

---

## | RESEARCH ARTICLE

### Multi-Cloud Interoperability: Integrating Microsoft Azure with AWS and Google Cloud

**Shekar Rao Lakavath**

*Independent Researcher, Bonneylake, Washington, USA*

**Corresponding Author:** Shekar Rao Lakavath, **E-mail:** [Shekar.lakavath@gmail.com](mailto:Shekar.lakavath@gmail.com)

---

#### | ABSTRACT

As organizations increasingly distribute workloads across more than one public cloud provider to avoid vendor lock-in, meet regulatory requirements, or exploit provider-specific strengths, the technical and governance challenge of multi-cloud interoperability has become a central concern for enterprise cloud architecture. Microsoft Azure, Amazon Web Services (AWS), and Google Cloud Platform each expose distinct application programming interfaces, identity models, and infrastructure-as-code tooling, creating friction whenever an organization needs workloads, identities, or data to move consistently across these environments. This article examines how Azure can be integrated with AWS and Google Cloud to achieve practical multi-cloud interoperability. The article proposes a four-layer interoperability architecture spanning unified governance, identity federation, workload portability, and network connectivity, compares Azure's native multi-cloud tooling against equivalent AWS and Google Cloud capabilities, and maps common interoperability challenges to specific mitigation approaches grounded in the literature. The findings indicate that container-based workload portability and standardized identity federation protocols provide the most mature and broadly supported interoperability mechanisms across all three providers, while infrastructure-as-code and service-level-agreement normalization remain comparatively immature and require deliberate architectural investment. The article concludes with practical recommendations for organizations planning or operating a multi-cloud environment spanning Azure, AWS, and Google Cloud.

#### | KEYWORDS

Multi-cloud, cloud interoperability, Microsoft Azure, AWS, Google Cloud Platform, vendor lock-in, cloud portability, identity federation

#### | ARTICLE INFORMATION

**ACCEPTED:** August 19, 2024    **PUBLISHED:** November 25, 2024    **DOI:** <https://doi.org/10.61424/ijans.v2i2.1075>

---

#### 1. Introduction

The early years of cloud computing were characterized by organizations selecting a single public cloud provider and building their infrastructure and applications around that provider's specific services and application programming interfaces. As cloud adoption matured, this single-provider model exposed a structural risk: an organization that builds deeply around one provider's proprietary services faces substantial switching costs if it later needs to change providers, negotiate better terms, or comply with data-residency requirements that a single provider cannot satisfy across all relevant jurisdictions (Opara-Martins, Sahandi, & Tian, 2016).

Multi-cloud architecture, in which an organization deliberately operates workloads across two or more public cloud providers, has emerged as a response to this risk. Zhang, Wu, and Cheung (2013) were among the first to systematically survey the taxonomies and standards relevant to cloud interoperability, defining interoperability as

the ability of multiple cloud providers to work together and highlighting that most practical interoperability efforts at the time focused narrowly on infrastructure-as-a-service compatibility rather than the full application stack. In the years since, interoperability research has expanded to address identity federation, workload portability, service discovery, and unified governance across providers (Kaur, Sharma, & Kahlon, 2017).

Microsoft Azure, AWS, and Google Cloud Platform remain the three dominant public cloud providers, and organizations increasingly need to integrate workloads, identities, and governance across some combination of the three. This article addresses three questions. First, what categories of interoperability challenge does the multi-cloud research literature identify as most persistent, and how have proposed standards and tools addressed them? Second, how do Azure's native multi-cloud capabilities compare against equivalent tooling in AWS and Google Cloud? Third, what specific architectural approach and mitigation strategy addresses each major interoperability challenge in a practical Azure-AWS-Google Cloud deployment? The remainder of the article reviews the relevant literature, proposes a four-layer multi-cloud interoperability architecture, compares native tooling across the three providers, maps interoperability challenges to mitigations, and discusses practical implications.

