International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Cloud-Native Application Refactoring Using Azure and OpenShift: A Comparative Study

Ramadevi Nunna *

Vijayanagara Sri Kirshnadevaraya University, Bellary, Karnataka.

International Journal of Science and Research Archive, 2026, 18(01), 371-382

Publication history: Received on 30 November 2025; revised on 04 January 2026; accepted on 09 January 2026

Article DOI: https://doi.org/10.30574/ijsra.2026.18.1.3369

Abstract

The shift from monolithic systems to cloud-native architectures has become a critical priority for modern enterprises seeking greater scalability, agility, and operational efficiency. Cloud-native principles emphasize re-architecting legacy applications to align them with contemporary infrastructure standards, incorporating container orchestration, microservice-based designs, and modern deployment practices. The paper provides a thorough comparative analysis of two prominent platforms, Microsoft Azure and Red Hat OpenShift, and how they are effective in enabling cloud-native refactoring programs. It interrogates their architectural designs, service platform, DevOps implementation, security, cost platform, and scalability. Azure has the advantages of managed services and extensive integration with the Microsoft ecosystem, whereas open-source OpenStack has more control, flexibility of deployment to be hybrid, and is well-aligned with open-source technologies. The results are summed up with strategic recommendations on which platform organizations should choose in order to meet the needs of cloud-native modernization and address their long-term objectives.

Keywords: Cloud-native refactoring; Azure; OpenShift; Kubernetes; Microservices

1. Introduction

The software development paradigm shift from monolithic architectures to micro services has increased the shift to cloud-based applications in industries. Such applications are also tailor-crafted to scale and bend to take advantage of the scaling, flexibility, and resiliency provided by the current cloud computing systems. Moving the traditional applications to the cloud-native format will require breaking down tightly integrated systems into free-standing, loosely coupled services, which are more easily scaled and managed. The new cloud-native technologies of containers, Kubernetes, and continuous integration/continuous deployment (CI/CD) pipelines have changed how enterprises design, deploy, and operate applications in the cloud [1][2][3].

Two of the largest platforms that allow cloud-native transformation include Microsoft Azure and Red Hat OpenShift. Azure gives a whole range of services and infrastructure to host, deploy, and scale cloud-native applications, whereas OpenShift, which runs on Kubernetes, is an efficient container orchestration and application development platform. The two platforms are different in terms of methodologies, services, and toolchains used to refactor legacy apps into cloud-native apps [4][5][6]. Although Azure is tightly connected with the rest of the Microsoft environment and offers platform-as-a-service (PaaS), OpenShift is more flexible and controllable and is frequently favored by businesses in need of hybrid and multi-cloud computing [7][8]. The cloud refactoring process does not start and end with the migration process, but extends to include re-architecting of the applications so that they can be used to the maximum of cloud infrastructure in terms of dynamism and scalability. Refactoring can be motivated by such goals as improved scalability, lower cost of operation, maintainability, and faster development speed. The results of all these are, however, heavily

* Corresponding author: Ramadevi Nunna

reliant on the selection of platform and tooling and the underlying capabilities that the cloud environment offers [9][10]. That is why the comparative analysis of Azure and OpenShift in the context of cloud-native application refactoring is important to the companies that consider their cloud strategy. The scale of the overall enterprise systems requires a systematic way of choosing the use of the clouds, and in particular, one can refer to refactoring plans, potentially including re-coding, re-platforming, or even containerization of current applications. Moreover, the list of available services, such as Azure Kubernetes Service (AKS), Azure Functions, and Azure DevOps, is not comparable to the integrated developer experience of OpenShift, the ability to build applications using Source-to-Image (S2I), and the ability to integrate with the CI/CD platforms, such as Jenkins and Tekton [11][12][13]. Therefore, the choice between Azure and OpenShift in the refactoring of cloud-native applications will have to be made following a thorough analysis of their potential, weaknesses, and the suitability of their application to business needs.

In this paper, we compare the dynamics of the Azure and OpenShift platforms in terms of technical capabilities, flexibility of deployment, integration capabilities, development experience, and cost implications. This is aimed at offering a holistic approach to the concept of how each platform facilitates the refactoring of cloud-native applications. The research design will include the analysis of existing literature, industry examples, and documentation of platforms to evaluate the efficiency of each platform to assist in the transformation of modern applications [14-17]. The following group of sections goes into detail about the underlying ideas of the cloud-native approach to application development and refactoring. At this point, we look into the architecture and functionality of Microsoft Azure and Red Hat OpenShift. This is followed by a detailed comparison that is evidenced by the real-life cases and benchmarking findings that refer to secondary data. The discussion ends with a summary of the pros and cons that are involved in each platform. As a conclusion of the research, the article gives some recommendations on how one may choose a suitable platform depending on the reasons of the enterprise, technical limitations, and business strategies.

2. Cloud-Native Application Architecture and the Need for Refactoring

Building on the background of cloud platforms discussed above, it is important to consider the architectural philosophy of cloud-native applications and understand why businesses are increasingly choosing to refactor traditional systems. This part gives a theoretical and real-life architecture to the concept of cloud-native application design, as well as the motivation and business necessity to refactor existing applications with the help of such platforms as Azure and OpenShift. Cloud-native applications have been designed to perform well in the cloud by fully exploiting the dynamic features of platforms, including containers, microservices, DevOps automation, and elastic infrastructure. Fundamentally, these applications are created keeping in mind principles of scalability, fault tolerance, resilience, and service decentralization. They are usually made up of weakly connected services that interact over APIs or event-driven systems and are hence suitable candidates for modern development and deployment pipelines [18][19].

