NexaVM Technologies AG.

nKU Container Service

Technical White Paper

CONTENTS

1 Background

1.1 Container Service Development History

Evolution in Cloud for enterprise IT architecture and micro servicing of application architecture are two inevitable trends. Cloud computing has matured after years of development and has gradually penetrated the realm of traditional industries from the Internet. As of 2025, more than half of the enterprises have completed cloud transformation and migrated to private or public clouds. In comparison, private clouds are more suitable for medium and large companies, as they offer the advantages of reusing existing infrastructure, customizing business-specific solutions, controlling data security, and meeting compliance requirements.

Containers are the core of the new generation of cloud computing technology, due to its lightweight and flexible characteristics, inherently more suitable than VM instance for microservice architecture, is the first choice of cloud native applications. docker is currently the hottest open-source container technology. docker puts forward the concept of build, ship, run, standardize the delivery of building block standard, the application from the development, build, integration, deployment to the operation of the whole lifecycle perfect management. The emergence of Docker has led to the rapid evolution of the container ecosystem and the emergence and flourishing of container services around container technology.

Container cloud platform not only realizes the standardized delivery of infrastructure but also realizes a new form of CaaS with the application as the core, through the integration of development, operation and maintenance, application orchestration scheduling, service elasticity and scalability and other features.

1.2 Why Container Service?

Container technology for us to open a window, but really runs in the online environment, the need to solve the container environment related to the network, storage, cluster, cluster, cluster HA and other aspects of the problem, from the container to the container cloud of the evolution of the application are born.

With the development of virtualization technology, many industries have completed the process of physical machine virtualization. This virtualization technology reduces the operation and maintenance complexity to some extent and improves the utilization of resources. This virtualization technology and OpenStack etc. can be regarded as the development of IaaS in cloud computing.

With the continuous development and improvement of container technology, the value of container service has been gradually discovered, which can essentially better solve the problems faced by enterprises in the new form:

- Deployment of Standardized Applications
  Adopt container image configuration to achieve the standardization of the operating environment, shielding the application deployment process for different environments need environment configuration, installation steps and other complex processes. The original deployment, configuration of the operation and maintenance work in advance to the development and delivery stage, in the production of the image stage to solve the problems arising from the operation and maintenance on-line.

- Enhanced Resource Utilization
  Container service is a lightweight virtualization technology based on the operating system. Compared with traditional virtualization technology, multiple containers can share the kernel process and kernel resources of the operating system, which effectively saves the operating system-level resource overhead, and better utilizes the resources through the

enhancement of container density.

- Effective Integration of Existing Resources
  Containers can run in a variety of cloud service environments, effectively avoiding vendor binding, and can achieve unified management of heterogeneous basic resources that already exist in the enterprise, and this model of unified management of applications shields the environment from variability and reduces the difficulty of system operation and maintenance.

- Facilitate Devops Implementation on the Ground
  Container Build, Ship, Run concept and its technical characteristics, more than enough to better integrate with CI/CD technology, to promote the CI/CD concept of the ground, from the technical means to ensure that the project management approach and management concepts of the real effective landing.

- Accelerate Enterprise Software Asset Accumulation
  Mirror image repository can realize the function similar to application store through centralized management of application images, which is conducive to better precipitation and accumulation of enterprise software assets, to provide a variety of operating environments more quickly and efficiently.

1.3 Enterprise-Grade Private Container Service for Production Environments

With years of experience in system construction and operation and maintenance, NexaVM provides enterprise users with a one-stop container service - NexaVM NKU Enterprise Container Cloud Platform. While enjoying the high performance, high density, second-level elasticity and scalability, and cross-platform deployment brought by container technology, NexaVM NKU brings you intelligent monitoring and operation and maintenance, one-click application deployment, multi-tenant isolation, and 100% compatible with the community's standard open APIs,

which can help you unload your burden and easily go to the cloud.

1.4 IT Architecture Cloud Transformation

Starting from the bulky traditional architecture, the resources will be exclusive and may be heterogeneous physical servers, the use of cloud computing virtualization technology for integration, the emergence of the concept of the platform, with the elastic on-demand allocation of resources, from the infrastructure to standardized forms of delivery, and then according to the actual situation and application scenarios transition to the private cloud or public cloud, the private cloud is more suitable for large-scale companies, reuse of the existing infrastructure, customized targeted solutions, information security, meet compliance requirements, public cloud is more suitable for startups do not need to maintain their own infrastructure, the cluster size with the business development and expansion. Private cloud is more suitable for large companies, can reuse existing infrastructure, can be customized to meet the targeted solutions, information security, to meet compliance requirements, public cloud is more suitable for startups, do not need to maintain their own infrastructure, the cluster size with the development of the business to synchronize the expansion. Private and public clouds eventually converge to form a hybrid cloud, combining the advantages of both.

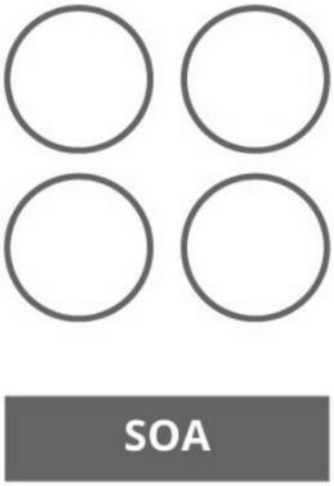
1.5 Application Architecture Micro Servicing

- Monomer Application

  All the functionality is packaged in one application. Particularly unwieldy for large-scale complex applications. Compilation time is too long; regression testing cycles are too long; development efficiency is reduced; it is not conducive to updating the technical framework; it is not scalable.

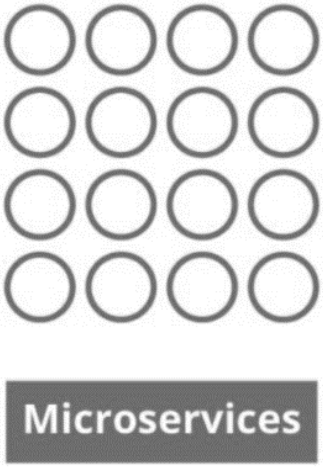
- Traditional SOA

  Split tightly coupled systems into business-oriented, coarse-grained, loosely coupled, stateless services. Expose services through standardized interfaces between applications. Traditional SOA implementations focus on ESBs, SOAP, and WSDL.



- microservice

  Service-as-a-product. De-ESB, decentralized, distributed. Fine granularity of services. Emphasize features of distributed systems such as elastic scaling, load balancing, service discovery, failover, high availability, etc. Independent deployment through automation.



1.6 What is a Container

Docker is a mainstream open-source container engine that lets developers package their applications as well as dependency packages into a portable container and then install it on any popular machine.

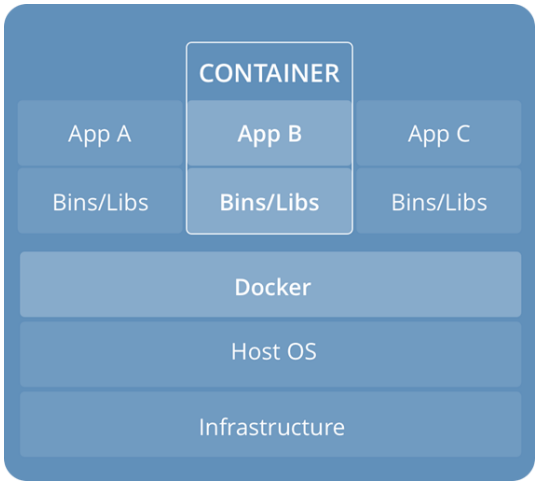
The idea of Docker is to realize "Build, Ship and Run Any App, Anywhere", i.e., through the packaging, distribution, deployment, and operation of the application life cycle management, to achieve the application components "once packaged, run everywhere! The purpose is to realize "Build, Ship and Run Any App, Anywhere".

1.7 Advantages of Containers

- (an official) Standard
  The image of a Docker container provides the complete runtime environment of the application and ensures the consistency of the environment in which the application runs.
- Lightweight
  It is a lightweight virtualization technology that does not require additional overhead projectors such as hardware virtualization and running a full operating system.
- Speedy
  Equivalent to special Linux processes, and therefore can achieve startup times of seconds, or even milliseconds.
- Efficiently
  A host with the same configuration can run a greater number of applications, improving resource utilization.

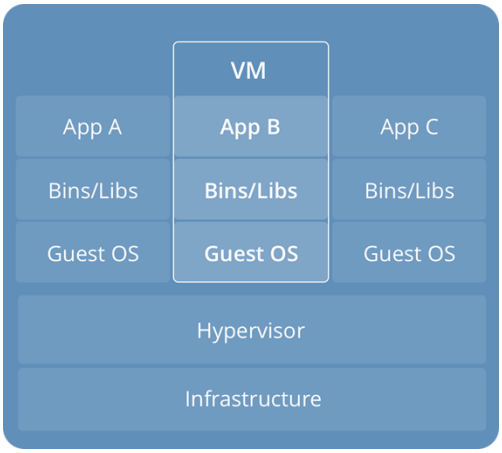
1.8 Containers vs. VM Instances

- Container



A container (CONTAINER) is an application-level abstraction that packages code and its dependent files together. Multiple containers can run on the same machine and share the operating system kernel with other containers, each running as an isolated process in user space. Containers take up less space than VM instances (container images are typically tens of MB in size) and can be started almost immediately.

