International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Exploration of Utilizing Cloud Computing with Microservices Architecture

Mohanraju Muppala

Email: *mohanraju.m[at]outlook.com*

Abstract: ***Cloud Computing is a flexible platform allowing you to deploy and scale your applications with more ease than you would have on your own hardware. It allows virtual hosting with no need to get heavy - duty machinery. Small applications, actions that need to be performed from time to time, such as sending out email messages or generating thumbnails can now be handed over to virtual machines on a pay - as - you - go basis, there to be activated and deactivated on demand. Compared to traditional hosting where you need to rent your own machine and set everything up yourself, the cost of using cloud hosting is usually much smaller. Normally, you would need to set your application with a sufficient level of demand for capacity that you would keep a machine running for a whole month. With cloud computing, servers can ramp up or down or be turned off entirely at low demand times.***

Keywords: Cloud Computing, Microservices Architecture, Scalability, Containerization, DevOps, Continuous Integration / Continuous Deployment (CI/CD)

1. Introduction

Indeed, it is one of the characteristics of cloud computing; scalability which offers the opportunity to handle fewer tasks at low - cost time and more tasks at high - cost time. Let everyone just pay for what they utilize at any given moment in time. The domain of microservices architecture is a new thing called microservices architecture. Rather than creating a single monolithic application, microservices would instead have several different small applications that all work together. Each application would fulfill its own single role while being part of a more significant contribution when working together. Microservice - based architecture is nowadays considered to be necessary in all large pieces of software due to the increasing pace of development [1].

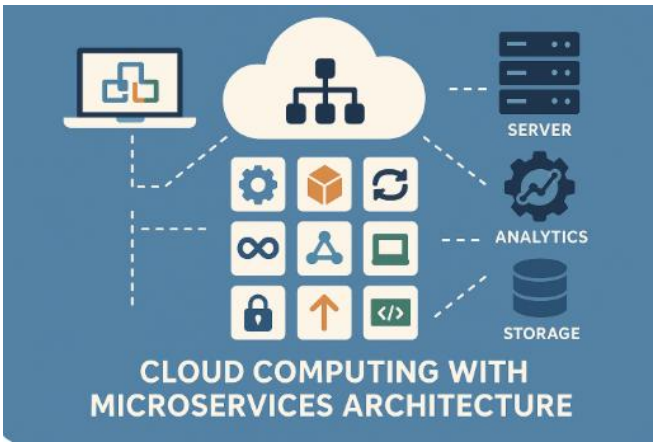

Figure 1

2. Understanding Cloud Computing

Definition and Characteristics Cloud computing has been a frequently discussed topic in the current area of information technology and development. It is increasingly used across global businesses and organizations. However, what is meant by cloud computing? In simple words, cloud computing is a computing technology that uses the internet and the web to store, manage, and process data. The cloud allows people to access information at any time and from anywhere. In addition, cloud computing provides a variety of ways regarding how a company or organization can meet their computing needs. The cloud will give everyone's services to be offered and helps to have easy network access. Cloud computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models. Since its inception, cloud computing technology requires the systems to be connected to the Internet 24 hours a day without interruption. Cloud computing has offered very significant and fundamental effects on the world of computing. It has torn down the traditional computing architecture, in which a single computer is used for computing, processing, managing, and storing data, and the small computer connected to the Internet is used to access remote information that requires a significant number of resources, such as software, hardware, and computing resources. In general, cloud computing is meant to offer services with good quality, use easy, services are behind pay matters, and very cheap to support.

2.1 Definition and Characteristics

Cloud computing may be defined as a model that enables convenient, on - demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing is composed of two primary components, which are models and service models. The cloud computing service model is based on virtualization and service concepts.

2.2 Types of Cloud Services

Internet addresses on personal computers, businesses, and organizations allow access to cloud services. Thus, two types of clients use cloud services in different models: the

cloud service consumers, who pay a cloud service provider to use its services on - demand; and the cloud service providers, who create infrastructures to propose and deliver cloud services.

