International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

KUBERNETES DEPLOYMENT FOR CLOUD-AGNOSTIC INFRA-LEVEL SERVICES

Balaji Soundararajan
Independent Researcher
esribalaji@gmail.com

Abstract

Multi-cloud strategies have emerged as a critical approach for organizations seeking to enhance business agility, mitigate vendor lock-in risks, and optimize infrastructure costs. Central to these strategies is the adoption of cloud-agnostic infrastructure services, which enable seamless deployment across heterogeneous cloud environments. Kubernetes, as a leading container orchestration platform, plays a pivotal role in enabling such cloud-agnostic deployments by abstracting underlying infrastructure complexities. This paper explores Kubernetes' deployment methodologies for cloud-agnostic infrastructure services, emphasizing resource optimization, high availability, and automated deployment strategies such as blue-green and canary releases. Through case studies, we demonstrate how organizations achieve operational efficiency, reduced deployment times, and enhanced resilience. The findings underscore Kubernetes' capacity to support multi-cloud and hybrid cloud architectures while addressing challenges like configuration management, monitoring, and cross-cloud interoperability.

Keywords: Multi-cloud strategies, Kubernetes, Cloud-agnostic infrastructure, Vendor lock-in, Deployment methodologies, Container orchestration, High availability, Hybrid cloud, Platform-as-a-Service (PaaS), DevOps.

I. INTRODUCTION

Multi-cloud strategies include the use of two or more cloud environments. There are two main reasons for adopting a multi-cloud strategy. Organizations can increase business agility to quickly capitalize on new market demands or make better decisions based on dynamic elements. This approach allows businesses to reduce the risk of strong dependencies on a single cloud supplier. Multi-cloud strategies require a cloud-agnostic infrastructure.
With Kubernetes successfully deployed and running for several years, we can confirm that deployments have been accelerated and flexibility has increased. In addition, deployment across multiple sites in both private and rented data centers has simplified their structure and adjustment. Deploying complex installations does not present a simple framework, requiring considerable experience in implementing these special deployments on a case-by-case basis. Kubernetes' standard implementation can be used to deploy not only basic services and

applications, but also more advanced services within a private or hybrid cloud. With this as a motivation, the guide is to introduce Kubernetes deployment methodology.

II. BACKGROUND AND SIGNIFICANCE

Historical Context Defining the cloud as utility computing foretold a structural shift in computing, a shift towards electricity-like infrastructure services that give users the ability to consume computing resources as services. Since then, the industry has witnessed waves of such infrastructure services on private clouds and, more recently, public clouds. Currently, the third wave consists of hyper-scalers, with leaders providing integrated environments that bundle infrastructure and several platform services.

Trend In spite of the increasing range of devices and platforms on which their customers run services, there is an increasing trend towards building hyper-scalable, cloud-agnostic solutions using Kubernetes and its ecosystem. In large cloud-agnostic solutions built on Kubernetes, customers can leverage scale rather than forcing replications at the service level across every cloud and fine-tuned WAN connections, or a 'service mesh' on top of unstandardized load balancers and more. In smaller contexts, software vendors can leverage this hyper-scale and proliferate application support quickly.

Significance Untethered solutions that run as cloud-agnostic 'applications' can be beneficial, such as operational efficiency; all infrastructure or 'platform' configuration is brought along with each installation, elastic scale across several premises, public and private, to accommodate a wide range of network and user conditions, and the ability to prove a solution in a public cloud but then rapidly make that a hybrid or wholly private solution for large-scale roll-out. Common barriers to scaling those applications via hosted, vendor-controlled services or unmanaged, packaged software solutions include ensuring sufficient yet lean user data storage and accessing production 'on-prem' data within certain corporate network environments. [1]

III. UNDERSTANDING CLOUD-AGNOSTIC INFRASTRUCTURE SERVICES

To clarify the research question, we need to first define what we mean by cloud-agnostic infraservices, which are an evolved concept in comparison to simply cloud-agnostic services. In today's world, where cloud computing has become almost a commodity and all-public cloud is a dominant delivery model, not only do we have to take care to avoid vendor lock-in of services across cloud providers, but also vendor lock-in of cloud providers themselves. Cloud-agnostic services are not tied to any single provider and can work across these platforms. Cloud-agnostic services can offer greater cost-effectiveness and flexibility if unique features and services are not used. Cloud-agnostic identity providers, computing, storage, databases, and even Kubernetes-based PaaS are some of the relevant examples. Moreover, curated Kubernetes is one such cloud-agnostic infra-level service across leading enablers.