- VM Instance



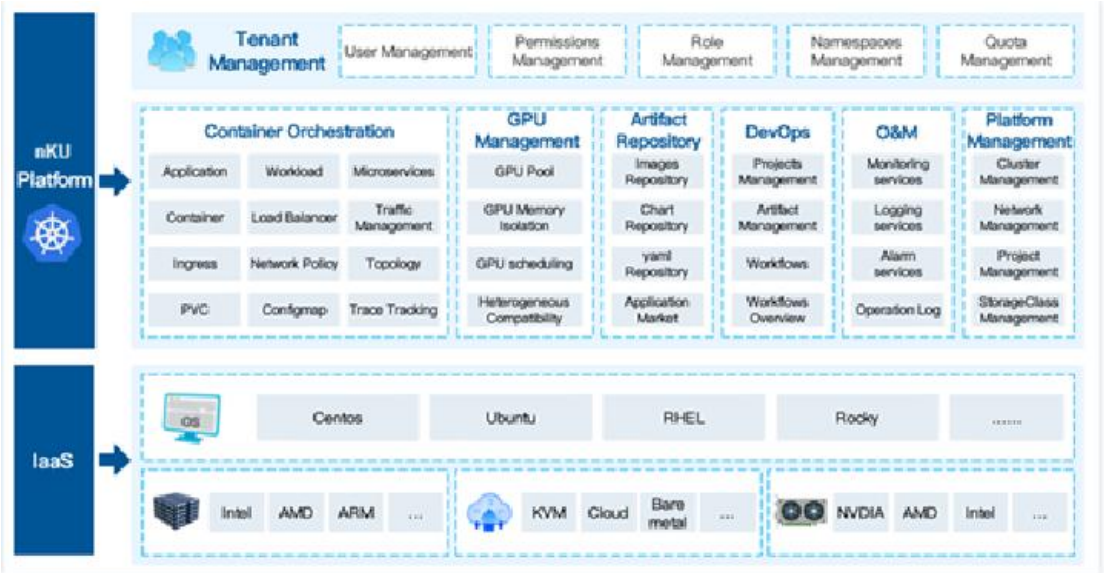
A VM instance (VM) is an abstraction of the physical hardware resources that turn a single server into multiple servers. A virtualization hypervisor allows multiple VM instances to run on a single machine. Each VM instance contains a full copy of the operating system, one or more applications, required binaries and libraries - taking up tens of gigabytes. VM instances are usually slow to start.

- Performance Comparison

2 Overall Product Architecture

2.1 Introduction to NexaVM nKU Container Service

NexaVM nKU Enterprise Container Cloud Platform is an out-of-the-box enterprise container service based on the Kubernetes core base. The product is designed with the core concept of "making containers simpler to use" and deeply integrates dozens of cloud-native open-source components in a pluggable way to provide easy-to-use, step-by-step, consistent operating experience. It provides easy-to-use, step-by-step, consistent operation experience of multi-cloud and multi-tenant management, multi-tenant management, application lifecycle management, unified management of containers and VM instances, shared GPU scheduling, CI/CD, microservices governance, etc., which maximizes the shielding of the underlying technical details, and greatly reduces the threshold of container technology use. It focuses on solving the challenges of infrastructure cloudization, application deployment simplification, and operation and maintenance management automation in enterprise production environments.



Core Objective: Help enterprises realize cloud computing transformation, accelerate application delivery, and reduce O&M costs

Core concepts:

- Rapid - Enables rapid application build, deployment, and operation and maintenance
- Simplicity - Provides a clean, unified view of the management interface that is easy to use yet flexible
- Reliable - meets the reliability requirements of platforms and applications in enterprise production environments

2.2 NexaVM nKU Container Service Functional Modules

nKU provides a one-stop container service solution. 30 minutes to set up a HA platform and start managing host clusters and container applications. Five functional modules cover all types of container service in the enterprise production environment landing all kinds of needs.

- Platform Management

  Consolidate existing infrastructure, whether local physical hosts or VM instances, or VMs on the public cloud, to consolidate compute, storage, and network resources in the form of host pools and allocate them on demand. Supports hybrid multi-cluster deployment, each host cluster corresponds to a Kubernetes cluster, and the Kubernetes host cluster network supports Calico network types.

- Container Service

  Provide the full life cycle care of container applications, and the operation support of applications includes load balancing, elastic scaling, fault self-healing, gray scale upgrading, configuration management and other functions. Support docking local, block storage, file storage multiple types of storage volumes to meet the persistence needs of stateful applications.

- Delivery Center

  The Delivery Center provides the ability to manage three types of repositories: YAML template repositories, image repositories, and application markets. The Delivery Center provides the ability to manage three types of repositories: template repositories, mirror repositories, and application marketplaces, making it easy

to install applications with just a few clicks. At the same time, it also meets the needs of beginners and advanced users to develop and manage applications through standardized delivery methods.
The standardized delivery process is conducive to the construction of industry standards.

Built-in enterprise-class image repository, and supports segregation by project as well as user authentication, and visual image management.

- Operations and Maintenance Center

Full graphical interface real-time monitoring/logging alarm function, unified collection of container logs, visual analysis of container logs, a variety of management, operation and maintenance means. management, operation and maintenance means. Platform inspection, auto inspection, automatic backup, smooth upgrading and other self-operation and maintenance means to comprehensively guarantee the platform's High Availability and Flexibility.

- DevOps

Simple configuration can automatically generate highly concurrent workflow, multiple microservices can be built, deployed and tested in parallel. The workflow can be automatically generated with high concurrency and multiple microservices can be built, deployed, and tested in parallel. Customized workflow steps, with manual review, flexible and controllable guarantee the quality of business delivery.

Comprehensive view of the system's state, including data on clusters, projects, environments, workflows, critical process pass rates, etc. Overview. Provide objective performance metrics of project dimensions such as build, test, deployment, etc., to accurately analyze the shortcomings of R&D performance and promote steady improvement.

2.3 NexaVM NKU Feature List

Form	Characterization	Descriptive
Fig. Beginning	Overview of resources	Real-time monitoring and display of clusters, nodes, applications, workloads, jobs, worker containers, and container groups with allowed overviews
		Real-time statistics of multi-tenant, local repository information situation
	Unread alarm statistics for the last seven days	Show unread alarm messages for the last 7 days
	Cluster Resource Utilization Statistics	Real-time monitoring and display of total cluster CPU and memory resource utilization
		Showing Top 3 Cluster Rankings for CPU, Memory Usage
		Real-time monitoring and display of the cluster's CPU and memory resource utilization in the last 24 hours
	Resource utilization statistics within the cluster	Showing Top 10 ranking of CPU, memory and pvc usage for container groups

Container Orchestration	application	Supports basic lifecycle management such as installing, deleting, etc. of applications
		Supports updating the application with the current deployment chart/new deployment chart and viewing the version comparison.
		Supports managing application history, including viewing, rollback, and deletion.
		Supports centralized management of application-related resources, including container groups, data volumes, services, and ingresses
	workload	Supports Deployment, StatefulSet, and DaemonSet workload types.
		Supports visual configuration of health checks, environment variables, lifecycle callbacks for workloads
		Supports mounting persistentVolume templates (StatefulSet types only), pvc, hostpaths, configSets and secret dictionaries for workloads.

		Support for setting GPU configurations for workload container groups, including the number of GPU cards used, GPU memory per GPU, and GPU memory percentage per GPU
		Supports selecting the GPU card to be used by GPU specification, or directly specifying the GPU card
		Supports multiple containers sharing a single GPU card with isolated GPU memory and multiple GPU cards for a single container.
		Support for setting node scheduling policies, affinity, anti-affinity, and tolerance for workload container groups
		Supports basic lifecycle management such as creation, update, restart, and deletion of workloads
		Supports managing workload history, including viewing Yaml, rollbacks
		Supports modification of workload configurations via both Yaml and forms

		Support for modifying workload container images
		Supports elastic workload scaling, including manual scaling and autoscaling types.
		Supports centralized management of workload-related resources, including: container groups, related services, etc.
		Support for centralized view of workload events
	job	Supports two types of jobs: Job, CronJob.
		Support for creating jobs from mirrors or from templates
		Support for visually configuring health checks, environment variables for jobs
		Supports mounting persistentVolumeClaims, configmaps, and secret dictionaries for job
		Support for setting GPU configurations for jobs, including the number of GPU cards used, GPU memory per GPU, and GPU memory percentage per GPU
		Supports selecting the GPU card to be used by GPU specification,

		or directly specifying the GPU card
		Supports multiple containers sharing a single GPU card with isolated GPU memory and multiple GPU cards for a single container.
		Supports basic lifecycle management such as creation, update, deletion, enablement and disablement of jobs
		Supports centralized management of job-associated resources, including: container groups, pvc
	pod	Support for viewing and deleting container groups
		Supports centralized management of containers in a pod, including viewing container information, container logs, saving containers as an image, and so on.
		Support for accessing container group endpoints
		Support for centralized viewing of container group events
	service	Supports four service types: ClusterIP, NodePort, Load Balance and Headless.

