ESSENTIAL KUBERNETES

10 Multi-Tenancy Best Practices for Namespaces as a Service (NaaS)

WHITE PAPER

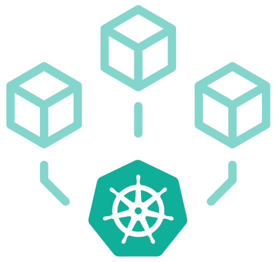
CONTENTS

Introduction: Multi-tenancy and Namespaces

Kubernetes multi-tenancy is a concept that defines how multiple tenants, such as teams or individuals within an organization, share the same Kubernetes infrastructure. Typically, this is done by slicing Kubernetes clusters into smaller environments called namespaces.

The concept is simple. But in practice, managing multi-tenancy across a large number of users or departments, sometimes across separate infrastructures, can be extremely difficult and risky.

And without the right guardrails in place, providing namespaces as a service to teams of developers can lead to performance and security challenges.



How Many Clusters is Right for My Organization?

It depends. The simplest approach for your organization might be to create a Kubernetes cluster for each application. However, this quickly becomes overly costly due to several factors that impact efficiency, resource utilization, and management complexity. Kubernetes is designed to manage multiple containerized applications efficiently within a single cluster. Deploying separate clusters for each application leads to resource wastage, increased operational overhead, and reduced scalability.

Why? For starters, because each cluster consumes resources such as memory, CPU, and network bandwidth. By provisioning a new cluster for every application, resources are underutilized as each cluster typically has its own control plane, nodes, and networking infrastructure. This results in increased infrastructure costs and inefficient resource allocation.

Moreover, managing multiple clusters introduces complexity in terms of configuration, monitoring, and security. Each cluster requires its own setup, upgrades, and maintenance, leading to duplicated efforts and potential inconsistencies. Scaling becomes challenging, as it's harder to distribute resources optimally across isolated clusters. Additionally, security concerns arise when each cluster requires separate access controls, certificates, and network policies. Coordinating updates, patches, and security measures across numerous clusters is time-consuming and error-prone.

In contrast, using a single cluster with appropriate namespaces for different applications allows efficient resource sharing, streamlined management, and better scalability. This concept, called multi-tenancy, simplifies deployment, monitoring, and updates, reducing operational overhead and enhancing resource utilization.

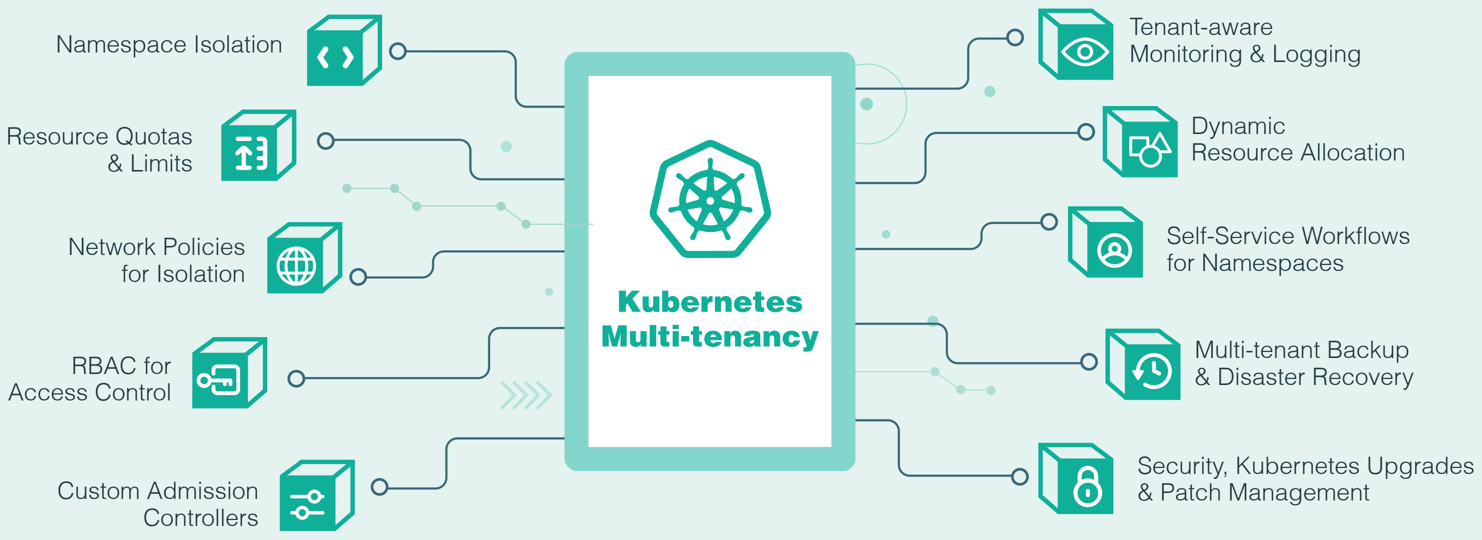
But managing namespaces isn't exactly straightforward, particularly regarding resource allocation, access control, and security. If mastered, however, the enterprise can gain immensely via the sharing of cloud resources and a lowered total cost of cloud computing. On average, Rafay customers see a 63% savings in cloud costs by leveraging multi-tenancy and namespaces.

