Journal of Advanced Research in Operating System Development and Evolution
Volume 1, Issue 1 - 2025, Pg. No. 1-8
Peer Reviewed Journal

Review Article

Advancements in Virtualization and Containerization: Redefining Resource Management and Scalability in Modern Operating Systems

Jass Abrahim

Student, Government Engineering College Idukki, Idukki Township, Kerala, India

I N F O

E-mail Id:
jassabrahim@gmail.com
Orcid Id:
https://orcid.org/0009-0009-5353-3971

How to cite this article:
Abrahim J. Advancements in Virtualization and Containerization: Redefining Resource Management and Scalability in Modern Operating Systems. *J Adv Res Oper Syst Dev Evol* 2025; 1(1): 1-8.

Date of Submission: 2025-01-29
Date of Acceptance: 2025-03-07

A B S T R A C T

The rapid evolution of operating systems (OS) has been significantly influenced by the growing need for scalability, flexibility, and efficient resource management. Virtualization and containerization have emerged as two transformative technologies that address these needs. This review article explores the advancements in virtualization and containerization, focusing on their impact on modern OS architecture, resource management strategies, and scalability. We examine how these technologies have evolved over time, their current state, and the challenges faced by OS developers in integrating and optimizing these approaches for contemporary computing environments.

Keywords: Virtualization, Containerization, Operating Systems

Introduction

Virtualization: Evolution and Advancements

Virtualization has revolutionized the way computing resources are managed, offering increased flexibility, scalability, and efficiency. Over time, virtualization technologies have significantly evolved, becoming more integrated into cloud infrastructures, improving resource management, and supporting a wide variety of workloads. Below are further details regarding the key advancements in virtualization:

Early Virtualization

The concept of virtualization can be traced back to the 1960s, primarily used in mainframe computing. Early virtualization systems, like IBM's CP-40 (Control Program), were designed to enable time-sharing, allowing multiple users to run programs simultaneously on a single machine. These systems aimed to optimize resource utilization, enabling more efficient use of the mainframe's computational power. Early virtual machines (VMs) were large and cumbersome, but they laid the foundation for the resource abstraction and isolation models we use today.[1]

- **Mainframe and Time-sharing:** The primary goal of early virtualization was to increase efficiency by allowing multiple users to share a single machine. This was achieved through the concept of virtual memory, where each user was provided an isolated and time-controlled environment.[2]
- **Hardware Abstraction:** Early virtual systems abstracted hardware to provide a consistent interface to the user while managing the underlying hardware resources efficiently.

Server Virtualization

- The rise of server virtualization in the 2000s brought a major transformation to data centers and enterprise computing. Technologies such as VMware, Xen, and later, KVM (Kernel-based Virtual Machine), introduced the ability to create multiple virtual machines (VMs) on a single physical machine. This revolutionized IT infrastructure, enabling organizations to run multiple operating systems on a single physical server.[3]

- **Virtual Machine Monitors (VMMs):** The introduction of hypervisors (VMMs) played a key role in enabling server virtualization. These hypervisors allow for the isolation and management of multiple VMs, each with its own operating system and applications, all running on the same physical host.[4]
- **Type 1 Hypervisors:** These run directly on the hardware (bare-metal hypervisors), such as VMware ESXi and Microsoft Hyper-V, providing better performance and efficiency.
- **Type 2 Hypervisors:** These run on top of an existing operating system (hosted hypervisors), like VMware Workstation or Oracle VirtualBox. While Type 2 hypervisors are less efficient than Type 1, they are more flexible and user-friendly.
- **Resource Consolidation and Efficiency:** Server virtualization allowed data centers to consolidate workloads by running multiple virtual machines on fewer physical servers. This reduced hardware requirements, lowered energy consumption, and increased operational efficiency.[5]

Hardware-Assisted Virtualization

- With the advent of hardware-assisted virtualization, virtualization performance became even more efficient. Intel VT-x (Intel Virtualization Technology) and AMD-V (AMD Virtualization) introduced special processor extensions that enable the hypervisor to more effectively manage and isolate VMs, improving performance and reducing overhead.
- **Hardware Support:** These technologies provided direct support from the CPU, allowing for faster context switching between virtual machines and reducing the need for complex software emulation. This advancement enabled more efficient virtualization of resource-intensive workloads.
- **Minimized Overhead:** Virtualization overhead, previously a limiting factor, was significantly reduced with hardware support. As a result, virtualization became a viable solution for a wider range of applications, including high-performance computing (HPC) workloads, enterprise applications, and cloud services.[6]
- **Extended Isolation:** Hardware-assisted virtualization also enhanced the security and isolation of virtual environments. Virtual machines could now run more independently from one another, with reduced risk of interference or security breaches between VMs on the same host.[7]

