Cloud Container Instance

Service Overview

Issue 01

Date 2023-10-13

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property 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 Cloud 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 Overview

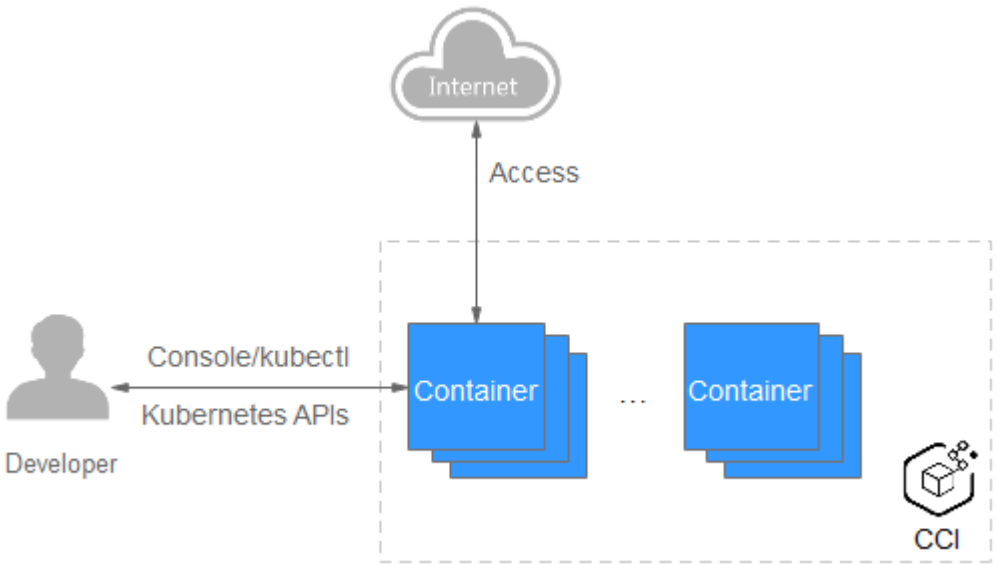
What Is CCI

Cloud Container Instance (CCI) is a serverless container engine that allows you to run containers without creating or managing clusters.

Traditionally, to run containerized workloads using Kubernetes, you need to create a Kubernetes server cluster first. In the serverless model, a cloud provider runs servers and dynamically allocates resources so that you can build and run applications without having to worry about server statuses. This model helps you improve development efficiency and reduce IT costs.

CCI uses the serverless model that allows you to directly create and use containerized workloads on the console or by calling kubectl or Kubernetes APIs, and pay only for the resources consumed by these workloads.

Figure 1-1 Using CCI



Functions

One-stop Container Lifecycle Management

CCI allows you to run containers without creating or managing server clusters. With the serverless model, you can deploy and run workloads on the console or by using kubectl or Kubernetes APIs, and pay only for the resources consumed by these workloads.

Heterogeneous Network Access

Various network access modes and load balancing at both Layer 4 and Layer 7 are available to meet scenario-specific needs.

Choices of Persistent Storage Volumes

Data can be stored in Elastic Volume Service (EVS) and SFS Turbo.

Fast Auto Scaling

Scaling policies can be user-defined to implement elastic scaling within 1 second. In addition, these policies can be combined flexibly to cope with traffic surge during peak hours.

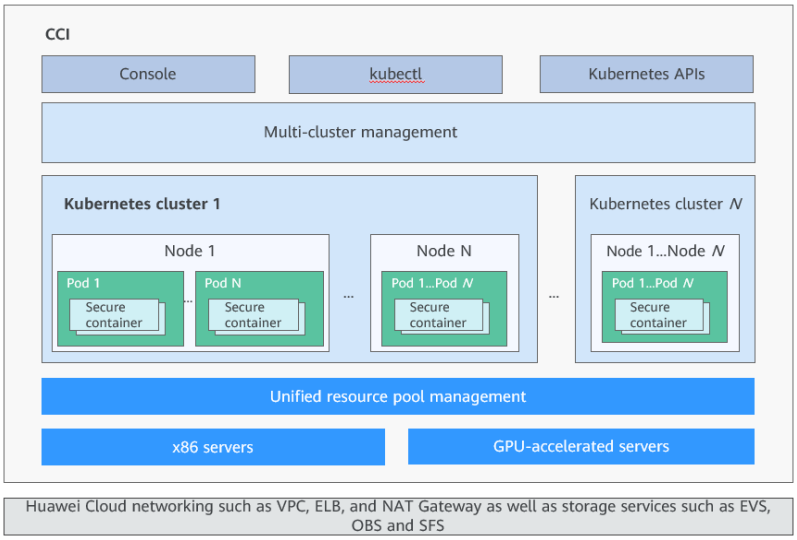
Comprehensive Container Status Monitoring

The resources consumed by containers are monitored, such as the CPU usage, memory usage, GPU compute usage, and GPU memory usage, from which you can determine the running status of the containers in real time.

Architecture

CCI integrates heterogeneous Kubernetes clusters and network and storage services, enabling you to easily create and use containerized workloads using the console, kubectl, or Kubernetes APIs.

Figure 1-2 Architecture



- CCI is deeply integrated with network services, for example Virtual Private Cloud (VPC), Elastic Load Balancer (ELB), and NAT Gateway, as well as storage services such as Elastic Volume Service (EVS).

- CCI supports high-performance and heterogeneous computing architectures such as x86, GPU, and Ascend so that containers directly run on physical servers.
- CCI uses secure containers and in-house hardware virtualization acceleration technologies for VM-level isolation, thereby providing high-performance and secure container services.
- With unified cluster management and workload scheduling, you do not need to manage clusters.
- The **Kubernetes**-based workload model provides fast workload deployment, elastic load balancing, auto scaling, and blue-green release.

CCI Learning Path

You can **learn more about CCI** so that you can use CCI and perform O&M with ease.

2 Advantages

Out of the Box

The serverless container model allows you to run containers by using the console, kubectl, or Kubernetes APIs without creating a Kubernetes cluster.

Fast Scaling

Kubernetes cluster resources are unlimited from a single user's perspective. Resources can be scaled in seconds, helping you cope with service changes and ensure SLAs.

Per-second Billing

Resources can be billed by the second to reduce costs.

Native Platform

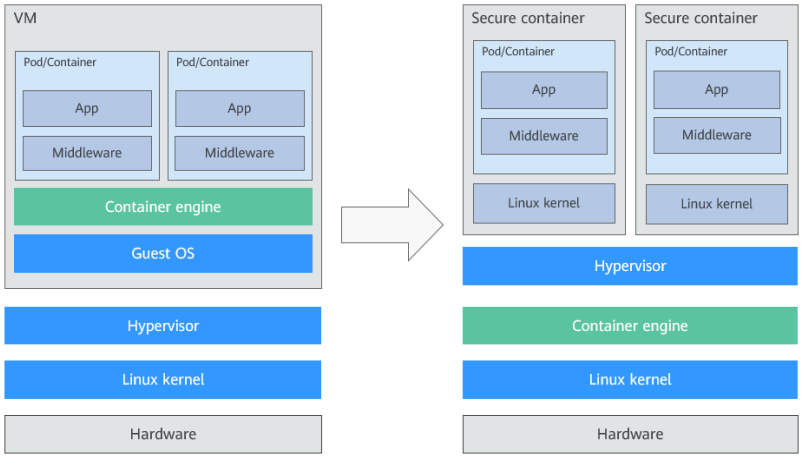
- The latest versions in Kubernetes are updated in a timely manner.
- Kubernetes native APIs are supported.

