Cloud Search Service

Overview

Issue 01

Date 2025-01-23

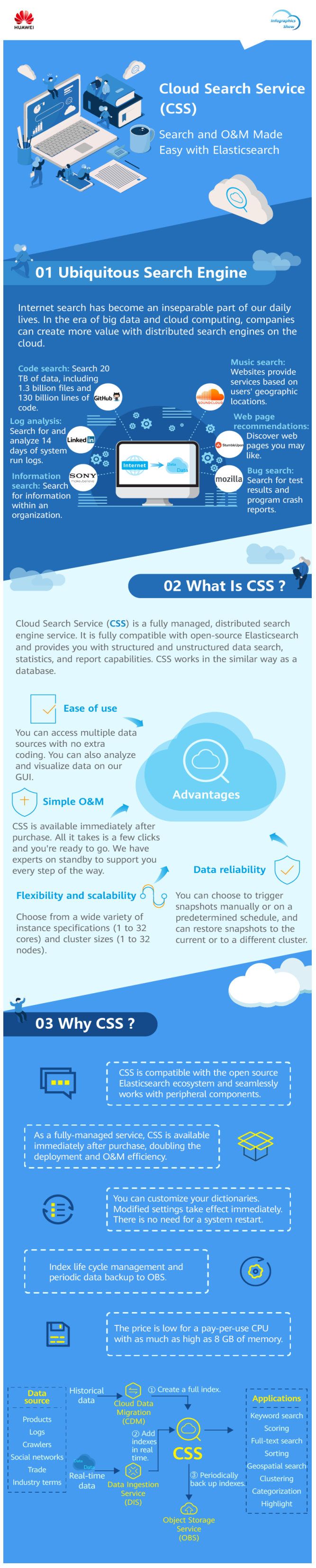
HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 CSS Infographics



Cloud Search Service (CSS)

Search and O&M Made Easy with Elasticsearch

01 Ubiquitous Search Engine

Internet search has become an inseparable part of our daily lives. In the era of big data and cloud computing, companies can create more value with distributed search engines on the cloud.

Code search: Search 20 TB of data, including 1.3 billion files and 130 billion lines of code.

Log analysis: Search for and analyze 14 days of system run logs.

Information search: Search for information within an organization.

Music search: Websites provide services based on users' geographic locations.

Web page recommendations: Discover web pages you may like.

Bug search: Search for test results and program crash reports.

02 What Is CSS ?

Cloud Search Service (**CSS**) is a fully managed, distributed search engine service. It is fully compatible with open-source Elasticsearch and provides you with structured and unstructured data search, statistics, and report capabilities. CSS works in the similar way as a database.

Advantages

Ease of use

You can access multiple data sources with no extra coding. You can also analyze and visualize data on our GUI.

Simple O&M

CSS is available immediately after purchase. All it takes is a few clicks and you're ready to go. We have experts on standby to support you every step of the way.

Flexibility and scalability

Choose from a wide variety of instance specifications (1 to 32 cores) and cluster sizes (1 to 32 nodes).

Data reliability

You can choose to trigger snapshots manually or on a predetermined schedule, and can restore snapshots to the current or to a different cluster.

03 Why CSS ?

CSS is compatible with the open source Elasticsearch ecosystem and seamlessly works with peripheral components.

As a fully-managed service, CSS is available immediately after purchase, doubling the deployment and O&M efficiency.

You can customize your dictionaries. Modified settings take effect immediately. There is no need for a system restart.

Index life cycle management and periodic data backup to OBS.

The price is low for a pay-per-use CPU with as much as high as 8 GB of memory.

2 What Is Cloud Search Service?

CSS

CSS is a fully managed, distributed search service based on open source Elasticsearch and OpenSearch. You can use it for structured and unstructured data search, and enable vector-based composite search, statistics generation, and reporting. CSS is a fully hosted cloud service of the ELK Stack and is compatible with open-source Elasticsearch, Logstash, Kibana, and Cerebro.

- Elasticsearch and OpenSearch

  Elasticsearch and OpenSearch are open-source distributed search engines that can be deployed in standalone or cluster mode. As the heart of the ELK Stack, Elasticsearch clusters support multi-condition search, statistical analysis, and create visualized reports of structured and unstructured text. For details about Elasticsearch, see the **Elasticsearch: The Definitive Guide**. For details about the OpenSearch search engine, see **OpenSearch Documentation**.

  CSS can be automatically deployed, where you can quickly create Elasticsearch and OpenSearch clusters. It provides the search engine optimization practices with zero O&M. Additionally, it has a robust monitoring system to present you key metrics, including clusters and query performance so that you can focus on the business logic.

- Logstash

  Logstash is an open-source data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to your desired destination.

  Huawei Cloud Logstash is a fully managed data ingestion and processing service that is completely compatible with open-source Logstash. You can quickly create Logstash clusters in CSS. Data is scattered across many different systems in different formats. Huawei Cloud Logstash helps you get insights by easily processing data from a variety of data sources and dumping it to Elasticsearch clusters or other systems.

Functions

- Open-source compatibility

  Freely use native Elasticsearch and OpenSearch APIs and other software in the ecosystem, such as Logstash, Beats, and Kibana.

- Support for a variety of data sources

  A few simple configurations allow you to smoothly connect to multiple data sources, such as FTP, OBS, HBase, and Kafka. No extra coding is required.
- One-click operation

  One-click cluster application, capacity expansion, and restart from small-scale testing to large-scale rollout
- Flexible dictionary management

  You can custom your dictionaries. Modified settings take effect immediately without system restart.
- User-defined snapshot policies

  Trigger backup snapshots manually or configure an automated schedule.

Access Mode

You can access the public cloud platform with HTTPS-based application programming interfaces (APIs) or from a web-based console.

- Using APIs

  If you need to integrate CSS into a third-party system for secondary development, you can use APIs to access CSS. For details, see the ***Cloud Search Service API Reference***.
- Web-based console

  If no secondary development is involved, the CSS management console is a convenient way to access CSS. If you have registered with the public cloud, log in to the management console and search for **Cloud Search Service** in the service list. If you do not sign up for the public cloud, click **Register**. In the displayed window, specify your basic information and click **Register**.

3 Advantages

CSS has the following features and advantages.

Efficient and Ease of Use

You can get insights from terabyte-scale data in milliseconds. In addition, you can use the visualized platform for data display and analysis.

Flexible and Scalable

You can request resources as needed and perform capacity expansion online with zero service interruption.

Easy O&M

CSS is a fully-managed, out-of-the-box service. You can start using it with several clicks, instead of managing clusters.

Kernel Enhancement

- **Vector search**

  When you search for unstructured data, such as images, videos, and corpuses, the nearest neighbors or approximate nearest neighbors are searched based on feature vectors. For details, see **Vector Search**.
- **Decoupled storage and compute**

  CSS provides an API for freezing indexes. Hot data stored on SSD can be dumped to OBS to reduce data storage costs and decouple compute from storage. For details, see **Storage-Compute Decoupling**.
- **Flow control**

  CSS can control traffic at the node level. You can configure the blacklist and whitelist, the maximum concurrent HTTPS connections, and the maximum HTTP connections for a node. Each function has an independent control switch. For details, see **Flow Control**.
- **Large query isolation**

  CSS allows you to separately manage large queries. You can isolate query requests that consume a large amount of memory or take a long period of time. For details, see **Large Query Isolation**.

- **Index monitoring**

  CSS monitors various metrics of the running status and change trend of cluster indexes to measure service usage and handle potential risks in a timely manner, ensuring that clusters can run stably. For details, see **Index Monitoring**.
- **Enhanced monitoring**

  CSS supports enhanced cluster monitoring. It can monitor the P99 latency of cluster search requests and the HTTP status codes of clusters. For details, see **Enhanced Monitoring**.

High Reliability

You can choose to trigger snapshots manually or on a periodic basis for backup and restore snapshots to the current or other clusters. Snapshots of a cluster can be restored to another cluster to implement cluster data migration. For details, see **Index Backup and Restoration**.

- Automatic backup using snapshots

  CSS provides the backup function. You can enable the automatic backup function on the CSS management console and set the backup period based on the actual requirements.

  Automatic backup is to back up the index data of a cluster. Index backup is implemented by creating cluster snapshots. For backup of the first time, you are advised to back up all index data.

  CSS allows you to store the snapshot data of Elasticsearch instances to OBS, thereby achieving cross-region backup with the cross-region replication function of OBS.
- Restoring data using snapshots

  If data loss occurs or you want to retrieve data of a certain period, click **Restore** in the **Operation** column in the **Snapshots** area to restore the backup index data to the specified cluster by using existing snapshots.

High Security

CSS uses network isolation in addition to various host and data security measures.

- Network isolation

  The network is divided into two planes, service plane and management plane. The two planes are deployed and isolated physically to ensure the security of the service and management networks.
  - Service plane: refers to the network plane of the cluster. It provides service channels for users and delivers data definition, index, and search capabilities.
  - Management plane: This is mainly the management console, where you manage CSS.
  - VPC security groups or isolated networks ensure the security of hosts.
- Access control
  - Using the network access control list (ACL), you can permit or deny the network traffic entering and exiting the subnets.

