云搜索服务

用户操作指南

天翼云科技有限公司

目录

产品概述

产品定义

产品简介

云搜索服务是一款全托管的在线分布式搜索服务，为结构化/非结构化数据提供低成本、高性能及可靠性的检索、分析服务能力。基于云上产品可以实现快速地对大量文本数据进行全文搜索，向量检索等高级搜索功能，从而快速搭建商品检索、企业文档检索、APP搜索等各种检索分析服务。

产品功能

兼容开源引擎

提供 Elasticsearch 和 OpenSearch 引擎，100%兼容开源和生态。提供自研增强内核，支持全文检索、向量检索、混合检索等多元搜索方式，具备分析、聚合、高亮显示等能力，实时可靠。

便捷操作

通过云搜索服务控制台，可以一键订购实例，开箱即用，主要操作一键可达，实例便捷操作。

灵活迁移

支持实例之间的数据迁移，可以从自建集群、低版本集群轻松迁移上云。

访问方式

公有云提供了 Web 化的服务管理平台，即天翼云官网管理控制台。用户可以注册天翼云账号登录使用，目前云搜索服务提供主子账号可访问管理控制台共享主账号订购的实例管理权限，子账号暂不具备订购权限。

产品优势

天翼云云搜索服务具备以下特点和显著优势：

高效易用

升级搜索组件内核，吞吐性能提升 30%，可将数据经由可视化平台进行分析展示。

无忧运维

全托管服务，开箱即用，主要操作一键可达，专业团队贴身看护。

高安全性

云搜索服务主要从以下几个方面保障数据和业务运行安全:
网络隔离:

- 整个服务部署于用户公有云上专享的虚拟私有云中，提供安全隔离的网络环境，保证用户大数据集群的业务、管理的安全性。通过结合虚拟私有云的子网划分、路由控制、安全组等功能，可以为用户提供高安全、高可靠的网络隔离环境。
- 管理平面和用户服务隔离部署，控制台用于管理云搜索服务。

主机安全

通过 VPC 及安全组专有网络来确保主机的安全。针对云主机，天翼云也提供如下安全措施：

- 操作系统内核安全加固
- 更新操作系统最新补丁
- 操作系统权限控制
- 操作系统端口管理
- 操作系统协议与端口防攻击
- 云主机监控
- 防 DDoS 攻击
- 定期备份数据
- 增加登录密码强度

数据安全

- 数据安全在云搜索服务中，通过多副本机制保证用户的数据安全。

- 支持用户认证与细粒度级别鉴权。
- 云搜索数据存储在天翼云云硬盘产品中，云硬盘采用三副本冗余机制，定期备份，保证上层服务数据的高容灾性。

内核增强

天翼云在开源 Elasticsearch 和 OpenSearch 的基础上做了大量集群能力增强。例如：

天翼云云搜索在开源 Elasticsearch 基础上实现了对向量检索功能的支持；

针对中文分词场景进行优化，提升准确性，在多项中文分词数据集上的平均 F1-score 可以达到 0.87；

在默认的 LZ4 和 BEST_COMPRESSION 压缩算法之外，额外实现了对 ZSTD 压缩算法的支持，在性能几乎没有损失的情况下，压缩率提升了 10%以上；

自研实现了流量控制插件，可以实时监控集群的 Search 请求情况，将集群的请求流量控制在一个合理的范围内；

天翼云自研搜索引擎还实现了对跨集群复制、简繁体转换、异步搜索、SQL 功能、细粒度权限管控等多种搜索高阶功能的支持。

具体各个内核的增强介绍和使用示例可参考增强特性章节。

产品组件

Elasticsearch

Elasticsearch 是一个分布式全文检索引擎，能够自动管理索引和查询在集群中的分布方式，实现极其流畅的操作。无论是结构化、非结构化的文本、数字数据还是地理空间数据，Elasticsearch 都能支持快速搜索的方式高效的存储和构建索引。

- 高拓展、高实时的搜索与数据分析引擎。
- 生态完整，可叠合 Logstash、Kibana 作为集成解决方案收集存储分析呈现数据。
- 采用 RestfulAPI 标准，可通过 http 接口使用 JSON 格式进行操作数据。

Kibana

Kibana 是一个开源的可用于集群开发、运维；数据检索与分析及数据可视化平台，与 Elasticsearch 搜索引擎一起使用。通过 Kibana 可以搜索、查看存放在

Elasticsearch 索引中的数据，也可以实现以图表、地图等方式展示数据。

云搜索服务的 Elasticsearch 集群默认提供 Kibana，无需安装部署，绑定弹性 IP 后即可一键访问 Kibana。云搜索服务兼容了开源 Kibana 可视化展现和 Elasticsearch 统计分析能力。

- 支持包括甘特图在内的多种数据呈现方式
- 支持多种数据统计方式
- 支持时间、标签等各种维度分类
- 支持可视化索引管理功能
- 支持 SQL 化查询数据方式
- 支持多用户角色的权限管理

OpenSearch

OpenSearch 是一个基于 Elasticsearch 开源版研发的开源的搜索和分析引擎，OpenSearch 继承了 Elasticsearch 的许多特性，并且提供了一些额外的功能和改进，以支持更广泛的社区和用例。OpenSearch 相比 Elasticsearch，在性能和功能上都有一定的提升。同时天翼云在此基础上升级了内核能力，对搜索引擎在分词、流量管控、存算分离等方面予以增强。

OpenSearch Dashboards

OpenSearch Dashboards 是从 Kibana 7.10.2 分叉而来的开源分支，现在作为 Apache 2.0 许可的独立开源项目运行。它提供了直观的可用于创建、共享、交互式的开发、运维数据分析平台。与 Opensearch 搭配使用，并提供角色基础访问控制能力。以下简称 Dashboards。

- 支持多用户角色的权限管理
- 支持包括甘特图在内的多种数据呈现方式
- 支持多种数据统计方式
- 支持高阶的索引管理能力
- 支持通知告警功能

- 支持基于 SQL 的查询能力
- 支持 CSV、PDF、Excel 等文件的报告生成功能

应用场景

日志检索分析

云搜索服务可以采集分析各类以文件形式存储的日志，包括不限于服务器、容器日志等。结合消息队列 Kafka、Logstash 对日志数据进行采集清洗，进入 Elasticsearch 或 OpenSearch 中存储检索分析，并通过 Kibana 或 Dashborads 进行可视化呈现。

- 高稳定：通过读写限流、查询熔断、集群监控等方式保障。
- 性能优化：通过支持压缩算法、内核增强节省存储空间，增强读写速率、降低响应延迟。
- 易用性：支持多种统计分析方法、划分维度，丰富的可视化查询，可对接 BI 实现拖拽式报表。

站内搜索

面对海量数据，提供分布式的信息检索工具，可根据网站内容进行关键字检索，也可搭建商品检索或推荐系统。

- 实时检索：支持模糊搜索，数据来源多样且持续写入，站内资料或商品信息更新数秒内即可被检索。
- 分类统计：检索同时可以将符合条件的结果进行排序或分类统计。
- 多场景适配：可用于网站、移动端应用搜索、企业内部应用等多种需要快速检索的场景。

数据分析

提供开箱即用的可视化、高聚合分析能力，便捷高效的对海量数据实时分析。更好地适配实时多维分析在线查询服务场景，满足高实时性，高查询 QPS，低查询延迟的数据分析要求。

- 快速响应：支持海量的、数据源多样、字段不固定的数据高并发处理，毫秒级响应。
- 复杂查询：支持大批量模糊关键字搜索及一些聚合的复杂查询。

- 向量检索：基于向量特征相似度匹配，支持图文搜索、地理位置搜索。

术语解释

实例

云搜索服务是以实例为单位进行组织，一个实例代表一个独立运行的搜索服务，与集群维度对应，由多个相同规格的云主机节点构成。

索引（Index）

用于存储 Elasticsearch 或 OpenSearch 的数据，是一个或多个分片分组在一起的逻辑空间，类似于传统数据库，存储具有相同结构的文档。

文档（Document）

Elasticsearch 或 OpenSearch 存储的实体，是可以被索引的基本单位，相当于关系型数据库中的行，代表一个具体的实体记录。

分片（Shard）

分片是索引的逻辑分区，用于水平分割数据，提高存储和查询的可扩展性。当您创建一个索引时，您可以根据实际情况指定分片的数量。每个分片托管在实例中的任意一个节点中，且每个分片本身是一个独立的、全功能的“索引”。

分片的数量只能在创建索引前指定，且在索引创建成功后无法修改。

副本（Replica）

副本是实际存储索引分片的一个副本。可以理解为备分片。副本的存在可以预防单节点故障。使用过程中，您可以根据业务情况增加或减少副本数量。

映射（Mapping）

用来定义字段的类型，可以根据数据自动创建。

字段（Field）

组成文档的最小单位，代表特定属性或数据项，每个字段都有一个数据类型和相关设置。

与天翼云其他服务之间关系

相关服务	交互功能
虚拟私有云(CT-VPC ，Virtual Private Cloud)	云搜索服务在购买前需要提前开通虚拟私有云，实例将建立在同资源池下指定的子网内，VPC 之间网络隔离，可为用户提供安全的网络环境。
弹性云主机（CT-ECS，Elastic Cloud Server）	云搜索服务的每个实例节点即为一台弹性云主机，创建实例时会根据购买需求自动下单对应数量，主要提供计算分析服务。
云硬盘（CT-EVS，Elastic Volume Service）	云搜索服务使用云硬盘存储用户的索引数据，创建实例时会根据购买需求将云硬盘自动挂载在对应节点。
弹性 IP（EIP，Elastic IP）	云搜索服务如需要开放公网访问，需要购买弹性 IP 或 IPv6 带宽并绑定到实例对应组件上。
对象存储服务（CT-ZOS，Zettabyte Object Storage）	云搜索服务的实例快照及所需要安装的自定义插件均需要开通对象存储服务，进行存储和读取。
统一身份认证服务（Identity and Access Management，简称 IAM）	云搜索服务使用天翼云官网统一身份认证服务进行身份鉴权。

约束与限制

实例与节点

- 公测期仅允许一个账号开通一个在用实例。
- 每个实例最小节点数量为 3，最大支持 100 节点，公测期间请以订购页面约束为准。
- 网络访问
- 实例创建完成后，不支持切换 VPC 网络，请在开通前提前规划业务部署。
- 为保证数据安全，Kibana、Dashboards 需设置密码和访问安全策略。
- 实例使用

- 公测期间暂不支持对实例的配置进行修改，如您需要修改，请提报工单至工程师协助修改。

性能说明

Elasticsearch 性能数据

通过 Elasticsearch 官方提供的 benchmark 工具 rally2.2.0，测试的集群规格为一个 4u16g，存储使用通用型 SSD，单节点存储容量 500GB 节点数为 3 的集群。版本为 Elasticsearch 7.10.2。

测试结果如下：

Metric	Task	Value	Unit
Cumulative indexing time of primary shards	-	9.6826	min
Min cumulative indexing time across primary shards	-	5e-05	min
Median cumulative indexing time across primary shards	-	1.59631	min
Max cumulative indexing time across primary shards	-	1.6559	min
Cumulative indexing throttle time of primary shards	-	0	min
Min cumulative indexing throttle time across primary shards	-	0	min
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	-	0.790833	min

Cumulative merge count of primary shards	-	13	-
Min cumulative merge time across primary shards	-	0	min
Median cumulative merge time across primary shards	-	0.117217	min
Max cumulative merge time across primary shards	-	0.170483	min
Cumulative merge throttle time of primary shards	-	0.15735	min
Min cumulative merge throttle time across primary shards	-	0	min
Median cumulative merge throttle time across primary shards	-	0.0214667	min
Max cumulative merge throttle time across primary shards	-	0.0472833	min
Cumulative refresh time of primary shards	-	0.714117	min
Cumulative refresh count of primary shards	-	118	-
Min cumulative refresh time across primary shards	-	0.000166667	min
Median cumulative refresh time across primary shards	-	0.09015	min
Max cumulative refresh time across primary shards	-	0.156217	min
Cumulative flush time of primary shards	-	0.573367	min
Cumulative flush count of primary shards	-	8	-
Min cumulative flush time across primary shards	-	0.000183	min

		333	
Median cumulative flush time across primary shards	-	0.05215	min
Max cumulative flush time across primary shards	-	0.1521	min
Total Young Gen GC	-	1.4	s
Total Old Gen GC	-	0	s
Store size	-	3.24094	GB
Translog size	-	5.6345e-07	GB
Heap used for segments	-	1.29252	MB
Heap used for doc values	-	0.0896034	MB
Heap used for terms	-	0.980042	MB
Heap used for norms	-	0.132935	MB
Heap used for points	-	0	MB
Heap used for stored fields	-	0.0899429	MB
Segment count	-	181	-
Min Throughput	index-append	154995	docs/s
Median Throughput	index-	164787	docs/

	append		s
Max Throughput	index-append	182229	docs/s
50th percentile latency	index-append	186.381	ms
90th percentile latency	index-append	246.258	ms
100th percentile latency	index-append	7814.53	ms
50th percentile service time	index-append	186.381	ms
90th percentile service time	index-append	246.258	ms
100th percentile service time	index-append	7814.53	ms
error rate	index-append	0	%

OpenSearch 性能数据

通过 OpenSearch 官方提供的 benchmark 工具 opensearch-benchmark1.6.0，测试的集群规格为一个 4u16g，存储使用通用型 SSD，单节点存储容量 500GB 节点数为 3 的集群。版本为 OpenSearch 2.9.0。

测试结果如下：

Metric	Task	Value	Unit
Cumulative indexing time of primary shards	-	8.95952	min
Min cumulative indexing time across primary shards	-	0	min
Median cumulative indexing time across primary shards	-	1.42328	min
Max cumulative indexing time across primary shards	-	1.55992	min
Cumulative indexing throttle time of primary shards	-	0	min
Min cumulative indexing throttle time across primary shards	-	0	min
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	-	0.142	min
Cumulative merge count of primary shards	-	9	-
Min cumulative merge time across primary shards	-	0	min
Median cumulative merge time across primary shards	-	0.0203	min
Max cumulative merge time across primary shards	-	0.0313	min
Cumulative merge throttle time of primary shards	-	0	min
Min cumulative merge throttle time across primary shards	-	0	min

Median cumulative merge throttle time across primary shards	-	0	min
Max cumulative merge throttle time across primary shards	-	0	min
Cumulative refresh time of primary shards	-	1.24425	min
Cumulative refresh count of primary shards	-	153	-
Min cumulative refresh time across primary shards	-	0	min
Median cumulative refresh time across primary shards	-	0.1552	min
Max cumulative refresh time across primary shards	-	0.263117	min
Cumulative flush time of primary shards	-	1.04765	min
Cumulative flush count of primary shards	-	9	-
Min cumulative flush time across primary shards	-	0	min
Median cumulative flush time across primary shards	-	0.0887167	min
Max cumulative flush time across primary shards	-	0.258517	min
Total Young Gen GC	-	1.445	s
Total Old Gen GC	-	0	s
Store size	-	2.83533	GB
Translog size	-	5.64465e-05	GB

Heap used for segments	-	0	MB
Heap used for doc values	-	0	MB
Heap used for terms	-	0	MB
Heap used for norms	-	0	MB
Heap used for points	-	0	MB
Heap used for stored fields	-	0	MB
Segment count	-	151	-
Min Throughput	index-append	173799	docs/s
Median Throughput	index-append	189543	docs/s
Max Throughput	index-append	202549	docs/s
50th percentile latency	index-append	160.137	ms
90th percentile latency	index-append	210.292	ms
100th percentile latency	index-append	12273.8	ms
50th percentile service time	index-append	160.137	ms

90th percentile service time	index-append	210.292	ms
100th percentile service time	index-append	12273.8	ms
error rate	index-append	0	%

性能受硬件影响比较大，以上数据仅代表当次测试结果，产品实际性能以实测为准。如果您对产品性能有较高要求，建议选择性能较优的存储资源。

计费说明

资源节点

目前一类资源节点已在**华东 1** 资源池开通，更多资源池规划部署中。因用户权限不同，实际可选资源池请以控制台实际可见区域为准。

您可以在华东 1 资源池中选择不同的可用区去使用：可用区 1、可用区 2、可用区 3。

实例规格及规划建议

天翼云云搜索服务，支持根据业务需求，灵活选择合适的实例配置。我们根据天翼云搜索团队丰富的实际业务经验，在此提供一些搜索引擎常见使用场景下，配置选择的建议。您可以根据业务的读写请求、数据存算和搜索与分析等需求进行参考，当然，也需要您根据业务的实际使用情况逐步去探索。

实例版本：

我们同时提供 Elasticsearch 和 OpenSearch 两种选择。

天翼云基于 Elasticsearch7.10.2，默认搭配同版本的 Kibana 使用，并在开源版本做了大量的能力增强，包括压缩算法、中文分词、SQL 兼容、异步搜索、向量检索、跨实例复制、索引管理、拼音分词、简繁体转换、HDFS 存储等，并进行了安全漏洞修复、BUG 修复、性能优化等。

天翼云 OpenSearch 基于 OpenSearch2.9.0 版本打造，默认搭配同版本 OpenSearch

Dashboards 使用。在开源版本的基础上也做了大量的能力增强和优化，包括中文分词优化、流量控制、监控告警、对象存储适配、拼音分词、简繁体转换等。

计算节点选型：

目前支持的节点规格如下，不同资源池支持在售机型不同，云搜索服务会根据产品规划需要适时调整在售机型，请以购买页可见机型为准。

机型类型	机型名称	CPU	Mem	适合场景
通用型	esearch-4c16g	4	16	适合平衡型场景，适用于大多数通用搜索和分析任务。 当实例需要处理中等规模的数据集，并且查询和索引操作相对平衡时，这种比例可以提供足够的处理能力和内存资源。
	esearch-8c32g	8	32	
计算加强型	esearch-8c16g	8	16	适合 CPU 密集型操作，如需要进行大量数据聚合或实时分析的场景。 当查询操作非常复杂，需要进行大量的数据计算和聚合时，较高的 CPU 资源可以提高处理速度。
	esearch-16c32g	16	32	
内存优化型	esearch-4c32g	4	32	适合内存密集型操作，如大量数据的索引和搜索。 当数据集较大，需要频繁进行全文搜索或复杂查询时，较高的内存可以提高缓存效率，减少磁盘 I/O 操作。

产品价格

当前产品处于公测期，云搜索服务本身不收取资源及软件服务费用。

但因部分功能使用到天翼云官网其他商用产品（如对象存储、弹性 IP、IPv6 带宽等），需要您自行开通并按开通规格付费使用。

计费模式

计费项

云搜索服务的计费项由节点规格费用、节点存储费用组成。Kibana/Dashboards 节点为默认开通，该节点独立收费。公测期间该部分免费。

如果需要开放节点公网访问，或使用快照、插件等能力，您另需要根据实际购买情况进行付费。

计费方式

当前云搜索服务一类节点仅支持包周期的计费方式。

包年/包月：根据版本/资源组的购买时长，一次性支付费用。最短时长为 1 个月，实际可订购时长以页面显示为准。

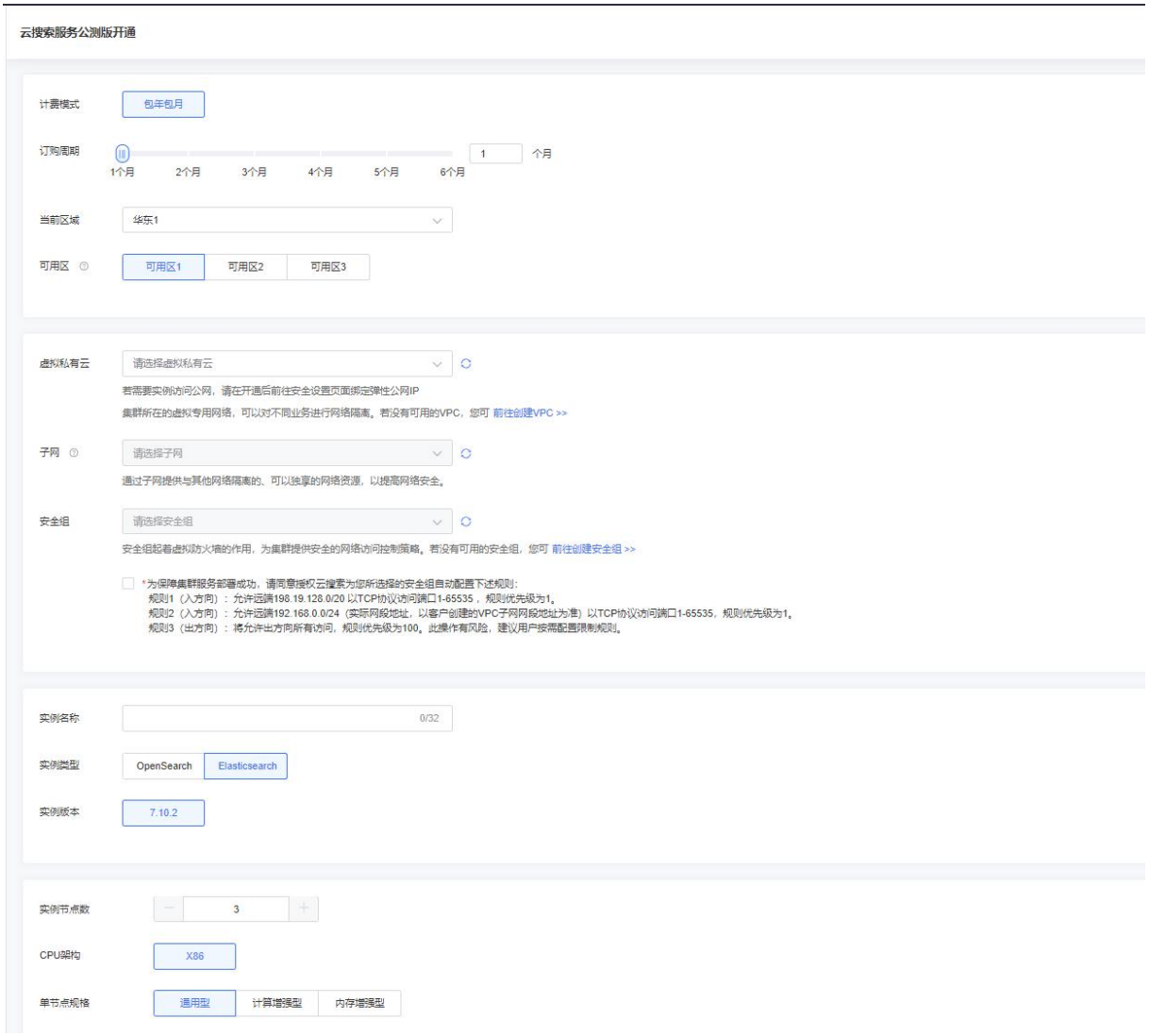
购买

1. 注册天翼云官网账号，登录后进入官网首页。

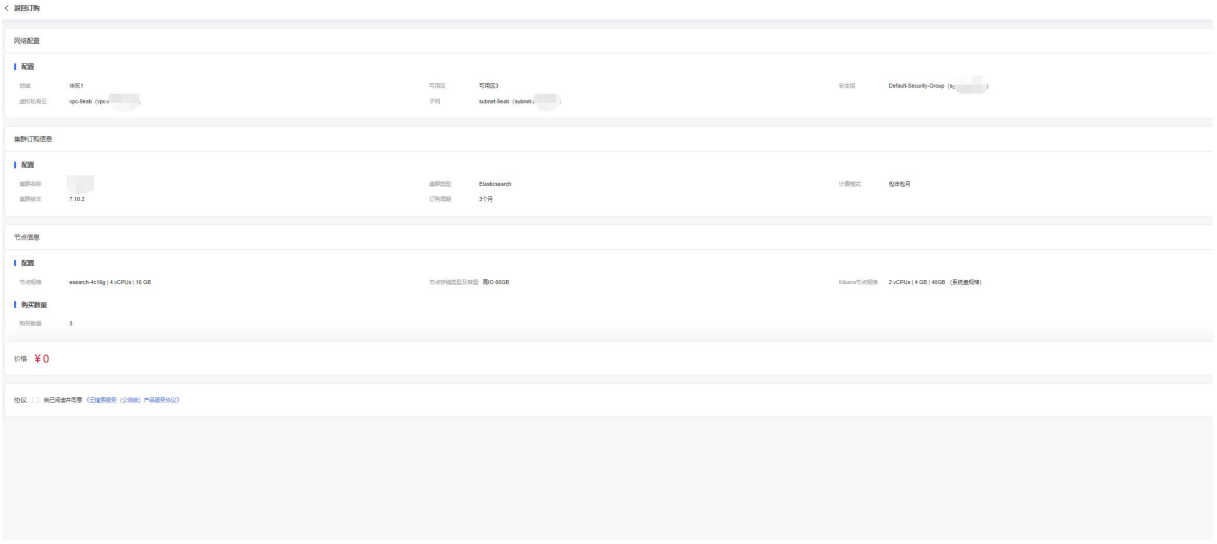
2. 在官网首页，单击左上角“产品”，【大数据>云搜索服务】。

3. 在【云搜索服务】产品页，单击【立即开通】。

4. 在云搜索服务实例创建页面选择计费模式、订购周期、当前区域（即资源池）为资源节点内支持的一类节点、可用区、实例类型、实例版本、实例名称、节点规格等基础配置信息后，按页面提示并根据需要进行配置，然后点击下一步。



5. 按页面提示，勾选相关协议，公测期间支付 0 元，即可完成订购，等待资源开通完成，对应实例处于“运行中”状态，即为成功。



续订

您可以从控制台列表页或费用中心进行续费。

1. 访问控制台

选择对应的云搜索服务实例，点击“更多”>“续订”，在跳转的费用中心页面完成续费操作。

2. 直接在天翼云官网“我的”>“费用中心”>“订单管理”>“续订管理”中选择订单进行续订。

退订

如您不再需要某一实例，可以退订实例释放资源。

约束限制

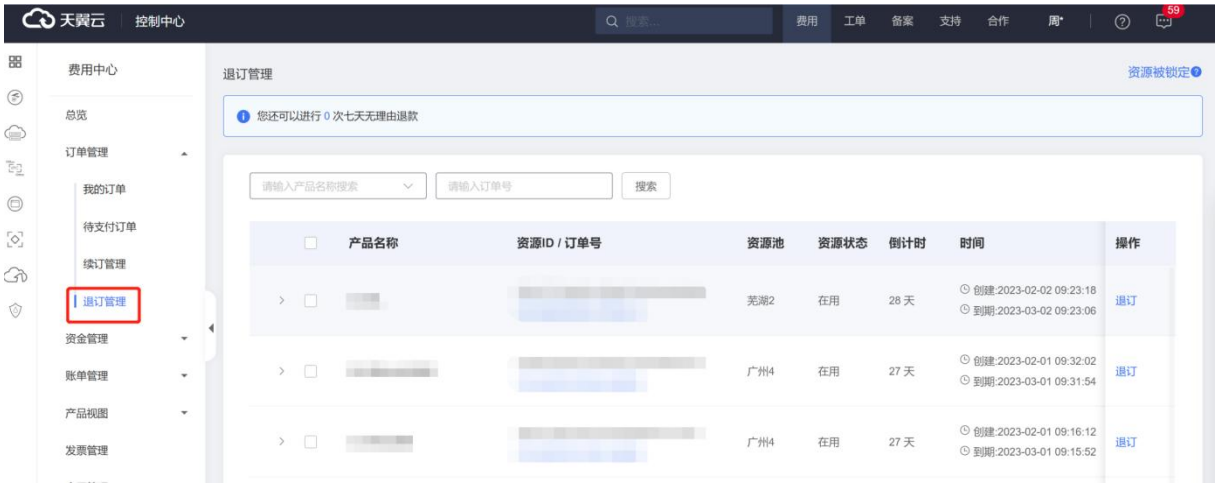
1、退订实例时，云搜索服务会释放掉全部包括云主机、云硬盘的所有通过云搜索服务购买的资源，您自行购买、设置的如 VPC、弹性 IP、对象存储等资源不会退订，您需要自行前往相关产品控制台退订。

2、退订操作，资源会立即释放，数据会清理，请提前备份数据。

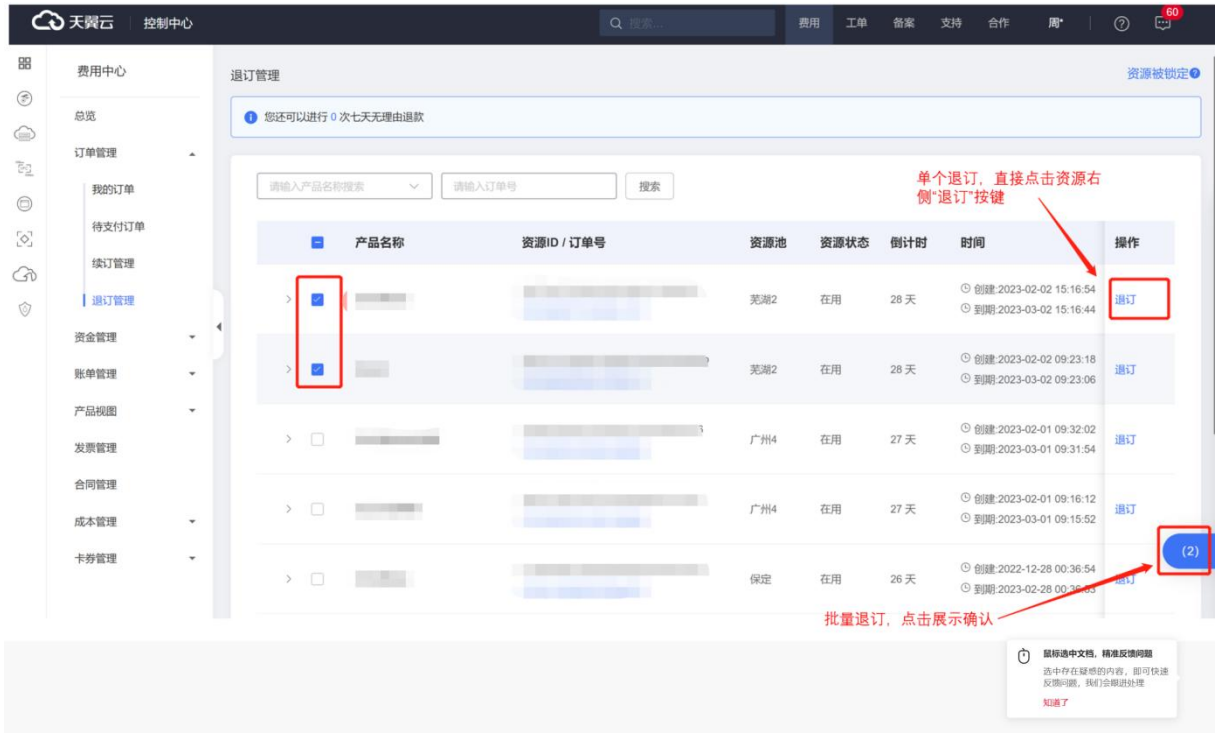
3、公测期间暂不涉及退费。

方法一、通过订单管理退订

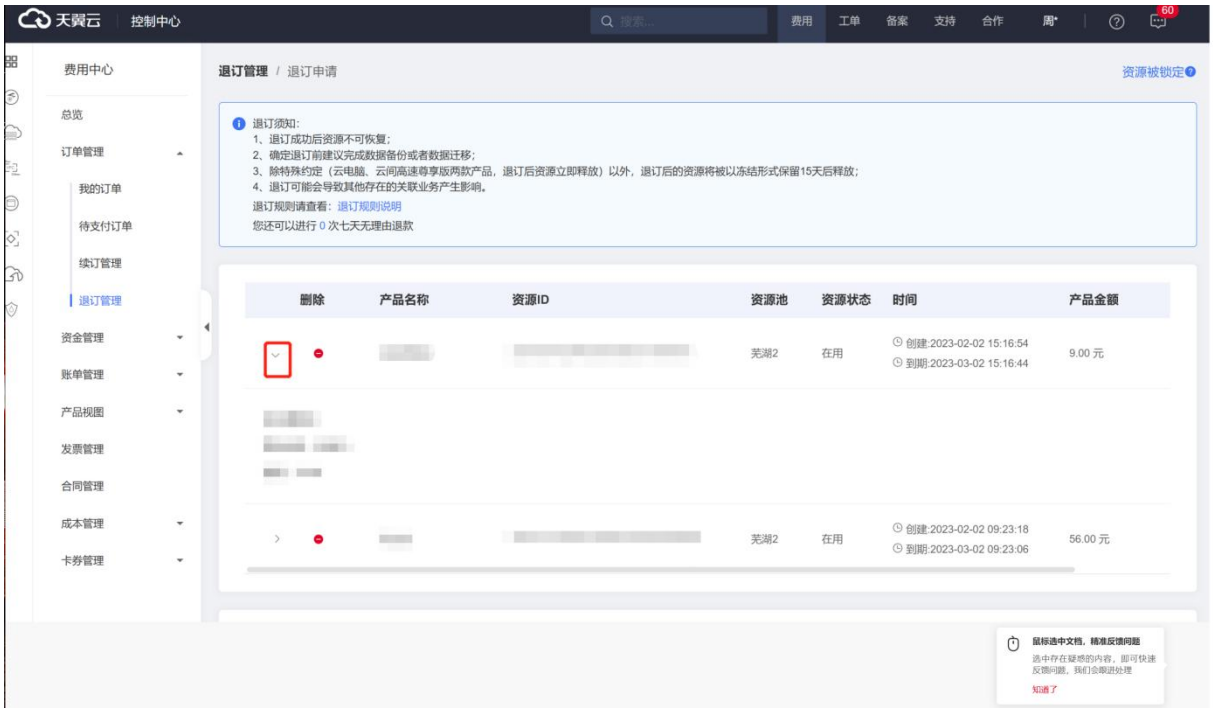
1. 登录天翼云官网，进入“费用中心”>“订单管理”>“退订管理”页面。



2. 选择要退订的资源，点击退订，天翼云目前支持单个退订和批量退订。



3. 退订前仔细阅读退订须知，然后确认退订资源信息。



4. 信息确认无误后勾选协议，点击退订并再次确认，确认后即刻完成退订。

方法二：从云搜索服务控制台退订

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要退订的实例。
2. 在操作列单击“更多”>“退订”。

3. 在跳转的页面进行退订操作，与订单管理方式退订操作一致。

快速入门

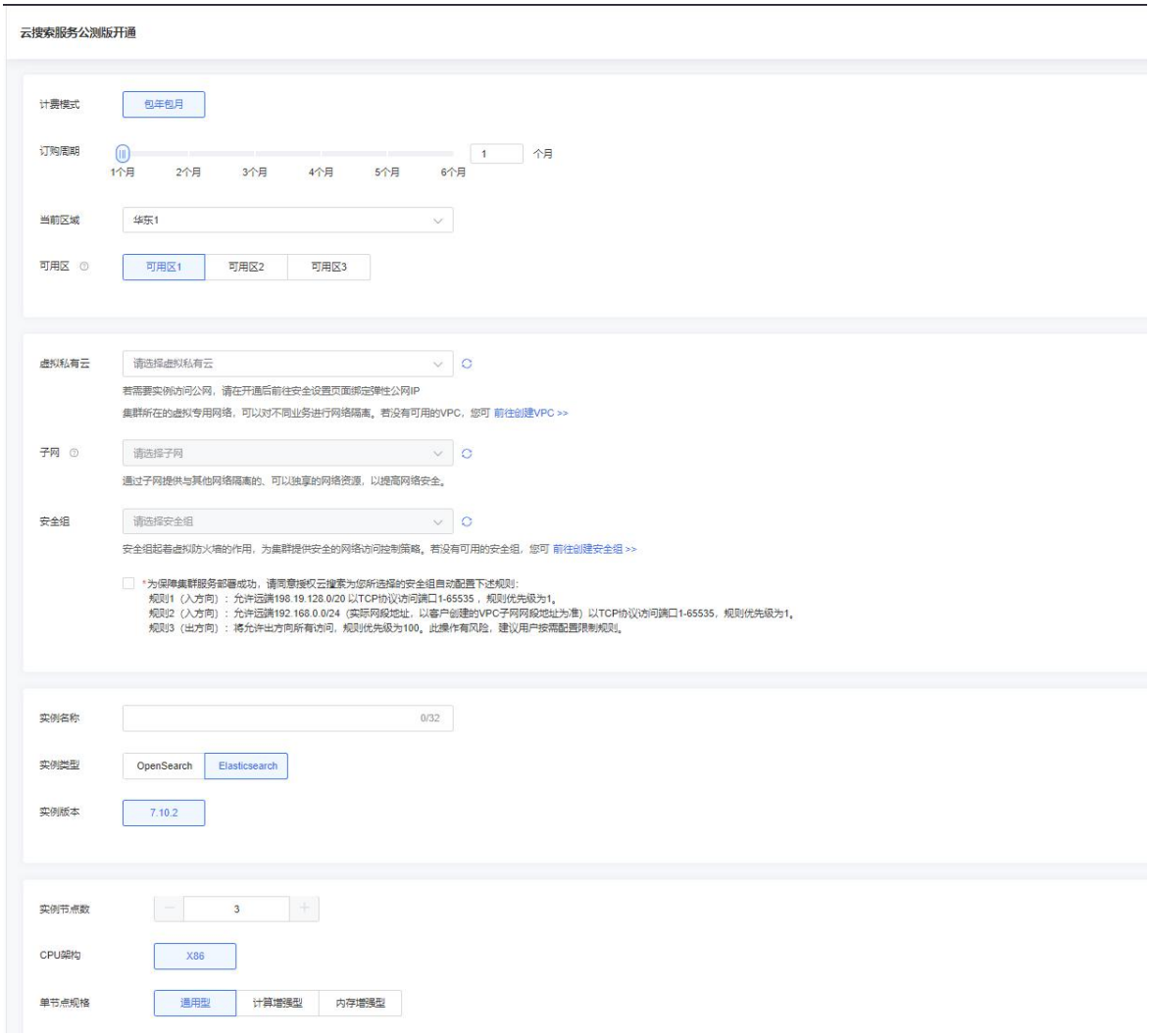
使用 Elasticsearch 搜索数据

这里，我们以一个示范的例子，向您简单介绍天翼云 Elasticsearch 实例可以提供的检索和分析服务，包括创建索引、数据导入、数据搜索、筛选排序等。

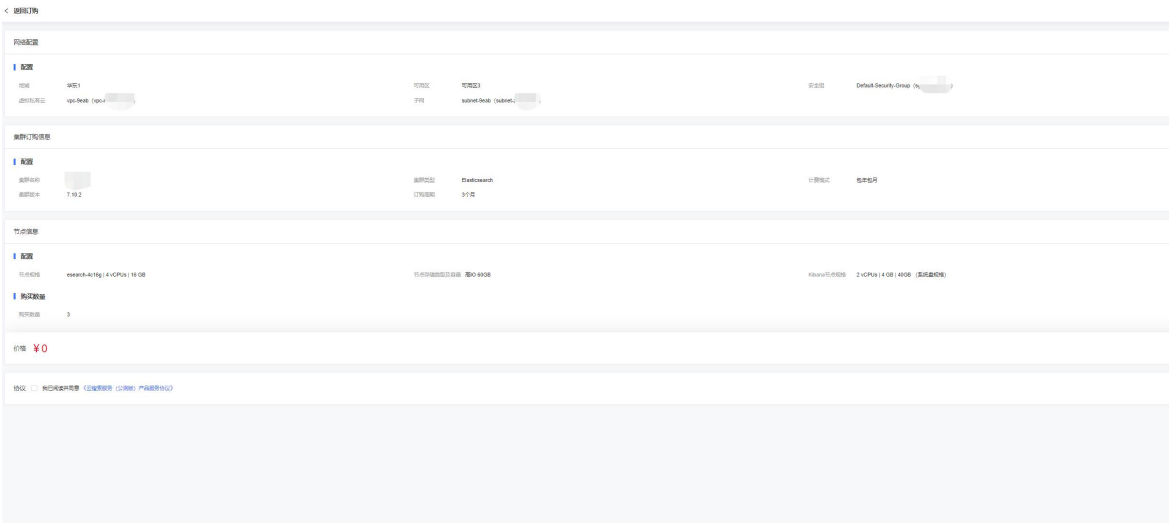
1.创建实例

a. 在【云搜索服务】产品页，单击【立即开通】；

b. 在云搜索服务实例创建页面选择实例类型为 Elasticsearch、订购周期、当前区域（即资源池）、可用区、填写实例名称、节点规格等基础配置信息后，按页面提示并根据需要进行配置，然后点击下一步。



1.3 按页面提示，勾选相关协议，公测期间支付 0 元，即可完成订购，等待资源开通完成，对应实例处于“运行中”状态，即为成功。



2. 导入数据

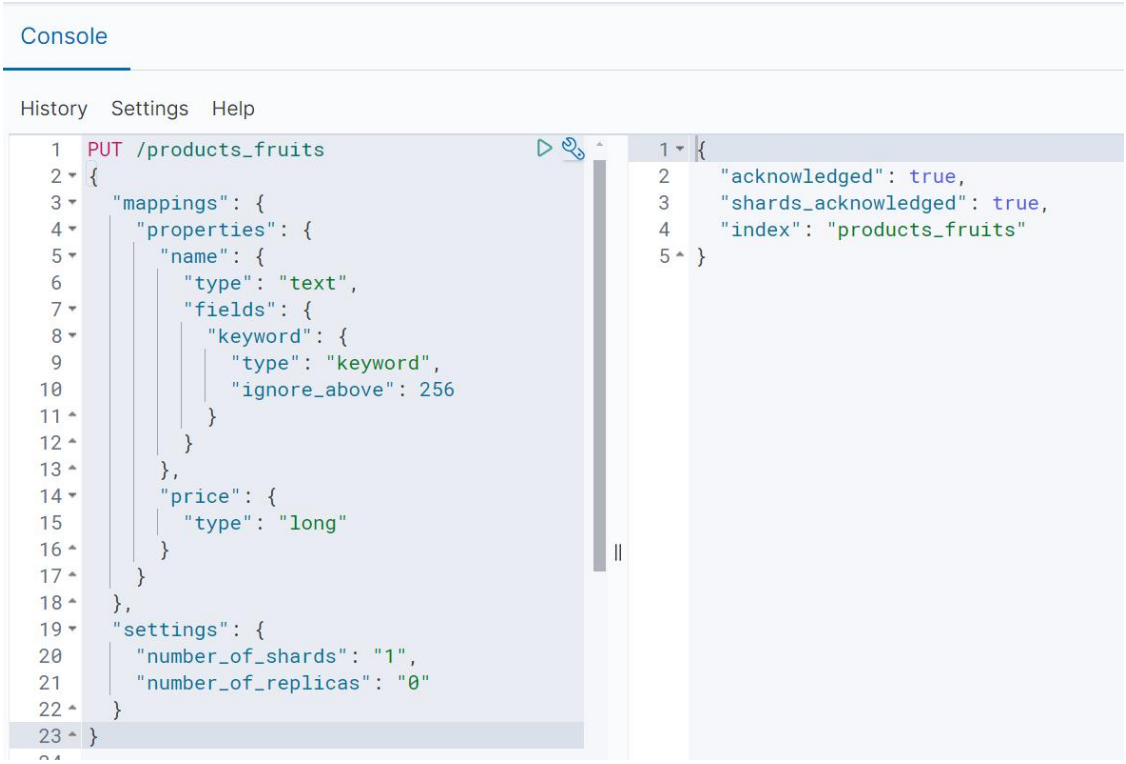
ElasticSearch 实例支持多种导入方式，常用的业务场景是把 Logstash 作为源接入搜索实例，在此，为了演示方便，我们以商品检索为例子，从 Kibana 的 DevTools