		Supports customizing the external IP address for the LoadBalancer service.
		Supports basic lifecycle management such as creation, update, and deletion of services
		Supports modification of service configuration via both Yaml and forms
		Supports centralized view of service events
	ingress	Supports basic lifecycle management such as creation, update, and deletion of ingresses
		Supports centralized management of ingress rules, including adding, modifying, and deleting rules.
		Support for viewing routing gateway addresses and forward policy access urls
		Support for centralized view of ingress events
	network policy	Supports the creation of network policies for workloads to control the communication behavior of workloads and

		protect applications from network attacks.
		Supports basic lifecycle management such as creation and deletion of network policies
		Supports modification of network policy configurations via both Yaml and forms
	pvc	Supports distributed storage with pvcs for persistent storage of workload data.
		Supports basic persistentVolumes management such as creation, expansion, and deletion of pvc.
		Support for centralized view of pvc events
	configmap	Supports basic lifecycle management such as creation, update, and deletion of configmaps
	secret	Supports three types of secret dictionaries: Opaque, TLS certificate, and image repository login key.
		Supports basic lifecycle management such as creation, update, and deletion of secret dictionaries

	microservices application	A microservices application is a collection of resource objects such as workloads, services, and service mesh resources
		Supports basic lifecycle management such as creation and deletion of microservices applications
		Supports accessing microservices applications to the platform for visualization governance
		Supports grayscale release strategy for multiple microservices applications, can route traffic to new and old versions according to traffic weight and Request Content to ensure smooth application upgrades.
		Provide traffic monitoring for new and old versions during canary release process
	service topology	Provide visual application topology, intuitively displaying invocations, dependencies and traffic monitoring data between microservices applications
		Supports historical and real-time viewing of topology maps

		Supports configuration of load balancing, connection pooling, Circuit Breaking and other traffic management rules for microservices applications
	trace tracking	Provide trace tracking function, support to query the call chain between services in the current namespace by service name or TraceID.
		Comprehensively monitor the invocation status of services in the invocation chain, invocation time consumption and other key metrics
artifact repository	repository	Support for public and private repository types
		Supports basic lifecycle management such as creation, modification, and deletion of local repository
		Centralized management of image repository, including uploading, exporting, downloading, and deleting image tags.
		Supports 3 types of mirror upload: online upload, file upload, command line upload

		Support centralized management of Chart deployments in the repository, including uploading and downloading Chart, viewing and deleting Chart versions, and installing applications based on Chart.
		Support to set the project to which the local repository belongs to
	YAML template repo	Template repository for managing YAML templates
		Provide a variety of resources for reference, example YAML templates, users can create templates based on the example directly
		Supports basic lifecycle management such as creation, modification, and deletion of templates
	application market	Provide zookeeper, kafka, mysql, redis and other official Chart deployment charts
		Supports one-click publishing of Chart deployment packages from the application market to deploy as application instances

Container O&M	monitoring panel	Supports one-click jump to monitoring panel to view detailed monitoring data of the cluster
		Supports viewing the overall resource usage of the cluster, including CPU utilization, memory utilization, network traffic, etc.
		Support for viewing CPU and memory resource usage of container groups on nodes
		Support for viewing workload resource utilization
		Support for viewing network resource usage for workloads, container groups
	log panel	Provide a unified portal for users to centrally view all container logs in the cluster
		Supports querying logs by cluster, namespace, workload, container group, container, keyword, time range
		Support for exporting logs
		Supports viewing log contexts; supports searching contexts by keywords; supports exporting log contexts

	Alarm messages	Supports centralized viewing and management of all alarm messages on the platform
		Support filtering alarm messages by resource name, time period
		Supports marking alarm messages as read
	alarm	Supports basic lifecycle management such as creating, enabling, disabling, deleting, etc. of alarms
		Supports three Emergency Levels: Emergency, Critical, and Alert
		Supports on-demand turn-on of recurring alarm notifications
		One alarm supports monitoring of multiple resources at the same time
	notification object	The notification object is the way for users to receive alarm messages, and supports four types of notification objects: system, email notification, enterprise WeChat, and nails.
		Supports basic lifecycle management such as creation, modification, and deletion of notification objects.

	current task	Supports centralized viewing and management of in-progress operations
		Support for viewing current job details
	Operation History	Support for centralized view of historical jobs
		Supports filtering historical jobs by time period, job results, and operation source
		Support for viewing historical job details
cluster management	cluster	Supports basic lifecycle management clusters such as creation, retry installation, deletion, etc. of multiple clusters
		Support for Kubernetes Cluster HA
		Supports customized deployment of functional components in clusters
		Support for Kubernetes version 1.24~1.30 of NAMI's Kubernets clusters
		Support CPU, GPU heterogeneous clusters unified construction management and resource unified planning and scheduling

		Supports real-time view of cluster CPU, memory, storage and other resource usage
	node	Supports basic lifecycle control plan such as adding and deleting nodes.
		Supports stop/resume node scheduling to enable scheduling for node maintenance
		Supports real-time view of node's CPU/memory utilization rate, request rate, and limit rate, which facilitates timely understanding of node's resource allocation situation
		Supports centralized viewing of GPU card information on nodes, including basic GPU card information, memory usage, scheduled container pod information, real-time monitoring data, etc.
	GPU pool	Supports viewing all GPU cards under the current cluster by cluster, and the total display usage of all GPUs under the cluster, including: video memory utilization rate, video memory request rate

		Support to view detailed information of GPU manufacturer, model, memory, memory utilization rate, memory request rate, etc.
		Supports visual monitoring of GPU, including: GPU utilization, GPU memory utilization, power consumption, temperature
		Supports centralized view of information about scheduled pods on GPUs
	storage class	Supports basic lifecycle management of storage class creation, update, deletion, etc.
	external network	Supports basic lifecycle management such as creation and deletion of external networks
		Supports adding and removing network segments for external networks
		Supports setting external network sharing mode to allow global or specified projects to use external network resources
		Supports viewing network segment IP usage to improve network planning efficiency

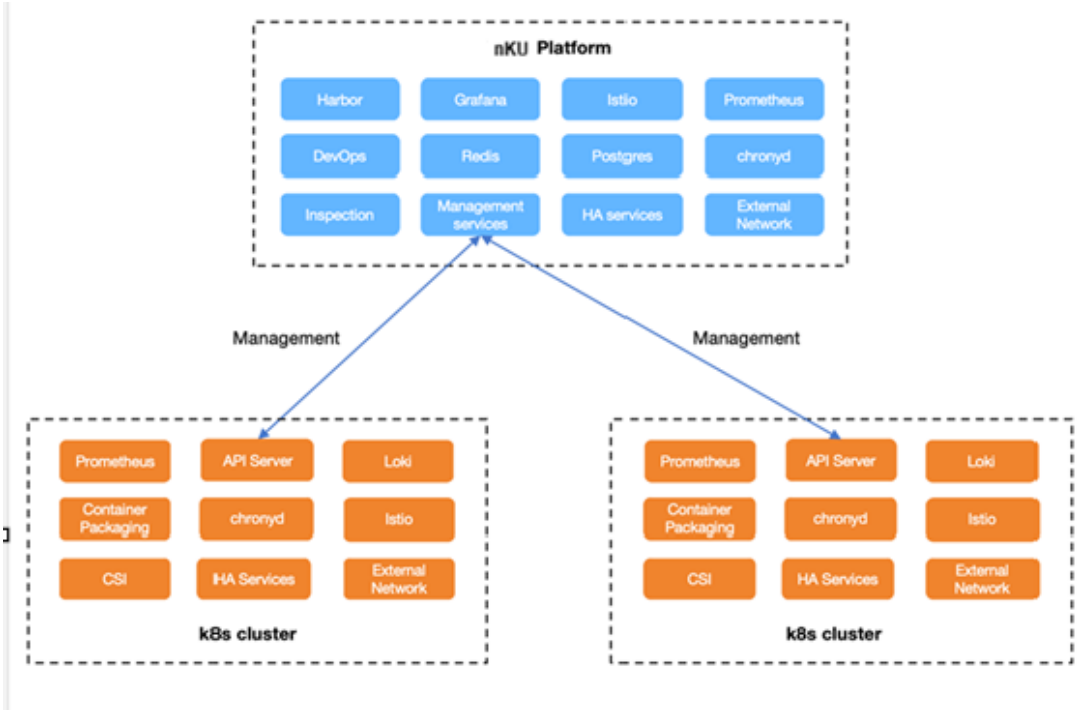
	project	Supports basic lifecycle management such as project creation, editing, deletion, etc.
		Support for adding members and namespaces to projects and centralized management
		Supports resource segregation and privilege control through projects
	subscribers	Support for local and third-party users
		Supports basic lifecycle management such as creating, enabling, disabling, deleting, and resetting passwords for local users
		Supports basic lifecycle management such as synchronization and deletion of third-party users
		Support for setting users as administrators to gain access to the platform
		Support for removing users as administrators
		Support for adding users to a project to get project permissions

		Support for removing users from projects
	namespace	Supports basic lifecycle management such as creation and deletion of namespaces
		Supports setting namespace resource quotas to effectively control user/team resource usage
		Supports setting GPU memory quota for namespaces by manufacturer to effectively control user/team GPU resource usage.
		Support for setting namespace default resource limits
		Support for adding namespaces to/removing them from projects
		Support for adding labels, annotations to namespaces
set up	mailbox server	Supports basic lifecycle management such as adding, modifying and deleting mailbox servers.
		Supports STARTTLS, SSL/TLS and NONE encryption types.
	Theme Appearance	Support customized setting of platform theme appearance, such as platform title, Logo, etc.