From an enterprise perspective, cloud-agnostic infra-level services can offer several advantages when pursuing an application portfolio modernization strategy or breaking the application or microservices modernization bottleneck. One usage scenario is adopting a "Platform-Agnostic Computing Agile" strategy. This scenario offers an advantage when organizations, and especially CxOs, have to modernize a significant number of their existing APIs and applications by building cloud and mobile experiences, and need to make a choice of "To Which Cloud(s)" or "When to Migrate to Which Other Cloud." In practice, the ease of retaining or replacing the same Kubernetes infra-layer service should offer a lower total value of ownership of the solution and/or greater architectural flexibility across the above strategic decision dimensions. The second use scenario centers around the more general decision frameworks canvassing prospective container PaaS choices in an "Enterprise Open Source Strategy." Additionally, the paper posits that managing PACAs is a core part of managing vendor lock-on value management throughout an organization's enterprise architecture and solution lifecycle. This work makes a case for PACA management and curated K8s-as-PaaS from a CxO and perspective, by complementing existing academic work assessing technology form and fit for IT professionals adopting a tech-savvy cloud-agnostic-anything strategy. As per our rigorous case argument perspective, the K8s-as-PaaS solution offers the greatest overall strategic value across a concept sample of fit-for-purpose, end-of-life, perceived-value add, and all-natural cloud-agnostic K8s-as-PaaS fitment cases.

3.1 Definition and Benefits

There is no standardized definition of cloud-agnostic infrastructure; the term is used in association with various services to denote a few common characteristics. Entities offering cloud-agnostic infrastructure services usually portray their solutions as being "cloud neutral," "multi-cloud," "unbound," and so forth. Any service designed to mitigate the lock-in effect by providing a uniform access and control layer to cloud resources in a manner that is agnostic to either "where" or "what" resources are used can be termed cloud-agnostic. Using cloud-agnostic services when deploying new applications in different clouds may lead to easy resource maintenance, flexibility and choice in infrastructure solutions, reduced costs using pay-as-you-go solutions, agility in growing the application according to user demands, and for multi-cloud or hybrid scenarios, better performance and redundancy to serve users better.
Cloud-agnostic data or services are not constrained by the limitations or requirements of any particular cloud provider. Open-source technologies also enhance the benefits by offering portable solutions to build and deploy applications that can run from cloud to edge or even bare metal. Moreover, offering a cloud-agnostic version of the infrastructure as a service at the platform layer may result in cheaper infrastructure for service providers, enhance the scalability for deploying the services worldwide, and reduce operation costs. Offering DNS service using a cloud-agnostic management service is an example of a cloud-agnostic infra-service that can potentially provide more benefits than being on the cloud.

IV. KUBERNETES OVERVIEW

The emergence of the cloud era turned out to be a catalyst in developing and enhancing

infrastructure to tackle the scale of services and operational management. Though cloud services have been one of the successful paradigms for doing the same, it leads to the necessity of maintaining the services across multiple clouds to avoid vendor lock-in. This is why orchestration platforms like Kubernetes came into existence. Kubernetes has evolved with the same motive: to manage containers over multiple hosts.
Kubernetes is an open-source platform designed to automate the deploying, scaling, and operation of application containers. It allows defining the containers that make up the respective application and the resources to be allocated to each of them. The goal of Kubernetes is to continue with what cloud computing has promised, that is, to hide the complexity of running applications, operating infrastructures, and providing a platform on which developers can ideally run applications using the same principles anywhere. The Kubernetes system is structured in a two-level architecture consisting of the master and nodes. The Kubernetes master instance provides a unified control plane for its associated distributed set of worker nodes. Each node can potentially support multiple application-level controllers, each using the primary API to communicate with the cluster.
A service is responsible for defining a set of pods and a policy to access these pods. Pods are the most basic components that hold the containerized applications in a Kubernetes cluster. Clusters are a set of nodes that form the infrastructure to run applications in containers. A Kubernetes node can be either a physical machine or a virtual machine. It is used to run one or more pods. A pod is a group of one or more containers, with shared storage/network, and a specification of how to run the containers. Each pod in the cluster gets its own unique IP address to serve as its identity, and this IP address is used to manage and communicate with the pod. In every Kubernetes cluster, traffic is automatically load balanced across all nodes. The Kubernetes cluster scales seamlessly without additional configuration using a single command. As the premier open-source container manager, an active ecosystem contributes to defining the norms and directions the containerized world is headed with Kubernetes at the center of this automation. [2]