控制台里调用 Rest API 导入数据。

a. 先在 Kibana 的左边栏里选择“Dev Tools”，进入控制台。

其中左边屏幕为命令行调用 API 栏，右边屏幕为调用结果返回栏。

b. 首先，在 console 界面，我们创建索引 products_fruits 来定义存储数据的 Schema。

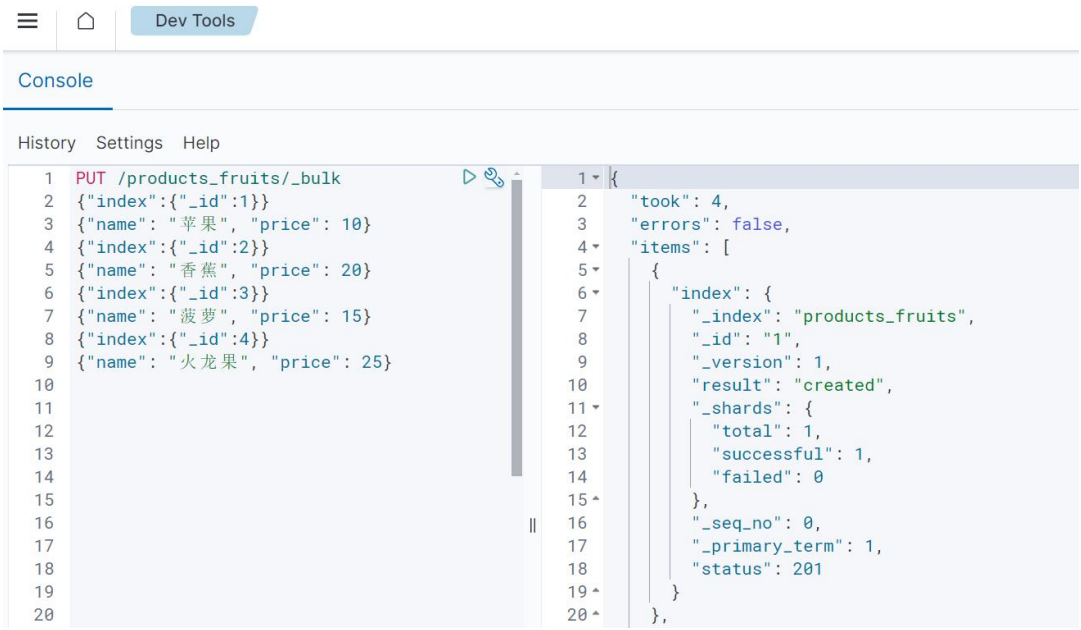


具体语句如下：

```
PUT /products_fruits
{
  "mappings": {
    "properties": {
      "name": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword",
            "ignore_above": 256
          }
```

```
        }
      },
      "price": {
        "type": "long"
      }
    }
  },
  "settings": {
    "number_of_shards": "1",
    "number_of_replicas": "0"
  }
}
```

c. 然后我们在 console 里执行 bulk 插入语句，来将数据导入到索引里。



返回结果里 errors 为 false，表示数据全部导入，具体语句如下：

PUT /products_fruits/_bulk

{"index":{"_id":1}}

{"name": "苹果", "price": 10}

{"index":{"_id":2}}

{"name": "香蕉", "price": 20}

{"index":{"_id":3}}

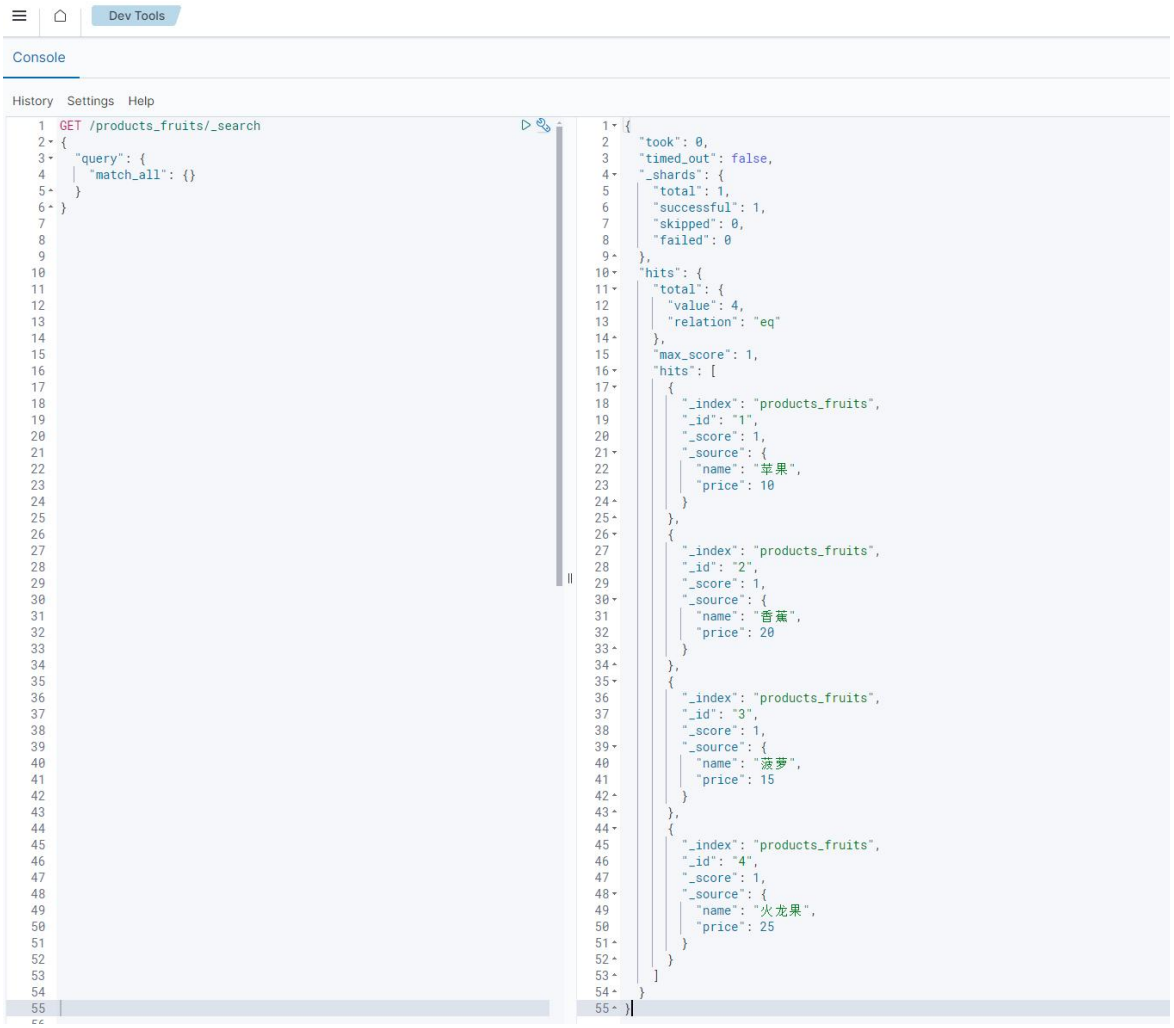
{"name": "菠萝", "price": 15}

{"index":{"_id":4}}

{"name": "火龙果", "price": 25}

3. 检索数据

首先，我们进行基本的全文检索，不设置匹配条件，可以得到所有数据。



检索语句如下：

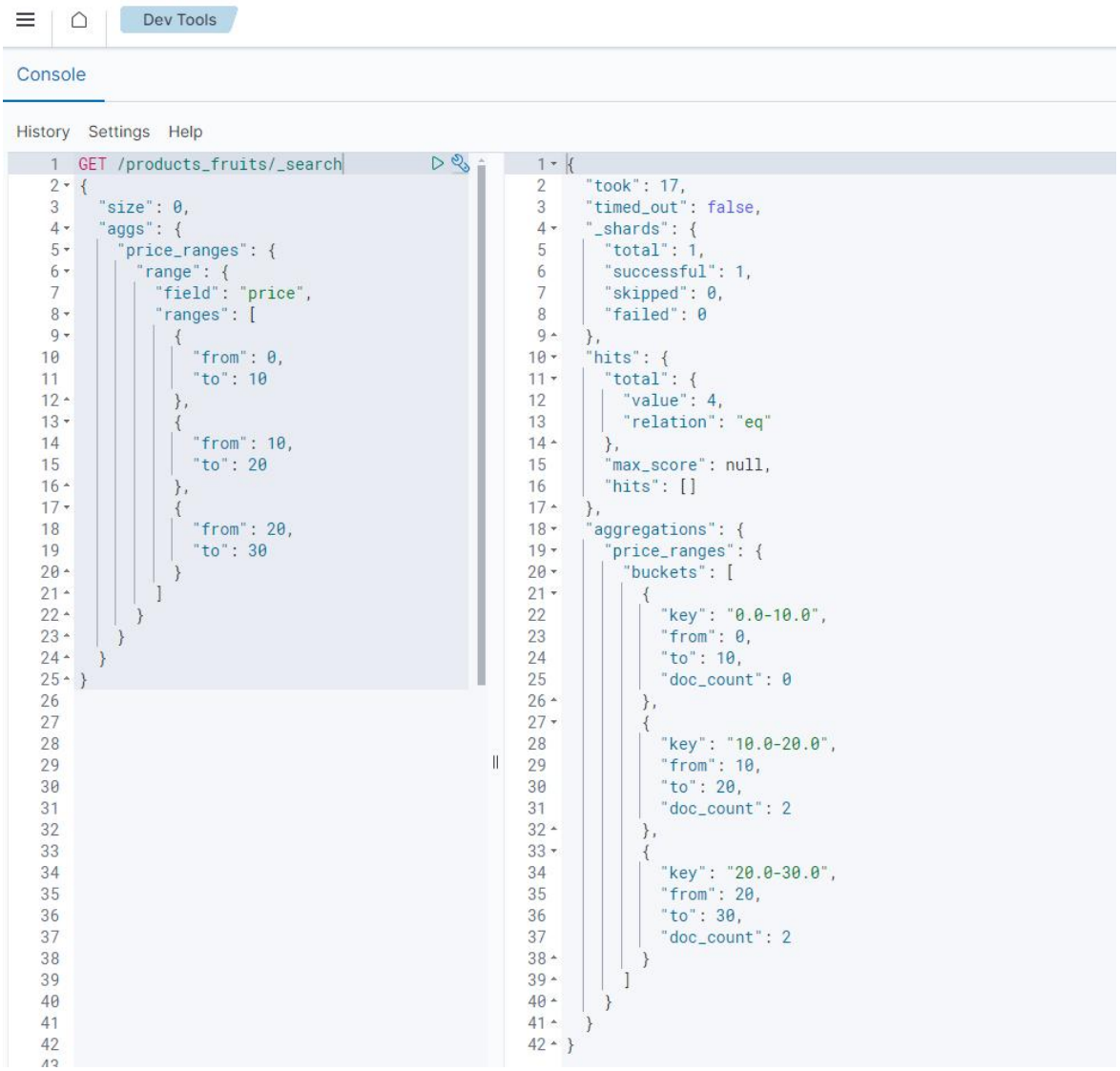
```
GET /products_fruits/_search

{

  "query": {

    "match_all": {}
```

```
  }
   }
```

同样，我们可以设置检索条件、指定分词器、指定评分等多种条件根据 Rest API 的语法对检索的匹配数据进行筛选，得到我们希望的数据。

4. 聚合数据

同样，Elasticsearch 也可以提供数据聚合，我们按照价格区间将商品进行聚合，返回的结果如下：



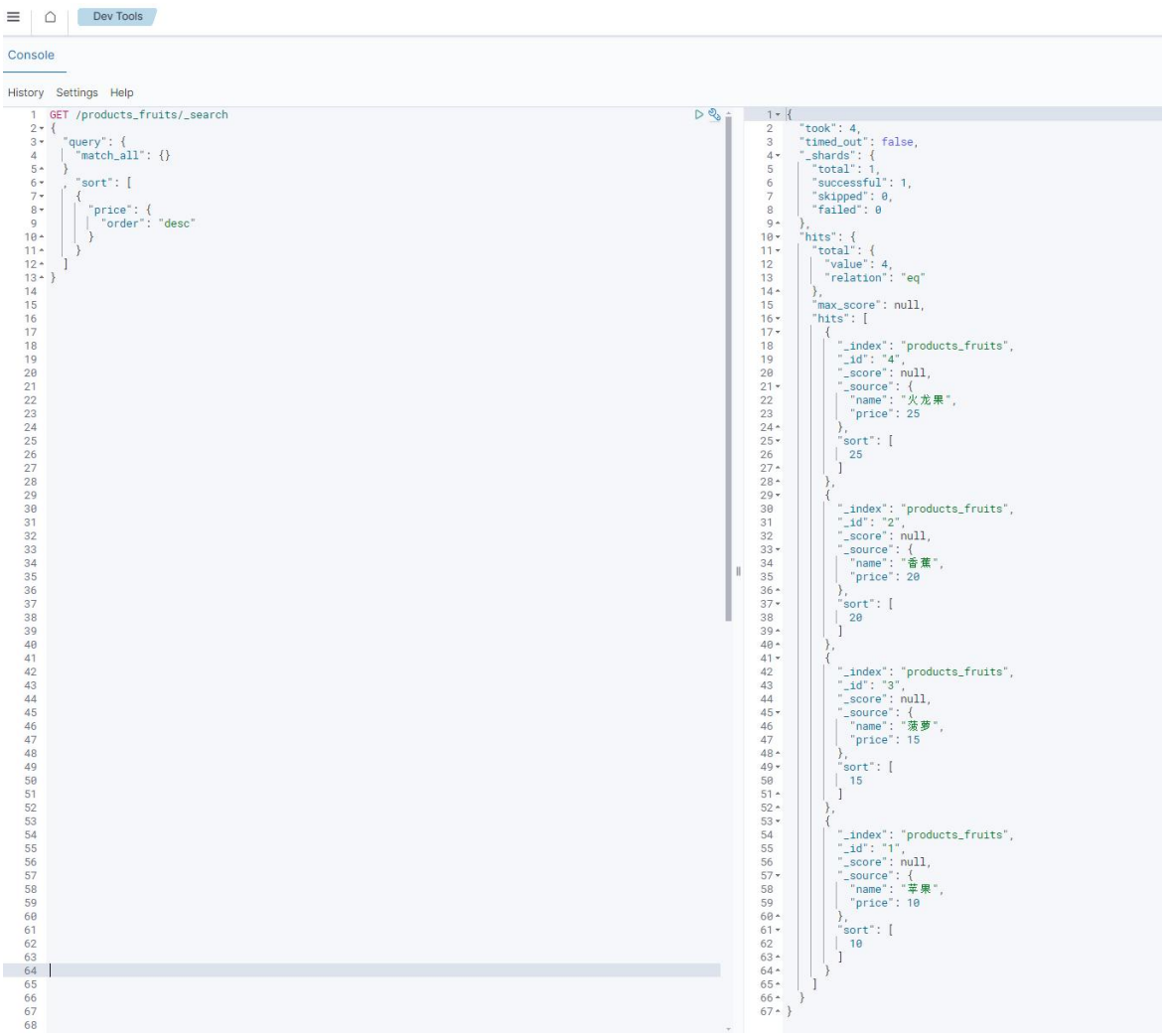
这里的聚合语句如下：

```
GET /products_fruits/_search
{
  "size": 0,
  "aggs": {
```

```
    "price_ranges": {
      "range": {
        "field": "price",
        "ranges": [
          {
            "from": 0,
            "to": 10
          },
          {
            "from": 10,
            "to": 20
          },
          {
            "from": 20,
            "to": 30
          }
        ]
      }
    }
  }
}
```

5. 数据排序

同样，数据结果排序，也是 Elasticsearch 一个重要的用途，我们这里按照价格排序将结果检索出来，右侧边栏按照价格降序进行输出，如下：



排序语句如下：

```
GET /products_fruits/_search
{
  "query": {
    "match_all": {}
  }
  , "sort": [
    {
      "price": {
        "order": "desc"
      }
    }
```

```
]
}
```

6. 删除索引数据

当索引不再使用的时候，我们执行语句删除索引即可。


```
DELETE /products_fruits
```

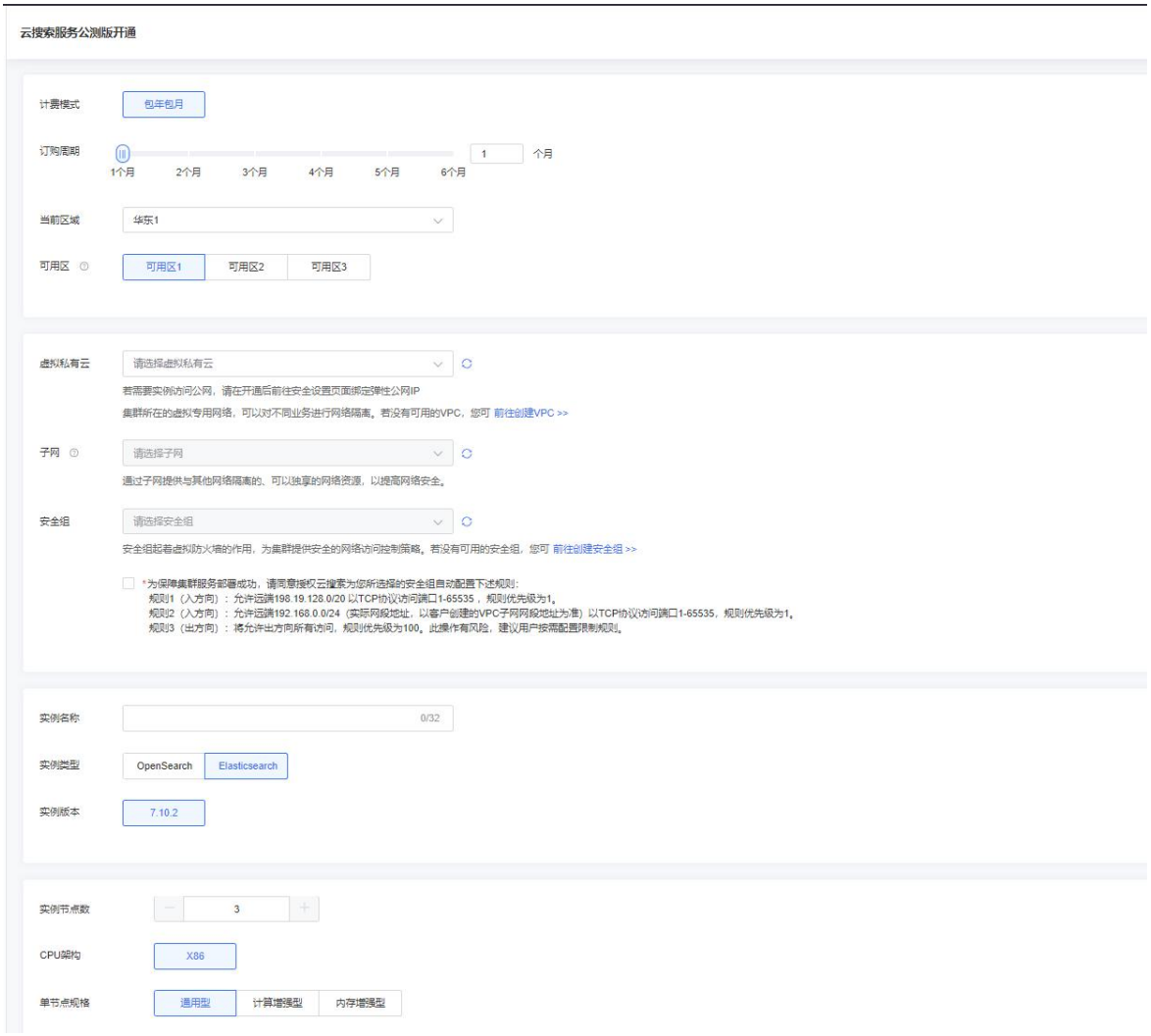
使用 OpenSearch 搜索数据

这里，我们以几个示范的例子，向您简单介绍天翼云 OpenSearch 实例可以提供的检索和分析服务，包括创建索引、数据导入、数据搜索、筛选排序等。

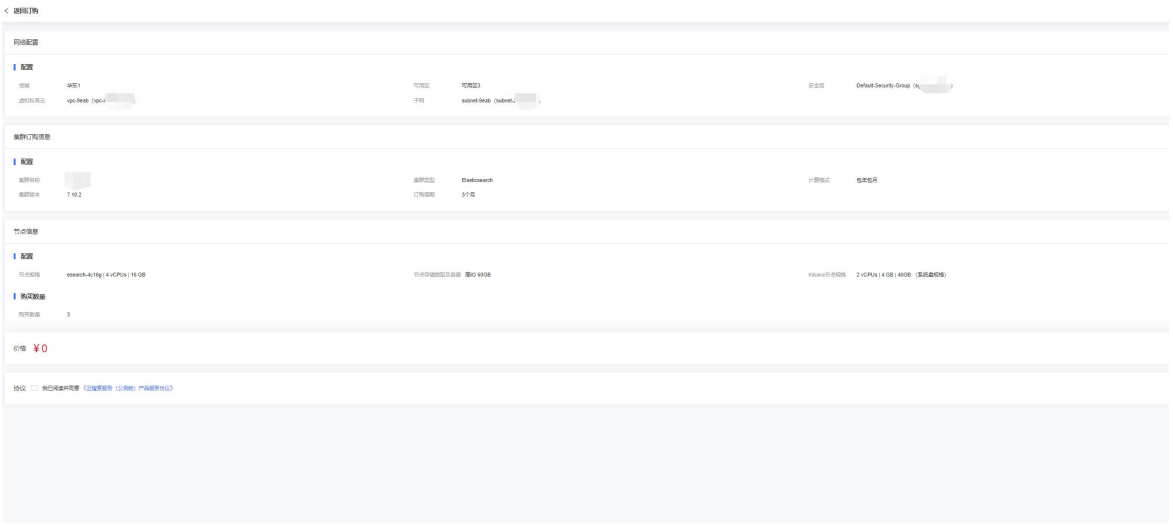
1. 创建实例

a. 在“云搜索服务”产品页，单击“立即开通”。

b. 在云搜索服务实例创建页面选择计费模式、订购周期、当前区域（即资源池）、可用区、实例类型、实例版本、实例名称、节点规格等基础配置信息后，按页面提示并根据需要进行配置，然后点击下一步。



c. 按页面提示，勾选相关协议，公测期间支付 0 元，即可完成订购，等待资源开通完成，对应实例处于“运行中”状态，即为成功。



2. 导入数据

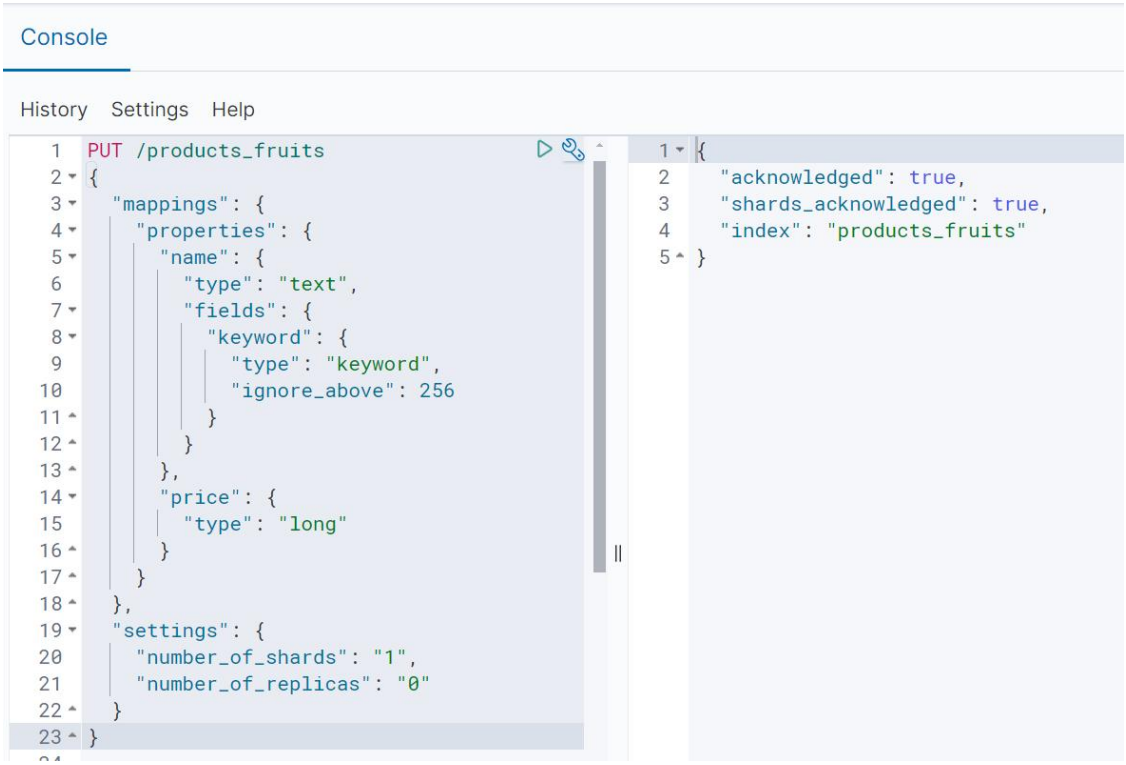
OpenSearch 实例支持多种导入方式，常用的业务场景是把 Logstash 作为源接入搜索实

例，在此，为了演示方便，我们以商品检索为例子，选择从 Dashboards 的 DevTools 控制台里调用 Rest API 导入数据。

a. 先在 Dashboards 的左边栏里选择“DevTools”，进入控制台。

其中左边屏幕为命令行调用 API 栏，右边屏幕为调用结果返回栏。

b. 首先，在 console 界面，我们创建索引 products_fruits 来定义存储数据的 Schema



具体语句如下：

```
PUT /products_fruits
{
  "mappings": {
    "properties": {
      "name": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword",
            "ignore_above": 256
          }
```

```
        }
      },
      "price": {
        "type": "long"
      }
    }
  },
  "settings": {
    "number_of_shards": "1",
    "number_of_replicas": "0"
  }
}
```

c.然后我们在 console 里执行 bulk 插入语句，来将数据导入到索引里。

返回结果里 errors 为 false，表示数据全部导入，具体语句如下：

PUT /products_fruits/_bulk

{"index":{"_id":1}}

{"name": "苹果", "price": 10}

{"index":{"_id":2}}

{"name": "香蕉", "price": 20}

{"index":{"_id":3}}

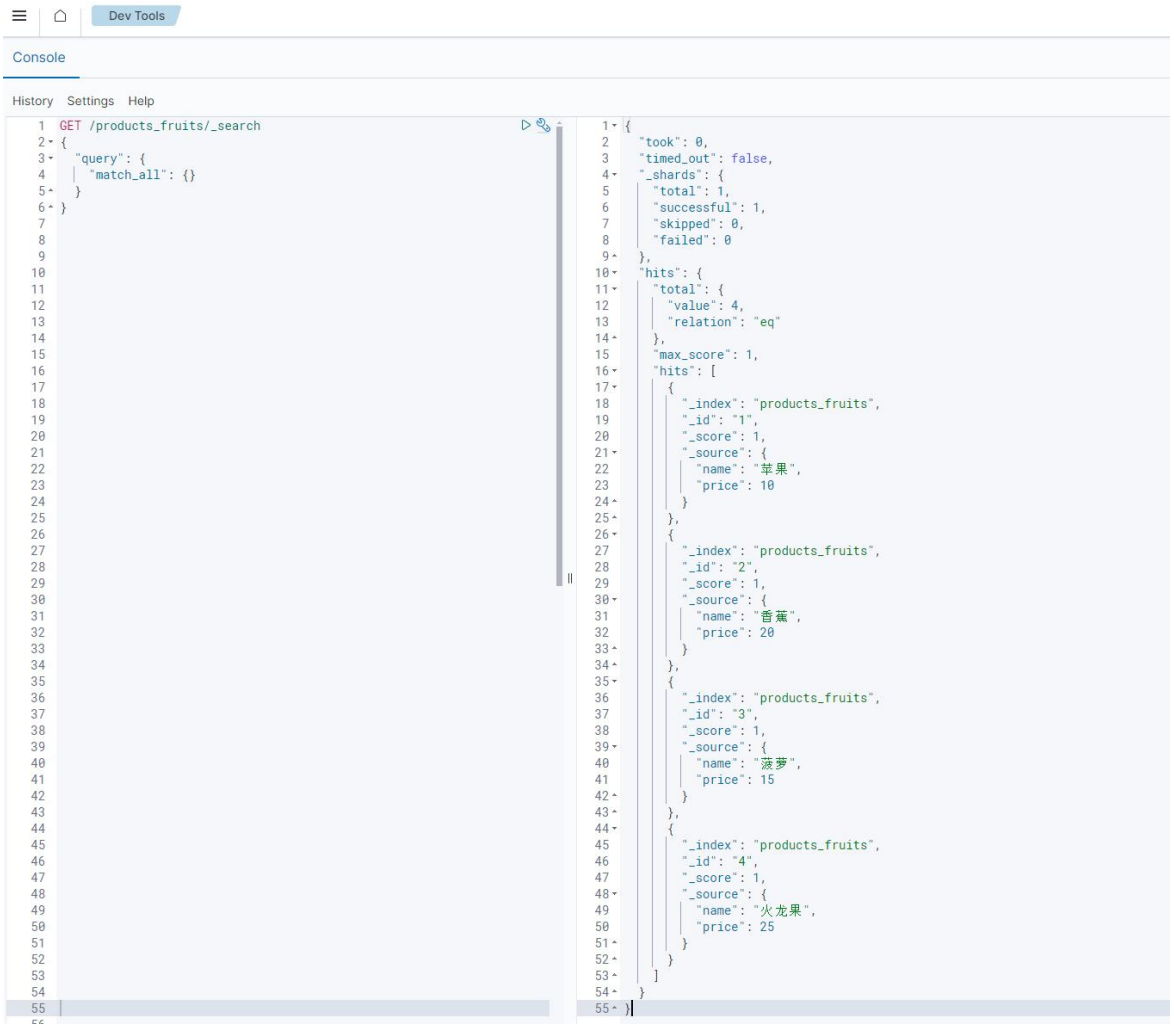
{"name": "菠萝", "price": 15}

{"index":{"_id":4}}

{"name": "火龙果", "price": 25}

3. 检索数据

首先，我们进行基本的全文检索，不设置匹配条件，可以得到所有数据。



检索语句如下：

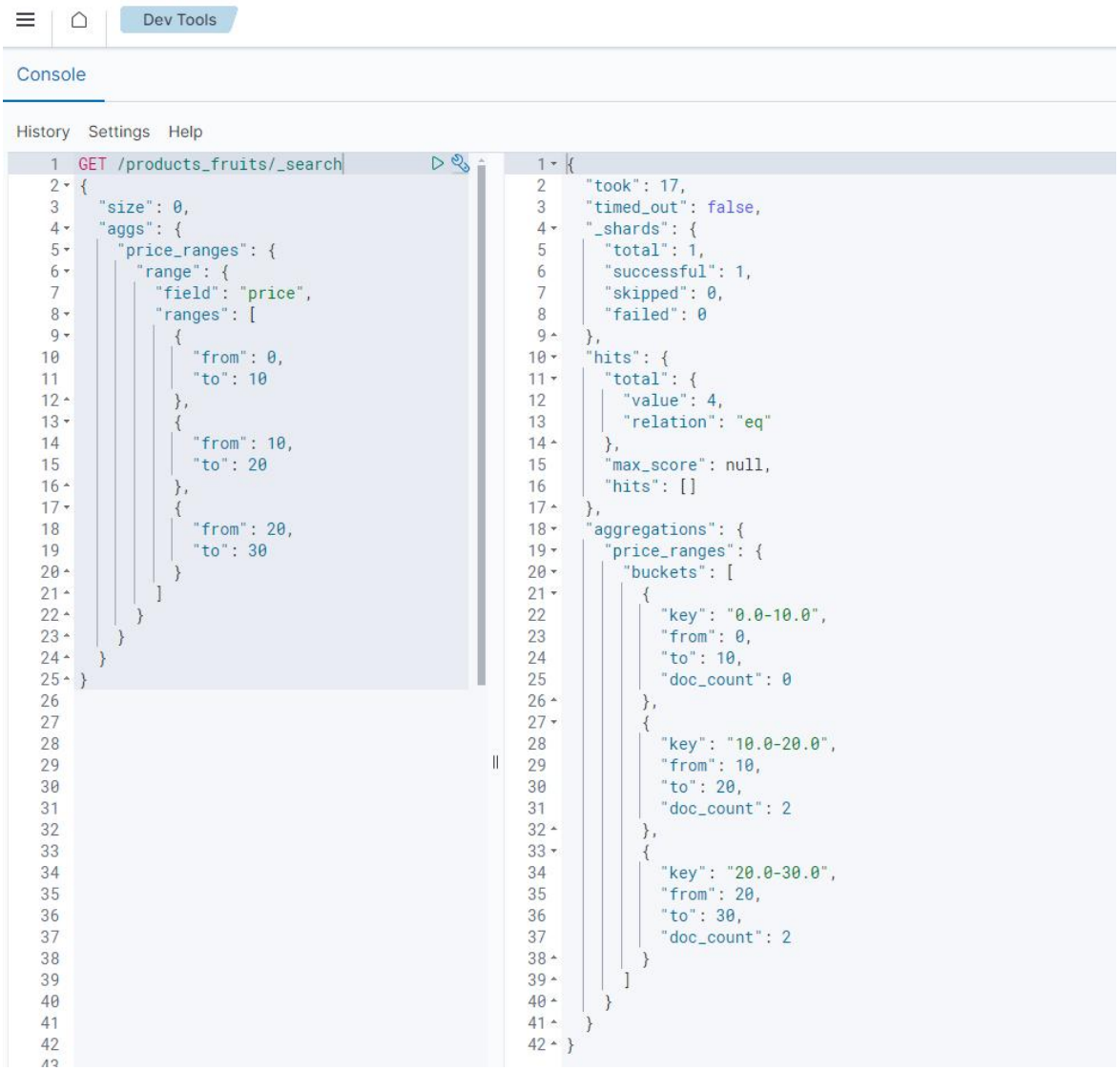
```
GET /products_fruits/_search

{

  "query": {

    "match_all": {}
```

```
  }
   }
```

同样，我们可以设置检索条件、指定分词器、指定评分等多种条件根据 Rest API 的语法对检索的匹配数据进行筛选，得到，我们希望的数据。

4. 聚合数据

同样，OpenSearch 也可以提供数据聚合，我们按照价格区间将商品进行聚合，返回的结果如下：



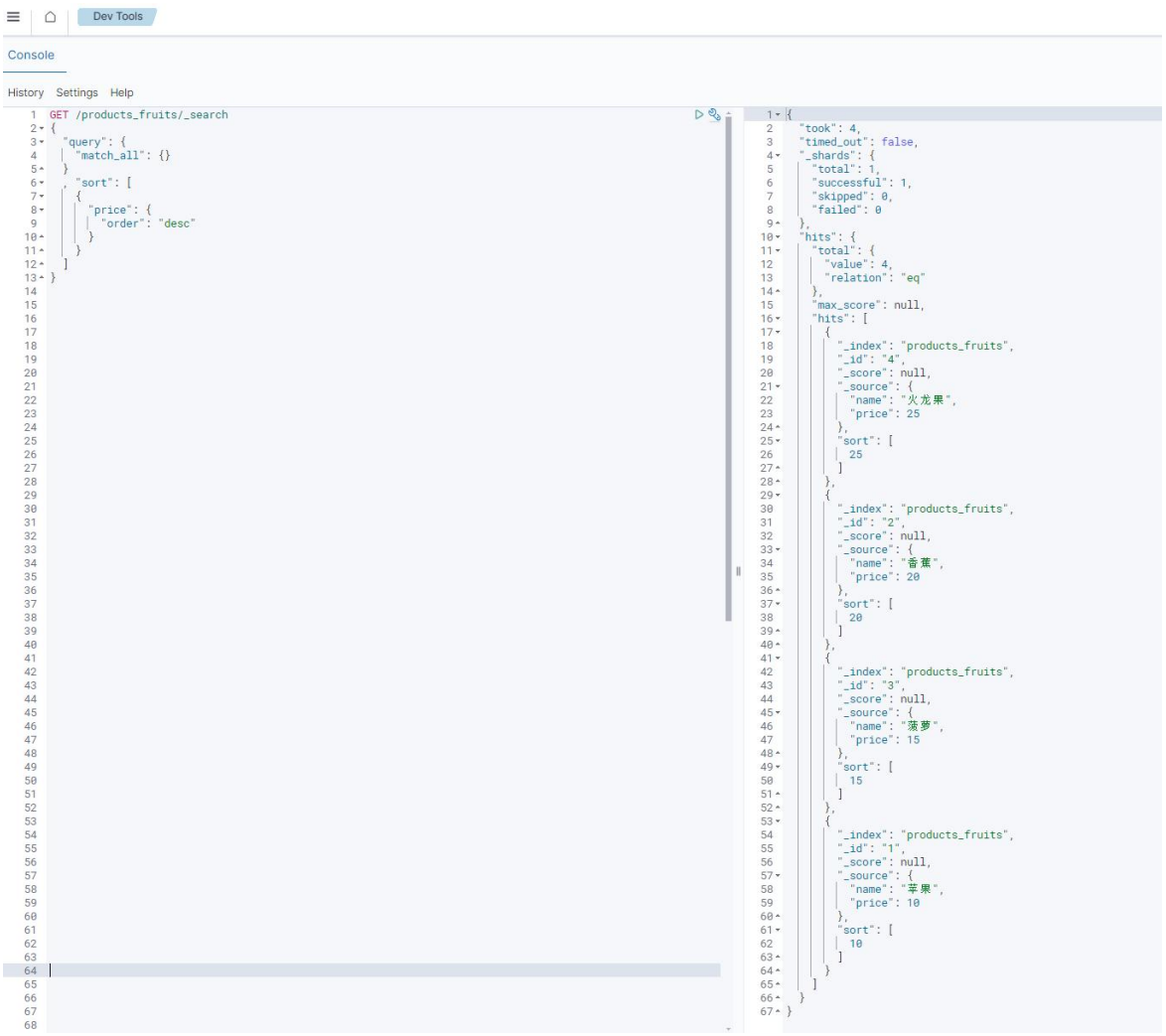
这里的聚合语句如下：

```
GET /products_fruits/_search
{
  "size": 0,
  "aggs": {
```

```
    "price_ranges": {
      "range": {
        "field": "price",
        "ranges": [
          {
            "from": 0,
            "to": 10
          },
          {
            "from": 10,
            "to": 20
          },
          {
            "from": 20,
            "to": 30
          }
        ]
      }
    }
  }
}
```

5. 数据排序

同样，数据结果排序，也是 OpenSearch 一个重要的用途，我们这里按照价格排序将结果检索出来，右侧边栏按照价格降序进行输出，如下：



排序语句如下：

```
GET /products_fruits/_search
{
  "query": {
    "match_all": {}
  }
  , "sort": [
    {
      "price": {
        "order": "desc"
      }
    }
```

```
]
}
```

