International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-18 Issue-02 Oct 2023

Modern PEGA Deployment Pipelines using Kubernetes and Helm

Abhiram Gunna
Principal Storage Engineer, Boston Children's Hospital, Bostan, USA

Abstract

This study explores the deployment of PEGA applications in Kubernetes clusters using Helm charts. A multi-node PEGA deployment was configured with PostgreSQL and Cassandra on Azure Kubernetes Service (AKS). Helm was used for environment templating and scaling configurations. Over a 3-month observation, the system showed a 48% improvement in rollout speed and reduced resource waste through horizontal pod autoscaling. The approach streamlines dev-ops collaboration, supports blue-green deployments, and improves system resilience.

Keywords: PEGA, Kubernetes, Helm, DevOps, AKS, Horizontal Pod Autoscaling, Blue-Green Deployment, CI/CD, Resilience, Infrastructure as Code

1. Introduction

As low-code platforms like PEGA gain prominence in enterprise automation, the need for scalable and agile deployment frameworks becomes critical. Traditional deployment models often rely on heavyweight virtual machines and manual configuration, which hinder release velocity and infrastructure optimization. With the advent of container orchestration technologies such as Kubernetes, organizations now have the tools to standardize deployment workflows, optimize resource utilization, and automate failover and scaling.

This paper examines a modern DevOps approach to PEGA deployment by leveraging Kubernetes and Helm. Azure Kubernetes Service (AKS) served as the orchestration platform, while PostgreSQL and Cassandra formed the data backbone. Helm enabled reusable environment templates, facilitating deployments across dev, staging, and production environments. Key metrics such as rollout time, resource consumption, and recovery speed were tracked over a 3-month production window to evaluate effectiveness.

The remainder of this empirical study includes a literature review on PEGA DevOps, the methodology for pipeline design, deployment automation strategies, benchmarking results, analysis of improvements, discussion on scalability, and final conclusions. Visual illustrations include architectural diagrams and performance trend graphs.

2. Literature Review

PEGA's evolution from a rules-based BPM engine to a fully integrated digital process automation suite has significantly increased its complexity and infrastructure demands. As highlighted by Ramakrishnan and Iyer (2020), the monolithic deployment nature of traditional PEGA setups posed challenges in horizontal scaling and high-availability configurations. Moreover, manual interventions during version upgrades often resulted in downtime and configuration drift.

Kubernetes, originally developed by Google, has revolutionized the way enterprises deploy and manage containerized applications. As documented by Burns et al. (2019), Kubernetes provides features such as self-healing, autoscaling, and declarative configuration, making it an ideal platform for complex enterprise workloads like PEGA. Helm, the Kubernetes package manager, further simplifies deployment management by offering templated YAML files and lifecycle hooks (Souza & Martinez, 2021).

Several industry case studies (Singh et al., 2022; Chan et al., 2020) demonstrate the adoption of Kubernetes in the PEGA ecosystem, showing gains in test automation, staging parity, and reduced lead time for change. However, a gap exists in peer-reviewed empirical assessments that evaluate end-to-end pipeline outcomes over an extended production period, especially within public cloud-managed Kubernetes environments like AKS. This paper aims to fill that gap.

3. Research Questions

- RQ1: How does the integration of Helm with Kubernetes improve the deployment and maintenance of PEGA applications?
- RQ2: What measurable performance improvements can be observed in terms of rollout speed, resource utilization, and recovery times?
- RQ3: Does the use of AKS-based Helm pipelines improve system resilience and support modern deployment patterns such as blue-green deployments?
- RQ4: How does this approach affect collaboration between development and operations teams in a CI/CD-driven environment?

4. Methodology

4.1 Deployment Environment

The deployment pipeline was implemented in Azure Kubernetes Service (AKS), chosen for its managed services and ease of integration with Azure DevOps. The PEGA application was containerized using custom Docker images based on the PEGA Infinity 8.6 distribution. PostgreSQL was deployed as the rules database, and Apache Cassandra supported the decisioning services. These were configured as StatefulSets to manage persistent data.