  - Internal security infrastructure (including the network firewall, intrusion detection system, and protection system) can monitor all network traffic that enters or exits the VPC through the IPsec VPN.
  - User authentication and index-level authentication are supported. CSS also supports interconnection with third-party user management systems.
- Data security
  - In CSS, a multi-replica mechanism is used to ensure data security.
  - Communication between the client and server can be encrypted using SSL.
- Operation audit

  Cloud Trace Service (CTS) can be used to perform auditing on key logs and operations.

High Availability

To prevent data loss and minimize the cluster downtime in case of service interruption, CSS supports cross-AZ cluster deployment. When creating a cluster, you can select two or three AZs in the same region. The system will automatically allocate nodes to these AZs. If an AZ is faulty, the remaining AZs can still run properly, significantly enhancing cluster availability and improving service stability. For more information, see **Deploying a Cross-AZ Cluster**.

4 Product Components

CSS supports Kibana and Cerebro.

Kibana

Kibana is an open-source data analytics and visualization platform that works with Elasticsearch. You can use Kibana to search for and view data stored in Elasticsearch indexes and display data in charts and maps. For details about Kibana, visit **https://www.elastic.co/guide/en/kibana/current/index.html**.

By default, the Elasticsearch cluster of CSS provides the access channel to Kibana. You can quickly access Kibana without installing it. CSS is compatible with Kibana visualizations and Elasticsearch statistical and analysis capabilities.

- Over 10 data presentation modes
- Nearly 20 data statistics methods
- Classification in various dimensions, such as time and tag

Cerebro

Cerebro is an open-source Elasticsearch web admin tool built using Scala, Play Framework, AngularJS, and Bootstrap. Cerebro allows you to manage clusters on a visualized page, such as executing REST requests, modifying Elasticsearch configurations, monitoring real-time disks, cluster loads, and memory usage.

By default, the Elasticsearch cluster of CSS provides the access channel to Cerebro. You can quickly access Cerebro without installing it. CSS is fully compatible with the open-source Cerebro and adapts to the latest 0.8.4 version.

- Elasticsearch visualized and real-time load monitoring
- Elasticsearch visualized data management

5 Scenarios

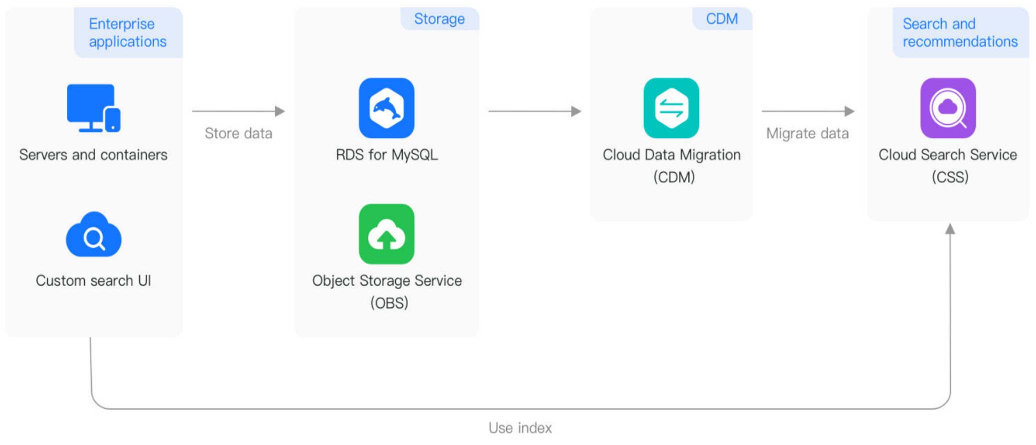
CSS can be used to build search boxes for websites and apps to improve user experience. You can also build a log analysis platform with it, facilitating data-driven O&M and business operations. CSS vector search can help you quickly build smart applications, such as AI-based image search, recommendation, and semantic search.

Site Search

CSS can be used to search for website content by keyword as well as search for and recommend commodities on e-commerce sites.

- Real-time search: When site content is updated, you can find the updated content in your search within minutes, or even just seconds.
- Categorized statistics: You can apply search filters to sort products by category.
- Custom highlight style: You can define how the search results are highlighted.

Figure 5-1 Site search

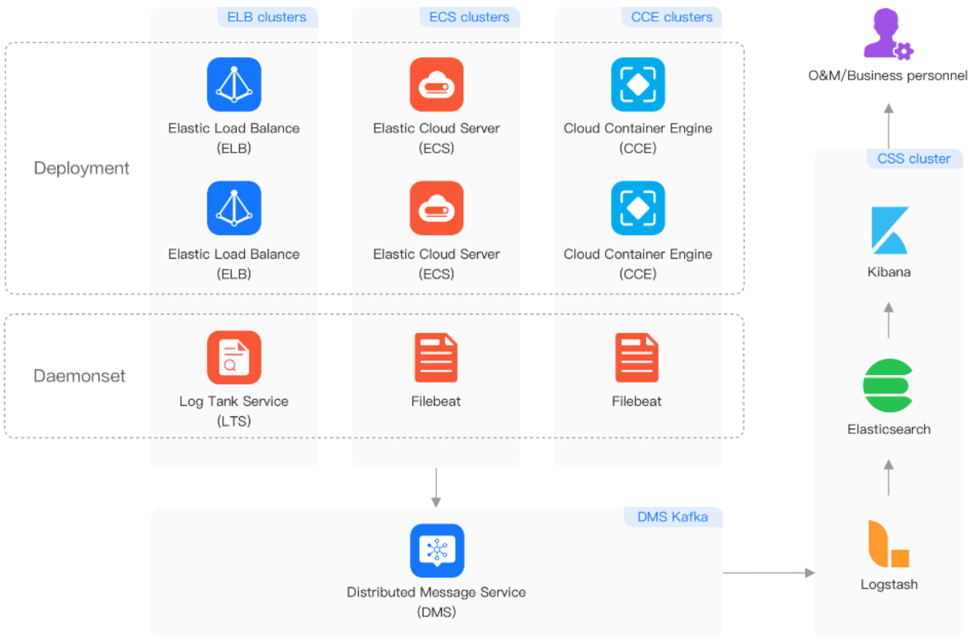


All-Scenario Log Analysis

Analyze the logs of Elastic Load Balance (ELB), servers, containers, and applications. In CSS, the Kafka message buffer queue is used to balance loads in peak and off-peak hours. Logstash is used for data extract, transform and load (ETL). Elasticsearch retrieves and analyzes data. The analysis results are visualized by Kibana and presented to you.

- High cost-effectiveness: CSS uses the Kunpeng computing power, separates cold and hot storage, and decouples computing and storage resources, achieving high performance and reducing costs by over 30%.
- Ease of use: Perform queries in a GUI editor. Easily create reports using drag-and-drop components.
- Powerful processing capability: CSS can import hundreds of terabytes of data per day, and can process petabytes of data.

Figure 5-2 All-scenario log analysis

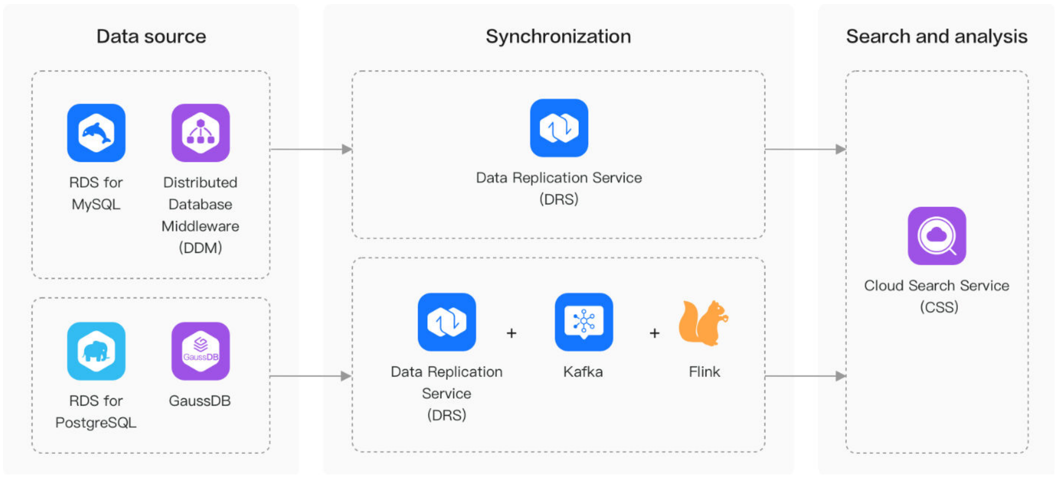

ELB clusters
ECS clusters
CCE clusters
O&M/Business personnel
Deployment
Elastic Load Balance (ELB)
Elastic Cloud Server (ECS)
Cloud Container Engine (CCE)
Elastic Load Balance (ELB)
Elastic Cloud Server (ECS)
Cloud Container Engine (CCE)
CSS cluster
Kibana
Daemonset
Log Tank Service (LTS)
Filebeat
Filebeat
Elasticsearch
DMS Kafka
Distributed Message Service (DMS)
Logstash

Database Query Acceleration

CSS can be used to accelerate database queries. E-commerce and logistics companies have to respond to a huge number of concurrent order queries within a short period of time. Relational databases, although having good transaction atomicity, are weak in transaction processing, and can rely on CSS to enhance OLTP and OLAP capabilities.

