Cloud Container Engine

Service Overview

Issue 01

Date 2022-08-10

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Cloud Native 2.0 and Huawei Cloud

As one of the earliest adopters of the container technology, Huawei has implemented the technology in multiple internal products since 2013 and started to widely use Kubernetes in 2014. In this course, Huawei has accumulated rich practical experience and provides fully vetted, full-stack container services for enterprise users to migrate applications to the cloud and succeed in the Cloud Native 2.0 era.

Now on Huawei Cloud, you can use standardized, easy-to-deploy cloud native infrastructure services to run your applications.

Cloud Native 2.0

Cloud Native Development

Cloud native technologies, such as container, microservice, and dynamic orchestration, are booming and have become an important driving force for service innovation. Many enterprises in industries such as finance, manufacturing, and Internet have applied these technologies to their core services. Use cases in more service scenarios are on the go, and the industry ecosystem is expanding.

From "On Cloud" to "In Cloud"

New enterprise applications are built on cloud native technologies. Applications, data, and AI are managed in the cloud throughout their lifecycle. Existing applications are organically coordinated with new ones.

New Cloud Native Enterprises

Cloud Native 2.0 is a new phase for intelligent upgrade of enterprises. Huawei Cloud is ready to provide resources you will need for this upgrade that features efficient resource use, agile applications, business intelligence, and secure, trustworthy services.

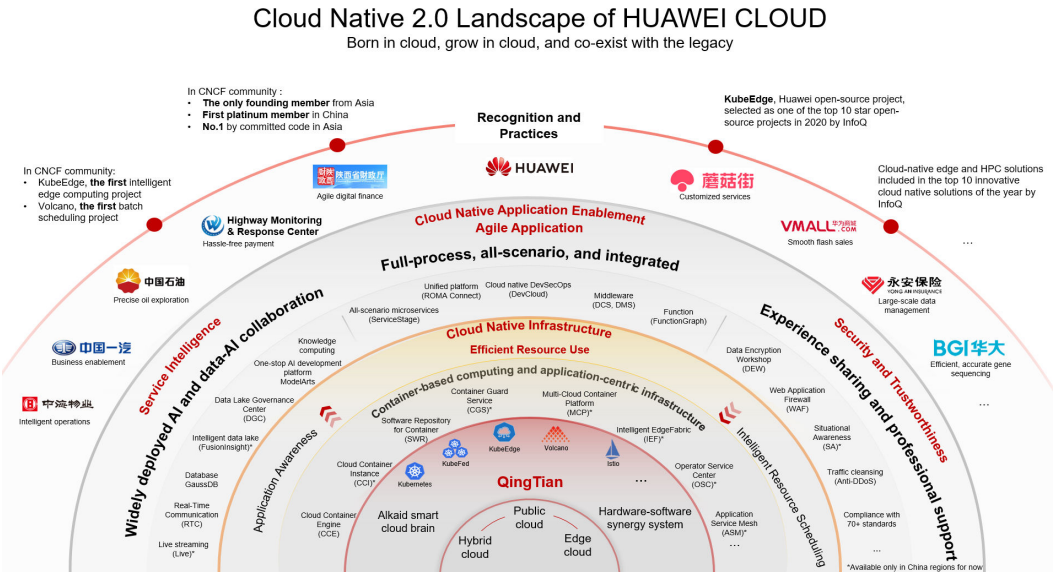
Cloud Native Landscape of Huawei Cloud

Huawei Cloud is deploying cloud native in infrastructure services, making them application-centric.

These infrastructure services include Cloud Container Engine (CCE), Software Repository for Container (SWR), Intelligent EdgeFabric (IEF), and Application

Orchestration Service (AOS). Based on these services, Huawei Cloud develops four cloud native solutions (bare-metal computing, high-performance computing, hybrid cloud, and edge computing) for building high-performance infrastructure, distributed service architecture, and comprehensive cloud-native application ecosystem.

Figure 1-1 Huawei Cloud offerings for Cloud Native 2.0



Cloud Native Infrastructure

Huawei Cloud provides customers with cloud native infrastructure services to help customers redefine their infrastructure, enable ubiquitous applications, and refactor application architecture. With these services, applications can run on a shared base and collaborate with each other across clouds and between clouds and edges to accelerate service innovation.

- **Cloud Container Engine (CCE)** allows you to create highly scalable, high-performance, enterprise-class Kubernetes clusters to run containers. CCE provides full-stack container services, including cluster and application lifecycle management, service mesh, Helm charts, add-ons, and scheduling. With CCE, you can easily deploy, manage, and scale containerized applications on Huawei Cloud.
- **Software Repository for Container (SWR)** hosts container images that can be used to quickly deploy containerized applications. It provides easy, secure, and reliable management over container images throughout their lifecycle.
- **Container Guard Service (CGS)** scans vulnerabilities and configurations in images, helping enterprises detect the container environment, which cannot be found by the traditional security software. CGS also delivers functions such as process whitelist configuration, read-only file protection, and container escape detection to minimize the security risks for a running container.
- **Intelligent EdgeFabric (IEF)** extends cloud applications to edge nodes and associates edge and cloud data, meeting customer requirements for remote control, data processing, analysis, decision-making, and intelligence of edge computing resources. IEF also provides unified on-cloud O&M capabilities,

such as device/application monitoring and log collection, to achieve edge-cloud synergy.

- **Multi-Cloud Container Platform (MCP)** is developed on Huawei Cloud container technologies and community-advanced cluster federation. It can centrally manage multiple clusters across clouds and allows uniform deployment and traffic distribution of multi-cluster application. In addition to multi-cloud disaster recovery, MCP can also separate services and data, development and production, and computing and services.

Cloud Native Application Enablement

Huawei Cloud enables customers with full-stack cloud native capabilities to support agile applications, business intelligence, secure and trustworthy services, and continuous evolution.

Agile Application

- **DevCloud** is a one-stop, cloud-based DevOps platform built with Huawei's practices of nearly three decades in R&D, together with its cutting-edge R&D ideas, and state-of-the-art R&D tools. These out-of-the-box cloud services enable you to manage projects, host code, run pipelines, check code, and build, deploy, test, and release your applications in the cloud anytime, anywhere.
- **ServiceStage** is an application and microservice management platform that facilitates application deployment, monitoring, O&M, and governance. ServiceStage provides a full-stack solution for enterprises to develop microservice, mobile, and web applications. This solution helps enterprises easily migrate various applications onto the cloud, allowing enterprises to focus on service innovation for digital transformation.
- **ROMA Connect** is a full-stack application and data integration platform designed for diverse service scenarios. ROMA Connect provides lightweight message, data, API, device, and model integration for cloud and on-premises applications across regions to simplify enterprise cloudification, helping enterprises achieve digital transformation.
- **Distributed Message Service (DMS) for Kafka** is a message queuing service based on the open-source Apache Kafka. It provides Kafka premium instances with isolated computing, storage, and bandwidth resources. DMS for Kafka allows you to apply resources and configure topics, partitions, and replicas based on service requirements. It can be used out of the box and frees you from deployment and O&M so that you can focus on the agile development of your applications.
- **FunctionGraph** hosts and computes event-driven functions. All you need to do is write your code and set conditions.

Service Intelligence

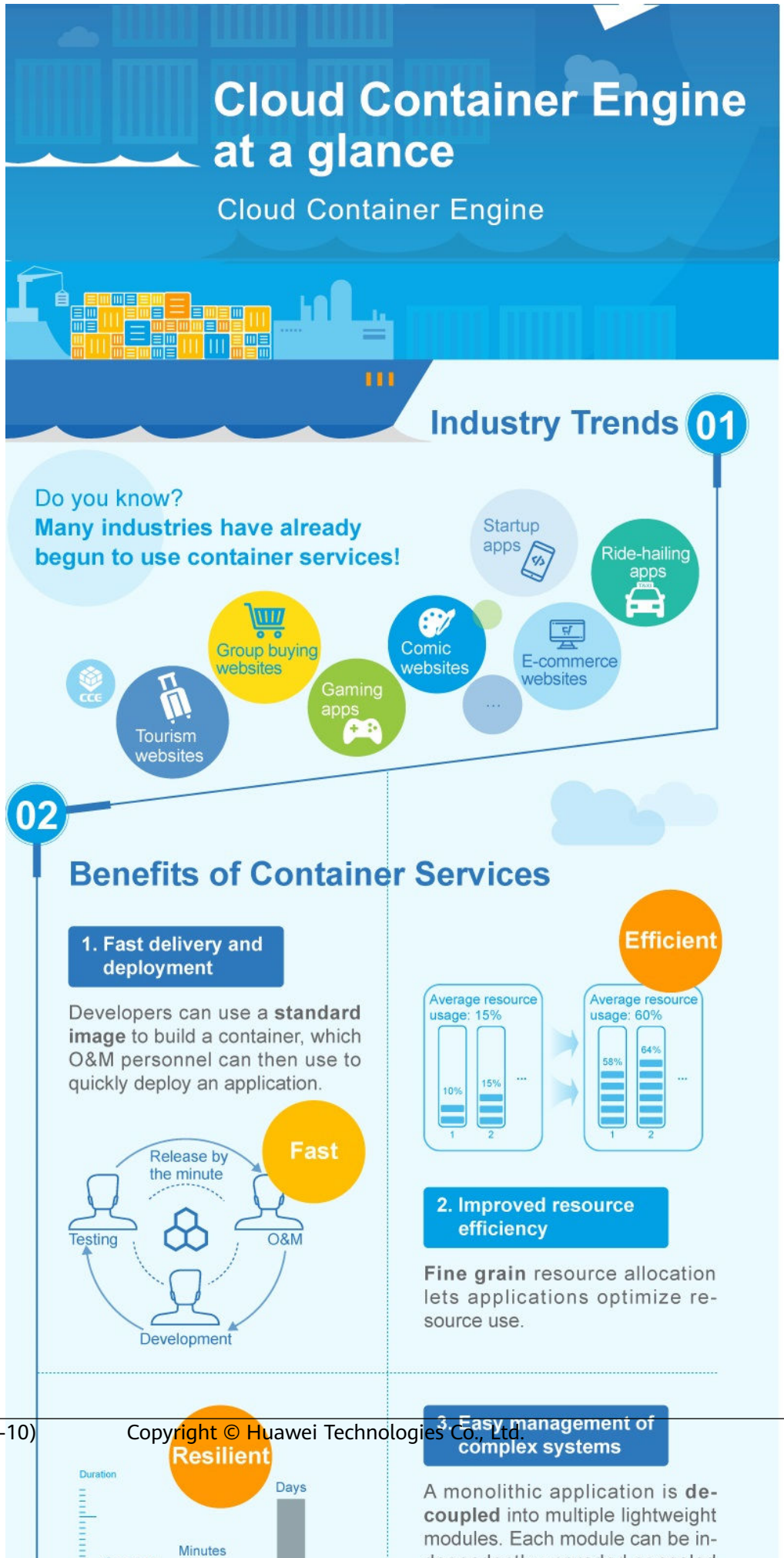
- **ModelArts** is a one-stop AI development platform. For machine learning and deep learning, it supports data preprocessing, semi-automated data labeling, distributed training, automated model building, and on-demand deployment of device-edge-cloud models. ModelArts can help AI developers build models quickly and manage the lifecycles of AI workflows.
- **GaussDB(for MySQL)** is a next-generation, enterprise-class distributed database service that is fully compatible with MySQL. It uses a decoupled

compute-storage architecture and data functions virtualization (DFV) storage that auto-scales up to 128 TB per DB instance. There is no need to deal with sharding and there is virtually no risk of data loss. It combines the high availability and performance of commercial databases with the cost-effectiveness of open-source databases.

Security and Trustworthiness

- **Data Security Center (DSC)** is a next-generation cloud data protection platform that protects your assets with functions such as risk classification, sensitive data identification, watermark source tracing, and static data masking. DSC monitors data security and gives you a comprehensive view of your data security in the cloud.
- **Host Security Service (HSS)** helps you identify and manage the assets on your servers, eliminate risks, and defend against intrusions and web page tampering. There are also advanced protection and security operations functions available to help you easily detect and handle threats.
- **Anti-DDoS** protects Huawei Cloud resources, including Elastic Cloud Servers (ECSs), load balancers, and Bare Metal Servers (BMSs), against layer-4 to layer-7 distributed denial of service (DDoS) attacks and sends alarms immediately when detecting an attack. In addition, Anti-DDoS improves the bandwidth utilization to further safeguard user services.

2 CCE Infographic



3 What Is Cloud Container Engine?

Cloud Container Engine (CCE) is a highly scalable, enterprise-class hosted Kubernetes service for you to run containers and applications. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

Why CCE?

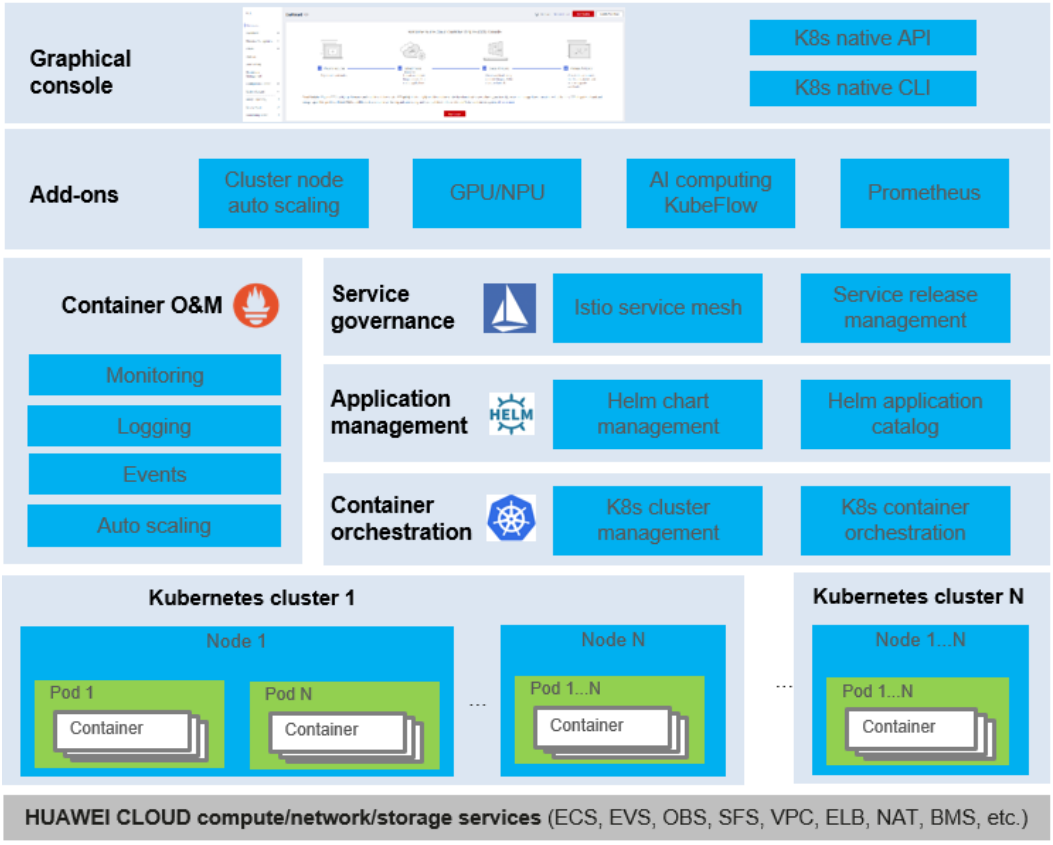
CCE is deeply integrated with cloud services, including high-performance compute (ECS//BMS), networking (VPC/EIP/ELB), and storage (EVS/OBS/SFS) services. It supports heterogeneous computing architectures such as GPU, NPU, and Arm. Supporting multi-AZ and multi-region disaster recovery, CCE ensures high availability of **Kubernetes** clusters.

