World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Review Article)

Check for updates

Virtualization in the Cloud Era: Understanding the relationship between VMs, containers and serverless

Arunkumarreddy Yalate *

Mutual Of Omaha, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(02), 1920-1927

Publication history: Received on 04 April 2025; revised on 13 May 2025; accepted on 15 May 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.2.0705

Abstract

This article examines the complex landscape of virtualization technologies that have transformed modern cloud infrastructure, focusing on the relationship between virtual machines (VMs), containers, and serverless computing. The article analyzes how these paradigms represent a spectrum of increasing abstraction and decreasing developer control, each offering distinct advantages for specific workloads and organizational contexts. The article provides practitioners with a decision framework for selecting appropriate virtualization approaches. The article investigation reveals that rather than converging on a single paradigm, successful organizations implement polyglot virtualization strategies that leverage each technology's strengths while mitigating limitations through cross-paradigm integration patterns. The article explores how VMs continue to excel in regulated environments requiring strong isolation, containers dominate microservices architectures and CI/CD pipelines, and serverless computing optimizes cost and developer productivity for event-driven workloads. The article further analyzes orchestration ecosystems across paradigms and examines emerging trends including container-native virtualization, expanded serverless capabilities, and advanced security innovations that are reshaping the virtualization landscape. This holistic perspective provides essential guidance for architects and decision-makers navigating the complex trade-offs of modern cloud infrastructure.

Keywords: Virtualization Paradigms; Container Orchestration; Serverless Computing; Hybrid Cloud Architectures; Infrastructure Abstraction

1. Introduction

The landscape of computing infrastructure has undergone a profound transformation over the past two decades, evolving from physical hardware to increasingly abstract virtualization technologies. This evolution reflects the industry's relentless pursuit of resource efficiency, operational agility, and cost optimization. Virtual machines (VMs) emerged in the early 2000s as the first widespread virtualization paradigm, enabling organizations to partition physical servers into multiple isolated environments, significantly improving hardware utilization and management flexibility. By 2013, container technologies like Docker revolutionized application packaging and deployment by providing a lightweight alternative to VMs, while more recently, serverless computing has pushed abstraction even further by eliminating infrastructure management concerns for developers [1].

The coexistence of these three paradigms—VMs, containers, and serverless computing—presents both opportunities and challenges for organizations navigating modern cloud infrastructure. Each approach offers distinct advantages in specific contexts, with traditional VMs providing strong isolation guarantees essential for regulated industries, containers enabling microservice architectures and streamlined CI/CD pipelines, and serverless computing optimizing for cost-efficiency in event-driven workloads. However, selecting the appropriate virtualization technology requires nuanced understanding of their underlying mechanisms, operational implications, and architectural trade-offs.

* Corresponding author: Arunkumarreddy Yalate.

This article examines the complex relationship between these virtualization paradigms in contemporary cloud environments. We analyze their fundamental differences in resource management, isolation properties, scalability characteristics, and operational overhead. Through comparative analysis and use case studies, we provide a decision framework for practitioners to select appropriate technologies based on workload requirements. Furthermore, we explore orchestration tools that have evolved alongside each paradigm, from VMware vSphere for VM management to Kubernetes for container orchestration and AWS Step Functions for serverless workflows. Finally, we examine how these virtualization approaches complement one another in hybrid and private cloud environments, where organizations must balance innovation with compliance requirements, legacy systems integration, and strategic flexibility.

2. Foundational Technologies and Concepts

2.1. Hypervisor-based Virtualization Principles

Hypervisor-based virtualization creates virtual machines that emulate physical hardware, enabling multiple operating systems to run on a single physical server. Type 1 (bare-metal) hypervisors run directly on hardware, while Type 2 hypervisors operate as applications within a host OS. Modern hypervisors employ hardware-assisted virtualization through technologies like Intel VT-x and AMD-V to reduce performance overhead. Memory management relies on techniques such as shadow page tables or Extended Page Tables (EPT) to maintain isolation between VMs [2].