Virtualization in Cloud Computing

- The advent of cloud computing has been deeply intertwined with the advancements in virtualization. Virtualization serves as the backbone of most modern cloud infrastructures, enabling the dynamic allocation of resources, load balancing, and scalability across massive cloud environments.
- **Hypervisors in Cloud Environments:** Cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud rely on virtualization technologies to host and manage virtual machines in their data centers. Virtualization in the cloud allows for the flexible and on-demand allocation of computational resources, creating scalable environments that can quickly adapt to varying workloads.[8]
- **Elasticity and Resource Pooling:** Virtualization allows cloud providers to pool their physical resources and allocate them dynamically to meet fluctuating demand. Users can scale up or scale down the number of VMs they need based on real-time requirements, facilitating highly efficient and responsive cloud environments.
- **Virtual Machine Managers (VMMs):** Hypervisors and VMMs play a critical role in managing virtual machines in cloud computing. Solutions like VMware vSphere, OpenStack, and KVM help facilitate the management of resources and workloads within cloud environments, enabling features such as live migration, resource scheduling, and disaster recovery.
- **Cloud-Native Virtualization:** Virtualization has evolved in cloud-native architectures, where containers, which are a lightweight form of virtualization, have emerged as a preferred solution for deploying applications. However, virtualization still plays an important role in providing the necessary isolation and resource control for certain workloads in cloud environments.

The Role of Virtualization in Hybrid and Multi-Cloud Environments

- With the rise of multi-cloud and hybrid-cloud strategies, virtualization continues to play a critical role in facilitating seamless migration of workloads between different cloud providers and on-premise infrastructure.[9]
- **Workload Portability:** Virtualization enables workload portability across different cloud environments. VMs can be migrated from one data center or cloud to another, providing flexibility and resilience in multi-cloud strategies.
- **Consistency Across Clouds:** By virtualizing workloads and abstracting the underlying hardware, organizations can ensure that applications behave consistently,

regardless of the cloud provider or physical infrastructure.

Future Directions and Innovations in Virtualization

- While virtualization has made substantial advancements, the technology continues to evolve. Some of the future directions for virtualization technology include:
- **Integration with Containers:** Hybrid environments that combine virtualization with containerization are gaining popularity. This allows organizations to benefit from the strong isolation provided by VMs, while leveraging the efficiency and portability of containers.
- **Lightweight Hypervisors:** Future virtualization solutions may incorporate lightweight hypervisors that focus on minimal overhead, enabling better performance for modern workloads, including edge computing and IoT.
- **Serverless Computing:** Virtualization is expected to play a supporting role in the evolution of serverless computing, where developers can deploy applications without managing the underlying infrastructure. Virtual machines and containers are often the foundational layers that enable serverless platforms to run code in response to events.[10]
- Virtualization has undergone significant advancements over the decades, becoming a cornerstone of modern IT infrastructure. From early mainframe time-sharing systems to the cloud-native environments of today, virtualization has continually improved to meet the growing demand for scalability, flexibility, and efficient resource management. In combination with containerization and cloud technologies, virtualization remains a critical component of modern operating systems, data centers, and cloud infrastructures. As technology continues to evolve, virtualization will remain at the forefront of enabling the next generation of computing, offering increasingly powerful and flexible solutions for resource management, isolation, and deployment.

Containerization: The Rise of Lightweight Virtualization (Continued)

Containerization represents a paradigm shift from traditional virtualization by providing a more lightweight and efficient method for isolating applications. Containers are designed to package and run applications along with their dependencies in a standardized, portable environment, leveraging the underlying host OS kernel for efficiency. Unlike virtual machines, which require full hardware virtualization, containers are an abstraction at the application layer.

Concept of Containers

Containers enable the creation of isolated environments where applications can run without needing their own operating system. Instead of running an entire guest OS in a VM, containers share the host OS's kernel, allowing for much lower overhead and faster startup times.