	3rd-party authentication	Supports seamless access to 3rd-party authentication systems by adding 3rd-Party Authentication Servers to enable corresponding accounts to log in to the cloud platform without confidentiality.
		Support adding 3rd-Party Authentication Server Type: OIDC Server
		Supports setting synchronization mapping rules for OIDC servers, including: user mapping rules
		Supports basic lifecycle management such as adding, modifying, and deleting 3rd-Party Authentication Servers.
	AccessKey Management	Provide identity credentials to access the platform API with full creator permissions
		Supports basic lifecycle management such as generation, enabling, disabling, and deletion of AccessKey.

3 Key Technology Design

3.1 Overall Technical Architecture Design



The overall technical architecture of NexaVM nKU enterprise container service includes two parts: the nKU management platform and business Kubernetes clusters, and the nKU management cluster is responsible for managing multiple business clusters.

The NKU management cluster includes the following core components:

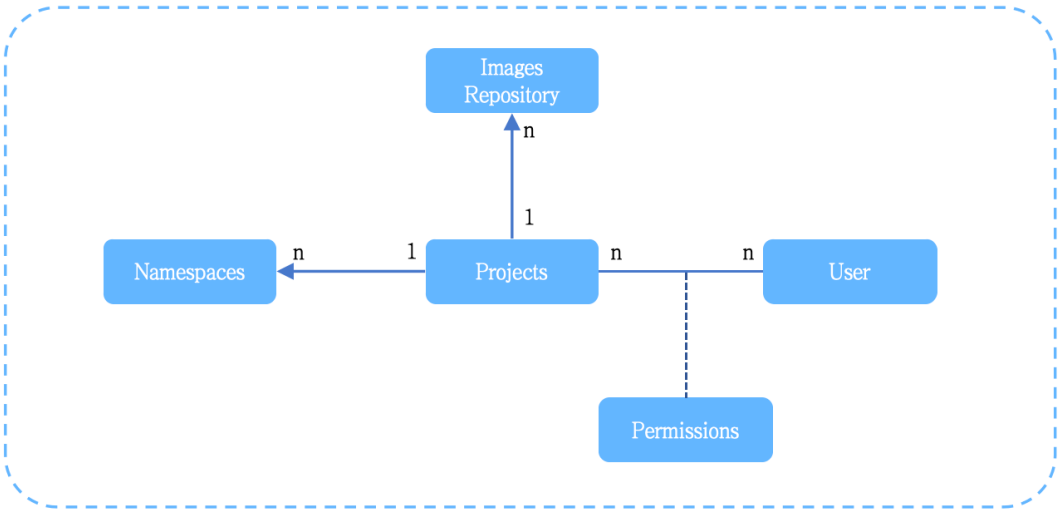
- Harbor: provides storage and distribution of container images
- Grafana: Provides Cluster and Container Monitoring Data Viewing Capabilities
- Istio: providing microservices governance capabilities
- Prometheus: Data Collection for Cluster and Container Monitoring
- DevOps: Providing CI/CD functionality
- Redis: used to store business data for NKU's managed services
- Postgres: used to store business data for NKU's managed services
- chronyd: for time synchronization between management platform and business k8s clusters
- Inspection service: provide one-click inspection function
- Managed services: nKU Management Console

- Highly Available Components: Enabling Rapid Recovery of Container Services
- External network: provide LoadBalancer type Service function

The business k8s cluster includes the following core components:

- Prometheus: Data Collection for Cluster and Container Monitoring
- API Server: K8s important management API layer, responsible for providing restful api access to endpoints, and data persistence to etcd.
- Loki: Provides container log collection and query functionality
- Container Packaging: Realize packaging running containers into an image and push it to the image repository
- chronyd: for time synchronization between management platform and business k8s clusters
- Istio: providing microservices governance capabilities
- Storage plug-ins: used to interface with external storage clusters, e.g. Ceph, NFS
- Highly Available Components: Enabling Rapid Recovery of Container Services
- External network: provide Load Balancer type Service function

3.2 Multi-Tenant Design



The nKU multi-tenant management feature includes the following key concepts:

- Users: Users created in the platform, support operations such as

enabling, disabling, resetting password, setting as administrator, canceling administrator, joining projects, and so on.

- Permissions: read-only and read/write permissions are included to control sub-user operating privileges.
- Project: a project is a type of tenant , which is used to implement resource isolation and user management.
  - Once a user joins a project, he or she has the privilege to view and manage the resources under the project.
  - All resources created by the user under the current project belong to this project.
  - Different users can view the same resources under the same project, but resources between different projects are isolated from each other.
  - Supports the inclusion of users belonging to the same team or with the same function in the same project, thus ensuring that different teams have clear permissions, resources will not interfere with each other, to protect the security and confidentiality of project data.
- Namespace: Provides virtual isolation for Kubernetes clusters, where resources in different namespaces are isolated from each other.

Namespaces have the following characteristics:

Resource Segregation and Privilege Control

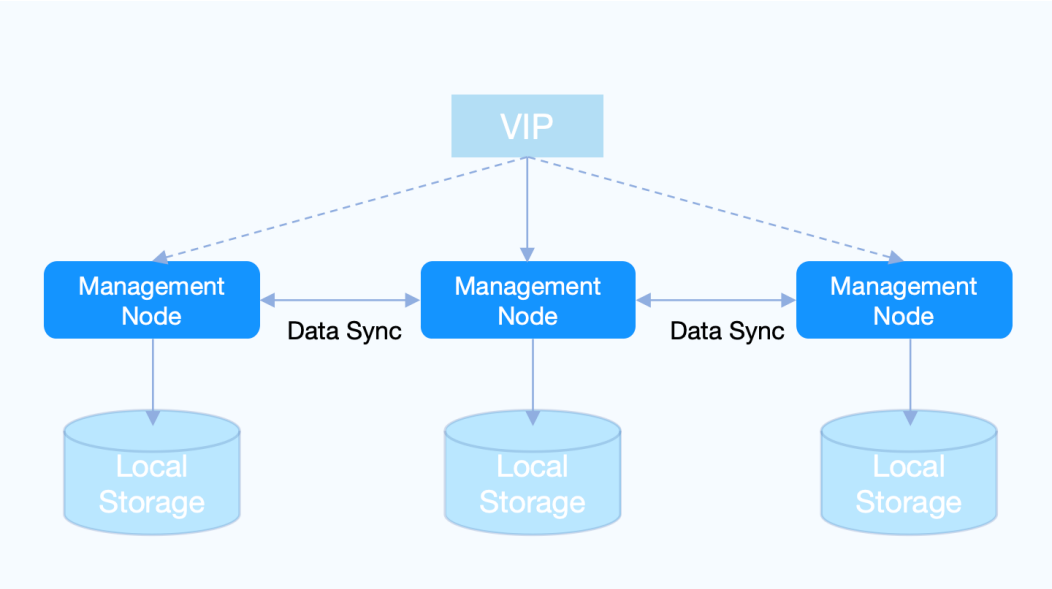
- Resource isolation: Users can create multiple namespaces according to business requirements, and different namespaces manage resources for different purposes, thus realizing the effective distribution of workspaces, for example, placing the development environment, testing environment, and the coordination environment resources under different namespaces.
- Permission Control: Namespaces can be assigned to different projects so that members of the project can view and manage resources under the namespace. A namespace can only be assigned to one project at a time, and no other users can view and manage the namespace except the members and administrators under the project.

Resource usage control

- Users can set resource quotas for namespaces, which can be used to effectively control resource usage when multiple teams or users share cluster resources.

- Support setting resource quota, when the corresponding resource in the namespace exceeds the set quota, it will not be able to create new resources, the existing resources will not be affected:

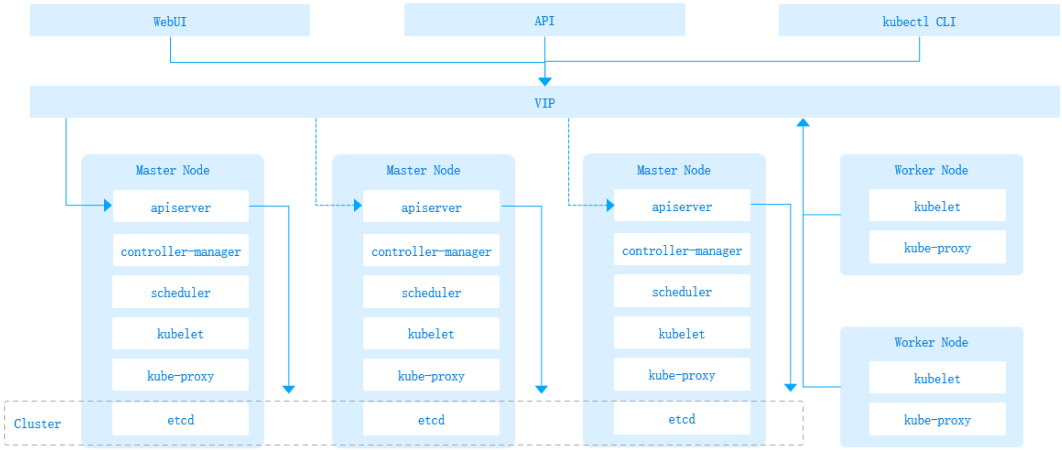
3.3 Highly Available Design of the Management Platform



nKU control plane high availability based on metallb to achieve the master-slave high availability program, by the metallb external VIP (Virtual IP), external access to the nKU platform through the VIP, when the master node failure, by the metallb to achieve the master-slave VIP switching, when the original master node recovery, continue to maintain the node role of the slave node. postgres high availability program using the mainstream master-slave replication scheme, the master node to copy the data to the slave node. Availability program uses the mainstream master-slave replication scheme, where the master node replicates data to the slave nodes. It also uses shared storage to store persistent data for platform components.