2.2. Container Technology Fundamentals

Containers package applications with their dependencies while sharing the host OS kernel, eliminating the need for guest OS instances. Core technologies include namespaces for isolating system resources, cgroups for resource control, and union file systems for efficient image building. Container runtimes like containerd or CRI-O implement the Open Container Initiative (OCI) specification, providing standardized container execution. Docker popularized containers through developer-friendly tooling, while container orchestration platforms emerged to manage container deployments at scale [3].

2.3. Serverless Computing Architecture

Serverless computing abstracts infrastructure management entirely, allowing developers to focus solely on application logic through function-as-a-service (FaaS) offerings. These platforms handle provisioning, scaling, and infrastructure management automatically. Functions execute in ephemeral environments, typically triggered by events, with billing based on execution time and resources consumed. Cold starts occur when functions initialize after periods of inactivity, creating latency challenges for certain workloads. Serverless architectures typically integrate with cloud provider services for event sources, storage, and service composition.

2.4. Abstraction Layers Across Paradigms

The three paradigms form a spectrum of increasing abstraction and decreasing developer control. VMs abstract hardware but require OS management; containers abstract OS dependencies but require container orchestration; serverless abstracts infrastructure entirely but introduces vendor dependencies. Higher abstraction generally correlates with faster deployment, reduced operational overhead, and more constrained execution environments. Organizations often employ multiple abstraction layers simultaneously, selecting the appropriate level based on workload requirements, control needs, and developer expertise.

3. Comparative Analysis of Virtualization Approaches

3.1. Resource Allocation and Utilization Patterns

VMs allocate fixed resources to guests, often resulting in underutilization. Containers share kernel resources more efficiently, enabling higher density but risking resource contention. Serverless platforms dynamically allocate resources per invocation, optimizing utilization through statistical multiplexing across many workloads. Resource efficiency typically increases moving from VMs to containers to serverless, while predictability and isolation decrease along the same spectrum [4].

3.2. Isolation Mechanisms and Security Boundaries

VMs provide strong isolation through hardware virtualization and separate OS instances, making them suitable for multi-tenant environments with strict security requirements. Containers offer weaker isolation, relying on kernel features that can potentially be compromised. Serverless functions typically run in containers or lightweight VMs with additional security controls enforced by the provider. Defense-in-depth approaches often combine these technologies, using VMs to isolate tenant workloads while leveraging containers within those boundaries.

3.3. Scalability Characteristics and Limitations

VMs scale with minutes of provisioning time and significant memory overhead, making rapid scaling challenging. Containers start in seconds with lower overhead, enabling more responsive scaling through orchestration platforms. Serverless functions scale near-instantly to thousands of concurrent executions with zero management overhead, though subject to provider limits. Each technology faces distinct scaling limitations: VM density is constrained by hardware resources, container scalability by orchestration complexity, and serverless by cold start latency and execution time limits.

3.4. Operational Overhead and Management Complexity

VM operations require managing guest OS patching, licensing, and hardware allocation. Container operations focus on orchestration, networking, and storage persistence challenges. Serverless reduces operational concerns but introduces monitoring, debugging, and vendor integration complexity. As abstraction increases, operational responsibilities shift from infrastructure to application concerns, though the overall complexity may remain similar but redistributed across different domains of expertise.

3.5. Performance Benchmarks Across Use Cases

Performance characteristics vary significantly by workload type. Compute-intensive tasks show minimal overhead differences between paradigms. I/O-intensive workloads experience greater performance variation, with VMs benefiting from direct hardware access and containers from reduced context switching. Serverless platforms demonstrate excellent performance for bursty, event-driven workloads but suffer from cold start latency for infrequently accessed functions. Network-intensive applications may experience additional overhead in containerized and serverless environments due to overlay networks and security boundaries.