- **Resource Efficiency:** Containers use fewer resources because they avoid the need for a full OS instance for each application. Instead, they share the host OS kernel while maintaining isolated user spaces for each container, resulting in faster performance and reduced resource consumption.
- **Portability:** Containers are highly portable. Since they include all dependencies required to run an application (libraries, binaries, and configuration files), containers can be easily moved across different systems, whether on a developer's machine or in a production environment. This portability is one of the key advantages of containerization.
- **Isolation:** Containers provide a lightweight form of isolation through mechanisms such as namespaces (which isolate resources like processes, networking, and file systems) and cgroups (which control the allocation of resources like CPU and memory). While they don't offer the same level of isolation as VMs, containers strike a balance between efficiency and security.[11]

Docker

Docker revolutionized containerization by providing a standardized platform for developing, shipping, and running applications in containers. Docker abstracts much of the complexity of containerization, allowing developers to easily create containerized applications and deploy them in various environments.

- **Simplified Containerization:** Docker provides a simple command-line interface and graphical tools that allow developers to quickly create, configure, and run containers. The Dockerfile syntax allows for easy customization of containers, defining the environment in which applications should run.
- **Docker Hub:** Docker also popularized the concept of a registry (Docker Hub) where users can share and distribute container images. This repository has become an essential part of the container ecosystem, enabling quick access to pre-configured containers.
- **Ecosystem:** Docker's widespread adoption has led to the creation of a large ecosystem of tools and services that support containerized environments, from orchestration tools to CI/CD pipelines. Docker's easy-to-use interface and consistent behavior across systems have made it the most popular containerization platform.

Kubernetes

As containerization became more prevalent, the need for managing large clusters of containers across multiple hosts became critical. Kubernetes, an open-source container orchestration platform, emerged as the standard solution for automating the deployment, scaling, and management of containerized applications.

- **Orchestration and Automation:** Kubernetes automates the process of scaling, distributing, and load balancing containers, which are typically grouped into pods (the smallest deployable unit in Kubernetes). Kubernetes also manages networking, storage, and configuration of containerized applications in a declarative way.[12]
- **Auto-Scaling and Load Balancing:** Kubernetes ensures high availability by automatically scaling applications based on demand and distributing traffic to containers across a cluster. This capability makes it particularly suited for cloud-native applications that require rapid scalability.
- **Fault Tolerance:** Kubernetes provides mechanisms for self-healing, meaning that if a container or pod fails, Kubernetes automatically restarts or replaces it to maintain application availability.
- **Multi-Cloud and Hybrid Cloud Support:** Kubernetes supports multi-cloud and hybrid-cloud environments, enabling organizations to run containerized workloads across different infrastructure providers while maintaining a consistent deployment experience.

Key Advancements in Containerization

The development of containerization has been accompanied by numerous key advancements that have further improved its efficiency, security, and usability.

Container Orchestration

As container adoption grew, so did the complexity of managing large-scale deployments. This gave rise to orchestration platforms like Kubernetes, Docker Swarm, and Apache Mesos, which automate the deployment, scaling, and operation of containerized applications.

- **Kubernetes:** The dominant container orchestration tool, Kubernetes provides robust features such as automatic scaling, load balancing, and rolling updates. It allows for the management of complex, multi-container applications and has become the de facto standard for container orchestration in cloud environments.
- **Docker Swarm:** Docker Swarm is Docker's own container orchestration tool, which integrates seamlessly with Docker's ecosystem and provides a simpler, less feature-rich alternative to Kubernetes. It is easier to set up and use for small-to-medium-sized containerized applications.
- **Service Discovery and Networking:** Orchestrators provide automatic service discovery, meaning that containers can easily find and communicate with each other in a dynamic environment. Kubernetes, for example, creates DNS records for services that enable containers to address each other by name rather than IP address.[13]

Security Enhancements

As containers have become more widely used, securing containerized applications and environments has become a key area of focus. Although containers share the host OS kernel, mechanisms have been developed to address the security concerns unique to containerization.