3.4 Kubernetes Cluster HA Design

The management node of the container service base Kubernetes cluster HA is designed using the officially recommended high availability scheme of Kubernetes and consists of three management nodes to form a container service cluster, each management node mainly contains kube-apiserver, kube-scheduler-manager, kube-controller Each management node contains three components. At the same time, each management plane contains an etcd node, and the cluster composed of three etcd nodes serves as the metadata storage node of Kubernetes, realizing a high degree of reliability and consistency of the cluster data.

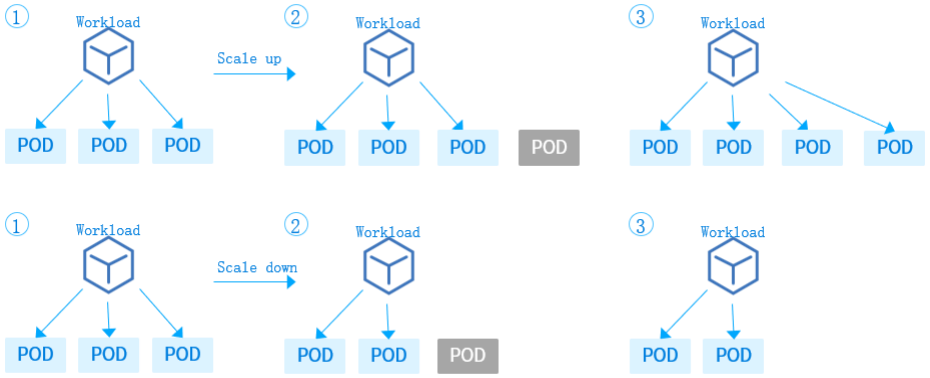


3.5 Container Orchestration

Fully integrated Kubernetes container orchestration engine, providing a standard enterprise-class container operating environment. Provides standard container resource management through a unified management interface, which improves product ease of use and reduces the learning cost of users using the container platform. At the same time, it realizes flexible container orchestration scheduling, rapid deployment and efficient delivery, and provides a platform-based container infrastructure. The functional modules provided by the container service platform have the following characteristics.

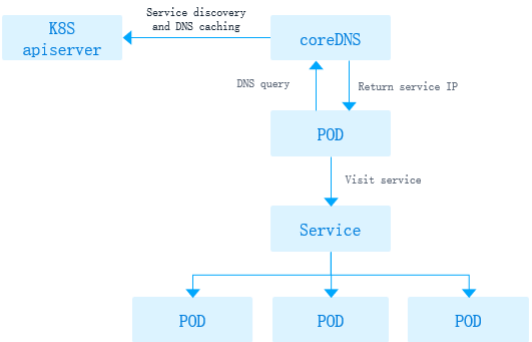
3.5.1 Horizontal Expansion Design

Whether your load is a single component or composed of multiple components working together, in a container service you can run it in a group of containers. You can also horizontally scale up or down containers through a simple UI interface or by setting up elastic scaling policies, which helps to quickly deploy or reclaim business resources at business scale.



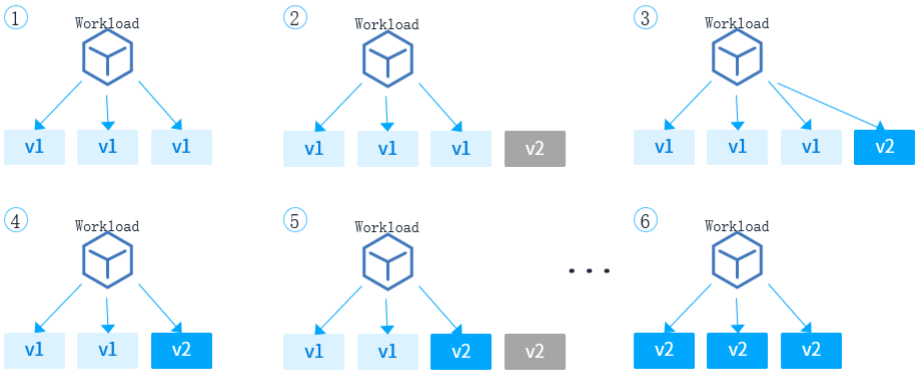
3.5.2 Service Discovery and Load Balancing

Container services have built-in DNS functionality. On DNS, each service corresponds to one or more A records, providing the same DNS name for a set of container groups for the purpose of service discovery. Exposing container application services using DNS names or IPs can effectively load-balance traffic to different container pods, thus spreading the access traffic to keep the container application stable. Users do not need to use additional service discovery mechanism, can be based on the container service's own ability to achieve service discovery and load balancing.



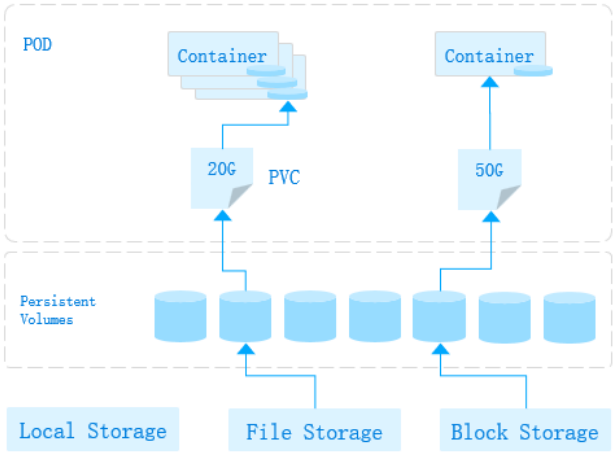
3.5.3 Rolling Updates

Rolling updates use batch-by-batch individual updates for each instance instead of all instances at the same time to achieve the update and upgrade method without interrupting the service. You can update the running applications in the container at one time or in a batch, depending on the update requirements of the application.



3.5.4 Storage Arrangement

Support for docking distributed storage, built-in CSI storage plug-in to meet the container data persistence needs. Meanwhile, the unique storage orchestration function encapsulates and optimizes the storage class (storageClass), storage volume (persistentVolume), and storage volume claims (persistentVolumeClaims) of the Kubernetes resource type and uniformly provides container data persistence services through the "pvc". "to provide container data persistence services.



3.5.5 Application Management

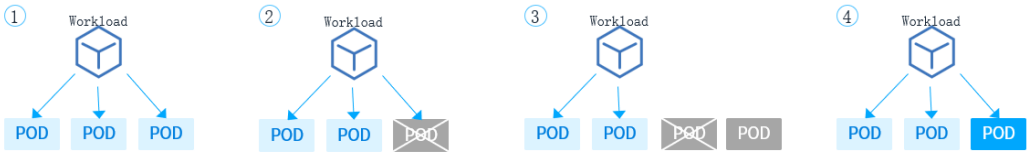
Thanks to the Kubernetes package management tool helm, a set of many dispersed Kubernetes resources can be packaged and unified maintenance and management, which has the characteristics of shareable, easy to distribute, fast build, parameter separation, etc. gradually become the best way to build software for Kubernetes. NexaVM Edge integrates helm's excellent package management capabilities and gradually develops a unique application management methodology based on its experience and customer feedback. NexaVM Edge integrates helm's excellent package management capabilities, and based on experience and customer feedback, has gradually formed a set of unique application management methods. NexaVM Edge integrates helm's excellent package management capabilities and has gradually developed a unique application management methodology based on experience and customer feedback. Application management provides application lifecycle management of helm chart packages, unified view of business containers' Installation Status, Kubernetes resource configuration, chart package installations, and version updates and rollbacks.

3.5.6 Resource Requests and Limits

Provides resource quota management for CPU and memory through resource requests and resource limits. When a container specifies resource requests and limits, the container service can make better decisions to manage the container's resources.

3.5.7 Self-Healing

In the event of container startup failure or node failure, the failed container is restarted, replaced or redeployed to ensure that the expected number of replicas is reached. It also works with health check settings to automatically kill containers that fail the health check (determined to be unable to provide service) and will not process client requests until it is ready to do so, ensuring that online services are not interrupted.

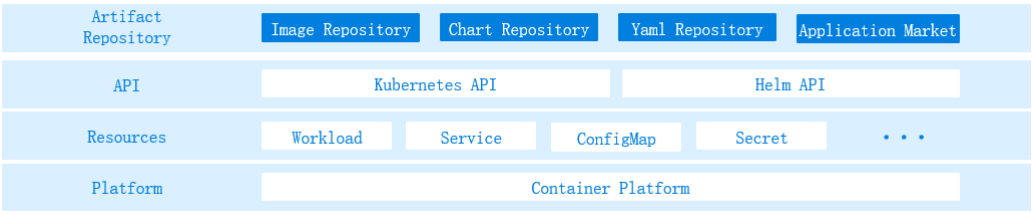


3.5.8 Configuration Management

Supports the use of configmaps and secret dictionaries to store and manage configuration information, such as application configuration files, passwords, OAuth tokens, or other file information. Keys and application configurations can be deployed and updated without rebuilding the container image and without exposing the keys in the stack configuration.