Refactoring the application to be cloud-native typically requires re-architecting the internal structure of the application without altering its external behavior. This is a different process than simply moving and migrating applications to the cloud, which can still have an underlying legacy of restrictions in architecture. Conversely, refactoring is that which will attempt of re-architecting applications to access cloud-native features of auto-scaling, container orchestration, infrastructure-as-code, and microservice separation. The major benefit of the strategy is that it also allows better system performance, more maintainable, shorter time-to-market, and increased resource efficiency [20][21].

The growing need for digital agility can be identified as one of the factors that caused the transition to cloud-native refactoring. Organizations need to act quickly in adapting to new customer demands, competition, and new technological advancements. The monolithic applications of the old kind are usually too rigid to be able to respond to these pressures. By refactoring them into microservices, one can independently develop, test, and deploy single components and thus reduce the risk of system-wide failures and faster innovation cycles [22][23]. Besides that, the spread of DevOps practices has also promoted the cloud-native movement. DevOps prioritizes continuous delivery and continuous integration (CI/CD), automated testing, and quick deployment of microservices and container-based applications are more favorable. Both Azure and OpenShift can be successfully used in terms of their strong support of CI/CD pipelines, container life-cycle management, and the integration with popular DevOps tools, so they can be the wise options in the context of refactoring programmes [24][25].

Cloud-native application architecture is also more or less compatible with Kubernetes, which is the de facto standard for the orchestration of containers. Kubernetes enables applications to be deployed in containers, managing their scaling, failover, and deployment automatically over distributed systems. Both Azure and OpenShift are based on Kubernetes, where Azure offers Azure Kubernetes Service (AKS), and OpenShift offers an improved Kubernetes platform with enterprise-grade functionality. This common base allows emphasizing the significance of keeping the specifics of each platform in mind when it comes to application refactoring [26][27]. Security and compliance are also

important elements that impact the transition to cloud-native refactoring. Monolithic applications tend to use perimeter-based models of security that are no longer viable in the dynamic and distributed cloud environments. Cloud-native architecture embraces the zero-trust model and leverages capabilities such as service meshes, role-based access control (RBAC), and policy-as-code frameworks to ensure security across all layers. Azure and OpenShift assist in these current security paradigms with native services and third-party integrations, which make it possible to use refactored applications when deployed securely in public, private, and hybrid clouds [28][29].

Besides, cloud computing economics also gives an added incentive to refactoring. Autoscaling, right-sizing, and ephemeral compute models are used to allocate resources more efficiently by refactoring cloud-native applications. This can be translated to a reduction in the operational cost, especially when working on a load that is variable or easy to predict. Azure and open-source resource optimization tools and cost management dashboards can help enterprises attain these financial efficiencies, but their pricing models vary and need to be carefully considered during planning phases of refactoring [30].

With the development of cloud-native technologies, an ecosystem of open-source tools, libraries, and frameworks that make these applications possible to develop and manage is growing. The tools, such as Visual Studio, Azure DevOps, and GitHub Actions, can be easily supported by Azure as it is a part of the larger Microsoft ecosystem. However, OpenShift is more open, standards-based, and interoperable, which means that it can be used in a multi-cloud and hybrid environment. The two platforms are compatible with the entire cloud-native stack (both container registries and observability frameworks and service meshes), differing in user experience, vendor lock-in, and flexibility in customization.

Finally, the choice to rewrite legacy applications into cloud-native versions is not a technical one; it is a strategic change that affects the processes of development, operational patterns, and business performance. Business organizations should determine the complexity, cost, and payback of refactoring as well as align these factors with platform capabilities. In the given context, Azure and OpenShift are two major paradigms of cloud-native application refactoring: the former is based on a tightly coupled cloud ecosystem, whereas the latter is an open and flexible environment hardened in the enterprise. Having identified the key driving forces and architectural principles of cloud-native refactoring, now well-understood, the following section will give a detailed analysis of Microsoft Azure. It will discuss the particular tools, services, and architectural designs of Azure that enable cloud-native transformation with its contribution to the contemporary enterprise application refactoring.

3. Microsoft Azure for Cloud-Native Refactoring

Continuing on the premise of cloud-native designs and the business case for refactoring legacy systems, we are now considering Microsoft Azure, one of the most widely adopted cloud platforms globally. In this part, the process of refactoring cloud-native applications with the help of Azure will be critically reviewed to highlight the technical capabilities of the platform, tools, services, and operational advantages of the product to enterprises that are undergoing the digital transformation process.

Microsoft Azure has differentiated itself as a complete spectrum of cloud computing platform that provides Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) solutions. The platform provided by Azure has a rich ecosystem, including Azure Kubernetes Service (AKS), Azure App Service, Azure Functions, Azure DevOps, and is connected to GitHub, which is applicable to organizations that seek cloud-native refactoring. Such services offer the foundations needed to realize the transformation of the traditional monolithic applications into modular, containerized, and scalable microservices-based architectures [1]. Azure Kubernetes Service (AKS) can be considered one of the main features of Azure when it comes to supporting cloud-native refactoring. AKS is a service managed Kubernetes; it simplifies the complexities of managing a Kubernetes cluster and also provides the ability to orchestrate a containerized workload. AKS also provides automatic upgrades, monitoring, and scaling, which is why it is a good option when configuring microservices on a large scale. Enterprises would be able to move application components to containers and deploy them on the AKS in full-fledged CI/CD pipelines, logging, and health monitoring [4]. Where stateless architecture or event-driven architecture are more desirable, Azure provides Azure Functions, a serverless computing service that enables a developer to code without configuring a server. This helps in breaking down applications into small and independent event-based services. Azure Functions may be effortlessly linked with other Azure services such as Azure Blob Storage, Event Grid, or Azure Service Bus to create reactive systems that are cost-efficient and highly responsive. To refactor monolithic applications, the model offers an alternative to divide components into discrete, scalable, and low-maintenance functions [5][6].