There are three service models of cloud computing: Infrastructure as a Service, Platform as a Service, and Software as a Service. With the Infrastructure as a Service model, the service provider delivers to clients a physical or virtual infrastructure, which consists of servers, storage devices, networking equipment, and/or a proper environment for them to connect on the Internet. These resources must be configured, managed, and maintained by the clients. On the other hand, the Platform as a Service model adds to the Infrastructure as a Service model a set of instruments allowing the deployment and management of services and applications over the Infrastructure as a Service infrastructure. With this model, the cloud service provider is responsible for the management and maintenance of the infrastructure and tools.

2.3 Benefits of Cloud Computing

Organizations utilize cloud services for several reasons. They do so to reduce costs, improve time to market, leverage organizational skills, and reduce risks. In today's global environment, organizations use technology just like any other resource. They take only the quantity they need at a specific point in time. If demand goes up, they can consume more, and if demand drops off, they can consume less. Because the technology cost is mostly variable with commercial clouds, the business case around design is different. When technology is mostly fixed cost, the organization optimizes the design.

Mappings are done as part of the architecture - phase activities. Network infrastructure, computing resources, and storage are mapped to the different computing elements in the design. Mappings for Commercial Clouds introduce another layer of consideration to design decisions [2]. For example, if the cost of incoming and outgoing bandwidth to the cloud is expensive, the architect may need to rethink design decisions that result in moving a lot of data in and out of the cloud. A commercial cloud vendor provides a catalog of infrastructure services, but on - demand configuration and provisioning increase the overhead for the application architect and implementation engineer. The organization is incorporating reductions in design and development time, maintenance, and operational resources to justify building new applications on these commercial clouds rather than traditional data centers [2].

The vendor - branded services also reduce the effort associated with the architecture and implementation phases because they provide not just a picture but a pre - designed piece of infrastructure service to use. Organizations are finding it easier to quickly build applications on commercial clouds. The appeal of less upfront money, fixed costs, and the unlimited capacity that commercial clouds provide takes a lot of the risk of building a new application for the organization out.

3. Overview of Microservices Architecture

Definition and Principles: Historically, most applications have been created in a monolithic structure, with all elements and services in one large codebase with an interdependent structure. This has served well for large organizations, with staff resources bringing in efficiencies through specialization and risk mitigation through segmentation. But the sequential and interdependent nature of monolithic systems creates challenges for fast - moving companies where agility is paramount. Businesses are pressured to produce new features more rapidly than could be delivered through the monolithic structure. As applications have grown larger, monolithic architecture has become increasingly difficult to manage and maintain. These difficulties have caused a shift away from the traditional monolithic architecture model for application development, favoring a microservices architecture design that utilizes component - based technology.

Microservices architecture (MSA) is based on the creation of independently deployable mini applications that work independently but are integrated into a greater, interdependent whole. MSA is based on the following design principles: componentization via services; organized around business capabilities; products, not projects; smart endpoints and dumb pipes; decentralized governance; decentralized data management [4].

The fundamental concept behind MSA is that a large application is broken down into small services that are loosely coupled with each other and implemented by small and autonomous teams. Business capabilities are independently deployable and focus on a business goal that creates direct business value. Each microservice can implement a cluster of related functions that are multilingual and decoupled.

The Architectural Perspective
Those waves' impact has been reflected in how microservice applications have evolved from an architectural perspective. Figure 2 illustrates four generations of microservice architecture.