- High performance: Retrieve data from hundreds of millions of records within milliseconds. Text, time, numeric, and spatial data types are supported.
- High scalability: CSS can be scaled to have over 200 data nodes and over 1000 columns.
- Zero service interruption: The rolling restart and dual-copy mechanisms can avoid service interruption in case of specifications change or configuration update.

Figure 5-3 Database query acceleration

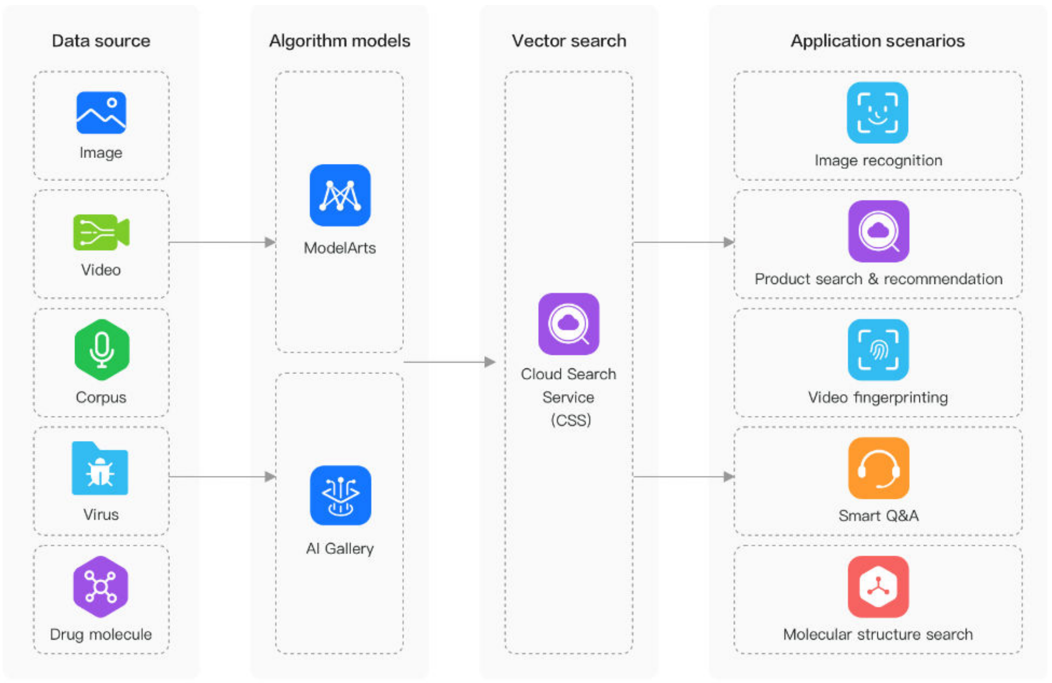

Data source
Synchronization
Search and analysis
RDS for MySQL
Distributed Database Middleware (DDM)
Data Replication Service (DRS)
RDS for PostgreSQL
GaussDB
Data Replication Service (DRS)
+
Kafka
+
Flink
Cloud Search Service (CSS)

Vector Search

When you search for unstructured data, such as images, videos, and corpuses, the nearest neighbors or approximate nearest neighbors are searched based on feature vectors. This has the following advantages:

- Efficiency and reliability: The Huawei Cloud vector search engine provides ultimate search performance and distributed disaster recovery capabilities.
- Abundant indexes: Multiple indexing algorithms and similarity measurement methods are available and can meet diverse needs.
- Easy learning: CSS is fully compatible with the open-source Elasticsearch ecosystem.

Figure 5-4 Vector search



6 Billing

CSS supports two billing modes: pay-per-use and yearly/monthly. The latter is more cost-effective. For pricing details, see **CSS Price Calculator**.

Billing Items

CSS bills you for your selected instance specifications and usage duration.

Table 6-1 CSS billing

Billing Item	Description
Node Specifications	Specify the instance type and specifications (vCPUs and memory), required duration, and the number of purchased instances. You can purchase node instances in the pay-per-use or yearly/monthly mode.
Node storage	Specifies the disk type. You can select disks of a type based on your business scenario. The billing standard varies depending on the disk type. Billing mode: pay-per-use or yearly/monthly. CSS provides the following types of disks: ● **Common I/O** ● **High I/O** ● **Ultra-High I/O**
Bandwidth	Specifies the bandwidth. When you enable the public IP address function or Kibana public IP address for a cluster, you will be billed for the bandwidth. Billing mode: pay-per-use or yearly/monthly. CSS provides the following types of bandwidth: ● **Low bandwidth** (1 to 5 Mbit/s) ● **High bandwidth** (6 to 2,000 Mbit/s) The billing automatically falls into low or high bandwidth based on the bandwidth you select.

Billing Modes

- Pay-per-use

  In pay-per-use mode, you are billed for a full hour even though you use CSS for less than 1 hour. You can enable or disable CSS as you like. This mode is suitable if you want more flexibility and control on resource usage.
- Yearly/Monthly billing

  In yearly/monthly mode, you need to pay for the service duration you selected in one-off mode. The service duration range is one month to three years. This mode provides a larger discount than pay-per-use and is recommended for long-term users.

Changing the Billing Mode

- Change the billing mode

  Changing from pay-per-use to yearly/monthly: After the billing mode is changed from pay-per-use to yearly/monthly, a new order is then generated for you, and the new billing mode takes effect immediately after you pay for the order.

  Changing from yearly/monthly to pay-per-use: The pay-per-use billing mode will take effect after the original yearly/monthly subscription expires.
- Change the node storage and quantity

  If the existing cluster adopts the pay-per-use billing mode, the modified nodes and node storage in the cluster will also be billed in the pay-per-use mode by default.

  For the modified nodes and storage capacity in a yearly/monthly cluster, their billing period starts from their provisioning time and ends with the yearly/monthly subscription of the cluster. For example, assume that your cluster is billed on a monthly basis and is scaled out on January 20. If the monthly subscription renews on January 30, you will have to pay for the fees generated by the new resources from January 20 to January 30.
- Change node specifications

  After node specifications are modified, nodes are billed based on the new specifications.
- Change bandwidth

  If you change the Internet access bandwidth for a cluster or Kibana, you will be billed based on the new bandwidth.

Renewal

You can renew a resource package upon its expiration, or you can set auto-renewal rules for a resource package. For more information about renewing resource packages, see **Renewal Management**.

Expiration and Overdue Payment

If your account is in arrears, you can view the arrears details in the Billing Center. To prevent related resources from being stopped or released, top up your account in a timely manner. If your account is in arrears, top up your account within the specified period. For details, see **Topping Up an Account**.

7 Differences Between Elasticsearch Cluster Versions

Table 7-1 Features of different Elasticsearch cluster versions

Version Feature	**5.*X* Version**	**6.*X* Version**	**7.*X* Version**
Supported types	An index can contain multiple types. The name of each type can be customized.	An index can contain only one type, and the type name can be customized.	An index can contain only one type. The type name and **_doc** are fixed.
Client access	TransportClient is supported. TCP and HTTP can be used for connection requests at the same time.	TransportClient is supported. TCP and HTTP can be used for connection requests at the same time. Java High Level REST Client is recommended.	Only RestClient is supported. Only HTTP can be used for connection requests. Java High Level REST Client is recommended.

