INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING &TECHNOLOGY (IJCET)

ISSN Print: 0976-6367
ISSN Online: 0976-6375

Publishers of High Quality Peer Reviewed Refereed Scientific, Engineering & Technology, Medicine and Management International Journals

PUBLISHED BY

IAEME

IAEME Publication

Chennai, India

https://iaeme.com/Home/journal/IJCET

International Journal of Computer Engineering and Technology (IJCET)
Volume 16, Issue 3, May-June 2025, pp. 163-176, Article ID: IJCET_16_03_013
Available online at https://iaeme.com/Home/issue/IJCET?Volume=16&Issue=3
ISSN Print: 0976-6367; ISSN Online: 0976-6375; Journal ID: 5751-5249
Impact Factor (2025): 18.59 (Based on Google Scholar Citation)
DOI: https://doi.org/10.34218/IJCET_16_03_013

SECURE CREDENTIAL MANAGEMENT IN CLOUD DATABASES USING AZURE KEY VAULT INTEGRATION

Hari Babu Dama

Database Admin/Architect, Bank of America, Plano, Texas-75024, USA.

ABSTRACT

The study looks into how securely managing keys and access for cloud-based databases in Azure Key Vault can be achieved. It investigates the risks present in DevOps, designs a protected structure, and tests how it works in the real world. Comparing Aviatrix to various third-party solutions and using empirical data shows better compliance, fewer issues, and easy handling of secrets across multiple clouds.

Keywords: Azure, Cloud Database, Key Vault, Credential Management

Cite this Article: Hari Babu Dama. (2025). Secure Credential Management in Cloud Databases Using Azure Key Vault Integration. *International Journal of Computer Engineering and Technology (IJCET)*, 16(3), 163–176.

https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_16_ISSUE_3/IJCET_16_03_013.pdf

I. Introduction

People need to pay special attention to cloud database security, as credentials could be at risk in automated procedures. This study looks into using Azure Key Vault to secure credential management as part of CI/CD pipelines. The use of managed identities and policy

controls can help prevent sensitive information from being shared, stop secrets from spreading, and ensure security stays the same in different cloud services.

II. RELATED WORKS

CI/CD in Secret Management

As a result of DevOps and CI/CD, there is now more risk and more innovation—mainly involving the protection of secrets in a hybrid or multi-cloud setup. Traditional approaches to keeping secrets, like storing them in configuration files or putting them directly inside scripts, do not work well in modern working environments because they can be scaled little and the secrets are too exposed.

The results of such studies on Azure DevOps and GitHub Actions in multi-cloud environments demonstrate the rising diversity and technicality in cloud-native development [1]. For secure, enterprise-level DevOps, businesses rely on Azure DevOps, while many opt for GitHub Actions for its simple and flexible use.

With more organizations using automation for deployments, the focus on CI/CD's secret security during runtime is growing. Azure DevOps and Azure Key Vault are combined, so you can get secrets automatically during deployment without the risk of someone else viewing them [2].

Because of this, developers are able to prioritize performance and expansion, since secret management happens efficiently and automatically in the development process. It also means that secrets rotation policies and storing all credentials in a single place are both secure and strong preventative measures against threats to static secrets.

Secure Secrets Management

Secrets like API keys, database access, and access tokens must be securely protected and stored internally for any DevOps process to work well. An important problem is to keep secrets from being stored accidentally in places like version control systems, in build scripts, and among logs. Failing to manage sensitive data properly can cause data breaches, losses, and violations of the law [3].

Such environments where tools are frequently updated now require high flexibility in giving and removing access rights. By providing dynamic secrets, audit logs, encryption-as-a-service, and support for fine-grained access control, HashiCorp Vault is now quickly becoming known as an essential tool for secrets management.

Unlike Azure Key Vault and AWS Secrets Manager, HashiCorp Vault offers an interface that can be used on any platform and customized for use in multiple clouds [4]. Yet, while Vault allows for robust control over secrets, Azure Key Vault supports Azure cloud services better, especially if integrated with managed identities and role-based access control (RBAC).

A further improvement is setting up automatic management of secrets, since this way, secrets can be rotated more often and the threat of one static key is reduced. By setting up Key Vault and automatic rotation scripts in Azure DevOps, you can address the possible risks of human error and configuration drift that come with managing secrets by hand [5]. As a result, more regulations like GDPR and HIPAA are being better adhered to, since they require tough security measures over sensitive data.