6. 删除索引数据

当索引不再使用的时候，我们执行语句删除索引即可：

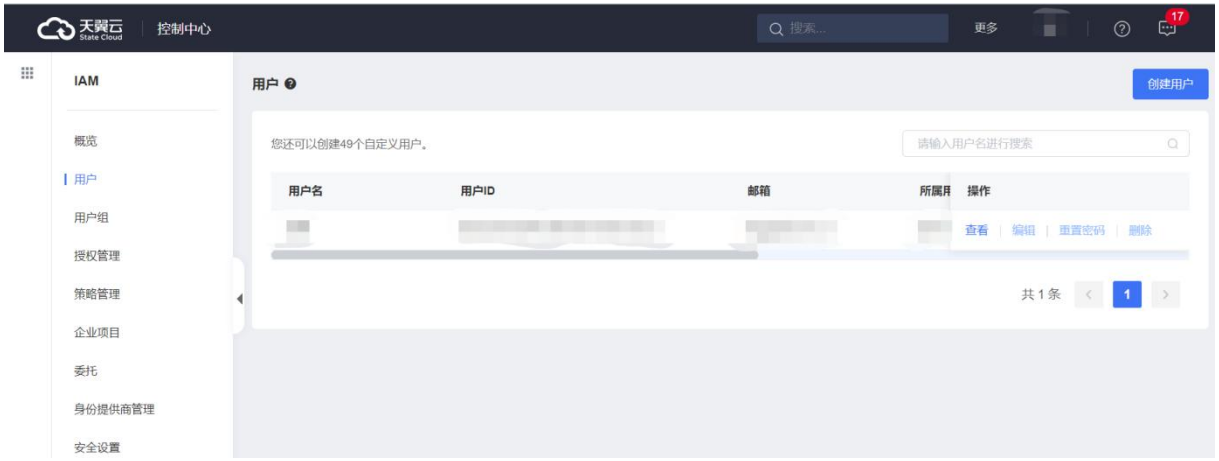
DELETE /products_fruits。

用户指南

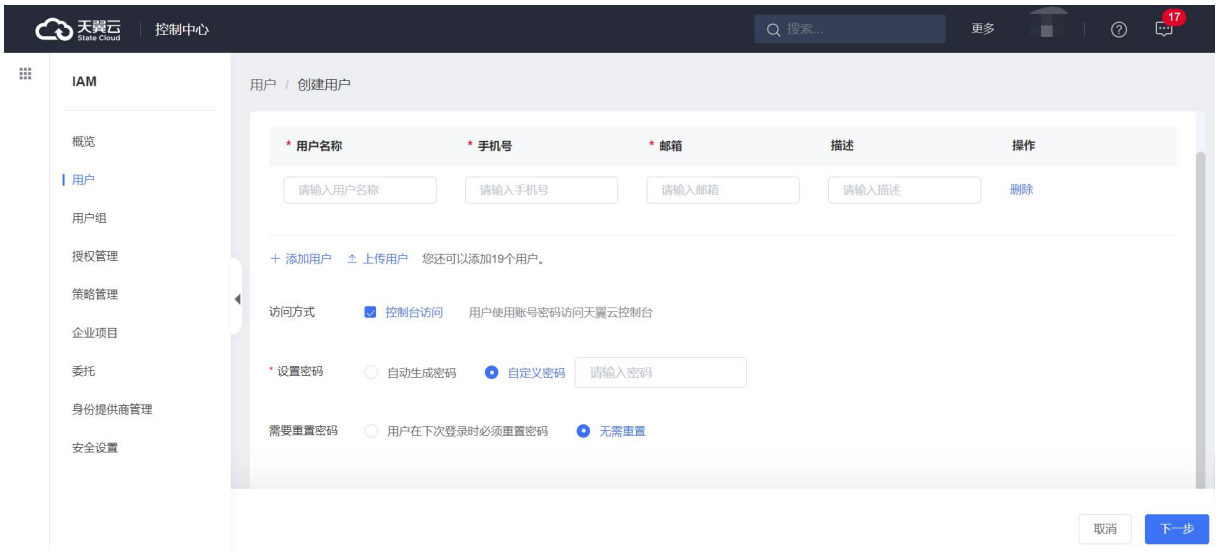
权限管理

控制台授权

您可通过天翼云官网“我的”>“账号中心”>“统一身份认证”进入到天翼云官网的 IAM 服务中。角色点击“用户”>“创建用户”来为当前账号下的 IAM 账号。



根据页面提示输入信息，完成 IAM 用户创建。



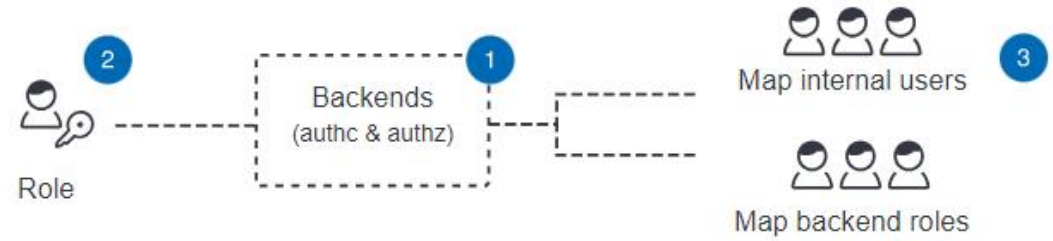
IAM 账号可拥有与主账号一致的控制台访问权限，如需为云搜索实例创建用户，请参见云搜索服务内实例用户及权限。

云搜索服务内实例用户及权限

系统默认有 admin 用户，在 admin 用户下，可以创建不同用户。

用户就是数据安全、访问安全层面的管理。

账号密码提供了身份认证（authc），通过角色的映射实现了授权（authz），两者共同实现了多用户下的安全控制。



Elasticsearch 和 OpenSearch 的此项功能类似，下面以 OpenSearch Dashboards 为例说明：

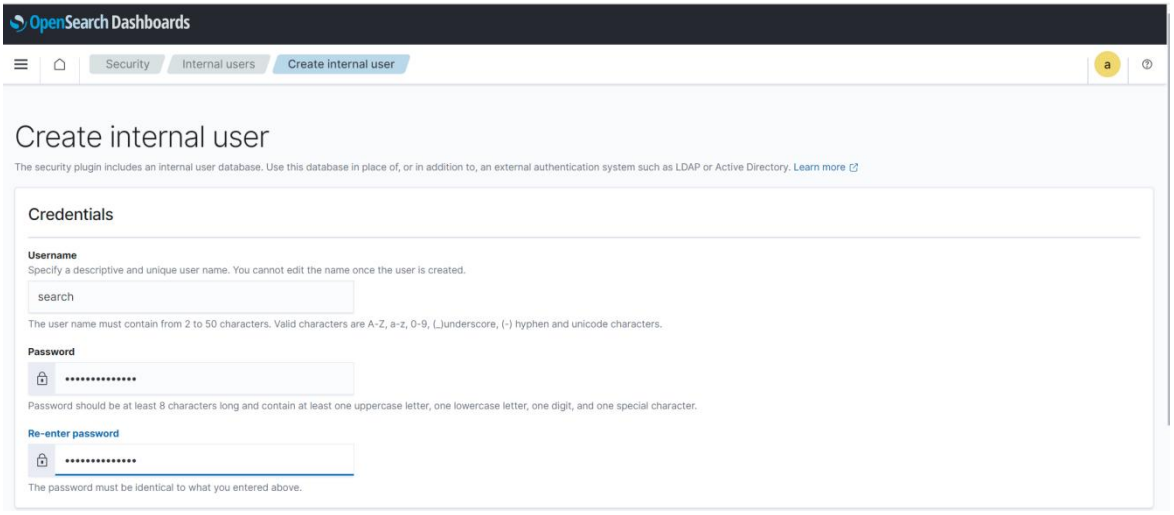
利用 OpenSearch Dashboards 的左边栏 Security 菜单中，用户可以实现集群、索引、文档、字段等不同粒度的访问权限管控。并且提供灵活的用户删除、用户和角色的绑定和解绑。

下面，我们就以创建新用户并赋予它一个具备一定的访问权限的角色为例：

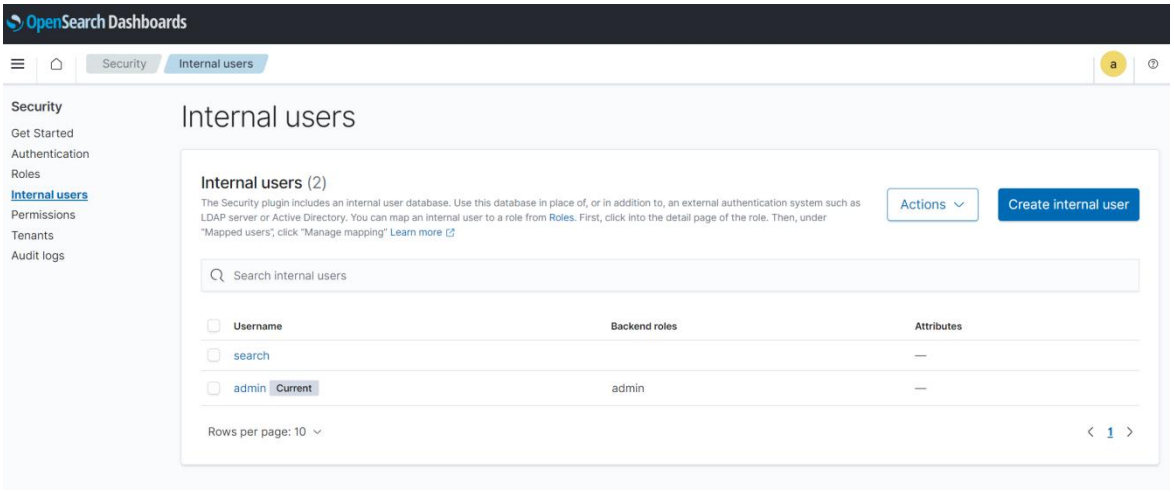
首先，我们必须登录 admin 账号。

1. 创建用户 Internal users

点击左边栏 Security 后，选择右上角 Create internal user 来创建用户。在页面中，我们输入用户名密码，并点击右下角“Create”创建用户。



用户创建完成后，我们自动返回 Internal users 界面，可以看到 search 用户已经创建完成。



2. 创建角色

角色通过索引、实例多维度的精细访问控制，实现对实例、索引权限的安全组划分。

如果复用已有的默认角色（不可删除），则无需创建，可以跳过此步骤。下面我们将演示如何自定义角色。

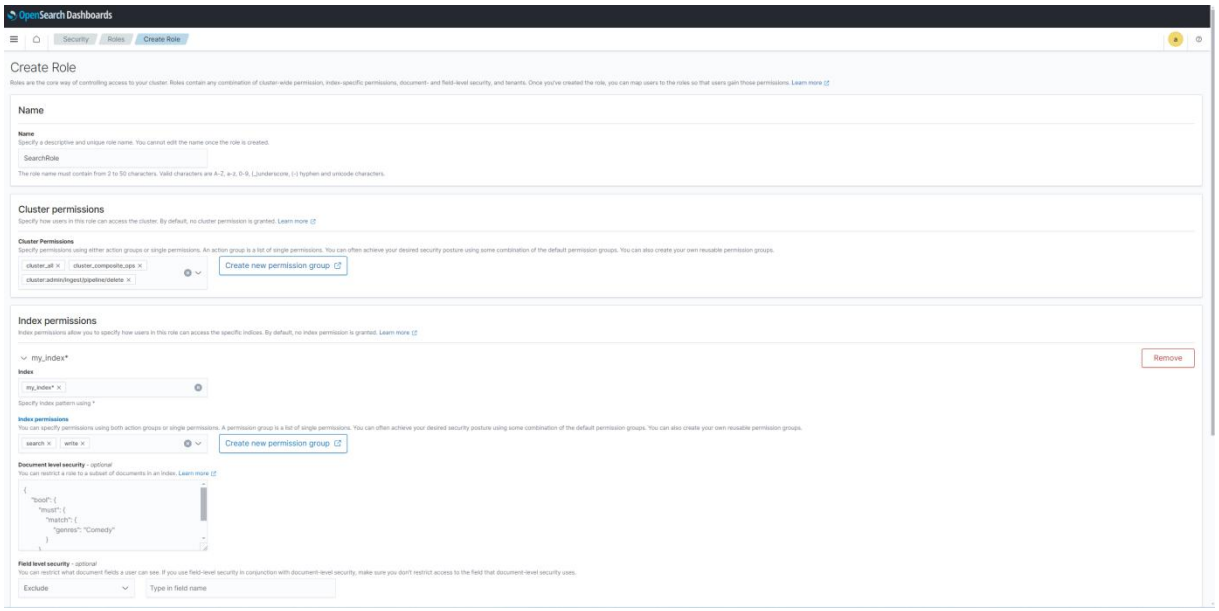
首先，登录左边栏 Security 界面，选择 Roles，并且在右上角点击 Create role：

a. 我们分别创建了角色名 SearchRole，并且，在 Cluster permission 中给它赋予了相应的安全组权限。

b. 我们给它设定了索引级别的更细粒度的 Index permission。

c. 下面还可以设置 Document 和 Field 维度的进一步细粒度的权限控制。

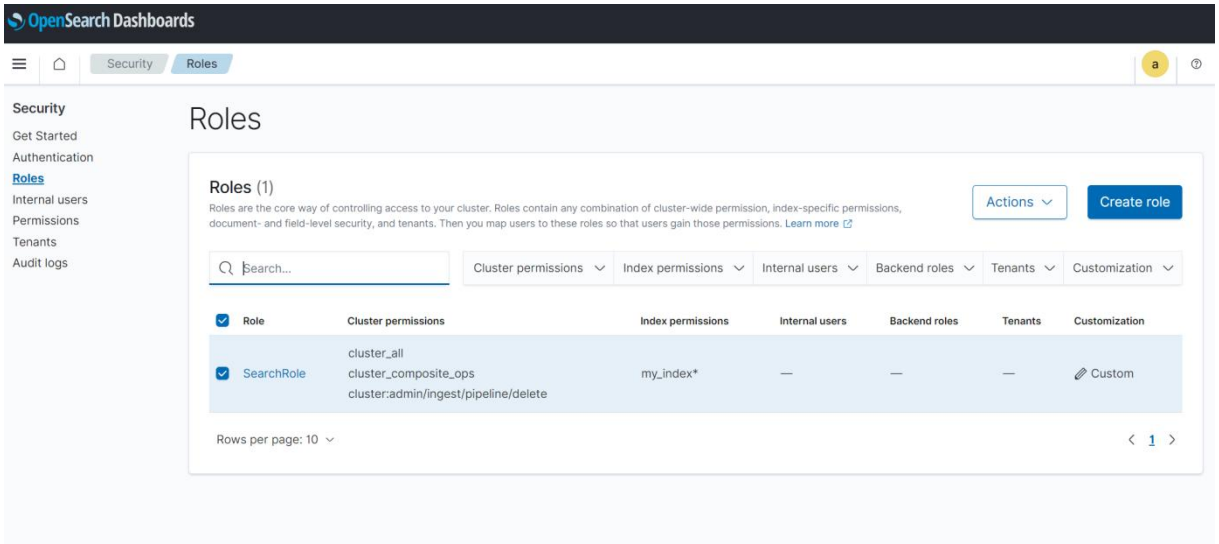
d. 点击右下角的 Create ，就可以创建好角色 SearchRole。



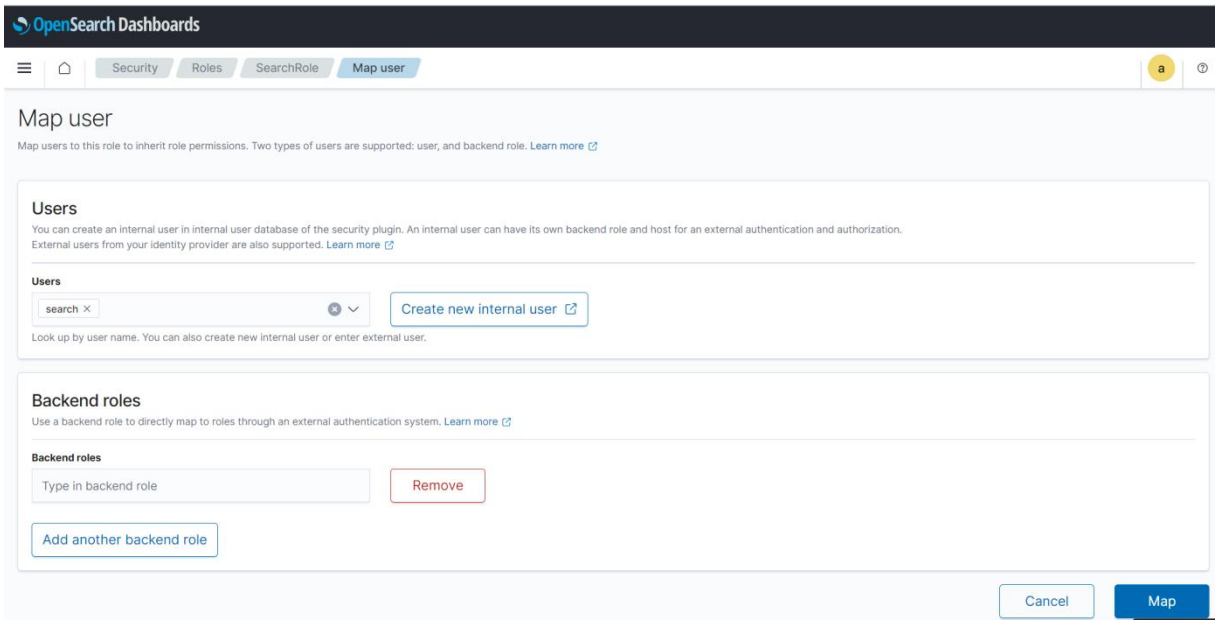
3、映射用户和角色

将创建的用户和角色绑定。

首先，登录左边栏 Security 界面，选择 Roles，并找到刚创建好的角色 SearchRole。



点击角色进入，并选择 Mapped users。将我们创建的用户，映射到这个角色上，并点击右下角 Map 完成映射。



这样，我们就可以用赋予了新角色的账号直接登录了。

Elasticsearch 实例创建及使用

创建 Elasticsearch 实例

前提条件

1. 已在官网完成用户账号注册和实名认证。
2. 已完成实例规划。

操作步骤

您有两种路径可以进入 Elasticsearch 实例开通页面。

1. 登录官网，您可在云搜索服务产品详情页点击“立即开通”；

2. 登录官网，进入云搜索服务控制台后，点击“创建实例”可以进入。

订购页面填写

1. 进入订购页面后，您需要对如下信息进行填写/选择：

参数类型	参数	说明
基础订购信息	计费模式	实例目前仅支持包年/包月模式。 包年/包月：根据实例购买时长，一次性支付实例费用。公测期间最短时长为 1 个月，最长时长为 6 个月。如公测到期时间早于实例的订购到期时间，则有效时间截止至公测到期时间。

	订购周期	在包年包月模式下，您需要选择购买时长。
	当前区域	选择实例的所在区域。 不同区域的云服务产品之间内网互不相通。请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。
	可用区	选择实例所在工作区域下关联的可用区。
网络配置	虚拟私有云	指定实例所在的虚拟专用网络（VPC），实现不同业务的网络隔离。如您没有合适的 VPC，请前往创建虚拟私有云页面创建完成后，回当前页面刷新后选择。
	子网	实例使用子网实现与其他网络的隔离，并独享所有网络资源，以提高网络安全。 选择当前虚拟私有云下实例需要的子网。
	安全组	安全组为实例提供安全的网络访问控制策略。 为了顺利部署实例，我们将为您选择的安全组默认配置下述规则： 规则 1（入方向）：允许远端 198.19.128.0/20 以 TCP 协议访问端口 1-65535 ，规则优先级为 1。 规则 2（入方向）：允许远端 192.168.0.0/24（实际网段地址，以客户创建的 VPC 子网网段地址为准）以 TCP 协议访问端口 1-65535，规则优先级为 1。 规则 3（出方向）：将允许出方向所有访问，规则优先级为 100。
配置实例	实例名称	您可自定义实例名称，可输入的字符范围为长度为 0~32 个字符，只能包含数字、字母、中划线（-）和下划线（_），且必须以字母开头。
	实例类型	云搜索服务支持 Elasticsearch 和 OpenSearch 两种组件。
	实例版本	选择所需的实例版本，支持的版本以页面可选项为准。
选购资源	实例节点数	实例中需要部署节点数，公测期为小规模实例，请以页面可下单数量为准。

	CPU 架构	目前支持 X86 类型。
	节点规格	实例的节点规格，可按需选购，同一实例仅支持一种规格，请确认后下单。
	节点存储规格	选择存储类型，支持的类型可能随资源池不同有差异，请以页面可选的为准。
	单节点存储量	您可根据业务数据容量评估，选购合适的单节点存储量，创建完成后，不支持缩容节点存储容量，请基于业务量合理选择容量。
	Kibana 节点规格	云搜索服务会默认为您开通一个小规格节点部署 Kibana 节点。该节点正常计费，公测期免费。
实例密码设置	实例密码设置	设置 Elasticsearch 的管理员访问密码； 实例内账号和角色请登录 Kibana 进行设置。

2.信息填写完毕后，进入下一步信息确认页；请您仔细核对订购信息及订购协议，勾选协议后进入下一步即可提交。

3.缴费完成后，请返回购买实例对应的实例管理列表页面，您购买的实例都将展示在此，当实例从“创建中”变为“运行中”时，即可使用实例。

如实例创建失败，系统将自动退单销毁，期间不会记录使用时长，不会收取费用，您可再次下单尝试，或工单联系工程师为您处理。

访问 Elasticsearch 实例

概览

天翼云云搜索服务 Elasticsearch 实例支持多种连接方式：

Kibana 访问

Elasticsearch 实例支持通过 Kibana 前端可视化界面连接。用户可通过登录页面输入用户名和密码进行访问，默认用户名为 admin，密码为创建实例时设置的管理员密码。

Curl 命令行方式

在开启安全认证的情况下，用户可以通过 Curl 命令行与 Elasticsearch 实例进行交互。例如，可以使用 curl 发送 HTTP 请求来执行各种操作，如创建索引、查询数据、更新文档等。这种方式适合快速测试和管理实例。

Python client 访问

支持通过官方的 Elasticsearch Python 客户端（elasticsearch-py）访问实例。Python 客户端提供丰富的 API，用于查询、索引、删除数据，适合需要通过 Python 脚本进行大规模数据处理、分析以及自动化管理的场景。

Java client 访问

提供对 Elasticsearch 的 Java API 支持，使用 RestHighLevelClient 类与实例进行交互。Java 客户端广泛应用于企业级 Java 应用中，适合复杂的集成场景，如与 Spring 框架结合进行实时搜索和分析。

Go client 访问

提供轻量且高效的连接方式，适合基于 Go 语言开发的高并发、低延迟的应用。Go 客户端提供对 Elasticsearch REST API 的完整支持，适合需要高性能和可扩展性的场景。

通过 Curl 命令行接入 Elasticsearch 实例

概述

使用 Curl 是最简单直接的方式访问 Elasticsearch 实例。它允许用户通过命令行发送 HTTP 请求与集群进行交互，例如创建索引、查询集群状态等操作。

前提条件

已开通天翼云云搜索服务 Elasticsearch 实例。

实例已绑定公网 IP。具体可参考“实例公网访问”章节。

本地已安装 Curl 工具。

操作步骤

使用以下 Curl 命令访问 Elasticsearch 集群：

curl -u <user>:<password> "http://<host>:<port>"

<user>：Elasticsearch 集群用户名，比如 admin。

<password>：该用户密码，比如用户配置的 Elasticsearch 集群 admin 用户密码。

<host>：主机 IP，即集群绑定的公网 IP。

<port>：端口号，一般是 9200。

通过 Java 客户端接入 Elasticsearch 实例

概述

Java 是官方推荐的编程语言之一，使用 Elasticsearch 提供的 Java REST 客户端可以轻松与集群进行交互，包括索引管理、数据查询、插入文档等操作，适用于构建基于 Java 的大规模应用。

前提条件

已开通天翼云云搜索服务 Elasticsearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已在本地安装了 JDK（推荐 JDK 8 及以上版本）。

已配置 Maven 或 Gradle 项目依赖以支持 Elasticsearch Java 客户端。

操作步骤

在项目中引入 Elasticsearch 客户端依赖。Maven 依赖配置如下：

```
<dependency>
  <groupId>org.elasticsearch.client</groupId>
  <artifactId>elasticsearch-rest-high-level-client</artifactId>
  <version>7.10.2</version>
</dependency>
```

使用以下代码连接到 Elasticsearch 集群：

```
import org.apache.http.HttpHost;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestHighLevelClient;
public class ElasticsearchJavaClient {
    public static void main(String[] args) {
        // 初始化客户端
        RestHighLevelClient client = new RestHighLevelClient(
            RestClient.builder(new HttpHost("<host>", 9200, "http"))
                    .setDefaultCredentialsProvider(new
BasicCredentialsProvider().setCredentials(
                            AuthScope.ANY, new
UsernamePasswordCredentials("<user>", "<password>")
                    )));
```

```
        // 执行操作，例如创建索引等
        // ...
        // 关闭客户端
        client.close();
    }
}
```

<host>：集群绑定的公网 IP。

<user>：Elasticsearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

在执行具体操作时，例如创建索引：

```
CreateIndexRequest request = new CreateIndexRequest("my_index");
CreateIndexResponse createIndexResponse = client.indices().create(request,
RequestOptions.DEFAULT);
```

操作完成后关闭客户端：

```
client.close();
```

通过 Python 客户端接入 Elasticsearch 实例

概述

Python 客户端（elasticsearch-py）是 Elasticsearch 官方提供的库，允许开发者通过简单的 Python 代码与集群交互，支持查询、插入、删除索引等操作，适用于快速开发和轻量级的应用场景。

前提条件

已开通天翼云云搜索服务 Elasticsearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已在本地安装 Python 3.x 版本。

已安装 Elasticsearch 官方 Python 客户端库。

操作步骤

安装 Python 客户端库：

```
pip install elasticsearch
```

使用以下代码连接到 Elasticsearch 集群：

```
from elasticsearch import Elasticsearch

# 连接到 Elasticsearch 集群

es = Elasticsearch(

    hosts=["http://<host>:9200"],

    http_auth=("<user>", "<password>")

)

# 创建索引操作

es.indices.create(index="my_index", ignore=400)
```

<host>：集群绑定的公网 IP。

<user>：Elasticsearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

执行查询操作：

```
response = es.get(index="my_index", id=1)

print(response)
```

通过 Go 客户端接入 Elasticsearch 实例

概述

Go 客户端（elasticsearch-go）是 Elasticsearch 官方提供的 Golang 库，适用于构建高性能的应用程序。它提供了与 Elasticsearch 集群进行交互的完整 API，支持索引创建、数据查询等操作。

前提条件

已开通天翼云云搜索服务 Elasticsearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已安装 Go 语言开发环境。

已安装 Elasticsearch 官方 Go 客户端库。

操作步骤

安装 Go 客户端库：

```
go get github.com/elastic/go-elasticsearch/v7
```

使用以下代码连接到 Elasticsearch 集群：

```
package main
```

```
import (
    "context"
    "fmt"
    "log"
    "github.com/elastic/go-elasticsearch/v7"
)
func main() {
    // 创建 Elasticsearch 客户端
    es, err := elasticsearch.NewClient(elasticsearch.Config{
        Addresses: []string{"http://<host>:9200"},
        Username: "<user>",
        Password: "<password>",
    })
    if err != nil {
        log.Fatalf("Error creating the client: %s", err)
    }
    // 创建索引
    res, err := es.Indices.Create("my_index")
    if err != nil {
        log.Fatalf("Error creating index: %s", err)
    }
    fmt.Println(res)
}
```

<host>：集群绑定的公网 IP。

<user>：Elasticsearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

导入数据至 Elasticsearch 实例

Elasticsearch 实例数据导入方式

在 Elasticsearch 中，导入数据是一项核心操作，可以通过多种方式来实现。

天翼云云搜索 Elasticsearch 实例中，有多种方式可以导入数据，常见的几种导入方式包括：Elasticsearch 客户端、Beats、Logstash、Kibana 或其他数据导入工具。可以根据实际的需求来选择合适的方式来导入数据至 Elasticsearch 实例。

支持的导入方式如下表：

导入方式	适用场景
Elasticsearch 客户端	适合开发者，处理复杂业务逻辑和批量数据操作。
Beats	适合实时日志、指标的轻量级数据采集。
Logstash	适合复杂的数据处理管道和多源数据整合。
Kibana	适合快速测试、调试和简单数据管理。
Curl 命令行	适合轻量、临时和自动化脚本操作。

这里主要介绍使用 Elasticsearch 客户端、Beats、Logstash、Kibana、Curl 命令行的方式导入数据至 Elasticsearch 实例。

您也可以通过自行开发或者适用第三方数据导入工具将各种数据导入到 Elasticsearch 实例。

使用 Elasticsearch 客户端导入数据至 Elasticsearch 实例

Elasticsearch 提供官方的客户端库，支持多种编程语言，如 Java、Python、JavaScript 等。

适用场景

编程场景：当你有自定义应用程序，需要通过代码直接与 Elasticsearch 交互时，Elasticsearch 客户端提供了灵活的 API 进行复杂查询和批量导入数据。

批量数据导入：通过客户端库可以实现大规模数据的分块导入，并发写入，适用于处理大数据量的场景。

动态数据处理：如果数据在导入前需要复杂的逻辑处理，可以通过编程语言和客户端实现定制的数据流。

前提条件

已经开通天翼云云搜索 Elasticsearch 实例。

能够通过 HTTP 访问 Elasticsearch 实例。

1. 客户端使用实例。

这里以 Python 和 Java 客户端为例。

a. 使用 Python 客户端 (elasticsearch-py)。

Python 客户端 elasticsearch-py 是一个与 Elasticsearch 交互的轻量级库。使用它，你可以通过 index 方法将数据导入到指定索引中。

```
from elasticsearch import Elasticsearch

# 创建 Elasticsearch 客户端

es = Elasticsearch("http://ip:9200")

# 要导入的数据

data = {

    "title": "Elasticsearch 入门",

    "content": "Elasticsearch 是一款分布式搜索引擎...",

    "date": "2024-08-23"

}

# 将数据导入到名为 "articles" 的索引

response = es.index(index="articles", document=data)

print(response)
```

b. 使用 Java 客户端。

Java 是 Elasticsearch 的主要编程语言之一，其官方客户端提供了丰富的功能。以下示例展示了如何使用 Java 客户端导入数据：

```
import org.elasticsearch.client.RestClient;

import org.elasticsearch.client.RestHighLevelClient;

import org.elasticsearch.client.RequestOptions;

import org.elasticsearch.action.index.IndexRequest;

import org.elasticsearch.action.index.IndexResponse;

import org.elasticsearch.common.xcontent.XContentType;

public class DataImporter {
```

```
    public static void main(String[] args) throws Exception {
        RestHighLevelClient client = new RestHighLevelClient(
            RestClient.builder(new HttpHost("ip", 9200, "http"))
        );
        String jsonString = "{" +
                "\"title\":\"Elasticsearch 入门\"," +
                "\"content\":\"Elasticsearch 是一款分布式搜索引擎...\"," +
                "\"date\":\"2024-08-23\"" +
                "}";
        IndexRequest request = new IndexRequest("articles");
        request.source(jsonString, XContentType.JSON);
        IndexResponse response = client.index(request,
RequestOptions.DEFAULT);
        System.out.println(response.getId());
        client.close();
    }
}
```

使用自建 Beats 导入数据至 Elasticsearch 实例

Beats 是轻量级的数据收集器，专门用于将各种日志、指标、网络数据发送到 Elasticsearch。常用的 Beats 包括 Filebeat、Metricbeat、Packetbeat 等。

Filebeat 是一种轻量级日志收集器,通常用于将文件系统中的日志文件或事件日志发送到 Elasticsearch 或 Logstash。它适合简单的日志采集场景，能够有效处理系统、应用程序日志等。本文将以 Filebeat 为例，导入数据至 Elasticsearch 实例。

适用场景

轻量数据收集：适用于需要从大量分布式系统、服务器、容器中收集日志、监控指标等情况。实时日志监控：Beats 能够实时收集日志并传送到 Elasticsearch，是实时日志分析场景的理想选择。

低延迟要求的场景：Beats 设计为轻量级工具，占用资源少，适合资源受限的环境。

前提条件

已经开通天翼云云搜索 Elasticsearch 实例。

已经部署 Filebeat 且打通和 Elasticsearch 实例之间的网络。

1. Filebeat 配置。

Filebeat 采集日志，需要配置 Filebeat 的配置文件，设置具体需要采集的日志路径。

具体根据实际的部署路径配置 filebeat.yml。

```
#采集日志
filebeat.inputs:
- type: log
  # 采集的日志文件的路径。替换为自己日志的路径，可以使用通配符。
  paths:
    - /your_path/*.log
output.elasticsearch:
  # ip 替换为 Elasticsearch 实例的地址。
  hosts: ["http://{ip}:9200"]
  # 传入 Elasticsearch 实例的用户名和密码
  username: "********"
  password: "********"
```

2. 启动 Filebeat 导入数据。

可以使用下面的命令在命令行来启动 Filebeat 导入数据。

```
./filebeat -e -c filebeat.yml
```

使用自建 Logstash 导入数据至 Elasticsearch 实例

Logstash 是一个开源的服务器端实时数据处理工具，支持从多个数据源中提取数据，经过处理后将数据导入到 Elasticsearch 中。它非常适合处理流数据，如日志、监控数据和指标数据。

适用场景

日志数据、监控数据、流数据等。

前提条件

已经开通天翼云云搜索 Elasticsearch 实例。

已经部署 Logstash 且打通和 Elasticsearch 实例之间的网络。

1. Logstash 配置

Logstash 可以接收很多数据源，可以根据实际的需求配置。

这里使用 Filebeat 作为 Logstash 的输入。

配置 your_logstash.conf 管道文件。

Logstash 需要接收 Filebeat 的输出并进行处理，示例配置如下：

```
input {
  beats {
    port => 5044
  }
}
# 对数据进行处理。
filter {
    # mutate {
    #   remove_field => ["@version"]
    # }
}
output{
     elasticsearch{
       # Elasticsearch 实例的访问地址。
       hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}",
"http://{ip}:{port}"]
       # 访问 Elasticsearch 实例的用户名和密码，如无安全机制可不配置。
       user => "********"
       password => "********"
       # 配置写入的索引名,示例如下。
       index => "filebeat-logstash-es-%{+YYYY.MM.dd}"
     }
 }
```

2. 启动 Logstash 导入数据

可以使用下面的命令在命令行来启动 logstash 导入数据。

```
./bin/logstash -f your_logstash.conf
```

使用 Kibana 导入数据至 Elasticsearch 实例

Kibana 提供的 Dev Tools 控制台允许直接在浏览器中通过 RESTAPI 向 Elasticsearch 发出查询和数据操作请求。

Kibana 可视化界面提供了简单的数据导入功能，适用于小规模数据的手动导入场景，特别是在测试和快速验证场景中非常实用。

适用场景

快速测试与调试：适用于开发阶段测试查询语句、创建索引、插入和更新数据等操作。

简单数据管理：如果只是少量数据操作，如插入、更新、删除数据或查看结果，Kibana 提供了方便的 Web 界面操作。

无需编程环境：不需要安装额外的客户端工具，只要能访问 Kibana 即可操作 Elasticsearch。

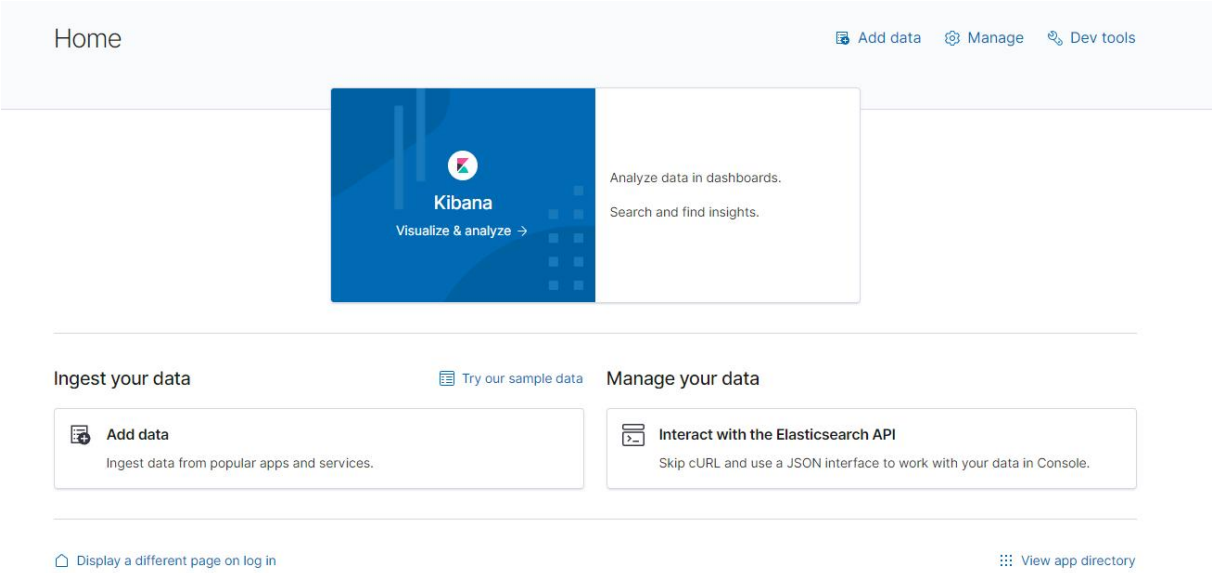
前提条件

已经开通天翼云云搜索 Elasticsearch 实例。

查看 Kibana 的终端可以访问到云搜索实例，设置好 5601 端口的网络安全策略。

实际操作

点击 Kibana 输入用户名登录后，进入首页。



右上角可以点击 Dev tools 工具进入开发工具界面。

如果没有索引，可以创建一个测试索引名为 test_index。

```
PUT /test_index
{
  "mappings": {
    "properties": {
      "mytest": {
        "type": "text"
      }
    }
  }
}
```

执行创建索引操作会返回如下信息。

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "test_index"
}
```

如果有索引可以使用已有的索引。

往索引中写入数据，以上面创建的索引为例。

a. 写入单条数据

可以使用下面命令写入一条 id 为 1 的数据。

```
POST /test_index/_doc/1
{
  "mytest": "xiaoming"
}
```

执行上面的命令，返回下面的结果即导入数据成功。

```
{
  "_index" : "test_index",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

b. 批量写数据

可以使用下面命令写入一批数据。

```
POST /test_index/_bulk
{"index":{"_id": 1}}
{"mytest": "xiaoming"}
{"index":{"_id": 2}}
{"mytest": "xiaohong"}
{"index":{"_id": 3}}
{"mytest": "xiaoli"}
{"index":{"_id": 4}}
{"mytest": "xiaozhang"}
{"index":{"_id": 5}}
{"mytest": "xiaowang"}
```

执行上面的命令，返回下面的结果即导入数据成功。


```
{
  "took" : 10,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "test_index",
        "_type" : "_doc",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
          "successful" : 2,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    },
    ....
    {
      "index" : {
        "_index" : "test_index",
        "_type" : "_doc",
```

```
        "_id" : "5",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
          "successful" : 2,
          "failed" : 0
        },
        "_seq_no" : 4,
        "_primary_term" : 1,
        "status" : 201
      }
    }
  ]
}
```


使用 Curl 命令导入数据至 Elasticsearch 实例

Curl 是一个用于与 Web 服务器进行交互的命令行工具，可以直接发送 HTTP 请求与 Elasticsearch 通信。

适用场景

快速脚本化操作：对于批量操作、简单的查询和数据导入，Curl 结合 Bash 脚本可以快速实现自动化任务。

轻量级客户端：不需要安装任何客户端库，只要有 HTTP 请求能力，Curl 就能与 Elasticsearch 交互。

临时操作：适用于快速执行临时任务或小规模的数据操作，比如单条数据的插入、查询、删除等。

前提条件

已经开通天翼云云搜索 Elasticsearch 实例。

能够通过公网或者内网访问到 Elasticsearch 实例。

操作步骤

1. 使用 Curl 命令导入数据。

a. 导入单条数据。

使用 Curl 可以通过 HTTP POST 请求将数据导入到 Elasticsearch 的指定索引中。

```
curl -X POST "http://ip:9200/articles/_doc" -H "Content-Type:
application/json" -d'
{
  "title": "通过 curl 导入数据",
  "content": "curl 是一款命令行工具，可以与 Elasticsearch 进行交互...",
  "date": "2024-08-23"
}
```

执行以上命令后，你会收到 Elasticsearch 返回的 JSON 响应，包含导入数据的_id 等信息。

b. 批量导入数据。

使用 Curl 也可以通过 Bulk API 实现批量数据导入。以下是一个简单的示例：

```
curl -X POST "http://ip:9200/_bulk" -H "Content-Type: application/json" -d'
{"index":{"_index":"articles"}}
{"title":"文章一","content":"是第一篇文章的内容...","date":2024-08-23"}
{"index":{"_index":"articles"}}
{title":"文二","content":"这是第二篇文章的内容...","date":"2024-08-23"}
```

在这个示例中，每个文档都以{"index":{"_index":"articles"}}作为前缀，后跟实际的数据内容。每条数据之间要用换行符分隔。

使用 Elasticsearch 实例搜索数据

搜索数据语法示例

Elasticsearch 是一个强大的搜索引擎，支持丰富的查询语法，能够满足各种复杂的搜索需求。本文将介绍一些基础且常用的搜索语法示例，帮助你快速上手 Elasticsearch 的数据搜索功能。

1. 基础搜索语法

a.简单搜索

最简单的搜索方式是直接在指定的索引中搜索包含特定关键字的文档。

```
GET /my_index/_search
{
  "query": {
    "match": {
      "content": "Elasticsearch"
    }
  }
}
```

以上查询会在 my_index 索引中搜索 content 字段中包含 "Elasticsearch" 的文档。

b.匹配所有文档

如果你想匹配索引中的所有文档，可以使用 match_all 查询。

```
GET /my_index/_search
{
  "query": {
    "match_all": {}
  }
}
```

这个查询会返回 my_index 中的所有文档。

c.精确匹配 (Term Query)

term 查询用于精确匹配指定字段的内容，适用于关键词搜索。

```
GET /my_index/_search
{
  "query": {
    "term": {
      "status": "active"
    }
  }
}
```

该查询会返回 status 字段值为 active 的文档。

2. 组合查询

a. 布尔查询 (Bool Query)

bool 查询允许将多个查询组合在一起。它支持 must、should、must_not 和 filter 子句，分别对应 AND、OR、NOT 和过滤条件。

```
GET /my_index/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "content": "Elasticsearch" } },
        { "term": { "status": "active" } }
      ],
      "filter": [
        { "range": { "date": { "gte": "2024-01-01" } } }
      ]
    }
  }
}
```

此查询会返回满足以下条件的文档：

content 字段包含 "Elasticsearch"。

status 字段为 active。

date 字段大于等于 "2024-01-01"。

b. 范围查询 (Range Query)

范围查询允许你搜索数值、日期或其他可比较字段的范围内的值。

```
GET /my_index/_search
{
  "query": {
```

```
    "range": {
      "price": {
        "gte": 100,
        "lte": 200
      }
    }
  }
}
```

这个查询会返回 price 字段值在 100 到 200 之间的文档。

3. 多字段查询

a. 多字段匹配 (Multi-Match Query)

multi_match 查询用于在多个字段中搜索相同的关键词。

```
GET /my_index/_search
{
  "query": {
    "multi_match": {
      "query": "Elasticsearch",
      "fields": ["title", "content"]
    }
  }
}
```

这个查询会在 title 和 content 字段中搜索 "Elasticsearch"。

b. 字段存在查询 (Exists Query)

exists 查询用于查找某个字段存在的文档。

```
GET /my_index/_search
{
  "query": {
    "exists": {
```

```
      "field": "user"
    }
  }
}
```

这个查询会返回所有 user 字段存在的文档。

4. 排序与分页

a. 排序 (Sort)

你可以根据一个或多个字段对搜索结果进行排序。

```
GET /my_index/_search
{
  "sort": [
    { "date": { "order": "desc" } },
    { "price": { "order": "asc" } }
  ],
  "query": {
    "match_all": {}
  }
}
```

此查询会先按 date 字段降序排序，再按 price 字段升序排序。

b. 分页 (Pagination)

分页可以通过 from 和 size 参数来实现。

```
GET /my_index/_search
{
  "from": 10,
  "size": 5,
  "query": {
    "match_all": {}
  }
```

```
}
```

这个查询会跳过前 10 条记录，并返回接下来的 5 条记录。

5. 高级搜索

a. 嵌套查询 (Nested Query)