## **2. Literature Review**

### **2.1 Foundations of Cloud Computing and Interoperability**

Armbrust et al. (2010) provided one of the most widely cited foundational descriptions of cloud computing, characterizing it as the combination of software-as-a-service applications and the data-center hardware and software that provide those services, and identifying the difficulty of migrating applications between providers as a specific obstacle to broader cloud adoption. This early observation anticipated the interoperability research agenda that followed. Zhang, Wu, and Cheung (2013) subsequently produced a comprehensive survey of cloud interoperability taxonomies and standards, concluding that interoperability efforts fall into distinct client-centric approaches, where the consuming organization builds abstraction layers over multiple providers, and provider-centric approaches, where providers adopt common standards directly.

Kaur, Sharma, and Kahlon (2017) extended this taxonomy with a comprehensive review of interoperability and portability approaches in interconnected clouds, finding that research interest in the topic increased substantially between 2013 and 2016 as organizations moved from single-provider deployments toward more distributed cloud strategies. Nogueira, Moreira, Lucrédio, and Garcia (2016) conducted a systematic literature review specifically focused on the challenges of developing interoperable cloud applications, distinguishing between interoperability, which concerns the ability of systems to exchange and use information, and portability, which concerns the ability to move an application or its data from one environment to another without substantial modification, a distinction that remains central to the architecture proposed later in this article.

### **2.2 Multi-Cloud Architectures and Brokering**

As organizations began operating across multiple providers simultaneously, researchers proposed brokering and orchestration architectures to manage the resulting complexity. Petcu (2013) characterized the emerging multi-cloud research agenda, distinguishing between multi-cloud architectures in which an application is deliberately distributed across providers and the more limited case of an application simply being portable between providers without necessarily running on more than one simultaneously. Grozev and Buyya (2014) provided a taxonomy of inter-cloud architectures and application brokering approaches, categorizing brokering strategies according to whether they operate at the infrastructure, platform, or application level, and finding that most mature brokering tools at the time operated primarily at the infrastructure level.

Jrad, Tao, and Streit (2013) proposed a broker-based framework specifically for multi-cloud workflows, demonstrating how a broker component could select the most appropriate provider for each stage of a distributed workflow based on cost and performance criteria. Toosi, Calheiros, and Buyya (2014) surveyed interconnected cloud computing environments more broadly, proposing a taxonomy that distinguishes federation, in which providers cooperate directly, from multi-cloud brokering, in which a third-party or client-side component mediates across otherwise non-cooperating providers, a distinction directly relevant to whether an organization can rely on native provider tooling such as Azure Arc or must build its own cross-cloud abstraction layer.

Saxena, Gupta, Singh, and Lee (2021) conducted a more recent survey and comparative study of multi-cloud architectures, concluding that despite a decade of research progress, cloud federation remains in what they characterize as a maturing phase, with configuration complexity, security consistency, and unified management still representing open challenges rather than solved problems. This assessment is consistent with the persistent gap this article identifies between the maturity of container-based workload portability and the comparative immaturity of unified governance tooling across providers.

### **2.3 Portability Standards and Vendor Lock-In**

A parallel research stream has focused on standardizing how cloud application topologies and deployments are described, so that a single specification can be deployed across multiple providers. Binz, Breitenbücher, Kopp, and Leymann (2014) described the Topology and Orchestration Specification for Cloud Applications (TOSCA), a standard for specifying cloud application topologies and their deployment and orchestration in a provider-agnostic manner. Di Martino, Cretella, and Esposito (2015) classified cloud interoperability and portability use cases and definitions, providing a structured basis for evaluating which categories of application are best suited to TOSCA-based or similar standardized deployment descriptors. Loutas, Kamateri, and Tarabanis (2011) proposed a semantic interoperability framework specifically for platform-as-a-service offerings, addressing the additional complexity that arises when, unlike infrastructure-as-a-service, the platform layer itself varies substantially in the abstractions and programming models each provider exposes.