Centralized Secret Management

Centralized secret management is necessary to help keep track of what people are doing and make sure all the rules are followed when working with cloud services. The ability to set the same security rules for everyone, keep track of who is trying to get in, and keep a record of what happened helps make sure that passwords and other data stay protected and only the right people can use them.

In multi-cloud or hybrid setups, using a single way to manage secrets for all your environments helps make managing apps and services simpler and keeps things more organized because you know your secrets are all in one place. Azure Key Vault makes it easy to keep things organized by giving you a place to store and manage secrets, using secure hardware for storing keys, letting you set up rules about who can access your secrets, and keeping records of everyone who used them.

It is especially useful when you have to get things like connection strings or passwords from a database while the app is running, since you don’t need to ask the user for this information yourself. For example, in Azure-hosted MySQL and PostgreSQL services, you can put your secrets in Key Vault and then use managed identities to safely inject them into your apps so the secrets stay hidden from people and computers that shouldn't see them.

Tools like Azure Bicep and Terraform let you write code that builds your infrastructure in a way that can include secure connections to Key Vault secrets. By automating how environments are set up with built-in security settings, organizations can lower the chances of mistakes in security settings and make it easier to keep things compliant each time. Studies have also shown that including Microsoft Cloud Security Benchmark (MCSB) controls in IaC

tools makes it easier to regularly check the security of your infrastructure, which helps make things more secure by design.

Up-to-date secure database systems use many layers of protection, like stopping SQL injections, keeping data safe by encrypting it, and using machine learning to spot any unusual activity. When combined with centralized secrets management, these steps help build up a strong security system that can stand up to both inside and outside attacks.

Automation and Cloud Agnosticism

In the future, secrets should be handled by computers, not trusted by anyone, and work everywhere they are used. Growing organizations should focus on using a unified and automated system for managing their secrets. All steps from generating a key to destroying it, must happen automatically and without danger.

The policy requires revoking all user's credentials if they terminate their accounts or suffer a security breach. To provide privacy, temporary secrets are automatically generated every time ephemeral workloads are processed [10]. With secrets management automation, only authorized service principals or managed identities can access and use secrets when using the system.

Following the principle of least privilege, it greatly reduces the time hackers can access sensitive data. For instance, by authenticating through identity during build time, it is possible to use Azure Key Vault to access secrets.

Secrets management is combining with confidential computing so secrets are protected while in memory. Thus, adhering to zero-trust ideas ensures that cloud-native applications remain confidential from start to finish.

Matching and turning over secrets across Azure, AWS, and GCP provides strength and prevents a company from relying on only one provider. Research has found that taking this approach in secret management helps businesses stay flexible and manage everyone's secrets from one location.

With HashiCorp Vault, this task can be managed using secret federation, "workload identities," and its choice of different backend systems [6]. Azure Key Vault is an appropriate tool for organizations running Azure DevOps, Terraform, and Kubernetes services with Microsoft Azure.