One more important service of cloud-native provided by Azure is Azure App Service, a PaaS solution that provides a variety of frameworks and languages, such as .NET, Java, Node.js, and Python. The service presents built-in load balancing, termination of the SSL, and continuous deployment that eases the deployment of refactored applications. Applications can be containerized and deployed to Azure App Service, or hybrid deployments can be done with AKS when a more complex deployment is required. Moreover, App Service can be deployed in containers, thus it can be used with step-by-step modernization plans [7][8]. The combination of Azure DevOps and GitHub Actions offers complete CI/CD solutions that are needed in a cloud-native development environment. Azure DevOps consists of planning tools, build tools, testing tools, and application delivery tools. The pipelines can be set to automatically construct and deploy container programs to either AKS or App Service. This close-knit environment allows development teams to enforce agile workflows and shorten cycle periods, which is a major goal of cloud-native transformation [9][10]. In addition, the Artifact service by Azure can be used to manage the packages of teams and improve the modular development.

Governance and security-wise, Azure provides Azure Policy, Azure Security Center, and Azure Key Vault, which prove essential in controlling access, securing secrets, and ensuring compliance in the refactoring process. These are used to enforce and support zero-trust architecture, role-based access control (RBAC), and policy-as-code. This is more so in organizations operating in regulated sectors, whereby security and compliance of the application have to be ensured throughout the lifecycle [11][12]. A cloud-native application cannot be complete without observability and monitoring. Azure offers Azure Monitor, log analytics, and application insights to monitor metrics, performance, and log data, as well as trace application behavior. Such services assist in the detection of bottlenecks in the performance, memory leaks, or network latencies within refactored applications. The in-built dashboards and alerting systems also support the proactive incident response and workflow-based automated remediation, which are significant in a large-scale cloud-native environment.

Azure has one great asset: its hybrid cloud, mainly Azure Arc and Azure Stack. Azure Arc enables enterprises to operate Kubernetes groups and other assets in on-premises, multi-cloud, and edge setups with the help of Azure Resource Manager. This facilitates the ability to have uniform governance and DevOps across non-homogeneous environments. In the case of those organizations that are refactoring legacy applications that cannot be fully migrated to the cloud, an option offered by Azure Stack is to modernize on-premises systems with cloud-native methods. Also, Azure can work with numerous databases and storage options that are essential to cloud-native applications, such as Azure Cosmos DB, Azure SQL, and Azure Blob Storage. Such services are scalable, highly available, and globally distributed. As an example, Cosmos DB offers multi-region writes and milliseconds of latency, which is suitable when working with microservices that need real-time data access. In refactoring traditional database elements, Azure supports data partitioning to support schema transformation and API migration strategies to support microservice architectures [17][18].

The cost management and optimization tools that are present in Azure will also aid in the refactoring process through visibility of costs, forecasting capacity. Azure Cost Management and Azure Advisor provide information on the resources that are underutilized and recommend resizing or scheduling. Containers and serverless functions use refactored applications and thus are priced by consumption, which is less expensive than infrastructure provisioned permanently [19][20]. The wide use of Azure is also improved by open-source technology that supports it and the capacity to integrate with development models. It is compatible with Linux-based workloads, Docker Hub container images, Helm charts, Terraform, Ansible, and Jenkins. This interoperability enables the deployment of teams using an already established DevOps pipeline or container strategy without any major retooling, which saves time-to-market for refactored applications in the Azure ecosystem.

To conclude, Microsoft Azure is a cloud-native application refactoring platform based on an integrated and enterprise-level platform. Azure initiative offers a broad set of refactoring options, including simple containerization, all the way to sophisticated microservices transformation, with a variety of services such as containerization, serverless computing, CI/CD, monitoring, and cost management. Nevertheless, the benefit of Azure is that it is highly integrated into the Microsoft ecosystem, which can be a problem with organizations that need vendor-neutral or multi-cloud solutions. This trade-off is made more evident in comparison with OpenShift, which we discuss in the following section. In order to see the wider choice that cloud-native refactoring opens, we now move to OpenShift, which is a platform based on Kubernetes and focuses on flexibility, portability, and support of hybrid clouds.

4. OpenShift for Cloud-Native Refactoring

After analysing the features of Microsoft Azure in the area pertaining to the refactoring of cloud-native applications, it is essential to discuss Red Hat OpenShift as the opposing force in the given comparative analysis, as shown in Figure 1. OpenShift is an application that is powerful and uses Kubernetes to provide support in managing container orchestration, microservices architecture, and DevOps automation at an enterprise level. Its open-source nature,

coupled with its elasticity and focus on a hybrid and multi-cloud deployment framework, makes OpenShift a strong option among those companies that have initiated application modernization projects. As part of this section, we compare how OpenShift facilitates refactoring of cloud-native applications using its architecture, tooling, development processes, and operational efficiencies.

OpenShift is based on Kubernetes, though, as opposed to the upstream vanilla Kubernetes distribution, OpenShift adds enterprise-grade functionality such as built-in CI/CD pipelines, source-to-image (S2I) builds, integrated logging and observability, and a powerful security framework. This enables the organizations to consider OpenShift as a one-stop shop to write, deploy, and maintain refactored applications. OpenShift can run any type of stateless and stateful workloads to support most types of refactoring activities, including breaking monoliths down into microservices, to those that deploy containerized legacy applications on a controlled platform [1][2]. The Source-to-Image (S2I) functionality is one of the contributions the cloud-native ecosystem makes to OpenShift. S2I is a framework that enables developers to create reproducible container images straight out of source code. This simplifies the refactoring of legacy applications, particularly applications written using languages such as Java, .NET Core, or Python, since it removes the requirement to create Dockerfiles manually or to build up and maintain image builds. The developers need only to push their source code to the built-in Git storage of OpenShift, and the platform manages the containerization, the build triggers, and deployments, which speeds up the entire process of refactoring [3][4].