Petcu, Macariu, Panica, and Craciun (2013) examined the practical challenges of achieving cloud application portability, moving from theoretical standards toward concrete implementation guidance, and concluding that containerization was emerging as the most practically effective portability mechanism at the application layer, a conclusion that has since been strongly validated by the widespread adoption of Kubernetes as a portable container orchestration layer across all major cloud providers (Pahl, 2015).

Vendor lock-in, the condition in which switching costs from proprietary services make changing providers impractical, has been studied both as a technical and a business problem. Opara-Martins, Sahandi, and Tian (2016) conducted a business-perspective critical analysis of vendor lock-in based on a survey of 114 organizations, finding that lock-in risk is exacerbated as workloads migrate from on-premises to cloud infrastructure and identifying contractual and architectural strategies, including preferring providers that support standardized data formats and APIs, as practical mitigations. Satzger, Hummer, Inzinger, Leitner, and Dustdar (2013) proposed the concept of a meta-cloud layer that abstracts over individual provider APIs, arguing that such an abstraction layer, rather than any single provider's roadmap, should be the basis for avoiding long-term lock-in.

### **2.4 Service Description, Comparison, and Discovery**

Comparing services meaningfully across providers has proven to be a persistent challenge, given that providers rarely expose directly comparable service specifications. Li, Yang, Kandula, and Zhang (2010) proposed CloudCmp, an empirical benchmarking framework for comparing the performance and cost of public cloud providers along standardized metrics, providing one of the earliest rigorous methodologies for cross-provider comparison. Jula, Sundararajan, and Othman (2014) conducted a systematic literature review of cloud computing service composition, finding that most service composition research assumed a single-provider context and identifying multi-provider service composition as a comparatively underexplored area at the time of their review.

### **2.5 Identity, Governance, and Hybrid Architectures**

Effective multi-cloud operation depends on consistent identity and governance mechanisms spanning providers. Kertész, Kecskeméti, and Brandic (2014) proposed a self-adaptive service virtualization approach for enforcing consistent service-level-agreement guarantees across heterogeneous cloud environments, addressing the problem that each provider defines and measures service-level commitments differently. Ferry, Rossini, Chauvel, Morin, and Solberg (2013) proposed a model-driven approach to multi-cloud provisioning, deployment, monitoring, and adaptation, demonstrating that declarative, model-based specifications could reduce the manual configuration burden of operating consistent policies across multiple cloud accounts.

Gonzalez et al. (2012) conducted a quantitative analysis of security concerns across cloud computing environments, finding that identity and access management inconsistencies represented one of the most frequently cited barriers to secure multi-provider operation, a finding that motivates the identity-federation layer proposed in the architecture that follows. Aryotejo, Kristiyanto, and Mufadhol (2018) examined hybrid cloud architectures that bridge private and public cloud infrastructure, finding that the governance and connectivity patterns developed for hybrid cloud, such as private network interconnection and federated identity, transfer directly to the multi-public-cloud interoperability problem addressed in this article.

Taken together, this literature indicates that multi-cloud interoperability is not a single problem but a set of related challenges operating at different layers: governance and policy, identity, application and data portability, and network connectivity. The following sections propose an architecture that addresses each layer explicitly in the specific context of integrating Azure with AWS and Google Cloud.

### 3. A Multi-Cloud Interoperability Architecture for Azure, AWS, and Google Cloud

Building on the layered interoperability challenges identified above (Toosi, Calheiros, & Buyya, 2014; Nogueira, Moreira, Lucrédio, & Garcia, 2016), this article proposes a four-layer architecture for integrating Azure with AWS and Google Cloud. Figure 1 illustrates the architecture, spanning a unified governance layer, identity federation, workload portability, and network connectivity, sitting above the three underlying cloud providers.

