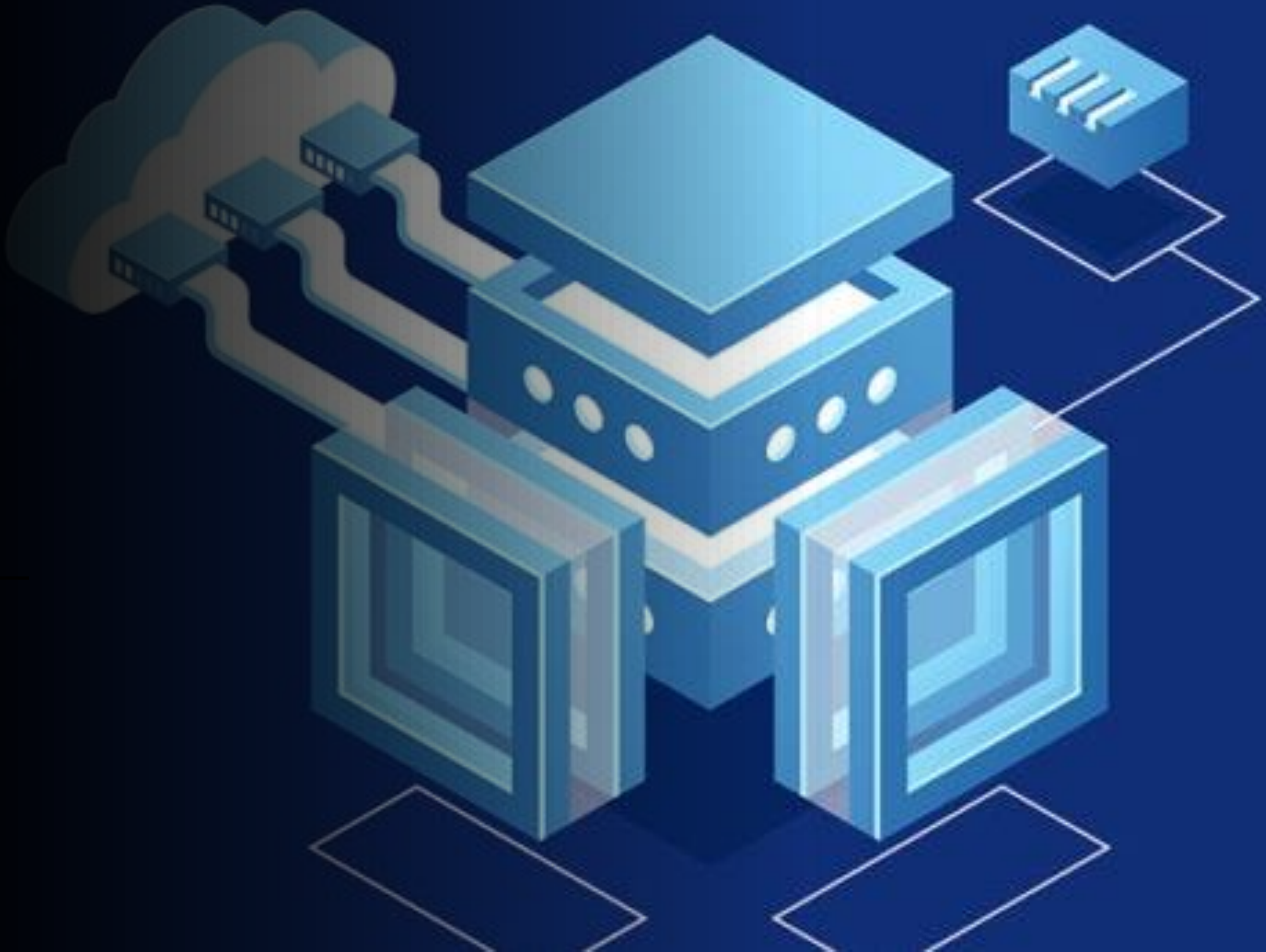
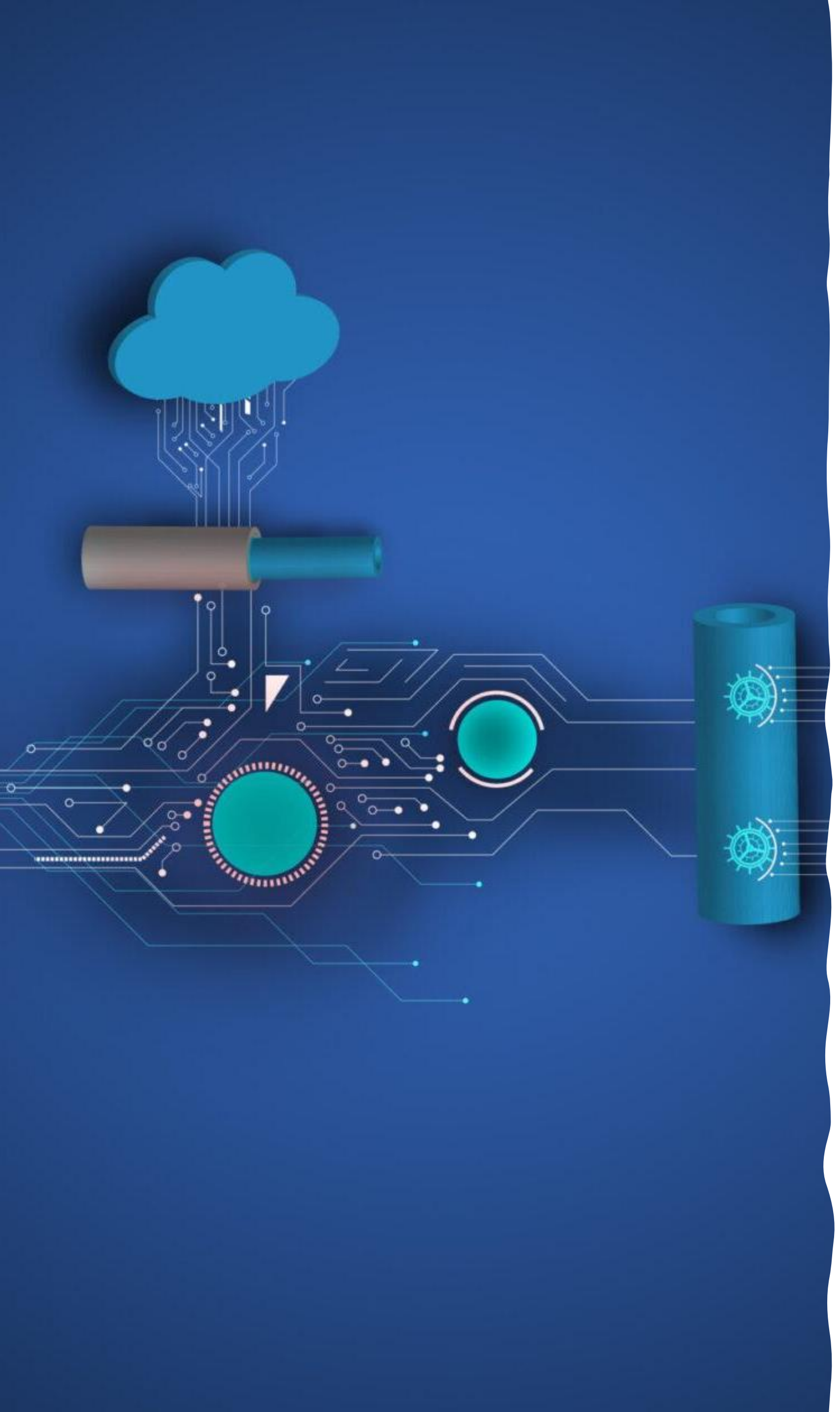
Understanding Modern Cloud Architectures