Huawei Cloud is one of world's first Kubernetes Certified Service Providers (KCSPs) and China's first participant in the Kubernetes community. It has long been contributing to open-source container communities and taking lead in the container ecosystem. Huawei Cloud is also a founder and platinum member of Cloud Native Computing Foundation (CNCF). CCE is one of the world's container services to first pass the Certified Kubernetes Conformance Program.

For more information, see **Product Advantages** and **Application Scenarios**.

Product Architecture

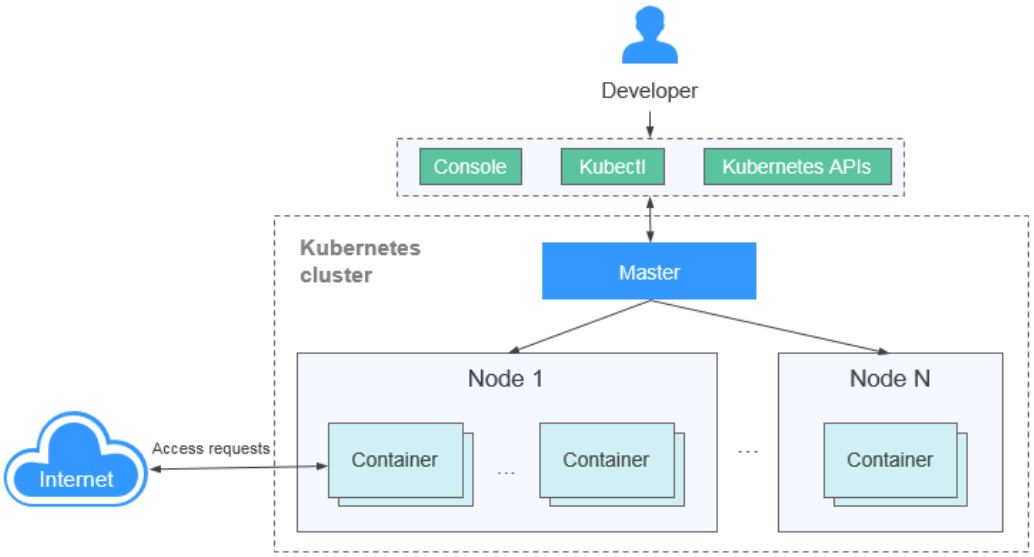
Figure 3-1 CCE architecture



Accessing CCE

You can use CCE via the CCE console, kubectl, or Kubernetes APIs. **Figure 3-2** shows the process.

Figure 3-2 Accessing CCE



CCE Learning Path

You can click **here** to learn about the fundamentals about CCE so that you can use CCE and perform O&M with ease.

4 CCE Turbo Cluster

To deploy containers at a large scale, you need a more powerful container cluster type that delivers higher performance, faster scaling, and more efficient scheduling.

With CCE Turbo clusters, you can enjoy accelerated computing, networking, and scheduling to drive application innovations.

Cluster Advantages

- **Accelerated computing**

  On an infrastructure featuring software-hardware synergy, computing consumes fewer resources but delivers better performance.

- **Accelerated networking**

  The Cloud Native Network 2.0 model eliminates performance loss by flattening the network and enables smoother communication between applications.

- **Accelerated scheduling**

  Intelligent hybrid scheduling simplifies application and resource management.

- **Security isolation**

  The secure container engine provides VM-level security isolation for applications.

Comparison Between CCE Turbo Clusters and CCE Clusters

The following table lists the differences between CCE Turbo clusters and CCE clusters:

Table 4-1 Cluster types

Dimension	Sub-dimension	CCE Turbo Cluster	CCE Cluster
Cluster	Positioning	Next-generation container cluster for Cloud Native 2.0 with accelerated computing, networking, and scheduling	Standard cluster for common commercial use
	Node type	Hybrid deployment of VMs and bare-metal servers	Hybrid deployment of VMs and bare-metal servers
Networking	Model	**Cloud Native Network 2.0**: applies to large-scale and high-performance scenarios. Networking scale: 2000 nodes	**Cloud-native network 1.0** for scenarios that do not require high performance or involve large-scale deployment. • Tunnel network model • VPC network model
	Performance	The VPC network and container network are flattened into one, achieving zero performance loss.	The VPC network is overlaid with the container network, causing certain performance loss.
	Container network isolation	Pods can be directly associated with security groups to configure isolation policies for resources inside and outside a cluster.	• Tunnel network model: Network isolation policies are supported for intra-cluster communication (by configuring network policies). • VPC network model: Isolation is not supported.
Security	Isolation	• Physical machine: Kata containers, providing VM-level isolation. • VM: Common containers are deployed.	Common containers are deployed and isolated by Cgroups.

How to Buy

Learn how to **buy and configure a CCE Turbo cluster**.

5 Function Overview

CCE enables you to manage multiple resource objects, including clusters, nodes, node pools, workloads, add-ons, and charts. Other advanced features include affinity-based scheduling, auto scaling, container networking and storage, permission management, and system steward.

Cluster

CCE is a hosted Kubernetes service that simplifies the deployment and management of containerized applications. With CCE, you can easily create Kubernetes clusters, deploy containerized applications, and manage and maintain them.

- **One-stop deployment and O&M**: You can create a Kubernetes container cluster in just a few clicks, without needing to set up Docker or Kubernetes environments. Automatic deployment and O&M of containerized applications can be performed all in one place throughout the application lifecycle.
- **Multiple cluster types**: CCE works closely with heterogeneous infrastructure services, including high-performance Elastic Cloud Server (ECS), Bare Metal Server (BMS), and GPU-Acceleration Cloud Server (GACS). You can choose the cluster type best suited to your needs and quickly create clusters while CCE handles all the complexity of cluster management.

Table 5-1 Cluster management functions

Function	Description
CCE Turbo cluster	CCE Turbo cluster supports hybrid deployment of VMs and bare metal servers (BMSs) to deliver high performance based on infrastructure services.
CCE cluster	CCE cluster supports hybrid deployment of VMs and bare-metal servers (BMSs), and heterogeneous nodes such as GPU and NPU nodes. You can run your containers in a secure and stable container runtime environment based on a high-performance network model.

Function	Description
Kunpeng cluster	Kunpeng cluster (with Arm-based instruction sets) enables containers to run on cloud servers that use Arm architecture and Kunpeng processors. Kunpeng-accelerated cloud servers are easy to deploy and provide comparable scaling and scheduling performance as x86-based cloud servers at only a fraction of what x86-based cloud servers would cost.
Cluster auto scaling	CCE automatically scales a cluster by adding or releasing worker nodes. For example, when workloads cannot be scheduled into the cluster due to insufficient cluster resources, scale-out will be automatically triggered.
Cluster upgrade	You can upgrade your cluster to the latest Kubernetes version or a bug-fixed version on the CCE console.
Cluster monitoring	You can view the monitoring metrics to learn the resource usage of each cluster master node in real time, and receive and respond to alarms in a timely manner.

Node

A container cluster consists of a set of worker machines, called nodes, that run containerized applications. A node can be a virtual machine (VM) or a physical machine (PM), depending on your service requirements. The components on a node include kubelet, container runtime, and kube-proxy. CCE uses high-performance Elastic Cloud Servers (ECSs) or Bare Metal Servers (BMSs) as nodes to build highly available Kubernetes clusters.

Table 5-2 Node management functions

Function	Description
Adding a node	You can add nodes to a CCE cluster by purchasing or accepting nodes. Accepting nodes is to add purchased ECSs to a CCE cluster. Heterogeneous nodes, such as VMs, bare metal servers, GPU-enabled nodes, and NPU-enabled nodes, can be purchased and added.
Monitoring a node	CCE uses the Cloud Eye service to monitor nodes. Each node corresponds to an ECS.
Resetting a node	When a node in a CCE cluster is reset, services running on the node will also be deleted. Exercise caution when performing this operation. This function is supported for clusters of v1.13 and later.
Deleting a node	When a node in a CCE cluster is deleted, services running on the node will also be deleted. Exercise caution when performing this operation.

Node Pool

You can create a node pool for a CCE cluster to quickly create, manage, and destroy nodes without affecting the entire cluster. All nodes in a custom node pool have identical parameters and node type. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool.

Table 5-3 Node pool management functions

Function	Description
Creating a node pool	You can create and view node pools.
Managing a node pool	You can modify the parameters of a node pool, delete or clone a node pool, and migrate nodes.

Workload

A workload is an application running on Kubernetes. No matter how many components are there in your workload, you can run it in a group of Kubernetes pods. A workload is an abstract model of a group of pods in Kubernetes. Workloads classified in Kubernetes include Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

CCE provides Kubernetes-native container deployment and management and supports lifecycle management of container workloads, including creation, configuration, monitoring, auto scaling, upgrade, uninstall, service discovery, and load balancing.

Table 5-4 Workload management functions

Function	Description
Setting container specifications	CCE allows you to set resource limits for added containers during workload creation. You can apply for and limit the CPU and memory quotas used by each pod in the workload, and set whether to use GPU and Ascend 310 for each pod.
Setting container lifecycle hooks	CCE provides callback functions (hooks) for the lifecycle management of containerized applications. For example, if you want a container to perform a certain operation before stopping, you can register a hook.

Function	Description
Setting container startup command	When creating a workload or job, you can use an image to specify the processes running in the container. The default command in your image is run. To run a specific command or use new values, configure the following settings: • **Working directory**: working directory of the command. If the working directory is not specified in the image or on the console, the default value is **/**. • **Command**: command that controls the running of an image. • **Args**: parameters passed to the running command.
Setting container health check	CCE can regularly check the health status of containers during container running. If the health check function is not configured, a pod cannot detect service exceptions or automatically restart the service to restore it. This will result in a situation where the pod status is normal but the service in the pod is abnormal. CCE provides the following health check probes: 1. **Liveness probe** checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness probe of a container fails, the cluster restarts the container. If the liveness probe is performed successfully, no operation is executed. 2. **Readiness probe** checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some applications to start before they can provide services. This is because they need to load disk data or rely on the startup of an external module. In this case, application processes are running, but the applications are not ready to provide services. This is where the readiness probe enters. If the container readiness probe fails, the cluster masks all requests sent to the container. If the container readiness probe is performed successfully, the container can be accessed.
Setting environment variables	An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.
Collecting logs	CCE allows you to configure policies for collecting, managing, and analyzing workload logs periodically. These policies can prevent logs from being over-sized.

Affinity and Anti-Affinity Scheduling

You can constrain which AZs and nodes your workloads are eligible or forbidden to be scheduled on. You can also define rules to describe which workloads will or will not be co-located with your workloads. Affinity scheduling allows workloads to be physically closer to user location and makes routing paths between containers as short as possible, which in turn reduces network overhead. Anti-affinity scheduling prevents a single point of failure by banning co-location of instances belonging to the same workload. It also prevents interfering workloads from affecting each other by not allowing them to run on the same node or AZ.

Table 5-5 Scheduling policies

Function	Description
Custom scheduling policy	A custom scheduling policy allows you to customize node affinity, workload affinity, or workload anti-affinity. A combination of custom scheduling policies can better meet your service requirements.
Simple scheduling policy	A simple scheduling policy provides basic, one-for-all scheduling functions. It allows you to configure affinity between workloads and AZs, between workloads and nodes, or between workloads.

Networking

By deeply integrating the Kubernetes networking capabilities with VPC, CCE provides stable and high-performance networking for mutual access of workloads in complex scenarios.

Table 5-6 Networking-related functions

Function	Description
Service	Services allow you to access a single or multiple containerized applications. Each Service has a fixed IP address and port during its lifecycle and targets one or more backend pods. In this way, frontend clients do not need to keep track of these pods, allowing pods to be added or reduced without worrying IP address changes. CCE supports the following types of Services: • ClusterIP: The Service is only reachable from within the cluster. • NodePort: The Service is accessed using the private IP address or EIP of the node. • LoadBalancer: The Service is accessed using a load balancer. • DNAT: The Service is accessed using a DNAT gateway.

Function	Description
Layer-7 load balancing (ingress)	Ingresses use shared and dedicated load balancers. Compared with layer-4 load balancing, layer-7 load balancing additionally supports Uniform Resource Locator (URL) configurations and routes access traffic to Services based on URLs. Services will also act according to the URLs.
Network policy	CCE has enhanced the Kubernetes-based network policy feature, allowing network isolation in a cluster by configuring network policies. This means that a firewall can be set between pods. For example, to make a payment system accessible only to specified components for security purposes, you can configure network policies.
Network attachment definition	A network attachment definition is a type of Custom Resource Definitions (CRDs) in a cluster. It provides configuration items, such as VPC and subnet, for containers to connect to the elastic network interface (ENI). Workloads associated with network attachment definitions can connect to the ENI, so that the containers can be directly bound with the ENIs to expose Services externally.

Persistent Volume (PV)

In addition to using local disks for storage, CCE can store workload data using cloud storage services. Currently, the following types of cloud storage are supported: Elastic Volume Service (EVS), Object Storage Service (OBS), Scalable File Service (SFS), and SFS Turbo.