Figure 1. Multi-Cloud Interoperability Architecture Integrating Azure, AWS, and Google Cloud

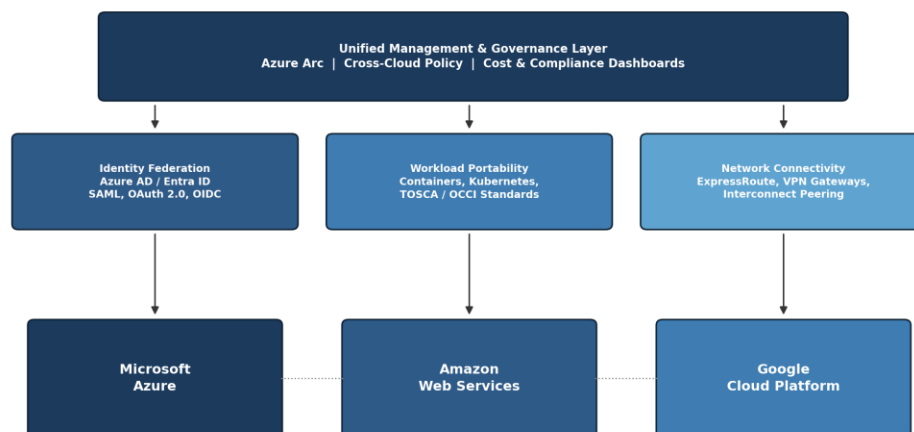


Figure 1. Multi-cloud interoperability architecture integrating Azure, AWS, and Google Cloud, showing the unified governance layer, the three interoperability-enabling layers (identity federation, workload portability, network connectivity), and the underlying cloud providers they connect.

#### 3.1 Unified Governance Layer

Azure Arc extends Azure's policy, role-based access control, and monitoring capabilities to resources running on AWS, Google Cloud, and on-premises infrastructure, providing the kind of unified control plane that Saxena, Gupta, Singh, and Lee (2021) identify as still comparatively immature across the industry. This layer allows an organization to apply consistent governance and cost-visibility policy regardless of which underlying provider hosts a given resource.

#### 3.2 Identity Federation Layer

Azure Active Directory (Entra ID) federates identity with AWS IAM Identity Center and Google Cloud Identity through standard SAML, OAuth 2.0, and OpenID Connect protocols, addressing the identity-consistency barrier

documented by Gonzalez et al. (2012). This allows a single set of user credentials and access policies to govern resources across all three providers rather than requiring separately managed identity stores.

### 3.3 Workload Portability Layer

Containerized workloads orchestrated through Kubernetes, available natively as Azure Kubernetes Service, Amazon EKS, and Google GKE, provide the most mature portability mechanism identified in the literature (Pahl, 2015; Petcu, Macariu, Panica, & Craciun, 2013). Where declarative infrastructure specification is required beyond the container layer, TOSCA-based or Terraform-based infrastructure-as-code provides a provider-agnostic alternative to provider-specific templates such as Azure Bicep or AWS CloudFormation, consistent with the portability standard proposed by Binz, Breitenbücher, Kopp, and Leymann (2014).

### 3.4 Network Connectivity Layer

Private network interconnection between providers, using Azure ExpressRoute, AWS Direct Connect, and Google Cloud Interconnect, allows workloads distributed across providers to communicate without traversing the public internet, following the hybrid connectivity patterns documented by Aryotejo, Kristiyanto, and Mufadhol (2018) in the context of private-public hybrid cloud architecture.

## 4. Comparing Native Multi-Cloud Tooling Across Azure, AWS, and Google Cloud

Table 1 compares Azure's native capability at each interoperability dimension identified above against the equivalent tooling available in AWS and Google Cloud.