Group members

- Semini J.P.D.L. IT20241346
- Sathsarani B.A.D.A IT20160098
- Liyanage A.L.D.K.S IT20146474
- Nayanananda W.A.K.D IT20237622
- Jayathunga T. M. IT20146238

Microservices architecture

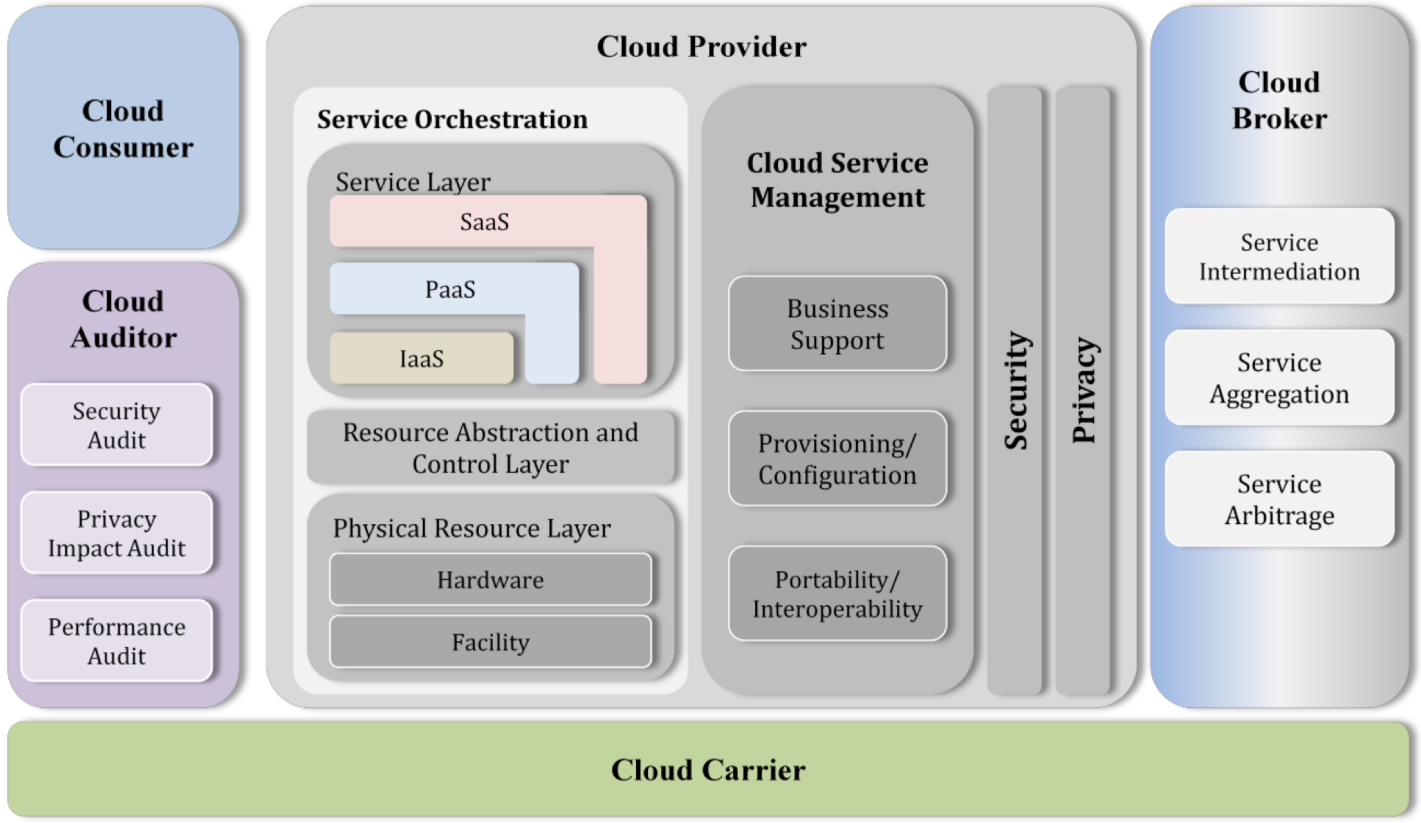




Microservices architecture

A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently.

Microservices architecture



Characteristics of microservices

Decentralized: Microservices are self-contained and do not rely on a central monolithic architecture.

Independence: Each microservice has its own database and can be developed using different technologies and programming languages.

Scalability: Microservices can be scaled individually to meet the specific demands of each service.

Resilience: If one microservice fails, it doesn't necessarily bring down the entire application.

Continuous Deployment: Microservices are designed for continuous integration and continuous deployment (CI/CD).

Benefits of Microservices

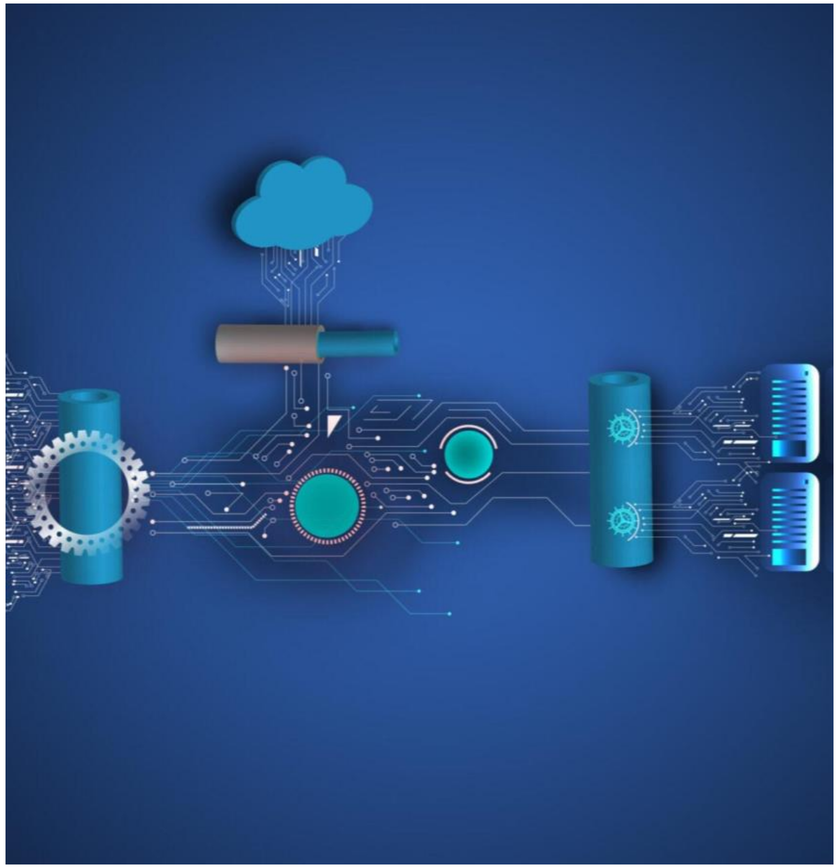
Agility: Easier to develop, test, and deploy, allowing for faster updates and feature additions.

Scalability: Each microservice can be scaled independently to meet varying workloads.

Fault Isolation: Failures in one microservice do not affect others, leading to increased application resilience.

Technology Diversity: Developers can choose the most suitable technology stack for each microservice.

Improved Collaboration: Teams can work on different microservices concurrently, promoting collaboration and parallel development.



