# MODERNIZING APPLICATIONS: INTEGRATING MICROSERVICES, SERVERLESS ARCHITECTURE, AND CLOUD COMPUTING

Group ID = CC - 006

# CONTENT WITH CAPTION

1. Microservices architecture overview
2. Serverless architecture overview
3. Cloud Computing Integration
4. Cloud Services
5. A real-world scenario

IT20081416
Ahamed M.M.Z

# Microservices Architecture Overview

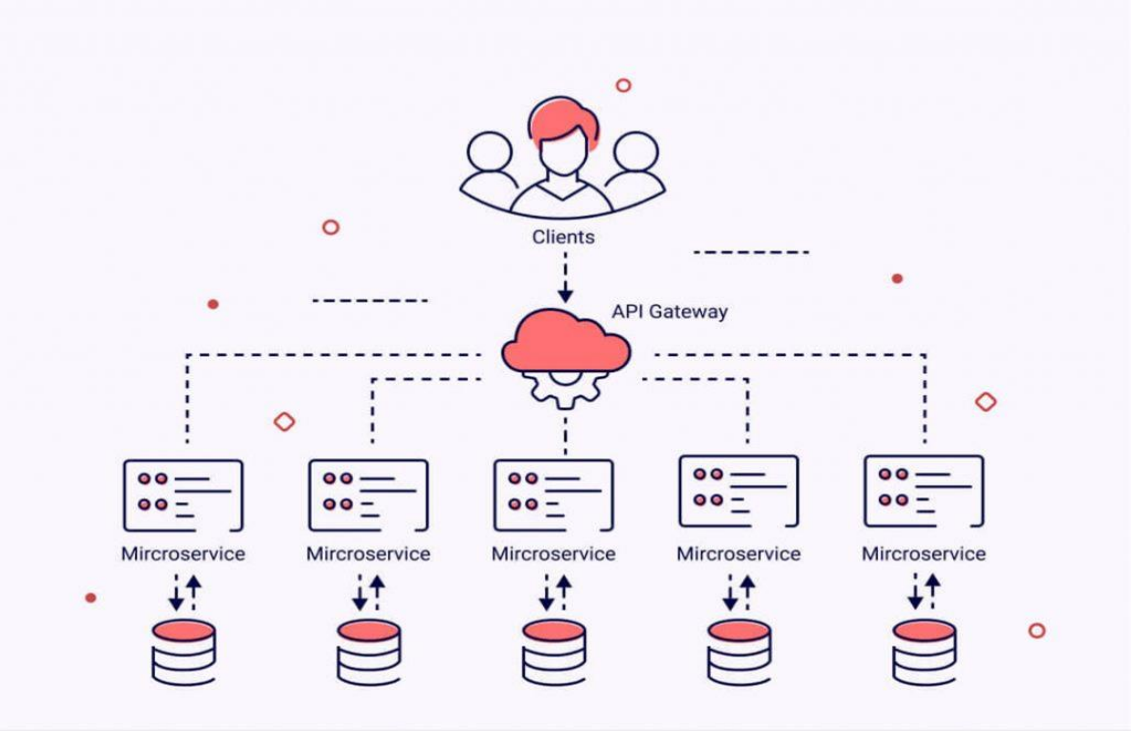
**Group ID = CC - 006**

## Characteristics of Microservices

- Decentralization
- Independence
- Scalability
- Resilience
- Continuous Delivery

## What is a Microservices architecture?

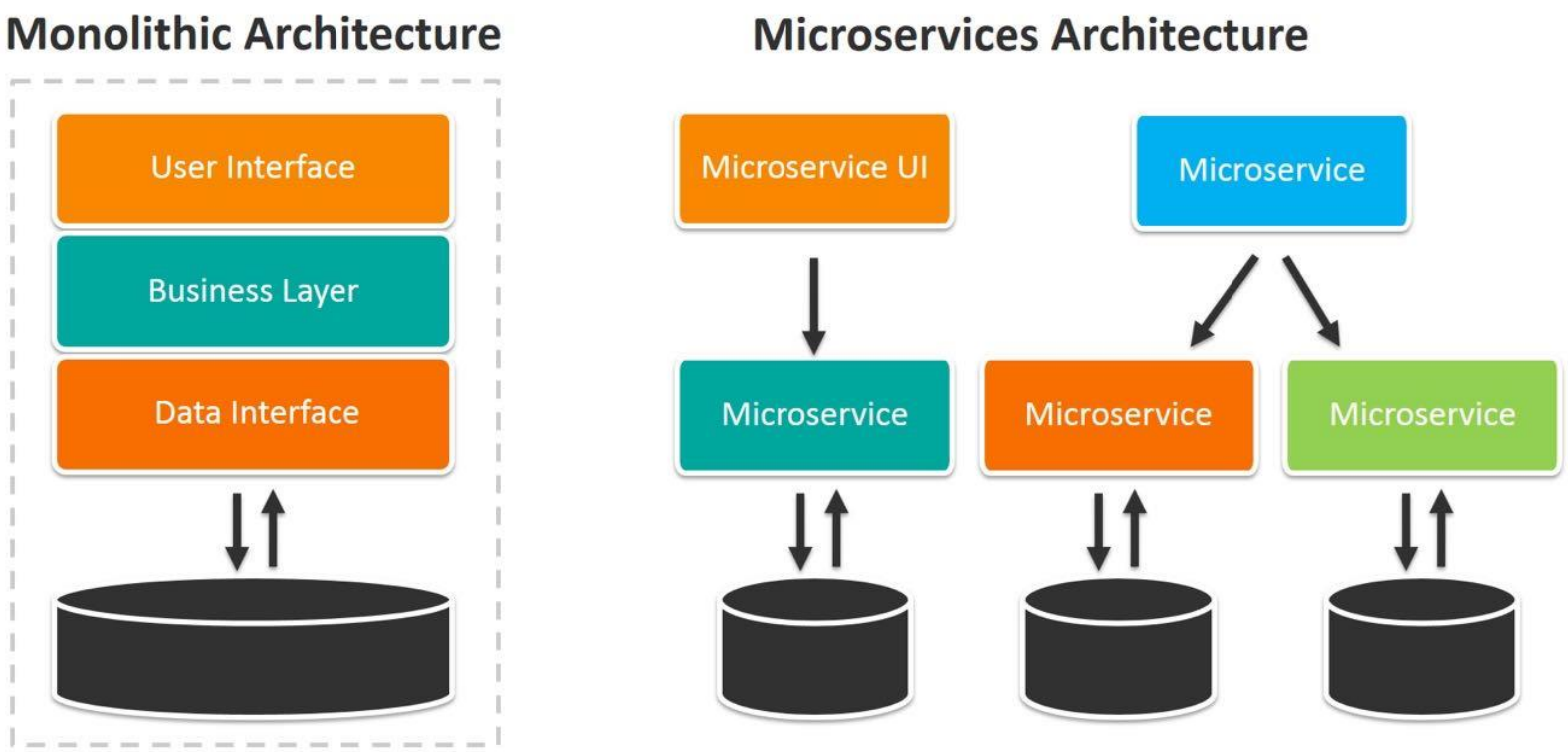
- A microservices architecture is **a type of application architecture where the application is developed as a collection of services**.

