Computer Fraud and Security
ISSN (online): 1873-7056

Application of Modernization Platform in Container Cloud Platform Management and Practice

Shanshan Pan*
School of Data Science, Xi'an Eurasia University, Xi'an, Shaanxi, China
**Corresponding Author.*

Abstract:
With the rapid development of industrial internet platforms, enterprises' demand for cloud-native architectures has significantly increased to meet the requirements of large-scale business applications for flexibility and scalability. However, many enterprises face the challenge of lacking standardized and normalized container cloud platforms during the process of digital transformation. This paper proposes and introduces an application modernization platform based on Kubernetes, aiming to provide enterprises with key functions such as efficient cluster management, automated application building and release, and multi-tenant support, thereby significantly improving the efficiency of enterprises in building and running modern applications using Kubernetes. Through in-depth analysis of the platform's overall architecture design, core technologies and implementation methods, as well as specific implementation and test results of system functions, combined with practical application cases, this paper comprehensively demonstrates the application effects and significant advantages of the platform in practice.

Keywords: container cloud platform, Kubernetes, application modernization, cluster management, multi-tenant

INTRODUCTION

Research Purpose

With the widespread application of cloud computing and containerization technologies, cloud-native architecture has gradually become the first choice for enterprises to build and deploy applications. However, in embracing this trend, enterprises generally face the following key challenges: how to efficiently manage and operate Kubernetes clusters, and how to achieve rapid application building and release. These challenges not only restrict enterprises' pace of technological innovation but also put forward higher requirements for their business agility. This paper aims to explore and design an application modernization platform that integrates various key technologies and methods to create a complete container cloud platform solution [1]. This platform will provide enterprises with efficient Kubernetes cluster management capabilities and automated application building and release processes, helping enterprises comprehensively enhance their digital transformation efficiency and solve many pain points encountered during the implementation of cloud-native architecture.

Project Background

As the construction of industrial internet platforms continues to advance, enterprises increasingly need to leverage cloud-native architectures to achieve higher flexibility and scalability when responding to large-scale business requirements. However, in practical applications, many enterprises face numerous issues in Kubernetes cluster management, application deployment, and operations and maintenance due to the lack of a unified and standardized container cloud platform. These issues include low deployment efficiency, high operation and maintenance costs, and the absence of a stable development environment. Therefore, developing a Kubernetes-oriented container cloud platform solution has become an urgent need for enterprises' digital transformation [1]. This platform must not only meet the efficiency requirements of cluster management but also support diverse application scenarios, thereby helping enterprises standardize cloud-native technology practices, reduce technical complexity, and accelerate business innovation and delivery efficiency.

OVERALL DESIGN

Overview of System Functions

An application modernization platform is a comprehensive solution dedicated to providing enterprises with all-around support for efficient management of Kubernetes clusters and the application development lifecycle. This

platform encompasses four core functions: cluster management, application management, build management, and project user management. In terms of cluster management, it supports the rapid creation of new clusters from mainstream cloud service providers, as well as the integration and unified management of self-built or hosted clusters, enabling full lifecycle management including resource optimization, version upgrades, performance monitoring, and data backup and recovery. Additionally, it leverages AI and big data analytics to enhance operational efficiency. The application management platform offers diverse deployment options, supporting rapid deployment of basic applications, refined deployment for complex scenarios, and Helm package management. It enables rapid iteration and release of applications, while ensuring high availability and stability through intelligent traffic distribution, gradual upgrades, and automated operation and maintenance functions. The build management platform supports multiple build methods, achieving end-to-end automated builds from code to images, and integrates with mainstream CI/CD tools to assist enterprises in efficiently practicing DevOps.Lastly, the project user management platform ensures the security, standardization, and efficiency of resource usage through multi-level permission control, fine-grained permission settings, multi-project collaboration, and team collaboration tools. It also enhances security and auditing capabilities.

System Design

Overall architecture design of the platform

The platform adopts a classic three-tier architecture model, divided into a front-end layer, a back-end layer, and a database layer. Each layer has clear functions and works together to form an efficient and stable container cloud platform [2]. The overall architecture is shown in Figure 1.