Challenges of Microservices

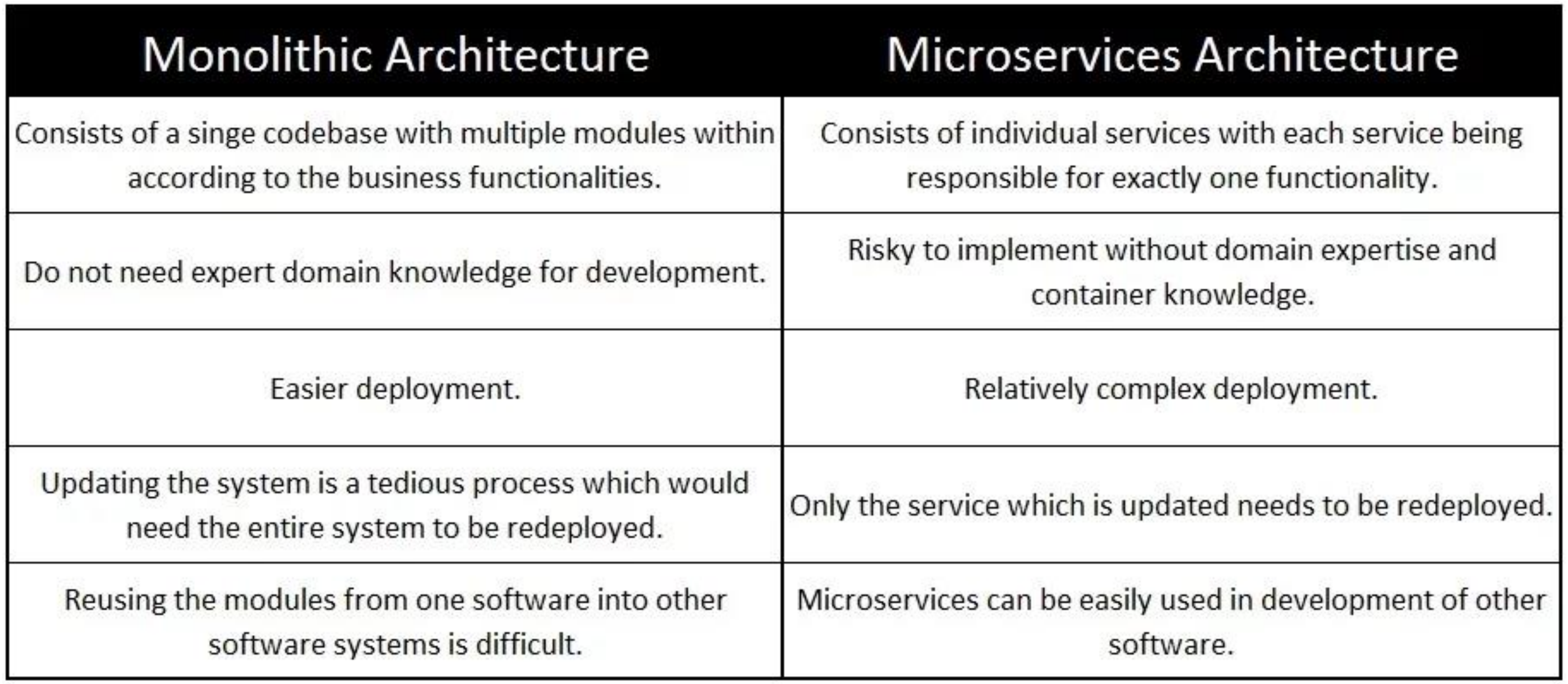
Complexity: Managing a distributed system can be more complex than a monolithic application.

Service Coordination: Effective communication between services must be established and maintained.

Data Management: Handling data consistency and transactions across services can be challenging.

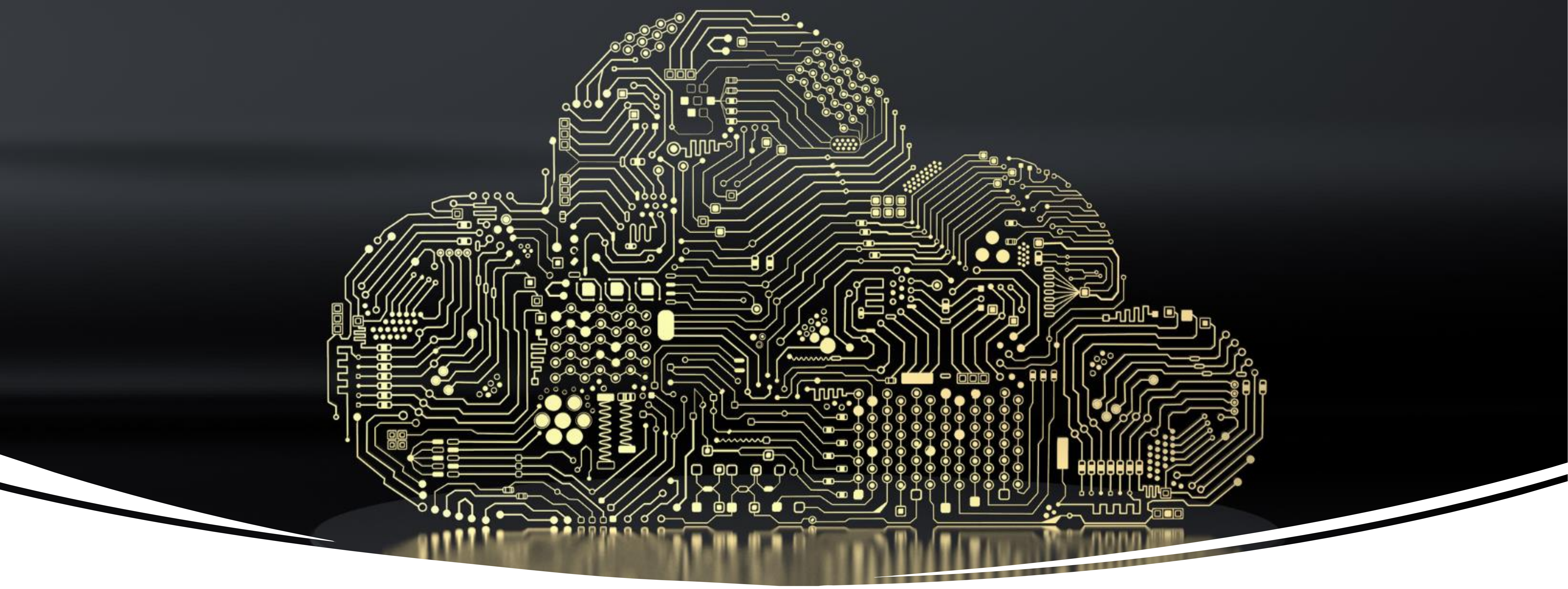
Operational Overhead: Operating and monitoring multiple services can require more resources and expertise.

Microservices Architectures vs Monolithic Architectures



Monolithic Architecture	Microservices Architecture
Consists of a singe codebase with multiple modules within according to the business functionalities.	Consists of individual services with each service being responsible for exactly one functionality.
Do not need expert domain knowledge for development.	Risky to implement without domain expertise and container knowledge.
Easier deployment.	Relatively complex deployment.
Updating the system is a tedious process which would need the entire system to be redeployed.	Only the service which is updated needs to be redeployed.
Reusing the modules from one software into other software systems is difficult.	Microservices can be easily used in development of other software.