High Security

CCI provides VM-level isolation without compromising the startup speed, offering you better container experience. CCI has the following security features:

- Secure containers are used.
- Kernel virtualization based on secure containers provides comprehensive security isolation and protection
- In-house virtualization acceleration technologies improve the performance of secure containers.

Figure 2-1 Hard multi-tenancy brought by secure containers



3 Application Scenarios

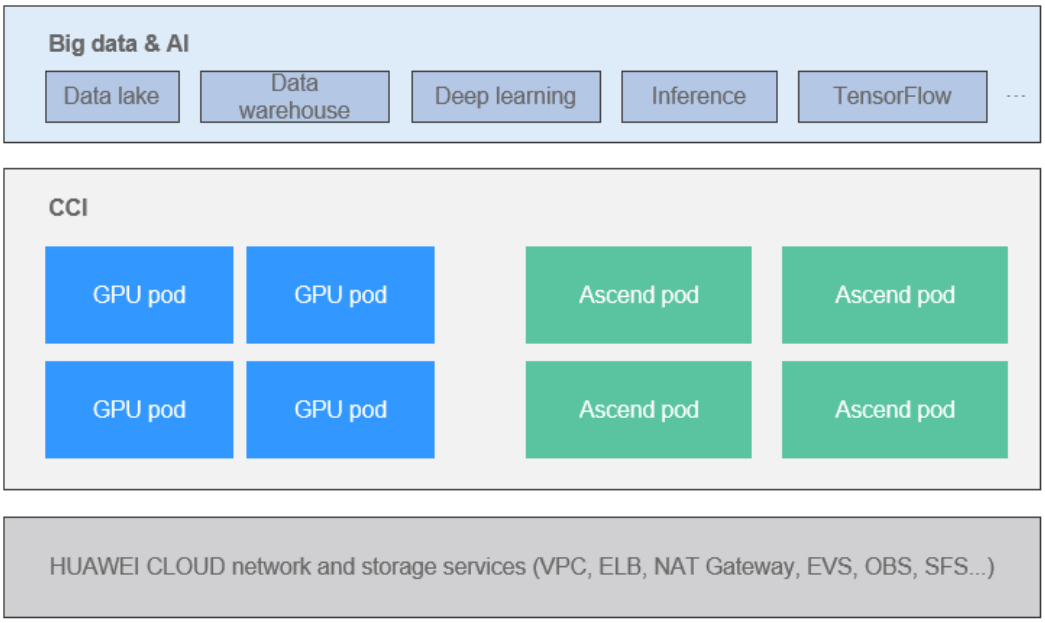
Big Data and AI Computing

Currently, most big data and AI training applications (such as TensorFlow and Caffe) run in containerized mode. These applications are GPU intensive and require high-performance network and storage. In addition, as these applications are task-based, resources must be quickly allocated upon task creation and released upon task completion.

The following CCI features make it suitable for running these applications:

- Accelerated computing with heterogeneous GPUs and Ascend chips (in-house AI chips)
- Large-scale, high-concurrency container creation and management
- On-demand usage and billing

Figure 3-1 Big data and AI computing



Scientific Computing

Scientific R&D in fields such as genomics and drug development requires high-performance and high-density computing. In addition, scientific computing is generally task-based and resources need to be quickly allocated and released. Therefore, a low-cost computing platform with automated O&M is required.

The following CCI features make it suitable for computing in this scenario:

- High-performance computing and network, and high I/O storage
- Resource scaling in seconds minimizes resource consumption
- No O&M required for clusters and servers, greatly reducing O&M costs
- On-demand usage and billing

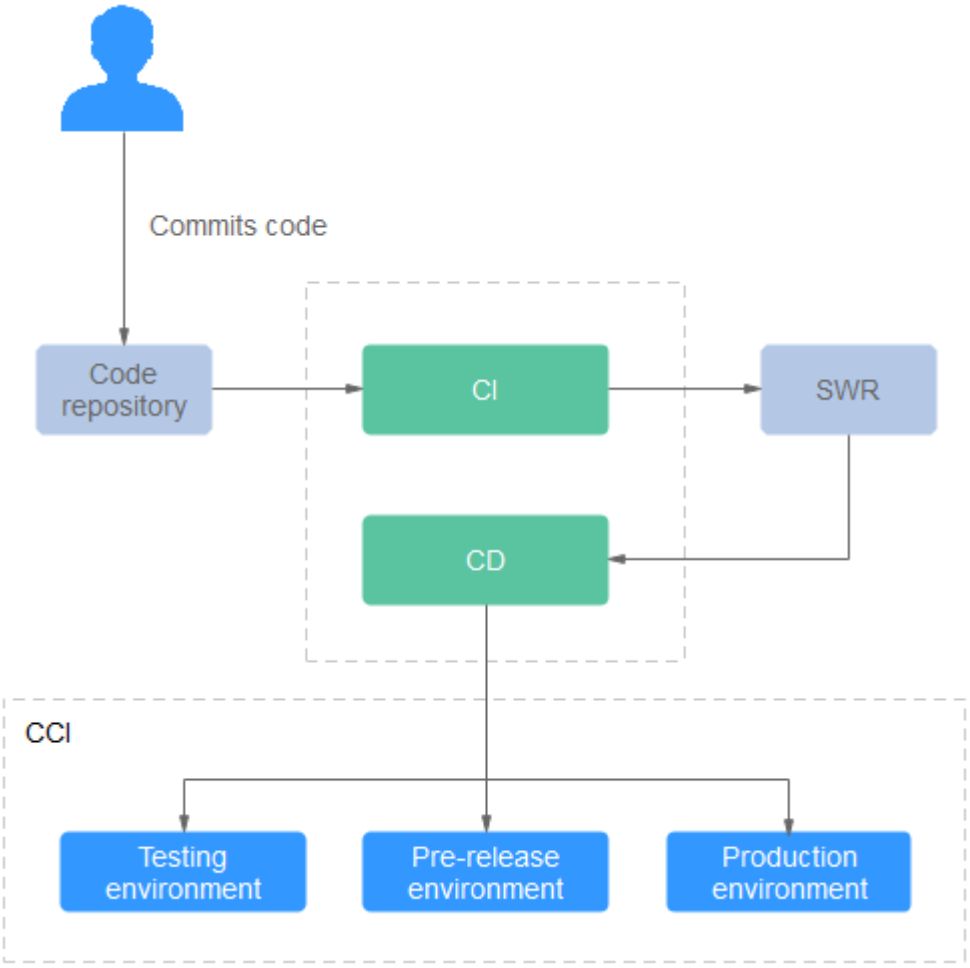
DevOps/Continuous Delivery

Software development enterprises need a complete DevOps process from code submission to application deployment to improve the development efficiency. DevOps processes such as continuous integration/continuous delivery (CI/CD) are generally task-based computing and require quick resource allocation and release.