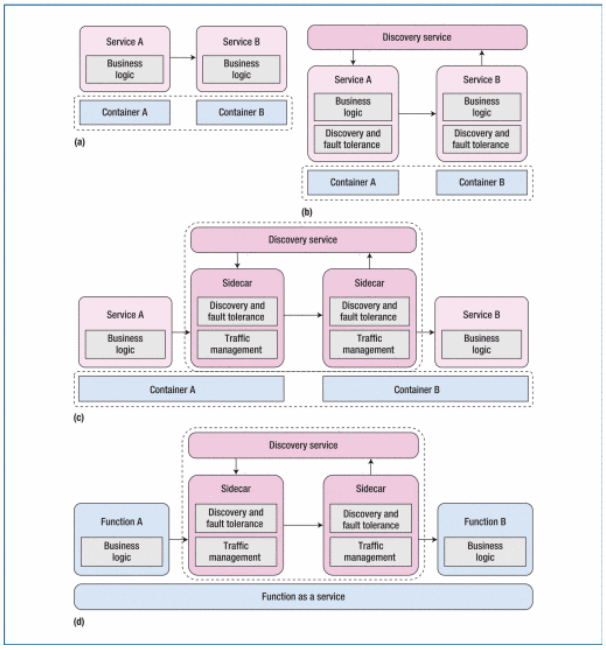

Figure 2

3.2. Microservices vs. Monolithic Architecture

To understand how microservices are a better option compared to monolithic designs; it is important to examine the challenges of monolithic structures. Traditional monolithic architecture keeps the backend services of multiple functions or processes tightly coupled within a single code package, while microservices - oriented structure centers function independently operable and manageable components. Three significant differences between monolithic and microservice architectures should be noted in how they operate.

3.3 Definition and Principles

Microservices are a software development architecture with a variety of specialized services that are distinct building blocks utilized collectively to help organizations achieve their goals. Each service is devised with a narrow mission, as opposed to monolithic architectures that utilize a single, large service that is multifunctional. With microservices, the services execute their tasks, and they communicate with other services over frequently used HTTP REST or asynchronous messaging patterns. Microservices are well - matched for organizations with aggrandized software volume and growth objectives. The key principle is thus that the envisioned software product is systematically created with extensive collections or suites of software components that interact over well - defined interfaces and protocols. Any and all software capable of accomplishing the goal can be developed and concurrently deployed, frequently written in divergent programming languages, utilizing different frameworks, deployed on diverse operating systems and hardware individually from other components [4, 5].

The microservices architecture, employed on cloud computing platforms, followed the monolithic architecture and the service - oriented architecture. Monolithic architecture deploys applications as a single service while SOA creates larger services given the advantages accrued from dividing business logic into multiple smaller services with well - defined interfaces. However, SOA would only provide backward compatibility with concepts and premises that resulted in large overhead ratios. In response, cloud computing offered an alternative infrastructure and platform model to avoid the high deployment costs while microservices provided an architecture, unlike the monolithic solution, that fully exploits the microservices strengths and decouples the specialized services with low - cost communication resources that could easily interact to create a responsive software solution with components that could be flexibly connected.

3.4 Microservices vs. Monolithic Architecture

For example, a large e - commerce application may use a monolithic architecture where the product catalog, shopping cart, checkout and payment processes, and order fulfilment subsystems are connected tightly into one single application. If any single part of the application faces a spike in traffic and needs to be scaled, you need to scale the whole single application on several large application servers, even if other sub - components are not under load. Whenever you need to make a change to any of the components, such as changing the way you search for products, you need to deploy the entire application. Also, if a change you made contains a bug, it may bring down the entire application affecting every function. Furthermore, scaling out to many application servers is expensive, because each application server uses larger amounts of expensive memory and CPU to run the massive single application.

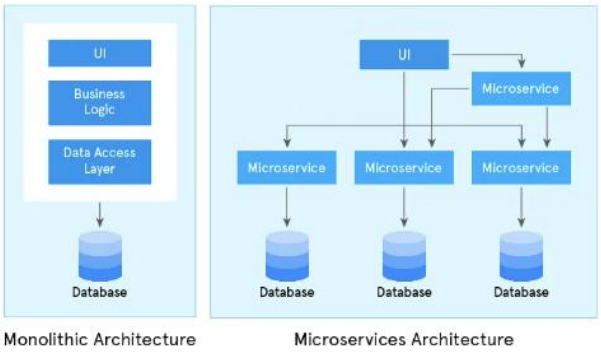

Figure 3

3.5 Advantages of Microservices

Microservices architecture is a technology formed by small services or applications using independent technologies that work together based on a centralized management system using clear and lightweight mechanisms for such cooperation. Essentially, microservices decompose each layer of the software application, presenting the advantages of distributed systems to this architecture. Each service is a development cycle that includes build, test, release, and deploy, which can independently scale, create fault isolation, and establish clear location transparency [6, 7] Moreover, each microservice is a software component that is created based on certain requirements and business concerns related to the qualified, resilience, and security participants' domain.