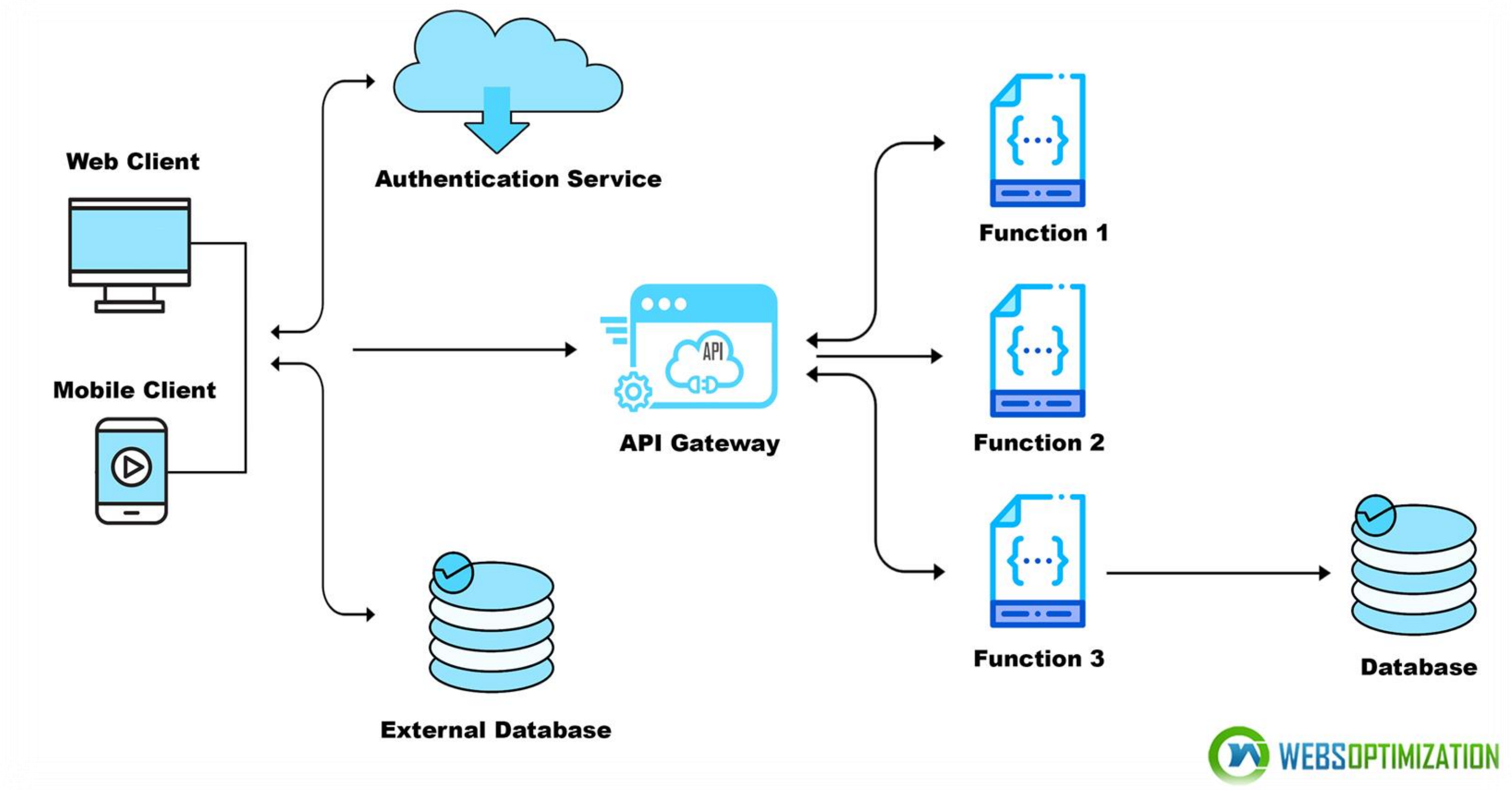
Serverless Architecture



Serverless Architecture

- Serverless architecture is an approach to software design that allows developers to build and run services without having to manage the underlying infrastructure. Developers can write and deploy code, while a cloud provider provisions servers to run their applications, databases, and storage systems at any scale.

Serverless Architecture



Benefits of Serverless Architecture

- **Auto-Scaling**: Serverless platforms automatically scale resources to match the workload, handling traffic spikes and reducing costs during idle periods.
- **Pay-Per-Use:** You only pay for the compute resources used during the execution of your functions, leading to cost-efficiency.
- **Event-Driven:** Serverless functions are triggered by events such as HTTP requests, database changes, or message queues.
- **Reduced Management Overhead:** Serverless eliminates server provisioning, maintenance, and updates, reducing operational overhead.
- **Faster Time-to-Market:** Developers can focus on writing code and deploy functions quickly, accelerating development cycles.

Limitations of Serverless Architecture

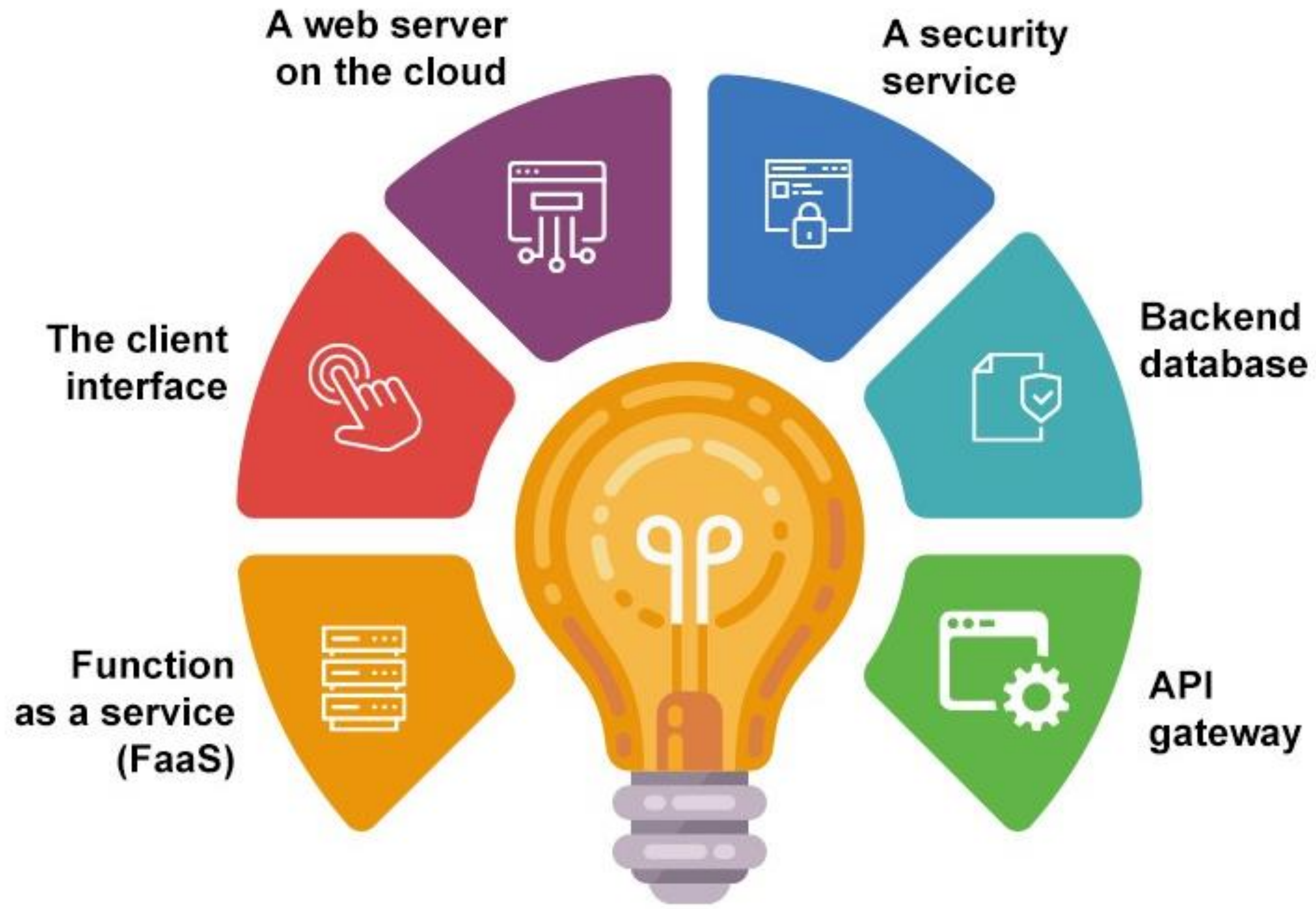
Security: As the third-party vendor manages the entire back-end, there is a risk of data security, especially for applications that handle personal or sensitive data.