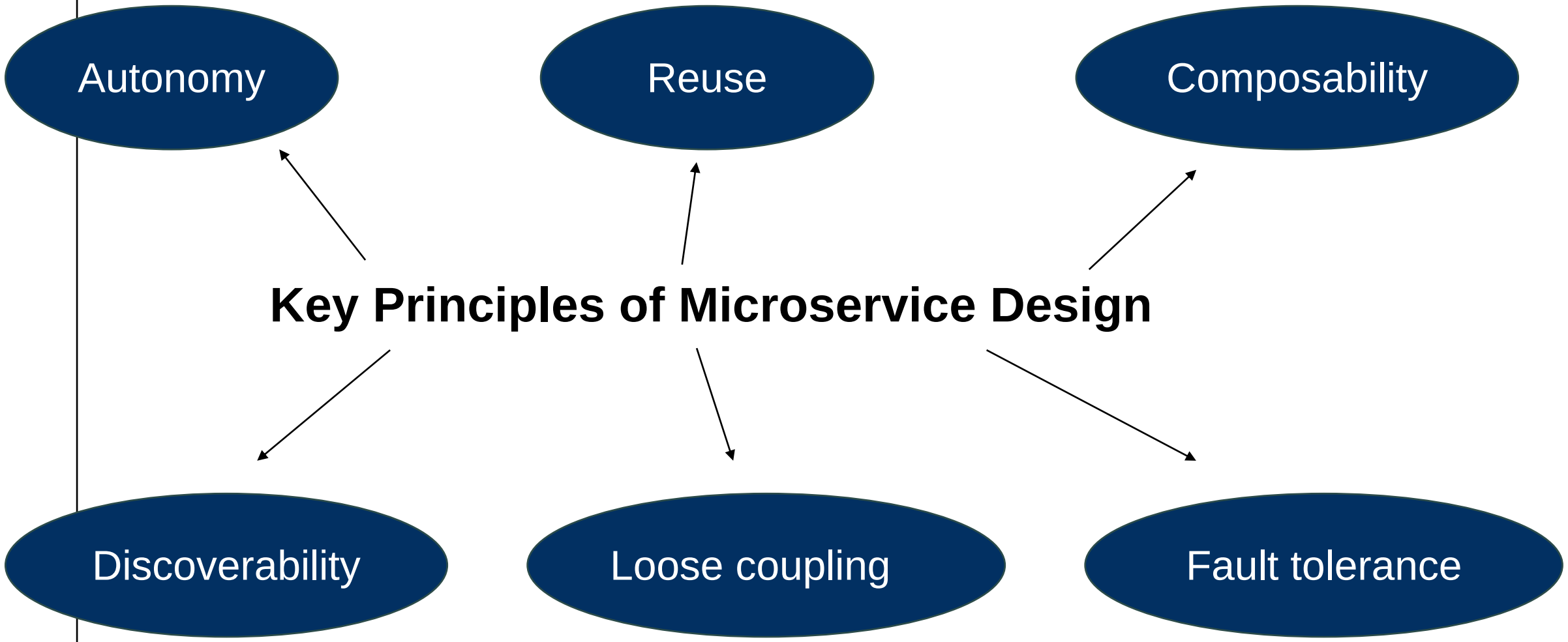


# Advantages and Disadvantages of Microservices

| **Advantages** | Disadvantages |
| --- | --- |
| Easier Scaling Up | Increased Complexity of Communication |
| Improved Fault Tolerance | Requires More Resources |
| Ease of Understanding of the Codebase | Global Testing and Debugging is Difficult |
| Scope for Experimenting | Not Practical for Small Applications |
| Independent Deployment | Relatively Complex Deployment |

# Microservices vs. Monolithic Architecture




Autonomy
Reuse
Composability
Key Principles of Microservice Design
Discoverability
Loose coupling
Fault tolerance

# REFERENCES

[*1*] JASENKA DIZDAREVI**Ć**, F. C. (19 Feb 2019). A Survey of Communication Protocols for Internet of Things.

