pISSN:2349-5707 Volume:12, Issue 09, September -2025 eISSN:2349-5715

Architecting Secure, Multi-Tenant Cloud Systems: Integrating Zero-Trust, Containerization, and Data-Centric Protections for Resilient Cloud Services

John R. Bennett
Global Institute of Computing, United Kingdom

ABSTRACT: This article presents an integrative, theory-driven and practice-oriented examination of secure architectures for multi-tenant cloud systems. Drawing on foundational definitions and early problem framing in cloud computing and synthesising contemporary work on container-based multitenancy, zero-trust paradigms, and data security mechanisms, the paper develops a conceptual framework for building resilient, privacy-preserving, and operationally efficient multi-tenant cloud services. The framework emphasises a layered approach that combines (1) formal tenant isolation through container and micro-virtualization strategies, (2) identity and access management implemented via zero-trust principles, (3) hardware-rooted data protection and attestation using Trusted Platform Modules and related techniques, and (4) resource allocation and energy-aware orchestration that preserves security quotas without compromising performance. Methodologically, the research synthesises evidence from literature reviews, comparative architectural analysis, and thought experiments grounded in canonical cloud components (compute, storage, networking, orchestration). The results articulate specific design patterns and trade-offs, highlight operational constraints such as latencies introduced by isolation, and expose gaps where current cloud services offer partial but insufficient protection. The discussion interrogates the theoretical implications for multi-tenant security, examines counter-arguments (e.g., performance vs. security trade-offs), and maps an agenda for experimental validation and incremental industry adoption. The article concludes with prescriptive recommendations for architects, cloud service providers, and researchers focused on advancing secure multi-tenant cloud infrastructures that are compatible with contemporary serverless and container orchestration models. Throughout, claims and recommendations are grounded in extant literature to ensure reproducibility and scholarly rigor.

Keywords: multi-tenant cloud security; zero-trust; containerization; trusted platform module; data protection; resource orchestration; cloud architecture

INTRODUCTION

Cloud computing fundamentally reconfigures how computational resources and services are provisioned, consumed, and managed. The canonical definition provided by leading standards bodies frames cloud computing as on-demand, scalable, and pooled resources delivered over a networked environment (Mell & Grance). This pooled model, while economically and operationally beneficial, creates a distinctive set of security challenges because multiple, independent tenants share underlying physical and logical infrastructure (Vouk, 2008). Early scholarship identified broad concerns—data confidentiality, integrity, availability, and regulatory compliance—while more recent work has spotlighted architectural techniques for enforcing multitenancy and tenant isolation (Vouk, 2008; Fiaidhi et al., 2012).

As cloud platforms matured, new forms of tenancy emerged: beyond virtual machines to containers, serverless functions, and managed platform services. Containerization has enabled higher density and faster deployment cycles, but it also complicated the isolation boundary because containers share a kernel and other host resources (Truyen et al., 2016). Consequently, the notion of multitenancy requires careful revisiting: enforcing tenancy is no longer solely a matter of hypervisor configuration but involves fine-grained orchestration, runtime policy enforcement, and often hardware support (Truyen et al., 2016; Fiaidhi et al., 2012). The proliferation of managed database services, analytical engines, and serverless offerings—illustrated by widely

used services like Amazon Aurora Serverless, AWS Redshift, Azure Cosmos DB, and others—further multiplies the potential attack surface by coupling multi-tenant data stores with event-driven compute (Amazon Aurora Serverless, 2020; AWS Redshift, 2020; A Technical Overview of Azure Cosmos DB, 2020; AWS Athena, 2020).