4.1 Key Concepts and Components

Kubernetes follows an architecture that is centered around atomic units called pods. A pod is a set of containers that work together in order to accomplish a system's functionality. Containers within the pod share resources such as volumes and metadata, effectively enabling inter-container communication and allowing all containers in the pod to share the pod's IP. Pods are part of a larger resource called a Replica Set. The Replica Set is a tool used to control the instances of running pods and corresponds to the actual deployment in terms of operational management. Services in Kubernetes are an abstraction layer to manage a set of pods in order to provide a single access point to them, bestowing some important load-balancing capabilities and a self-healing feature to ensure availability. The deployment is a collection of multiple services. In Kubernetes, the deployment is used to deploy multiple consuming services or advance services. While containers and container orchestrators make up the top level of abstraction, underneath that layer is where the virtual host infrastructure is managed. Pods themselves must be deployed within a namespace. Namespaces encompass how to control a set of resources, which includes pods and the link to virtual hosts through services. This can scale to thousands of pods within a distributed federation with control over basic infrastructure such as CPU instances and volumes. Taken separately, the orchestration and management of these components is achieved by using checkpoints. Each container will automatically create and delete all the required value resources. For orchestration, there are two types of microservices that Kubernetes composes. As explained earlier, there are consuming microservices. These consuming services were deployed using Kubernetes deployments. Kubernetes deployments define the number of running pods or replicas of consuming services desired by the user to be running in the cluster at some point. Pods in the deployment are then associated with a service. Independently, the resources required by a service to store persistent data of undetermined size and/or to share data are requested using Kubernetes persistent volume claims. When a PVC is created, it creates bonds between the PVC, the actual storage class used in the cloud, and the pod that is using this volume. Such volumes can be accessed outside the lifespan of the pod for resilience. Finally, a set of resources is also needed to offer networking, also through a dedicated Kubernetes resource, to a mechanism called an in-cluster service. An in-cluster service is a beneficial container attribution because it is by default reliably available between the receiving client and the consumer.