[2]Carlos Oberdan Rolim, F. L. (2016). A Cloud Computing Solution for Patient's Data Collection in Health Care.

[3]Christos Stergiou, K. E.-G. (28 November 2016). Secure integration of IoT and Cloud Computing.

[4] Adhitya Bhawiyuga*1, D. P. (June 2019). Architectural design of IoT-cloud computing.

[5]Sara Rodríguez, D. I. ( 16 Nov 2017). Cloud Computing *Integrated into Service-Oriented.*

IT20122782
Amani M. P. N.

# Serverless Architecture Overview

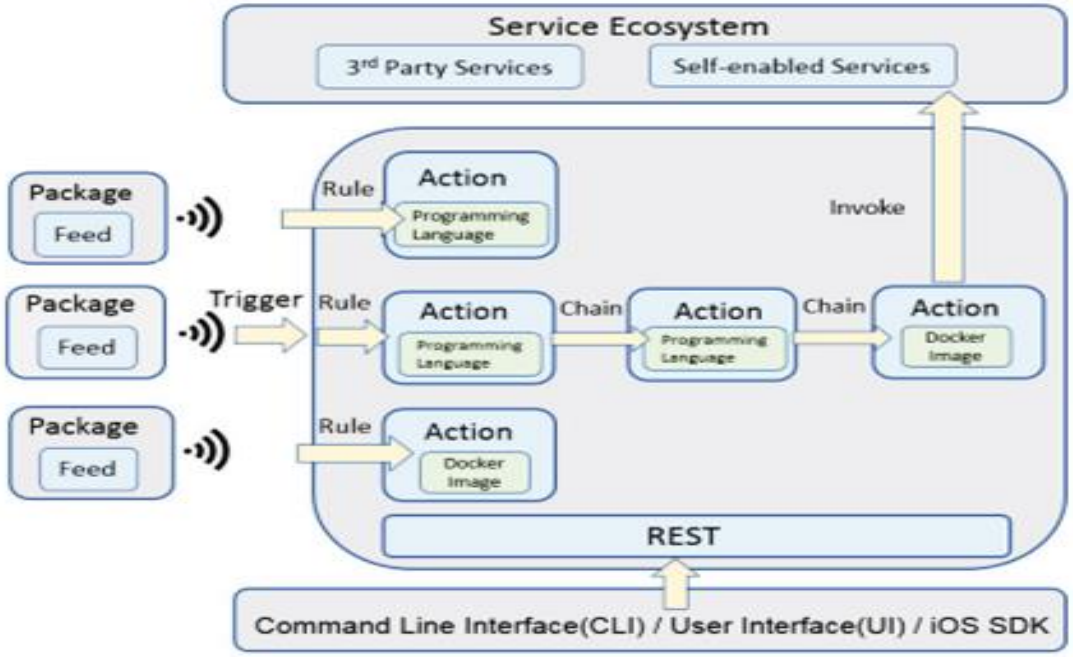
**Group ID = CC - 006**

## Key features of serverless architecture

1. Event-Driven
2. No Server Management
3. Pay-Per-Use
4. Stateless
5. Automatic Scaling

## What is serverless computing?

"Serverless Computing is a form of cloud computing which allows to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user provided functions.“ [1]




Serverless architecture

# Serverless computing providers

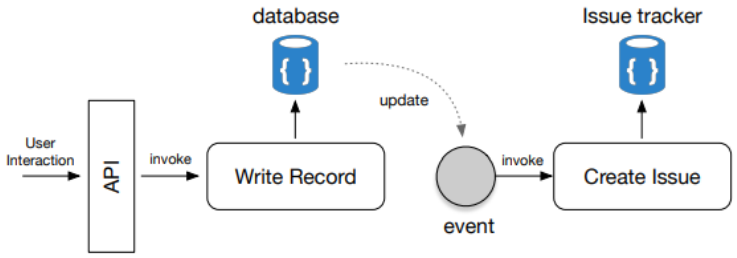
1. Amazon Web Services (AWS) Lambda: AWS Lambda is one of the most popular serverless platforms, offering support for various programming languages and a wide range of event sources. It integrates seamlessly with other AWS services.

2. Microsoft Azure Functions: Azure Functions is Microsoft's serverless offering. It supports multiple programming languages and can be integrated with Azure services like Azure Cosmos DB, Azure Event Hubs, and Azure Logic Apps.

3. Google Cloud Functions: Google Cloud Functions allows developers to run single-purpose functions in response to events from Google Cloud services or HTTP requests. It is integrated with other Google Cloud services like Cloud Pub/Sub and Cloud Storage.

4. IBM Cloud Functions: IBM's serverless platform, IBM Cloud Functions, is based on Apache OpenWhisk. It supports various languages and can be used to build event-driven applications.

5. Alibaba Cloud Function Compute: Alibaba Cloud offers Function Compute, a serverless platform that can be used for event-driven computing, such as processing HTTP requests, timers, and message queue triggers.