Table 1 Comparison of Virtualization Paradigms [2-4]

Characteristic	Virtual Machines	Containers	Serverless
Abstraction Level	Hardware	Operating System	Application
Resource Allocation	Fixed allocation, potential underutilization	Shared kernel resources, higher density	Dynamic per-invocation allocation
Isolation	Strong (hardware-level)	Moderate (kernel-level)	Variable (provider-dependent)
Startup Time	Minutes	Seconds	Milliseconds
Management Overhead	OS maintenance, patching, licensing	Orchestration, networking, storage	Monitoring, debugging, vendor integration
Typical Use Cases	Regulated environments, legacy systems	Microservices, CI/CD pipelines	Event-driven workloads, bursty traffic

4. Use Case Analysis: When to Choose Each Paradigm

4.1. VM Suitability: Isolation-critical Environments (Financial/Healthcare)

Virtual machines remain indispensable in regulated industries where strong workload isolation is paramount. In healthcare organizations processing Protected Health Information (PHI) under HIPAA regulations, VMs provide clear security boundaries that simplify compliance verification and attestation. Financial institutions handling sensitive payment data leverage VM isolation to implement strict multi-tenant architectures, ensuring that compromises in one application cannot affect others. VMs also excel in legacy modernization scenarios where organizations need to maintain older operating systems while migrating to cloud environments. Their hardware-level isolation guarantees make them suitable for workloads with specific licensing requirements tied to physical hardware characteristics [5].

4.2. Container Advantages: Microservices Architectures and CI/CD Pipelines

Containers deliver compelling benefits for microservice architectures by packaging services with their dependencies while minimizing resource overhead. Development teams achieve significant productivity gains through consistent environments across development, testing, and production. Container images serve as immutable deployment units, enabling reliable continuous delivery pipelines and simplified rollback strategies. Organizations implementing DevOps practices benefit from containers' rapid startup times and efficient resource utilization, allowing developers to run production-equivalent environments locally. The standardized OCI image format facilitates multi-cloud portability, reducing vendor lock-in concerns. These characteristics make containers particularly effective for web applications, stateless services, and batch processing workloads with moderate isolation requirements.

4.3. Serverless Benefits: Event-driven, Cost-optimized Workloads

Serverless computing demonstrates clear advantages for intermittent, event-driven workloads with variable demand patterns. Common applications include API backends, data processing pipelines, scheduled tasks, and webhook handlers. The pay-per-use pricing model eliminates costs during idle periods, creating significant savings for bursty workloads compared to continuously running alternatives. Development velocity increases as teams focus on business logic without managing infrastructure. Serverless particularly excels for organizations implementing event-driven architectures where components communicate through events rather than direct calls. The automatic scaling capabilities handle traffic spikes effectively without pre-provisioning capacity, though cold starts introduce latency considerations for latency-sensitive applications.

4.4. Decision Framework for Technology Selection

Organizations should apply a structured decision framework when selecting virtualization technologies. The framework begins with workload characteristics assessment: isolation requirements, scaling patterns, state management needs, and performance constraints. Additional factors include operational considerations (team expertise, existing investments), compliance requirements, and financial models (CapEx vs. OpEx preferences). Many organizations implement a tiered approach, using VMs for isolation-critical workloads, containers for stateless applications and microservices, and serverless for event-driven components with variable demand. This hybrid approach optimizes for both security and developer productivity, allowing selective abstraction based on workload characteristics rather than forcing a one-size-fits-all solution.

5. Orchestration Ecosystems

5.1. VM Management: VMware vSphere and Alternatives

VMware vSphere remains the dominant enterprise VM management platform, providing centralized control of virtualized resources through features like vMotion (live migration), DRS (distributed resource scheduling), and HA (high availability). Alternative solutions include Microsoft Hyper-V with System Center Virtual Machine Manager, offering deep Windows integration, and open-source KVM managed through platforms like oVirt or Proxmox. Cloud providers offer managed VM services like AWS EC2, Azure VMs, and Google Compute Engine, which integrate infrastructure-as-code capabilities through services like AWS CloudFormation and Azure Resource Manager. These tools have evolved to incorporate advanced automation features, though they generally operate with longer provisioning timelines than container or serverless alternatives [6].

5.2. Container Orchestration: Kubernetes Ecosystem

Kubernetes has emerged as the de facto standard for container orchestration, providing declarative configuration, self-healing capabilities, and extensible architecture. The ecosystem includes essential components like networking solutions (Calico, Cilium), service meshes (Istio, Linkerd), and storage providers (Rook, Portworx). Cloud providers offer managed Kubernetes services like GKE, EKS, and AKS, reducing operational overhead while maintaining workload portability. The Cloud Native Computing Foundation (CNCF) fosters an extensive ecosystem of complementary tools addressing monitoring, security, and developer workflows. Enterprise adoption typically includes CI/CD integration through tools like ArgoCD and Flux, implementing GitOps patterns for continuous deployment and configuration management.