Table 5-7 Storage-related functions

Function	Description
Using local disks as storage volumes	You can mount the file directory of the host where a container is located to a specified container path (corresponding to hostPath in Kubernetes). Alternatively, you can leave the source path empty (corresponding to emptyDir in Kubernetes). If the source path is left empty, a temporary directory of the host will be mounted to the mount point of the container. A specified source path is used when data needs to be persistently stored on the host, while emptyDir is used when temporary storage is needed.
Using EVS disks as storage volumes	EVS disks can be attached to containers. By using EVS volumes, you can attach the remote file directory of a storage system into a container so that data in the data volume is permanently preserved. Even if the container is deleted, the data in the data volume is still stored in the storage system.

Function	Description
Using SFS file systems as storage volumes	You can create SFS volumes and mount them to specific container paths. The volumes created by the underlying SFS service can also be used. SFS volumes are suited for scenarios where data needs to be persisted and read by and written to multiple nodes. Such scenarios include media processing, content management, big data analysis, and workload analysis.
Using OBS buckets as storage volumes	You can create OBS volumes and mount them to a container path. OBS applies to scenarios such as cloud workloads, data analysis, content analysis, and hotspot objects.
Using SFS Turbo file systems as storage volumes	You can create SFS Turbo volumes and mount them to a container path. SFS Turbo file systems are fast, on-demand, and scalable, which are suitable for DevOps, containerized microservices, and enterprise office applications.
Creating snapshots and backups	CCE works with EVS to provide the snapshot function. A snapshot is a complete copy or image of EVS disk data at a certain point of time, which is of great help to data DR.

Add-ons

CCE provides multiple types of add-ons to extend cluster functions and satisfy diverse requirements.

- CCE supports open APIs and native APIs from the community.
- CCE provides kubectl-related add-on and community-native kubectl.

Ecosystem Tools

CCE works seamlessly with Helm.

Table 5-8 Kubernetes ecosystem

Function	Description
Chart management	Helm is a Kubernetes package manager that makes it simple to deploy and manage packages (also called charts). A chart is a collection of files that describe a related set of Kubernetes resources. The use of charts handles all the complexity in Kubernetes resource installation and management. In CCE, you can upload charts to deploy applications. • Uploaded charts are user-defined, which simplify workload deployment.

Auto Scaling

CCE allows you to scale your clusters and workloads both manually and automatically. Any auto scaling policies can be flexibly combined to deal with in-the-moment load spikes.

Table 5-9 Auto scaling functions

Function	Description
Workload scaling	CCE supports HPA and CustomedHPA policies. • HPA: a policy that implements horizontal scaling of pods in Kubernetes. In a CCE HPA policy, you can configure cooldown time window and scaling thresholds based on the Kubernetes HPA. • CustomedHPA: a policy that scales Deployments based on metrics (such as CPU usage and memory usage) or at a periodic interval (every day/week/month or at a specific time point). This type of policy is an enhanced auto scaling capability developed by Huawei Cloud.
Node scaling	CCE provides node scaling through the autoscaler add-on. Nodes with different specifications can be automatically added across AZs on demand.

Permissions

CCE permissions management allows you to assign permissions to IAM users and user groups under your tenant accounts. CCE combines the advantages of Identity and Access Management (IAM) and Kubernetes Role-based Access Control (RBAC) authorization to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, cluster-scoped authorization, and namespace-wide authorization.

Table 5-10 Permission-related functions

Function	Description
Cluster-level permissions	CCE cluster-level permissions are assigned based on **IAM system policies** and **custom policies**. You can use user groups to assign permissions to IAM users.
Namespace-level permissions	You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC authentication on access to APIs using IAM tokens.

6 Product Advantages

Why CCE?

CCE is a container service built on popular Docker and Kubernetes technologies and offers a wealth of features best suited to enterprises' demand for running container clusters at scale. With unique advantages in system reliability, performance, and compatibility with open-source communities, CCE can suit the particulars of enterprises interested in building container clouds.

Easy to Use

- Creating a Kubernetes cluster is as easy as a few clicks on the web user interface (WebUI). The Kubernetes cluster supports management of VM nodes or bare-metal nodes and applies to the scenario where VMs and physical machines are used together.
- Automatic deployment and O&M of containerized applications can be performed all in one place throughout the application lifecycle.
- Clusters and workloads can be resized in just a few clicks on the WebUI. Any auto scaling policies can be flexibly combined to deal with in-the-moment load spikes.
- The WebUI walks you through the steps required to upgrade Kubernetes clusters.
- Support for Helm charts offers out-of-the-box usability.

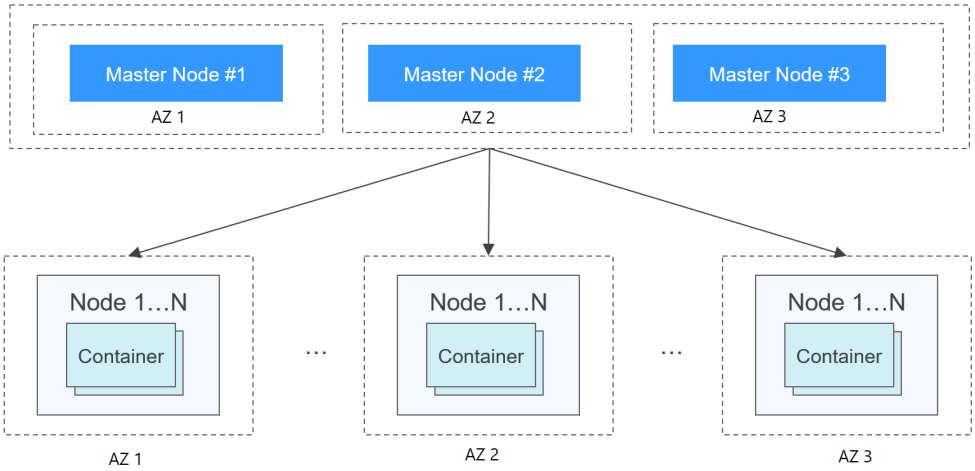
High Performance

- CCE draws on years of field experience in compute, networking, storage, and heterogeneous infrastructure. You can concurrently launch containers at scale.
- The bare-metal NUMA architecture and high-speed InfiniBand network cards yield three- to five-fold improvement in computing performance.

Highly Available and Secure

- High reliability: You can deploy three master nodes in different AZs for the cluster control plane to ensure high availability of your services. Nodes and workloads in a cluster can be load balanced across AZs to form a multi-active architecture that ensures service continuity even when one of the hosts or equipment rooms is down or an AZ is hit by natural disasters.

Figure 6-1 High-availability setup of clusters



- Secure: Clusters are private and completely controlled by users with deeply integrated IAM and Kubernetes RBAC. You can set different RBAC permissions for IAM users on the console.

Open and Compatible

- CCE is built on the open-source Docker technology that automates deployment, resource scheduling, service discovery, and dynamic scaling of containerized applications.
- CCE is built on Kubernetes and compatible with Kubernetes native APIs, kubectl (a command line interface), and Kubernetes/Docker native releases. Updates from Kubernetes and Docker communities are regularly incorporated into CCE.

Comparative Analysis of CCE and On-Premises Kubernetes Cluster Management Systems

Table 6-1 CCE clusters versus on-premises Kubernetes clusters

Area of Focus	On-Premises Kubernetes Cluster Management Systems	CCE
Ease of use	Cluster management is complex. You have to handle all the complexity in installing, operating, scaling, configuring, and monitoring Kubernetes cluster management infrastructure. Each cluster upgrade requires tremendous manual adjustment, imposing a heavy burden on O&M personnel.	**Easy to manage and use clusters** You can create and upgrade Kubernetes container clusters in just a few clicks, without needing to set up Docker or Kubernetes environments. Automatic deployment and O&M of containerized applications can be performed on the console all in one place throughout the application lifecycle. Support for Helm charts offers out-of-the-box usability. Using CCE clusters is as simple as choosing a container cluster and the jobs that you want to run in the cluster. CCE then completes cluster management so you can focus on developing containerized applications.
Scalability	You have to manually evaluate service load and cluster health before deciding to resize a cluster.	**Managed scaling service** CCE can automatically resize clusters and workloads as resource usage changes. Combined use of auto scaling policies can flexibly scale clusters and workloads to meet fluctuating demands.
Reliability	Only one master node is available in a cluster. Once the master node is down, the entire cluster as well as all the applications in the cluster will become out of service.	**High availability** If **High Availability** is set to **Yes** when you create a cluster, three master nodes will be created in the cluster, avoiding single points of failure on the cluster control plane.

Area of Focus	On-Premises Kubernetes Cluster Management Systems	CCE
Efficiency	You have to either build image repositories or revert to third-party image repositories. Images are pulled from repositories in serial.	**Rapid image deployment and continuous integration** CCE works with SoftWare Repository for Container (SWR) to support DevOps pipelines and eliminate the need to manually write Dockerfiles or Kubernetes manifests. With ContainerOps pipeline templates, you can define how to build container images, push them to repositories, and deploy them. Images are pulled from repositories in parallel.
Cost	Heavy upfront investment is required in installing, managing, and scaling cluster management infrastructure.	**Cost effective** You only pay for the infrastructure resources required to store and run applications, as well as the master nodes in the cluster.

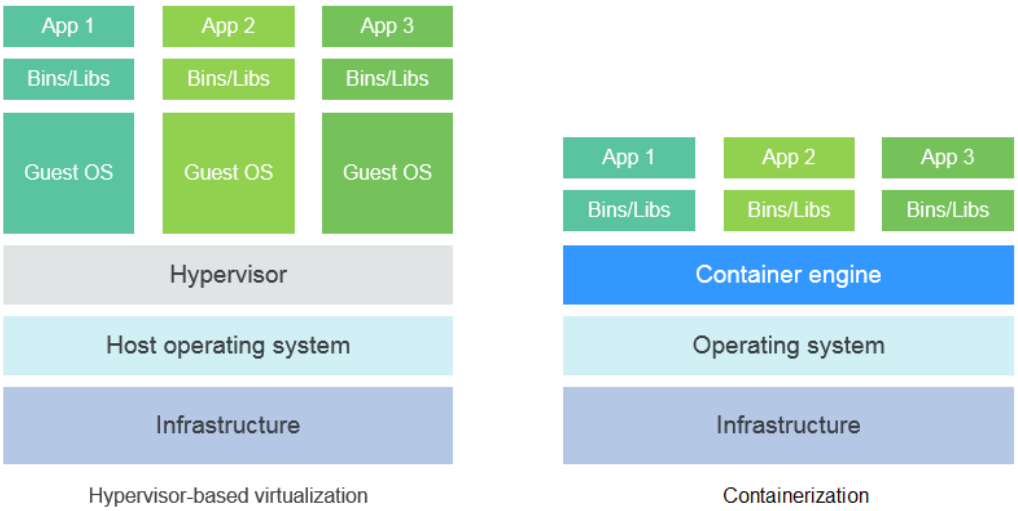
Why Containers?

Docker is written in the Go programming language designed by Google. It provides operating-system-level virtualization: software processes are isolated from each other by using Linux Control Groups (cgroups), namespaces, and Union FS technologies (for example, AUFS). Everything needed to run a software process is packed into a container. Containers are isolated from each other and from the host.

Docker has moved forward to enhance container isolation: containers have their own file systems, and they cannot see each other's processes or network interfaces. This simplifies container creation and management.

The traditional virtualization technology provides hardware-level virtualization. It creates a set of virtual machines, each with a complete operating system and application inside. Containers, on the other hand, do not have their own kernel and all call out to the same kernel of host OS. Furthermore, it is unnecessary to do any kind of virtualization the way it does with VMs. Therefore, Docker containers are smaller and faster than VMs.

Figure 6-2 Comparison between Docker containers and VMs



To sum up, Docker containers have many advantages over VMs.

Resource utilization

With no overhead for virtualizing hardware and running a complete OS, containers can outperform VMs in application execution speed, memory loss, and file storage speed.

Start speed

It takes several minutes to start an application on a VM. Docker containerized applications run directly on the host kernel and there is no need to start a complete operating system along with the applications. The startup time can be reduced to seconds or even milliseconds, greatly saving your time in development, testing, and deployment.

Consistent environment

One of the biggest problems that developers always have to deal with is the difference in the environments where they run their applications. Difference between development, testing, and production environments prevents some bugs from being discovered prior to rollout. A Docker container image includes everything needed to run an application and isolates the application from its environment. Therefore, containerized applications will always run the same across development, testing, and production environments.

Continuous delivery and deployment

For DevOps personnel, it would be ideal if applications can run anywhere after one-time creation or configuration.

Docker provides reliable and frequent container image build and deployment with quick, easy rollbacks (due to image immutability). Developers write Dockerfiles that contain all the instructions required to build container images and merge up-to-date instructions regularly into Dockerfiles, a practice known as Continuous Integration (CI). The Ops team can rapidly deploy images into production environment by letting Docker read instructions from Dockerfiles. The Ops team can even follow the Continuous Delivery/Deployment (CD) practice in which every

instruction change is automatically built, tested, and then pushed to a non-production testing environment.

The use of Dockerfiles makes the DevOps process visible to everyone in a DevOps team. In this way, the developer team can better understand both users' needs and the problems faced by the Ops team while maintaining the application. On the other hand, the Ops team can have some knowledge of the conditions that must be met to run the application. The knowledge is helpful when the Ops personnel deploy container images into production environment.

Portability

Docker ensures environmental consistency across development, testing, and production, and so Docker containers can be portable anywhere. They work uniformly, regardless of whether they run on physical machines, virtual machines, public clouds, private clouds, or even laptops. You can migrate applications from one platform to another without worrying that the environment change will cause the applications unable to work.

Application update

Docker images are composed of layers. Each layer is only stored once and different images can contain the exact same layers. This makes distribution efficient because layers that have already been transferred as part of the first image do not need to be transferred again when transferring the other image that also has these layers. To update a containerized application, you can either edit the top-most writable layer in the final image or add layers to the base image. In addition, Docker collaborates with open-source project teams to maintain a large number of high-quality official images. You can directly use them in the production environment or easily build new images based on them.

