



Enterprise Kubernetes Management Requirements

Deploying, operating and securing
Kubernetes clusters and their applications
in a multi-cluster multi-cloud world

WHITE PAPER

CONTENTS

- 03** INTRODUCTION
- 04** MULTI DISTRIBUTION
- 05** MULTI CLOUD
- 06** MULTI PERSONA
- 07** APPLICATION OPERATIONS
- 08** VISIBILITY, GOVERNANCE
& SECURITY
- 09** CONCLUSION
- 09** THE RAFAY KUBERNETES
MANAGEMENT CLOUD

INTRODUCTION

What are the key components required for your multi-cluster management solution?

For companies modernizing their applications, it is a foregone conclusion that Kubernetes is the de facto container orchestration layer for managing containerized applications. Regardless of how and where you deploy Kubernetes clusters, many first-time users think running one or two Kubernetes clusters will be sufficient for their company's production needs. Yet, there are actually many reasons that will drive your organization to eventually run many separate clusters instead, including:

- **Security Requirements:** A set of services may have unique hardening and data retention requirements. It may make sense to deploy these services in a hardened cluster with stringent auditing, authorization, permissions management and logging policies. But doing this in every cluster may lead to unnecessary slowdowns and overhead.
- **Scaling Requirements:** If a few services have massive scaling requirements, it may be best to deploy them into smaller dedicated clusters to protect against potential "noisy neighbors" services.
- **Blast Radius:** Every time Kubernetes or a supporting component (e.g. service mesh or metrics collection packages) is upgraded, other services may need to be updated to work with the new version. This type of upgrade can impact schedules across multiple teams and is minimized by running smaller clusters.
- **Integration Requirements:** Some services may need a special admission controller or high-speed storage. Services that need such integrations may be best grouped together in clusters that are pre-configured with required packages or the right storage class, while other services can be deployed on clusters running vanilla Kubernetes.
- **Custom Enhancements:** Some services may lead the DevOps team to develop enhancements to Kubernetes. To protect against unforeseen side effects, these services can be deployed on customized clusters.
- **Network Load Requirements:** Services that are expected to drive high network load (by way of Kubernetes API calls) are best deployed on separate clusters to protect other services that may get starved otherwise.

Thus, running and managing multiple clusters is not an 'if' for your organization. And with a growing cluster fleet that may span multiple clouds, data centers and Kubernetes distributions, Site Reliability Engineering (SRE) and IT Ops teams need the right tools and services to manage these fleets.

This whitepaper will examine the elements required to ensure your Kubernetes deployments are positioned to meet your organization's current & future requirements for all aspects of Kubernetes lifecycle management.

The elements are:

- Multi Distribution
- Multi Cloud
- Multi Persona
- Application Operations
- Visibility and Governance
- Data Encryption and Security



Distributions are the building blocks and tools enterprises need in order to build a Kubernetes solution. Nothing is pre-built or pre-packaged.

MULTI DISTRIBUTION

It is important to realize there are many different versions (also known as distributions or 'distros') of Kubernetes. These distros share many common Kubernetes features, yet they all have their own unique tools for installing and working with Kubernetes. While standardizing your company on a single Kubernetes distro may seem desirable, achieving this is hard. Organizations run different distros for a number of good (and not-so-good) reasons including:

- **Mergers and Acquisitions:** Acquired entities might be running a different Kubernetes distro than their new parent.
- **Distributed and Siloed Teams:** Individual teams tend to adopt a Kubernetes distro and stick with it. As teams get reorganized and integrated, the result may be that several Kubernetes distros are in active use across an organization.
- **Technology Adoption Timeframes:** Similarly, teams who have adopted Kubernetes at different times may have standardized on different distros.
- **Vendor Preferences:** Technical debt and long-term vendor contracts may sway teams to use different Kubernetes distributions.

Your organization might already be running more than one distro and not even realize it.