3.6 Delivery Center

Delivery Center provides the ability to manage three types of repositories: YAML template repository, image repository, and application market, which makes it easy to install applications with just a few clicks. At the same time, it also meets the needs of beginners and advanced users to develop and manage applications through standardized delivery methods, and the formation of a standardized delivery process is conducive to the construction of industry standards.



3.6.1 YAML Template Repo

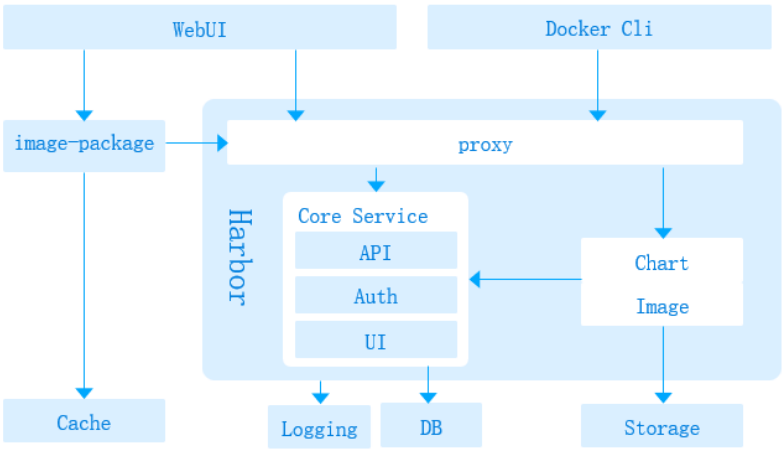
Provides easy-to-use template functionality while allowing users to edit and save templates. Enables users to create the resources they need from default templates without having to master the complex helm language.

3.6.2 Component Micro Servicing

The components of the delivery center are designed and decoupled as microservices. For example, image repository component services, application deployment services, image packaging services, image upload/download services and so on. All services are deployed as containers for process isolation, and at the same time, CPU, memory and other resource limits can be limited so that other components are not affected in case of any service failure.

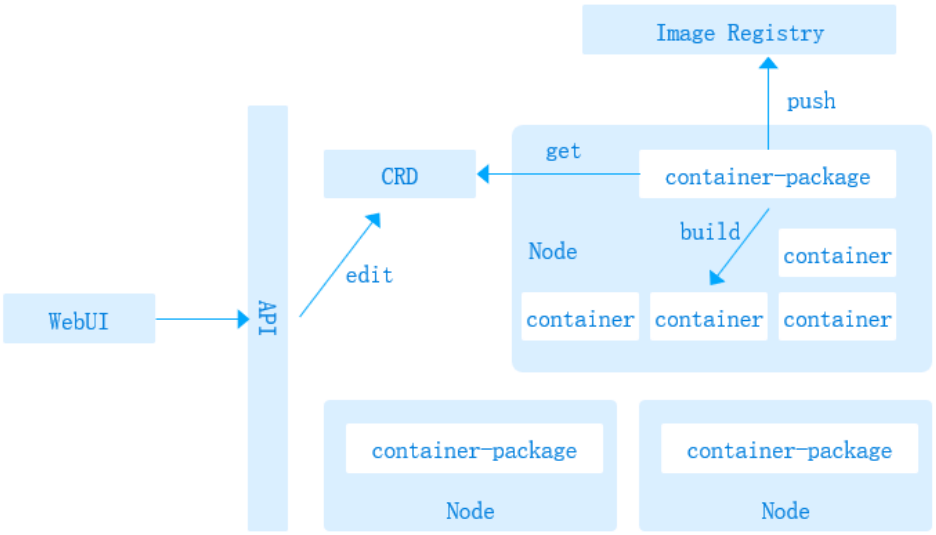
3.6.3 Repository

Local Repository provides enterprise-level container image repository management services. It helps users easily upload, download and manage container images and application packages through online upload, command line or UI upload. Through the permission management function, different users can have different access rights to the repository, to effectively prevent unauthorized access.



3.6.4 Container Packaging Images

Delivery Center provides container-package components to easily save changes to the container (such as installing software, configuration files, uploading data, etc.) as an image to the platform's built-in local repository, so that users can make container image repository simply and efficiently.



3.6.5 Application Visualization Deployment

The application market uses helm to realize application deployment. In addition to providing the regular deployment of helm to provide YAML parameter configuration, it also provides a form to fill in the configuration parameters, and provides a comparison with the original YAML modification, according to the content of the form backfill to the YAML and complete the deployment of the application. The form-based configuration helps developers to quickly configure the necessary parameters, and at the same time reduces the problems of formatting errors and configuration omissions caused by editing YAML.

3.7 Operations and Maintenance Center

3.7.1 Monitoring and Alarm Module

The monitoring module collects monitoring metrics at the cluster, storage, workload, and container service levels, and puts cluster components, application status statistics, resource usage, and other aggregated data that need attention at the operation and maintenance stage through a unified multi-monitoring chart, which makes it easy to intuitively view multiple monitoring metrics for a variety of resources, and to quickly obtain the status of the cloud-native clusters. In addition, it is also compatible with the standard Prometheus API, which

can be interfaced with Grafana and other mainstream monitoring systems.

Based on the data indicators collected by the monitoring module, the alarm module flexibly defines the alarm rules through visualization, pushes the alarm notification in real time after the unified analysis of the data, and notifies the operation personnel in the first time by the browser page notification, email and other ways, which effectively improves the efficiency of the operation and maintenance and saves the operation and maintenance cost, and meets the requirements of most operation and maintenance scenarios.

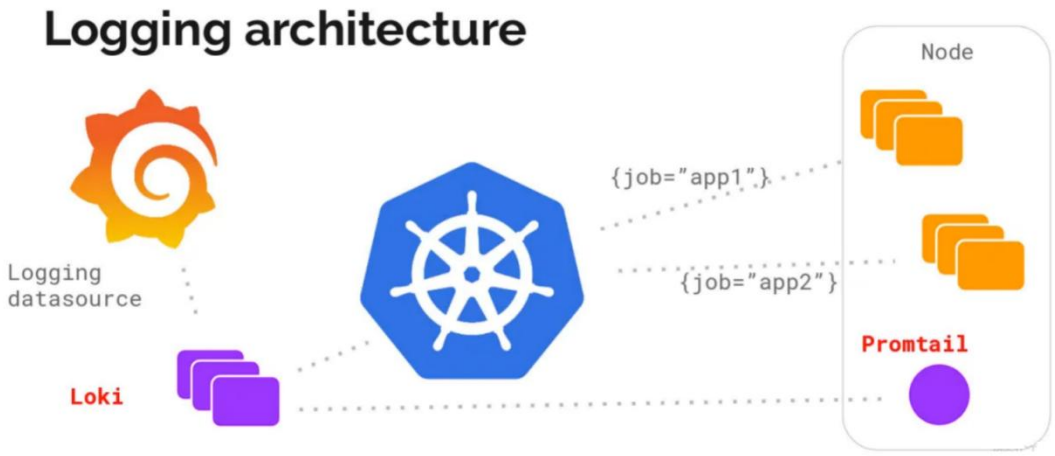
3.7.2 log Module

uKU implements log collection by integrating loki. When creating a business k8s cluster, the loki service is deployed on the management node for log summarization and indexing, and a promtail Pod is deployed on each node, which is responsible for collecting all container logs on that node and sending them to the loki service. promtail collects the standard output of the containers and the standard error output.

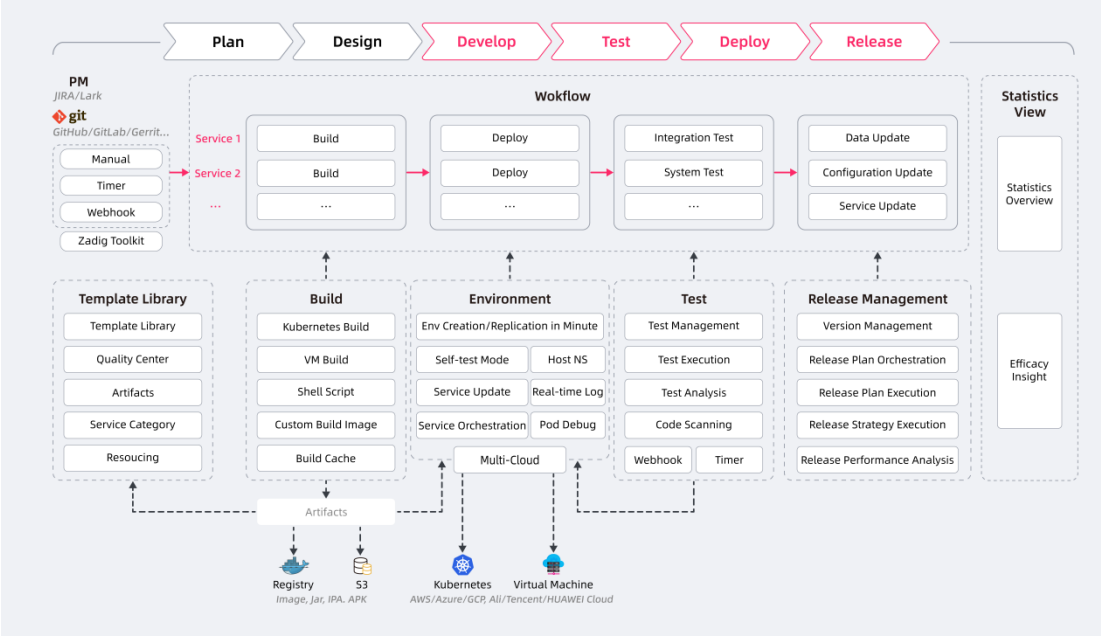
It mainly consists of log collectors, log storage and log display. Log module lightweight design and the use of labels as an index, then the traditional log collection of the scheme using full-text search to index the logs (such as EFK) is faster and more efficient, but also greatly reduces the storage of the log index. In addition, the log module has the following features:

Supports unified collection and centralized display of container logs, not limited to standard output/standard errors of containers.