The other deal that OpenShift is doing well is its Developer Experience (DevEx). The platform consists of a powerful web-based console and CLI tools, which simplify the complexity of Kubernetes and open cloud-native development to a larger group of engineers. As a case in point, the Developer Console of OpenShift is able to visualize application topologies, monitor applications in real time, and offer pipeline integrations to allow the developers to manage refactored applications without needing an in-depth understanding of Kubernetes. This is especially fitting for refactoring projects, in which a team can be moving out of the old-fashioned development processes to the new DevOps-oriented processes [5][6].

OpenShift also integrates well with Jenkins and Tekton in terms of CI/CD, and thus, allows completely automated build and deploy pipelines to containerized applications. Jenkins pipelines can be tweaked to support advanced deployment processes, and Tekton is a Kubernetes-native alternative that is more scalable and cloud-native. The tools can support continuous integration and delivery of refactored applications and provide faster release cycles, and promote consistency across development environments [7][8]. Security is one of the main issues in any refactoring effort, and OpenShift deals with it in a secure-by-default fashion. In contrast to most Kubernetes distributions, OpenShift uses stringent security and includes Security Context Constraints (SCC), integrated role-based access control (RBAC), and network policies. These characteristics assist in making sure that the refactored applications will not compromise the enterprise security standards. The built-in policies and compliance functionality of OpenShift offer a critical base in a controlled setting where the security governance is paramount [9][10]. OpenShift is also superior in the field of multi-cloud and hybrid-cloud implementation, which is very important when the enterprise uses different infrastructures. With OpenShift Container Platform (OCP), companies are able to deploy the same application stack in-premise, in public clouds (including AWS, Azure, and Google Cloud), or in an edge environment. This feature can be especially useful when it is necessary to refactor an application that has to communicate with legacy systems that cannot be fully transferred to the cloud. The similar operating model on these environments by OpenShift helps the applications to have a uniform behavior wherever they are deployed [11][12]. There is also the Operator Framework support that supports the deployment and Lifecycle management of complex stateful applications on the platform. In refactoring applications that are dependent on databases, message queues, or any other stateful service, operators make it easier to manage these dependencies. The OpenShift provides certified Operators of popular services like MongoDB, PostgreSQL, and Kafka, enabling teams to seamlessly integrate these services into cloud-native environments and repeat that integration [13][14].

The Service Mesh (an Istio-based service) offered by OpenShift also enables refactoring by offering observability, traffic, and security to microservices. When applications are broken down into smaller services, communication between the legacy applications cannot be easily managed. The OpenShift Service Mesh can support circuit breakers, retries, traffic shaping, and mutual TLS without any modification of application code. This abstraction makes the refactoring process easier and improves operational visibility and resilience [15][16]. Regarding observability and monitoring, OpenShift integrates the Elastic Stack (ELK), Prometheus, and Grafana as default, allowing one to monitor the application performance and infrastructure metrics in real-time. In the case of refactored applications, this enables the team to catch anomalies, trace inter-service communication, and also proactively solve performance problems. Such monitoring instruments play a vital role during the dynamic behavior management of cloud-native systems and scale system reliability [17][18].

Storage and state management are also handled well by OpenShift, using Container Storage Interface (CSI) support, and integration with software-defined storage solutions such as OpenShift Data Foundation (ODF). This can be used to provide persistent storage to stateful applications in the refactoring process, so that workloads that are based on databases, caching layers, or file systems can operate reliably in a containerized environment. OpenShift is defined as dynamic provisioning, volume expansion, and snapshotting, all of which are essential to the current data-driven applications [19][20]. The efficiency of costs and optimization of resources are handled with the help of the OpenShift cluster autoscaling, horizontal pod autoscaling, and resource quotas, which ensure that refactored applications use as few resources as possible. The Metering Operator of open shift is capable of monitoring resource usage per project and team, which gives visibility to cost and chargeback in a multi-tenant environment. Although OpenShift does not come with a cost of a license, the efficiency with which the platform brings about resource utilization can compensate for the operational expenses in the long term [21][22].

An important strategic advantage of OpenShift is that it is vendor-neutral. Since OpenShift is open-source and is integrated with open standards, it enables the enterprise to prevent vendor lock-in, which is a concern when using proprietary cloud solutions. This has become attractive to the organization that needs to keep its application ecosystems portable and under long-term control, particularly when it comes to refactoring of mission-critical workloads [23][24].

Notwithstanding these advantages, not everything is going to be smooth sailing in OpenShift. The platform needs a more initial learning curve and expertise on Kubernetes and the Red Hat tooling. Its deployment and operational complexity are more than that of a managed service such as AKS on Azure, especially when deployed on self-management. Nevertheless, the challenges are usually superseded by the flexibility, control, and consistency of the platform in a large-scale enterprise setting, particularly when the team using the platform has an established DevOps culture [25][26]. Finally, OpenShift is a complex and scalable solution for migrating old applications to cloud-native systems. Its long history of developer experience, built-in CI/CD tooling, solid security position, and multi-cloud capabilities make it a powerful platform on which modernization efforts can be undertaken. The architectural consistency and extensibility of OpenShift will deliver long-term gains of agility, control, and resiliency to the complex and scaled refactored applications. In order to have a global perspective of the refactoring landscape, the following section provides a comparative study of Azure and OpenShift, generalizing their benefits, weaknesses, and applicability to various applications.