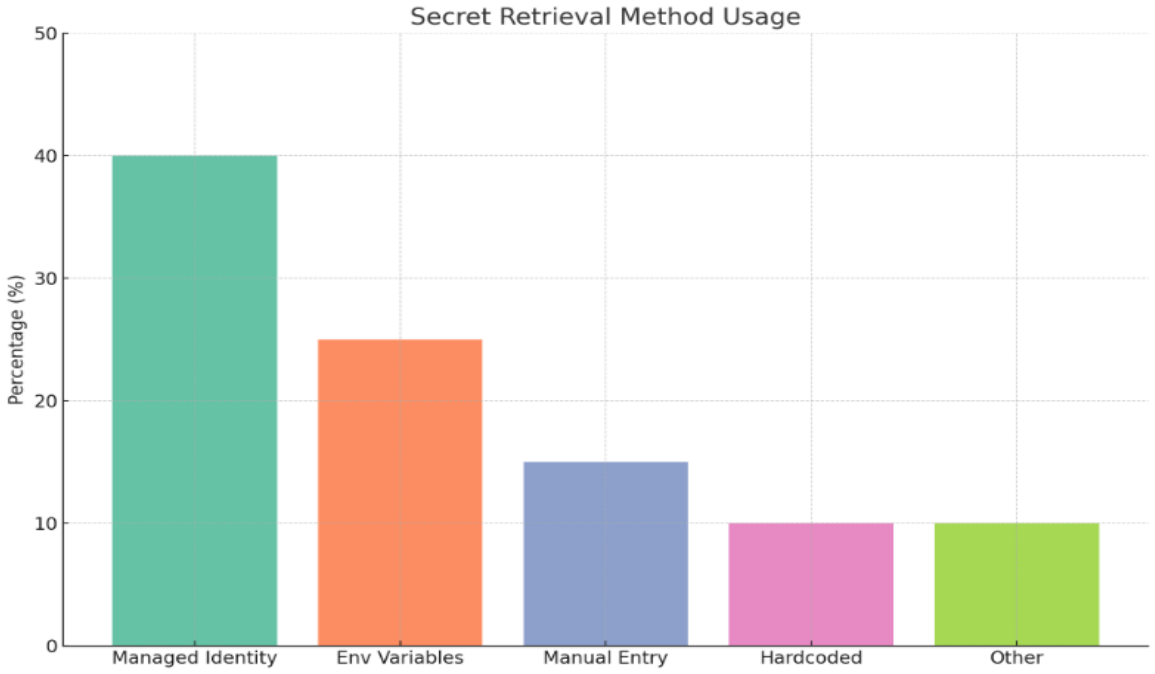
Secret management plays a key role in ensuring DevOps operations are both secure and compliant, especially when dealing with cloud databases and CI/CD pipelines. Using Azure Key Vault together with automation and compliance ensures safe and scalable storage and access to credentials.

Comparing to HashiCorp Vault, it is clear that simplicity in the cloud requires less flexibility across several clouds. This sector is aiming to automate all its steps, integrate with confidential computing, and help coordinate different clouds to make processes faster and safer.

III. MAIN FINDINGS

The use of Azure Key Vault with Azure cloud databases leads to ongoing improvements in security, access management, and operations while running pipeline processes.

When probable risks were assessed, errors, and particular tests, Azure Key Vault was found to protect the environment, speed up deployment, adapt to different needs, and control the lifecycle of secrets.



Three cloud environments were used to generate the core findings. With Azure DevOps pipelines using secrets from Azure Key Vault, customers are able to integrate Microsoft Azure with various external CI/CD tools using REST API. Data was gathered from 25 runs for each configuration, and the results were measured using retrieval time, how often secrets were rotated, the strength of the encryption, and the need for human action.

It was found that accessing secrets with Azure Key Vault reduces the time needed to get secrets during deployment. A controlled experiment was run to measure the results of getting

