China Search Kubernetes Application Platform Deployment Solution Postprint

Authors: SHAO Ke, Cai Guohua, Wan Guolei

Date: 2023-10-08T00:00:00+00:00

Abstract

In simple terms, Kubernetes is a system for managing containerized applications across hosts and a container orchestration tool. It implements a series of fundamental functions including application deployment, high-availability management, and elastic scaling, encapsulating them into a complete, simple, and easy-to-use RESTful API for service provision. The purpose of the Kubernetes application platform is to enhance server performance utilization, enable efficient deployment and elastic computing, provide high redundancy, and offer a customized PaaS service platform. We hope to create a standardized, modular, and universal system platform based on Kubernetes to support the application services of China Search.

Full Text

Preamble

China Search Kubernetes Application Platform Deployment Solution

Abstract: Kubernetes is a system for managing containerized applications across hosts and serves as a container orchestration tool. It implements a series of foundational functions including application deployment, high availability management, and elastic scaling, encapsulating them into a complete and user-friendly set of RESTful APIs for external services. The Kubernetes application platform aims to improve server performance utilization, enable efficient deployment and elastic computing, provide high redundancy, and deliver a customized PaaS service platform. Our goal is to create a standardized, modular, and universal system platform based on Kubernetes to support China Search' s application services.

Keywords: Kubernetes; containerization; PaaS platform; application services
Classification Code: TP393
Document Code: A

Article ID: 1671-0134(2019)05-113-05
DOI: 10.19483/j.cnki.11-4653/n.2019.05.037

Authors: Shao Ke, Cai Guohua, Wan Guolei

1. Project Background

The China Search Kubernetes Application Platform is a PaaS service platform built on Kubernetes as its foundation, integrating project management systems, CMDB management systems, and a unified CI/CD operations platform. Its purpose is to provide users with a unified management platform for Docker applications and centralized resource allocation and management. By encapsulating Kubernetes resource management interfaces as application functions and implementing visual web management, the platform significantly lowers the barrier to using Kubernetes. We also plan to gradually improve the application ecosystem within the platform, providing rich components and interfaces for user invocation. Development began in July 2018, with the platform going live in October 2018 and undergoing continuous iterative development. Currently, the platform supports cross-network-segment compute nodes, flannel and calico networking, 10-gigabit network distributed storage, and other resources, with dozens of projects running stably on the platform.

At its core, Kubernetes provides APIs externally for building high-level applications while offering a pluggable application execution environment internally. Its architecture can be understood as a layered design similar to Linux: the core layer provides fundamental APIs; the application layer handles deployment (stateless applications, stateful applications, batch jobs, clustered applications, etc.) and routing (service discovery, DNS resolution, etc.), along with Service Mesh (partially located in the application layer); the management layer encompasses system metrics (for infrastructure, containers, and networks), automation (auto-scaling, dynamic provisioning, etc.), and policy management (RBAC, Quota, PSP, NetworkPolicy, etc.), also including Service Mesh components; the interface layer consists of kubectl command-line tools, client SDKs, and cluster federation; and finally, the ecosystem layer above the interface comprises two categories—external components including logging, monitoring, configuration management, and CI/CD, and internal components such as CRI (Container Runtime Interface), CNI (Container Network Interface), CSI (Container Storage Interface), image registries, and cluster configuration and management.

2. Kubernetes Introduction

Kubernetes is a container orchestration and scheduling engine open-sourced by Google based on Borg. As one of the most important components of CNCF (Cloud Native Computing Foundation), its goal extends beyond being an orchestration system to providing a specification that allows you to describe cluster architecture and define the desired end-state of services. Kubernetes automatically brings the system to and maintains this desired state. As the cornerstone

of cloud-native applications, Kubernetes functions as a cloud operating system with a layered architecture conceptually similar to Linux.

3. Kubernetes Architecture

3.1 Deployment Solution

The node configuration for a high-availability cluster is as follows:

- **etcd nodes:** Provide the configuration management database for Kubernetes with high requirements for stability and performance.
- **Master nodes (2):** Kubernetes control plane nodes that can be co-located with etcd nodes in an HA+ active-standby configuration, requiring high stability.
- **Node nodes (20-100):** Nodes running application workloads, where machine configurations can be upgraded or additional nodes added as needed. Requirements include CPU with 24+ cores and physical memory of 64GB+.
- **Storage nodes (10-20):** Nodes providing network distributed storage services, deploying GlusterFS cluster services with node expansion capabilities as required. These require high stability, 10-gigabit networking, and large hard drives.
- **Harbor services (2):** Private image registry services in HA+ active-standby mode with high stability requirements.