Concurrently, the security research and practice communities have embraced zero-trust as a paradigm shift: rather than assuming perimeter security, zero-trust assumes compromise and therefore requires continuous verification of identities, devices, and contextual signals for every access request (Hariharan, 2025). Applying zero-trust in multi-tenant clouds demands reconciliation between ephemeral workloads (short-lived containers and functions), identity lifecycles, and enforcement points dispersed across orchestration planes and data planes (Hariharan, 2025; Truyen et al., 2016). Additionally, hardware roots of trust such as Trusted Platform Modules (TPMs) provide mechanisms for attestation and cryptographic anchoring of devices and platform states—proposals that have been advanced to protect data-at-rest and to secure remote attestation workflows in cloud storage (Patel & Kumar, 2013; Abburu, 2012).

Despite an intensifying body of literature, critical gaps remain. Empirical studies of cross-tenant attacks specifically targeting container orchestration layers are sparse; comparative analyses of trade-offs between micro-virtualization (e.g., Firecracker) and traditional VM isolation for security properties are incomplete; and there is limited synthesis that connects energy-aware scheduling, resource allocation, and security SLAs in a way that cloud data center operators can operationalise (Okonor et al., 2020; Amazon Firecracker, 2020). This article addresses these gaps by integrating existing threads into a coherent design and evaluation framework that illuminates both theoretical implications and concrete architectural choices for secure multi-tenant cloud systems.

METHODOLOGY

This research employs a methodologically plural approach tailored for a conceptual and design-oriented article: literature synthesis, architectural analysis, scenario-based thought experiments, and design pattern extraction. The approach intentionally emphasises rigorous textual synthesis rather than empirical experimentation because the user's input consists of an explicit reference set to be used as the foundation.

The first stage consists of a structured literature synthesis of the provided references to extract core themes, mechanisms, and claims. Each source is interrogated for: definitions of cloud constructs, identified security threats, proposed mitigation techniques, and operational assumptions (Vouk, 2008; Mell & Grance; Ramachandra et al., 2017). This stage maps conceptual building blocks—tenant isolation, cryptographic protections, identity/authorization models, container vs. VM trade-offs, and orchestration constraints—into a consistent taxonomy.

The second stage undertakes comparative architectural analysis. Canonical cloud components (compute, storage, networking, orchestration) are analysed under multiple isolation strategies: hypervisor-based VMs, containerization with kernel sharing, micro-virtualization (e.g., Firecracker), and hardware-enabled partitioning (TPM attestation and similar). The analysis uses a set of evaluation criteria synthesized from the literature: isolation strength, performance overhead, manageability, scalability, attestation feasibility, and compatibility with serverless workloads (Truyen et al., 2016; Amazon Firecracker, 2020; Patel & Kumar, 2013).

The third stage constructs scenario-based thought experiments. These hypothetical but realistic scenarios—such as a multi-tenant SaaS offering hosting confidential customer analytics pipelines or a serverless function chain accessing shared database shards—are used to stress the architecture and reveal practical trade-offs.

Scenarios are grounded in typical cloud service offerings and configurations described in vendor documentation and research overviews (Amazon RDS Multi-AZ, 2020; Azure SQL DB Automatic Tuning, 2020).

The final stage synthesises prescriptive patterns and recommendations into an integrative framework. Each pattern is described in operational detail and linked back to literature evidence to ensure traceability. Limitations of the approach are documented and used to motivate a research agenda for future empirical validation (Ramachandra et al., 2017; Okonor et al., 2020).

This mixed methodological approach prioritises theoretical granularity, architectural clarity, and normative guidance suitable for researchers and practitioners designing secure multi-tenant cloud systems.

RESULTS

The literature synthesis and architectural analysis produced four primary findings that form the backbone of the proposed secure multi-tenant cloud framework.