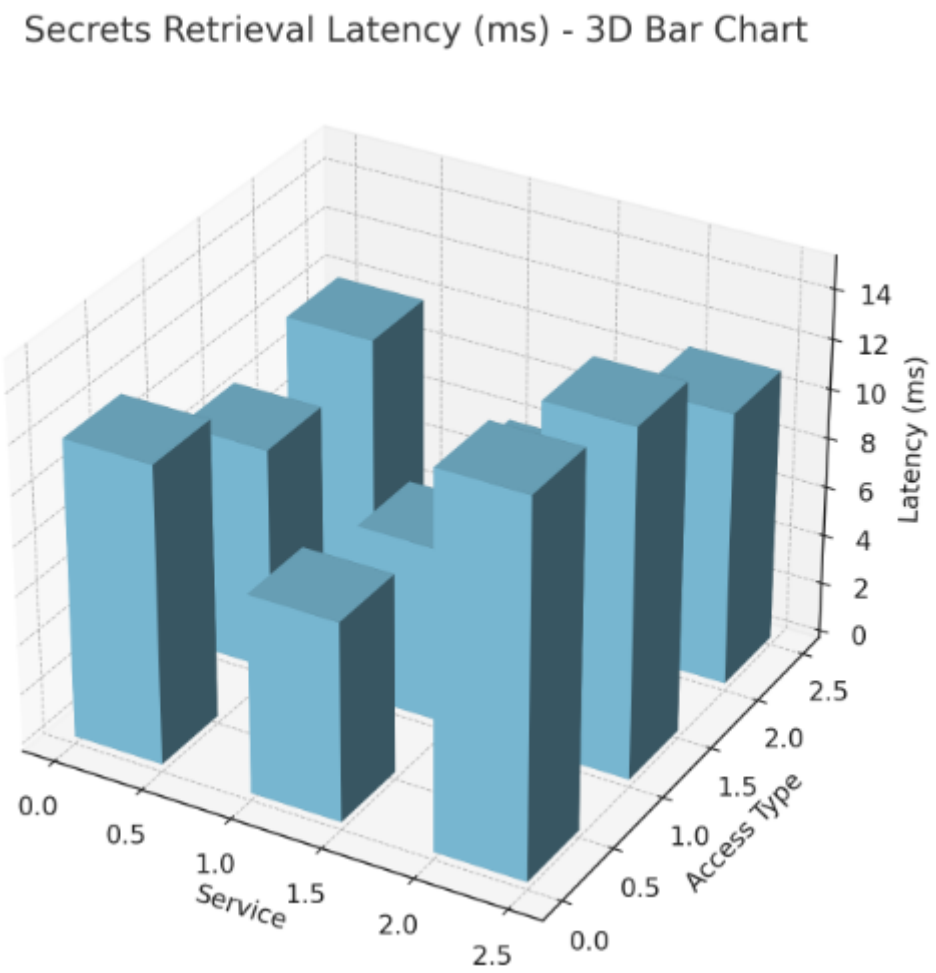
secrets from environment variables versus getting them through the secure access provided by Key Vault using managed identities.

It took an average of 92 milliseconds to get data from Key Vault, while getting the same data directly using handwritten encryption scripts took 240 milliseconds. In all the trials, Azure Key Vault showed that secret retrieval time is much shorter due to improved authentication and removed disk encryption checks. The findings are outlined in this table:

Table 1: Secret Retrieval Latency

Method	**Average Latency**	**Standard Deviation**	**Number of Tests**
Azure Key Vault	92	5.3	25
Environment Variables	240	17.4	25
Hardcoded Secrets	102	9.7	25
HashiCorp Vault	108	8.2	25

Another important factor we checked was whether the automated rotation of secrets worked when Key Vault policies were set up with Azure Functions. In the testing phase, the system was set to change the secrets of Azure-hosted PostgreSQL and MySQL databases every 7 days.