Flow control for Issue Tracking

## Use cases and scenarios for serverless applications

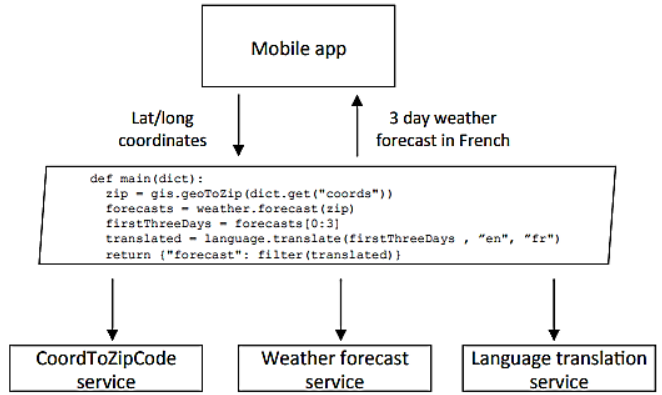
1. Web and Mobile Backends: Serverless is often used to build the backend for web and mobile applications. It can handle tasks like user authentication, data processing, and database operations in response to API requests.

2. Image and Video Processing: Serverless functions can process media files, such as resizing images, generating thumbnails, or transcoding videos.

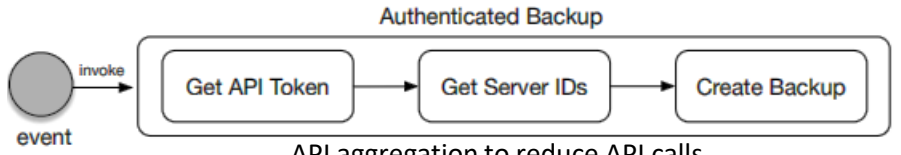
3. API Gateway: Building API endpoints and managing API traffic is a common use case for serverless, allowing for easy scaling and cost-effective management of APIs.

4. Event-Driven Applications: Serverless is ideal for event-driven architectures, where functions are triggered by events like user actions, system events, or third-party integrations.

5. Real-time Data Processing: Serverless can process real-time data streams, such as IoT data, sensor data, or logs, to perform tasks like data transformation, aggregation, and alerting.




API composition




API aggregation to reduce API calls

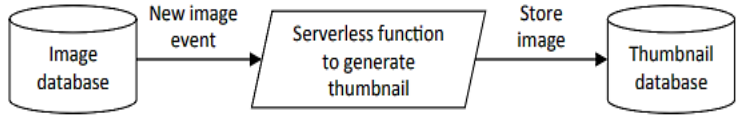



Image processing

# Pros and cons of serverless architecture

| Advantages | Disadvantages |
| --- | --- |
| **Event-Driven: Serverless is inherently event-driven, making it suitable for real-time applications and microservices architecture.** | **Security Concerns: Serverless applications require careful security configuration to prevent unauthorized access and data breaches.** |
| **Cost-Efficiency: Serverless services often charge based on actual usage, so you only pay for the compute resources you consume. This can lead to cost savings, especially for sporadic workloads.** | **Vendor Lock-In: Serverless functions are often tightly integrated with a specific cloud provider, making it challenging to migrate to another platform without significant rework.** |
| **Auto-Scaling: Serverless platforms automatically scale resources based on incoming requests or events, ensuring optimal performance and minimal downtime during traffic spikes.** | **Debugging and Monitoring: Debugging and monitoring serverless functions can be more challenging, as you have less visibility into the underlying infrastructure.** |
| **Simplified Operations: Serverless abstracts server management, OS patching, and infrastructure scaling, reducing the operational burden on development and IT teams.** | **Limited Language Support: Some serverless platforms support a limited set of programming languages, which can be a constraint for developers.** |
| **Rapid Development: Developers can focus on writing code without dealing with server provisioning, resulting in faster development cycles and quicker time-to-market.** | **State Management: Managing application state across stateless functions can be complex and may require additional services.** |

# Serverless vs. traditional server-based approaches

| Component | Serverless Approach | Traditional Server-Based Approach |
|---|---|---|
| Resource Management | Serverless abstracts server management, so developers do not have to worry about provisioning, configuring, or maintaining servers. | Traditional server-based approaches require provisioning, configuring, and managing physical or virtual servers, which can be resource-intensive. |
| Cost Model | Serverless is typically billed based on usage, so you only pay for the actual compute resources consumed during the execution of functions or applications. | Costs are incurred for servers and infrastructure, regardless of whether they are actively serving requests. This can lead to higher operational costs. |
| Scalability | Serverless platforms automatically scale based on demand. Resources are allocated or deallocated as needed, ensuring efficient resource utilization. | Scaling in traditional approaches can be more complex and may involve manual intervention, potentially leading to slower response times during workload fluctuations. |
| Event-Driven | Serverless is inherently event-driven, making it suitable for applications that respond to specific events like HTTP requests, database changes, or file uploads. | - |
| Developer Focus | Developers can focus on writing code and building features rather than dealing with server-related tasks. | - |
| Maintenance | - | Server maintenance tasks, such as applying security patches and updates, are the responsibility of the organization. |
| Resource Utilization | - | Resource utilization can be less efficient, with servers often being underutilized, especially during periods of low traffic. |

# REFERENCES

[1] Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uță, A., & Iosup, A. (2018). Serverless is more: From paas to present cloud computing. IEEE Internet Computing, 22(5), 8-17.

[2] Nguyen, D. (2019). Serverless Architecture on AWS.

[3] Sewak, M., & Singh, S. (2018, April). Winning in the era of serverless computing and function as a service.