Version Feature	5. *X* Version	6. *X* Version	7. *X* Version

	The following is an example of using TransportClient to access an Elasticsearch cluster:
	`//Initialize the client and connect to port 9300.`
	`TransportClient client = new PreBuiltTransportClient(Settings.EMPTY)`
	`        .addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName("host1"), 9300))`
	`        .addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName("host2"), 9300));`
	`//Close the client.`
	`client.close();`
		The following is an example of using Java High Level REST Client to access a cluster:
		`//Initialize the client and connect to port 9200.`
		`RestHighLevelClient client = new RestHighLevelClient(`
		`        RestClient.builder(`
		`                new HttpHost("localhost", 9200, "http"),`
		`                new HttpHost("localhost", 9201, "http")));`
		`//Close the client.`
		`client.close();`	

Template configuration	The **template** field is used to create a template.
	Example:
	`PUT _template/template_1`
	`{`
	`  "template": "te*",`
	`  "settings": {`
	`    "number_of_shards": 1`
	`  }`
	`}`
		The **index_pattern** field is used to create a template.
		Example:
		`PUT _template/template_1`
		`{`
		`  "index_patterns": ["te*"],`
		`  "settings": {`
		`    "number_of_shards": 1`
		`  }`
		`}`	

Boolean type parsing	In Elasticsearch 5.x, **true**, **false**, **on**, **off**, **yes**, **no**, **0**, and **1** can all be parsed as Boolean values.	Only **true** and **false** are supported. If other values are used, an error occurs.
		Only in Elasticsearch 6.*x* or 7.*x*, errors will be reported for the following statements:
		`GET data1/_search`
		`{`
		`  "profile": "noprofile",`
		`  "query": {`
		`    "match_all": {}`
		`  }`
		`}`	

Version Feature	5. *X* Version	6. *X* Version	7. *X* Version

JSON format verification	Duplicate keys are allowed in JSON and will be automatically deleted in the background.	Duplicate keys are not allowed in JSON. Otherwise, a parsing error is reported.
		Only in Elasticsearch 6.*x* or 7.*x*, errors will be reported for the following statements:
		`POST data1/doc`
		`{`
		`  "isl": 0,`
		`  "isl": 1`
		`}`	

DELETE document	If **index1** does not exist and you run the **DELETE index1/doc/1** command, the system will create **index1**.	If you run a command to delete an index that does not exist, an error message is displayed.	
_alias API validation	The **index** field in the _alias API can be specified as an alias and can be parsed properly. You can also use an alias to delete an index.	The **index** field in the _alias API can only be specified as an index name and cannot be an alias. To delete an index, the index name is required.	

Version Feature	5. *X* Version	6. *X* Version	7. *X* Version

	The following command can run properly in Elasticsearch 5.*x*, but an error is reported when you run it in Elasticsearch 6.*x* or 7.*x*.
	`PUT log-2023.11.11`
	`POST _aliases`
	`{`
	`  "actions": [`
	`    {`
	`      "add": {`
	`        "index": "log-2023.11.11",`
	`        "alias": "log"`
	`      }`
	`    }`
	`  ]`
	`}`
	`POST _aliases`
	`{`
	`  "actions": [`
	`    {`
	`      "remove": {`
	`        "index": "log",`
	`        "alias": "log"`
	`      }`
	`    }`
	`  ]`
	`}`
	Error message:
	`{`
	`  "error" : {`
	`    "root_cause" : [`
	`      {`
	`        "type" : "illegal_argument_exception",`
	`        "reason" : "The provided expression [log] matches an alias, specify the corresponding concrete indices instead."`
	`      }`
	`    ],`
	`    "type" : "illegal_argument_exception",`
	`    "reason" : "The provided expression [log] matches an alias, specify the corresponding concrete indices instead."`
	`  },`
	`  "status" : 400`
	`}`		

Default configurations	The default number of shards for a new index: 5		The default number of shards for a new index: 1

Version Feature	5. *X* Version	6. *X* Version	7. *X* Version

Default routing	In Elasticsearch 5.x/6.x, the following formula is used to calculate the shard where the document should be located:
	`shard_num = hash(_routing) % num_of_primary_shards`
			In Elasticsearch 7.x, the following formula is used to calculate the shard where the document should be located:
			`routing_factor = num_routing_shards / num_primary_shards`
			`shard_num = (hash(_routing) % num_routing_shards) / routing_factor`
			The following command can be used to specify the value of **num_routing_shards**:
			`index.number_of_routing_shards`
			If this parameter is not explicitly specified, Elasticsearch automatically calculates the value to split indexes.

Version Feature	5. *X* Version	6. *X* Version	7. *X* Version
Refresh time	By default, the refresh operation is performed every second.		If **index.refresh_interval** is not explicitly specified and indexes do not receive the **search** request for long time (the duration is specified by **index.search.idle.after** and the value is 30 seconds by default), Elasticsearch does not periodically refresh until when a new search request is received. In this case, search requests are not returned until the next refresh is complete. Therefore, the first search request takes a long time.
Parent fuse	The parent fuse is triggered when the sum of memory statistics in multiple child fuses exceeds the threshold. The default threshold is 70%.		The parent fuse is triggered when the heap memory usage exceeds the threshold. The default threshold is 95%.
Field Data fuse threshold	The default value of **indices.breaker.fielddata.limit** is 60%.		The default value of **indices.breaker.fielddata.limit** is 40%.
The _all field	Supported	Discarded	Deleted

Version Feature	**5.** ***X*** **Version**	**6.** ***X*** **Version**	**7.** ***X*** **Version**
hits.total returned by the search API	**hits.total** returned by the search API is a number, indicating the number of hits. `{ "took": 0, "timed_out": false, "_shards": { "total": 5, "successful": 5, "failed": 0 }, "hits": { "total": 4, "max_score": 1, } }`		**hits.total** is not a number. `{ "took" : 76, "timed_out" : false, "_shards" : { "total" : 1, "successful" : 1, "skipped" : 0, "failed" : 0 }, "hits" : { "total" : { "value" : 4, "relation" : "eq" }, "max_score" : 1.0 } }` In the preceding information: **value** indicates the number of matched records. **relation** indicates whether the number of hit records in the **value** parameter is accurate. **eq** indicates an accurate value. **gte** indicates that the number of hit records is greater than or equal to the **value** parameter.
_cache/clear API	The **POST** and **GET** methods are supported.		Only the **POST** method is supported.

8 Security

8.1 Shared Responsibilities

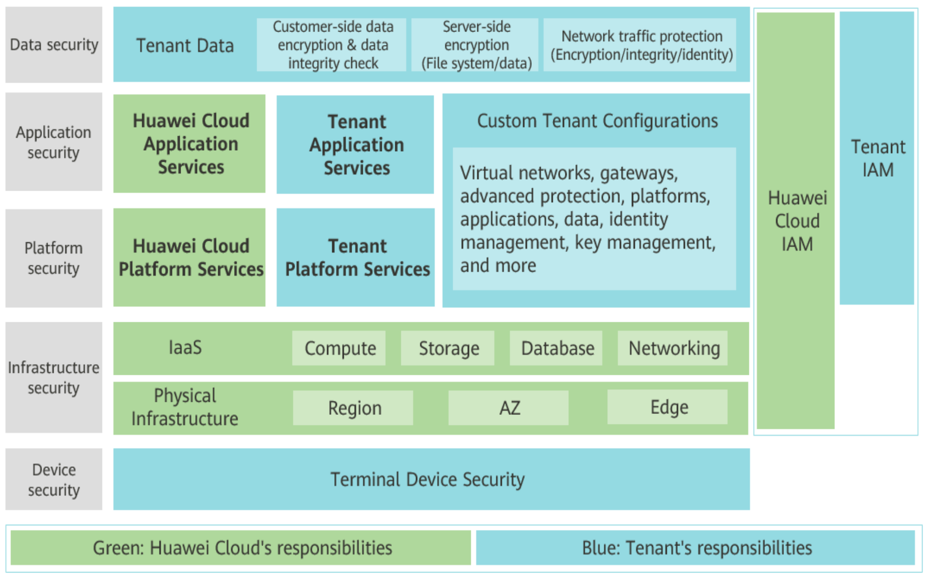
Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To cope with emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system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

Figure 8-1 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 8-1 Huawei Cloud shared security responsibility model


Data security
Tenant Data
Customer-side data encryption & data integrity check
Server-side encryption (File system/data)
Network traffic protection (Encryption/integrity/identity)
Application security
Huawei Cloud Application Services
Tenant Application Services
Custom Tenant Configurations
Virtual networks, gateways, advanced protection, platforms, applications, data, identity management, key management, and more
Platform security
Huawei Cloud Platform Services
Tenant Platform Services
Infrastructure security
IaaS
Compute
Storage
Database
Networking
Physical Infrastructure
Region
AZ
Edge
Device security
Terminal Device Security
Huawei Cloud IAM
Tenant IAM
Green: Huawei Cloud's responsibilities
Blue: Tenant's responsibilities

8.2 Authentication and Access Control

CSS uses Identity and Access Management (IAM) and cluster security mode to perform authentication and access control for service resources and security clusters, respectively. The two modules are independent of each other.

IAM is used to control resource operation permissions on the CSS management plane. If you need to assign different permissions to employees in your organization to access your CSS resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your CSS resources. For details about CSS permission management, see **Permissions Management**.

If the security mode is enabled for a cluster, identity authentication is required when users access the cluster. You can also authorize other users to access Kibana of the security cluster. For details, see **Creating a User and Granting Permissions by Using Kibana**. CSS supports identity authentication and access control only for clusters in security mode.

8.3 Data Protection Technologies

CSS uses network isolation, in addition to various host and data security measures.

- Network isolation

  The entire network is divided into two planes: service plane and management plane. The two planes are deployed and isolated physically to ensure the security of the service and management networks.

  – Service plane: This is the network plane of the cluster. It provides service channels for users and delivers data definitions, indexing, and search capabilities.

