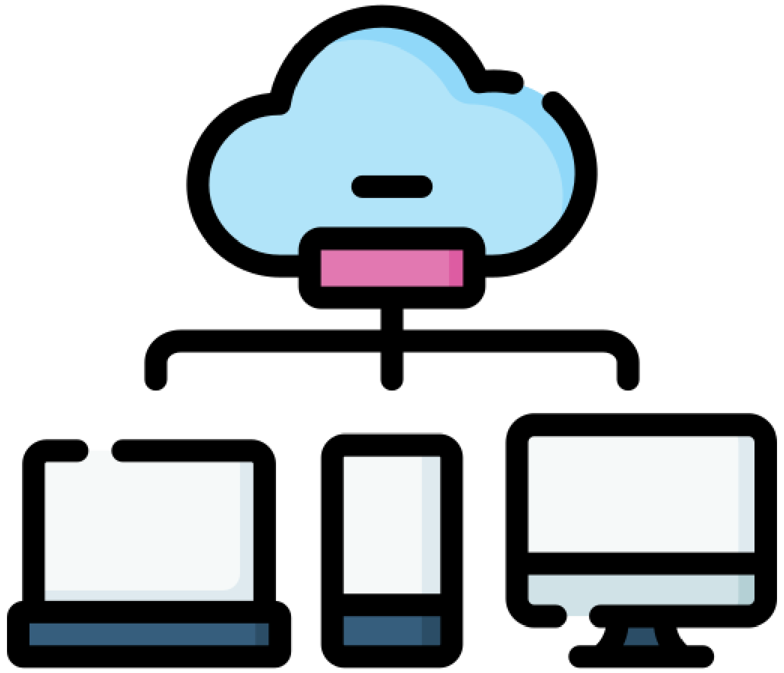
Applications

Multiple computer systems

Services stringing

Orchestration

The procedure comprises several automated operations and involves multiple systems.



It makes it easier for data teams to manage complicated activities and workflows.

Orchestration

Orchestration is used to:

Automate functions

Save time

Reduce functions

Boost efficiencies

Automation vs. Orchestration

Often used interchangeably, they have different meanings:

Automation

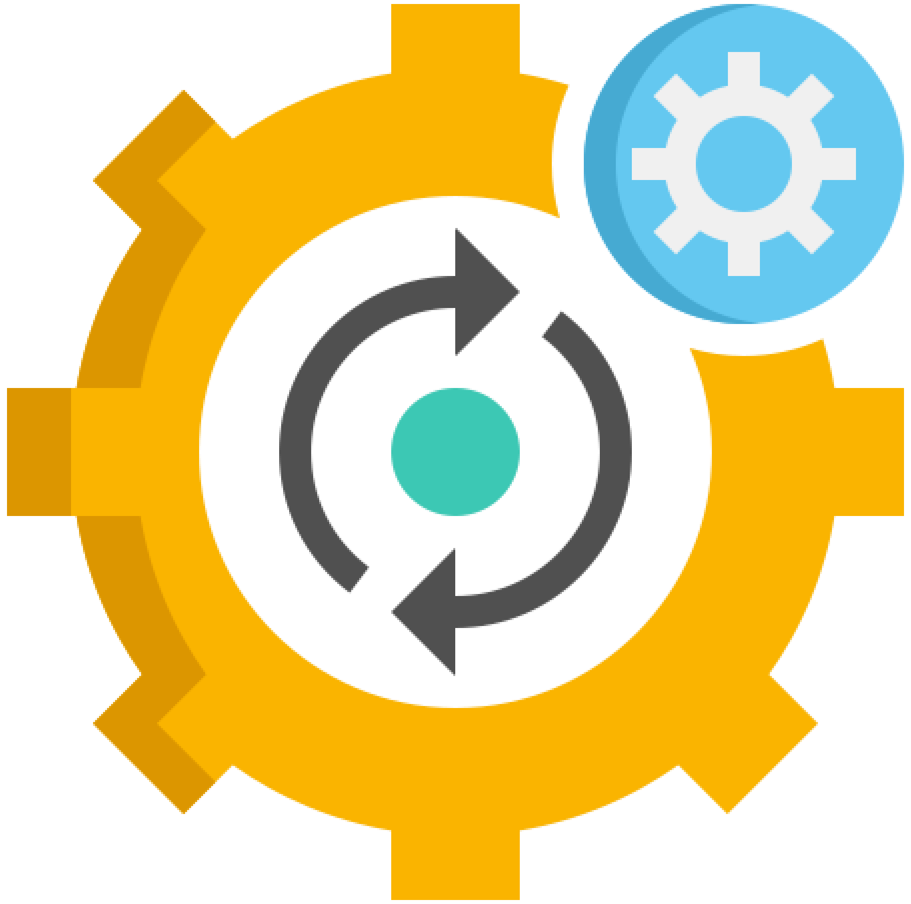
- Specific task is completed without human intervention
- Process of automating the daily task to improve efficiency

Orchestration

- Configuration of multiple tasks into one complete end-to-end process
- Automating many tasks together, not of a single task but an entire IT-driven process

Orchestration Tools

Orchestration should react to events or activities and make judgments based on the outputs of one automated task.