1. Isolation Must Be Multi-Dimensional and Contextual (Compute, Storage, Network, Control Plane).

Isolation is not a singular property but a set of interdependent guarantees across compute, storage, networking, and control planes. Early cloud analyses emphasised the economic pooling dimension while flagging isolation as a continuous concern (Vouk, 2008). Later work specifically on enforcing multitenancy shows that a combination of software controls (namespaces, cgroups), orchestration policies, and runtime enforcement is required to achieve tenant separation in container architectures (Fiaidhi et al., 2012; Truyen et al., 2016). The result of comparative analysis reveals that no single isolation mechanism suffices: hypervisor VMs offer strong kernel separation but impose higher resource overheads, whereas containers provide fast, dense deployment with weaker hardware separation, and micro-VMs (lightweight VMs) occupy an intermediate point offering a potentially attractive trade-off between isolation and density (Amazon Firecracker, 2020; Truyen et al., 2016).

2. Zero-Trust Principles Must Be Applied as End-to-End Policy, Not Merely at the Edge.

Zero-trust shifts verification from the perimeter to every access decision (Hariharan, 2025). Policies cannot be boiled down to network ACLs or perimeter firewalls alone. The analysis indicates that for multi-tenant clouds, zero-trust must integrate identity lifecycle management, continuous device attestation (including ephemeral runtime attestation for short-lived containers), contextual policy evaluation, and policy enforcement distributed across control and data planes. Implementing continuous verification in ephemeral compute environments is challenging but feasible with tight integration between orchestration, identity providers, and attestation services (Hariharan, 2025; Patel & Kumar, 2013).

3. Data-Centric Protections Anchored in Hardware Roots of Trust Improve Confidentiality Guarantees.

Solutions that decouple encryption from tenant keys and incorporate hardware-based attestation mechanisms (e.g., TPM) provide stronger guarantees against certain classes of attacks such as hypervisor compromise or insider threats (Patel & Kumar, 2013; Abburu, 2012). The literature indicates that TPM-based models can ensure that only vetted platform states can access keys and decrypt tenant data. However, practical deployment complexity and key management remain significant operational hurdles, and adoption requires standardized attestation APIs and orchestration integration (Patel & Kumar, 2013; Abburu, 2012).

4. **Resource Allocation Strategies Must Reconcile Security SLAs with Energy and Performance Objectives.**

Intelligent agent-based allocation strategies can simultaneously address energy efficiency and tenant isolation constraints. Work on intelligent VM allocation suggests that algorithmic orchestration can achieve energy gains while respecting resource quotas and predicted workload variability (Okonor et al., 2020). However, ensuring that allocation does not violate tenant isolation—especially when rapid consolidation or migration is used for energy savings—requires explicit security constraints in scheduling policies, attestation of migration targets, and post-migration verification. This coupling between sustainability goals and security represents both an operational opportunity and a complexity that cloud operators must manage (Okonor et al., 2020).

Each finding is explained in detail below with nuanced analysis and architectural implications.

DISCUSSION

This section interprets the results, explores theoretical and practical tensions, and outlines limitations, counter-arguments, and a research agenda.

Isolation Complexity and Architectural Trade-offs

The recognition that isolation is multi-dimensional compels architects to select and combine mechanisms based on the threat model and workload characteristics. Hypervisor-based VMs historically provided the primary isolation model and remain appropriate for tenants that require strong kernel isolation or for handling untrusted workloads (Vouk, 2008). The VM model, however, carries resource inefficiencies: slower scaling, larger memory footprint, and greater cold-start latencies. Containers and microservices architectures favour developer productivity, fast scaling, and resource efficiency but require compensatory measures to address kernel sharing risks. Micro-VMs (e.g., Firecracker) and minimal-attack-surface hypervisors aim to marry container density with VM-like isolation, but empirical evidence is still emerging about their resilience to sophisticated cross-tenant attacks (Amazon Firecracker, 2020; Truyen et al., 2016). Architects must therefore frame choices in explicit trade-off terms: use hypervisor VMs where tenancy demands absolute isolation; use containers with hardened kernel configurations and runtime policy enforcement for internal, trusted workloads; prefer micro-VMs when both density and improved isolation are required.