Table 6-2 Containers versus traditional VMs

Feature	Containers	VMs
Start speed	In seconds	In minutes
Disk capacity	MB	GB
Performance	Near-native performance	Weak
Per-machine capacity	Thousands of containers	Tens of VMs

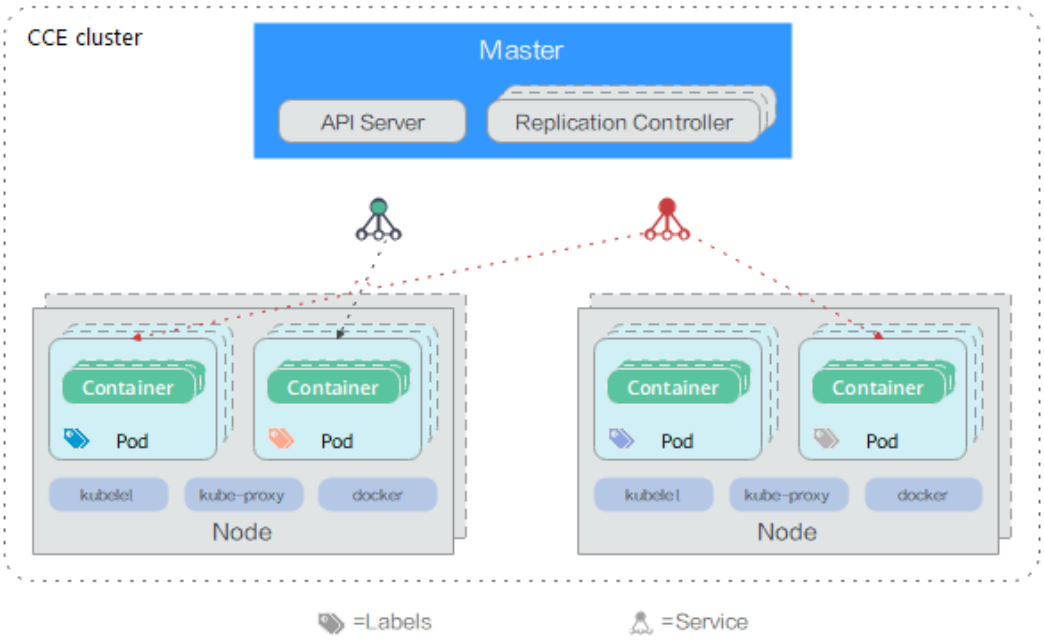
7 Application Scenarios

7.1 Infrastructure and Containerized Application Management

Application Scenario

CCE clusters support management of x86 and Arm resource pools. You can create Kubernetes clusters, deploy containerized applications, and manage and maintain the clusters.

Figure 7-1 CCE cluster



Benefits

Containerization reduces application deployment resource costs, streamlines deployment and upgrade, and achieves uninterrupted services during upgrades.

Advantages

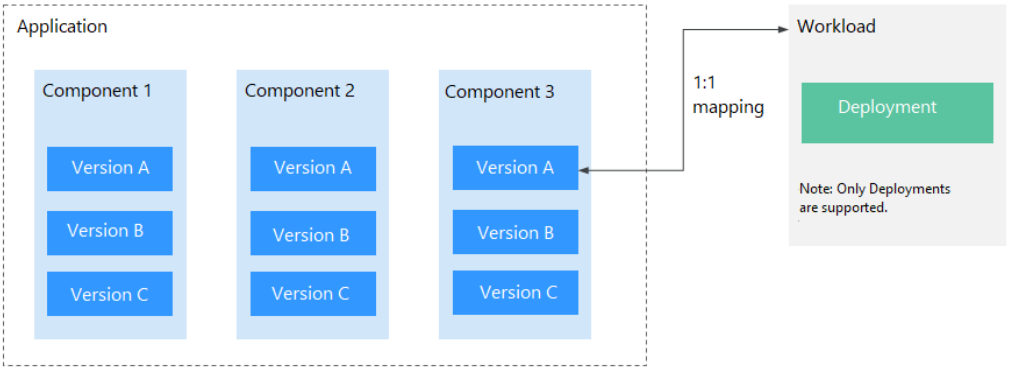
- Deployment of multiple types of workloads

  Supports Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.
- Application upgrade

  Supports in-replace upgrade, rolling upgrade by proportion or by number of pods, and upgrade rollback.
- Auto scaling

  Supports auto scaling of nodes and workloads.

Figure 7-2 Workload



7.2 Auto Scaling in Seconds

Application Scenarios

- Traffic surges brought by promotions and flash sales on online shopping apps and websites
- Fluctuating service loads of live streaming
- Increase in the number of game players that go online in certain time periods

Benefits

CCE automatically adapts the amount of computing resources to fluctuating service loads according to auto scaling policies you configured. To scale computing resources at the cluster level, CCE adds or reduces cloud servers. To scale computing resources at the workload level, CCE adds or reduces containers.

Advantages

- Flexible

  Allows multiple scaling policies and scales containers within seconds when specified conditions are met.
- Highly available

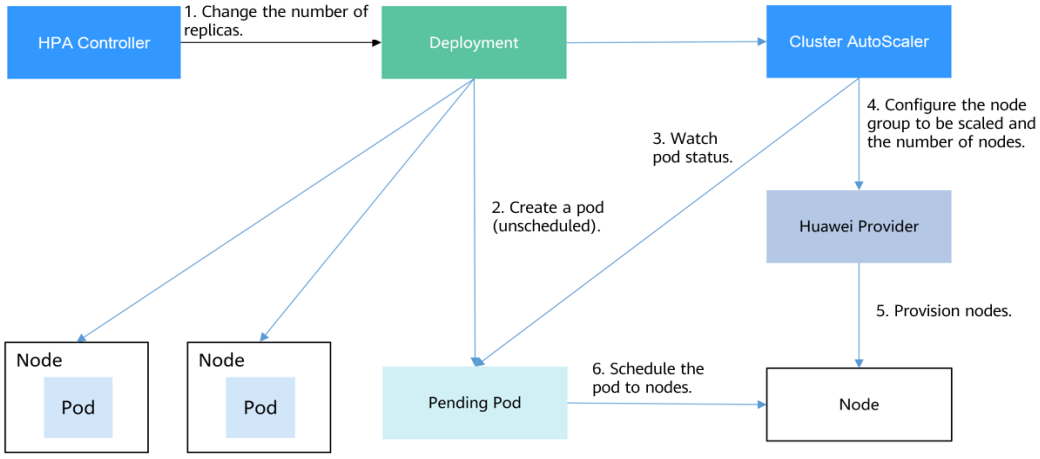
  Automatically detects the pod running status in auto scaling groups and replaces unhealthy pods with new ones.

- Lower costs
  Charges you only for the cloud servers you use.

Related Services

HPA (Horizontal Pod Autoscaling) + CA (Cluster AutoScaling)

Figure 7-3 How auto scaling works



7.3 Microservice Traffic Management

Application Scenarios

Large enterprise systems are becoming more complex, beyond what traditional system architectures can handle. A popular solution is microservice. Complex applications are divided into smaller components called microservices. Microservices are independently developed, deployed, and scaled. The combined use of microservices and containers streamlines microservice delivery while improving application reliability and scalability.

Microservices make distributed architectures possible. However, more microservices indicate more complexity in O&M, commissioning, and security management of these architectures. Developers are often troubled by writing additional code for microservice governance and integrating the code into their service systems. In this regard, CCE provides an efficient solution to free you from management workload.

Benefits

CCE is deeply integrated with Application Service Mesh (ASM), which allows you to complete grayscale release, observe your traffic, and control traffic flow without changing your code.

Advantages

- Out-of-the-box usability
  ASM can be started in just a few clicks and works seamlessly with CCE to intelligently control traffic flow.

- Intelligent routing

  HTTP/TCP connection policies and security policies can be enforced without modifying code.

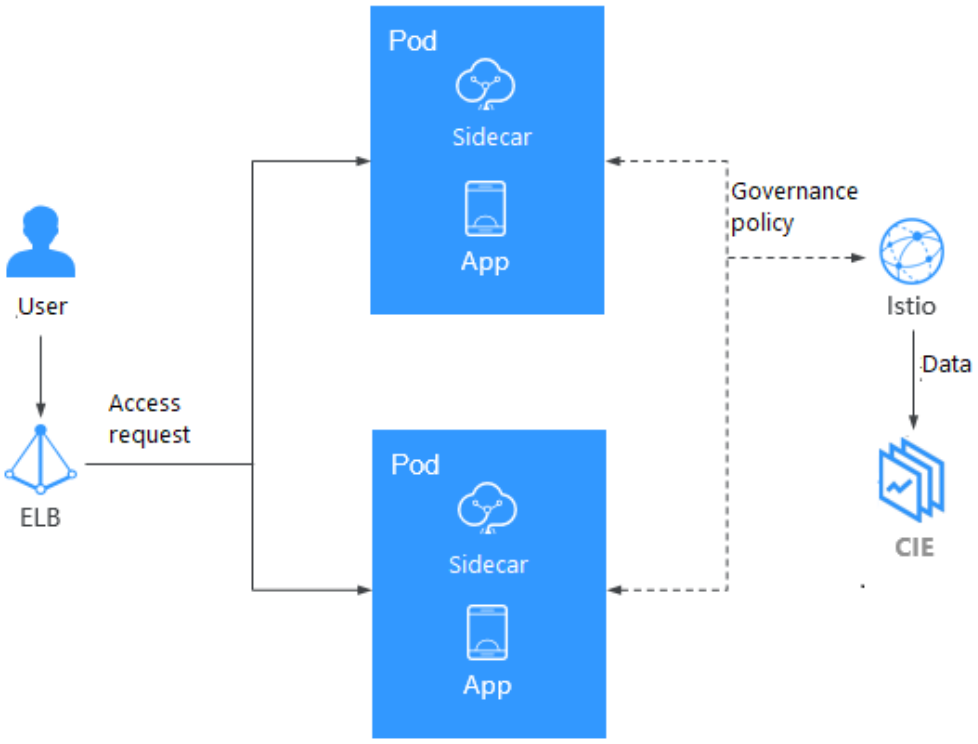
- Visibility into traffic

  Based on the monitoring data that is collected non-intrusively, ASM works closely with Application Performance Management (APM) to provide a panoramic view of your services, including real-time traffic topology, call tracing, performance monitoring, and runtime diagnosis.

Related Services

Elastic Load Balance (ELB), Application Performance Management (APM), and Application Operations Management (AOM)

Figure 7-4 Microservice governance



7.4 DevOps and CI/CD

Application Scenario

Your applications and services may receive a lot of feedback and requirements. To release new features and improve user experience, you need fast continuous integration (CI). An efficient tool to support CI is container. By deploying containers, you can streamline the process from development, testing, to release and realize continuous delivery (CD).

Benefits

CCE works with SWR to support DevOps that will automatically complete code compilation, image build, grayscale release, and deployment based on source code. Traditional CI/CD systems can be connected to containerize legacy applications.

Advantages

- Efficient process management

  Reduces scripting workload by more than 80% through streamlined process interaction.

- Flexible integration

  Provides various APIs to integrate with existing CI/CD systems for in-depth customization.

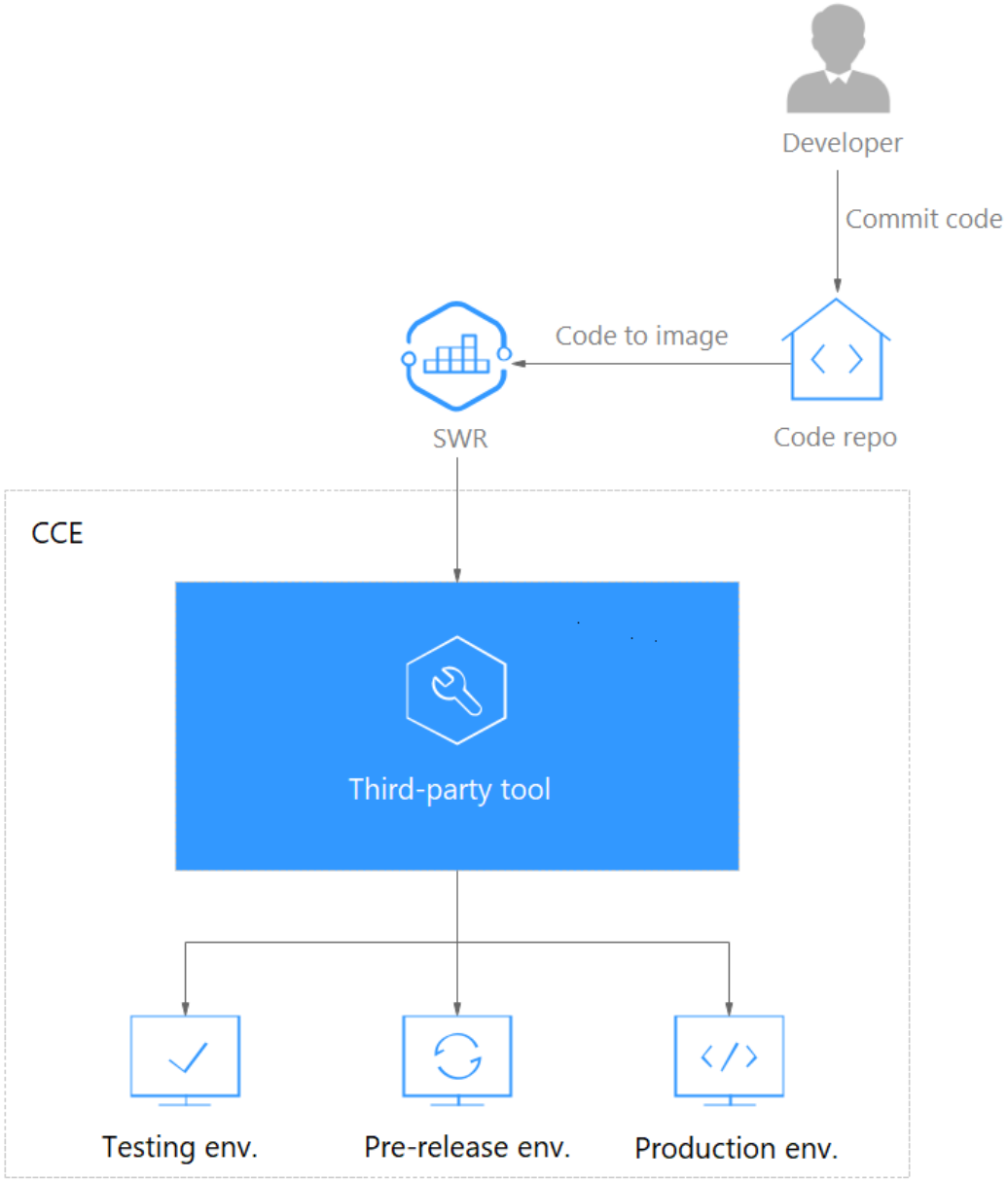
- High performance

  Schedules tasks flexibly with a fully containerized architecture.