In this white paper, we will explore essential Kubernetes multi-tenancy best practices that are required to help organizations harness the power of Kubernetes and deliver namespaces as a service.

CONFIGURATION
MONITORING
SECURITY

Managing Multiple Clusters

Considerations for Namespaces as a Service


Namespace Isolation
Resource Quotas & Limits
Network Policies for Isolation
RBAC for Access Control
Custom Admission Controllers
Kubernetes Multi-tenancy
Tenant-aware Monitoring & Logging
Dynamic Resource Allocation
Self-Service Workflows for Namespaces
Multi-tenant Backup & Disaster Recovery
Security, Kubernetes Upgrades & Patch Management

Namespace Isolation

Namespaces are a fundamental feature in Kubernetes that provide a way to partition a single cluster into multiple virtual clusters. Leveraging namespaces for multi-tenancy allows you to logically isolate different tenants within the same cluster. It's crucial to establish a clear naming convention for namespaces that reflects the tenant it belongs to.

However, namespaces alone are not enough to achieve complete isolation. Network policies and role-based access control (RBAC) must be properly configured with namespaces (see below) to control communication and access between namespaces. This way, tenants can only interact with resources in their designated namespace, ensuring separation and security.

Resource Quotas and Limits

Resource quotas and limits play a significant role in multi-tenant Kubernetes environments. Quotas help allocate a specific amount of resources (CPU, memory, storage) to each tenant, preventing one tenant from consuming excessive resources and impacting others. Limits, on the other hand, ensure that a tenant's workload doesn't exceed its allocated resources, thereby maintaining predictable performance and availability.

Properly defining and enforcing resource quotas and limits prevents resource contention, guarantees fair resource distribution, and safeguards against resource-hogging applications.

Network Policies for Isolation

Kubernetes Network Policies are a powerful tool to control network communication between different tenants' workloads. By defining and applying network policies, you can specify which pods can communicate with each other based on labels, namespaces, or IP ranges. This fine-grained control ensures that tenants' workloads communicate only with the necessary and approved components, enhancing security and isolation.

RBAC for Access Control

Role-Based Access Control (RBAC) is crucial in a multi-tenant Kubernetes environment to regulate who can access and manage resources. It enables administrators to define roles and permissions for different users or groups, granting them appropriate access levels to specific namespaces or resources.

Implementing RBAC ensures that tenants have the necessary permissions to manage their own resources while preventing unauthorized access or accidental modifications that could impact other tenants.

Custom Admission Controllers

Admission controllers are plugins that intercept and modify requests to the Kubernetes API server before they are persisted in the cluster. Custom admission controllers can be developed to enforce organization-specific policies or checks during resource creation and updates.