Zero-Trust in Ephemeral Environments: Feasibility and Overheads

Applying zero-trust to ephemeral, autoscaling environments introduces practical complications. The zero-trust principle demands continuous verification of identity and device posture for each request (Hariharan, 2025). In serverless architectures, identity tokens for functions must be minted and revoked efficiently; attestation of runtime environment for very short lived functions must be near-instantaneous to avoid prohibitive latency. This requires rethinking identity infrastructure: adopting fast token services, ephemeral keying materials, and hardware attestation that can operate under tight timing constraints (Hariharan, 2025; Patel & Kumar, 2013). There are overheads: increased request latency, more complex key management, and richer telemetry demands. Yet failure to adopt these practices leaves cloud systems vulnerable to token theft, lateral movement, and runtime compromise. Therefore the operational calculus must weigh the incremental latency and complexity against curtailed risk exposure.

Hardware Roots of Trust: Security Benefits and Deployment Challenges

TPM-based attestation and hardware anchored encryption can materially increase guarantees that data-at-rest and data-in-use are only accessible by authorised and verified platform states (Patel & Kumar, 2013; Abburu,

2012). The appeal is particularly strong for regulated industries and high-value data workloads. However, significant barriers impede broad adoption: heterogeneous hardware across data centers, need for standards for attestation and policy expression, and integration complexities with orchestration systems. For cloud providers, incremental deployment often begins with specialised instance types that expose attestation APIs to tenants; wider adoption requires consumer demand, standardisation, and tooling maturity. One must also consider the risk that hardware attestation only provides probabilistic assurances if boot chains or firmware are not comprehensively managed. In short, hardware roots of trust are powerful but not panacean.

Resource Scheduling, Security SLAs, and Energy Efficiency

The literature presents agent-based approaches to VM allocation that can balance energy efficiency and SLA compliance (Okonor et al., 2020). Extending such approaches to multi-tenant contexts entails embedding security constraints into scheduling objectives—i.e., do not co-locate tenants with incompatible isolation requirements, prefer nodes with verified attestation states for sensitive tenants, and avoid consolidation that undermines cryptographic separation. Doing so increases scheduling complexity and may reduce energy savings compared to unconstrained consolidation; nevertheless, it aligns operational behaviour with security commitments. The broader implication is that sustainability initiatives and security obligations are not orthogonal; they must be co-designed.

Counter-arguments and Alternative Views

A plausible counter-argument is that too much isolation and verification produces unusable systems: high latency, gnarly key management burdens, and reduced developer velocity. This is a valid concern and one that pushes toward pragmatic, risk-based approaches. Not every workload requires hardware attestation or immutable, cryptographically-sealed runtime environments. For workloads with low sensitivity, well-configured containers with strong network segmentation and policy enforcement may suffice. Another counter-argument is the potential for attack surface expansion through attestation and identity infrastructure—centralised token services and attestation APIs can become targets. The answer is layered defence: multiple, distributed verification points, diversification of attestation providers, and careful hardening of identity services.

Limitations of the Current Analysis

This article is grounded in literature synthesis and conceptual analysis rather than primary empirical evaluation. While the architectural patterns and trade-offs are theoretically motivated and supported by cited works, the absence of experimental data or benchmark comparisons reduces empirical certitude. The references include vendor documentation and whitepapers (e.g., Amazon and Azure service overviews), which while authoritative on interface semantics, may not candidly report worst-case failure modes; researchers and practitioners should therefore validate recommended patterns with realistic stress tests. Finally, the article relies on the integrity and scope of the provided reference list; missing sources or emergent technologies post-dating the referenced works could further affect the recommendations.

Future Research Directions

The results motivate several practical and theoretical research directions. Empirically, systematic benchmarks comparing hypervisor VMs, containers with hardened kernels, and micro-VMs (like Firecracker) on both performance and security metrics would be valuable. Research should also produce standardised attestation APIs and interoperable tooling for integrating TPM attestation with major orchestration platforms. Another promising avenue is co-design of energy-aware schedulers that treat security as a first-class constraint rather

than an afterthought. Finally, formal modelling of zero-trust policy propagation and verification in dynamic, autoscaling environments could yield provable guarantees about access control under churn.