**Table 1. Native Multi-Cloud Tooling Comparison Across Azure, AWS, and Google Cloud**

| Interoperability Dimension         | Azure Native Capability                                                                                               | Equivalent in AWS / Google Cloud                                                                                                    |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Hybrid & multi-cloud control plane | Azure Arc extends governance, policy, and monitoring to resources running on AWS, GCP, and on-premises infrastructure | AWS: no direct equivalent for managing non-AWS resources; Google: Anthos extends GKE-based management to AWS and on-premises        |
| Identity federation                | Azure AD (Entra ID) supports SAML, OAuth 2.0, and OIDC federation with external identity providers                    | AWS IAM Identity Center and Google Cloud Identity both support SAML/OIDC federation but require separate configuration per platform |
| Workload portability               | Azure Kubernetes Service runs standard, unmodified Kubernetes workloads                                               | Amazon EKS and Google GKE also run standard Kubernetes, enabling container-based portability across all three providers             |
| Network interconnection            | Azure ExpressRoute and VPN Gateway provide private connectivity to AWS Direct Connect and Google Cloud Interconnect   | AWS Direct Connect and Google Cloud Interconnect provide symmetric connectivity options back to Azure                               |
| Infrastructure-as-code portability | Bicep and ARM templates are Azure-specific; Terraform provides a cloud-agnostic alternative                           | AWS CloudFormation and Google Deployment Manager are similarly provider-specific; Terraform remains                                 |

|                            |                                                                                                                     |                                                                                                              |
|----------------------------|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
|                            | supported across all three providers                                                                                | the common cross-cloud option                                                                                |
| Data portability standards | Supports open formats (Parquet, Delta Lake) and OCCI/TOSCA-aligned deployment descriptors for cross-cloud migration | AWS and Google Cloud support the same open data formats, though native storage APIs remain provider-specific |

As Table 1 indicates, container orchestration through Kubernetes is the interoperability dimension with the most symmetric support across all three providers, consistent with the portability conclusions of Pahl (2015), while unified cross-cloud governance remains asymmetric: Azure Arc and Google Anthos both extend management beyond their home provider, but AWS currently offers no direct equivalent for managing non-AWS resources, a gap organizations must address through third-party tooling or custom governance automation.

### 5. Mapping Interoperability Challenges to Mitigation Approaches

Table 2 synthesizes the interoperability challenges most frequently documented in the literature reviewed above and maps each to a specific mitigation approach, along with representative supporting sources.

**Table 2. Multi-Cloud Interoperability Challenges and Mitigation Approaches**

| Multi-Cloud Interoperability Challenge                                    | Mitigation Approach                                                                                                   | Supporting Literature                                                              |
|---------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Vendor lock-in from proprietary APIs and service models                   | Adopt cloud-agnostic infrastructure-as-code (e.g., Terraform) and open orchestration standards such as TOSCA          | Opara-Martins, Sahandi, & Tian (2016); Binz, Breitenbücher, Kopp, & Leymann (2014) |
| Inconsistent identity and access models across providers                  | Federate identity through SAML/OIDC-based single sign-on rather than maintaining separate credential stores per cloud | Zhang, Wu, & Cheung (2013); Kaur, Sharma, & Kahlon (2017)                          |
| Fragmented service discovery and comparison across providers              | Use semantic service description frameworks and cloud service brokers to compare and select services across providers | Loutas, Kamateri, & Tarabanis (2011); Li, Yang, Kandula, & Zhang (2010)            |
| Difficulty portably deploying applications across heterogeneous platforms | Containerize workloads using Kubernetes, which runs consistently across Azure, AWS, and Google Cloud                  | Pahl (2015); Petcu, Macariu, Panica, & Craciun (2013)                              |
| Inconsistent SLA definitions and enforcement across providers             | Apply SLA-mapping and self-adaptive service virtualization frameworks that normalize provider-specific guarantees     | Kertész, Kecskeméti, & Brandic (2014)                                              |
| Manual, error-prone                                                       | Adopt model-driven, declarative                                                                                       | Ferry, Rossini, Chauvel, Morin,                                                    |

|                                                                    |                                                                                                                     |                                                                      |
|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| provisioning across multiple cloud accounts                        | multi-cloud provisioning and monitoring tools                                                                       | & Solberg (2013); Jrad, Tao, & Streit (2013)                         |
| Absence of unified governance and cost visibility across providers | Deploy a unified control-plane layer such as Azure Arc to apply consistent policy and cost governance across clouds | Saxena, Gupta, Singh, & Lee (2021); Toosi, Calheiros, & Buyya (2014) |

The mapping in Table 2 illustrates that most mitigation approaches trade a degree of provider-specific optimization for portability; adopting Terraform instead of Bicep, for example, sacrifices some Azure-specific convenience features in exchange for a single infrastructure specification that also targets AWS and Google Cloud, echoing the abstraction-layer tradeoff that Satzger, Hummer, Inzinger, Leitner, and Dustdar (2013) describe in their meta-cloud proposal.