[4] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., ... & Suter, P. (2017). Serverless computing: Current trends and open problems. Research advances in cloud computing, 1-20.

[5] Wen, J., Chen, Z., Jin, X., & Liu, X. (2023). Rise of the planet of serverless computing: A systematic review. ACM Transactions on Software Engineering and Methodology.

IT20147396
Peiris B.M.G

# Cloud Computing Integration

**Group ID = CC - 006**

# 1. Understanding Cloud Computing Models (IaaS, PaaS, SaaS)

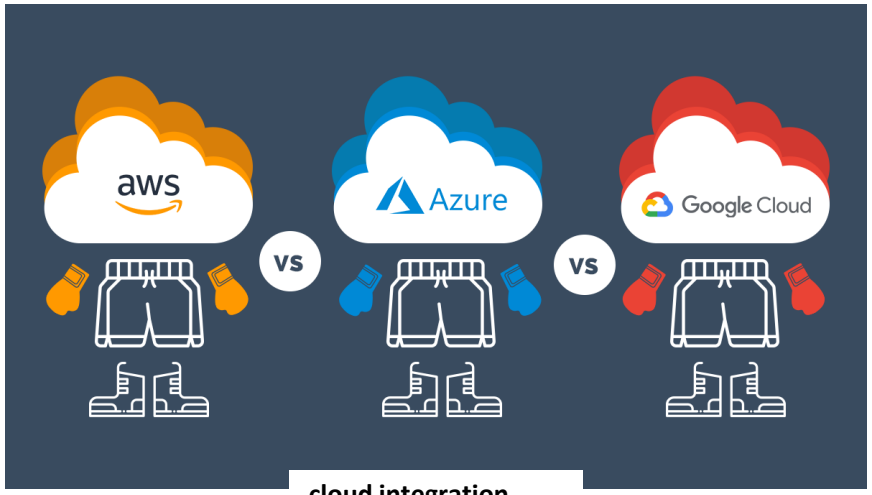
| Cloud Computing Models | | |
|---|---|---|
| IAAS | PAAS | SAAS |
| IaaS provides virtualized computing resources over the internet. Users can rent virtual machines, storage, and networking on a pay-as-you-go basis. | PaaS offers a platform and environment for developers to build, deploy, and manage applications without worrying about the underlying infrastructure. | SaaS delivers software applications over the internet on a subscription basis. |
| This model is suitable for those who want more control over their infrastructure. | It simplifies application development. | Users can access software through web browsers, eliminating the need for local installations. |

## 2. Integration of Microservices and Serverless with Cloud Providers

•Microservices and serverless architectures are designed to work seamlessly with cloud providers like AWS, Azure, and Google Cloud.

•Cloud providers offer specialized services for microservices, such as container orchestration (e.g., Kubernetes) and serverless computing platforms (e.g., AWS Lambda, Azure Functions).

•Integration involves deploying microservices and serverless functions on cloud infrastructure, leveraging cloud-native tools and services for scalability, monitoring, and management.




**cloud integration**

**Reference 2,3**

## 3. BENEFITS OF USING CLOUD SERVICES FOR APPLICATION MODERNIZATION:

| | |
|---|---|
| **Scalability** | Cloud services provide on-demand scaling, allowing applications to handle variable workloads efficiently. |
| **Cost-Efficiency** | Pay-as-you-go pricing models reduce upfront infrastructure costs and optimize spending. |
| **Flexibility** | Cloud platforms offer a wide range of services, enabling developers to choose the best-fit solutions. |
| **Global Reach** | Cloud providers have data centers worldwide, ensuring low-latency access for global users. |
| **Automated Management** | Cloud services often include automated management tools for tasks like patching and backups. |

# 4. CLOUD SECURITY CONSIDERATIONS:

| | |
|---|---|
| **Identity and Access Management (IAM)** | Implementing robust IAM policies to control user access and permissions. |
| **Data Encryption** | Encrypting data at rest and in transit to protect against breaches. |
| **Compliance** | Ensuring compliance with industry standards (e.g., GDPR, HIPAA) and best practices. |
| **Security Monitoring** | Leveraging cloud-native security monitoring and logging tools. |
| **Incident Response** | Developing incident response plans and conducting regular security audits. |

# 5. DEVOPS PRACTICES IN A CLOUD-NATIVE ENVIRONMENT:

**Collaboration:**

Promoting collaboration between development and operations teams to facilitate faster development cycles.

**Continuous Integration/Continuous Deployment (CI/CD):**

Automating application testing and deployment pipelines.

**Infrastructure as Code (IaC):**

Treating infrastructure configuration as code to ensure consistency and version control.

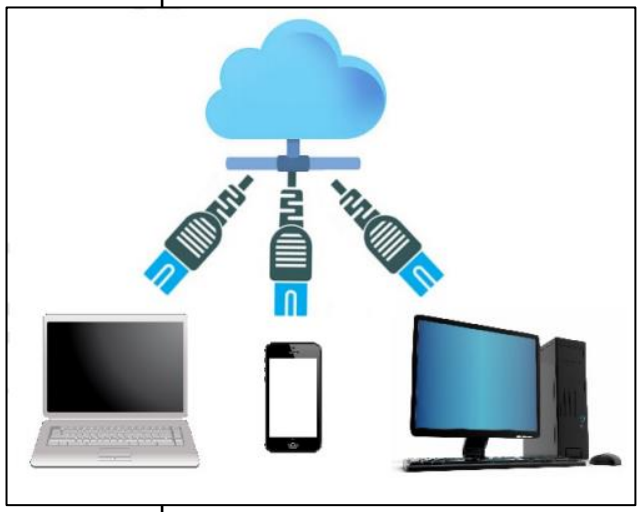
**Microservices Orchestration:**

Coordinating microservices deployment and scaling in a dynamic cloud environment.

