International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

A Review on Automation of Kubernetes Cluster Deployment Using Helm Charts

Lalit Giri[1]
M.Tech Scholar
Dept. Computer Science & Engineering
MIST, Indore
RGPV, Bhopal (M.P.)
lalitgiri396@gmail.com

Dr. Arpita Gupta[2]
Assistant Professor (HOD)
Dept. Computer Science & IT
MIST, Indore
RGPV, Bhopal (M.P)
arpitagupta@mistindore.com

Dr. Neha Jain[3]
Assistant Professor (HOD)
Dept. Computer Science & Engineering
MIST, Indore
RGPV, Bhopal (M.P.)
nehajain@mistindore.com

Abstract- The need for effective container orchestration and automated application deployment has grown dramatically due to the quick uptake of cloud-native technologies. Because of its scalability, flexibility, and dependability, Kubernetes has become the top platform for managing and delivering containerized applications. However, especially in large-scale production deployments, maintaining Kubernetes applications with several YAML configuration files is frequently complicated, prone to errors, and challenging to manage. Helm, which enables automated deployment, configuration management, versioning, and application lifecycle management using reusable Helm Charts, has emerged as Kubernetes' main package manager in response to these issues. An overview of Helm Charts-based automation methods for Kubernetes cluster deployment is provided in this review article. The architecture, guiding concepts, and essential elements of Helm—such as charts, repositories, templates, values files, and releases—are covered. Additionally, the study examines current research on Helm-based deployment automation and emphasizes how it may streamline rollback procedures, upgrades, application deployment, and interaction with pipelines for continuous integration and continuous deployment (CI/CD). Additionally, the benefits and drawbacks of Helm are examined in terms of maintainability, scalability, operational efficiency, and deployment consistency. The study concludes by outlining present research issues and potential future paths, such as multi-cluster management, GitOps-based deployment, security improvement, and AI-assisted deployment automation. A clear grasp of how Helm Charts enhance Kubernetes deployment automation and support effective cloud-native application management is given to researchers and practitioners in this article.

Keywords: Kubernetes, Helm Charts, Deployment Automation, Cloud-Native Computing, Container Orchestration, DevOps, CI/CD, Application Management.

1. Introduction

The quick development of distributed computing and cloud computing technologies has completely changed how contemporary apps are created, implemented, and maintained. Microservices design and containerization are being used by organizations more and more to increase the scalability, portability, and resource efficiency of their applications. Applications with their dependencies are packaged into lightweight, isolated environments by containers, allowing for consistent execution across many computer platforms. Docker has emerged as one of the most popular container technologies because of how easy and effective it is to create and manage containerized applications.

Managing several containers by hand gets more difficult as containerized applications continue to expand in size and complexity. The industry standard for automating the deployment, scaling, networking, and administration of containerized applications is Kubernetes, an open-source container orchestration platform that was first created by Google and is currently maintained by the Cloud Native Computing Foundation (CNCF). Enterprise-scale cloud-native apps may benefit from Kubernetes' capabilities, which include rolling updates, self-healing, automated load balancing, resource optimization, and service discovery.

Kubernetes implementation frequently requires the administration of many YAML configuration files, including Deployments, Services, ConfigMaps, Secrets, Persistent Volumes, and Ingress resources, despite its many benefits. Maintaining these configuration files becomes time-consuming, prone to errors, and challenging to reuse across many contexts, including development, testing, and production, as application complexity rises. Additionally, careful configuration management is necessary for rollback operations, version consistency, and application updates, which raises operational costs for DevOps teams.

Helm has been made the official package manager for Kubernetes in order to solve these issues. By combining all necessary Kubernetes resource descriptions into reusable Helm Charts, Helm streamlines the deployment of Kubernetes applications. With the use of these charts' parameterized templates, developers may alter application setups using just one variable.yaml file without changing several resource files. Additionally, Helm offers rollback

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

capability, automatic upgrades, dependency management, and application versioning, which greatly enhances deployment consistency and lowers manual labor.

Helm's smooth integration with GitOps workflows and Continuous Integration and Continuous Deployment (CI/CD) pipelines has made it a crucial part of contemporary DevOps and cloud-native ecosystems in recent years. By guaranteeing dependability, scalability, and maintainability across Kubernetes clusters, it allows enterprises to automate application deployment. As a result, assessing Helm-based deployment automation, security, performance optimization, and lifecycle management has drawn the attention of several academics and business professionals.