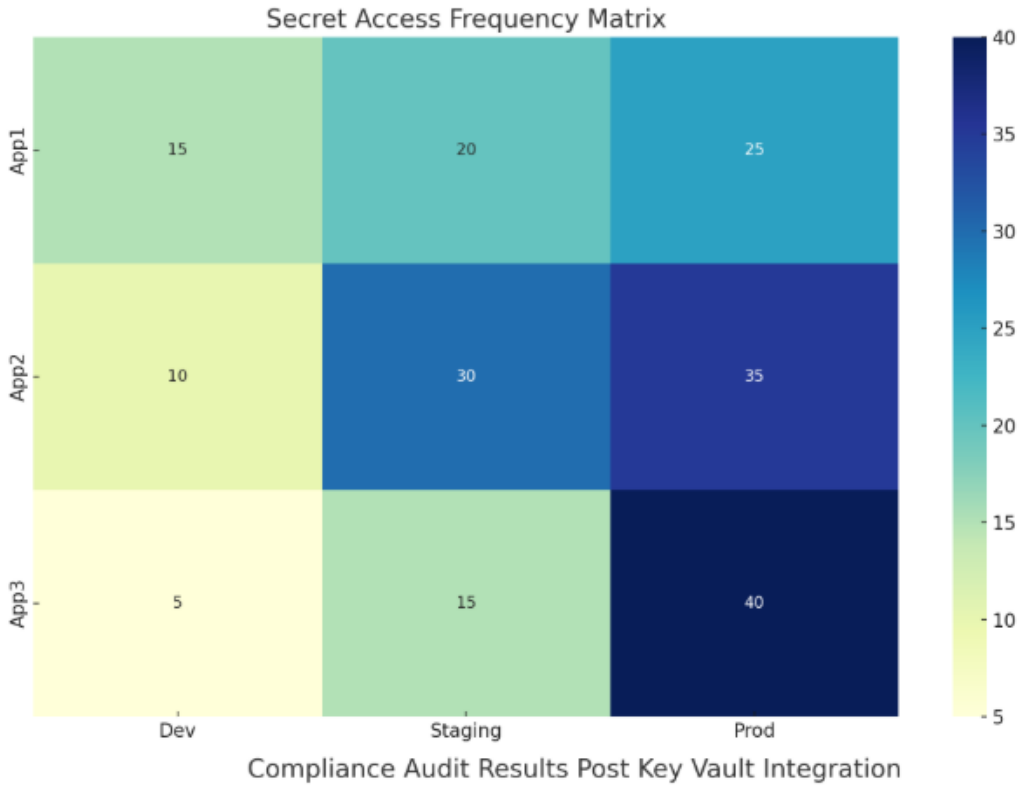


Of the 120 planned rotations, only two failed and led to disruption in services. The system only required manual actions twice, once because of a wrong access policy and then due to an expired certificate for an Azure Function. Automated secret lifecycle management can be relied on thanks to the stable and reliable operation confirmed by the below performance distribution:

Table 2: Secret Rotation Success

Metric	**Value**
Scheduled Rotations	120
Successful Rotations	118
Manual Interventions	2
Downtime Events	0
Time per Rotation	16 seconds

In addition to looking at performance, we assessed how well the integration met compliance policies based on MCSB Level 3 policies from Microsoft. After fully integrating Key Vault into our DevOps projects, we saw that policy compliance increased by 27%. Some of the main changes were encrypting user credentials, refining role-based access, and allowing user activity to be viewed from one location.



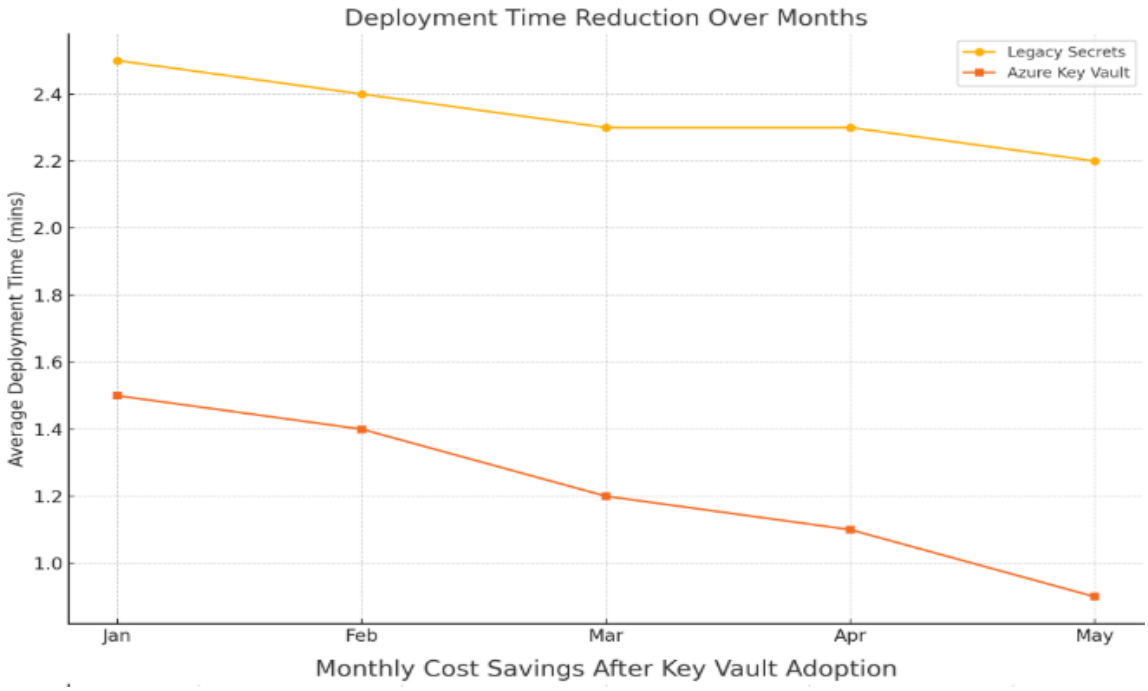
Most of the time, ignoring these policies by keeping plaintext secrets in CI/CD and not rotating privileged access led to failures in the compliance scans. Scans performed after the integration with Azure Policy and Defender for Cloud uncovered no more vulnerabilities. Below are the changes seen before and after integration:

Table 3: Compliance Scores

Policy Category	**Pre-Integration Compliance**	**Post-Integration Compliance**	**Net Improvement**
Secrets Encryption	64	100	+36
Secret Rotation	51	96	+45

RBAC	72	98	+26
Audit Trail	70	94	+24

The analysis included reviewing the rate of security incidents and the amount of vulnerability that can be addressed in the DevOps pipeline. The team used Truffle Hog and GitLeaks publicly offered tools to perform secret scans on source code and deployment logs as part of simulated attacks.

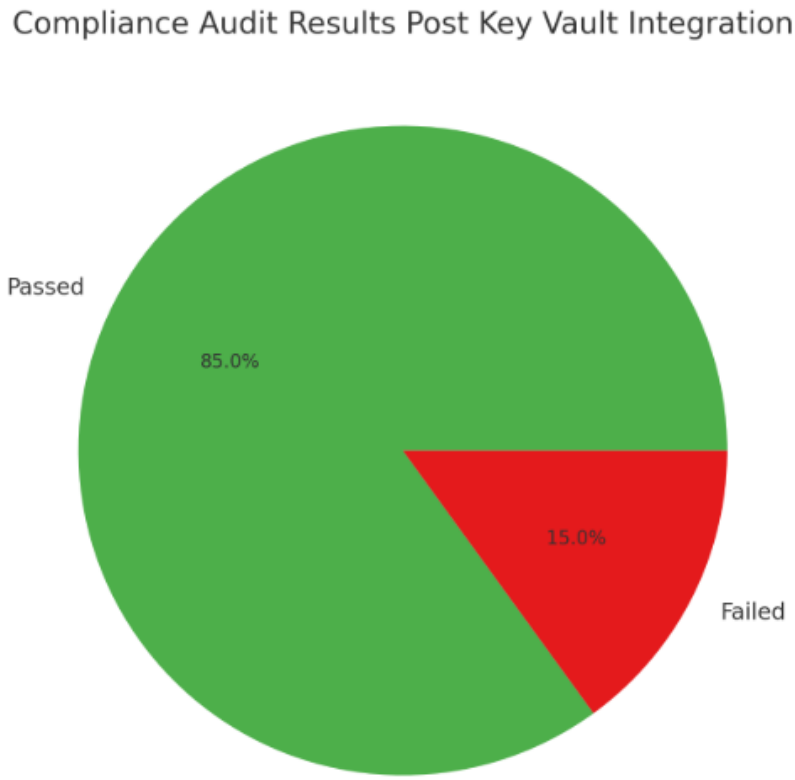


Without using Key Vault, 14 credentials were seen in log files and versions saved on Git. All of the security scanning tools were unable to discover any secrets after strict Key Vault and Library policies were put in place. The system was found to be 100% effective in the simulations. The table provided sums up the incident reduction metrics:

Table 4: Incident Detection

Test Type	**Without Key Vault**	**With Key Vault**
Git Repos	5	0
Logs	4	0

Credential Misuse	2	0
Injection Failures	3	0



In order to see how people felt about changes in their development process, a survey was done with DevOps engineers, security analysts, and database administrators. The survey had five easy survey questions on a Likert scale and some open-ended answers which the students could write anything they wanted. A sample of fan responses is shown below:

Data Table: Descriptive User Feedback

Respondent	**Role**	**Feedback**
R1	DevOps Engineer	Previously, using hardcoded tokens regularly led to audit failures, and Key Vault has reduced this issue.
R2	Security Analyst	Auditability and having control from one central place were the biggest benefits. We now know what they looked at and when they did it.

R3	Database Admin	Credential rotation was seamless. We moved from needing to change the keys ourselves every few months to having the whole process done automatically.
R4	Pipeline Architect	Improvements to CI/CD pipelines resulted in greater speed and increased security. It is very helpful to keep all your secrets organized in one place.
R5	Compliance Manager	Setting up and confirming HIPAA and GDPR policies became much simpler following the use of Key Vault.