- **SELinux and AppArmor:** Security modules like SELinux (Security-Enhanced Linux) and AppArmor offer enhanced security for containers by enforcing access controls and restricting the actions containers can perform on the host OS.
- **Namespaces and Cgroups:** Namespaces provide isolation for containers, ensuring that each container has its own set of resources (like network interfaces and file systems), while cgroups limit and prioritize resources such as CPU, memory, and disk I/O. These mechanisms help prevent containers from interfering with each other and provide a secure environment.
- **Container Scanning and Security Tools:** To address vulnerabilities, tools like Clair, Anchore, and Trivy perform security scanning of container images to identify known vulnerabilities in dependencies and libraries, reducing the risk of introducing security flaws into production environments.

Serverless Computing

The rise of serverless computing has further emphasized the importance of containerization. Serverless computing abstracts the underlying infrastructure entirely, allowing developers to focus solely on writing code that is executed in response to events.

- **Function-as-a-Service (FaaS):** Serverless architectures often use containers as the execution environment for short-lived tasks or functions, such as AWS Lambda or Google Cloud Functions. These functions are packaged into containers, ensuring consistency across environments.
- **Scaling with Containers:** Serverless platforms dynamically allocate containers to run code only when needed, automatically scaling based on demand and charging only for the resources consumed during execution. This serverless approach benefits from containerization's efficiency and portability, making it a perfect fit for cloud-native applications.

Impact on Resource Management and Scalability

Both virtualization and containerization technologies have fundamentally transformed how operating systems manage resources and scale applications.

Resource Isolation

- **Virtual Machines:** VMs provide strong isolation, creating fully independent virtualized environments that simulate separate physical machines. This isolation ensures that VMs can run different operating systems with minimal interference.
- **Containers:** Containers are lighter and share the host OS kernel but still ensure separation through namespaces, which provide process, file system, and network isolation. Although containers offer less isolation than VMs, they are highly efficient and suitable for microservices and stateless applications.

Efficient Resource Utilization

- **Virtualization:** Through resource consolidation, virtualization allows multiple VMs to run on a single physical machine, improving resource utilization. Hypervisors manage the allocation of resources between VMs, ensuring that each VM receives an appropriate share of the physical host's capabilities.
- **Containers:** Containers enable even greater efficiency because they share the same OS kernel and avoid the overhead of running a full guest OS. This allows for a higher density of applications on the same host and faster startup times.

Scalability

- **Virtualization:** Virtualization supports horizontal scaling by creating and deploying multiple VMs to handle increasing loads. Each new VM is an independent environment, and scaling involves adding more VMs across physical servers.
- **Containers:** Containers enable microservice architectures, which can scale horizontally by adding or removing container instances based on demand. Container orchestration platforms like Kubernetes automate this scaling process, allowing for seamless handling of fluctuating traffic.[14]

In summary, virtualization and containerization technologies have drastically improved resource management and scalability, making it possible to run applications more efficiently and scale them dynamically in modern cloud and hybrid environments.

Integration and Hybrid Models

As modern operating systems evolve, there is an increasing trend to integrate both virtualization and containerization technologies in hybrid models to maximize the benefits of both. These hybrid approaches aim to address the diverse workload requirements and the need for greater flexibility, scalability, and resource optimization across different environments.

Containerized Virtual Machines (VMs)

One of the innovative approaches to integration involves combining containerization and virtualization into a single system. The idea is to run containers within virtual machines, taking advantage of both technologies' strengths:

- **VM Isolation and Container Efficiency:** Virtual machines provide strong isolation and security by running separate OS instances on a host, whereas containers are lightweight and fast. By running containers within VMs, the system can benefit from the robustness and isolation offered by VMs while enjoying the speed, portability, and resource efficiency of containers.
- **Use Case:** This hybrid model is useful for applications that require enhanced isolation, such as multi-tenant cloud environments, but still need the flexibility and resource efficiency offered by containers. For example, in cloud infrastructures, a hypervisor might manage a set of VMs, and within those VMs, containers can be used to deploy microservices.

This approach has become particularly useful in multi-cloud environments where workloads need to be isolated but still need the speed and portability offered by containers.[15]

Virtualization and Microservices

- Microservices architecture is increasingly popular in cloud-native applications, and containers are often used as the core technology for building and deploying microservices due to their lightweight nature. However, virtualization still plays a crucial role in ensuring fault tolerance, security, and resource isolation at a more granular level:
- **Microservice Containers:** Containers allow microservices to be packaged with their dependencies and deployed independently, ensuring each microservice runs in its own environment. Kubernetes and Docker are commonly used for orchestrating these microservices across a cluster of machines.
- **Virtualization for Fault Tolerance:** While containers provide application-level isolation, virtualization ensures that workloads are isolated at the hardware level, providing an extra layer of fault tolerance. In environments where high availability and security are critical, virtualization can be used in tandem with containers to ensure more robust isolation.[16]

In this model, the containerized microservices are deployed in virtualized environments to ensure greater control over the infrastructure while maintaining the efficiency and scalability of containers.