## 6. Discussion

The analysis presented above supports several practical conclusions for organizations integrating Azure with AWS and Google Cloud. First, container-based workload portability is the most mature interoperability mechanism available today; because Kubernetes exposes a consistent API regardless of the underlying provider, workloads can move between Azure Kubernetes Service, Amazon EKS, and Google GKE with comparatively little modification, validating the portability-through-containerization conclusion reached by Petcu, Macariu, Panica, and Craciun (2013) more than a decade ago.

Second, identity federation through open standards such as SAML and OpenID Connect provides a more durable foundation for multi-cloud identity than any provider-specific identity service used in isolation. Given Gonzalez et al.'s (2012) finding that identity and access management inconsistency is a leading barrier to secure multi-cloud operation, organizations should treat identity federation configuration as a foundational architectural decision rather than an afterthought layered on once workloads are already distributed across providers.

Third, unified governance remains the least mature layer of the architecture proposed here. While Azure Arc and Google Anthos both extend native management beyond their home provider, no single tool today provides fully symmetric governance across all three major providers, consistent with Saxena, Gupta, Singh, and Lee's (2021) characterization of cloud federation as still maturing. Organizations should expect to supplement native tooling with custom automation or third-party cloud management platforms to achieve fully consistent policy enforcement across Azure, AWS, and Google Cloud.

Fourth, the choice between provider-specific and cloud-agnostic infrastructure-as-code involves a genuine tradeoff rather than a straightforward best practice. Provider-specific templates such as Azure Bicep typically expose newly released features more quickly and with better first-party support, while cloud-agnostic tools such as Terraform, aligned with the TOSCA portability philosophy described by Binz, Breitenbücher, Kopp, and Leymann (2014), provide a single specification that reduces the operational burden of maintaining separate provisioning logic per provider. Organizations should make this choice deliberately based on how much of their infrastructure genuinely needs cross-provider portability versus how much is intentionally Azure-specific.

Finally, service comparison and selection across providers remains methodologically difficult. The benchmarking approach pioneered by Li, Yang, Kandula, and Zhang (2010) with CloudCmp remains a useful template: organizations should define standardized, workload-representative benchmarks rather than relying solely on vendor-published specifications when deciding which provider should host a given workload within a multi-cloud architecture.

## 7. Recommendations

Organizations integrating Azure with AWS and Google Cloud should: (1) standardize on Kubernetes for containerized workloads to maximize portability across all three providers, following the portability conclusions of Pahl (2015) and Petcu, Macariu, Panica, and Craciun (2013); (2) federate identity through SAML or OpenID Connect from the outset of multi-cloud planning rather than retrofitting federation after provider-specific identity silos are already established; (3) deploy Azure Arc, or an equivalent third-party governance platform, to achieve consistent policy and cost visibility across providers, recognizing the governance gap documented by Saxena, Gupta, Singh, and Lee (2021); (4) adopt cloud-agnostic infrastructure-as-code such as Terraform for workloads that genuinely require cross-provider portability, reserving provider-specific templates for intentionally single-provider components; (5) establish private network interconnection between providers using ExpressRoute, Direct Connect, and Cloud Interconnect rather than routing inter-cloud traffic over the public internet; and (6) apply the same disciplined, phased project-governance approach recommended for large infrastructure programs in other domains, including data-driven resource allocation and earned-value cost tracking (Agumalu, 2021, 2022), when sequencing a multi-cloud migration or integration effort, so that interoperability investments are prioritized according to documented risk and business value rather than implemented reactively.