**Monitoring and Logging:**

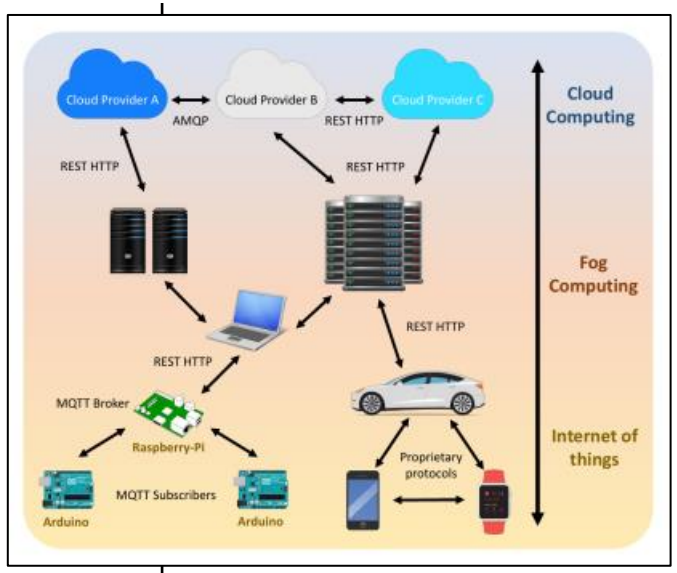
Using cloud-native monitoring tools to gain visibility into application performance.

# Cloud Integration



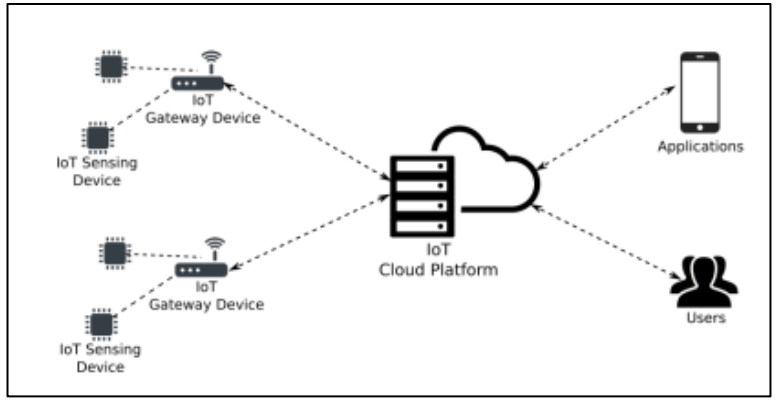
Example 1: cloud integration

Reference 1



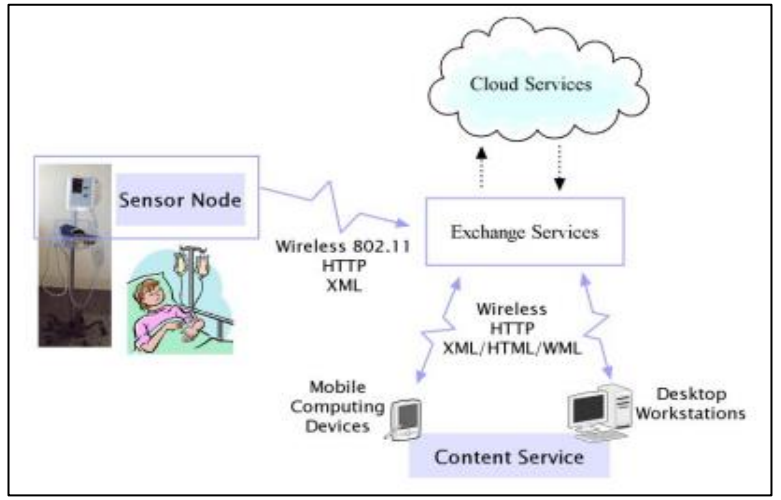

Example 2: cloud integration

Reference 2




Example 3: cloud integration

Reference 3




Example 4: cloud integration

Reference 4

# REFERENCES


[1]Adhitya Bhawiyuga*1, D. P. (June 2019). *Architectural design of IoT-cloud computing.*

[2]Carlos Oberdan Rolim, F. L. (2016). *A Cloud Computing Solution for Patient's Data Collection in Health Care .*

[3]Christos Stergiou, K. E.-G. (28 November 2016). *Secure integration of IoT and Cloud Computing.*

[4]JASENKA DIZDAREVI**Ć**, F. C. (19 Feb 2019). *A Survey of Communication Protocols for Internet of Things.*

[5]Sara Rodríguez, D. I. ( 16 Nov 2017). *Cloud Computing Integrated into Service-Oriented.*

IT18097870
A.L.A.A.M Ambegoda

# Cloud Services

**Group ID = CC - 006**

# 1.Overview of major Cloud providers

- Amazon Web Services(AWS) : Offers a vast range of services including computing ,storage ,databases, and machine learning.
- Microsoft Azure : Offers services for virtual machines ,app services, and AI.
- Google Cloud : Google's cloud offering with a strong focus on data analytics and machine learning.