3.2 Overall Architecture

All servers are divided into three major components: master services, node services, and storage services. Master servers must be on the same network segment to implement load balancing, while storage servers should also be on the same network segment with 10-gigabit networking for efficiency and stability. Node servers have lower network dependency, requiring only routability between services. All cluster servers can use CentOS 7.4 as the operating system, with yum repositories for CentOS and EPEL, consistent timezone settings, and time synchronization across nodes.

The master architecture comprises the etcd cluster and Kubernetes master service components. Kubernetes uses etcd to store all data, making it one of the most critical components. For security, etcd requires both server certificates and client certificates.

Master nodes primarily contain three components: apiserver, scheduler, and controller-manager. The apiserver provides REST API interfaces for cluster management, including authentication, authorization, data validation, and cluster state changes. Only the API Server directly operates etcd; other modules query or modify data through the API Server, which serves as the hub for data interaction and communication between modules. The scheduler is responsible for assigning and scheduling Pods to node nodes within the cluster, listening to

the kube-apiserver and querying unassigned Pods to allocate nodes according to scheduling policies. The controller-manager consists of a series of controllers that monitor the entire cluster state through the apiserver and ensure the cluster maintains its desired state.

High availability for master nodes primarily involves achieving high availability of the apiserver component through haproxy load balancing configuration. Node servers mainly include docker services and kube-node components.

3.2.1 Docker Starting from version 1.13, Docker sets the default policy of the FORWARD chain in the iptables filter table to DROP, causing ping failures to Pod IPs on other nodes. Therefore, a default allow rule must be added to the FORWARD chain: `iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT`. The Docker image registry uses China Search' s internal private registry at https://reg.docker.chinaso365.com, which can be configured in `/etc/docker/daemon.json` to replace Docker' s default registry.

3.2.2 Kube-node Kube-node is the node in the cluster that hosts applications. As a prerequisite, the kube-master node must be deployed first (because user role binding and approval of kubelet TLS certificate requests are required). The following components need deployment: Docker for running containers, Calico for container network configuration, kubelet as the primary component on kube-node, and kube-proxy for publishing application services and load balancing.

3.3 Network Architecture

Kubernetes configures its network through CNI drivers that invoke various network plugins. Common CNI plugins include flannel, calico, and weave, each with distinct advantages that they continue to learn from one another. For instance, when all node nodes are on a layer-2 network, flannel provides a hostgw implementation to avoid the UDP encapsulation overhead of vxlan implementation, while calico offers an IPinIP option for L3 Fabric utilizing GRE tunnel encapsulation. Based on our business requirements, we adopted Calico networking.

Calico is a pure layer-3 data center network solution based on BGP (no Overlay required) that integrates well with IaaS and container platforms such as OpenStack, Kubernetes, AWS, and GCE. Calico implements an efficient vRouter on each compute node using the Linux Kernel for data forwarding, with each vRouter propagating routing information for workloads running on it throughout the entire Calico network via the BGP protocol—small-scale deployments can interconnect directly, while large-scale deployments can use designated BGP route reflectors. This ensures all workload-to-workload data traffic is interconnected via IP routing. Calico node networking can directly utilize data center network structures (whether L2 or L3) without additional NAT, tunnels, or

Overlay Networks. Additionally, Calico provides rich and flexible network Policies based on iptables, ensuring multi-tenant isolation, security groups, and other reachability restrictions through ACLs on each node.

3.4 Storage Architecture

GlusterFS is the optimal storage solution for Kubernetes private deployment schemes. It is an open-source distributed file system with powerful horizontal scaling capabilities, supporting petabytes of storage capacity and thousands of clients through expansion. GlusterFS aggregates physically distributed storage resources via TCP/IP or InfiniBand RDMA networks, using a single global namespace to manage data. Based on a stackable user-space design, GlusterFS delivers excellent performance for various data workloads.