4. Integration of Cloud Computing and Microservices

We carefully studied these two technologies separately, but that is not enough. In our understanding, microservices were designed from scratch, considering the latest advances in practice and theory in distributed systems, cloud computing, and virtualization. Cloud Computing provides not only an execution environment for Microservices but also valuable services that are in the core of Microservices design – service discovery, logging management, message - oriented middleware and orchestration. We analyze the symbiosis of Cloud Computing and Microservices.

Microservices designed for a PaaS environment could also be deployed in an IaaS environment. However, this alternative has several burdens, such as the cost of managing many instances, the use of virtualization overhead and the high cost of infrastructure utilization with resulting load spikes [8, 9]. Enterprise Clouds designed for supporting many different enterprise information systems could be leveraged as an alternative to the use of PaaS. Either way, Vertical Microservices designed for a PaaS solution decrease design and management costs. Deploying these Microservices in a private or hybrid Cloud can be a good solution. Several enterprise Clouds have been designed to accommodate a large number of Microservices, supporting Container techniques and solutions.

4.1 Deployment Strategies

In the monolithic style of development, the entire code needs to be deployed together as a single entity. In case of minor code changes in one of the services, complete software is required to be redeployed. In agile methodology of development, multiple teams need to work on different functionalities of the web application, and code changes could occur frequently in a matter of days, therefore, such a long cycle time for code deployment is prone to lead to missing out on project deadline. However, in microservices, the services can be deployed independently in different containers encapsulating their dependencies. Microservices deployed in cloud environments can effectively support high - throughput, real - time data analytics such as tensor decomposition for atmospheric modeling and environmental monitoring [3]. Code changes on one service will result in the redeployment of only that service without affecting the other services. Small teams can also independently develop and deploy their assigned microservices, which leads to code deployment across the services to occur frequently. Such continuous delivery practice leads to getting the code changes to production and available to users as soon as possible therefore resulting in higher customer satisfaction.

4.2 Scalability and Flexibility

Microservices processes individual components of a larger application as separate services, each operating alongside one another to create the larger application. The cloud is the ideal environment for this kind of architecture due to its scalability and flexibility features. The cloud has a virtually limitless capacity for adding or subtracting resources as needed. This becomes useful when a company is subjected to unpredictable loads. For example, a news website that only has huge traffic spikes when major news events occur, such as natural disasters, not just the day - to - day traffic a website gets. The microservices architecture allows organizations to scale applications automatically with just a few changes to code. Containers combined with Kubernetes help set rules to adjust the application services as needed while switching from a private data center to the cloud. Due to the cloud's virtually limitless supply of resources, microservices help many businesses and start to evolve its infrastructure. Therefore, the cloud must decide how to meet those needs [10, 11].

4.3 Cost Efficiency

Cost efficiency is one of the main things that companies consider about which determines whether to adopt microservices architecture and development. Cloud providers offer different pricing models that you must choose carefully according to your business workload. For example, serverless architecture can be the best for microservices workloads that have unpredictable traffic because you are charged per execution, making it possibly cheaper. Serverless is not suitable for workloads that have to call pretty much the same service all the time. This microservice also needs to store state due to how business logic for this microservice is implemented, which means that you will have to suffer from the latency hit. In essence serverless architecture can have higher cost if traffic is stable for multiple hours during the day.

To achieve the best cost efficiency, combine multiple cloud services to satisfy your current needs and business growth. Do not solely rely on one provider for all your cloud services. Do not use just serverless architecture for all microservices in your business. You can distribute different services to the different cloud providers. Create hybrid cloud architecture depending on your business requirements. Use container - based architecture for services that continuously run in each time window strata. For example, if you experience a big change in your resources over the year, you may prefer some combination of pricing models and instances of your preferred cloud provider instead of serverless architecture. If your business is really sensitive to costs, you can build your own on - prem microservices architecture and also lean towards barebone do it on yourself type cloud providers instead of serverless services.