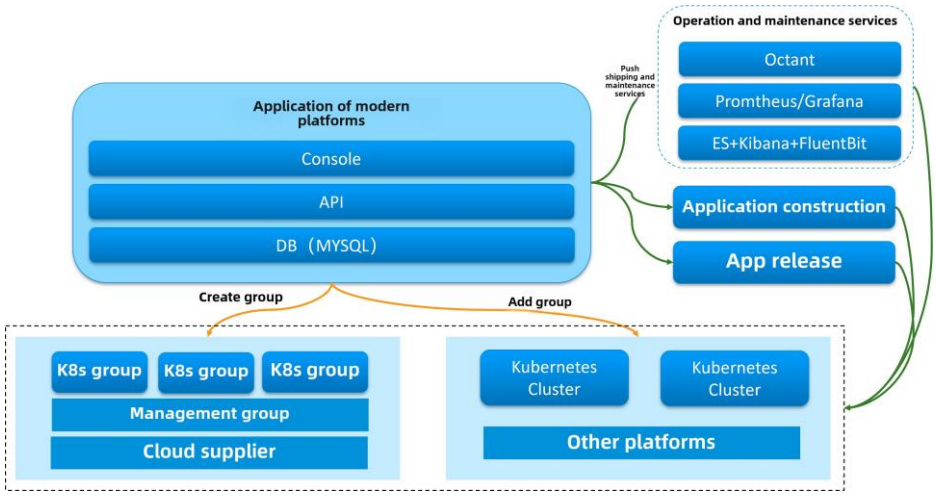

Figure 1. Overall architecture diagram of the platform

The front-end is responsible for providing an intuitive and user-friendly interface, supporting user interaction with the platform. It is developed using mainstream front-end frameworks (such as React, Vue, or Angular) and supports multi-terminal access (PC, mobile devices) through responsive design. Core functions include user identity authentication and authorization management, real-time visual display of cluster and application status, and convenient operation entries such as one-click deployment and automated building, supporting multi-language switching and enhancing internationalization capabilities.

The back-end layer serves as the core component of the platform, responsible for handling complex business logic, interacting with Kubernetes clusters, and implementing front-end/back-end communication as well as security control. Developed using high-performance programming languages such as Java, Go, and Python, and based on a microservices architecture, it ensures the system's ability to scale horizontally and handle high-concurrency requests. The core functions of the backend include the unified management of API gateways, which seamlessly interface with the front-end and third-party services to implement permission verification and traffic control. Additionally, it interacts with Kubernetes clusters through the Kubernetes API server to support operations such as cluster creation, monitoring, and maintenance, while also deeply implementing core business logic for application building, deployment, scaling, and more. Furthermore, the backend is responsible for scheduling tasks

such as timed backups and grayscale releases, and it collaborates closely with the managed Kubernetes clusters to complete these tasks. To ensure the stable operation of the platform, the backend also collects log information from both the platform and the clusters, providing real-time alerts and fault diagnosis capabilities to support the operations team [3].

The database Layer is used to store the platform's core data and user information, enabling efficient data querying, persistence, and backup. It combines relational databases (such as MySQL, PostgreSQL) with distributed storage solutions (such as ETCD, Redis). The relational databases store critical data including user information, project configurations, and operational logs. ETCD stores distributed configurations and status information for Kubernetes clusters, ensuring data consistency. Redis caches frequently accessed data to accelerate query response times. Core functions include user registration, role assignment, and authentication information storage, as well as saving project configurations, application versions, and build records. Additionally, it provides automated backup and disaster recovery capabilities to ensure data security and persistence.

The platform utilizes container technology and operates on a server cluster, leveraging container orchestration tools such as Docker Compose or Kubernetes to ensure high availability and scalability. Its backend dynamically allocates and manages cluster resources through standardized Kubernetes API interfaces. Additionally, the platform supports a distributed deployment architecture, enabling it to run across multiple servers or cloud environments, thereby meeting the high concurrency demands of large-scale users [4].

The platform deploys operational services such as monitoring, log collection, and auto-scaling components on the managed Kubernetes cluster, achieving comprehensive status monitoring and health management within the cluster. It supports running users' business systems on the managed cluster, enabling efficient application deployment and operation through the platform's application management functions. The platform also facilitates multi-cluster collaborative management, allowing applications to migrate, scale, or deploy across multiple clusters and regions to meet complex business scenarios. Through a well-designed three-tier architecture, the platform excels in functional scalability, performance, and security, providing users with a stable, efficient, and flexible container cloud platform solution [5].

Project support and resource isolation design

Each project has its own independent resource quotas (such as CPU, memory, storage, etc.), strictly controlling resource usage through Kubernetes' Resource Quota and Limit Range functions to prevent resource contention or overuse between projects. Administrators can dynamically adjust resource quotas based on project needs to meet different project stages. Real-time monitoring of project resource usage provides over-allocation alerts to ensure efficient resource utilization and fair allocation.

Each project independently configures user roles and permissions, supporting multi-level permission assignments such as super administrators, project administrators, and ordinary users. It ensures that different users can only access resources and functions corresponding to their role permissions, minimizing the risk of human error or malicious behavior. Administrators can finely set user permissions, such as restricting specific operations of certain users (e.g., deleting applications, scaling clusters). It supports users participating in collaboration in multiple projects with different roles while ensuring resource isolation between projects.

Each project independently manages its clusters, applications, and resources, with operations not interfering with each other. At the database level, independent project table designs ensure logical separation of data. Network traffic within and outside projects is controlled through Kubernetes Network Policy to prevent unauthorized traffic access between projects. Each project corresponds to one or more Kubernetes Namespaces, realizing physical isolation of resources and clear demarcation of operation domains [5].

Each project can independently configure cluster resources, application templates, CI/CD processes, etc., meeting the personalized needs of different business lines. The platform provides each project with independent operation audit logs and runtime logs, facilitating project administrators to track user behaviors and analyze issues. Through a visual interface, it provides multi-level views of projects, clusters, applications, and resources, allowing users to quickly understand the runtime status and resource usage of each project.