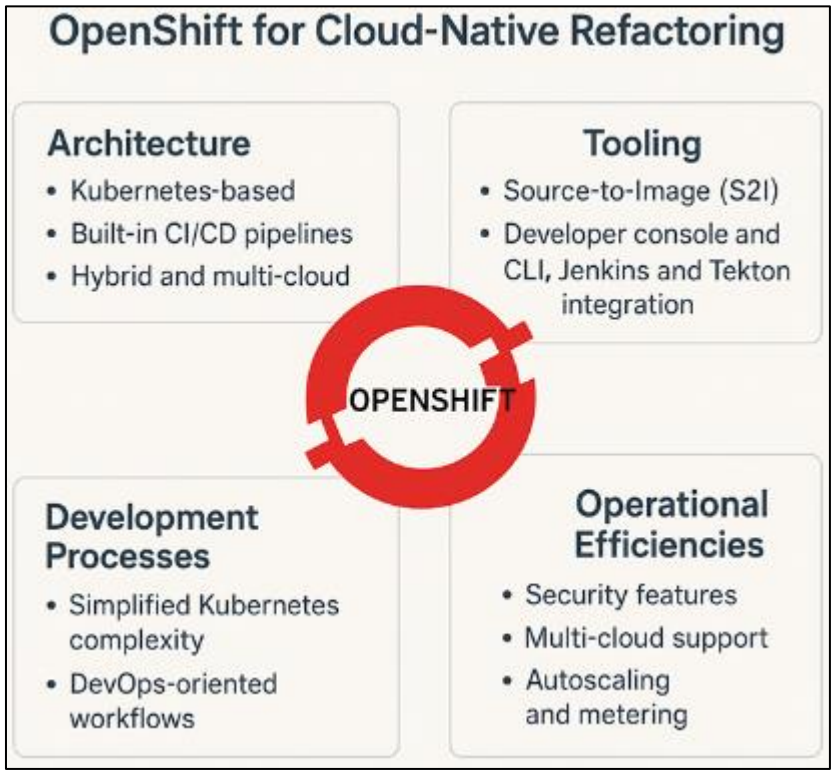

Figure 1 Key Capabilities of OpenShift for Cloud-Native Refactoring

A structured overview highlighting OpenShift's strengths in architecture, tooling, development processes, and operational efficiencies to support application modernization.

5. Comparative Analysis of Azure and OpenShift for Refactoring

After analyzing the features of the Microsoft Azure and Red Hat OpenShift systems separately, the comparative analysis will offer the required synthesis that would help to understand how one of the systems contributes to cloud-native application refactoring. The comparison throws light on the key points like flexibility of deployment, integration of services, security, scalability, developer experience, ecosystem support, and cost management. The knowledge of these dimensions is essential in the strategic decision-making undertaken by organizations in the course of refactoring on the platform that best suits their technical and business goals.

The Azure and OpenShift approach to platform architecture, in particular, differ greatly. Azure offers a managed service paradigm where the complexity of the infrastructure is massively removed from the user. This simplifies the process of teams adopting Kubernetes by using Azure Kubernetes Service (AKS), integrating with Azure-native services, and using PaaS services such as Azure App Service and Azure Functions to quickly deploy refactored parts. By contrast, OpenShift is more open and has a self-contained setup, which not only comes with Kubernetes orchestration but also with improved build systems, routing, monitoring, and logging tools. This is an all-in-one solution that provides more control, yet it needs more set-up and expertise [1][2]. Both platforms support scalability and performance management, which is applied differently. The elasticity at Azure is associated with its close association with the Universal Azure infrastructure. Some services, such as AKS, are automatically scaled depending on the demands, and serverless services, such as Azure Functions, scale based on the consumption. Azure Monitor and Application Insights are performance diagnostics that can be optimized to optimize the consumption of resources. OpenShift also includes horizontal pod autoscaling, cluster autoscaling, and resource quotas, but includes more inclusive native observability (Prometheus and Grafana). In addition, the compatibility of OpenShift with any infrastructure lets enterprises optimize workloads to reduce latency, cost, and regulatory requirements as it can run both in a public cloud, a private cloud, or on-premises [14].

The two platforms offer robust support for CI/CD and DevOps integration; however, their ecosystems differ significantly. The addition of Azure to Azure DevOps and GitHub Actions forms an uninterrupted CI/CD pipeline, which favors companies that are already part of the Microsoft ecosystem. GitHub Codespaces, package feeds, release pipelines, and infrastructure-as-code (such as Bicep and ARM templates) have end-to-end DevOps functionality. Instead, OpenShift provides a natively supported Jenkins, Tekton, and ArgoCD, flexible pipeline definitions, and Operator-based lifecycle management. The OpenShift Pipelines implemented on Tekton are also Kubernetes native and consistent with the cloud-native development principles. This is particularly appealing to a team already possessing a DevOps culture, or those that require increased flexibility in their pipeline configuration [25][26].

One of the biggest distinguishing factors is the application build and deployment model. Azure promotes a hybrid of PaaS and containerized solutions, with Azure App Service being deployed when a fast solution is needed and AKS being utilized in more complicated container orchestration. This hybrid deployment model is convenient, yet it may result in architectural inconsistency due to uncontrolled implementation. The S2I feature of OpenShift offers a rather more integrated and automated way of creating containers based on source code, lessening the burden on developers and simplifying the process of refactoring. OpenShift has a consistent build and deployment workflow, which is particularly useful when dealing with large teams that operate in the environment of multiple services and environments [7][18]. Security and governance are part and parcel of refactoring, particularly in financial, healthcare, and government sectors. Azure provides security services at the enterprise level, including Azure Active Directory, Azure Policy, Azure Defender, and Key Vault. These offer identity management, compliance, and data protection services across the cloud-native workloads. Meanwhile, OpenShift uses a design of security by default. It uses stringent Security Context Constraints (SCC), implements RBAC on a namespace basis, and has compliance profiles (e.g., NIST, PCI DSS). Besides, the platform supports network policies and is integrated with SELinux, which is why it is the most appropriate solution in enterprises with high security requirements [9].