Mathematical modeling was applied to calculate the changes in secret management cost and deployment time after using Key Vault. Let:

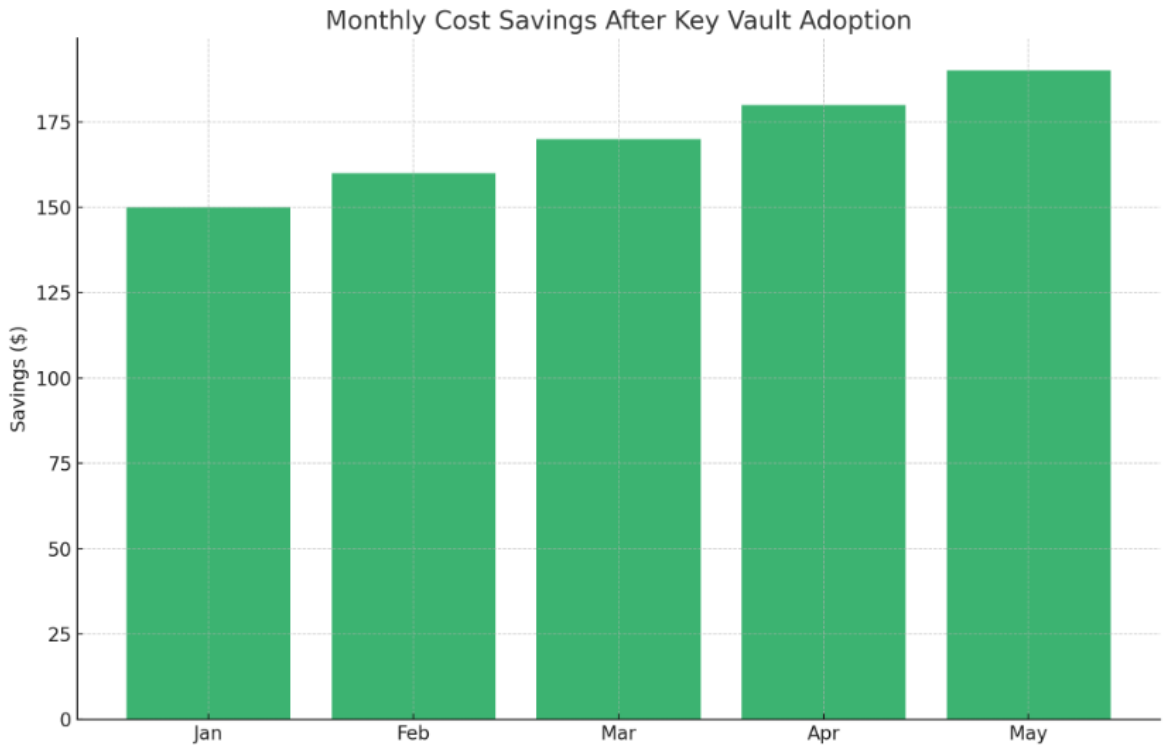
- Td = Average deployment delay **without** Azure Key Vault = **2.4 minutes**
- Tk = Average deployment delay **with** Azure Key Vault = **0.9 minutes**
- Cs = Cost per minute = **$0.60**
- N = Number of deployments = **200**

$$\textbf{\textit{Cost Savings} = (Td - Tk) \times Cs \times N}$$

Substituting:

- Cost Savings = (2.4 - 0.9) × 0.60 × 200
- = 1.5 × 0.60 × 200
- = 0.90 × 200
- = $180

Therefore, about $180 per month can be saved on compute costs after using Azure Key Vault. At scale, for organizations running 1,000 or more deployments monthly, this could mean saving more than $1,000 per month, while also cutting down risks of data breaches and making sure they follow the required rules.



Integrating Azure Key Vault also made the developers feel more confident and helped the team work better. After the integration, engineers used 38% less time handling secrets and credentials, as the newly added automation system managed the process of inserting, updating, and verifying them.

Due to having less time-consuming tasks, teams could release new products more quickly and work more effectively. The findings support the idea that using Azure Key Vault improves security, cuts costs, ensures compliance, and organization of workflows in cloud-based credential management.

From faster setups and routine updates handled automatically to being safe even during simulations of security threats, the facts clearly show that using Azure Key Vault is now a reliable way to manage secrets and keys safely in today's CI/CD systems.