5.3. Serverless Workflow Management: AWS Step Functions and Competitors

AWS Step Functions provides visual workflow orchestration for serverless applications, coordinating function executions through state machines defined in Amazon States Language. Similar offerings include Azure Logic Apps, Google Cloud Workflows, and IBM Cloud Functions Composer. These services enable complex business processes by connecting multiple functions with conditional logic, error handling, and parallel execution. Open-source alternatives like Temporal and Netflix Conductor offer vendor-neutral workflow engines with hybrid deployment options. Regardless of platform, these tools address the "function composition problem" inherent in serverless architectures, providing state management and coordination for otherwise stateless functions.

5.4. Cross-paradigm Management Tools and Approaches

Organizations increasingly deploy cross-paradigm management tools to unify operations across VMs, containers, and serverless resources. Infrastructure-as-code solutions like Terraform and Pulumi enable consistent provisioning across all paradigms. Observability platforms including Datadog, New Relic, and Dynatrace provide unified monitoring across virtualization boundaries. Service mesh implementations extend beyond containers to include VMs and sometimes serverless functions, creating consistent networking policies and telemetry. Cloud management platforms (CMPs) offer centralized governance regardless of underlying virtualization technology. These cross-cutting tools reflect the reality that most organizations maintain heterogeneous environments rather than standardizing on a single virtualization paradigm.

6. Virtualization in Hybrid and Private Cloud Environments

6.1. Traditional Virtualization Persistence in Enterprise Settings

Despite cloud migration trends, traditional VM-based virtualization maintains a significant presence in enterprise environments. Organizations have invested heavily in VM infrastructure, skilled personnel, and established operational practices that continue to deliver value. On-premises virtualization remains essential for workloads with specialized hardware requirements, extreme performance needs, or data gravity constraints. Many enterprises operate in a steady state where 60-70% of workloads remain virtualized rather than containerized or serverless. This persistence is particularly evident in industries with legacy applications that cannot be easily refactored for newer paradigms. VM technology continues to evolve with features like nested virtualization, enhanced security capabilities, and integration with cloud management platforms [7].

Table 2 Orchestration Ecosystems by Virtualization Paradigm [6,7]

Paradigm	**Key Orchestration Tools**	**Primary Features**	**Cloud Provider Implementations**
Virtual Machines	VMware vSphere, Microsoft Hyper-V, KVM/oVirt	Live migration, high availability, resource scheduling	AWS EC2, Azure VMs, Google Compute Engine
Containers	Kubernetes, Docker Swarm	Declarative configuration, self-healing, service discovery	GKE, EKS, AKS
Serverless	AWS Step Functions, Azure Logic Apps, Google Cloud Workflows	Visual workflow orchestration, state machines, parallel execution	AWS Lambda, Azure Functions, Google Cloud Functions
Cross-paradigm	Terraform, Pulumi, service meshes	Infrastructure-as-code, unified observability	Cloud management platforms (CMPs)

6.2. Compliance and Regulatory Considerations

Regulatory frameworks significantly influence virtualization strategy in regulated industries. GDPR, HIPAA, PCI-DSS, and sector-specific regulations impose requirements around data residency, access controls, and audit capabilities that often favor traditional virtualization with its well-established compliance controls. VM-based environments provide clearer tenant boundaries and more straightforward audit trails than containerized alternatives. Private cloud deployments using VMs enable organizations to maintain physical control over regulated data while leveraging some cloud operational benefits. However, certification programs from major cloud providers and container platforms are gradually addressing these concerns, allowing regulated workloads to adopt newer paradigms with appropriate controls.