5. Challenges in Implementing Microservices in the Cloud

There are challenges in implementing microservices in the cloud environment. First, to leverage the cloud features and make the operations smooth, the Physical Management task is mostly automated. Multi - tenancy and lack of control on the hardware result in the complexity of managing physical infrastructure. Therefore in microservices, to cope up with Failure of hardware, loss of Service, and Security, one needs to add more function and layers to manage the physical layer. Second, unlike Monolithic where all are designed and implemented in a single architectural process, here each microservice must go through its own release cycles and needs to appear in the business with all the data consistency.

Mostly the business critical as well as non - business critical data lies in storage along with but managed by its own microservice.

5.1 Complexity of Management

The microservices architecture abstracts and decouples application development into multiple distributed independent services. Microservices are mostly fine - grained, composing a business function, and requests are done through APIs. Therefore, implementing an application with microservices consists of creating several microservices and enabling services to communicate with each other. However, for providing an application as a service for users, in addition to implementing services, enterprise - scale functions are necessary such as service registration, discovery, API gateway for service routing, orchestration, load balancing, communication, logging, performance monitoring, tracing, security, authentication, access control, and fault tolerance.

Although cloud computing is a ready service solution enabling any function related to application hosting, the number of enterprise functions is significantly large. Therefore, when utilizing microservices in the cloud, it introduces even larger complexity in orchestrating infrastructure for providing the above - mentioned functionalities. In addition, as microservices wrap internal business functions, unplanned functional changes and bugs can negatively affect other dependent services. Any sudden impact to one dependent service from one microservice increases the impact scope for overall application service functions provided to users. Asymmetric failure of microservices can also lead to an abrupt impact on application services such as increases in latency and error. In the microservices architecture, there is no monolithic application to monitor for identifying bugs and optimizing performance or load balancing across servers hosting an application. Each part of the application process is co - hosted with the associated business functionalities of microservices on different servers, and distributed and inaccessible process monitoring can mislead identifying causes for solving application performance issues.

5.2 Data Consistency Issues

A plethora of data management and consistency propagation problems arise due to the distributed, concurrent, and failure - prone characteristics of microservice - based cloud applications. Data consistency problems are complex and challenging to handle regardless of the adopted technology. In particular, the exposed microservice interfaces, which may be wrapped in a controller for purposes like authentication, security, and fault tolerance, necessitate complex transaction and message - oriented event propagation programming for calls visiting multiple microservices. At the other extreme, each microservice may communicate exclusively through the event - based ISC with an underlying event broker and/or a message queue. The latter, though easier to manage, requires an even more strict application design and an event structure that supports optimal message routing and type filtering.

5.3 Security Concerns

The microservices architecture inherits the security drawbacks of conventional designs, such as denial of service and data security. When additional elements such as message brokers, admin panels, service broker, and dynamic service discovery are introduced into a web service, the vulnerabilities become more pronounced. In microservices, all services are interrelated; hence the security of each service is crucial. User authentication, perhaps the most important step in securing a microservices ecosystem, has its own challenges. Users are constantly changing, and they interact with services in small and unpredictable ways. When a user makes a request, how can a service be certain that only that user is allowed to interact with it at that exact moment? Resource servers need reassurance from authorization servers that users really are who they say they are; and authentication between resource servers can often require heavy and expensive data transfers over corporate connections.

6. Best Practices for Cloud - Based Microservices

Cloud - based microservices architectures can provide many advantages. However, if not designed or constructed properly, microservice applications can have performance issues or can be even more fragile than a traditional application. This section covers best practices for designing and implementing microservices that take full advantage of cloud computing, providing scalability and resiliency.

Service Design Principles Designing microservices is challenging and there are many considerations to keep in mind. Fortunately, the microservices community has established a number of general principles that have been shown to work well. Microservices are like cloud services as they solve a specific business problem for an external customer, are versioned, and are loosely coupled with other services. Unlike traditional cloud services, we build microservices using a system of systems approach. Microservices are then used together to build a microservice application.

6.1 Service Design Principles

Building cloud microservices is not only about following design and architectural patterns. It emphasizes that microservices are an investment by themselves requiring proper development practices, tools, and a degree of operational overhead and integration effort. Like any other services, microservices also involve careful thinking on how to build and expose the service interfaces that other services will utilize. Microservices should be granular enough to perform a specific task, and developers can choose the best tools to efficiently build the services. Combined, these services form the entire colleague of functionalities. An individual microservice can be best suited for a specific business goal or problem.