## 2. Cloud Service Categories

| Storage as a Service | Database as a Service | Function as a Service |
|---|---|---|
| 1.Provides scalable and accessible data storage solutions. | Offers managed database solutions that eliminate the need for database administration tasks. | Enables the execution of code in response to events without managing server infrastructure, |
| 2.Ideal for data backup,file sharing,and content delivery. | Example providers:Amazon RDS,Azure SQL Database,Firebase. | Ideal for building microservices and event-driven applications. |

## 3.Choosing the right cloud service for your application

**1.Understand applications' requirements**

**2.Define budget**

**3.Consider the type of application**

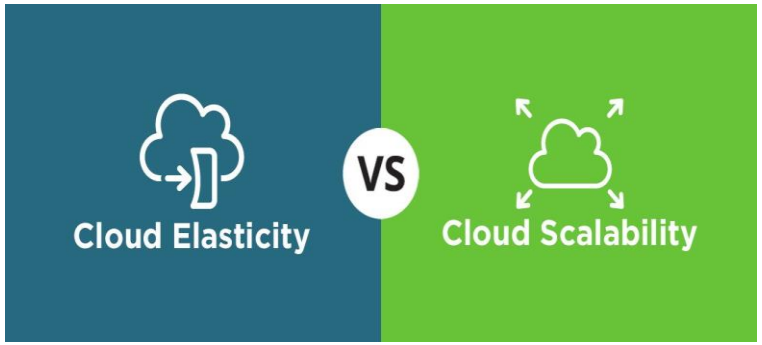
**4.Data Management**

**5.Security and Compliance**

# 4. SCALABILITY AND ELASTICITY IN CLOUD

| Scalability | Elasticity |
|---|---|
| Focus on long-term growth | Helps with sudden influx of workload |
| Mostly used by huge brands and companies to deal with workflow demands | Famously used in smaller businesses to deal with influx |
| A one-time investment that saves long-term expenditure | Helps prevent disasters in the server |
| Pre-planed notion uses by growing business | Used as per workflow demands in the industry |





# 5. SERVERLESS SERVICES

1.Amazon API Gateway : Provides a fully managed service for creating and publishing RESTful APIs.

2. Azure logic Apps : A workflow automation platform that enables you to create serverless workflows by connecting various Azure and third-party services.

3. Google Cloud Functions : Google Cloud Platform's serverless compute service runs code in response to events from various GCP services like Cloud Storage, Cloud Pub/Sub, and Firestore.

# REFERENCES

[1]Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared. Grid Computing Environments Workshop, 1-10. DOI: 10.1109/GCE.2008.4738445

[2]T. Author and J. Author, "Scalability and Performance of Cloud Services," IEEE Transactions on Cloud Computing, vol. 3, no. 2, pp. 94-102, May 2019.

[3]A. Researcher and B. Scholar, "Security Considerations in Cloud Services," in Proceedings of the IEEE International Conference on Cloud Computing, San Francisco, CA, July 2020, pp. 175-182.

[4][4] "Amazon Web Services," AWS, https://aws.amazon.com/, accessed Oct. 15, 2023.

[5] National Institute of Standards and Technology. (2011, September). The NIST Definition of Cloud Computing (Special Publication 800-145). https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Dilshan P.A.D.S.D
IT20178154

# Real World Scenario

**Group ID = CC - 006**

# NETFLIX

- Netflix is the world's leading internet television network.
- Founded in 1997
- Website launched in 1998.
- The company operated strictly as a subscription-based, mail-order DVD service until 2007.
- Introduced online streaming in 2008.
- 231 million members in more than 190 countries enjoying 125 million hours of TV shows and movies.

# WHY?

Netflix needed to find a method to store all its data due to its large user base, and a typical in-house data center was rapidly becoming too inefficient. They required scalability in their infrastructure.


# CHOICES

Recruit world-class data center operations build team and build new datacenters.

OR

Use the Elastic Compute Service of AWS

# AND THEY CHOSE THE AWS

NETFLIX

Netflix on AWS

## CHALLENGES

### Competition

- Amazon Prime is another online streaming service that uses AWS
- Understand how AWS separated from Amazon Prime

### Capacity

- Experiments to see what worked

amazon Prime

### Business

- Had to sign up for an Enterprise License Agreement

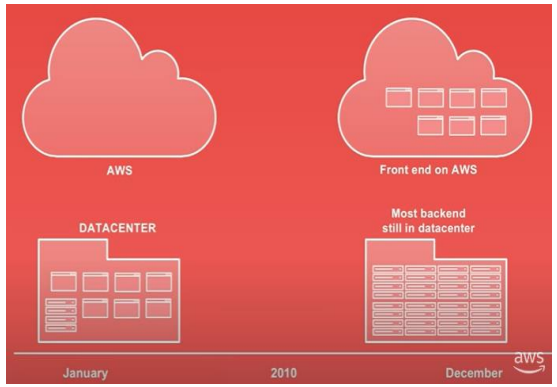
### Publicity

- New York Times story about Netflix and AWS in 2010

### Migrating to Cloud

- Start of 2010 Netflix decided not to build any more data centers and started to move to AWS before the end of 2010
- They start with the simplest API service
- Next the simplest web page
- Then pages and API one by one

### December 2010,



# ARCHITECTURAL DECISIONS AND TECHNOLOGY STACK

Netflix experienced a major outage in 2008 when they used Monolithic architecture