6.3. Integration Patterns Across Paradigms

Most enterprises implement hybrid approaches that integrate multiple virtualization paradigms. Common patterns include "VM-as-platform" approaches where VMs host container orchestration clusters, providing isolation boundaries while enabling container density within those boundaries. API gateway patterns connect serverless functions to containerized or VM-based services. Data plane separation approaches keep stateful components in VMs while moving stateless workloads to containers or serverless. Event mesh architectures enable cross-paradigm communication through message brokers and event streams. These integration patterns recognize that different workloads have different requirements, allowing organizations to select appropriate technologies for each component while maintaining interoperability.

6.4. Migration Strategies Between Virtualization Approaches

Organizations employ several strategies when migrating between virtualization paradigms. The "lift and shift" approach moves applications to cloud VMs with minimal changes, providing a foundation for incremental modernization. "Containerize in place" strategies package applications in containers while maintaining VM-based infrastructure. "Refactor and rearchitect" approaches decompose monolithic applications into microservices suitable for containers or serverless deployment. "Strangler fig" patterns gradually replace components of legacy systems with modern alternatives. Successful migrations typically involve discovering application dependencies, identifying suitable candidates for each paradigm, establishing common networking and security models, and implementing robust CI/CD pipelines that support heterogeneous deployment targets.

Table 3 Migration Strategies Between Virtualization Paradigms [5, 7]

Strategy	Description	Benefits	Challenges	Suitable For
Lift and Shift	Move applications to cloud VMs with minimal changes	Speed, low risk, minimal refactoring	Limited cloud-native benefits	Legacy applications, time-sensitive migrations
Containerize in Place	Package applications in containers while maintaining VM infrastructure	Improved deployment consistency, gradual modernization	Partial transformation, potential complexity	Organizations building DevOps capabilities
Refactor and Rearchitect	Decompose monolithic applications into microservices	Full cloud-native benefits, optimal resource usage	Time-intensive, requires architectural changes	Strategic applications with long-term value
Strangler Fig Pattern	Gradually replace components of legacy systems	Incremental risk management, continuous delivery	Extended parallel operation, integration complexity	Complex systems that cannot be replaced at once

7. Future Trends and Emerging Paradigms

7.1. Evolution of Serverless Beyond Functions

Serverless is evolving beyond simple function execution toward comprehensive application platforms. Container-based serverless offerings like AWS Fargate and Azure Container Instances provide serverless operational models for container workloads. Database services including DynamoDB, Cosmos DB, and Firebase offer serverless data persistence with automatic scaling. Serverless frameworks increasingly support stateful workloads through durable execution contexts and improved state management. Edge computing platforms are adopting serverless models to deploy code closer to users. These developments suggest convergence toward a broader "serviceful" computing model where managed services handle infrastructure concerns regardless of the underlying execution model [8].

7.2. Container-native Virtualization Advances

Container-native virtualization blends container and VM paradigms to address security and compatibility challenges. Technologies like Kata Containers, AWS Firecracker, and Google gVisor create lightweight VM boundaries around containers, providing stronger isolation while maintaining container-like startup performance. Kubernetes operators for VMs enable unified management of containers and VMs through the same orchestration layer. Hardware-assisted virtualization extensions continue to reduce virtualization overhead, enabling more efficient nested virtualization. These advances are creating a spectrum of isolation options rather than a binary choice between containers and VMs, allowing security teams to implement appropriate isolation based on workload sensitivity.

7.3. Security Innovations Across Virtualization Layers

Security innovations are addressing key challenges in multi-tenant virtualization. Confidential computing technologies like AMD SEV, Intel SGX, and ARM TrustZone protect data in use from privileged users and cloud providers. Zero-trust architectures implement consistent identity-based access controls across all virtualization paradigms. Supply chain security tools verify container and function provenance through signed images and attestation. Runtime security monitoring detects and prevents abnormal behavior in virtualized environments. These advances collectively enable deployment of sensitive workloads in shared infrastructure with enhanced isolation guarantees, accelerating adoption of newer virtualization paradigms in security-conscious organizations.