Through a combination of logical and physical isolation, the data and operations of projects are completely independent, preventing the leakage of sensitive information and resource abuse. The platform automatically executes isolation and permission control based on predefined security policies, reducing human operation risks. It provides independent backup and recovery functions for each project, ensuring quick recovery of project runtime status in case of failure or data loss.

User management and role permission design

The platform provides clear role division and permission management for users of different identities, ensuring efficient collaboration among various personnel while realizing multi-tenancy capabilities. The following is the user management and role permission design of the platform, with the logical architecture shown in Figure 2.

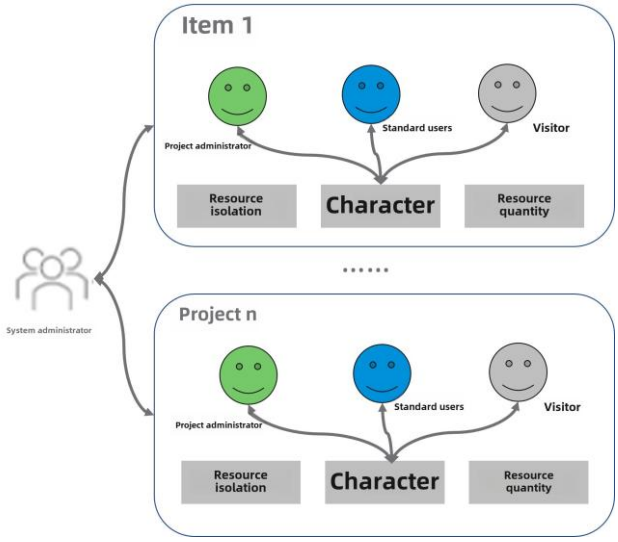

Figure 2. User management logical architecture

In the design and operation of the management platform, we first established a clear framework for roles and permissions. Super administrators are responsible for comprehensively managing all projects and users, including allocating resource quotas, setting global platform configurations, and authorizing project administrator permissions to specific users or adjusting their roles. Project administrators have full control over the projects assigned to them, enabling them to add, delete, or modify user roles within the projects, manage clusters, resource quotas, and application deployments, while also defining and adjusting CI/CD processes and policies within the projects to ensure efficient operation and isolation from other projects [6]. Their responsibilities focus on the operation and maintenance management of designated projects, ensuring the rational use of resources and smooth team collaboration. In contrast, regular user roles are dedicated to specific operational tasks within projects, such as application deployment, task execution, and log viewing, without the authority to modify project configurations or user roles.

To achieve multi-tenancy capabilities, we ensure that each project's resource quotas, namespaces, network policies, etc., can be independently configured, and data storage and management are also independent, effectively preventing information leakage or data misoperations. Users can only access and operate resources within their own projects, ensuring data security and privacy. Super administrators can flexibly adjust users' role permissions in specific projects based on actual needs, supporting a single user to hold multiple roles in different projects, such as being a regular user in one project and a project administrator in another.

Regarding workflow and collaboration mechanisms, super administrators coordinate all projects and resource allocations from a global perspective, responsible for the overall operation and maintenance of the platform and user permission management. Project administrators delve into individual projects, focusing on resource management and team collaboration, formulating and implementing workflows and policies within the projects. Regular users, centered around project goals, execute specific tasks such as application deployment, status monitoring, and log analysis. The platform facilitates seamless collaboration among different roles through a

unified permission system and notification mechanism. Project administrators can efficiently assign tasks to regular users in groups and monitor progress in real-time through operation logs, ensuring transparency and efficiency in workflows. Additionally, the platform comprehensively records operation logs for all roles, providing traceability for permission management, and is equipped with real-time notification and security audit functions to further ensure the stability and security of project operations.

Super administrators coordinate all projects and resources from a global perspective, responsible for user permission assignment and overall platform operation and maintenance. Project administrators focus on resource management and team collaboration within a single project, formulating workflows and strategies within the project. Ordinary users perform specific tasks around project goals, such as application deployment, status monitoring, and log analysis. The platform promotes seamless collaboration between different roles through a unified permission system and notification mechanism. Project administrators can assign tasks to ordinary user groups and monitor progress through operation logs, ensuring efficient and transparent workflows. The platform records operation logs for all roles, providing traceability for permission management. It provides real-time notification and security audit functions to ensure the stability and security of project operation.

Platform usage process and responsibility division

The platform's usage process is divided into three main steps based on user roles, from system initialization to business delivery and use, ensuring clear responsibilities and efficient processes at each stage. As can be seen from user management and roles, the platform serves multiple identities, and their respective scopes of work and workflows on the platform are shown in Figure 3.