The following CCI features make it suitable for computing in this scenario:

- Automation for the entire CI/CD process, with no cluster creation and maintenance required
- Image-based delivery, allowing for consistency between the development and production environments
- On-demand usage and billing

Figure 3-2 DevOps/Continuous delivery



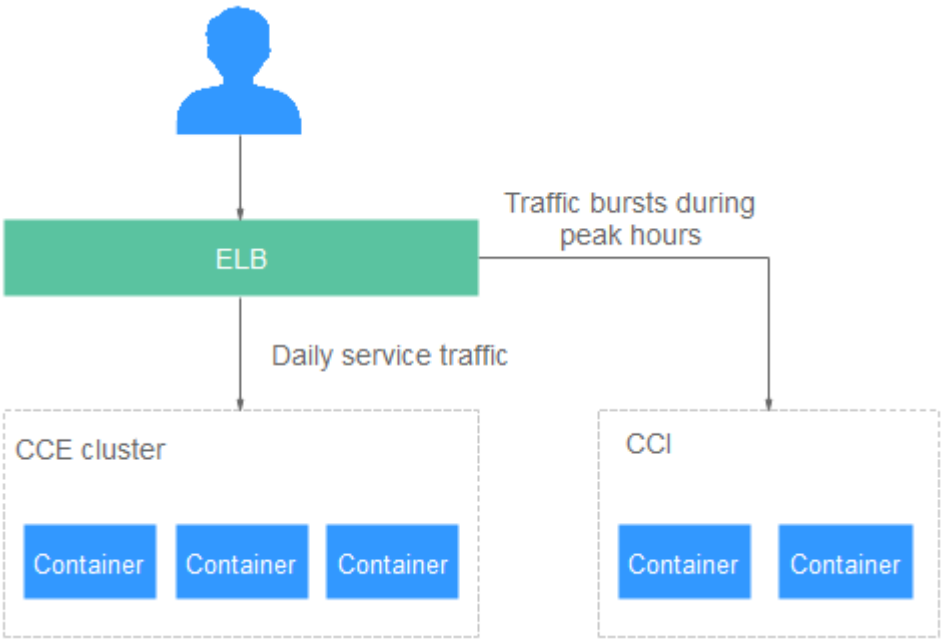
Services with Fluctuating Traffic

Some types of applications, such as live video, media information, e-commerce, and online education, have obvious service peaks and troughs. For these applications, resources need to be expanded rapidly during peak hours without breaking the bank.

The following CCI features make it suitable for these applications:

- **Fast scaling**: CCI can quickly take over services from CCE to ensure uptime during peak hours.
- **Low-cost, flexible billing modes**: When services are stable, they can be run on CCE and be periodically billed. During peak hours, additional services can be run on CCI and be billed based on the usage. This mode greatly reduces costs.

Figure 3-3 Auto scaling



4 Basic Concepts

CCI provides enhanced features based on the **Kubernetes** workload model, including security isolation, fast workload deployment, elastic load balancing, auto scaling, and blue-green release.

The graphical CCI console provides end-to-end user experience. In addition, CCI supports Kubernetes native APIs and kubectl. Before using CCI, you are advised to understand related basic concepts.

Image

A container image is a special file system that provides the programs, libraries, resources, and configuration files required for running a container. A Docker image also contains configuration parameters, for example, anonymous volumes, environment variables, and users. An image does not contain any dynamic data, and its content will not be changed after creation.

Container

The relationship between a Docker image and a container is similar to that between a class and an instance in object-oriented programming. Images are static, and containers are entities of running images. A container can be created, started, stopped, deleted, and suspended.

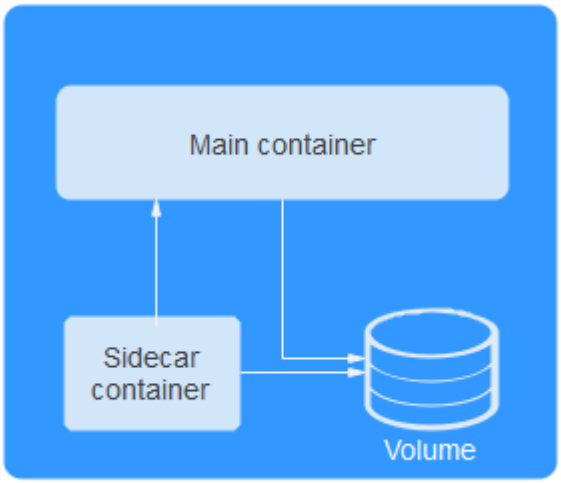
Namespace

A namespace provides a method of allocating resources among multiple users. When you have a large number of projects and personnel, you can define namespaces by project attributes, such as production, test, and development.

Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates one or more containers, storage resources, a unique network IP address, and options that govern how the containers should run.

Figure 4-1 Pod



Pods can be used in either of the following ways:

- One container runs in one pod. This is the most common usage of pods in Kubernetes. You can view the pod as a single encapsulated container, but Kubernetes directly manages pods instead of containers.
- Multiple containers that need to be coupled and share resources run in a pod.

In Kubernetes, pods are rarely created directly. Instead, controllers such as Deployments are used to create and manage pods. Controllers create and manage multiple pods, and provide replica management, rolling upgrade, and self-healing capabilities. A controller typically uses a pod template to create corresponding pods.

For details, see **Pods**.

Init Container

Before containers that run applications are started, one or some init containers are started first. If there are multiple init containers, they will be started in the defined sequence. The application containers are started only after all init containers run to completion and exit. Storage volumes in a pod are shared. Therefore, the data generated in the init containers can be used by the application containers.

Init containers can be used in multiple Kubernetes resources, such as Deployments and jobs. They perform initialization before application containers are started.

For details, see **Init Containers**.

Label

A label is a key-value pair attached to an object and is used to transfer user-defined attributes.

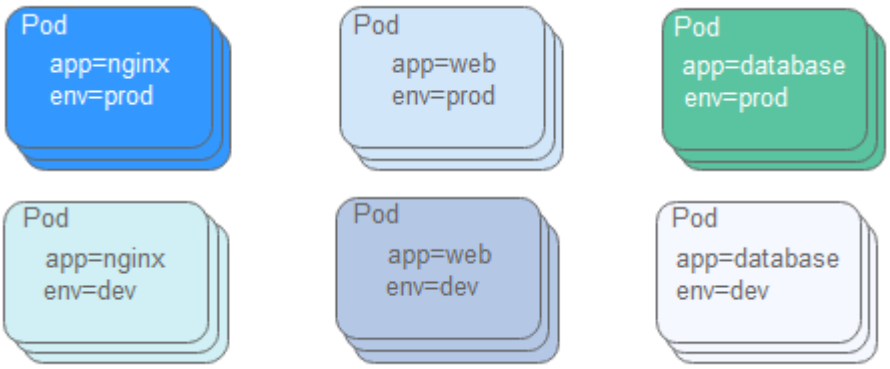
Labels are often used to select objects that meet conditions from a group of objects. Labels are currently the most important node grouping method in Kubernetes.