Architectural Framework and Design Patterns

This section presents a synthesis of concrete design patterns and a coherent architecture that integrates previous findings. The proposed framework is layered: Physical & Hardware Layer, Virtualization & Runtime Layer, Orchestration & Control Layer, Data Protection Layer, and Trust & Identity Layer. Each layer is described with concrete measures, trade-offs, and references to support claims.

Physical & Hardware Layer

Design Goal: Provide hardware capabilities that support attestation, secure boot, and cryptographic key protection.

Measures: Deploy a fleet mix that includes nodes with TPM 2.0 (or equivalent) support and enable secure boot and measured boot procedures. Nodes intended for sensitive tenants should expose attestation endpoints that orchestrators can query before placement decisions (Patel & Kumar, 2013). Hardware features such as memory encryption (where available) can be used to provide additional in-memory confidentiality for tenant data.

Trade-offs: Hardware standardisation is expensive. Rolling out attestation hardware across an entire fleet increases acquisition costs and complicates spare capacity management. The practical approach is staged deployment: create instance types with hardware attestation as premium offerings for tenants with high assurance needs (Patel & Kumar, 2013).

Virtualization & Runtime Layer

Design Goal: Achieve strong isolation for diverse workloads while enabling efficient scaling and fast start times.

Measures: Adopt a stratified virtualization model:

• For untrusted or mixed-risk workloads, use hypervisor VMs to preserve kernel isolation (Vouk, 2008).

• For microservices requiring high density, use hardened containers with kernel hardening, namespaces, seccomp filters, and runtime policy enforcement (Truyen et al., 2016).

• For serverless and short-lived workloads where both speed and stronger isolation are desired, deploy micro-virtualization (e.g., Firecracker) instances that provide a middle ground. Amazon Firecracker demonstrates how minimal VMs can reduce attack surface while maintaining density and fast startup times (Amazon Firecracker, 2020).

Trade-offs: Each mode carries overheads: hypervisor VMs increase resource use; containers reduce isolation; micro-VMs require orchestration support and may be immature in some ecosystems. Decision criteria must include sensitivity of tenant data, expected workload churn, and cost constraints.

Orchestration & Control Layer

Design Goal: Implement policy-driven placement, runtime verification, secure configuration, and lifecycle management.

Measures: Integrate zero-trust policy engines into orchestrators so that placement decisions consider security attestation, identity proofs, and tenant isolation requirements (Hariharan, 2025). Implement policy templates that encode tenant requirements: allowed co-tenants, required attestation state, maximum sharing density, and migration constraints. Orchestration must support continuous monitoring and revocation workflows—if attestation fails post-placement, the system must isolate or evict the workload.

Trade-offs: Orchestration complexity increases, requiring richer telemetry and faster decision loops. This may slow placement and scaling operations; thus orchestration pipelines should be optimised for parallel policy checks and pre-warmed resource pools with known attestation states (Truyen et al., 2016).

Data Protection Layer

Design Goal: Provide data confidentiality, integrity, and auditability across storage and in-transit contexts.

Measures: Adopt data-centric protections:

• Tenant-specific key management with hardware protection: tenant keys stored in hardware-protected modules or derived via attested key exchange protocols so that only verified runtime states can decrypt data (Patel & Kumar, 2013).

• End-to-end encryption for tenant data in transit and at rest, with secure key lifecycle management (Abburu, 2012).

• Shard data to reduce blast radius and enable granular revocation. Sharding also requires enforcing cross-tenant access controls at the storage API layer to prevent privilege escalation or cross-tenant side channels (Ramachandra et al., 2017).

• For analytic workloads, enforce differential access patterns and incorporate query-level enforcement to prevent data exfiltration through aggregated responses.