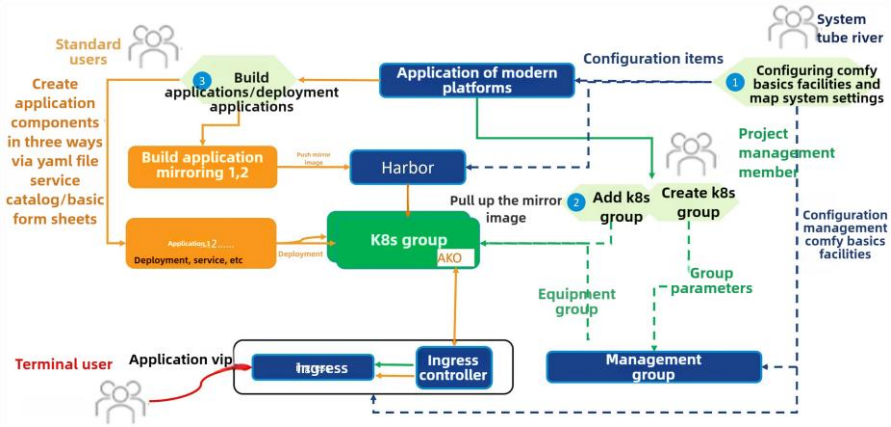

Figure 3. Platform workflow

The system administrator is initially responsible for the platform's initialization and configuration, which encompasses various system settings such as network, storage, and security policies, ensuring the platform aligns with the deployment environment. They also need to initialize a multi-tenant project architecture, create project templates or basic configuration rules, and set up multiple projects, assigning administrators for each while allocating global resource quotas such as cluster resources and storage space appropriately. During platform operation, the system administrator must regularly check the platform's status, perform system backups, troubleshoot issues, and manage global permissions and security policies to ensure platform stability and security. After completing these initialization tasks, administrative privileges are transferred to the project administrators.

Upon taking over, the project administrators create or integrate Kubernetes clusters based on project requirements, configure resource allocation and access policies to ensure efficient cluster operation. They also add project members, assign regular user permissions, and implement fine-grained permission controls to safeguard resource usage security and compliance. Additionally, project administrators plan business applications, create deployment templates and CI/CD processes, and oversee regular users in completing application deployments while allocating external access addresses to ensure smooth business launches and operations.

Lastly, regular users are responsible for executing application deployment operations on the platform, including image pulling, service orchestration, and publishing, as well as generating external access addresses and

configuring domain mapping or load balancing strategies to ensure seamless application access. They also need to monitor application status, conduct troubleshooting, and perform version upgrades to maintain application stability and updates. End-users, whether internal or accessing via the internet, can log in to the business system through the access addresses published on the platform, facilitated by the platform's configuration.

KEY TECHNOLOGIES AND METHODS

Cluster Management Technology

The platform has achieved fully automated cluster management processes through deep integration with the ClusterAPI project from the cloud-native community, significantly enhancing the efficiency and accuracy of cluster deployment and operations. Firstly, the platform supports automated cluster creation. Users can define cluster specifications declaratively through YAML files, including key parameters such as node count, version, and network configuration. For instance, by writing a simple YAML configuration file, users can specify details like cluster name, network CIDR blocks, control plane endpoints, and more. The platform then automatically generates and deploys the cluster based on these configurations, greatly simplifying the complexity of cluster creation [6].

Example configuration:

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
name: demo-cluster
spec:
clusterNetwork:
pods:
cidrBlocks: ["192.168.0.0/16"]
controlPlaneEndpoint:
host: 192.168.1.1
port: 6443
```

Secondly, the platform offers efficient resource management and operation capabilities. Users can easily achieve dynamic scaling of cluster resources, whether expanding or contracting, by modifying the node configurations in the YAML file to meet business growth or resource optimization needs. The platform leverages ClusterAPI for seamless management of underlying resources, ensuring minimal impact on application operations during these changes. Additionally, the platform supports automatic detection and upgrading of cluster versions, utilizing security validation and rolling upgrade techniques to maintain high availability during upgrades.

The platform also excels in reducing error rates and enhancing reproducibility. Cluster creation, scaling, upgrades, and other operations are all based on a unified YAML configuration file, achieving process standardization and automation. This significantly reduces manual intervention, lowering the risk of configuration errors and environmental inconsistencies caused by manual operations. Furthermore, the platform records detailed logs of every cluster operation, facilitating problem tracing for system and project administrators and improving problem-solving efficiency.

Lastly, the platform provides visual cluster management functions. Through a graphical interface, administrators can real-time monitor cluster status, including key information such as node count, resource utilization, Pod distribution, and more [7]. Administrators can also trigger operations like scaling and upgrading directly from the console, further enhancing the management experience. The cluster management mode is illustrated in Figure 4.

In summary, by adopting ClusterAPI and declarative configuration management, the platform has not only significantly enhanced the efficiency of cluster creation and operations but also provided enterprises with a standardized and automated Kubernetes cluster management solution. This approach reduces operational complexity while improving the platform’s adaptability to various operating environments, laying a solid foundation for enterprises to build cloud-native applications [8]. The code snippet demonstrates how the platform

achieves automated deployment and management of Kubernetes clusters by parsing YAML configurations, interacting with the SupervisorClient, and following other necessary steps.

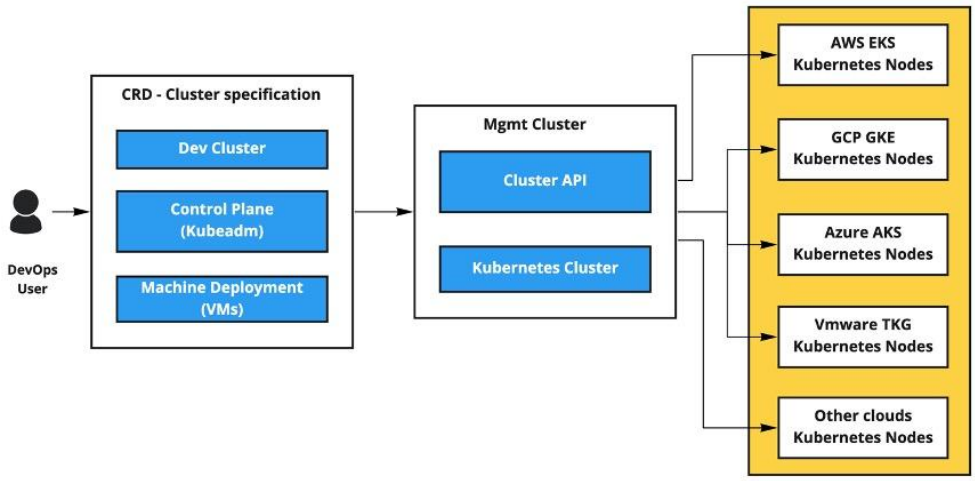

Figure 4. Cluster management mode

Code Snippet Demonstration:

```
func (k *TKG2Skubernetes) ProvisionKubernetes() (string, model.ClusterState, []byte, error) {
sc, err := k.newSupervisorClient()
if err != nil {
return “”, model.ClusterStateUnknown, nil, err
}

defer k.log.Sync()

spec := &model.ClusterTKG2Sspec{}
if err := json.Unmarshal(k.Cluster.Spec, spec); err != nil {
panic(err)
}
for _, each := range k.RelatedImageRegistries {
if len(each.CACert) != 0 {
spec.TrustedCAs = append(spec.TrustedCAs, each.CACert)
}
}

k.log.Infof(“checking state of cluster %s…”, strconv.Quote(k.Cluster.Name))
info, err := sc.GetClusterInfo(spec.Namespace, k.Cluster.Name)
if err != nil {
if k8s.IsErrObjectNotExists(err) {
k. log. Infof (“cluster %s does not exist. A new workload cluster will be provisioned.”, strconv.Quote(k.Cluster.Name))
// provisioning
p, err := sc.CreateCluster(k.Cluster, spec, false)
k.log.Debugf(“apply cluster with the following manifest:\n%s\n”, strings.ReplaceAll(string(p), “\n”, “\n> “))
```

Operation and Maintenance Services and Application Management Technology

The platform adopts Helm Chart as the core technology for operations and maintenance services as well as application delivery management, fully leveraging its powerful package management capabilities to achieve automated deployment and management of Kubernetes applications. Below are the key technical features and methods of the platform in operations and maintenance services and application management:

Firstly, Helm Chart, as a packaging format for describing Kubernetes applications, integrates all the necessary Kubernetes resource definitions for the application, such as Pods, Deployments, Services, Ingress, etc. Each Helm Chart includes a Chart.yaml file for describing the Chart's metadata, a values.yaml file for providing default configuration values, and a templates/ directory for storing Kubernetes resource templates [9]. This structure allows users to customize configurations by modifying the values.yaml file or command-line parameters, thus flexibly specifying application versions, resource limits, environment variables, and other information without altering the application code.

Example Helm chart file structure:

```
my-app/
Chart.yaml # Chart metadata
values.yaml # Default value configuration
templates / # Kubernetes resource templates (such as deployments, services, etc.)
```

Secondly, the platform supports integration with private image repositories for storing and managing Helm Charts. Users can upload packaged Helm Charts to the platform's private repository, ensuring the security and reliability of application deployments. Through API integration with the private image repository, users can obtain and deploy the required version of Helm Chart via the Helm client or the platform's provided operational interface. The platform also offers version control functionality, allowing users to revert to specific versions after multiple deployments, facilitating rollback operations to avoid application failures caused by configuration errors or version incompatibilities.

In terms of automated application deployment, the platform provides both graphical and command-line interfaces. Users only need to upload the Helm Chart and input deployment parameters, and the platform will automatically handle the entire Kubernetes deployment process [10]. Based on the Helm Chart templates, the platform can automatically generate Kubernetes resource objects and deploy them to the target cluster using the Helm client. By dynamically passing in parameters such as environment, version, and resource limits, the platform can flexibly adjust the deployment content to meet the needs of different environments. At the same time, users can also use the values.yaml file for detailed configuration customization.