For multi-tenancy, custom admission controllers can help enforce naming conventions, resource labels, and other policies specific to your organization, enhancing consistency and adherence to best practices across tenants

Tenant-aware Monitoring and Logging

Monitoring and logging are essential components of managing a Kubernetes cluster, but in a multi-tenant environment, they become even more critical. Each tenant's workloads should be monitored and logged separately to facilitate troubleshooting, performance optimization, and resource usage analysis.

Implementing a tenant-aware monitoring and logging solution ensures that each tenant can monitor their own resources without access to other tenants' data, maintaining privacy and security.

Dynamic Resource Allocation

In a multi-tenant Kubernetes setup, workloads from different tenants may experience varying levels of load. Implementing dynamic resource allocation mechanisms, such as Horizontal Pod Autoscaling and Cluster Autoscaler, helps in automatically adjusting resource allocation based on actual usage.

Dynamic resource allocation ensures that resources are efficiently distributed among tenants, providing optimal performance and cost-effectiveness.

Self-Service Workflows for Namespaces

Self-service namespaces are Kubernetes namespaces that can be created by developers, data scientists, researchers, etc. on-demand without the need of an administrator of the cluster.

By providing a self-service workflow for namespaces, development teams gain autonomy and the ability to manage their own environments. This speeds deployment as they can independently create and modify namespaces without relying on central IT or cluster administrators. This autonomy can lead to faster development cycles and better collaboration among different development teams sharing the same Kubernetes cluster.

Providing namespaces in a self-service fashion compared to submitting ticket requests to administrators for the creation of clusters eliminates the back and forth of tickets in order to provide access to infrastructure.

Multi-tenant Backup and Disaster Recovery

Kubernetes backup and disaster recovery strategies should be tailored to a multi-tenant Kubernetes environment. Each tenant's data, configuration, and state should be isolated and restorable independently. Regularly test backup and restore procedures to ensure that tenants can recover their data and applications in case of failures.

Security, Kubernetes Upgrades and Patch Management

Security is paramount in any Kubernetes environment, and multi-tenancy adds an extra layer of complexity. Stay updated with security patches and updates for Kubernetes and its components. Perform regular security audits, vulnerability assessments, and penetration testing.

Note that any security patches or upgrades applied to a cluster affect every namespace running on that cluster. So, each application running in its own namespace will need to be patched. Typically, this is done by each application team as part of the application and Kubernetes lifecycle.