6.2 Monitoring and Logging

As microservices operate independently of one another, and communicate via APIs, this can create difficulties for management tools that are primarily used for managing large monolithic apps. Therefore, you need specialized tools for monitoring and logging microservices. In most cases, implementing logging in a microservices environment means you will have to use a distributed system log collector to aggregate, process, and store logs from multiple services and hosts. This can be done in a variety of ways, but one of the most common is to use tools that implement the storage and processing architecture of a distributed system. The processing feature handles data processing at massive scale. The ecosystem also includes tools for log collection, as well as for batching and event - time scheduling for processing. Interfaces also support plug - ins designed for batch processing, which is known for its rapid in - memory processing for computing on data.

6.3 Continuous Integration and Deployment

The goal of Continuous Integration (CI) is to provide rapid feedback, relieving the team from the time - consuming process of assessing the quality of integration on a daily or weekly basis. Automated testing is a key enabler of CI; no CI system would be accepted if it couldn't provide sufficient coverage and confidence. CI provides rapid feedback for code - level changes, but what about the refactoring of services, especially for microservices that are responsible for multiple business capabilities? Even with the best - designed systems, there will always be a temptation to do more than one thing in a service, services will always overlap with each other to a certain degree, and business changes will result in broken contracts. Hence it is imperative to have CI pipelines to verify that service refactoring's haven't broken anything. These pipelines should also provide a high level of developer productivity. Getting developer feedback quickly enough is critical to avoid long waves of broken build states and encourage developers to keep the process clean. Whenever possible, teams should use semantic versioning and put services in a "safe" suspect mode.

7. Case Studies

Cloud computing and Microservices technology give many advantages, like flexibility, scalability, and other advantages; many companies are migrating their traditional software to cloud Microservices architecture including banking, space, retail, hospitality, and other types of applications. This section presents several case studies which are implementing and utilizing the Cloud Microservices architecture, technology, and methods for the solution of business and technical tasks and give some recommendations.

In our first example, we present the case study of a bank, which includes an example of a successful case. The bank incurred big financial costs because of a code service outage. In general, the bank's digital services are using the technologies of Cloud, Artificial Intelligence, and other areas for many advantages like cost, security, flexibility, and others. The bank created many software services for Anti - Money Laundering and Passport Management. The services are implemented as Microservices, and the bank is running them on a Kubernetes engine and a cloud platform.

In the second example, we explore the case study of a company, which is utilizing best practices and expertise for business advantages. The company is where Cloud Microservices architecture and technologies are used, for example, a test automation platform. This test automation platform is widely used for automated functional tests for various desktop, mobile, and web applications. This case study gives us some advantages of Microservices, Cloud, and Microservices - based architecture. The main business and technical advantages are speed, efficiency, and flexibility of the solution utilizing Microservices compared to Monolith.

7.1 Successful Implementations

Cloud technologies have redefined existing businesses or enabled the creation of new businesses by lowering the cost structure, increasing the scale, and providing the speed to experiment and innovate. Some of the successful implementations of existing businesses are Netflix, Uber, Airbnb, and Spotify. There are also cloud - native businesses that are created using Cloud MIME services, such as Instagram.

7.2 Lessons Learned from Failures

The cloud is a commodity that provides you with an infrastructure of servers. Running a microservice over cloud infrastructure makes the microservices architecture more scalable, reliable, and manageable. However, managing a distributed application is not straightforward. A failed deployment or service can cause downtime. There are many problems to solve when microservices run over the cloud. In this section, we discuss lessons learned from large - scale and wide adoption failures from enterprises.