For example, you may create labels (**tier**=frontend, **app**=myapp) to mark frontend pods and labels (**tier**=backend, **app**=myapp) to mark backend pods. You can then

use selectors to select pods with specific labels and apply services or Deployments to these pods.

For details, see **Labels**.

Figure 4-2 Pods organized with labels



Deployment

Deployment is a type of pod controller.

A Deployment can contain one or more pods. Each pod has the same role, and the system automatically distributes requests to the pods of a Deployment. All pods for deploying a Deployment share storage volumes.

When using a Deployment, you only need to describe your desired pod status. The Deployment will help you change the current pod status to the target status.

For details, see **Deployments**.

Job

A job is a resource object that Kubernetes uses to control batch tasks. A job is different from a long-term servo workload (such as Deployment). The former is completed when a specified number of successful completions is reached, while the latter runs unceasingly if not terminated. The pods managed by a job automatically exit after successfully completing the job based on user configurations.

This run-to-completion feature of jobs is especially suitable for one-off tasks, such as continuous integration (CI). It works with the per-second billing of CCI to implement pay-per-use in real sense.

For details, see **Jobs**.

Cron Job

A cron job runs a job periodically on a specified schedule. A cron job object is similar to a line of a crontab file in Linux.

For details, see **CronJob**.

Service

Kubernetes pods are mortal. They can be created and destroyed. Once destroyed, they cannot be resurrected. Pod controllers create and destroy pods dynamically (for example, when scaling out or in or during rolling upgrades). Each pod obtains its own IP address, but the IP address is not always stable or dependable. This leads to a problem: if some set of pods (backends) provides services to other pods (frontends) inside a Kubernetes cluster, how do those frontends find out and connect to the corresponding backends?

A Kubernetes service (sometimes referred to as a microservice) defines a logical set of pods and a policy to access them. The set of pods targeted by a service is usually determined by a label selector.

Consider an image processing backend that is running with three pod replicas as an example. These replicas are interchangeable (frontends does not need to know which backend they call). The pods that form the backend set may change, and the frontends do not need to be aware of that or keep track of the list of backends themselves. A Kubernetes service enables this decoupling.

For details, see **Service**.

Ingress

Services and pods can be accessed only through an internal IP address. An external request needs to be forwarded by a load balancer to the NodePort exposed by the service on a node and then be forwarded by kube-proxy to corresponding pods.

An ingress is a set of rules that allow access from outside a cluster to services within the cluster. You can configure externally-accessible URLs, load balancers, SSL, and name-based virtual hosts for an ingress.

For details, see **Ingress**.

PVC

A PersistentVolumeClaim (PVC) is a request for storage by a user. Similar to a pod which requests CPU and memory, a PVC requests storage resources. On CCI, you can apply for storage resources such as EVS disks and SFS file systems using PVCs.

For details, see **Persistent Volumes**.

ConfigMap

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

For details, see **ConfigMaps**.

Secret

A secret is a Kubernetes object for storing sensitive data such as passwords, tokens, certificates, and private keys. A secret can be loaded to a container as environment variables when the container is started.

For details, see **Secrets**.

5 Security

5.1 Shared Responsibilities

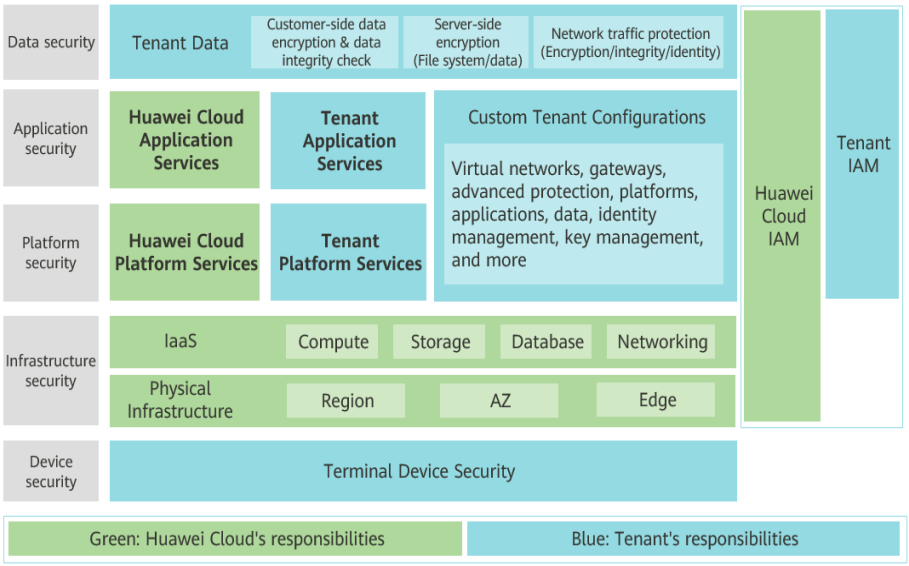
Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To cope with emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system for different regions and industries based on Huawei's unique software and hardware advantages, laws, regulations, industry standards, and security ecosystem.

Figure 5-1 illustrates the responsibilities shared by Huawei Cloud and users.

- **Huawei Cloud**: Ensure the security of cloud services and provide secure clouds. Huawei Cloud's security responsibilities include ensuring the security of our IaaS, PaaS, and SaaS services, as well as the physical environments of the Huawei Cloud data centers where our IaaS, PaaS, and SaaS services operate. Huawei Cloud is responsible for not only the security functions and performance of our infrastructure, cloud services, and technologies, but also for the overall cloud O&M security and, in the broader sense, the security and compliance of our infrastructure and services.
- **Tenant**: Use the cloud securely. Tenants of Huawei Cloud are responsible for the secure and effective management of the tenant-customized configurations of cloud services including IaaS, PaaS, and SaaS. This includes but is not limited to virtual networks, the OS of virtual machine hosts and guests, virtual firewalls, API Gateway, advanced security services, all types of cloud services, tenant data, identity accounts, and key management.

Huawei Cloud Security White Paper elaborates on the ideas and measures for building Huawei Cloud security, including cloud security strategies, the shared responsibility model, compliance and privacy, security organizations and personnel, infrastructure security, tenant service and security, engineering security, O&M security, and ecosystem security.