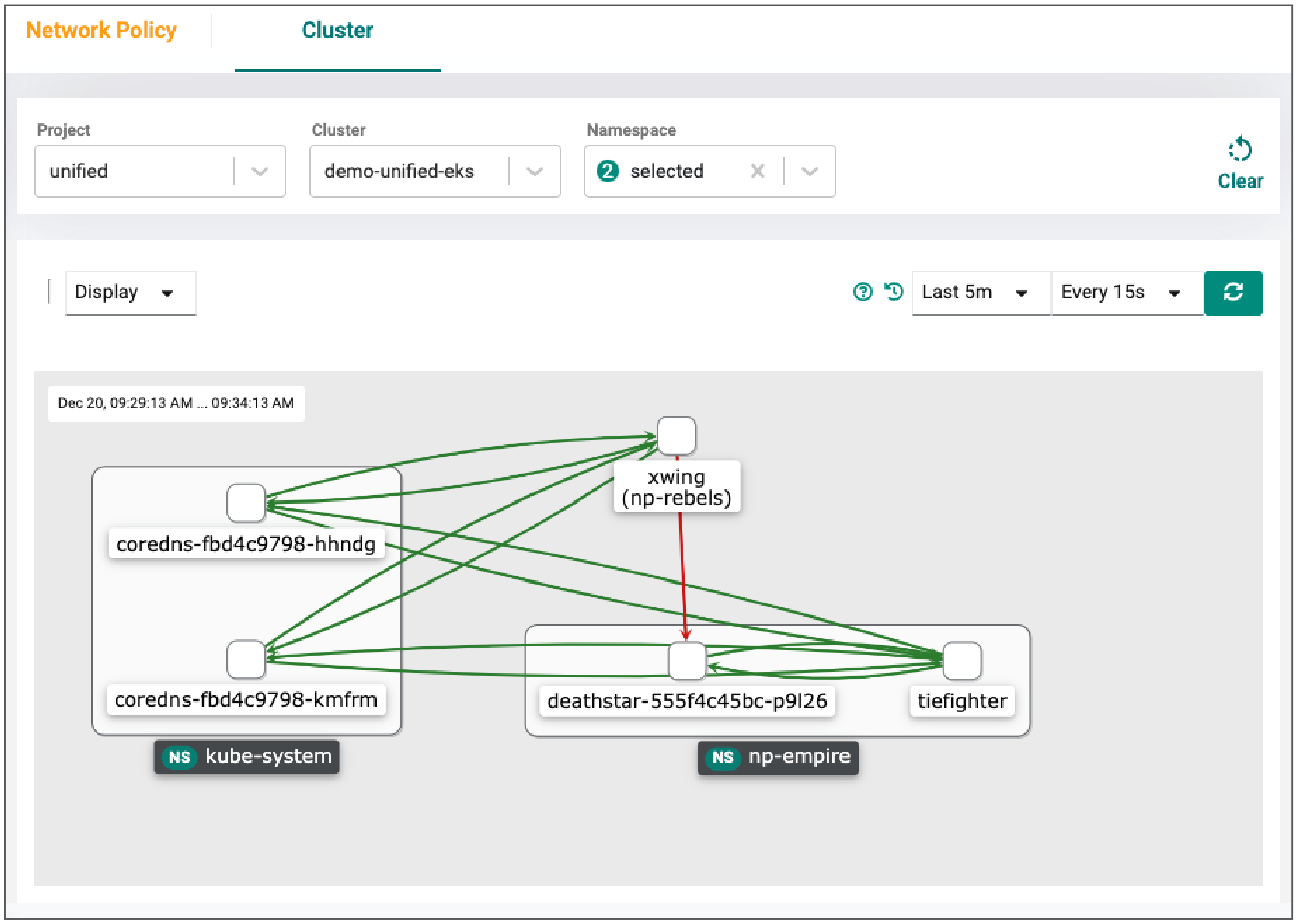
It is also recommended to utilize container scanning tools to ensure that tenants' container images are free from vulnerabilities before deployment. Implement policies that require tenants to use only trusted container registries.

How Rafay Powers Namespaces as a Service

The Rafay Cloud Automation Platform includes out-of-box tooling that platform teams can leverage at scale to address challenges around shared clusters. Examples include:

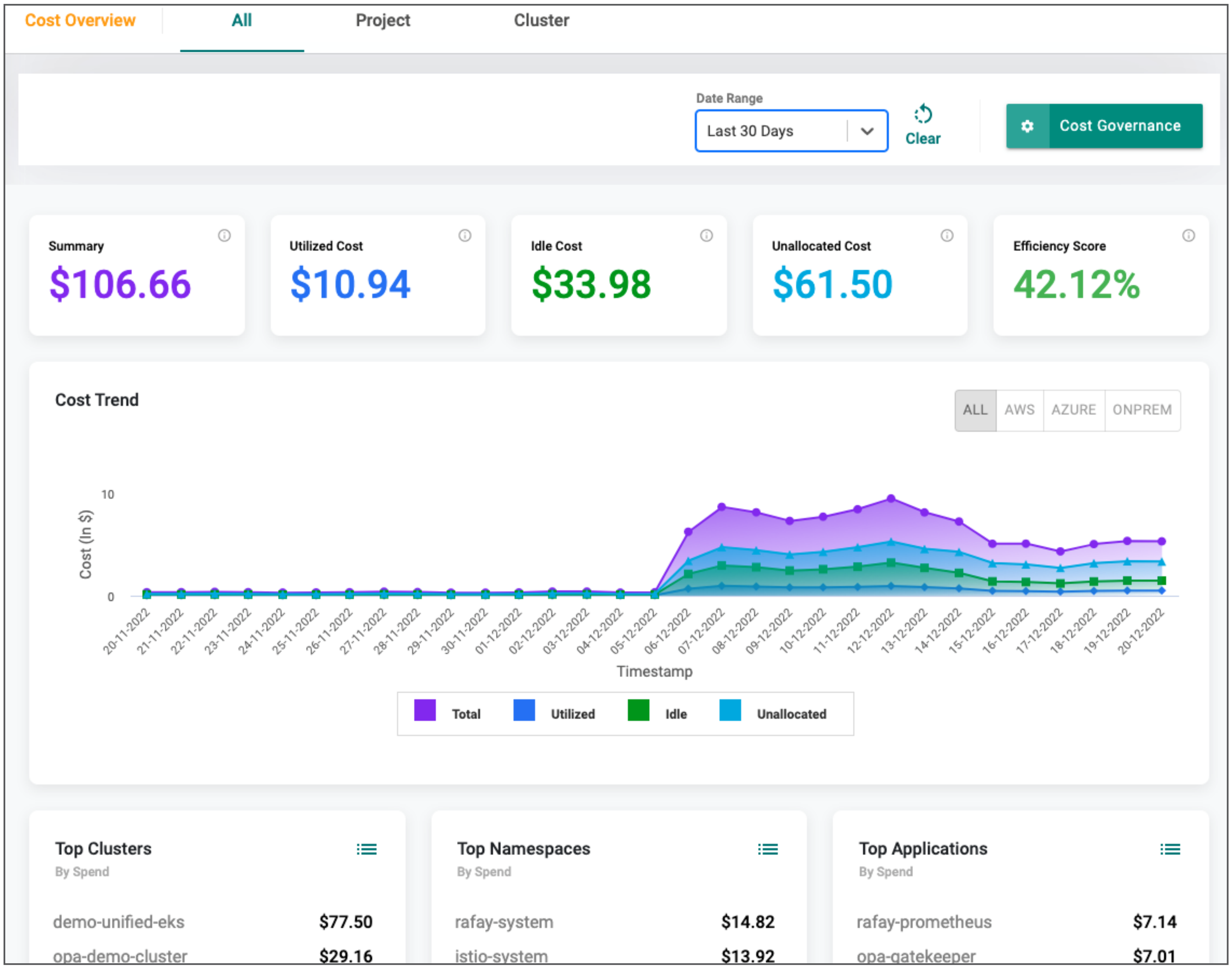
Network Policy Manager

By default, namespaces are not isolated in Kubernetes environments. The Rafay Cloud Automation Platform provides an easy way to configure/enforce network policies for namespace isolation.



Cost Management

Granular visibility into resource utilization & cost metrics by namespace, labels, workloads are available that help platform teams enable showback/chargeback models or drive cost optimization exercises.



Summary	Utilized Cost	Idle Cost	Unallocated Cost	Efficiency Score
$106.66	$10.94	$33.98	$61.50	42.12%