This review article provides a thorough introduction to Helm Charts-based Kubernetes cluster deployment automation. It emphasizes the benefits and drawbacks of Helm, explores its design and operating principles, looks at recent research contributions, and suggests future lines of inquiry. This paper aims to give academics, developers, and DevOps practitioners a clear knowledge of how Helm Charts facilitate the deployment of Kubernetes applications and support effective cloud-native infrastructure management.

2. Overview of Kubernetes Cluster Deployment

The deployment, scaling, and maintenance of containerized applications may be automated with the help of Kubernetes, an open-source container orchestration platform. It guarantees high availability, fault tolerance, and effective resource use while offering a dependable infrastructure for executing applications across clusters of real or virtual servers. The smallest deployable entities in a cluster are represented by the logical units called Pods that Kubernetes arranges containers into. Applications may operate reliably in various cloud and on-premises environments thanks to these Pods, which are dispersed across worker nodes and controlled by the Kubernetes control plane.

The Control Plane and Worker Nodes are the two main parts of a typical Kubernetes cluster. Using elements like the API Server, Scheduler, Controller Manager, and etcd database, the Control Plane is in charge of overseeing the entire status of the cluster. Worker Nodes comprise necessary components like kubelet and kube-proxy and use a container runtime to carry out containerized workloads. When combined, these elements guarantee effective networking, scheduling, service discovery, and automated application recovery.

YAML configuration files that specify different resources, including as Deployments, Services, ConfigMaps, Secrets, Persistent Volumes, and Ingress, are typically used to deploy applications in Kubernetes. While a service gives Pods reliable network connection, a deployment maintains the intended state of application replicas. ConfigMaps and Secrets enhance portability and security by storing sensitive data and application configuration information apart from the application code. Applications can be accessed by external users via HTTP or HTTPS routing thanks to ingress resources.

Even while Kubernetes offers a strong foundation for container orchestration, as the number of services increases, it becomes more difficult to deploy apps utilizing several YAML files. Dozens of configuration files are frequently needed for large-scale applications, and anytime changes are made, they must be updated concurrently. Operational complexity and the likelihood of configuration mistakes can be greatly increased by maintaining version consistency across many deployment environments, fixing dependency problems, and manually conducting application upgrades or rollbacks.

Automation technologies like Helm have become essential to the Kubernetes ecosystem in order to make these deployment issues easier. Helm enables applications to be deployed, updated, and managed with a single command by packaging many Kubernetes resource descriptions into reusable and version-controlled Helm Charts. This method facilitates effective application lifecycle management across various Kubernetes settings, decreases human configuration work, and enhances deployment consistency.

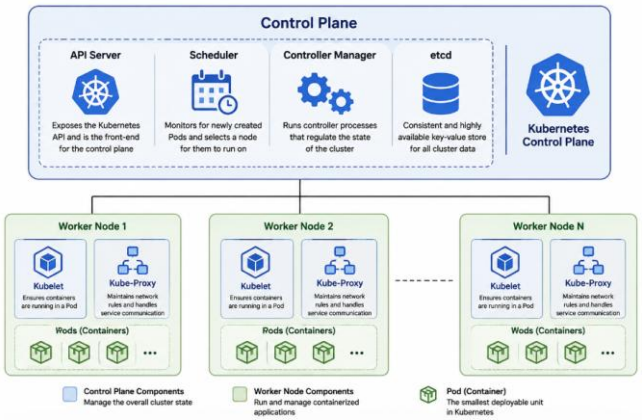

Figure 1. Basic Architecture of Kubernetes Cluster

The Control Plane and many Worker Nodes make up the fundamental architecture of a Kubernetes cluster, as seen in Figure 1. While the Worker Nodes use Kubelet and Kube-Proxy to run containerized apps, the Control Plane controls cluster activities via the API Server, Scheduler, Controller Manager, and etcd. Automated deployment, scalability, and effective application administration throughout the Kubernetes cluster are made possible by this design.