Figure 5-1 Huawei Cloud shared security responsibility model



5.2 Identity Authentication and Access Control

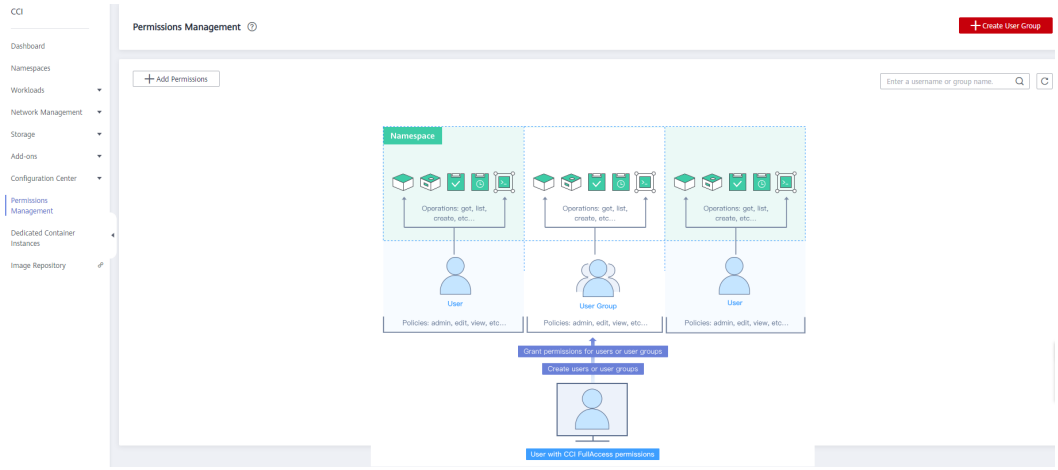
CCI permissions management allows you to grant permissions to your IAM users and user groups. It combines the advantages of Kubernetes Role-based Access Control (RBAC) authorization and Identity and Access Management (IAM) to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, namespace-level authorization, and namespaced resource authorization.

- **Namespace-level permissions**: permissions granted based on Kubernetes RBAC roles. You can authorize users or user groups to perform operations on Kubernetes resources under specific namespace.
- **CCI permissions:** permissions granted based on IAM fine-grained authorization. You can authorize users to perform operations on namespaces, such as creating and deleting namespaces.

NOTE

- If you enable RBAC when you create a namespace, access to resources under the namespace is controlled by RBAC policies. If RBAC is disabled, RBAC policies will not take effect.
- After you create a namespace with RBAC enabled, you must authorize IAM users to perform operations on the namespace.
- The network, ClusterRole, and RoleBinding resources are not affected by RBAC policies but are controlled only by IAM fine-grained authentication. The network resources are controlled by network-related actions, and ClusterRole and RoleBinding are controlled by RBAC-related actions.
- You can grant permissions for all namespaces of an IAM user at the same time.

Figure 5-2 CCI permissions management



Namespace Permissions

Kubernetes RBAC APIs define four objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding. Currently, CCI supports only ClusterRole and RoleBinding. The two objects are described as follows:

- **ClusterRole** specifies which actions can be performed on which resources. In the RBAC API, a role contains rules that represent a set of permissions. A role within a Kubernetes cluster is defined by a ClusterRole.
- **RoleBinding** binds roles to subjects (including users and user groups). A RoleBinding grants the permissions defined in a role to a user or user group. The user or group has the permissions granted through the bound ClusterRole.

Table 5-1 Two objects declared by the RBAC API

Type	Description
ClusterRole	A ClusterRole can be used to grant access to resources in a cluster.
RoleBinding	A RoleBinding binds a ClusterRole to subjects (users) in a namespace, granting the ClusterRole's permissions to those users.

CAUTION

Currently, you can only use ClusterRole to create a RoleBinding in a namespace.

In CCI, you can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

Currently, there are four roles: **cluster-admin**, **admin**, **edit**, and **view**. For details, see **Table 5-2**.

Table 5-2 User/user group roles

Default ClusterRole	Description
cluster-admin	Allows access to all Kubernetes resource objects.
admin	Allows admin access that can be granted within a namespace using a RoleBinding. If used in a RoleBinding, it allows read/write access to most resources in a namespace. It does not allow write access to resource quota or to the namespace itself.
edit	Allows read/write access to most resources in a namespace.
view	Allows read-only access to most objects in a namespace. It does not allow access to secrets.

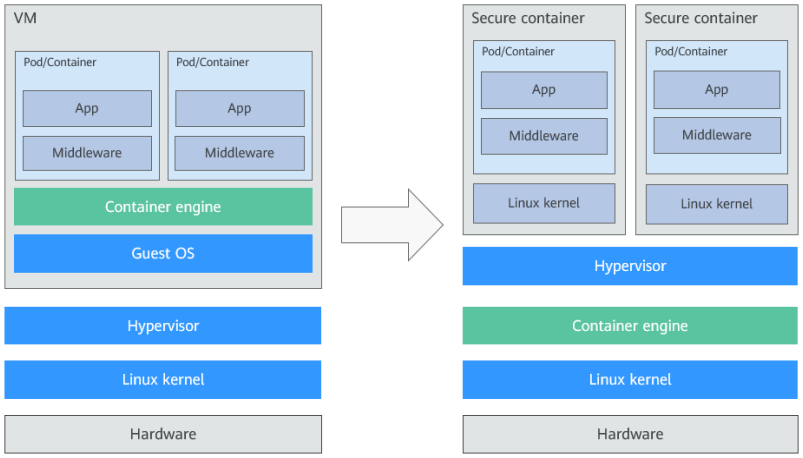
For more information about Kubernetes RBAC authorization, see **Using RBAC Authorization**.

5.3 Data Protection

CCI provides VM-level isolation without compromising the startup speed, offering you better container experience. CCI has the following security features:

- Secure containers are used.
- Kernel virtualization based on secure containers provides comprehensive security isolation and protection
- In-house virtualization acceleration technologies improve the performance of secure containers.

Figure 5-3 Hard multi-tenancy brought by secure containers



SSL

Secure Sockets Layer (SSL) is a protocol designed to protect security and data integrity for Internet communications.

You can upload an SSL certificate to CCI. In HTTPS access, CCI will automatically install it to the layer-7 load balancer for data transmission encryption. For details, see **SSL Certificates**.

Secret

A secret is a Kubernetes object for storing sensitive data such as passwords, tokens, certificates, and private keys. A secret can be loaded to a container as environment variables when the container is started.

For details, see **Secret**.

5.4 Audit and Logging

Audit

Cloud Trace Service (CTS) records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, track resource changes, audit compliance, and locate faults.

After you enable CTS and create and configure a tracker, CTS records cloud resource operation requests initiated from the CCI console or open APIs as well as responses to the requests.

For details about how to enable and configure CTS, see **Enabling CTS**.

For details about CCI operations that can be tracked by CTS, see **CCI Operations Supported by CTS**.

CCI records and reports audit logs to CTS for you to query and analyze. For details about configuration, see **Viewing Logs in CTS**.

Logging

CCI allows you to manage logs. You can configure the log path and reporting address of a container. The Fluent Bit integrated in a pod collects logs from the log path and reports the logs to LTS. For details, see **Log Management**.