  – Management plane: This is the management console, where you manage CSS.

- Host security

  CSS provides the following security measures:

  - The VPC security group ensures the security of the hosts in a VPC.
  - Network access control lists (ACLs) allow you to control what data can enter or exit your network.
  - The internal security infrastructure (including the network firewall, intrusion detection system, and protection system) monitors all network traffic that enters or exits the VPC through an IPsec VPN.

- Data security

  Multiple replicas, cross-AZ deployment of clusters, and third-party (OBS) backup of index data ensure the security of user data.

8.4 Audit and Logs

Auditing

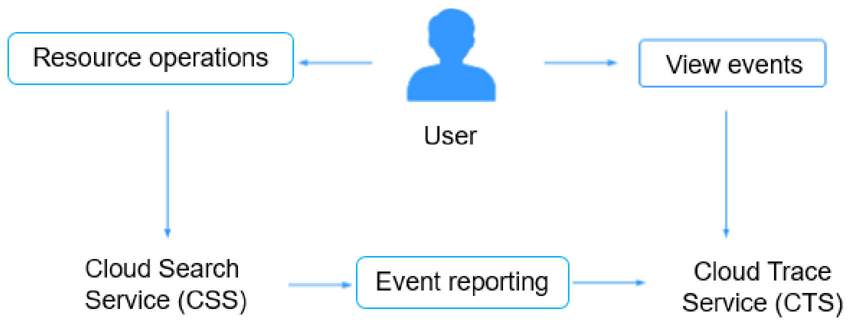
Cloud Trace Service (CTS) records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, track resource changes, audit compliance, and locate faults.

After you enable CTS and create a tracker, CTS starts to record CSS operations for audit.

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

For details about CSS operations that can be recorded by CTS, see **Key Operations Recorded by CTS**.

Figure 8-2 CTS



After CTS is enabled, CTS starts recording CSS operations. The CTS management console stores the operation records generated in the last seven days. For details about how to view operation records for the last seven days on the CTS console, see **Querying Real-Time Traces**.

Logging

CSS allows you to back up and query logs for locating faults. You can back up cluster logs in OBS buckets and download required log files from OBS for fault analysis and locating. For details about CSS log management, see **Managing Logs**.

8.5 Security Risk Monitoring

Cloud Eye is a monitoring platform for Huawei Cloud resources. It provides capabilities such as real-time monitoring, alarm reporting, resource grouping, and website monitoring. Cloud Eye can monitor metrics of CSS clusters and nodes and visualize the monitoring information in reports.

For details about CSS metrics that can be monitored by Cloud Eye, see **Supported Metrics**. Cloud Eye allows you to configure threshold-crossing alarms for specified monitoring metrics.

9 Permissions Management

If you need to assign different permissions to employees in your organization to access your CSS resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control.

If the current account has met your requirements, you do not need to create an independent IAM user for permission management. Then you can skip this section. This will not affect other functions of CSS.

With IAM, you can use your account to create IAM users for your employees and assign permissions to the users to control their access to your resources. IAM is free of charge. You pay only for the resources you purchase. For more information about IAM, see **IAM Service Overview**.

Permissions Management

New IAM users do not have any permissions assigned by default. You need to first add them to one or more groups and attach policies or roles to these groups. The users then inherit permissions from the groups and can perform specified operations on cloud services based on the permissions they have been assigned.

CSS is a project-level service deployed in specific physical regions. Therefore, CSS permissions are assigned to projects in specific regions and only take effect in these regions. If you want the permissions to take effect in all regions, you need to assign the permissions to projects in each region. When accessing CSS, the users need to switch to a region where they have been authorized to use cloud services.

You can use roles and policies to grant users permissions.

- 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 grant permissions, you need to also assign dependency roles. Roles are not ideal for fine-grained authorization and secure access control.
- Policies: A type of fine-grained authorization mechanism that defines the permissions for performing operations on specific cloud resources under certain conditions. This mechanism allows for more flexible authorization. Policies allow you to meet requirements for more secure access control. For example, CSS administrators can only grant CSS users the permissions needed

for managing a particular type of CSS resources. Most policies define permissions based on APIs. For the API actions supported by CSS, see **Permissions Policies and Supported Actions**.

Table 9-1 lists all the system-defined roles and policies supported by CSS.

- **CSS Administrator** depends on the roles of other services to execute its permissions. Therefore, if you assign the **Elasticsearch Administrator** role to a user, assign its dependency roles at the same time.
- **CSS FullAccess** and **CSS ReadOnlyAccess** can be used to control the resources that users can access. For example, if you want your software developers to use CSS resources but not delete them or perform any high-risk operations, you can create IAM users for these software developers and assign them only the permissions required for using CSS resources.

Table 9-1 CSS system permissions

Role/Policy Name	Type	Description	Dependency
CSS Administrator	System-defined role	Full permissions for CSS. This role depends on the **Tenant Guest**, **Server Administrator**, and **IAM ReadOnlyAccess** roles in the same project.	The **VPCEndpoint Administrator** system role is required for accessing a cluster through a VPC endpoint. The **CES Administrator** system role is required for using the Cloud Eye monitoring service. Some operations depend on the following permissions: ● View the agency list: iam:agencies:listAgencies iam:permissions:listRolesForAgency iam:permissions:listRolesForAgencyOnProject ● Automatically create an agency: iam:agencies:listAgencies iam:agencies:createAgency iam:permissions:grantRoleToAgency ● Display enterprise projects and predefined tags on the console: eps:enterpriseProjects:list tms:predefineTags:list ● Use the snapshot, word dictionary, and log management functions: obs:bucket:Get* obs:bucket:List* obs:object:List* obs:object:Get*
CSS FullAccess	System-defined policy	Full CSS permissions granted through policies. Users with these permissions can perform all operations on CSS. Some functions depend on corresponding permissions. To use certain functions, you need to enable the dependent permissions in the same project.	

Role/Policy Name	Type	Description	Dependency
			obs:bucket:HeadBucket obs:object:PutObject obs:object:DeleteObject ● bss:order:update bss:order:pay
CSS ReadOnlyAccess	System-defined policy	Read-only permissions for CSS. Users with these permissions can only view CSS data. Some functions depend on corresponding permissions. To use certain functions, you need to enable the dependent permissions in global services.	Some operations depend on the following permissions: ● View the agency list: iam:agencies:listAgencies iam:permissions:listRolesForAgency iam:permissions:listRolesForAgencyOnProject ● Display enterprise projects and predefined tags on the console: eps:enterpriseProjects:list tms:predefineTags:list ● Use the snapshot, word dictionary, and log management functions: obs:bucket:Get* obs:bucket:List* obs:object:List* obs:object:Get* obs:bucket:HeadBucket

Table 9-2 lists the common operations supported by each system permission of CSS. Please choose proper system permissions according to this table.

Table 9-2 Common operations supported by each system-defined policy

Operation	CSS FullAccess	CSS ReadOnlyAccess	CSS Administrator	Remarks
Creating a cluster	√	x	√	-
Querying the cluster list	√	√	√	-
Querying the cluster details	√	√	√	-
Deleting a cluster	√	x	√	-
Restarting a cluster	√	x	√	-
Expanding cluster capacity	√	x	√	-
Adding instances and expanding instance storage capacity	√	x	√	-
Querying tags of a specified cluster	√	√	√	-
Querying all tags	√	√	√	-
Loading a custom word dictionary	√	x	√	Depends on OBS and IAM permissions
Querying the status of a custom word dictionary	√	√	√	-

Operation	CSS FullAccess	CSS ReadOnlyAccess	CSS Administrator	Remarks
Deleting a custom word dictionary	√	x	√	-
Automatically setting basic configurations of a cluster snapshot	√	x	√	Depends on OBS and IAM permissions
Modifying basic configurations of a cluster snapshot	√	x	√	Depends on OBS and IAM permissions
Setting the automatic snapshot creation policy	√	x	√	-
Querying the automatic snapshot creation policy	√	√	√	-
Manually creating a snapshot	√	x	√	-
Querying the snapshot list	√	√	√	-
Restoring a snapshot	√	x	√	-
Deleting a snapshot	√	x	√	-

Operation	CSS FullAccess	CSS ReadOnlyAccess	CSS Administrator	Remarks
Disabling the snapshot function	√	x	√	-
Modifying specifications	√	x	√	-
Scaling in clusters	√	x	√	-

Helpful Links

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

10 Specifications

When creating a CSS cluster, you can select specifications as required. For details about the specifications and application scenarios, see **Table 10-1**.

Table 10-1 Node specifications

CPU Architecture	Specification Type	CPU/Memory Ratio	Application Scenario
x86	Compute-intensive	1:2	This specification provides strong CPUs and is suitable for search scenarios that require high computing performance and low latency, such as e-commerce and app search. It is recommended when ultra-high I/O disks are used. This specification is expensive and has higher reliability than that of the NVMe local disk cluster.
	Disk-intensive	1:8	This specification uses local SAS pass-through disks that have large space. It is applicable to scenarios where a large amount of data is stored, such as searching for logs and public opinions. Generally, it is recommended for cold nodes.
	General computing	1:4	This is the default specification, which is widely used in diverse scenarios and can meet your common requirements.
	Memory-optimized	1:8	This specification provides extra large memory. It is recommended for tasks that require large memory but not particularly fast responses, such as multi-aggregation (in the **filedata** heap), sorting, and column storage format **DocValue** (out-of-heap memory).