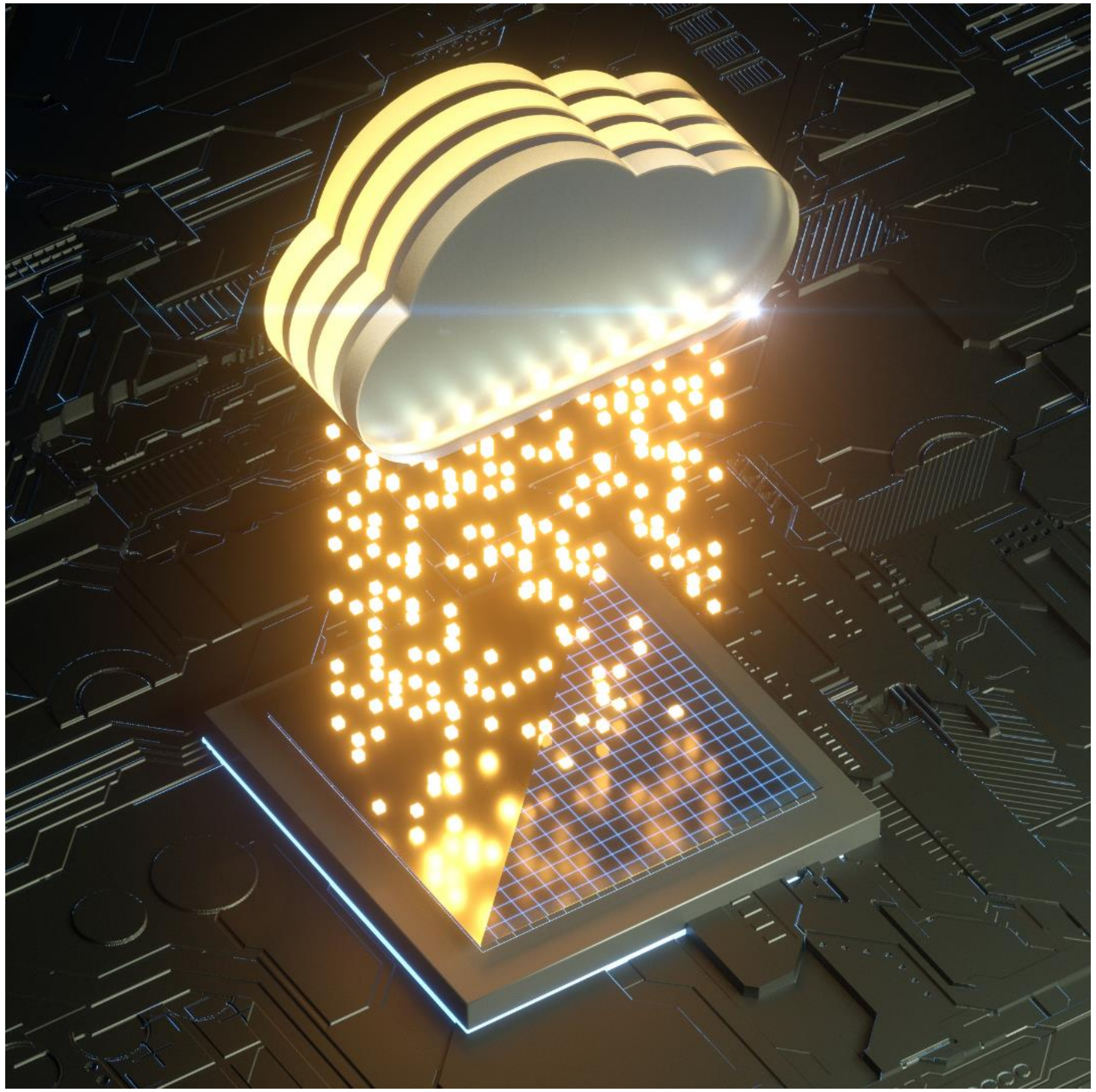
Third-party dependency:The developers have limited control over the architecture platform, so they depend on the service providers for debugging and monitoring the application.

Architecture Complexity: Due to more functions, developers face complexity and implementation problems as it takes time to assess, implement and test the functions.

IBM App Connect: An all-in-one integration tool that enables the admin to create workflows that decide how data is transferred from one application to another.

Key components of serverless architecture





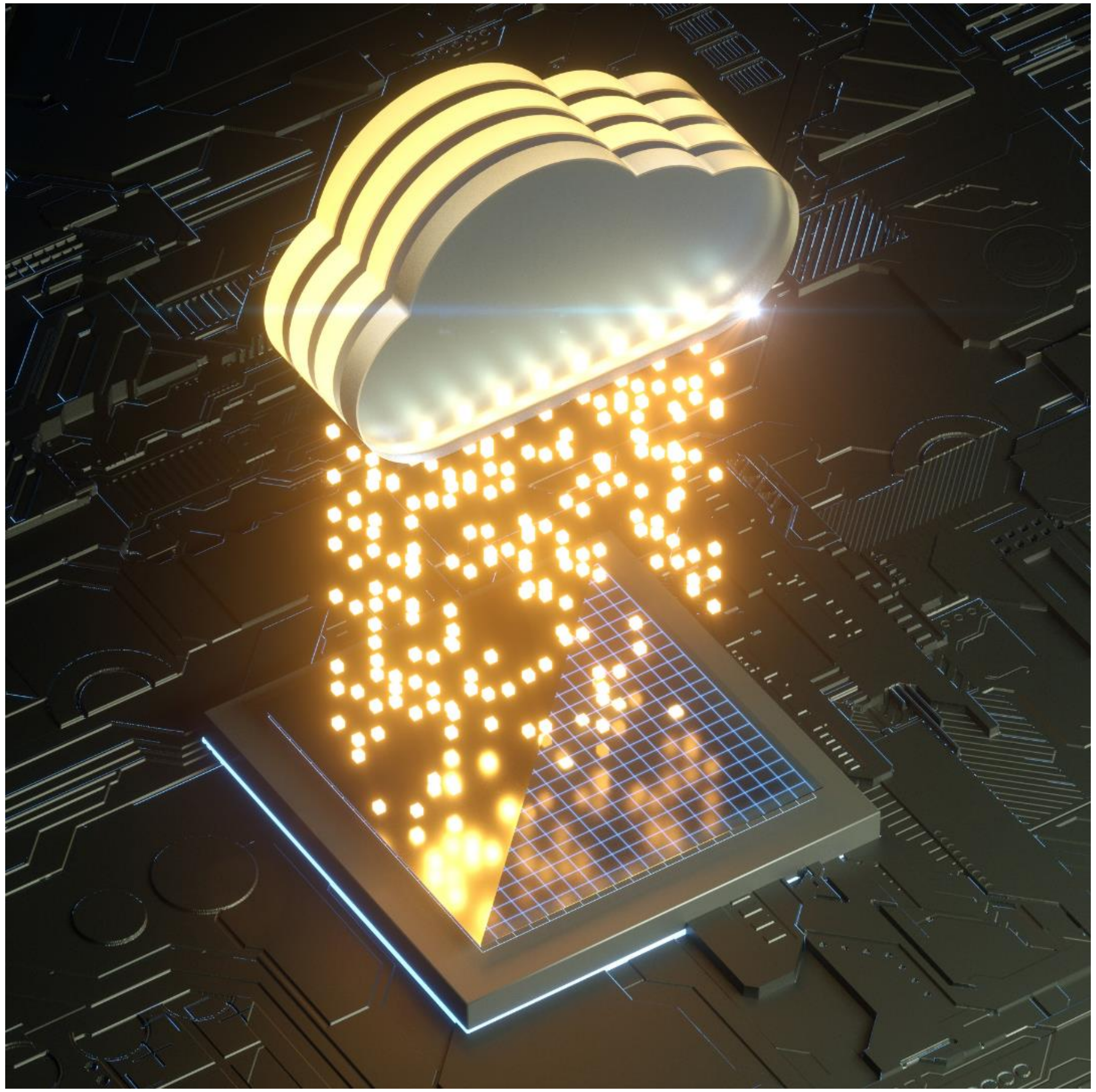
The Role of Cloud Providers in Serverless Computing