The security log capability of CCI has been interconnected with CTS, CLS, LTS, and AOM. The security and integrity capabilities of CCI are traced by related services.

5.5 Security Risk Monitoring

Viewing Pod Monitoring Data on AOM

CCI works with AOM to monitor workloads, helping you obtain their running status.

You can monitor basic resources, applications, logs, and alarms about CCI on the AOM console.

For details, see **Monitoring Management**.

Pod Resource Monitoring Metrics

CCI supports basic monitoring of pod resources with multiple metrics, such as metrics for CPU, memory, disk, and network.

Pods have built-in system agents, which provide pod and container monitoring metrics in HTTP services by default.

- For details about the resource monitoring metrics supported by CCI, see **Resource Monitoring Metrics**.
- For details about basic pod resource monitoring, see **Pod Resource Monitoring Metrics**.

6 Constraints

This section describes the constraints on using CCI.

Constraints on CCI Instances

The following table lists the constraints on CCI instances.

Item	Constraint
Account for creating CCI instances	The account must have complete real-name authentication.
Resource quotas of a single account	Log in to the Huawei Cloud management console and choose **Resources** > **My Quotas** > **Quotas** to view the total quota and usage of each resource. **NOTE** If a quota cannot meet your business requirements, you can increase the quota. For more information on quotas, see **Quotas**.
Number of vCPUs per CCI instance	0.25 to 32 vCPUs, or 48 vCPUs or 64 vCPUs
Supported OS	Only Linux
Network type	Only VPC

Constraints on Kubernetes

For security purposes, CCI does not support Kubernetes functions listed in the following table.

Unavailable Function	Description	Recommended Alternative Solution
hostPath	Mounts a file on the local host to a pod.	EVS disks or SFS file systems

Unavailable Function	Description	Recommended Alternative Solution
hostNetwork	Maps the host port to a pod.	Load balancing (type=LoadBalancer)
DaemonSet	Ensures that there is only one copy of pod on each node.	Deploy multiple containers in a pod in the form of sidecar.
Privileged permission	Allows a container to have the privileged permission.	Use the security context to enable the privileged mode for the pod.
Service with type set to NodePort	Maps the host port to a pod.	Load balancing (type=LoadBalancer)

Constraints on Pod Specifications

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

The pod specifications you select must meet the following requirements.

Table 6-1 Requirements on pod specifications

Item	Value Range
Number of vCPUs	• 0.25 to 32 vCPUs, or 48 vCPUs or 64 vCPUs • The vCPUs of a single container must be an integer multiple of 0.25.
Memory	• 1 GiB to 512 GiB • The memory must be an integer multiple of 1 GiB.
vCPU/memory ratio	Between 1:2 and 1:8
Containers in a pod	A maximum of five containers A single container has at least 0.25 vCPUs and 0.2 GiB of memory. The maximum resource specification of a container is the same as that of a pod.
All containers in the pod and init containers	The request and limit of the two types of containers are the same.

NOTE

- GPU-accelerated pods are temporarily unavailable because GPU resources are insufficient.
- For more information, see Calculating Pod Specifications.
- Init containers are specialized containers that run before application containers startup in a pod. For details, see **Initializing a Container**.

Constraints on Pod Storage Space

If no EVS disk is mounted, application data is stored in the rootfs of the container. The following table lists the storage space limit of each type of pod.

Table 6-2 Storage space limit of each type of pod

Pod Type	Storage Space
General-computing pod	20 GB
GPU-related pod	20 GB

7 Billing

Billing Item

Instance resources include vCPU, memory, and GPU. You are charged based on the actual instance specifications you apply for and the actual running duration (by second) of your instance. The charging duration starts from the time when the container image is downloaded (docker pull) to the time when your CCI instance is stopped.

Billing Mode

CCI supports pay-per-use or package-based billing. For details, see **Product Pricing Details**.

Currently, CCI is billed on a pay-per-use basis.

NOTE

A core-hour indicates the number of vCPUs multiplied by time. For example, 730 core-hours mean that you can use 730 vCPUs for one hour or one vCPU for 730 hours.

- 1 core-hour = 1 x 3600 core-seconds
- 1 core-hour indicates that a vCPU is used for one hour.
- A core-second indicates that a vCPU is used for one second.

Pay-per-use billing

You will be charged by second for each instance and the billing statistics are presented by hour.

8 Permissions Management

If you need to assign different permissions to employees in your enterprise to access your CCI resources, Identity and Access Management (IAM) is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, enabling secure access to your cloud resources.

With IAM, you can use your account to create IAM users, and assign permissions to the users to control their access to specific resources. For example, some software developers in your enterprise need to use CCI resources but should not be allowed to delete the resources or perform any other high-risk operations. In this scenario, you can create IAM users for the software developers and grant them only the permissions required for using CCI resources.

If your account does not require individual IAM users for permissions management, skip this section.

IAM can be used free of charge. You pay only for the resources in your account. For more information, see the **IAM Service Overview**.

CCI Permissions

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.

CCI is a project-level service deployed and accessed in specific physical regions. To assign CCI 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 CCI, the users need to switch to a region where they have been authorized to use this 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 grant CCI users only the permissions for managing a certain type of CCI resources. Most policies define permissions based on APIs. For the API actions supported by CCI, see **Permissions Policies and Supported Actions**.

Table 8-1 lists all the system-defined roles and policies supported by CCI.

Table 8-1 System-defined roles and policies supported by CCI

Role/Policy Name	Description	Type
CCI FullAccess	Full permissions for CCI. Users granted these permissions can create, delete, query, and update all CCI resources.	System-defined policy
CCI ReadOnlyAccess	Read-only permissions for CCI. Users granted these permissions can only view CCI resources.	System-defined policy
CCI CommonOperations	Common user permissions for CCI. Users granted these permissions can perform all operations except creating, deleting, and modifying role-based access control (RBAC) policies, networks, and namespaced resources.	System-defined policy
CCI Administrator	Administrator permissions for CCI. Users granted these permissions can create, delete, query, and update all CCI resources.	System-defined role

Table 8-2 lists the permissions granted by a CCI FullAccess policy.

Table 8-2 Permissions granted by a CCI FullAccess policy

Action	Description
cci:*:*	Perform all operations on Cloud Container Instance (CCI).
vpc:*:*	Perform all operations on Virtual Private Cloud (VPC).
elb:*:*	Perform all operations on Elastic Load Balance (ELB).
sfs:*:*	Perform all operations on Scalable File Service (SFS).
evs:*:*	Perform all operations on Elastic Volume Service (EVS).

Action	Description
aom:*:*	Perform all operations on Application Operations Management (AOM).
apm:*:*	Perform all operations on Application Performance Management (APM).
swr:*:*	Perform all operations on Software Repository for Container (SWR).
nat:*:*	Perform all operations on NAT Gateway.
kms:cmk:*	Perform all operations on Data Encryption Workshop (DEW).