11 Constraints

This topic describes limits on the node quantity and resource quotas of a CSS cluster. For details about the limits and limitations of different features provided by CSS, see relevant topics in the CSS User Guide.

Maximum and Minimum Numbers of Nodes in a Cluster

The following tables provide the maximum and minimum numbers of nodes each CSS cluster can have.

Table 11-1 Maximum and minimum numbers of nodes in an Elasticsearch or OpenSearch cluster

Number of Nodes	Limit
Maximum number of nodes in a cluster	Default: **32**. Maximum: 200. To change the default value, contact technical support.
Minimum number of nodes in a cluster	1

Table 11-2 Maximum and minimum numbers of nodes in a Logstash cluster

Number of Nodes	Limit
Maximum number of nodes in a cluster	100
Minimum number of nodes in a cluster	1

Quotas

CSS uses the following resource quotas:

- Number of instances

- CPUs
- Memory capacity in GB
- Number of disks
- Disk size (GB)

For details about how to check and modify quotas, see **Quotas**.

12 Performance Metrics

This section describes the result of testing performance of CSS clusters in version 7.6.2 by using Rally 1.0.0 provided by Elasticsearch.

In this test, the official geonames is used. The size is 3.2 GB and there are 11,396,505 documents. The index uses six shards (five shards by default). For details about performance metrics, see the official document **https://esrally.readthedocs.io/en/stable/summary_report.html#summary-report**.

The following table lists the performance test result of a cluster with three nodes of the **ess.spec-4u16g** specification.

Metric	Task	Value	Unit
Cumulative indexing time of primary shards	-	11.95073333	min
Min cumulative indexing time across primary shards	-	0	min
Median cumulative indexing time across primary shards	-	2.339941667	min
Max cumulative indexing time across primary shards	-	2.470116667	min
Cumulative indexing throttle time of primary shards	-	0	min
Min cumulative indexing throttle time across primary shards	-	0	min

Metric	Task	Value	Unit
Median cumulative indexing throttle time across primary shards	-	0	min
Max cumulative indexing throttle time across primary shards	-	0	min
Cumulative merge time of primary shards	-	4.21495	min
Cumulative merge count of primary shards	-	65	-
Min cumulative merge time across primary shards	-	0	min
Median cumulative merge time across primary shards	-	0.813216667	min
Max cumulative merge time across primary shards	-	0.974483333	min
Cumulative merge throttle time of primary shards	-	0.83345	min
Min cumulative merge throttle time across primary shards	-	0	min
Median cumulative merge throttle time across primary shards	-	0.157775	min
Max cumulative merge throttle time across primary shards	-	0.24605	min
Cumulative refresh time of primary shards	-	2.164983333	min

Metric	Task	Value	Unit
Cumulative refresh count of primary shards	-	291	-
Min cumulative refresh time across primary shards	-	0	min
Median cumulative refresh time across primary shards	-	0.425391667	min
Max cumulative refresh time across primary shards	-	0.450516667	min
Cumulative flush time of primary shards	-	0.1559	min
Cumulative flush count of primary shards	-	11	-
Min cumulative flush time across primary shards	-	0	min
Median cumulative flush time across primary shards	-	0.0248	min
Max cumulative flush time across primary shards	-	0.043433333	min
Total Young Gen GC	-	6.421	s
Total Old Gen GC	-	0	s
Store size	-	3.124213032	GB
Translog size	-	2.790678718	GB
Heap used for segments	-	15.03110981	MB
Heap used for doc values	-	0.043689728	MB
Heap used for terms	-	13.85075188	MB
Heap used for norms	-	0.077697754	MB

Metric	Task	Value	Unit
Heap used for points	-	0.266856194	MB
Heap used for stored fields	-	0.792114258	MB
Segment count	-	99	-
Min Throughput	index-append	92446.94	docs/s
Median Throughput	index-append	92935.55	docs/s
Max Throughput	index-append	93217.68	docs/s
50th percentile latency	index-append	176.7329985	ms
90th percentile latency	index-append	285.5450693	ms
100th percentile latency	index-append	333.228537	ms
50th percentile service time	index-append	176.7329985	ms
90th percentile service time	index-append	285.5450693	ms
100th percentile service time	index-append	333.228537	ms
error rate	index-append	0	%
Min Throughput	index-stats	90.04	ops/s
Median Throughput	index-stats	90.06	ops/s
Max Throughput	index-stats	90.11	ops/s
50th percentile latency	index-stats	3.6713165	ms
90th percentile latency	index-stats	3.919960223	ms
99th percentile latency	index-stats	4.500246093	ms
99.9th percentile latency	index-stats	20.14171663	ms
100th percentile latency	index-stats	21.36778278	ms
50th percentile service time	index-stats	3.604376499	ms

Metric	Task	Value	Unit
90th percentile service time	index-stats	3.8517339	ms
99th percentile service time	index-stats	4.36148177	ms
99.9th percentile service time	index-stats	20.0748024	ms
100th percentile service time	index-stats	21.300971	ms
error rate	index-stats	0	%
Min Throughput	node-stats	90.05	ops/s
Median Throughput	node-stats	90.09	ops/s
Max Throughput	node-stats	90.32	ops/s
50th percentile latency	node-stats	4.056046	ms
90th percentile latency	node-stats	4.256959922	ms
99th percentile latency	node-stats	7.993649534	ms
99.9th percentile latency	node-stats	15.0162469	ms
100th percentile latency	node-stats	18.79192022	ms
50th percentile service time	node-stats	3.989104	ms
90th percentile service time	node-stats	4.1902188	ms
99th percentile service time	node-stats	7.39785926	ms
99.9th percentile service time	node-stats	14.95028028	ms
100th percentile service time	node-stats	15.226284	ms
error rate	node-stats	0	%
Min Throughput	default	50.03	ops/s
Median Throughput	default	50.04	ops/s
Max Throughput	default	50.09	ops/s

Metric	Task	Value	Unit
50th percentile latency	default	2.890284501	ms
90th percentile latency	default	3.054330301	ms
99th percentile latency	default	3.41013575	ms
99.9th percentile latency	default	4.536945459	ms
100th percentile latency	default	5.063877001	ms
50th percentile service time	default	2.82345	ms
90th percentile service time	default	2.987489999	ms
99th percentile service time	default	3.34539951	ms
99.9th percentile service time	default	4.466092296	ms
100th percentile service time	default	4.996857	ms
error rate	default	0	%
Min Throughput	term	150.06	ops/s
Median Throughput	term	150.09	ops/s
Max Throughput	term	150.14	ops/s
50th percentile latency	term	2.822069666	ms
90th percentile latency	term	2.927460233	ms
99th percentile latency	term	3.585279107	ms
99.9th percentile latency	term	9.586351776	ms
100th percentile latency	term	13.36534567	ms
50th percentile service time	term	2.755832	ms

Metric	Task	Value	Unit
90th percentile service time	term	2.8613018	ms
99th percentile service time	term	3.4037467	ms
99.9th percentile service time	term	4.571924473	ms
100th percentile service time	term	13.301659	ms
error rate	term	0	%
Min Throughput	phrase	149.99	ops/s
Median Throughput	phrase	150.07	ops/s
Max Throughput	phrase	150.13	ops/s
50th percentile latency	phrase	3.207932333	ms
90th percentile latency	phrase	3.514073	ms
99th percentile latency	phrase	26.65015757	ms
99.9th percentile latency	phrase	38.92041855	ms
100th percentile latency	phrase	40.044182	ms
50th percentile service time	phrase	3.1409695	ms
90th percentile service time	phrase	3.3666699	ms
99th percentile service time	phrase	9.39342965	ms
99.9th percentile service time	phrase	18.80974216	ms
100th percentile service time	phrase	21.417291	ms
error rate	phrase	0	%
Min Throughput	country_agg_uncached	4.01	ops/s
Median Throughput	country_agg_uncached	4.01	ops/s

Metric	Task	Value	Unit
Max Throughput	country_agg_uncached	4.01	ops/s
50th percentile latency	country_agg_uncached	153.726532	ms
90th percentile latency	country_agg_uncached	156.0977097	ms
99th percentile latency	country_agg_uncached	167.696362	ms
100th percentile latency	country_agg_uncached	198.43754	ms
50th percentile service time	country_agg_uncached	153.606521	ms
90th percentile service time	country_agg_uncached	155.9869715	ms
99th percentile service time	country_agg_uncached	167.5793267	ms
100th percentile service time	country_agg_uncached	198.325432	ms
error rate	country_agg_uncached	0	%
Min Throughput	country_agg_cached	100.04	ops/s
Median Throughput	country_agg_cached	100.05	ops/s
Max Throughput	country_agg_cached	100.07	ops/s
50th percentile latency	country_agg_cached	2.7020445	ms
90th percentile latency	country_agg_cached	2.783604899	ms
99th percentile latency	country_agg_cached	3.03382523	ms
99.9th percentile latency	country_agg_cached	3.635769276	ms
100th percentile latency	country_agg_cached	4.106574	ms
50th percentile service time	country_agg_cached	2.6356045	ms