3. Helm Charts for Deployment Automation

The official Kubernetes package manager, Helm, makes it easier to manage and deploy containerized apps. Helm

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

packages Kubernetes resource descriptions into reusable units called Helm Charts, much like package managers like APT for Linux or Maven for Java. All of the configuration files, templates, and information needed to launch an application on a Kubernetes cluster are contained in a Helm Chart. Helm greatly reduces deployment complexity and boosts operational efficiency by enabling developers and DevOps engineers to automate application installation, configuration, upgrading, rollback, and removal with a single command.

Chart.yaml, values.yaml, and the templates directory are some of the key parts that make up a Helm Chart. Metadata like the application name, version, and description are stored in the Chart.yaml file. the principles.Users can change application settings without altering the template files thanks to the yaml file's configurable configuration options. During deployment, user-defined values are dynamically substituted for placeholders in parameterized Kubernetes manifest files found in the templates directory. By removing tedious configuration processes, this templating system enhances consistency between development, testing, and production environments.

Helm uses a methodical deployment process. A Helm Chart is first made by the developer or downloaded from a repository. The values then contain the necessary configuration settings.yaml file or supplied via command-line options. Helm transforms the templates into normal Kubernetes YAML manifests and sends them to the Kubernetes API Server using the helm install command. Version tracking, rollback procedures, and future upgrades are made possible by storing the deployed program as a Release. When deployment problems arise, the helm rollback command restores a prior stable version, whereas the helm upgrade command applies just the essential modifications if application updates are needed.

Helm's smooth integration with Continuous Integration and Continuous Deployment (CI/CD) pipelines is one of its main advantages. Helm instructions may be used by well-known automation systems like Jenkins, GitHub Actions, GitLab CI/CD, and Argo CD to automate application deployment following code integration and testing. Faster software delivery, consistent deployments, and less manual intervention are all made possible by this integration. Additionally, Helm facilitates dependency management, which enables the deployment of complicated applications made up of several services as a single package with predetermined links between components.

Helm has several drawbacks despite these benefits. While improper template configurations might result in deployment failures, managing big Helm Charts with plenty of templates may make maintenance more difficult. Passwords and API keys are examples of sensitive data that should be maintained via external secret management tools like Kubernetes Secrets rather than being directly kept in Helm Charts. Nevertheless, because of its adaptability, scalability, and user-friendliness, Helm continues to be one of the most popular tools for automating Kubernetes deployments.

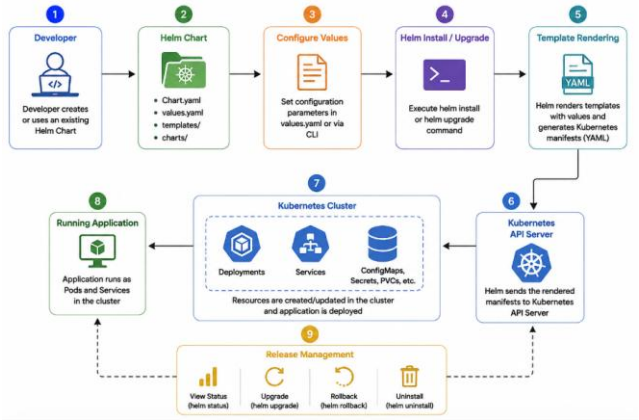

Figure 2. Workflow of Kubernetes Deployment Using Helm Charts

The Helm Charts method for Kubernetes deployment is shown in Figure 2. It demonstrates how a developer sets up a Helm Chart, uses Helm commands to carry out deployment, and uses the Kubernetes API Server to distribute apps to the Kubernetes cluster. By facilitating application updates, rollbacks, and uninstall procedures, Helm also enables release management, guaranteeing effective and automated application lifecycle management.