Table 8-3 lists the permissions granted by a CCI ReadOnlyAccess policy.

Table 8-3 Permissions granted by a CCI ReadOnlyAccess policy

Action	Description
cci:*:get	View details about all CCI resources.
cci:*:list	List all CCI resources.
vpc:*:get	View details about all VPC resources.
vpc:*:list	List all VPC resources.
ecs:*:get	View details about all ECS resources.
ecs:*:list	List all ECS resources.
elb:*:get	View details about all ELB resources.
elb:*:list	List all ELB resources.
sfs:*:get*	View details about all SFS resources.
sfs:*:list	List all SFS resources.
evs:*:get*	View details about all EVS resources.
evs:*:list	List all EVS resources.
aom:*:get	View details about all AOM resources.
aom:*:list	List all AOM resources.
amp:*:get	View details about all APM resources.
apm:*:list	List all APM resources.
swr:*:get	View details about all SWR resources.
swr:*:list	List all SWR resources.

Action	Description
nat:*:get	View details about all NAT Gateway resources.
nat:*:list	List all NAT Gateway resources.
kms:cmk:get	Query key information.
kms:cmk:list	List all keys.

Table 8-4 lists the permissions granted by a CCI CommonOperations policy.

Table 8-4 Permissions granted by a CCI CommonOperations policy

Action	Description
cci:rbac:get	Query RBAC policy details.
cci:rbac:list	List all RBAC policies.
cci:namespace:get	Query namespace details.
cci:namespace:list	List all namespaces.
cci:network:get	Query network details.
cci:network:list	List all networks.
cci:namespaceSubResource:*	Perform all operations on namespaced resources.
cci:addonTemplate:*	Perform all operations on add-on templates.
cci:addonInstance:*	Perform all operations on add-on instances.
vpc:*:*	Perform all operations on VPC.
elb:*:*	Perform all operations on ELB.
evs:*:*	Perform all operations on EVS.
aom:*:*	Perform all operations on AOM.
apm:*:*	Perform all operations on APM.
swr:*:*	Perform all operations on SWR.
nat:*:*	Perform all operations on NAT Gateway.
kms:cmk:*	Perform all operations on DEW.

Table 8-5 lists the actions associated with CCI fine-grained policies.

Table 8-5 Actions associated with CCI fine-grained policies

Action	Description
CCI:rbac:get	Query RBAC details.
CCI:rbac:list	List all RBAC policies.
CCI:rbac:update	Update RBAC policies.
CCI:rbac:delete	Delete RBAC policies.
CCI:rbac:create	Create RBAC policies.
CCI:namespaceSubResource:Create	Create resources in namespaces.
CCI:namespaceSubResource:List	List all Kubernetes resources.
CCI:namespaceSubResource:Get	Query Kubernetes resources.
CCI:namespaceSubResource:Delete	Delete Kubernetes resources.
CCI:namespaceSubResource:Update	Update Kubernetes resources.
CCI:network:update	Update networks.
CCI:network:create	Create networks.
CCI:network:delete	Delete networks.
CCI:network:list	List all networks.
CCI:network:get	Query network details.
CCI:addonInstance:create	Create add-on instances.
CCI:addonInstance:update	Update add-on instances.
CCI:addonInstance:delete	Delete add-on instances.
CCI:addonInstance:get	Query add-on instance details.
CCI:addonInstance:list	List all add-on instances.
CCI:addonTemplate:list	List all add-on templates.
CCI:addonTemplate:get	Query add-on template details.
CCI:namespace:get	Query details about a specified namespace.

Action	Description
CCI:namespace:update	Update namespaces.
CCI:namespace:create	Create namespaces.
CCI:namespace:list	List all namespaces.
CCI:namespace:delete	Delete namespaces.

Table 8-6 lists the common operations supported by each system-defined policy or role of CCI. Select the policies or roles as required.

Table 8-6 Common operations supported by each system-defined policy or role of CCI

Operation	CCI FullAccess	CCI ReadOnlyAccess	CCI CommonOperations
Creating Deployments	√	x	√
Deleting Deployments	√	x	√
Viewing Deployments	√	√	√
Upgrading workloads	√	x	√
Scaling workloads	√	x	√
Deleting pods	√	x	√
Viewing pods	√	√	√
Creating jobs	√	x	√
Deleting jobs	√	x	√
Viewing jobs	√	√	√
Creating cron jobs	√	x	√
Deleting cron jobs	√	x	√
Viewing cron jobs	√	√	√
Viewing the resource usage	√	√	√
Adding EVS volumes	√	x	√

Operation	CCI FullAccess	CCI ReadOnlyAccess	CCI CommonOperations
Deleting EVS volumes	√	x	√
Viewing EVS volumes	√	√	√
Creating ConfigMaps	√	x	√
Deleting ConfigMaps	√	x	√
Viewing ConfigMaps	√	√	√
Creating secrets	√	x	√
Deleting secrets	√	x	√
Viewing secrets	√	√	√
Adding SSL certificates	√	x	√
Deleting SSL certificates	√	x	√
Viewing SSL certificates	√	√	√
Adding log storage	√	x	√
Viewing logs	√	√	√
Installing add-ons	√	x	√
Deleting add-ons	√	x	√
Viewing add-ons	√	√	√
Viewing permissions	√	√	√
Granting permissions	√	x	x
Deleting permissions	√	x	x
Querying details about a specified namespace	√	x	√
Creating namespaces	√	x	x
Deleting namespaces	√	x	x
Creating networks	√	x	x
Deleting networks	√	x	x
Listing all networks	√	√	√

Operation	CCI FullAccess	CCI ReadOnlyAccess	CCI CommonOperations
Querying network details	√	√	√

Helpful Links

- IAM Service Overview
- Creating a User and Granting CCI Permissions
- Permissions Policies and Supported Actions

9 Region and AZ

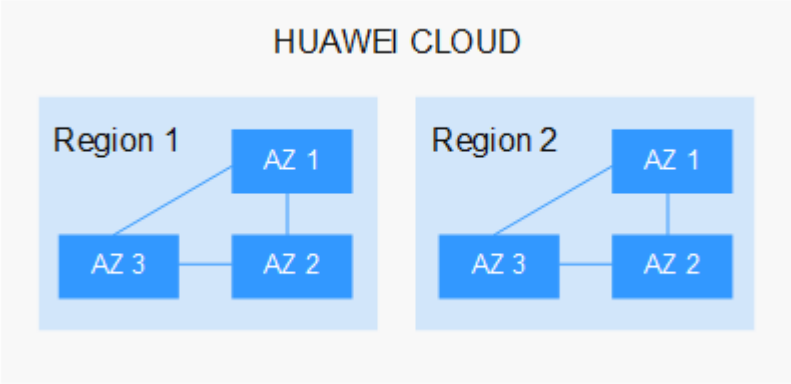
Concept

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 into universal regions and dedicated regions. A universal region provides universal cloud services for common tenants. A dedicated region provides specific services for specific tenants.
- 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 within a region are interconnected using high-speed optical fibers, to support cross-AZ high-availability systems.