Related Services

Software Repository for Container (SWR), Object Storage Service (OBS), and Virtual Private Network (VPN)

Figure 7-5 How DevOps works



7.5 Hybrid Cloud Architecture

Application Scenarios

- Multi-cloud deployment and disaster recovery

  To achieve high service availability, you can deploy applications on container services from multiple cloud providers. When a cloud is down, application load will be automatically distributed to other clouds.

- Traffic distribution and auto scaling

  Large enterprise systems need to span cloud facilities in different regions. They also need to be automatically resizable — they can start small and then scale up as system load grows. This frees enterprises from the costs of

planning, purchasing, and maintaining more cloud facilities than needed and transforms large fixed costs into much smaller variable costs.

- Migration to the cloud and database hosting

  Finance, security, and other industries with a top concern for data confidentiality want to keep critical systems in local IDCs while moving other systems to the cloud. All systems, no matter in local IDCs or in the cloud, are expected to be managed using a unified dashboard.

- Separation of development from deployment

  To ensure IP security, you can set up the production environment on a public cloud and the development environment in a local IDC.

Benefits

Applications and data can be seamlessly migrated between your on-premises network and the cloud, facilitating resource scheduling and disaster recovery (DR). This is made possible through environment-independent containers, network connectivity between private and public clouds, and the ability to centrally manage containers on CCE and your private cloud.

Advantages

- On-cloud DR

  Multicloud helps protect systems from outages. When a cloud is faulty, system loads are automatically diverted to other clouds to ensure service continuity.

- Automatic traffic distribution

  Access latency is reduced by directing user requests to the regional cloud that is closer to where the users are. Once the applications in local IDCs are overloaded, some of the application access requests can be distributed to the cloud with auto scaled nodes and containers.

- Separated service deployments and shared resources

  CCE allows separate storage for sensitive and general service data, separate deployments in the development environment and the production environment, and separate running of computing-intensive and general services. Through auto scaling and unified cluster management, your on-premises and cloud resources can efficiently work together.

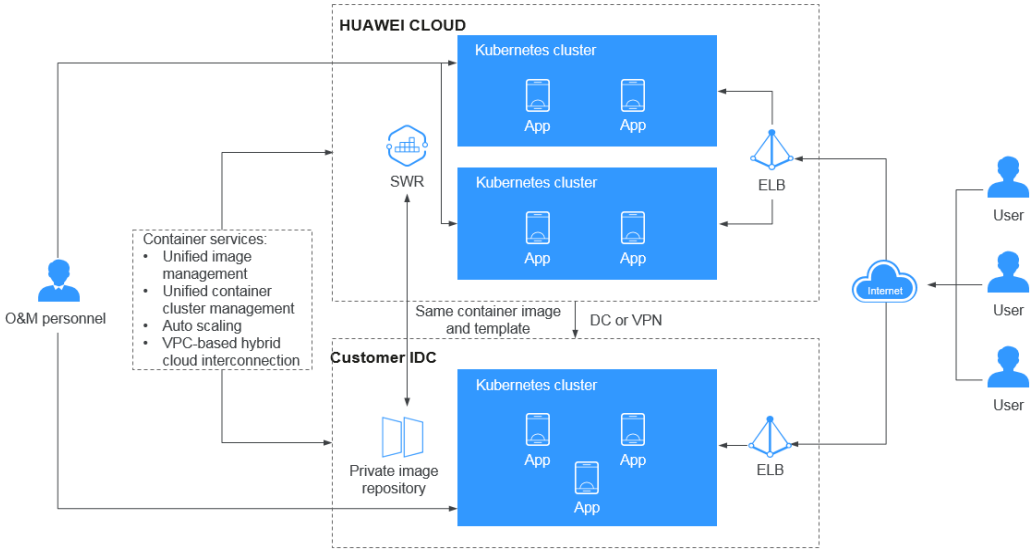
- Lower costs

  Public cloud resource pools can respond quickly to load spikes by auto provisioning resources. Manual operations and maintenance are no longer needed and you can save big.

Related Services

Elastic Cloud Server (ECS), Direct Connect (DC), Virtual Private Network (VPN), and Software Repository for Container (SWR)

Figure 7-6 How hybrid cloud works



7.6 High-Performance Scheduling

CCE integrates Volcano to support high-performance computing.

Volcano is a Kubernetes-native batch processing system. Volcano provides a universal, scalable, and stable platform to run big data and AI jobs. It is compatible with general computing frameworks for AI, big data, gene sequencing, and rendering tasks. Volcano's excellence in task scheduling and heterogeneous chip management makes task running and management more efficient.

Application Scenario 1: Hybrid Deployment of Multiple Types of Jobs

Multiple types of domain frameworks are developed to support business in different industries. These frameworks, such as Spark, TensorFlow, and Flink, function irreplaceably in their service domains. They are not working alone, as services and businesses are becoming increasingly complex. However, resource scheduling becomes a headache as clusters in these frameworks grow larger and a single service may have fluctuating loads. Therefore, a unified scheduling system is in great demand.

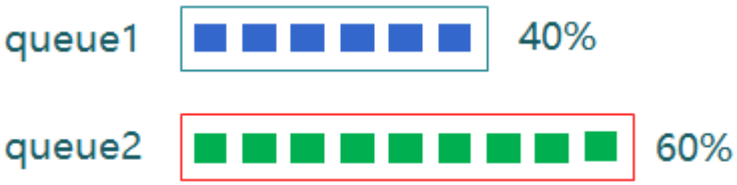
Volcano abstracts a common basic layer for batch computing based on Kubernetes. It supplements Kubernetes in scheduling and provides flexible and universal job abstractions for computing frameworks. These abstractions (Volcano Jobs) are implemented through multi-task templates to describe multiple types of jobs (such as TensorFlow, Spark, MPI, and PyTorch). Different types of jobs can be run together, and Volcano uses its unified scheduling system to realize cluster resource sharing.

Application Scenario 2: Scheduling Optimization in Multi-Queue Scenarios

Resource isolation and sharing are often required when you use a Kubernetes cluster. However, Kubernetes does not support queues. It cannot share resources when multiple users or departments share a machine. Without queue-based resource sharing, HPC and big data jobs cannot run.

Volcano supports multiple resource sharing mechanisms with queues. You can set the **weight** for a queue. The cluster allocates resources to the queue by calculating the ratio of the weight of the queue to the total weight of all queues. You can also set the resource **capability** for a queue to determine the upper limit of resources that can be used by the queue.

For example, in the following figure, queue 1 is allocated 40% of the cluster resources, and 60% for queue 2. In this way, two queues can be mapped to different departments or projects to use resources in the same cluster. If a queue has idle resources, they can be allocated to jobs in another queue.

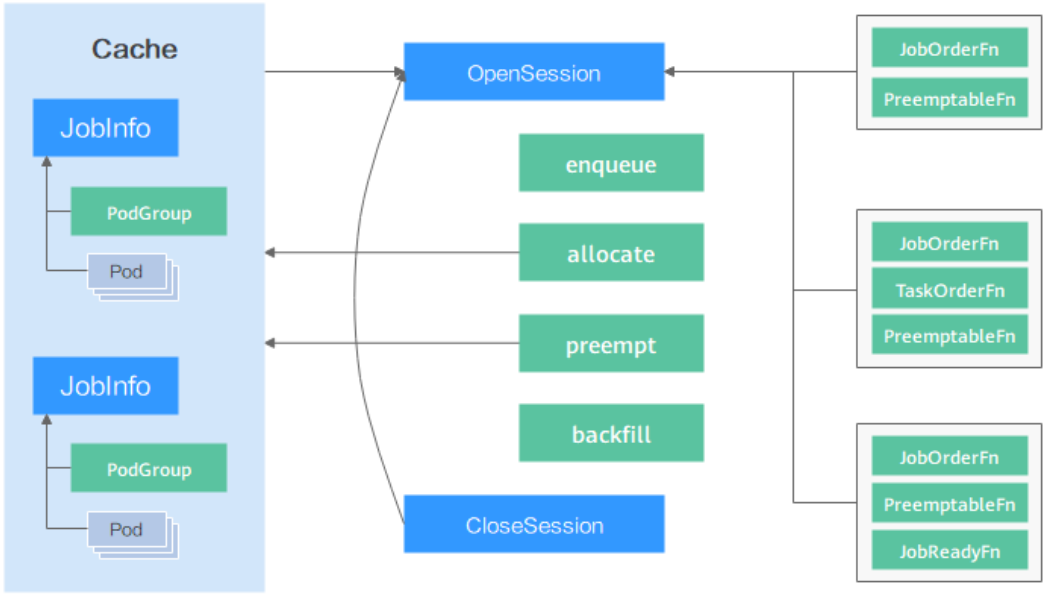


Application Scenario 3: Multiple Advanced Scheduling Policies

Containers are scheduled to nodes that satisfy their requirements on compute resources, such as CPU, memory, and GPU. Normally, there will be more than one qualified node. Each could have different volume of available resources left for new workloads. Volcano automatically analyzes the resource utilization of each scheduling plan and help you achieve the optimal deployment results in great ease.

The following figure shows how the Volcano scheduler schedules resources. First, the scheduler loads the pod and PodGroup information in the API server to the scheduler cache. In a scheduler session, Volcano goes through three phases: OpenSession, action calling, and CloseSession. In OpenSession, the scheduling

policy you configured in the scheduler plugin is loaded. In action calling, the configured actions are called one by one and the loaded scheduling policy is used. In CloseSession, final operations are performed to complete scheduling.



Volcano scheduler provides plugins to support multiple scheduling actions (such as enqueue, allocate, preempt, reclaim and backfill) and scheduling policies (such as gang, priority, drf, proportion and binpack). You can configure them as required. The APIs provided by the scheduler can also be used for customized development.

Application Scenario 4: High-Precision Resource Scheduling

Volcano provides high-precision resource scheduling policies for AI and big data jobs to improve compute efficiency. Take TensorFlow as an example. Configure affinity between ps and worker and anti-affinity between ps and ps, so that ps and worker to the same node. This improves the networking and data interaction performance between ps and worker, thereby improving the compute efficiency. However, when scheduling pods, the default Kubernetes scheduler only checks whether the affinity and anti-affinity configurations of these pods conflict with those of all running pods in the cluster, and does not consider subsequent pods that may also need scheduling.

The task-topology algorithm provided by Volcano calculates the task and node priorities based on the affinity and anti-affinity configurations between tasks in a job. The task affinity and anti-affinity policies in a job and the task-topology algorithm ensure that the tasks with affinity configurations are preferentially scheduled to the same node, and pods with anti-affinity configurations are scheduled to different nodes. The difference between the task-topology algorithm and the default Kubernetes scheduler is that the task-topology algorithm considers the pods to be scheduled as a whole. When pods are scheduled in batches, the affinity and anti-affinity settings between unscheduled pods are considered and applied to the scheduling processes of pods based on priorities.

Benefits

Running containers on high-performance GPU-accelerated cloud servers significantly improves AI computing performance by three to five folds. GPUs can

cost a lot and sharing a GPU among containers greatly reduces AI computing costs. In addition to performance and cost advantages, CCE also offers fully managed clusters that will hide all the complexity in deploying and managing your AI applications so you can focus on high-value development.

Advantages

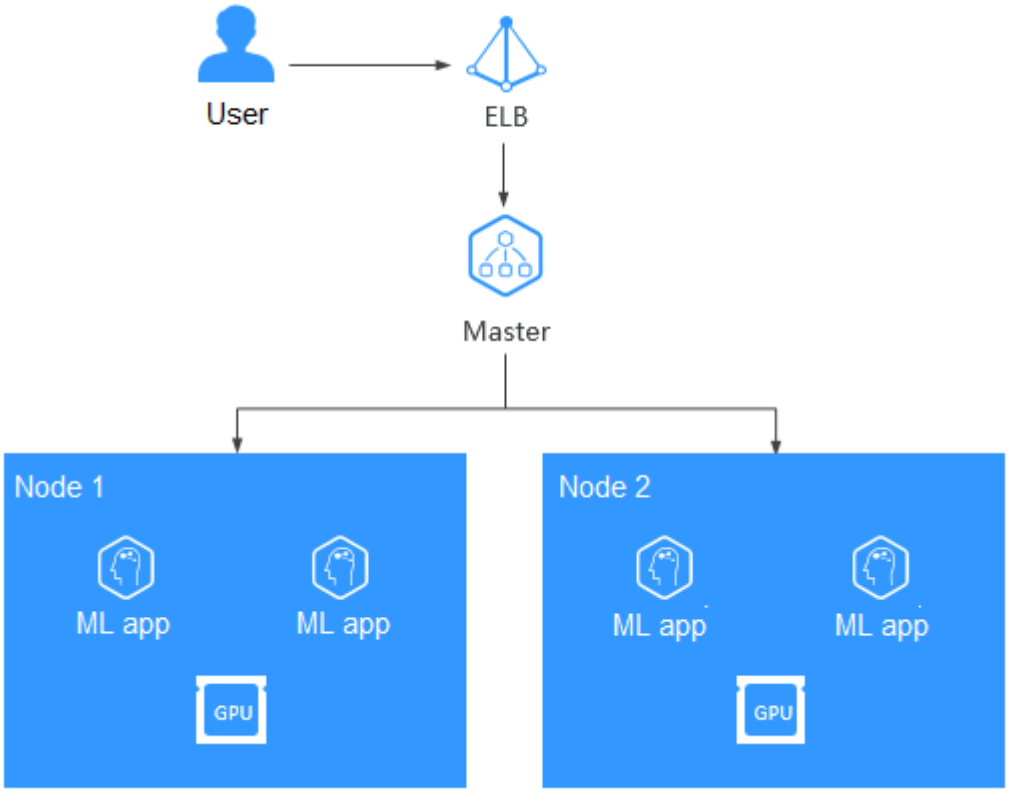
By integrating Volcano, CCE has the following advantages in running high-performance computing, big data, and AI jobs:

- **Hybrid deployment** of HPC, big data, and AI jobs
- **Optimized multi-queue scheduling**: Multiple queues can be used for multi-tenant resource sharing and group planning based on priorities and time periods.
- **Advanced scheduling policies**: gang scheduling, fair scheduling, resource preemption, and GPU topology
- **Multi-task template**: You can use a template to define multiple tasks in a single Volcano Job, beyond the limit of Kubernetes native resources. Volcano Jobs can describe multiple job types, such as TensorFlow, MPI, and PyTorch.
- **Job extension plugins**: The Volcano Controller allows you to configure plugins to customize environment preparation and cleanup in stages such as job submission and pod creation. For example, before submitting a common MPI job, you can configure the SSH plugin to provide the SSH information of pod resources.