Bare Metal Containers

Bare metal containers refer to containers running directly on physical hardware without the use of a hypervisor or virtual machine layer. This approach can be highly efficient for specific workloads requiring minimal overhead and near-native performance:

- **Isolation with Performance:** By running directly on bare metal, containers can avoid the resource overhead introduced by hypervisors, offering improved performance for high-performance computing (HPC) or latency-sensitive applications.
- **Edge Computing:** Bare metal containers are also gaining traction in edge computing environments, where low-latency and high-performance solutions are essential. By avoiding the overhead of virtualization, these systems can provide faster processing times, essential for real-time applications like IoT, autonomous systems, and smart cities.

Bare metal containers combine the performance benefits of running directly on hardware with the operational flexibility and portability of containers, making them a powerful option for environments that require the highest levels of efficiency.

Challenges and Future Directions

While virtualization and containerization have brought significant advancements in operating systems and computing architectures, several challenges still need to be addressed to fully realize their potential. These challenges encompass areas such as security, performance, and management complexity, which continue to be focal points for research and development.

Security Concerns

- **Container Security:** One of the main challenges in containerization is the security risk associated with sharing the host OS kernel among containers. Since containers do not run their own independent OS, a vulnerability in the host kernel can potentially expose all containers to attack.
- **Security Mechanisms:** Techniques such as seccomp, AppArmor, and user namespaces are being implemented to enhance security by restricting the actions that containers can perform. Container scanning tools also help detect vulnerabilities in container images before they are deployed.
- **VM Security:** Virtual machines provide stronger isolation due to their independent operating systems, making them inherently more secure in multi-tenant environments. However, they are still susceptible to hypervisor vulnerabilities, and new threats such as side-channel attacks on VM isolation need continuous monitoring.[17]

The development of security-enhancing technologies and best practices will continue to be a critical area of focus for both virtualization and containerization moving forward.

Performance Overhead

- **Overhead in Virtualization:** While virtualization offers excellent isolation, the overhead of running multiple guest operating systems on the same hardware can be significant, especially for resource-intensive applications.
- **Container Performance:** Containers are more efficient than VMs because they share the host OS kernel, but they still introduce some overhead, particularly in workloads that require high resource isolation or utilize extensive networking or storage operations.

As containerization technology evolves, the performance gap between containers and bare-metal performance will likely decrease. New techniques such as lightweight hypervisors or improved container runtimes (e.g., gVisor, Kata Containers) may help reduce overhead.

Complexity in Management

As containerization and virtualization technologies grow in complexity, managing and orchestrating large-scale infrastructures can become challenging.

- **Container Orchestration:** Platforms like Kubernetes have become essential for managing containers in production environments, but these tools come with their own set of challenges. The complexity of Kubernetes clusters, along with the steep learning curve and resource consumption, can be barriers to effective adoption and management.
- **Unified Management Platforms:** There is a need for integrated management platforms that can seamlessly handle both containerized and virtualized environments. Such platforms would provide a unified approach to managing workloads, networking, and security across hybrid infrastructures.

The goal will be to simplify management while maintaining scalability and flexibility. This will require more advanced automation, better monitoring and diagnostics, and integration between various management tools.

Future Directions

Several exciting innovations are on the horizon that will drive the next wave of improvements in both virtualization and containerization technologies:

Lightweight Hypervisors

One of the key future directions is the development of lightweight hypervisors that combine the advantages of both containerization and virtualization:

- **Hybrid Hypervisors:** These hypervisors would aim to offer the same isolation and fault tolerance of VMs while providing the resource efficiency and speed of containers. Innovations in this space could combine the best of both worlds, offering high-performance virtualization with minimal overhead.

- **Improved Container Runtimes:** Enhanced container runtimes may offer greater isolation and security while reducing performance penalties, making containers suitable for a broader range of workloads.