We recommend using dispersed volumes with specified redundancy block ratios (3:1) to achieve high availability while providing optimal cost-performance IO throughput and network performance. Performance testing in a 10-gigabit network environment shows approximately 30% of local disk IO performance, while in a gigabit network environment, performance is nearly double that of Alibaba Cloud NAS services. Test data is as follows:

- 10-gigabit environment: 655,360 bytes (655 kB) copied, 0.00240202 s, 273 MB/s
- Gigabit environment: 655,360 bytes (655 kB) copied, 0.0103234 s, 63.5 MB/s
- Local disk: 655,360 bytes (655 kB) copied, 0.000753129 s, 870 MB/s
- Alibaba Cloud NFS: 655,360 bytes (655 kB) copied, 0.0177044 s, 37.0 MB/s

3.5 Technical Implementation Details

3.5.1 Cluster Deployment **Deployment method:** Ansible unified deployment management tool enables fully automated installation of etcd services, Docker services, master nodes, node nodes, cluster networking, and certificates.

Cluster management: Ansible unified deployment management tool enables node addition/deletion, master node addition/replacement, etcd node addition/replacement, cluster upgrades, and backup recovery.

3.5.2 Main Software and Versions

- Docker: 18.06.1-ce
- Kubernetes: 1.13
- Calico: 3.2.24
- Coredns: 3.4.1
- GlusterFS: 1.2.6
- Harbor: 4.1.5

3.5.3 Iptables/IPVS Because Calico networking and kube-proxy extensively use iptables rules, all iptables policy rules must be cleared before installation. Firewall solutions like CentOS' s firewalld, which are based on iptables, should be uninstalled directly to avoid unnecessary conflicts.

The kube-proxy component monitors changes to services and endpoints in the API server, providing dynamic load balancing for services within the Kubernetes cluster. Before Kubernetes v1.10, this was primarily implemented through iptables—a stable and recommended approach—but generated too many iptables rules when services were numerous, causing significant performance issues at scale. The IPVS high-performance load balancing mode, GA in v1.11, uses incremental updates and can maintain connections during service updates.

3.5.4 Add-ons **DNS** is the first component that needs deployment in a Kubernetes cluster, with other pods using it for domain name resolution services. It primarily resolves cluster service names (SVC) and Pod hostnames. Currently, Kubernetes v1.9+ offers two options: kube-dns and coredns, with either deployable.

Ingress serves as the entry point for external access to the Kubernetes cluster, forwarding user URL requests to different services. Ingress functions like an nginx reverse proxy server, with its rule definitions constituting URL routing information. Implementation requires deploying an Ingress controller (such as traefik or ingress-nginx), which listens to ingress and service changes through the apiserver and configures load balancing according to rules to provide access entrypoints, serving a service discovery function.

Heapster monitors entire cluster resources through a three-step process: first, the cAdvisor built into kubelet collects container resource usage on its node; second, heapster collects node and container resource usage from the APIs provided by kubelet; finally, heapster persists data storage into influxdb.

The **EFG plugin** is a logging solution for Kubernetes projects, comprising three components: Elasticsearch for log storage and search, Fluentd for sending Kubernetes cluster logs to Elasticsearch, and Grafana for visual interface viewing and retrieval of data stored in Elasticsearch.

3.5.6 Cluster Upgrades Cluster upgrades carry certain risks, requiring etcd data backup before upgrading. **Fast upgrades** refer to upgrading only the Kubernetes version, commonly used for bug fixes and important feature releases. Fast upgrades can be implemented smoothly without business interruption. **Other upgrades** involve upgrading Kubernetes components including etcd and Docker versions, requiring detailed upgrade plans and consideration of potential business interruption.

4. China Search Kubernetes Application Platform

The China Search Kubernetes Application Platform is a containerized application platform for managing data center host clusters, designed to improve server performance utilization, enable efficient deployment and elastic computing, provide high redundancy, and deliver a customized PaaS service platform. The platform' s goal is to make deploying containerized applications simple and efficient.

4.1 Unified Resource Allocation and Real-time Statistical Analysis

Building on efficient Kubernetes cluster management, we have implemented statistical analysis of overall application status. Unlike traditional physical-layer monitoring, the application platform acquires, analyzes, and displays resource allocation, service health status, and application saturation at the application layer. Through cluster node status monitoring and management, administrators can view cluster server operation status in real-time, helping them understand the overall situation and promptly identify and troubleshoot potential faults.