Related Services

GPU-accelerated Cloud Server (GACS), Elastic Load Balance (ELB), and Object Storage Service (OBS)

Figure 7-7 How AI computing works



8 Notes and Constraints

This section describes the notes and constraints on using CCE.

Clusters and Nodes

- After a cluster is created, the following items cannot be changed:
  - Cluster type. For example, change a **Kunpeng cluster** to a **CCE cluster**.
  - Number of master nodes in the cluster.
  - AZ of a master node.
  - Network configuration of the cluster, such as the VPC, subnet, container CIDR block, Service CIDR block, IPv6 settings, and kube-proxy (forwarding) settings.
  - Network model. For example, change the **tunnel network** to the **VPC network**.
- Applications cannot be migrated between different namespaces.
- Currently, the created ECS instances (nodes) support the **pay-per-use** and **yearly/monthly** billing modes. Other resources (such as load balancers) support the pay-per-use billing mode. You can change the billing mode from pay-per-use to yearly/monthly on the management console for the created ECS instances.
- Nodes created during cluster creation support **pay-per-use** and **yearly/monthly** billing modes, but with the following constraints:
  - If the cluster to be created is pay-per-use, the nodes created in the cluster must also be pay-per-use.
  - If the cluster to be created is billed on a yearly/monthly basis, nodes in the cluster are either pay-per-use or billed on a yearly/monthly basis.
  - If nodes added after cluster creation are billed on a yearly/monthly basis, they need to be renewed separately from the cluster.

  Note: If you purchase a node after a cluster is created, the billing mode of the node is not restricted by that of the cluster.
- Underlying resources, such as ECSs (nodes), are limited by quotas and their inventory. Therefore, only some nodes may be successfully created during cluster creation, cluster scaling, or auto scaling.
- The ECS (node) specifications must be higher than 2 cores and 4 GB memory.

- To access a CCE cluster through a VPN, ensure that the VPN CIDR block does not conflict with the VPC CIDR block where the cluster resides and the container CIDR block.

Networking

- By default, a NodePort Service is accessed within a VPC. If you need to use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.
- LoadBalancer Services allow workloads to be accessed from public networks through **ELB**. This access mode has the following restrictions:
  - It is recommended that automatically created load balancers not be used by other resources. Otherwise, these load balancers cannot be completely deleted, causing residual resources.
  - Do not change the listener name for the load balancer in clusters of v1.15 and earlier. Otherwise, the load balancer cannot be accessed.
- Constraints on network policies:
  - Only clusters that use the tunnel network model support network policies.
  - Egresses are not supported for network policies.
  - Network isolation is not supported for IPv6 addresses.

Volumes

- Constraints on EVS volumes:
  - By default, CCE creates EVS disks billed in **pay-per-use** mode. To use EVS disks billed in **yearly/monthly** mode, see **Yearly/Monthly-Billed EVS Disks**.
  - EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple jobs.
  - Data in a shared disk cannot be shared between nodes in a CCE cluster. If the same EVS disk is attached to multiple nodes, read and write conflicts and data cache conflicts may occur. When creating a Deployment, you are advised to create only one pod if you want to use EVS disks.
  - For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

    For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.
  - When you create a StatefulSet and add a cloud storage volume, existing EVS volumes cannot be used.
  - EVS disks that have partitions or have non-ext4 file systems cannot be imported.
  - Container storage in CCE clusters of Kubernetes 1.13 or later version supports encryption. Currently, E2E encryption is supported only in certain regions.
  - EVS volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.

- Constraints on SFS volumes:
  - Container storage in CCE clusters of Kubernetes 1.13 or later version supports encryption. Currently, E2E encryption is supported only in certain regions.
  - Volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.
- Constraints on OBS volumes:
  - CCE clusters of v1.7.3-r8 and earlier do not support OBS volumes. You need to upgrade these clusters or create clusters of a later version that supports OBS.
  - Kunpeng clusters do not support obsfs. Therefore, parallel file systems cannot be mounted.
  - Volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.
- Constraints on snapshots and backups:
  - The snapshot function is available **only for clusters of v1.15 or later** and requires the CSI-based everest add-on.
  - The subtype (common I/O, high I/O, or ultra-high I/O), disk mode (SCSI or VBD), data encryption, sharing status, and capacity of an EVS disk created from a snapshot must be the same as those of the disk associated with the snapshot. These attributes cannot be modified after being queried or set.

Services

A Service is a Kubernetes resource object that defines a logical set of pods and a policy by which to access them.

A maximum of 6,000 Services can be created in each namespace.

CCE Cluster Resources

There are resource quotas for your CCE clusters in each region.

Item	Constraints on Common Users
Total number of clusters in a region	50
Number of nodes in a cluster (cluster management scale)	You can select 50, 200, 1000, or 2000 nodes. A maximum of 5000 nodes are supported.
Maximum number of container pods created on each worker node	This number can be set on the console when you are creating a cluster. In the VPC network model, a maximum of 256 pods can be created.

Dependent Underlying Cloud Resources

Category	Item	Constraints on Common Users
Compute	Pods	1,000
	Cores	8,000
	RAM capacity (MB)	16384000
Networking	VPCs per account	5
	Subnets per account	100
	Security groups per account	100
	Security group rules per account	5000
	Routes per route table	100
	Routes per VPC	100
	VPC peering connections per region	50
	Network ACLs per account	200
	Layer 2 connection gateways per account	5
Load balancing	Elastic load balancers	50
	Load balancer listeners	100
	Load balancer certificates	120
	Load balancer forwarding policies	500
	Load balancer backend host group	500
	Load balancer backend server	500

9 Pricing Details

Billing Items

Cloud Container Engine (CCE) is free of charge. You only pay for the resources (such as nodes) created when you are using CCE. There are two types of billing items:

1. **Clusters**: The cluster fee is the cost of resources used by master nodes. The fee varies with the cluster type and cluster size. Cluster types include VM cluster and BMS cluster (the number of master nodes determines whether a cluster is highly available). Cluster size (also called management scale) indicates the maximum number of nodes allowed in a cluster.

   NOTE

   The management scale indicates the number of ECSs or BMSs in a cluster.

   For more details, see **CCE Pricing Details**.
2. **IaaS resources**: The cost of IaaS resources created to run worker nodes in your cluster is billed. IaaS resources, which are created either manually or automatically, include ECSs, EVS disks, EIPs, bandwidth, and load balancers.

   For more pricing details, see **Product Pricing Details**.

Billing Modes

CCE is billed on a pay-per-use or yearly/monthly basis.

- Pay-per-use: It is a pay-after-use mode. Billing starts when a resource is provisioned and stops when the resource is deleted. You can use cloud resources as required and stop paying for them when you no longer need them. There is no upfront payment for excess capacity.

NOTE

The following are pricing principles in the case of CCE cluster hibernation or node shutdown. Note that there are many types of cluster nodes and ECS is used as an example.

- **Cluster hibernation**: After a cluster is hibernated, the billing of resources used by master nodes will stop.
- **Node shutdown**: Worker node billing stops when the node is stopped. Note that hibernating a cluster will not stop worker nodes in the cluster. To stop an ECS, log in to the ECS console. For details, see **Stopping a Node**.

  Stopped ECSs are not billed. For details, see **ECS Billing**.

- Yearly/monthly: It is a pay-before-use mode. Yearly/monthly billing provides a more significant discount than pay-per-use and is recommended for long-term use of cloud services. When you purchase a yearly/monthly package, the system will deduct the package cost from your cloud account based on the chosen specifications.
- Billing mode change: The billing mode cannot be changed within the billing cycle.

NOTICE

- Clusters follow a tiered pricing plan. Pricing for each tier varies with cluster size and type.
- Once a monthly/yearly subscription has expired or a pay-per-use resource becomes in arrears, HUAWEI CLOUD provides a period of time during which you can renew the resource or top up your account. Within the grace period, you can still access and use your cloud service. For details, see **What Is a Grace Period? How Long Is the Grace Period of HUAWEI CLOUDWhat Is a Retention Period? How Long Is the Retention Period of HUAWEI CLOUD**.

Configuration Changes

From pay-per-use to yearly/monthly billing: You can change the cluster billing mode from pay-per-use to yearly/monthly billing. After the change, master nodes, worker nodes, and cloud resources (such as EVS disks and EIPs) used by your cluster will all be billed on a yearly/monthly basis and a new order will be generated. The nodes and cloud resources will be ready for use immediately after you pay for the new order.

From yearly/monthly billing to pay-per-use: Clusters billed on a yearly/monthly basis cannot change to pay-per-use within the billing cycle. Note that pay-per-use clusters can be directly deleted, but clusters billed on a yearly/monthly basis cannot be deleted. To stop using the clusters billed on a yearly/monthly basis, go to the Billing Center and **unsubscribe from them**.

Notes

- Cash coupons will not be returned after you downgrade specifications of the cloud servers that are purchased using cash coupons.
- You will need to pay the price difference between the original and new specifications after upgrading cloud server specifications.

- Downgrading cloud server specifications (the amount of CPU or memory resources) will impair cloud server performance.
- If you downgrade cloud server specifications and then upgrade it to the original specifications, you will still need to pay the price difference incurred by the upgrade.

10 Permissions Management

CCE allows you to assign permissions to IAM users and user groups under your tenant accounts. CCE combines the advantages of Identity and Access Management (IAM) and Kubernetes Role-based Access Control (RBAC) to provide a variety of authorization methods, including IAM fine-grained/token authorization and cluster-/namespace-scoped authorization.

CCE permissions are described as follows:

- **Cluster-level permissions**: Cluster-level permissions management evolves out of the system policy authorization feature of IAM. IAM users in the same user group have the same permissions. On IAM, you can configure system policies to describe which IAM user groups can perform which operations on cluster resources. For example, you can grant user group A to create and delete cluster X, add a node, or install an add-on, while granting user group B to view information about cluster X.

  Cluster-level permissions involve CCE non-Kubernetes APIs and support fine-grained IAM policies and enterprise project management capabilities.

- **Namespace-level permissions**: You can regulate users' or user groups' access to **Kubernetes resources**, such as workloads, jobs, and Services, in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC authentication on access to APIs using IAM tokens.

  Namespace-level permissions involve CCE Kubernetes APIs and are enhanced based on the Kubernetes RBAC capabilities. Namespace-level permissions can be granted to IAM users or user groups for authentication and authorization, but are independent of fine-grained IAM policies. For details, see **Using RBAC Authorization**.

CAUTION

- **Cluster-level permissions** are configured only for cluster-related resources (such as clusters and nodes). You must also configure **namespace permissions** to operate Kubernetes resources (such as workloads, jobs, and Services).
- After you create a cluster of v1.11.7-r2 or later, CCE automatically assigns the cluster-admin permissions of all namespaces in the cluster to you, which means you have full control on the cluster and all resources in all namespaces.

Cluster-level Permissions (Assigned by Using IAM System Policies)

By default, new IAM users do not have permissions assigned. You need to add a user to one or more groups, and attach permissions policies or roles to these groups. Users inherit permissions from the groups to which they are added and can perform specified operations on cloud services based on the permissions.

CCE is a project-level service deployed and accessed in specific physical regions. To assign CCE permissions to a user group, specify the scope as region-specific projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing CCE, the users need to switch to a region where they have been authorized to use the CCE service.

You can grant users permissions by using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. This mechanism provides only a limited number of service-level roles for authorization. When using roles to assign permissions, you need to also assign other roles on which the permissions depend to take effect. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- Policies: A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization, meeting requirements for secure access control. For example, you can assign users only the permissions for managing a certain type of clusters and nodes.

Table 10-1 lists all the system permissions supported by CCE.

Table 10-1 System permissions supported by CCE

Role/ Policy Name	Description	Type	Dependencies
CCE Administrator	Read and write permissions for CCE clusters and all resources (including workloads, nodes, jobs, and Services) in the clusters	Role	Users granted permissions of this policy must also be granted permissions of the following policies: **Global service project**: OBS Buckets Viewer and OBS Administrator **Region-specific projects**: Tenant Guest, Server Administrator, ELB Administrator, SFS Administrator, SWR Admin, and APM FullAccess **NOTE** Users with both **CCE Administrator** and **NAT Gateway Administrator** policies can use NAT Gateway functions for clusters.
CCE FullAccess	Common operation permissions on CCE cluster resources, excluding the namespace-level permissions for the clusters (with Kubernetes RBAC enabled) and the privileged administrator operations, such as agency configuration and cluster certificate generation	Policy	None.
CCE ReadOnly Access	Permissions to view CCE cluster resources, excluding the namespace-level permissions of the clusters (with Kubernetes RBAC enabled)	Policy	None.