如果文档中包含嵌套对象或数组，nested 查询可以用于对嵌套字段进行搜索。

```
GET /my_index/_search
{
  "query": {
    "nested": {
      "path": "comments",
      "query": {
        "bool": {
          "must": [
            { "match": { "comments.author": "John" } },
            { "range": { "comments.date": { "gte": "2024-01-01" } } }
          ]
        }
      }
    }
  }
}
```

此查询会返回 comments 数组中包含 author 为 John 且 date 在 "2024-01-01" 之后的文档。

b. 高亮显示 (Highlighting)

highlight 用于在搜索结果中突出显示匹配的关键词。

```
GET /my_index/_search
{
  "query": {
```

```
    "match": { "content": "Elasticsearch" }
  },
  "highlight": {
    "fields": {
      "content": {}
    }
  }
}
```

该查询会在搜索结果中高亮显示 content 字段中匹配到的 "Elasticsearch" 关键词。

以上是一些 Elasticsearch 的基础搜索语法示例，这些语法能够帮助进行有效的数据查询。根据实际业务需求，可以进一步组合这些查询来实现更复杂的搜索功能。

使用 Elasticsearch 实例向量检索功能增强搜索能力

概述

向量检索（Vector Search）是 Elasticsearch 的高级功能，允许用户在高维向量空间中进行相似性搜索，超越了传统的关键词匹配方式。通过将文本、图像等数据转换为向量表示，基于向量之间的距离进行搜索，适合自然语言处理、推荐系统和计算机视觉等复杂场景。

天翼云云搜索服务开通的 Elasticsearch 支持通过近似最近邻（ANN）搜索算法实现高效的向量索引结构，使得在处理大规模数据集时依然能保持高效的查询速度和准确性。

前提条件

已开通天翼云云搜索服务 Elasticsearch 集群。

Elasticsearch 版本支持 KNN 向量检索功能（当前版本默认支持）。

本地环境已配置好 API 访问权限，且能够通过 API 与集群通信。

操作步骤

1. 创建支持向量检索的索引

首先，需要创建一个支持向量检索的索引。可以使用以下命令为一个包含向量字段的索引启用 KNN 功能。

```
PUT my-knn-index-1
{
```

```
  "settings": {
    "index": {
      "knn": true,
      "knn.algo_param.ef_search": 100
    }
  },
  "mappings": {
    "properties": {
      "category": {
        "type": "keyword"
      },
      "brand": {
        "type": "keyword"
      },
      "style": {
        "type": "keyword"
      },
      "my_vector": {
        "type": "knn_vector",
        "dimension": 3
      }
    }
  }
}
```

knn: 设置为 true 启用向量检索。

dimension: 定义向量的维度，在这个例子中为 3。

2. 插入向量数据

创建索引后，可以插入带有向量字段的数据文档。以下是插入不同类型商品的向量示例：

```
PUT my-knn-index-1/_doc/1
{
  "category": "electronics",
  "brand": "brandA",
  "style": "modern",
  "my_vector": [0.5, 0.8, 0.3]
}
PUT my-knn-index-1/_doc/2
{
  "category": "furniture",
  "brand": "brandB",
  "style": "vintage",
  "my_vector": [0.2, 0.4, 0.7]
}
PUT my-knn-index-1/_doc/3
{
  "category": "clothing",
  "brand": "brandC",
  "style": "casual",
  "my_vector": [0.9, 0.1, 0.6]
}
```

3. 执行向量检索查询

插入数据后，用户可以通过查询指定的向量来查找与之相似的数据。以下示例将基于向量 [0.5, 0.8, 0.3] 进行 KNN 检索，返回与之最相似的 2 条记录。

```
POST my-knn-index-1/_search
{
  "size": 10,
```

```
  "query": {
    "knn": {
      "my_vector": {
        "vector": [0.5, 0.8, 0.3],
        "k": 2
      }
    }
  }
}
```

vector: 查询的向量值。

k: 返回与查询向量最相似的 k 个结果，此处为 2。

4. 查询返回示例

返回结果中将包含与查询向量最相似的文档及其相似度得分（_score）：

```
{
  "took" : 654,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 3,
      "relation" : "eq"
    },
```

```
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "my-knn-index-1",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "category" : "electronics",
          "brand" : "brandA",
          "style" : "modern",
          "my_vector" : [0.5, 0.8, 0.3]
        }
      },
      {
        "_index" : "my-knn-index-1",
        "_id" : "2",
        "_score" : 0.7092199,
        "_source" : {
          "category" : "furniture",
          "brand" : "brandB",
          "style" : "vintage",
          "my_vector" : [0.2, 0.4, 0.7]
        }
      }
    ]
  }
}
```

通过这些步骤，用户可以在 Elasticsearch 集群上实现基于向量的高效相似性搜索，支持从多维数据中快速找到最相似的结果，从而提升搜索体验和智能化水平。

插件管理

系统默认插件

系统默认插件为天翼云云搜索服务预置的插件，不支持卸载。

默认预设的插件列表功能版本如下

插件名称	插件描述	插件版本
analysis-hanlp	优化过的 HanLP 中文分词插件	7.10.2
analysis-ik	ik 中文分词插件，支持自定义词典	7.10.2
analysis-pinyin	拼音分词插件	7.10.2
analysis-stconvert	STConvert 插件，支持中文简体和中文繁体相互转换	7.10.2
opendistro-asynchronous-search	异步搜索插件	1.13.0.1
opendistro-cross-cluster-replication	跨实例复制插件	1.13.0.0
opendistro-index-management	索引管理插件	1.13.2.1
opendistro-job-scheduler	任务调度插件	1.13.0.0
opendistro-knn	向量检索引擎插件，可支撑图像搜索、语音识别和商品推荐等向量检索场景的需求	1.13.0.0
opendistro-sql	sql 查询插件	1.13.2.0

opendistro_security	安全插件	1.13.1.0
repository-hdfs	Hadoop 分布式文件系统 HDFS（Hadoop Distributed File System）存储库插件，提供了对 HDFS 存储库的支持	7.10.2
repository-s3	支持将数据存入天翼云对象存储 ZOS 的插件	7.10.2

自定义插件管理

当您需要使用自定义插件或系统默认插件中不包含的开源插件时，可通过云搜索服务的自定义插件上传与安装功能，在实例中上传并安装对应插件。本文介绍具体的操作方法。

前提条件

1. 准备待上传的插件，并确保插件的可用性和安全性。
2. 建议在上传插件前，先在本地自建 Elasticsearch 实例（与实例相同版本）上进行测试，成功后再进行上传。
3. 开通与云搜索实例同地域的对象存储服务，并上传好需要安装的插件至对象存储的桶中。

使用限制

1. 不支持安装与默认插件一样的插件，包括默认插件的其他版本；
2. 不支持同时安装/卸载两个以上的插件；
3. 云搜索服务不保留用户插件的安装文件，请您自行备份；
4. 插件文件后缀需要为.zip；
5. 不支持上传安装带权限属性的插件。

警告：安装自定义插件操作会触发实例重启插件本身可能影响实例的稳定性，请务必保证自定义插件的可用性和安全性，建议在业务低峰期进行操作。

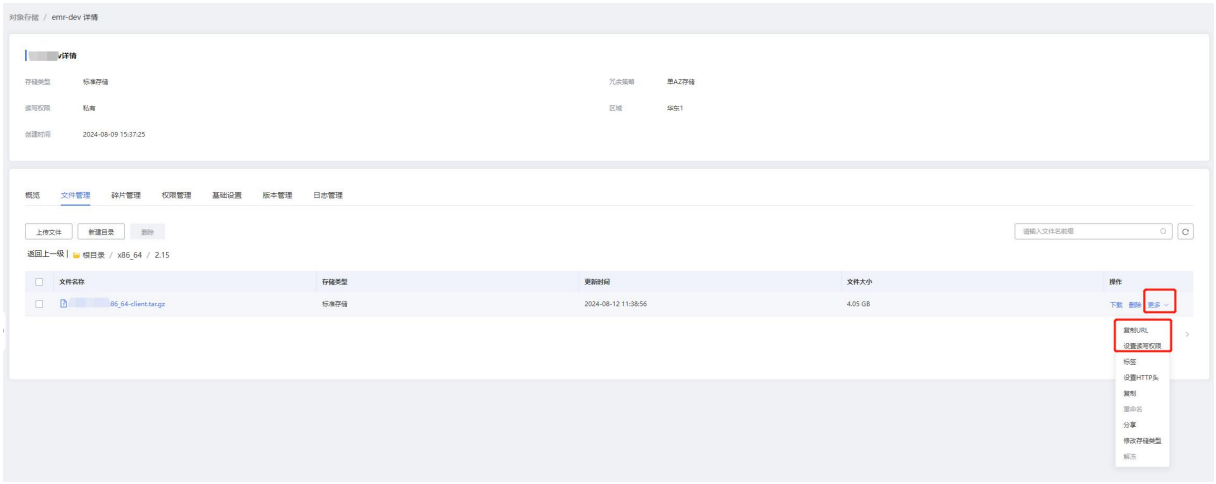
操作步骤

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。
2. 在实例详情页选择“插件管理”—“自定义插件”，单击上传插件按钮。

3. 在弹出的页面上填写：

- 插件名称，仅支持字母、数字和字符，请和插件包名称保持一致；
- 插件名称查询路径：插件包内 properties 文件中找到 pluginName。
- 查看插件包内名称方法：
- 插件地址：填写天翼云对象存储同地域资源池下的地址，获取路径为：
- 对象存储控制台——点击对应桶名称进入详情——点击文件管理，找到对应的插件，点击右侧更多——复制 URL。

注意：需要开启对应文件的可读权限。



- 插件版本：插件目前的版本，格式为数字和点组合，如：7.10 或 2.9.0 等，非必填。
- 插件描述：用于帮助用户识别插件功能，非必填。

填写完毕后点击安装并重启，插件会先在实例安装，安装完毕后实例将自动进行重启。期间请勿对实例进行其他操作。

4. 当实例重启完成，插件状态变成“已安装”则表示插件已经安装完毕。若插件处于“未安装”状态，则说明安装失败，您可根据提示调填写内容再次重试，或通过工单联系我们协助处理。您也可以删除该插件信息。
5. 已经安装完毕的插件，如果您不再使用，可单击插件右侧的卸载，卸载此插件。卸载插件如需再次安装，请参照前述步骤执行。

实例网络及安全设置

实例公网访问

新开启的云搜索实例默认不具备公网访问能力，您如果需要通过公网访问 Kibana 或 Elasticsearch 需要为其绑定弹性 IP 或 IPv6 带宽，并配置安全组信息，才可使用公网访问。

约束限制

1. 开启公网访问后，会因此产生流量费用，请您提前根据自身需求，购买合适的产品，公测期该费用照常收取。
2. 配置完成后，需要前往安全组设置页面，配置公网访问白名单后，才可正常使用。
3. 如果需要使用 IPv6 访问，需要在开通虚拟私有云 VPC 时即选择开通 IPv6 能力的子网，并在下单时选择该子网，不支持实例开通后再升级 IPv6。

开通 IPv6 访问能力的实例

您需要在订购时选择具备 IPv6 的虚拟私有云，选择子网后，会提示“该子网已开通 IPv6”。并在 IPv6 访问处开启开关，如关闭，则仅可通过 IPv4 访问实例。

配置实例公网访问

您可以对已开通的实例进行公网访问的配置、修改、查看、解绑操作。

1. 录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。

2. 在详情页面里选择“安全设置”，在弹出的页面上选择需要绑定的公网 IP 类型，如果为 IPv4，请在下拉列表中选择弹性 IP 地址；如果为 IPv6，请选择 IPv6 的带宽名称。如果绑定失败，可以等待几分钟后再次尝试重新绑定。绑定的弹性 IP 或 IPv6 带宽需要处于空闲状态。

IP 绑定过后，要补充安全组方可实现本地电脑公网访问 Kibana 或连接 Elasticsearch

3. 修改绑定弹性 IP 或 IPv6 带宽，也需要在当前页面选择对应要修改的项目，点击“修改公网 IP”进行重新绑定。

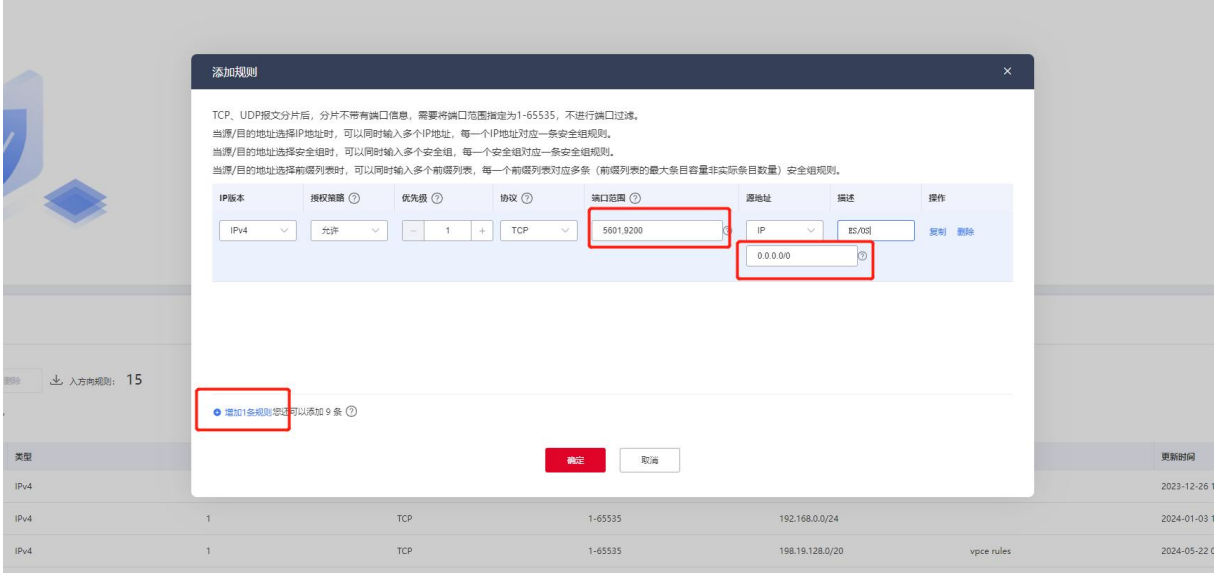
4. 解绑弹性 IP 或 IPv6 带宽，可关闭公网访问状态，或点击已绑定的项目后的解绑按钮，即可解绑当前的公网访问能力。

5. 实例退订将自动解绑已绑定的弹性 IP 或 IPv6 带宽，如弹性 IP 或 IPv6 带宽不再使用，您需要另外前往弹性 IP 的控制台退订相应的弹性 IP 或 IPv6 带宽才会停止对应产品的计费。

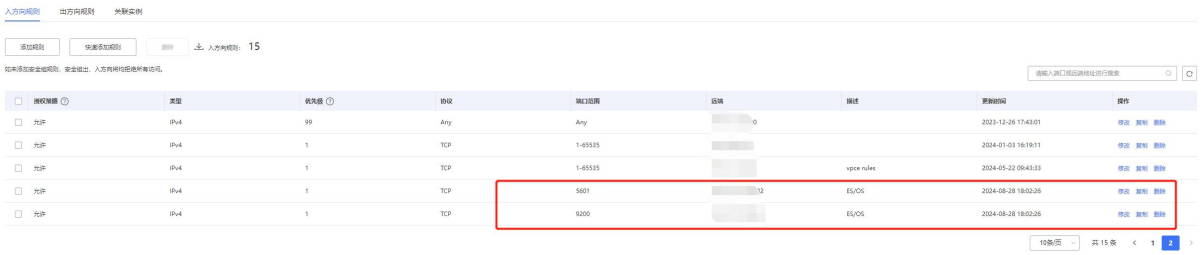
配置安全组白名单

操作步骤：

在控制台点击实例所在安全组，入方向规则点击添加规则，在弹出的填写框内的端口处填写“5601,9200”，选择需要配置的策略为 IPv4 或 IPv6，在源地址下方的 IP 地址格子中填写需要访问设备的出口公网 IP 地址，点击确定保存。



成功后会在安全组增加两条规则，此时可以通过绑定的公网 IP 地址端口访问对应对象。



通过公网 IP 地址接入实例

公网访问配置完成后，实例将会获得一个“公网访问”的 IP 地址，用户可以通过公网 IP 地址和端口接入实例。

例如，Kibana 可直接点击页面链接进行访问。

Elasticsearch 实例可以通过 Curl 命令查询索引信息

curl -u username:password -k 'https://10.62.179.32:9200/_cat/indices'其中 username 和 password 表示实例的用户名和密码。

实例密码修改

如果您在使用实例过程中需要重置实例管理员密码，可通过以下步骤进行重置。

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。
2. 在详情页面里选择“安全设置”，并点击下方的密码重置按钮，在显示的填写框处进行密码重置。请按照密码设置规范，填写新密码。

注意：

1. 密码为数字、大写字母、小写字母、特殊符号（@$!%*#_~?&）的组合。
2. 长度限制为 12-26 位。
3. 不能包含账号信息、字典序及键盘序。

管理实例

查看实例信息

在实例管理列表页、详情页，您可以查看实例版本、节点状态、使用情况等信息。

实例列表信息介绍

实例列表页为您展示全部购买的实例清单，如为子账号，则可见主账号的实例信息。

参数	描述
名称	订购时设置的实例名称，单击实例名称可以进入实例名称可以进入实例详情页。
状态	展示目前实例状态： • 运行中：正常在有效期内，运行状态正常的实例； • 创建中：刚下单还在订购开通部署中的实例； • 处理中：处于重启等正常操作下的实例； • 已冻结：包周期已到期的实例，可续费或退订销毁； • 已销毁：实例已删除，资源已回收的实例；如遇到资源不足

	或其他原因导致开通部署失败，将自动销毁，可再次下单或工单联系技术支持。 • 异常：实例不可连接或可连接但不可用。 注意：实例处于异常状态时，您可以尝试重启处理，若依旧失败，请及时联系技术支持。
索引数量	实例内的索引数量
存储用量	磁盘已使用的存储总量
版本	展示实例的版本号
计费模式	包年包月
到期时间	包年包月实例使用，表示包周期可使用的截止时间
创建时间	实例的创建时间
操作	展示实例可执行的操作入口，包含重启、续订等针对实例的操作，当前实例无法操作某项时，按钮将置灰。

实例基础信息页介绍

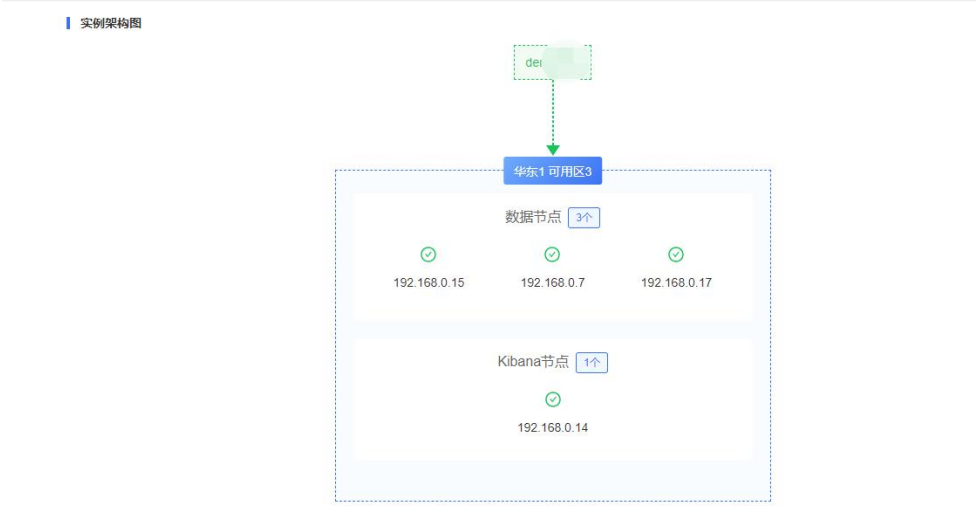
在实例的基础信息页面，可以获取实例的内网地址，版本节点信息等等。

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。
2. 在实例的详情页可查看实例的购买信息、健康情况、内网地址、和节点的主机健康情况。

实例状态说明：

- 健康：对应 Elasticsearch 实例 health 为 Green 时的状态，实例健康；
- 可用：对应 Elasticsearch 实例 health 为 Yellow 时的状态，主分片可用，部分副本缺失；
- 异常：对应 Elasticsearch 实例 health 为 Red 时的状态，部分主分片不可用，可能已经丢失数据。

3. 右侧实例架构图中展示的为当前实例所在区域、节点数量和节点主机状态。



重启实例

实例出现异常，或您有其他需要时，可以通过控制台重启实例。

前提条件

- 实例处于运行中或异常状态，未处于其他操作引起的重启中，未被冻结；
- 当实例处于运行中时，确认已停止数据写入、检索操作，否则重启实例可能会带此时写入的数据丢失、搜索不到数据等情况。

操作步骤

1. 登录云搜索服务管理控制台，在左侧导航栏，选择对应的实例类型，进入管理列表界面。
2. 在对应实例的“操作”列中单击“更多>重启”。
3. 重启实例后，请刷新页面，观察状态。重启过程中，实例状态为“处理中（重启）”，如果实例状态变更为“运行中”，表示实例已重启成功。

实例监控

实例提供支持查看云搜索服务实例的核心指标监控能力，方便用户掌握实例情况，及时处理实例异常。

使用限制

每次查询最多可选择 10 个节点/索引进行查看。

操作步骤

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要查看的实例点击操作栏的

监控按钮或点击实例名称，进入实例详情页。

2. 在实例详情页选择实例监控。

3. 实例监控共提供 3 种维度的指标查看

- 实例级：提供当前实例的整体情况，部分指标可切换查看各项指标的最大值和平均值；
- 节点级：选择单个或多个节点查看对应指标；
- 索引级：选择实例中已有的 open 状态的索引一个或多个进行查看。

云搜索服务支持的指标清单：

实例支持查看的指标

指标名称	指标含义	取值范围
健康状态	该指标用于统计测量监控对象的状态。	green（健康） yellow（可用） red（异常）
磁盘使用率	该指标用于统计测量对象的磁盘使用率。	0-100%
JVM 内存使用率	实例各个节点平均/最大 JVM 使用率	0-100%
CPU 使用率	实例各个节点平均/最大 CPU 使用率	0-100%
实例写入 QPS	当前实例的每秒写入速率	≥ 0
实例查询 QPS	当前实例的每秒查询速率	≥ 0
实例索引数量	当前实例的索引数量	≥ 0
实例分片数量	当前实例的分片数量	≥ 0
实例文档数量	当前实例的文档数量	≥ 0
写入延迟	完成单次索引写入的平均/最大时间	≥ 0 ms
查询延迟	完成单次搜索操作所需的平均/最大时间	≥ 0 ms
1 分钟负载	实例 1 分钟所有节点的平均/最大负载	≥ 0

节点维度

指标名称	指标含义	取值范围
磁盘使用率	该指标用于统计测量对象的磁盘使用率。	0-100%
CPU 使用率	指定节点 CPU 使用率	0-100%
内存使用率	指定节点 CPU 使用率	0-100%

索引维度

指标名称	指标含义	取值范围
索引文档数	索引中所包含的文档数量	≥ 0
索引总耗时	对应索引在监控时间范围内的所有操作总耗时	≥ 0ms
get 总请求数	对应索引在监测时间范围的 get 请求数	≥ 0
get 请求总耗时	对应索引在监测时间范围的 get 请求总耗时	≥ 0ms
search 总请求数	对应索引在监测时间范围的 search 请求数	≥ 0
search 请求总耗时	对应索引在监测时间范围的 search 请求总耗时	≥ 0ms

OpenSearch 实例创建及使用

创建 OpenSearch 实例

前提条件

1. 已在官网完成用户账号注册和实名认证。
2. 已完成实例规划。

操作步骤

您有两种路径可以进入 Elasticsearch 实例开通页面。

1. 登录官网，您可在云搜索服务产品详情页点击“立即开通”。
2. 登录官网，进入云搜索服务控制台后，点击“创建实例”可以进入。

订购页面填写

1. 进入订购页面后，您需要对如下信息进行填写/选择：

参数类型	参数	说明
基础订购信息	计费模式	实例目前仅支持包年/包月模式。 包年/包月：根据实例购买时长，一次性支付实例费用。公测期间最短时长为 1 个月，最长时长为 6 个月。如公测到期时间早于实例的订购到期时间，则有效时间截止至公测到期时间。
	订购周期	在包年包月模式下，您需要选择购买时长。
	当前区域	选择实例的所在区域。 不同区域的云服务产品之间内网互不相通。请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。
	可用区	选择实例所在工作区域下关联的可用区。
网络配置	虚拟私有云	指定实例所在的虚拟专用网络（VPC），实现不同业务的网络隔离。如您没有合适的 VPC，请前往创建虚拟私有云页面创建完成后，回当前页面刷新后选择。
	子网	实例使用子网实现与其他网络的隔离，并独享所有网络资源，以提高网络安全。 选择当前虚拟私有云下实例需要的子网。
	安全组	安全组为实例提供安全的网络访问控制策略。 为了顺利部署实例，我们将为您选择的安全组默认配置下述规则： 规则 1（入方向）：允许远端 198.19.128.0/20 以 TCP 协议访问端口 1-65535 ，规则优先级为 1。 规则 2（入方向）：允许远端 192.168.0.0/24（实际网段地址，以客户创建的 VPC 子网网段地址为准）以 TCP 协议访问端口 1-65535，规则优先级为 1。 规则 3（出方向）：将允许出方向所有访问，规则优先级为 100。

配置实例	实例名称	您可自定义实例名称，可输入的字符范围为长度为 0~32 个字符，只能包含数字、字母、中划线（-）和下划线（_），且必须以字母开头。
	实例类型	云搜索服务支持 Elasticsearch 和 OpenSearch 两种组件。
	实例版本	选择所需的集群版本，支持的版本以页面可选项为准。
选购资源	实例节点数	实例中需要部署节点数，公测期为小规模集群，请以页面可下单数量为准。
	CPU 架构	支持 X86 类型。
	节点规格	实例的节点规格，可按需选购，同一实例仅支持一种规格，请确认后下单。
	节点存储规格	选择存储类型，支持的类型可能随资源池不同有差异，请以页面可选的为准，性能参考。
	单节点存储量	您可根据业务数据容量评估，选购合适的单节点存储量，创建完成后，不支持缩容节点存储容量，请基于业务量合理选择容量。
	Dashboards 节点规格	云搜索服务会默认为您开通一个小规格节点部署 Dashboards 节点。该节点正常计费，公测期免费。
实例密码设置	实例密码设置	设置 OpenSearch 的管理员访问密码； 实例内账号和角色请登录 Dashboards 的进行设置。参见文档。

2. 信息填写完毕后，进入下一步信息确认页；请您仔细核对订购信息及订购协议，勾选协议后进入下一步即可提交。
3. 缴费完成后，请返回购买实例对应的实例管理列表页面，您购买的实例都将展示在此，当实例从“创建中”变为“运行中”时，即可使用实例。
4. 如实例创建失败，系统将自动退单销毁，期间不会记录使用时长，不会收取费用，您可再次下单尝试，或工单联系工程师为您处理。

访问 OpenSearch 实例

概览

天翼云云搜索服务 OpenSearch 实例支持多种连接方式：

OpenSearch Dashboards 访问

OpenSearch 实例支持通过 OpenSearch Dashboards 可视化界面进行连接。用户可以通过登录页面输入用户名和密码访问实例，默认用户名为 admin，密码为创建实例时设置的管理员密码。Dashboards 提供了强大的数据可视化和管理功能。

Curl 命令行方式

用户可以使用 Curl 命令行与 OpenSearch 实例进行通信。curl 命令允许执行多种操作，如创建索引、查询数据、更新和删除文档，适合用于快速测试和管理实例。

Python client 访问

OpenSearch 提供官方 Python 客户端（opensearch-py），允许开发者通过 API 与 OpenSearch 实例交互。该客户端适用于数据索引、复杂查询和自动化脚本，适合大规模数据处理和分析场景。

Java client 访问

Java 客户端提供对 OpenSearch 实例的深度集成，使用 RestHighLevelClient 等 API 与 OpenSearch 通信。Java 客户端非常适合在企业级应用中进行复杂搜索、实时数据处理等操作，能够与 Spring 等框架很好地结合。

Go client 访问

Go 客户端支持与 OpenSearch REST API 高效交互，特别适合需要高并发、低延迟的应用。通过 Go 客户端，可以实现对 OpenSearch 实例的快速连接和操作，常用于高性能、可扩展的云搜索服务场景。

通过 Curl 命令行接入 OpenSearch 实例

概述

使用 Curl 是最简单直接的方式访问 OpenSearch 集群。它允许用户通过命令行发送 HTTP 请求与集群进行交互，例如创建索引、查询集群状态等操作。

前提条件

已开通天翼云云搜索服务 OpenSearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

本地已按照 CURL 工具。

操作步骤

使用以下 Curl 命令访问 OpenSearch 集群：

curl -u <user>:<password> "http://<host>:<port>"

<user>：OpenSearch 集群用户名，比如 admin。

<password>：该用户密码，比如用户配置的 OpenSearch 集群 admin 用户密码。

<host>：主机 IP，即集群绑定的公网 IP。

<port>：端口号，一般是 9200。

通过 Java 客户端接入 OpenSearch 实例

概述

使用 OpenSearch 提供的 Java 客户端，用户可以通过 Java 应用与集群交互，进行索引管理、数据查询、插入文档等操作。适合大规模 Java 应用开发。

前提条件

已开通天翼云云搜索服务 OpenSearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已在本地安装 JDK（推荐 JDK 8 及以上版本）。

已配置 Maven 或 Gradle 项目以支持 OpenSearch Java 客户端。

操作步骤

在项目中引入 OpenSearch 客户端依赖。Maven 依赖配置如下：

```
<dependency>
  <groupId>org.opensearch.client</groupId>
  <artifactId>opensearch-rest-high-level-client</artifactId>
  <version>2.9.0</version>
</dependency>
```

使用以下代码连接到 OpenSearch 集群：

```
import org.apache.http.HttpHost;
import org.opensearch.client.RestClient;
import org.opensearch.client.RestHighLevelClient;
```

```
public class OpenSearchJavaClient {
    public static void main(String[] args) {
        // 初始化客户端
        RestHighLevelClient client = new RestHighLevelClient(
            RestClient.builder(new HttpHost("<host>", 9200, "http"))
                    .setDefaultCredentialsProvider(new
BasicCredentialsProvider().setCredentials(
                            AuthScope.ANY, new
UsernamePasswordCredentials("<user>", "<password>")
                    )));
        // 执行操作，例如创建索引等
        // ...
        // 关闭客户端
        client.close();
    }
}
```

<host>：集群绑定的公网 IP。

<user>：OpenSearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

执行创建索引的操作：

```
CreateIndexRequest request = new CreateIndexRequest("my_index");
CreateIndexResponse createIndexResponse = client.indices().create(request,
RequestOptions.DEFAULT);
```

操作完成后记得关闭客户端：

```
client.close();
```

通过 Python 客户端接入 OpenSearch 实例

概述

Python 客户端（opensearch-py）是 OpenSearch 官方提供的库，可以用来与集群交互，

支持数据查询、索引管理等操作，适合快速开发和轻量级应用。

前提条件

已开通天翼云云搜索服务 OpenSearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已在本地安装 Python 3.x 版本。

已安装 OpenSearch 官方 Python 客户端库。

操作步骤

安装 Python 客户端库：

```
pip install opensearch-py
```

使用以下代码连接到 OpenSearch 集群：

```
from opensearchpy import OpenSearch

# 连接到 OpenSearch 集群

client = OpenSearch(

    hosts=["http://<host>:9200"],

    http_auth=("<user>", "<password>")

)

# 创建索引操作

client.indices.create(index="my_index", ignore=400)
```

<host>：集群绑定的公网 IP。

<user>：OpenSearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

执行查询操作：

```
response = client.get(index="my_index", id=1)

print(response)
```

通过 Go 客户端接入 OpenSearch 实例

概述

Go 客户端（opensearch-go）是 OpenSearch 官方提供的 Golang 库，适用于构建高性能的应用程序。它提供了与 OpenSearch 集群交互的 API，支持索引创建、数据查询等操作。

前提条件

已开通天翼云云搜索服务 OpenSearch 集群。

集群已绑定公网 IP。具体可参考“实例公网访问”章节。

已安装 Go 语言开发环境。

已安装 OpenSearch 官方 Go 客户端库。

操作步骤

安装 Go 客户端库：

```
go get github.com/opensearch-project/opensearch-go
```

使用以下代码连接到 OpenSearch 集群：

```
package main

import (
    "context"
    "fmt"
    "log"
    "github.com/opensearch-project/opensearch-go"
)

func main() {
    // 创建 OpenSearch 客户端
    client, err := opensearch.NewClient(opensearch.Config{
        Addresses: []string{"http://<host>:9200"},
        Username:  "<user>",
        Password:  "<password>",
    })
    if err != nil {
        log.Fatalf("Error creating the client: %s", err)
    }
    // 创建索引
    res, err := client.Indices.Create("my_index")
```

```
    if err != nil {
        log.Fatalf("Error creating index: %s", err)
    }
    fmt.Println(res)
}
```

<host>：集群绑定的公网 IP。

<user>：OpenSearch 集群用户名，例如 admin。

<password>：用户密码，例如 admin 用户的密码。

导入数据至 OpenSearch 实例

OpenSearch 实例数据导入方式

在 OpenSearch 中，导入数据是一项核心操作，可以通过多种方式来实现。

天翼云云搜索 OpenSearch 实例中，有多种方式可以导入数据，常见的几种导入方式包括：OpenSearch 客户端、Beats、Logstash、OpenSearch-Dashboards 或其他数据导入工具。可以根据实际的需求来选择合适的方式来导入数据至 OpenSearch 实例。

支持的导入方式如下表：

导入方式	适用场景
OpenSearch 客户端	适合开发者，处理复杂业务逻辑和批量数据操作。
Beats	适合实时日志、指标的轻量级数据采集。
Logstash	适合复杂的数据处理管道和多源数据整合。
OpenSearch-Dashboards	适合快速测试、调试和简单数据管理。
Curl 命令行	适合轻量、临时和自动化脚本操作。

这里主要介绍使用 OpenSearch 客户端、Beats、Logstash、OpenSearch-Dashboards、Curl 命令行的方式导入数据至 OpenSearch 实例。

您也可以通过自行开发或者适用第三方数据导入工具将各种数据导入到OpenSearch 实例。

使用 OpenSearch 客户端导入数据至 OpenSearch 实例

OpenSearch 提供官方的客户端库，支持多种编程语言，如 Java、Python、JavaScript

等。

1. 适用场景。

编程场景：当你有自定义应用程序，需要通过代码直接与 OpenSearch 交互时，OpenSearch 客户端提供了灵活的 API 进行复杂查询和批量导入数据。

批量数据导入：通过客户端库可以实现大规模数据的分块导入，并发写入，适用于处理大数据量的场景。

动态数据处理：如果数据在导入前需要复杂的逻辑处理，可以通过编程语言和客户端实现定制的数据流。

2. 前提条件。

已经开通天翼云云搜索 OpenSearch 实例。

能够通过 HTTP 访问 OpenSearch 实例。

3. 客户端使用实例。

这里以 Python 和 Java 客户端为例。

a. 使用 Python 客户端 (opensearch-py)。

Python 客户端 opensearch-py 是一个与 OpenSearch 交互的轻量级库。使用它，你可以通过 index 方法将数据导入到指定索引中。

```
from opensearchpy import OpenSearch
# 创建 OpenSearch 客户端
os_client = OpenSearch("http://ip:9200")
# 要导入的数据
data = {
    "title": "OpenSearch 入门",
    "content": "OpenSearch 是一款分布式搜索引擎，用于高效搜索和分析大量数据。",
    "date": "2024-08-23"
}
# 将数据导入到名为"articles"的索引
response = os_client.index(index="articles", body=data)
print(response)
```

b. 使用 Java 客户端。

Java 是 OpenSearch 的主要编程语言之一，其官方客户端提供了丰富的功能。以下示例展示了如何使用 Java 客户端导入数据：

```
import  org.opensearch.client.RestClient;
import  org.opensearch.client.RestHighLevelClient;
import  org.opensearch.client.RequestOptions;
import  org.opensearch.action.index.IndexRequest;
import  org.opensearch.action.index.IndexResponse;
import  org.opensearch.common.xcontent.XContentType;
public  class  DataImporter  {

        public  static  void  main(String[]  args)  throws  Exception  {
                RestHighLevelClient  client  =  new  RestHighLevelClient(
                        RestClient.builder(new  HttpHost("ip",  9200,  "htt
p"))
                );
                String  jsonString  =  "{"  +
                        "\"title\":\"OpenSearch 入门\","  +
                        "\"content\":\"OpenSearch 是一款分布式搜索引
擎...\","  +
                        "\"date\":\"2024-08-23\""  +
                        "}";
                IndexRequest  request  =  new  IndexRequest("articles");
                request.source(jsonString,  XContentType.JSON);

                IndexResponse  response  =  client.index(request,  RequestO
ptions.DEFAULT);
```

```
                System.out.println(response.getId());


                client.close();
        }
}
```

使用自建 Beats 导入数据至 OpenSearch 实例

Beats 是轻量级的数据收集器，专门用于将各种日志、指标、网络数据发送到 Elasticsearch。常用的 Beats 包括 Filebeat、Metricbeat、Packetbeat 等。

Filebeat 是一种轻量级日志收集器,通常用于将文件系统中的日志文件或事件日志发送到 OpenSearch 或 Logstash。它适合简单的日志采集场景，能够有效处理系统、应用程序日志等。本文将以 Filebeat 为例，导入数据至 OpenSearch 实例。

1. 适用场景。

轻量数据收集：适用于需要从大量分布式系统、服务器、容器中收集日志、监控指标等情况。实时日志监控：Beats 能够实时收集日志并传送到 OpenSearch，是实时日志分析场景的理想选择。

低延迟要求的场景：Beats 设计为轻量级工具，占用资源少，适合资源受限的环境。

2. 前提条件。

已经开通天翼云云搜索 OpenSearch 实例。

已经部署 Filebeat 且打通和 OpenSearch 实例之间的网络。

在 OpenSearch 的配置文件中配置：

compatibility.override_main_response_version: true

并重启 OpenSearch 实例。

3. Filebeat 配置。

Filebeat 采集日志，需要配置 Filebeat 的配置文件，设置具体需要采集的日志路径。

具体根据实际的部署路径配置 filebeat.yml

```
#采集日志
filebeat.inputs:
- type: log
  # 采集的日志文件的路径。替换为自己日志的路径，可以使用通配符。
```

```
  paths:
    - /your_path/*.log

#OpenSearch 可以使用 Elasticsearch 的输出插件。
output.elasticsearch:
  # ip 替换为 OpenSearch 实例的地址。
  hosts: ["http://{ip}:9200"]
  # 传入 OpenSearch 实例的用户名和密码
  username: "********"
  password: "********"
```

4. 启动 Filebeat 导入数据。

可以使用下面的命令在命令行来启动 Filebeat 导入数据。

```
./filebeat -e -c filebeat.yml
```

使用自建 Logstash 导入数据至 OpenSearch 实例

Logstash 是一个开源的服务器端实时数据处理工具，支持从多个数据源中提取数据，经过处理后将数据导入到 OpenSearch 中。它非常适合处理流数据，如日志、监控数据和指标数据。

适用场景

日志数据、监控数据、流数据等。

前提条件

已经开通天翼云云搜索 OpenSearch 实例。

已经部署 Logstash 且打通和 OpenSearch 实例之间的网络。

操作步骤

1. Logstash 配置

Logstash 可以接收很多数据源，可以根据实际的需求配置。

这里使用 Filebeat 作为 Logstash 的输入。

配置 your_logstash.conf 管道文件。

Logstash 需要接收 Filebeat 的输出并进行处理，示例配置如下：

```
input {
```

```
  beats {
    port => 5044
  }
}
# 对数据进行处理。
filter {
    # mutate {
    #   remove_field => ["@version"]
    # }
}
# 输出到 OpenSearch 实例。
output{
OpenSearch{
# OpenSearch 实例的访问地址。
hosts => ["http://{ip}:{port}", "http://{ip}:{port}", "http://{ip}:{port}"]
# 访问 OpenSearch 实例的用户名和密码，如无安全机制可不配置。
user => "********"
password => "********"
# 配置写入的索引名,示例如下。
index => "filebeat-logstash-os-%{+YYYY.MM.dd}"
}
}
```

2. 启动 Logstash 导入数据

可以使用下面的命令在命令行来启动 logstash 导入数据。

```
./bin/logstash -f your_logstash.conf
```

使用 OpenSearch Dashboards 导入数据至 OpenSearch 实例

OpenSearch Dashboards 可视化界面提供的 Dev Tools 控制台允许直接在浏览器中通过 RESTAPI 向 OpenSearch 实例发出查询和数据操作请求。