```
replicaCount: 3
image:
repository: my-app-image
tag: v1.2.0
resources:
limits:
cpu: "500m"
memory: "512Mi"
```

Furthermore, the platform supports integration with the CI/CD tool chain, enabling automated deployment processes and improving the efficiency of application delivery. When developers update the application code and submit it to the code repository, the platform automatically triggers the Helm Chart deployment, ensuring that the application version is always up-to-date. After deployment, the platform automatically collects and displays the application's running status, logs, and performance metrics, helping users promptly identify potential issues [11]. Through integrated monitoring services, users can view the application's resource usage and adjust the application configuration based on actual needs.

The platform also provides grayscale release and rollback support functionality. Users can control the simultaneous operation of different application versions, gradually promoting the new version to reduce system downtime and risks. By configuring parameters, users can set a specific proportion of traffic to be switched to the new version of the application, gradually verifying its stability. If issues arise with the new version, users can utilize the rollback operation supported by Helm to restore the application to a previous stable version [12]. The platform's easy-to-use rollback functionality ensures the continuity and availability of application services.

In summary, by adopting Helm Chart technology, the platform not only simplifies the deployment process of Kubernetes applications, improves deployment flexibility and reliability, but also provides enterprises with powerful operations and maintenance management functions. Below is a code implementation snippet to illustrate part of the logic in the platform's deployment process:

```
func (k *GenericInstaller) provision(ext *model.ClusterExtension, enable bool) error {
defer k.log.Sync()
if k.UnmarshalEnvFunc == nil {
panic(fmt.Errorf("UnmarshalEnvFunc must not be nil"))
}
env := make(map[string]string)
if err := k.UnmarshalEnvFunc(k.Extension, env); err != nil {
k.log.Errorf("unexpected error: %v.", err)
return err
}
env[SharedEnvVarEnableExtension] = strconv.FormatBool(enable)
env[SharedEnvVarClusterName] = k.Cluster
env[SharedEnvVarClusterClass] = k.ClusterClass.String()
env[SharedEnvVarImageRegistry] = k.ImageRegistry.Endpoint
env[SharedEnvVarDockerConfig] = defDockerCfgDir
env[SharedEnvVarUseNodePortService] = strconv.FormatBool(!k.LoadBalancerEnabled)
// ...
}
```

Multi-Tenancy Technology

The platform achieves robust multi-tenancy support through its project user management functionality, ensuring independent resources and permissions among different tenants (i.e., projects) while also guaranteeing data and operational isolation for each tenant on the platform. The key technical features and implementation methods of the platform's multi-tenancy support are as follows:

Firstly, the platform allocates independent resource quotas for each project, including computing resources (such as CPU, memory), storage resources, and network bandwidth, to ensure that resource consumption by one project does not impact the normal operation of others [13]. Each project can manage its own Kubernetes cluster, and the platform supports a multi-cluster architecture, allowing for expansion, upgrades, and other operations tailored to project needs, maintaining independence between projects. (For example: Project A-Allocated 4 nodes, 16 GB of memory, 50 GB of storage. Project B-Allocated 2 nodes, 8 GB of memory, 20 GB of storage).

Secondly, through namespaces within Kubernetes clusters and permission controls within the platform, complete data isolation is achieved among projects. This ensures that users from one project cannot access or modify resources, configurations, and logs of other projects [14].

Furthermore, the platform supports each project in configuring its own monitoring and logging system to avoid confusion in monitoring data. It provides visual data presentation for multiple projects, enabling administrators to view resource usage and performance indicators for each project. Additionally, the platform audits and records all project operations, ensuring traceability of user actions. Super administrators can view global audit logs, while project administrators can only access logs within their own projects.

Lastly, the platform integrates an enterprise-level identity authentication system, utilizing roles and permissions for user authentication. Only authorized users can access and operate corresponding resources. It supports authentication mechanisms such as LDAP, OAuth2, etc., ensuring that the platform's identity verification and security controls meet enterprise-level requirements. All sensitive project data (such as configuration files, certificates, etc.) is encrypted for storage, and cross-project access is secured to prevent data leakage. Data transmission within the platform is encrypted using TLS to ensure data integrity and confidentiality during transit.

SYSTEM IMPLEMENTATION AND TESTING

Cluster Management Implementation and Testing

Project administrators can create and add clusters through the platform interface and modify cluster settings as needed. The platform supports operations such as creating clusters, adjusting the number of cluster nodes (including control nodes and worker nodes), and deleting clusters. Through actual testing, the platform performs exceptionally well in cluster management, efficiently completing various tasks with a simple and easy-to-use operation process [15]. The cluster management operation interface is shown in Figure 5.

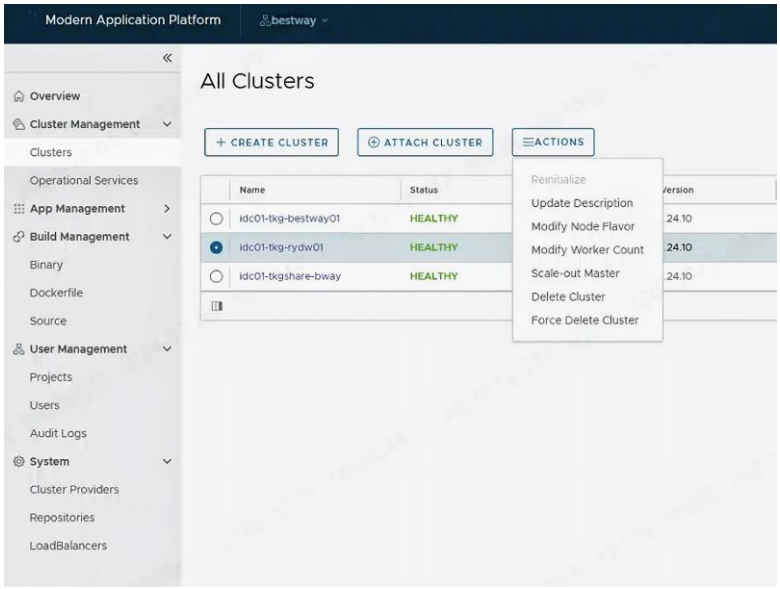

Figure 5. Cluster management interface

Operation and Maintenance Service Implementation and Testing

The platform provides rich operation and maintenance services, including cluster resource management components (such as Octant), monitoring components (such as Grafana and Prometheus), log analysis visualization suites (such as Elasticsearch and Fluent Bit), and more. Project administrators can enable, modify, or disable these functions at any time. Through actual testing, the operation and maintenance services run stably and reliably, providing enterprises with comprehensive cluster monitoring and management capabilities. The operation and maintenance service interface is shown in Figure 6.

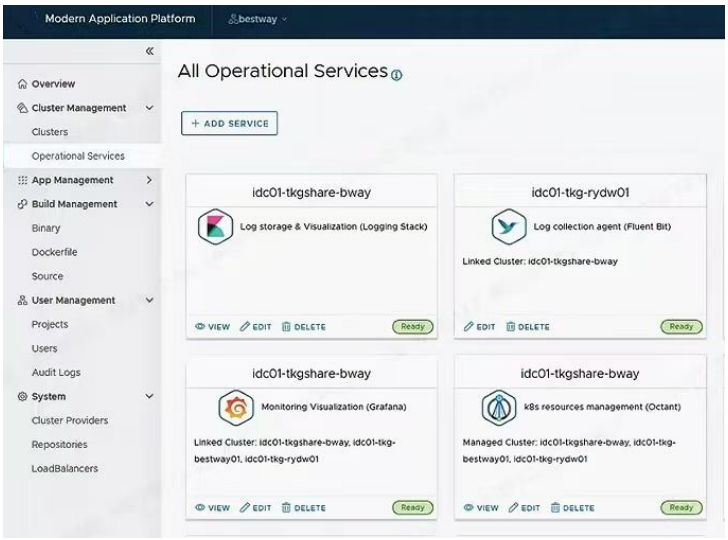

Figure 6. Operation and maintenance service interface

Application Management Implementation and Testing

The platform supports application deployment and release, offering three deployment forms: Basic, Raw, and Helm. Users can easily create, deploy, and release applications through the platform interface. Through actual case testing, the platform performs exceptionally well in application management, significantly improving the efficiency and reliability of application deployment. Additionally, the platform supports load balancing and gray-scale release functions between multiple clusters, providing enterprises with more flexible application deployment strategies. The application management interface is shown in Figure 7.