Table 10-2 Common operations supported by CCE system policies

Operation	CCE ReadOnlyAccess	CCE FullAccess	CCE Administrator
Creating a cluster	x	√	√
Deleting a cluster	x	√	√
Updating a cluster, for example, updating cluster node scheduling parameters and providing RBAC support to clusters	x	√	√
Upgrading a cluster	x	√	√
Waking up a cluster	x	√	√
Hibernating a cluster	x	√	√
Listing all clusters	√	√	√
Querying cluster details	√	√	√
Adding a node	x	√	√
Deleting one or more nodes	x	√	√
Updating a cluster node, for example, updating the node name	x	√	√
Querying node details	√	√	√
Listing all nodes	√	√	√
Listing all jobs	√	√	√
Deleting one or more cluster jobs	x	√	√
Querying job details	√	√	√
Creating a storage volume	x	√	√
Deleting a storage volume	x	√	√
Performing operations on all Kubernetes resources	√	√	√

Operation	CCE ReadOnlyAccess	CCE FullAccess	CCE Administrator
Performing all operations on an Elastic Cloud Server (ECS)	x	√	√
Performing all operations on Elastic Volume Service (EVS) disks EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.	x	√	√
Performing all operations on VPC A cluster must run in a VPC. When creating a namespace, you need to create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.	x	√	√
Viewing details of all resources on an ECS In CCE, a node is an ECS with multiple EVS disks.	√	√	√
Listing all resources on an ECS	√	√	√
Viewing details about all EVS disk resources EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.	√	√	√
Listing all EVS resources	√	√	√

Operation	CCE ReadOnlyAccess	CCE FullAccess	CCE Administrator
Viewing details about all VPC resources A cluster must run in a VPC. When creating a namespace, you need to create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.	√	√	√
Listing all VPC resources	√	√	√
Viewing details about all Elastic Load Balance (ELB) resources	x	x	√
Listing all ELB resources	x	x	√
Viewing Scalable File Service (SFS) resource details	√	√	√
Listing all SFS resources	√	√	√
Viewing Application Operations Management (AOM) resource details	√	√	√
Listing AOM resources	√	√	√
Performing all operations on AOM auto scaling rules	√	√	√

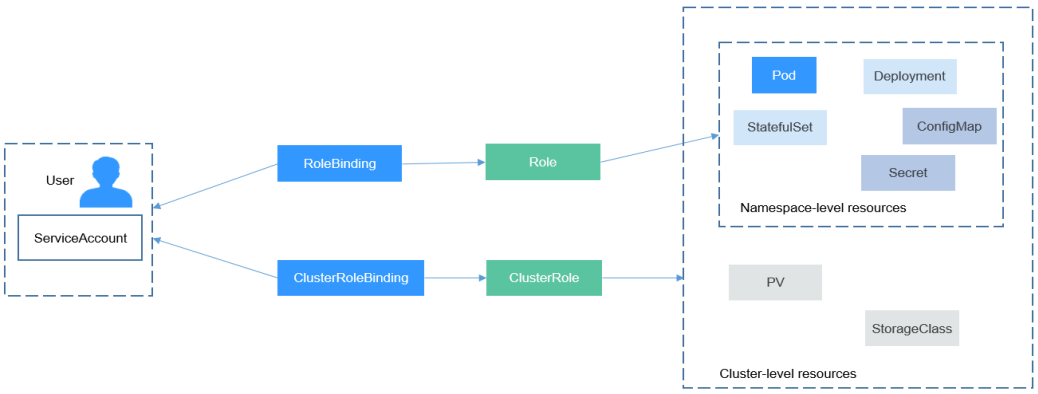
Namespace-level Permissions (Assigned by Using Kubernetes RBAC)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- Role: defines a set of rules for accessing Kubernetes resources in a namespace.
- RoleBinding: defines the relationship between users and roles.
- ClusterRole: defines a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).
- ClusterRoleBinding: defines the relationship between users and cluster roles.

Role and ClusterRole specify actions that can be performed on specific resources. RoleBinding and ClusterRoleBinding bind roles to specific users, user groups, or ServiceAccounts. See the following figure.

Figure 10-1 Role binding



On the CCE console, you can assign permissions to a user or user group to access resources in one or multiple namespaces. By default, the CCE console provides the following five ClusterRoles:

- view: has the permission to view namespace resources.
- edit: has the permission to modify namespace resources.
- admin: has all permissions on the namespace.
- cluster-admin: has all permissions on the cluster.
- psp-global: controls sensitive security aspects of the pod specification.

In addition to cluster-admin, admin, edit, and view, you can define Roles and RoleBindings to configure the permissions to add, delete, modify, and query resources, such as pods, Deployments, and Services, in the namespace.

Helpful Links

- **IAM Service Overview**
- **Granting Cluster-level Permissions**
- **Permissions Policies and Supported Actions**

11 Basic Concepts

11.1 Basic Concepts

CCE provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

The graphical CCE console enables E2E user experiences. In addition, CCE supports native Kubernetes APIs and kubectl. Before using CCE, you are advised to understand related basic concepts.

Cluster

A cluster is a group of one or more cloud servers (also known as nodes) in the same subnet. It has all the cloud resources (including VPCs and compute resources) required for running containers.

Node

A node is a cloud server (virtual or physical machine) running an instance of the Docker Engine. Containers are deployed, run, and managed on nodes. The node agent (kubelet) runs on each node to manage container instances on the node. The number of nodes in a cluster can be scaled.

Node Pool

A node pool contains one node or a group of nodes with identical configuration in a cluster.

Virtual Private Cloud (VPC)

A VPC is a logically isolated virtual network that facilitates secure internal network management and configurations. Resources in the same VPC can communicate with each other, but those in different VPCs cannot communicate with each other by default. VPCs provide the same network functions as physical networks and also advanced network services, such as elastic IP addresses and security groups.

Security Group

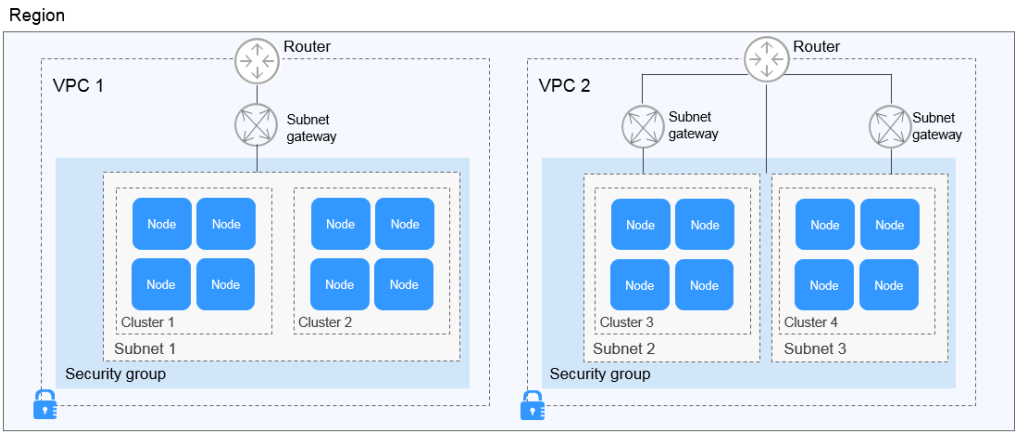
A security group is a collection of access control rules for ECSs that have the same security protection requirements and are mutually trusted in a VPC. After a security group is created, you can create different access rules for the security group to protect the ECSs that are added to this security group.

Relationship Between Clusters, VPCs, Security Groups, and Nodes

As shown in **Figure 11-1**, a region may comprise multiple VPCs. A VPC consists of one or more subnets. The subnets communicate with each other through a subnet gateway. A cluster is created in a subnet. There are three scenarios:

- Different clusters are created in different VPCs.
- Different clusters are created in the same subnet.
- Different clusters are created in different subnets.

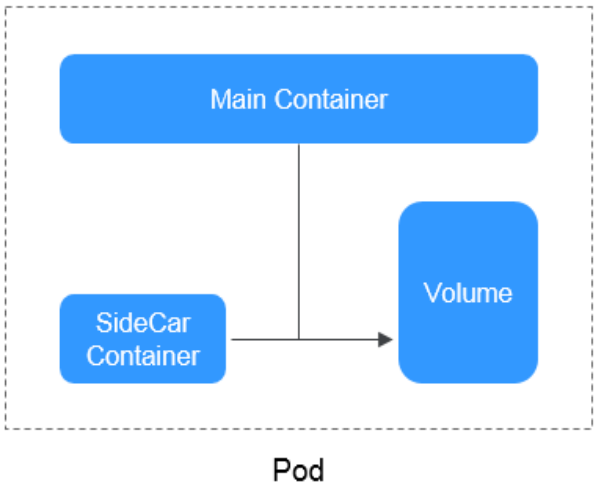
Figure 11-1 Relationship between clusters, VPCs, security groups, and nodes



Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates an application container (or, in some cases, multiple containers), storage resources, a unique network IP address, and options that govern how the containers should run.

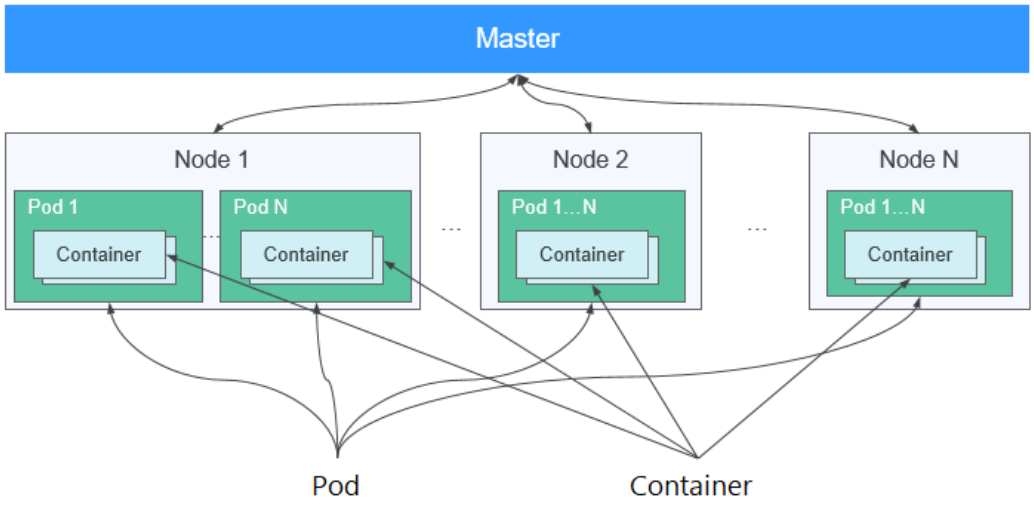
Figure 11-2 Pod



Container

A container is a running instance of a Docker image. Multiple containers can run on one node. Containers are actually software processes. Unlike traditional software processes, containers have separate namespace and do not run directly on a host.

Figure 11-3 Relationships between pods, containers, and nodes

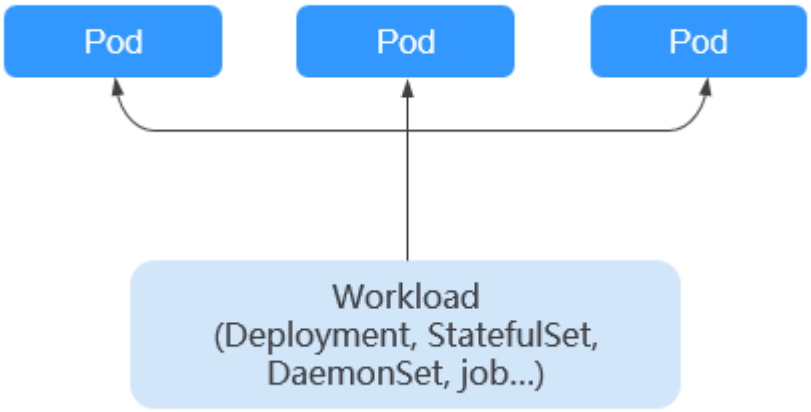


Workload

A workload is an application running on Kubernetes. No matter how many components are there in your workload, you can run it in a group of Kubernetes pods. A workload is an abstract model of a group of pods in Kubernetes. Workloads classified in Kubernetes include Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

- **Deployment**: Pods are completely independent of each other and functionally identical. They feature auto scaling and rolling upgrade. Typical examples include Nginx and WordPress.
- **StatefulSet**: Pods are not completely independent of each other. They have stable persistent storage, and feature orderly deployment and deletion. Typical examples include MySQL-HA and etcd.
- **DaemonSet**: A DaemonSet ensures that all or some nodes run a pod. It is applicable to pods running on every node. Typical examples include Ceph, Fluentd, and Prometheus Node Exporter.
- **Job**: It is a one-time task that runs to completion. It can be executed immediately after being created. Before creating a workload, you can execute a job to upload an image to the image repository.
- **Cron job**: It runs a job periodically on a given schedule. You can perform time synchronization for all active nodes at a fixed time point.

Figure 11-4 Relationship between workloads and pods

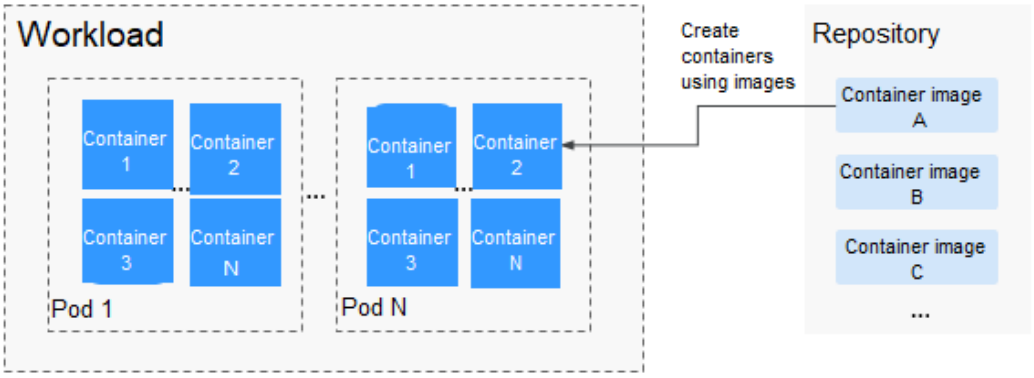


Image

Docker creates an industry standard for packaging containerized applications. Docker images are like templates that include everything needed to run containers, and are used to create Docker containers. In other words, Docker image is a special file system that includes everything needed to run containers: programs, libraries, resources, and configuration files. It also contains configuration parameters (such as anonymous volumes, environment variables, and users) required within a container runtime. An image does not contain any dynamic data. Its content remains unchanged after being built. When deploying containerized applications, you can use images from Docker Hub, SoftWare Repository for Container (SWR), and your private image registries. For example, a Docker image can contain a complete Ubuntu operating system, in which only the required programs and dependencies are installed.

Images become containers at runtime, that is, containers are created from images. Containers can be created, started, stopped, deleted, and suspended.

Figure 11-5 Relationship between images, containers, and workloads



Namespace

A namespace is an abstract collection of resources and objects. It enables resources to be organized into non-overlapping groups. Multiple namespaces can be created inside a cluster and isolated from each other. This enables namespaces to share the same cluster services without affecting each other. Examples:

- You can deploy workloads in a development environment into one namespace, and deploy workloads in a test environment into another namespace.
- Pods, Services, ReplicationControllers, and Deployments belong to a namespace (named **default**, by default), whereas nodes and PersistentVolumes do not belong to any namespace.