The most basic, pure distribution is from kubernetes.io. This distro assumes you are going to build Kubernetes from scratch (the hard way) and only provides the basic building blocks. There are vendor specific distros which are essentially commercially supported versions of upstream Kubernetes with an opinionated set of open source add-ons optimized for certain environments. A managed Kubernetes service from cloud providers such as Amazon, Google and Microsoft provides a cloud-based distribution that offers some conveniences for cluster provisioning and bespoke integrations with the cloud provider's services. If your organization is using a managed Kubernetes service and an upstream Kubernetes, or both, your effort to create, operate, monitor and upgrade multiple clusters is effectively doubled because:

- You have to invest in two parallel platforms or end up making massive investments in internal platforms to consolidate the two.
- With two platforms, ensuring that all environments comply with security and governance requirements is non-trivial.
- Ensuring the same continuous integration (CI) system is able to deploy applications across all environments is also a significant investment of DevOps resources.
- Although DevOps teams may have made investments in automation, providing the operations and security teams with multi-cluster visibility and governance is an unmet requirement.



Multi-cloud deployments are already here, but it is up to Kubernetes users to build their own tools to manage this environment, or painstakingly manage each cloud environment separately.

MULTI CLOUD

Multi-cloud deployments represent a number of technical, operational and business reasons that make a multi-cloud environment advantageous:

- **Diversification and Risk Reduction:** In order to avoid vendor lock-in and reliance on a single vendor, and to minimize other risk factors, a multi-cloud strategy is desirable.
- **Security and Geo-Fencing:** Privacy, compliance and other legal issues may dictate specific cloud providers or the use of multiple providers.
- **Services Specialization:** As cloud vendors seek to differentiate their services, selectively choosing vendors based on unique service offerings, regional pricing or service footprint may become an important factor. In addition, to accommodate legacy infrastructure and applications needs that are limited to only certain providers, a multi-cloud strategy may be the only alternative.

Cloud providers make it easy (and entice users) to use their managed Kubernetes services. But they don't provide multi-cloud solutions. After all, if a company is running an application in GCP, it makes sense for them to use GCP's managed Kubernetes service - GKE. Similarly, if a company is running an application in Amazon, it would behoove them to think about Amazon EKS first.

These managed Kubernetes offerings blunt a number of the pain points exposed by Kubernetes, particularly around control plane management. However, if you are running Kubernetes in multi-clouds, how do you solve common problems such as visibility, monitoring, security and governance across all these environments? Today, you solve problems distro by distro, environment by environment. It's a huge and complex task, requiring strong Kubernetes and cloud expertise.



MULTI PERSONA

Clusters can be created with configurable personas to ease the replication and management of clusters based on their unique configuration and provisioning attributes.

Kubernetes clusters are often unique, each with their own set of capabilities and traits. What happens when you want to replicate a cluster's configuration, provisioning and other key attributes in other locations? What if you want to create and manage clusters based on their functionality and 'personality'?"

The ability to manage clusters based on personas is key. You will likely want to configure, manage and operate GPU clusters differently than clusters optimized to host databases.

You might have some clusters hardened to support PCI or HIPAA environments. You might have some cluster personalities that support only development clusters with low form factor hardware. Being able to confidently and accurately replicate cluster personas is critical to operating a fleet of clusters. By programmatically creating and maintaining clusters based on personas, DevOps teams avoid all of the inherent problems associated with manual configuration, provisioning changes and the ensuing drift.





Kubernetes-resident applications require their own management, which is called Application Operations or Application Lifecycle Management (ALM).

APPLICATION OPERATIONS

Once Kubernetes clusters are up and running, there is the on-going need to deploy, operate, monitor, and upgrade containerized applications across said clusters. Though Kubernetes empowers developers to utilize modern architectures to enable application operations, it can also be really complicated.

Kubernetes is not an out-of-the-box, “complete” solution. In fact, when someone says they are using Kubernetes clusters, what they really mean is they are using Kubernetes and a number of other open source or off-the-shelf packages that go with it. Kubernetes is elegantly architected to interact with additional open source or commercially available software packages and tools to round out its functions and tasks. As your business grows, the ALM-specific feature requests from SRE, engineering, compliance and security teams will continue to increase the complexity of your application environment. These may include:

- **Application Templating:** Configuration of app elements, definitions, and policies such that they can be successfully operated on a reusable and repeatable basis.
- **Private Registry Integration:** Clusters need to pull container images from a combination of public and private container registries. Access to private container registries requires the provisioning of authentication credentials.
- **Secrets Distribution and Updates:** Kubernetes objects store sensitive data such as access credentials, private keys, etc using encryption. There are multiple ways of distributing secrets to applications deployed in Kubernetes.
- **App Health Checks:** Determine the health of individual apps on a container-by-container basis.
- **Horizontal Pod Autoscaling:** Ensures consistent application service levels during higher or lower usage periods by automatically spinning up or down additional application container replicas.
- **Global Load Balancing:** Direct user traffic to clusters deployed across multiple clusters.
- **App Ingress Configuration:** Defines how to route inbound requests to backend resources in a Kubernetes cluster.
- **Container Upgrades:** There are several options for performing software application upgrades including attempting to perform a successful upgrade on a single “canary” cluster before upgrading the remaining clusters.
- **Troubleshooting and Diagnostics:** Centralize tools and alerts in one secure location for triage and analysis.

If you are building and supporting an in-house Kubernetes platform, you are employing highly skilled resources to develop and support code and scripts to automate all aspects of cluster management and application operations. This is a certainty. It is also an expensive commitment to have highly paid resources maintain an ever-evolving platform that is not core to your business.



Ensure visibility across clusters, guardrails for teams and secure, harden all clusters.

VISIBILITY, GOVERNANCE & SECURITY

Like any other computing entity, Kubernetes clusters need to be monitored to ensure performance and availability and they must be protected from attacks. Kubernetes multi cluster management best practices dictate that organizations need to incorporate:

- **Visibility and Governance:** Organizations that manage multiple Kubernetes clusters need a centralized multi-cluster operations console to provide deep visibility and state monitoring with a near real-time view of managed clusters. Teams should be able to get down to the “pod” and “namespace” level on a selected Kubernetes cluster, or entire cluster fleet, to check on its health status. The ability to continuously monitor the health of both the cluster and the deployed applications is paramount.
- **Hardening Clusters:** Kubernetes clusters should be reinforced to ensure that critical cluster-wide configuration settings are used and enabled using strong encryption and secure access methodologies. Organizations should ensure that clusters are compliant with internal policies to guarantee that clusters always have mission critical security and software add-ons that may be necessary for activity monitoring.
- **Zero Trust and Secure Access:** Organizations managing a global cluster fleet cannot risk having their cluster control plane open to attackers on the Internet. The networking complexities related to managing distributed clusters (cloud, datacenter, edge) from a central management system that has direct access to the kubeapi server introduces network security complexities that grow with cluster sprawl. Implementing a “Zero Trust Model” to access the cluster’s control plane can go far to both simplify management and increase security levels.
- **Encrypted Data and Secrets and Management:** Generally speaking, data both in-flight and at-rest must be encrypted. Organizations can prevent orphaned secrets on clusters by automating and securing the delivery, provisioning and de-provisioning of secrets across multiple clusters. Additionally, secrets should be automatically injected just in time (JIT) as the application workload is deployed to the target clusters.



CONCLUSION

Increasingly, companies are running Kubernetes clusters and applications around the globe using multiple distributions and multiple cloud environments. Specifically, you may end up running Kubernetes clusters on-premises and in different cloud provider regions, putting the burden of normalizing configuration and deployment across cloud provider environments on your team. You may also end up developing an operations platform to run your application across multiple clusters. This leads to investing DevOps resources in building tooling and infrastructure automation in-house; a long, complicated and costly proposition.

THE RAFAY KUBERNETES MANAGEMENT CLOUD

The Rafay Kubernetes Management Cloud is a turnkey offering that automates Kubernetes cluster management, application deployment and operations at scale. Rafay KMC enables development, operations and site reliability engineering teams to govern, manage and monitor multiple Kubernetes clusters and distros, and operating applications in Kubernetes environments running on-premise, in the cloud or at the Edge. Rafay accelerates team productivity resulting in faster time-to-market for new business capabilities.

Rafay KMC eliminates the need to build an in-house platform for Kubernetes cluster management and application deployment in cloud, on premises and Edge environments. Organizations can now achieve their desired levels of reliability, availability and performance with any combination of compute environments.

Learn More About Rafay Systems