Figure 9-1 shows the relationship between regions and AZs.

Figure 9-1 Regions and AZs



Huawei Cloud provides services in many regions around the world. You can select a region and an AZ based on requirements. For more information, see **Huawei Cloud Global Regions**.

Selecting a Region

When selecting a region, consider the following factors:

- Location

  It is recommended that you select the closest region for lower network latency and quick access.

- 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.

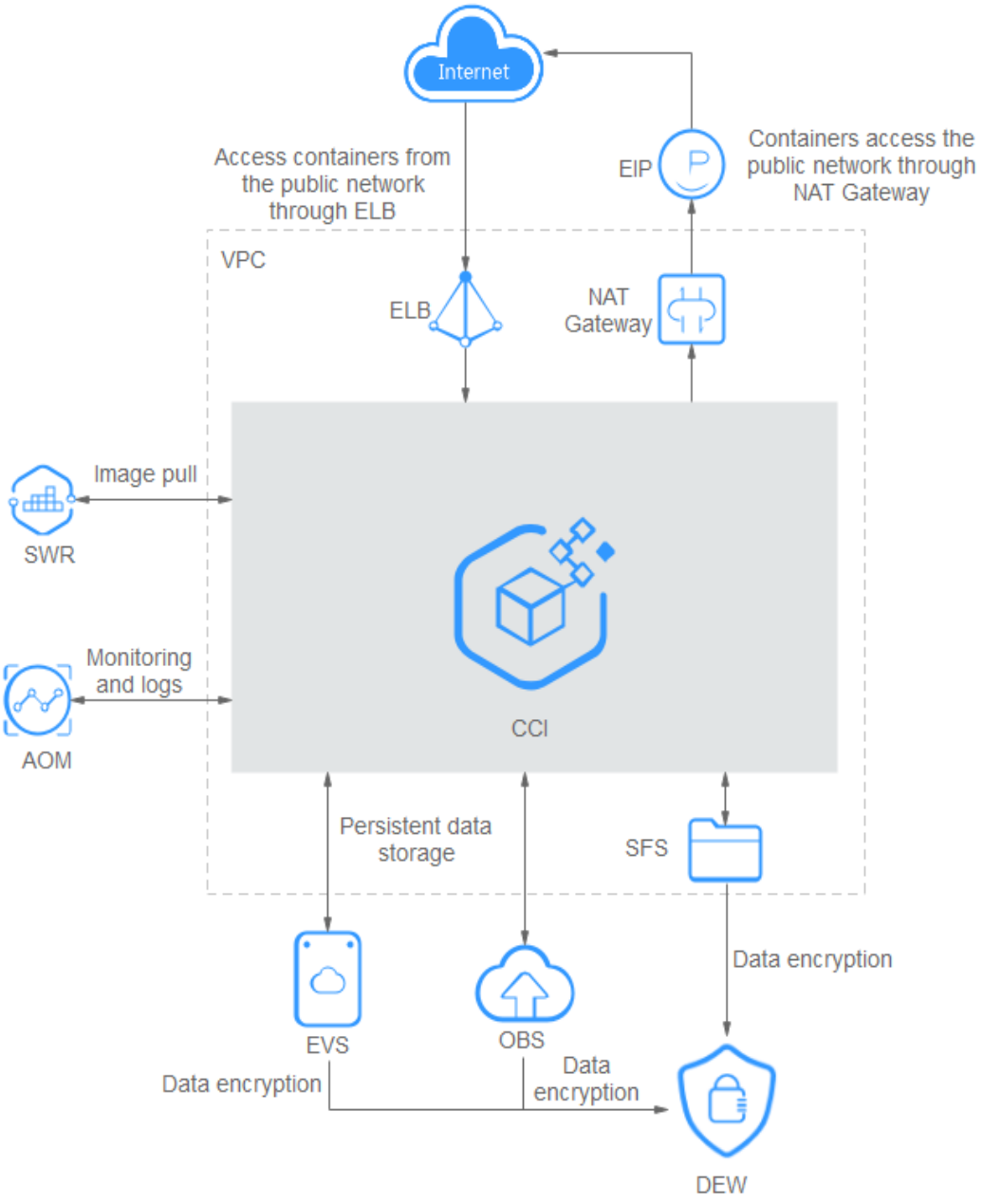
Regions and Endpoints

Before you use an API to call resources, specify its region and endpoint. For more details, see **Regions and Endpoints**.

10 Related Services

CCI needs to be interconnected with other cloud services. It requires permissions to access the following cloud services.

Figure 10-1 Relationships between CCI and other services



- **SWR**

  Software Repository for Container (SWR) provides easy, secure, and reliable management of container images throughout their lifecycles, facilitating quick deployment of containerized services.

  You can create workloads using SWR images.

- **VPC**

  A Virtual Private Cloud (VPC) is a private and isolated virtual network on the cloud platform. You can configure the CIDR block, subnets, and security groups, assign EIPs, and allocate bandwidth for a VPC.

  When creating a namespace, you must associate it with a VPC. All containers to be created in the namespace will run in the VPC.

- **ELB**

  Elastic Load Balance (ELB) automatically distributes access traffic to multiple Elastic Cloud Servers (ECSs) to balance the loads. It enhances an application's fault tolerance level and capabilities of providing service externally.

  You can access a container workload from an external network through an elastic load balancer.

- **AOM**

  Application Operations Management (AOM) is a one-stop platform for O&M personnel to monitor application and resource operating statuses in real time. By analyzing dozens of metrics, alarms, and logs, you can quickly locate root causes to ensure smooth running of services.

  AOM collects the .log files of containers from CCI to facilitate log query and viewing. In addition, AOM provides resource monitoring to enable CCI to automatically scale resources.

- **EVS**

  Elastic Volume Service (EVS) provides persistent block storage services. With data redundancy and cache acceleration technologies, EVS disks can deliver high-reliability, durable, stable performance with low latency. You can format an EVS disk, create a file system, and store data persistently.

  You can use an EVS disk as persistent storage for a container and mount the EVS disk to the container when creating a workload.

- **ECS**

  Elastic Cloud Server (ECS) allows you to create cloud servers that provide scalable, on-demand computing resources for secure, flexible, and efficient applications.

  CCI imports data to SFS through ECS for container services to use.

- **NAT Gateway**

  The NAT Gateway service provides source network address translation (SNAT), which translates private IP addresses to a public IP address by binding an elastic IP address (EIP) to the gateway.

  You can use NAT Gateway to set SNAT rules to allow containers in a VPC to access the Internet.

- **Data Encryption Workshop (DEW)**

  DEW is a cloud data encryption service. It covers Key Management Service (KMS), Key Pair Service (KPS), and Dedicated Hardware Security Module (Dedicated HSM). DEW uses HSMs to secure your keys, and can be integrated with other cloud services to meet even the most demanding scenarios. Additionally, DEW enables you to develop customized encryption applications.