Supports categorizing and querying logs by namespace, workload, container group, container, and so on.



3.8 DevOps Design



➢ Flexible and easy-to-use highly concurrent workflows

Simple configuration can automatically generate high concurrency workflow, multiple microservices can be built, deployed and tested in parallel, greatly improving the efficiency of code verification. Customized workflow steps, with manual approval, flexible and controllable to ensure the quality of business delivery.

➢ Cloud-native environment for developers

Create or replicate a complete set of isolated environments in minutes to cope with frequent business changes and product iterations. Based on a full set of benchmark environments, developers quickly provide a set of independent self-test environments. One-click hosting cluster resources can easily debug existing services and verify business code.

➢ Efficient and collaborative test management

Conveniently interface with Jmeter, Pytest and other mainstream testing frameworks, cross-project management and precipitation of UI, API, E2E test case assets. Provide developers with front-end test validation capability through workflow. Fully release testing value through continuous testing and quality analysis.

➢ Powerful and maintenance-free template library

Cross-project sharing of K8s YAML templates, Helm Chart templates, build templates, workflow templates, and so on, realizes unified management of configuration. Based on a set of templates, hundreds of microservices can be created, and development engineers can use a small number of configurations for self-service, which significantly reduces the burden of operation and maintenance management.

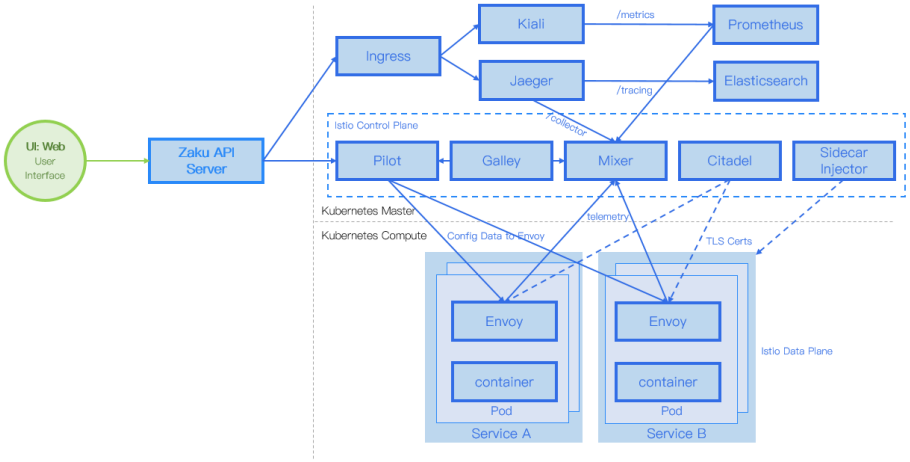
➢ Secure and reliable install management

Customized workflow connects people, processes, internal and external systems for compliance and approval, and supports flexible scheduling of blue-green, canary, batch grayscale release strategy. Installation Status is presented through multi-cluster and multi-project perspectives, realizing the transparency and reliability of the installation process.

➢ Objective and accurate performance insights

Comprehensively understand the state of system operation, including cluster, project, environment, workflow, key process passes rate and other data overview. Provide objective performance metrics for project dimensions such as build, test, deployment, etc. to accurately analyze R&D performance shortcomings and promote steady improvement.

3.9 Service governance design



NKU realizes the service management function by integrating Istio.

The core functions mainly include traffic management, application topology and trace tracking. In the deployment of K8s cluster, you can choose to turn on the service governance function, will be in the K8s cluster management node will be deployed in the Istio control plane components, enable the service governance function of each Pod will be automatically injected into the Istio agent component Envoy, in the form of Sidecar, as the core component of the data plane.

The main function of the Istio control plane is to perform traffic control by configuring and managing the Envoy agent and configuring the Mixer to enforce traffic policy and collect telemetry data (Telemetry). core component description:

- Pilot: is the core component of Istio that implements traffic management, its main role is to configure and manage the Envoy agent, and provide service discovery capabilities
- Galley: Provides configuration management and validation of the Istio API, mainly for Pilot and Mixer configurations.
- Mixer: main function is to provide policy control and collect telemetry data from Envoy agents
- Citadel: security-related components, mainly responsible for key and certificate management
- Sidecar Injector: Provides automatic injection of Envoy agents into Pods.
- The main function of the Istio data plane is to take over all traffic in and out of the service, passing and controlling all network communication between the service and the Mixer component. core component description:
- Envoy: The only data plane component in the Istio architecture, Envoy provides dynamic service discovery, load balancing, TLS, HTTP/2 and gRPC proxying, fuses, Circuit Breaking, traffic splitting, canary releases, fault injection and more.
- All Envoy's metrics data collected by Mixer is stored in Prometheus in the K8s cluster, accessed by Kiali, and the NKU platform accesses Kiali through Ingress to get data on the application topology to show the real-time application topology.
- Jaeger's Collector component periodically goes to Mixer to collect Tracing related data, which is stored in Elasticsearch in the K8s cluster and accessed by Jaeger Query, and NKU platform gets the data related to trace tracking by accessing

Jaeger.

The NKU platform interacts with the Pilot by defining various Istio-implemented K8s CRDs (e.g., VirtualService, DestinationRule).

3.10 GPU Management Scheduling

GPU management scheduling container-based ultra-precise GPU virtualization technology, fully enhances the utilization rate of graphics cards and GPU memory, greatly reducing the cost of enterprise hardware and resource management support MB level (1%) granularity of the graphics memory slicing technology, without additional authorization. Supports multi-tenant computing power isolation based on a single card and supports domestic card compatibility.

Architecture Description

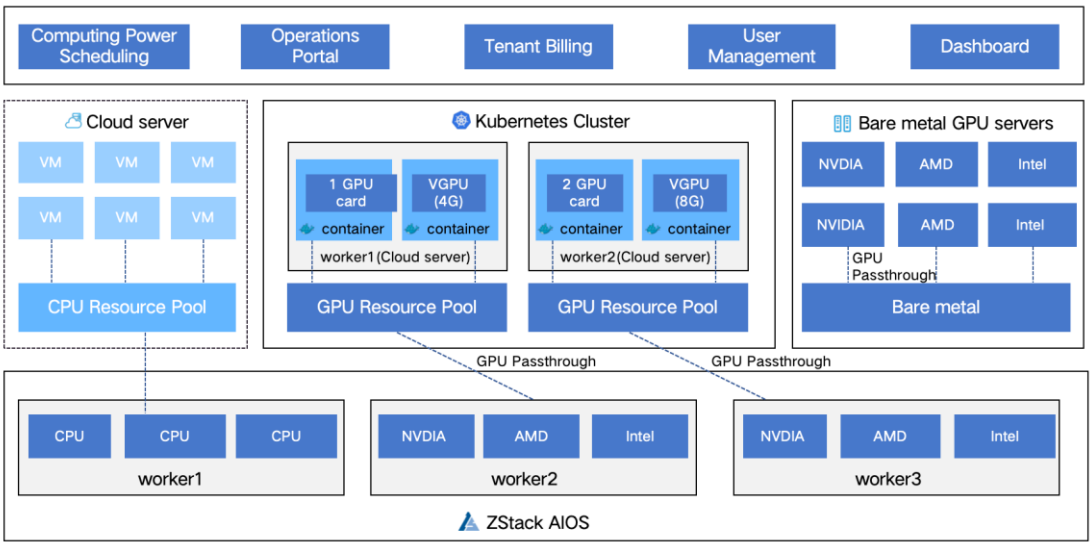
- Compute nodes need to pass-through GPUs to cloud hosts for deployment using cloud hosts
- NKU GPU virtualization component automatically identifies and virtualizes GPUs on the server, enabling multiple containers to share a single GPU card without the need to purchase additional GPU virtualization licenses.