OpenSearch Dashboards 可视化界面提供了简单的数据导入功能，适用于小规模数据的

手动导入场景，特别是在测试和快速验证场景中非常实用。

1. 适用场景。

快速测试与调试：适用于开发阶段测试查询语句、创建索引、插入和更新数据等操作。

简单数据管理：如果只是少量数据操作，如插入、更新、删除数据或查看结果，OpenSearch Dashboards 提供了方便的 Web 界面操作。

无需编程环境：不需要安装额外的客户端工具，只要能访问 OpenSearch Dashboards 即可操作 OpenSearch。

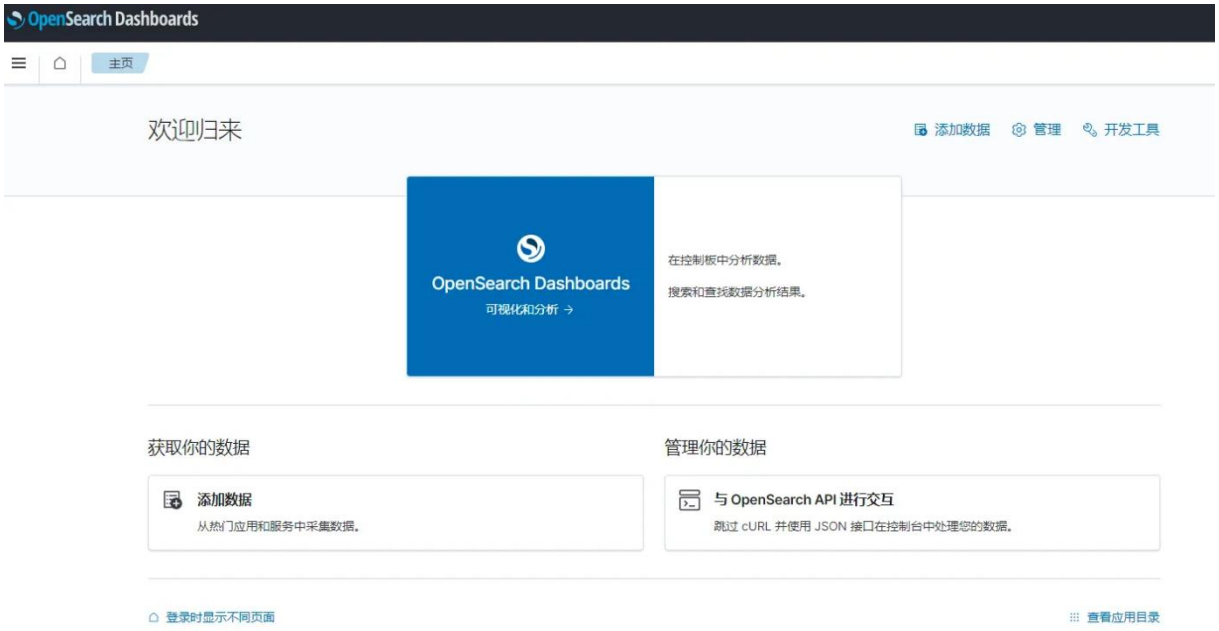
2. 前提条件。

已经开通天翼云云搜索 OpenSearch 实例。

查看 OpenSearch Dashboards 的终端可以访问到云搜索实例，设置好 5601 端口的网络安全策略。

3. 实际操作。

点击 OpenSearch Dashboards 输入用户名登录后，进入首页。



右上角可以点击开发工具进入开发工具界面。

如果没有索引，可以创建一个测试索引名为 test_index。

```
PUT /test_index
{
  "mappings": {
```

```
    "properties": {
      "mytest": {
        "type": "text"
      }
    }
  }
}
```

执行创建索引操作会返回如下信息。

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "test_index"
}
```

如果有索引可以使用已有的索引。

往索引中写入数据，以上面创建的索引为例。

a. 写入单条数据

可以使用下面命令写入一条 id 为 1 的数据。

```
POST /test_index/_doc/1
{
  "mytest": "xiaoming"
}
```

执行上面的命令，返回下面的结果即导入数据成功。

```
{
  "_index" : "test_index",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
```

```
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

b. 批量写数据

可以使用下面命令写入一批数据。

```
POST /test_index/_bulk
{"index":{"_id": 1}}
{"mytest": "xiaoming"}
{"index":{"_id": 2}}
{"mytest": "xiaohong"}
{"index":{"_id": 3}}
{"mytest": "xiaoli"}
{"index":{"_id": 4}}
{"mytest": "xiaozhang"}
{"index":{"_id": 5}}
{"mytest": "xiaowang"}
```

执行上面的命令，返回下面的结果即导入数据成功。

```
{
  "took" : 10,
  "errors" : false,
  "items" : [
    {
```

```
  "index" : {
    "_index" : "test_index",
    "_type" : "_doc",
    "_id" : "1",
    "_version" : 1,
    "result" : "created",
    "_shards" : {
      "total" : 2,
      "successful" : 2,
      "failed" : 0
    },
    "_seq_no" : 0,
    "_primary_term" : 1,
    "status" : 201
  }
},
....
{
  "index" : {
    "_index" : "test_index",
    "_type" : "_doc",
    "_id" : "5",
    "_version" : 1,
    "result" : "created",
    "_shards" : {
      "total" : 2,
      "successful" : 2,
```

```
        "failed" : 0
      },
      "_seq_no" : 4,
      "_primary_term" : 1,
      "status" : 201
    }
  }
 ]
}
```

使用 Curl 命令导入数据至 OpenSearch 实例

Curl 是一个用于与 Web 服务器进行交互的命令行工具，可以直接发送 HTTP 请求与 OpenSearch 通信。

适用场景

快速脚本化操作：对于批量操作、简单的查询和数据导入，Curl 结合 Bash 脚本可以快速实现自动化任务。

轻量级客户端：不需要安装任何客户端库，只要有 HTTP 请求能力，Curl 就能与 OpenSearch 交互。

临时操作：适用于快速执行临时任务或小规模的数据操作，比如单条数据的插入、查询、删除等。

前提条件

已经开通天翼云云搜索 OpenSearch 实例。

能够通过公网或者内网访问到 OpenSearch 实例。

操作步骤

1. 使用 Curl 命令导入数据。

a. 导入单条数据。

使用 Curl 可以通过 HTTP POST 请求将数据导入到 OpenSearch 的指定索引中。

```
curl -X POST "http://ip:9200/articles/_doc" -H "Content-Type:
application/json" -d'
{
```

```
  "title": "通过 curl 导入数据",
  "content": "curl 是一款命令行工具，可以与 OpenSearch 进行交互...",
  "date": "2024-08-23"
}
```

执行以上命令后，你会收到 OpenSearch 返回的 JSON 响应，包含导入数据的_id 等信息。

b. 批量导入数据。

使用 Curl 也可以通过 Bulk API 实现批量数据导入。以下是一个简单的示例：

```
curl -X POST "http://ip:9200/_bulk" -H "Content-Type: application/json" -d'
{"index":{"_index":"articles"}}
{"title":"文章一","content":"是第一篇文章的内容...","date":2024-08-23"}
{"index":{"_index":"articles"}}
{title":"文二","content":"这是第二篇文章的内容...","date":"2024-08-23"}
```

在这个示例中，每个文档都以{"index":{"_index":"articles"}}作为前缀，后跟实际的数据内容。每条数据之间要用换行符分隔。

使用 OpenSearch 实例搜索数据

搜索数据语法示例

OpenSearch 是一个强大的搜索引擎，支持丰富的查询语法，能够满足各种复杂的搜索需求。本文将介绍一些基础且常用的搜索语法示例，帮助你快速上手 OpenSearch 的数据搜索功能。

1. 基础搜索语法

a. 简单搜索

最简单的搜索方式是直接在指定的索引中搜索包含特定关键字的文档。

```
GET /my_index/_search
{
  "query": {
    "match": {
      "content": "OpenSearch"
```

```
    }
  }
}
```

以上查询会在 my_index 索引中搜索 content 字段中包含 "OpenSearch" 的文档。

b. 匹配所有文档

如果你想匹配索引中的所有文档，可以使用 match_all 查询。

```
GET /my_index/_search
{
  "query": {
    "match_all": {}
  }
}
```

这个查询会返回 my_index 中的所有文档。

c. 精确匹配 (Term Query)

term 查询用于精确匹配指定字段的内容，适用于关键词搜索。

```
GET /my_index/_search
{
  "query": {
    "term": {
      "status": "active"
    }
  }
}
```

该查询会返回 status 字段值为 active 的文档。

2. 组合查询

a. 布尔查询 (Bool Query)

bool 查询允许将多个查询组合在一起。它支持 must、should、must_not 和 filter 子句，分别对应 AND、OR、NOT 和过滤条件。

```
GET /my_index/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "content": "OpenSearch" } },
        { "term": { "status": "active" } }
      ],
      "filter": [
        { "range": { "date": { "gte": "2024-01-01" } } }
      ]
    }
  }
}
```

此查询会返回满足以下条件的文档：

content 字段包含 "OpenSearch"。

status 字段为 active。

date 字段大于等于 "2024-01-01"。

b. 范围查询 (Range Query)

范围查询允许你搜索数值、日期或其他可比较字段的范围内的值。

```
GET /my_index/_search
{
  "query": {
    "range": {
      "price": {
        "gte": 100,
        "lte": 200
      }
```

```
    }
  }
}
```

这个查询会返回 price 字段值在 100 到 200 之间的文档。

3. 多字段查询

a. 多字段匹配 (Multi-Match Query)

multi_match 查询用于在多个字段中搜索相同的关键词。

```
GET /my_index/_search
{
  "query": {
    "multi_match": {
      "query": "OpenSearch",
      "fields": ["title", "content"]
    }
  }
}
```

这个查询会在 title 和 content 字段中搜索 "OpenSearch"。

b. 字段存在查询 (Exists Query)

exists 查询用于查找某个字段存在的文档。

```
GET /my_index/_search
{
  "query": {
    "exists": {
      "field": "user"
    }
  }
}
```

这个查询会返回所有 user 字段存在的文档。

4. 排序与分页

a. 排序 (Sort)

你可以根据一个或多个字段对搜索结果进行排序。

```
GET /my_index/_search
{
  "sort": [
    { "date": { "order": "desc" } },
    { "price": { "order": "asc" } }
  ],
  "query": {
    "match_all": {}
  }
}
```

此查询会先按 date 字段降序排序，再按 price 字段升序排序。

b. 分页 (Pagination)

分页可以通过 from 和 size 参数来实现。

```
GET /my_index/_search
{
  "from": 10,
  "size": 5,
  "query": {
    "match_all": {}
  }
}
```

这个查询会跳过前 10 条记录，并返回接下来的 5 条记录。

5. 高级搜索

a. 嵌套查询 (Nested Query)

如果文档中包含嵌套对象或数组，nested 查询可以用于对嵌套字段进行搜索。

```
GET /my_index/_search
{
  "query": {
    "nested": {
      "path": "comments",
      "query": {
        "bool": {
          "must": [
            { "match": { "comments.author": "John" } },
            { "range": { "comments.date": { "gte": "2024-01-01" } } }
          ]
        }
      }
    }
  }
}
```

此查询会返回 comments 数组中包含 author 为 John 且 date 在 "2024-01-01" 之后的文档。

b. 高亮显示 (Highlighting)

highlight 用于在搜索结果中突出显示匹配的关键词。

```
GET /my_index/_search
{
  "query": {
    "match": { "content": "OpenSearch" }
  },
  "highlight": {
    "fields": {
      "content": {}
```

```
    }
  }
}
```

该查询会在搜索结果中高亮显示 content 字段中匹配到的 "OpenSearch" 关键词。

以上是一些 OpenSearch 的基础搜索语法示例，这些语法能够帮助你进行有效的数据查询。根据你的业务需求，你可以进一步组合这些查询来实现更复杂的搜索功能。

使用 OpenSearch 实例向量检索功能增强搜索能力

概述

向量检索（Vector Search）是 OpenSearch 的高级功能，它允许用户在高维向量空间中进行相似性搜索。这一功能不仅基于传统的关键词匹配，还支持通过向量表示的方式来处理更复杂的查询场景，例如自然语言处理、推荐系统和计算机视觉等。

天翼云云搜索服务开通的 OpenSearch 集群通过集成近似最近邻（ANN）搜索算法，确保在大规模数据集上实现高效、精准的向量检索，使用户可以快速找到与查询向量最相似的结果。

前提条件

已开通天翼云云搜索服务 OpenSearch 集群。

OpenSearch 版本支持 KNN 向量检索功能（当前版本默认支持）。

本地环境已配置好 API 访问权限，且能够通过 API 与集群通信。

操作步骤

1. 创建支持向量检索的索引

在 OpenSearch 中，可以通过以下命令创建一个启用了 KNN 功能的索引，用于向量检索：

```
PUT my-knn-index-1
{
  "settings": {
    "index": {
      "knn": true,
      "knn.algo_param.ef_search": 100
    }
```

```
  },
  "mappings": {
    "properties": {
      "category": {
        "type": "keyword"
      },
      "brand": {
        "type": "keyword"
      },
      "style": {
        "type": "keyword"
      },
      "my_vector": {
        "type": "knn_vector",
        "dimension": 3
      }
    }
  }
}
```

knn: 设置为 true 以启用向量检索功能。

dimension: 指定向量的维度，这里设置为 3。

2. 插入向量数据

创建好索引后，可以通过以下命令插入具有向量字段的数据：

```
PUT my-knn-index-1/_doc/1
{
  "category": "electronics",
  "brand": "brandA",
```

```
  "style": "modern",
  "my_vector": [0.5, 0.8, 0.3]
}

PUT my-knn-index-1/_doc/2
{
  "category": "furniture",
  "brand": "brandB",
  "style": "vintage",
  "my_vector": [0.2, 0.4, 0.7]
}

PUT my-knn-index-1/_doc/3
{
  "category": "clothing",
  "brand": "brandC",
  "style": "casual",
  "my_vector": [0.9, 0.1, 0.6]
}
```

3. 执行向量检索查询

数据插入完成后，可以通过向量进行检索。以下是一个查询示例，它将基于向量 [0.5, 0.8, 0.3] 进行 KNN 搜索，并返回最相似的 2 条记录：

```
POST my-knn-index-1/_search
{
  "size": 10,
  "query": {
    "knn": {
```

```
      "my_vector": {
        "vector": [0.5, 0.8, 0.3],
        "k": 2
      }
    }
  }
}
```

vector: 要进行相似性检索的向量值。

k: 返回与查询向量最相似的 k 个结果，此例中为 2。

4. 查询返回结果示例

以下为检索后的返回结果，其中包含与查询向量最相似的数据文档及其相似度得分（_score）：

```
{
  "took" : 200,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 3,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
```

```
      {
        "_index" : "my-knn-index-1",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "category" : "electronics",
          "brand" : "brandA",
          "style" : "modern",
          "my_vector" : [0.5, 0.8, 0.3]
        }
      },
      {
        "_index" : "my-knn-index-1",
        "_id" : "2",
        "_score" : 0.7092199,
        "_source" : {
          "category": "furniture",
          "brand": "brandB",
          "style": "vintage",
          "my_vector": [0.2, 0.4, 0.7]
        }
      }
    ]
  }
}
```

通过这些步骤，用户可以在 OpenSearch 集群中轻松实现基于向量的相似性搜索功能，支持高效处理海量数据并提升搜索体验。

插件管理

默认插件

系统默认插件为天翼云云搜索服务预置的插件，不支持卸载。

默认预设的插件列表功能版本如下

插件名	描述	版本
analysis-hanlp	优化过的 HanLP 中文分词插件	2.9.0
flowcontrol	自研流量控制插件，可进行流控管控、限流等	2.9.0.0
opensearch-analisys-pinyin	拼音分词插件	2.9.0
opensearch-analisys-stconvert	STConvert 插件，支持中文简体和中文繁体相互转换	2.9.0
opensearch-analysis-ik	ik 中文分词插件，支持自定义词典	2.9.0
opensearch-asynchronous-search	异步搜索插件	2.9.0.0
opensearch-cross-cluster-replication	跨集群复制插件	2.9.0.0
opensearch-geospatial	geospatial 插件	2.9.0.0
opensearch-index-management	索引管理插件	2.9.0.0
opensearch-job-scheduler	任务调度插件	2.9.0.0

opensearch-knn	向量检索引擎插件，可支撑图像搜索、语音识别和商品推荐等向量检索场景的需求	2.9.0.0
opensearch-notifications	消息通知插件	2.9.0.0
opensearch-notifications-core	消息通知 core 插件	2.9.0.0
opensearch-security	安全插件	2.9.0.0
opensearch-sql	sql 查询插件	2.9.0.0
prometheus-exporter	Opensearch 的 prometheus exporter 插件	2.9.0.0
repository-hdfs	Hadoop 分布式文件系统 HDFS（Hadoop Distributed File System）存储库插件，提供了对 HDFS 存储库的支持	2.9.0
repository-s3	支持将数据存入天翼云对象存储 ZOS 的插件	2.9.0

自定义插件管理

当您需要使用自定义插件或系统默认插件中不包含的开源插件时，可通过云搜索服务的自定义插件上传与安装功能，在实例中上传并安装对应插件。

前提条件

1. 准备待上传的插件，并确保插件的可用性和安全性。
2. 建议在上传插件前，先在本地自建 Elasticsearch 集群（与实例相同版本）上进行测试，成功后再进行上传。
3. 开通与云搜索实例同地域的对象存储服务，并上传好需要安装的插件至对象存储的桶中。

使用限制

1. 不支持安装与默认插件一样的插件，包括默认插件的其他版本。
2. 不支持同时安装/卸载两个以上的插件。

3. 云搜索服务不保留用户插件的安装文件，请您自行备份。

4. 插件文件后缀需要为.zip。

5. 不支持上传安装带权限属性的插件。

警告：安装自定义插件操作会触发实例重启插件本身可能影响实例的稳定性，请务必保证自定义插件的可用性和安全性，建议在业务低峰期进行操作。

操作步骤

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。

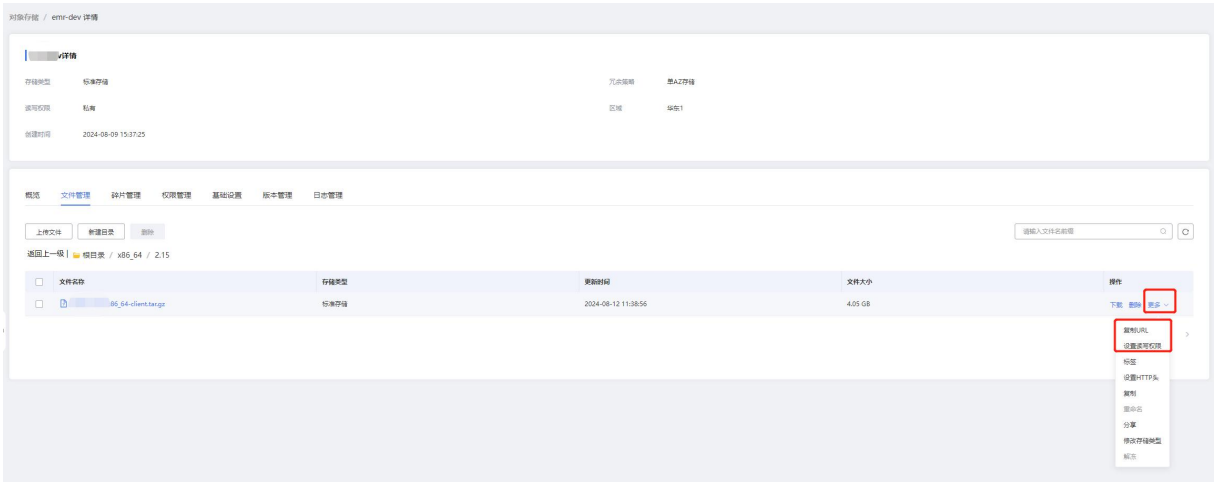
2. 在实例详情页选择“插件管理”—“自定义插件”，单击上传插件按钮。

3. 在弹出的页面上填写：

- 插件名称，仅支持字母、数字和字符，请和插件包名称保持一致；
- 插件名称查询路径：插件包内 properties 文件中找到 pluginName。
- 插件地址：填写天翼云对象存储同地域资源池下的地址，获取路径为：

对象存储控制台——点击对应桶名称进入详情——点击文件管理，找到对应的插件，点击右侧更多——复制 URL。

注意：需要开启对应文件的可读权限。



- 插件版本：插件目前的版本，格式为数字和点组合，如：7.10 或 2.9.0，非必填。
- 插件描述：用于帮助用户识别插件功能，非必填。

填写完毕后点击安装并重启，插件会先在实例安装，安装完毕后实例将自动进行重启。期间请勿对实例进行其他操作。

4. 当实例重启完成，插件状态变成“已安装”则表示插件已经安装完毕。若插件处于

“未安装”状态，则说明安装失败，您可根据提示调填写内容再次重试，或通过工单联系我们协助处理。您也可以删除该插件信息。

5. 已经安装完毕的插件，如果您不再使用，可单击插件右侧的卸载，卸载此插件。卸载插件如需再次安装，请参照前述步骤执行。

实例网络及安全设置

实例公网访问

新开启的云搜索实例默认不具备公网访问能力，您如果需要通过公网访问 Dashboards 或 OpenSearch 需要为其绑定弹性 IP 或 IPv6 带宽，并配置安全组信息，才可使用公网访问。

约束限制

1. 开启公网访问后，会因此产生流量费用，请您提前根据自身需求，购买合适的产品，公测期该费用照常收取。
2. 配置完成后，需要前往安全组设置页面，配置公网访问白名单后，才可正常使用。
3. 如果需要使用 IPv6 访问，需要在开通虚拟私有云 VPC 时即选择开通 IPv6 能力的子网，并在下单时选择该子网，不支持实例开通后再升级 IPv6。

开通 IPv6 访问能力的实例

您需要在订购时选择具备 IPv6 的虚拟私有云，选择子网后，会提示“该子网已开通 IPv6”。并在 IPv6 访问处开启开关，如关闭，则仅可通过 IPv4 访问实例。

配置实例公网访问

您可以对已开通的实例进行公网访问的配置、修改、查看、解绑操作。

1. 录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。

2. 在详情页面里选择“安全设置”，在弹出的页面上选择需要绑定的公网 IP 类型，如果为 IPv4，请在下拉列表中选择弹性 IP 地址；如果为 IPv6，请选择 IPv6 的带宽名称。如果绑定失败，可以等待几分钟后再次尝试重新绑定。绑定的弹性 IP 或 IPv6 带宽需要处于空闲状态。

IP 绑定过后，要补充安全组方可实现本地电脑公网访问 Kibana 或连接 Elasticsearch

3. 修改绑定弹性 IP 或 IPv6 带宽，也需要在当前页面选择对应要修改的项目，点击

“修改公网 IP”进行重新绑定。

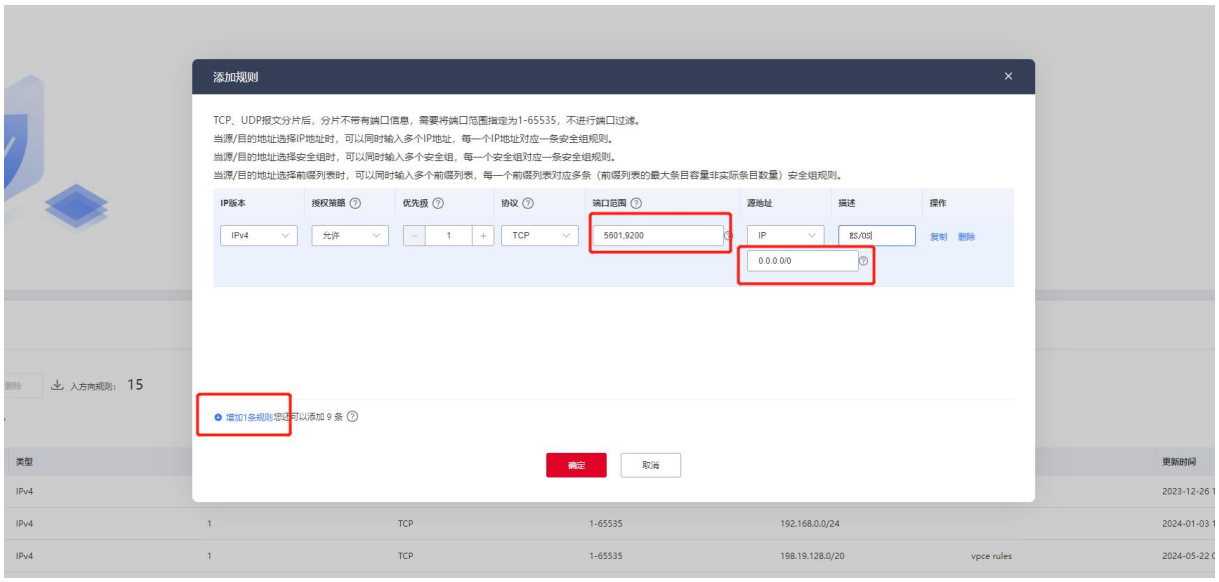
4. 解绑弹性 IP 或 IPv6 带宽，可关闭公网访问状态，或点击已绑定的项目后的解绑按钮，即可解绑当前的公网访问能力。

5. 实例退订将自动解绑已绑定的弹性 IP 或 IPv6 带宽，如弹性 IP 或 IPv6 带宽不再使用，您需要另外前往弹性 IP 的控制台退订相应的弹性 IP 或 IPv6 带宽才会停止对应产品的计费。

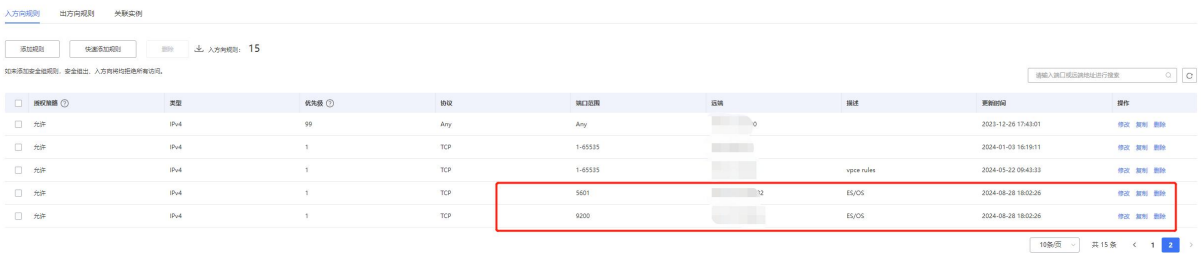
配置安全组白名单

操作步骤：

在控制台点击实例所在安全组，入方向规则点击添加规则，在弹出的填写框内的端口处填写“5601,9200”，选择需要配置的策略为 IPv4 或 IPv6，在源地址下方的 IP 地址格子中填写需要访问设备的出口公网 IP 地址，点击确定保存。



成功后会在安全组增加两条规则，此时可以通过绑定的公网 IP 地址端口访问对应对象。



通过公网 IP 地址接入实例

公网访问配置完成后，实例将会获得一个“公网访问”的 IP 地址，用户可以通过公网 IP 地址和端口接入实例。

例如，Dashboards 可直接点击页面链接进行访问。

OpenSearch 实例可以通过 Curl 命令查询索引信息

curl -u username:password -k 'https://10.62.179.32:9200/_cat/indices'其中 username 和 password 表示实例的用户名和密码。

实例密码修改

如果您在使用集群过程中需要重置集群管理员密码，可通过以下步骤进行重置。

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。

2. 在详情页面里选择“安全设置”，并点击下方的密码重置按钮，在显示的填写框处进行密码重置。请按照密码设置规范，填写新密码。

注意：

1. 密码为数字、大写字母、小写字母、特殊符号（@$!%*#_~?&）的组合。

2. 长度限制为 12-26 位。

3. 不能包含账号信息、字典序及键盘序。

管理实例

查看实例信息

在实例管理列表页、详情页，您可以查看实例版本、节点状态、使用情况等信息。

实例列表信息介绍

实例列表页为您展示全部购买的实例清单，如为子账号，则可见主账号的实例信息。

参数	描述
名称	订购时设置的实例名称，单击实例名称可以进入实例名称可以进入实例详情页。
状态	展示目前实例状态： • 运行中：正常在有效期内，运行状态正常的实例； • 创建中：刚下单还在订购开通部署中的实例； • 处理中：处于重启等正常操作下的实例； • 已冻结：包周期已到期的实例，可续费或退订销毁； • 已销毁：实例已删除，资源已回收的实例；如遇到资源不足或其他原因导致开通部署失败，将自动销毁，可再次下单或工单联系技术支持。 • 异常：集群不可连接或可连接但不可用。 注意：集群处于异常状态时，您可以尝试重启处理，若依旧失败，请及时联系技术支持。
索引数量	实例内的索引数量
存储用量	磁盘已使用的存储总量
版本	展示实例的版本号
计费模式	包年包月
到期时间	包年包月实例使用，表示包周期可使用的截止时间
创建时间	实例的创建时间
操作	展示实例可执行的操作入口，包含重启、续订等针对实例的操作，当前实例无法操作某项时，按钮将置灰。

实例基础信息页介绍

在实例的基础信息页面，可以获取实例的内网地址，版本节点信息等等。

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入

详情页。

2. 在实例的详情页可查看实例的购买信息、健康情况、内网地址、和节点的主机健康情况。
3. 实例状态说明：

- 健康：对应 OpenSearch 实例 health 为 Green 时的状态，实例健康；
- 可用：对应 OpenSearch 实例 health 为 Yellow 时的状态，主分片可用，部分副本缺失；
- 异常：对应 OpenSearch 实例 health 为 Red 时的状态，部分主分片不可用，可能已经丢失数据。

4. 右侧实例架构图中展示的为当前实例所在区域、节点数量和节点主机状态。

重启实例

实例出现异常，或您有其他需要时，可以通过控制台重启实例。

前提条件

- 实例处于运行中或异常状态，未处于其他操作引起的重启中，未被冻结；
- 当实例处于运行中时，确认已停止数据写入、检索操作，否则重启集群可能会带此时写入的数据丢失、搜索不到数据等情况。

操作步骤

1. 登录云搜索服务管理控制台，在左侧导航栏，选择对应的实例类型，进入管理列表界面。

2. 在对应实例的**“操作”**列中单击“更多>重启”。

3. 重启实例后，请刷新页面，观察状态。重启过程中，实例状态为“处理中（重启）”，如果实例状态变更为“运行中”，表示实例已重启成功。

实例监控

实例提供支持查看云搜索服务实例的核心指标监控能力，方便用户掌握实例情况，及时处理实例异常。

使用限制

每次查询最多可选择 10 个节点/索引进行查看。

操作步骤

1. 登录云搜索服务控制台，进入实例管理列表页，选择需要查看的实例点击操作栏的监控按钮或点击实例名称，进入实例详情页。
2. 在实例详情页选择实例监控。
3. 实例监控共提供 3 种维度的指标查看

- 实例级：提供当前实例的整体情况，部分指标可切换查看各项指标的最大值和平均值；
- 节点级：选择单个或多个节点查看对应指标；
- 索引级：选择实例中已有的 open 状态的索引一个或多个进行查看。

云搜索服务支持的指标清单：

集群支持查看的指标

指标名称	指标含义	取值范围
健康状态	该指标用于统计测量监控对象的状态。	green（健康） yellow（可用） red（异常）
磁盘使用率	该指标用于统计测量对象的磁盘使用率。	0-100%
JVM 内存使用率	实例各个节点平均/最大 JVM 使用率	0-100%

CPU 使用率	实例各个节点平均/最大 CPU 使用率	0-100%
实例写入 QPS	当前实例的每秒写入速率	≥ 0
实例查询 QPS	当前实例的每秒查询速率	≥ 0
实例索引数量	当前实例的索引数量	≥ 0
实例分片数量	当前实例的分片数量	≥ 0
实例文档数量	当前实例的文档数量	≥ 0
写入延迟	完成单次索引写入的平均/最大时间	≥ 0 ms
查询延迟	完成单次搜索操作所需的平均/最大时间	≥ 0 ms
1 分钟负载	实例 1 分钟所有节点的平均/最大负载	≥ 0

节点维度

指标名称	指标含义	取值范围
磁盘使用率	该指标用于统计测量对象的磁盘使用率。	0-100%
CPU 使用率	指定节点 CPU 使用率	0-100%
内存使用率	指定节点 CPU 使用率	0-100%

索引维度

指标名称	指标含义	取值范围
索引文档数	索引中所包含的文档数量	≥ 0
索引总耗时	对应索引在监控时间范围内的所有操作总耗时	≥ 0ms
get 总请求数	对应索引在监测时间范围的 get 请求数	≥ 0
get 请求总耗时	对应索引在监测时间范围的 get 请求总耗时	≥ 0ms
search 总请求数	对应索引在监测时间范围的 search 请求数	≥ 0
search 请求总耗时	对应索引在监测时间范围的 search 请求总耗时	≥ 0ms

增强特性

概览

天翼云在开源 Elasticsearch 和 OpenSearch 的基础上做了大量内核能力增强。具体支持对应表如下：

功能项	天翼云云搜索 OpenSearch 增强	天翼云云搜索 Elasticsearch 增强	开源 Elasticsearch	开源 OpenSearch
向量检索	支持	支持	不支持	支持
压缩算法	支持	支持	不支持	支持
跨集群复制	支持	支持	不支持	支持
SQL 功能	支持	支持	不支持	支持
简繁体转换	支持	支持	不支持	不支持
细粒度权限	支持	支持	不支持	支持
异步搜索	支持	支持	不支持	支持
流量控制	支持	不支持	不支持	不支持
分词增强	支持	支持	不支持	不支持

向量检索

支持向量检索（Vector Search）是搜索引擎的一个高级功能，它允许用户在高维向量空间中进行相似性搜索，而不仅仅是基于传统的关键词匹配。

向量检索的核心在于，它通过将文本、图像或其他数据转换为向量（即一组多维的数值表示），然后基于这些向量之间的距离来查找相似的项目。与传统的基于关键字的检索方法相比，向量检索更适合处理复杂的数据类型，如自然语言处理、推荐系统和计算机视觉等场景。

在搜索引擎中，支持向量检索的功能通过集成高效的向量索引结构（如近似最近邻搜索算法）实现。这使得搜索引擎能够在处理大规模数据集时，依然保持高效的查询速度和准确性。通过向量检索，用户可以在海量数据中快速找到与查询向量最相似的结果，从而提升搜索体验和应用的智能化水平。

Elasticsearch 支持向量检索示例：

创建索引：

```
PUT my-knn-index-1
{
"settings": {
"index": {
"knn": true,
"knn.algo_param.ef_search": 100
    }
  },
"mappings": {
"properties": {
"category": {
"type": "keyword"
},
"brand": {
"type": "keyword"
},
"style": {
"type": "keyword"
},
"my_vector": {
"type": "knn_vector",
"dimension": 3
}
    }
  }
}
```

插入数据：

```
PUT my-knn-index-1/_doc/1
{
"category": "electronics",
"brand": "brandA",
"style": "modern",
"my_vector": [0.5, 0.8, 0.3]
}
PUT my-knn-index-1/_doc/2
{
"category": "furniture",
"brand": "brandB",
"style": "vintage",
"my_vector": [0.2, 0.4, 0.7]
}
PUT my-knn-index-1/_doc/3
{
"category": "clothing",
"brand": "brandC",
"style": "casual",
"my_vector": [0.9, 0.1, 0.6]
}
```

查询：

```
POST my-knn-index-1/_search
{
"size": 10,
"query": {
"knn": {
```

"my_vector": {
"vector": [0.5, 0.8, 0.3],
"k": 2
}
    }
  }
}

返回结果：

{
"took" : 654,
"timed_out" : false,
"_shards" : {
"total" : 1,
"successful" : 1,
"skipped" : 0,
"failed" : 0
  },
"hits" : {
"total" : {
"value" : 3,
"relation" : "eq"
},
"max_score" : 1.0,
"hits" : [
{
"_index" : "my-knn-index-1",
"_type" : "_doc",
"_id" : "1",
"_score" : 1.0,

"_source" : {
"category" : "electronics",
"brand" : "brandA",
"style" : "modern",
"my_vector" : [
0.5,
0.8,
0.3
]
}
},
{
"_index" : "my-knn-index-1",
"_type" : "_doc",
"_id" : "2",
"_score" : 0.7092199,
"_source" : {
"category" : "furniture",
"brand" : "brandB",
"style" : "vintage",
"my_vector" : [
0.2,
0.4,
0.7
]
}
},
{
"_index" : "my-knn-index-1",

"_type" : "_doc",
"_id" : "3",
"_score" : 0.57471263,
"_source" : {
"category" : "clothing",
"brand" : "brandC",
"style" : "casual",
"my_vector" : [
0.9,
0.1,
0.6
]
}
}
]
}
}

压缩算法

天翼云云搜索服务，在 OpenSearch 和 Elasticsearch 都实现了对多种压缩算法的支持，其中包括 Zstandard (zstd) 压缩算法。

Zstandard 是一种无损压缩算法，具有高压缩率和高速压缩/解压速度的优点。相较于传统的压缩算法（如 Gzip 或 LZ4），Zstandard 可以在更短的时间内实现更好的压缩比，这对于需要处理大量数据的搜索引擎尤其有利。

通过支持 zstd 压缩算法，搜索引擎能够有效降低索引和存储的磁盘空间需求，同时加快数据传输速度，进一步优化搜索引擎的性能。这对于需要处理大量数据的应用场景，尤其是高并发和低延迟要求的场景，显得尤为重要。

Codec	Indexing time (ms)	Disk usage (MB)	Retrieval time per 10k docs (ms)

BEST_SPEED	35383	90.175	190.17524
BEST_COMPRESSION (vanilla zlib)	76671	58.682	1910.42106
BEST_COMPRESSION (Cloudflare zlib)	54791	58.601	1395.53593
ZSTD (level=1)	42433	70.527	240.04036
ZSTD (level=3)	53426	68.737	259.61897
ZSTD (level=6)	100697	66.283	251.91177
ZSTD dict (level=1)	50571	69.860	254.10496
ZSTD dict (level=3)	60580	68.690	266.72929
ZSTD dict (level=6)	128322	65.605	251.91177

使用示例：

在 OpenSearch 中创建一个使用 zstd 作为压缩算法的索引的命令：

```
PUT my_index
{
  "settings": {
    "index": {
      "codec": "zstd"
    }
  }
}
```

跨集群复制

跨集群复制（Cross-Cluster Replication，CCR）是搜索引擎的一项重要功能，旨在实现不同集群间的实时数据同步和复制。这一功能对于构建全球分布式系统、提高数据高可用性、支持灾备恢复、以及优化跨地域数据访问具有显著意义。

天翼云云搜索服务在 OpenSearch 和 Elasticsearch 都实现了跨集群复制功能。

核心原理

跨集群复制通过在不同的搜索集群之间建立主从索引关系来实现数据同步。一个集群中的索引被设为主索引（Leader Index），负责处理数据的写入操作。其他集群中的从索引（Follower Index）则持续地从主索引中拉取数据更新，确保各个集群的数据保持一致。

这种主从架构确保了写入操作的集中管理，减少了数据冲突的风险，并维护了数据的一致性。

应用场景与优势

分布式系统

在更大范围内部署应用时，CCR 允许数据在不同地理位置的集群之间同步。通过在用户所在地附近的集群中设置从索引，可以显著降低访问延迟，提高查询性能。例如，上海的主集群可以通过 CCR 将数据复制到北京的从集群，从而优化这些地区用户的体验。

灾备恢复

CCR 是构建高可用性和灾备系统的关键。主集群若发生故障，其他地理位置的从集群可以迅速接管，保障业务连续性。这种多集群架构能够有效防范单点故障，提高系统的可靠性。

多集群架构的弹性扩展

随着业务的增长，CCR 支持将数据和查询负载分布到多个集群中，从而实现弹性扩展。通过部署多个从集群来分担查询压力，可以提升系统的整体性能，满足更高的用户需求和并发请求。

读写分离

在高性能场景下，CCR 支持读写分离。主集群专注于处理数据写入，而从集群则负责处理查询请求。这种架构优化了查询性能，尤其适合需要高频读写操作的大数据环境。

技术实现与管理

部署跨集群复制需要在集群之间建立信任关系，并配置索引的复制参数。搜索引擎提供了灵活的 API 和管理工具，方便用户管理和控制跨集群复制任务。通过这些工具，用

户可以轻松启动或暂停复制、调整复制策略，以及根据业务需求灵活配置复制模式。

CCR 支持实时复制和按需复制，用户可以根据具体需求选择适合的复制方式。无论是对数据的一致性要求极高的场景，还是需要降低跨集群网络传输开销的场景，CCR 都能提供可靠的解决方案。

操作示例

在 leader 集群插入数据：

```
PUT ccr_test
{"settings": {"number_of_shards": 1,"number_of_replicas": 0}}
POST ccr_test/_doc/
{
    "name": "robert",
    "age": 30,
    "gender": "male"
}
```

在 follower 集群配置：

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "remote": {
        "leader-cluster": {
          "seeds": ["ip:9300"]
        }
      }
    }
  }
}
```

开始复制：

```
PUT _opendistro/_replication/follower-01/_start
{
```

```
"remote_cluster":"leader-cluster",
  "remote_index": "ccr_test"
}
```

返回：

```
{
  "acknowledged" : true
}
```

在 leader 集群插入数据：

```
POST ccr_test/_doc/
{
    "name": "jane",
    "age": 25,
    "gender": "female"
}
```

再插一条：

```
POST ccr_test/_doc/
{
    "name": "Jane",
    "age": 18,
    "gender": "female"
}
```

在 follower 集群查询,数据会自动复制过去：

```
GET follower-01/_search
```

返回结果：

```
{
  "took" : 435,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
```

```
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 2,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" : "follower-01",
      "_type" : "_doc",
      "_id" : "WU_QFo4BrTIYgmxo8ErH",
      "_score" : 1.0,
      "_source" : {
        "name" : "robert",
        "age" : 30,
        "gender" : "male"
      }
    },
    {
      "_index" : "follower-01",
      "_type" : "_doc",
      "_id" : "pGiQF44BZY5qpm7NPGR_",
      "_score" : 1.0,
      "_source" : {
        "name" : "Jane",
```

```
            "age" : 18,
            "gender" : "female"
          }
        }
      ]
    }
}
```

SQL 功能

天翼云云搜索服务中的 OpenSearch 和 Elasticsearch 都支持 SQL 查询，这是其数据查询与分析能力的一个重要扩展。通过将 SQL 语言引入搜索引擎，用户可以使用熟悉的关系数据库查询语法对非结构化和半结构化的数据进行查询与分析。这种支持极大地降低了学习成本，使得传统数据库用户能够轻松过渡到使用搜索引擎进行复杂数据操作。

核心原理

SQL 是一种广泛使用的结构化查询语言，传统上用于关系数据库中数据的查询和管理。搜索引擎引入 SQL 支持后，用户可以直接在搜索引擎中运行标准的 SQL 查询语句。这些查询语句被翻译为搜索引擎的原生查询 DSL（Domain-Specific Language），并在后台执行。用户不仅能够通过 SELECT、WHERE、GROUP BY、ORDER BY 等常见的 SQL 语法来检索数据，还可以进行复杂的聚合操作和数据过滤。

应用场景与优势

降低学习成本：

对于熟悉 SQL 的用户，搜索引擎提供了一个熟悉的查询接口，无需学习新的查询语言即可开始使用。这对于企业级用户，尤其是那些已有大量 SQL 查询场景的团队，降低了迁移成本。

复杂查询与分析：

SQL 允许用户轻松编写复杂的查询，包括多层次的聚合、连接、子查询等操作。这使得搜索引擎成为强大的数据分析工具，能够处理结构化和半结构化数据的多维度分析。

兼容性与集成：

搜索引擎的 SQL 支持可以与各种 BI 工具、数据可视化平台和 ETL 流程无缝集成。这意味着用户可以在现有的 SQL 工作流中直接利用搜索引擎，执行实时分析和报告生成。

实时分析：

通过 SQL 查询，用户可以对实时索引的数据进行分析，从而在海量数据中快速提取有

价值的信息。这对于需要实时监控和数据洞察的业务场景，显得尤为重要。

技术实现与管理

使用搜索引擎的 SQL 支持非常简单。用户可以通过 OpenSearch Dashboards / Kibana 提供的 SQL 工作台直接运行 SQL 查询，或通过 REST API 接口在应用程序中集成 SQL 查询。搜索引擎将这些 SQL 查询转换为原生的查询语句，并返回结果。

搜索引擎还支持 SQL 的多种格式输出，包括 JSON、CSV、TXT 等，方便用户在不同场景中使用查询结果。此外，用户可以使用 SQL 进行复杂的聚合查询和时间序列分析，充分利用 搜索引擎强大的数据处理能力。

操作示例：

我们以 Elasticsearch 为例。

创建索引：