7.4. Industry Adoption Patterns and Predictions

Industry trends indicate continued diversification of virtualization approaches rather than convergence on a single paradigm. We predict accelerated VM-to-container migration for non-legacy workloads, with containers becoming the default deployment model for new applications by 2026. Serverless adoption will expand beyond simple event handlers to encompass complex applications, particularly for organizations prioritizing developer productivity over operational control. Multi-cloud strategies will drive demand for abstraction layers that span virtualization technologies. Edge computing will accelerate hybrid architectures combining on-premises and cloud resources. Security and compliance capabilities will mature across all paradigms, reducing barriers to adoption for regulated industries. Overall, we expect organizations to implement strategic polyglot virtualization, selecting appropriate technologies based on workload requirements rather than standardizing on a single approach.

Table 4 Emerging Trends in Virtualization Technologies [7, 8]

Trend	Key Technologies	Impact	Industry Implications
Serverless Evolution	AWS Fargate, Azure Container Instances, serverless databases	Expanded application scope beyond functions	Convergence toward "serviceful" computing models
Container-native Virtualization	Kata Containers, AWS Firecracker, gVisor	Enhanced isolation with container-like performance	Spectrum of isolation options rather than binary choices
Confidential Computing	AMD SEV, Intel SGX, ARM TrustZone	Protection of data-in-use from privileged users	Accelerated adoption in regulated industries
Edge Computing Integration	IoT platforms, edge serverless offerings	Distributed execution closer to data sources	Hybrid architectures combining edge and cloud resources

8. Conclusion

The evolution of virtualization technologies from VMs to containers to serverless computing represents a continuum of increasing abstraction and decreasing operational overhead, each offering distinct advantages for specific workloads and organizational contexts. As the article demonstrates, these paradigms are best viewed as complementary rather than competitive approaches within a comprehensive cloud strategy. The persistence of VM technology in regulated environments, the dominance of containers for microservices architectures, and the rapid adoption of serverless for event-driven workloads illustrate that no single virtualization approach satisfies all requirements. Forward-looking organizations are implementing polyglot virtualization strategies that leverage each technology's strengths while mitigating its limitations through cross-paradigm integration patterns and unified management tools. As container-native virtualization advances blend the security benefits of VMs with the efficiency of containers, and as serverless platforms expand beyond functions to support more complex application architectures, the boundaries between paradigms will continue to blur. This convergence, coupled with innovations in confidential computing and zero-trust security models, will enable increasingly sophisticated workloads to migrate toward higher levels of abstraction without compromising on security or performance. The most successful cloud adoption strategies will remain those that match workload characteristics to appropriate virtualization technologies while maintaining operational consistency and security across paradigms.

References

[1] Ioana Baldini, Paul Castro et al. "Serverless computing: Current trends and open problems. In Research Advances in Cloud Computing (pp. 1-20). Springer, Singapore. 28 December 2017. https://doi.org/10.1007/978-981-10-5026-8_1

[2] Paul Barham, Boris Dragovic, et al. "Xen and the art of virtualization". ACM SIGOPS Operating Systems Review, 37(5), 164-177, 19 October 2003. https://doi.org/10.1145/1165389.945462

[3] Brendan Burns, Brian Grant,et al. "Borg, Omega, and Kubernetes". Communications of the ACM, 59(5), 50-57, 26 April 2016. https://doi.org/10.1145/2890784

[4] Johann Schleier-Smith, Vikram Sreekanti, et al. "What serverless computing is and should become: The next phase of cloud computing". Communications of the ACM, 64(5), 76-84, 26 April 2021. https://doi.org/10.1145/3406011

[5] Wayne Jansen, Timothy Grance, T. (December 2011). "Guidelines on security and privacy in public cloud computing". NIST Special Publication 800-144. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-144

[6] John Fink. "Docker: a Software as a Service, Operating System-Level Virtualization Framework". Code4Lib Journal, Issue 25, 2014-07-21. https://journal.code4lib.org/articles/9669

[7] David Bernstein. "Containers and Cloud: From LXC to Docker to Kubernetes". IEEE Cloud Computing, 1(3), 81-84, 30 September 2014 . https://doi.org/10.1109/MCC.2014.51

[8] Paul Castro, Vatche Ishakian, et al. "The Rise of Serverless Computing". Communications of the ACM, 62(12), 44-54, 21 November 2019. https://doi.org/10.1145/3368454