These cloud providers offer cloud computing services that fall under two groups:

1. Backend-as-a-Service (BaaS): BaaS is a cloud-based computing model that manages the back-end side of the web or mobile application development. Here the developers only write and maintain the frontend.

Examples of BaaS providers are –

- Firebase
- Azure
- Leancloud



The Role of Cloud Providers in Serverless Computing

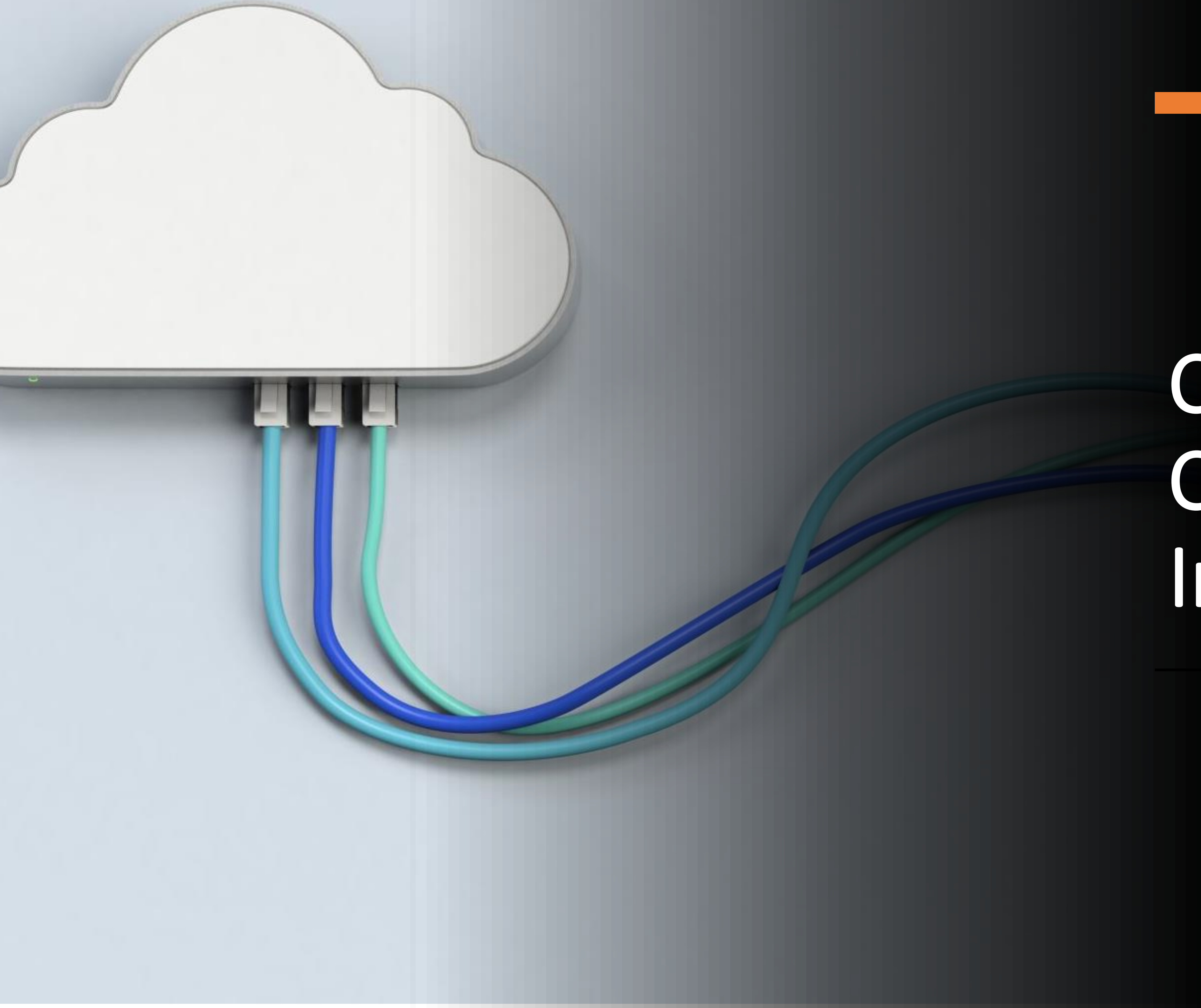
2. Function-as-a-Service (FaaS): FaaS is an event-driven execution model that lets you run small modules of code in the cloud. It offers users to execute already designed codes and is also popularly used for its real-time data processing.

Examples of FaaS providers are –

- AWS Lambda (Amazon)
- Microsoft Azure Functions
- Google Cloud Functions
- IBM OpenWhisk

Serverless vs. Traditional Server-Based Approaches

ASPECT	SERVERLESS ARCHITECTURE	TRADITIONAL SERVER-BASED APPROACHES
Scalability	Auto-scales automatically	Manual provisioning and scaling
Cost Efficiency	Pay-per-use model	Fixed costs with constant provisioning
Maintenance	Minimal maintenance	Ongoing server maintenance required
Flexibility	Event-driven, granular	Often monolithic applications
Development Speed	Faster development	Slower due to infrastructure management



Cloud Computing Integration

WHAT IS CLOUD INTEGRATION?

Cloud integration is a system of tools and technologies that connects various applications, systems, repositories, and IT environments for the real-time exchange of data and processes.

Types of Cloud Services

IaaS (Infrastructure as a Service):

- Provides virtualized computing resources over the internet.
- Users manage the operating system, applications, and data.
- Examples: AWS EC2, Azure Virtual Machines.

PaaS (Platform as a Service):

- Offers a platform with development tools and services to build, deploy, and manage applications.
- Users focus on application development, not infrastructure.
- Examples: Google App Engine, Heroku.

SaaS (Software as a Service):

- Delivers software applications over the internet on a subscription basis.
- Users access the software via a web browser, with no need for installation or maintenance.
- Examples: Microsoft 365, Salesforce, Dropbox.