```
PUT employees
{
  "mappings": {
    "properties": {
      "name": { "type": "text" },
      "age": { "type": "integer" },
      "position": { "type": "text" },
      "department": { "type": "text" }
    }
  }
}
```

插入数据：

```
POST employees/_doc
{
  "name": "John Doe",
  "age": 30,
  "position": "Software Engineer",
  "department": "Engineering"
}
```

使用 SQL 进行查询：

```
POST _opendistro/_sql
{
  "query": "SELECT * FROM employees WHERE age > 25"
}
```

返回结果：

```
{
  "schema": [
    {
      "name": "name",
      "type": "text"
    },
    {
      "name": "position",
      "type": "text"
    },
    {
      "name": "department",
      "type": "text"
    },
    {
      "name": "age",
      "type": "integer"
    }
  ],
  "datarows": [
    [
      "John Doe",
```

```
      "Software Engineer",
      "Engineering",
      30
    ]
  ],
  "total": 1,
  "size": 1,
  "status": 200
}
```

通过支持 SQL，搜索引擎极大地扩展了其数据查询与分析的能力，使用户能够在一个平台上利用标准 SQL 语法处理大规模的非结构化数据。无论是在降低学习门槛、增强数据分析能力，还是在兼容现有 SQL 工作流与实时数据分析等方面，搜索引擎的 SQL 支持都为用户提供了一个强大而灵活的数据操作工具。

简繁体转换

天翼云云搜索服务中的 OpenSearch 和 Elasticsearch 都支持简繁体转换功能，这是其文本处理和搜索能力的一项重要增强。通过集成简繁体转换，搜索引擎能够在处理中文内容时自动进行简体与繁体字的相互转换，从而提升搜索的准确性和用户体验。这个功能对使用不同中文书写系统的用户尤其有用，确保了无论是简体还是繁体中文，都可以获得一致的搜索结果。

核心原理

中文存在简体和繁体两种书写形式，不同地区的用户可能使用不同的形式。然而，在搜索系统中，用户希望无论使用哪种形式输入，系统都能返回相关的结果。搜索引擎通过内置的简繁体转换功能，可以在数据索引和查询阶段自动进行转换。

在数据索引阶段，搜索引擎可以将存储的文本内容统一转换为简体或繁体形式，从而标准化数据。在查询阶段，当用户输入简体或繁体查询词时，系统会自动将其转换为与索引数据一致的形式进行匹配。这种双向转换确保了搜索的全面性和一致性。

应用场景与优势

提升搜索准确性：

通过简繁体转换，用户无论输入简体还是繁体字，系统都能准确地匹配到相关内容。这大大提高了搜索的准确性，减少了因书写形式不同而导致的搜索结果不一致问题。

用户体验优化：

对于面向全球华人用户的应用程序和网站，简繁体转换功能能够确保不同地区的用户都能获得一致的搜索体验，无需手动切换书写形式。这提升了跨地区用户的满意度。

支持多语言环境：

在多语言或多地区的应用中，搜索引擎的简繁体转换功能帮助开发者轻松管理和处理不同中文形式的数据，确保多语言环境中的中文内容都能被正确索引和检索。

文本标准化：

对于需要进行文本分析或数据挖掘的场景，简繁体转换功能可以将文本内容标准化，统一成一种形式进行处理，从而简化分析过程并提高数据处理效率。

技术实现与管理

启用简繁体转换功能非常简单。用户可以在搜索引擎的索引设置中配置相应的转换器，在数据索引时指定需要将文本内容转换为简体或繁体。查询时，搜索引擎会自动处理用户输入的查询词，将其与标准化后的数据进行匹配。

此外，搜索引擎的简繁体转换功能支持多种配置，用户可以根据具体需求选择仅在索引时转换、仅在查询时转换，或同时在索引和查询时都进行转换。

操作示例：

创建索引：

```
PUT teststconvert
{
    "settings": {
        "analysis": {
            "analyzer": {
                "tsconvert": {
                    "tokenizer": "tsconvert"
                }
            },
            "tokenizer": {
                "tsconvert": {
                    "type": "stconvert",
                    "delimiter": "#",
                    "keep_both": false,
```

```
                    "convert_type": "t2s"
                }
            },
            "filter": {
                "tsconvert": {
                    "type": "stconvert",
                    "delimiter": "#",
                    "keep_both": false,
                    "convert_type": "t2s"
                }
            },
            "char_filter": {
                "tsconvert": {
                    "type": "stconvert",
                    "convert_type": "t2s"
                }
            }
        }
    }
}
```

测试分词器：

```
GET teststconvert/_analyze
{
  "tokenizer": "keyword",
  "filter": ["lowercase"],
  "char_filter": ["tsconvert"],
  "text": "国际國際"
}
```

返回结果：


```
{
  "tokens" : [
    {
      "token" : "国际国际",
      "start_offset" : 0,
      "end_offset" : 4,
      "type" : "word",
      "position" : 0
    }
  ]
}
```


通过支持简繁体转换，搜索引擎在中文内容的处理和搜索方面提供了更大的灵活性和准确性。无论是在提升搜索精度、优化用户体验，还是在支持多语言环境和文本标准化方面，简繁体转换功能都为用户提供了一个强大的工具，确保在复杂的中文书写环境中实现一致和高效的搜索体验。

细粒度权限

天翼云云搜索服务中的 OpenSearch 和 Elasticsearch 都支持细粒度权限管控，包括文档级别和字段级别的权限控制，为企业级用户提供了精确的数据访问管理能力。这一功能使得管理员能够针对不同的用户或角色，灵活地配置他们对特定文档或字段的访问权限，确保敏感信息得到有效保护，同时实现数据的高效共享和管理。

核心原理

细粒度权限管控允许管理员在多个层次上定义用户访问权限。具体来说，文档级别的权限控制使得管理员能够基于文档的内容或属性，决定某个用户是否可以访问或查询该文档。字段级别的权限控制则进一步细化，允许管理员指定某些用户只能查看或操作文档中的特定字段，而隐藏其他字段内容。

例如，在一个包含敏感信息的客户数据库中，管理员可以配置权限，使得某些用户只能访问客户的联系方式，而不能查看财务信息或个人身份信息。通过这样的精细控制，搜索引擎能够确保数据的安全性和合规性，同时满足不同业务场景下的访问需求。

应用场景与优势

数据安全与隐私保护：

通过文档级别和字段级别的权限控制，企业可以确保敏感信息仅对经过授权的用户可见。这在金融、医疗等对数据隐私有严格要求的行业中尤为重要，能够帮助企业满足合规性要求。

个性化数据访问：

细粒度权限管控允许为不同用户或角色定制数据访问视图。例如，在一个销售系统中，销售人员可能只需要访问客户的基本信息，而经理则可以查看更详细的销售记录和分析数据。这种个性化的权限配置提高了工作效率。

多租户环境：

在多租户环境中，细粒度权限管控能够有效隔离不同租户的数据，确保每个租户只能访问自己的数据。即使多个租户共享同一个搜索引擎集群，他们的数据依然能够得到严格的保护。

最小权限原则：

细粒度权限控制支持最小权限原则（Principle of Least Privilege），确保用户仅能访问其工作所需的最小数据范围，从而减少潜在的安全风险。

技术实现与管理

管理员可以通过 OpenSearch 的安全模块，使用文档级别安全（Document-Level Security, DLS）和字段级别安全（Field-Level Security, FLS）配置细粒度权限。DLS 允许基于查询条件限制用户对特定文档的访问，而 FLS 则允许管理员指定哪些字段对特定用户可见或不可见。

这些权限配置可以通过 OpenSearch 的 REST API 或管理工具来实现和管理。管理员还可以结合角色和用户组的概念，为不同用户群体配置统一的权限策略，从而简化权限管理流程。

操作示例：

我们创建一个 role public_role，

创建一个 user public_user1，

我们目标是让这个 user 只能查看 pub 开头的索引里 public 字段为 true 的文档。

创建角色：

```
PUT _plugins/_security/api/roles/public_role
{
  "cluster_permissions": [
```

```
    "*"
  ],
  "index_permissions": [{
    "index_patterns": [
      "pub*"
    ],
    "dls": "{\"term\": { \"public\": \"true\"}}",
    "allowed_actions": [
      "read"
    ]
  }]
}
```

创建用户：

```
PUT _plugins/_security/api/internalusers/public_user1
{
  "password": "******"
}
```

创建 Mapping：

```
PUT _plugins/_security/api/rolesmapping/public_role
{
  "users" : [ "public_user1" ]
}
```

创建索引：

pub 索引，插入两条数据

```
POST pub_index/_doc/
{
    "name": "robert",
    "age": "30",
    "public": "true"
```

```
}
POST pub_index/_doc/
{
    "name": "mike",
    "age": "55",
    "public": "false"
}
```

非 pub 索引

```
POST sec_index/_doc/
{
    "name": "jane",
    "age": "18",
    "public": "true"
}
```

我们开始搜索，预期是 public_user1 搜索 pub_index 可以搜出一条，搜索 sec_index 搜不出来。而 admin 用户搜索 pub_index 可以搜出两条，搜索 sec_index 可以搜出一条。

```
GET pub_index/_search
{"size": 10,"query": {"match_all": {}}}
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
```

```
      "relation" : "eq"
    },
    "max_score" : 2.0,
    "hits" : [
      {
        "_index" : "pub_index",
        "_id" : "xBnhOokBxCORqeQrGobf",
        "_score" : 2.0,
        "_source" : {
          "name" : "robert",
          "age" : "30",
          "public" : "true"
        }
      }
    ]
  }
}
GET pub_index/_search
{"size": 10,"query": {"match_all": {}}}
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
```

```
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "pub_index",
        "_id" : "xBnhOokBxCORqeQrGobf",
        "_score" : 1.0,
        "_source" : {
          "name" : "robert",
          "age" : "30",
          "public" : "true"
        }
      },
      {
        "_index" : "pub_index",
        "_id" : "xRnhOokBxCORqeQrMobq",
        "_score" : 1.0,
        "_source" : {
          "name" : "mike",
          "age" : "55",
          "public" : "false"
        }
      }
    ]
  }
}
```

```
GET sec_index/_search
{"size": 10,"query": {"match_all": {}}}
{
  "error" : {
    "root_cause" : [
      {
        "type" : "security_exception",
        "reason" : "no permissions for [indices:data/read/search] and User
[name=public_user1, backend_roles=[], requestedTenant=null]"
      }
    ],
    "type" : "security_exception",
    "reason" : "no permissions for [indices:data/read/search] and User
[name=public_user1, backend_roles=[], requestedTenant=null]"
  },
  "status" : 403
}
GET sec_index/_search
{"size": 10,"query": {"match_all": {}}}
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
```

```
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "sec_index",
        "_id" : "xhnhOokBxCORqeQrTYbM",
        "_score" : 1.0,
        "_source" : {
          "name" : "jane",
          "age" : "18",
          "public" : "true"
        }
      }
    ]
  }
}
```


搜索引擎的细粒度权限管控功能通过支持文档级别和字段级别的权限控制，为企业提供了强大的数据安全和管理能力。这种精确的权限配置不仅帮助企业保护敏感信息，还能够在多租户环境和个性化数据访问需求下提供高效的解决方案。通过细粒度的权限管控，企业可以确保数据的安全性、隐私性和合规性，同时实现灵活的业务支持。

异步搜索

天翼云云搜索服务中的 OpenSearch 和 Elasticsearch 都支持异步搜索功能（Asynchronous Search），这一功能极大地提升了在处理长时间运行查询时的用户体验和系统效率。通过异步搜索，用户可以在后台执行耗时较长的查询任务，而无需等待查询结果的即时返回。这一功能对于大数据集、复杂查询以及需要持续获取查询状态的场景特别有用。

核心原理

在传统的同步搜索模式中，用户发出查询请求后，必须等待查询结果返回才能继续其他操作。如果查询涉及大规模数据处理或复杂的计算，这可能会导致用户界面的阻塞和等待时间过长。异步搜索通过将查询任务分离到后台执行，解决了这一问题。

当用户发起异步搜索请求时，搜索引擎会立即返回一个查询任务 ID，而查询本身在后台继续运行。用户可以通过这个任务 ID 随时查询任务的进展情况、获取部分结果或在任务完成后检索最终的完整结果。这种模式下，用户可以在查询结果生成的过程中继续执行其他操作，显著提高了系统的响应性和用户的工作效率。

应用场景与优势

处理复杂查询：

对于涉及大量数据处理或复杂计算的查询，异步搜索允许这些任务在后台执行，避免了用户界面因长时间等待而被冻结的情况。这尤其适用于分析海量数据、执行深度聚合或跨多索引的查询任务。

提高系统性能与效率：

异步搜索将长时间运行的任务移至后台执行，减少了同步操作对系统资源的占用，优化了集群的整体性能。同时，用户能够并行处理其他任务，提升了操作效率。

断点续查与容错处理：

异步搜索支持断点续查，用户可以在查询任务中断或超时时重新查询任务状态或继续执行未完成的任务。这种容错机制增强了查询的可靠性，特别是在网络波动或系统故障情况下。

查询任务管理：

用户可以通过任务 ID 管理和监控查询任务，包括取消正在运行的查询、检查任务进度、或者在任务完成后获取结果。这种灵活的任务管理方式为用户提供了更好的控制能力。

技术实现与管理

使用异步搜索功能非常简单，用户可以通过搜索引擎的 REST API 发起异步查询请求。系统会返回一个任务 ID，用户可以使用该 ID 查询任务状态、获取中间结果或最终结果。搜索引擎还提供了管理接口，允许用户查看当前的异步任务列表、取消任务或调整任务的超时时间。

在高并发或复杂查询场景下，异步搜索减少了前端的等待时间，使得用户可以在任务后台执行时继续其他工作，从而提高了工作流的效率和用户体验。

操作示例：

我们以 Elasticsearch 为例。

创建索引

```
PUT city-data
{
  "settings": {
    "number_of_replicas": 0
  },
  "mappings": {
    "properties": {
      "city": {
        "type": "keyword"
      }
    }
  }
}
```

插入数据

```
POST city-data/_doc
{
  "city": "Shanghai"
}
POST city-data/_doc
{
  "city": "Beijing"
}
POST city-data/_doc
{
  "city": "Guangzhou"
}
```

提交异步搜索请求

```
POST
_opendistro/_asynchronous_search/?pretty&size=10&wait_for_completion_timeou
t=1ms&keep_on_completion=true&request_cache=false
{
  "aggs": {
    "city": {
      "terms": {
        "field": "city",
        "size": 10
      }
    }
  }
}
```

这个命令会返回一个 id：

```
  "id" :
"FnJRN1FZek04UTFldERyWThfZUtMaFEDOTMyFEhuZkZFbzRCbERJd09IVCOzTG9IATY=",
```

获取异步搜索结果：

```
GET
_opendistro/_asynchronous_search/FnJRN1FZek04UTFldERyWThfZUtMaFEDOTMyFEhuZk
ZFbzRCbERJd09IVCOzTG9IATY=?pretty
```

返回：

```
{
  "id" :
"FnJRN1FZek04UTFldERyWThfZUtMaFEDOTMyFEhuZkZFbzRCbERJd09IVCOzTG9IATY=",
  "state" : "STORE_RESIDENT",
  "start_time_in_millis" : 1709711940616,
  "expiration_time_in_millis" : 1709798340616,
  "response" : {
    "took" : 90,
```

```
"timed_out" : false,
"_shards" : {
  "total" : 3,
  "successful" : 3,
  "skipped" : 0,
  "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 3,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" : "city-data",
      "_type" : "_doc",
      "_id" : "G3fFEo4BlDIwOHT-froB",
      "_score" : 1.0,
      "_source" : {
        "city" : "Shanghai"
      }
    },
    {
      "_index" : "city-data",
      "_type" : "_doc",
      "_id" : "HHfFEo4BlDIwOHT-frrH",
      "_score" : 1.0,
      "_source" : {
```

```
        "city" : "Beijing"
      }
    },
    {
      "_index" : "city-data",
      "_type" : "_doc",
      "_id" : "HXfFEo4BlDIwOHT-f7oM",
      "_score" : 1.0,
      "_source" : {
        "city" : "Guangzhou"
      }
    }
  ]
},
"aggregations" : {
  "city" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "Beijing",
        "doc_count" : 1
      },
      {
        "key" : "Guangzhou",
        "doc_count" : 1
      },
      {
        "key" : "Shanghai",
```

```
            "doc_count" : 1
          }
        ]
      }
    }
  }
}
```

搜索引擎的异步搜索功能为处理复杂和长时间运行的查询提供了极大的灵活性和效率。通过将查询任务移至后台执行，异步搜索不仅提升了系统的性能，还改善了用户的操作体验。无论是在处理大数据集、优化系统资源利用，还是在实现查询任务的灵活管理方面，异步搜索都为用户提供了一个强大而高效的解决方案。

流量控制

仅支持 OpenSearch2.9.0 版本

Flowcontrol 流量控制插件是天翼云搜索引擎团队自研的插件，可以帮助提高集群的高可用性和稳定性。该插件可以实时监控集群的 Search 请求情况，将集群的请求流量控制在一个合理的范围内。由于 OpenSearch 和 Elasticsearch 搜索引擎本身并不聚焦于流量管控方面建设，因此，Flowcontrol 组件可以在搜索引擎内部，不引入额外组件，实现流量管控功能。

操作示例

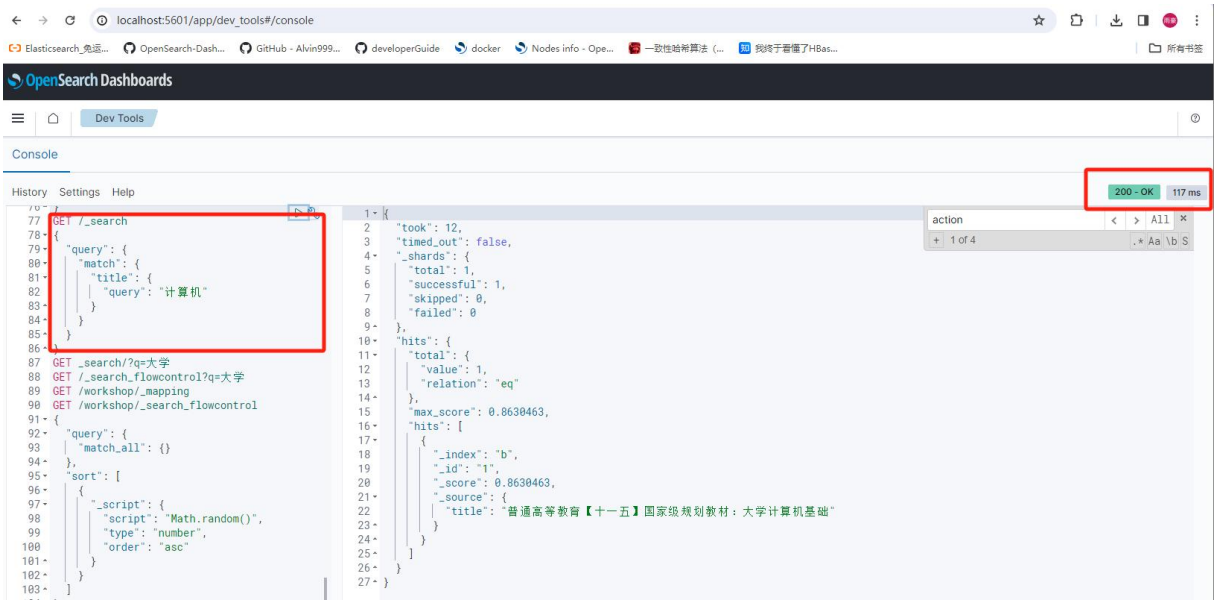
在 OpenSearch 组件的 config/opensearch.yml 配置文件中已默认设置了集群的 QPS 值为 100，假如我们希望设置集群 QPS 为 1000，可以调用 Rest API 来配置

```
PUT _cluster/settings
{
  "transient": {
    "flowcontrol.search.qps":"1000"
  }
}
```

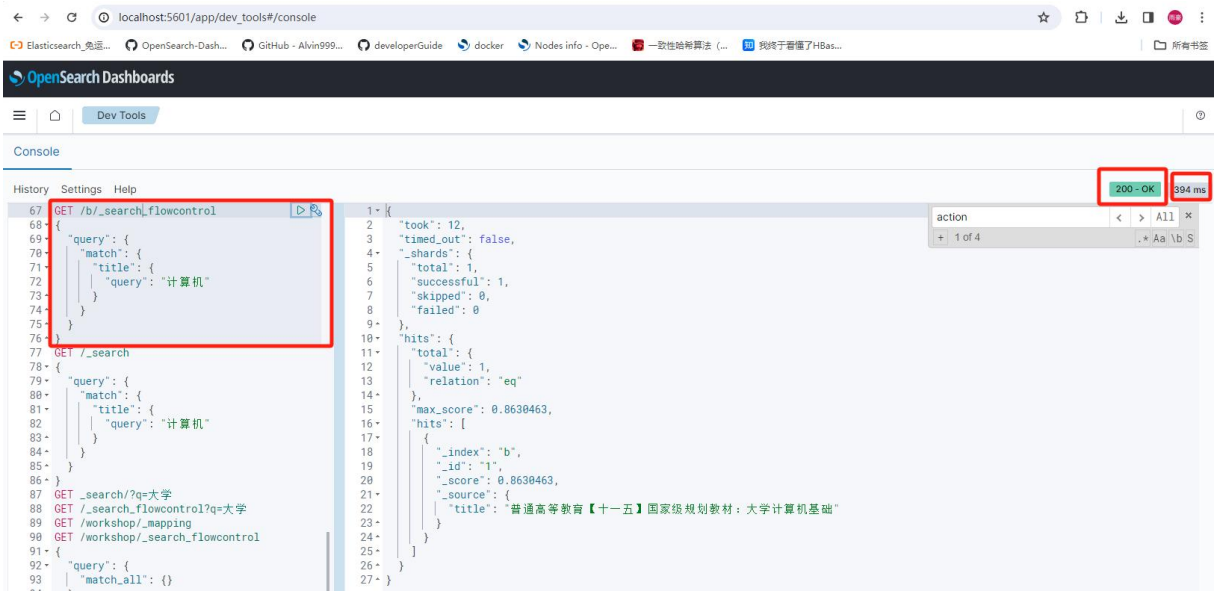
使用说明

因为依赖插件机制实现，所以，需要对原来的 endponit 进行简单改写，将_search 替换为_search_flowcontrol 即可，其他所有查询等均无需改动。

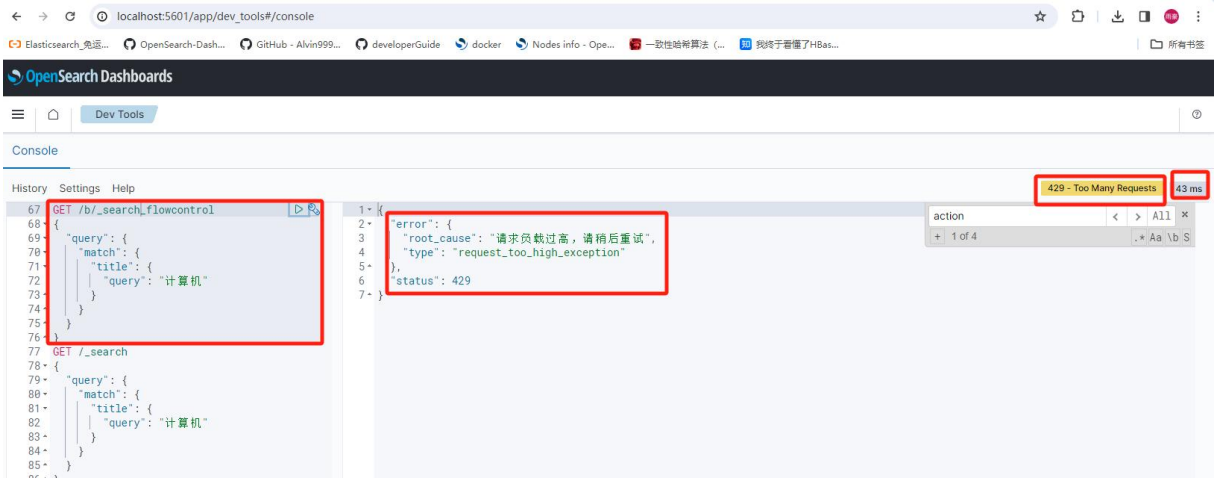
首先，作为参照，我们记录一次正常的 search 请求：



然后，当集群压力很小的时候，我们替换流量控制 endpoint 以后，进行查询，会得到一模一样的返回结果，论证了替换 endpoint 对于搜索结果是完全无影响的。如下图所示：



之后，我们通过一个多线程的脚本，模拟高通量查询，压测使得集群的 QPS 超过阈值。我们再进行同样的请求：



可以看到，请求返回了 http_code=429 的异常，提示了集群负载过高的异常提示。不仅如此，flowcontrol 流量控制插件，可以有效减少高负载情况下，集群内对索引的查询、聚合等操作，可以一定程度上避免集群过高负载带来的 OOM 等问题，从而有效提升集群的稳定性和可靠性。

中文分词增强

在全文搜索引擎如 OpenSearch 和 Elasticsearch 中，分词器是其核心功能。分词器负责将输入的字符流分割成独立的词元，这些词元是后续搜索和索引操作的基础。通过分词，连续的文本被转换为一系列便于计算机处理的单元，从而实现全文搜索。OpenSearch 和 Elasticsearch 提供了一些默认的分词器，但是只支持英文分词，对中文分词不友好。

OpenSearch 和 Elasticsearch 有相关的开源中文分词插件，如 IK 分词器和 HanLP 分词器。然而开源版本的中文分词器在处理中文文本时仍然会出现一些歧义问题或未登陆词汇的情况，因此天翼云云搜索服务对中文分词进行了增强。

天翼云云搜索服务中的 OpenSearch 和 Elasticsearch 对 HanLP 分词器进行中文分词增强，这是对其文本处理和搜索能力的一项重要增强。通过增强中文分词能力，搜索引擎能够在处理中文文本时更加精准地分词，从而提升搜索结果的准确性和用户体验。

核心原理

在 OpenSearch 和 Elasticsearch 中，中文分词器负责将输入的中文文本（如文档内容）分割成独立的词元，处理后的词元被存储在 Elasticsearch 的倒排索引中，以便后续的搜索操作能够快速检索到相关文档。搜索阶段，用户输入的查询字符串同样需要经过分词器的处理，分词器将查询字符串分割成词元，这些词元将用于在倒排索引中查找匹配的文档。搜索过程可能会涉及多个词元的组合查询，搜索引擎会根据查询语句的语法和逻辑，执行相应的搜索算法来找到匹配的文档。中文分词器的选择和配置对搜索效果和性能具有重要影响。用户需要根据文本的语言、特点和需求，选择合适的分词器。

天翼云云搜索服务提供的中文分词器

IK 分词器：天翼云云搜索服务中的 IK 分词器是开源分词器，包括最大化分词模式（ik_max_word）、最小化分词模式（ik_smart）。允许自定义词典，支持定制化的分词处理，适应特定场景和行业的需求。

IK 插件包含的分词器:ik_smart、ik_max_word。

ik_smart 模式：智能分词模式，采用较为灵活的中文分词算法，能够对中文文本进行智能的切分，以保留尽可能多的语义信息。适用于一般的全文搜索、文本分析和检索需求。

ik_max_word 模式：最大化分词模式，会尽可能多地将文本切分成单个词语，从而获取尽可能多的候选词。适用于对文本进行细粒度的分析和处理。

HanLP 分词器：HanLP 分词器是源于一个开源且功能强大的 HanLP 自然语言处理工具包，它由一系列模型和算法组成。目前天翼云云搜索服务中的 HanLP 分词器是基于开源 HanLP 分词器对 hanlp、hanlp_crf 等分词器进行了一定程度的优化，以适应不同的场景。

目前提供 hanlp、hanlp_crf、hanlp_nlp、hanlp_speed 等分词器，针对不同的应用场景，可以选择使用不同的分词器。

HanLP 插件包含的分词器：hanlp、hanlp_crf、hanlp_nlp、hanlp_speed、hanlp_n_short、hanlp_dijkstra、hanlp_index。

hanlp 分词器，基于词典的分词

当需要快速处理大量文本，且对分词精度要求不是特别高时，词典分词是一个很好的选择。它侧重于分词速度，能够每秒处理数千万字符，非常适合对实时性要求较高的场景；在内存资源有限的环境下，词典分词由于其较低的内存占用，也是一个理想的选择。另外 hanlp 分词器经过优化分词效果也比较不错，适合通用的场景，平衡了分词精度和分词速度。

hanlp_crf 和 hanlp_nlp 分词器，基于模型的分词器

hanlp_crf 和 hanlp_nlp 分词器在处理复杂文本结构时表现出色，能够准确识别并处理句子中的长距离依赖关系。由于其较强的泛化能力，在处理特定领域的文本时（如新闻、法律、医疗等），能够提供更全面、准确的文本分析结果。适用于对分词准确程度要求很高，不追求分词速度的场景下，且比较消耗内存资源。

hanlp_speed，极速词典分词

在需要极快速度处理文本的场景下，极速词典分词是最佳选择，它通过优化词典结构和

算法实现了超高的分词速度。在内存资源非常有限的环境下，极速词典分词由于其低内存占用特性而更具优势。

适合追求分词速度，而对精度要求不是很高的情况下。

hanlp_n_short 和 hanlp_dijkstra

构建词图并寻找最短路径的方式来实现分词，能够较好地识别并处理新词和未登录词。对于包含大量新词、缩写、专业术语等复杂文本，能够提供更准确的分词结果，效率不如词典分词，优于算法分词。

hanlp_index 和 ik_max_word 类似：会尽可能多地将文本切分成单个词语，从而获取尽可能多的候选词。适用于对文本进行细粒度的分析和处理。

中文分词增强的优势：

相比开源中文分词器，优化后的部分中文分词器在搜索结果上更具有优势；相比友商自研的中文分词器，优化后的中文分词器在搜索结果以及分词效率上更有优势。另外，天翼云云搜索服务中文分词增强模块内置了多种中文分词器，可以适应不同的场景，用户可以添加自定义词库来提高未登陆词的分词精度。

使用示例：

测试分词器分词效果

```
GET _analyze
{
    "text": "美国阿拉斯加州发生 8.0 级地震",
    "analyzer": "hanlp"
}
```

返回结果：

```
{
    "tokens": [{
        "token": "美国",
        "start_offset": 0,
        "end_offset": 2,
        "type": "nsf",
        "position": 0
```

```
}, {
    "token": "阿拉斯加州",
    "start_offset": 2,
    "end_offset": 7,
    "type": "nsf",
    "position": 1
}, {
    "token": "发生",
    "start_offset": 7,
    "end_offset": 9,
    "type": "v",
    "position": 2
}, {
    "token": "8.0",
    "start_offset": 9,
    "end_offset": 12,
    "type": "m",
    "position": 3
}, {
    "token": "级",
    "start_offset": 12,
    "end_offset": 13,
    "type": "q",
    "position": 4
}, {
    "token": "地震",
    "start_offset": 13,
```

```
        "end_offset": 15,
        "type": "n",
        "position": 5
    }]
}
```

使用其他分词器可以在 analyzer 字段指定。

创建 mappings 的时候可以在字段中指定分词器

PUT demo

```
{
    "mappings": {
        "properties": {
            "field1": {
                "type": "text",
                "analyzer": "hanlp"
            }
        }
    }
}
```

返回结果

```
{
    "acknowledged": true,
    "shards_acknowledged": true,
    "index": "demo"
}
```

常见问题

产品咨询类

云搜索服务目前支持哪些资源池？

目前上线仅支持华东 1 资源池，请关注产品动态或资源节点跟进查看最新的资源池开通情况。

云搜索服务如何保证数据和业务运行安全？

主机安全

云搜索服务提供如下安全措施：

- 通过 VPC+安全组来确保 VPC 内主机的安全。
- 通过网络访问控制列表，可以允许或拒绝进出各个子网的黑白名单管控。
- 内部安全基础设施（包括网络防火墙、入侵检测和防护系统）等。
- 数据安全
- 云搜索服务内部，通过多副本、权限管控可对索引数据进行保护隔离。
- 网络安全
- 整个网络部署分两个部分进行，两部分物理隔离，保证业务、管理各自网络安全。
- 业务部分：主要是实例的网络，支持为用户提供业务通道，对外提供数据定义、索引、搜索能力。
- 管理部分：主要是管理控制台，用于管理云搜索服务。

云搜索服务创建实例时有哪些节点存储选项？

公测期间我们支持高 I/O 和通用 SSD 规格，限制最大单盘 500GB。商用后以订购页面支持的规格为准。

云搜索服务的实例节点磁盘空间用于存放哪些文件？

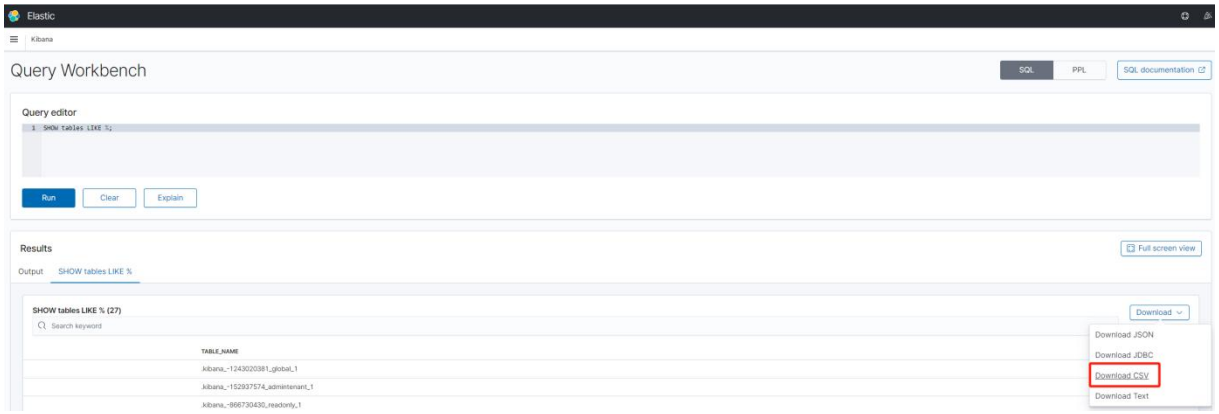
- 日志文件：Elasticsearch 日志。
- 数据文件：Elasticsearch 索引文件。
- 其他文件：集群配置文件、操作系统占用等。

云搜索服务使用的数据压缩算法是什么？

天翼云云搜索服务，除了支持搜索引擎自带的LZ4和DEFLATE算法外，还额外支持了ZSTD压缩算法。具体介绍和使用方法可参考 “增强特性” > “压缩算法” 章节。

云搜索服务中 Kibana 如何导出数据？

在天翼云云搜索服务中心，可以在 Kibana 服务中利用 Query Workbench 插件将查询到的数据进行导出，如图所示，在 Kibana 中点击左侧框的 Query Workbench，然后，查询出想要的数据，点击 Download 选择合适的文件格式进行数据导出。



计费类

云搜索服务公测期间如何计费？

当前云搜索服务一类节点仅支持包周期的计费方式。

包年/包月：根据版本/资源组的购买时长，一次性支付费用。最短时长为 1 个月，实际可订购时长以页面显示为准。

云搜索服务的计费项由节点规格费用、节点存储费用组成。Kibana/Dashboards 节点为默认开通，该节点独立收费。公测期间该部分免费。

如果需要开放节点公网访问，或使用快照、插件等能力，您另需要根据实际购买情况进行付费。

如何退订云搜索服务？

您有两种退订云搜索服务的方法：

方法一、通过订单管理退订。登录天翼云官网，进入“费用中心-订单管理-退订管理”页面，选择需要退订的资源，核对信息并逐步退订；

方法二、从云搜索服务控制台退订。登录云搜索服务控制台，进入实例管理列表页，选择需要退订的实例，点击“更多”>“退订”，后按找方法一中的步骤逐步退订。

详细方法参见云搜索服务退订介绍文档。

如何在公测期续订云搜索服务？

您有两种方式可以对公测期的实例进行续订操作。

1. 访问控制台，选择对应的云搜索服务实例，点击“更多”>“续订”，在跳转的费用中心页面完成续费操作。
2. 直接在天翼云官网“我的”>“费用中心”>“订单管理”>“续订管理”中选择订单进行续订。

需要注意的是，公测期间续订最多持续到公测截止时间，公测结束后需要进行转商后方可继续使用。

操作使用类

云搜索服务是否支持和 Logstash 对接？

支持，Logstash 支持使用 7.10.2 版本。目前云搜索服务暂未支持购买 Logstash 组件，您可通过购买云主机自建的方式实现，后续我们将持续迭代，请关注产品动态。

云搜索服务实例的管理员密码忘记了怎么办？

当您想要更换购买时设定的管理员密码，或者忘记了管理员密码时，可以进行重置。

1. 在实例列表中选择需要重置密码的实例，点击实例名称进入详情页，选择安全设置页签。
2. 在页面中的密码重置位置，点击重置密码，填入符合规则的新密码并确认输入。

注意：

1. 密码为数字、大写字母、小写字母、特殊符号（@$!%*#_~?&）的组合。
2. 长度限制为 12-26 位。
3. 不能包含账号信息、字典序及键盘序。

云搜索服务是否支持开源组件对应的 API？

天翼云云搜索产品完全兼容 Elasticsearch 和 OpenSearch 的开源 API。我们实现了与 Elasticsearch/OpenSearch 相同的接口和功能,确保用户可以使用现有的 API 和工具与我们的服务进行交互。无论是进行索引管理、查询、聚合分析还是使用全文搜索功能，用户都可以使用标准的 Elasticsearch/OpenSearch REST API 来执行这些操作。

云搜索服务支持哪些搜索功能？

云搜索服务支持的搜索功能包括强大的全文检索，高亮显示，切面搜索，近实时索引，动态聚类，丰富的文档（如 Word、PDF 等格式）处理和地理信息搜索等。

用户自建 Kibana 节点如何访问云搜索服务的 Elasticsearch 集群？

1. 首先我们需要创建一个天翼云弹性云主机（CT-ECS）。这里，需要保证一下两点：
   a. CT-ECS 和云搜索服务在同一个 VPC 下。
   b. CT-ECS 需要绑定公网弹性 IP，并且在安全组配置里，开放 5601 端口。
2. 获取云搜索服务的内网地址。选中待访问的的 Elasticsearch 实例，进入“基本信息”，可以在”实例架构图”看到数据节点对应的 IP，此 IP 列表即为 Elasticsearch 集群的 IP 地址列表。
3. 在开通的 CT-ECS 机器内搭建 Kibana 服务，并且在 config/kibana.yml 文件中进行配置修改。下面的配置文件仅作为示例参考：