Orchestration Tools

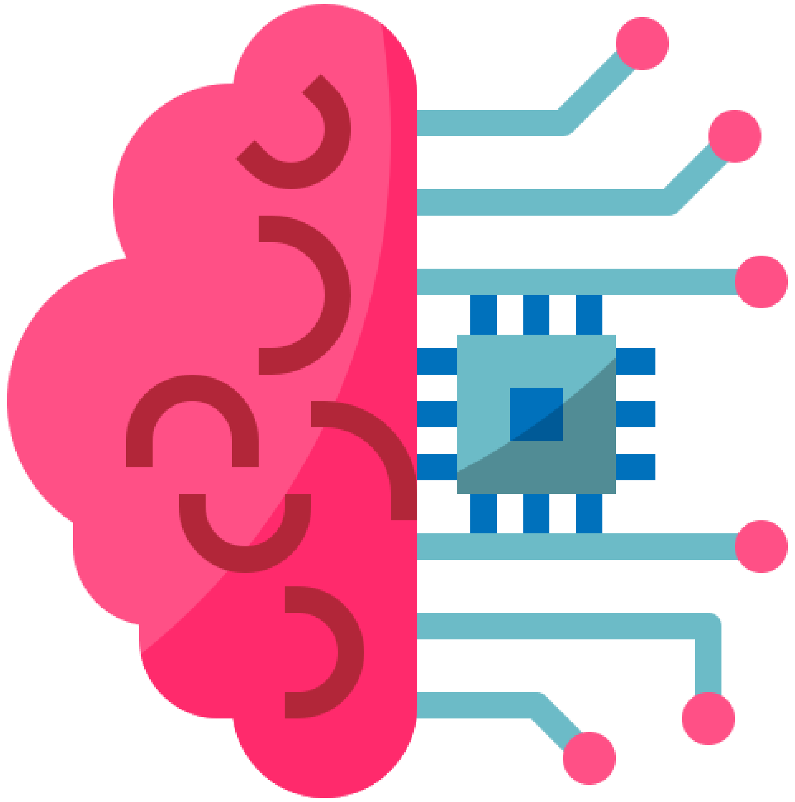
01 It requires the heavy lifting of data teams and specific technologies to create, manage, monitor, and reliably run pipelines.

02 Tools are different from the actual data or the machine learning tasks.

03 Activities are fragmented across the company, and users are forced to switch contexts frequently.

Orchestration Tools

The demand for orchestration tools has increased with companies undertaking more AI initiatives.



Orchestration Tools

It helps to unlock the full benefit of the orchestration with the framework to automate workloads.

Microservices Security

Microservices Security

There is a chance of vulnerability points when dealing with microservices.

Security is a cross-cutting concern for development teams.

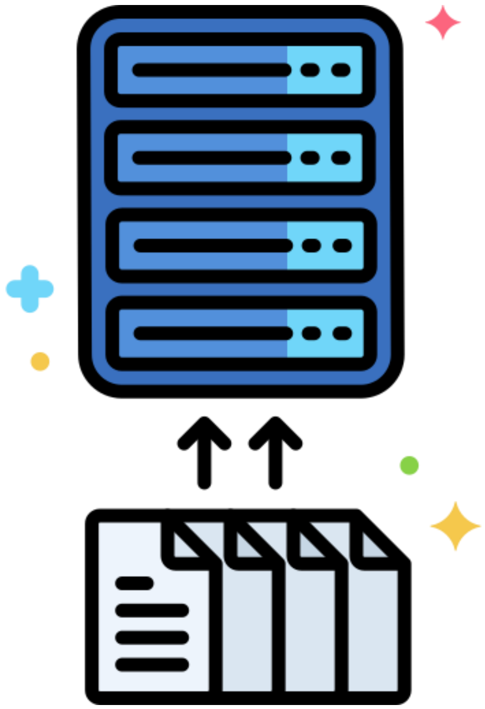
Secure Design

SAST detects vulnerabilities in code or external libraries.

DAST works externally, where malicious attacks are mimicked to check for various vulnerabilities.

External Dependencies

Using third-party libraries and external dependencies for a solution increases the possibility of vulnerabilities.



A thorough scan of the external and third-party libraries should be done whenever they are upgraded with new code contributions.

Secure HTTPS

Phishing and Credential Stuffing are common attacks.

Deploying HTTPS throughout the microservices architecture can enhance the protection of the infrastructure against network-based security breaches.

OAuth

It is an industry-standard protocol that must be implemented for user authorization.

It helps to achieve secure server-to-server communication between API Client and API Server.

Encryption

It stores secrets in environment variables.

Encrypt secrets using industry-standard tools:

HashiCorp Vault

Microsoft Azure Key Vault

Google Cloud Key Management

Amazon KMS

Create Microservice with JDBC

Problem Statement:

You have been asked to create a microservice using Spring Boot and JDBC to interact with a MySQL database.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Creating a new Spring Starter project
2. Creating a welcome page
3. Creating the Product model class
4. Setting up the database configuration
5. Creating the ProductRepository interface
6. Creating the ProductController class
7. Creating the Response class
8. Configuring the CRUD methods
9. Running and testing the application

Create Microservice with MongoDB

Problem Statement:

You have been asked to create a microservice using Spring Boot and MongoDB to perform CRUD operations on a user collection in a MongoDB database.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Creating a new Spring Starter project
2. Creating the Address Model class
3. Creating the User Model class
4. Configuring the MongoDB database
5. Setting up the database configuration
6. Creating the UserRepository interface
7. Creating the UserController class
8. Creating the Response class
9. Configuring the CRUD methods
10. Running and testing the application

Key Takeaways

- Microservices is a collection of small services designed to be self-contained and run applications.
- Microservices break down an application into smaller sections, which helps understand where a problem occurs.
- In a monolithic architecture, various elements of a software application are integrated into a single program with a single tier.
- Deploying HTTPS throughout the microservices architecture can enhance the protection of the infrastructure against network-based security breaches.

TECHNOLOGY

Thank You