Trade-offs: Hardware-backed key management complicates multi-region replication and disaster recovery. Sharding increases operational complexity for cross-shard queries and transactional consistency. Consequently, data architecture must be co-designed with application semantics.

Trust & Identity Layer

Design Goal: Ensure continuous verification of entities (users, services, devices) and provide robust authorization mechanisms.

Measures: Implement zero-trust identity flows: short-lived tokens, mutual TLS for service-to-service communication, continuous posture checking, and contextual policy evaluation that considers time, location (logical location within the cloud), and behavior. Provide tenants with the ability to bring their own keys (BYOK) integrated into the provider's hardware attestation model, enabling tenants to retain control over key material while benefiting from provider scale (Patel & Kumar, 2013).

Trade-offs: BYOK and complex attestation integration can become user experience hurdles. Providers must balance security features with ease of use, offering progressive disclosure of complexity: default secure options that work out of the box, and advanced configurations for tenants demanding higher assurance.

Design Patterns and Implementation Recipes

From the layered architecture, extract several repeatable patterns:

Secure Placement Pattern: Use attestation-aware placement for sensitive tenants. Before assigning a node, orchestrator queries attestation service and confirms the measured boot state. If mismatch occurs, reject the node or initiate remediation (Patel & Kumar, 2013).

Ephemeral Credential Pattern: Mint ephemeral credentials for functions and containers that expire quickly and require periodic refresh via strong authentication. Tie credential minting to attestation events to ensure credentials are only provided to verified runtime instances (Hariharan, 2025).

Data Envelope Pattern: Encrypt data with tenant-specific keys; wrap keys in hardware-backed key encryption keys so that decryption is allowed only by nodes that present valid hardware attestation and runtime integrity proof (Abburu, 2012; Patel & Kumar, 2013).

Constrained Consolidation Pattern: Embed tenant isolation constraints as first-class scheduling parameters. Only consolidate tenants whose isolation requirements and trust profiles permit co-location. Use server tagging and placement constraints to enforce policies (Okonor et al., 2020).

Operational Considerations and Governance

Security architecture does not operate in a vacuum; governance, compliance, observability, and cost control are crucial.

Observability and Telemetry

Operational security requires rich telemetry: attestation logs, identity lifecycle events, network flows, and cryptographic key usage metrics. Observability systems must support high-fidelity forensic queries while preserving tenant privacy—telemetry should be partitioned such that tenant-specific logs are accessible only to authorised tenant operators and to provider security teams under strict controls.

Compliance and Auditability

Regulated industries often require demonstrable guarantees about data handling. Hardware attestation provides audit-grade evidence but must be complemented by immutable logging, policy artifacts, and periodic compliance reports. Providers should support audit endpoints and exportable evidence packages that tenants can use for compliance validation (Patel & Kumar, 2013).

Economics and Pricing Models

Security features cost money—attestation hardware, hardened nodes, premium network isolation, and security-aware scheduling all raise operating costs. Cloud providers must design pricing models that expose these options as purchasable features: e.g., premium attested nodes, secure storage tiers with hardware key protection, and managed BYOK services. Tenants can then choose assurance levels appropriate to their data sensitivity.

Human Factors and Developer Experience

To ensure adoption, security must be accessible. That means providing secure defaults, clear SDKs for ephemeral credential patterns, managed services that hide attestation complexity when not required, and educational resources that explain trade-offs. Overly complex APIs for attestation and key management will hinder adoption, particularly among small and medium enterprises.

CONCLUSION

The rapid transition to containerized, serverless, and highly managed cloud services necessitates a rethinking of secure multi-tenant architectures. This article synthesises core literature and vendor practices to propose an integrative framework that couples multi-dimensional isolation, zero-trust continuous verification, hardware-anchored data protection, and security-aware resource orchestration. The key messages are fourfold: isolation must be considered across compute, storage, network, and control planes; zero-trust must be operationalised for ephemeral workloads; hardware roots of trust can materially elevate confidentiality guarantees but demand operational investment; and schedulers must embed security constraints alongside energy and performance objectives.