In terms of ecosystem and tooling support, the fact that Azure is integrated with the Microsoft ecosystem, in particular, Visual Studio, Power BI, Microsoft 365, and Dynamics, is a major benefit to those businesses who already use these tools. The Azure Marketplace also has a wide variety of third-party tools, SDKs, and services. OpenShift is open-source and focuses on interoperability and extensibility. It embraces an extensive selection of development languages, databases, CI/CD, and observability products, and is usually without as many restrictions and is more open than Azure. Such a vendor-neutral approach is advantageous to organizations that require flexibility over the long term and do not want to be locked into the cloud [11]. Another dimension open to competition is portability and hybrid cloud readiness, which

OpenShift demonstrates. As Azure has gone a long way with Azure Arc and Azure Stack, it is still essentially bound to the Microsoft ecosystem. However, it is built on the ground to enable hybrid and multi-cloud strategies, which OpenShift does. It is interoperable with AWS, Google Cloud, Azure, bare-metal, l and private clouds. The portability comes in handy, especially when refactoring applications that need to cut across cloud and on-premises or when they are planning their cloud exit strategies [1][14].

The platform is also determined by cost and licensing models. Azure has consumption-based pricing models for the majority of its services, such as containers, functions, and databases, which are simpler to optimize the cost based on usage. But it also brings along with it the unknown expenses of inter-service communication, storage, and ingress/egress charges. The licensing of OpenShift is more predictable since it is done by the number of nodes/cores, particularly when managed by self-managed environments. There are, however, chances that the cost of operation could be more as a result of the infrastructure management. When deciding about refactoring projects, organizations have to pay close attention to the TCO (Total Cost of Ownership), which involves balancing licensing, operational efficiency, and support requirements. As a developer experience, Azure offers an integrated development experience over Visual Studio, Azure CLI, and web portals that help simplify development, deployment, and monitoring. It enables full-stack application development and preconfigured environments and has a low entry barrier. OpenShift offers a better developer experience to Kubernetes ecosystems, such as interactive dashboards, topology visualizations, and application creation via Git imports or templates. Azure is supposedly simpler to work with, but OpenShift is more customizable and transparent during the development process.

Community and Support Azure have access to the global presence and broad documentation of Microsoft, and enterprise support for most agreements. OpenShift also has a strong community support, especially that of the Kubernetes and Red Hat ecosystems. Additionally, Red Hat subscription model comes with support, updates, and long-term maintenance, which is highly rated in production- scale refactoring projects. The compatibility of OpenShift with open-source technologies guarantees the OpenShift solution remains in sync with the developments of the Kubernetes ecosystem that offer faster uptake of new standards and best practices [19][20].

To sum up, both Azure and OpenShift have great cloud-native application refactoring, although they fit various strategic priorities and profiles of organizations. Azure has a high deployment, integration, and scale in the controlled environment; thus, it is best suited to companies that demand simplicity, speed, and an effortless assortment with Microsoft services. Compared to OpenShift, the latter is more flexible, controllable, and open, which is why it could be used by companies with hybrid-cloud services, regulatory concerns, or the desire to rely on open-source ecosystems. The best option does not necessarily rely only on technical capabilities, but it is also based on the compatibility with enterprise architecture, team maturity, regulatory environment, and long-term cloud strategy. The major findings and strategic guidance of choosing the right platform for cloud-native application refactoring according to the organizational priorities and technical needs will be recapped in the following section.

To complement the narrative comparison, the following table offers a synthesized decision matrix that organizations can use to evaluate platform fit for cloud-native application refactoring based on different strategic and operational criteria.

Table 1 Strategic Decision Matrix for Choosing Between Azure and OpenShift for Refactoring

Evaluation Criteria	Microsoft Azure	Red Hat OpenShift
Ideal Enterprise Profile	Microsoft-stack-centric organizations seeking rapid deployment	Enterprises favoring open-source, hybrid/multi-cloud setups
Infrastructure Management	Fully managed (AKS, App Service, Functions)	Self-managed or managed OpenShift; full control over the stack
Compliance Readiness	Broad compliance templates (HIPAA, ISO, GDPR, etc.)	Deep infrastructure-level compliance control (SCC, SELinux)
CI/CD Model	Integrated via Azure DevOps and GitHub Actions	Native pipelines with Jenkins, Tekton, ArgoCD
Security Posture	Policy-based (Azure Policy, Defender)	Secure by default; granular container and network policies

Evaluation Criteria	Microsoft Azure	Red Hat OpenShift
Deployment Flexibility	Primarily Azure cloud, some hybrid via Azure Arc	Public, private, on-premise, and edge ready (true hybrid)
Learning Curve	Low to medium (GUI-rich, Microsoft ecosystem)	Medium to high (requires Kubernetes and DevOps proficiency)
Vendor Lock-in Risk	Higher (tied to Azure ecosystem and APIs)	Lower (open standards, portability)
TCO Predictability	Usage-based, variable costs	Subscription or per-core model; predictable in long-term use
Tooling Ecosystem	Rich Microsoft + third-party integrations	Open-source tooling, Operator Framework support

This comparative matrix illustrates that platform suitability is not absolute but highly contextual, dependent on factors like regulatory exposure, infrastructure control needs, DevOps maturity, and openness to vendor lock-in. These comparative insights further strengthen the case for strategic alignment when selecting a cloud-native refactoring platform.