Top Clusters
By Spend

demo-unified-eks	$77.50
opa-demo-cluster	$29.16

Top Namespaces
By Spend

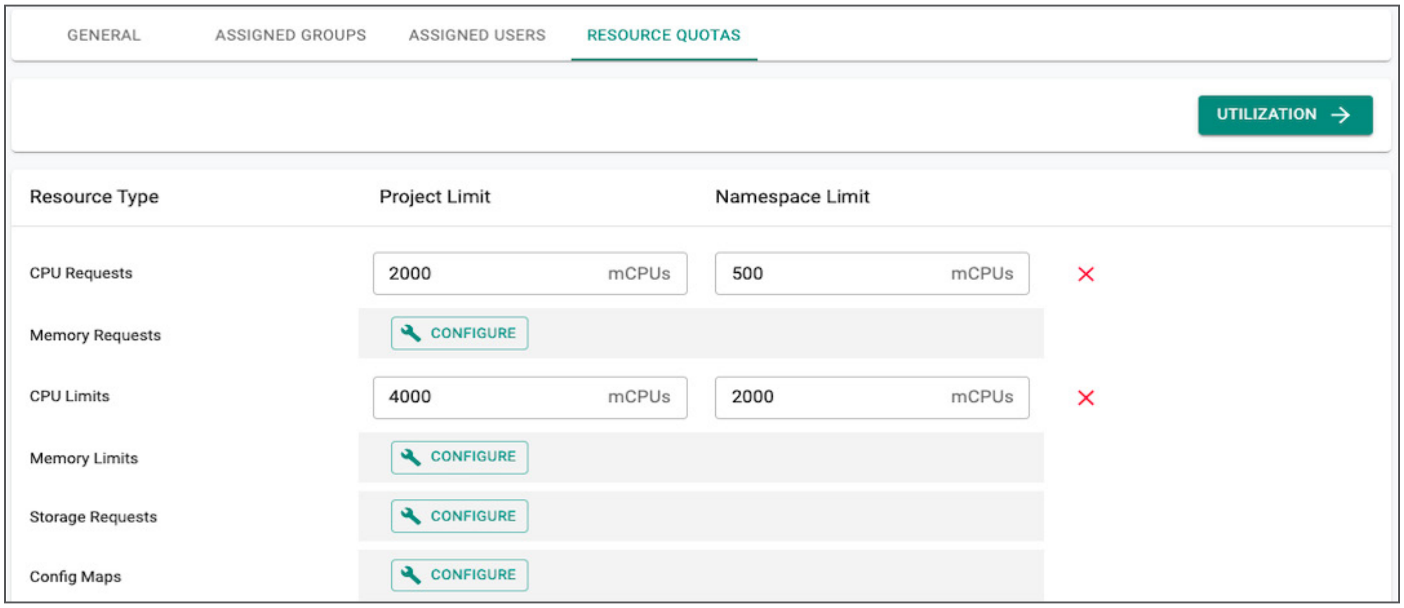
rafay-system	$14.82
istio-system	$13.92

Top Applications
By Spend

rafay-prometheus	$7.14
opa-gatekeeper	$7.01

Workspaces

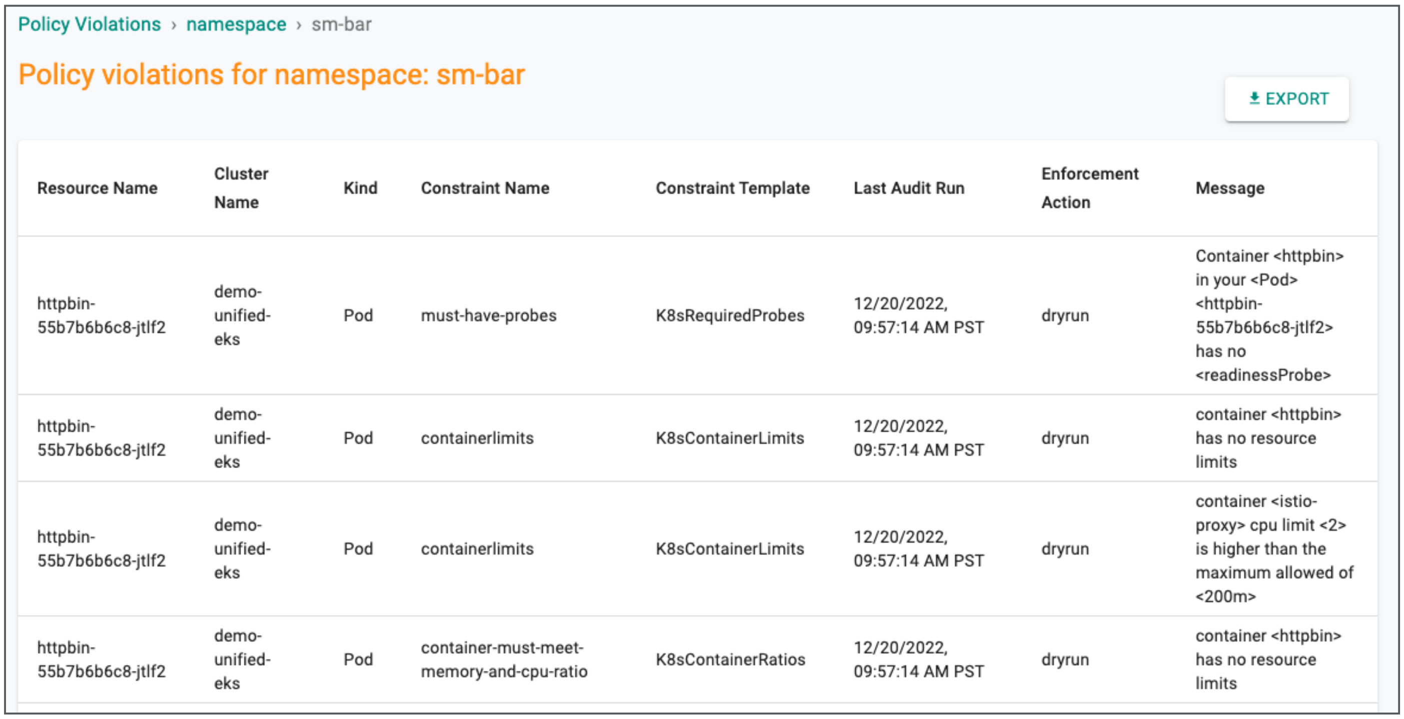
Platform teams can create projects/workspaces and assign resource quotas to application teams to enable a self-service model. This allows application teams to create/manage namespaces (without cluster-wide privileges) within assigned quotas and create pipelines to deploy applications.