Metric	Task	Value	Unit
90th percentile service time	country_agg_cached	2.717349899	ms
99th percentile service time	country_agg_cached	2.93948264	ms
99.9th percentile service time	country_agg_cached	3.567144201	ms
100th percentile service time	country_agg_cached	4.039871999	ms
error rate	country_agg_cached	0	%
Min Throughput	scroll	20.04	pages/s
Median Throughput	scroll	20.05	pages/s
Max Throughput	scroll	20.07	pages/s
50th percentile latency	scroll	421.9468245	ms
90th percentile latency	scroll	433.3017323	ms
99th percentile latency	scroll	450.0724775	ms
100th percentile latency	scroll	505.502723	ms
50th percentile service time	scroll	421.0948965	ms
90th percentile service time	scroll	432.4389587	ms
99th percentile service time	scroll	449.2045264	ms
100th percentile service time	scroll	504.653479	ms
error rate	scroll	0	%
Min Throughput	expression	2	ops/s
Median Throughput	expression	2	ops/s
Max Throughput	expression	2	ops/s
50th percentile latency	expression	270.920167	ms

Metric	Task	Value	Unit
90th percentile latency	expression	277.4334041	ms
99th percentile latency	expression	286.5631326	ms
100th percentile latency	expression	293.09254	ms
50th percentile service time	expression	270.662187	ms
90th percentile service time	expression	277.1779957	ms
99th percentile service time	expression	286.3073191	ms
100th percentile service time	expression	292.826178	ms
error rate	expression	0	%
Min Throughput	painless_static	1.5	ops/s
Median Throughput	painless_static	1.5	ops/s
Max Throughput	painless_static	1.5	ops/s
50th percentile latency	painless_static	360.9218617	ms
90th percentile latency	painless_static	368.2584616	ms
99th percentile latency	painless_static	382.3877013	ms
100th percentile latency	painless_static	425.989704	ms
50th percentile service time	painless_static	360.5910995	ms
90th percentile service time	painless_static	367.9205895	ms
99th percentile service time	painless_static	382.0613883	ms
100th percentile service time	painless_static	425.659728	ms
error rate	painless_static	0	%
Min Throughput	painless_dynamic	1.5	ops/s

Metric	Task	Value	Unit
Median Throughput	painless_dynamic	1.5	ops/s
Max Throughput	painless_dynamic	1.5	ops/s
50th percentile latency	painless_dynamic	354.4270103	ms
90th percentile latency	painless_dynamic	362.9108269	ms
99th percentile latency	painless_dynamic	409.7732626	ms
100th percentile latency	painless_dynamic	410.1049017	ms
50th percentile service time	painless_dynamic	354.0901565	ms
90th percentile service time	painless_dynamic	362.5730453	ms
99th percentile service time	painless_dynamic	409.4442952	ms
100th percentile service time	painless_dynamic	409.777646	ms
error rate	painless_dynamic	0	%
Min Throughput	decay_geo_gauss_function_score	1	ops/s
Median Throughput	decay_geo_gauss_function_score	1	ops/s
Max Throughput	decay_geo_gauss_function_score	1	ops/s
50th percentile latency	decay_geo_gauss_function_score	354.387216	ms
90th percentile latency	decay_geo_gauss_function_score	358.9124798	ms
99th percentile latency	decay_geo_gauss_function_score	363.9485787	ms
100th percentile latency	decay_geo_gauss_function_score	371.780245	ms
50th percentile service time	decay_geo_gauss_function_score	353.7158425	ms

Metric	Task	Value	Unit
90th percentile service time	decay_geo_gauss_function_score	358.2845019	ms
99th percentile service time	decay_geo_gauss_function_score	363.275623	ms
100th percentile service time	decay_geo_gauss_function_score	371.114045	ms
error rate	decay_geo_gauss_function_score	0	%
Min Throughput	decay_geo_gauss_script_score	1	ops/s
Median Throughput	decay_geo_gauss_script_score	1	ops/s
Max Throughput	decay_geo_gauss_script_score	1	ops/s
50th percentile latency	decay_geo_gauss_script_score	379.4620745	ms
90th percentile latency	decay_geo_gauss_script_score	383.2876548	ms
99th percentile latency	decay_geo_gauss_script_score	389.7544834	ms
100th percentile latency	decay_geo_gauss_script_score	395.75293	ms
50th percentile service time	decay_geo_gauss_script_score	378.8137045	ms
90th percentile service time	decay_geo_gauss_script_score	382.6389076	ms
99th percentile service time	decay_geo_gauss_script_score	389.1097136	ms
100th percentile service time	decay_geo_gauss_script_score	395.100654	ms
error rate	decay_geo_gauss_script_score	0	%
Min Throughput	field_value_function_score	1.5	ops/s
Median Throughput	field_value_function_score	1.5	ops/s
Max Throughput	field_value_function_score	1.51	ops/s

Metric	Task	Value	Unit
50th percentile latency	field_value_function_score	142.4418055	ms
90th percentile latency	field_value_function_score	146.0292471	ms
99th percentile latency	field_value_function_score	149.4448299	ms
100th percentile latency	field_value_function_score	154.4188467	ms
50th percentile service time	field_value_function_score	141.8792295	ms
90th percentile service time	field_value_function_score	145.4722711	ms
99th percentile service time	field_value_function_score	148.8731825	ms
100th percentile service time	field_value_function_score	153.87006	ms
error rate	field_value_function_score	0	%
Min Throughput	field_value_script_score	1.5	ops/s
Median Throughput	field_value_script_score	1.5	ops/s
Max Throughput	field_value_script_score	1.51	ops/s
50th percentile latency	field_value_script_score	200.310233	ms
90th percentile latency	field_value_script_score	206.2690364	ms
99th percentile latency	field_value_script_score	216.7453505	ms
100th percentile latency	field_value_script_score	252.6694313	ms
50th percentile service time	field_value_script_score	199.886616	ms
90th percentile service time	field_value_script_score	205.7897592	ms
99th percentile service time	field_value_script_score	216.2602712	ms

Metric	Task	Value	Unit
100th percentile service time	field_value_script_score	252.180659	ms
error rate	field_value_script_score	0	%
Min Throughput	random_function_score	1.5	ops/s
Median Throughput	random_function_score	1.5	ops/s
Max Throughput	random_function_score	1.5	ops/s
50th percentile latency	random_function_score	242.6018717	ms
90th percentile latency	random_function_score	251.1366288	ms
99th percentile latency	random_function_score	290.9842466	ms
100th percentile latency	random_function_score	307.5584597	ms
50th percentile service time	random_function_score	242.149128	ms
90th percentile service time	random_function_score	250.6830153	ms
99th percentile service time	random_function_score	290.5378949	ms
100th percentile service time	random_function_score	307.111375	ms
error rate	random_function_score	0	%
Min Throughput	random_script_score	1.5	ops/s
Median Throughput	random_script_score	1.5	ops/s
Max Throughput	random_script_score	1.5	ops/s
50th percentile latency	random_script_score	258.3288777	ms
90th percentile latency	random_script_score	262.5996219	ms

Metric	Task	Value	Unit
99th percentile latency	random_script_score	276.7350459	ms
100th percentile latency	random_script_score	278.8234443	ms
50th percentile service time	random_script_score	257.8902625	ms
90th percentile service time	random_script_score	262.1680452	ms
99th percentile service time	random_script_score	276.3056912	ms
100th percentile service time	random_script_score	278.384714	ms
error rate	random_script_score	0	%
Min Throughput	large_terms	1.5	ops/s
Median Throughput	large_terms	1.5	ops/s
Max Throughput	large_terms	1.5	ops/s
50th percentile latency	large_terms	429.023917	ms
90th percentile latency	large_terms	438.5573247	ms
99th percentile latency	large_terms	468.2661402	ms
100th percentile latency	large_terms	494.4412297	ms
50th percentile service time	large_terms	428.772941	ms
90th percentile service time	large_terms	438.29435	ms
99th percentile service time	large_terms	468.0068679	ms
100th percentile service time	large_terms	494.168992	ms
error rate	large_terms	0	%
Min Throughput	large_filtered_terms	1.5	ops/s