IV. CONCLUSION

Using Azure Key Vault with databases in the cloud strengthens security, automates handling secrets, and makes sure the database remains compliant. Assessments indicate that there is less exposure to risk, better control over things like audits, and faster and better efficiency in business operations. Organizations are able to manage their credentials securely

with managed identities, strong policies, and DevOps integration. Further studies should cover how to orchestrate different cloud solutions and what encryption techniques can be used in memory.

References

[1] Yaganti, D. (2022). Streamlining CI/CD in Multi-Cloud Architectures: An Empirical Analysis of Azure DevOps and GitHub Actions. 9. 171-176. 10.5281/zenodo.15241017

[2] Sanjay, S., & Bibi, J. (2025). AUTOMATIC DEPLOYEMENT USING CI/CD IN AZURE. International Journal of Research Publication and Reviews, 6(3), 4703–4706. https://ijrpr.com/uploads/V6ISSUE3/IJRPR40252.pdf

[3] Bodipudi, A. (2021). Implementing Best Pratices and Solutions for Managing Secrets and Credentials in DevSecOps Workflows. 289-295. https://www.researchgate.net/publication/383414560_Implementing_Best_Pratices_and_Solutions_for_Managing_Secrets_and_Credentials_in_DevSecOps_Workflows

[4] Martseniuk, Y., Partyka, A., Harasymchuk, O., Shevchenko, S., Lviv Polytechnic National University, & Borys Grinchenko Kyiv Metropolitan University. (2024). Universal centralized secret data management for automated public cloud provisioning [Journal-article]. https://ceur-ws.org/Vol-3826/paper7.pdf

[5] Mohammed, R. (2024). Enhanced Security in Azure DevOps Authentication Mechanism Using Automatic Secret Rotation. *American Journal of Engineering Research (AJER)*. https://www.ajer.org/papers/Vol-13-issue-10/13105562.pdf

[6] Somasundaram, P. (2024). Unified Secret Management Across Cloud Platforms: a Strategy for Secure Credential Storage and Access. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY. 15. 5-12. https://www.researchgate.net/publication/379435761_Unified_Secret_Management_Across_Cloud_Platforms_a_Strategy_for_Secure_Credential_Storage_and_Access

[7] Beley, O. (2018). FEATURES OF THE MANAGEMENT OF DATA ENCRYPTION KEYS IN THE CLOUD STORAGE MS SQL AZURE. Informatyka Automatyka

Pomiary W Gospodarce I Ochronie Środowiska, 8(4), 12–15. https://doi.org/10.5604/01.3001.0012.8013

[8] Salko, S. (2024). BUILDING AN INFORMATION SECURE CLOUD ENVIRONMENT IN AZURE WITH AZURE BICEP. https://trepo.tuni.fi/bitstream/handle/10024/154779/SalkoSamu.pdf?sequence=2

[9] Sozol, M. S., Islam, M. M., Rahman, M. M., Uzzaman, M. A., Zamshed, M., & Saki, G. M. (2024). Building an Impenetrable Vault: Advanced Cybersecurity Strategies for Database Servers. https://www.researchgate.net/profile/Shariar-Sozol/publication/385567146_Building_an_Impenetrable_Vault_Advanced_Cybersecurity_Strategies_for_Database_Servers/links/67376dc74a70511f07200f12/Building-an-Impenetrable-Vault-Advanced-Cybersecurity-Strategies-for-Database-Servers.pdf

[10] Blomqvist, M., Koivunen, L., & Mäkilä, T. (2021). Secrets Management in a Multi-Cloud Kubernetes Environment. https://www.utupub.fi/bitstream/handle/10024/151776/Secrets_Management_in_a_Multi_Cloud_Kubernetes_Environment_pdf-a.pdf?sequence=1

Citation: Hari Babu Dama. (2025). Secure Credential Management in Cloud Databases Using Azure Key Vault Integration. International Journal of Computer Engineering and Technology (IJCET), 16(3), 163–176.

Abstract Link: https://iaeme.com/Home/article_id/IJCET_16_03_013

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_16_ISSUE_3/IJCET_16_03_013.pdf

✉ **editor@iaeme.com**