## 8. Conclusion

Multi-cloud interoperability has evolved from an academic taxonomy problem into a practical architectural requirement for organizations seeking to avoid vendor lock-in, meet regulatory data-residency obligations, and exploit the comparative strengths of different cloud providers. This article has proposed a four-layer architecture integrating Microsoft Azure with AWS and Google Cloud, spanning unified governance, identity federation, workload portability, and network connectivity, and has compared native tooling and mapped common interoperability challenges to specific mitigations across all three providers. The literature reviewed indicates that container-based portability and open identity federation standards represent the most mature interoperability mechanisms available today, while unified cross-cloud governance and service-level-agreement normalization remain comparatively immature, requiring organizations to supplement native provider tooling with deliberate architectural investment and disciplined project governance. As in the broader project and infrastructure governance literature (Adenike, 2018, 2022; Agumalu, 2021, 2022), technology alone does not resolve the interoperability challenge; well-sequenced, well-governed implementation determines whether a multi-cloud architecture delivers its intended flexibility and resilience benefits in practice. Future research should empirically evaluate the operational cost and reliability outcomes of production multi-cloud deployments spanning Azure, AWS, and Google Cloud to validate the architectural tradeoffs identified here.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

**Publisher's Note:** All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers

**Artificial Intelligence (AI) Use Disclosure:** The authors declare that no artificial intelligence tools were used in the preparation of this manuscript.

## References

- [1] Adenike, A. (2018). Strengthening financial market infrastructure in emerging economies: Lessons from central banking operations. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, 10(02), 177–182. <https://doi.org/10.18090/samriddhi.v10i02.12>
- [2] Adenike, A. (2022). Central bank digital currency (CBDC) adoption challenges and remittance impacts: Empirical evidence from Nigeria's eNaira compared with global retail pilots and US Federal Reserve policy stance. *International Journal of Humanities and Information Technology*, 4(01-03), 214–233. <https://doi.org/10.21590/ijhit.04.01-3.16>
- [3] Agumalu, S. A. (2021). Project management strategies for improving NGO operational efficiency in Eastern Nigeria: A data-driven assessment of resource allocation, reporting, and project planning practices. *International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)*, 4(1), 4351–4360. <https://doi.org/10.15662/IJRPETM.2021.0501005>