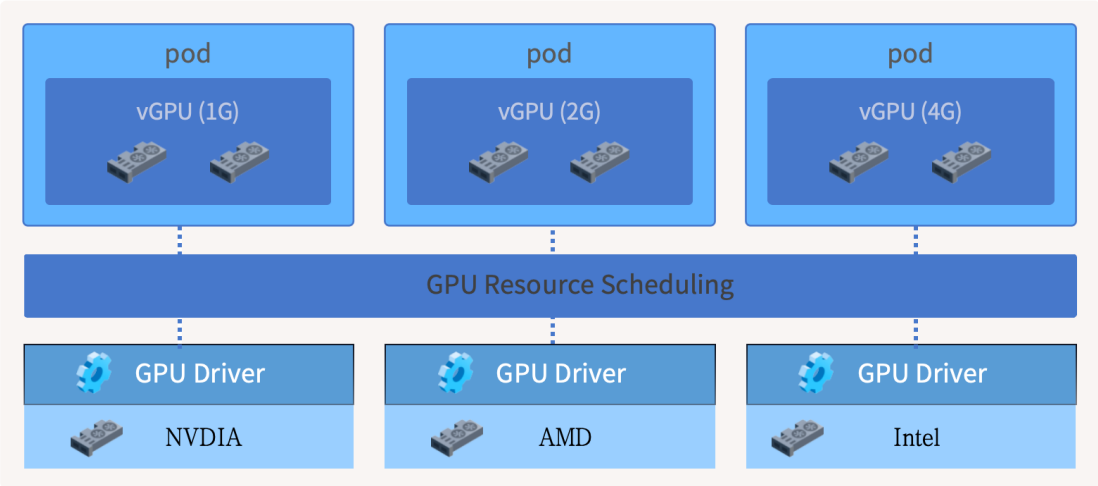
GPU Scheduling Management

- GPU card auto-recognition
- Supports fragment scheduling to improve GPU resource utilization
- Supports whole card scheduling, single container using multiple cards to meet performance requirements
- Supports isolation of GPU and GPU memory usage between containers on the same card
- Supports both domestic and NVIDIA cards

GPU, CPU Fusion Computing Architecture

- Automatic scheduling of GPU applications to GPU nodes
- Fully utilize the CPU resources of the GPU host





4 Highlights & Features

4.1.1 Featured Functions

4.1.2 Unified Management of Multiple Clouds and Clusters

- Multi Cloud Unified Management

  Through the cloud deployment of a unified operation and maintenance center platform to achieve unified monitoring, unified management and unified maintenance for NexaVM NKU enterprise-class container service deployed in multiple clouds or regions.

- Cloud Edge Data Synergy
  The cloud with multiple edge container services allows for management as well as collaboration of multiple types of data.
- Edge Multi-Cluster Autonomy
  The local NexaVM NKU enterprise container service at the edge can autonomously manage multiple self-built Kubernetes clusters as well as existing external Kubernetes clusters, realizing unified management of multiple clusters without the need to intervene on the cloud platform.

4.1.3 Out-of-the-Box

- Quick Installation
  The platform can be automatically deployed by configuring the server IP and other information through the guided installation program.
- Offline Installation
  Offline installation solves the problem of installing in scenarios with no Internet access or restricted network, while saving time for installation.
- Integrated Cluster Management
  It supports the creation of new Kubernetes clusters on the nKU platform, as well as integrating Kubernetes clusters of different Kubernetes versions and vendors on a unified platform, with no dependency on Kubernetes versions.

4.1.4 Applications Markets

- One-click Deployment
  NKU's built-in application market, which includes applications commonly used by enterprises, supports search by name and indexing by category, and can be deployed with one click, which can help enterprises build their environments quickly.
- Version Management
  The content of each version change of the application can be clearly viewed. When there is a new update of an application in the store, the deployed application can be upgraded to the latest version with one click and the application data can be

retained.

- Graphic Presentation
  An interactive, interfaced approach to application authoring allows users to create their own application templates without the need to have a basic knowledge of authoring file syntax.
- Application Topology
  The application topology helps to understand the dependencies between services and dissects the related services between containers in a visual way.

4.1.5 Elastic Telescoping (i.e. flexible)

- Various scaling types
  Supports manual and autoscaling types. The manual scaling type allows users to manually expand or contract the application according to its current state or load and specify the number of containers to be scaled each time. The autoscaling type allows the system to automatically scale according to the current metrics of the application.
- Resilient self-healing
  By setting a minimum number of healthy containers and performing uninterrupted health checks on containers, you can always keep the number of healthy running containers and automatically replace unhealthy containers to ensure real-time availability for everyday scenarios.
- Indicators for different applications
  Supports load determination based on many different application metrics, such as: CPU usage and memory usage.

4.1.6 Multi-dimensional Monitoring Alarms

- Real Time Monitoring
  Supports monitoring of regular metrics (CPU, memory, volume, network) of hosts and applications to achieve all-round, multi-dimensional real-time monitoring.
- Historical Data
  Customize the monitoring history data retention period to

analyze the monitoring data trend in the history period and understand the resource usage of the application or container.

- Multiple Alarm Rule Types
  Supports 2 types of checking: threshold checking and abnormality checking to meet different monitoring needs in various scenarios.
- Multiple Alarm Notification Methods
  It supports the regular alarm notification methods based on browser page notification and email, and at the same time, it can quickly integrate enterprise WeChat, enterprise nails and other alarm methods.
- Seamless Integration
  Based on Prometheus mainstream open-source monitoring framework can be seamlessly connected with enterprise monitoring platform, big data analysis platform.

4.1.7 Comprehensive Container Log Collection

- Harmonized and Centralized Collection
  Supports unified collection and centralized display of container logs and collection of logs from the standard output of containers.
- Quick Search
  Supports keyword query to quickly retrieve container logs, match the context of the corresponding logs, and locate the problem faster.

4.2 Core Advantages

4.2.1 Conform

- A complete solution resulting from the integration of quality community resources.
  The container ecosystem is evolving very rapidly, and it is necessary to eliminate the wheat from the chaff and deeply integrate the strengths of each family. nKU has gathered years of experience in system construction and operation and maintenance and provides a one-stop container service solution for enterprise users.

4.2.2 Compatibility

- Application-centric container service that balances microservices applications and legacy applications
  While containers are naturally suited to microservices architecture applications, there are still many traditional applications that need to run in containers, and micro servicing these traditional applications is time-consuming and unnecessary. nKU supports containerizing traditional applications, and scheduling them as applications, just like microservices architecture applications, to help enterprises migrate traditional applications to the cloud.

4.2.3 Performances

- Seek higher performance and reliability with the ability to integrate multiple types of storage and networking solutions.
  In production environments, storage and network performance loss is a very important issue. nKU supports multiple types of storage, empowering stateful containers with secure and reliable persistence, while balancing performance and flexibility with cross-host inter-container communication.

4.2.4 Operation and Maintenance (O&M)

- Comprehensive O&M Services and Toolchain
  The difficulty of cloud platforms lies in operation and maintenance. The high-density deployment and rapid changes of container clusters greatly increase the difficulty of operation and maintenance, making traditional manual operation and maintenance almost impossible. nKU provides perfect operation and maintenance services and tool chain to help operation and maintenance personnel to visualize, monitor and analyze the cluster status, and customize operation and maintenance policies to realize automation easily.

4.2.5 Liberalization

- Open API, fully compatible with community API standards
  The rapid changes in the container ecosystem make any closed technology immediately obsolete. nKU adheres to an open philosophy with a completely open API and 100% compatibility with the community-standard Kubernetes API. Leveraging community resources with no vendor lock-in.

4.2.6 Controllable

- Resource scheduling and allocation, project staff authorization, everything is under control.
  nKU uses a project-based multi-tenant management model to support resource allocation and isolation between projects and supports a project management model to customize project users for fine-grained access control. Integration with existing enterprise authentication systems to meet security compliance audit requirements.

5 Typical Application Scenarios of nKU

Whether it's traditional applications or microservices applications, open-source software or commercial software, they can be containerized and then deployed and run on the NKU platform in a unified manner. nKU will improve the performance, security, reliability, and reduce the operation and maintenance costs of these applications.

5.1 Microservices Architecture

- Accelerate business iterations and reduce correlation risk.
  The microservices application splits a monolithic application into several loosely coupled services that can be developed and deployed independently. Developers can use the most suitable tools and technology stack to develop individual services and deliver them as mirrors. nKU provides full pipeline support from development to deployment for microservice architecture applications, helping organizations achieve agile development.

5.2 Continuous Integration & Continuous Deployment

- Increase Delivery Efficiency by more than 10 times
  Migrate to a modern, rapid iteration pipeline that utilizes continuous integration and continuous deployment to ensure the delivery of more reliable software. nKU helps organizations automate the entire DevOps process from code submission to application deployment, automatically building images and testing, accelerating the handling of code changes, reporting issues ahead of time, and improving the quality of their software.

5.3 Big Data & Machine Learning

- Rapidly build data processing platforms and quickly allocate scheduling resources
  NKU helps enterprises quickly build big data and machine learning platforms on existing host clusters, scheduling computing resources for data analysis and training in seconds and avoiding the impact on existing business through isolation technology. It helps data engineers to quickly obtain resources for data analysis without the need for time-consuming and laborious request resources and building environments.

5.4 Hybrid Cloud Elastic Scaling

- Cross-cloud automatic expansion/reduction to face traffic changes with ease
  Cross-region, cross-cloud deployment of workloads, whether on physical, virtual or cloud hosts, using the same container image with no vendor lock-in. Monitor application and host metrics, triggering autoscaling when thresholds are reached, leveraging the unlimited elasticity of the public cloud and automatically reclaiming resources.