Cloud Migration Strategies

Rehosting (Lift and Shift):

- Migrate existing applications to the cloud with minimal modifications.
- Suitable for cost savings and quick migration.

Replatforming:

- Make slight adjustments to applications to take advantage of cloud benefits.
- Improve performance and reduce costs.

Refactoring (Re-architecting):

- Restructure and optimize applications for the cloud.
- Achieve greater scalability, performance, and cost efficiency.

Rebuilding:

- Rebuild applications from scratch using cloud-native services.
- Highest level of cloud optimization but requires significant effort.

Hybrid Cloud and Multi-Cloud Strategies

Hybrid Cloud:

Definition: A hybrid cloud is a combination of private and public cloud resources, allowing data and applications to be shared between them.

Advantages:

Flexibility, Scalability, Cost-Efficiency, Data Security

Multi-Cloud:

Definition: Multi-cloud involves using services and resources from multiple public cloud providers, such as AWS, Azure, Google Cloud, and others, to meet specific requirements.

Advantages: Vendor Neutrality, Risk Mitigation, Service Optimization, Geographical Reach

Cloud Security and Compliance

Security in the Cloud:

- Cloud providers offer robust security measures, but users are responsible for securing their data and applications.
- Implement encryption, identity and access management (IAM), and regular audits.

Compliance:

- Cloud services often comply with industry-specific regulations (e.g., HIPAA, GDPR).
- Understand compliance requirements and ensure cloud services adhere to them.

Shared Responsibility Model:

- Differentiates between cloud provider and user responsibilities for security.
- Users handle data and application security, while providers secure the infrastructure.

Best Practices:

- Regularly update and patch applications and systems.
- Monitor and audit activities for security incidents.
- Implement disaster recovery and backup plans.

Cloud Integration Platforms

Zapier: Provides code-free application integration to expanding companies.

MuleSoft Anypoint: Empowers management of all the application programming interfaces (APIs) and integrations over a centralized platform.

Dell Boomi: Allows fusing the apps, data, and processes into one. It helps manage data quality governance, application integration, and B2B network and build workflows with minimal coding.

IBM App Connect: An all-in-one integration tool that enables the admin to create workflows that decide how data is transferred from one application to another.

Microsoft Azure Logic Apps: Enables administrators to automate workflows that integrate apps and business data across on-premises systems and cloud services.

Cloud Services

Infrastructure as a Service (IaaS)

Definition: IaaS provides virtualized computing resources over the internet, including virtual machines, storage, and networking.

User Responsibilities: Users manage the operating system, applications, and data.

Use Cases: Ideal for businesses that need scalable infrastructure without the responsibility of managing physical hardware.

Examples: AWS EC2, Azure Virtual Machines, Google Compute Engine.

Platform as a Service (PaaS)

Definition: PaaS offers a platform with development tools and services for building, deploying, and managing applications.

User Focus: Users concentrate on application development, while the platform handles infrastructure management.

Benefits: Accelerates development, reduces administrative overhead, and ensures platform-level scalability.

Examples: Google App Engine, Heroku, Azure App Service.

Software as a Service (SaaS)

Definition: SaaS delivers software applications over the internet on a subscription basis.

Access: Users can access the software via a web browser without installation or maintenance.

Use Cases: Common for productivity tools, collaboration, and business applications.

Examples: Microsoft 365, Salesforce, Dropbox.

Containers and Kubernetes

Containers: Containers package applications and their dependencies for efficient and consistent deployment across different environments.

Kubernetes: Kubernetes is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications.

Benefits: Portability, scalability, and efficient resource utilization.

Examples: Docker for containers, Kubernetes for orchestration.

Serverless and Function as a Service (FaaS)

Serverless: Serverless computing allows developers to build and run applications without managing servers. Resources automatically scale with demand.

Function as a Service (FaaS): In FaaS, applications are broken down into individual functions that are triggered by events and executed on-demand.

Benefits: Cost-efficiency, scalability, reduced operational overhead, and event-driven architecture.

Examples: AWS Lambda, Azure Functions, Google Cloud Functions.

A real-world scenarios

How Microservices Improved a Company's Scalability

Scenario: A popular e-commerce company experienced rapid growth in customer traffic during the holiday season. Their monolithic architecture struggled to handle the load, resulting in slow response times and occasional outages.

Solution: The company transitioned to a microservices architecture, breaking down its monolith into smaller, more manageable services. Each service could scale independently based on its specific load, ensuring a seamless shopping experience during peak times.

Results:

Improved scalability, ensuring high performance during peak demand.

Enhanced fault isolation, reducing the impact of service failures.

Faster development cycles with smaller, focused teams.

Serverless Solutions for Cost Savings

Scenario: A financial services startup aimed to develop a customer-facing chatbot to provide assistance and information to users. They needed a cost-effective solution while ensuring excellent user experience.

Solution: The startup adopted serverless architecture to build and deploy the chatbot. By using a Function as a Service (FaaS) approach, they only paid for the chatbot's actual usage, which resulted in substantial cost savings.

Results:
Significantly reduced infrastructure costs due to pay-as-you-go pricing.
Scalable to handle increased user demand without additional expenses.
The focus remained on chatbot development and user experience.

Cloud Services for a Digital Transformation in Healthcare

Scenario: A large healthcare institution with multiple hospitals, clinics, and administrative departments aimed to undergo a comprehensive digital transformation to streamline patient care, improve access to medical records, and enhance collaboration among healthcare professionals.

Solution: The healthcare institution adopted a combination of cloud services to address their diverse needs. Here's how different cloud services were employed:

- **IaaS:** They utilized Infrastructure as a Service (IaaS) for hosting electronic health records (EHR) systems and databases, ensuring high availability and scalability.
- **PaaS:** A Platform as a Service (PaaS) solution was implemented for developing and running custom healthcare applications, enabling rapid development and deployment without worrying about underlying infrastructure.
- **SaaS:** They integrated various Software as a Service (SaaS) solutions for specific needs, such as patient appointment scheduling and telemedicine services, allowing for cost-effective and efficient management.