6. Conclusion and Recommendations

Based on the close analysis and the comparative analysis of Microsoft Azure and Red Hat OpenShift, it can be stated that the two platforms provide powerful frameworks that can be used to refactor cloud-native applications. But their solutions differ greatly when it comes to architecture, their philosophy of operation, integration ability, and strategic fit. This last part summarizes the lessons learned in the course of the research and provides strategic advice to companies that contemplate or are in the process of cloud-native transformation.

Refactoring of cloud-native applications is not only a technical task but a paradigm of transforming the software system architecture, implementation, and maintenance. It requires a platform that will enhance modularization, scalability, security, automation, and operational effectiveness. At that, both Azure and OpenShift meet the challenge by offering container orchestration, development models that are amenable to microservice development, CI/CD tooling, and compliance mechanisms that are designed to match the contemporary software development paradigms. Nonetheless, the choice between either of the two is dependent on a delicate knowledge of the current ecosystem of each organization, their vision of strategic direction, culture of development, and the regulatory environment.

Azure stands out because it is heavily integrated with the Microsoft ecosystem and integrates with such tools as Visual Studio, Azure DevOps, GitHub, and Microsoft 365. This close integration helps in fast onboarding, simplified workflows, and an integrated development process, particularly among businesses that are already using Microsoft technologies. The model of managed services offered by Azure lowers the complexity of infrastructure provisioning and maintenance, which is why it is the best fit for businesses that value speed to market and minimal operational overhead, as well as platform-driven innovation. Its resource allocation approach based on consumption also fits the patterns of dynamically allocated resources of cloud-native applications, which provide financial flexibility to both startups and large organizations.

On the other hand, OpenShift provides an open-standard platform, and it is flexible, portable, and enterprise-controllable. Enhanced with built-in services, such as Source-to-Image (S2I), Jenkins / Tekton CI/CD, Prometheus-based monitoring, and a service mesh of complex microservices communication, its Kubernetes base is an added advantage. The value proposition of OpenShift is that it is capable of providing a unified experience in development and operation on various platforms, including public clouds, private data centers, and the edges. This is why it is especially appropriate for the industry with a high demand to have data residency, a hybrid IT approach, or dependence on legacy infrastructure. In firms that have developed a DevOps culture and want to be vendor-neutral, OpenShift offers an extensible and transparent platform that can be used to support complex application refactoring without sacrificing flexibility. Conversely, companies that are interested in the convenience, quick implementation, and well-integrated services might consider Azure as more beneficial, especially in cases where their employees already have some prior knowledge of Microsoft technologies. In addition, Vladimir notes that the Kubernetes (AKS), serverless, and PaaS offerings managed by Azure have offered incremental modernization avenues, which are appropriate in the context of incremental refactoring.

Security and compliance are the key factors in the choice of platforms. Azure provides the tools of complete governance, such as Azure Policy and Security Center, and built-in compliance templates that correlate to the global standards. OpenShift, on its part, implements security down to the container level (via Linux security modules) and up to the network segmentation and access controls, which are tightly built into the design of the cluster. The companies that work in such industries as healthcare, finance, or government tend to choose OpenShift because of the strong security customization and the control of the infrastructure, especially in the case of the regulated or sovereign cloud. The platforms are also further differentiated in terms of cost. The pay-as-you-go plan of Azure is beneficial when workloads have unpredictable demand, whereas open shift licensing can provide a better estimate of the spending in stable and high-volume setups. Operational needs of OpenShift may be complex and overwhelming; however, unless it is taken as a managed product like Red Hat OpenShift on Azure (ARO), AWS (ROSA), or Google Cloud. The OpenShift managed versions allow a balance to be struck between control and convenience to enable its usage by organizations that do not have advanced Kubernetes skills.

In light of these findings, the following strategic recommendations can be offered for enterprises evaluating cloud-native application refactoring platforms:

- **Assess Organizational Ecosystem Compatibility**: If the enterprise is heavily invested in Microsoft tools and practices, Azure provides a seamless and accelerated pathway to cloud-native refactoring. OpenShift is better suited for organizations with a strong Linux background or those using diverse technology stacks and requiring interoperability.
- **Define Cloud Strategy Scope (Public, Private, Hybrid, Multi-cloud)**: For hybrid and multi-cloud strategies, OpenShift's infrastructure agnosticism offers greater flexibility. For public cloud-focused deployments, Azure's integrated offerings simplify adoption.
- **Evaluate Operational Maturity and Skills Availability**: Azure's managed services are appropriate for teams with limited Kubernetes experience, while OpenShift requires (or benefits more from) a team familiar with containerization, Kubernetes internals, and DevOps tooling.
- **Prioritize Security, Compliance, and Data Sovereignty**: Organizations operating under strict regulatory frameworks may benefit from OpenShift's customizable security layers and on-premise deployment capabilities. Azure remains suitable where cloud-native compliance templates suffice, especially in global, distributed operations.
- **Adopt a Phased Refactoring Approach**: In either platform, phased refactoring, starting with containerization, moving toward microservices and serverless models, enables risk mitigation and better resource allocation. Azure's modular services and OpenShift's extensible pipelines support such incremental strategies.
- **Leverage Platform-Specific Tooling for Optimization**: Utilize Azure Cost Management, Application Insights, and Azure Advisor to optimize costs and performance during refactoring. In OpenShift, employ Prometheus/Grafana, the Metering Operator, and Cluster Autoscaler for fine-grained control and observability.

Finally, both Azure and OpenShift are cloud-native platforms, both of which are of particular strength. The choice between one and the other cannot be confined to the technical features, but must be directed by organizational long-term goals, compliance requirements, and vision of architecture. There are instances when a hybrid model that uses both platforms, like using OpenShift in Azure, can be the best available, where operational consistency is coupled with platform-native integration. With cloud-native refactoring becoming the foundation of the digital transformation, enterprises need to be provided not only with the appropriate platform but also with the appropriate strategies, expertise, and governance models to get ahead. The given comparative analysis highlights the importance of a specific approach to selecting the platform, which can be based on technical evidence and strategic thinking.