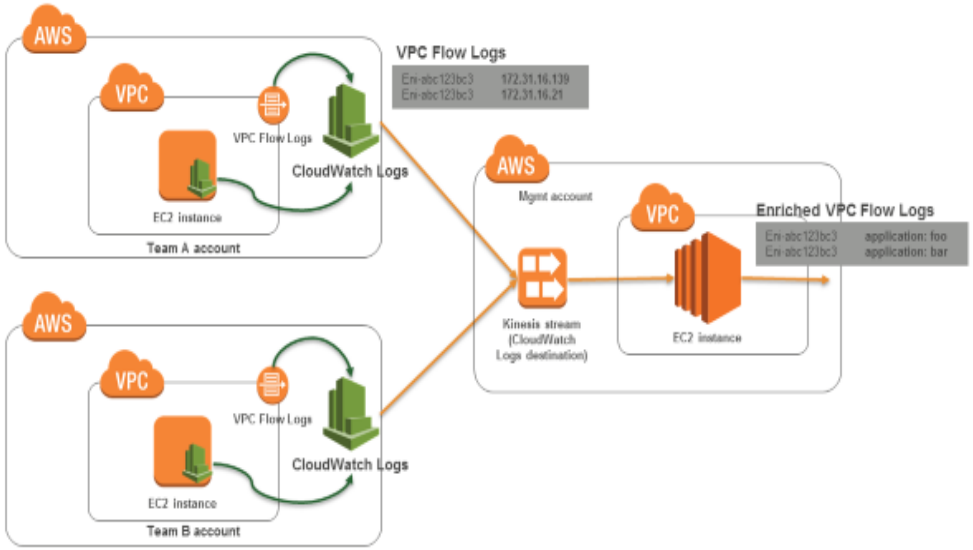
Now Netflix Uses Microservice Architecture on AWS

Now they have more than 1000 microservices each control a specific aspect of colossal Netflix operation.

Netflix uses,

- Hosting ---------- AWS EC2
- Storage ---------- AWS S3
- Content Delivery- Cloudfront
- Database---------- Relational Database Service (RDS), DynamoDB

Netflix cloud database followed a pay-as-you-go basis, which helped them save cost whenever they rolled out the AI-based feature called top personalized recommendation.




amazon web services | Professional Services

# BENEFITS AND OUTCOMES

### Availability of Service

After the move to the cloud, Netflix got rid of SQL nightmares and has been available to customers 99.99% of the time.

### Scalability

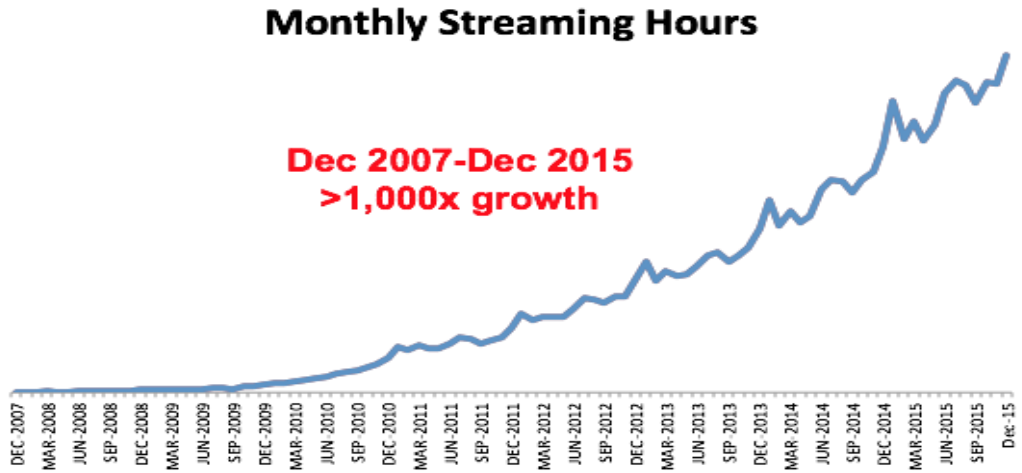
Netflix streaming service has increased tremendously as sometimes streaming demand increases by 50%

### Cost Effectiveness

The cost per streaming price started to go down in 2011 and it is even now half of the price when the company was using its data centers.

## Outcomes

- Eight times as many streaming members than 2008.
- Overall viewing grows.
- Netflix welcomed an additional 1.75 million subscribers and earned $1.31 in profit on $8.16 billion in revenue



# LESSONS LEARNED FROM NETFLIX MODERNIZING

- It’s not about cost saving
- Availability
- Cloud Native
- Changing management resources
- Reorganization of application architecture
- Changing IT Organization

NETFLIX

Netflix on AWS

## Tips for Others

I. Define objectives and motivations for migrating to the cloud.
II. Select a cloud provider that aligns with requirements and goal
III. Design application and infrastructure for redundancy and resilience
IV. Prioritize  security
V. Invest in training and upskilling your team
VI. Maintain clear and comprehensive documentation.
VII. Stay updated with industry trends.

# REFERENCES


1. Leveraging the Revolutionary Paradigm of Cloud Computing: The Case of Netflix  Dr. Harman Preet Singh* and Dr. Anurag Agarwal**

1. https://www.mygreatlearning.com/blog/top-companies-that-succeeded-with-cloud-computing/

1. https://aws.amazon.com/solutions/case-studies/innovators/netflix/

1. https://www.cloudsine.tech/news-trends/case-study-how-netflix-uses-cloud-for-innovation-agility-and-scalability/

1. https://www.cloudzero.com/blog/netflix-aws

1. https://about.netflix.com/en/news/completing-the-netflix-cloud-migration

# THANK YOU!