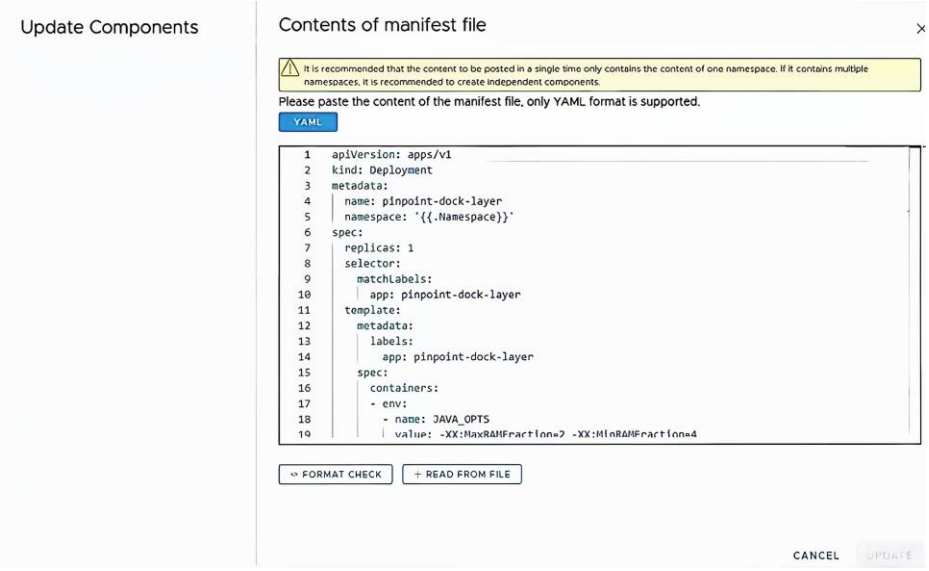

Figure 7. Application management interface

Build Management Implementation and Testing

The platform supports packaging application binaries or code into application images, and provides management and status viewing functions for the build process. Users can submit build tasks through the platform interface and view the build status and results in real-time. Through actual testing, the build process has proven to be stable and reliable, capable of generating application images that meet expectations. Additionally, the platform supports multiple build methods (such as binary packages, Dockerfiles, and source code) to cater to the build needs of different users. The build management interface is shown in Figure 8.

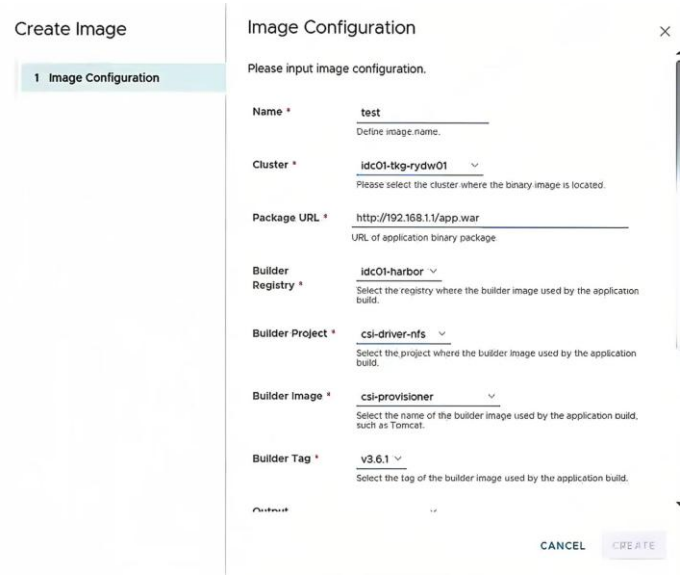

Figure 8. Build management interface

CONCLUSION

The modern application platform provides enterprises with a comprehensive container cloud platform solution by integrating various key technologies and methods. In core areas such as Kubernetes cluster management, application build and release, and multi-tenancy support, the platform performs exceptionally well, significantly enhancing enterprises' application development and operation and maintenance efficiency under the cloud-native architecture. Through this platform, enterprises can not only achieve efficient and stable container cluster management but also accelerate application delivery and deployment, reduce operation and maintenance complexity, and thus provide strong support for business innovation. In the future, as enterprises' digital transformation needs continue to deepen, we will continuously optimize platform functions to further enhance user experience and platform performance. At the same time, the platform will expand support for more scenarios and technologies, striving to provide enterprises with more intelligent, efficient, and flexible cloud-native solutions to contribute to the process of enterprises' digital transformation.

ACKNOWLEDGEMENTS

We sincerely thank the "14th Five-Year" Education Science Planning Project Review Committee of Shaanxi Province for its support and funding of the research project "Research on Personalized Learning Resource Push for the Learning Process" (Project Approval Number: SGH22Y1801). This research provides a new perspective and technical path for future in-depth research in the field of educational technology.

REFERENCES

[1] Yang Zheming, Qu Liangyong, Ni Zhipeng, et al. Research and Practical Application of Integrated Container Cloud Platform. Sino-Arab Technology Forum (Chinese and English), 2024, (02): 87-91.
[2] Zhu Xiaoliang. Design of a Kubernetes-Based Data Center Cloud Platform for Colleges and Universities. Computer Knowledge and Technology, 2023, 19(22): 81-84.
[3] Ye Chaowu, Xu Shengchao. A Design and Deployment Method of Container Cloud Platform Based on Docker. Information and Computers (Theoretical Edition), 2023, 35(18): 65-67+71.
[4] Xia Lijuan, Wuri Na, Pan Xin. Research and Implementation of the Kubernetes Container Cloud Orchestration System. Automation Applications, 2023, 64(14): 224-227.
[5] Guo Lei, Mao Lingyan. Design and Practice of an Enterprise-Level Container Cloud Platform Based on Kubernetes. Information Technology and Standardization, 2022, (09): 73-77.
[6] Qi Fazhan. Research on Cluster Management Strategies for Network Security. Information and Computer (Theoretical Edition), 2024, 36(17): 133-135.
[7] Kang Mingfa. Research on Resource Management Mechanisms in Multi-Cluster Container Cloud Environments. Hangzhou Dianzi University, 2024.
[8] Chen Junxiang. Analysis of Integration Solutions for Container Cloud and Virtualization Platforms. Jiangsu Communication, 2023, 39(01): 102-106.
[9] Xingjia L, Li P, Shijun L. An online service provisioning strategy for container-based cloud brokers. Journal of Network and Computer Applications, 2023, 214
[10] Mario V G, J. J G M. A container-based cloud-native architecture for the reproducible execution of multi-population optimization algorithms. Future Generation Computer Systems, 2021, 116234-252.
[11] Yang Xiao. Microservice Construction Based on Bazel and Helm. Information and Computer (Theoretical Edition), 2023, 35(01): 150-152.
[12] Liu Jun, Li Xiongqing, Sun Qiongwei, et al. Research and Practice of Cloud-Native Full-Link Gray-Scale Deployment Technology. Application of Electronic Technique, 2023, 49(04): 73-77.
[13] Gu Xiaoteng, Zhang Zhihua, Xu Baojian, et al. Cloud Platform Network Security Design and Application Based on Multi-Tenant Virtual Environment. Intelligent Internet of Things Technology, 2024, 56(04): 69-72.
[14] Ming Meng, Zhao Changlin. Addressing Security Issues in Multi-Tenant Technology. Network Security and Informatization, 2022, (04): 106-107.
[15] Yan Shuanghua. Sectorized Management of Project Clusters. Chemical Fertilizer Design, 2024, 62(03): 61-62.