8. Future Trends in Cloud Computing and Microservices

Cloud computing and microservices are still young and innovative technologies in computing. Future advancements in microservices - based cloud computing are increasingly influenced by the integration of cognitive AI systems, neuro - symbolic reasoning, and human - centric intelligence frameworks [20]. However, almost 30 years of cloud computing and 10 years of microservices have already changed how business is being modeled and approached in both development and utilization areas. Cloud enables empowering services developed in traditional, event - driven, service - oriented, microservices, and so on models to be deployed in an on - demand platform that enables easy scaling, reliable backup, disaster recovery, high availability, and other traditional enterprise - grade services/products being offered to the market, but these at a low - cost, pay - per - usage price. Microservices make enabling that deployment simpler and more productive from a business perspective. Besides being simple, small, and isolated, microservices are based on RESTful communications

relying on cloud APIs. Also, the usage of containerization enables portable deployment processes through Integrated Development Environment for microservices, Continuous Integration, and Continuous Development Release pipelines enabled by the Kubernetes orchestration tool.

Besides these benefits gained in the organization, business, and technical areas, both technologies are still evolving and adapting to Enterprise Architecture as a Service engaging technology in both areas. Emerging technologies such as Blockchain, Quantum Computing, Internet of Things, Artificial Intelligence, and Machine Learning teach that vision and experiences in engaging both areas for Business as a Service also show that. Experienced engaged problems into operational demands that BaaS operates in. For the next decade and further, we can forecast (i) cloud to support and engage on themselves cycle - and disaster - driven business and operations; (ii) cloud to engage emerging technologies and microservices further, teaching upcoming solutions; (iii) platform embedded in code; and (iv) business and specialized integrated BaaS.

8.1 Emerging Technologies

Cloud computing has emerged as a new area of Information Technology (IT) and Business and is transforming the way organizations are delivering IT Services and managing Business Processes. Cloud Computing is gaining popularity due to its cost - efficient model, which utilizes Resource Sharing and On - Demand Service Availability over Internet for required resources by Organizations. Cloud Computing is a Cooperative Platform for On - Demand Access for Shared Pools of System Resources and Services that are Configured to Reduce Management Overhead and Maximize Resource Utilization and Efficiency. Different Models of Service Delivery and Deployment Availability in Cloud Computing such as Infrastructure as a Service, Platform as a Service, Software as Service, and Business Process as a Service. Cloud Computing gives different advantages to users and their business in enabling costs, flexibility, reliability, availability, and scalability. Cloud Computing is further advanced with Virtualization, Distributed Computing, Grid Computing, Parallel Computing, Utility Computing, and Semantic web.

8.2 Predictions for the Next Decade

Since the Cloud Era began, we saw many new solutions that emerged to utilize the power of the cloud. At the end of the first decade of the Cloud, enterprises are all in the cloud and building new applications and services for their customers in the cloud. On a single cloud computing service, millions of customers use various services for computing, application hosting, storage, networking, databases, analytics, development, security, and enterprise applications.

The advancements in cloud technology, as well as the increasing migration of enterprise workloads and IT services to the cloud and their participation in the ecosystem, will help to drive increasing amounts of enterprise workloads and services onto the cloud. At the beginning of the next decade, cloud computing will shift from big data to smart data. New big - data capabilities will need the accompaniment of smart - data capabilities, as users increasingly look to analyze their cost - effectiveness in operations, as well as the overall cost - effectiveness of applications and data across their enterprise operations [21, 22, 23]. Users will look for domain - specific big - data service solutions that give them more flexibility and usability in their domain of interest than a general - purpose data service. As such, the data service will be pushed to the back plane [24]. All IT workloads and delivery models within the organization will then shift to using smart data to take advantage of other capabilities and efficiencies or services.

9. Conclusion

As a conclusion, this paper presents a complete exploration of utilizing cloud computing with Microservices Architecture. During our research, we present a background and rationale for the creation of Microservices by addressing the problems that arose due to the utilization of monolithic architectures as the basis for development of distributed systems over the past decade. Further, we address the workings of a Microservices Architecture by wrapping together the core principles and advantages for the utilization of distributed systems today with Microservices. Also, we did a deeper analysis of the Microservices technology stack by investigating each of the software platforms, development framework, containerization and orchestration, communication, database, and management tools required for the construction of a Microservices Architecture. Afterward, we present a case study that utilizes the Microservices Architecture along with a cloud service to develop a conference management system for biomedical research conferences.