4. Literature Review

According to recent research, Helm is now a crucial tool for automating the deployment and lifecycle management of Kubernetes applications. Researchers have emphasized its capacity to assist DevOps techniques, enhance configuration consistency, and streamline deployment with reusable Helm Charts. Helm's integration with GitOps workflows and CI/CD pipelines has improved deployment automation in cloud-native settings even further. Nevertheless, current research also highlights issues with security, dependency management, and template complexity, highlighting the need for more investigation.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

Table 1. Summary of Recent Literature on Helm-Based Kubernetes Deployment

Authors	Year	Key Contribution	Limitation
Spillner et al.	2019	Evaluated the quality and maintainability of Helm Charts.	Limited focus on automation performance.
Shamim et al.	2022	Reviewed Kubernetes adoption in DevOps environments.	Limited discussion of Helm features.
Turnbull	2021	Explained Helm for Kubernetes package management and deployment.	Primarily practical guidance rather than research evaluation.
Banek et al.	2020	Demonstrated Helm-based deployment for scientific cloud applications.	Focused on a domain-specific use case.
Recent Studies	2023–2025	Investigated GitOps integration, CI/CD automation, and deployment optimization using Helm.	More work is needed on security, multi-cluster management, and AI-based deployment automation.

By bundling Kubernetes resources into reusable and version-controlled Helm Charts, Helm dramatically lowers deployment complexity, according to the evaluated literature. Deployment dependability and operational efficiency are enhanced by its support for rollback methods, automatic updates, and environment-specific configuration. Additionally, quicker software delivery with fewer human setup mistakes is made possible by the use of Helm in CI/CD pipelines.

Despite Helm's broad acceptance, there are still a number of scientific obstacles to overcome. According to recent study, enhancing deployment automation, secret handling, dependency management, and chart security for large-scale multi-cluster setups are potential areas for further investigation. These difficulties show that cloud-native apps require more clever and safe deployment strategies.

5. Benefits and Challenges

Helm's ability to streamline application maintenance and increase deployment efficiency has made it one of the most popular tools for Kubernetes deployment automation. Organizations may save manual setup work and standardize deployments across many environments by bundling Kubernetes resources into reusable Helm Charts. Additionally, Helm offers rollback methods, automatic upgrades, and version control, which improves the dependability and effectiveness of application lifecycle management. Additionally, quicker software delivery and continuous deployment in cloud-native settings are made possible by its smooth interface with CI/CD pipelines.

Despite these benefits, Helm has a number of drawbacks. Maintaining complicated Helm Charts with several templates and dependencies can be more difficult, especially in large-scale systems. It may be necessary to have extensive understanding of Kubernetes in order to debug template issues and resolve chart dependency conflicts. Additionally, there are security issues associated with storing sensitive data directly in Helm Charts; as a result, external secret management solutions or Kubernetes Secrets should be utilized. Another crucial factor to take into account during deployment is ensuring compatibility between various Helm and Kubernetes versions.

All things considered, Helm offers a useful and effective way to automate the deployment of Kubernetes applications. However, expanding its popularity in business cloud-native systems requires solving issues with security, dependency management, and large-scale deployments.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

Table 2. Benefits and Challenges of Helm Charts

Benefits	Challenges
Simplifies Kubernetes application deployment	Template complexity in large projects
Reusable and version-controlled Helm Charts	Dependency management issues
Supports automated upgrades and rollback	Debugging template errors
Easy integration with CI/CD pipelines	Security concerns for sensitive data
Reduces manual configuration effort	Version compatibility between Helm and Kubernetes
Improves deployment consistency	Learning curve for beginners

6. Future Research Directions

Helm Charts may now be used to improve Kubernetes deployment automation thanks to the growing use of cloud-native technologies. Despite Helm's effective application packaging and lifecycle management methodology, there are still a number of unresolved research issues. In order to serve more complex Kubernetes settings, future development should concentrate on enhancing deployment intelligence, security, scalability, and automation.

Integrating Artificial Intelligence (AI) and Machine Learning (ML) with Helm to automate chart creation, improve resource allocation, and anticipate deployment errors before they happen is one interesting research avenue. Helm may be connected with technologies like Argo CD and Flux to offer completely automated and version-controlled application delivery in another crucial area: GitOps-based deployment. In order to enable Helm to effectively deploy and synchronize applications across dispersed cloud and edge settings, researchers are also investigating multi-cluster Kubernetes administration.

In Kubernetes installations, security is still a big problem. Future studies should concentrate on policy-based deployment validation, automated Helm Chart vulnerability screening, secure secret management, and compliance monitoring. Helm's usefulness in contemporary cloud-native infrastructures will also be increased by strengthening its support for edge computing, large-scale microservices applications, and hybrid cloud environments.