Cloud Services for a Digital Transformation in Healthcare

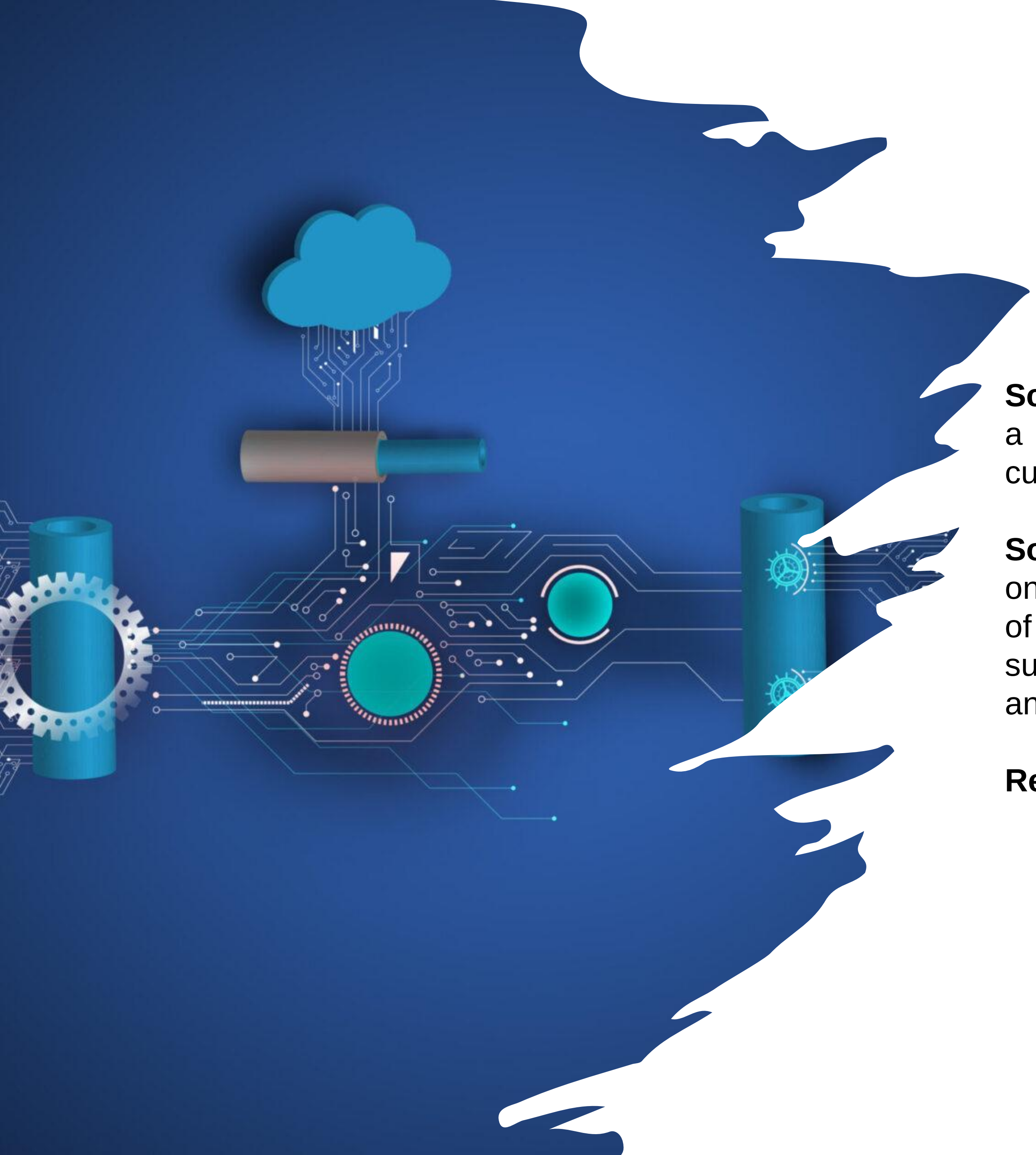
Results:

Improved patient care: Access to real-time medical records and collaboration tools led to faster and more informed decision-making by healthcare providers.

Scalability and cost efficiency: Cloud-based infrastructure allowed for the rapid scaling of resources during peak demand periods and cost savings during quieter times.

Enhanced data security: Rigorous data protection measures ensured patient data privacy and compliance with healthcare regulations.

Streamlined operations: Administrative tasks were automated, reducing manual workloads and optimizing resource allocation.



Cloud Integration for a Digital Transformation Project

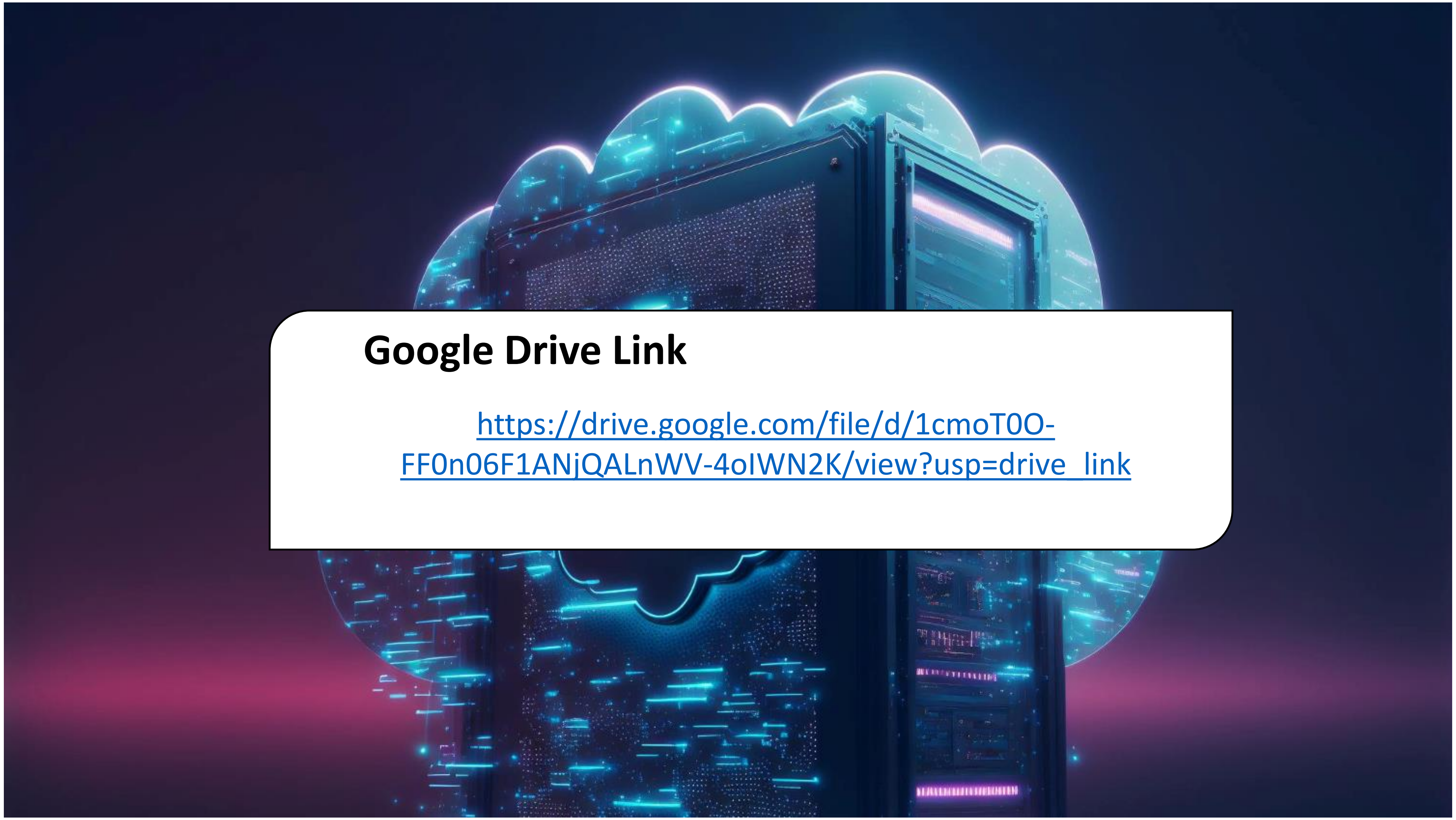
Scenario: A traditional manufacturing company decided to embark on a digital transformation journey to optimize its operations, improve customer service, and increase data-driven decision-making.

Solution: The company adopted a multi-cloud strategy, integrating on-premises systems with cloud services. They utilized a combination of IaaS, PaaS, and SaaS to support various aspects of their business, such as inventory management, customer relationship management, and data analytics.

Results:

- Enhanced agility and flexibility for adapting to market changes.
- Improved operational efficiency through automation and real-time data access.
- Better customer service and satisfaction due to more robust systems.

Thank you



Google Drive Link

https://drive.google.com/file/d/1cmoT0O-FF0n06F1ANjQALnWV-4oIWN2K/view?usp=drive_link