Service

A Service is an abstract method that exposes a group of applications running on a pod as network services.

Kubernetes provides you with a service discovery mechanism without modifying applications. In this mechanism, Kubernetes provides pods with their own IP addresses and a single DNS for a group of pods, and balances load between them.

Kubernetes allows you to specify a Service of a required type. The values and actions of different types of Services are as follows:

- **ClusterIP**: ClusterIP Service, as the default Service type, is exposed through the internal IP address of the cluster. If this mode is selected, Services can be accessed only within the cluster.
- **NodePort**: NodePort Services are exposed through the IP address and static port of each node. A ClusterIP Service, to which a NodePort Service will route, is automatically created. By sending a request to <NodeIP>:<NodePort>, you can access a NodePort Service from outside of a cluster.
- **LoadBalancer (ELB)**: LoadBalancer (ELB) Services are exposed by using load balancers of the cloud provider. External load balancers can route to NodePort and ClusterIP Services.
- **DNAT**: A DNAT gateway translates addresses for cluster nodes and allows multiple cluster nodes to share an EIP. DNAT Services provide higher reliability than EIP-based NodePort Services in which the EIP is bound to a single node

and once the node is down, all inbound requests to the workload will be distributed.

Layer-7 Load Balancing (Ingress)

An ingress is a set of routing rules for requests entering a cluster. It provides Services with URLs, load balancing, SSL termination, and HTTP routing for external access to the cluster.

Network Policy

Network policies provide policy-based network control to isolate applications and reduce the attack surface. A network policy uses label selectors to simulate traditional segmented networks and controls traffic between them and traffic from outside.

ConfigMap

A ConfigMap is used to store configuration data or configuration files as key-value pairs. ConfigMaps are similar to secrets, but provide a means of working with strings that do not contain sensitive information.

Secret

Secrets resolve the configuration problem of sensitive data such as passwords, tokens, and keys, and will not expose the sensitive data in images or pod specs. A secret can be used as a volume or an environment variable.

Label

A label is a key-value pair and is associated with an object, for example, a pod. Labels are used to identify special features of objects and are meaningful to users. However, labels have no direct meaning to the kernel system.

Label Selector

Label selector is the core grouping mechanism of Kubernetes. It identifies a group of resource objects with the same characteristics or attributes through the label selector client or user.

Annotation

Annotations are defined in key-value pairs as labels are.

Labels have strict naming rules. They define the metadata of Kubernetes objects and are used by label selectors.

Annotations are additional user-defined information for external tools to search for a resource object.

PersistentVolume

A PersistentVolume (PV) is a network storage in a cluster. Similar to a node, it is also a cluster resource.

PersistentVolumeClaim

A PV is a storage resource, and a PersistentVolumeClaim (PVC) is a request for a PV. PVC is similar to pod. Pods consume node resources, and PVCs consume PV resources. Pods request CPU and memory resources, and PVCs request data volumes of a specific size and access mode.

Auto Scaling - HPA

Horizontal Pod Autoscaling (HPA) is a function that implements horizontal scaling of pods in Kubernetes. The scaling mechanism of ReplicationController can be used to scale your Kubernetes clusters.

Affinity and Anti-Affinity

If an application is not containerized, multiple components of the application may run on the same virtual machine and processes communicate with each other. However, in the case of containerization, software processes are packed into different containers and each container has its own lifecycle. For example, the transaction process is packed into a container while the monitoring/logging process and local storage process are packed into other containers. If closely related container processes run on distant nodes, routing between them will be costly and slow.

- Affinity: Containers are scheduled onto the nearest node. For example, if application A and application B frequently interact with each other, it is necessary to use the affinity feature to keep the two applications as close as possible or even let them run on the same node. In this way, no performance loss will occur due to slow routing.
- Anti-affinity: Instances of the same application spread across different nodes to achieve higher availability. Once a node is down, instances on other nodes are not affected. For example, if an application has multiple replicas, it is necessary to use the anti-affinity feature to deploy the replicas on different nodes. In this way, no single point of failure will occur.

Node Affinity

By selecting labels, you can schedule pods to specific nodes.

Node Anti-Affinity

By selecting labels, you can prevent pods from being scheduled to specific nodes.

Pod Affinity

You can deploy pods onto the same node to reduce consumption of network resources.

Pod Anti-Affinity

You can deploy pods onto different nodes to reduce the impact of system breakdowns. Anti-affinity deployment is also recommended for workloads that may interfere with each other.

Resource Quota

Resource quotas are used to limit the resource usage of users.

Resource Limit (LimitRange)

By default, all containers in Kubernetes have no CPU or memory limit. LimitRange (**limits** for short) is used to add a resource limit to a namespace, including the minimum, maximum, and default amounts of resources. When a pod is created, resources are allocated according to the **limits** parameters.

Environment Variable

An environment variable is a variable whose value can affect the way a running container will behave. A maximum of 30 environment variables can be defined at container creation time. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

The function of setting environment variables on CCE is the same as that of specifying ENV in a Dockerfile.

11.2 Mappings Between CCE and Kubernetes Terms

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of container clusters. It is a container orchestration tool and a leading solution based on the distributed architecture of the container technology. Kubernetes is built on the open-source Docker technology that automates deployment, resource scheduling, service discovery, and dynamic scaling of containerized applications.

This topic describes the mappings between CCE and Kubernetes terms.

Table 11-1 Mappings between CCE and Kubernetes terms

CCE	Kubernetes
Cluster	Cluster
Node	Node
Node pool	NodePool
Container	Container
Image	Image
Namespace	Namespace
Deployment	Deployment
StatefulSet	StatefulSet
DaemonSet	DaemonSet
Job	Job

CCE	Kubernetes
Cron job	CronJob
Pod	Pod
Service	Service
ClusterIP	Cluster IP
NodePort	NodePort
LoadBalancer	LoadBalancer
Layer-7 load balancing	Ingress
Network policy	NetworkPolicy
Chart	Template
ConfigMap	ConfigMap
Secret	Secret
Label	Label
Label selector	LabelSelector
Annotation	Annotation
Volume	PersistentVolume
PersistentVolumeClaim	PersistentVolumeClaim
Auto scaling	HPA
Node affinity	NodeAffinity
Node anti-affinity	NodeAntiAffinity
Pod affinity	PodAffinity
Pod anti-affinity	PodAntiAffinity
Webhook	Webhook
Endpoint	Endpoint
Quota	Resource Quota
Resource limit	Limit Range

11.3 Regions and AZs

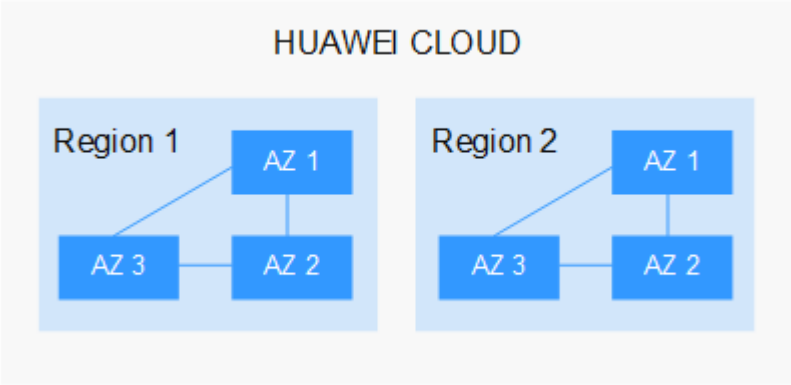
Definition

A region and availability zone (AZ) identify the location of a data center. You can create resources in a specific region and AZ.

- Regions are divided based on geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are shared within the same region. Regions are classified as universal regions and dedicated regions. A universal region provides universal cloud services for common domains. A dedicated region provides services of the same type only or for specific domains.
- An AZ contains one or more physical data centers. Each AZ has independent cooling, fire extinguishing, moisture-proof, and electricity facilities. Within an AZ, computing, network, storage, and other resources are logically divided into multiple clusters. AZs in a region are interconnected through high-speed optic fibers. This is helpful if you will deploy systems across AZs to achieve higher availability.

Figure 11-6 shows the relationship between the region and AZ.

Figure 11-6 Regions and AZs



Huawei Cloud provides services in many regions around the world. You can select a region and AZ as needed. For more information, see **Global Products and Services**.

How to Select a Region?

When selecting a region, consider the following factors:

- Location

  Select a region close to you or your target users to reduce network latency and improve access rate. Chinese mainland regions provide basically the same infrastructure, BGP network quality, as well as operations and configurations on resources. If you or your target users are in the Chinese mainland, you do not need to consider the network latency differences when selecting a region.

  - If you or your target users are in the Asia Pacific region, except the Chinese mainland, select the **CN-Hong Kong**, **AP-Bangkok**, or **AP-Singapore** region.
  - If you or your target users are in South Africa, select the **AF-Johannesburg** region.
  - If you or your target users are in Europe, select the **EU-Paris** region.

  - If you or your target users are in Latin America, select the **LA-Santiago** region.

    NOTE

    The **LA-Santiago** region is located in Chile.

- Resource price

  Resource prices may vary in different regions. For details, see **Product Pricing Details**.

Selecting an AZ

When deploying resources, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs within the same region.
- For lower network latency, deploy resources in the same AZ.

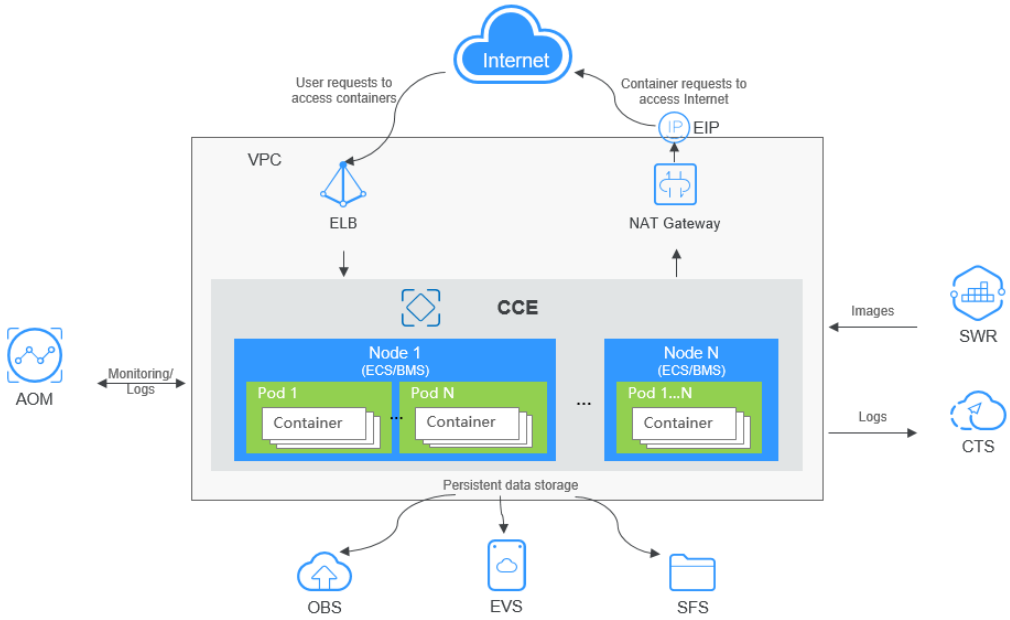
Regions and Endpoints

When using an API to access resources, you must specify a region and endpoint. For details, see **Regions and Endpoints**.

12 Related Services

CCE works with the following cloud services and requires permissions to access them.

Figure 12-1 Relationships between CCE and other services



Relationships Between CCE and Other Services

Table 12-1 Relationships between CCE and other services

Service	Relationship	Related Features
Elastic Cloud Server (ECS)	An ECS with multiple EVS disks is a node in CCE. You can choose ECS specifications during node creation.	• **Buying a Node** • **Accepting Existing Nodes into a Cluster**

Service	Relationship	Related Features
Virtual Private Cloud (VPC)	For security reasons, all clusters created by CCE must run in **VPCs**. When creating a namespace, you need to create a VPC or bind an existing VPC to the namespace so all containers in the namespace will run in this VPC.	**Buying a CCE Cluster**
Elastic Load Balance (ELB)	CCE works with ELB to load balance a workload's access requests across multiple pods. When **ELB** is used, the load balancer's address, instead of the workload address, is exposed to users. User requests first arrive at ELB via a public network and then routed by ELB to different pods of the workload.	• **Creating a Deployment** • **Creating a StatefulSet** • **LoadBalancer**
NAT Gateway	The NAT Gateway service provides source network address translation (SNAT) for container instances in a VPC. The SNAT feature translates private IP addresses of these container instances to the same EIP, which is a public IP address reachable on Internet. You can define SNAT rules on the **NAT gateway** to let containers access the Internet.	• **Creating a Deployment** • **Creating a StatefulSet** • **DNAT**
Software Repository for Container (SWR)	An image repository is used to store and manage Docker images. You can create workloads from images in **SWR**.	• **Creating a Deployment** • **Creating a StatefulSet**
Elastic Volume Service (EVS)	EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed. An ECS with multiple EVS disks is a node in CCE. You can choose ECS specifications during node creation.	**Using EVS Volumes**

Service	Relationship	Related Features
Object Storage Service (OBS)	OBS provides stable, secure, cost-efficient, and object-based cloud storage for data of any size. With OBS, you can create, modify, and delete buckets, as well as uploading, downloading, and deleting objects. CCE allows you to create an OBS volume and attach it to a path inside a container.	**Using OBS Volumes**
Scalable File Service (SFS)	SFS is a shared, fully managed file storage service. Compatible with the Network File System protocol, SFS file systems can elastically scale up to petabytes, thereby ensuring top performance of data-intensive and bandwidth-intensive applications. You can use SFS file systems as persistent storage for containers and attach the file systems to containers when creating a workload.	**Using SFS Volumes**
Application Operations Management (AOM)	AOM collects container log files in formats like .log from CCE and dumps them to AOM. On the AOM console, you can easily query and view log files. In addition, AOM monitors CCE resource usage. You can define metric thresholds for CCE resource usage to trigger auto scaling.	**Collecting Standard Output Logs of Containers**
Cloud Trace Service (CTS)	CTS records operations on your cloud resources, allowing you to query, audit, and backtrack resource operation requests initiated from the management console or open APIs as well as responses to these requests.	**CCE Operations Supported by CTS**