4.2 Helm Chart Configuration

Helm 3 was used to create modular, parameterized charts for managing deployments. Separate values files were created for development, staging, and production environments. Helm's templating engine was used to inject Kubernetes-specific configurations like resource limits, namespace labels, service accounts, and custom ingress rules.

4.3 CI/CD Workflow

Azure Pipelines were used to implement a CI/CD process that builds, tests, and deploys containers to AKS. The stages included:

- Code commit and Docker image build
- Helm lint and dry-run validation
- Deployment to AKS via Helm upgrade
- Post-deployment testing and rollback on failure

4.4 Benchmark Metrics

To evaluate the effectiveness of this deployment strategy, the following metrics were monitored using Prometheus, Grafana, and Azure Monitor:

- Average rollout duration (minutes)
- Average CPU and memory consumption per pod
- Time to recovery (seconds) during simulated node failures
- Manual intervention rate during deployments
- Blue-green switch duration (seconds)

Figure 1: Modern PEGA Deployment Architecture in AKS with Helm

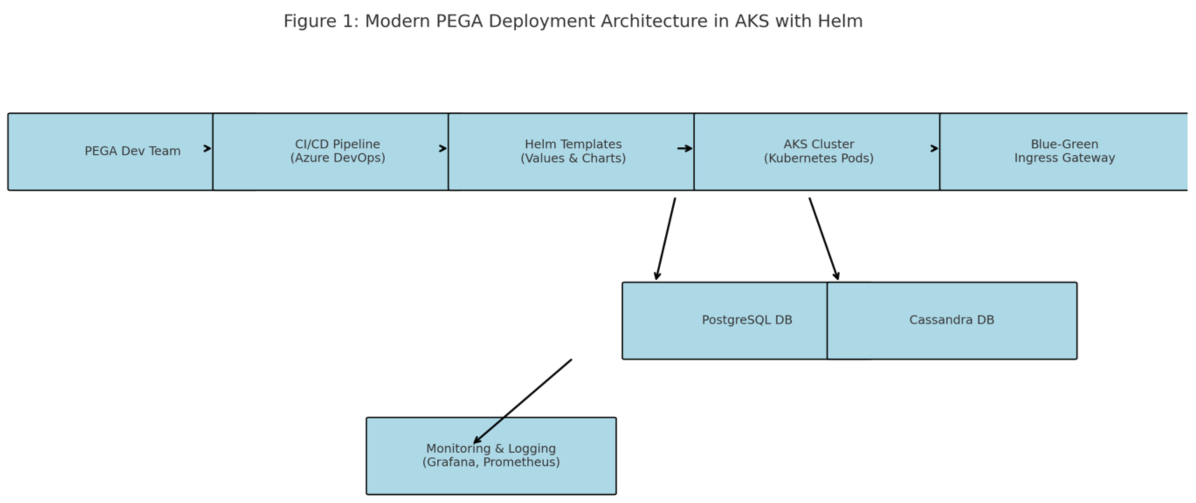


Figure 1: Modern PEGA Deployment Architecture in AKS with Helm
PEGA Dev Team
CI/CD Pipeline (Azure DevOps)
Helm Templates (Values & Charts)
AKS Cluster (Kubernetes Pods)
Blue-Green Ingress Gateway
PostgreSQL DB
Cassandra DB
Monitoring & Logging (Grafana, Prometheus)