4.2 Unified Project Management

The Kubernetes Application Platform adopts a project-based resource and application management model that aligns with conventional operations management thinking. Users apply for permissions on the platform by project to obtain resources required for services, including CPU usage quotas, memory usage quotas, and storage usage quotas. Projects are isolated from each other, with support for private image registry authentication, kube-api interface call authentication, and other access controls.

The project management model corresponds to Kubernetes namespaces, which are a method for partitioning cluster resources among multiple users (through resource quotas) and intended for environments where multiple users are distributed across multiple teams or projects. Namespaces provide name scoping where resource names must be unique within a namespace but not across namespaces. In Kubernetes, objects in the same namespace have the same access control policies by default.

The platform supports project member management, defining multiple roles within project management to supplement and extend Kubernetes' s application management permission mechanisms on the namespace foundation. **Project administrators** have permissions to apply for and modify project resources. **Project leaders** have permissions to modify project application information. **Project members** have permissions to access and control project applications.

4.3 Application Service Management

The application service management module uniformly manages all applications under a project, serving as the entry point for application configuration and ac-

cess. It includes creating application services, viewing application status, and obtaining application service information. Platform application services correspond to POD services scheduled in Kubernetes—groups of one or more containerized instances (Docker containers) with shared storage/network and specifications for how to run the containers. Pod contents are always co-located and co-scheduled, running in a shared context. Pods model an application-specific "logical host" containing one or more relatively tightly coupled application containers.

4.4 Support for Configuration Management, Task Management, Monitoring, and Log Analysis

Through the platform' s application management functions, we can perform standardized rapid configuration and deployment of applications running on Kubernetes. Application logic is automatically configured through application orchestration, configmap definitions or updates through configuration management, and application services can be started, upgraded, rolled back, or stopped through task management. Application monitoring enables real-time invocation to view application monitoring metrics and operation logs.

4.5 Support for Full Application Log Collection

In addition to Docker operation logs collected by default in Kubernetes, the platform also supports user-defined application log collection—log files generated by applications. Through the platform' s application log collection module, log collection services can be configured and assigned to projects, automatically associating with project storage services for log upload. It supports multi-log format parsing, classification index configuration, custom index formats, and custom log retention periods.

4.6 Support for Full Container Terminal Operations

The platform supports accessing each application container through the Kubernetes API, enabling command-line operations via web terminals for intuitive application runtime control. Container terminal access is controlled through Kubernetes API access controls and application platform permission controls, ensuring both high security and operational convenience.

4.7 Support for Stateful Services

Stateful services are a Kubernetes application deployment mode relative to stateless services. The platform supports both stateless and stateful application modes. Using the stateful application deployment mode, each POD in an application has independently allocated and fixed SVCIP addresses and storage space. If containers restart or drift, their used resources remain unchanged, making virtualization closer to physical layer devices. Through stateful application mode,

clustered services such as redis, kafka, and MongoDB can be implemented in Kubernetes.

4.8 Support for Application Backup and Replication

The platform provides application backup and replication functions to quickly address repetitive deployment or multi-environment deployment needs. Through application replication, test environments, pre-release environments, and production environments can be quickly synchronized, or component services with identical application requirements can be rapidly deployed.

4.9 External Access Domain Name Configuration Management

The platform enables external access to Kubernetes applications through domain name configuration, implementing Ingress configuration by calling Kubernetes interfaces to provide application external service egress points. It supports custom domain names to access internal services, with support for custom domain configuration, path configuration, and both HTTP and TCP modes.

4.10 Third-Party Interface SDK

Application services running under the Kubernetes platform sometimes require inter-service calls or state control across different applications. Based on container isolation mechanisms, we implement remote calls through APIs and have encapsulated a set of third-party interface services in the Kubernetes application platform to meet internal project application inter-access needs. The platform provides a unified client SDK for Java and Python projects to satisfy such requirements. The call process is as follows: Container a calls the SDK to execute commands on container b. The SDK implementation steps are: (1) Container a calls the SDK with request parameters including the project' s private access key and command line; (2) The SDK calls the Project token interface for authentication, returning a token on success or an error code on failure; (3) The SDK carries the token to request the Kubernetes API and invoke the operation command interface; (4) The Kubernetes API sequentially executes operation commands on container b and asynchronously returns results to the SDK; (5) The SDK asynchronously returns execution results to the application on container a.

(Author Affiliation: China Search Information Technology Co., Ltd.)

Note: Figure translations are in progress. See original paper for figures.