Edge Computing

The increasing importance of edge computing—where computing resources are deployed closer to the data source, such as IoT devices—will likely drive further innovations in both virtualization and containerization:

- **Low-Latency Solutions:** Edge computing often requires low-latency solutions for real-time decision-making, which means optimizing resource management in highly distributed environments.
- **Resource Efficiency:** Containerization, coupled with lightweight virtualization, will become a key technology for deploying efficient applications at the edge, particularly in environments with limited hardware resources.

Security Advancements

The development of new security frameworks and tools will be crucial to addressing the risks associated with containerized and virtualized environments:

- **Stronger Isolation:** Research into improving isolation between containers, as well as hypervisor security, will be essential in mitigating vulnerabilities that could be exploited.
- **Zero Trust Models:** Future containerization and virtualization systems may integrate zero-trust security models, where every interaction and component is assumed to be untrusted, and continuous verification is required.

Serverless and Containers

Serverless computing will continue to evolve, and containerization will play a critical role in serverless execution environments. Containers will provide the underlying infrastructure for serverless architectures, with auto-scaling and resource optimization becoming even more sophisticated.

Conclusion

Virtualization and containerization have dramatically reshaped the landscape of operating systems and modern computing environments. Virtualization provides strong isolation, scalability, and fault tolerance, while containerization offers a more lightweight, flexible, and efficient solution for application deployment. The integration of both technologies in hybrid models will continue to define the next generation of OS design. Despite their advancements, security, performance overhead, and management complexity remain key challenges. Future research and development will focus on overcoming these obstacles, with the goal of enhancing the efficiency, scalability, and security of modern OS environments. As these technologies evolve, they will drive further innovations in cloud-native applications, edge computing, and the management of increasingly complex, large-scale infrastructures.

References

1. VMware. VMware vSphere: Virtualization software for building cloud infrastructures.
2. Red Hat. Kubernetes: Orchestrate containers with Kubernetes.
3. Docker Inc. Docker: Build, Ship, and Run Any App, Anywhere.
1. Hennessy JL, Patterson DA. Computer Architecture: A Quantitative Approach. 5th ed. Amsterdam: Elsevier; 2011.
2. Joshi S, Mandal S, Dey D. Cloud Computing: Concepts, Technology & Architecture. 1st ed. Berlin: Springer; 2011.
3. Smith M, Chen S. Virtualization Technologies in the Cloud Computing Era. J Cloud Comput. 2018;7(1):10-25. doi:10.1186/s13677-018-0139-2.
4. Bernstein D. Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Comput. 2014;1(3):81-84. doi:10.1109/MCC.2014.51.
5. De Moura D. Securing Containers: An Overview. Int J Inf Sec. 2020;16(2):25-34. doi:10.1007/s10207-020-0541-0.
6. Kivity A, Laor D, Lublin M, Lutz B. KVM: The Linux Virtual Machine Monitor. In: Proceedings of the 2007 Linux Symposium; 2007 Jul 17-20; Ottawa, Canada. Ottawa: USENIX Association; 2007. p. 225-230.
7. Miller A, Phillips L. The Evolution of Virtualization Technologies and their Role in Cloud Computing. J Cloud Inf. 2019;5(2):45-56.
8. Ruan H, Jiang W, Tang L. Performance Comparison of Virtualization and Containerization Technologies. J Comput Sci. 2020;12(3):89-102. doi:10.1016/j.jcsc.2020.03.003.
9. Billions R. Performance Overhead of Containerization in Cloud Environments. Comput Syst Res. 2021;18(4):158-166.
10. Li Y, Wang Z, Zhang S. The Future of Edge Computing: Virtualization and Containerization Approaches. Future Gener Comput Syst. 2021;117:139-149. doi:10.1016/j.future.2020.08.048.
11. He K, Lee D. A Survey on Security in Containerized Environments. ACM Comput Surv. 2019;52(4):1-28. doi:10.1145/3242673.

12. Bui T. Analysis of docker security. arXiv preprint arXiv:1501.02967. 2015 Jan 13.
13. Naylor D, Tiel A. Containerization and Virtualization: A Comparative Study of Technology Adoption. Comput Manage. 2019;47(8):47-53.
14. Cunningham A. The Rise of Hybrid Cloud Computing Models. J Cloud Technol. 2020;10(2):113-121.