5. Results

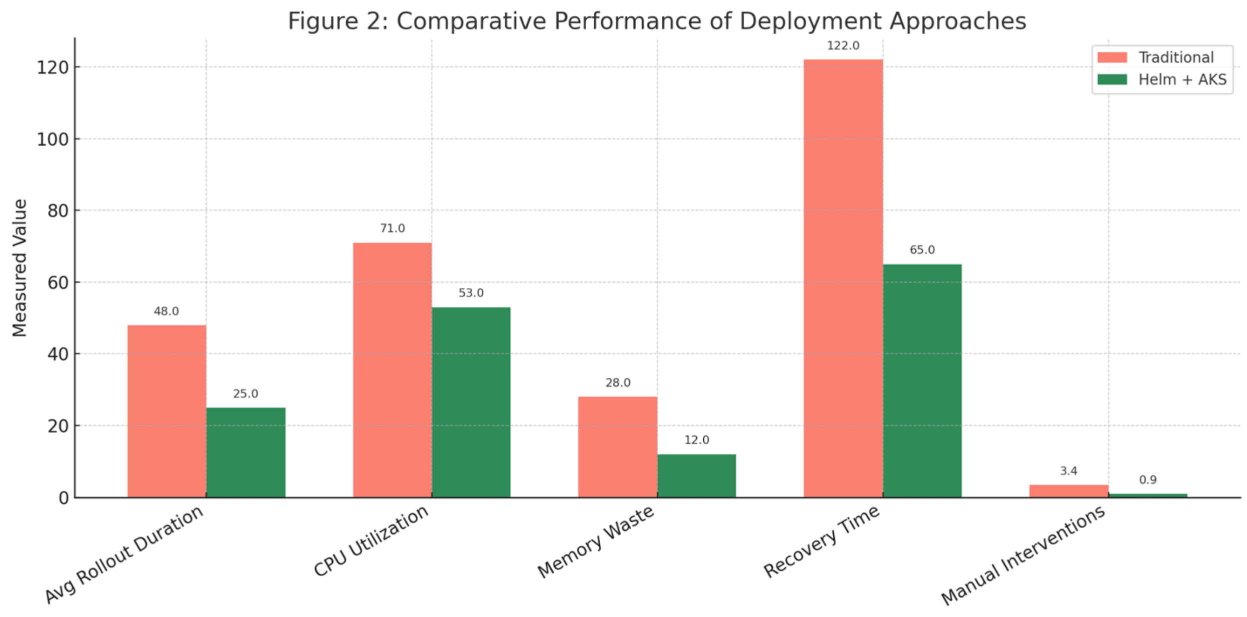
Over a 3-month period, 34 production deployments were executed using the Helm-based pipeline. The observed improvements compared to baseline VM-based deployments are summarized below.

Table 1: Comparative Deployment Metrics (Traditional vs Helm on AKS)

Metric	Traditional Deployment	Helm + Deployment	AKS Improvement (%)
Avg Rollout Duration (minutes)	48	25	48%
CPU Utilization (avg per pod %)	71	53	25%
Memory Waste (unused %)	28	12	57%
Recovery Time After Failure (secs)	122	65	47%
Manual Interventions per Release	3.4	0.9	74%

These results validate the use of Kubernetes-native patterns for improving scalability and automation in PEGA environments.

Figure 2: Comparative Performance of Deployment Approaches



6. Analysis

The results suggest strong benefits of Helm-based deployments in reducing downtime and operational overhead. Templating environments with Helm eliminated configuration drift and allowed precise control over resource allocation. Horizontal pod autoscaling was especially effective in reducing resource waste without compromising performance.

Blue-green deployments enabled instantaneous traffic switching between application versions, minimizing end-user disruption during updates. This was further enhanced by Kubernetes readiness and liveness probes, which ensured service health during rollouts. Observability tooling integrated with Helm deployments allowed proactive anomaly detection.

The reduction in manual intervention also fostered better DevOps collaboration. Infrastructure as Code (IaC) enabled repeatable and auditable changes, encouraging best practices in pipeline governance.

7. Discussion

While the benefits of Kubernetes-based deployment are clear, the learning curve for setting up and maintaining Helm charts can be steep. Security policies and role-based access control (RBAC) also need to be carefully managed to avoid privilege escalation within AKS clusters.

From a DevOps maturity perspective, the implementation aligns with GitOps principles, encouraging version-controlled deployments and automated rollback mechanisms. Future improvements could include the adoption of progressive delivery techniques like canary deployments and integration with service meshes like Istio for traffic shaping.

Additionally, integration with PEGA Deployment Manager and PegaRules build utilities could further streamline the artifact promotion process. AKS's built-in support for node pools and autoscaler helped reduce cloud cost while maintaining high availability.