References

[1] Burns, B., Beda, J., Hightower, K., & Evenson, L. (2022). *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc."

[2] Shyam, G. (2021). *Cloud computing: Concepts and technologies*. CRC Press.

[3] Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture patterns to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10ccc) (pp. 583-590). IEEE.*

[4] Maissen, P., Felber, P., Kropf, P., & Schiavoni, V. (2020, July). Faasdom: A benchmark suite for serverless computing. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems* (pp. 73-84).

[5] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, *150*, 77-97.

[6] Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A software architect's perspective*. Addison-Wesley Professional.

[7] Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C. C., Khandelwal, A., Pu, Q., ... & Patterson, D. A. (2019). Cloud programming simplified: A Berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*.

[8] Chen, L., Babar, M. A., & Zhang, H. (2010, April). Towards an evidence-based understanding of electronic data sources. In the *14th International conference on evaluation and assessment in software engineering (EASE)*. BCS Learning & Development.

[9] Voorsluys, W., Broberg, J., & Buyya, R. (2011). Introduction to cloud computing. *Cloud computing: Principles and paradigms*, 1-41.

[10] Theodoropoulos, T., Rosa, L., Benzaid, C., Gray, P., Marin, E., Makris, A., ... & Tserpes, K. (2023). Security in cloud-native services: A survey. *Journal of Cybersecurity and Privacy*, *3*(4), 758-793.

[11] Nkisi-Orji, I., Wiratunga, N., Palihawadana, C., Recio-García, J. A., & Corsar, D. (2020, June). Cloud CBR: Towards microservices-Oriented case-based reasoning. In *International Conference on Case-Based Reasoning* (pp. 129-143). Cham: Springer International Publishing.

[12] Sewak, M., & Singh, S. (2018, April). Winning in the era of serverless computing and function as a service. In *2018 3rd International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE.

[13] Ibryam, B., & Huß, R. (2022). *Kubernetes patterns*. "O'Reilly Media, Inc.".

[14] Chhapola, A., Shrivastav, A., Ravi, V. K., Jampani, S., Gudavalli, S., & Goel, P. (2022). Cloud-native DevOps practices for SAP deployment. *International Journal of Research in Modern Engineering and Emerging Technology*, *10*(2), 95-116.

[15] ABAYOMI, A. A., ODOFIN, O. T., OGBUEFI, E., ADEKUNLE, B. I., AGBOOLA, O. A., & OWOADE, S. (2020). Evaluating Legacy System Refactoring for Cloud-Native Infrastructure Transformation in African Markets.

[16] Obioha Val, O., Selesi-Aina, O., Kolade, T. M., Gbadebo, M. O., Olateju, O., & Olaniyi, O. O. (2024). Real-Time Data Governance and Compliance in Cloud-Native Robotics Systems. *Oluwatosin and Kolade, Titilayo Modupe and Gbadebo, Michael Olayinka and Olateju, Omobolaji and Olaniyi, Oluwaseun Oladeji, Real-Time Data Governance and Compliance in Cloud-Native Robotics Systems (November 12, 2024)*.

[17] Barrett, H., Matthews, J., Ford, A., & Castro, H. (2023). Observability and Monitoring Using Prometheus and Grafana in Cloud Setups.

[18] Peter, H. (2022). Monitoring and Observability in DevOps Environments.

[19] Beattie, T., Hepburn, M., O'Connor, N., & Spring, D. (2021). *DevOps Culture and Practice with OpenShift: Deliver continuous business value through people, processes, and technology*. Packt Publishing Ltd.

[20] Pednekar, A. S. (2025). *A Comparative Analysis of Kubernetes and OpenShift based on Workloads using Different Hardware Architecture* (Doctoral dissertation, Dublin, National College of Ireland).

[21] Fontana, G., & Pecora, R. (2022). *OpenShift Multi-Cluster Management Handbook*.

[22] Wang, Z., Yan, W., & Wang, W. (2020, June). Revisiting cloud migration: strategies and methods. In *Journal of Physics: conference series* (Vol. 1575, No. 1, p. 012232). IOP Publishing.

[23] Gudi, S. R. (2025). A Comparative Analysis of Pivotal Cloud Foundry and OpenShift Cloud Platforms. *Emerging Frontiers Library for The American Journal of Applied Sciences*, *7*(07), 20-29.

[24] Praveen, K. K. AI-Driven Hybrid Strategies for Cloud Migration and Modernization of Java Applications. *J Artif Intell Mach Learn & Data Sci 2025*, *3*(3), 2848-2853.

[25] Motamary, S. (2021). Implementing Infrastructure-as-Code for Telecom Networks: Challenges and Best Practices for Scalable Service Orchestration. *Available at SSRN 5240118*.

[26] Hessellund, A. (2004). Refactoring as a Technique for the Reengineering of Legacy Systems. *ITU, Kobenhavn*.

[27] OpenShift, R. H. (2021). Red Hat OpenShift. *Europe*, *800*(7334), 2835.

[28] Confidently, L. A. O., Chakraborty, M., & Kundan, A. P. (2021). Monitoring Cloud-Native Applications.

[29] Borges, M., Barros, E., & Maia, P. H. (2018). Cloud restriction solver: A refactoring-based approach to migrate applications to the cloud. *Information and Software Technology*, *95*, 346-365.

[30] Attah, R. U., Garba, B. M. P., Gil-Ozoudeh, I., & Iwuanyanwu, O. (2024). Strategic frameworks for digital transformation across logistics and energy sectors: Bridging technology with business strategy. *Open Access Res J Sci Technol*, *12*(02), 070-80.