The framework is intentionally prescriptive: it offers concrete patterns and layering that practitioners can adapt. Yet it also recognises trade-offs—performance overheads, integration complexity, and economic cost—and therefore advocates for risk-based adoption where high assurance tenants receive premium protections and lower-sensitivity workloads benefit from hardened defaults.

Future work should empirically validate the relative security and performance of micro-VMs, containers with hardened kernels, and traditional VMs under adversarial conditions. Standardising attestation APIs and integrating them into mainstream orchestration tooling would significantly lower the barrier to adoption. Finally, developing scheduling algorithms that reconcile sustainability goals with security SLAs presents a pressing and societally significant research direction.

REFERENCES

1. M.A. Vouk, (2008) “Cloud computing–issues, research and implementations”, CIT. Journal of Computing and Information Technology, Vol. 16, No. 4, pp235-246.

2. Patel & M. Kumar, (2013) “A Proposed Model for Data Security of Cloud Storage Using Trusted Platform Module”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 4.

3. D.P.D.S. Abburu, (2012). “An Approach for Data Storage Security in Cloud Computing”, IJCSI International Journal of Computer Science Issues, Vol. 9, No. 2.

4. Hariharan, R. (2025). Zero trust security in multi-tenant cloud environments. Journal of Information Systems Engineering and Management, 10.

5. G. Ramachandra, M. Iftikhar, and F. A. Khan, ‘A Comprehensive Survey on Security in Cloud Computing’, Procedia Comput. Sci., vol. 110, pp. 465–472, 2017, doi: 10.1016/j.procs.2017.06.124.

6. P. Mell and T. Grance, ‘The NIST Definition of Cloud Computing’, p. 7.

7. E. Truyen, D. Van Landuyt, V. Reniers, A. Rafique, B. Lagaisse, and W. Joosen, ‘Towards a containerbased architecture for multi-tenant SaaS applications’, in Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware - ARM 2016, Trento, Italy, 2016, pp. 1–6. doi: 10.1145/3008167.3008173.

8. J. Fiaidhi, I. Bojanova, J. Zhang, and L.-J. Zhang, ‘Enforcing Multitenancy for Cloud Computing Environments’, IT Prof., vol. 14, no. 1, pp. 16–18, Jan. 2012, doi: 10.1109/MITP.2012.6.

9. O. M. Okonor, M. Adda, and A. Gegov, ‘Intelligent Agent-based Technique For Virtual Machine Resource

Allocation For Energy-Efficient Cloud Data Centres', WSEAS Trans. Commun., vol. 19, pp. 37–46, Apr. 2020, doi: 10.37394/23204.2020.19.5.

10. Amazon Athena. (2020). Retrieved from https://aws.amazon.com/athena/

11. Amazon Firecracker. (2020). Retrieved from https://aws.amazon.com/about-aws/whatsnew/2018/11/firecracker-lightweight-virtualization-for-serverless-computing/

12. Amazon RDS Multi-AZ. (2020). Retrieved from https://aws.amazon.com/rds/features/multi-az/

13. AWS Redshift. (2020). Retrieved from https://aws.amazon.com/redshift/

14. Amazon Aurora Serverless. (2020). Retrieved from https://aws.amazon.com/rds/aurora/serverless/

15. Apache Hadoop. (2020). Retrieved from http://hadoop.apache.org

16. A Technical Overview of Azure Cosmos DB. (2020). Retrieved from https://azure.microsoft.com/en-us/blog/a-technical-overview-of-azure-cosmos-db/

17. Azure SQL DB Automatic Tuning. (2020). Retrieved from https://docs.microsoft.com/en-us/sql/relational-databases/automatic-tuning/automatic-tuning