References

[1] Chantry, M., Christensen, H., Dueben, P., & Palmer, T. (2021). Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft AI. *Philosophical Transactions of the Royal Society A*, *379* (2194), 20200083.
[2] Panda, Sibaram Prasad. “Securing 5G Critical Interfaces: A Zero Trust Approach for Next - Generation Network Resilience. ” IEEE, 2025.
[3] Shivadekar, S., Yesha, Y., Halem, M., & Yang, Z. (2023). Multi - variable tensor decomposition analytics. Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2023). https: //doi. org/10.1109/IGARSS52108.2023.10281918
[4] Jamshidi, Pooyan, et al. "Microservices: The journey so far and challenges ahead. " *IEEE Software* 35.3 (2018): 24 - 35.
[5] Shivadekar, Samit. (2025). Secure Multi - Tenant Architectures in Microsoft Fabric: A Zero - Trust Perspective.10.5281/zenodo.15700280.
[6] Panda, Sibaram Prasad. “Artificial Intelligence Across Borders: Transforming Industries Through Intelligent Innovation. ” Deep Science Publishing, Deep Science Publishing, 2025.

[7] Mahesh, B. (2020). Machine learning algorithms - a review. *International Journal of Science and Research (IJSR). [Internet]*, *9* (1), 381 - 386.
[8] Zhou, Z. H. (2021). *Machine learning*. Springer nature.
[9] Alpaydin, E. (2021). *Machine learning*. MIT press.
[10] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349* (6245), 255 - 260.
[11] Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol.1, No.9). New York: McGraw - hill.
[12] Hacker, P., Engel, A., & Mauer, M. (2023, June). Regulating ChatGPT and other large generative AI models. In *Proceedings of the 2023 ACM conference on fairness, accountability, and transparency* (pp.1112 - 1123).
[13] Irrgang, C., Boers, N., Sonnewald, M., Barnes, E. A., Kadow, C., Staneva, J., & Saynisch - Wagner, J. (2021). Towards neural Earth system modelling by integrating artificial intelligence in Earth system science. *Nature Machine Intelligence*, *3* (8), 667 - 674.
[14] Wu, T. Y., Majeed, A., & Kuo, K. N. (2010). An overview of the healthcare system in Taiwan. *London journal of primary care*, *3* (2), 115 - 119.
[15] Yenduri, G., Ramalingam, M., Selvi, G. C., Supriya, Y., Srivastava, G., Maddikunta, P. K. R.,. . . & Gadekallu, T. R. (2024). Gpt (generative pre - trained transformer) –a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*.
[16] Panda, Sibaram Prasad. "AI in Decision Support Systems. " International Journal of Advanced Multidisciplinary Research, Cases and Practices, 2021. doi: 10.5281/zenodo.15448128.
[17] Floridi, L. (2023). AI as agency without intelligence: On ChatGPT, large language models, and other generative models. *Philosophy & technology*, *36* (1), 15.
[18] Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT), 48* (3), 128 - 138.
[19] Panda, Sibaram Prasad. "Relational No SQL and Artificial Intelligence Integrated Database Architectures. " Deep Science Publishing., 2025.
[20] Rath, Tapan & Panda, Sibaram. (2025). Infrastructure as Code (IaC) Evolution - A Comparative Study of Terraform and AWS Cloud Formation. International Journal of Science and Research (IJSR).1476 - 1483.10.21275/SR25523162246.
[21] Shivadekar, S. (2025). Artificial Intelligence for Cognitive Systems: Deep Learning, Neuro - symbolic Integration, and Human - Centric Intelligence. Deep Science Publishing. https: //doi. org/10.70593/978 - 93 - 7185 - 611 - 9
[22] Shivadekar, Samit. (2025). AI for Climate Change and Sustainability.3.394 - 413.10.5281/zenodo.15716495.
[23] Thönes, Johannes. "Microservices. " *IEEE software* 32.1 (2015): 116 - 116.
[24] Dragoni, Nicola, et al. "Microservices: yesterday, today, and tomorrow. " *Present and ulterior software engineering* (2017): 195 - 216.
[25] Barnes, E. A., Toms, B., Hurrell, J. W., Ebert-Uphoff, I., Anderson, C., & Anderson, D. (2020). Indicator patterns of forced change learned by an artificial neural network. *Journal of Advances in Modeling Earth Systems*, *12* (9), e2020MS002195.