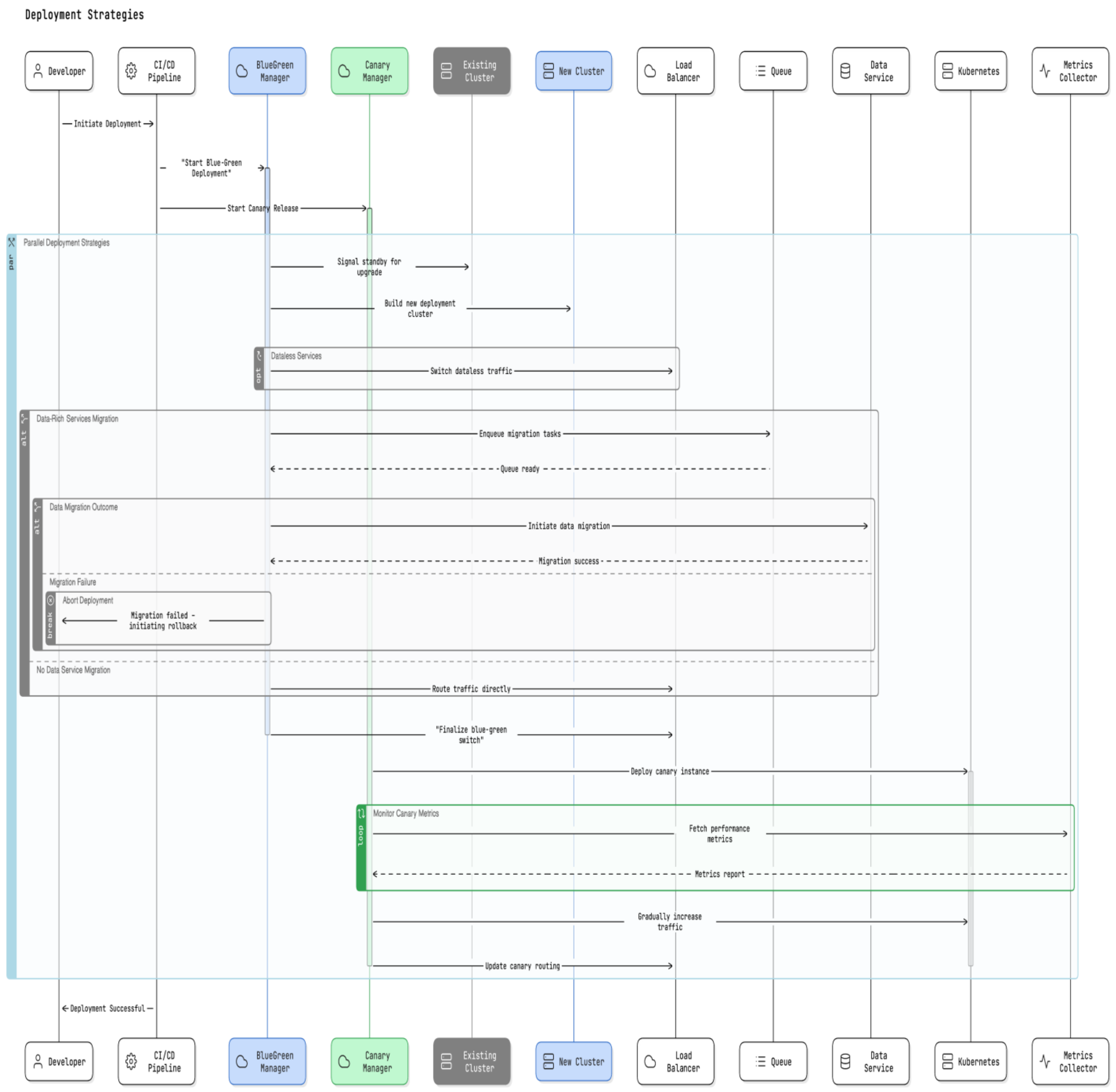
V. DEPLOYMENT STRATEGIES FOR CLOUD-AGNOSTIC INFRA-LEVEL SERVICES

There can be different deployment strategies to avoid cloud-provider-specific managed services, and cloud-agnostic infra-level services need to consider at least a few important strategies. Multi-region deployment is a basic method to support services to be tolerant of cloud failures and ensure that data in different clusters are eventually consistent. This is useful for cloud-agnostic services, but it is not the most efficient way to serve users from different geographical zones. Blue-green deployment is frequently used for systems or applications with low deployment complexity. A second deployment cluster is built from scratch alongside the original one. Once the new cluster is built, dataless services are switched to it first. Data-rich services are then migrated with a queue-like mechanism used to ensure that requests can be handled during the migration. When most user-facing services are migrated to the new cluster, a last migration is initiated to ensure the data is consistent in the new cluster. Because both clusters are fully functional and accessible temporarily, there is no stop of request serving throughout the two deployments.

For longer deployment times, a related strategy is the concept of canary release. In canary release, or phased rollout, is a technique to reduce the risk of introducing a new software version in production by gradually rolling out the change to a small subset of users before increasing the rollout to the entire user base and fully committing to the new version. Blue-green and canary deployments are usually supported by a set of automation tools. A package manager for Kubernetes is often used for CI/CD integration. Charts can be used to define, install, and upgrade even the most complex applications. It also supports versions to control the

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

upgrade process. Although manual strategies allow for better control, they are not feasible when considering the frequency of deployments in a DevOps environment. As a result, we will pursue further discussions from the perspective of fully automated canary and blue-green deployment using Kubernetes.



While a certain development strategy may fit one scenario well, it is unlikely to be suitable for another. The selection of a specific deployment strategy depends on various levels, from application design and business requirements to user geography and technical team capabilities. Blue-green deployment and canary release are two widely used deployment strategies. In a practical environment, they may also be combined according to particular situations and considering individual components of a cloud-agnostic infra system. Companies are investigating the option of multi-cloud deployments for their enhanced risk management characteristics. As the system grows, existing deployment strategies may not satisfy the need to guarantee little or no service downtime. The capabilities in a loose-coupled cloud-agnostic infra-level service may have to be diverged. In this section, two deployment strategies, named blue-green deployment and canary release, have been considered. Each strategy has its advantages and limitations, and will perform differently in practice. [3]

VI. KUBERNETES DEPLOYMENT BEST PRACTICES

Resource Optimization

Proper resource management ensures a smooth operation of Kubernetes. These optimizations should provide a reduced need for over-provisioned resources on the infrastructural layer and ensure that the application layer can rely on the resources it expects from the system, allowing it to be more robust and more responsive to changes in load.