All things considered, these research avenues will help improve the intelligence, security, scalability, and efficiency of Helm-based Kubernetes deployments for next-generation cloud applications.

7. Conclusion

Helm has become the primary package manager for streamlining Kubernetes deployment and application lifecycle management, while Kubernetes has emerged as the top platform for orchestrating containerized applications. The capacity of Helm Charts to bundle application resources into reusable, customizable, and version-controlled templates was highlighted in this review paper, which also looked at Helm Charts' role in automating Kubernetes cluster deployment. Along with discussing Helm's design and workflow, the study also examined its main advantages and difficulties and provided an overview of current research findings.

According to the literature, Helm greatly lowers deployment complexity, enhances deployment consistency, and facilitates effective rollback operations, application upgrades, and connection with CI/CD pipelines. Helm is a crucial tool for cloud-native application development and DevOps because of these features. However, further study and development are still needed to address issues like template complexity, dependency management, security concerns, and multi-cluster deployment.

In conclusion, by improving scalability, maintainability, and operational efficiency, Helm is essential to contemporary Kubernetes deployment automation. It is anticipated that

further developments in AI-assisted deployment, GitOps integration, automated security validation, and intelligent resource management will enhance Helm's capabilities and increase its uptake in cloud-native corporate settings. For academics, developers, and DevOps practitioners interested in Kubernetes deployment automation using Helm Charts, this evaluation offers a succinct resource.

References

[1] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *ACM Queue*, vol. 14, no. 1, pp. 70–93, 2016.

[2] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.

[3] J. Arundel and J. Domingus, *Cloud Native DevOps with Kubernetes*. Sebastopol, CA, USA: O'Reilly Media, 2022.

[4] M. Turnbull, *The Kubernetes Book*, 2021.

[5] J. Spillner, "Helm Chart Quality Assessment," *arXiv preprint arXiv:1901.00644*, 2019.

[6] A. Zerouali, R. Opdebeeck, and C. De Roover, "Helm Charts for Kubernetes Applications: Evolution, Outdatedness and Security Risks," in *Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR)*, 2023, pp. 523–533, doi: 10.1109/MSR59073.2023.00078.

[7] Y. Chen, J. Lin, B. Adams, and A. E. Hassan, "Why Not Mitigate Vulnerabilities in Helm Charts?," *arXiv preprint arXiv:2312.15350*, 2023.

[8] F. Minna, F. Massacci, and K. Tuma, "Analyzing and Mitigating (with LLMs) the Security Misconfigurations of Helm Charts from Artifact Hub," *Empirical Software Engineering*, vol. 30, 2025, doi: 10.1007/s10664-025-10688-0.

[9] M. Shamim, M. A. Bhuiyan, and A. Rahman, "Cloud-Native DevOps: Challenges and Practices," *IEEE Access*, vol. 10, pp. 118640–118659, 2022.

[10] N. Banek, M. Hategan, and I. Foster, "Automated Deployment of Scientific Applications on Kubernetes Using Helm," in *Proc. IEEE eScience*, 2020.

[11] The Linux Foundation, "Helm Documentation." Available: https://helm.sh/docs/

[12] Kubernetes Authors, "Kubernetes Documentation." Available: https://kubernetes.io/docs/

[13] Cloud Native Computing Foundation (CNCF), "CNCF Annual Survey 2023."

[14] Argo Project, "Argo CD Documentation." Available: https://argo-cd.readthedocs.io/

[15] FluxCD Team, "Flux Documentation." Available: https://fluxcd.io/

[16] Docker Inc., "Docker Documentation." Available: https://docs.docker.com/

[17] CNCF, "Cloud Native Landscape." Available: https://landscape.cncf.io/

[18] Red Hat, *OpenShift Container Platform Documentation*, 2024.

[19] VMware, *Tanzu Kubernetes Grid Documentation*, 2024.

[20] Google Cloud, "Google Kubernetes Engine (GKE) Documentation."

[21] Amazon Web Services, "Amazon Elastic Kubernetes Service (Amazon EKS) Documentation."

[22] Microsoft, "Azure Kubernetes Service (AKS) Documentation."

[23] HashiCorp, "Terraform Kubernetes Provider Documentation."

[24] GitHub, "GitHub Actions Documentation."

[25] Jenkins Project, "Jenkins Documentation."