Metric	Task	Value	Unit
Median Throughput	large_filtered_terms	1.5	ops/s
Max Throughput	large_filtered_terms	1.5	ops/s
50th percentile latency	large_filtered_terms	433.0397738	ms
90th percentile latency	large_filtered_terms	443.241508	ms
99th percentile latency	large_filtered_terms	460.8045067	ms
100th percentile latency	large_filtered_terms	486.396965	ms
50th percentile service time	large_filtered_terms	432.7802525	ms
90th percentile service time	large_filtered_terms	442.9739873	ms
99th percentile service time	large_filtered_terms	460.7444745	ms
100th percentile service time	large_filtered_terms	486.145846	ms
error rate	large_filtered_terms	0	%
Min Throughput	large_prohibited_terms	1.5	ops/s
Median Throughput	large_prohibited_terms	1.5	ops/s
Max Throughput	large_prohibited_terms	1.5	ops/s
50th percentile latency	large_prohibited_terms	430.1467708	ms
90th percentile latency	large_prohibited_terms	436.8730103	ms
99th percentile latency	large_prohibited_terms	484.5697929	ms
100th percentile latency	large_prohibited_terms	492.75088	ms
50th percentile service time	large_prohibited_terms	429.8833325	ms

Metric	Task	Value	Unit
90th percentile service time	large_prohibited_terms	436.6196592	ms
99th percentile service time	large_prohibited_terms	484.3087876	ms
100th percentile service time	large_prohibited_terms	492.492977	ms
error rate	large_prohibited_terms	0	%
Min Throughput	desc_sort_population	1.5	ops/s
Median Throughput	desc_sort_population	1.51	ops/s
Max Throughput	desc_sort_population	1.51	ops/s
50th percentile latency	desc_sort_population	45.9402765	ms
90th percentile latency	desc_sort_population	49.01190953	ms
99th percentile latency	desc_sort_population	58.5120831	ms
100th percentile latency	desc_sort_population	60.027354	ms
50th percentile service time	desc_sort_population	45.2962825	ms
90th percentile service time	desc_sort_population	48.3757462	ms
99th percentile service time	desc_sort_population	57.86711494	ms
100th percentile service time	desc_sort_population	59.377354	ms
error rate	desc_sort_population	0	%
Min Throughput	asc_sort_population	1.5	ops/s
Median Throughput	asc_sort_population	1.51	ops/s
Max Throughput	asc_sort_population	1.51	ops/s

Metric	Task	Value	Unit
50th percentile latency	asc_sort_population	46.02105783	ms
90th percentile latency	asc_sort_population	48.79212977	ms
99th percentile latency	asc_sort_population	55.94577758	ms
100th percentile latency	asc_sort_population	72.898199	ms
50th percentile service time	asc_sort_population	45.37886	ms
90th percentile service time	asc_sort_population	48.1426418	ms
99th percentile service time	asc_sort_population	55.30153109	ms
100th percentile service time	asc_sort_population	72.260339	ms
error rate	asc_sort_population	0	%
Min Throughput	desc_sort_geonameid	1.5	ops/s
Median Throughput	desc_sort_geonameid	1.51	ops/s
Max Throughput	desc_sort_geonameid	1.51	ops/s
50th percentile latency	desc_sort_geonameid	52.22274167	ms
90th percentile latency	desc_sort_geonameid	69.4325779	ms
99th percentile latency	desc_sort_geonameid	79.57920996	ms
100th percentile latency	desc_sort_geonameid	80.11872267	ms
50th percentile service time	desc_sort_geonameid	51.6055115	ms
90th percentile service time	desc_sort_geonameid	68.801679	ms
99th percentile service time	desc_sort_geonameid	79.41158055	ms

Metric	Task	Value	Unit
100th percentile service time	desc_sort_geonameid	79.465491	ms
error rate	desc_sort_geonameid	0	%
Min Throughput	asc_sort_geonameid	1.5	ops/s
Median Throughput	asc_sort_geonameid	1.51	ops/s
Max Throughput	asc_sort_geonameid	1.51	ops/s
50th percentile latency	asc_sort_geonameid	51.35154333	ms
90th percentile latency	asc_sort_geonameid	52.2966503	ms
99th percentile latency	asc_sort_geonameid	55.33079961	ms
100th percentile latency	asc_sort_geonameid	55.520544	ms
50th percentile service time	asc_sort_geonameid	50.7138335	ms
90th percentile service time	asc_sort_geonameid	51.6588923	ms
99th percentile service time	asc_sort_geonameid	54.68967127	ms
100th percentile service time	asc_sort_geonameid	54.874135	ms
error rate	asc_sort_geonameid	0	%

13 Related Services

Figure 13-1 shows the relationships between CSS and other services.

Figure 13-1 Relationships between CSS and other services

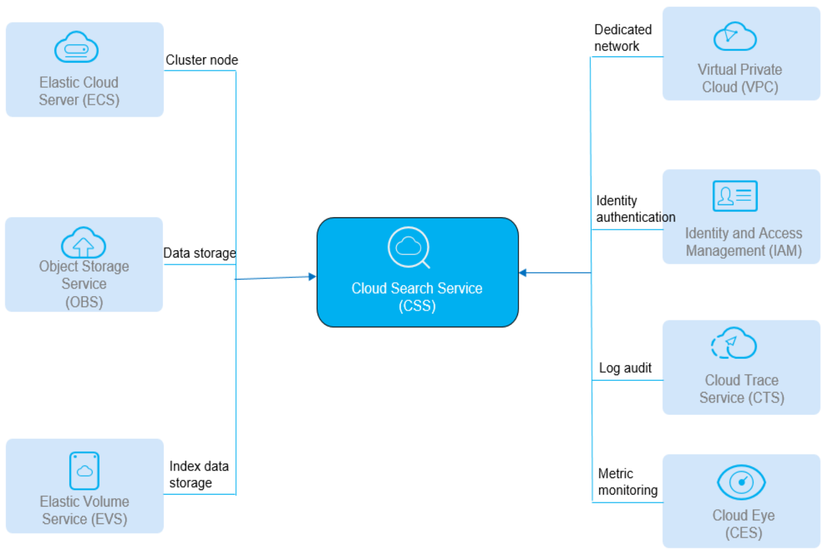

Elastic Cloud Server (ECS)
Cluster node
Object Storage Service (OBS)
Data storage
Elastic Volume Service (EVS)
Index data storage
Cloud Search Service (CSS)
Dedicated network
Virtual Private Cloud (VPC)
Identity authentication
Identity and Access Management (IAM)
Log audit
Cloud Trace Service (CTS)
Metric monitoring
Cloud Eye (CES)

Table 13-1 Relationships between CSS and other services

Service	Description
Virtual Private Cloud (VPC)	CSS clusters are created in the subnets of a VPC. VPCs provide a secure, isolated, and logical network environment for your clusters. For details, see **Creating a VPC User Guide**.
Elastic Cloud Server (ECS)	In a CSS cluster, each node represents an ECS. When you create a cluster, ECSs are automatically created.
Elastic Volume Service (EVS)	CSS uses EVS to store index data. When you create a cluster, EVSs are automatically created for cluster data storage.

Service	Description
Object Storage Service (OBS)	Snapshots of CSS clusters are stored in OBS buckets. For details, see **Object Storage Service User Guide**.
Identity and Access Management (IAM)	IAM authenticates access to CSS. For details, see **Identity and Access Management User Guide**.
Cloud Eye	CSS uses Cloud Eye to monitor cluster metrics in real time. The supported CSS metrics include the disk usage and cluster health status. You can learn about the disk usage of the cluster based on the disk usage metric. You can learn about the health status of a cluster based on the cluster health status metric. For details, see **Cloud Eye User Guide**.
Cloud Trace Service (CTS)	With CTS, you can record operations associated with CSS for query, audit, and backtracking operations. For details, see **Cloud Trace Service Guide**.

14 Basic Concepts

Cluster

CSS provides functions on a per cluster basis. A cluster represents an independent search service that consists of multiple nodes.

Index

An index stores Elasticsearch data. It is a logical space in which one or more shards are grouped.

Shard

An index can potentially store a large amount of data that can exceed the hardware limits of a single node. To solve this problem, Elasticsearch provides the ability to subdivide your index into multiple pieces called shards. When you create an index, you can simply define the number of shards that you want. Each shard is in itself a fully-functional and independent "index" that can be hosted on any node in the cluster.

You need to specify the number of shards before creating an index and cannot change the number after the index is successfully created.

Replica

A replica is a copy of the actual storage index in a shard. It can be understood as a backup of the shard. Replicas help prevent single point of failures (SPOFs). You can increase or decrease the number of replicas based on your service requirements.

Document

An entity for Elasticsearch storage. Equivalent to the row in the RDB, the document is the basic unit that can be indexed.

Document Type

Similar to a table in the RDB, type is used to distinguish between different data.

In versions earlier than Elasticsearch 7.*x*, each index can contain multiple document types. Elasticsearch defines a type for each document.

Elasticsearch 7.*x* and later versions only support documents of the .doc type.

Mapping

A mapping is used to restrict the type of a field and can be automatically created based on data. It is similar to the schema in the database.

Field

The field is the minimum unit of a document. It is similar to the column in the database.