- [4] Agumalu, S. A. (2022). A predictive cost-management and benefits realization model for international projects using earned value management (EVM) to reduce cost overruns and improve project outcomes. *International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)*, 5(2), 6517–6527.
- [5] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50–58. <https://doi.org/10.1145/1721654.1721672>
- [6] Aryotejo, G., Kristiyanto, D. Y., & Mufadhol, M. (2018). Hybrid cloud: Bridging of private and public cloud computing. *Journal of Physics: Conference Series*, 1025, 012091. <https://doi.org/10.1088/1742-6596/1025/1/012091>
- [7] Binz, T., Breitenbücher, U., Kopp, O., & Leymann, F. (2014). TOSCA: Portable automated deployment and management of cloud applications. In A. Bouguettaya, Q. Z. Sheng, & F. Daniel (Eds.), *Advanced Web Services* (pp. 527–549). Springer. [https://doi.org/10.1007/978-1-4614-7535-4\\_22](https://doi.org/10.1007/978-1-4614-7535-4_22)
- [8] Di Martino, B., Cretella, G., & Esposito, A. (2015). Classification and positioning of cloud definitions and use case scenarios for portability and interoperability. In *2015 3rd International Conference on Future Internet of Things and Cloud* (pp. 538–544). IEEE. <https://doi.org/10.1109/FiCloud.2015.119>
- [9] Ferry, N., Rossini, A., Chauvel, F., Morin, B., & Solberg, A. (2013). Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In *2013 IEEE Sixth International Conference on Cloud Computing* (pp. 887–894). IEEE. <https://doi.org/10.1109/CLOUD.2013.133>
- [10] Gonzalez, N., Miers, C., Redigolo, F., Simplicio, M., Carvalho, T., Näslund, M., & Pourzandi, M. (2012). A quantitative analysis of current security concerns and solutions for cloud computing. *Journal of Cloud Computing*, 1, 11. <https://doi.org/10.1186/2192-113X-1-11>
- [11] Grozev, N., & Buyya, R. (2014). Inter-cloud architectures and application brokering: Taxonomy and survey. *Software: Practice and Experience*, 44(3), 369–390. <https://doi.org/10.1002/spe.2168>
- [12] Jrad, F., Tao, J., & Streit, A. (2013). A broker-based framework for multi-cloud workflows. In *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds* (pp. 61–68). ACM. <https://doi.org/10.1145/2462326.2462339>
- [13] Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications*, 41(8), 3809–3824. <https://doi.org/10.1016/j.eswa.2013.12.017>
- [14] Kaur, K., Sharma, S., & Kahlon, K. S. (2017). Interoperability and portability approaches in inter-connected clouds: A review. *ACM Computing Surveys*, 50(4), 49. <https://doi.org/10.1145/3092698>
- [15] Kertész, A., Kecskeméti, G., & Brandic, I. (2014). An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous cloud environments. *Future Generation Computer Systems*, 32, 54–68. <https://doi.org/10.1016/j.future.2012.05.016>
- [16] Li, A., Yang, X., Kandula, S., & Zhang, M. (2010). CloudCmp: Comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (pp. 1–14). ACM. <https://doi.org/10.1145/1879141.1879143>
- [17] Loutas, N., Kamateri, E., & Tarabanis, K. (2011). A semantic interoperability framework for cloud platform as a service. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science* (pp. 388–395). IEEE. <https://doi.org/10.1109/CloudCom.2011.60>
- [18] Nogueira, E., Moreira, A., Lucrédio, D., & Garcia, V. (2016). Issues on developing interoperable cloud applications: Definitions, concepts, approaches, requirements, characteristics and evaluation models. *Journal of Software Engineering Research and Development*, 4, 7. <https://doi.org/10.1186/s40411-016-0033-6>
- [19] Opara-Martins, J., Sahandi, R., & Tian, F. (2016). Critical analysis of vendor lock-in and its impact on cloud computing migration: A business perspective. *Journal of Cloud Computing*, 5, 4. <https://doi.org/10.1186/s13677-016-0054-z>
- [20] Pahl, C. (2015). Containerization and the PaaS cloud. *IEEE Cloud Computing*, 2(3), 24–31. <https://doi.org/10.1109/MCC.2015.51>
- [21] Petcu, D. (2013). Multi-cloud: Expectations and current approaches. In *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds* (pp. 1–6). ACM. <https://doi.org/10.1145/2462326.2462328>
- [22] Petcu, D., Macariu, G., Panica, S., & Craciun, C. (2013). Portable cloud applications: From theory to practice. *Future Generation Computer Systems*, 29(6), 1417–1430. <https://doi.org/10.1016/j.future.2012.01.009>
- [23] Satzger, B., Hummer, W., Inzinger, C., Leitner, P., & Dustdar, S. (2013). Winds of change: From vendor lock-in to the meta cloud. *IEEE Internet Computing*, 17(1), 69–73. <https://doi.org/10.1109/MIC.2013.19>
- [24] Saxena, D., Gupta, I., Singh, A. K., & Lee, C.-N. (2021). A survey and comparative study on multi-cloud architectures: Emerging issues and challenges for cloud federation (arXiv:2108.12831). arXiv. <https://doi.org/10.48550/arXiv.2108.12831>
- [25] Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys*, 47(1), 7. <https://doi.org/10.1145/2593512>
- [26] Zhang, Z., Wu, C., & Cheung, D. W. (2013). A survey on cloud interoperability: Taxonomies, standards, and practice. *ACM SIGMETRICS Performance Evaluation Review*, 40(4), 13–22. <https://doi.org/10.1145/2479942.2479945>