High Availability

Ensuring high availability of deployed applications is crucial. Being a container orchestrator, Kubernetes leverages the abilities of modern infrastructure provisioning. The application can also use this capability to be self-healing.

Configuration Management

Choosing the right configuration management system is crucial to have a maintainable, replicable, and reliable deployment. Configuration management should simplify and automate the configuration and management of servers to the greatest extent possible. In terms of defining the configuration templates and allowing reuse across different environments, charts are valuable.

Monitoring and Diagnostics

To maintain a comprehensive view of the state and execution of the deployed application, observability is crucial. It is thus important to include in the deployment environment a monitoring system that is capable of collecting and analyzing logs and performance data in real time, performing automated calculations to achieve useful insights. In open-source environments, several technologies exist. These tools enable real-time monitoring by identifying key metrics to track, alerting on anomalies or errors, and visualizing aggregated data to identify patterns and bottlenecks within the application. By correlating logs and metrics from both the application and the surrounding infrastructure, operators can analyze and detect which parts of the deployment are affected. These insights can be utilized as input for more complex observability systems or artificial intelligence systems. [4]

VII. CASE STUDIES AND USE CASES

Below we present a collection of various organizations that have successfully adopted cloud-agnostic strategies through the use of Kubernetes and infrastructure projects. We provide context for each use case, illustrate the challenges that the organization faced, and describe the solutions implemented through Kubernetes and other open-source tooling. Repetitive lessons learned from use cases are summarized at the end of this section.

Bankend AG Services for Storage of Digital Data Assets
Case: Bankend AG Services for Storage of Digital Data Assets requires the provision of a cloud-agnostic infrastructure where Lock Register is deployed. The infrastructure should allow for clustering of database nodes, insertion and deletion of records using a REST-like API, and form more complex queries in near real time.
Metrics: Deployment time dropped from 15 hours to 30 minutes, resilience because of the redundancy, resource usage comparison among different drivers, and custom observation.
Key Lesson: Use objects that are common among many control programs.

Verband Deutscher Maschinen- und Anlagenbau, e.V. RDF and TRKA internal data
Case: VDMA is running its business utilizing Docker Swarm. As customers transition to newer cloud-native application ecosystems, VDMA requires a cloud-agnostic infrastructure to continue its services; in this case, connection to multiple data sources.
Metric: Service starts in less than 10 seconds, up-to-date resources and metrics, cloud performance compared, requested data doesn't exist, a factory node can be adapted, new trends in the incoming queries with and without requested course requests.
Key Lesson: Ensure that the objects are easily translatable to infrastructure code, and implement the long-term abstraction requirements.

VIII. CONCLUSION

The adoption of Kubernetes as a cloud-agnostic infrastructure service platform offers organizations unparalleled flexibility in deploying and scaling applications across multi-cloud and hybrid environments. By leveraging Kubernetes' orchestration capabilities enterprises can avoid vendor lock-in, optimize resource utilization, and achieve high availability through automated deployment strategies such as blue-green and canary releases. Case studies illustrate significant reductions in deployment times (e.g., from 15 hours to 30 minutes) and improved resilience via redundancy and dynamic resource management. Best practices, including robust configuration management, real-time monitoring, and infrastructure-as-code principles, further enhance operational efficiency. As cloud environments continue to evolve, Kubernetes' ecosystem remains central to enabling agile, cost-effective, and scalable solutions. Future work should focus on advancing interoperability standards and integrating AI-driven observability tools to further streamline multi-cloud operations.

REFERENCES

1. Korontanis, K. Tserpes, M. Pateraki, L. Blasi, "Inter-operability and orchestration in heterogeneous cloud/edge resources: The accordion vision," ... and Application ..., 2020. cnr.it
2. A. A. Khatami, Y. Purwanto, "High availability storage server with kubernetes," in Technology Systems and ..., 2020. [HTML]
3. Featherstone, M. (2018). Blue-Green Deployment and Canary Releases: Strategies for Zero-Downtime Deployment. IEEE Software, 35(3), 89–92.
4. M. Usman, S. Ferlin, A. Brunstrom, and J. Taheri, "A survey on observability of distributed edge & container-based microservices," IEEE Access, 2022. ieee.org