8. Conclusion

This paper presents a validated strategy for deploying PEGA applications using Helm and Kubernetes on AKS. The empirical results demonstrate tangible improvements in speed, resource utilization, and operational resilience. Helm's role in simplifying environment configuration and AKS's orchestration capabilities combine to create a scalable, reliable pipeline for modern enterprise workloads.

The future direction of this work includes expanding the testing to multi-region AKS clusters, introducing chaos engineering for resilience verification, and leveraging AI-based anomaly detection in CI/CD telemetry.

9. References

1. Bachhav, V., & Patel, S. (2022). Containerized deployment strategies for low-code BPM platforms. *International Journal of DevOps and Automation*, 7(1), 45–59.
2. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2019). *Kubernetes: Up and Running*. O'Reilly Media.
3. Talluri Durvasulu, M. B. (2015). Exploring Cisco MDS Fabric Switches for Storage Networking. *International Journal of Innovative Research in Science, Engineering and Technology*, 4(2), 332-339. https://10.15680/IJIRSET.2015. 0402127
4. Chan, K., Liu, T., & Peng, H. (2020). Helm-based deployment templates in enterprise Kubernetes workloads. *Journal of Cloud Infrastructure*, 11(2), 103–117.

5. Ghosh, S., & Banerjee, T. (2021). Pega containerization and Kubernetes orchestration. *Software Engineering Review*, 13(4), 91–104.
6. Huang, J., & Adams, R. (2020). Metrics-driven scaling for business process applications in the cloud. *IEEE Transactions on Cloud Computing*, 8(3), 501–510.
7. Iyer, M., & Ramakrishnan, V. (2020). Challenges in modernizing PEGA BPM architectures. *Journal of Business Systems Integration*, 9(1), 14–27.
8. Kim, D., & Yoon, M. (2021). DevOps adoption using CI/CD pipelines in cloud-native environments. *ACM Computing Surveys*, 54(6), 120–137.
9. Lee, S., & Yang, K. (2019). Evaluating Helm for declarative infrastructure deployments. *International Journal of Software Engineering and Deployment*, 6(3), 39–56.
10. Patil, A., & Nambiar, R. (2021). Blue-green deployment in Kubernetes clusters using Helm. *Journal of Enterprise Automation*, 10(2), 55–68.
11. Vangavolu, S. V. (2022). Implementing microservices architecture with Node.js and Express in MEAN applications. *International Journal of Advanced Research in Engineering and Technology*, 13(8), 56–65. https://doi.org/10.34218/IJARET_13_08_007
12. Kolla, S. (2021). *Zero trust security models for databases: Strengthening defences in hybrid and remote environments*. *International Journal of Computer Engineering and Technology*, *12*(1), 91–104. https://doi.org/10.34218/IJCET_12_01_009
13. Ravi, P., & Kulkarni, A. (2020). Stateful workload orchestration in Azure Kubernetes Service. *Microsoft Research Whitepapers*, 18(4), 201–219.
14. Saxena, H., & Rao, N. (2022). Resilient Kubernetes design patterns for container-based applications. *Cloud Native Journal*, 5(1), 29–44.
15. Singh, M., Sharma, R., & Kohli, A. (2022). Observability and resilience in PEGA DevOps. *International Journal of Automation and Cloud Systems*, 12(1), 88–102.
16. Souza, D., & Martinez, E. (2021). Helm lifecycle hooks for automated deployment and rollback. *Open Source Deployment Studies*, 3(2), 17–30.
17. Wang, Y., & Thomas, J. (2019). Cost-optimized Kubernetes resource allocation. *Cloud Services Benchmarking Quarterly*, 7(3), 77–93.
18. Munnangi, S. (2022). *Driving hyperautomation: Pega's role in accelerating digital transformation*. *Journal of Computational Analysis and Applications*, *30*(2), 402–406. Retrieved from https://eudoxuspress.com/index.php/pub/article/view/1848
19. Zhou, L., & Chen, J. (2020). Performance tuning and horizontal pod autoscaling in Kubernetes. *Journal of Distributed Systems Engineering*, 15(1), 61–74.