Resource Type	Project Limit	Namespace Limit	
CPU Requests	2000 mCPUs	500 mCPUs	✕
Memory Requests	CONFIGURE		
CPU Limits	4000 mCPUs	2000 mCPUs	✕
Memory Limits	CONFIGURE		
Storage Requests	CONFIGURE		
Config Maps	CONFIGURE		

Policy Management

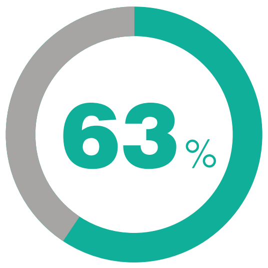
The platform provides tooling to configure/enforce namespace specific OPA Gatekeeper policies and provides centralized visibility around policy violations to application teams for the resources that they own.



Policy violations for namespace: sm-bar

Resource Name	Cluster Name	Kind	Constraint Name	Constraint Template	Last Audit Run	Enforcement Action	Message
httpbin-55b7b6b6c8-jtlf2	demo-unified-eks	Pod	must-have-probes	K8sRequiredProbes	12/20/2022, 09:57:14 AM PST	dryrun	Container <httpbin> in your <Pod> <httpbin-55b7b6b6c8-jtlf2> has no <readinessProbe>
httpbin-55b7b6b6c8-jtlf2	demo-unified-eks	Pod	containerlimits	K8sContainerLimits	12/20/2022, 09:57:14 AM PST	dryrun	container <httpbin> has no resource limits
httpbin-55b7b6b6c8-jtlf2	demo-unified-eks	Pod	containerlimits	K8sContainerLimits	12/20/2022, 09:57:14 AM PST	dryrun	container <istio-proxy> cpu limit <2> is higher than the maximum allowed of <200m>
httpbin-55b7b6b6c8-jtlf2	demo-unified-eks	Pod	container-must-meet-memory-and-cpu-ratio	K8sContainerRatios	12/20/2022, 09:57:14 AM PST	dryrun	container <httpbin> has no resource limits

Namespaces Are a Game-Changer, If Done Correctly



Lower cost of cloud computing for Rafay customers

Implementing namespaces as a service in Kubernetes can be a game-changer for organizations seeking efficient resource utilization and streamlined Kubernetes management. By following these essential best practices, you can successfully navigate the complexities of multi-tenancy while maintaining security, isolation, and scalability. The added benefit is a lower total cost of cloud computing - 63% on average for Rafay customers.

Learn More About Rafay Systems

Ready to find out why so many enterprises and platform teams have partnered with Rafay for Kuberenetes multi-tenancy? **Sign up for a free trial today**.

RAFAY

Rafay Systems, Inc. | 530 Lakeside Dr. Ste 220 Sunnyvale, CA 94085 | 669.336.4800 | rafay.co