```
elasticsearch.username: "***"   //用户名
elasticsearch.password: "***"   //密码
server.port: 5601
server.host: "::"
server.maxPayloadBytes: 1048576
logging.dest: {log_path}//log 文件挂载的目录
i18n.locale: zh-CN
elasticsearch.hosts: [IP1，IP2，IP3...]//Elasticsearch 集群的 IP 地址
```

云搜索服务的实例是否支持跨 VPC 的数据迁移？

默认情况下，跨 VPC 的数据，无论在 Elasticsearch 还是 OpenSearch 中均不支持直接的数据迁移。此外，需要注意下版之间的兼容性。

这里主要有以下几种方法解决：

1. 将两个 VPC 之间的网络打通，然后用户天翼云弹性云主机自建 Logstash 服务，分别于两个 VPC 进行互通，通过 Logstash 将数据进行迁移。
2. 通过天翼云对象存储 ZOS，分别将第一个 VPC 下的索引的照存储在 ZOS 上，再将另一个 VPC 下的集群挂载此 ZOS，从上面恢复索引快照。
3. 将两个 VPC 之间网络直接互通，利用 Reindex 的方式直接进行在线数迁移据。

云搜索服务的实例如何连接公网访问？

新开启的云搜索实例默认不具备公网访问能力，您需要为其绑定弹性 IP 或 IPv6 带宽，并配置安全组信息，才可使用公网访问。

约束限制

1. 开启公网访问后，会因此产生流量费用，请您提前根据自身需求，购买合适的产品，公测期该费用照常收取。
2. 配置完成后，需要前往安全组设置页面，配置公网访问白名单后，才可正常使用。
3. 如果需要使用 IPv6 访问，需要在开通虚拟私有云 VPC 时即选择开通 IPv6 能力的子网，并在下单时选择该子网，不支持实例开通后再升级 IPv6。

开通 IPv6 访问能力的实例

您需要在订购时选择具备 IPv6 的虚拟私有云，选择子网后，会提示“该子网已开通 IPv6”。并在 IPv6 访问处开启开关，如关闭，则仅可通过 IPv4 访问实例。

配置实例公网访问

您可以对已开通的实例进行公网访问的配置、修改、查看、解绑操作。

1. 录云搜索服务控制台，进入实例管理列表页，选择需要设置的实例点击名称进入详情页。

2. 在详情页面里选择“安全设置”，在弹出的页面上选择需要绑定的公网 IP 类型，如果为 IPv4，请在下拉列表中选择弹性 IP 地址；如果为 IPv6，请选择 IPv6 的带宽名称。如果绑定失败，可以等待几分钟后再次尝试重新绑定。绑定的弹性 IP 或 IPv6 带宽需要处于空闲状态。

IP 绑定过后，要补充安全组方可实现本地电脑公网访问 Kibana 或连接 Elasticsearch

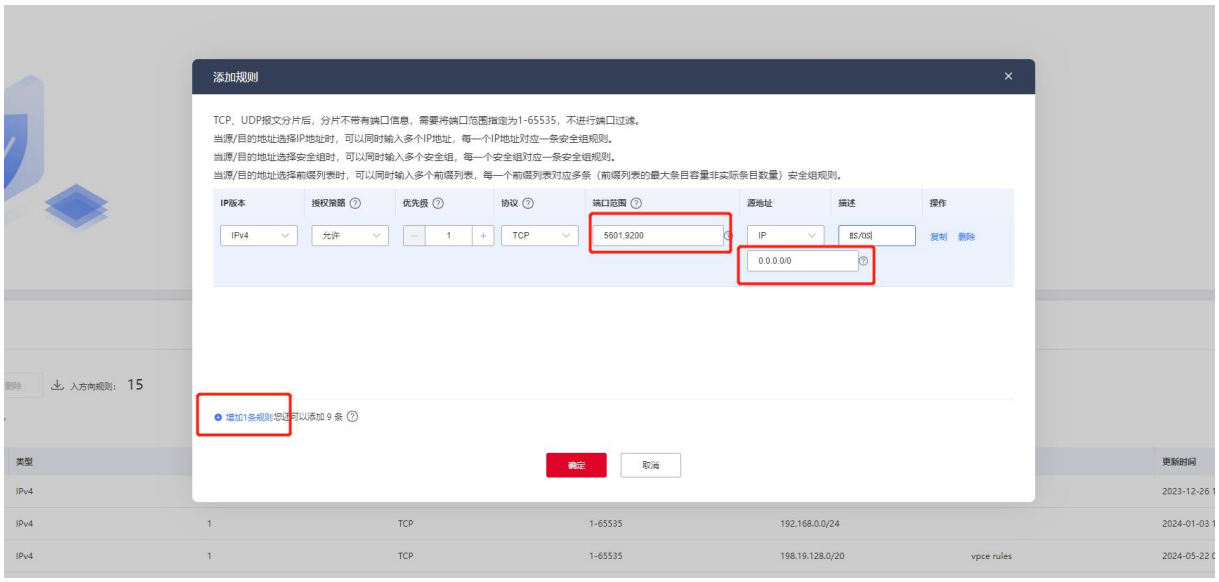
3. 修改绑定弹性 IP 或 IPv6 带宽，也需要在当前页面选择对应要修改的项目，点击

“修改公网 IP”进行重新绑定。

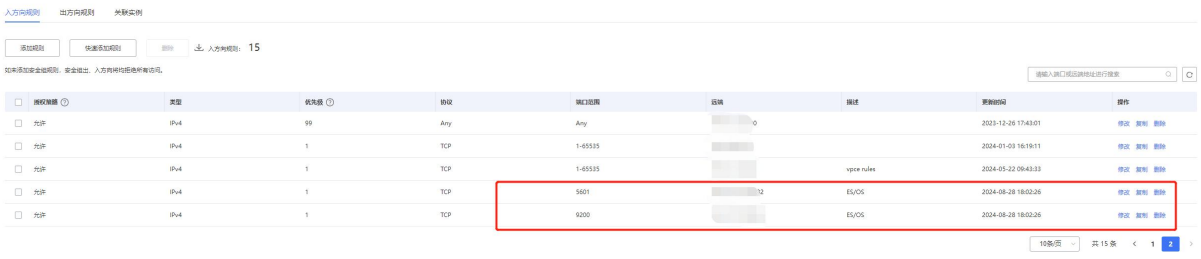
配置安全组白名单

操作步骤：

在控制台点击实例所在安全组，入方向规则点击添加规则，在弹出的填写框内的端口处填写“5601,9200”，选择需要配置的策略为 IPv4 或 IPv6，在源地址下方的 IP 地址格子中填写需要访问设备的出口公网 IP 地址，点击确定保存。



成功后会在安全组增加两条规则，此时可以通过绑定的公网 IP 地址端口访问对应对象。



通过公网 IP 地址接入实例

公网访问配置完成后，实例将会获得一个“公网访问”的 IP 地址，用户可以通过公网 IP 地址和端口接入实例。

例如，Dashboards 可直接点击页面链接进行访问。

OpenSearch 实例可以通过 Curl 命令查询索引信息

curl -u username:password -k 'https://10.62.179.32:9200/_cat/indices'其中

username 和 password 表示实例的用户名和密码。

如何查看实例的磁盘使用量和索引总量？

常见需求：

实例监控可以细粒度的集群相关的监控统计信息。但是，对于集群的整体使用情况，我们往往需要一个宏观的粗粒度统计情况。

解决方式：

在云搜索服务控制台的实例列表页，我们提供了一个宏观的整体的集群使用情况的统计

进入控制台的实例列表页，就能看到相关的集群宏观统计信息：

如图所示，我们可以看到整个集群的磁盘使用量和索引的总个数，方便我们宏观观测集群

如何在云搜索服务中配置索引的副本数量？

索引副本是原始分片（主分片）的副本，用于提供高可用性和负载均衡。适当地配置副本数量可以提升系统的容灾能力和查询性能，但过多的副本会消耗更多的存储和计算资源。

操作方法

1. 创建索引时指定副本数量： 可以在创建索引时，通过设置 number_of_replicas 参数来配置副本数量。例如：

```
PUT /my_index
{
  "settings": {
    "index": {
      "number_of_replicas": 1
```

```
    }
  }
}
```

2. 动态调整现有索引的副本数量： 对于已经创建的索引，可以随时调整副本数量。以下命令将副本数量调整为 2：

```
PUT /my_index/_settings
{
  "index": {
    "number_of_replicas": 2
  }
}
```

3. 副本数量与性能：

- 副本数为 0：无冗余，适合开发环境，但不推荐在生产环境中使用。
- 副本数为 1 或更多：能够提升查询性能，同时增加数据冗余，建议生产环境至少设置 1 个副本。

如何查看云搜索服务搜索引擎中索引的分片数和副本数？

解决方案：

可以通过以下几种方式查看索引的分片和副本数：

1. 查看索引的设置： 使用以下命令可以查看指定索引的详细设置，包括分片数和副本数：

```
GET /my_index/_settings
```

2. 查看所有索引的分片和副本信息： 可以查询集群中所有索引的分片和副本数量：

```
GET _cat/indices?v&h=index,pri,rep
```

其中，pri 表示主分片数量，rep 表示副本数量。

问题排查类

云搜索实例开通及访问类

实例开通失败怎么处理？

如果遇到云搜索实例开通失败，可能因为：

1、网络安全组设置出现变动。为保证实例的开通顺畅，我们会默认为您配置如下安全组规则：

规则 1（入方向）：允许远端 198.19.128.0/20 以 TCP 协议访问端口 1-65535 ，规则优先级为 1。

规则 2（入方向）：允许远端 192.168.0.0/24(实际网段地址，以客户创建的 VPC 子网网段地址为准)以 TCP 协议访问端口 1-65535，规则优先级为 1。

规则 3（出方向）：将允许出方向所有访问，规则优先级为 100。此操作有风险，建议用户按需配置限制规则。

如您在开通过程中，对安全组规则进行了修改，则可能会导致无法自动部署，请先前往安全组配置控制台进行检查。

2、资源未能成功开通，如资源池剩余资源不足，资源接口临时故障等；

3、资源开通成功但自动部署过程中出现后端服务问题。

针对 2、3 两种情况，建议您可以先重试开通，如果重试依然失败，可通过工单联系工程师处理。

实例连接不上如何处理？

当您遇到实例无法连接的情况时，请按照以下步骤进行排查和解决：

1. 检查网络配置

确保您的实例所在的子网和安全组配置正确，允许入站和出站的网络流量。检查安全组规则，确保允许来自客户端的 IP 地址的入站访问。

2. 确认服务状态

从控制台查看服务状态，检查搜索引擎实例的运行状态是否正常。

3. 验证实例端口

确认实例的开放端口（默认情况下为 9200）在安全组中被允许访问。如果需要连接到其他端口（例如 Elasticsearch 集群的 9300 端口），请确认该端口也被允许。

4. 检查客户端配置

确保客户端的配置正确，特别是实例的 URL、端口号以及认证信息（用户名、密码或 API 密钥）是否填写正确。

如果使用了 HTTPS 连接，确认 SSL 证书配置正确，并且客户端信任该证书。

5. 联系天翼云技术支持

如果经过上述排查步骤仍无法解决问题，请联系天翼云的技术支持团队，提供相关的错误信息、日志和配置截图，以便更快地诊断和解决问题。

插件安装不成功

插件安装不成功可以从以下几个方向排查：

1. 插件准备：

   插件文件本身符合 Elasticsearch 或 OpenSearch 的编写规范；

   插件是否适配当前的 Elasticsearch 或 OpenSearch 版本；

   插件包上传到天翼云的对象存储中并开启可读权限；

   插件不在该实例默认插件清单中或是已经安装过的插件。

2. 重启过程：

   如遇安装过程完成，未能成功重启，可在实例列表页尝试再次重启，如尝试不成功，可提报工单联系工程师处理。

云搜索实例使用类

云搜索服务单个集群中分片过多有什么影响？

原因分析：

分片是数据在集群中的分布单元，适当的分片数量可以提高数据分布的灵活性和并行处理能力，但分片过多会导致以下问题：

资源开销增加：每个分片都会消耗一定的系统资源（CPU、内存、文件句柄）。过多的分片会增加系统开销，导致集群性能下降。

管理复杂性增加：更多的分片意味着更复杂的分片管理，包括分片的分配、迁移和恢复都会变得更为复杂。

搜索性能降低：虽然分片可以并行处理查询，但过多的小分片可能导致每个分片包含的数据太少，反而增加了查询开销，影响性能。

解决方案：

1. 控制分片数量：建议一个分片大小控制在 10GB 至 50GB 之间。对于较大的数据集，可以适当增加分片数量，但应避免每个节点上存在过多分片。
2. 合并小分片：如果存在过多小分片，可以使用_shrink API 来合并分片。例如，将一个包含 10 个小分片的索引缩减为 5 个分片：

```
POST /my_index/_shrink/my_new_index
```

3. 定期评估分片设置：监控分片数量，确保每个节点上的分片数量合理（通常不应超过 1000 个）。

为什么云搜索服务中的搜索引擎状态变为黄色（Yellow）？

原因分析：

1. 分片副本未分配：集群状态为黄色通常表示主分片已分配，但副本分片未能成功分配。这可能是因为集群中节点数量不足，无法容纳副本分片。
2. 节点资源不足：即使集群中有足够的节点，如果某些节点资源（如磁盘空间、内存等）不足，副本分片也可能无法分配，导致集群状态为黄色。
3. 节点故障：如果某个节点宕机或网络不稳定，集群可能无法将副本分片分配到该节点上，从而使集群状态变为黄色。

解决方案：

1. 扩展节点数量：如果集群中的数据节点数量不足，可以通过添加新节点来解决。例如，如果当前集群只有 1 个数据节点，可以添加更多节点以容纳副本分片。
2. 检查节点资源：确保所有节点有足够的磁盘空间、内存和 CPU 资源。如果某些节点资源不足，可以进行扩容，或手动将副本分片分配到资源充足的节点上。
3. 手动分配副本分片：如果是由于网络或节点问题导致副本分片未分配，可以使用 _cluster/reroute 命令手动迁移分片，强制将副本分片分配到其他可用节点。

为什么云搜索服务中的搜索引擎频繁出现分片不均衡的现象？

原因分析：

1. 节点资源差异：如果集群中的节点资源不均衡（如 CPU 或存储容量不同），分片可能会更倾向于分配到资源较强的节点上，导致分片不均衡。

2. 分片数量变化频繁：在高频索引创建或删除的环境下，分片的分配和回收会不断变动，可能导致短时间内出现分片不均衡的情况。
3. 手动分配分片：通过手动设置分片分配策略，可能导致某些节点上的分片数量过多，而其他节点上的分片较少。
4. 缺乏自动负载平衡：如果集群未启用自动重新分配分片的策略，当集群中节点出现负载不均时，系统不会自动调整分片位置。

解决方案：

1. 平衡节点资源：确保集群中的所有节点资源尽量一致，包括 CPU、内存、存储和 I/O 性能，以避免因为资源差异导致的分片不均衡。
2. 优化索引和删除操作：减少频繁的索引创建和删除操作，合理规划索引生命周期，减少分片频繁变动的情况。
3. 自动负载均衡：启用自动分片重分配功能，确保集群在检测到节点间分片不均衡时可以自动调整。可以使用以下命令开启：

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.rebalance.enable": "all"
  }
}
```

这将允许集群自动对所有类型的分片进行重新平衡。

如何清理云搜索服务中搜索引擎的索引数据？

原因分析：

当索引中的数据不再需要时，清理过期或无用的数据可以释放存储空间，优化集群性能。常见的清理方式包括删除整个索引或根据特定条件删除部分文档。

解决方案：

1. 删除整个索引： 如果整个索引不再需要，可以直接删除索引：

```
DELETE /my_index
```

2. 删除部分文档： 如果只需删除符合条件的部分数据，可以使用_delete_by_query

API。例如，删除满足条件的文档：

```
POST /my_index/_delete_by_query
{
  "query": {
    "range": {
      "timestamp": {
        "lt": "now-30d"
      }
    }
  }
}
```

上述命令将删除 30 天前的所有文档。

3. 自动化清理（索引状态管理 ISM）： 为了定期自动删除过期数据，可以使用索引状态管理（ISM），自动定义索引的生命周期，包括删除阶段。例如：

```
PUT _opendistro/_ism/policies/my_policy
{
  "policy": {
    "schema_version": 1,
    "default_state": "active",
    "states": [
      {
        "name": "active",
        "actions": [],
        "transitions": [
          {
            "state_name": "delete",
            "conditions": {
              "min_index_age": "30d"
            }
```

```
        }
      ]
    },
    {
      "name": "delete",
      "actions": [
        {
          "delete": {}
        }
      ],
      "transitions": []
    }
  ]
 }
}
```

这将自动在 30 天后删除索引。

为什么云搜索服务中的索引写入速度突然下降？

原因分析：

1. 写入冲突：当多个客户端同时向同一个索引写入数据时，可能会发生写入冲突，导致部分写入操作被推迟或重试，从而降低写入速度。
2. 磁盘 I/O 限制：写入操作需要频繁访问磁盘。如果磁盘 I/O 性能不佳或资源被其他任务占用，写入速度会受到影响。
3. 缓冲区溢出：云搜索服务在写入数据时会使用内存缓冲区。如果缓冲区满了，系统会强制刷新到磁盘，这个过程可能会拖慢写入速度。
4. 垃圾回收（GC）问题：如果节点的 JVM 频繁进行垃圾回收，特别是 Full GC，系统性能会受到影响，导致写入速度下降。

解决方案：

1. 优化写入并发：避免高并发写入到同一索引，可以通过拆分索引或批量写入方式减少冲突。调整客户端的并发写入线程数和批量写入大小。
2. 提升磁盘性能：使用更高性能的磁盘设备（如 SSD），确保磁盘 I/O 不是瓶颈。检查系统中是否有其他进程占用了磁盘资源，影响了写入速度。
3. 调整刷新间隔：可以通过增加刷新间隔来减少缓冲区强制刷新到磁盘的频率，例如：

```
PUT INDEX_NAME/_settings
{
  "index.refresh_interval": "30s"
}
```

这将延长刷新时间，允许更多的数据在内存中积累，从而减少写入延迟。

4. 优化垃圾回收设置：监控 JVM 的垃圾回收行为，必要时升级到 G1 GC 或调整堆内存大小，减少 GC 对性能的影响。

为什么云搜索服务中的搜索结果不准确或不完整？

原因分析：

1. 索引未及时刷新：当数据被写入索引后，可能需要一段时间才能被搜索引擎查询到。如果未及时刷新，搜索结果可能不包含最新的数据。
2. 分片不均衡导致的查询瓶颈：如果某些节点上的分片过多，查询请求集中在这些节点上，可能会导致部分分片未能及时返回结果，从而导致查询结果不完整。
3. 文档丢失或损坏：在写入操作中，系统崩溃或网络中断可能导致部分文档未正确写入或损坏，导致搜索结果不准确。
4. 查询不优化：搜索查询使用了不正确或未优化的查询语法，导致查询条件不准确，进而影响搜索结果的完整性。

解决方案：

1. 手动刷新索引：如果需要实时获取数据，可以手动刷新索引，以确保最新数据可供查询。例如：

```
POST INDEX_NAME/_refresh
```

2. 重新平衡分片：检查分片分布是否均衡，必要时手动迁移分片，确保所有节点的负载均衡，以提高查询效率和结果准确性。

3. 恢复或重建索引：如果确认索引中有丢失或损坏的文档，可以通过备份恢复索引或重新创建索引，确保数据完整性。
4. 优化查询：检查查询语法和条件，确保查询逻辑正确。如果查询结果仍不符合预期，尝试使用不同的查询方式（如 bool 查询、短语查询）来提升准确性。

如何清理云搜索服务搜索引擎的缓存？

原因分析：

缓存用于加速查询操作，云搜索服务会将常见的查询结果或热数据缓存在内存中，但在某些情况下，缓存可能需要手动清理，例如缓存命中率下降或内存占用过高时。

解决方案：

1. 清理字段数据缓存： 字段数据缓存用于存储字段值，可以通过以下命令清理字段缓存：

```
POST /_cache/clear?fielddata=true
```

2. 清理查询缓存： 查询缓存用于缓存查询结果，可以通过以下命令清理查询缓存：

```
POST /_cache/clear?query=true
```

3. 清理整个缓存： 如果希望清理所有缓存，包括字段数据缓存、查询缓存和请求缓存，可以使用：

```
POST /_cache/clear
```

4. 监控缓存使用： 定期监控缓存使用情况，避免缓存过度占用内存。例如，使用以下命令查看缓存统计信息：

```
GET /_nodes/stats/indices/fielddata?human
```

为什么云搜索服务中的磁盘空间使用增长异常快？

原因分析：

1. 未及时删除过期数据：某些索引可能存储了过期数据，但没有定期清理。这通常出现在日志类索引中，如果没有设置索引状态管理（ISM），大量过期数据会长期占用磁盘空间。
2. 索引分片过多：如果集群中索引分片过多，且每个分片包含的实际数据量较少，系统会占用更多的磁盘空间来维护这些分片的元数据和存储结构。
3. 重复数据：索引中的数据存在重复存储的现象，比如数据冗余或未优化的文档结构，

这会导致磁盘使用量激增。

解决方案：

启用索引状态管理（ISM）：为日志或其他过期数据的索引设置生命周期策略，自动删除过期的索引。例如：

```
PUT _opendistro/_ism/policies/my_policy
{
  "policy": {
    "schema_version": 1,
    "default_state": "active",
    "states": [
      {
        "name": "active",
        "actions": [],
        "transitions": [
          {
            "state_name": "delete",
            "conditions": {
              "min_index_age": "30d"
            }
          }
        ]
      },
      {
        "name": "delete",
        "actions": [
          {
            "delete": {}
          }
        ],
        "transitions": []
```

```
      }
    ]
  }
}
```

这将自动在 30 天后删除旧索引。

2. 减少分片数量：避免创建过多分片，推荐每个分片的大小在 10GB-50GB 之间。如果存在很多小分片，可以通过合并索引来优化磁盘使用。
3. 数据去重与优化：检查索引的数据结构，减少重复存储。可以通过优化文档设计或启用压缩（如_source 字段压缩）来节省磁盘空间。

为什么云搜索服务中的索引恢复速度较慢？

原因分析：

1. 分片大小过大：索引分片的大小过大会导致恢复速度变慢，特别是在涉及大量数据时。数据量大意味着从存储中读取分片和将其分配到节点上的过程会花费更多时间。
2. 节点资源不足：如果集群中的节点资源（如 CPU、内存或 I/O 性能）不足，分片恢复过程可能会被资源瓶颈限制，导致恢复速度变慢。
3. 网络延迟：在集群中进行分片恢复时，如果节点之间的网络延迟较高，数据传输速度会降低，进而影响恢复时间。
4. 并发恢复限制：默认情况下，OpenSearch/Elasticsearch 会限制一次可以同时恢复的分片数量，以避免集群过载。如果这个数量设置得过低，恢复时间会因此延长。

解决方案：

1. 合理设置分片大小：避免分片过大，建议单个分片的大小控制在 50GB 以下。如果索引的数据量较大，可以通过增加分片数量来分散负载，提升恢复效率。
2. 扩展集群资源：根据需求扩展节点的 CPU 和内存资源，并确保存储性能足够支撑恢复操作。
3. 优化网络配置：在分布式集群环境中，确保节点间的网络延迟尽可能低，必要时可以使用高速网络设备或调整网络配置以优化数据传输性能。
4. 提高并发恢复数：可以调整并发恢复分片的数量，以加快恢复速度。相关配置参数为 cluster.routing.allocation.node_concurrent_recoveries，默认值通常为 2，可以适当增加该值，例如：

```
PUT _cluster/settings
```

```
{
  "persistent": {
    "cluster.routing.allocation.node_concurrent_recoveries": 5
  }
}
```

实例可观测性及运维

实例磁盘使用率过高的影响是什么？

观测现象：

通过天翼云搜索控制台的实例监控可以发现，当磁盘使用率过高时，可能会对于业务产生显著的影响。

解决方案：

默认情况下，搜索集群会对磁盘使用率进行警戒水位线管理：

1. 低警戒水位线管理：当磁盘使用率达到 85%时，无法在此节点分配新的分片。
2. 高警戒水位线管理：当磁盘使用率达到 90%时，此时，集群尝试对节点的分片进行重新分配，此时将对集群内所有节点的分片进行影响。
3. 洪泛警戒水位线管理：当磁盘使用率达到%，此时，将对集群的索引开启强制只读模式（index.blocks.read_only_allow_delete），会严重影响业务的写入，防止资源耗尽引发的集群服务宕机。

因此，当磁盘使用率过高时，应该重点注意集群的性能，及时扩容，避免对业务进行影响。此外，也可以及时删除无用的索引，释放磁盘空间。此时，如果依旧是 readonly，需要执行以下操作，恢复集群的写权限

```
PUT _settings
{
   "index.blocks.read_only_allow_delete": null
}
```

实例内存使用率过高的影响是什么？

观测现象：

我们通过观察天翼云云搜索实例中的实例监控，可以看到节点的内存使用率和 JVM 内存

使用率等内存监控指标，当这些指标较高时，可能会对于集群的性能有明显影响。

问题解决：

需要明确的是，在 Elasticsearch 集群中，根据设置，我们往往会分配机器内存一半的量来分配给 JVM，以供给 Elasticsearch 服务使用。剩下的内存，绝大部分被分配给了 Lucene 用来支持索引的底层服务。因此系统的总内存使用率往往处于高位，这个是常见的现象。

但是长期的内存高使用率，不仅有可能诱发 OOM 故障，也对于大批量写入和查询有性能影响，我们建议，当内存使用率长期处于高位的时候，应该密切观察内存相关指标。最好通过水平扩容或者垂直扩容来提升集群的规格，避免业务受损。

最佳实践

迁移集群

使用自建 Logstash 迁移 Elasticsearch 实例间数据

Logstash 是一个开源的数据处理管道工具，广泛用于数据收集、处理和传输。它通常作为“ELK Stack”（Elasticsearch、Logstash、Kibana、Beats）的一个核心组件，用于处理结构化和非结构化数据。

自建 Logstash 可以实现将源 Elasticsearch 实例（如天翼云、自建或第三方 Elasticsearch 实例）中的数据迁移至天翼云 Elasticsearch 实例。在升级实例版本、实例架构调整、或跨区域的实例数据迁移时，可以选择使用 Logstash 迁移源 Elasticsearch 实例数据。(推荐使用 Logstash 7.10.2 版本)。

Logstash 的方式迁移数据支持跨大版本，且迁移方式灵活，下表是支持的集群版本：

源\目标	Elasticsearch 7.10.2	OpenSearch 2.9.0
Elasticsearch 6.x	√	√
Elasticsearch 7.x 小于 7.10.2	√	√
Elasticsearch 7.x	√	√

大于 7.10.2		
Elasticsearch 8.x	√	√

本文以自建 Elasticsearch 7.10.2 版本迁移至天翼云 Elasticsearch 实例为例子。

1. 前提条件。

已经创建天翼云 Elasticsearch 实例。

已经申请天翼云云服务器且自行部署 Logstash 或者在其他服务器部署 Logstash（Logstash OSS 版本推荐 7.10.2 版本）

自建 Logstash 能够访问目的 Elasticsearch 实例以及需要迁移的源 Elasticsearch 实例。

2. Logstash 工作模型。

Logstash 工作模型核心部分为三部分：输入（Input）、过滤器（Filter）、输出（Output），按照配置管道文件的顺序对数据进行提取、处理转换、输出。

a.输入（Input）

Logstash 支持多种数据输入源，如文件、数据库、消息队列以及 Elasticsearch 等。在我们的场景中，源 Elasticsearch 实例就是输入数据源。Logstash 会批量提取源 Elasticsearch 实例中的数据。

b.过滤器（Filter）

过滤器是可选的，用于对输入数据进行实时处理和转换。它提供了一些强大的插件，可以对数据进行解析、变换、裁剪或其他操作。在我们的场景中，可以选择是否使用过滤器来处理迁移中的数据。

例如删除源数据中不需要迁移的字段等操作。

c.输出（Output）

Logstash 的输出插件负责将处理后的数据写入到目标位置，这可以是文件、数据库、消息队列，或者像本例中的天翼云 Elasticsearch 实例。

3. 迁移必备的信息。

源实例（天翼云、自建或第三方 Elasticsearch 实例）访问地址、用户名以及密码。

目的实例（天翼云 Elasticsearch 实例）访问地址、用户名以及密码。

提前在目的实例创建好源实例中的待迁移索引的索引结构。

4. 测试 Elasticsearch 实例服务正常。

curl http://{ip}:{port}

分别将 ip 和 port 替换为源实例以及目的实例的实际 ip 地址和端口号。

5. 使用自建 Logstash 全量迁移数据。

例如在 Logstash 目录下创建一个管道文件，es-es.conf

写入下面的配置

```
input{
     elasticsearch{
        # 源 Elasticsearch 实例的访问地址
        hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}",
"http://{ip}:{port}"]
        # 访问源 Elasticsearch 实例的用户名和密码，如无安全机制可不配置
        user => "*********"
        password => "********"
        # 配置源实例中待迁移的索引，可以使用通配符
        index => "index1, index2"
        # 查询 Elasticsearch 实例包含元数据
        docinfo => true
        # 使用多个切片提高吞吐量，合理值的范围从 2 到大约 8，一般不要超过索引
分片数
        slices => 2
        # Logstahs 每次查询 Elasticsearch 实例返回的最大数据条数
        size => 1000
     }
 }
 # 在此对数据进行处理
 filter {
   mutate {
     # 移除 logstash 增加的字段
```

```
    remove_field => ["@metadata", "@version"]
  }
 }
 output{
    elasticsearch{
      # 目标 Elasticsearch 实例的访问地址
      hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}",
"http://{ip}:{port}"]
      # 访问目标 Elasticsearch 实例的用户名和密码，如无安全机制可不配置
      user => "********"
      password => "********"
      # 配置目标实例的索引，如下配置时和源实例保持一致
      index => "%{[@metadata][_index]}"
      document_id => "%{[@metadata][_id]}"
    }
 }
```

使用下面的命令检查 Logstash 管道文件的配置是否正确。

./bin/logstash -tf es-es.conf

```
2024-10-11T17:52:37,526 | INFO | logstash.runner | Starting Logstash {"logstash.version"=>"7.10.
2", "jruby.version"=>"jruby 9.2.13.0 (2.5.7) 2020-08-03 9a89c94bcc OpenJDK 64-Bit Server VM 25.3
52-b11 on 1.8.0_352-b11 +indy +jit [linux-x86_64]"}
2024-10-11T17:52:38,126 | WARN | logstash.config.source.multilocal | Ignoring the 'pipelines.yml
' file because modules or command line options are specified
2024-10-11T17:52:39,680 | INFO | org.reflections.Reflections | Reflections took 66 ms to scan 1
urls, producing 23 keys and 47 values
Configuration OK
2024-10-11T17:52:40,393 | INFO | logstash.runner | Using config.test_and_exit mode. Config Valid
ation Result: OK. Exiting Logstash
```

Result：OK 即检查通过，语法没有问题。检查通过后使用下面的命令启动 Logstash 管道。

./bin/logstash -f es-es.conf

完成迁移后对比迁移前后索引结构、索引中数据条数、索引存储等指标，保证数据全部迁移完毕。

6. 使用自建 Logstash 增量迁移数据。

增量迁移数据和全量迁移数据类似，区别在于增量迁移需要待迁移的索引中有增量字段。

增量迁移数据的 Logstash 管道配置文件和全量迁移数据不同在于需要配置具体的’query’。例如，有一个索引中有一个时间字段’created_at’，类型为’date’，可能的值例如，"2024-10-11T12:34:56Z"。

那么需要添加如下配置，’query’参数的内容是 Elasticsearch 实例查询的语法，例如，

```
query =>
'{"query":{"bool":{"should":[{"range":{"created_at":{"from":"2024-10-11T12:34:56Z"}}}]}}}'
```

需要根据具体索引结构来修改。

整个管道文件完成的配置如下：

```
input{
    elasticsearch{
        # 源 Elasticsearch 实例的访问地址
        hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}", "http://{ip}:{port}"]
        # 访问源 Elasticsearch 实例的用户名和密码，如无安全机制可不配置
        user => "*********"
        password => "********"
        # 配置源实例中待迁移的索引，可以使用通配符
        index => "index1, index2"
        query =>
'{"query":{"bool":{"should":[{"range":{"created_at":{"from":"2024-10-11T12:34:56Z"}}}]}}}'
        # 查询 Elasticsearch 实例包含元数据
        docinfo => true
        # 使用多个切片提高吞吐量，合理值的范围从 2 到大约 8，一般不要超过索引分片数
        slices => 2
        # Logstahs 每次查询 Elasticsearch 实例返回的最大数据条数
        size => 1000
    }
```

```
}

# 在此对数据进行处理
filter {
  mutate {
    # 移除 logstash 增加的字段
    remove_field => ["@metadata", "@version"]
  }
}
output{
    elasticsearch{
      # 目标 Elasticsearch 实例的访问地址
      hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}",
"http://{ip}:{port}"]
      # 访问目标 Elasticsearch 实例的用户名和密码，如无安全机制可不配置
      user => "********"
      password => "********"
      # 配置目标实例的索引，如下配置时和源实例保持一致
      index => "%{[@metadata][_index]}"
      document_id => "%{[@metadata][_id]}"
    }
}
```

检查并启动 Logstash 即可完成增量迁移。

使用 Reindex

通过搜索引擎内部支持的 Reindex 指令进行数据迁移，也是一种常见的云搜索数据迁移场景。

Reindex 方式适用场景：

源云搜索实例和目标搜索实例网络互通。

无需引入额外外部工具，仅仅依靠 API 即可实现。

对迁移速度没有过高要求。

可以按条件筛选进行数据迁移，查询筛选语句、painless 脚本全支持。

适配性

Elasticsearch 版本间，除了 Elasticsearch8.X 向 Elasticsearch 7.X 迁移，其余均支持。

Elasticsearch 数据往 OpenSearch2.9.版本迁移，全部支持。

待迁移集群版本	ElasticSearch7.10.2	OpenSearch2.9.0
Elasticsearch 6.x	√	√
Elasticsearch 7.x	√	√
Elasticsearch 8.x	×	√

使用方式：如您需要使用该方式进行实例迁移，请在官网提报工单，由工程师为您进行相关配置处理。

示例说明：

我们以数据从 ElasticSearch 7.10.2 往 OpenSearch 2.9.0 迁移为例，将 geonames 索引迁移：

首先，我们会在目标实例的 OpenSearch 的 config/opensearch.yml 里添加远端的 IP 白名单配置：

```
reindex.remote.whitelist: ["IP_source:9200"]
```

然后，我们在目标实例创建 index（如果没有额外的分片和 mapping 的设置，可以跳过）之后，在目标实例上进行 Reindex 操作

```
POST /_reindex?wait_for_completion=false
{
  "source": {
    "remote": {
      "host": "http://IP_source:9200",
      "username": "{username}",
      "password": "{password}"
```

```
    },
    "index": "geonames",
    "size": 10000
  },
  "dest": {
    "index": "geonames"
  }
}
```

Reindex 性能调优

1. 增加 batch_size，默认 1000，一般一个 batch 在 5M-15M 数据时，性能比较好。根据文档特性，合理修改 batch_size。

2. 底层调用 scroll。slices 大小的设置可以手动指定，或者设置 slices 为 auto，auto 的含义是：针对单索引，slices 大小=分片数；针对多索引，slices=分片的最小值。

3. 可以先设置 replica 为 0，后续再修改 setting。

4. 大量写入情况下，先禁止 refresh。设置 refresh{ "refresh_interval": -1 }，迁移完成后，再打开。

优化集群性能

优化写入性能：

优化 Elasticsearch 集群的写入性能是确保数据高效、快速存储的关键。以下是一些方法和最佳实践，可以帮助提高 Elasticsearch 集群的写入性能。

合适的副本数：默认情况下，Elasticsearch 索引有 1 个副本。为了提高写入性能，可以减少副本数，因为每个副本会占用额外的写入资源。然而，减少副本数会降低数据的高可用性，需要在性能和可用性之间进行权衡。

分片数量：索引的分片数量会影响写入性能。一般来说，更多的分片可以提高并行写入的性能，但过多的分片也会导致资源浪费。建议基于数据规模合理设置分片数量，并根据实际情况进行调整。

刷新间隔（refresh interval）：默认的刷新间隔为 1 秒，可以通过增加刷新间隔来减

少 Elasticsearch 对磁盘的频繁写入，从而提高写入性能。将 index.refresh_interval 设置为较长的时间，例如 30s 或 60s，但要注意这会延迟数据的可见性。

合并策略：使用 index.merge.scheduler.max_thread_count 参数来控制合并的线程数，合理配置可以减轻写入时的磁盘 I/O 压力。

批量写入（Bulk API）：使用 Bulk API 可以将多个文档的写入请求批量处理，减少网络和资源开销。推荐将每批次的文档数量控制在合理范围（如 500-1000 个文档），以平衡单次请求的大小和系统的稳定性。

避免嵌套文档和父子关系：如果可能，尽量避免使用嵌套文档和父子关系，因为这些操作会增加写入的复杂度和资源消耗。

优化查询性能：

文档结构设计:避免使用过多的嵌套结构和过深的嵌套字段。嵌套文档虽然可以满足复杂的数据结构需求，但会显著增加查询的复杂度和时间。

尽量使用扁平化的数据模型，这样可以减少在查询时的计算和数据加载时间。

合适的字段类型:根据需要选择正确的字段类型。例如，数值类型字段（integer、float 等）比字符串类型字段更容易索引和查询，查询速度也更快。

对于大文本字段，考虑使用 text 类型，并结合 keyword 字段来处理精确匹配的需求。

优化映射:禁用不需要的字段索引，例如，通过将不需要搜索的字段设置为 index: false，可以减少索引的大小和查询时的开销。

合理使用 doc_values，将其关闭以减少内存使用，但在需要进行排序或聚合的字段上仍然保持开启。

查询合并与简化:尽量合并多个查询条件，避免过多的布尔查询（bool query）和过滤器（filter）。这不仅可以减少查询的复杂度，还能降低查询的时间开销。

在查询中使用 filter 而非 query 来过滤不影响得分计算的条件，因为 filter 查询不会计算相关性得分，性能更高。

缓存利用:利用 Elasticsearch 的缓存机制，例如 request cache 和 filter cache，对经常使用的查询进行缓存，以减少查询响应时间。

对于相同或类似的查询，使用_cache 选项启用缓存，例如在 bool 查询的 filter 部分。

分页优化:使用 search_after 而非 from+size 进行深分页。from+size 在深度分页时的性能较差，因为它需要跳过大量数据。

使用 scroll API 来处理需要大量数据返回的场景，如全量导出，但要注意 scroll 在返回大量数据时的性能开销。

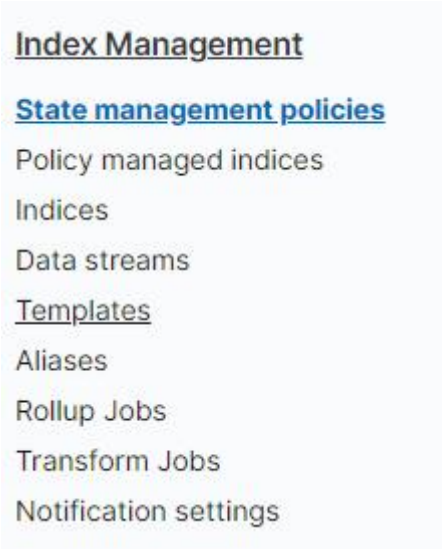
聚合优化:仅在必要时使用聚合查询,因为聚合查询通常计算开销较大。合理使用 bucket 和 metric 聚合，避免不必要的聚合操作。

使用 composite 聚合替代 terms 聚合处理大量唯一值时的场景，composite 聚合可以分步处理并返回更稳定的结果。

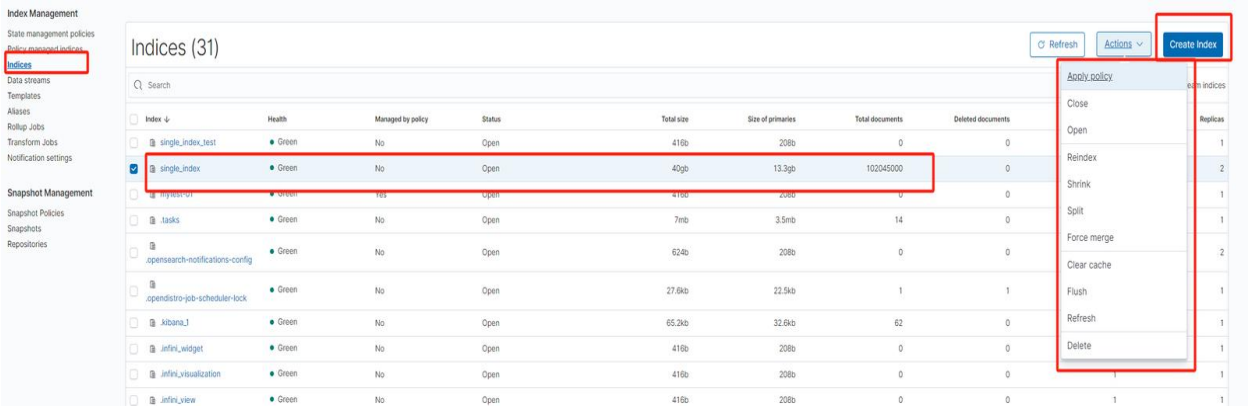
管理索引

天翼云云搜索提供了高级的索引管理功能，通过该插件可以可视化查看并管理索引、创建索引策略、创建 roll-up 作业，通过 Kibana 和 Opensearch-Dashboards 可视化对索引进行管理。

下面，我们以 Opensearch-Dashboards 的索引管理界面来进行演示，首先，是进入索引管理插件页面：



1. 索引相关信息的查看和基本操作:

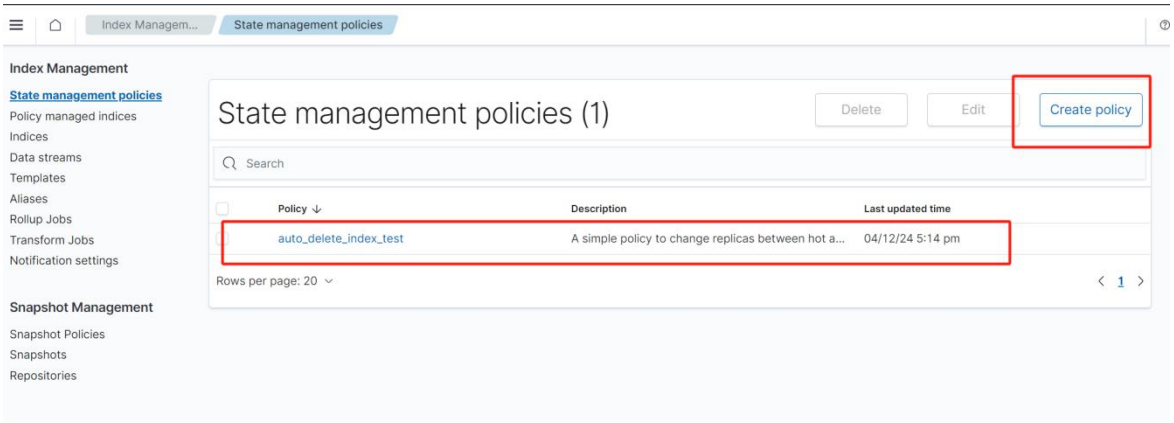


如图所示，支持以下几种管理方式：

- 显示索引相关的信息
- 还提供了一些界面化的高阶操作，比如 reindex clear cache delete 等指令，基本就是替代了直接查看的一些指令。
- 还可以直接创建索引，常规操作
- 可以查看索引是否被某些策略控制

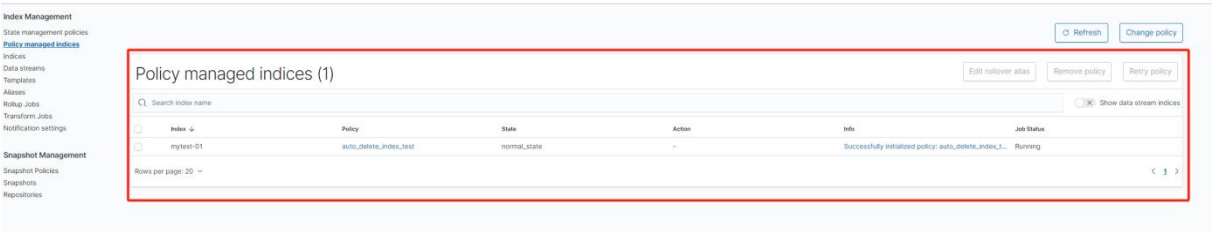
2. 可以使用索引策略来对索引进行管理。

下面我们以自动清理索引策略为例子，进行演示。

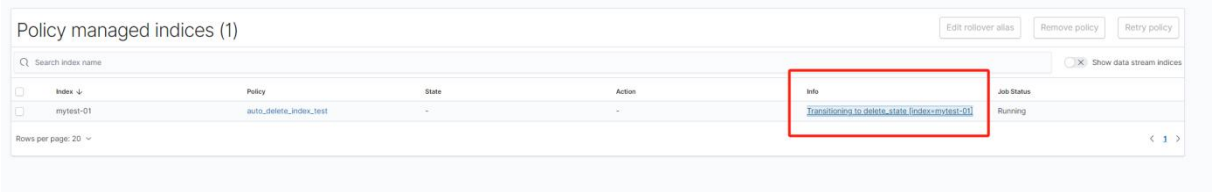


如图，我们创建了一个索引策略叫 auto_delete_index_test，其中索引策略的配置和解释如下：

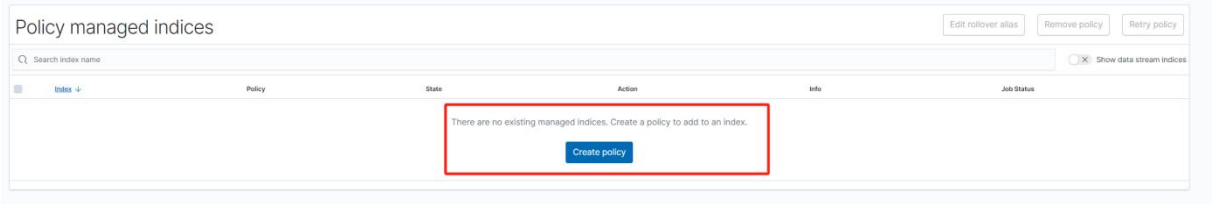
然后，我们创建一个 mytest-01 索引，格式符合索引模板的名称。创建好后，我们看到，该索引被索引策略所纳管。



当一段时间过后，索引符合了被删除的策略，状态变为了 to_delete_state 状态，如图所示：



最终，后台的策略执行完后，我们再次查看，索引最终被删除了。

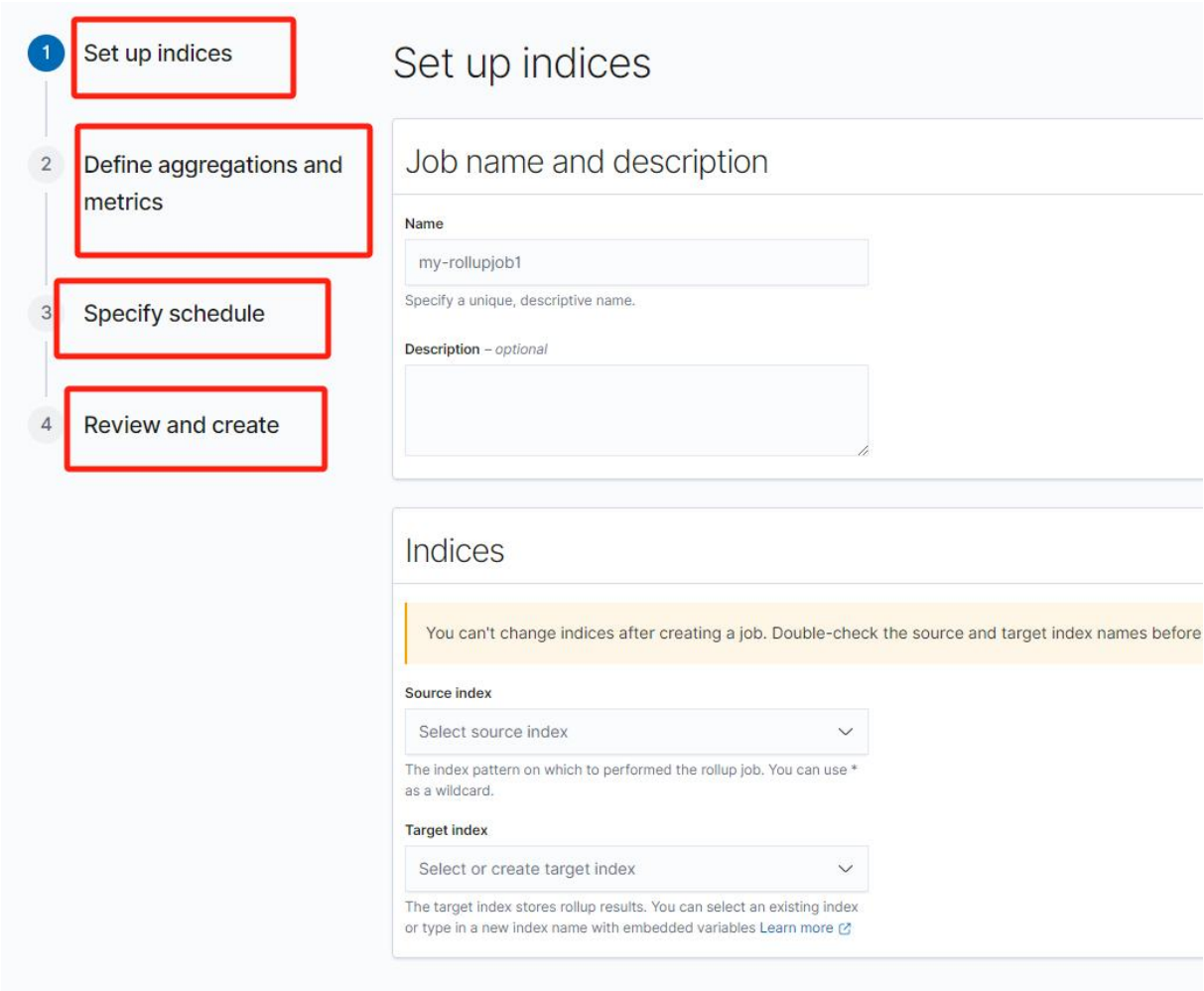


3. rollup 作业管理

Rollup Job 是一个用于汇总和聚合索引中数据的任务。它可以在原始索引上执行一系列聚合操作，并将结果存储在新的汇总索引中。这种汇总和聚合可以大大降低数据量，并提高查询性能。

在某些场景下，比如流量、点击、访问等等采集的很细粒度的日志数据，近几小时、几天的数据，有一定的参考价值。非常久远的数据，可以直接归档删了。但是，对于比如 7-90 天的数据，如果仍然很细粒度保存，那么价值也不是特别大。因此，将细粒度的数据进行 rollup 处理，然后再删除原始索引，对于使用和查询上基本无任何影响，但是可以极大地降低存储空间。

多数场景下，创建合适的 rollup job 可以减少 90%存储。



在索引管理界面，执行如下操作：

1. 创建一个 rollup job，并且选择适用哪些索引，并且要重新生成新的索引；
2. 定义聚合规则---比如按照 1h 的维度，去 sum/avg/max/min 某些特定的指标；
3. 指定 job Scheduler；
4. 后台系统运行 rollup jobs。

创建好后，系统会在后台自动执行 rollup job。

实践案例

使用 OpenSearch、自建 Filebeat 和 Dashboards 构建网络拨测功能

使用 Opensearch、Filebeat 和 Dashboards 集群搭建拨测功能与数据展示大盘，可以实时地、统一地、方便一键查看多个天翼云多可用区之间或地域之间的网络平均时延。这些数据可以为用户在搭建服务时选择更适合的地域或可用区。

应用场景

本文以 Opensearch、Filebeat 和 Dashboards 为例，搭建一个拨测功能及结果数据的展示大盘。使用 Filebeat 采集探测机器探测的网络数据，发送到 kafka 进行多个探测节点的数据汇聚以后存储到 Opensearch 中，最后通过 Dashboards 进行数据的可视化展示。该方案可以用于以下场景：

1. 客户部署业务的服务区选择：云下各地域访问阿里云地域的平均时延。您可以参考性能观测数据，在搭建服务时选择更适合的地域或可用区
2. 客户物理链路的选择：查看跨地域连接物理链路的时延情况，以便您选择更适合您业务的链路类型

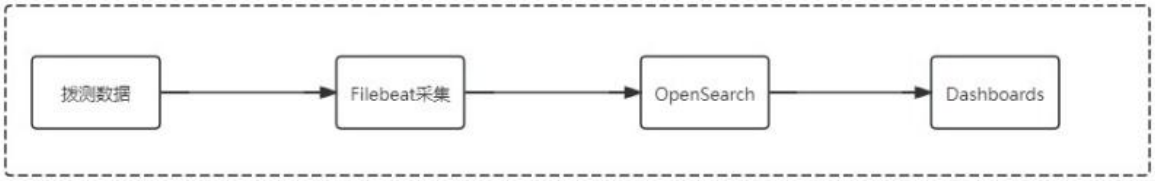
方案架构：

OpenSearch 是一款开源的分布式搜索和分析套件，衍生自 Elasticsearch OSS 7.10.2。可提供轻松执行交互式日志分析、实时应用程序监控、和数据分析等基础能力。

Filebeat 归属于 Beats 家族，使用 go 语言开发，是一个轻量的日志收集器，因为轻量所以适用于部署在需要收集日志的服务器中

Dashboards 为 OpenSearch 提供一个开源的数据分析和可视化平台，用于对

Elasticsearch 中的数据进行搜索、查看和交互。



方案优势：

- 实时性：提供实时数据收集和分析能力。
- 轻量级：对系统资源的消耗非常低。它设计用于高性能和低延迟，可以一键部署在虚拟机环境。
- 省时省力：无需额外开发数据展示界面，简化研发工作量。
- 可扩展：水平扩展能力强，可以处理 PB 级别的数据。

前提条件

1.已部署 OpenSearch 集群，操作步骤请参见部署 OpenSearch 集群。

2.已申请天翼云弹性云服务器 ECS，并安装了 Filebeat 环境，购买 ECS 请参见快速购买和使用 Linux ECS。

操作步骤

Step1.登录各数据采集节点的 ECS，部署并配置 Filebeat。

```
[root@wuhan41 filebeat]# ls
filebeat-8.12.2-linux-x86_64
[root@wuhan41 filebeat]# systemctl status filebeat
● filebeat.service - Filebeat is a lightweight shipper for metrics.
   Loaded: loaded (/etc/systemd/system/filebeat.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-08-12 11:59:52 CST; 1 months 18 days ago
     Docs: https://www.elastic.co/products/beats/filebeat
 Main PID: 30864 (filebeat)
   Memory: 70.8M
   CGroup: /system.slice/filebeat.service
           └─30864 /opt/filebeat/filebeat-8.12.2-linux-x86_64/filebeat -e -c /opt/filebeat/filebeat-8.12.2-linux-x86_64/filebeat.yml -path.hom...
```

step1.1：下载 filebeat-8.12.2-linux-x86_64

step1.2：解压压缩包到指定路径：

tar -xvzf /opt/filebeat/filebeat-8.12.2-linux-x86_64.tar.gz -C /opt/filebeat/

step1.3：配置 filebeat.yml

```
filebeat.inputs:
- type: filestream
  id: icmp-id
  enabled: true
  paths:
    - /opt/moose_ping/output/result/icmp_*.log
  fields:
    kafka_topic: "icmp-probe"
- type: filestream
  id: http-id
  enabled: true
  paths:
    - /opt/moose_ping/output/result/http_*.log
  fields:
    kafka_topic: "http-probe"
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
  reload.period: 10s
output.kafka:
  enabled: true
  hosts: ["kafka 机器 ip:kafka 端口"]
```

```
    codec.format:
      string: '%{[message]}'
    topic: "%{[fields.kafka_topic]}"
  processors:
    - add_host_metadata:
        when.not.contains.tags: forwarded
    - add_cloud_metadata: ~
    - add_docker_metadata: ~
    - add_kubernetes_metadata: ~
  logging.level: warning
```

step1.4：配置 systemd 服务

```
  # 创建 systemd 服务文件
  echo "Creating filebeat.service systemd service file..."
  cat <<\EOF | sudo tee /etc/systemd/system/filebeat.service
  [Unit]
  Description=Filebeat is a lightweight shipper for metrics.
  Documentation=https://www.elastic.co/products/beats/filebeat
  Wants=network-online.target
  After=network-online.target


  [Service]
  Environment="LOG_OPTS=-e"
  Environment="CONFIG_OPTS=-c
```

```
/opt/filebeat/filebeat-8.12.2-linux-x86_64/filebeat.yml"

Environment="PATH_OPTS=-path.home
/opt/filebeat/filebeat-8.12.2-linux-x86_64/filebeat -path.config
/opt/filebeat/filebeat-8.12.2-linux-x86_64 -path.data
/opt/filebeat/filebeat-8.12.2-linux-x86_64/data -path.logs
/opt/filebeat/filebeat-8.12.2-linux-x86_64/logs"

ExecStart=/opt/filebeat/filebeat-8.12.2-linux-x86_64/filebeat $LOG_OPTS
$CONFIG_OPTS $PATH_OPTS

Restart=always

[Install]

WantedBy=multi-user.target

EOF

echo "filebeat.service systemd service file created."

# 给予 systemd 服务文件可执行权限

sudo chmod +x /etc/systemd/system/filebeat.service

# 启用并启动 Filebeat 服务

echo "Enabling and starting Filebeat service..."

sudo systemctl daemon-reload

sudo systemctl enable filebeat

sudo systemctl start filebeat

echo "Filebeat service has been started."

# 输出 Filebeat 的进程状态

echo "Filebeat service status:"

sudo systemctl status filebeat | cat
```

Step2：登录数据归集节点的 ECS 并部署 filebeat（与步骤 2 部署方式相同）

数据归集节点 filebeat.yml

```
filebeat.inputs:
- type: kafka
  enabled: true
  hosts:
    - kafka 机器 ip:kafka 端口
  topics: ["icmp-probe"]
  group_id: "filebeat-icmp-probe-opensearch-test"
  worker: 6
  fields:
    type: "icmp"
- type: kafka
  enables: true
  hosts:
    - kafka 机器 ip:kafka 端口
  topics: ["http-probe"]
  group_id: "filebeat-http-probe-opensearch-test"
  fields:
    type: "http"
filebeat.config.modules:
  enabled: false
  path: /opt/filebeat/filebeat-8.12.2-linux-x86_64/modules.d/*.yml
```

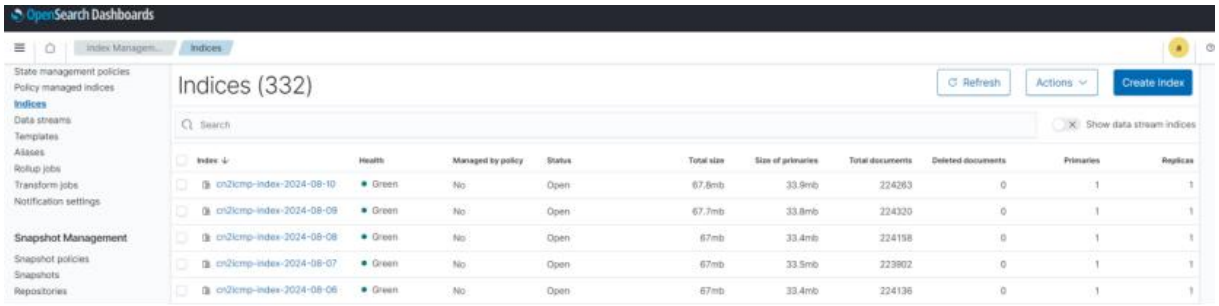
```
  reload.enabled: false
setup.template.settings:
  index.number_of_shards: 1
setup.kibana:
processors:
  - decode_json_fields:
      fields: ["message"]
      overwrite_keys: true
      target: ""
  - drop_fields:
      when:
        equals:
          fields.type: "icmp"
      fields:
["log","ecs","agent","host","input","kafka","Total","SourceIP","RemoteI
P","JobId","message","Rtts ms"]
      ignore_missing: true
  - drop_fields:
      when:
        equals:
          fields.type: "http"
      fields:
["log","ecs","agent","host","input","kafka","message","JobId","SourceIP
","HttpUrl"]
```

```
      ignore_missing: true
output.elasticsearch:
  enabled: true
  hosts: ["https://OpenSearch 机器 ip:OpenSearch 端口"]
  username: "OpenSearch 用户名"
  password: "OpenSearch 密码"
  ssl.verification_mode: none
  worker: 6
  indices:
    - index: "icmp-index-%{+yyyy-MM-dd}"
      when.contains:
        fields:
          type: "icmp"
    - index: "http-index-%{+yyyy-MM-dd}"
      when.contains:
        fields:
          type: "http"
logging.level: info
seccomp:
   default_action: allow
   syscalls:
   - action: allow
     names:
```

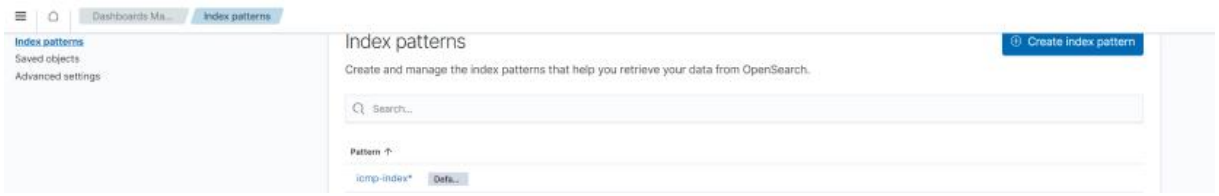
- rseq

Step3: 配置 OpenSearch

step3.1: 查看是否数据成功投递到了 OpenSearch 中

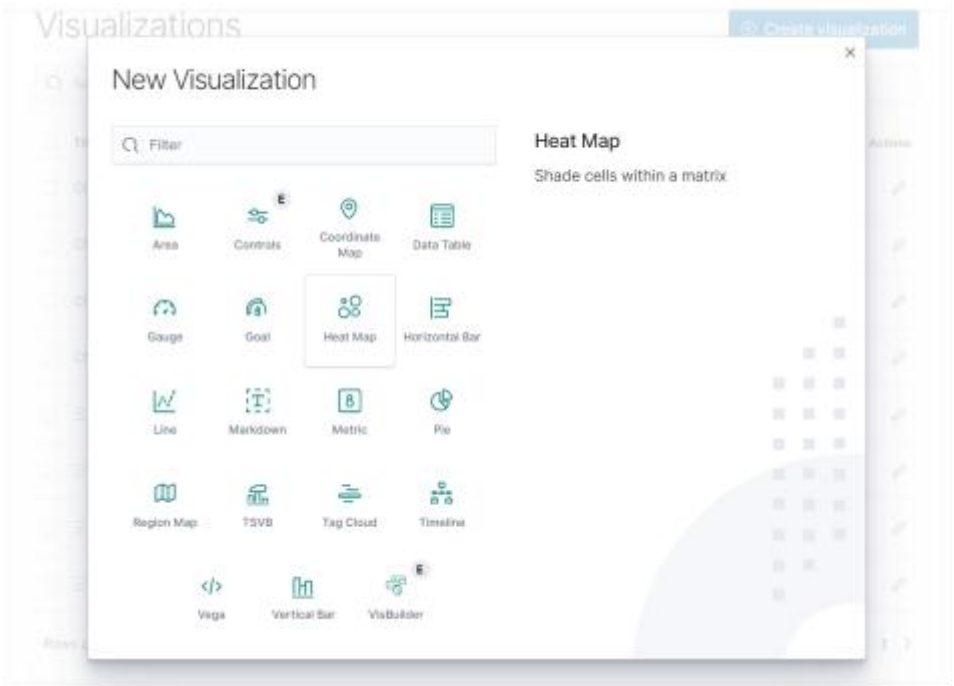


step3.2: 创建 Visualization 需要用到的 index-pattern

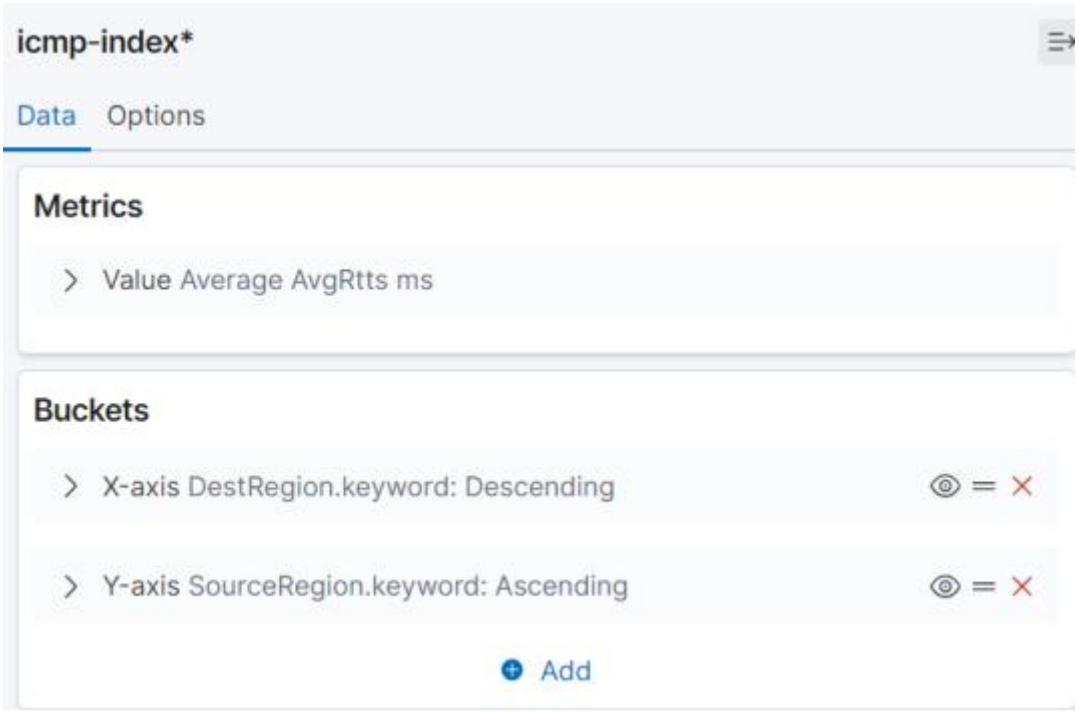


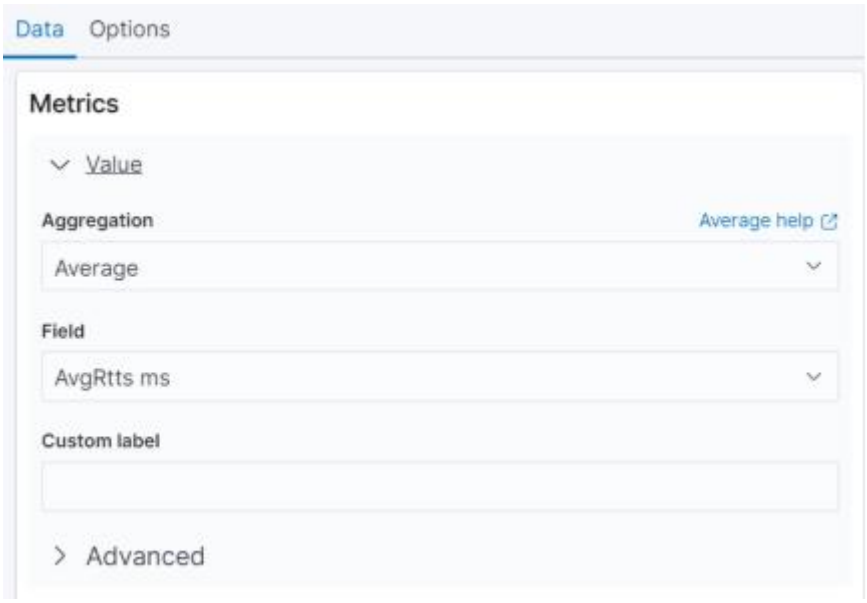
Step4：配置 Visualization

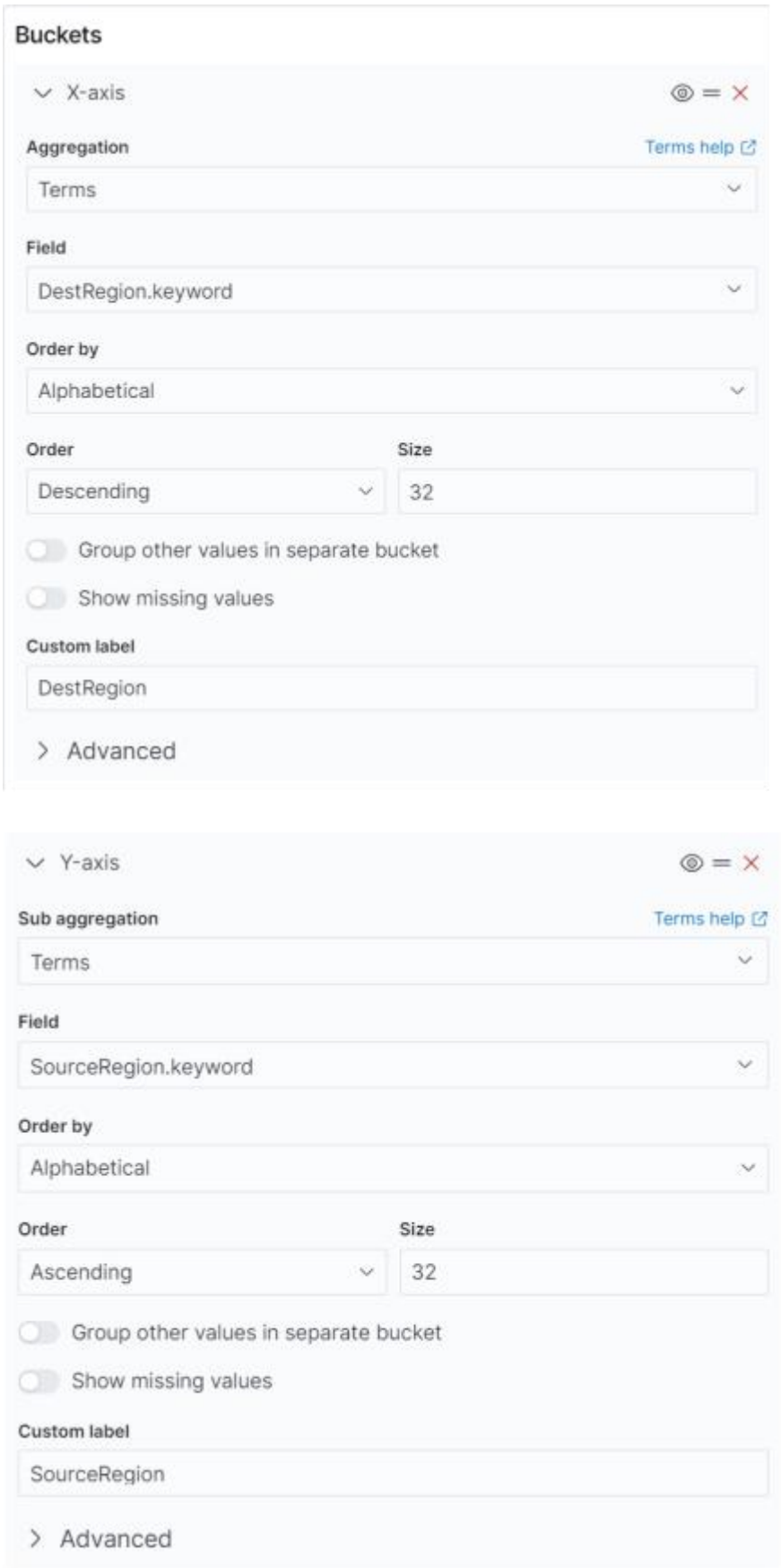
step4.1：创建 Visualization

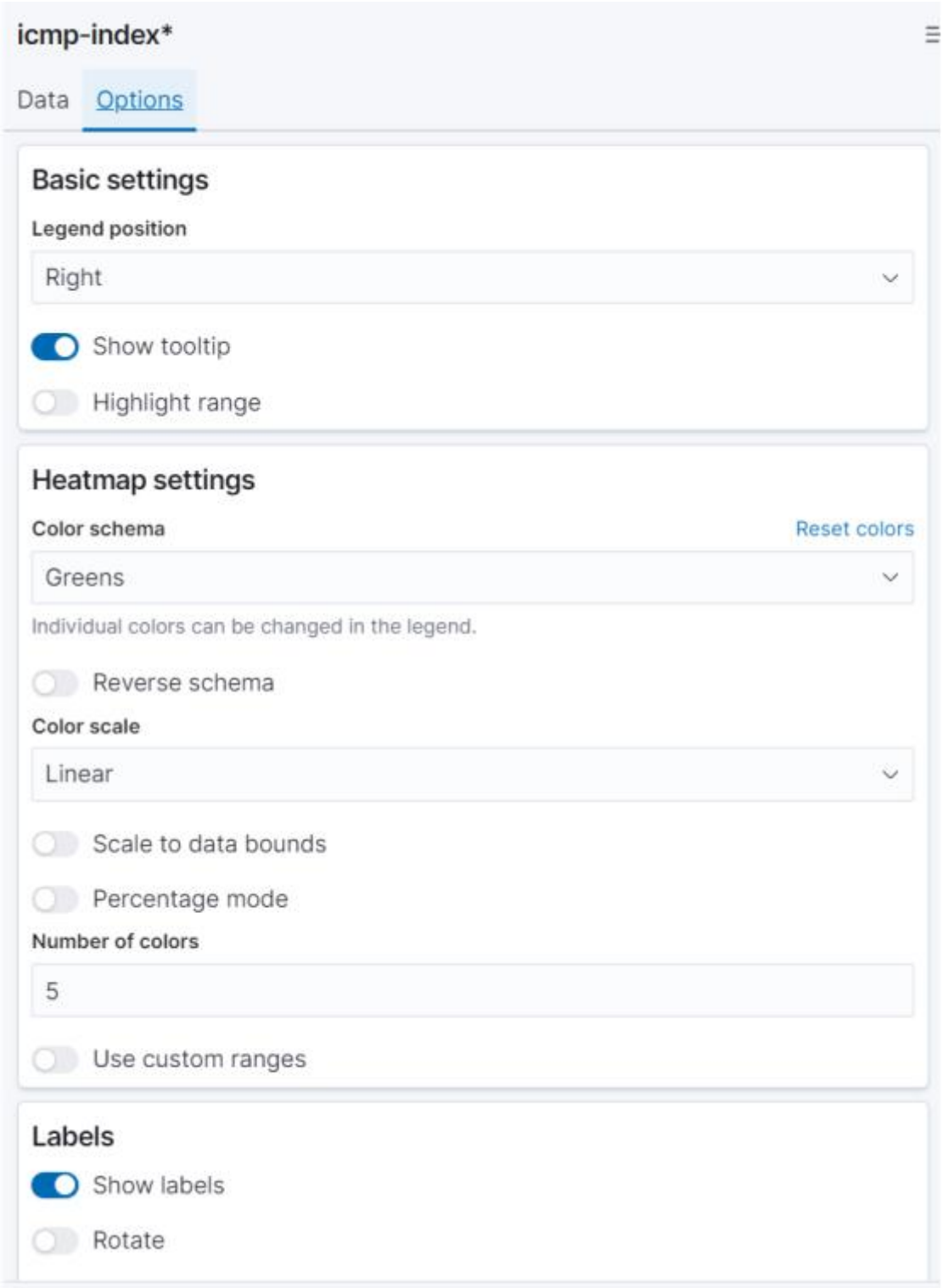


step4.2：配置横纵坐标


icmp-index*
Data
Options
Metrics
Value Average AvgRtts ms
Buckets
X-axis DestRegion.keyword: Descending
Y-axis SourceRegion.keyword: Ascending
Add


Data
Options
Metrics
Value
Aggregation
Average help
Average
Field
AvgRtts ms
Custom label
Advanced


Buckets
X-axis
Aggregation
Terms help
Terms
Field
DestRegion.keyword
Order by
Alphabetical
Order
Size
Descending
32
Group other values in separate bucket
Show missing values
Custom label
DestRegion
Advanced
Y-axis
Sub aggregation
Terms help
Terms
Field
SourceRegion.keyword
Order by
Alphabetical
Order
Size
Ascending
32
Group other values in separate bucket
Show missing values
Custom label
SourceRegion
Advanced



step4.3：展示效果图

使用 Elasticsearch、Kibana 实例以及自建 Logstash 搭建日志分析平台

使用 ELK Stack（Elasticsearch、Logstash、Kibana、Beats）进行日志管理是一种流行的开源解决方案，用于集中式日志收集、存储、分析和可视化。ELK Stack 可以从分布式系统中采集和聚合日志，帮助运维和开发人员更好地理解系统的运行状态、排查问题并监控关键业务指标。

本文将使用 Filebeat、Logstash、Elasticsearch、Kibana 搭建一个简单的日志分析平台，其中需要自行部署 Filebeat 和 Logstash 并且打通和天翼云云搜索 Elasticsearch、Kibana 实例之间的网络。

1.ELK Stack 组件。

Elasticsearch 是一个分布式搜索引擎，作为 ELK 堆栈的核心，它负责存储和索引日志数据。Elasticsearch 提供强大的全文搜索、过滤和分析功能，可以快速处理大规模数据并允许实时查询。

Logstash 是一个数据处理管道工具，负责从各种输入源收集数据，进行过滤、处理并将其输出到 Elasticsearch。Logstash 支持多种数据源（如文件、数据库、消息队列），并且能够通过过滤器对数据进行处理，比如解析、格式转换等。

Kibana 是一个数据可视化和分析工具，允许用户在浏览器中直观地查询和展示 Elasticsearch 中存储的日志数据。Kibana 提供多种图表、仪表盘和地图，可以帮助用户监控系统、分析日志、生成报表等。

Beats 是一组轻量级数据收集器，用于将各种类型的数据发送到 Elasticsearch 或 Logstash 进行索引和分析。其中 Filebeat 是 ELK Stack 中的一个轻量级日志数据收集器，用于收集日志文件数据。它监视指定的日志文件或位置，并将其发送到 Elasticsearch 或 Logstash 以进行存储和分析。

2. 工作机制。

首先使用轻量级的 Filebeat 采集日志、其次使用 Logstash 接收 Filebeat 的输出，并对日志进行解析、添加或删除字段等处理、最后将数据输出到天翼云 Elasticsearch 实例，通过 Kibana 实例在前端可视化展示。

3. 前提条件。

已经开通天翼云云搜索 Elasticsearch 和 Kibana 实例。

已经部署 Filebeat 和 Logstash 并且打通和天翼云云搜索实例之间的网络。（推荐使用 Filebeat 和 Logstash 7.10.2 版本）。

查看 Kibana 的终端可以访问到云搜索实例，设置好 5601 端口的网络安全策略。

4. 配置 Filebeat。

Filebeat 采集日志，配置具体的日志路径。

```
#采集日志
filebeat.inputs:
- type: log
  # 采集的日志文件的路径。替换为自己日志的路径，可以使用通配符。
  paths:
    - /your_path/*.log
output.logstash:
  hosts: ["{logstash_ip}:5044"]
```

5. 配置 Logstash。

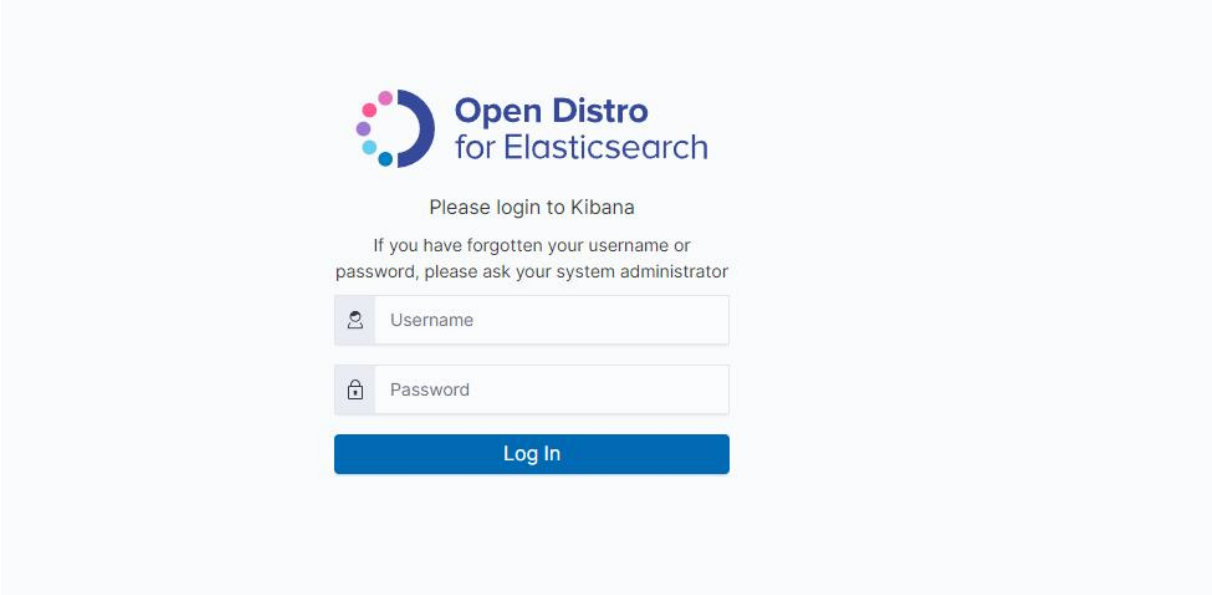
Logstash 需要接收 Filebeat 的输出并进行处理，示例配置如下：

```
input {
  beats {
```

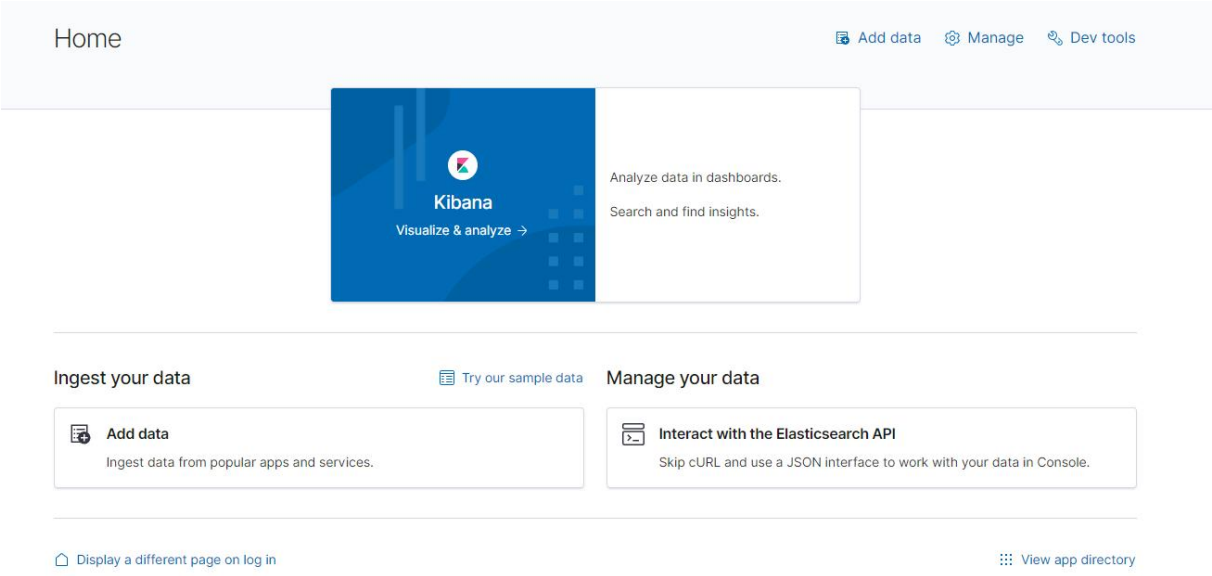
```
    port => 5044
  }
}
# 对数据进行处理。
filter {
    # mutate {
    #   remove_field => ["@version"]
    # }
}
output{
     elasticsearch{
       # Elasticsearch 实例的访问地址。
       hosts =>  ["http://{ip}:{port}", "http://{ip}:{port}",
"http://{ip}:{port}"]
       # 访问 Elasticsearch 实例的用户名和密码，如无安全机制可不配置。
       user => "********"
       password => "********"
       # 配置写入的索引名,示例如下。
       index => "filebeat-logstash-es-%{+YYYY.MM.dd}"
     }
 }
```

6. 使用 Kibana 进行可视化查询。

启动 Filebeat 和 Logstash 后，日志会不断被采集到 Elasticsearch 实例中。可以在浏览器中打开 Kibana 界面，http:{ip}:5601。



输入正确的用户名和密码，可以登录到系统中。



可以使用 Dev tools 进行查询。

```
GET {your_index}/_search
{
  "query": {
    "match": {
      "{your_filed}": "XXXXXX"
    }
```

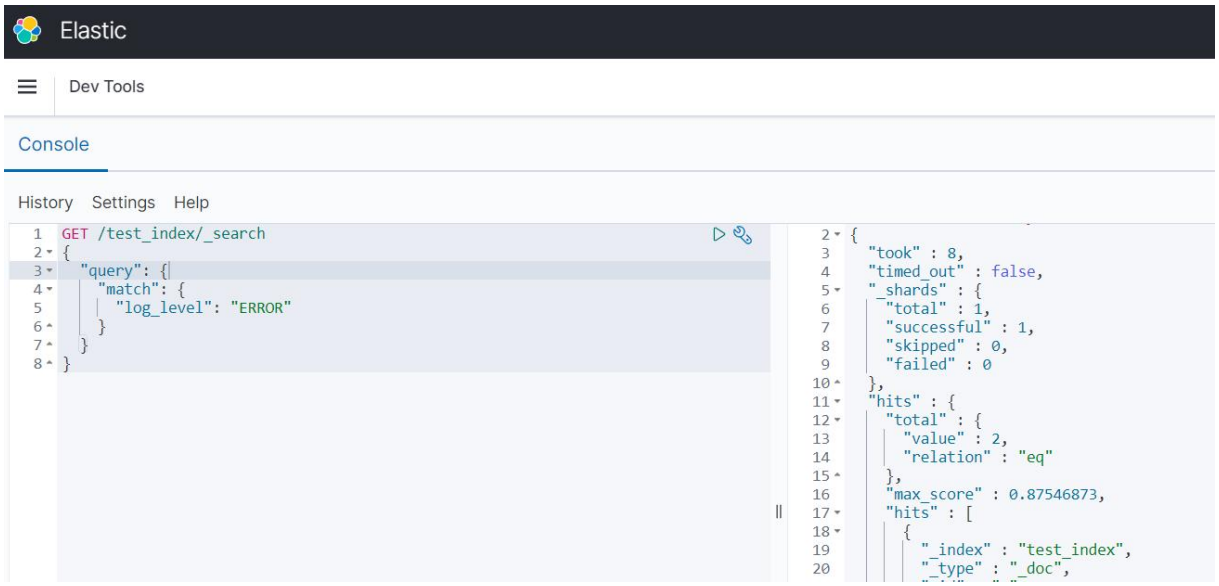
```
  }
}
```

例如有一个名 test_index 的索引，用来存储日志信息。其中有一个 log_level 字段，用来存储日志级别，例如 INFO、ERROR。

可以适用下面的命令来查询 ERROR 的数据。

```
GET /test_index/_search
{
  "query": {
    "match": {
      "log_level": "ERROR"
    }
  }
}
```

在 Kibana 中查询的效果如下。



OpenSearch 和 OpenSearch-Dashboards 也适用这套方案。在实际的使用过程中将 Elasticsearch 和 Kibana 替换为 OpenSearch 和 OpenSearch-Dashboards，同时修改 Logstash 配置输出到 OpenSearch 实例即可。